Copyright © 2019 Tech Science Press CMES, vol.118, no.1, pp.1-14, 2019

Nonlinear Activation Functions in CNN Based on Fluid Dynamics and Its Applications

Kazuhiko Kakuda[1, *], Tomoyuki Enomoto[1] and Shinichiro Miura[2]

Abstract: The nonlinear activation functions in the deep CNN (Convolutional Neural Network) based on fluid dynamics are presented. We propose two types of activation functions by applying the so-called parametric softsign to the negative region. We use significantly the well-known TensorFlow as the deep learning framework. The CNN architecture consists of three convolutional layers with the max-pooling and one fully-connected softmax layer. The CNN approaches are applied to three benchmark datasets, namely, MNIST, CIFAR-10, and CIFAR-100. Numerical results demonstrate the workability and the validity of the present approach through comparison with other numerical performances.

Keywords: Deep learning, CNN, activation function, fluid dynamics, MNIST, CIFAR-10, CIFAR-100.

1 Introduction

The state-of-the-art on the deep learning in artificial intelligence is nowadays indispensable in engineering and science fields, such as robotics, automotive engineering, web-informatics, bio-informatics, and so on. There are recently some neural networks in the deep learning framework [LeCun, Bengio and Hinton (2015)], i.e., CNN (Convolutional Neural Networks) to recognize object images [Fukushima and Miyake (1982); LeCun, Bottou, Bengio et al. (1998); Krizhevsky, Sutskever and Hinton (2012)], RNN (Recurrent Neural Networks) to process time-series data [Rumelhart, Hinton and Williams (1986)], and so forth.

The appropriate choice of the activation functions for neural networks is a key factor in the deep learning simulations. Heretofore, there have been significantly proposed various activation functions in the CNN/RNN-frameworks. The standard activation function is the rectified linear unit (ReLU) introduced firstly by Hahnloser et al. [Hahnloser, Sarpeshkar, Mahowald et al. (2000)] in the theory of symmetric networks with rectification (It was called a rectification nonlinearity or ramp function [Cho and Saul (2009)]). Nair et al. [Nair and Hinton (2010)] have successfully performed by applying the ReLU activation functions based on the restricted Boltzmann machines to the deep

[1] Department of Mathematical Information Engineering, College of Industrial Technology, Nihon University, Chiba 275-8575, Japan.

[2] Department of Liberal Arts and Basic Sciences, College of Industrial Technology, Nihon University, Chiba 275-8576, Japan.

[*] Corresponding Author: Kazuhiko Kakuda. Email: kakuda.kazuhiko@nihon-u.ac.jp.

neural networks. The activation function of ReLU has been widely used by many researchers for visual recognition tasks [Glorot, Bordes and Bengio (2011); Krizhevsky, Sutskever and Hinton (2012); Srivastava, Hinton, Krizhevsky et al. (2014); LeCun, Bengio and Hinton (2015); Kuo (2016); Agarap (2018)]. The ReLU activation function leads to better recognition performances than conventional sigmoid/*tanh* units involving the vanishing gradient problem, while has parameter-free, and zero-gradients in the negative part.

In order to provide meaningful such negative values, there have been presented some activation functions, such as leaky rectified linear unit (LReLU) [Maas, Hannun and Ng (2013)], parametric rectified linear unit (PReLU) [He, Zhang, Ren et al. (2015)], exponential linear unit (ELU) [Clevert, Unterthiner and Hochreiter (2016)], and so forth. The LReLU has been slightly improved for ReLU by replacing the negative part of the ReLU with a linear function involving small constant gradient. The PReLU has been generalized by adaptively learning the parameters introduced in the negative part of LReLU. They have improved significantly the learning performances on large image datasets called *ImageNet*. Clevert et al. [Clevert, Unterthiner and Hochreiter (2016)] have proposed an activation function, ELU, and shown applicability and validity for various benchmark datasets. As another approach, Goodfellow et al. [Goodfellow, Warde-Farley, Mirza et al. (2013)] have also proposed an activation function called maxout that has features both for optimization and model averaging with *dropout* [Hinton, Srivastava, Krizhevsky et al. (2012)].

In our previous work, we have presented newly the characteristic function (i.e., activation function) as an optimum function which is derived on the advection-diffusion system in fluid dynamics framework [Kakuda (2002)]. The purpose of this paper is to propose the activation functions based on the concept of fluid dynamics framework. We present two types of activation functions by applying the so-called parametric softsign [Glorot and Bengio (2010)] to the negative part of ReLU. By using the well-known TensorFlow [Abadi, Agarwal, Barham et al. (2015)] as the deep learning framework, we utilize the CNN architecture that consists of three convolutional layers with the max-pooling and one fully-connected softmax layer. The workability and the validity of the present approach are demonstrated on three benchmark datasets, namely, MNIST [LeCun, Bottou, Bengio et al. (1998)], CIFAR-10 and CIFAR-100 [Krizhevsky and Hinton (2009)], through comparison with other numerical performances.

2 Construction of nonlinear activation functions

2.1 Neural network model

In the field of neural networks, the input-output (*I/O*) relationship known as the back-propagation is represented by inputs U_j, output V_j and the characteristic function h (i.e., activation function) as follows:

$$V_j = h(U_j) \tag{1}$$

$$U_j = \sum_{i=1}^{n} S_{ij} w_{ij} + I_j - T_j \tag{2}$$

where S_{ij} are *j*-th input values as shown in Fig. 1, w_{ij} are the connection weights, I_j is the bias value, and T_j denotes threshold.

The sigmoid function (see Fig. 2(a)) has been mainly used as the following continuous function.

$$h(v) = \frac{1}{2}\left\{1 + tanh\left(\frac{v}{2k}\right)\right\} \tag{3}$$

where *k* is *ad hoc* parameters.

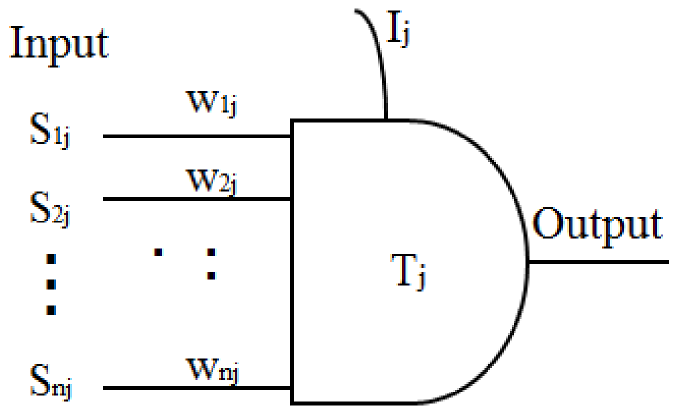


Figure 1: Neuron model

2.2 Nonlinear activation functions based on fluid dynamics

Heretofore, we have presented the following activation function as an optimum function which is derived on the steady advection-diffusion system in fluid dynamics framework [Kakuda (2002)] (see Eq. (25) in next Subsection 2.3).

$$h(v) = \frac{1}{2}\{1 + \hat{g}(v)\} \tag{4}$$

$$\hat{g}(v) = coth(\gamma) - \frac{1}{\gamma} \tag{5}$$

where $\gamma = v/2k > 0$.

Mizukami [Mizukami (1985)] has presented significantly the following approximation function instead of Eq. (5) involving the singularity.

$$\tilde{g}(v) = 1 - \frac{1}{\gamma+1} \tag{6}$$

Therefore, we obtain the functions by substituting Eq. (6) into Eq. (4) and considering the sign of *v* as follows (see Fig. 2(b)):

$$h(v) = \begin{cases} \frac{1}{2}\left(2 - \frac{1}{1+|\gamma|}\right) & (\gamma \geq 0) \\ \frac{1}{2}\left(\frac{1}{1+|\gamma|}\right) & (\gamma < 0) \end{cases} \tag{7}$$

In this stage, we adjust the functions, *h*(*v*), so that $g(0) = 0$.

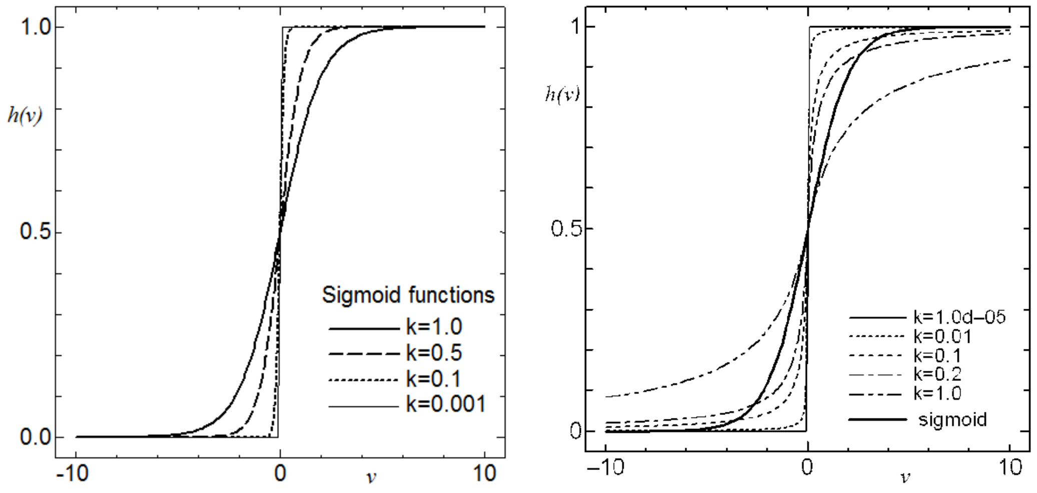
$$g(v) = \hat{\sigma}\{h(v) - h(0)\}, \;\; h(0) = \frac{1}{2} \tag{8}$$

As a result, we obtain the following form by taking into account that $\hat{\sigma} = 2\kappa$.

$$g(v) = \frac{\kappa v}{\kappa + |v|} \tag{9}$$

in which $\kappa = 2k$. Eq. (9) represents the softsign function with $\kappa = 1$ [Glorot and Bengio (2010)]. The so-called parametric softsign is equivalent to the ReLU [Nair and Hinton (2010)] under the conditions, such as $\kappa = +\infty$ for $v \geq 0$ and $\kappa = 0$ for $v < 0$.

In order to avoid zero-gradients in the negative part of v, by applying Eq. (9) to the negative region, we propose two types of activation function involving parameter, a, as follows (see Fig. 3):



(a) Sigmoid functions with k (b) Characteristic functions of Eq. (7)

Figure 2: Characteristic functions

Rational-type activation function and its derivatives

$$g(v) = \begin{cases} v & (v \geq 0) \\ \frac{av}{a+|v|} & (v < 0) \end{cases} \tag{10}$$

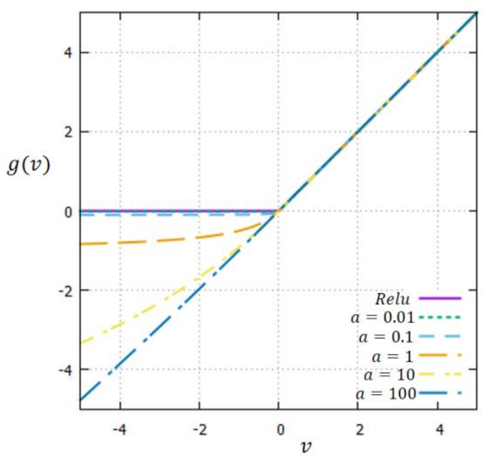
$$\frac{\partial g(v)}{\partial v} = \begin{cases} 1 & (v \geq 0) \\ \left(\frac{a}{a+|v|}\right)^2 & (v < 0) \end{cases}, \quad \frac{\partial g(v)}{\partial a} = \begin{cases} 0 & (v \geq 0) \\ \frac{v|v|}{(a+|v|)^2} & (v < 0) \end{cases} \tag{11}$$

Exponential-type activation function and its derivatives

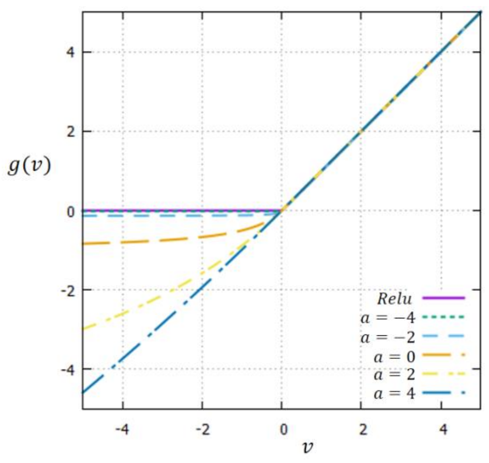
$$g(v) = \begin{cases} v & (v \geq 0) \\ \frac{e^a v}{e^a+|v|} & (v < 0) \end{cases} \tag{12}$$

$$\frac{\partial g(v)}{\partial v} = \begin{cases} 1 & (v \geq 0) \\ \left(\frac{e^a}{e^a+|v|}\right)^2 & (v < 0) \end{cases}, \quad \frac{\partial g(v)}{\partial a} = \begin{cases} 0 & (v \geq 0) \\ \frac{e^a v|v|}{(e^a+|v|)^2} & (v < 0) \end{cases} \tag{13}$$

The corresponding derivatives of the activation functions are also shown in Fig. 4.

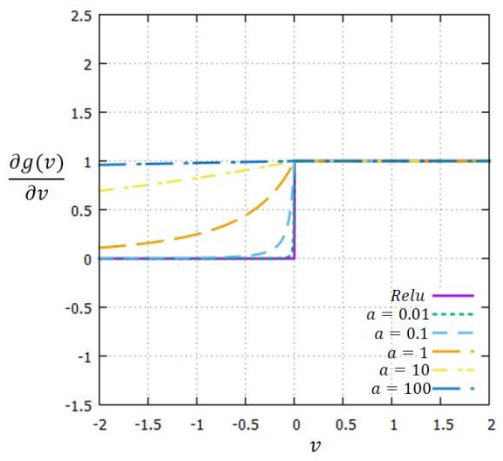


(a) Rational-type activation function

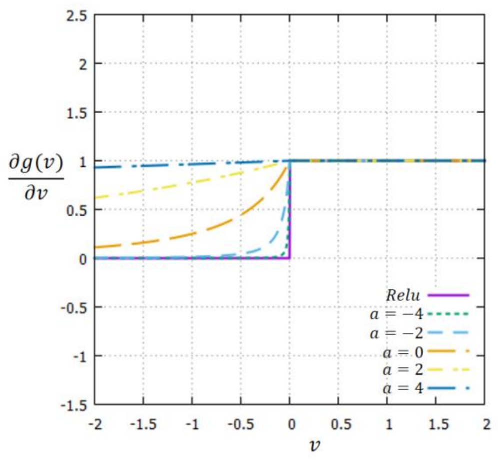


(b) Exponential-type activation function

Figure 3: Nonlinear activation functions



(a) Rational-type activation function



(b) Exponential-type activation function

Figure 4: Derivatives of activation functions

2.3 Steady advection-diffusion equation

2.3.1 Problem statement

Let us briefly consider the one-dimensional advection-diffusion equation in spatial coordinate, x, given by

$$f_{,x} = k\varphi_{,xx} \tag{14}$$

with the adequate boundary conditions, where $f = u\varphi$, u and k are the given velocity and diffusivity, respectively.

2.3.2 Finite element formulation

In order to solve the flux, $f = u\varphi$, in a stable manner, we shall adopt the Petrov-Galerkin finite element formulation using exponential weighting function [Kakuda and Tosaka (1992)]. On the other hand, the conventional Galerkin finite element formulation can be applied to solve numerically Eq. (14).

First of all, we start with the following weighted integral expression in a subdomain $\Omega_i = [x_{i-1}, x_i]$ with respect to weighting function $\widetilde{w}$:

$$\int_{\Omega_i} (f - u\varphi)\widetilde{w}dx = 0 \tag{15}$$

The weighting function $\widetilde{w}$ can be chosen as a general solution which satisfies

$$u\widetilde{w} + \Delta x_i \sigma(u)\widetilde{w}_{,x} = 0 \tag{16}$$

where $\Delta x_i = x_i - x_{i-1}$, and $\sigma(u)$ denotes some function described by Yee et al. [Yee, Warming and Harten (1985)], which is sometimes referred to as the coefficient of numerical viscosity. The solution of Eq. (16) is as follows:

$$\widetilde{w} = Ae^{-\hat{a}x} \tag{17}$$

where A is a constant, and $\hat{a} = u/\Delta x_i \sigma(u)$.

By applying the piecewise linear function to the flux f and φ, we obtain the following integral form

$$\int_{\Omega_i} M_\alpha N_\beta \, dx f_\beta - u \int_{\Omega_i} M_\alpha N_\beta \, dx \varphi_\beta = 0 \tag{18}$$

in which

$$M_\alpha = e^{-\hat{a}(x - x_\alpha)} \qquad (\alpha = 1, 2) \tag{19}$$

Here, applying an element-wise mass lumping to the first term of the left-hand side of Eq. (18), and carrying out exactly those integrals in Eq. (18), we can obtain the following numerical fluxes $f_{i-1/2}$ and $f_{i+1/2}$ in the subdomains Ω_i and Ω_{i+1}, respectively

$$f_{i-1/2} = f_i + \frac{u}{2}\left[1 + \left\{sgn(\gamma)coth|\gamma| - \frac{1}{\gamma}\right\}\right](\varphi_{i-1} - \varphi_i) \tag{20}$$

$$f_{i+1/2} = f_i + \frac{u}{2}\left[-1 + \left\{sgn(\gamma)coth|\gamma| - \frac{1}{\gamma}\right\}\right](\varphi_i - \varphi_{i+1}) \tag{21}$$

where $\gamma = u/2\sigma(u)$, and $sgn(\gamma)$ denotes the signum function.

Let us next derive the Galerkin finite element model for Eq. (14). The weighted residual equation in Ω_i is given as follows:

$$\backslash\int_{\Omega_i} (f_{,x} - k\varphi_{,xx}) w dx = 0 \tag{22}$$

In this stage, we assume a uniform mesh $\Delta x_i = \Delta x$ for simplicity of the formulation. Taking into consideration the continuity of $\varphi_{,x}$ at nodal point i, we can obtain the following discrete form

$$f_{i-1/2} - f_{i+1/2} + \frac{k}{\Delta x}(\varphi_{i-1} - 2\varphi_i + \varphi_{i+1}) = 0 \tag{23}$$

Substituting Eq. (20) and Eq. (21) into Eq. (23) and after some manipulations, we obtain the following finite difference form

$$\frac{u}{2\Delta x}(\varphi_{i+1} - \varphi_{i-1}) = (k + \tilde{k})\frac{\varphi_{i-1} - 2\varphi_i + \varphi_{i+1}}{\Delta x^2} \tag{24}$$

where for any velocity u

$$\tilde{k} = \frac{|u|\Delta x}{2}\left\{coth|\gamma| - \frac{1}{|\gamma|}\right\} \tag{25}$$

Using the element Peclet number $P_e(\equiv \Delta xu/2k)$ as γ, we reduce Eq. (24) to the following form

$$\{sgn(P_e) - coth|P_e|\}\varphi_{i+1} + 2coth|P_e|\varphi_i - \{sgn(P_e) + coth|P_e|\}\varphi_{i-1} = 0 \tag{26}$$

This equation has the same structure as the SUPG scheme developed by Brooks et al. [Brooks and Hughes (1982)], and it leads to nodally exact solutions for all values of P_e [Christie, Griffiths, Mitchell et al. (1976)].

3 CNN architecture

We adopt the similar approach as the *PReLU* [He, Zhang, Ren et al. (2015)] which can be trained using back-propagation and optimized simultaneously with other layers. As the variables, v and a, in Eq. (10) through Eq. (13), we define v_j and a_j with respect to the input and the coefficient, respectively, on the j-th channel. The momentum approach when updating a_j is given as follows:

$$\Delta a_j \coloneqq \mu\Delta a_j + \varepsilon\frac{\partial E}{\partial a_j} \tag{27}$$

$$\frac{\partial E}{\partial a_j} = \sum_{v_j}\frac{\partial E}{\partial g(v_j)}\frac{\partial g(v_j)}{\partial a_j} \tag{28}$$

where E represents the objective function, μ is the momentum to accelerate learning, ε is the learning rate. The parameters, a_j, are optimally obtained by using back-propagation analysis.

Fig. 5 shows the CNN architecture consisting of three convolutional (i.e., *conv*) layers with some max-pooling and one fully-connected (i.e., *fc*) *softmax* layer.

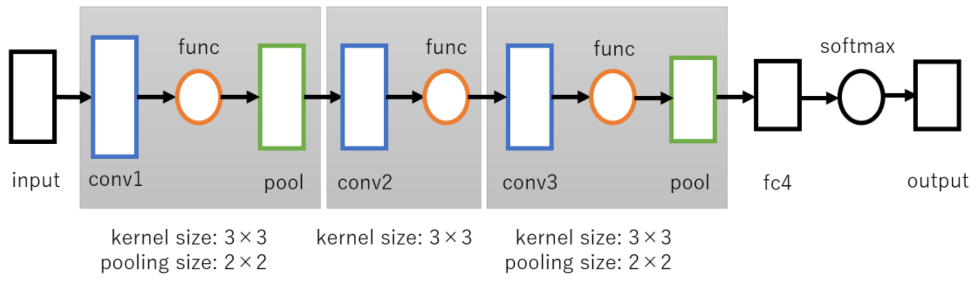


Figure 5: CNN architecture

4 Numerical experiments

In this section, we use the well-known TensorFlow [Abadi, Agarwal, Barham et al. (2015)] as the deep learning framework, and present numerical performances obtained from applications of the above-mentioned CNN approach to three typical datasets, namely, MNIST [LeCun, Bottou, Bengio et al. (1998)], CIFAR-10 and CIFAR-100 [Krizhevsky and Hinton (2009)]. We utilize the Adam [Kingma and Ba (2015)] as the learning algorithm for the stochastic gradient-based optimization.

The model is previously trained for some epochs on mini-batches of size 100 with the learning rate, $\varepsilon = 10^{-3}$, and the momentum, $\mu = 0$. The specification of CPU and GPU using CUDA is summarized in Tab. 1.

Table 1: A summary of the specification of CPU and GPU

CPU	Intel® Core™ i7-8700 K
Cores	12
Base Clock	4.2 GHz
Cashe Memory	12 MB
GPU	NVIDIA® GeForce GTX 1080
Global Memory	8 GB
CUDA core	2560
GPU Max Clock rate	1607 MHz
GPU Boost Clock rate	1733 MHz
Memory Clock rate	10 Gbps
Memory Bandwidth	320 GB/s
CUDA Driver	Version 9.0

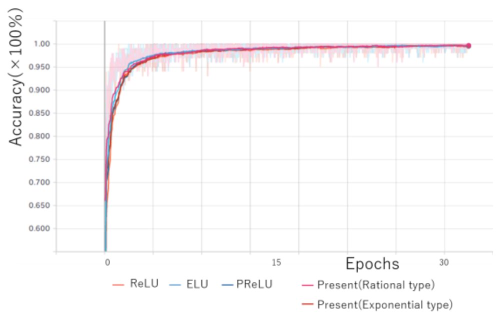
4.1 MNIST

Let us first consider the MNIST dataset which consists of 28×28 pixel gray-scale handwritten digit images with 50,000 for training and 10,000 for testing images.

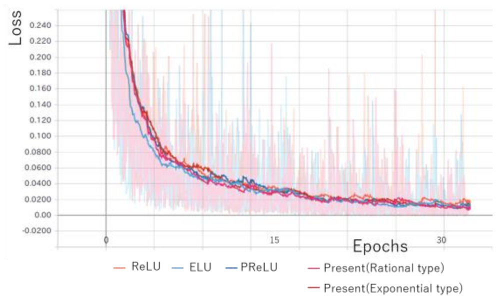
Fig. 6 shows the behaviors of training accuracy and loss (i.e., cross-entropy) obtained by using various activation functions for the MNIST. The corresponding validation accuracy and loss behaviors are shown in Fig. 7. We can see from Fig. 6 and Fig. 7 that our approaches are similar to the ones using other activation functions. Tab. 2 summarizes the transitions of the learned parameter, a, at each layer of CNN architecture (see Fig. 5) for the MNIST. The validation accuracy rate and loss for the MNIST are given in Tab. 3. In this case, the quantitative agreement between our results and other ones appears also satisfactory.

Table 2: Transitions of the learned parameter, a, for the MNIST

	PReLU	Present Rational-type	Present Exponential-type
Initial value	0.25	1.0	0.0
conv1	0.3629	0.8776	-0.1513
conv2	0.1410	1.091	0.01067
conv3	0.1241	1.054	0.03090

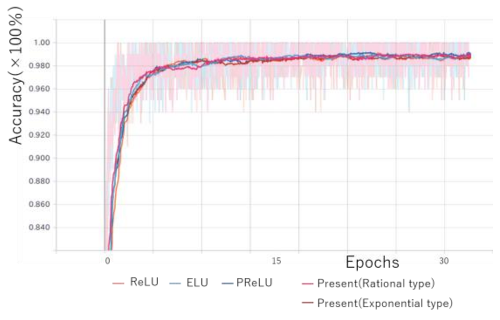


(a) Training accuracy

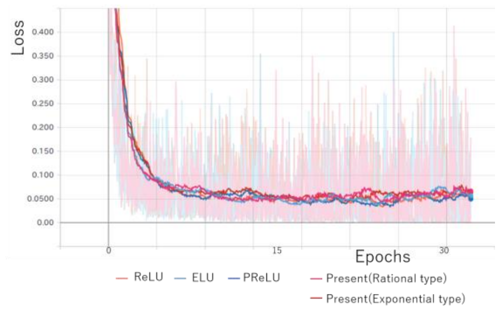


(b) Training loss

Figure 6: Training accuracy and loss behaviors for the MNIST



(a) Validation accuracy



(b) Validation loss

Figure 7: Validation accuracy and loss behaviors for the MNIST

Table 3: Accuracy rate and loss for the MNIST

	ReLU	ELU	PReLU	Present Rat.-type	Present Exp.-type
Accuracy [%]	98.92	98.61	98.79	98.77	98.79
Loss	0.050503	0.065178	0.054787	0.055806	0.052109

4.2 CIFAR-10

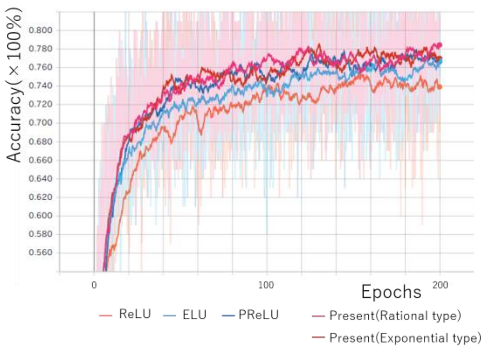
As the second benchmark dataset, we consider the CIFAR-10 which consists of 32×32 color images drawn from 10 classes with 50,000 for training and 10,000 for testing images.

Fig. 8 shows the behaviors of training accuracy and loss (i.e., cross-entropy) obtained by using various activation functions for the CIFAR-10. The corresponding validation accuracy and loss behaviors are shown in Fig. 9. We can see from Fig. 8 and Fig. 9 that our approaches outperform entirely the ones using other activation functions. Tab. 4 summarizes the transitions of the learned parameter, *a*, at each layer of CNN architecture (see Fig. 5) for the CIFAR-10. The validation accuracy rate and loss for the CIFAR-10 are given in Tab. 5. For the accuracy rate on the CIFAR-10, we obtain best result of

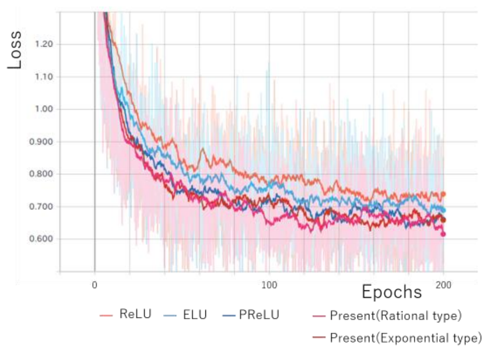
80.76% using the exponential-type activation function. On the other hand, our approaches for the loss outperform other ones.

Table 4: Transitions of the learned parameter, *a*, for the CIFAR-10

	PReLU	Present Rational-type	Present Exponential-type
Initial value	0.25	1.0	0.0
conv1	0.05603	0.1178	-2.110
conv2	0.02731	0.1032	-2.029
conv3	0.03020	0.2062	-1.656

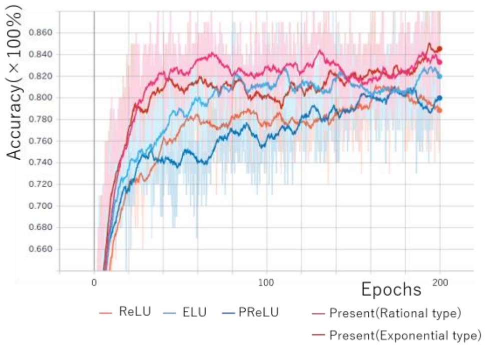


(a) Training accuracy

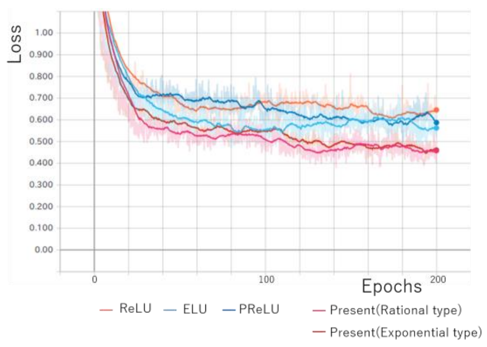


(b) Training loss

Figure 8: Training accuracy and loss behaviors for the CIFAR-10



(a) Validation accuracy



(b) Validation loss

Figure 9: Validation accuracy and loss behaviors for the CIFAR-10

Table 5: Accuracy rate and loss for the CIFAR-10

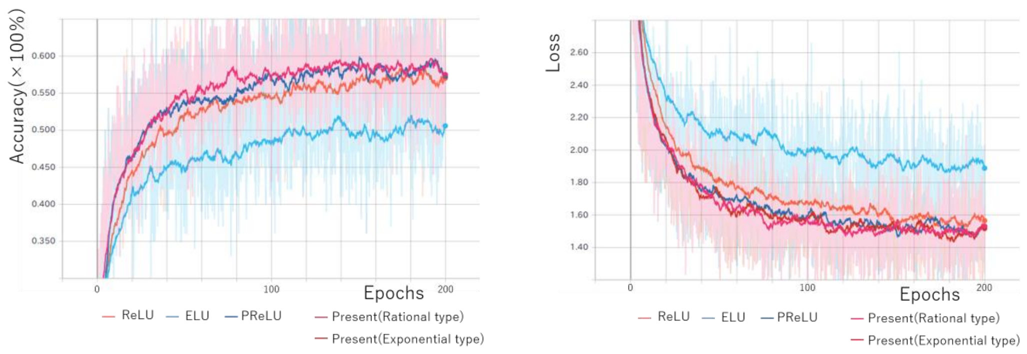
	ReLU	ELU	PReLU	Present Rat.-type	Present Exp.-type
Accuracy [%]	77.68	80.40	79.76	80.23	80.76
Loss	0.56802	0.50304	0.55520	0.38574	0.39621

4.3 CIFAR-100

As the third benchmark dataset, the CIFAR-100 is the same size and format as the CIFAR-10 one, while contains 100 classes for consisting of 20 super-classes with five classes each.

Table 6: Transitions of the learned parameter, *a*, for the CIFAR-100

	PReLU	Present Rational-type	Present Exponential-type
Initial value	0.25	1.0	0.0
conv1	0.01511	0.02385	-3.405
conv2	0.02066	0.06526	-2.455
conv3	0.02695	0.08438	-2.564

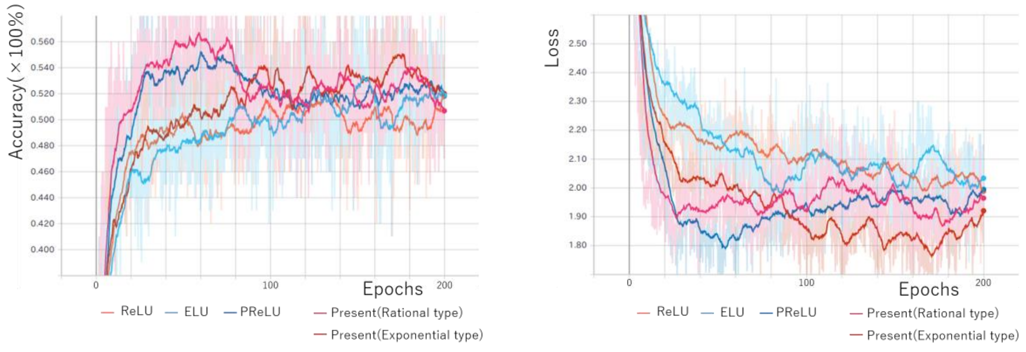


(a) Training accuracy (b) Training loss

Figure 10: Training accuracy and loss behaviors for the CIFAR-100

Table 7: Accuracy rate and loss for the CIFAR-100

	ReLU	ELU	PReLU	Present Rat.-type	Present Exp.-type
Accuracy [%]	54.96	56.03	56.56	56.91	56.52
Loss	1.83145	1.71715	1.81471	1.66373	1.55405



(a) Validation accuracy (b) Validation loss

Figure 11: Validation accuracy and loss behaviors for the CIFAR-100

Fig. 10 shows the behaviors of training accuracy and loss obtained by using various activation functions for the CIFAR-100. The corresponding validation accuracy and loss behaviors are shown in Fig. 11. We can see from Fig. 10 and Fig. 11 that our approaches outperform the ones using other activation functions. Tab. 6 summarizes the transitions of the learned parameter, *a*, at each layer of CNN architecture for the CIFAR-100. The validation accuracy rate and loss for the CIFAR-100 are given in Tab. 7. For the accuracy rate on the CIFAR-100, we obtain best result of 56.91% using the rational-type activation function. On the other hand, our approaches for the loss outperform also other ones.

5 Conclusions

We have proposed new activation functions which were based on the steady advection-diffusion system in fluid dynamics framework. In our formulation, two types of activation functions have been reasonably presented by applying the so-called parametric softsign to the negative part of ReLU. By using the TensorFlow as the deep learning framework, we have utilized the CNN architecture that consists of three convolutional layers with some max-pooling and one fully-connected softmax layer.

The performances of our approaches were carried out on three benchmark datasets, namely, MNIST, CIFAR-10 and CIFAR-100, through comparison with the ones using other activation functions. The learning performances demonstrated that our approaches were capable of recognizing somewhat accurately and in less loss (i.e., cross-entropy) the object images in comparison with other ones.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z. et al. (2015): *TensorFlow: large-scale machine learning on heterogeneous system*. https://www.tensorflow.org/.

Agarap, A. F. M. (2018): Deep learning using rectified linear units (ReLU). *arXiv:1803.08375v1*.

Brooks, A.; Hughes, T. J. R. (1982): Streamline upwind/Petrov-Galerkin formulations for convection dominated flow with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, vol. 32, pp.

199-259.

Cho, Y.; Saul, L. K. (2009): Kernel methods for deep learning. *Advances in Neural Information Processing Systems*, vol. 22, pp. 342-350.

Christie, I.; Griffiths, D. F.; Mitchell, A. R.; Zienkiewicz, O. C. (1976): Finite element methods for second order differential equations with significant first derivatives. *International Journal for Numerical Methods in Engineering*, vol. 10, pp. 1389-1396.

Clevert, D. A.; Unterthiner, T.; Hochreiter, S. (2016): Fast and accurate deep network learning by exponential linear units (ELUs). *International Conference on Learning Representations* (*ICLR*), arXiv:1511.07289v5.

Fukushima, K.; Miyake, S. (1982): A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, vol. 15, pp. 455-469.

Glorot, X.; Bengio, Y. (2010): Understanding the difficulty of training deep feedforward neural networks. *13th International Conference on Artificial Intelligence and Statistics*, pp. 249-256.

Glorot, X.; Bordes, A.; Bengio, Y. (2011): Deep sparse rectifier neural networks. *14th International Conference on Artificial Intelligence and Statistics*, pp. 315-323.

Goodfellow, I. J.; Warde-Farley, D.; Mirza, M.; Courville, A.; Bengio, Y. (2013): Maxout networks. *30th International Conference on Machine Learning*, pp. 1319-1327.

Hahnloser, R. H. R.; Sarpeshkar, R.; Mahowald, M. A.; Douglas, R. J.; Seung, H. S. (2000): Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, vol. 405, pp. 947-951.

He, K.; Zhang, X.; Ren, S.; Sun, J. (2015): Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. *IEEE International Conference on Computer Vision*, *a*rXiv:1502.01852v1.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. R. (2012): Improving neural networks by preventing co-adaptation of feature detectors. *Technical Report*, arXiv:1207.0580v1.

Kakuda, K. (2002): Applications of fluid dynamic approach to neural network. *15th Computational Mechanics Conference*, JSME, pp. 529-530. (In Japanese)

Kakuda, K.; Tosaka, N. (1992): Finite element approach for high Reynolds number flows. *Theoretical and Applied Mechanics*, vol. 41, pp. 223-232.

Kingma, D. P.; Ba, J. L. (2015): ADAM: a method for stochastic optimization. *International Conference on Learning Representations*, arXiv:1412.6980v8.

Krizhevsky, A.; Hinton, G. E. (2009): *Learning multiple layers of features from tiny images*. *Technical Report*, University of Toronto, Canada.

Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2012): ImageNet classification with deep convolutional neural networks. *25th International Conference on Neural Information Processing Systems*, pp. 1097-1105.

Kuo, C. C. J. (2016): Understanding convolutional neural networks with a mathematical model. *arXiv:1609.04112v2*.

LeCun, Y.; Bengio, Y.; Hinton, G. E. (2015): Deep learning. *Nature*, vol. 521, pp. 436-

444.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. (1998): Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324.

Maas, A. L.; Hannun, A. Y.; Ng, A. Y. (2013): Rectifier nonlinearities improve neural network acoustic models. *30th International Conference on Machine Learning*.

Mizukami, A. (1985): An implementation of the streamline-upwind/Petrov-Galerkin method for linear triangular elements. *Computer Methods in Applied Mechanics and Engineering*, vol. 49, pp. 357-364.

Nair, V.; Hinton, G. E. (2010): Rectified linear units improve restricted Boltzmann machines. *27th International Conference on Machine Learning*, pp. 807-814.

Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. (1986): Learning representations by back-propagating errors. *Nature*, vol. 323, pp. 533-536.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. R. (2014): Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958.

Yee, H. C.; Warming, R. F.; Harten, A. (1985): Implicit total variation diminishing (TVD) schemes for steady-state calculations. *Journal of Computational Physics*, vol. 57, pp. 327-360.