IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 2, 2019, accepted October 20, 2019, date of publication November 8, 2019, date of current version November 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2952465

Robust and Unified VLC Decoding System for Square Wave Quadrature Amplitude Modulation Using Deep Learning Approach

SYUKRON ABU ISHAQ ALFAROZI[1], (Member, IEEE), KITSUCHART PASUPA[1], (Senior Member, IEEE), HIROMICHI HASHIZUME[2], (Member, IEEE), KUNTPONG WORARATPANYA[1], (Member, IEEE), AND MASANORI SUGIMOTO[3], (Member, IEEE)

[1]Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
[2]National Institute of Informatics, Tokyo 101-8430, Japan
[3]Graduate School of Information Science and Technology, Hokkaido University, Sapporo 060-0814, Japan

Corresponding author: Kuntpong Woraratpanya (kuntpong@it.kmitl.ac.th)

This work was supported in part by the King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand, and in part by the ASEAN University Network/Southeast Asia Engineering Education Development Network (AUN/SEED-Net) through the Japan International Cooperation Agency (JICA), Japan.

ABSTRACT We have proposed a square wave quadrature amplitude modulation (SW-QAM) scheme for visible light communication (VLC) using an image sensor in our previous work. Here, we propose a robust and unified system by using a neural decoding method. This method offers essential SW-QAM decoding capabilities, such as LED localization, light interference elimination, and unknown parameter estimation, bundled into a single neural network model. This work makes use of a convolutional neural network (CNN) that has a capability in automatic learning of unknown parameters, especially when it deals with images as an input. The neural decoding method can provide good solutions for two difficult conditions that are not covered by our previous SW-QAM scheme: unfixed LED positions and multiple point spread functions (PSFs) of multiple LEDs. Responding to the above solutions, three recent CNN architectures—VGG, ResNet, and DenseNet—are modified to suit our scheme and other two small CNN architectures—VGG-like and MiniDenseNet—are proposed for low computing devices. Our experimental results show that the proposed neural decoding method performs better in terms of error rate than the theoretical decoding, an SW-QAM decoder with a *Wiener* filter, in different scenarios. Furthermore, we experiment on the problem of moving camera, i.e., the unfixed position of LED points. For this case, a spatial transformer network (STN) layer is added to the neural decoding method for solving the moving camera problem, and the method with the new layer achieves a remarkable result.

INDEX TERMS Visible light communication, image sensor communication (ISC), SW-QAM, optical camera communication (OCC), neural decoding, convolutional neural network (CNN), deep learning.

I. INTRODUCTION

The emergence of machine learning (ML), especially artificial neural network (ANN), in visual recognition, has become a hot topic due to the applicable training of ANN [1] on a powerful graphics processing unit (GPU) machine. In 2012, AlexNet proposed by Krizhevsky *et al.* [2] was a convolutional neural network (CNN) breakthrough for classifying large image datasets with superb performance. CNN architecture has several spatial convolutional layers that are able to learn automatically from data through the back-propagation algorithm. Since then, there have been various computer vision tasks that take advantage of CNN developments such as image classification [3], object detection [4], and tracking tasks [5].

In this paper, the great potential of CNN is also applied to visible light communication (VLC) research especially for the square wave quadrature amplitude modulation (SW-QAM) scheme [6]. In image-sensor-based VLC, an image sensor or a camera plays the role of a receiver that

The associate editor coordinating the review of this manuscript and approving it for publication was Mithun Mukherjee.

captures a sequence of LED images and then decodes the captured images back to a sequence of symbols as described in SW-QAM. The role of the image sensor can be viewed as a visual recognition task. Therefore, CNN can be a good solution for a fully automatic SW-QAM decoder. The SW-QAM decoder requires three important techniques to make it works properly: (i) detecting the center of the LEDs, (ii) removing the light interference among LEDs for multiple LED setting, and (iii) decoding the encoded symbols from the captured images. These difficulties and complexities can be eliminated by replacing the SW-QAM decoder with the neural decoder. In this paper, we focus on the use of multiple LEDs due to its high complexity. Absolutely, this scheme still works for a single LED.

Here, we propose a novel decoding method based on our SW-QAM scheme but using a neural network instead. This decoding method is called neural decoding. (Hereinafter, SW-QAM will be called a theoretical decoding.) By using neural decoding, we are not only able to reduce light interference and estimate other unknown parameters efficiently, but also able to decode the encoded symbols in a single stage. Furthermore, in a real world situation, the camera may be moving while capturing a frame that may cause the positions of LED points not to be fixed in the image area. With the modularity of the neural network model, the moving camera problem can be solved by adding a spatial transformer network (STN) [7] layer into the neural decoder.

The main contributions of our work are listed as follows.

- A neural decoding method that is robust to light interference is proposed when multiple LEDs are used.
- An STN layer embedded into the neural decoding method for solving a moving camera problem is introduced.
- The comparative performances among five neural decoding methods and the theoretical decoding method, an SW-QAM decoder with a *Wiener* filter, are demonstrated.

The rest of this paper is organized as follows. Section II briefly reviews recent studies on VLC technology and applications of neural networks in a communication research field. Section III explains the proposed method, neural decoding architecture and its implementation in our developed scheme. The experimental and comparative results of all decoding methods are discussed in Section IV. Finally, we conclude our work in Section V.

II. BRIEF REVIEW

Applications of deep learning to communication research have been proposed in various works, such as channel modeling, equalization, decoding, demodulation or prediction [8], [9]. For instances, O'Shea and Hoydis [10] introduced a channel modeling that provides an end-to-end learning framework for encoding and decoding processes in a physical layer. The whole communication system (from the transmitter to receiver) is modeled based on an autoencoder neural network architecture such that all the modulation and demodulation techniques are automatically optimized by training algorithms. Moreover, they also developed another architecture called radio transformer network (RTN) derived from STN to improve the performance of their autoencoder. The RTN defines a parametric transformation of a symbol at the receiver as a correction for the autoencoder's error so that the performance of the entire system can be improved. O'Shea's work demonstrates that deep learning is applicable to communication systems and that deep learning is robust to various unknown parameters such as channel imperfection and non-linearity [11].

Mohammad *et al.* [12] implemented a demodulation method based on a deep convolutional neural network (DCNN) and compared its performance to those of other machine learning and non-learning methods for demodulation of a Rayleigh-faded wireless data signal with several settings of signal-to-noise ratio (SNR). They showed that DCNN was able to achieve a lower bit error rate than other methods in all experimental scenarios. Therefore, an application of CNN to a demodulation method can provide a substantial benefit of decreasing the bit error probabilities.

Furthermore, in VLC research area, Haigh *et al.* [13] used a neural network not for a decoding or demodulation purpose but as an equalizer for the decoding method of an on-off keying (OOK) modulation scheme. Equalization mitigates the effect of inter-symbol interference (ISI) that degrades the performance of a system. Therefore, by using an equalizer, the authors were able to increase the data transmission rate. In their work, the authors used a neural network equalizer with 10 hidden neurons. The system was able to achieve a maximum communication speed of 170 Mbps under a targeted bit error rate (BER) of 10^{-6}. This was a successful case of solving an unknown parameter problem through data-driven learning.

Moreover, in an image-sensor based VLC for localization research, Ifthekhar *et al.* [14] applied a neural network (NN) localization method and used a stereo vision technique for vehicle positioning. They introduced two positioning methods: a cooperative-vehicle positioning (CVP) system using optical camera communication (OCC) method in complementary with a computer vision technique and an NN based technique. They used the light from LEDs at the rear of a car to communicate with another car installed with a stereo camera so that the latter car can determine the position of the former car. This work shows that the NN method outperformed the computer vision method with OCC based technique. Moreover, NN does not need complex mathematical modeling to build a model. NN only needs to learn from a set of training data then it can build a model automatically through a learning algorithm.

Lastly, an application of a neural network to VLC with an image sensor for decoding purposes was introduced in [15]. This method was applied to a vehicle-to-vehicle (V2V) communication scheme by using modulated rear LEDs of a car. They utilized two different neural networks as a complementary method to localize the LEDs in the captured images and

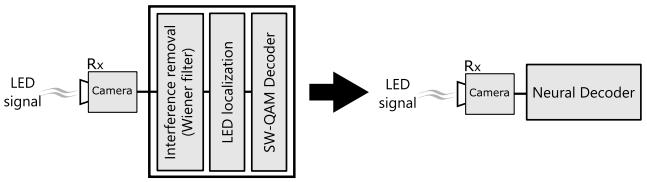


FIGURE 1. A unified system implemented by a replacement of the SW-QAM decoding scheme (on left-hand side) with the neural decoding scheme (on right-hand side).

to classify the transmitted symbols. They showed that the use of a neural network method yielded better performance than the computer vision technique that they tested. Hence, we can conclude that one of the main reasons for the achievement of NN applications is that the neural network was able to overcome unknown parameter problems such as non-linearity problem, noise estimation, and channel imperfection by automatic learning from real data.

As described in [6], SW-QAM decoder requires essential complementary techniques such as LED localization, light interference elimination, and noise removal. However, these techniques may provide a sub-optimal performance because the performance of the entire system will depend on the performance of each particular method. Hence, we propose a novel decoding scheme, as shown in Fig. 1, for SW-QAM by accommodating all of these methods into a single neural network model so that we have a unified system that can decode symbols efficiently.

III. PROPOSED METHOD

In this paper, we propose a novel neural decoding method based on SW-QAM scheme. The proposed method is fully decodable and independent from any additional techniques.

A. NEURAL DECODING

Neural network (NN) is an efficient machine that is able to be applied to solve a variety of problems such as prediction, recognition, and fitting tasks. Moreover, it is also powerful in estimating unknown parameters. Due to NN's capability in a variety of tasks, we can take the advantage of NN on light interference elimination, noise removal, and correct decoding in VLC with a camera receiver, especially when working with an SW-QAM modulation scheme [6]. The problem of using a camera as a receiver in VLC is no different from that in computer vision. Not surprisingly, some problems in VLC with a camera can be solved by a computer vision method.

According to the results of our SW-QAM scheme in [6], the theoretical encoding and decoding always perform well in a controllable situation. However, in practice, the existence of a large amount of noise and light interference becomes a serious problem in the decoding stage. Moreover, in terms of light interference elimination, different situations may have different point spread functions (PSFs), so we need to model the specific PSF for each. Responding to the above solution, we propose a neural decoding method that has great potential for solving such a problem. As illustrated

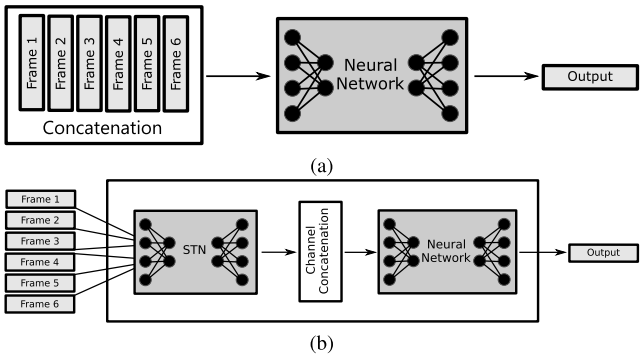


FIGURE 2. An implementation of two neural decoding architectures for (a) fixed and (b) unfixed LED positions.

in Fig. 1, it can be seen that our proposed scheme uses a single neural decoder instead of multiple complementary methods, each of which overcomes a specific problem such as light interference elimination and LED localization. Inside our proposed scheme, the neural decoding does not require the PSF modeling to reduce the light interference among LEDs as the *Wiener* deconvolution method does, as described in [6]. Also, the position of the LED points and the image features are automatically detected in the training stage of the neural network. In other words, all of the tasks of the SW-QAM decoder can be done by the neural network itself.

Fig. 2 shows two neural decoding architectures for fixed and unfixed LED positions in an implementation scheme. The fixed LED position means that the camera is not moved, so the LED position in the image is fixed, while the unfixed LED position means that the camera is moving, so the LED position is not fixed. For the fixed LED position model architectures as illustrated in Fig. 2(a), the input of the neural network consists of 6 frames, i.e., 3 frames from a preamble symbol and the other 3 frames from a data symbol. This setting is suitable for performing the decoding method since each symbol is related to the preamble information as explained in [6]. These 6 frames are concatenated and then fed to the neural decoder that has 6 channels for inputs. Each channel is for each frame which is a grayscale image. Then, the neural network processes these input images to obtain the outputs in terms of amplitude and phase of the modulated signal.

We encode each symbol in a 64-QAM constellation scheme, with the maximum value of 0.7 in the x and y axes, such that the first symbol is located at $(-0.7, 0.7)$ and the preamble symbol is located at $(-1, 0)$ as shown in Fig. 3. The outputs of the neural decoding are two regression values for the x-y coordinates of a decoded symbol in the 64-QAM constellation. Moreover, when we use n LEDs as a transmitter, the number of neural network outputs are n pairs of x-y coordinates. In our proposed scheme, the outputs of the neural network are four nodes, i.e., two pairs of x-y coordinates since we use two LEDs as the transmitter. This type of output is selected for our work so that it can be easy to visually compare them with those from

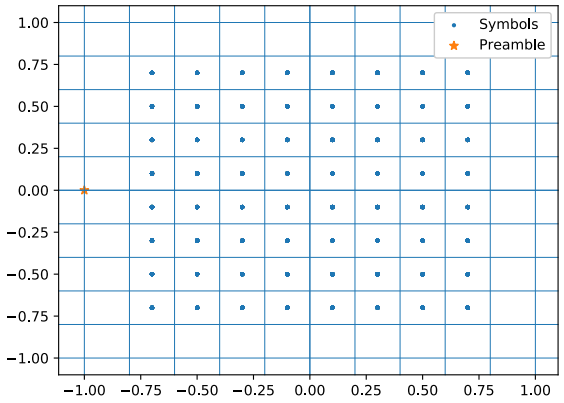


FIGURE 3. A 64-QAM constellation scheme for the encoded symbols and the preamble symbol at a phase $\theta = \pi$ and an amplitude $A = 1$.

the theoretical decoding method. However, in our preliminary experiment, we found that this regression output type yields better performance than a 64-symbol or 6-bit classification output type. Therefore, the regression loss function or mean squared error (MSE) is chosen for all of the neural decoding training phases to estimate the x-y coordinates in the 64-QAM constellation.

Another advantage of neural decoding is its use of an STN layer for the automatic transformation of input images. The STN layer can help overcome a moving camera problem that usually occurred in a real world situation. On the other hand, if we used the theoretical decoding, we would be required to use an essential technique for detecting the LED position in the image area. In our previous SW-QAM scheme, manual detection was used because the LED position was fixed. Furthermore, if we use a *Wiener* filter to improve the decoding function, the PSF might be changed due to the camera moving in 3-dimensional space. This is the reason why STN, one of the neural network layer types, is applied to overcome the problem of a moving object in the image, for the unfixed LED position. The STN is also trainable through a backpropagation algorithm. The input to the STN is a single frame, then 6 outputs from 6 input frames are concatenated to it and then fed into the next layer as illustrated in Fig. 2(b). In other words, it is the same as the architecture in Fig. 2(a) with an additional preceded STN layer to its input layer.

With NN's capabilities for accommodating several complementary methods, we can have a unified system for decoding by means of neural decoding. Then, its algorithmic procedure can be described as follows.

Given a sequence of LED images, containing the symbols that are encoded by SW-QAM encoder [6]:

Step 1. Concatenate three frames of a preamble symbol and three frames of a received symbol as the input of neural decoder.

Step 2. Predict the x-y coordinate in the QAM constellation of the symbol using the neural decoder.

Step 3. Classify/decode the symbol based on the grid location in the QAM constellation.

B. NEURAL ARCHITECTURES

We modify the recent three key neural network architectures, ResNet [16], DenseNet [17], and VGG [18] with batch normalization. Our modified neural network architectures are m-ResNet18, m-DenseNet121, and m-VGG16bn. Moreover, we also propose the other two small neural network architectures, VGG-like and MiniDenseNet, with a smaller number of layers and filters. Those three key architectures are modified so that their first convolutional layer can support 6-channel filters that match the 6 channels of the input. The last pooling layer is also adapted because the size of the spatial dimension of the input image is different from those of the original architectures. For example, our input image is 128×128 pixels while the original input image of these architectures is 224×224 pixels, thus, the last pooling layer or the fully connected layer needs to be adapted accordingly to the smaller size of input images. For the last layer, if the architecture has a fully connected (FC) layer, we use a single FC layer with 512 nodes and an output layer that has 4 output nodes for x-y regression of two LEDs.

The detailed configurations of CNN's architectures are shown in Table 1. Those architectures consist of three main blocks: initiation, building, and FC. In the initiation block, the first convolutional layer contains 64 filters with stride $s = 2$. In the building block, the first to the fourth blocks contain 64, 128, 256, and 512 filters, respectively, in each convolutional layer for all of the architectures, except for the VGG-like architecture of which the first to the fourth convolutional layers have 16, 16, 32, and 32 filters, respectively. Moreover, each architecture has different building blocks; the VGG, ResNet, and DenseNet use ordinary, residual, and dense blocks, respectively. For the FC block, the last pooling layer of the ResNet, DenseNet, and MiniDenseNet architectures is a global pooling layer that provides 1×1 spatial dimension output so that the FC block has an input of dimension C which is the number of channels of the last convolutional layer. Note that VGG architecture does not have an initiation block.

In this study, a VGG-like architecture is used with filter sizes of 5 and 7, while most of the modified CNNs that we work with have a filter size of 3. This increases the neural network capacity while still uses a small number of features and layers. However, the number of filters is still much smaller than those used by other architectures. In this way, the neural network size is much smaller than the original VGG (lower complexity). Please note that the convolutional and FC layers are followed by Leaky-ReLU [19] as an activation function.

Furthermore, for MiniDenseNet, we maintain the architecture of DenseNet but apply a smaller building block `block_config = (2,2,4,4)` and growth rate `k = 16` for it. This is the smallest model with the lowest complexity (see Table 2) among the others, especially when compared to the original DenseNet121 architecture of which building block and growth rate are `block_config = (6,12,24,16)` and `k = 64`, respectively.

TABLE 1. Neural decoding architectures.

Block	m-VGG16bn[a]	m-ResNet18[a]	m-DenseNet121[a] $k = 64$	VGG-like[b]	MiniDenseNet[a] $k = 16$
Init block		conv $7 \times 7, s : 2$	conv $7 \times 7, s : 2$ maxpool $3 \times 3, s : 2$		conv $7 \times 7, s : 2$ maxpool $3 \times 3, s : 2$
Block 1	[conv 3×3] $\times 2$ maxpool $2 \times 2, s : 2$	$\begin{bmatrix}\text{conv } 3 \times 3\\ \text{conv } 3 \times 3\end{bmatrix} \times 2$	$\begin{bmatrix}\text{conv } 1 \times 1\\ \text{conv } 3 \times 3\end{bmatrix} \times 6$ conv 1×1, /2 avgpool $2 \times 2, s : 2$	[conv 7×7]$\times 2$ avgpool $2 \times 2, s : 2$	$\begin{bmatrix}\text{conv } 1 \times 1\\ \text{conv } 3 \times 3\end{bmatrix} \times 2$ conv 1×1, /2 avgpool $2 \times 2, s : 2$
Block 2	[conv 3×3]$\times 2$ maxpool $2 \times 2, s : 2$	$\begin{bmatrix}\text{conv } 3 \times 3\ s : 2\\ \text{conv } 3 \times 3\end{bmatrix}$ $\begin{bmatrix}\text{conv } 3 \times 3\\ \text{conv } 3 \times 3\end{bmatrix}$	$\begin{bmatrix}\text{conv } 1 \times 1\\ \text{conv } 3 \times 3\end{bmatrix} \times 12$ conv 1×1, /2 avgpool $2 \times 2, s : 2$	[conv 5×5]$\times 2$ avgpool $2 \times 2, s : 2$	$\begin{bmatrix}\text{conv } 1 \times 1\\ \text{conv } 3 \times 3\end{bmatrix} \times 2$ conv 1×1, /2 avgpool $2 \times 2, s : 2$
Block 3	[conv 3×3]$\times 3$ maxpool $2 \times 2, s : 2$	$\begin{bmatrix}\text{conv } 3 \times 3\ s : 2\\ \text{conv } 3 \times 3\end{bmatrix}$ $\begin{bmatrix}\text{conv } 3 \times 3\\ \text{conv } 3 \times 3\end{bmatrix}$	$\begin{bmatrix}\text{conv } 1 \times 1\\ \text{conv } 3 \times 3\end{bmatrix} \times 24$ conv 1×1, /2 avgpool $2 \times 2, s : 2$	[conv 5×5]$\times 2$ avgpool $2 \times 2, s : 2$	$\begin{bmatrix}\text{conv } 1 \times 1\\ \text{conv } 3 \times 3\end{bmatrix} \times 4$ conv 1×1, /2 avgpool $2 \times 2, s : 2$
Block 4	[conv 3×3]$\times 3$ maxpool $2 \times 2, s : 2$ [conv 3×3]$\times 3$ maxpool $2 \times 2, s : 2$	$\begin{bmatrix}\text{conv } 3 \times 3\ s : 2\\ \text{conv } 3 \times 3\end{bmatrix}$ $\begin{bmatrix}\text{conv } 3 \times 3\\ \text{conv } 3 \times 3\end{bmatrix}$ avgpool 4×4	$\begin{bmatrix}\text{conv } 1 \times 1\\ \text{conv } 3 \times 3\end{bmatrix} \times 16$ avgpool 4×4	[conv 5×5]$\times 2$ avgpool $2 \times 2, s : 2$	$\begin{bmatrix}\text{conv } 1 \times 1\\ \text{conv } 3 \times 3\end{bmatrix} \times 4$ avgpool 4×4
FC block	FC 8192×512 Dropout $p : 0.5$ FC 512×4	FC $C \times 4$	FC $C \times 4$	FC 2048×512 FC 512×4	FC $C \times 4$

** /2 notation means that it reduces the number of channels by half, and C is the number of last channels in the last convolutional layer.

[a] The number of filters in each convolutional layer from blocks 1 to 4 is 64, 128, 256, and 512, respectively. Each convolutional layer is followed by a batch normalization layer and a ReLU activation function layer. The FC layer is followed by a ReLU function without batch normalization.

[b] The number of filters in each convolutional layer from blocks 1 to 4 is 16, 16, 32, and 32, respectively. Each convolutional layer and FC layer are followed by a Leaky-ReLU activation function layer without a batch normalization layer.

TABLE 2. Comparative performances in terms of error rate on both lighting conditions for a fixed LED position dataset.

Method	Dark		Bright		#FLOPS
	$Err(\%)$	$rmse$	$Err(\%)$	$rmse$	(G)
Theoretical Decoding					
SW-QAM	**0**	0.0389	2.30	0.0442	**0.0041**
Neural Decoding					
m-VGG16bn	1.04	0.0330	11.80	0.0508	5.0633
m-ResNet18	1.49	0.0246	*1.33*	*0.0288*	0.6334
m-DenseNet121	*0.02*	**0.0038**	3.29	**0.0239**	0.9795
VGG-like	2.79	0.0375	6.13	0.0458	0.3892
MiniDenseNet	**0**	*0.0227*	**1.06**	0.0346	*0.1248*

Bold numbers indicate the best results, while *Italic* numbers indicate the second best results.

In the case of unfixed LED positions, an STN layer is required for the automatic transformation of input images. It consists of grid sampling that produces the same resolution of 128 × 128 pixels and a very simple localization network, `Sequential(conv11 x4 s:4, conv7 x8 s:4, conv3 x16 s:1, FC128, FC6)` with a ReLU function. The number of STN output parameters is 6 nodes that can perform the linear transformation. With this configuration, the STN is able to localize LED points in the image.

In conclusion of this section, our five modified architectures can be viewed in two perspectives: (i) architectures with deep and deeper layers and (ii) architectures with and without skip connections. If we focus on the number of layers as a key point, m-VGG16bn, VGG-like, m-ResNet18, and MiniDenseNet, are the architectures with deep layers, while m-DenseNet121 is with deeper layers. On the other hand, if we focus on the skip connection as a key point, m-ResNet18, m-DenseNet121, and MiniDenseNet are the architectures with skip connections, while m-VGG16bn and VGG-like are without skip connections.

IV. EXPERIMENTS AND DISCUSSION

In this section, we compare the performance of our previous theoretical decoding, an SW-QAM decoder with a *Wiener* filter, and five modified architectures of neural decoding under the same settings of experimental setup as those described in SW-QAM [6].

A. DATA COLLECTION AND PREPARATION

We collected more experimental data than those contained in the dataset for SW-QAM [6]. Here, the additional datasets consist of 15 training and 5 validation sets. For the testing sets, we used the same datasets as those reported in [6] (5 testing sets). Hence, we had 25 different experimental datasets in total for each situation where each dataset contained 2000 frames. In addition, we also tested the five modified architectures of neural decoding in different scenarios: fixed camera position, unfixed (unstable) camera position,

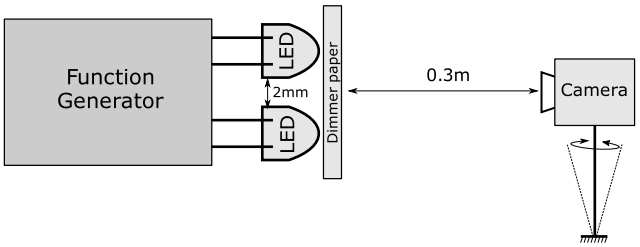


FIGURE 4. The camera is randomly moving around by a monopod standing on the ground for collecting the unfixed position datasets. Note that the experimental equipment was set the same as [6].

dark lighting condition, and bright lighting condition. Thus, there were 100 datasets with four scenarios. For the unfixed position datasets, the camera was randomly moved in such a way that the two LEDs are still in the frame, as depicted in Fig. 4. Thus, the orientation of the camera and the distance between the camera and LED will be changed accordingly.

With the same setting of the experimental scheme of multiple LEDs for comparison as reported in [6], at the transmitter side, the symbol frequency f_s was sent at 20 Hz, generated from a function generator. At the receiver side, the global shutter camera was placed 30 cm far from the LEDs. The camera configuration was set with a frame rate of $f_c = 60$ Hz, an exposure time ratio of $\eta = 0.5$ and an aperture of $f/6$. For SW-QAM decoder setting, the *snr* parameter of *Wiener* deconvolution was set to a constant value of 0.02, the same value as set in [6]. For the PSF, we estimated using a single predefined LED image used for all frames. Hereafter, the SW-QAM decoder refers to SW-QAM with *Wiener* filter for theoretical decoding. Furthermore, the preamble detection was applied to all datasets for detecting the preamble frames as defined in [6].

In the algorithm configuration, we used PyTorch [20] and Numpy [21] libraries for implementing CNNs based on our SW-QAM scheme. All CNNs were trained with our datasets and their minibatch and epoch were set to 32 and 200, respectively. We also used Adam algorithm [22] with a small initial learning rate of 10^{-4}, due to its fast convergence time and good performance in our pre-experimental trials. We saved the best model that has the lowest error rate based on the evaluation of the validation dataset among 200 epochs and then test it on the testing dataset. Note that the validation and testing datasets were excluded from the training scheme of updating model parameters.

B. PERFORMANCE EVALUATION

In this subsection, we evaluated the performance of the proposed scheme on two datasets: (i) a fixed LED position dataset that the camera for image acquisition is stable and (ii) an unfixed LED position dataset that the camera is unstable or moving around while keeping the LED points inside the frame. For the fixed LED position dataset, both the theoretical and neural decoding methods were evaluated. On the other hand, for the unfixed LED position dataset, only the neural decoding methods were evaluated. The performance metrics used for evaluation are error rate for decoded accuracy, *rmse* for precision, and FLOPS for model complexity. Further, we use a constellation diagram to visualize the classification of decoded symbols. Here, we plotted the x-y coordinate of two LEDs in the 64-QAM constellation for all scenarios to visualize the performance of each method. A grid center of constellation represents a reference point. A distance between the correct decoded symbol and the reference point represents precision measured in terms of *rmse*. Most importantly, the robustness of the neural decoders is evaluated by different environments such as dark- and bright-lighting conditions and by different conditions such as moving and non-moving cameras. The achievement of robustness can be reflected through our used evaluation metrics, i.e., error rate and *rmse*.

1) FIXED LED POSITION DATASET

The performance of all test methods was evaluated with the fixed LED position dataset on two lighting conditions, i.e., dark and bright. The comparative results in terms of error rate, *rmse*, and the estimated number of floating operations (FLOPS) are shown in Table 2.

For the dark lighting condition, SW-QAM and MiniDense-Net were the best performed methods without any misclassified symbol, error rate of 0%. m-DenseNet121 was the second best with a very small error of 0.02%. The other three neural decoding methods were comparable; m-VGG16bn and m-ResNet18 result error rates of 1.04% and 1.49%, respectively, while VGG-like was the worst one with a little high error rate of 2.79%. In terms of *rmse*, m-DenseNet121 was the best performed method with *rmse* of 0.0038 that is around 10 times smaller than other methods. Here, the value of *rmse* tells us how the average spread distance of decoded symbols is far from the reference point. A lower *rmse* means the less spread of decoded symbols away from the reference point. Clearly, m-DenseNet121 had a very small deviation as shown in Fig. 5(d). Generally, a lower *rmse* leads to a lower error rate, but it is not always true. The accuracy of the model does not always depend on the *rmse* value, but the precision does. For instance, m-DenseNet121 yielded more precision, a very small value of *rmse*, than SW-QAM and MiniDenseNet, but it provided a bit error, a misclassified symbol. This happened when we look at Fig. 5(d), there was a misclassified symbol, denoted by an orange cross, at the grid of (0.7, 0.7). In this case, m-DenseNet121 was sacrificing one symbol while keeping the *rmse* sufficiently low. In summary, Fig. 5 shows how the decoded symbols spread around the reference points. If we look at the spread of symbols, m-ResNet18 was more precise than m-VGG16bn, but its accuracy was a little lower.

For the bright lighting condition, MiniDenseNet and m-ResNet18 were the first and second best performed methods with error rates of 1.06% and 1.33%, respectively. On the other hand, m-VGG16bn and VGG-like were the two worst performed methods. For the theoretical decoding, SW-QAM was the third best. Its accuracy for the bright

lighting condition was not as good as that for the dark lighting condition. This happened because of the sub-optimal complementary methods, i.e., *Wiener* deconvolution and LED localization, in estimating the frame intensity. For the *Wiener* deconvolution, PSF modeling is needed; therefore, when the non-optimal PSF is used, it usually affects the overall performance of decoding. For the LED localization, the LED position can be manually localized, because it was fixed at a certain point. However, if the camera is moving (it can say that the LED is not fixed), we need a computer vision technique for localization. For this reason, the performance of SW-QAM depends on those of the complementary methods.

To make SW-QAM optimal, we need to optimize all of the complementary methods. In terms of *rmse*, m-DenseNet121 and m-ResNet18 were the first and second best methods with spread values of 0.0239 and 0.0288, respectively. Here, MiniDenseNet was the third one. Fig. 6 shows the decoded symbol spread which relates to the *rmse* value for all test methods. Evidently, m-DenseNet121 provided a good precision, although its accuracy was less than those of MiniDenseNet and m-ResNet18.

The use of a unified system containing only a single neural network model is one of the supporting reasons why the neural decoding outperforms the theoretical decoding. In addition, the unified system helps us easy to optimize a single model instead of multiple components of the SW-QAM decoder.

Furthermore, a figure of 64-QAM constellation can visually reflect the effectiveness of (i) architectures with deep and deeper layers and (ii) architectures with and without skip connections. The figure shows the performance in terms of precision in decoding the encoded symbols. When look into Figs. 5 and 6, the deeper layers of the models, m-DenseNet121 and MiniDenseNet, provide better precision, compared to m-VGG16bn and VGG-like. On the other hand, when we consider two different architectures, with and without skip connections, m-VGG16bn (without a skip connection) and m-ResNet18 (with skip connections) are two different architectures but not so much difference in the number of layers. The architecture with skip connections, m-ResNet18, gives more precision than the one without a skip connection, like m-VGG16bn. This is because the skip connection, a bypass connection from a particular layer to the deeper layer [23], helps the optimization algorithm (gradient descent) to converge easily, due to the large gradient flow. Thus, the model generally has excellent generalization performance. In addition, the more layer (deeper) architecture with skip connections, m-DenseNet121, provides the less spread of decoded symbols as shown in Fig. 5(d) and Fig. 6(d). However, MiniDenseNet, a small version of m-DenseNet121, resulted a bit better than m-ResNet18 in dark lighting condition, while m-ResNet18 resulted a bit better than MiniDenseNet in bright lighting condition. Indeed, m-ResNet18 and MiniDenseNet are with skip connections, but their inside architectures are different in building blocks; m-ResNet18 uses basic residual blocks, while MiniDenseNet uses bottleneck dense blocks. The bottleneck blocks reduce the number of parameters using 1×1 convolution technique, thus leading to complexity reduction, as shown in Table 2. Therefore, the architecture that uses bottleneck block can go deeper with less complexity than that with basic block. In this case, in terms of complexity, m-ResNet18 is higher than MiniDenseNet. In terms of the number of layers, m-ResNet18 is less than MiniDenseNet (29 layers). Generally, the deeper architecture has better feature representations that lead to the better performance [24].

TABLE 3. Comparative performances in terms of error rate on both lighting conditions for an unfixed LED position dataset.

Method	Dark		Bright		#FLOPS
	$Err(\%)$	$rmse$	$Err(\%)$	$rmse$	(G)
Neural Decoding					
m-VGG16bn	3.25	0.0411	8.90	0.0549	5.0682
m-ResNet18	**0.52**	0.0229	4.13	0.0436	0.6382
m-DenseNet121	0.95	**0.0134**	*3.16*	**0.0340**	0.9843
VGG-like	6.56	0.0449	14.09	0.0642	*0.3941*
MiniDenseNet	*0.90*	*0.0288*	**2.73**	*0.0376*	**0.1296**

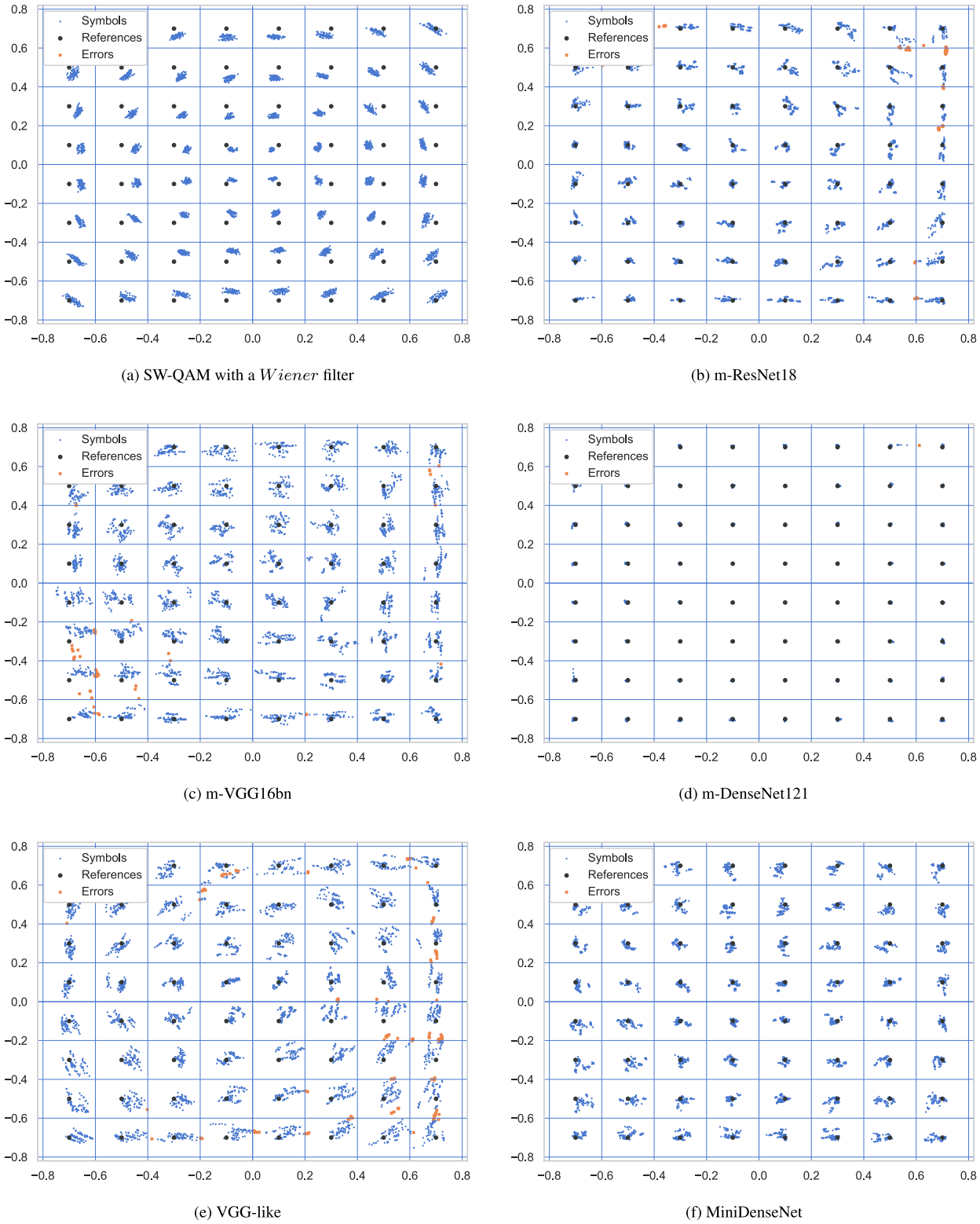
Bold numbers indicate the best results, while *Italic* numbers indicate the second best results.

2) UNFIXED LED POSITION DATASET

In this experiment, the theoretical decoding, SW-QAM, was excluded from the performance comparison. The main reason was that the position of the LED center and the PSF of these datasets were different from the fixed LED position dataset because the camera was not stable. Hence, SW-QAM needs complementary methods such as localization, object detection, and PSF modeling. These computer vision methods are beyond the scope of our discussion.

Responding to automatic image transformation, an STN layer is added in front of the main decoder as shown in Fig. 2(b) to appropriately transform the captured images. In our implementation, we use an STN layer that provides 6 output parameters. These 6 parameters play an essential role in performing linear transformation such as rotation, translation, dilation, cropping, and skewing of images. To demonstrate the capability of our neural decoder in solving the moving camera/object problem, we trained neural decoding with an STN layer such that the STN and the main decoder are initialized to the identity transformation and the pretrained CNN from the fixed LED position dataset, respectively, as schematically shown in Fig. 2(a). In this experiment, five modified CNN architectures in conjunction with an STN layer: m-VGG16bn, m-ResNet18, m-DenseNet121, VGG-like, and MiniDenseNet were tested with the unfixed LED position dataset. We implemented the main neural decoder with transfer learning by the use of the pretrained model from the fixed position dataset models, and then we trained the whole networks. The test results are shown in Table 3.

In the case of dark lighting condition, m-ResNet18 was the best and followed by MiniDenseNet with error rates



(a) SW-QAM with a $Wiener$ filter

(b) m-ResNet18

(c) m-VGG16bn

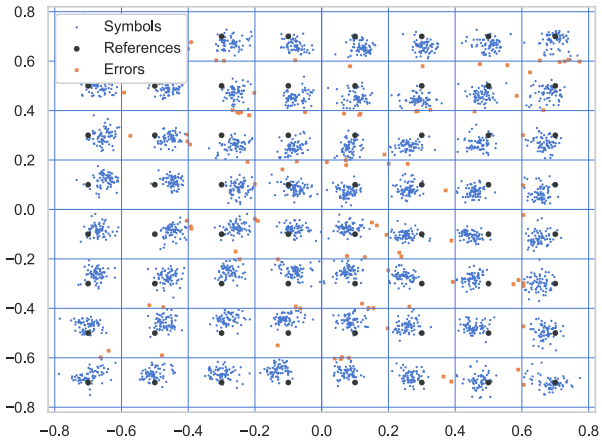
(d) m-DenseNet121

(e) VGG-like

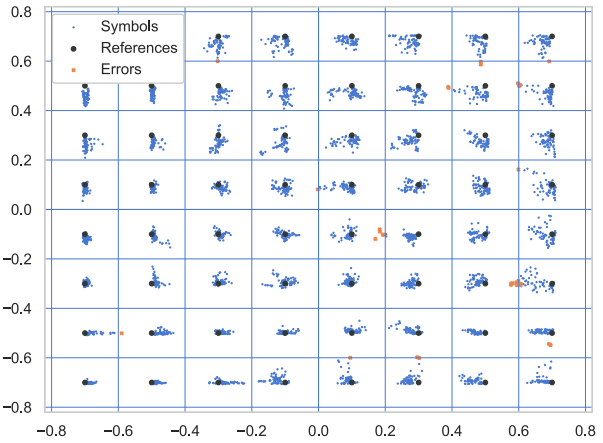
(f) MiniDenseNet

FIGURE 5. Comparative performances of the theoretical and neural decoding methods in dark lighting condition: both methods are tested with the fixed LED position dataset and the test results are plotted in a 64-QAM constellation.

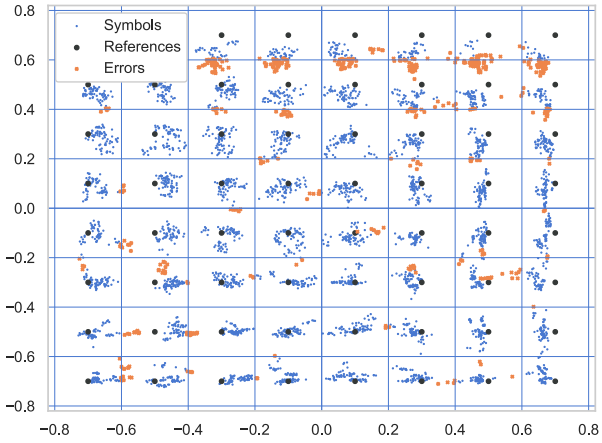
of 0.52% and 0.90%, respectively. From Fig. 7, it can be seen that m-ResNet18 and MiniDenseNet showed satisfactory constellation plots, although m-DenseNet121 was more excellent because it had the lowest *rmse* value of 0.0134. For bright lighting condition, MiniDenseNet was the best performed method with an error of 2.73% and



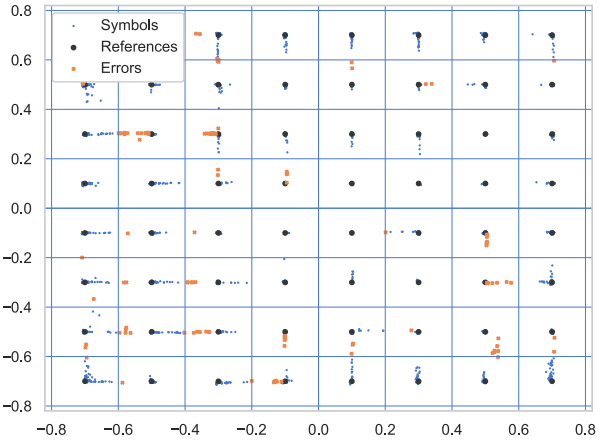
(a) SW-QAM with a $Wiener$ filter



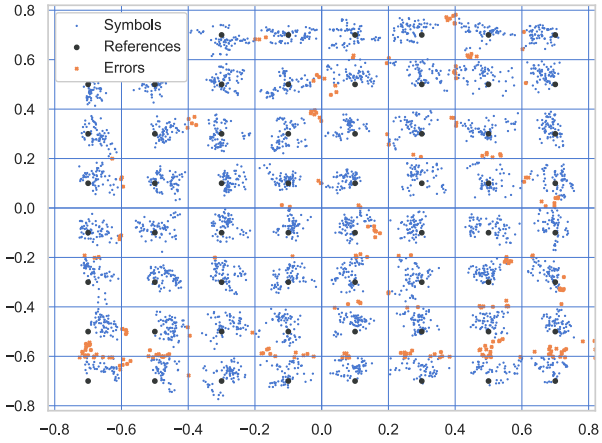
(b) m-ResNet18



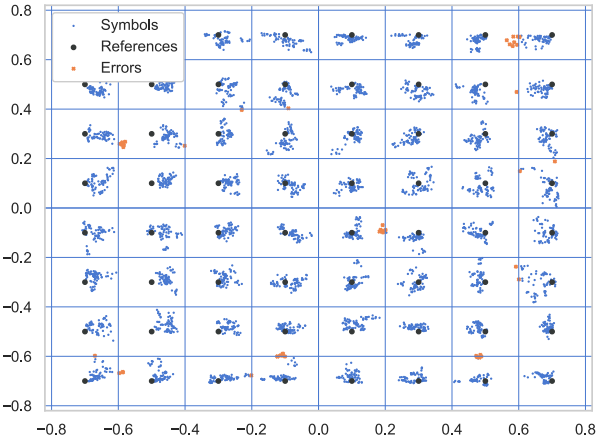
(c) m-VGG16bn



(d) m-DenseNet121



(e) VGG-like



(f) MiniDenseNet

FIGURE 6. **Comparative performances of the theoretical and neural decoding methods in bright lighting condition: both methods are tested with the fixed LED position dataset and the test results are plotted in a 64-QAM constellation.**

followed by m-DenseNet121 with an error of 3.16%. Again, m-DenseNet121 showed the less spread points with *rmse* value of 0.0340, as depicted in Fig. 8. However, m-VGG16bn and VGG-like performed defectively in most cases either for error rate or *rmse* as shown in Table 3 and Figs. 7 and 8. In addition, it was the same as in the fixed LED position dataset;

the architectures with skip connections, m-DenseNet121, m-ResNet18, and MiniDenseNet, performed better in terms of precision (spread) as can be seen in the decoded symbol constellation results in Figs. 7 and 8. m-DenseNet121 was still the most precise model in all datasets as depicted in Figs. 5, 6, 7, and 8, but it had a trade-off between precision and model complexity (FLOPS).

C. A SINGLE MODEL FOR ALL SITUATIONS

From the performance evaluation in four different situation datasets, i.e., fixed LED dark-, fixed LED bright-, unfixed LED dark-, unfixed LED bright-lighting datasets, the MiniDenseNet was the most promising model with high accuracy and the lowest complexity. In this case, we had four separately different models for each situation. Here, we trained a single model that can accommodate all situations using MiniDenseNet architecture. We used a MiniDenseNet with an STN layer because the datasets included the unfixed LED images.

The trained model from the unfixed dark-lighting dataset was used and retrained on all four training datasets. In this case, we used 60 (4×15) datasets for training, 20 (4×5) datasets for evaluation, and 20 (4×5) datasets for testing. This model yielded a low error rate of 0.05% and *rmse* of 0.0255, as shown in Table 4. Surprisingly, the error rate and *rmse* of this MiniDenseNet model were lower than those of MiniDenseNet models that were trained on each specific dataset—see Tables 2, 3, and 4. Fig. 9 shows the comparison of this MiniDenseNet and those MiniDenseNet models trained on each specific dataset. Clearly, the MiniDenseNet model trained on all datasets performed better than those trained on each specific dataset. In other words, this model is able to accommodate different types of PSFs with excellent generalization performance using only a single model. In terms of model complexity, it only consumes 0.13 GFLOPS, the same as the MiniDenseNet model in the unfixed situation.

In the best practice of neural network training, increasing training sets will make better a generalization and avoid an overfitting problem [25]–[27]. In our case, the model learns a rich variant of data, i.e., we trained the model on a larger dataset so that the model can learn a better understanding of the data variant. With this strategy, the model can have a better generalization than those trained on each specific dataset.

Furthermore, we applied a data augmentation technique to the training scheme to increase image data diversity using random rotation and flipping transformations [28], [29]. We used random rotation with a probability of 0.75 in the range of -10 to 10 degrees. For the random flipping transformation, we used a probability of 0.5 either for horizontal or vertical flips. However, the horizontal flip affected the position of the LEDs, in which the left LED moved to the right and the right LED moved to the left. Thus, we also need to flip the output of the data. Then, we trained the previously trained MiniDenseNet with this augmentation setting, using one cycle policy [30] for 2 × 20 epochs (20 epochs for one cycle) from FastAI library [31]. As expected, the performance of this model performed better, compared to the model trained without using the data augmentation technique as shown in Table 4 and Fig. 9.

TABLE 4. Performances of MiniDenseNet trained on all datasets with and without augmentation at each situation.

Dataset	Without augmentation		With augmentation	
	$Err(\%)$	$rmse$	$Err(\%)$	$rmse$
Fixed dark	**0**	0.0168	**0**	**0.0149**
Fixed bright	0.25	0.0247	**0.23**	**0.0208**
Unfixed dark	0.34	0.0234	**0.23**	**0.0213**
Unfixed bright	1.42	0.0340	**1.04**	**0.0304**
All datasets	0.50	0.0255	**0.34**	**0.0225**

As discussed so far in this section, it can be concluded that the neural decoder is robust to the light interference, since it can accommodate different types of PSFs in different situations of the unstable position of LEDs in the image.

D. DISCUSSION

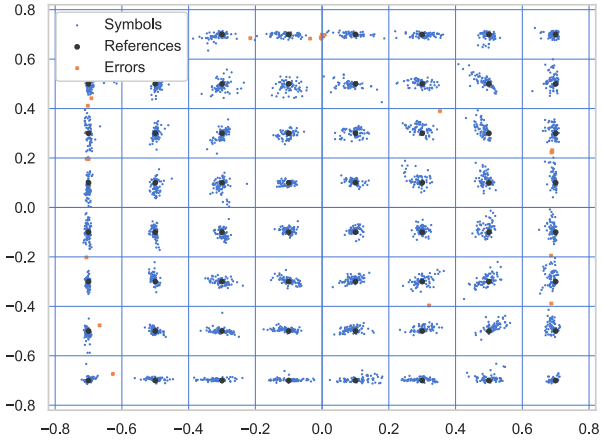
Here, several aspects of the neural and theoretical decoding, i.e., model complexities, model architectures, unknown parameters, and building blocks, are discussed.

1) MODEL COMPLEXITY

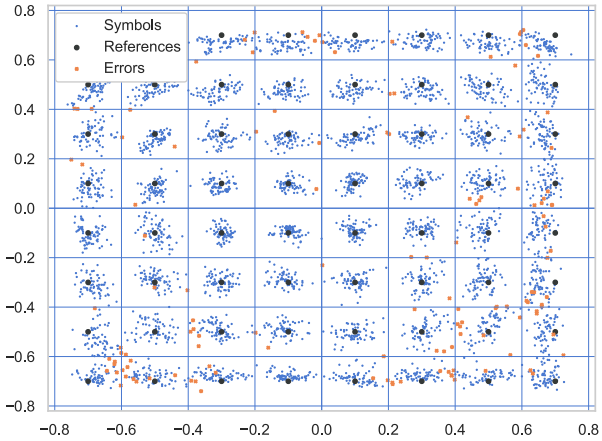
We refer to the estimated number of FLOPS as the model complexity. The last column of Tables 2 and 3 shows the number of FLOPS required for each model. The complexity can reflect the inference time. The less the model complexity, the less the inference time.

For the theoretical decoding, an SW-QAM with a *Wiener*, the most computational cost lies in Fast Fourier Transform (FFT) operation that requires around $N \log_2 N$ FLOPS for each transformation. As *Wiener* deconvolution needs three transformations for an input image, a PSF image, and an inverse transformation, so we have $3N \log_2 N$ FLOPS. For a single decoding step, we require 6 images, so the total complexity is $6 \times 3N \log_2 N$. In our experiment, N is equal to the number of pixels of an image, i.e., $N = 128 \times 128$, which leads to the complexity of 0.0041 GigaFLOPS, regardless of the localization method.

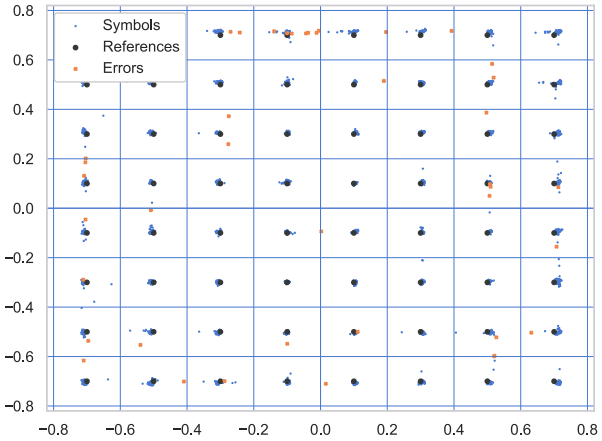
For models of neural decoding, their complexities were much higher than that of theoretical decoding. These complexities were part of profiling library for a PyTorch model, i.e., torchscope. However, for the neural decoding, we have a unified system instead of a modular system. Thus, we only need to optimize a single model. Here, the lowest complexity among the neural decoding model is MiniDenseNet whose complexity is around 100 times of SW-QAM. For the unfixed LED position, the complexity is slightly different from the fixed one, because the STN architecture is a small model as explained in Section III-B.



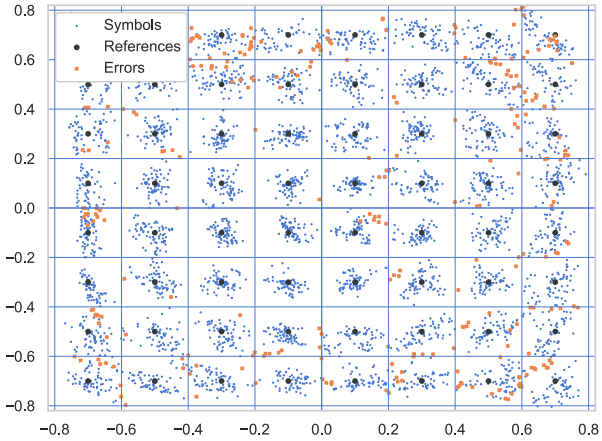
(a) m-ResNet18



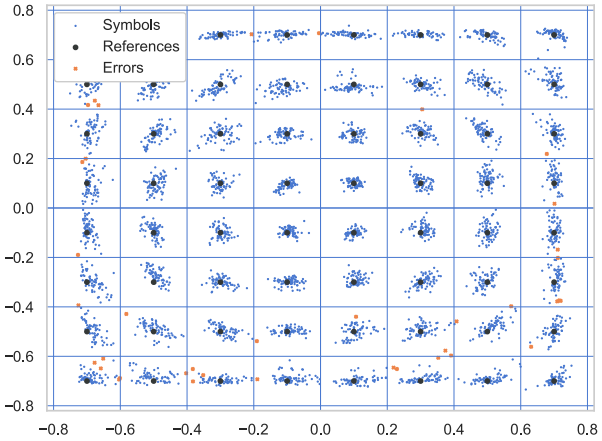
(b) m-VGG16bn



(c) m-DenseNet121



(d) VGG-like

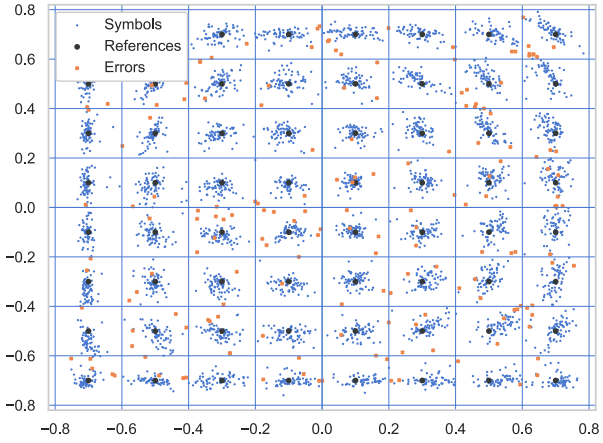


(e) MiniDenseNet

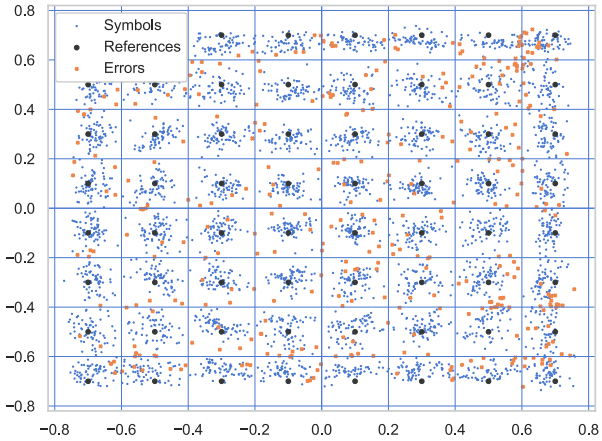
FIGURE 7. Comparative performances of neural decoding methods in dark lighting condition when tested on the unfixed LED position dataset and plotted in a 64-QAM constellation.

2) MODEL ARCHITECTURE

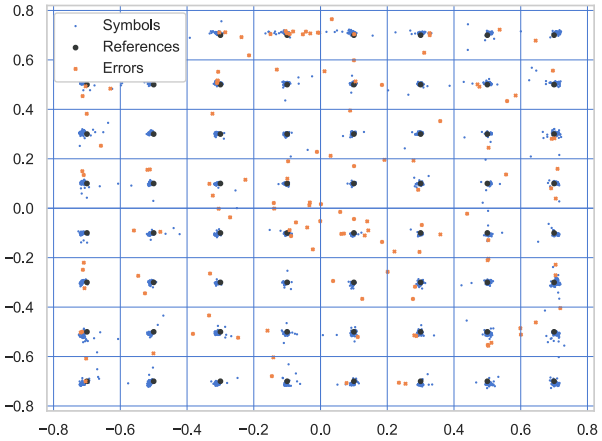
From experimental results, m-VGG16bn and VGG-like architectures were poor performance, either in terms of error rate or *rmse*, in most cases. It can be seen that, in Tables 2 and 3 and Figs. 5–8, the architectures without skip connections are the worst ones among all of the test neural



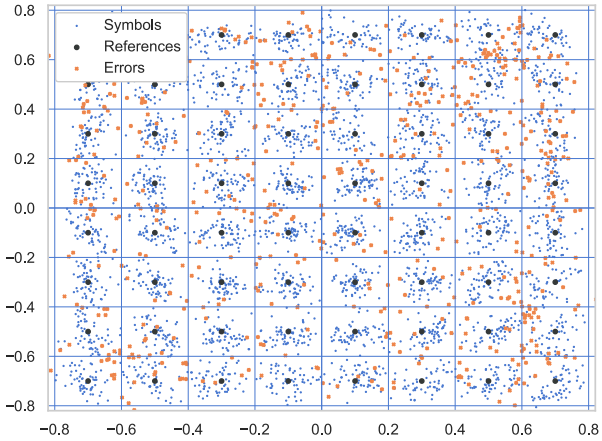
(a) m-ResNet18



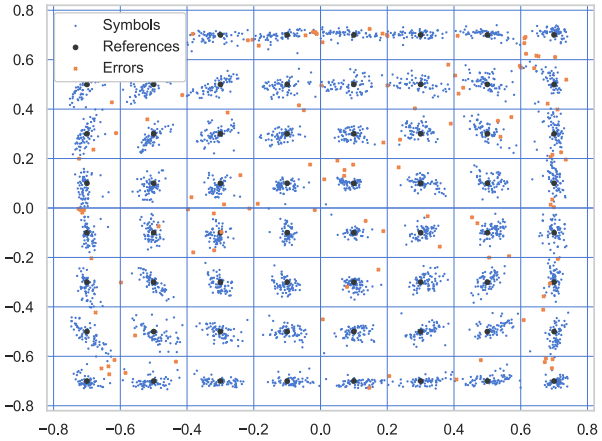
(b) m-VGG16bn



(c) m-DenseNet121



(d) VGG-like



(e) MiniDenseNet

FIGURE 8. **Comparative performances of neural decoding methods in bright lighting condition when tested on the unfixed LED position dataset and plotted in a 64-QAM constellation.**

decoding architectures. On the other hand, the architectures with skip connections commonly perform better, because the gradient effectively flows through the skip connections. The help of skip connection makes a model easy to scale. In general, the deeper the model, the better the performance [16], [23], [32]. As shown in Tables 2 and 3,

(a) error rate (b) $rmse$

FIGURE 9. Comparative performances of MiniDenseNet trained on all datasets with(out) augmentation and each specific dataset.

m-DenseNet121 had more layers (deeper) than others, so it yielded the lowest *rmse*. On the other hand, in the case of training data, that statement is always true for the model that has skip connections. However, if the model is very big (deeper), it may lead to the overfitting problem. It is similar to a fitting problem with a higher polynomial order. The higher the order, the better the performance. It is always true on the training data but is not on testing data. One may need to add more training data when going deeper to avoid the overfitting problem. Note that the optimization was subjected to the MSE loss function. Thus, the best model is the one that has the lowest MSE according to the objective function.

To explain the skip connection, let us model the skip connection on ResNet [16] as follows,

$$\mathcal{H}(\vec{x}) = \mathcal{F}(\vec{x}) + \vec{x},$$

where $\vec{x}$ is the features in a particular layer and $\mathcal{F}(\vec{x})$ is the features in the deeper layer. Thus, the learning algorithm can set the $\mathcal{F}(\vec{x})$ to be zero if the features from deeper layer does not decrease the loss, so that $\mathcal{H}(\vec{x}) = \vec{x}$. In other words, $\mathcal{H}(\vec{x})$, the deeper layer, is at least as better as $\vec{x}$, the previous layer. For the DenseNet architecture, it is the same as ResNet, but instead of using addition operator "+", it uses a concatenation of the features, which have the same intuitive explanation as to the ResNet architecture.

However, the deeper model may lead to more complexity; the model requires a larger number of parameters and computations. Also, it may lead to the overfitting problem; it fits well on training data, but not on testing data. Therefore, there is a trade-off between the lowest MSE model and the model complexity. However, in our case, MiniDenseNet had lower complexity and more accurate in terms of a very low error rate, although its *rmse* was slightly higher than m-DenseNet121.

3) UNKNOWN PARAMETER

An SW-QAM decoder requires estimating every single parameter to accurately decode the symbols. Here, we estimated the interference and removed it using *Wiener* deconvolution with $snr = 0.02$. In practice, there might be other parameters such as white noise, intensity non-linearity, etc. These undefined parameters, that we call them unknown parameters, are commonly hard to estimate and define. For instance, non-linearity is one of the undefined parameter problems of SW-QAM. This phenomenon can be seen in Figs. 5(a) and 6(a); the amplitudes of decoded symbols in the grid of (0.1,0.7), (0.1,−0.7), (0.7,0.1), and (−0.7,0.1) are degraded, they are far from the reference points, when compared to those in the grid of (0.7,0.7), (0.7,−0.7), (−0.7,0.7), and (−0.7,−0.7). This problem can be solved by adding an equalization method to SW-QAM. It means that one more complementary method is required for SW-QAM. However, in this case, the performance of the SW-QAM decoding is sufficient. Thus, we do not use the equalization method.

On the other hand, the neural decoding method can accommodate all of the unknown parameters included the non-linearity through its learning process as generally shown in the result of neural decoding methods. Therefore, we just need to optimize the objective function and then the unknown parameters will be estimated automatically.

4) THE BUILDING BLOCK

The neural decoding method is able to automatically detect features through its learning stage. This ability is revealed by the use of the unfixed LED position dataset, which contains unknown parameters of light interference, PSF, and noise. As a result, the neural decoding function is still able to correctly decode and tends to be better than the theoretical decoding. This is a salient feature of the neural decoding scheme, which makes it possible to apply for other situations and environments. However, a valid encoding function is an important aspect of a neural decoding function to accurately predict symbols. Therefore, the neural decoding still depends on the theoretical encoding as previously proposed in [6].

On the other hand, the theoretical decoding, SW-QAM, requires several complementary methods that overcome a specific type of problem. Table 5 summarizes four important requirements of the theoretical and neural decoding schemes: data modeling, LED localization, PSF modeling,

TABLE 5. A comparison of the requirements of the theoretical and neural decoding schemes.

Requirements	Theoretical decoding	Neural decoding
Data modeling	PSF image (small)	Large image dataset for training
LED localization	Manual	Learned (STN)
PSF modeling	Manual	Learned
Unknown Parameters	Need to specify	Learned

and unknown parameters. Absolutely, the theoretical decoding scheme requires a small model for a PSF image, whereas the neural decoding scheme requires a large image dataset for training. As expected, the neural decoding scheme has advantages over the theoretical decoding scheme in LED localization, PSF modeling, and unknown parameters. In other words, LED points, PSF images, and unknown parameters are automatically localized, modeled, and estimated, respectively, by our proposed CNN architectures.

V. CONCLUSION

We have proposed a neural decoding method based on an SW-QAM scheme for VLC. In real situations, the SW-QAM decoder becomes inefficient for decoding the encoded signal, especially in the case of a moving camera as shown with an unfixed LED position dataset. However, it is easily solved by the neural decoding embedded with an STN layer. Based on the neural decoding concept, we can also develop a robust and unified system for our decoding purposes, regardless of LED localization, light interference elimination, noise removal, and other unknown parameters. All can be solved by a single system, i.e., neural decoding with a training algorithm. In this paper, we have shown the use of a single MiniDenseNet model that provides excellent generalization performance in all experimental scenarios.

Nevertheless, the neural decoding needs a large dataset for training when compared to the theoretical decoding. From a comparison of both theoretical and neural decoding methods on a fixed LED position dataset, the results are comparable. Specifically, the neural decoding is slightly better than the theoretical decoding in terms of accuracy. On an unfixed LED position dataset, the performance of the neural decoding with an STN layer tends performed sufficiently well. However, note that the neural decoding still depends on the theoretical encoding, SW-QAM encoding scheme, because if a symbol is invalidly encoded and modulated, no decoding scheme will successfully decode that symbol.

In this paper, we focus on the capability of CNN for the role of decoding function in different situations. However, we still have some rooms for neural decoding improvement. For instance, one can improve the performance of neural decoding by carefully redesigning CNN architectures and applying optimization techniques in the training stage.

ACKNOWLEDGMENTS

The authors would like to thank our colleagues—Mr. S. Amaya and Mr. S. Shimada—who have helped us to collect all of the datasets and provided a very useful discussion on this work. They would also like to thank Hokkaido University for providing the equipment for the experiment.

REFERENCES

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[5] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung, "Transferring rich feature hierarchies for robust visual tracking," 2015, *arXiv:1501.04587*. [Online]. Available: https://arxiv.org/abs/1501.04587

[6] S. A. I. Alfarozi, K. Pasupa, H. Hashizume, K. Woraratpanya, and M. Sugimoto, "Square wave quadrature amplitude modulation for visible light communication using image sensor," *IEEE Access*, vol. 7, pp. 94806–94821, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8760247

[7] M. Jaderberg, K. Simonyan, K. Kavukcuoglu, and A. Zisserman, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.

[8] M. Ibnkahla, "Applications of neural networks to digital communications—A survey," *Signal Process.*, vol. 80, no. 7, pp. 1185–1215, 2000.

[9] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A survey on machine-learning techniques in cognitive radios," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1136–1159, 3rd Quart., 2013.

[10] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[11] T. Schenk, *RF Imperfections in High-Rate Wireless Systems Impact and Digital Compensation*. Dordrecht, The Netherlands: Springer, 2008.

[12] A. S. Mohammad, N. Reddy, F. James, and C. Beard, "Demodulation of faded wireless signals using deep convolutional neural networks," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 969–975.

[13] P. A. Haigh, Z. Ghassemlooy, S. Rajbhandari, I. Papakonstantinou, and W. Popoola, "Visible light communications: 170 Mb/s using an artificial neural network equalizer in a low bandwidth white light configuration," *J. Lightw. Technol.*, vol. 32, no. 9, pp. 1807–1813, May 1, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6781659/

[14] M. S. Ifthekhar, N. Saha, and Y. M. Jang, "Stereo-vision-based cooperative-vehicle positioning using OCC and neural networks," *Opt. Commun.*, vol. 352, pp. 166–180, Oct. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0030401815003569

[15] T. Nguyen, A. Islam, and Y. M. Jang, "Region-of-interest signaling vehicular system using optical camera communications," *IEEE Photon. J.*, vol. 9, no. 1, Feb. 2017, Art. no. 7900720.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[17] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, vol. 1, no. 2, pp. 2261–2269.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[19] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, Jun. 2013, vol. 30, no. 1, Art. no. 3.

[20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS Autodiff Workshop*, 2017, pp. 1–4. [Online]. Available: https://openreview.net/pdf?id=BJJsrmfCZ

[21] E. Jones, T. Oliphant, and P. Peterson. (2001). *SciPy: Open Source Scientific Tools for Python*. [Online]. Available: http://www.scipy.org/
[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[23] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.
[24] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 4278–4284. [Online]. Available: https://arxiv.org/abs/1602.07261
[25] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA, USA: MIT Press, 2009.
[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[27] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," 2016, *arXiv:1611.03530*. [Online]. Available: https://arxiv.org/abs/1611.03530
[28] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*. [Online]. Available: https://arxiv.org/abs/1312.4400
[29] D. Ho, E. Liang, I. Stoica, P. Abbeel, and X. Chen, "Population based augmentation: Efficient learning of augmentation policy schedules," 2019, *arXiv:1905.05393*. [Online]. Available: https://arxiv.org/abs/1905.05393
[30] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1—Learning rate, batch size, momentum, and weight decay," 2018, *arXiv:1803.09820*. [Online]. Available: https://arxiv.org/abs/1803.09820
[31] J. Howard *et al.* (2018). *Fastai*. [Online]. Available: https://github.com/fastai/fastai
[32] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," in *Deep Learning and Data Labeling for Medical Applications*. Cham, Switzerland: Springer, 2016, pp. 179–187.

SYUKRON ABU ISHAQ ALFAROZI (M'18) received the B.Eng. degree in electrical engineering and information technology from Universitas Gadjah Mada, Yogyakarta, Indonesia, in 2014. He is currently pursuing the Ph.D. degree in information technology with the King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, under the Sandwich Program with Hokkaido University, Sapporo, Japan. His research interests include machine learning/deep learning and its application, computer vision, and signal processing.

KITSUCHART PASUPA (M'12–SM'16) received the B.Eng. degree in electrical engineering from the Sirindhorn International Institute of Technology, Thammasat University, Thailand, in 2003, and the M.Sc.Eng. and Ph.D. degrees in automatic control and systems engineering from the Department of Automatic Control and Systems Engineering, The University of Sheffield, in 2004 and 2008, respectively. He was a Research Fellow with the University of Southampton and The University of Sheffield. He is currently an Associate Professor with the Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. His main research interest includes the application of machine learning techniques in the real-world application.

HIROMICHI HASHIZUME (M'05) received the B.E., M.E., and D.E. degrees in electronic engineering from the University of Tokyo, Tokyo, Japan, in 1979, 1981, and 1984, respectively. He is currently a Professor with the Information Systems Architecture Science Research Division, National Institute of Informatics, Japan. His research field was computer networks and telecommunication, however, recently his interest has shifted to mathematical modeling of communication systems.

KUNTPONG WORARATPANYA (M'17) received the B.Ind.Tech. degree in computer technology, the M.Eng. degree in computer engineering, and the D.Eng. degree in electrical engineering from the King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, in 1992, 1996, and 2005, respectively. He is currently an Assistant Professor with the Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang. His research interests include stereoscopic acquisition and compression, multimedia coding and processing, signal processing, speech recognition and processing, pattern recognition and image processing, computer vision, and machine learning/deep learning.

MASANORI SUGIMOTO (M'97) received the B.E., M.E., and D.E. degrees in aeronautics and astronautics from the University of Tokyo, Tokyo, Japan, in 1990, 1992 and 1995, respectively. He is currently a Professor with the Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan. His research interests include acoustic engineering, signal processing, artificial intelligence, and human–computer interaction technologies for designing smart systems and environments.