World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:15, No:6, 2021

Loss Function Optimization for CNN-Based Fingerprint Anti-Spoofing

Yehjune Heo

***Abstract*—**As biometric systems become widely deployed, the security of identification systems can be easily attacked by various spoof materials. This paper contributes to finding a reliable and practical anti-spoofing method using Convolutional Neural Networks (CNNs) based on the types of loss functions and optimizers. The types of CNNs used in this paper include AlexNet, VGGNet, and ResNet. By using various loss functions including Cross-Entropy, Center Loss, Cosine Proximity, and Hinge Loss, and various loss optimizers which include Adam, SGD, RMSProp, Adadelta, Adagrad, and Nadam, we obtained significant performance changes. We realize that choosing the correct loss function for each model is crucial since different loss functions lead to different errors on the same evaluation. By using a subset of the Livdet 2017 database, we validate our approach to compare the generalization power. It is important to note that we use a subset of LiveDet and the database is the same across all training and testing for each model. This way, we can compare the performance, in terms of generalization, for the unseen data across all different models. The best CNN (AlexNet) with the appropriate loss function and optimizers result in more than 3% of performance gain over the other CNN models with the default loss function and optimizer. In addition to the highest generalization performance, this paper also contains the models with high accuracy associated with parameters and mean average error rates to find the model that consumes the least memory and computation time for training and testing. Although AlexNet has less complexity over other CNN models, it is proven to be very efficient. For practical anti-spoofing systems, the deployed version should use a small amount of memory and should run very fast with high anti-spoofing performance. For our deployed version on smartphones, additional processing steps, such as quantization and pruning algorithms, have been applied in our final model.

***Keywords*—**Anti-spoofing, CNN, fingerprint recognition, loss function, optimizer.

I. Introduction

FINGERPRINT recognition is a strong and useful biometric system because of the uniqueness of fingerprints. It can be used as a secure and easy way to identify humans. However, spoof fingerprints violate the secureness of fingerprint recognition systems. For fingerprint recognition systems to be more secure and widely implemented, the systems need to have liveness detection schemes that can detect spoof fingerprints from the live ones. One of the methods in liveness detection techniques in hardware [1] is the detection of blood pressure. Atsushi et al. [2] demonstrated the correlation between live fingerprints and the blood movement inside the skin layer. The problem with these techniques is that it is expensive to be used widely. In order to provide secure and economical fingerprint recognition systems, the use of CNN is widely investigated. It is possible to train CNN in order to distinguish between fake and live fingerprints. Researchers have been trying to find the most accurate CNN that can distinguish between live and spoof fingerprints. See [16], [27], [28] for various attempts on detecting spoof fingerprints. Eunsoo et al. [3] demonstrated a CNN for fingerprint liveness detection using the Gram module. The model had an error rate of 2.61%, which shows the potential of using CNN for fingerprint liveness detection.

AlexNet [12], VGGNet [14], and ResNet [15] are used in this paper's experiment. The three CNNs are tested with various types of loss functions and optimizers. The loss functions used in the experiments include Cross-Entropy, Center Loss, Cosine Proximity, and Hinge Loss. These 3 are the main loss functions that are used for classification purposes. The optimizers used in the experiments are Adam, SGD, RMSprop, Adadelta, Adagrad, and Nadam. Among the CNN models with high accuracy, the parameters and mean average error rates are also obtained. The use of parameters is to calculate each model's memory usage, and the mean average rate (mae) is used to find the time taken to predict and output a result. Comparing the models' efficiency on the database associated with the accuracy, memory usage, and the amount of time for output in the aspect of different loss functions and optimizers is the main purpose of this paper.

The paper is organized as follows. Section II contains the necessary background information. Section II *A* includes the necessity of detecting spoof fingerprints, and Section II *B* explains the difference between live and spoof fingerprints, and Section II *C* types of CNN that will be used in this experiment. The types of loss functions and optimizers that will be used in this experiment are explained in Section III with mathematics and description. In Section IV, the experiments will be demonstrated by finding the proper model and reinforcing that model. The results will be in Section V and the conclusion will be followed in Section VI.

II. Background

A. Spoofing Attacks

There have been many attacks by using fake fingerprints. See [4]-[8] for various spoof attempts. Philip et al. [8] demonstrated an algorithm, DeepMasterPrints, that used deep learning to create a spoof fingerprint and tested it on a database. The spoof fingerprint matched 78% of the real one in the lowest security level of 1%. Philip also said that "the underlying method is likely to have broad applications in fingerprint security as well as fingerprint synthesis." The use of deep learning to make

Yehjune Heo is with Korea International School, Korea, Republic Of (e-mail: yehjune111@gmail.com).

high-quality spoof fingerprints will lead to a more serious problem to solve.

Recently, there was a case where famous smartphones have been easily unlocked by using just silicone phone cases [6]. These problems indicate that still, many fingerprint recognition systems have difficulty in classifying between live and spoof fingerprints, although there have been significant improvements in this research.

Implementing liveness detection algorithms into the systems is a probable method only for systems that require strong security. In order to implement fingerprint recognition for everyone to use in important cases, like verifying a bank account, the system needs to be more reliable and cheaper to implement. The use of CNN models can be a reliable method in distinguishing spoof and live fingerprints, and it is also cheaper to implement than hardware anti-spoofing systems.

B. Difference Between Live and Spoof Fingerprints

The difference between live and spoof fingerprints can be found from the scanned images from the sensor.

(a) (b)

Fig. 1 Image of a spoof fingerprint made out of gelatine (a) and a live fingerprint (b)

Fig. 1 shows the difference between spoof (a) and live (b) fingerprints. Live fingerprints, when contacted by the sensor, show natural pattern distribution over the fingerprint and natural movements. Live fingerprint images also show the distribution of pores which are small to be made by spoof fingerprints. The image of a live fingerprint in Fig. 1 shows the overall gray region, specific ridges, and distribution of pores. Spoof fingerprints, on the other hand, show unnaturally distributed patterns, dark on some regions, and abnormal ridges. Spoof fingerprints also show unnatural boundary, white or black blob, and abnormally projected histogram. The image of a live fingerprint in Fig. 1 shows random white and black regions, white spots at the edge, abnormal ridges, and no pores. Pores are too small to be found in spoof fingerprints, so it is a distinct feature from live and spoof fingerprints. Pores can be used in high-quality sensors to classify between spoof and live fingerprints. Anil et al. [9] demonstrated a fingerprint recognition algorithm using the detection of pores. This algorithm can be useful in detecting spoof and live fingerprints. However, the difference between live and spoof fingerprints are unrecognizable in many current fingerprint recognition systems. There needs to be a more efficient way to distinguish live and spoof fingerprints.

C. Types of CNN

CNN is a deep learning neural network that is usually used in image classification. In 1988 Kunihiko et al. [10] was the first to propose a hierarchy of neural networks capable of visual pattern recognition. The paper shows the basic structure of DNN (Deep Neural Networks) and the use of neocognitron as a universal pattern-recognizer. In 1998, LeCun et al. [11] represented LeNet, which held the basic architecture of CNN in 1998. LeNet contains the basic convolution layers and pooling layers, and it also uses backpropagation.

There are various types of CNN that can be used for classification purposes. Krizhevsky et al. [12] demonstrated AlexNet, a neural network that achieved first place in the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) [13] with the lowest error rate. AlexNet was not the first CNN, but it was the first useful CNN architecture, and the dropout technique used by AlexNet is the basis for every CNN used in the present days.

Karen et al. [14] demonstrated VGGNet in ILSVRC 2014 and achieved first place in the localization task. VGGNet is an improved version of AlexNet, that uses large kernel filters followed by multiple 3x3 kernels. However, ResNet developers, Kaiming et al. [15], suggested a graph about gradient vanishing in VGGNet, a degradation problem. As the network depth increases, accuracy gets saturated and then degrades rapidly. The solution they suggested was to make shortcuts for the gradients. The shortcut connections turn the network into its counterpart residual version allowing the layers to directly use the data when the input and output are the same dimensions. DenseNet is an improved version of ResNet, and it was demonstrated by Huang et al. [16]. DenseNet has shortcuts for every layer so all of the layers are connected. DenseNet is very similar to ResNet except for an equation that sums the input instead of concatenating it, which leads to a substantially different behavior.

The types of CNN introduced above are the CNNs that had high performance in the ILSVRC. They are all useful CNNs for classification, however, they all have different architectures. To classify between spoof and live fingerprints, the CNN that has the highest accuracy in fingerprint databases needs to be investigated.

III. Loss Functions and Optimization

Finding the accurate loss functions and optimizers for the model is important to perform with high accuracy. This section introduces the types and basics of the loss functions and optimizers that this experiment uses.

A. Loss Function

CNN trains by optimizing parameters through loss functions. The loss function compares the accuracy from the training database to a validation database. Choosing the correct loss function for the model is crucial since different loss functions lead to different errors on the same evaluation.

Table I shows four loss functions used in this paper with

mathematics and description. The loss functions are Cross Entropy, Center Loss, Cosine Proximity, which are the main loss functions that are used for classification purposes [19]. The four loss functions will be collaborated with various optimizers to reinforce the performance of the models.

TABLE I
TABLE OF LOSS FUNCTIONS USED IN THIS EXPERIMENT

Loss Function	Mathematics	Description
Cross Entropy	$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \times \log(p(y_i)) + (1-y) \times \log(1-p(y_i))$	Cross entropy functions [17], [30], [31] measures the difference between two or more probability distributions. The equation uses a logarithmic equation to find out the difference between the predicted values and actual values.
Center Loss	$L_c = \frac{1}{2}\sum_{i=1}^{m} \|\|x_i - c_{yi}\|\|_2^2$	Center loss function [18], [32], [33] finds the center of the dataset by reducing the distance from each data. The equation characterizes the intra-class variations and takes the entire training set into account and averages the features.
Cosine Proximity	$\cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$	Cosine Proximity function [34], [35] measures the similarity of two vectors. The equations calculate the cosine of the angle between them, which is the similarity of the two vectors.
Hinge Loss	$l(y) = \max(0, 1 - t \times y)$	Hinge Loss function [36] measures the maximum margin between data points. Hinge Loss function is a convex function, which cannot be used with most of the usual convex optimizers.

Table of loss functions that are included in the experiment. The table also includes the mathematics and description of each loss function.

TABLE II
TABLE OF OPTIMIZERS USED IN THIS EXPERIMENT

Loss Function	Mathematics	Description
Adam	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \varepsilon}} \times m_t$	Adam [20], [21] optimizer calculates an exponential moving average of the gradient and the squared gradient, and the parameters control the decay rates of these moving averages.
SGD	$\theta = \theta - \eta \times \nabla_\theta J(\theta; x^i, y^i)$	SGD [20], [22] optimizer uses gradient descent on a random point on the entire dataset. The optimizer updates its parameter for each training example and label. It also reduces redundant computations for large datasets.
RMSProp	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} \times g_t$	RMSProp [20], [23] optimizer normalizes the gradients by an exponential moving average of the magnitude of the gradient for each parameter. RMSprop is developed to resolve Adagrad's radically diminishing learning rate problem.
Adadelta	$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} \times g_t$	Adadelta [20], [24] optimizer is an extension from Adagrad optimizer that reduces its aggressively decreasing learning rate. Instead of accumulating all past gradients, the optimizer adapts the learning rate based on the moving window of gradient updates.
Adagrad	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \times g_t$	Adagrad [20], [25] optimizer adapts the learning rate to the parameters, which is more parameter updates for infrequent features and less parameter updates for frequently occurring features.
Nadam	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \varepsilon}} \times (\beta_1 m_t + \frac{(1-\beta_1) g_t}{1-\beta_1^t})$	Nadam [20], [26] optimizer is the combination of Adam and NAG. The learning rate is accelerated by summing up the exponential decay of previous and current gradient's moving averages.

Table of optimizers that are included in the experiment. The table also includes the mathematics and description of each optimizer.

B. Optimizer

Optimizers are used to minimize the error between the training database and the validation database. After the loss function calculates the errors, the optimizer calculates the gradient, which is the derivative of the loss function, and this gradient is used to adjust the weights of the model for better performance. Different optimizers lead to different adjustments of the weights of the model, which can be crucial on the accuracy.

Table II shows six optimizers used in this paper with mathematics and description. The optimizers are Adam, SGD, RMSprop, Adadelta, Adagrad, and Nadam, which are the main optimizers that are used for CNN training.

IV. METHOD

The fingerprint database is from Livdet 2017 database [29]. The models in this paper use 1000 images from the datasets from 2009 to 2017. It is important to note that we use a subset of LiveDet and the database is the same across all training and testing for each model. This way, we can compare the performance, in terms of generalization, for the unseen data across all different models.

AlexNet, VGGNet, and ResNet were also reinforced to be the most accurate models for detecting spoof fingerprints.

The paper's experiment AlexNet has 26 layers, no data augmentation, pool size of (2,2), kernel size of (3,3), average pooling type, dropout rate of 0.5, zero padding of (1,1), and activation functions of relu and softmax. VGGNet has 22 layers, no data augmentation, pool size of (2,2), kernel size of (2,2), max pooling type, dropout rate of 0.5, zero padding of (1,1), and activation functions of relu and softmax. ResNet, no data augmentation, pool size of (2,2), kernel size of (3,3), max pooling type, dropout rate of 0.5, zero padding of (1,1), and activation functions of softmax.

The models were trained each time with one of the loss functions and one of the optimizers together, and the accuracies were recorded.

V. RESULT

The accuracy, mean average error rate, and the parameters are compared to find the most accurate CNN model based on loss functions and optimizers for the anti-spoofing fingerprint

database.

TABLE III
ALEXNET LOSS FUNCTION OPTIMIZATION ACCURACY

	Adam	SGD	RMSProp	Adadelta	Adagrad	Nadam
Cross Entropy	94.05%	94.76%	91.43%	93.57%	94.76%	95.24%
Center Loss	84.29%	93.33%	90.24%	92.14%	88.81%	91.43%
Cosine Proximity	90.95%	93.33%	95%	93.57%	94.53%	94.52%
Hinge Loss	93.57%	94.05%	91.92%	95%	94.52%	94.52%

Table of AlexNet's accuracies of various loss functions and optimizers on the anti-spoofing fingerprint database.

TABLE IV
VGGNET LOSS FUNCTION OPTIMIZATION ACCURACY

	Adam	SGD	RMSProp	Adadelta	Adagrad	Nadam
Cross Entropy	93.81%	92.14%	90.95%	92.62%	50.48%	50.48%
Center Loss	49.52%	50.48%	50.48%	90.24%	50.48%	50.48%
Cosine Proximity	93.13%	87%	92.62%	94.76%	50.48%	50.48%
Hinge Loss	50.48%	82.86%	50.48%	50.48%	50.48%	50.48%

Table of VGGNet's accuracies of various loss functions and optimizers on the anti-spoofing fingerprint database.

TABLE V
VGGNET LOSS FUNCTION OPTIMIZATION ACCURACY

	Adam	SGD	RMSProp	Adadelta	Adagrad	Nadam
Cross Entropy	92.76%	91.33%	89.87%	91.05%	86.86%	89.76%
Center Loss	50.48%	49.05%	49.58%	50.42%	50.16%	50.49%
Cosine Proximity	79.05%	90%	88.33%	90.24%	89.05%	88.57%
Hinge Loss	91.19%	92.14%	88.1%	90.71%	90.71%	84.05%

Table of ResNet's accuracies of various loss functions and optimizers on the anti-spoofing fingerprint database.

Table III shows that AlexNet performs high on the database with most of the loss functions and optimizers. 2,271,194 parameters were trained for AlexNet. AlexNet with Cross Entropy loss function and Nadam optimizer performed an accuracy of 95.24%, which is the highest in this experiment and with a mean average error rate of 0.05. Also, Cosine Proximity with RMSprop and Hinge Loss with Adadelta performed high accuracy of 95% with a mean average error rate of 0.06. Overall, AlexNet with Cross Entropy loss function performs a high average accuracy of 93.97%. Using Nadam as the optimizer performs the highest average accuracy of 93.93%. From the three models, disregarding the models that failed to train, AlexNet performed the highest average accuracy of 92.9%.

Table IV shows that VGGNet performs high in some models, but some models fail to train. 19,110,162 parameters were trained for VGGNet. VGGNet with Cosine Proximity loss function and Adadelta optimizer performed the highest among VGGNet models with an accuracy of 94.76% and with a mean average error rate of 0.07. VGGNet with Cosine Proximity and Adam, Cross Entropy and Adam performed high with an accuracy of higher than 93% and a mean average error rate of 0.08. For VGGNet, Center Loss and Hinge Loss functions were not able to train the model and Adagrad, Nadam optimizers also did not work. Disregarding some of the models that failed to train, VGGNet has an average accuracy of 91.01%.

Table V shows that ResNet genuinely does not perform with a high accuracy on the database. 11,180,674 parameters were trained for ResNet. ResNet with Cross Entropy loss function and Adam optimizer performed the highest among ResNet models with an accuracy of 92.76% and with a mean average error rate of 0.12. Center Loss was not able to train for ResNet and other loss functions with optimizers performed with an accuracy of lower than 92%. Disregarding some of the models that failed to train, ResNet has an average accuracy of 91.01%.

In the aspect of loss functions and optimizers, Center Loss function did not perform well with VGGNet and ResNet on the anti-spoofing fingerprint database. Hinge Loss function also did not perform well with VGGNet on the database. Cross Entropy loss function performed the highest with an average accuracy of 92.19%. Adadelta optimizer performed the highest among optimizers with an average accuracy of 92.39%. SGD, RMSprop, and Nadam performed really high with AlexNet on the database. However, Nadam and Adagrad did not work with VGGNet and failed to train.

VI. CONCLUSION

This paper studies the use of loss functions and optimizers for CNNs for fingerprint anti-spoofing. Among the models, AlexNet with Cross Entropy loss function and Nadam optimizer performed the highest accuracy, with an accuracy of 95.24%. The model performed with a mean average error rate of 0.05 and trained 2,271,194 parameters. The low computation time and low memory usage reinforces the efficiency of the model. Although AlexNet has less complexity over other CNN models it is proven to be very efficient. For practical anti-spoofing systems, the deployed version should use a small amount of memory and should run very fast with high anti-spoofing performance. For practical anti-spoofing systems, the deployed version should use a small amount of memory and should run very fast with high anti-spoofing performance. For

our deployed version on smartphones, additional processing steps, such as quantization and pruning algorithms, have been applied in our final model.

REFERENCES

[1] T. van der Putte and J. Keuning, "Biometrical Fingerprint Recognition: Don't Get Your Fingers Burned" in *Smart Card Research and Advanced Applications*, pp. 289-306, 2000.

[2] A. Hori and I. Fujieda, "Study on Blood Movement During Fingerprint Input Actions", *International Journal of Optomechatronics*, Vol. 2, pp.390-400, 2008.

[3] E. Park, X. Cui, W. Kim, and Haki. Kim, "End-to-End Fingerprints Liveness Detection using Convolutional Networks with Gram module", *ArXiv*, 2018.

[4] E. Marasco and A. Ross, "A Survey on Antispoofing Schemes for Fingerprint Recognition Systems" in *ACM Comput. Surv.*, Vol. 27, pp. 28:1-28:36, 2014.

[5] Arstechnica, "Brazilian docs fool biometrics scanners with bag full of fake fingers", Available at: https://arstechnica.com, 2013. Accessed on: 9 June 2020.

[6] Arstechnica, "Anyone can fingerprint unlock a Galaxy S10--just grab a clear phone case", Available at: https://arstechnica.com, 2019. Accessed on: 9 June 2020.

[7] T. Matsumoto, H. Matsumoto, K. Tamada, and S. Hoshino, "Impact of artificial "gummy" fingers on fingerprint systems", in *Proceedings of SPIE - The International Society for Optical Engineering*, Vol. 4677, 2002.

[8] P. Bontrager, A. Roy, J. Togelius, N. Memon and A. Ross, "DeepMasterPrints: Generating MasterPrints for Dictionary Attacks via Latent Variable Evolution" in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1-9, 2018.

[9] A. Jain, Y. Chen, and M. Demirkus," Pores and Ridges: Fingerprint Matching Using Level 3 Features" in *18th International Conference on Pattern Recognition (ICPR'06)*, pp. 477-480, 2006.

[10] K. Fukushima, "Neocognitron A Hierarchical Neural Network Capable of Visual Pattern Recognition", *Neural Networks*, Vol. 1, pp. 119-130, 1988.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition" in *Proceedings of the IEEE.*, Vol. 86, pp. 2278-2324, 1998.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Learning Convolutional Neural Networks" in *Advances in neural information processing systems 25(2)*, 2012.

[13] ImageNet, "ImageNet Large Scale Visual Recognition Challenge", Available at: http://www.image-net.org/challenges/LSVRC/, 2015. Accessed on: 9 June 2020.

[14] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *ArXiv*, 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition" in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, 2016.

[16] E. Park, W. Kim, Q. Li, J. Lim, and H. Kim, "Fingerprint Liveness Detection Using CNN Features of Random Sample Patches" in *2016 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pp. 1-4, 2016.

[17] P-T de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A Tutorial on the Cross-Entropy Method", *Annals of Operations Research*, Vol. 134, pp. 19-67, 2005.

[18] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A Discriminative Feature Feature Learning Approach for Deep Face Recognition" in *Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016*, Vol. 9911, pp. 499-515, 2016.

[19] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri, "Are Loss Functions All the Same?", *Neural Computation*, Vol. 16, Issue. 5, pp. 1063-1076, 2003.

[20] R. Sebastian, "An overview of gradient descent optimization algorithms", *ArXiv*, Vol. arXiv:1609.04747v2 (cs.LG), 2016.

[21] D. P. Kingma and J. L. Ba, "Adam: A Method For Stochastic Optimization", *CoRR*, Vol. arXiv:1412.6980v9 (cs.LG), 2014.

[22] R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin and P. Richtarik, "SGD: General Analysis and Improved Rates", *ArXiv*, Vol. arXiv:1901.09401v4 (cs.LG), 2019.

[23] M. C. Mukkamala and M. Hein, "Variants of RMSProp and Adagrad with Logarithmic Regret Bounds" in *17th International Conference on Machine Learning (ICML)*, 2017.

[24] M. D. Zeiler, "ADADELTA: An adaptive learning rate method", *ArXiv*, Vol. arXiv:1212.5701v1 (cs.LG), 2012.

[25] J. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization" in *Journal of Machine Learning Research*, Vol. 12, pp. 2121-2159, 2011.

[26] T. Dozat, "Incorporating Nesterov Momentum into Adam" in *16th International Conference on Learning Representations (ICLR)*, 2016.

[27] R. F. Nogueira, R. de A. Lotufo, and R. C. Machado, "Fingerprint Liveness Detection Using Convolutional Neural Networks". in *IEEE Transactions on Information Forensics and Security*, Vol.11, No.6, pp. 1206-1213, 2016.

[28] D. Uliyan, S. Sadeghi, and H. A. Jalab, "Anti-spoofing method for fingerprint recognition using patch based deep learning machine" in *Engineering Science and Technology, an International Journal*, 2019.

[29] LivDet, "LivDet Databases", Available at: livdet.org, Accessed on 2009.

[30] D. P. Kroese, R. Y. Rubinstein, S. Porotsky, A. L. D. Ltd, and T. Taimre, "Cross-Entropy Method" in *Encyclopedia of Operations Research and Management Sciences*, 3rd ed, Springer-Verlang, 2012.

[31] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer. "Chapter 3. The Cross-Entropy Method for Optimization" in *Handbook of Statistics*, Vol. 31, pp. 35-59, 2013.

[32] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A Comprehensive Study on Center Loss for Deep Face Recognition" in *Int J Comput Vis* 127, pp. 668-683, 2019.

[33] P. Ghosh and L. S. Davis, "Understanding Center Loss Based Network for Image Retrieval with Few Training Data" in *2018 European Conference on Computer Vision (ECCV)*, pp. 717-722, 2019.

[34] E. Garcia, "Cosine Similarity Tutorial" in *ResearchGate*, 2015.

[35] F. Rahutomo, T. Kitasuka, and M. Aritsugi, "Semantic Cosine Similarity" in *7th Interntaional Student Conference on Advanced Science and Technology (ICAST)*, 2019.

[36] C. Gentile and M. Warmuth, "Linear Hinge Loss and Average Margin" in *Advances in Neural Information Processing Systems (NIPS)*, Vol .11, pp. 225-231, 1998.