AI Applications Lecture 12

Image Generation 2: Natural Image Decoder from Latent Space/Low-Dimensional Images

SUZUKI, Atsushi
Jing WANG

Outline

Introduction

Recap: Pipeline Components

$$\begin{tikzcd}
\left(\boldsymbol{c}^{[i]}\right)_{i=1}^{m} \arrow[r] & \text{TextEnc}^{[i]}_{\boldsymbol{\alpha}^{[i]}} : \mathcal{V}^{*} \to \mathbb{R}^{d^{[i]}} \arrow[d, "\left(\boldsymbol{e}^{[i]}\right)_{i=1}^{m}"] & \\
\boldsymbol{u} \arrow[r] & \text{LatentGen}_{\boldsymbol{\beta}} : \left(\prod_{i=1}^{m} \mathbb{R}^{d^{[i]}}\right) \times \mathcal{U} \to \mathcal{Z} \arrow[d, "\boldsymbol{z}"] & \\
& \text{Dec}_{\boldsymbol{\gamma}} : \mathcal{Z} \to \mathcal{I} \arrow[r] & \underline{\boldsymbol{X}}
\end{tikzcd}$$

Figure 1: Recap of the three-component pipeline.

Learning Outcomes

- **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**.

Preparation: Mathematical Notations

Notation (1/3): Definition and Sets

- **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 $k \in \mathbb{Z}_{>0} \cup \{+\infty\}$, $[1, k]_{\mathbb{Z}} \coloneqq \{1, 2, \dots, k\}$. When $k = +\infty$, $[1, k]_{\mathbb{Z}} = \mathbb{Z}_{>0}$.

Notation (2/3): Functions, Vectors, Sequences

- **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 $\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$.
    - The standard inner product is

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle := \sum_{i=1}^{n} u_i v_i \tag{1}$$

.

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

Notation (3/3): Matrices and Tensors

- **Matrix:**
  - Let the (i,j)-th element of $\boldsymbol{A} \in \mathbb{R}^{m,n}$ be $a_{i,j}$, $\boldsymbol{A} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix}$.
  - The transpose $\boldsymbol{A}^\top \in \mathbb{R}^{n,m}$ is

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

  - The transpose of a vector is

$$\boldsymbol{v}^\top = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix} \in \mathbb{R}^{1,n}. \tag{3}$$

- **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 $\underline{\boldsymbol{A}}$.

The Goal of a Natural Image Decoder

Definition and Goal

$$\mathrm{Dec}_\gamma : \mathcal{Z} \to \mathcal{I} \tag{4}$$

A natural image decoder is trained such that $\mathrm{Dec}_\gamma(\mathcal{Z})$ becomes a **subset** that sufficiently covers the set of **natural images**, reducing the burden on other components by allowing them to output in low-dimensional $\mathcal{Z}$ instead of high-dimensional $\mathcal{I}$.

Subset Manifold Intuition (Figure)

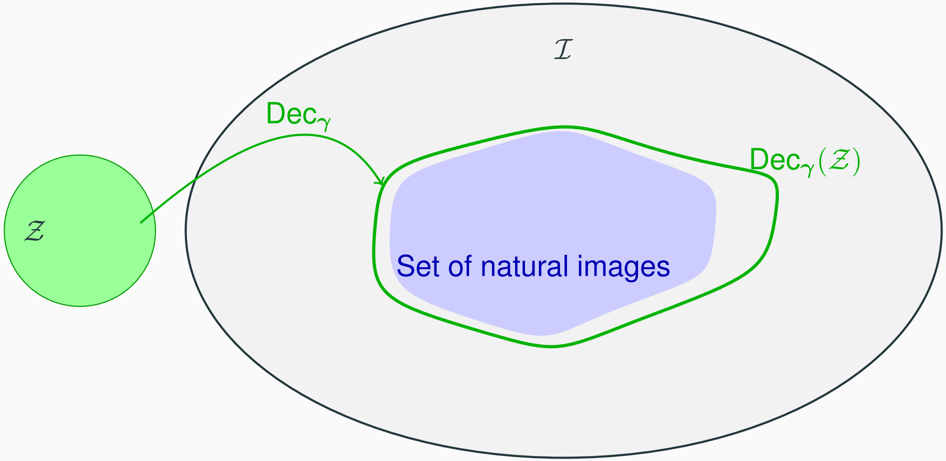

Figure 2: The image of the natural image decoder is a **subset** of the high-dimensional image space $\mathcal{I}$ and covers the region of natural images of interest.

Why Not the Entire Space? (Discrete View)

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 $d_{\mathcal{Z}} < d_{\mathcal{I}}$.

Why Not the Entire Space? (Continuous View)

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

Definition: Hausdorff Measure and Dimension (1/2)

Definition (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

$$\mathcal{H}^s_\delta(A) := \inf \left\{ \sum_{i=1}^{\infty} \omega_s \left(\frac{1}{2} \operatorname{diam} U_i \right)^s \,\middle|\, A \subset \bigcup_{i=1}^{\infty} U_i,\ \operatorname{diam} U_i < \delta \right\} \tag{5}$$

and

$$\mathcal{H}^s(A) := \lim_{\delta \downarrow 0} \mathcal{H}^s_\delta(A) \tag{6}$$

is called the s-dimensional Hausdorff measure. Here, $\operatorname{diam} U := \sup\{\|x - y\| : \ x, y \in U\}$.

Definition: Hausdorff Measure and Dimension (2/2)

ω_s is defined using the Gamma function Γ as $\omega_s := \frac{\pi^{s/2}}{\Gamma\left(\frac{s}{2}+1\right)}$. 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

$$\dim_{\mathcal{H}}(A) := \inf\{s \geq 0 : \ \mathcal{H}^s(A) = 0\} = \sup\{s \geq 0 : \ \mathcal{H}^s(A) = \infty\} \tag{7}$$

. [1]

Remark on Definition

Remark

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 $[0, +\infty]$.

Intuition: Measuring with Wrong Dimension

A measure means “volume”. For a square, the 1D Hausdorff measure behaves like measuring length—yielding $+\infty$; the 3D Hausdorff measure behaves like measuring volume—yielding 0; only at $s = 2$ it is finite positive. This matches our intuitive dimension.

Proposition: Hypercube Dimension

Proposition (Hausdorff dimension of a hypercube)

Let $d \in \mathbb{Z}_{>0}$. We rigorously find the Hausdorff dimension of $A := [0,1]^d \subset \mathbb{R}^d$ using the normalization ($\omega_s(\frac{1}{2}\operatorname{diam})^s$) in Definitions (5), (6)*. The conclusion is*

$$\dim_{\mathcal{H}}(A) = d \tag{8}$$

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

Lipschitz Maps (1/2)

Definition (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 $u, v \in X$,

$$d_Y\big(f(u), f(v)\big) \leq L\, d_X(u, v) \tag{9}$$

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

$$\|f(u) - f(v)\|_2 \leq L\,\|u - v\|_2 \tag{10}$$

.

Neighborhoods and Local Lipschitz (2/2)

Definition (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 $B(x, r) \subset U$.

Definition (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 $U_x \subset X$ of x and a constant $L_{U_x} \geq 0$ such that for all $u, v \in U_x$,

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

holds.

Examples: Lipschitz and Local Lipschitz (1/2)

Example (Examples of functions with Lipschitz and locally Lipschitz properties)

On $(\mathbb{R}, |\cdot|)$:

- $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ is **globally Lipschitz.** Since $f'(x) = \sigma(x)\,(1 - \sigma(x))$ and $0 \leq f'(x) \leq 1/4$,

$$|\sigma(u) - \sigma(v)| \leq \tfrac{1}{4}|u - v|. \tag{12}$$

- $f(x) = x^2$ is **not globally Lipschitz** but **locally Lipschitz.** For $|x|, |y| \leq R$,

$$|x^2 - y^2| = |x + y|\,|x - y| \leq 2R\,|x - y|. \tag{13}$$

Examples: Lipschitz and Local Lipschitz (2/2)

Example (cont.)

- $f(x) = \sqrt{x}$ on $[0, \infty)$ is **not locally Lipschitz at** 0. Assuming $|\sqrt{u} - \sqrt{v}| \leq L|u - v|$ in $(0, \delta)$ and setting $v = 0$ yields $\frac{1}{\sqrt{t}} \leq L$, a contradiction as $t \downarrow 0$.
- $f(x) = \mathrm{ReLU}(x) = \max\{0, x\}$ is **globally Lipschitz** with $L = 1$, by case analysis:

$$u, v \geq 0 : \ |f(u) - f(v)| = |u - v|, \tag{14}$$

$$u, v \leq 0 : \ |f(u) - f(v)| = 0 \leq |u - v|, \tag{15}$$

$$u \geq 0 \geq v : \ |f(u) - f(v)| = u \leq u - v = |u - v|. \tag{16}$$

Theorem: Dimension Non-Increase Under Locally Lipschitz Maps

Theorem (Hausdorff dimension under Lipschitz maps)

For a set $A \subset \mathbb{R}^n$ and a locally Lipschitz map $f : \mathbb{R}^n \to \mathbb{R}^m$,

$$\dim_{\mathcal{H}}(f(A)) \leq \dim_{\mathcal{H}}(A) \tag{17}$$

holds. [1]

Remark: Space-Filling Curves vs NNs

Remark

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.

Fully Convolutional VAE and Computational Cost Reduction

Meaning of Fully Convolutional

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 §**??**). This allows generating images at arbitrary resolutions by changing the number of tiles in the input latent (see §**??**).

Computational Cost Reduction: Latent vs Pixel (1/2)

In a typical 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]. The **spatial cost** of a latent-space UNet is approximately $1/8^2 = 1/64$ compared to a pixel-space UNet.

---

[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)

Computational Cost Reduction: FLOPs Scaling (2/2)

(Downsampling Factor and FLOPs Scaling) When computing in a latent space with s-times downsampling and comparable architecture, main convolution FLOPs decrease approximately as $1/s^2$. Thus, if $s = 8$, reduction is $\approx 1/64$.

(Derivation) The FLOPs for one 2D convolution layer are roughly

$$H_{\mathrm{out}} W_{\mathrm{out}} \cdot K_H K_W \cdot C_{\mathrm{in}} C_{\mathrm{out}} \tag{18}$$

. With $H \mapsto H/s$, $W \mapsto W/s$, the product $H_{\mathrm{out}} W_{\mathrm{out}}$ scales by $1/s^2$.

Convolutional Operations and the Natural Image Decoder

Variable-Length Input/Output Motivation

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. Naturally, composing variable-length input/output functions results in another variable-length input/output function.

Remark: Convolution Sign Convention

Remark (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 and 2 D signal

$$(y \text{ by } \star)_t := \sum_{i \in I_K} k_i \, x_{t+i}, \quad (Y \text{ by } \star)_{u,v} := \sum_{i \in I_{K_H}} \sum_{j \in I_{K_W}} K_{i,j} \, X_{u+i,\, v+j}, \tag{19}$$

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

$$(y \text{ by } *)_t = \sum_i k_i \, x_{t-i}, \qquad (Y \text{ by } *)_{u,v} = \sum_{i,j} K_{i,j} \, X_{u-i,\, v-j} \tag{20}$$

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: On “n-dimensional Convolution”

Remark (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.

1D Convolution: Definitions and Layers

Goal: Variable-Length Parametric Function (1D)

Definition (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:

- **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** $\mathsf{InLen2OutLen} : \mathcal{D}_{\mathrm{in}} \to \mathbb{Z}_{\geq 0}$.
- 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:

$$f_{\boldsymbol{\theta}}^{(L_{\mathrm{in}})} : \mathbb{R}^{L_{\mathrm{in}}} \to \mathbb{R}^{\mathsf{InLen2OutLen}(L_{\mathrm{in}})}. \tag{21}$$

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

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

Mathematical Definition: 1D $\star$ Operator

Definition (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

$$I_K := \begin{cases} \{-r, -r+1, \dots, 0, \dots, +r\}, & K = 2r+1 \text{ (odd)}, \\ \{-r+1, -r+2, \dots, +r\}, & K = 2r \text{ (even)} \end{cases} \tag{23}$$

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:

$$y_t := \sum_{i \in I_K} k_i \, x_{t+i}, \qquad t \in \mathbb{Z} \tag{24}$$

The size of the output of the $\star$ operation

Remark (The size of the output of the $\star$ operation)

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: 1D $\star$ Operation — Setup

Example (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. (24):

Example Step $t = 0$

$$y_0 = 2 \cdot x_{-1} + (-1) \cdot x_0 + 3 \cdot x_1 = 0 - 1 + 6 = 5.$$

Example Step $t = 1$

$$y_1 = 2 \cdot x_0 + (-1) \cdot x_1 + 3 \cdot x_2 = 2 - 2 + 0 = 0.$$

Example Step $t = 2$

$$y_2 = 2 \cdot x_1 + (-1) \cdot x_2 + 3 \cdot x_3 = 4 - 0 - 3 = 1.$$

Example Step $t = 3$

$$y_3 = 2 \cdot x_2 + (-1) \cdot x_3 + 3 \cdot x_4 = 0 + 1 + 0 = 1.$$

Example Step $t = 4$

$$y_4 = 2 \cdot x_3 + (-1) \cdot x_4 + 3 \cdot x_5 = -2 - 0 + 0 = -2.$$

Example Step $t = 5$ and Result

$$y_5 = 2 \cdot x_4 + (-1) \cdot x_5 + 3 \cdot x_6 = 0.$$

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

Exercise: 1D $\star$ Operation — Setup

Exercise (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. (24), $L_{\text{out}} = 5 + 2 - 1 = 6$, and for each $t = 0, \ldots, 5$:

Answer Step $t = 0$

$$y_0 = 1 \cdot x_0 + (-2) \cdot x_1 = 3 - 0 = 3.$$

Answer Step $t = 1$

$$y_1 = 1 \cdot x_1 + (-2) \cdot x_2 = 0 - (-2) \cdot 2 = 4.$$

Answer Step $t = 2$

$$y_2 = 1 \cdot x_2 + (-2) \cdot x_3 = -2 - 2 = -4.$$

Answer Step $t = 3$

$$y_3 = 1 \cdot x_3 + (-2) \cdot x_4 = 1 - 8 = -7.$$

Answer Step $t = 4$

$$y_4 = 1 \cdot x_4 + (-2) \cdot x_5 = 4 - 0 = 4.$$

Answer Step $t = 5$ and Result

$$y_5 = 1 \cdot x_5 + (-2) \cdot x_6 = 0.$$

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

1D Convolutional Layer: Parameters and Shapes

Definition (1D Convolutional Layer (with stride/padding/dilation, multi-channel, activation; variable-length parametric function by $\star$))

This layer is defined as a variable-length parametric function per Definition 8.

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 (23). The learnable parameters are:

$$\boldsymbol{k}^{[c,r]} = (k_i^{[c,r]})_{i \in I_K}, \qquad b^{[c]} \in \mathbb{R} \tag{25}$$

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

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

$$\mathsf{InLen2OutLen}(L) \;:=\; \left\lfloor \frac{L + 2P - D\,(K-1) - 1}{S} \right\rfloor + 1 \;\in\; \mathbb{Z}_{\geq 0}. \tag{26}$$

1D Convolutional Layer: Forward and Activation

Definition (continued)

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 $\boldsymbol{x}^{[r]} \in \mathbb{R}^{L}$ $(r = 1, \dots, m)$ and for each output channel as $\boldsymbol{y}^{[c]} \in \mathbb{R}^{\mathsf{InLen2OutLen}(L)}$ $(c = 1, \dots, n)$. We define the zero-padded vector $\tilde{\boldsymbol{x}}^{[r]} \in \mathbb{R}^{L+2P}$ as:

$$\tilde{x}_t^{[r]} = \begin{cases} x_{t-P}^{[r]}, & t \in [0, L+2P-1]_{\mathbb{Z}}, \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

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

$$y_t^{[c]} := \sum_{r=1}^{m} \sum_{i \in I_K} k_i^{[c,r]} \, \tilde{x}_{tS+iD}^{[r]} + b^{[c]} \quad (c = 1, \dots, n), \tag{28}$$

$$\widehat{\boldsymbol{Y}} := \mathsf{Act}(\boldsymbol{Y}) \quad \text{(Activation Act is applied element-wise).} \tag{29}$$

Figure: 1D 1ch→1ch, $L = 4, K = 2, S = 1$

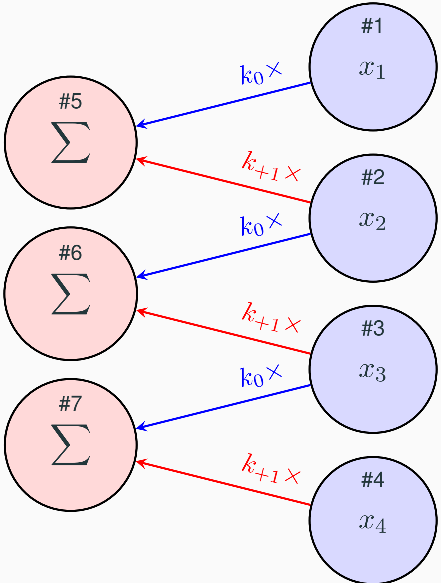

Figure: 1D 1ch→1ch, $L = 3, K = 2, S = 1$

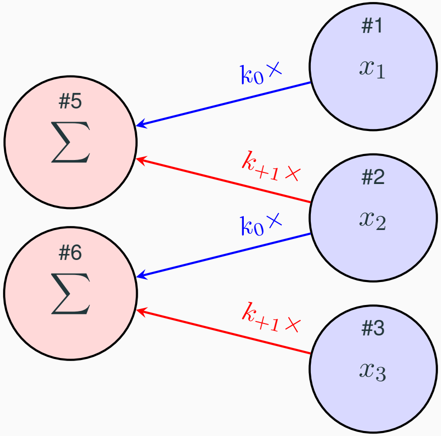

Figure 4: 1D 1ch→1ch, $L = 3, K = 2, S = 1$ ($\star$ operation). $L_{\text{out}} = 2$.

Figure: 1D 1ch$\rightarrow$2ch, $L = 4, K = 2, S = 1$

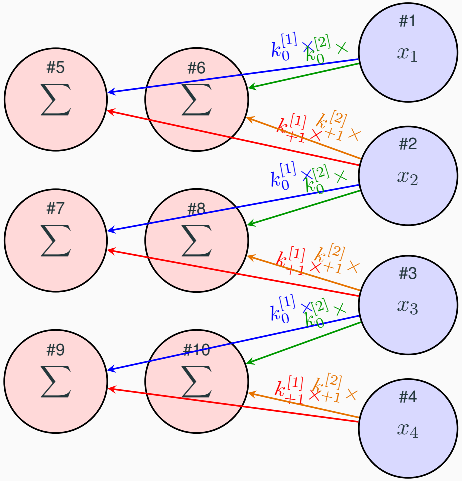

Figure 5: 1D 1ch$\rightarrow$2ch, $L = 4, K = 2, S = 1$ ($\star$ operation).

Figure: 1D 1ch→2ch, $L = 3, K = 2, S = 1$

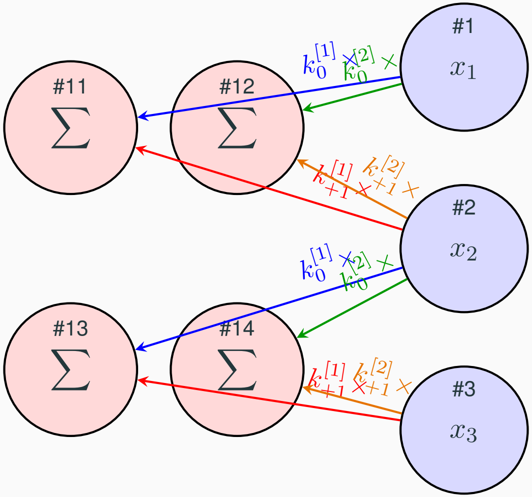

Figure 6: 1D 1ch→2ch, $L = 3, K = 2, S = 1$ ($\star$ operation).

Figure: 1D 2ch→2ch, $L = 4, K = 2, S = 1$

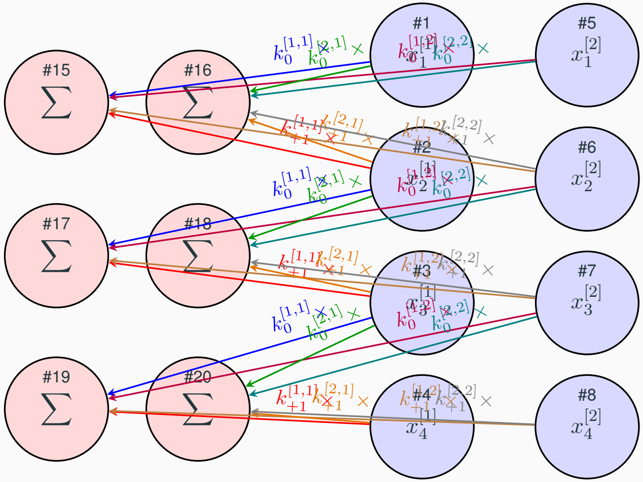

Figure 7: 1D 2ch→2ch, $L = 4, K = 2, S = 1$ ($\star$ operation; missing edges completed).

Figure: 1D 2ch→2ch, $L = 3, K = 2, S = 1$

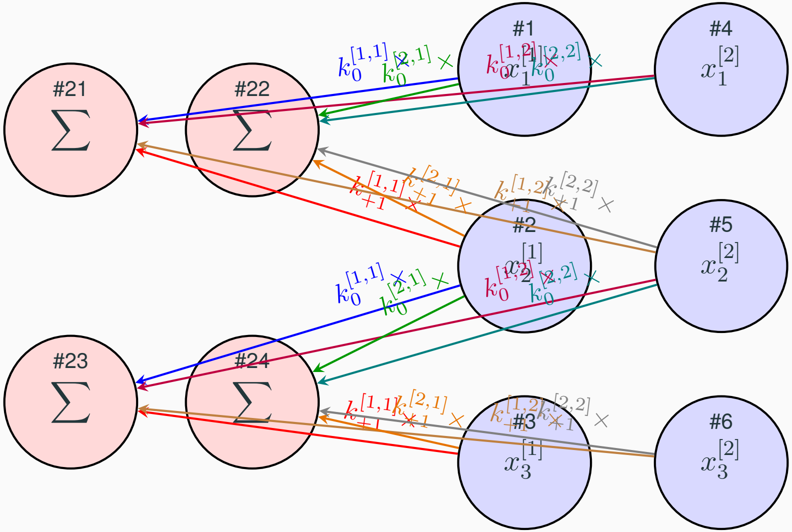

Figure 8: 1D 2ch→2ch, $L = 3, K = 2, S = 1$ (* operation)

Figure: 1D $K = 3, S = 1, P = 1, D = 1$

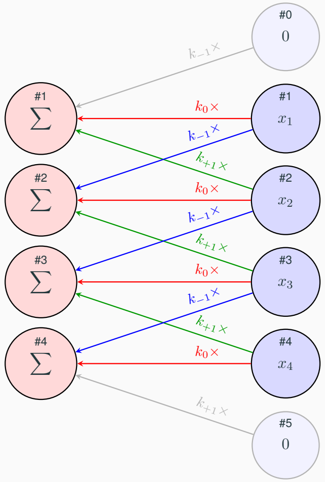

Figure 9: 1D 1ch→1ch, $L = 4, K = 3, S = 1, P = 1, D = 1$ ($\star$ operation; $L_{\text{out}} = 4$).

Figure: 1D $K = 3, S = 2, P = 1, D = 1$

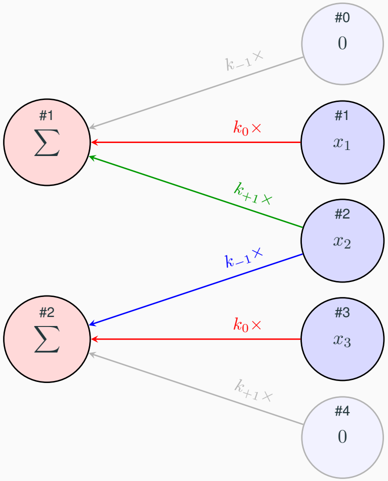

Figure 10: 1D 1ch$\rightarrow$1ch, $L = 3, K = 3, S = 2, P = 1, D = 1$ ($\star$ operation; $L_{\text{out}} = 2$).

Figure: 1D $K = 3, S = 1, P = 1, D = 2$

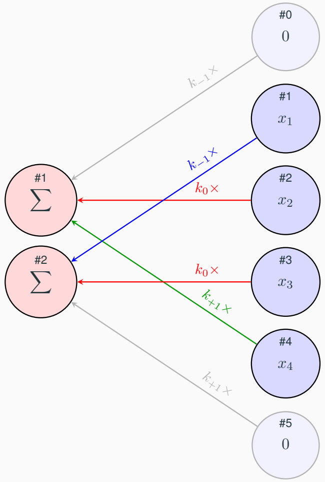

Figure 11: 1D 1ch→1ch, $L = 4, K = 3, S = 1, P = 1, D = 2$ ($\star$ operation; $L_{\text{out}} = 2$).

1D Nearest-Neighbor Upscaler

Definition (1D Nearest-Neighbor Upscaler (Variable-Length))

Given an upscale factor $s \in \mathbb{Z}_{\geq 2}$. Parameters are empty ($d_{\mathrm{param}} = 0$), the acceptable set is $\mathcal{D}_{\mathrm{in}} = \mathbb{Z}_{\geq 0}$, and the input/output length mapping is defined by:

$$\mathsf{InLen2OutLen}(L) := sL \tag{30}$$

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

$$\left(f^{(L)}(\boldsymbol{x})\right)_t \;:=\; x_{\lfloor t/s \rfloor}, \qquad t \in [0, sL-1]_{\mathbb{Z}} \tag{31}$$

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

Figure: 1D NN Upscaler Example

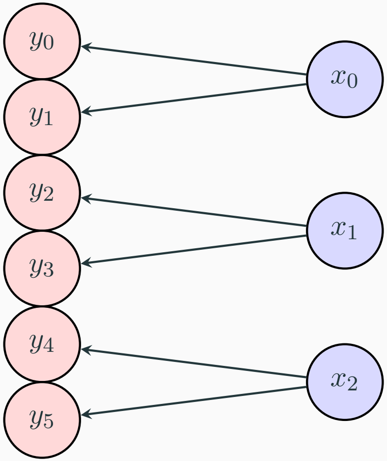

Figure 12: 1D Nearest-Neighbor Upscaler (example with $s = 2$, $L = 3$).

1D Group Normalization

Definition (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 $G_k = \{(k-1)\frac{c}{g} + 1, \ldots, k\frac{c}{g}\}$:

$$\mu_k := \frac{1}{|G_k|\, L} \sum_{i \in G_k} \sum_{t=1}^{L} f_t^{[i]}, \qquad \sigma_k^2 := \frac{1}{|G_k|\, L} \sum_{i \in G_k} \sum_{t=1}^{L} \left(f_t^{[i]} - \mu_k\right)^2. \tag{32}$$

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

$$\widehat{f}_t^{[i]} := \gamma_i \frac{f_t^{[i]} - \mu_{k(i)}}{\sqrt{\sigma_{k(i)}^2 + \varepsilon}} + \beta_i, \qquad i \in G_{k(i)},\ t = 1, \ldots, L. \tag{33}$$

2D Convolution: Definitions and Layers

Goal: Variable-Length Parametric Function (2D)

Definition (Variable-Length Parametric Function (2D))

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

$$F_{\boldsymbol{\theta}}^{(H,W)} : \mathbb{R}^{H \times W} \to \mathbb{R}^{H_{\mathrm{out}}(H,W) \times W_{\mathrm{out}}(H,W)} \tag{34}$$

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

Mathematical Definition: 2D $\star$ Operator

Definition (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

$$I_{K_H},\, I_{K_W} \text{ are defined similarly to (23).} \tag{35}$$

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:

$$Y_{u,v} := \sum_{i\in I_{K_H}} \sum_{j\in I_{K_W}} K_{i,j}\, X_{u+i,\, v+j}, \qquad (u,v) \in \mathbb{Z}^2 \tag{36}$$

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: 2D $\star$ Operation — Result

Example (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. (36), $H_{\text{out}} = 3$, $W_{\text{out}} = 3$, and:

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

Exercise: 2D $\star$ Operation — Setup

Exercise (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: One Entry Demonstration

Answer

For each (u, v), we evaluate by substituting directly into Eq. (36). For example, $Y_{0,2}$ is:

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

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

2D Convolutional Layer: Parameters and Output Size

Definition (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 15. 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:

$$\boldsymbol{K}^{[c,r]} = (K_{i,j}^{[c,r]})_{i\in I_{K_H}, j\in I_{K_W}}, \qquad b^{[c]} \in \mathbb{R}. \tag{37}$$

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

$$\tilde{X}_{p,q}^{[r]} = \begin{cases} X_{p-P_H, q-P_W}^{[r]}, & p \in [0, H+2P_H-1]_{\mathbb{Z}},\ q \in [0, W+2P_W-1]_{\mathbb{Z}}, \\ 0, & \text{otherwise} \end{cases} \tag{38}$$

The output spatial size is:

2D Convolutional Layer: Forward and Activation

Definition (continued)

For each c and (p, q):

$$Y_{p,q}^{[c]} := \sum_{r=1}^{m} \sum_{i \in I_{K_H}} \sum_{j \in I_{K_W}} K_{i,j}^{[c,r]} \, \tilde{X}^{[r]}_{pS_H+iD_H,\, qS_W+jD_W} \; + \; b^{[c]}, \tag{41}$$

$$\underline{\widehat{\boldsymbol{Y}}} := \text{Act}(\underline{\boldsymbol{Y}}) \, . \tag{42}$$

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

Figure: 2D 1ch→1ch, Stride 1

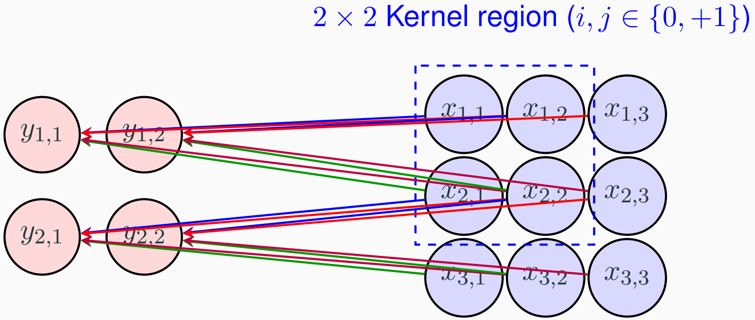

Figure 13: Example of 2D 1ch→1ch (3×3 input, 2×2 kernel, stride 1; $\star$ operation). $H_{\text{out}} = W_{\text{out}} = 2$ (Eq. (39),(40)).

Figure: 2D $m$ch$\rightarrow$ $n$ch Schematic

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

$$Y^{[c]}_{p,q} = \sum_{r=1}^{m} \sum_{i\in I_{K_H}} \sum_{j\in I_{K_W}} K^{[c,r]}_{i,j}\, \tilde{X}^{[r]}_{pS_H+iD_H,\, qS_W+jD_W} + b^{[c]}. \tag{43}$$

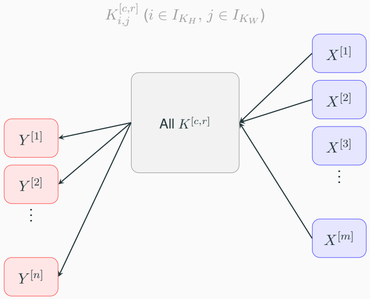

Figure 14: Schematic of 2D $m$ch$\rightarrow$ $n$ch (Eq. (39) – (41), (43); $\star$ operation).

2D Nearest-Neighbor Upscaler

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

Given an upscale factor $s \in \mathbb{Z}_{\geq 2}$. $d_{\text{param}} = 0$, 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 $\boldsymbol{X} \in \mathbb{R}^{H \times W}$,

$$\left(F^{(H,W)}(\boldsymbol{X})\right)_{sp+a,\ sw+b} := X_{p,q}, \quad a, b \in [0, s-1]_{\mathbb{Z}},\ (p, q) \in [0, H-1]_{\mathbb{Z}} \times [0, W-1]_{\mathbb{Z}}. \tag{44}$$

2D Group Normalization

Definition (Group Normalization (2D; Variable-Length))

With the same settings as Definition 14, 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. (32) and (33), we define a family of variable-length parametric functions $\{\widehat{\underline{\boldsymbol{F}}}\}$ with $d_{\text{param}} = 2c$ (the output spatial size is the same as the input, (H, W)).

Fully-Convolutional Networks and Significance

Fully-Convolutional Networks (FCN)

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.)

Summary and Next Lecture Preview

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 22).

Significance of Being an Autoencoder

AE Pair: Motivation and Structure

The encoder $\mathrm{Enc}_{\boldsymbol{\eta}} : \mathcal{I} \to \mathcal{Z}$ and decoder $\mathrm{Dec}_{\gamma} : \mathcal{Z} \to \mathcal{I}$ satisfy, for $\underline{\boldsymbol{X}} \sim p_{\mathrm{data}}$,

$$\underline{\boldsymbol{X}} \approx \mathrm{Dec}_{\gamma}\big(\mathrm{Enc}_{\boldsymbol{\eta}}(\underline{\boldsymbol{X}})\big) \tag{45}$$

with reconstruction measures such as L^1/L^2 loss or perceptual loss.

Practical Significance: Compatibility

Compatibility: Multiple decoders Dec_{γ_1}, Dec_{γ_2} can satisfy (45) for a fixed encoder. Fixing Enc_{η} and retraining a decoder preserves the latent representation specification and interoperability.

Significance of Being a Variational Autoencoder

VAE: Setup and Caveat

A variational autoencoder (VAE) [3] is used, trained with a reconstruction term and a regularization term. Practical objectives can be more complex; here we state a basic form.

VAE Objective (1/3): Encoders and Reparametrization

Definition (VAE Objective Function using an Uncorrelated Gaussian Distribution)

For each input image $\underline{\boldsymbol{X}} \in \mathcal{I}$, define

$$\text{MeanEnc}_{\boldsymbol{\eta}_{\text{mean}}} : \mathcal{I} \to \mathbb{R}^d, \qquad \boldsymbol{\mu}(\underline{\boldsymbol{X}}) := \text{MeanEnc}_{\boldsymbol{\eta}_{\text{mean}}}(\underline{\boldsymbol{X}}), \tag{46}$$

$$\text{SDEnc}_{\boldsymbol{\eta}_{\text{SD}}} : \mathcal{I} \to \mathbb{R}^d_{>0}, \qquad \boldsymbol{\sigma}(\underline{\boldsymbol{X}}) := \text{SDEnc}_{\boldsymbol{\eta}_{\text{SD}}}(\underline{\boldsymbol{X}}) \tag{47}$$

and set

$$\boldsymbol{\eta} := \left(\boldsymbol{\eta}_{\text{mean}}, \boldsymbol{\eta}_{\text{SD}}\right). \tag{48}$$

Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and define

$$\boldsymbol{z}_{\boldsymbol{\epsilon}} := \boldsymbol{\mu}(\underline{\boldsymbol{X}}) + \boldsymbol{\sigma}(\underline{\boldsymbol{X}}) \odot \boldsymbol{\epsilon}, \qquad \hat{\boldsymbol{X}}_{\boldsymbol{\epsilon}} := \text{Dec}_{\gamma}\left(\boldsymbol{z}_{\boldsymbol{\epsilon}}\right) \tag{49}$$

VAE Objective (2/3): Loss Function

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

$$\mathcal{L}(\boldsymbol{\eta}, \gamma) := \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}\left[\ell\big(\underline{\boldsymbol{X}}, \underline{\hat{\boldsymbol{X}}}_{\boldsymbol{\epsilon}}\big)\right] \; + \; \beta \sum_{i=1}^{d} \big(\mu_i(\underline{\boldsymbol{X}})^2 + \sigma_i(\underline{\boldsymbol{X}})^2 - \log \sigma_i(\underline{\boldsymbol{X}})^2 - 1\big) \tag{50}$$

with weight $\beta > 0$; the second term is the closed-form KL divergence [3].

VAE Objective (3/3): Remarks

Remark

- In practice, ℓ is often the L^1 loss. Compared to squared error, L^1 is robust to extreme outliers.
- Practical losses often include LPIPS [5] and adversarial terms [2]. They are **quite complex**.
- Each term in (50) requests:
  - **Reconstruction**: Encoder/decoder are **approximate inverses**.
  - **Regularization (concentration to origin)**: μ, σ concentrate near the origin so latents near the origin decode to natural images.
  - **Small perturbation robustness**: The decoder learns that small latent perturbations induce small output perturbations.

Summary and Next Lecture

Summary

- **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**: A decoder enables significant cost reduction since we only need to **generate the low-dimensional latent**.
- **Fully Convolutional VAE**: Fully-convolutional structure gives arbitrary output sizes by input sizing; as an autoencoder it has an approximate inverse; variational learning adds origin-concentration and robustness.

Next Lecture

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 i

[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.

References ii

[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.

References iii

[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.