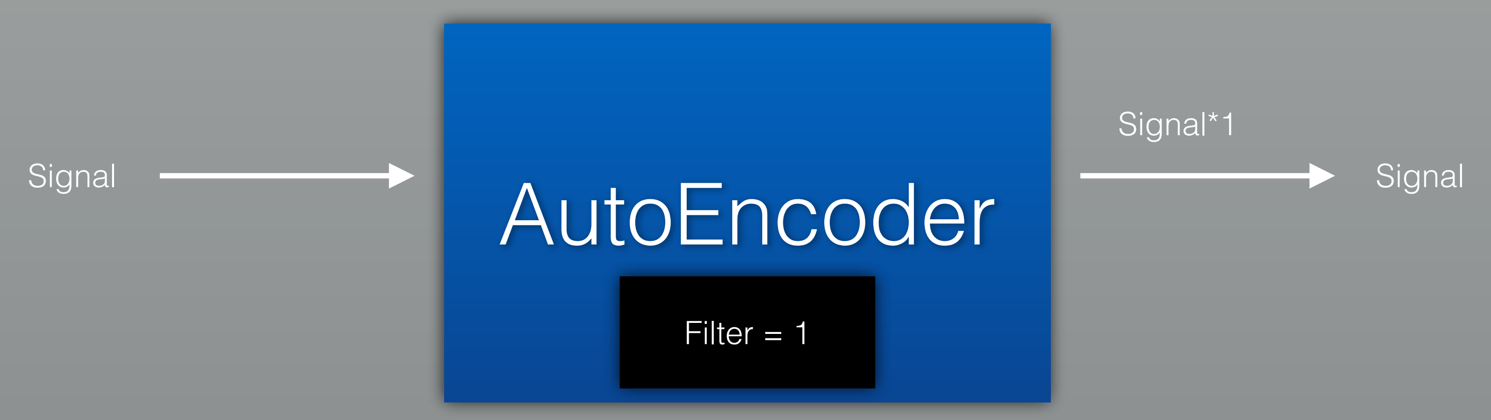
Chapter Notes



Signal

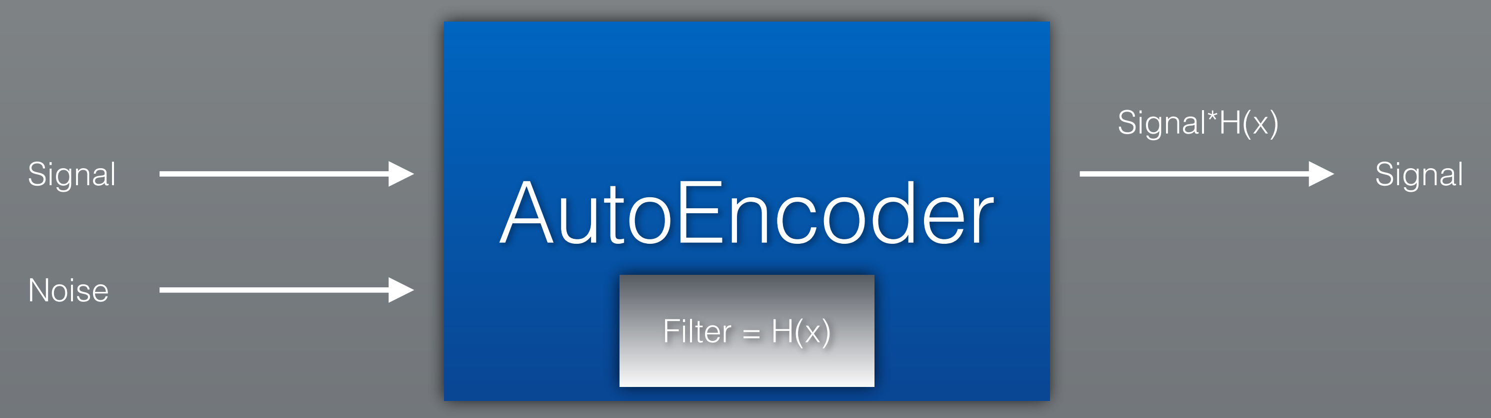
AutoEncoder

Filter = 1

Signal*1

Signal

Model create a passthrough filter. Autoencoder is not constrained, forced to learn anything



Signal

Noise

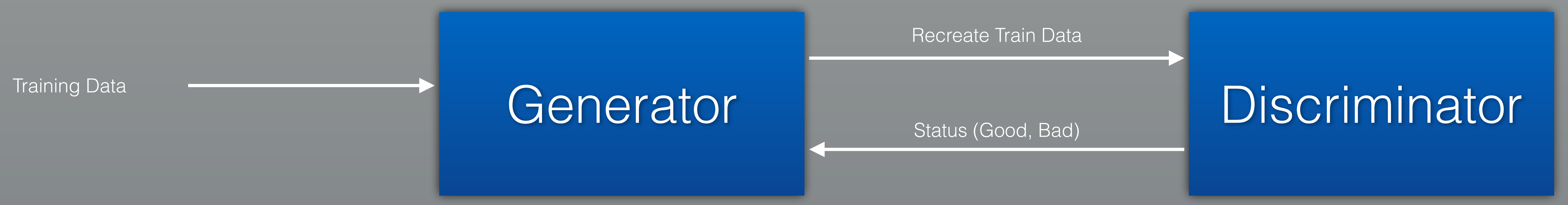
AutoEncoder

Filter = H(x)

Signal*H(x)

Signal

Model learns to generate filter to generate original signal



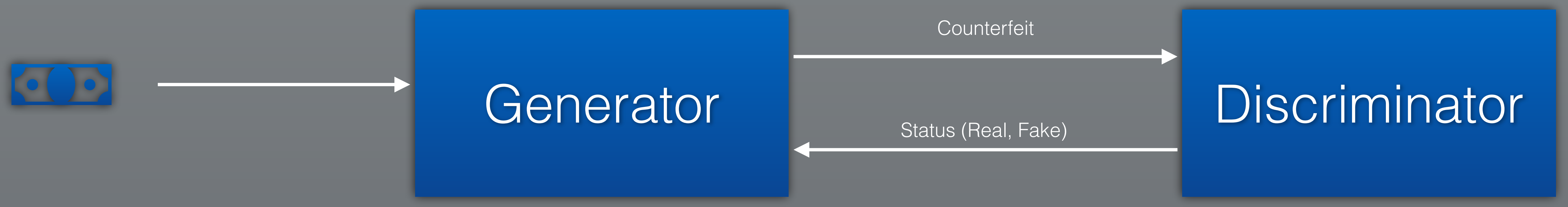
Training Data

Generator

Recreate Train Data

Status (Good, Bad)

Discriminator



Generator

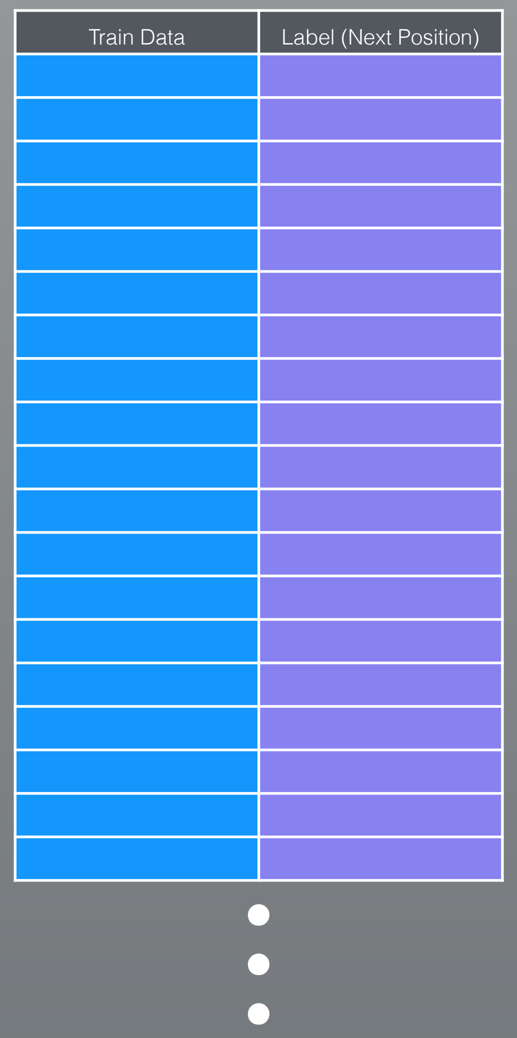
Counterfeit

Status (Real, Fake)

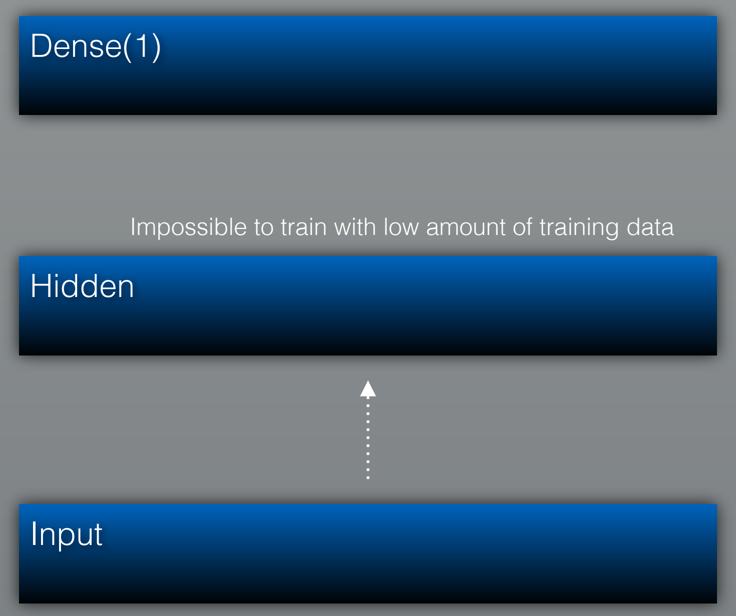
Discriminator

Unsupervised Pretraining Using Stacked AutoEncoders

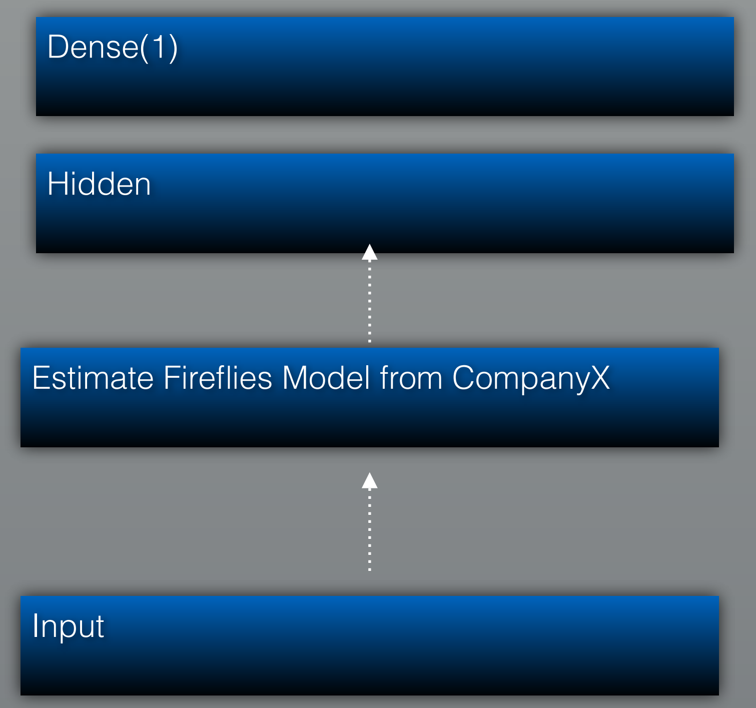
Bee path training data; not enough training



$$n = 1000$$



Use replacement neural network ; a base layer which already 'learned about path of an insect' and train upper layers. Follow best practices when training pre-trained models



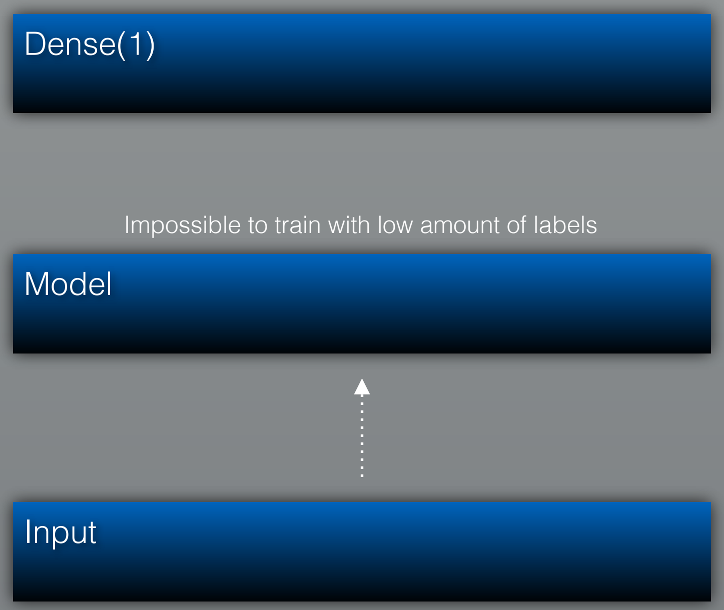
Model gets a learning boost (making limited training data not a big deal) by using network that resolves a similar task.

Unsupervised Pretraining Using Stacked AutoEncoders

Bee path training data; plenty of unlabeled data

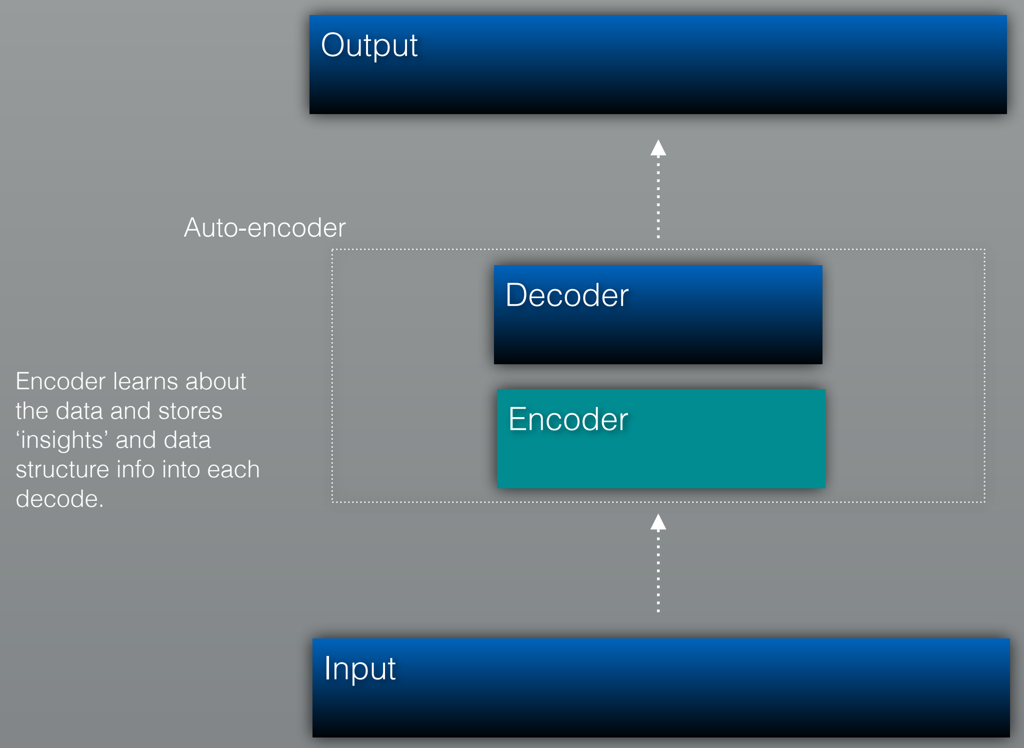
Train Data	Label (Next Position)
🟦	🟪
🟦	
🟦	🟪
🟦	🟪
🟦	
🟦	🟪
🟦	
🟦	
🟦	
🟦	
🟦	🟪
🟦	
🟦	
🟦	🟪
🟦	
🟦	
🟦	
🟦	🟪
🟦	
🟦	

⋮



Impossible to train with low amount of labels

Train AutoEncoder, learn 'codings' from **ALL** training data

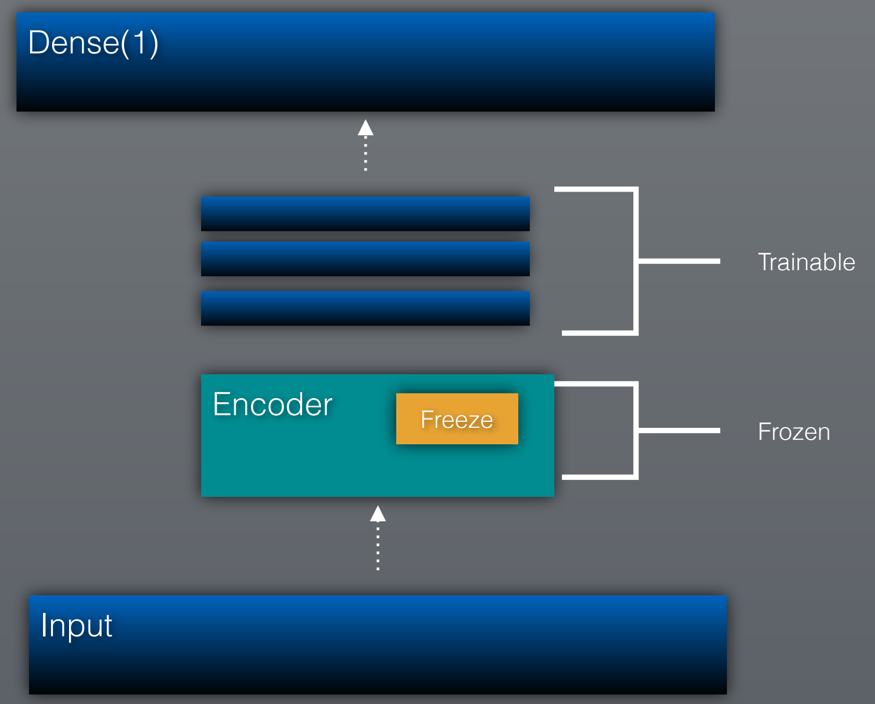


Encoder learns about the data and stores 'insights' and data structure info into each decode.

Train Data	Label (Next Position)
🟦	🟪
🟦	🟪
🟦	🟪
🟦	🟪
🟦	🟪
🟦	🟪
🟦	🟪

⋮

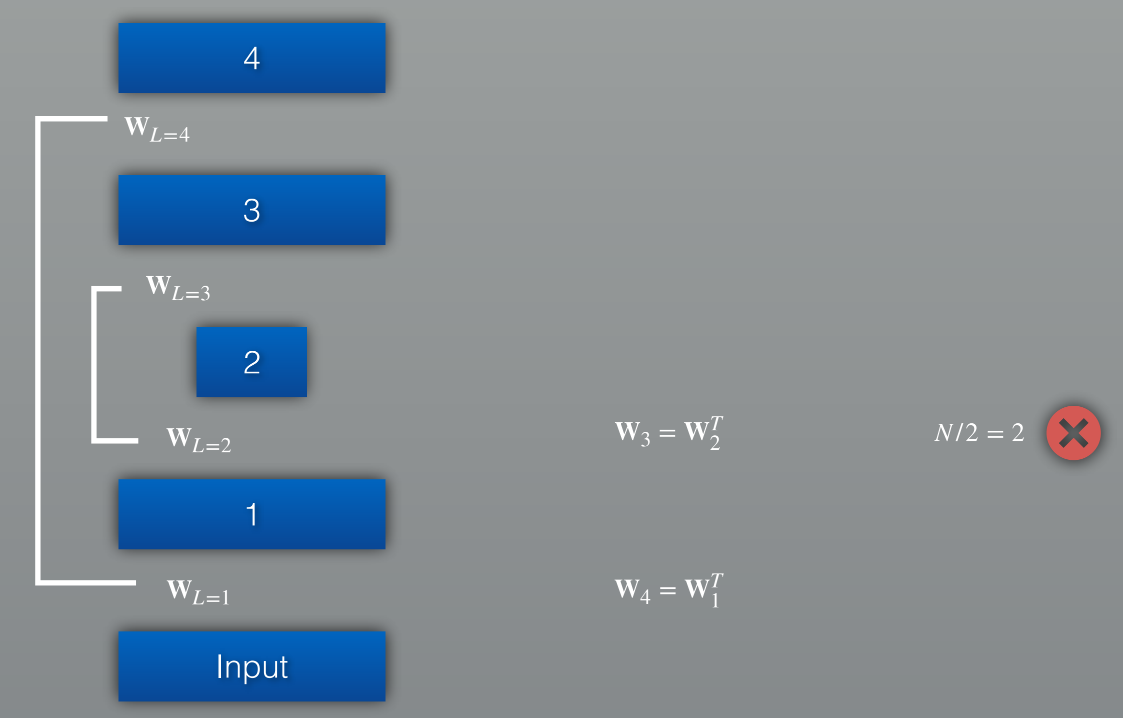
Use encoder to **train** task (i.e. Bee Path) using **labeled data** only. After training the model should generalize well when inferring on unlabeled data.



Tying Weights

$N_{layers=4}$

$\mathbf{W}_{N-L+1} = \mathbf{W}_L^T$

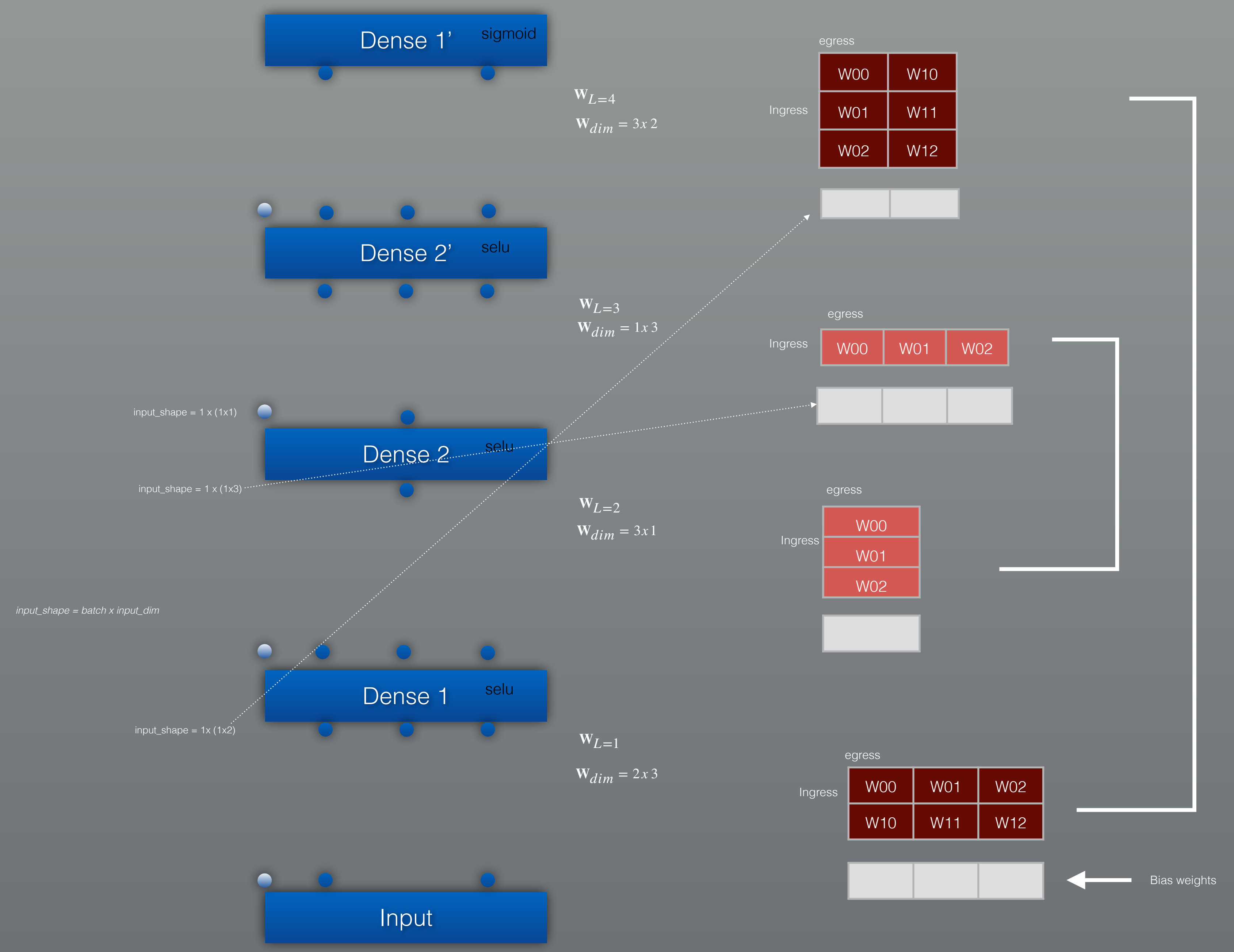


$\mathbf{W}_3 = \mathbf{W}_2^T$

$N/2 = 2$ ☒

$\mathbf{W}_4 = \mathbf{W}_1^T$

Model trains 2 weight matrices instead of 4 if they were not ‘common’



$\mathbf{W}_{L=4}$
$\mathbf{W}_{dim} = 3x\,2$

W00	W10
W01	W11
W02	W12

$\mathbf{W}_{L=3}$
$\mathbf{W}_{dim} = 1x\,3$

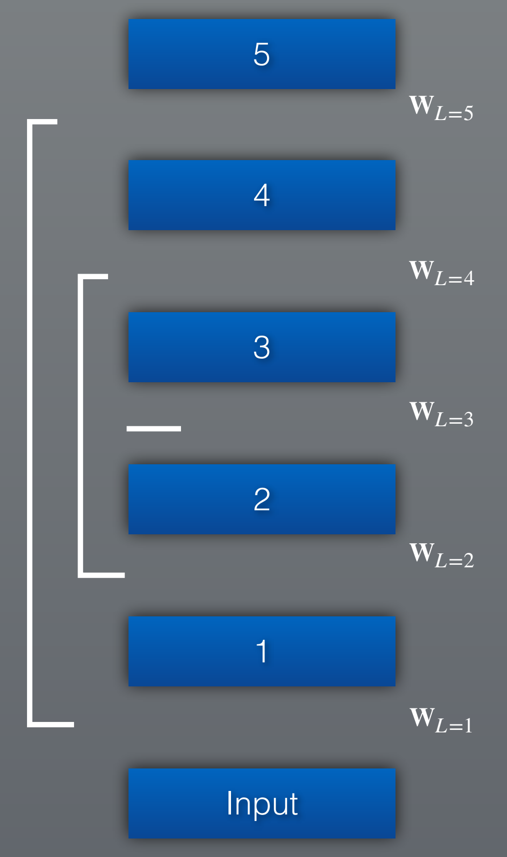
W00	W01	W02

$\mathbf{W}_{L=2}$
$\mathbf{W}_{dim} = 3x\,1$

W00
W01
W02

$\mathbf{W}_{L=1}$
$\mathbf{W}_{dim} = 2x\,3$

W00	W01	W02
W10	W11	W12



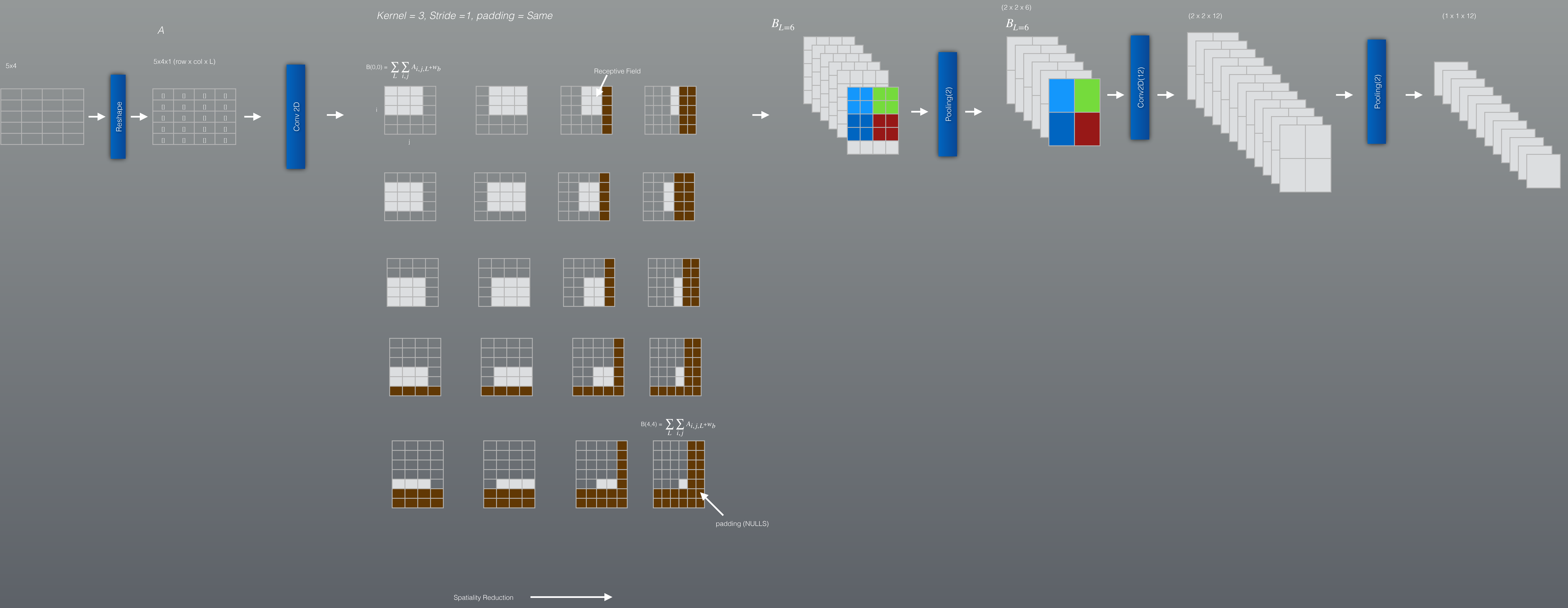
$\mathbf{W}_{L=5}$ $\mathbf{W}_1 = \mathbf{W}_5^T$

$\mathbf{W}_{L=4}$ $\mathbf{W}_2 = \mathbf{W}_4^T$

$\mathbf{W}_{L=3}$ $\mathbf{W}_3 = \mathbf{W}_3^T$ $N/2 = 2$ ☒

$\mathbf{W}_{L=2}$ $\mathbf{W}_4 = \mathbf{W}_2^T$

$\mathbf{W}_{L=1}$ $\mathbf{W}_5 = \mathbf{W}_1^T$



Kernel = 3, Stride =1, padding = Same

A

5x4

Reshape

5x4x1 (row x col x L)

Conv 2D

B(0,0) = $\sum_{L} \sum_{i,j} A_{i,j,L} + w_b$

i

j

Receptive Field

B(4,4) = $\sum_{L} \sum_{i,j} A_{i,j,L} + w_b$

padding (NULLS)

Spatiality Reduction

$B_{L=6}$

Pooling(2)

(2 x 2 x 6)

$B_{L=6}$

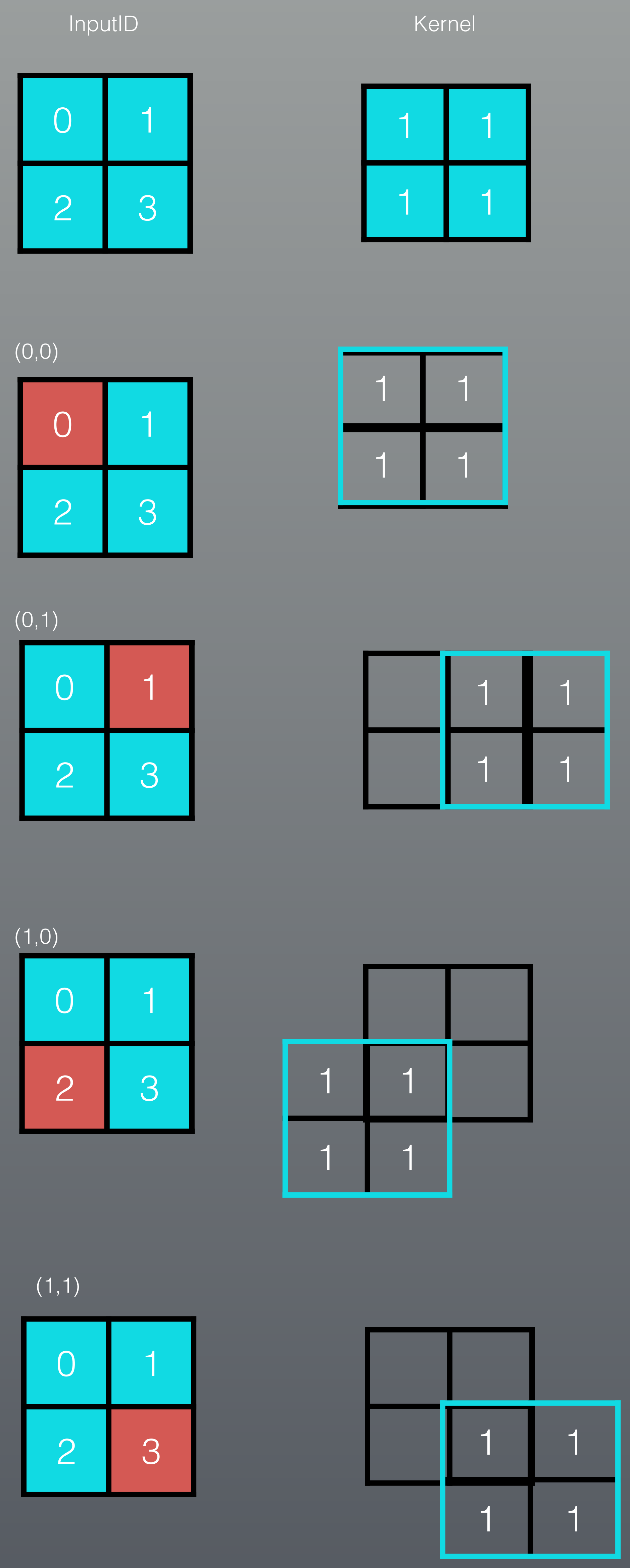
Conv2D(12)

(2 x 2 x 12)

Pooling(2)

(1 x 1 x 12)

CONV2DTranpose (Strides = 1, Kernel_size= 2)



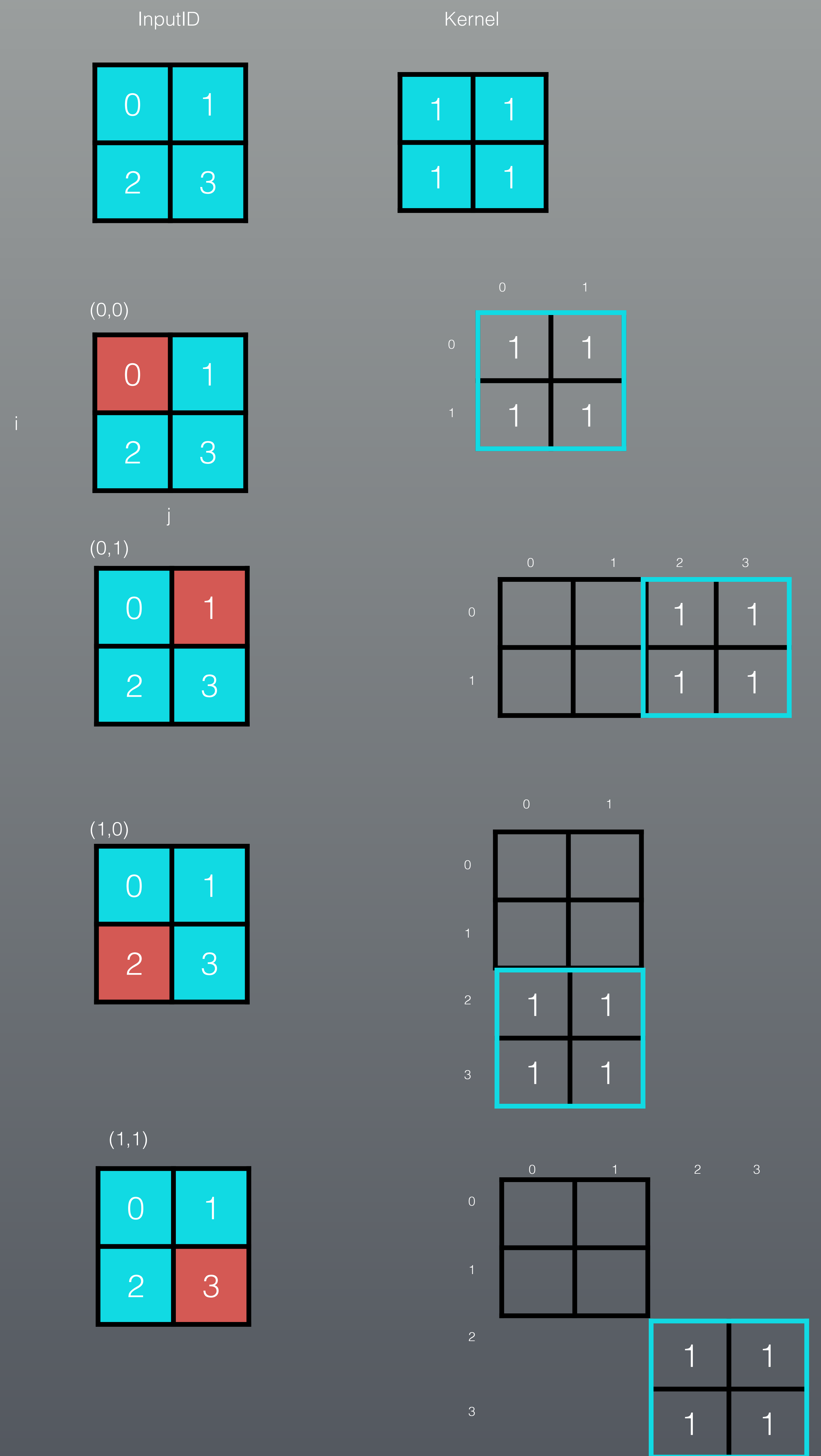
1	2	1
2	4	2
1	2	1

Max index

$i \times s_i$ to $\underbrace{i \times s_i + k_{hsize} - 1}$ i_max_index = 2

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 2

CONV2DTranpose (Strides = 2, Kernel_size= 2)



1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 3

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 3

CONV2DTranpose (Strides = 2, Kernel_size= 3, same=valid)

$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 6

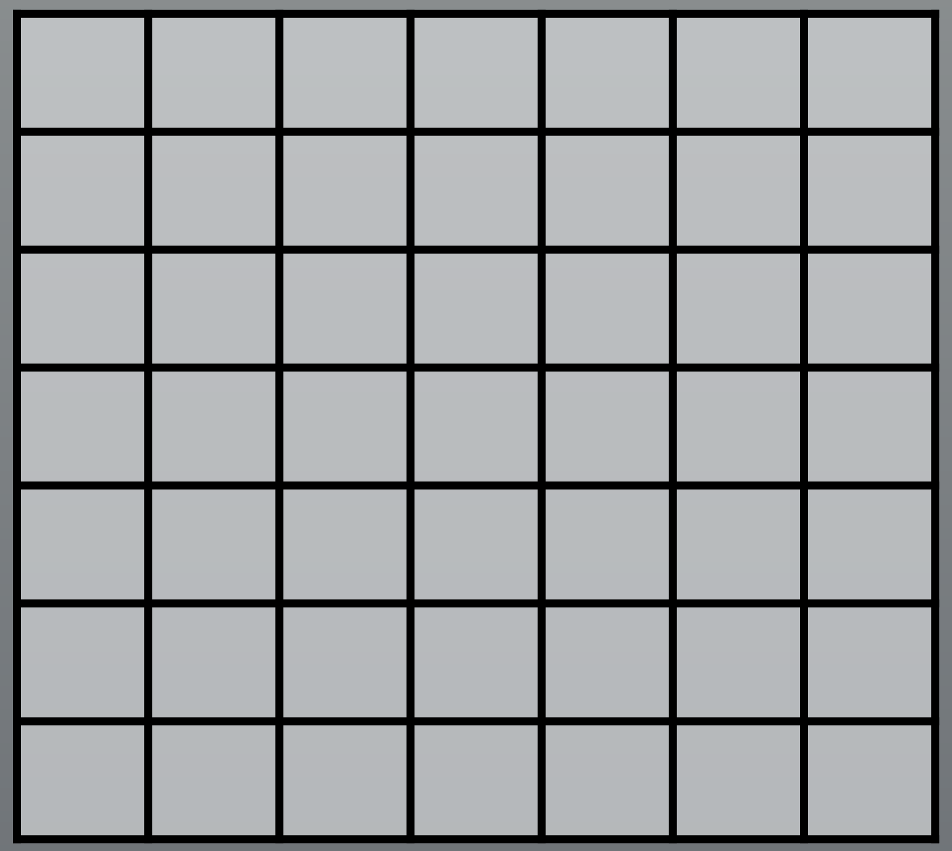
$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 6

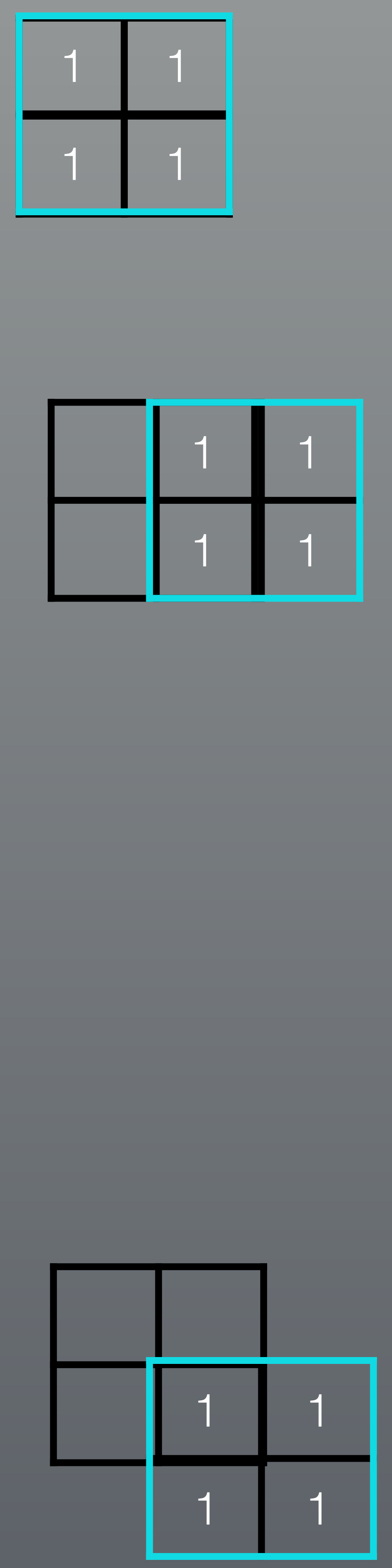
InputID

Kernel

1	1	1
1	1	1
1	1	1

Output Dimensions (Upsample)





CONV2DTranpose (Strides = 2, Kernel_size= 3, same=valid)

InputID

0	1	2
3	4	5
6	7	8

Kernel

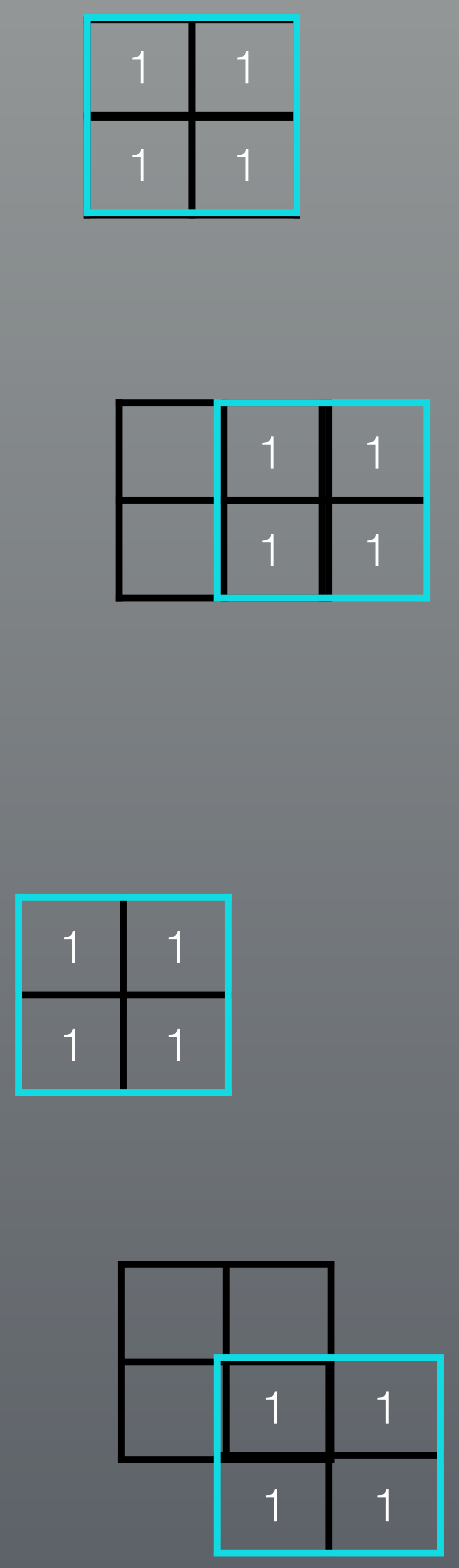
1	1	1
1	1	1
1	1	1

$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 6

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 6

0	1	2
3	4	5
6	7	8

1	1	1				
1	1	1				
1	1	1				



CONV2DTranpose (Strides = 2, Kernel_size= 3)

InputID

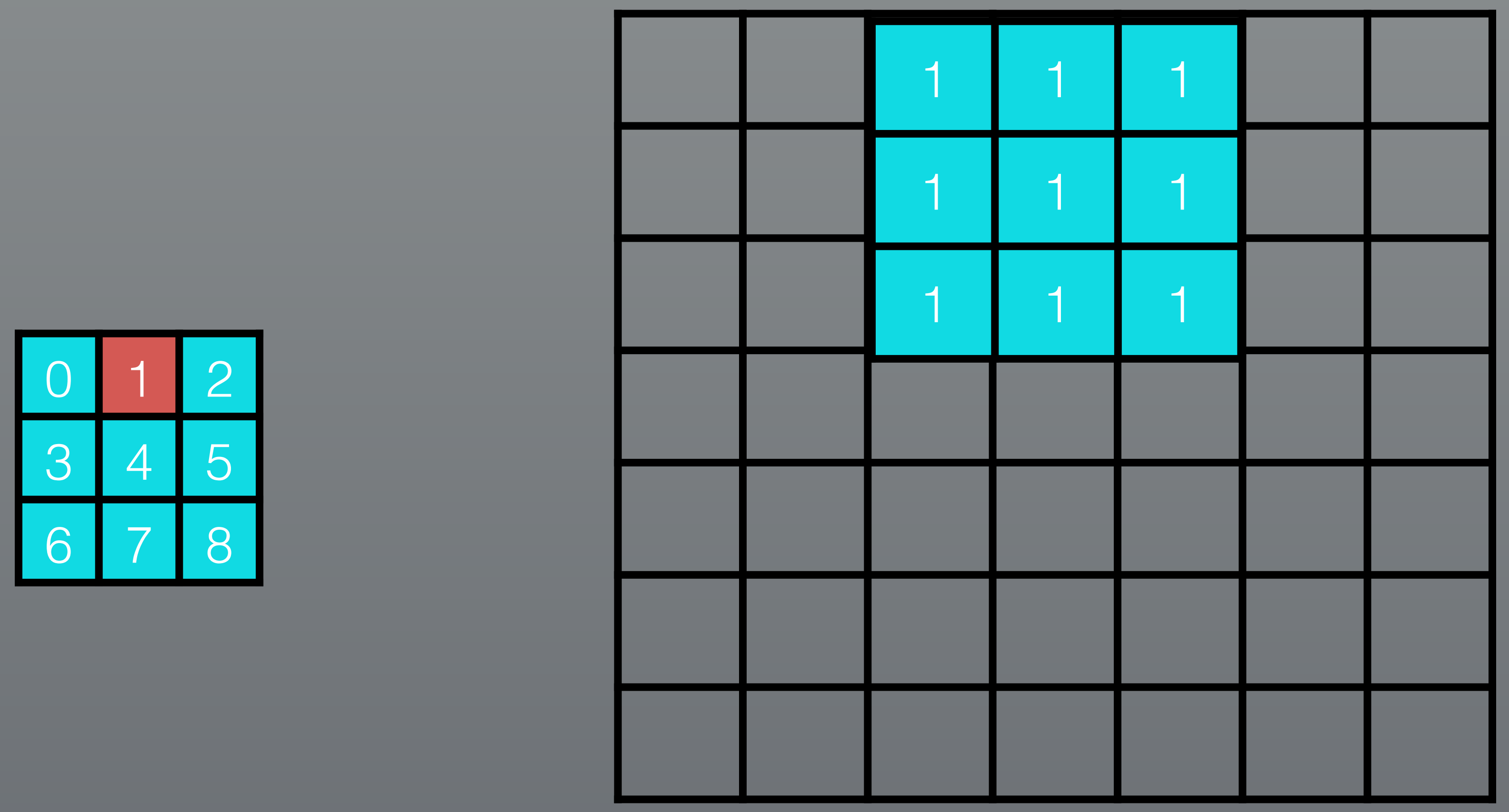
0	1	2
3	4	5
6	7	8

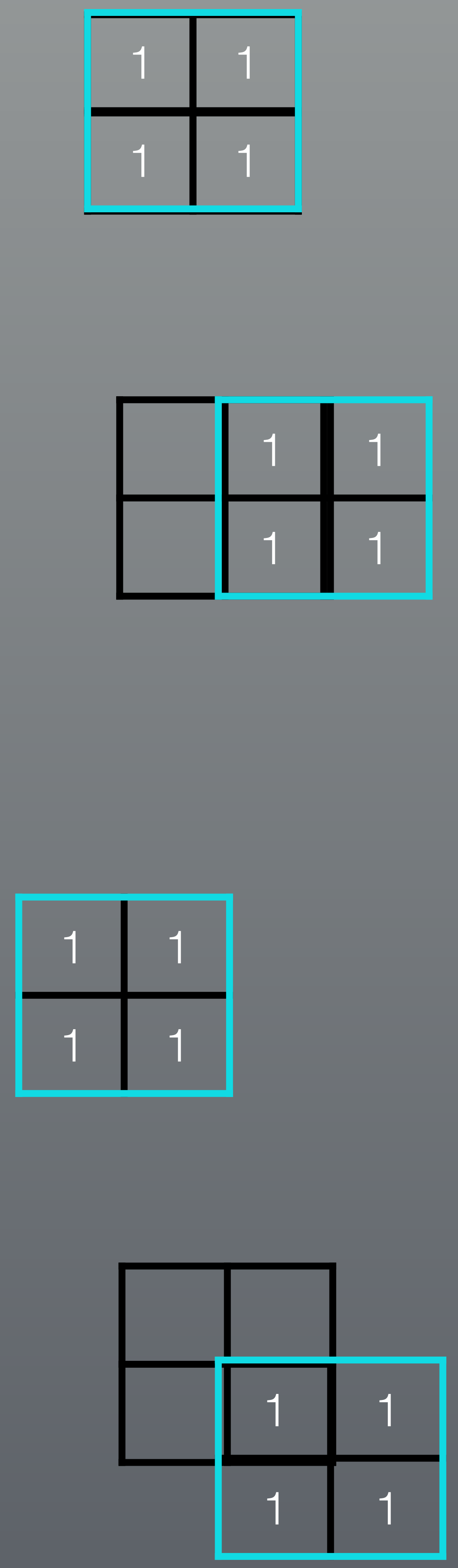
Kernel

1	1	1
1	1	1
1	1	1

$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 6

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 6





CONV2DTranpose (Strides = 2, Kernel_size= 3)

InputID

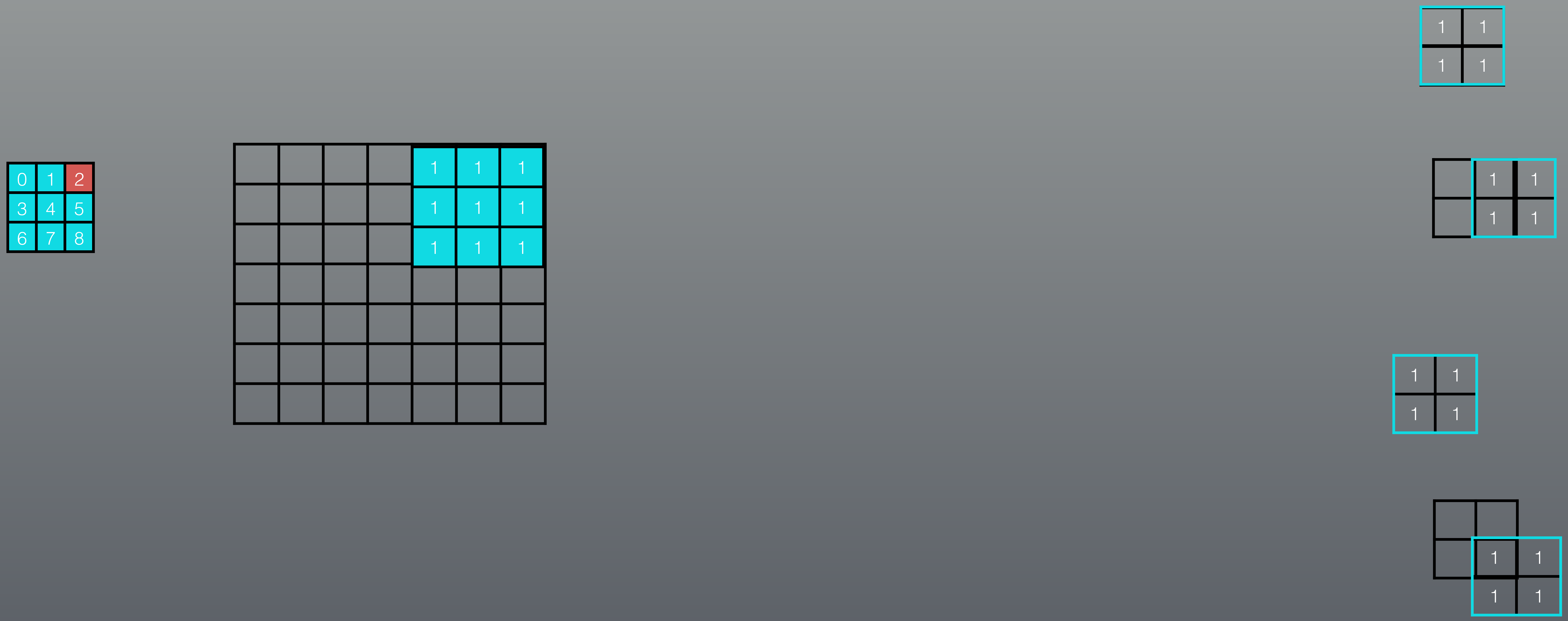
0	1	2
3	4	5
6	7	8

Kernel

1	1	1
1	1	1
1	1	1

$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 6

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 6



CONV2DTranpose (Strides = 2, Kernel_size= 3)

InputID

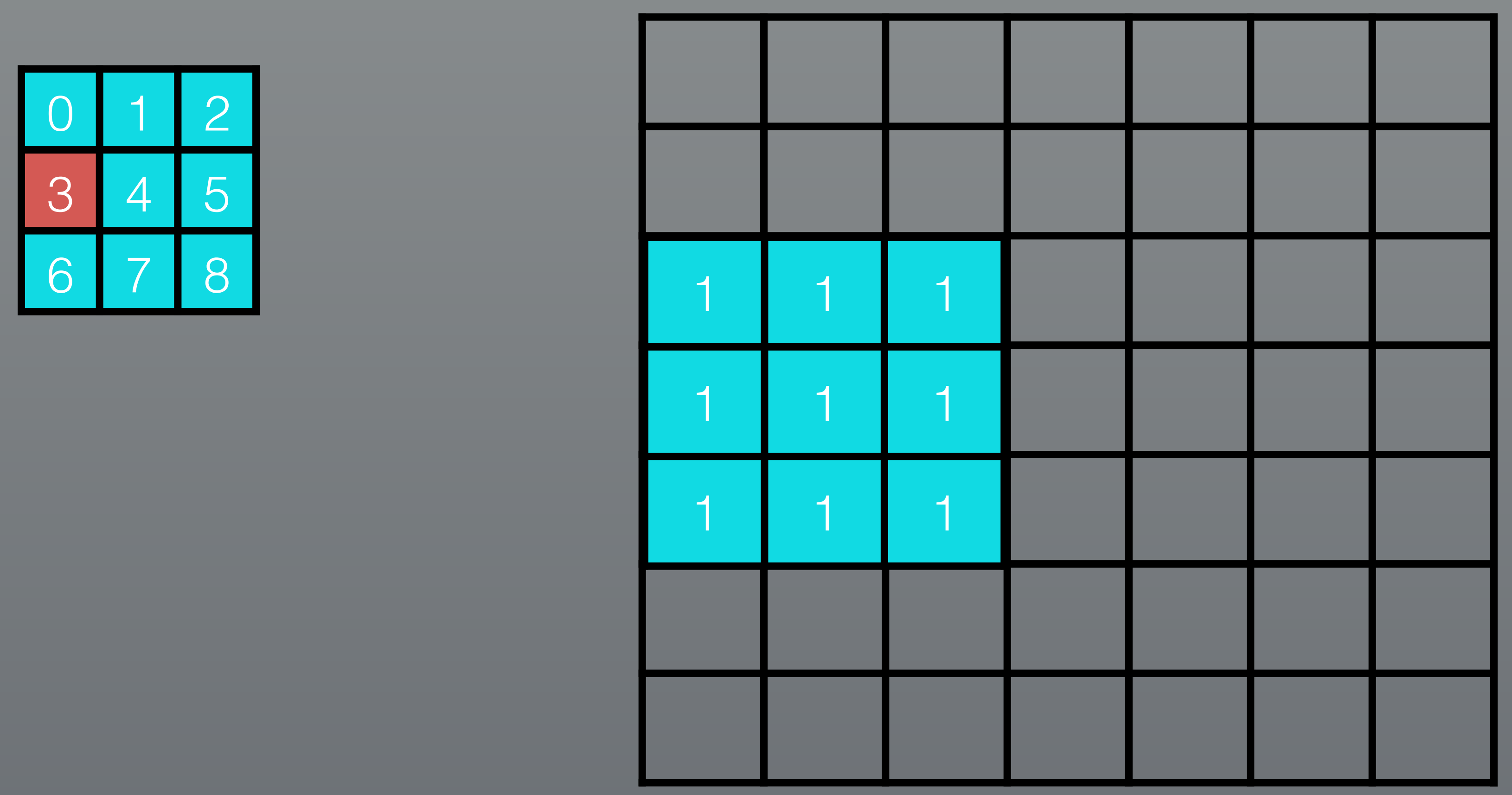
0	1	2
3	4	5
6	7	8

Kernel

1	1	1
1	1	1
1	1	1

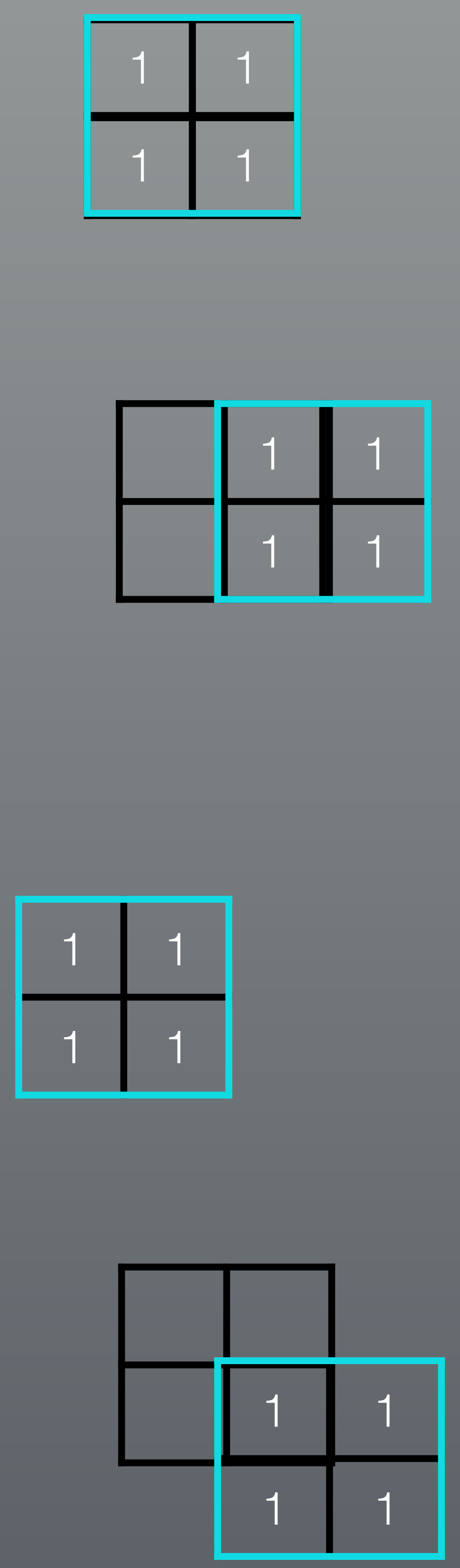
$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 6

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 6



0	1	2
3	4	5
6	7	8

1	1	1				
1	1	1				
1	1	1				



CONV2DTranpose (Strides = 2, Kernel_size= 3)

InputID

0	1	2
3	4	5
6	7	8

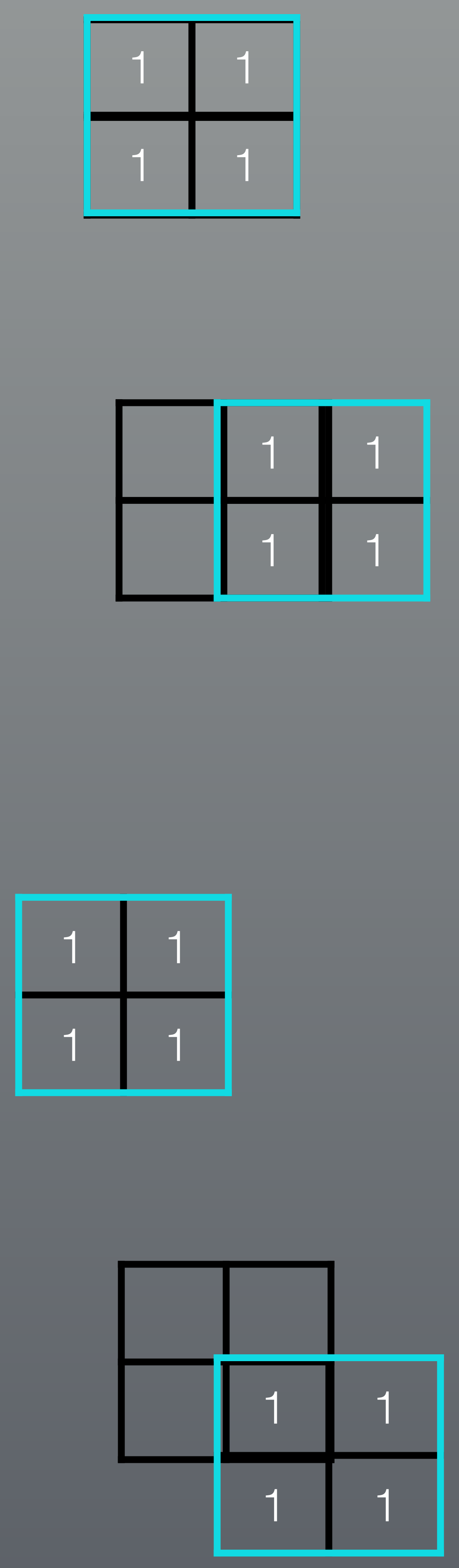
Kernel

1	1	1
1	1	1
1	1	1

$$\begin{array}{l} i \times s_i \text{ to } \overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}} \\ j \times s_j \text{ to } j \times s_j + k_{wsize} - 1 \end{array}$$

i_max_index = 6

j_max_index = 6



0	1	2
3	🟥 4	5
6	7	8

		1	1	1		
		1	1	1		
		1	1	1		

CONV2DTranpose (Strides = 2, Kernel_size= 3)

InputID

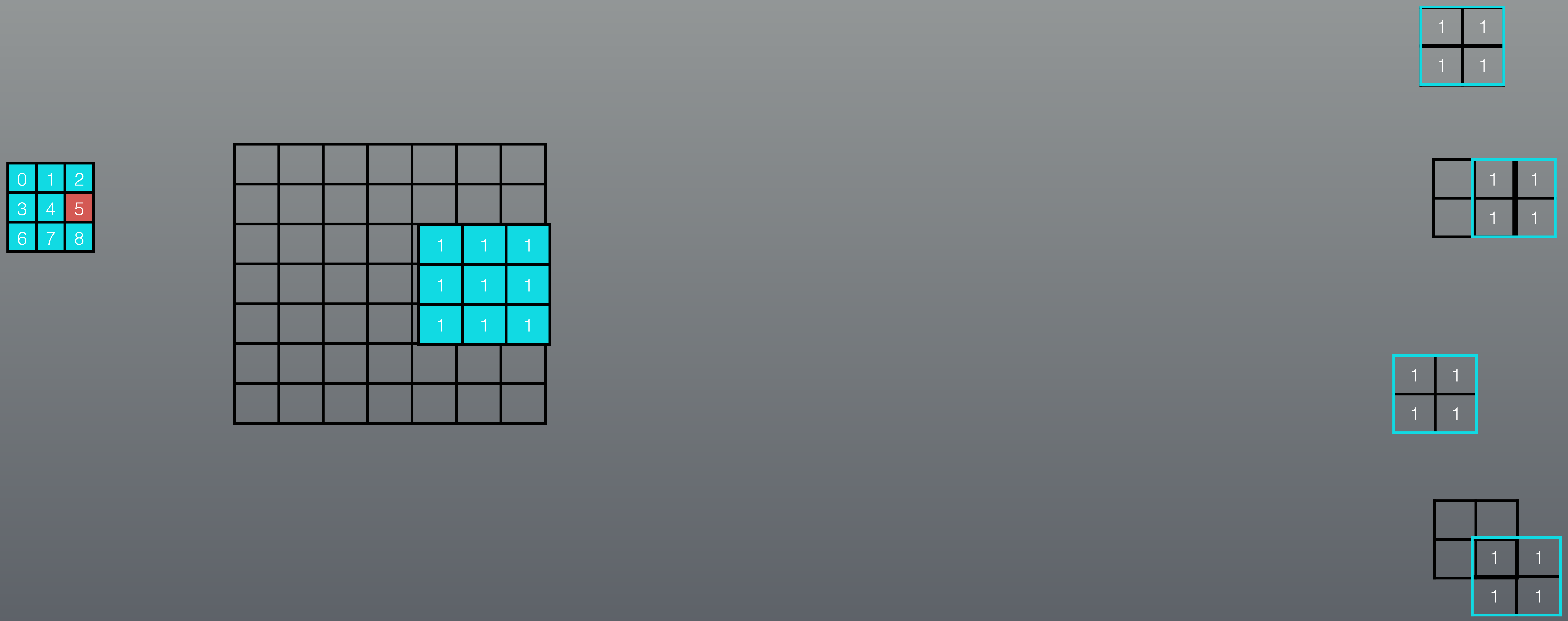
0	1	2
3	4	5
6	7	8

Kernel

1	1	1
1	1	1
1	1	1

$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 6

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 6



CONV2DTranpose (Strides = 2, Kernel_size= 3)

InputID

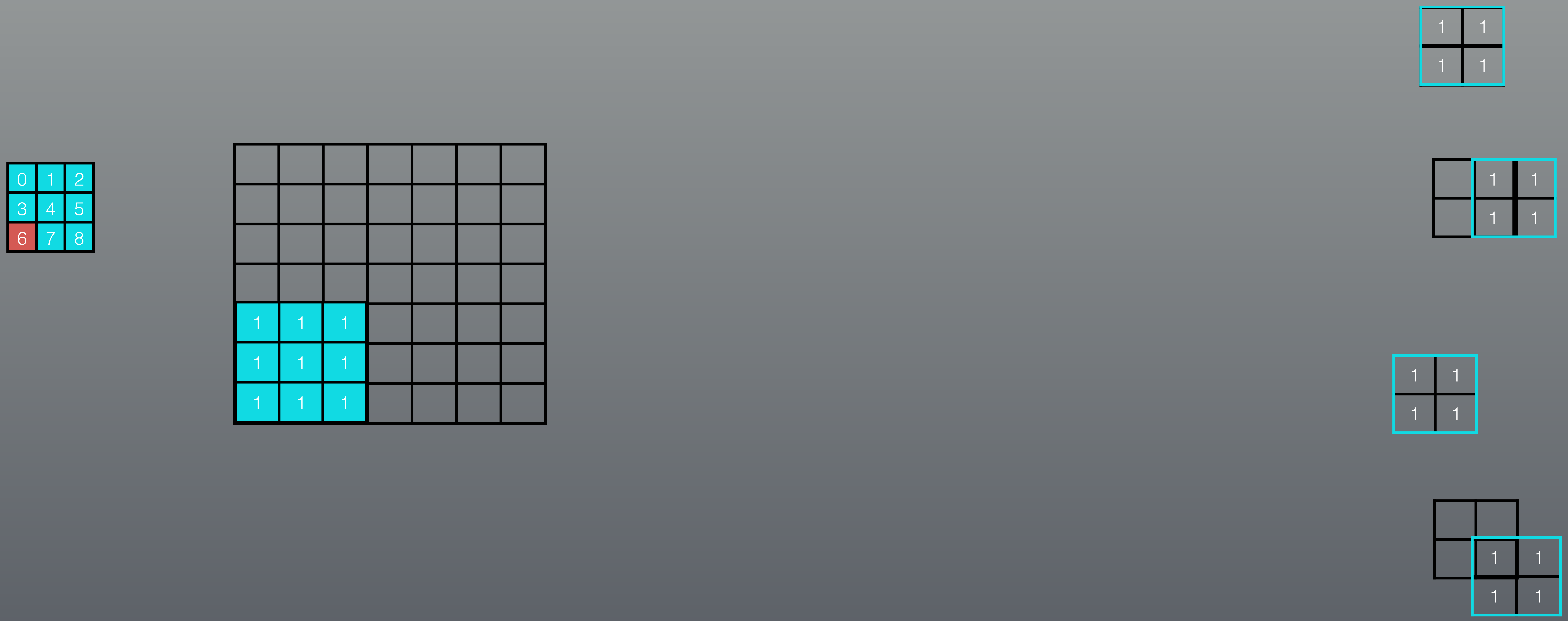
0	1	2
3	4	5
6	7	8

Kernel

1	1	1
1	1	1
1	1	1

$$i \times s_i \text{ to } \overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}} \qquad \text{i\_max\_index = 6}$$

$$j \times s_j \text{ to } j \times s_j + k_{wsize} - 1 \qquad \text{j\_max\_index = 6}$$



CONV2DTranpose (Strides = 2, Kernel_size= 3)

InputID

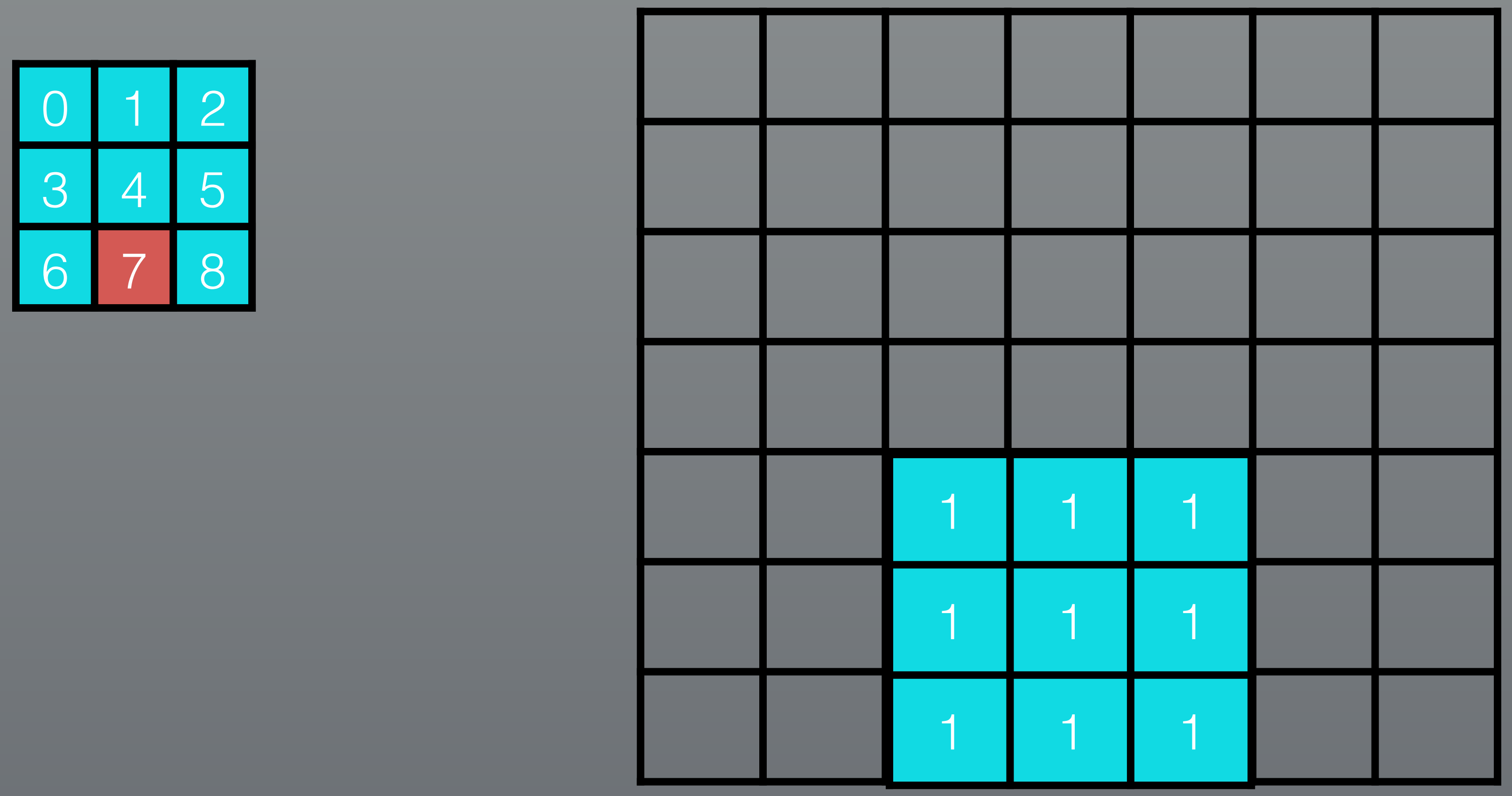
0	1	2
3	4	5
6	7	8

Kernel

1	1	1
1	1	1
1	1	1

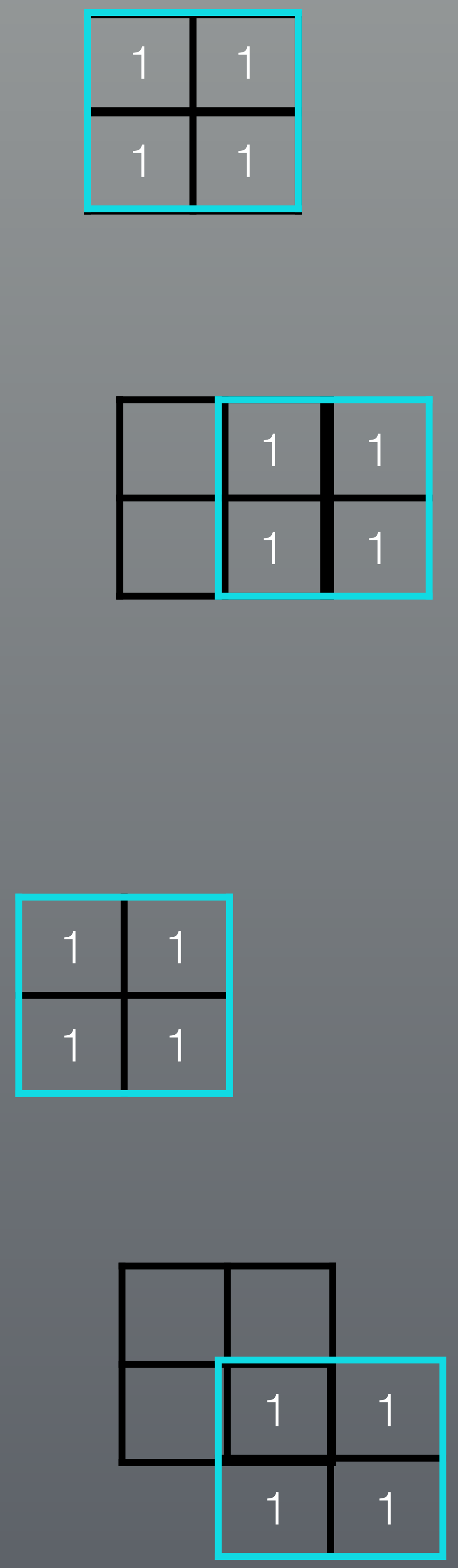
$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 6

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 6



0	1	2
3	4	5
6	7	8

		1	1	1		
		1	1	1		
		1	1	1		



CONV2DTranpose (Strides = 2, Kernel_size= 3)

InputID

0	1	2
3	4	5
6	7	8

Kernel

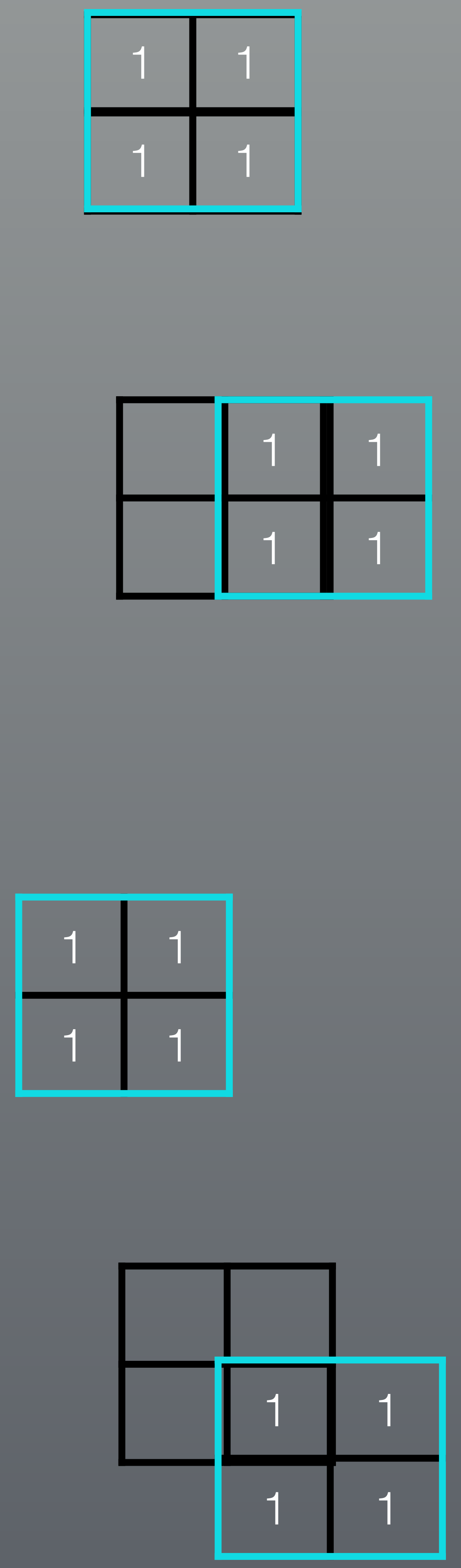
1	1	1
1	1	1
1	1	1

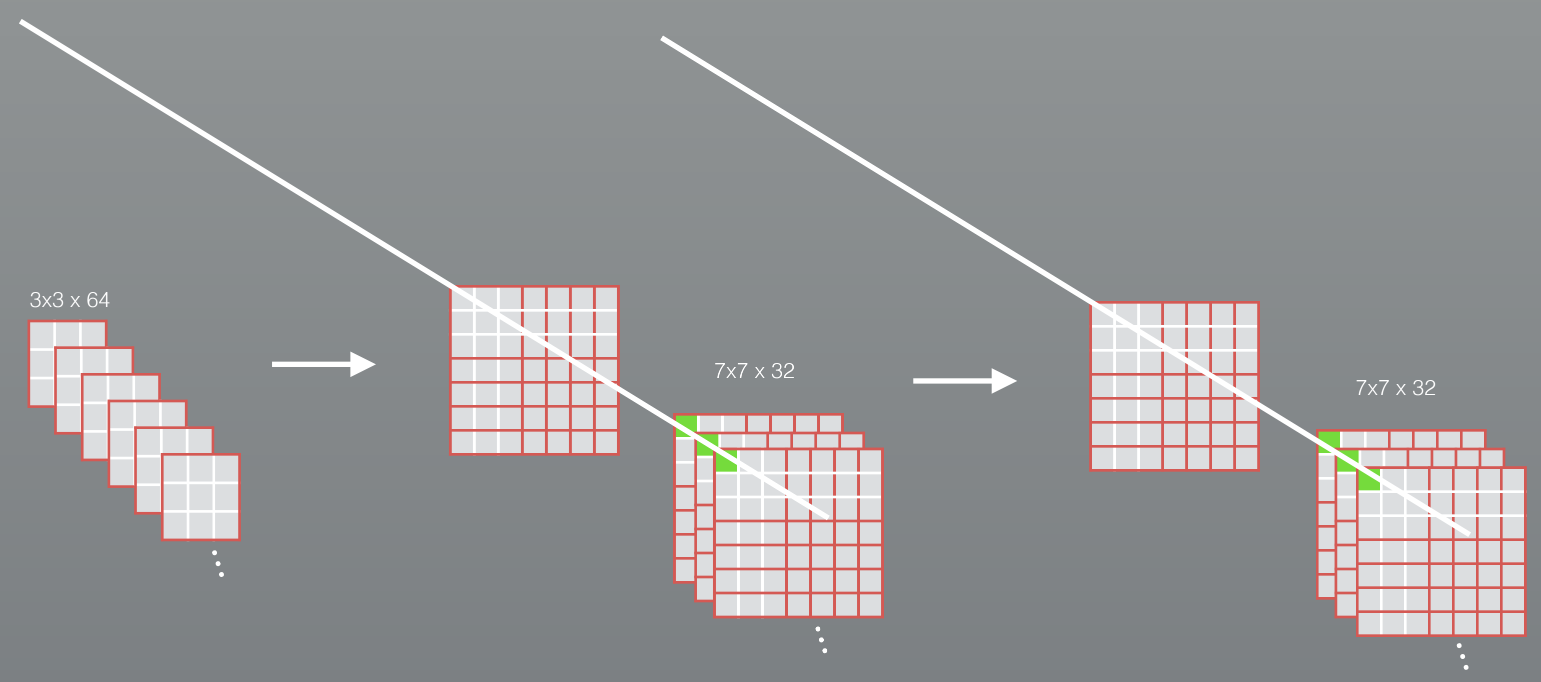
$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 6

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 6

0	1	2
3	4	5
6	7	🟥 8

				1	1	1
				1	1	1
				1	1	1



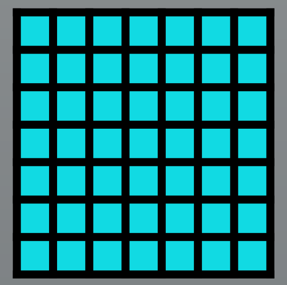

3x3 x 64
7x7 x 32
7x7 x 32

CONV2DTranpose (Strides = 2, Kernel_size= 3, same=padding)

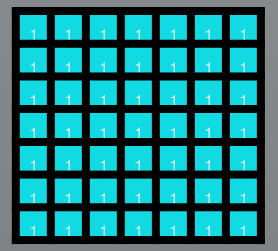
$$i \times s_i \text{ to } \overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}} \qquad \text{i\_max\_index = 13}$$

$$j \times s_j \text{ to } j \times s_j + k_{wsize} - 1 \qquad \text{j\_max\_index = 13}$$

InputID

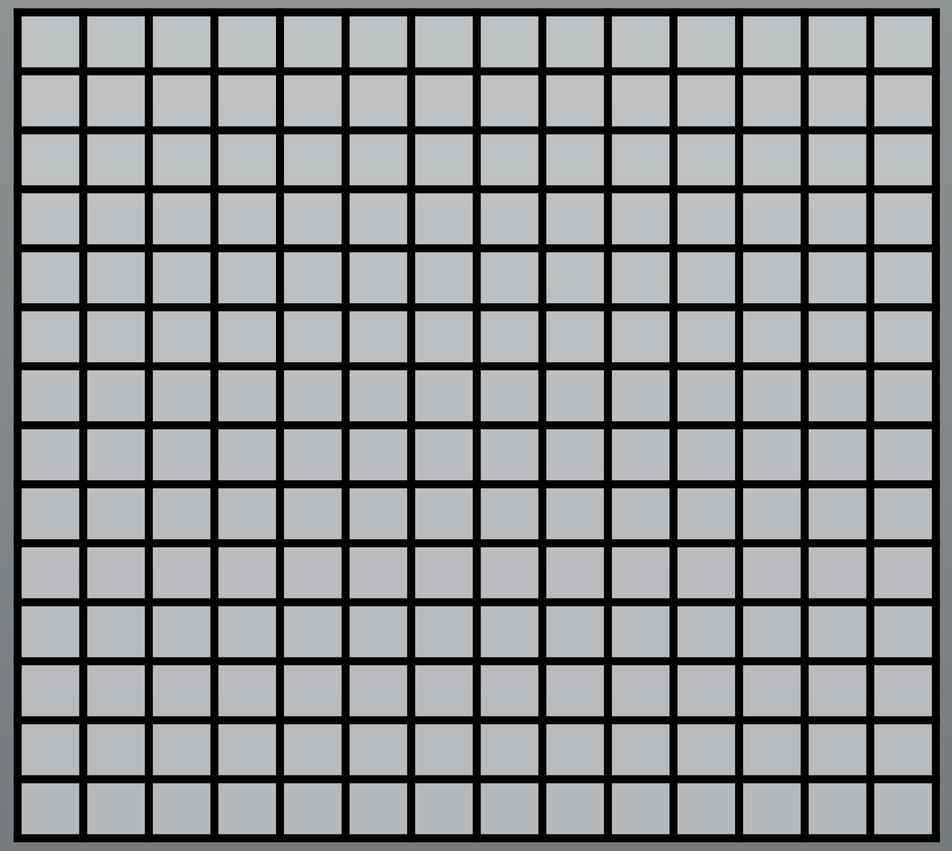


Kernel



1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

Output Dimensions (Upsample)

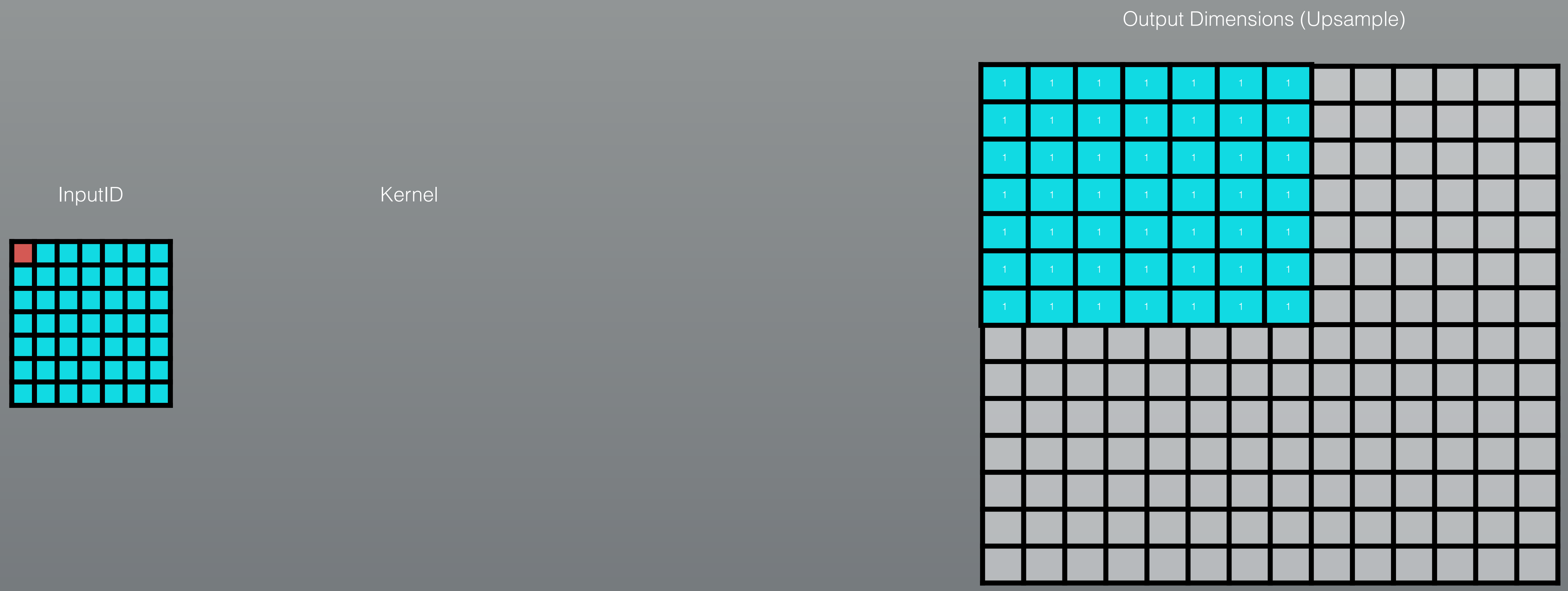


CONV2DTranpose (Strides = 2, Kernel_size= 3)

$$\underbrace{}_{}\overset{\text{Max index}}{\overbrace{i \times s_i \text{ to } i \times s_i + k_{hsize} - 1}}$$

$i \times s_i$ to $i \times s_i + k_{hsize} - 1$ i_max_index = 13

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 13



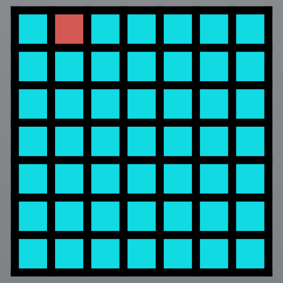
1	1	1	1	1	1	1							
1	1	1	1	1	1	1							
1	1	1	1	1	1	1							
1	1	1	1	1	1	1							
1	1	1	1	1	1	1							
1	1	1	1	1	1	1							
1	1	1	1	1	1	1							

CONV2DTranpose (Strides = 2, Kernel_size= 3)

$$i \times s_i \text{ to } \overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}} \qquad \text{i\_max\_index = 13}$$

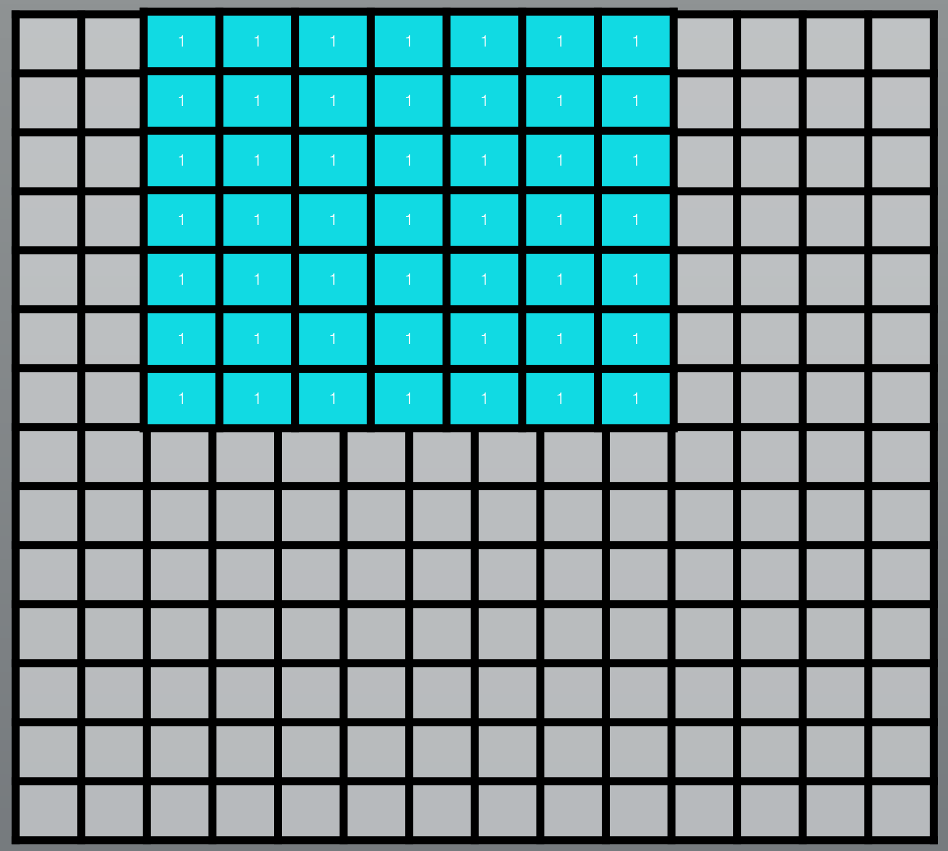
$$j \times s_j \text{ to } j \times s_j + k_{wsize} - 1 \qquad \text{j\_max\_index = 13}$$

InputID



Kernel

Output Dimensions (Upsample)



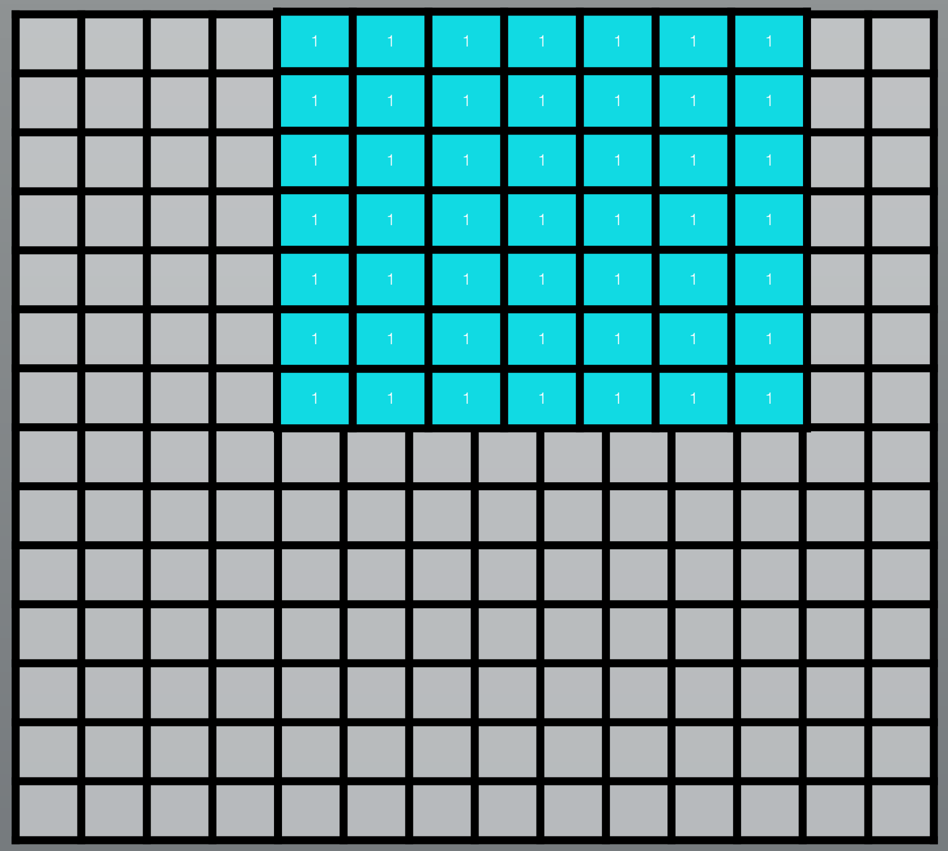
CONV2DTranpose (Strides = 2, Kernel_size= 3)

Max index

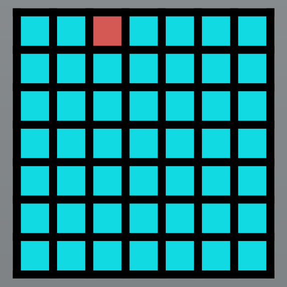
$$\underbrace{i \times s_i \text{ to } i \times s_i + k_{hsize} - 1}$$ i_max_index = 13

$$j \times s_j \text{ to } j \times s_j + k_{wsize} - 1$$ j_max_index = 13

Output Dimensions (Upsample)



InputID



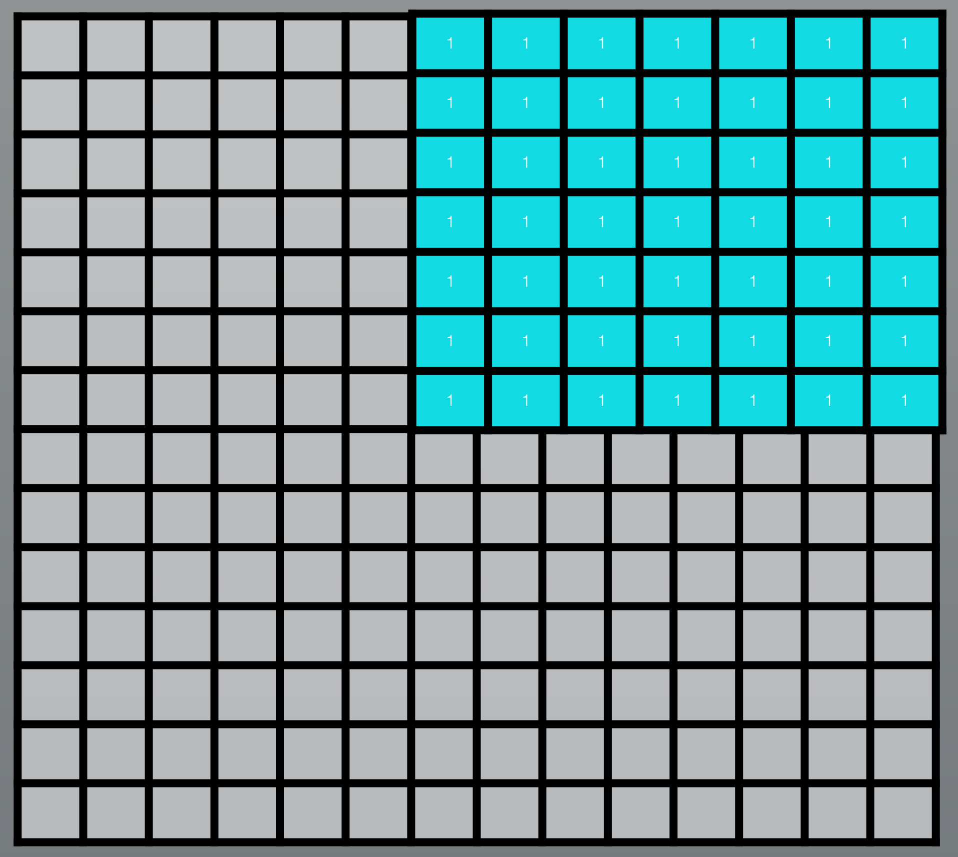
Kernel

CONV2DTranpose (Strides = 2, Kernel_size= 3)

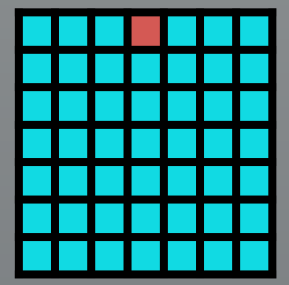
$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 13

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 13

Output Dimensions (Upsample)



InputID



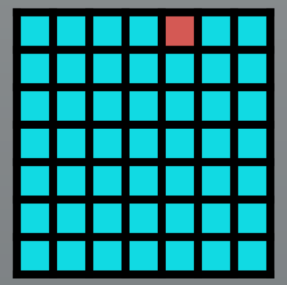
Kernel

CONV2DTranpose (Strides = 2, Kernel_size= 3)

$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 13

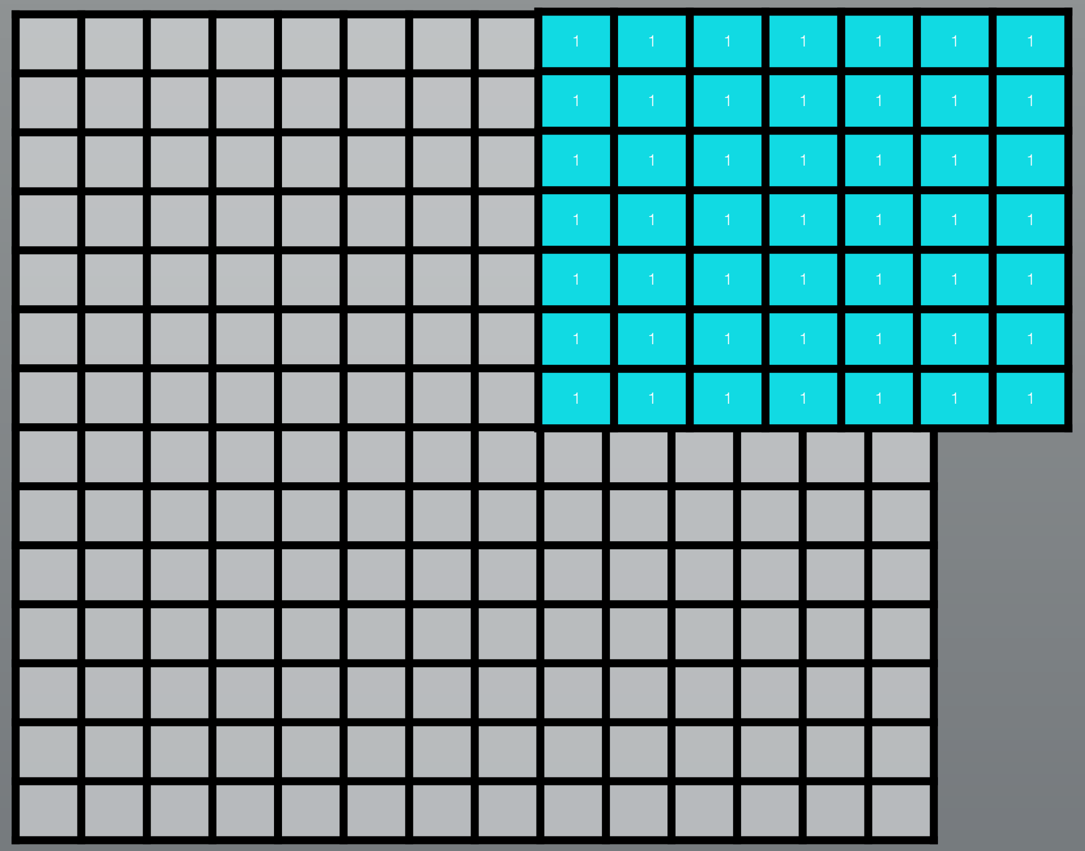
$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 13

InputID



Kernel

Output Dimensions (Upsample)



CONV2DTranpose (Strides = 2, Kernel_size= 3)

$$i \times s_i \text{ to } \overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$$

$$j \times s_j \text{ to } j \times s_j + k_{wsize} - 1$$

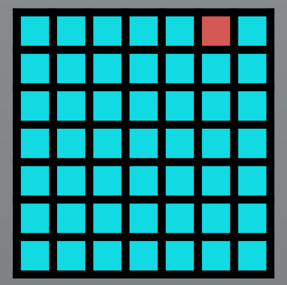
i_max_index = 13

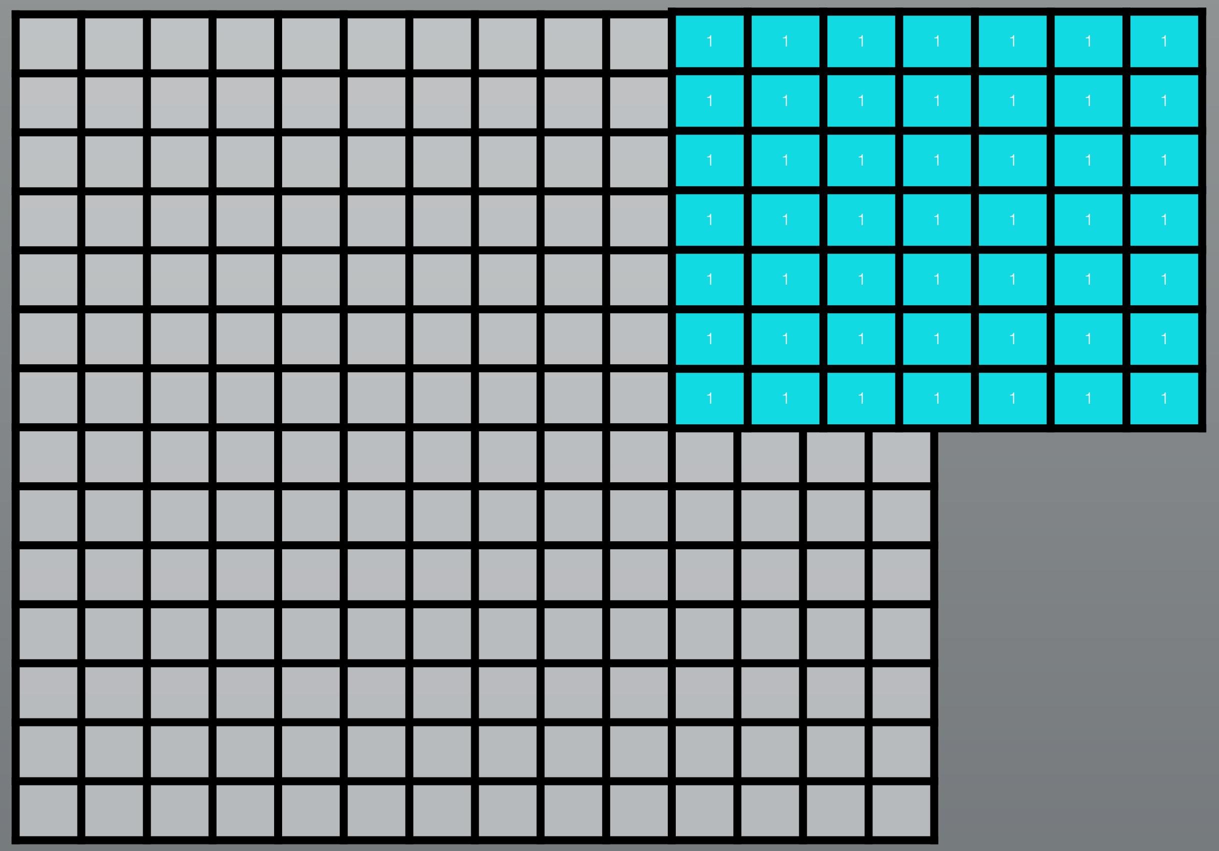
j_max_index = 13

Output Dimensions (Upsample)

InputID

Kernel



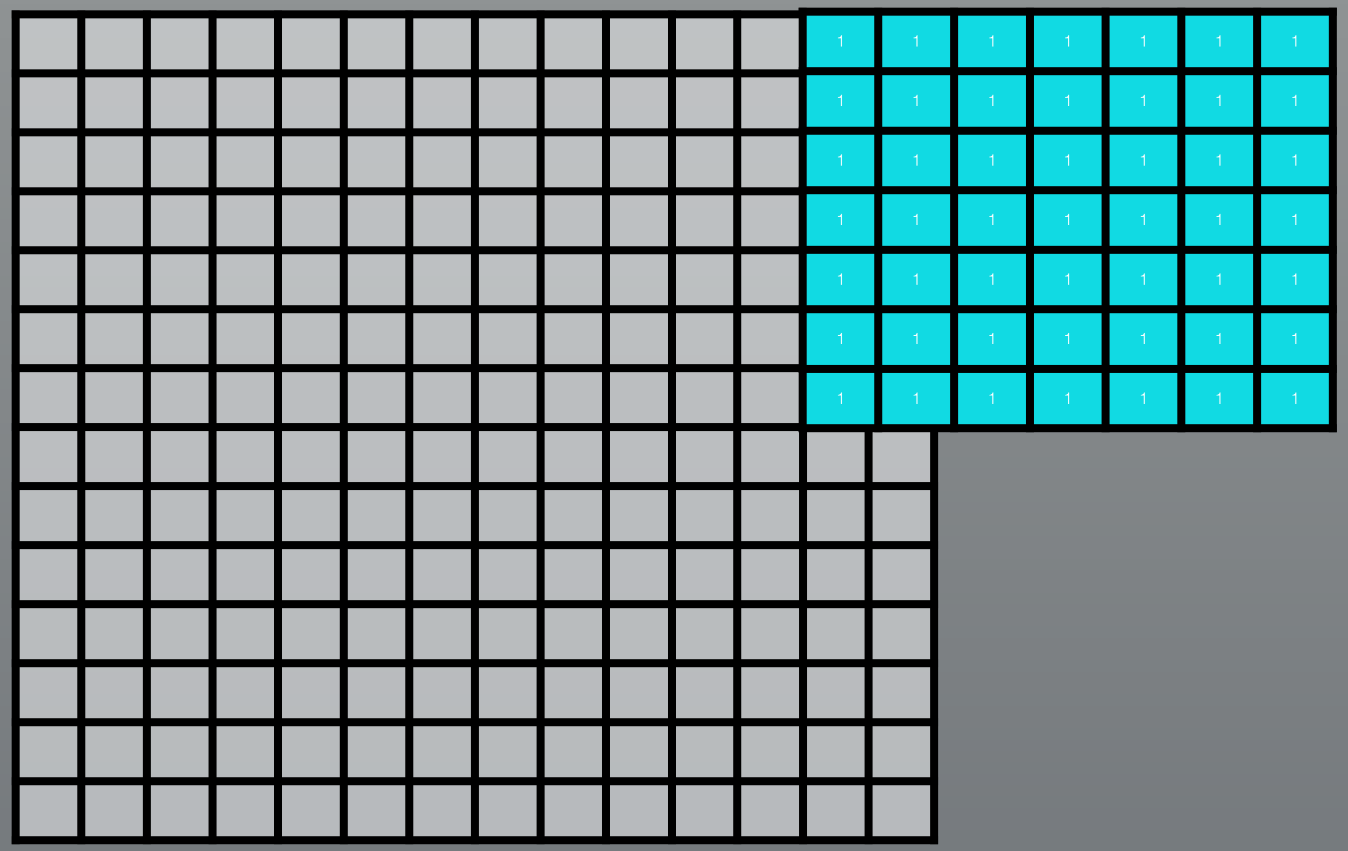


CONV2DTranpose (Strides = 2, Kernel_size= 3,)

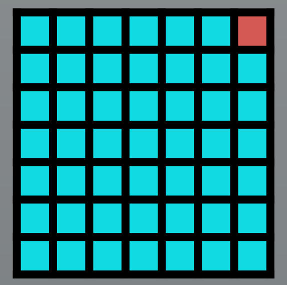
$i \times s_i$ to $\overbrace{i \times s_i + k_{hsize} - 1}^{\text{Max index}}$ i_max_index = 13

$j \times s_j$ to $j \times s_j + k_{wsize} - 1$ j_max_index = 13

Output Dimensions (Upsample)



InputID

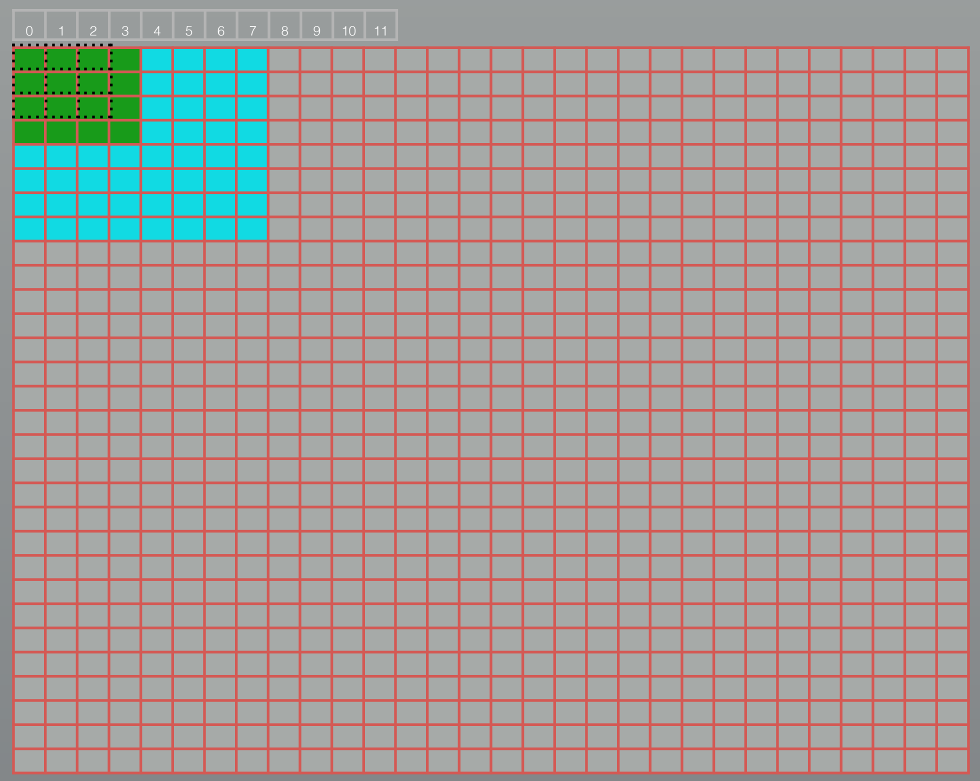


Kernel

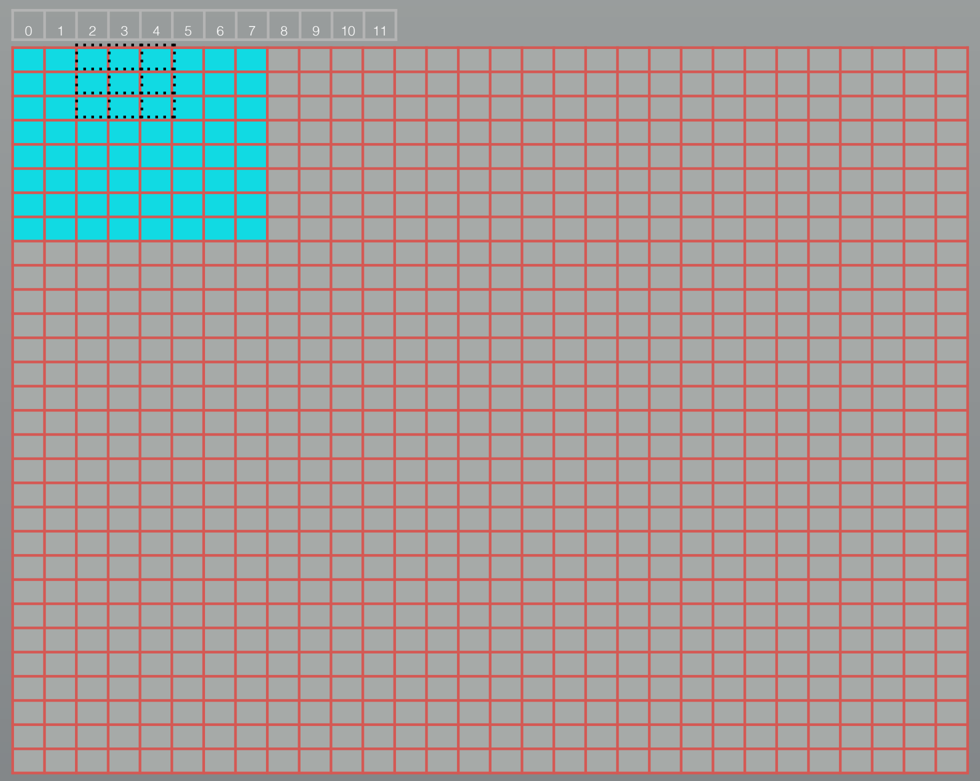
Input = 4x4 Strides = 2 Kernel = 3 SAME (zero pads)

Output = 8 x 8

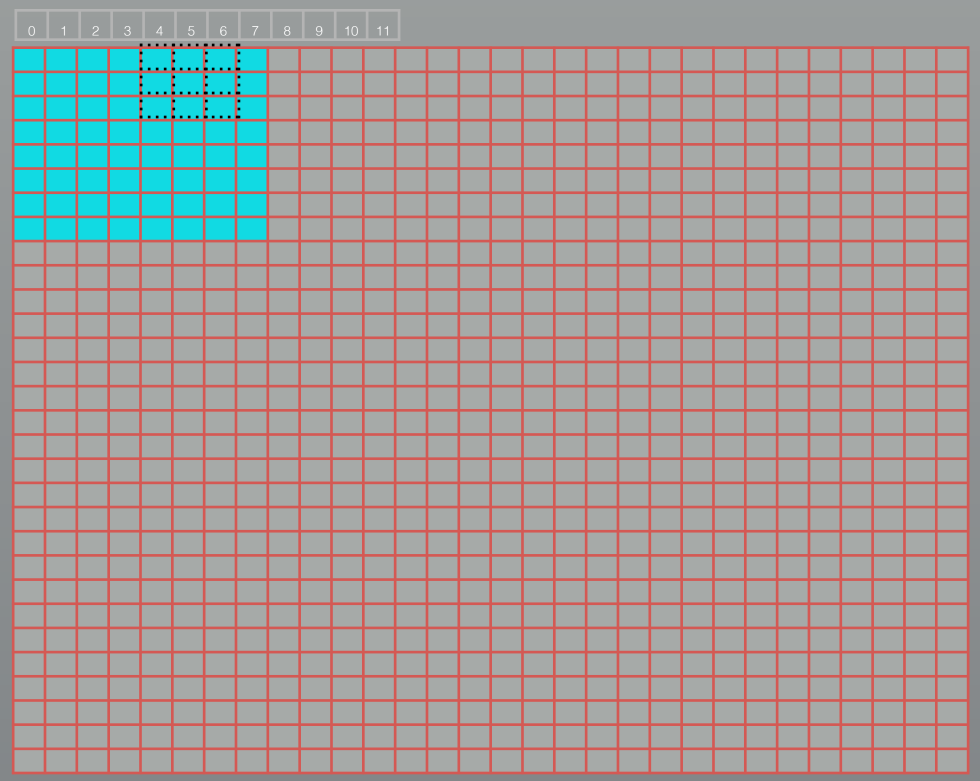
(0,0)



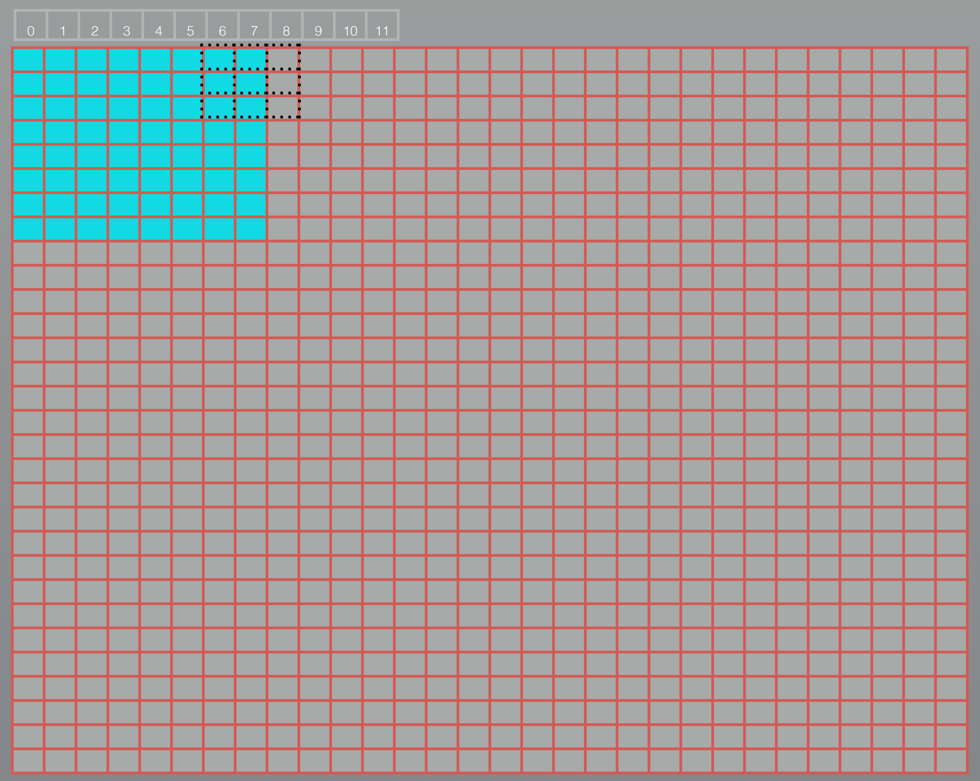
(0,1)

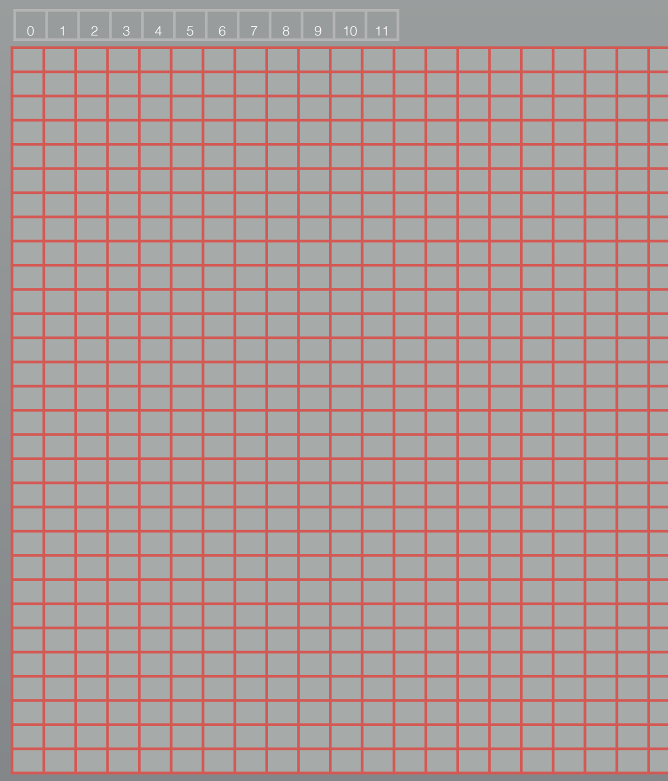


(0,2)



(0,3)



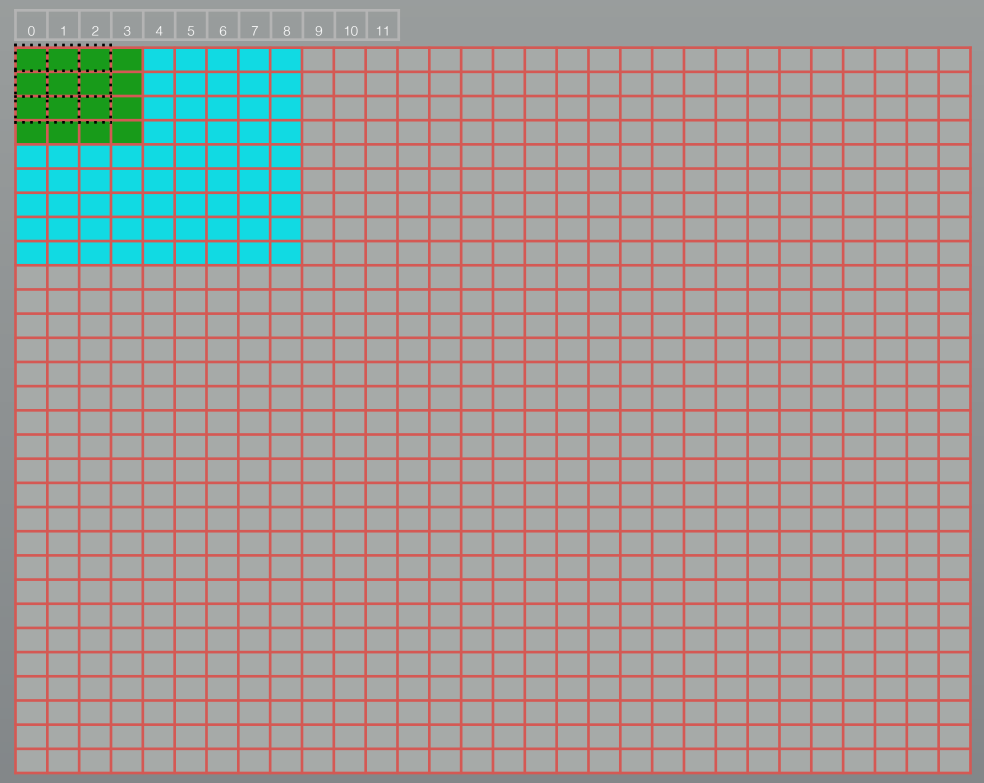


(0,2)

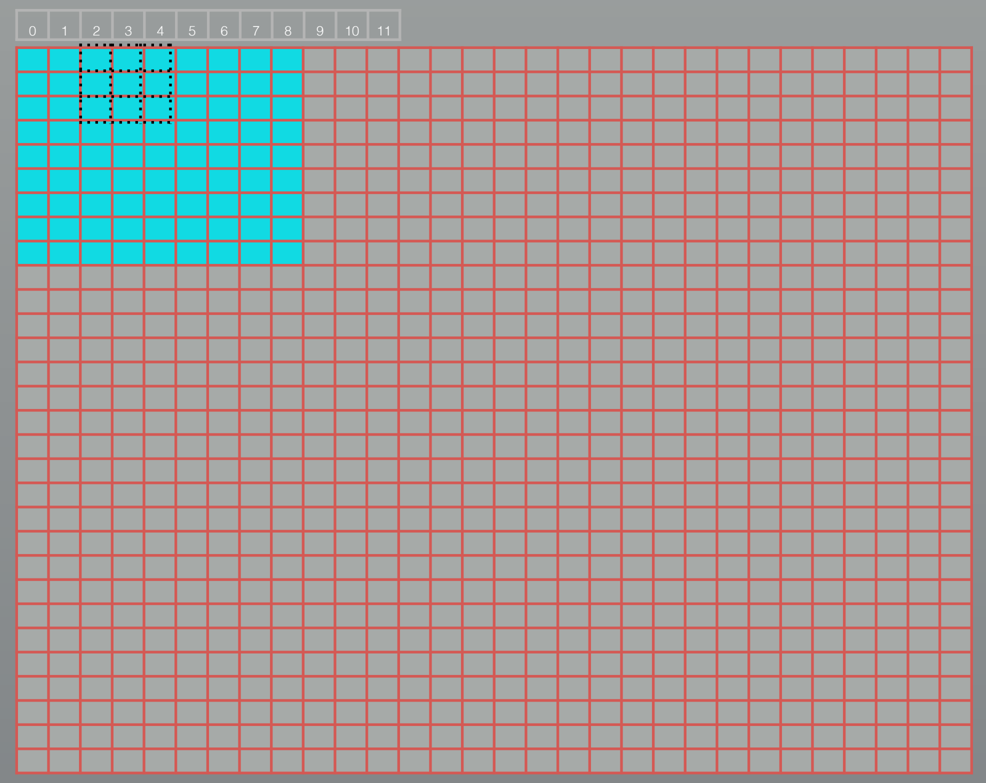
Input = 4x4   Strides = 2   Kernel = 3   VALUE (Truncate)

Output = 9 x 9

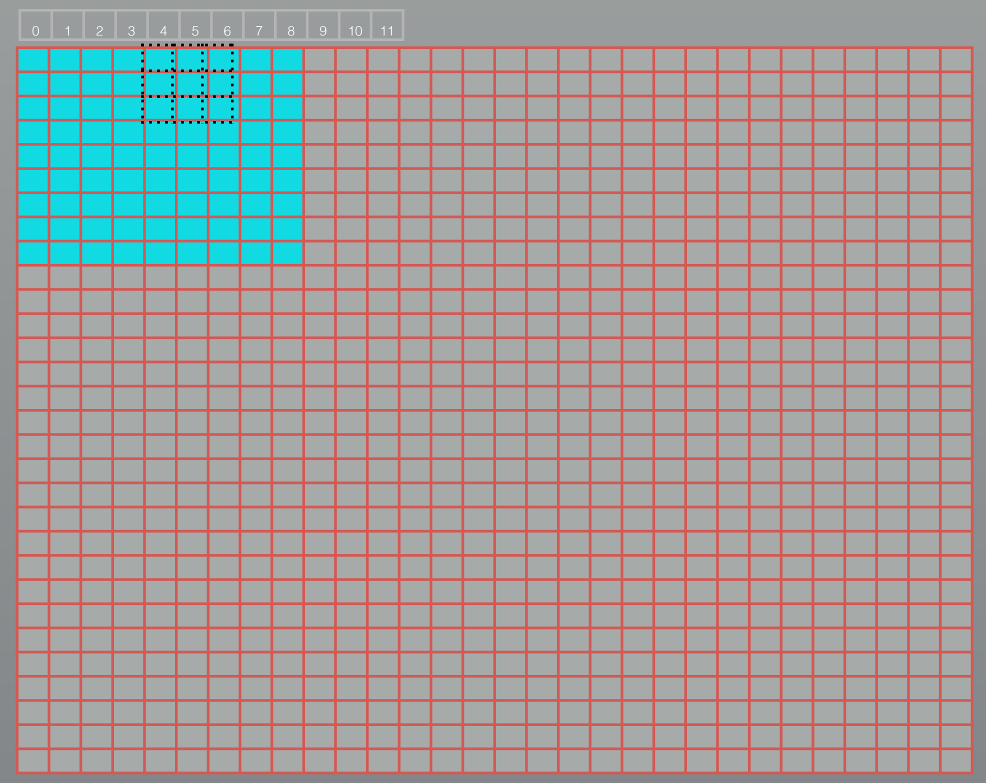
(0,0)



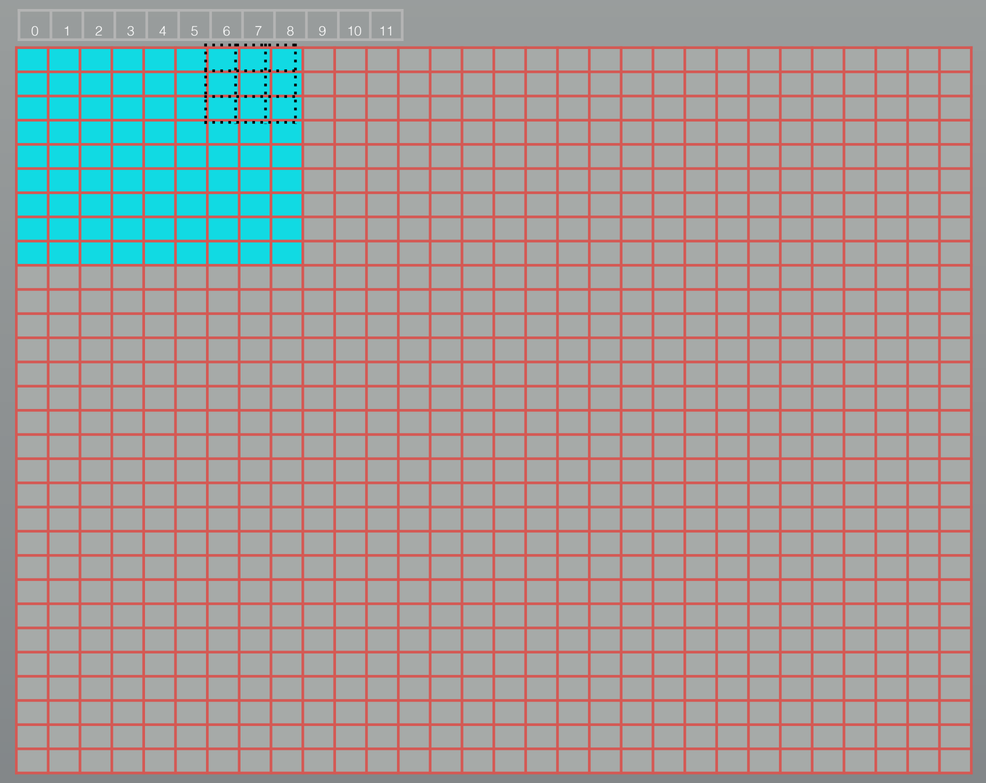
(0,1)



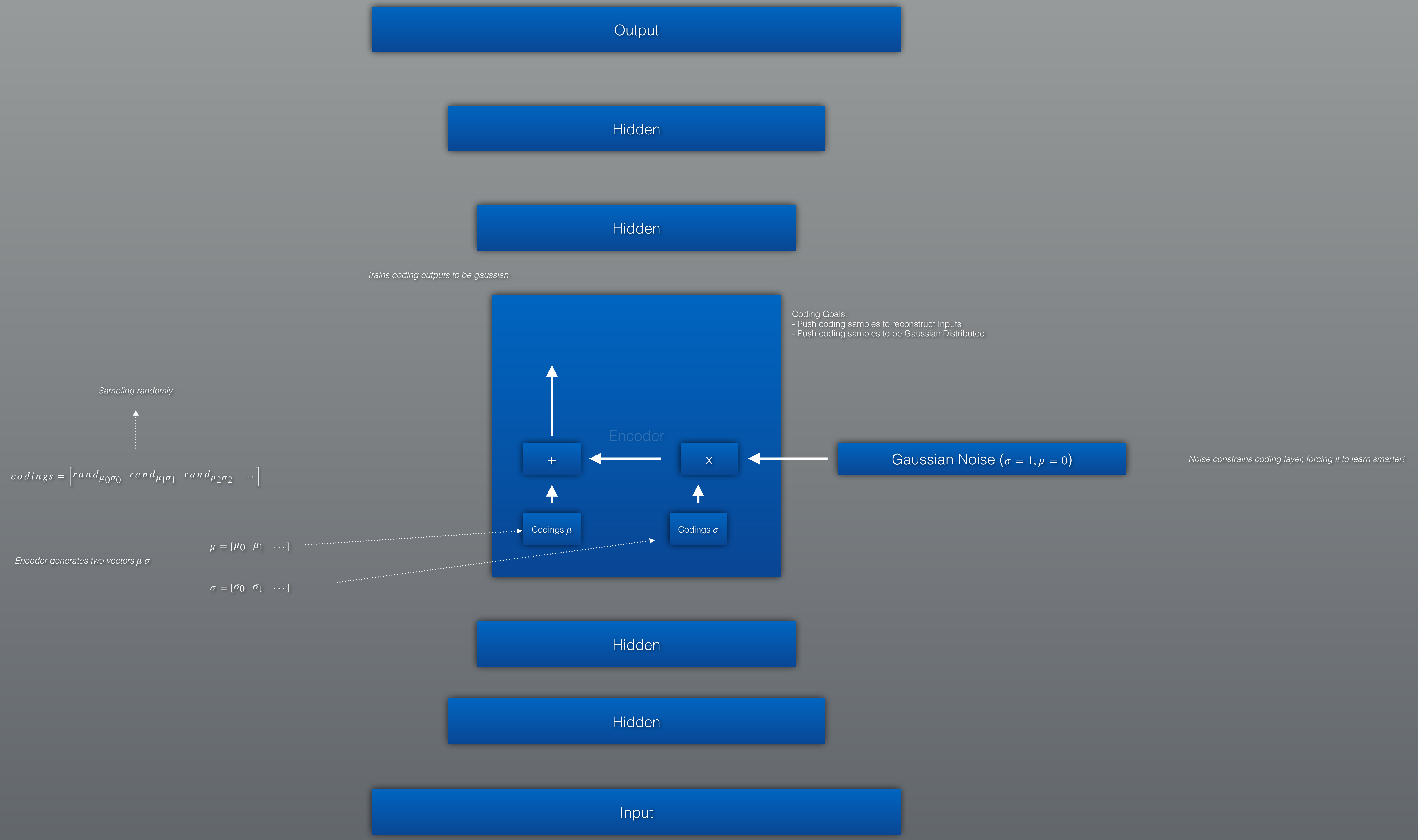
(0,2)



(0,3)



Variational Autoencoder



Trains coding outputs to be gaussian

Coding Goals:
- Push coding samples to reconstruct Inputs
- Push coding samples to be Gaussian Distributed

Sampling randomly

$$codings = \begin{bmatrix} rand_{\mu_0\sigma_0} & rand_{\mu_1\sigma_1} & rand_{\mu_2\sigma_2} & \cdots \end{bmatrix}$$

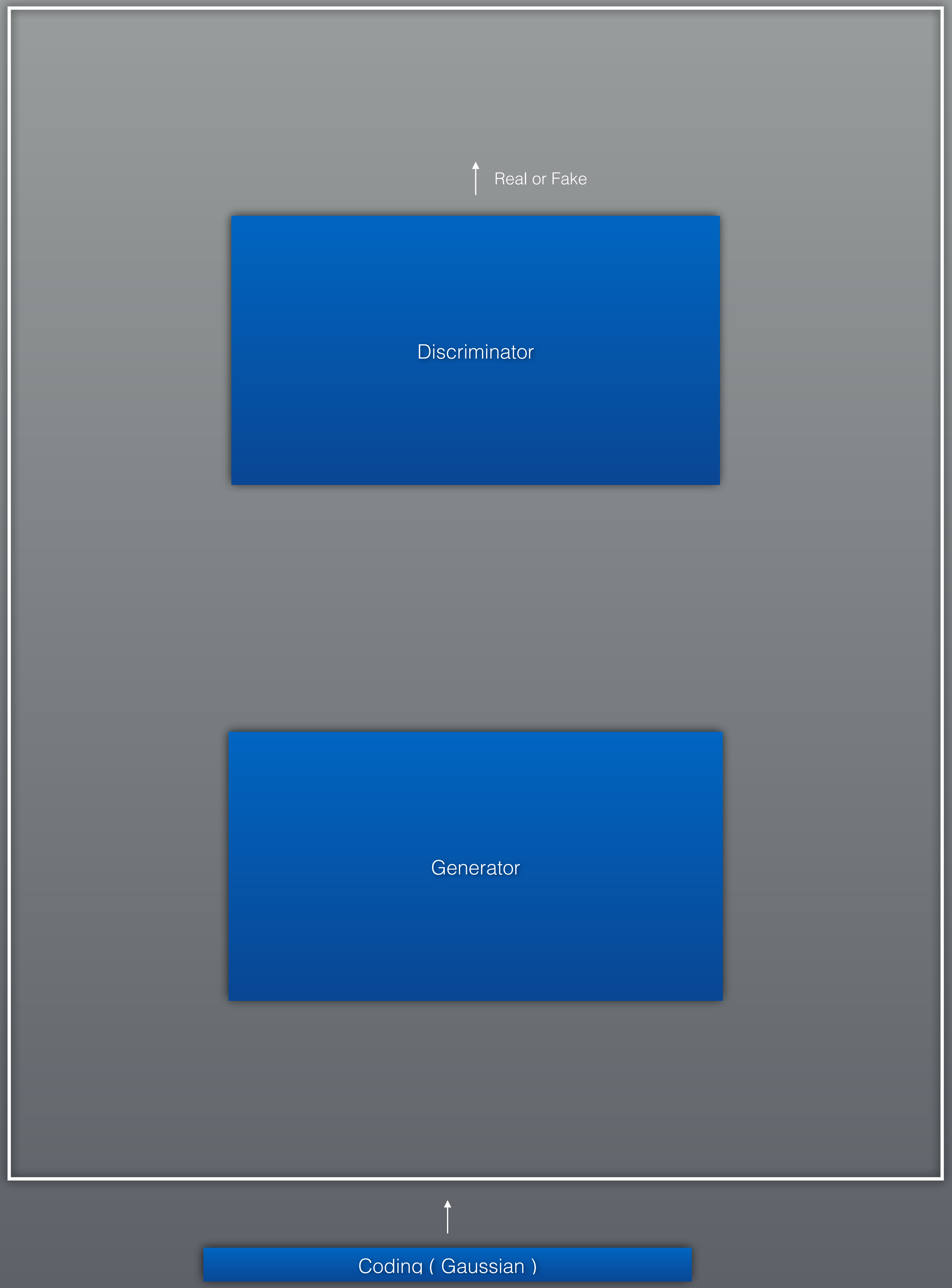
Noise constrains coding layer, forcing it to learn smarter!

$$\mu = [\mu_0 \quad \mu_1 \quad \cdots]$$

Encoder generates two vectors μ σ

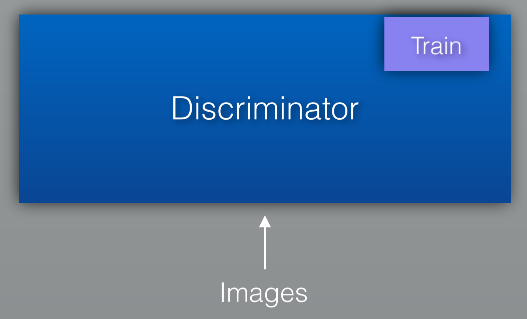
$$\sigma = [\sigma_0 \quad \sigma_1 \quad \cdots]$$

Variational Autoencoder



Phase 1

Discriminator



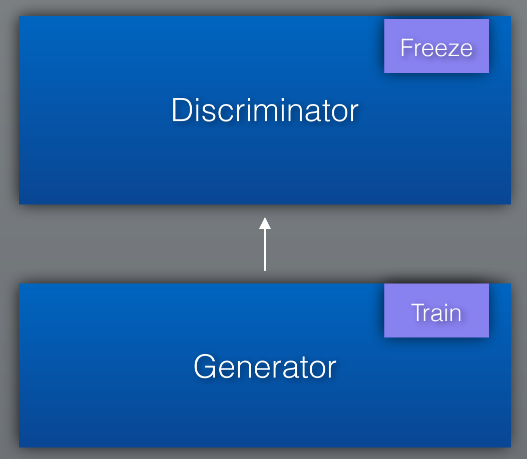
Images

Generator Image	0
⋮	⋮
Real Image	1
⋮	⋮

Train Discriminator to ‘understand’ generated images vs real images

Phase 2

Generator



Generator Image	1
⋮	⋮

Train Generator to create realistic images by learning from the discriminators reaction.

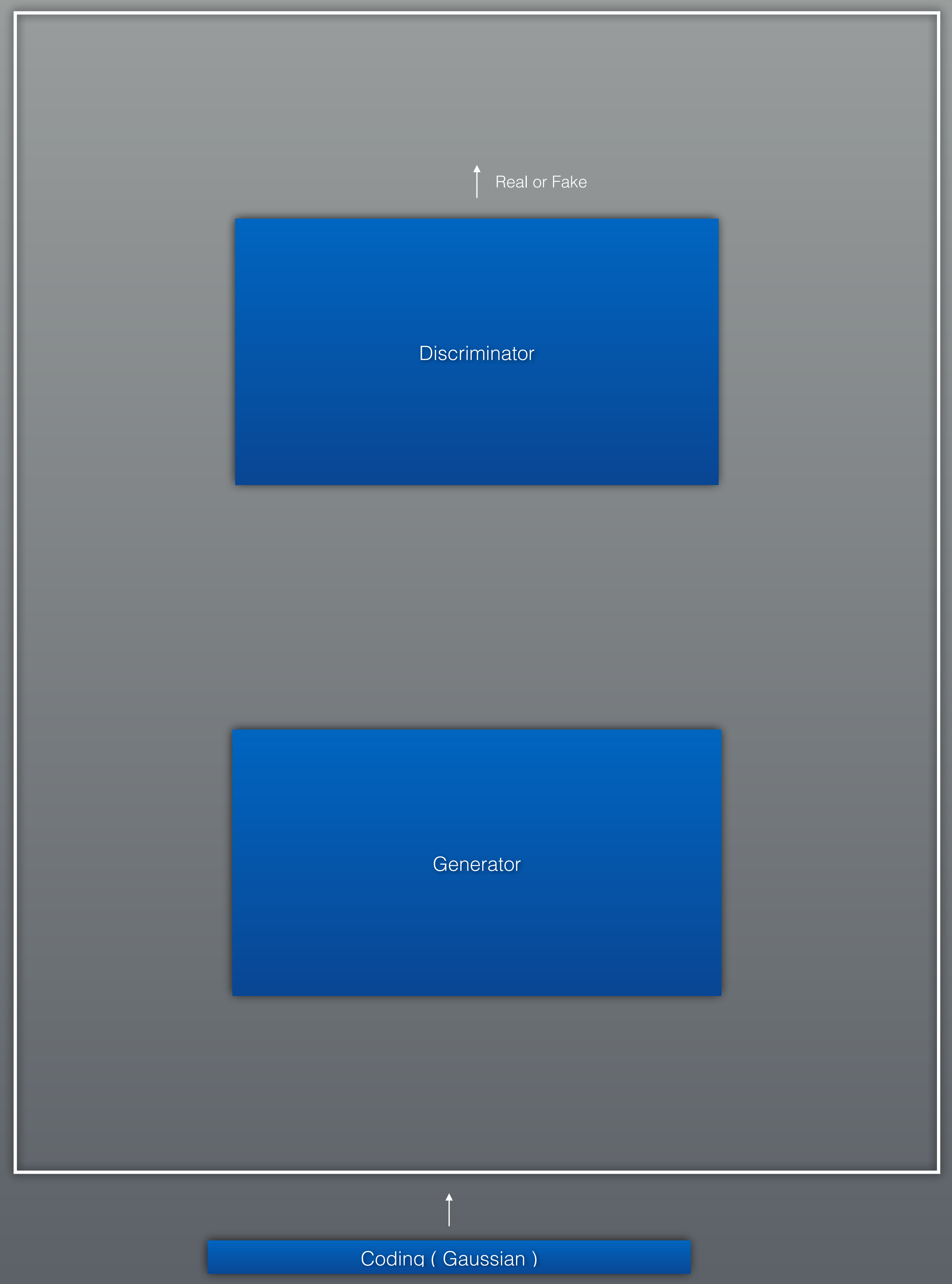
Discriminator (Fail):
“You think this is an image, this is not an image, this region needs to be re-done. Please fix”

Generator:
I have acquired new skills.

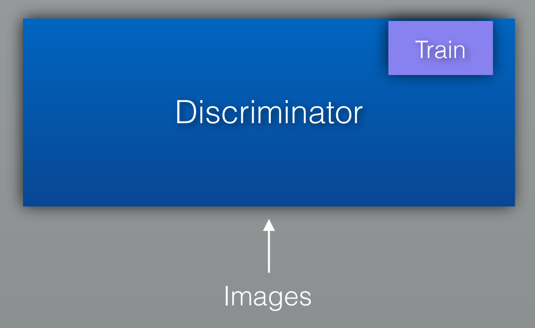
Discriminator (Pass):
“This image is an image great work”

Generator:
Nothing to really learn, but thanks for confirming

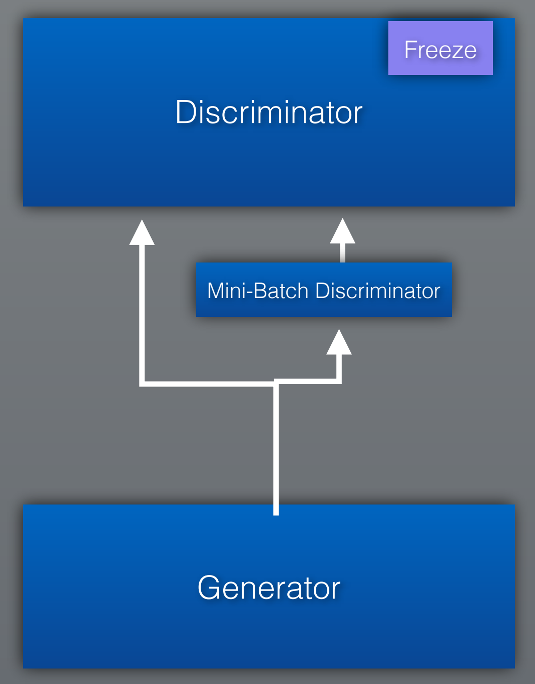
Variational Autoencoder



Phase 1



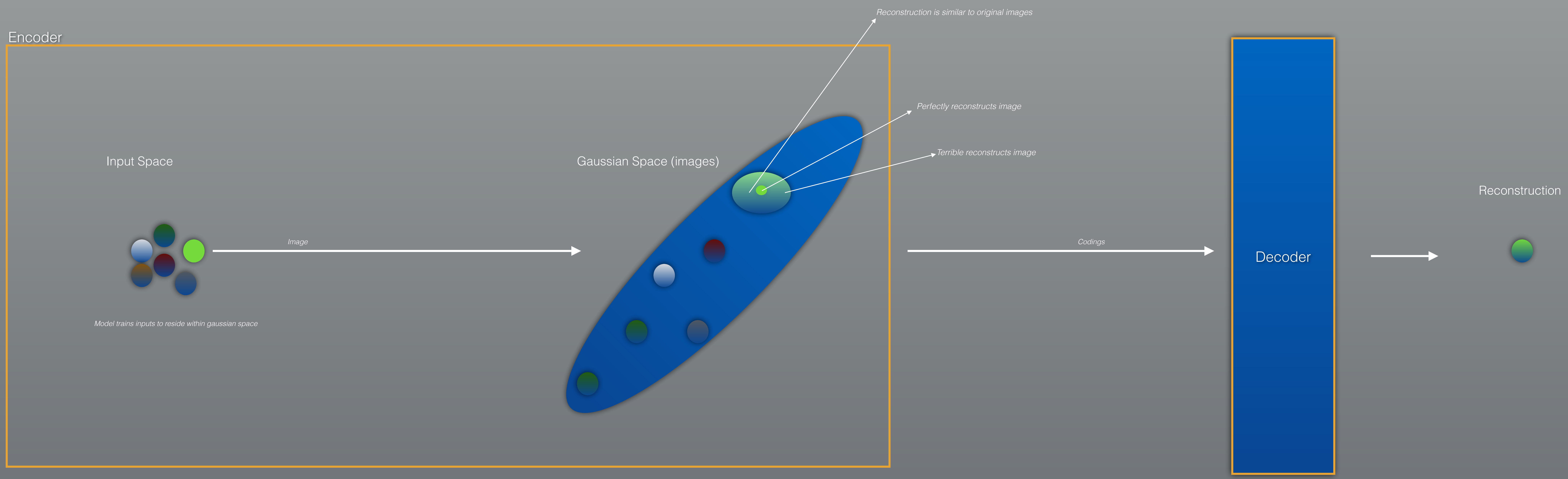
Phase 2



Mini-Batch Discriminator:
Measure similarity between images in batch. For instance a score between 0 and 1 per instance in batch

Notes

Variational Autoencoder



Encoder

Input Space

Model trains inputs to reside within gaussian space

Image

Gaussian Space (images)

Reconstruction is similar to original images

Perfectly reconstructs image

Terrible reconstructs image

Codings

Decoder

Reconstruction