Analysis of Mouth Shape Deformation Rate for the Automatic Generation of Japanese Utterance Images
Twuyoshi Miyazakia and Nakashima Toyoshirob

Heuristics for Daihinmin and their Effectiveness
Yasuhiro Tajimaab, Kouzou Tagashirac and Genichiro Kikui

Analysis Method Focusing on Peaks for Spatial Evaluation of Impression
Shunsuke Akai, Teruhisa Hochin and Hiroki Nomiya

Estimating Prediction Interval of Cumulative Number of Software Faults Using Back Propagation Algorithm
Begum Momotaz and Tadashi Dohi
Analysis of Mouth Shape Deformation Rate for the Automatic Generation of Japanese Utterance Images

Twsuyoshi Miyazaki\textsuperscript{a}
Kanagawa Institute of Technology, Japan
Nakashima Toyoshir\textsuperscript{b}
Sugiyama Jogakuen University, Japan

Abstract

We have conducted research on machine lip-reading. We propose a method to reproduce utterance images without voice from Japanese syllabary. A sequence of codes, mouth shapes sequence code (MSSC) is generated. The MSSC expresses the order of mouth shapes when a Japanese word is uttered. The utterance images are then generated by using mouth images that correspond to the MSSC, and deformed mouth shape images are generated by morphing. However, the deformation rate of the mouth shapes is determined experimentally from real utterance images. As a result, incongruous mouth shape deformation images were generated. In this paper, the deformation rate of mouth shapes is analyzed using real utterance images captured by a high-speed camera. We propose a method to generate utterance images based on the results. Finally, the mean opinion score of the subjects are presented, and we evaluate the effectiveness of the proposed method.

Keywords: lip-reading training, teaching materials, computer graphics

1. Introduction

We have been doing research on technology to support communication for the hearing-impaired. Typically, the hearing-impaired communicate using sign language, written communication, and lip-reading. Sign language is effective; however, it requires both participants to possess sign language skills. Therefore, the number of people that the hearing-impaired can communicate with is limited. When the hearing-impaired communicate with people without sign language skills, they typically use written communication. With written communication, the hearing-impaired can communicate with more people, but it takes longer to establish mutual understanding. Consequently, it is not a particularly efficient way to conduct a conversation. By using lip-reading, the hearing-impaired can communicate with the hearing-enabled; however, lip-reading skills are required.

Typically, the hearing-impaired learn lip-reading from a trainer in a face-to-face manner or from educational video materials that focus on utterance images. The learning efficacy of the former is higher than the latter; however, the trainer must be able to lip read, and learning sessions occur at a specific time. Moreover, while using video materials, hearing-impaired individuals can acquire skills at their convenience. However, lip movements for spoken phrases are difficult to convey accurately in a video. Therefore, the proficiency of learning by using video materials varies. If the utterance images of phrases can be generated, we consider that such images can be used as lip-reading teaching materials.

Studies to generate utterance images via a computer include “visual speech synthesis” (VSS). VSS generates lip movement images synchronized with voice data. VSS renders the mouth shape and facial expression to a 3D face model by using features generated from the voice data [1, 2]. In a study of an anthropomorphic spoken dialog agent, a computer-generated model agent moves the mouth in sync with synthesized speech and shows physical manerisms that are similar to speaking [3]. In these studies, the goal was to synchronize lip deformation with the voice. However, we aim to generate utterance images that can be used as teaching materials in order to acquire lip-reading skills. Therefore, the lip movements of the utterance images must be realistic. To determine mouth deformation, statistical methods, such as the hidden Markov model, have been used [2]. Japanese language has unique relationships between voice and mouth shape [4]; therefore, our study utilizes this association.

We have proposed a method to generate utterance images without voice automatically from the kana of a Japanese word [5]. Using the proposed method, it is possible to generate utterance images that represent correct mouth shape deformation. In that study, an Android application was developed experimentally [6]. However, these images were generated at 30 frames per second (fps), and the deformation rate of mouth shape was set based on observation. Consequently, the deformation of mouth shape was inadequate.

In this study, the mouth shape deformation rate derived from the movement of lips is analyzed in detail from images captured by a high-speed camera. Images are generated based on the deformation rate and are evaluated by subjects.
2. Analysis of Mouth Shape Deformation Rate

To analyze lip movement when Japanese words and phrases are uttered, a high-speed camera (250 fps capture rate) captures the images of the lips. To facilitate analysis, four blue markers are used, as seen in Figure 1. The nostril area is also tracked because the face moves when people speak.

![Figure 1](image)

Figure 1 Four blue markers for lip movement analysis

2.1 Acquiring Sequential Coordinates of the Markers

The process flow used to detect the markers from an image captured by the camera is shown in Figure 2. First, the color space is converted into the HSV color space. Second, the blue pixels of the markers in the hue plane are extracted. Then, the marker areas (i.e., blue pixels) and the background are separated by binarization. However, areas that are not the markers and pixels referred to as holes are not detected adequately. These areas or pixels are small; thus, misdetected areas can be cleared and the holes are filled with the opening and closing processes, respectively. Finally, the four marker areas are detected with the above process and each central coordinate is calculated. Thus, time series coordinates are acquired using the applied process.

For example, images generated by each process are shown in Figure 3. Figure 3(b) shows a binarized image in which the pixels of Figure 3(a) were divided by the marker color. Here, white pixels belong to the marker, and black pixels belong to the background. However, this image still includes misdetected areas and holes. Figure 3(c) shows the image obtained after the opening process was applied to the image shown in Figure 3(b). The misdetected areas are cleaned with this process. Figure 3(d) shows an image obtained after the Closing process was applied to the image shown in Figure 3(c). The holes are filled with this process.

![Figure 2](image)

Figure 2 Marker detection flow

![Figure 3](image)

Figure 3 (a) Captured mouth image, (b) binarized image with marker color, (c) image obtained from the opening process, and (d) an image obtained from the closing process

2.2 Approximate Equation of Mouth Shape Deformation Rate

After analyzing the changes in the sequential coordinate values of the four markers, it became clear that the upper lip and both corners of the mouth do not move frequently. In addition, the lower lip scarcely moved horizontally; however, it was observed to move significantly in vertical directions. Figure 4 shows the sequential y-coordinate
values of the lower lip when a person uttered “MA-MI-MU-ME-MO.”

As can be seen in Figure 4, in the period changing in upward slant to the right, the lower lip moves downward, and in the period changing in downward sloping, it moves upward. Figure 5 shows the values for period (a) in Figure 4 and a cubic equation curve. Figure 6 shows the values of period (b) (Figure 4) and the cubic equation curve. As can be seen, the movement of the lower lip can be approximated by a cubic equation.

3. Generation of Japanese Utterance Images

Deformed mouth shape images are generated using a cubic equation. A method that we have proposed previously is used to generate Japanese utterance images[5].

First, the utterance word is input in Japanese. The word is converted into “mouth shape sequence code” (MSSC) [4]. The deformed mouth shape images are generated from the MSSC with a morphing method [5] because MSSC expresses a sequence of mouth shapes when a word is uttered. The images shown in Table 1 are used as key frames when deformed mouth shape images are generated. However, there are lip, teeth, and buccal areas (inside of mouth) in the images, and these areas differ from the mouth shapes. Therefore, it is necessary to consider this difference when deformed mouth shape images are generated.

Table 1 Images of basic mouth shape, and the presence of teeth and buccal areas

<table>
<thead>
<tr>
<th>Basic mouth shape</th>
<th>Image</th>
<th>Teeth area</th>
<th>Buccal area</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td><img src="image1.png" alt="Image" /></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>I</td>
<td><img src="image2.png" alt="Image" /></td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>U</td>
<td><img src="image3.png" alt="Image" /></td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>E</td>
<td><img src="image4.png" alt="Image" /></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>O</td>
<td><img src="image5.png" alt="Image" /></td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>X</td>
<td><img src="image6.png" alt="Image" /></td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

For example, when the mouth shape deforms from A to E, each area can be transformed between corresponding areas because the teeth and buccal areas exist. When the mouth
shape deforms from \( I \) to \( O \), the transformation between the corresponding areas cannot be processed because each area differs. Therefore, the deformed area is compounded with a gray image.

When utterance images are displayed at 60 fps (Section 2.1), the number of frames required for the deformation of the mouth shape is 7. The mouth shape deformation rate for the frame is shown in Figure 7. Generated mouth shape images are shown in Figure 8, which shows mouth shape deformation from closed mouth to mouth shape /a/. The frames #0 and #6 are the key frames, and frames #1–#5 are the generated images.

4. Experiments

We performed experiments to evaluate the generation method for Japanese utterance images. We evaluated the fluency of utterance images and the comprehension of uttered words with 15 subjects who did not possess lip-reading skills. The utterance speeds of the images were “slow,” “normal,” and “fast.” Even if the number of the mouth shape deformation frames changes according to the changes in utterance speed, the mouth shape deformation rate can be calculated easily from the approximate equation described in Section 2.2.

4.1 Fluency of the Utterance Images

We presented the images three times for each utterance speed, and the subjects evaluated the images on a 5-point scale. The utterance words are shown in Table 2. A high rank indicates that the mouth shape deformation was fluent. Mean opinion scores are shown in Table 3 and illustrated in a bar chart in Figure 9.

Table 2 Utterance words and their mouth shape sequence code for the first experiment

<table>
<thead>
<tr>
<th>#</th>
<th>Word (in Japanese)</th>
<th>English</th>
<th>MSSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>KATATSUMURI</td>
<td>a snail</td>
<td>-AIE=UXE-IXU</td>
</tr>
<tr>
<td>2</td>
<td>KAWAKUDARI</td>
<td>going downstream in a boat</td>
<td>-AIA=UAE=I</td>
</tr>
<tr>
<td>3</td>
<td>KAMISHIBAI</td>
<td>a story told with pictures</td>
<td>-AXI=IXA=I</td>
</tr>
<tr>
<td>4</td>
<td>ASESUMENTO</td>
<td>an assessment</td>
<td>-AIE=UXE=I</td>
</tr>
<tr>
<td>5</td>
<td>SUPOTTORAITO</td>
<td>a spotlight</td>
<td>-UXO=UXO=I</td>
</tr>
</tbody>
</table>

Table 3 Mean opinion scores of utterance images

<table>
<thead>
<tr>
<th>#</th>
<th>Word (in Japanese)</th>
<th>Normal</th>
<th>Slow</th>
<th>Fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>KATATSUMURI</td>
<td>3.60</td>
<td>4.20</td>
<td>3.13</td>
</tr>
<tr>
<td>2</td>
<td>KAWAKUDARI</td>
<td>3.93</td>
<td>3.20</td>
<td>3.00</td>
</tr>
<tr>
<td>3</td>
<td>KAMISHIBAI</td>
<td>2.33</td>
<td>2.60</td>
<td>2.33</td>
</tr>
<tr>
<td>4</td>
<td>ASESUMENTO</td>
<td>4.07</td>
<td>4.67</td>
<td>3.80</td>
</tr>
<tr>
<td>5</td>
<td>SUPOTTORAITO</td>
<td>3.40</td>
<td>3.47</td>
<td>2.67</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>3.47</td>
<td>3.51</td>
<td>3.03</td>
</tr>
</tbody>
</table>

Figure 9 Results of mean opinion score

As a result, we consider that the images for word #5 (ASESUMENTO) are fluent because the scores are close to 4 for each utterance speed. On the other hand, the images for word #3 (KAMISHIBAI) are not fluent because the scores are less than 3. Here, we consider that consecutive voicing of syllables “MI” and “SHI” is a factor. Both mouth shapes are the same because the two sounds possess the same vowel. There were subjects who commented that it was difficult to distinguish “MI” and “SHI.” Thus, it is necessary to improve the display method of the mouth for similar vowel sounds.
4.2 Comprehension of the Utterance Word

The utterance images of the words shown in Table 4 were generated. Here, the utterance speed was “Normal” and “Slow.” As with the previous experiment, the utterance images were shown to the subjects three times. However, the subjects were informed about a category hint (Table 4) rather than the uttered word.

Table 4 Utterance words and their mouth shape sequence code

<table>
<thead>
<tr>
<th>#</th>
<th>Word (in Japanese)</th>
<th>English</th>
<th>MSSC</th>
<th>Category hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUIKA</td>
<td>a watermelon</td>
<td>Ｕ－Ｉ－Ａ</td>
<td>Vegetables or fruits</td>
</tr>
<tr>
<td>2</td>
<td>KARASU</td>
<td>a crow</td>
<td>ＡＩ－Ａ</td>
<td>Bird</td>
</tr>
<tr>
<td>3</td>
<td>FUDEBAKO</td>
<td>a pen case</td>
<td>Ｕ－Ｉ－Ｅ－Ｏ</td>
<td>Stationery</td>
</tr>
<tr>
<td>4</td>
<td>KAWASAKI</td>
<td>Kawasaki city</td>
<td>Ａ－Ａ－Ｉ</td>
<td>Name of a city in Japanese</td>
</tr>
<tr>
<td>5</td>
<td>UDEDOKEI</td>
<td>a wristwatch</td>
<td>Ｕ－Ｉ－Ｅ－Ｉ</td>
<td>Wearable item</td>
</tr>
<tr>
<td>6</td>
<td>SAIBANSYO</td>
<td>a court</td>
<td>Ｉ－Ｉ－Ａ－Ｉ－Ｕ</td>
<td>Landmark</td>
</tr>
</tbody>
</table>

Numerical comprehension rates for each word are shown in Table 5 and charted graphically in Figure 10. The comprehension rates for words #2 and #3 were high; however, only one subject understood word #6. The subjects commented that they could understand the last part of word #6 but could not understand the former part. In addition, it was difficult to comprehend word #6 because the word had more “beginning mouth shapes”[4] than the other words.

Table 5 Comprehension rates of utterance words

<table>
<thead>
<tr>
<th>#</th>
<th>Word</th>
<th>Normal</th>
<th>Slow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUIKA</td>
<td>60.0%</td>
<td>73.3%</td>
</tr>
<tr>
<td>2</td>
<td>KARASU</td>
<td>80.0%</td>
<td>86.7%</td>
</tr>
<tr>
<td>3</td>
<td>FUDEBAKO</td>
<td>80.0%</td>
<td>86.7%</td>
</tr>
<tr>
<td>4</td>
<td>KAWASAKI</td>
<td>66.7%</td>
<td>80.0%</td>
</tr>
<tr>
<td>5</td>
<td>UDEDOKEI</td>
<td>53.3%</td>
<td>73.0%</td>
</tr>
<tr>
<td>6</td>
<td>SAIBANSYO</td>
<td>6.7%</td>
<td>6.7%</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>57.8%</td>
<td>67.8%</td>
</tr>
</tbody>
</table>

Figure 10 Correct answer rates

However, the comprehension rate was high, with the exception of word #6, because the subjects did not have lip-reading skills. From these results, the evaluation of the fluency of the utterance images was not high in the first experiment. However, utterance images for words that could be understood were generated. In addition, for most words, the utterance speed was regulated appropriately because the comprehension rate of slow utterances was higher than the norm.

5. Conclusion

We have proposed a method to generate Japanese utterance images automatically for lip-reading teaching materials for the hearing impaired. The deformation of mouth shape was analyzed using a camera that can capture images at a high frame rate. In addition, the deformation rate of mouth shapes was derived from the results. Experiments that examined the fluency of the utterance images did not show positive evaluation. However, it is considered that the utterance images showed correct deformation of mouth shape because an experiment designed to examine the comprehension of an utterance word from utterance images showed positive evaluation. Thus, it is considered effective to generate images with different utterance speeds because the comprehension rate of slow utterance images was high. In the future, we plan to incorporate the proposed method into a lip-reading training application that runs on Android devices.

References


Tsuyoshi Miyazaki is an associate professor in the department of Information and Computer Sciences at Kanagawa Institute of Technology. Dr. Miyazaki’s research focus is information engineering for human well-being and image recognition. He has investigated issues related to supporting the hearing-impaired, machine lip-reading, computer vision, and image processing. He is a member of the Information Processing Society of Japan and the Institute of Electronics, Information and Communication Engineers of Japan.

Toyoshiro Nakashima is a professor in the department of Culture-Information Studies at Sugiyama Jogakuen University. Dr. Nakashima’s research focus is software engineering and information technology. He has primarily worked in the research and development of Programmable Logic Controllers and retail systems at the Omron Corporation. He has also investigated issues in embedded system software development, human interfaces, and electronic educational materials at Sugiyama Jogakuen University. He is a member of the Institute of Electrical Engineers of Japan, the Information Processing Society of Japan, the Institute of Electronics, Information and Communication Engineers of Japan, the Japan Society for Simulation Technology, and the IEEE Computer Society.
Heuristics for Daihinmin and their effectiveness

Yasuhiro Tajima\textsuperscript{a,b}, Kouzou Tagashira\textsuperscript{c}, Genichiro Kikui\textsuperscript{a}
Okayama Prefectural University

Abstract

Heuristics for games is one of the most important factors to solve or to make a game playing program. Especially, there are many heuristics for perfect information games. On the other hand, imperfect information game is not studied enough to make a program which can overcome a human player. In this paper, we show four heuristics for the card game “Daihinmin” and evaluate their effectiveness by some experiments. Our heuristics are implemented on the program called ”kou” which is the champion program at UECda-2014 light class. This program uses only heuristics to play Daihinmin. In evaluation experiments, we show the strongness of ”kou” by matching past champions which contain both of Monte-Carlo algorithm and heuristic algorithm. The result is that our heuristics are effective and our program is as strong as Monte-Carlo algorithm.

Keywords: Daihinmin, imperfect information game, heuristics, card game, evaluate function.

1. Introduction

Game study is one of the symbolic field of study of intelligence. If a computer wins a human, everybody would recognize a solid progress of computer science. Study on games can be classified into perfect information games and imperfect information games. The perfect information game can be solved by game tree search. The $\alpha - \beta$ method is a famous game tree searching method. In recent years, Monte-Carlo tree search\textsuperscript{[3]} is known as effective method for game tree search. Especially, Go is the most successful application of Monte-Carlo tree search\textsuperscript{[4]}. Computer Shogi is also the frontier of studies on perfect information games.

In [5], we can see that recent computer systems can win a professional shogi player in even conditions. Solving these perfect information games is usually executed by a deterministic algorithm. Then, these results can be useful for another algorithm study. Especially, automata theory and its application\textsuperscript{[10]} is the basic and important problem of computer science. Game tree search algorithm and studies on perfect information games bring useful results for these fields.

For perfect information games, heuristics are also important as the game tree search algorithm. Even if there is a fast game tree search algorithm, a bad evaluation function leads lose easily. To make a good evaluation function, heuristics for the game is important. Recently, many evaluation functions are made by machine learning method but features of machine learning are usually constructed from heuristic ideas of the games.

Unfortunately, such heuristic ideas can not be re-used for another game. Heuristics are deeply dependent on each game and specific heuristics tend to be more important for the game.

On the other hand, there is no effective tree searching method for imperfect information games. There is a difficulty that possible states are too many because of imperfectness, and then there are too many branches if we create a game tree. But some studies and progress can be found on imperfect information games [1][2]. Many card games are imperfect information games. In addition, many card games uses standard pack of 52 cards with one or two jokers. Thus, we can regard these card games as imperfect information but not random game except the first shuffling.

“Daihinmin” is a famous card game in Japan. This game is similar to Big-Two which is also a card game mainly played in China\textsuperscript{[7]}. The player is more than 2, and then Daihinmin is a multi-player, imperfect information game. There is a competition of computer Daihinmin called “UECda”\textsuperscript{[11]}. In this annually competition, Monte-Carlo simulation method is an effective move searching method\textsuperscript{[8]} and all champions uses this method in recent years.

In this paper, we show some heuristics for Daihinmin and evaluate whose effects by experiments. The base program to implement our heuristics is called “kou” which is the champion program at light class in UECda-2014 and this does not use Monte-Carlo method. This program is as strong as “snowl”\textsuperscript{[9]} which is the champion program at UECda-2010 and uses Monte-Carlo method. In addition, the time complexity of our program is about one hundred
times faster than that of Monte-Carlo method program. This is a new possibility of heuristics on computer Daihinmin.

2. Rules of “Daihinmin”

Daihinmin is a multi-player imperfect information game and uses standard pack of 52 cards with one joker. The number of players is arbitrarily more than 2 but 5 players are the best balance to play. In the following, we assume that the number of players is 5.

At first, shuffled almost even number of cards are provided to every player. The 53 cards can not be divided evenly by 5 players, thus three players have 11 cards and two players have 10 cards. The rank of cards has the order 3,4,5,6,7,8,9,T,J,Q,K,A,2 where “2” is the strongest card and “3” is the weakest. In contrast, the suit “c”, “d”, “h”, “s” is not ordered.

The goal of this game is to get rid of all hands as fast as possible, and

- the fastest player is called “Daifugo”,
- the second fastest player is called “Fugo”,
- the third player is called “Heimin”,
- the fourth player is called “Himin” and
- the fifth, that is the latest, player is called “Daihinmin.”

The round ends when Daihinmin is decided. One round consists of some tricks and a trick is led by the player who has taken the previous trick. The first trick in the round is led by the player who has the card “3d.” The player who leads a trick can play arbitrarily and the trick is followed by plays. Every player sits along a playing order. The order is changed after some rounds end.

The “play” of this game can be classified into followings.

- Every single card is a “Single” play.
- Plural cards are one of “Pair”, “Triple” or “Quartet” play if all ranks of cards are the same.
- For example, a set of “4s” and “4d” is Pair and “Qc, Qh, Qs” is Triple.
- On the other hand, “7s, 8s” or “6c, 6d, 6h” are not a play.
- The play named “Kaidan” is constructed of more than three cards such that all cards have the same suit and ranks are successive.
- For example, “4h, 5h, 6h” is Kaidan with three cards, but both of “6c, 7s, 8c” and “8d, Td, Jd, Qd” are not Kaidan.

We call “Single”, “Pair”, “Triple”, “Quartet” or “Kaidan” with the number of cards “types of play.”

For example, the type of “7s” is Single, the type of “Ts Js Qs Ks” is Kaidan with four cards and the type of “3c, 4c, 5c” is Kaidan with three cards.

Every type of play has a specific order of strength as follows.

- On the type of Single, Pair, Triple or Quartet:
  The rank of the play represents the order.
  For example, on the type of Triple, “6s, 6c, 6h” is weaker than “7h, 7d, 7c” and it is stronger than “4h, 4d, 4s.” On the other hand, “6d, 6c, 6h” has the equal strength.
  For another example, “2h, 2d” is the strongest on the type of Pair.
- On the type of Kaidan:
  Let \( \alpha \) and \( \beta \) be plays whose type is Kaidan. If the highest rank in \( \alpha \) is lower than the lowest rank in \( \beta \) then \( \beta \) is stronger than \( \alpha \), or vice versa. If the condition is not satisfied then \( \alpha \) and \( \beta \) are incomparable.
  For example, “3d, 4d, 5d” is weaker than “6c, 7c, 8c” but “4s, 5s, 6s” is incomparable to both of them.
  Of course, “3d, 4d, 5d” and “6h, 7h, 8h, 9h” are not in the same type then these two plays are incomparable.

The player can select pass or play. Every play must be stronger than the last play in the trick. If the player has no stronger play then he must pass.

Once pass is selected, the player can not play until the trick has been ended. When all players pass, the trick is taken by whom the last play does.

From the shuffled card providing, this game is non-deterministic. This game is zero-sum because one player becomes Daifugo then other player can not be Daifugo. Thus, the game “Daihinmin” is multi-player, zero-sum, finite, non-deterministic, imperfect information game.

There are some special rules depend on play.

- Joker can be used as a substitution of any cards.
  The play “7d, 9d, Joker” is equivalent to “7d, 8d, 9d.” Therefore, “Ac, 2c, Joker” is the strongest Kaidan with three cards. The single Joker is the strongest Single but only “3s” can defeat it.
- “Shibari”:
  When succeeding two plays have corresponding suits, then the trick is in Shibari.
  If Shibari is initiated, all succeeding plays must also have corresponding suits. In other words, when Shibari begins then possible plays are restricted such that the suits is corresponding until the end of the trick.
  For example, assume that the last play is “6d” and next player plays “9d”, then the successive player can only play “Td”, “Jd”, “Qd”, “Kd”, “Ad”, “2d” or “joker.”
Assume that he plays “Ad” then the next player can only play “2d” or “joker.”

For another example, assume that the last play on the trick is “7s, 7d” and the next player plays “Qs, Kd”, then the successive player can only play “Ks, As, Ad” or “2s, 2d.”

- “8” (eight card):
  If a play contains a card whose rank is “8”, then the trick is terminated. Then a new trick is led by this player.

- “kakumei” (=revolution):
  Quartet or Kaidan with more than five cards can initiate the “kakumei” (which means “revolution”).
  When kakumei begins, the order of the rank becomes upside down. In the revolution, the rank “3” is the strongest and “2” is the weakest. Twice kakumei brings the order into normal.

Card exchange:
When a successive round starts, following card change is done after dealing the cards.
- Daihinmin (5th at the last round) must submit the best and the second best Single card to Daifugo (1st at the last round), and Daifugo must return arbitrary two cards to Daihinmin.
- Hinmin (4th at the last round) must submit the best Single card to Fugo (2nd at the last round), and Fugo must return arbitrary one card to Hinmin.

When a round has been ended, every player gets the following points:
- Daifugo : 5 points
- Fugo : 4 points
- Heimin : 3 points
- Hinmin : 2 points
- Daihinmin : 1 point

After plural rounds, players' ranking is decided by their total points.

3. UECda – The computer Daihinmin competition

In the University of Electro-Communications, Japan, computer Daihinmin competition is held every year[11]. The competition has two classes. One is “light class” whose program is restricted to heuristic or some light algorithms. The other is “unlimited class” in which any algorithm is allowed. The competition sets the following environment.

- The number of player is just 5.
- After 100 rounds are played, players' playing order (= sitting position) is changed randomly.
- Total 1000 or 4000 rounds are done and the total of points decides the players' rank.

In UECda-2014, we won the “light class” with the program named “kou” and we introduce our heuristics in this paper.

4. Heuristics

4.1 Algorithm overview

The following is the overview of the algorithm of “kou” which is the winner of UECda-2014 light class.

1. Finding better combination of plays in the hand such that
   - each of plays has no overlapping and
   - the number of plays is nearly minimum.
2. Select the play which has the highest evaluation value to end the trick. We define the evaluation function for this selection. This function includes some heuristics and special rules to make a priority on plays.
3. If such play is not found, select the play such that “weakness of the hand” is not increased. This means that remaining hand is stronger than the present hand. Usually, the weakest will be played with this criterion.

The combination of plays in hand is made by the following algorithm.

1. Find all Kaidan whose cards are not any member of Pairs, Triples or Quartets.
2. Let the strongest card and “8” be Single.
3. Find Quartets, Triples and Pairs from the rest of the hand.
4. At last, all the rest cards are Singles.

This algorithm divides player's hand into plays which are not overlapping.

Heuristics used in the evaluation function are as follows.

4.2 Evaluation to end the trick

The evaluation value is found for each play in the hand. If this value is higher than that of other plays, we think the play is stronger and it tends to end the trick. The evaluation value for a play is calculated by the followings.

1. If the play can end the trick, i.e. there is no possible play which is stronger than the play, then the evaluation value is 100. If the move contains “8” then it is 101.
2. Joker's value is 100 if “3s” is already played, otherwise it is 1.
3. If the play is Kaidan,

   \[ 100 - (\text{possible Kaidan plays}) \times (\text{finished player} + 1) \]
is the evaluation value.
Here, (possible Kaidan plays) is the number of possible Kaidan which can be played by enemies.
(finished player) is the number of players who have finished this round.

3. If there are more than 3 players remaining and the play is Pair, Triple or Quartet then

\[ 100 - \sum_{i=1}^{m} f_i(n) \]

is the evaluation value.
Here, \( m \) is the number of possible stronger plays which is in enemies' hand.
Each \( i \) means a play which is in enemies' hand.
\( f_i(n) \) is found by the following.
Let \( n = (\text{the number of stronger cards}) - (\text{the number of cards of my play}) + (\text{finished player}) \).
Then, it is defined as follows.
\[
\begin{align*}
  f_i(n) &= \begin{cases} 
    4 & (n \leq 0) \\
    9 & (n = 1) \\
    15 & (n = 2) \\
    24 & (n \geq 3)
  \end{cases}
\end{align*}
\]
This means stronger play \( i \) takes smaller value of \( f_i() \), then the evaluation value will be bigger.

4. Otherwise, i.e. if the move is Single, Pair, Triple or Quartet,

\[ 100 - ((\text{stronger total}) \times 30) \]

is the evaluation value.
Here, (stronger total) is the sum of ranks of plays which are in enemies' hands.

5. If these values are less than or equal to 0, let it 1.

If this value of the play is equal to or greater than 95, then we think the play will end the trick. We call this value “strength” of the play.
Let \( m \) be a play, then the strength of \( m \) is denoted by \( S_m \).

4.3 “Shibari” priority

When the player can initiate Shibari by the play \( m \), the strength of such play \( S_m \) is modified to upper limit if the following condition holds.

- After this Shibari, the player has the strongest move.

If this condition does not hold, \( m \) will not be played.

4.4 Trick leading play

We define play priority for every play \( m \) to lead the trick.
The play priority \( p_m \) for a play \( m \) is calculated by the following.

- If \( m \) is Kaidan and contains “8” then \( p_m = 9 \).
- If \( m \) is Joker then \( p_m = 3 \).
- If \( 1 \leq s_m \leq 94 \) then

\[ p_m = 20 + 2(\text{rank of } m). \]
Here, (rank of \( m \)) is 1 if play ”m” is the weakest and
13 if play “m” is the strongest among plays whose types are the same.
For example, a Kaidan “4s 5s 6s” has the (rank of \( m \)) of 2 because there exists only weaker play “3s 4s 5s.”
- If \( 95 \leq s_m \) then \( p_m = 110 - s_m \).

If \( p_m \) is bigger, the play “\( m \)” has high priority. If there are some plays whose priority are the same, then the best play is selected from the following order :
Kaidan, Triple, Pair and Single.

This value is also used to play when there is no play to end the trick. In such case, combining the following hand weakness, the best play will be selected.

4.5 Strong play reservation

We define the hand weakness. The hand weakness is the sum of the following \( w_m \) for every play \( m \) in the hand.
\[
w_m = \begin{cases} 
  2 & (1 \leq s_m \leq 30) \\
  1 & (31 \leq s_m \leq 60) \\
  0 & (61 \leq s_m \leq 90) \\
  -1 & (91 \leq s_m)
\end{cases}
\]
This value means how the remaining hand is weak. If there is the play \( m \) such that \( s_m \) is high but the hand weakness is also high, then \( m \) will not be played. This is balancing procedure to avoid wasting strong moves.

5. Evaluation experiments

5.1 Comparing with past champions

The following is the list of past light class champions.
“chibiHana” is the C version of Kishimen_2013 which is the champion of the light class in UECda-2013. This program makes combination of plays whose number is the minimum. The finish search of this program is sophisticated and Shibari strategy is also implemented.

“Party” is the champion of the light class in UECda-2012. This program tends to have plays which can end the trick. Joker will be used to play weaker cards. Shibari is also considered to end the trick.

Comparing with these two programs, we set the match among the following five programs.

- kou
- chibiHana
- Party
- default
- default

Here, default is the basic strategy program which plays the weakest in the hand. Table 1 is the scores of this setting.

Table 1 : Results among light class champions

<table>
<thead>
<tr>
<th></th>
<th>1000</th>
<th>4000</th>
<th>7000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>kou</td>
<td>3660</td>
<td>14954</td>
<td>26178</td>
<td>37413</td>
</tr>
<tr>
<td>chibiHana</td>
<td>3493</td>
<td>14236</td>
<td>24817</td>
<td>35460</td>
</tr>
<tr>
<td>Party</td>
<td>3271</td>
<td>12698</td>
<td>22208</td>
<td>31536</td>
</tr>
<tr>
<td>default</td>
<td>2324</td>
<td>9198</td>
<td>16017</td>
<td>22900</td>
</tr>
<tr>
<td>default</td>
<td>2252</td>
<td>9014</td>
<td>15780</td>
<td>22691</td>
</tr>
</tbody>
</table>

From this result, “kou” is the strongest heuristic algorithm comparing with past two years champions. Especially, “kou” marks high scores at all rounds from 1000 to 10000.

5.2 Comparing with Monte-Carlo algorithms

Monte-Carlo algorithm is also effective to Daihinmin. In recent years, all champions at “unlimited class” are made with Monte-Carlo simulation. In Daihinmin, we can not make a search tree because of imperfectness. Thus, all programs have the depth 1 tree for Monte-Carlo simulations. In addition, opponents’ hands are simulated by random card delivering. The following is the list of famous Monte-Carlo Daihinmin programs.

- “beersong” is the champion program at UECda-2013.
- “paoon” is the champion program at UECda-2012.
- “crow” is the champion program at UECda-2011[6].
- “snowl” is the champion program at UECda-2010[9].

Table 2 is the scores of these four programs and “kou.”

Table 2 : Results against Monte-Carlo programs

<table>
<thead>
<tr>
<th></th>
<th>100</th>
<th>300</th>
<th>500</th>
<th>700</th>
<th>900</th>
</tr>
</thead>
<tbody>
<tr>
<td>kou</td>
<td>344</td>
<td>920</td>
<td>1481</td>
<td>2021</td>
<td>2528</td>
</tr>
<tr>
<td>beersong</td>
<td>335</td>
<td>999</td>
<td>1541</td>
<td>2213</td>
<td>2853</td>
</tr>
<tr>
<td>paoon</td>
<td>299</td>
<td>891</td>
<td>1570</td>
<td>2209</td>
<td>2893</td>
</tr>
<tr>
<td>crow</td>
<td>246</td>
<td>851</td>
<td>1451</td>
<td>2068</td>
<td>2678</td>
</tr>
<tr>
<td>snowl</td>
<td>276</td>
<td>839</td>
<td>1457</td>
<td>1989</td>
<td>2548</td>
</tr>
</tbody>
</table>

Figure 1 : Scores among light class champions

Figure 2 : Scores against Monte-Carlo programs
From this result, "kou" is as strong as "snowl" which is the early Monte-Carlo algorithm client of Daihinmin.

Time complexity of "kou" is very low comparing with any Monte-Carlo programs. We don't have precise data but kou's calculation time is about hundred times faster than other Monte-Carlo programs. Thus, we can say that kou marks the highest scores per second.

**5.3 Heuristics effect**

To evaluate the heuristics used in "kou", we have made some modified players to compare with the original "kou."

The modified player is as follows.

1. **Evaluation to end the trick:**
   - The original "kou" uses 95 as the threshold value to decide that the play can end the trick or not. So, we prepare
     - w100,
     - w90 and
     - w80
   - whose thresholds are 100, 90 and 80, respectively.

2. **Shibari priority:**
   - We prepare programs
     - lock+ and
     - lock-.  Here, lock+ plays Shibari if it possible and lock- never plays Shibari.

3. **Trick leading play:**
   - Two programs are prepared which are
     - weak and
     - single.
   - At the leading of the trick, weak always plays the weakest play in the hand. On the other hand, single always plays the weakest Single.

4. **Strong play reservation:**
   - The program
     - use_2 : which does not use “hand weakness” value and “Strong play reservation” heuristic.

Every modified program and the original “kou” are independently matched with the four same enemies. Here, all enemies are the same program which is one of chibiHana, Party or default. For example, we set a test match with one w100 and four chibiHana.

In the followings, all experiments are evaluated by 1000 rounds.

**5.3.1 Evaluation to end the trick**

Table 3 shows the score difference between the modified program and the original program.

<table>
<thead>
<tr>
<th>modified program</th>
<th>w100</th>
<th>w90</th>
<th>w80</th>
</tr>
</thead>
<tbody>
<tr>
<td>opponent</td>
<td>score diff.</td>
<td>score diff.</td>
<td>score diff.</td>
</tr>
<tr>
<td>chibiHana</td>
<td>-2132</td>
<td>+403</td>
<td>+210</td>
</tr>
<tr>
<td>Party</td>
<td>-767</td>
<td>+663</td>
<td>+970</td>
</tr>
<tr>
<td>default</td>
<td>-825</td>
<td>+15</td>
<td>+121</td>
</tr>
</tbody>
</table>

The program w100 is weaker than the original from this result. On the other hand, w90 and w80 are stronger than the original, thus there will be more optimal threshold value under 95. For chibiHana, the threshold may be between 90 and 95, but it will be under 80 for Party and default. More detailed experiments and analysis are needed to decide the threshold value. Now, we can conclude that 95 is not bad value and this heuristic is effective for various enemies.

**5.3.2 Shibari priority**

Table 4 shows the score difference between lock+ or lock- and the original program.

<table>
<thead>
<tr>
<th>modified program</th>
<th>lock-</th>
<th>lock+</th>
</tr>
</thead>
<tbody>
<tr>
<td>opponent</td>
<td>score diff.</td>
<td>score diff.</td>
</tr>
<tr>
<td>chibiHana</td>
<td>-1211</td>
<td>-720</td>
</tr>
<tr>
<td>Party</td>
<td>-117</td>
<td>-287</td>
</tr>
<tr>
<td>default</td>
<td>-757</td>
<td>-203</td>
</tr>
</tbody>
</table>

From this result, the score of lock- is 1211 less than that of the original. Every difference is minus then we can conclude that both of lock+ and lock- are weaker than the original “kou.” Especially, the difference against chibiHana is bigger than the others. This is caused that chibiHana's “end the trick” strategy is more effective than Party and default.

**5.3.3 Trick leading play**

Table 5 shows the score difference about weak and single.

<table>
<thead>
<tr>
<th>modified program</th>
<th>weak</th>
<th>single</th>
</tr>
</thead>
<tbody>
<tr>
<td>opponent</td>
<td>score diff.</td>
<td>score diff.</td>
</tr>
<tr>
<td>chibiHana</td>
<td>-165</td>
<td>-1086</td>
</tr>
<tr>
<td>Party</td>
<td>-679</td>
<td>-376</td>
</tr>
<tr>
<td>default</td>
<td>-856</td>
<td>-895</td>
</tr>
</tbody>
</table>

Both modified program weak and single become weaker than the original.

Thus we can say that the trick leading heuristic is also effective. For chibiHana, single’s score is lower than over
1000 points. We can say that types of plural cards are more important against chibiHana. On the other hand, weak’s score is lower than single against Party. It implies that selection of the trick leading play is effective against Party. There are not difference between the score of weak and single against default. This is from that default also leads the trick by single and the weakest play.

5.3.4 Strong play reservation

Table 6 shows the score difference about use_2.

<table>
<thead>
<tr>
<th>opponent</th>
<th>use_2 score diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>chibiHana</td>
<td>-157</td>
</tr>
<tr>
<td>Party</td>
<td>+100</td>
</tr>
<tr>
<td>default</td>
<td>+102</td>
</tr>
</tbody>
</table>

Evaluation of this heuristic's effectiveness needs careful discussion. For chibiHana, this heuristic is effective. But it is not effective for the other two enemies. Obviously, default is the weakest among chibiHana, Party and default. Thus, strong play reserving is effective for the enemies which has complicated algorithm. For a simple enemy, simple play will be effective rather than reserving strong plays.

6. Conclusions

In this paper, we introduce the heuristics of “kou” which is the champion program at UECda-2014 light class. The main heuristics are the following four point of view.

- Evaluation to end the trick
- Shibari priority
- Trick leading play
- Strong play reservation

All of these heuristics are effective but Strong play reservation is not effective for some simple algorithm enemies. Moreover, there is possibility that the threshold value or some parameters can be sophisticated by optimization technique or machine learning method.

Now, such parameters are provided by hard coding in the program and the values are also selected by heuristic method. When we optimize these values then our program will become more stronger. This is remained for a future study.

In addition, we have shown the strongness of “kou” against past champions and Monte-Carlo algorithms. Especially, “kou” is little stronger than “snow” which is the champion program at UECda-2010 and it is famous as the turning point that Monte-Carlo algorithm is useful for Daihinmin.

It is known that Monte-Carlo algorithm takes much time to play. By our brief counting, heuristic algorithms are one hundred times faster than Monte-Carlo algorithms. Thus, We can show a new possibility of Daihinmin player by heuristic algorithm.

References

**Yasuhiro Tajima** received his Ph.D. from The University of Electro-Communications in 2001. He had joined Ishikawajima-Harima Heavy Industries Co., Ltd., and Tokyo University of Agriculture and Technology. Currently, he is an associate professor of Okayama Prefectural University. His research interests are in Machine Learning and Text Mining. He is a member of IEICE, JSAI and IPSJ.

**Kouzou Tagashira** has received his B.E. degree from Okayama Prefectural University in 2015. Currently he is a student of Graduate School of Computer Science and Systems Engineering, Okayama Prefectural University. He is the main programmer of the Daihinmin program called “Kou” and won the championship in light class at UECda-2014.

**Genichiro Kikui** his Ph.D. degree in informatics from Kyoto University in 2007, respectively. He joined the Nippon Telegraph and Telephone Corporation (NTT) in 1986, and he was a visiting researcher at the Center for the Study of Language and Information (CSLI) at Stanford University in 1997. He has been a professor with Okayama Prefectural University in Okayama, Japan since 2011. He is interested in natural language understanding.
Analysis Method Focusing on Peaks for Spatial Evaluation of Impression

Shunsuke Akai¹, Teruhisa Hochin² and Hiroki Nomiya³
Kyoto Institute of Technology, Japan

Abstract
This paper proposes an analysis method of the evaluation results obtained through the Impression Evaluation Method by Space (IEMS). The IEMS uses a plane containing impression words as the Kansei space. The impression of an object is specified by circling the areas matching the impression. The degree of matching the impression is expressed by painting color. As the impression words can be moved and/or added in the IEMS, it is difficult to analyze the evaluation results obtained from many subjects. The proposed analysis method focuses on the peaks of darkness. It is called the analysis method focusing on the peaks of darkness (abbr. AM_PD). By mapping the peaks of the darkness in each evaluation result to the same Kansei space, this method can analyze characteristic impressions. This paper proposes an algorithm of extracting obvious peaks, and a method of mapping peaks in an individual Kansei space to the baseline Kansei space, which uses the spatial relationships of impression words in Kansei space. This paper shows the soundness of the proposed method by presenting the results of the mapping of peaks.

Keywords: Analysis method, Impression evaluation, Kansei, Peaks, Mapping.

1. Introduction

In recent years, in addition to functions and convenience, Kansei topics such as design have become important. Kansei is a word that means how people feel. Kansei topics include design of product [1], retrieval of data [2], usage of emotion [3, 4], user adaptation [5], and impression evaluation [6-9]. Because Kansei is vague, it is difficult to precisely capture and quantify it. Moreover, because Kansei differs for each person, it is difficult to evaluate.

The semantic differential (SD) method [10] is often used as an evaluation method of such Kansei. As the SD method digitizes impressions, it enables statistical processing and makes it possible to perform various analyses. However, to enable statistical processing, the evaluation is required to be performed in a predefined range. As a result, it is difficult to evaluate vague aspects of Kansei. A method enabling the evaluation of the vagueness of Kansei is required. Although various research efforts concerning the expression of Kansei have been conducted [1, 6, 7], such an evaluation method has not yet been established. An impression evaluation method considering the vagueness of Kansei has been proposed in order to overcome this issue [8, 9]. The proposed method uses a plane containing impression words. The impression of an object is specified by circling the areas matching the impression. The degree of matching of the impression is expressed by the painting color. This method is called the Impression Evaluation Method by Space (IEMS). The IEMS provides to users a commonly used baseline Kansei space. Users can modify this baseline space as needed.

Two analysis methods for the evaluation results of the IEMS have also been proposed. One focuses on the baseline Kansei space [11, 12]. It spatially shows average values and coefficients of variation of scores of the evaluation results. By using this method, the characteristics of the impression of objects and the dispersion among subjects could easily be obtained. This method is called the analysis method focusing on the baseline Kansei space (abbr, AM_BKS).

The other method focuses on the impression words [13, 14]. The numbers of the impression words circled are spatially displayed. This method enables us to visually capture the tendency of impression of objects because impression words having similar impression are placed nearer than those having different impression in the Kansei space. This method is called the analysis method focusing on the impression words (abbr, AM_IW).

The AM_BKS, however, cannot be used when the impression words in the Kansei space are moved. As the impression words can be moved and/or added in the IEMS, this restriction is very serious. On the other hand, the AM_IW cannot consider the darkness of a color.

This paper proposes another new analysis method that can consider the darkness of a color, and can be used even when the impression words in the Kansei space are moved. This method focuses on the peaks of darkness. By mapping peaks of the darkness in each evaluation result to the same Kansei space, this method can analyze characteristic impressions. First, the obvious peaks of the darkness are automatically extracted from all evaluation results [15]. Next, those peaks are mapped to the baseline Kansei space by using the spatial relationships of impression words in

¹ Department of Information Science
Matsugasaki, Sakyo-ku, Kyoto 606-8585, Japan
Currently with Kanden System Solutions Co., Inc.
akai-shunsuke@js-sol.jp

² Information and Human Sciences
Matsugasaki, Sakyo-ku, Kyoto 606-8585, Japan
hochin@kit.ac.jp

³ Information and Human Sciences
Matsugasaki, Sakyo-ku, Kyoto 606-8585, Japan
nomiya@kit.ac.jp
Kansei space [16]. Soundness of the proposed method is shown by presenting the results of the mapping of peaks.

The remainder of the paper is organized as follows. Section 2 shows the evaluation method of Kansei called IEMS, and the analysis methods already proposed for IEMS, AM_BKS and AM_IW. Section 3 proposes a new analysis method AM_PD. Section 4 proposes an extraction method of peaks. Section 5 proposes a method of mapping peaks in individual Kansei space to the baseline Kansei space. Section 6 shows several analysis results of AM_PD. Section 7 gives some considerations. Finally, Section 8 concludes the paper.

2. Impression Evaluation Method by Space

2.1 Kansei Space

The impression evaluation method by space (IEMS) uses a Kansei space. The Kansei space is the space imagined in evaluating the impression of an object. For example, when the impression of a landscape is evaluated, it can be expressed with words such as “beautiful.” In addition, the degree of fitness to an impression word can also be expressed. It is believed that people have some impression expression items in their minds, and they compute the degree of fitness of each item to the impression of an object.

Impression words are usually used as impression expression items because words can easily express an impression (something considered by the authors to be very important) [8, 9]. In the Kansei space, the more similar the impressions of two impression words are, the closer these impression words are. It is thought that the Kansei space changes according to experience and learning. The Kansei space depends on the person.

2.2 Summary of IEMS

The IEMS uses a plane containing impression words as the Kansei space. The impression of an object is specified by circling the areas matching the impression. The degree of matching the impression is expressed by the painting color. In other words, the more closely the impression is matched, the darker the color that is used. The color gray is used for these areas. A special brush, which makes painted areas dark gradually, is used.

As it is difficult to create the Kansei space mentioned above from scratch, the IEMS provides to users a commonly used baseline Kansei space. Users can modify this baseline space as needed.

The baseline Kansei space has been obtained by applying multi-dimensional scaling [17] to the results of a questionnaire about the dissimilarity between impression words [8, 9]. It is shown in Fig. 1. The impression words of the baseline Kansei space are called baseline impression words.

2.3 Evaluation rules used in the IEMS

The evaluation rules used in the IEMS are as follows:

- The impression words selected to express the impression of an object are circled.
- One or more impression words can be circled in an area.
- One or more circled areas can exist.
- If the space does not have the desired impression words, the impression words can be added.
- If the impression points-of-view are different, the circled areas are different. For example, when we see a starry sky, some people feel bright and dark. They can evaluate this impression by circling the area near the word “bright” and circling another area near the word “dark.”
- If a user does not agree with the position of impression words, the impression words can be moved.
- Two or more Kansei spaces can be used when users cannot sufficiently evaluate impressions through one Kansei space. In other words, when users want to evaluate different impression points-of-view by using the same impression word, or when the areas overlap, users can evaluate an impression through two or more Kansei spaces.

2.4 Impression Evaluation System

The impression evaluation system based on the IEMS has been implemented. This system enables subjects to evaluate their impression of a picture. The initial screen of this system is shown in Fig. 2. The Kansei space is presented in the center of the screen, while a picture is presented on the right side. Pictures are presented in a random order. A subject evaluates his or her impression of a picture in the space by using the buttons and text boxes available in the left and bottom sides of the screen.

In this system, the height and the width of the Kansei space are 600 pixels. Each pixel has 256 scales as the degree of darkness (0 shows white and 255 shows black).
2.5 Impression Evaluation Experiment

An impression evaluation experiment with five pictures was conducted using IEMS [8, 9]. Five pictures used in this experiment are presented in Fig. 3 [18-21]. The order of presenting pictures was random. Japanese words were used in the experiment. Twelve subjects participated in this experiment.

2.6 Conventional Analysis Methods

1) Analysis method focusing on the baseline Kansei space (AM_BKS)

It is difficult to compare evaluation results because the positions of impression words may be different. Here, we consider the situation that objects are evaluated by using the Kansei space, where impression words are not moved. It means that the objects are evaluated through the same indicator. In this case, the average value and the coefficient of variation of the degrees painted by subjects at each pixel are used to obtain the tendency of the evaluation results [11, 12]. The baseline Kansei space and the one with additional impression words are of this kind of Kansei space.

As the average values and the coefficients of variation are calculated for all of the pixels in the baseline Kansei space, these could spatially be presented. The space whose pixels represent average values (coefficients of variation, respectively) is called an AVG (CV) space.

2) Analysis method focusing on impression words (AM_IW)

In the IEMS, the impression words expressing the impression of an object are circled. It is considered that the characteristics of the objects can be captured by investigating the impression words circled. The analysis method focusing on impression words counts the numbers of subjects circling the impression words, and displays the numbers near the impression words in the baseline Kansei space [13, 14]. This method enables us to visually capture the tendency of impression of objects because impression words having similar impression are placed nearer than those having different impression in the Kansei space.

3. Analysis Method Focusing on Peaks of Darkness

The AM_BKS cannot be used when the impression words in the Kansei space are moved. The AM_IW cannot consider the darkness of a color. The analysis method that can consider the darkness of a color and can be used even when the impression words are moved is required.

The meaning of the darkness of each pixel is different in each subject because the positions of impression words are different. So, it is difficult to consider the darkness of all pixels. More important places should be focused on.

It is thought that obviously dark places than surrounding ones are more important. The analysis method focusing on these places (AM_PD), which are called peaks, is proposed.

First, obvious peaks are extracted automatically from all evaluation results. However, the positions of the peaks cannot be compared directly because the positions of impression words may be different. So the peaks in each subject's Kansei space are mapped to the baseline Kansei space. This mapping is based on the relationships of impression words.

An example of the analysis by AM_PD is shown in Fig. 4 to Fig. 6. Fig. 4 shows an example of the evaluation result and the peaks of darkness, which are depicted with crosses. Fig. 5 shows the mapping result from the evaluation result.
shown in Fig. 4. Fig. 6 shows the mapping result from the evaluation results of several subjects.

For realizing AM_PD, the method for extracting the obvious peaks of the darkness, and the one for mapping peaks to the baseline Kansei space are required. Here, it is difficult to extract peaks automatically. When a fixed threshold value in obtaining peaks is used, false peaks are wrongly extracted, and some peaks cannot be extracted. This paper proposes the method of extracting peaks [15]. It is also difficult to map peaks of individual Kansei spaces to the baseline Kansei space because the linear transformation cannot always map peaks correctly. This paper also proposes the mapping method [16].

4. Extraction of Peaks

Extracting peaks automatically is desired. A peak consists of darker pixels than surrounding pixels. However, there is some possibility of extracting peaks that is unintentional for the subject. Decision concerning peaks depends on each person. Here, obvious peaks captured by many subjects are treated [15].

The algorithm of extracting peaks is presented. Next, the parameters are experimentally decided.

4.1 Algorithm of extracting peaks

The algorithm of extracting peaks is as follows:

1) Scale-down of evaluation result
   The evaluation result obtained by using the IEMS is expressed in the Kansei space. The height and the width of this space are 600 pixels. This space is transformed to a small Kansei space for reducing processing times and averaging darkness of pixels. The height and the width of a small Kansei space are 200 pixels. Pixels in the small Kansei space are average values of nine pixels, which are in the circled areas, in the Kansei space.

2) Investigation of pixels that is not an obvious peak
   For all pixels in the areas, the darkness of the center pixel is compared with the eight surrounding pixels (not considered outside the area). It is determined the center pixel is not a peak if the number of surrounding pixels darker than the center pixel is more than half of all the surrounding pixels. In addition, it is determined the pixel is not a peak if the darkness of that pixel is lighter than parameter X.

3) Create a list of pixels that may be a peak
   All pixels in the areas except for the pixels determined by 2) are sorted in descending order by the number of times of drawing pixels.
4) Select the pixel most likely as a peak from the list

Subject to the highest 10% pixels of the pixel list made by 3), the average of the darkness of the square centering on each pixel is calculated. There are three length types of the square. The length of "small," "middle," and "large" squares are five, nine, and fifteen pixels, respectively. First, it is determined the pixel that is the highest average of the darkness in a "small" square is a most likely peak. If not uniquely determined, the length of the square is changed to the one of "middle" and "large" squares. And, the same calculations are conducted. If not uniquely determined in a "large" square, the darkest pixel among them is selected as the most likely peak.

5) Determining whether the selected pixel is a peak

It is determined that the pixel selected at 4) is a peak if that pixel is surrounded by "very light" pixels than the first pixel and the lines of the area. However, if the "very dark" pixels than the first pixel or already checked pixels are included, that pixel is not determined as a peak. When the difference between the darkness of a pixel and that of the first one is larger than the threshold value Y, the pixel is decided to be "very light" or "very dark".

6) Judgment whether all peaks have been obtained

If all pixels in the list have been checked, then the procedure is terminated. Selected pixels as a peak are recovered by multiplying them by three. If not, checked pixels are removed from the list. And the procedure is repeated from 3).

4.2 Deciding parameters

The algorithm described above selects different peaks according to the parameter values. It is necessary to obtain obvious peaks captured by many subjects. A questionnaire survey on the peaks in the evaluation results is conducted. The values of parameters X and Y are determined based on the result of the questionnaire survey.

Five subjects participated in this questionnaire survey. Four subjects are males, and one is female. All subjects are university students from 22 to 28 years old.

Subjects mark a pixel if they feel that pixel is a peak. Ten evaluation results selected by one of the authors are used. In this paper, the pixels marked by four or more subjects are defined as the obvious peaks.

The precision rate and the recall one were used as indexes for determining the values of the parameters X and Y. The precision rate is obtained by dividing the number of the obvious peaks in the extracted ones by the number of the extracted one. The recall rate is obtained by dividing the number of the obvious peaks in the extracted ones by the number of all of the obvious peaks.

First, peaks are extracted under the condition that the value of the parameter X is the same as that of the parameter Y. We examine 30, 50, and 70 as the value of the parameters. Table 1 shows the average values of the precision and the recall rates of ten evaluation results for three cases.

When the precision rate is high, the recall rate is low. Conversely, when the recall rate is high, the precision rate is low. By analyzing the results of the questionnaire survey, it is estimated that light pixels are not selected as peaks.

From the questionnaire survey, it is thought that thin pixels are hard to become obvious peaks. One pixel not marked to anyone was extracted when parameter Y is low. So, we tried to use 150 and 40 as the values of the parameters X and Y, respectively, to extract peaks. The precision rate and the recall one reached to 93.04% and 100%, respectively.

An example of extracting the obvious peaks is shown in Fig. 7. Red cross marks show the extracted peaks. It is considered that the obvious peaks in the evaluation results can be extracted by using this algorithm.

5. Mapping of Peaks

The method of mapping peaks based on the spatial relationships of impression words is proposed. The method is based on the ratio of the distances of pairs of impression words [16].

5.1 Algorithm of Mapping Peaks

The algorithm of mapping a peak consists of two steps. The first is to obtain the precedence of impression words. The second is to determine the position of a peak in the baseline Kansei space.

1) Determining precedence of impression words

Based on the distance from a peak, the precedence of impression words is determined. Here, the impression words circled with a peak are considered to be more important than those outside of the circle. It is important to consider this aspect because there may be impression words that are outside of the circle, but whose distances to the peak are short. When only the distance is considered, the impression words not circled with a peak become more important than...
Figure 7. Example of extracting the obvious peaks.

those circled with the peak. Therefore, impression words are firstly divided into two subsets: a set of impression words circled with a peak, and a set of those not circled with the peak.

The algorithm determining precedence of impression words is as follows.

[Step 1] The impression words circled with a peak are obtained.

[Step 2] The precedence of the impression words obtained in Step 1 is determined by sorting them according to the distance from the peak. The nearest impression word has the highest priority. A list of the impression words, say $L_c$, is created. The impression words are ordered according to their priorities.

[Step 3] The precedence of the impression words not circled with a peak is similarly determined, and a list, say $L_{nc}$, is similarly created.

[Step 4] A list of the impression words $L$ is obtained by inserting the list $L_{nc}$ after the list $L_c$.

2) Determining the position of a peak in the baseline Kansei space

The point in the baseline Kansei space, which is the best for a peak, is decided. The point is decided such that the spatial relationship between impression words and the point in the baseline Kansei space is the same as that between impression words and the peak. To this end, a kind of dissimilarity, which is precisely described next, is used. As this dissimilarity depends on the position in the baseline Kansei space, it is represented with $DIS(x, y)$ for the point $(x, y)$ in the baseline Kansei space. The point $(x, y)$ having the smallest $DIS(x, y)$ is decided to be the position of a peak in the baseline Kansei space.

3) Spatial Dissimilarity

Let $L_z$ be the list having top $z$ impression words of the list $L$. The number $z$ is experimentally decided as described later.

First, we define the dissimilarity $D_{ind}^{ind}$ as the Euclidean distance between the positions of a peak and the $k$th impression word in $L_z$ in the individual Kansei space. Next, we introduce the following ratio $R_{ind}^{ind}$:

$$R_{ind}^{ind}_{ij} = D_{ind}^{ind}_{i} / (D_{ind}^{ind}_{i} + D_{ind}^{ind}_{j}).$$  \hspace{2cm} (1)

Next, we define $D_{base}^{base}(x, y)_k$ as the Euclidean distance between the point $(x, y)$ and the position of the $k$th impression word in $L_z$ in the baseline Kansei space.

Similar to $R_{ind}^{ind}_{ij}$, we introduce the ratio $R_{base}^{base}(x, y)_{ij}$ in the baseline Kansei space:

$$R_{base}^{base}(x, y)_{ij} = \frac{D_{base}^{base}(x, y)_i}{D_{base}^{base}(x, y)_i + D_{base}^{base}(x, y)_j}.\hspace{2cm} (2)$$

We define the difference $diff(x, y)_{ij}$ as follows:

$$diff(x, y)_{ij} = 0 \hspace{2cm} (i=j)$$

$$diff(x, y)_{ij} = |R_{ind}^{ind}_{ij} - R_{base}^{base}(x, y)_{ij}| \ast W_{cl} \ast W_{co} \hspace{2cm} (i \neq j)$$  \hspace{2cm} (4)

Here, $W_{cl}$ and $W_{co}$ are the weights of the closeness and that of the coexistence, respectively. It is considered that the impression words close to the peak are important. The weight $W_{cl}$ is increased according to the closeness to the peak. Whether an impression word is in the area where a peak exists, or not is also considered to be important. This is because the impression a user felt includes the impression represented by the word and the peak when the impression word and the peak are in the area representing the impression the user felt. These weights are experimentally decided as described later.

The difference of the point $(x, y)$ and the peak, which is denoted as $DIS(x, y)$, is defined by

$$DIS(x, y) = \frac{\sum_{i=1}^{I} \sum_{j=1}^{I} diff(x, y)_{ij}}{2}.\hspace{2cm} (5)$$

5.2 Deciding Parameter Values

Here, the values of the parameters are experimentally decided. The parameter values tried to be decided are the number of impression words used in deciding the position of a peak, the weight of closeness of a peak to an impression word, and the weight of coexistence of a peak and impression words.

1) Number of the impression words used

The total number of impression words is thirty-seven. Using all impression words is time-consuming, and is not effective. We try to use ten and twenty impression words, which are approximately a quarter and a half of impression words, respectively.
2) Weight of closeness $W_{cl}$

This is the weight for the closeness of a peak to an impression word. Let an index $i$ be $ord_i-1$, where $ord_i$ is the order of the $i$th impression word in the list $L_z$ of the impression words used. The equations used in calculating the weight for the $i$th and $j$th impression words in the list $L_z$ are:

$$ (5 - 0.25 \times index_i) \times (5 - 0.25 \times index_j) $$

or

$$ (5 - 3 \log_{10}(index_i + 1)) \times (5 - 3 \log_{10}(index_j + 1)) $$

Equations (6) and (7) are referred to as “normal” and “log,” respectively.

3) Weight of coexistence $W_{co}$

Whether an impression word is in the area where a peak exists, or not is considered to be important. This is because the impression a user felt includes the impression represented by the word and the peak when the impression word and the peak are in the area representing the impression the user felt.

Let $S$ be the area where a peak exists. For the $i$th and $j$th impression words in the list $L_z$, the weight $W_{co}$ is one when both words are not in the area $S$. The weight $W_{co}$ is two when one word is in $S$, but the other is not in $S$. When both words are in $S$, $W_{co}$ is four or eight.

4) Experiment

The best combination of the values of the three parameters is experimentally obtained. To this end, all of the combinations as shown in Table 2 are used in mapping peaks.

The peaks of three evaluation results are tried to be mapped. These evaluation results are shown in Fig. 8. Japanese words were used in the experiment. The evaluation result shown in Fig. 8(a) is a typical one. That shown in Fig. 8(b) is for the test of impression words inside and outside the area where a peak exists. Two peaks are in the same area in the evaluation result shown in Fig. 8(c).

Better combinations obtained are shown in Table 3. Precise mapping results, which are included in our previous study [16], are omitted because of space limitation. As only $C7$ commonly appears in all of the mapping results, the combination $C7$ is considered to be the best. The combination $C7$, which is the combination of the number of words used is 20, $W_{cl}$ is “log,” and $W_{co}$ is 4, is adopted.

<table>
<thead>
<tr>
<th>No.</th>
<th>Number of words used</th>
<th>$W_{cl}$</th>
<th>$W_{co}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>10</td>
<td>normal</td>
<td>4</td>
</tr>
<tr>
<td>C2</td>
<td>10</td>
<td>normal</td>
<td>8</td>
</tr>
<tr>
<td>C3</td>
<td>10</td>
<td>log</td>
<td>4</td>
</tr>
<tr>
<td>C4</td>
<td>10</td>
<td>log</td>
<td>8</td>
</tr>
<tr>
<td>C5</td>
<td>20</td>
<td>normal</td>
<td>4</td>
</tr>
<tr>
<td>C6</td>
<td>20</td>
<td>normal</td>
<td>8</td>
</tr>
<tr>
<td>C7</td>
<td>20</td>
<td>log</td>
<td>4</td>
</tr>
<tr>
<td>C8</td>
<td>20</td>
<td>log</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 3: Better Combinations

<table>
<thead>
<tr>
<th>Target</th>
<th>Better combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 8(a)</td>
<td>C1, C5, C6, C7</td>
</tr>
<tr>
<td>Fig. 8(b)</td>
<td>C5, C6, C7</td>
</tr>
<tr>
<td>Fig. 8(c)</td>
<td>C1, C2, C3, C4, C7, C8</td>
</tr>
</tbody>
</table>

Figure 8. Evaluation results whose peaks are tried to be mapped.
6. Analysis Examples

We try to map the peaks of the pictures shown in Fig. 3 to the baseline Kansei space. The results of mapping shown in Figs. 9 to 13 are for Picture (a) to (e) shown in Fig. 3, respectively. The peaks mapped are shown in the crosses. The impression words connected to a peak are those circled with the peak.

In Fig. 9, there are many peaks around “natural,” “quiet,” “beautiful,” and “refreshing.” Many subjects felt these impressions to Picture (a) shown in Fig. 8(a). Almost all of lines exist between “quiet,” and “stylish.” Therefore, many subjects felt the impressions in the area between “quiet,” and “stylish,” especially “natural,” “quiet,” “beautiful,” and “refreshing.”

For Picture (b) shown in Fig. 8(b), many peaks appear in the right side and the bottom of the baseline Kansei space. These include the impression words “small,” “weak,” and “cute.”
There are many peaks around “sharp” and “strong” for Picture (c) shown in Fig. 8(c). Many subjects felt these impressions and their mixtures.

For Picture (d) shown in Fig. 8(d), there are fewer peaks than the other pictures. A few impression words including “mysterious,” “beautiful,” and “flashy” are selected. Many subjects felt these impressions. As there are few lines between these impression words, it is considered that there are little relationships among these impression words.

There are many peaks in the middle, the bottom, and the right upper areas in the baseline Kansei space for Picture (e) shown in Fig. 8(e). The middle (bottom, and right upper, respectively) area includes “mysterious” and “natural” (“beautiful” and “excellent,” and “sad”). This picture seems to include these three aspects.

7. Considerations

The method for mapping peaks enables us to evaluate the individual impressions, which subjects felt, in the common space, i.e., the baseline Kansei space. We could compare them, which cannot be compared in the individual Kansei spaces as they are. As we can analyze the evaluation results of IEMS, it is considered that the availability of IEMS is improved.

In many cases, many peaks exist around the positions of the impression words in the baseline Kansei space. This enables us to evaluate the representative impressions.

It was confirmed that some peaks exist in the area between the words “sharp” and “strong” for Picture (c). This means that some subjects felt the impressions which cannot be expressed by using a single impression word. For Picture (c), there are some peaks around the impression word “dark.” These peaks exist in a scattered manner rather than a concentrated one. The impressions that are mainly dark differ from one another. These relate to the impressions “sad,” “heavy,” and “cold.”

There are two peaks on the edges of the baseline Kansei Space in the analysis result shown in Fig. 12. It is considered that there may be no point in the baseline Kansei space for the impression that each peak represents. We can catch this point as the characteristic impression. This will help us analyze the impressions of the target evaluated.

8. Conclusion

This paper proposed an analysis method focusing on the peaks of darkness (abbr. AM_PD) for the evaluation results obtained through the Impression Evaluation Method by Space (IEMS). AM_PD enables us to analyze characteristic impressions by using the AM_PD because peaks are mapped to the baseline Kansei space, and can be compared with each other. In this paper, the method of obtaining peaks, and that of mapping peaks in an individual Kansei space to the baseline Kansei space were proposed. The mapping method proposed in this paper used the spatial relationships of impression words in Kansei space. The soundness of the proposed method was shown by presenting examples of the mapping of peaks. The tendency of the impressions that many users feel can easily be captured. That of the impressions that a few users feel can also be captured because the peaks of such impressions appear as isolated points in the baseline Kansei space. It was also shown that the peaks between the impression words were successfully mapped. This is effective in representing the nature of the vagueness of Kansei.

Applying AM_PD to other impression evaluation results is in future work. Three analysis methods have been proposed for the IEMS. Comparison of these three analysis methods is also in future work.

References


Shunsuke Akai received his ME degree in Information Science from Kyoto Institute of Technology. His research focus is Kansei engineering, especially impression evaluation and vagueness of Kansei.

Teruhisa Hochin is a professor on Information and Human Sciences at Kyoto Institute of Technology. His research interests include multimedia Kansei retrieval and multimedia databases. He is a member of ACIS, IEEE, ACM, Institute of Electronics, Information and Communication Engineers (IEICE), Information Processing Society of Japan (IPSJ), Database Society of Japan (DBSJ), Japan Society for Archeological Information (JSAI), and Japan Society of Kansei Engineering (JSKE). He has given presentations at national and international conferences, symposiums, and workshops. He has reviewed papers for IJCIS, ISI, IJNDC, Kansei Engineering International Journal, and so on.

Hiroki Nomiya is an assistant professor in the department of Information Science at Kyoto Institute of Technology. His research focus is multimedia data engineering, image recognition and machine learning. He has been engaged in the research of multimedia data retrieval, especially emotional video scene retrieval using facial expression recognition based on machine learning. He has given presentations at national and international conferences and workshops. He is an active member of the Information Processing Society of Japan and the Japanese Society for Artificial Intelligence. He has reviewed papers for IEICE (the Institute of Electronics, Information and Communication Engineers) Transactions on Information and Systems.
Estimating Prediction Interval of Cumulative Number of Software Faults Using Back Propagation Algorithm
Begum Momotaz* and Tadashi Dohi
Hiroshima University, Japan

Abstract

In this paper, we apply the well-known back propagation algorithm for feedforward neural network architectures and the delta method to construct the prediction intervals of cumulative number of software faults, where the underlying fault count data are governed by the Poisson law. To deal with the Poisson count data in the neural network computing, we use three data-transform methods for preprocessing the Poisson data; Bartlett transform, Anscombe transform and Fisz transform. In numerical experiments with eight real software development project data sets, we evaluate the one-stage look-ahead prediction interval in sequential software testing, and compare these data transform methods in terms of average relative error, coverage rate and prediction interval width.

Keywords: software reliability, fault prediction, prediction interval, back propagation algorithm, delta method.

1. Introduction

Our modern society depends on computer systems, so that there are many different applications, such as nuclear reactors, financial trading systems and medical systems, etc. Software controls our daily activities of computer systems. Failure will occur even when software is used. Nowadays, predicting software faults is an applicable issue and a major anxiety for software developers and engineers. In software development project, most of software engineers set goals to provide high quality software. Especially reliability of software plays a vital role in software development process, because software reliability is one of the most fundamental but significant quality attributes [6][31][32][34][43]. Many quantitative methods such as reliability modeling, development cost prediction, release-decision analysis and fault-tolerance, have been developed in software engineering [14][15][20][21][42] and been based on software reliability growth models (SRGMs).

The major drawbacks of SRGMs and artificial network models are that their goodness-of-fit performance and the prediction performance strongly depend on the data, so it is always needed to select the best SRGM by estimating the latent number of faults in each software product. In general, SRGMs can be classified into different several classes, such as error seeding models, failure rate models, and nonhomogeneous Poisson process (NHPP) models [6][31][32][34][43].

On the other hand, there are two classes of SRGMs; analytical SRGMs and data-driven artificial neural network models. However, none of these models satisfy the requirement levels of software developers [38]. Furthermore prediction of software faults is an important measurement to find reliable software during the software operational phase. Sometimes professional managers use any predictor for software quality after release. Aabaei et al [2] explained software fault prediction based on different machine learning techniques such as decision trees, decision tables, random forest, neural network, naïve Bayes and distinctive artificial immune systems classifiers. They made a conclusion that random forest outperforms the other methods.

In recent years, artificial neural network approaches have proven to be a universal approximation for arbitrary non-linear continues function with arbitrary accuracy [7][8][10][11][18][19][26][27][28][29][30][36][40]. Besides, artificial neural network models need to specify the network architecture, but do not unrealistic assumptions with respect to software failure data and tend to be superior to the classical SRGMs due to their freedom on model selection. They have been becoming an alternative to the traditional prediction method based on SRGMs [16]. Specifically, feedforward backpropagation networks have shown their advantages over analytical models in prediction on the number of software faults.

Artificial neural network is a collection of several processing nodes called artificial neurons. These neurons are designed on the basis of study of the behavior of biological neuron. Basically it is good at modeling nonlinear relationships and interaction, while conventional statistical analysis in most cases implicitly assumes a linear relationship between independent variables and dependent variables. Neural network builds its own models through...
a learning process, whether the relationships among variables are linear or not. Neural networks can also deal with missing or incomplete data, and can update the weights among input, output and intermediate nodes, so that even incomplete data can contribute to learning and can produce desired output results. Back-error propagation is one of the most widely used neural network paradigms and has been applied successfully to many application studies in a broad range of areas [3][33][37][39].

Cai et al. [7] examined that handling datasets with ‘smooth’ trends is more effectiveness in the neural network approach than handling datasets with large fluctuations. They found that the neural network approach is much better in prediction than SRGMs. Sherer [35] predicted software faults in several NASA projects with neural networks [13][22][23]. It should be noted that the neural network-based approach has some drawbacks for application in software reliability assessment. The major problem is the design of neural network architecture, involving the number of input neurons in each layer and the number of hidden layers, so that both of them must be determined carefully through trial-and-error heuristics. Caruana [9] showed generalization results on a variety of problems as the size of the networks varies. Second, in related works on neural network application to predict the number of software faults, the prediction is always deterministic. In other words, the point prediction of the number of software faults detected at each testing day is given as an output of complex non-linear functions with trained parameters. In this case, it can be expected that the accuracy of future point forecast significantly reduces. More specifically, when the training data in neural networks is sparse, the point prediction in neural computing may be less reliable [4].

A feed forward neural network or multilayer perceptron (MLP) has an input layer, an output layer, and one or more hidden layers. Since a neural network with one hidden layer is capable of approximating any arbitrary function, in this paper we consider this simplest neural network architecture with only one single layer of hidden layer. One of the most important problems in neural network design is deciding the optimal size of the hidden layer (or the optimal number of hidden neurons) in order to achieve the best approximation/prediction performance for a given application. When the neural network is trained, the error of training set becomes small for training data. On the other hand, when new data is available to the neural network, the error may be extremely large [3]. In this paper, we concentrate on this issue and try to determine the optimal number of hidden neurons and input neuron through experiments.

In this paper we derive prediction intervals of the cumulative number of software faults detected at each testing day using the MLP neural network, where the simplest three layers MLP is assumed with the well-known back-propagation algorithm. To predict the number of software faults, we impose a plausible assumption that the underlying fault-detection process obeys the Poisson law with an unknown parameter. Since it is appropriate to input the training data as real number in the conventional MLP neural networks, we propose to apply three data transform methods from the Poisson count data to the Gaussian data: Bartlett transform [5], Anscombe transform [1] and Fisz transform [12]. We experiment to find out the significance of the neural network architecture for input neuron and hidden neuron, and apply the delta method [4] to get the prediction intervals of the cumulative number of software faults in the one-stage look ahead prediction. In numerical experiments with actual software development project data, we show that our methods are useful to calculate the prediction intervals on the on-going progress of software debugging process.

2. Background

2.1 Neural Network Architecture

Artificial neural networks are widely used for functional approximation and statistical inference. The term of “artificial neural network” usually refers to mathematical model employed in neural network computing and artificial intelligence. Neural networks have the learning ability based on the training data or initial experiences, similar to the human brain. Although the neural network in the human brain is composed of a large number of highly interconnected processing elements (neurons) working in parallel, much simpler structure with input layer, hidden layer and output layer is assumed for the common MLP feed forward artificial neural network. It consists of an input layer with some inputs, hidden layer with hidden neurons and one output layer. The input layer of neuron can be used to capture the inputs from the outside world. The hidden layer of neurons has no communication with the external world, but the output layer of neuron sends the final output to the external world. The main function of hidden layer neurons is to receive the inputs and weights from the previous layer and to transfer the aggregated information to the output layer by any transfer function. This output can act as an input of the output layer. The input layer neurons do not have any computational task. It just receives inputs and associated weights, and passes them to the next layer.

The value coming out an input unit (neuron) is labeled by $x_i$ for $i = 1, 2, 3, \ldots, n$, where $n$ is the number of input neurons. There are also two special inputs, bias labeled by $B_1$ and $B_2$, which always have the unit values. These inputs are used to evaluate the bias to the hidden nodes and output nodes, respectively. Let $z_i (j = 1, 2, 3, \ldots, m)$ be the hidden neuron output, where $m$ is the number of hidden neurons in the hidden layer. Let $w_{ij}$ be the weight from $i$-th input to $j$-th hidden neuron, where $w_{0j}$ denotes the bias weight form $B_1$ to $j$-th hidden
neuron. The bias input node $B_1$ is connected to all the hidden neurons and $B_2$ is connected to the output neuron. Figure 1 illustrates the MLP neural network architecture under consideration, where forward information corresponds to the data processing which is output from the feed forward architecture, and backward information denotes the data through the back-propagation algorithm. Each hidden neuron calculates the weighted sum of the input neuron $z_j$ in the following equation:

$$z_j = \sum_{i=1}^{n} w_{ij} x_i + w_{0j} B_1,$$

Substituting the weighted sum into the thresholding function which is typically either a step function or a sigmoid function, we can obtain the output from the hidden neuron $\tilde{z}_j$, where

$$\varphi(z_j) = \frac{1}{1+\exp(-z_j)}.$$

Then, the output from the output layer is given by $y = \sum_{j=2}^{m} w_{jd} \tilde{z}_j + w_{0d} B_2$,

where $y$ is the summative and weighted inputs from each hidden neuron of the output layer, $w_{jd}$ is the weight going from the $j$-th hidden neuron to the output neuron, and $w_{0d}$ represents the weight from the bias to the output neuron. Finally, the predicted value of the network, $\tilde{y}$, is calculated as $\varphi(y) = \frac{1}{1+\exp(-y)}$.

### 2.2 Back Propagation Algorithm

In our MLP architecture, all the processing units of input layer are interconnected to all the processing units of the hidden layers, and all the processing units of the hidden layer are interconnected to an output unit, where each weight is associated with each connection. The well-known gradient descent method is used to update the weights so as to minimize the squared error between the network output value and the target output value. Then, each weight is adjusted using the gradient descent, according to its contribution to the error. This procedure is iteratively made for each layer of the network, starting with the last set of weights, and working back towards the input layer, until the desired output is achieved. This is called the back propagation algorithm [27] and is the standard learning algorithm for artificial neural networks. The average sum of squared errors ($E^2$) for the input pattern is represented as

$$E^2 = \frac{\sum_{r=1}^{N} (Y_r - \tilde{Y}_r)^2}{(N - 1)}, r = 1, 2, ..., N,$$

where $\tilde{Y}_r$ is the predicted value (point prediction as the neural network output) at $r$-th testing day, $Y_r$ is the true value of the number of detected software faults and $N$ is the prediction period (integer value).

### 2.3 Data Transform

It is common to input real value data in the MLP neural network. Since our problem is to predict the number of software faults newly detected at the next testing day, however, the underlying data should be integer. In general, it is convenient to treat real number in almost all neural network computing, and to apply the useful property of the Gauss distribution for constructing the prediction intervals approximately (e.g., see [24]). Hence we suppose that the software fault count is described by the Poisson law [31][32][34][43]. In the existing literature, some of authors concern the prediction of the software fault-detection time and handle the real number in their neural network calculations.

We apply Bartlett transform [5], Anscombe transform [1] and Fisz transform [12] as the most well-known normalizing and variance-stabilizing data transforms. The Anscombe’s square-root transform (AT) is widely used to pre-process the Poisson data before processing the Gaussian data. Taking the AT, the cumulative number of software fault data can be approximately transformed to the Gaussian data: $S_r = 2\sqrt{Y_r + 3/8}$, where $Y_r$ is the cumulative number of software faults at $r$-th testing day. The AT is a natural extension of the well-known Bartlett transform (BT), which is known as the most fundamental data transform tool in statistics, where BT is defined by $B_r = 2\sqrt{Y_r + 1/2}$. Finally, the Fisz transform (FT) is characterized by the following square root transform as an extension of BT: $FT_r = \sqrt{Y_r + 1} + \sqrt{Y_r}$.

### 2.4 Delta Method

Delta method is known as an elementary method of propagation of errors. It is a commonly used approach which is easily implemented, not computer-intensive, and
can be robustly applied to many situations such as an approximation of the variance for an arbitrary functional of random variables, based on Taylor series expansions. It is also used to find an asymptotically statistical estimator from the knowledge of the limiting variance [17][24][41].

Let $\delta_r^T$ be the output gradient vector with respect to gradient values for all output and hidden neurons in the MLP neural network.

$$\delta_r^T = [\delta_{O_1}, \delta_{O_2}, ..., \delta_{O_r}, \delta_{H_1}, \delta_{H_2}, ..., \delta_{H_m}],$$

where $\delta_{O_r} (r = 1,2,3,...,N)$ are the output gradient of the output layer, $\delta_{H_j} (j = 1,2,3,...,m)$ are the output gradient of the hidden neuron, $m$ indicates the number of hidden neurons of a hidden layer and $a_r$ is the $r$-th actual value. Then, we have

$$\delta_{O_r} = \hat{y}_r (1-\hat{y}_r) (a_r - \hat{y}_r),$$

$$\delta_{H_j} = \delta_{O_r} w_{jd} \phi(H_j) \left(1-\phi(H_j)\right).$$

In practice, the neural network parameters such as pervious weights have to be adjusted by minimizing the average $E^2$.

Let $\Delta w_r$ is the Jacobian matrix with respect to all updated weight parameters from an output neuron to hidden neurons. It is computed for all the training samples, where

$$\Delta w_r = \begin{bmatrix} w_{1d(new)} & w_{11(new)} & w_{12(new)} & \cdots & w_{1l(new)} \\ w_{2d(new)} & w_{12(new)} & w_{22(new)} & \cdots & w_{12(l(new))} \\ w_{3d(new)} & w_{13(new)} & w_{23(new)} & \cdots & w_{13(l(new))} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{jd(new)} & w_{lj(new)} & w_{jd(new)} & \cdots & w_{lj(new)} \end{bmatrix}.$$ 

In the above equation, $w_{jd(new)} (d = 1)$ are the new weights of hidden neurons connected to the output neuron, and $w_{lj(new)}$ are the new weights of $n$ input neurons which are connected to $m$ hidden neurons in the hidden layer. These weights, $w_{jd(new)}$ and $w_{lj(new)}$, are given as follows.

$$w_{lj(new)} = w_{ij} + \alpha w_{ij} + \eta \delta_{H_j} x_i,$$

$$w_{jd(new)} = w_{jd} + \alpha w_{jd} + \eta \delta_{O_r} \hat{y}_r.$$

Define the PIs of software fault by $[P_{-I_{low}}, P_{-I_{up}}]$ in the MLP neural network computing. Then, the lower limit and upper limit of PI are given by

$$P_{-I_{low}} = \hat{y}_r - t_{N,p}^{1/2} E \sqrt{1 + \delta_r^T (\Delta w_r^T \Delta w_r)^{-1} \delta_r},$$

$$P_{-I_{up}} = \hat{y}_r + t_{N,p}^{1/2} E \sqrt{1 + \delta_r^T (\Delta w_r^T \Delta w_r)^{-1} \delta_r}.$$ 

In the above equations $t_{N,p}^{1/2}$ is the $(a/2)$ - quantile of the student $t$-distribution function with $(N- p)$ degree of freedom, and $p$ is the number of inputs in the neural network.

The Jacobian matrix ($\Delta w_r$) and its gradient value ($\delta_r^T$) are quite hard to obtain with all input data, so delta method contains a somewhat puzzling issue to construct PIs. However the other calculations are comparatively modest. In this paper, the Jacobian matrix and the gradient value are calculated and estimated at off-line, although they can be potential sources of computational error for constructing PIs. In addition, the quality of PIs and their optimal values of gradient and Jacobian matrix must be carefully checked to satisfy the convergence condition that the minimum error is achieved at a tolerance level.

3. Numerical Illustrations

3.1 Experimental Process

The overall experimental process is described in Figure 2. In the first step, we consider two cases on preprocessing the dataset; one is data transforming from the Poisson data to the Gaussian data with BT, AT and FT, another without transform. The data with/without transform are input to the neural network, where the output are the point prediction of the cumulative number of software faults detected at the next testing day. At the last step we assess the PIs of the cumulative number of software faults by using the delta method.

3.2 Setup

We use eight real project data sets cited in [31]; DS1–DS8, which consist of the software-fault count data. Table I summarizes the data sets and their cumulative numbers of software faults detected in testing. To find out the desired output via the back propagation algorithm, we need much computation cost to calculate the gradient descent and the Jacobian matrix. Especially, the momentum ($\alpha$) and the learning rate ($\eta$) are the most important turning parameters, where $\alpha$ adjusts the weights and $\eta$ depends on the convergence speed in the back propagation algorithm. We carefully examine $\alpha$, $\eta$, the initial guess of weight, number of total iterations and the tolerance level parameters in pre-experiments [4].
3.3 Prediction Performance

<table>
<thead>
<tr>
<th>DS #</th>
<th>Number of Faults</th>
<th>Project Type</th>
<th>Source # [Citation]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>62</td>
<td>Command and Control subsystem</td>
<td>Data &amp; Analysis Center for Software (DACS) [31]</td>
</tr>
<tr>
<td>2</td>
<td>41</td>
<td>Flight Data subsystem</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>114</td>
<td>Command and Data subsystem</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>Real Time Command &amp; Control</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>73</td>
<td>Commercial Subsystem</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>181</td>
<td>Command and Data subsystem</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>81</td>
<td>Brazilian Electronic Switching System</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>140</td>
<td>Telecommunications switch software</td>
<td></td>
</tr>
</tbody>
</table>

The prediction performance is evaluated in sequential software testing, so that we make the one-stage look-ahead prediction based on the past observation and sequentially evaluate the prediction performance. As a prediction performance measure, we introduce the average relative error. Suppose that the observation point is the \((k+1)\)-st testing day and that \((k-1)\) software fault counts data are available. For the actual value on the number of software fault counts at the \(k\)-th testing day, the relative error \((RE)\) is defined by

\[
RE = \left| \frac{\hat{y}_r - Y_r}{a_r} \right|
\]

where \(\hat{y}_r\) is the prediction value at \(r\)-th testing time and \(Y_r\) is the \(r\)-th actual value (training data). The average relative error takes account of the past history and is defined by \(AE = \frac{1}{N} \sum_{r=1}^{N} RE_r\).

### A Effect of number of hidden nodes

One of the major problems facing by researchers is the selection of input and hidden neurons number for the architecture of neural networks. To keep relatively small errors on software fault prediction, it is very important to train the neural network. The minimal error reflects better stability, and higher error reflects worst stability. We experiment with different numbers of hidden neurons for a fixed number of input neurons for all datasets. The excessive hidden neurons cause the so-called over fitting problem and tend to overestimate the complexity of the target problem. To choose the optimal number of hidden neurons we follow the well-known "rules of thumb" for choosing a suitable architecture. In this paper we determine the number of hidden nodes by \((2/3)\) (number of inputs+outputs) [44]. Table II lists abbreviations used in this paper.

Table III presents the prediction performance based on AE for eight datasets with and without data transform, where the values in “Architecture” denote the number of input neurons, the number of hidden neurons and the output neuron \((1)\). According to rules of thumb, we fix input neuron numbers and change the hidden neurons. From this result, FT provides the better prediction performance in all cases, except in DS#6, because, the neural network gives the relatively small output values for this dataset, so among other transforms. In addition, we observe that the architecture, \(5:3:1, 5:4:1, 10:7:1, 15:10:1\) and \(20:14:1\), lead to less error than the other architecture. From the results we find that the rule of thumb is rather accurate to obtain the nearly optimal network architecture.

### B Effect of number of input nodes

In most situations, there is no way to determine the best number of input nodes without training several networks and estimating the generalization error. If we have too many input neurons, we may get higher training error and higher
generalization error, due to under-fitting and large statistical bias. Once the best number of hidden neurons is known, we can change the input neuron numbers and can make a different architecture to find out the optimal number of input neurons. Table IV presents the prediction performance based on AE for all datasets with and without data transform. It is shown that when the number of input neurons increases, the error rate also increases, but some cases e.g., DS#3–DS#5 without transform provide the better results for 11:7:1, 12:10:1 and 12:7:1 than FT. It should be noted that the neural network has one major drawback for application in software reliability assessment, i.e., in the common neural network computing the initial weight is randomly selected, so it acts as a trial-and-error heuristics.

### TABLE IV. AVERAGE RELATIVE ERROR IN POINT PREDICTION.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Architecture</th>
<th>AT</th>
<th>FT</th>
<th>BT</th>
<th>No Transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5:3:1</td>
<td>0.3370</td>
<td>0.1041</td>
<td>0.4139</td>
<td>0.1433</td>
</tr>
<tr>
<td>2</td>
<td>5:3:1</td>
<td>0.5148</td>
<td>0.9771</td>
<td>1.3048</td>
<td>2.1271</td>
</tr>
<tr>
<td>3</td>
<td>5:3:1</td>
<td>0.5236</td>
<td>0.1255</td>
<td>0.4253</td>
<td>0.3215</td>
</tr>
<tr>
<td>4</td>
<td>5:3:1</td>
<td>0.2417</td>
<td>0.5412</td>
<td>1.1774</td>
<td>1.1478</td>
</tr>
<tr>
<td>5</td>
<td>5:3:1</td>
<td>0.1639</td>
<td>0.2685</td>
<td>0.8222</td>
<td>0.3348</td>
</tr>
<tr>
<td>6</td>
<td>5:3:1</td>
<td>1.5326</td>
<td>1.0235</td>
<td>1.1253</td>
<td>1.9356</td>
</tr>
<tr>
<td>7</td>
<td>5:3:1</td>
<td>1.7969</td>
<td>0.5747</td>
<td>0.9225</td>
<td>0.9162</td>
</tr>
<tr>
<td>8</td>
<td>5:3:1</td>
<td>0.9043</td>
<td>0.2685</td>
<td>0.7069</td>
<td>0.6712</td>
</tr>
<tr>
<td>9</td>
<td>5:3:1</td>
<td>2.5369</td>
<td>1.2536</td>
<td>3.1235</td>
<td>3.3698</td>
</tr>
<tr>
<td>10</td>
<td>5:3:1</td>
<td>2.1862</td>
<td>1.2515</td>
<td>1.9189</td>
<td>1.3079</td>
</tr>
<tr>
<td>11</td>
<td>5:3:1</td>
<td>0.6845</td>
<td>0.6122</td>
<td>0.8647</td>
<td>0.7851</td>
</tr>
<tr>
<td>12</td>
<td>5:3:1</td>
<td>1.1627</td>
<td>0.1048</td>
<td>2.1147</td>
<td>1.2387</td>
</tr>
<tr>
<td>13</td>
<td>5:3:1</td>
<td>1.3831</td>
<td>0.9713</td>
<td>1.1568</td>
<td>0.6985</td>
</tr>
<tr>
<td>14</td>
<td>5:3:1</td>
<td>0.8704</td>
<td>0.7001</td>
<td>0.9881</td>
<td>0.9257</td>
</tr>
<tr>
<td>15</td>
<td>5:3:1</td>
<td>1.7845</td>
<td>1.0134</td>
<td>1.1573</td>
<td>1.6134</td>
</tr>
<tr>
<td>16</td>
<td>5:3:1</td>
<td>1.9481</td>
<td>1.2763</td>
<td>1.1398</td>
<td>1.1398</td>
</tr>
<tr>
<td>17</td>
<td>5:3:1</td>
<td>1.9592</td>
<td>1.0431</td>
<td>1.1207</td>
<td>1.2078</td>
</tr>
<tr>
<td>18</td>
<td>5:3:1</td>
<td>1.9475</td>
<td>1.3231</td>
<td>1.8357</td>
<td>0.9147</td>
</tr>
<tr>
<td>19</td>
<td>5:3:1</td>
<td>0.8387</td>
<td>0.6914</td>
<td>1.3731</td>
<td>0.9357</td>
</tr>
<tr>
<td>20</td>
<td>5:3:1</td>
<td>2.0931</td>
<td>1.0537</td>
<td>1.4526</td>
<td>1.0535</td>
</tr>
</tbody>
</table>

In Fig. 3, we illustrate the time-dependent behavior of RE and compare four data transform methods (one is no transform) with DS3. From this result, it can be observed that the steadiest method is FT. In the middle and latter testing phases of software testing, the common approach without transform gives the superior results, comparing with other phases. On the other hand, the most classical BT and AT deliver the aiming results and give the growing trend as the software testing goes on in comparison of all transform methods.

### 3.4 One-stage look ahead point prediction

To get the one-stage look ahead point prediction value, we use DS # 4. Noting that our dataset contains 22 days, we wish to know 23rd point prediction value of the cumulative number of software faults for all transform methods. From the result of Tables III and IV, it is seen that FT can give less error rate and more reliable prediction results than the others. Figure 4 shows that result based on the best neural network architecture. From this figure we can say that FT gives little larger value than the others.

### 3.5 PI Assessment

In order to measure the quality of PIs on the prediction of software fault counts, we need to define three prediction measures called the PI coverage Probability (PICP), the mean prediction interval width (MPIW) [4] and PI-normalized averaged width (PINAW) [25]. Assume the significance level as 95%. PCIP is the portion of the number
of the software fault counts covered by the PI, and is defined by
\[ \text{PICP} = \frac{\sum_{i=1}^{N} c_i}{N}, \]

where
\[ c_i = \begin{cases} \mathbf{1}, & \hat{y}_i \in [L_i, U_i] \\ 0, & \hat{y}_i \notin [L_i, U_i] \end{cases} . \]

Let \( L_i \) and \( U_i \) be the lower and upper prediction limits, \( CP_i \) be the coverage probability at \( i \)-th testing day \( (i=1, 2, \ldots, k-1) \) and \( \hat{y}_i \) be the predicted number of software faults. Then, MPIW evaluates the width of PIs, and is defined by
\[ \text{MPIW} = \frac{\sum_{i=1}^{N} (U_i - L_i)}{N}. \]

PI-normalized averaged width (PINAW) quantifies the wide constructed PIs; \( \text{PINAW} = \frac{\text{MPIW}}{R} \), where \( R \) is the range of the underlying target, and is used to compare PIs.

In Table V we give the prediction PI measures for eight datasets. It can be seen that the no transform works to increase PICP because the corresponding MPIWs become wider. In practical usage of PIs, the associated coverage rate is large enough if the width is narrow. From this table we can say that in most of the cases FT provides narrow width with higher coverage rate excluding DS# 6. On the other hand, BT can deliver the better result than FT. Since the sharper PIs are theoretically more informative and practically more useful than the wider PIs, it is noted that PINAWs indicate the sharpness of PIs, so that smaller PINAW is regarded as better PIs.

Since software is written by humans, errors will be always involved in the product. From this well-known fact,
it can be recognized that the PIs can predict the number of software fault counts under uncertainty, which will experience in the future operational phase, and can be useful for the probabilistic inference with subjective significance level controlled by the software test manager.

In Figs. 5-8, we depict the sequential prediction results of software fault counts under uncertainty, which will experience in the future operational phase, and can be useful for the probabilistic inference with subjective significance level controlled by the software test manager.

![Graph showing Number of Faults](image)

**Fig. 8.** Case with no transform (DS8:5:4).

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Prediction PI Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Transform</td>
<td>PICP</td>
</tr>
<tr>
<td>DSI(5,3)</td>
<td>No Transform</td>
</tr>
<tr>
<td></td>
<td>AT</td>
</tr>
<tr>
<td></td>
<td>FT</td>
</tr>
<tr>
<td></td>
<td>BT</td>
</tr>
<tr>
<td>DSI(5,4)</td>
<td>No Transform</td>
</tr>
<tr>
<td></td>
<td>AT</td>
</tr>
<tr>
<td></td>
<td>FT</td>
</tr>
<tr>
<td></td>
<td>BT</td>
</tr>
<tr>
<td>DSI(10,7)</td>
<td>No Transform</td>
</tr>
<tr>
<td></td>
<td>AT</td>
</tr>
<tr>
<td></td>
<td>FT</td>
</tr>
<tr>
<td></td>
<td>BT</td>
</tr>
<tr>
<td>DSI(14,10)</td>
<td>No Transform</td>
</tr>
<tr>
<td></td>
<td>AT</td>
</tr>
<tr>
<td></td>
<td>FT</td>
</tr>
<tr>
<td></td>
<td>BT</td>
</tr>
<tr>
<td>DS11(10,7)</td>
<td>No Transform</td>
</tr>
<tr>
<td></td>
<td>AT</td>
</tr>
<tr>
<td></td>
<td>FT</td>
</tr>
<tr>
<td></td>
<td>BT</td>
</tr>
<tr>
<td>DSI(5,4)</td>
<td>No Transform</td>
</tr>
<tr>
<td></td>
<td>AT</td>
</tr>
<tr>
<td></td>
<td>FT</td>
</tr>
<tr>
<td></td>
<td>BT</td>
</tr>
<tr>
<td>DSI(4,5)</td>
<td>No Transform</td>
</tr>
<tr>
<td></td>
<td>AT</td>
</tr>
<tr>
<td></td>
<td>FT</td>
</tr>
<tr>
<td></td>
<td>BT</td>
</tr>
</tbody>
</table>

In Figs. 5-8, we depict the sequential prediction results of software fault counts and their PIs with four data transform methods (including the case with no transform) for the different datasets, where the best architecture in each dataset is applied. It can be found that the PIs with FT can cover both of the one-stage look ahead point prediction and the actual data itself. Figure 7 show that BT for DS8 does not cover the point prediction in PIs. Figure 8 illustrates that the case with no transform for DS8 contains the PI coverage the point prediction value in all cases.

### 4. Discussion and Conclusion

In this paper we have derived the prediction interval of the cumulative number of software faults in testing phase and investigated the effect of a number of hidden and input nodes on prediction accuracy according to the rules of thumb. The experimental results have shown that the proposed approach gave the acceptable results for prediction using the different neural network architecture. In numerical experiments with actual software development project data, we have evaluated the resulting prediction intervals and found that those could cover both the point prediction and the actual data in their regions.

In future, these experimental results have to be justified through Monte Carlo simulation, by comparing the “real” prediction intervals under the well-defined parametric circumstance. Also, we will apply the other PIs methods to construct the long-term prediction and optimize the weight for the neural network architecture to get the exact PIs.

### Acknowledgement

The first version of this paper was presented at The 3rd International Conference on Applied Computing & Information Technology (ACIT2015), Okayama, Japan, July 12-16, 2015. The first author was partially supported by the MEXT (Ministry of Education, Culture, Sports, Science, and Technology), Japan government scholarship.

### References


Begum Momotaz received the B.Sc. and M.Sc. degrees in Computer Science & engineering from Dhiaka University of Engineering & Technology (DUET), Gazipur, Bangladesh in 2008 and 2013, respectively. In 2013, she joined the Department of Computer science & Engineering, DUET, Gazipur, Bangladesh, as an Assistant Professor. She is now working toward the Ph.D. degree in the Graduate School of Engineering, Hiroshima University, Japan.

Tadashi Dohi received the B.Sc., M.Sc., and Ph.D. degrees in engineering from Hiroshima University, Hiroshima, Japan, in 1989, 1991, and 1995, respectively. In 1992, he joined the Department of Industrial and Systems Engineering, Hiroshima University, Japan, as an Assistant Professor. Since 2002, he has been working as a Full Professor in the Department of Information Engineering, Graduate School of Engineering, Hiroshima University. In 1992 and 2000, he was a Visiting Research Scholar in University of British Columbia, Canada, and Duke University, Durham, NC, USA, respectively, on leave of absence from Hiroshima University. His research areas include software reliability engineering, dependable computing, and performance evaluation. He is a regular member of ORSJ, IEICE, IPSJ, REAJ and IEEE.
**Information for Authors**

**Scope**

The aim of the International Journal of Computer & Information Science is to present information on well defined theoretical results and empirical studies, to describe current and prospective practices and techniques, to provide critical analysis of the use of CASE tools and to present broad issues in the fields of Computer Science and Engineering, Software Engineering and Information Science and Engineering. A primary aim of the journal will be to make the research accessible and useful to practitioners through carefully written methodology section and discussions of research results in ways that relates to practitioners.

**Submission Policy**

1. The IJCIS does not accept manuscripts or works that have been published in a journal, but accepts substantially revised and expanded versions of papers that were previously presented at conferences. In this case, the author should inform the Editor-in-Chief of the previous publication when the paper is submitted.

2. Submissions, referencing and all the correspondence will be conducted by e-mail.

3. Manuscripts should be in English and are subject to peer review before selection.

**Procedure for Submission for Review**

1. Send your complete text of manuscript to the Editor-in-Chief (e-mail: lee@cps.cmich.edu) via e-mail as an attachment document either in Word, PDF or PostScript format.

2. It may be useful for non-English speakers to have a native English speaker read through the article before submission.

3. Your e-mail message (to the Editor-in-Chief) should include the title of your paper, author(s) information (department/section, university/company, phone/fax numbers, e-mail address), the corresponding author’s complete address (mailing address), and area(s) of the paper.

**Style of Manuscript to be Submitted for Review**

1. The text should be double-spaced, two columned, single-sided in Times 12-point type on 8.5-inch x 11-inch paper with margins of 1 inch on all four sides. It should not exceed 30 double-spaced pages. All pages must be numbered sequentially.

2. Provide a 200-250 word abstract, 5-7 keywords. Your abstract should briefly summarize the essential contents.

3. All figures/tables must be captioned, numbered and referenced in the text.

4. An acknowledgement section may be presented after the conclusion section.

5. References should appear at the end of the paper with items referred to by numbers in square brackets.

   **Journal Papers:** author’s first initials, followed by last name, title in quotation marks, journal (in Italics), volume and issue number, inclusive pages, month and year. For e.g.:


   **Papers of Proceedings:** author’s first initials, followed by last name, title in quotation marks, proceedings / symposium / workshop name (in Italics), location, month, and year, and inclusive page numbers. For e.g.:


   **Books:** author’s first initials, followed by last name, title (in Italics), edition, location: publisher, year. For e.g.:


**Accepted Manuscript Submission**

All accepted papers are edited electronically. You are required to supply an electronic version of the accepted paper. The detailed guidelines for preparing your final manuscript will be sent upon article acceptance. It is also available on the ACIS website at [http://acisinternational.org](http://acisinternational.org).

**Page Charges**

After a manuscript has been accepted for publication, the author will be invoiced a page charge of $400.00 USD for up to 10 pages if he/she is not a member of ACIS and thereafter an additional $30 per printed page to cover part of publication after the receipt of your final manuscript (camera ready).