arXiv:2302.01191v2 [math.OA] 6 Jul 2024

非交换 C^* 代数网络：利用 C^* 代数中强大的乘积结构学习神经网络

Ryuichiro Hataya* YukaHashimoto*RIKEN AIP
NTT& RIKEN AIP

摘要

我们提出了一种新的神经网络参数空间推广形式，将非交换C^*代数纳入其中，而这种代数具有丰富的非交换乘积结构。我们展示了这种非交换结构能够对神经网络学习产生强大的影响。我们的框架应用范围广泛，例如通过相互作用同时学习多个相关神经网络，以及学习关于群作用的等变特征。数值实验验证了该框架的有效性及其潜在优势。

1 引言

将神经网络的参数空间推广到实数之外，可以带来一些有趣的性质。例如，将复数（Hirose, 1992; Nishikawa et al., 2005; Amin et al., 2008; Yadav et al., 2005; Trabelsi et al., 2018; Lee et al., 2022）或四元数（Nitta, 1995; Arena 等人, 1997; Zhu 等人, 2018; Gaudet 和 Maida, 2018）用作神经网络参数，更为直观且有效，尤其适用于信号处理、计算机视觉和机器人学领域。作为这些数的推广，Clifford 代数支持更灵活的几何数据处理，并被应用于神经网络，以处理数据中丰富的几何关系（Pearson and Bisset, 1994; Buchholz, 2005; Buchholz and Sommer, 2008; Rivera-Rovelo 等人, 2010; Zang 等人, 2022; Brandstetter 等人, 2022; Ruhe 等人, 2023a,b），例如由物理定律推导出的约束。与这些着眼于参数值几何视角的方法不同，另一种推广方向是使用函数值参数（Rossi and Conan-Guez, 2005; Thind et al., 2023），从而拓宽神经网络在函数型数据上的应用。

在这项工作中，我们进一步将这些神经网络的参数空间推广到非交换 C^*代数，并引入非交换 C^*代数网络。 C^*代数是复数的推广，具有乘积和对合结构（Murphy，1990；Hashimoto 等人，2021年）。 C^*代数的典型例子包括方阵空间、紧空间上的连续函数空间以及克利福德代数（ Harpe，1972）。因此， C^*代数是对本节开头讨论的神经网络参数空间的推广。

事实上，Hashimoto 等人（2022年）已经将交换 C^*-代数引入神经网络参数中，而这正是我们工作所用非交换 C^*-代数的一个特例。具体而言，他们采用紧空间上的连续函数作为交换 C^*-代数，并提出了对函数值神经网络的一种新解释：无穷多个实值或复值神经网络连续地组合成一个单一的 C^*-代数网络。例如，对于同一任务，使用不同训练数据集或不同初始参数的网络可以

*作者对本项工作贡献相同。

连续组合，从而利用无穷多个组合网络之间的共享信息，实现高效学习。网络之间的这种交互类似于从相关任务中学习，例如集成学习（Dong 等人，2020年；Ganaie 等人，2022年）和多任务学习（Zhang 和 Yang，2022年）。然而，Hashimoto 等人（2022年）关注的 C^*-代数中的乘积结构是交换的，并且仅限于两个函数的逐点乘积。因此，他们的网络无法利用 C^*-代数丰富的乘积结构，反而需要专门设计的损失函数来促成必要的交互。

为了充分利用丰富的乘积结构，我们需要使用非交换 C^*-代数。交换 C^*-代数的一个简单例子是对角矩阵构成的空间，而非交换代数的一个例子则是方形非对角矩阵构成的空间。一对对角矩阵的乘积，是通过将各个对角元素分别相乘来计算的。另一方面，两个方形非对角矩阵的乘积，是不同元素之间乘积的总和；每个所得的对角元素都通过非对角元素依赖于其他对角元素，这可以解释为对角元素之间的交互。将这种源自非交换性的乘积结构用作神经网络参数时，能够发挥强大的作用。以矩阵上的 C^*-代数为例，参数为非对角矩阵的神经网络，可以将每个对角元素视为此类网络的参数，从而自然地诱导多个具有实值或复值参数的神经网络之间的交互。由于交互已编码在参数中，非交换 C^*-代数网络无需专门设计损失函数来诱导交互。这一特性使所提出的框架与现有的交换 C^*-代数网络形成了鲜明对比。非交换 C^*-代数的另一个例子是群C^*-代数，它由定义在群上的函数构成，其乘积结构由卷积给出。将其用作神经网络的参数空间时，无需设计特殊的网络架构，该网络便天然具有群等变性。我们通过多种数值实验验证了这些非交换 C^*-代数网络的能力。

本文的主要贡献总结如下：

- 我们将神经网络的参数空间推广到非交换 C^*-代数，并引入非交换 C^*-代数网络。它是 Hashimoto 等人（2022年）提出的交换C^*-代数网络的推广。非交换 C^*-代数网络能够利用非交换 C^*-代数强大的乘积结构来学习神经网络。

- 作为具体例子，我们给出一般非交换 C^*-代数网络的两种实例：矩阵上的 C^*-代数网络和群 C^*-代数网络。矩阵上的 C^*-代数网络能够自然地结合多个具有相互作用的实值或复值神经网络。具有群 C^*-代数参数的神经网络无需修改神经网络结构即可自然实现群等变性。

- 数值实验展示了这些非交换 C^*-代数网络的有效性：神经网络之间的相互作用以及群等变性。

我们强调， C^*-代数是神经网络的强大工具，而我们的工作为其应用提供了许多重要见解。

2 背景

本节回顾本文所需的 C^* 代数以及现有的 C^* 代数网络的数学背景。关于 C^* 代数的更多理论细节，请参见例如 Murphy（1990）。

2.1 C^*-代数

C^* 代数是复数值空间的一种推广，具有乘积、对合 * 和范数等结构。

定义 1（C^*-代数）集合 $\mathcal{A}$ 称为 C^*-代数，若其满足以下条件：*1.* $\mathcal{A}$ 是定义在 $\mathbb{C}$ 上的代数，并配备一个双射 $(\cdot)^*:\mathcal{A}\to\mathcal{A}$，且对于 α、$\beta\in\mathbb{C}$ 和 c、$d\in\mathcal{A}$满足以下条件：

- $(\alpha c+\beta d)^*=\alpha\overline{c}^*+\beta\overline{d}^*$,
- $(cd)^*=d^*c^*$,
- $(c^*)^*=c$。

2. $\mathcal{A}$ 是一个赋范空间，范数为 $\|\cdot\|$，且对于 c、$d\in\mathcal{A}$、$\|cd\|\le\|c\|\,\|d\|$ 成立。此外，$\mathcal{A}$ 关于 $\|\cdot\|$是完备的。

3. 对于 $c\in\mathcal{A}$，$\|c^*c\|=\|c\|^2$（满足 C^*-性质）。

C^*-代数中的乘积结构可以是交换的，也可以是非交换的。

例 1（交换 C^*-代数）设 $\mathcal{A}$ 是紧 *Hausdorff* 空间 $\mathcal{Z}$ 上连续函数构成的空间。我们可以通过如下定义将 $\mathcal{A}$ 视为一个 C^*-代数：

- 乘积：两个函数的逐点乘积，即对于 a_1，$a_2\in\mathcal{A}$，$a_1a_2\left(z\right)=a_1\left(z\right)a_2\left(z\right)$。
- 对合：逐点取复共轭，即对于 $a\in\mathcal{A}$，$a^*\left(z\right)=\overline{a\left(z\right)}$。
- 范数：上确界范数，即对于 $a\in\mathcal{A}$，$\|a\|=\text{上确界}_{z\in\mathcal{Z}}\left|a\left(z\right)\right|$。

在这种情况下，$\mathcal{A}$ 中的乘积是交换的。

示例 2（非交换 C^*-代数）令 $\mathcal{A}$ 为希尔伯特空间 $\mathcal{H}$上的有界线性算子空间，记作 $\mathcal{B}(\mathcal{H})$。我们可以通过如下定义将 $\mathcal{A}$ 视为一个 C^*-代数

- 乘积：两个算子的复合，
- 对合：算子的伴随，
- 范数：算子的算子范数，即对于 $a\in\mathcal{A}$，$\|a\|=\text{上确界}_{v\in\mathcal{H},\ \|v\|_{\mathcal{H}}=1}\|av\|_{\mathcal{H}}$。

这里，$\|\cdot\|_{\mathcal{H}}$ 是 $\mathcal{H}$中的范数。在这种情况下，$\mathcal{A}$ 中的乘积是非交换的。注意，如果 $\mathcal{H}$ 是一个维数为有限自然数 d的空间，那么 $\mathcal{A}$ 中的元素是由 d 矩阵构成的 d。

例 3（群 C^*-代数）有限群 G 上的群 C^*-代数记作$C^*(G)$，它是从 G 到 $\mathbb{C}$ 的映射集合，并配备以下乘积、对合和范数：

- 乘积：$(a\cdot b)(g)=\sum_{h\in G}a(h)b(h^{-1}g)$ 对 $g\in G$,
- 对合：$a^*(g)=\overline{a(g^{-1})}$ 对 $g\in G$,
- 范数：$\|a\|=\sup_{[\pi]\in\hat{}G}\|\pi(a)\|$,

$\hat{}$其中，G 是 G 的不可约酉表示的等价类集合。本文主要关注 $C^*(G)$ 的乘积结构。关于群表示的详细内容，请参见 *Kirillov*（*1976*年）。若 $G=\mathbb{Z}/p\mathbb{Z}$，则 $C^*(G)$ 与循环矩阵的 C^*-代数 C^*-同构（*Hashimoto* 等人，*2023*年）。另请注意，若 G 是非交换的，则 $C^*(G)$ 也可能是非交换的。

例 4（克利福德代数）对于配备二次型 $Q(v)=\|v\|_{\mathcal{H}}^2$ 的实希尔伯特空间 $\mathcal{H}$ 及其标准正交基 $\{e_1,\ldots,e_n\}$（属于 $\mathcal{H}$），克利福德代数记作 $Cl(\mathcal{H})$，其构造方式为 $e_{i_1}\cdots e_{i_k}$，其中 $1\le i_1<\cdots i_k\le n$且 $0\le k\le n$。令 $\{\tilde{e}_i\}$为 $Cl(\mathcal{H})$的标准正交基。对于 $e=\sum_i\alpha_i\tilde{e}_i$，我们将范数 $\|e\|_2$定义为 $\|e\|_2^2=\sum_i\alpha^2$。我们可以将 $Cl(\mathcal{H})$ 视为实 C^*代数，其

- 乘积：乘积结构定义为 $e_ie_j = -e_je_i$ 和 $e_i^2 = Q(e_i)$。
- 对合： $e^* = -e$ 对 $e \in Cl(\mathcal{H})$。
- 范数： $\|e\| = \sup_{\|f\|_2=1} \|ef\|_2$。

2.2 C^*-代数网络

Hashimoto 等人（2022年）提出，将实值神经网络参数推广为交换的 C^*-代数值参数，从而能够将多个实值模型表示为单个C^*-代数网络。这里，我们简要回顾现有的（交换） C^*-代数网络。令 $\mathcal{A} = C(\mathcal{Z})$为紧豪斯多夫空间 $\mathcal{Z}$上连续函数构成的交换 C^*-代数。令 H为网络深度， $N_0, \ldots, N_H$ 为每层的宽度。对于 $i = 1, \ldots, H$，令$W_i : \mathcal{A}^{N_{i-1}} \to \mathcal{A}^{N_i}$ 为由$\mathcal{A}^{N_i}$中的 $N_i \times N_{i-1}$ $\mathcal{A}$值矩阵和$\mathcal{A}$值偏置向量定义的仿射变换。此外，令非线性激活函数 $\sigma_i : \mathcal{A}^{N_i} \to \mathcal{A}^{N_i}$。交换 C^*-代数网络 f : $\mathcal{A}^{N_0} \to \mathcal{A}^{N_H}$ 定义如下:

$$f = \sigma_H \circ W_H \circ \cdots \circ \sigma_1 \circ W_1. \tag{1}$$

通过将神经网络参数推广为函数，我们可以连续地组合多个标准（实值）神经网络，从而使它们能够高效学习。本文将 C^*-代数网络中的每个实值网络称为子模型。下面我们给出一个交换C^*-代数网络的示例。为简化记号，我们重点讨论网络不含偏置的情形。不过，同样的论证也适用于网络含有偏置的情形。

2.2.1 对角矩阵的情形

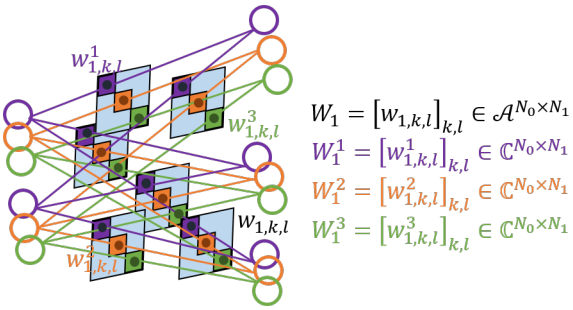
若 $\mathcal{Z}$ 是有限集，则 $\mathcal{A} = \{a \in \mathbb{C}^{d\times d} \mid a \text{ 是对角矩阵}\}$。 C^*-代数网络 f 在$\mathcal{A}$ 上对应于 d 个彼此独立的实值或复值子模型。在 $\mathcal{A} = C(\mathcal{Z})$ 的情形下，我们可以认为无穷多个网络被连续地组合在一起，而具有对角矩阵的 C^*-代数网络 f是 C^*-代数网络在 $C(\mathcal{Z})$ 上的离散化。事实上，记 x^j为由 $x \in \mathcal{A}^N$的第 j个对角元素组成的向量，其定义为$\mathbb{C}^N$ 中的向量，其中第 k个元素是 $\mathcal{A}$-值 k的第 x个元素的第 j个对角元素。假设激活函数 σ_i： $\mathcal{A}^N \to \mathcal{A}^N$ 定义为 $\sigma_i(x)^j = \tilde{\sigma}_i(x^j)$，其中 $\tilde{\sigma}_i$： $\mathbb{C}^N \to \mathbb{C}^N$。由于当 a_1， $a_2 \in \mathcal{A}$ 时， a_1a_2 的第j个对角元素是 a_1 的第 j个元素与 a_2的乘积，因此我们有

$$f(x)^j = \tilde{\sigma}_H \circ W_H^j \circ \cdots \circ \tilde{\sigma}_1 \circ W_1^j, \tag{2}$$

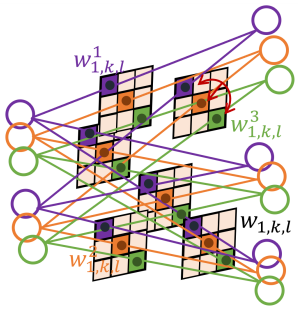
其中， $W_i^j \in \mathbb{C}^{N_i\times N_{i-1}}$ 是一个矩阵，其第（k， l）个条目为$W_i \in \mathcal{A}^{N_i\times N_{i-1}}$的第（$k$， l）个条目的第 j条对角线。图1(a)示意性地展示了定义在对角矩阵上的 C^*代数网络。

3 非交换 C^*代数网络

尽管现有的 C^*-代数网络为将 C^*-代数应用于神经网络提供了一个框架，但它关注的是交换 C^*-代数，其乘积结构较为简单。因此，我们将现有的交换 C^*-代数网络推广到非交换 C^*-代数。由于非交换 C^*-代数中的乘积结构比交换 C^*-代数中的更复杂，它们使神经网络能够更高效地学习数据特征。例如，如果我们关注矩阵的 C^*-代数，那么神经网络参数描述的就是多个实值子模型之间的相互作用（见第 3.1.1 节）。



(a) 基于对角矩阵的 C^*-代数网络（交换）。蓝色部分表示对角矩阵中的零部分（非对角部分）。每个对角元素都对应一个独立的子模型。



(b) 基于非对角矩阵的 C^*-代数网络（非交换）。与对角矩阵的情况不同，非对角部分（橙色部分）不为零。非对角元素会在多个子模型之间产生交互。

图 1：从子模型之间交互的角度看，交换与非交换 C^*-代数网络的区别。

设 $\mathcal{A}$ 是一个一般的 C^*-代数，并考虑与公式（1）形式相同的网络 f。我们强调，在我们的框架中，$\mathcal{A}$ 的选择不受交换 C^*-代数的限制。下面列出 $\mathcal{A}$ 的示例及其在学习神经网络方面的有效性。

3.1 神经网络的 C^*-代数示例

通过展示若干 C^*-代数的示例，我们说明 C^*-代数网络是一个灵活且统一的框架，可将 C^*-代数纳入神经网络。正如上一节所述，为简化起见，本小节重点讨论网络不含偏置的情况。

3.1.1 非对角矩阵

令 $\mathcal{A} = \mathbb{C}^{d \times d}$。注意，$\mathcal{A}$ 是一个非交换 C^*代数。当然，也可以考虑矩阵数据，但我们按照第 2.2 节的思路，重点关注子模型之间的交互。在这种情况下，与网络 (2) 不同，对于 a_1，a_2，$a_3 \in \mathcal{A}$，$a_1 a_2 a_3$ 的第 j 个对角元素不仅取决于 a_2的第 j 个对角元素，也取决于 a_2的其他对角元素。因此，$f(x)^j$不仅取决于第 2.2.1 节讨论的、对应于第 j 个对角元素的子模型，也取决于其他子模型。 $\mathcal{A}$中的非对角元素会在 d 个实值或复值子模型之间引发相互作用。

子模型之间的交互可能与去中心化点对点机器学习有关，参见 Vanhaesebrouck 等（2017）；Bellet 等（2018）。在这种机器学习中，每个智能体无需与其他智能体共享数据，同时通过通信利用其他智能体的信息来提升自身能力。在我们的情形中，一个智能体对应一个子模型，而通信通过交互实现。我们将在第 4.1 节中通过数值实验观察交互对 $\mathcal{A}$ 中非对角元素的影响。图 1（b）以示意图的形式展示了作用于非对角矩阵的 C^*-代数网络。

3.1.2 分块对角矩阵

令 $\mathcal{A}=\{a\in\mathbb{C}^{d\times d}\mid a=\mathbf{diag}(\mathbf{a}_1,\ldots,\mathbf{a}_m),\mathbf{a}_i\in\mathbb{C}^{d_i\times d_i}\}$。两个分块对角矩阵 $a=\mathbf{diag}(\mathbf{a}_1,\ldots,\mathbf{a}_m)$ 和 $b=\mathbf{diag}(\mathbf{b}_1,\ldots,\mathbf{b}_m)$ 的乘积可写为

$$ab=\mathrm{diag}(\mathbf{a}_1\mathbf{b}_1,\ldots,\mathbf{a}_m\mathbf{b}_m).$$

与第 2.2.1 节类似，我们用 $\mathbf{x}^j$ 表示由 $x\in\mathcal{A}^N$ 的第 j 个对角块组成的 $N\times d_j$ 矩阵。假设激活函数 $\sigma_i:\mathcal{A}^N\to\mathcal{A}^N$ 定义为 $\sigma_i(x)=\mathbf{diag}(\tilde{\boldsymbol{\sigma}}_i^1(\mathbf{x}^1)\ldots,\tilde{\boldsymbol{\sigma}}_i^m(\mathbf{x}^m))$，其中 $\tilde{\boldsymbol{\sigma}}_{i,j}:\mathbb{C}^{N\times d_j}\to\mathbb{C}^{N\times d_j}$。那么，我们有

$$\mathbf{f}(\mathbf{x})^j=\tilde{\boldsymbol{\sigma}}_H^j\circ\mathbf{W}_H^j\circ\cdots\circ\tilde{\boldsymbol{\sigma}}_1^j\circ\mathbf{W}_1^j, \tag{3}$$

其中，$\mathbf{W}_i^j\in(\mathbb{C}^{d_j\times d_j})^{N_i\times N_{i-1}}$ 是分块矩阵，其 (k,l) 项为 $W_i\in\mathcal{A}^{N_i\times N_{i-1}}$ 的 (k,l) 项的第 j 个块对角部分。在这种情况下，我们有 m 组子模型，每组均由第 3.1.1 节所述的相互作用的 d_j 个子模型组成。实际上，分块对角情形推广了第 2.2.1 节和第 3.1.1 节中所述的对角情形和非对角情形。若对所有 $j=1,\ldots,m$ 均有 $d_j=1$，则网络 (3) 将简化为采用对角矩阵的网络 (2)。若 $m=1$ 且 $d_1=d$，则网络 (3) 将简化为采用 $d\times d$ 非对角矩阵的网络。

3.1.3 循环矩阵

设 $\mathcal{A}=\{a\in\mathbb{C}^{d\times d}\mid a$ 是一个循环矩阵$\}$。这里，循环矩阵 a 是表示为

$$a=\begin{bmatrix}a_1 & a_d & \cdots & a_2\\ a_2 & a_1 & \cdots & a_3\\ & \ddots & \ddots & \\ a_d & a_{d-1} & \cdots & a_1\end{bmatrix}$$

对于 a_1，……，$a_d\in\mathbb{C}$。注意，在这种情况下，$\mathcal{A}$ 是交换的。循环矩阵可由离散傅里叶矩阵对角化，如下所示（Davis，1979）。我们用 F 表示离散傅里叶变换矩阵，其 $(i,\ j)$ 元素为 $\omega^{(i-1)(j-1)}/\sqrt{p}$，其中 $\omega=\mathrm{e}^{2\pi\sqrt{-1}/d}$。

引理 1 任意循环矩阵 a 均可分解为 $a=F\Lambda_aF^*$，其中

$$\Lambda_a=\mathrm{diag}\left(\sum_{i=1}^d a_i\omega^{(i-1)\cdot 0},\ldots,\sum_{i=1}^d a_i\omega^{(i-1)(d-1)}\right).$$

由于 $ab=F\Lambda_a\Lambda_bF^*$ 对于 $a,\ b\in\mathcal{A}$，a 与 b 的乘积对应于 a 和 b 的各个傅里叶分量相乘。假设激活函数 $\sigma_i:\mathcal{A}^N\to\mathcal{A}^N$ 的定义满足：$(F^*\sigma_i(x)F)^j$ 等于 $\hat{}\,\tilde{\sigma}_i((FxF^*)^j)$，其中 $\hat{}\,\tilde{\sigma}_i:\mathbb{C}^N\to\mathbb{C}^N$。于是，我们得到网络

$$(F^*f(x)F)^j=\hat{\tilde{\sigma}}_H\circ\hat{W}_H^j\circ\cdots\circ\hat{\tilde{\sigma}}_1\circ\hat{W}_1^j, \tag{4}$$

其中，$\hat{}\,W_i^j\in\mathbb{C}^{N_i\times N_{i-1}}$ 是一个矩阵，其 (k,l) 项为 $(Fw_{i,k,l}F^*)^j$，其中 $w_{i,k,l}$ 是 $W_i\in\mathcal{A}^{N_i\times N_{i-1}}$ 的 (k,l) 项。网络 (4) 的第 j 个子模型对应于第 j 个傅里叶分量的网络。

注 1 网络 *(4)* 的第 j 个子模型不会与第 j 个傅里叶分量之外的其他傅里叶分量对应的子模型发生交互。这一事实对应于在此情况下 $\mathcal{A}$ 是交换的。与第 *3.1.1* 节的情况类似，如果我们将 $\mathcal{A}$ 设为非循环矩阵，那么不同傅里叶分量对应的子模型之间就会发生相互作用。

3.1.4 非对称群上的群 C^*代数

令 G 为集合 $\{1,\dots,d\}$ 上的对称群，并令 $\mathcal{A} = C^*(G)$。注意，由于 G 是非交换的， $C^*(G)$ 也是非交换的。那么，输出 $f(x) \in \mathcal{A}^{N_H}$ 是 G 上的 $\mathbb{C}^{N_H}$ 值映射。利用例 3 中引入的乘积结构，我们可以构造一个将数据置换纳入考量的网络。事实上，权重矩阵 $W \in \mathcal{A}^{N_{i-1}\times N_i}$ 的一个元素 $w \in \mathcal{A}$ 是 G 上的函数。因此， $w(g)$ 描述了与置换 $g \in G$ 对应的权重。由于 $x \in C^*(G)$ 与 w 的乘积定义为 $wx(g) = \sum_{h\in G} w(h)x(h^{-1}g)$，通过应用 W，与各个置换对应的所有权重都会作用于输入。例如，令 $z \in \mathbb{R}^d$，并将 $x \in C^*(G)$ 设为 $x(g) = g\cdot z$，其中 $g\cdot z$ 是 g 在 z 上的作用，即相对于 g 的 z 置换。这样，我们就能同时输入 z 的所有置换模式，并凭借 $C^*(G)$ 中的乘积结构，使网络通过这些置换之间的相互作用进行学习。对于输出，如果网络经过训练，使输出 y 成为 G 上的常值函数，即对于某个常数 c，有 $y(g) = c$，那么这意味着 c 关于 g 不变，也就是说，它关于该置换不变。我们将在第 4.2 节通过数值实验研究群 C^* 代数网络在置换不变问题上的应用。

注 *2* 如果激活函数 σ 定义为 $\sigma(x)(g) = \sigma(x(g))$，即逐元素作用于 x，那么网络 f 具有置换等变性。也就是说，即使将输入 $x(g)$替换为某个 $h \in G$对应的 $x(gh)$，输出 $f(x)(g)$也会替换为 $f(x)(gh)$。这是因为 $C^*(G)$中的乘积被定义为卷积。卷积的这一特性已在群等变神经网络中得到研究（*Lenssen* 等人，*2018*年；*Cohen* 等人，*2019*年；*Sannai* 等人，*2021*年；*Sonoda* 等人，*2022*年）。上述 C^*代数网络的设置从 C^*代数的角度为我们提供了一种设计群等变网络的方法。

注 3 由于 G 的元素个数为 $d!$， $C^*(G)$中的元素（即定义在 G上的函数）可表示为 $d!$ 维向量。当 d 较大时，我们需要一种高效计算的方法，这属于未来的工作。

3.1.5 希尔伯特空间上的有界线性算子

对于函数型数据，我们也可以将 $\mathcal{A}$ 设为无限维空间。有人提出使用无限维C^*代数来分析函数型数据（Hashimoto 等人，2021年）。我们也可以将这一思想用于神经网络。设 $\mathcal{A} = \mathcal{B}(L^2(\Omega))$，其中 Ω 为测度空间。令 $\mathcal{A}_0 = \{a \in \mathcal{A} \mid a \text{ 为乘法算子}\}$。这里，乘法算子 a 是一个线性算子，其定义为：对于某个 $u \in L^\infty(\Omega)$， $av = v\cdot u$。空间 $\mathcal{A}_0$ 是对角矩阵空间向无限维空间的推广。如果将权重矩阵的元素限制为 $\mathcal{A}_0$，那么我们可以得到无穷多个彼此不存在相互作用的子模型。由于输出属于 $\mathcal{A}_0^{N_H}$，因此可以将函数型数据作为输出。与矩阵的情况类似（参见第 3.1.1 节），通过将权重矩阵的元素设为 $\mathcal{A}$ 中的元素，我们可以利用无穷多个子模型之间的相互作用。

3.2 利用 C^*-代数网络逼近具有相互作用的函数

我们考察 C^*-代数网络能够近似哪类函数。在本小节中，我们证明 C^*-代数网络的普适性，这描述了模型的表征能力。我们重点关注 $\mathcal{A} = \mathbb{C}^{d\times d}$ 的情形。考虑一个浅层网络 $f\colon \mathcal{A}^{N_0} \to \mathcal{A}$定义为 $f(x) = W_2^*\sigma(W_1x+b)$，其中 $W_1 \in \mathcal{A}^{N_1\times N_0}$、 $W_2 \in \mathcal{A}^{N_1}$和 $b \in \mathcal{A}^{N_1}$。令 $\tilde{f}\colon \mathcal{A}^{N_0} \to \mathcal{A}$ 为具有如下形式的函数

$\tilde{f}(x) = [\sum_{j=1}^{d} f_{kj}(x^l)]_{kl}$，其中 f_{kj}：$\mathbb{C}^{N_0 d} \to \mathbb{R}$。这里，我们滥用记号，将 $x^l \in \mathbb{C}^{N_0 d}$ 记作 x 的第 l 列，并将其视为一个 $N_0 d \times d$ 矩阵。假设 f_{kj} 表示为

$$f_{kj}(x) = \int_{\mathbb{R}} \int_{\mathbb{R}^{N_0 d}} T_{kj}(w, b)\sigma(w^* x + b)\mathrm{d}w\,\mathrm{d}b \tag{5}$$

对于某个 T_{kj}： $\mathbb{R}^{N_0 d} \times \mathbb{R} \to \mathbb{R}$。根据脊波变换理论，在大多数现实设置下，这样的 T_{kj} 都存在（Cand`es，1999年；Sonoda 和 Murata，2017年）。例如，Sonoda 和 Murata（2017年）给出了如下结果。

命题 *1* 令 $\mathcal{S}$ 为 $\mathbb{R}$ 上的快速衰减函数空间， $\mathcal{S}_0'$ 为 $\mathbb{R}$上的 *Lizorkin* 分布空间的对偶空间。假设函数 $\tilde{f}$ 具有 $\tilde{f}(x) = [\sum_{j=1}^{d} f_{kj}(x^l)]_{kl}$的形式，且 f_{kj} 和 $\hat{} f_{kj}$ 属于 $L^1(\mathbb{R}^{N_0 d})$，其中 $\hat{} f$ 表示函数 f的傅里叶变换。进一步假设 σ 属于 $\mathcal{S}_0'$，并且存在 $\psi \in \mathcal{S}$，使得 $\int_{\mathbb{R}} \hat{\psi}(x)\hat{\sigma}(x)/|x|\mathrm{d}x$ 非零且有限。则存在 T_{kj}： $\mathbb{R}^{N_0 d} \times \mathbb{R} \to \mathbb{R}$，使得 f_{kj} 具有公式（*5*）的表示形式。

这里，Lizorkin 分布空间定义为 $\mathcal{S}_0 = \{\phi \in \mathcal{S} \mid \int_{\mathbb{R}} x^k \phi(x)\mathrm{d}x = 0\ k \in \mathbb{N}\}$。注意，ReLU 属于 S_0'。我们通过将 Lebesgue 测度替换为 $\sum_{i=1}^{N_1} \delta_{w_{ij}}$和 $\sum_{i=1}^{N_1} \delta_{b_{ij}}$，对方程 (5) 进行离散化，其中 δ_w是以 w为中心的 Dirac 测度。于是，$\tilde{f}(x)$ 的 $(k,\ l)$ 项可写为

$$\sum_{j=1}^{d} \sum_{i=1}^{N_1} T_{kj}(w_{ij}, b_{ij})\sigma(w_{ij}^* x^l + b_{ij}).$$

令 $W_2 \in \mathcal{A}^{N_1}$的第 i 个元素为 $[T_{kj}(w_{ij},\ b_{ij})]_{kj}$， $W_1 \in \mathcal{A}^{N_1 \times N_0}$的 $(i,\ m)$ 项为$[(w_{i,\ j})_{md+l}]_{jl}$，$b \in \mathcal{A}^{N_1}$ 的第 i 个元素为 $[b_j]_{jl}$，可得

$$\sum_{j=1}^{d} \sum_{i=1}^{N_1} T_{kj}(w_{ij}, b_{ij})\sigma(w_{ij}^* x^l + b_{ij}) = (W_2^k)^* \sigma(W_1 x^l + b^l),$$

这就是 $f(x)$ 的 $(k,\ l)$ 项。因此，在满足命题 1 中假设的条件下，任何形式为 $\tilde{f}(x) = [\sum_{j=1}^{d} f_{kj}(x^l)]_{kl}$的函数都可以表示为 C^*-代数网络。

备注 4 正如我们在第 *2.2.1* 节和第 *3.1.1* 节中所讨论的，定义在矩阵上的 C^*-代数网络可以˜视为 d 个相互作用的子模型。上述论证要求， $f(x)$和 $f(x)$的第 l 列仅取决于 x^l。因此，在这种情况下，如果我们输入与第 l 个子模型相对应的数据 x^l，那么输出就是 $\mathcal{A}$ 值输出 $f(x)$的第 l 列。另一方面，权重矩阵 W_1 和 W_2 以及偏置 b 会共同用于为任意子模型提供输出，即 W_1、 W_2和 b 是利用与所有子模型相对应的数据学习得到的。因此， W_1、 W_2和 b 在子模型之间产生相互作用。

4 实验

在本节中，我们通过考察矩阵上的非对角 C^*-代数网络，从交互和群 C^*-代数网络的等变性质角度，以数值实验展示非交换 C^*-代数网络的能力。为简化实验，我们使用 C^*-代数值多层感知机（多层感知机）。不过，它们也可以自然地扩展到其他神经网络，例如卷积神经网络。有关实验的更多信息，请参见附录 A。

4.1 C^*-代数矩阵网络

在由非对角矩阵参数构成的矩阵上的非交换 C^*-代数网络中，预期每个子模型都会与其他子模型交互，从而比由对角矩阵构成的交换对应模型取得更好的性能。我们在图像分类和神经隐式表示（NIR）任务中展示了这种交互的有效性。实验采用类似的点对点学习设置，数据在各个子模型之间分离。

记号参见第 3.1.1 节。训练第 j 个子模型（$j = 1$、2、.....、d）时，将一个原始的 N_0 维输入数据点 $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_0}] \in \mathbb{R}^{N_0}$ 转换为其对应的表示 $x \in \mathcal{A}^{N_0} = \mathbb{R}^{N_0 \times d \times d}$，使得对于 $i = 1$、2、.....、N_0，$x_{i,\,j,\,j} = \boldsymbol{x}_i$，其他情况下为 0。 C^*-代数网络 $y \in \mathcal{A}^{N_H}$ 的 N_H 维输出与目标 $t \in \mathcal{A}^{N_H}$ 之间的损失计算为 $\ell\left(y_{:,\,j,\,j}, t_{:,\,j,\,j}\right) + \frac{1}{2}\sum_k, \quad (l \neq j) \quad \left(y^2_{k,\,j,\,l} + y^2_{k,\,l,\,j}\right)$，其中 ℓ 是某种损失函数；图像分类任务使用均方误差（MSE），神经隐式表示（NIR）任务使用 Huber损失。第二项和第三项将输出中的非对角元素抑制至 0。在这两个示例中，我们使用带泄漏的ReLU作为激活函数，并且仅将其应用于预激活值的对角元素。

4.1.1 图像分类

我们使用 MNIST（Le Cun 等，1998年）、Kuzushiji-MNIST（Clanuwat 等，2018年）和 Fashion-MNIST（Xiao 等，2017年）开展图像分类任务实验，这些数据集由 10 类 28×28 灰度图像组成。每个子模型都在从原始训练数据中抽取的互不重叠子集上进行训练，然后在完整的测试数据上进行评估。每个子集均以平衡方式抽取，即每个类别包含相同数量的训练样本。作为基线，我们使用基于对角矩阵的交换 C^*-代数网络，该网络由相同的子模型组成，但无法与其他子模型进行交互。非交换模型和交换模型共享以下超参数：层数设为 4，隐藏层大小设为 128，模型训练 30 个训练轮数。

表1展示了平均测试准确率。如表中所示，非交换 C^*代数网络始终优于其交换对应模型，这一优势十分显著，尤其是在子模型数量为40时。性能提升可能归因于非交换模型的可训练参数多于交换模型。因此，表2还比较了宽度为128的交换 C^*代数网络与宽度为8的非交换 C^*代数网络，两者的可学习参数数量几乎相同。我们仍然可以观察到，非交换 C^*代数网络大幅优于交换模型。这些结果表明，非交换 C^*代数网络的性能不能仅用可学习参数的数量来解释：子模型之间的交互提供了关键能力。

此外，表3表明，相关任务通过交互有助于提升性能。具体而言，我们为每个数据集分别准备五个子模型，数据集为 MNIST、Kuzushiji-MNIST 和 Fashion-MNIST，并同时训练一个由 15 个子模型组成的单一（非）交换 C^*代数网络。除了交换 C^*代数网络（其中子模型之间没有交互）和非交换 C^*代数网络（其中每个子模型都可以与其他任意子模型交互）之外，我们还使用了一个具有块对角矩阵的非交换 C^*代数网络（见第3.1.2节），其中每个子模型只能与在同一数据集上训练的其他子模型交互。请注意，这种块对角变体等同于表1中的模型。表3显示，完全非交换的 C^*代数网络优于块对角网络，这表明任务内交互和任务间交互都有助于提升性能。

表1：交换与非交换 C^*-代数网络在测试数据集上的矩阵平均测试准确率。每个子模型使用 1,000 个数据点进行训练。当子模型数量为 40 时，子模型之间的相互作用使非交换 C^*-代数网络的性能显著提升。

数据集	子模型数量	交换	非交换
MNIST	5	0.711 ± 0.009	0.863 ± 0.001
	10	0.718 ± 0.012	0.871 ± 0.000
	20	0.704 ± 0.005	0.884 ± 0.000
	40	0.679 ± 0.003	0.892 ± 0.000
Kuzushiji-MNIST	5	0.452 ± 0.018	0.598 ± 0.001
	10	0.469 ± 0.007	0.612 ± 0.001
	20	0.459 ± 0.004	0.627 ± 0.001
	40	0.446 ± 0.001	0.635 ± 0.000
Fashion-MNIST	5	0.635 ± 0.004	0.777 ± 0.001
	10	0.637 ± 0.006	0.790 ± 0.000
	20	0.634 ± 0.005	0.793 ± 0.000
	40	0.624 ± 0.002	0.794 ± 0.000

表 2：由 20 个子模型组成的矩阵上的交换与非交换 C^*-代数网络的平均测试准确率。非交换模型的宽度设为 8，以使其可学习参数的数量与交换基线相匹配。

数据集	交换	非交换
MNIST	0.704 ± 0.005	0.833 ± 0.001
Kuzushiji-MNIST	0.459 ± 0.004	0.558 ± 0.001
Fashion-MNIST	0.634 ± 0.005	0.722 ± 0.003

表3：在三个数据集上同时训练的五个子模型的平均测试准确率。完全非交换的 C^*-代数网络在 Kuzushiji-MNIST 和 Fashion-MNIST 上优于块对角非交换的 C^*-代数网络，这表明交互可以利用相关任务。

数据集	交换	非交换	
		块对角	Full
MNIST	0.694 ± 0.013	0.861 ± 0.001	0.874 ± 0.000
Kuzushiji-MNIST	0.451 ± 0.006	0.596 ± 0.001	0.621 ± 0.001
Fashion-MNIST	0.634 ± 0.005	0.774 ± 0.000	0.791 ± 0.000

4.1.2 神经隐式表示

在下一个实验中，我们使用基于矩阵的 C^*代数网络，学习二维图像的隐式表示，将每个像素坐标映射到其 RGB 颜色（Sitzmann 等人，2020年；Xie 等人，2022年）。具体而言，将 $[0,1]^2$ 中的输入坐标转换为$[-1,1]^{320}$ 中的随机傅里叶特征，然后将其转换为基于矩阵的 C^*代数表示，作为基于矩阵的C^*代数网络的输入。与图像分类任务类似，我们比较非交换

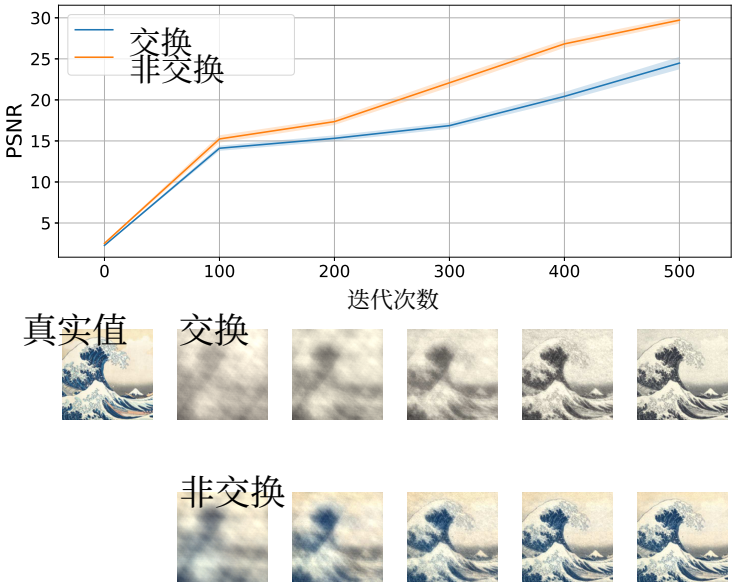


图 2：下方图像（上）的隐式表示的平均峰值信噪比，以及每经过 100 次迭代重建的真实值图像（下）。非交换 C^*-代数网络比其交换对应模型更快地学习图像的几何结构和颜色。

将交换神经隐式表示与神经隐式表示进行比较，采用以下超参数：层数设为 6，隐藏维度设为 256。这些神经隐式表示从大都会艺术博物馆的浮世绘图像[1] 以及 AFHQ 数据集中的猫的照片（Choi 等人，2020）中学习 128×128 像素图像。

图 2（上）展示了对应于下方图像的子模型平均峰值信噪比曲线。交换和非交换 C^*-代数网络均由五个子模型组成，分别在五幅浮世绘图像上训练（另见图 6）。作为质量指标的峰值信噪比在非交换神经隐式表示中增长更快，相应地，它比交换版本更快地学会真实值图像的细节（图 2 下）。值得注意的是，非交换表示在学习初期就能重现颜色，而交换表示在训练 500 次迭代后仍保持单色。猫的图像中也观察到了类似趋势（图 3），这些结果进一步凸显了交互的有效性。更长期的结果见图 7。

这种用于二维图像的隐式神经表示可以扩展到表示三维模型。图 4 展示了使用相同的 C^*-代数多层感知机，在 ShapeNet 数据集（Chang 等人，2015）中的三个 3D椅子上进行训练后，得到的三维隐式表示的合成视图。所展示的位姿在训练期间未曾出现。同样，非交换隐式神经表示重建的椅子模型噪声伪影更少，表明交互有助于高效学习。详情和结果请参见附录 A 和 B。

4.2 群 C^*-代数网络

作为 C^*-代数网络的另一个实验示例，我们展示群 C^*-代数网络；我们在第 3.1.4 节中介绍了这种网络。群 C^*-代数网络以对称群上的函数为输入，并返回群上的函数作为输出。

请参阅章节 关于记号，请参见 3.1.4。一个群 C^*-代数网络在数据 $\{(x, y) \in$ 上训练

[1]https://www.metmuseum.org/art/the-collection

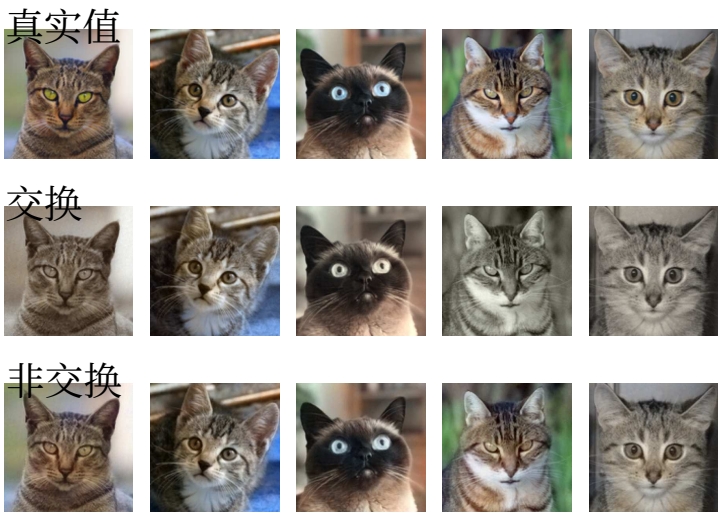


图 3：交换和非交 C^*代数网络在经过 500 次训练迭代后的真实值图像及其隐式表示。非交换 C^* 代数网络能够更忠实地再现色彩。



图 4：交换和非交换C^*代数网络在经过 5000 次训练迭代后的三维隐式表示合成视角。非交换 C^* 代数网络能够生成更精细的细节。请注意，左侧的椅子无法由交换 C^* 代数网络合成。

$\mathcal{A}^{N_0} \times \mathcal{A}^{N_H}\}$，其中 x和 y是 N_0 维和 N_H 维的向量值函数。在实际应用中，这类函数可以表示为实张量，例如 $x \in \mathbb{C}^{N_0 \times \#G}$，其中 $\#G$ 表示 G 的大小。利用第 3.1.4 节所述的函数间乘积以及逐元素加法，可以在 $\mathcal{A}$ 上构造线性层，进而构造多层感知机。与基于矩阵的 C^*-代数网络类似，我们使用 Leaky ReLU 作为激活函数。

群 C^*-代数网络最简单的任务之一是学习置换不变表示，例如预测给定 d 数字之和。在这种情况下， x 是一个输出 d 数字特征置换的函数，而 $y(g)$ 是一个常函数，对所有 $g \in G$ 都返回这些数字之和。在这项实验中，我们使用预训练卷积神经网络提取的 MNIST 数字特征，并将其表示为 32 维向量。为简化问题，我们选取数字之和小于 10 的样本，并训练模型使用交叉熵损失对给定数字之和进行分类。

表4：不变DeepSet模型、等变DeepSet模型和群 C^*-代数网络在数字求和任务测试数据上的平均测试准确率（训练100轮后）。群 C^*-代数网络只需较少的数据即可学习。

数据集大小	不变 DeepSet	等变 DeepSet	群 C^*-代数网络
1k	0.408 ± 0.014	0.571 ± 0.021	0.783 ± 0.016
5k	0.731 ± 0.026	0.811 ± 0.007	0.922 ± 0.003
10k	0.867 ± 0.021	0.836 ± 0.009	0.943 ± 0.005
50k	0.933 ± 0.005	0.862 ± 0.002	0.971 ± 0.001

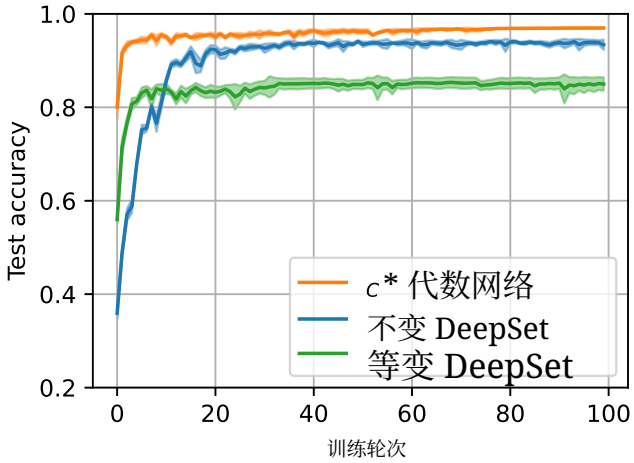


图5：在数字求和任务的 10k 条数据上训练的不变 DeepSet、等变 DeepSet 和群C^*-代数网络的平均测试准确率曲线。群 C^*-代数网络能够更高效、更有效地学习。

我们将层数设为 4，隐藏维度设为 32。作为比较，我们准备了置换不变和置换等变 DeepSet 模型（Zaheer 等人，2017年），它们采用特殊结构来实现置换不变性或等变性，其参数数量与群 C^*-代数网络相同。

表4 展示了 $d = 3$ 时不同训练数据集大小下该任务的结果。表中最突出的是，群 C^*-代数网络始终以较大优势胜过 DeepSet 模型，尤其是在训练数据数量有限时。此外，如图5所示，群 C^*-代数网络的收敛所需迭代次数远少于 DeepSet 模型。这些结果表明，群 C^*-代数网络中乘积结构引入的归纳偏置是有效的。

5 相关工作

将代数结构应用于神经网络的研究已有数十年历史。例如，四元数被用于空间变换、多维信号和彩色图像（Nitta，1995；Arena 等人，1997；Zhu 等人，2018；Gaudet 和 Maida，2018）。克利福德代数（或几何代数）是四元数的一种推广，将克利福德代数应用于神经网络也已得到研究，以提取数据的几何结构（Rivera-Rovelo 等人，2010；Zang 等人，2022；Brandstetter 等人，2022；Ruhe 等人，2023a、b）。Hoffmann 等人（2020年）考虑了参数为矩阵值的神经网络，以提高参数和计算效率，并

实现神经网络的丰富结构。 C^*-代数是一个统一上述所有代数的通用框架。本节将回顾将代数应用于神经网络的现有研究，并讨论采用 C^*-代数的优势。

四元数是复数的一种推广。四元数表示为 $a + b\mathrm{i} + c\mathrm{j} + d\mathrm{k}$，其中 $a, b, c, d \in \mathbb{R}$。这里，i、j 和 k 是满足 $\mathrm{i}^2\mathrm{j}^2 = \mathrm{k}^2 = -1$、 $\mathrm{ij} = -\mathrm{ji} = \mathrm{k}$、 $\mathrm{ik} = -\mathrm{ki} = \mathrm{j}$ 以及 $\mathrm{jk} = -\mathrm{kj} = \mathrm{i}$ 的基元素。Nitta（1995）；Arena 等人（1997）提出并分析了采用四元数值参数的神经网络。由于三维空间中的旋转可以用四元数表示，因此四元数可用于控制机器人的位置（Fortuna 等人，1996）。近来，有研究提出并探讨了使用四元数表示彩色图像，并通过四元数版本的卷积神经网络对其进行分析的方法（Zhu 等人，2018年；Gaudet 和 Maida，2018年）。

克利福德代数是四元数的一种推广，使我们能够提取数据的几何结构。它自然地统一了实数、向量、复数、四元数、外代数等。技术细节请参见例 4。我们不仅有向量 $e_1, \ldots, e_n$，还有形式为 $e_i e_j$ （双向量）、$e_i e_j e_k$ （三向量）等的元素。利用这些不同类型的向量，我们可以描述不同领域中的数据。Brandstetter 等人（2022年）；Ruhe 等人（2023a）提出，使用克利福德代数神经网络求解偏微分方程，并通过克利福德代数描述不同场之间的相关性。人们还提出了基于克利福德代数的群等变网络，用于提取在群作用下保持等变的特征（Ruhe 等人，2023b）。Zang 等人（2022年）使用参数取值为克利福德代数的残差网络分析交通数据，以考虑其在空间域和时间域中的相关性。Rivera-Rovelo 等人（2010）使用克利福德代数网络近似二维或三维物体的表面。Hoffmann 等人（2020年）探讨了将神经网络参数推广为矩阵的做法。这些网络在参数规模和计算成本方面可能具有优势。他们还探讨了采用矩阵值参数时网络设计的灵活性。

C^*-代数是一个更一般的框架，统一了函数代数、矩阵、算子和克利福德代数。与其他代数相比，采用 C^*-代数的一个优势在于，可以直接推广与神经网络相关的概念。这得益于对合、范数和 C^*-性质的结构。例如，我们可以通过 C^*-代数推广希尔伯特空间，得到希尔伯特 C^*-模（Lance，1995）。由于输入空间和输出空间都是希尔伯特空间，因此可以利用希尔伯特 C^*-模理论分析C^*-代数网络。我们还可以自然地推广再生核希尔伯特空间（RKHS），得到再生核希尔伯特 C^*-模（RKHM）（Hashimoto 等人，2021年）。RKHM 使我们能够将 C^*-代数网络与核方法联系起来（Hashimoto 等人，2023年）。

从应用于神经网络的角度来看， C^*-代数和克利福德代数都能让我们引入相互作用。克利福德代数可以利用双向量和三向量来描述数据各组成部分之间的关系。 C^*-代数也可以利用其乘积结构引入数据各组成部分之间的交互。此外，正如我们在第 3.1.1 节中讨论的那样，它还可以引入子模型之间的交互。正如我们在第 3.1.4 节中讨论的那样，我们的框架还使我们能够构建群等变神经网络。

6 结论与讨论

在本文中，我们将神经网络参数空间推广到了非交换C^*代数。其丰富的乘积结构为神经网络带来了强大的性质。例如，基于非对角矩阵的 C^*代数网络能够使其子模型相互作用，而一个

群 C^*代数网络能够学习置换等变特征。我们已通过图像分类、神经隐式表示和数字求和任务等各种任务，从实验上验证了这些性质的有效性。

非交换 C^*-代数网络目前在实际应用中的一项局限是计算成本较高。实验中使用的基于矩阵的非交换 C^*-代数网络，其通信复杂度随子模型数量呈二次增长，与分布式计算中的“全归约”集体操作相同。通过参数共享，或借鉴自注意力及其高效变体，为非对角元素引入结构等方式，可以缓解这一复杂度。群 C^*-代数网络的时间复杂度甚至随集合大小呈阶乘增长，因此集合规模一旦增大，很快就会变得难以实现。通过使用参数不变/等变神经网络（如 DeepSet）表示参数，或许可以缓解这种高昂的复杂度。尽管计算负担沉重，引入非交换 C^*-代数仍能带来其他方法无法实现的有趣性质。我们将关于可扩展性的进一步研究留待未来开展。

一个重要且有趣的未来研究方向是应用无限维 C^*-代数。本文主要关注有限维 C^*-代数。我们展示了 C^*-代数中的乘积结构是神经网络的一种强大工具，例如，即使在有限维情形下，也能通过学习相互作用和群等变性（或不变性）发挥作用。无限维 C^*-代数使我们能够分析函数型数据，例如时间序列数据和空间数据。我们相信，应用无限维 C^*-代数能够成为一种高效的数据提取工具，即使观测数据存在部分缺失也是如此。将我们的框架实际应用于基于无限维 C^*-代数的函数型数据，是我们未来的研究方向。

我们基于非交换 C^*代数的框架具有普适性，应用范围广泛。我们相信，该框架为学习神经网络开辟了一条新途径。

参考文献

A. Hirose，《连续复值反向传播学习》，Electronics Letters 28 (1992) 1854–1855。I. Nishikawa、K. Sakakibara、T. Iritani、Y. Kuroe，《两种复值 Hopfield 网络及其在交通信号控制中的应用》，载于：IJCNN，2005。M. F. Amin、M. M. Islam、K. Murase，《用于实值分类问题的单层复值神经网络及其集成》，载于：IJCNN，2008。A. Yadav、D. Mishra、S. Ray、R. N. Yadav、P. Kalra，《复值神经网络的表示：一种实值方法》，载于：ICISIP，2005。C. Trabelsi、O. Bilaniuk、Y. Zhang、D. Serdyuk、S. Subramanian、J. F. Santos、S. Mehri、N. Ros- tamzadeh、Y. Bengio、C. J. Pal，《深度复数网络》，载于：ICLR，2018。C. Lee、H. Hasegawa、S. Gao，《复值神经网络：全面综述》，IEEE/CAA 自动化学报 9 (2022) 1406–1426。T. Nitta，《反向传播算法的四元数版本》，载于：ICNN，第 5 卷，1995，第 2753–2756 页。P. Arena、L. Fortuna、G. Muscato、M. G. Xibilia，《用于逼近四元数值函数的多层感知机》，Neural Networks 10 (1997) 335–342。

X. Zhu、Y. Xu、H. Xu、C. Chen，《四元数卷积神经网络》，载于：ECCV，2018。C. J. Gaudet、A. S. Maida，《深度四元数网络》，载于：IJCNN，IEEE，2018，第 1–8 页。J. Pearson、D. Bisset，《克利福德域中的神经网络》，载于：ICNN，1994。S. Buchholz，《基于克利福德代数的神经计算理论》，博士论文，2005。S. Buchholz、G. Sommer，《关于克利福德神经元和克利福德多层感知器》，Neural Networks 21 (2008) 925–935。J. Rivera-Rovelo、E. Bayro-Corrochano、R. Dillmann，《用于二维轮廓和三维表面重建的几何神经计算》，载于：Geometric Algebra Computing，Springer，2010，第 191–209 页。D. Zang、X. Chen、J. Lei、Z. Wang、J. Zhang、J. Cheng、K. Tang，《用于交通数据预测的多通道几何代数残差网络》，IET智能交通系统 16 (2022) 1549–1560。J. Brandstetter、R. v. d. Berg、M. Welling、J. K. Gupta，《用于偏微分方程建模的克利福德神经层》，2022。ArXiv:2209.04934。D. Ruhe、J. K. Gupta、S. de Keninck、M. Welling、J. Brandstetter，《几何克利福德代数网络》，2023a。ArXiv:2302.06594。D. Ruhe、J. Brandstetter、P. Forr´e，《克利福德群等变神经网络》，2023b。ArXiv:2305.11141。F. Rossi、B. Conan-Guez，《函数型多层感知器：一种用于函数型数据分析的非线性工具》，Neural Networks 18 (2005) 45–60。B. Thind、K. Multani、J. Cao，《函数型输入的深度学习》，Journal of Computational and Graphical Statistics 32 (2023) 171–180。G. J. Murphy，《C^*-代数与算子理论》，Academic Press，1990。Y. Hashimoto、I. Ishikawa、M. Ikeda、F. Komura、T. Katsura、Y. Kawahara，《再生核希尔伯特 C^*-模与核均值嵌入》，Journal of Machine Learning Research 22 (2021) 1–56。P. Harpe，《希尔伯特空间的克利福德代数与旋量群》，Compositio Mathematoca 25 (1972) 245–261。Y. Hashimoto、Z. Wang、T. Matsui，《C^*-代数网络：将神经网络参数推广到 C^*-代数的新方法》，载于：ICML，2022。X. Dong、Z. Yu、W. Cao、Y. Shi、Q. Ma，《集成学习综述》，Frontiers of Computer Science 14 (2020) 241–258。M. A. Ganaie、M. Hu、A. K. Malik、M. Tanveer、P. N. Suganthan，《集成深度学习：综述》，Engineering Applications of Artificial Intelligence 115 (2022) 105151。Y. Zhang、Q. Yang，《多任务学习综述》，IEEE知识与数据工程汇刊 34 (2022) 5586–5609。A. A. Kirillov，《表示理论要义》，Springer，1976年。

Y. Hashimoto, M. Ikeda, H. Kadri, 《RKHM 中的学习：核机器的一个 C^*代数变体》，载于：AISTATS，2023年。P. Vanhaesebrouck, A. Bellet, M. Tommasi, 《网络上的个性化模型去中心化协同学习》，载于：AISTATS，2017年。A. Bellet, R. Guerraoui, M. Taziki, M. Tommasi, 《个性化与隐私保护的点对点机器学习》，载于：AISTATS，2018年。P. J. Davis, 《循环矩阵》，Wiley，1979。J. E. Lenssen, M. Fey, P. Libuschewski, 《群等变胶囊网络》，载于：NeurIPS，2018年。T. S. Cohen, M. Geiger, M. Weiler, 《齐性空间上等变卷积神经网络的一般理论》，载于：NeurIPS，2019年。A. Sannai, M. Imaizumi, M. Kawano, 《通过商特征空间改进群不变/等变深度网络的泛化界》，载于：UAI，2021年。S. Sonoda, I. Ishikawa, M. Ikeda, 《基于群上的脊波分析的群卷积神经网络普适性》，载于：NeurIPS，2022年。E. J. Cand`es, 《神经网络的调和分析》，应用与计算调和分析 6 (1999) 197–218。S. Sonoda, N. Murata, 《具有无界激活函数的神经网络是通用逼近器》，应用与计算调和分析 43 (2017年) 233–268。Y. Le Cun, L. Bottou, Y. Bengio, P. Haffner, 《应用于文档识别的基于梯度的学习》，IEEE会刊 86 (1998年) 2278–2324。T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, D. Ha, 《面向日本古典文学的深度学习》，arXiv (2018年)。H. Xiao, K. Rasul, R. Vollgraf, 《Fashion-MNIST：用于机器学习算法基准测试的新型图像数据集》，arXiv (2017年)。V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, G. Wetzstein, 《具有周期激活函数的隐式神经表示》，载于：NeurIPS，2020年。Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, S. Sridhar, 《视觉计算及其他领域中的神经场》，计算机图形学论坛 (2022年)。Y. Choi, Y. Uh, J. Yoo, J.-W. Ha, 《StarGAN v2：面向多个域的多样化图像合成》，载于：CVPR，2020年。A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, F. Yu, 《ShapeNet：信息丰富的三维模型库》，技术报告，斯坦福大学 — 普林斯顿大学 — 丰田芝加哥技术研究所，2015年。M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, A. J. Smola, 《深度集合》，载于：NeurIPS，2017年。

J. Hoffmann, S. Schmitt, S. Osindero, K. Simonyan, E. Elsen, 《Algebranets》，2020年。ArXiv:2006.07360。L. Fortuna, G. Muscato, M. Xibilia, 《用于机器人定位的超复神经网络平台》，载于：1996年 IEEE国际电路与系统研讨会。连接世界的电路与系统。（ISCAS 96），1996。E. C. Lance, 《Hilbert C^*-模——算子代数学家工具包》，伦敦数学学会讲义系列，第210卷，剑桥大学出版社，纽约，1995。J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, 《JAX：Python+NumPy 程序的可组合变换》，2018年。网址：http://github.com/google/jax。P. Kidger, C. Garcia, 《Equinox：通过可调用 PyTrees 和筛选变换在 JAX 中构建神经网络》，Neural Information Processing Systems 2021 可微编程研讨会（2021年）。I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, J. Quan, G. Papamakarios, R. Ring, F. Ruiz, A. Sanchez, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, L. Wang, W. Stokowiec, F. Viola, 《DeepMind JAX 生态系统》，2020年。网址：http://github.com/deepmind。D. P. Kingma, J. L. Ba, 《Adam：一种随机优化方法》，载于：ICLR，2015。M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. T. Barron, R. Ng, 《用于优化基于坐标的神经表示的学习到的初始化》，载于：CVPR，2021年。

A 实现细节

我们使用 JAX（Bradbury 等，2018年）、Equinox（Kidger 和 Garcia，2021年）以及 optax（Babuschkin 等，2020年）实现了 C^*-代数网络。对于矩阵上的 C^*-代数网络，我们使用 Adam优化器（Kingma 和 Ba，2015），学习率为 1.0×10^{-5}；3D神经隐式表示实验除外，其 Adam 初始学习率设为 4.5×10^{-4}。对于由 d 个子模型组成的非交换C^*-代数网络，学习率按$\sqrt{d}$缩放。对于群-C^*-代数网络，我们采用 Adam优化器，学习率为 1.0×10^{-3}。除 2D神经隐式表示（每个批次包含所有像素）和 3D神经隐式表示（批量大小设为 4）外，所有实验的批量大小均设为 32。清单 1 和 2 展示了使用 NumPy 实现的 C^*-代数网络线性层，这些实现与第 4.1 节和第 4.2 节实验中使用的实现等价。

3D神经隐式表示（第4.1.2节）的实现基于一种简单的类 NeRF 模型及其渲染器，二者均取自 Tancik 等人（2021年）。训练时，采用 ShapeNet 数据集（Chang 等人，2015）中每个 3D椅子的 25 个视角及其 64×64 像素参考图像。与二维实验使用相同的 C^*-代数多层感知机，仅超参数有所不同：层数为四层，隐藏维度大小为 128。

第 4.2 节中使用的置换不变 DeepSet 模型以一个包含四层多层感知机（采用双曲正切激活）、求和池化和线性分类器的结构处理每个数据样本。置换等变 DeepSet 模型由四个置换等变层组成，采用双曲正切激活、最大池化和线性分类器，其设计遵循 Zaheer 等人（2017年）的点云分类方法。虽然我们尝试将带泄漏ReLU激活用于群 C^*-代数网络，但这一设置在置换不变 DeepSet 中产生了次优结果。DeepSet 模型的隐藏维度设为 96，以匹配群 C^*-代数网络的浮点数参数数量。

```
导入 NumPy 作为np

def matrix_值_线性(weight: Array, 偏置: Array, 输入: Array ) -> Array:"""
weight: Array of shape {output_dim}x{input_dim}x{dim_matrix}x{dim_matrix}
bias: Array of shape {output_dim}x{dim_matrix}x{dim_matrix}
input: Array of shape {input_dim}x{dim_matrix}x{dim_matrix}
out: Array of shape {output_dim}x{dim_matrix}x{dim_matrix}"""输出 = []for _
weight, b in zip(weight, bias): tmp =np.zeros(weight.shape[2:]) for w, i in zip(_
weight, input): tmp += w @ i +b out.append(tmp) return np.array(out)
```

列表 1：第 4.1 节中使用的、基于矩阵的 C^* 代数网络线性层的 NumPy 实现。



图 6：交换与非交换 C^*-代数网络经过 500 次训练迭代后的真实值图像及其隐式表示。

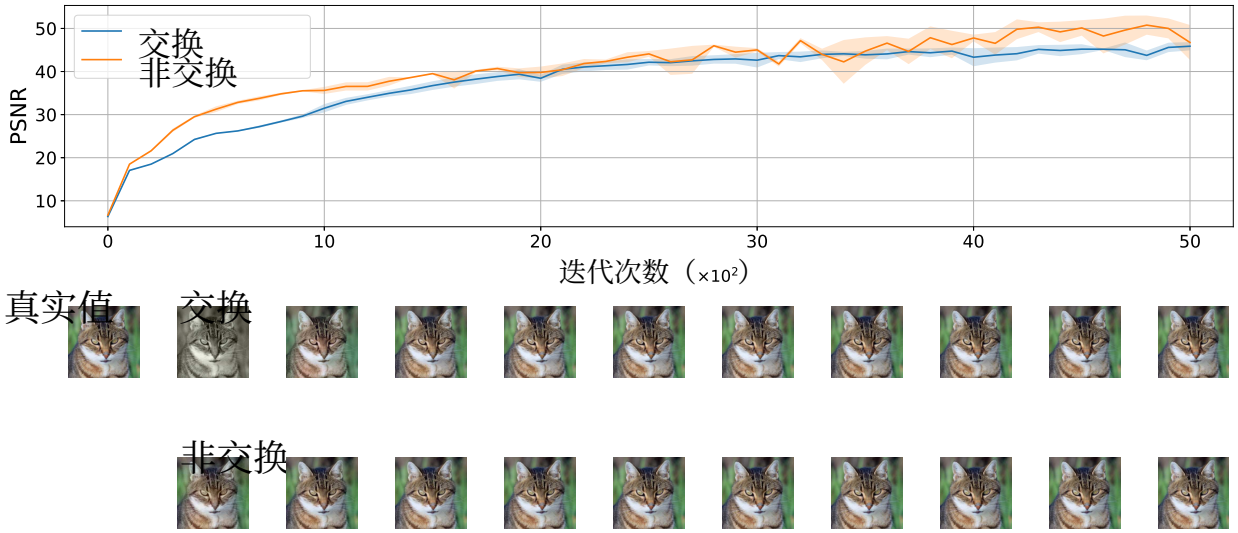


图 7：在五张猫的图片（上）上训练的交换与非交换 C^*-代数网络对图像进行隐式表示所得的平均峰值信噪比，以及每经过 500 次迭代对真实值图像的重建结果（下）。

B 更多结果

图 6 和图 7 展示了二维隐式神经表示的补充图示（第4.1.2节）。图 6 是正文中图 3 的浮世绘版本。同样，非交换 C^* 代数网络比交换网络更快地学习颜色细节。图 7 展示了图像在不同随机状态下初始化后，经过 5,000 次迭代的三个神经隐式表示的平均峰值信噪比曲线。虽然效果不如初始阶段显著，但收敛后，非交换 C^* 代数网络仍优于交换网络。

表5和图8展示了三维隐式神经表示的更多结果（第4.1.2节）。表5列出了合成视角的平均峰值信噪比。从图4和图8中的合成视角可以看出，非交换 C^*-代数网络生成的输出噪声较少，因此峰值信噪比较高。

图 9 展示了 group C^*-代数网络和 DeepSet 模型在子



图8：椅子的隐式表示的合成视图。

表5：合成视图的平均峰值信噪比。视角的指定位姿在训练期间未出现。

交换	非交换
18.40 ± 4.30	25.22 ± 1.45

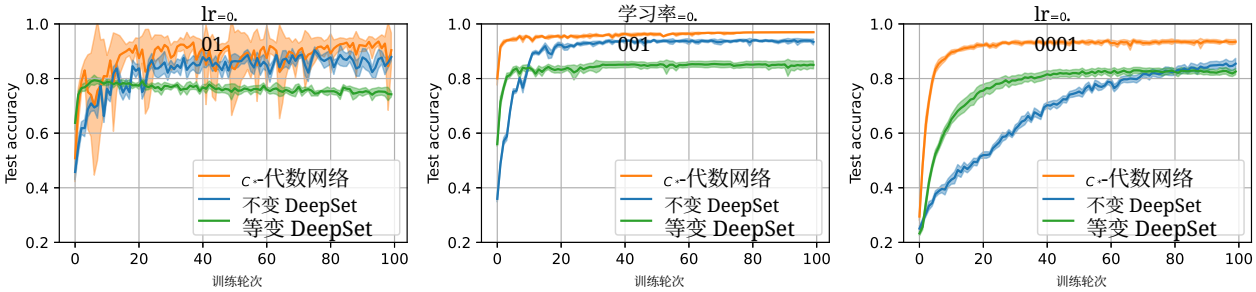


图 9：群 C^*-代数网络与 DeepSet 模型在不同学习率下的测试准确率曲线对比。

不同学习率下的数字任务。如图5所示，当学习率为 0.001 时，群 C^*-代数网络在较宽的学习率范围内，都比 DeepSet 模型用少得多的迭代次数收敛；不过，当学习率较大（0.01）时，所提出的模型表现不稳定。

```
import datacl
    asses import numpy as np @dataclasses.dataclass class 置换:
# helper class to handle permutationvalue: np.ndarray def inverse(self) ->置换:
return 置换(self.value.argsort()) def __mul__(self, perm: 置换)-> 置换: return 置换
(self.value[perm.value]) def __eq__(self, other: 置换)-> bool: return np.all(self.value ==
other.value) @staticmethod def create_hashtable(set_size: int) -> Array: perms = [置换
(np.array(p)) for p in permutations(range(set_size))]map ={v: i for i, v in
enumerate(perms)} out = []for perm in perms: out.append([map[perm.inverse() * _
perm] for _perm in perms]) return np.array(out)
```

```
定义 群_线性(权重: 数组, 偏置: 数组,
输入: 数组 ) -> 数组:"""
    weight: {output_dim}x{input_dim}x{group_size}bias: {output_dim}x{group_size}
    input: {input_dim}x{group_size}out: {output_dim}x{group_size}"""哈希表 = 置换.
    create_哈希表(set_大小) # {group_size}x{group_size}g = np.arange(hashtable.shape
    [0]) 输出 = []对于 _权重 in weight: tmp0 = []对于 y in g: tmp1 = []对于 w, f in zip(_
    weight, input): tmp2 = []对于 x in g: tmp2.append(w[x] * f[hashtable[x, y]])
    tmp1.append(sum(tmp2)) tmp0.append(sum(tmp1)) out.append(tmp0) return
    np.array(out) +偏置
```

清单 2：第 4.2 节中使用的群 C^*-代数线性层的 NumPy 实现。