

**Doctoral Dissertation**

**A Conversational System  
for Interactive Image Editing**

Seitaro Shinagawa

July 31, 2020

Graduate School of Information Science  
Nara Institute of Science and Technology

A Doctoral Dissertation  
submitted to Graduate School of Information Science,  
Nara Institute of Science and Technology  
in partial fulfillment of the requirements for the degree of  
Doctor of ENGINEERING

Seitaro Shinagawa

Thesis Committee:

Professor Satoshi Nakamura	(Supervisor)
Professor Yasuhiro Mukaigawa	(Co-supervisor)
Professor David Traum	(Co-supervisor; USC, USA)
Associate Professor Sakriani Sakti	(Co-supervisor)
Assistant Professor Koichiro Yoshino	(Co-supervisor)

# A Conversational System for Interactive Image Editing\*

Seitaro Shinagawa

## Abstract

Systems with natural language interfaces such as conversational interface are useful for human users in human–system collaboration tasks. Interactive image editing task is a task that uses natural language interfaces, which is a potential application for non-skilled users. If users want to create an imagined image, they can ask the system to create the image as we usually do with skilled sketch artists. This thesis presents an interactive image editing system based on neural network image generative models, which proactively communicates with users to create the desired image.

This thesis addresses the following two challenging problems for the interactive image editing task. The first problem is that the systems have to handle various editing requests from the users in natural language, which include requests for a slight change of images. This problem gives us an advantage over the previously known image retrieval systems, which can only provide images existing in the database. Chapter 4 addresses this problem in our editing system. We propose an interactive image editing framework based on machine learning systems, neural network-based image generative models. This framework aims at training a model to automatically learn relationships between the change of images and the natural language editing requests. The model can directly estimate and generate new images from given previous (source) images and the users’ natural language instructions (editing requests) to generate a fixed image as the user demanded. To evaluate the model in this framework, (i) we demonstrate how our editing

---

\*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, July 31, 2020.

model works in an artificial dataset, which is automatically created from a public handwritten dataset. (ii) We also present a more practical task, avatar face illustration editing, whose instructions are collected from human annotators. These tasks require systems to handle various editing requests. With these datasets, we present a difficulty in deciding the editing region of the source image that is mentioned in the editing requests. We propose source image masking (SIM) to mitigate this difficulty. The SIM explicitly identifies the region mentioned in the editing request. We demonstrate that the system with the SIM architecture (w/SIM) outperforms the system without the SIM architecture (w/o SIM) in most of the editing requests in the dataset.

The second problem is that the systems have to handle the uncertainty of the generated images due to the diversity of editing requests. Machine learning models are trained with the limited dataset, and in general, the users have different knowledge, skills, and cultures. Therefore, miscommunication between the systems and the users is inevitable. In other words, it is difficult for a single editing model to perform correctly for all editing requests. This problem forces the users to learn how the system behaves through many trial-and-errors, which bothers the users. Conventional image generation or editing systems have the same problem. A naive strategy for the system to solve the problem is showing the generated images from different editing models to confirm the most relevant image to the user's request every time. However, this strategy has a problem that the system makes the interactive process redundant. Chapter 5 addresses this problem in our editing system. We propose a proactive confirmation method that enables the system to confirm with the user when the system is tentative about selecting a better image to match the user's editing requests. We defined an uncertainty score by using the entropy of the generated image to decide the system action to confirm. We demonstrate our method achieves a lower number of confirmation to the users with better image qualities through the dialogues.

**Keywords:**

image generation, image editing, natural language interface, dialogue system, adversarial learning

# Contents

Acknowledgements . . . . .	xii
<b>1 Introduction</b>	<b>1</b>
1.1. Background . . . . .	1
1.1.1 Drawing assistance systems with a natural language interface	3
1.1.2 An alternate approach: image retrieval systems . . . . .	4
1.2. Problems in existing works . . . . .	4
1.3. Approaches in this thesis . . . . .	5
1.3.1 Neural network-based models for semantic image editing with natural language . . . . .	6
1.3.2 Interactive image editing with a system’s proactive confir- mation . . . . .	7
1.4. Contribution of thesis . . . . .	9
1.5. Outline of the thesis . . . . .	9
<b>2 Related works</b>	<b>11</b>
2.1. Concurrent works . . . . .	11
2.1.1 Connection between interactive image editing and other vision-and-language tasks . . . . .	11
2.1.2 Comparison our system and the other interactive image editing systems . . . . .	12
2.2. Background techniques . . . . .	12
2.2.1 Evaluation metric for image quality . . . . .	12
2.2.2 Basic neural networks modeling . . . . .	13
2.2.3 Convolutional neural network (CNN) . . . . .	16
2.2.4 Long-short term memory network (LSTM) . . . . .	16

2.2.5	Deep convolutional generative adversarial network (DCGAN)	18
2.2.6	Conditional generation with DCGAN for caption-to-image generation . . . . .	20
<b>3</b>	<b>Interactive image editing</b>	<b>23</b>
3.1.	Task setting . . . . .	23
3.2.	Data collection . . . . .	26
3.2.1	MNIST editing with artificial instruction (Artificial MNIST) dataset . . . . .	26
3.2.2	Avatar image editing with human instruction (AIMI) dataset	29
<b>4</b>	<b>Image editing with natural language instruction</b>	<b>35</b>
4.1.	Introduction . . . . .	35
4.2.	Model architecture . . . . .	37
4.2.1	Our modification of conditional DCGAN for semantic image editing with instruction . . . . .	38
4.2.2	source image masking (SIM) . . . . .	41
4.3.	Experimental settings with Artificial MNIST dataset . . . . .	41
4.4.	Results with Artificial MNIST dataset . . . . .	42
4.4.1	Generated examples . . . . .	42
4.4.2	Qualitative analysis of instruction vectors . . . . .	42
4.5.	Experimental settings with Avatar Image Manipulation with Instruction dataset . . . . .	46
4.6.	Results with Avatar Image Manipulation with Instruction dataset	48
4.6.1	Analysis on the instruction types . . . . .	52
4.6.2	Comparison with the larger image size with the SIM model	54
4.7.	Discussion . . . . .	55
<b>5</b>	<b>An entropy-based confirmation for image editing dialogue</b>	<b>59</b>
5.1.	Introduction . . . . .	59
5.2.	Conditional image-to-image generation with a masking module . .	61
5.3.	System’s confirmation of action decisions based on mask entropy .	62
5.4.	Experimental settings . . . . .	63
5.4.1	Dataset . . . . .	63

5.4.2	Training models . . . . .	63
5.4.3	User evaluation of image editing dialogue . . . . .	64
5.4.4	Necessity of confirmation (limitation of a single model) . . . . .	65
5.4.5	Effectiveness of confirmation strategy . . . . .	65
5.5.	Results . . . . .	65
5.5.1	Necessity of confirmation (limitation of a single model) . . . . .	66
5.5.2	Effectiveness of confirmation strategy . . . . .	69
5.6.	Discussion . . . . .	71
<b>6</b>	<b>Conclusion of this thesis</b>	<b>75</b>
6.1.	Remaining problems and future directions . . . . .	76
	References . . . . .	79
	Appendix . . . . .	91
A.	Snippet for generating Artificial MNIST . . . . .	91
B.	Model architectures for the experiment . . . . .	92





# List of Figures

2.1	A simple 2-layer feed-forward neural network. . . . .	14
2.2	A calculation example of convolutional neural network. Input is $3 \times 3$ squared with 3 channels. Kernel window size is (3, 3), stride is 2, and padding is 1. Padding values represent white color with dot line in the input. . . . .	17
3.1	Overview of interactive image editing. . . . .	25
3.2	Generation procedure for Artificial MNIST dataset. . . . .	27
3.3	Rendering examples. ( $w, h$ ) denotes resizing factors for width and height of the original MNIST images. . . . .	28
3.4	MNIST sample transformation with manual transition corresponding to the given instruction command ( <i>action, digit, direction</i> ). . . . .	29
3.5	Avatar creation website for collecting avatar images. . . . .	30
3.6	Crowd-sourcing instruction scripts shown to crowd workers. . . . .	31
3.7	Annotated instruction examples via crowd-sourcing. . . . .	32
4.1	Comparison of natural language conditioned image generation framework between the existing cap2image (left) and the proposed image manipulation with instruction (IMI) (right). . . . .	36
4.2	IMI model architectures for (a) Artificial MNIST editing, (b) avatar editing, and (c) avatar editing with source image masking. . . . .	38
4.3	Generated examples of IMI model using handwritten digit manipulation dataset (Artificial MNIST). . . . .	43

4.4	Cosine similarities between the instruction vectors from the instructions that contain {"expand", "compress", "move"} (left) and {"put", "remove"} (right) . . . . .	44
4.5	Enlarged cosine similarities of "move"- "move" block in Figure 4.4	45
4.6	Distribution of variance of similarities of each group of instruction vectors . . . . .	46
4.7	Generated results on each phase . . . . .	47
4.8	Model selection using feature matching loss . . . . .	48
4.9	Histogram of SSIM between the generated and target images using w/o and w/ SIM models. . . . .	49
4.10	Subjective evaluation of generated images between w/o and w/ SIM models. The attached numbers to the labels denote the actual number of vote by the evaluators. The evaluators saw image pairs (A, B). One is generated from w/o model, and the other is from w/ SIM model. We asked the evaluators to select five-grade: (1: A is much better than B, 2: A is better than B, 3: the results are comparable, 4: B is better than A, 5: B is much better than the A). We de-anonymized whether images are from w/o or w/ SIM model, and visualized the number of the votes. . . . .	50
4.11	Generated examples using w/o SIM model and w/ SIM model . .	51
4.12	SSIM comparison between w/ and w/o SIM models on the first 11 instruction types. The plots are ordered in ground truth masks (gtmask), which indicate the number of different pixels between source and target images. There are three SSIMs: original source-target SSIM (base SSIM); generated-target SSIM subtracted by the base SSIM (relative SSIM); relative SSIM but outside of edit region covered with gtmask is ignored (relative SSIM (gtmask)).	52
4.13	SSIM distributions on the last 11 of 22 instruction types. The settings are same as Figure 4.12 but the instruction types are the last 11 types. . . . .	53

4.14	SSIM comparison between image size of 64 and 128 on the first 11 instruction types. The plots are ordered in ground truth masks (gtmask), which indicate the number of different pixels between source and target images. There are three SSIMs: original source–target SSIM (base SSIM); generated–target SSIM subtracted by the base SSIM (relative SSIM); relative SSIM but outside of edit region covered with gtmask is ignored (relative SSIM (gtmask)).	55
4.15	SSIM comparison between image size of 64 and 128 on the last 11 of 22 instruction types. The settings are same as Figure 4.14 but the instruction types are the last 11 types. . . . .	56
5.1	Overall interactive image editing flow described in Section 3.1 with the proposed confirmation strategy (blue box in the right) and DCGAN-based w/ and w/o mask (SIM) models described in Chapter 4 (green box in the right). . . . .	61
5.2	Experimental results of image editing dialogue between 18 evaluators (users) and the system: <i>#user turn</i> denotes total number of user actions ( <i>making editing request</i> and <i>selecting an image</i> ); (smaller is better). <i>SSIMimprovement</i> denotes the relative SSIM based on the first SSIM (higher is better). Each plot on each figure represents the trail of <i>SSIMimprovement</i> in one dialogue task between the system and an evaluator. $\alpha$ indicates threshold for system to select confirmation: (a) $\alpha = 0.0$ , (b) $\alpha = 0.25$ , (c) $\alpha = 0.50$ , (d) $\alpha = 0.75$ , (e) $\alpha = 1.0$ , and (f) random: system randomly selects confirmation. If $\alpha$ becomes small, the system tends to select confirmation with lower uncertainty score. Note that every <i>SSIMimprovement</i> is calculated after user’s action. Therefore, when the system selects confirmation after the user makes an editing request, <i>SSIMimprovement</i> keeps identical value. Degradation as dialogues progress is caused by image editing models. . . .	66
5.3	Visualization of average and standard deviation on each figure of Figure 5.2. . . . .	67
5.4	A case with low SSIM improvement. . . . .	68

5.5	Histogram of $SSIM_{improvement}/\#user\ turn$ at end of dialogues on each strategy: higher $SSIM_{improvement}/\#user\ turn$ dialogue created more similar images to goal and more efficient dialogues. . . . .	69
5.6	Distribution of $\#user\ turn$ on each strategy: smaller $\#user\ turn$ dialogue represents more efficient dialogue. . . . .	70
5.7	Dialogue example with $\alpha = 0.50$ (confirmation threshold $-\alpha \log 0.5 = 0.35$ ). $i$ indicates turn index defined in Chapter 3. $\#user\ turn$ denotes number of user actions, which represents total number of <i>making an editing request</i> and <i>selecting an image</i> . We put the source-goal SSIM next to each source image when the system decides a generated image of each turn. . . . .	72
5.8	Inefficient dialogue example with random confirmation: $i$ indicates turn index defined in Chapter 3. $\#user\ turn$ denotes number of user actions, which represents total number of <i>making editing request</i> and <i>selecting an image</i> . . . . .	73

# List of Tables

6.1	IMI model for Artificial MNIST dataset. . . . .	94
6.2	IMI model for AIMI dataset. . . . .	95
6.3	IMI model with Source image masking for AIMI dataset. . . . .	96

## Acknowledgements

First and foremost, I would like to express my gratitude to Professor Satoshi Nakamura for welcoming me into AHC lab, and his hospitable supervision from the start of my doctor course. I am very impressed by his considerate supervision. He has not only directed me to advance my research but also presented what an excellent researcher/educator is and how to be. He also has paid his great effort to construct an excellent research laboratory. His supportive laboratory gave me a comfortable laboratory life. I spent a great time discussing with the lab members and met a lot of research ideas and topics. This expanded my view. His tireless supervision dictated my career decision, and I would like to look up to him, an excellent researcher and a good educator, in the future.

Thank you to Professor Yasuhiro Mukaigawa for agreeing to serve as members of my doctoral committee, and for carefully reviewing this thesis and providing insightful advice.

I would like to thank Professor David Traum and the members of USC Institute for Creative Technologies for welcoming me as an intern member and the careful and lively discussion on my internship research project. Professor David Traum has provided me with the opportunity to study at USC, giving me a chance to challenge the new research topic. The experience of studying abroad at USC was precious. It has strengthened my confidence in the research.

I would also like to thank Associate Professor Sakriani Sakti for her kind and critical advice and discussion to advance my research. Her insight and knowledge backed by the rich experience gave me a lot of clues when I faced problems in my research.

I would also like to thank Assistant Professor Koichiro Yoshino for incredible supervision as the closest supervisor for me. I have learned a lot of things from him, for example, how to advance my research, how to construct or write papers, how to do public relations as a researcher, how to keep the motivation and mind as a researcher, and how to face troublesome problems in various situations. He also gave me many opportunities to attend conferences and study meetings. I learned how important social activities are in research progress.

Of course, I would like to thank the member of AHC lab, student and staffs. There are so many others that have helped me along my way that it would be

impossible to name them all, but there are a few people that deserve special mention. Thanks to Dr. Yu Suzuki and Dr. Hiroki Tanaka, and Dr. Takatomo Kano for giving me critical questions and advice in the discussion. I would like to thank Ms. Manami Matsuda and Ms. Miho Hayashi for the grateful support for my research activities. Thanks to them, I felt comfortable in the laboratory during my doctor course. Thanks to Dr. Takuya Hiraoka for giving me critical advice at the beginning of my doctor course. I would also like to thank Dr. Masahiro Mizukami and Mr. Kyoshiro Sugiyama for spending much time with me talking a lot of things, for example, dialogue system research topics, troubles, and ramen. Most of my great ramen experiences in Nara were carried out with them. I would like to thank Dr. Hayato Maki and Mr. Yusuke Oda for discussing various topics and sharing ideas of research and machine learning techniques. I was impressed by them, and they encouraged me to learn modern machine learning techniques faster. Thanks to Mr. Kazuya Ikuta and Ms. Sara Asai for accepting me as their tutor on their research. They had great curiosity and sustainable effort. They often came to me and asked questions and discussed many things with me. They encouraged my understanding on various topics about research to be deep. They sometimes supported me by showing a technique they found or correcting my mistake. They provided me with a challenging but exciting essence on education.

During my doctor course, I was also encouraged by the help of my friends. I would also like to thank the people in twitter who discussed research topics with me, shared some excellent papers, and gave me advice that supported my research activities. Finally, thanks to Mom, Dad, and all of the rest of the family for endlessly encouraging and understanding my studying in the doctor course.

# Chapter 1

## Introduction

### 1.1. Background

Images, pictures, and illustrations are useful for communicating what we are imagining. For example, the sketches of a suspect's face based on eyewitness observations support police departments that are spreading and sharing information to find that suspect. However, creating images requires talent, lots of time, and much training, especially for non-skilled users. Various systems or tools currently assist the creation of images, such as Photoshop; however, manipulating these tools also requires drawing skill with them. Is there a better way for users to easily create images without such specific drawing skills? Suppose our case. When we do not have drawing skills, we can communicate in natural language with others who can draw. We can ask them to draw the image we are holding in our mind. For example, for sketching a suspect, an eyewitness describes the person's face to a professional police artist who reconstructs it based on the eyewitness's testimony. Another example is a business transaction where a client asks a designer to create a car design. Since natural language communication is a basic human skill, many people can use it regardless of their drawing skills. Therefore, a conversational interface that enables a drawing system to communicate with users in natural language has potential to effectively assist its users.

This thesis aims to achieve a conversational drawing system on an image editing task. An image editing is a specific task of image creation tasks. In a typical image creation task, a source image may or may not be present. For



example, a real situation for a car design, the designer/artist may show image templates to a client who must select one that matches their requirements and draw from the selected template. The artist may also start from scratch. An image editing task, our task scenario, assumes that a source image to be edited exists in the task beginning. The reason we focus on the specific task is that we can find a partially relevant image by existing image processing technologies to reduce initial cost. For example, we can access a large number of image resources by web search engines for general design tasks and access an image by collage technologies. Such technologies allow a user to synthesize an image by combining partial image templates in a specific task, such as drawing a face. On the other hand, interactive process to receive the desired image are inevitable and requires much cost even in a general image creation task. Finding an image that exactly matches the user desire is exhausting if we have a large number of images. Furthermore, these image search or template approaches can only provide an available image in existing databases. Thus, we believe there is a much space to consider the question “how can we receive further improvement based on a partially relevant image toward the desired image with small effort?” We believe that the development of image editing technology has a potential to bridge the gap between the available images, using image search or synthesizing templates, and the actual images that satisfy user requirements. We also believe that this technology will benefit not only non-skilled people but also experienced sketch artists who want to improve communication with their clients.

In this chapter, first, Sections 1.1.1 and 1.1.2 introduce the image editing task tackled by our conversational drawing system and its advantages. Second, in Section 1.2, we discuss the challenges for achieving our conversational drawing system. Third, Section 1.3 describes our approaches to solve these problems. Fourth, Section 1.4 summarizes the contributions of our thesis. Finally, Section 1.5 introduces its outline.

### 1.1.1 Drawing assistance systems with a natural language interface

Historically speaking, the idea of systems that support human drawing with a natural language interface is not new. Many studies have been investigated for a long time and feature two main streams.

The first stream includes systems that obey user requests and perform certain manipulations for users. In the mid-1900s, this initial concept was introduced by SHRDLU [66], which performed such object-movement manipulations as grabbing and repositioning things based on natural language requests. After SHRDLU, similar studies appeared in the creation of 3D graphics scenes [1, 12]. Such systems also performed object-movement manipulations based on user input: “Put [object A] [position] [object B]” to render 3D graphics scenes.

The second stream is comprised of systems that support the direct manipulation of graphical user interfaces with natural language. An early drawing system was proposed in the 1990s that accepts such multi-modal inputs as a mouse, keyboard, and voice input [23]. In this system, natural language was input as voice input, which was mainly used to assist other modalities (mouse or keyboard). Since the early study was interested in the time synchronization of the voice input and other modalities, the voice input was comprised of such simple commands as “move this here.” VoiceDraw [21] accepted only voice input for line drawing. It allowed users to input both specific voice commands and also a flexible hands-free line drawing by changing the user’s voice duration. Combining voice commands and gaze also worked well for flexible line drawing [62].

A commonality in the studies of the two streams is that systems require the user to enter simple natural language commands that correspond to the simple manipulations of drawing tools, such as “move this here.” In the first stream, users can easily change the position of objects. The second stream’s systems basically require that users possess drawing skills. Difficult manipulations for non-skilled users include drawings that are comprised of such semantic changes as “make his sunglasses round” or “make the hair a little longer and wavy” for drawing a face. Therefore, to address this problem and assist non-skilled users, we tackle semantic image manipulation (editing) task with natural language.

### 1.1.2 An alternate approach: image retrieval systems

Image retrieval systems find images in databases that are relevant to the given user’s queries, such as images [38], tags [48], and natural language sentences [31]. These image retrieval systems are promising approach to find images that are roughly relevant to what users are imagining. However, the performance of image retrieval systems depends on the database’s size because they can only present images that exist in image databases. Retrieval-based image editing for slight change requests, including “make the hair a little longer and wavy” in face editing requires many examples of the same face with different kinds of sunglasses or hairstyles and appropriate annotations. On the other hand, generation-based image editing learns to generate a new image from a training set composed of independent pairs of pre- and post-edited examples. Its advantage is that it does not need to cover every combination of the changes, e.g., sunglasses or hairstyles, on each face to work well. For this reason, our thesis focuses on generation-based editing rather than retrieval-based editing. We note that the fundamental advantage of generation-based image editing systems can be combined with existing image retrieval systems. Although we only focus on generation approach in this study, we expect that a unified system that combines our system and image retrieval systems will be improve the performance in the future.

## 1.2. Problems in existing works

Semantic image editing with natural language described above faces the following two challenges. The first is how to generate a new image based on users’ editing requests in natural language. Although primitive operations such as moving objects are feasible with handcrafted rules, manually achieving such semantic changes as “make his sunglasses round” or “make the hair a little longer and wavy” are very challenging. Furthermore, image retrieval systems are ineffective for editing requests that represent slightly changed images, such as “make the hair a little longer and wavy.”

The second problem is that the systems have to handle various editing requests from users in natural language. Suppose that a user wants to add sunglasses to an image. The request might be “add sunglasses,” “make the glasses darker,”

or “put on black glasses.” User requests reflect a wide array from which to express specific operations. Editing systems must be able to absorb such diversity. Moreover, suppose a communication task where a client asks a professional designer to create a car advertisement. How would the designer react if the client made the following request: “Is it possible to make this car more stylish” If the designer has experience managing such a “make stylish” operation from their client, they can imagine possible target operations. However, if this is the first request she’s received from that client, they will probably be confused because the “make stylish” operation is ambiguous. The request might refer to the color or the shape of the car. In such cases in human-to-human communication, we can easily clarify the client’s intention: “Are you referring to the car’s color or its shape?” The next time the same client asks for the same “make stylish” service, the designer can successfully do the desired operation. The meaning of natural language often reflects context, including individual preference and the relationships among people, time, place, and culture. Since eliminating this ambiguity is impossible, grounding each other’s intentions through dialogue is essential in semantic image editing with natural language. Creating an image editing system that can accept and handle all editing requests is basically impossible. Therefore, semantic image editing systems with natural language require functionality for grounding the intentions between a system and its users. Based on natural language interaction, we call such semantic image editing tasks interactive image editing tasks in this thesis.

### **1.3. Approaches in this thesis**

First, Chapter 3 introduces and defines interactive image editing tasks. Since we found no existing dataset for such tasks when we began our research, Chapter 3 also introduces our data collection of two datasets. We first collected an artificially generated editing dataset from a public, handwritten dataset [35] (Artificial MNIST) to investigate the operation of our editing model, which is described in Section 1.3.1. We also collected a more practical editing dataset, which is an avatar image editing with human instruction (AIMI) dataset, whose images are automatically collected from an avatar creation website where editing requests

are manually annotated by humans by crowd-sourcing.

Our proposed approach consists of two parts. First, we solved the first problem and part of the second one by constructing single-turn image editing models (Section 1.3.1). Second, we solved the second problem by constructing an interactive image editing system using image editing models shown in (Section 1.3.2).

### **1.3.1 Neural network-based models for semantic image editing with natural language**

As described from Sections 1.1.1 to 1.2, semantic image editing requires feasibility for more complicated editing requests than simple operations such as object movement. The following is a common research question in the vision-and-language research field: “What is an effective way to unify image semantics and natural language semantics?” In recent years, machine learning approaches using neural networks have been successful in computer vision tasks, such as image recognition [33, 53], object detection [18], semantic segmentation [39], action recognition [56], and image generation [50], and in such natural language processing as machine translation [4, 59], question answering [57], and chat-based dialogue systems [63]. Using techniques from both fields, many vision-and-language tasks have also been successful, including image retrieval using natural language query [31], visual question answering [3, 17], generating descriptions from images called image captioning [8, 64], and generating images from captions called text-to-image or caption-to-image [42, 51]. The advantage of machine learning approaches is that they can model the relationship between images and natural language without handcrafted rules. The performances of neural networks are especially good because they can approximate complicated mapping between images and natural language.

Moreover, studies on collaborative systems that can properly instruct users have also been emerging in a field called vision-and-dialogue. Vision-and-dialogue systems are extensions of the vision-and-language field. For example, dialogue-based interactive image retrieval [14, 20] and visual dialogues [13, 15] can be viewed as a dialogue extension of visual question answering and image-captioning. Interactive image editing resembles a dialogue extension of text-to-image.

Chapter 4 builds a neural network-based image editing model for our interactive image editing task. We start by extending the existing text-to-image generation model [51], which consists of long-short term memory neural networks (LSTM) [24] for a text encoder and generative adversarial networks (GANs) [19] for an image decoder. These text encoders and image decoders are trained together, and the entire model learns the mapping function between the input text and the output image without handcrafted rules. Our image editing model generates a new image based on a source image, which is the image previously shared by the system and the user, based on an editing request from the user. To create this editing model, we combine an additional image encoder based on a convolutional neural network (CNN) [34] for source image input with the text-to-image model [51] and train it to learn a mapping function between an input pair source image and editing request and output image.

In Chapter 4 we also propose source image masking (SIM), which identifies the region that should be changed to preserve the other regions to improve the image editing performance. We demonstrate that in most editing requests a system with SIM architecture (w/ SIM) outperforms a system without it (w/o SIM).

### **1.3.2 Interactive image editing with a system’s proactive confirmation**

Functionality for grounding intentions between the system and users is achieved by introducing a confirmation strategy to the system. Confirmation [11], which is an effective strategy to deal with input uncertainty, has mainly been investigated in the spoken dialogue system research field [32, 45]. Such spoken dialogue systems must address the mistakes of speech recognition or natural language understanding. In this situation, confirmation effectively manages the dialogue process. A confirmation method, based on confidence measures [32], calculates the confidence score of each content word among the speech recognition candidates. The system provides confirmation to the user when the confidence in the content word in the user utterance is uncertain. The confirmation method for a document retrieval dialogue task is based on minimizing the Bayes risk [45], which must be calculated by a classification model.

Interactive image editing systems also have to handle the uncertainty of the generated images due to the array of editing requests because users generally have different knowledge, skills, and cultures. Moreover, machine learning models are trained with limited datasets. Therefore, it is difficult for a single editing model to perform correctly every editing request. When the system’s output is different from the user’s intent, the user needs to try again. This situation levies an additional cost on users who must learn how the system behaves through much trial-and-error. Our interactive image editing system tackles this problem by incorporating a confirmation strategy.

Chapter 5 introduces a method for system’s proactive confirmation strategy. A naive strategy through which the system solves the uncertainty problem shows generated images from different editing models to confirm the most relevant image to the user’s request every time. However, this strategy makes the interactive process redundant. Chapter 5 addresses this problem in our editing system. We propose a proactive confirmation method that enables the system to confirm its tentativeness with users about selecting a better image to match their editing requests. We defined an uncertainty score using the generated image’s entropy to decide which system action to confirm. We demonstrate that our method reduces the number of confirmations to users with better image qualities through dialogues. Similar to an existing confirmation method that uses confidence measures [32], our proposed confirmation method provides a confidence score for confirmation. However, the calculation of confidence scores is based on the entropy of the image generation system. In contrast to the existing confirmation method that requires a classification model to calculate the Bayes risk [45], our entropy-based method requires no additional model or dialogue data for training the model.

Note that controlling generated images is an essential problem in GAN-based image generation because of the instability of the generated image’s quality. The truncation trick, which restricts acceptable samples on latent space  $z$ , satisfactorily stabilizes the image quality in conditional image generation [6]. Unfortunately, it fails to provide any information about uncertainty. Our entropy-based method provides uncertainty scores for generated images. Uncertainty detection for GAN-based models has been scrutinized in anomaly detection [55, 2]. How-

ever, it measures the distances between the generated images and the samples in a training dataset without indicating their suitability for the given condition. Our entropy-based method is based on a mask made from the given condition that provides a confidence score that represents the suitability of the generated image for the given condition.

## 1.4. Contribution of thesis

The following are the contributions of this thesis:

- We collect two datasets for an interactive image editing task: an artificially generated editing dataset (Artificial MNIST) and an avatar image manipulation with human instruction (AIMI) dataset (Chapter 3).
- We propose a neural network-based image editing model that accepts diverse editing requests for an interactive image editing system (Chapter 4).
- We propose source image masking (SIM) that identifies the region to be changed and preserves the other regions to improve the performance of the image editing (Chapter 4).
- We propose an entropy-based confirmation strategy that enables our interactive editing system to proactively confirm that the dialogue process has become effective (Chapter 5).

## 1.5. Outline of the thesis

This thesis consists of the following chapters. Chapter 3 represents the task setting of interactive image editing and introduces the data collection of an artificially generated editing dataset (Artificial MNIST) and an avatar image manipulation with human instruction (AIMI) dataset. Chapter 4 first represents our experiment for our proposed neural network-based image editing models on the Artificial MNIST and AIMI datasets. We also analyze the problem of the generated result with the AIMI dataset. Second, we introduce an experiment with source image masking (SIM) with the AIMI dataset to alleviate this problem.



Chapter 5 represents an experiment on the proposed entropy-based confirmation strategy. Chapter 6 summarizes our entire result and discusses existing limitations and future works.

# Chapter 2

## Related works

This chapter introduces a connection to other existing works related to our interactive image editing system: concurrent works and background techniques for our system.

### 2.1. Concurrent works

#### 2.1.1 Connection between interactive image editing and other vision-and-language tasks

The most conspicuous point of interactive image editing from text-to-image [42, 51] is that the former requires consistent, created images. Editing requests from users represents what and how to change the current image shared by the user and the system. The system needs to identify the region to be changed and preserve the other regions. Second, interactive image editing systems proactively engage with users. User initiative systems, such as text-to-image, force users to learn how they work through trial-and-error; interactive image editing systems can potentially reduce this trial-and-error.

Image retrieval systems are also promising for obtaining desirable images required by users. However, they can only provide existing images in the image database. They proficiently search for images that moderately match the images desired by users; they are less effective at getting images that are not in the database, for example, slightly different images.

## 2.1.2 Comparison our system and the other interactive image editing systems

Some previous works resemble our interactive image editing system. Conversational image editing systems [43, 37] understand user utterances and identify user intentions in interactive image editing tasks by existing image editing software, such as Adobe Photoshop and OpenCV [5]. These systems, supported by such editing tools, are adept at editing requests, including “delete the background” and “move the red chair.” However, these tools struggle to reflect editing requests that represent such semantic changes as “open the eyes of this face” and “make the face laugh.” Our interactive image editing system accepts these editing requests using a neural network-based image generative model. We used generative adversarial networks (GANs) [19] for an image editing module. Our editing model directly models the relationship between the changes of images and editing requests and generates edited results from scratch.

## 2.2. Background techniques

### 2.2.1 Evaluation metric for image quality

Image generation tasks, including our image editing task, need to evaluate the performance of models or systems. Structured Similarity (SSIM) [65] is a standard metric to evaluate image similarity between given two images. If a task requires to evaluate a generated image  $X$  compared with the ground truth image  $Y$ , SSIM provides the score  $SSIM(X, Y)$  as follows,

$$SSIM_{ch}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (2.1)$$

$$SSIM(X, Y) = \frac{1}{CN} \sum_{ch=1}^C \sum_{i=1}^N SSIM_{ch}(x_{i,ch}, y_{i,ch}). \quad (2.2)$$

$x_{i,ch}$  and  $y_{i,ch}$  are the  $i$ -th local patches of channel  $ch$  of  $X$  and  $Y$ . Whole patches are derived by vertically and horizontally sliding a squared window with width  $L$  one-by-one.  $\mu_x$ ,  $\mu_y$  are their mean, and  $\sigma_x^2$ ,  $\sigma_y^2$ , and  $\sigma_{xy}$  are their variance and

co-variance.  $C_1, C_2$  are constant values. For the whole experiment, we adopted commonly used parameters:  $L = 7$ ,  $C_1 = (255 \cdot 0.01)^2$ ,  $C_2 = (255 \cdot 0.03)^2$ .

Note, Inception Score [54] and Fréchet Inception Distance [22] are also well-known standard metrics to evaluate the image quality of generated images. They compare the distribution of generated image features with that of real image features, which are extracted from a well-trained image classification model, respectively. They are superior to SSIM to evaluate image fidelity and diversity of generated images; however, they do not care about a single pair of two images, and they require a high performance pre-trained classification model. Our image editing task requires to evaluate single pairs of source and target images, and the collected avatar image editing dataset has no label information on each image. Thus, we used SSIM.

### 2.2.2 Basic neural networks modeling

Neural networks are a statistical approach to model data generation process or approximate target function from given collected dataset. They have been successful in many field of computer vision such as image recognition [33, 53], object detection [18], semantic segmentation [39], action recognition [56], and image generation [50]; and natural language processing such as machine translation [4, 59], question answering [57], and chat-based dialogue system [63]. An important point of neural network is modularity. It allow us to unify different modality with specified encoders or decoders. The unified model can be optimized via back-propagation [34] and makes it possible to optimize the complex relationship between image and natural language directly.

We describe the optimization procedure for a neural network called training or learning. Suppose that we have a dataset composed of  $(\mathbf{x}, \mathbf{t})$  samples, both  $\mathbf{x}$  and  $\mathbf{t}$  are 2-dimensional data, and try to approximate a function  $f$ , where  $\mathbf{t} = f(\mathbf{x})$ , with a simple 2-layer fully-connected feed-forward network shown in Figure 2.1. A neural network is defined as input variable  $\mathbf{x}^{(1)}$ , hidden variable  $\mathbf{x}^{(2)}$ , output variable  $\mathbf{x}^{(3)}$ , and weight parameters  $w_{ij}^{(k)}$  ( $i, j, k$  are the indexes of each weight) between variables. Note that variables with value 1 are bias parameters but they can be seen as weights in this form. The approximating function is realized by optimizing these weight parameters. When input  $\mathbf{x}$  is set to  $\mathbf{x}^{(1)}$ , the neural net

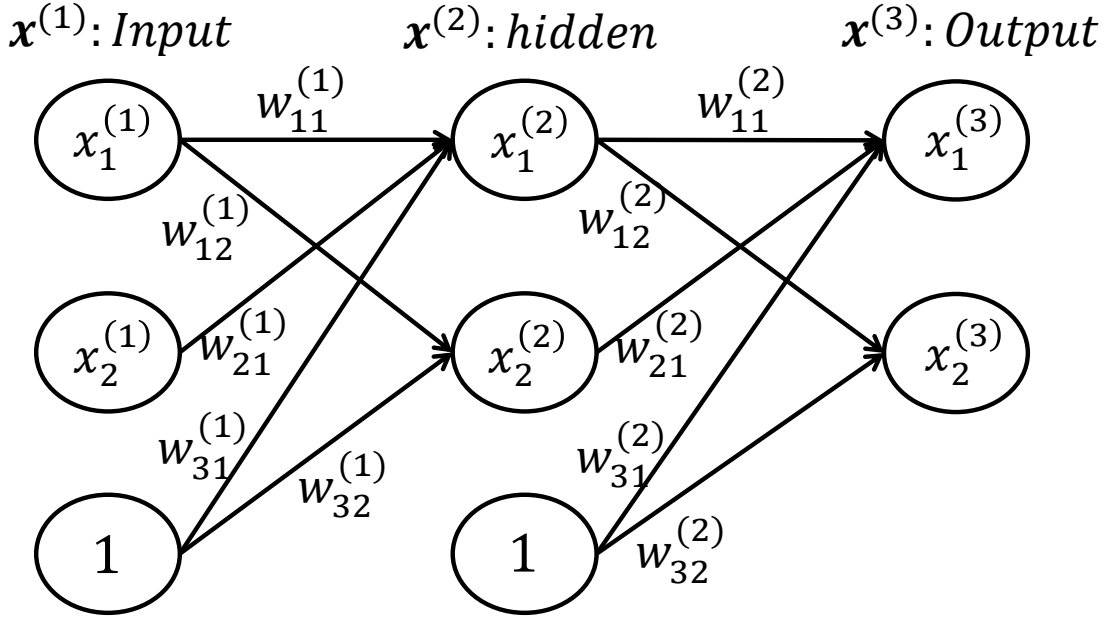


Figure 2.1. A simple 2-layer feed-forward neural network.

forwarding it to the next layer in turn as below,

$$u_j^{(k+1)} = \sum_{i=1} w_{ij}^{(k)} x_i^{(k)}, \quad (2.3)$$

$$x_j^{(k+1)} = \sigma \left( u_j^{(k+1)} \right). \quad (2.4)$$

$\sigma(\cdot)$  is activation function. There is various types of activation functions such as sigmoid,  $\text{sigmoid}(u) = \frac{1}{1+\exp(-u)}$ ; hyperbolic tangent,  $\text{tanh}(u) = \frac{1+\exp(-2u)}{1+\exp(2u)}$ ; rectified linear unit [28, 47],  $\text{ReLU}(u) = \max(0, u)$ ; leaky rectified linear unit [40],  $\text{LeakyReLU}(u) = \max(\text{slope} \cdot u, u)$ , where slope is a hyper parameter.  $\text{ReLU}(u)$  and  $\text{LeakyReLU}(u)$  are more useful to avoid gradient vanishing problem in deeper layers architecture. Therefore, we use them mainly except for the final layer of each module because of calculating objective function error or getting higher performance after investigating several settings.

After getting output variable  $\mathbf{x}^{(3)}$ , the neural network compare the difference between  $\mathbf{t}$  and  $\mathbf{x}^{(3)}$  with objective function. For example, mean squared error is

calculated as below,

$$E_{\mathbf{w}}(\mathbf{x}^{(3)}, \mathbf{t}) = \frac{1}{2M} \sum_{i=1} (x_i^{(3)} - t_i)^2, \quad (2.5)$$

where  $M$  is mini-batch size. Neural networks allows parallel processing for its training because its calculation is composed of operation between vectors and matrices. Large mini-batch size makes the training faster and more stable in general. If  $t$  is not real value but a 1-hot vector, which is a discrete representation; only one dimension is one and the others are zero, cross entropy is suitable for the objective function. The cross entropy for binary variable ( $E_{\mathbf{w}}(\mathbf{y}, \mathbf{t}) = CE(\mathbf{y}, \mathbf{t})$ ) is defined as,

$$CE(\mathbf{y}, \mathbf{t}) = -\frac{1}{M} \sum \{t \log(y) + (1 - t) \log(1 - y)\}. \quad (2.6)$$

The cross entropy for multiple variables called softmax cross entropy ( $E_{\mathbf{w}}(\mathbf{y}, \mathbf{t}) = SCE(\mathbf{y}, \mathbf{t})$ ) is defined as,

$$SCE(\mathbf{y}, \mathbf{t}) = -\frac{1}{M} \sum t \log y. \quad (2.7)$$

Using these objective functions, the neural networks can optimize its weight by back-propagation [34]. To calculate  $w_{11}^{(2)}$ , its value for optimization called gradient is defined as,

$$-\frac{\partial E_{\mathbf{w}}}{\partial w_{11}^{(2)}} = -\frac{\partial E_{\mathbf{w}}}{\partial x_1^{(3)}} \cdot \frac{\partial x_1^{(3)}}{\partial u_1^{(3)}} \cdot \frac{\partial u_1^{(3)}}{\partial w_{11}^{(2)}}. \quad (2.8)$$

In the same way, the gradient for  $w_{11}^{(2)}$  is,

$$-\frac{\partial E_{\mathbf{w}}}{\partial w_{11}^{(1)}} = -\frac{\partial E_{\mathbf{w}}}{\partial x_1^{(2)}} \cdot \frac{\partial x_1^{(2)}}{\partial u_1^{(2)}} \cdot \frac{\partial u_1^{(2)}}{\partial w_{11}^{(1)}}, \quad (2.9)$$

where

$$\begin{aligned} \frac{\partial E_{\mathbf{w}}}{\partial x_1^{(2)}} &= \frac{\partial E_{\mathbf{w}}}{\partial x_1^{(3)}} \cdot \frac{\partial x_1^{(3)}}{\partial u_1^{(3)}} \cdot \frac{\partial u_1^{(3)}}{\partial x_1^{(2)}} + \frac{\partial E_{\mathbf{w}}}{\partial x_2^{(3)}} \cdot \frac{\partial x_2^{(3)}}{\partial u_2^{(3)}} \cdot \frac{\partial u_2^{(3)}}{\partial x_1^{(2)}} \\ &= \sum_{j=1} w_{1j}^{(2)} \sigma' \left( u_j^{(3)} \right) \frac{\partial E_{\mathbf{w}}}{\partial x_j^{(3)}}. \end{aligned} \quad (2.10)$$

As we described in (2.8), (2.9), and (2.10), the neural networks can calculate the gradient of each weight by propagating the gradient of each variables in the previous layer from output layer to input layer.

### 2.2.3 Convolutional neural network (CNN)

Convolutional neural network (CNN) [35] is a kind of feed-forward neural networks, but it is not fully-connected. It is a locally-connected neural network. It has been successful in computer vision tasks. We use CNN to encode source image. Suppose that input  $x$  is an image (2-dimensional data with channel  $CH$ ), forward pass is defined as,

$$u_{ij}^k = \sum_{ch=1}^{CH} \sum_{s=0}^{m-1} \sum_{t=0}^{n-1} w_{s,t}^k x_{ch,(i+s)(j+t)} + b^k. \quad (2.11)$$

$m$  and  $n$  is local receptive field size of input  $x$  connected to  $u_{ij}^k$ .  $w_{s,t}^k$  is called kernel window because it is  $m \times n$  rectangle. Each  $u_{ij}^k$  is calculated on each local receptive field, which is determined by kernel window size, stride, and padding parameters. The stride represents the distance between two consecutive positions of the kernel. The padding represents the number of zero values concatenated at the edge of the input. The horizontal and vertical dimension of the output  $(O_w, O_h)$  is determined by input size  $(I_w, I_h)$ , kernel window size  $(K_w, K_h)$ , stride  $S$ , and padding  $P$  as follows:

$$O_w = \frac{I_w - K_w + 2P}{S} + 1, \quad (2.12)$$

$$O_h = \frac{I_h - K_h + 2P}{S} + 1. \quad (2.13)$$

Output channel is determined by the number of kernel windows  $N$ . For example, Figure 2.2 visualizes the calculation when the input is a 3-channel ( $I_{ch} = 3$ ) image with  $(I_w, I_h) = (3, 3)$ ,  $(K_w, K_h) = (3, 3)$ , stride  $S = 2$ , and padding  $P = 1$ . Padding values represent white colour with dot line in the input. In this case, input dimension is  $(I_{ch}, I_w, I_h) = (3, 3, 3)$ , and output dimension is  $(N, O_w, O_h) = (1, 2, 2)$

We use CNN for image encoder  $E_{im}$

### 2.2.4 Long-short term memory network (LSTM)

Long-short term memory network (LSTM) [24] is suitable for modeling temporary data. It has been successful in language modeling [58], encoding text [9, 10].

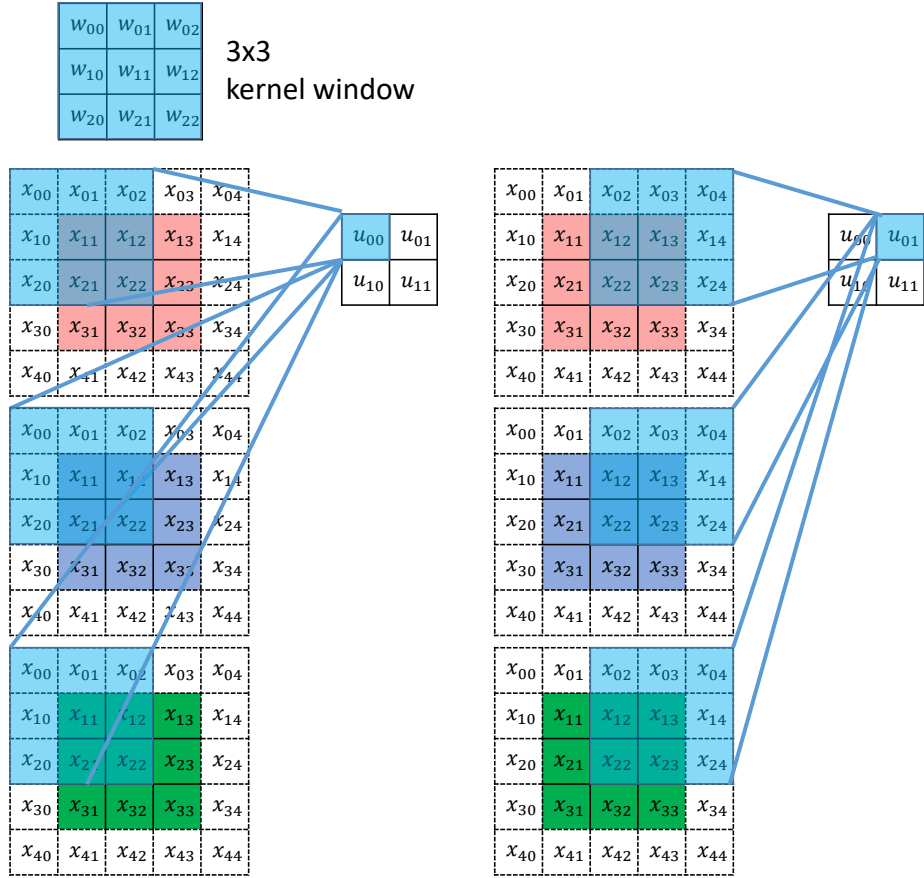


Figure 2.2. A calculation example of convolutional neural network. Input is  $3 \times 3$  squared with 3 channels. Kernel window size is  $(3, 3)$ , stride is 2, and padding is 1. Padding values represent white color with dot line in the input.

We use LSTM for encoding instruction data. Suppose  $I$  is an input sentence  $I = (x_1, x_2, \dots, x_T)$  ( $x_i$  is  $i$ -th word). LSTMs have initial value of hidden and cell variables (both are initialized by zero vectors) and update them one-by-one as the given each word sequentially. Encoding  $i$ -th word  $x_i$  is defined as,

$$\mathbf{z}_t = \tanh(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1}), \quad (2.14)$$

$$\mathbf{g}_{i,t} = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1}), \quad (2.15)$$

$$\mathbf{g}_{f,t} = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1}), \quad (2.16)$$

$$\mathbf{g}_{o,t} = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1}), \quad (2.17)$$



Each  $\mathbf{W}_\cdot$  is fully-connected layer.  $\mathbf{z}_t$  is also a hidden vector.  $\mathbf{g}_{i,t}$ ,  $\mathbf{g}_{f,t}$ , and  $\mathbf{g}_{o,t}$  are gating variables to approximate a complex function.

$$\hat{\mathbf{z}}_t = \mathbf{z}_t \odot \mathbf{g}_{i,t} \quad (2.18)$$

$$\hat{\mathbf{c}}_{t-1} = \mathbf{c}_{t-1} \odot \mathbf{g}_{f,t} \quad (2.19)$$

$$\mathbf{c}_t = \hat{\mathbf{c}}_{t-1} + \tanh(\hat{\mathbf{z}}_t) \quad (2.20)$$

$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \odot \mathbf{g}_{o,t} \quad (2.21)$$

Note that  $\odot$  indicates the Hadamard product. After processing all words, we use the final output  $\mathbf{h}_T$  as the extracted feature from the given  $I$ . To simplify this whole extraction procedure, we use  $\phi^i = E_i(I)$ , where  $\phi^i = \mathbf{h}_T$ .

## 2.2.5 Deep convolutional generative adversarial network (DCGAN)

A Deep Convolutional Generative Adversarial Network (DCGAN) [50] is a commonly used generative model for image generation. DCGAN is composed of CNN-based generator  $G$  and discriminator  $D$  for adversarial learning [19]. The generator is defined as,

$$\hat{\mathbf{X}} = G(\mathbf{z}). \quad (2.22)$$

It generates image  $\hat{\mathbf{X}}$ , which has same dimension as given training image  $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$ , from given noise  $\mathbf{z} \in \mathbb{R}^Z$  (e.g., Gaussian:  $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$ , where  $\mathbf{I}$  is an identity matrix).  $C$ ,  $H$ ,  $W$  are the dimension of channel, height, and width of an image, respectively.  $Z$  denotes the dimension of noise vector  $\mathbf{z}$ . The discriminator is defined as,

$$\hat{y} = D(\mathbf{x}). \quad (\mathbf{x} \in \{\mathbf{X}, \hat{\mathbf{X}}\}) \quad (2.23)$$

$\hat{y}$  represents a scalar output of the discriminator  $D$  with range  $[0, 1]$ . In training, the discriminator  $D$  receives two types of input  $X$  and  $\hat{X}$ , image of training sample and image generated by the generator  $G$  as shown in (2.22). It learns to predict  $\hat{y} = 1$  if  $\mathbf{x}$  is a target image  $X$  (real), or predict  $\hat{y} = 0$  if  $\mathbf{x}$  is a generated target image  $\hat{X}$  by the generator  $G$  (fake). The classified output  $\hat{y}$  will be used

to train the generator. DCGAN is optimized by the following objective:

$$\begin{aligned} \min_{\boldsymbol{\theta}_G} \max_{\boldsymbol{\theta}_D} V(G, D) &= \mathbb{E}_{\mathbf{X} \sim p_{data}} [\log D(\mathbf{X})] \\ &+ \mathbb{E}_{\mathbf{z} \sim p_z} [\log(1 - D(G(\mathbf{z})))] . \end{aligned} \quad (2.24)$$

$\boldsymbol{\theta}_G$  and  $\boldsymbol{\theta}_D$  are the trainable parameters of the generator and the discriminator.  $p_{data}$  and  $p_z$  denote the data and noise distributions. Adversarial learning resembles a mini-max game between the generator and the discriminator. The discriminator is optimized to correctly classify generated images from the generator (fake) and training examples (real). On the other hand, the generator is optimized to trick the discriminator into predicting the generated images as training examples. This competitive training improves the image modeling performance [50]. To stabilize the training, we rewrite the objectives [19] and get the following training objectives:

$$\min_{\boldsymbol{\theta}_D} \mathcal{L}_D = -\mathbb{E}_{\mathbf{X} \sim p_{data}} [\log D(\mathbf{X})] - \mathbb{E}_{\mathbf{z} \sim p_z} [\log(1 - D(G(\mathbf{z})))] , \quad (2.25)$$

$$\min_{\boldsymbol{\theta}_G} \mathcal{L}_G = -\mathbb{E}_{\mathbf{z} \sim p_z} [\log D(G(\mathbf{z}))] . \quad (2.26)$$

### Batch normalization

Batch normalization [25] is an important technique to stabilize the training of DCGAN. It normalizes an activation vector  $\mathbf{u}$  shown in (2.3) and (2.4).  $\mathbf{u} = (u_1, u_2, \dots, u_n)$  with  $n$ , and normalizes it over  $\hat{u}_k$  is defined as,

$$\hat{u}_k = \frac{u_k - \mathbb{E}[u_k]}{\sqrt{\text{Var}[u_k]}} . \quad (2.27)$$

Note that  $\mathbb{E}[u_k]$  represents  $\sum_{k=1}^n u_k$ , and  $\text{Var}[u_k]$  represents variance of  $u$ . In practice, this calculation applies to the input on each channel independently. In the test time, we use the statistics of the test batch due to better performance than the aggregated statistics of the training batch.

### Transposed convolution

An ordinary convolution layer squeezes the width and height dimension of the input; however, the CNN-based generator needs up-sampling the squeezed feature

vector to generate an image. Transposed convolution layer, which is also known as fractionally-strided convolution layer or deconvolution layer, is an inverted function of the ordinary convolution layer; thus, they are used in the generator.

To calculate transposed convolution layer, let us suppose an example of ordinary convolution layer with input size  $(I_w, I_h) = (4, 4)$ , kernel window size  $(K_w, K_h) = (3, 3)$ , stride  $S=1$ , and padding  $P = 0$ . This convolution layer operation can be expressed by a matrix operation [16]. The weight matrix  $\mathbf{C}$  for the convolution layer on each channel is defined as,

$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}. \quad (2.28)$$

Note that input and output vector is channel-wisely unrolled from left to right, top to bottom in this situation. The output dimension is 4, which means output size  $(O_w, O_h) = (2, 2)$  after reshaping. Transposed convolution layer is defined by  $\mathbf{C}^T$ .

## 2.2.6 Conditional generation with DCGAN for caption-to-image generation

The original DCGAN is an unconditional generation model. It was expanded to conditional image generation with given condition such as semantic labels [49] (e.g., dog, cat) and captions [51] (e.g., “This bird has a yellow beak and black body.”). Our image editing task is a conditional generation with a source image and natural language instruction. Our editing model is based on the caption-to-image generation [51] because they are not semantic label expression.

This caption-to-image generation model [51] consists of three modules: a LSTM-based caption encoder  $E_c$  to extract features of input captions, generator  $G$ , and discriminator  $D$ .  $G$  and  $D$  are almost same modules as the unconditional

DCGAN. The modifications from (2.22) and (2.23) are as follows,

$$\hat{\mathbf{X}} = G(\mathbf{z}, \boldsymbol{\phi}), \quad (2.29)$$

$$\hat{y} = D(\mathbf{x}, \boldsymbol{\phi}). \quad (x \in \{\mathbf{X}, \hat{\mathbf{X}}\}). \quad (2.30)$$

$\boldsymbol{\phi}$  is a conditional feature extracted from the input caption through the LSTM-based caption encoder  $E_c$  as described in Section 2.2.4.

For training of the conditional DCGAN, the discriminator must learn to distinguish not only whether its input image is real or generated but also whether its input image is matched to the given condition. Matching aware method [51] realizes this function by modifying the conditional DCGAN objectives described in Equation (2.25) and (2.26) as follows,

$$\mathcal{L}_{D_{X_r}} = -\mathbb{E}_{\mathbf{X} \sim p_{data}} [\log D(\mathbf{X}, \boldsymbol{\phi}_r)], \quad (2.31)$$

$$\mathcal{L}_{D_{X_w}} = -\mathbb{E}_{\mathbf{X} \sim p_{data}} [\log(1 - D(\mathbf{X}, \boldsymbol{\phi}_w))], \quad (2.32)$$

$$\mathcal{L}_{D_{\hat{X}_r}} = -\mathbb{E}_{\mathbf{X} \sim p_{data}} \mathbb{E}_{\mathbf{z} \sim p_z} [\log(1 - D(G(\mathbf{z}, \boldsymbol{\phi}_r), \boldsymbol{\phi}_r))], \quad (2.33)$$

$$\mathcal{L}_{G_{\hat{X}_r}} = -\mathbb{E}_{\mathbf{X} \sim p_{data}} \mathbb{E}_{\mathbf{z} \sim p_z} [\log D(G(\mathbf{z}, \boldsymbol{\phi}_r), \boldsymbol{\phi}_r)]. \quad (2.34)$$

The right condition  $\boldsymbol{\phi}_r$  is matched to the real image  $\mathbf{X}$  or the generated image  $\hat{\mathbf{X}}$ . In other words,  $\boldsymbol{\phi}_r$  is extracted from the caption aligned with  $\mathbf{X}$  or is identical to the input condition to generate  $\hat{\mathbf{X}}$  through  $\hat{\mathbf{X}} = G(\mathbf{z}, \boldsymbol{\phi}_r)$ . On the other hand, the wrong condition  $\boldsymbol{\phi}_w$  represents mismatched condition to the real image  $\mathbf{X}$  or the generated image  $\hat{\mathbf{X}}$ . In practice,  $\boldsymbol{\phi}_w$  is created by randomly permutating  $\boldsymbol{\phi}_r$  in a mini-batch. Objective (2.31) encourages the discriminator to classify a real image and a matched image-condition pair as real. Objective (2.32) encourages the discriminator to classify a real image but a mismatched pair as fake. Objective (2.33) encourages the discriminator to classify a generated image where the pair is matched as fake. Objective (2.34) encourages the generator to trick the discriminator into classifying the generated image with the matched pair as real.



# Chapter 3

## Interactive image editing

This chapter introduces a setting for interactive image editing task.

### 3.1. Task setting

First, we describe the interactive image editing dialogue task. It has a human user and a system. The dialogue’s purpose is to generate goal image  $X^g$ , which is the user’s designed image, through a dialogue. The user makes natural language instruction  $I$ , which represents an editing request to change the current image closer to the goal. The system generates a new image based on the previous image when the user requests to change.

- Step 1 First source image  $X_0^s$  and goal image  $X^g$  are given to the user.
- Step 2 At the  $i$ -th turn interaction, the user makes a natural language instruction  $I_i$  to edit previous image  $X_{i-1}^s$ .
- Step 3 The system generates a new image  $X_i$  based on the instruction  $I_i$  and the previous image  $X_{i-1}^s$ .
- Step 4 The system resets  $X_i$  as new source image  $X_i^s$ , and the user chooses whether to continue the dialogue. If the user decides to continue, go to Step 2 with  $i += 1$ . If the user decides to stop the dialogue, the dialogue is finished, and the image  $X_i^s$  is compared with goal image  $X^g$  to evaluate their similarity.

Note that since the goal image is invisible to the system, it cannot be optimized directly to generate the goal image. At the end of the task, the user and the system receive a similarity score between the final source image  $X_i^s$  goal image  $X^g$ . It represents the performance of the task. In this thesis, we used Structured Similarity (SSIM) [65] for the similarity score as described in Section 2.2.1.

If we have several image generators on Step 3), the system must choose one image as a new image of  $X_i$ . When the system cannot choose between images, one solution is to seek confirmation from the user about which image is better. We assume that the system has multiple image candidates  $X_{i,1}, X_{i,2}, \dots, X_{i,n}$  in Step 3) and two choices:  $\{confirm, not\ confirm\}$ . If it selects *confirm*, the following sub-steps of the confirmation procedure are inserted before Step 4):

- 3-c1) The system shows image candidates to the user to confirm which image is relevant to the request.
- 3-c2) The user selects the most relevant image. The system sets the selected image as its generated image of  $X_i$ .

Figure 3.1 summarizes a single  $i$ -th turn that is defined as a sequential process to decide next source image  $X_i^s$  from current source image  $X_{i-1}^s$ . Since the confirming steps make the interaction redundant, the system has to reduce the number of confirming actions. Criteria exist upon which the system selects *confirm* or *not confirm* (see Section 5.3).

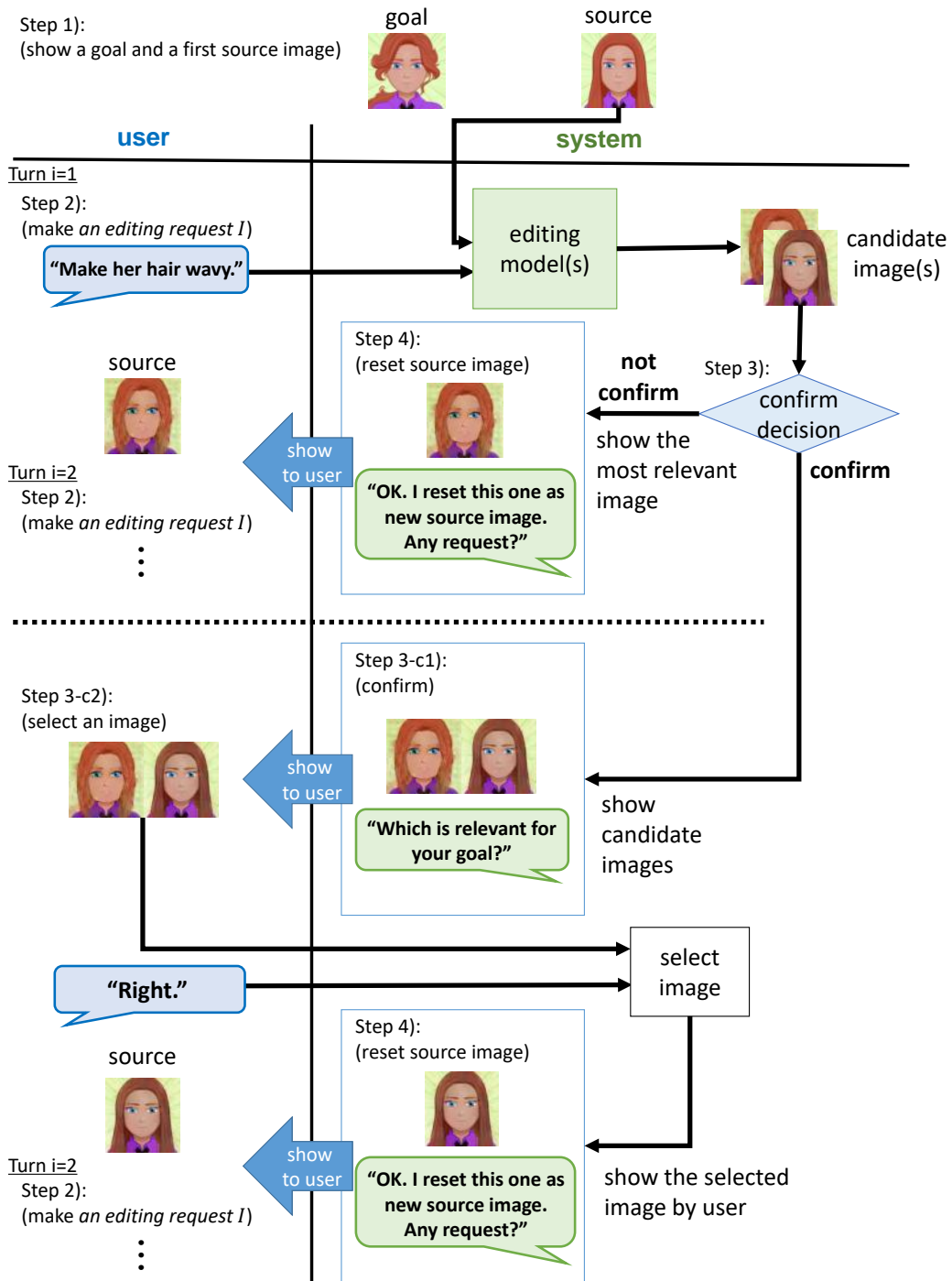


Figure 3.1. Overview of interactive image editing.



## 3.2. Data collection

We created two datasets for our interactive image generation task. First, MNIST editing with artificial instruction (Artificial MNIST) dataset is an automatically generated dataset made from original handwritten MNIST dataset [35]. The instructions of this dataset, natural language instructions, are also annotated automatically. We expect this dataset enables us to analyze our proposed model easily. Second, Avatar image editing with human instruction (AIMI) dataset is automatically collected dataset from an avatar creation website. The instructions of this dataset are manually collected from a human via crowd-sourcing. This dataset is more practical than Artificial MNIST because it has a larger variety of both images and instructions.

### 3.2.1 MNIST editing with artificial instruction (Artificial MNIST) dataset

We describe the procedure to build MNIST editing with artificial instruction (Artificial MNIST) dataset, which is automatically created from MNIST dataset [35]. Artificial MNIST consists of triplets of (source image, target image, instruction). Figure 3.2 represents the procedure to generate the triplets from original MNIST samples automatically. First, we randomly selected 1,000 samples on each digit zero to nine from the original MNIST training set of 60,000 samples and obtained 10,000 samples in total. Of course, we can use a more substantial size of samples; however, 10,000 samples is enough for our experiment in minimum training time cost. On each sample, we generate a triplet as below,

- step1 We render source image to a canvas from given source parameter set ( $digit, x, y, w, h$ ).
- step2 We calculate target parameter set from the source parameter set and given instruction composed of a triplet of ( $action, digit, direction$ ).
- step3 We verify that the target parameter set is possible to render. If it is impossible, we cease creating this triplet.
- step4 If the target parameter set is possible to render, we render the target image to a canvas from the target parameter set.

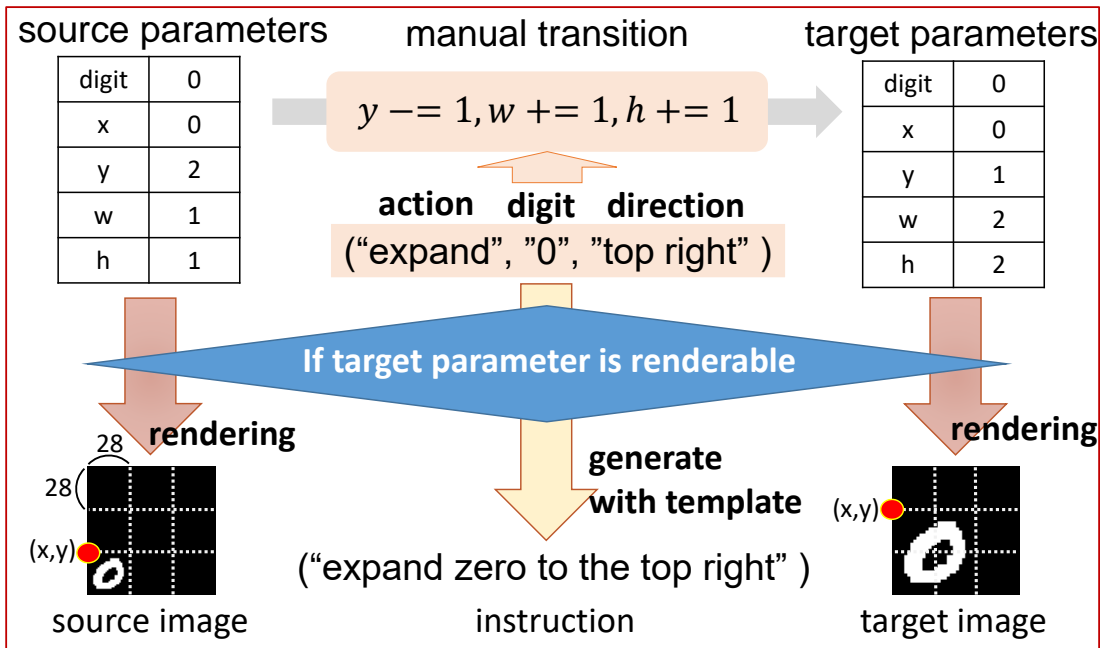


Figure 3.2. Generation procedure for Artificial MNIST dataset.

Note that the canvas has a size of 84x84 pixel that is three times larger than that of the original MNIST samples (size of 28x28 pixel). In source and target parameter set, *digit* is identical to the digit class of the original MNIST sample. We considered the canvases that has no digit, whose rendering parameter is defined by  $(x, y, w, h) = (0, 0, 0, 0)$ , as additional digit class; therefore, we have 11-class of *digit*.  $(x, y)$  represents horizontal and vertical position, and has  $\{3,3\}$ -class, and  $(w, h)$  represents width and height of image, and has  $\{4,4\}$ -class. Appendix A describes the examples of variables to render source and target image pairs with instruction. We tried all possible source parameter sets and got 31 unique images on each digit sample.

Figure 3.3 represents the rendering procedure detail for the source images at step1. Suppose that we have a sample of digit zero to render in the canvas. The valid rendering parameter set  $(0, x, y, w, h)$  must satisfy  $1 \leq w \leq 3$  and  $1 \leq h \leq 3$ . In rendering, we horizontally expand the digit zero to be  $w$  times as large as the original one and vertically expand it to be  $h$  times as well. If  $w = 1$  or  $h = 1$ , there are three choices to render as ["left", none, "right"] or ["top", none, "bottom"]. If

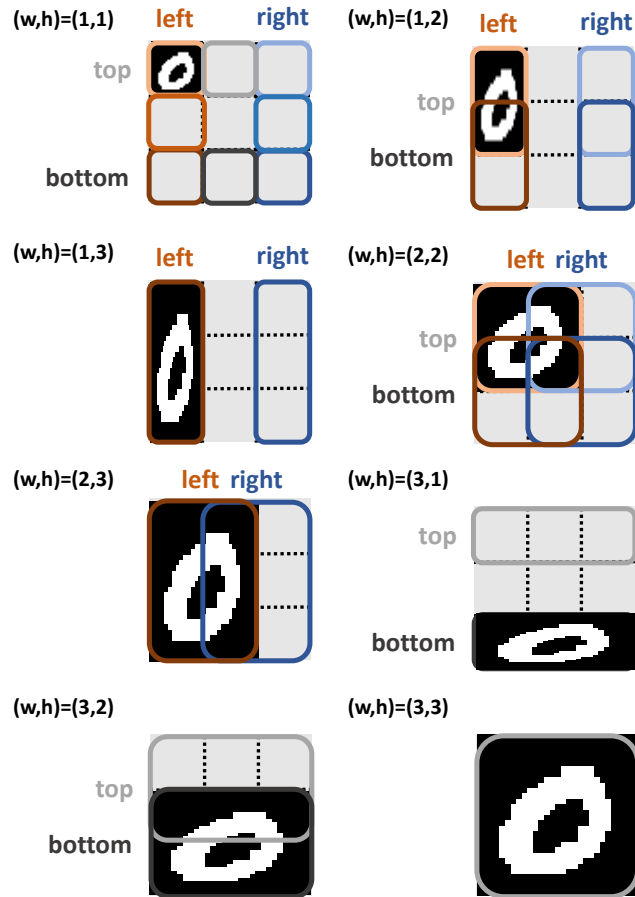


Figure 3.3. Rendering examples.  $(w, h)$  denotes resizing factors for width and height of the original MNIST images.

$w = 2$  or  $h = 2$ , we have two choices on each [*left*, *right*] or [*top*, *bottom*]. If  $w = 3$  or  $h = 3$ , we have no choice to the direction. On each  $(w, h)$ , We tried all combination of vertical and horizontal choices.

Figure 3.4 represents the rendering procedure detail to render the target image from step2 to step4. We have action set {*put*, *remove*, *expand*, *compress*, *move*}, direction set {*top*, *left*, *right*, *bottom*, *top left*, *top right*, *bottom left*, *bottom right*}. Each  $(x, y, w, h)$  denotes the source parameter set. Command *expand* calculates the target parameter set from given the source parameter set. It enlarges the source image to the direction corresponding

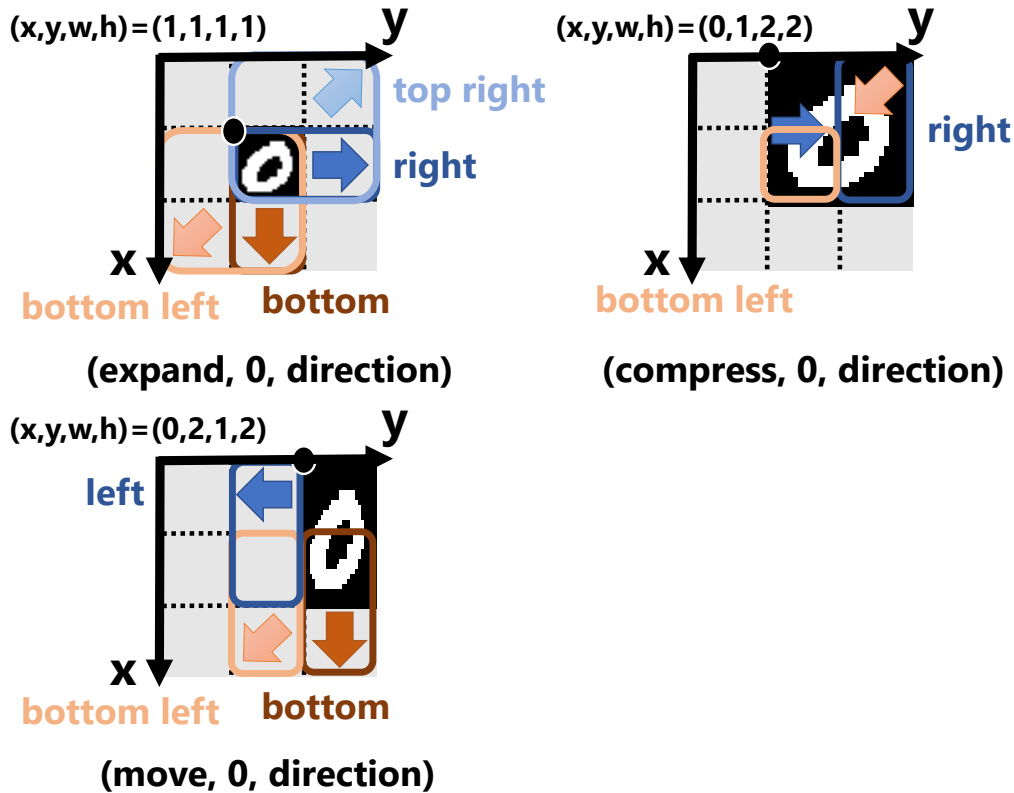


Figure 3.4. MNIST sample transformation with manual transition corresponding to the given instruction command  $(action, digit, direction)$ .

to the “*direction*.” In the same way, command “*compress*” makes the source image smaller. Command “*move*” keeps the size of the source image and shift it to the “*direction*.”

Finally, we got 369 patterns of triplets on each sample and got 3,690,000 triplets in total.

### 3.2.2 Avatar image editing with human instruction (AIMI) dataset

We constructed a dataset for IMI task, which consists of triplets: (source image, target image, instruction). We collected avatar image pairs in AvatarMaker.com<sup>1</sup>.

<sup>1</sup><http://avatarmaker.com/>

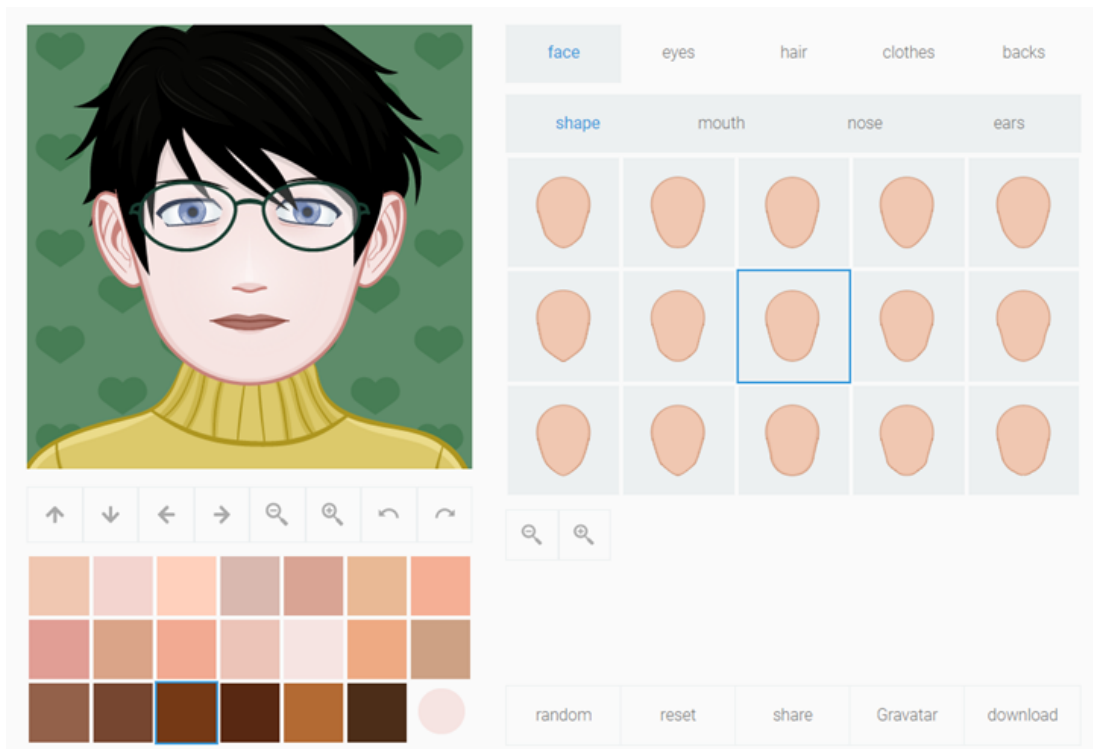


Figure 3.5. Avatar creation website for collecting avatar images.

Figure 3.5 shows a screenshot of avatar creation graphical user interface in the website. An avatar has various attributes, i.e., gender, eyes, mouth, background, and so on. The construction procedure of the dataset is as follows:

- An source image is generated with random attributes.
- One attribute of the source image is changed to generate a target image.
- Both of the source image and the target image are shown to crowd workers on crowd-sourcing, and they add an instruction that describes the difference between the source and target image.

Moreover, we automatically normalized the first character into lowercase and removed period. In practice, Figure 3.6 denotes the actual instruction scripts shown to crowd-sourcing workers. Some target images are very similar to the source images; thus, it is hard for the workers to discriminate them in seconds. To

## Overview

In this task, you will describe an **instruction**, which shows the request to modify the **source image** to the **target image**. The target image shows if you put your mouse pointer on the (source) image. Please imagine a situation you are a client and ask designer to modify current (source) image to desired (target) image.

## Steps

1. View the source image
2. Compare the target image with source image by putting or removing mouse pointer on the image
3. Find a difference between the source and target image
4. **Select a part of face** that you find is different
5. Enter an **instruction**

## Rules Tips

- The instruction should be **one sentence**.
- The instruction should be **imperative form**.
- The instruction should be written by **natural language manner**. Please **avoid** to fill out the instruction by a symbolic sequence, for example, "Eye, color, brown"
- The instruction should include the descriptive details of the difference. For example, **Bad**: "put a glasses." **Good**: "put a glasses with clear lenses whose frame color is green." The minimum character length of instruction is 15.
- **Don't use any proper names**, for example, "Make her hair color like Emma Watson."

## Instruction Examples


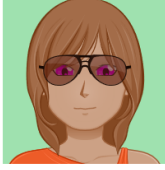




Source	Target (with mouse over)	Different part selection instruction
		<p>What part is different between the images? (required)</p> <ul style="list-style-type: none"> <li><input type="radio"/> Hair (on head)</li> <li><input checked="" type="radio"/> Around Eyes</li> <li><input type="radio"/> Ears</li> <li><input type="radio"/> Nose</li> <li><input type="radio"/> Around Mouth</li> <li><input type="radio"/> Face shape</li> <li><input type="radio"/> Clothes</li> </ul> <p>Provide an instruction to modify the source image to the target image. (required)</p> <input type="text" value="put a sunglasses with brown lenses and black frame."/>
		<p>What part is different between the images? (required)</p> <ul style="list-style-type: none"> <li><input type="radio"/> Hair (on head)</li> <li><input checked="" type="radio"/> Around Eyes</li> <li><input type="radio"/> Ears</li> <li><input type="radio"/> Nose</li> <li><input type="radio"/> Around Mouth</li> <li><input type="radio"/> Face shape</li> <li><input type="radio"/> Clothes</li> </ul> <p>Provide an instruction to modify the source image to the target image. (required)</p> <input type="text" value="make the transparency of glasses lower and frame thinner."/>
		<p>What part is different between the images? (required)</p> <ul style="list-style-type: none"> <li><input type="radio"/> Hair (on head)</li> <li><input type="radio"/> Around Eyes</li> <li><input type="radio"/> Ears</li> <li><input type="radio"/> Nose</li> <li><input type="radio"/> Around Mouth</li> <li><input checked="" type="radio"/> Face shape</li> <li><input type="radio"/> Clothes</li> </ul> <p>Provide an instruction to modify the source image to the target image. (required)</p> <input type="text" value="make his cheeks slim."/>

Figure 3.6. Crowd-sourcing instruction scripts shown to crowd workers.

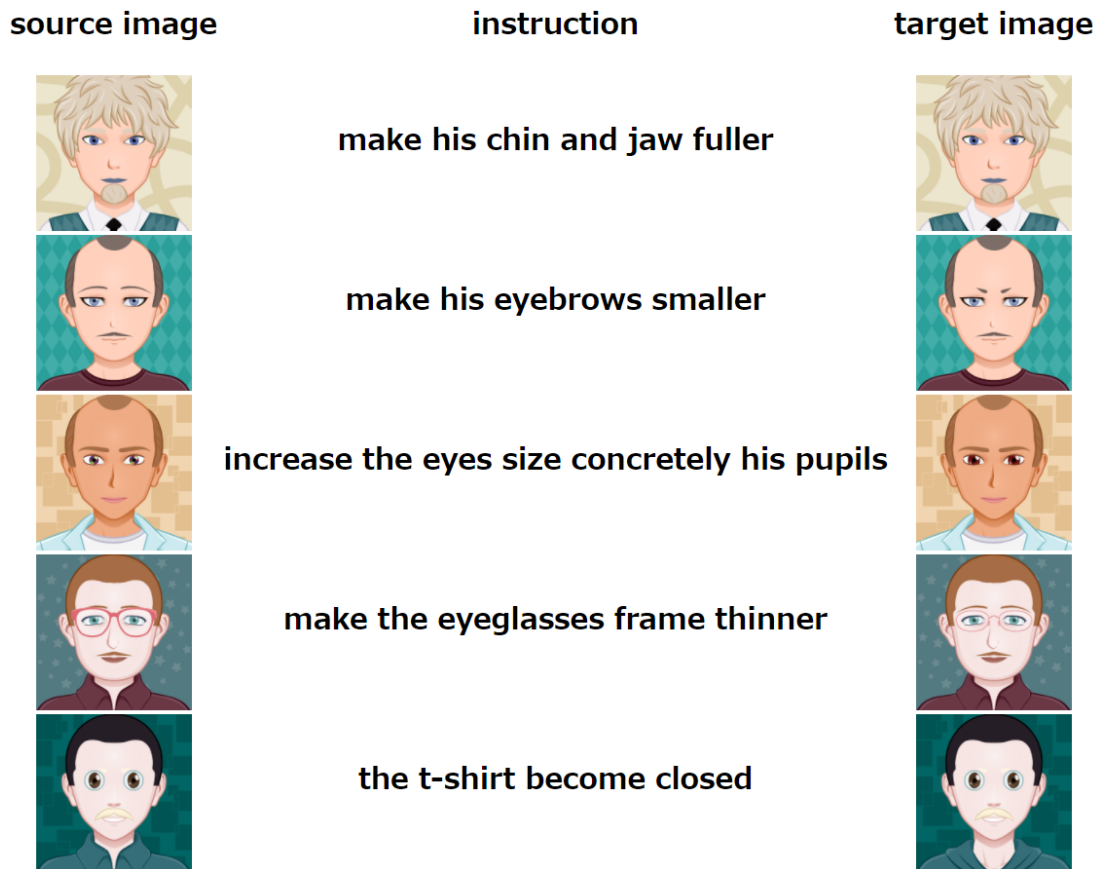


Figure 3.7. Annotated instruction examples via crowd-sourcing.

alleviate this problem, we added a function on the target images with the mouse pointer. The workers can alternatively see the target and the source image with putting and removing mouse pointer on the second column images, respectively. To control annotation quality, we also showed several annotation rules to the workers as follows.

1. The instruction should be one sentence.
2. The instruction should be an imperative form.
3. The instruction should be written by natural language manner. Please avoid to fill out the instruction by a symbolic sequence, for example, “Eye, colour, brown.”

4. The instruction should include the descriptive details of the difference. For example, Bad: “put a glasses.” Good: “put a glasses with clear lenses whose frame colour is green.” The minimum character length of instruction is fifteen.
5. Do not use any proper names, for example, “Make her hair colour like Emma Watson.”

After the data collection, we got 12,000 examples in total. We found terrible annotated examples, e.g., written by other languages such as Spanish or Russian, annotated instruction represents the inverted change from target to source. We removed or corrected them manually, and finally, we got 4,756 valid examples of (source image, target image, instruction) triplets as shown in Figure 3.7. We divided the data into train, validation, and test set in proportions of 90%, 5% and 5%, namely 4,296, 230, and 230 examples. We also collected randomly generated images without any instructions – total size: 161,065.





# Chapter 4

## Image editing with natural language instruction

This chapter describes our proposed method for the interactive image editing system with natural language instruction (editing request), which can generate a target image from a source image and an instruction sentence that describes the difference between the source and the target image. The system makes it possible to modify a generated image interactively and make natural language conditioned image generation more controllable. We construct a neural network that handles image vectors in latent space to transform the source vector to the target vector by using the vector of instruction. Additionally, we propose source image masking (SIM), a method for masking source image to clarify the changing points of the generated image from the instruction. The experimental results indicate that the proposed framework successfully generates the target image using a source image and instruction on editing in artificial MNIST and avatar dataset. Moreover, we show that SIM makes the training more stable and faster.

### 4.1. Introduction

Specialized skills are required to create a commercially available image. One way to obtain a required image at low cost is by finding existing images through an image search. However, it is difficult to obtain what was exactly imagined since the desired image may not exist on the Web. An automatic image-generation system

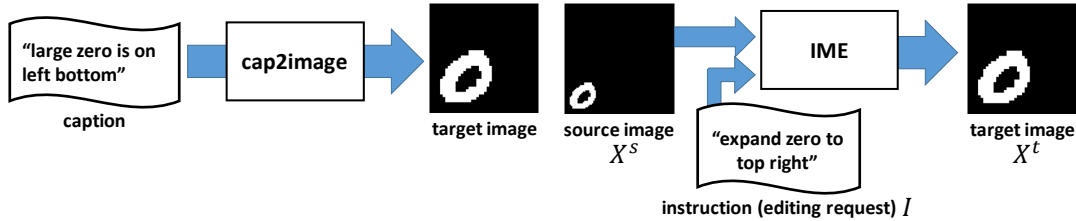


Figure 4.1. Comparison of natural language conditioned image generation framework between the existing cap2image (left) and the proposed image manipulation with instruction (IMI) (right).

using natural language has a potential to generate what was actually imagined without requiring any special skill or cost. The solution to this challenging task should address not only practical benefits but also contributions of bridging natural language understanding with image processing. Image generation task from natural language have been investigated as “cap2image” and several deep neural networks (DNN)-based generative models are successful [42, 51]. Although it is difficult to define the relationships between languages and images clearly, DNN-based models make it possible to align these relationships in the latent space. The network is composed of *language-encoder* and *image-decoder*. Long short-term memory (LSTM) [24] is generally used as a *language-encoder*, and several network structures; Variational auto-encoder [30], generative adversarial network (GAN) [19], and pixelCNN [61] are used as an *image-decoder*.

To the best of our knowledge, Reed et al. [51] firstly succeeded to construct a discriminable image generator conditioned by a caption based on a deep convolutional generative adversarial network (DCGAN) [50]. We start at this work from a different viewpoint; we focus on a practical problem in this task. It is possible that they generate a slightly different image from what the user actually wanted. Our motivation is to tackle this point by introducing an interactive editing framework and make a generated image modifiable with natural language.

Figure 4.1 shows the difference between the cap2image framework and the proposed framework. Compared with the cap2image framework models, the proposed framework model generates a new image from the source image and the instruction that represents the difference between the source and the target im-

age. We believe the advantage of the proposed framework is to allow users to modify the source image that has been generated. Furthermore, users only have to focus on the difference and represent it as a natural language. It is not only more comfortable for the user to use but also easier for *language-encoder* to learn because the instruction with a few different information will be much shorter than caption with all information of the desired image. We define a latent space composed of image feature vectors and set a problem of editing as a transformation of a vector in latent space. The manipulated image is generated from the latent vector that is transformed from the latent vector of the original image by the embedded natural language instruction.

Kiros et al. [31] reported that it is possible to learn a model whose shared latent space between languages (captions) and images in a DNN has the characteristic of additivity. Reed et al. [52] reported that it is possible to generate the target image using image analogy. According to these properties, we realize the image editing system by bridging the analogy in the latent space of the image and natural language instruction, as  $\{source\ image\} + "instruction" = \{target\ image\}$ . We confirm that there are many related works to edit image flexibly using user hand-drawing [7, 67]. However, manipulating images with the natural language will be useful to get a moderate image easily if we can bridge the natural language and modification, which contains many drawing operations.

Each source image on the bottom-left side has a different digit object (zero and one) and different position and size; however, the difference to their target images are the same, "expand [digit] to the left bottom." After the training, a similar analogy will have similar vector subtractions that are embedded instruction.

## 4.2. Model architecture

Figure 4.2 represents network architectures of the proposed IMI framework to generate an image from a source image and a language instruction. From left to right, each model is used for (a) Artificial MNIST editing, (b) avatar editing, and (c) avatar editing with source image masking. The framework is composed of an encoder and a decoder (image generator). In this section, we describe our proposed methods to construct our editing models based on the existing works

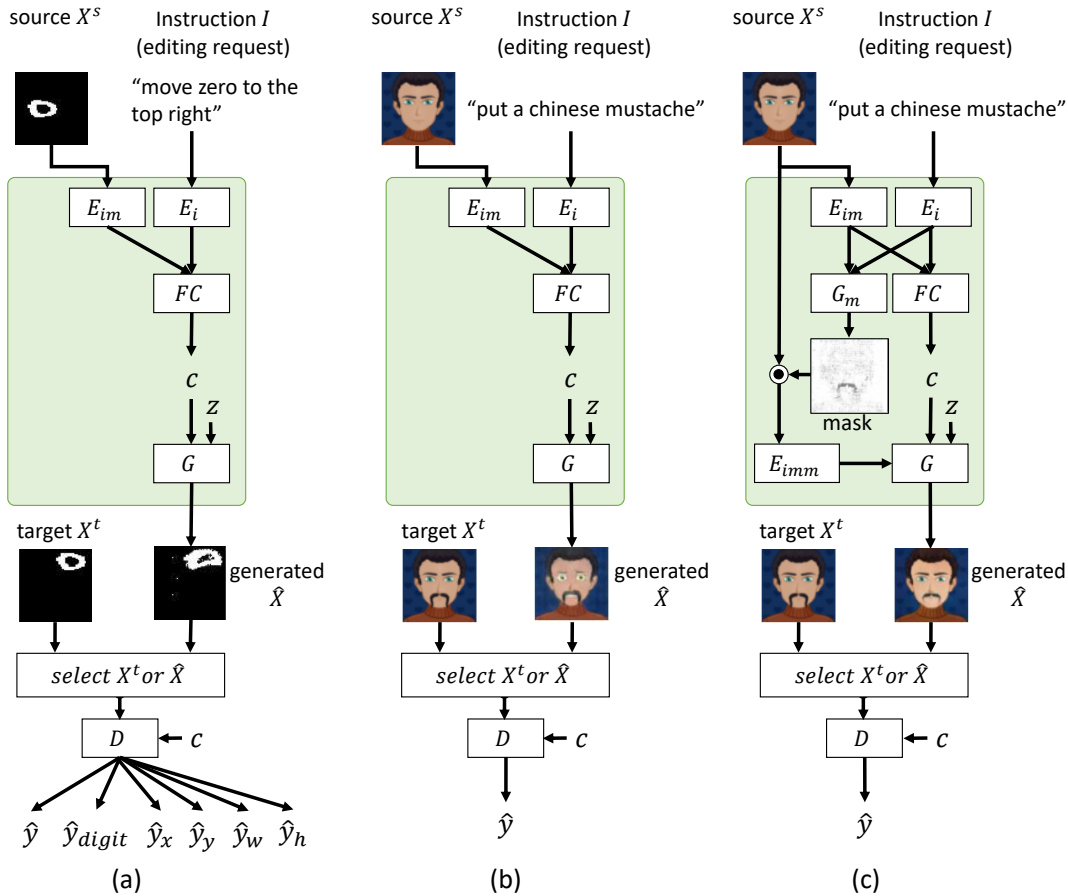


Figure 4.2. IMI model architectures for (a) Artificial MNIST editing, (b) avatar editing, and (c) avatar editing with source image masking.

described in 2.2.2 to 2.2.6.

#### 4.2.1 Our modification of conditional DCGAN for semantic image editing with instruction

We propose a conditional DCGAN for semantic image editing with natural language instruction based on the conditional DCGAN for caption-to-image generation [51]. Our image editing task needs to control a generated image not only based on textual information (instruction) but also given source image. For this reason, we modify the encoder part. It learns a function  $\phi = f(\mathbf{X}^s, I)$  that infers

target image feature  $\phi$  from the unified representation of source image  $\mathbf{X}^s$  and instruction  $I$ . The encoder part consists of source image encoder  $E_{im}$ , instruction (editing request) encoder  $E_i$ , and a 1-layer fully-connected layer  $FC$ . Function  $\phi = f(\mathbf{X}^s, I)$  is defined:

$$\phi^{im} = E_{im}(\mathbf{X}^s), \quad (4.1)$$

$$\phi^i = E_i(I), \quad (4.2)$$

$$\begin{aligned} \phi &= f(\mathbf{X}^s, I) \\ &= FC(\phi^{im}, \phi^i) \\ &= \text{sigmoid}(\mathbf{W}_{im}\phi^{im} + \mathbf{W}_i\phi^i). \end{aligned} \quad (4.3)$$

We used 4-layer convolutional neural networks [35] for  $E_{im}$  and 1-layer long short-term memory neural networks [24] for  $E_i$ .

For the training, we also modify the procedure of creating the wrong condition  $\phi_w$  on the matching aware method. An image editing model needs two abilities: it can edit a source image according to a given instruction, and it can preserve the source image if the instruction is meaningless. For this reason, we use multiple types of triplet  $(\phi_r, \phi_w, \mathbf{X})$  to calculate Equation (2.31) to (2.34). Suppose that training examples are composed of a triplet of source image  $\mathbf{X}^s$ , target image  $\mathbf{X}^t$ , and instruction  $I$ . The patterns of the triplet  $(\phi_r, \phi_w, \mathbf{X})$  have three types:  $(f(\mathbf{X}^s, I), f(\mathbf{X}^s, \emptyset), \mathbf{X}^t)$ ,  $(f(\mathbf{X}^s, \emptyset), f(\mathbf{X}^t, \emptyset), \mathbf{X}^s)$ , and  $(f(\mathbf{X}^t, \emptyset), f(\mathbf{X}^s, \emptyset), \mathbf{X}^t)$ .  $I = \emptyset$  denotes  $\phi^i = \mathbf{0}$  in practice and it represents a meaningless instruction. The first type  $(\phi_r, \phi_w, \mathbf{X}) = (f(\mathbf{X}^s, I), f(\mathbf{X}^s, \emptyset), \mathbf{X}^t)$  encourages the model to learn to edit the source image according to the instruction. It forces the edited image to be far from the source image. The second type  $(\phi_r, \phi_w, \mathbf{X}) = (f(\mathbf{X}^s, \emptyset), f(\mathbf{X}^t, \emptyset), \mathbf{X}^s)$  and the third type  $(\phi_r, \phi_w, \mathbf{X}) = (f(\mathbf{X}^t, \emptyset), f(\mathbf{X}^s, \emptyset), \mathbf{X}^t)$  encourage the model to learn to preserve the source image with a meaningless instruction.

The overall objectives to update are the following two parts:

$$\min_{\theta_D, \theta_{Enc}} \mathcal{L}_D = \lambda_{X_r} \mathcal{L}_{D_{X_r}} + \lambda_{X_w} \mathcal{L}_{D_{X_w}} + \lambda_{\hat{X}_r} \mathcal{L}_{D_{\hat{X}_r}}, \quad (4.4)$$

$$\min_{\theta_G, \theta_{Enc}} \mathcal{L}_G = \lambda_{g_{\hat{X}_r}} \mathcal{L}_{G_{\hat{X}_r}} + \lambda_f \mathcal{L}_{fmatch}. \quad (4.5)$$

The parameters  $\theta_D$ ,  $\theta_G$ , and  $\theta_{Enc}$  are the trainable parameters of  $D$ ,  $G$ , and the encoder part, respectively. In each iteration, we use an alternate update rule

[54]. The model uses Equation (4.4) if  $\mathcal{L}_D > \mathcal{L}_G$ , and otherwise it uses Equation (4.5). The additional objective  $\mathcal{L}_{fmatch}$  represents the objective of the feature matching [54] to stabilize the training of G and D. It is achieved by the sum of the mean squared error between the latent features extracted from real image  $\mathbf{X}$  and generated image  $\hat{\mathbf{X}}$ , which corresponds to  $\mathbf{X}$ , on each layer of  $D$ .  $\lambda_{Xr}$ ,  $\lambda_{Xw}$ ,  $\lambda_{\hat{X}r}$ ,  $\lambda_{g\hat{X}fr}$ , and  $\lambda_f$  are the coefficients of each objective. We use 1.0 for each coefficient.

### Auxiliary classifiers for Artificial MNIST editing

It has been reported that additional classification task by adding classifiers to the discriminator stabilizes GAN learning if the dataset contains additional labels to apply classification [49, 54]. Artificial MNIST has labels of digit, digit position  $(x, y)$ , digit size  $(w, h)$ ; therefore, we add classifiers for them. With the target label  $\mathbf{t} = [\mathbf{t}_{digit}, \mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_w, \mathbf{t}_h]$ , the additional objective  $\mathcal{L}_{label}(\phi^d, \mathbf{t})$  for the classifiers are defined as below,

$$\begin{aligned} \mathcal{L}_{label}(\phi^d, \mathbf{t}) &= \mathcal{L}_{digit}(\phi^d, \mathbf{t}_{digit}) \\ &+ \mathcal{L}_x(\phi^d, \mathbf{t}_x) + \mathcal{L}_y(\phi^d, \mathbf{t}_y) \\ &+ \mathcal{L}_w(\phi^d, \mathbf{t}_w) + \mathcal{L}_h(\phi^d, \mathbf{t}_h), \end{aligned} \quad (4.6)$$

$$\mathcal{L}_{digit}(\phi^d, \mathbf{t}_{digit}) = SCE(\mathbf{W}_{digit}\phi^d, \mathbf{t}_{digit}), \quad (4.7)$$

$$\mathcal{L}_x(\phi^d, \mathbf{t}_x) = SCE(\mathbf{W}_x\phi^d, \mathbf{t}_x), \quad (4.8)$$

$$\mathcal{L}_y(\phi^d, \mathbf{t}_y) = SCE(\mathbf{W}_y\phi^d, \mathbf{t}_y), \quad (4.9)$$

$$\mathcal{L}_w(\phi^d, \mathbf{t}_w) = SCE(\mathbf{W}_w\phi^d, \mathbf{t}_w), \quad (4.10)$$

$$\mathcal{L}_h(\phi^d, \mathbf{t}_h) = SCE(\mathbf{W}_h\phi^d, \mathbf{t}_h). \quad (4.11)$$

$SCE(\mathbf{y}, \mathbf{t})$  denotes softmax cross entropy described in Equation (2.7).  $\phi^d$  is the output of the previous layer of the last layer of the discriminator. Rewriting Equation (4.4) and (4.5), the final objectives in Artificial MNIST editing are defined as,

$$\min_{\theta_D, \theta_{Enc}} \mathcal{L}_D = \lambda_{Xr}\mathcal{L}_{D_{Xr}} + \lambda_{Xw}\mathcal{L}_{D_{Xw}} + \lambda_{\hat{X}r}\mathcal{L}_{D_{\hat{X}r}} + \mathcal{L}_{label}, \quad (4.12)$$

$$\min_{\theta_G, \theta_{Enc}} \mathcal{L}_G = \lambda_{g\hat{X}r}\mathcal{L}_{G_{\hat{X}r}} + \lambda_f\mathcal{L}_{fmatch} + \mathcal{L}_{label}. \quad (4.13)$$

### 4.2.2 source image masking (SIM)

The image editing model based on DCGAN, as shown in Figure 4.2 (b) sometimes offers drastic changes to the source image, which are inappropriate to a cooperative process with users. To prevent this problem, we propose an additional module called source image masking (SIM), which functions as a constraint on DCGAN for image editing. The modified model with SIM is shown in Figure 4.2 (c). The SIM idea is to explicitly indicate the editing points on the source image with masking. SIM consists of two parts, mask generator  $G_m$  and image encoder with mask  $E_{imm}$ . We next define the procedure for generating and forwarding a mask:

$$\mathbf{m}_{mono} = G_m(\phi^{im}, \phi^i), \quad (4.14)$$

$$\phi^{imm} = E_{imm}(\mathbf{X}^s \odot \mathbf{m}_{color}). \quad (4.15)$$

$\mathbf{m}_{color}$  is a channel-wise copied mask from mono-channel mask  $\mathbf{m}_{mono}$ .  $\odot$  indicates the Hadamard product.  $\phi^{imm}$  is fed into  $G$  as additional input. Rewriting (2.29), we get,

$$\hat{\mathbf{X}} = G(\mathbf{z}, \phi, \phi^{imm}). \quad (4.16)$$

For the training, the generator part objective described in Equation (4.5) includes optimization of the parameters of  $G_m$  and  $E_{imm}$  as follows,

$$\min_{\theta_G, \theta_{Enc}, \theta_{SIM}} \mathcal{L}_G = \lambda_{g\hat{\mathbf{X}}r} \mathcal{L}_{G_{\hat{\mathbf{X}}r}} + \lambda_f \mathcal{L}_{fmatch}. \quad (4.17)$$

The parameter set  $\theta_{SIM}$  consists of whole parameters of  $G_m$  and  $E_{imm}$ . Note that we also tried adding  $\theta_{SIM}$  into the discriminator part objective described in Equation (4.4), but the model suffered from training instability.

## 4.3. Experimental settings with Artificial MNIST dataset

We prepared 3,690,000 triplets in accordance with the data preparation as we described in Section 3.2. We divided them into training: 90% and validation:



10%. For the test, we used another 100 samples on each  $0 \sim 9$  class from the original MNIST, we obtained 1,000 samples for testing. We used Chainer<sup>1</sup>[60] for the implementation. We used the following conditions: images are resize to 64x64, latent-space dimension = 128, optimization = *Adam* [29] (initialized by  $\alpha = 2.0 \times 10^{-4}, \beta = 0.5$ ), and training-time epochs = 20.

## 4.4. Results with Artificial MNIST dataset

### 4.4.1 Generated examples

Figure 4.3 shows examples generated with the IMI model. Source images and instructions (first and second columns from left) were given to the model. The generated images, the target (gold) images, and the SSIMs are shown in the third to fifth columns from left. From these examples, the IMI model generated similar images to the target images, especially regarding positions and sizes. However, digit shapes were distorted as SSIM is low.

### 4.4.2 Qualitative analysis of instruction vectors

Figure 4.4 visualizes the cosine similarities of instruction vectors with the IMI model’s instruction encoder  $E_i$ . Since the norm of each vector is normalized to 1, the difference between the vectors is determined only by the angle between the vectors, so we used the cosine similarity as a measure of similarity between vectors. To produce the left figure, first we prepared 240 instructions that were automatically generated from the template “[action] [digit] to the [direction].” with all combination of [action] from {“expand”, “compress”, “move”}, [digit] from {“zero”, “one”, “two”, “three”, “four”, “five”, “six”, “seven”, “eight”, “nine”}, and [direction] from {“top”, “left”, “right”, “bottom”, “top left”, “top right”, “bottom left”, “bottom right”}. After that, we sorted them in action-direction-digit order as (“move”, “top”, “zero”), (“move”, “top”, “one”), ..., (“move”, “bottom right”, “nine”), (“expand”, “top”, “zero”), ..., (“compress”, “top”, “zero”), ..., (“compress”, “bottom right”, “nine”). Then, we extract instruction vectors from them through the IMI model’s instruction encoder  $E_i$ . Simi-

---

<sup>1</sup><http://chainer.org/>











































source	instruction	generated	target	SSIM
	remove zero			0.980
	put one on the right .			0.948
	put four on the bottom left .			0.908
	move seven to the bottom .			0.908
	move six to the bottom .			0.883
	move seven to the bottom right .			0.854
	compress seven to the left .			0.826
	expand five to the bottom .			0.797
	expand six to the bottom .			0.747
	move five to the top .			0.747
	move seven to the top .			0.680
	expand one to the top right .			0.679
	expand six to the top .			0.613
	expand zero to the bottom .			0.245

Figure 4.3. Generated examples of IMI model using handwritten digit manipulation dataset (Artificial MNIST).

larly, to produce the right figure, we prepared 90 instructions from the template “put [digit] on the [position],” and 10 instructions from the template “remove [digit]” to compare the instructions about “put” and “remove.” with all combination of [digit] from {“zero”, “one”, “two”, “three”, “four”, “five”, “six”, “seven”, “eight”, “nine”}, and [position] from {“top”, “left”, “right”, “bottom”, “top left”,

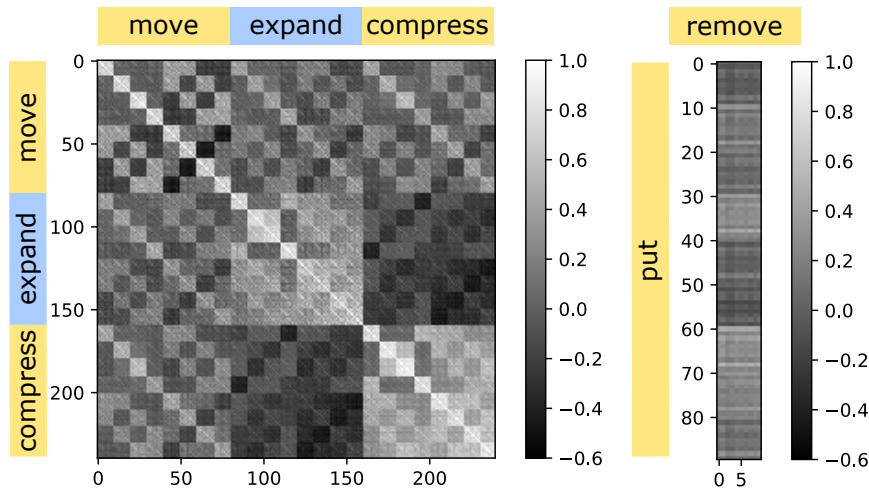


Figure 4.4. Cosine similarities between the instruction vectors from the instructions that contain {“expand”, “compress”, “move”} (left) and {“put”, “remove”} (right)

“top right”, “bottom left”, “bottom right”, “middle”}. After that, we sorted them in put-position-digit as (“put”, “top”, “zero”), (“put”, “top”, “one”), ..., (“put”, “middle”, “nine”) and (“remove”, “zero”), ..., (“remove”, “nine”), respectively. Then we extract instruction vectors from them through the IMI model’s instruction encoder  $E_i$ . Note that all instructions were the same as the instruction used in training.

Figure 4.4 shows that the map is clearly separated into a specific size of blocks. The large block “expand”-“compress” in the left figure indicates that the model interpreted “expand” and “compress” as an inverted concept because this area had a negative correlation. Furthermore, in the block of “move”-“move” (the right figure), the enlarged part of the red square of the left figure, is also clearly separated by small blocks and the cosine similarities follow the direction similarity as well. These results indicate that instruction vectors learned the concept of verb and direction. However, the concept of number is not significant in the instruction vectors. We guess that it is because we used just one digit operation in the experiment. We also tried the visualization of “put” and “remove,” but the clear blocks did not appear as shown in Figure 4.4 (right). We guess that

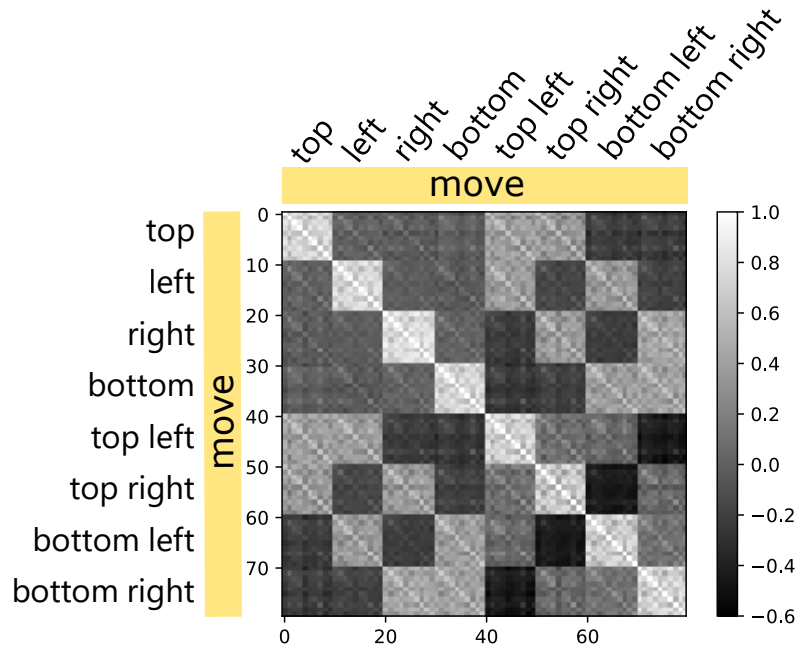


Figure 4.5. Enlarged cosine similarities of “move”-“move” block in Figure 4.4

this is because the concept of position or number is learned independently.

Figure 4.5 indicates the enlarged cosine similarities of “move”-“move” block in Figure 4.4. It seems that it is composed of  $8 \times 8$  blocks of the similarities. Each block consists of vectors extracted from the same [direction] instructions such as ‘move zero to the top.’ Each block’s cosine similarities have similar values; therefore, [digit] information does not affect the generation results compared with [action] and [direction].

Moreover, as shown in Figure 4.6, we also visualized the intra-block variances by explicitly grouping the same [action], [direction], or [digit] instruction vectors as a block, respectively. In other words, the horizontal axis represents the one remaining element when two of the elements (action, direction, digit) are fixed, and the vertical axis represents the variance values of the similarity maps obtained from different instruction vectors with only this one remaining element. For example, if we fix action and direction and create a similarity map with the only digit as different instruction vectors, we can obtain a total of 24 variance values of  $10 \times 10$  similarity maps for each pair of (action, direction), since there are 3,

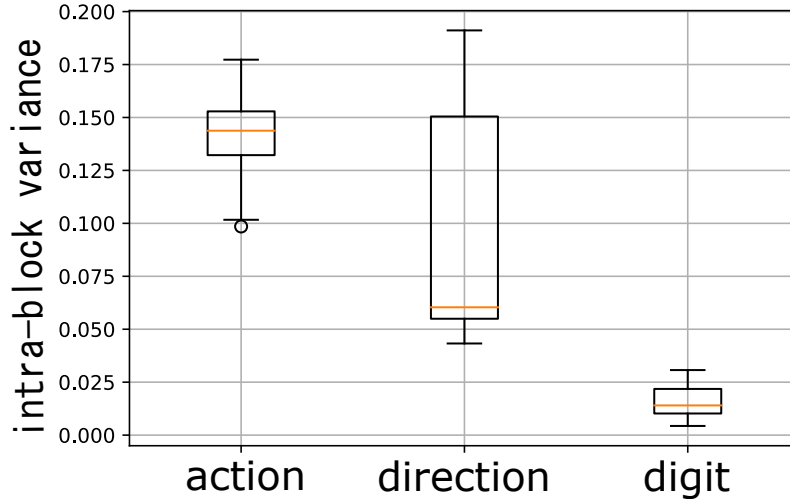


Figure 4.6. Distribution of variance of similarities of each group of instruction vectors

8, and 10 types of action, direction, and digit, respectively. The smaller variance element indicates that it less contribute to the instruction vector change. Since the variance values obtained for the [digit] in Figure 4.6 are small and distributed, the contribution of [digit] is smaller in the instruction vector change.

We conducted a subjective evaluation of three subjects. Each subject evaluated similarities of generated images and target (gold) images with 5 degrees (5=very similar, 1=very different). The subjective evaluation is composed of 100 example following Figure 4.3 format without SSIM. The rate of the score was {1: 9.00%, 2: 10.7%, 3: 18.3%, 4: 16.7%, 5: 45.3%}.

## 4.5. Experimental settings with Avatar Image Manipulation with Instruction dataset

Avatar Image Manipulation with Instruction (AIMI) dataset has a size of 4,296 samples for training; however, it does not have enough samples to train image generator. To stabilize the generator training, we used 161,065 images without

source target	instruction							
 	"put a black beard that ends before the ears"							
phase	1	500	700	1,000	2,000	4,000	4,350	5,000
w/o SIM (baseline)								
w/ SIM (proposed)								

Figure 4.7. Generated results on each phase

instruction. During training, we repeated the training without instructions (w/o instruction) and with instructions (w/ instruction) phase of 2,200 examples alternatively. We defined one phase as every 2,200 examples processed.

In w/o instruction training phase, we utilize an image as a source and a target image. We train a model as an auto-encoder to generate the same image to a given source image. The instruction vector was set by zero vector. The aim of the w/o instruction training is supporting the *image-generator* learning to generate clear target images. Note that the masking layer is not trained in this phase because there are no differences between the source and target images.

In w/ instruction training phase, we used the full triplets of (source image, target image, instruction). We found that our editing model with SIM struggled to learn to generate mask, but the performance was not good enough. It would be because of the lack of data size of the full triplets. To encourage the model to learn to generate mask, we used ground truth masks in training. A ground truth mask is automatically created by comparing a pair of source and the target image. In each position of pixels, the mask’s pixel value is set by zero if the target pixel value is different from the source pixel, otherwise set by one. We added an objective for the mask generation with mean squared error in the training. This objective is added to Equation (4.17).

We trained the models using *Adam* [29] ( $\alpha = 2.0 \times 10^{-4}, \beta = 0.5$ ) until 5000 phases. Images are resized to  $64 \times 64$ . Hidden size is 128 for  $\phi^i$  and  $\phi^{fc}$ ; 1024 for  $\phi^{im}$ ;  $512 \times 4 \times 4$  for  $\phi^{imm}$ . Batch size is 64. Vocabulary size is 1892.

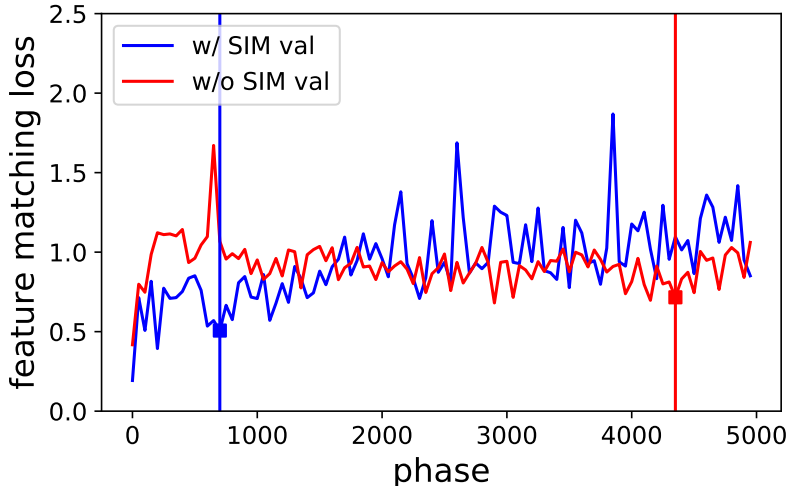


Figure 4.8. Model selection using feature matching loss

## 4.6. Results with Avatar Image Manipulation with Instruction dataset

Figure 4.7 indicates generated examples on each phase for an example in validation set. While w/o SIM took over 4,000 phases to generate a similar image to the target image, w/ SIM generated a more similar image under 1,000 phases. This indicates that introducing SIM makes the training more stable and faster. To select models for the evaluation, we looked at the loss curve line of feature matching in training shown in Figure 4.8. Although the training GANs is not stable, we found that feature-matching loss is relatively useful to select the better model. Judging from the generation results on each phase and the whole validation loss, we selected the model of phase 4,350 for the baseline and phase 700 for the proposed model, respectively.

We used the test set (230 examples) for the quantitative evaluations. As the objective evaluation, we compared MSE scores between generated images and target images by using the baseline and the proposed model. The results were  $4.92 \times 10^{-2}$  for the w/o SIM (baseline) and  $3.31 \times 10^{-2}$  for the w/ SIM (proposed). It indicates that the generated examples by using the proposed model are better

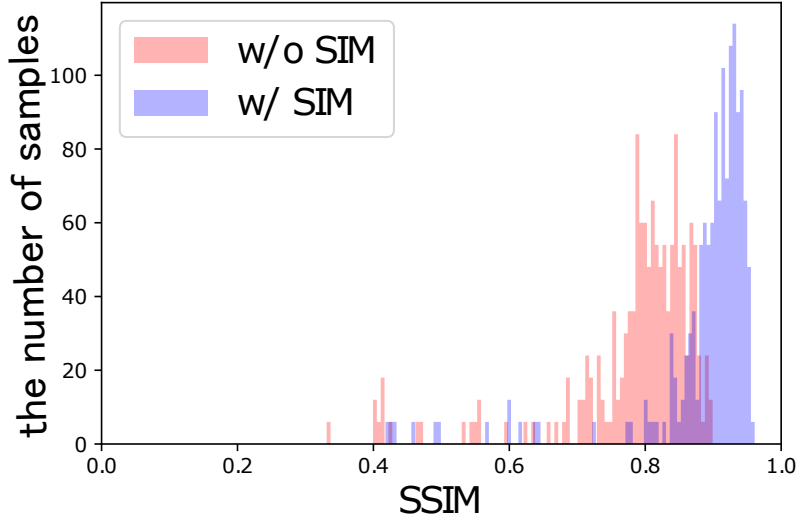


Figure 4.9. Histogram of SSIM between the generated and target images using w/o and w/ SIM models.

than that of baseline. We also compared SSIM scores between generated images and target images by using the w/o SIM model and the w/ SIM model. Figure 4.9 represents the histogram of SSIM between the generated and target images using w/o and w/ SIM models. Higher SSIM distribution is better. Figure 4.9 shows w/ SIM model performed over 0.8 on the most of examples and outperform the SSIM score of w/o SIM model.

As the subjective evaluation, we showed a source image, an instruction, and generated images by using the baseline and the proposed model to human evaluators. We used crowd sourcing<sup>2</sup> for the evaluation. Each crowd worker provided a preference for randomly-ordered each pair of generated images (A, B) in five-grade (1: A is much better than B, 2: A is better than B, 3: the results are comparable, 4: B is better than A, 5: B is much better than the A). We considered reversed order cases, and prepare 460 examples to be evaluated by human evaluators in total. Three workers evaluated on each example and each worker evaluated up to 10 examples. Finally, we obtained 1,380 evaluated results. Figure 4.10 visualizes the proportion of each grade after restoring reversed-ordered

<sup>2</sup>Crowdfower: <https://www.crowdfower.com/>



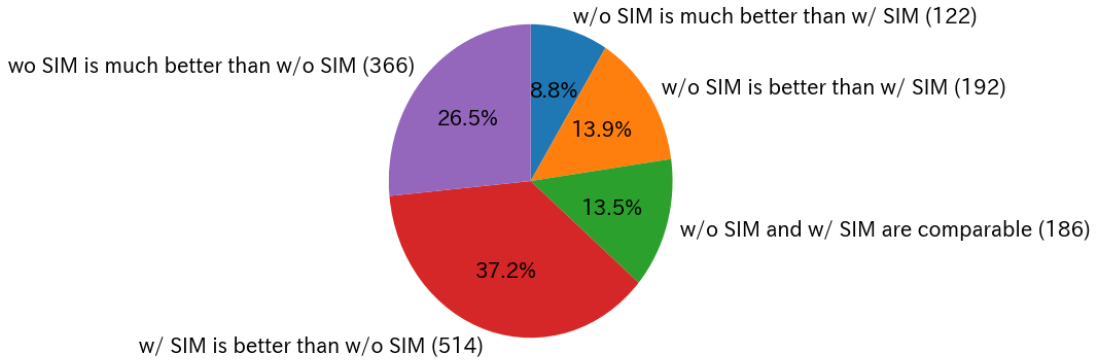


Figure 4.10. Subjective evaluation of generated images between w/o and w/ SIM models. The attached numbers to the labels denote the actual number of vote by the evaluators. The evaluators saw image pairs (A, B). One is generated from w/o model, and the other is from w/ SIM model. We asked the evaluators to select five-grade: (1: A is much better than B, 2: A is better than B, 3: the results are comparable, 4: B is better than A, 5: B is much better than the A). We de-anonymized whether images are from w/o or w/ SIM model, and visualized the number of the votes.

examples. The workers who preferred the w/o SIM model selected “w/o SIM is much better than w/ SIM” (blue) or “w/o SIM is better than w/ SIM” (orange). The workers who preferred the w/ SIM model selected or “w/ SIM is better than w/o SIM” (red) or “w/ SIM is much better than w/o SIM” (violet). The total proportion of subjects who preferred the generated images of the proposed w/ SIM model (red and violet) is over 60%. It indicates that the w/ SIM model generated better images than the w/o SIM model.

We compared the generated examples in the test set for qualitative evaluation. Figure 4.11 shows the generated examples using the baseline and the proposed model. The texts above each image line show the given instructions. From the left of each images line, the source image, the target image, generation results of the baseline method, generation results of the proposed method and the instruction are listed. The generation results of the baseline method have two images: the generated image and the visualization of changing points by the pixel-wise squared error between the source and the generated image. The generation results of the

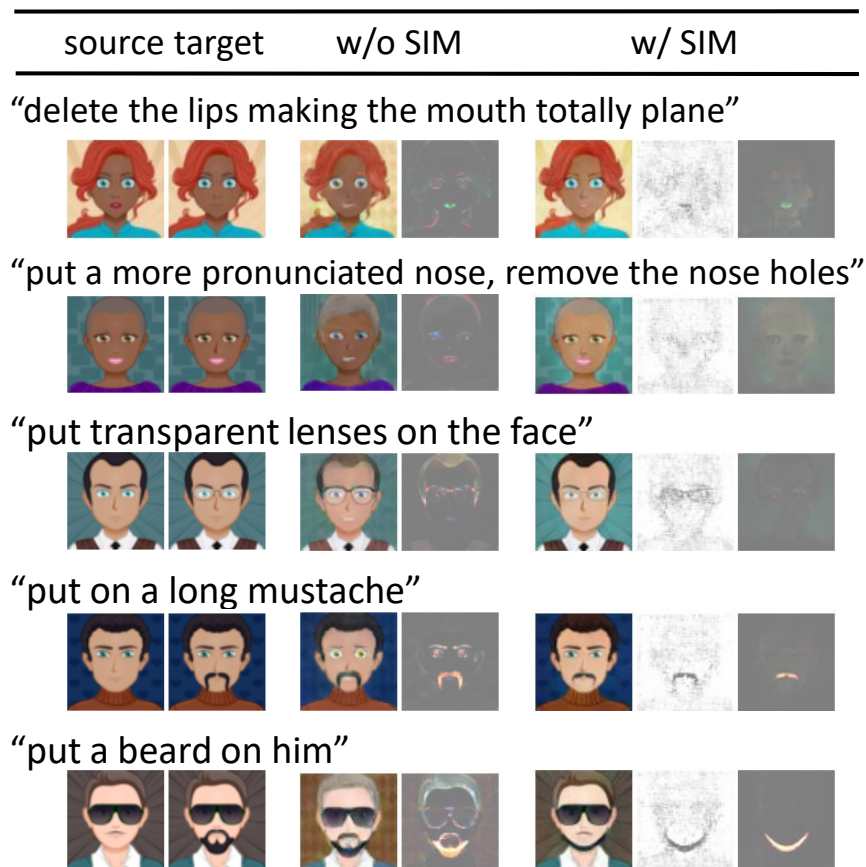


Figure 4.11. Generated examples using w/o SIM model and w/ SIM model

proposed method have three images: the generated image, the visualization of the mask and the visualization of changing points as well.

Both models successfully generated changed images according to the instructions; however, the baseline model suffers from the co-occurrence of undesired changes, i.e., the texture or colour of the background, shape of hair, eyes or mouth. On the other hand, the proposed model successfully kept the details of the source images, which is not mentioned in instructions. We found that the proposed model can preserve details of source images; therefore, the model works better for small changing, i.e., changing nose, mouth, ears, which are difficult for the baseline model.

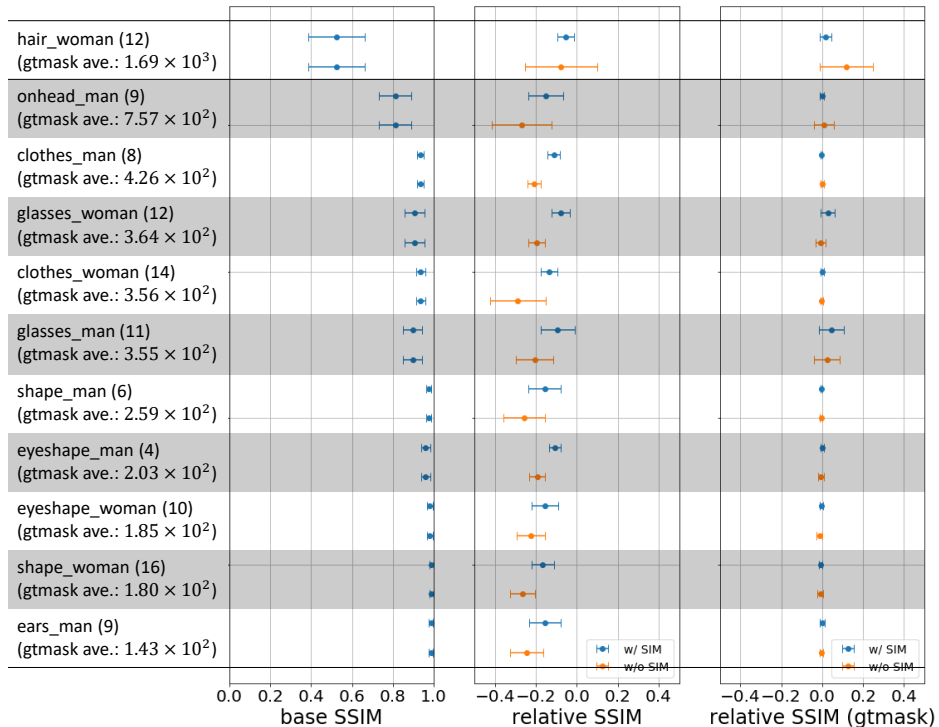


Figure 4.12. SSIM comparison between w/ and w/o SIM models on the first 11 instruction types. The plots are ordered in ground truth masks (gtmask), which indicate the number of different pixels between source and target images. There are three SSIMs: original source–target SSIM (base SSIM); generated–target SSIM subtracted by the base SSIM (relative SSIM); relative SSIM but outside of edit region covered with gtmask is ignored (relative SSIM (gtmask)).

#### 4.6.1 Analysis on the instruction types

We conducted a detailed analysis of the generated images with w/o and w/ SIM models on 22 instruction types, e.g., editing woman hair, man beard. These instruction types were automatically annotated when we collected the image pairs. In the data collection described in Section 3.2.2, the selected attributes correspond to the instruction types. We visualized two SSIM distributions with w/o and w/ SIM models on each instruction type.

Figure 4.12 and Figure 4.13 visualize SSIM distributions on the first 11 and the last 11 instruction types in the test set, respectively. The left column names

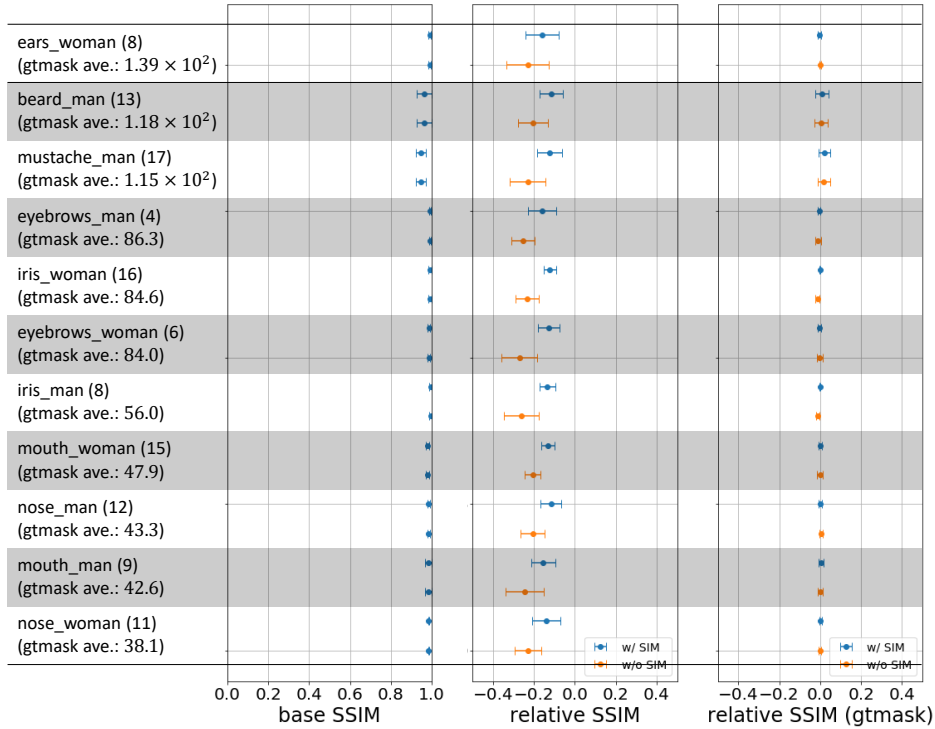


Figure 4.13. SSIM distributions on the last 11 of 22 instruction types. The settings are same as Figure 4.12 but the instruction types are the last 11 types.

such as “hair\_woman” represent the instruction types. Each white and grey row block represents the same type of instruction. The blocks are ordered in large average of ground truth masks (gtmask), which indicate the number of different pixels between source and target image pairs. The numbers that are attached to the tail of the instruction type names indicate the number of samples in the test set. On each block, there are three figures with two SSIM distributions by w/ SIM model and w/o SIM model. From left to right, base SSIM represents the SSIMs between the original pairs of source and target image; relative SSIM represents the SSIMs between the pairs of generated image and target image that is subtracted by the base SSIM; relative SSIM (gtmask) represents the relative SSIM that replaced outside the region of gtmask with the source image. Note, the two SSIM distributions on each block in base SSIM are same for easy comparison. Furthermore, gtmask is not available in a real scenario. It is just for analyzing the performance of image editing.

Higher relative SSIM (gtmask) represents the improvement of the region to edit. It ignores the other region although relative SSIM considered the whole region. Judging from Figure 4.12 and Figure 4.13, each instruction type has similar relative SSIM (gtmask) between the result with w/ SIM model and w/o SIM models but the relative SSIMs of w/o SIM model become much worse than that of w/ SIM model. It supports the result that SIM contributes to mitigating the unintentional change of the outside of the edit region. Instruction types with a large region to edit (large gtmask average) such as “hair\_woman” tends to be difficult for w/ SIM model, but w/o SIM has a higher possibility to improve the source images than w/ SIM model.

Judging from the SSIM in the relative SSIM (gtmask), our editing models successfully improved the SSIM with a space to improve the base SSIM such as “hair\_woman” and “glasses\_woman” but tend to fail to edit the instruction type with extremely high base SSIM such as “shape\_man” and “eyeshape\_man.”

## 4.6.2 Comparison with the larger image size with the SIM model

We conducted an additional analysis on w/ SIM model to investigate the effect of image size to its performance. We compared image size of  $128 \times 128$  with  $64 \times 64$ . Note that above experiments were conducted with  $64 \times 64$ . For the experiment with image size  $128 \times 128$ , we conducted the experiment under the same settings described in Section 4.5 and selected the model with 1700 phase Figure 4.14 and Figure 4.15 visualize SSIM distributions image size of 64 ( $64 \times 64$ ; blue) and 128 ( $128 \times 128$ ; orange) on the first 11 and the last 11 instruction types in the test set, respectively. The settings are almost same as Figure 4.12 and Figure 4.13 but the comparison is conducted between different image size of  $64 \times 64$  and  $128 \times 128$ . Judging from relative SSIM (gtmask) of them, the twice resolution size did not affect the performance improvement with edit region. On the other hand, the model with larger image size worsened relative SSIM.

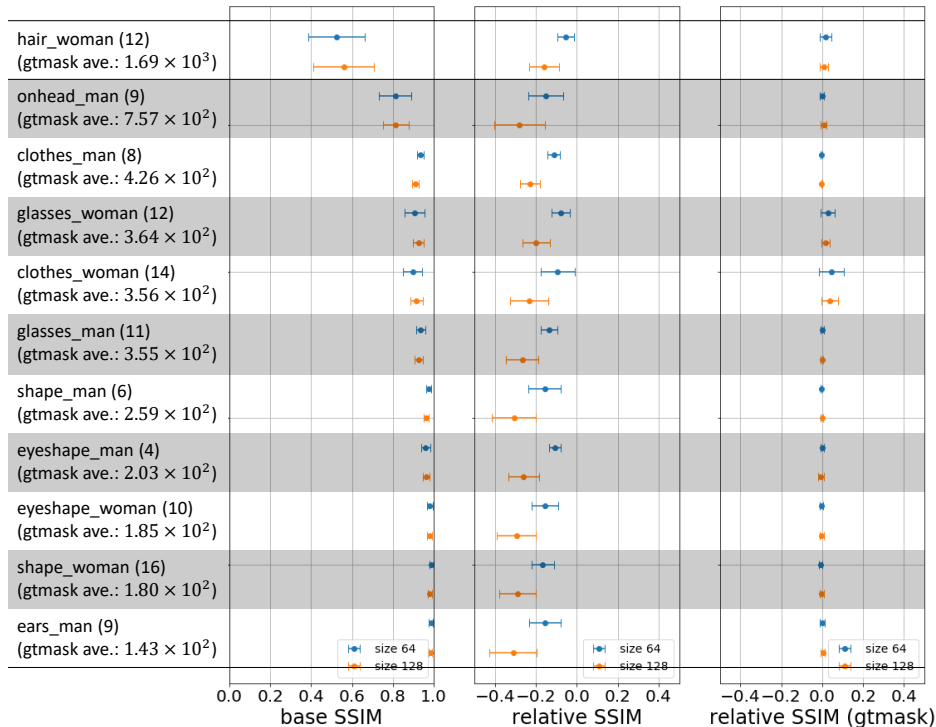


Figure 4.14. SSIM comparison between image size of 64 and 128 on the first 11 instruction types. The plots are ordered in ground truth masks (gtmask), which indicate the number of different pixels between source and target images. There are three SSIMs: original source–target SSIM (base SSIM); generated–target SSIM subtracted by the base SSIM (relative SSIM); relative SSIM but outside of edit region covered with gtmask is ignored (relative SSIM (gtmask)).

## 4.7. Discussion

This chapter introduced an image editing framework using natural language instruction. We conducted the two image editing tasks: a handwritten digit editing task with Artificial MNIST dataset and an avatar image editing task with AIMI dataset.

The results of the handwritten digit editing task indicate that our editing model can edit a source image according to the instructions for digit deletion, adding, and movement. The edited images tend to be accurately changed in discretized spatial size and position. However, a problem exists. The generated

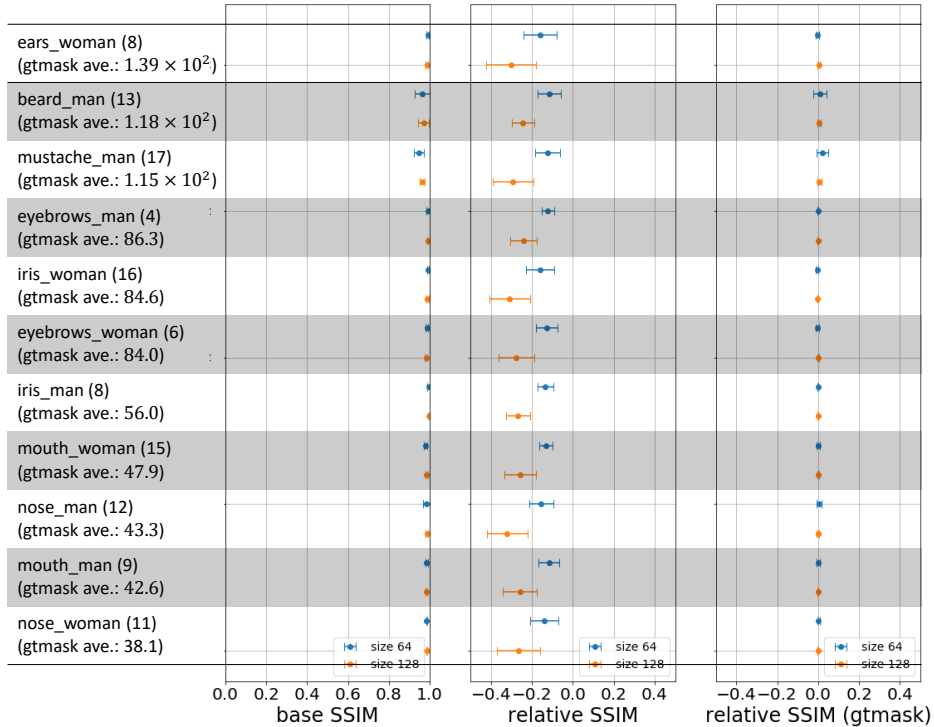


Figure 4.15. SSIM comparison between image size of 64 and 128 on the last 11 of 22 instruction types. The settings are same as Figure 4.14 but the instruction types are the last 11 types.

digit detail tends to be collapsed as a source digit image size is large. Furthermore, the training requires an auxiliary classifier with a digit, size, and position labels. It implies supportive spatial information would be essential for the editing, which requires spatial movement.

On the other hand, the avatar image editing task with AIMI dataset has no positional movement of parts of images but requires our editing model to handle various instructions collected from human annotators. Although our editing model without source image masking (w/o SIM) suffered from undesired changes in its generated image, our editing model with source image masking (w/ SIM) successfully suppressed this undesired changes. However, editing large part of images such as hair editing is difficult for w/ SIM model. The SIM works as a constraint to preserve a source image. Using w/o SIM model has a possibility to be preferable in this case.

Our editing models on avatar image editing tend to fail to edit tiny parts such as eyebrows and mouth. Enlarged image size of  $128 \times 128$  does not improve the performance well. It seems because our editing model generator considers a whole image region when it generates a new image, and the edit region is still not prominent compared to the whole region even in the larger image size. A possible solution is editing a region after cropping [36]; however, it requires pre-defined and fixed masks to crop an edit region. It does not cost to create edit region for AIMI dataset because it consists of the parts with a fixed position. However, more complicated image editing tasks that it is challenging to define an editing region are future challenging tasks, for example, human hand-drawing illustrations or art editing.

Furthermore, we would like to discuss other limitations of our w/ and w/o SIM model toward other editing tasks. The first limitation exists in data collection. Our editing models require slightly different pairs of the source and target image, which can be expressed as a simple instruction. Slightly different pair collection is difficult on some editing tasks. For example, a realistic human face collection for our image editing task is difficult because any two different people have many differences on their faces. In this situation, our editing model with source image masking cannot handle this problem because any instructions will create a mask that indicates the whole face as an editing region. A promising task for our editing model is a creating illustration. If we record artists work continuously, each moment can be a feasible dataset for our editing model.

The second is the limitation of feasible instructions. In real situations, the desired edit depends on users. For example, in face editing, a request to change the colour may or may not imply the other hair colour such as eyebrows and moustache should also be changed. The correct intention is variable on task settings, for example, use case of editing, user’s cultural background, and social trend. As the number of users becomes large, it will be hard to handle a minor intention because a general machine learning model learns major relationships in the given dataset. Toward a useful tool, our editing models need to consider this proper use for individual users through user modelling. Automatic adjustment to an individual user via dialogue is a useful function but a challenging topic for our future direction.





# Chapter 5

## An entropy-based confirmation for image editing dialogue

This chapter introduces an interactive image editing system with an entropy-based confirmation strategy. The system works on the interactive image editing task defined in Section 3.1. This system has two image editing models, DCGAN-based w/o SIM model and w/ SIM model introduced in Chapter 4, that accepts natural language instruction (editing request). Asking the user to select a relevant image from the candidates generated from the multiple models enables the system to handle various editing requests; however, every time confirmation causes redundant dialogues. To achieve more efficient dialogues, we demonstrate that our proposed confirmation strategy enables the system to reduce redundant confirmations in the interactive editing task between user evaluators and the system.

### 5.1. Introduction

Image editing systems that accept natural language requests often face ambiguities caused by natural language. Unlike general image-to-image translation tasks [26], such editing systems must fill in the gap between ambiguous natural language requests and possible generation with additional conditions. For example, the following natural language request, “make this avatar’s hair short,” lacks a specific objective image or criterion for creating the image desired by the user.

It could be “make this avatar’s hair short by her ears” in a less ambiguous case. However, such lack often occurs in real situations. This is one challenging obstacle that must be solved to generate images with a given text. Asking the user about the ambiguity is one way to solve the problem. This solution is one of our motivations to introduce an interactive process in image editing. A trade-off also exists between the generated image quality and the constraints on the image generation system. For example, a masking mechanism is an efficient way to improve the quality of generated images in image-to-image translation tasks [36, 44, 46]. Even in image editing with natural language, such generation systems based on masking constraints generate more accurate images than a system without them because it can distinguish those parts mentioned in the user’s requests. However, such a strong constraint limits significant changes to the image. For example, in interactive image editing, it is difficult for systems with a strong constraint to work on such a request as “make the current portrait’s hair longer” because the requested will significantly change the image. In such cases, using a generation system without any constraints can create more relevant images to the user’s intention.

Considering a problematic case where the system cannot decide which generated image is better as an editing result for users, one possible solution is direct confirmation with them. However, asking users to choose a single image for every request is completely unreasonable. Thus, the system is expected to ask them when it is unsure, which is the best image to present.

Figure 5.1 shows the overview of our system in the interactive image editing task defined in Section 3.1. We assume that our interactive image editing system has two different types of image editing models introduced in Chapter 4: a model with a strong constraint based on the source image masking, “w/ mask”; and a model without a constraint, “w/o mask”. We tackle this problem to find a better dialogue strategy using these two models and introduce an uncertainty score based on the entropy of the generated masks to decide the best model to a given instruction. The system confirms with the user when it is tentative about selecting a better image to match the user’s editing intent using uncertainty scores. Section 5.2 introduces a general conditional image-to-image generation with mask generation and Section 5.3 introduces the method of this strategy

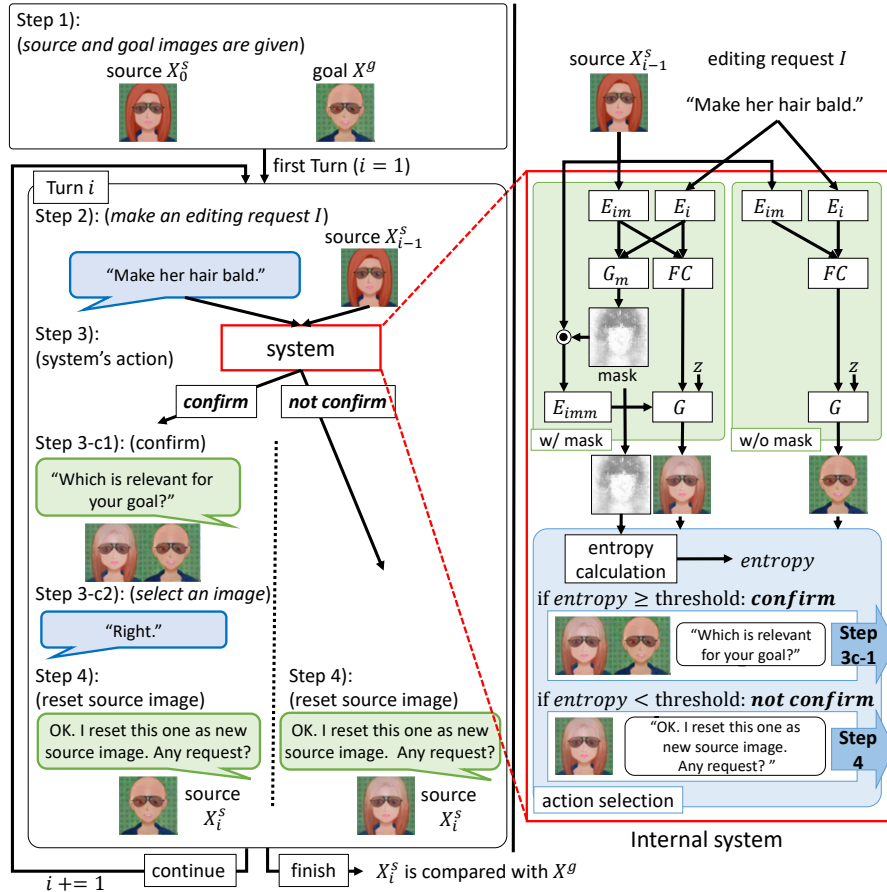


Figure 5.1. Overall interactive image editing flow described in Section 3.1 with the proposed confirmation strategy (blue box in the right) and DCGAN-based w/ and w/o mask (SIM) models described in Chapter 4 (green box in the right).

based on the generated mask.

## 5.2. Conditional image-to-image generation with a masking module

We assume that system’s image editing model has a mask generator, which works as a constraint to improve image-to-image generation [36, 44, 46], including our image editing task. General conditional image-to-image generation model with a

masking module generates a mask  $\mathbf{M}$  and it affects the final output  $\hat{\mathbf{X}}$  with the input pair of a source image  $\mathbf{X}^s$  and a condition  $\mathbf{c}$  as follows,

$$\mathbf{M} = f_m(\mathbf{X}^s, \mathbf{c}), \quad (5.1)$$

$$\hat{\mathbf{X}} = f_g(\mathbf{X}^s, \mathbf{c}, \mathbf{M}). \quad (5.2)$$

The input  $(\mathbf{X}^s, \mathbf{c})$  changes into the mask  $M$  through an mask generation function  $f_m$ . The final output is generated with the input  $(\mathbf{X}^s, \mathbf{c})$  and additional mask condition  $\mathbf{M}$ . Revisiting Equations from (4.1) to (4.3) and from (4.14) to (4.16), our image editing model with source image masking generates a mask with  $G_m$  with input with a source image  $\mathbf{X}^s$  and an instruction  $I$  as a condition  $\mathbf{c}$ .

Each pixel value of the generated mask has a range  $[0, 1]$ . The mask can be seen as element-wise attention; in other words, the mask represents the region to be changed or preserved. If the editing model cannot decide that a pixel should be changed or preserved, the mask value will be 0.5. Therefore, we use the entropy of the mask for the system’s confirmation.

### 5.3. System’s confirmation of action decisions based on mask entropy

The blue box in the right figure of Figure 5.1 represents the system’s confirmation action selection module. Confirmation that shows multiple editing results to users from multiple models is a safe action; however, the user must pay an additional cost for responding to the confirmation. When a confirming action must be selected, basing it on some uncertainty scores of image generation will smooth the dialogue. We used the entropy scores of the generated image as the uncertainty scores and calculated the entropy:

$$entropy = -\frac{1}{WH} \sum_i^W \sum_j^H \{m_{ij} \log(m_{ij}) + (1 - m_{ij}) \log(1 - m_{ij})\} \leq -\log 0.5. \quad (5.3)$$

We define  $m_{ij}$  as the value of the predicted mask at the (i,j)-th position with width  $W$  and height  $H$ . The system has a confirmation threshold  $-\alpha \log 0.5$  ( $0 \leq \alpha \leq 1$ ). It selects *confirm* if  $entropy \geq -\alpha \log 0.5$ , otherwise it selects *not confirm*. In

the experiment, we tried several  $\alpha$  using  $m_{mono}$  described in Equation (4.14) for the entropy calculation, which decides the system’s dialogue strategy in Section 4.2.2.

## 5.4. Experimental settings

We conducted experimental dialogues to investigate the effectiveness of our proposed dialogue strategy. In this section, we describe the dataset for the image editing dialogues, the training details of each model, and the user evaluation settings.

### 5.4.1 Dataset

For training w/ and w/o mask models and evaluation, we utilized the Avatar Image Manipulation with an Instruction dataset. The task is portrait image editing based on instructions, which indicate natural language editing requests. The data consist of 22 types of editing, e.g., changing a beard, eyebrows, and or hair. Each sample is composed of a triplet of {source image, target image, instruction}. We split the dataset into *train* : *validation* : *test* = 4,296 : 230 : 230 according to existing work. We also used 161,065 examples of independent images to improve the generator’s image modelling.

### 5.4.2 Training models

During the training, we alternatively repeated the training of the image generator and the image editing. In the image generator training phase, we trained the model as an auto-encoder to generate the same image to the given source image for stabilizing the generator. We also set the instruction vector to zero in this training phase. This process enhances the generator’s ability to generate clear images. In the image editing training phase, we utilized full triplets of {source image, target image, instruction}. The dataset consists of the editing requests that represent only one attribute change such as hair change; thus, we can prepare ground truth of the mask by comparing a pair of source and target images to improve the SIM’s mask generator  $G_m$  training. We used the ground truth mask

in training, whose pixels were set to zero where the pixels in the same position of the source and target images are different, or otherwise, they are set to one. We also provided a mask loss function as a mean squared error between the generated mask and the ground truth one to improve the SIM model. We trained the models using *Adam* [29] ( $\alpha = 2.0 \times 10^{-4}$ ,  $\beta = 0.5$ ) until 5000 phases. The images were resized to  $64 \times 64$ . The following are the hidden sizes: 128 for  $\phi^i$  and  $\phi$ , 1024 for  $\phi^{im}$ , and  $512 \times 4 \times 4$  for  $\phi^{imm}$ . The batch size is 64, and the vocabulary size is 1892.

### 5.4.3 User evaluation of image editing dialogue

In a pilot study, we found that the w/ mask model tends to successfully edit in a single turn a small region, such as changing eye colour or adding a moustache or glasses. However, the w/ mask model often fails to edit a large region of the source image, such as changing hairstyle. Therefore, we focused on hair editing to evaluate the image editing dialogue. We evaluated our proposed confirmation strategy in two aspects. First, we evaluated the necessity of confirmation by comparing between the strategy without confirmation using the w/ mask model and strategy with confirmation using both the w/o and w/ mask models. Second, we evaluated the effectiveness of the confirmation strategy by comparing the strategy without confirmation or a random strategy with the others. We used 21 patterns (9 for male portraits and 12 for female portraits) as pairs of source and goal images and conducted image editing dialogue experiments with human evaluators. The evaluators were 18 people whose TOEIC scores exceeded 730 and could use English for daily use. At the task’s beginning, the evaluators looked at the source and goal images and talked with our interactive image editing system, which has different dialogue strategies. Each pattern was evaluated by three evaluators over the following six strategies: the system selected *confirm* with thresholds  $\alpha = 0.0, 0.25, 0.50, 0.75, \text{ and } 1.0$ , (Section 5.3) and randomly selected *confirm*. We compared these different strategies to identify the effectiveness of our proposed method on the problem of interactive image editing. Note that  $\alpha$  represents proactiveness for confirmation: when  $\alpha = 0.0$ , the system selects *confirm* every time; and  $\alpha = 1.0$ , it selects *not confirm* every time. In other words,  $\alpha = 1.0$  corresponds to the case where the system uses the w/ mask model

every time. Therefore, we can evaluate the necessity of confirmation by comparing  $\alpha = 1.0$  with the others to evaluate the effectiveness of the confirmation strategy by comparing  $\alpha = 1.0$  or a random strategy with the others, as described below.

#### 5.4.4 Necessity of confirmation (limitation of a single model)

Confirmation is useful when the system needs to deal with multiple editing results from multiple models. It is difficult for a single editing model to accept every editing request because a trade-off exists between editing flexibility and model constraints. We first investigated how the single w/ mask model works on an interactive image editing task. We compared models with different confirmation strategy settings for the improvement of image quality through dialogues (higher is better).

#### 5.4.5 Effectiveness of confirmation strategy

Second, we investigated the effectiveness of our proposed confirmation strategy. If our confirmation method works with appropriate timing, it will improve performance (higher image quality with shorter dialogue length).

### 5.5. Results

Next we describe and discuss our experimental results in the two aspects described in Sections 5.4.4 and 5.4.5. To address the question in Sections 5.4.4, we visualize whole plot of user–system dialogue samples, and we show that the system strategy with confirmation can improve the SSIM between source and target image through dialogue although the strategy without confirmation cannot improve the SSIM. To address the question in Section 5.4.5, We conducted additional analysis on two metrics  $SSIM_{improvement}/\#user\ turn$  and  $\#user\ turn$  to discuss the difference of strategies because it is hard to discuss the difference between the confirmation strategies with the average and standard deviation of the  $SSIM_{improvement}$  of the plots of dialogues.



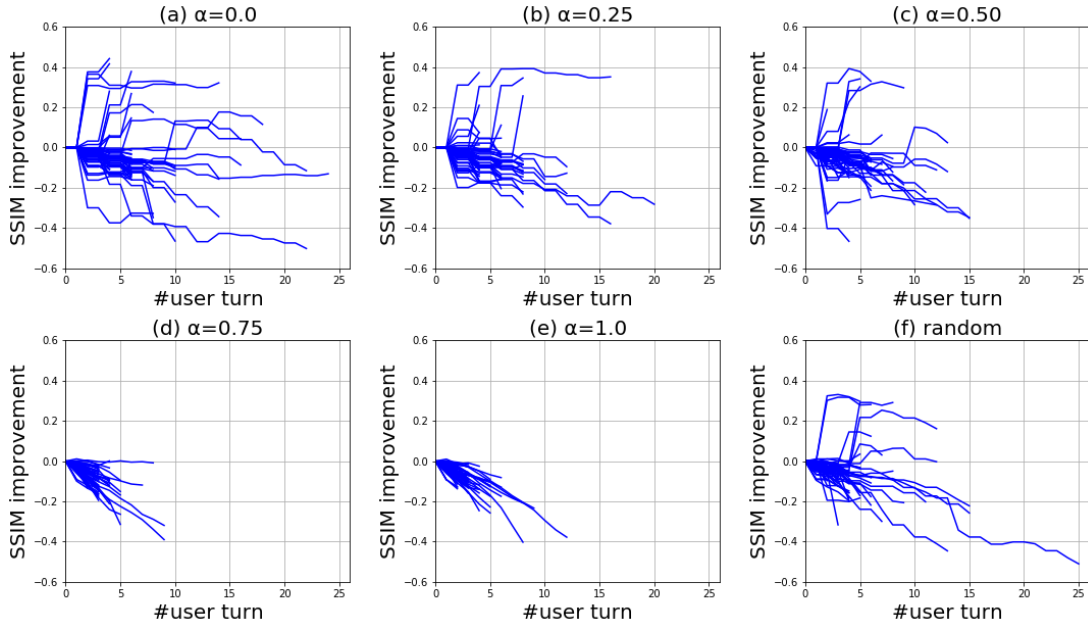


Figure 5.2. Experimental results of image editing dialogue between 18 evaluators (users) and the system: *#user turn* denotes total number of user actions (*making editing request* and *selecting an image*); (smaller is better). *SSIMimprovement* denotes the relative SSIM based on the first SSIM (higher is better). Each plot on each figure represents the trail of *SSIMimprovement* in one dialogue task between the system and an evaluator.  $\alpha$  indicates threshold for system to select confirmation: (a)  $\alpha = 0.0$ , (b)  $\alpha = 0.25$ , (c)  $\alpha = 0.50$ , (d)  $\alpha = 0.75$ , (e)  $\alpha = 1.0$ , and (f) random: system randomly selects confirmation. If  $\alpha$  becomes small, the system tends to select confirmation with lower uncertainty score. Note that every *SSIMimprovement* is calculated after user’s action. Therefore, when the system selects confirmation after the user makes an editing request, *SSIMimprovement* keeps identical value. Degradation as dialogues progress is caused by image editing models.

### 5.5.1 Necessity of confirmation (limitation of a single model)

Figure 5.2 indicates the relative changes of SSIM from the current image to the goal image and plots the all dialogues on each setting as the dialogue progressed. Figure 5.3 visualizes average and standard deviation of the plots in

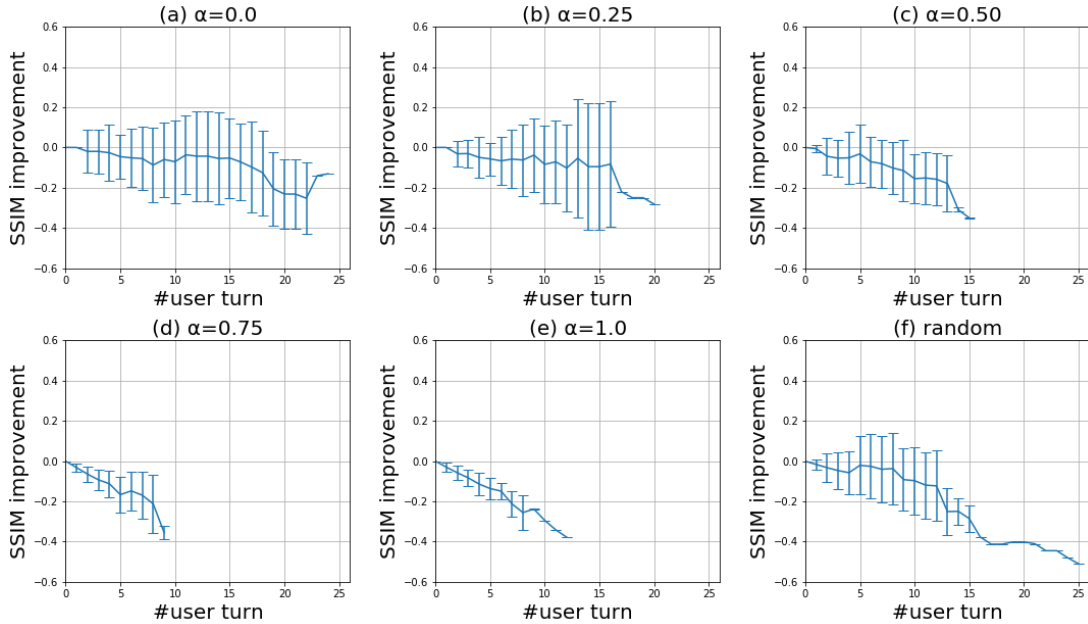


Figure 5.3. Visualization of average and standard deviation on each figure of Figure 5.2.

Figure 5.2. These figures aim to show the necessity of confirmation. *#user turn* denotes the total number of the user actions of *give query* and *select image* (smaller is better). *SSIMimprovement* denotes relative SSIM, which is subtracted from the first source–goal SSIM. A  $i$ -th turn *SSIMimprovement* is defined as  $SSIMimprovement = SSIM(X_i^s, X^g) - SSIM(X_0^s, X^g)$  (higher is better).

*SSIMimprovement* with higher  $\alpha$  ( $\alpha = 0.75, 1.0$ ) only worsened as the dialogue progressed. It is because the system with higher  $\alpha$  did not confirm with the evaluators and only provided the generated image from w/ SIM model.

On the other hand, *SSIMimprovement* with lower  $\alpha$  ( $\alpha = 0.0, 0.25, 0.50$ ) indicates that some dialogue examples achieved  $SSIMimprovement > 0$  that indicates a better SSIM than before the dialogue. This result indicates that the w/o mask model is necessary to get better SSIM scores to change a broader region, such as a woman’s hair. Thus, the confirmation is necessary to improve SSIM through dialogues.

Note that the degradation on each turn was caused by the image editing mod-

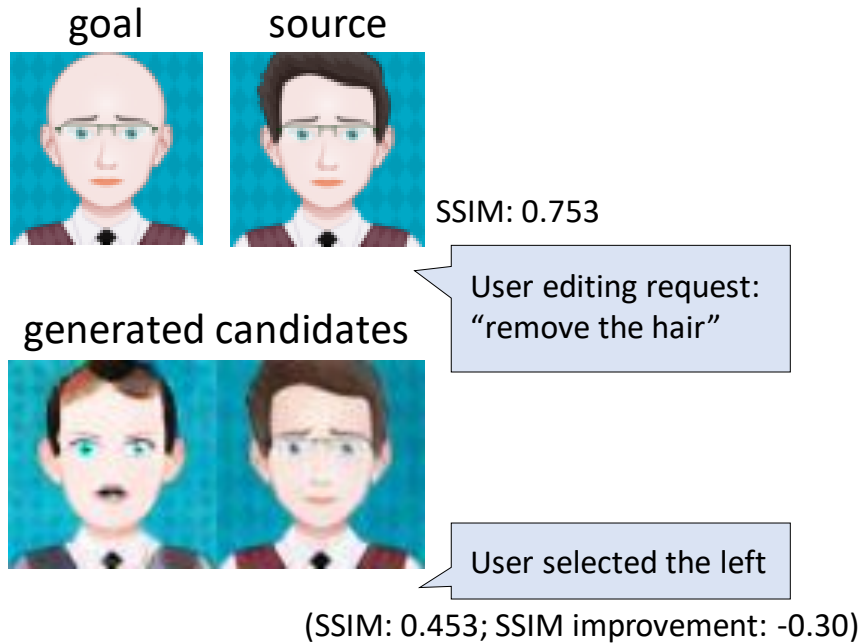


Figure 5.4. A case with low SSIM improvement.

els, which were trained with single turn editing triplets of {source image, target image, editing request}. In other words, the models were inadequately generalized to the degraded source images. The degradation was gradually strengthened as the dialogue progressed. Sudden degradation of SSIM with  $\alpha = 0.0$  and  $\alpha = 0.50$  was caused by the user preference. Figure 5.4 shows the actual case of  $\alpha = 0.0$  with SSIM improvement of  $-0.3$  at  $\#userturn = 2$ . The user added an editing request "remove the hair", and the system showed two generated candidates from w/o SIM and w/ SIM model. The user selected w/o SIM model's left image, which was partially successful in removing the hair but the other region was also unintentionally changed. It caused the lower SSIM. Other settings with lower  $\alpha$  or random can also cause this case. On the other hand, higher  $\alpha$  does not cause this case because it does not confirm and only provide the generated image from w/ SIM model.

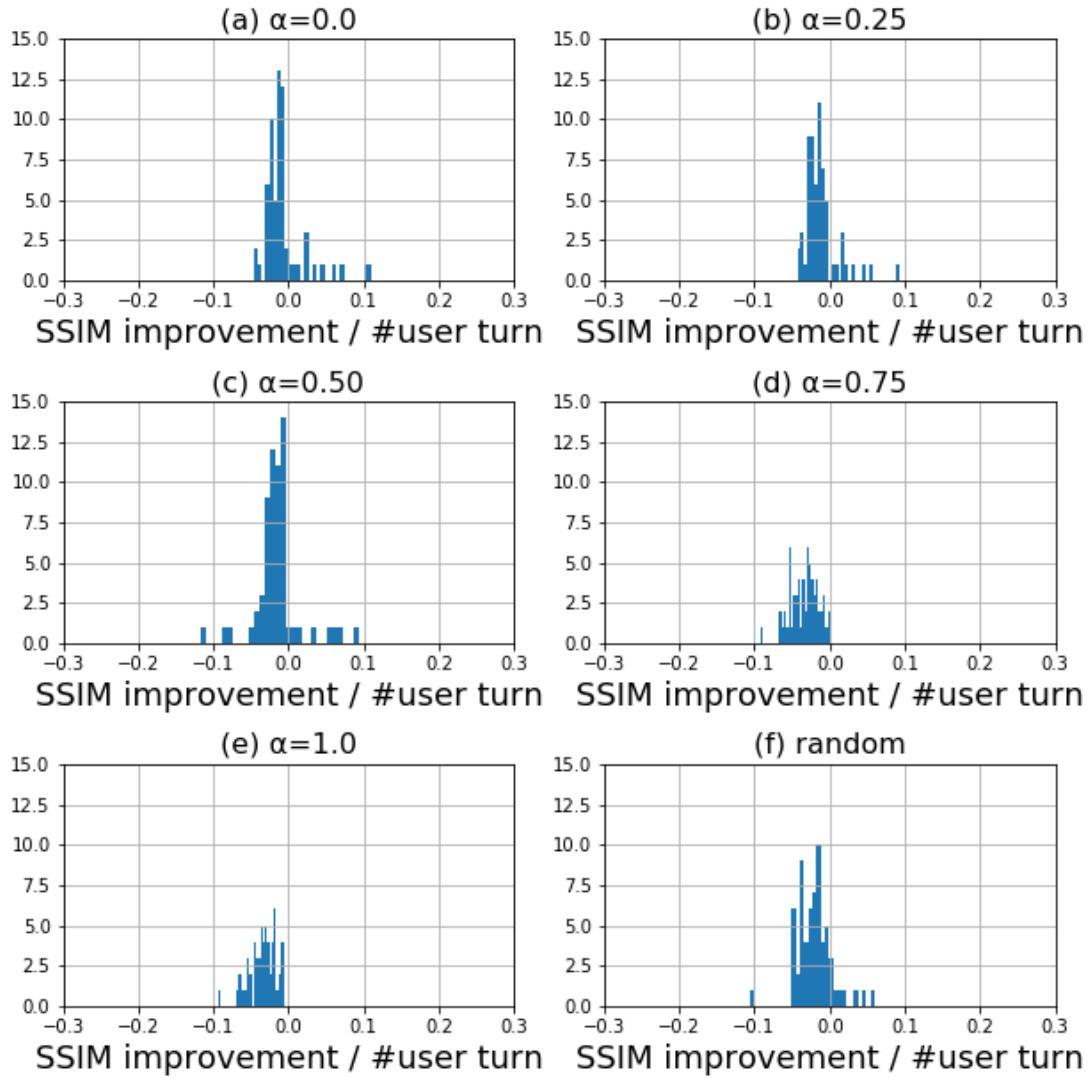


Figure 5.5. Histogram of  $SSIM_{improvement}/\#user\ turn$  at end of dialogues on each strategy: higher  $SSIM_{improvement}/\#user\ turn$  dialogue created more similar images to goal and more efficient dialogues.

### 5.5.2 Effectiveness of confirmation strategy

Even in Figure 5.3, it is hard to discuss the difference between the confirmation strategies with the average and standard deviation of the  $SSIM_{improvement}$ . We conducted additional analysis on two metrics  $SSIM_{improvement}/\#user\ turn$

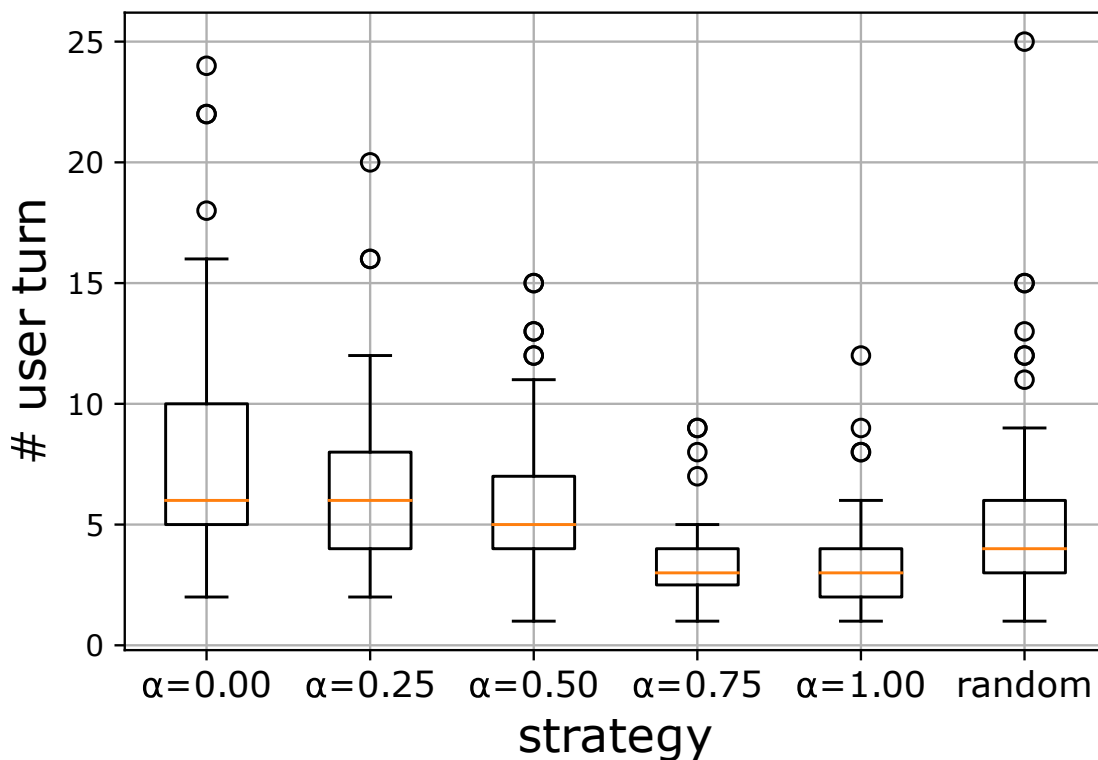


Figure 5.6. Distribution of  $\#user\ turn$  on each strategy: smaller  $\#user\ turn$  dialogue represents more efficient dialogue.

and  $\#user\ turn$  only.

#### Analysis with $SSIM_{improvement}/\#user\ turn$

An effective dialogue strategy satisfies not only the improvement of the image quality but also the efficiency of image editing dialogue; a shorter dialogue is better. To evaluate the whole dialogue performance in these two aspects, we visualized the histogram of  $SSIM_{improvement}/\#user\ turn$  collected from the end of the dialogues (Figure 5.5). We applied Mann–Whitney U test [41] to compare (a)  $\alpha = 0.0$ , (b)  $\alpha = 0.25$ , (c)  $\alpha = 0.50$ , and (d)  $\alpha = 0.75$  with (e)  $\alpha = 1.0$  and (f) *random*, and found significance of p-value  $< 0.001$  on the following: (a)  $\alpha = 0.0$  and (e)  $\alpha = 1.0$ , (b)  $\alpha = 0.25$  and (e)  $\alpha = 1.0$ , (c)  $\alpha = 0.50$  and (e)  $\alpha = 1.0$ , and (a)  $\alpha = 0.0$  and (f) *random*. This result indicates that the strategies with (a)  $\alpha = 0.0$  and (b)  $\alpha = 0.25$  realized a better SSIM with fewer

dialogue turns than the strategies with rarely confirming strategies ((d)  $\alpha = 0.75$  and (e)  $\alpha = 1.0$ ) or random confirmation strategy.

However, (a)  $\alpha = 0.0$  and (b)  $\alpha = 0.25$  were confirmed in most cases. When we compared all combinations of the two strategies in {(a)  $\alpha = 0.0$  and (b)  $\alpha = 0.25$ , and (b)  $\alpha = 0.50$ } , they were comparable.

### **Analysis with *#user turn***

We showed the distribution of *#user turn* for each strategy in Figure 5.6 to compare their effectiveness. We found a significance of p-value  $< 0.001$  between (c)  $\alpha = 0.50$  and (a)  $\alpha = 0.0$ , indicating that (c)  $\alpha = 0.50$  was a more efficient dialogue.

Although (c)  $\alpha = 0.50$  was not significant compared with (f) *random*, we found some interesting cases where the system used *confirm* and *not confirm* more properly than random. Figure 5.7 shows a dialogue example where the user discovered a good strategy. First, they tried to change the hair to a ponytail. The system successfully generated a ponytail image, but unintentionally changed the eyes to green. The user asked the system to change the eyes back to blue, and it successfully obeyed without any redundant confirmation on this turn. On the other hand, with the random confirmation strategy, the system occasionally confirmed with inappropriate timing. Figure 5.8 indicates an inefficient case. The system should have used *confirm* for editing request on  $i = 2$ , which indicate requests for changing to a smaller part. The user cannot fundamentally avoid such cases with the random confirmation strategy.

## **5.6. Discussion**

This chapter introduced an entropy-based confirmation method using a masking mechanism for interactive image editing. The mask mechanism is useful to deal with such complicated conditions as natural language, but such a strong constraint limits the acceptable language requests. In an avatar image editing task with natural language editing requests, changing such vast regions as the hair is restricted in the w/ mask constraint model. The system’s capability to confirm an action provides a chance to select a relevant image generated from both the

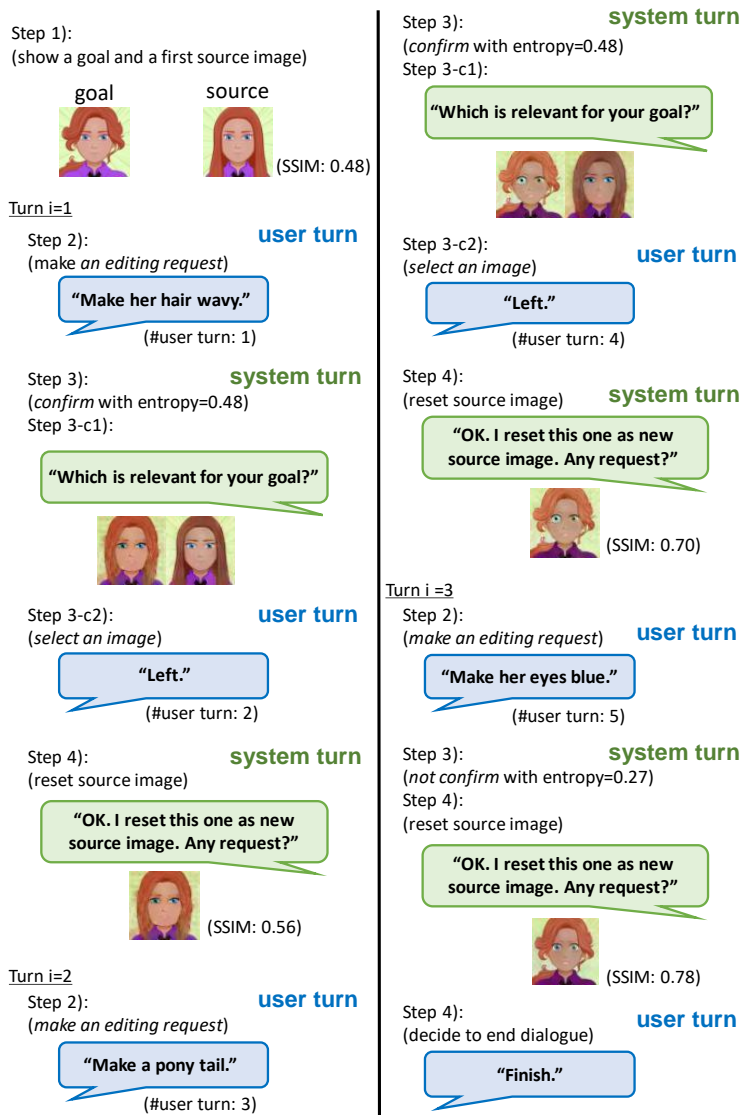


Figure 5.7. Dialogue example with  $\alpha = 0.50$  (confirmation threshold  $-\alpha \log 0.5 = 0.35$ ).  $i$  indicates turn index defined in Chapter 3.  $\#user\ turn$  denotes number of user actions, which represents total number of *making an editing request* and *selecting an image*. We put the source-goal SSIM next to each source image when the system decides a generated image of each turn.

w/o and w/ mask models. We demonstrated that our proposed strategy led to more similar images with fewer dialogue turns during human evaluations. We also

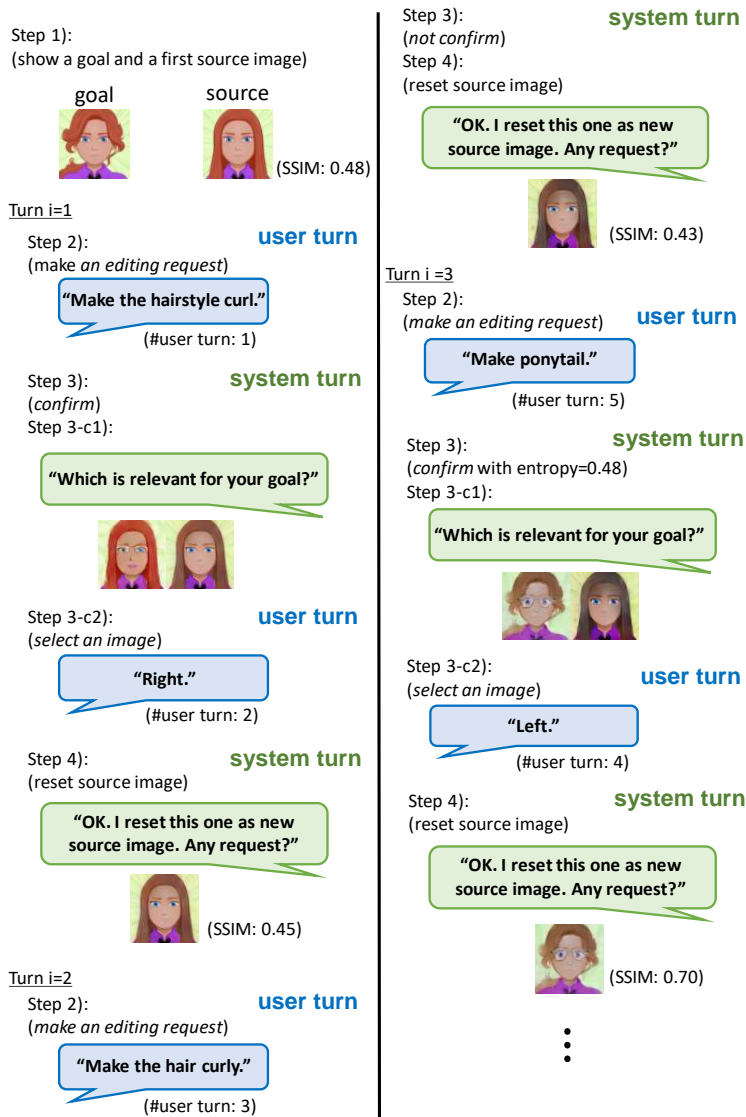


Figure 5.8. Inefficient dialogue example with random confirmation:  $i$  indicates turn index defined in Chapter 3.  $\#user\ turn$  denotes number of user actions, which represents total number of *making editing request* and *selecting an image*.

showed a compelling case where our confirmation method achieved an efficient dialogue strategy. It first changed a large part and then fine-tuned a small part.

A limitation of our confirmation strategy is that the confirmation threshold is a fixed parameter. The required sensitivity for the confirmation must be different



on user context, which includes such as the user purpose and user’s cultural background. Optimizing the confirmation threshold through dialogue feedback is useful for image editing. It is a challenging problem in our future work.

Another limitation is that the system is only aware of how good an edited image by w/ mask model is. The system cannot care about the result of the edited image by w/o model. If w/ mask model fails to edit the source image and w/o mask model successfully generates a right image, the user can select it. On the other hand, if both w/ and w/o mask models fail to edit the source image, the system forces the user to choose a failure image. The strategy without confirmation forces users to choose the failure image every time. It causes the users to give up in shorter dialogue.

Note that this task setting focused on a controlled setting to compare different confirmation strategy easily. In real scenarios, the system can select several choices when both edited images from the two models are not right. For example, the system can generate another image and confirm with the users again; the system also can ask the users to provide supportive information or redoing the edit with another editing request to generate more accurate images. They are a promising approach to make our editing model useful in the interactive image editing in future work. However, accepting many functions to the system makes it difficult to find the optimal behaviour for the system. One possible way is using reinforcement learning, although it requires a lot of dialogue data samples. We believe that our work could be the first step to simulate interactive image editing toward collecting dialogue dataset.

# Chapter 6

## Conclusion of this thesis

This thesis proposed an interactive image editing system that interacts with users using natural language. It focused on the editing requests for small and semantic changes of images that are difficult with image existing retrieval systems. We tackled two challenging problems due to the various editing requests to realize the system.

The first problem is that the system has to generate a new image according to users' editing request in natural language, including the requests for slight changes of images. Chapter 4 addressed this problem in our editing system. We proposed an interactive image editing framework based on a machine learning approach with neural network-based image generative models. It enabled the system to generate a new image from scratch according to the given source image and the user's editing request. We evaluated our editing model with an artificial dataset, which is automatically created from a public handwritten dataset, and more practical avatar face illustration editing, whose instructions are collected from human annotators. The first experiment with the artificial dataset, we verified our editing model worked for the limited type of instructions. However, the second experiment with avatar dataset, we found our editing model (without Source image masking model; w/o SIM) occurred unintentional changes, which are not mentioned in editing request. Our additionally proposed Source image masking model (w/ SIM) outperform the w/o SIM model in both objective evaluation and subjective evaluation.

The second problem is that the systems have to handle the uncertainty of the

generated images because a single editing model cannot accept all of the various editing requests. We proposed a confirmation strategy based on the uncertainty of generated image calculation with the mask. It enabled the system to confirm with the user only if the system was not confident to show the generated image from w/ SIM model. The w/ SIM model accepted most of the editing request for slight changes but was not good at that for significant change such as hair change in avatar editing dataset. The system’s capability to confirm action provided a chance to select a relevant image generated from both the w/o and w/ mask models. We demonstrated that our proposed strategy led to more similar images with lower the number of dialogue turns during human evaluations. We also showed an interesting case of whether our confirmation method achieved an efficient dialogue strategy. It first changed a large part and then fine-tuned a small part.

## 6.1. Remaining problems and future directions

In this section, we discuss the limitation of our proposed approaches.

First, our editing model with Artificial MNIST dataset has a problem that the detail of a source image collapses in a generated image on object movement, expansion, and compression. One promising solution in these editing requests is incorporating cropping spatial region and transform it using spatial transformer [27], which enables the affine transformation of images. Our image editing models require a large amount of pair of images and instruction. Replacing a part of edit operation with an existing operation such as affine transformation and existing image retrieval system will be helpful to expand feasible domain for editing.

Second, our editing models tend to fail to edit a small part, such as eyebrows and mouth. To encourage the model to pay attention to the small part, a cropping approach that edits regions after cropping is a possible way. It might be suitable for the editing task that can define the masks to crop; however, complicated tasks to define masks such as human hand-drawing or sketches editing would be still challenging topics for our editing models.

Third, our editing models require slightly different pairs of the source image

and target image, which can be expressed as a simple instruction. Slightly different pair collection is difficult on some editing tasks. For example, a realistic human face collection for our image editing task is difficult because any two different people have many differences on their faces. In this situation, our editing model with source image masking cannot handle this problem because any instructions will create a mask that indicates the whole face as an editing region. A promising task for our editing model is a creating illustration. If we record artists work continuously, each moment can be a feasible dataset for our editing model.

Fourth, the desired edit can be changed by user contexts. For example, in face editing, a request to change the colour may or may not imply the other hair colour such as eyebrows and moustache should also be changed. The correct intention is variable on task settings, for example, use case of editing, user's cultural background, and social trend. As the number of users becomes large, it will be hard to handle a minor intention because a general machine learning model learns major relationships in the given dataset. Toward a useful tool, our editing models need to consider this proper use for individual users through user modelling. Automatic adjustment to the individual user via dialogue is a useful function but a challenging topic for our future direction.

Fifth, the generated images are gradually degraded as the dialogue progresses in Chapter 5. It is because the models are trained with single turn editing dataset. If the dataset contains multiple turn dialogue data, it will be less problematic. However, collecting enough dialogue data for training image generator costs high. We need to investigate to improve the model, for example, data augmentation or regularization for the editing models.

Sixth, our proposed system's confirmation action in Chapter 5 is feasible for the model with a masking mechanism. The system is only aware of the uncertainty of the generated images of w/ mask model; therefore, if the system selects confirm, the other candidates can also be bad generated images. To solve this problem, we need to collect dialogue data to enable our system to learn more adaptive strategies using reinforcement learning.

Seventh, our confirmation strategy is based on fixed confirmation threshold. The required sensitivity for the confirmation must be different on user context,

which includes such as the user purpose and user’s cultural background. Optimizing the confirmation threshold through dialogue feedback is useful for image editing. It is a challenging problem in our future work.

Finally, this thesis only introduced system’s confirmation action by providing candidate images. However, there is a limitation of candidate images the users can see at once. To handle extremely ambiguous editing requests such as “make it stylish,” system needs to clarify users’ intention in natural language. It will be helpful to provide more relevant candidates quickly, and it will lead to more effective dialogue. For example, if the system can prepare a large number of candidate images for the ambiguous request, it takes much time for the user to find a relevant image. On the other hand, if the system can ask “Does stylish mean colour shape?,” the system can reduce the candidates according to the users’ answer. It will be useful for users. Considering another system actions such as asking additional supportive information to users or redoing the edit with another editing request are also beneficial for a real scenario of interactive image editing with natural language. However, accepting many functions to the system makes it difficult to find the optimal behaviour for the system. One possible way is using reinforcement learning, although it requires a lot of dialogue data sample. We believe that our work could be the first step to simulate interactive image editing toward collecting dialogue dataset.

## References

- [1] Giovanni Adorni, Mauro Di Manzo, and Fausto Giunchiglia. Natural language driven image generation. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, pages 495–500, Stanford, California, USA, July 1984. Association for Computational Linguistics.
- [2] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision*, pages 622–637. Springer, 2018.
- [3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2425–2433. IEEE Computer Society, 2015.
- [4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [5] G. Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- [6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *The International Conference on Learning Representations*, 2019.
- [7] Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *The International Conference on Learning Representations*, 2017.
- [8] Xinlei Chen and C. Lawrence Zitnick. Mind’s eye: A recurrent visual representation for image caption generation. In *IEEE Conference on Computer*

*Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2422–2431. IEEE Computer Society, 2015.

- [9] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111. Association for Computational Linguistics, 2014.
- [10] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014.
- [11] Herbert H Clark and Susan E Brennan. Grounding in communication. 1991.
- [12] Sharon Rose Clay and Jane Wilhelms. Put: language-based interactive manipulation of objects. *IEEE Computer Graphics and Applications*, 16(2):31–39, 1996.
- [13] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, Stefan Lee, José M. F. Moura, Devi Parikh, and Dhruv Batra. Visual dialog. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(5):1242–1256, 2019.
- [14] Abhishek Das, Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2970–2979. IEEE Computer Society, 2017.

- [15] Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. Guesswhat?! visual object discovery through multi-modal dialogue. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4466–4475. IEEE Computer Society, 2017.
- [16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *CoRR*, abs/1603.07285, 2016.
- [17] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 457–468. The Association for Computational Linguistics, 2016.
- [18] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 580–587. IEEE Computer Society, 2014.
- [19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
- [20] Xiaoxiao Guo, Hui Wu, Yu Cheng, Steven Rennie, Gerald Tesauro, and Rogério Schmidt Feris. Dialog-based interactive image retrieval. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 676–686, 2018.



- [21] Susumu Harada, Jacob O. Wobbrock, and James A. Landay. Voicedraw: a hands-free voice-driven drawing application for people with motor impairments. In Enrico Pontelli and Shari Trewin, editors, *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2007, Tempe, Arizona, USA, October 15-17, 2007*, pages 27–34. ACM, 2007.
- [22] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6626–6637, 2017.
- [23] Mayumi Hiyoshi and Hideo Shimazu. Drawing pictures with natural language and direct manipulation. In *15th International Conference on Computational Linguistics, COLING 1994, Kyoto, Japan, August 5-9, 1994*, pages 722–726, 1994.
- [24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.
- [26] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976. IEEE, 2017.
- [27] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In Corinna Cortes, Neil D.

- Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2017–2025, 2015.
- [28] Kevin Jarrett, Koray Kavukcuoglu, Marc’Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pages 2146–2153. IEEE Computer Society, 2009.
- [29] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The International Conference on Learning Representations*, 2015.
- [30] Diederik Kingma and Max Welling. Auto-encoding variational bayes. In *The International Conference on Learning Representations*, 2014.
- [31] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. 2014.
- [32] Kazunori Komatani and Tatsuya Kawahara. Flexible mixed-initiative dialogue management using concept-level confidence measures of speech recognizer output. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 467–473. Association for Computational Linguistics, 2000.
- [33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.
- [34] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation ap-

- plied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [35] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [36] Xiaodan Liang, Hao Zhang, Liang Lin, and Eric Xing. Generative semantic manipulation with mask-contrasting gan. In *The European Conference on Computer Vision*, pages 558–573, 2018.
- [37] T.-H. Lin, T. Bui, D. S. Kim, and J. Oh. A multimodal dialogue system for conversational image editing. In *Proceedings of The Second Workshop on Conversational AI at the Thirty-second Conference on Neural Information Processing Systems (NeurIPS 2018)*, November 2018.
- [38] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognit.*, 40(1):262–282, 2007.
- [39] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3431–3440. IEEE Computer Society, 2015.
- [40] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.
- [41] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.
- [42] Elman Mansimov, Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. In *The International Conference on Learning Representations*, 2016.

- [43] Ramesh Manuvinakurike, Trung Bui, Walter Chang, and Kallirroi Georgila. Conversational image editing: Incremental intent identification in a new dialogue task. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 284–295, Melbourne, Australia, July 2018. Association for Computational Linguistics.
- [44] Youssef Alami Mejjati, Christian Richardt, James Tompkin, Darren Cosker, and Kwang In Kim. Unsupervised attention-guided image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 3693–3703, 2018.
- [45] Teruhisa Misu and Tatsuya Kawahara. Bayes risk-based dialogue management for document retrieval system with speech interface. *Speech Communication*, 52(1):61–71, 2010.
- [46] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Instagan: Instance-aware image-to-image translation. In *The International Conference on Learning Representations (ICLR)*, 2019.
- [47] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814. Omnipress, 2010.
- [48] Shawn D. Newsam, Baris Sumengen, and B. S. Manjunath. Category-based image retrieval. In *Proceedings of the 2001 International Conference on Image Processing, ICIP 2001, Thessaloniki, Greece, October 7-10, 2001*, pages 596–599. IEEE, 2001.
- [49] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *NIPS 2016 Workshop on Adversarial Training*, 2016.
- [50] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2016.

- [51] Scott Reed, Zeynep Akata, Xinchun Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text-to-image synthesis. In *The International Conference on Machine Learning*, 2016.
- [52] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1252–1260, 2015.
- [53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- [54] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.
- [55] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer, 2017.
- [56] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 568–576, 2014.
- [57] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448, 2015.

- [58] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, pages 194–197. ISCA, 2012.
- [59] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.
- [60] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in Advances in Neural Information Processing Systems*, 2015.
- [61] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.
- [62] Jan van der Kamp and Veronica Sundstedt. Gaze and voice controlled drawing. In Veronica Sundstedt and Charlotte C. Sennersten, editors, *NGCA 2011, First Conference on Novel Gaze-Controlled Applications, Karlskrona, Sweden, May 26 - 27, 2011*, page 9. ACM, 2011.
- [63] Oriol Vinyals and Quoc V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015.
- [64] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3156–3164. IEEE Computer Society, 2015.
- [65] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

- [66] Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.
- [67] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision*, 2016.

## Publication lists

### Related to this thesis

#### Journal papers

1. Seitaro Shinagawa, Koichiro Yoshino, Sakti Sakriani, Yu Suzuki, Satoshi Nakamura.  
Image Manipulation System with Natural Language Instruction.  
IEICE Transactions on Information and Systems, Vol.J102-D, No.8, pp.514–529, August, 2019. (Chapter 3)
2. Seitaro Shinagawa, Koichiro Yoshino, Seyed Hossein, Alavi, Kallirrooi Georgila, David Traum, Sakriani Sakti, Satoshi Nakamura.  
An Interactive Image Editing System using an Uncertainty-based Confirmation Strategy.  
submitted to IEEE Access (Chapter 4, under review)

#### Conference papers (peer reviewed)

1. S.Shinagawa, K.Yoshino, S.Sakuriani, Y.Suzuki, S.Nakamura.  
Interactive image manipulation with natural language instruction commands. NIPS workshop, 2017. (Chapter 3)

#### Conference papers (without review)

1. Seitaro Shinagawa, Koichiro Yoshino, Sakriani Sakti, Yu Suzuki, Satoshi Nakamura.  
Interactive Avatar Image Manipulation with Unconstrained Natural Language Instruction using Source Image Masking.  
The 21st Meeting on Image Recognition and Understanding (MIRU), OS2-S3, 2018 【Selected as short oral】 【Student Award】 (Chapter 3)
2. 品川 政太郎, 吉野 幸一郎, サクティ サクリアニ, 中村 哲.  
DNNによる画像操作システム.  
電子情報通信学会パターン認識・メディア理解 (PRMU) 研究会, 2017 【PRMU 月間ベストプレゼンテーション賞】 (Chapter 3)



## Others

### Conference papers (peer reviewed)

1. Sara Asai, Koichiro Yoshino, Seitaro Shinagawa, Sakriani Sakti, Satoshi Nakamura.  
Emotional Speech Corpus for Persuasive Dialogue System.  
In Proceedings of 12th International Conference on Language Resources and Evaluation, Marseilles, France, 2020 (Accepted)

### Conference papers (without review)

1. 石井 大地, 久保 尋之, 品川 政太朗, 船富 卓哉, 前島 謙宣, 中村 哲, 向川 康博.  
セマンティックセグメンテーションと閉領域内のラベル投票によるアニメ線画の自動彩色.  
電子情報通信学会パターン認識・メディア理解 (PRMU) 研究会, 2019
2. 浅井 沙良, 品川 政太朗, 吉野 幸一郎, サクリアニ サクティ, 中村 哲.  
説得対話システムにおける感情表現を反映させた応答生成モデルの構築.  
情報処理学会第 240 回自然言語処理研究会, 2019
3. 生田 和也, 品川 政太朗, 吉野 幸一郎, 鈴木 優, 中村 哲.  
伝達内容を考慮して言語化するニューラル言語生成の検討. 情報処理学会第 232 回自然言語処理 (NL) 研究会, 2017
4. 品川 政太朗, 吉野 幸一郎, ニュービッグ グラム, 中村 哲.  
キャプションからの画像生成を行うニューラルネットへの対話的修正の導入と検討.  
2016 年度人工知能学会全国大会, 1A4-OS-27b-4, 2016

# Appendix

## A. Snippet for generating Artificial MNIST

The following snippet (stored in json) shows a part of example of information sets to generate triplets on a digit class five. Each key, e.g. "[1, 0, 2, 2]->[0, 0, 3, 3]", represents the source and target image parameter set  $(x, y, w, h)$  for rendering. Instruction (as shown in "instruction") is automatically generated by combination of pre-defined templates and corresponding "command" set, which consists of action and direction information of the instructions.

```
1 {
2   "[1, 0, 2, 2]->[0, 0, 3, 3]": {
3     "ID": 90,
4     "instruction": "expand five to the bottom left .",
5     "command": [
6       "expand",
7       [
8         "bottom",
9         "left"
10      ]
11    ]
12  },
13  "[0, 0, 2, 1]->[1, 0, 1, 1]": {
14    "ID": 207,
15    "instruction": "compress five to the right .",
16    "command": [
17      "compress",
18      [
19        "right"
20      ]
21    ]
22  },
23  "[0, 0, 2, 3]->[1, 0, 2, 3]": {
24    "ID": 282,
25    "instruction": "move five to the right .",
26    "command": [
27      "move",
28      [
```

```

29         "right"
30     ]
31 ]
32 },
33 "[0, 0, 0, 0]->[0, 2, 1, 1]": {
34     "ID": 6,
35     "instruction": "put five on the bottom left .",
36     "command": [
37         "put",
38         [
39             "bottom",
40             "left"
41         ]
42     ]
43 },
44 "[1, 2, 1, 1]->[0, 0, 0, 0]": {
45     "command": [
46         "remove"
47     ],
48     "instruction": "remove five",
49     "ID": 335
50 },

```

## B. Model architectures for the experiment

Table 6.1, Table 6.2, and Table 6.3 show the detailed network architectures for IMI models. Table 6.1 represents IMI model for Artificial MNIST dataset. Table 6.2 shows IMI model for AIMI dataset, and Table 6.3 represents IMI model with source image masking for AIMI dataset in the experiment. In “*Input*  $\rightarrow$  *OutputShape*,” the vectors with three dimension indicates (*channel, height, width*).  $w^t$  and  $e^t$  denote t-th word one-hot vector and its embed vector of Instruction Encoder ( $E_i$ ).  $\phi_i$  represent  $\phi_i^T$  where the input instruction sequence length is  $T$ .  $z$  denotes a noise vector sampled from Normal distribution. In Layer Information, Linear denotes a fully-connected layer, “CONV” represents a convolution layer with the number of kernels (N), kernel size (K), stride size (S), padding size (P), i.e., “CONV-(N64, K4x4, S2, P1)” is composed of the convolution layer with 64 kernels, kernel window of (*height, width*) = (4, 4), stride size of 2, padding size

of 1. “TCONV” represents a transposed convolution layer, which is available as “`chainer.links.Deconvolution2D`” in Chainer. “BN” denotes a batch normalization layer. We used the following activation functions: sigmoid (Sigmoid), hyperbolic tangent (Tanh), rectified linear unit (ReLU), leaky rectified linear unit (Leaky ReLU) of the slope of 0.2.

Table 6.1. IMI model for Artificial MNIST dataset.

module	Input $\rightarrow$ Output Shape	Layer Information
Image Encoder ( $E_{im}$ )	$(3, 64, 64) \rightarrow (64, 32, 32)$	CONV-(N64, K4x4, S2, P1), LeakyReLU
	$(64, 16, 16) \rightarrow (128, 16, 16)$	CONV-(N128, K4x4, S2, P1), LeakyReLU, BN
	$(128, 16, 16) \rightarrow (256, 8, 8)$	CONV-(N256, K4x4, S2, P1), LeakyReLU, BN
	$(256, 8, 8) \rightarrow (512, 4, 4)$	CONV-(N512, K4x4, S2, P1), LeakyReLU, BN
	$(512, 4, 4) \rightarrow (512 \cdot 4 \cdot 4)$	Reshape
	$(512 \cdot 4 \cdot 4) \rightarrow \phi_{im}(1024)$	Linear
Instruction Encoder ( $E_i$ )	$w^t(1892) \rightarrow e^t(128)$	Linear
	$e^t(128), \phi_i^{t-1}(128) \rightarrow \phi_i^t(128)$	Linear, Tanh
Fully-connected ( $FC$ )	$\phi_{im}(1024), \phi_i(128) \rightarrow \phi_{fc}(128)$	Linear
Generator ( $G$ )	$z(128), \phi_{fc}(128) \rightarrow (512 \cdot 4 \cdot 4)$	Linear, BN, ReLU
	$(512 \cdot 4 \cdot 4) \rightarrow (512, 4, 4)$	Reshape
	$(512, 4, 4) \rightarrow (256, 8, 8)$	TCONV-(N256, K4x4, S2, P1), BN, ReLU
	$(256, 8, 8) \rightarrow (128, 16, 16)$	TCONV-(N128, K4x4, S2, P1), BN, ReLU
	$(128, 16, 16) \rightarrow (64, 32, 32)$	TCONV-(N64, K4x4, S2, P1), BN, ReLU
	$(64, 32, 32) \rightarrow (3, 64, 64)$	TCONV-(N3, K4x4, S2, P1), Tanh
Discriminator ( $D$ )	$(3, 64, 64) \rightarrow (64, 32, 32)$	CONV-(N64, K4x4, S2, P1), LeakyReLU
	$(64, 16, 16) \rightarrow (128, 16, 16)$	CONV-(N128, K4x4, S2, P1), LeakyReLU, BN
	$(128, 16, 16) \rightarrow (256, 8, 8)$	CONV-(N256, K4x4, S2, P1), LeakyReLU, BN
	$(256, 8, 8) \rightarrow (512, 4, 4)$	CONV-(N512, K4x4, S2, P1), LeakyReLU, BN
	$(512, 4, 4) \rightarrow (512 \cdot 4 \cdot 4)$	Reshape
	$(512 \cdot 4 \cdot 4) \rightarrow h(128)$	Linear, LeakyReLU
	$h(128), \phi_{fc}(128) \rightarrow (128)$	Linear, LeakyReLU
	$h(128) \rightarrow (11)$	Linear
	$h(128) \rightarrow (3)$	Linear
	$h(128) \rightarrow (3)$	Linear
	$h(128) \rightarrow (4)$	Linear
	$h(128) \rightarrow (4)$	Linear
$(128) \rightarrow (1)$	Linear, Sigmoid	

Table 6.2. IMI model for AIMI dataset.

module	Input $\rightarrow$ Output Shape	Layer Information
Image Encoder ( $E_{im}$ )	$(3, 64, 64) \rightarrow (64, 32, 32)$	CONV-(N64, K4x4, S2, P1), LeakyReLU
	$(64, 16, 16) \rightarrow (128, 16, 16)$	CONV-(N128, K4x4, S2, P1), LeakyReLU, BN
	$(128, 16, 16) \rightarrow (256, 8, 8)$	CONV-(N256, K4x4, S2, P1), LeakyReLU, BN
	$(256, 8, 8) \rightarrow (512, 4, 4)$	CONV-(N512, K4x4, S2, P1), LeakyReLU, BN
	$(512, 4, 4) \rightarrow (512 \cdot 4 \cdot 4)$	Reshape
	$(512 \cdot 4 \cdot 4) \rightarrow \phi_{im}(1024)$	Linear
Instruction Encoder ( $E_i$ )	$w^t(1892) \rightarrow e^t(128)$	Linear
	$e^t(128), \phi_i^{t-1}(128) \rightarrow \phi_i^t(128)$	Linear, Tanh
Fully-connected ( $FC$ )	$\phi_{im}(1024), \phi_i(128) \rightarrow \phi_{fc}(128)$	Linear, Sigmoid
Generator ( $G$ )	$z(128), \phi_{fc}(128) \rightarrow (512 \cdot 4 \cdot 4)$	Linear, BN, ReLU
	$(512 \cdot 4 \cdot 4) \rightarrow (512, 4, 4)$	Reshape
	$(512, 4, 4) \rightarrow (256, 8, 8)$	TCONV-(N256, K4x4, S2, P1), BN, ReLU
	$(256, 8, 8) \rightarrow (128, 16, 16)$	TCONV-(N128, K4x4, S2, P1), BN, ReLU
	$(128, 16, 16) \rightarrow (64, 32, 32)$	TCONV-(N64, K4x4, S2, P1), BN, ReLU
	$(64, 32, 32) \rightarrow (3, 64, 64)$	TCONV-(N3, K4x4, S2, P1), Tanh
Discriminator ( $D$ )	$(3, 64, 64) \rightarrow (64, 32, 32)$	CONV-(N64, K4x4, S2, P1), LeakyReLU
	$(64, 16, 16) \rightarrow (128, 16, 16)$	CONV-(N128, K4x4, S2, P1), LeakyReLU, BN
	$(128, 16, 16) \rightarrow (256, 8, 8)$	CONV-(N256, K4x4, S2, P1), LeakyReLU, BN
	$(256, 8, 8) \rightarrow (512, 4, 4)$	CONV-(N512, K4x4, S2, P1), LeakyReLU, BN
	$(512, 4, 4) \rightarrow (512 \cdot 4 \cdot 4)$	Reshape
	$(512 \cdot 4 \cdot 4) \rightarrow h(128)$	Linear, LeakyReLU
	$h(128), \phi_{fc}(128) \rightarrow (128)$	Linear, LeakyReLU
	$(128) \rightarrow (1)$	Linear, Sigmoid

Table 6.3. IMI model with Source image masking for AIMI dataset.

module	Input $\rightarrow$ Output Shape	Layer Information
Generator ( $G$ )	$z(128), \phi_{fc}(128) \rightarrow (512 \cdot 4 \cdot 4)$	Linear, BN, ReLU
	$(512 \cdot 4 \cdot 4) \rightarrow h(512, 4, 4)$	Reshape
	$h(512, 4, 4), \phi_{imm}(512, 4, 4)$	
	$\rightarrow (1024, 4, 4)$	Concatenate channel
	$(1024, 4, 4) \rightarrow (256, 8, 8)$	TCONV-(N256, K4x4, S2, P1), BN, ReLU
	$(256, 8, 8) \rightarrow (128, 16, 16)$	TCONV-(N128, K4x4, S2, P1), BN, ReLU
	$(128, 16, 16) \rightarrow (64, 32, 32)$	TCONV-(N64, K4x4, S2, P1), BN, ReLU
	$(64, 32, 32) \rightarrow (3, 64, 64)$	TCONV-(N3, K4x4, S2, P1), Tanh
Discriminator ( $D$ )	$(3, 64, 64) \rightarrow (64, 32, 32)$	CONV-(N64, K4x4, S2, P1), LeakyReLU
	$(64, 16, 16) \rightarrow (128, 16, 16)$	CONV-(N128, K4x4, S2, P1), LeakyReLU, BN
	$(128, 16, 16) \rightarrow (256, 8, 8)$	CONV-(N256, K4x4, S2, P1), LeakyReLU, BN
	$(256, 8, 8) \rightarrow (512, 4, 4)$	CONV-(N512, K4x4, S2, P1), LeakyReLU, BN
	$(512, 4, 4) \rightarrow (512 \cdot 4 \cdot 4)$	Reshape
	$(512 \cdot 4 \cdot 4) \rightarrow h(128)$	Linear, LeakyReLU
	$h(128), \phi_{fc}(128) \rightarrow h(1)$	Linear, Sigmoid
Mask Generator ( $G_m$ )	$\phi_{im}(1024), \phi_i(128) \rightarrow (512 \cdot 4 \cdot 4)$	Linear, BN, ReLU
	$(512 \cdot 4 \cdot 4) \rightarrow (512, 4, 4)$	Reshape
	$(512, 4, 4) \rightarrow (256, 8, 8)$	TCONV-(N256, K4x4, S2, P1), BN, ReLU
	$(256, 8, 8) \rightarrow (128, 16, 16)$	TCONV-(N128, K4x4, S2, P1), BN, ReLU
	$(128, 16, 16) \rightarrow (64, 32, 32)$	TCONV-(N64, K4x4, S2, P1), BN, ReLU
	$(64, 32, 32) \rightarrow (1, 64, 64)$	TCONV-(N1, K4x4, S2, P1), Sigmoid
Image Encoder with mask ( $E_{imm}$ )	$(3, 64, 64) \rightarrow (64, 32, 32)$	CONV-(N64, K4x4, S2, P1), LeakyReLU
	$(64, 16, 16) \rightarrow (128, 16, 16)$	CONV-(N128, K4x4, S2, P1), LeakyReLU, BN
	$(128, 16, 16) \rightarrow (256, 8, 8)$	CONV-(N256, K4x4, S2, P1), LeakyReLU, BN
	$(256, 8, 8) \rightarrow (512, 4, 4)$	CONV-(N512, K4x4, S2, P1), LeakyReLU, BN