Comparisons of Deep Learning Algorithms for MNIST in Real-Time Environment

Akmaljon Palvanov and Young Im Cho

Department of Computer Engineering, Gachon University, Seongnam, Korea

Abstract

Recognizing handwritten digits was challenging task in a couple of years ago. Thanks to machine learning algorithms, today, the issue has solved but those algorithms require much time to train and to recognize digits. Thus, using one of those algorithms to an application that works in real-time, is complex. Notwithstanding use of a trained model, if the model uses deep neural networks it requires much more time to make a prediction and becomes more complicated as well as memory usage also increases. It leads real-time application to delay and to work slowly even using trained model. A memory usage is also essential as using smaller memory of trained models works considerable faster comparing to models with huge pre-processed memory. For this work, we implemented four models on the basis of unlike algorithms which are capsule network, deep residual learning model, convolutional neural network and multinomial logistic regression to recognize handwritten digits. These models have unlike structure and they have showed a great results on MNIST before so we aim to compare them in real-time environment. The dataset MNIST seems most suitable for this work since it is popular in the field and basically used in many state-of-the-art algorithms beyond those models mentioned above. We purpose revealing most suitable algorithm to recognize handwritten digits in real-time environment. Also, we give comparisons of train and evaluation time, memory usage and other essential indexes of all four models.

Keywords: Capsule networks, Dynamic routing, Residual learning, CNN, Logistic regression

1. Introduction

In recent years, machine learning models have been grown significantly and still escalating. Especially, deep neural networks have achieved great success in various applications [1, 2], particularly in tasks involving visual information. There have introduced many state-of-the-art models in the field that perform dissimilar tasks with a high accuracy and very effectively. Developers, today, are able to use those models to apply for other tasks say, recognizing hundreds of millions of images or classifying huge datasets consisted of high dimensions.

Convolutional neural networks (CNNs) are very convenient and are widely used in classification, localization, detection, and more other tasks. Using convolution operation we can build and train new models but achieving high accuracy in image recognition is complex as accurately prediction depends on several factors. Those factors might be dataset which is being used, network architecture and so on. The depth of the network also affects to gain better performance. Thereby many deep CNNs and deep neural network models are being

Received: Mar. 17, 2018

Revised : Jun. 9, 2018

Accepted: Jun. 12, 2018

Correspondence to: Young Im Cho

(yicho@gachon.ac.kr)

©The Korean Institute of Intelligent Systems

©This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

preferably used.

However, instead of using convolution operation logistic regression or multinomial logistic regression (MLR) can also be used. Logistic regression is a valuable tool for analyzing information that incorporates categorical response factors, such as presence or absence of a species in quadrats, presence of disease or harm to seedlings [3].

Residual learning techniques is, however, one of the deepest models unlike others, it consists of more than hundred layers, and inputs are concatenated after some activation function then propagate forward. That gives to the network an ability of deeper learning all features of each input. Among all models new capsule network is last state-of-the-art model in the task of recognizing hand written digits using MNIST dataset and also have shown good performance on CIFAR-10 dataset achieving 10.6% error rate [4]. Since network architecture is totally unlike to traditional models mentioned above thereby we will explain it later on in detail.

As previously mentioned, we use the MNIST dataset to perform the experiment of this study. The MNIST dataset is well known in introducing machine learning for several reasons. One of them could be that is complex to classify due to the irregularities in handwriting, however, also easy enough because there are no other irregularities. All digits are oriented correctly and there is no clamor around the handwriting itself [5].

For this work we implemented four different models and feed them with the same inputs so we can see superiorities and drawbacks of networks. Also, we propose applying those models to real-time application and to evaluate their time and memory usage in order to further increase app's immediateness, to do this we created Java based GUI application to test it with new inputs that can be drawn by users, it simultaneously predicts results and gives answer based on one of the trained models and can be seen which model works faster and its memory usage in the hardware these two issues are very critical issues especially for real-time applications, as well as other features.

Firstly, we will introduce some models in Section 2 as a related work, then we will explain our theories in Section 3. We will explain our experiments in Section 4, and finally we will conclude our research in Section 5.

2. Related Works

2.1 Neural Network with Multinomial Logistic Regression (MLR)

MLR is a logistic regression that is designed to find a solution

for multiclass classification tasks. In other words, MLR is a model which predicts probabilities of various possible outcomes of a categorically distributed dependent element, given a set of independent elements [6]. It can be used when the dependent variable is nominal and falls into a class among many classes (e.g. the MNIST dataset has nine classes).

MLR classifier [5, 7] is commonly applied as an alternative to naive Bayes classifier since it does not assume statistical independence of the unsystematic features, it is common and requires a little time to learn but it becomes unhurried as using a large number of classes to learn.

2.2 CNN

CNN is commonly used to recognize visual patterns directly from pixel images with variability and attempts to learn the relationship between the inputs and the outputs, it also stores the learned experience in their filter weights. The role of the nonlinear activation function just after the convolution operation is one of the challenge for comprehending CNN [8, 9]. The most ascendant work gained using CNN is introduced by Krizhevsky et al. [10].

Authors of [10] used CNN for the ImageNet classification competition. Diverse other methods and techniques were proposed later on increase CNN performance as revealed in [11–14]. A little while back, wide verity of works have been proposed to enhance image classification accuracy results using various techniques. So, suggested approaches are proposed for different applications such image recognition [15, 16], object detection [17, 18], segmentation [19, 20] and other tasks [21–23]. Studies and practices show that there are some issues regarding convolutional nets. Regarding them we will discuss later on.

2.3 Deep Residual Learning or ResNet

ResNet is one of the deepest model in the field which has been introduced in [24], it is considerable deep comparing to others models proposed previously. One of the similarity of the ResNet to other models that mentioned above is using convolution operation, but structure and depth of the network is very different.

Similar to [24], in image recognition challenge, is [25] proposing a representation that encodes by the residual vectors with respect to a dictionary. Also [26] is possible be formulated as a probabilistic version [25]. The likely method used in ResNet, namely shortcut connections, has been practiced and studied for a long time [26]. An initial study of training multi-layer

perceptron is to adding linear layers connected from the each network inputs to the outputs [27].

Concurrent with [24], it is proposed shortcut connections with gating functions [28–30]. In [29, 30] networks have not showed accuracy achieves with exceedingly increased depth (e.g., over 100 layers).

2.4 Capsule Networks

Capsule network is last state-of-the-art model on MNIST dataset and also demonstrated high accuracy on CIFAR-10 dataset too. To be more prices, a capsule can be considered as a group of neurons. Activity vector of capsule exposes instantiation parameters of a particular entity type. They might be a part of a whole object or an object whole. In other word, computer graphics rendering process performs a task that takes a whole object and using transformation matrix converts that object into a pose of the object's part but the authors of [4] tried to do a reverse process. They wanted their network to take a pose of the part and using inverse matrix converts it into a pose of a whole object.

Also, position of any part of the inputs are attentively learned by this models. Since other models output the same result notwithstanding small change in orientation or in position. Capsule nets does not. For the network orientation and position of the inputs matters. For instance, CNN models use subsampling or pooling operation as a result position of the input's parts are lost and much percentage of the whole input is just removed because of pooling operation.

To do this, filters are utilized. That is not good representation of the human vision system as based on idea of the authors of [4], human's brain has an inbuilt mechanism working with low level visual data. Whereas CNN uses filters to extract high level information from low level visual data and does not use any routing mechanism like capsule nets, we will mention about it in next paragraph.

Routing visual data like human brain does and being translation equivariance gives the model an ability to generalize data using small amount of data for training. CNN and other models are translation invariant so they need to be fed with more data for generalizing, in other words, other models require more data in contrast to capsule net. Capsules give those mentioned opportunities since capsule outputs a vector whereas neuron outputs a value. The length of the activity vector is used for representing the probability that the entity exists and its orientation for representing the instantiation parameters.

From [4], it can be seen that multi-layer CapsNet reaches state-of-the-art performance on MNIST and is better than CNNs in the task of recognizing highly-overlapping digits. They use routing-by-agreement mechanism, the mechanism that lower-level capsule sends its output to higher-level capsules whose activity vector has a bigger scalar product with the prediction which is coming from the lower-level capsule, to gain these results.

3. Methods

3.1 Neural Network Using Logistic Regression

Our method is based on [31] we implemented and tested the model using just logistic regression and TensorFlow. Regression methods have already become an integral element of analysis, describe the relationship between a response variable and one more explanatory variables. It is often the case that the outcome variable is discrete, taking on two or more possible values [4]. According to [32] the conditional probability of a class k given the input x has distribution

$$p_k(x; w) = \frac{\exp(W_k^T x)}{1 + \sum_{i=1}^{K-1} \exp(W_i^T x)},$$

$$p_K(x; w) = \frac{1}{1 + \sum_{i=1}^{K-1} \exp(W_i^T x)}, \quad (1)$$

where k runs from 1 to $K - 1$ (K - a number of classes). The conditional probabilities Eq. (1) sum up to 1 therefore the model was reported by $K - 1$ log-odds, in [32] used maximum likelihood method Eq. (2) to fit the model

$$\log \frac{p_k(x; w)}{p_K(x; w)} = w_k^T, \quad k \in \{1, \dots, K - 1\}. \quad (2)$$

We created logistic regression model and trained it using MNIST. After training process we got our pre-trained model that can be used in real-time application. While training process we evaluated time and memory use in hardware. We get pretty similar results to in accuracy and loss. However, other approaches do not detailed inform about time spent and memory usage of the model. There are no quiet clear comparison results with other models that how fast the model can work and predict handwritten digit in real-time application. So we give those comparison results in Table 1.

Figure 1. A bottleneck building block.

3.2 CNN

In this experiment we built models consists of two hidden layers followed by two fully connected layers. After each activation function we use 2×2 max-pooling operating, for activation function we choose TensorFlow's ReLU operation. While training a dropout operation is also used so the network not to suffer from overfitting. Batch size for all four models is equal to 50. Train results are shown in experiments section.

3.3 ResNet

As described in [24] residual network is different from a plain network with its shortcut connections. Unlike plain network we added shortcut connection to each three pair layers 1×1 , 3×3 and 1×1 filters instead of two layers and use bottleneck building block. In this work we use 34-layer residual network. Our residual network model follows as [24] and consists of 34 parameter layers.

The desire for using building block Figure 1 is that while training a signal is necessary for changing weights that arises from the end of the network by comparing with ground-truth. Increasing depth leads the prediction to become small at the initial layers means that earlier layers are learned almost negligibly and this is called vanishing gradient. Thus, this building block can be great solution to avoid this problem in deeper networks, as proved in [24]. Also, it can share parameters better thanks to having shortcut connections.

3.4 Capsule Network

We implemented our CapsNet model based on [4]. Architecture of the model described in Figure 2, as we mentioned above this model tries to perform inverse graphics. The model includes three simple layers followed by fully connected layers. Initial

layer is a common convolution layer that takes activation vectors and reshapes them using squashing function.

We use ReLU as a transfer function and the squashing function for non-linearity is given in Eq. (3). We added tiny change to the squashing function given in [2], it is shown in Eq. (4), and since v_j output of a capsule j and s_j might share 0 vector. As a consequence the function which computes the norm of vector Eq. (4) returns undefined result if there is or are 0 vector(s). Additionally, to avoid dividing by 0 we in $\frac{s_j}{\|s_j\|}$, we add small value ($\gamma = 1e - 6$) and use $\sqrt{\sum s_j^2 + \gamma}$ instead of $\|s_j\|$.

$$v_j = \frac{\|s\|^2}{1 + \|s\|^2} \frac{s_j}{\sqrt{\sum s_j^2 + \gamma}}, \quad (3)$$

$$v_j = \frac{\|s\|^2}{1 + \|s\|^2} \frac{s_j}{\|s_j\|}. \quad (4)$$

A dimension of the activation vector can be more than one. Each activation vector's length represent estimated probability of presence in other words it represents dissimilar features of the input image, for instance, it might be orientation of an image part, thickness of a line or shape etc. After reshaping and squashing we will send them to primary capsules. Based on [4] routing by agreement algorithm works between only primary and digital capsule layers.

Capsules try to send information/output vectors to the capsules above them. As a regularization is used reconstruction loss. While training, inputs are encoded and all digit capsules are masked out except a correct digit capsule's activity vector, it is shown in Figure 2 with the name V5. Last three layers are fully connected layers and their main role in there is a decoding. After decoding we get a reconstructed digit [4]. This model does not use pooling operation as network will loss all information concerning position of each part of the input if pooling is used.

This superiority (not using pooling) is good especially when we are tackling with object detection and any segmentation tasks. Only for image classification, convolution plus pooling combination is good however CapsNet is good at all three tasks. Not only an accurately classifying but also time allocating for training process, working in real-time environment and memory occupying in the storage are also superior comparing to reminder models implemented for MNIST.

On the other hand, CapNets have some drawbacks, say, they yet cannot reconstruct high resolution images that makes the it difficult to applying the models to huge datasets consists of

Figure 2. A network architecture for CapsNet, consists of three layers.

images with high resolutions. In addition to this if images are colored and resolution is higher it takes much time to training because of the loop in routing by agreement [4]. As a human vision system has been observed in [33, 34], CapNets also have a disability namely “visual crowding” means that they cannot see two very close identical objects. In our future works we will focus on these issues and try to apply to bigger datasets.

4. Experiments

To evaluate our models we built Java based GUI application so we can evaluate the models with our own inputs, Figure 3. The app runs in the Java Virtual Machine (JVM) where pre-trained models are called and new written inputs are recognized. JVM is a machine that provides runtime environment in which Java bytecode can be executed. We trained our models and evaluated their training time, memory spent and other features. After training process had been finished we provided experiments to evaluate prediction time using several new inputs with our application. The evaluation process depicted in Figure 3.

Using our app users can write any digit and the same inputs will be recognized using four models. We used special methods

that normalizes the inputs before prediction. It makes the app more robust and user friendly. Inputs can be written in any readable style, any size and any part of the interface. Before making prediction the application normalizes and centralizes inputs. We do not evaluate this part of the experiment and do not compute the time to normalizing inputs, only the recognizing time evaluated.

A glance at the Figure 4 provided reveals that the greatest and stable accuracy belongs to CapsNet model. It is also obvious from the Figure 5 that the loss function of CapsNet is also significantly low among other three models. As you can see from the graphs ResNet and CNN show similar results while multinomial regression model shows by far worst outcome.

CapsNet reveals best results, it reaches high accuracy in short time. At the beginning of the iteration accuracy decreases but after 2,000 iterations it stabilizes and maintain average accuracy around 99.4%. Other models do not show such high accuracy.

It is also important that the fastest recognition is made by CapsNet 1-2 seconds, even though the less memory is used by regression model Table 1. Other models work very slowly and it leads to delay in real-time application.

However, ResNet and CapsNet uses a lot more memory than

Figure 3. The Java based GUI application and the process recognizing handwritten digit. The same inputs are given and four trained models recognize a written digit. Predicted digit, allocated time to make a prediction and network accuracy of each model are shown.

Figure 4. Accuracy rate throughout epochs.

other models. Additionally, the other table shows some other vital indexes, based on the Table 2, regression and CNN models require much less time to train 2.5 and 21 minutes, respectively.

ResNet require more time to train because of its deep architecture and uses much bigger memory to save that model. However, it recognizes digits much faster than CNN even though it had deeper network structure, it spends 5-7 seconds to make

Figure 5. Loss function of all models.

prediction. Since it shares the same parameters during training process and uses shortcut connections to avoid vanishing and degradation problems as mentioned earlier. This behavior allows the network to act much faster during prediction, as showed in [24] our results also reflected similar outcomes concerning this condition. Unlikely, CapsNet also uses big memory to save the model 49.4 mb but it recognizes digits very fast and very accurately. Other essential factor is number of inputs:

Table 1. Evaluation time and memory usage

	Regression model	CNN	ResNet	CapsNet
Evaluation time (s)	3-4	7-12	5-7	1-2
Memory (MB)	3.1	38	57	49.4

Table 2. Accuracy, training time and number of inputs

	Regression model	CNN	ResNet	CapsNet
Dataset	MNIST			
Accuracy (%)	92.1	98.1	97.3	99.4
Training time (min)	2.5	21	53	47
Number of inputs	50,000	50,000	50,000	40,000

we use 50,000 inputs to train our models accept CapsNet. We reduce the number of inputs for CapsNet and use 40,000 inputs to feed the network. Surprisingly, our CapsNet model reaches the highest accuracy among remainders with smaller number of inputs Table 2.

5. Conclusion

To sum up briefly, according to our experiment, we can reach to very high accuracy, even using smaller inputs, in CapsNet model. Also it works very fast in real-time application and recognizes handwritten digits considerable fast (within 1-2 seconds) comparing to other models. It is the most suitable model for real-time application as other three different models delay during evaluation and cannot reach as high accuracy as CapsNet model. Although memory usage of the trained model is more than CNN and regression model it predicts results better than both two models.

Also, based on our experiments, CapsNet require less time to train and less time to save the model in contrast ResNet.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgements

This research was supported by the Ministry of Science and ICT, Korea, under the Information Technology Research Center sup-

port program (No. IITP-2018-2017-0-01630) supervised by the Institute for Information & communications Technology Promotion (IITP) and NRF project (No. 20151D1A1A01061271, Intelligent Smart City Convergence Platform).

References

- [1] K. I. Kim and K. M. Lee, "Context-aware information provisioning for vessel traffic service using rule-based and deep learning techniques," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 18, no. 1, pp. 13-19, 2018. <http://doi.org/10.5391/IJFIS.2018.18.1.13>
- [2] H. W. Lee, N. R. Kim, and J. H. Lee, "Deep neural network self-training based on unsupervised learning and dropout," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 17, no. 1, pp. 1-9, 2017. <https://doi.org/10.5391/IJFIS.2017.17.1.1>
- [3] W. A. Bergerud, "Introduction to regression models: with worked forestry examples," *British Columbia Ministry of Forests*, Victoria, Canada, 1996.
- [4] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3859-3869, 2017.
- [5] A. Alsaafin and A. Elnagar, "A minimal subset of features using feature selection for handwritten digit recognition," *Journal of Intelligent Learning Systems and Applications*, vol. 9, no. 4, pp. 55-68, 2017. <https://doi.org/10.4236/jilsa.2017.94006>
- [6] J. Engel, "Polytomous logistic regression," *Statistica Neerlandica*, vol. 42, no. 4, pp. 233-252, 1988. <https://doi.org/10.1111/j.1467-9574.1988.tb01238.x>
- [7] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2005. <https://doi.org/10.1002/0471722146>
- [8] C. C. J. Kuo, "Understanding convolutional neural networks with a mathematical model," *Journal of Visual Communication and Image Representation*, vol. 41, pp. 406-413, 2016. <https://doi.org/10.1016/j.jvcir.2016.11.003>
- [9] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proceedings of 7th International Conference on Document Analysis and Recognition*, Edinburgh, UK, 2003, pp. 958-962.

- [10] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097-1105, 2012.
- [11] M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, 2007, pp. 1-8. <https://doi.org/10.1109/CVPR.2007.383157>
- [12] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," 2013, Available <https://arxiv.org/abs/1301.3557>
- [13] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning*, Atlanta, GA, 2013, pp. 1319-1327.
- [14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, 2014.
- [15] F. Lauer, C. Y. Suen, and G. Bloch, "A trainable feature extractor for handwritten digit recognition," *Pattern Recognition*, vol. 40, no. 6, pp. 1816-1824, 2007. <https://doi.org/10.1016/j.patcog.2006.10.011>
- [16] C. Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, San Diego, CA, 2015, pp. 562-570.
- [17] K. He, X. Zhang, S. Ren and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904-1916. <https://doi.org/10.1109/TPAMI.2015.2389824>
- [18] R. Girshick, "Fast R-CNN," in *Proceedings of 2015 IEEE International Conference on Computer Vision*, Santiago, Chile, 2015, pp. 1440-1448.
- [19] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," 2013, Available <https://arxiv.org/abs/1301.3572>
- [20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 580-587. <https://doi.org/10.1109/CVPR.2014.81>
- [21] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. FeiFei, "ImageNet: a large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, 2009, pp. 248-255. <https://doi.org/10.1109/CVPR.2009.5206848>
- [22] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915-1929. <https://doi.org/10.1109/TPAMI.2012.231>
- [23] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal, Canada, 2009, pp. 609-616. <https://doi.org/10.1145/1553374.1553453>
- [24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 2016, pp. 770-778. <https://doi.org/10.1109/CVPR.2016.90>
- [25] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704-1716, 2012. <https://doi.org/10.1109/TPAMI.2011.235>
- [26] C. M. Bishop, "Probability density estimation," in *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press, 1995, pp. 33-76.
- [27] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press, 1996. <https://doi.org/10.1017/CBO9780511812651>
- [28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997. <https://doi.org/10.1162/neco.1997.9.8.1735>

- [29] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," 2015, Available <https://arxiv.org/abs/1505.00387>
- [30] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," *Advances in Neural Information Processing Systems*, vol. 28, pp. 2377-2385, 2015.
- [31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, et al., "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Savannah, GA, 2016, pp. 265-283.
- [32] A. Jain, P. Pereira, and S. You, "Classification of modified MNIST," Available <https://www.cs.mcgill.ca/~syous3/papers/comp-598-MNIST.pdf>
- [33] D. M. Levi and T. Carney, "Crowding in peripheral vision: why bigger is better," *Current Biology*, vol. 19, no. 23, pp. 1988-1993, 2009. <https://doi.org/10.1016/j.cub.2009.09.056>
- [34] D. Whitney and D. M. Levi, "Visual crowding: a fundamental limit on conscious perception and object recog-

niton," *Trends in Cognitive Sciences*, vol. 15, no. 4, pp. 160-168, 2011. <https://doi.org/10.1016/j.tics.2011.02.005>

Akmaljon Palvanov received his B.S. in telecommunication technologies from Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi, Tashkent, Uzbekistan, in 2017. He is currently pursuing his M.S. in computer engineering at Gachon University. His current research interests include AI, data analysis and smart city.

E-mail: akmaljon.palvanov@gmail.com

Young Im Cho received her B.S., M.Sc., and Ph.D. from the Department of Computer Science, Korea University, Korea, in 1988, 1990, and 1994, respectively. She is a professor at Gachon University. Her research interest includes AI, big data, information retrieval, smart city, etc.

E-mail: yicho@gachon.ac.kr