Adaptive Drawing Behavior by Visuomotor Learning Using Recurrent Neural Networks

Kazuma Sasaki and Tetsuya Ogata

***Abstract*—Drawing is a medium that represents an idea as drawn lines, and drawing behavior requires complex cognitive abilities to process visual and motor information. One way to understand aspects of these abilities is constructing computational models that can replicate these abilities rather than explaining the phenomena by building plausible models by a top-down manner. In this paper, we proposed a supervised learning model that can be trained using examples of visuomotor sequences from drawings made by human. Additionally, we demonstrated that the proposed model has functions of: 1) associating motions to depict the given picture image and 2) adapting to drawing behavior to complete a given part of the drawing process. This dynamical model is implemented by recurrent neural networks that have images and motion as their input and output. Through experiments that involved learning human drawing sequences, the model was able to associate appropriate motions to achieve depiction targets while adapting to a given part of the drawing process. Furthermore, we demonstrate that including visual information in the model improved performance robustness against noisy lines in the input data.**

***Index Terms*—Adaptation, drawing ability, recurrent neural networks, visuomotor learning.**

I. Introduction

Drawing is a nonverbal, intuitive, visual medium that can help us represent a large variety of things. Drawing behavior requires fundamental cognitive abilities to process visual information and produce lines through motor activities [1]; therefore, drawing behavior is used to measure visual constructive ability (i.e., the clock drawing test [2]) or the unconscious mind [3]. Many studies have attempted to understand this ability by constructing concrete models that can explain experimental results of drawing tests or neural activities [4]. However, understanding drawing ability remains challenging because of the many variations in drawing styles that originate from individual factors associated with those performing the drawing, such as their cultural or development characteristics [5] or the language of representation of the subject [1].

Instead of building concrete models that can explain the phenomenon, computational models have been constructed to replicate humans cognitive abilities by implicitly obtaining underlying functions through the course of development [6]. The motivation of this paper is to develop computational models that can replicate the fundamental aspects of drawing abilities. Specifically, this paper focuses on the ability to depict a visual information of a subject by producing a set of drawing motions. In this case, the drawer needs to process visual information from the subject and reproduce it through a series of drawing motions. In other words, the drawer associates motion to draw a picture from the static image of the picture [7], [8]. According to psychological studies, this association is based on the visuomotor memory of the drawing motion and visual information, including feedback from the drawn pictures, during the production [9]. Additionally, this process is involved even in situations where the subject is passively viewed as animations of line drawings [10].

Computational drawing models have been mainly investigated for computational arts [11], [12] or robotics challenges [13], [14], where the systems were mostly designed by elaborating rules to draw the desired pictures. In these cases, there are strong assumptions made related to model features in a top-down manner. Developmental systems for replicating drawing ability are studied in the field of cognitive developmental robotics or machine learning for artificial intelligence. Mohan *et al.* [15] proposed a model that is trained to reproduce given lines as a combination of the assumed primitives of robot motions with their method employing primitives for handwritten shapes based on catastrophe theory [16]. However, the designed primitives represent strong assumptions based on psychological studies. In fact, hardly any evidence exists explaining that the human drawing process follows these assumed primitives. Unsupervised training methods do not assume any concrete primitives of shapes [17]; however, the steps used to obtain drawing order are not shared with humans.

An approach not requiring explicit primitives related to drawing strategy is required to develop supervised learning models trained to imitate the given drawing actions. Mochizuki *et al.* [18] proposed a model based on recurrent neural networks ([19], RNNs) capable of making a humanoid robot draw single lines through the developmental learning of motor experiences. Another existing study concerns an RNN trained using sequences of drawings by human

Manuscript received December 8, 2017; revised June 3, 2018 and August 13, 2018; accepted August 20, 2018. Date of publication September 3, 2018; date of current version March 11, 2019. This work was supported in part by JST CREST under Grant JPMJCR15E3, in part by MEXT Grant-in-Aid for Scientific Research under Grant 15H01710, and in part by the Program for Leading Graduate Schools, "Graduate Program for Embodiment Informatics" of the Ministry of Education, Culture, Sports, Science, and Technology. *(Corresponding author: Kazuma Sasaki.)*

The authors are with the Graduate School of Fundamental Science and Engineering, Waseda University, Shinjuku 169-8050, Japan (e-mail: ssk.sasaki@suou.waseda.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCDS.2018.2868160

2379-8920 © 2018 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

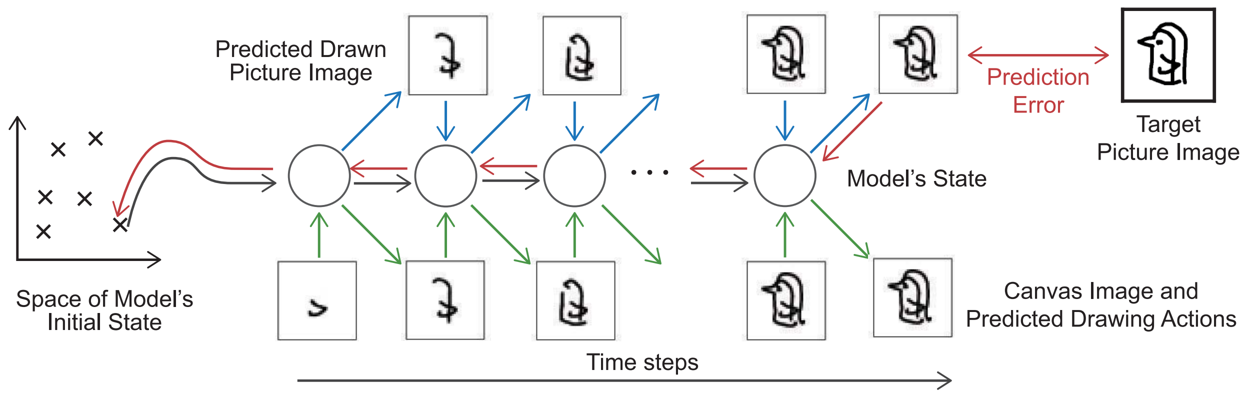
Predicted Drawn Picture Image
Prediction Error
Target Picture Image
Model's State
Space of Model's Initial State
Canvas Image and Predicted Drawing Actions
Time steps

Fig. 1. Example of drawing sequence processed by the proposed model. The black lines indicate the forward path associated with the hidden state of the model h_t, the green lines correspond to the input image from the canvas that is drawn by the action predicted during the previous step, and the blue lines describe the visual feedback loop when the new initial state is optimized to minimize the image-prediction error, which will be back-propagated through the path shown as the red line.

comprising the status of a pen [20]. This paper employed long short-term memory [21], which enabled the model to be trained with thousands of long sequences and to be generalized. However, these studies did not include visual information of the depiction target or visual feedback during production.

In this paper, we propose a supervised learning model to organize the visuomotor information of the given drawing experiences. We experimentally showed that the proposed model has the following aspects of the two fundamental human drawing behaviors related to visuomotor processing: 1) association of drawing motions to depict the given picture image and 2) adaptation of drawing behavior to complete a given part of the drawing process. The first behavior corresponds to the ability to draw multiple lines to achieve the given target image. We previously studied the concept of associating the drawing motion of single lines through a supervised learning model [22], [23]; however, the model proposed in this paper draws multiple lines. Another aspect of this model concerns the addition of lines before the model starts drawing and the requirement that the model complete the drawing. In this case, the model needs to flexibly change its behavior depending on what is drawn on the canvas. This adaptive ability also contributes to applications that can help humans draw pictures. The RNN model of sketching by Ha and Eck [20] can continue drawing when the beginning part of a sequence is provided, but it does not have any target to depict and therefore cannot control what is drawn.

The remainder of this paper is organized as follows. In Section II, we define the drawing sequence the proposed model will learn as a finite visuomotor sequence. Then, the architecture of the proposed model is described. Then, we describe how the model infers drawing actions when a part of the drawing is given. In Section III, learning experiments on 30 classes of sketching samples are described. In these experiments, we demonstrate that the model has the above-mentioned abilities. In Section IV, we compare the adaptive abilities between the proposed model and that of Ha and Eck [20]. In Section V, the adaptive abilities of the proposed model and the limitations are discussed. Finally, this paper is concluded in Section VI.

II. Supervised Learning Model for Visuomotor Adaptation Reusing Memory

In this section, we first define the visuomotor drawing sequences learned by the proposed model. Then, the proposed model is introduced. Then, we describe how to enable the association of drawing motions from an image and adaptation to part of the visual drawing process. Note that the models defined in this section are basic models. A detailed implementation of the experiments is described at the beginning of Sections III and IV.

A. *Visuomotor Process of Drawing*

In this paper, sketching is assumed to be a finite visuomotor process, as shown in Fig. 1. A process X comprises the set of the drawn-picture image on the canvas and the drawing action and state of a drawing agent as follows:

$$X = (x_1, x_2, \dots, x_t, \dots, x_T) \tag{1}$$

$$x_t = (i_t, a_t) \tag{2}$$

where i_t is a drawn-picture image at time step $t = (1, 2, \dots, T)$. Drawn picture image i_t is changed by a drawing action specified by the agent a_t. In this paper, the drawing action corresponds to the trajectory of a pen that includes not only the position but also the information to check whether the pen is touching this paper. At the beginning of the process, i_1 and a_1 are shared with all processes (i.e., i_1 is an image without any lines), and a_1 indicates that nothing is drawn. The state of the process x_t evolves as the drawing action alters the image until the maximum step of the process T.

In this paper, we consider a function that advances the state of the finite visuomotor process. Furthermore, the next step state is assumed to be determined by not only the current state x_t but also the hidden state h_t that represents a memory of the agent

$$x_{t+1}, h_t = f(x_t, h_{t-1}). \tag{3}$$

Approximation of the function f can be achieved by supervised training using neural networks [24]. The model is implemented by an RNN that recently showed a good ability

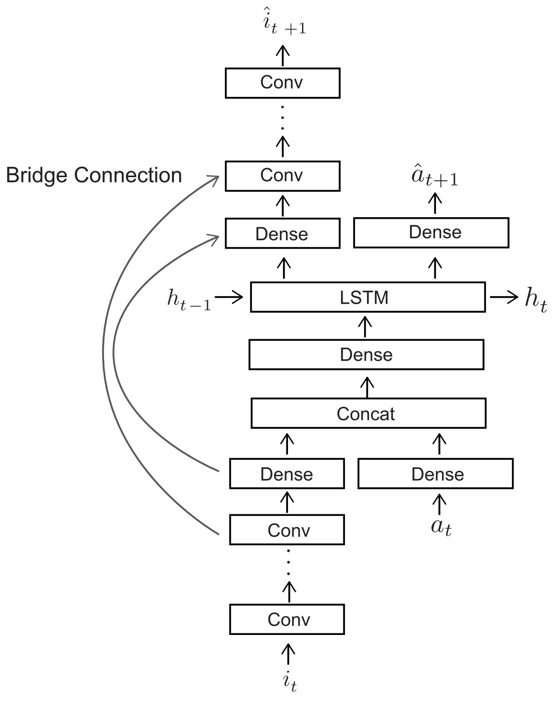

Fig. 2. Architecture of the proposed model. Each box indicates a part of the forwarding process. Conv: Convolution2D or Transposed-Convolution2D with a nonlinear activation function; Dense: linear mapping with an activation function. LSTM: long short-term memory cell. Further, i_t and $\hat{i}_t$ represent the drawn-image input and the image predicted by the model at time step t, respectively. a_t is the input action, $\hat{a}_{t+1}$ corresponds to the predicted action, and h_t indicates the hidden state of the LSTM cells.

to learn complex time-series data for language processing [25], image generation from captions [26], and continuous-sequence generation for a reinforcement agent [27]. When the model draws the picture, it accepts the current picture image i_t, the drawing action taken at the previous step, and the hidden state obtained at the previous step (i_t, a_{t-1}, h_{t-1}). The prediction by the model is conditioned by the hidden state at the first step $h_{t=0}$ as well as the image inputs.

B. Model Architecture

To enable visuomotor sequence modeling by neural networks, we propose a novel RNN architecture, as described in Fig. 2. The model sequentially predicts the data for the next step that comprises an image and a corresponding drawing action by updating the state of the hidden unit. The input image i_t is processed by the image-encoding function, which is implemented by convolutional layers. The encoded feature is concatenated with another feature of the input-action data encoded by the dense connected layer. The concatenated feature becomes the input of the LSTM layer. The output of the LSTM layer is split into the image and action features. The action feature is decoded into the action predicted by the dense connected layer, and the image decoder corresponds to transposed convolution layers accepting not only the input from the LSTM layer but also the intermediate features of the image-encoding part through the bridge connections (see Appendix A). These connections reduce the burden of encoding spatial information of the input image when the output image is similar to the input because the model does not need to predict all of the image features but only the difference between the input and output [28]. The state of the LSTM layer is recursively fed to the above-mentioned forwarding process. The state at the first step is provided by an additional dense connected layer as follows:

$$h_{t=0} = \tanh\left(W^{\text{init}}\bar{h}\right) \tag{4}$$

where W^{init} corresponds to the weight matrix and $\bar{h}$ is the "initial value" vector having dimensionality $h_{t=0}$, because preparing $h_{t=0}$ for all of the sequences has a large computational cost. We also allow this initial value to be trained to distinguish the difference between the trained sequences [29].

C. Training Model

We adopt a supervised learning method based on a stochastic gradient-descent (SGD) method with an Adam optimizer [30] to train the proposed RNN model. During the forward propagation process to obtain predictions, the model accepts input from the previous predictions at each step, with the exception of the first step, as follows:

$$i_t = \begin{cases} i_t & (t = 1) \\ \hat{i}_{t-1} & (t > 1) \end{cases} \tag{5}$$

$$a_t = \begin{cases} a_t & (t = 1) \\ \hat{a}_{t-1} & (t > 1) \end{cases} \tag{6}$$

where $\hat{i}_t$ is the image prediction and $\hat{a}_t$ is the action prediction. As mentioned previously, the initial image input i_1 represents the picture without any lines, and a_1 corresponds to the fixed initial status of the pen. From these initial inputs and the initial value, the model generates a visuomotor sequence by recursively feeding the previous output into itself. Note that the initial values are initialized to zero for each training sequence. These initial values as well as the other learnable parameters, can be changed during the training process.

After generation, we calculate the error between the prediction and the target data following the objective function L to obtain the gradients of all learnable parameters. L is obtained by accumulating prediction errors at all time steps t as follows:

$$L = \sum_{t=2}^{T}\left(L^{\text{img}}\left(\hat{i}_t, i_t\right) + L^{\text{act}}(\hat{a}_t, a_t)\right) \tag{7}$$

where L^{img} corresponds to the error between the predicted image $\hat{i}_t$ and the target image data i_t, and L^{act} represents the error between the predicted action $\hat{a}_t$ and target-action data a_t. The target image and action data are obtained by collecting human drawing data.

In the first experiment, we utilized the original training dataset collected by a human drawer. Another dataset provided by the QuickDraw! Project [31] was used for the second experiment. Note that the implementation of L^{img} and L^{act} depends on the target data. Further detail concerning these functions and the representation of the action predictions are given in Sections III and IV, respectively.

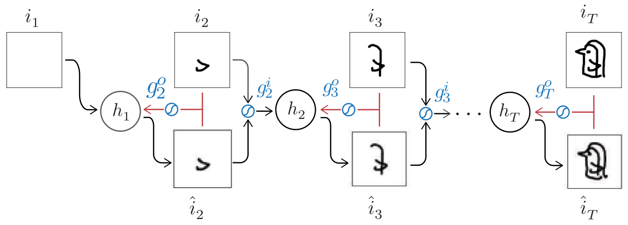

Fig. 3. Gating for propagating path of initial value regression.

D. Initial Value Regression

The adaptive behavior to depict the given target image is implemented through regression of a new initial state $h_{t=0}$ with fixing the weights of the model [22], [23]. The objective function of this optimization is the error between the target image and the image predicted by the model at the final step T. The predicted image is obtained by the forward propagation of the model by recursively providing information from the previous step as input. This means that the model iterates the internal simulations using its own predictions rather than taking actions to obtain images.

In addition, the models behavior can follow the intermediate step of the drawing process by changing the objective function to regress the initial state. When a beginning part of the drawing process is provided to the model, we add the other image prediction losses of the given part of the process. We assume that the part of the process is obtained as the process of an incomplete picture drawn by another agent. In this case, the model must adapt the given incomplete drawing process and draw the target image. To enable partial accumulation of the image prediction error, we introduce gating values representing image inputs and outputs as described in Fig. 3. These gating values comprise input and output gating values that help to determine the source of the input and mask the prediction error, respectively. These values are represented by a vector with elements 1 or 0. The objective function involves the regression task based on accumulated prediction error from images partially masked by the output-gating values

$$L^{\text{reg}} = \sum_{t=2}^{T} g_t^o \odot L^{\text{img}}\left(\hat{i}_t, i_t\right) \tag{8}$$

where g_t^o is the gating value of losses. Image prediction $\hat{i}_t$ is obtained by the forwarding process of the trained model. The input to the model is determined by the input-gating values g_t^i as follows:

$$i_t = \begin{cases} i_t & (t = 1) \\ g_t^i \odot i_t + \left(1 - g_t^i\right) \odot \hat{i}_{t-1} & (t > 1) \end{cases} \tag{9}$$

$$a_t = \begin{cases} a_t & (t = 1) \\ \hat{a}_{t-1} & (t > 1). \end{cases} \tag{10}$$

If $g_t^i = 1$, the model receives inputs from the status of the canvas; otherwise, the model recursively uses its previous prediction. Note that both of the gating values are determined by the experimenter as a hyperparameter. For example, $g_2^i, g_3^i = 1$, $g_2^o, g_3^o, g^T = 1$, and the others are zero when the first two lines are already drawn.

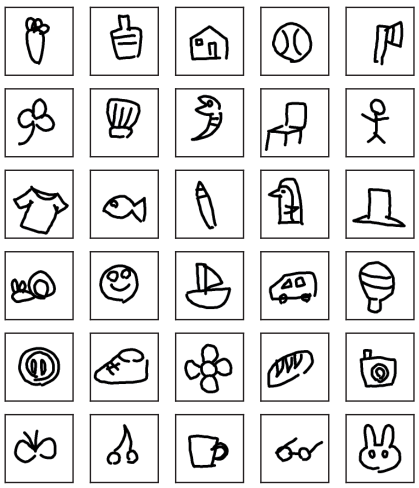

Fig. 4. Example pictures from the training dataset for the first experiments.

III. Experiments on the Association of Drawing Motion From Image

We conducted experiments on learning sketch samples to demonstrate the drawing ability of the proposed model. First, we show that the model successfully remembers the examples of drawn sequences. Next, its adaptive ability to draw a given target image was confirmed. In this case, the model was required to adapt not only to draw the depiction target but also to continue drawing from the beginning of the provided drawing. Finally, we tested the generalization ability for line ordering. In this case, the model is required to complete a given part of the picture that was drawn in not trained line ordering.

A. Experimental Settings

To render images by drawing actions, we utilized a drawing simulator, which draws black lines of a fixed thickness and therefore does not take pressure into consideration. The training dataset was collected by a human drawer (Fig. 4) and contained 30 types of pictures with multiple lines. Each type of picture contained four samples with the same number of lines (3–5 lines). The drawn-picture image was 128×128 pixels. The drawing action was a vector representing the position of the pen and its status depicting whether or not the pen was touching the canvas [32]. To enable the forward propagation of multiple sequences as a batch, the length of each sequence was aligned to be 100 steps by repeating the last steps data until 100 steps. Further detail of the implementations is available on Appendix B.

Optimizing an RNN for visuomotor sequences is challenging because it incurs a high computational cost when the process is long. Therefore, we split this training task into two parts: 1) drawing a single line and 2) organizing combinations of lines. The proposed model accordingly comprised an RNN for drawing a single line (LineRNN) and another for controlling LineRNN (PicRNN). Both of these RNN models shared similar structures but were trained as different models.

LineRNN was trained with single line-stroke sequences, with the input comprising the drawn-picture image i_t^L, and the corresponding pen status as follows:

$$x_t^L = \left(h_t^L, i_t^L, p_t\right) \tag{11}$$
$$p_t = (x_t, y_t, d_t) \tag{12}$$

where x_t and y_t are the absolute position of the canvas and the status of the pen, respectively. All sequences were normalized to start at the center of the canvas and return to the starting position. After returning to the starting position, the pen stays there without drawing anything (i.e., $d_t = (0, 1)$ until the time step reaches the end of the sequence. L^{img} for LineRNN represents the cross-entropy of the binary image as follows:

$$L_t^{\text{img}}\left(i_t^L, \hat{i}^L\right) = i_t^L \log \hat{i}^L + \left(1 - i_t^L\right) \log\left(1 - \hat{i}_t^L\right). \tag{13}$$

The loss function for the pen status L^{act} is given by the negative log likelihood of a bivariate normal distribution and the cross-entropy for the classification as follows:

$$L_t^{\text{act}}(p_t, \hat{p}_t) = -\log \mathcal{N}\left(x_t, y_t | \hat{\mu}_t^x, \hat{\mu}_t^y, \hat{\sigma}_t^x, \hat{\sigma}_t^y, \hat{\rho}_t\right) - d_t \log \hat{d}_t \tag{14}$$

where $\hat{\mu}_t^x$, $\hat{\mu}_t^y$, $\hat{\sigma}_t^x$, $\hat{\sigma}_t^y$, and $\hat{\rho}_t$ are the means of the x-coordinate, the y-coordinate, the variance of the x-coordinate, the variance of the y-coordinate, and the covariance, respectively.

PicRNN was trained not to predict the status of the pen but to predict the order of the lines as the initial values of LineRNN along with the probability depicting whether the drawing process has ended

$$x_t^P = \left(h_t^P, i_t^P, a_t^P\right) \tag{15}$$
$$a_t^P = \left(\bar{h}_t^L, q_t\right) \tag{16}$$

where h_t^P represents the state of PicRNN, i_t^P is the input image to the model, $\bar{h}_t^L$ corresponds to the initial value of LineRNN, and q_t indicates the probability of "end-of-drawing." Note that the forwarding of PicRNN is processed as each line is drawn by LineRNN; therefore, i_t^P is the image after a single line is added. L^{img} is the same as (13). L^{act} comprises the mean square loss for vector regression and cross-entropy loss of q_t. To prepare the training dataset for PicRNN, LineRNN is trained in advance because the initial value $\bar{h}^L$ becomes part of the training dataset. Note that $\bar{h}^L$ is normalized not to exceed the range of the activation function. In the experiment, we used tanh as the activation function. Therefore, $\bar{h}^L$ is normalized by tanh and normalized again by artanh an input to LineRNN. The sequences of the training dataset for PicRNN are aligned to have the same length. At the first step, h_t^L is set to be a zero vector and $q_1 = (1, 0)$. PicRNN generates the sequence that comprises an initial state of LineRNN and image data from the initial value of the hidden state. After drawing all of the lines, $\bar{h}_t^L$ is kept, and q_t continues to be (0, 1) until the desired maximum length of sequence.

The parameters for LineRNN and PicRNN are described in Tables I and II, respectively. The hidden units of LineRNN and PicRNN correspond to LSTM cells with peep-hole connections [21]. The activation function for all layers is tanh to normalize the range of the output to [0, 1]. The dimensional size of initial hidden states is, respectively, indicated in the bottom line of each table.

TABLE I
PARAMETERS OF LINERNN

PART	NUM
Image Encoder	4 (conv), 2 (dense)
Image Decoder	4 (conv), 2 (dense)
Action Encoder	1 (dense)
Action Decoder	1 (dense)
RNN	200 cells
Initial State	30 dims

TABLE II
PARAMETERS OF PICRNN

PART	NUM
Image Encoder	4 (conv), 2 (dense)
Image Decoder	4 (conv), 2 (dense)
Action Encoder	-
Action Decoder	1 (dense)
State Encoder	1 (dense)
State Decoder	2 (dense)
RNN	200 cells
Initial State	10 dims

Both of the RNN models are separately trained using an Adam optimizer [30] with $\alpha = 0.001$ and $\beta 1 = 0.75$. First, LineRNN is trained to represent all the lines in the dataset. Next, PicRNN is trained with the obtained initial values of LineRNN and the corresponding drawn images. Note the training data of PicRNN has all of the possible line combinations except the one used for making the test dataset. We utilized the test dataset to confirm the generalization ability of the models for the line ordering.

When a new initial value is obtained, Adam optimization is utilized. For each regression task, 50 candidates for the initial values of PicRNN are initialized using a normal distribution with zero means and a variance of 1. To speed up the optimization, half of the worst candidates that lead to higher loss are replaced by a normal distribution whose mean is based on the best candidates derived from every 100 training iterations. After optimization, the model uses one of the acquired initial values that give the lowest loss to draw.

When the model draws without any part of the process given, the gating values for the initial-value regression are given to obtain a prediction error at the final step and acquire any inputs at time step $t > 1$ (i.e., $g_i = (1, 0, 0, \ldots, 0)$ and $g^o = (0, 0, \ldots, 0, 1)$). If part of the first line is provided by the experimenter, the gating values are $g_i = (1, 1, 0, 0, \ldots, 0)$ and $g^o = (0, 1, 0, 0, \ldots, 0, 1)$. In the situation of untrained line orderings, the gating values are given as follows:

$$g_t^i = \begin{cases} 1 & (1 \le t \le T) \\ 0 & (T < t) \end{cases} \tag{17}$$
$$g_t^o = \begin{cases} 1 & (1 \le t \le T \wedge t = T) \\ 0 & (T < t < T) \end{cases} \tag{18}$$

where T is the length of the process. In the experiment, we set $T = 7$ because of the maximum length of the sequence in the training dataset.

B. Results

The results generated from one of the trained initial values are shown in Fig. 5. Fig. 5(a) shows the ground truth of the

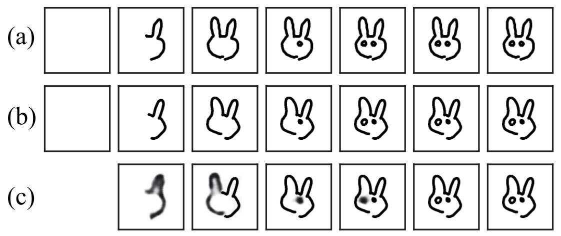

Fig. 5. Results generated by the trained RNNs. (a) Ground truth from the training dataset. (b) Pictures drawn by the model. (c) Predicted drawing images.

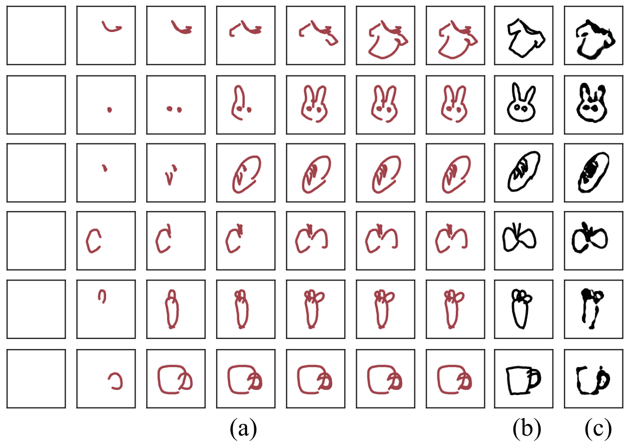

Fig. 6. Results of initial-value regression. (a) Pictures drawn by the model. (b) Depiction target of the images. (c) Predicted drawing images at the final step of the sequence associated with the initial value.

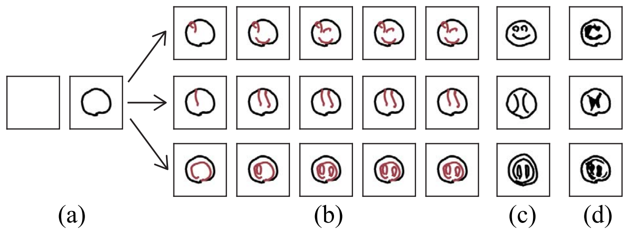

Fig. 7. Results of initial-value regression on sketches starting from a line. Black lines: lines given by the experimenter. Red lines: lines drawn by the model. (a) First two lines utilized as the conditional input to the regression. (b) Snapshot of the drawing process generated by the model after regression. (c) Depiction target. (d) Image at the end of drawing generated by the model.

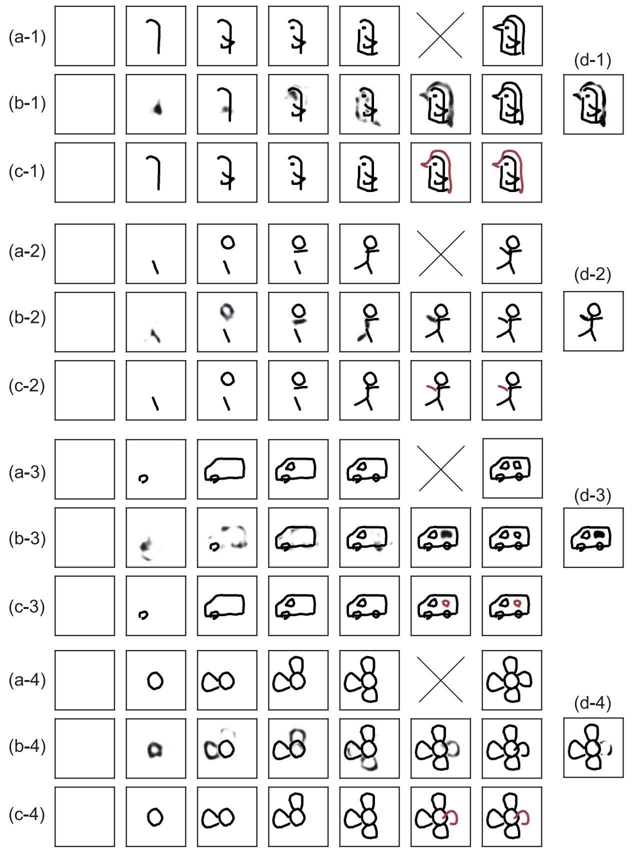

Fig. 8. Results of initial-value regression on untrained combinations of lines. Black lines: lines given by the experimenter. Red lines: lines drawn by the model. (a-1) to (a-4): The lines provided to the model. (b-1) to (b-4): The predicted image by PicRNN during the drawing process. (c-1) to (c-4): Lines drawn by the associated drawing motions.

dataset. Fig. 5(b) and (c) shows the pictures drawn by the predicted actions of the model and the predicted drawing images $\hat{i}_t^P$, respectively. To draw pictures, the model starts from the acquired initial value of PicRNN, which corresponds to the ground truth in the training dataset. As shown in the figure, the model adds lines according to its prediction of the drawn image and finally achieves the picture that is similar to the ground truth.

Fig. 6 describes the generation results using the initial values obtained by regression to minimize the prediction error of the image at the final step. As shown in the figure, the model generated the image which is visually close to the target image. The model drew pictures by generating the pen's position sequences using the obtained initial value, as shown by Fig. 6(a). Finally, the drawing process is finished at the maximum time step, and the pictures are similar to the depiction target. These drawing results suggest that the model can draw the target image without intervention during the drawing process.

The results of initial-value regression when the first line is given are shown in Fig. 7. Note that we assumed that all of the given parts of the drawing process were the same but the depiction targets were different. The model clearly changed its prediction to that of the given image of the depiction target.

Fig. 8 shows additional initial-value regression results. This regression task required the model to add the last line in order to sketch the depiction target when the other lines were provided through untrained ordering. Additionally, the depiction targets were also untrained. As shown in Fig. 8, the model added the last line similar to the ground truth from the untrained data. These results suggested that the model generalized the line ordering from the pictures in the dataset.

IV. Comparison of the Proposed Model With RNN Model Without Vision

To clarify the difference between the proposed model and RNN models without vision, we conducted adaptation experiments on associating drawing motion when the beginning part of the process is given. In these experiments, the models were

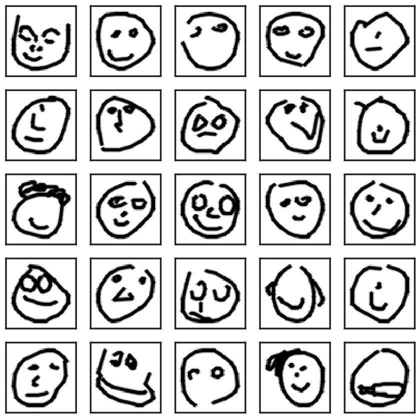
Fig. 9. Example pictures from the training dataset for the second experiments.

required to complete the given drawing process. Further, the given process included additional lines that were unrelated to the given target image. In this case, the model needed to be robust against these "noisy" lines. Specifically, the case of face drawings was examined. The scenario assumed for these experiments was that the models depicted faces when the outlines of faces were provided by the experimenter. The noisy lines were represented by short lines added to the outside of the outline.

A. Experimental Settings

We utilized sketch-rnn by Ha and Eck [20] as the model for comparison. This model is a variational autoencoder [33] with the encoder and the decoder part implemented by LSTMs. In these experiments, the unconditional model (decoder RNN-only) was utilized. The model had 1024 LSTM cells and a 5-D initial state. First, the initial state was initialized by a zero vector, and the pen-stroke data for the input lines were provided to allow acquisition of the updated state. The model then initiated drawing until it inferred the end of the drawing. The model predicts the pen status as a Gaussian mixture distribution. Ha *et al.* introduced a "temperature" τ hyperparameter with a value of (0, 1) to control the level of randomness to mix predicted Gaussian components. We chose $\tau = 0.25$ because this value demonstrated the best result in several trials.

In the second experiment, we did not split the learning task into two sub-RNN models, but we utilized a single model whose structure is mostly same as LineRNN in the last experiments. To enable training visuomotor sequences using the small RNN model, we reduce the length of sequences and the resolution of the image. The training dataset is the QuickDraw Dataset [31] from the Magenta project [34], whose example pictures are shown in Fig. 9. To create the training dataset, we randomly select 1000 sequences comprised less than 100 steps from the original dataset. Note that the length of the sequences for the first experiment was 500, and the sequences for this experiment are not divided by lines. The position of the pen was converted to the absolute point not exceeding the canvas of the simulator. The images comprised 64×64 binary pixels, where the resolution of the first experiment was 128.

TABLE III
PARAMETERS OF MODEL

PART	NUM
Image Encoder	4 (conv), 2 (dense)
Image Decoder	4 (conv), 2 (dense)
Action Encoder	1 (dense)
Action Decoder	1 (dense)
RNN	256 cells
Initial State	10 dims

As the test dataset, we utilize a picture of a face as the target image in Fig. 10.

The objective function comprised the loss of image and the action. The image loss L^{img} was the same as (13), and the loss of action is given as follows:

$$L^{\mathrm{act}} = -\log \mathcal{N}\big(x_t, y_t | \hat{\mu}_t^x, \hat{\mu}_t^y, \hat{\sigma}_t^x, \hat{\sigma}_t^y, \hat{\rho}_t\big) - r_t \log \hat{r}_t \quad (19)$$

where $\hat{r}_t$ represents the predicted probability of the pen status, which is utilized in sketch-rnn, and r_t indicates the corresponding target data. The parameters of the model are described in Table III. The objective function is the same as that used by LineRNN in the first experiment. After training this model, the optimization process for the new initial value was conducted. The 50 candidates of the initial values were initialized by uniform distribution in $[-0.1, 0.1]$ and updated by SGD with a momentum optimizer. Note that the annealing method for reinitialization of the candidates was not used in these experiments.

B. Results

The drawing results from both sketch-rnn and the proposed model are summarized in Fig. 10. Each row in the figure indicates the input lines shown in each first column, respectively. Note that rows (a-1) to (a-3) indicate the input lines without any noise, and (b-1) to (b-5) correspond to the case where very short lines were added outside of the outline as noise. The second column shows the results drawn by sketch-rnn, and the third column shows results drawn by the proposed model following the regression process of the initial value for the given target image and the given input lines. As shown in the figure, both models drew the other parts of the face inside of the output line. The expressions on the faces (e.g., the shape of the eyes or mouth; the presence or absence of a nose) drawn by sketch-rnn differed from the depiction-target because this model was unable to accept any predictive image data describing what the drawn-picture should be. The results for the input lines containing noise suggested that sketch-rnn was sensitive to noise, resulting in the drawn parts of the faces straying from the outline. The comparison results showed that the proposed model succeeded in drawing inside the outline.

V. Discussion

Our goal was to construct a supervised learning model to accept and predict visual information of a drawn-picture at each step during the production process. By adding this additional modality, the model becomes capable of replicating a

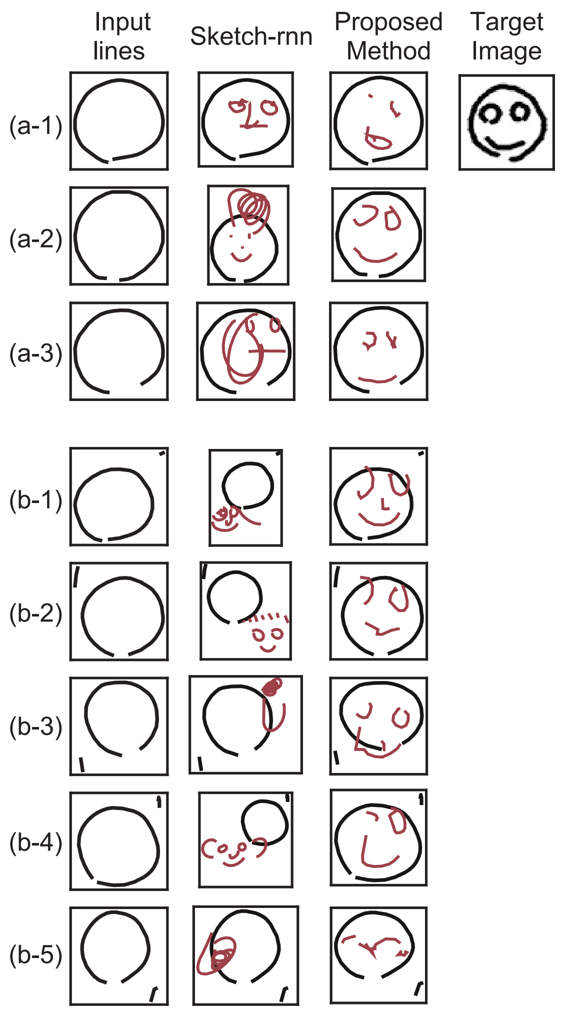

Fig. 10. Results of drawing adaptation to the given input lines. Black lines: lines given by the experimenter. Red lines: lines drawn by the models.

few functions related to the human drawing ability. The model was allowed to accept the information depicting what is drawn on the canvas, suggesting that the model can acquire visual feedback from the canvas that potentially includes intervention in the process by another drawer. In the case of humans, visual feedback activates the premotor area associated with writing [35] and the recognition of letters [36]. Additionally, even when visual feedback is not obtained, the subject uses the dynamic information of shapes (e.g., stroke order or direction) by seeing static images [8]. In our first experiment, the model switched its behavior to depict a given target image or visual feedback provided as input strokes. We believe that this adaptation corresponds to the conversion from a static image (the depiction target) to a dynamical drawing motion. According to models of human drawing ability [4], language information potentially influences the depiction target. However, the model proposed in this paper does not accept language or distinguishable information concerning the pictures as input. Therefore, we considered incorporating cases related to human tasks in this context and involving examination of nonletter representations, such as artificial letters or simple shapes.

In the experiments using the first dataset (Section III), the model demonstrated its visuomotor adaptation ability for two cases: 1) association drawing motion without intervention and 2) the first line is given by the experimenter. Furthermore, the model successfully added the line when the rest of the lines were given according to an untrained order. However, the confirmation of the generalization ability for accepting unknown line shapes remains because the given lines are obtained from the training dataset.

In Section IV, we compared the adaptation ability of the proposed model and another RNN model without vision. In these experiments, the proposed model demonstrated better performance relative to continuing the given drawing with noisy lines. Even when we considered only the case with small noisy lines, the drawing results showed apparent distortions. We consider that the depiction performance decreases as the level of noise increases. Thus, the model will not be able to draw meaningful pictures.

When we compare the pictures drawn by the proposed model between the two experiments, the quality of the drawn pictures from the second experiment was worse than that of the pictures of the first experiment. We surmise that this difference in quality was caused by the variety of expressions in the dataset. In the first experiment, the dataset had several variations of the face, but the faces mostly have similar expressions (shape of eyes, position of mouse, etc.). The dataset used in the second experiment had many variations because the pictures were drawn by many different people.

Regarding the contribution this model makes to computational applications related to human drawing, this method represents an end-to-end model of a drawing process that does not require any prior knowledge of the drawing strategy or shapes. This means that the proposed model is not limited to specific types of sketches. However, the model has difficulty optimizing learning processes involving complex drawing sequences. For example, realistic drawings, which typically have more than thousands of lines, have a high computational cost. Also, a more detailed picture will require a higher resolution image. We consider that the special capability of the proposed model can be improved by considering splitting learning task into a part of image [37] or by acquiring dimensional compressed image features [38].

Another limitation of the proposed method is that the model cannot suggest turn-taking during interactions. Regarding human/machine-sketching interactions, turn-taking plays an important role. In such a case, the system can automatically draw one part of the picture while the user can focus on other parts that need additional attention. Our future work includes allowing the RNN to output the prediction along with associated uncertainty levels [39].

VI. Conclusion

In this paper, we proposed a supervised leaning model to replicate human functions related to visuomotor memory for drawing behavior. The proposed model learns sketching behavior as a visuomotor process that involves drawing actions and the corresponding process associated with drawing images.

This consideration enables the model to hold a visuomotor memory of the drawing process that enables it to determine a drawing process by minimizing the error between the predicted images and the given images through the depiction target and the conditional input strokes. This model is implemented by an RNN trained to be a dynamic model of the drawing. The association process for drawing a given depiction target is realized by regression of the initial state of the hidden unit of the RNN. The model was examined to evaluate its adaptive abilities related to accepting provided raster images and input-image sequences as conditional strokes as inputs to the drawing process. One direction for further studies involves comparison of model outputs against results from human drawings following consideration of language information related to the depiction target.

Appendix A
Detail of the Image Encoder and Decoder

The image encoder and decoder in the RNNs used in the proposed model comprise four convolutional layers and one dense linear-mapping layer. The forwarding process of the convolutional layers in the image decoder is given as follows:

$$y = \tanh(W * x) \tag{20}$$

where W is the weight matrix and x indicates the input to the layer. To reduce dimensionality, all the convolutional operations are processed with stride two. However, the forwarding process of the convolutional layers in the decoder portion involves the transposed version of the encoder, and it also accepts the image features in the encoder through the bridge connections

$$y = \tanh(x * W + m) \tag{21}$$

where m is the image feature from the paired layer in the encoder. Note that the stride value is also two as in the case of the decoder.

Appendix B
Software Implementation Detail

To conduct the experiments, we utilized a simulation environment which was implemented mainly using Python. The simulated canvas accepts a drawing command and gives an access to a drawn picture image rendered by Python Image Library. Learning RNN models were implemented using Tensorflow. The training loss was converged in less than a few days using a GPU (NVIDIA Titan X). The implementation of the LSTM layer was based on [21].

References

[1] P. V. Sommers, "A system for drawing and drawing-related neuropsychology," *Cogn. Neuropsychol.*, vol. 6, no. 2, pp. 117–164, Mar. 1989.

[2] M. Freedman *et al.*, *Clock Drawing: A Neuropsychological Analysis*. New York, NY, USA: Oxford Univ. Press, 1994.

[3] S. Kahill, "Human figure drawing in adults: An update of the empirical evidence, 1967–1982," *Can. Psychol.*, vol. 25, no. 4, pp. 269–292, 1984.

[4] S. McCrea, "A neuropsychological model of free-drawing from memory in constructional apraxia: A theoretical review," *Amer. J. Psychiatry Neurosci.*, vol. 2, no. 5, pp. 60–75, 2014.

[5] M. Amenomori, A. Kono, J. S. Fournier, and G. A. Winer, "A cross-cultural developmental study of directional asymmetries in circle drawing," *J. Cross Cult. Psychol.*, vol. 28, no. 6, pp. 730–742, 1997.

[6] M. Asada, K. F. MacDorman, H. Ishiguro, and Y. Kuniyoshi, "Cognitive developmental robotics as a new paradigm for the design of humanoid robots," *Robot. Auton. Syst.*, vol. 37, nos. 2–3, pp. 185–193, Nov. 2001.

[7] M. K. Babcock and J. J. Freyd, "Perception of dynamic information in static handwritten forms," *Amer. J. Psychol.*, vol. 101, no. 1, pp. 111–130, 1988.

[8] J. J. Freyd, "Dynamic mental representations," *Psychol. Rev.*, vol. 94, no. 4, pp. 427–438, 1987.

[9] A. H. Waterman, J. Havelka, P. R. Culmer, L. J. B. Hill, and M. Mon-Williams, "The ontogeny of visual-motor memory and its importance in handwriting and reading: A developing construct," in *Proc. Roy. Soc. B Biol. Sci.*, vol. 282, no. 1798, Jan. 2015, Art. no. 20140896.

[10] Y. Yuan and S. Brown, "The neural basis of mark making: A functional MRI study of drawing," *PLoS ONE*, vol. 9, no. 10, Oct. 2014, Art. no. e108628.

[11] P. Tresset and F. F. Leymarie, "Portrait drawing by Paul the robot," *Comput. Graph.*, vol. 37, no. 5, pp. 348–363, Aug. 2013.

[12] H. Cohen, *The First Artificial Intelligence Coloring Book*. Boston, MA, USA: Addison-Wesley, 1983.

[13] S. Kudoh, K. Ogawara, M. Ruchanurucks, and K. Ikeuchi, "Painting robot with multi-fingered hands and stereo vision," *Robot. Auton. Syst.*, vol. 57, no. 3, pp. 279–288, Mar. 2009.

[14] S. Mueller, N. Huebel, M. Waibel, and R. D'Andrea, "Robotic calligraphy—Learning how to write single strokes of Chinese and Japanese characters," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tokyo, Japan, Nov. 2013, pp. 1734–1739.

[15] V. Mohan *et al.*, "Teaching a humanoid robot to draw 'shapes,'" *Auton. Robots*, vol. 31, no. 1, pp. 21–53, Jul. 2011.

[16] V. S. Chakravarthy and B. Kompella, "The shape of handwritten characters," *Pattern Recognit. Lett.*, vol. 24, no. 12, pp. 1901–1913, 2003.

[17] N. Xie, H. Hachiya, and M. Sugiyama, "Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting," *IEICE Trans. Inf. Syst.*, vol. E96.D, no. 5, pp. 1134–1144, Jun. 2013.

[18] K. Mochizuki, S. Nishide, H. G. Okuno, and T. Ogata, "Developmental human–robot imitation learning of drawing with a neuro dynamical system," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Manchester, U.K., 2013, pp. 2336–2341.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. [Online]. Available: http://www.nature.com/doifinder/10.1038/323533a0

[20] D. Ha and D. Eck, "A neural representation of sketch drawings," *Arxiv*, pp. 1–20, Apr. 2017. [Online]. Available: https://arxiv.org/abs/1704.03477

[21] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, 2014, pp. 338–342.

[22] K. Sasaki, K. Noda, and T. Ogata, "Visual motor integration of robot's drawing behavior using recurrent neural network," *Robot. Auton. Syst.*, vol. 86, pp. 184–195, Dec. 2016.

[23] K. Sasaki, H. Tjandra, K. Noda, K. Takahashi, and T. Ogata, "Neural network based model for visual-motor integration learning of robot's drawing behavior: Association of a drawing motion from a drawn image," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 86, Sep. 2015, pp. 2736–2741.

[24] M. I. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cogn. Sci.*, vol. 16, no. 3, pp. 307–354, 1992.

[25] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, pp. 1–23, Sep. 2016.

[26] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *CoRR*, vol. abs/1411.4555, pp. 1–9, Nov. 2014.

[27] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, pp. 1–19, Feb. 2016.

[28] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.0, pp. 1–8, May 2015.

[29] R. Nishimoto and J. Tani, "Learning to generate combinatorial action sequences utilizing the initial sensitivity of deterministic dynamical systems," *Neural Netw.*, vol. 17, no. 7, pp. 925–933, 2004.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, Dec. 2014, pp. 1–15. [Online]. Available: http://arxiv.org/abs/1412.6980

[31] *The Quick, Draw! Dataset*. Accessed: Dec. 2, 2017. [Online]. Available: https://github.com/googlecreativelab/quickdraw-dataset

[32] A. Graves, "Generating sequences with recurrent neural networks," *Arxiv*, pp. 1–43, Aug. 2013. [Online]. Available: http://arxiv.org/abs/1308.0850

[33] S. R. Bowman *et al.* (Nov. 2015). *Generating Sentences From a Continuous Space*. [Online]. Available: http://arxiv.org/abs/1511.06349

[34] *Magenta*. Accessed: Dec. 2, 2017. [Online]. Available: https://magenta.tensorflow.org/

[35] M. Longcamp, J.-L. Anton, M. Roth, and J.-L. Velay, "Visual presentation of single letters activates a premotor area involved in writing," *Neuroimage*, vol. 19, no. 4, pp. 1492–1500, 2003.

[36] J. Parkinson and B. Khurana, "Temporal order of strokes primes letter recognition," *Quart. J. Exp. Psychol.*, vol. 60, no. 9, pp. 1265–1274, 2007.

[37] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "DRAW: A recurrent neural network for image generation," in *Proc. 32nd Int. Conf. Mach. Learn. Res. (PMLR)*, vol. 37. Lille, France, Jul. 2015, pp. 1462–1471. [Online]. Available: http://proceedings.mlr.press/v37/gregor15.html

[38] D. P. Kingma, "Fast gradient-based inference with continuous latent variable models in auxiliary form," *Arxiv*, pp. 1–9, Jun. 2013. [Online]. Available: https://arxiv.org/abs/1306.0733

[39] Y. Chen *et al.*, "Emergence of interactive behaviors between two robots by prediction error minimization mechanism," in *Proc. Joint IEEE Int. Conf. Develop. Learn. Epigenet. Robot. (ICDL-EpiRob)*, Sep. 2016, pp. 302–307.

Kazuma Sasaki received the B.S., M.S., and D.E. degrees in engineering from Waseda University, Shinjuku, Japan, in 2013, 2015, and 2018, respectively.

Since 2018, he has been with DWANGO Company, Ltd., Tokyo, Japan, as an Engineer. His current research interests include human's drawing ability, sketch recognition systems, and autonomous picture generation systems.

Tetsuya Ogata received the B.S., M.S., and D.E. degrees in mechanical engineering from Waseda University, Shinjuku, Japan, in 1993, 1995, and 2000, respectively.

He was a Research Fellow of JSPS, Tokyo, Japan, a Research Associate with Waseda University, a Research Scientist with RIKEN Brain Science Institute, Wako, Japan, and an Associate Professor with Kyoto University, Kyoto, Japan. He is currently a Professor with the Faculty of Science and Engineering, Waseda University. Since 2017, he has been a Joint Appointed Fellow with the Artificial Intelligence Research Center, AIST, Warrendale, PA, USA. His current research interests include neural models for robots, dynamics of human–robot mutual adaptation, and intersensory translation.