AI Applications Lecture 12
Image Generation 2: Natural Image Decoder from Latent Space/Low-Dimensional Images

SUZUKI, Atsushi
Jing WANG

2025-10-28

Contents

1 Introduction

1.1 Recap of Previous Lecture

In the previous lecture, we saw that a practical **text-to-image** AI pipeline consists of three components: a **text encoder**, a **latent generator** (image generator in a low-dimensional latent space), and a **natural image decoder** (Figure 1).

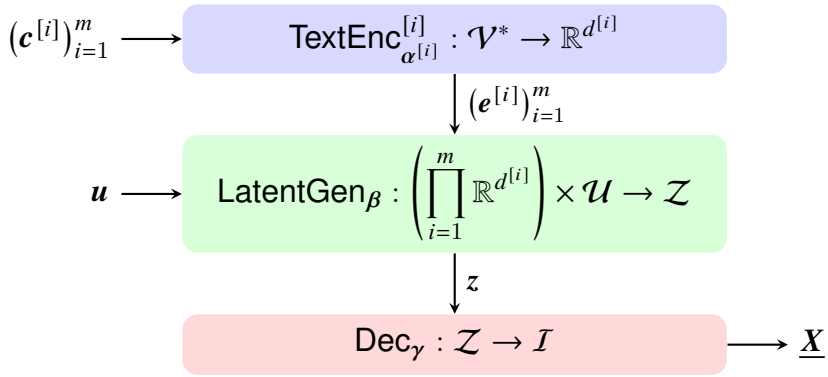

Figure 1: Recap of the three-component pipeline.

1.2 Learning Outcomes

The learning outcomes for this lecture are as follows. By the end of this lecture, students should be able to explain the following:

- **Computational cost reduction**: Be able to strictly explain how a natural image decoder reduces the overall **computational cost** in a practical pipeline.
- **Meaning of the three key properties**: Be able to explain the significance of the three key properties of a **fully-convolutional variational autoencoder (VAE)** used as a natural image decoder—namely, **being an autoencoder** (an Encoder/Decoder pair), being based on the **variational principle**, and **being fully convolutional**—in the context of **practical generative AI for images**.

2 Preparation: Mathematical Notations

- **Definition:**
  - $(\mathrm{LHS}) \coloneqq (\mathrm{RHS})$: The left-hand side is defined by the right-hand side. For example, $a \coloneqq b$ defines a by b.
- **Set:**
  - Sets are often denoted by uppercase calligraphic letters. Example: $\mathcal{A}$.

- $x \in \mathcal{A}$: The element x belongs to the set $\mathcal{A}$.
- $\{\}$: The empty set.
- $\{a, b, c\}$: The set consisting of elements a, b, c (extensional notation).
- $\{x \in \mathcal{A} \mid P(x)\}$: Set-builder notation (intensional notation).
- $|\mathcal{A}|$: The number of elements (cardinality). (In this lecture, we generally assume finite sets).
- $\mathbb{R}$: The set of all real numbers. The notations $\mathbb{R}_{>0}, \mathbb{R}_{\geq 0}, \mathbb{Z}, \mathbb{Z}_{>0}, \mathbb{Z}_{\geq 0}$ have their usual meanings.
- $[1, k]_{\mathbb{Z}}$: For $k \in \mathbb{Z}_{>0} \cup \{+\infty\}$, $[1, k]_{\mathbb{Z}} \coloneqq \{1, 2, \ldots, k\}$. When $k = +\infty$, $[1, k]_{\mathbb{Z}} = \mathbb{Z}_{>0}$.

- **Function:**
  - $f : \mathcal{X} \to \mathcal{Y}$ is a map (function). $y = f(x)$ is the output $y \in \mathcal{Y}$ for $x \in \mathcal{X}$.

- **Vector:**
  - Vectors are assumed to be column vectors. $\boldsymbol{v} \in \mathbb{R}^n$ is

$$\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}. \tag{1}$$

  - The standard inner product is

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle \coloneqq \sum_{i=1}^{d_{\mathrm{emb}}} u_i v_i \tag{2}$$

    .

- **Sequence:**
  - $\boldsymbol{a} : [1, n]_{\mathbb{Z}} \to \mathcal{A}$ is called a sequence of length n from the set $\mathcal{A}$. $a_i \coloneqq \boldsymbol{a}(i)$. Notations for finite and infinite sequences are used accordingly.

- **Matrix:**
  - Let the (i, j)-th element of $\boldsymbol{A} \in \mathbb{R}^{m,n}$ be $a_{i,j}$,

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix}. \tag{3}$$

- The transpose $\boldsymbol{A}^\top \in \mathbb{R}^{n,m}$ is

$$\boldsymbol{A}^\top = \begin{bmatrix} a_{1,1} & a_{2,1} & \cdots & a_{m,1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,n} & a_{2,n} & \cdots & a_{m,n} \end{bmatrix}. \tag{4}$$

- The transpose of a vector is

$$\boldsymbol{v}^\top = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix} \in \mathbb{R}^{1,n}. \tag{5}$$

- **Tensor:**
  - Refers to a multi-dimensional array. A vector is a 1st-order tensor, a matrix is a 2nd-order tensor, and 3rd-order or higher tensors are written as $\underline{\boldsymbol{A}}$.

3 The Goal of a Natural Image Decoder

3.1 Goal Reconfirmation: Computational Cost Reduction and Semantic Constraints

A natural image decoder is a map

$$\mathrm{Dec}_\gamma : \mathcal{Z} \to \mathcal{I} \tag{6}$$

from a low-dimensional latent space $\mathcal{Z} \subset \mathbb{R}^{h \times w \times c}$ to a high-dimensional image space $\mathcal{I} \subset \mathbb{R}^{H \times W \times C}$, which is trained such that its image $\mathrm{Dec}_\gamma(\mathcal{Z})$ becomes a **subset** that sufficiently covers the set of **natural images** (though not the entire high-dimensional image space) (Figure 2). This **reduces the burden on other components**. That is, with an ideal natural image decoder, other components only need to output a point in the low-dimensional latent space $\mathcal{Z}$ instead of directly outputting a point in the high-dimensional image space $\mathcal{I}$. Hereafter, this input space $\mathcal{Z}$ is called the **latent space**.

3.2 Why It Cannot Cover the Entire High-Dimensional Space (Discrete and Continuous Perspectives)

In the discussion of the previous section, it is important that we gave up on covering the entire high-dimensional image space from the beginning. When the domain is low-dimensional, one cannot expect the image of a continuous function to cover a high-dimensional space.

From a discrete perspective, the total number of representable vectors at a fixed precision increases exponentially with the dimension. Therefore, it is trivial that Dec_γ cannot cover the entire high-dimensional space when $d_\mathcal{Z} < d_\mathcal{I}$.

From a continuous perspective, we can discuss this using the Hausdorff dimension.

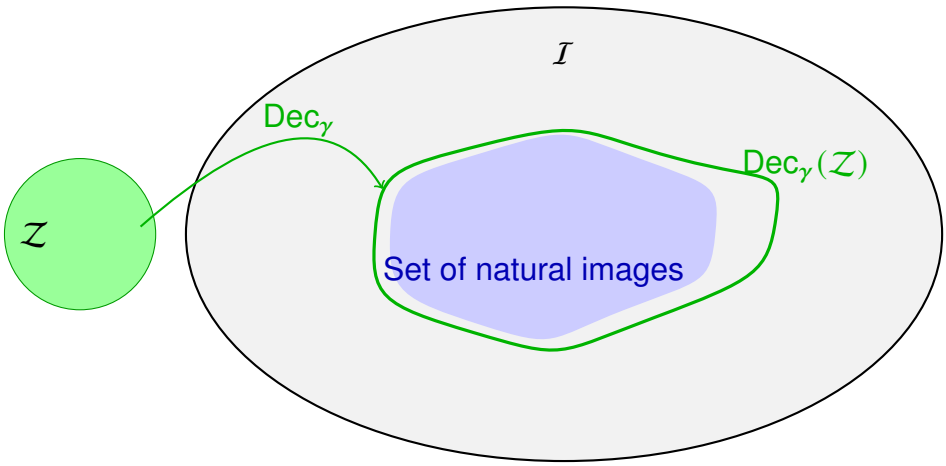

Figure 2: The image of the natural image decoder is a **subset** of the high-dimensional image space $\mathcal{I}$ and covers the region of natural images of interest.

Definition 3.1 (Hausdorff measure and dimension)**.** For a non-empty subset $A \subset \mathbb{R}^n$ and $s \geq 0$, the s-dimensional Hausdorff outer measure for $\delta > 0$ is defined as

$$\mathcal{H}^s_\delta(A) \coloneqq \inf\left\{\sum_{i=1}^{\infty} \omega_s \left(\frac{1}{2}\operatorname{diam} U_i\right)^s \,\middle|\, A \subset \bigcup_{i=1}^{\infty} U_i,\ \operatorname{diam} U_i < \delta\right\} \tag{7}$$

and

$$\mathcal{H}^s(A) \coloneqq \lim_{\delta \downarrow 0} \mathcal{H}^s_\delta(A) \tag{8}$$

is called the s-dimensional Hausdorff measure. Here, $\operatorname{diam} U \coloneqq \sup\{\|x-y\| :\ x, y \in U\}$. ω_s is defined using the Gamma function Γ as $\omega_s \coloneqq \frac{\pi^{s/2}}{\Gamma\left(\frac{s}{2}+1\right)}$. Note that this is defined such that the volume of a d-dimensional hypersphere of radius r is $\omega_s r^d$. The Hausdorff dimension is defined by

$$\dim_{\mathcal{H}}(A) \coloneqq \inf\{s \geq 0 :\ \mathcal{H}^s(A) = 0\} = \sup\{s \geq 0 :\ \mathcal{H}^s(A) = \infty\} \tag{9}$$

. [1]

Remark 3.1. Although it is non-trivial, the Hausdorff dimension of a non-empty subset of a metric space can always be defined and always has a value in $[0, +\infty]$.

A measure, in essence, means ”volume”. Intuitively, the s-dimensional Hausdorff measure of a subset A is the sum of the hypervolumes of sufficiently small hyperspheres covering it, where the hypervolume of each hypersphere is treated as if it were an s-dimensional sphere corresponding to its radius. For a 2-dimensional object, let’s consider a square, for example. The act of considering the 1-dimensional Hausdorff measure is like forcibly measuring its length; since a square is like an infinite collection of 1-dimensional line segments, the 1-dimensional Hausdorff measure of a square is expected to be $+\infty$. On the other hand,

the act of considering the 3-dimensional Hausdorff measure of a square is like forcibly measuring its volume; since a square is like a cuboid with height 0, the 3-dimensional Hausdorff measure of a square is 0. A similar argument actually holds for a positive real number s: the s-dimensional Hausdorff measure of a square is zero if $s < 2$, positive infinity if $s > 2$, and takes a finite positive value only when $s = 2$. Therefore, the Hausdorff dimension of a square is 2, which matches our familiar, naive concept of dimension. This also holds for hypercubes of more general dimensions.

Proposition 3.1 (Hausdorff dimension of a hypercube)**.** Let $d \in \mathbb{Z}_{>0}$. We rigorously find the Hausdorff dimension of $A \coloneqq [0,1]^d \subset \mathbb{R}^d$ using the normalization $(\omega_s(\frac{1}{2}\operatorname{diam})^s)$ in Definitions (7), (8). The conclusion is

$$\dim_{\mathcal{H}}(A) = d \tag{10}$$

. In other words, $\mathcal{H}^s(A) = 0$ for $s > d$ (upper bound) and $\mathcal{H}^s(A) = \infty$ for $s < d$ (lower bound).

Now, what we want to know is how the dimension changes via a neural network. The function represented by a neural network is not necessarily differentiable everywhere, but in a normal situation where learning is done by backpropagation, it is at least a locally Lipschitz function. Below, we define Lipschitz continuity (globally) and local Lipschitz continuity, and finally, we will see that the Hausdorff dimension of the image of a locally Lipschitz function does not become larger than the Hausdorff dimension of its domain.

Definition 3.2 (Lipschitz map)**.** For metric spaces (X, d_X) and (Y, d_Y), a map $f : X \to Y$ is said to be **Lipschitz** if there exists a constant $L \geq 0$ such that for all $u, v \in X$,

$$d_Y\big(f(u), f(v)\big) \leq L\, d_X(u, v) \tag{11}$$

holds. The smallest such L is called the Lipschitz constant. In particular, for a map from $(\mathbb{R}^n, \|\cdot\|_2)$ to $(\mathbb{R}^m, \|\cdot\|_2)$, this is

$$\|f(u) - f(v)\|_2 \leq L\, \|u - v\|_2 \tag{12}$$

.

Definition 3.3 (Neighborhood in a metric space)**.** For a metric space (X, d) and a point $x \in X$, a set $U \subset X$ is a **neighborhood** of x if there exists a radius $r > 0$ such that the open ball $B(x, r) \coloneqq \{y \in X : \ d(x, y) < r\}$ satisfies $B(x, r) \subset U$. That is, a neighborhood is a set that contains x in its interior.

Definition 3.4 (Locally Lipschitz map)**.** A map $f : X \to Y$ between metric spaces (X, d_X) and (Y, d_Y) is **locally Lipschitz** if for each point $x \in X$, there exists a neighborhood (Definition 3.3) $U_x \subset X$ of x and a constant $L_{U_x} \geq 0$ such that for all $u, v \in U_x$,

$$d_Y\big(f(u), f(v)\big) \leq L_{U_x}\, d_X(u, v) \tag{13}$$

holds.

Example 3.1 (Examples of functions with Lipschitz and locally Lipschitz properties)**.** We prove the properties of each function step-by-step below (all on $(\mathbb{R}, |\cdot|)$).

- $f(x) = \sigma(x) := \frac{1}{1+e^{-x}}$ (Sigmoid) is **globally Lipschitz**. Indeed, $f'(x) = \sigma(x)\,(1-\sigma(x))$, so $0 \le f'(x) \le 1/4$ holds for all x. By the Mean Value Theorem, for any u, v:

$$|\sigma(u) - \sigma(v)| = |f'(\xi)|\,|u-v| \le \tfrac{1}{4}|u-v| \quad (\text{for some } \xi \text{ between } u, v). \tag{14}$$

  Thus, it is Lipschitz with $L = 1/4$.

- $f(x) = x^2$ is **not globally Lipschitz** but is **locally Lipschitz**. First, for any L, $|x^2 - y^2| = |x+y|\,|x-y|$. Since $|x+y| \le L$ cannot hold for all x, y, no global L exists (contradiction as $x \to \infty$). On the other hand, on any interval $[-R, R]$ for $R > 0$:

$$|x^2 - y^2| = |x+y|\,|x-y| \le 2R\,|x-y|, \qquad (|x|, |y| \le R), \tag{15}$$

  holds. Thus, $[-R, R]$ is a Lipschitz region with $L_U = 2R$. Since we can take a sufficiently small R for the neighborhood of any point x_0, it follows that f is locally Lipschitz.

- Consider $f(x) = \sqrt{x}$ on the domain $[0, \infty)$. This is **not locally Lipschitz at point** 0. We show this by contradiction. Assume there exist a neighborhood $U = (0, \delta)$ of 0 and $L > 0$ such that

$$|\sqrt{u} - \sqrt{v}| \le L\,|u-v| \quad (u, v \in U) \tag{16}$$

  holds. Substituting $v = 0$ and $u = t \in (0, \delta)$, we get

$$\sqrt{t} \le Lt \quad \Longleftrightarrow \quad \frac{1}{\sqrt{t}} \le L. \tag{17}$$

  As $t \downarrow 0$, the left side diverges to $+\infty$, which is a contradiction. Thus, f is not locally Lipschitz at 0. However, on $[\varepsilon, \infty)$ $(\varepsilon > 0)$, it is Lipschitz because $f'(x) = \frac{1}{2}x^{-1/2} \le \frac{1}{2}\varepsilon^{-1/2}$.

- $f(x) = \mathrm{ReLU}(x) := \max\{0, x\}$ is **globally Lipschitz** (even with the non-differentiable point 0). We analyze by cases for any u, v:

$$u, v \ge 0: \ |f(u) - f(v)| = |u - v|, \tag{18}$$

$$u, v \le 0: \ |f(u) - f(v)| = |0 - 0| = 0 \le |u - v|, \tag{19}$$

$$u \ge 0 \ge v: \ |f(u) - f(v)| = |u - 0| = u \le u - v = |u - v|. \tag{20}$$

  In all cases, $|f(u) - f(v)| \le |u - v|$ holds, so it is Lipschitz with $L = 1$.

Theorem 3.1 (Hausdorff dimension under Lipschitz maps)**.** For a set $A \subset \mathbb{R}^n$ and a locally Lipschitz map $f : \mathbb{R}^n \to \mathbb{R}^m$,

$$\dim_{\mathcal{H}}(f(A)) \leq \dim_{\mathcal{H}}(A) \tag{21}$$

holds. [1]

Remark 3.2. Examples where the dimension of the image is larger than the domain, such as **space-filling curves (Peano curves)**, are not locally Lipschitz. Functions represented by neural networks are, in principle, **locally Lipschitz** maps (activations are piecewise Lipschitz) to enable learning via gradient methods. Therefore, they cannot cover the entire high-dimensional space.

4 Fully Convolutional VAE (AutoencoderKL) and Computational Cost Reduction

4.1 Meaning of "Fully Convolutional": Output Size can be Controlled by Input Size

Fully-convolutional refers to the property where all layers consist of **local operations** such as convolution or upsampling, and the network **can accept inputs of variable dimensions (sizes), and the output dimensions (size) are automatically and uniquely determined** according to the input dimensions (see §5). This property allows for the generation of images at arbitrary resolutions simply by changing the number of tiles in the input latent (see §5.3).

4.2 Rigorous Explanation of Computational Cost Reduction (Latent vs. Pixel)

In a typical example of the Stable Diffusion v1 series, the latent space is $8\times$ downsampled from the pixel space and has 4 channels ($H/8 \times W/8 \times 4$).[1] [2] See also [4] for the theoretical background. Since the main FLOPs of convolution and self-attention are roughly proportional to the number of spatial locations, the **spatial cost** of a latent-space UNet is approximately $1/8^2 = 1/64$ compared to a pixel-space UNet (channel differences are considered separately).

(Downsampling Factor and FLOPs Scaling) When performing computations in a latent space, which is s-times downsampled from the input, using the same architecture (compa-

[1] See CompVis Stable Diffusion v1 README. `https://github.com/CompVis/stable-diffusion` (Accessed: 2025-10-26)

[2] See the inference configuration `v1-inference.yaml` for Stable Diffusion v1.5. `https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5/blob/main/v1-inference.yaml` (Accessed: 2025-10-26)

rable scale of kernels, layers, and channels), the main FLOPs of convolutional operations decrease approximately in proportion to $1/s^2$. Therefore, if $s = 8$, the reduction is $\approx 1/64$.
(Derivation) The FLOPs for one 2D convolution layer are roughly proportional to

$$H_{\text{out}} W_{\text{out}} \cdot K_H K_W \cdot C_{\text{in}} C_{\text{out}} \tag{22}$$

. If we only consider the spatial size change $H \mapsto H/s$ and $W \mapsto W/s$, $H_{\text{out}} W_{\text{out}}$ becomes $1/s^2$ times smaller, and this proportionality factor remains even when stacking layers.

5 Convolutional Operations and the Significance of a Natural Image Decoder Being a Fully-Convolutional Neural Network

In the task of natural image generation, while the size of the output image is often predetermined, it can vary from one instance to another. It is inefficient to consider completely different neural networks for multiple output image sizes and to hold parameter vector values for each. Therefore, it is desirable to have a single parameter vector, which, when given the size of the output image, defines a single function. In this course, we will refer to this property as being **"variable-length input/output"**. A fully-convolutional neural network is precisely a neural network that possesses this variable-length input/output property. In this lecture, we will not provide a complete description of practical fully-convolutional neural networks, but we will explain the functions (layers) that serve as their components. Naturally, composing variable-length input/output functions results in another variable-length input/output function.

Remark 5.1 (Sign Convention for "Convolution" in This Lecture)**.** The term "convolution" as used in this lecture refers to **cross-correlation**, which is common in implementation. That is, for a 1D signal

$$(y \text{ by } \star)_t \coloneqq \sum_{i \in I_K} k_i \, x_{t+i}, \tag{23}$$

and for a 2D signal

$$(Y \text{ by } \star)_{u,v} \coloneqq \sum_{i \in I_{K_H}} \sum_{j \in I_{K_W}} K_{i,j} \, X_{u+i,\, v+j}, \tag{24}$$

it is defined by (with zero-padding). The standard **convolution** in mathematics and signal processing involves flipping the kernel

$$(y \text{ by } *)_t = \sum_i k_i \, x_{t-i}, \qquad (Y \text{ by } *)_{u,v} = \sum_{i,j} K_{i,j} \, X_{u-i,\, v-j} \tag{25}$$

thus, the sign of the indices is reversed. The symbol $\star$ in this lecture is used to explicitly denote **cross-correlation** as aligned with implementation.

Remark 5.2 (On the Term "n-dimensional Convolution")**.** Conventionally, the term **n-dimensional convolution** is used, but strictly speaking, it often refers to the **order** of the tensor, i.e., the number of spatial axes. In these notes, we follow convention and use the terms 1D/2D.

5.1 1D Convolution: Separating the Mathematical Definition and Layer Definition

5.1.1 Goal: Variable-Length Parametric Function for One-Dimensional Tensors

Definition 5.1 (Variable-Length Parametric Function (1D))**.** A variable-length parametric function $f_{(\cdot)}$ that takes **one-dimensional tensors (column vectors)** as input/output is a tuple consisting of the following four elements:

- **Parameter dimension** $d_{\mathrm{param}} \in \mathbb{Z}_{>0}$.
- **Set of acceptable input dimensions** $\mathcal{D}_{\mathrm{in}} \subset \mathbb{Z}_{\geq 0}$.
- **Input/output length mapping** InLen2OutLen : $\mathcal{D}_{\mathrm{in}} \to \mathbb{Z}_{\geq 0}$.
- A family of maps $\left\{ f_{\boldsymbol{\theta}}^{(L_{\mathrm{in}})} \right\}_{\boldsymbol{\theta} \in \mathbb{R}^{d_{\mathrm{param}}}}$ defined for each $L_{\mathrm{in}} \in \mathcal{D}_{\mathrm{in}}$, such that:

$$f_{\boldsymbol{\theta}}^{(L_{\mathrm{in}})} \ : \ \mathbb{R}^{L_{\mathrm{in}}} \to \mathbb{R}^{\mathsf{InLen2OutLen}(L_{\mathrm{in}})}. \tag{26}$$

At this time, the **variable-length parametric function** $f_{(\cdot)}$ itself is defined as:

$$\forall\, \boldsymbol{\theta} \in \mathbb{R}^{d_{\mathrm{param}}},\ \forall\, \boldsymbol{x} \in \bigcup_{L \in \mathcal{D}_{\mathrm{in}}} \mathbb{R}^{L} : \quad f_{\boldsymbol{\theta}}(\boldsymbol{x}) \ \coloneqq \ f_{\boldsymbol{\theta}}^{(\dim(\boldsymbol{x}))}(\boldsymbol{x}) \ \in \ \mathbb{R}^{\mathsf{InLen2OutLen}(\dim(\boldsymbol{x}))} \tag{27}$$

5.1.2 Mathematical Definition of the 1D $\star$ Operator

Definition 5.2 (1D $\star$ Operator (1D cross-correlation; basic form))**.** Let $\boldsymbol{x} \in \mathbb{R}^{L}$ be a sequence and $K \in \mathbb{Z}_{>0}$ be the kernel length. The index set I_K is given by

$$I_K \coloneqq \begin{cases} \{-r, -r+1, \ldots, 0, \ldots, +r\}, & K = 2r+1 \text{ (odd)}, \\ \{-r+1, -r+2, \ldots, +r\}, & K = 2r \text{ (even)} \end{cases} \tag{28}$$

and the kernel $\boldsymbol{k} = (k_i)_{i \in I_K}$ is arranged in this order (e.g., k_{-1}, k_0, k_{+1} if $K = 3$, and k_0, k_{+1} if $K = 2$). We set zero-padding $x_j = 0$ for $j \notin [0, L-1]_{\mathbb{Z}}$. The $\star$ **operation** $\boldsymbol{y} = \boldsymbol{k} \ \star \ \boldsymbol{x}$ is defined

by:

$$y_t \coloneqq \sum_{i \in I_K} k_i \, x_{t+i}, \qquad t \in \mathbb{Z} \tag{29}$$

As the minimal finite interval excluding t where $y_t = 0$, the **output length** is $L_{\text{out}} = L + K - 1$, and we consider $t \in [0, L_{\text{out}} - 1]_{\mathbb{Z}}$.

Example 5.1 (Complete Numerical Example (1D, basic $\star$ operation))**.** Let the input be $\boldsymbol{x} = (x_0, x_1, x_2, x_3) = (1, 2, 0, -1)$ with $L = 4$, and the kernel be $(k_{-1}, k_0, k_{+1}) = (2, -1, 3)$ with $K = 3$, $I_3 = \{-1, 0, +1\}$. We set $x_{-1} = x_4 = x_5 = \cdots = 0$. From Eq. (29):

$$y_0 = 2 \cdot x_{-1} + (-1) \cdot x_0 + 3 \cdot x_1 = 0 - 1 + 6 = 5, \tag{30}$$
$$y_1 = 2 \cdot x_0 + (-1) \cdot x_1 + 3 \cdot x_2 = 2 - 2 + 0 = 0, \tag{31}$$
$$y_2 = 2 \cdot x_1 + (-1) \cdot x_2 + 3 \cdot x_3 = 4 - 0 - 3 = 1, \tag{32}$$
$$y_3 = 2 \cdot x_2 + (-1) \cdot x_3 + 3 \cdot x_4 = 0 + 1 + 0 = 1, \tag{33}$$
$$y_4 = 2 \cdot x_3 + (-1) \cdot x_4 + 3 \cdot x_5 = -2 - 0 + 0 = -2, \tag{34}$$
$$y_5 = 2 \cdot x_4 + (-1) \cdot x_5 + 3 \cdot x_6 = 0. \tag{35}$$

Thus, $L_{\text{out}} = 6$ and $\boldsymbol{y} = (5, 0, 1, 1, -2, 0)$.

Exercise 5.1 (Complete Numerical Example (1D, basic $\star$ operation 2))**.** Let the input be $\boldsymbol{x} = (3, 0, -2, 1, 4)$ with $L = 5$, and the kernel be $(k_0, k_{+1}) = (1, -2)$ with $K = 2$, $I_2 = \{0, +1\}$ (where $x_j = 0$ for $j \notin [0, 4]_{\mathbb{Z}}$). From Eq. (29), $L_{\text{out}} = 5 + 2 - 1 = 6$, and for each $t = 0, \ldots, 5$:

$$y_0 = 1 \cdot x_0 + (-2) \cdot x_1 = 3 - 0 = 3, \tag{36}$$
$$y_1 = 1 \cdot x_1 + (-2) \cdot x_2 = 0 - (-2) \cdot 2 = 4, \tag{37}$$
$$y_2 = 1 \cdot x_2 + (-2) \cdot x_3 = -2 - 2 = -4, \tag{38}$$
$$y_3 = 1 \cdot x_3 + (-2) \cdot x_4 = 1 - 8 = -7, \tag{39}$$
$$y_4 = 1 \cdot x_4 + (-2) \cdot x_5 = 4 - 0 = 4, \tag{40}$$
$$y_5 = 1 \cdot x_5 + (-2) \cdot x_6 = 0. \tag{41}$$

Therefore, $\boldsymbol{y} = (3, 4, -4, -7, 4, 0)$.

Answer. We apply Eq. (29) for each time t. Due to zero-padding, $x_{-1} = x_5 = 0$.

$$
\begin{aligned}
t = 0 &: y_0 = k_0 x_0 + k_{+1} x_1 = 3 + (-2) \cdot 0 = 3, \\
t = 1 &: y_1 = k_0 x_1 + k_{+1} x_2 = 0 + (-2) \cdot (-2) = 4, \\
t = 2 &: y_2 = k_0 x_2 + k_{+1} x_3 = -2 + (-2) \cdot 1 = -4, \\
t = 3 &: y_3 = k_0 x_3 + k_{+1} x_4 = 1 + (-2) \cdot 4 = -7, \\
t = 4 &: y_4 = k_0 x_4 + k_{+1} x_5 = 4 + (-2) \cdot 0 = 4, \\
t = 5 &: y_5 = k_0 x_5 + k_{+1} x_6 = 0.
\end{aligned}
$$

Thus, $\boldsymbol{y} = (3, 4, -4, -7, 4, 0)$.

5.1.3 1D Discrete Convolutional Layer (Definition by cross-correlation ⋆)

Definition 5.3 (1D Convolutional Layer (with stride/padding/dilation, multi-channel, activation; variable-length parametric function by ⋆))**.** This layer is defined as a variable-length parametric function per Definition 5.1.

Parameters: We are given stride $S \in \mathbb{Z}_{>0}$, padding $P \in \mathbb{Z}_{\geq 0}$, dilation $D \in \mathbb{Z}_{>0}$, input/output channel counts $m, n \in \mathbb{Z}_{>0}$, and kernel length $K \in \mathbb{Z}_{>0}$. The index set I_K is as in (28). The learnable parameters are:

$$
\boldsymbol{k}^{[c,r]} = (k_i^{[c,r]})_{i \in I_K}, \qquad b^{[c]} \in \mathbb{R} \tag{42}
$$

Acceptable input lengths: $\mathcal{D}_{\text{in}} = \mathbb{Z}_{\geq 0}$.

Input/output length mapping: For an input length $L \in \mathbb{Z}_{\geq 0}$,

$$
\mathsf{InLen2OutLen}(L) \;:=\; \left\lfloor \frac{L + 2P - D\,(K-1) - 1}{S} \right\rfloor + 1 \;\in\; \mathbb{Z}_{\geq 0}. \tag{43}
$$

Definition of map family: For any $L \in \mathcal{D}_{\text{in}}$, let the input be $\boldsymbol{X} \in \mathbb{R}^{m \times L}$ and the output be $\boldsymbol{Y} \in \mathbb{R}^{n \times \mathsf{InLen2OutLen}(L)}$. We denote the sequence for each input channel as $\boldsymbol{x}^{[r]} \in \mathbb{R}^{L}$ $(r = 1, \ldots, m)$ and for each output channel as $\boldsymbol{y}^{[c]} \in \mathbb{R}^{\mathsf{InLen2OutLen}(L)}$ $(c = 1, \ldots, n)$. We define the zero-padded vector $\tilde{\boldsymbol{x}}^{[r]} \in \mathbb{R}^{L+2P}$ as:

$$
\tilde{x}_t^{[r]} = \begin{cases} x_{t-P}^{[r]}, & t \in [0, L + 2P - 1]_{\mathbb{Z}}, \\ 0, & \text{otherwise} \end{cases} \tag{44}
$$

At this time, for $t \in [0, \mathrm{InLen2OutLen}(L) - 1]_{\mathbb{Z}}$:

$$y_t^{[c]} \coloneqq \sum_{r=1}^{m} \sum_{i \in I_K} k_i^{[c,r]} \, \tilde{x}_{tS+iD}^{[r]} \; + \; b^{[c]} \quad (c = 1, \ldots, n), \tag{45}$$

$$\widehat{\boldsymbol{Y}} \coloneqq \mathrm{Act}(\boldsymbol{Y}) \quad \text{(Activation Act is applied element-wise).} \tag{46}$$

From the above, we give $\{f_{\boldsymbol{\theta}}^{(L)}\}$ as $f_{\boldsymbol{\theta}}^{(L)}(\boldsymbol{X}) = \widehat{\boldsymbol{Y}}$ ($\boldsymbol{\theta}$ is the concatenation of all coefficients in (42)).

Below are diagrams of the basic form **without S, P, D** and examples including S/P/D.

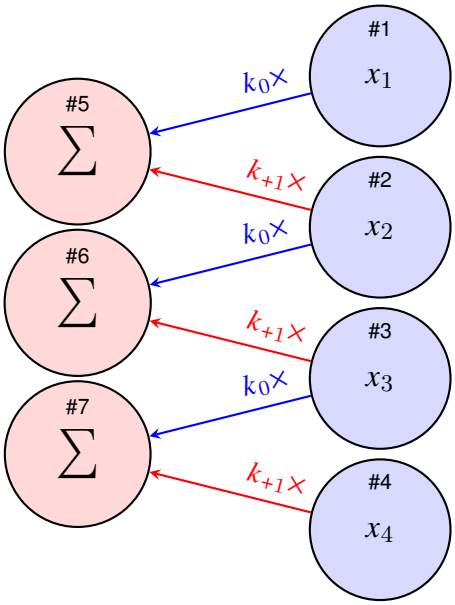

Figure 3: 1D 1ch→1ch, $L = 4, K = 2, S = 1$ (by Eq. (43), $L_{\mathrm{out}} = 3$). Edges of the same color represent shared parameters ($\star$ operation).

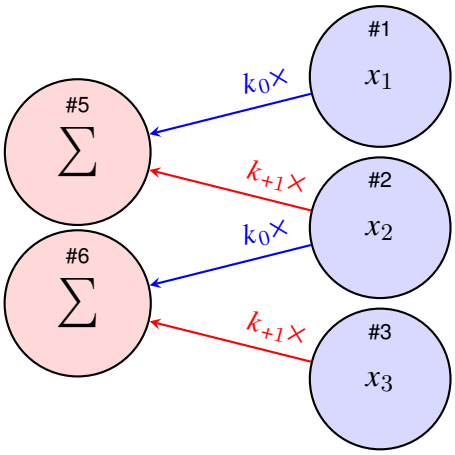

Figure 4: 1D 1ch→1ch, $L = 3, K = 2, S = 1$ ($\star$ operation). $L_{\mathrm{out}} = 2$.

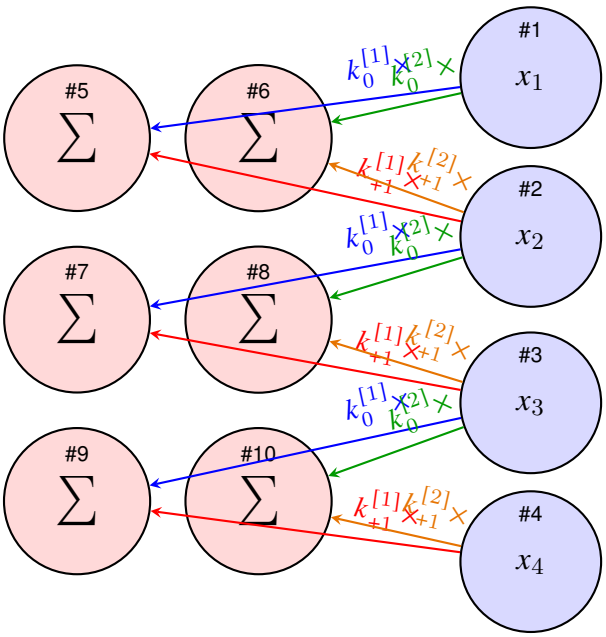

Figure 5: 1D 1ch→2ch, $L = 4, K = 2, S = 1$ ($\star$ operation).

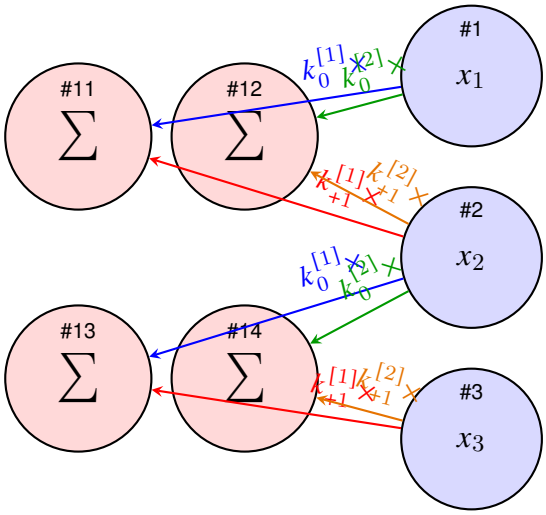

Figure 6: 1D 1ch→2ch, $L = 3, K = 2, S = 1$ ($\star$ operation).

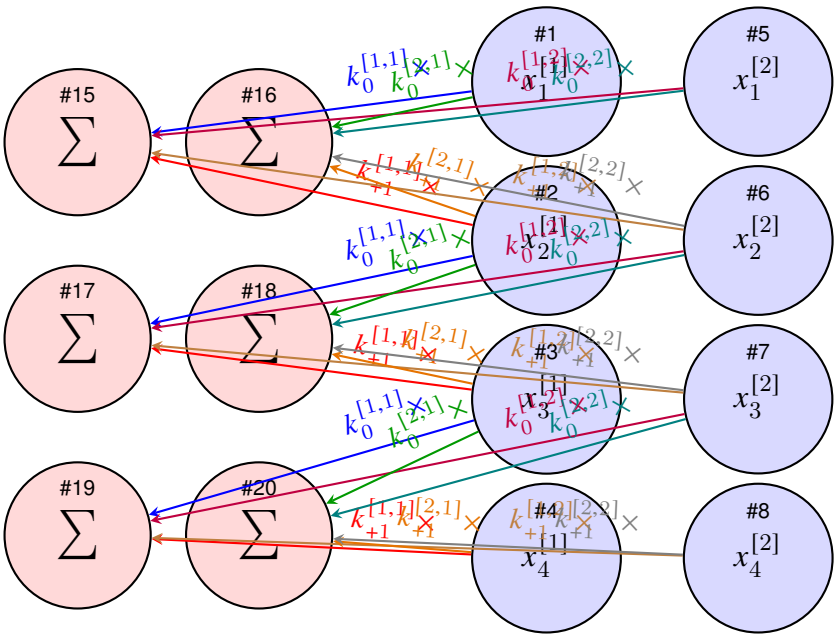

Figure 7: 1D 2ch→2ch, $L = 4, K = 2, S = 1$ ($\star$ operation; missing edges completed).

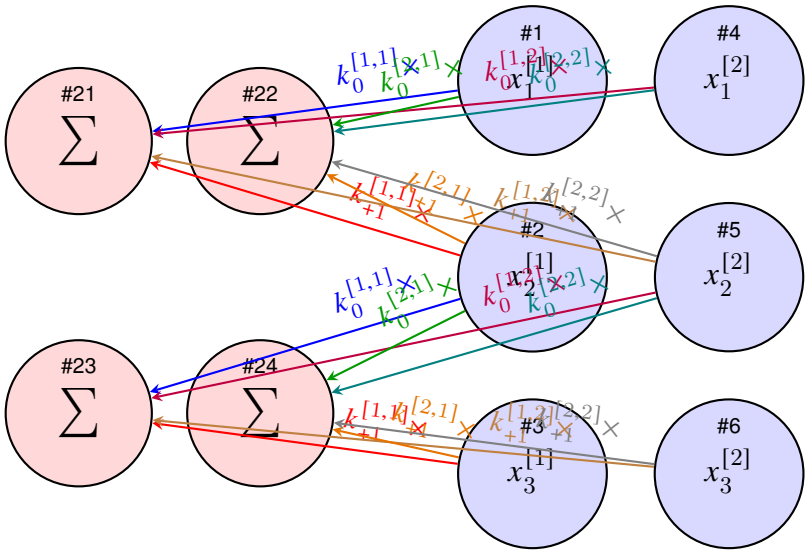

Figure 8: 1D 2ch→2ch, $L = 3, K = 2, S = 1$ ($\star$ operation).

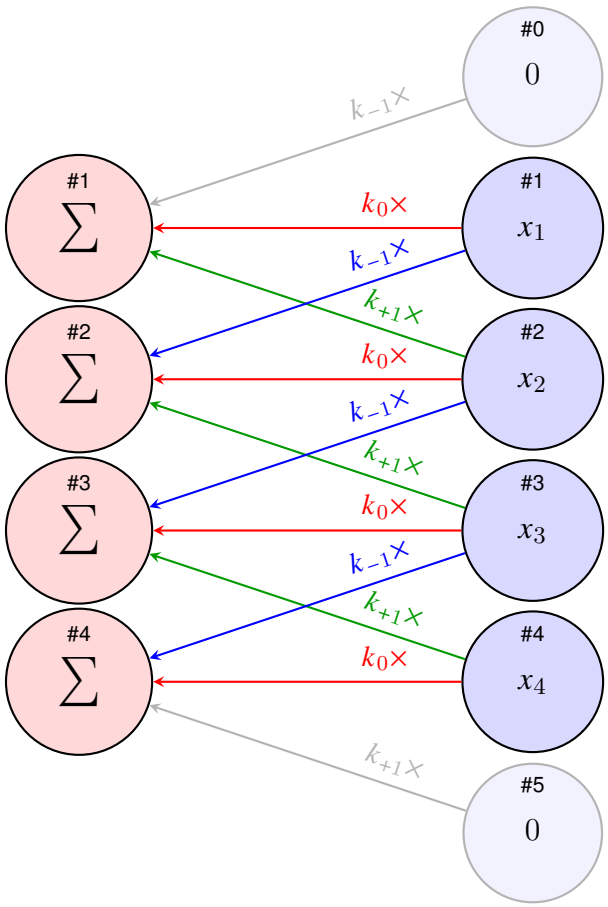

Figure 9: 1D 1ch→1ch, $L = 4, K = 3, S = 1, P = 1, D = 1$ (⋆ operation; $L_{\text{out}} = 4$).

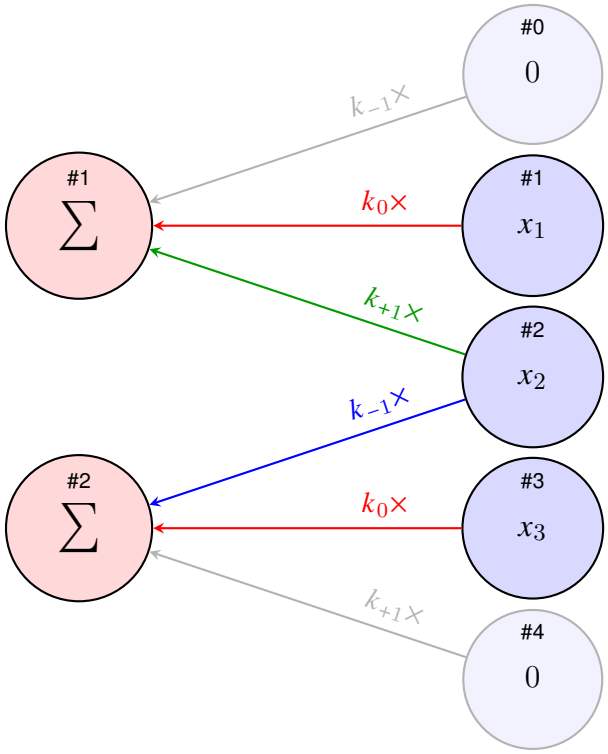

Figure 10: 1D 1ch→1ch, $L = 3, K = 3, S = 2, P = 1, D = 1$ (⋆ operation; $L_{\text{out}} = 2$).

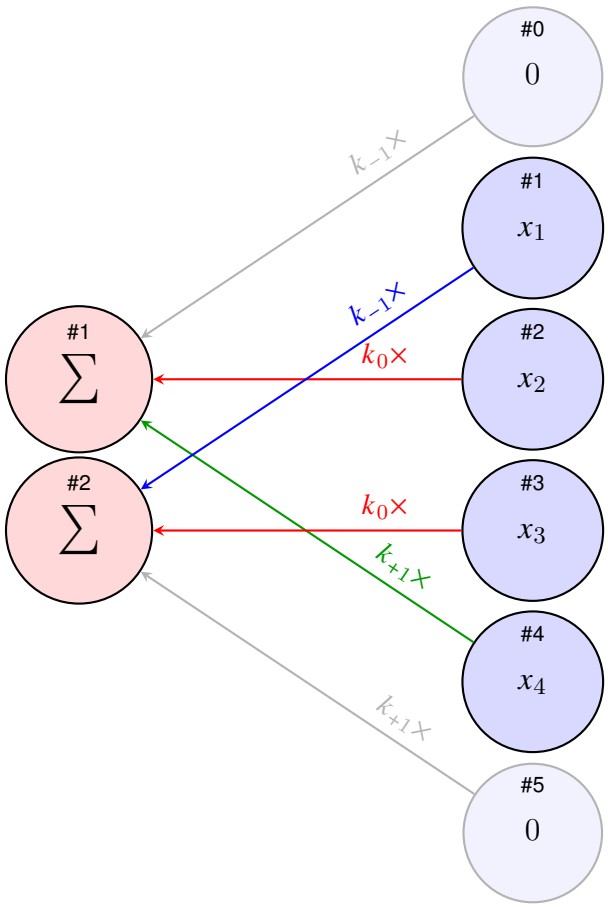

Figure 11: 1D 1ch→1ch, $L = 4, K = 3, S = 1, P = 1, D = 2$ ($\star$ operation; $L_{\text{out}} = 2$).

5.1.4 1D Nearest-Neighbor Upscaler (NN Upsample; Variable-Length)

Definition 5.4 (1D Nearest-Neighbor Upscaler (Variable-Length))**.** Given an upscale factor $s \in \mathbb{Z}_{\geq 2}$. Parameters are empty ($d_{\text{param}} = 0$), the acceptable set is $\mathcal{D}_{\text{in}} = \mathbb{Z}_{\geq 0}$, and the input/output length mapping is defined by:

$$\mathsf{InLen2OutLen}(L) \coloneqq sL \tag{47}$$

For any L and input $\boldsymbol{x} \in \mathbb{R}^L$, it is defined by:

$$\left(f^{(L)}(\boldsymbol{x})\right)_t \;\coloneqq\; x_{\lfloor t/s \rfloor}, \qquad t \in [0, sL-1]_{\mathbb{Z}} \tag{48}$$

(where $\lfloor \cdot \rfloor$ is the floor function).

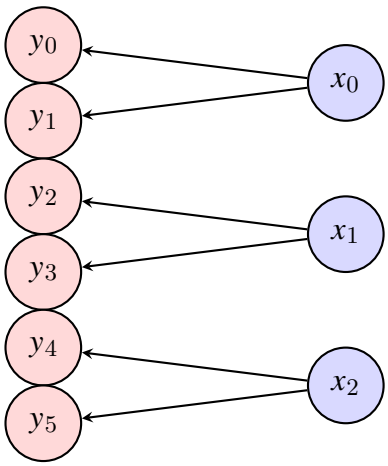

Figure 12: 1D Nearest-Neighbor Upscaler (example with $s = 2$, $L = 3$).

5.1.5 1D Group Normalization (Variable-Length)

Definition 5.5 (Group Normalization (1D; Variable-Length))**.** Given the number of channels $c \in \mathbb{Z}_{>0}$, and the number of groups g such that $g \mid c$ (where "group" here refers to a simple subset, not a group in the algebraic sense). The input is $\boldsymbol{F} \in \mathbb{R}^{c\times L}$ (each row is $\boldsymbol{f}^{[i]} \in \mathbb{R}^{L}$). For each group $G_k = \{(k-1)\frac{c}{g} + 1, \ldots, k\frac{c}{g}\}$:

$$\mu_k \coloneqq \frac{1}{|G_k|\, L} \sum_{i\in G_k} \sum_{t=1}^{L} f_t^{[i]}, \qquad \sigma_k^2 \coloneqq \frac{1}{|G_k|\, L} \sum_{i\in G_k} \sum_{t=1}^{L} \big(f_t^{[i]} - \mu_k\big)^2. \tag{49}$$

Using learnable coefficients $\gamma_i, \beta_i \in \mathbb{R}$ ($i = 1, \ldots, c$) and $\varepsilon > 0$:

$$\widehat{f}_t^{[i]} \coloneqq \gamma_i \frac{f_t^{[i]} - \mu_{k(i)}}{\sqrt{\sigma_{k(i)}^2 + \varepsilon}} + \beta_i, \qquad i \in G_{k(i)},\ t = 1, \ldots, L. \tag{50}$$

The above has a number of learnable parameters independent of L, and the output dimension of the function is determined depending on L, thus it provides a variable-length parametric function within the framework of Definition 5.1 (the output length is the same as the input length).

5.2 2D Convolution: Separating the Mathematical Definition and Layer Definition

5.2.1 Goal: Variable-Length Parametric Function for Two-Dimensional Tensors

Definition 5.6 (Variable-Length Parametric Function (2D))**.** A variable-length parametric function $F_{(\cdot)}$ that takes **two-dimensional tensors** as input/output consists of $d_{\text{param}} \in \mathbb{Z}_{>0}$, acceptable sets $\mathcal{H}, \mathcal{W} \subset \mathbb{Z}_{\geq 0}$, an input/output mapping $\text{HW2HW}_{\text{out}} : \mathcal{H} \times \mathcal{W} \to \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0}$, and a family for each (H, W):

$$F_{\boldsymbol{\theta}}^{(H,W)} : \mathbb{R}^{H \times W} \to \mathbb{R}^{H_{\text{out}}(H,W) \times W_{\text{out}}(H,W)} \tag{51}$$

The function itself is defined as $F_{\boldsymbol{\theta}}(\boldsymbol{X}) \coloneqq F_{\boldsymbol{\theta}}^{(\dim_1(\boldsymbol{X}), \dim_2(\boldsymbol{X}))}(\boldsymbol{X})$.

5.2.2 Mathematical Definition of the 2D $\star$ Operator

Definition 5.7 (2D $\star$ Operator (2D cross-correlation; basic form))**.** Let the input be $\boldsymbol{X} \in \mathbb{R}^{H \times W}$, and the filter sizes be $K_H, K_W \in \mathbb{Z}_{>0}$. The index sets

$$I_{K_H},\ I_{K_W} \text{ are defined similarly to (28).} \tag{52}$$

Let the kernel be $\boldsymbol{K} = (K_{i,j})_{i \in I_{K_H},\, j \in I_{K_W}}$. We set zero-padding $X_{p,q} = 0$ for $(p, q) \notin [0, H-1]_{\mathbb{Z}} \times [0, W-1]_{\mathbb{Z}}$, and define $\boldsymbol{Y} = \boldsymbol{K} \star \boldsymbol{X}$ as:

$$Y_{u,v} \coloneqq \sum_{i \in I_{K_H}} \sum_{j \in I_{K_W}} K_{i,j}\, X_{u+i,\, v+j}, \qquad (u, v) \in \mathbb{Z}^2 \tag{53}$$

Restricting to the minimal rectangular region that can be non-zero, $H_{\text{out}} = H + K_H - 1$, $W_{\text{out}} = W + K_W - 1$, and we consider $u \in [0, H_{\text{out}} - 1]_{\mathbb{Z}}$, $v \in [0, W_{\text{out}} - 1]_{\mathbb{Z}}$.

Example 5.2 (Complete Numerical Example (2D, basic $\star$ operation))**.**

$$\boldsymbol{X} = \begin{bmatrix} 1 & 0 \\ 2 & -1 \end{bmatrix} \in \mathbb{R}^{2 \times 2}, \quad \boldsymbol{K} = \begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix} \in \mathbb{R}^{2 \times 2},$$

Here, we let $I_{K_H} = I_{K_W} = \{0, +1\}$. From Eq. (53), $H_{\text{out}} = 3$, $W_{\text{out}} = 3$, and:

$$\boldsymbol{Y} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Exercise 5.2 (Complete Numerical Example (2D, basic $\star$ operation 2))**.**

$$\boldsymbol{X} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 3 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 3}, \quad \boldsymbol{K} = \begin{bmatrix} 2 & 0 \\ 1 & -1 \end{bmatrix} \in \mathbb{R}^{2\times 2},$$

Here, we let $I_{K_H} = I_{K_W} = \{0, +1\}$. $H_{\text{out}} = 3$, $W_{\text{out}} = 4$, and:

$$\boldsymbol{Y} = \begin{bmatrix} -4 & 4 & 5 & 0 \\ -2 & 6 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Answer. For each (u, v), we evaluate by substituting directly into Eq. (53). For example, $Y_{0,2}$ is:

$$Y_{0,2} = \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} K_{i,j}\, X_{0+i,\,2+j} = K_{0,0}X_{0,2}+K_{0,1}X_{0,3}+K_{1,0}X_{1,2}+K_{1,1}X_{1,3} = 2\cdot 2+0\cdot 0+1\cdot 1+(-1)\cdot 0 = 5$$

The other values in the table are obtained by similar calculations.

5.2.3 2D Discrete Convolutional Layer (Definition by cross-correlation $\star$)

Definition 5.8 (2D Convolutional Layer (with stride/padding/dilation, multi-channel, activation; variable-length parametric function by $\star$))**.** This layer is defined in the sense of Definition 5.6. Let the input be $\underline{\boldsymbol{X}} \in \mathbb{R}^{m\times H\times W}$ ($\boldsymbol{X}^{[r]}$ is the feature map of input channel r), and the output be $\underline{\boldsymbol{Y}} \in \mathbb{R}^{n\times H_{\text{out}}\times W_{\text{out}}}$. We are given $S_H, S_W \in \mathbb{Z}_{>0}$, $P_H, P_W \in \mathbb{Z}_{\geq 0}$, $D_H, D_W \in \mathbb{Z}_{>0}$. The learnable parameters are:

$$\boldsymbol{K}^{[c,r]} = (K_{i,j}^{[c,r]})_{i\in I_{K_H},\, j\in I_{K_W}}, \qquad b^{[c]} \in \mathbb{R}. \tag{54}$$

We define the zero-padding $\tilde{X}_{p,q}^{[r]}$ as:

$$\tilde{X}_{p,q}^{[r]} = \begin{cases} X_{p-P_H,\,q-P_W}^{[r]}, & p \in [0, H + 2P_H - 1]_{\mathbb{Z}},\ q \in [0, W + 2P_W - 1]_{\mathbb{Z}}, \\ 0, & \text{otherwise} \end{cases} \tag{55}$$

The output spatial size is:

$$H_{\text{out}} \coloneqq \left\lfloor \frac{H + 2P_H - D_H\,(K_H - 1) - 1}{S_H} \right\rfloor + 1, \tag{56}$$

$$W_{\text{out}} \coloneqq \left\lfloor \frac{W + 2P_W - D_W\,(K_W - 1) - 1}{S_W} \right\rfloor + 1. \tag{57}$$

For each c and (p, q):

$$Y^{[c]}_{p,q} := \sum_{r=1}^{m} \sum_{i \in I_{K_H}} \sum_{j \in I_{K_W}} K^{[c,r]}_{i,j} \, \tilde{X}^{[r]}_{pS_H+iD_H, qS_W+jD_W} + b^{[c]}, \tag{58}$$

$$\underline{\widehat{Y}} := \mathrm{Act}(\underline{Y}) \, . \tag{59}$$

Using $\star$ notation, $\sum_{i,j} K^{[c,r]}_{i,j} \, \tilde{X}^{[r]}_{pS_H+iD_H, qS_W+jD_W}$ can be expressed as a subsampling of a dilated cross-correlation.

Examples are shown below (Fig. 13, 14).

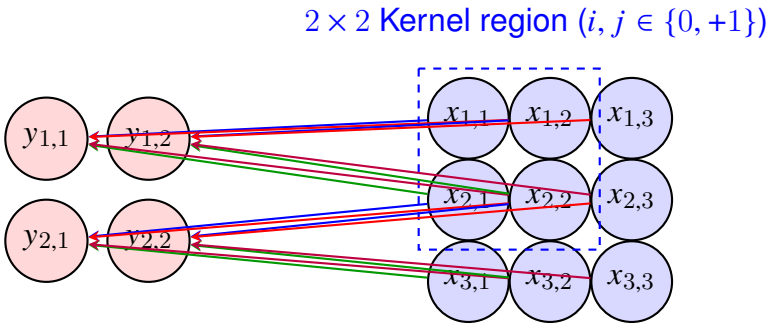

Figure 13: Example of 2D 1ch→1ch (3×3 input, 2×2 kernel, stride 1; $\star$ operation). $H_{\mathrm{out}} = W_{\mathrm{out}} = 2$ (Eq. (56),(57)).

For input $\underline{X} \in \mathbb{R}^{m \times H \times W}$ and output $\underline{Y} \in \mathbb{R}^{n \times H_{\mathrm{out}} \times W_{\mathrm{out}}}$,

$$Y^{[c]}_{p,q} = \sum_{r=1}^{m} \sum_{i \in I_{K_H}} \sum_{j \in I_{K_W}} K^{[c,r]}_{i,j} \, \tilde{X}^{[r]}_{pS_H+iD_H, qS_W+jD_W} + b^{[c]}. \tag{60}$$

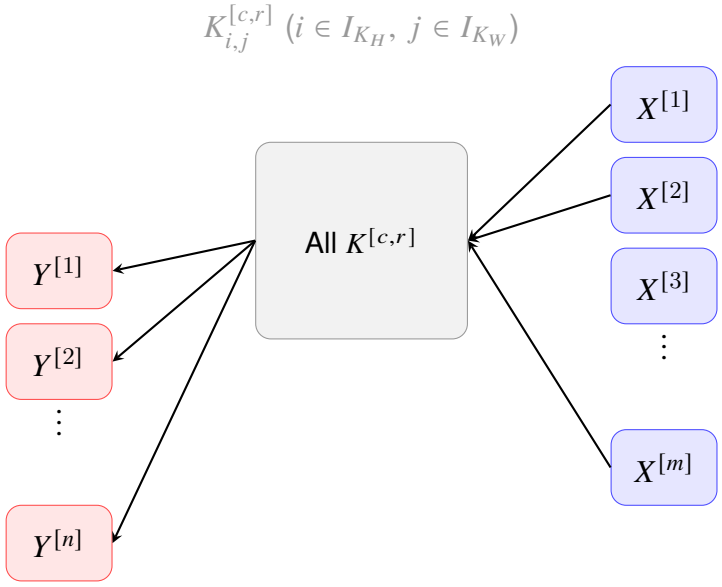

Figure 14: Schematic of 2D $m$ch→ $n$ch (Eq. (56) – (58), (60); $\star$ operation).

5.2.4 2D Nearest-Neighbor Upscaler (Variable-Length)

Definition 5.9 (2D Nearest-Neighbor Upscaler (Variable-Length))**.** Given an upscale factor $s \in \mathbb{Z}_{\geq 2}$. $d_{\mathrm{param}} = 0$, the acceptable sets are $\mathcal{H} = \mathcal{W} = \mathbb{Z}_{\geq 0}$, and the input/output mapping is $(H, W) \mapsto (sH, sW)$. For an input $\boldsymbol{X} \in \mathbb{R}^{H \times W}$,

$$\left(F^{(H,W)}(\boldsymbol{X})\right)_{sp+a,\ sw+b} \coloneqq X_{p,q}, \quad a, b \in [0, s-1]_{\mathbb{Z}},\ \ (p, q) \in [0, H-1]_{\mathbb{Z}} \times [0, W-1]_{\mathbb{Z}}. \tag{61}$$

5.2.5 2D Group Normalization (Variable-Length)

Definition 5.10 (Group Normalization (2D; Variable-Length))**.** With the same settings as Definition 5.5, the **layer output is defined** according to any non-negative integers H, W. That is, when giving $\widehat{\underline{\boldsymbol{F}}}$ for $\underline{\boldsymbol{F}} \in \mathbb{R}^{c \times H \times W}$ using Eq. (49) and (50), we define a family of variable-length parametric functions $\{\widehat{\underline{\boldsymbol{F}}}\}$ with $d_{\mathrm{param}} = 2c$ (the output spatial size is the same as the input, (H, W)).

5.3 Fully-Convolutional Networks and the Significance of Their Use

A **Fully-Convolutional Network (FCN)** is (though we do not give a strict definition) a neural network that is, as a whole, **variable-length input/output**. It is constructed by composing variable-length layers, such as the convolutional layers, upscalers, and Group Normalization introduced so far, as well as element-wise operations (like simple addition). Because an FCN is variable-length input/output, if a natural image decoder is given as an FCN, one only needs to obtain a single appropriate parameter vector; to output an image of a desired size, one just needs to choose the input size appropriately. (Of course, the quality of the output image is also usually a concern, in which case it is necessary to properly determine not only the size but also the values of the input.)

6 Significance of the Natural Image Decoder Being an Autoencoder

In this section, we will carefully describe the significance of the natural image decoder being structured as the **decoder** of an **autoencoder**.

6.1 Motivation and Structure

The fact that the natural image decoder is an autoencoder implies that a **corresponding encoder** exists, and a map from the high-dimensional natural image space $\mathcal{I}$ to the low-dimensional latent space $\mathcal{Z}$ coexists. That is, we consider a pair of an encoder $\mathrm{Enc}_{\boldsymbol{\eta}} : \mathcal{I} \to \mathcal{Z}$ and a decoder $\mathrm{Dec}_{\boldsymbol{\gamma}} : \mathcal{Z} \to \mathcal{I}$, and train the parameters $(\boldsymbol{\eta}, \boldsymbol{\gamma})$ such that for frequently

occurring data $\underline{\boldsymbol{X}} \sim p_{\mathrm{data}}$,

$$\underline{\boldsymbol{X}} \approx \mathrm{Dec}_{\boldsymbol{\gamma}}(\mathrm{Enc}_{\boldsymbol{\eta}}(\underline{\boldsymbol{X}})) \tag{62}$$

holds. Measures of proximity used include L^1/L^2 loss, perceptual loss, etc. The above equation (62) sets the goal that the encoder and decoder are, **in a sense, inverse maps** of each other.

6.2 Practical Significance as Compatibility

The encoder performs **dimensionality reduction**, and in general, its right inverse is not unique. Therefore, multiple different decoders $\mathrm{Dec}_{\boldsymbol{\gamma}_1}$ and $\mathrm{Dec}_{\boldsymbol{\gamma}_2}$ can be **compatible** (satisfying the performance of Eq. (62)) with the same encoder $\mathrm{Enc}_{\boldsymbol{\eta}}$. In application, by fixing the encoder and retraining a new decoder, one can **replace the decoder while maintaining the latent representation specification**. On the other hand, if the encoder is not fixed and only a map from low to high dimensions is trained, a **degree of freedom for coordinate replacement** in the target space remains, and decoders that achieve a similar image set but are **not mutually compatible as maps** may be trained. Thus, the autoencoder framework ensures **interoperability**.

7 Significance of the Natural Image Decoder Being a ”Variational” Autoencoder

As a natural image decoder used in practical pipelines, a variational autoencoder (VAE) [3] is used, which is among autoencoders, trained using an objective function that includes regularization and noise. In this section, we introduce the most basic objective function of a VAE and explain what properties a VAE trained with it is expected to have.

Note that the objective functions actually used in the public training process are **far more complex than what is introduced here**, and will be explained in the appendix. Please be aware that the explanation here is merely a description of the basic motivation.

7.1 VAE Objective Function Using an Uncorrelated Gaussian Distribution

In this section, we fully describe the objective function using only the **uncorrelated Gaussian assumption**, **linear algebra**, and **random number sampling**, which are common in implementations.

Definition 7.1 (VAE Objective Function using an Uncorrelated Gaussian Distribution)**.** For

each input image $\underline{\boldsymbol{X}} \in \mathcal{I}$, we define an encoder that outputs a mean vector

$$\mathrm{MeanEnc}_{\boldsymbol{\eta}_{\mathrm{mean}}} : \mathcal{I} \to \mathbb{R}^d, \qquad \boldsymbol{\mu}(\underline{\boldsymbol{X}}) \coloneqq \mathrm{MeanEnc}_{\boldsymbol{\eta}_{\mathrm{mean}}}(\underline{\boldsymbol{X}}), \tag{63}$$

and an encoder that outputs the standard deviation for each component

$$\mathrm{SDEnc}_{\boldsymbol{\eta}_{\mathrm{SD}}} : \mathcal{I} \to \mathbb{R}^d_{>0}, \qquad \boldsymbol{\sigma}(\underline{\boldsymbol{X}}) \coloneqq \mathrm{SDEnc}_{\boldsymbol{\eta}_{\mathrm{SD}}}(\underline{\boldsymbol{X}}) \tag{64}$$

(σ is a positive real number for each element). The entire set of parameters for the VAE encoder is then

$$\boldsymbol{\eta} \coloneqq (\boldsymbol{\eta}_{\mathrm{mean}}, \boldsymbol{\eta}_{\mathrm{SD}}) \tag{65}$$

. We **sample** a standard normal random vector $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and generate the latent vector by

$$z_{\boldsymbol{\epsilon}} \coloneqq \boldsymbol{\mu}(\underline{\boldsymbol{X}}) + \boldsymbol{\sigma}(\underline{\boldsymbol{X}}) \odot \boldsymbol{\epsilon} \tag{66}$$

($\odot$ is the element-wise product). The decoder provides

$$\hat{\boldsymbol{X}}_{\boldsymbol{\epsilon}} \coloneqq \mathrm{Dec}_{\boldsymbol{\gamma}}(z_{\boldsymbol{\epsilon}}) \tag{67}$$

. Using a reconstruction loss function $\ell : \mathcal{I} \times \mathcal{I} \to \mathbb{R}_{\geq 0}$, the objective function is

$$\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma}) \coloneqq \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \Big[\underbrace{\ell(\underline{\boldsymbol{X}}, \hat{\underline{\boldsymbol{X}}}_{\boldsymbol{\epsilon}})}_{\text{Reconstruction term}} \Big] + \beta \underbrace{\sum_{i=1}^{d} \left(\mu_i(\underline{\boldsymbol{X}})^2 + \sigma_i(\underline{\boldsymbol{X}})^2 - \log \sigma_i(\underline{\boldsymbol{X}})^2 - 1 \right)}_{\text{Regularization (concentration to origin)}} \tag{68}$$

($\beta > 0$ is a weight, and the regularization term is the closed-form KL divergence [3] written in linear algebraic terms).

Remark 7.1. In practice, ℓ is often the L^1 loss (1-norm loss). Compared to the squared error loss (L^2), L^1 linearly suppresses the contribution of outliers, thus having the property of being **robust to extreme outliers**.

Remark 7.2. Practical loss functions often include additional terms, such as those based on deep feature-based perceptual similarity (**learned perceptual image patch similarity; LPIPS**) [5], or **adversarial losses** [2] that use a discriminator trained with **Generative Adversarial Networks (GAN)**. They are **quite complex**.

Remark 7.3. Each term in Equation (68) requests the following:

- **Reconstruction term**: Ensures the encoder/decoder are **approximate inverses** of each other (a requirement for general AEs).
- **Regularization (concentration to origin)**: Encourages μ and σ to concentrate near

the origin, so the latent space is **distributed around the origin**. This makes it more likely that even if the **output of the latent generator has some error**, as long as it is near the origin, it will be decoded into a natural image.

- **Small perturbation robustness**: While suppressing the excessive growth of σ, the decoder is trained such that **small latent perturbations result in small output perturbations** (the normalization and residual structure also contribute).

These three requirements make it highly probable that an output close to the target image can be obtained, even if the output of the low-dimensional latent space image generator is not perfectly ideal.

8 Summary and Next Lecture Preview

8.1 Summary Corresponding to Learning Outcomes

- **Definition of Natural Image Decoder**: A map from low-dimensional $\mathcal{Z}$ to high-dimensional $\mathcal{I}$, whose image is a **subset** of the natural image domain.
- **Computational Cost Reduction**: Having a decoder allows for significant computational cost reduction because one only needs to **generate the low-dimensional latent**.
- **Fully Convolutional VAE**: Using a fully-convolutional VAE as a natural image decoder provides several benefits. By simply defining the input size appropriately, one can obtain an output of the desired size. Also, being an autoencoder, it always has an approximate inverse map, and multiple maps can be obtained for different purposes while maintaining compatibility. **Variational learning** provides origin-concentration and robustness (Definition 7.1).

8.2 Next Lecture Preview (Outlook)

Next time, we will detail the **reverse diffusion process**, which generates the appropriate low-resolution images to be passed to the natural image decoder (composed of a VAE, etc.).

References

[1] Kenneth J. Falconer. Fractal Geometry: Mathematical Foundations and Applications. John Wiley & Sons, Chichester, 3 edition, 2014.

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In

Advances in Neural Information Processing Systems (NeurIPS), pages 2672–2680, 2014.

[3] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[4] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10684–10695, 2022.

[5] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 586–595, 2018.

A Proof of Proposition 3.1

Proof. **Preparation (Notation)**. ω_s is the volume of the unit-radius open ball in $\mathbb{R}^s$. For normalization convenience, $\mathcal{H}^d$ matches the Lebesgue measure on $\mathbb{R}^d$ (this fact is not used in the proof itself, but is noted for consistency).
Upper bound: If $s > d$, then $\mathcal{H}^s(A) = 0$ For any $m \in \mathbb{Z}_{>0}$, we divide $A = [0,1]^d$ equally into d-dimensional cubes each of side length

$$\ell := \frac{1}{m} \tag{69}$$

. The number of these small cubes is

$$N = m^d \tag{70}$$

, and the diameter of each small cube Q is, by Euclidean geometry,

$$\operatorname{diam}(Q) = \sqrt{d}\,\ell = \frac{\sqrt{d}}{m} \tag{71}$$

. Let U_i be these N small cubes, and choose δ such that

$$\delta > \operatorname{diam}(Q) = \frac{\sqrt{d}}{m} \tag{72}$$

. Then $\{U_i\}_{i=1}^N$ is a $\delta-$ cover of A. From definition (7),

$$\mathcal{H}_\delta^s(A) \le \sum_{i=1}^{N} \omega_s \left(\frac{1}{2}\operatorname{diam} U_i\right)^s = N\,\omega_s \left(\frac{1}{2}\cdot\frac{\sqrt{d}}{m}\right)^s$$
$$= m^d\,\omega_s\left(\frac{\sqrt{d}}{2}\right)^s m^{-s} = \omega_s\left(\frac{\sqrt{d}}{2}\right)^s m^{\,d-s}. \tag{73}$$

Here, since $s > d$, $d - s < 0$, thus

$$\lim_{m\to\infty}\ \omega_s\left(\frac{\sqrt{d}}{2}\right)^s m^{\,d-s} = 0. \tag{74}$$

For any $\varepsilon > 0$, by taking a sufficiently large m, (73) becomes less than ε, so

$$\mathcal{H}_\delta^s(A) = 0 \quad \text{(for sufficiently small } \delta > 0). \tag{75}$$

By the limit (8),

$$\mathcal{H}^s(A) = \lim_{\delta\downarrow 0} \mathcal{H}_\delta^s(A) = 0. \tag{76}$$

Thus, from the definition (9), $\dim_{\mathcal{H}}(A) \le s$ follows, and since $s > d$ was arbitrary,

$$\dim_{\mathcal{H}}(A) \le d. \tag{77}$$

Lower bound: If $s < d$, then $\mathcal{H}^s(A) = \infty$ Fix an arbitrary $\delta \in (0, 1)$, and take any $\delta-$ cover $\{U_i\}_{i=1}^\infty$ of A satisfying $\operatorname{diam} U_i < \delta$. For each U_i, we can choose an open ball B_i of radius

$$r_i \coloneqq \frac{1}{2}\operatorname{diam} U_i \tag{78}$$

(i.e., such that $U_i \subset B_i$). Then, by the definition of Lebesgue measure λ_d and the unit ball volume ω_d,

$$\lambda_d(B_i) = \omega_d\, r_i^d = \omega_d \left(\frac{1}{2}\operatorname{diam} U_i\right)^d. \tag{79}$$

From the covering property $A \subset \bigcup_i U_i \subset \bigcup_i B_i$ and subadditivity,

$$1 = \lambda_d(A) \le \lambda_d\Big(\bigcup_{i=1}^{\infty} B_i\Big) \le \sum_{i=1}^{\infty} \lambda_d(B_i) = \sum_{i=1}^{\infty} \omega_d \left(\frac{1}{2}\operatorname{diam} U_i\right)^d. \tag{80}$$

Therefore,

$$\sum_{i=1}^{\infty} \left(\frac{1}{2}\operatorname{diam} U_i\right)^d \ \ge\ \frac{1}{\omega_d}. \tag{81}$$

Here, assume $s < d$. For each i, $\operatorname{diam} U_i < \delta$ and $s - d < 0$. From the monotonicity of negative

exponents ($0 < a < b$ and $\alpha < 0$ implies $a^\alpha \geq b^\alpha$),

$$\left(\frac{1}{2}\operatorname{diam} U_i\right)^{s-d} \geq \left(\frac{1}{2}\delta\right)^{s-d}. \tag{82}$$

Hence,

$$\begin{aligned}\sum_{i=1}^{\infty} \omega_s \left(\frac{1}{2}\operatorname{diam} U_i\right)^s &= \omega_s \sum_{i=1}^{\infty} \left(\frac{1}{2}\operatorname{diam} U_i\right)^d \left(\frac{1}{2}\operatorname{diam} U_i\right)^{s-d} \\ &\geq \omega_s \left(\frac{1}{2}\delta\right)^{s-d} \sum_{i=1}^{\infty} \left(\frac{1}{2}\operatorname{diam} U_i\right)^d \quad \text{(by (82))} \\ &\geq \omega_s \left(\frac{1}{2}\delta\right)^{s-d} \frac{1}{\omega_d} \quad \text{(by (81))}. \end{aligned} \tag{83}$$

The above inequality holds for "any" $\delta-$ cover, so it is also valid for the $\inf$ in the definition (7). Therefore,

$$\mathcal{H}^s_\delta(A) \geq \frac{\omega_s}{\omega_d} \left(\frac{1}{2}\delta\right)^{s-d}. \tag{84}$$

Here, since $s - d < 0$, as $\delta \downarrow 0$, $(\frac{1}{2}\delta)^{s-d} \to +\infty$, and

$$\mathcal{H}^s(A) = \lim_{\delta \downarrow 0} \mathcal{H}^s_\delta(A) = \infty. \tag{85}$$

Thus, from (9), $\dim_{\mathcal{H}}(A) \geq s$. Since $s < d$ was arbitrary,

$$\dim_{\mathcal{H}}(A) \geq d. \tag{86}$$

Conclusion Combining (77) and (86),

$$\dim_{\mathcal{H}}([0,1]^d) = d \tag{87}$$

is shown. □

B Proof of Theorem 3.1

Definition B.1 (Open ball, base, second countable, separable)**.** In a metric space (X, d), for a center $x \in X$ and radius $r > 0$, we define the open ball as $B(x, r) \coloneqq \{y \in X : d(x, y) < r\}$. A family of sets $\mathcal{B}$ is a **base** if for any open set U and any $x \in U$, there exists some $B \in \mathcal{B}$ such that $x \in B \subset U$. The space is **second countable** if it has a countable base. It is **separable** if it has a countable dense subset.

Lemma B.1 ($\mathbb{R}^n$ is second countable)**.** We equip $\mathbb{R}^n$ with the Euclidean distance $d(x, y) =$

$\|x - y\|_2$. The collection of all open balls

$$\mathcal{B} \coloneqq \{\, B(q, r) :\ q \in \mathbb{Q}^n,\ r \in \mathbb{Q}_{>0} \,\} \tag{88}$$

with rational lattice points $q \in \mathbb{Q}^n$ and rational radii $r \in \mathbb{Q}_{>0}$ is countable and forms a base for $\mathbb{R}^n$. Therefore, $\mathbb{R}^n$ is second countable.

Proof. Since $\mathbb{Q}^n$ and $\mathbb{Q}_{>0}$ are countable, $\mathcal{B}$ is countable. For any open set U and $x \in U$, by the definition of an open set, there exists $r > 0$ such that $B(x, r) \subset U$. By the countable denseness, we can choose $q \in \mathbb{Q}^n$ and $r' \in \mathbb{Q}_{>0}$ satisfying $0 < \|x - q\|_2 < \frac{r}{2}$ and $0 < r' < \frac{r}{2}$. Then by the triangle inequality, $B(q, r') \subset B(x, r) \subset U$ and $x \in B(q, r')$. Thus $\mathcal{B}$ is a base. □

Proof of Theorem 3.1. We provide a complete proof. By local Lipschitz continuity, for each $x \in \mathbb{R}^n$, there exists an open neighborhood U_x and a constant $L_x \geq 0$ such that $\|f(u) - f(v)\| \leq L_x \|u - v\|$ for $u, v \in U_x$. By Lemma B.1, $\mathbb{R}^n$ is second countable and has a countable base $\mathcal{B}$. For each x, we choose a base element $B_x \in \mathcal{B}$ such that $x \in B_x \subset U_x$ (Definition B.1). Then

$$\{B_x :\ x \in \mathbb{R}^n\} \subset \mathcal{B} \tag{89}$$

is a cover of $\mathbb{R}^n$, but since $\mathcal{B}$ is countable, we can enumerate it countably as $\{B_{x_j}\}_{j \in \mathbb{Z}_{>0}}$. For each j, let $L_j \coloneqq L_{x_j}$. Then for any $u, v \in B_{x_j}$,

$$\|f(u) - f(v)\| \leq L_j \|u - v\|. \tag{90}$$

Fix $s \geq 0$ and take any $\varepsilon > 0$. For each j, by the definition of the δ_j-outer measure (Eq. (7)) of $A \cap B_{x_j}$, we take a countable family $\{V_{j,k}\}_{k \in \mathbb{Z}_{>0}}$ satisfying $\operatorname{diam} V_{j,k} < \delta_j$ such that

$$A \cap B_{x_j} \subset \bigcup_{k=1}^{\infty} V_{j,k}, \qquad \sum_{k=1}^{\infty} (\operatorname{diam} V_{j,k})^s \leq \mathcal{H}^s_{\delta_j}(A \cap B_{x_j}) + \varepsilon 2^{-j}. \tag{91}$$

Furthermore, we may assume $V_{j,k} \subset B_{x_j}$ (since B_{x_j} is open, we can shrink $V_{j,k}$ if necessary). From Eq. (90),

$$\operatorname{diam} f(V_{j,k}) \leq L_j \operatorname{diam} V_{j,k}. \tag{92}$$

Since $f(A) \cap f(B_{x_j}) \subset \bigcup_k f(V_{j,k})$,

$$\mathcal{H}^s_{L_j \delta_j}\big(f(A) \cap f(B_{x_j})\big) \leq \sum_{k=1}^{\infty} \big(\operatorname{diam} f(V_{j,k})\big)^s \leq L_j^s \sum_{k=1}^{\infty} \big(\operatorname{diam} V_{j,k}\big)^s. \tag{93}$$

Thus,

$$\mathcal{H}^s_{L_j \delta_j}\big(f(A) \cap f(B_{x_j})\big) \leq L_j^s \Big(\mathcal{H}^s_{\delta_j}(A \cap B_{x_j}) + \varepsilon 2^{-j}\Big). \tag{94}$$

Summing over j, and letting $\delta' \coloneqq \inf_j L_j \delta_j > 0$ (which is fine since $L_j, \delta_j > 0$), by the

subadditivity of outer measure,

$$\begin{aligned}\mathcal{H}^s_{\delta'}\big(f(A)\big) &\le \sum_{j=1}^{\infty} \mathcal{H}^s_{L_j\delta_j}\big(f(A)\cap f(B_{x_j})\big)\\ &\le \sum_{j=1}^{\infty} L_j^s \mathcal{H}^s_{\delta_j}\big(A\cap B_{x_j}\big) + \varepsilon\sum_{j=1}^{\infty} L_j^s 2^{-j}. \end{aligned} \tag{95}$$

Since $\varepsilon > 0$ is arbitrary, and we can let $\delta_j \downarrow 0$, we take the limit:

$$\mathcal{H}^s\big(f(A)\big) \le \sum_{j=1}^{\infty} L_j^s \mathcal{H}^s\big(A\cap B_{x_j}\big). \tag{96}$$

Now, assume $s > \dim_{\mathcal{H}}(A)$. By definition (9), $\mathcal{H}^s(A) = 0$. By monotonicity and countable subadditivity, the right-hand side is 0. Therefore, $\mathcal{H}^s(f(A)) = 0$, and thus $\dim_{\mathcal{H}}(f(A)) \le s$. Since $s > \dim_{\mathcal{H}}(A)$ was arbitrary, the conclusion (21) follows. □