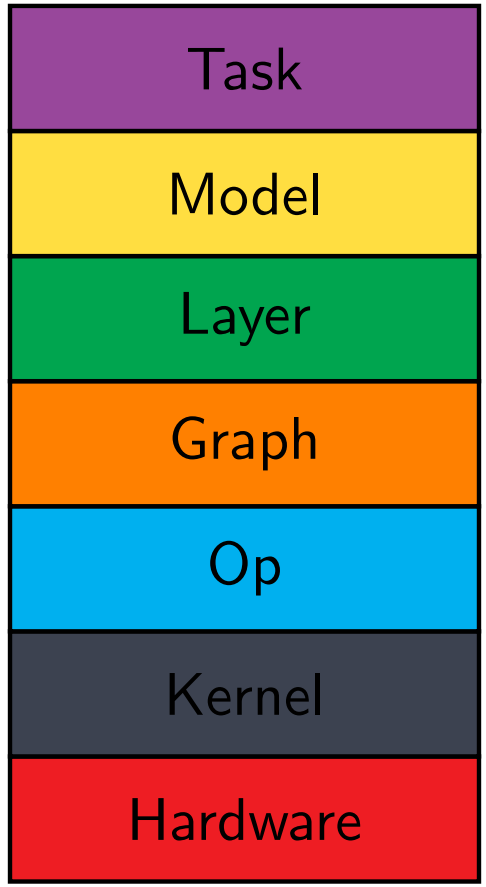
```
>>> c++ deep-learning.cpp

>>> ./a.out data.csv

>>> 42
```

Strategy



OSI Model of Machine Learning

Task Layer

$f\left(\text{  }\right) = \texttt{Teapot}$

Classification*

$f(\text{文内容}) =$ 

Translation*

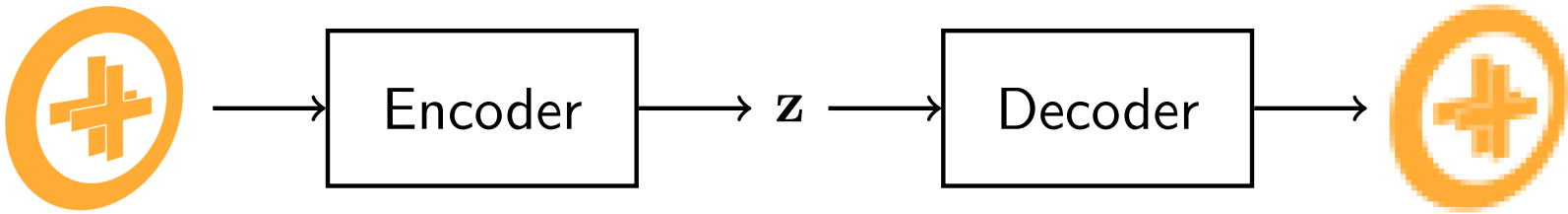
$f\left(\text{  }\right) =$ 

Generative†

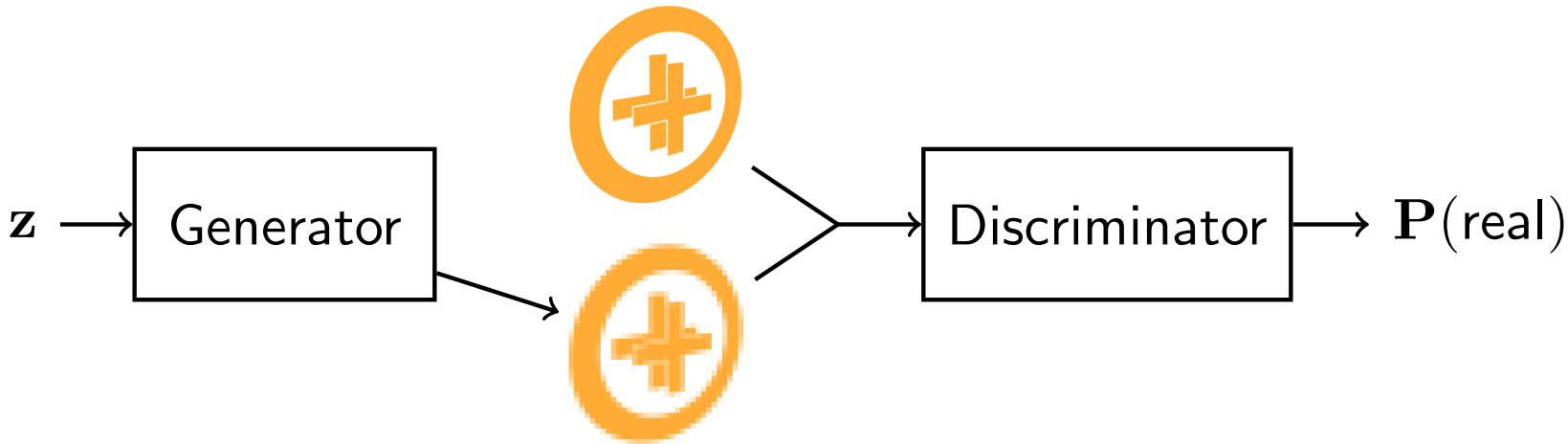
$\star$ Supervised

$\dagger$ Unsupervised

Model Layer



Autoencoder



Generative Adversarial Networks (GAN)

Generative Adversarial Networks

$$\underbrace{\mathbf{z} \sim P_{\text{noise}}(\mathbf{z})}_{\text{Noise}} \quad \underbrace{\mathbf{x} \sim P_{\text{data}}(\mathbf{x})}_{\text{Real Images}}$$

$$\underbrace{G(\mathbf{z})}_{\text{Fake Images}} \quad \underbrace{D(\mathbf{x})}_{\text{Probability of Being Real}}$$

Generative Adversarial Networks

$$\underbrace{\mathbf{z} \sim P_{\text{noise}}(\mathbf{z})}_{\text{Noise}} \quad \underbrace{\mathbf{x} \sim P_{\text{data}}(\mathbf{x})}_{\text{Real Images}}$$

$$\underbrace{G(\mathbf{z})}_{\text{Fake Images}} \quad \underbrace{D(\mathbf{x})}_{\text{Probability of Being Real}}$$

$$\frac{\int_{-\infty}^{\infty} \hat{f}(\xi)\, e^{2\pi i \xi x}\, d\xi + \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} + \begin{Bmatrix} e^2 \\ \pi \end{Bmatrix}}{\frac{1}{\sqrt{2\pi\hbar}} \int_{\text{all space}} e^{-i\mathbf{p}\cdot r/\hbar} \Psi(\mathbf{r}, s_z, t) d^3\mathbf{r} - \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}}$$

Generative Adversarial Networks

$$\underbrace{\mathbf{z} \sim P_{\text{noise}}(\mathbf{z})}_{\text{Noise}} \quad \underbrace{\mathbf{x} \sim P_{\text{data}}(\mathbf{x})}_{\text{Real Images}}$$

$$\underbrace{G(\mathbf{z})}_{\text{Fake Images}} \quad \underbrace{D(\mathbf{x})}_{\text{Probability of Being Real}}$$

$$\min_G \max_D \log(D(\mathbf{x})) + \log(1 - D(G(\mathbf{z})))$$

Generative Adversarial Networks

BEGAN: Berthelot et al. (2017)

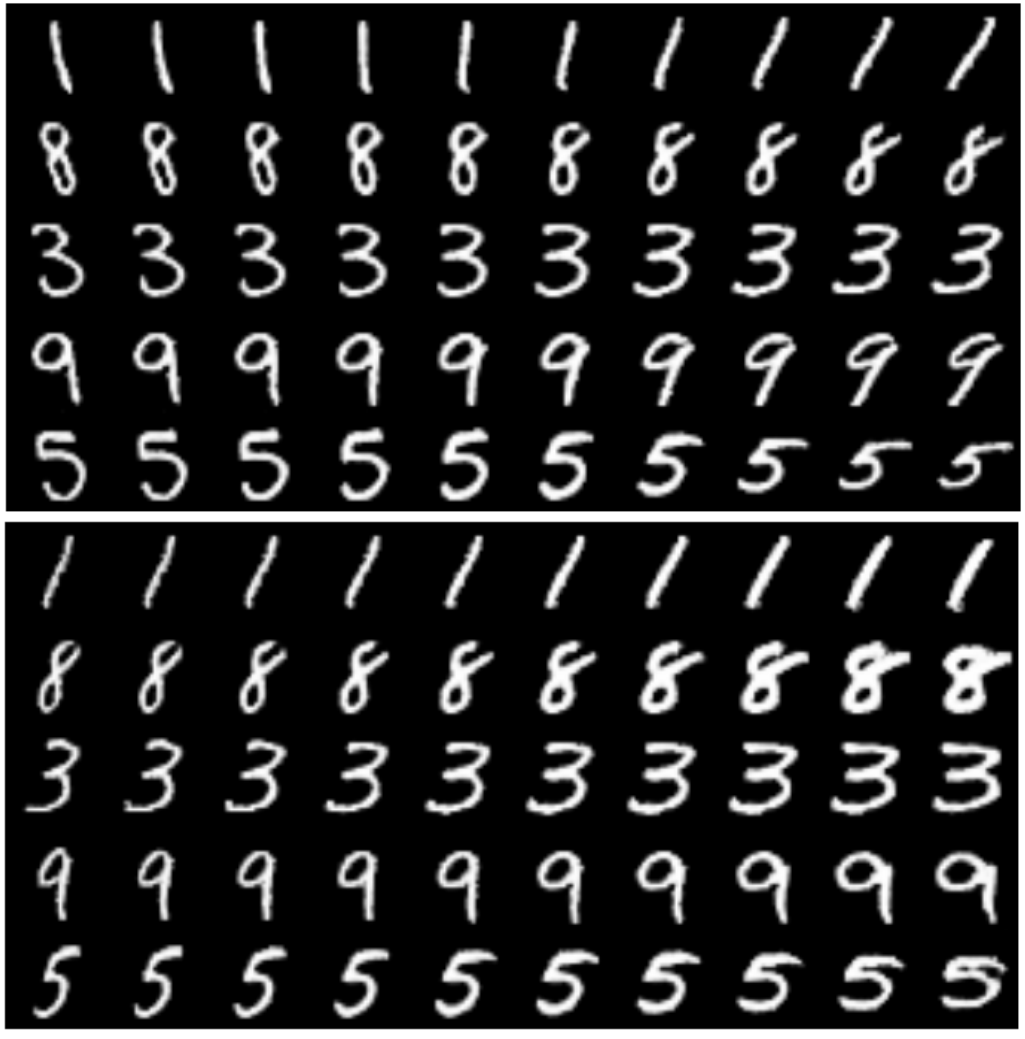
Generative Adversarial Networks

LSGAN: Mao et al. (2016)

Generative Adversarial Networks

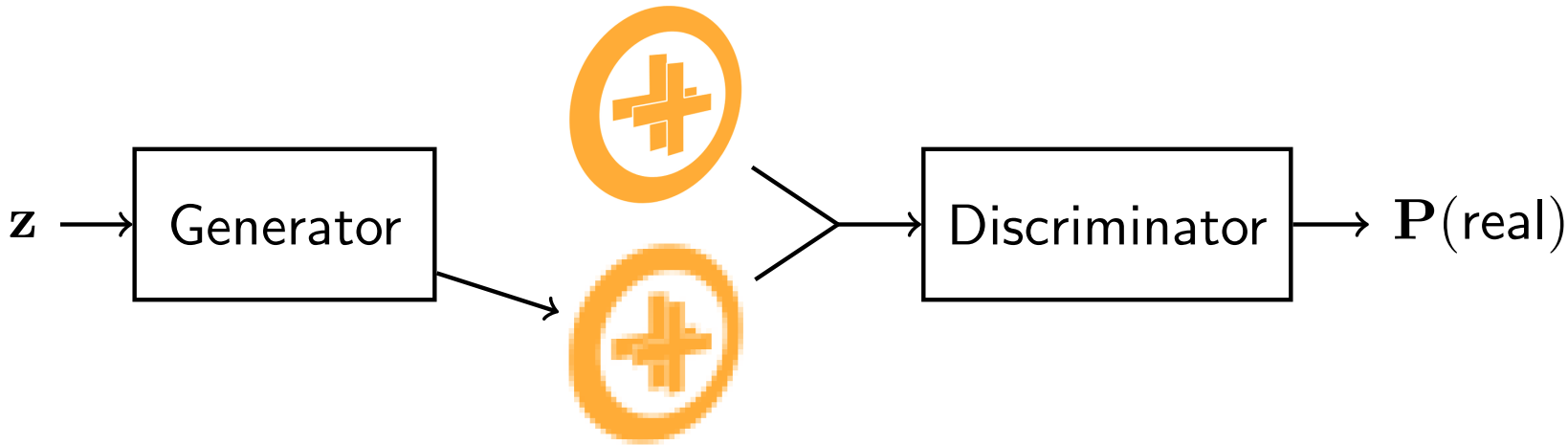
BEGAN: Berthelot et al. (2017)

Generative Adversarial Networks



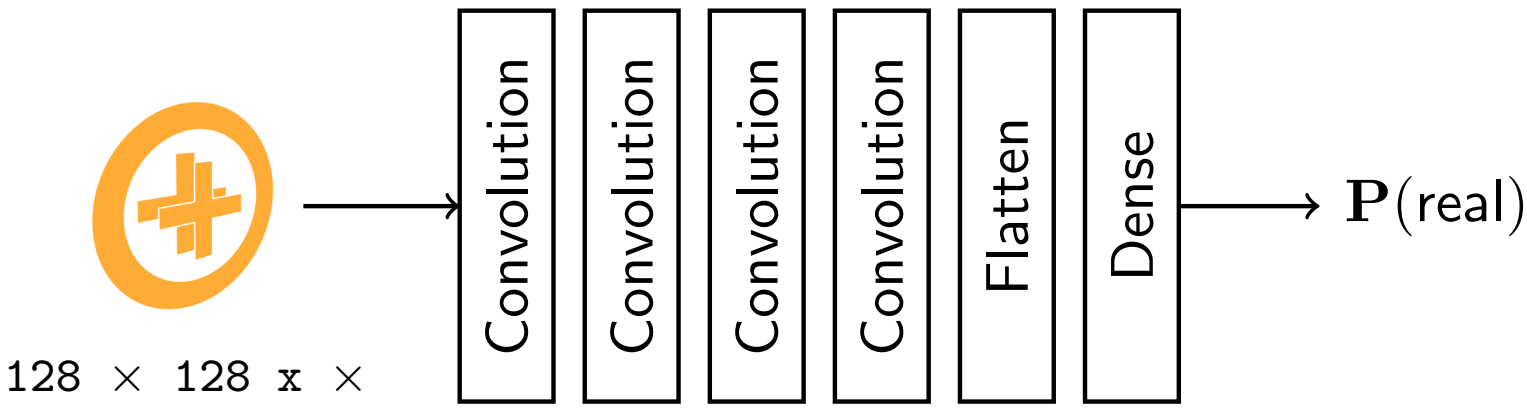
InfoGAN: Chen et al. (2016)

Layer Layer

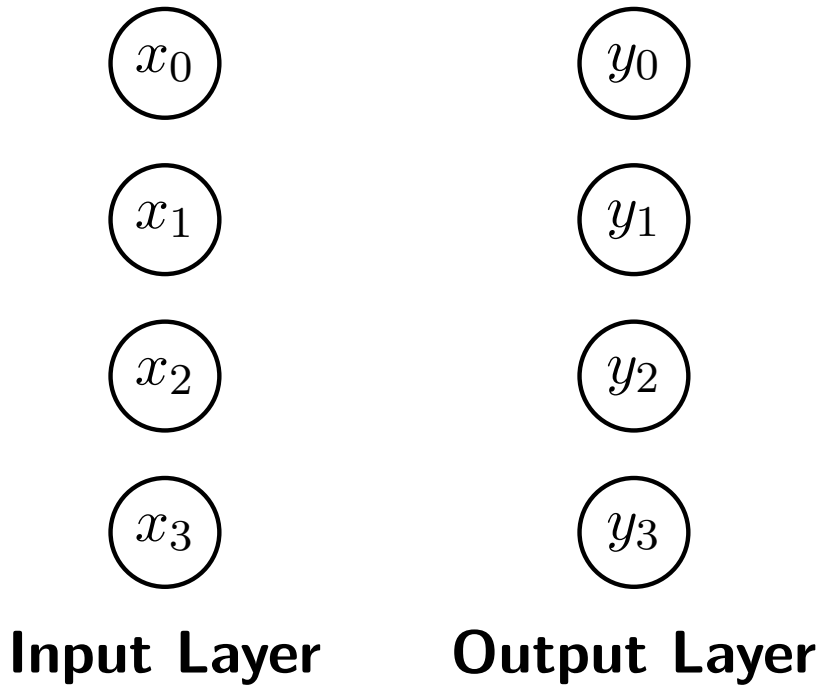


Layer Layer

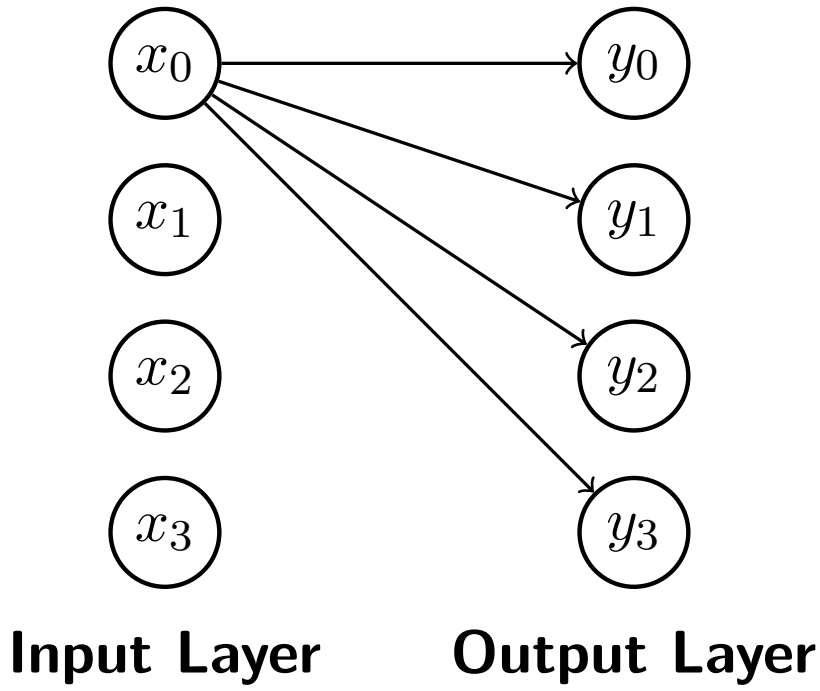
Layer Layer


128 × 128 x ×
Convolution
Convolution
Convolution
Convolution
Flatten
Dense
P(real)

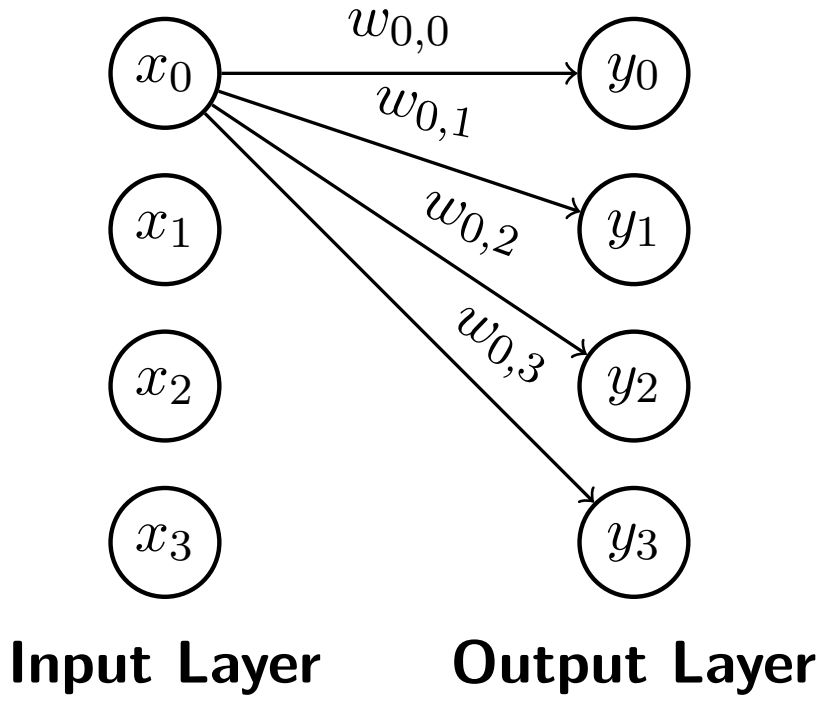
Layers: Dense



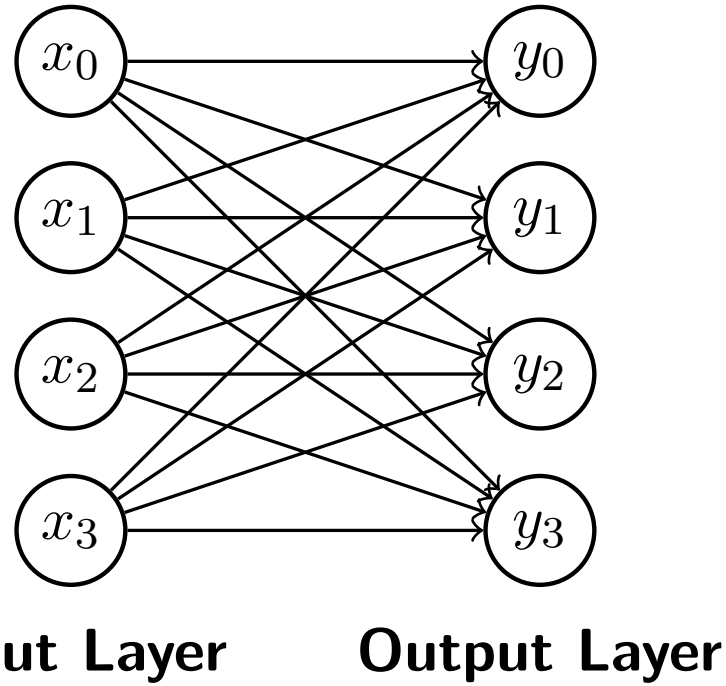
Layers: Dense



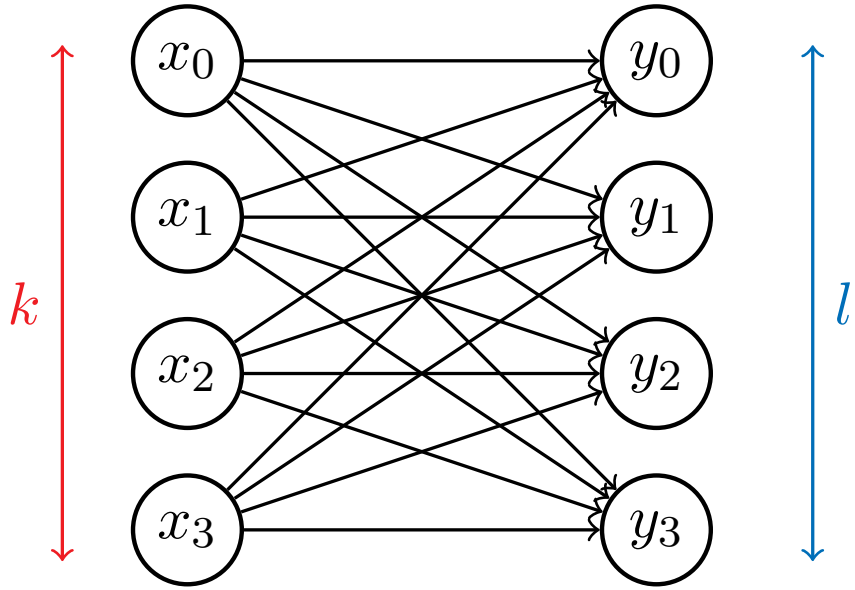
Layers: Dense



Layers: Dense

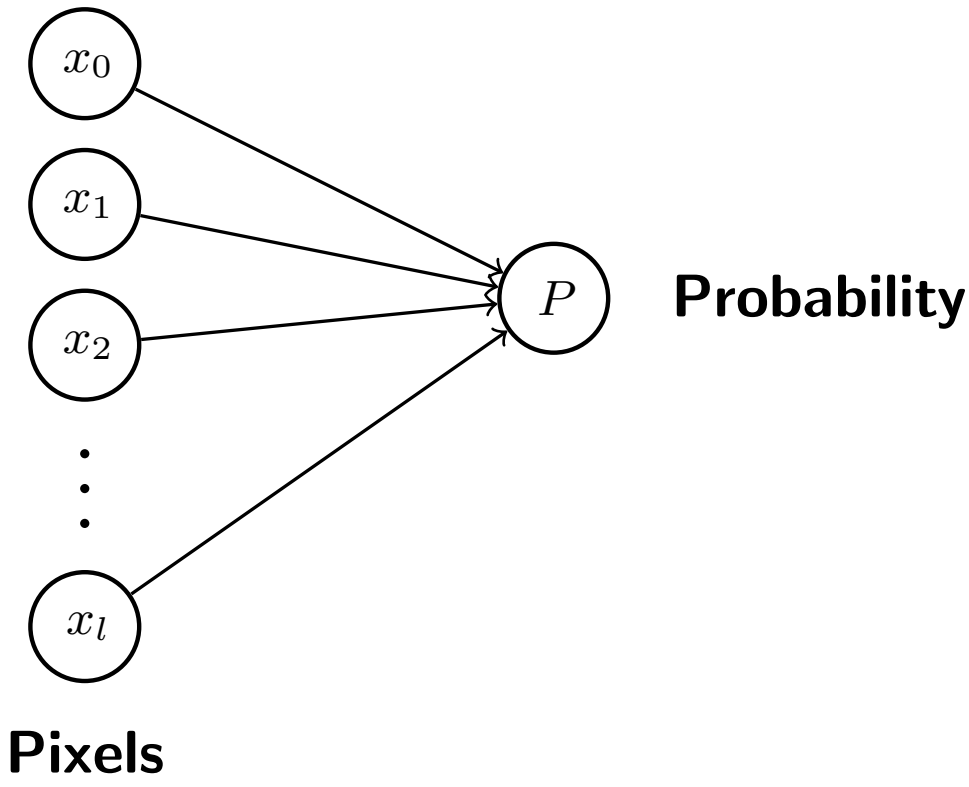


Layers: Dense

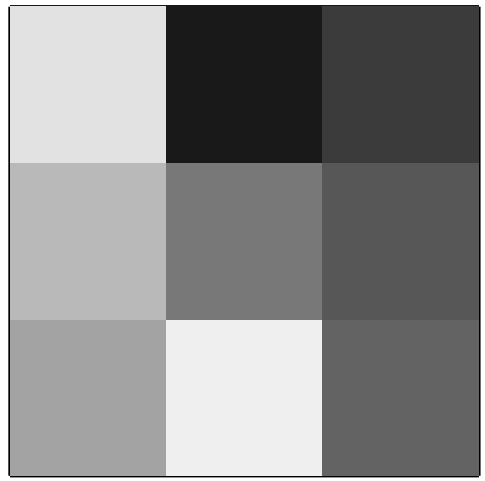


$$
\overset{\overset{k}{\longleftrightarrow}}{n\left\updownarrow\begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} \\ x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,0} & x_{n,1} & x_{n,2} & x_{n,3} \end{bmatrix}\right.} \times \overset{\overset{l}{\longleftrightarrow}}{\begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix}} = \begin{bmatrix} y_{0,0} & y_{0,1} & y_{0,2} & y_{0,3} \\ y_{1,0} & y_{1,1} & y_{1,2} & y_{1,3} \\ \vdots & \vdots & \vdots & \vdots \\ y_{n,0} & y_{n,1} & y_{n,2} & y_{n,3} \end{bmatrix}
$$

Layers: Dense



Layers: Convolution



Image

Layers: Convolution

0.4	0.9	0.1
0.7	0.2	0.6
0.8	0.3	0.5

Image

Layers: Convolution

0.4	0.9	0.1
0.7	0.2	0.6
0.8	0.3	0.5

Image

5.7	2.4
3.1	0.9

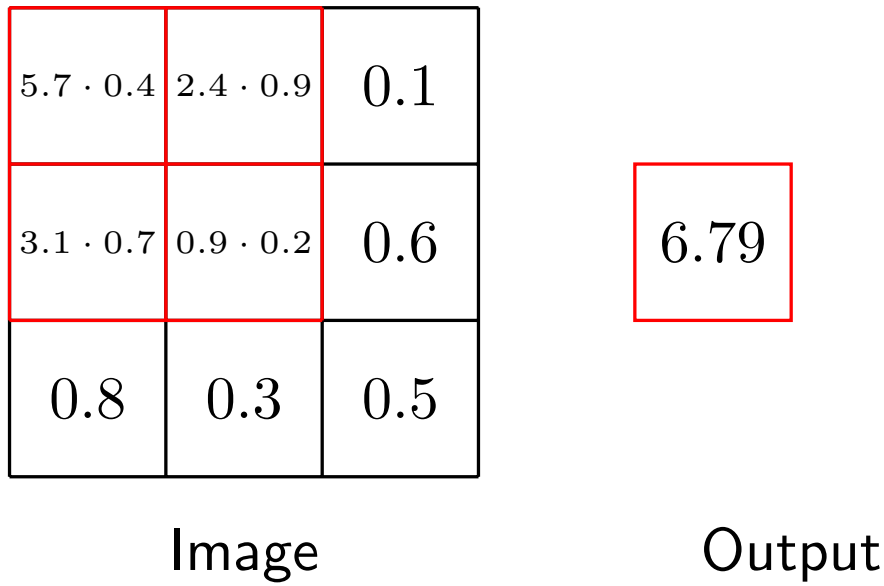
Kernel

Layers: Convolution

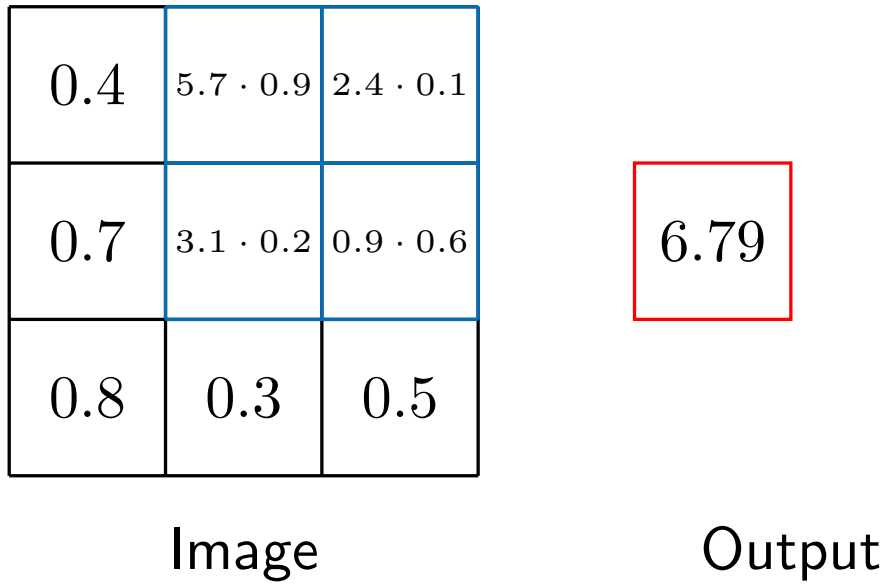
$5.7 \cdot 0.4$	$2.4 \cdot 0.9$	0.1
$3.1 \cdot 0.7$	$0.9 \cdot 0.2$	0.6
0.8	0.3	0.5

Image

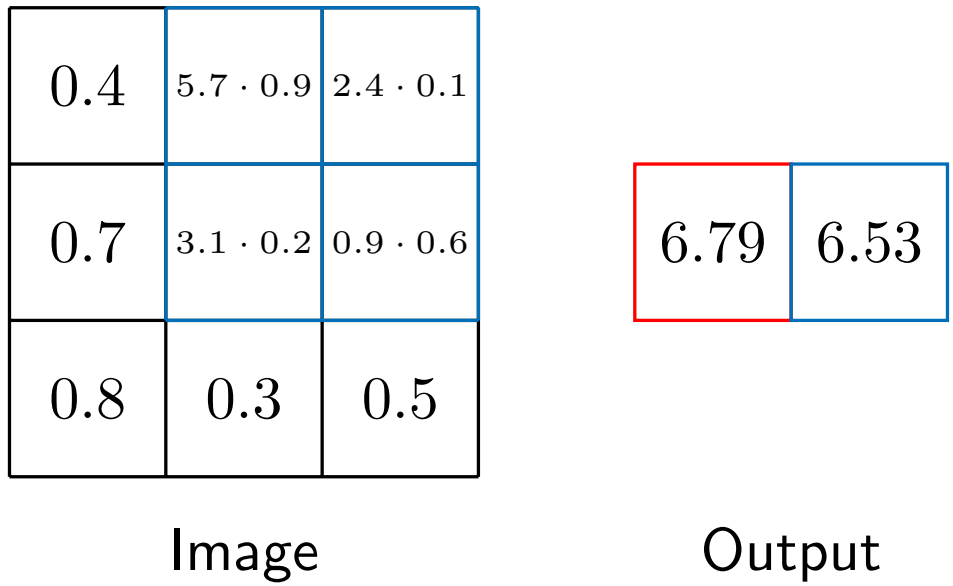
Layers: Convolution



Layers: Convolution

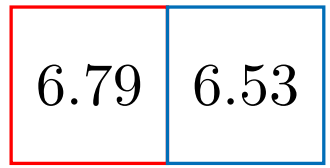


Layers: Convolution



Layers: Convolution

Image



Output

Layers: Convolution

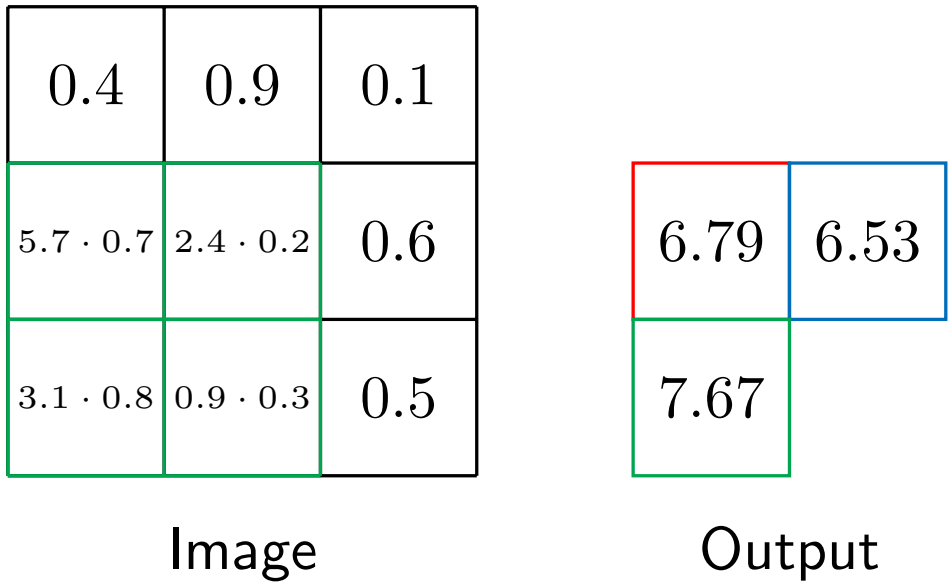
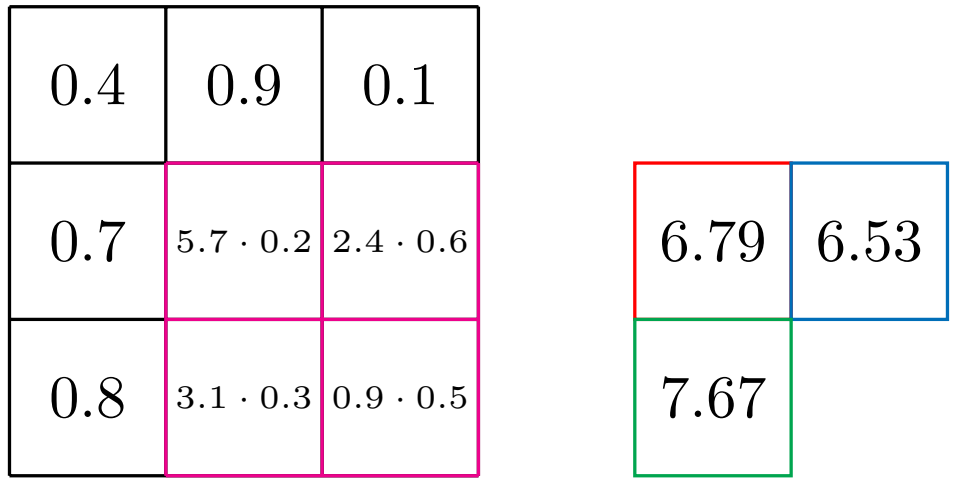
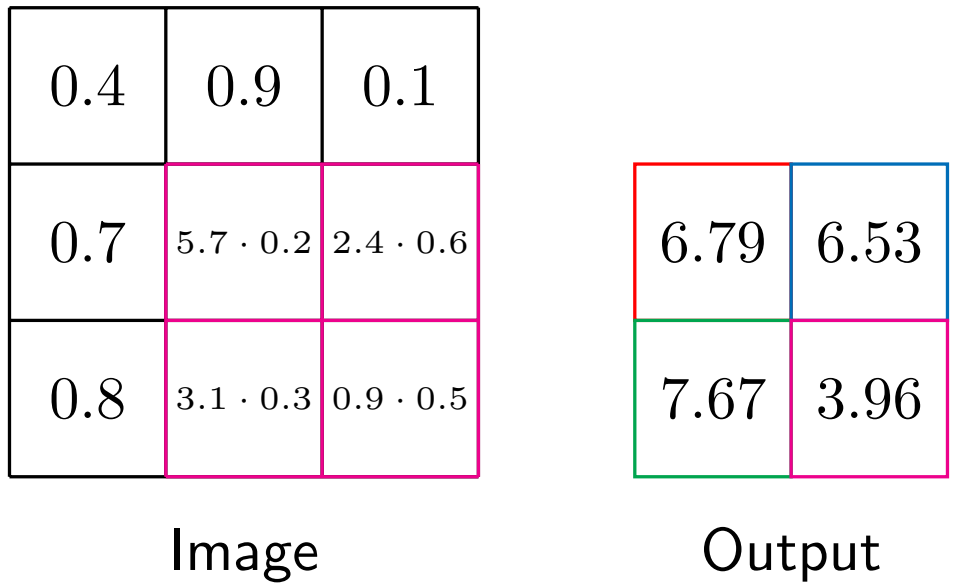


Image Output

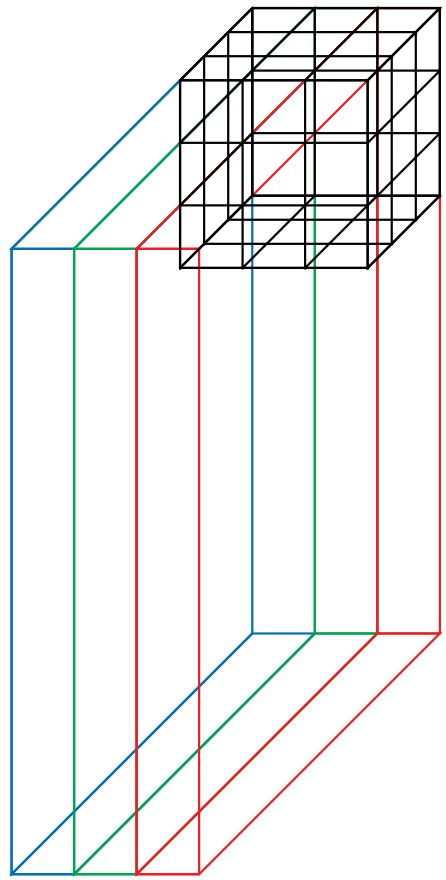
Layers: Convolution



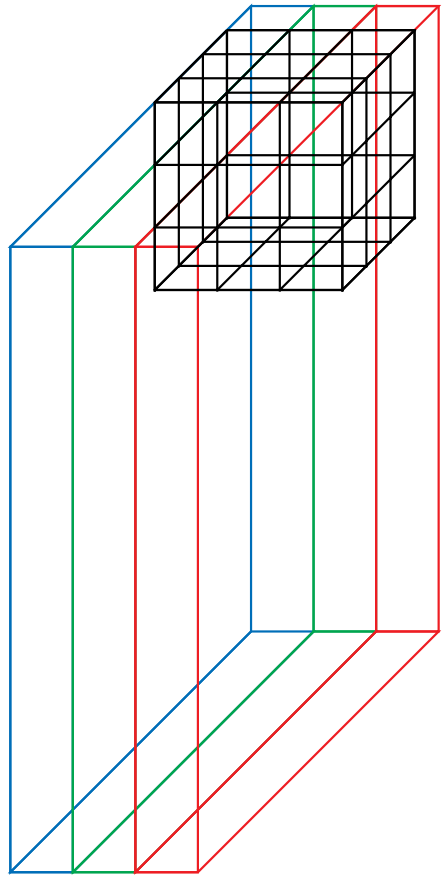
Layers: Convolution



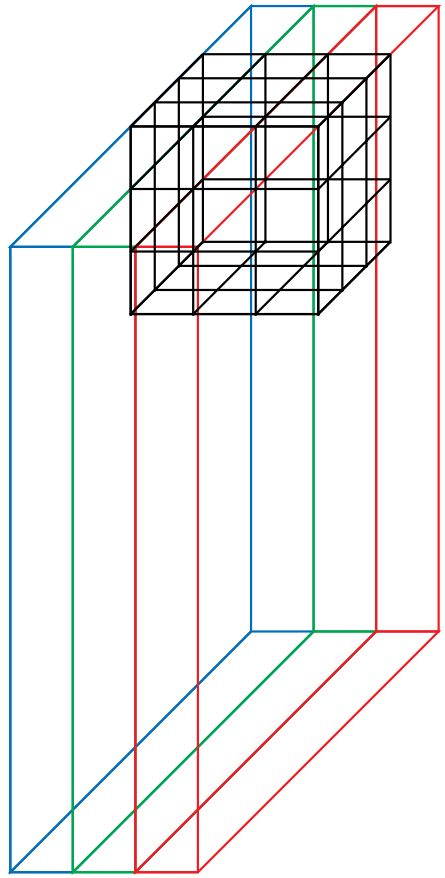
Layers: Convolution



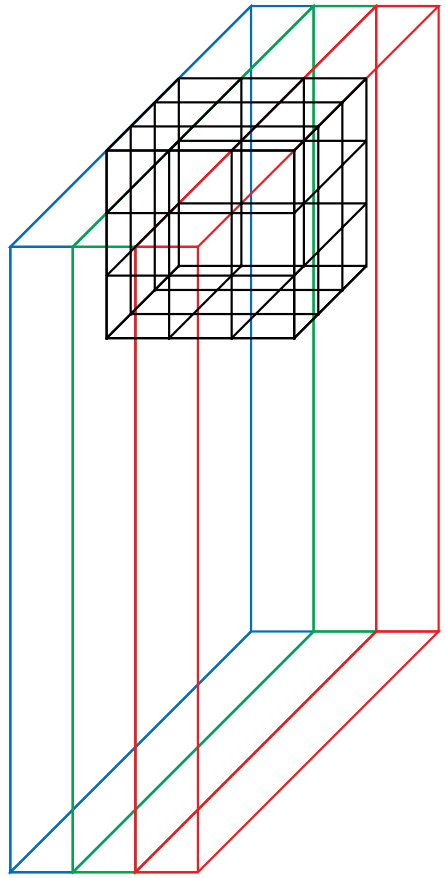
Layers: Convolution



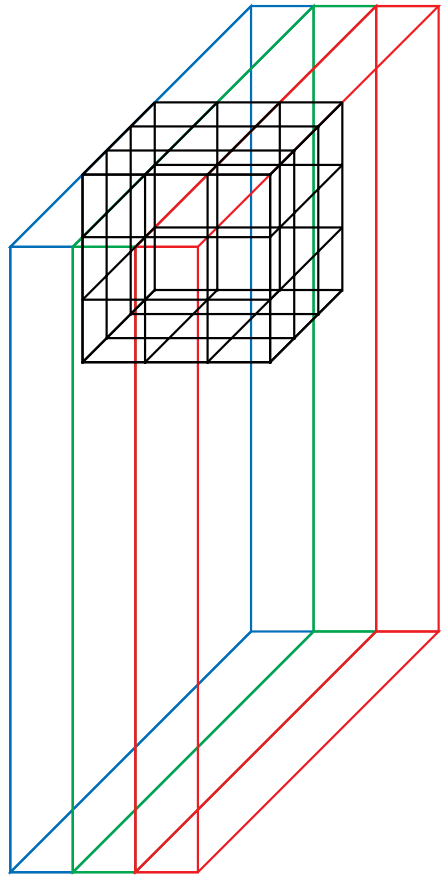
Layers: Convolution



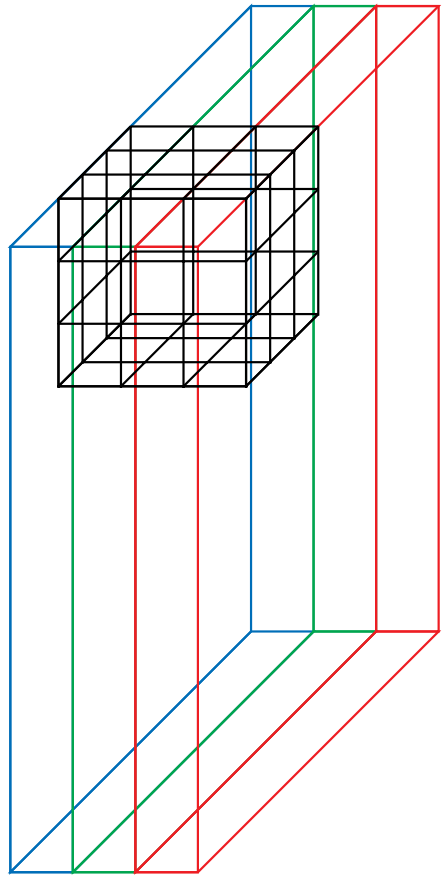
Layers: Convolution



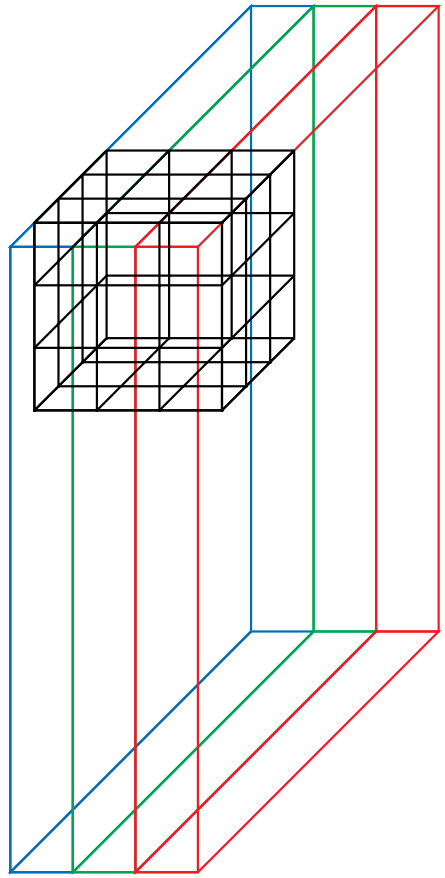
Layers: Convolution



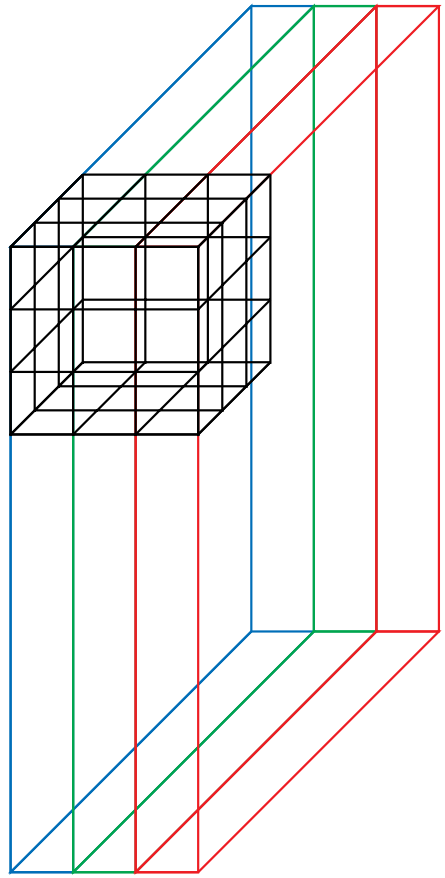
Layers: Convolution



Layers: Convolution



Layers: Convolution



Layers: Convolution

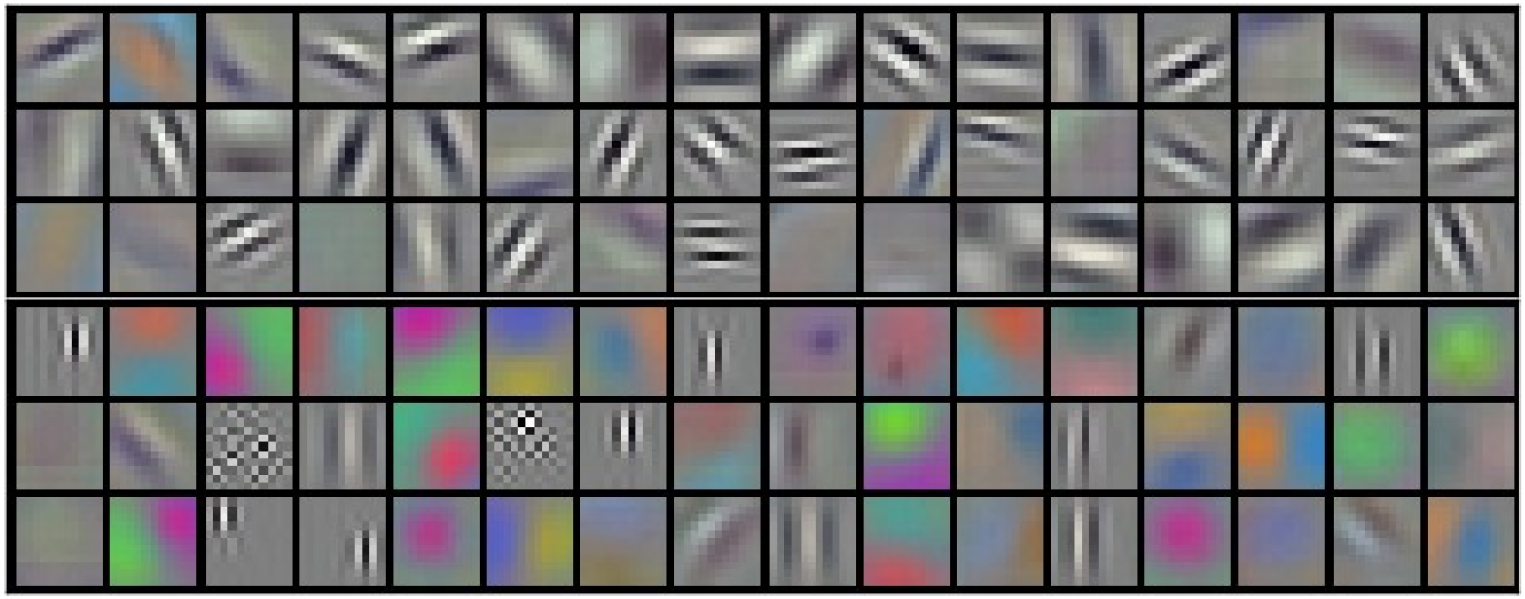
0	1	0
0	1	0
0	1	0

Patterns

0	0	0
-1	1	0
0	0	0

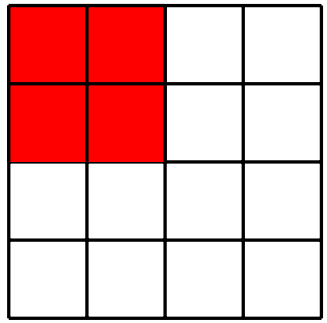
Edges

Layers: Convolution

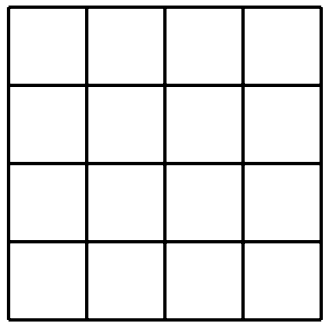


AlexNet: Krizhevsky et al. (2012)

Layer: Convolution

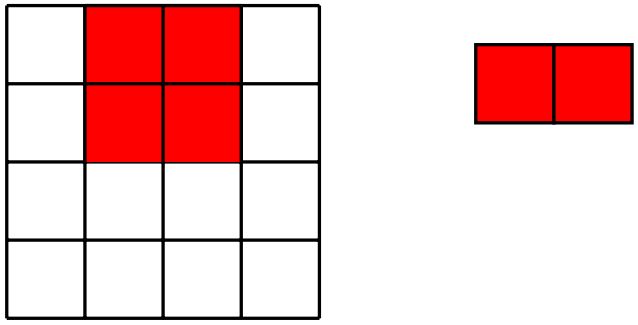


Stride 1

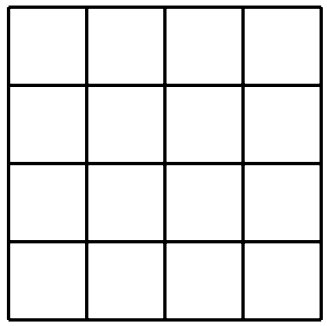


Stride 2

Layer: Convolution

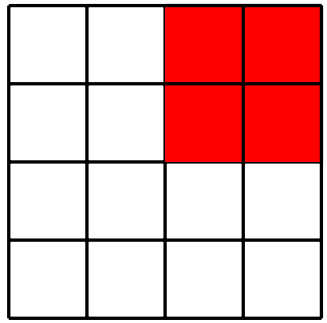


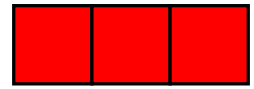
Stride 1



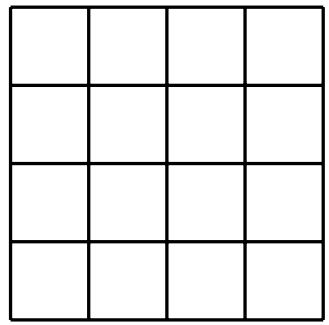
Stride 2

Layer: Convolution



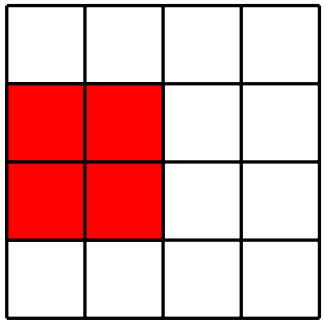


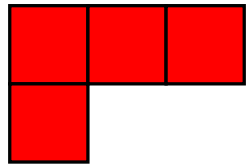
Stride 1



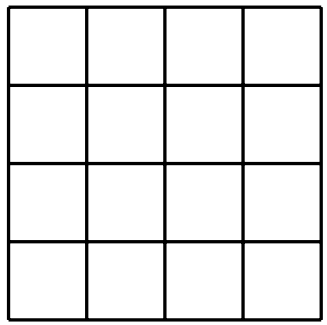
Stride 2

Layer: Convolution



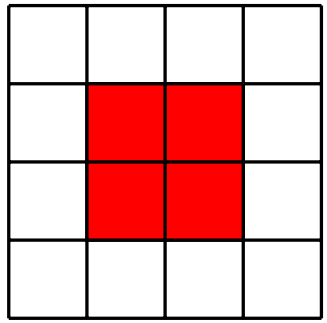


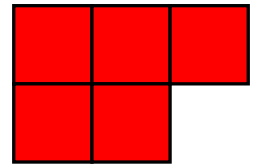
Stride 1



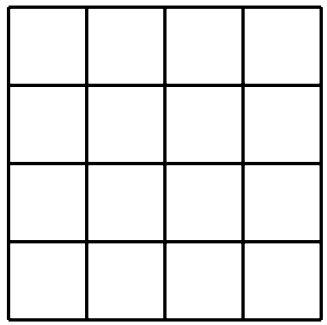
Stride 2

Layer: Convolution



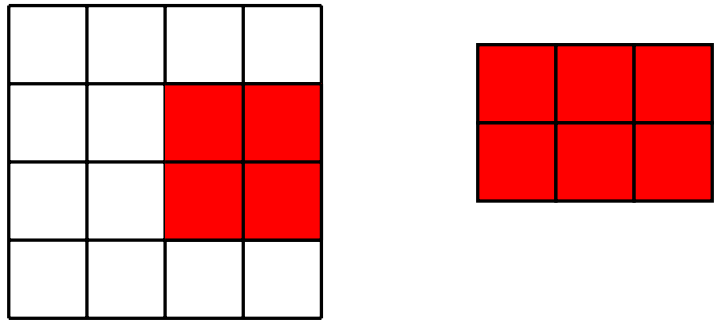


Stride 1

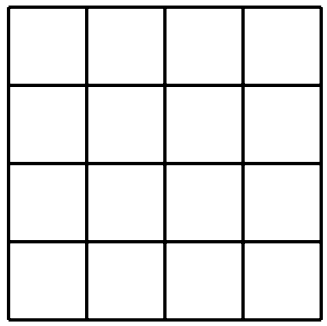


Stride 2

Layer: Convolution

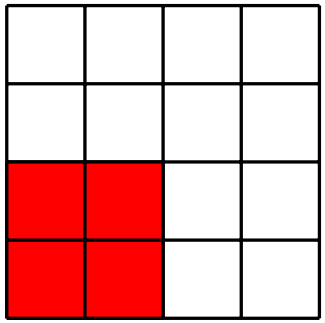


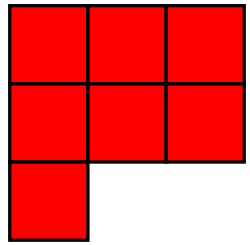
Stride 1



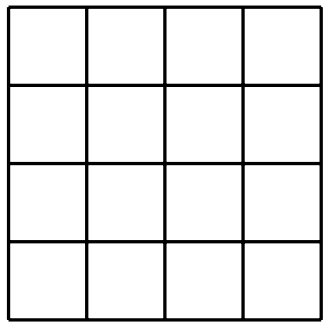
Stride 2

Layer: Convolution



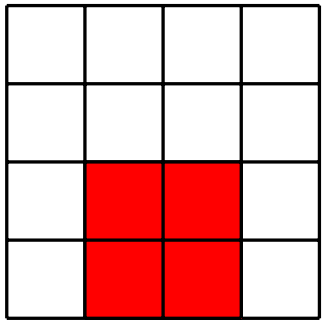


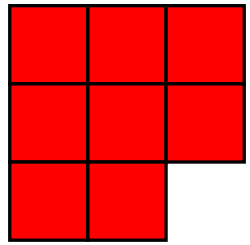
Stride 1



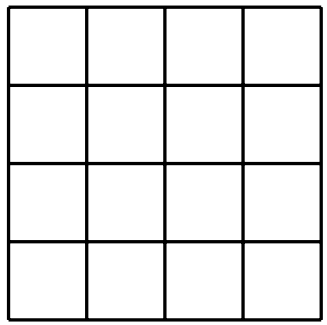
Stride 2

Layer: Convolution



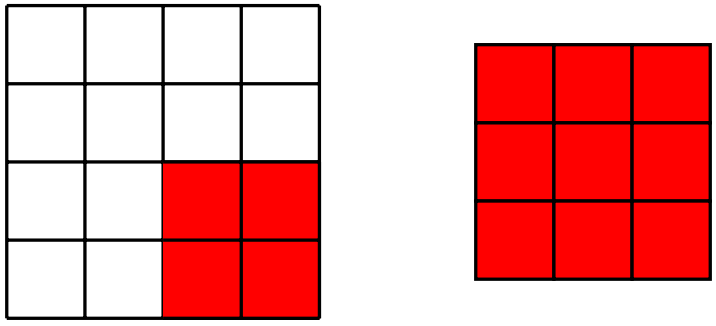


Stride 1

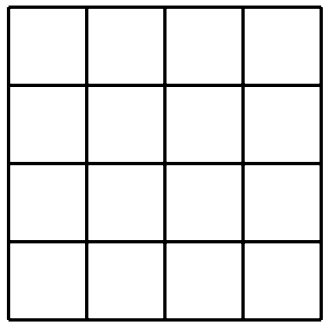


Stride 2

Layer: Convolution

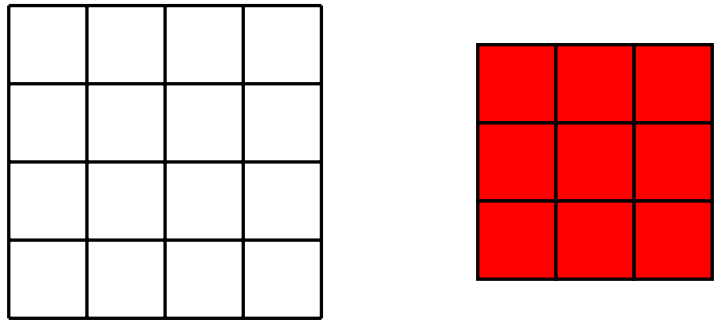


Stride 1

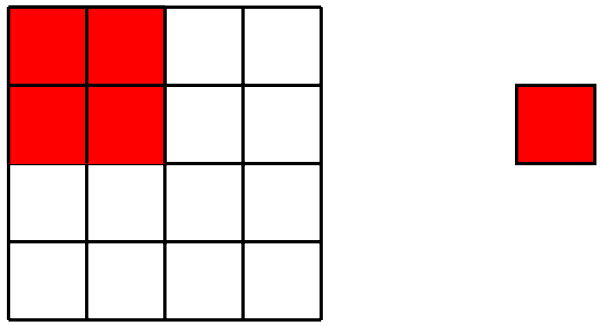


Stride 2

Layer: Convolution

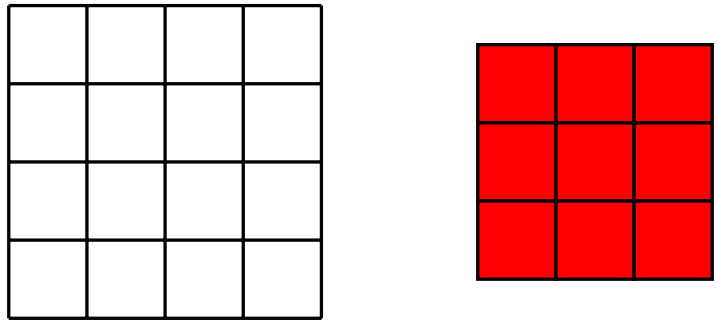


Stride 1

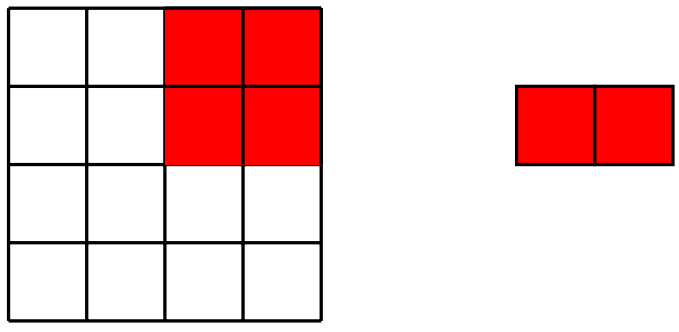


Stride 2

Layer: Convolution

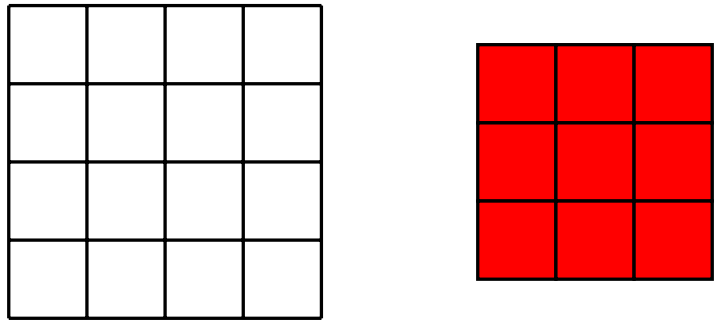


Stride 1

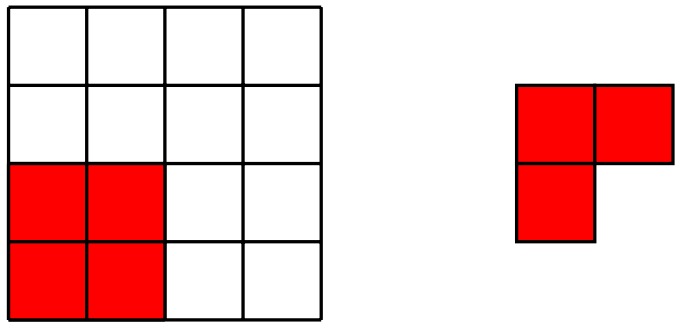


Stride 2

Layer: Convolution

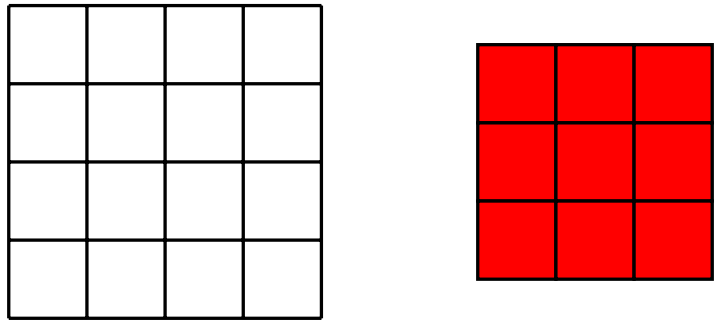


Stride 1

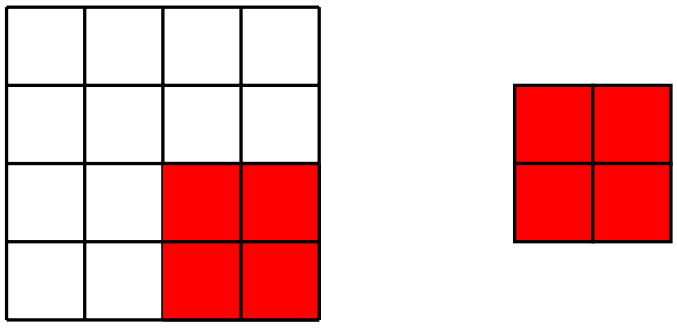


Stride 2

Layer: Convolution

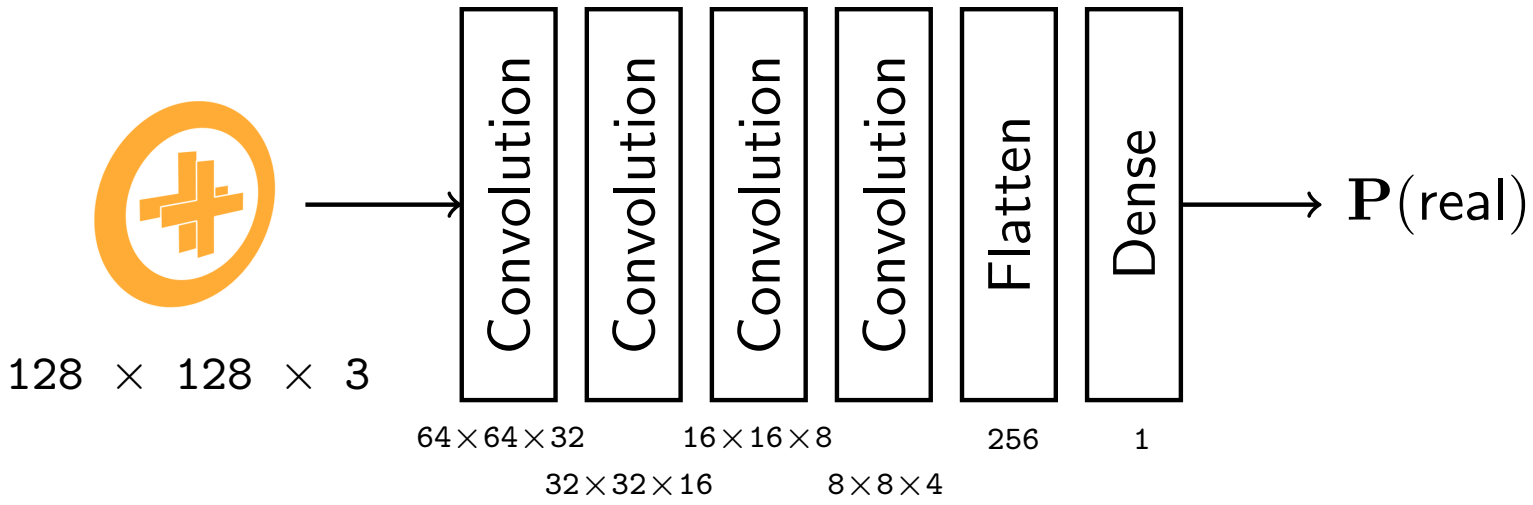


Stride 1



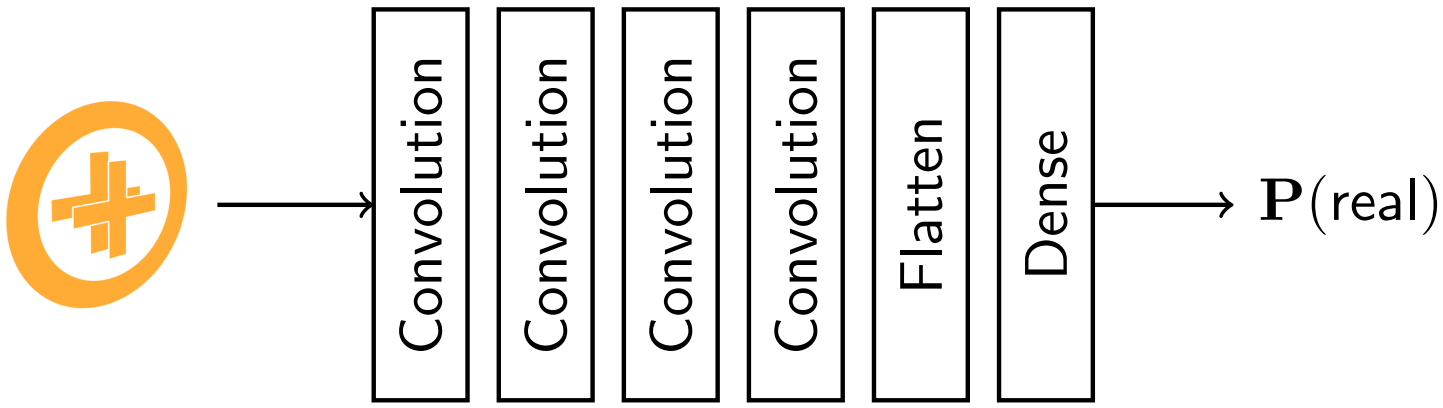
Stride 2

Layer: Convolution

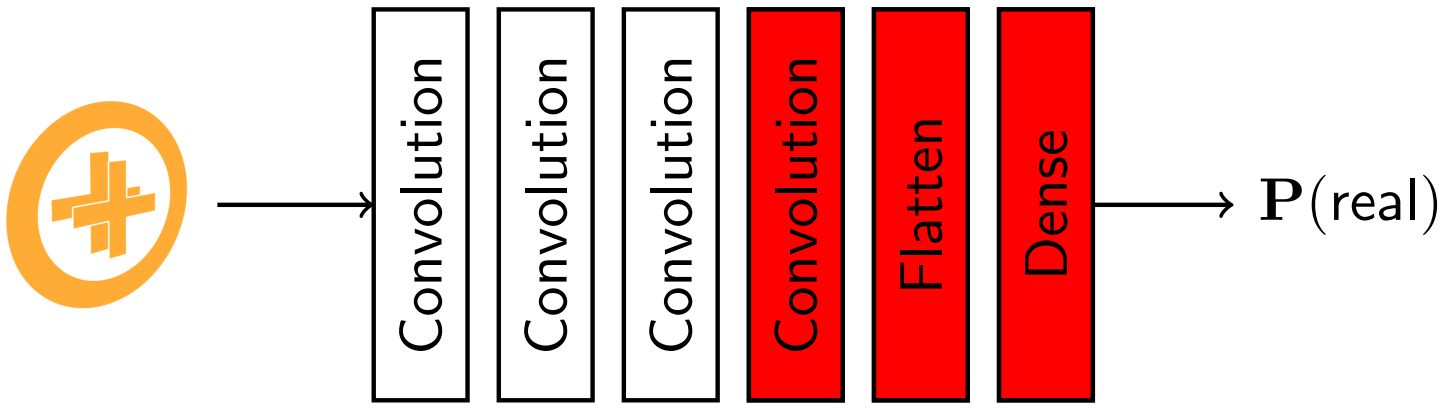


Kernel: 5×5
Stride: 2

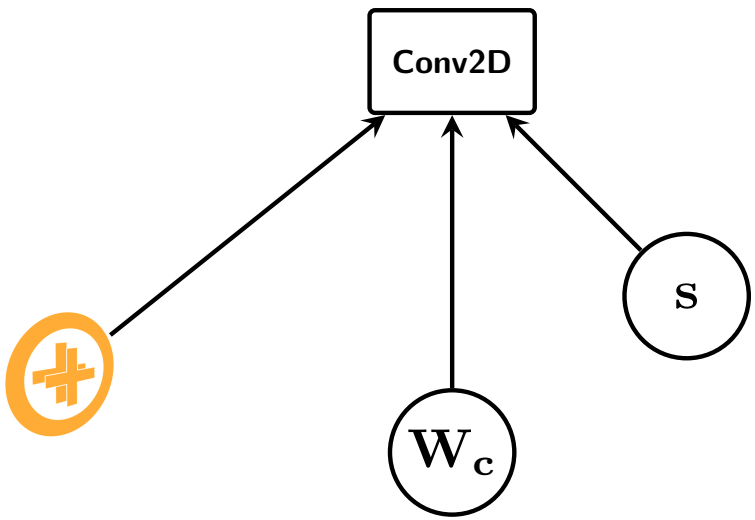
Graph Layer


Convolution
Convolution
Convolution
Convolution
Flatten
Dense
$\mathbf{P}(\text{real})$

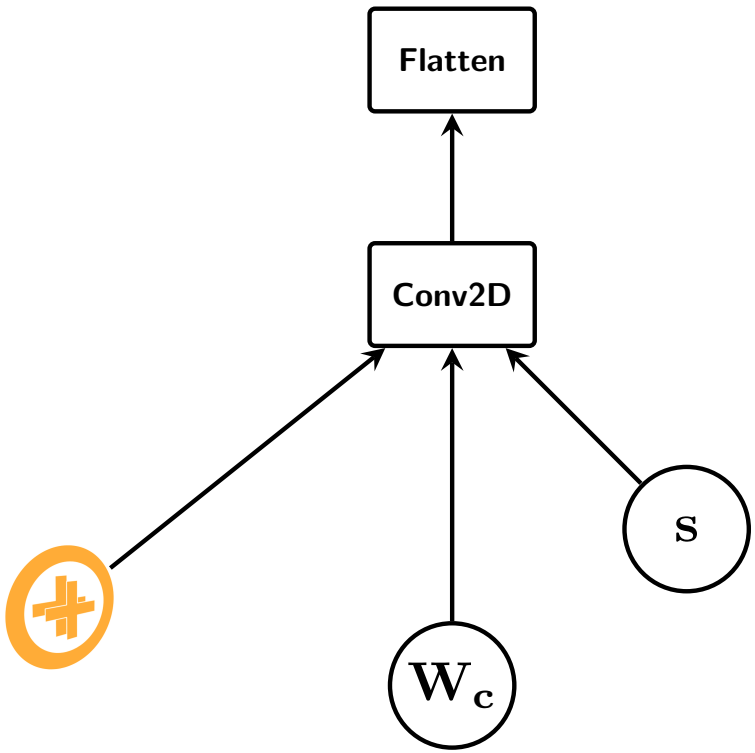
Graph Layer


Convolution
Convolution
Convolution
Convolution
Flatten
Dense
$\mathbf{P}(\text{real})$

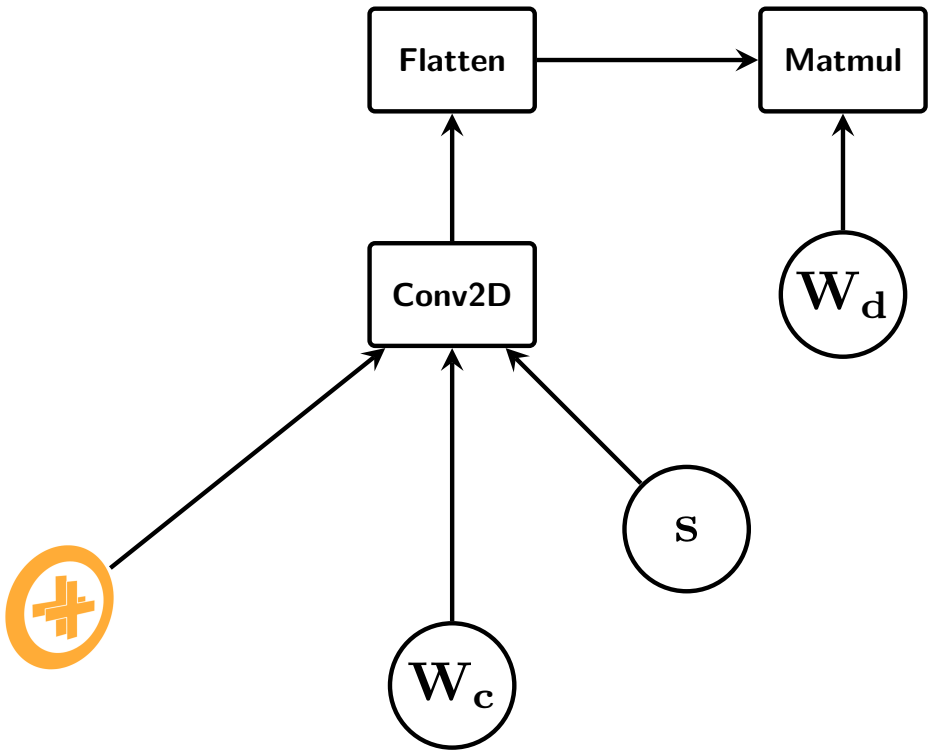
Graph Layer



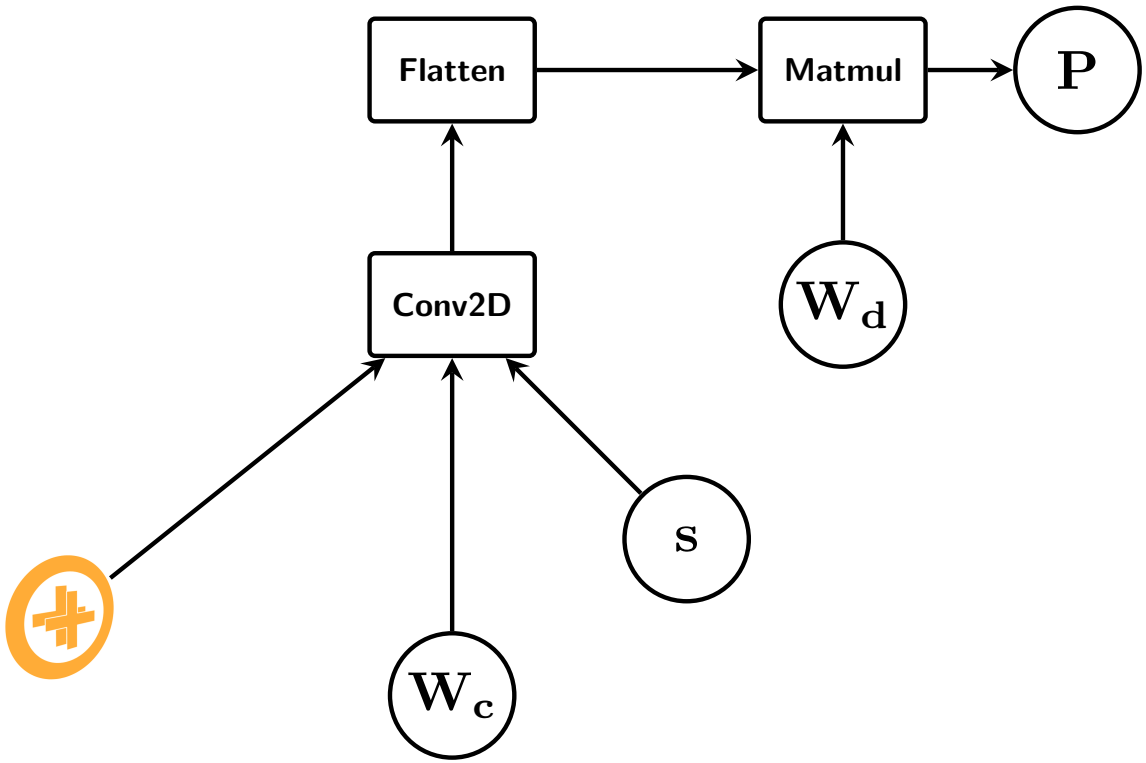
Graph Layer



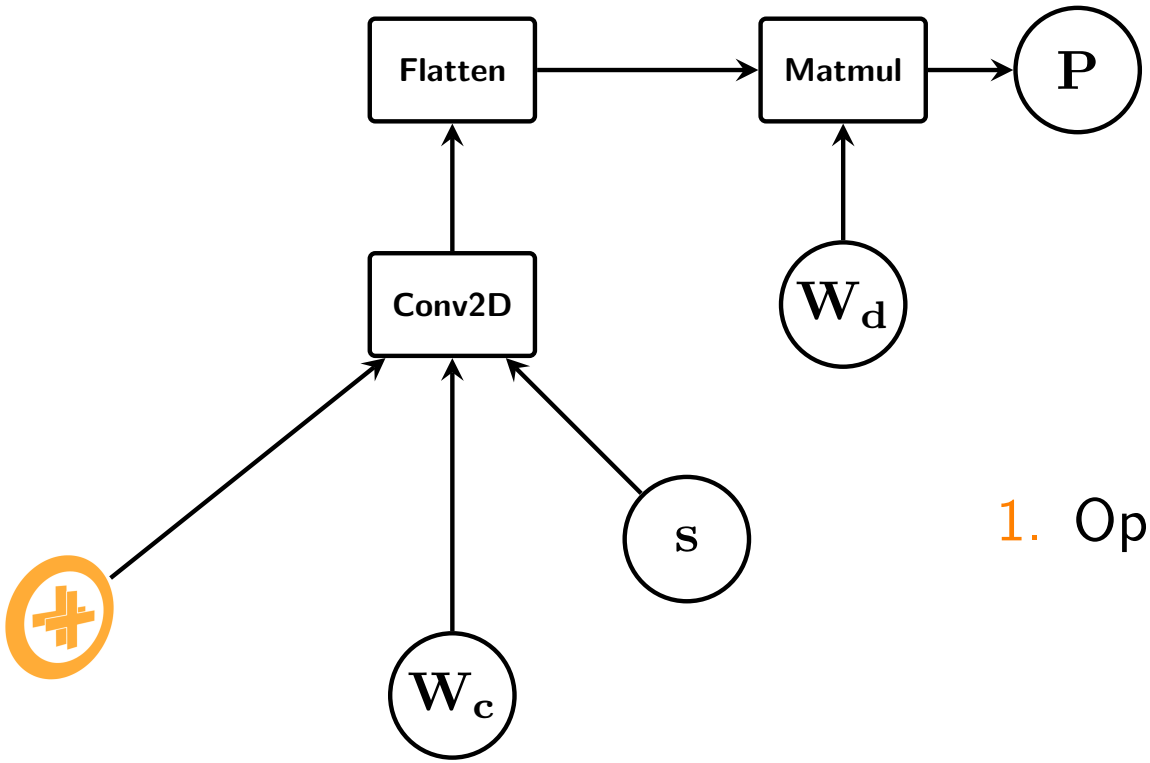
Graph Layer



Graph Layer

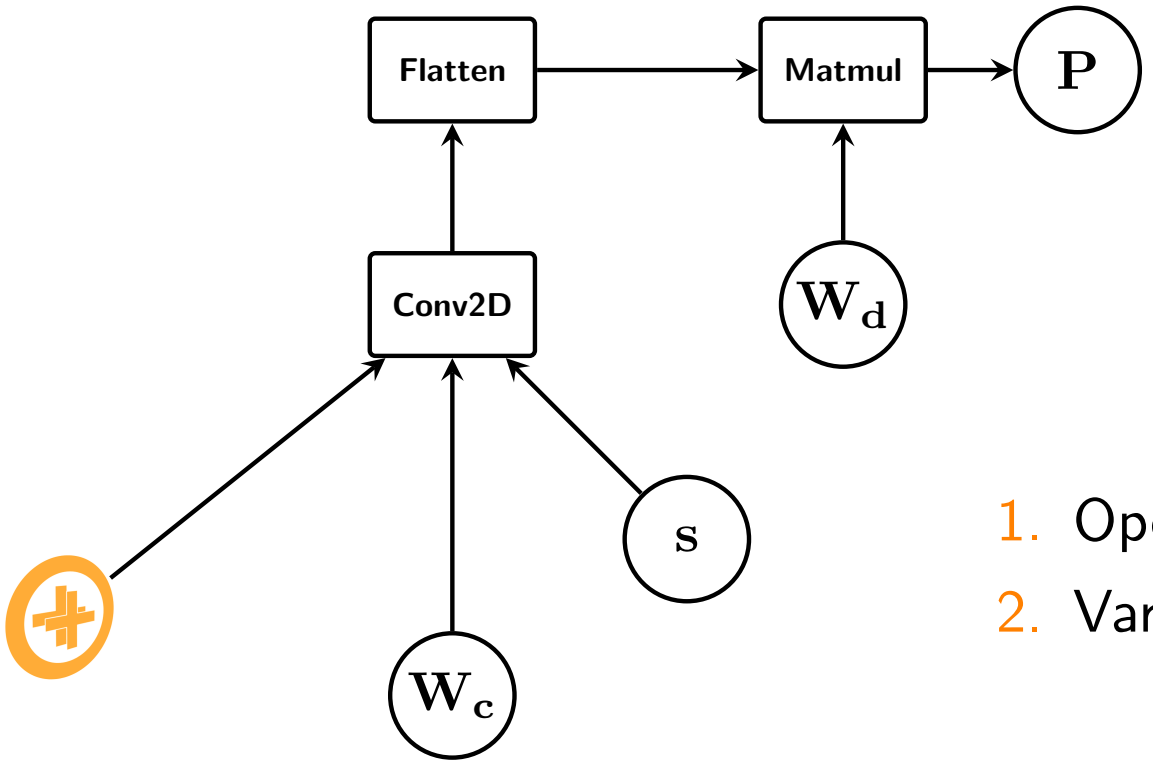


Graph Layer



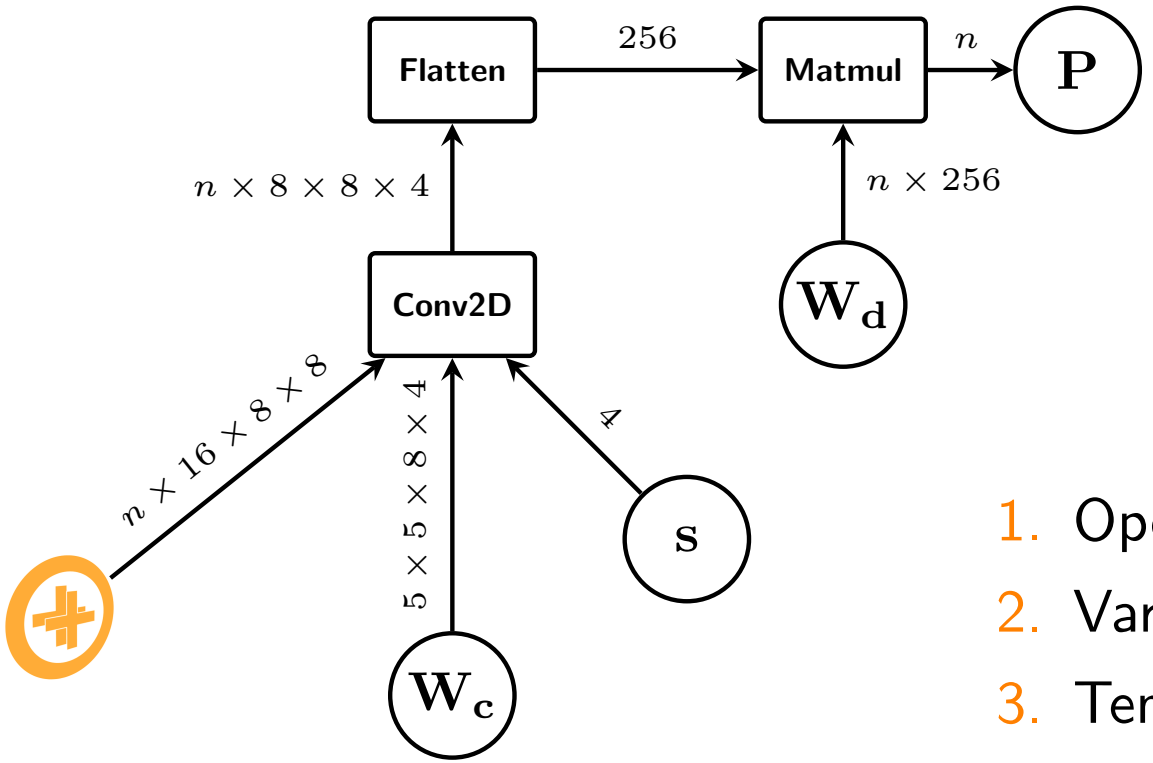
1. Operations

Graph Layer



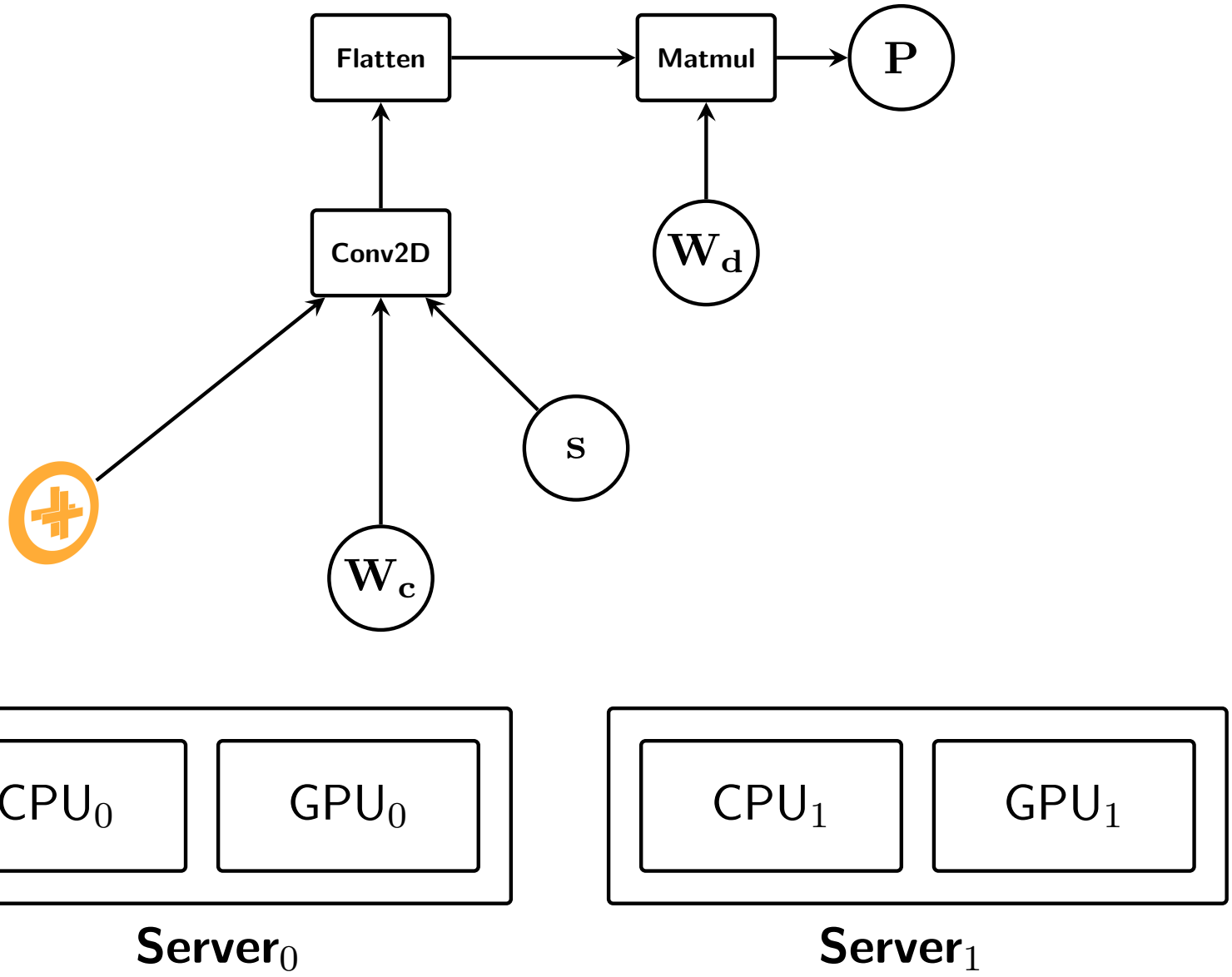
1. Operations
2. Variables

Graph Layer

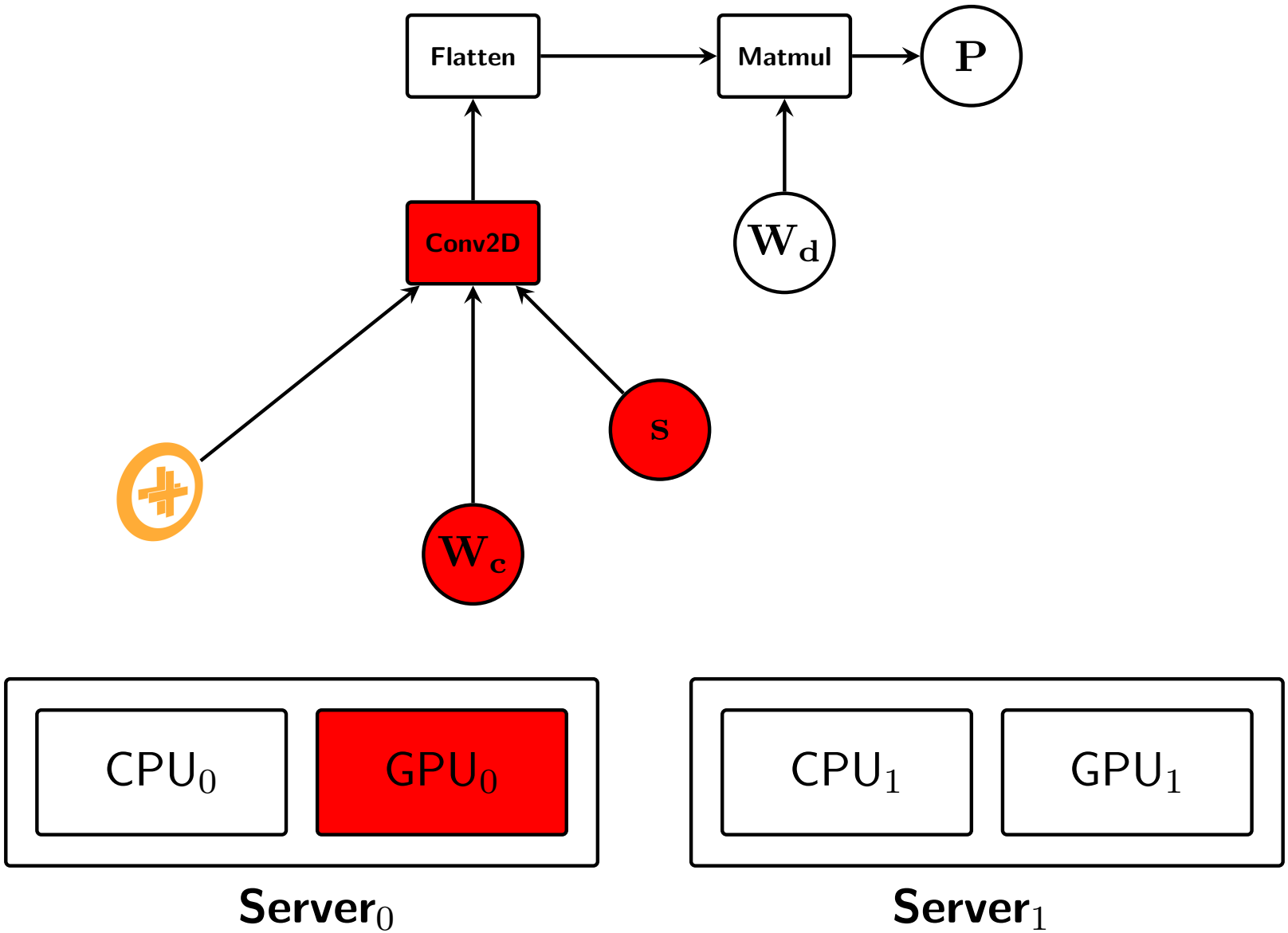


1. Operations
2. Variables
3. Tensors

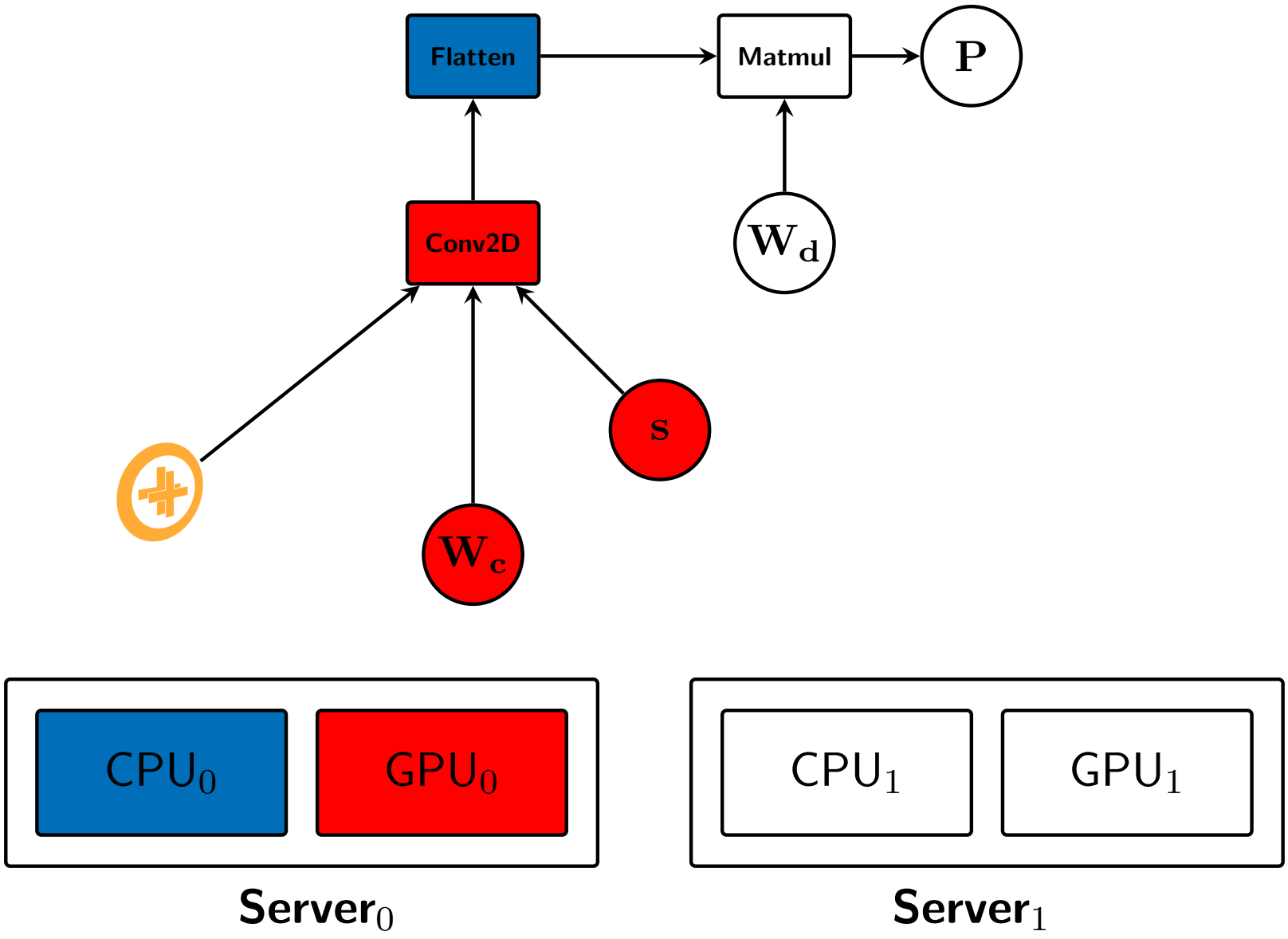
Graph Layer


Flatten
Matmul
P
Conv2D
$\mathbf{W_d}$
$\mathbf{s}$
$\mathbf{W_c}$
CPU_0
GPU_0
CPU_1
GPU_1
$\mathbf{Server_0}$
$\mathbf{Server_1}$

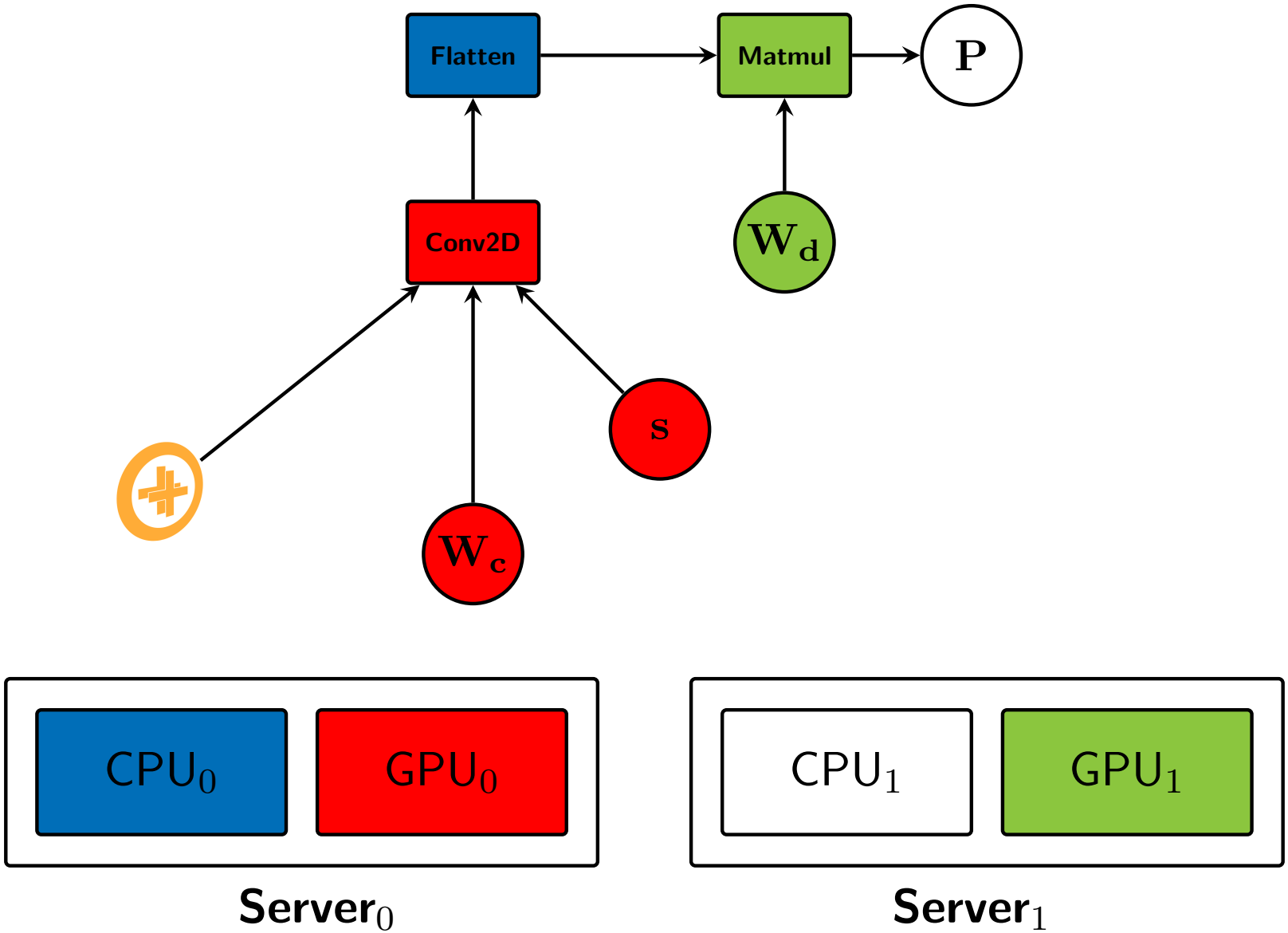
Graph Layer


Flatten
Matmul
P
Conv2D
$\mathbf{W_d}$
S
$\mathbf{W_c}$
CPU_0
GPU_0
CPU_1
GPU_1
$\mathbf{Server_0}$
$\mathbf{Server_1}$

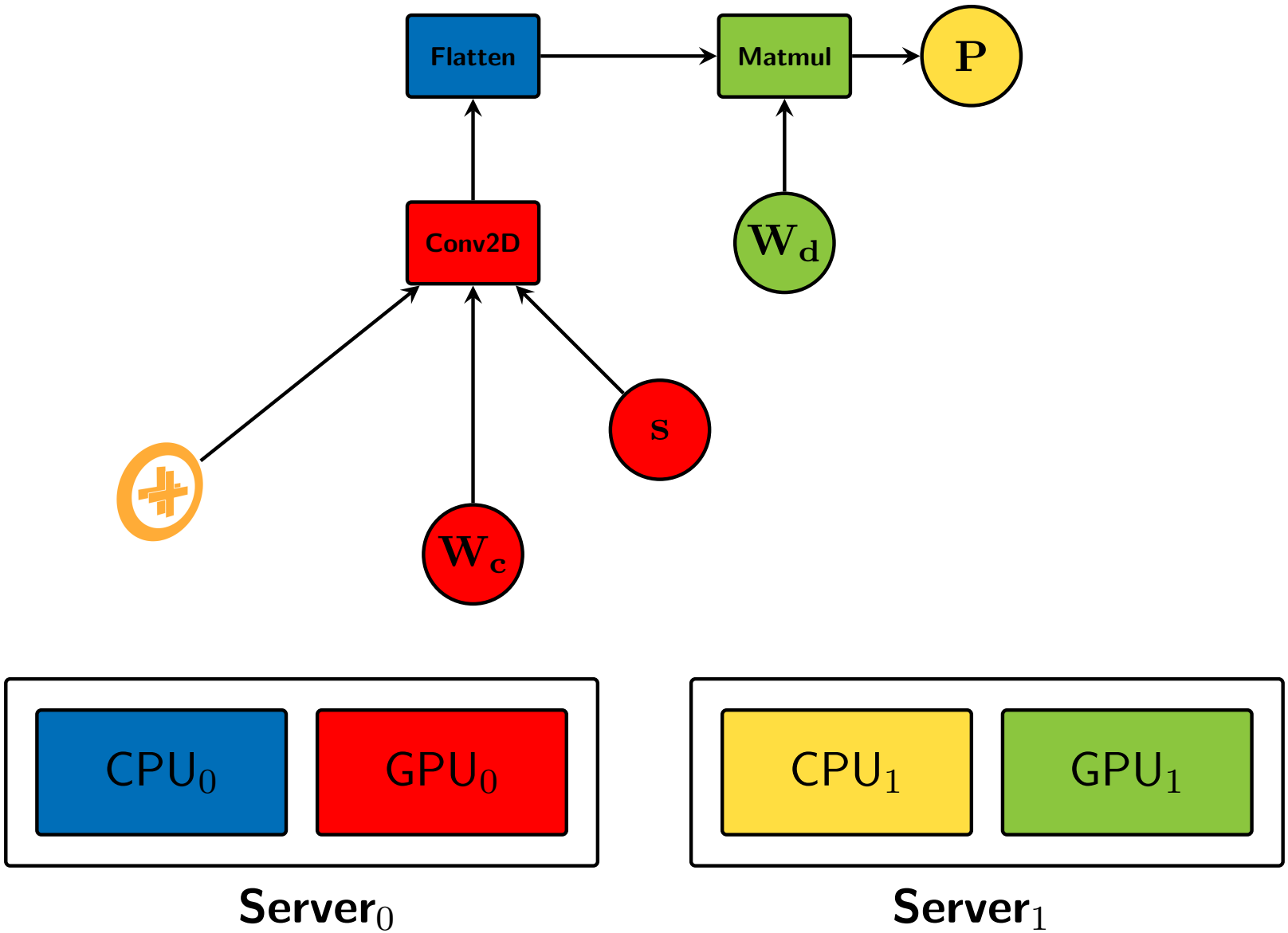
Graph Layer


Flatten
Matmul
P
Conv2D
$\mathbf{W_d}$
S
$\mathbf{W_c}$
CPU_0
GPU_0
CPU_1
GPU_1
$\mathbf{Server_0}$
$\mathbf{Server_1}$

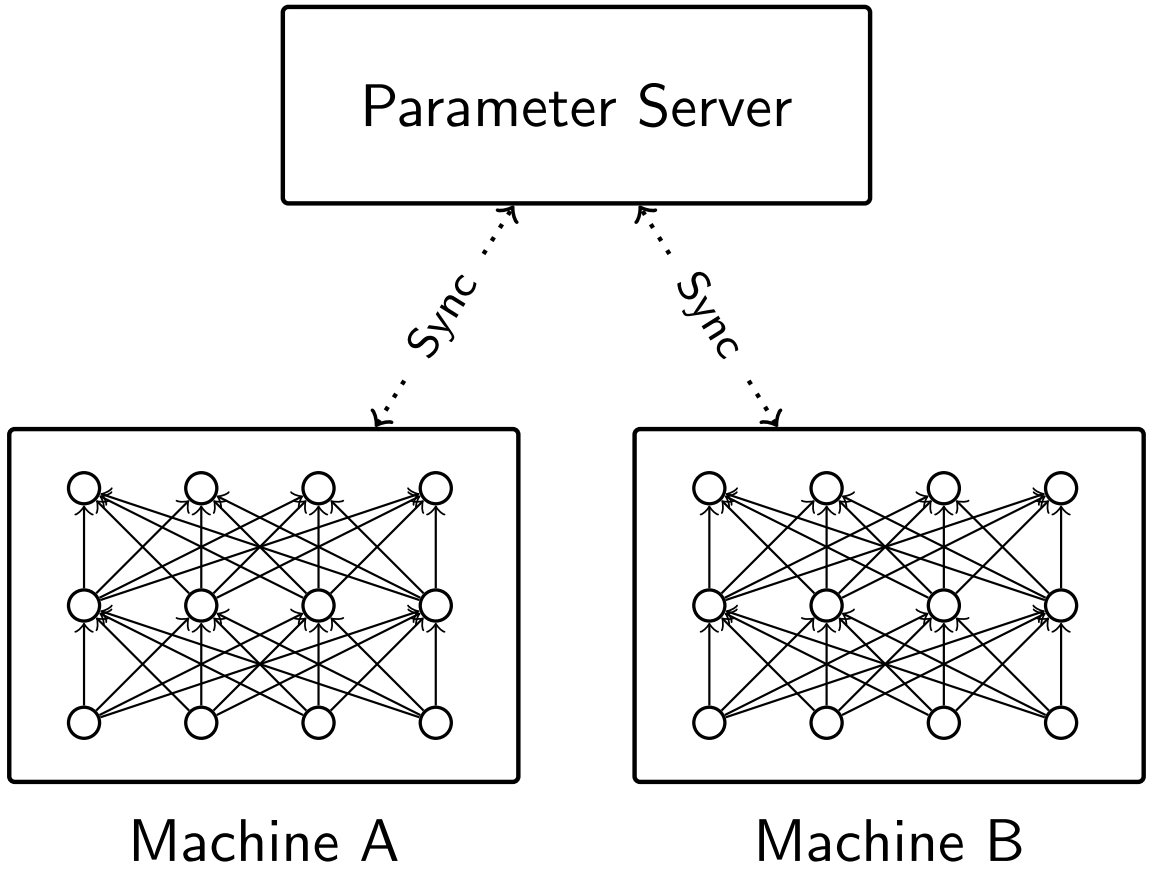
Graph Layer


Flatten
Matmul
P
Conv2D
$\mathbf{W_d}$
$\mathbf{s}$
$\mathbf{W_c}$
CPU_0
GPU_0
CPU_1
GPU_1
$\mathbf{Server_0}$
$\mathbf{Server_1}$

Graph Layer

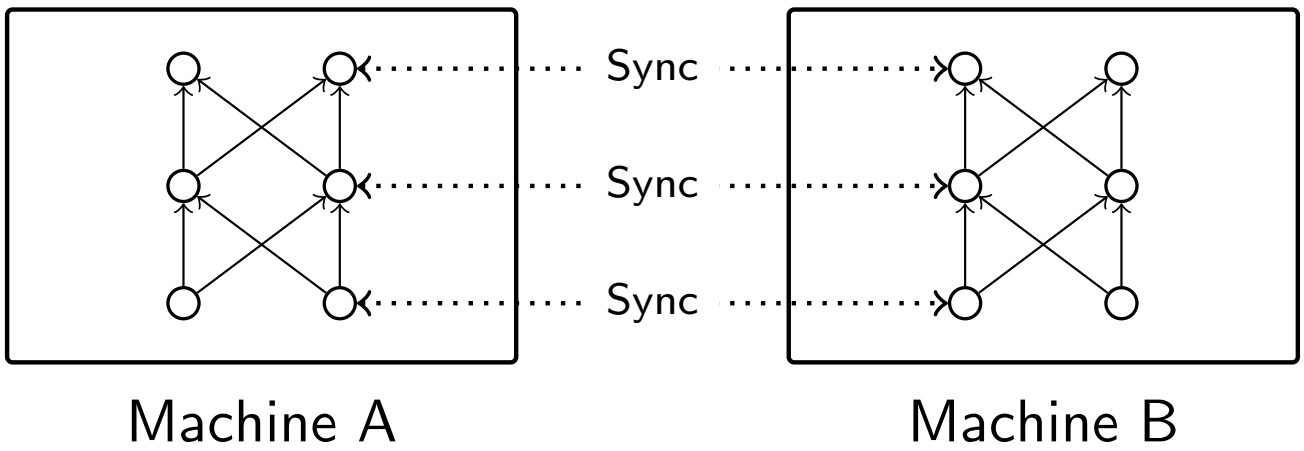

Flatten
Matmul
P
Conv2D
$\mathbf{W_d}$
$\mathbf{s}$
$\mathbf{W_c}$
CPU_0
GPU_0
CPU_1
GPU_1
$\mathbf{Server}_0$
$\mathbf{Server}_1$

Graph Layer Parallelism



Data Parallelism

Graph Layer Parallelism



Model Parallelism

Static Graphs

- Declarative programming model
- Static shapes and sizes
- Graphs defined once
- Can be compiled and optimized

```
Matrix<6, 9> a = ...;
Matrix<6, 9> b = ...;
Matrix<16, 6> c = ...;
scalar s = 7;

d := s * a + b;
e := matmul(c, d);

result := evaluate(e);
```

Static Graphs

- Declarative programming model
- Static shapes and sizes
- Graphs defined once
- Can be compiled and optimized

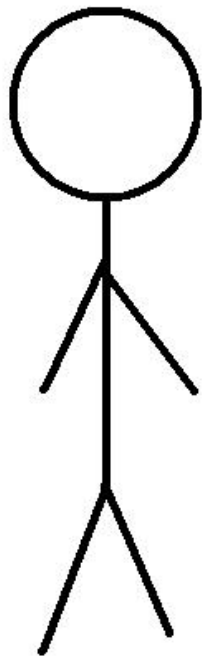
```
x = if_clause(condition,
              then_value,
              else_value);

while_loop([a, b],
           condition_function,
           body_function);
```

Dynamic Graphs

- Imperative programming model
- Dynamic shapes and sizes
- Graphs defined by run

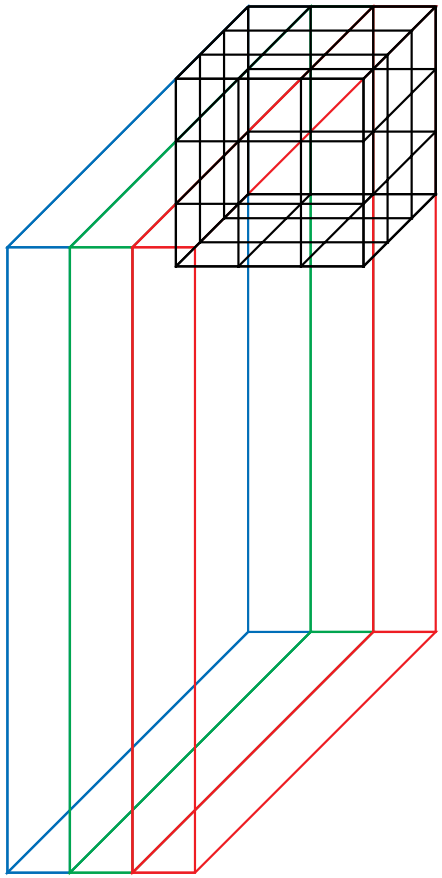
```
Matrix a = ...;
Matrix b = ...;
Matrix c = ...;
scalar s = 7;

if s > 0 {
    d := s * a + b;
} else {
    d := a - b;
}

while s > 0 {
  x = input();
  c = matmul(c, x);
}

result := c;
```

Op Layer

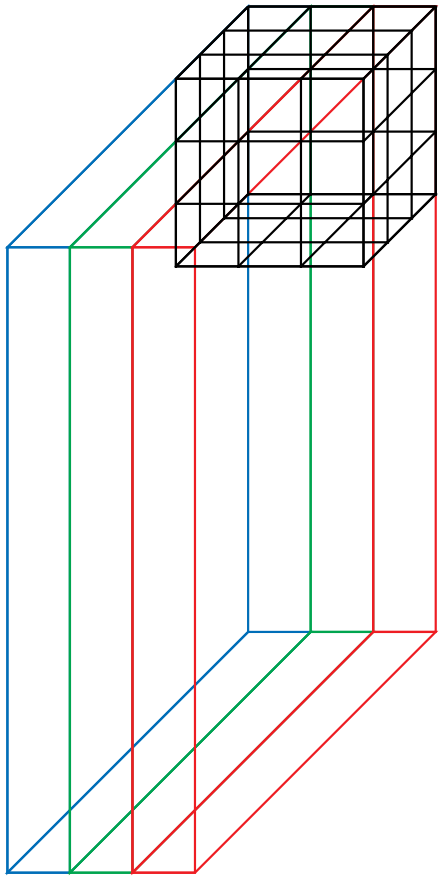


Bob

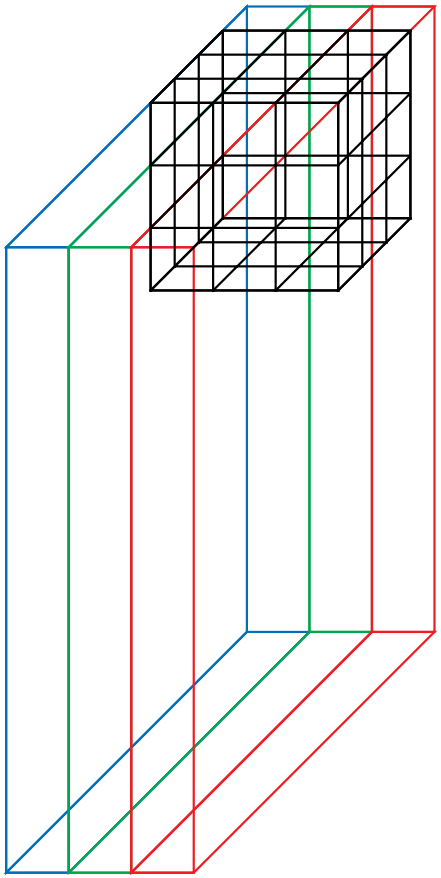
im2col



im2col

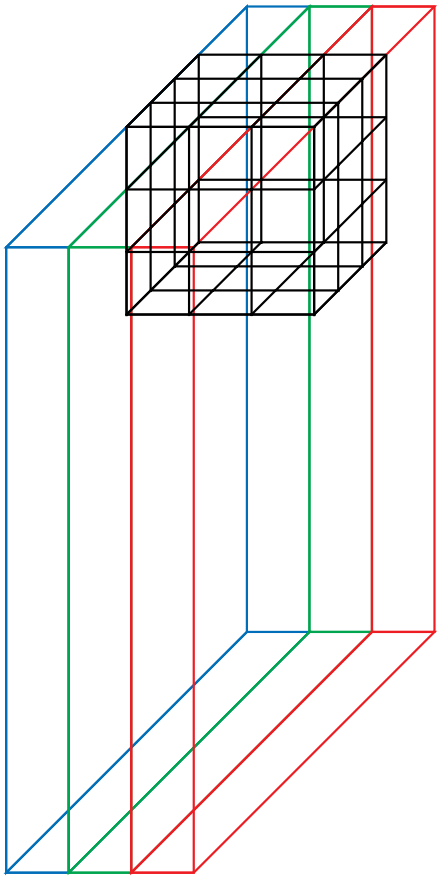


im2col



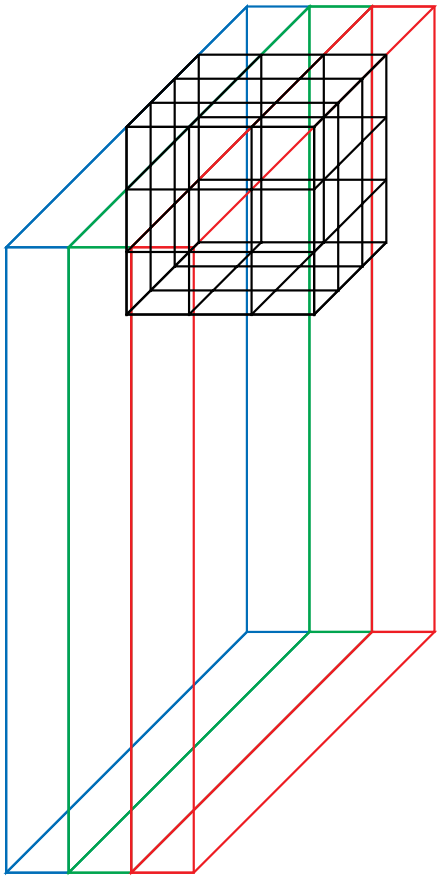
x_0	x_1	x_2	$\cdots$	x_{24}	x_{25}	x_{26}
x_0	x_1	x_2	$\cdots$	x_{24}	x_{25}	x_{26}

im2col



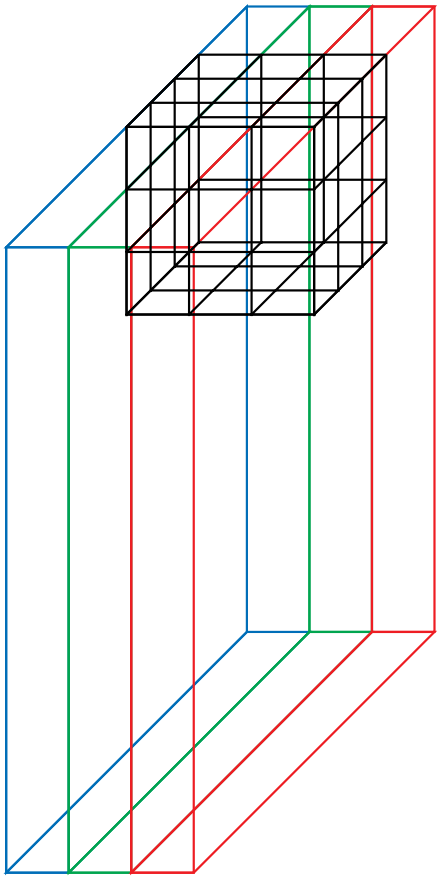
x_0	x_1	x_2	$\cdots$	x_{24}	x_{25}	x_{26}
x_0	x_1	x_2	$\cdots$	x_{24}	x_{25}	x_{26}
x_0	x_1	x_2	$\cdots$	x_{24}	x_{25}	x_{26}

im2col



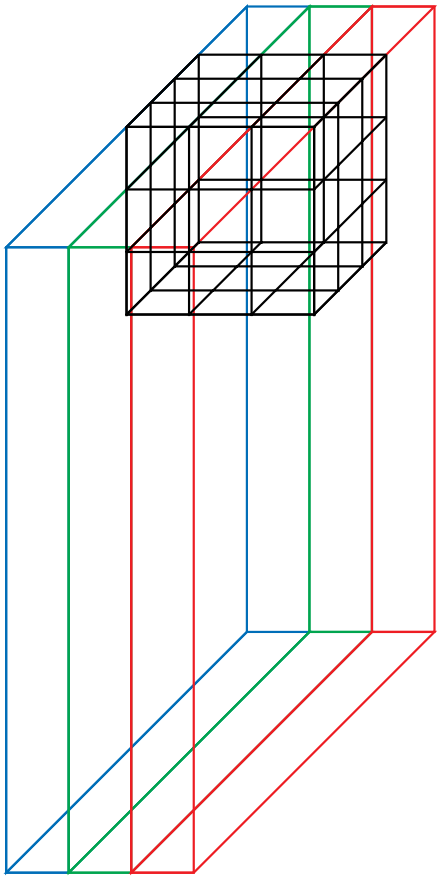
$$
\begin{array}{|c|c|c|c|c|c|c|}
\hline
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
\hline
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
\hline
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
\hline
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
\hline
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
\hline
\end{array}
$$

im2col



$$
\begin{bmatrix}
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26}
\end{bmatrix}
\begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ \vdots \\ k_{24} \\ k_{25} \\ k_{26} \end{bmatrix}
$$

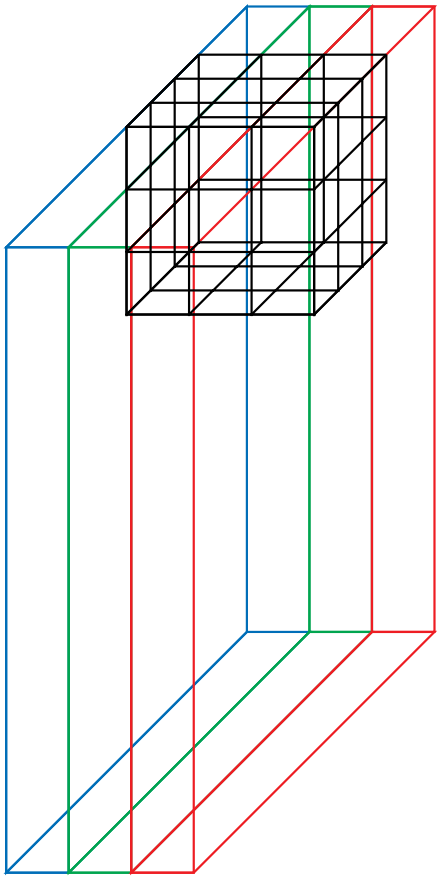
im2col



$$
\begin{array}{|c|c|c|}\hline x_0 & x_1 & x_2 \\ \hline x_0 & x_1 & x_2 \\ \hline x_0 & x_1 & x_2 \\ \hline \vdots & \vdots & \vdots \\ \hline x_0 & x_1 & x_2 \\ \hline \end{array}
\quad \cdots \quad
\begin{array}{|c|c|c|}\hline x_{24} & x_{25} & x_{26} \\ \hline x_{24} & x_{25} & x_{26} \\ \hline x_{24} & x_{25} & x_{26} \\ \hline \vdots & \vdots & \vdots \\ \hline x_{24} & x_{25} & x_{26} \\ \hline \end{array}
$$

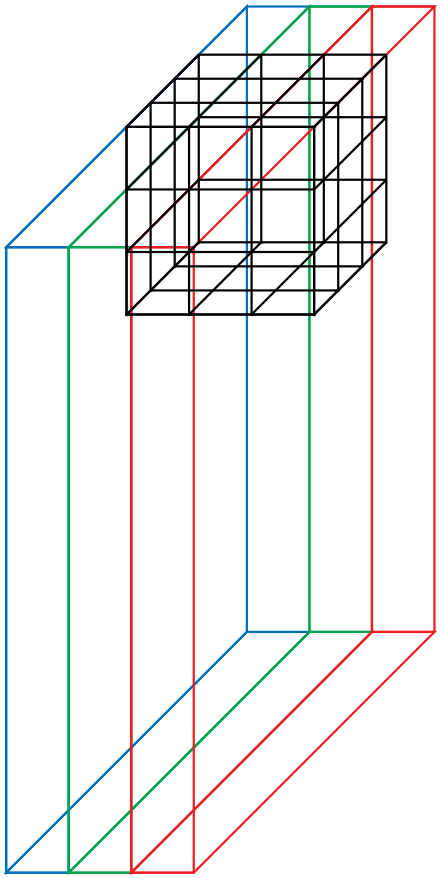
$$
\begin{array}{|c|c|}\hline k_0 & k_0 \\ \hline k_1 & k_1 \\ \hline k_2 & k_2 \\ \hline \vdots & \vdots \\ \hline k_{24} & k_{24} \\ \hline k_{25} & k_{25} \\ \hline k_{26} & k_{26} \\ \hline \end{array}
$$

im2col



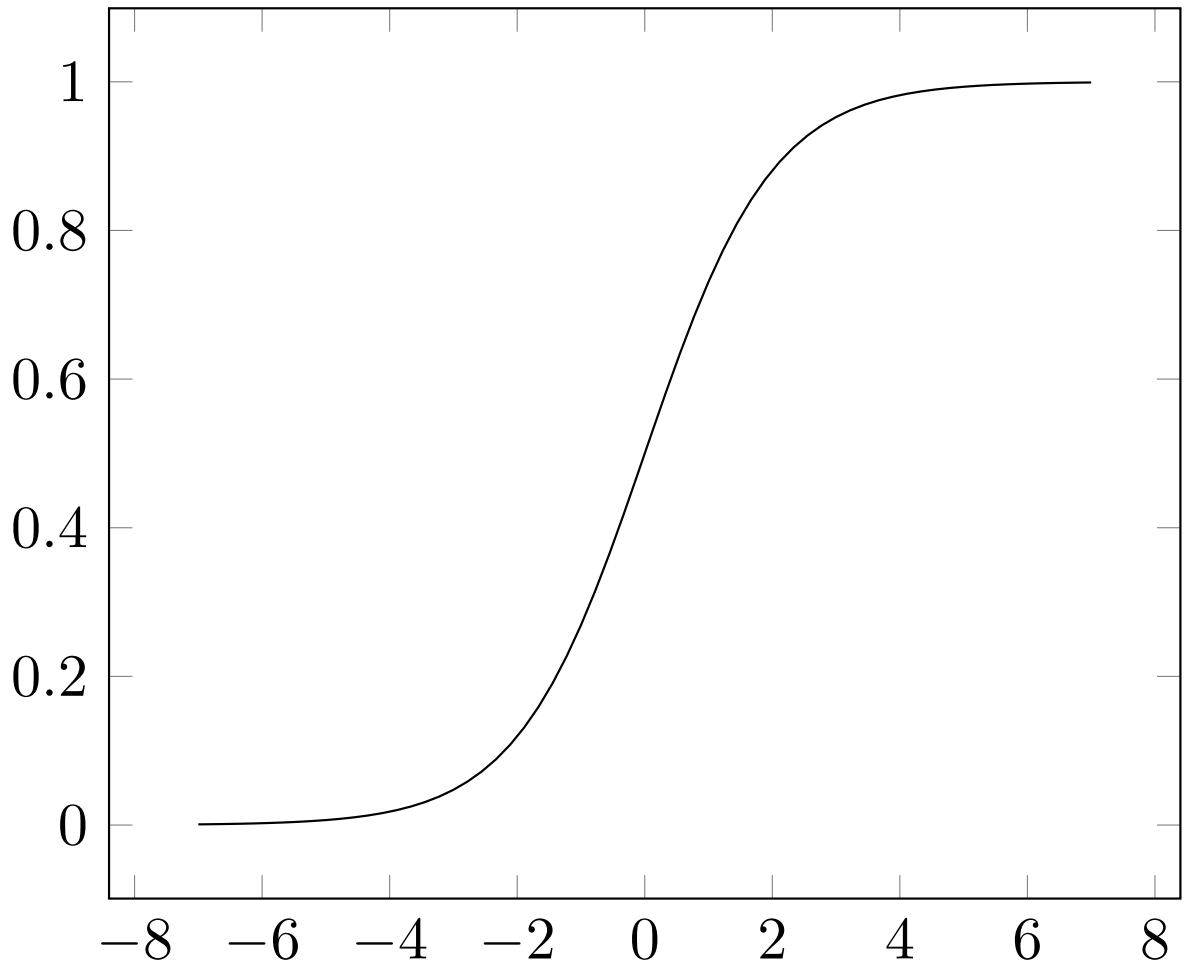
$$
\begin{bmatrix}
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26}
\end{bmatrix}
\begin{bmatrix}
k_0 & k_0 & k_0 \\
k_1 & k_1 & k_1 \\
k_2 & k_2 & k_2 \\
\vdots & \vdots & \vdots \\
k_{24} & k_{24} & k_{24} \\
k_{25} & k_{25} & k_{25} \\
k_{26} & k_{26} & k_{26}
\end{bmatrix}
$$

im2col



$$
\begin{bmatrix}
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
x_0 & x_1 & x_2 & \cdots & x_{24} & x_{25} & x_{26}
\end{bmatrix}
\times
\begin{bmatrix}
k_0 & k_0 & k_0 \\
k_1 & k_1 & k_1 \\
k_2 & k_2 & k_2 \\
\vdots & \vdots & \vdots \\
k_{24} & k_{24} & k_{24} \\
k_{25} & k_{25} & k_{25} \\
k_{26} & k_{26} & k_{26}
\end{bmatrix}
$$

Kernel Layer



Sigmoid

Kernel Layer

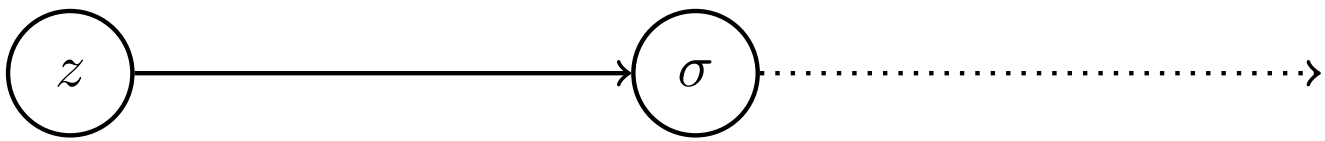
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Forward Pass

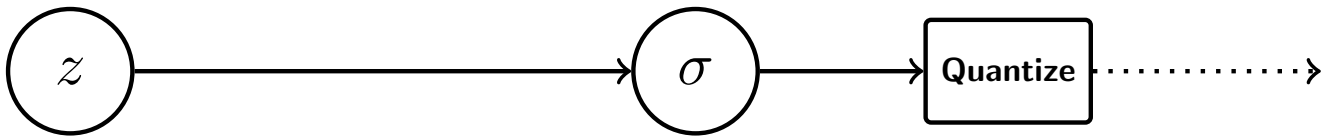
$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

Backward Pass

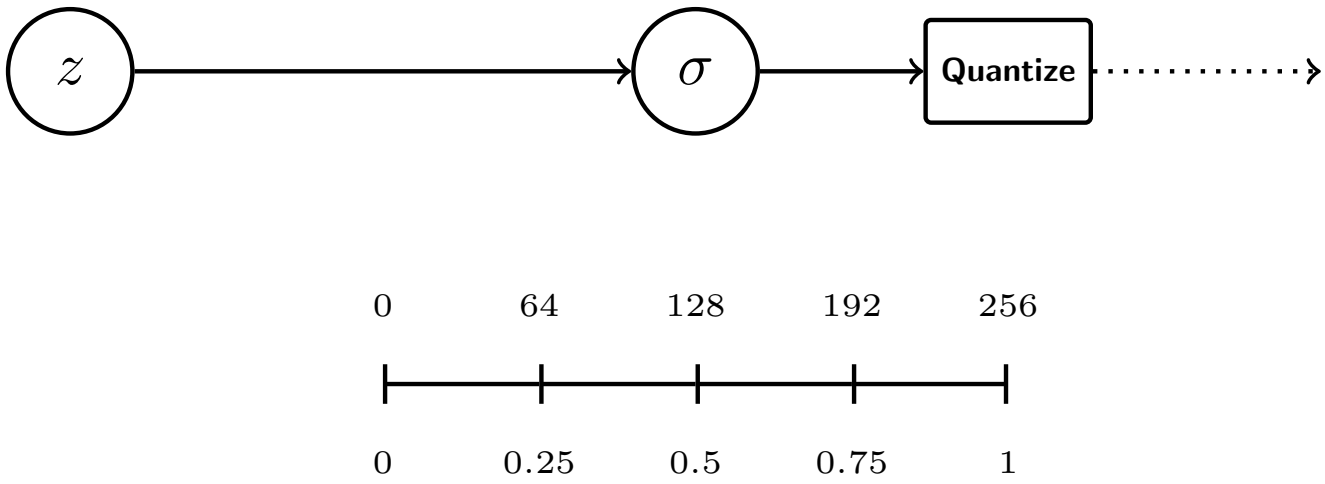
Quantization



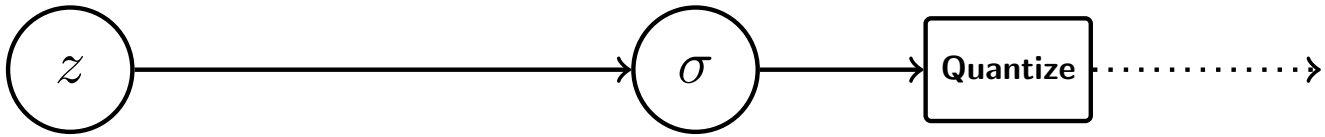
Quantization

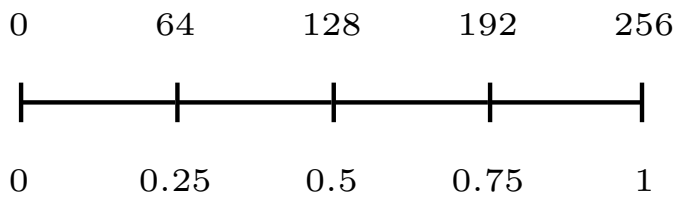


Quantization



Quantization





`github.com/google/gemmlowp`

Hardware Layer

Hardware Layer

CPU

Hardware Layer

CPU

GPU

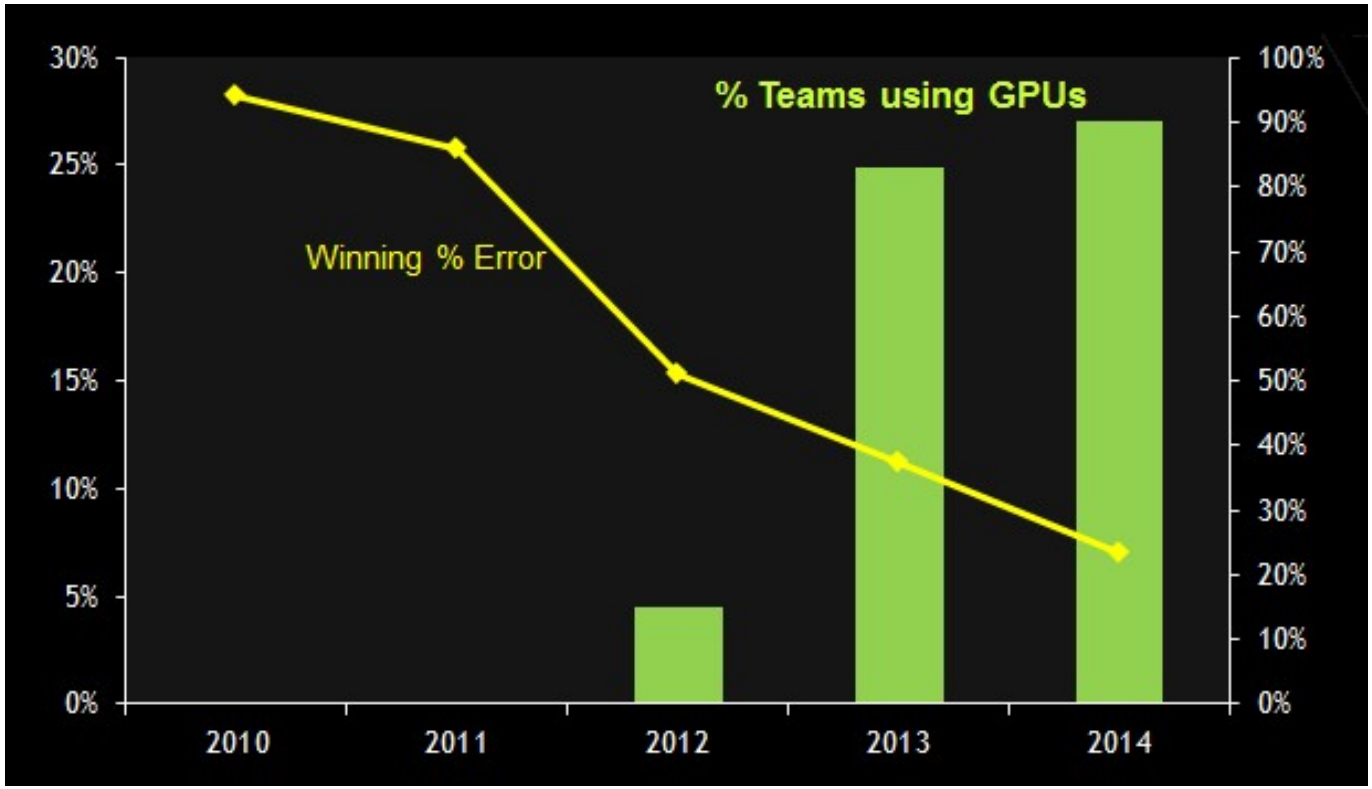
Hardware Layer

CPU

GPU

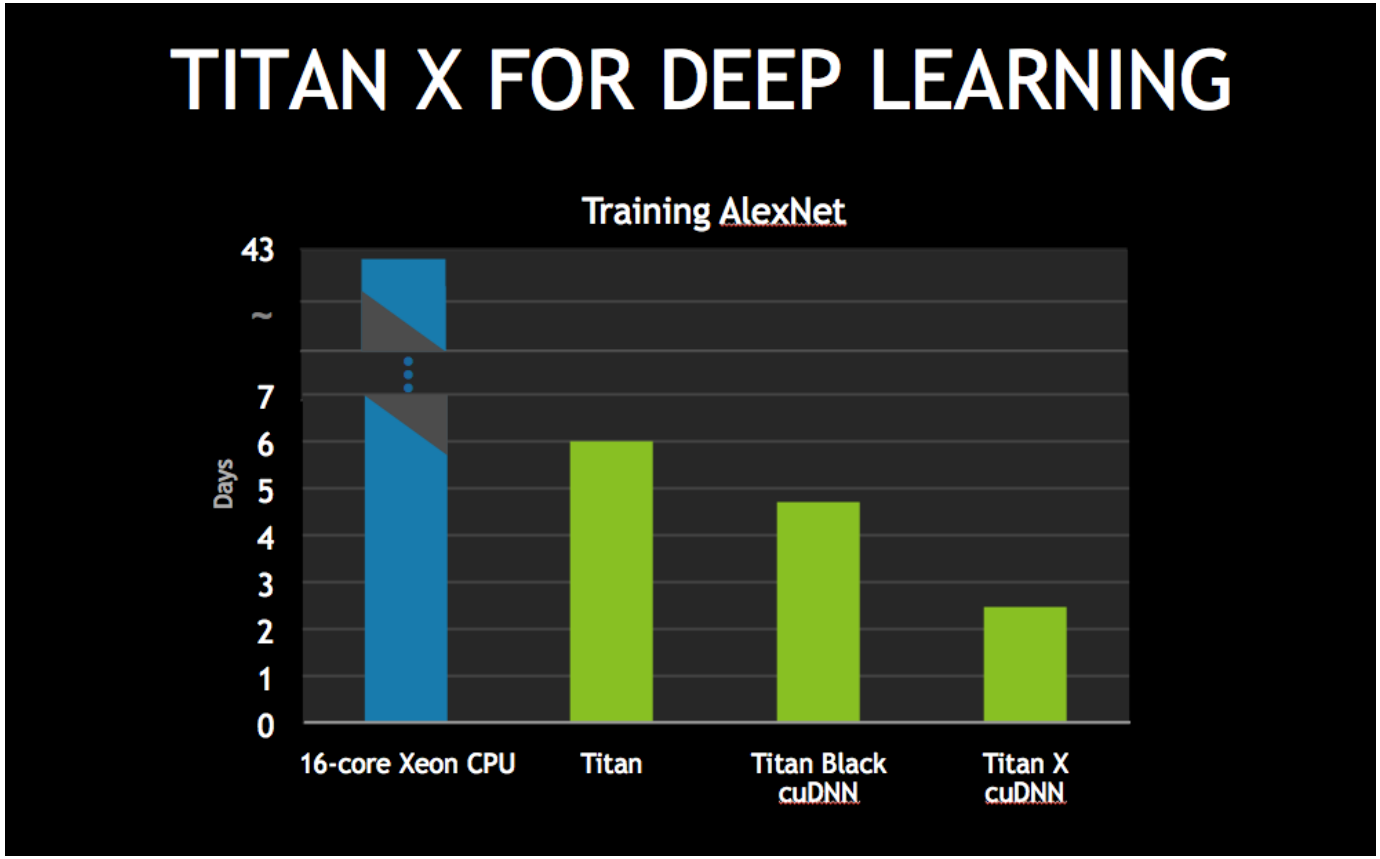
ASIC

GPUs



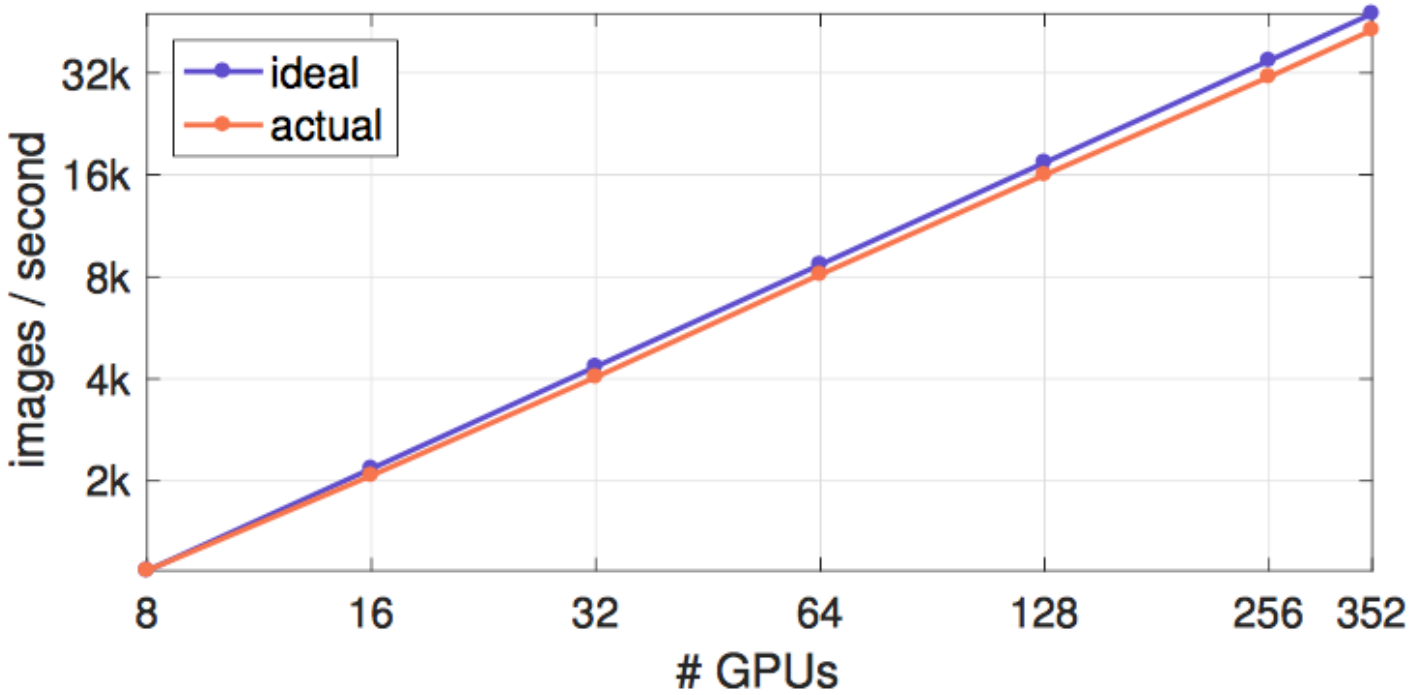
Teams using GPUs at ImageNet Competition

GPUs



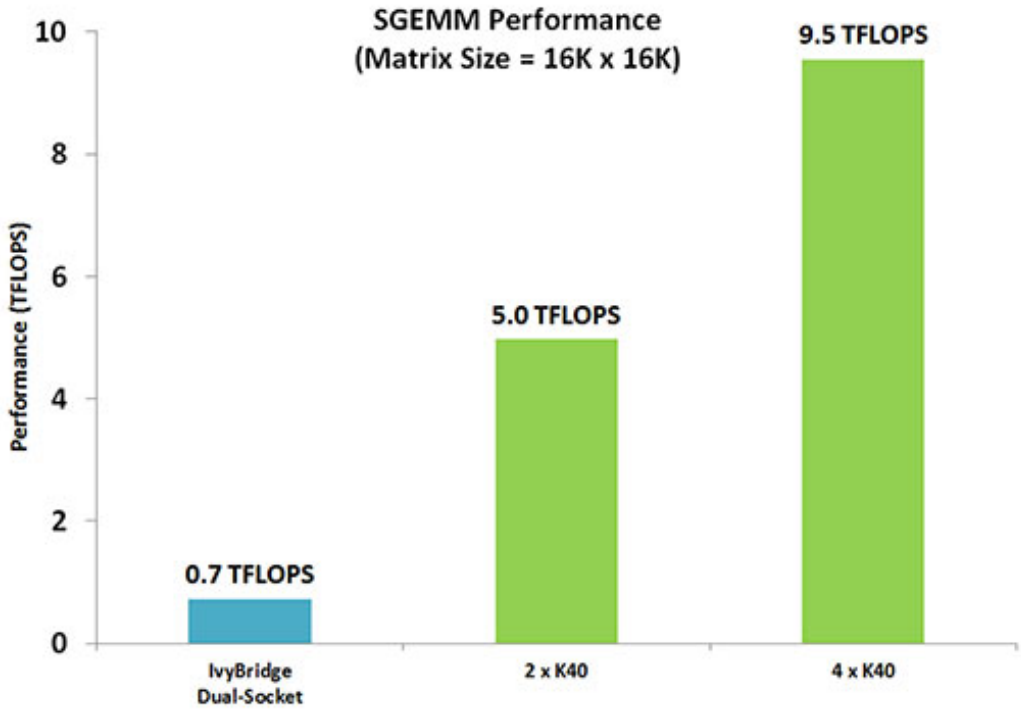
Training Time for ImageNet

GPUs

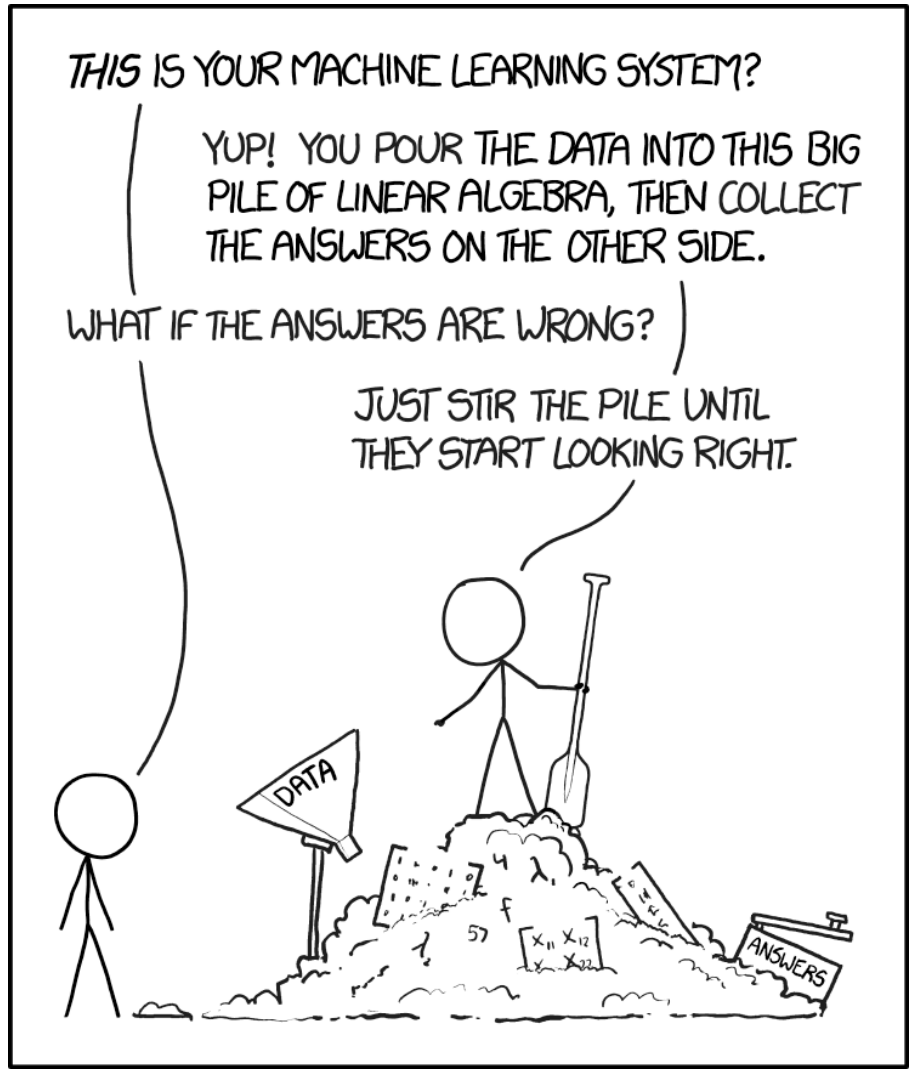


Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour
Goyal et al. (2012)

GPUs



Single Precision General Matrix/Matrix Multiply Performance


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Hardware: GPU

NVIDIA Stock

Deep Learning Trend

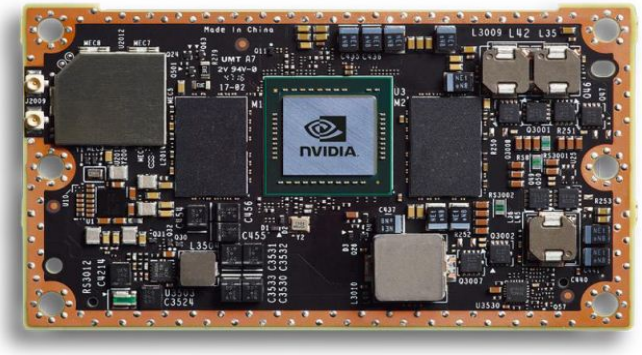
Hardware: GPU

Hardware: GPU

Titan X

Hardware: GPU

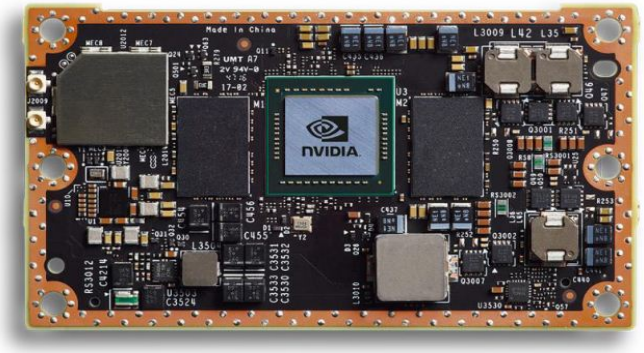
Titan X



Jetson TX2

Hardware: GPU

Titan X



Jetson TX2

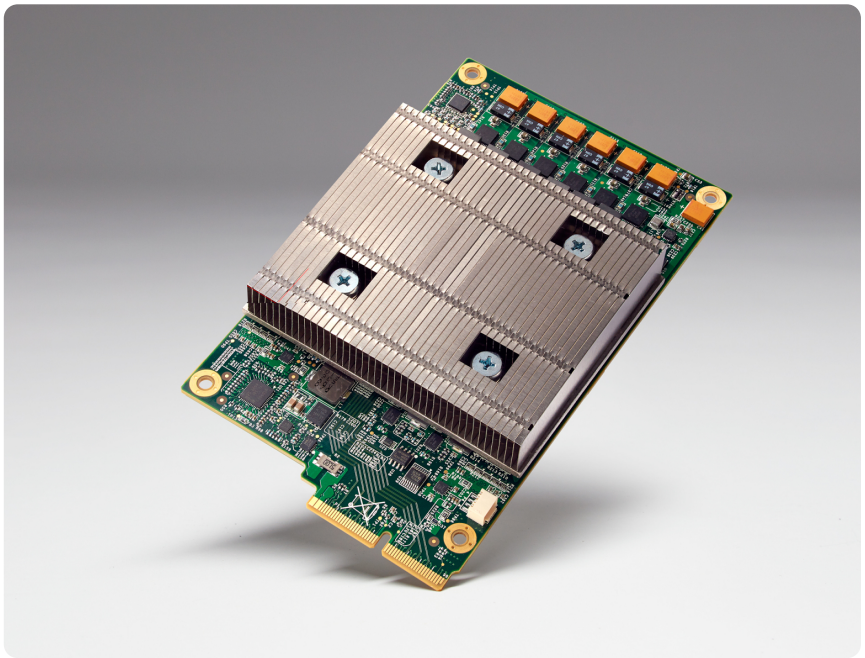
DGX-1

Hardware: Big Basin

Big Basin

- 8 NVIDIA Tesla P100 GPUs
- NVLink (12x faster than PCIe)
- 16 GB RAM
- 10.6 TFLOPS
- Reduced Precision Computing

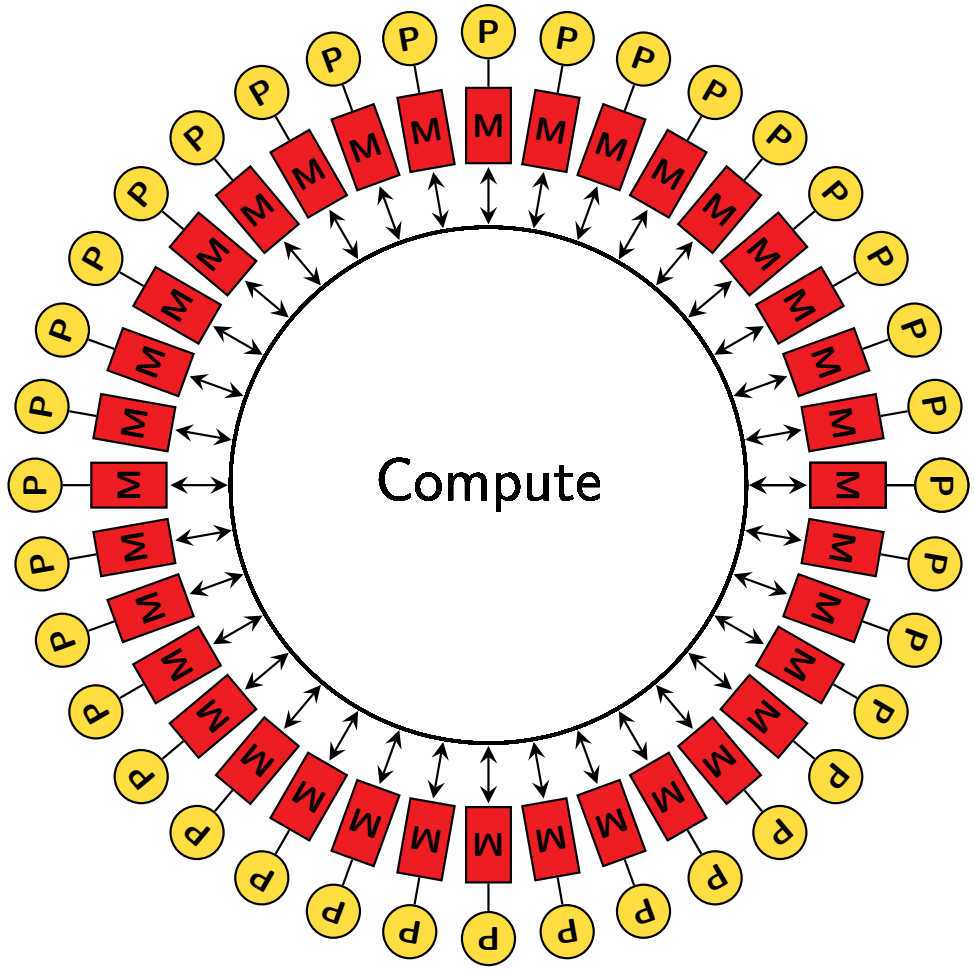
Hardware: TPU



TPU

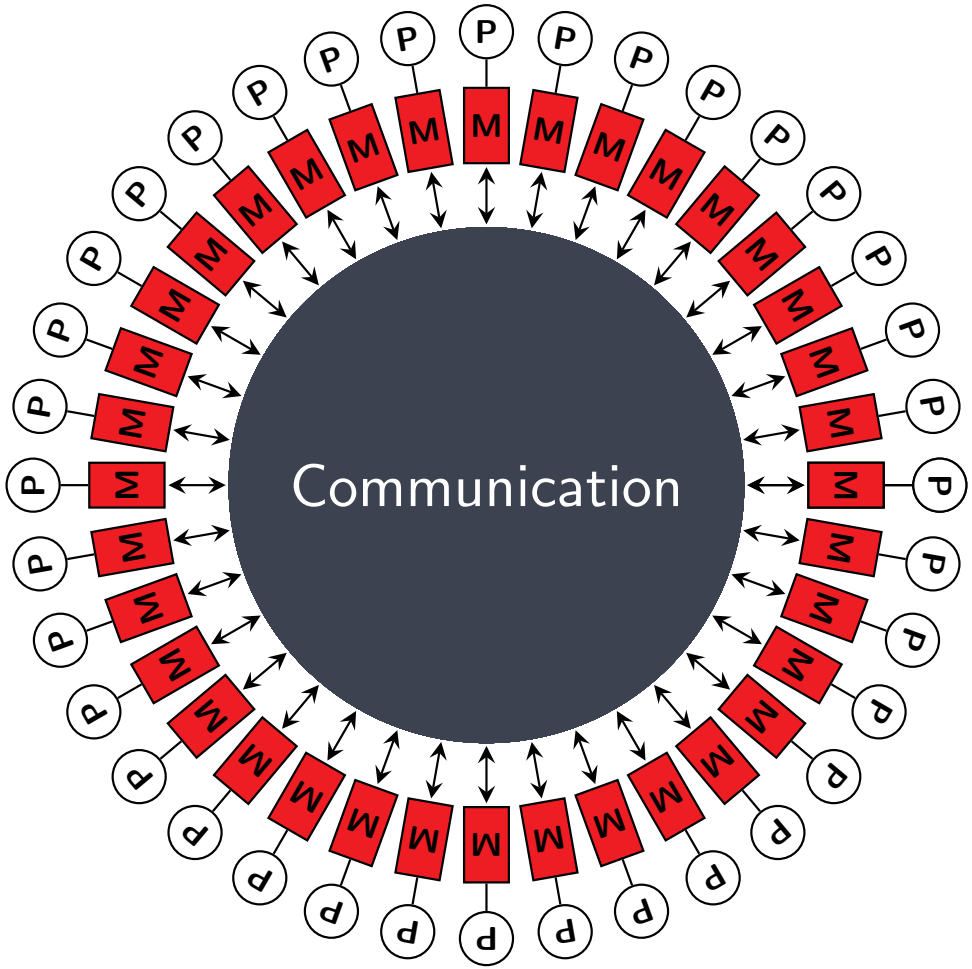
- *Coprocessor*
- 92 TOPS for 8-bit int
- 42 TFLOPS (TPU2)
- 24 MB Memory

Hardware: IPU



Bulk Synchronous Parallelism

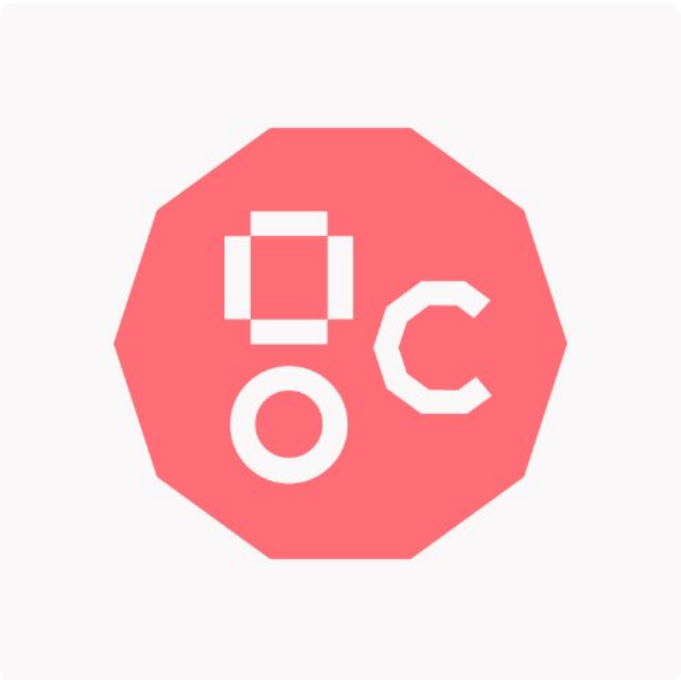
Hardware: IPU



Bulk Synchronous Parallelism

Hardware: IPU

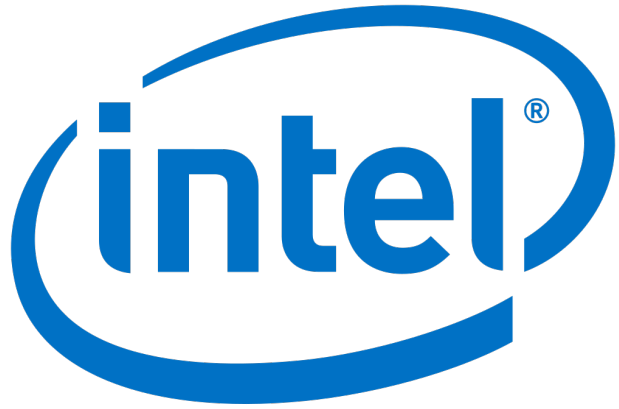
- Startup from Bristol (UK)
- Graphcore Colossus
- TBR later this year
- 1000 Processors/Chip
- Mixed Precision Arithmetic
- No DRAM
- "Thrives on Sparsity"



Graphcore

Kernel Layer

cuDNN/cuBLAS



MKL

Demo: Convolution w/ cuDNN

Demo: TensorFlow Custom Op

Graph Layer: TensorFlow

Graph Layer: TensorFlow

- Google (2015)
- Static Graphs
- GPUs ✓
- Distributed ✓
- C++ API ✗
- Very "Deployable"

Graph Layer: PyTorch/Caffe2

- Facebook (2016)
- Dynamic Graphs
- For Research
- GPUs ✓
- Distributed ✓
- Pure Python

- Facebook (2017)
- Static Graphs
- For Deployment
- GPUs ✓
- Distributed ✓
- C++ API ✓

Graph Layer: MXNet

- Community, then Amazon
- Apache Incubator
- Highly Modular
- GPUs ✓
- Distributed ✓
- APIs for many languages (C++ ✓)

Graph Layer: Others

Demo: MXNet Classifier

Demo: Loading a TensorFlow graph

How do I continue?

Resources

- Machine Learning @ Coursera
- Deep Learning @ Coursera
- Deep Learning (Google) @ Udacity

Resources

- Machine Learning @ Coursera
- Deep Learning @ Coursera
- Deep Learning (Google) @ Udacity

- `mxnet.incubator.apache.org/architecture`
- `github.com/dmlc/nnvm`
- Intro to Parallel Programming (NVIDIA) @ Udacity
- A Tour of TensorFlow, Goldsborough (2016)

Resources

- Machine Learning @ Coursera
- Deep Learning @ Coursera
- Deep Learning (Google) @ Udacity

- `mxnet.incubator.apache.org/architecture`
- `github.com/dmlc/nnvm`
- Intro to Parallel Programming (NVIDIA) @ Udacity
- A Tour of TensorFlow, Goldsborough (2016)

`github.com/peter-can-talk/cppcon-2017`

Q & A