C^*-algebra Net: A New Approach Generalizing Neural Network Parameters to C^*-algebra

Yuka Hashimoto [1] **Zhao Wang** [1 2] **Tomoko Matsui** [3]

Abstract

We propose a new framework that generalizes the parameters of neural network models to C^*-algebra-valued ones. C^*-algebra is a generalization of the space of complex numbers. A typical example is the space of continuous functions on a compact space. This generalization enables us to combine multiple models continuously and use tools for functions such as regression and integration. Consequently, we can learn features of data efficiently and adapt the models to problems continuously. We apply our framework to practical problems such as density estimation and few-shot learning and show that our framework enables us to learn features of data even with a limited number of samples. Our new framework highlights the potential possibility of applying the theory of C^*-algebra to general neural network models.

1. Introduction

Continuation of neural network models has been discussed and successfully applied to practical problems and theoretical analyses. Chen et al. (2018) proposed to regard the transformation of input variables as a continuous dynamical system, which is called the neural ODE. While the neural ODE focus on the continuation of layers (vertical direction), continuation of units (horizontal direction) is also discussed. By regarding the action of a weight matrix to variables as integrations and using an integral representation of the model, we can use the theory of harmonic analysis to analyze the model theoretically (Candès, 1999; Sonoda & Murata, 2017; Sonoda et al., 2021). In these frameworks, we regard the parameters of a classical model being obtained by a discretization of functions, which enables us to use tools for functions such as derivative and integration for practical applications and theoretical analyses of the model.

In this paper, we propose a brand new framework that generalizes the parameters (weights) of models. Unlike previous works regarding the continuation of neural network models, we do not consider the continuation of the parameters of *a single* classical model to functions. Instead, we generalize each $\mathbb{R}$-valued parameter to a C^*-algebra-valued one, which corresponds to an aggregation of *multiple* classical models and a continuation of the parameters of the multiple models to functions. C^*-algebra is a generalization of the space of complex numbers. Typical examples are the space of continuous functions on a compact space, the space of L^∞ functions on a σ-finite measure space, and the space of bounded linear operators on a Hilbert space. In this paper, we focus on the C^*-algebra of the space of continuous functions on a compact space. Practically, we can use tools for functions such as regression and integration to learn multiple models efficiently and to adapt the models to problems continuously. Theoretically, our new framework highlights the potential possibility of applying the theory of C^*-algebra to general neural network models. Fig. 1 shows an overview of our framework schematically. The application of C^*-algebra to data analysis is discussed by Hashimoto et al. (2021). We focus on the practical applications to neural network in this paper. To the best of our knowledge, this is the first paper that applies the theory of C^*-algebra to neural network models.

Our contributions are as follows:

- We propose a generic framework of neural network with C^*-algebra-valued parameters.
- We propose a gradient descent method to learn the model on C^*-algebra.
- We apply our framework to practical problems such as density estimation and few-shot learning.

Regarding the second one, since C^*-algebra admits a generalization of Hilbert space, which is called Hilbert C^*-module, we can generalize the classical gradient descent method on the Hilbert space associated with the parameters

[1]NTT Network Service Systems Laboratories, NTT Corporation, Tokyo, Japan [2]Institute for Disaster Response Robotics, Waseda University, Tokyo, Japan [3]Department of Statistical Modeling, the Institute of Statistical Mathematics, Tokyo, Japan. Correspondence to: Yuka Hashimoto <yuka.hashimoto.rw@hco.ntt.co.jp>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

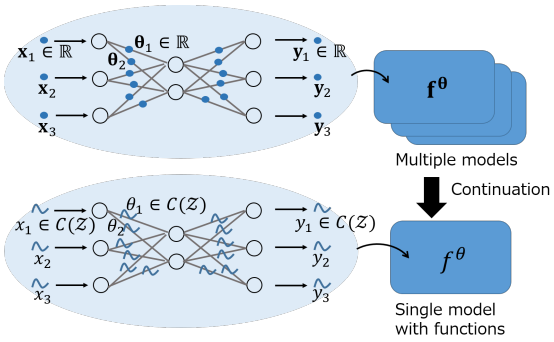


Figure 1. Overview of our framework.

of a model to that on the Hilbert C^*-module associated with the C^*-algebra-valued parameters.

The remainder of this paper is organized as follows. In Section 3, we review mathematical notions related to C^*-algebra and Hilbert C^*-module. In Section 4, we propose a model with C^*-algebra-valued parameters. Then, in Section 5, we discuss practical applications and show numerical results. We conclude the paper in Section 6. The source code of this paper is available at `https://www.rd.ntt/e/ns/qos/person/hashimoto/code_c_star_net.zip`.

Notations Bold letters denote $\mathbb{R}$-valued objects or maps from $\mathbb{R}^{d_1}$ to $\mathbb{R}^{d_2}$ for some $d_1, d_2 \in \mathbb{N}$. Italic letters denote $\mathcal{A}$-valued objects or maps from $\mathcal{A}^{d_1}$ to $\mathcal{A}^{d_2}$.

2. Motivation of applying C^*-algebra to neural networks

Because C^*-algebra and Hilbert C^*-module are natural generalizations of the space of complex numbers and Hilbert space, we can naturally generalize neural networks by using C^*-algebra. Let $\mathcal{A}$ be the C^*-algebra of the space of continuous functions on a compact space. Then, by generalizing the real-valued parameters in the neural networks to C^*-algebra-valued ones, we can combine multiple real-valued models continuously. For example, we can apply our framework to ensemble learning. In standard ensemble learning, we learn multiple models separately and aggregate the results of the models after finishing the learning process. This approach may not be efficient since the multiple models do not interact during the learning process, although each model is learned for the same or related task. On the other hand, by using the $\mathcal{A}$-valued parameters, we introduce a continuous dependence between different models and can learn the multiple models with interactions. As a result, we expect that our method outperforms the standard ensemble learning. For further details of applications of our framework to practical situations, see Section 5.

3. Background

In this section, we review mathematical notions required for the remaining part of this paper. In Subsection 3.1, we review C^*-algebra, and in Subsection 3.2, we review Hilbert C^*-module. All the definitions and examples in this section are standard terminologies in C^*-algebra, and they are adopted from Hashimoto et al. (2021). The standard definitions related to the definitions in this section is provided in Section A. For further details of C^*-algebra and Hilbert C^*-module, see (Lance, 1995; Murphy, 1990; Hashimoto et al., 2021).

3.1. C^*-algebra

C^*-algebra is a generalization of the space of complex numbers.

Definition 3.1 (C^*-algebra)**.** A set $\mathcal{A}$ is called a *C^*-algebra* if it satisfies the following conditions:

1. $\mathcal{A}$ is an algebra (See Definition A.1) over $\mathbb{C}$ and equipped with a bijection $(\cdot)^* : \mathcal{A} \to \mathcal{A}$ that satisfies the following conditions for $\alpha, \beta \in \mathbb{C}$ and $c, d \in \mathcal{A}$:
   - $(\alpha c + \beta d)^* = \overline{\alpha} c^* + \overline{\beta} d^*$,
   - $(cd)^* = d^* c^*$,
   - $(c^*)^* = c$.
2. $\mathcal{A}$ is a normed space with $\|\cdot\|$, and for $c, d \in \mathcal{A}$, $\|cd\| \leq \|c\| \, \|d\|$ holds. In addition, $\mathcal{A}$ is complete with respect to $\|\cdot\|$.
3. For $c \in \mathcal{A}$, $\|c^* c\| = \|c\|^2$ holds.

We introduce important notions related to C^*-algebra.

Definition 3.2 (Multiplicative identity)**.** The *multiplicative identity* of a C^*-algebra $\mathcal{A}$ is the element $a \in \mathcal{A}$ that satisfies $ac = ca = c$ for any $c \in \mathcal{A}$. We denote by $1_{\mathcal{A}}$ the multiplicative identity of $\mathcal{A}$.

Definition 3.3 (Positive)**.** An element c of $\mathcal{A}$ is called *positive* if there exists $d \in \mathcal{A}$ such that $c = d^* d$ holds. We denote by $\mathcal{A}_+$ the subset of $\mathcal{A}$ composed of all positive elements in $\mathcal{A}$.

An important example of C^*-algebras is the space of continuous functions on a compact space, on which we focus in this paper.

Example 3.4. Let $\mathcal{Z}$ be a compact space and let $C(\mathcal{Z})$ be the Banach space of continuous functions equipped with the sup norm. Let $\cdot : C(\mathcal{Z}) \times C(\mathcal{Z}) \to C(\mathcal{Z})$ be defined as $(c \cdot d)(z) = c(z) d(z)$ for $c, d \in C(\mathcal{Z})$ and $z \in \mathcal{Z}$. In addition, let $(\cdot)^* : C(\mathcal{Z}) \to C(\mathcal{Z})$ be defined as $c^*(z) = \overline{c(z)}$ for $c \in C(\mathcal{Z})$. Then, $C(\mathcal{Z})$ is a C^*-algebra. The multiplicative identity is the constant function whose value is 1 at any $z \in \mathcal{Z}$. For $c \in C(\mathcal{Z})$, c is positive if and only if $c(z) \geq 0$ for any $z \in \mathcal{Z}$.

3.2. Hilbert C^*-module

Hilbert C^*-module is a generalization of Hilbert space. We first introduce C^*-module, which is a generalization of vector space. Then, we introduce $\mathcal{A}$-valued inner product and Hilbert C^*-module.

Definition 3.5 (C^*-module)**.** Let $\mathcal{M}$ be an abelian group with operation $+$ and let $\mathcal{A}$ be a C^*-algebra. If $\mathcal{M}$ is equipped with an $\mathcal{A}$-multiplication (See Definition A.2), $\mathcal{M}$ is called a C^*-*module* over $\mathcal{A}$.

Definition 3.6 ($\mathcal{A}$-valued inner product)**.** Let $\mathcal{A}$ be a C^*-algebra and let $\mathcal{M}$ be a C^*-module over $\mathcal{A}$. A $\mathbb{C}$-linear map with respect to the second variable $\langle \cdot, \cdot \rangle : \mathcal{M} \times \mathcal{M} \to \mathcal{A}$ is called an $\mathcal{A}$-valued *inner product* if it satisfies the following conditions for $u, v, p \in \mathcal{M}$ and $c, d \in \mathcal{A}$:

- $\langle u, vc + pd \rangle = \langle u, v \rangle c + \langle u, p \rangle d$, $\quad$ - $\langle v, u \rangle = \langle u, v \rangle^*$,
- $\langle u, u \rangle$ is positive, $\quad$ - If $\langle u, u \rangle = 0$ then $u = 0$.

Definition 3.7 (Norm)**.** Let $\mathcal{A}$ be a C^*-algebra and let $\mathcal{M}$ be a C^*-module over $\mathcal{A}$ equipped with an $\mathcal{A}$-valued inner product $\langle \cdot, \cdot \rangle$. The (real-valued) norm $\| \cdot \|$ on $\mathcal{M}$ is defined by $\|u\| = \left\| \langle u, u \rangle \right\|_{\mathcal{A}}^{1/2}$, where $\| \cdot \|_{\mathcal{A}}$ is the norm in $\mathcal{A}$.

Definition 3.8 (Hilbert C^*-module)**.** Let $\mathcal{A}$ be a C^*-module. Let $\mathcal{M}$ be a C^*-module over $\mathcal{A}$ equipped with an $\mathcal{A}$-valued inner product defined in Definition 3.6. If $\mathcal{M}$ is complete with respect to the norm $\| \cdot \|$ defined in Definition 3.7, it is called a *Hilbert C^*-module* over $\mathcal{A}$ or *Hilbert $\mathcal{A}$-module*.

4. Neural network on C^*-algebra

In this section, we propose a new framework with C^*-algebra that generalizes the existing framework of neural networks. In Subsection 4.1, we formulate the existing framework of neural networks. Then, in Subsection 4.2, we generalize the existing framework to that on C^*-algebra.

4.1. Existing framework of neural networks

We begin by formulating the existing framework of neural networks. We consider a network with $H \in \mathbb{N}$ hidden layers. Let $N_0, \ldots, N_{H+1}$ be natural numbers each of which represents the dimension of a layer. Inputs are vectors in $\mathbb{R}^{N_0}$ and outputs are vectors in $\mathbb{R}^{N_{H+1}}$. In addition, for $i = 1, \ldots, H+1$, let $\mathbf{W}_i : \mathbb{R}^{N_{i-1}} \to \mathbb{R}^{N_i}$ be an $N_{i-1} \times N_i$ matrix and $\boldsymbol{\sigma}_i : \mathbb{R}^{N_i} \to \mathbb{R}^{N_i}$ be an (often nonlinear) activation function. The neural network model $\mathbf{f} : \mathbb{R}^{N_0} \to \mathbb{R}^{N_{H+1}}$ is defined as

$$\mathbf{f} = \boldsymbol{\sigma}_{H+1} \circ \mathbf{W}_{H+1} \circ \boldsymbol{\sigma}_H \circ \mathbf{W}_H \circ \cdots \circ \boldsymbol{\sigma}_1 \circ \mathbf{W}_1. \quad (1)$$

We fix $\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_{H+1}$ and find best possible matrices $\mathbf{W}_1, \ldots, \mathbf{W}_{H+1}$ by minimizing a loss function $\mathbf{L}$. We regard the set of matrices $\mathbf{W}_1, \ldots, \mathbf{W}_{H+1}$ as an N-dimensional vector of parameters, which is denoted as $\boldsymbol{\theta}$, where $N = \sum_{i=1}^{H+1} N_{i-1} N_i$. Then, we set a loss function $\mathbf{L} : \mathbb{R}^N \to \mathbb{R}_+$, which depends on the parameters (and usually on inputs and outputs). Here, $\mathbb{R}_+$ is the set of all non-negative real numbers. We learn the parameter $\boldsymbol{\theta}$ by minimizing the loss function. The minimization of the loss function is implemented by a gradient descent method such as stochastic gradient descent (SGD) and Adam (Kingma & Ba, 2015). We compute the gradient $\nabla_{\boldsymbol{\theta}} \mathbf{L}$ of the loss function $\mathbf{L}$ and generate a sequence $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots$ using the gradient for finding a best possible $\boldsymbol{\theta}$.

4.2. Generalization to C^*-algebra

4.2.1. Formulation

To improve the representational power of the model, we generalize the parameter θ on $\mathbb{R}^N$, which is a Hilbert space, to a Hilbert C^*-module. Let $\mathcal{A}$ be a commutative unital C^*-algebra. By the Gelfand–Naimark theorem, there exists a compact Hausdorff space $\mathcal{Z}$ such that $\mathcal{A}$ is isometrically $*$-isomorphic to the C^*-algebra $C(\mathcal{Z})$, the space of continuous functions on $\mathcal{Z}$ (see Example 3.4). Therefore, we focus on the case of $\mathcal{A} = C(\mathcal{Z})$ for a compact Hausdorff space $\mathcal{Z}$ in the remaining parts of this paper. As before, let $N_0, \ldots, N_{H+1}$ be natural numbers each of which represents the dimension of a layer. However, in this case, we consider $\mathcal{A}^{N_0}$-valued inputs and $\mathcal{A}^{N_{H+1}}$-valued outputs. Since $\mathcal{A}$ is a function space, the inputs and outputs should be functions, which enables us to analyze functional data. On the other hand, for scalar-valued data $\mathbf{x} \in \mathbb{R}$, we can transform $\mathbf{x}$ into an appropriate function such as the constant function $x \equiv \mathbf{x}$ (see Subsections 5.1 and 5.2 for practical applications). In addition, for $i = 1, \ldots, H+1$, let $W_i : \mathcal{A}^{N_{i-1}} \to \mathcal{A}^{N_i}$ be an $N_{i-1} \times N_i$ $\mathcal{A}$-valued matrix and $\sigma_i : \mathcal{A}^{N_i} \to \mathcal{A}^{N_i}$ be an (often nonlinear) activation function. The neural network model $f : \mathcal{A}^{N_0} \to \mathcal{A}^{N_0}$ is defined as

$$f = \sigma_{H+1} \circ W_{H+1} \circ \sigma_H \circ W_H \circ \cdots \circ \sigma_1 \circ W_1 \quad (2)$$

in the same manner as Eq. (1). We regard the set of $\mathcal{A}$-valued matrices $W_1, \ldots, W_{H+1}$ as an N-dimensional $\mathcal{A}$-valued vector of parameters, which is denoted as θ, where $N = \sum_{i=1}^{H+1} N_{i-1} N_i$. Then, we set an $\mathcal{A}$-valued loss function $L : \mathcal{A}^N \to \mathcal{A}_+$, which depends on the $\mathcal{A}$-valued parameters. Here, $\mathcal{A}_+$ is the set of all positive elements in $\mathcal{A}$ (see Definition 3.3).

4.2.2. Learning C^*-algebra-valued parameters

To implement a gradient descent method on $\mathcal{A}^N$ and minimize the $\mathcal{A}$-valued loss function, we propose a practical approach to applying the gradient descent method proposed by Hashimoto et al. (2021) to our case. In more detail, we add the regularization term and simultaneously learn multiple models with interactions. We first define a gradient of $\mathcal{A}$-valued functions on $\mathcal{A}^N$.

Definition 4.1 ($\mathcal{A}$-valued gradient)**.** Let $L : \mathcal{A}^N \to \mathcal{A}$ be an

$\mathcal{A}$-valued function defined on $\mathcal{A}^N$ and let $\theta \in \mathcal{A}^N$. Assume there exists $\xi \in \mathcal{A}^N$ such that for any $\delta \in \mathcal{A}^N$ and any $z \in \mathcal{Z}$,

$$\lim_{\delta \to 0,\ \delta \in \mathcal{A}^N \setminus \{0\}} \frac{L(\theta+\delta)(z) - L(\theta)(z) - \langle \xi, \delta \rangle (z)}{\|\delta\|} = 0.$$

In this case, we define ξ as the *$\mathcal{A}$-valued gradient* of L at θ and denote it by $\nabla_{\mathcal{A},\theta} L$.

Example 4.2. Assume there exists a function $\tilde{L} : \mathbb{R}^N \times \mathcal{Z} \to \mathbb{R}$ such that $L(\theta)(z) = \tilde{L}(\theta(z), z)$, that is, we can decompose L into $\mathbb{R}$-valued functions indexed by z on $\mathbb{R}^N$. The function $\tilde{L}(\cdot, z)$ corresponds to an $\mathbb{R}$-valued (standard) loss function at $z \in \mathcal{Z}$. Assume $\tilde{L}(\cdot, z)$ has the (standard) gradient $\nabla_{\theta(z)} \tilde{L}(\cdot, z) \in \mathbb{R}^N$ for each $z \in \mathcal{Z}$. If the map $z \mapsto \nabla_{\theta(z)} \tilde{L}(\cdot, z)$ is contained in $\mathcal{A}^N$, then it is the $\mathcal{A}$-valued gradient.

We generate a sequence $\theta_0, \theta_1 \dots$ using the $\mathcal{A}$-valued gradient for finding a best possible θ to minimize the $\mathcal{A}$-valued loss function L. The basic gradient descent scheme is

$$\begin{aligned} &\theta_0 \in \mathcal{A}^N, \\ &\theta_{t+1} = \theta_t - \nabla_{\mathcal{A},\theta_t} L \cdot \eta_t \ (t = 0, 1, \dots), \end{aligned} \tag{3}$$

where $\eta_t \in \mathcal{A}_+$ is the learning rate.

Example 4.3. Assume there exists a function $\tilde{L} : \mathbb{R}^N \times \mathcal{Z} \to \mathbb{R}$ such that $L(\theta)(z) = \tilde{L}(\theta(z), z)$. Then, for each $z \in \mathcal{Z}$, the scheme (3) is reduced to

$$\begin{aligned} &\theta_0(z) \in \mathbb{R}^N, \\ &\theta_{t+1}(z) = \theta_t(z) - \eta_t(z) \nabla_{\theta_t(z)} \tilde{L}(\cdot, z) \ (t = 0, 1, \dots), \end{aligned}$$

which is the standard gradient descent scheme on $\mathbb{R}^N$ with the learning rate $\eta_t(z)$. Thus, computing the scheme (3) is equivalent to computing the standard gradient descent scheme on $\mathbb{R}^N$ simultaneously for all $z \in \mathcal{Z}$ in this case. If the standard gradient descent at each $z \in \mathcal{Z}$ generates a sequence $\theta_0(z), \theta_1(z), \dots$ in $\mathbb{R}^N$ converging to some $\theta^*(z)$, then we obtain a sequence $\theta_0, \theta_1, \dots$ in $\mathcal{A}^N$ converging pointwise to the function θ^*.

The above example implies that if $L(\theta)$ is defined as $L(\theta)(z) = \tilde{L}(\theta(z), z)$ for a function $\tilde{L} : \mathbb{R}^N \times \mathcal{Z} \to \mathbb{R}$, then the scheme (3) is computed without any interactions among variables $z \in \mathcal{Z}$. To learn the model with interactions among z, we assume $\mathcal{Z}$ is a compact finite measure space, and add an L_1 regularization term to the loss function L as

$$L\mathrm{reg}(\theta) = L(\theta) + \int_{z \in \mathcal{Z}} L(\theta)(z) dz 1_{\mathcal{A}} \cdot \lambda,$$

where $1_{\mathcal{A}}$ is the multiplicative identity of $\mathcal{A}$ ($1_{\mathcal{A}} \equiv 1$, see Definition 3.2) and $\lambda \in \mathcal{A}_+$ is a hyperparameter. Since the regularization term is a constant function with respect to z, it affects uniformly on $L_{\mathrm{reg}}(\theta)$ at any $z \in \mathcal{Z}$. It has an effect of aggregating the gradient at each $z \in \mathcal{Z}$ and adding the aggregated gradient to the gradient at each $z \in \mathcal{Z}$. Indeed, since $\nabla_{\mathcal{A},\theta} L \in \mathcal{A}^N$, each element of $\nabla_{\mathcal{A},\theta} L$ is integrable. Thus, we have

$$\nabla_{\mathcal{A},\theta} L_{\mathrm{reg}}(\theta) = \nabla_{\mathcal{A},\theta} L + \int_{z \in \mathcal{Z}} (\nabla_{\mathcal{A},\theta} L)(z) dz \odot \mathbf{1}_{\mathcal{A}} \lambda,$$

where $\mathbf{1}_{\mathcal{A}} = [1_{\mathcal{A}}, \dots, 1_{\mathcal{A}}]^T \in \mathcal{A}^N$ and $\odot$ represents the element-wise product of $\int_{z \in \mathcal{Z}} (\nabla_{\mathcal{A},\theta}) L(z) dz \in \mathbb{C}^N$ and $\mathbf{1}_{\mathcal{A}} \in \mathcal{A}^N$.

In practical computations, we cannot handle functions in the infinite-dimensional space $\mathcal{A}$. Therefore, we set a finite-dimensional subspace $\mathcal{V}$ of $\mathcal{A}^N$ and a map $P : \mathcal{A}^N \to \mathcal{V}$. Moreover, we set the learning rate η_t and the hyperparameter λ as constant functions $\eta_t \equiv \tilde{\eta}_t$ and $\lambda \equiv \tilde{\lambda}$ for some $\tilde{\eta}_t, \tilde{\lambda} \in \mathbb{R}+$. Then, the practical scheme is

$$\begin{aligned} &\theta_0 \in \mathcal{V}, \\ &\theta_{t+1} = \theta_t - \tilde{\eta}_t P(\nabla_{\mathcal{A},\theta_t} L) \\ &\quad - \tilde{\lambda} \int_{z \in \mathcal{Z}} P(\nabla_{\mathcal{A},\theta} L)(z) dz \odot \mathbf{1}_{\mathcal{A}} \ (t = 0, 1, \dots). \end{aligned} \tag{4}$$

Typically, we set $P(\theta)$ as a regression of $\theta(z_1), \dots, \theta(z_n)$ in $\mathcal{V}$, where $z_1, \dots, z_n$ for some $n \in \mathbb{N}$ are properly chosen fixed points. As a result, the sequence $\theta_0, \theta_1, \dots$ is contained in $\mathcal{V}$, which enables us to compute the scheme (4) practically. In addition, by applying P, the scheme becomes interactive with respect to $z \in \mathcal{Z}$. That is, $P(\theta)(z_0)$ is determined by using the values $\theta(z)$ for $z \neq z_0$. Regarding the choice of $\mathcal{V}$, we choose it on the basis of the following fact.

Fact 4.4. *Let $\mathcal{Z}$ be a compact metric space and let $d_{\mathcal{Z}}$ be the metric on $\mathcal{Z}$. Let $\zeta_0, \zeta_1, \dots \in \mathcal{A}$ be a sequence such that there exists a function ζ^* on $\mathcal{Z}$, $\lim_{t \to \infty} \zeta_t(z) = \zeta^*(z)$ for any $z \in \mathcal{Z}$. If $\zeta_0, \zeta_1, \dots$ is uniformly Lipschitz continuous, that is, there exists $C > 0$ such that for any $t \in \mathbb{N}$,*

$$|\zeta_t(z_1) - \zeta_t(z_2)| \leq C d_{\mathcal{Z}}(z_1, z_2),$$

then $\zeta_0, \zeta_1, \dots$ converges uniformly to ζ^* and $\zeta^* \in \mathcal{A}$.

On the basis of Fact 4.4, we set $\mathcal{V} = \mathrm{Span}\{v_1, \dots, v_l\}^N$ for $l \in \mathbb{N}$ and Lipschitz continuous functions $v_1, \dots, v_l \in \mathcal{A}$. Then, each element $\theta_{t,j} \in \mathcal{A}$ of θ_t is represented as $\sum_{i=1}^n c_{t,i} v_i$ for some $c_{t,i} \in \mathbb{C}$. If there exists $C_i > 0$ such that for any $t \in \mathbb{N}$ $|c_{t,i}| \leq C_i$ for $i = 1, \dots, n$, then the sequence $\theta_{0,j}, \theta_{1,j}, \dots$ is uniformly Lipschitz. Thus, if it converges pointwise, the convergence is uniform. Uniform convergence is a stronger notion than pointwise convergence and preserves properties of the sequence of functions to its limit. For example, the uniform limit of a sequence of uniformly continuous functions is also uniformly continuous, but the pointwise limit of a sequence of continuous functions

is not always continuous. Thus, the uniform convergence is more suitable when we need the parameter θ to preserve properties as a function throughout the gradient descent.

Remark 4.5. The scheme 4 is a basic scheme of the $\mathcal{A}$-valued gradient descent. We can also consider variants of the scheme such as SGD and Adam in the same manner as the standard gradient descent on $\mathbb{R}^N$.

Computation complexity The computation complexity of learning the C^*-algebra net is $O(pl)$, where p is the number of parameters of the network and l is the number of basis functions in $\mathcal{V}$. It is l times as large as that of the corresponding $\mathbb{R}$-valued model.

We illustrate the difference between $\mathbb{R}$-valued and $\mathcal{A}$-valued models for a specific case of the linear regression problem on $\mathbb{R}$ in Section C.

5. Applications

In this section, we show examples of practical applications of our framework. In Subsection 5.1, we apply our framework to density estimation with normalizing flow. In Subsection 5.2, we apply our framework to few-shot learning. By using a model with function-valued parameters, we can learn features of data efficiently even with a limited number of samples. Our framework is general and its application is not restricted to the above two cases. In Subsection 5.3, we show other examples of applications.

5.1. Density estimation

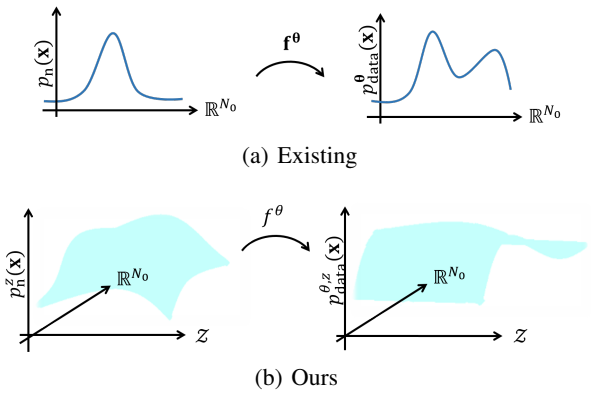
5.1.1. Density estimation with normalizing flow

We can improve density estimation by replacing parameters in a model by $\mathcal{A}$-valued ones. In this paper, we focus on using normalizing flow (Dinh et al., 2014; 2017; Teshima et al., 2020), but the application is not limited to normalizing flow. We first review density estimation with normalization flow briefly. Let Ω be a probability space. We construct $\mathbf{f} = \mathbf{f}^{\boldsymbol{\theta}}$ in Eq. (1) as a measurable, invertible, and differentiable map that transforms a normal distribution into the distribution of data. That is, if the distribution of a random variable X defined on Ω and taking its value in $\mathbb{R}^{N_0}$ is the normal distribution, then the distribution of $\mathbf{f}^{\boldsymbol{\theta}}(X)$ is the distribution of data. The density $p^{\boldsymbol{\theta}}_{\mathrm{data}}$ of $\mathbf{f}^{\boldsymbol{\theta}}(X)$ is calculated as

$$p^{\boldsymbol{\theta}}_{\mathrm{data}}(\mathbf{x}) = p_{\mathrm{n}}\big((\mathbf{f}^{\boldsymbol{\theta}})^{-1}(\mathbf{x})\big)\big|\ \det\big(\nabla_{\mathbf{x}}(\mathbf{f}^{\boldsymbol{\theta}})^{-1}\big)\big| \quad (5)$$

for $x \in \mathbb{R}^{N_0}$, where p_{n} is the density of the normal distribution. By using the formula (5), we compute the likelihood of given samples $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^{N_0}$ and set the loss function $\mathbf{L} : \mathbb{R}^N \to \mathbb{R}_+$ as a negative log likelihood:

$$\mathbf{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \log p^{\boldsymbol{\theta}}_{\mathrm{data}}(\mathbf{x}_i).$$



(a) Existing

(b) Ours

Figure 2. Difference between the existing method and our method of density estimation with normalizing flow. (Although the set $\mathcal{Z}$ in (b) is depicted as if it is a one-dimensional space, we can set $\mathcal{Z}$ as an arbitrary compact space.)

5.1.2. Generalization to C^*-algebra

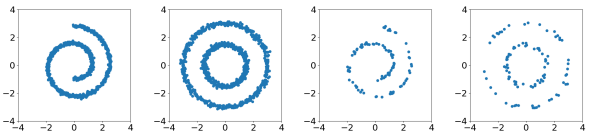
We generalize the above setting to that with C^*-algebra. Let $\mathcal{Z}$ be a compact probability space and let X be a random variable defined on $\mathcal{Z} \times \Omega$ and taking its value in $\mathbb{R}^{N_0}$. Assume that for any $\omega \in \Omega$, $X(\cdot, \omega)$ is continuous on $\mathcal{Z}$. Let $\mathcal{D}$ be the probability measure on $\mathcal{Z}$. The map $f = f^{\theta}$ in Eq. (2) is constructed so that if for any $z \in \mathcal{Z}$, the distribution of a random variable $X(z, \cdot)$ is a normal distribution on $\mathbb{R}^{N_0}$, then the distribution of the random variable $\omega \mapsto f^{\theta}(X(\cdot, \omega))(z)$ is the distribution of data. The density $p^{\theta,z}_{\mathrm{data}}$ of the random variable $\omega \mapsto f^{\theta}(X(\cdot, \omega))(z)$ is calculated as

$$p^{\theta,z}_{\mathrm{data}}(\mathbf{x}) = p^{z}_{\mathrm{n}}\big((f^{\theta})^{-1}(x)(z)\big)\ \big|\det\big(\nabla_{\mathcal{A},x}(f^{\theta})^{-1}(z)\big)\big| \quad (6)$$

for $\mathbf{x} \in \mathbb{R}^{N_0}$, where $x \in \mathcal{A}^{N_0}$ is the constant function $x \equiv \mathbf{x}$. By using the formula (6), we compute the likelihood of given samples $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^{N_0}$ and set the loss function $L : \mathcal{A}^N \to \mathcal{A}_+$ as a negative log likelihood:

$$L(\theta)(z) = -\sum_{i=1}^{n} \log p^{\theta,z}_{\mathrm{data}}(\mathbf{x}_i). \quad (7)$$

We apply the $\mathcal{A}$-valued gradient descent method proposed in Subsection 4.2.2 to minimize the loss function (7). Since $L(\theta)(z)$ depends only on $\theta(z)$, we have $L(\theta)(z) = \tilde{L}(\theta(z), z)$ for some function $\tilde{L} : \mathbb{R}^N \times \mathcal{Z} \to \mathbb{R}$. As we explained in Example 4.3, the $\mathcal{A}$-valued gradient of L is calculated by the standard gradient of $\tilde{L}(\cdot, z)$. After learning the model, we obtain the distribution $p^{\theta,z}_{\mathrm{data}}$ at each $z \in \mathcal{Z}$. Ideally, $p^{\theta,z}_{\mathrm{data}}$ is independent of z since the distribution of data is independent of z. Practically, we get an estimation of the density $\tilde{p}^{\theta}_{\mathrm{data}}$ of data by integrating $p^{\theta,z}_{\mathrm{data}}$ as $\tilde{p}^{\theta}_{\mathrm{data}}(\mathbf{x}) = \int_{z \in \mathcal{Z}} p^{\theta,z}_{\mathrm{data}}(\mathbf{x}) d\mathcal{D}(z)$.



(a) Swiss roll (b) Circles (c) Swiss roll (training) (d) Circles (training)

Figure 3. Samples in the datasets ((c) and (d) show training samples)

5.1.3. Numerical results

We show the validity of applying our method to density estimation with normalizing flow numerically. We generated two toy datasets: swiss roll and circles. Each dataset contains 1000 samples, illustrated in Fig. 3. We put 100 samples for training samples from the datasets. The estimation of the densities is challenging since the amount of training samples is small. We used masked autoregressive flow (Papamakarios et al., 2017) to construct the invertible differentiable map f^θ, and for the gradient descent method, we used Adam. We set the learning rate so that it decays polynomially starting from 0.001 with decay rate 0.5. We set $\mathcal{Z} = [-4, 4] \times [-4, 4] \subseteq \mathbb{R}^2$ and set the finite-dimensional subspace $\mathcal{V}$ of $\mathcal{A}^N$ as $\mathrm{Span}\{v_1, \dots, v_l\}^N$, where $l = 9$, $v_i(z) = e^{-10\|z_i - z\|^2}$, $z_1 = [0, 0]$, and $z_{4i+j+1} = [(2 + i)\sin(2\pi(j - 1 + 0.5i)/4), (2 + i)\cos(2\pi(j - 1 + 0.5i)/4)]$ for $i = 0, 1$ and $j = 1, \dots, 4$. The density $p_{\mathrm{n}}(z, \cdot)$ of the base normal distribution at $z \in \mathcal{Z}$ is set as the normal distribution with mean z and standard deviation 1. We set the map $P : \mathcal{A} \to \mathcal{V}$ as the kernel ridge regression, where the ith element of $P(\theta)$ for $\theta = [\theta_1, \dots, \theta_N]^T \in \mathcal{A}^N$ is computed as $[v_1, \dots, v_n](G + \mu I)^{-1}[\theta_i(z_1), \dots, \theta_i(z_l)]^T$. Here, G is the Gram matrix whose (i, j)-element is $e^{-10\|z_i - z_j\|^2}$ and $\mu \in \mathbb{R}_+$ is a hyperparameter of the regression that controls the strength of a regularization term of the regression. In this experiment, we set $\mu = 0.1$. In addition, we set the hyperparameter $\tilde{\lambda}$ in Eq. (4) as 0.3 and set the distribution $\mathcal{D}$ on $\mathcal{Z}$ as the uniform distribution on $\bigcup_{i=1}^9 \{z \in \mathcal{Z} \mid \|z - z_i\| \leq 0.05\}$. We compared our method proposed in Subsection 5.1.2 with two straightforward methods, (1) the standard method explained in Subsection 5.1.1, called "standard", and (2) learning the model whose base normal distribution is $p_{\mathrm{n}}(z_i, \cdot)$ for $i = 1, \dots 9$ separately and computing the mean value of the results, called "discrete". Note that the method (1) corresponds to the case of $l = 1$ and $\tilde{\lambda} = \mu = 0$ and the method (2) corresponds to the case of $l = 9$ and $\tilde{\lambda} = \mu = 0$ of the proposed method.

Fig. 4 shows the estimated densities and negative log-likelihoods computed with the 900 samples other than the training samples in each dataset. We see that our method

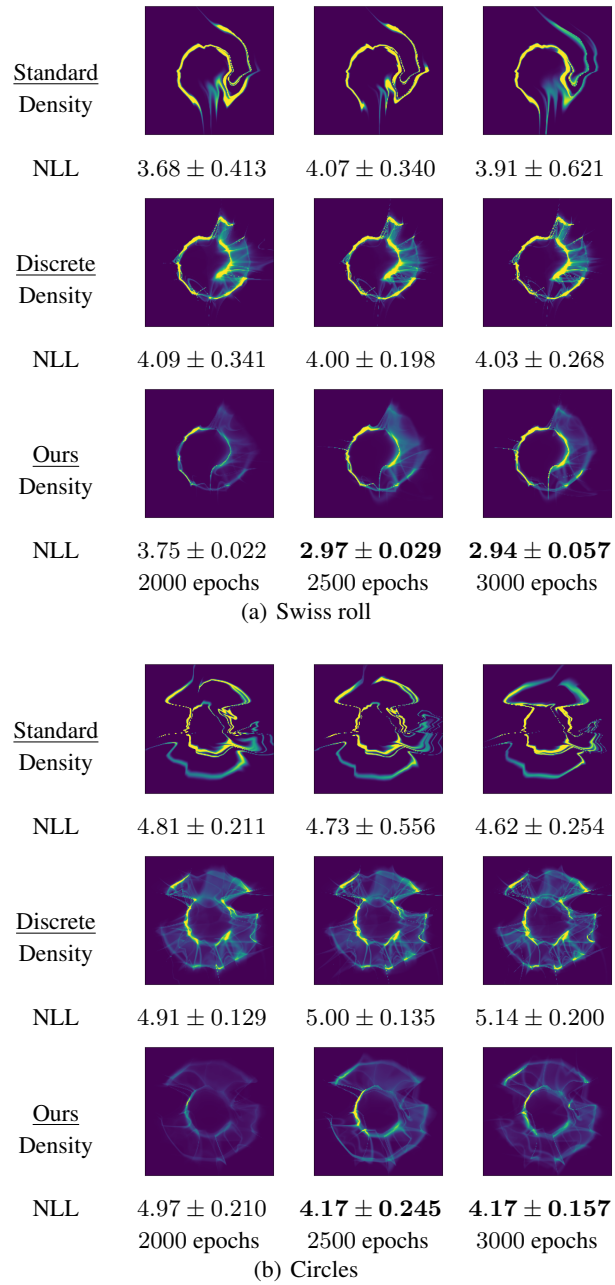


Figure 4. Comparison among estimated densities and negative log-likelihoods (NLLs) with existing methods and our method. (Each value of negative log likelihood is an average (± a standard deviation) over 5 independent runs.)

with 2500 or 3000 epochs gives the best estimation of the densities, and its negative log-likelihoods are the smallest for each dataset. Since the amount of samples is small, the standard method fails to capture the detailed shape of the density and its shape is blurred. By learning multiple models separately and computing the mean value, the shape becomes more clear, but not smooth. Our method captures

the shape clearly and smoothly because it allows us to learn multiple models simultaneously with interactions by using smooth functions.

Computation complexity and memory usage The method "discrete" learns 9 real-valued models separately, and in the above experiment, we compared our proposed method with $l = 9$ to the method "discrete". The computation complexities of these two methods are the same. Precisely, in our method, we compute the approximation of elements in $\mathcal{A}$ when updating the parameters. (The operator P in Eq. (4)) However, the computation complexity of this part can be ignored since we used the kernel ridge regression and $l \ll p$. The approximation of a function $a \in \mathcal{A}$ is computed by just the multiplication of an $l \times l$ fixed matrix to the vector $[a(z_1), \ldots, a(z_l)]$. Our method outperforms the method "Discrete" since our method properly combines multiple models continuously. Regarding the memory usage, the amount of memory only increases linearly with respect to l.

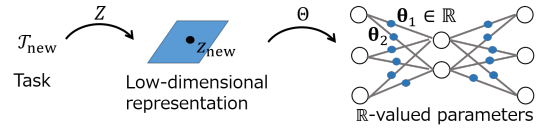
5.2. Few-shot learning

5.2.1. Few-shot learning for classification

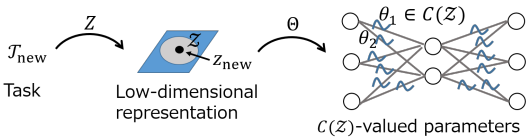
We can improve the accuracy of few-shot learning by replacing parameters in a model by $\mathcal{A}$-valued ones. In this paper, we focus on classification tasks, but the application of our framework is not limited to classification tasks. Few-shot learning challenges a model to be learned with a limited number of samples (Fei-Fei et al., 2006; Lake et al., 2011). We first formulate the setting for a supervised classification task briefly. We construct $\mathbf{f} = \mathbf{f}^{\boldsymbol{\theta}}$ in Eq. (1) as a map which maps a sample such as an image to its label, that is, the probability to belong to each class. For given samples $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^{N_0}$ and their labels $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^{N_{H+1}}$, we set the loss function $\mathbf{L} : \mathbb{R}^N \to \mathbb{R}_+$ as the cross categorical entropy:

$$\mathbf{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \left\langle \mathbf{y}_i, \log(\mathbf{f}^{\boldsymbol{\theta}}(\mathbf{x}_i)) \right\rangle .$$

For few-shot learning, one approach to learning the model effectively is meta-learning (Ravi & Larochelle, 2017; Finn et al., 2017; Rusu et al., 2019). Before learning a model for a new task $\mathcal{T}_{\text{new}}$, we learn a meta-model with multiple tasks $\mathcal{T}_1, \ldots, \mathcal{T}_m$ to extract common features among all tasks. For example, Rusu et al. (2019) propose to learn a model to obtain a map Z that maps a task $\mathcal{T}_i$ to its low-dimensional representation z_i and a map Θ that maps z_i into the parameter of a model specific to the task $\mathcal{T}_i$. Then, for a given new task $\mathcal{T}_{\text{new}}$, we can get a corresponding low-dimensional representation $z_{\text{new}} = Z(\mathcal{T}_{\text{new}})$ and a parameter $\Theta(z_{\text{new}})$. By using $\Theta(z_{\text{new}})$ as the initial value of the parameter θ, we can learn the model for the new task $\mathcal{T}_{\text{new}}$ efficiently even with a limited number of samples for the new task.



(a) Existing



(b) Ours

Figure 5. Difference between the existing method and our method for the transformation of a task to parameters of the model.

5.2.2. Generalization to C^*-algebra

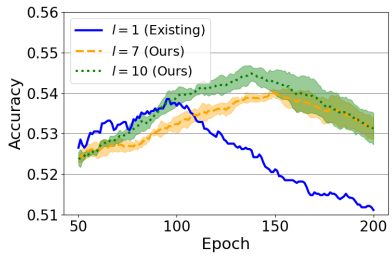
We generalize the above setting to that with C^*-algebra. We construct $f = f^{\theta}$ in Eq. (2) as a map that maps the constant function $x \equiv \mathbf{x}$ for a sample $\mathbf{x}$ to the constant function $y \equiv \mathbf{y}$ for its label $\mathbf{y}$. As described in Section 5.2.1, for a given new task $\mathcal{T}_{\text{new}}$, we can get a corresponding low-dimensional representation $z_{\text{new}} = Z(\mathcal{T}_{\text{new}})$. However, in this case, we use also vectors in the neighborhood of $z_{\text{new},1}$. We set $\mathcal{Z}$ as a neighborhood of z_{new} and set the $\mathcal{A}$-valued parameter θ as a function on $\mathcal{Z}$. For given samples $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^{N_0}$ and its labels $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^{N_{H+1}}$, we set the loss function $L : \mathbb{R}^N \to \mathbb{R}_+$ as the cross categorical entropy:

$$L(\theta) = -\sum_{i=1}^{n} \left\langle y_i, \log(f^{\theta}(x_i)) \right\rangle .$$

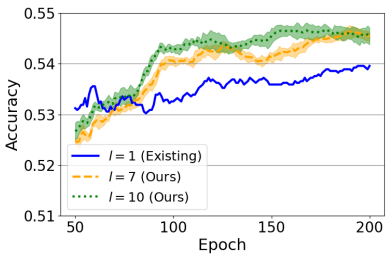
By using the restriction of the function Θ on $\mathcal{Z}$ as the initial value of the $\mathcal{A}$-parameter θ, we can learn the model for the new task $\mathcal{T}_{\text{new}}$ using the information about tasks distributed in the neighborhood of the new task in the space of the low-dimensional representations. This makes learning the model more efficient even with a limited number of samples for the new task.

5.2.3. Numerical results

We show the validity of applying our method to few-shot learning numerically. We used the miniImageNet dataset (Vinyals et al., 2016), which is composed of 100 classes, each of which has 600 images, and considered the 5-way 1-shot task in the same manner as Subsection 4.2 in (Rusu et al., 2019). We randomly split the dataset into the train data with 1 image and the test data with 599 images. In this case, each task is the classification with respect to randomly selected 5 classes with 1 sample in

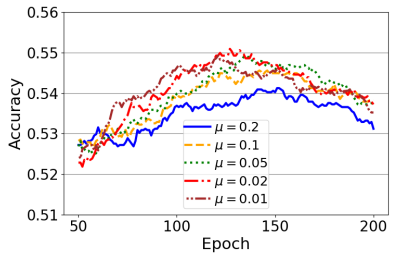


(a) Task 1

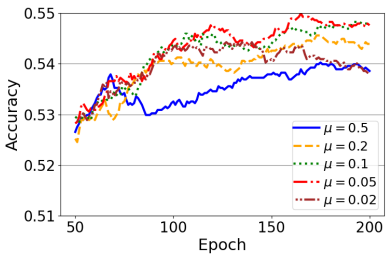


(b) Task 2

Figure 6. Test accuracies of the new tasks with different values of l. (Each value is an average ($\pm$ a standard deviation) over 5 independent runs with different values of $z_{j,i}$ ($j = 1, \dots, 5$, $i = 2, \dots, l$). $\mu = 0.05$ for Task 1 and $\mu = 0.1$ for Task 2.)



(a) Task 1



(b) Task 2

Figure 7. Test accuracies of the new tasks with different values of μ. ($l = 10$ for both tasks)

each class. We set f^θ as the one-layer softmax classifier as used by Rusu et al. (2019), and for the gradient descent method, we used Adam with learning rate 0.001. In this case, the number of parameters N is $N_1 N_0$, where $N_0 = 640$ and $N_1 = 5$. We first learn the model proposed by Rusu et al. (2019) and obtain maps Z and Θ explained in Subsection 5.2.1. In their model, the dimension of the low-dimensional representation is 320 and for $j = 1, \dots, N_1$, elements $N_0(j-1)+1 \sim N_0 j$ of Θ only depends on elements $64(j-1)+1 \sim 64j$ in the space of the low-dimensional representation. Thus, we set $\mathcal{Z} \subseteq \mathbb{R}^{320}$. For a new task $\mathcal{T}_{\text{new}}$, we set the finite-dimensional subspace $\mathcal{V}$ of $\mathcal{A}^N$ as $\oplus_{j=1}^{N_1} \mathrm{Span}\{v_{j,1}, \dots, v_{j,l}\}^{N_0}$. Here, $l \leq 10$, $v_{j,i}(z) = e^{-10\|z_{j,i} - p_j(z)\|^2}$, $z_{j,1} = Z(\mathcal{T}_{\text{new}})_{64(j-1)+1:64j}$, and $z_{j,i}$ for $i = 2, \dots, l$ are randomly drawn from the normal distribution with mean $z_{j,1}$ and standard deviation 0.01. Moreover, p_j is the projection that maps z to $z_{64(j-1)+1:64j}$ and for a finite-dimensional vector v, $v_{i:j}$ denotes the $j-i+1$-dimensional vector composed of elements $i \sim j$ of v. We set the map $P : \mathcal{A} \to \mathcal{V}$ as the kernel ridge regression in the same manner as Section 5.1.3.

For randomly selected two new tasks, we compared the accuracy of the classification among different values of l. We set $l = 1, 7, 10$ for both tasks. For $l > 1$, the output (the probability to belong to each class) is a function on $\mathcal{Z}$. We computed the test accuracy of the output at z_1. Fig. 6 shows the results. Note that $l = 1$ corresponds to the standard few-shot learning (with $\mathbb{R}$-valued parameters) explained in 5.2.1.

We can get higher accuracy as l becomes larger. Fig. 7 shows the accuracy with different values of the hyper parameter η of the kernel ridge regression P. In this experiment, we fixed another hyperparameter λ as 0. If $\mu = 0$, then our method is equivalent to the existing method. Thus, if μ is too small, the accuracy is not so high. On the other hand, if μ is large, then the mean squared error of the kernel ridge regression becomes large. Thus, if μ is too large, the accuracy is not so high.

5.3. Other applications

Although in Subsections 5.1 and 5.2 we focus on density estimation and few-shot learning, we can apply our framework to other applications. We list examples of other applications and discuss connections with existing methods below.

Ensemble learning Ensemble learning combines multiple models to obtain better generalization performance (Dong et al., 2020; Ganaie et al., 2021). Our framework allows us to combine the models continuously. Indeed, the case of $\tilde{\lambda} = \eta = 0$ and $\mathcal{D} = \sum_{i=1}^{l} \delta_{z_i}/l$ in Subsection 5.1.3 is equivalent to the existing framework of ensemble learning. In general, let $\mathcal{Z} = \{z_1, \dots, z_m\}$ be a finite discrete set and let P and $\tilde{\lambda}$ in Eq. (4) be the identity map and 0, respectively. In addition, let the loss function L be defined as $L(\theta)(z) = \tilde{L}(\theta(z), z)$. Then, our framework is reduced to the ensemble learning because for each $i = 1, \dots, m$, $f^\theta(\cdot)(z_i)$ in Eq. (2) is the classical model with $\mathbb{R}$-valued parameters, and the learning process is independent of that for $j \neq i$. By setting $\mathcal{Z}$ as an infinite set and

P as a map different from the identity map, we can learn multiple models more efficiently.

Generating time-series or spatial data We set $\mathcal{Z} \subseteq \mathbb{R}$ for time-series data and we set $\mathcal{Z} \subseteq \mathbb{R}^2$ or $\mathcal{Z} \subseteq \mathbb{R}^3$ for spatial data. Then, we set $f = f^\theta$ in Eq. (2) so that $f^\theta(\cdot)(z)$ is a generative model such as GAN (Goodfellow et al., 2014; Karras et al., 2020), VAE (Kingma & Welling, 2014; Gregor et al., 2015), or normalizing flow (Dinh et al., 2014; 2017; Teshima et al., 2020) for any $z \in \mathcal{Z}$. Since the outputs of the model are functions on $\mathcal{Z}$ in our framework, we can generate time-series or spatial data as a function on $\mathcal{Z}$ rather than a discrete series.

Learning distributions of parameters Distributions of parameters of models have been studied (Pennington et al., 2018; Sonoda et al., 2021). Franchi et al. (2020) propose learning distributions of parameters of a model rather than their values. They assume that the distributions are normal distributions. Using our framework, we can consider more general distributions. Let $\mathcal{Z}$ be a probability space and we set the $\mathcal{A}$-valued parameters $\theta \in \mathcal{A}^N$ as random variables taking their values in $\mathbb{R}$. If we limit the distributions of θ to Dirac measures, then our framework is reduced to be a classical model with $\mathbb{R}$-valued parameters. If we limit the distributions of θ to normal distributions, then our framework is reduced to be the framework proposed by Franchi et al. (2020).

Generalizing complex-valued networks Using complex-valued variables and parameters of models for taking advantage of the arithmetic of complex numbers has been studied (Bassey et al., 2021; Hirose, 1992; Amin et al., 2008; Nishikawa et al., 2005; Yadav et al., 2005). Since C^*-algebra is a generalization of the space of complex numbers, our framework generalizes complex-valued networks. In addition, the C^*-algebra $C(\mathcal{Z})$ is the space of complex-valued continuous functions on $\mathcal{Z}$. Thus, by using our framework, we can aggregate multiple complex-valued networks in the same way as real-valued networks.

6. Conclusion

In this paper, we proposed a new general framework of neural networks on C^*-algebra. We focused on the C^*-algebra of the space of continuous functions on a compact space and provided a gradient descent method for learning the model on C^*-algebra. By generalizing the parameters of a model to functions, we can use tools for functions such as regression and integrations, which enables us to learn features of data efficiently and adapt the models to problems continuously. We applied our framework to density estimation and few-shot learning and showed the validity of our framework. Our framework is valid for a wide range of practical applications and not limited to the above cases.

References

Amin, M. F., Islam, M. M., and Murase, K. Single-layered complex-valued neural networks and their ensembles for real-valued classification problems. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 2500–2506, 2008.

Bassey, J., Li, X., and Qian, L. A survey of complex-valued neural networks. arXiv:2101.12249, 2021.

Candès, E. J. Harmonic analysis of neural networks. *Applied and Computational Harmonic Analysis*, 6(2):197–218, 1999.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. In *Proceedings of the International Conference on Learning Representations Workshop*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Dong, X., Yu, Z., Cao, W., Shi, Y., and Ma, Q. A survey on ensemble learning. *Frontiers of Computer Science*, 14: 241–258, 2020.

Fei-Fei, L., Fergus, R., and Perona, P. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Franchi, G., Bursuc, A., Aldea, E., Dubuisson, S., and Bloch, I. TRADI: Tracking deep neural network weight distributions. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)*, 2020.

Ganaie, M. A., Hu, M., Tanveer, M., and Suganthan, P. N. Ensemble deep learning: A review. arXiv:2104.02395, 2021.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Proceedings of the*

Advances in Neural Information Processing Systems 27 (NIPS), 2014.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Hashimoto, Y., Ishikawa, I., Ikeda, M., Komura, F., Katsura, T., and Kawahara, Y. Reproducing kernel Hilbert C^*-module and kernel mean embeddings. *Journal of Machine Learning Research*, 22(267):1–56, 2021.

Hirose, A. Continuous complex-valued back-propagation learning. *Electronics Letters*, 28:1854–1855, 1992.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

Lance, E. C. *Hilbert C^*-modules – a Toolkit for Operator Algebraists*. London Mathematical Society Lecture Note Series, vol. 210. Cambridge University Press, 1995.

Murphy, G. J. *C*-Algebras and Hilbert Space Operators*. Academic Press, 1990.

Nishikawa, I., Sakakibara, K., Iritani, T., and Kuroe, Y. 2 types of complex-valued hopfield networks and the application to a traffic signal control. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, volume 2, pp. 782–787, 2005.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Proceedings of the Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.

Pennington, J., Schoenholz, S., and Ganguli, S. The emergence of spectral universality in deep networks. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Sonoda, S. and Murata, N. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017.

Sonoda, S., Ishikawa, I., and Ikeda, M. Ridge regression with over-parametrized two-layer networks converge to Ridgelet spectrum. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

Teshima, T., Ishikawa, I., Tojo, K., Oono, K., Ikeda, M., and Sugiyama, M. Coupling-based invertible neural networks are universal diffeomorphism approximators. In *Proceedings of the Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., and Wierstra, D. Matching networks for one shot learning. In *Proceedings of the Advances in Neural Information Processing Systems 29 (NIPS)*, 2016.

Yadav, A., Mishra, D., Ray, S., Yadav, R., and Kalra, P. Representation of complex-valued neural networks: a real-valued approach. In *Proceedings of 2005 International Conference on Intelligent Sensing and Information Processing*, pp. 331–335, 2005.

Appendix

A. Definitions related to Section 3

We provide the standard definitions related to Section 3.

Definition A.1 (Algebra)**.** A set $\mathcal{A}$ is called an *algebra* on a field $\mathbb{F}$ if it is a vector space equipped with an operation $\cdot : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ that satisfies the following conditions for $b, c, d \in \mathcal{A}$ and $\alpha \in \mathbb{F}$:

• $(b + c) \cdot d = b \cdot d + c \cdot d$, • $b \cdot (c + d) = b \cdot c + b \cdot d$, • $(\alpha c) \cdot d = \alpha(c \cdot d) = c \cdot (\alpha d)$.

The symbol $\cdot$ is omitted when doing so does not cause confusion.

Definition A.2 (Multiplication)**.** Let $\mathcal{M}$ be an abelian group with operation $+$ and let $\mathcal{A}$ be a ring. For $c, d \in \mathcal{A}$ and $u, v \in \mathcal{M}$, if an operation $\cdot : \mathcal{M} \times \mathcal{A} \to \mathcal{M}$ satisfies

• $(u + v) \cdot c = u \cdot c + v \cdot c$, • $u \cdot (c + d) = u \cdot c + u \cdot d$, • $u \cdot (cd) = (u \cdot c) \cdot d$, • $u \cdot 1_{\mathcal{A}} = u$ if $\mathcal{A}$ is unital,

then $\cdot$ is called an *$\mathcal{A}$-multiplication*. The symbol $\cdot$ is omitted when doing so does not cause confusion.

B. Proof of Fact 4.4

Fact 3.4. *Let $\mathcal{Z}$ be a compact metric space and let $d_{\mathcal{Z}}$ be the metric on $\mathcal{Z}$. Let $\zeta_0, \zeta_1, \ldots \in \mathcal{A}$ be a sequence such that there exists a function ζ^* on $\mathcal{Z}$, $\lim_{t\to\infty} \zeta_t(z) = \zeta^*(z)$ for any $z \in \mathcal{Z}$. If $\zeta_0, \zeta_1, \ldots$ is uniformly Lipschitz continuous, that is, there exists $C > 0$ such that for any $t \in \mathbb{N}$,*

$$|\zeta_t(z_1) - \zeta_t(z_2)| \leq C d_{\mathcal{Z}}(z_1, z_2),$$

then $\zeta_0, \zeta_1, \ldots$ converges uniformly to ζ^* and $\zeta^* \in \mathcal{A}$.

Proof. For $z, c \in \mathcal{Z}$ and $s, t \in \mathbb{N}$, we have

$$|\zeta_s(z) - \zeta_t(z)| \leq |\zeta_s(z) - \zeta_s(c)| + |\zeta_s(c) - \zeta_t(c)| + |\zeta_t(c) - \zeta_t(z)| \leq 2C d_{\mathcal{Z}}(z, c) + |\zeta_s(c) - \zeta_t(c)|, \tag{8}$$

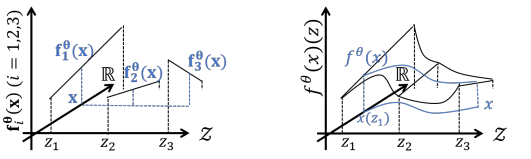
where the last inequality holds since $\zeta_0, \zeta_1, \ldots$ is uniformly Lipschitz continuous. Let $\epsilon > 0$. Since $\zeta_t(c)$ converges to $\zeta^*(c)$, there exists $T_c > 0$ such that for any $s, t \geq T_c$, $|\zeta_s(c) - \zeta_t(c)| \leq \epsilon$ holds. Let $\mathcal{Z}_c = \{z \in \mathcal{Z} \mid d_{\mathcal{Z}}(z, c) < \epsilon\}$. Then, we have $\mathcal{Z} = \bigcup_{c\in\mathcal{Z}} \mathcal{Z}_c$. Since $\mathcal{Z}$ is compact, there exists $n \in \mathbb{N}$ and $c_1, \ldots, c_n \in \mathcal{Z}$ such that $\mathcal{Z} = \bigcup_{i=1}^{n} \mathcal{Z}_{c_i}$. For $z \in \mathcal{Z}_{c_i}$, by Eq. (8), we have

$$|\zeta_s(z) - \zeta_t(z)| \leq 2C\epsilon + \epsilon$$

for $s, t \geq T_{c_i}$. Thus, for any $s, t \geq \max_{i\in\{1,\ldots,n\}} T_{c_i}$ and $z \in \mathcal{Z}$, we have $|\zeta_s(z) - \zeta_t(z)| \leq (2C + 1)\epsilon$, which implies the uniform convergence of ζ_t. Since $\mathcal{A} = C(\mathcal{Z})$ is the Banach space equipped with the sup norm, $\zeta^* \in \mathcal{A}$. □

C. Difference between $\mathbb{R}$-valued and C^*-algebra-valued models

We illustrate the difference between $\mathbb{R}$-valued and C^*-algebra-valued models for a specific case of the linear regression problem on $\mathbb{R}$. Let $z_1, z_2, z_3 \in \mathcal{Z}$ be variables that characterize models (e.g., the index of tasks and the class of data). For



(a) $\mathbb{R}$-valued model (b) C^*-algebra-valued model

Figure 8. Difference between $\mathbb{R}$-valued and C^*-algebra-valued models.

$\mathbb{R}$-valued models, we separately learn three models and get $\mathbf{f}_i^{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}_{i,1}\mathbf{x} + \boldsymbol{\theta}_{i,2}$ for $i = 1, 2, 3$ and $\mathbf{x} \in \mathbb{R}$. Here, $\boldsymbol{\theta}_{i,j} \in \mathbb{R}$ is an $\mathbb{R}$-valued parameter. On the other hand, for C^*-algebra-valued models, we aggregate three models continuously and represents them as a function. Then, we learn the model and get $f^{\theta}(x)(z) = \theta_{i,1}(z)x(z) + \theta_{i,2}(z)$ for $i = 1, 2, 3$ and $x \in \mathcal{A}$, which corresponds to learn the models simultaneously with interactions.