Data- and Algorithm-Hybrid Approach for Imbalanced Data Problems in Deep Neural Network

Rheza Harliman and Kaoru Uchida

***Abstract*—Data imbalance is one of the problems that we face when applying machine learning to real-world problems, especially in image classification. With all the improvements in machine learning, especially deep learning, research in this area is drawing more attention from academics and even industry. To address this imbalanced data problem, we adopt a hybrid (algorithm and data) approach that consists of data manipulation and weighted loss function in this paper. We propose Ripple-SMOTE as a novel oversampling method to generate synthetic data for preprocessing. A deep neural network and the weighted loss function is applied so it will not treat all classes equally. We also use a pre-trained model and fine tune it to improve the classification accuracy. In this paper, we report the evaluation results using imbalanced data sets based on MNIST, CUReT texture set, and Malware data set, and show that our approach significantly improves the performance in imbalanced data cases and outperforms the conventional approaches, especially in handling minority classes.**

***Index Terms*—Deep neural network, imbalanced data, oversampling.**

I. Introduction

The Performance of machine learning, specifically deep learning, heavily depends on the quality of data. Most deep CNNs are trained by properly designed balanced data [1]. However, imbalanced distribution for each class is such a common thing that we face it very often if we are to tackle real world problems, for example, fraudulent credit card transactions, medical diagnosis, software defects, etc.

Class imbalance occurs when the instances of one class are far less in number than the instances of another class [2]. The main issue regarding imbalanced learning is the likelihood of the imbalanced data to compromise the performance of standard learning algorithm [3].

There are three reasons that cause the compromises [4]. The first reason is that the lack of data in the minority class makes it difficult to detect regularities within the minority class. Thus, the learned decision boundaries are less likely to approximate the true decision boundaries. Secondly, there are many classification algorithms that utilize a general bias for better generalization and to avoid overfitting during training. Last is the noise exerts a greater impact on minority class rather than majority, because the data limitation in the minority class makes it difficult for a classifier to distinguish noise. This is more problematic especially in extreme cases where the number of noisy samples is greater than the actual minority samples. If it happens, overfitting will likely happen again.

To solve these problems, research is ongoing with variety of techniques. Approaches to solve this imbalanced data are broadly divided into two categories [5]: Algorithm- and Data-approaches. Most of the algorithm level methods typically create a new or improved current algorithm to fit the biased data distribution. One of them uses the cost-sensitive learning [6] or a loss function [7]. By amplifying misclassification of the minority class and suppressing the majority, it improves the training performance. Data approaches include oversampling, undersampling, resampling, and data manipulation. Some of the famous oversampling methods are SMOTE [8], Borderline-SMOTE [9], safe-level SMOTE [10] and ADASYN [11]. Oversampling is used to generate synthetic samples to improve the number of samples in minority class. There's also combination of oversampling and undersampling method like SMOTE-ENN [12] and SMOTE-TOMEK [13].

The rest of paper is organized as follows; in Section II, we discuss related works. Section III introduces our approach to address the imbalance problem using a hybrid approach. In Section IV we explain about the evaluation and its result, then conclude it in Section V.

II. Related Works

A. Synthetic Minority Oversampling Technique (SMOTE)

SMOTE (Synthetic Minority Oversampling Technique) [8] is used to balance the minor data set by increasing its samples. It creates synthetic samples with considering the nearest neighbor value. SMOTE consists of several steps:

a) For samples in minority classes, use Euclidean Distance to calculate the distance between each sample and apply k-nearest neighbor from those samples.

b) Take *n*-samples randomly from k-nearest neighbor results. Yan [5] proposes another way in counting how many samples are to be created using the equation below:

$$n = \text{round}(\text{imbalancedratio}) - 1$$

$$\text{imbalancedratio} = \frac{S_{\max}}{S_{\min}}$$

c) From the set of samples y_n taken from k-nearest neighbor, construct a new synthetic sample based on interpolation formula

$$x_{new} = x + rand(0,1) \times (y_n - x), i = 1,2,...n$$

Manuscript received March 5, 2018; revised May 7, 2018. This work was supported in part JSPS Grants-in-Aid for scientific Research (KAKENHI) grant number JP17K00138.

The authors are with Graduate School of Computer and Information Sciences, Hosei University, Tokyo, Japan (e-mail: rheza.harliman.5q@stu.hosei.ac.jp).

By applying the oversampling, it is expected that the classifier can build a larger decision region that contains nearby minority class points.

Tomek link [12] is an undersampling method that only removes samples belonging to majority class. Given two samples from different classes, E_i and E_j, a pair is called tomek links if there's no sample E_l, such distance $(E_i, E_i) <$ distance(E_i, E_j). When a Tomek link is performed between two samples, either one of these samples is noise or both are samples lying on a borderline.

Tomek link in SMOTE-Tomek is used for cleaning the samples after applying the oversampling, since interpolating minority class samples expand the minority class clusters and those samples might have gone too deeply into majority class space. Fig. 1 shows the procedure of SMOTE-Tomek.

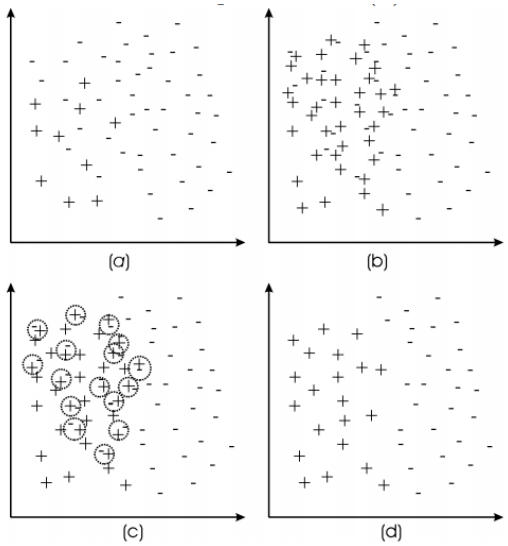


Fig 1. Balancing Data set (a) original data set; (b) oversampled data set; (c) Tomek links identification; (d) borderline and nose examples removal.

B. Loss Function

Softmax loss [8] consists of softmax regression and entropy loss that is widely used in multi-class classification. Suppose we have a *K*-class training set consisting of *n* samples: $\{x^{(i)}, y^{(i)}\}$ where *x* is a sample vector and *y* is the label. If $a_j (j = 1, 2, \ldots, K)$ is the output unit from the fully connected layer, then the probability function that $x^{(i)}$ *is j* can be formulated as:

$$P_j^{(i)} = \frac{\exp(a_j^{(i)})}{\sum_{l=0}^{K} \exp(a_l^{(i)})}$$

And to minimize the entropy loss function, we can use formula like below.

$$J_0 = -\frac{1}{n}\{\sum_{i=1}^{n}\sum_{j=0}^{K}(y(i) = j)\log P_j^{(i)}\}$$

The approach of weighted softmax loss, that is supposed to solve this limitation can be formulated as below,

$$J_0 = -\frac{1}{n}\{\sum_{i=1}^{n}\sum_{j=0}^{K} w \times (y(i) = j)\log P_j^{(i)}\}$$

$$w = 1 + \frac{S_{\max} - S_k}{\beta \times S_{\max}}$$

where β is the parameter that scales the weighted loss (*w*). This means that majority classes will gain more weights compared to minority classes.

III. Proposed Method

The hybrid approach we propose here is a combinational approach to solve the imbalanced data problem of data manipulation and algorithm. For data manipulation, we employ synthetic oversampling using our novel implementation in SMOTE and we use weighted loss function as the algorithm approach.

A. Ripple-SMOTE

One of our basic ideas to solve the imbalance problem is to reduce the sample imbalanced ratio between majority class data and minority class data.

Reducing the ratio's differences between samples can be done by manipulating the data by oversampling of minority class samples and/or downsampling of majority class samples. In this paper we propose a novel approach to improve SMOTE (Ripple-SMOTE) in order to achieve better data proportion and better prediction.

Not like the existing oversampling methods, our method strengthens the borderline of minority samples and improve the features by creating synthetic samples by taking *i*-farthest data from centroids and downsampling *n*-nearest data from the majority set. Fig. 2 is the simulation of the classes samples with centroids. The x with green color in the center of the data is the centroid. The line drawn is the border that is taken from the farthest sample. Arrows pointed to samples shown some of the farthest samples around the centroid.

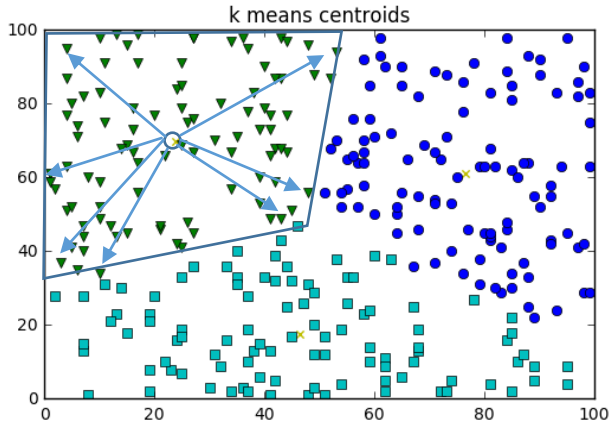


Fig. 2. Centroid data and farthest data simulation.

Suppose the whole training set is *D*, and *J* for majority samples, *M* for minority samples,

$$J = \{j_1, j_2, j_3, \ldots, j_k\}, M = \{m_1, m_2, m_3, \ldots, m_l\}$$

where *k* and *l* are number of majority and minority samples respectively. Following are the detail steps for Ripple-SMOTE.

One of the most important points in this method is use of the centroid of the data for each class. Suppose M_n^c is the sample from minority class *c*, $center^c$ is the centroid of class *c*, and N_c is number of samples in class c, we can calculate the centroid at that class with following formula,

$$\text{center}^c = \frac{\sum_{n=1}^{N_c^m} M_n^c}{N_c^m}$$

Next, we should find the borderline by finding the distance from each of samples in minority to the centroid. Taking the insight from the approach of borderline-SMOTE [7], we calculate all the distance from the majority sample with the distance from the farthest sample of the minority class. Suppose the distance between the farthest sample from majority sample is t^{maj}, and $t_n^{\min}$ is the distance from centroid to farthest sample, then we can identify which majority samples that may be misclassified into the minority and vice versa. We take *n*-samples of majority samples that has smallest t^{maj} and remove it from minority class and store the farthest minority sample into a *FARTHEST* set.

$$FARTHEST^c = \{m'_1, m'_2, m'_3, ..., m'_l\},$$
$$0 \le FARTHEST_l^c \le N_c$$

The imbalance rate is important in the next step. We are giving each class an imbalance rate with following formula,

$$\text{imbalancedrate} = \frac{\max(N_c^j)}{N_c^m}$$

We are assuming that making synthetic samples only from borderline of the data is not enough. There is still room for improvement especially when we try to get nearer to the centroid from borderline. Based on our finding this area is far enough from centroid so it has more unique features and in the other hand, it is also far enough from borderline to get it misclassified as another class.

Our method generates synthetic sample based on SMOTE. SMOTE needs pair of samples to create the synthetic sample. At the first ripple, we take the farthest sample from the centroid as the first sample, and then find its nearest neighbor that is closer to centroid as the second sample. Second ripple is starts from the set of second sample as the farthest sample from the centroid and find again its nearest neighbor that is closer to centroid as the pair and repeat the same procedure for the third ripple and on. For each *FARTHEST* sample, we search for the nearest neighbor with *Imbalanced Rate* as the limit number. The nearest set from borderline is becoming the first ripple. Number of ripple can be varying, suppose *R* is the set of the ripple, the sample's distance from each ripple to centroid should satisfy,

$$\left\| (m'_i - m_i) \right\| < \left\| (\text{center}_c - m_i) \right\|$$

For each item in the ripple set, we create the synthetic sample. SMOTE is used to produce synthetic samples of the data set.

$$S_i = m_i + rand(0,1) \times (m_l - m'_i), i = 1,2,...N_r$$

We repeat this procedure for each ripple available. All the variables in this method are vector. With this approach we can see that the new synthetic data is created starting from borderline point and getting near to the centroid for each ripple available.

B. Neural Network with Weighted Loss Function

In this paper, we also propose a weighted softmax layer at the end of the deep neural network architecture. Network architecture that we used in this research fully connected layers with weighted loss layer at the end. We implement two types of network for this research, the first model is as shown in Fig. 4, with 1 convolutional layer for MNIST. For texture and malware test, we are using VGG16 model that already pre-trained using ImageNet for transfer learning purpose. We are using the pre-trained model to give more knowledge to small set like texture and malware. We borrow the idea from Yue [10] to put the weighted loss function at the end of our network. We decide the weights for softmax loss using imbalanced rate with the same formula that already defined in the previous section.

IV. Evaluation and Discussion

We conducted performance evaluations using MNIST handwritten digit data set, CUReT (Columbia-Utrecht) texture data set. Since both of them are not imbalanced, we pick up several classes and reduce the number of samples in those classes significantly, to make them artificially imbalanced. We also conducted a test on imbalance set like Malware data set to evaluate the performance.

A. MNIST Data Set

In this test, we randomly reduce the number of samples in 1, 3, 5, and 7 to 200. We did not do any modification to the rest of the class. Classes with digits 0, 2, 4, 6, 8, and 9 are majority class and others with 200 samples are minority class. By this augmentation, we got the imbalance rate around 30 for each of the minority class. We leave the test set as it is, with MNIST original 10000 test samples in total. Fig. 3 shows the examples of synthetic samples generated using our method.

TABLE I: Result Table for MNIST Data Set after 15th Epoch

Algorithm	Loss	Error Rate	Improvement
Hybrid Ripple-SMOTE	0.6448	3.95%	70.45%
Ripple-SMOTE	0.1597	4.54%	66.04%
SMOTE-ENN	0.1723	4.24%	68.28%
SMOTE-TOMEK	0.1601	4.19%	68.66%
SMOTE	0.3456	9.07%	32.16%
Borderline-SMOTE	0.1909	4.77%	64.32%
Original Imbalanced	0.4984	13.37%	-

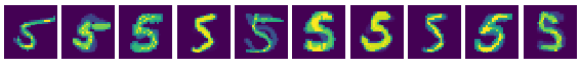


Fig. 3. SMOTE synthetic sample for class 5.

Table I is the classification results for MNIST data set. The results shown in Table I reveals that our hybrid method can improve the performance of imbalanced classification by 70 percent from the original imbalanced set. If we compare it to common oversampling methods like SMOTE and borderline-SMOTE, we improved the performance by 50 percent and 4.83 percent respectively compared to cases only using Ripple-SMOTE. By using our hybrid method reduced the error rate 6.84 percent for SMOTE-ENN and 5.73 percent for SMOTE-TOMEK.

Data shown in Table II reveals that our proposed method can reach 6.4 percent classification error rate on minority

classes, which is better than 8.77 percent by SMOTE-ENN and 14.19 percent by SMOTE-TOMEK. Ripple-SMOTE's performance has good performance on minority classes because it is not just created synthetic samples on the border of minority class (like borderline-SMOTE) but also makes more samples nearer to centroid. By using this approach, we believe that we created synthetic samples that has unique features, because it nears the border of the class, but also better assurance that the sample is still in the class by generating samples based on the "ripple" movement that keeps getting nearer to the centroid of the corresponding class.

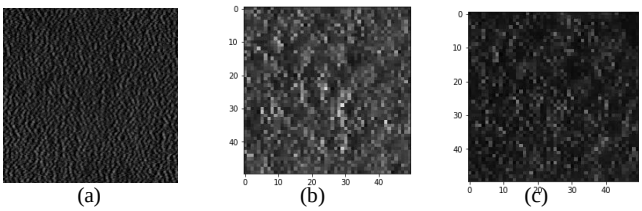
TABLE II: MINORITY CLASSIFICATION ERROR RATE TABLE

Class	Validation Samples	Ripple-SMOTE	SMOTE-ENN	SMOTE-TOMEK
1	1135	1.41%	2.12%	2.03%
3	1010	19.92%	17.63%	19.83%
5	892	26.08%	27.05%	26.88%
7	1028	17.364%	20.18%	19.83%

B. Texture Data Set

We then conducted a performance evaluation using a texture data set, to see if our approach is applicable to floor texture image classification and identification problem [14] [15]. Texture data from CUReT (grey) [16] set consists of 61 classes with 92 samples for each class including test set. We take 20% of the set to be used as the test set. Figure 4 shows the example of texture data set for training samples.

To create the imbalanced problem in this data set, we removed some samples from some classes, and made the imbalanced vary between 1.01 until 2 imbalanced rates for each class. Fig. 4 shows an example of synthetic samples generated for texture data set.



(a) (b) (c)

Fig 4. Synthetic sample generated for texture sample. (a) is the real image from data set class 2; (b) is scaled image for training purpose; (c) is synthetic image created.

TABLE III: RESULT TABLE FOR TEXTURE DATA SET

Algorithm	Loss	Error Rate	Loss (Best)	Error Rate
Hybrid Ripple-SMOTE	0.3932 (42)	8.54%	0.3285	5.97%
Ripple-SMOTE	0.7912 (22)	18.19%	0.5948	15.98%
SMOTE-ENN	0.6829 (28)	16.36%	0.4639	11.78%
SMOTE-Tomek	0.3390 (30)	6.73%	0.3390	6.73%
Original Imbalanced	0.4879 (28)	12.67%	0.5138	10.75%

We evaluate texture a little bit differently from the MNIST case. With small number of samples available, we decided to use transfer learning from pre-trained imagenet model of VGG16. We also used early-stopping in our model. Table III shows the classification performance result. The numbers in brackets show the number of iteration when the training stopped. We set the tolerance of early-stopping to 5. We did not evaluate SMOTE, and borderline-SMOTE because we wanted to focus more on the combination of oversampling and undersampling algorithm's performance.

Rippled-SMOTE improved by 72.6 percent by using the hybrid approach. By using our hybrid method, its best performance improved the imbalance texture classification by 44.5 percent. It also surpassed SMOTE-TOMEK best performance by 11.3 percent. Compared to normal weight loss, our proposed approach in hybrid Ripple-SMOTE starts the learning slowly. In texture case, it reached its peak after 36th epoch compared to other methods that reached the peak performance around 20th epoch. Through this experiment, we could verify that our approach can also be applied to floor image classification and identification problem.

C. Malware Data Set

We conducted a test on Ripple-SMOTE using malware imbalanced set created by Nataraj [17]. This experiment uses the same model with transfer learning as the texture test. Fig. 5 is the structure of a malware image of Dontovo. Nataraj [17] made the data set by converting malware binary to an 8-bit vector then to a grayscale image. Fig. 6 is the binary to grayscale image conversion scheme.

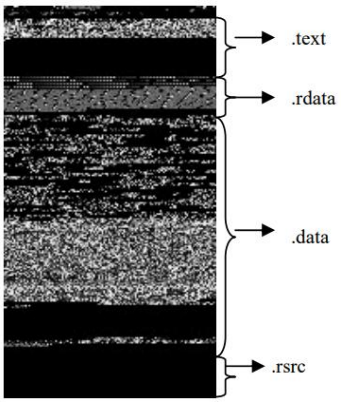


Fig. 5. Various sections of Trojan: Dontovo.A.

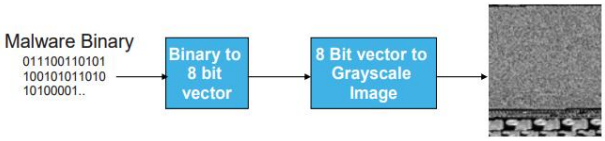


Fig. 6. Malware binary to image conversion scheme.

The malware data set is an imbalanced data set consisting of 9339 images of malware binary that are classified into 25 classes. The biggest class consists of 2949 samples (*Allaple.A*) and the smallest class consists of 80 samples (*Skintrin.N*). This makes the imbalanced rate of those two classes to 36.86. This set originally doesn't have test set, so we randomly chose 30% of the training set as the test set.

TABLE IV: RESULT TABLE FOR MALWARE RATA SET

Algorithm	Loss	Error Rate
Ripple-SMOTE	0.524	15.56%
SMOTE-ENN	2.083	56.71%
SMOTE-TOMEK	0.612	18.63%
SMOTE	0.4334	13.17%
Original Imbalanced	0.5126	16.74%

As we can see from the results shown in Table IV, our approach outperforms the original imbalanced case,

Ripple-SMOTE and SMOTE. Now we analyze the confusion matrix closely, since the average classification accuracy (error rate) can be overly influenced by the performance on majority set because of its large number of samples. We measured the minority performance using *TPR* (*True Positive Rate*). *TP* and *FN* stand for True Positive and False Negative respectively. Table V shows minority sets with less than 200 samples and differences in classification result between Ripple-SMOTE, SMOTE-Tomek and original imbalanced.

$$TPR = \frac{TP}{TP + FN}$$

Table V shows the minority set performance results (TPR) of original, SMOTE, and Ripple-SMOTE cases.

TABLE V: TRUE POSITIVE RATE RESULT TABLE FOR MALWARE DATA SET

Class	Original	SMOTE	Ripple-SMOTE
Adialer.C	0	0	**0.94871795**
Agent.FYI	**1**	0.25	**1**
Aleuron.gen!J	0	0	**0.10909091**
C2LOP.P	0	**0.075**	0
Dialplatform.B	0.9772	0.0172	**0.98360656**
Dontovo.A	**1**	**1**	0.85964912
Lolyda.AA1	**0.9696**	0.9636	0.95454545
Lolyda.AA2	**0.66**	0.5813	0.46551724
Lolyda.AA3	**1**	**1**	0.46153846
Lolyda.AT	0	0	**0.05769231**
Malex.gen!J	0	0	**0.05128205**
Rbot!gen	0	0	**0.06666667**
Skintrim.N	0	0	**0.39215686**
Swizzor.gen!E	0	0	**0.02702703**
Swizzor.gen!I	0	**0.1**	0

From Table V, we can see that the original set has the average accuracy of 29.51% for minority sets, SMOTE 20.98% and Ripple-SMOTE 33.56%. Ripple-SMOTE improved the performance in recognizing minority by 13.72% compared to original set and 59.96% compared to original SMOTE. Our Ripple-SMOTE is not working well with Lolyda classes because of our undersampling might accidentally remove important features from other Lolyda class. Lolyda.AA1 got better result because we augment the sample in Lolyda.AA1 before other Lolyda family. Colored cell in Table V shows that our Ripple-SMOTE performs better at 9 classes and proves that our Ripple-SMOTE performs better in handling synthetic minority set.

V. CONCLUSION AND FUTURE WORK

This paper proposes Ripple-SMOTE as a novel oversampling. We also propose a hybrid method to improve imbalanced set performance by tuning weights for each class. In Ripple-SMOTE (non-hybrid), oversampling starts from borders and the ripple moves toward the centroid, then synthetic samples are generated based on the nearest neighbor sample from the ripples. We strengthen the border by also undersampling the nearest majority data within minority samples.

We have conducted three sets of performance evaluation, MNIST digit image classification, texture image classification, and malware standard dataset, and have shown that our approach significantly improves the performance in imbalanced data cases and outperforms the conventional approaches, especially in handling minority classes.

We notice that linear interpolation method like SMOTE will make noisy images at some extent if the nearest neighbor is not near enough. For example, Fig. 7 is one of the noisy images created using our Ripple-SMOTE.

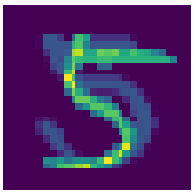


Fig. 7. Noisy synthetic sample generated by Ripple-SMOTE.

For the time being, we are working on improving the synthetic sample quality by applying GAN [18] to make the synthetic samples more realistic and minimize the noise generated by Ripple-SMOTE.

Currently, this research just evaluates images set. As the future works, we are going to improve several points, such as,

1) Improve the undersampling performance
2) Noise cleansing method to improve image quality
3) Improve the robustness for small set
4) Evaluate the robustness on other data set

REFERENCES

[1] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[2] E. Kriminger, J. C. Principe, and C. lakshminaryan, "Nearest neighbor distribution for imbalanced classification," in *Proc. International Joint Conference on Neural Networks*, 2012.

[3] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge Data Engineering*, vol. 21, no. 9, pp. 1263-1284, 2009.

[4] G. M. Weiss, "Mining with rarity: A unifying framework," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7-11, 2004.

[5] Q. Yan, F. Meng, and Q. Sun, An oversampling method based on shapelet extraction for imbalanced time series classification.

[6] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-Sensitive learning of deep feature representations from imbalanced data," *IEEE Transactions on neural Networks and Learning Systems*, 2017.

[7] S. Yue, "Imbalanced malware images classification: a CNN based approach," *arXiv preprint*, 2017.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence research*, 16:321-357, 2002.

[9] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," *Advances in Intelligent Computing*, pp. 878-887, 2005.

[10] C. Bunkhumpornpst, K. Sinapiromsaran, and C. Lurnisap, "Safe-level SMOTE: Safe-level-synthetic minority oversampling technique for handling class imbalanced problem," in *Proc. AKDDM*, 2009.

[11] T. Maciejewski and J. Stefanowski, "Local neighborhood extension of SMOTE for mining data," in *Proc. CIDM.*, Apr. 2011.

[12] G. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 20-29, 2004.

[13] G. Batista, B. Bazzan, M. Monard, "Balancing training data for automated annotation of keywords: A case study," *WOB*, pp. 10-18, 2003.

[14] S. Fujita, T. Fujita, and K. Uchida, "Floor fingerprint verification using a gravity-aware smartphone," in *Proc. the 5th IIAE International Conference on Intelligent Systems and Image Processing*, 2017, pp. 311-318.

[15] K. Uchida and S. Fujita, "Indoor location estimation based on robust floor fingerprint identification," in *Proc. 2017 International Conference on Indoor Positioning and Indoor Navigation*, Sapporo, Japan, 2017.

[16] K. Dana, S. Nayar, B. Ginneken, and J. Koenderink, "Reflectance and texture of real- world surfaces," in *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, 1997, pp. 151–157.
[17] L. Nataraj, S. karthikeyan, G. Jacob, and B. S. Manjunath, "Malware Images: Visualization and automatic classification," in *Proc. International Symposium on Visualization for Cyber Security*, 2011.
[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "General adversarial network," *arXiv*, 2014 .

Rheza Harliman was born in Jakarta, Indonesia, in 1991. He received his bachelor's degree in computer science from Multimedia Nusantara University, Indonesia, in 2013. He worked as an android programmer until 2015 before he moved to Japan. In April 2016 he started to pursue master's degree in Hosei University is going to graduate in March 2018. His research interests include image processing, machine learning, and pattern recognition.

Kaoru Uchida was born in Tokyo, Japan, in 1961. He received his B.E. degree in mathematical engineering and information physics from the University of Tokyo, Japan, in 1984, his M.S. degree in computer science from Stanford University, U.S.A., in 1991, and his Ph.D. degree in Information Sciences from Tohoku University, Japan, in 2003.

He was formerly with NEC Corporation, where he was engaged in research in image processing, pattern recognition and biometric personal identification, and also in design and development of mobile terminals, smart devices and services whereupon. Since April 2014, he has been a professor at the Graduate School of Computer and Information Sciences, Hosei University, in Tokyo, Japan. His current research interests include pattern recognition, biometric personal identification, machine learning, deep learning, and their real-world applications.

Prof. Uchida received the International Standard Development Award from Information Technology Standards Commission of Japan, Information Processing Society of Japan, in 2008, for his contribution as the editor in the development of ISO/IEC19795-3.