http://www.icact.org/journal.asp

ICACT-TACT JOURNAL

Transactions on Advanced Communications Technology

Volume 7 Issue 6, November. 2018, ISSN: 2288-0003

GIRI
Global IT Research Institute

Journal Editorial Board

■ Editor-in-Chief

■ Editors

Editor Guide

■ Introduction for Editor or Reviewer

All the editor group members are to be assigned as a evaluator(editor or reviewer) to submitted journal papers at the discretion of the Editor-in-Chief. It will be informed by eMail with a Member Login ID and Password.

Once logined the Website via the Member Login menu in left as a evaluator, you can find out the paper assigned to you. You can evaluate it there. All the results of the evaluation are supposed to be shown in the Author Homepage in the real time manner. You can also enter the Author Homepage assigned to you by the Paper ID and the author's eMail address shown in your Evaluation Webpage. In the Author Homepage, you can communicate each other efficiently under the peer review policy. Please don't miss it!

All the editor group members are supposed to be candidates of a part of the editorial board, depending on their contribution which comes from history of ICACT TACT as an active evaluator. Because the main contribution comes from sincere paper reviewing role.

■ Role of the Editor

The editor's primary responsibilities are to conduct the peer review process, and check the final camera-ready manuscripts for any technical, grammatical or typographical errors.

As a member of the editorial board of the publication, the editor is responsible for ensuring that the publication maintains the highest quality while adhering to the publication policies and procedures of the ICACT TACT(Transactions on the Advanced Communications Technology).

For each paper that the editor-in-chief gets assigned, the Secretariat of ICACT Journal will send the editor an eMail requesting the review process of the paper.

The editor is responsible to make a decision on an "accept", "reject", or "revision" to the Editor-in-Chief via the Evaluation Webpage that can be shown in the Author Homepage also.

■ Deadlines for Regular Review

Editor-in-Chief will assign a evalaution group(a Editor and 2 reviewers) in a week upon receiving a completed Journal paper submission. Evaluators are given 2 weeks to review the paper. Editors are given a week to submit a recommendation to the Editor-in-Chief via the evaluation Webpage, once all or enough of the reviews have come in. In revision case, authors have a maximum of a month to submit their revised manuscripts. The deadlines for the regular review process are as follows:

Evalution Procedure	Deadline
Selection of Evaluation Group	1 week
Review processing	2 weeks
Editor's recommendation	1 week
Final Decision Noticing	1 week

■ Making Decisions on Manuscript

Editor will make a decision on the disposition of the manuscript, based on remarks of the reviewers. The editor's recommendation must be well justified and explained in detail. In cases where the revision is requested, these should be clearly indicated and explained. The editor must then promptly convey this decision to the author. The author may contact the editor if instructions regarding amendments to the manuscript are unclear. All these actions could be done via the evaluation system in this Website. The guidelines of decisions for publication are as follows:

Decision	Description
Accept	An accept decision means that an editor is accepting the paper with no further modifications. The paper will not be seen again by the editor or by the reviewers.
Reject	The manuscript is not suitable for the ICACT TACT publication.
Revision	The paper is conditionally accepted with some requirements. A revision means that the paper should go back to the original reviewers for a second round of reviews. We strongly discourage editors from making a decision based on their own review of the manuscript if a revision had been previously required.

■ Role of the Reviewer

Reviewer Webpage:

Once logined the Member Login menu in left, you can find out papers assigned to you. You can also login the Author Homepage assigned to you with the paper ID and author's eMail address. In there you can communicate each other via a Communication Channel Box.

Quick Review Required:

You are given 2 weeks for the first round of review and 1 week for the second round of review. You must agree that time is so important for the rapidly changing IT technologies and applications trend. Please respect the deadline. Authors undoubtedly appreciate your quick review.

Anonymity:

Do not identify yourself or your organization within the review text.

Review:

Reviewer will perform the paper review based on the main criteria provided below. Please provide detailed public comments for each criterion, also available to the author.

- How this manuscript advances this field of research and/or contributes something new to the literature?
- Relevance of this manuscript to the readers of TACT?
- Is the manuscript technically sound?
- Is the paper clearly written and well organized?
- Are all figures and tables appropriately provided and are their resolution good quality?
- Does the introduction state the objectives of the manuscript encouraging the reader to read on?
- Are the references relevant and complete?

Supply missing references:

Please supply any information that you think will be useful to the author in revision for enhancing quality of the paperor for convincing him/her of the mistakes.

Review Comments:

If you find any already known results related to the manuscript, please give references to earlier papers which contain these or similar results. If the reasoning is incorrect or ambiguous, please indicate specifically where and why. If you would like to suggest that the paper be rewritten, give specific suggestions regarding which parts of the paper should be deleted, added or modified, and please indicate how.

Journal Procedure

Dear Author,

➢ You can see all your paper information & progress.

➢ Step 1. Journal Full Paper Submission

Using the Submit button, submit your journal paper through ICACT Website, then you will get new paper ID of your journal, and send your journal Paper ID to the Secretariat@icact.org for the review and editorial processing. Once you got your Journal paper ID, never submit again! Journal Paper/CRF Template

➢ Step 2. Full Paper Review

Using the evaluation system in the ICACT Website, the editor, reviewer and author can communicate each other for the good quality publication. It may take about 1 month.

➢ Step 3. Acceptance Notification

It officially informs acceptance, revision, or reject of submitted full paper after the full paper review process.

Status	Action
Acceptance	Go to next Step.
Revision	Re-submit Full Paper within 1 month after Revision Notification.
Reject	Drop everything.

➢ Step 4. Payment Registration

So far it's free of charge in case of the journal promotion paper from the registered ICACT conference paper! But you have to regist it, because you need your Journal Paper Registration ID for submission of the final CRF manuscripts in the next step's process. Once you get your Registration ID, send it to Secretariat@icact.org for further process.

➢ Step 5. Camera Ready Form (CRF) Manuscripts Submission

After you have received the confirmation notice from secretariat of ICACT, and then you are allowed to submit the final CRF manuscripts in PDF file form, the full paper and the Copyright Transfer Agreement. Journal Paper Template, Copyright Form Template, BioAbstract Template,

Journal Submission Guide

All the Out-Standing ICACT conference papers have been invited to this "ICACT Transactions on the Advanced Communications Technology" Journal, and also welcome all the authors whose conference paper has been accepted by the ICACT Technical Program Committee, if you could extend new contents at least 30% more than pure content of your conference paper. Journal paper must be followed to ensure full compliance with the IEEE Journal Template Form attached on this page.

➢ How to submit your Journal paper and check the progress?

Step 1. Submit	Using the Submit button, submit your journal paper through ICACT Website, then you will get new paper ID of your journal, and send your journal Paper ID to the Secretariat@icact.org for the review and editorial processing. Once you got your Journal paper ID, never submit again! Using the Update button, you can change any information of journal paper related or upload new full journal paper.
Step 2. Confirm	Secretariat is supposed to confirm all the necessary conditions of your journal paper to make it ready to review. In case of promotion from the conference paper to Journal paper, send us all the .DOC(or Latex) files of your ICACT conference paper and journal paper to evaluate the difference of the pure contents in between at least 30% more to avoid the self replication violation under scrutiny. The pure content does not include any reference list, acknowledgement, Appendix and author biography information.
Step 3. Review	Upon completing the confirmation, it gets started the review process thru the Editor & Reviewer Guideline. Whenever you visit the Author Homepage, you can check the progress status of your paper there from start to end like this, " Confirm OK! -> Gets started the review process -> ...", in the Review Status column. Please don't miss it!

Volume. 7 Issue. 6

SvgAI – Training Methods Analysis of Artificial Intelligent Agent to use SVG Editor

Anh H. Dang*, Wataru Kameyama**

*GITS, Waseda University, Tokyo, Japan

** Faculty of Science and Engineering, Waseda University, Tokyo, Japan

danghoanganh@akane.waseda.jp, wataru@waseda.jp

***Abstract*— Deep reinforcement learning has been successfully used to train artificial intelligent (AI) agents, which outperforms humans in many tasks. The objective of this research is to train an AI agent to draw SVG images by using scalable vector graphic (SVG) editor with deep reinforcement learning, where the AI agent is to draw SVG images that are similar as much as possible to the given target raster images. In this paper, we propose framework to train the AI agent by value-function based Q-learning and policy-gradient based learning methods. With Q-learning based method, we find that it is crucial to distinguish the action space into two sets to apply a different exploration policy on each set during the training process. Evaluations show that our proposed dual ϵ-greedy exploration policy greatly stabilizes the training process and increases the accuracy of the AI agent. On the other hand, policy-gradient based training does not depend on external reward function. However, it is hard to implement especially in the environment with a large action space. To overcome this difficulty, we propose a strategy similar to the dynamic programming method to allow the agent to generate training samples by itself. In our evaluation, the highest score is archived by the agent trained by this proposed method. SVG images produced by the proposed AI agent have also superior quality compared to popular raster-to-SVG conversion softwares.**

***Keywords*—Reinforcement Learning, SVG, Exploration Policy, Q-learning**

I. INTRODUCTION

Besides defeating the world best human player in Go [1], AI agent (hereafter referred as agent) trained by deep reinforcement learning (RL) [2] has achieved human-level in a wide variety of tasks like playing 3D first-person shooter game [3], and enhances the capability of robotic automation [4]. For example, Mnih et. al. have introduced deep Q-network (DQN) [5] that plays Atari 2600 games well above the skill of human players and any other linear models. Subsequently, the works on prioritized experience replay [6], double Q-network [7], duel Q-network [8] and asynchronous actor-critic method [9] further enhance the efficiency of the training process.

On the subject of image understanding and raster-to-vector (R2V) conversion, Karpathy et. al. present a breakthrough work [10] on training a deep neural network (DNN) [11] for automatic image description. Beltramelli proposes Pix2Code DNN [12] that generates Extensible Markup Language (XML) based code from raster screenshot of graphical user interface (GUI).

Despite being a mature branch of research, image-processing based R2V conversion is not yet reliable [13]. Therefore, we propose a framework to train an agent to use SVG editor (hereafter referred as editor) with RL. The objective of this agent is to draw an SVG image that is similar as much as possible to a given target raster image. It can be considered as a new paradigm to solve the R2V problem.

In this paper, we concentrate on exploring the feasibility of this new paradigm by training the agent to work on randomly generated target images. A custom editor is created for carrying out the research, which has modeled after OpenAI Gym [14] environment due to its robustness in interface design.

We train the agent by using both Q-leaning based and policy-gradient based methods. We evaluate the agent performance by comparing the similarity between generated SVG images and target images. Finally, we compare the quality between the SVG image produced by the proposed agent with that produced by popular R2V softwares.

This paper is organized as follows: Section II describes related works on R2V and RL. Section III describes our proposal for both the agent and the environment design. Section IV describes the agent model. Then, Section V describes training and evaluation process in detail. Section VI concludes the paper with possible improvement and future works.

II. RELATED WORKS

The agent needs to be trained to ultimately convert a raster image to an equivalent vector representation by using the editor. Therefore, this research is related to not only RL but also a series of visual-vector cross model works. There are two categories of works related to this paper: image-processing based R2V and deep learning based R2V conversions.

A. Image-Processing Based R2V conversion

As mentioned in Section I, image-processing based R2V conversion is not yet reliable [13]. Major problems include difficulties of color quantization, aliasing effects, shift,

Manuscript received on Dec. 31, 2017. This work is a follow-up of the invited journal to the accepted & presented paper of the 20th International Conference on Advanced Communication Technology (ICACT2018).

Anh H. Dang is with Graduate School of Global Information and Telecommunication Studies, Waseda University, Tokyo, Japan (corresponding author, phone: +81-80-1367-9637, email: danghoanganh@akane.waseda.jp)

Wataru Kameyama is with School of Science and Engineering, Waseda University, Tokyo, Japan (email: wataru@waseda.jp)

superposition effects, and miss-identification of texture and text [15].

There are numerous works to try to solve the above-mentioned problems. For example, Kansal et. al. propose a framework to reproduce linear filled gradient [16]. Vector representation of halftone dots in binary images is presented by Kawamura et. al. [17]. However, they leave the work of identifying the type of problems to the human operator. Thus, for example, well-known conversion tools such as Potrace [18] still require human's intervention to achieve desirable results [19].

B. Deep Learning Based R2V conversion

Image annotation has been an active research area. It becomes overly crowded recently due to the advancement of deep learning based natural language processing and computer vision. Karpathy et. al. achieve a breakthrough with an end-to-end DNN model [10]. Learning only from images and corresponding annotations, this model not only recognizes and locates objects, but also annotates images at different hierarchical levels. The model is realized by the embedding visual model using RCNN (Regional Convolutional Neural Network) [20] and the language model using BRNN (Bi-directional Recurrent Neural Network) [21] into the same multimodal space.

Beltramelli proposes Pix2Code [12], an end-to-end DNN that generates XML based GUI code from mock-up images. Even though similar to [10] in general design, the visual model used in Pix2Code is a plain CNN block while language model is handled by a Long-Short Term Memory (LSTM) network [22] block. Another LSTM block is used to decode the network's output into code tokens. This work can be understood as a rigid version of automated image annotation. However, the model is not flexible because the visual presentations of all the GUI elements in this work are templated. Thus, it only works with GUI images based on the fixed templates.

C. Reinforcement Learning

The agent in deep RL holds a policy set that ultimately decides which action to be performed in the next step. At each state of time step t, the agent observes the state s_t of the environment, and decides action a_t based on its current policy π. Reward r_t is then given to the agent as the feedback from the environment. The objective of the training process is to train the agent so that its policy will result in maximized reward in the future. The expected reward is calculated as follows.

$$R_t = \sum_{i=t}^{T} \gamma^{i-t}\, r_{\mathrm{i}} \tag{1}$$

Where R_t is the sum of expected discounted reward at time step t, T is the time when the episode is terminated, and γ is the discount factor which lies within the range of [0 ... 1]. The higher the value is, the more important the future reward is.

1) Q-learning: In Q-learning, R_t is approximated by the Q function. This function returns an action-state value according to policy π as follows:

$$Q^{\pi}(s,a) = \mathbb{E}[R_t | s_t = s, a_t = a] \tag{2}$$

Hence, $Q^{\pi}(s,a)$ is essentially an expected future reward if the agent performs action a in the given state s. In practice, the action that gives the highest Q value is chosen to be executed by the agent. The optimal value $Q^*(s,a)$ is defined as follows:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) = \max_{\pi} \mathbb{E}[R_t | s_t = s, a_t = a] \tag{3}$$

Hence, using (1) and (2), equation (3) can be written as follows:

$$Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} \llbracket Q^*(s',a') | s,a \rrbracket\right] \tag{4}$$

Where s' and a' are the state and the action of the subsequent time step, respectively, and r is the immediate reward for action a given in state s. In deep Q-learning, this value is approximated by the neural network parameterized by θ:

$$Q_{\theta}(s,a) \approx Q^*(s,a) \tag{5}$$

Thus, with a trained agent, the Q value of an action a can be estimated as follows:

$$Q_{\theta}(s,a) \approx r_t + \gamma \max_{a'} Q_{\theta}(s',a') \tag{6}$$

Given that $y_t = r_t + \gamma \max_{a'} Q_{\theta}(s',a')$, the loss function is defined as follows:

$$L_t(\theta_t) = \mathbb{E}_{s,a,r,s'}\left[\left|y_t - Q_{\theta_t}(s,a)\right|^2\right] \tag{7}$$

In practice, instead of squared difference loss, Huber loss is usually used to stabilize the training process:

$$L(\alpha) = \begin{cases} \dfrac{1}{2}\alpha^2 & \text{if } |\alpha| \le \delta \\ \sigma\left(|\alpha| - \dfrac{1}{2}\delta\right) & \text{otherwise} \end{cases} \tag{8}$$

Where α is the difference between y_t and $Q_{\theta_t}(s,a)$, and δ is the point where the loss function change from quadratic to linear.

2) Policy-Gradient: Agents trained by the Q-learning method predict the state-action value as in formula (6), then the action is chosen deterministically based on this value. Thus, it heavily depends on the value function to result in better policy approximation. On the contrary, with policy-gradient based training, the agent is trained to output the action probability directly from a given state. Formally, policy-gradient optimizes policy θ to maximize the expected discount return R_t:

$$\theta = \operatorname*{argmax}_{\theta} \mathbb{E}[R_t] \tag{9}$$

To optimize the policy θ, the gradient of policy θ is given by:

$$\nabla_{\theta}\mathbb{E}[R_t] = \mathbb{E}[\nabla_{\theta} \log P(a_t) R_t] \tag{10}$$

Where $P(a_t)$ is the probability of action a at time step t. Thus, actions that lead to better expected reward R_t are encouraged. In order to train an agent using policy-gradient, R_t must be known or has to be approximated.

3) Experience Memory Replay: One of the most significant difficulties in training an agent is the strong correlation between the network policy and the action outcome of subsequent time steps. This difficulty makes online training impossible. To break the strong correlation, experience memory replay [6] is used.

In the experience memory replay, at every time step t, the experience (s_t, a_t, r_t, s_{t+1}) of the agent is stored in replay memory which is a large capacity queue. In popular works like DQN, it is usually set to store one million experiences. When it's full, the oldest experience in the queue is removed to make place for a new experience. The network is trained by using mini batches randomly drawn from this memory.

4) Exploration in RL: Training an agent using RL requires a right balance between exploitation and exploration. Exploitation is relying on the learned policy to improve the prediction accuracy while exploration allows the agent to seek for better potential solutions (i.e. avoiding sub-optimal trap). A popular exploration policy being used in RL is ϵ-greedy [2]. Under this policy, the output of the agent has an ϵ chance of being random.

There are other exploration policies based on randomization, such as Thompson sampling [23] and Bayesian sampling [24]. However, a variance of ϵ-greedy policy named reducing ϵ-greedy is most commonly used. With this policy, the agent starts with high exploration rate, and gradually reduces it throughout the training process. Thus, it allows the agent to explore in the beginning, and to focus more on exploitation in the later phase of the training process. In this paper, we evaluate the reducing ϵ-greedy variances only. All ϵ-greedy policies mentioned hereafter refer to reducing ϵ-greedy exploration policy.

III. SVG Editor

As seen in common in AI settings, the proposed framework consists of two parts: the agent and the environment. As shown in Fig. 1, the editor is playing the role of the environment in this research. For every time step, the agent observes the state of the editor (step 1). Then, the agent processes the observed state and sends a new action to the editor (step 2). So, the editor executes the action as requested and sends reward back to the agent (step 3). And the process goes back to step 1.

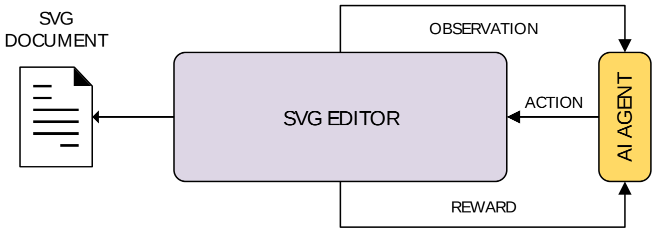


Fig. 1. The proposed framework of SvgAI. The editor is playing the role of the environment.

A. SVG Construction

SVG is an XML based vector image format. An SVG document is composed of different types of elements, such as path, circle, and rectangle. Previous works on R2V conversion usually produce SVG image using path element exclusively because this element is flexible and can be used to form any type of shape. Furthermore, using fundamental shape elements, such as rectangle, circle and arc, not only requires the higher level of visual understanding but also comes with none trivial challenges.

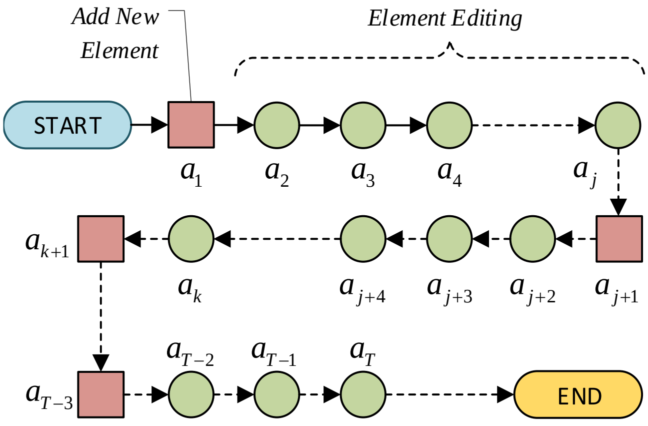


Fig. 2. The process of SVG composition: the agent keeps adding and editing new elements until the desired result is achieved. a_i is action given by the agent at time step t. T is the last time step in the episode.

On the other hand, using a large number of elements and paths significantly increases the SVG image size. Thus, the output SVG requires much more computational resource to be displayed. In this research, we utilize two fundamental shape elements (and with their transformations) that are square element and circle element for SVG image construction.

Fig. 2 shows the process of SVG composition, where every episode starts with a blank canvas. During the process, the agent is either adding a new element into the working image or editing the most recently added element.

The newly added element has a default presentation when it's firstly added. The default presentations of the circle and square elements are shown in sub-figure (a) and (g) of Fig. 5. As the agent keeps editing, the presentation of the element is to be updated. For example, subfigure (l) in Fig. 5 is the presentation of a circle element after 400 steps of editing.

B. Action Space

With the above-explained process, once a new element is added to the working SVG document, the old element is no longer editable. Thus, the consequence of adding new elements is more significant compared to editing them. Therefore, for Q-learning based training, it is crucial to distinguish the two sets of actions, set A and set B, and to apply separated exploration policies during the training process. Otherwise, it is impossible for the model to converge. Table I lists all the actions of set A and B supported by the editor. Set A consists of actions that add new element into the working SVG document, while set B consists of editing actions, i.e. element-shape manipulation actions.

TABLE I
List of Actions Supported By the Editor

Set	Action Id.	Description
A	0, 1	Add circle/square element
B	2, 3, 4, 5	Move element left/right/up/down
	6, 7, 8, 9	Compress/expand element horizontally/vertically
	10, 11	Rotate the element clockwise/counter clockwise
	12, 13	Reduce/increase element's line thickness
	14, .., 21	Adjust[a] element's line color (RGBα)
	22, .., 29	Adjust[a] element's fill color (RGBα)

[a] Increase/decrease value of each channel

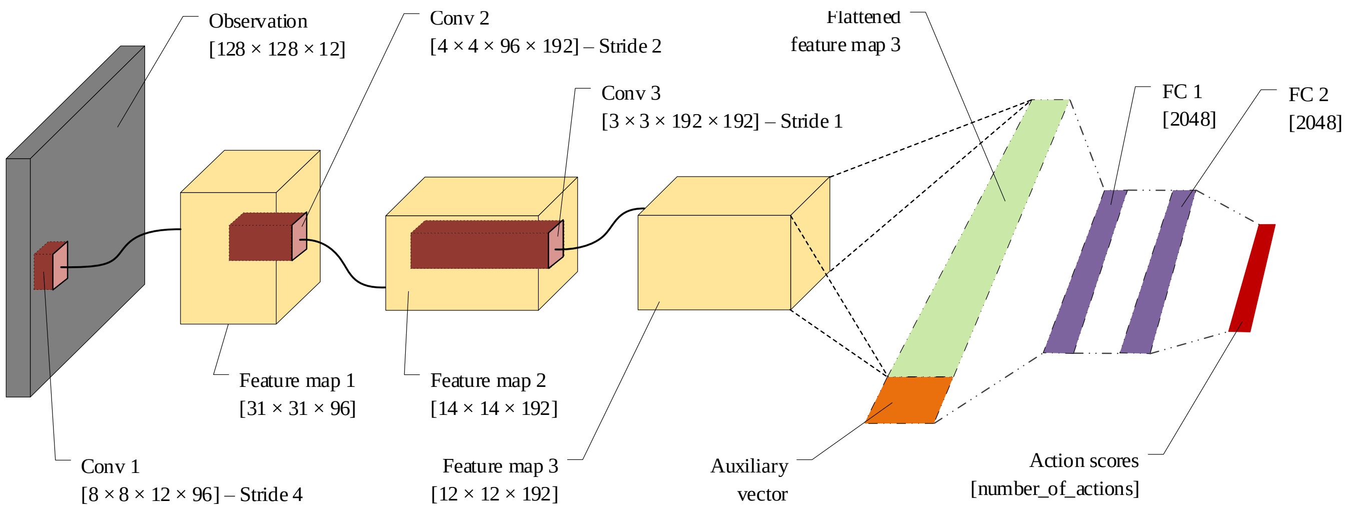


Fig. 3. The architechture of the agent. The stacked images explained in Secion III-C is processed by the convolution layers. Auxilary vector is concatenated with the output of the convolution layers before feeding to fully connected blocks at the end of the network.

C. State Observation

The state of the editor at any time step t consists of four components:

1. Raster version of the image being edited I_t.
2. Raster target image Y, i.e. the image which the agent attempts to compose by using the editor.
3. Raster image of the element being editted in I_t (component 1). Thus, , this component is exactly the same to I_t when only one element is in I_t.
4. An auxiliary vector that describes the state of the SVG element being edited (component 3). This auxiliary vector consist of paremeters such as orientation, position, line thickness, line color, and fill color.

All raster images are in the RGB α format with the resolution of 128 by 128 pixels. The first three component are stacked in channels dimention to form the image stack.

IV. Model

A. Network Architecture

Fig. 3 shows the architecture of the proposed agent. The above mentioned image stack is processed by the convolutional neural network (CNN) blocks. These blocks produce a feature map with around 27k parameters. The combination of this feature map and the auxiliary vector is then processed by a fully connected block to produce a score or probability for each action supported by the environment.

B. Error and Reward

With every action received from the agent, the score is calculated as follows:

$$g_t = \exp\left(-\frac{|I_t - Y|^2}{\sigma^2}\right) \tag{11}$$

Where g_t is the score at time step t, which describes the similarity between the rasterized version I_t of the working SVG document at time step t and the target image Y, and σ is the scaling factor. Thus, the domain of the score g_t is from 0 to 1 where 1 means perfect matching.

Base on the score g_t, the reward is given to the agent as follows:

$$r_t = \begin{cases} 1 + g_t & \mathit{if}\ g_t > g_{t-1} \\ 0 & \mathit{otherwise} \end{cases} \tag{12}$$

Where r_t is the immediate reward at time step t. Thus, the environment returns 0 reward when the action results in no improvement, and returns a small reward ranging from 1 to 2 depending on the similarity between the result and the target image.

Given v_t which is the number of SVG elements at time step t, the penalty p_t is given as follows:

$$p_t = \begin{cases} -1 & \mathit{if}\ v_t > v_{t-1} \\ 0 & \mathit{otherwise} \end{cases} \tag{13}$$

Hence, the discounted reward R_t at time step t is calculated as follows:

$$R_t = \sum_{i=t}^{T} (r_i + p_i)\gamma^{i-t} \tag{14}$$

Where γ is the discount factor and T is the length of the episode. By giving a penalty for adding elements more than necessary, the agent is discouraged to perform actions in set A. However, this technique is only feasible with a controlled environment in which the structure of a state is known. The training process is expected to be slower without this penalty.

C. Q-learning and Exploration Policy

In our early attempts, the conventional ϵ-greedy policy was used. On average, with the action sets exlained in Section III-B, there was a new element added into the image for every 14 adjustments during the exploration process. It means, regardless the value set to the exploration rate ϵ, actions in set A were explored more than necessary. This bad exploration is most likely making the network to be trapped in sub-optimal solution [25].

The simplest solution is to apply lower weight for the actions in set A during the random exploration process. The probability for an action being performed randomly by the ϵ-greedy policy is:

$$p(a_i) = \begin{cases} \epsilon\omega_a & \text{if } a_i \in \text{A} \\ \epsilon\omega_b & \text{if } a_i \in \text{B} \end{cases} \tag{15}$$

Where ω_A and ω_B are weights for the actions of set A and B. However, we observe that the agent trained by using this weighted ϵ-greedy policy often inserts incorrect element into the working image. A possible explanation for this problem is that the ϵ value is already saturated when the agent does not learn yet the long-term reward of adding the right element.

Prolonging the exploration phase does not improve the result because the agent has to keep exploring more on element editing (i.e. action set B) in order to discover better solution. Our approach to solve this problem is to apply the different reducing ϵ-greedy policies (hereafter referred as dual ϵ-greedy policy) on each action set. In this setting, there are two independent reducing ϵ-greedy polices applied for each action set:

$$p(a_i) = \begin{cases} \epsilon_a \omega_a & \text{if } a_i \in \mathrm{A} \\ \epsilon_b \omega_b & \text{if } a_i \in \mathrm{B} \end{cases} \quad (16)$$

Where ϵ_A and ϵ_B are weight values of two ϵ-greedy policies for the action set A and B, respectively. In this way, the action set A and B can be independently explored.

Algorithm 1 shows the pseudo code to train the agent using Q-learning paradigm with dual ϵ-greedy policy, where the random function has several forms (analogous to C++'s function overloading). The function Rnd(A) and Rnd(B) randomly select an action in set A and B. Rnd(X | W) picks element from X with weight W. The difficulty k is the minimum number of steps to compose the target image from a blank canvas as explained in Section IV-D.

D. Policy-Gradient

As mentioned in Section II-C-2, in order to train the agent using policy-gradient, R_t has to be known. This value can be approximated for training. Thus, it makes the efficiency of the training process again depend on an external function. Another solution is to fully unroll the episodes. Monte Carlo method is commonly used for this purpose. However, this method is intractable in the environment with a large action space, as it requires an immerse computational resource.

To overcome this difficulty, we unroll the episode and train the agent reversively. As shown in Fig. 4, supposing that k is the minimum number of steps to compose the target image from a blank canvas, if an agent is trained to perform last d step from time step t, action at time step $t-1$ can be unrolled in a brute force way:

$$P(a_i \mid s_{t-1}) = \frac{\left[D\left(s_{t|a_i}, Y\right) = \mathrm{d}\right]}{\sum_j \left[D\left(s_{t|a_j}, Y\right) = \mathrm{d}\right]} \quad (17)$$

Where $P(a_i \mid s_{t-1})$ is the probability of action a_i given the state s_{t-1} at time step $t-1$, $D\left(s_{t|a_i}, Y\right)$ is the minimum number of steps for the agent to finish the episode given in the state $s_{t|a_i}$ at time step t which is the result from action a_i from the last time step.

ALGORITHM 1. Q-LEARNING BASED TRAINING ALGORITHM

```
inputs:    Maximum time step $t_{max}$
           Number of actions $n$
output:    Optimized policy $\theta$
variables:    $M$ experience memory
              $L$ training interval
              $t$ time step
              $gt$ global counter

while not converged
    $Y \leftarrow$ random target image $Y$ with random $k$
    for $t = 0$ to $t_{max}$
        $gt \leftarrow gt + 1$
        $a_A \leftarrow \mathrm{Rnd(A)}$ ; $a_B \leftarrow \mathrm{Rnd(B)}$
        $a_Q \leftarrow \max_a \left(Q_\theta(s_t, a)\right)$
        $a_t \leftarrow \mathrm{Rnd}\left(a_A, a_B, a_Q \mid p(a_A), p(a_B), p(a_Q)\right)$
        unroll episode with action $a_t$
        add $episode$ to $M$
        if $gt$ mod $L = 0$
            train mini batch taken from $M$
        if $episode$ end
            break
```

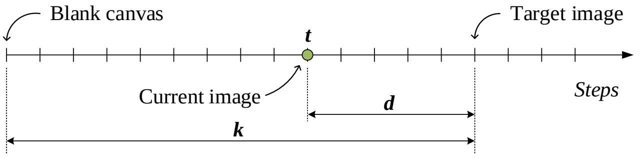


Fig. 4. The minimum number of steps to compose target image from a blank canvas is k. The minimum number of steps to compose target image from the current image is d.

Algorithm 2 shows the pseudo code of the policy-gradient based training. The agent is trained to work on the incrementally difficult states. The difficulty of a state is measured by distance d. Once the policy network θ_d for distance d is converged, it is then trained with more difficult distance $d+1$. $U(\cdot)$ is an uniform sampling function.

The primary disadvantage of this method is that it is only feasible with a controlled environment where the difficulty of the state can be calculated. However, it is useful in combination with techniques such as transfer learning when applying it for more complex data.

V. EVALUATION

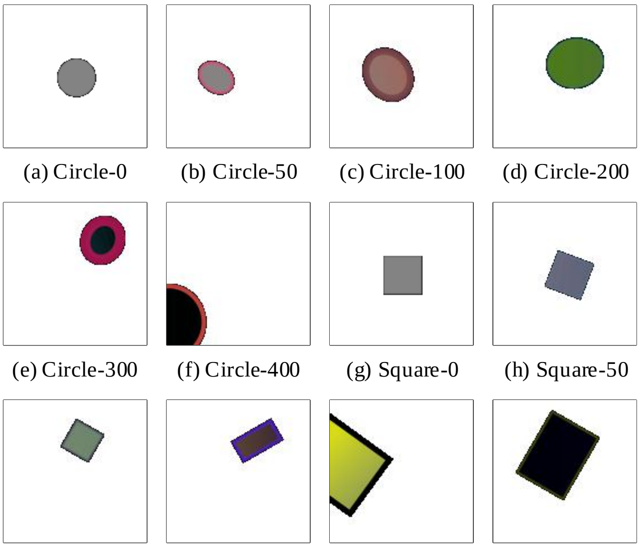
A. Dataset

Works on image-processing based R2V conversion have many diverse objectives. As the result, they mostly use relatively plain and small datasets. The diversity in research objectives also leads to the lack of unified evaluation dataset [15]. While the complexity of these data sets is suitable for this research, their modest size and heterogeneous properties are not adequate to be used to train DNN.

ALGORITHM 2. POLICY-GRADIENT BASED TRAINING ALGORITHM

```
inputs:    Upper limit $k_{max}$
           Number of actions $n$
output:    Optimized policy $\theta$
variables:    $M$ experience memory
              $gt$ global counter
              $L$ training interval

for $d = 1$ to $k_{max}$
    while $\theta_d$ is not converged:
        $k \leftarrow U([d..k_{max}])$
        $t \leftarrow k - d$
        with random $episode$ at difficulty $k$
            $Y \leftarrow$ last state of $episode$
            $s_{t-1} \leftarrow$ state at time step $t-1$ of $episode$
            for $i = 0$ to $n$
                if $D\left(s_{t|a_i}, Y\right) = d$:
                    add $(s_{t-1}, a_i)$ to $M$
                    $gt \leftarrow gt + 1$
                    if $gt$ mod $L = 0$
                        train mini batch taken from $M$
```



(a) Circle-0 (b) Circle-50 (c) Circle-100 (d) Circle-200

(e) Circle-300 (f) Circle-400 (g) Square-0 (h) Square-50

(i) Square-100 (g) Square-200 (k) Square-300 (l) Square-400

Fig. 5. Examples of images generated by training and evaluation processes. The caption under each image shows the name of the element and the minimum number of steps (k as in Fig. 4) required to compose that image from a blank canvas by using the editor.

With the above reasons, the training and the evaluation of the agent have been done on a randomly generated data set. It not only helps in avoiding the above-mentioned problems, but also provides a controlled level of difficulty and the uniformity of the dataset. Target images are created by rasterizing the randomly generated SVG documents. These documents contain a single shape element with difficulty k. Fig. 5 shows examples of target images in format <*element*>-<*k*>.

B. Episode Termination

The episode is terminated if one of the following conditions is met:

- After a fixed time step t_{max}. Since there is no further step after the termination, for Q-learning, the Q value as mentioned in (6) at the final step t_{max} is: $Q_\theta(s_{t_{max}}, a_{t_{max}}) = r_{t_{max}}$.
- When g_t in (11) is greater than a certain threshold.

C. Frame Skip

Due to the computational demand of the agent, it is inefficient to let the agent perform an action at every time step. The popular solution is the frame skip [14] where the agent only interacts with the environment in every r steps. Therefore, once an action is decided by agent, it repeats r-time steps. A popular value of r in many tasks is 4 as it is usually a good tradeoff between the performance and the training speed.

In this research, dismissing frame skip helps to improve the performance of the agent because one action repeated several times makes the agent miss its target due to overshooting.

TABLE II
HYPER-PARAMETER SETTING FOR EXPERIMENTS

		Conventional	Weighted	Dual
Minibatch size		32	32	32
Upper limit k_{max}		400	400	400
Train Interval L		16	16	16
Discount factor γ		0.999	0.999	0.999
Memory replay size		1e6	1e6	1e6
ϵ-greedy	max - min	1 – 0.1	1 – 0.1	-
	start - end	0 – 1e6	0 – 1e6	-
ϵ_A-greedy	max - min	-	-	1e-3 – 0
	start - end	-	-	5e6 – 10e
ϵ_B-greedy	max - min	-	-	0.999 – 0.
	start - end	-	-	0 – 1e6
W_A		1	1e-3	1
W_B		1	999e-3	1

D. Parameter Update

The agent does not update its parameter at every time step, but once for every L experiences added in the replay memory. Thus, given the batch size of N, one experience is learned by the agent N/L times in average.

E. Settings

We have implemented the agent using the model proposed in Section IV. We evaluate the performance of the agent trained by different training schemes:

- Policy-gradient
- Q-learning under different exploration policies
  - Conventional ϵ-greedy policy.
  - Weighted ϵ-greedy policy.
  - Dual ϵ-greedy policy as.

Table II shows the hyperparameters used for the experiments. Policy-gradient training experiments share the first 5 parameters with other experiments.

F. Performance

Fig. 6 shows the distribution of the evaluation scores of the agent under different training schemes. Similar to the training configuration, the target images used for evaluation are generated with random difficulty k given that $k \leq 400$. The box plot describes the distribution of evaluation scores by 5×10^4 iterations interval from 0 to 40×10^4. For each scheme, we train the agent 5 times. Each time, we evaluate the agent after every 100 training iteration and collect evaluation score calculated using equation (11). Thus, each box describes the distribution of 500 evaluation scores. The colored box indicates IQR (interquartile range). The horizontal line within the box indicates the median score, and the extended bar indicates the maximum and minimum scores. Dots indicate outliers. Our proposed dual ϵ-greedy policy not only shows significant performance gain but also highly stable compared to conventional ϵ-greedy policy and weighted ϵ-greedy policy during the training process. The agent trained by policy-gradient achieves the best result and the high stability when the number of iterations is more than or equal to 15×10^4.

G. Accuracy

With dual ϵ-greedy policy, our trained agent favors adding circle element and achieves higher score by editing circle element in general. To further analyze this observation, we evaluate each trained agents for 500 episodes with the same setting used in Section V-F and drill down the evaluation results. In general, evaluation results can be divided into two sets:

- Circle set: consists of episodes with target images that contain circle elements only (250 episodes for each agent).
- Square set: consists of episodes with target images that contain the square elements only (250 episodes for each agent).

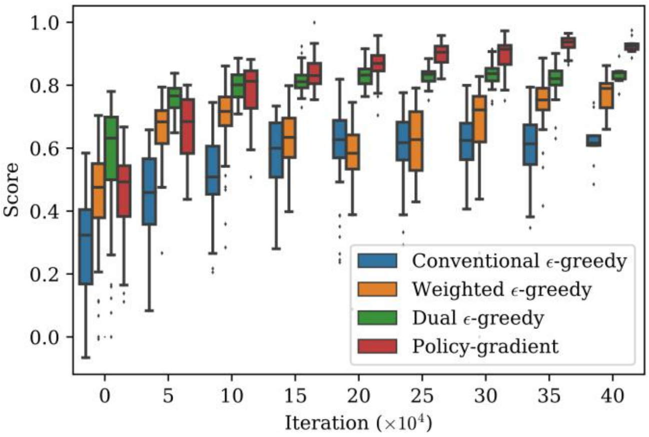


Fig. 6. Evaluation score distribution of the agent throughout the training process under different training scheme.

With each set, we analyze the performance of the agent on action set A and set B separately. The agent's performance on action set A is measured by the number of episodes where a correct shape element inserted into the working SVG document over the total number of episodes in the set. Correctly insterting a shape is important because further editing an incorrect element results in a large number of bad experience that negatively affects the agent's policy network. Table III shows the performance on action set A of the agents trained under different training schemes.

TABLE III
AGENT PERFORMANCE ON ACTION SET A

	Circle set	Square set
Policy-gradient	**0.99**	**0.95**
Q-learning		
Dual ϵ-greedy	0.94	0.76
Weighted ϵ-greedy	0.67	0.58
Conventional ϵ-greedy	0.53	0.61

To evaluate the agent's performance on action set B. We again evaluate each agent for 500 episodes with an element already inserted. The evaluation results are divided into two sets:

- Set 1: consists of episodes with target images that contain the same element with the pre-inserted element (250 episodes each agent).
- Set 2: consists of episodes with target images that contain an element that is different from the pre-inserted element (250 episodes for each agent).

Table IV shows the agent's performance on action set B. The evaluation scores of set 1 are in the grey background. For this set, the performance of agents trained by policy-gradient are better than the agents trained by Q-learning.

The evaluation scores of set 2 are in white background. Interestingly, for set 2, even with the wrong element pre-inserted, the performance of all the trained agents are over 0.5. This reflects the fact that all the agents are trained with a sustainable amount of episodes in which the wrong element is inserted. This result is also correlated to action set A performance shown in Table II: Because the agents trained by policy-gradient have high accuracy for action set A, they rarely experience the training episodes where the wrong element is added. Thus, on set 2, their performance is even lower than the performance of agents trained by dual ϵ-greedy policy (bolded italic v.s. italic on Table IV).

TABLE IV
AGENT PERFORMANCE ON ACTION SET B

Agent		Pre-insterted	Target Image	
			Circle	Square
Policy-Gradient		Circle	**0.97**	*0.65*
		Square	*0.71*	**0.94**
Q-learning	Dual ϵ-greedy	Circle	0.93	***0.78***
		Square	***0.72***	0.89
	Weighted ϵ-greedy	Circle	0.81	0.76
		Square	0.67	0.79
	Conventional ϵ-greedy	Circle	0.59	0.66
		Square	0.61	0.73

H. SVG Quality

We visually compare the SVG image produced by our trained agent with two popular opensource and commercial R2V solutions: Potrace and AutoTrace [26]. Fig. 7 shows the comparison between the outputs. As shown in the figure, Potrace not only has a problem on color quantization but also results in distorted circle. On the other hand, AutoTrace produces a much better result, however, the linear gradient fill has been converted into color blobs. Without any manual configurations prior to the conversion/drawing process, our agent produces significantly better result.

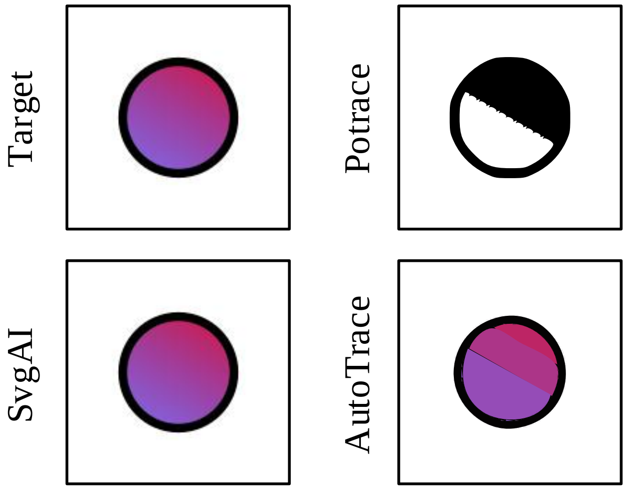


Fig. 7. Comparision between SVG images produced by our agent (SvgAI), Portrace and AutoTrace. SvgAI trained by Q-learning and policy-gradient produces identical result for this example.

Not only visually better, but the SVG images produced by our agent are also smaller both in file size and node counts. As shown in Table V, SVG images produced by our agent are 40% smaller in size and 63% smaller in node counts than the second best solution in average over 100 images where each one contains a single element, i.e. a circle or a square.

TABLE V
AVERAGE SVG SIZE COMPARISON

	Target	SvgAI	Potrace	AutoTrace
File Size	5.8KB	**852B**	1.4KB	15.5KB
Node Count	-	**1.4**	2.7	174
Color	≥ 16mil	≥ 16mil	2	10

VI. CONCLUSION

In this paper, we introduce a new paradigm to solve the R2V conversion problem. We propose an agent model and a framework to train the agent to use SVG editor by using Q-learning and policy-gradient. In order to successfully train the agent by using Q-learning, we divide the action space into two sets and apply independent exploration policies on each action set. Evaluation results show the efficiency of the proposed dual ϵ-greedy policy and policy-gradient. The SVG image quality produced by our agent is also superior compared to the popular software solutions. The problem of incorrect shape detection as explained in Section V-G shows a weakness of Q-learning in applying it to this problem. While policy-gradient produces the most efficient agent, it can only be used for managed environments. For future works, we will investigate and improve the training process for better reward back-propagation and for better shape detection accuracy.

REFERENCES

[1] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1, no. 1. MIT press Cambridge, 1998.

[3] G. Lample and D. S. Chaplot, "Playing FPS Games with Deep Reinforcement Learning.," in *AAAI*, 2017, pp. 2140–2146.

[4] Lin and Long-Ji, "Reinforcement learning for robots using neural networks." Carnegie Mellon University, 1992.

[5] V. Mnih *et al.*, "Human-level control through deep reinforcement learning.," *Nature*, vol. 518, no. 7540, pp. 529–33, Feb. 2015.

[6] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," Nov. 2015.

[7] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, pp. 2094–2100, 2016.

[8] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. JMLR.org, pp. 1995–2003, 2016.

[9] V. Mnih *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, vol. 48, pp. 1928–1937.

[10] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 664–676, Apr. 2017.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[12] T. Beltramelli, "pix2code: Generating Code from a Graphical User Interface Screenshot," May 2017.

[13] H. S. M. Al-Khaffaf, A. Z. Talib, and R. A. Salam, "Empirical performance evaluation of raster-to-vector conversion methods: A study on multi-level interactions between different factors," *IEICE Trans. Inf. Syst.*, vol. 94, no. 6, pp. 1278–1288, 2011.

[14] G. Brockman *et al.*, "OpenAI Gym," Jun. 2016.

[15] V. Lacroix, "Raster-to-Vector Conversion: Problems and Tools Towards a Solution A Map Segmentation Application," in *2009 Seventh International Conference on Advances in Pattern Recognition*, 2009, pp. 318–321.

[16] R. Kansal and S. Kumar, "A framework for detection of linear gradient filled regions and their reconstruction for vector graphics," 2013.

[17] K. Kawamura, H. Watanabe, and H. Tominaga, "Vector representation of binary images containing halftone dots," in *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)*, pp. 335–338.

[18] P. Selinger and P. Selinger, "Potrace: a polygon-based tracing algorithm," *IN HTTP://POTRACE.SOURCEFORGE.NET*, 2003.

[19] X. Hilaire and K. Tombre, "Robust and accurate vectorization of line drawings," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 890–904, Jun. 2006.

[20] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[21] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[22] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[23] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[24] J. Asmuth, L. Li, M. L. Littman, A. Nouri, and D. Wingate, "A Bayesian sampling approach to exploration in reinforcement learning," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 19–26.

[25] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[26] M. Weber, "Autotrace-converts bitmap to vector graphics." 2004.

Anh H. Dang (S'09) received his bachelor degree in business administration, information & communication technology from Ritsumeikan Asia Pacific University (Beppu, Oita, Japan) in 2010. He then received the master degree in computer science from Waseda University (Shinjuku, Tokyo, Japan) in 2012. Since 2012, he is a Ph.D. candidate at Waseda University. He is a member of IEEE, ACM, and IEICE. His research interests are machine learning, artificial intelligence, and computer vision.

Prof. Wataru Kameyama (M'86) received the bachelor's, master's, and D.Eng. degrees from the School of Science and Engineering, Waseda University, in 1985, 1987, and 1990, respectively. He joined ASCII Corporation in 1992, and was transferred to France Telecom CCETT from 1994 to 1996 for his secondment. After joining Waseda University as an Associate Professor in 1999, he has been a Professor with the Department of Communications and Computer Engineering, School of Fundamental Science and Engineering, Waseda University, since 2014. He has been involved in MPEG, MHEG, DAVIC, and the TV-Anytime Forum activities. He was a Chairman of ISO/IEC JTC1/SC29/WG12, and a Secretariat and Vice Chairman of the TV-Anytime Forum. He is a member of IEICE, IPSJ, ITE, IIEEJ, and ACM. He received the Best Paper Award of Niwa-Takayanagi in 2006, the Best Author Award of Niwa-Takayanagi in 2009 from the Institute of Image Information and Television Engineers, and the International Cooperation Award from the ITU Association of Japan in 2012.

Cognitive Workload Detection from Raw EEG-Signals of Vehicle Driver using Deep Learning

Mohammad A. Almogbel*, Anh H. Dang**, Wataru Kameyama***

*Department. of Computer Science and Communications Engineering,
Graduate School of Fundamental Science and Engineering, Waseda University, Tokyo, Japan.

**GITS, Waseda University, Tokyo, Japan

***Faculty of Science and Engineering, Waseda University, Tokyo, Japan

almogbela@ruri.waseda.jp, danghoanganh@akane.waseda.jp, wataru@waseda.jp

Abstract—**Electroencephalography (EEG) signals have been proven to be effective in evaluating human's cognitive state under specific tasks. Conventional classification models utilized for EEG classification heavily rely on signal pre-processing and hand-designed features. In this paper, we propose an end-to-end deep neural network which is capable of classifying multiple types of cognitive workload of a vehicle driver and the context of driving using only raw EEG signals as its input without any pre-processing nor the need for conventional hand-designed features. Data used in this study are collected throughout multiple driving sessions conducted on a high-fidelity driving simulator. Experimental results conducted on 4 channels of raw EEG data show that the proposed model is capable of accurately detecting the cognitive workload of a driver and the context of driving.**

Keywords— **Deep Learning, EEG, Neural Networks, Cognitive Workload, Driving, Stress**

I. Introduction

Maintaining safe levels of cognitive workload is extremely crucial to ensure optimal performance and attention whilst driving automobiles. Different methods have been widely investigated to monitor driver's cognitive workload including heart rate monitoring [1][2], galvanic skin response [3][4], facial expression [5][6] and so on [7][8][9][10]. One of the most popular measures for assessing the mental state of humans is electroencephalography (EEG) signals. EEG has reportedly shown success in evaluating the mental state such as drowsiness [11], mind wandering [12] and alertness [13].

Despite being an active research area, understanding and applying EEG signals are still limited due to the lack of true understanding of the brain's activities which prevents use of good EEG features from being engineered. Besides, EEG signals are usually strongly affected by noise and interference. Accurate reading requires a delicate equipment and sealing.

Conventional analyzing approaches make extensive use of Fourier transform in order to decompose EEG signal into multiple frequencies. Only well-known frequency is then used for feature design process. For example, the alpha band (8 to 15[Hz]) is correlated with relaxing, and the beta band (16 to 31[Hz]) associates with mental stress. Pre-processing methods such as Butterworth bandpass filter and stationary wavelet transform filters [14] are used to remove high and low frequency noises [13]. Several efforts in applying deep learning on EEG have been carried out [11][19][20][21]. However, all the previous researches still rely on signal pre-processing pipelines, and none come up with a system that can handle raw data directly. Thus, valuable information might be discarded during the pre-processing.

The purpose of this paper is to introduce an end-to-end deep neural network model that can directly infer cognitive workload and context of driving from raw EEG, where signal pre-processing as well as conventional hand-designed features are not required.

To evaluate the proposed model, EEG signal recordings have been carried out on a subject driving a vehicle in a relatively realistic simulated environment under several different types of cognitive workloads and contexts of driving.

Our experiment shows that the proposed model can accurately classify different labels of cognitive workload and the driving context from raw EEG signals of vehicle driver in a simulated environment without the use of any conventional hand-designed features.

This paper is organized as follows: Section II previews the related researches. Section III explains the proposed model used to classify the data. Section IV details the data collection methodology, and the conducted experiments for evaluating the proposed model. Section V summarizes the acquired results, and finally the conclusion is given in Section VI.

II. Related Works

Monitoring electrical brain activity in a non-invasive means is referred to as EEG in clinical context. Brain activities produce electrical charges caused by the neurons inside the brain, and voltage fluctuations are measured using conductive material called electrodes and a reference electrode attached to the head and scalp [15]. These voltages pass through an amplifier for analysis. Thanks to recent advancements, compact lightweight devices such as MindWave Mobile Headset [16], Emotiv Insight 5 [17] and Muse [18] which is used in this study, have been introduced

Manuscript received on Dec 31, 2017.

Mohammad A. Almogbel is with Graduate School of Fundamental Science and Engineering, Waseda University, Tokyo, Japan (Corresponding author, email: almogbela@ruri.waseda.jp)

Anh H. Dang is with GITS, Waseda University, Tokyo, Japan. (email: danghoanganh@akane.waseda.jp)

Wataru Kameyama is with Faculty of Science and Engineering, Waseda University, Tokyo, Japan (email: wataru@waseda.jp)

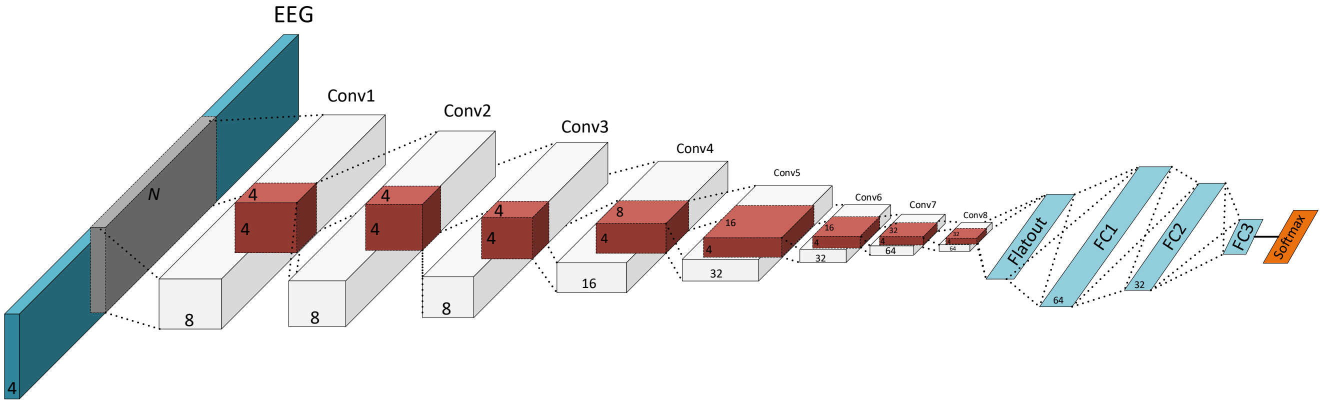


Fig. 1 Network Diagram. The blue cuboid denotes the four raw EEG channels while the gray cuboid inside denotes the slicing window. The white cuboid denotes the kernel while the red cubes denotes the filters applied within each convolutional layer. All the configuration are detailed in Table II.

into the market to allow a more consumer friendly means to monitor EEG signals.

Conventional approaches to classify EEG signals consist of decomposing the signals to extract features to be used for classification. Recent studies of EEG analysis to utilize deep learning with different pre-processing methodologies are shown in Table I.

In [11], FFT is used before classifying the signals to detect driver's drowsiness. [19] evaluates drivers' cognitive performance in a simulated environment using filtered frequency EEG. Similarly, [20] also proposes a model that can predict right and left hand movements with frequency filtered EEG signals. [21] applies a spatial filter before classifying pathological from normal EEG recordings.

TABLE I
RECENT STUDIES ON EEG ANALYSIS WITH DEEP LEARNING

Ref.	Pre-processing	Feature-Extraction	Classifier	Accuracy
[19]	Frequency filtering	CNN	1-layer ANN	86.06%
[11]	FFT	N/A	1-layer ANN	86.50%
[21]	Spatial Filter	CNN	1-layer ANN	84.80%
[20]	Frequency filtering	CNN	1-layer ANN	86.41%

III. PROPOSED MODEL

One of the major advantages of deep neural networks is the ability to extract, learn and generalize features without pre-processing [22]. In this study, we propose the use of deep convolutional networks to predict the cognitive workload of a driver and the context of driving by using raw EEG signals with no additional pre-processing.

CNN is inspired by biological visual system [23] where different levels of visual hierarchy are handled by different parts of the brain. Analogically, multi-layers of CNN generalize visual features in multiple levels. Since each layer of CNN acts as a filter, it is expected that CNN can be also used to extract high level features even in the case of EEG signals.

The architecture of the proposed network is visualized in Fig. 1. The network takes a vector of 4 × N as its input where 4 is the number of raw EEG channels collected from Muse device and N is the length of the signal. Throughout the paper we test the network with different lengths which is explained in detail in section IV-A.

The model consists of several convolutional layers that transform raw EEG data to feature vectors. These features are then flattened and processed by 3 layers of fully connected (FC) classifier blocks.

The FC block outputs a two-, three- or six-dimensional vector depending on the type of classification. Softmax function is used to interpret this vector into the probability of each classification class. Except the final layer of FC block, all layers in this deep network utilize rectified linear unit (RELU) [24] activation function. Batch normalization [25] is also used after every RELU activation function.

Our CNN filters are applied to all EEG channels in the same time. This method ensures that all possible combinations of features are captured. Small filter size and stride together with the high number of CNN layers also ensure that features from multiple hierarchical levels are extracted.

Different from images, EEG comprises multiple time series from each channel. For that reason, it is essential to design CNN filters to stride along time direction only. Thus, it must fully cover all other dimensions of input data.

Table II describes the network configuration for the proposed model. The output shape of each layer is omitted from the table since it is tested on different input sizes. Input sizes are explained in Table III in Section IV-A.

TABLE II
NETWORK CONFIGURATION

Convolutional Block			
	Filter Size	Strides	Activation
Conv1	8×4×4	2	RELU → BN
Conv2	8×4×4	2	RELU → BN
Conv3	8×4×4	2	RELU → BN
Conv4	16×4×8	2	RELU → BN
Conv5	32×4×16	2	RELU → BN
Conv6	32×4×16	2	RELU → BN
Conv7	64×4×32	4	RELU → BN
Conv8	64×4×32	4	RELU → BN
Flat-out			
Fully-Connected Block			
	No. units	Dropout	Activation
FC1	64	0.5	RELU → BN
FC2	32	0.5	RELU → BN
FC3	6/3/2	-	-

IV. EXPERIMENT

A. Data Collection

1) EEG

EEG recording has been done using Muse [18] headband. Fig. 2 shows the international 10-20 system that is used to describe the location of the electrodes on the head. The figure also explains the electrodes used in this device where 4-

channel configuration has been utilized; two dry electrodes on the forehead AF7 (Left) and AF8 (Right), two behind the ears TP9 (Left) and TP10 (Right), and a reference electrode (Fpz) in the middle of the forehead above the nasion. Muse is capable of recording EEG signals at the sampling rate of 256[Hz]. Moreover, it is very portable, lightweight, easy to use, and it utilizes low-energy Bluetooth technology to transmit data making it feasible for real-life usages.

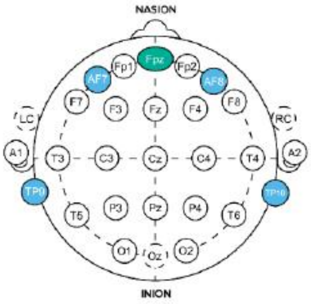


Fig 2. Locations of Scalp Electrodes in the International 10-20 System [18]

Raw EEG data are normalized by z-score for each of the 4 channels. Slicing window approach is utilized to prepare the data for training, where each slicing window is denoted by the time segments of the EEG data. Table III explains the different slicing window size used for training the model as well as the shape of the input data for the network. The slicing windows are overlapped with 1/256[sec] step.

TABLE III
INPUT SHAPES

Raw EEG Slicing Window Size	Input Shape
256Hz × 180 sec × 4 channels	46080 × 4
256Hz × 150 sec × 4 channels	38400 × 4
256Hz × 120 sec × 4 channels	30720 × 4
256Hz × 90 sec × 4 channels	23040 × 4
256Hz × 60 sec × 4 channels	15360 × 4
256Hz × 30 sec × 4 channels	7680 × 4

2) Driving Simulator

Data used in this study are collected through recordings conducted on a popular driving simulation video game (GTA-V free driving mode) [26] that is highly capable of representing the real life driving situations. GTA-V has achieved an increasing popularity amongst recent studies that utilizes driving simulators [27][28][29][30]. Other used equipment consists of Logitech G27 driving wheel and an ultrawide (21:9) curved monitor to better simulate natural vision inside a real car. The experiment environment is shown in Fig. 3.

Fig 3. Representation of the used Simulation Environment with Examples of City (Left) and Highway (Right).

3) Driving Sessions

The driving sessions were designed to accommodate broad scenarios of multiple cognitive workloads and contexts of driving. The data collection was conducted by breaking down the traffic flow/density inside the simulator environment into three levels; zero traffic (light), moderately dense traffic (medium) and high dense traffic (high). Table IV shows the traffic flow and density for each level where traffic flow denotes the average number of cars passing a single point per hour and traffic density denotes the average number of cars within 1 kilometer of a single point.

TABLE IV
EXPERIMENT CONDITIONS AND COMBINATIONS

	Traffic		Type Combinations	
	Flow	Density	City	Highway
Light	0 cars/min	0 cars/km	LC	LH
Medium	30 cars/min	22.5 cars/km	MC	MH
High	36 cars/min	33.23 cars/km	HC	HH

For each traffic flow/density, *city* and *highway* recordings were conducted, which means that the subject was advised to drive exclusively in both city and highway for each traffic flow/density. By doing so, six different types of driving scenarios are accumulated to evaluate the proposed model. Table IV also shows the experiment conditions and combinations with their abbreviations which are used hereafter, such as LC for light traffic in city.

Twelve sessions were collected for each of the six types, where one session was recorded for 15 minutes, which sums up to 72 sessions and 18 hours of raw EEG signals sampled at 256[Hz]. A total of 16,588,800 data points for each of the four raw channels was collected.

The recordings were conducted on a span of two months with random intervals and different times throughout the day. In this experiment, one subject of male with age 29 was joined, who drives frequently with 12 years of driving experience.

B. Types of Classification

To evaluate our proposed model, we considered the following classification problems:

1) Cognitive Workload Classification:

In this type of classification, we are interested in classifying the cognitive workload of the driver regardless of the driving context. In order to carry out this experiment, we group 6 types of driving sessions as explained in section IV-A into 3 classes with regards to the traffic flow/density,

- Low workload: LC, LH
- Medium workload: MC, MH
- High workload: HC, HH.

Traffic congestion is reportedly highly correlated with stress [31], and by grouping the sessions in such manner we investigate the correlation of EEG signals with the traffic/flow density.

Each class in this experiment consists of 24 merged sessions upon each traffic flow/density (5,529,600 × 4 data points). The last session of each merged class is used for evaluation. By doing so, the real-life situation, where the model requires to estimate the workload from future data by learning previous data, can be simulated.

Slight modifications in the network are as follows; the number of the units in FC1 is decreased to 32 in slicing window sizes of 90, 60 and 30 [sec]; it remains as the default configuration shown in Table II in slicing windows of size 180, 150 and 120 [sec].

2) Context Classification:

In this type of classification, we focus on classifying the

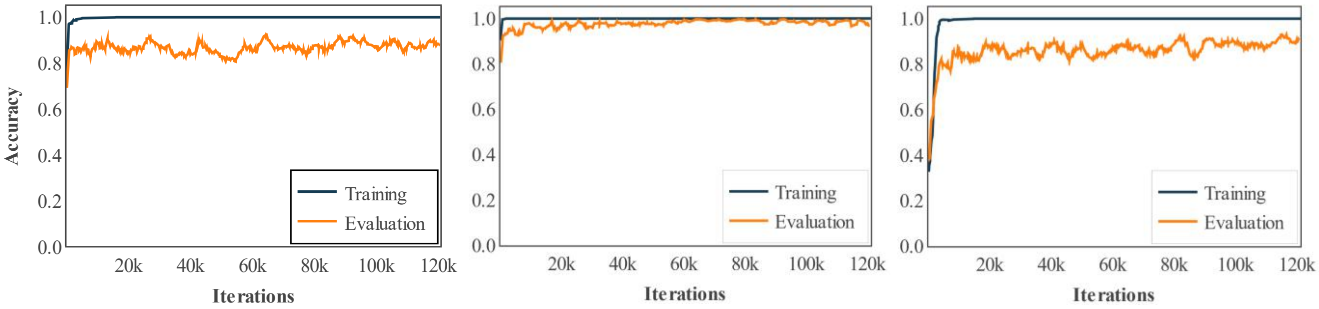


Fig. 4 Learning curve for training and evaluation of the model with the best performing slicing window size (150 [sec]) for workload (left), context (middle) and workload/context (right)

context of driving regardless of the cognitive workload of the driver. Inferring the context of driving directly from EEG signals can be helpful in analyzing the brain signals activity during different context of driving. Therefore, we group our EEG data into 2 classes: city driving and highway driving;

- City: LC, MC, HC
- Highway: LH, MH, HH

Each of the two classes yields 36 sessions (8,294,400 × 4 data points) for each of city and highway sessions. The last session of each merged class is used for evaluation. In this type of classification, the network configuration is unchanged from the default parameters as in Table II.

3) Context/Workload Classification:

Finally, we are interested in classifying the cognitive workload of the driver as well as the driving context at the same time. We assign a unique label to every cognitive workload and context mentioned above to perform a six-way classification for detecting both the cognitive workload of the driver and the context of driving.

Each of the six types yields 2,764,800 data points for every raw EEG channel. The last session for each type is used for evaluation. Slight modifications to the model are as follows; for the slicing window size of 180[sec], the number of units in FC1 is increased to 128. And for the slicing window sizes of 60/30 [sec], the number of units in the FC1 is decreased to 32 instead of 64, and the strides for the 7th convolutional layer is lowered to a stride of 2 down from a stride of 4. The rest remains the same as the default configuration shown in Table II.

C. Training and Evaluation Parameters

The model is trained for 10 epochs using RMSProp optimizer with a learning rate of 0.0001. An epoch represents the training of the model throughout all the data. Input data is divided into batches of 64. During the training the process 50% dropout is applied to the first and second FC layers.

To evaluate the model, a random batch of 64 and its corresponding label is sampled from the evaluation set to be used for cross-validation, and for every batch; recall, precision and accuracy scores are calculated.

V. Results & Discussion

Fig. 4 shows the learning curve for training and evaluation of the model for the best performing slicing window size of 150[sec] for all three types of classification.

1) Workload Classification

We train our model with data prepared using different sizes of slicing windows. Table V shows the evaluation result of our proposed model for each window after the training process. The bold type denotes the highest score among different slicing window sizes. As shown in the table, the model achieves the highest performance with larger slicing window sizes. By average, 120[sec] is the optimal window sizes for this experiment.

Furthermore, the network is capable of accurately classifying *low* workload sessions, whereas *high* workload data are sometimes misclassified as *medium* workload. Session types are grouped together with regards to the traffic flow/density, however the actual cognitive workload could not be correlated in such manner in real life situations.

2) Context Classification

Similar to the workload classification, Table VI shows the evaluation results of our model on context classification. This type of classification achieves the best overall results, and by average, 150[sec] is the optimal window sizes for this

TABLE V
Workload Classification Metrics For different Window Sizes

		180	150	120	90	60	30	Avg.
Light	Recall	**0.997**	0.979	0.960	0.928	0.936	0.791	0.932
	Precision	**1.000**	0.996	1.000	0.982	0.987	0.982	0.991
	Accuracy	**0.999**	0.991	0.984	0.970	0.976	0.926	0.974
	F1 Score	**0.999**	0.987	0.980	0.954	0.961	0.876	0.959
Medium	Recall	0.833	0.921	0.905	0.915	0.890	**0.974**	0.906
	Precision	**0.997**	0.878	0.950	0.673	0.867	0.673	0.840
	Accuracy	0.950	0.923	**0.953**	0.834	0.912	0.838	0.902
	F1 Score	0.908	0.899	**0.927**	0.776	0.878	0.796	0.864
High	Recall	**1.000**	0.871	**1.000**	0.627	0.912	0.749	0.860
	Precision	0.884	0.926	0.900	0.912	0.900	**0.986**	0.918
	Accuracy	0.951	0.932	**0.969**	0.855	0.933	0.908	0.925
	F1 Score	0.938	0.898	**0.947**	0.743	0.906	0.851	0.881
Average	Recall	0.943	0.923	**0.955**	0.823	0.912	0.838	0.899
	Precision	**0.960**	0.933	0.950	0.856	0.918	0.880	0.916
	Accuracy	0.967	0.949	**0.969**	0.886	0.940	0.891	0.934
	F1 Score	0.948	0.928	**0.951**	0.824	0.915	0.841	0.901

TABLE VI
Context Classification Metrics For different Window Sizes

		180	150	120	90	60	30	Avg.
City	Recall	0.885	**0.954**	0.862	0.918	0.886	0.903	0.901
	Precision	0.926	**0.939**	0.862	0.887	0.935	0.901	0.908
	Accuracy	0.996	0.998	0.985	**1.000**	0.995	0.981	0.976
	F1 Score	0.905	**0.947**	0.862	0.903	0.910	0.902	0.905
Highway	Recall	0.927	0.937	0.886	0.886	**0.946**	0.890	0.912
	Precision	0.890	**0.957**	0.886	0.921	0.901	0.896	0.908
	Accuracy	0.996	0.998	0.985	**1.000**	0.995	0.981	0.976
	F1 Score	0.908	**0.947**	0.886	0.903	0.923	0.893	0.910
Average	Recall	0.906	**0.946**	0.874	0.902	0.916	0.896	0.907
	Precision	0.908	**0.948**	0.874	0.904	0.918	0.898	0.908
	Accuracy	0.996	0.998	0.985	**1.000**	0.995	0.981	0.976
	F1 Score	0.906	**0.947**	0.874	0.903	0.916	0.897	0.907

experiment. The network can accurately distinguish between city and highway driving which shows a high correlation between the context of driving and the EEG signals.

3) Workload/Context Classification

Table VII shows the workload/context classification results. Similar to context classification, the window size of 150[sec] achieves the best performance by average. However, in general the network performance slightly drops in this type of classification. Since we train the network each class individually, each class yields smaller number of samples to train per class and the potential similarities between specific types of sessions such as MC and HH.

Low recall scores in LC and MH can be examined in smaller slicing window sizes due to the network misclassifying them for MC or HC.

Discussion

In Table I in section II we show recent reseach that utilize deep learning on EEG signals, however study does not impose in any way a direct comparison with the distinguished previous works because the used data, experimental conditions and classification targets are different in each, but rather explores and introduces the potential of using deep CNN architecture to eliminate the need for pre-processing and feature-extracion.

Nevertheless, our proposed model achieves high accuracy, recall and precision scores on raw EEG signals without applying any conventional pre-processing methodologies.

VI. Conclusion

Ever-proposed and conventional EEG analysis and classification studies are heavily reliant on signal pre-processing and hand-designed features. Such methodologies can be time consuming and potentially cause loss for viable information during the process. In this paper, we propose an end-to-end deep neural network which eliminates the necessity for such pre-processing pipelines, whilst achieving high classification performance.

Using only raw EEG signals from 4 channels as its input, the proposed model performs highly robust and accurate classification. The model is able to achieve an accuracy of 0.960 on average for the vehicle driver’s cognitive workload and the context of driving.

Future works include testing the proposed model with public available data sets. Further investigation will be

TABLE VII
Context/Workload Classification Metrics For different Window Sizes

`		180	150	120	90	60	30	Avg.
LC	Recall	0.745	0.773	0.744	**0.796**	0.685	0.704	0.748
	Precision	**1.000**	**1.000**	0.983	0.958	0.935	0.789	0.944
	Accuracy	0.963	**0.970**	0.951	0.958	0.942	0.911	0.949
	F1 Score	0.854	**0.872**	0.847	0.870	0.790	0.744	0.830
LH	Recall	**1.000**	0.994	0.972	0.889	0.995	0.711	0.928
	Precision	**1.000**	**1.000**	0.938	0.955	0.954	0.806	0.942
	Accuracy	**1.000**	**1.000**	0.984	0.977	0.992	0.926	0.980
	F1 Score	**1.000**	0.997	0.955	0.921	0.974	0.756	0.934
MC	Recall	0.722	0.682	0.693	0.892	0.679	**0.918**	0.757
	Precision	0.739	0.758	**0.771**	0.701	0.696	0.693	0.727
	Accuracy	0.898	0.905	0.905	0.913	0.895	**0.909**	0.904
	F1 Score	0.731	0.718	0.730	0.786	0.688	**0.789**	0.740
MH	Recall	0.725	**0.910**	0.809	0.898	0.517	0.807	0.777
	Precision	**1.000**	0.965	0.983	0.887	0.886	0.826	0.927
	Accuracy	0.955	**0.980**	0.973	0.960	0.914	0.945	0.954
	F1 Score	0.840	**0.937**	0.888	0.892	0.653	0.816	0.838
HC	Recall	0.932	0.918	**1.000**	0.635	0.867	0.784	0.857
	Precision	0.770	0.765	0.738	0.811	0.741	**0.914**	0.786
	Accuracy	0.939	0.927	0.928	0.913	0.919	**0.948**	0.929
	F1 Score	0.844	0.835	**0.849**	0.712	0.799	0.844	0.814
HH	Recall	0.932	0.918	**1.000**	0.635	0.867	0.784	0.951
	Precision	0.804	0.904	**0.921**	0.841	0.682	0.915	0.843
	Accuracy	0.954	0.977	**0.983**	0.953	0.905	0.964	0.956
	F1 Score	0.863	0.911	**0.959**	0.724	0.763	0.845	0.844
Avg.	Recall	0.843	0.866	**0.870**	0.791	0.768	0.785	0.836
	Precision	0.886	**0.899**	0.889	0.859	0.816	0.824	0.862
	Accuracy	0.952	**0.960**	0.954	0.946	0.928	0.934	0.945
	F1 Score	0.855	**0.878**	0.871	0.817	0.778	0.799	0.833

carried out by collecting more data from more subjects with different driving experiences.

REFERENCES

[1] N. Munla, M. Khalil, A. Shahin, and A. Mourad, “Driver stress level detection using HRV analysis,” *2015 Int. Conf. Adv. Biomed. Eng. ICABME 2015*, pp. 61–64, 2015.

[2] B. Eilebrecht *et al.*, “The relevance of HRV parameters for driver workload detection in real world driving,” *Comput. Cardiol. (2010).*, vol. 39, pp. 409–412, 2012.

[3] L. Boon-Leng, L. Dae-Seok, and L. Boon-Giin, “Mobile-based wearable-type of driver fatigue detection by GSR and EMG,” *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2016–Janua, pp. 1–4, 2016.

[4] V. Rajendra and O. Dehzangi, “Detection of distraction under naturalistic driving using Galvanic Skin Responses,” *2017 IEEE 14th Int. Conf. Wearable Implant. Body Sens. Networks, BSN 2017*, no. 2, pp. 157–160, 2017.

[5] M. Venturelli, G. Borghi, R. Vezzani, and R. Cucchiara, “Deep head pose estimation from depth data for in-car automotive applications,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10188 LNCS, pp. 74–85, 2018.

[6] H. Gao, A. Yuce, and J. P. Thiran, “Detecting emotional stress from facial expressions for driving safety,” *2014 IEEE Int. Conf. Image Process. ICIP 2014*, no. May 2015, pp. 5961–5965, 2014.

[7] A. Sahayadhas, K. Sundaraj, and M. Murugappan, “Detecting driver drowsiness based on sensors: A review,” *Sensors (Switzerland)*, vol. 12, no. 12, pp. 16937–16953, 2012.

[8] S. F. Liang, C. T. Lin, R. C. Wu, Y. C. Chen, T. Y. Huang, and T. P. Jung, “Monitoring driver's alertness based on the driving performance estimation and the EEG power spectrum analysis.,” *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, vol. 6, pp. 5738–5741, 2005.

[9] A. Palazzi, D. Abati, S. Calderara, F. Solera, and R. Cucchiara, “Predicting the Driver's Focus of Attention: the DR(eye)VE Project,” pp. 1–25, 2017.

[10] X. Li, H. Huang, and Y. Sun, “DriTri: An in-vehicle wireless sensor network platform for daily health monitoring,” *Proc. IEEE Sensors*, pp. 3–5, 2017.

[11] I. Belakhdar, W. Kaaniche, R. Djmel, and B. Ouni, “Detecting driver drowsiness based on single electroencephalography channel,” *13th Int. Multi-Conference Syst. Signals Devices, SSD 2016*, pp. 16–21, 2016.

[12] C. L. Baldwin, D. M. Roberts, D. Barragan, J. D. Lee, N. Lerner, and J. S. Higgins, “Detecting and Quantifying Mind Wandering during Simulated Driving,” *Front. Hum. Neurosci.*, vol. 11, no. August, pp. 1–15, 2017.

[13] L. Bi, R. Zhang, and Z. Chen, “Study on Real-time Detection of Alertness Based on EEG,” *2007 IEEE/ICME Int. Conf. Complex Med. Eng.*, pp. 1490–1493, 2007.

[14] S. S. Daud and R. Sudirman, “Butterworth Bandpass and Stationary Wavelet Transform Filter Comparison for Electroencephalography Signal,” *Proc. - Int. Conf. Intell. Syst. Model. Simulation, ISMS*, vol. 2015–Octob, pp. 123–126, 2015.

[15] S. Donald L. and F. H. L. da Silva, *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Lippincott Williams & Wilkins, 2011.

[16] NeuroSky, “MindWave Mobile Headset.,” 2015. [Online]. Available: https://store.neurosky.com/pages/mindwave.

[17] EMOTIV, “EMOTIV Insight 5 Channel Mobile EEG - Emotiv,” *Emotive*, 2018. [Online]. Available: https://www.emotiv.com/product/emotiv-insight-5-channel-mobile-eeg/.

[18] Interaxon, “Muse: the brain sensing headband,” *Tech. Specif. Valid. Res. use*, pp. 4–9, 2017.

[19] M. Hajinoroozi, Z. Mao, T. P. Jung, C. T. Lin, and Y. Huang, “EEG-based prediction of driver's cognitive performance by deep convolutional neural network,” *Signal Process. Image Commun.*, vol. 47, pp. 549–555, 2016.

[20] Z. Tang, C. Li, and S. Sun, “Single-trial EEG classification of motor imagery using deep convolutional neural networks,” *Optik (Stuttg).*, vol. 130, pp. 11–18, 2017.

[21] R. T. Schirrmeister, L. Gemein, K. Eggensperger, F. Hutter, and T. Ball, “Deep learning with convolutional neural networks for decoding and visualization of EEG pathology,” 2017.

[22] Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision,” *IEEE Int. Symp. Circuits Syst. Nano-Bio Circuit Fabr. Syst.*, pp. 253–256, 2010.

[23] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980.

[24] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” *Proc. 27th Int. Conf. Mach. Learn.*, no. 3, pp. 807–814, 2010.

[25] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” Feb. 2015.

[26] “Gta V,” 2013. [Online]. Available: https://www.rockstargames.com/V/.

[27] Y. Chen, W. Li, and L. Van Gool, “ROAD: Reality Oriented Adaptation for Semantic Segmentation of Urban Scenes,” 2017.

[28] P. Bazilinskyy *et al.*, “Eye movements while cycling in GTA V,” no. January, pp. 7–11, 2018.

[29] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, “Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks?,” *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 746–753, 2017.

[30] M. Angus *et al.*, “Unlimited Road-scene Synthetic Annotation (URSA) Dataset,” 2018.

[31] D. A. Hennessy, D. L. Wiesenthal, and P. M. Kohn, “The Influence of Traffic Congestion, Daily Hassles, and Trait Stress Susceptibility on State Driver Stress: An Interactive Perspective 1,” 2000.

Mohammad A. Almogbel (S'14) received his bachelor's degree in Information Systems from King Saud University, Riyadh, Saudi Arabia in 2009. He joined King Abdul-Aziz City for Science and Technology in Saudi Arabia as a researcher in the same year and received a scholarship to complete his graduate school in 2010. He then received master's degree in computer science from Waseda University in 2014 and he continued to pursue his Ph.D. since then. He is a member of IEEE, ITS and JSAE.

Anh H. Dang (S'09) received his bachelor's degree in business administration, information & communication technology from Ritsumeikan Asia Pacific University (Beppu, Oita, Japan) in 2010. He then received the master's degree in computer science from Waseda University (Shinjuku, Tokyo, Japan) in 2012. Since 2012, he is a Ph.D. candidate at Waseda University. He is a member of IEEE, ACM, and IEICE. His research interests are machine learning, artificial intelligence, and computer vision.

Wataru Kameyama (M'86) received the bachelor's, master's, and D.Eng. degrees from the School of Science and Engineering, Waseda University, in 1985, 1987, and 1990, respectively. He joined ASCII Corporation in 1992 and was transferred to France Telecom CCETT from 1994 to 1996 for his secondment. After joining Waseda University as an Associate Professor in 1999, he has been a Professor with the Department of Communications and Computer Engineering, School of Fundamental Science and Engineering, Waseda University, since 2014. He has been involved in MPEG, MHEG, DAVIC, and the TV-Anytime Forum activities. He was a Chairman of ISO/IECTC1/SC29/WG12, and a Secretariat and Vice Chairman of the TV-Anytime Forum. He is a member of IEICE, IPSJ, ITE, IIEEJ, and ACM. He received the Best Paper Award of Niwa-Takayanagi in 2006, the Best Author Award of Niwa-Takayanagi in 2009 from the Institute of Image Information and Television Engineers, and the International Cooperation Award from the ITU Association of Japan in 2012.

Volume 7 Issue 6, November. 2018, ISSN: 2288-0003

ICACT-TACT

JOURNAL

GIRI

Global IT Research Institute

1713 Obelisk, 216 Seohyunno, Bundang-gu, Sungnam Kyunggi-do, Republic of Korea 13591
Business Licence Number : 220-82-07506, Contact: tact@icact.org Tel: +82-70-4146-4991