Deep Learning for Humanists @DHSI2022 – Day 3

Hoyeol Kim

These slides will be shared on GitHub

Tensor

Rank	Type	Examples
0	Scalar	[1]
1	Vector	[1,1]
2	Matrix	[[1,1], [1,1]]
3	3 tensor	[[[1,1], [1,1]], [[1,1], [1,1]]]
n	N tensor	N*[…]]]]…

Tensor

Sentences
Hi Adam
Hi Kate
Hi Setsuko

Unique Words	Index	One Hot Encoding Vector
Hi	0	[1,0,0,0]
Adam	1	[0,1,0,0]
Kate	2	[0,0,1,0]
Setsuko	3	[0,0,0,1]

Tensor

Unique Words	Vector Representations
Hi Adam	[[1,0,0,0], [0,1,0,0]]
Hi Kate	[[1,0,0,0], [0,0,1,0]]
Hi Setsuko	[[1,0,0,0], [0,0,0,1]]

Mini batch input will be in the following:

Hi Adam Hi Kate Hi Setsuko

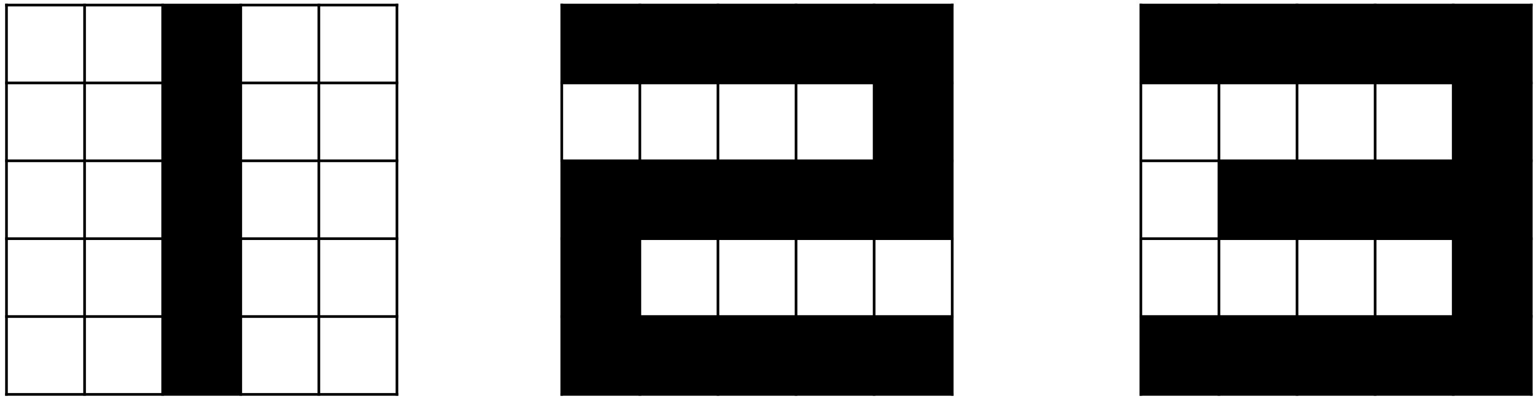
[[[1,0,0,0], [0,1,0,0]], [[1,0,0,0], [0,0,1,0]], [[1,0,0,0], [0,0,0,1]]]

(3, 2, 4) 3d tensor

Tensor in Grayscale Image

(3, 5, 5)

3 images　5 rows　5 columns



Tensor in RGB Color Image

(3, 5, 5, 3)

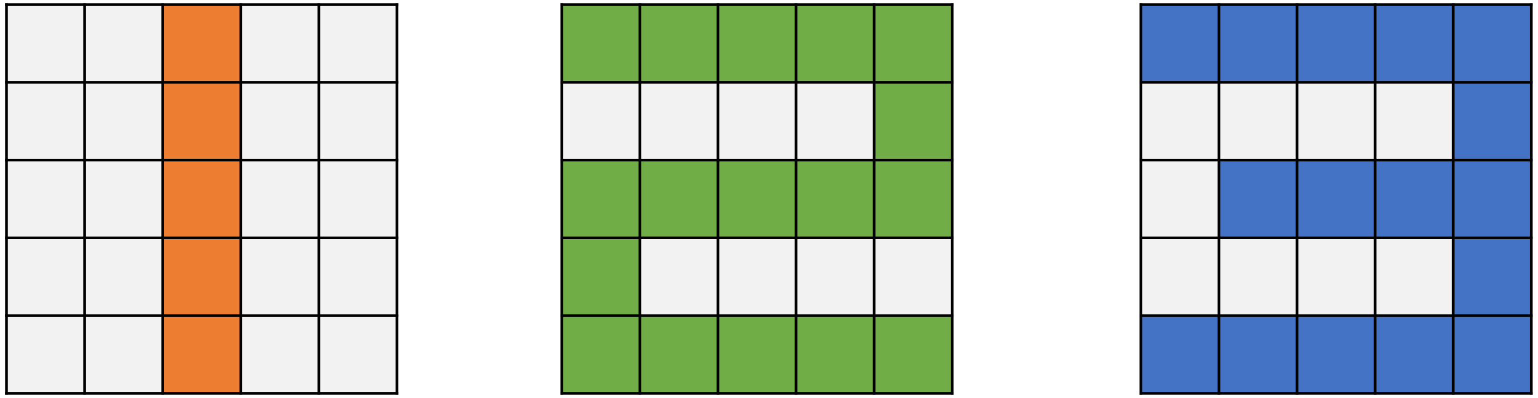
4d tensor

3 images

5 rows

5 columns

red, green, blue



Tensor

PyTorch: torch.tensor()

https://pytorch.org/docs/stable/tensors.html

TensorFlow: tf.Tensor()

https://www.tensorflow.org/api_docs/python/tf/Tensor

Albumentations: albumentations.pytorch.transforms.ToTensorV2()

https://albumentations.ai/docs/api_reference/pytorch/transforms/

TensorFlow: tf.convert_to_tensor()

https://www.tensorflow.org/api_docs/python/tf/convert_to_tensor

Mean Absolute Error = $\frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i |$

Mean Squared Error = $\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Binary Cross Entropy (Log Loss) =

$$-\frac{1}{n}\sum_{i=1}^{n} \mathrm{y_i} * \log(\hat{y}_i) + (1 - yi) * \log(1 - (\hat{y}_i))$$

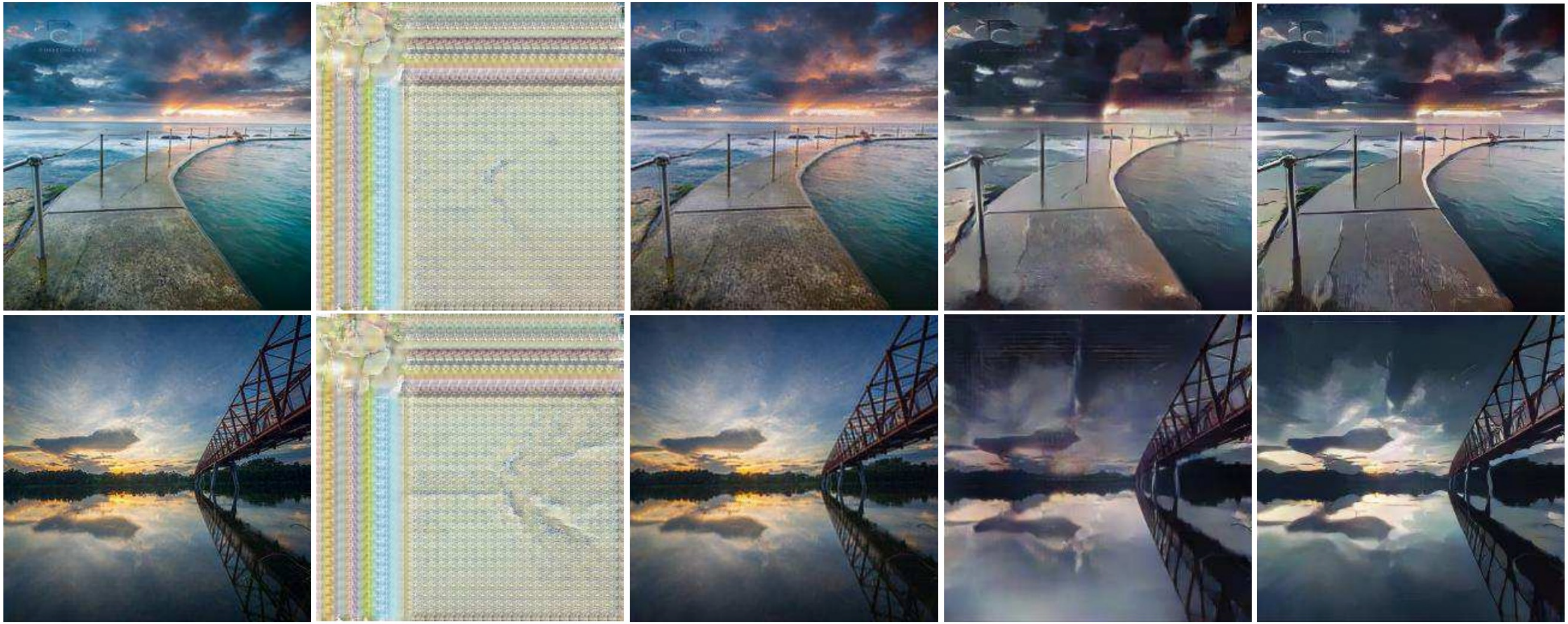
L1 & L2 Loss Functions

$$L1LossFunction = \sum_{i=1}^{n} |\, y_i - \hat{y}_i|$$

$$L2LossFunction = \sum_{i=1}^{n} (\, y_i - \hat{y}_i)^2$$

Loss

(a) Input Photo (b) Without Initialization (c) With L_2 loss (d) Without edge loss (e) CartoonGAN (ours)



The image is from "https://openaccess.thecvf.com/content_cvpr_2018/papers/Chen_CartoonGAN_Generative_Adversarial_CVPR_2018_paper.pdf"

PatchGAN

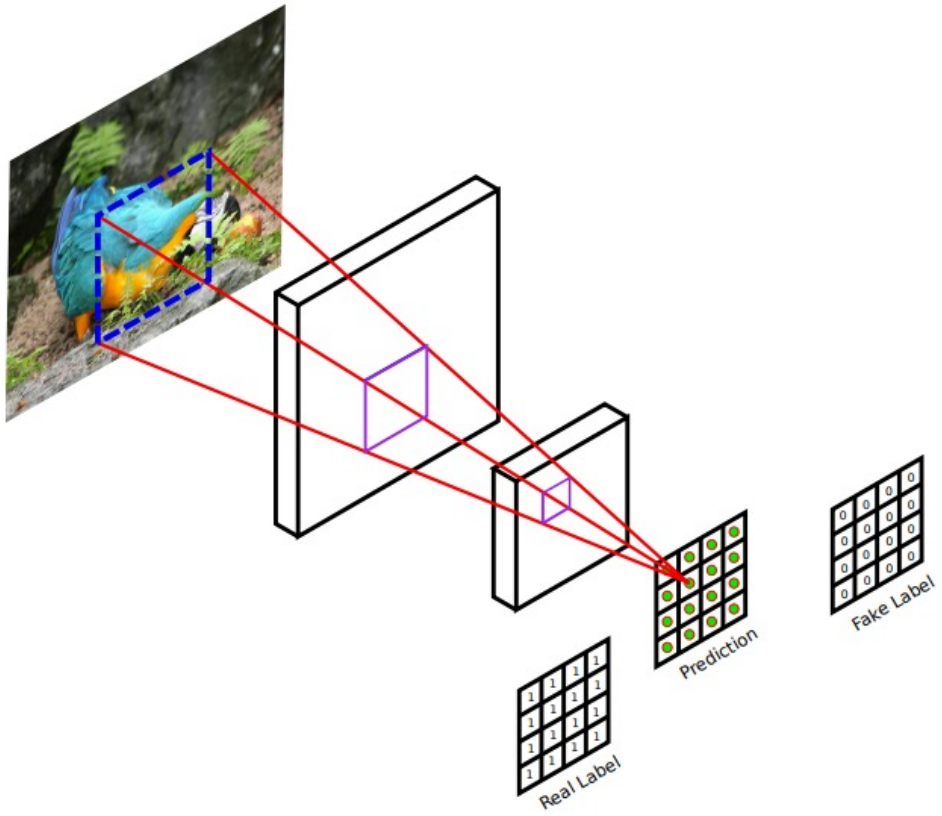
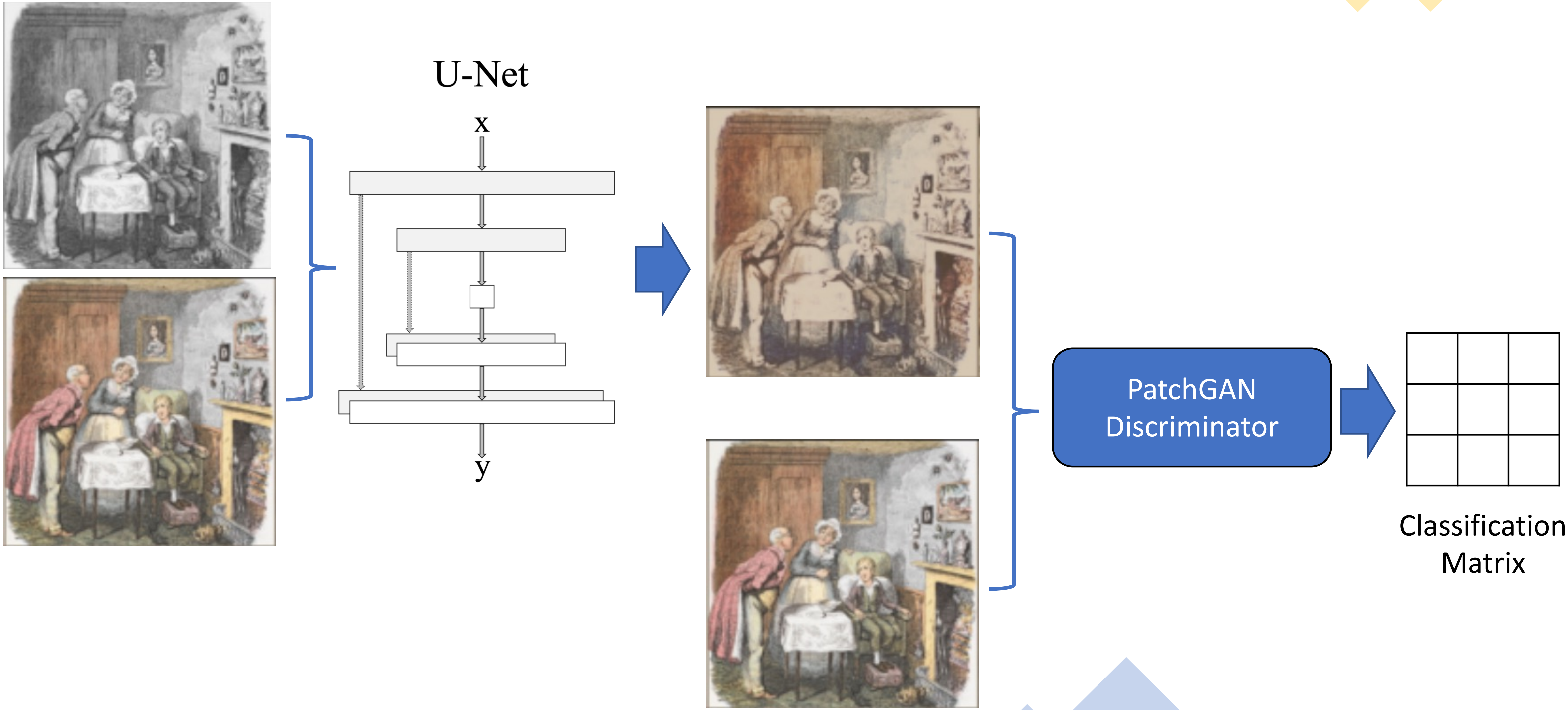


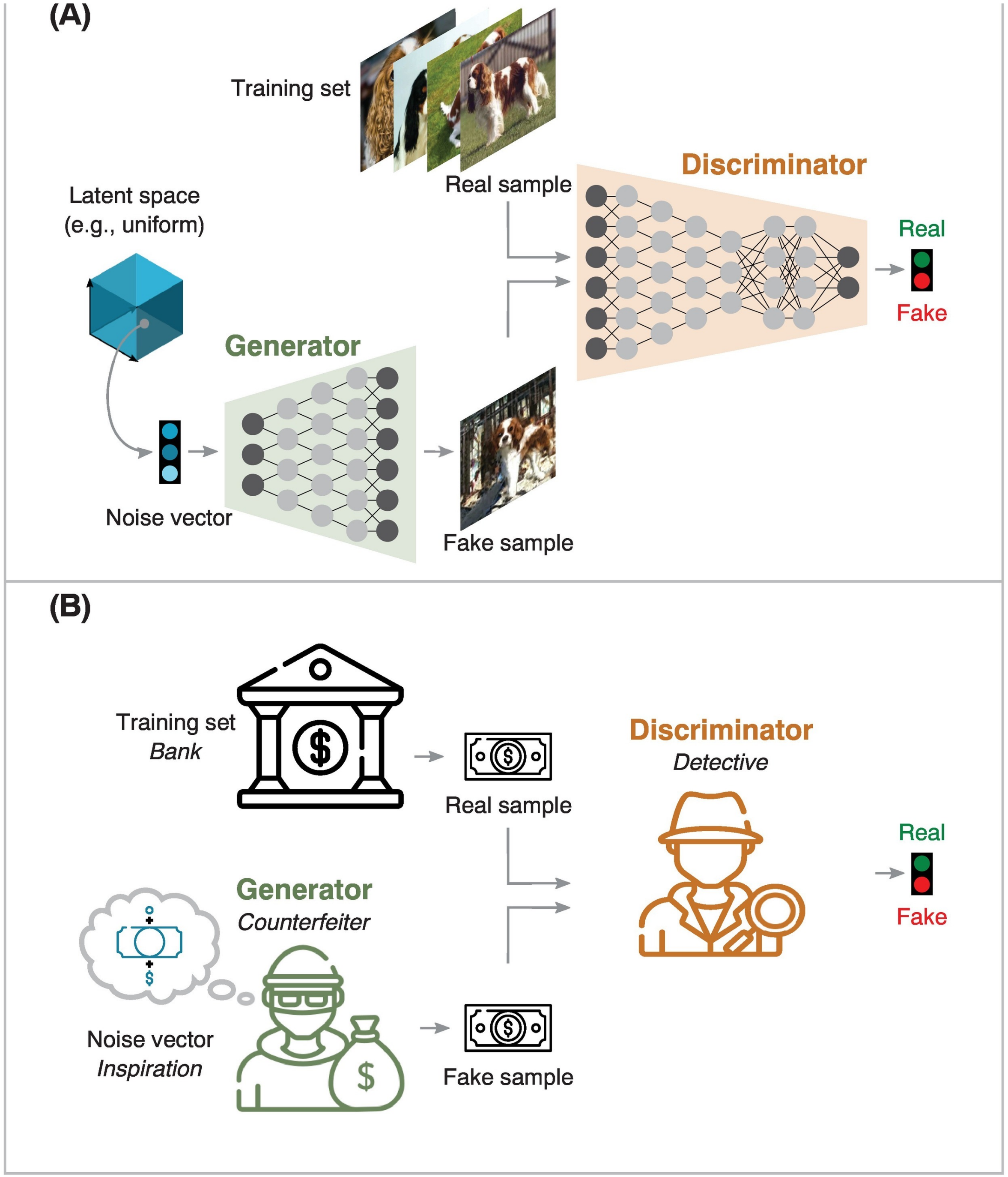
Image credit: Geoffrey Hinton

- PatchGAN is used in a variety of GAN-based models such as conditional GAN and StarGAN.
- L1 & L2 loss can create blurry images in image generation tasks. Using PatchGAN, it is possible to generate detailed images with high frequency by limiting the size of patch for structures.
- PatchGAN decides whether or not outputs are true created by a generator.
- PatchGAN is a convnet, which is different from a regular GAN discriminator.
- A regular GAN discriminator returns a single scalar output after getting a 256*256 image as an input image, whereas a PatchGAN discriminator returns arrays of X (N*N) after getting a 256*256 image as an input image.

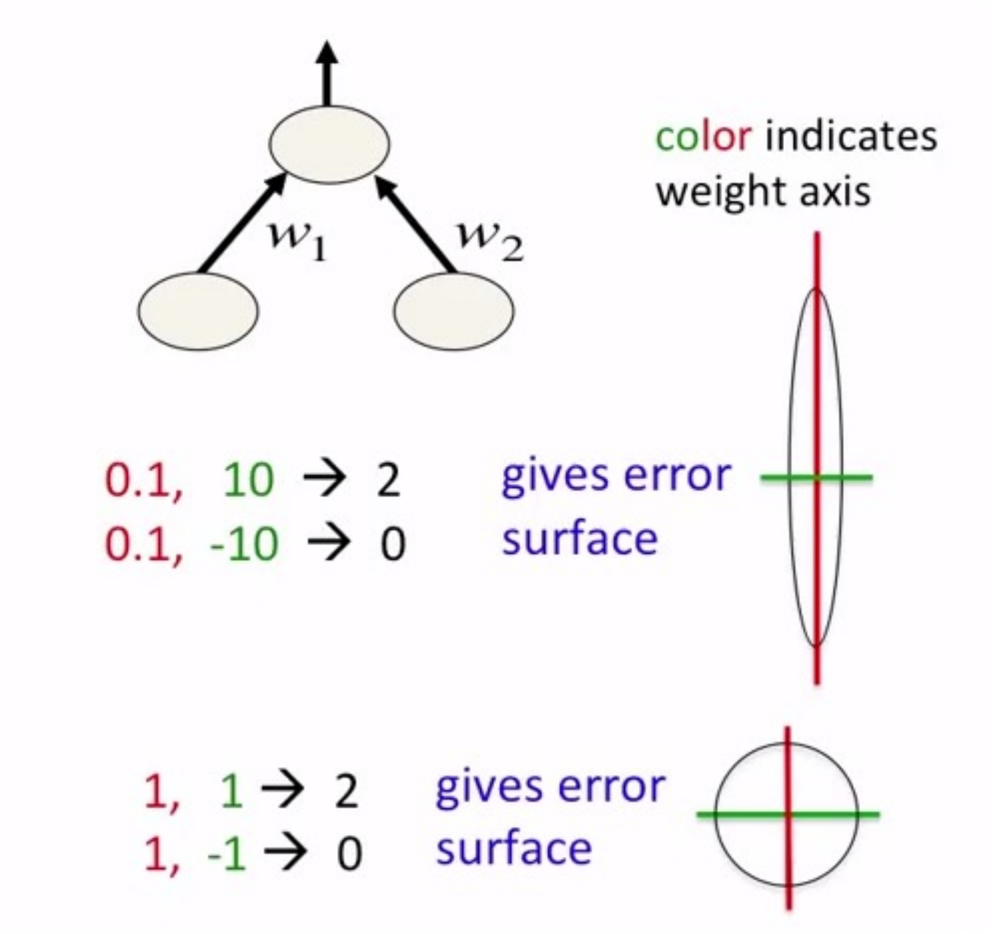
Pix2Pix



GAN


(A)
Training set
Real sample
Discriminator
Latent space
(e.g., uniform)
Real
Fake
Generator
Noise vector
Fake sample
(B)
Training set
Bank
Real sample
Discriminator
Detective
Real
Fake
Generator
Counterfeiter
Noise vector
Inspiration
Fake sample

Normalization



- The predictions of neural networks would be inaccurate if features were big in scale.
- If values were very high, it would take a lot of computation time and memory. For example, each pixel normally ranges from 0 to 255.

Image credit: Geoffrey Hinton

Normalization

Content StyleNet IN (ours) StyleNet BN

Image credit: Geoffrey Hinton


Any Questions?