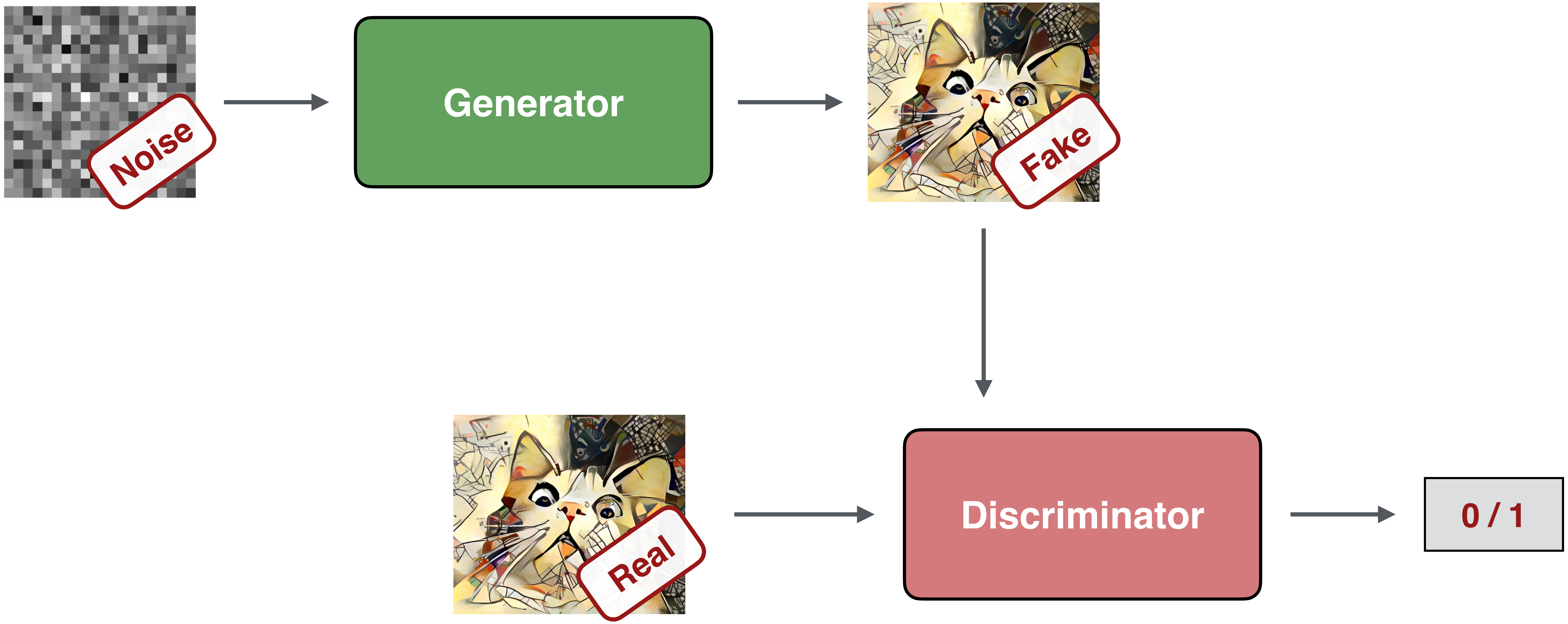
Tutorial 7: DCGAN

Practical Deep Learning for Science
23 May, 2024

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

- Nilotpal

GAN



Noise

Generator

Fake

Real

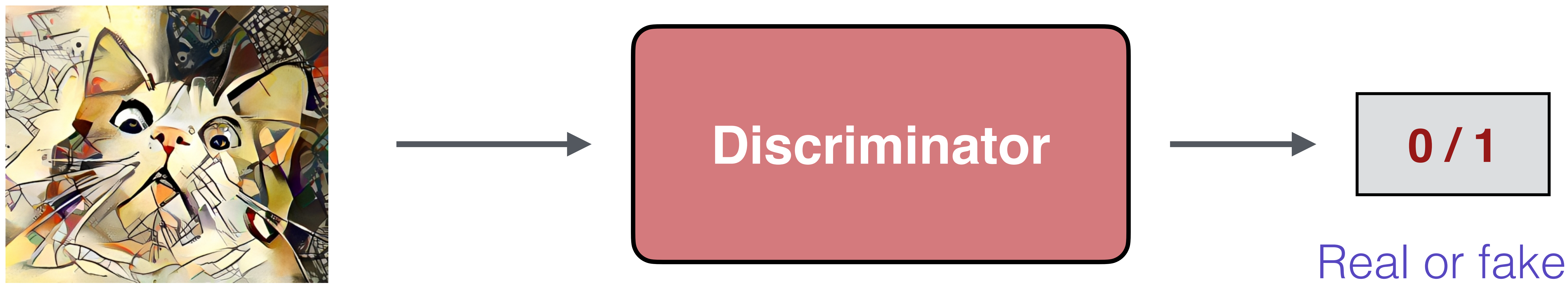
Discriminator

0 / 1

Regular classifier

Loss? It’s a bit tricky

Discriminator is a regular binary classifier



Discriminator

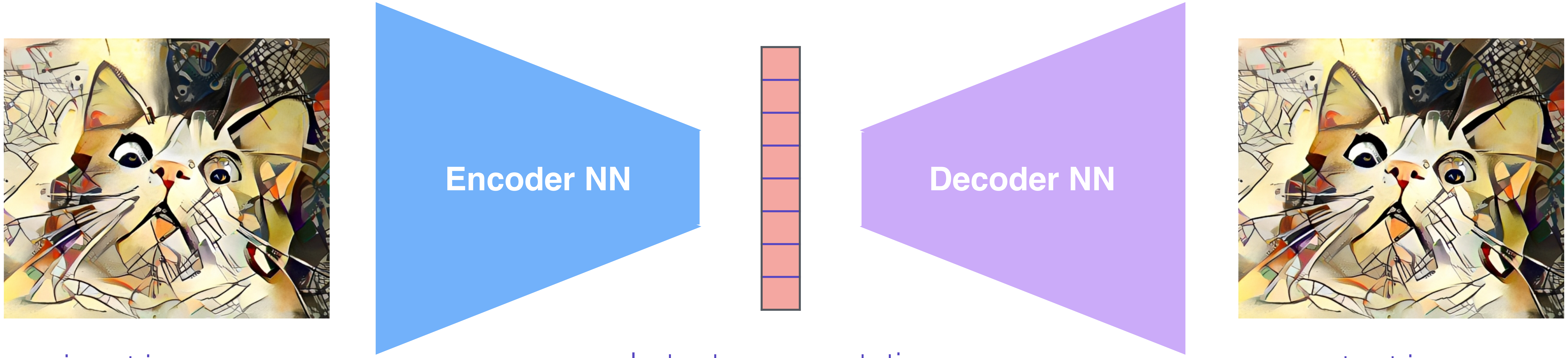
0 / 1

Real or fake?

Regular classifier

Binary Cross entropy loss

Generator loss?



input image
(x)

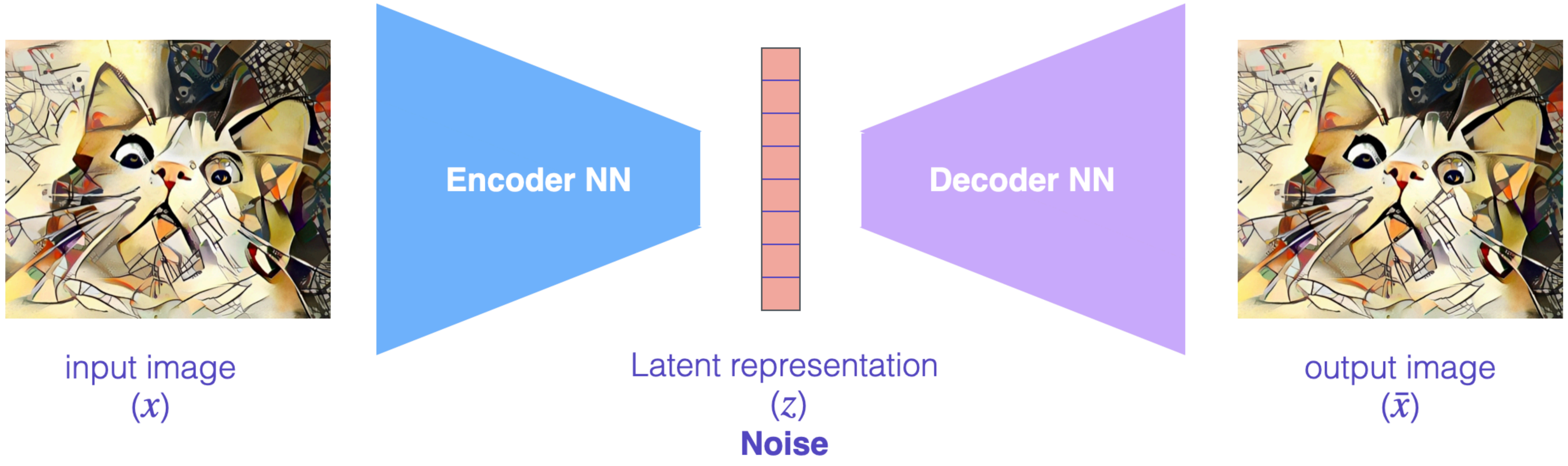
Latent representation
(z)
Noise

output image
$(\bar{x})$

AutoEncoder

Loss (input image, output image

Generator loss?



Encoder NN

Decoder NN

input image
(x)

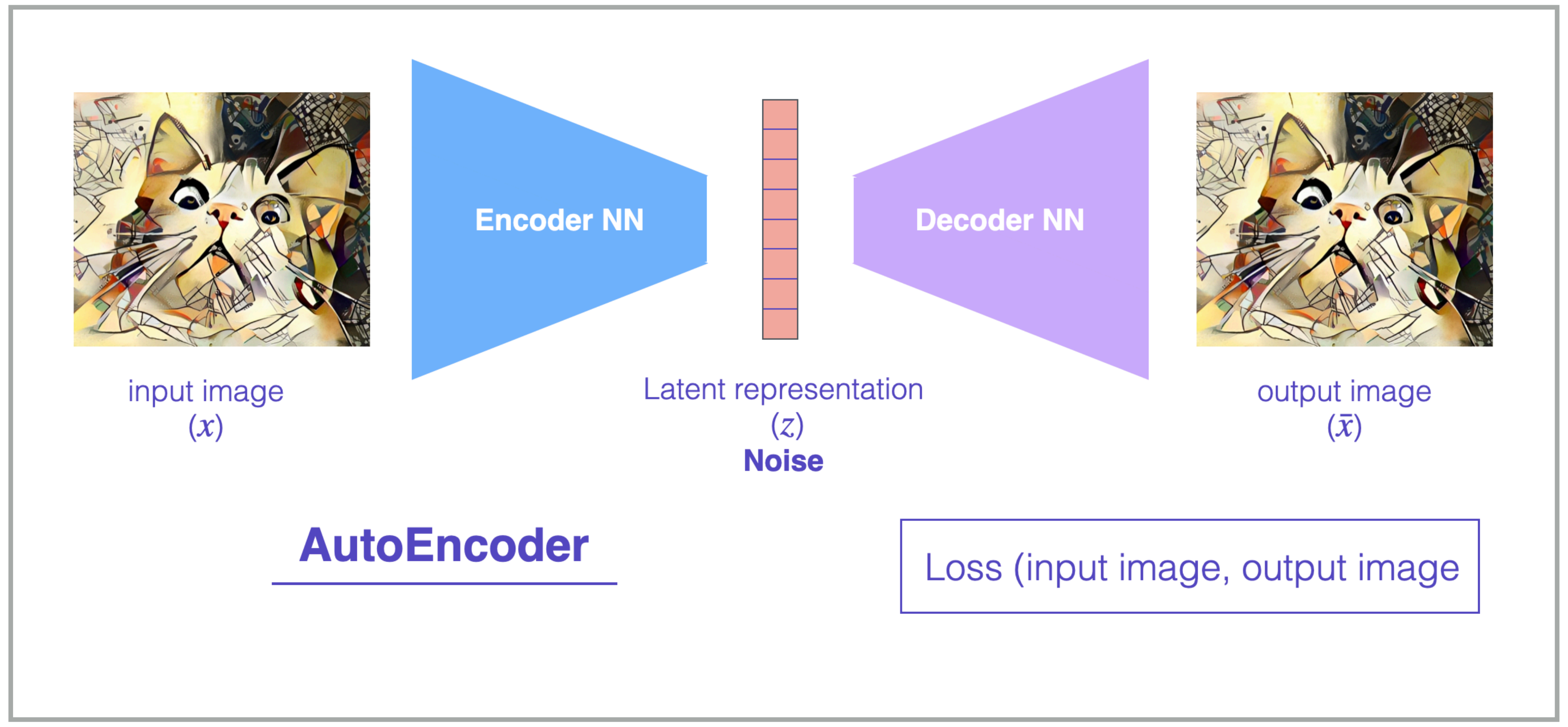
Latent representation
(z)
Noise

output image
$(\bar{x})$

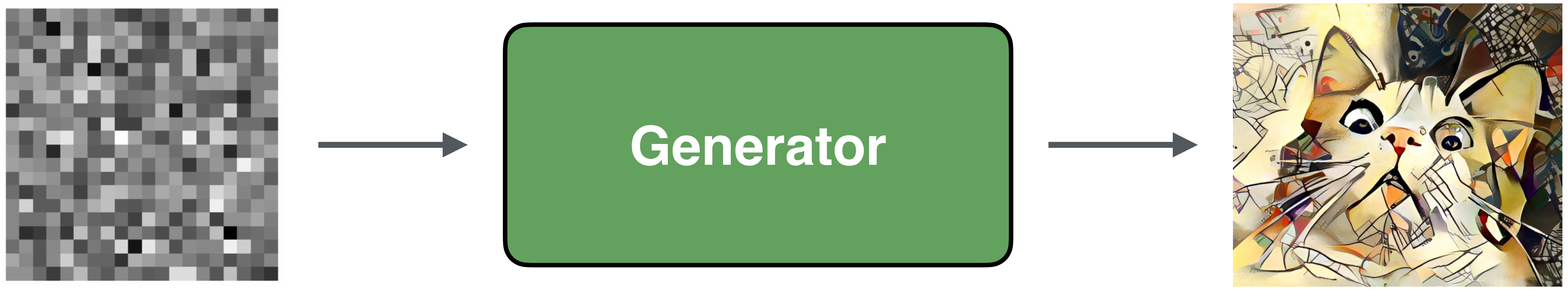
AutoEncoder

Loss (input image, output image

Generator loss?



No input image
Can't compute loss like in AE


Generator

Noise

Output image

Adversary

- Generator and Discriminator have exact opposite goal
- Will use the discriminator loss again
  - But we will flip the label. *Label the generated images as 1 (real)*
  - Want to increase the likelihood of the fake image being classified as real

- From the discriminator perspective
  - Classify real images as real, fake images as fake
- From Generator's perspective
  - *Generate fake images that gets classified as real*

GAN recipe

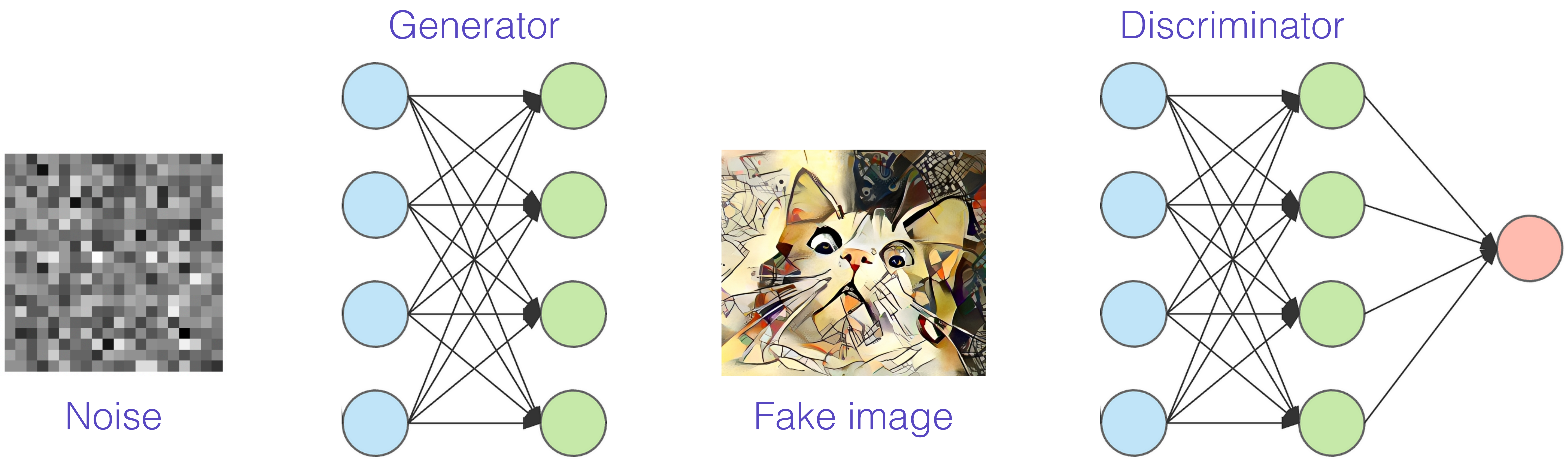
- Generate fake images
- Take real images
- Compute discriminator loss; update discriminator

- Label the fake images as real
- Compute discriminator loss; update generator

.detach()

During **discriminator training,** Torch will see **(Generator + Discriminator)** as **one big network**

Losses will get back propagated to generator as well

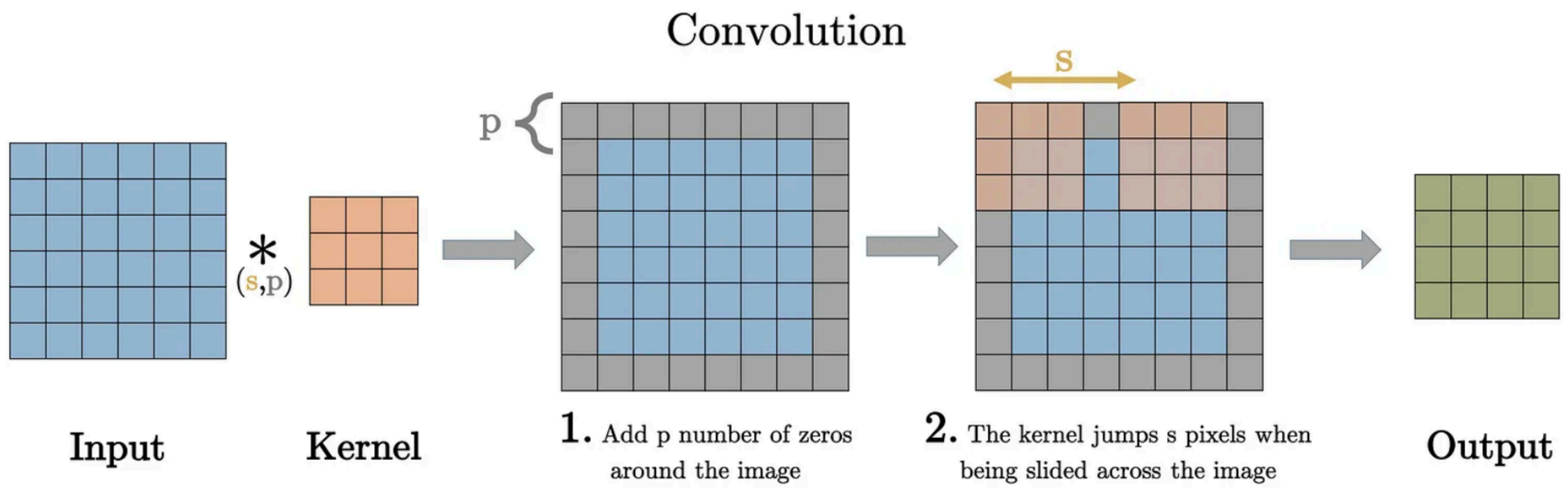


We need to detach the image from the generator with `fake_img.detach()`

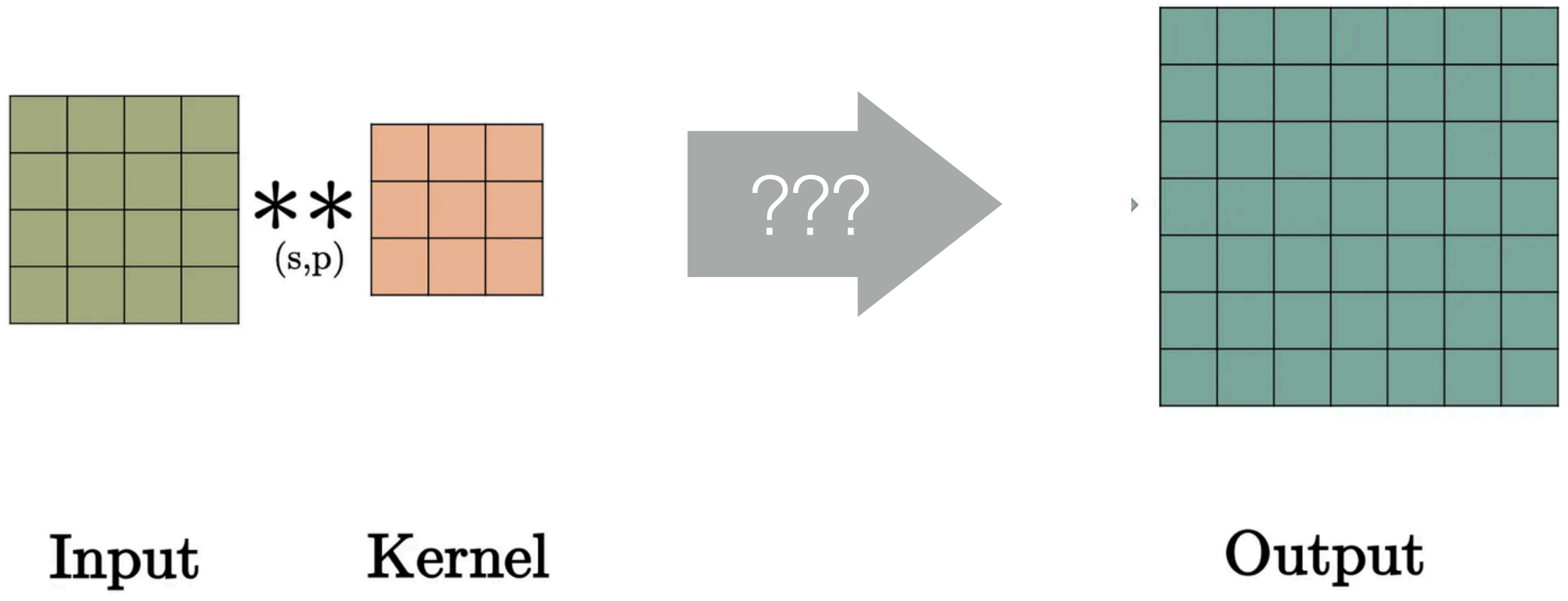
Transposed Convolution

Cons vs TransConv

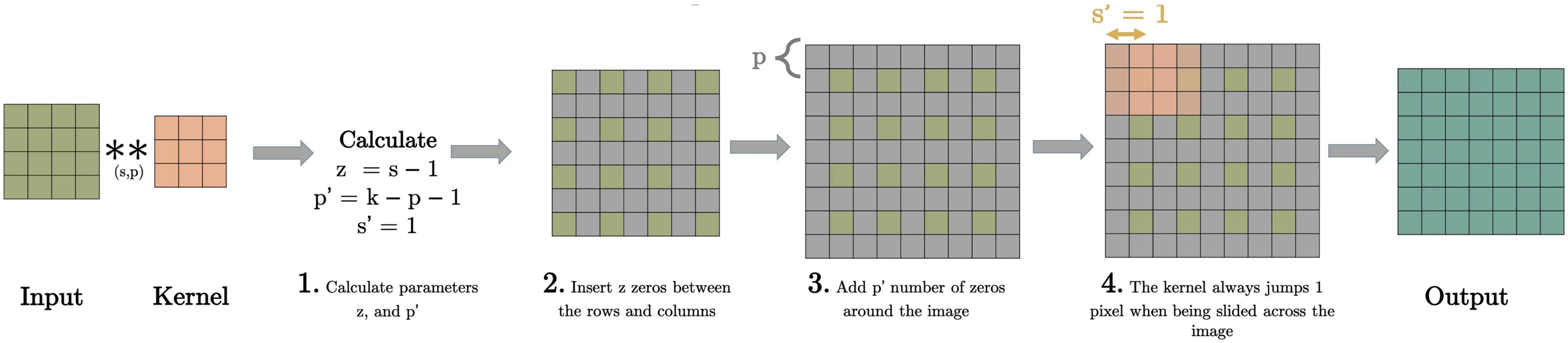
Downsample


Convolution
p
s
*
(s,p)
Input
Kernel
1. Add p number of zeros around the image
2. The kernel jumps s pixels when being slided across the image
Output

Upsample



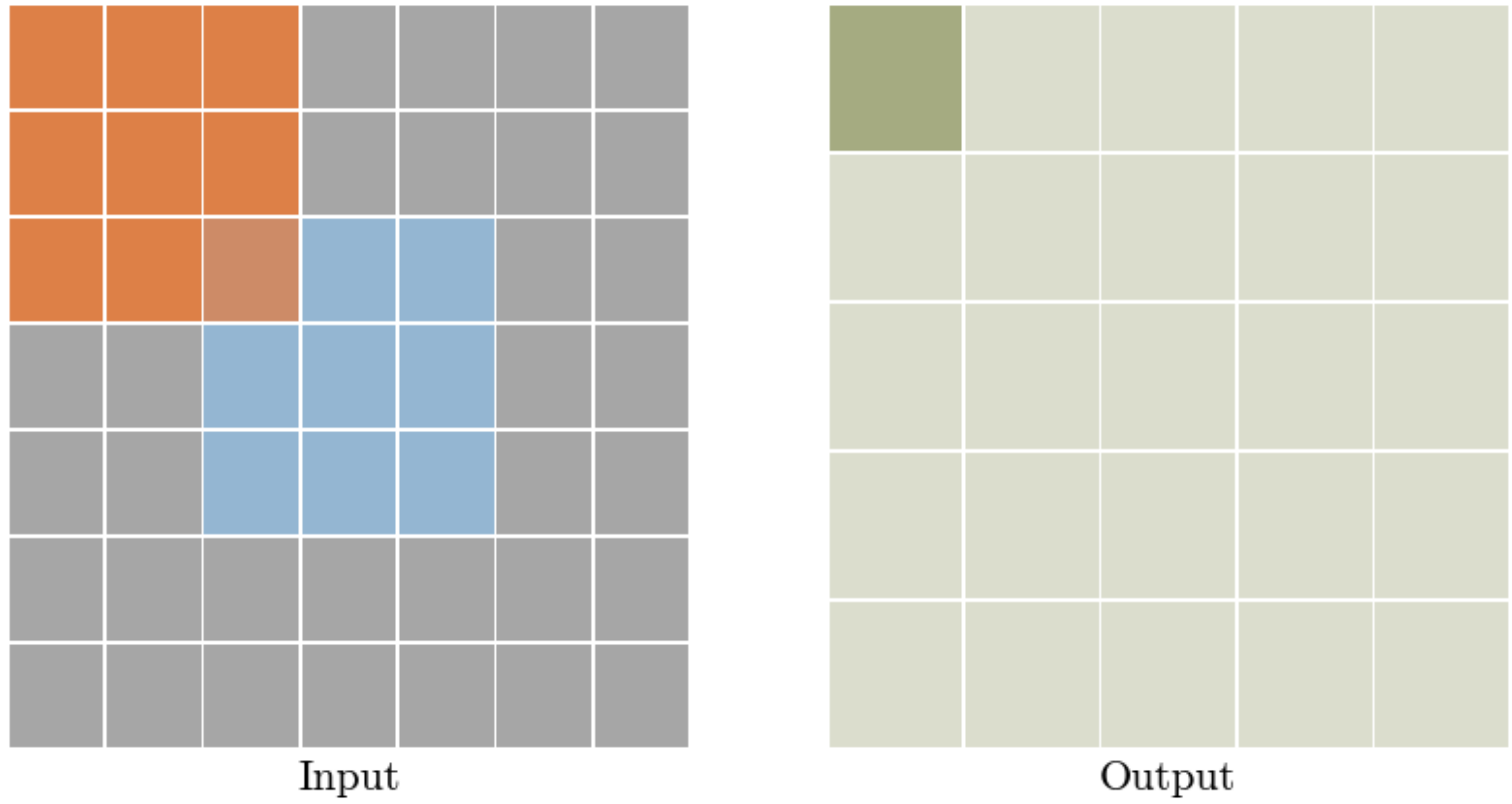
TransConv



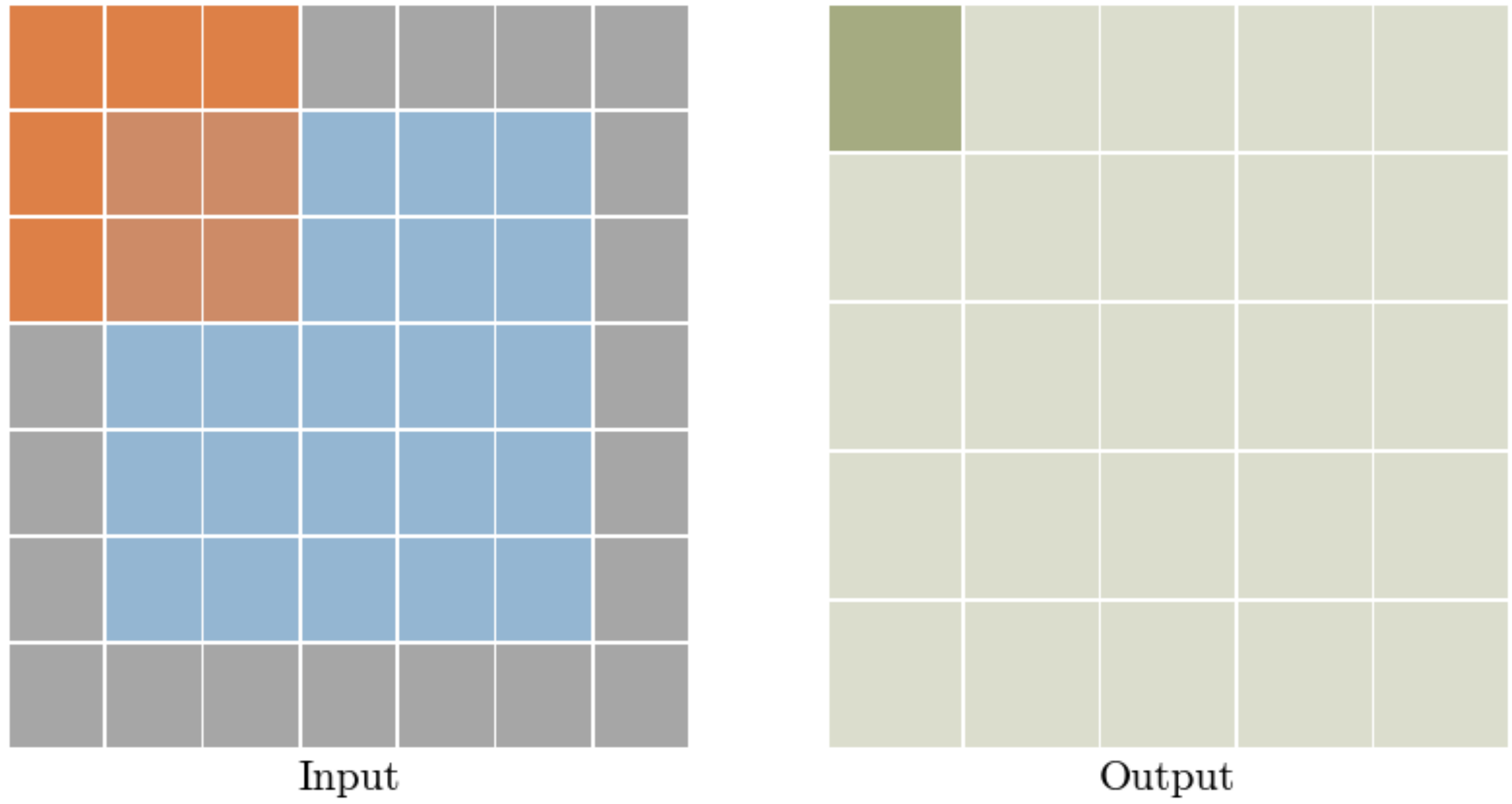
$**_{(s,p)}$
Calculate
$z = s - 1$
$p' = k - p - 1$
$s' = 1$
p
$s' = 1$
Input
Kernel
1. Calculate parameters z, and p'
2. Insert z zeros between the rows and columns
3. Add p' number of zeros around the image
4. The kernel always jumps 1 pixel when being slided across the image
Output

Checkout this article - https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11 (The graphics are taken from there)

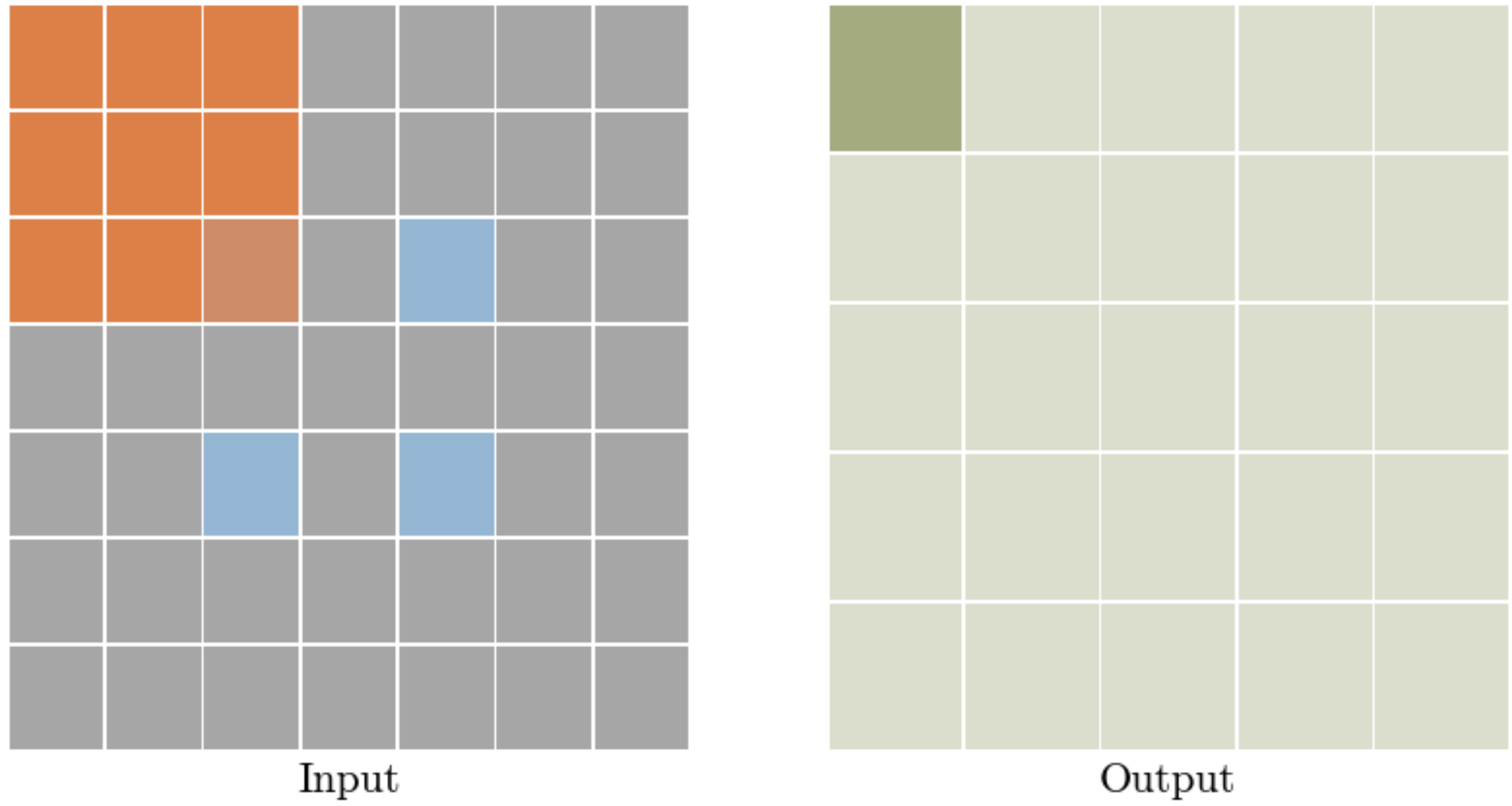
Type: transposed˙conv - Stride: 1 Padding: 0



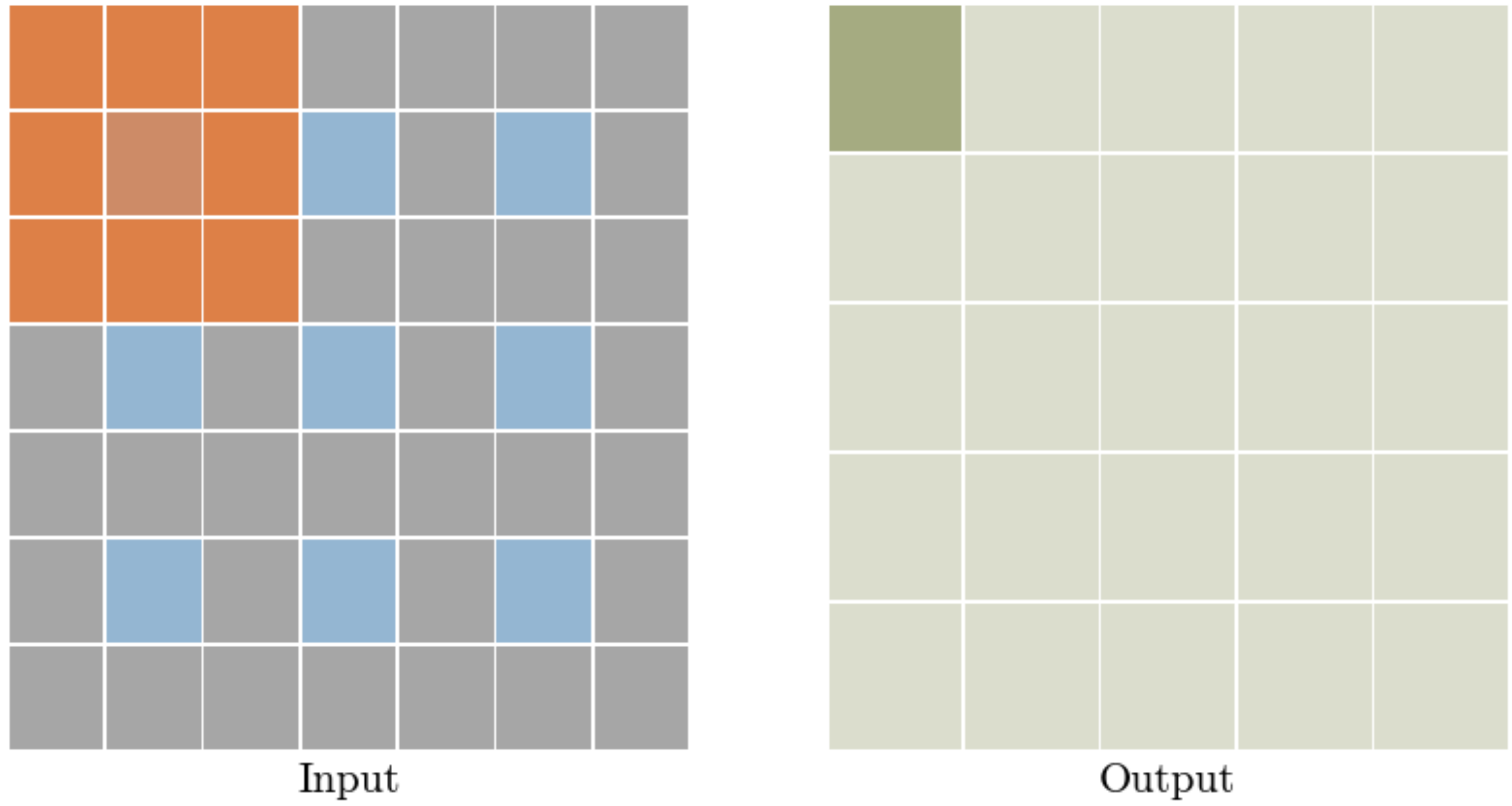
Type: transposed˙conv - Stride: 1 Padding: 1



Type: transposed˙conv - Stride: 2 Padding: 0



Type: transposed˙conv - Stride: 2 Padding: 1



Compute

$$\begin{aligned} z &= s - 1 \\ p' &= k - p - 1 \\ s' &= 1 \end{aligned}$$

Output size

$$o = (i - 1) \times s + k - 2p$$

k = kernel
p = padding
s = stride