Deep Learning: Models and Optimization

Marco Cuturi

ENSAE ParisTech
École nationale de la statistique et de l'administration économique
université PARIS-SACLAY

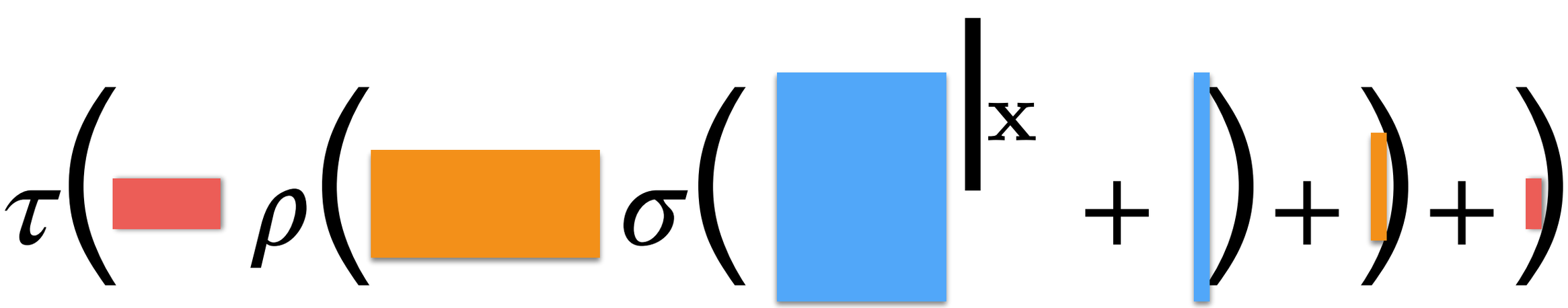
What is “deep learning”

Statistical machine learning problems handled with compositional models

What is “deep learning”

Statistical machine learning problems handled with compositional models

$$f(\mathbf{x}, \Theta) := g_3\left(g_2\left(g_1\left(\mathbf{x}, \theta_1\right), \theta_2\right), \theta_3\right)$$



What is “deep learning”

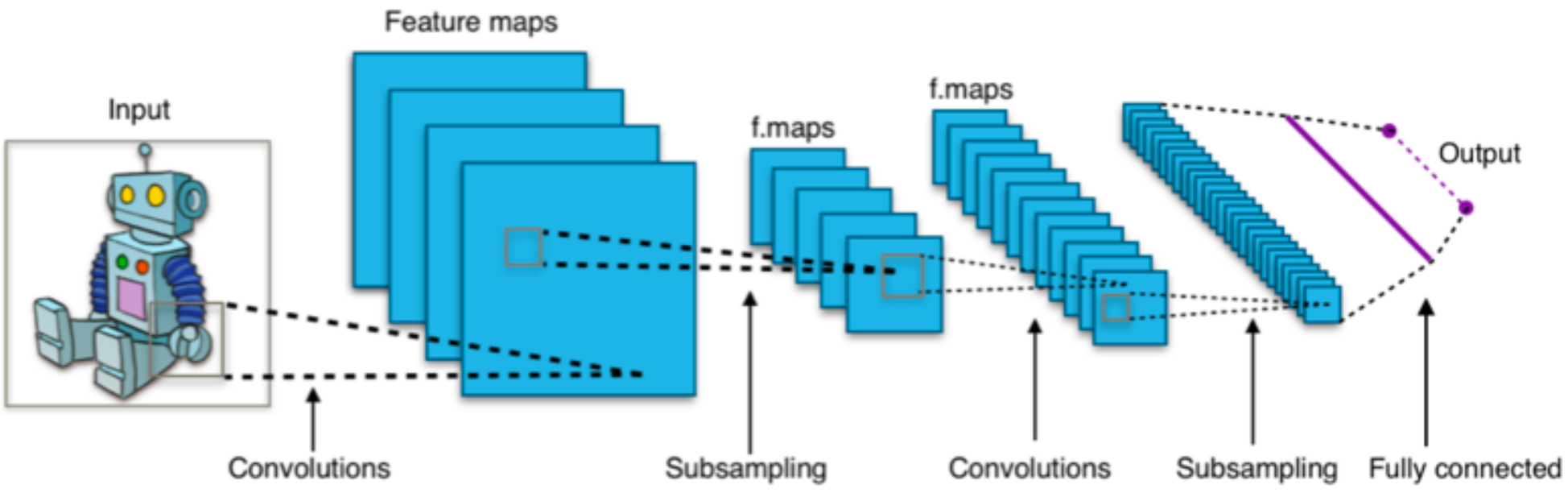
Statistical machine learning problems handled with compositional models

$$f(\mathbf{x}, \Theta) := g_3\left(g_2\left(g_1\left(\mathbf{x}, \theta_1\right), \theta_2\right), \theta_3\right)$$

What is “deep learning”

Statistical machine learning problems handled with compositional models

$$f(\mathbf{x}, \Theta) := g_3\left(g_2\left(g_1\left(\mathbf{x}, \theta_1\right), \theta_2\right), \theta_3\right)$$

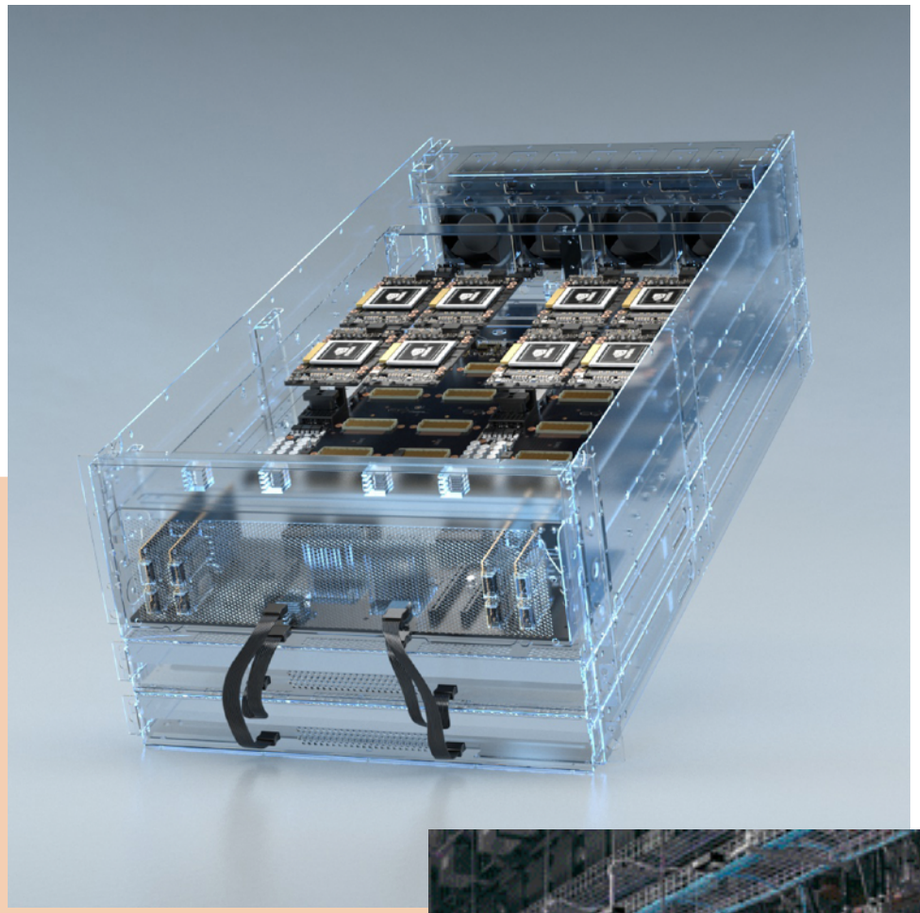


What is “deep learning”

Execution on
massively
parallel
hardware

What is “deep learning”

Execution on massively parallel hardware



What is “deep learning”

Automatic differentiation to compute gradients

What is “deep learning”

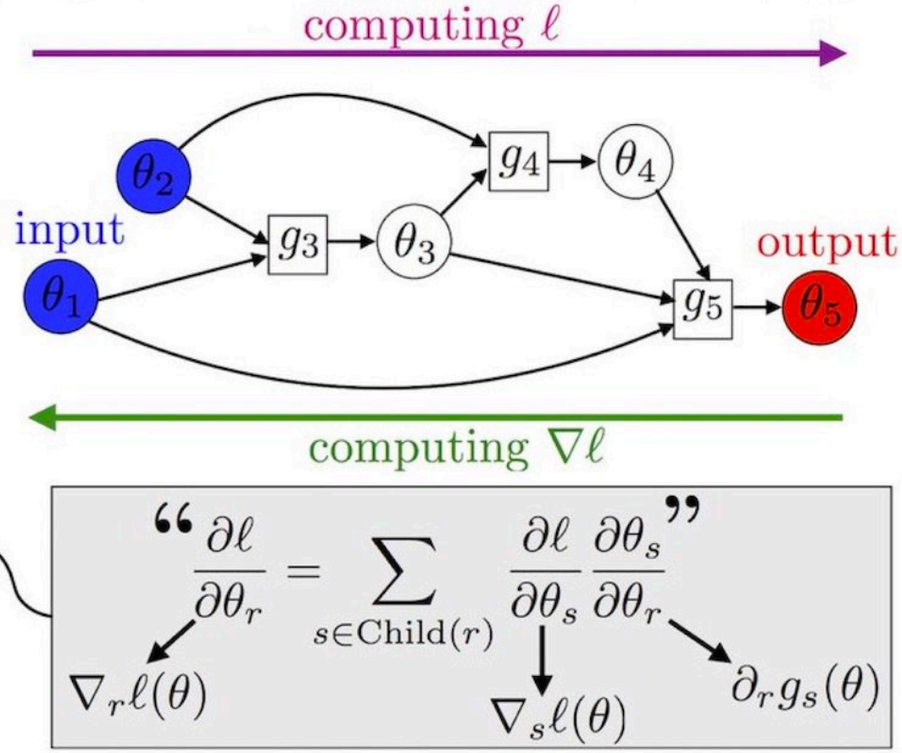
Computer program ⇔ directed acyclic graph ⇔ linear ordering of nodes $(\theta_r)_r$

forward

```
function ℓ(θ_1, ..., θ_M)
  for r = M+1, ..., R
    θ_r = g_r(θ_Parents(r))
  return θ_R
```

backward

```
function ∇ℓ(θ_1, ..., θ_M)
  ∇_R ℓ = 1
  for r = R−1, ..., 1
    ∇_r ℓ = Σ_{s∈Child(r)} ∂_r g_s(θ) ∇_s ℓ
  return (∇_1 ℓ, ..., ∇_M ℓ)
```



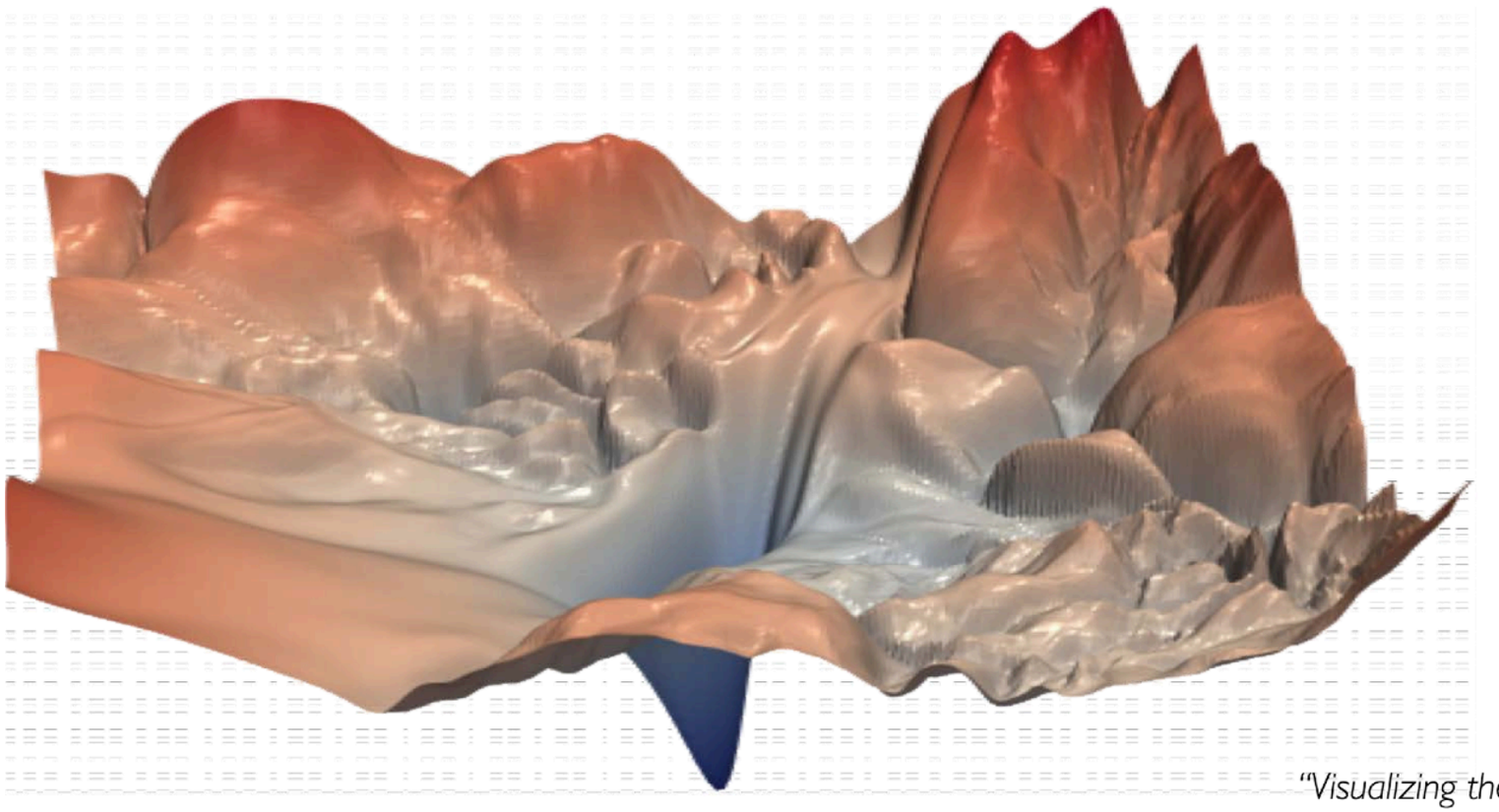
Automatic differentiation to compute gradients

source: Gabriel Peyré

What is “deep learning”

Non-convex
Stochastic Optimization

What is “deep learning”



“Visualizing the loss landscape of neural nets”. Dec 2017.

Non-convex
Stochastic Optimization

What is “deep learning”

Statistical machine learning problems handled with compositional models

Execution on massively parallel hardware

“ Deep Learning ”

Automatic differentiation to compute gradients

Non-convex *Stochastic* Optimization

Self-introduction

- ENSAE ('01) / MVA / Phd. ENSMP / Japan & US
  - post-doc then hedge-fund in Japan ('05~'08)
  - Lecturer @ Princeton University ('09~'10)
  - Assoc. Prof. @ Kyoto University ('10~'16)
  - Prof @ ENSAE since 9/'16.
  - Research scientist @ Google Brain since 10/'18
- Active in ML community, stats/optim flavor.
  - Attend & publish regularly in *NeurIPS* & *ICML*.
- Interests
  - Optimal transport, kernel methods, time series.

Practical Aspects

- Reach me:
  - Office: 3021
  - email: marco.cuturi@ensae.fr
  - page web: http://marcocuturi.net
- Lectures: structure
  - 4 x (90+90) min. class.
  - 2 x (90+90) min. python hands-on sessions
  - Validation through memoir.

Schedule

1. Introduction, Backpropagation

   Empirical risk minimization (R-ERM), specificities of deep learning, automatic differentiation.

2. Modeling blocks: convolution, recursion, attention
3. Accelerators, Non-convex optimization
4. Unsupervised problems: GANs / (V)AEs. Graph-NN

list of ingredients in *batch* ML

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathcal{X} \times \mathcal{Y})^n$$

samples from $p \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$

loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$

function class $\mathcal{F} = \{f_\theta : \mathcal{X} \to \mathcal{Y}, \theta \in \Theta\}$

regularizer $\psi : \Theta \to \mathbb{R}_+$

list of ingredients in *batch* ML

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathcal{X} \times \mathcal{Y})^n$$

$n \approx \infty$

$\dim(\mathcal{X}) \approx \infty$

$$\text{samples from } p \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$$

$$\text{loss function } l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$$

$$\text{function class } \mathcal{F} = \{f_\theta : \mathcal{X} \to \mathcal{Y}, \theta \in \Theta\}$$

$$\text{regularizer } \psi : \Theta \to \mathbb{R}_+$$

Goal of Batch ML

1. The gold standard: Risk Minimization

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathbb{E}_p[\boldsymbol{l}(f_{\boldsymbol{\theta}}(X), Y)]$$

2. The naive alternative: Empirical Risk Minimization

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{l}(f_{\boldsymbol{\theta}}(x_i), y_i)$$

Supervised ML

3. The reasonable compromise

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{l}(f_{\boldsymbol{\theta}}(x_i), y_i)$$

From an optimization point of view:

- parameter size is huge.
- loss and regularizer functions might be ugly.
- n points might be too much for a single RAM machine (~256Gb *vs.* a few terabytes of more for modern datasets).

Supervised ML

3. The reasonable compromise

$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{l}(f_{\boldsymbol{\theta}}(x_i), y_i) + \boldsymbol{\psi}(\boldsymbol{\theta})$$

From an optimization point of view:

- parameter size is huge.
- loss and regularizer functions might be ugly.
- n points might be too much for a single RAM machine (~256Gb *vs.* a few terabytes of more for modern datasets).

list of ingredients in online ML

$$(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}, t \geq 0.$$

each sampled from $p \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$

$$\text{loss function } l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$$

$$\text{function class } \mathcal{F} = \{f_\theta : \mathcal{X} \to \mathcal{Y}, \theta \in \Theta\}$$

$$\text{regularizer } \psi : \Theta \to \mathbb{R}_+$$

Online ML

1. Ideally, one wants to minimize the same risk

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathbb{E}_p[\boldsymbol{l}(f_{\boldsymbol{\theta}}(X), Y)]$$

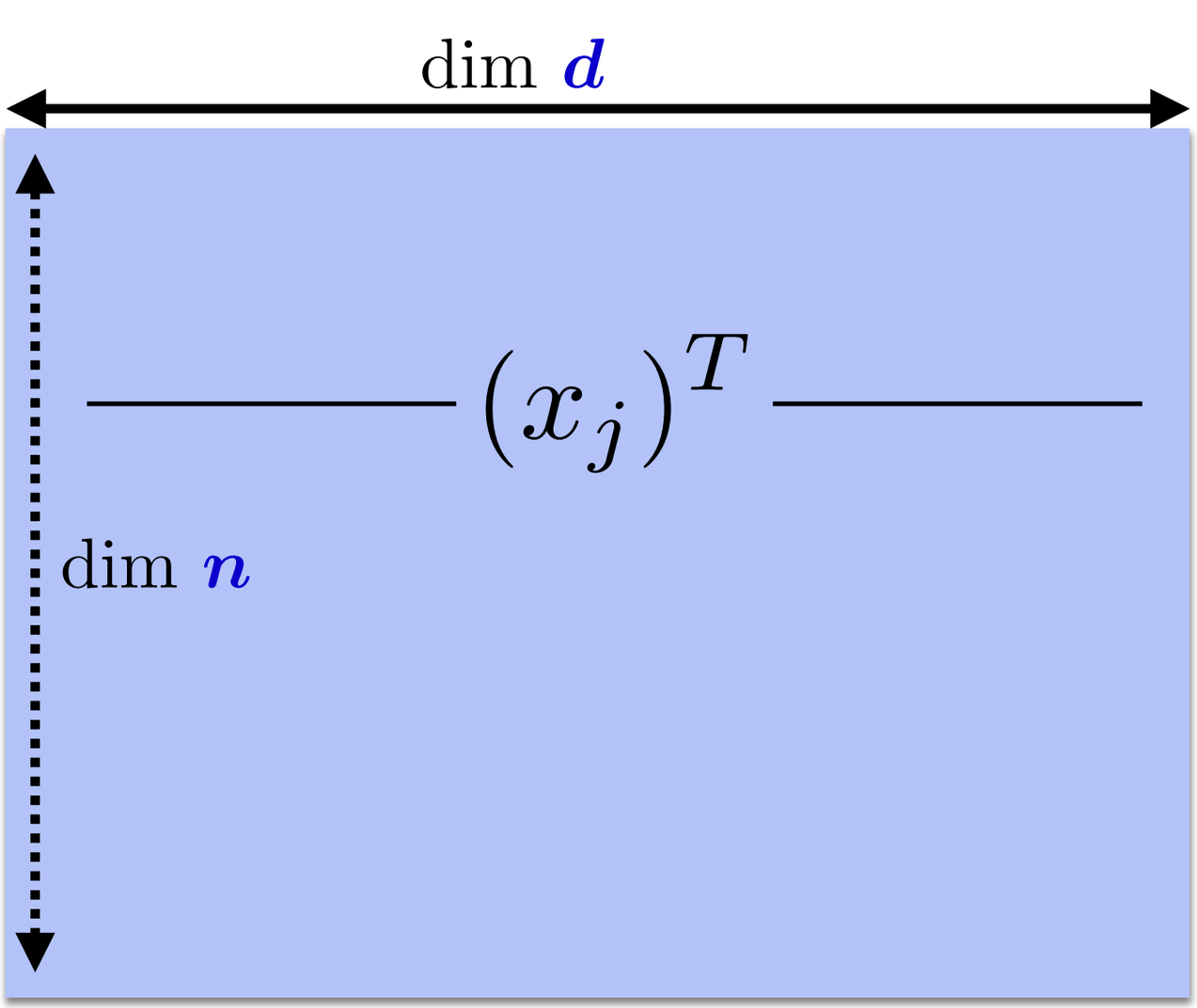
2. Due to practical constraints, samples only come **one by one**, each at a time t, and **cannot be stored**. Only previous parameter is stored.

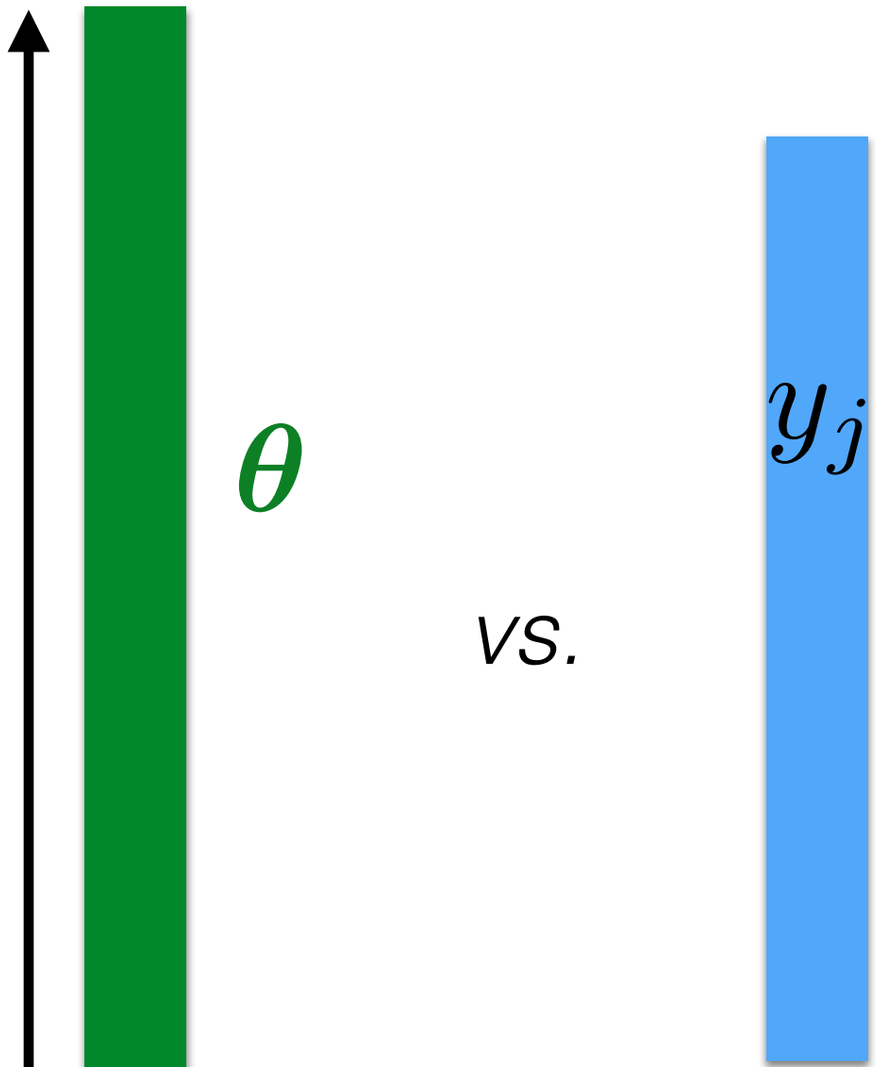
$$\boldsymbol{\theta}_t = F(\boldsymbol{l}(f_{\boldsymbol{\theta}_{t-1}}(x_t), y_t), \boldsymbol{\psi}, \boldsymbol{\theta}_{t-1})$$

From an optimization point of view:

- size of the problem is gone.
- refresh speed might be *very* fast.
- What update rule can we consider to guarantee good approx.?

Example: Regression (Regularized)





$$\min_{\theta,b} \frac{1}{n} \sum_{j=1}^{n} (x_j^T \theta + b - y_j)^2 + \lambda \|\theta\|_q^q$$

Example: Binary Classification *(linear)*

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathbb{R}^p \times \{-1, 1\})^n$$

$$\text{0-1 loss} : l(a, b) = \mathbf{1}_{a \neq b}$$

$$\mathcal{F} = \{f_\theta : x \mapsto \text{sign}(w^T x + b), \theta = (\omega, b) \in \mathbb{R}^{p+1}\}$$

$$\psi(\omega, b) = \frac{1}{2}\|w\|^2$$

$$\min_{\boldsymbol{w},\boldsymbol{b}} \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{(f_{\boldsymbol{w},\boldsymbol{b}}(x_i) \neq y_i)} + \frac{1}{2}\|\boldsymbol{w}\|^2$$

Cleaner optimization setup for ML

$$\{z_1, \ldots, z_n\} \in (\mathcal{X} \times \mathcal{Y})^n$$

$$\{l_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+\}_{\theta \in \Theta}$$

$$\psi : \Theta \to \mathbb{R}_+$$

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} l_{\boldsymbol{\theta}}(z_i) + \psi(\boldsymbol{\theta})$$

Examples

1. Ridge regression, Regression with L1-penalization.
2. Support Vector Machine.
3. Logistic regression.
4. Matrix completion with a nuclear norm regularizer.
5. Multilayer perceptron classifier

Automatic Differentiation

Automatic Differentiation

Automatic differentiation:

software to transform code for one function into code for the derivative of the function.

Automatic differentiation is **not**

numerical (approx) differentiation with perturbations

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right) \qquad \frac{\partial f(\mathbf{x})}{\partial x_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}$$

symbolic differentiation

$$\frac{d}{dx}\left(f(x) + g(x)\right) \rightsquigarrow \frac{d}{dx}f(x) + \frac{d}{dx}g(x)$$

$$\frac{d}{dx}\left(f(x)\,g(x)\right) \rightsquigarrow \left(\frac{d}{dx}f(x)\right) g(x) + f(x)\left(\frac{d}{dx}g(x)\right) \ .$$

Automatic Differentiation

$$l_1 = x$$
$$l_n + 1 = 4l_n(1 - l_n)$$

$$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$$

Source: *Automatic differentiation in machine learning, a survey, Baydin et. al, 2015*

Automatic Differentiation

$l_1 = x$
$l_n + 1 = 4l_n(1 - l_n)$

$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$

Manual Differentiation

$$f'(x) = 128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2)+ 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - 64x(1-2x)^2(1-8x+8x^2)^2 - 256x(1-x)(1-2x)(1-8x+8x^2)^2$$

Source: *Automatic differentiation in machine learning, a survey, Baydin et. al, 2015*

Automatic Differentiation

$l_1 = x$
$l_n + 1 = 4l_n(1 - l_n)$

$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$

Manual Differentiation

$f'(x) = 128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2)+ 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - 64x(1-2x)^2(1- 8x+8x^2)^2 - 256x(1-x)(1-2x)(1-8x+8x^2)^2$

Coding

```
f'(x):
  return 128x(1 - x)(-8 + 16 x)(1 - 2
     x)^2 (1 - 8 x + 8 x^2) + 64 (1 - x)(1
     - 2 x)^2 (1 - 8 x + 8 x^2)^2 - 64x(1 -
     2 x)^2 (1 - 8 x + 8 x^2)^2 - 256x(1 -
     x)(1 - 2 x)(1 - 8 x + 8 x^2)^2
```

f'(x_0) = $f'(x_0)$
Exact

Source: *Automatic differentiation in machine learning, a survey, Baydin et. al, 2015*

Automatic Differentiation

$l_1 = x$
$l_n + 1 = 4l_n(1 - l_n)$

$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$

Manual Differentiation

$f'(x) = 128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2)+ 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - 64x(1-2x)^2(1- 8x+8x^2)^2 - 256x(1-x)(1-2x)(1-8x+8x^2)^2$

Coding

Coding

```
f'(x):
  return 128x(1 - x)(-8 + 16 x)(1 - 2
     x)^2 (1 - 8 x + 8 x^2) + 64 (1 - x)(1
     - 2 x)^2 (1 - 8 x + 8 x^2)^2 - 64x(1 -
     - 8 x + 8 x^2)^2 - 256x(1 -
    )(1 - 8 x + 8 x^2)^2
```

$f'(x_0) = f'(x_0)$
Exact

```
f(x):
   v = x
   for i = 1 to 3
     v = 4v(1 - v)
   return v
```

or, in closed-form,

```
f(x):
   return 64x (1-x) (1-2x)^2 (1-8x+8x^2)^2
```

Source: *Automatic differentiation in machine learning, a survey, Baydin et. al, 2015*

Automatic Differentiation

$l_1 = x$
$l_n + 1 = 4l_n(1 - l_n)$

$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$

Manual Differentiation

$f'(x) = 128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2)+ 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - 64x(1-2x)^2(1- 8x+8x^2)^2 - 256x(1-x)(1-2x)(1-8x+8x^2)^2$

Coding

Coding

```
f(x):
   v = x
   for i = 1 to 3
     v = 4v(1 - v)
   return v
```

or, in closed-form,

```
f(x):
   return 64x (1-x) (1-2x)^2 (1-8x+8x^2)^2
```

Symbolic Differentiation of the Closed-form

```
f'(x):
  return 128x(1 - x)(-8 + 16 x)(1 - 2
     x)^2 (1 - 8 x + 8 x^2) + 64 (1 - x)(1
     - 2 x)^2 (1 - 8 x + 8 x^2)^2 - 64x(1 -
     2 x)^2 (1 - 8 x + 8 x^2)^2 - 256x(1 -
     x)(1 - 2 x)(1 - 8 x + 8 x^2)^2
```

f'(x$_0$) = $f'(x_0)$
Exact

Automatic Differentiation

```
f'(x):
   (v,v') = (x,1)
   for i = 1 to 3
     (v,v') = (4v(1-v), 4v'-8vv')
   return (v,v')
```

f'(x$_0$) = $f'(x_0)$
Exact

Automatic Differentiation

$l_1 = x$
$l_n + 1 = 4l_n(1 - l_n)$

$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$

Manual Differentiation

$f'(x) = 128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2)+ 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - 64x(1-2x)^2(1- 8x+8x^2)^2 - 256x(1-x)(1-2x)(1-8x+8x^2)^2$

Coding

Coding

```
f(x):
   v = x
   for i = 1 to 3
     v = 4v(1 - v)
   return v
```

or, in closed-form,

```
f(x):
   return 64x (1-x) (1-2x)^2 (1-8x+8x^2)^2
```

Symbolic Differentiation of the Closed-form

```
f'(x):
  return 128x(1 - x)(-8 + 16 x)(1 - 2
     x)^2 (1 - 8 x + 8 x^2) + 64 (1 - x)(1
     - 2 x)^2 (1 - 8 x + 8 x^2)^2 - 64x(1 -
     2 x)^2 (1 - 8 x + 8 x^2)^2 - 256x(1 -
     x)(1 - 2 x)(1 - 8 x + 8 x^2)^2
```

f'(x$_0$) $= f'(x_0)$
Exact

Automatic Differentiation

Numerical Differentiation

```
f'(x):
  (v,v') = (x,1)
  for i = 1 to 3
    (v,v') = (4v(1-v), 4v'-8vv')
  return (v,v')
```

f'(x$_0$) $= f'(x_0)$
Exact

```
f'(x):
  return (f(x + h) - f(x)) / h
```

f'(x$_0$) $\approx f'(x_0)$
Approximate

Automatic Differentiation

$l_1 = x$
$l_n + 1 = 4l_n(1 - l_n)$

$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$

Manual Differentiation

$f'(x) = 128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2)+ 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - 64x(1-2x)^2(1- 8x+8x^2)^2 - 256x(1-x)(1-2x)(1-8x+8x^2)^2$

Coding

Coding

```
f(x):
   v = x
   for i = 1 to 3
     v = 4v(1 - v)
   return v
```

or, in closed-form,

```
f(x):
   return 64x (1-x) (1-2x)^2 (1-8x+8x^2)^2
```

Symbolic Differentiation of the Closed-form

```
f'(x):
  return 128x(1 - x)(-8 + 16 x)(1 - 2
     x)^2 (1 - 8 x + 8 x^2) + 64 (1 - x)(1
     - 2 x)^2 (1 - 8 x + 8 x^2)^2 - 64x(1 -
     2 x)^2 (1 - 8 x + 8 x^2)^2 - 256x(1 -
     x)(1 - 2 x)(1 - 8 x + 8 x^2)^2
```

f'(x$_0$) $= f'(x_0)$
Exact

Automatic Differentiation

Numerical Differentiation

```
f'(x):
  (v,v') = (x,1)
  for i = 1 to 3
    (v,v') = (4v(1-v), 4v'-8vv')
  return (v,v')
```

f'(x$_0$) $= f'(x_0)$
Exact

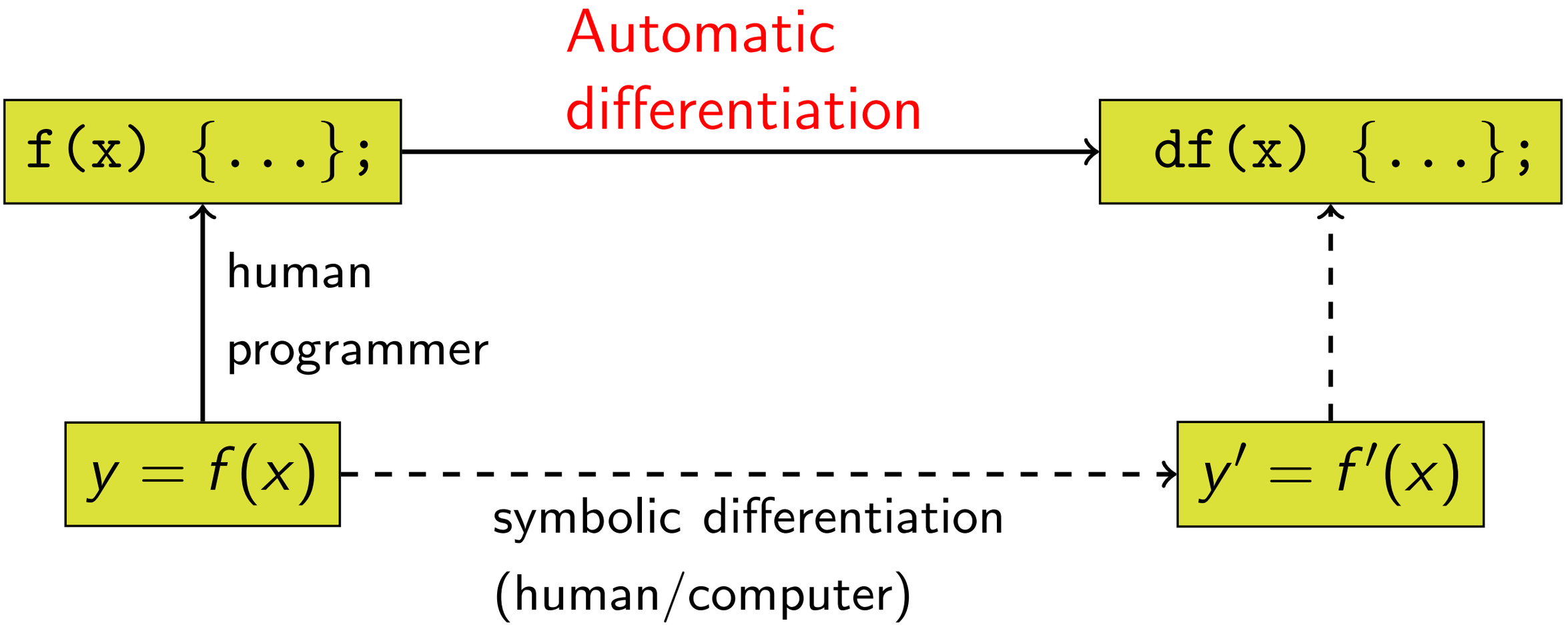
```
f'(x):
  return (f(x + h) - f(x)) / h
```

f'(x$_0$) $\approx f'(x_0)$
Approximate

Automatic Differentiation

n	l_n	$\frac{d}{dx} l_n$	$\frac{d}{dx} l_n$ (Optimized)
1	x	1	1
2	$4x(1-x)$	$4(1-x)-4x$	$4-8x$
3	$16x(1-x)(1-2x)^2$	$16(1-x)(1-2x)^2-16x(1-2x)^2-64x(1-x)(1-2x)$	$16(1-10x+24x^2-16x^3)$
4	$64x(1-x)(1-2x)^2(1-8x+8x^2)^2$	$128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2)+64(1-x)(1-2x)^2(1-8x+8x^2)^2-64x(1-2x)^2(1-8x+8x^2)^2-256x(1-x)(1-2x)(1-8x+8x^2)^2$	$64(1-42x+504x^2-2640x^3+7040x^4-9984x^5+7168x^6-2048x^7)$

Automatic Differentiation



Berland, Havard. “Automatic Differentiation (Lecture).” *Department of Mathematical Sciences, NTNU*, 2006, www.pvv.ntnu.no/~berland/resources/autodiff-triallecture.pdf.

Dual Numbers

Extend all numbers by adding a second component,

$$x \mapsto x + \dot{x}\mathbf{d}$$

- $\mathbf{d}$ is just a symbol distinguishing the second component,
- analogous to the imaginary unit $\mathbf{i} = \sqrt{-1}$.
- But, let $\mathbf{d}^2 = 0$, as opposed to $\mathbf{i}^2 = -1$.

Arithmetic on dual numbers:

$$(x + \dot{x}\mathbf{d}) + (y + \dot{y}\mathbf{d}) = x + y + (\dot{x} + \dot{y})\mathbf{d}$$

$$\begin{aligned}(x + \dot{x}\mathbf{d}) \cdot (y + \dot{y}\mathbf{d}) &= xy + x\dot{y}\mathbf{d} + \dot{x}y\mathbf{d} + \overbrace{\dot{x}\dot{y}\mathbf{d}^2}^{=0} \\ &= xy + (x\dot{y} + \dot{x}y)\mathbf{d}\end{aligned}$$

$$-(x + \dot{x}\mathbf{d}) = -x - \dot{x}\mathbf{d}, \qquad \frac{1}{x + \dot{x}\mathbf{d}} = \frac{1}{x} - \frac{\dot{x}}{x^2}\mathbf{d} \quad (x \neq 0)$$

Berland, Havard. “Automatic Differentiation (Lecture).” *Department of Mathematical Sciences, NTNU*, 2006, www.pvv.ntnu.no/~berland/resources/autodiff-triallecture.pdf.

Polynomials in Dual Numbers

Let

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n$$

and extend x to a dual number $x + \dot{x}\mathbf{d}$.
Then,

$$\begin{aligned} P(x + \dot{x}\mathbf{d}) &= p_0 + p_1(x + \dot{x}\mathbf{d}) + \cdots + p_n(x + \dot{x}\mathbf{d})^n \\ &= p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n \\ &\quad + p_1 \dot{x}\mathbf{d} + 2p_2 x\dot{x}\mathbf{d} + \cdots + np_n x^{n-1}\dot{x}\mathbf{d} \\ &= P(x) + P'(x)\dot{x}\mathbf{d} \end{aligned}$$

- $\dot{x}$ may be chosen arbitrarily, so choose $\dot{x} = 1$ (currently).
- *The second component is the derivative of $P(x)$ at x*

Berland, Havard. “Automatic Differentiation (Lecture).” *Department of Mathematical Sciences, NTNU*, 2006, www.pvv.ntnu.no/~berland/resources/autodiff-triallecture.pdf.

More Arithmetic

Similarly, one may derive

$$\sin(x + \dot{x}\mathbf{d}) = \sin(x) + \cos(x)\,\dot{x}\mathbf{d}$$
$$\cos(x + \dot{x}\mathbf{d}) = \cos(x) - \sin(x)\,\dot{x}\mathbf{d}$$
$$e^{(x+\dot{x}\mathbf{d})} = e^{x} + e^{x}\dot{x}\mathbf{d}$$
$$\log(x + \dot{x}\mathbf{d}) = \log(x) + \frac{\dot{x}}{x}\mathbf{d} \quad x \neq 0$$
$$\sqrt{x + \dot{x}\mathbf{d}} = \sqrt{x} + \frac{\dot{x}}{2\sqrt{x}}\mathbf{d} \quad x \neq 0$$

Berland, Havard. “Automatic Differentiation (Lecture).” *Department of Mathematical Sciences, NTNU*, 2006, www.pvv.ntnu.no/~berland/resources/autodiff-triallecture.pdf.

More Arithmetic

Derived from dual numbers:

A function applied on a dual number will return its *derivative* in the second/dual component.

We can extend to functions of many variables by introducing more dual components:

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

extends to

$$f(x_1 + \dot{x}_1 \mathbf{d}_1, x_2 + \dot{x}_2 \mathbf{d}_2) =$$
$$(x_1 + \dot{x}_1 \mathbf{d}_1)(x_2 + \dot{x}_2 \mathbf{d}_2) + \sin(x_1 + \dot{x}_1 \mathbf{d}_1) =$$
$$x_1 x_2 + (x_2 + \cos(x_1))\dot{x}_1 \mathbf{d}_1 + x_1 \dot{x}_2 \mathbf{d}_2$$

where $\mathbf{d}_i \mathbf{d}_j = 0$.

Berland, Havard. “Automatic Differentiation (Lecture).” *Department of Mathematical Sciences, NTNU*, 2006, www.pvv.ntnu.no/~berland/resources/autodiff-triallecture.pdf.

Forward Pass

Computer code for $f(x_1, x_2) = x_1 x_2 + \sin(x_1)$ might read

Original program

$w_1 = x_1$
$w_2 = x_2$
$w_3 = w_1 w_2$
$w_4 = \sin(w_1)$
$w_5 = w_3 + w_4$

Dual program

$\dot{w}_1 = 0$
$\dot{w}_2 = 1$
$\dot{w}_3 = \dot{w}_1 w_2 + w_1 \dot{w}_2 = 0 \cdot x_2 + x_1 \cdot 1 = x_1$
$\dot{w}_4 = \cos(w_1)\dot{w}_1 = \cos(x_1) \cdot 0 = 0$
$\dot{w}_5 = \dot{w}_3 + \dot{w}_4 = x_1 + 0 = x_1$

and

$$\frac{\partial f}{\partial x_2} = x_1$$

The chain rule

$$\frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial w_5}\frac{\partial w_5}{\partial w_3}\frac{\partial w_3}{\partial w_2}\frac{\partial w_2}{\partial x_2}$$

ensures that we can *propagate* the dual components throughout the computation.

Code Modification

function.c

```
double  f(double x1, double x2) {
  double w3, w4, w5;
  w3 = x1 * x2;

  w4 = sin(x1);

  w5 = w3 + w4;


  return w5;
}
```

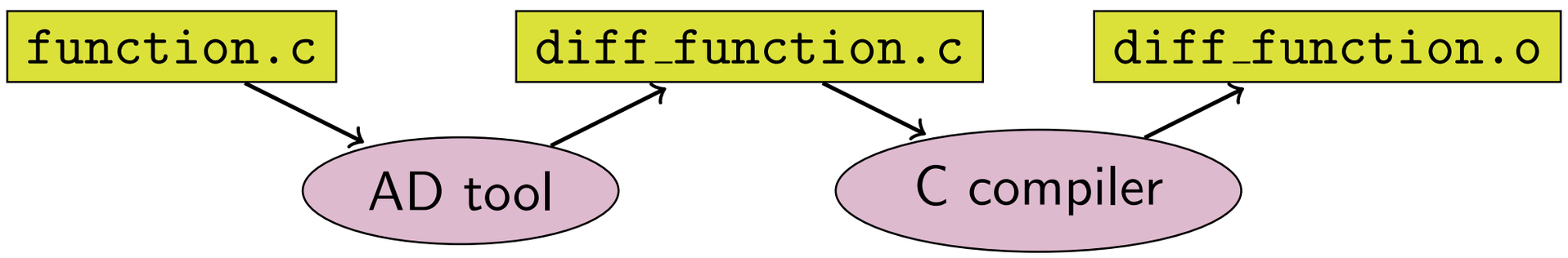
function.c

Source: Havard Berland, NTNU

Code Modification

diff_function.c

```
double* f(double x1, double x2, double dx1, double dx2) {
  double w3, w4, w5, dw3, dw4, dw5, df[2];
  w3 = x1 * x2;
  dw3 = dx1 * x2 + x1 * dx2;
  w4 = sin(x1);
  dw4 = cos(x1) * dx1;
  w5 = w3 + w4;
  dw5 = dw3 + dw4;
  df[0] = w5;
  df[1] = dw5;
  return df;
}
```

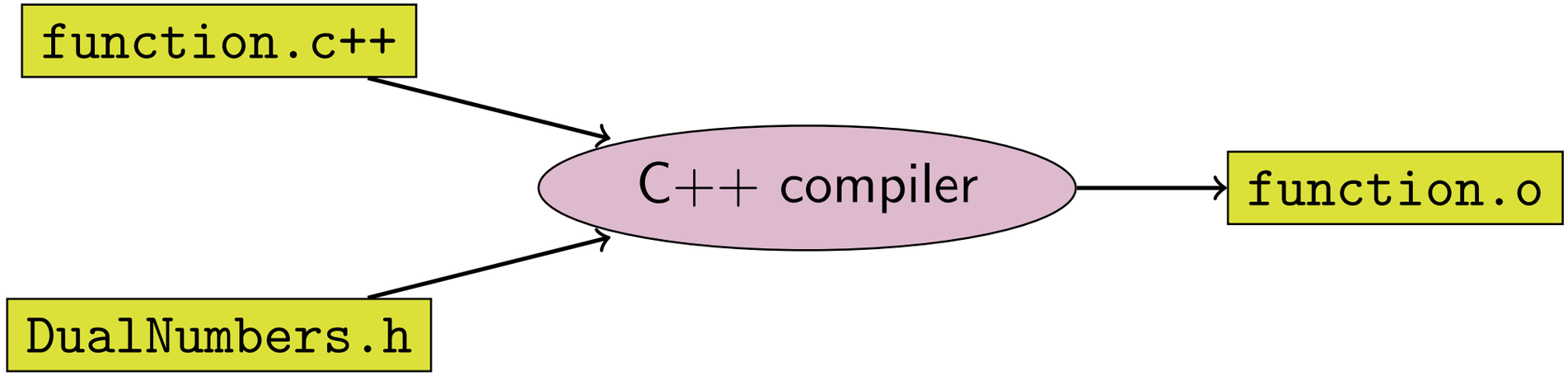


Source: Havard Berland, NTNU

Code Modification

function.c++

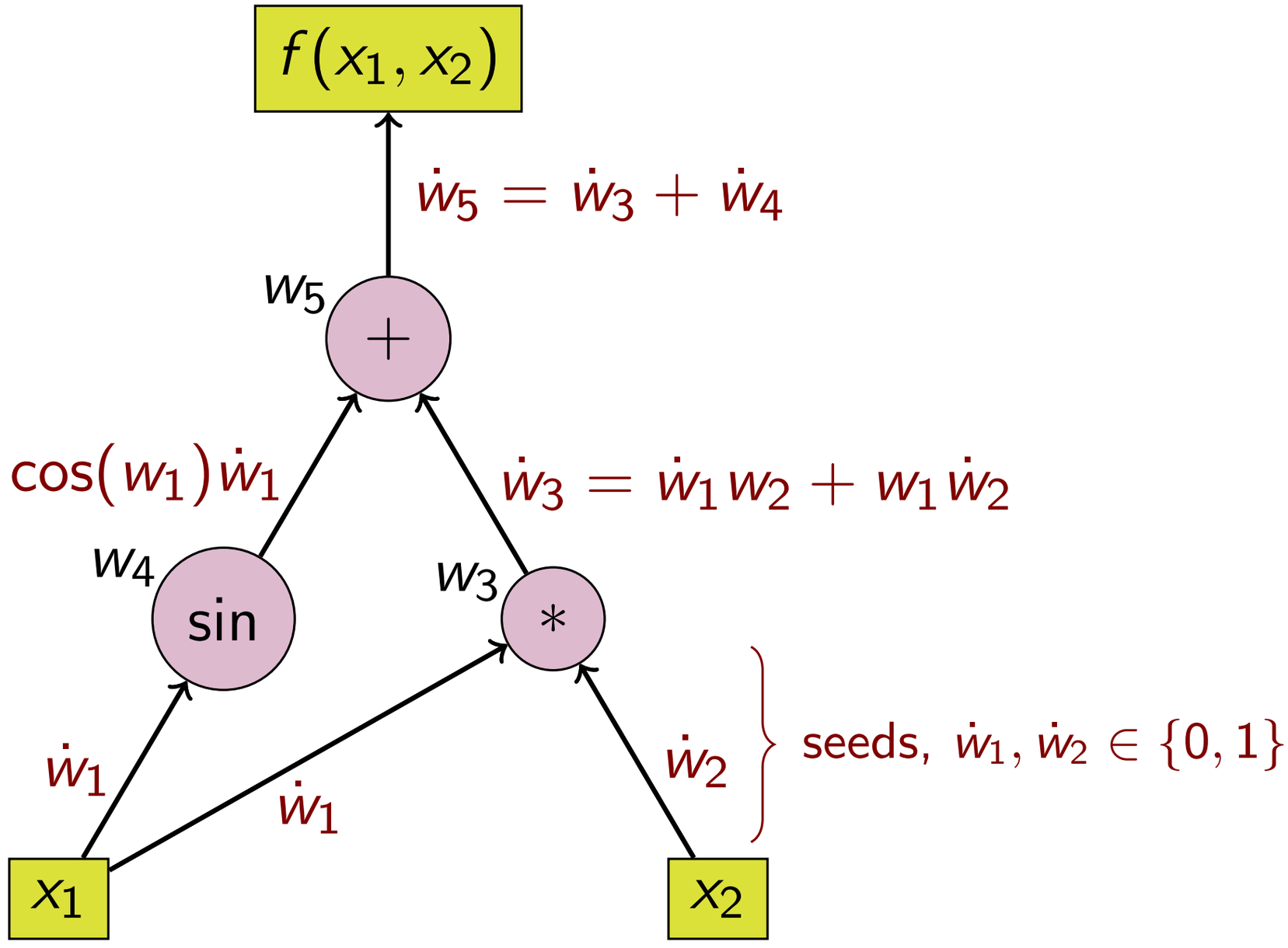
```
Number f(Number x1, Number x2) {
  w3 = x1 * x2;
  w4 = sin(x1);
  w5 = w3 + w4;
  return w5;
}
```



Source: Havard Berland, NTNU

Forward Pass

Forward propagation of derivative values

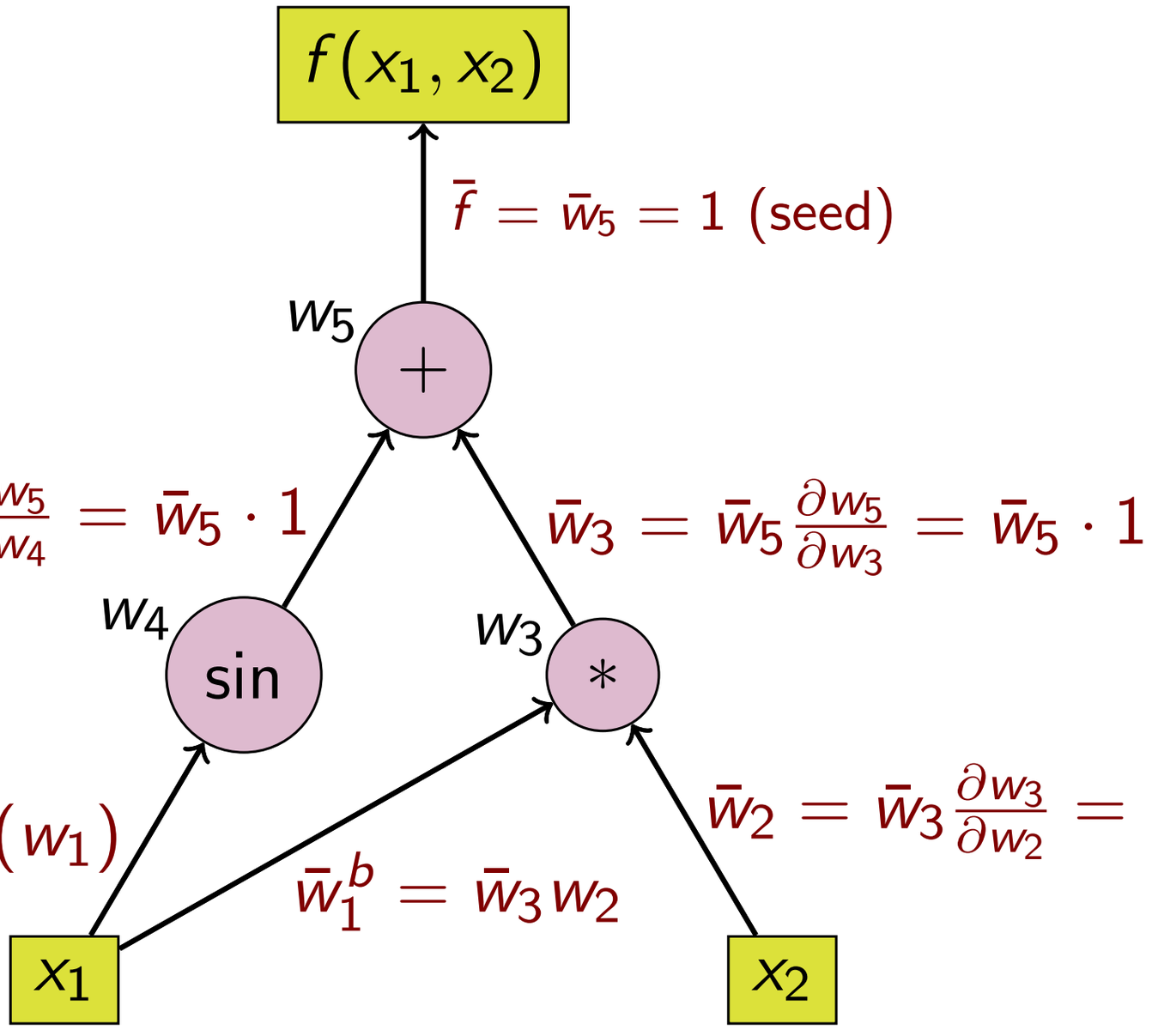


$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_1}\frac{\partial w_1}{\partial x} = \frac{\partial y}{\partial w_1}\left(\frac{\partial w_1}{\partial w_2}\frac{\partial w_2}{\partial x}\right) = \frac{\partial y}{\partial w_1}\left(\frac{\partial w_1}{\partial w_2}\left(\frac{\partial w_2}{\partial w_3}\frac{\partial w_3}{\partial x}\right)\right) = \cdots$$

Source: Havard Berland, NTNU

Backward Pass

Backward propagation of derivative values



$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_1}\frac{\partial w_1}{\partial x} = \left(\frac{\partial y}{\partial w_2}\frac{\partial w_2}{\partial w_1}\right)\frac{\partial w_1}{\partial x} = \left(\left(\frac{\partial y}{\partial w_3}\frac{\partial w_3}{\partial w_2}\right)\frac{\partial w_2}{\partial w_1}\right)\frac{\partial w_1}{\partial x} = \cdots$$

Source: Havard Berland, NTNU

Automatic Differentiation

Table 1: Forward AD example, with $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ at $(x_1, x_2) = (2, 5)$ and setting $\dot{x}_1 = 1$ to compute $\partial y / \partial x_1$.

Forward evaluation trace			Forward derivative trace		
v_{-1}	$= x_1$	$= 2$	$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
v_0	$= x_2$	$= 5$	$\dot{v}_0$	$= \dot{x}_2$	$= 0$
v_1	$= \ln v_{-1}$	$= \ln 2$	$\dot{v}_1$	$= \dot{v}_{-1} / v_{-1}$	$= 1/2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$	$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + v_{-1} \times \dot{v}_0$	$= 1 \times 5 + 2 \times 0$
v_3	$= \sin v_0$	$= \sin 5$	$\dot{v}_3$	$= \cos v_0 \times \dot{v}_0$	$= \cos 5 \times 0$
v_4	$= v_1 + v_2$	$= 0.6931 + 10$	$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
v_5	$= v_4 - v_3$	$= 10.6931 + 0.9589$	$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
y	$= v_5$	$= 11.6521$	$\dot{y}$	$= \dot{v}_5$	$= 5.5$

Automatic Differentiation

Table 2: Reverse AD example, with $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ at $(x_1, x_2) = (2, 5)$. Setting $\bar{y} = 1$, $\partial y/\partial x_1$ and $\partial y/\partial x_2$ are computed in one reverse sweep.

Forward evaluation trace

v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.6931 + 10$
v_5	$= v_4 - v_3$	$= 10.6931 + 0.9589$
y	$= v_5$	$= 11.6521$

Reverse adjoint trace

$\bar{x}_1$	$= \bar{v}_{-1}$		$= 5.5$
$\bar{x}_2$	$= \bar{v}_0$		$= 1.7163$
$\bar{v}_{-1}$	$= \bar{v}_{-1} + \bar{v}_1(\partial v_1/\partial v_{-1})$	$= \bar{v}_{-1} + \bar{v}_1/v_{-1}$	$= 5.5$
$\bar{v}_0$	$= \bar{v}_0 + \bar{v}_2(\partial v_2/\partial v_0)$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.7163$
$\bar{v}_{-1}$	$= \bar{v}_2(\partial v_2/\partial v_{-1})$	$= \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0$	$= \bar{v}_3(\partial v_3/\partial v_0)$	$= \bar{v}_3 \times \cos v_0$	$= -0.2837$
$\bar{v}_2$	$= \bar{v}_4(\partial v_4/\partial v_2)$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1$	$= \bar{v}_4(\partial v_4/\partial v_1)$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3$	$= \bar{v}_5(\partial v_5/\partial v_3)$	$= \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4$	$= \bar{v}_5(\partial v_5/\partial v_4)$	$= \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5$	$= \bar{y}$		$= 1$

$$\bar{v}_i = \frac{\partial y_j}{\partial v_i}$$

Automatic Differentiation

Given $F: \mathbf{R}^n \mapsto \mathbf{R}^m$ and the Jacobian $J = DF(\mathbf{x}) \in \mathbf{R}^{m \times n}$.

$$J = DF(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

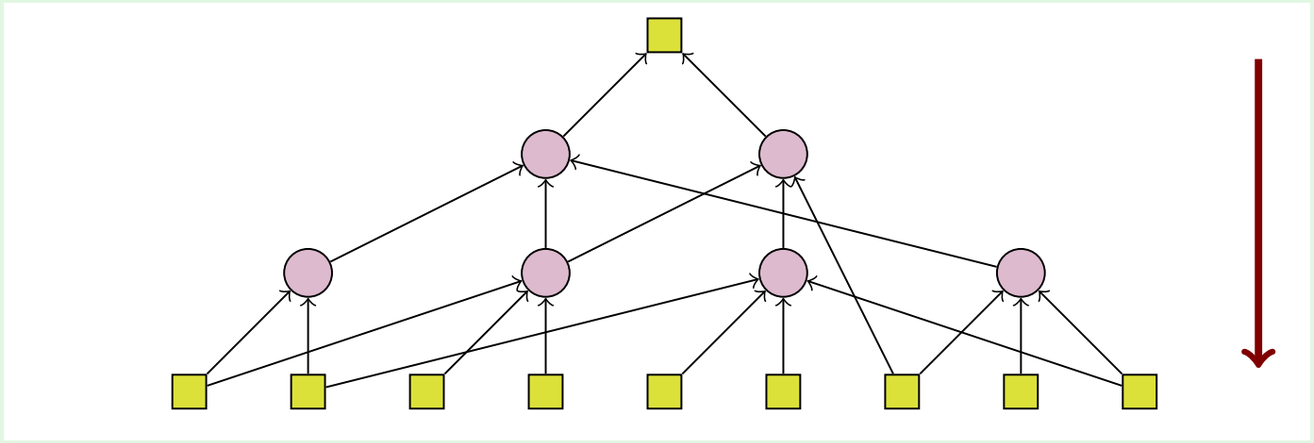
Backward sweep can compute one line

Forward sweep can compute one column

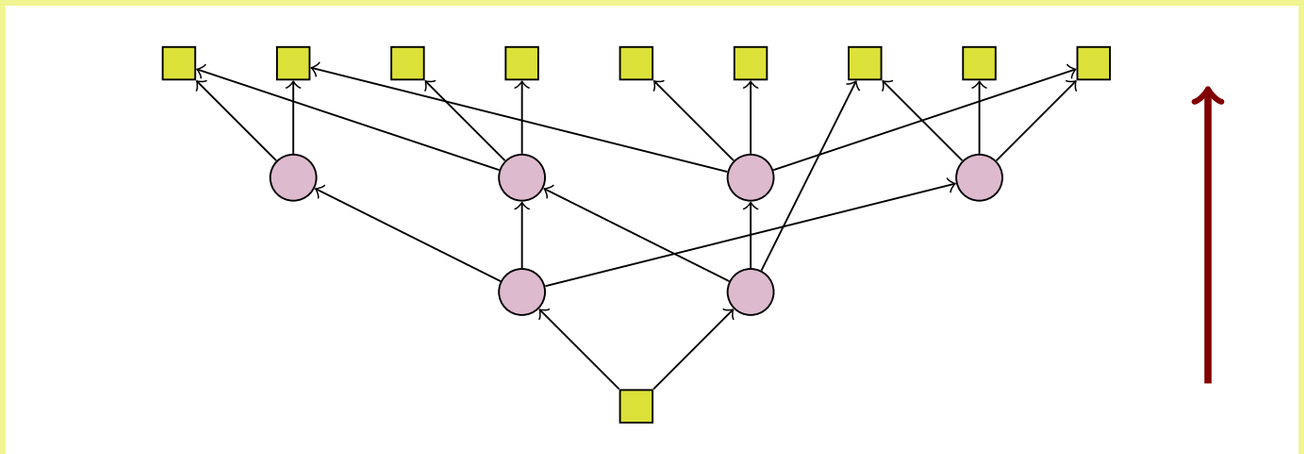
Source: Havard Berland, NTNU

Automatic Differentiation

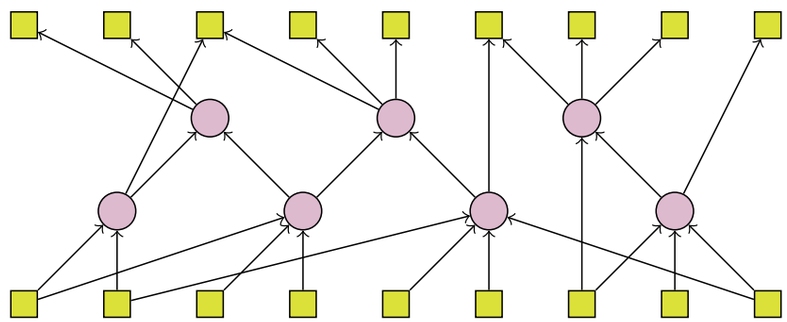
Reverse mode suitable for $F : \mathbb{R}^p \mapsto \mathbb{R}$



Forward mode suitable for $F : \mathbb{R} \mapsto \mathbb{R}^p$



$F : \mathbb{R}^d \mapsto \mathbb{R}^p$?



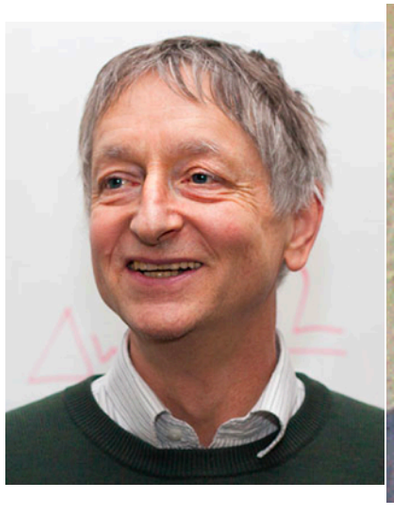
?

Source: Havard Berland, NTNU

Automatic Differentiation

Reverse mode accumulation was presented in master thesis [S. Linnainmaa] in 1970: *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*, printed in journal in '76.

Proposed lated in 1986 in the context of neural networks in a *Nature* paper



Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input-output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

$$x_j = \sum_i y_i w_{ji} \qquad (1)$$

Units can be given biases by introducing an extra input to each

Automatic Differentiation in ML

Currently: golden age of **FREE** AD frameworks

Theano (precursor, deprecated)

Tensorflow, JAX

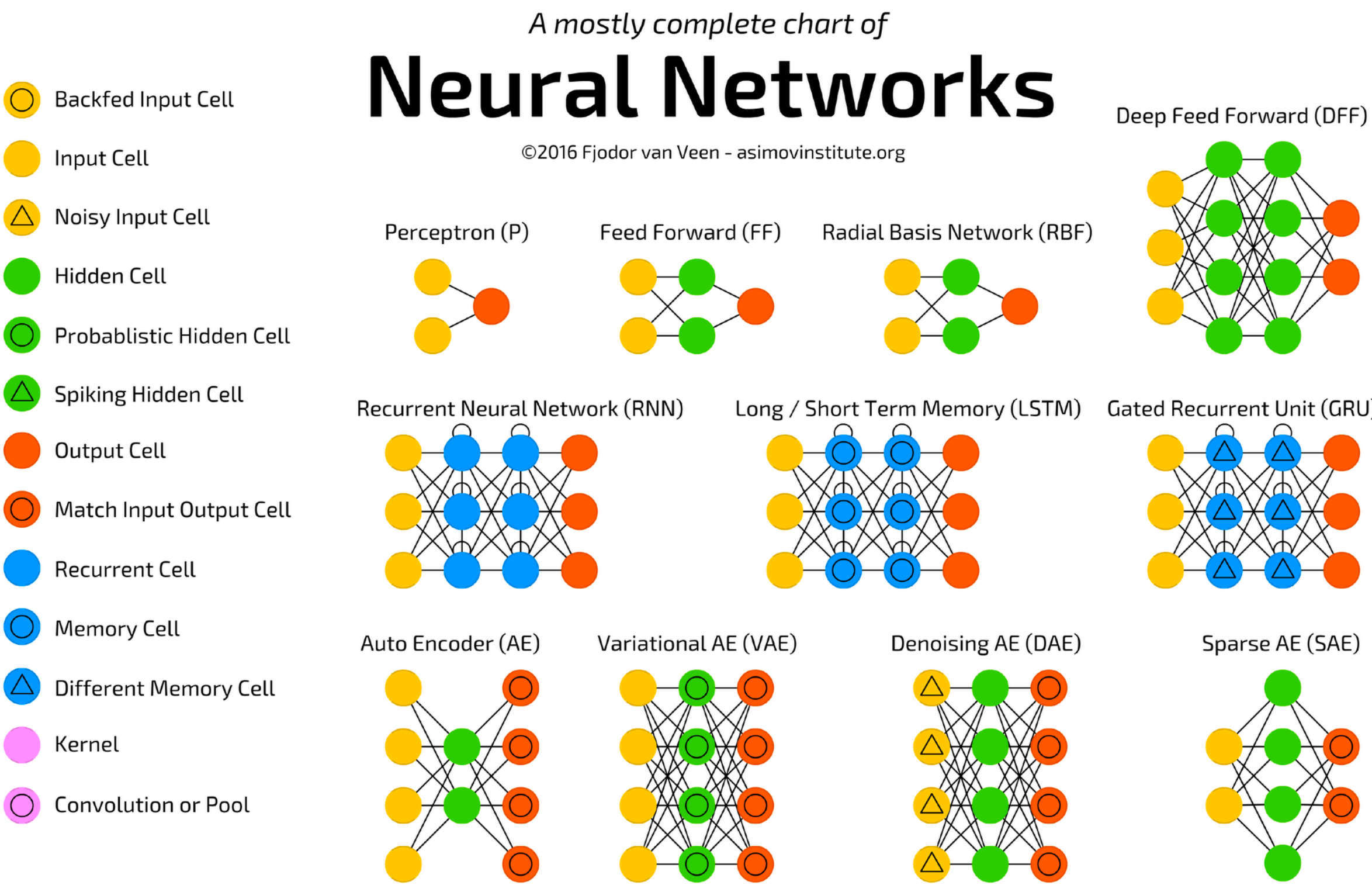
Pytorch

Schedule

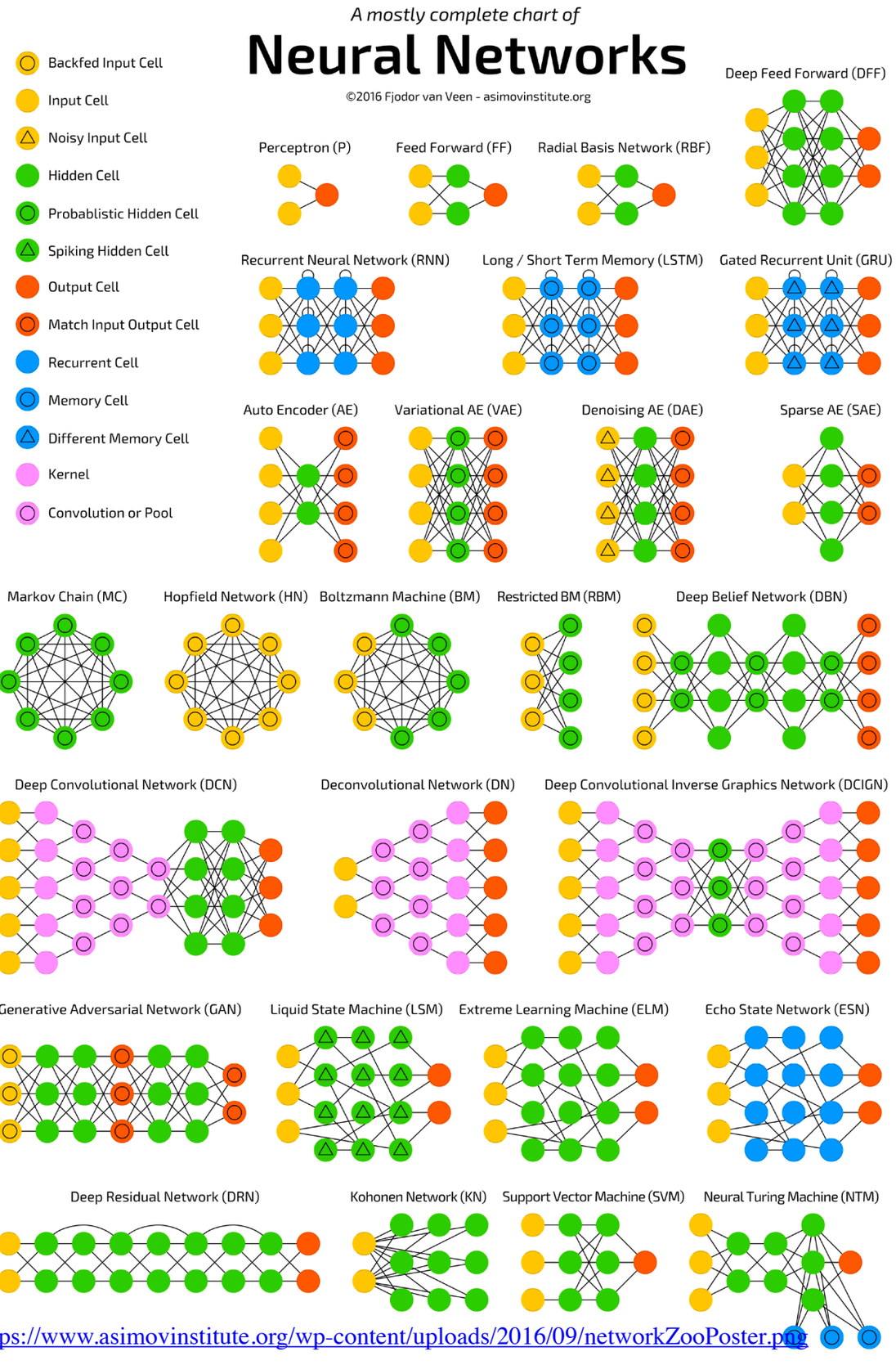
1. Introduction, Backpropagation
   Empirical risk minimization (R-ERM), specificities of deep learning, automatic differentiation.
2. Modeling blocks: convolution, recursion.
   Architecture of DL, building blocks from signal processing, mechanisms to handle varying-length inputs
3. Non-convex optimization
4. Unsupervised problems: GANs / (V)AEs.

Building Blocks

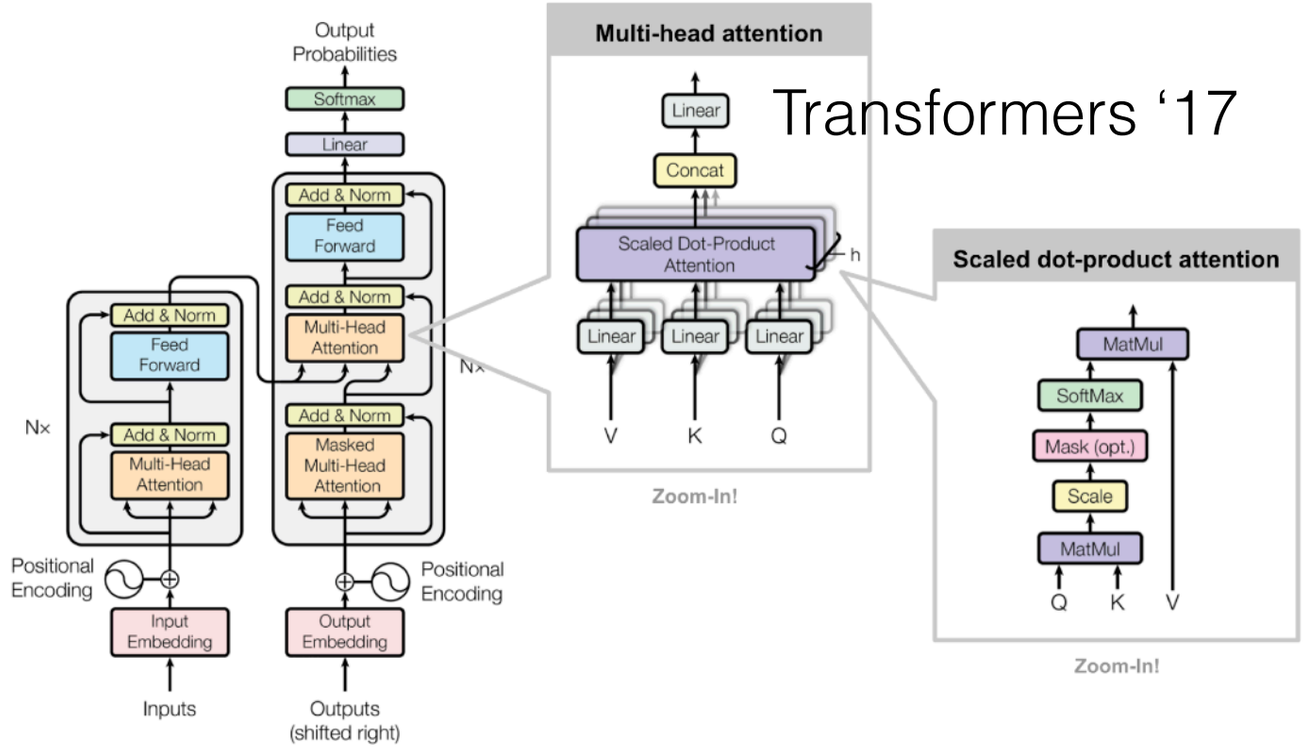
DL Architectures

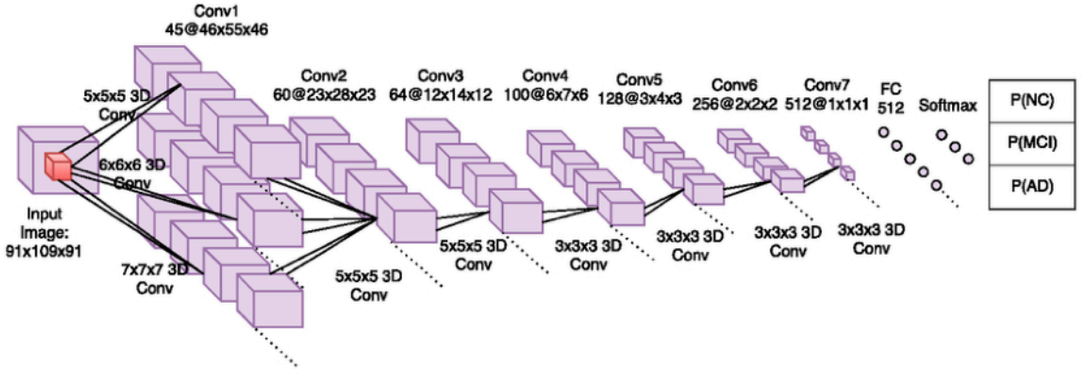


DL Architectures



https://www.asimovinstitute.org/wp-content/uploads/2016/09/networkZooPoster.png



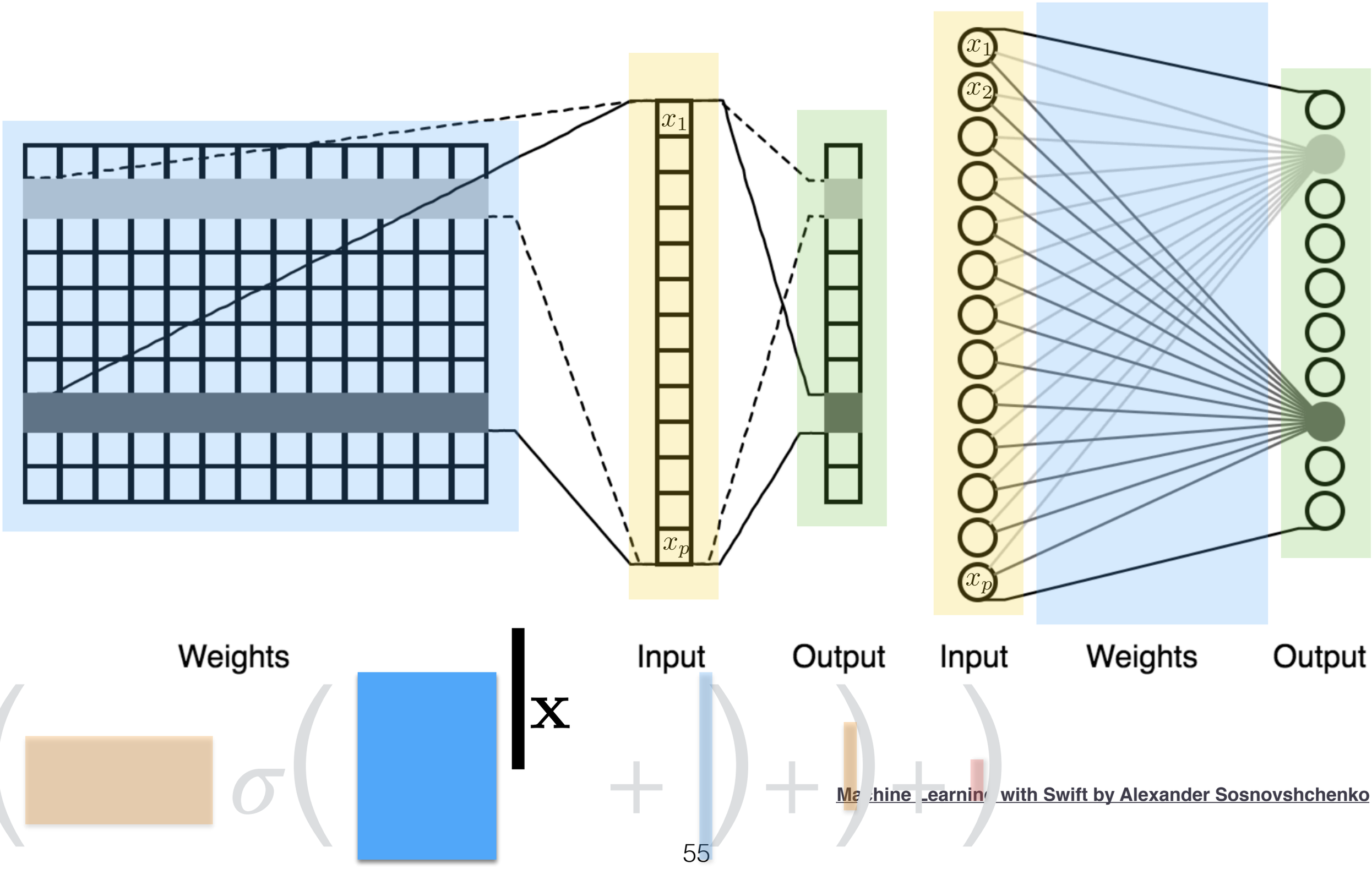


Building Block

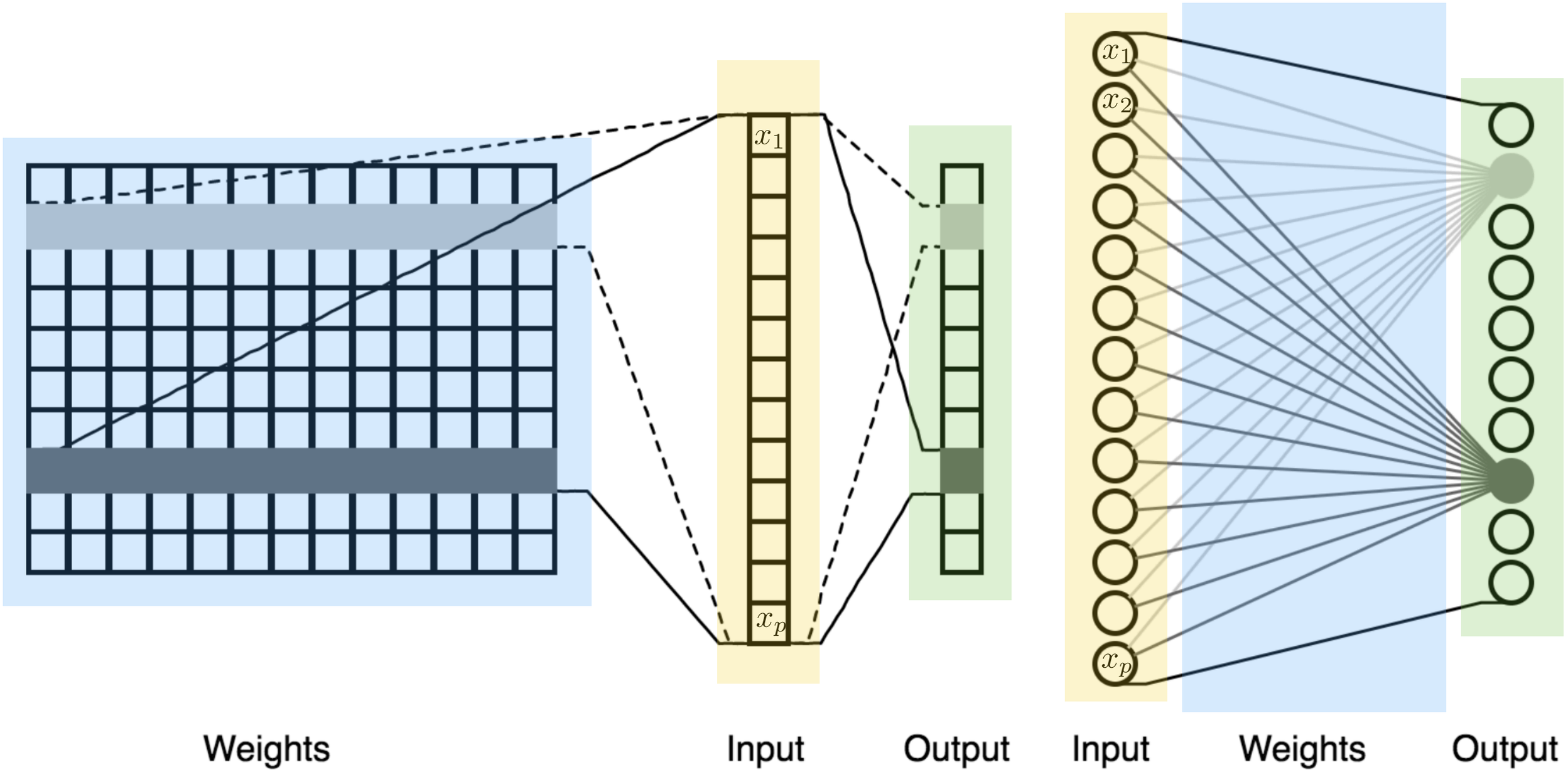
1. Vectors of features, no spatial prior knowledge
2. Vectors of spatial features: Convolutions
3. Varying-length inputs using recursion

Extensions to other types of inputs: Graphs

Vectors : Matrix Products Will Do

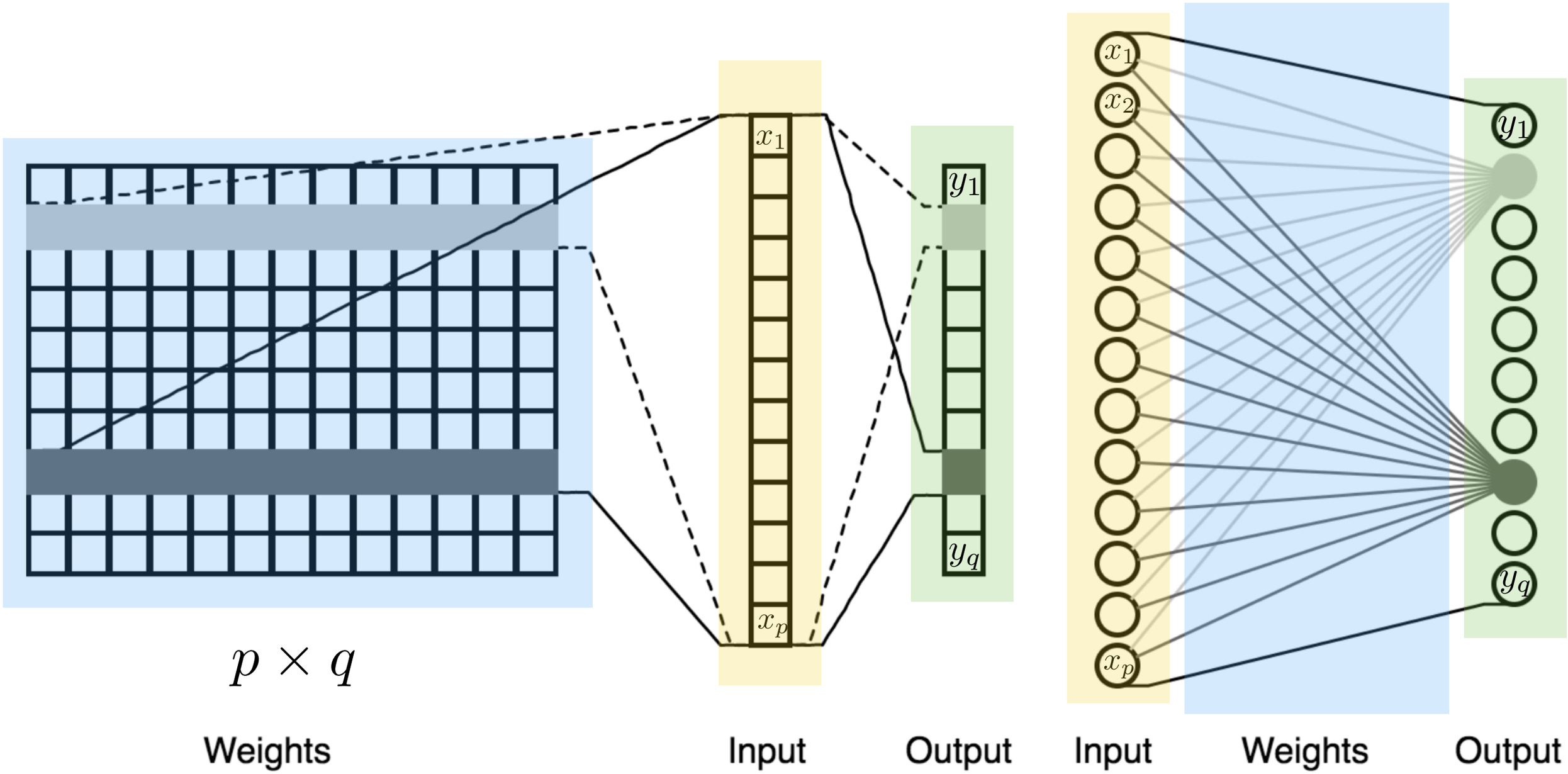


Vectors : Matrix Products Will Do



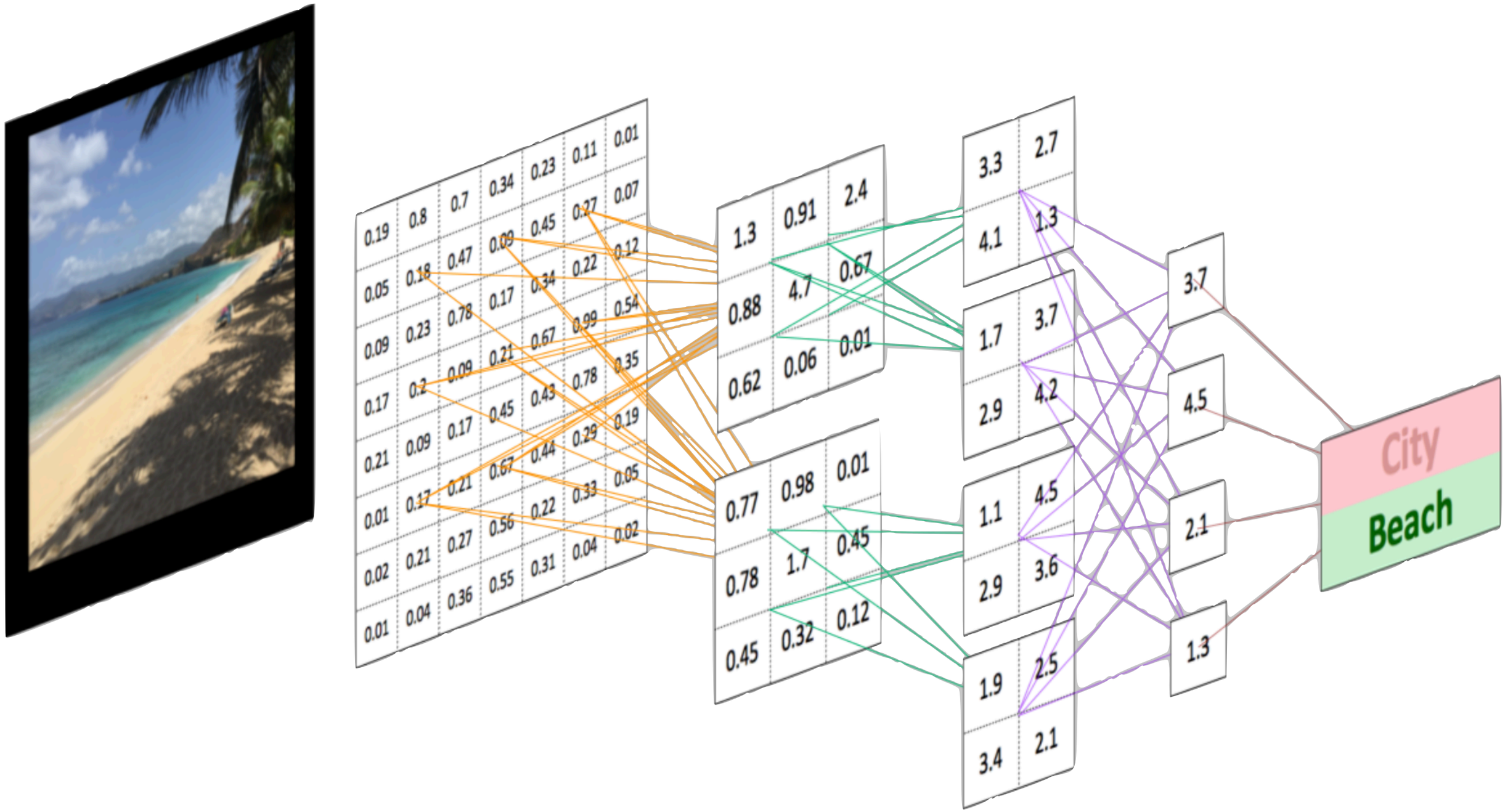
Machine Learning with Swift by Alexander Sosnovshchenko

Vectors : Matrix Products Will Do



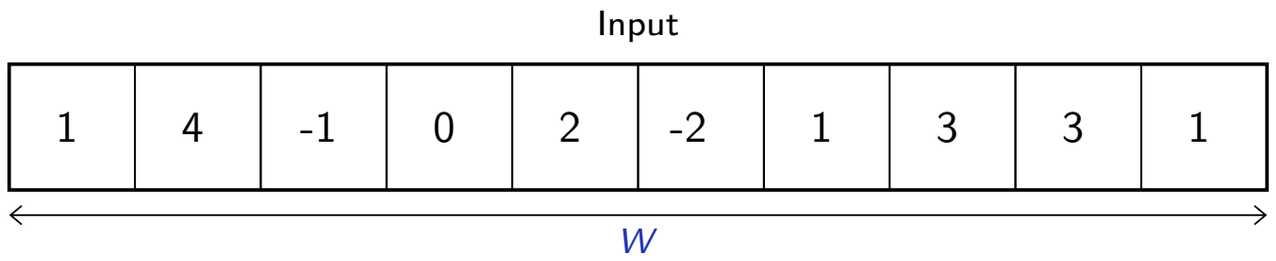
Problematic for large p, q

Convolutional Neural Networks



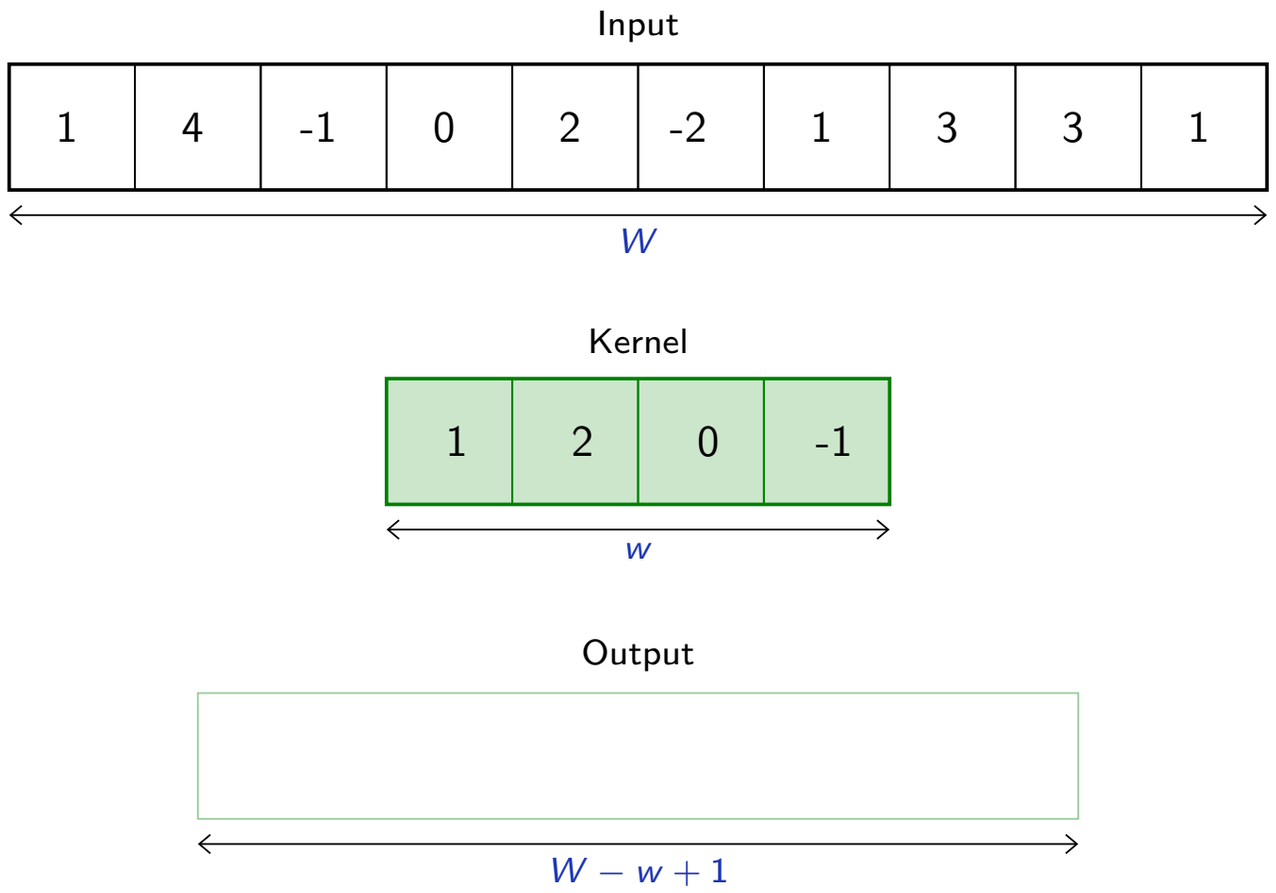
convolution = linear, local, used hierarchically

Convolution 1D



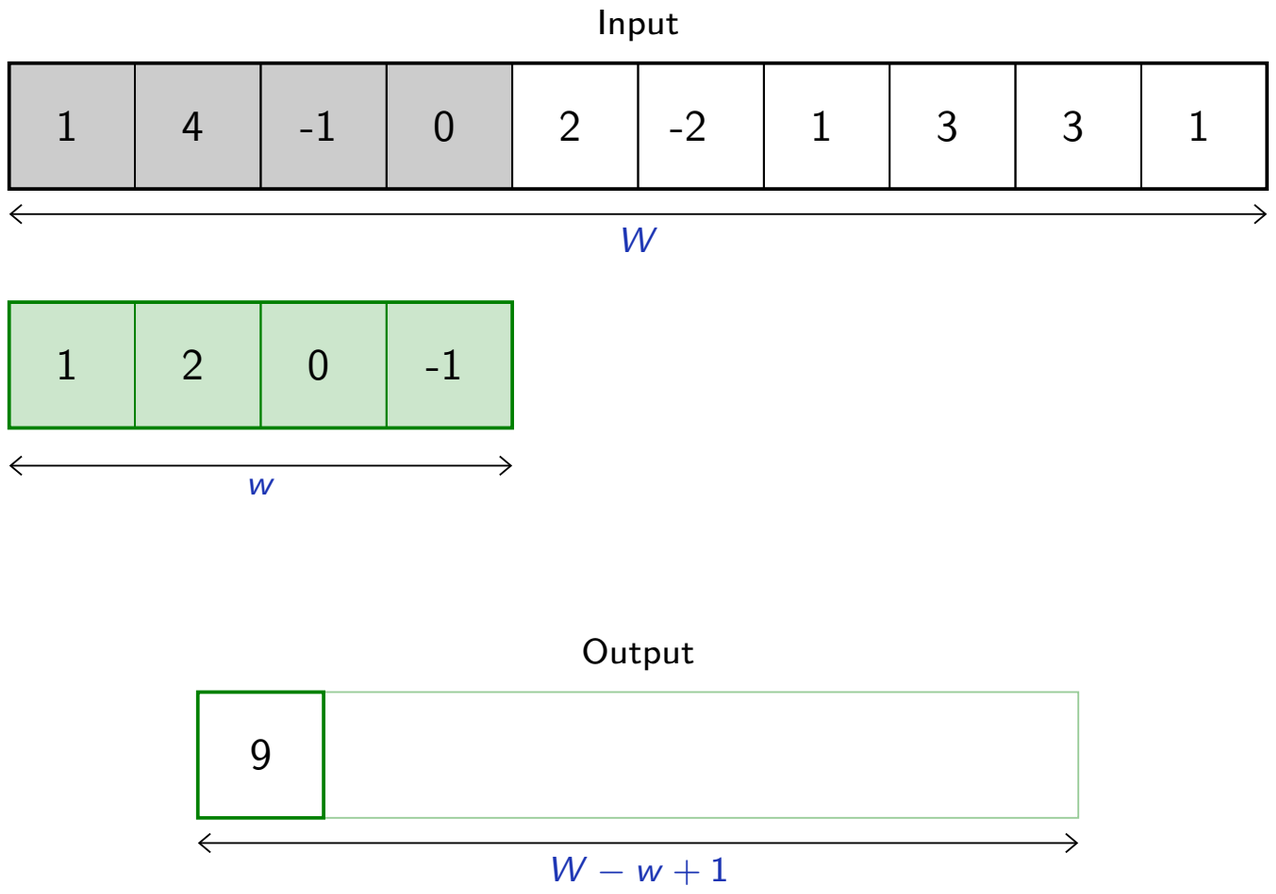
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 1D



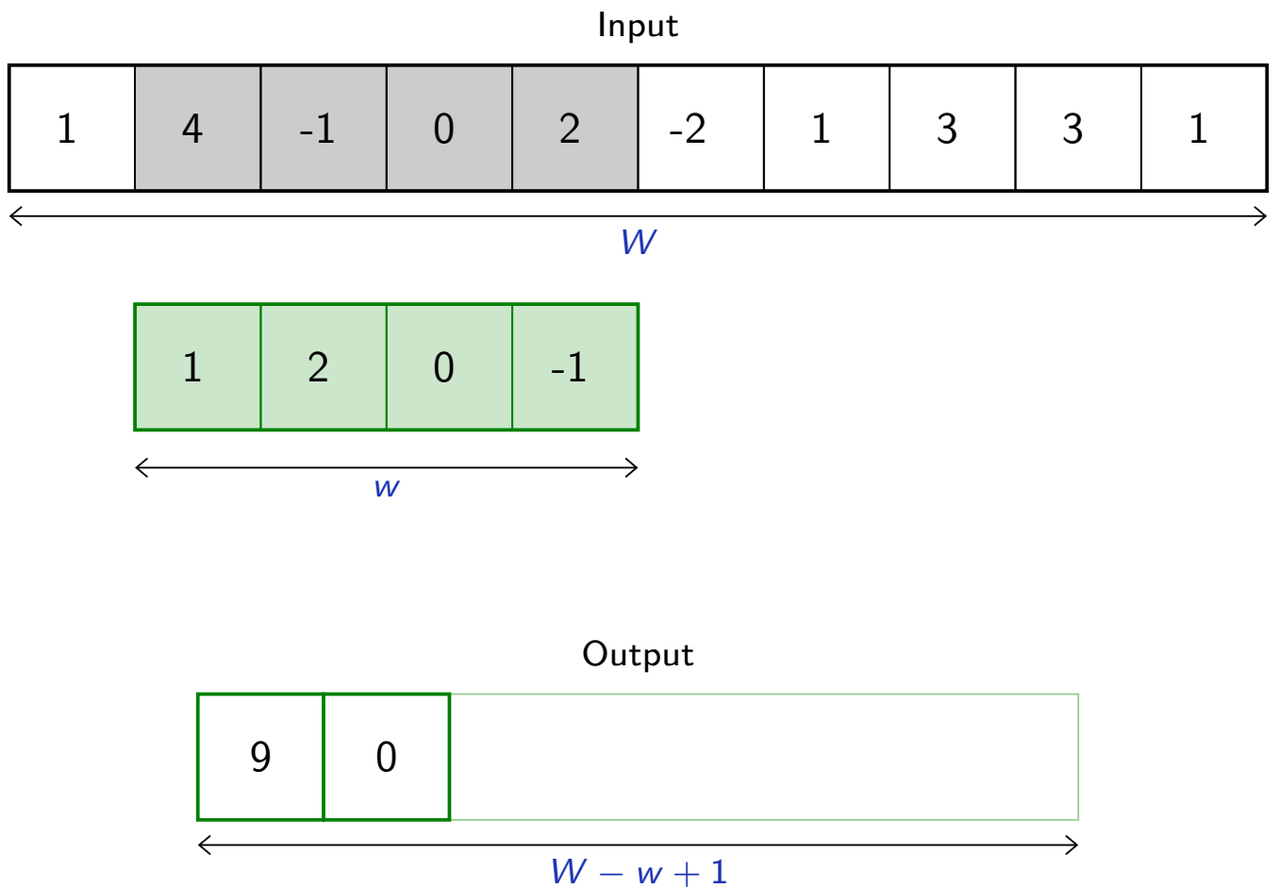
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 1D



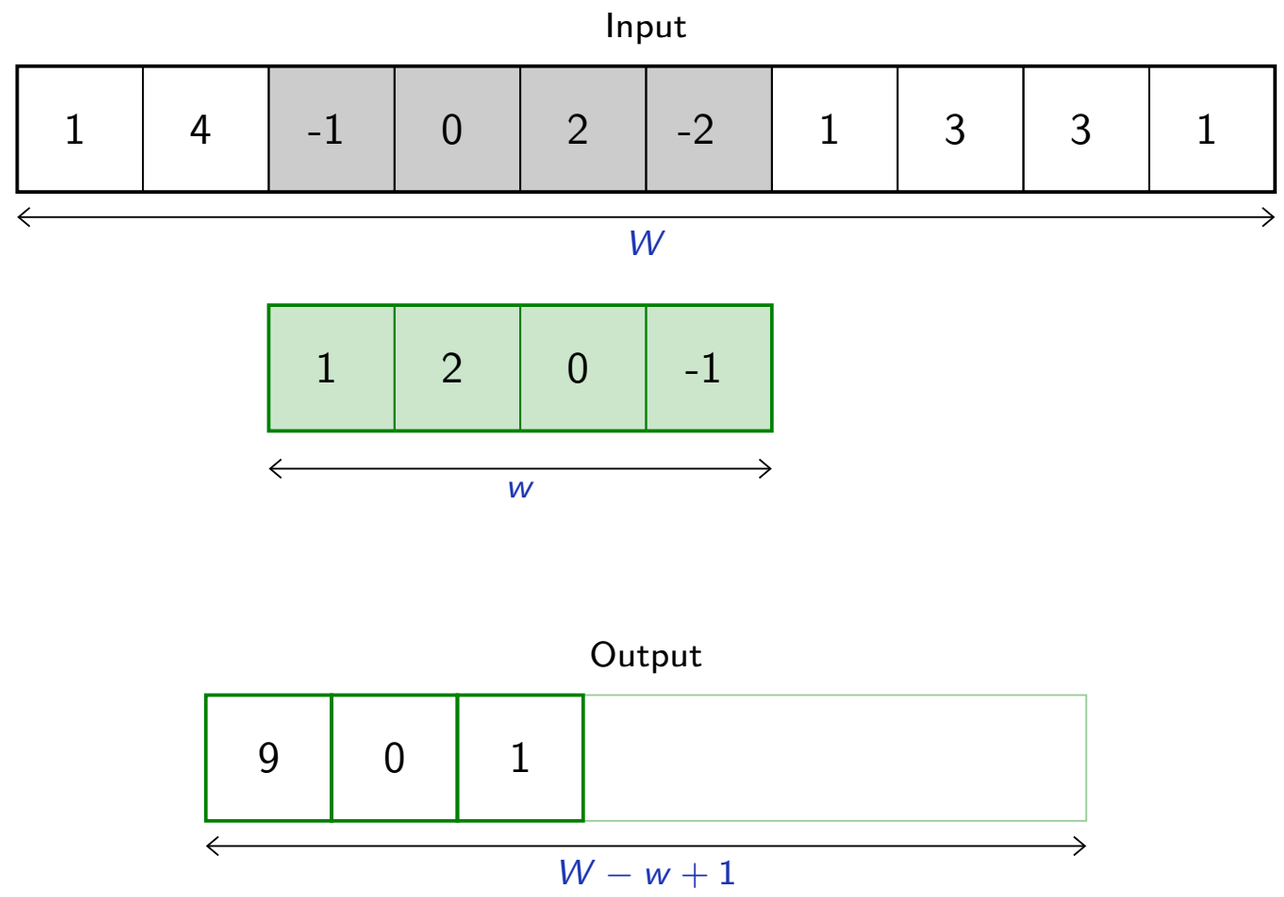
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 1D



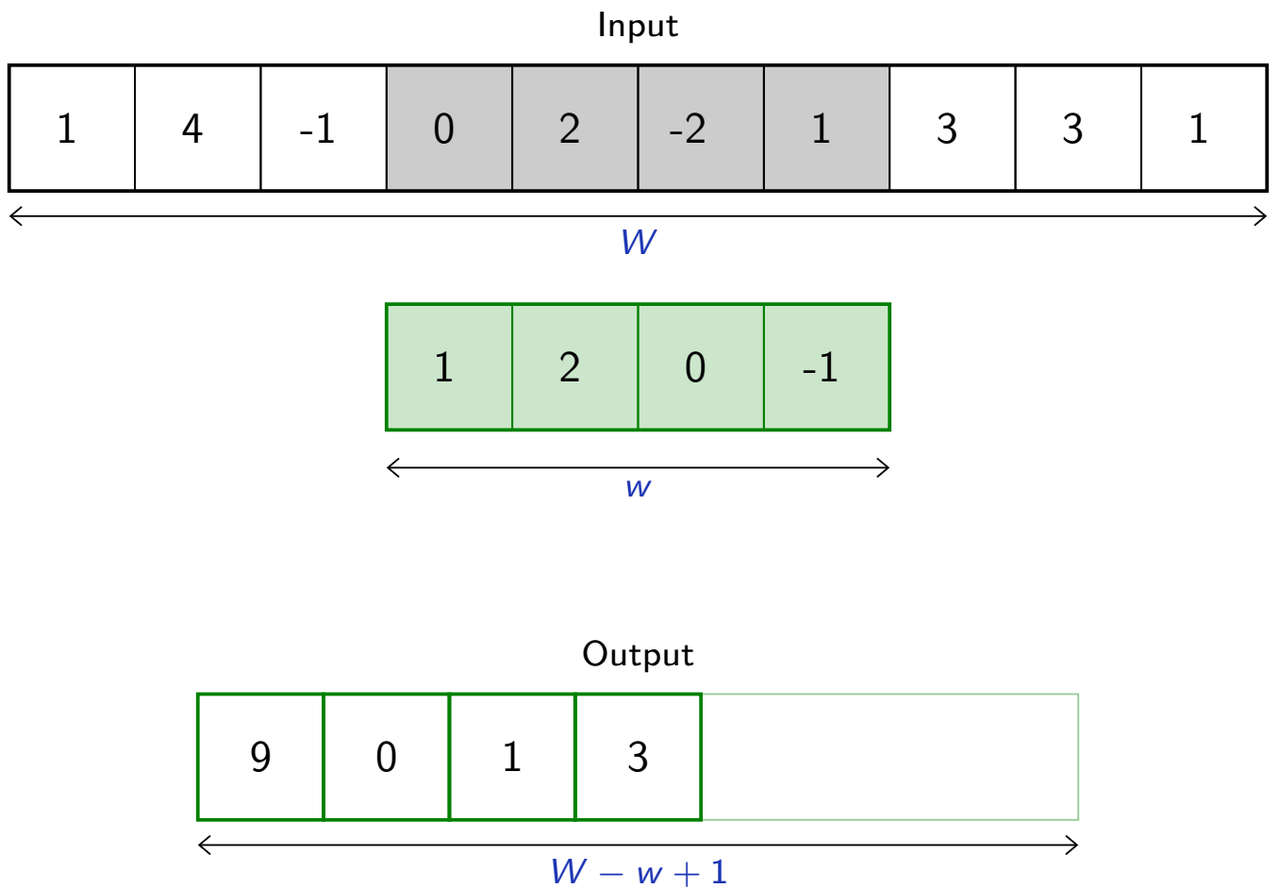
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 1D



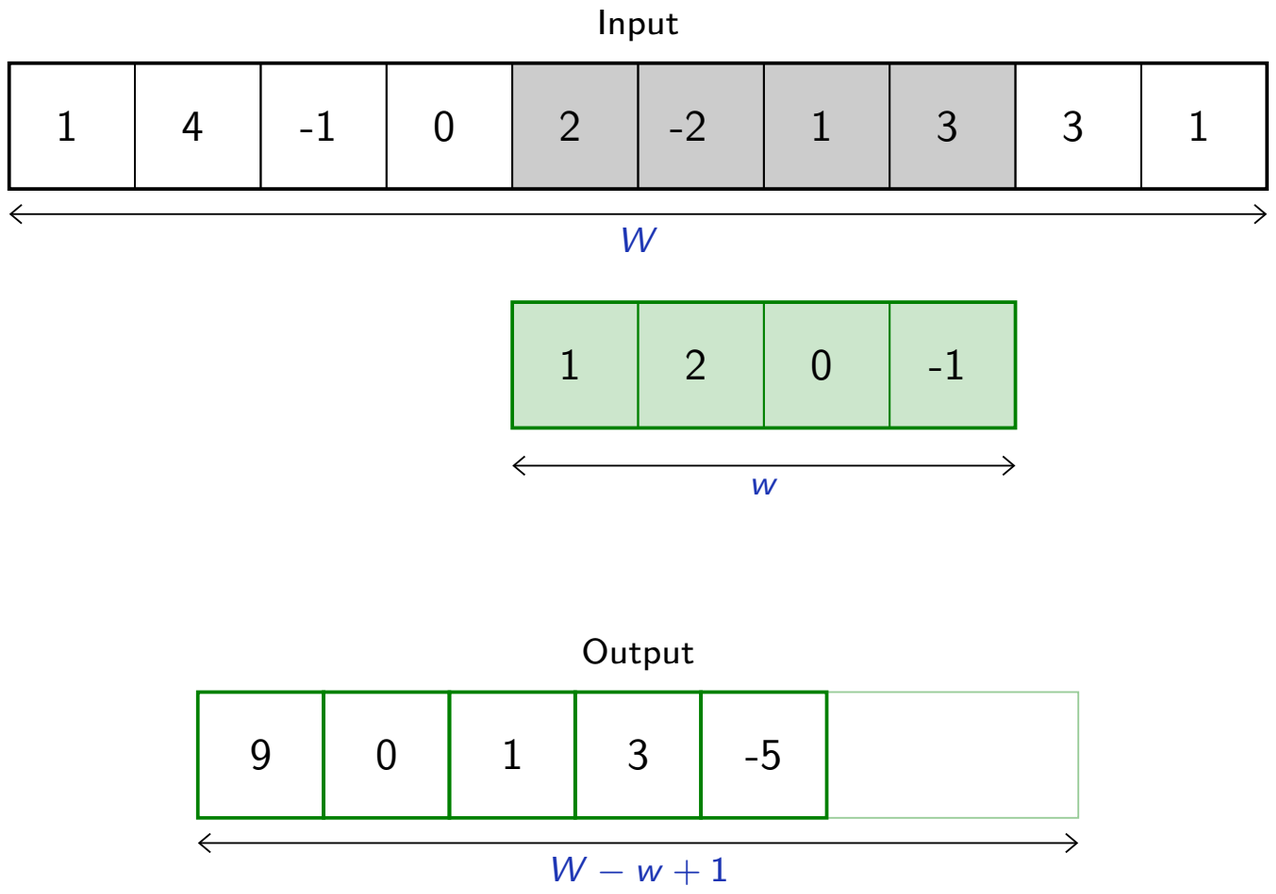
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 1D



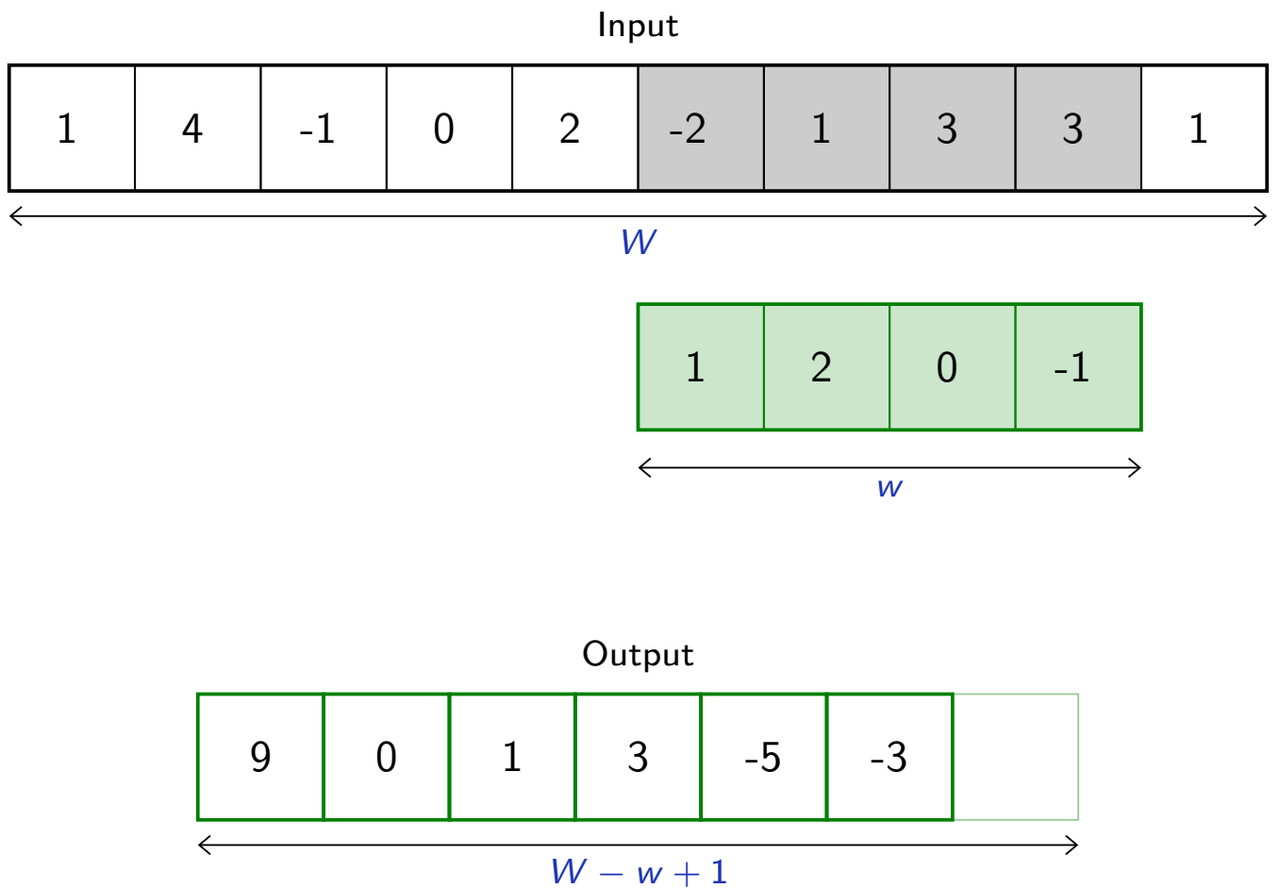
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 1D



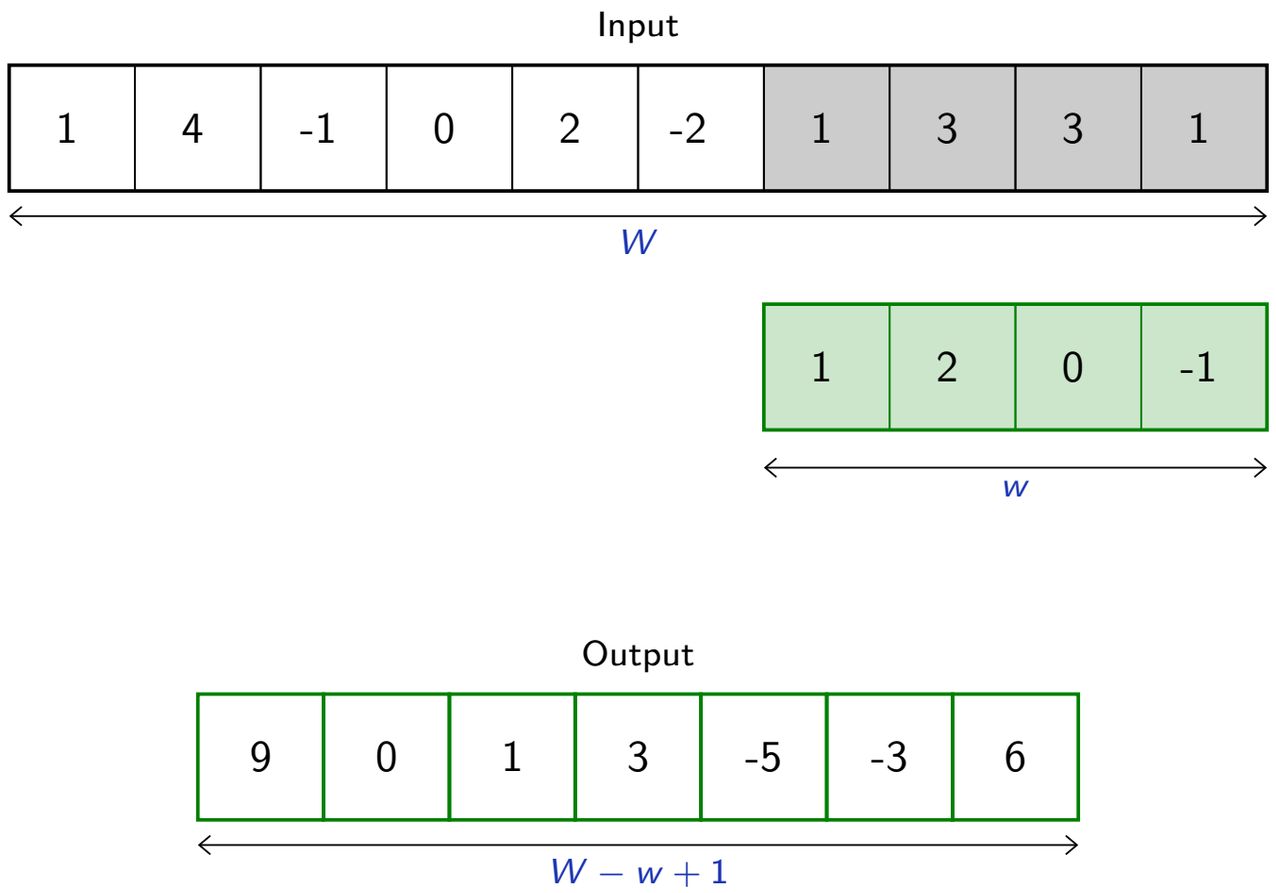
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 1D



François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

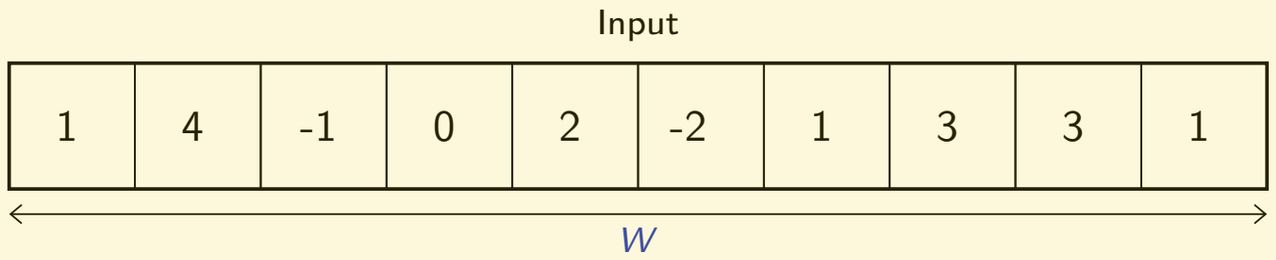
Convolution 1D



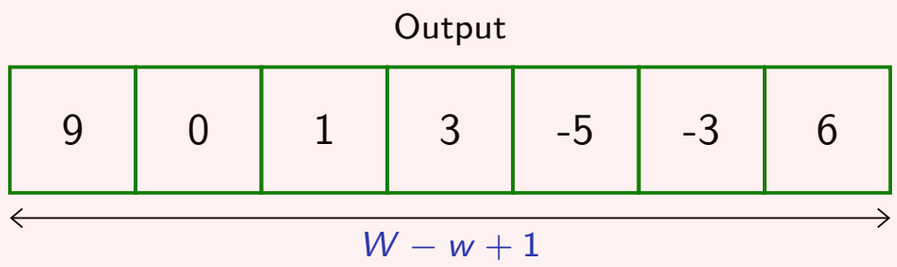
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 1D

$$x = (x_1, \ldots, x_W)$$



$$u = (u_1, \ldots, u_w)$$



$$(x \circledast u)_i = \sum_{j=1}^{w} x_{i-1+j}\, u_j$$

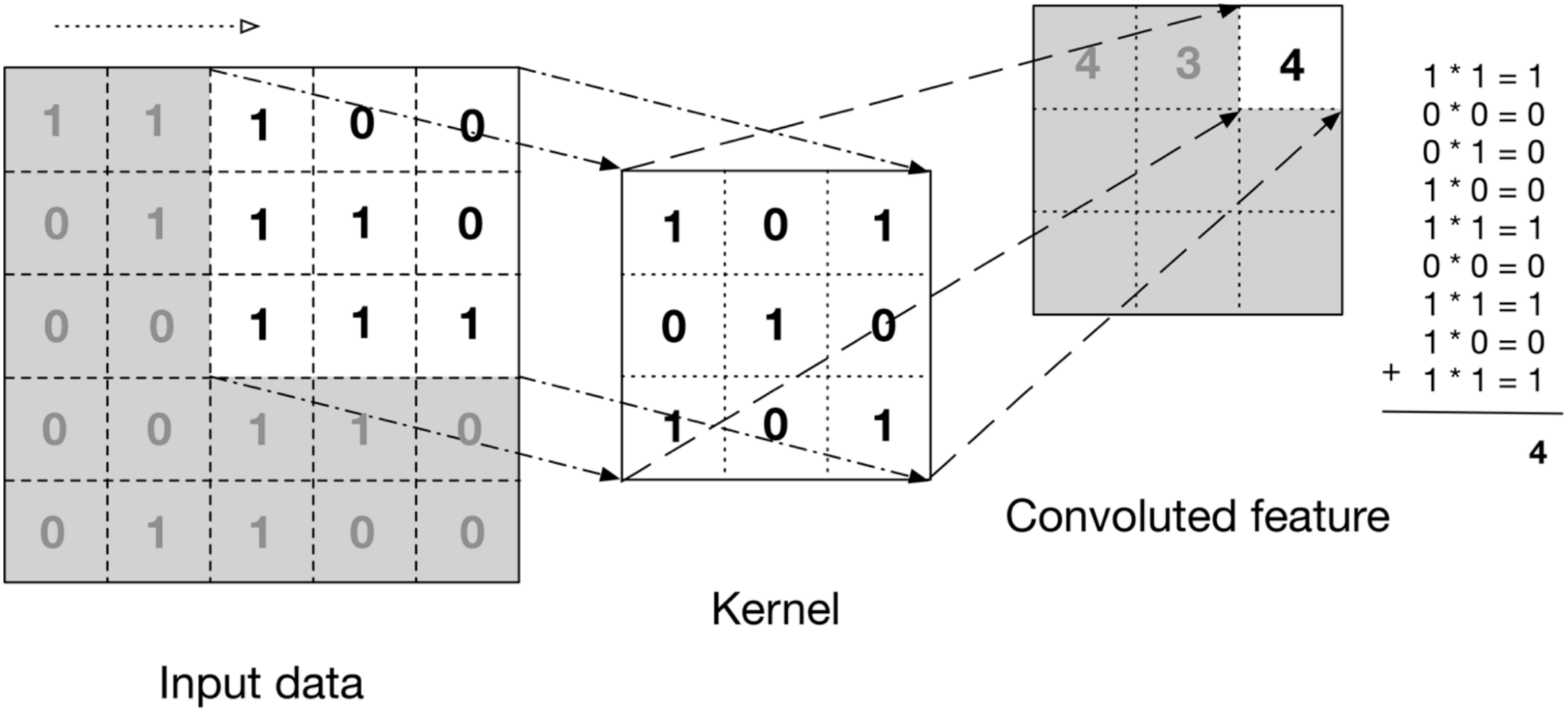
$$= (x_i, \ldots, x_{i+w-1}) \cdot u$$

My Data is 2D

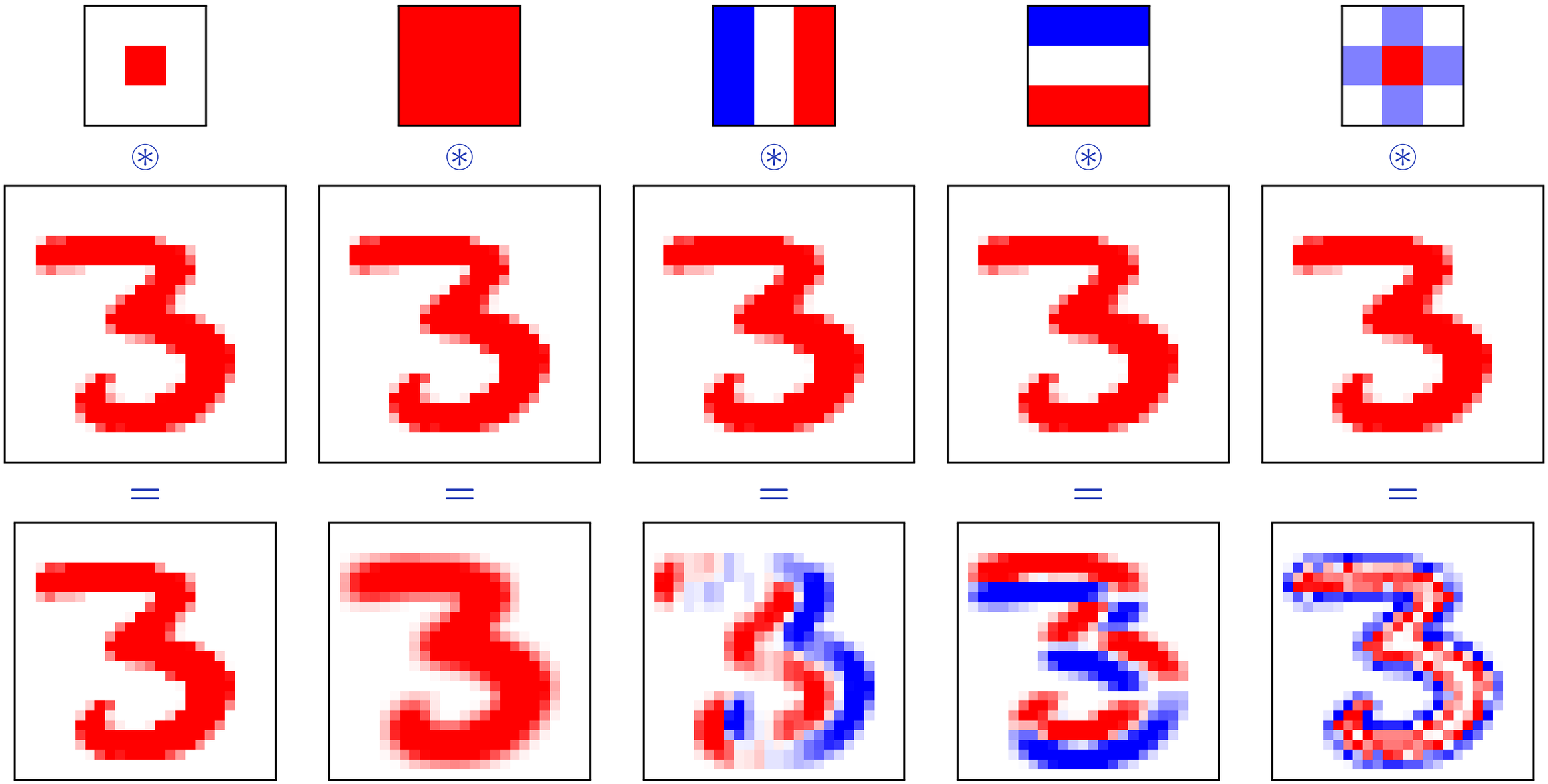
88	126	145	85	123	142	85	123	142	86	124
86	125	142	84	123	140	83	122	139	85	124
85	124	141	82	121	138	82	121	138	84	123
82	119	135	80	117	133	80	117	133	85	122
78	114	128	77	113	127	79	115	129	84	120
79	115	129	78	114	128	80	116	130	83	119
82	118	130	81	117	129	81	117	129	82	118
83	117	129	82	116	128	82	116	128	82	116
79	113	123	79	113	123	80	114	124	81	115
76	108	119	76	108	119	77	109	120	80	112
76	109	118	76	109	118	77	110	119	79	112

https://medium.com/deep-math-machine-learning-ai/chapter-8-0-convolutional-neural-networks-for-deep-learning-364971e34ab2

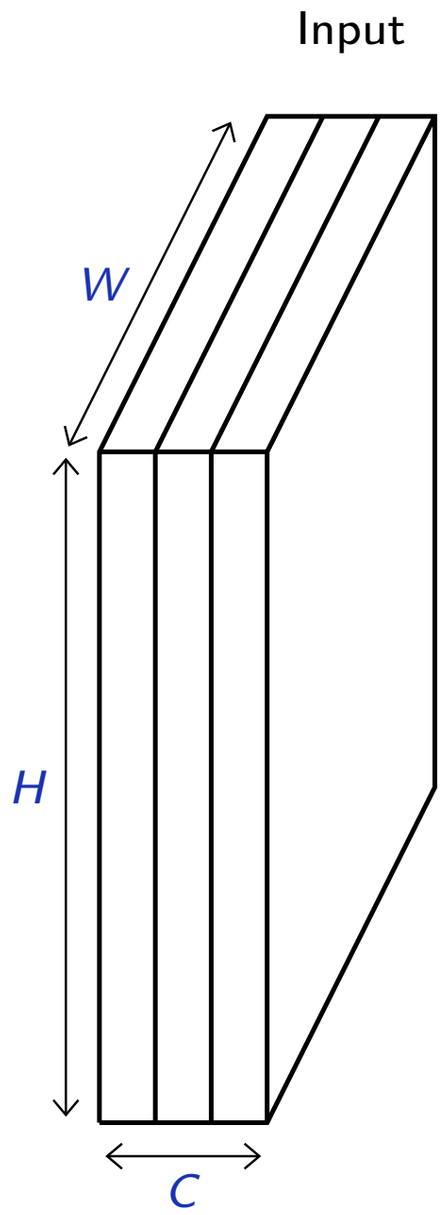
Convolutions 2D



Convolutions 2D

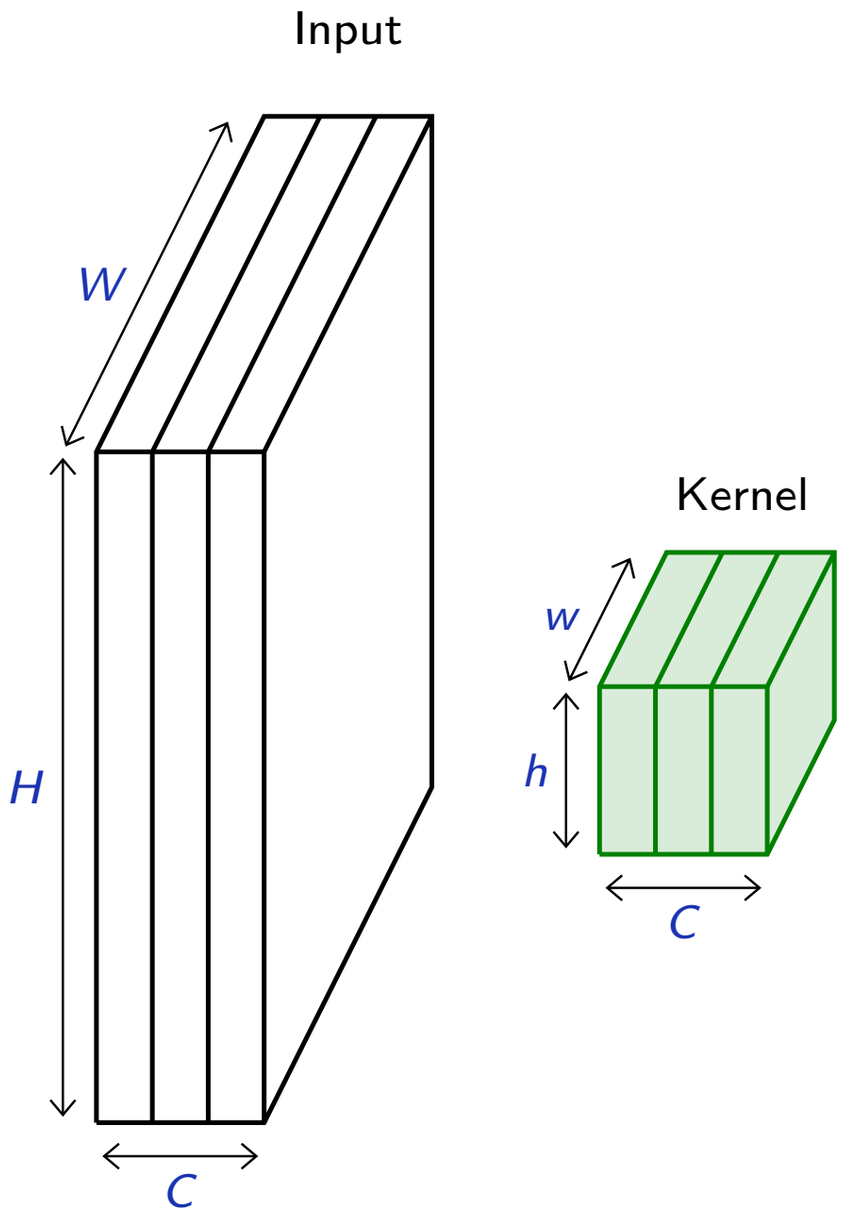


Convolution 2D + RGB Channels



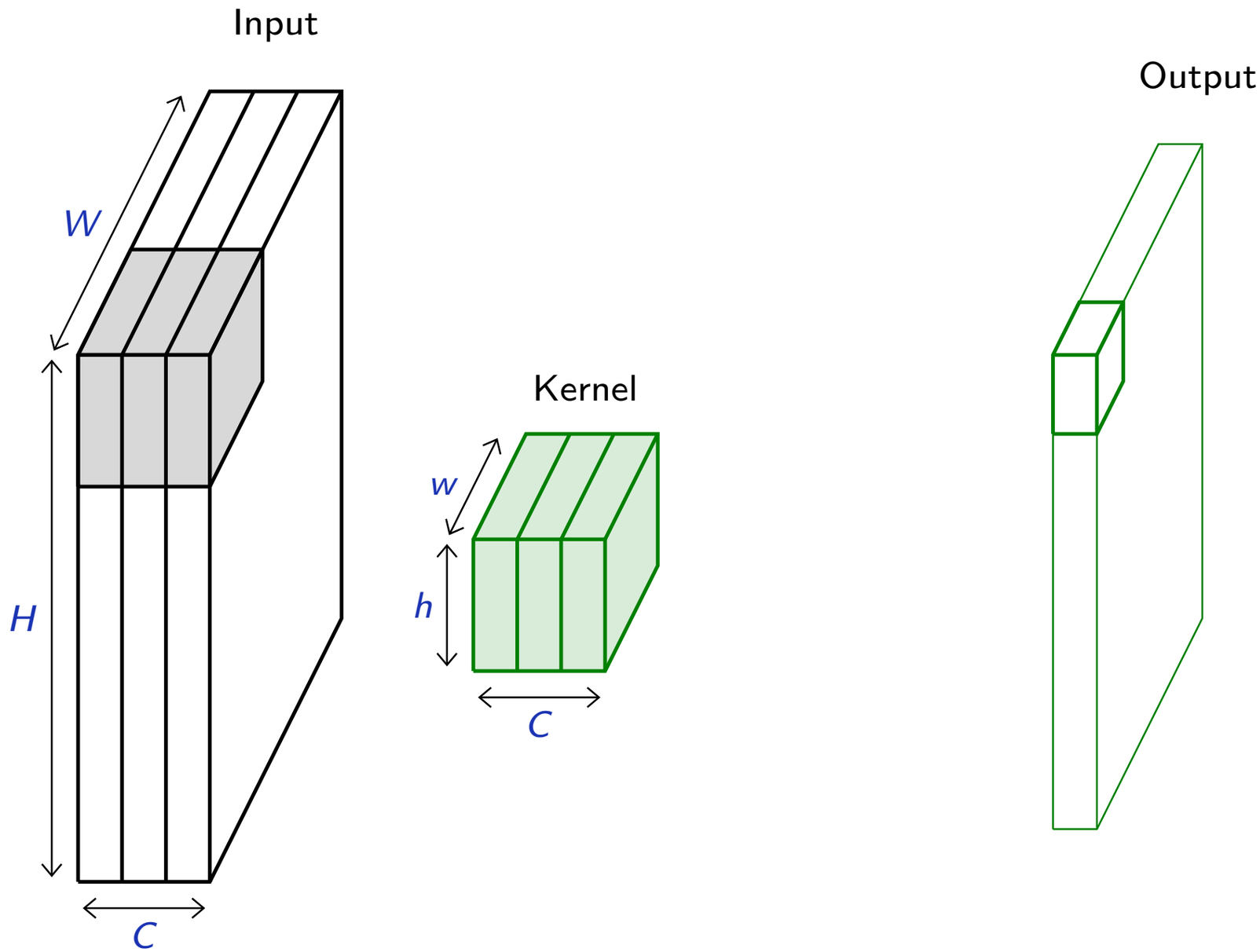
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



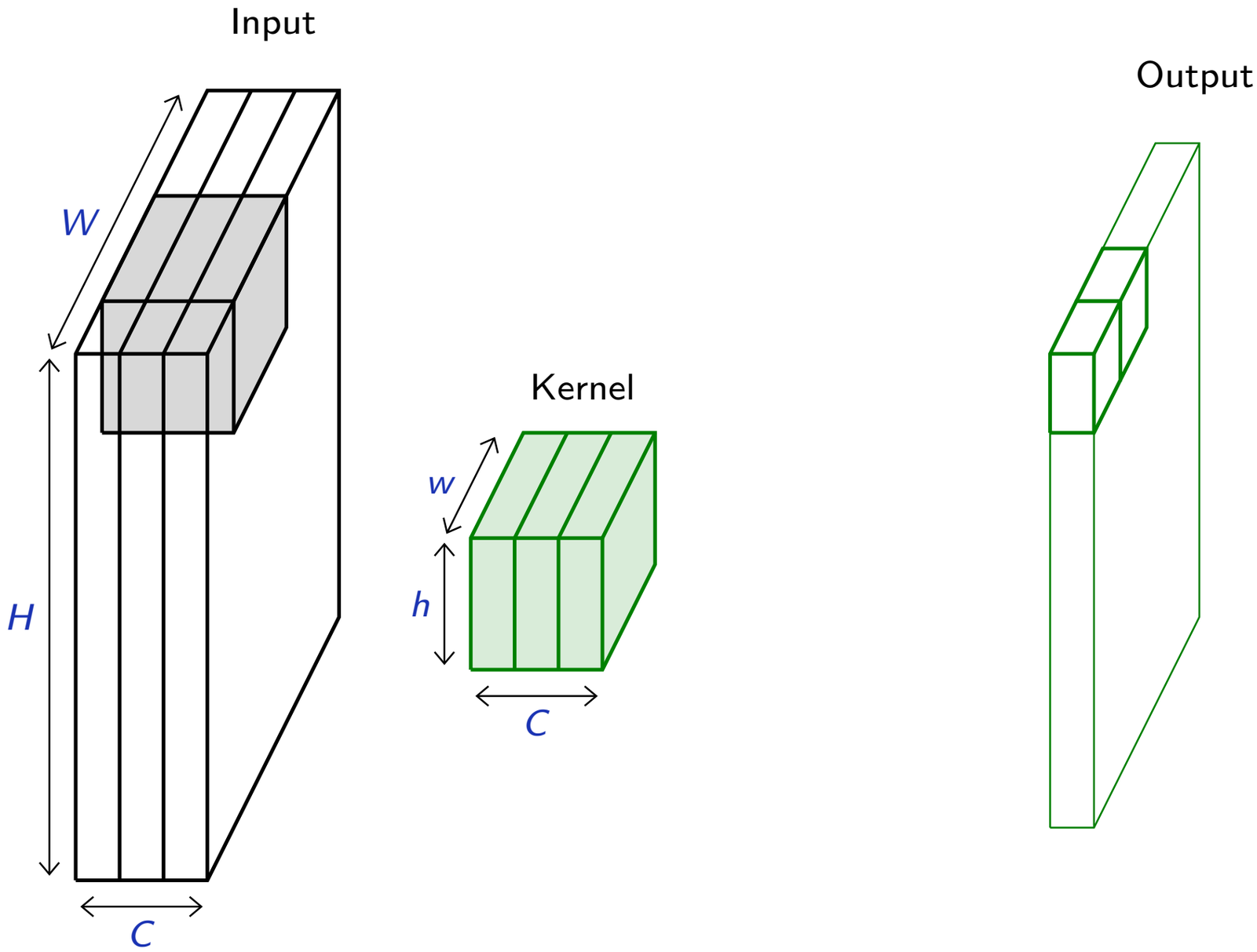
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



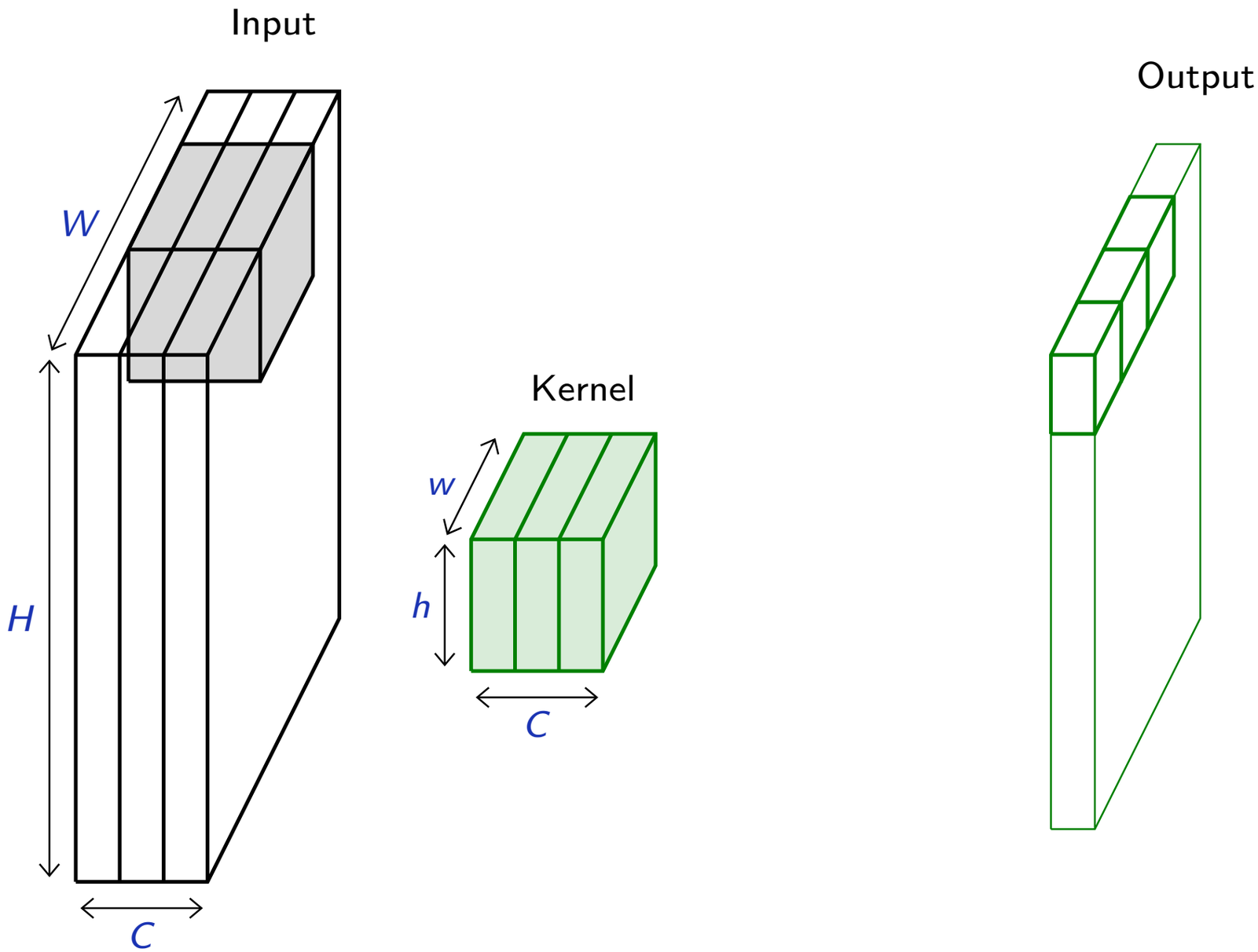
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



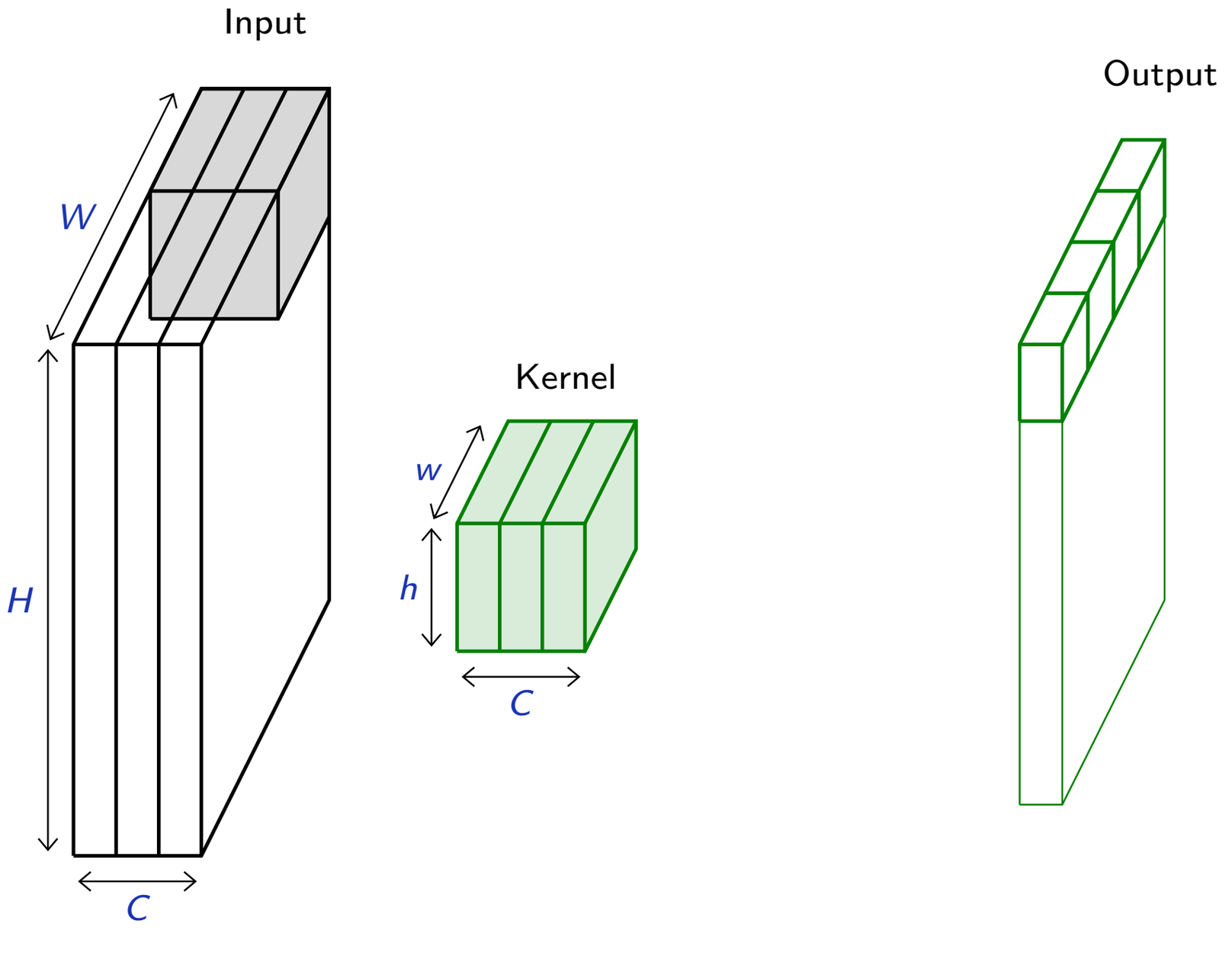
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



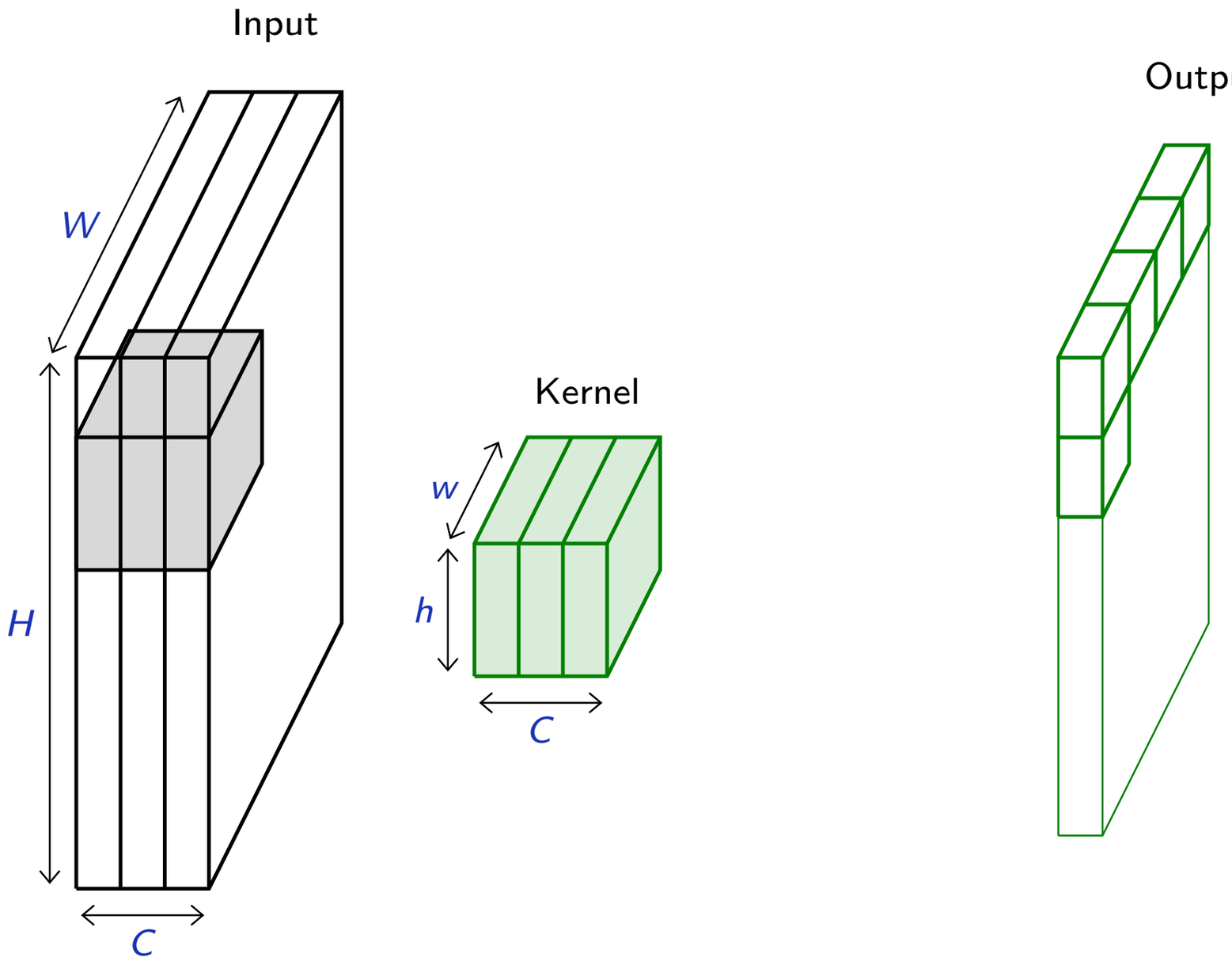
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



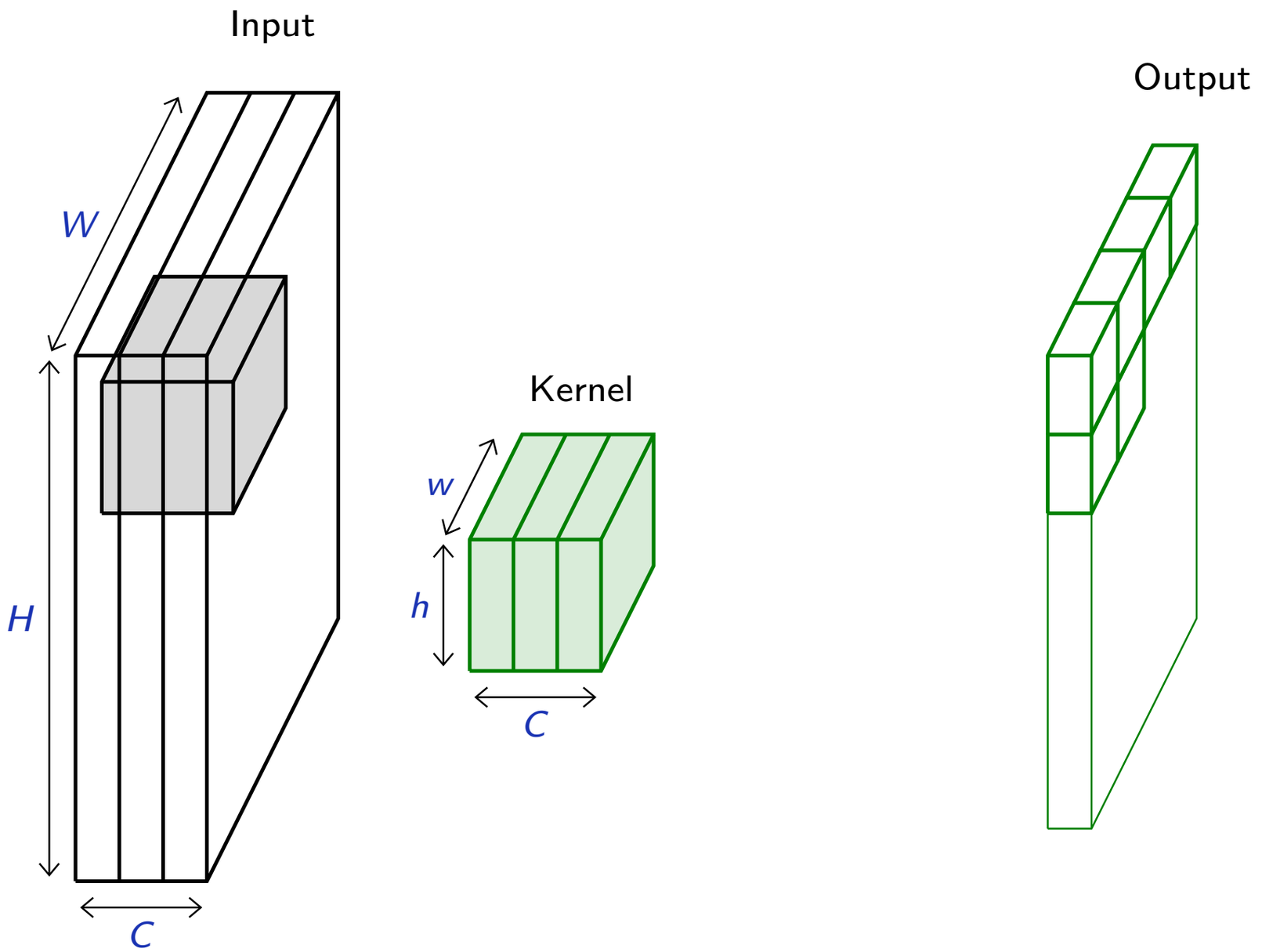
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



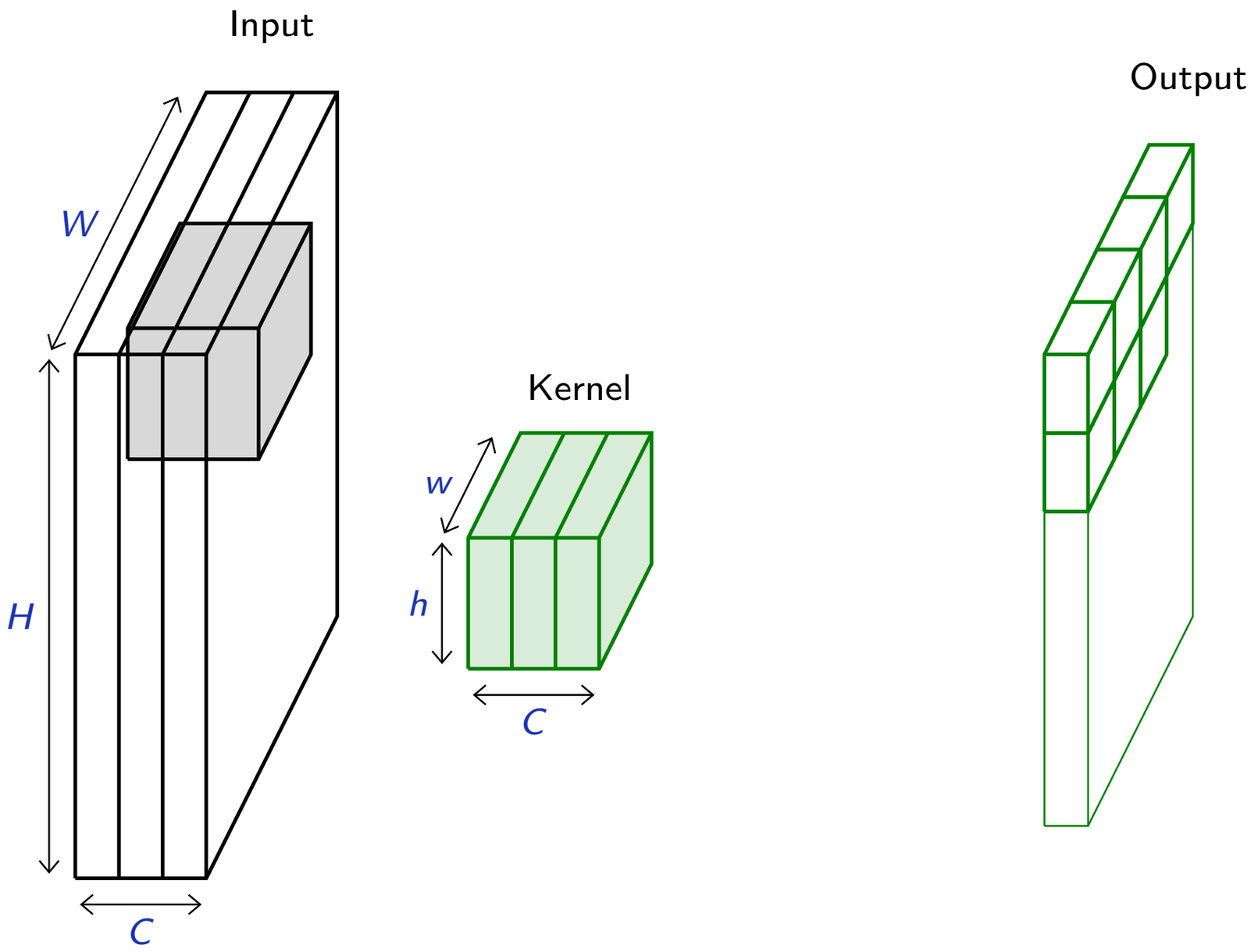
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



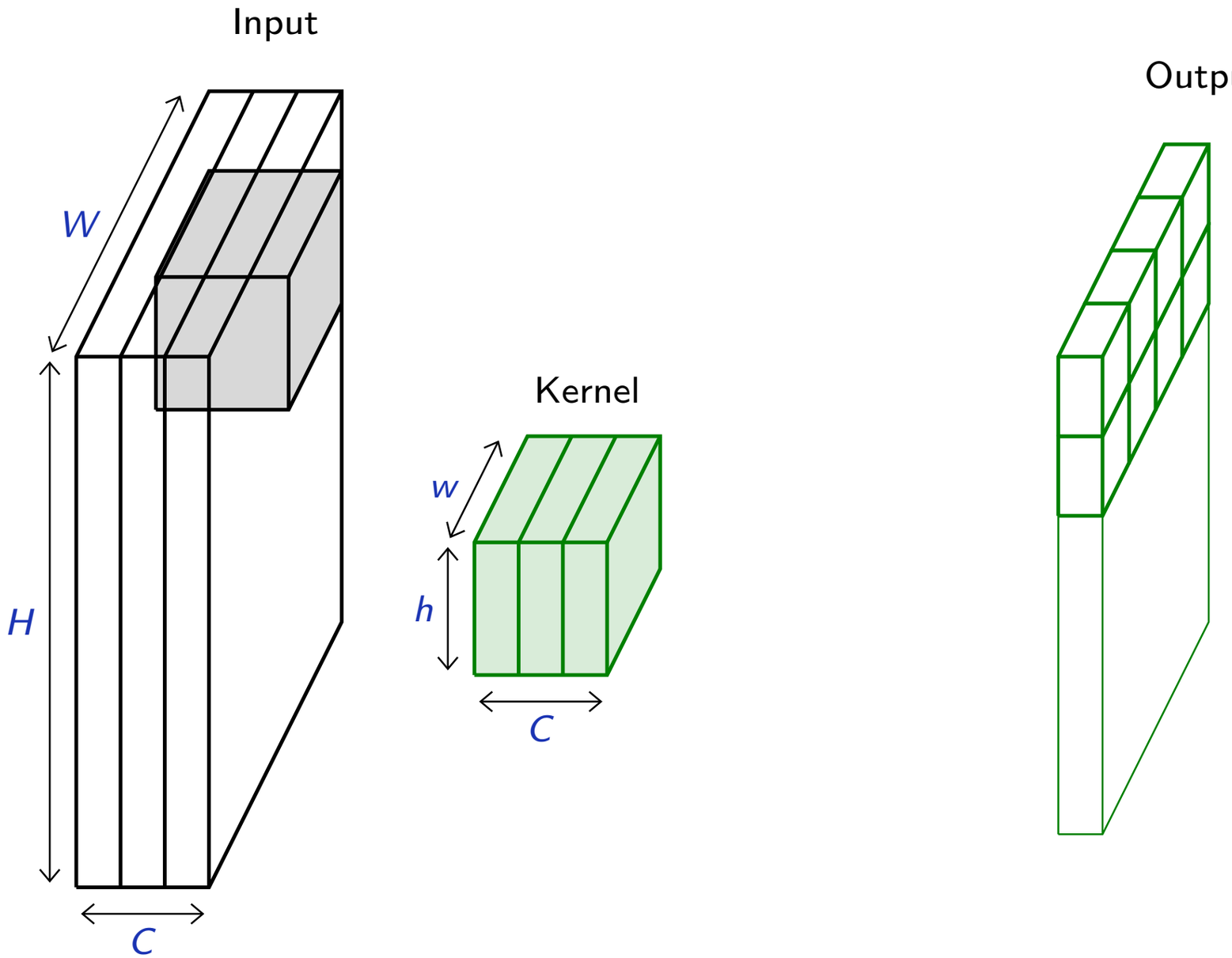
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



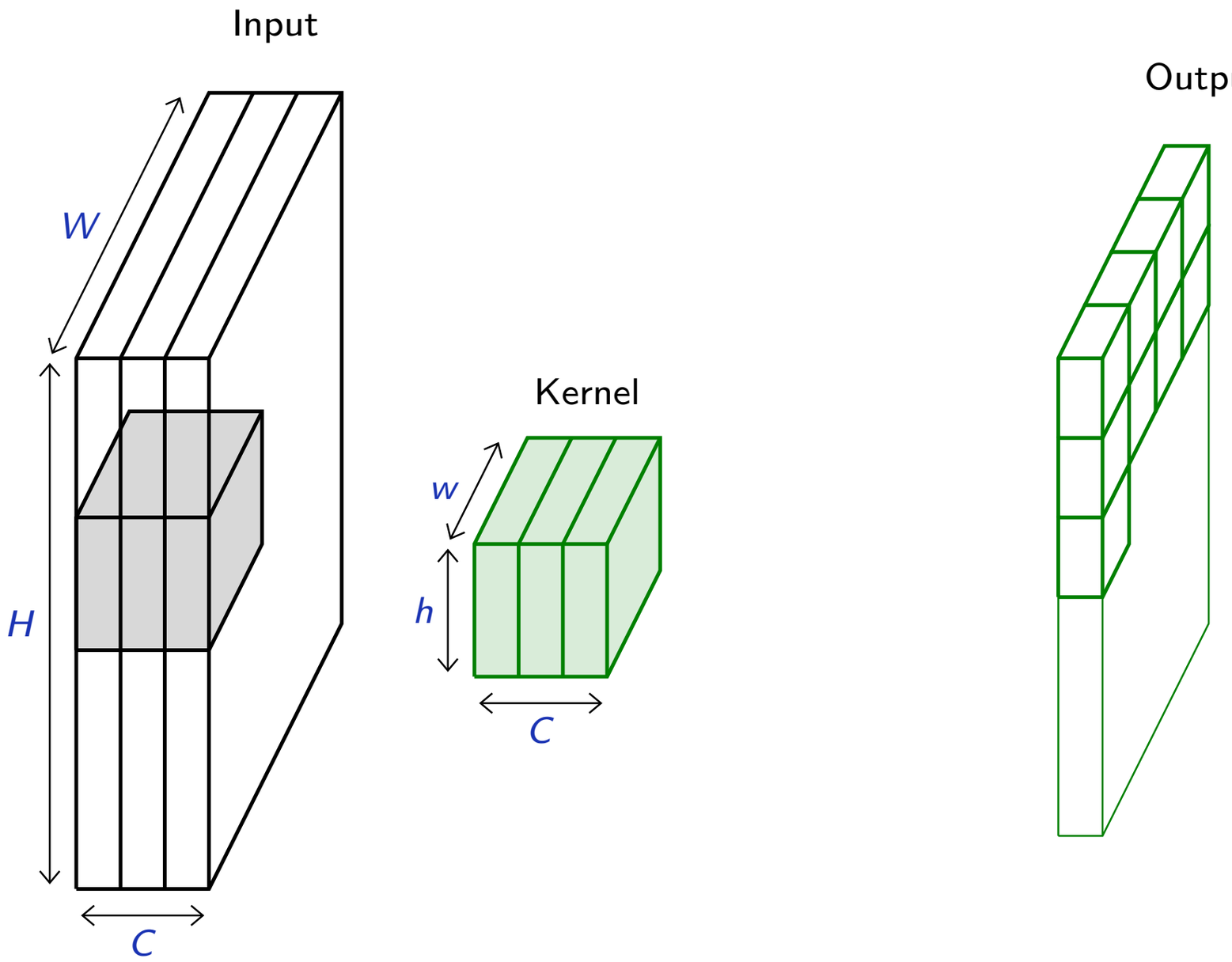
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



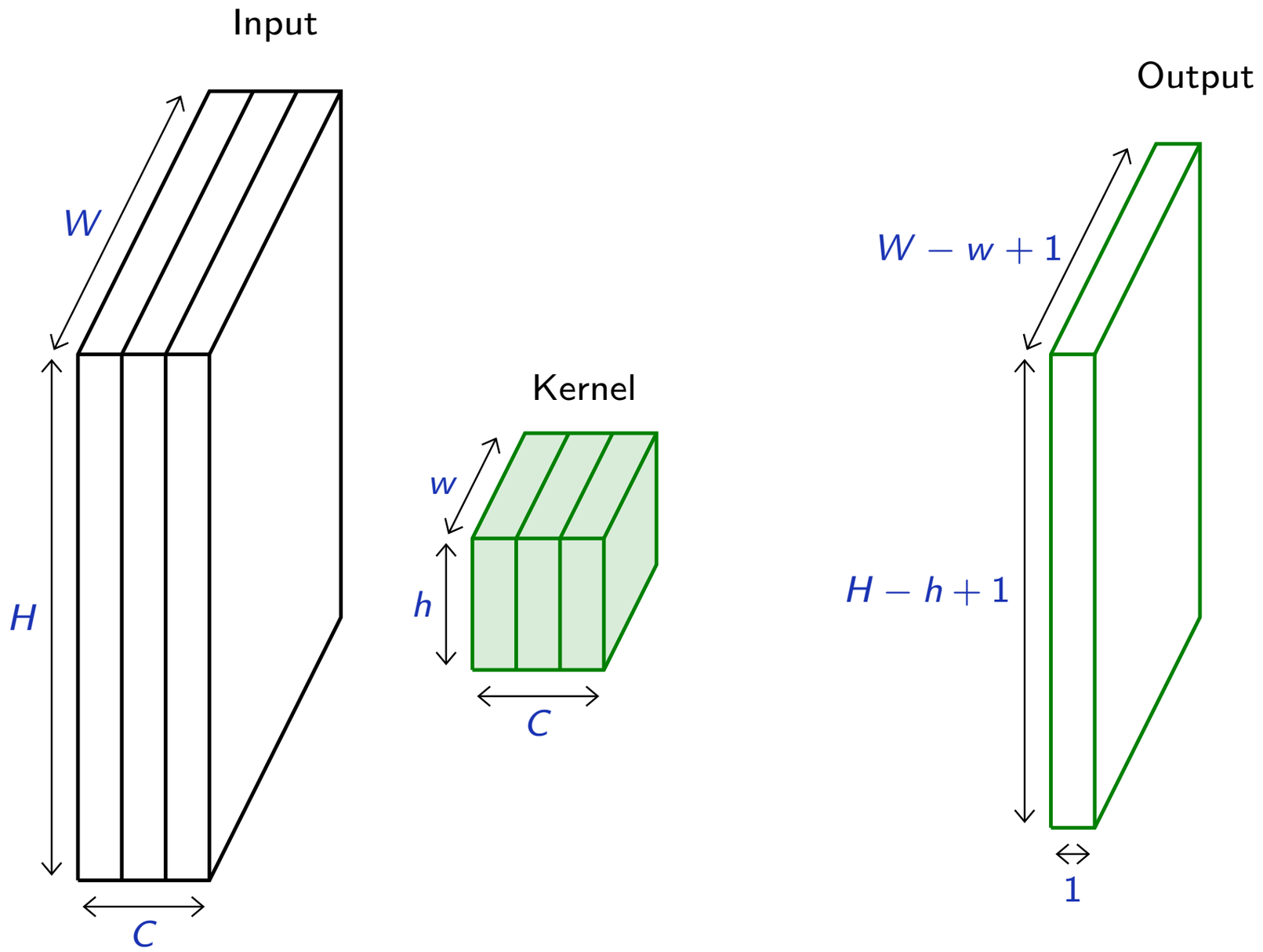
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



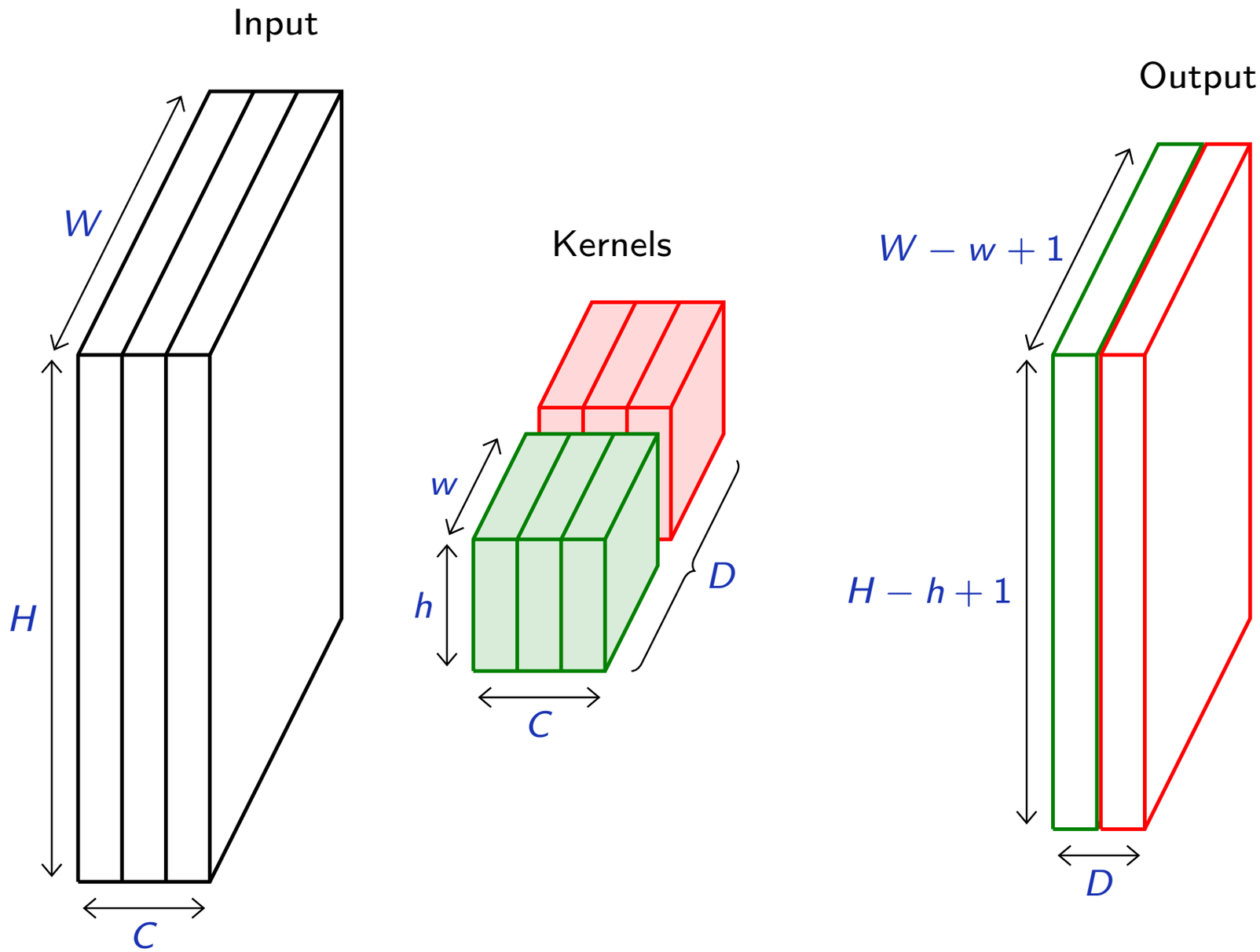
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



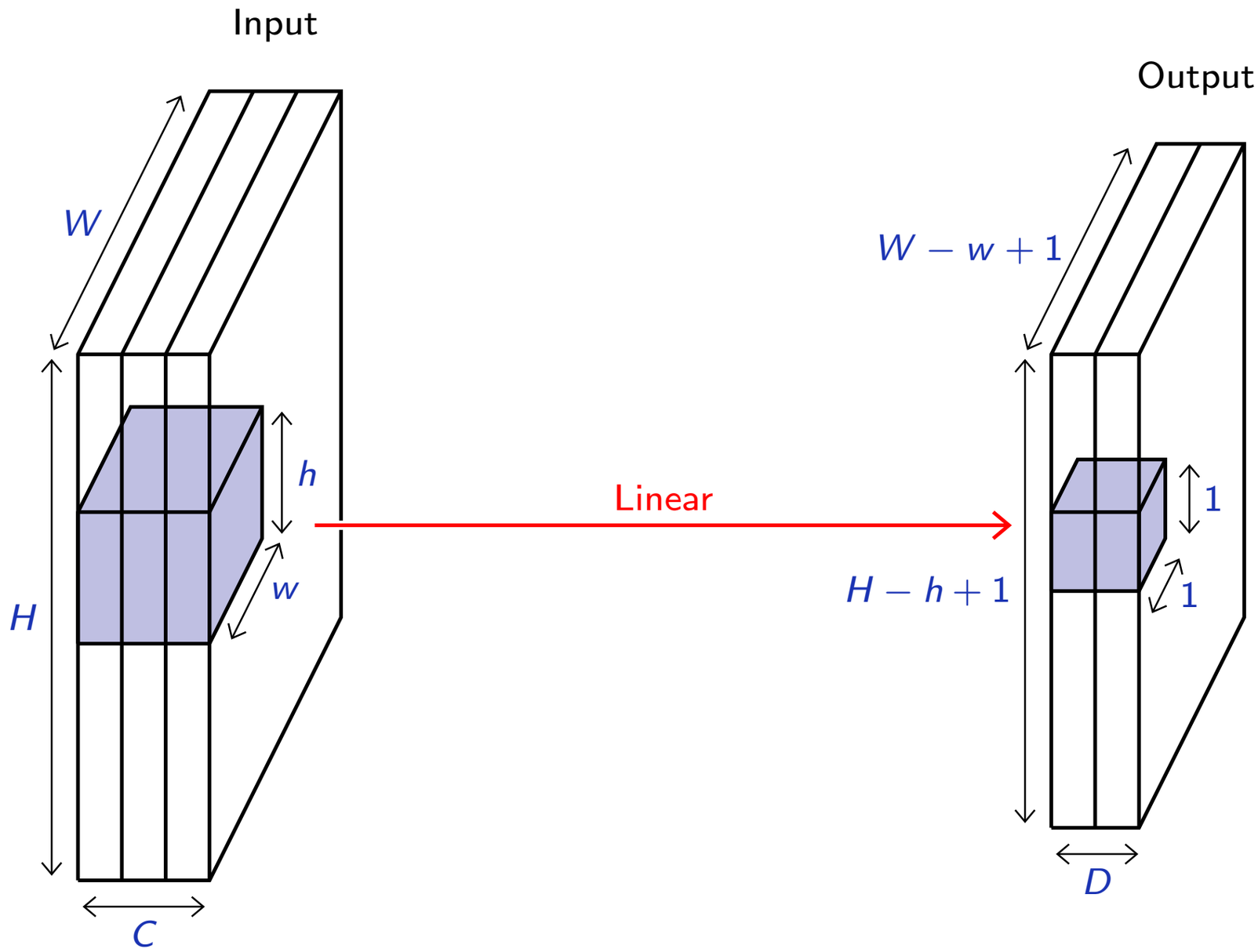
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



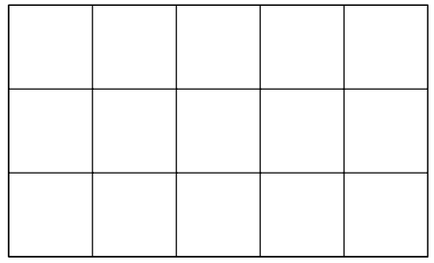
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

Convolution 2D + RGB Channels



François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

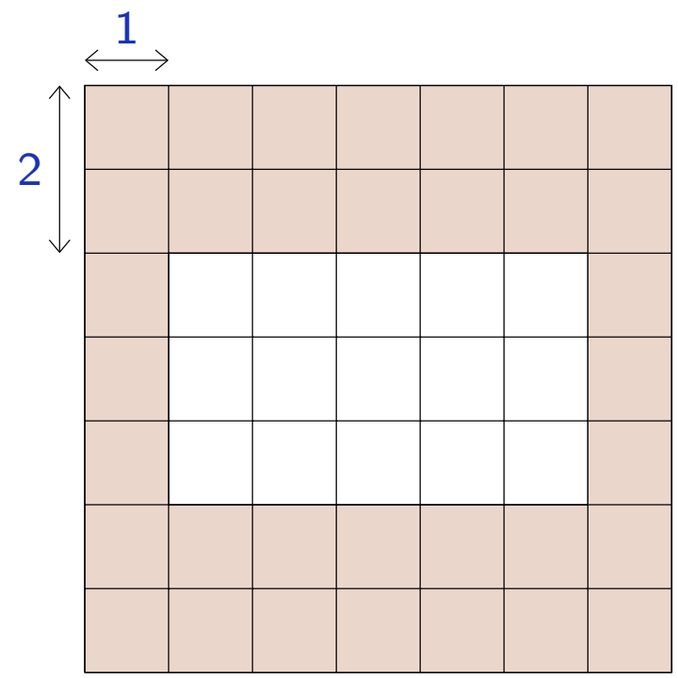
2D Convolutions / Local issues



Input

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

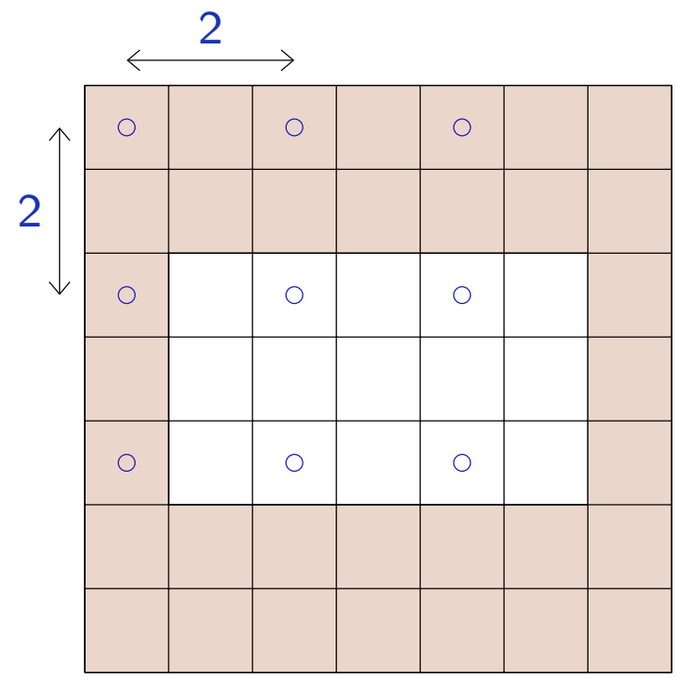
2D Convolutions / Local issues



Input

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

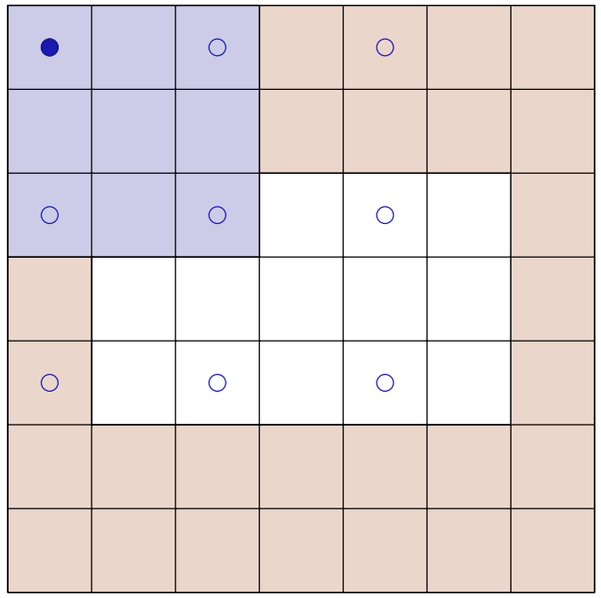
2D Convolutions / Local issues



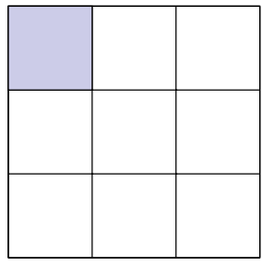
Input

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues



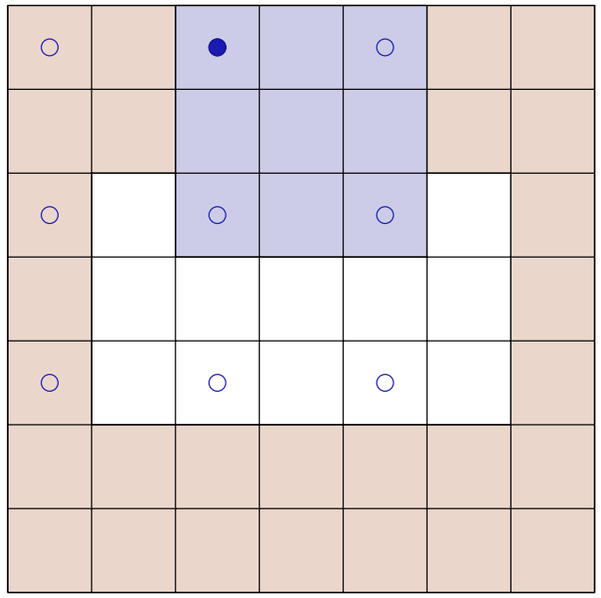
Input



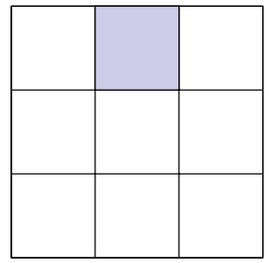
Output

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues



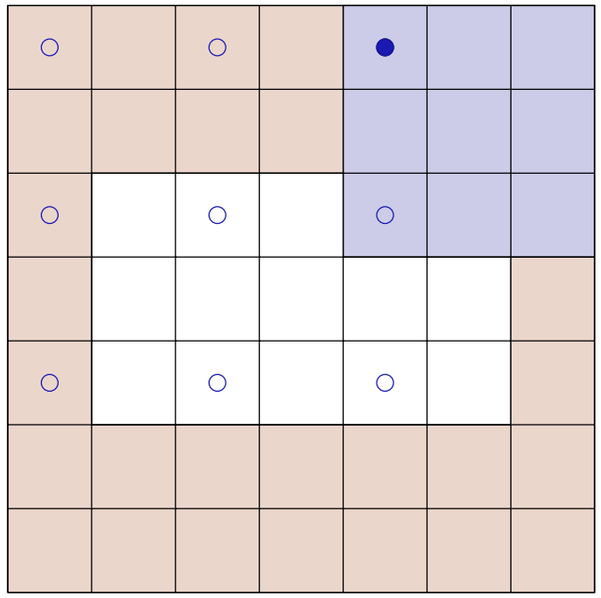
Input



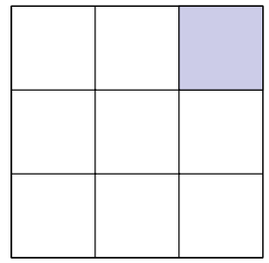
Output

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues



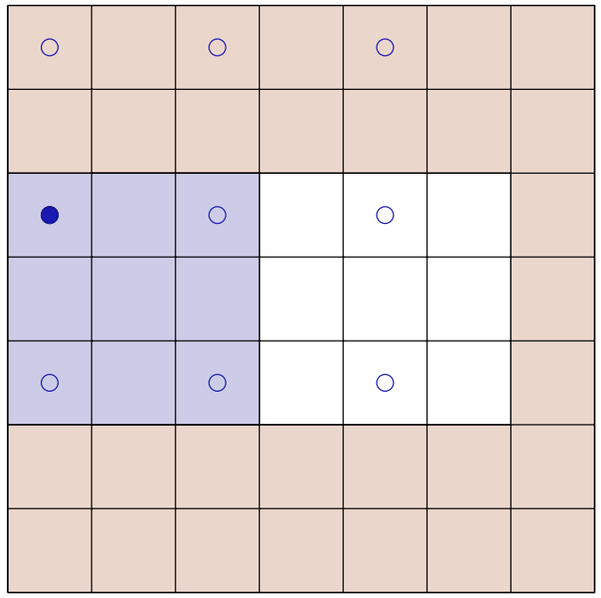
Input



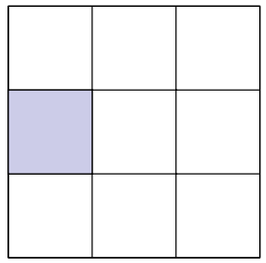
Output

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues



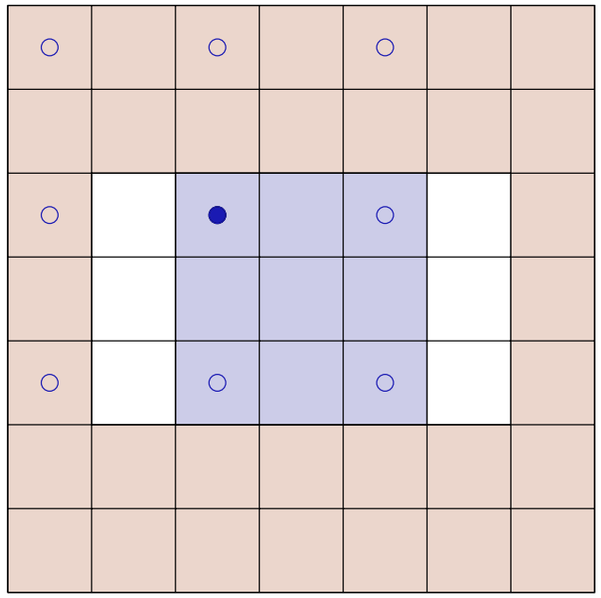
Input



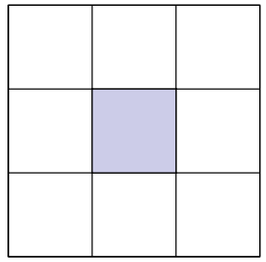
Output

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues



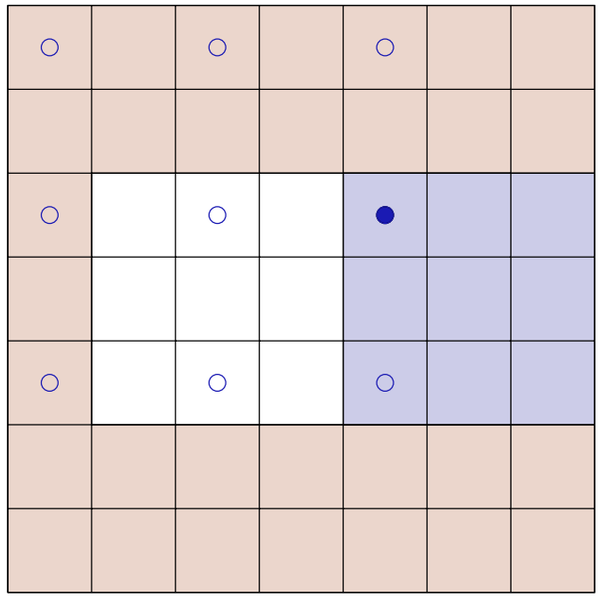
Input



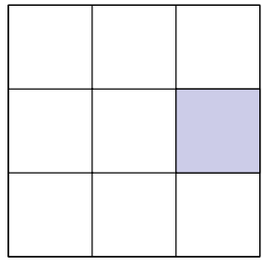
Output

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues



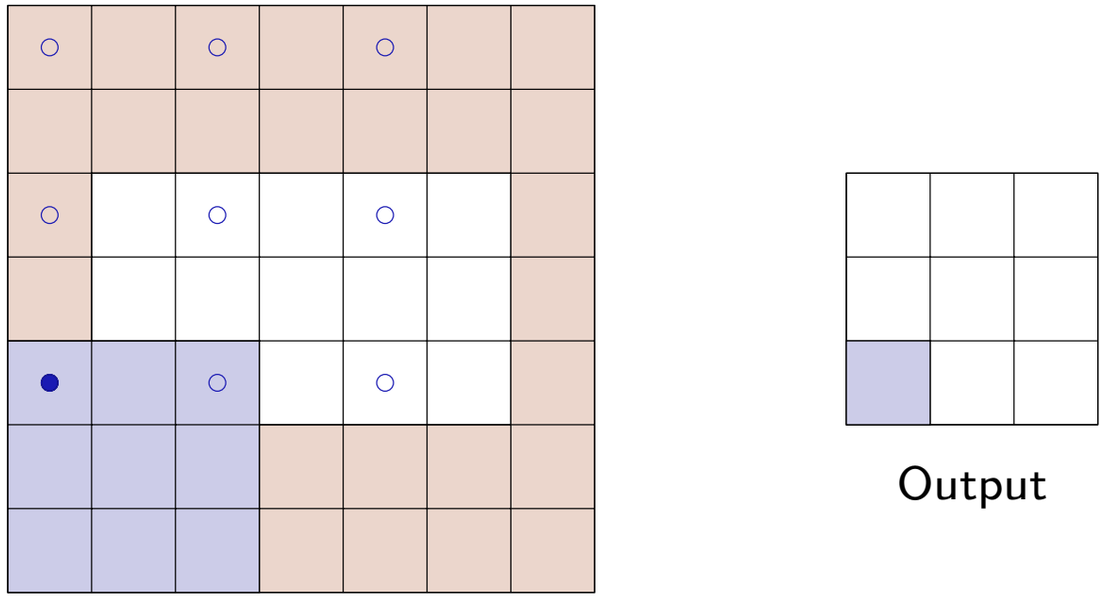
Input



Output

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues

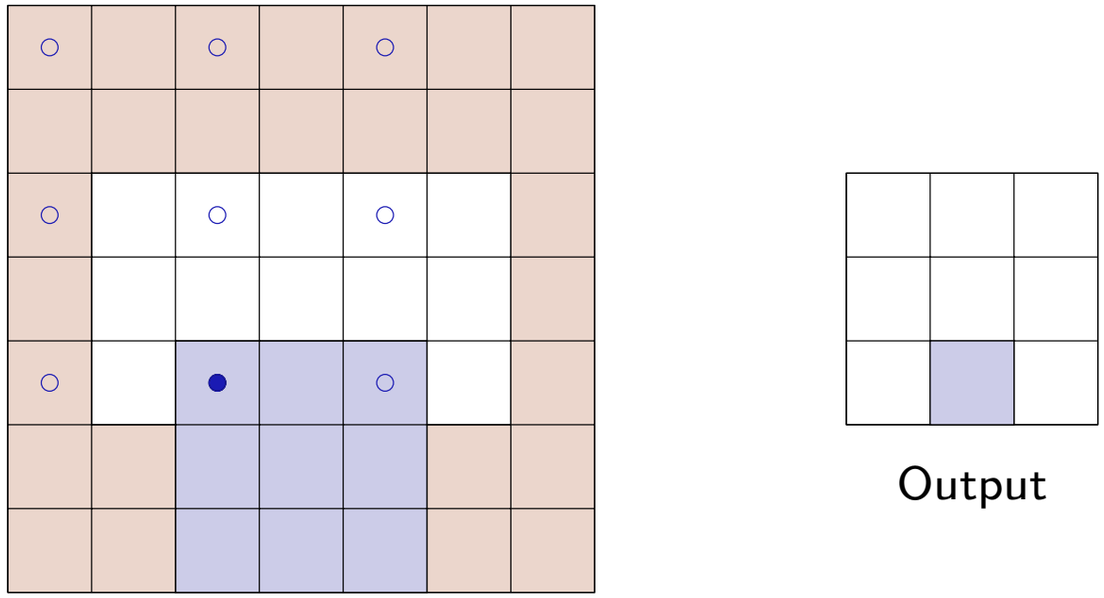


Input

Output

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues

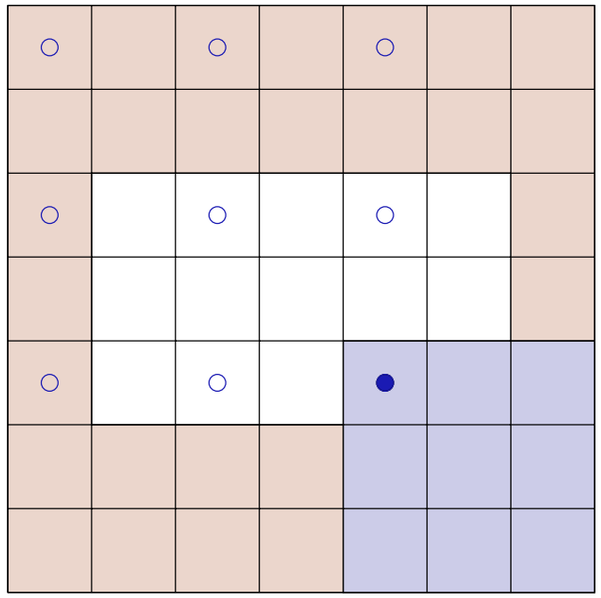


Input

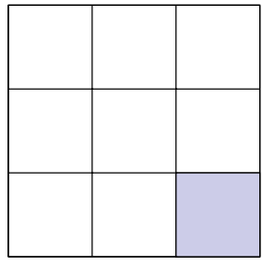
Output

François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues



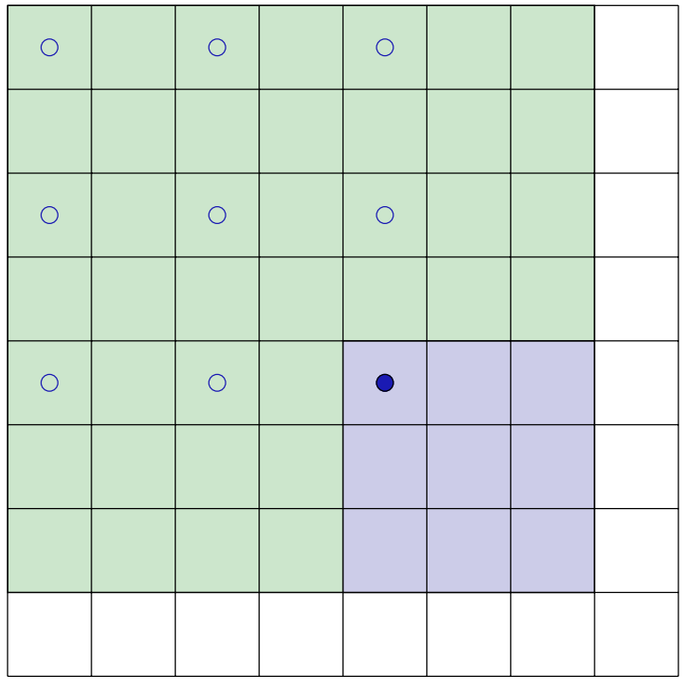
Input



Output

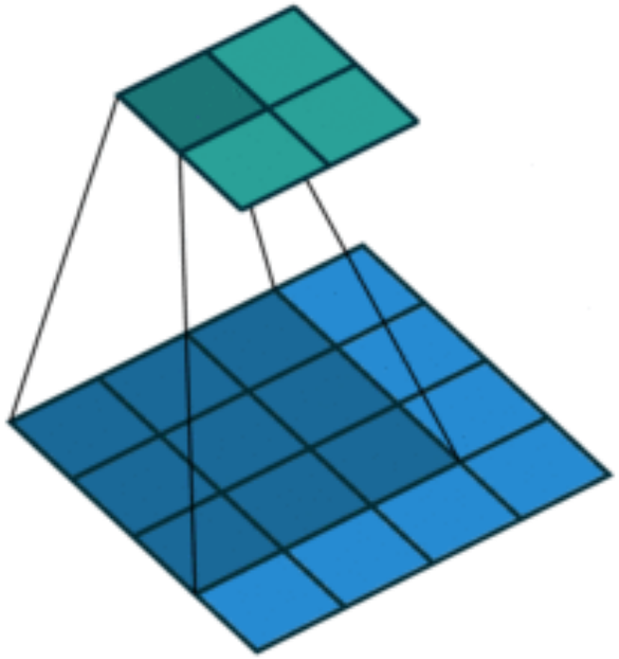
François Fleuret: https://fleuret.org/ee559/materials/ee559-slides-4-4-convolutions.pdf

2D Convolutions / Local issues

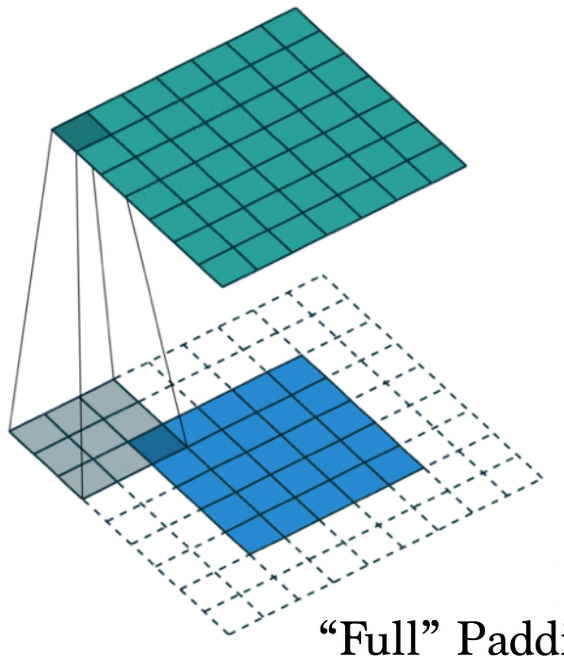


⚠ A convolution with a stride greater than 1 may not cover the input map completely, hence may ignore some of the input values.

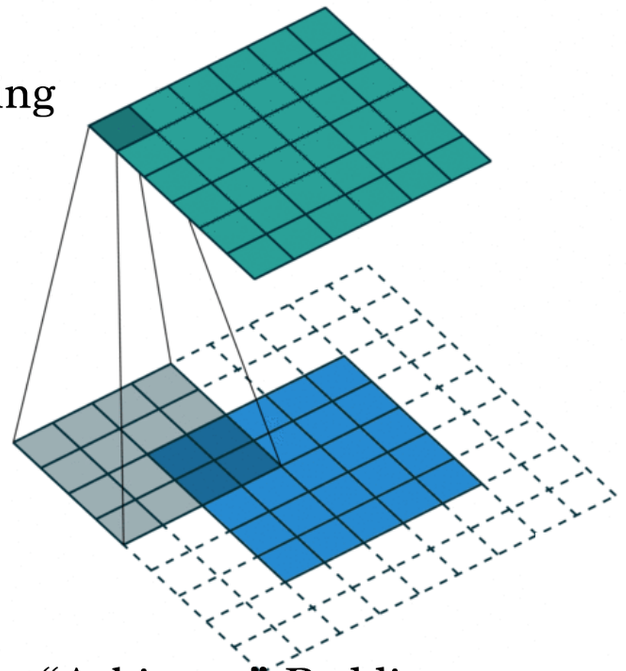
Convolutions 2D: Local Issues



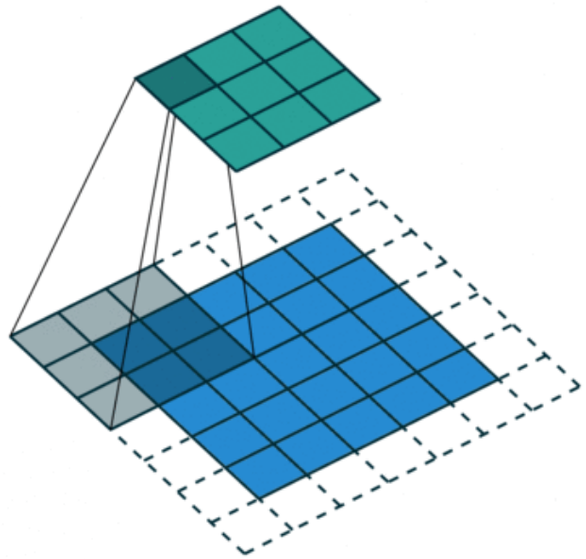
Strides



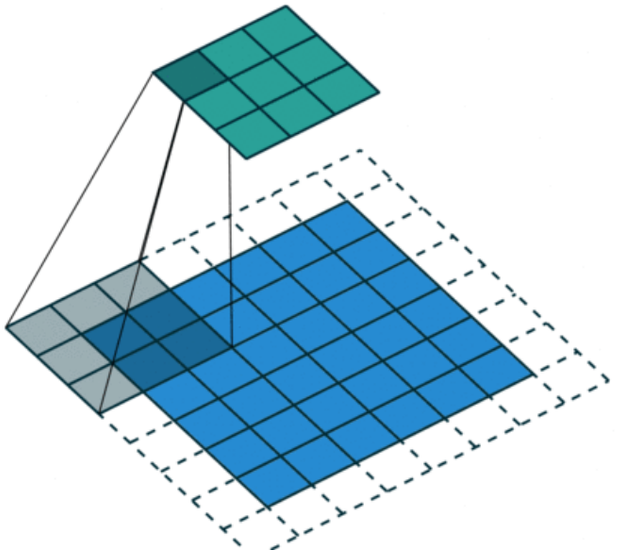
"Same" Padding



"Arbitrary" Padding

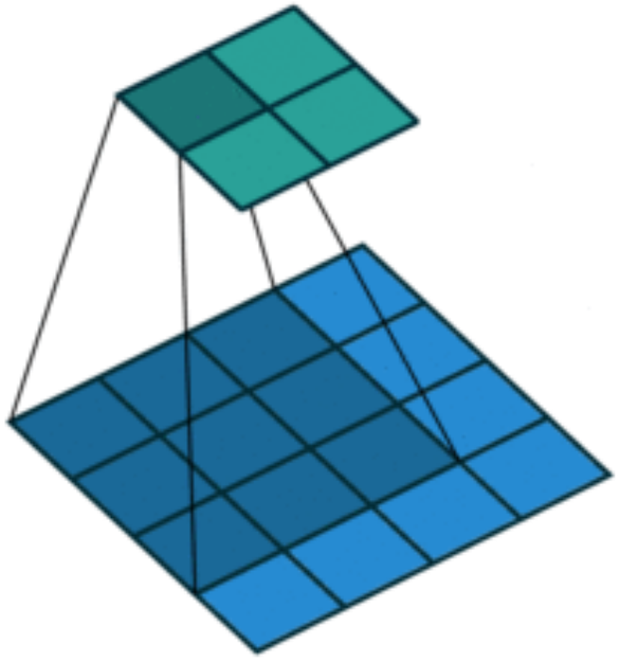


Both

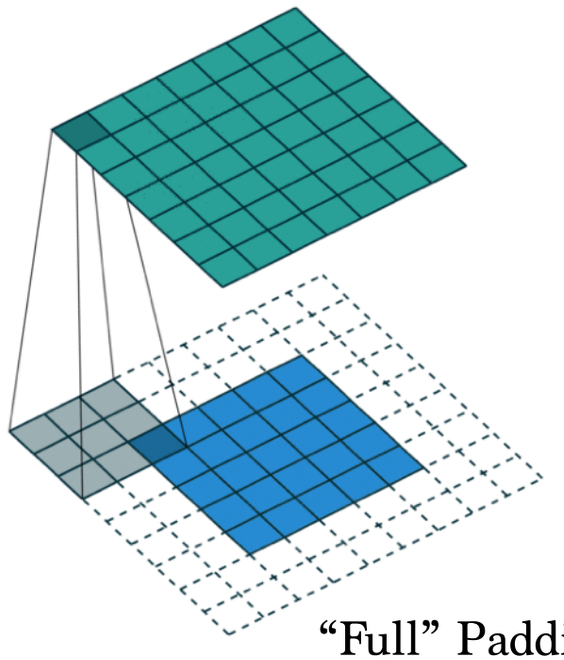


https://github.com/vdumoulin/conv_arithmetic

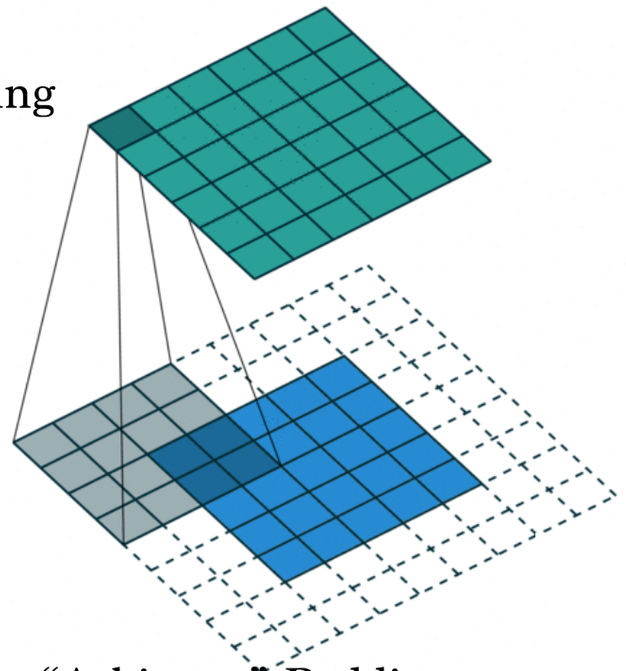
Convolutions 2D: Local Issues



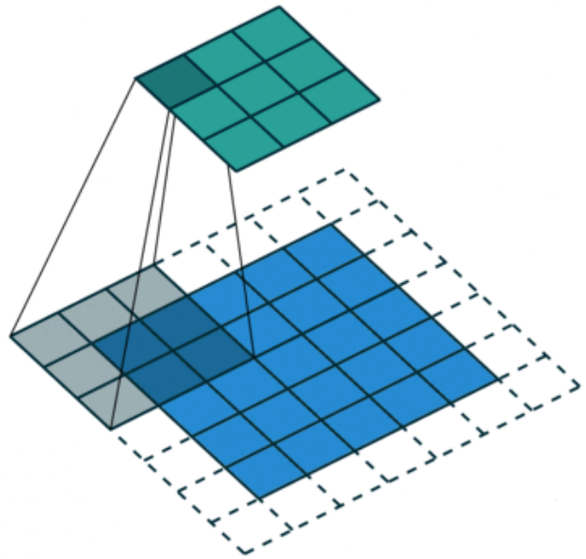
Strides



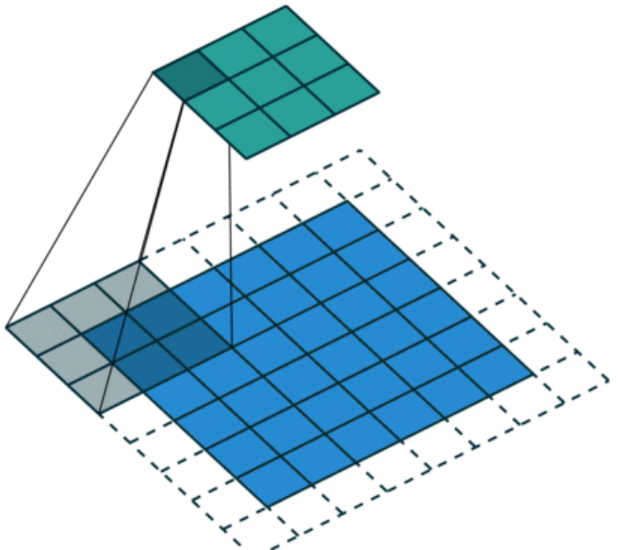
"Same" Padding



"Arbitrary" Padding

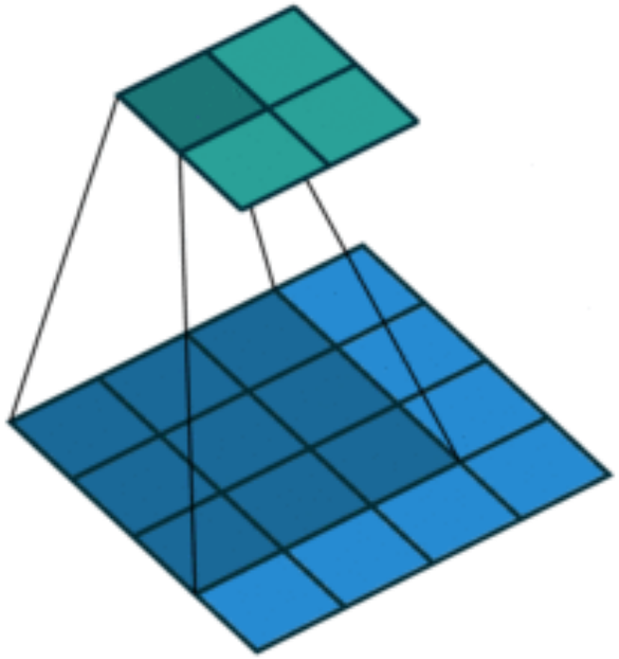


Both

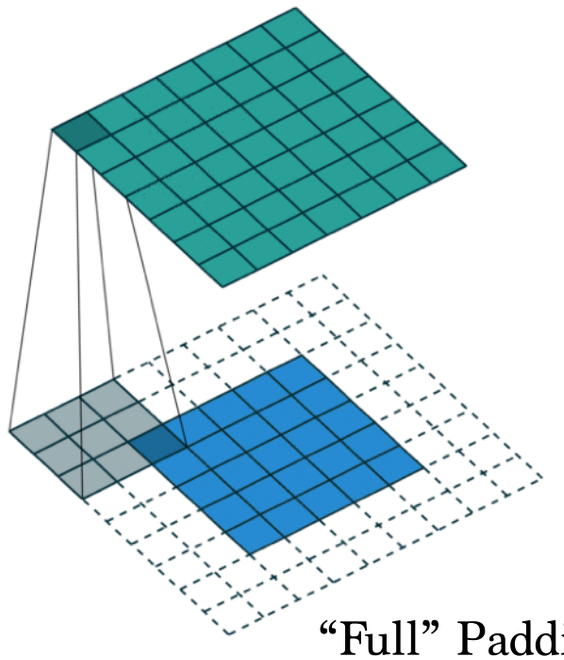


https://github.com/vdumoulin/conv_arithmetic

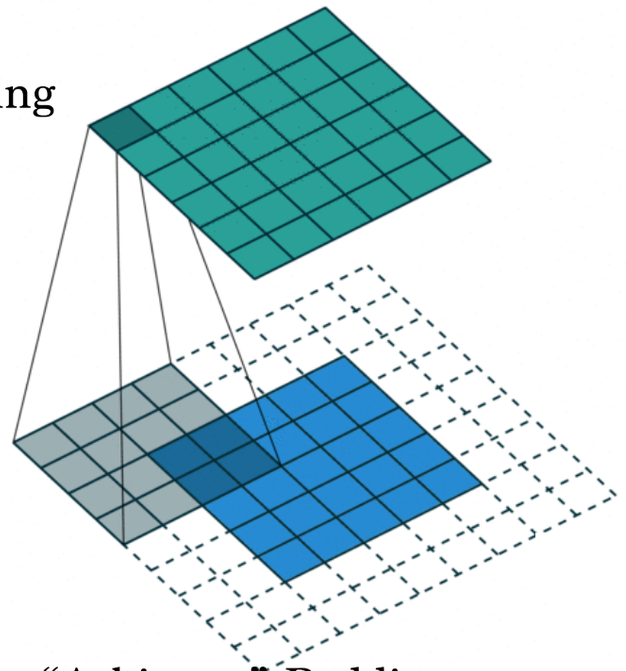
Convolutions 2D: Local Issues



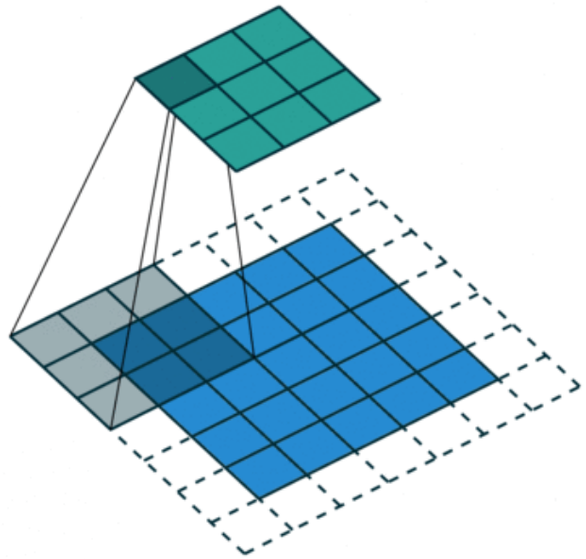
Strides



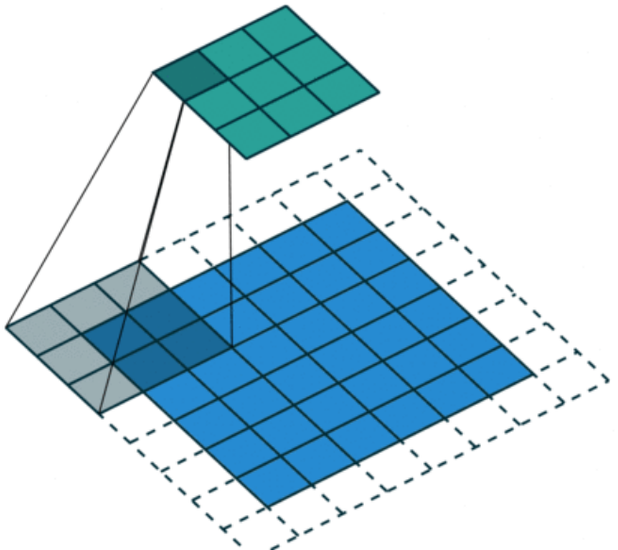
"Same" Padding



"Arbitrary" Padding

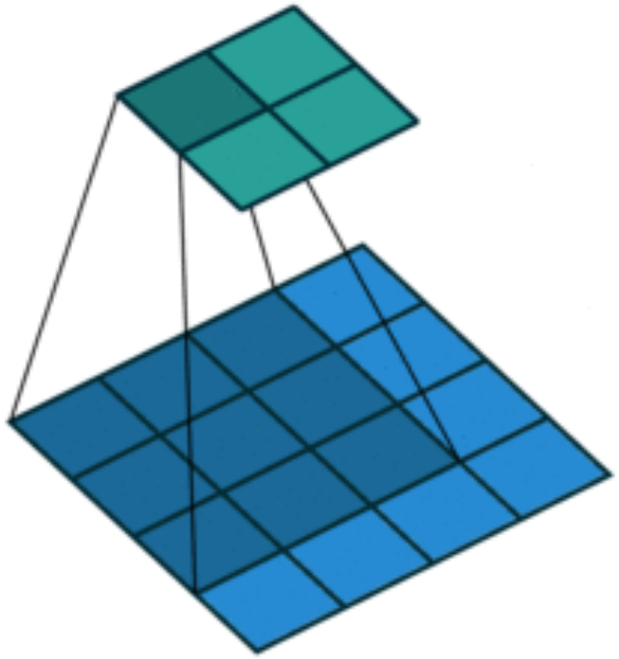


Both

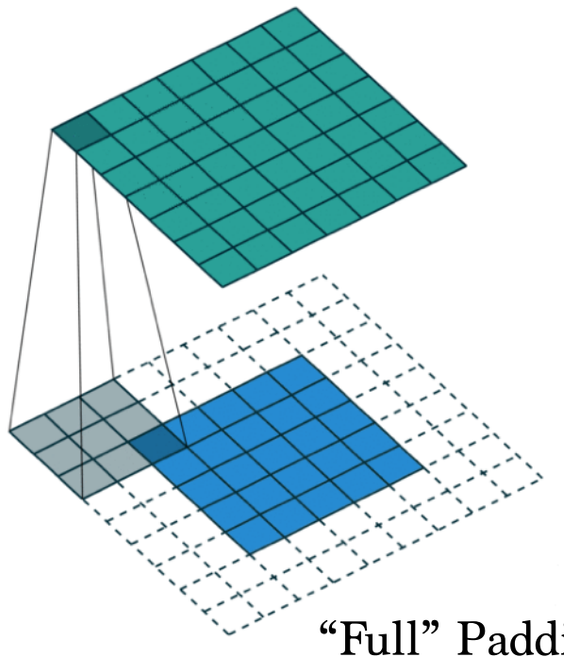


https://github.com/vdumoulin/conv_arithmetic

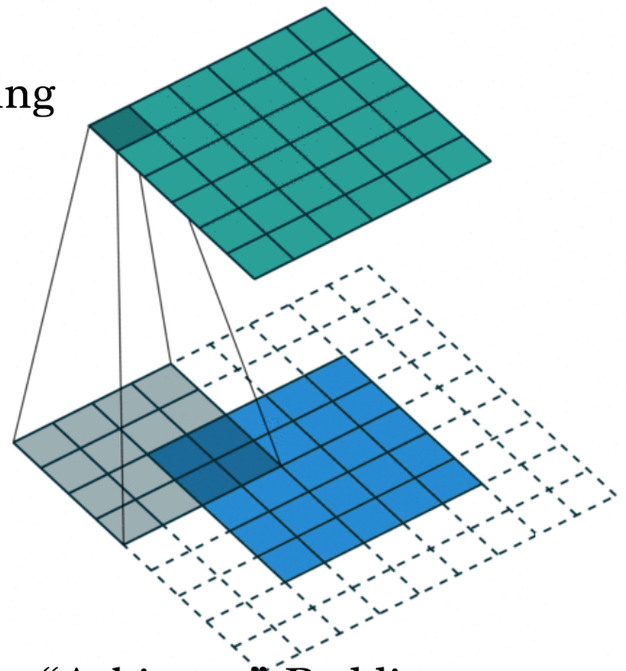
Convolutions 2D: Local Issues



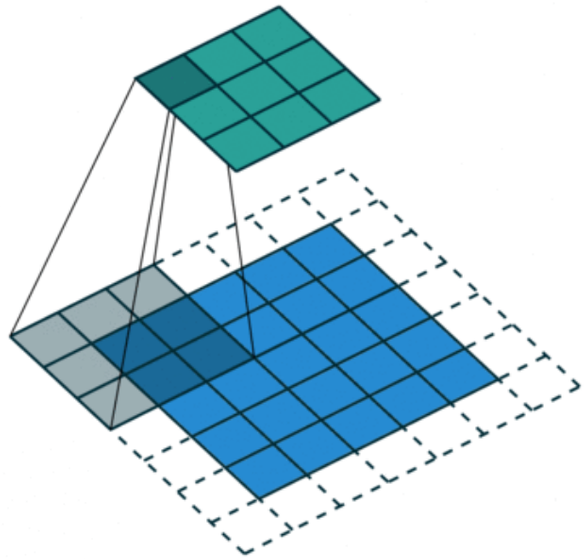
Strides



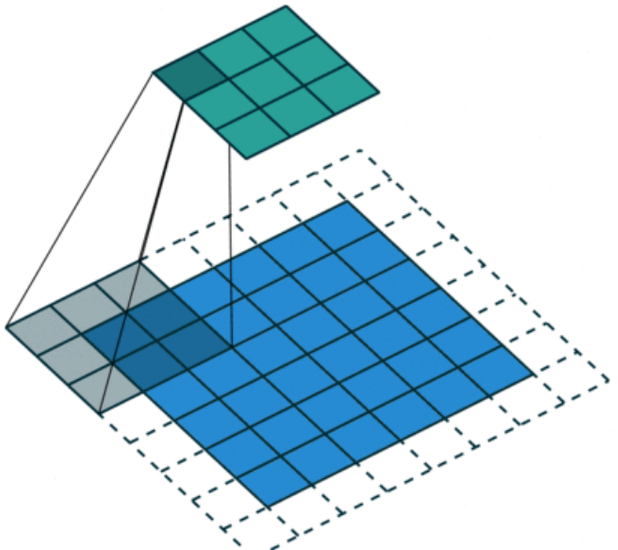
"Same" Padding



"Arbitrary" Padding

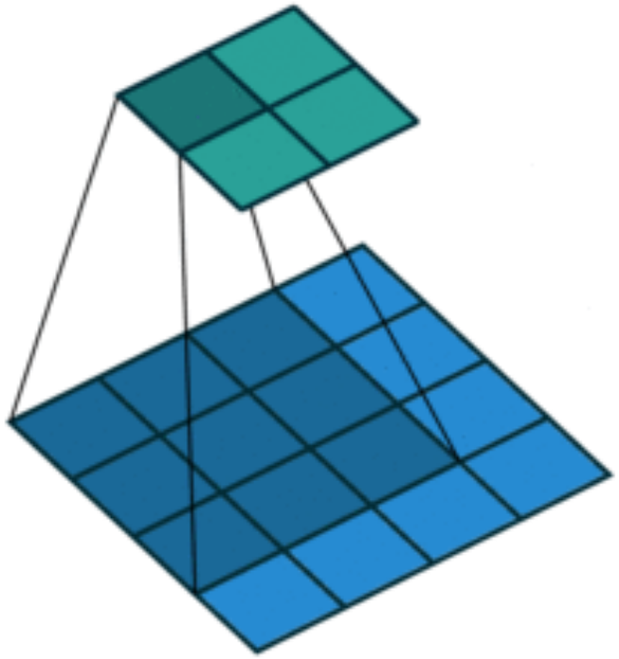


Both

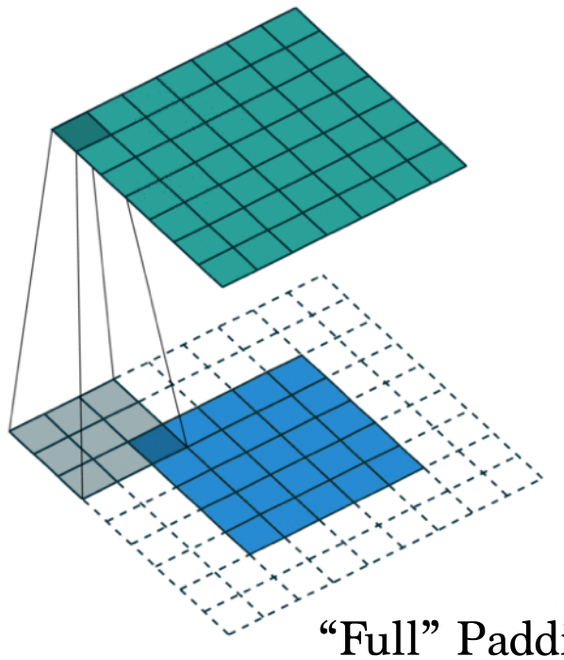


https://github.com/vdumoulin/conv_arithmetic

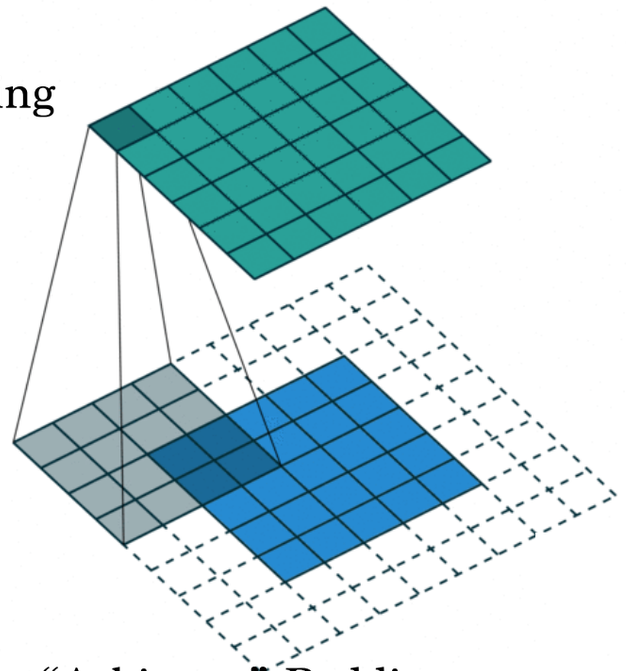
Convolutions 2D: Local Issues



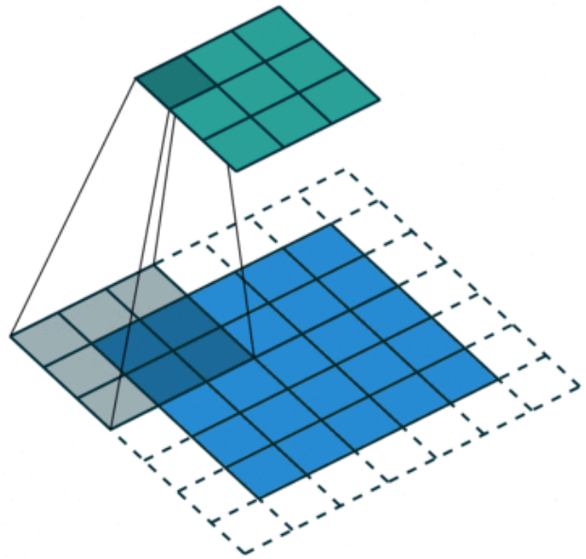
Strides



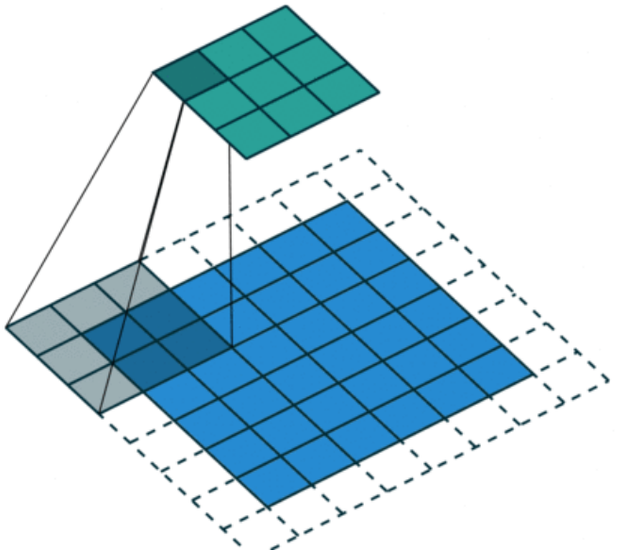
"Same" Padding



"Arbitrary" Padding

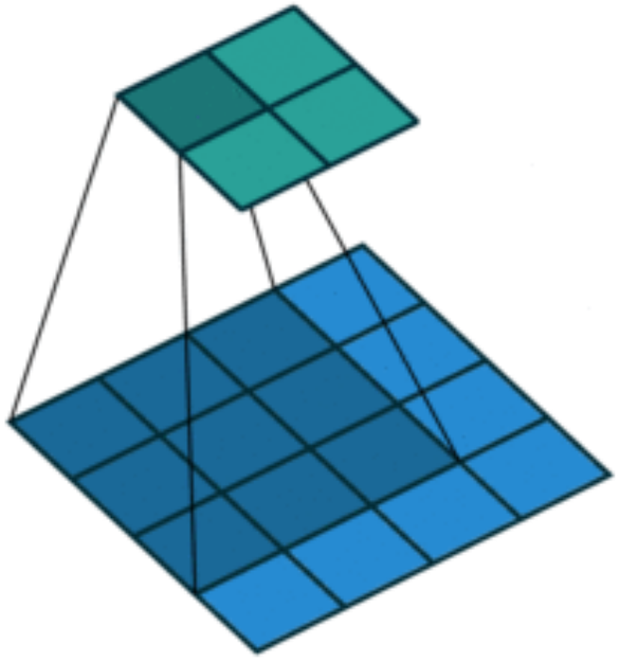


Both

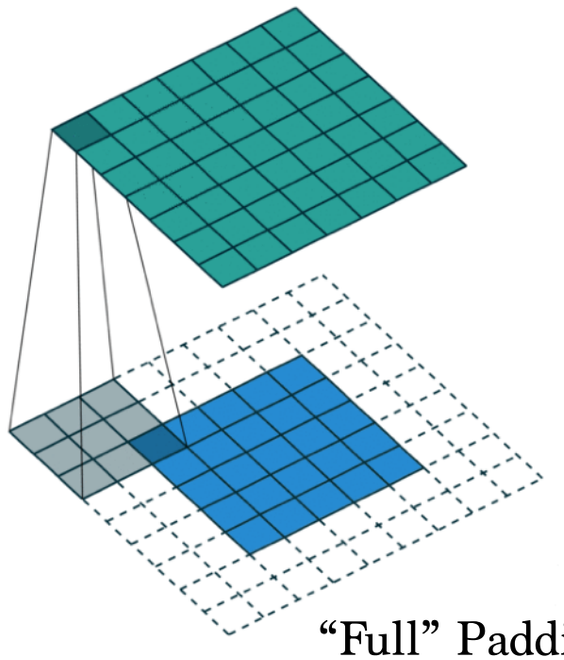


https://github.com/vdumoulin/conv_arithmetic

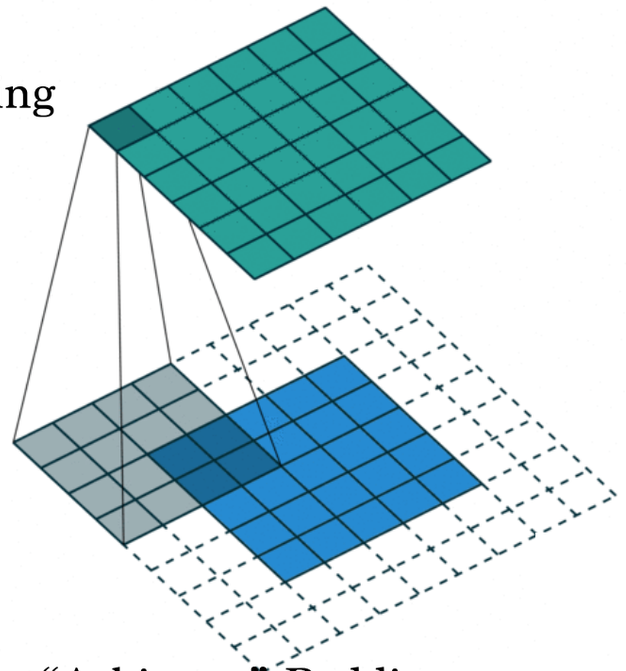
Convolutions 2D: Local Issues



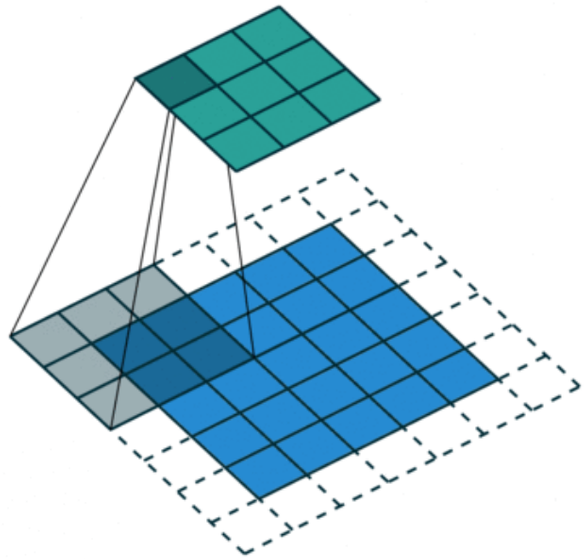
Strides



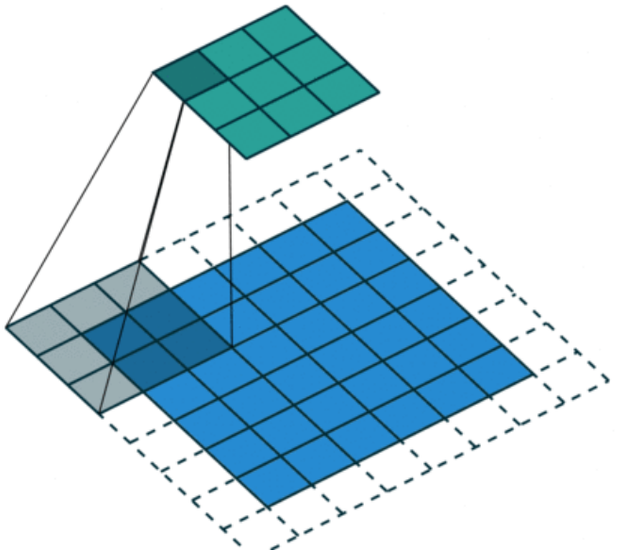
"Same" Padding



"Arbitrary" Padding

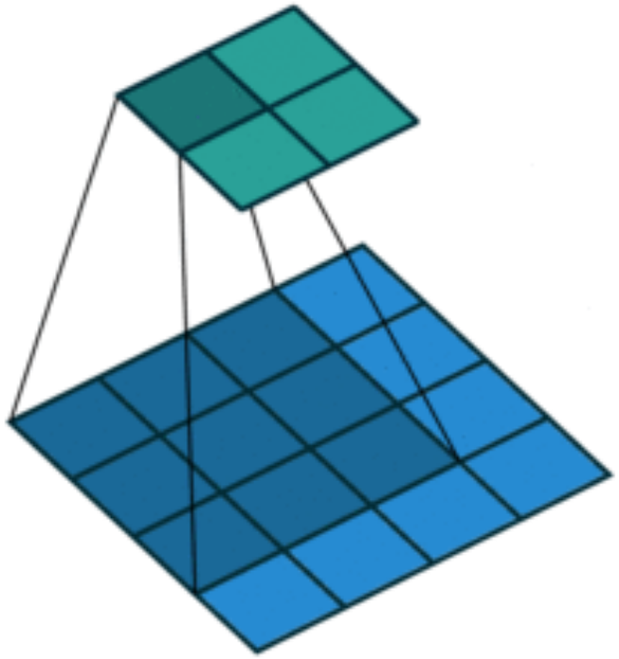


Both

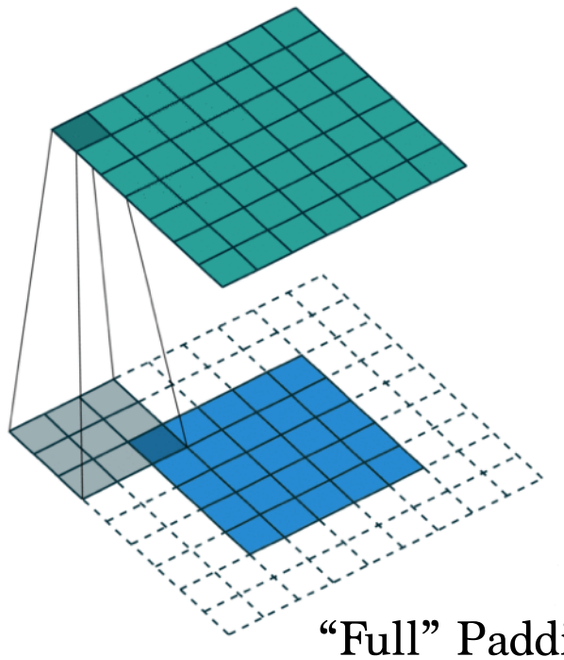


https://github.com/vdumoulin/conv_arithmetic

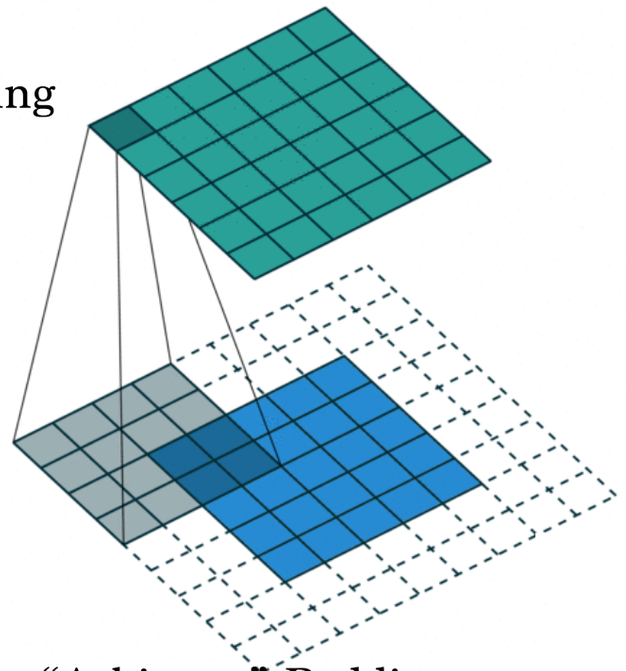
Convolutions 2D: Local Issues



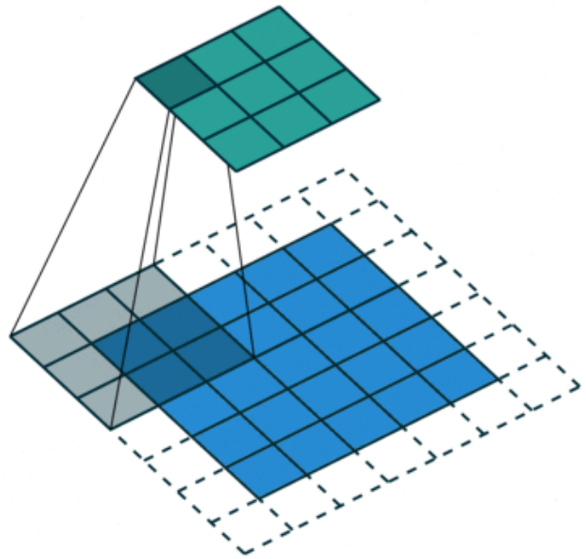
Strides



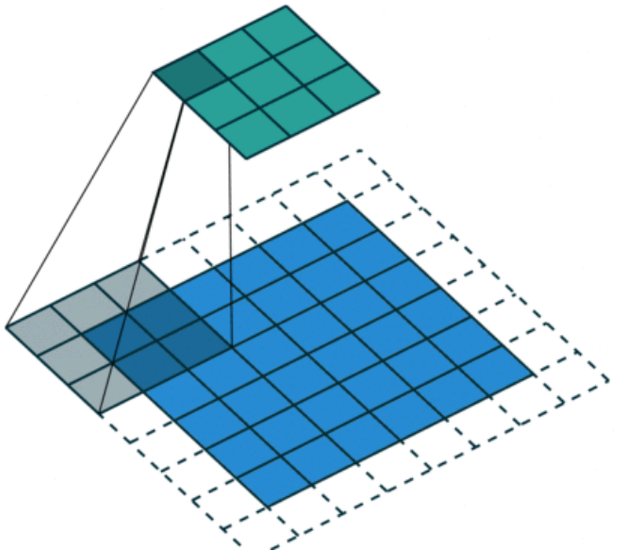
"Same" Padding



"Arbitrary" Padding

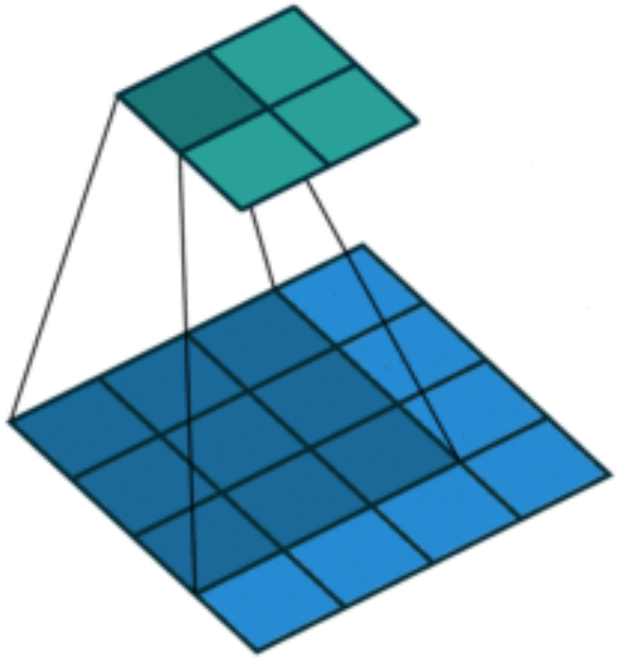


Both

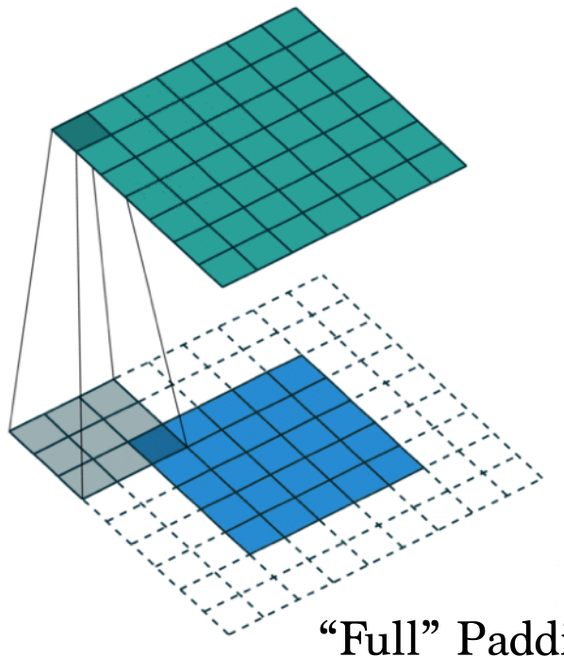


https://github.com/vdumoulin/conv_arithmetic

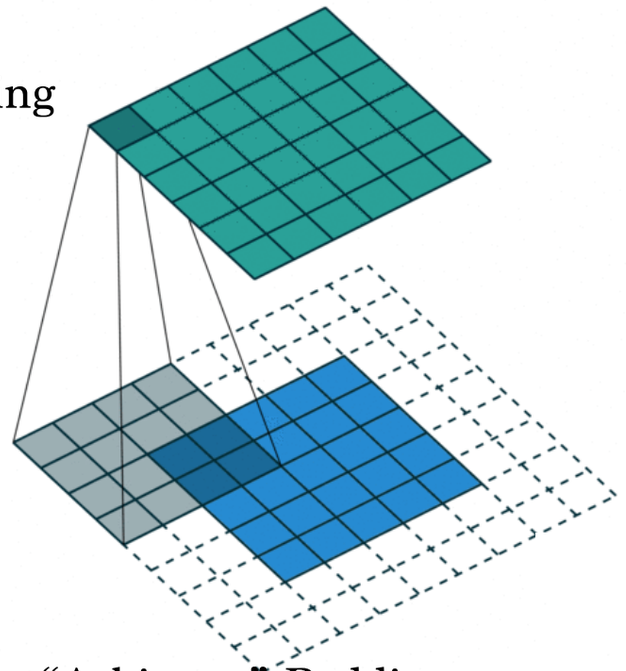
Convolutions 2D: Local Issues



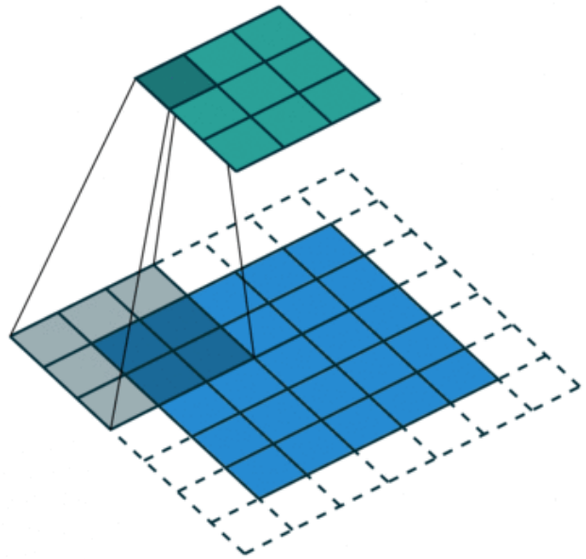
Strides



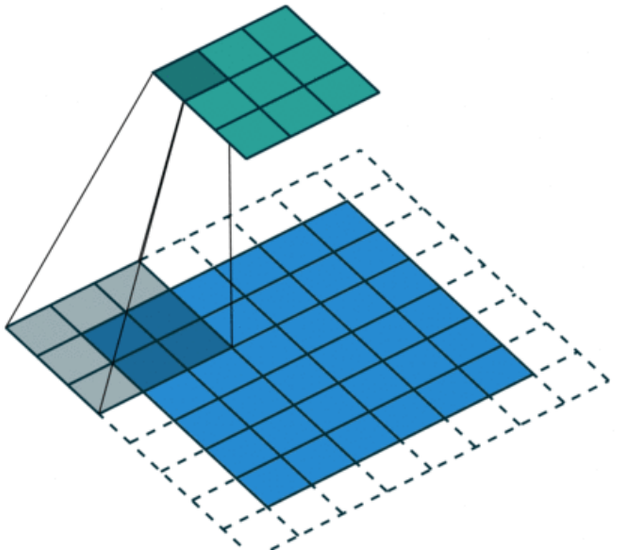
"Same" Padding



"Arbitrary" Padding

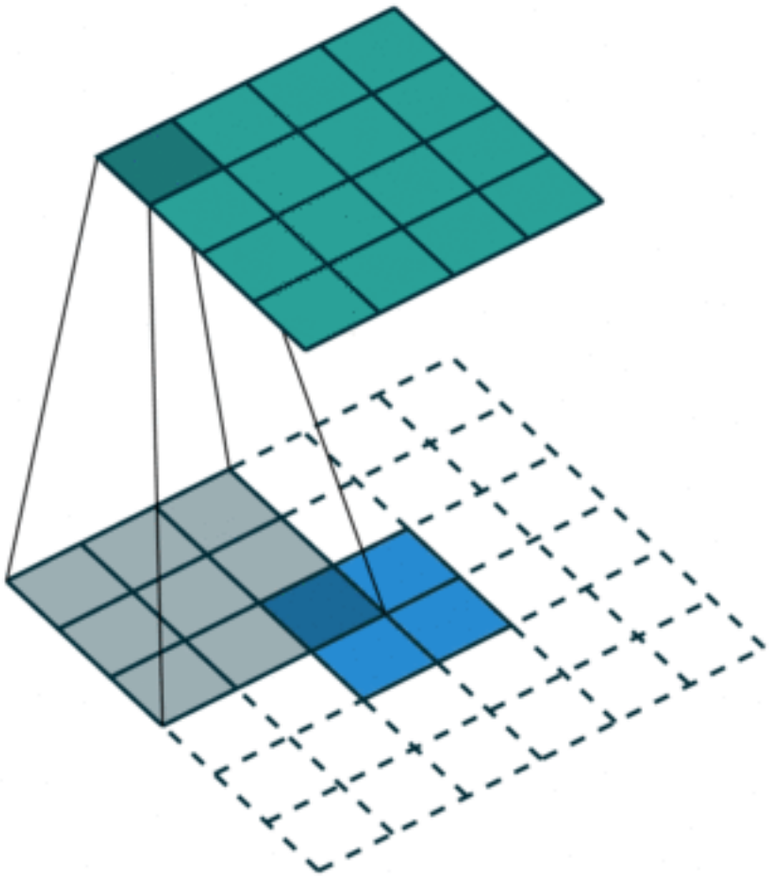


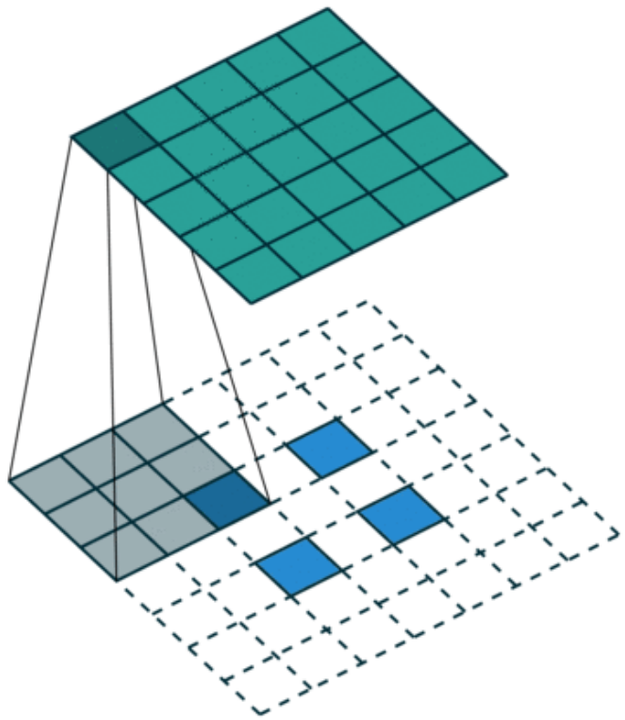
Both



https://github.com/vdumoulin/conv_arithmetic

De-convolutions 2D

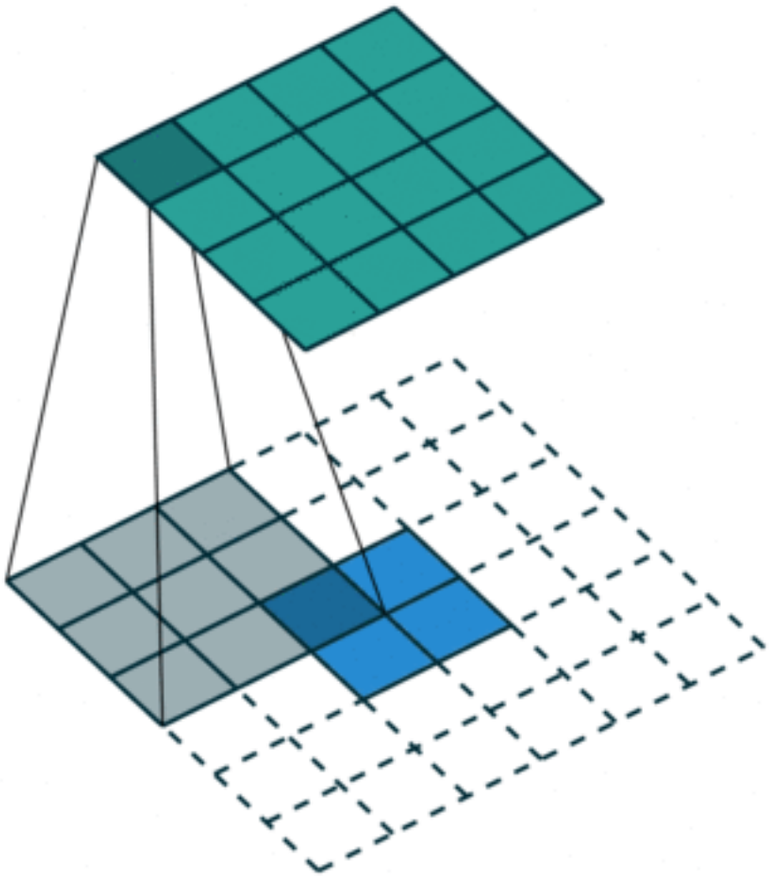


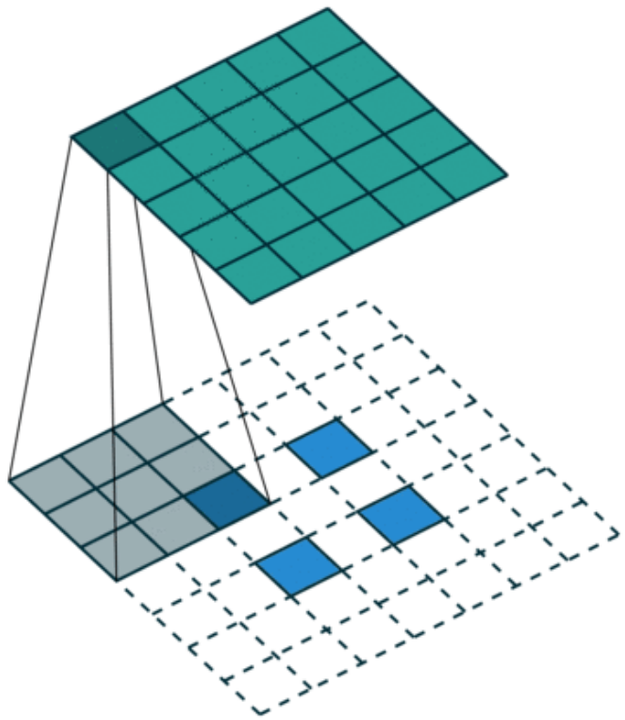


Strides

https://github.com/vdumoulin/conv_arithmetic

De-convolutions 2D

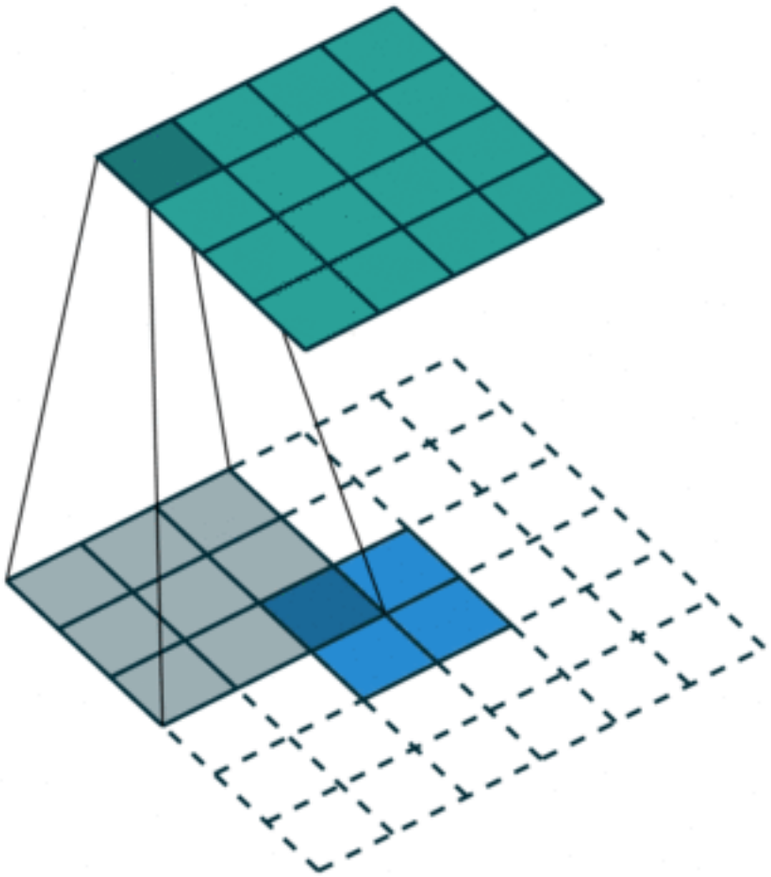


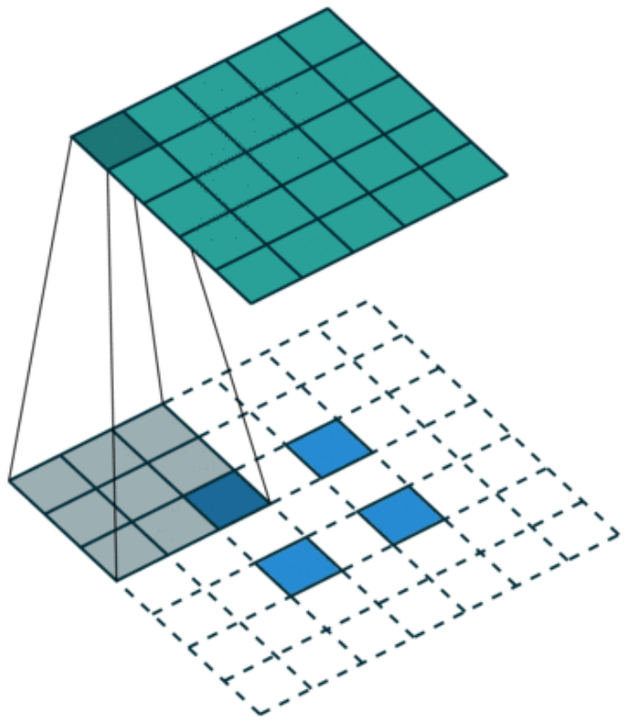


Strides

https://github.com/vdumoulin/conv_arithmetic

De-convolutions 2D

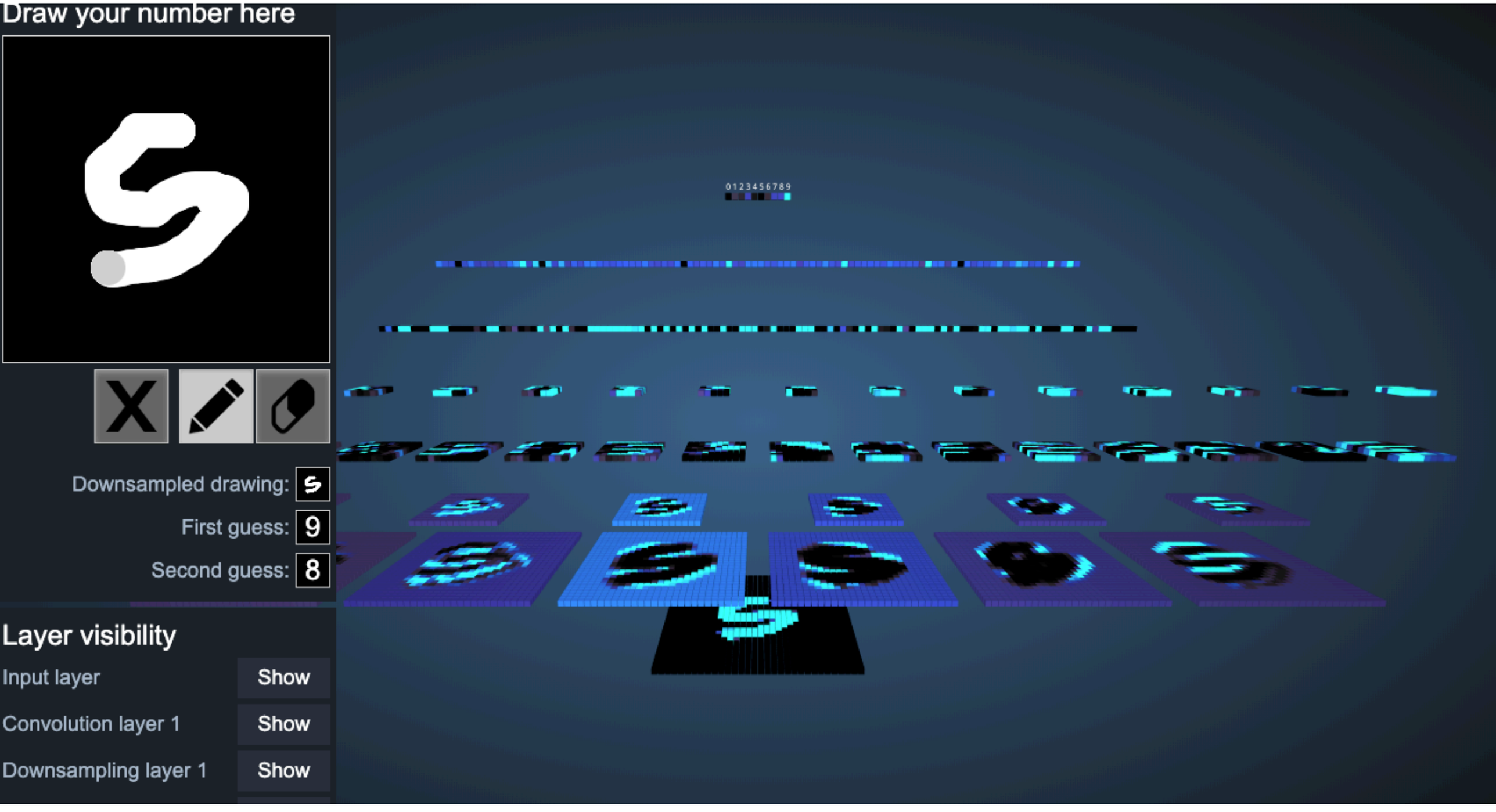




Strides

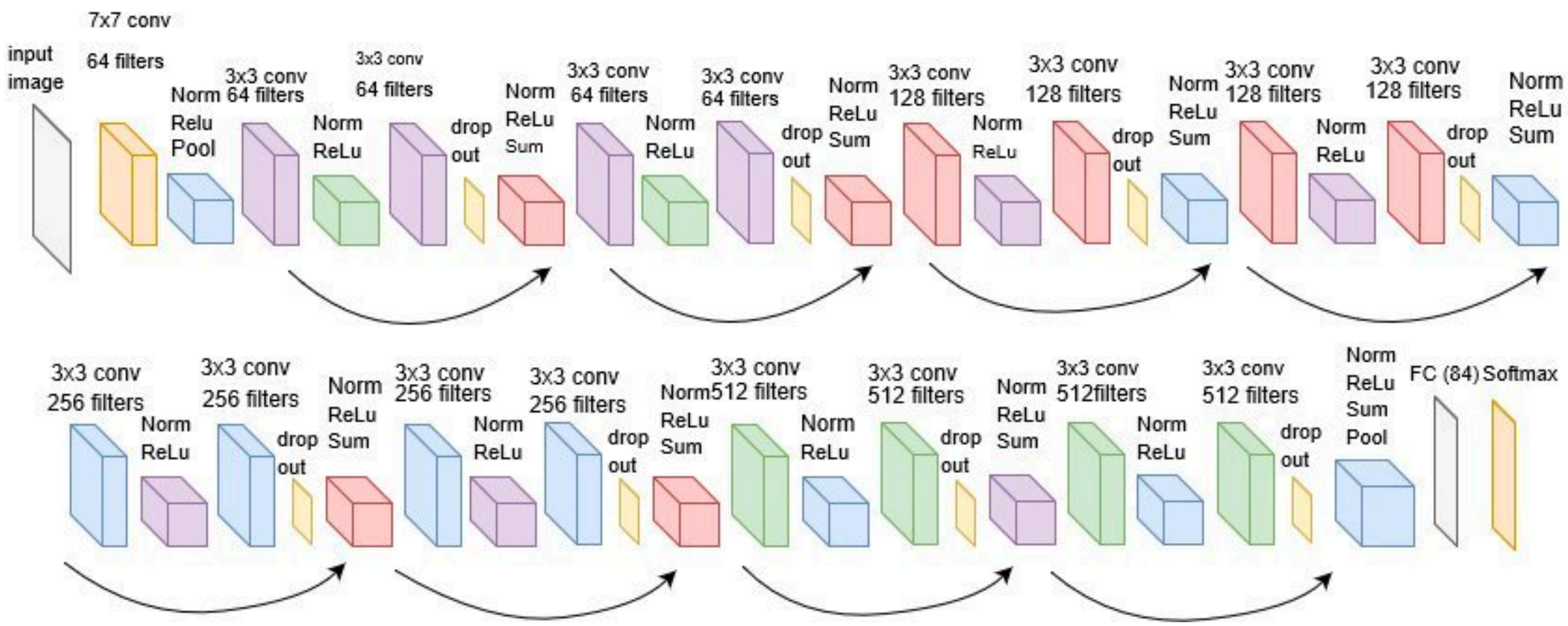
https://github.com/vdumoulin/conv_arithmetic

Animation



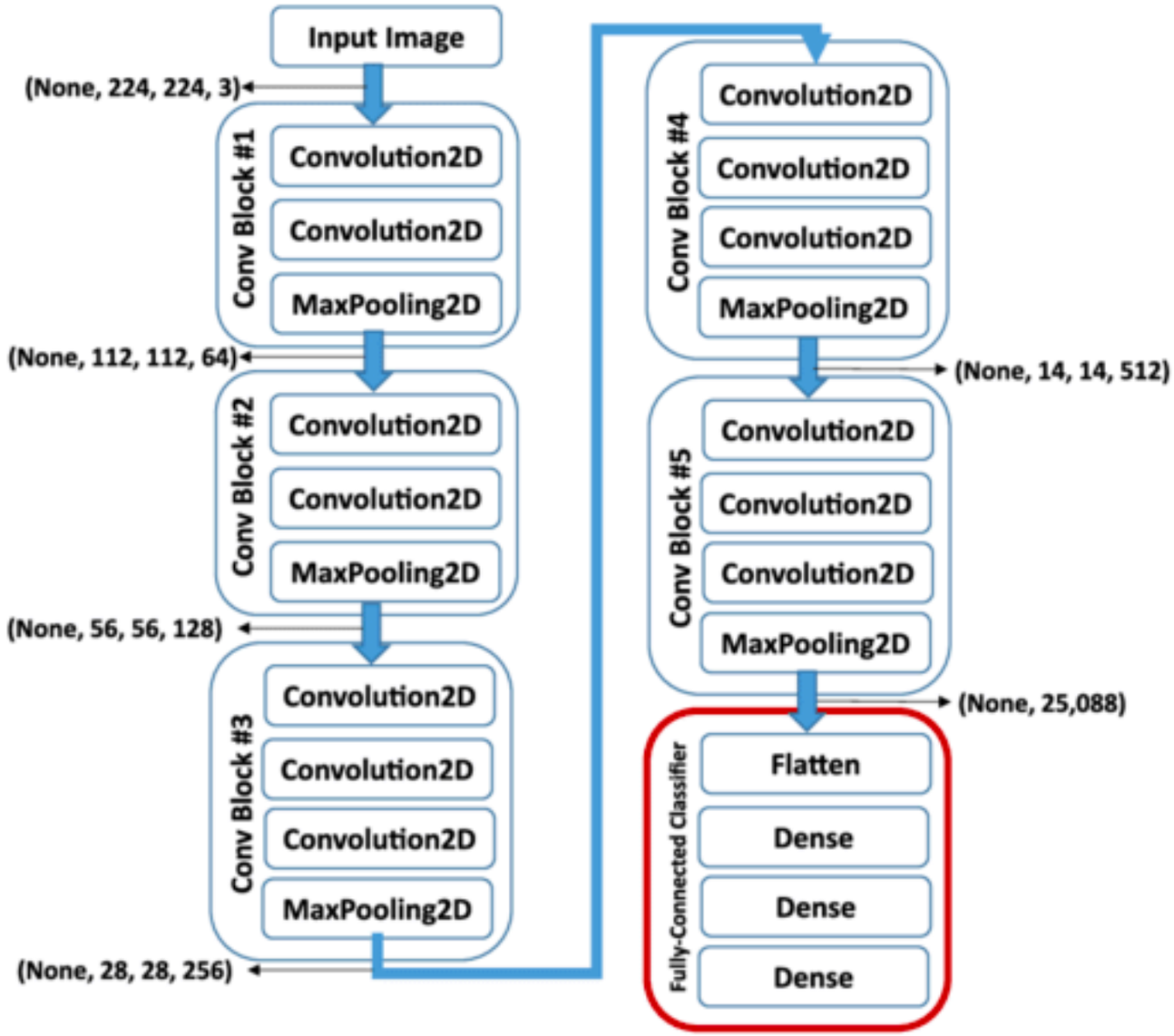
https://www.cs.ryerson.ca/~aharley/vis/conv/

Example : Resnet 18



https://github.com/vdumoulin/conv_arithmetic

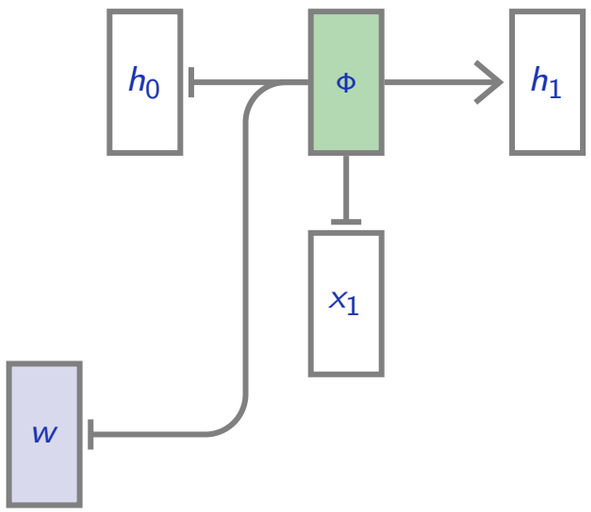
Example : VGG 16



https://github.com/vdumoulin/conv_arithmetic

Varying Length Inputs with Recursion

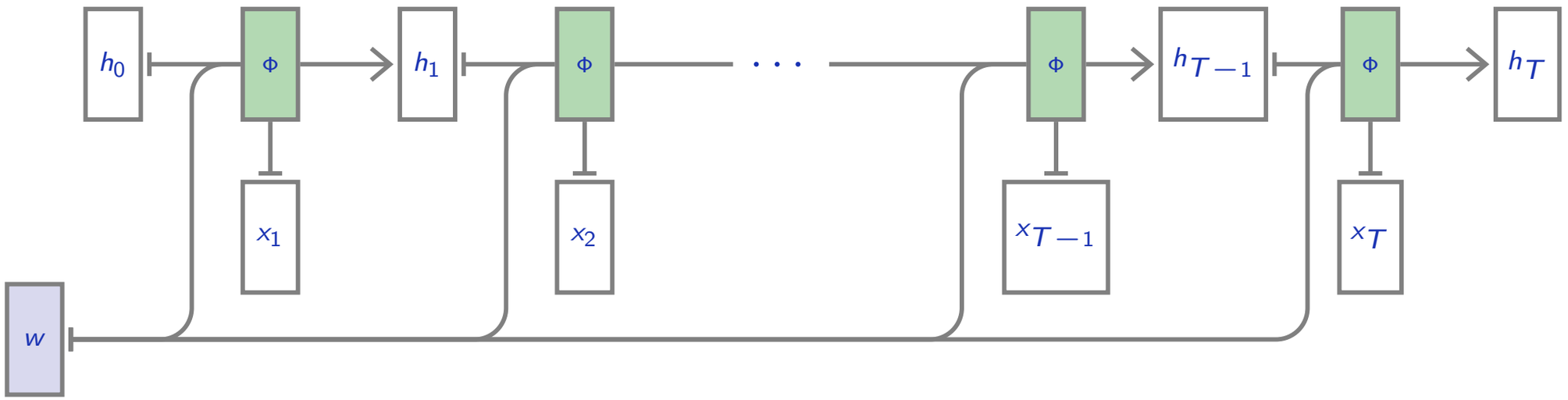
$$x = (x_1, x_2, \ldots, x_T)$$



François Fleuret:https://fleuret.org/ee559/materials/ee559-slides-12-1-RNN-basics.pdf

Varying Length Inputs with Recursion

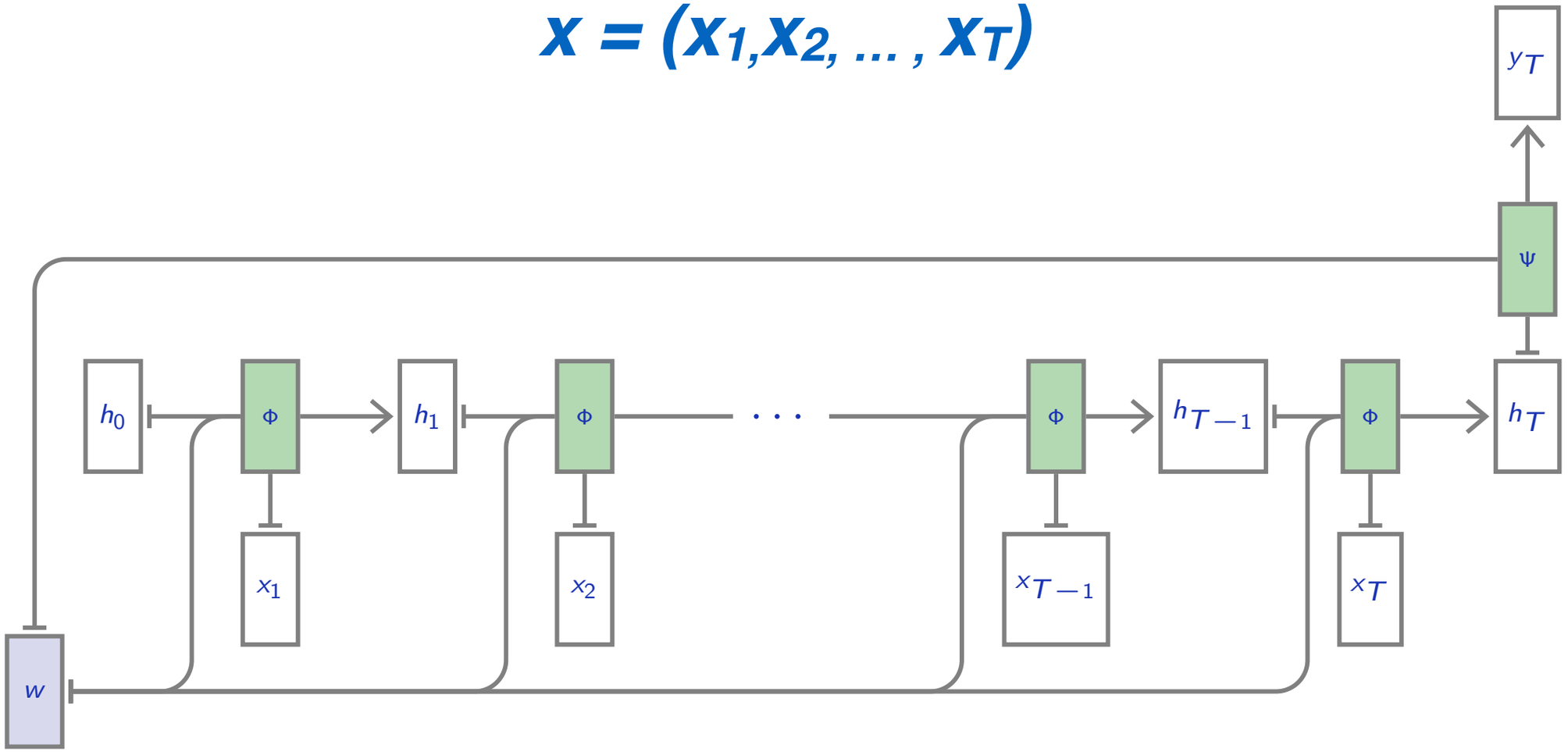
$$x = (x_1, x_2, \ldots, x_T)$$



François Fleuret:https://fleuret.org/ee559/materials/ee559-slides-12-1-RNN-basics.pdf

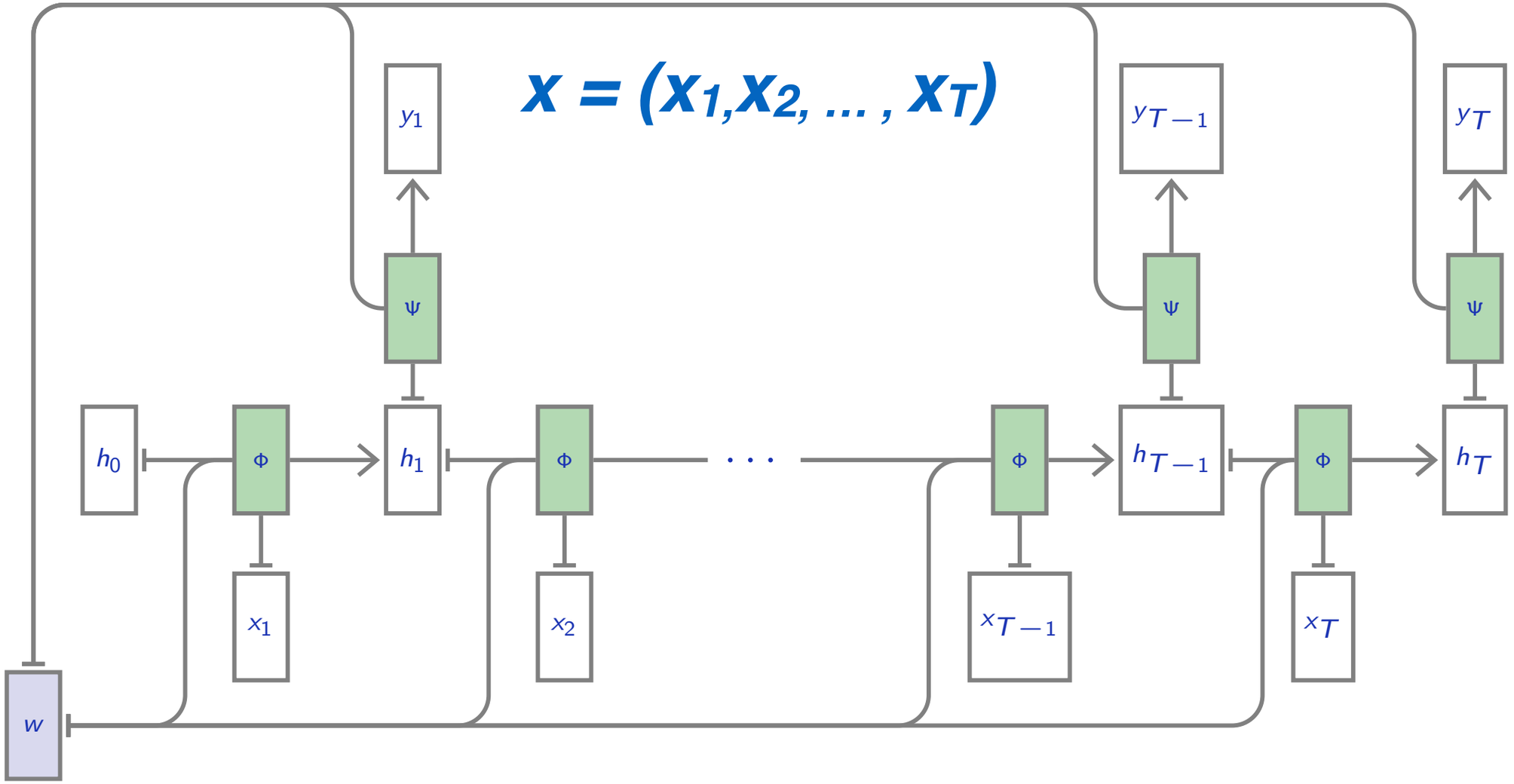
Varying Length Inputs with Recursion

$$x = (x_1, x_2, \ldots, x_T)$$



François Fleuret:https://fleuret.org/ee559/materials/ee559-slides-12-1-RNN-basics.pdf

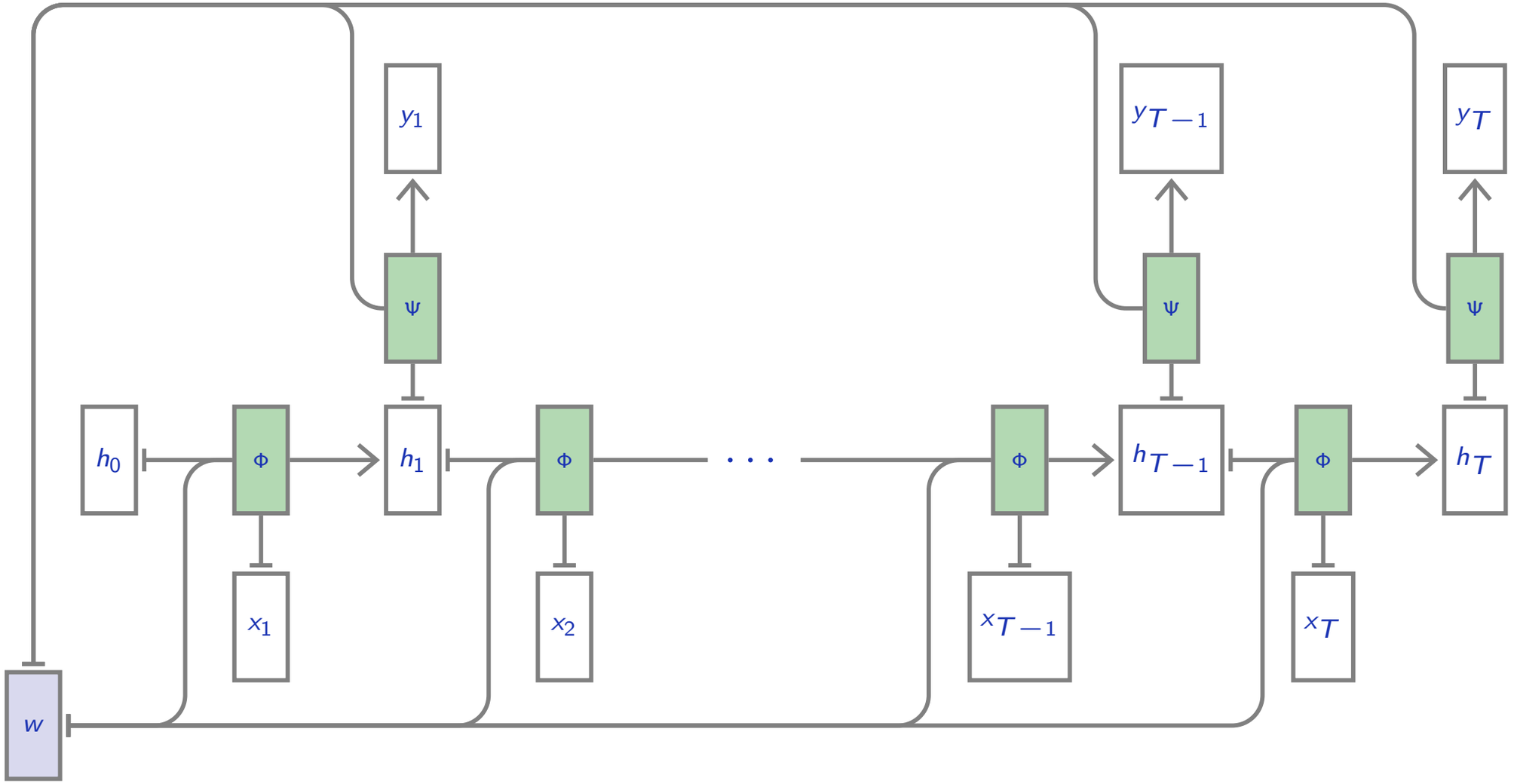
Varying Length Outputs



$$y = (y_1, y_2, \ldots, y_T)$$

François Fleuret:https://fleuret.org/ee559/materials/ee559-slides-12-1-RNN-basics.pdf

Varying Length Outputs



Even though the number of steps T depends on x, this is a standard graph of tensor operations, and autograd can deal with it as usual.

François Fleuret:https://fleuret.org/ee559/materials/ee559-slides-12-1-RNN-basics.pdf

Schedule

1. Introduction, Backpropagation
   Empirical risk minimization (R-ERM), specificities of deep learning, automatic differentiation.
2. Modeling blocks: convolution, recursion.
   Architecture of DL, building blocks from signal processing, mechanisms to handle varying-length inputs
3. Accelerators, Non-convex optimization, GANs / (V)AEs.
4. Graph NN / Theory of NN

Accelerators

Moore's Law

"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue"

Gordon Moore (Intel), 1965

"OK, maybe a factor of two every two years."

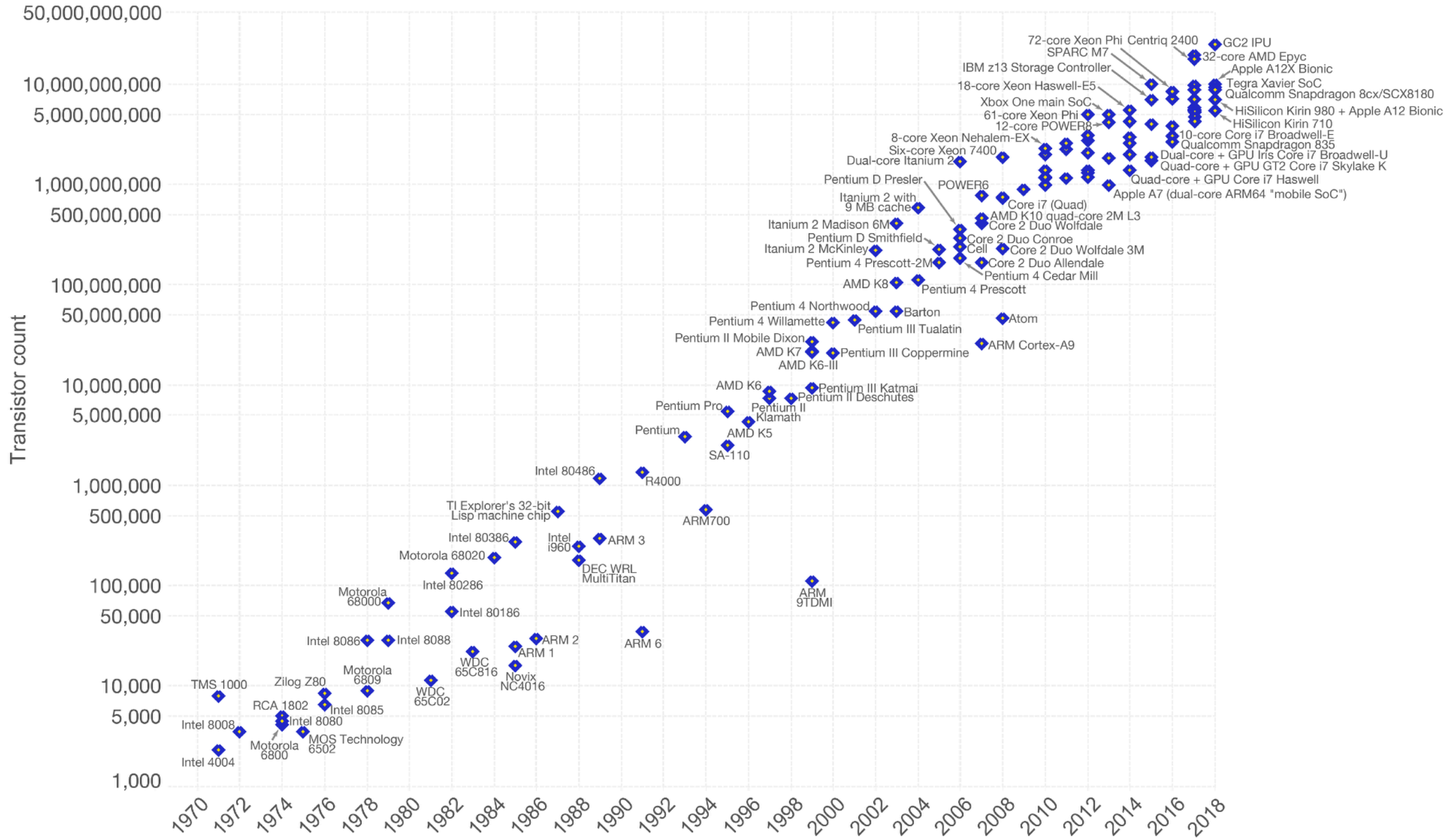
Gordon Moore (Intel), 1975 [paraphrased]

Moore’s Law

Moore’s Law – The number of transistors on integrated circuit chips (1971-2018)

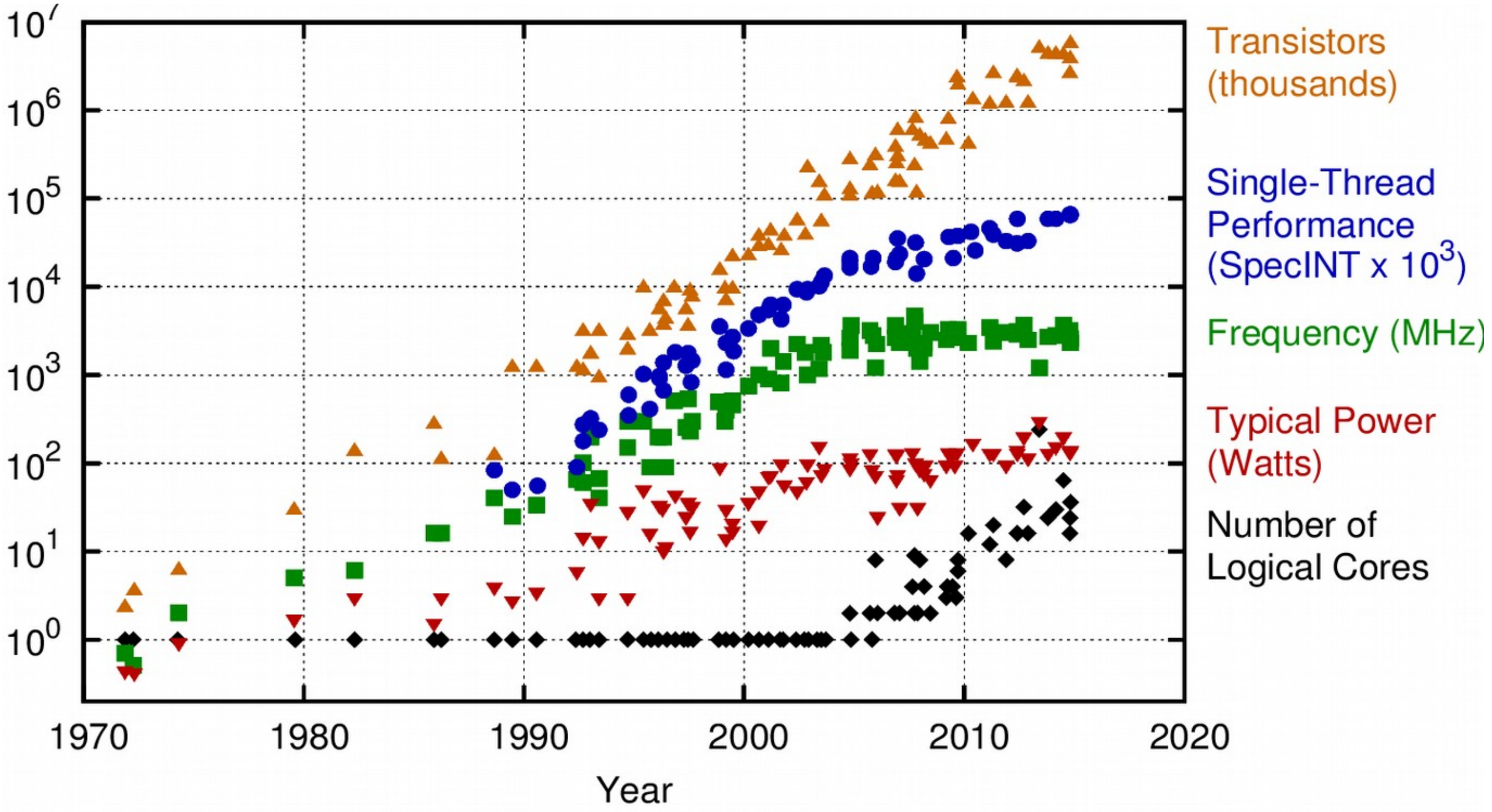
Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

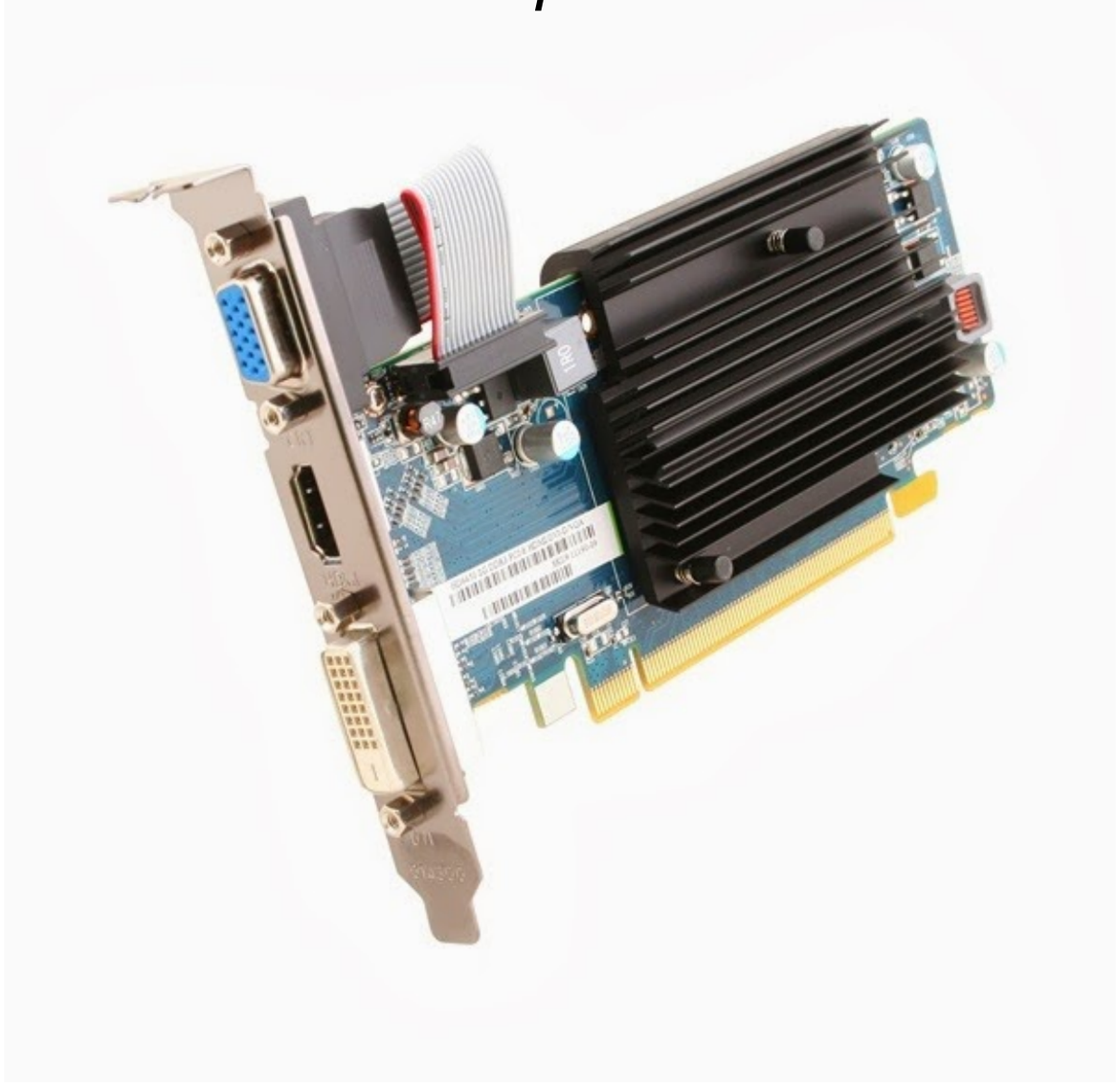
Moore's Law



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Solution: GPU

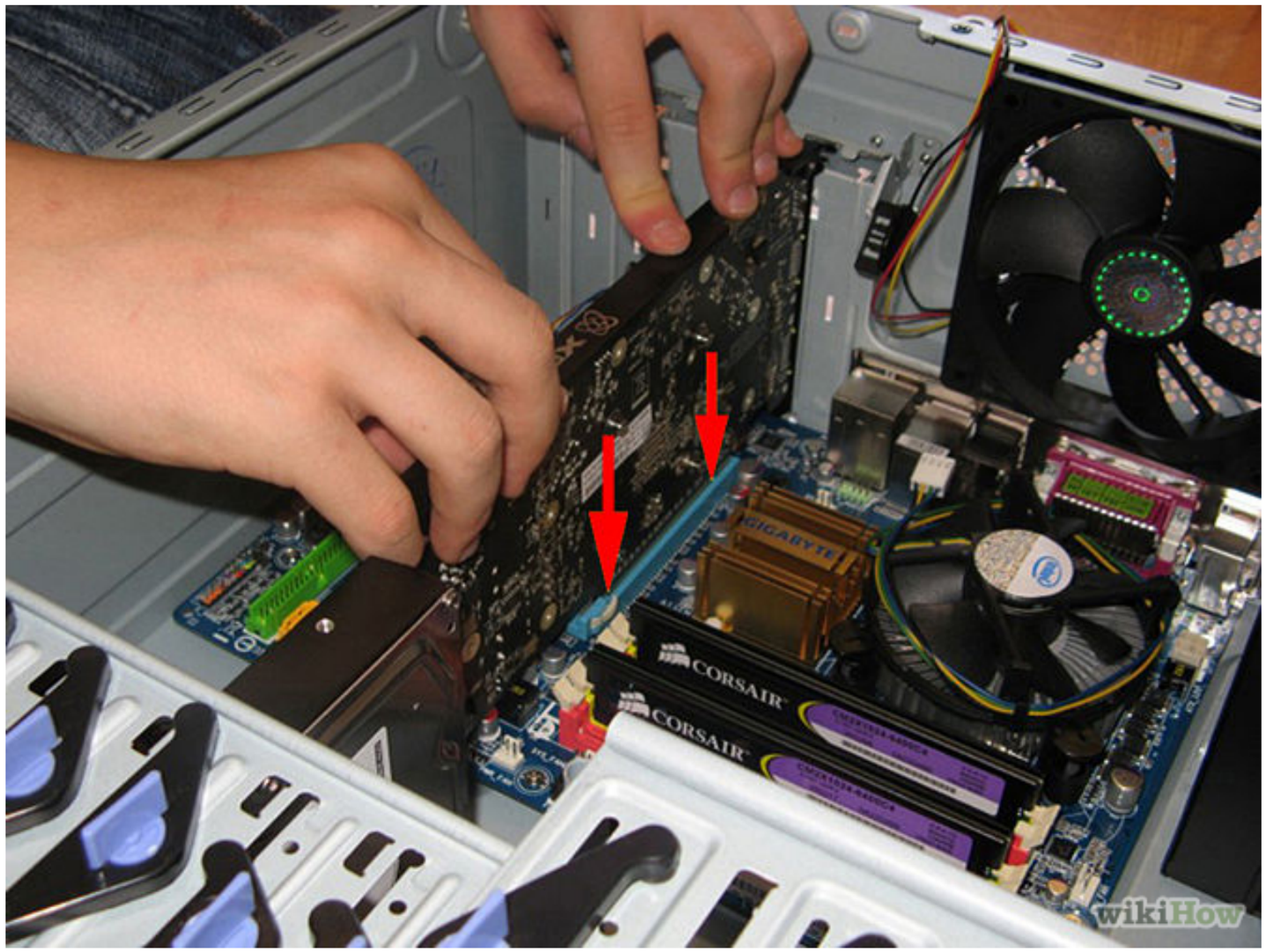
used to be a small piece of hardware…



GPU = Graphics Processing Unit

Solution: GPU

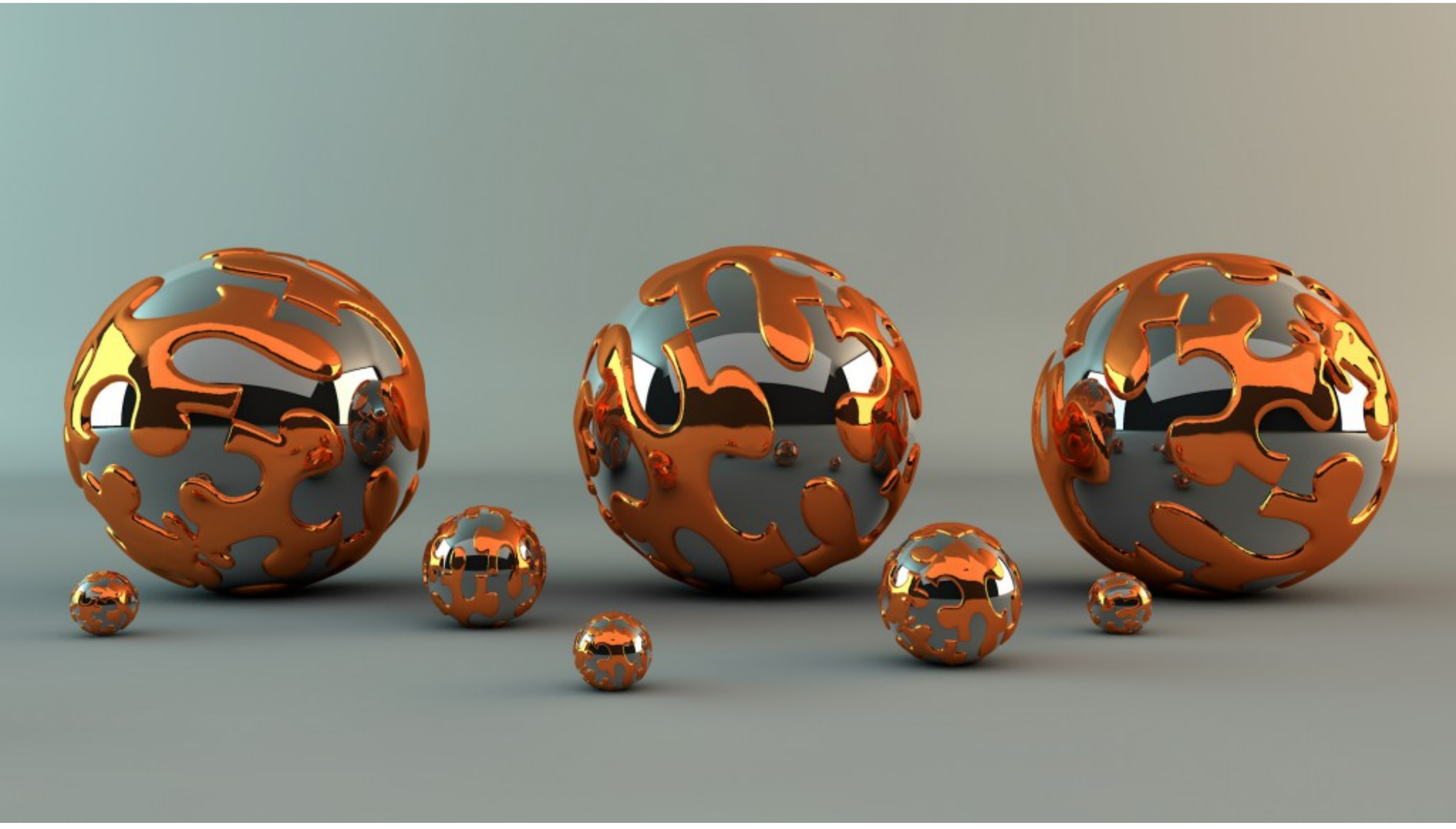
… plugged into computer, with video output…



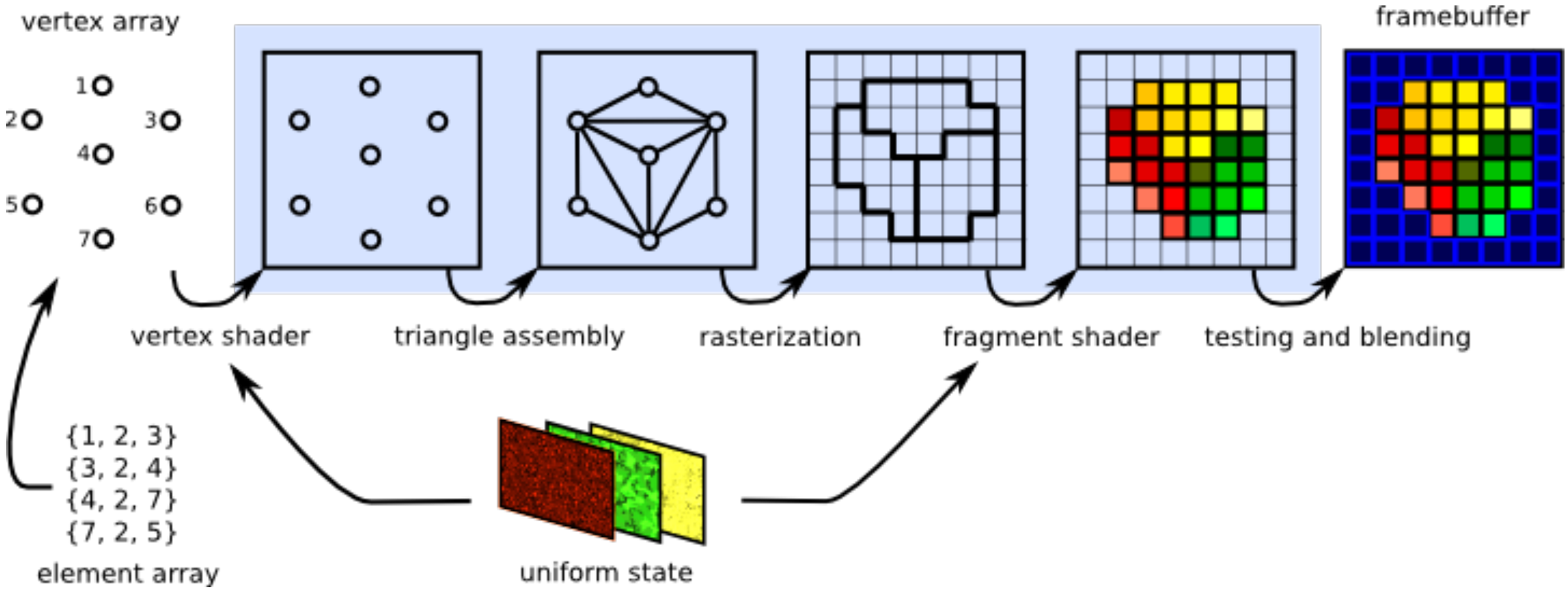
Solution: GPU

…of interest to gamers and video editors.

Graphics

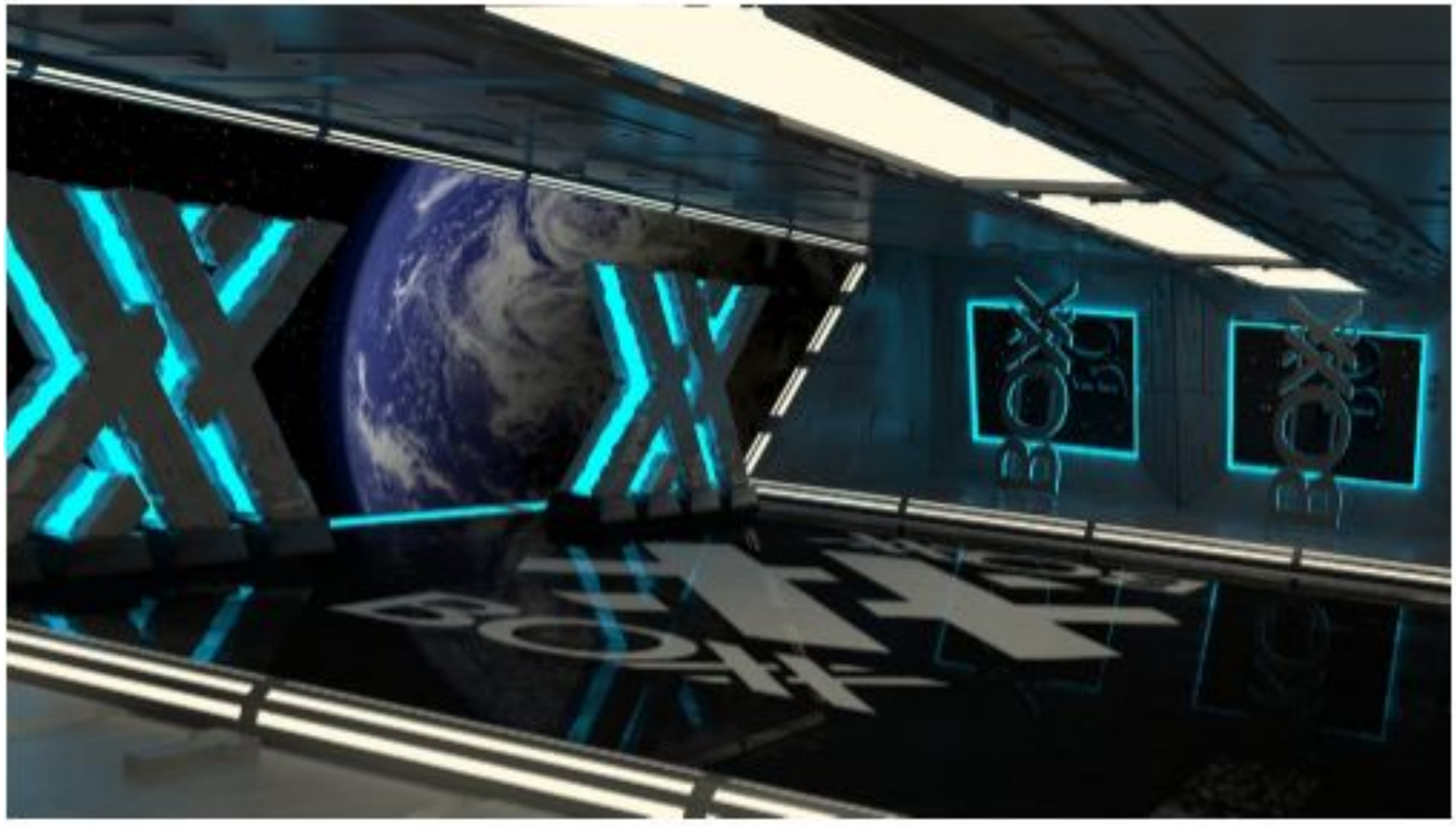


Graphics


vertex array
framebuffer
1
2
3
4
5
6
7
vertex shader
triangle assembly
rasterization
fragment shader
testing and blending
{1, 2, 3}
{3, 2, 4}
{4, 2, 7}
{7, 2, 5}
element array
uniform state

3D Rendering

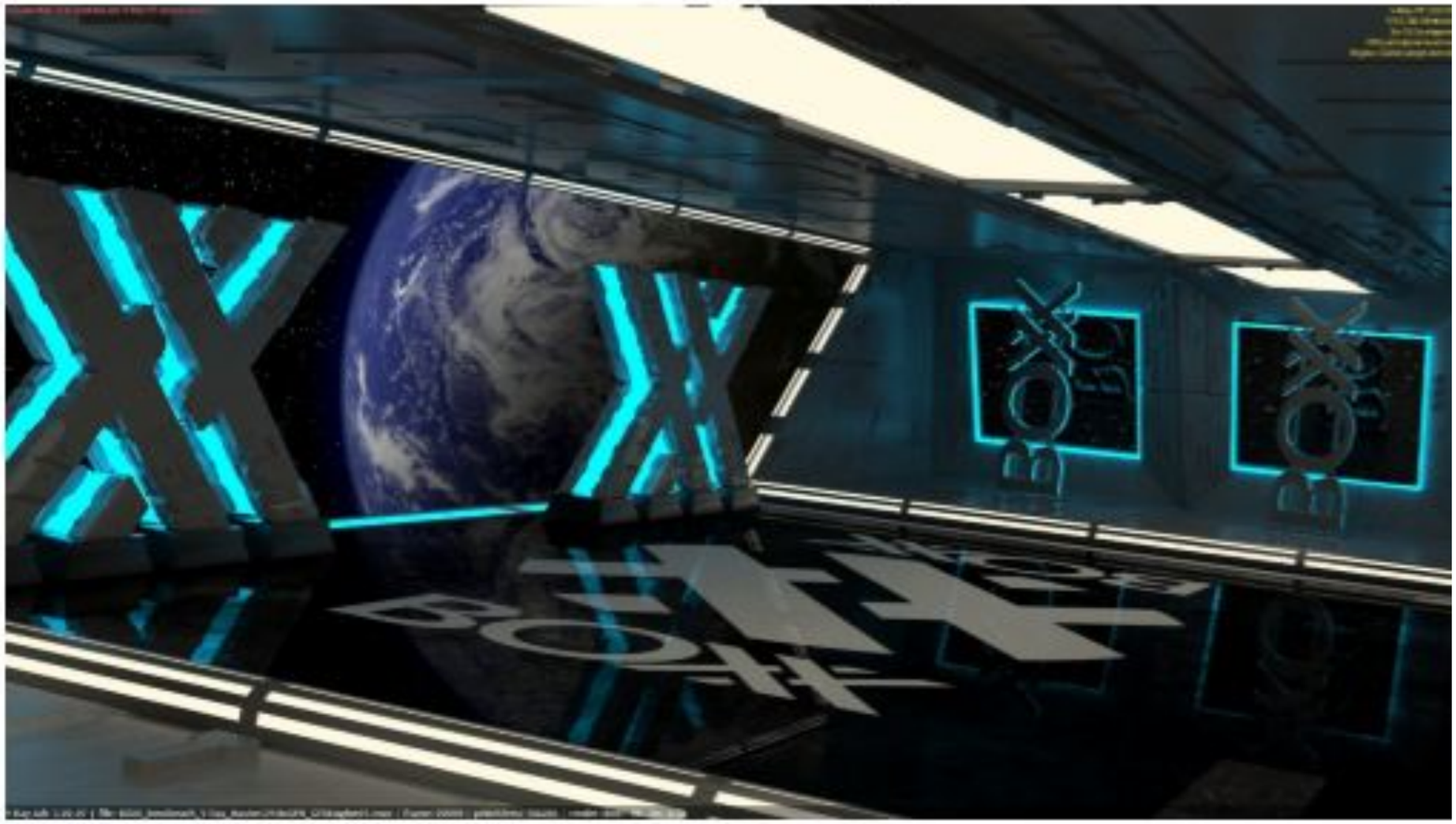
Rendered with V-Ray Advanced CPU



3.4 GHz 8 core Intel® Xeon®
Image Quality = 11.35
Render Time = 19 minutes 11 seconds

3D Rendering

Rendered with V-Ray RT GPU



High-end NVIDIA GPU with 2688 CUDA cores
Image Quality = 11.35
Render Time = 3 minutes 4 seconds

What are GPUs

Definition: GPU

A **programmable logic chip** (processor) specialized for **display functions.** The GPU renders images, animations and video for the computer's screen. GPUs are located on plug-in cards, in a chipset on the motherboard or in the same chip as the CPU.

A GPU performs parallel operations. Although it is used for 2D data as well as for zooming and panning the screen, a GPU is essential for smooth decoding and **rendering of 3D animations.**

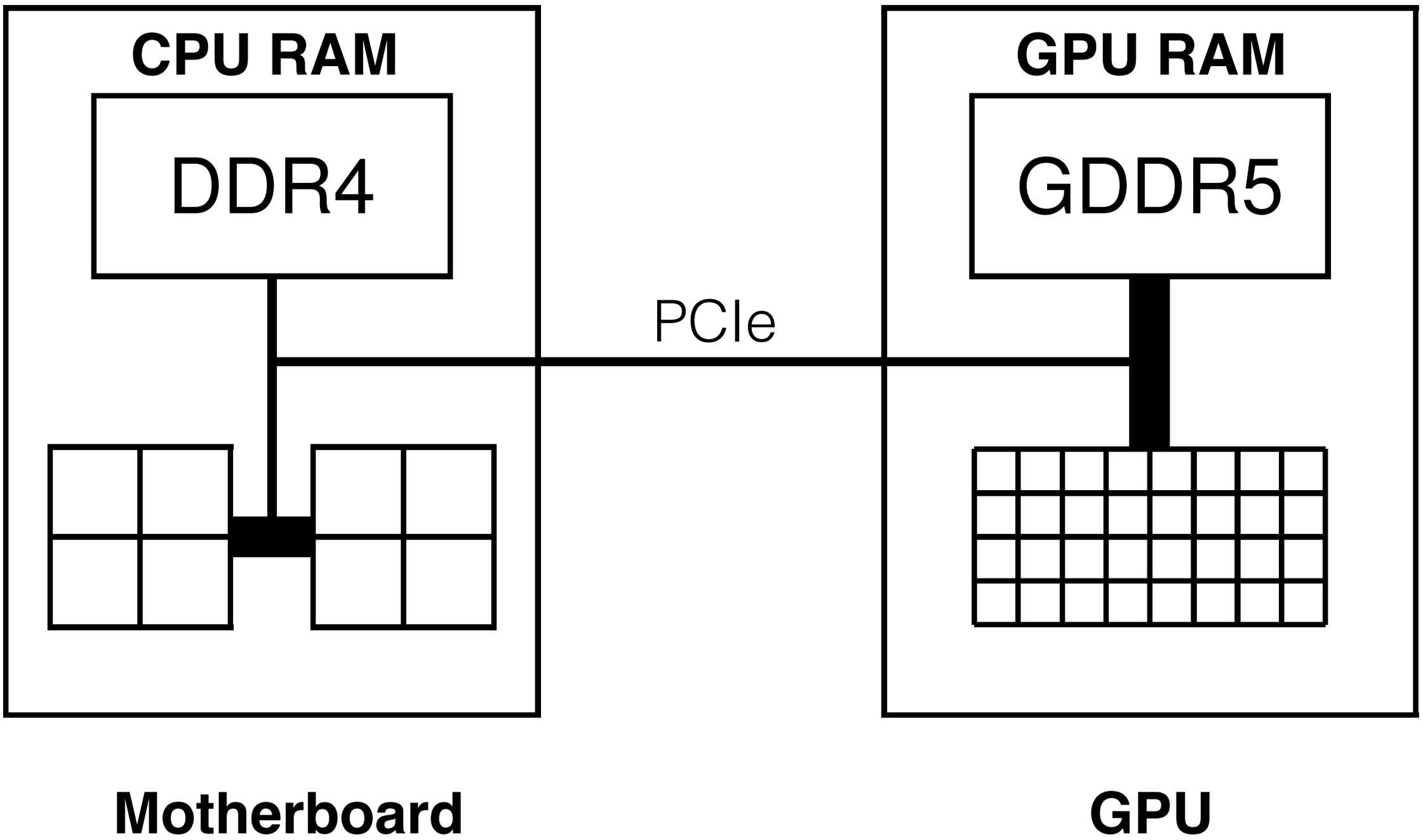
What are **GP**GPUs

Definition: **GP**GPU

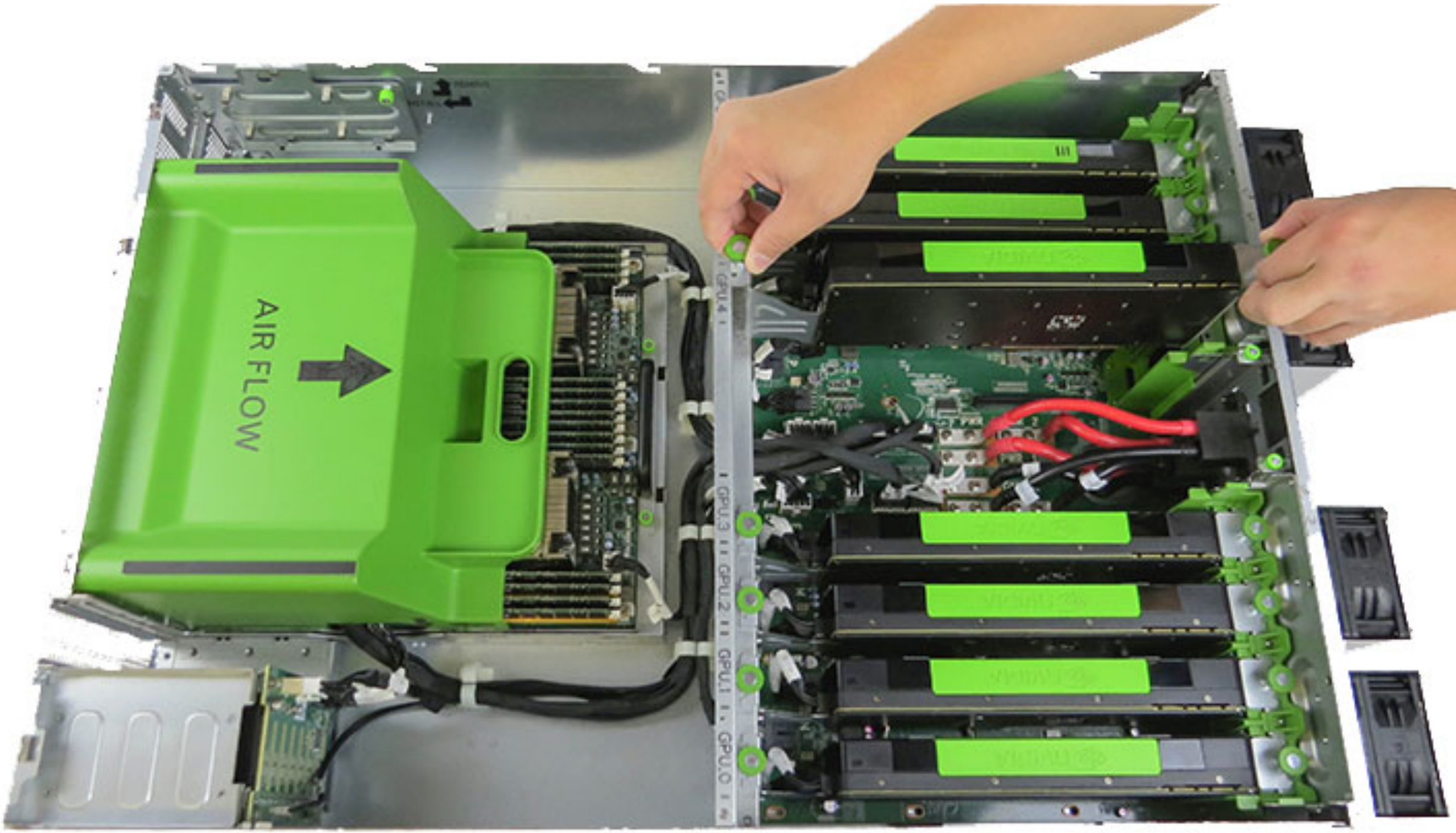
Using a GPU for general-purpose (**GP**) parallel processing applications rather than rendering images for the screen.

For fast results, applications such as sorting, **matrix algebra**, image processing and physical modeling require multiple sets of data to be processed in parallel.

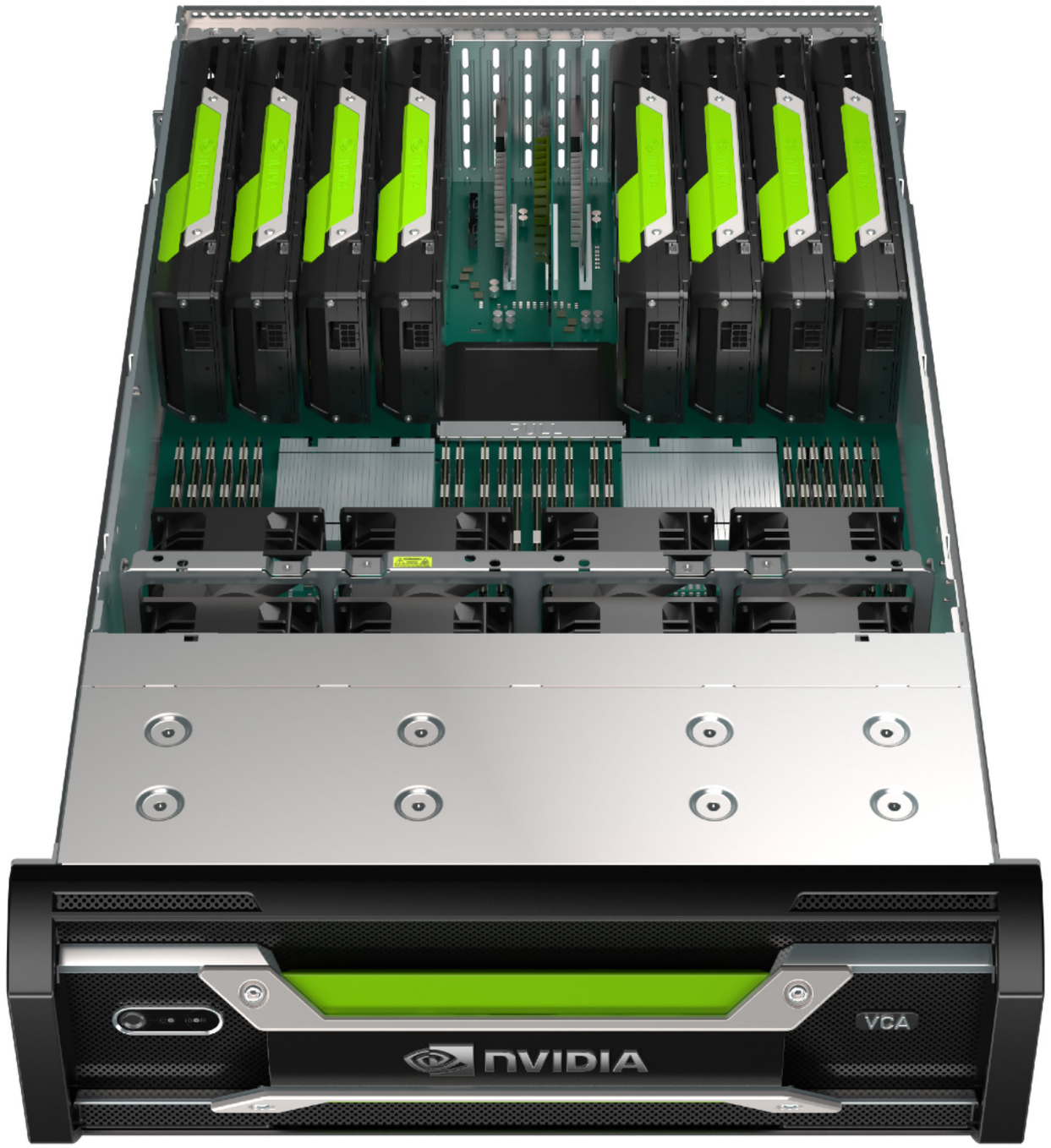
At very basic level...



In the real world



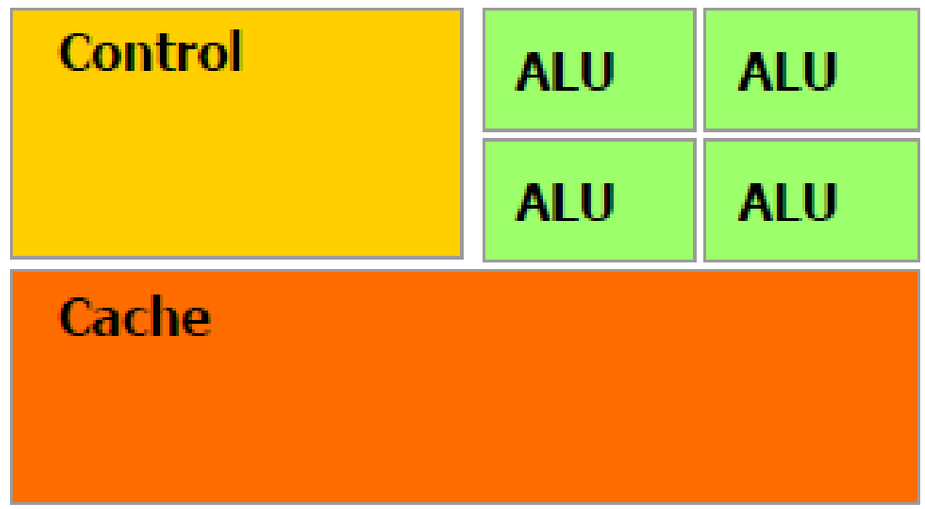
In the real world



CPU

Single Instructions, Multiple Data (SIMD)

large data-**caching**
large flow **control units**
few Arithmetic Logical Units (ALU, cores), but fast



Example: Intel Xeon E5-2670 CPU

8 cores (16 threads)
2.6 GHz
2.3 billion transistors
20 MB on chip cache
<u>Flexible</u> DRAM size

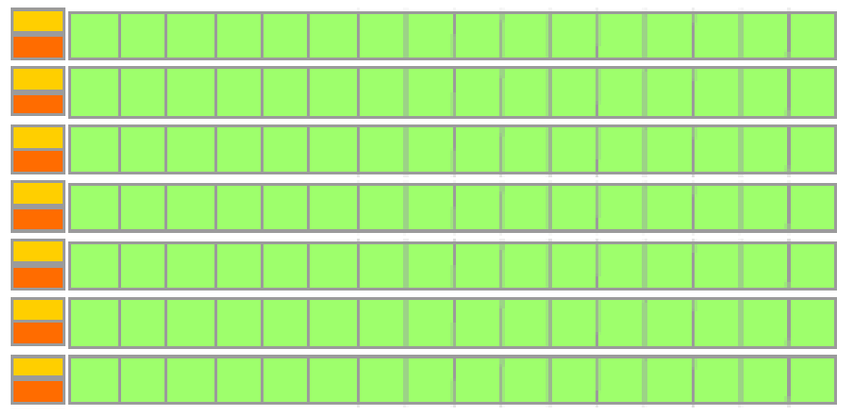
GPU

Single Instructions, Multiple Threads (SIMT)

small **cache, control flow**

Many ALUs (cores), slow.

Highly parallel.



Example: Kepler K20x GPU

2688 (14 x 192) cores

0.73 GHz

28nm features

7.1 billion transistors

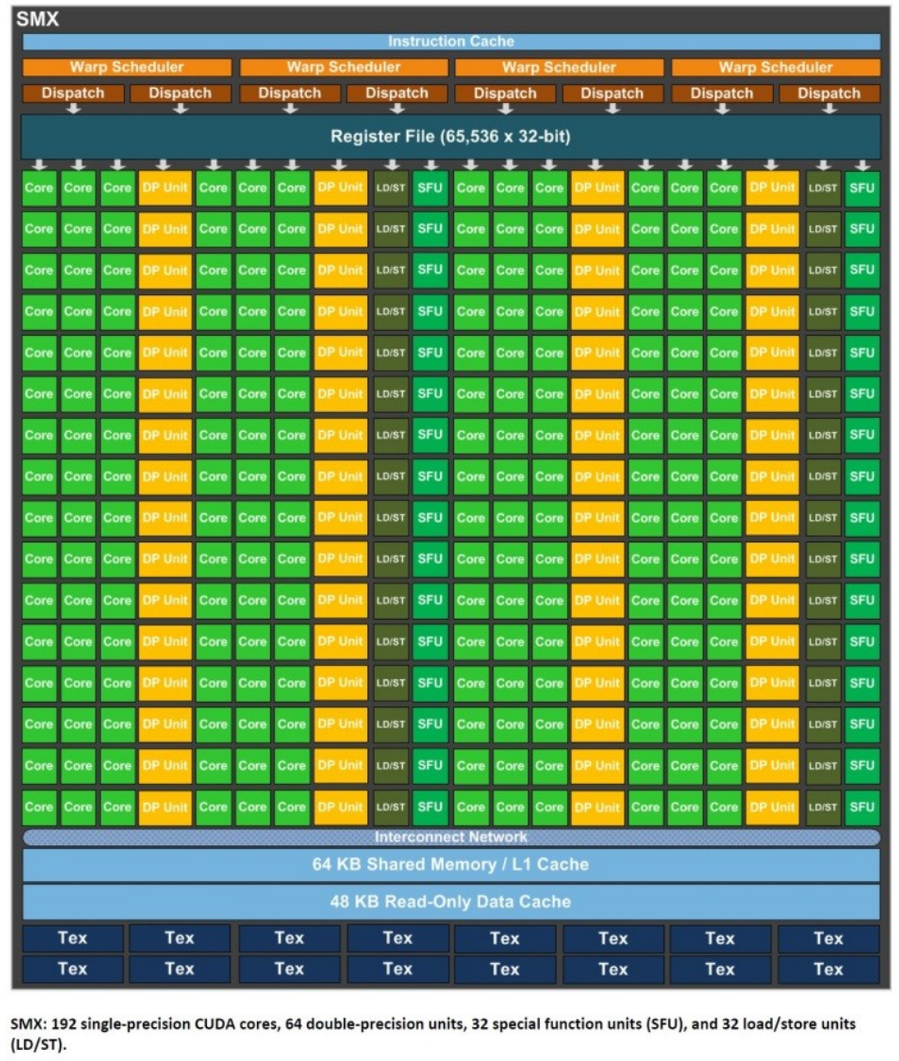
1.5 MB on-chip L2 cache

Only 6GB on chip memory

GPU *vs.* CPU

GPU *vs.* CPU

GPU Example: Kepler



SMX: 192 single-precision CUDA cores, 64 double-precision units, 32 special function units (SFU), and 32 load/store units (LD/ST).

Set of 14~15 SIMD Streaming Multiprocessors (SMX)

Each Multiprocessor has 192 cores, 64k L1 Cache.

Each SMX can handle up to 2000 threads.

GPU Example: Kepler

One SMX

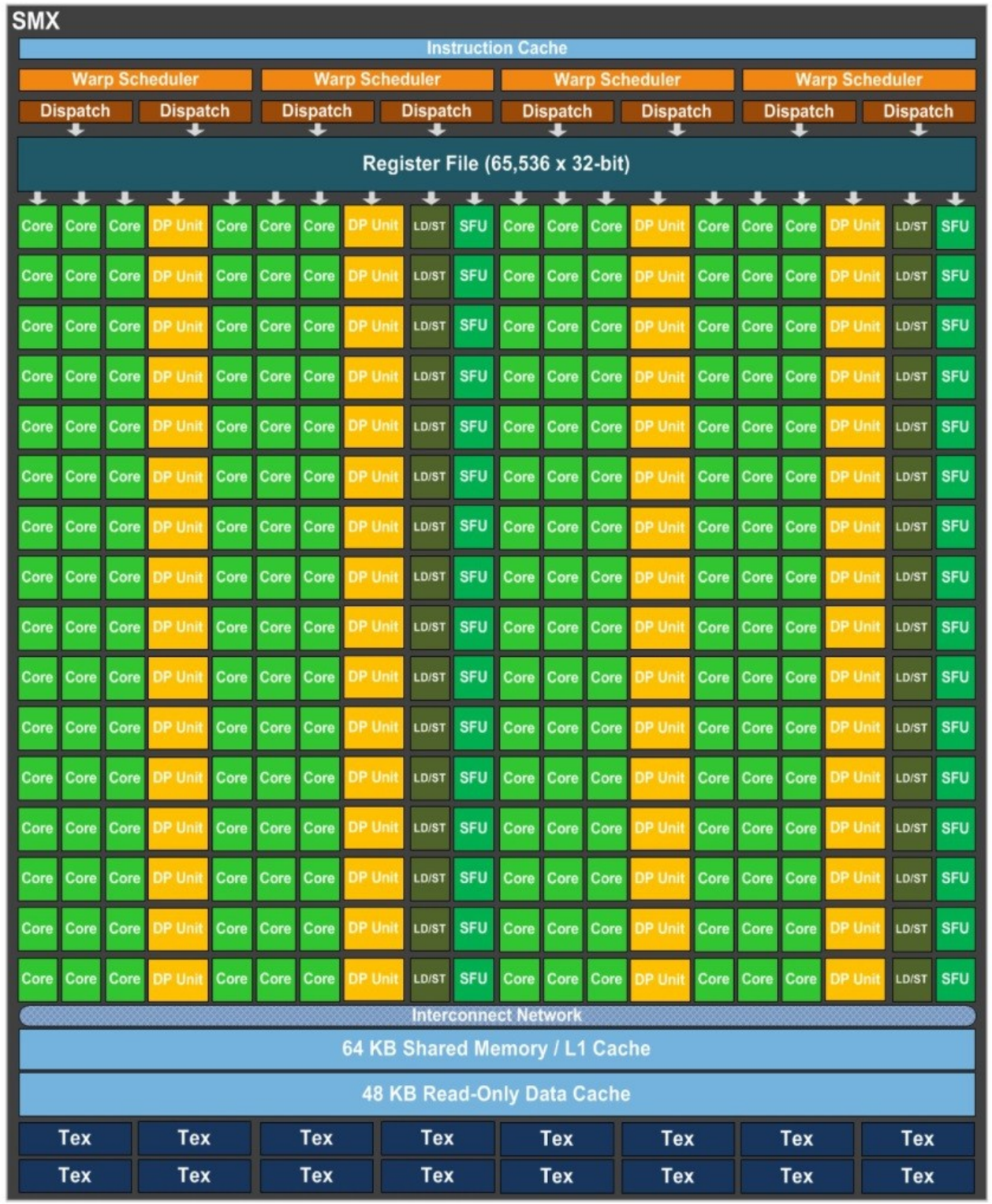
12 x 16=192 cores

32 Special Function Units

32 Load/Store Units

64 Double Precision Units

64k shared memory



SMX: 192 single-precision CUDA cores, 64 double-precision units, 32 special function units (SFU), and 32 load/store units (LD/ST).

GPU

GPU	G80	GT200	Fermi	Kepler
Transistors	681 million	1.4 billion	3.0 billion	7.0 billion
CUDA Cores	128	240	512 @ 1.15 GHz	2688 @ 0.73 GHz
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock	1344 FMA ops/clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock	2688 FMA ops/clock
Special Function Units (SFUs) / SM	2	2	4	32
Warp schedulers (per SM)	1	1	2	2
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB	Configurable 48 KB, 16 KB or 32 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB	Configurable 48 KB, 16 KB or 32 KB
L2 Cache	None	None	768 KB	1.5 MB
ECC Memory Support	No	No	Yes	Yes
Concurrent Kernels	No	No	Up to 16	Up to 32 + Dyn. Parallel
Load/Store Address Width	32-bit	32-bit	64-bit	64-bit

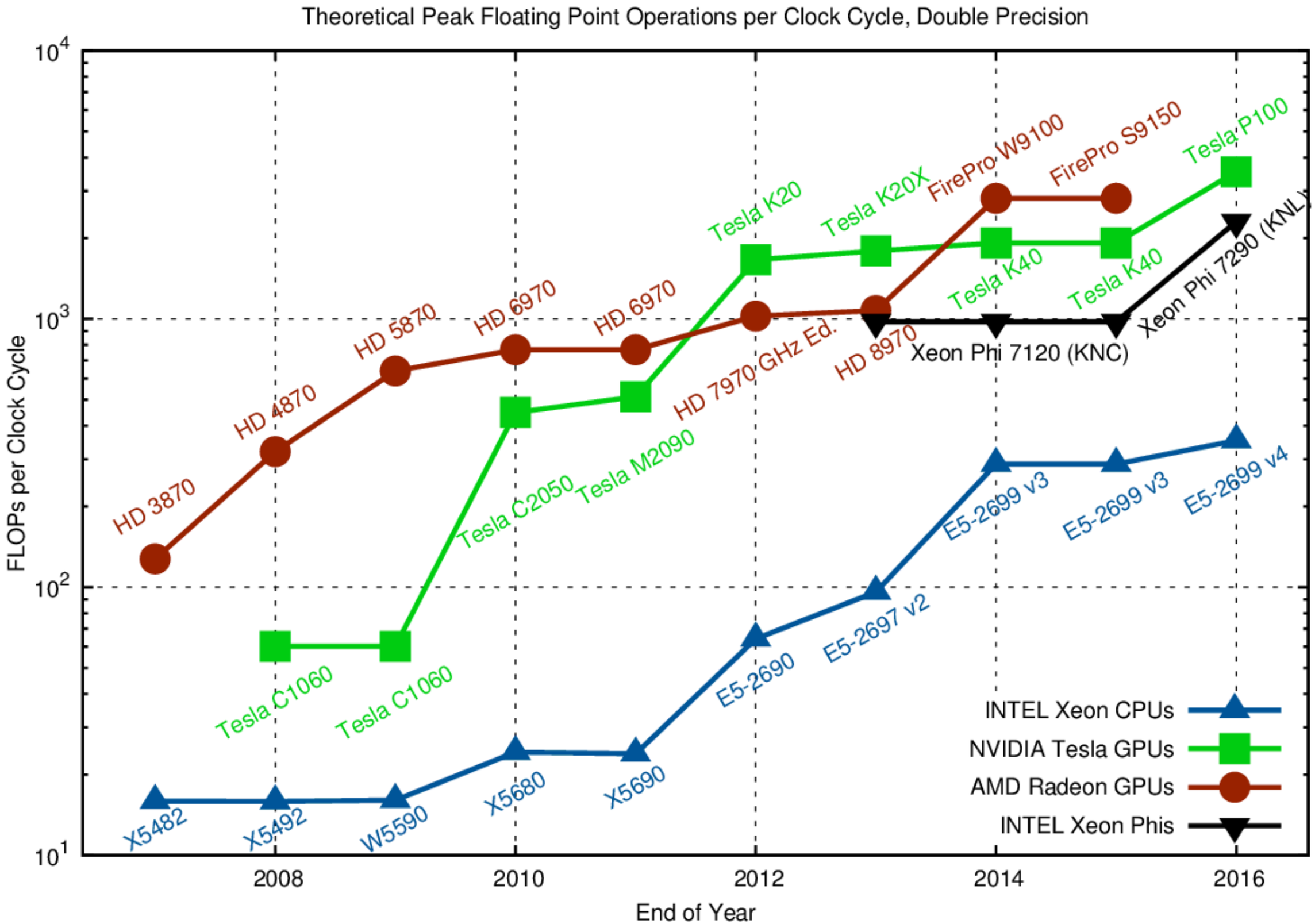
GPU

Because GPUs were designed to apply the same shading function to many pixels simultaneously, GPUs can be used to apply the same **simple** *function to many data points simultaneously*

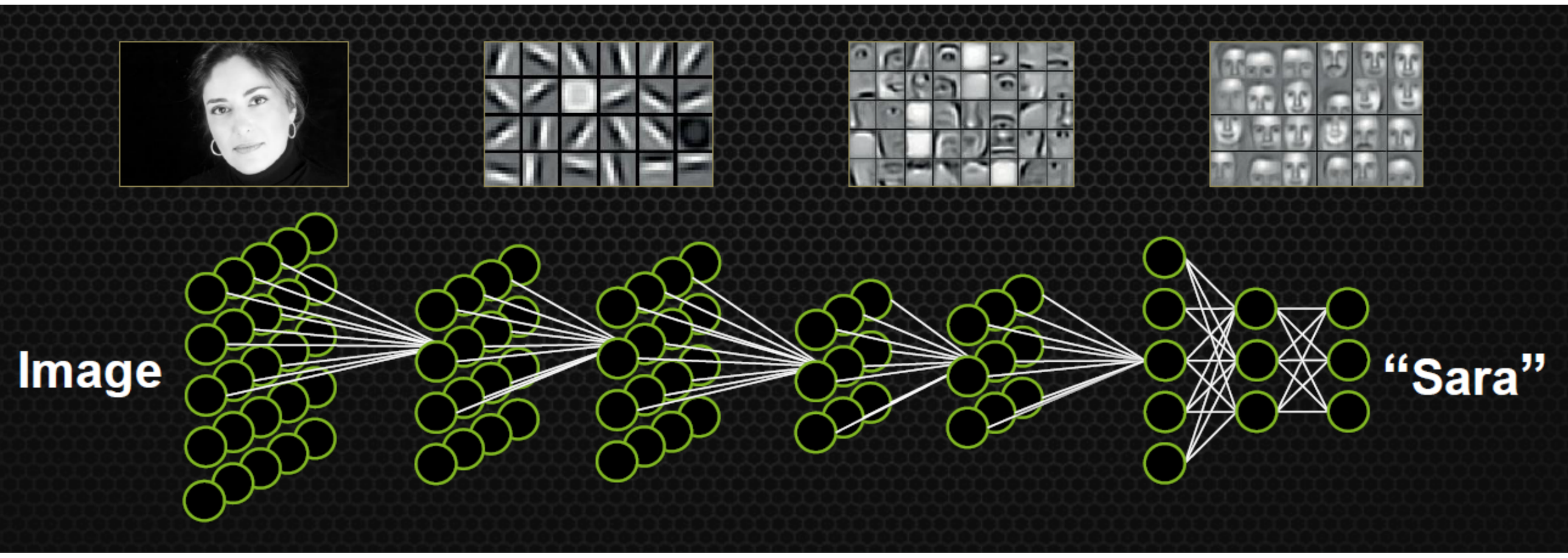
How simple?

Essentially, matrix algebra and special functions on each element (exp, log, sin etc...)

How fast?


Theoretical Peak Floating Point Operations per Clock Cycle, Double Precision
FLOPs per Clock Cycle
10^4
10^3
10^2
10^1
2008
2010
2012
2014
2016
End of Year
HD 3870
HD 4870
HD 5870
HD 6970
HD 6970
HD 7970 GHz Ed.
HD 8970
FirePro W9100
FirePro S9150
Tesla C1060
Tesla C1060
Tesla C2050
Tesla M2090
Tesla K20
Tesla K20X
Tesla K40
Tesla K40
Tesla P100
Xeon Phi 7120 (KNC)
Xeon Phi 7290 (KNL)
X5482
X5492
W5590
X5680
X5690
E5-2690
E5-2697 v2
E5-2699 v3
E5-2699 v3
E5-2699 v4
INTEL Xeon CPUs
NVIDIA Tesla GPUs
AMD Radeon GPUs
INTEL Xeon Phis

Crucial for Deep Learning



Why?

Multilayer Neural Networks only use element wise operations (hinge, softmax, tanh, sigmoid) and matrix products, exactly those operations that GPU are good for.

https://ruder.io/optimizing-gradient-descent/

Optimizers

https://ruder.io/optimizing-gradient-descent/

Batch Gradient Descent

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) + \lambda R(\theta)$$

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta).$$

Stochastic Gradient Descent

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) + \lambda R(\theta)$$

Sample one point i

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}).$$

Mini-batch Gradient Descent

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) + \lambda R(\theta)$$

Sample n points i

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}).$$

Momentum-based

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) + \lambda R(\theta)$$

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

$$\theta = \theta - v_t$$

Momentum parameter γ approx. set to 0.9

Nesterov Momentum

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) + \lambda R(\theta)$$

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$
$$\theta = \theta - v_t$$

Adagrad

$$g_{t,i} = \nabla_\theta J(\theta_{t,i}).$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t.$$

$$G_t = \sum_{i=0}^{t} (g_{t,i})^2$$

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}.$$

SGD

RMS-Prop

$$g_{t,i} = \nabla_\theta J(\theta_{t,i}).$$

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

$$G_t = \sum_{i=0}^{t} (g_{t,i})^2$$

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}.$$

SGD

Adam

$$g_{t,i} = \nabla_\theta J(\theta_{t,i}).$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

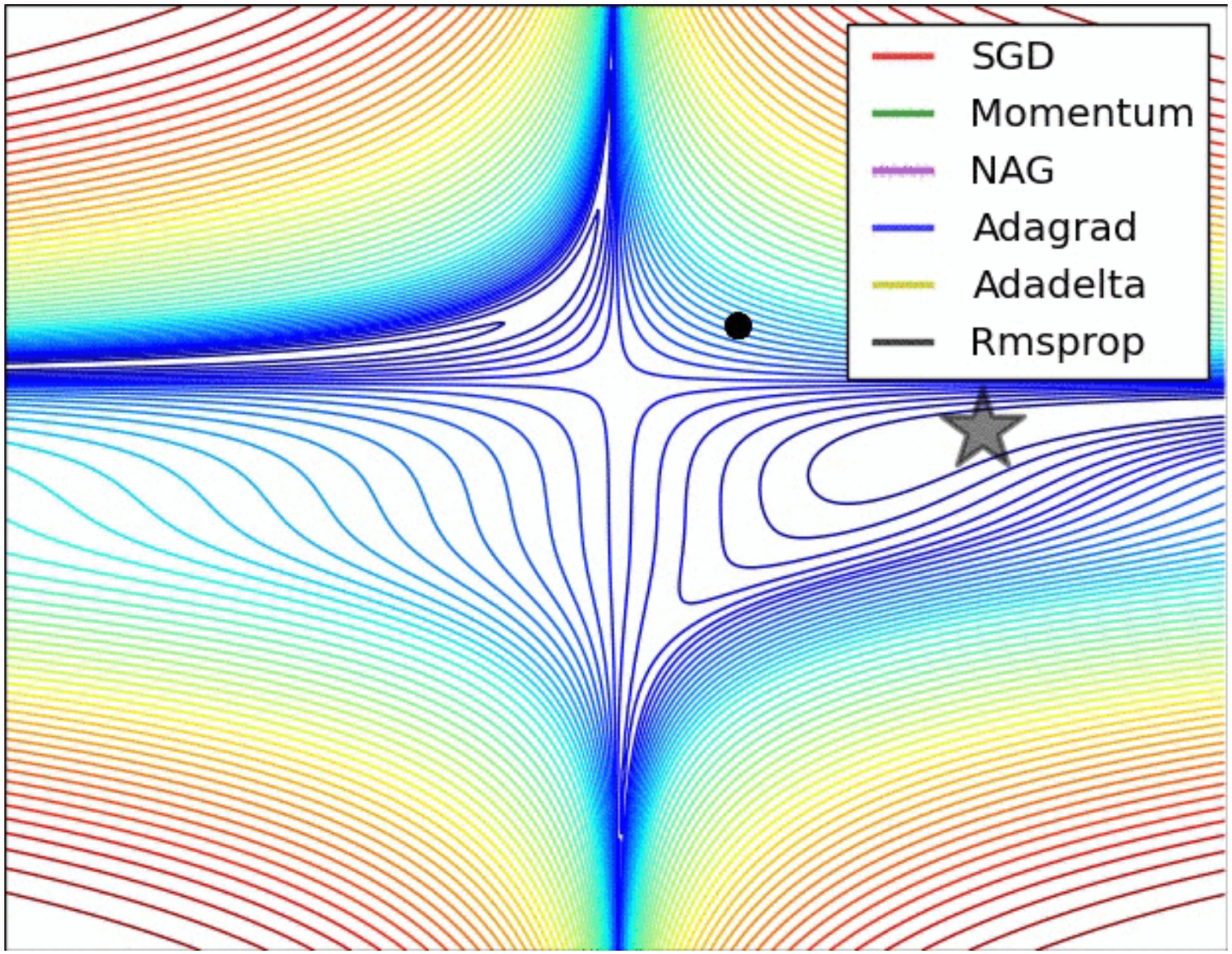
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

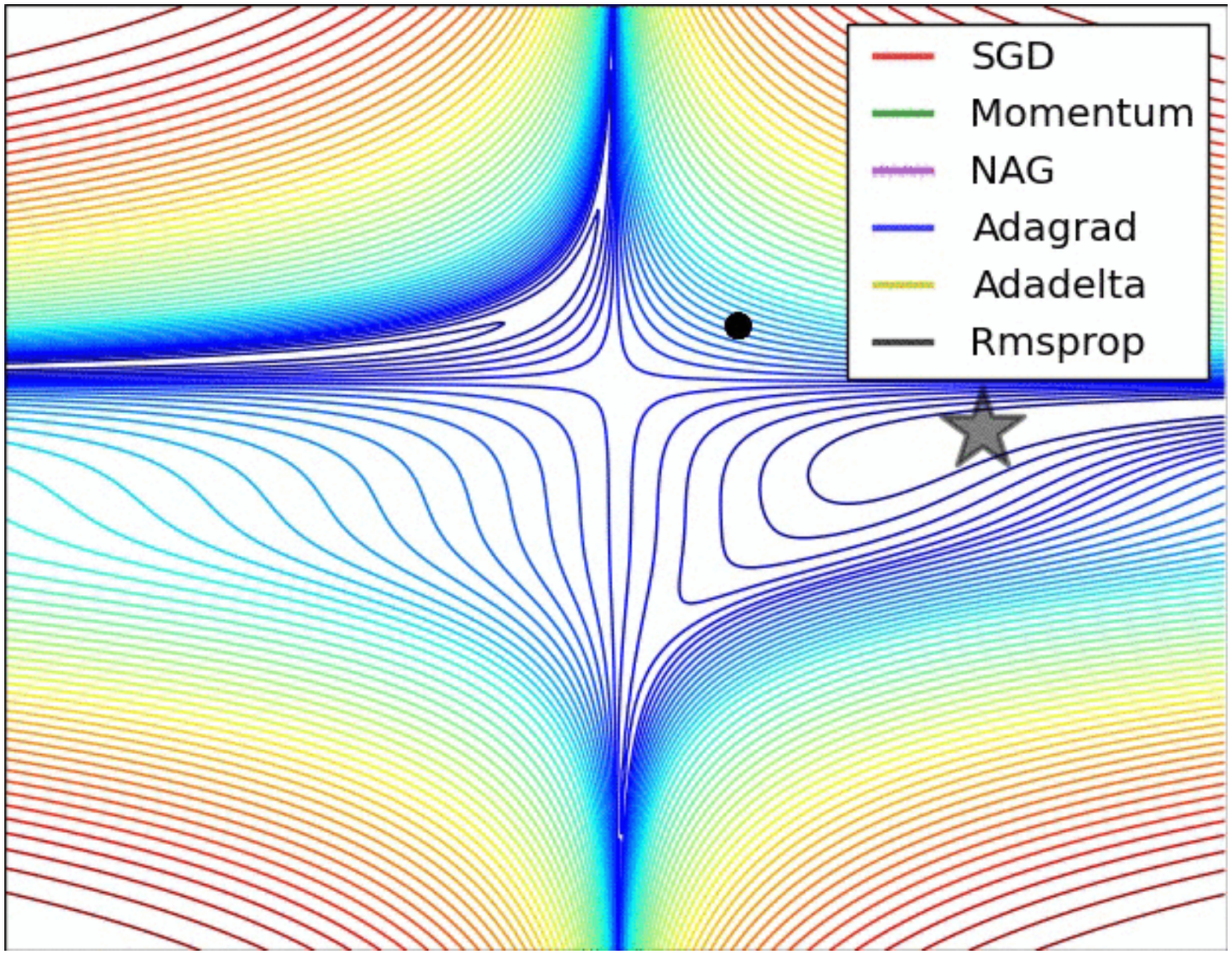
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.$$

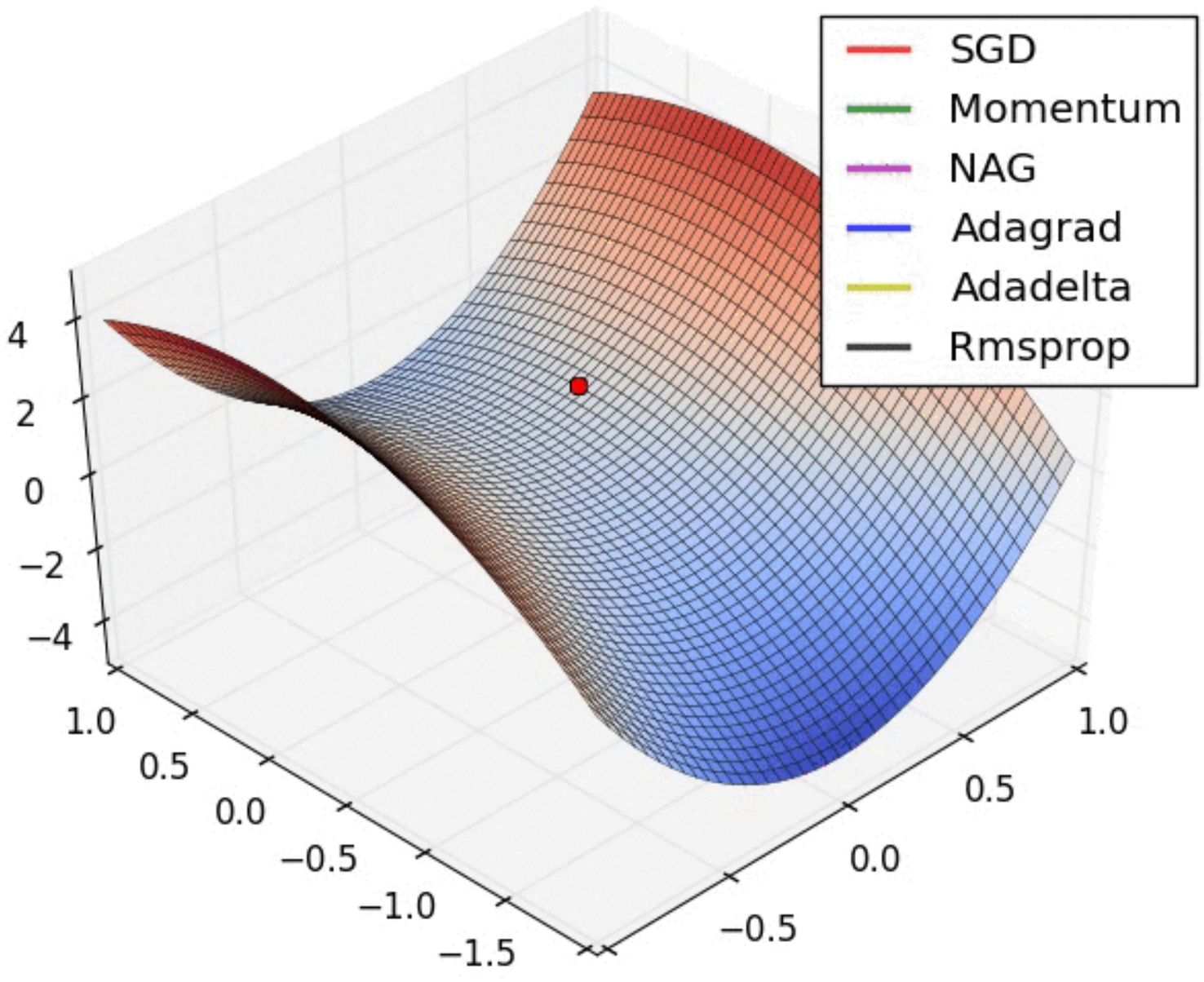
In action on non-convex objective



In action on non-convex objective

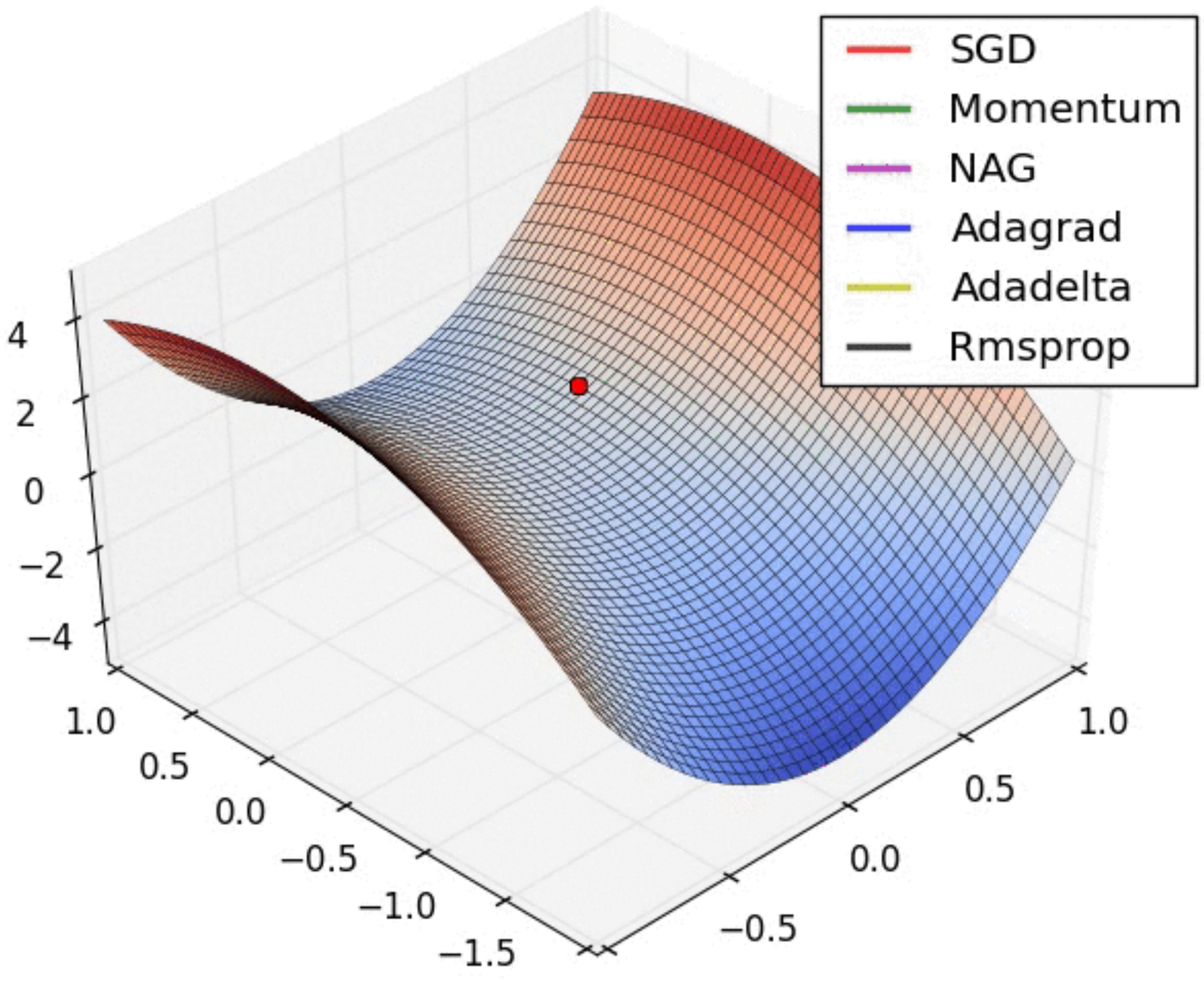


On saddle point problem



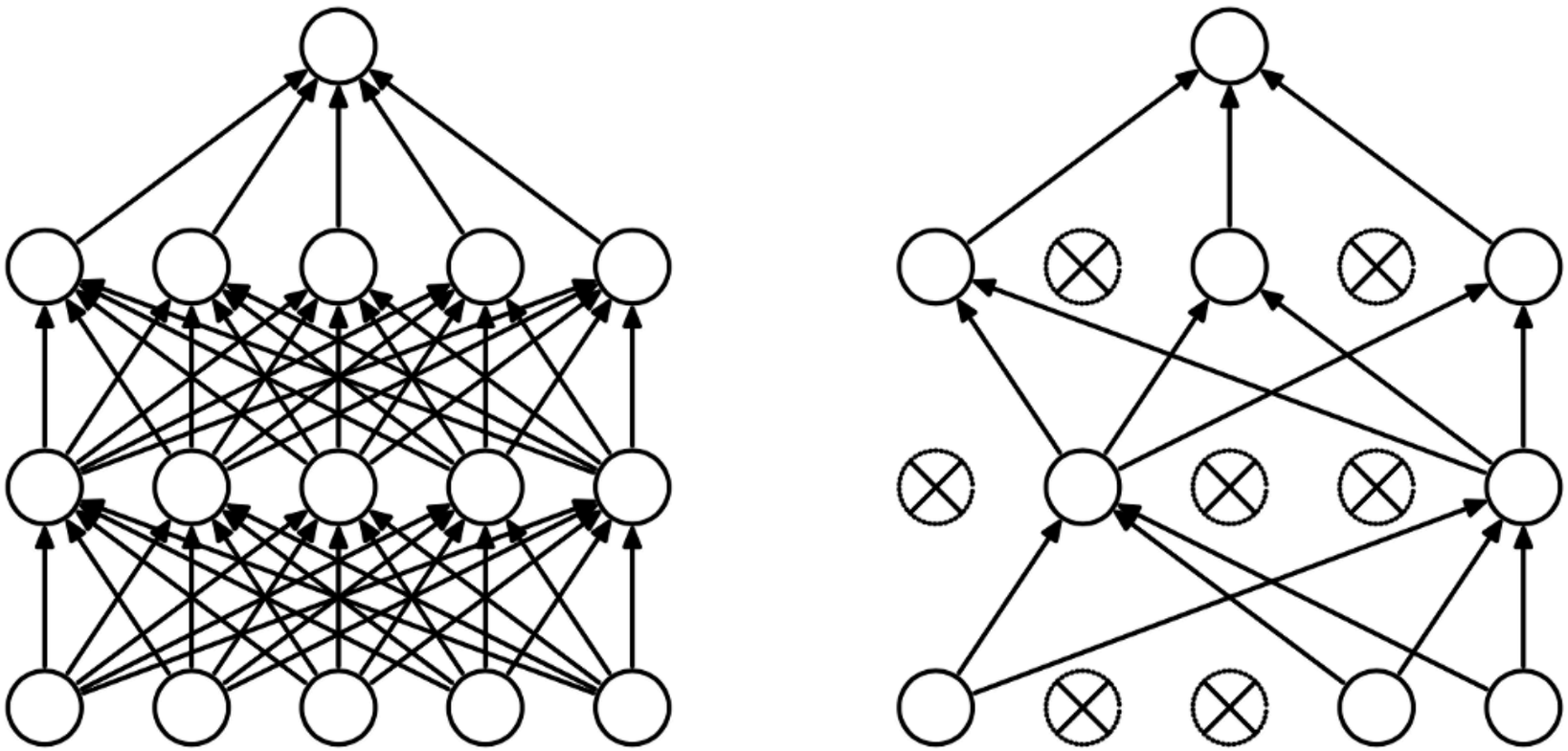
https://ruder.io/optimizing-gradient-descent/

On saddle point problem



https://ruder.io/optimizing-gradient-descent/

Robustness: Dropout



(a) Standard Neural Net

(b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Robustness: BatchNorm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

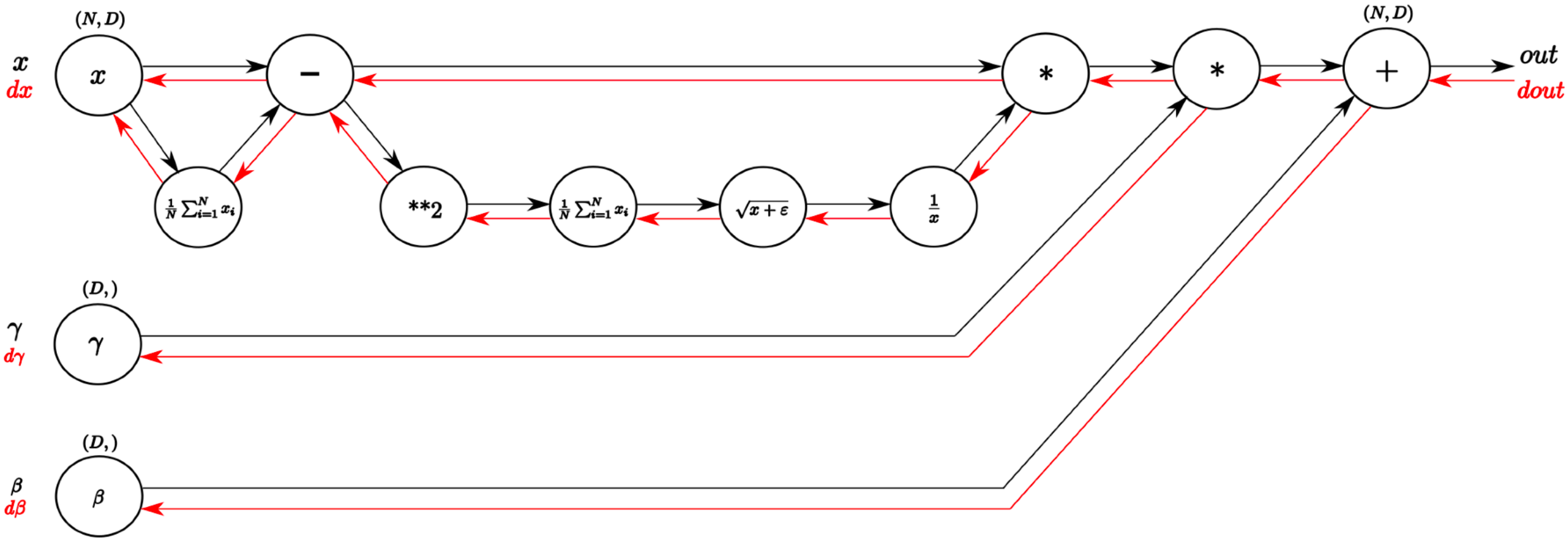
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

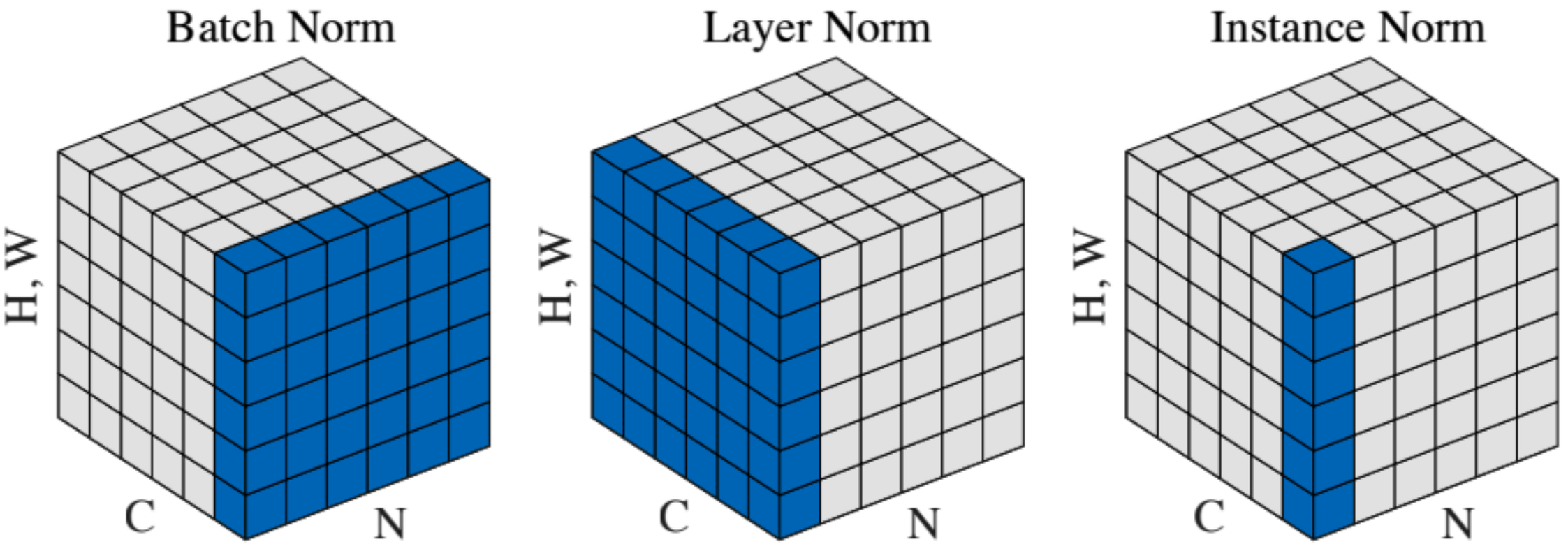
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Robustness: BatchNorm



https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html

Robustness: X-Norm



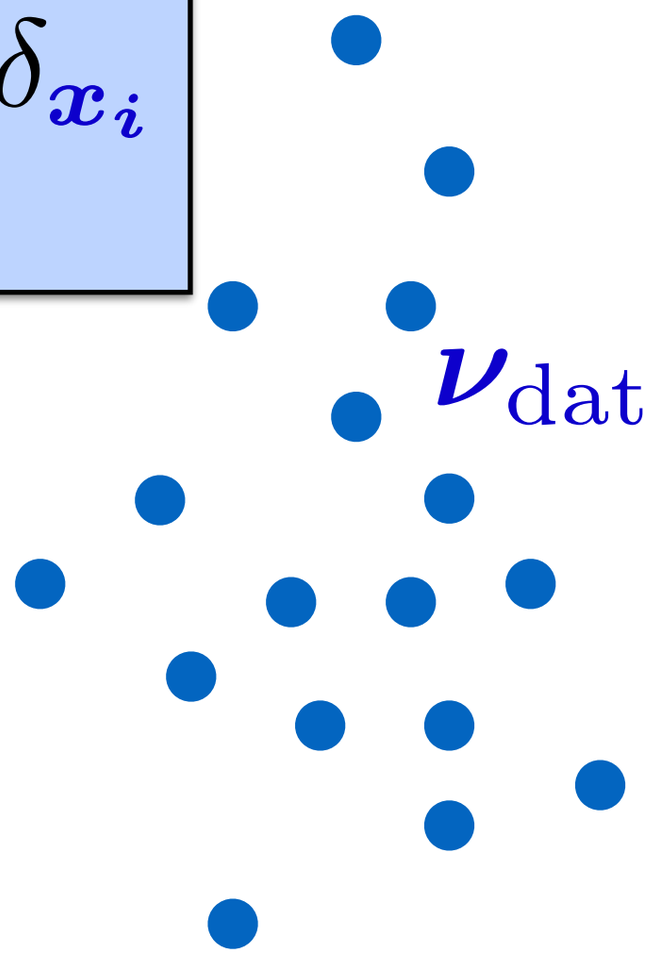
Wu, Y. and He, K., 2018. Group normalization. arXiv preprint arXiv: 1803.08494.

GANs

Generative Models

We collect data

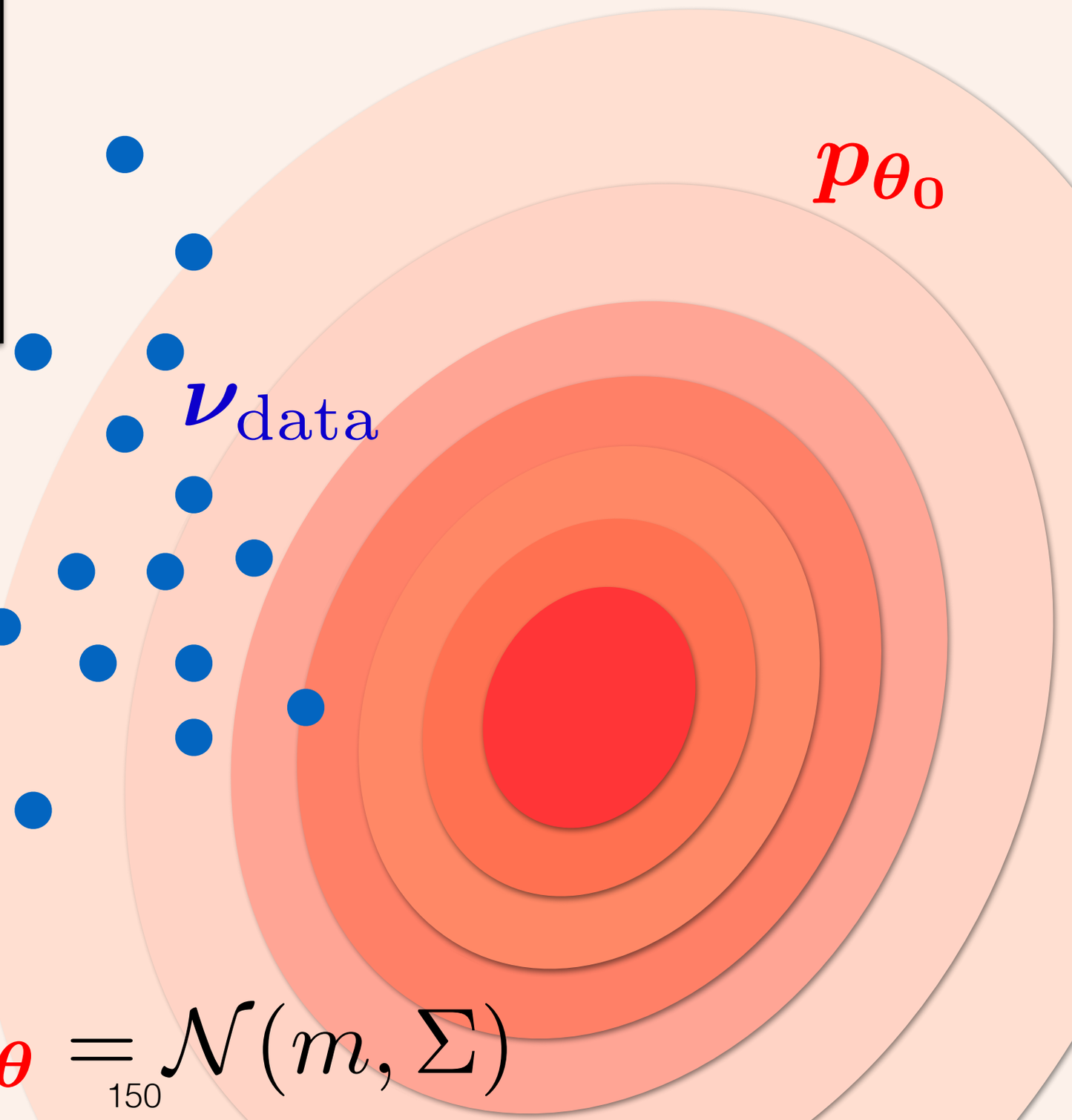
$$\boldsymbol{\nu}_{\text{data}} = \frac{1}{N}\sum_{i=1}^{N}\delta_{\boldsymbol{x_i}}$$



Generative Models

We collect data

$$\boldsymbol{\nu}_{\text{data}} = \frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{x}_i}$$

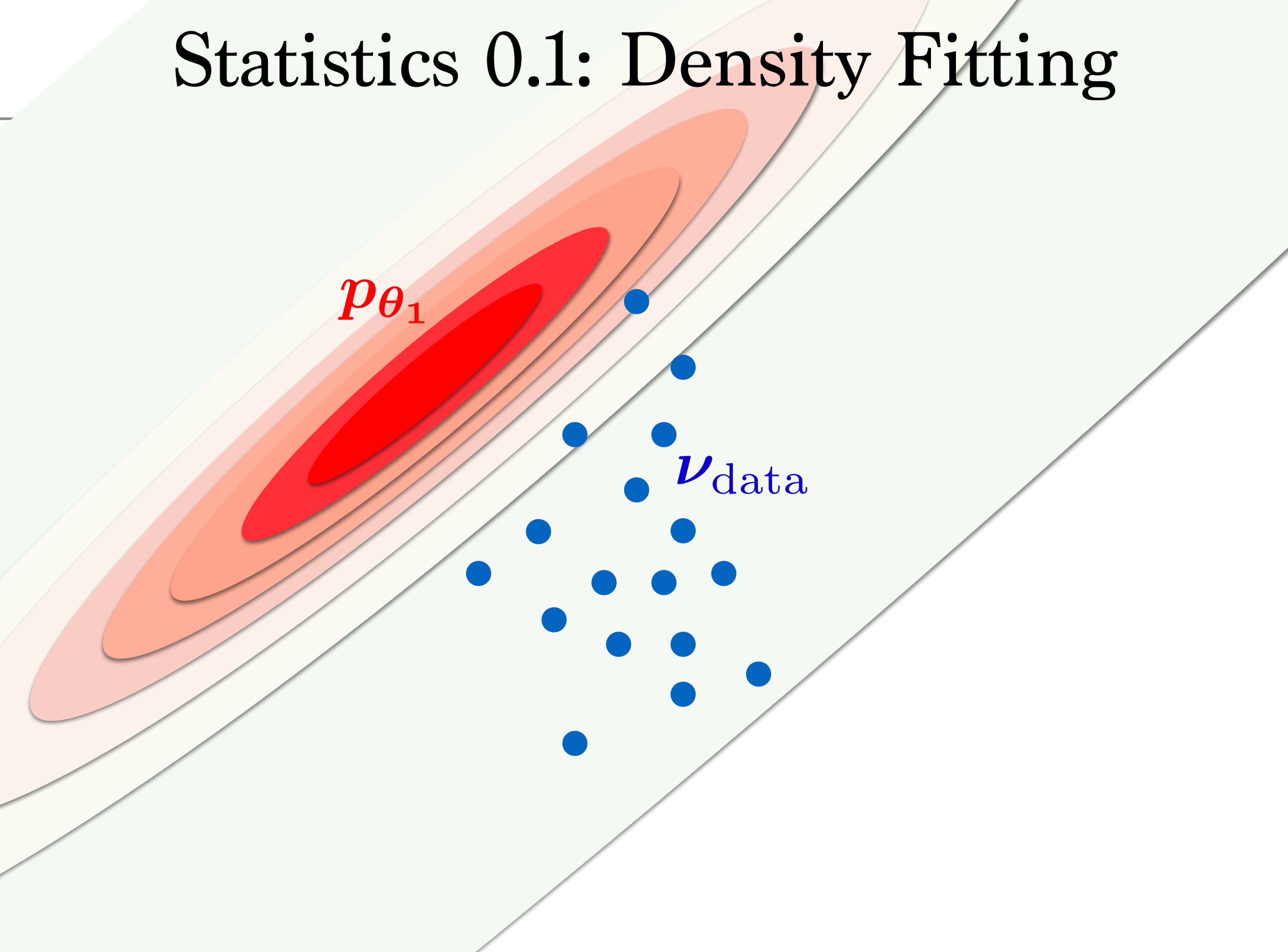


We fit a parametric family of densities

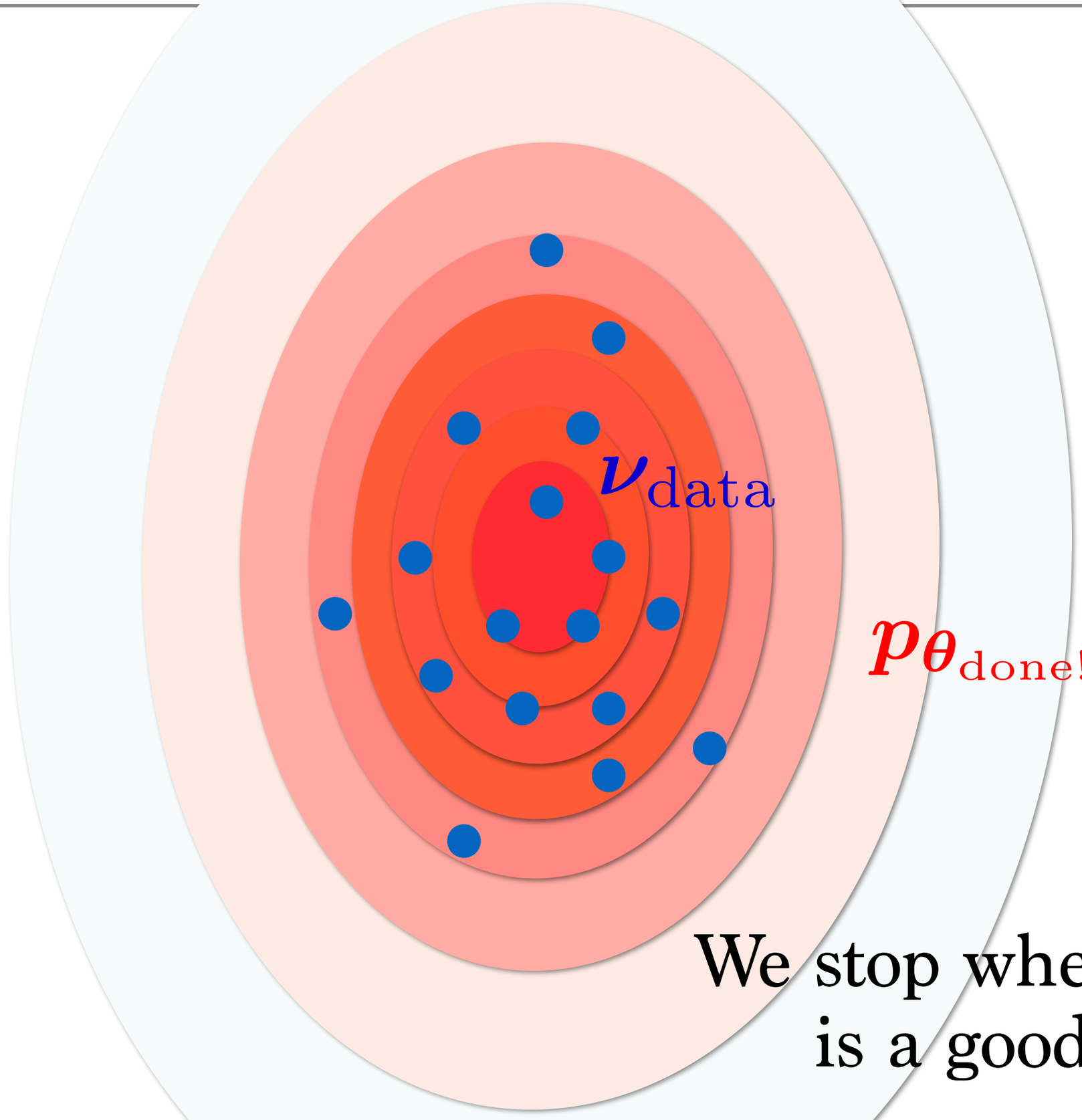
$$\{p_\theta, \theta \in \Theta\}$$

$$e.g.\ \boldsymbol{\theta} = (m, \Sigma); \boldsymbol{p_\theta} = \mathcal{N}(m, \Sigma)$$

Statistics 0.1: Density Fitting

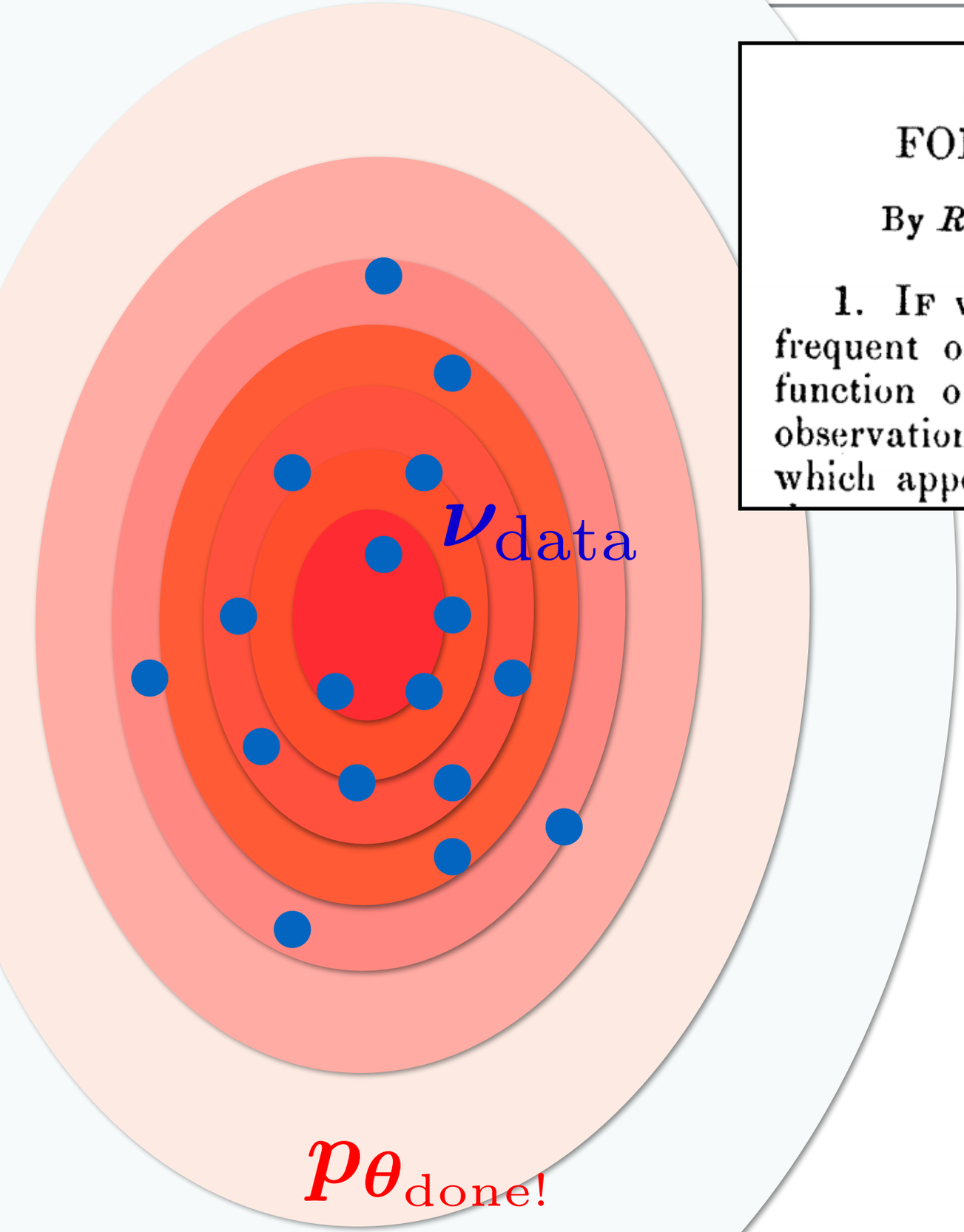


Statistics 0.1: Density Fitting



We stop when there is a good fit.

Maximum Likelihood Estimation



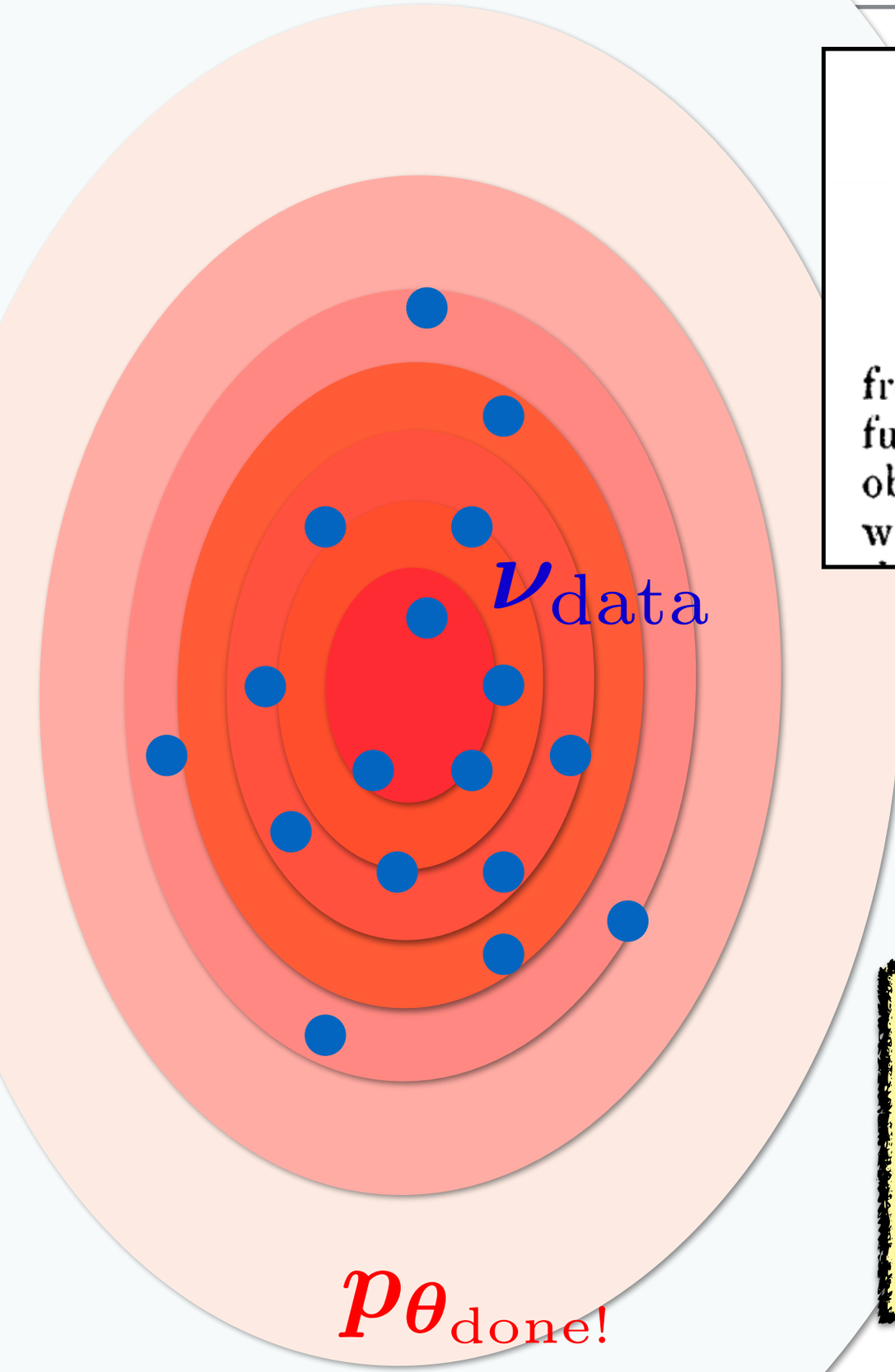
ON AN ABSOLUTE CRITERION
FOR FITTING FREQUENCY CURVES.

By *R. A. Fisher*, Gonville and Caius College, Cambridge.

1. If we set ourselves the problem, in its
frequent occurrence, of finding the arbitrary
function of known form, which best suit a
observations, we are met at the outset by an
which appears to invalidate any results we ma

$$\max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(x_i)$$

Maximum Likelihood Estimation



ON AN ABSOLUTE CRITERION
FOR FITTING FREQUENCY CURVES.

By *R. A. Fisher*, Gonville and Caius College, Cambridge.

1. If we set ourselves the problem, in its
frequent occurrence, of finding the arbitrary
function of known form, which best suit a
observations, we are met at the outset by an
which appears to invalidate any results we ma

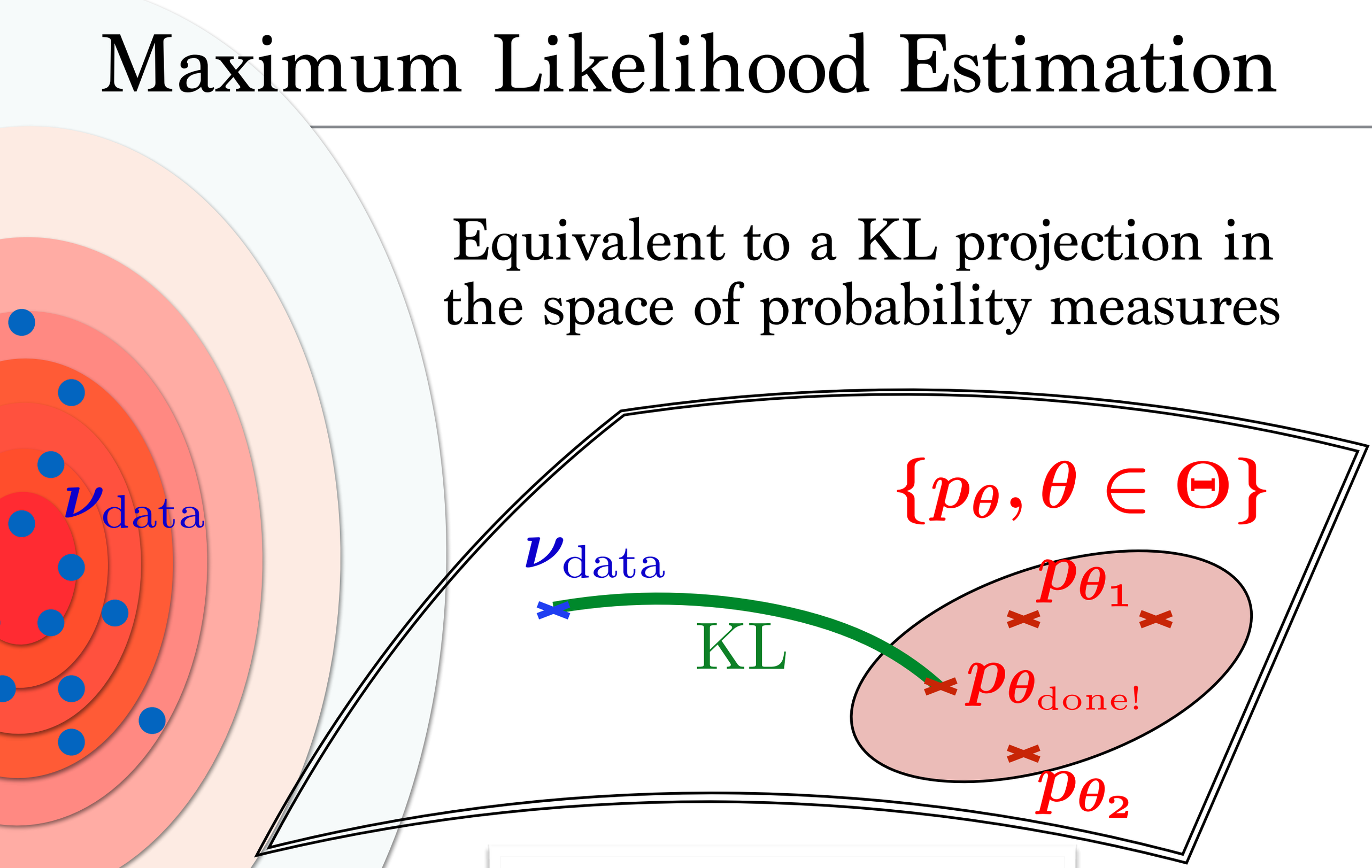
$$\max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(x_i)$$

$$\log 0 = -\infty$$

$p_\theta(x_i)$ **must** be > 0

Maximum Likelihood Estimation

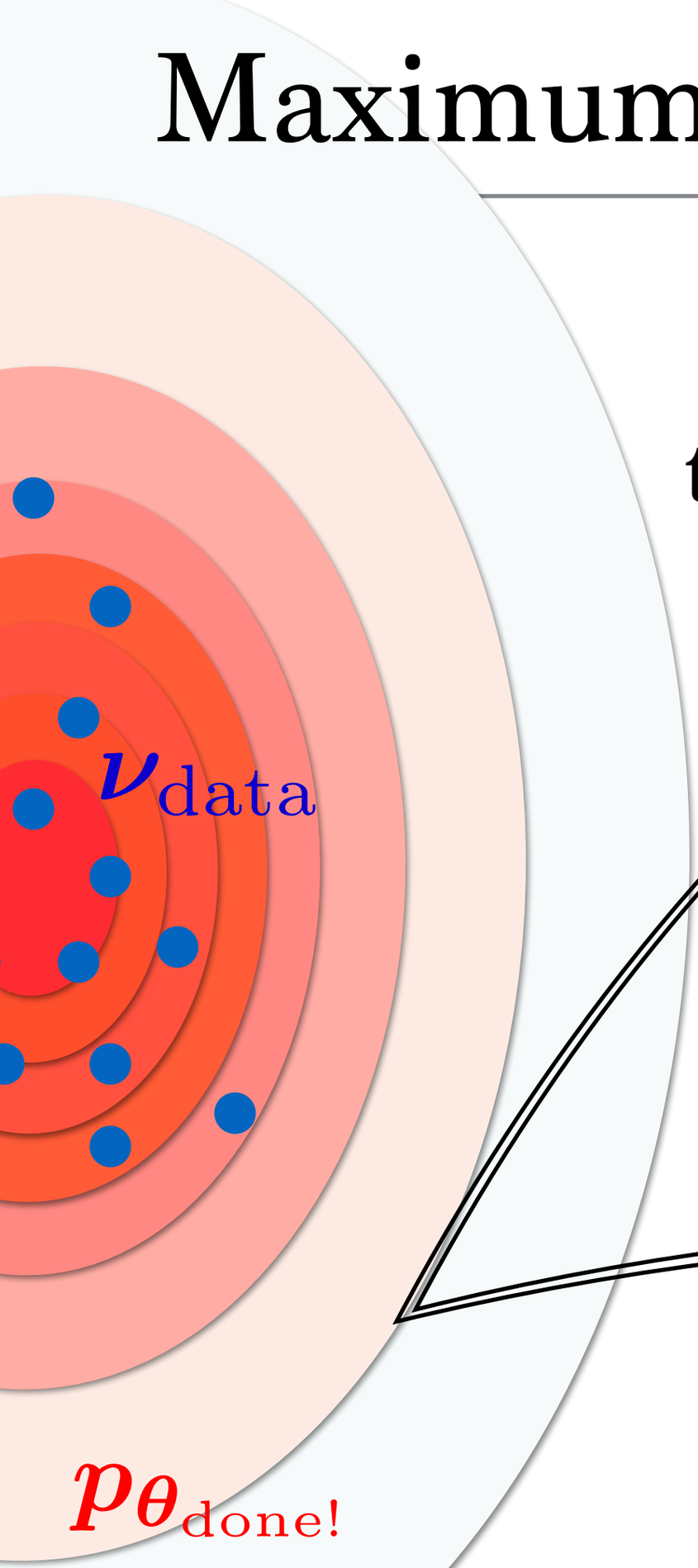
Equivalent to a KL projection in the space of probability measures

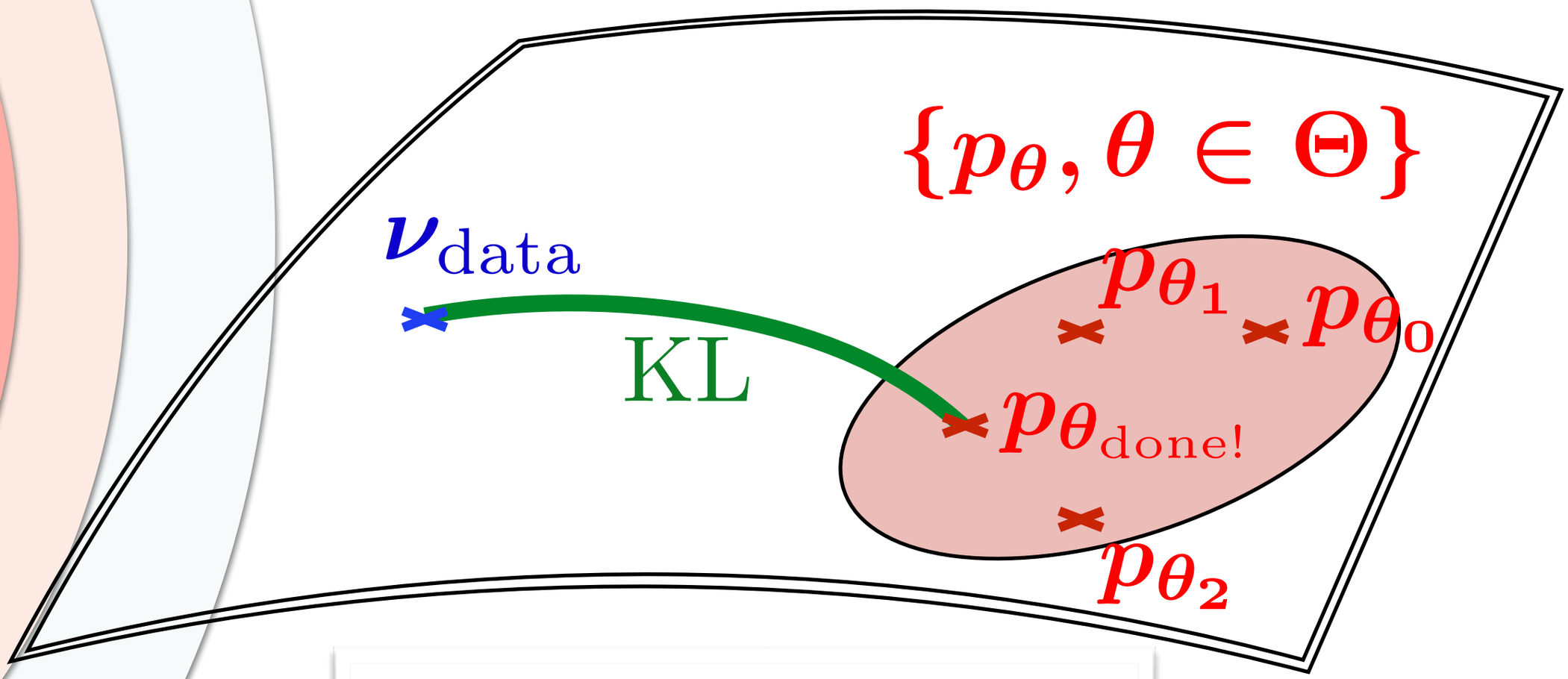


$$\min_{\theta \in \Theta} \mathrm{KL}(\nu_{\text{data}} \| p_\theta)$$

Maximum Likelihood Estimation

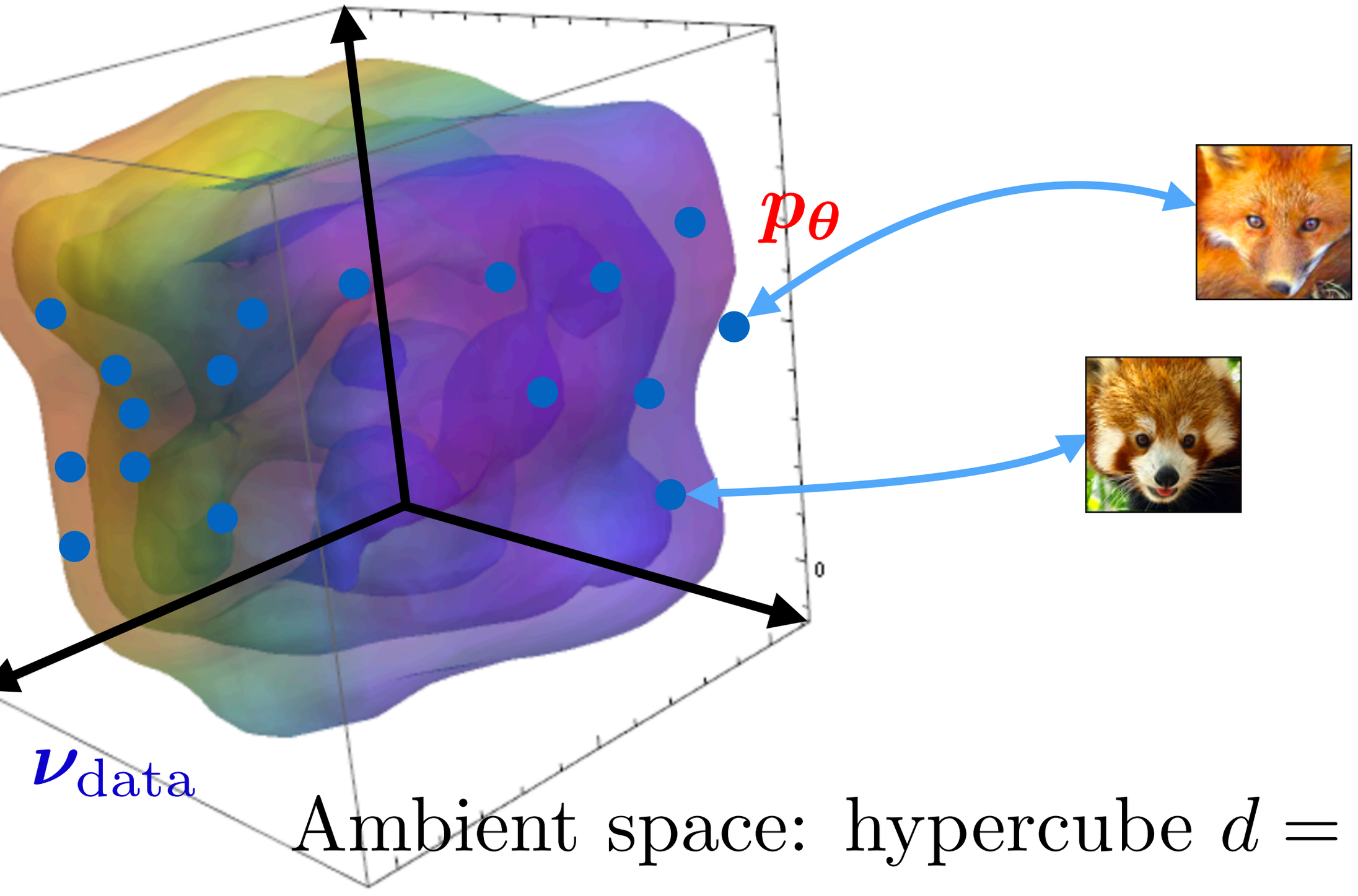
Equivalent to a KL projection in the space of probability measures





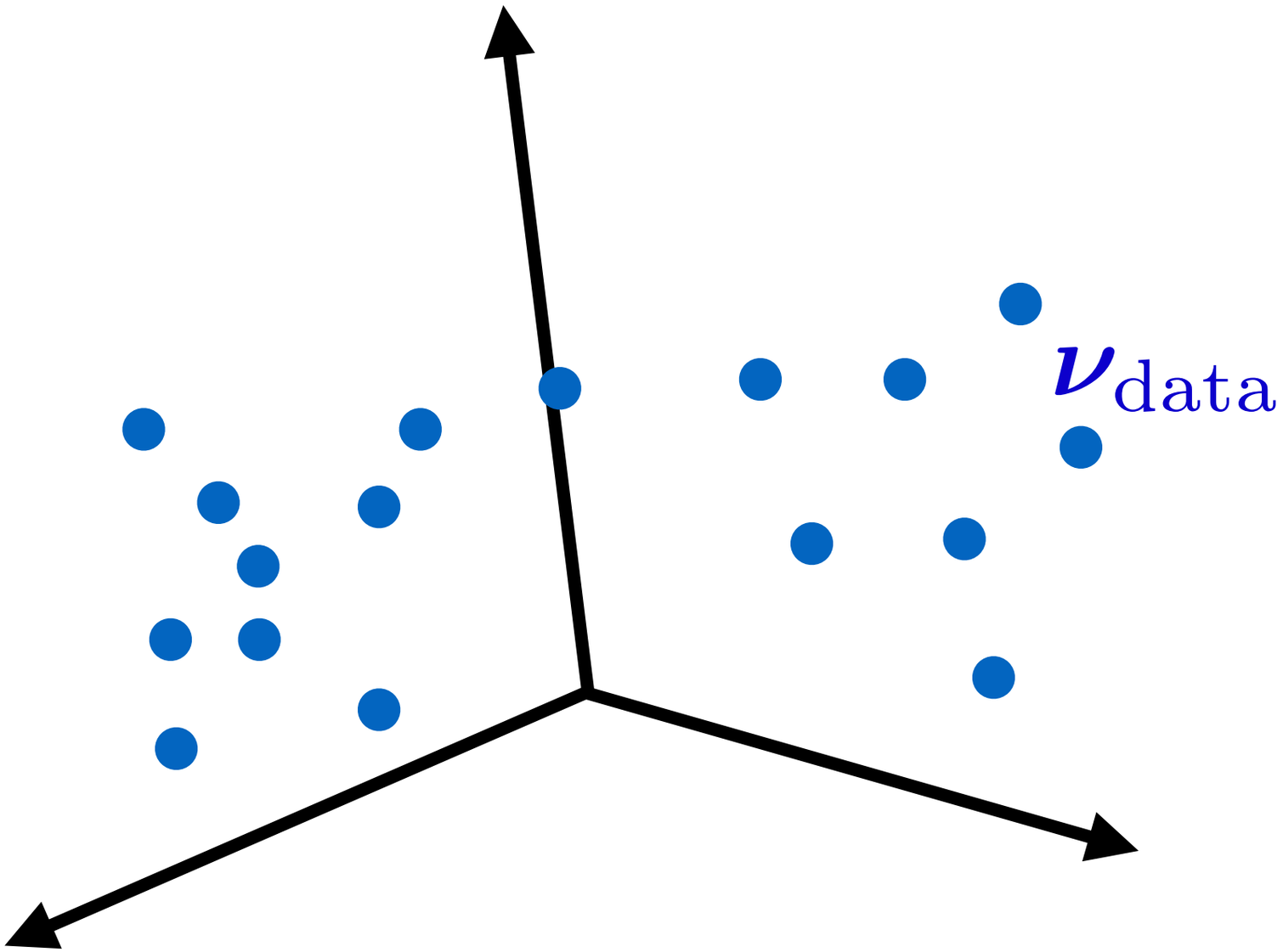
$$\min_{\theta \in \Theta} \mathrm{KL}(\nu_{\text{data}} \| p_\theta)$$

In higher dimensional spaces...

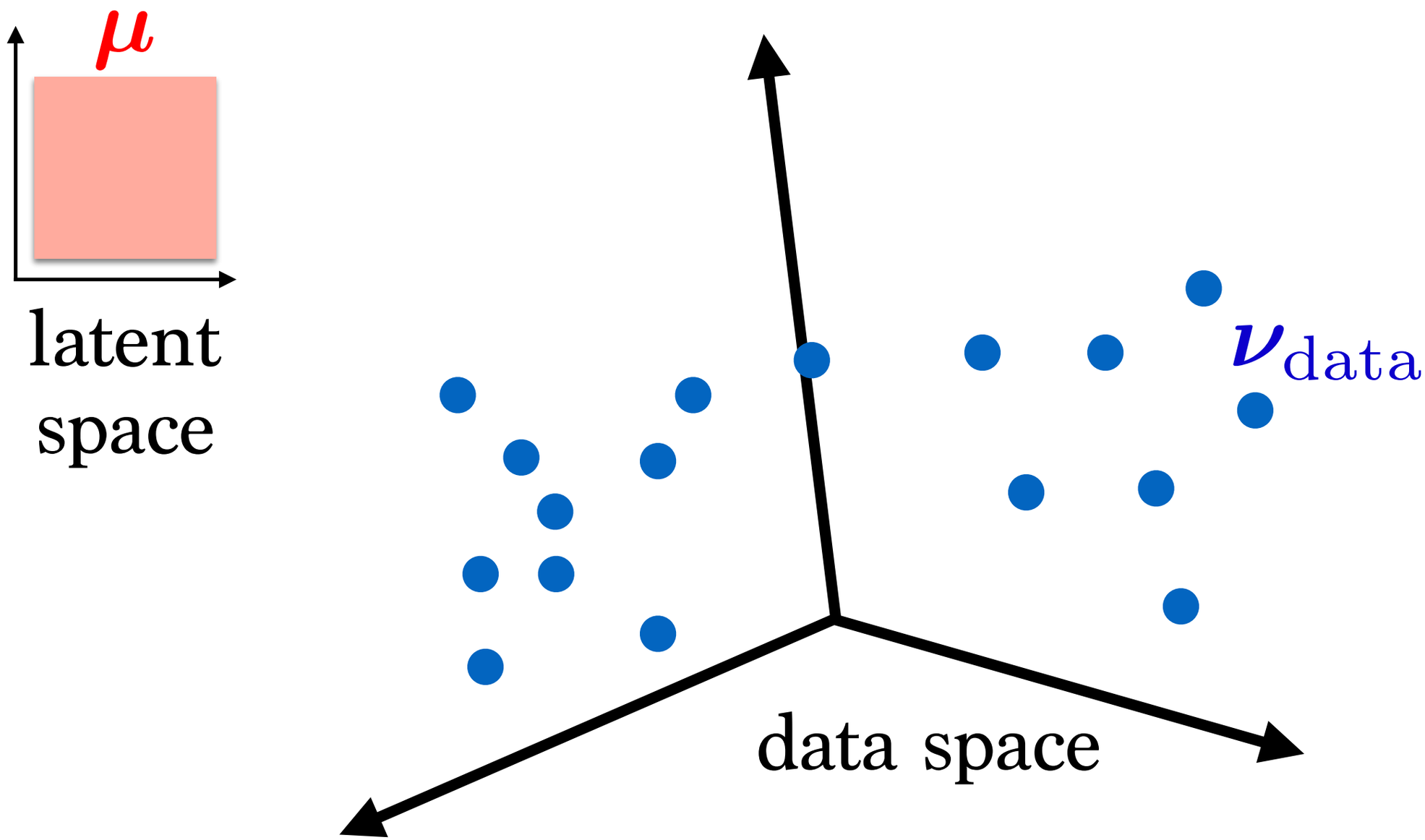


Ambient space: hypercube $d = 30.000$

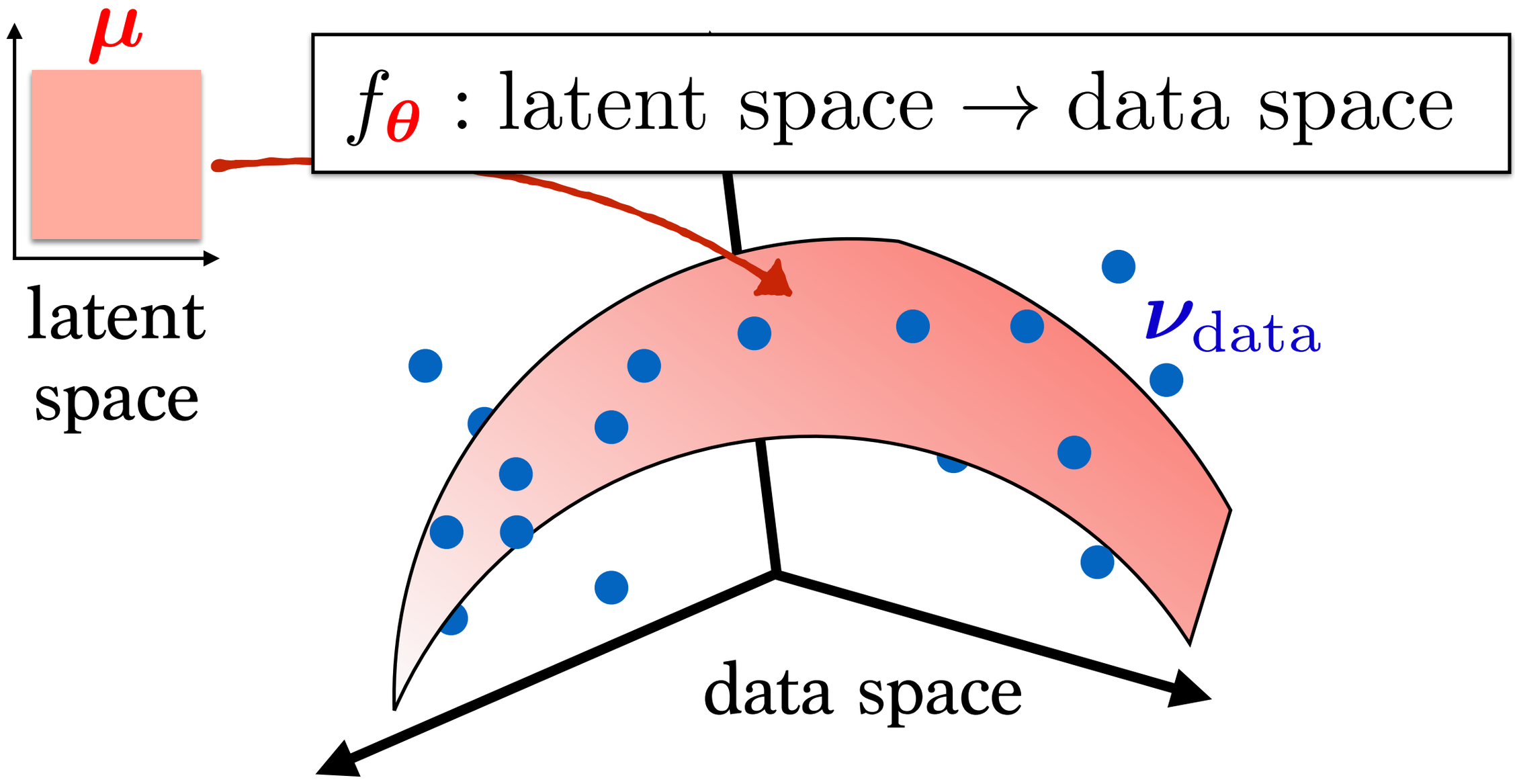
Generative Models



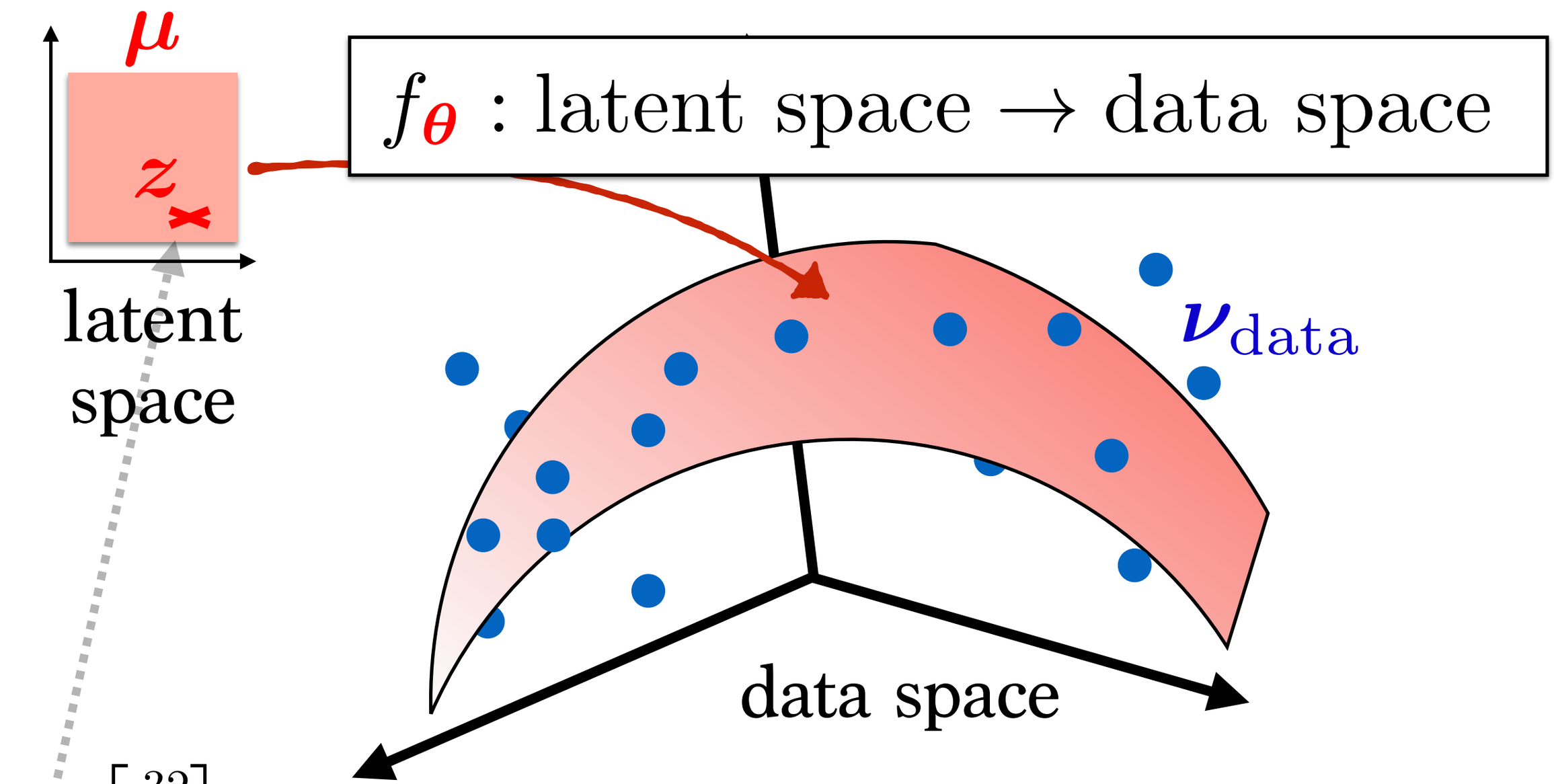
Generative Models



Generative Models

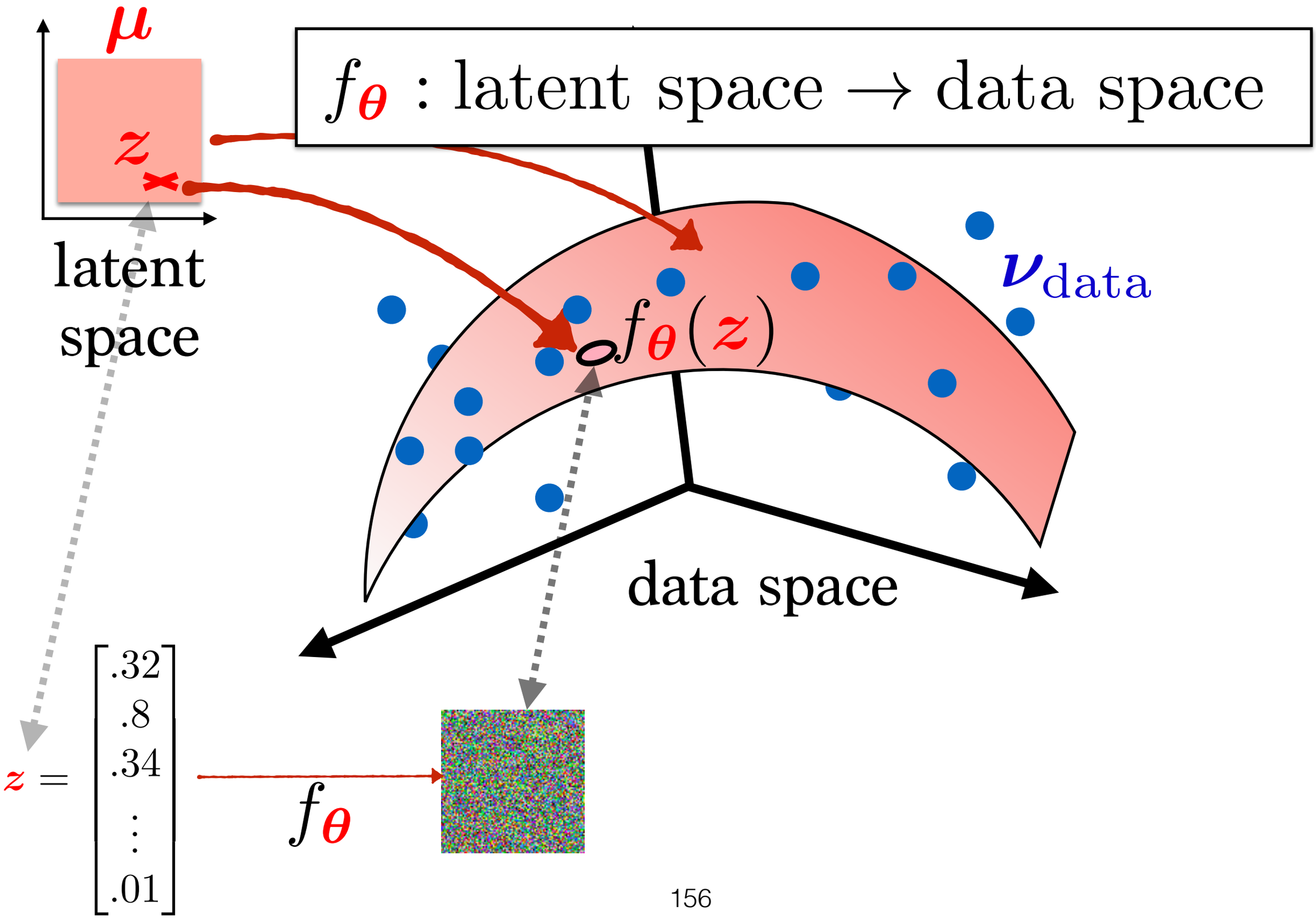


Generative Models

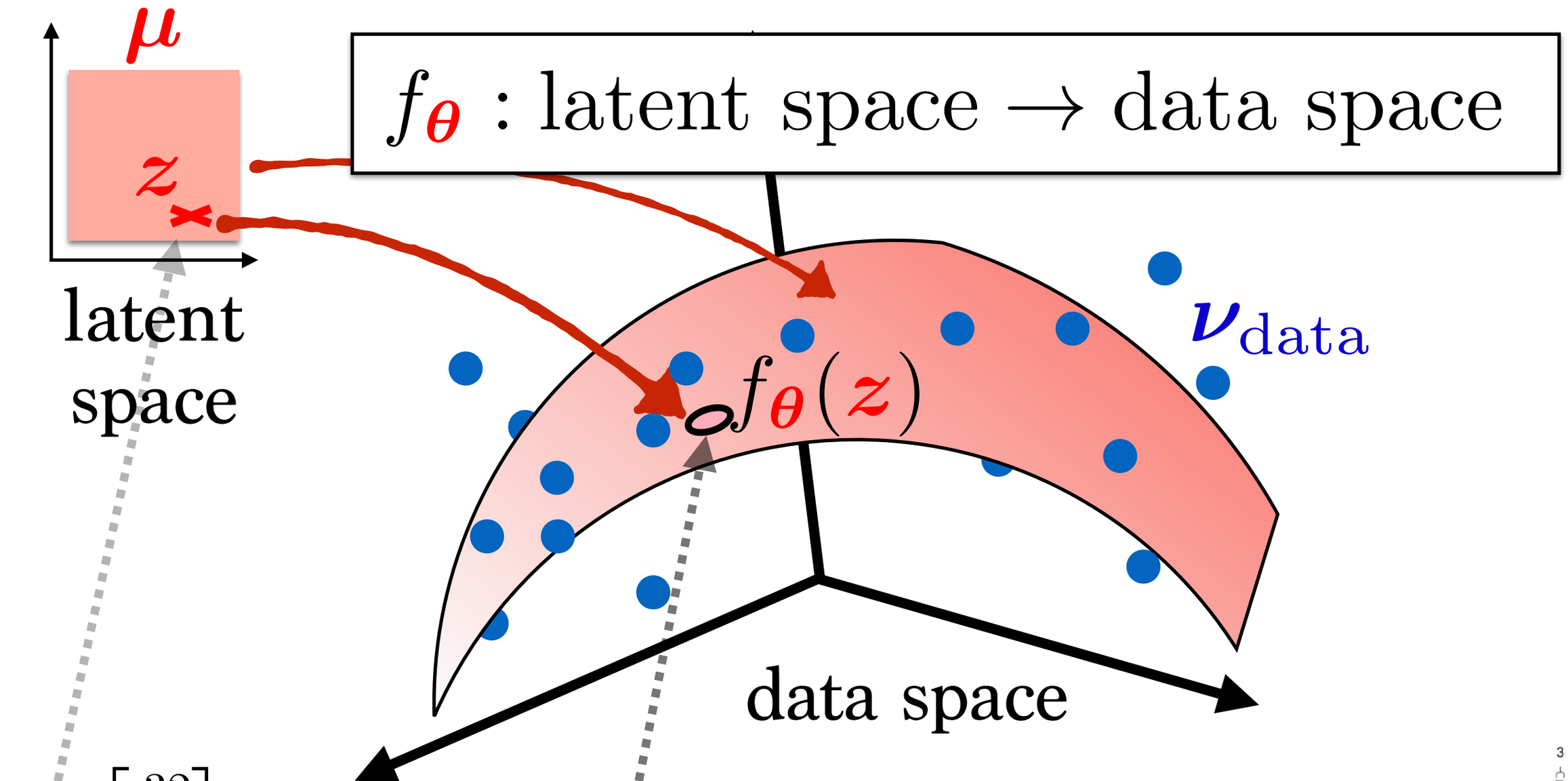


$$z = \begin{bmatrix} .32 \\ .8 \\ .34 \\ \vdots \\ .01 \end{bmatrix}$$

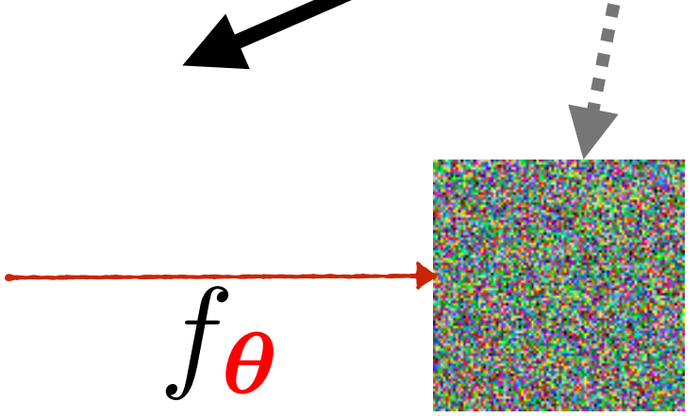
Generative Models

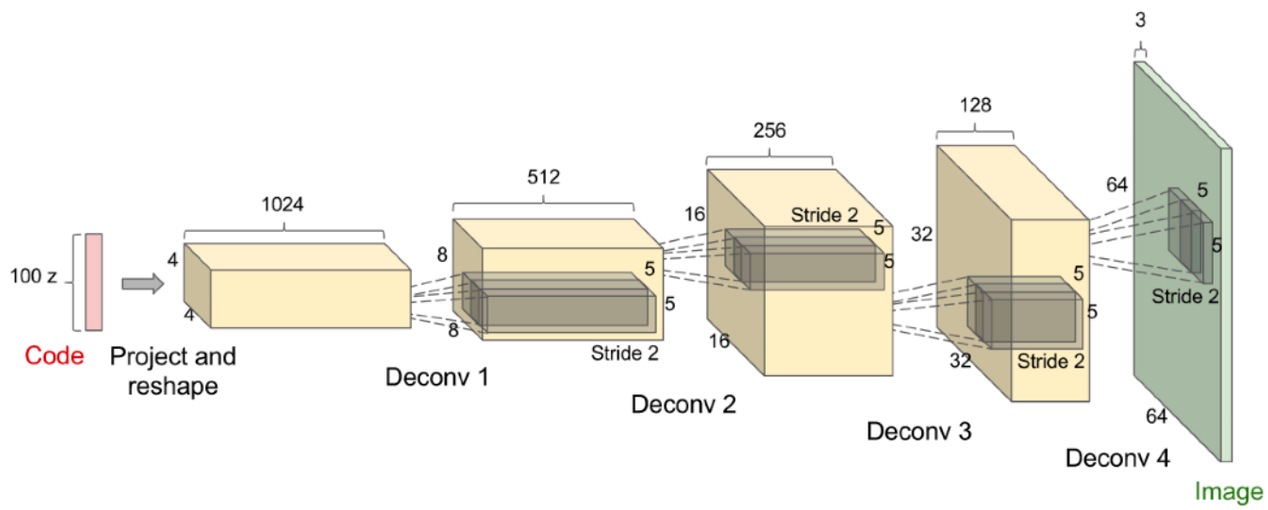


Generative Models

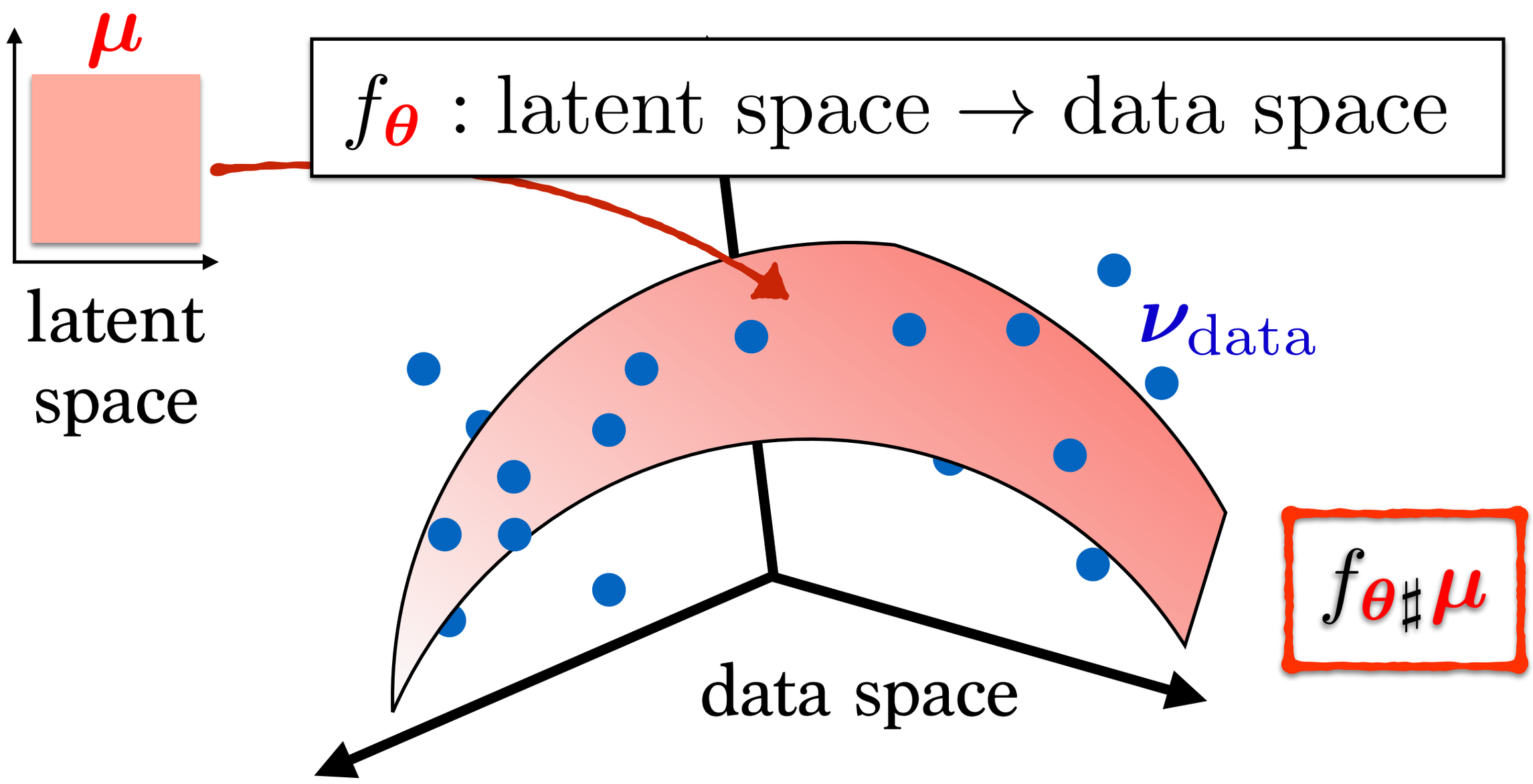


$$z = \begin{bmatrix} .32 \\ .8 \\ .34 \\ \vdots \\ .01 \end{bmatrix}$$

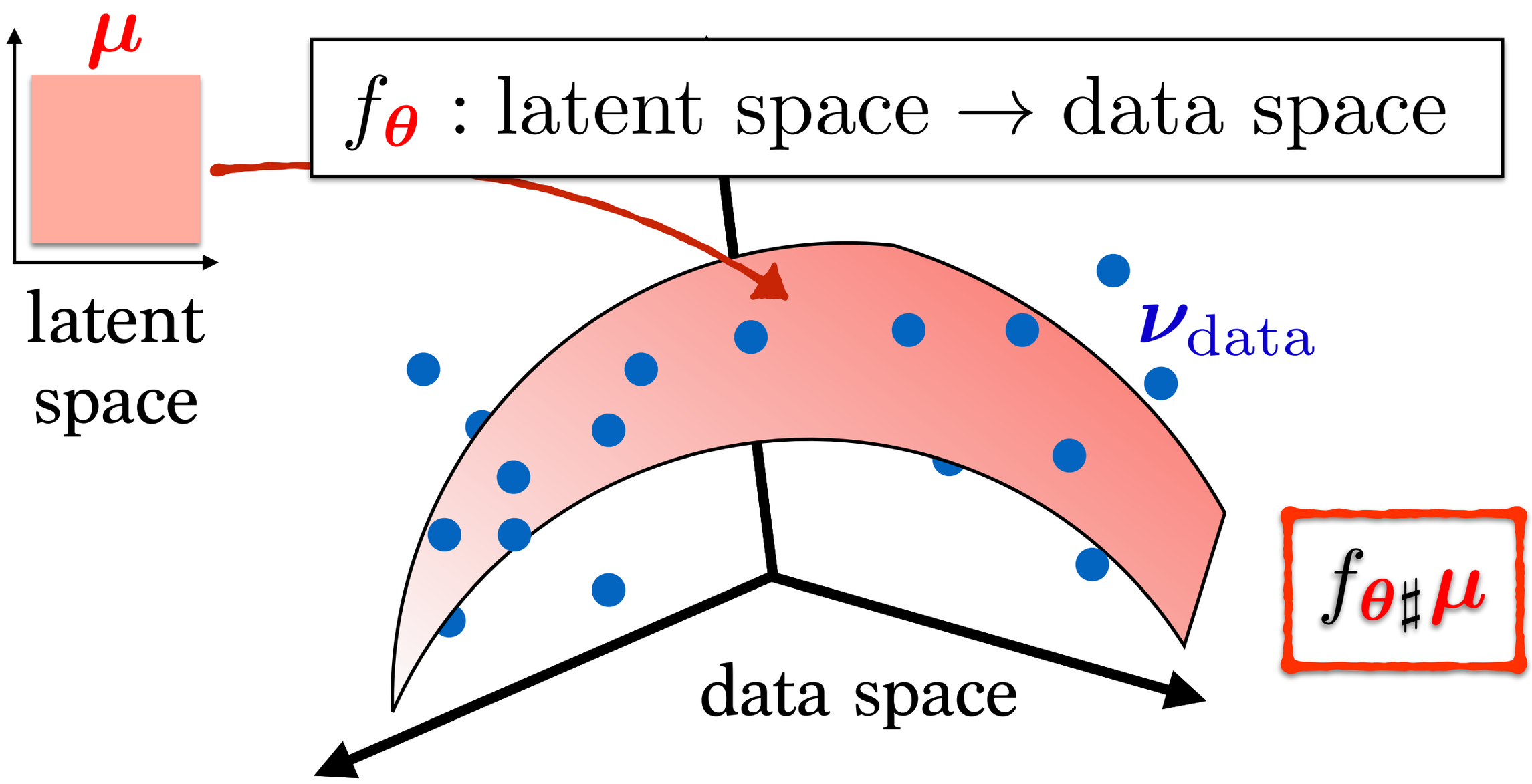




Generative Models

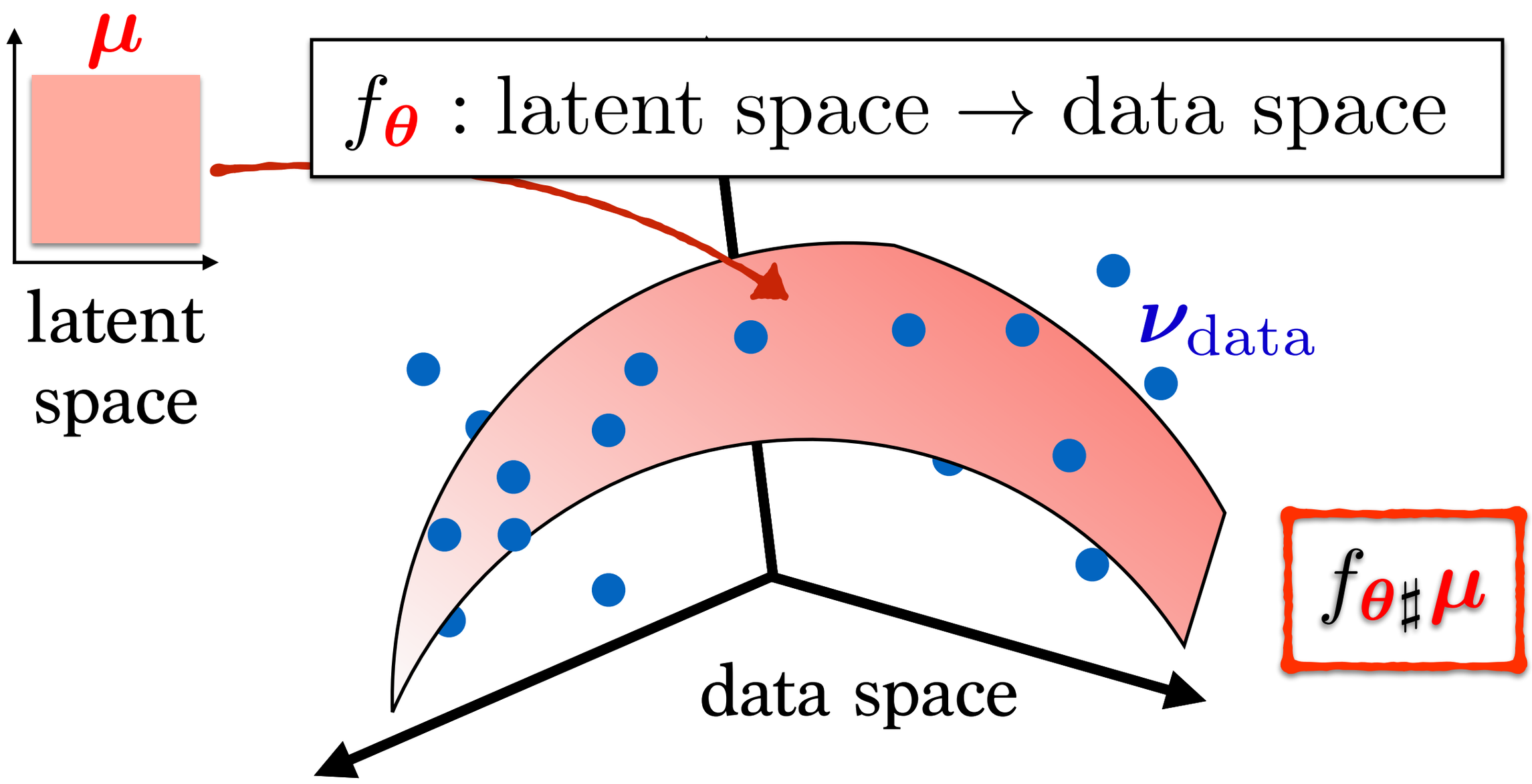


Generative Models



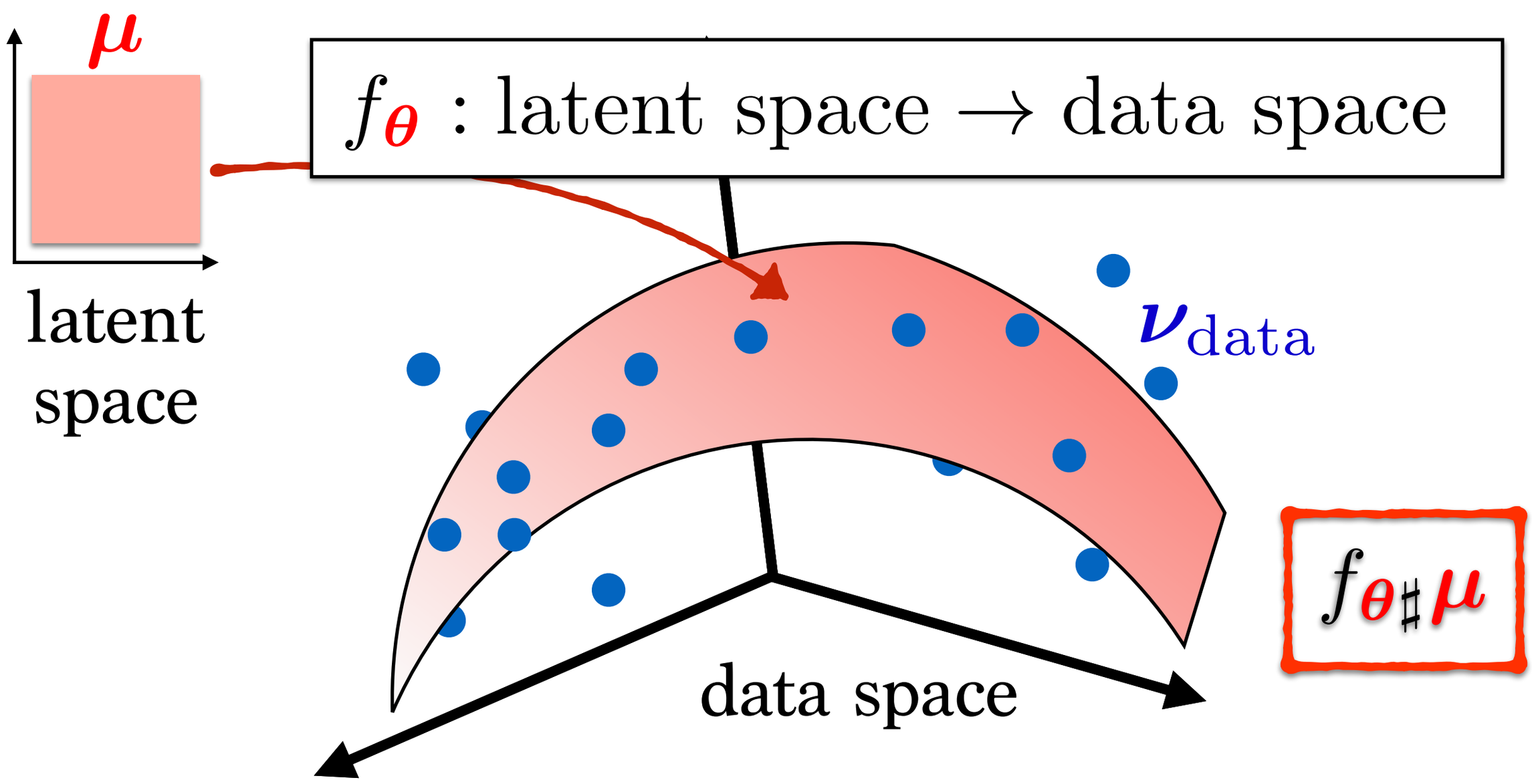
Goal: find θ such that $f_{\theta\sharp}\mu$ fits ν_{data}

Generative Models



Goal: find θ such that $f_{\theta\sharp}\mu$ fits ν_{data}

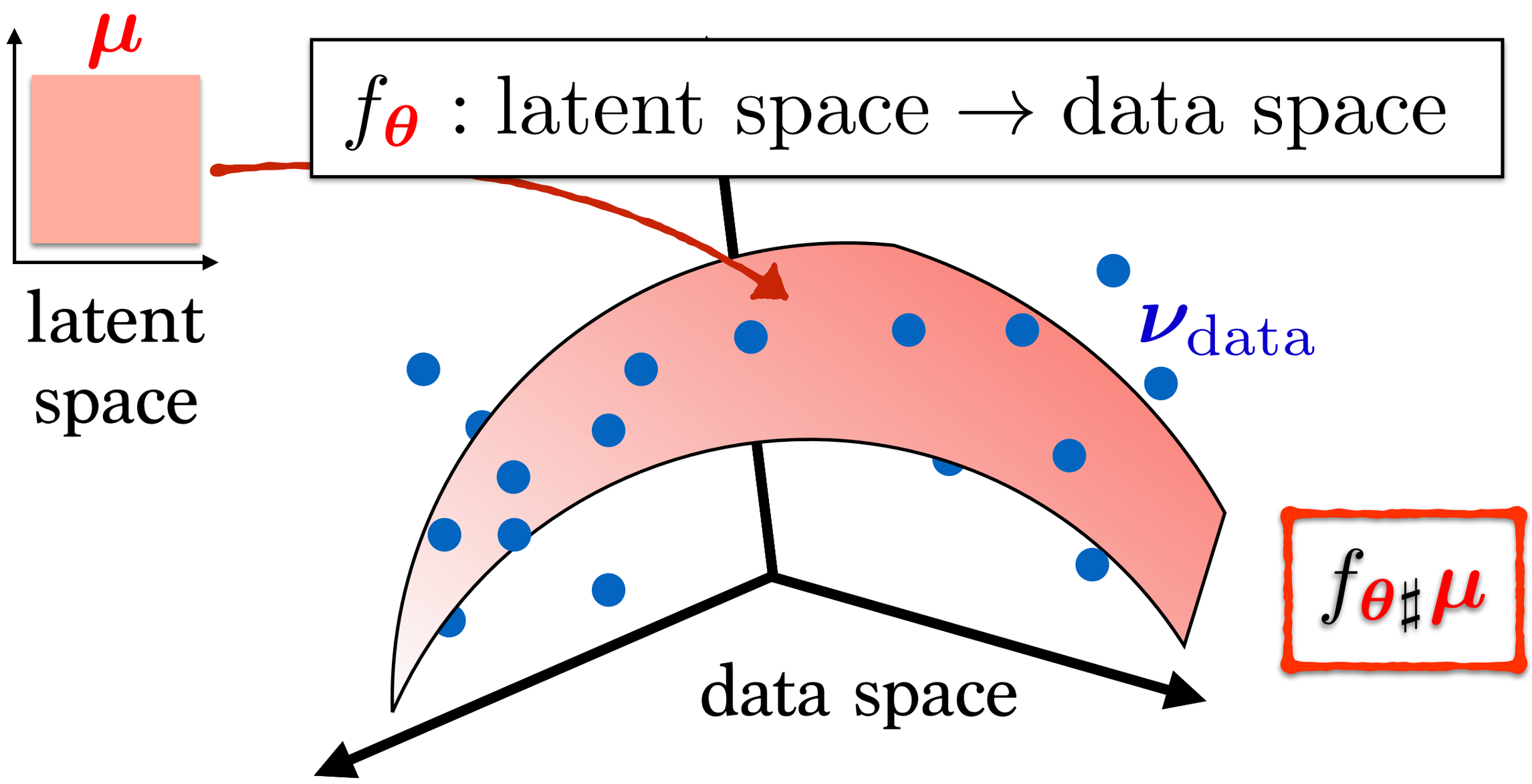
Generative Models



MLE

$$\max_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(x_i) \quad = \quad \min_{\theta\in\Theta} \mathrm{KL}(\nu_{\text{data}} \| p_\theta)$$

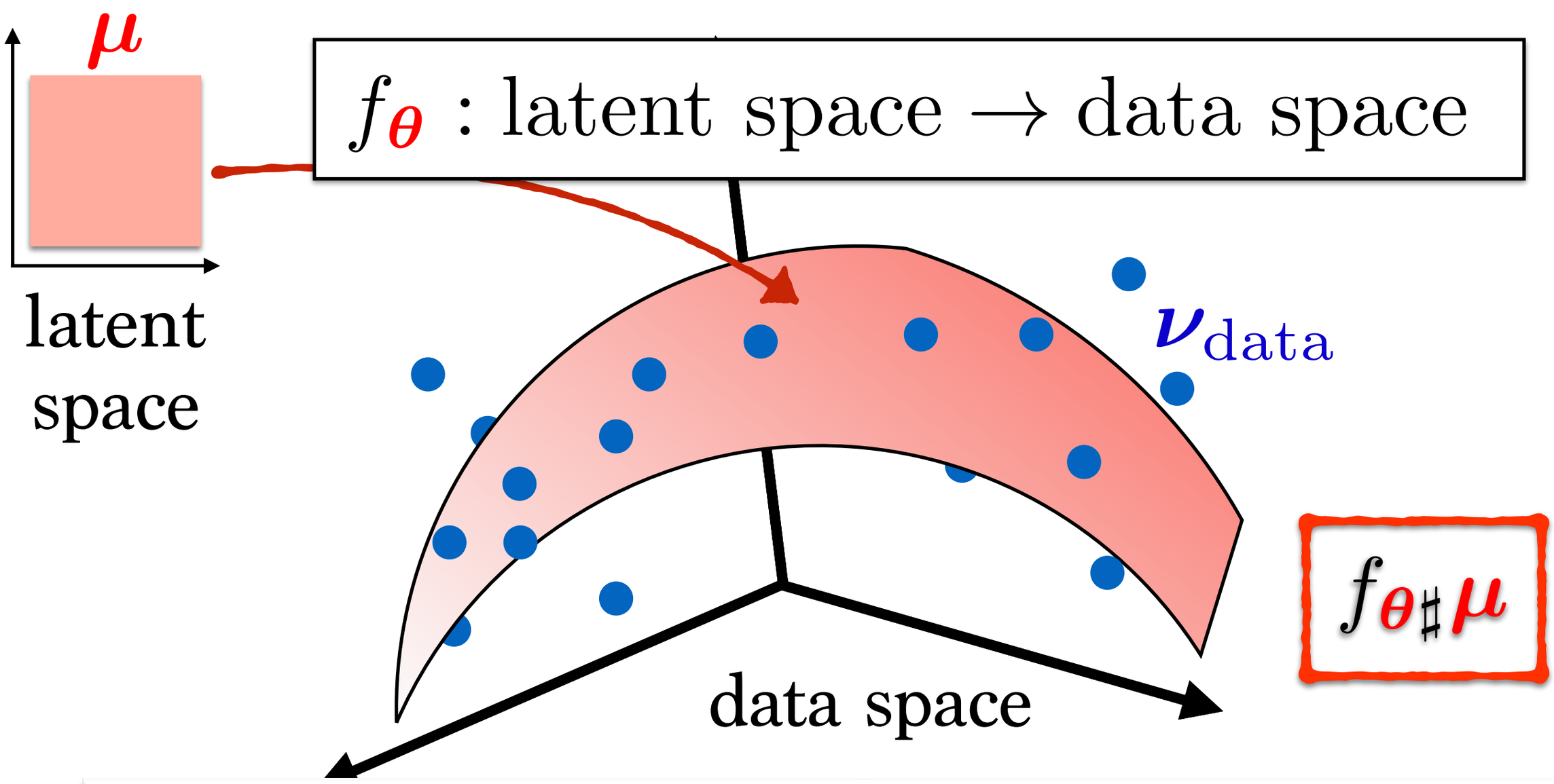
Generative Models



MLE

$$\max_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \log f_{\theta\sharp}\mu(x_i) \qquad \min_{\theta\in\Theta} \mathrm{KL}(\nu_{\text{data}} \| f_{\theta\sharp}\mu)$$

Generative Models

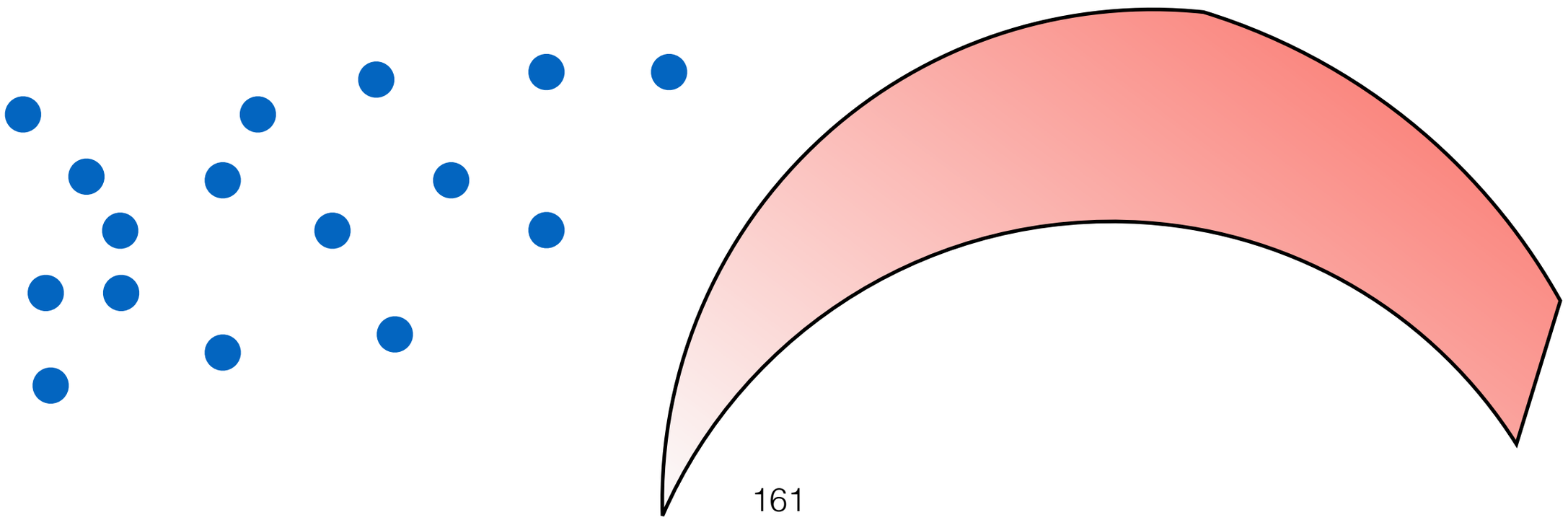


Need a more flexible **discrepancy function** to compare ν_{data} and $f_{\theta\sharp}\mu$

Workarounds?

- Formulation as **adversarial problem** **[GPM...'14]**

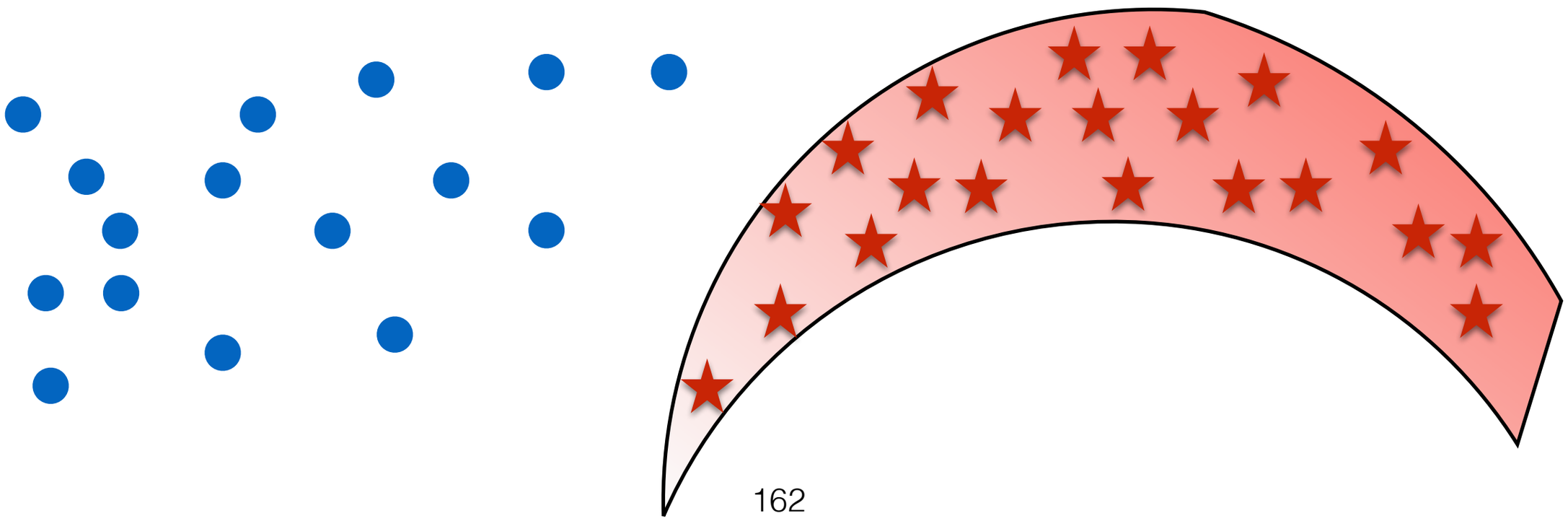
$$\min_{\boldsymbol{\theta}\in\Theta} \max_{\text{classifiers } \boldsymbol{g}} \text{Accuracy}_{\boldsymbol{g}}\left((\boldsymbol{f}_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, +1), (\boldsymbol{\nu}_{\text{data}}, -1)\right)$$



Workarounds?

- Formulation as **adversarial problem** **[GPM...'14]**

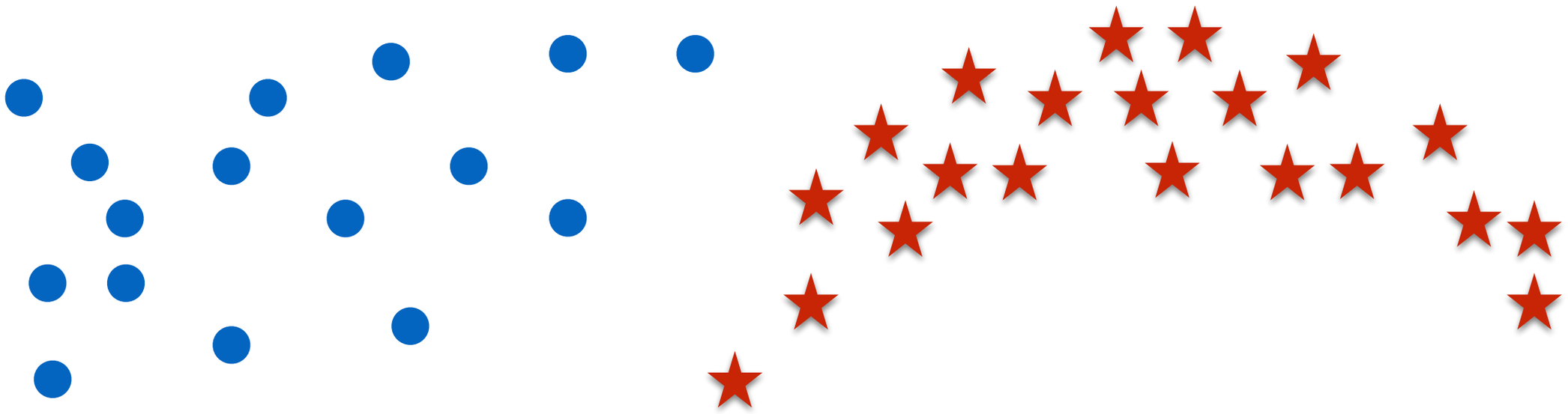
$$\min_{\boldsymbol{\theta}\in\Theta} \max_{\text{classifiers } \boldsymbol{g}} \text{Accuracy}_{\boldsymbol{g}}\left((\boldsymbol{f}_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, +1), (\boldsymbol{\nu}_{\text{data}}, -1)\right)$$



Workarounds?

- Formulation as **adversarial problem** **[GPM...'14]**

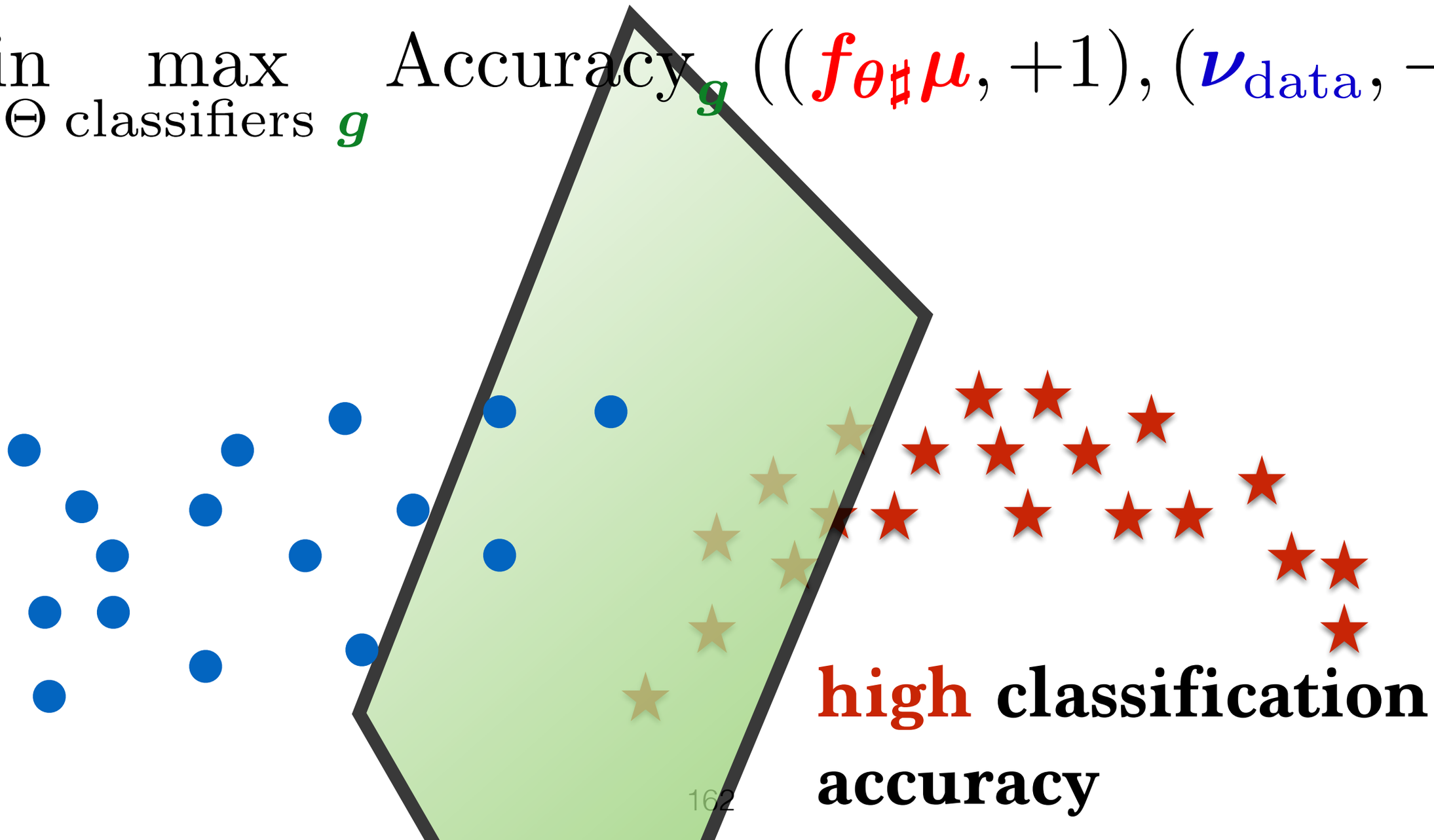
$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\text{classifiers } \boldsymbol{g}} \text{Accuracy}_{\boldsymbol{g}} \left((\boldsymbol{f}_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, +1), (\boldsymbol{\nu}_{\text{data}}, -1) \right)$$



Workarounds?

- Formulation as **adversarial problem** **[GPM...'14]**

$$\min_{\boldsymbol{\theta}\in\Theta} \max_{\text{classifiers } \boldsymbol{g}} \text{Accuracy}_{\boldsymbol{g}}\left((\boldsymbol{f}_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, +1), (\boldsymbol{\nu}_{\text{data}}, -1)\right)$$

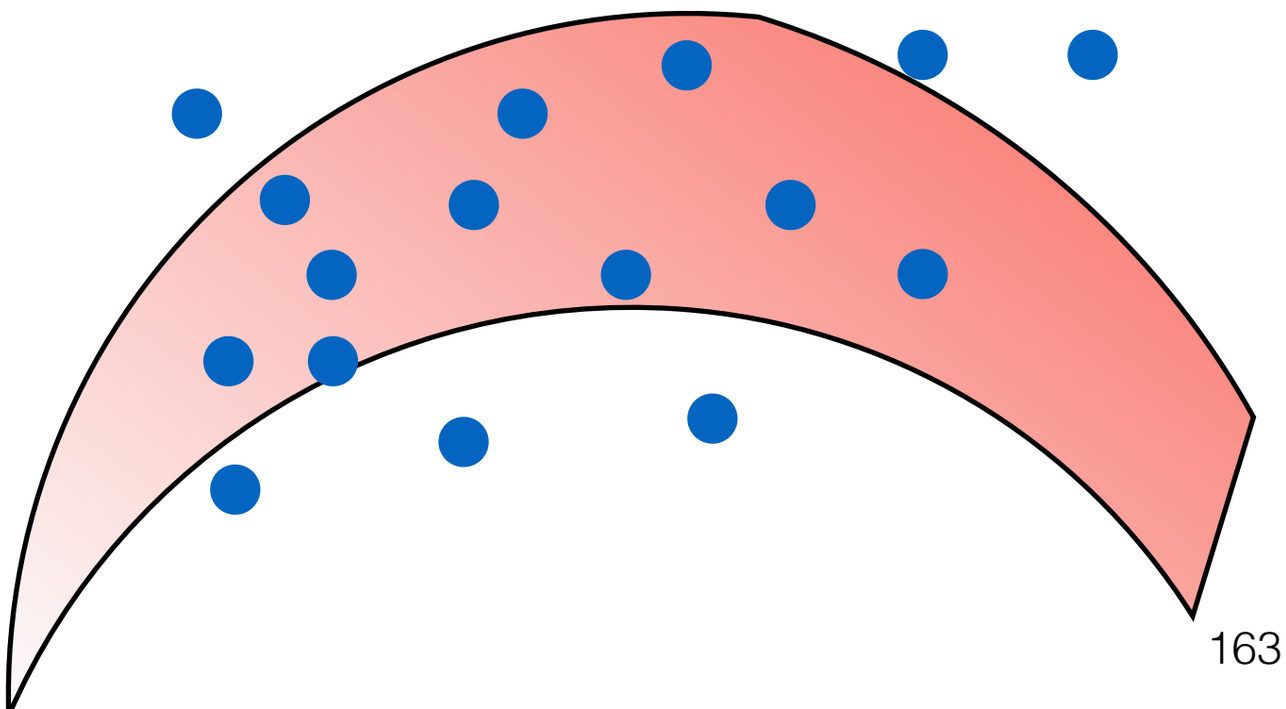


high classification accuracy

Workarounds?

- Formulation as **adversarial problem** **[GPM...'14]**

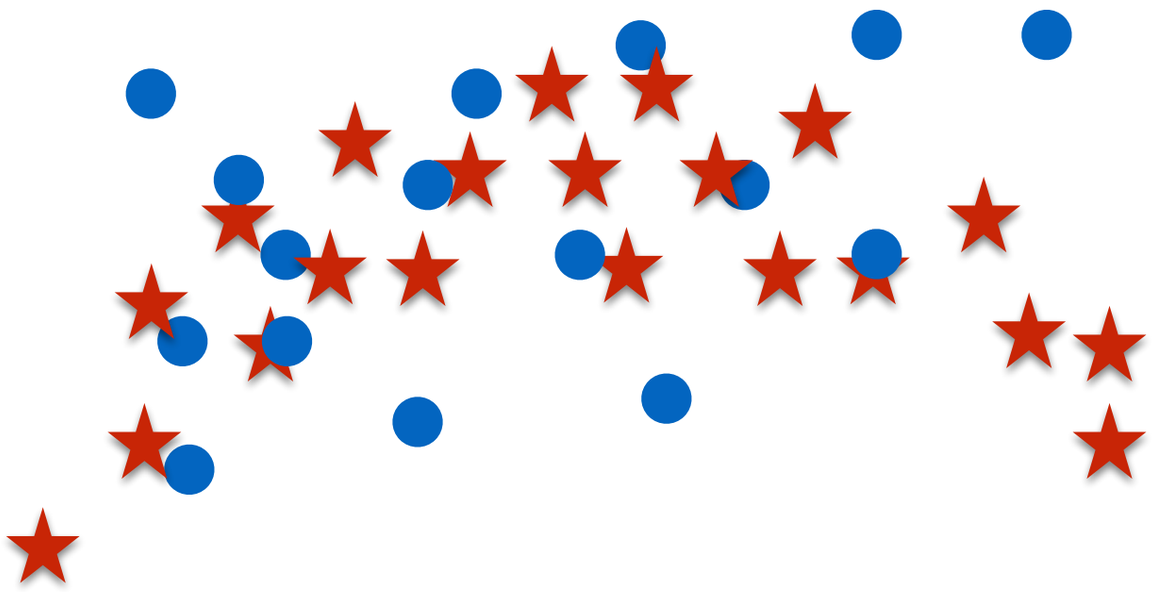
$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\text{classifiers } \boldsymbol{g}} \text{Accuracy}_{\boldsymbol{g}}\left((\boldsymbol{f}_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, +1), (\boldsymbol{\nu}_{\text{data}}, -1) \right)$$



Workarounds?

- Formulation as **adversarial problem** **[GPM...'14]**

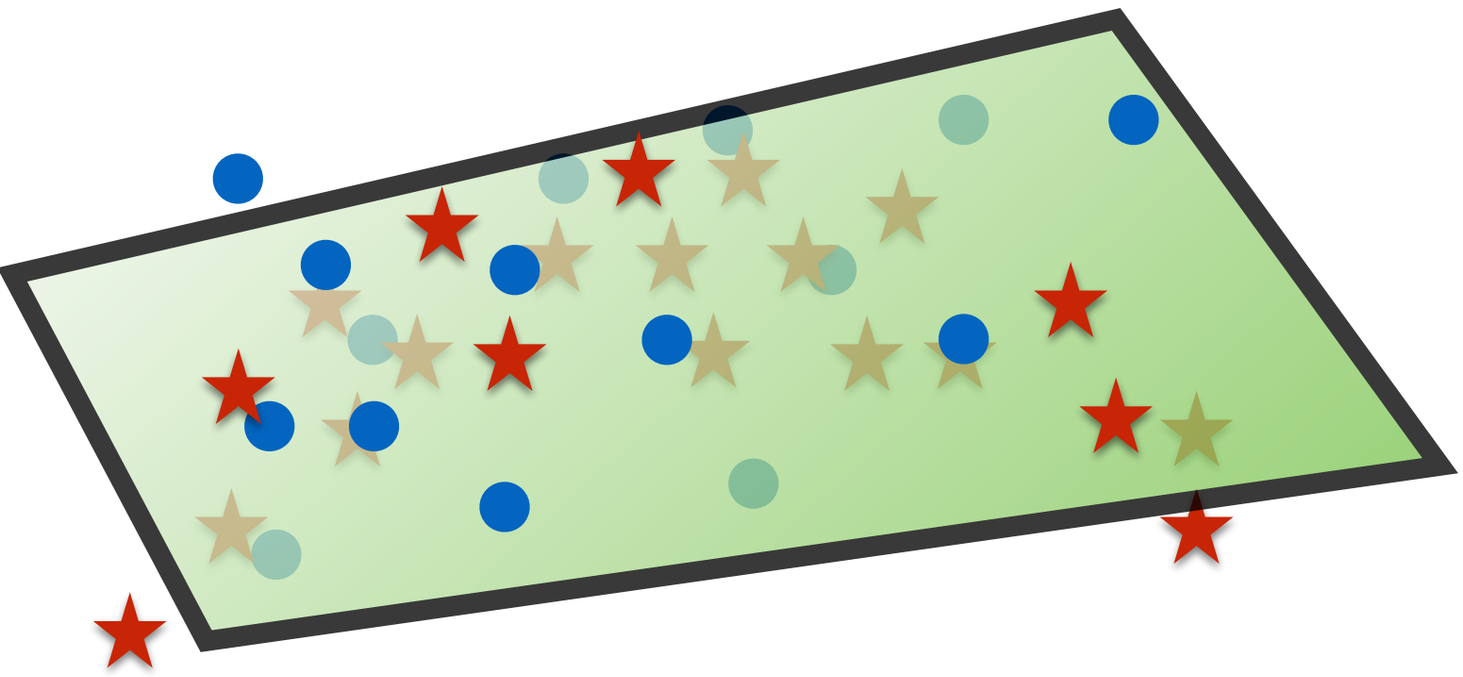
$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\text{classifiers } \boldsymbol{g}} \text{Accuracy}_{\boldsymbol{g}} \left((\boldsymbol{f}_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, +1), (\boldsymbol{\nu}_{\text{data}}, -1) \right)$$



Workarounds?

- Formulation as **adversarial problem** **[GPM...'14]**

$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\text{classifiers } \boldsymbol{g}} \text{Accuracy}_{\boldsymbol{g}} \left((\boldsymbol{f}_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, +1), (\boldsymbol{\nu}_{\text{data}}, -1) \right)$$



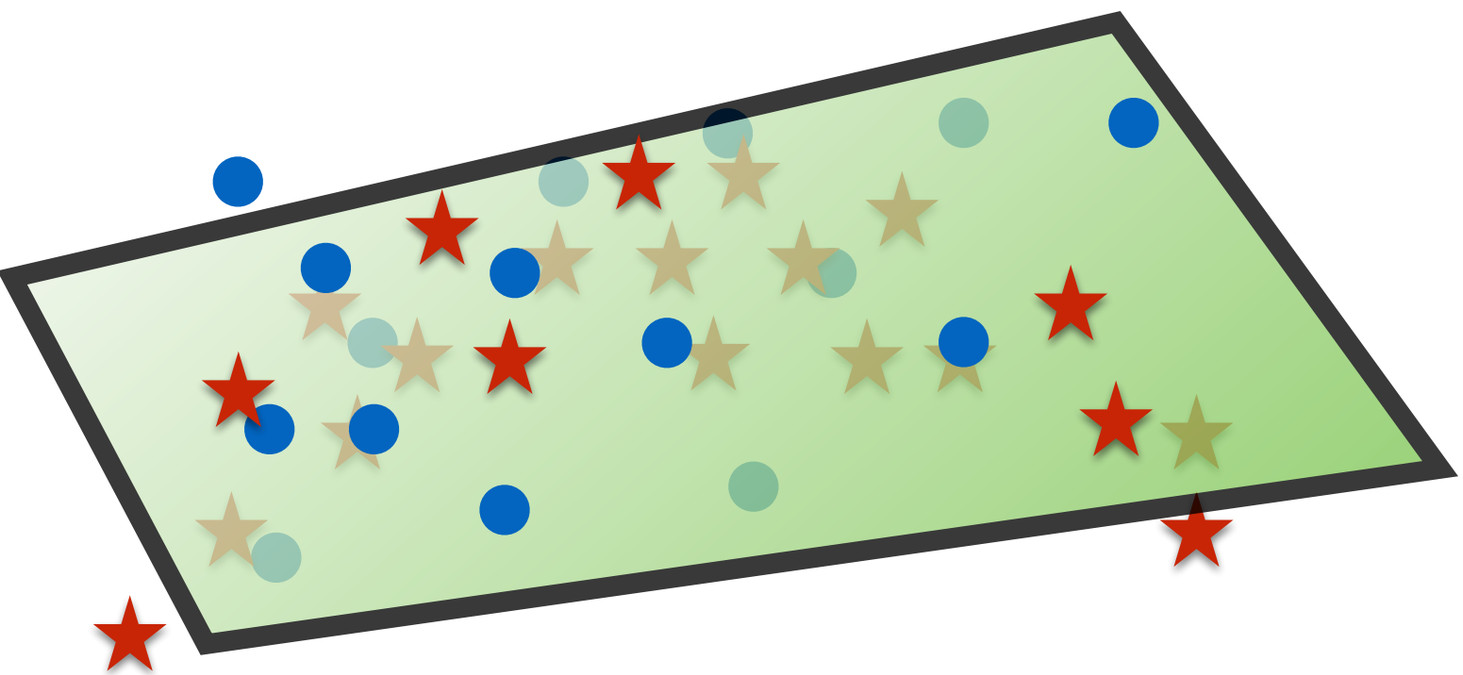
low classification accuracy...
is the goal.

Workarounds?

- Formulation as **adversarial problem** **[GPM...'14]**

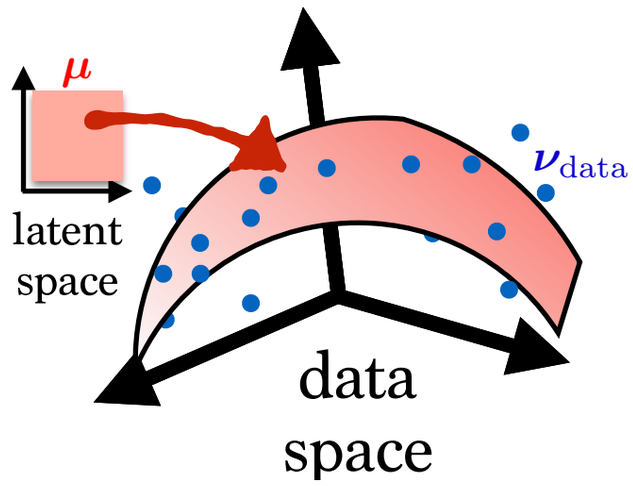
$$\min_{\boldsymbol{\theta}\in\Theta} \max_{\text{classifiers } \boldsymbol{g}} \text{Accuracy}_{\boldsymbol{g}}\left((\boldsymbol{f_\theta}\sharp\boldsymbol{\mu}, +1), (\boldsymbol{\nu}_{\text{data}}, -1)\right)$$

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_\theta(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



low classification accuracy...
is the goal.

Another idea?



- Use a **metric** Δ for probability measures, that can handle measures with non-overlapping supports:

$$\min_{\theta \in \Theta} \Delta(\nu_{\text{data}}, p_\theta), \quad \textbf{not} \min_{\theta \in \Theta} \mathrm{KL}(\nu_{\text{data}} \| p_\theta)$$

Minimum Δ Estimation

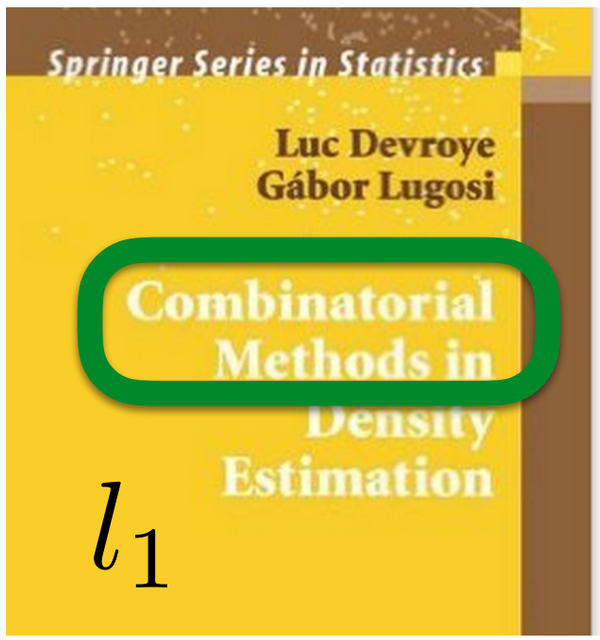
The Annals of Statistics
1980, Vol. 8, No. 3, 457–487

MINIMUM CHI-SQUARE, NOT MAXIMUM LIKELIHOOD!

BY JOSEPH BERKSON
Mayo Clinic, Rochester, Minnesota

ELSEVIER Computational Statistics & Data Analysis 29 (1998) 81–103

COMPUTATIONAL STATISTICS & DATA ANALYSIS



Minimum Hellinger distance estimation for Poisson mixtures

Dimitris Karlis, Evdokia Xekalaki*
Department of Statistics, Athens University of Economics and Business, 76 Patission Str., 104 34 Athens, Greece

ELSEVIER

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Statistics & Probability Letters 76 (2006) 1298–1302

STATISTICS & PROBABILITY LETTERS

www.elsevier.com/locate/stapro

On minimum Kantorovich distance estimators

Federico Bassetti[a], Antonella Bodini[b], Eugenio Regazzini[a,*]

Minimum Kantorovich Estimation

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Statistics & Probability Letters 76 (2006) 1298–1302
STATISTICS & PROBABILITY LETTERS
www.elsevier.com/locate/stapro

On minimum Kantorovich distance estimators

Federico Bassetti[a], Antonella Bodini[b], Eugenio Regazzini[a,*]

Use *Wasserstein distances* to define a loss between data and model.

$$\min_{\boldsymbol{\theta}\in\Theta} W(\boldsymbol{\nu}_{\text{data}}, p_{\boldsymbol{\theta}})$$

Minimum Kantorovich Estimators

$$\min_{\boldsymbol{\theta} \in \Theta} W(\boldsymbol{\nu}_{\text{data}}, f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu})$$

[Bassetti'06] 1st reference discussing this approach.

Challenge: $\nabla_{\boldsymbol{\theta}} W(\boldsymbol{\nu}_{\text{data}}, f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu})$?

[Montavon'16] use regularized OT in a finite setting.

[Arjovsky'17] *(WGAN)* uses a NN to approximate dual solutions and recover gradient w.r.t. parameter

[Bernton'17] *(Wasserstein ABC)*

[Genevay'17, Salimans'17] *(Sinkhorn approach)*

Algorithms - GAN

Remember: **logistic regression objective**

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

here $x^{(i)}$ is a point, $y^{(i)}$ a {0,1} label

Algorithms - GAN

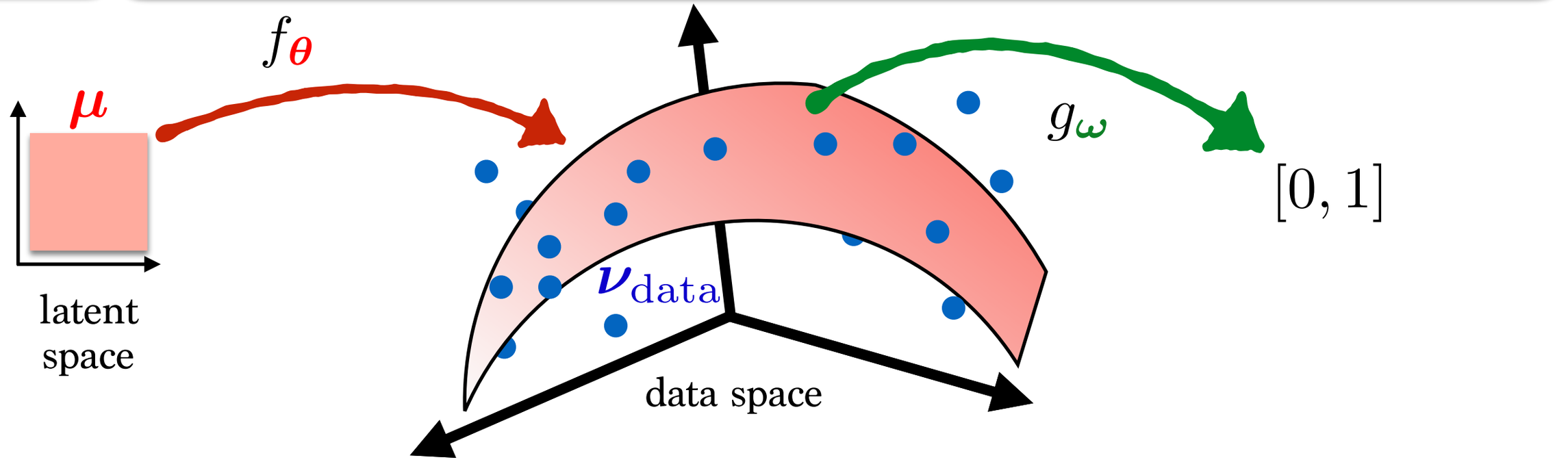
$$\min_{\theta} \max_{\omega} \mathbb{E}_{Z \sim \mu} \log g_{\omega}(f_{\theta}(Z)) + \mathbb{E}_{Y \sim \nu_{\text{data}}} \log(1 - g_{\omega}(Y))$$

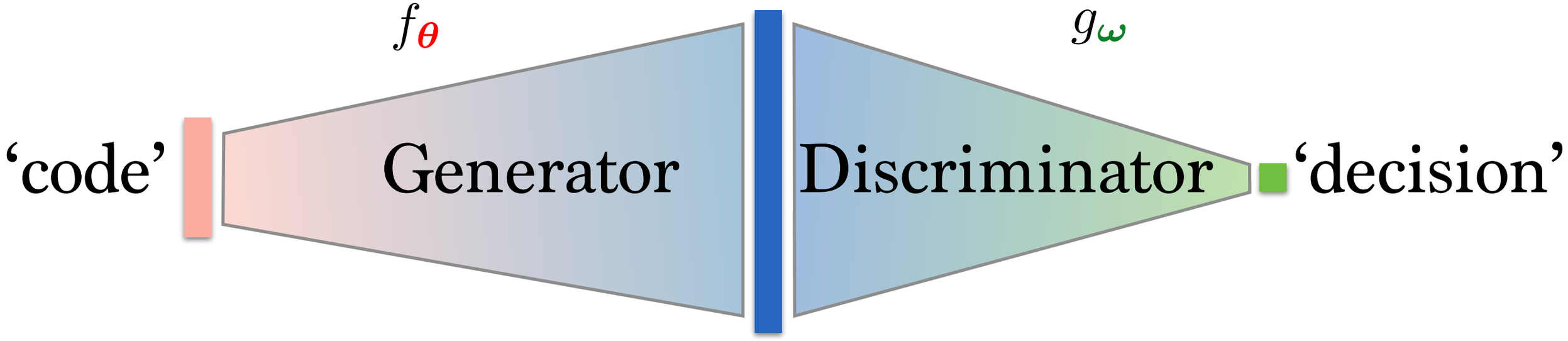
Remember: **logistic regression objective**

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

here $x^{(i)}$ is a point, $y^{(i)}$ a {0,1} label

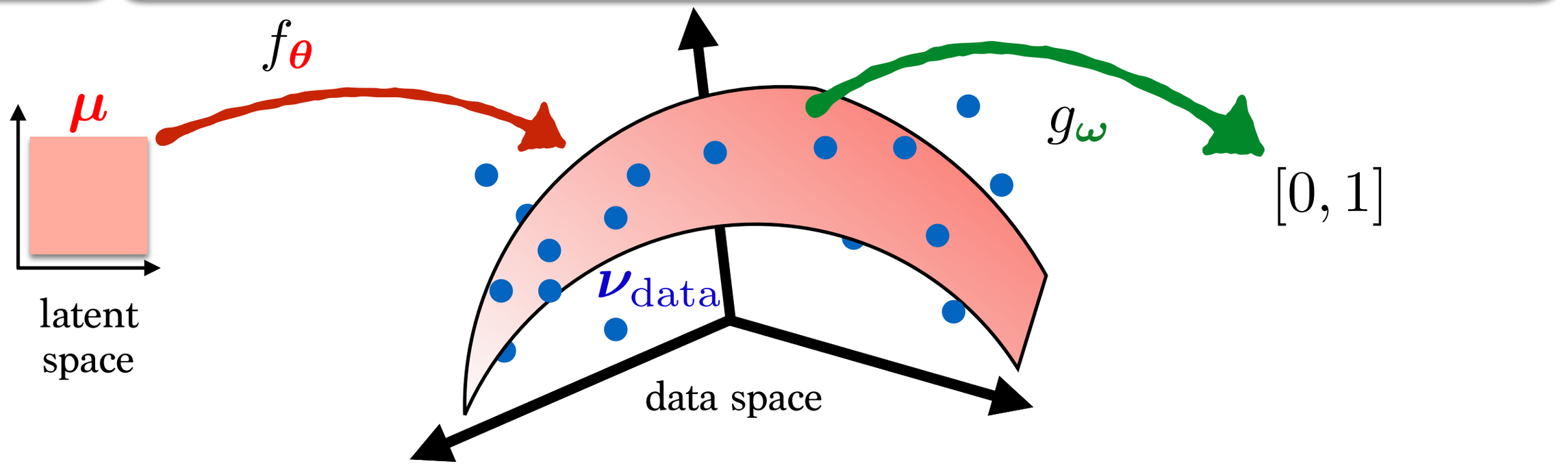
Algorithms - GAN

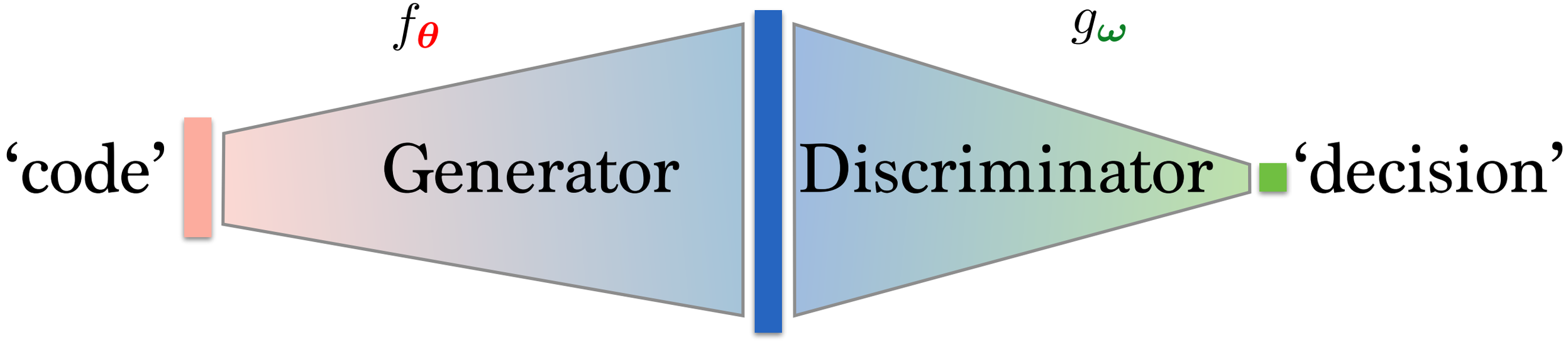




Algorithms - GAN

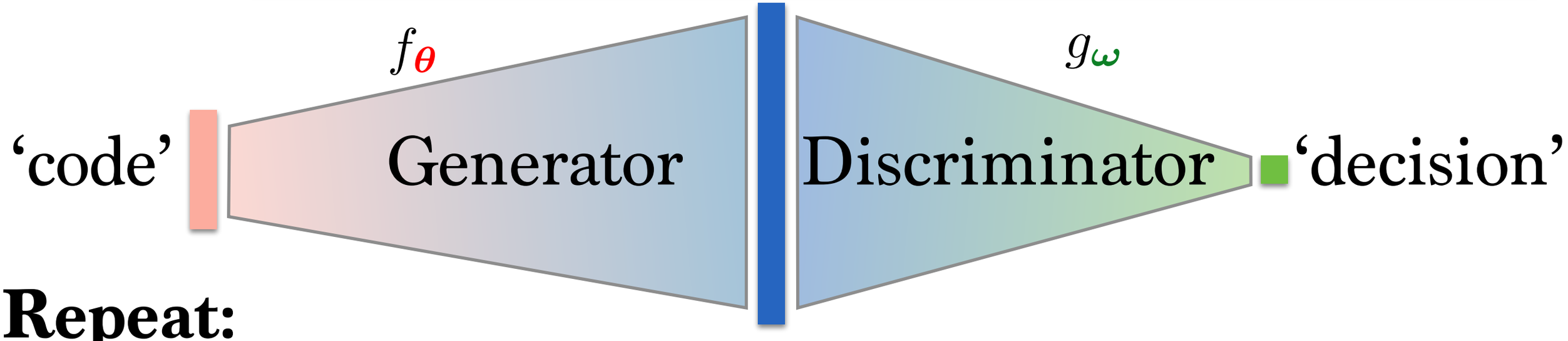
$$\min_{\theta} \max_{\omega} \mathbb{E}_{Z \sim \mu} \log g_{\omega}(f_{\theta}(Z)) + \mathbb{E}_{Y \sim \nu_{\text{data}}} \log(1 - g_{\omega}(Y))$$





Algorithms - GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



Repeat:

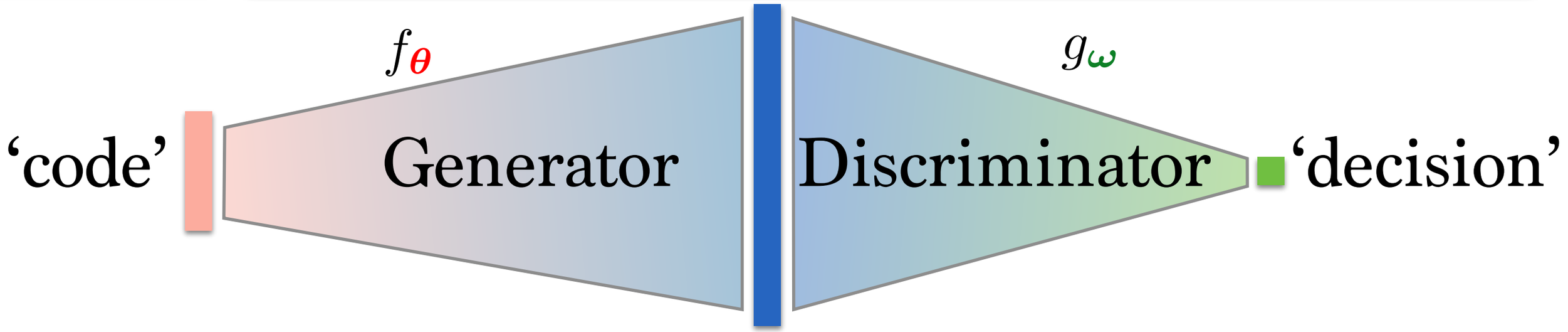
1. Sample $z_1, z_2, \ldots, z_n \sim \mu$, $y_1, y_2, \ldots, y_n \sim \nu$

2. $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} \left[\sum_{i=1}^{n} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{z_i})) \right]$$

3. $$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \nabla_{\boldsymbol{\omega}} \left[\sum_{i=1}^{n} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{z_i})) + \sum_{j=1}^{n} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{y_j})) \right]$$

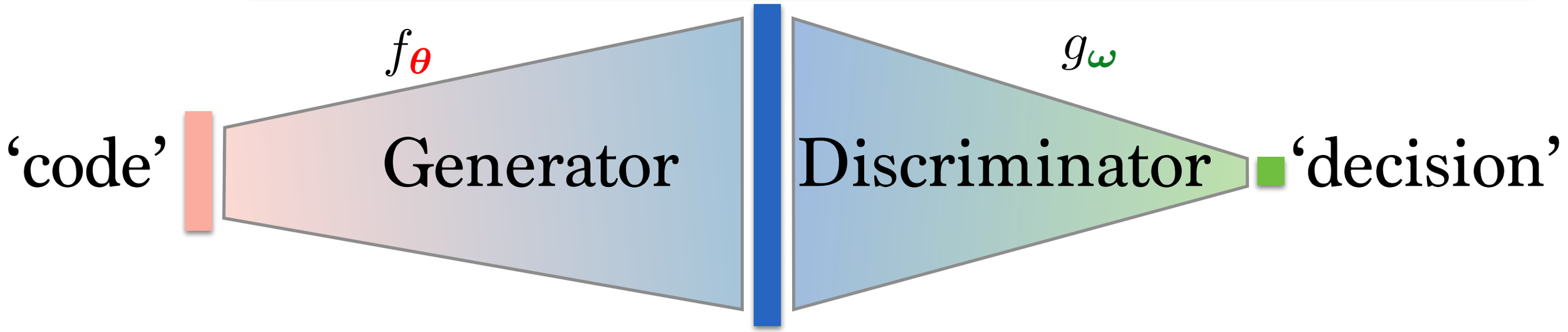
Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



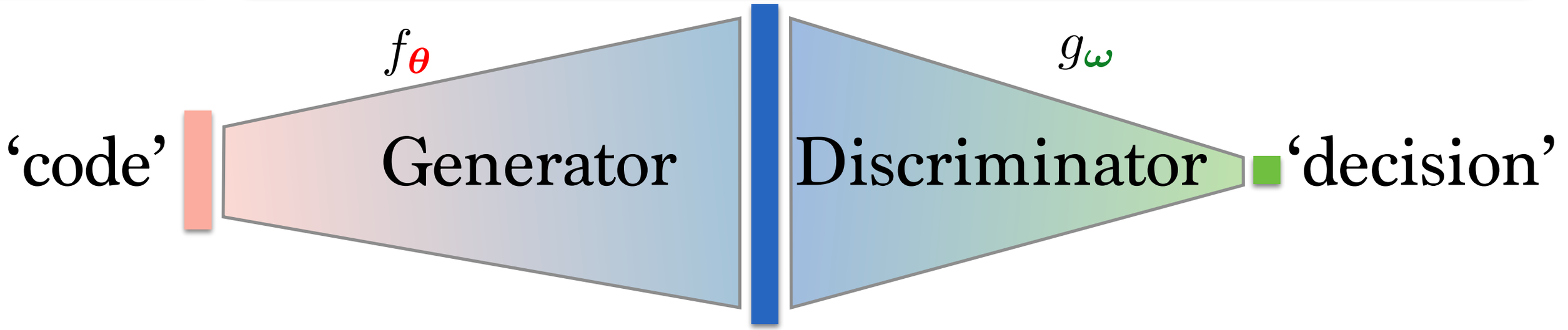
Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



Variants - Wasserstein GAN

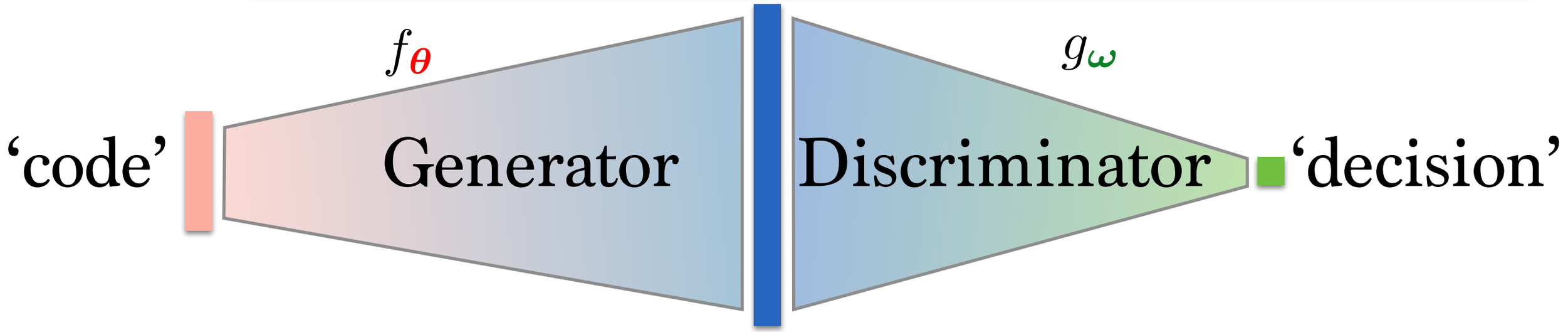
$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) - \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$

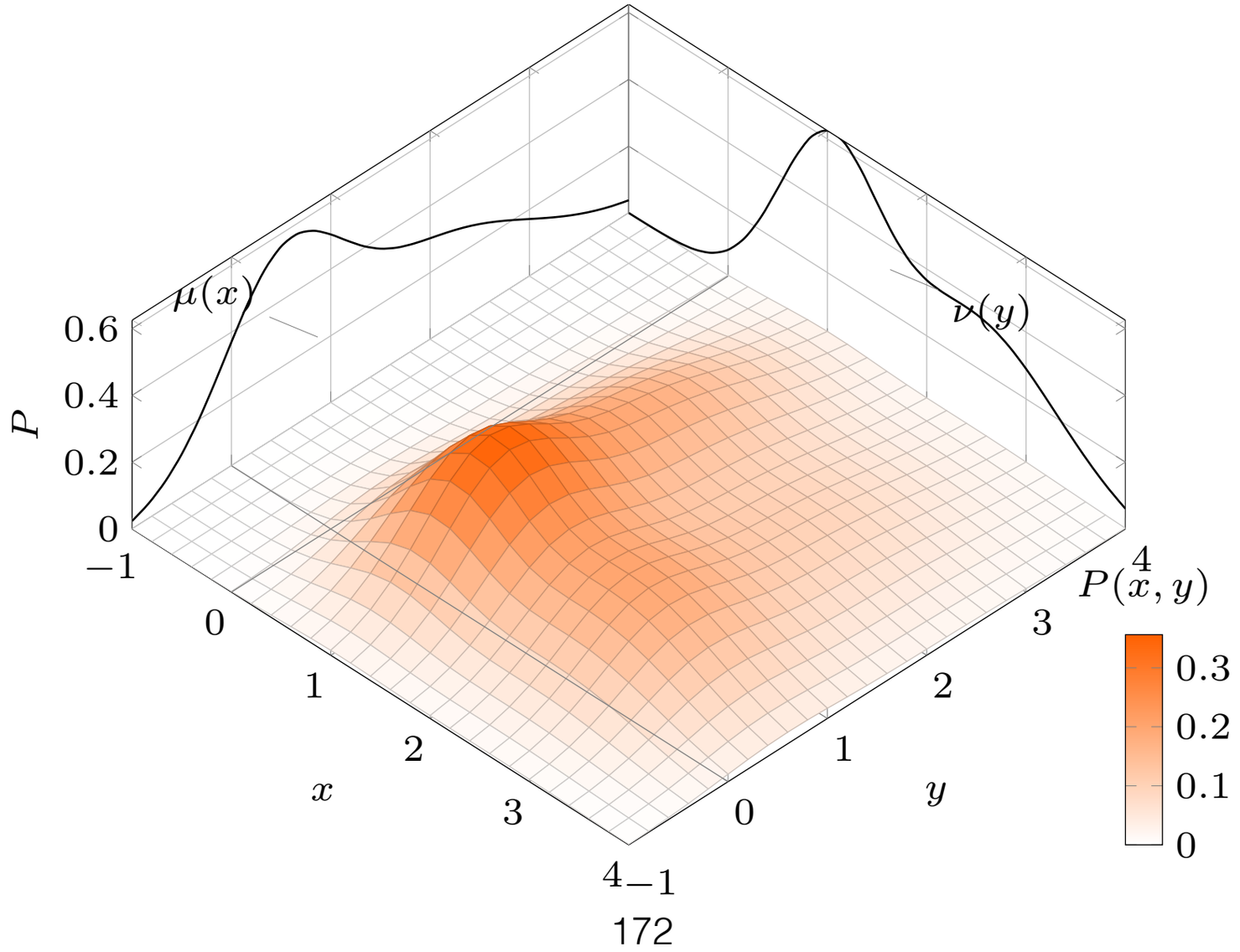


$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) - \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

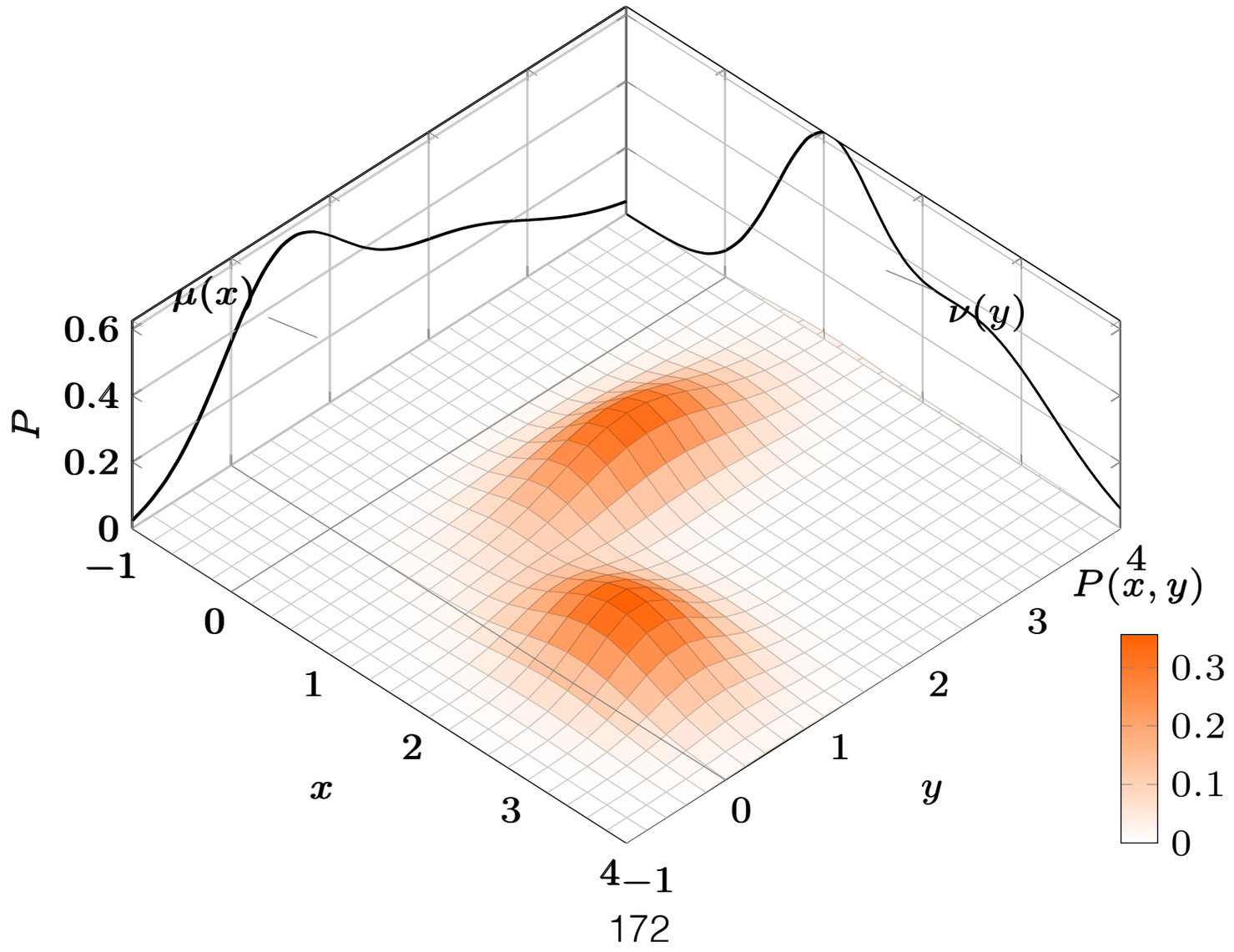
Wasserstein: Probability Couplings

$$\Pi(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \{\boldsymbol{P} \in \mathcal{P}(\Omega \times \Omega) | \forall \boldsymbol{A}, \boldsymbol{B} \subset \Omega,$$
$$\boldsymbol{P}(\boldsymbol{A} \times \Omega) = \boldsymbol{\mu}(\boldsymbol{A}), \boldsymbol{P}(\Omega \times \boldsymbol{B}) = \boldsymbol{\nu}(\boldsymbol{B})\}$$



Wasserstein: Probability Couplings

$$\Pi(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \{\boldsymbol{P} \in \mathcal{P}(\Omega \times \Omega) | \forall \boldsymbol{A}, \boldsymbol{B} \subset \Omega,$$
$$\boldsymbol{P}(\boldsymbol{A} \times \Omega) = \boldsymbol{\mu}(\boldsymbol{A}), \boldsymbol{P}(\Omega \times \boldsymbol{B}) = \boldsymbol{\nu}(\boldsymbol{B})\}$$



Kantorovich Problem

Def. Given $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$; a cost function $\boldsymbol{c}$ on $\Omega \times \Omega$, the Kantorovich problem is

$$\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \iint \boldsymbol{c}(x, y) \boldsymbol{P}(dx, dy).$$

PRIMAL

Kantorovich Problem

Def. Given $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$; a cost function $\boldsymbol{c}$ on $\Omega \times \Omega$, the Kantorovich problem is

$$\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \iint \boldsymbol{c}(x, y) \boldsymbol{P}(dx, dy).$$

PRIMAL

$$\sup_{\substack{\boldsymbol{\varphi} \in L_1(\boldsymbol{\mu}), \boldsymbol{\psi} \in L_1(\boldsymbol{\nu}) \\ \boldsymbol{\varphi}(x) + \boldsymbol{\psi}(y) \leq \boldsymbol{c}(x, y)}} \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu}.$$

DUAL

Kantorovich Problem

Def. Given $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$; a cost function $\boldsymbol{c}$ on $\Omega \times \Omega$, the Kantorovich problem is

$$\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \iint \boldsymbol{c}(x, y) \boldsymbol{P}(dx, dy).$$

PRIMAL

For two real-valued functions $\boldsymbol{\varphi}, \boldsymbol{\psi}$ on Ω,

$$(\boldsymbol{\varphi} \oplus \boldsymbol{\psi})(x, y) \stackrel{\text{def}}{=} \boldsymbol{\varphi}(x) + \boldsymbol{\psi}(y)$$

Kantorovich Problem

Def. Given $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$; a cost function $\boldsymbol{c}$ on $\Omega \times \Omega$, the Kantorovich problem is

$$\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \iint \boldsymbol{c}(x, y) \boldsymbol{P}(dx, dy).$$

PRIMAL

$$\sup_{\substack{\boldsymbol{\varphi} \in L_1(\boldsymbol{\mu}), \boldsymbol{\psi} \in L_1(\boldsymbol{\nu}) \\ \boldsymbol{\varphi} \oplus \boldsymbol{\psi} \leq \boldsymbol{c}}} \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu}.$$

DUAL

Deriving Kantorovich Duality

Let $\boldsymbol{\varphi}, \boldsymbol{\psi} : \Omega \to \mathbb{R}$, and $\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})$.

$$\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} = ?$$

Deriving Kantorovich Duality

Let $\boldsymbol{\varphi}, \boldsymbol{\psi} : \Omega \to \mathbb{R}$, and $\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})$.

$$\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} = ?$$

$$\iint \boldsymbol{\varphi}(\boldsymbol{x}) d\boldsymbol{P}(\boldsymbol{x}, y) + \iint \boldsymbol{\psi}(\boldsymbol{y}) d\boldsymbol{P}(x, \boldsymbol{y})$$

Deriving Kantorovich Duality

Let $\boldsymbol{\varphi}, \boldsymbol{\psi} : \Omega \to \mathbb{R}$, and $\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})$.

$$\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} = ?$$

$$\iint \boldsymbol{\varphi}(\boldsymbol{x}) d\boldsymbol{P}(\boldsymbol{x}, y) + \iint \boldsymbol{\psi}(\boldsymbol{y}) d\boldsymbol{P}(x, \boldsymbol{y})$$

Deriving Kantorovich Duality

Let $\boldsymbol{\varphi}, \boldsymbol{\psi} : \Omega \to \mathbb{R}$, and $\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})$.

$$\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} = \mathbf{0}$$

Deriving Kantorovich Duality

Let $\boldsymbol{\varphi}, \boldsymbol{\psi} : \Omega \to \mathbb{R}$, and $\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})$.

$$\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} = \mathbf{0}$$

Let $\boldsymbol{\varphi}, \boldsymbol{\psi} : \Omega \to \mathbb{R}$, and $\boldsymbol{P} \in \mathcal{P}_+(\Omega^2)$.

Deriving Kantorovich Duality

Let $\boldsymbol{\varphi}, \boldsymbol{\psi} : \Omega \to \mathbb{R}$, and $\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})$.

$$\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} = \mathbf{0}$$

Let $\boldsymbol{\varphi}, \boldsymbol{\psi} : \Omega \to \mathbb{R}$, and $\boldsymbol{P} \in \mathcal{P}_+(\Omega^2)$.

$$\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} =$$

$$\int \boldsymbol{\varphi} d \underbrace{(\boldsymbol{\mu} - \boldsymbol{P_X})}_{\neq 0} + \int \boldsymbol{\psi} d \underbrace{(\boldsymbol{\nu} - \boldsymbol{P_Y})}_{\neq 0}$$

and / or

Deriving Kantorovich Duality

$$\iota_{\Pi}(\boldsymbol{P}) = \sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \left[\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} \right]$$

$$= \begin{cases} 0 & \text{if } \boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu}), \\ +\infty & \text{otherwise.} \end{cases}$$

Deriving Kantorovich Duality

$$\iota_\Pi(\boldsymbol{P}) = \sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \left[\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} \right]$$

$$= \begin{cases} 0 & \text{if } \boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu}), \\ +\infty & \text{otherwise.} \end{cases}$$

$$\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu},\boldsymbol{\nu})} \iint \boldsymbol{c}\, d\boldsymbol{P}$$

Deriving Kantorovich Duality

$$\iota_\Pi(\boldsymbol{P}) = \sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \left[\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} \right]$$

$$= \begin{cases} 0 & \text{if } \boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu}), \\ +\infty & \text{otherwise.} \end{cases}$$

$$\inf_{\cancel{\boldsymbol{P} \in \Pi(\boldsymbol{\mu},\boldsymbol{\nu})}} \iint \boldsymbol{c}\, d\boldsymbol{P}$$

$$\inf_{\boldsymbol{P} \in \mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \boxed{\iota_\Pi(\boldsymbol{P})}$$

Deriving Kantorovich Duality

$$\iota_\Pi(\boldsymbol{P}) = \sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \left[\int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P} \right]$$

$$= \begin{cases} 0 & \text{if } \boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu}), \\ +\infty & \text{otherwise.} \end{cases}$$

$$\inf_{\boldsymbol{P} \in \mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \iota_\Pi(\boldsymbol{P})$$

Deriving Kantorovich Duality

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \iota_{\Pi}(\boldsymbol{P})$$

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P}$$

Deriving Kantorovich Duality

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \iota_{\Pi}(\boldsymbol{P})$$

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \iint \boldsymbol{c}\, d\boldsymbol{P} + \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu} - \iint \boldsymbol{\varphi} \oplus \boldsymbol{\psi} d\boldsymbol{P}$$

Deriving Kantorovich Duality

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \iota_{\Pi}(\boldsymbol{P})$$

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \iint \boldsymbol{c}\, d\boldsymbol{P} - \iint \boldsymbol{\varphi}\oplus\boldsymbol{\psi} d\boldsymbol{P} + \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu}$$

Deriving Kantorovich Duality

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \iota_{\Pi}(\boldsymbol{P})$$

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \iint (\boldsymbol{c} - \boldsymbol{\varphi} \oplus \boldsymbol{\psi}) d\boldsymbol{P} \qquad + \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu}$$

Deriving Kantorovich Duality

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \iota_{\Pi}(\boldsymbol{P})$$

$$\sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint (\boldsymbol{c} - \boldsymbol{\varphi} \oplus \boldsymbol{\psi}) d\boldsymbol{P} \qquad + \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu}$$

Deriving Kantorovich Duality

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \iota_{\Pi}(\boldsymbol{P})$$

$$\sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint (\boldsymbol{c} - \boldsymbol{\varphi} \oplus \boldsymbol{\psi}) d\boldsymbol{P} + \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu}$$

Deriving Kantorovich Duality

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \iota_{\Pi}(\boldsymbol{P})$$

$$\sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint (\boldsymbol{c} - \boldsymbol{\varphi} \oplus \boldsymbol{\psi}) d\boldsymbol{P} + \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu}$$

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega)} \iint (\boldsymbol{c} - \boldsymbol{\varphi} \oplus \boldsymbol{\psi}) d\boldsymbol{P} = \begin{cases} 0 & \text{if } \boldsymbol{c} - \boldsymbol{\varphi} \oplus \boldsymbol{\psi} \geq 0. \\ -\infty & \text{otherwise} \end{cases}$$

Deriving Kantorovich Duality

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint \boldsymbol{c}\, d\boldsymbol{P} + \iota_{\Pi}(\boldsymbol{P})$$

$$\sup_{\boldsymbol{\varphi},\boldsymbol{\psi}} \inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega^2)} \iint (\boldsymbol{c} - \boldsymbol{\varphi} \oplus \boldsymbol{\psi}) d\boldsymbol{P} + \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu}$$

$$\inf_{\boldsymbol{P}\in\mathcal{P}_+(\Omega)} \iint (\boldsymbol{c} - \boldsymbol{\varphi} \oplus \boldsymbol{\psi}) d\boldsymbol{P} = \begin{cases} 0 & \text{if } \boldsymbol{c} - \boldsymbol{\varphi} \oplus \boldsymbol{\psi} \geq 0. \\ -\infty & \text{otherwise} \end{cases}$$

$$\sup_{\boldsymbol{\varphi}\oplus\boldsymbol{\psi}\leq\boldsymbol{c}} \int \boldsymbol{\varphi} d\boldsymbol{\mu} + \int \boldsymbol{\psi} d\boldsymbol{\nu}.$$

DUAL

Wasserstein Distances

Let $p \geq 1$. Let $\boldsymbol{c}(x, y) := \boldsymbol{D}^p(x, y)$, a metric.

Def. The p-**Wasserstein distance** between $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$ is

$$W_p(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \left(\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \iint \boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y})^p \boldsymbol{P}(dx, dy) \right)^{1/p}.$$

Wasserstein Distances

Let $p \geq 1$. Let $\boldsymbol{c}(x, y) := \boldsymbol{D}^p(x, y)$, a metric.

Def. The p-**Wasserstein distance** between $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$ is

$$W_p(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \left(\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \iint \boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y})^p \boldsymbol{P}(dx, dy) \right)^{1/p}.$$

Wasserstein Distances

Let $p \geq 1$. Let $\boldsymbol{c}(x, y) := \boldsymbol{D}^p(x, y)$, a metric.

Def. The p-**Wasserstein distance** between $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$ is

$$W_p(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \left(\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \iint \boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y})^p \boldsymbol{P}(dx, dy) \right)^{1/p}.$$

Wasserstein Distances

Let $p \geq 1$. Let $\boldsymbol{c}(x, y) := \boldsymbol{D}^p(x, y)$, a metric.

Def. The p-**Wasserstein distance** between $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$ is

$$W_p(\boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} \left(\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \iint \boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y})^p \boldsymbol{P}(dx, dy) \right)^{1/p}.$$

$$W_1(\boldsymbol{\mu}, \boldsymbol{\nu}) = \sup_{\boldsymbol{\varphi}\ 1\text{-Lipschitz}} \int \boldsymbol{\varphi}(d\boldsymbol{\mu} - d\boldsymbol{\nu}).$$

W1

Wasserstein Distances

Let $p \geq 1$. Let $\boldsymbol{c}(x, y) := \boldsymbol{D}^p(x, y)$, a metric.

Def. The p-**Wasserstein distance** between $\boldsymbol{\mu}, \boldsymbol{\nu}$ in $\mathcal{P}(\Omega)$ is

$$\left(\inf_{\boldsymbol{P} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})} \int\int \boldsymbol{D}(x, y)^p \boldsymbol{P}(dx, dy) \right)^{1/p}.$$

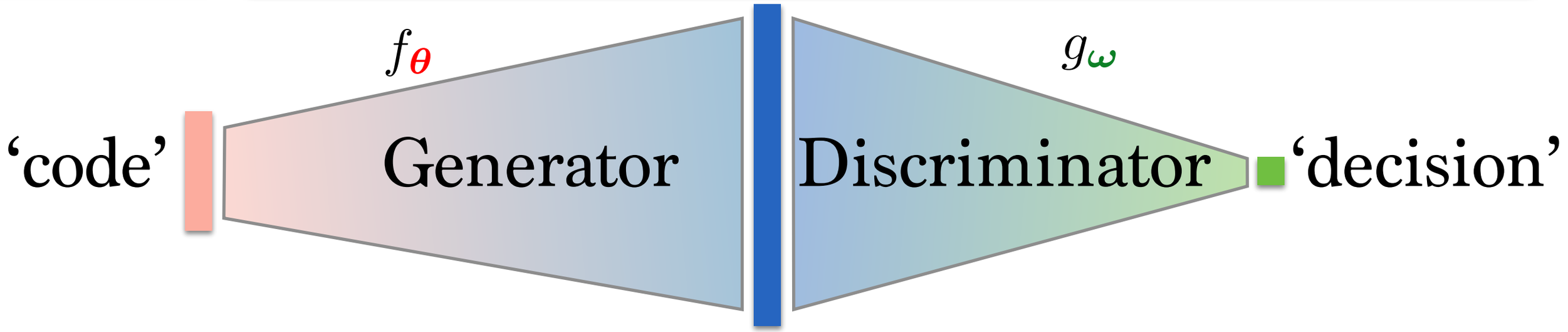
$\boldsymbol{\varphi}$1-Lipschitz $\Leftrightarrow \forall x, y, \|\boldsymbol{\varphi}(x) - \boldsymbol{\varphi}(y)\| \leq \boldsymbol{D}(x, y)$

$$W_1(\boldsymbol{\mu}, \boldsymbol{\nu}) = \sup_{\boldsymbol{\varphi}\ 1\text{-Lipschitz}} \int \boldsymbol{\varphi}(d\boldsymbol{\mu} - d\boldsymbol{\nu}).$$

W1

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$

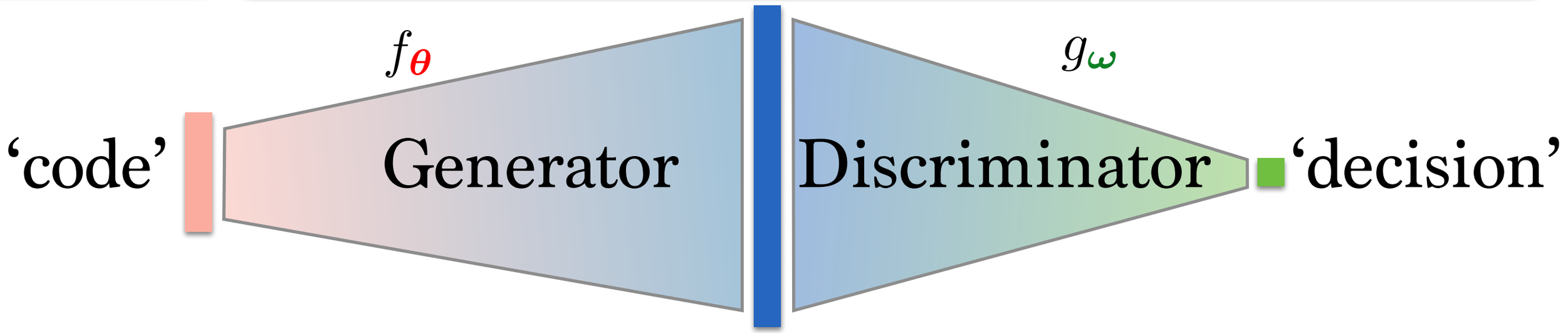


$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) - \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$

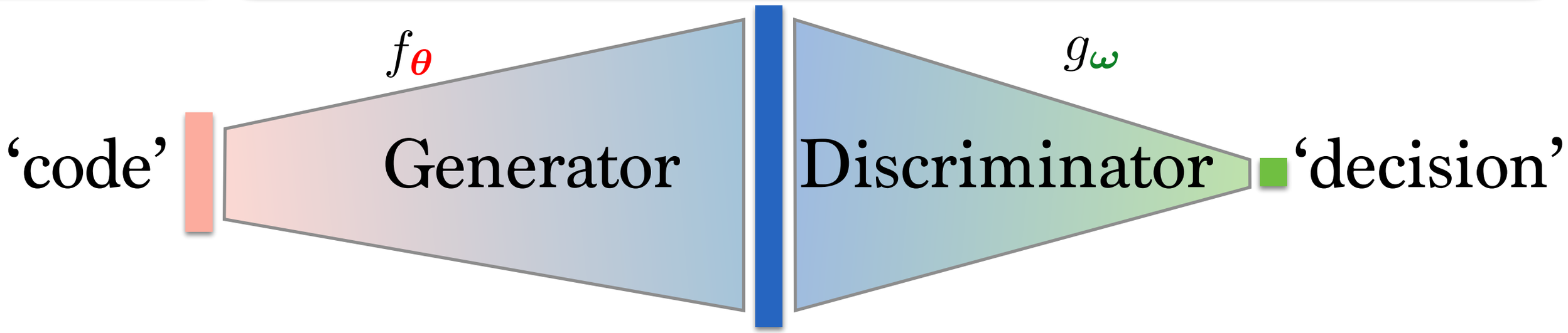


$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\textbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

$$W_1(\boldsymbol{\mu}, \boldsymbol{\nu}) = \sup_{\boldsymbol{\varphi}\ 1\text{-Lipschitz}} \int \boldsymbol{\varphi}(d\boldsymbol{\mu} - d\boldsymbol{\nu}).$$

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



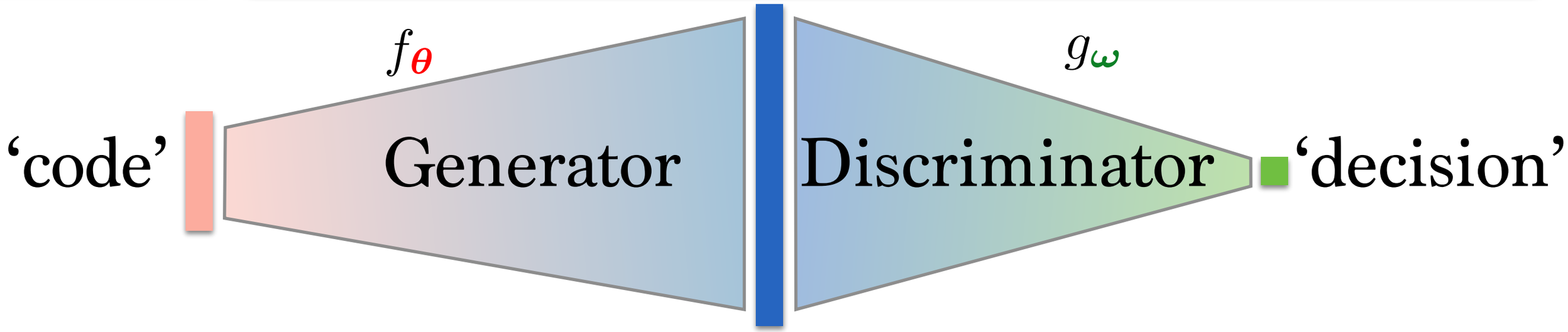
$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\textbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

$$W_1(\boldsymbol{\mu}, \boldsymbol{\nu}) = \sup_{\boldsymbol{\varphi}\ 1\text{-Lipschitz}} \int \boldsymbol{\varphi}(d\boldsymbol{\mu} - d\boldsymbol{\nu}).$$

W1

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



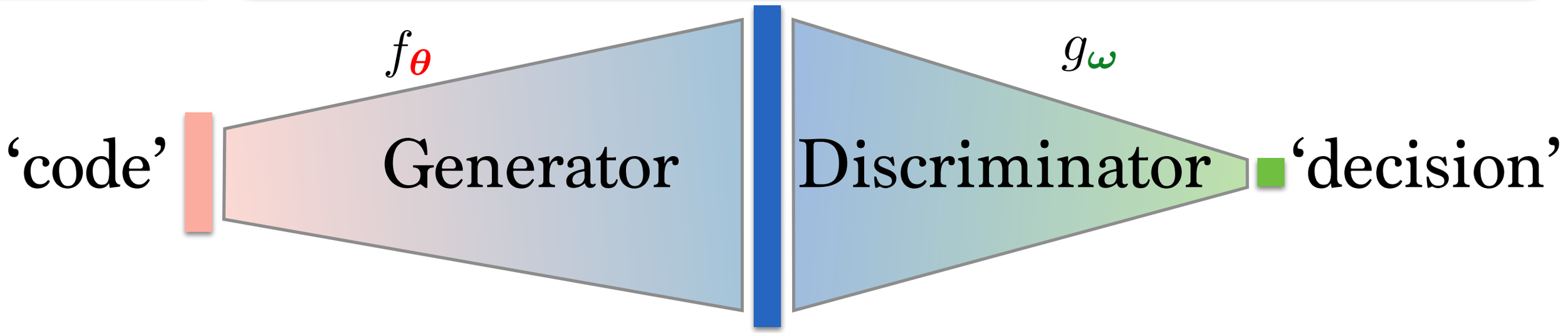
$$\min_{\boldsymbol{\theta}} \max_{\substack{\boldsymbol{\omega} \\ g_{\boldsymbol{\omega}}\ 1\text{-Lipschitz}}} \int_{\textbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

$$W_1(\boldsymbol{\mu}, \boldsymbol{\nu}) = \sup_{\boldsymbol{\varphi}\ 1\text{-Lipschitz}} \int \boldsymbol{\varphi}(d\boldsymbol{\mu} - d\boldsymbol{\nu}).$$

W1

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



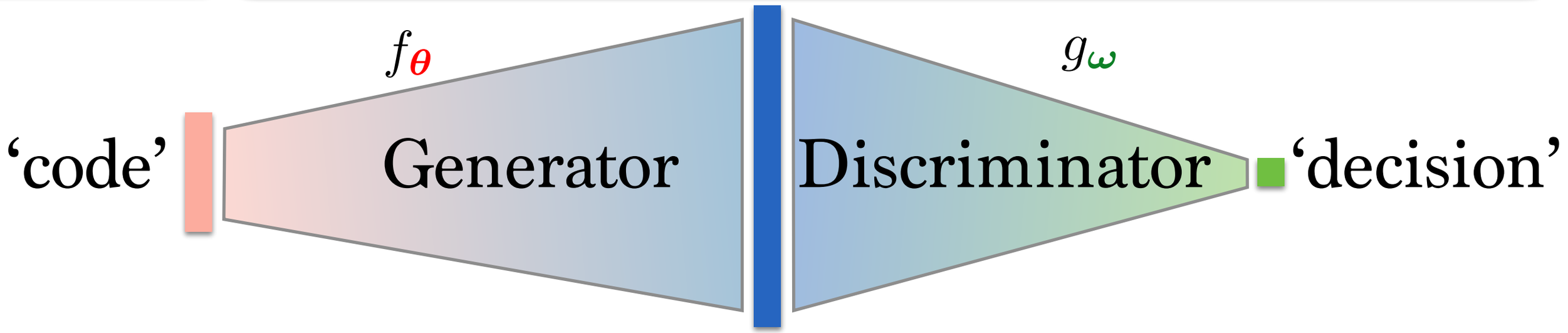
$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

$$\min_{\boldsymbol{\theta}} \textit{kind-of-}W_1(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, \boldsymbol{\nu}_{\text{data}})$$

$$W_1(\boldsymbol{\mu}, \boldsymbol{\nu}) = \sup \int_{\mathbf{data}} g_{\boldsymbol{\omega}}(d\boldsymbol{\mu} - d\boldsymbol{\nu})$$

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



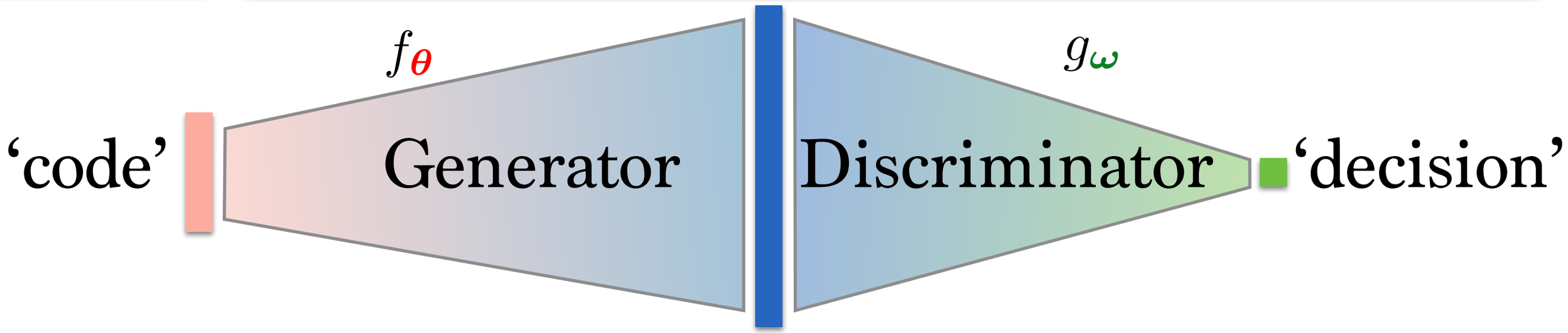
$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\textbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

$$\min_{\boldsymbol{\theta}} \textit{kind-of-}W_1(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, \boldsymbol{\nu}_{\text{data}})$$

$$W_1(\mu, \nu) = \sup_{\|g\|_L \le 1} \int g(d\mu - d\nu)$$

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{Z}\sim\boldsymbol{\mu}} \log g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{Z})) + \mathbb{E}_{\boldsymbol{Y}\sim\boldsymbol{\nu}_{\text{data}}} \log(1 - g_{\boldsymbol{\omega}}(\boldsymbol{Y}))$$



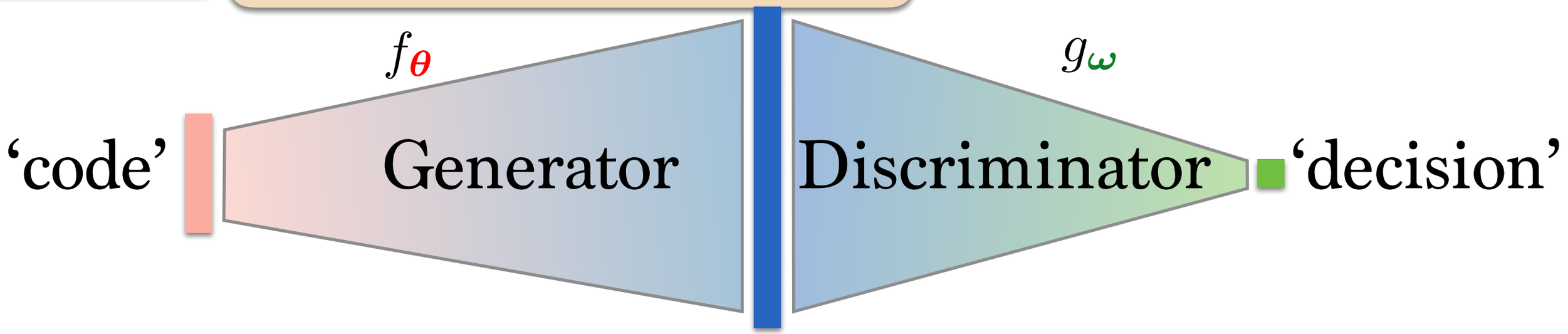
$$\min_{\boldsymbol{\theta}} \max_{\substack{\boldsymbol{\omega} \\ g_{\boldsymbol{\omega}} \text{ 1-Lipschitz}}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

$$\min_{\boldsymbol{\theta}} \textit{kind-of-}W_1(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu}, \boldsymbol{\nu}_{\text{data}})$$

$$W_1(\mu, \nu) = \sup_{\|f\|_{\text{Lip}} \leq 1} \int f(d\mu - d\nu)$$

Variants - Wasserstein GAN

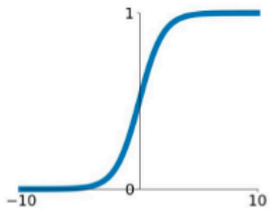
$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\mathrm{data}})$$



How to enforce 1-Lipschitz NN? consider activations

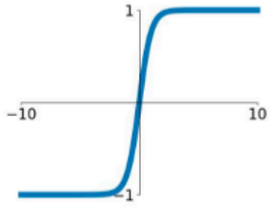
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



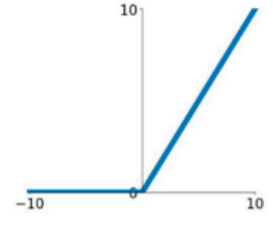
tanh

$\tanh(x)$



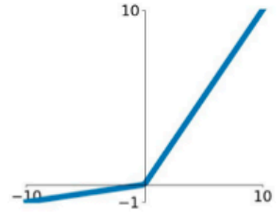
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

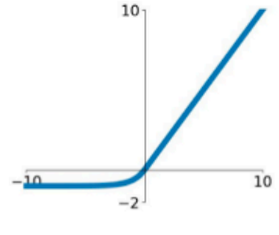


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

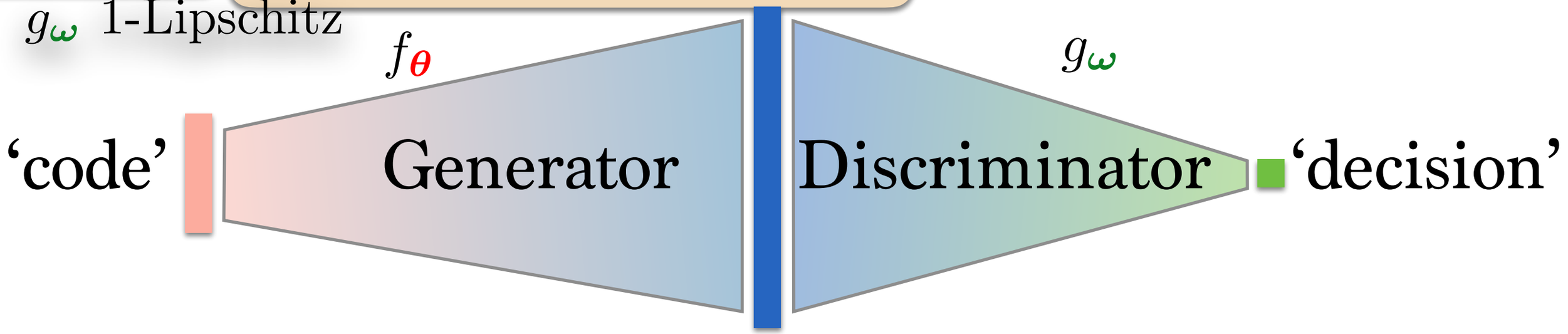


reference

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

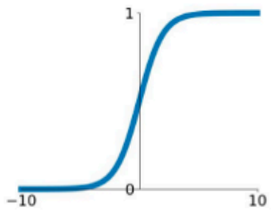
$g_{\boldsymbol{\omega}}$ 1-Lipschitz



How to enforce 1-Lipschitz NN? consider activations

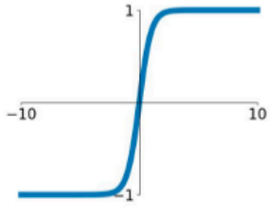
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



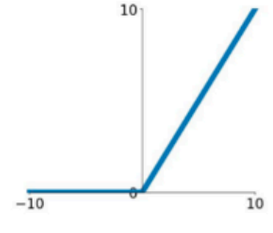
tanh

$\tanh(x)$



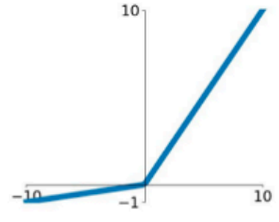
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

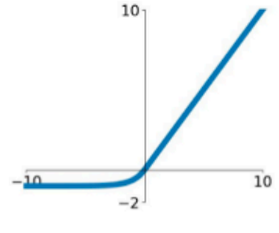


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

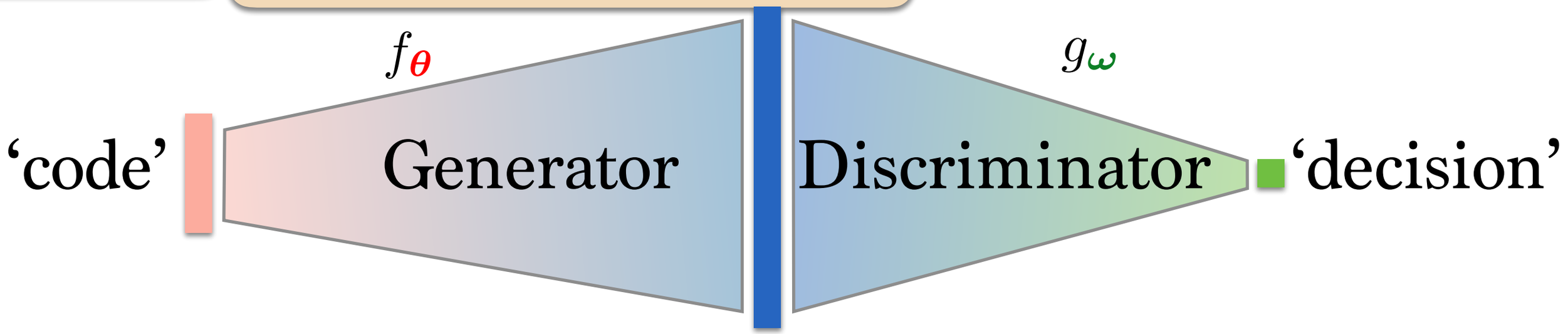
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



reference

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}} \, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\mathrm{data}})$$



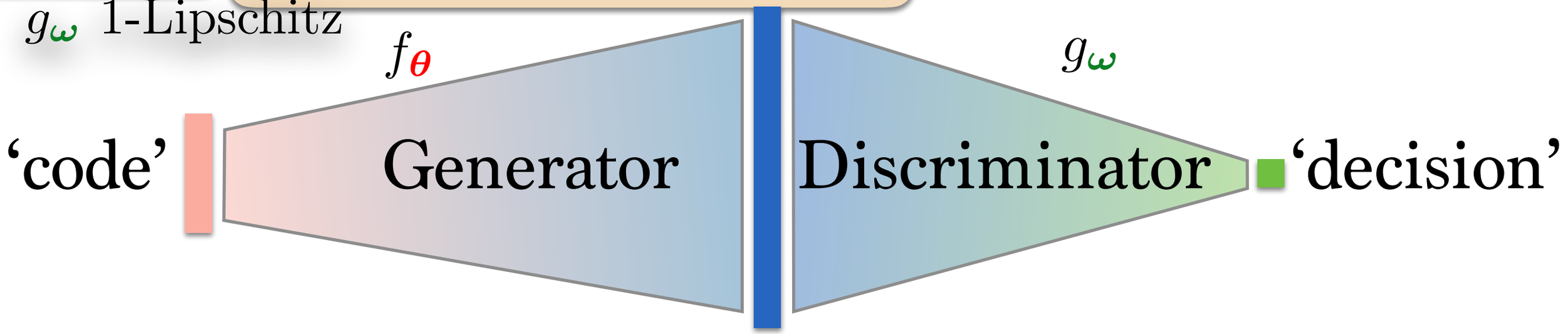
How to enforce 1-Lipschitz NN? consider activations

$$\mathrm{ReLU}(A\mathbf{x} + \mathbf{b}) \text{ is 1-Lipschitz if}$$
$$\text{all lines } A_i \text{ have 1-norm } \|A_i\|_1 \leq 1$$

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}} \, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

$g_{\boldsymbol{\omega}}$ 1-Lipschitz

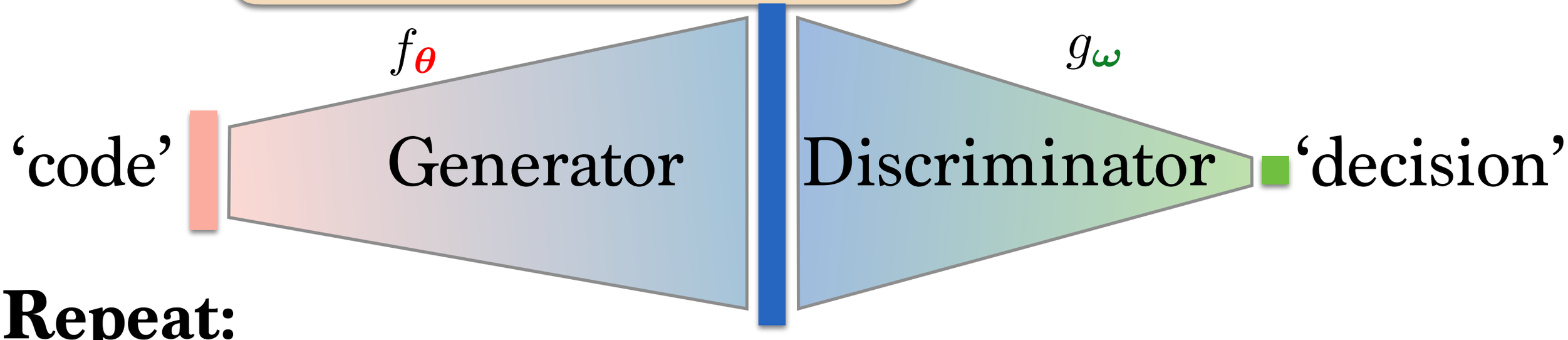


How to enforce 1-Lipschitz NN? consider activations

$$\text{ReLU}(A\mathbf{x} + \mathbf{b}) \text{ is 1-Lipschitz if}$$
$$\text{all lines } A_i \text{ have 1-norm } \|A_i\|_1 \leq 1$$

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$



Repeat:

1. Sample $z_1, z_2, \ldots, z_n \sim \mu$, $y_1, y_2, \ldots, y_n \sim \nu$

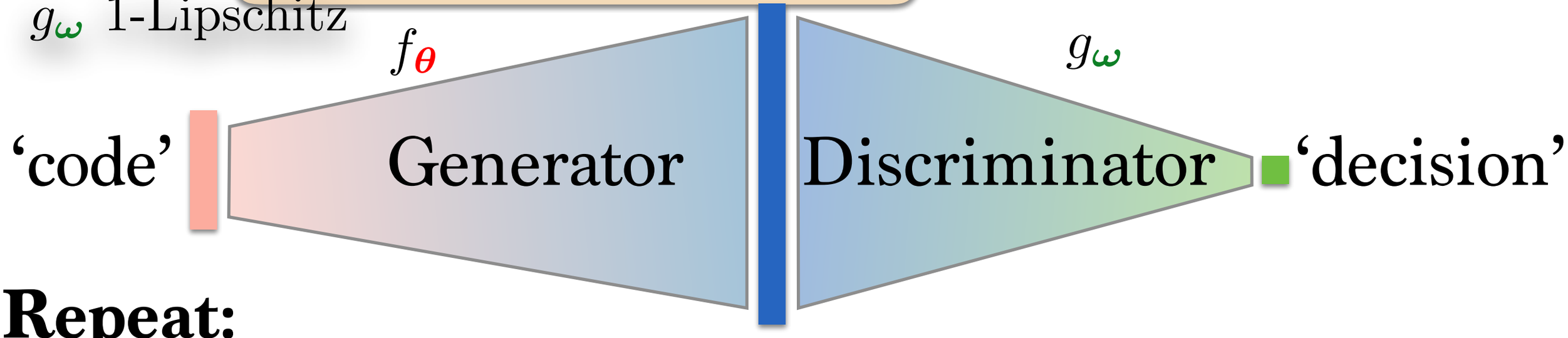
2. $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} \left[\sum_{i=1}^{n} g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{z_i})) \right]$$

3. $$\boldsymbol{\omega} \leftarrow \text{clip}\left(\boldsymbol{\omega} + \nabla_{\boldsymbol{\omega}} \left[\sum_{i=1}^{n} g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{z_i})) - \sum_{j=1}^{n} g_{\boldsymbol{\omega}}(\boldsymbol{y_j}) \right] \right)$$

Variants - Wasserstein GAN

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\omega}} \int_{\mathbf{data}} g_{\boldsymbol{\omega}}\, d(f_{\boldsymbol{\theta}\sharp}\boldsymbol{\mu} - \boldsymbol{\nu}_{\text{data}})$$

$g_{\boldsymbol{\omega}}$ 1-Lipschitz



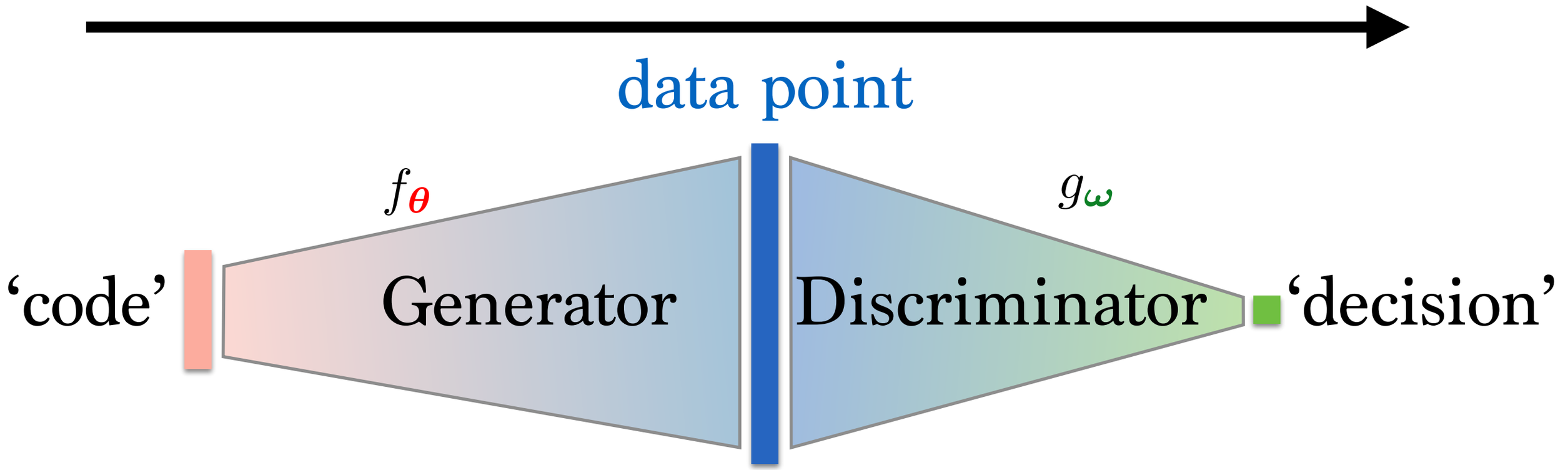
Repeat:

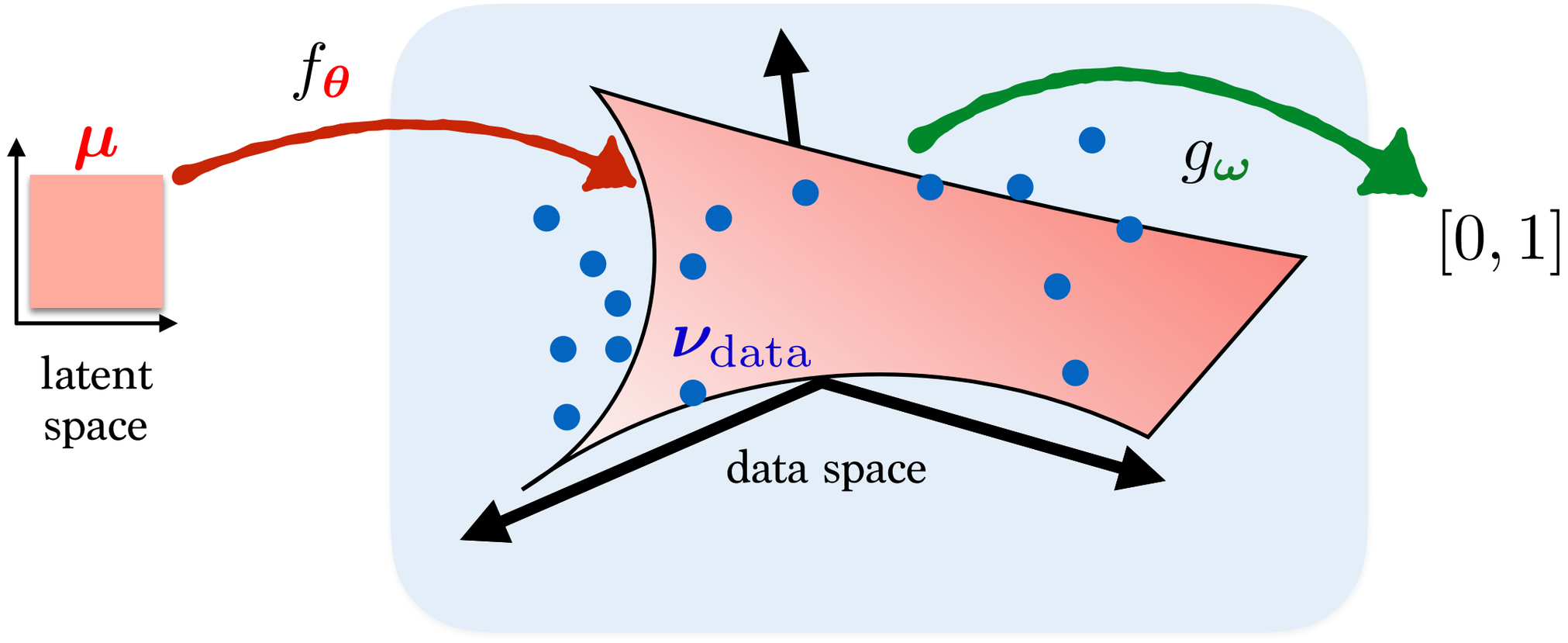
1. Sample $z_1, z_2, \ldots, z_n \sim \mu$, $y_1, y_2, \ldots, y_n \sim \nu$

2. $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} \left[\sum_{i=1}^{n} g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{z_i})) \right]$$

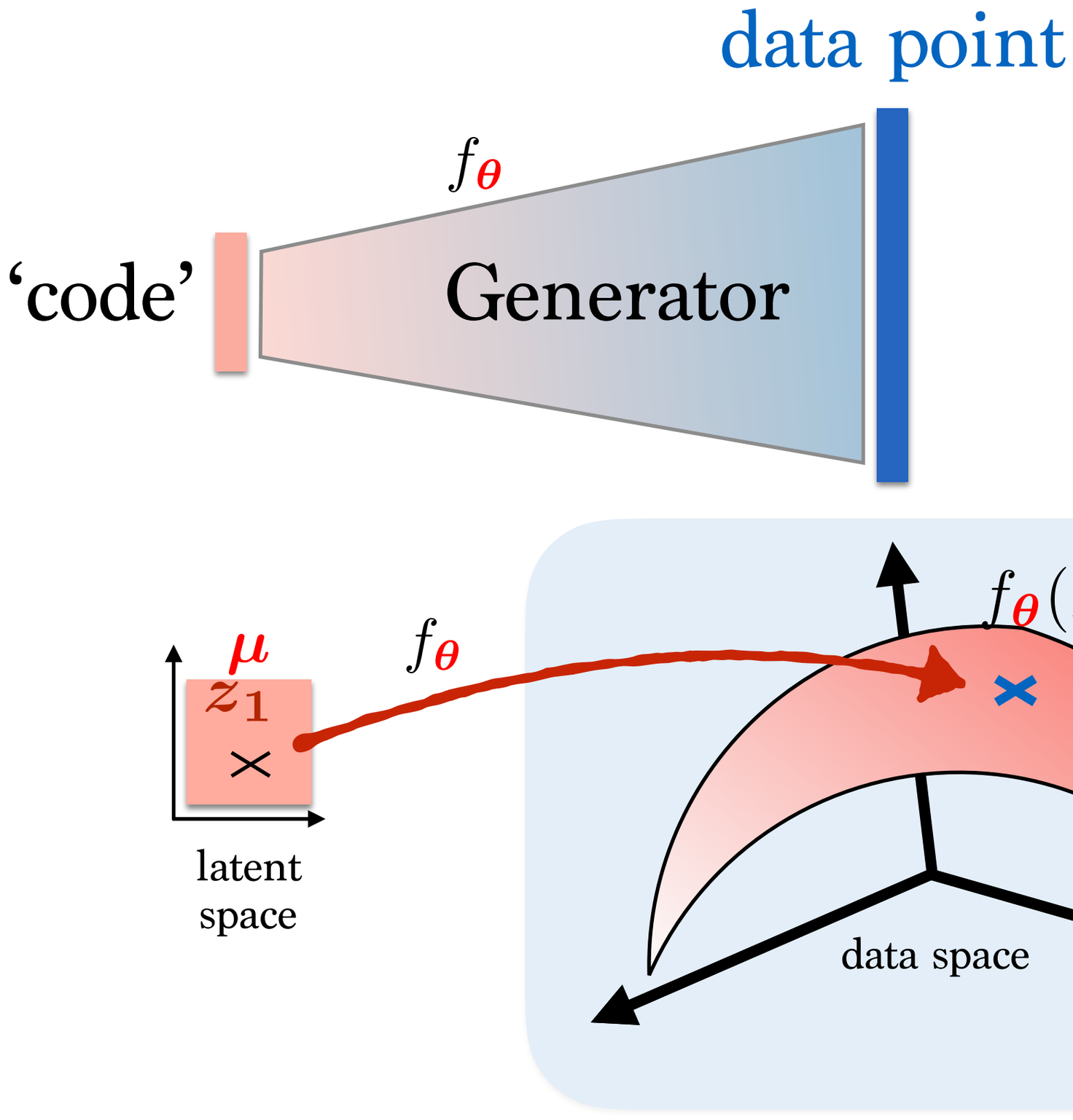
3. $$\boldsymbol{\omega} \leftarrow \text{clip}\left(\boldsymbol{\omega} + \nabla_{\boldsymbol{\omega}} \left[\sum_{i=1}^{n} g_{\boldsymbol{\omega}}(f_{\boldsymbol{\theta}}(\boldsymbol{z_i})) - \sum_{j=1}^{n} g_{\boldsymbol{\omega}}(\boldsymbol{y_j}) \right] \right)$$

GANs are trained, then used

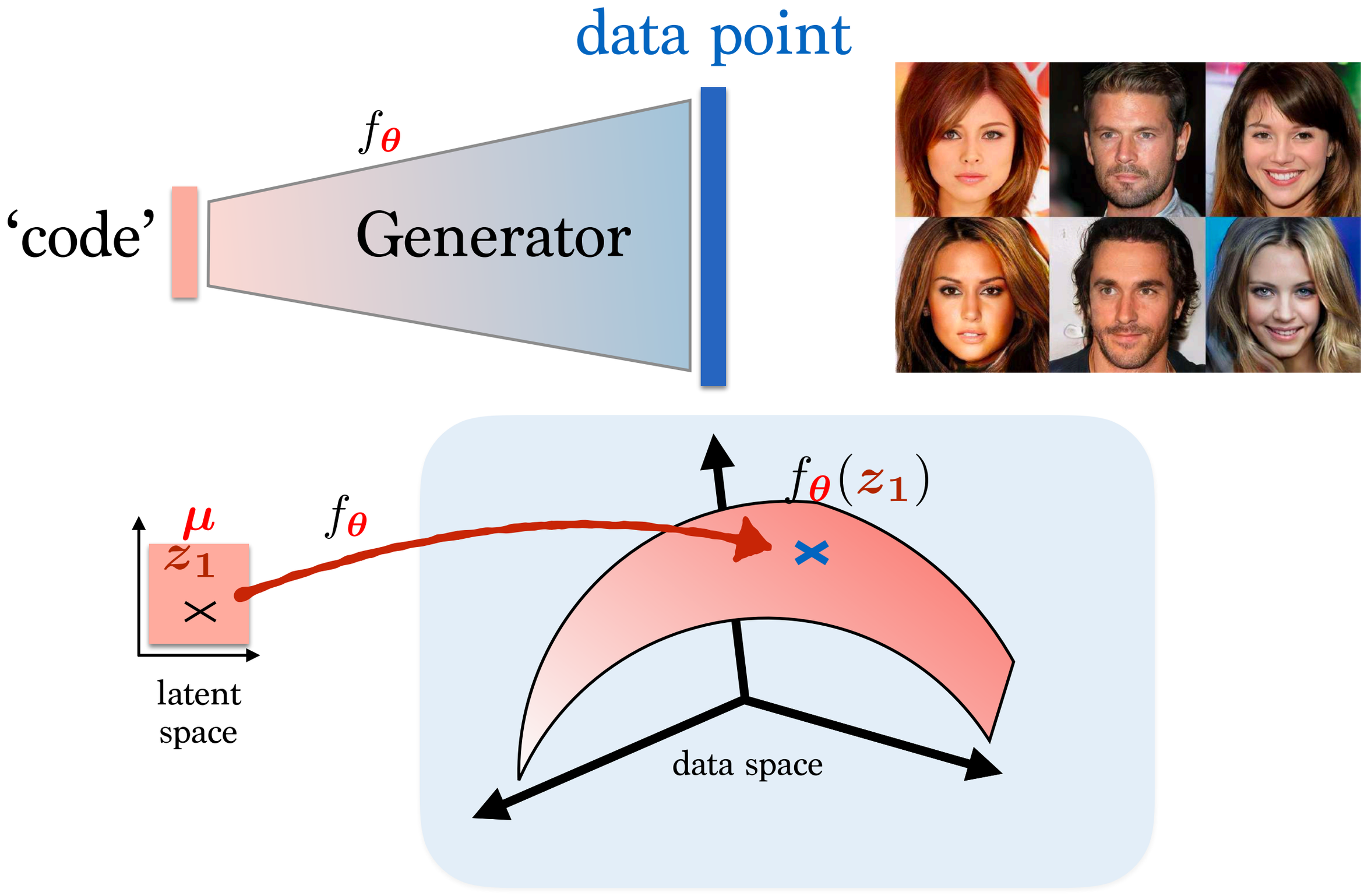




GANs are trained, then used

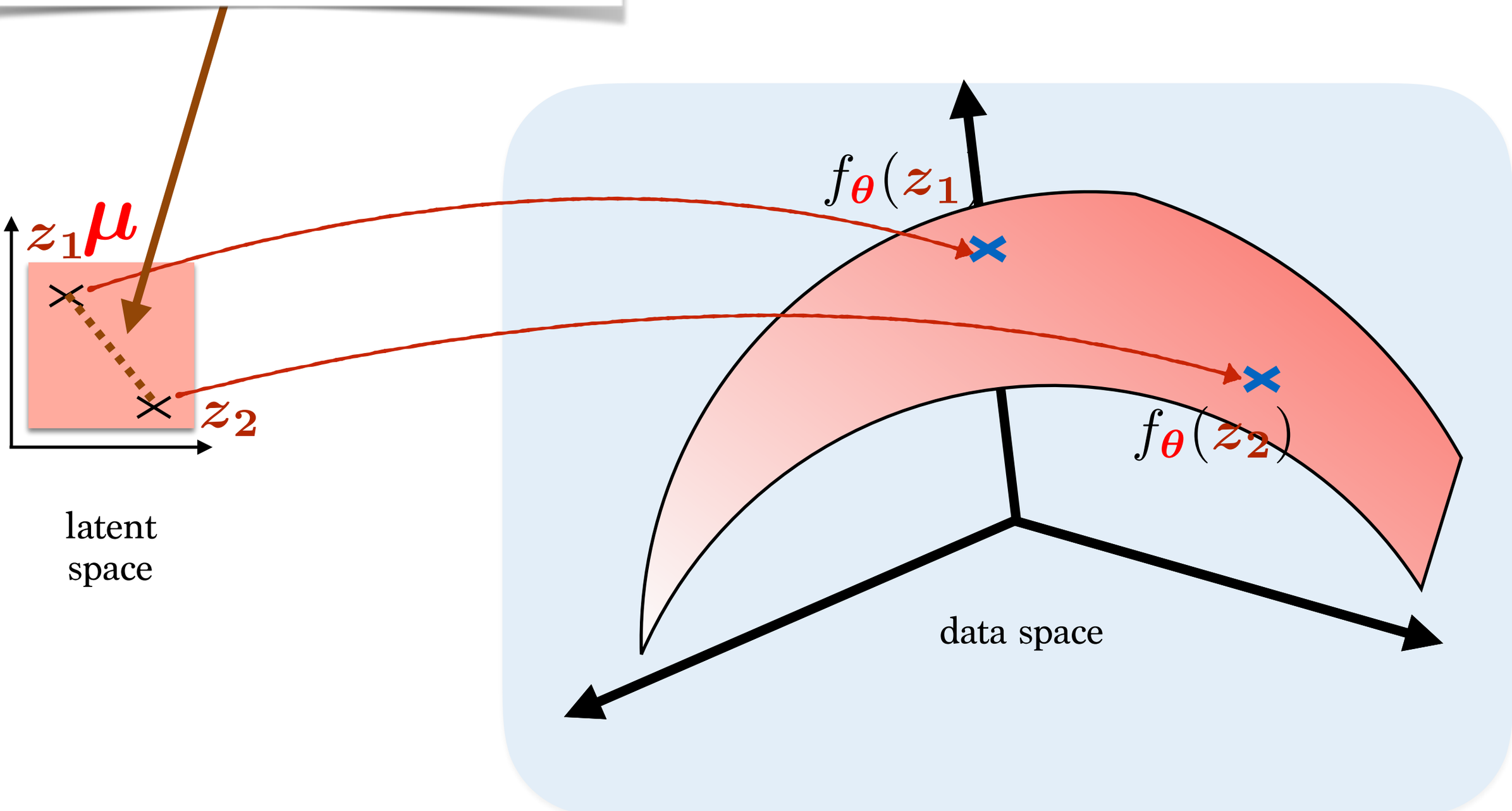


GANs are trained, then used



GANs are trained, then used

$$(1-\lambda)\boldsymbol{z_1} + \lambda \boldsymbol{z_2}, \lambda \in [0,1]\}$$



GANs are trained, then used

$(1-\lambda)\boldsymbol{z_1} + \lambda \boldsymbol{z_2}, \lambda \in [0,1]\}$

$\{f_{\boldsymbol{\theta}}\left((1-\lambda)\boldsymbol{z_1} + \lambda \boldsymbol{z_2}\right), \lambda \in [0,1]\}$

$\boldsymbol{\mu}$

$\boldsymbol{z_1}$

$\boldsymbol{z_2}$

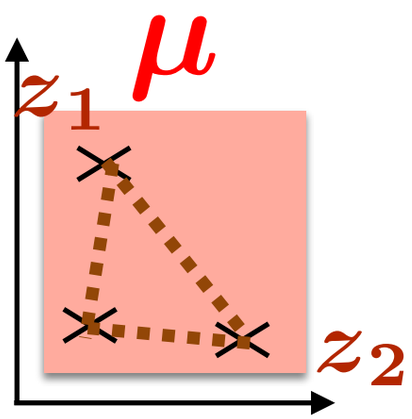
latent space

$f_{\boldsymbol{\theta}}(\boldsymbol{z_1})$

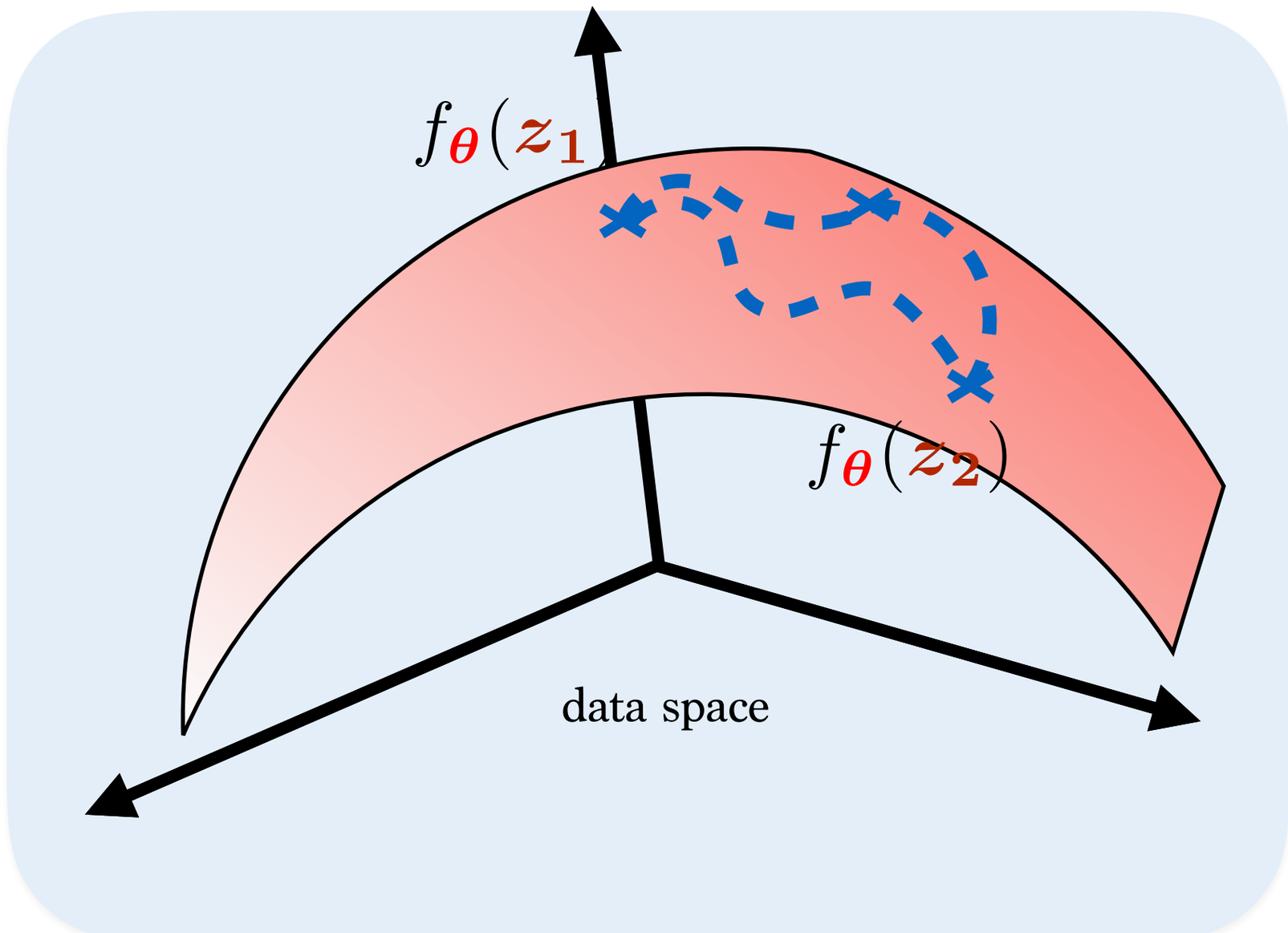
$f_{\boldsymbol{\theta}}(\boldsymbol{z_2})$

data space

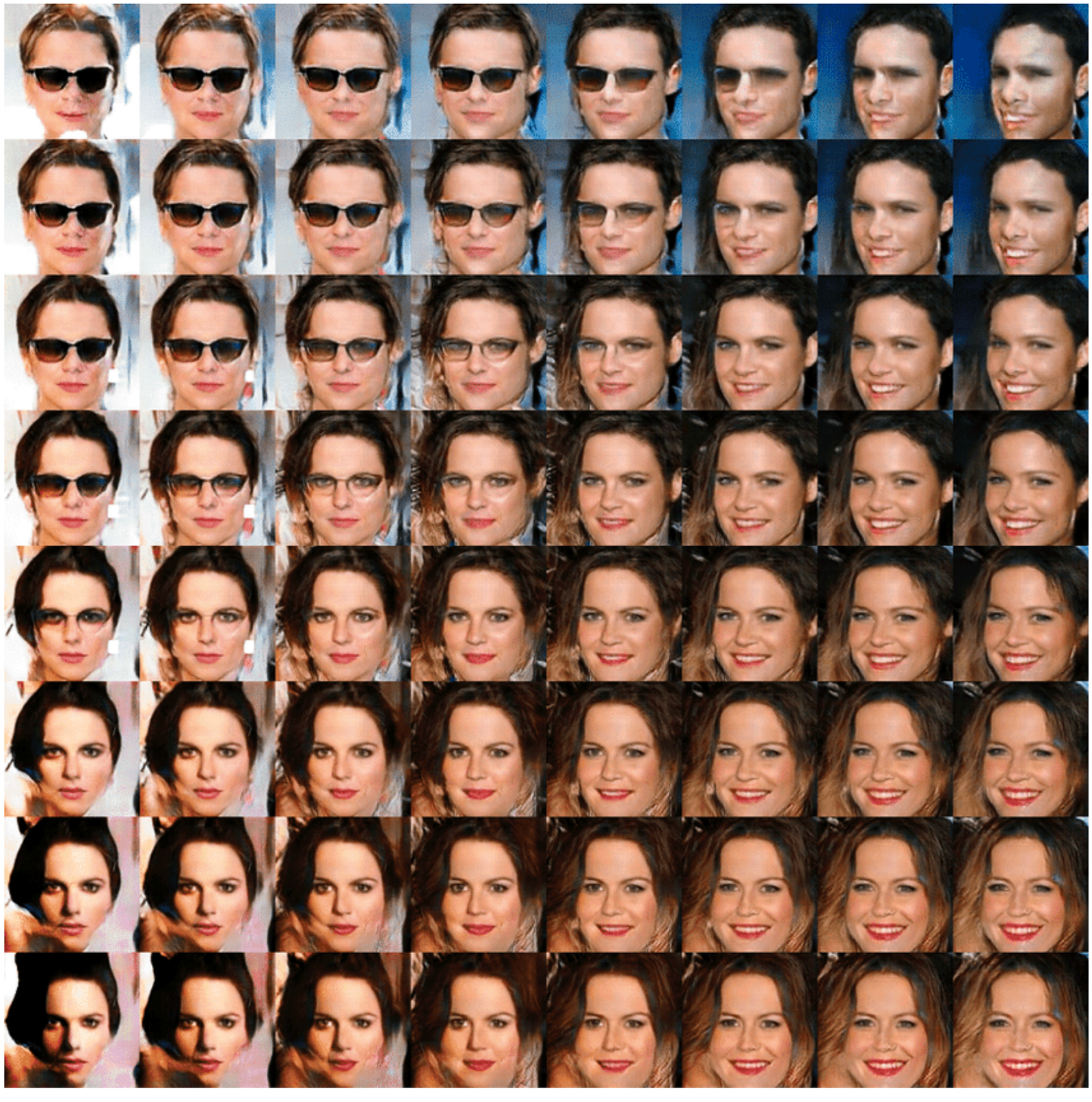
GANs are trained, then used



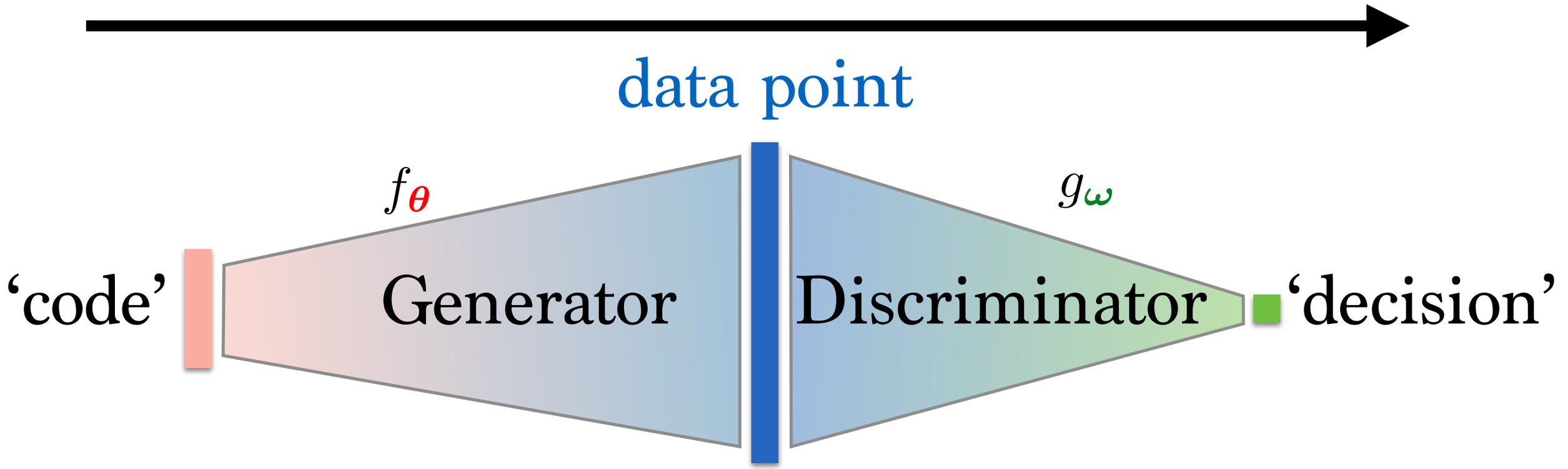
latent
space



GANs are trained, then used

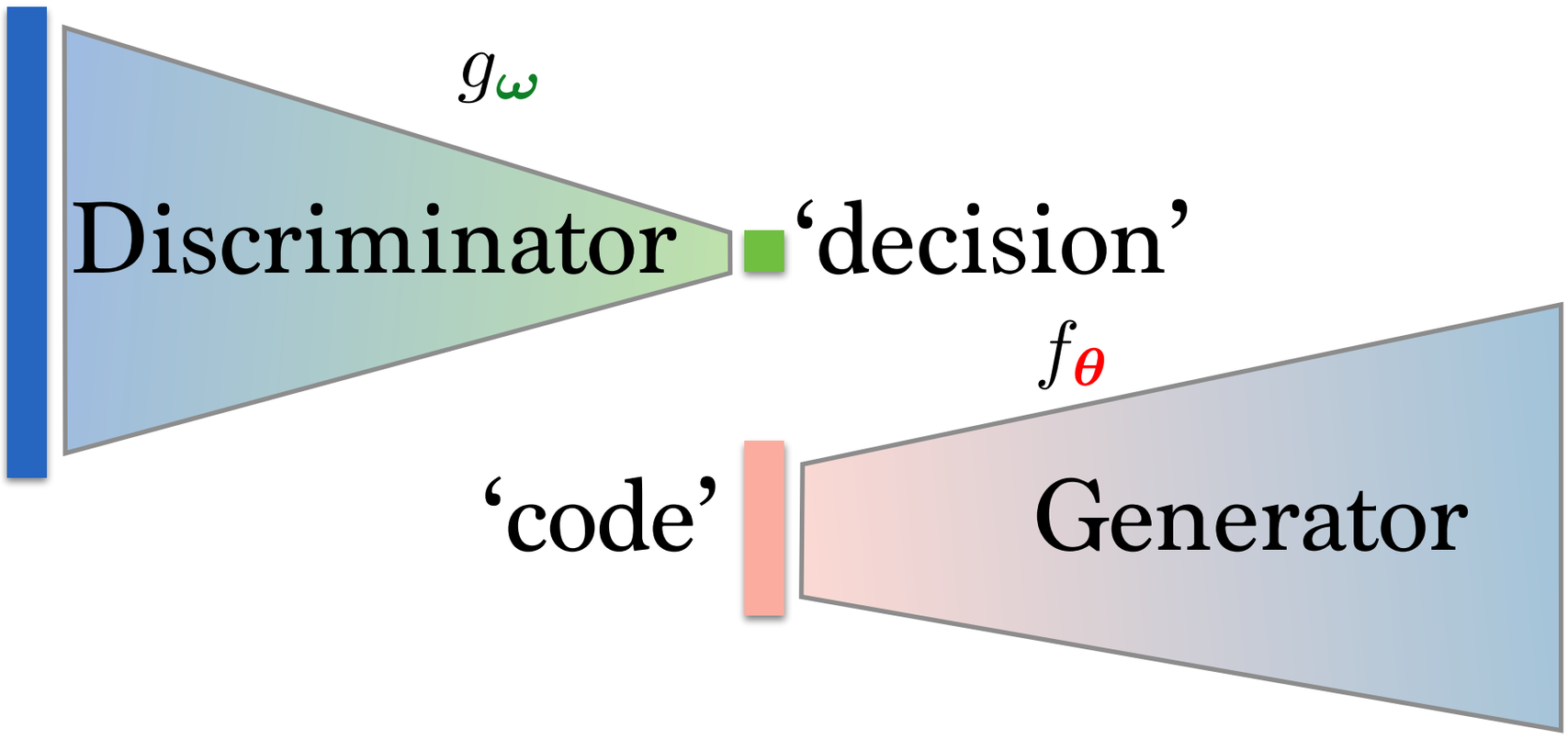


Auto-encoders

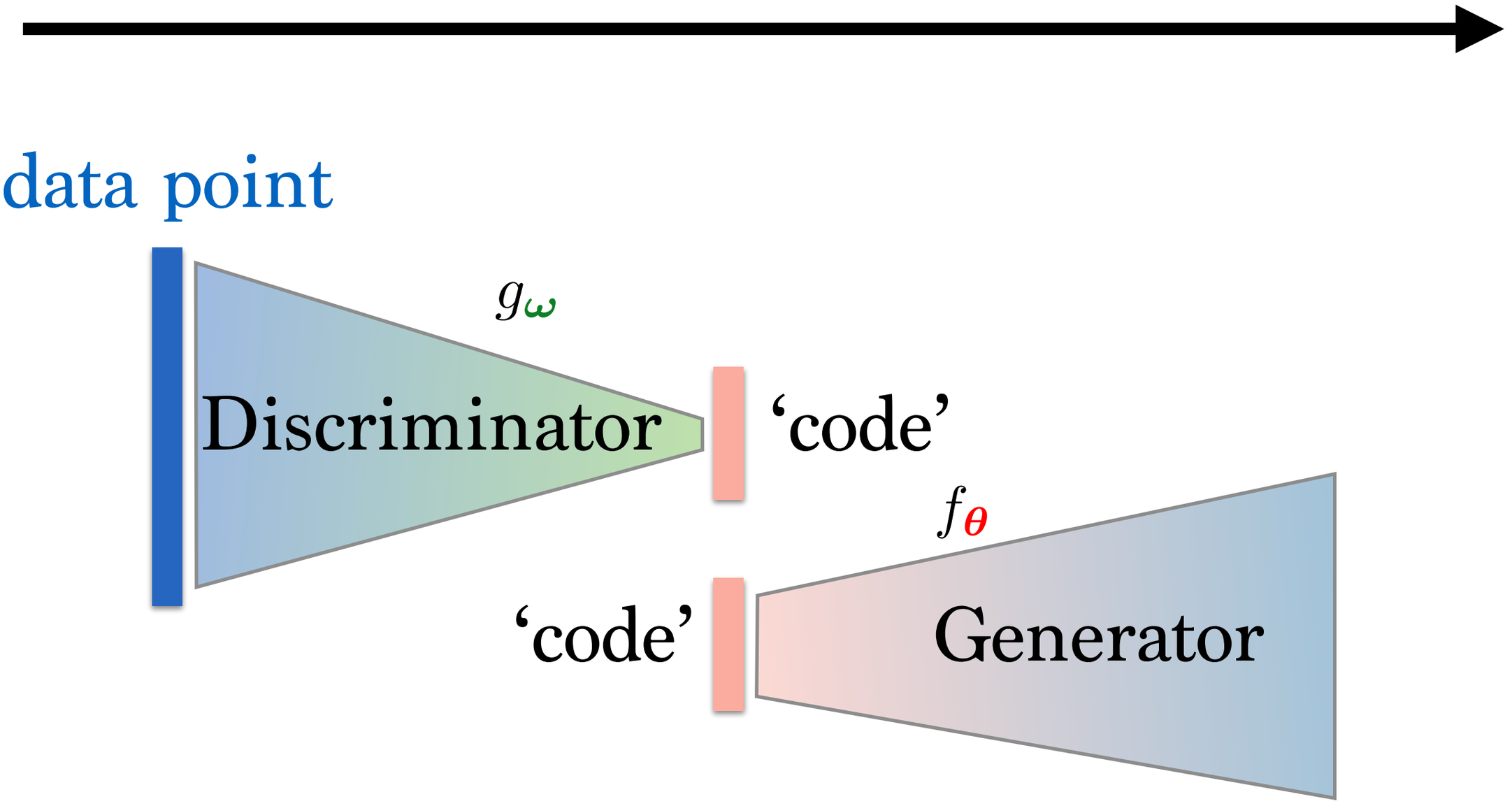


Auto-encoders

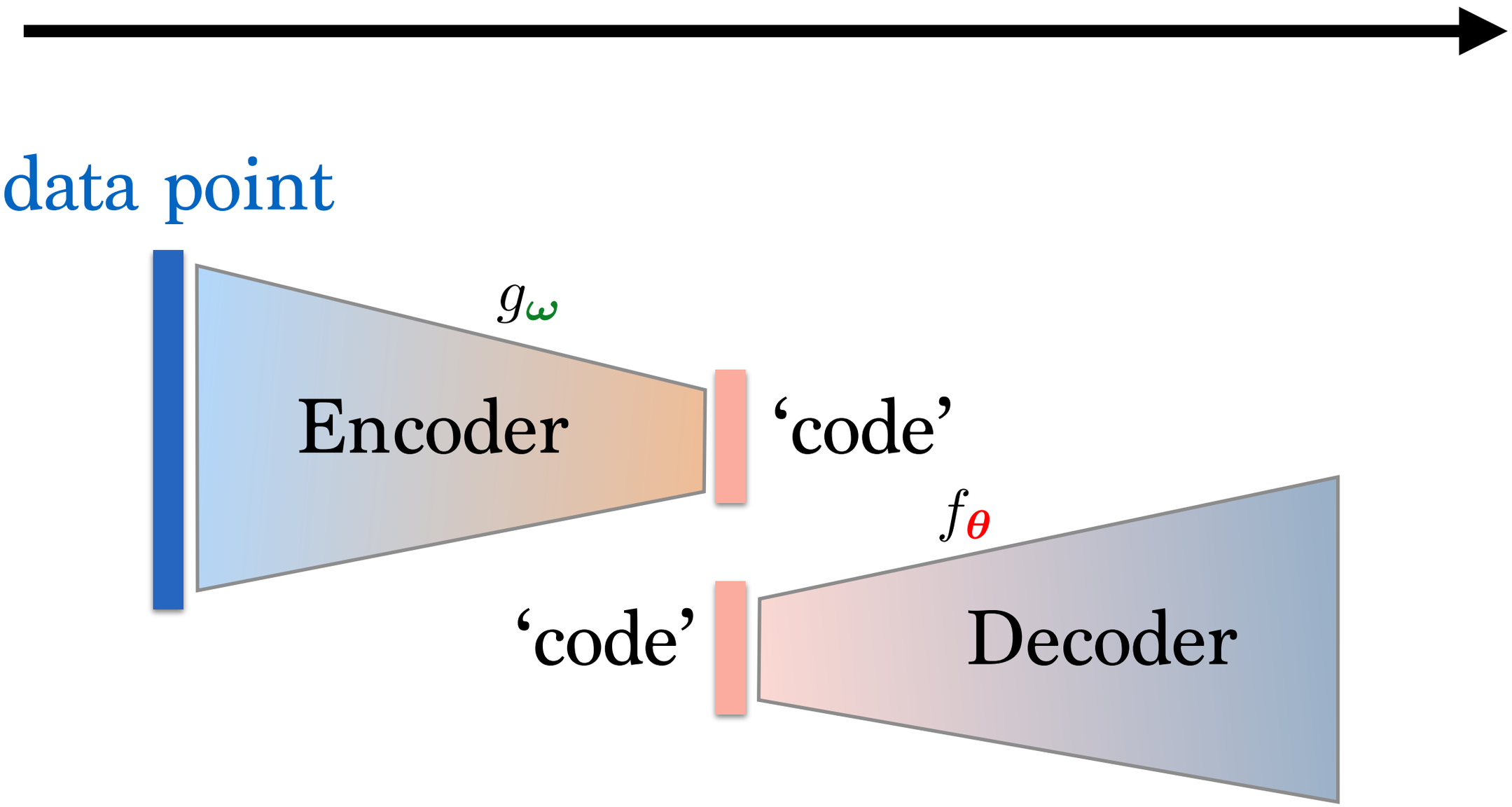
data point



Auto-encoders



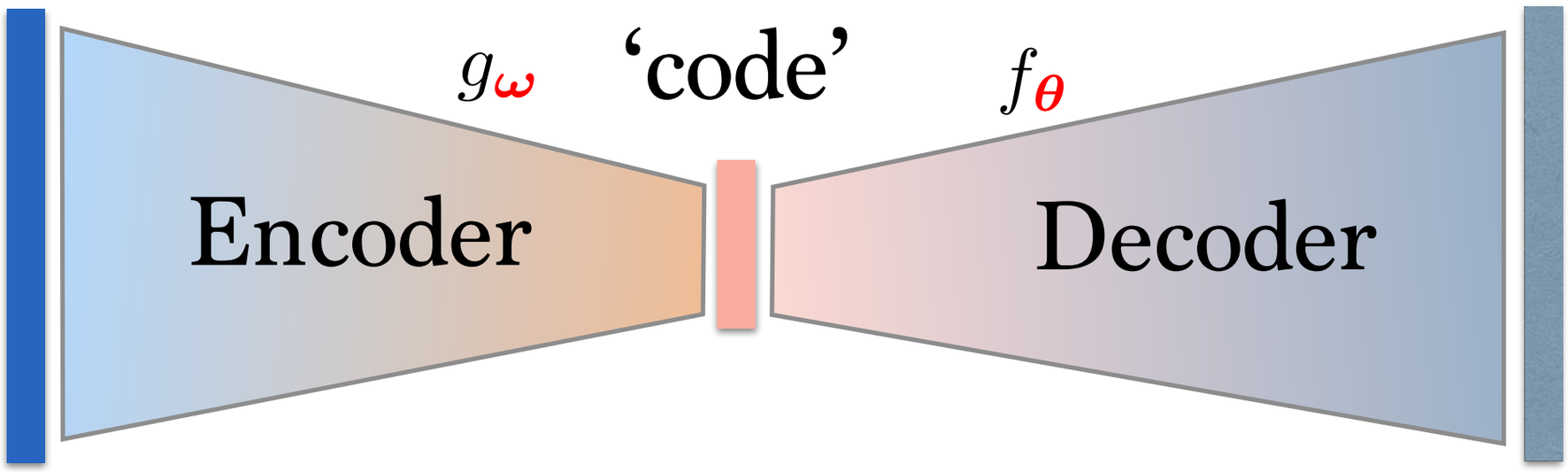
Auto-encoders



Auto-encoders

data point

‘reconstruction’



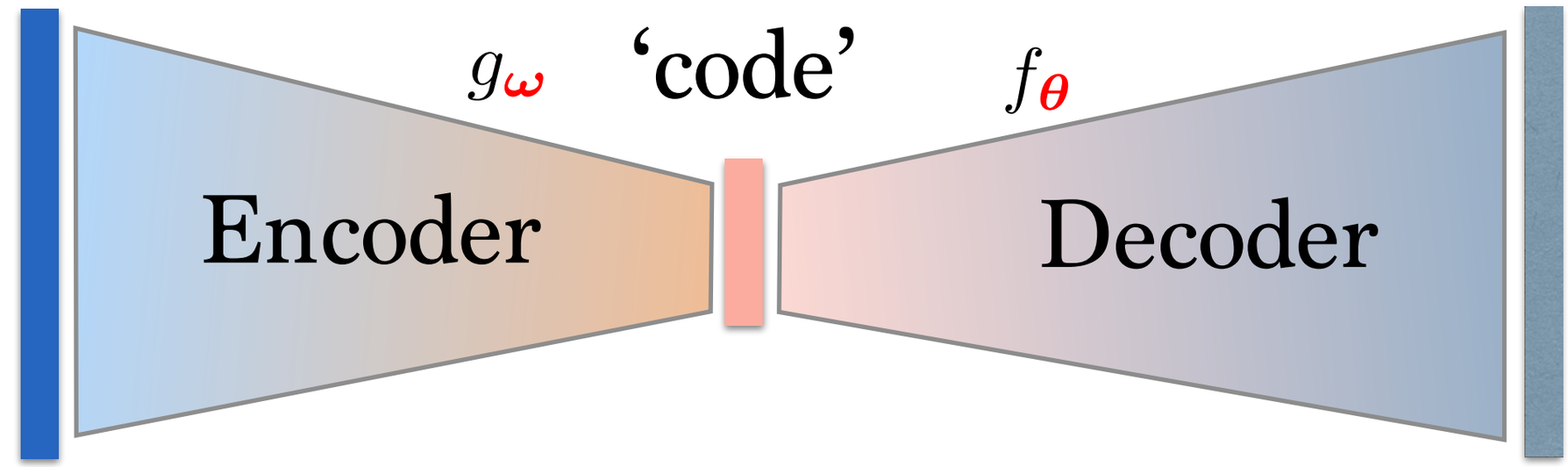
y

$f_{\theta} \circ g_{\omega}(y)$

Auto-encoders

data point

‘reconstruction’



$\boldsymbol{y}$

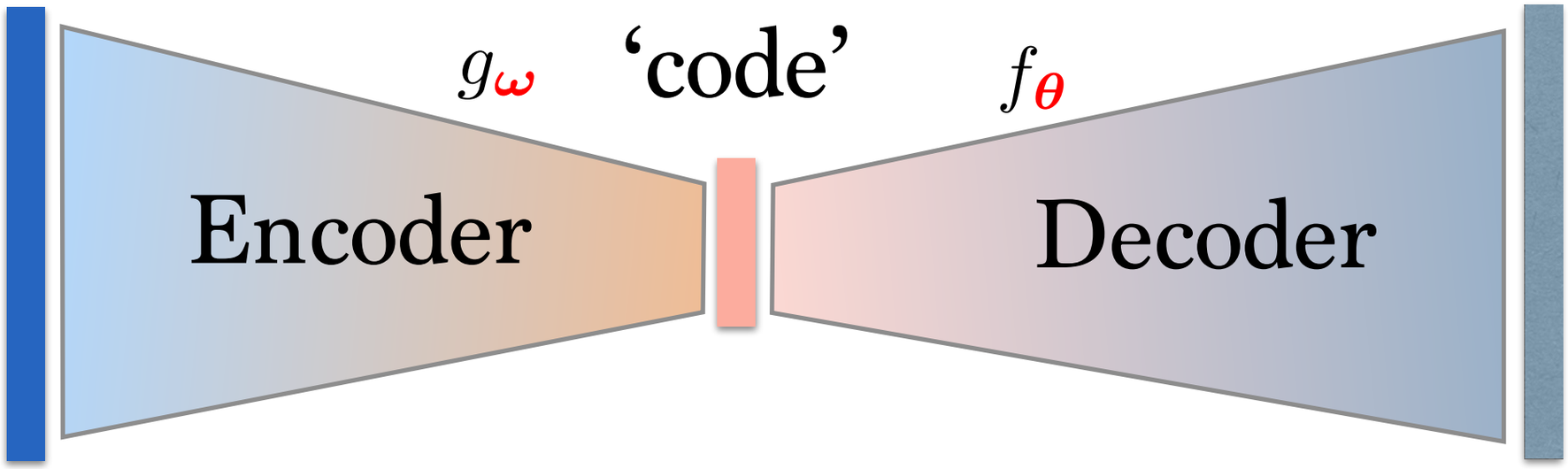
$f_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\omega}}(\boldsymbol{y})$

Goal : $\min_{\boldsymbol{\omega},\boldsymbol{\theta}} \Delta\left(\boldsymbol{y}, f_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\omega}}(\boldsymbol{y})\right)$

Auto-encoders

data point

‘reconstruction’



$\boldsymbol{y}$

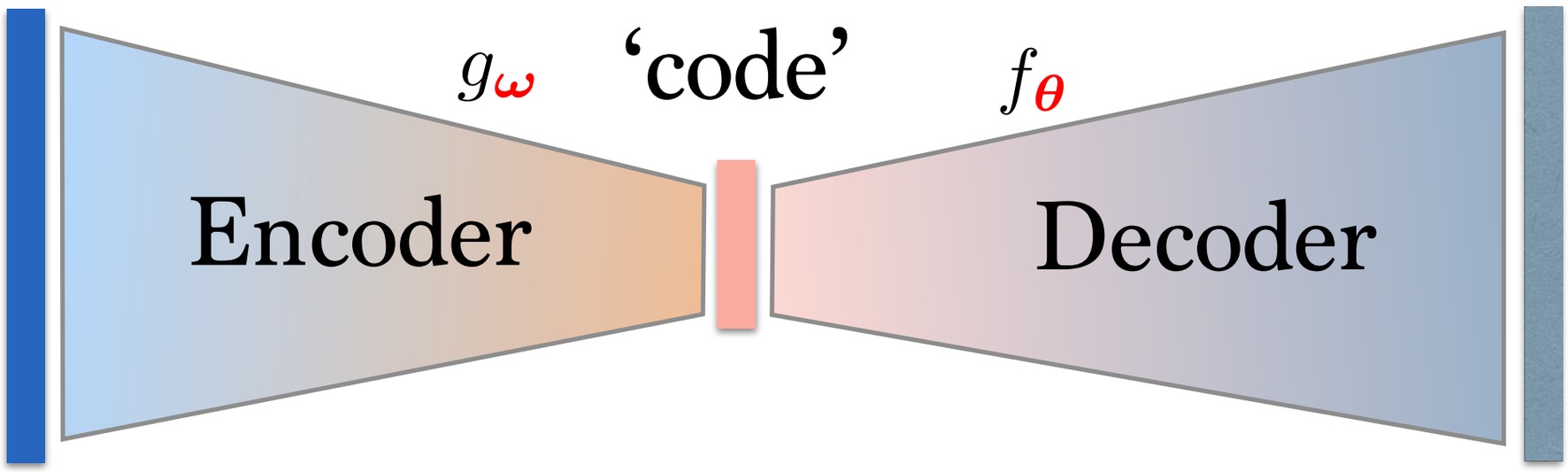
$f_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\omega}}(\boldsymbol{y})$

Goal : $\min_{\boldsymbol{\omega},\boldsymbol{\theta}} \sum_{\boldsymbol{i}} \Delta\left(\boldsymbol{y_i}, f_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\omega}}(\boldsymbol{y_i})\right)$

Auto-encoders

data point

‘reconstruction’



$\boldsymbol{y}$

$f_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\omega}}(\boldsymbol{y})$

$$\text{Goal : } \min_{\boldsymbol{\omega},\boldsymbol{\theta}} \sum_i \Delta\left(\boldsymbol{y_i}, f_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\omega}}(\boldsymbol{y_i})\right)$$

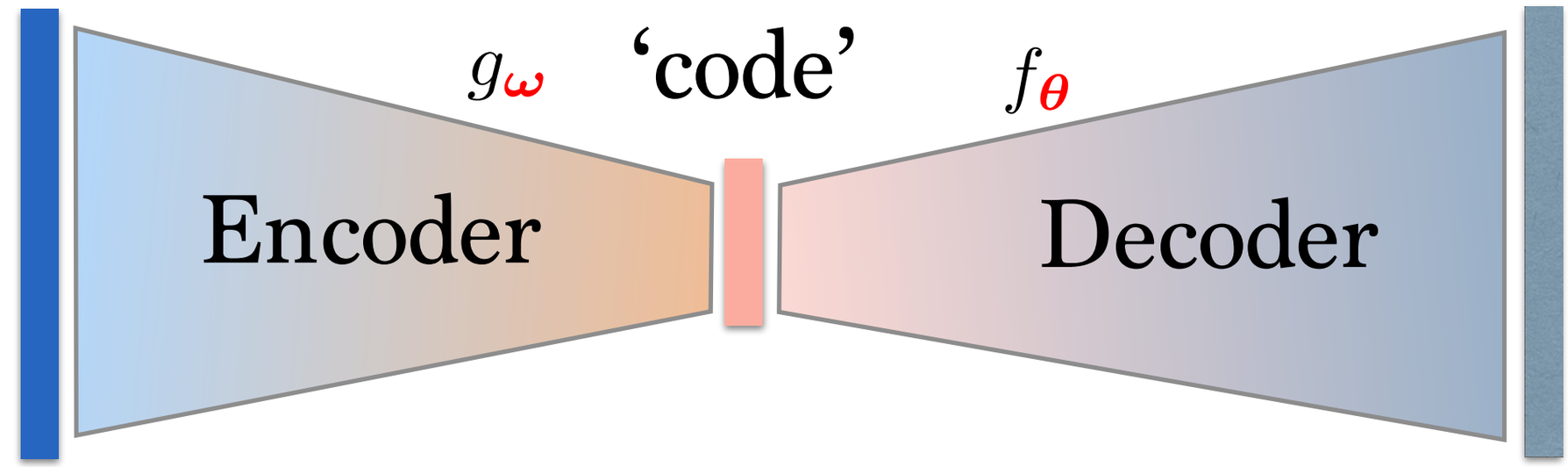
$$\text{Consider } f_{\boldsymbol{\theta}}(\boldsymbol{z}) = \boldsymbol{\theta z}, g_{\boldsymbol{\omega}}(\boldsymbol{y}) = \boldsymbol{\omega y}$$

$$\boldsymbol{\theta} \in \mathbb{R}^{p \times k} \text{ and } \boldsymbol{\omega} \in \mathbb{R}^{k \times p} \; \Delta = \|\cdot - \cdot\|^2$$

Auto-encoders

data point

‘reconstruction’



$\boldsymbol{y}$

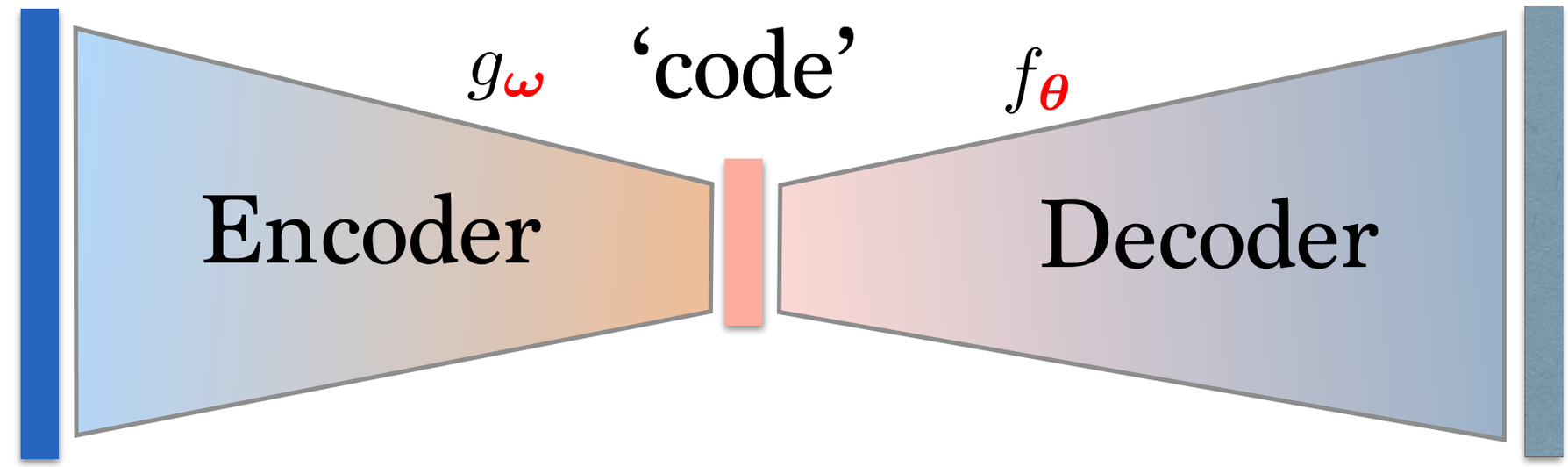
$f_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\omega}}(\boldsymbol{y})$

$$\min_{\substack{\boldsymbol{\theta} \in \mathbb{R}^{p \times k} \\ \boldsymbol{\omega} \in \mathbb{R}^{k \times p}}} \sum_{i} \|\boldsymbol{y}_i - \boldsymbol{\theta}\boldsymbol{\omega}\boldsymbol{y}_i\|^2$$

Auto-encoders

data point

‘reconstruction’



$\boldsymbol{y}$

$f_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\omega}}(\boldsymbol{y})$

$$\min_{\substack{\boldsymbol{\theta} \in \mathbb{R}^{p \times k} \\ \boldsymbol{\omega} \in \mathbb{R}^{k \times p}}} \sum_{i} \|\boldsymbol{y}_i - \boldsymbol{\theta}\boldsymbol{\omega}\boldsymbol{y}_i\|^2$$

Solution: PCA (after centering data)!