Learn from competition

Introduction to Generative Adversarial Network (GAN)

Weiguang Guan (guanw@sharcnet.ca)

SHARCNet/Compute Canada

Reference

- Tensorflow tutorial on Deep Convolutional Generative Adversarial Network (https://www.tensorflow.org/tutorials/generative/dcgan)
- Ian Goodfellow, et al, "**Generative Adversarial Networks**", Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014)
- Tero Karras, et al, "**Analyzing and Improving the Image Quality of StyleGAN**", CVPR 2020 (https://thispersondoesnotexist.com/)
- Vincent Dumoulin and Francesco Visin, "**A guide to convolution arithmetic for deep learning**", 2018, arXiv 1603.07285

Case study

- **Goal**: Generate digits that look like being written by human
- **Method**: train GAN networks
    - Keras/Tensorflow
- **Dataset**: MNIST handwritten digits

The philosophy behind GAN

Well, it may not be fine art. . .
But the prints go for about 50 bucks!

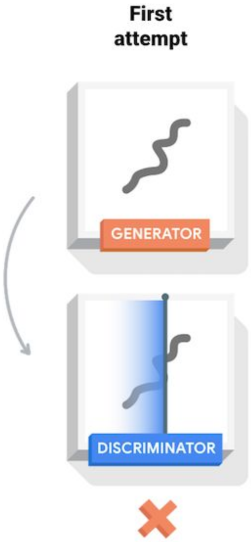
Generator ("the artist")

Battle

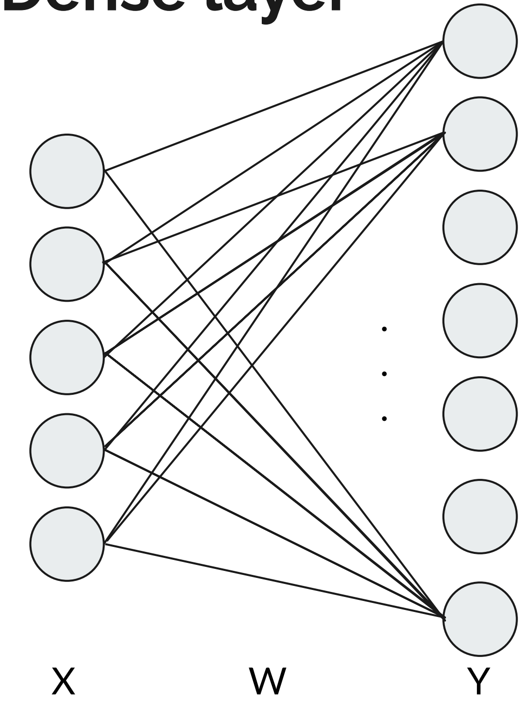
Discriminator ("the art critic")

The philosophy behind GAN

Is it a Cat?



Dense layer



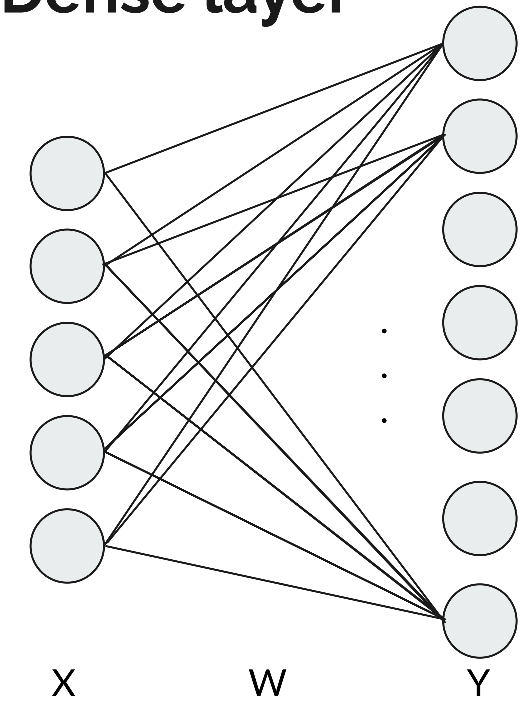
y1 = w11*x1 + w21*x2 + w31*x3 + w41*x4 + w51*x5 + b1

y2 = w12*x1 + w22*x2 + w32*x3 + w42*x4 + w52*x5 + b2

.
.
.

y7 = w17*x1 + w27*x2 + w37*x3 + w47*x4 + w57*x5 + b7

Dense layer



$$\begin{bmatrix} y1 \\ y2 \\ y3 \\ \dots \\ y7 \end{bmatrix} = \begin{bmatrix} w11 & w21 & w31 & w41 & w51 \\ w12 & w22 & w32 & w42 & w52 \\ w13 & w23 & w33 & w43 & w53 \\ \dots & \dots & \dots & \dots & \dots \\ w17 & w27 & w37 & w47 & w57 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ \dots \\ x5 \end{bmatrix} + \begin{bmatrix} b1 \\ b2 \\ b3 \\ \dots \\ b5 \end{bmatrix}$$

Trainable variables [w11, w12, … , w57] and [b1, b2, …, b5]

Animation from github.com/vdumoulin/conv_arithmetic

Convolutional layer (layers.Conv2D)

-1	-1	-1
0	0	0
1	1	1

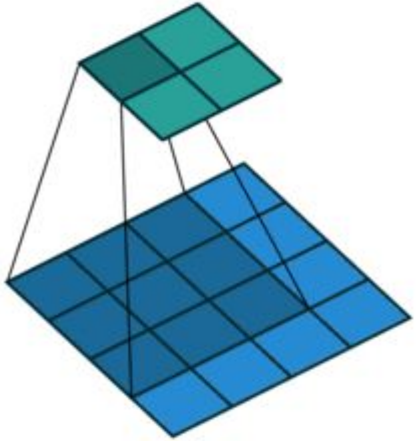
(-1x3) + (-1x0) + (-1x7) +

(0x8) + (0x1) + (0x4) +

(1x5) + (1x0) + (1x1) = **-4**

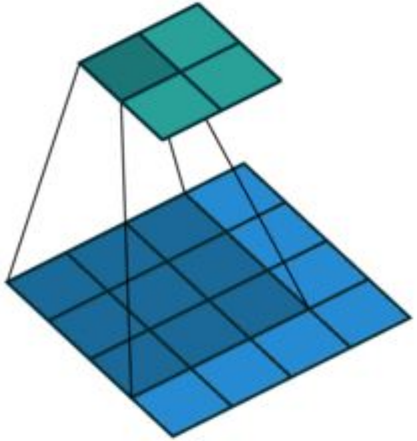
3	0	7	9
8	1	4	0
5	0	1	9
6	0	0	3

-4	-6
-6	-2

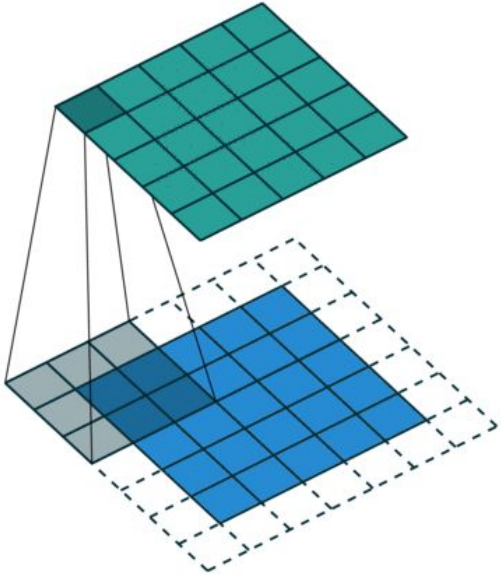


Animations from github.com/vdumoulin/conv_arithmetic

Convolutional layer (layers.Conv2D)



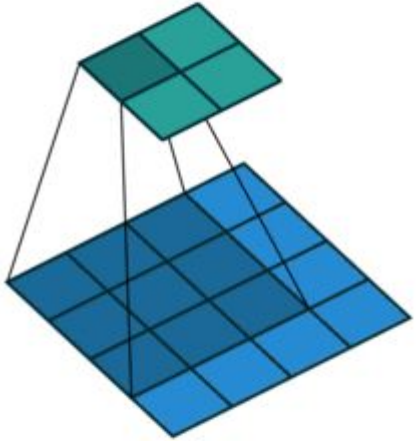
No padding, stride=1



Padding with zeros, stride=1

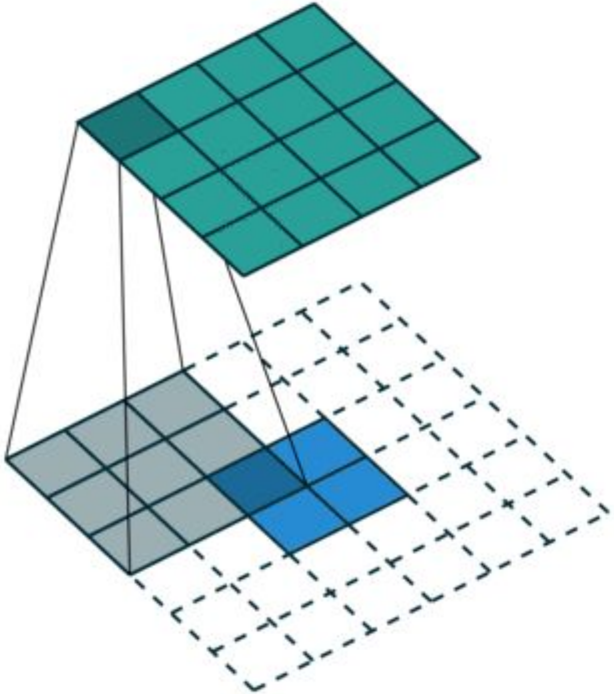
Animations from github.com/vdumoulin/conv_arithmetic

Transposed convolutional layer (layers.Conv2DTranspose)



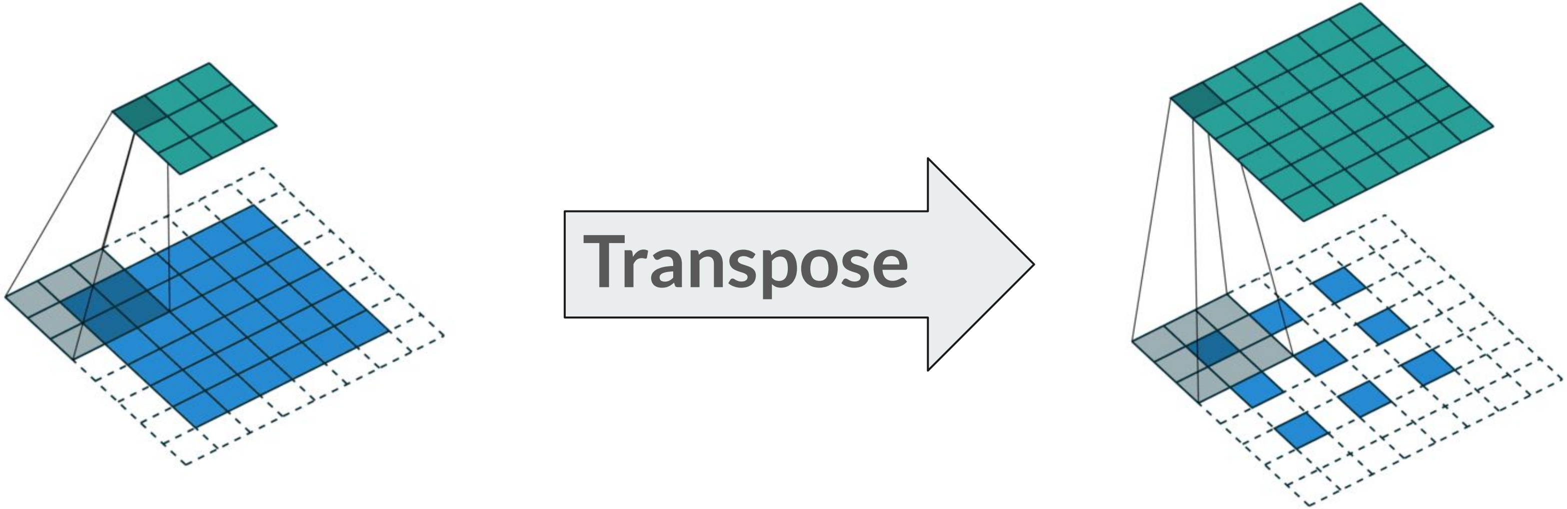
No padding, stride=1

Transpose



Animations from github.com/vdumoulin/conv_arithmetic

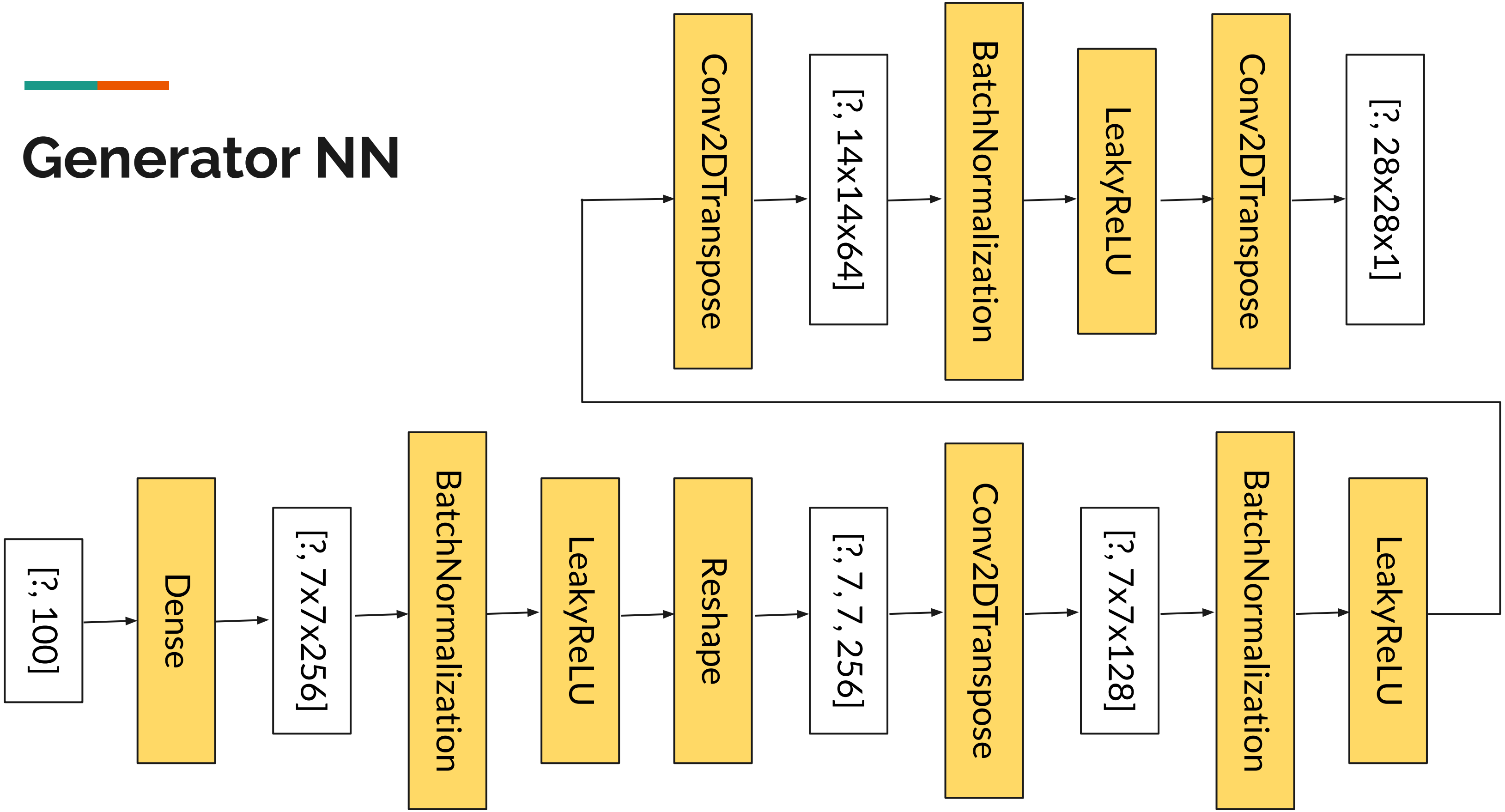
Transposed convolutional layer (layers.Conv2DTranspose)



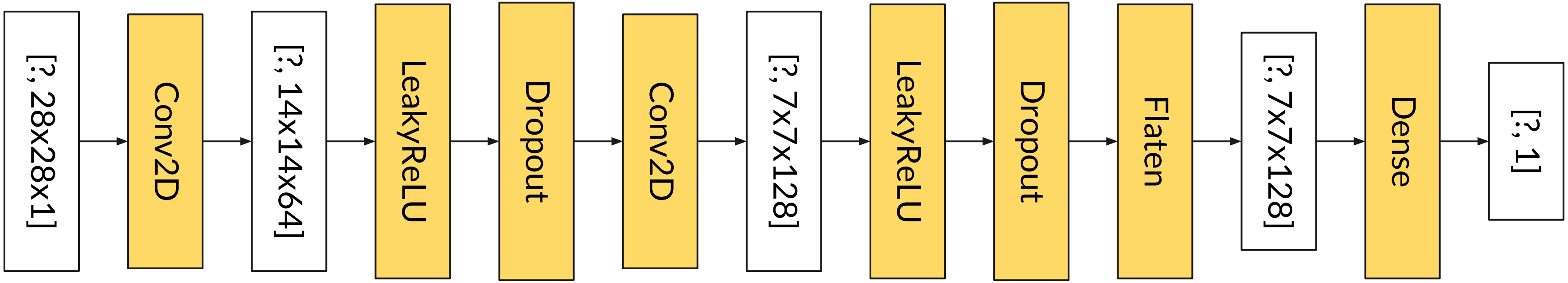
Transpose

Padding with zeros, stride=2

Generator NN



Discriminator NN



Loss function

tf.keras.losses.BinaryCrossentropy

1. Logits → probabilities (Let L be the output of the discriminator)

   q(1) = sigmod(L) = 1/(1+exp(-L))

   q(0) = 1 - q(1)

2. Cross entropy

   ∑p(x) log(q(x))

Case study

- Origin of the source: https://www.tensorflow.org/tutorials/generative/dcgan
- Complete course material that has been tested on Graham cluster: https://staff.sharcnet.ca/guanw/2021/dcgan.tar