SELECTIVE CLASSIFICATION CONSIDERING TIME SERIES CHARACTERISTICS FOR SPIKING NEURAL NETWORKS

M. Yumoto[*], *M. Hagiwara*[*]

Abstract: In this paper, we propose new methods for estimating the relative reliability of prediction and rejection methods for selective classification for spiking neural networks (SNNs). We also optimize and improve the efficiency of the RC curve, which represents the relationship between risk and coverage in selective classification. Efficiency here means greater coverage for risk and less risk for coverage in the RC curve. We use the model internal representation when time series data is input to SNN, rank the prediction results that are the output, and reject them at an arbitrary rate. We propose multiple methods based on the characteristics of datasets and the architecture of models. Multiple methods, such as a simple method with discrete coverage and a method with continuous and flexible coverage, yielded results that exceeded the performance of the existing method, softmax response.

Key words: *classification with reject option, selective classification, spiking neural networks, RC curve*

Received: May 2, 2022 **DOI:** 10.14311/NNW.2023.33.004
Revised and accepted: April 30, 2023

1. Introduction

In recent years, neural networks have been used not only in the field of scientific research but also in various fields such as applications in the real world. It is necessary to properly handle the reliability and uncertainty of predictions in order to incorporate and operate the prediction results output by the neural network into the real world system. With the social attention to artificial intelligence, research on the realization of safe and reliable artificial intelligence and the quantification of uncertainty has been active in recent years [1].

For example, when operating a system related to human life such as automatic driving and medical image analysis in the real world, it is useful to use the output result of the model together with uncertainty. This is because uncertainty can be used to make decisions from in-vehicle camera images, and the system and humans can work together to determine the presence or absence of illness and its location.

*Masaya Yumoto – Corresponding author; Masafumi Hagiwara; Keio University, Faculty of Science and Technology, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Japan, E-mail: myumoto85@keio.jp, hagiwara@keio.jp

Selective classification also helps determine how much humans and systems share tasks.

There are four major approaches to estimating uncertainty using neural networks. The first is a single deterministic method [2]. It makes predictions based on one forward path of one deterministic network. Uncertainty quantification uses an internal representation in the network or is represented by an external method. The second is Bayesian method [3]. This is called a stochastic neural network in which the model parameters are considered as random variables. Bayesian modeling is performed by assuming prior distribution in the model parameters of the neural network. The third is an ensemble method [4]. It combines the predicted values of several different deterministic networks at the time of inference. Research has been carried out for the purpose of improving accuracy and robust prediction, but it can also be used to estimate the uncertainty of neural networks. The fourth is the test-time augmentation method [5]. It makes predictions based on one deterministic network. During the test, the input data is expanded to generate prediction distributions for multiple predictions.

Although each approach has its advantages and disadvantages, there are some advantages of single deterministic methods over the others. The first one is the high computational efficiency in learning and evaluation. Single deterministic methods are applicable to one neural network, especially a trained neural network. In contrast, Bayesian methods require multiple samplings, ensemble methods require multiple models, and test-time augmentation methods require multiple expanded data, resulting in high computational cost. Another point is that there is no need to make changes to the existing network. In the case of the Bayesian methods and ensemble methods, it is necessary to change the network for approximate inference and prepare different models. On the other hand, single deterministic methods have the disadvantage of being highly dependent on initial values and hyperparameters. While other methods utilize the diversity of models and data in various approaches to quantify uncertainty, it is difficult to ensure the diversity in single deterministic methods. This is in a trade-off relationship with the amount of calculation.

Considering the above discussion, we employ a single deterministic method in this paper. Specifically, we treat SNN, which is a type of neural network and has strengths in data processing with spatio-temporal patterns and low power consumption. We propose indices and rejection methods for estimating the relative reliability of predictions for selective classification in SNN. We improve the efficiency of the RC curve that represents the relationship between risk and coverage in selective classification. Using the model internal representation, the prediction results are ranked and rejected at an arbitrary rate. Then, we verify the performance of multiple methods depending on the architecture of the model and the dataset.

Existing research on selective classification and prediction reliability is mainly about quantifying the reliability of predictions output by models for each piece of data. On the other hand, in this paper, we focus on making it possible to flexibly and advantageously select the rejection rate and the expected error rate to be tolerated by improving the efficiency of the RC curve for a certain number of data.

The rest of this paper is organized as follows: In Section 2, we will provide the background; The description of the proposed method is presented in Section 3; Section 4 reports the evaluation experiments; and the conclusion is given in Section 5.

2. Background

2.1 Selective classification

Selective classification [6] is a classification with a reject option. A rejection option is an option that allows the rejection of a prediction if it is not confident. Selective classification has the important concepts of risk and coverage. Risk is literally the risk, the error rate of the classification. Coverage is the ratio of data handled without rejection to the entire data. There is a trade-off between this risk and coverage.

Fig. 1 shows the RC plane and RC trade-offs [6], which are the entire area of risk and coverage. A point (r, c) on the RC plane is achievable if selective classification is possible such that the coverage is at least $c \in \mathbb{R}$ and the risk is at most $r \in \mathbb{R}$. The RC plane consists of all points (r, c), and the achievable zone means the area of achievable points (r, c). Conversely, the non achievable zone is an area that cannot be achieved. The upper envelope means the lower bound of the non achievable zone, and the lower envelope is the upper bound of the achievable zone. In the region between these two boundaries, the curve connecting the corresponding points (r, c) of risk (r) and coverage (c) is the RC curve. An efficient RC curve for selective classification is an RC curve that is close to the upper envelope.

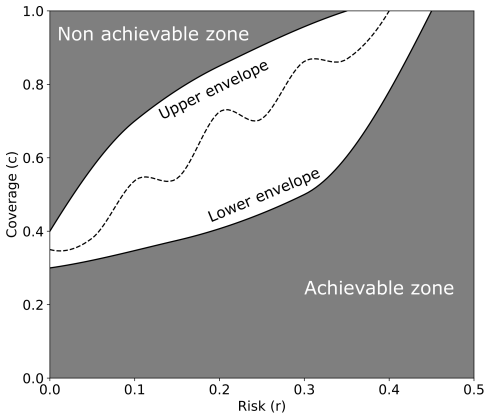

Fig. 1 *RC plane and RC trade-off.*

2.2 Spiking neural networks

SNN is a neural network that uses an artificial neuron model that focuses on the firing and spikes of neurons in the brain. In recent years, with the progress of deep learning in multi-layered artificial neural networks (ANNs), research on deep

learning in SNN is also progressing [7]. The input of SNN always contains time series information, and the internal representation of the model also has time series information. In this paper, we perform selective classification considering the characteristics of this time series internal representation.

3. Efficiency improvement of RC curve

The proposed method uses the time series internal representation of SNN. In this paper, we propose two kinds of methods: the case where the final layer of the model is a pooling layer and the case where it is a fully connected layer.

3.1 Problem statement

In this work, we address selective prediction in a multi-class classification problem [6]. Let $\mathcal{X}$ be a feature space and $\mathcal{Y}$ a label space. Let P be a distribution over $\mathcal{X} \times \mathcal{Y}$. Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{L}$ be a dataset sampled i.i.d. from P. A deep neural classifier f is a function $f : \mathcal{X} \to \mathcal{Y}$ and a loss function ℓ is $\ell : \mathcal{Y} \times \mathcal{Y} \to \{0, 1\}$, which is a 0/1 error.

A selective classifier is a pair (f, g), where f is a classification function and $g : \mathcal{X} \to \{0, 1\}$ is a selection function. g acts as a binary selector for f as follows:

$$(f, g)(x) \triangleq \begin{cases} f(x) & \text{if } g(x) = 1, \\ \text{reject} & \text{if } g(x) = 0. \end{cases} \tag{1}$$

Thus, the selective classifier rejects the prediction if $g(x) = 0$.

The selection function g is defined as follows:

$$g(x) \triangleq \begin{cases} 1 & \text{if } \kappa_{\mathrm{f}}(x) \geq \theta, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $\kappa_{\mathrm{f}} : \mathcal{X} \to \mathbb{R}$ is a confidence score function and $\theta \in \mathbb{R}$ is a threshold.

The risk of (f, g) for the dataset $\mathcal{D}$ is defined as:

$$\mathbf{R}(f, g|\mathcal{D}) \triangleq \frac{\frac{1}{L}\sum_{i=1}^{L} \ell(f(x_i), y_i) g(x_i)}{\mathbf{C}(g|\mathcal{D})}, \tag{3}$$

where $\mathbf{C}$ is the coverage for $\mathcal{D}$, which is defined as:

$$\mathbf{C}(g|\mathcal{D}) \triangleq \frac{1}{L}\sum_{i=1}^{L} g(x_i). \tag{4}$$

Each of the following proposed methods provides the different selection function g, the confidence score function κ_{f}, and the threshold θ.

3.2 The case where the final layer is a pooling layer

In this section, we propose five methods, “square matrix method”, “lower triangular matrix method”, “proposed without constraints”, “proposed with constraints”, and

"softmax response". SNN deals with a four-dimensional tensor called spike-wave tensor [8]. Using this tensor, we explain the input and output sizes. spike-wave tensor has the size of $T_{\max} \times F \times H \times W$, where $T_{\max} \in \mathbb{N}$ means the maximum possible number of time-steps, $F \in \mathbb{N}$ means feature (channel), $H \in \mathbb{N}$ means height, and $W \in \mathbb{N}$ means width. Since $T_{\max}$ is constant from the input layer to the output layer, it is omitted hereafter.

The final pooling layer is often the global max pooling layer or the global average pooling layer. In the global pooling layer, the maximum or average value of each feature F is taken, so a tensor with a size of $F_{\text{in}} \times 1 \times 1$ is output for an input with a size of $F_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}$. The total number of neurons in the layer immediately before the final layer is $F_{\text{in}}H_{\text{in}}W_{\text{in}}$. Let the number of classes in the classification be $L_{\text{class}} \in \mathbb{N}$. $F_{\text{in}}H_{\text{in}}W_{\text{in}}/L_{\text{class}}$ neurons correspond to each class.

Selective classification is performed using the time series internal representation in the layer immediately before this final layer. For each time-step, create a row vector in which the outputs of this layer for x_i are arranged in descending order according to the potential of the neuron, and connect them in the column direction. The matrix $\mathbf{A} \in \mathbb{Z}^{T_{\max} \times F_{\text{in}}H_{\text{in}}W_{\text{in}}}$ of the time series internal representation of the layer immediately before the final pooling layer is defined, where $a_{m,n}$ of $\mathbf{A}$ contains the number of the label of the class corresponding to the neuron. The subscripts of the components, m and n, contain any natural number less than or equal to the size of the matrix. Here, it is assumed that each class label is represented by a different integer. The vertical (column) direction means time, the 1st row corresponds to the first time-step, and the $T_{\max}$-th row corresponds to the last time-step. In the horizontal (row) direction, the 1st column corresponds to the maximum in the order of the value of the neuron potential, and the $F_{\text{in}}H_{\text{in}}W_{\text{in}}$-th column corresponds to the minimum.

Set any natural number $L_{\text{p}} \in \mathbb{N}$ less than or equal to $T_{\max}$. L_{p} is a variable for cutting out a part of $\mathbf{A}$, and cuts off the components in the time step of the former part that is not important for the judgment of rejection. The matrix $\mathbf{A}_{\text{p}} \in \mathbb{Z}^{(L_{\text{p}}+1) \times L_{\text{p}}}$ cut out from the $(T_{\max} - L_{\text{p}})$-th row to the $T_{\max}$-th row and from the 1st column to the L_{p}-th column of $\mathbf{A}$ is expressed as:

$$\mathbf{A}_{\text{p}} = \begin{pmatrix} a_{T_{\max}-L_{\text{p}},1} & a_{T_{\max}-L_{\text{p}},2} & \cdots & a_{T_{\max}-L_{\text{p}},L_{\text{p}}} \\ a_{T_{\max}-L_{\text{p}}+1,1} & a_{T_{\max}-L_{\text{p}}+1,2} & \cdots & a_{T_{\max}-L_{\text{p}}+1,L_{\text{p}}} \\ \vdots & \vdots & \ddots & \vdots \\ a_{T_{\max},1} & a_{T_{\max},2} & \cdots & a_{T_{\max},L_{\text{p}}} \end{pmatrix}. \tag{5}$$

Let $\mathbf{\Delta A}_{\text{p}} \in \mathbb{Z}^{L_{\text{p}} \times L_{\text{p}}}$ be the matrix obtained by taking the difference in the time (vertical) direction of $\mathbf{A}_{\text{p}}$, which is defined as:

$$\mathbf{\Delta A}_{\text{p}} = \begin{pmatrix} \Delta a_{T_{\max}-L_{\text{p}}+1,1} & \Delta a_{T_{\max}-L_{\text{p}}+1,2} & \cdots & \Delta a_{T_{\max}-L_{\text{p}}+1,L_{\text{p}}} \\ \Delta a_{T_{\max}-L_{\text{p}}+2,1} & \Delta a_{T_{\max}-L_{\text{p}}+2,2} & \cdots & \Delta a_{T_{\max}-L_{\text{p}}+2,L_{\text{p}}} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta a_{T_{\max},1} & \Delta a_{T_{\max},2} & \cdots & \Delta a_{T_{\max},L_{\text{p}}} \end{pmatrix}, \tag{6}$$

where $\Delta a_{m,n} = a_{m,n} - a_{m-1,n}$. When 1 time-step is advanced, if the label has not changed, it will be 0, and if the label has changed, it will be a number other than 0.

Set any natural number $N \in \mathbb{N}$ less than or equal to L_p. N is a variable that determines the size of the square matrix and lower triangular matrix of $N \times N$, which is the rejection criterion explained below. When the condition is whether all the components of the matrix are 0, the larger N is, the more difficult it is to satisfy the condition, the smaller the coverage, and the smaller the risk. Conversely, the smaller N, the easier it is to meet the conditions, the greater the coverage, and the greater the risk. $\mathbf{\Delta A}_{\mathrm{p},N} \in \mathbb{Z}^{N\times N}$ is a square matrix cut from the $(L_\mathrm{p} - N + 1)$-th row to the L_p-th row and from the 1st column to the N-th column of $\mathbf{\Delta A}_\mathrm{p}$, which is defined as:

$$\mathbf{\Delta A}_{\mathrm{p},N} = \begin{pmatrix} \Delta a_{T_\mathrm{max}-N+1,1} & \Delta a_{T_\mathrm{max}-N+1,2} & \cdots & \Delta a_{T_\mathrm{max}-N+1,N} \\ \Delta a_{T_\mathrm{max}-N+2,1} & \Delta a_{T_\mathrm{max}-N+2,2} & \cdots & \Delta a_{T_\mathrm{max}-N+2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta a_{T_\mathrm{max},1} & \Delta a_{T_\mathrm{max},2} & \cdots & \Delta a_{T_\mathrm{max},N} \end{pmatrix}. \tag{7}$$

Similarly, $\mathbf{\Delta A}_{\mathrm{tri},N} \in \mathbb{Z}^{N\times N}$ is a lower triangular matrix in which the components above the diagonal components of $\mathbf{\Delta A}_{\mathrm{p},N}$ are set to 0, which is defined as:

$$\mathbf{\Delta A}_{\mathrm{tri},N} = \begin{pmatrix} \Delta a_{T_\mathrm{max}-N+1,1} & 0 & \cdots & 0 \\ \Delta a_{T_\mathrm{max}-N+2,1} & \Delta a_{T_\mathrm{max}-N+2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Delta a_{T_\mathrm{max},1} & \Delta a_{T_\mathrm{max},2} & \cdots & \Delta a_{T_\mathrm{max},N} \end{pmatrix}. \tag{8}$$

The simplest rejection criteria is whether all components of $\mathbf{\Delta A}_{\mathrm{p},N}$ or $\mathbf{\Delta A}_{\mathrm{tri},N}$ are 0 or not in the following:

$$g(x) = \begin{cases} 1 & \text{if } \mathbf{\Delta A}_{\mathrm{p},N} \text{ or } \mathbf{\Delta A}_{\mathrm{tri},N} = \mathbf{0}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

We call these rejection method "square matrix method" and "lower triangular matrix method". However, these methods have drawbacks. The coverage delimiter is set to the L_p stage at the maximum, and any value cannot be selected for coverage, it becomes discrete, and the risk and coverage values cannot be flexibly selected. Therefore, we propose the following method.

First, $\mathbf{\Delta A}_\mathrm{p}$ is binarized to the matrix $\mathbf{B} \in \mathbb{Z}^{L_\mathrm{p}\times L_\mathrm{p}}$. All non-zero components of the matrix are set to 1. This is to express the presence or absence of a change in the label as a binary value. The component of $\mathbf{B}$ is denoted as:

$$b_{m,n} = \begin{cases} 0 & \text{if } \Delta a_{T_\mathrm{max}-L_\mathrm{p}+m,n} = 0, \\ 1 & \text{otherwise.} \end{cases} \tag{10}$$

Prepare a matrix $\mathbf{F} \in \mathbb{R}^{L_\mathrm{p}\times L_\mathrm{p}}$ with the same size as $\mathbf{B}$. $\mathbf{F}$ acts like a filter to make a rejection decision. Take the Hadamard product $\mathbf{H} \in \mathbb{R}^{L_\mathrm{p}\times L_\mathrm{p}}$ of $\mathbf{B}$ and $\mathbf{F}$. After that, the sum $S_\mathrm{H} \in \mathbb{R}$ of the components $h_{m,n}$ of $\mathbf{H}$ is taken. S_H is the value of the total change of the label, in which the change of the label in the range of interest of the time series internal representation is weighted by $\mathbf{F}$. $\mathbf{H}$ and S_H are defined as:

$$\mathbf{H} = \begin{pmatrix} b_{1,1} & \cdots & b_{1,L_\mathrm{p}} \\ b_{2,1} & \cdots & b_{2,L_\mathrm{p}} \\ \vdots & \ddots & \vdots \\ b_{L_\mathrm{p},1} & \cdots & b_{L_\mathrm{p},L_\mathrm{p}} \end{pmatrix} \odot \begin{pmatrix} f_{1,1} & \cdots & f_{1,L_\mathrm{p}} \\ f_{2,1} & \cdots & f_{2,L_\mathrm{p}} \\ \vdots & \ddots & \vdots \\ f_{L_\mathrm{p},1} & \cdots & f_{L_\mathrm{p},L_\mathrm{p}} \end{pmatrix}, \tag{11}$$

$$S_{\mathrm{H}} = \sum_{m=1}^{L_{\mathrm{p}}} \sum_{n=1}^{L_{\mathrm{p}}} h_{m,n}. \tag{12}$$

Then, S_{H} is calculated for all the data in the dataset. Here, let the total number of data in the dataset be $L_{\mathrm{ds}} \in \mathbb{N}$, and prepare the matrix $\mathbf{S} \in \mathbb{R}^{L_{\mathrm{ds}}}$. All the data here refers to the entire training data at the time of optimization and the entire test data at the time of evaluation. $\mathbf{S}$ is expressed as:

$$\mathbf{S} = [s_1, s_2, \ldots, s_{L_{\mathrm{ds}}}], \tag{13}$$

where s_i contains the value of S_{H} calculated for x_i. Since coverage is expressed between 0 and 1, so if the coverage is c, the model's predictions for the entire $100(1-c)\%$ data from the smaller value are rejected. Let s_{th} be the $L_{\mathrm{ds}}(1-c)$-th smallest component of $\mathbf{S}$, and the selection function g is in Eq. (2), where $\kappa_{\mathrm{f}} = s_i$, and $\theta = s_{\mathrm{th}}$.

Assuming that the number of coverage delimiters is $N_{\mathrm{sep}} \in \mathbb{N}$, the coverage matrix is $\mathbf{C} \in \mathbb{R}^{N_{\mathrm{sep}}}$, and the corresponding risk matrix is $\mathbf{R} \in \mathbb{R}^{N_{\mathrm{sep}}}$. The element c of the coverage matrix $\mathbf{C}$ is calculated in Eq. (4), and the element r of the risk matrix $\mathbf{R}$ is calculated in Eq. (3).

Here, it is necessary to adjust the components of the matrix $\mathbf{F}$ in order to efficiently reject and bring the RC curve closer to the ideal one. The method is explained below.

Fig. 2(a) shows an example of the RC curve and RC area before optimization. The RC area is calculated by summing up the rectangles with the difference between adjacent coverage as the height and the larger adjacent risk as the width. Let $(r_{n_{\mathrm{s}}}, c_{n_{\mathrm{s}}})$ be (r, c) separated by n_{s} from $(0, 0)$. The obtained RC area $S_{\mathrm{rc}} \in \mathbb{R}$ is denoted as:

$$S_{\mathrm{rc}}(\mathbf{R}, \mathbf{C}) = \sum_{n_s=0}^{N_{\mathrm{sep}}} \begin{cases} (c_{n_s+1} - c_{n_s}) r_{n_s+1} & \text{if } r_{n_s} < r_{n_s+1}, \\ (c_{n_s+1} - c_{n_s}) r_{n_s} & \text{otherwise.} \end{cases} \tag{14}$$

Minimizing this RC area corresponds to improving the efficiency of the RC curve. The gradient-free black-box optimization library Nevergrad [9] is used to minimize

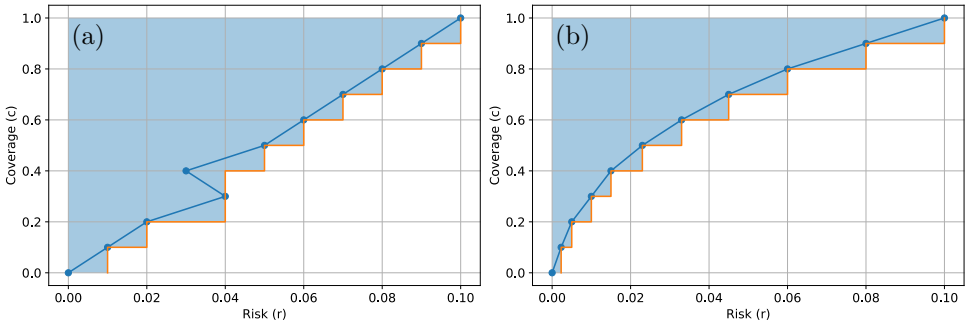

Fig. 2 *Examples of RC curve and RC area. (a) RC curve and RC area before optimization; (b) RC curve and RC area after optimization.*

the area. The parameter handled by Nevergrad is only the matrix $\mathbf{F}$, and the objective function that returns the RC area S_{rc} is minimized as follows:

$$\mathbf{F}^* = \underset{\mathbf{F}}{\operatorname{argmin}}\, S_{\mathrm{rc}}(\mathbf{R}_{g_{\mathbf{F}}}, \mathbf{C}). \tag{15}$$

Fig. 2(b) shows an example of the optimized RC curve and RC area.

Here we explain the optimization with constraints. In the case of constrained optimization, use the matrix $\mathbf{F}_{\mathrm{const}} \in \mathbb{R}^{L_{\mathrm{p}} \times L_{\mathrm{p}}}$ instead of the matrix $\mathbf{F}$. Then, black-box optimization is performed so as to minimize the RC area S_{rc} while satisfying the conditions in the following:

$$\mathbf{F}_{\mathrm{const}} = \begin{pmatrix} w_{L_{\mathrm{p}}-1} & w_{L_{\mathrm{p}}-2} & \cdots & w_1 & w_0 \\ w_{L_{\mathrm{p}}} & w_{L_{\mathrm{p}}-1} & \cdots & w_2 & w_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{2L_{\mathrm{p}}-3} & w_{2L_{\mathrm{p}}-4} & \cdots & w_{L_{\mathrm{p}}-1} & w_{L_{\mathrm{p}}-2} \\ w_{2L_{\mathrm{p}}-2} & w_{2L_{\mathrm{p}}-3} & \cdots & w_{L_{\mathrm{p}}} & w_{L_{\mathrm{p}}-1} \end{pmatrix}, \tag{16}$$

$$\text{s.t.} \quad w_0 < w_1 < \cdots < w_{2L_{\mathrm{p}}-3} < w_{2L_{\mathrm{p}}-2}. \tag{17}$$

The methods whose rejection criteria are adjusted by optimizing the matrices $\mathbf{F}$ and $\mathbf{F}_{\mathrm{const}}$ are called "proposed without constraints" and "proposed with constraints", respectively.

In general, the SNN model in which the final layer is the pooling layer does not include the softmax function in the model, unlike the SNN model in which the final layer is the fully connected layer. Therefore, the softmax function is applied to the internal representation of the final pooling layer, and softmax response [10] that serves as the baseline is used. Softmax response is a method that uses the maximum value of the softmax function as the reliability. Let the total number of neurons in the final pooling layer be $N_{\mathrm{pool}} \in \mathbb{N}$. Each class corresponds to neurons of $N_{\mathrm{pool}}/L_{\mathrm{class}}$. Then a row vector in which the outputs of the final pooling layer are arranged as they are for each time-step is created and each row vector is combined in the column direction. The matrix $\mathbf{P} \in \mathbb{R}^{T_{\max} \times N_{\mathrm{pool}}}$ of the time series internal representation of the final pooling layer is defined, where $p_{m,n}$ of $\mathbf{P}$ contains a real value that represents the value of the neuron's potential. The vertical (column) direction means time, and the 1st row corresponds to the first time-step, and the $T_{\max}$-th row corresponds to the last time-step. The horizontal (row) direction corresponds to the position of the neuron.

Let $\mathbf{Q} \in \mathbb{R}^{L_{\mathrm{class}}}$ be the matrix that extracts the neuron that has the maximum potential among the neurons corresponding to each class in the $T_{\max}$-th row of $\mathbf{P}$. The component of $\mathbf{Q}$ is denoted as:

$$q_k = \max\{p_{T_{\max}, N_{\mathrm{p/c}}(k-1)+1}, p_{T_{\max}, N_{\mathrm{p/c}}(k-1)+2}, \ldots, p_{T_{\max}, N_{\mathrm{p/c}}k}\}, \tag{18}$$

$$\text{where} \quad N_{\mathrm{p/c}} = N_{\mathrm{pool}}/L_{\mathrm{class}}.$$

$\mathbf{Q}$ has a large value for each component, and if the softmax function is applied, an extreme result is often obtained. Therefore, we use the matrix $\mathbf{Q}' \in \mathbb{R}^{L_{\mathrm{class}}}$, where

each component is divided by the mean of all components. The component of $\mathbf{Q}'$ is expressed as:

$$q'_k = \frac{q_k}{\frac{1}{L_{\text{class}}} \sum_{j=1}^{L_{\text{class}}} q_j}. \tag{19}$$

Applying the softmax function to $\mathbf{Q}'$ gives $\boldsymbol{\varphi}\mathbf{Q}' \in \mathbb{R}^{L_{\text{class}}}$. The following $\boldsymbol{\varphi}\mathbf{Q}'$ is obtained. Its component $\varphi q'_k$ is in the following:

$$\boldsymbol{\varphi}\mathbf{Q}' = [\varphi q'_1, \varphi q'_2, \ldots, \varphi q'_{L_{\text{class}}}], \tag{20}$$

$$\varphi q'_k = softmax(q'_k) = \frac{\exp(q'_k)}{\sum_{t=1}^{L_{\text{class}}} \exp(q'_t)}. \tag{21}$$

Selective classification is performed using $\boldsymbol{\varphi}\mathbf{Q}'$. $\mathbf{S}$ is created, and s_i in Eq. (13) contains the maximum value of the components of $\boldsymbol{\varphi}\mathbf{Q}'$ for x_i, $\max\{\varphi q'_k\}_{k=1}^{L_{\text{class}}}$. The selection function g is in Eq. (2), where $\kappa_{\text{f}} = s_i$, and $\theta = s_{\text{th}}$. This rejection method will be called "softmax response".

3.3 The case where the final layer is a fully connected layer

In this section, we propose three methods, "softmax response", "softmax difference", and "proposed". Selective classification is performed using the time series internal representation in the fully connected layer, which is the final layer. In general, the total number of neurons in the final fully connected layer is equal to the number of classes L_{class}. Row vectors of the outputs of the final fully connected layer are created and are arranged in the column direction, as they are, for each time-step. The matrix $\mathbf{D} \in \mathbb{R}^{T_{\max} \times L_{\text{class}}}$ of the time series internal representation of the final fully connected layer is defined, where $d_{m,n}$ of $\mathbf{D}$ contains a real value that represents the value of the neuron's potential. The vertical (column) direction means time, the 1st row corresponds to the first time-step, and the $T_{\max}$-th row corresponds to the last time-step. The horizontal (row) direction corresponds to the position of the neuron. The matrix $\boldsymbol{\varphi}\mathbf{D} \in \mathbb{R}^{T_{\max} \times L_{\text{class}}}$ is obtained by applying a softmax function in the horizontal (row) direction to the matrix $\mathbf{D}$. The component of $\boldsymbol{\varphi}\mathbf{D}$ is denoted as:

$$\varphi d_{m,n} = softmax(d_{m,n}) = \frac{\exp(d_{m,n})}{\sum_{u=1}^{L_{\text{class}}} \exp(d_{m,u})}. \tag{22}$$

Selective classification is performed using $\boldsymbol{\varphi}\mathbf{D}$. $\mathbf{S}$ is created, and s_i in Eq. (13) contains the maximum value of the components of $T_{\max}$-th row of $\boldsymbol{\varphi}\mathbf{D}$ for x_i, $\max\{\varphi d_{T_{\max},k}\}_{k=1}^{L_{\text{class}}}$. The selection function g is in Eq. (2), where $\kappa_{\text{f}} = s_i$, and $\theta = s_{\text{th}}$. This rejection method is referred as "softmax response" in this paper.

Let $\mathbf{E} \in \mathbb{R}^{T_{\max} \times L_{\text{class}}}$ be a matrix in which each column vector of $\boldsymbol{\varphi}\mathbf{D}$ is fixed and arranged in descending order by the value of $\varphi d_{T_{\max},n}$ in the final time-step. This is because the potential values of the neurons and their rankings are important for the rejection decision. Here, the vertical (column) matrix is preserved and only the horizontal (row) order is sorted according to the value of the potential in the final time-step. That is, the same column represents the value of the neuron's potential for the same label. The 1st column is for the class with the maximum potential

in the final time-step, and the L_{class}-th column is for the class with the minimum potential in the final time-step.

Here, the second rejection criteria is explained. It employs the value of $(e_{T_{\max},1} - e_{T_{\max},2})/e_{T_{\max},1}$ of $\mathbf{E}$. $\mathbf{S}$ is created, and s_i contains this value for x_i. The selection function g is in Eq. (2), where $\kappa_{\text{f}} = s_i$, and $\theta = s_{\text{th}}$. This rejection method is referred as "softmax difference".

The third method considers the entire time series in the internal representation. Set any divisor $T_{\text{g}} \in \mathbb{N}$ of $T_{\max}$. T_{g} is a variable for compressing $\mathbf{E}$ in the time direction, and can remove and smooth out the small perturbations in the potential that are not important for the rejection decision. The matrix $\mathbf{G} \in \mathbb{R}^{T_{\text{g}} \times L_{\text{class}}}$ is the mean of $\mathbf{E}$ every $T_{\text{m/g}}$ time-steps and its component is denoted as:

$$g_{m,n} = \frac{1}{T_{\text{m/g}}} \sum_{v=T_{\text{m/g}}(m-1)+1}^{T_{\text{m/g}}m} e_{v,n}, \tag{23}$$

$$\text{where} \quad T_{\text{m/g}} = T_{\max}/T_{\text{g}}.$$

Prepare a matrix $\mathbf{F} \in \mathbb{R}^{T_{\text{g}} \times L_{\text{class}}}$ with the same size as $\mathbf{G}$. Take the Hadamard product $\mathbf{H} \in \mathbb{R}^{T_{\text{g}} \times L_{\text{class}}}$ of $\mathbf{G}$ and $\mathbf{F}$. After that, the sum of the components of $\mathbf{H}$ is taken as S_{H}. The flow after that is the same as in Section 3.2. However, s_i in Eq. (13) contains the value of S_{H}. Unlike $\mathbf{B}$, whose component is 0 or 1, the component of $\mathbf{G}$ is a real value, so constraints like $\mathbf{F}_{\text{const}}$ do not work well. This rejection method is referred as "proposed".

4. Evaluation experiments

4.1 Experimental overview

In the experiments, the size of the RC area, risk, and coverage based on the rejection criteria of the proposed method are compared with softmax response as the baseline. The RC area can be used to evaluate the balance between risk and coverage of the entire RC curve. $[0.05, 0.1, 0.15, \ldots, 0.9, 0.95]$ is used as the coverage matrix for calculating the RC area, except for "square matrix method" and "lower triangular matrix method". DCSNN (deep convolutional spiking neural network) [8, 11] is used as a model of SNN whose the final layer is a pooling layer. ASF-BP (accumulated spiking flow-backpropagation) [12, 13] is used as a model of SNN whose the final layer is a fully connected layer. DCSNN is an SNN composed of three convolution layers and three pooling layers. ASF-BP is a learning rule of SNN, and it is possible to learn a model obtained by converting a deep model such as VGG-7 [14] into SNN. Here, we use a model converted to SNN based on VGG-7. The SNN using this learning rule is referred as "ASF-BP".

Tab. I shows the SNN model, the final layer of the model, the dataset, the original data format, the conversion method for handling by SNN, and the number of time-steps after conversion to time series data for experiments. Tab. II shows the parameters related to the proposed method. MNIST [15] is an image dataset of handwritten digits from '0' to '9'. Intensity-to-latency encoding [16] is performed on the MNIST data, which is used as the input for DCSNN. This encoding is

	Experiment 1	Experiment 2
Model	DCSNN	ASF-BP (VGG-7)
Final layer	Global pooling layer	Fully connected layer
Dataset	MNIST	CIFAR10-DVS
Original format	Static	Time series
Conversion	Intensity-to-latency encoding	Accumulation
Time-steps	15	100

Tab. I *Models and data used for experiments.*

	Experiment 1	Experiment 2
$\mathbf{C}$	$[0.05, 0.1, \ldots, 0.95]$	$[0.05, 0.1, \ldots, 0.95]$
N_{sep}	19	19
N_{pool}	200	–
L_{class}	10	10
L_{p}	10	–
$L_{\text{ds}}(training)$	8,000	9,000
$L_{\text{ds}}(test)$	2,000	1,000
T_{g}	–	10
T_{max}	15	100
F_{in}	200	–
H_{in}	4	–
W_{in}	4	–

Tab. II *Parameters related to the proposed methods.*

a process necessary for handling data that does not have temporal information in SNN. CIFAR10-DVS [17] is a recording of the moving CIFAR-10 [18] image displayed on the LCD monitor using an event camera, dynamic vision sensor (DVS). CIFAR-10 is an image dataset of color photographs of objects such as vehicles and animals. Each data is a spike train and time series data that can be used as it is as an input of SNN. The spike trains are accumulated at certain intervals to compress them into 100 time-steps.

4.1.1 Experiment 1 (Experiment of DCSNN)

Fig. 3 shows the architecture of DCSNN. There is a decision-making map after the final layer of DCSNN, the pooling layer (P3), and the class corresponding to the neuron with the largest potential in the pooling layer (P3) is used as the label for model prediction. The layer immediately before the pooling layer (P3) is the convolution layer (C3), and the time series internal representation in this layer (C3) is used.

The experiment using DCSNN as a model and MNIST as a dataset uses the method in Section 3.2. Nevergrad's NGOpt [19, 20] is used as the solver for black-box optimization. The *budget*, which means the number of search attempts, is set to 5,000. NGOpt serves as a selector for optimization algorithms [21]. Based on

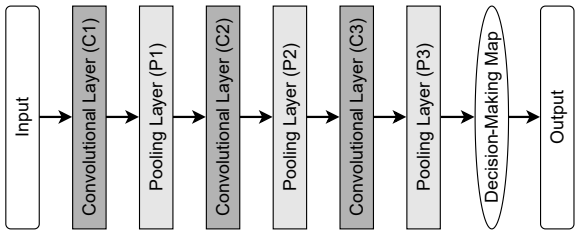

Fig. 3 *Architecture of DCSNN.*

existing benchmark data, either of the following two methods is selected. The first one is an automatic solver selection using high-level problem information. The second one is combination of optimization algorithms. There are more than 20 types of solvers, such as differential evolution [22] and Bayesian optimization [23], and NGOpt will select them according to the *budget*.

Originally, MNIST has 60,000 training data and 10,000 test data. This 60,000 training data is used for learning for DCSNN to perform the original classification task. Here, 8,000 of the 10,000 test data are used for optimization as training data for conducting experiments, and the remaining 2,000 are used as test data for conducting experiments. For cross-validation, the training data and test data are divided into five different patterns.

4.1.2 Experiment 2 (Experiment of ASF-BP)

Fig. 4 shows the architecture of ASF-BP. There is a softmax function after the fully connected layer (F2), which is the final layer of ASF-BP. The class corresponding to the neuron with the largest potential in the fully connected layer (F2) is used as the label for model prediction. We use the time series internal representation in this layer (F2).

For experiment using ASF-BP as a model and CIFAR10-DVS as a dataset, the method in Section 3.3 is used. The solver and *budget* for black-box optimization are the same as in Section 4.1.1. Originally, CIFAR10-DVS has 10,000 data. Although there is no fixed ratio between training data and test data, Hao Wu et al. [12] uses 9,000 data for training in order to perform the original classification task in ASF-BP. Here, according to this, 9,000 is used as training data for con-

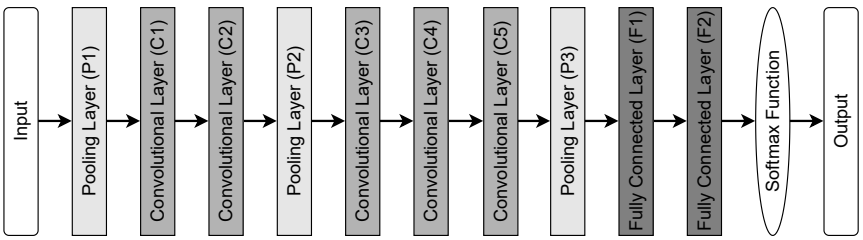

Fig. 4 *Architecture of ASF-BP.*

ducting experiments, and the remaining 1,000 is used as test data for conducting experiments. Set five types of random seeds and perform optimization five times.

4.2 Experimental results

4.2.1 Experiment 1

Tab. III shows the RC area in the training data and test data of each method. Since cross-validation was performed, the mean and standard deviation are shown. The RC curves in the test data are shown in Fig. 5(a). Fig. 5(b) shows the RC curves in the training data. The mean obtained by cross-validation is shown. In the test data, *proposed with constraints* showed the minimum RC area. Additionally, lower triangular matrix method and *proposed with constraints* showed a smaller RC area than softmax response, which is the baseline. In the training data, *proposed without constraints* showed the minimum RC area.

Method	Training data ($\times 10^{-3}$)	Test data ($\times 10^{-3}$)
Softmax response (baseline)	7.809(0.925)	8.042(3.806)
Square matrix method	8.552(0.877)	8.914(3.652)
Lower triangular matrix method	7.048(0.642)	7.511(2.781)
Proposed without constraints	**4.879**(**0.720**)	9.106(3.559)
Proposed with constraints	7.157(0.724)	**7.383**(**3.253**)

Tab. III *RC area in Experiment 1.*

4.2.2 Experiment 2

Tab. IV shows the RC area in the training data and test data of each method. In the proposed method, five types of random seeds were set and optimization was performed five times, so the mean and standard deviation were shown. Tab. V shows the risks for coverage in the test data. The risks for coverage in the training data is shown in Tab. VI. In the test data, *proposed* showed the smallest RC area. Regarding the risk for coverage in the test data, the risk of *proposed* was lower than that of softmax response, especially when the coverage was 0.4 or less. For the risk in the training data, the risk of *proposed* was equal to or smaller than the risk of the other methods in all coverage.

Method	Training data ($\times 10^{-2}$)	Test data ($\times 10^{-2}$)
Softmax response (baseline)	3.364	22.04
Softmax difference	2.881	22.13
Proposed	**2.733**(**0.007**)	**21.86**(**0.087**)

Tab. IV *RC area in Experiment 2.*

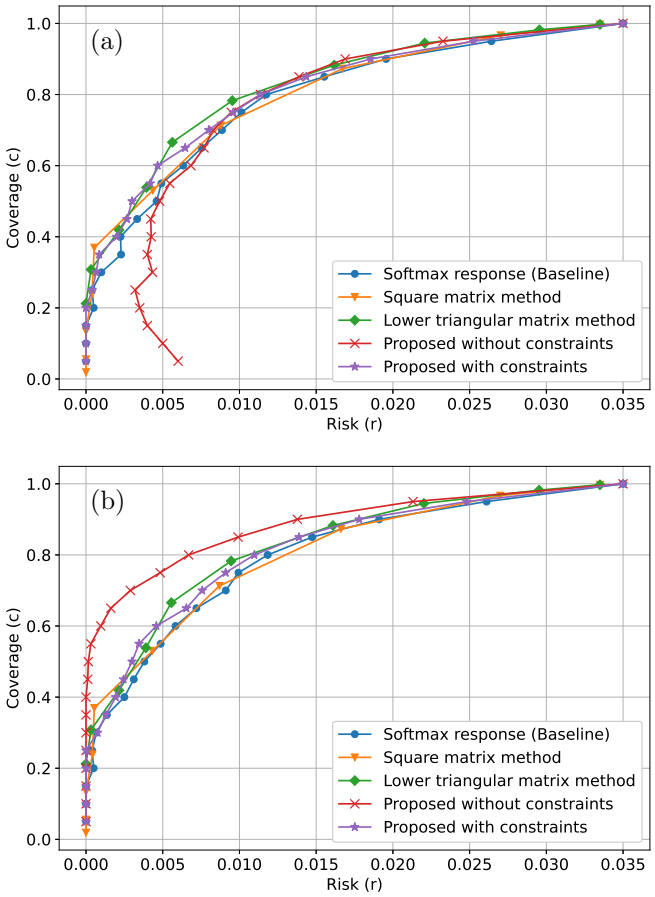

Fig. 5 *RC curves in Experiment 1. (a) RC curves in test data; (b) RC curves in training data.*

Method	Coverage [%]				
	0.1	0.2	0.3	0.4	0.5
Softmax response (baseline)	2.000	7.000	11.67	18.25	**22.20**
Softmax difference	**1.000**	5.500	**10.00**	18.25	23.60
Proposed	**1.000**	**4.800**	10.60	**16.75**	23.20

Tab. V *Risk of test data in Experiment 2.*

4.3 Discussions

4.3.1 Experiment 1

The proposed method without constraints was able to minimize the RC area in the training data. However, in the test data, the RC area could not be made

Method	Coverage [%]				
	0.2	0.3	0.4	0.5	0.6
Softmax response (baseline)	0.000	0.222	0.472	1.044	2.296
Softmax difference	0.000	0.111	0.306	0.600	1.389
Proposed	0.000	**0.074**	**0.250**	**0.502**	**1.085**

Tab. VI *Risk of training data in Experiment 2.*

smaller than the proposed method with constraints. It is considered that the proposed method without constraints was overfitted to the training data and the generalization ability was insufficient. The RC curve in the training data of the proposed method is shown in Fig. 5(b), and in the test data is shown in Fig. 5(a). According to these figures, without constraints, the RC curve in the training data has a risk of almost zero, especially in the range where the coverage is less than 0.5. The RC curve in the test data has the smallest risk at the point of coverage of 0.25, and after that, the risk does not decrease even if the coverage is narrowed. This is not an efficient rejection criteria. On the other hand, with constraints, the degree of risk reduction for coverage in the training data is modest. Even in the test data, the risk for coverage decreases almost monotonously, and it is considered that more efficient rejection criteria can be set compared to the case without constraints.

Fig. 6(a) shows the heatmap of the matrix $\mathbf{F}$ when the optimization was performed without constraints. Fig. 6(b) shows the heatmap of the matrix $\mathbf{F}_{\text{const}}$ when the optimization was performed with constraints. According to Fig. 6(a), it can be seen that the change in the label in the lower part in the vertical (column) direction, that is, in the latter part of the time step, greatly contributes to the judgment of whether or not the prediction should be rejected. It can also be seen that the change in the label in the left part in the horizontal (row) direction, that is, in the upper part of the potential of the neuron, has a large effect. In particular, the large components of the matrix $\mathbf{F}$ are concentrated in the parts where the time-

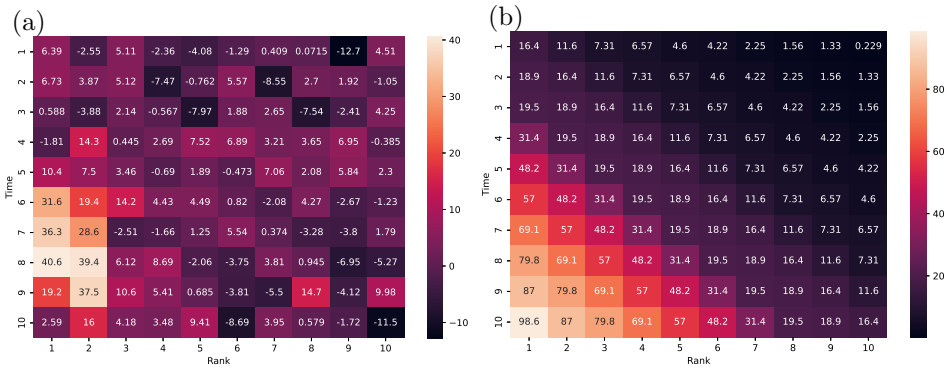

Fig. 6 *Heatmap of the matrix* $\mathbf{F}$ *of the proposed method in Experiment 1. (a) Unconstrained matrix* $\mathbf{F}$*; (b) Constrained matrix* $\mathbf{F}_{\text{const}}$*.*

steps are 6 to 10 and the ranks of potential's values are 1st and 2nd. Both of them are natural. This is because the label corresponding to the 1st and 2nd neurons of potential's values in the latter part of time-steps, greatly affects the prediction result. However, if we look closely at Fig. 6(a), we can notice a pattern in which the difference between the component with a high weight and the component with a low weight is large, the positive and negative signs differ between adjacent components, and the values differ greatly. It is considered that these factors lead to the difference in generalization ability when compared with the constrained matrix $\mathbf{F}_{\text{const}}$ in Fig. 6(b). Depending on the constraints, the component corresponding to the higher value of the neuron's potential than the lower value is always given more weight in the same row. In the same column, more weight is always given to the component that corresponds to the later part of the time step than the earlier part. It is considered that these strong constraints contribute to the improvement of generalization ability.

4.3.2 Experinment 2

By minimizing the RC area in the training data, the results exceeding the baseline were obtained in the test data. Fig. 7 shows the heatmap of the matrix $\mathbf{F}$ when optimized. $f_{10,1}$ is the largest component and $f_{10,2}$ is the smallest component in the matrix $\mathbf{F}$. This indicates that the matrix $\mathbf{F}$ has a property close to softmax response. This is because $f_{10,1}$ is the component corresponding to the mean of 10 time-steps including the final time-step of the neuron with the maximum potential in the final time-step. It also shows that it has a property close to softmax difference. This is because the relationship between the positive $f_{10,1}$ and the negative $f_{10,2}$ is similar to $(e_{100,1} - e_{100,2})/e_{100,1}$. It is considered that the remaining components of the 10×10 matrix, while including the above two properties, provide more flexible expressive power as a rejection criteria.

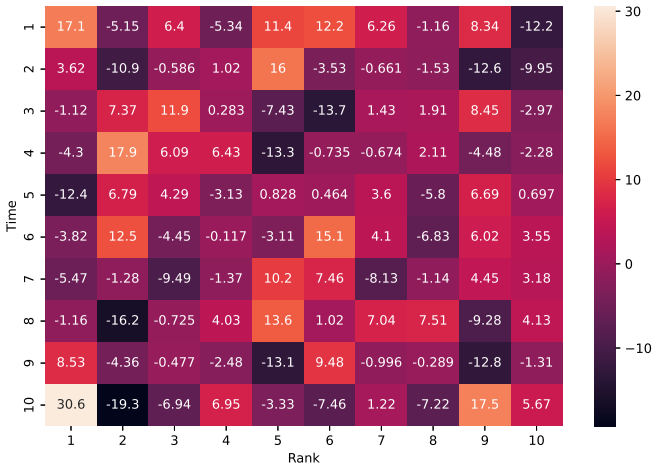

Fig. 7 *Heatmap of the matrix* $\mathbf{F}$ *of the proposed method in Experiment 2.*

5. Conclusion

In this paper, we proposed rejection methods using internal representation of SNN, selective classification using them, and efficiency improvement method of RC curve. Most of existing studies on selective classification and prediction reliability focus on quantifying the reliability of the model's output predictions for each piece of data. On the other hand, in this paper, we focused on improving the efficiency of the RC curve for a certain number of data so that the rejection rate and the expected error rate to be allowed can be selected advantageously. The proposed method uses the internal representation of the model, ranks the prediction results, and rejects them at an arbitrary rate. Using the proposed method and the existing method, experiments were conducted to compare the size of the RC area according to the rejection criteria. As a result, it was confirmed that the proposed method enables selective classification based on efficient rejection criteria while ensuring generalization ability.

References

[1] GAWLIKOWSKI J., TASSI C.R.N., ALI M., LEE J., HUMT M., FENG J., KRUSPE A., TRIEBEL R., JUNG P., ROSCHER R., SHAHZAD M., YANG W., BAMLER R., ZHU X.X. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021, doi: 10.48550/arXiv.2107.03342.

[2] MALININ A., GALES M. Predictive Uncertainty Estimation via Prior Networks. *Advances in Neural Information Processing Systems*, 2018, 31, Available from: https://proceedings.neurips.cc/paper/2018/file/3ea2db50e62ceefceaf70a9d9a56a6f4-Paper.pdf

[3] GAL Y., GHAHRAMANI Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 1050–1059. Available from: http://proceedings.mlr.press/v48/gal16.pdf

[4] VALDENEGRO-TORO M. Deep sub-ensembles for fast uncertainty estimation in image classification. *arXiv preprint arXiv:1910.08168*, 2019, doi: 10.48550/arXiv.1910.08168.

[5] MOLCHANOV D., LYZHOV A., MOLCHANOVA Y., ASHUKHA A., VETROV D. Greedy policy search: A simple baseline for learnable test-time augmentation. In: *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, 2020, 124, pp. 1308–1317, Available from: http://proceedings.mlr.press/v124/lyzhov20a/lyzhov20a.pdf

[6] EL-YANIV R., WIENER Y. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*. 2010, 11(5), pp. 1605–1641, Available from: http://jmlr.org/papers/v11/el-yaniv10a.html

[7] TAVANAEI A., GHODRATI M., KHERADPISHEH S.R., MASQUELIER T., MAIDA A. Deep learning in spiking neural networks. *Neural Networks*, 2019, 111, pp. 47–63, doi: 10.1016/j.neunet.2018.12.002.

[8] MOZAFARI M., GANJTABESH M., NOWZARI-DALINI A., MASQUELIER T. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in Neuroscience*, 2019, 13(625), doi: 10.3389/fnins.2019.00625.

[9] RAPIN J., TEYTAUD O. Nevergrad - A gradient-free optimization platform [software]. 2018 [accessed 2022-04-03]. Available from: https://GitHub.com/FacebookResearch/Nevergrad

[10] GEIFMAN Y., EL-YANIV R. Selective classification for deep neural networks. *arXiv preprint arXiv:1705.08500*, 2017, doi: 10.48550/arXiv.1705.08500.

[11] MOZAFARI M., GANJTABESH M., NOWZARI-DALINI A., THORPE S.J., MASQUELIER T. Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognition*, 2019, 94, pp. 87–95, doi: 10.1016/j.patcog.2019.05.015.

[12] WU H., ZHANG Y., WENG W., ZHANG Y., XIONG Z. ZHA Z.J., SUN X., WU F. Training spiking neural networks with accumulated spiking flow. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, 25(12) pp. 10320–10328. Available from: https://www.aaai.org/AAAI21Papers/AAAI-4138.WuHao.pdf

[13] LEE C., SARWAR S.S., PANDA P., SRINIVASAN G., ROY K. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 2020, 14(119), doi: 10.3389/fnins.2020.00119.

[14] SIMONYAN K., ZISSERMAN A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014, doi: 10.48550/arXiv.1409.1556.

[15] LECUN Y. BOTTOU L. BENGIO Y. HAFFNER P. Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*, 1998, 86(11), pp. 2278–2324, doi: 10.1109/5.726791.

[16] GAUTRAIS J., THORPE S. Rate coding versus temporal order coding: a theoretical approach. *Biosystems*, 1998, 48(1–3), pp. 57–65, doi: 10.1016/S0303-2647(98)00050-1.

[17] LI H., LIU H., JI X., LI G. SHI L. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in Neuroscience*, 2017, 11(309), doi: 10.3389/fnins.2017.00309.

[18] KRIZHEVSKY A. HINTON G. *Learning Multiple Layers of Features from Tiny Images* [online]. Toronto, 2009, Master's Thesis, University of Toronto in Toronto[viewed 2022-04-03]. Available from: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[19] BENNET P., DOERR C., MOREAU A., RAPIN J., TEYTAUD F., TEYTAUD O. Nevergrad: black-box optimization platform. *ACM SIGEVOlution*, 2021, 14(1), pp. 8–15, doi: 10.1145/3460310.3460312.

[20] RAPIN J., BENNET P., CENTENO E., HAZIZA D., MOREAU A., TEYTAUD O. Open source evolutionary structured optimization. In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, Association for Computing Machinery, 2020, pp. 1599–1607, doi: 10.1145/3377929.3398091.

[21] MEUNIER L., RAKOTOARISON H., WONG P.K., ROZIERE B., RAPIN J., TEYTAUD O., MOREAU A., DOERR C. Black-Box Optimization Revisited: Improving Algorithm Selection Wizards through Massive Benchmarking. *IEEE Transactions on Evolutionary Computation*, 2021, doi: 10.1109/TEVC.2021.3108185.

[22] DAS S., SUGANTHAN P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 2010, 15(1), pp. 4–31, doi: 10.1109/TEVC.2010.2059031.

[23] RAPONI E., WANG H., BUJNY M., BORIA S., DOERR C. High dimensional bayesian optimization assisted by principal component analysis. *International Conference on Parallel Problem Solving from Nature*, 2020, pp. 169–183, doi: 10.1007/978-3-030-58112-1_12.