شبکه‌های مولد تخاصمی
Generative Adversarial Networks (GAN)

Alireza Akhavanpour

Akhavanpour.ir
CLASS.VISION

Agenda

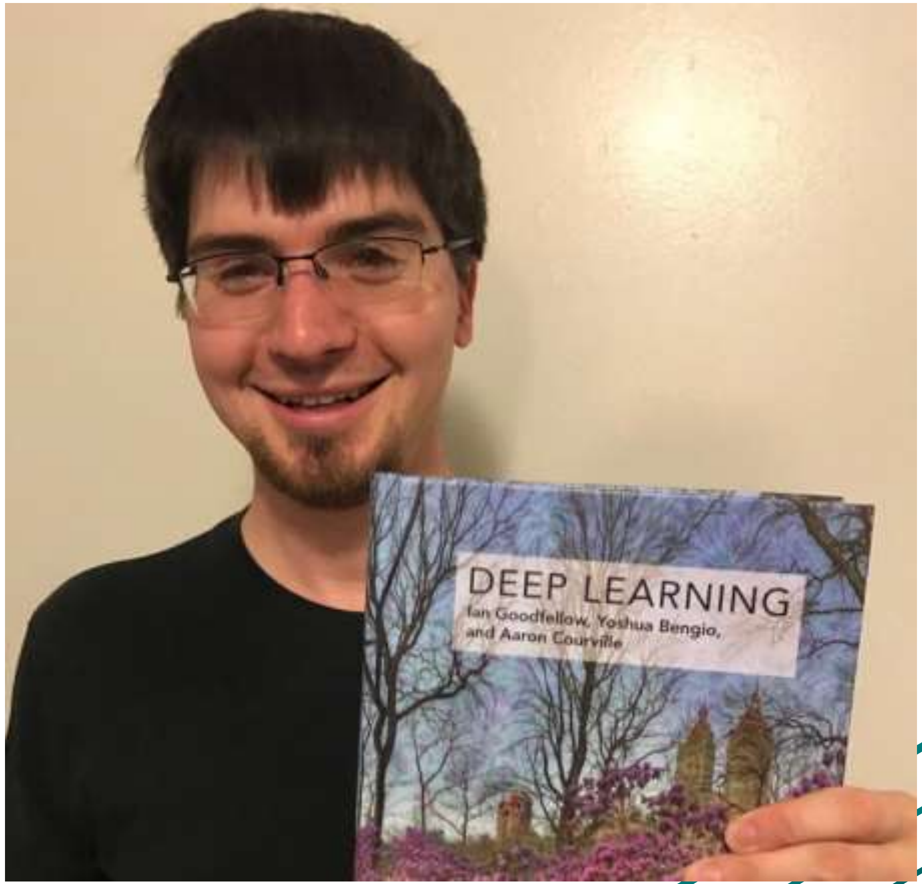
- Simple GAN
- Transposed convolution & DCGAN
- TF2 for researchers! (pre-requirement)
- Auto-Encoders
- U-Net & Segmentation
- GAN overriding Model.train_step
- Pix2pix
- Cycle-GAN

Agenda

- **Simple GAN**
- Transposed convolution & DCGAN
- TF2 for researchers! (pre-requirement)
- Auto-Encoders
- U-Net & Segmentation
- GAN overriding Model.train_step
- Pix2pix
- Cycle-GAN

What is GAN?

- They were first introduced by Ian Goodfellow *et al.* in 2014.
- Can learn to draw samples from a model that is similar to data that we give them.

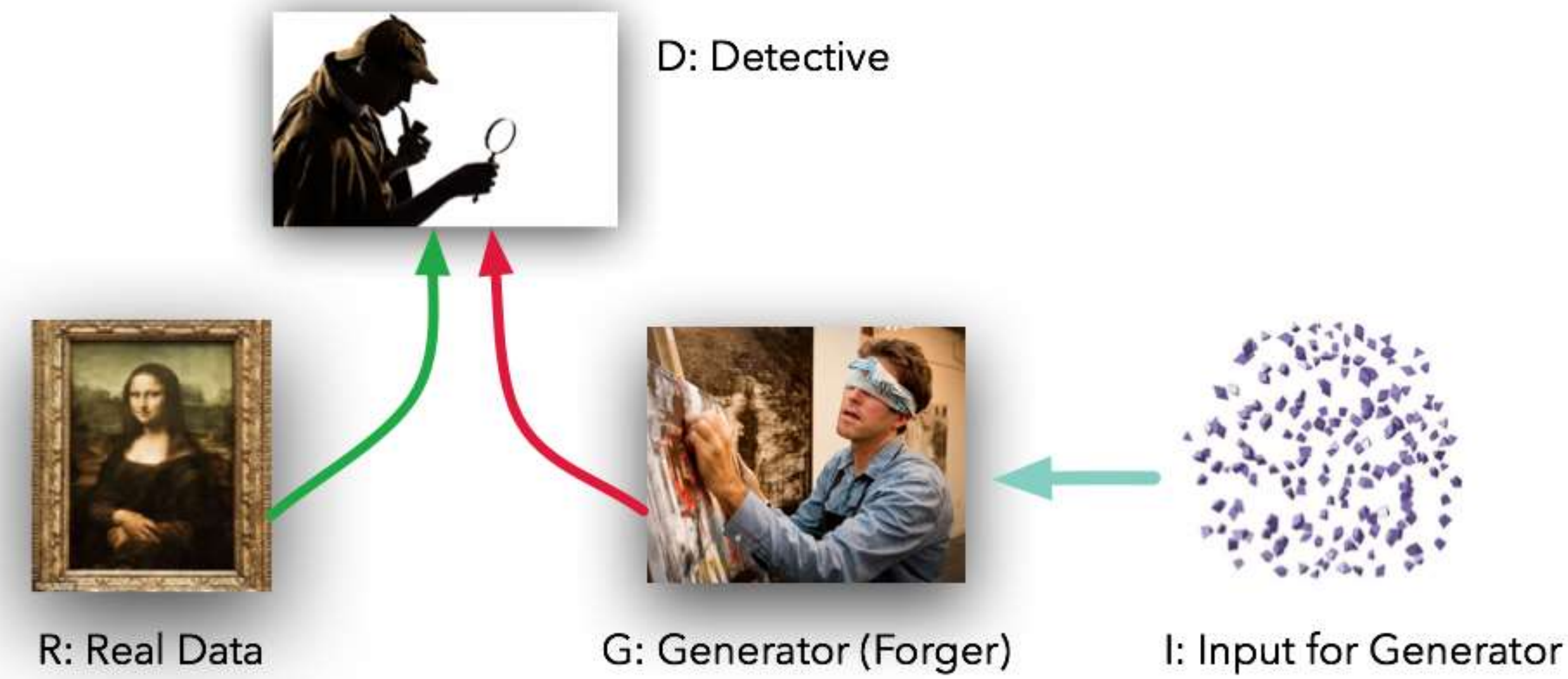


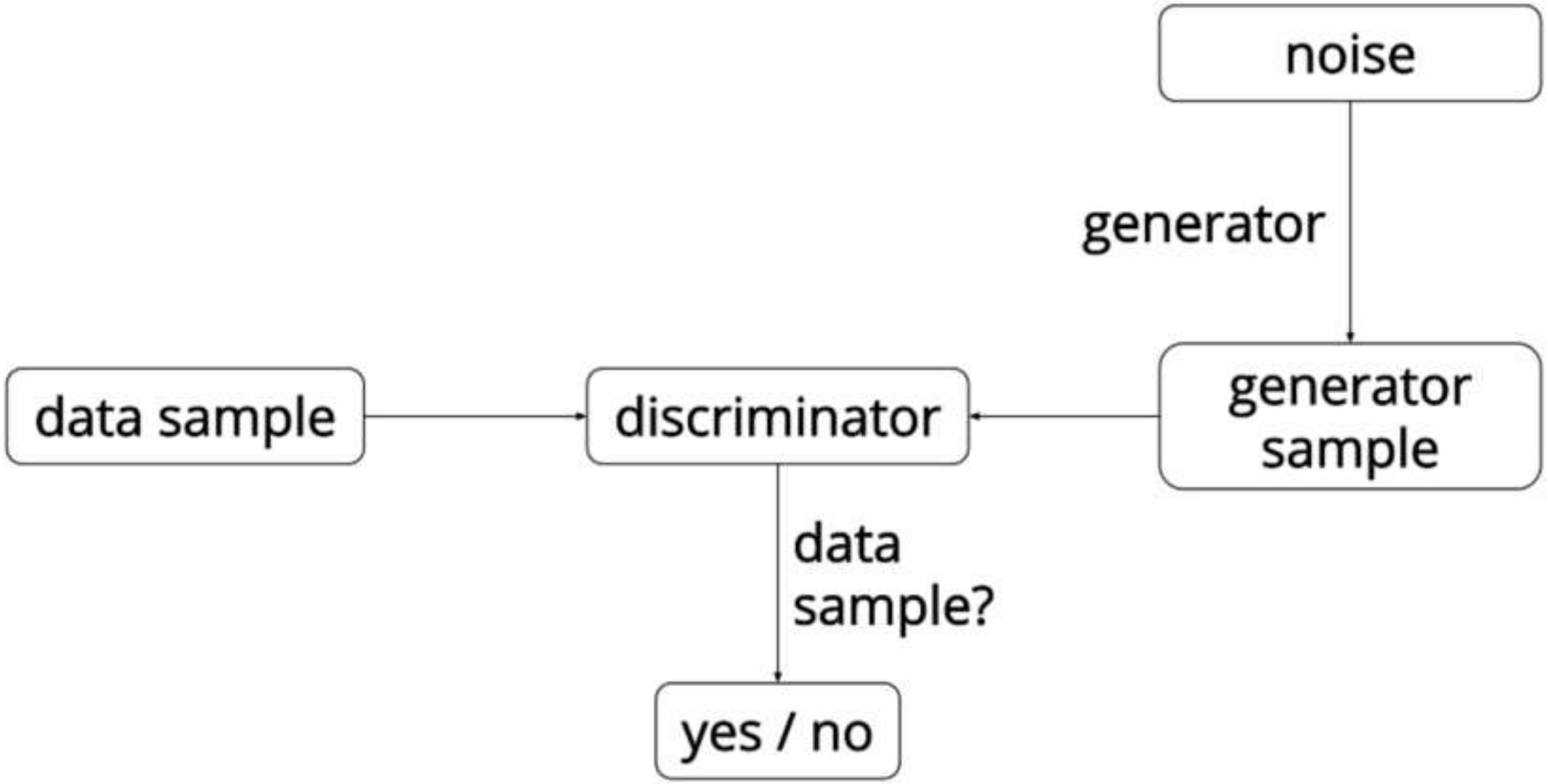
They have achieved some incredible results:

Some the latest results from NVIDIA

https://research.nvidia.com/sites/default/files/pubs/2017-10_Progressive-Growing-of/karras2018iclr-paper.pdf

Generative Adversarial Networks





Source: *https://ishmaelbelghazi.github.io/ALI*

Reza Zadeh @Reza_Zadeh · 15m

GANs in nature: Cuckoos (Generator) lay eggs in Warbler (Discriminator) bird's nests, tricking warblers to raise cuckoo chicks. Warblers are evolving to recognize cuckoo eggs to destroy them. Simultaneously, cuckoos evolve to produce more warbler-like eggs cam.ac.uk/research/featu...

https://www.cam.ac.uk/research/features/the-reed-warbler-and-the-cuckoo-an-escalating-game-of-trickery-and-defence

GAN Lab

https://poloclub.github.io/ganlab/

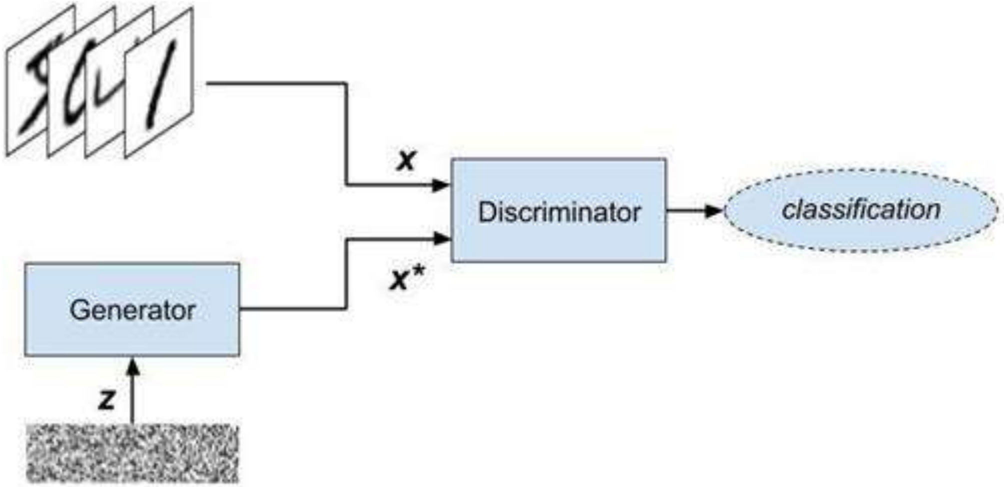
پیاده سازی ۱

http://nbviewer.jupyter.org/github/alireza-akhavan/class.vision/blob/master/55-simple-GAN.ipynb

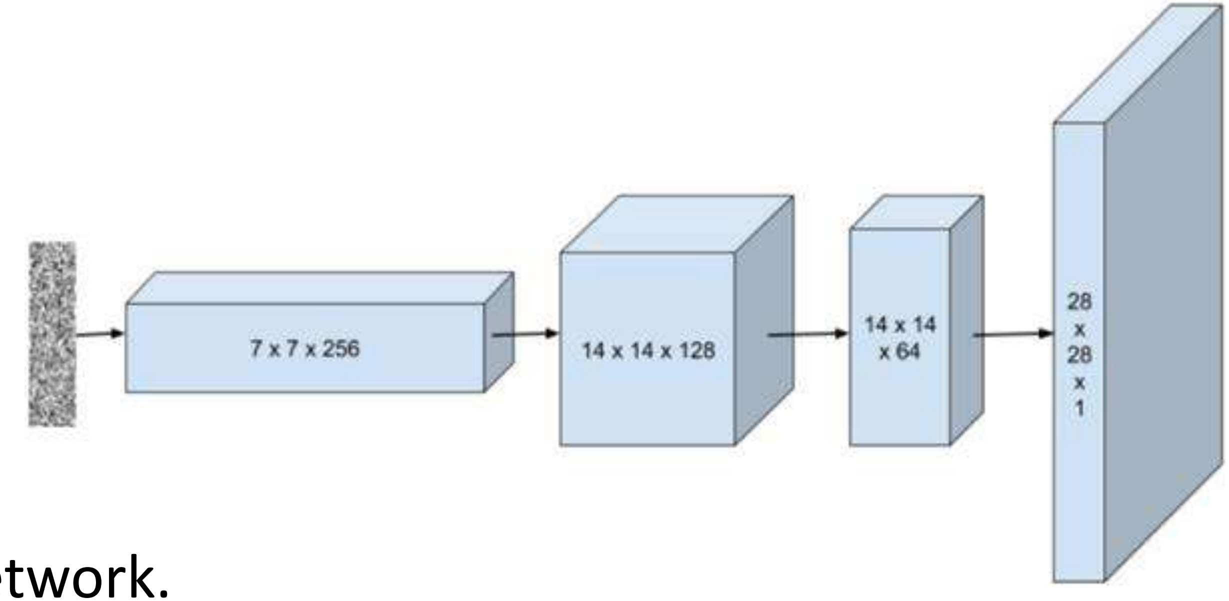
Agenda

- Simple GAN
- **Transposed convolution & DCGAN**
- TF2 for researchers! (pre-requirement)
- Auto-Encoders
- U-Net & Segmentation
- GAN overriding Model.train_step
- Pix2pix
- Cycle-GAN

Deep Convolutional GAN



The Generator Network with Transposed convolutions



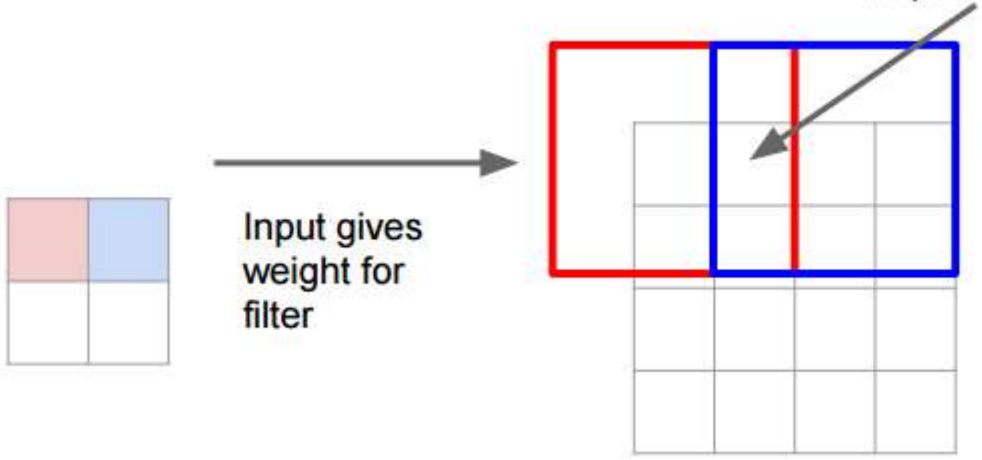
The Generator Network.

The Generator takes in a random noise vector as input and produces a 28 x 28 x 1 image. It does so by multiple layers of transposed convolutions. In between the convolutional layers, we apply batch normalization to stabilize the training process. (Image is not to scale.)

3 x 3 "deconvolution", stride 2 pad 1

Sum where output overlaps

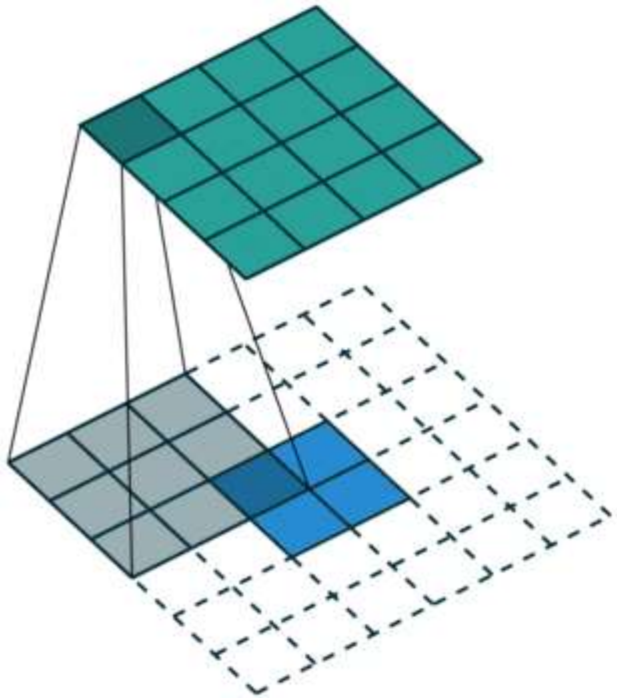
Same as backward pass for normal convolution!



Input gives weight for filter

Input: 2 x 2

Output: 4 x 4



Transposed convolution

Input

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

Kernel

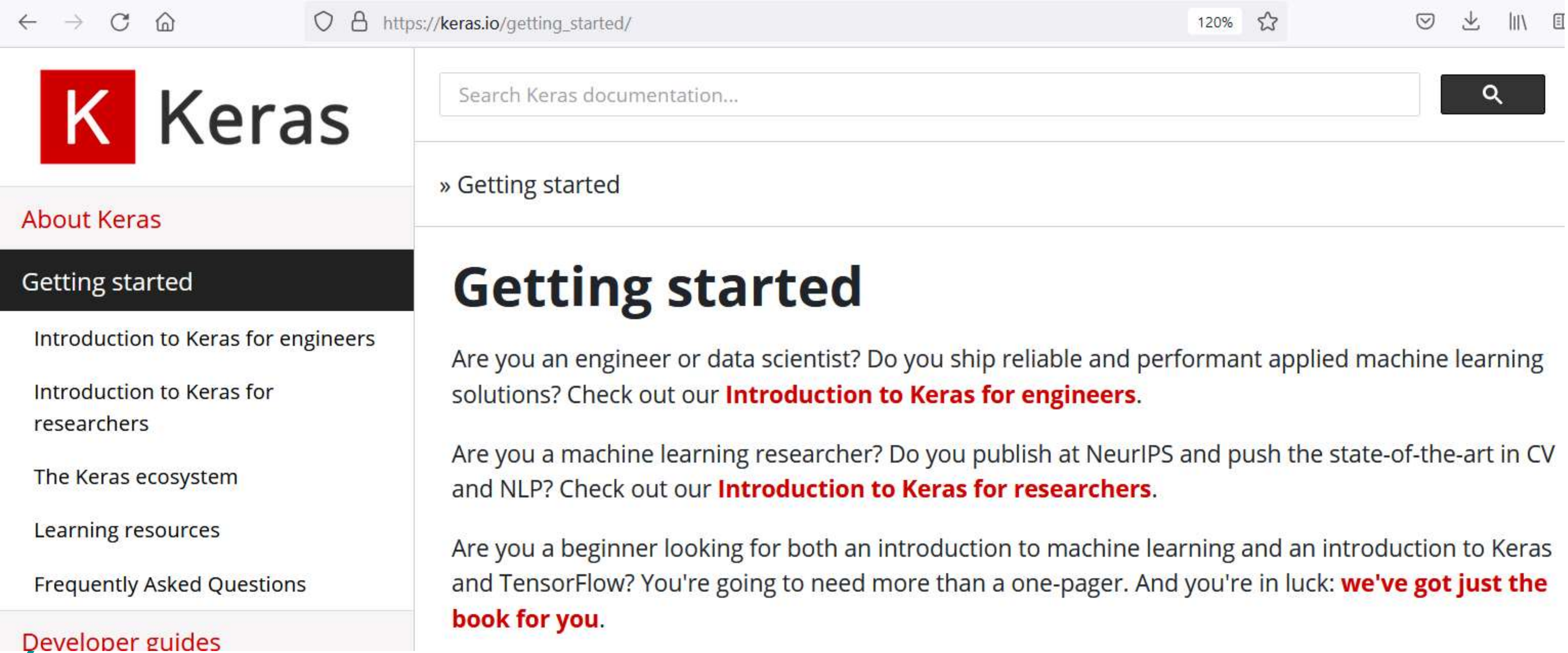
1	1	1
1	1	1
1	1	1

Output

1	2	3	3	2	1
2	4	6	6	4	2
3	6	9	9	6	3
3	6	9	9	6	3
2	4	6	6	4	2
1	2	3	3	2	1

Agenda

- Simple GAN
- Transposed convolution & DCGAN
- **TF2 for researchers! (pre-requirement)**
- Auto-Encoders
- U-Net & Segmentation
- GAN overriding Model.train_step
- Pix2pix
- Cycle-GAN


https://keras.io/getting_started/

Keras

Search Keras documentation...

About Keras

Getting started

Introduction to Keras for engineers

Introduction to Keras for researchers

The Keras ecosystem

Learning resources

Frequently Asked Questions

Developer guides

» Getting started

Getting started

Are you an engineer or data scientist? Do you ship reliable and performant applied machine learning solutions? Check out our **Introduction to Keras for engineers**.

Are you a machine learning researcher? Do you publish at NeurIPS and push the state-of-the-art in CV and NLP? Check out our **Introduction to Keras for researchers**.

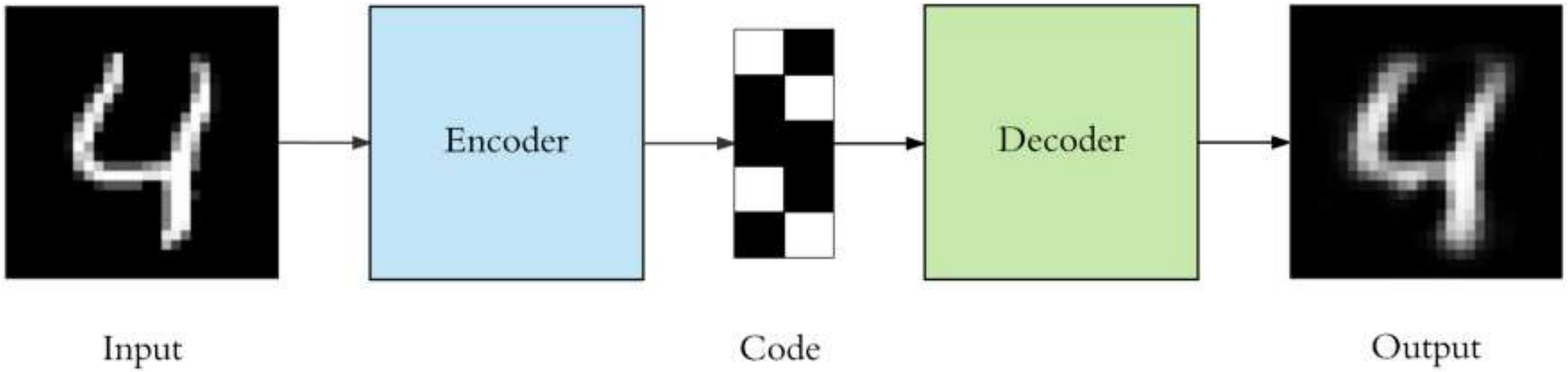
Are you a beginner looking for both an introduction to machine learning and an introduction to Keras and TensorFlow? You're going to need more than a one-pager. And you're in luck: **we've got just the book for you**.

https://keras.io/getting_started/intro_to_keras_for_researchers

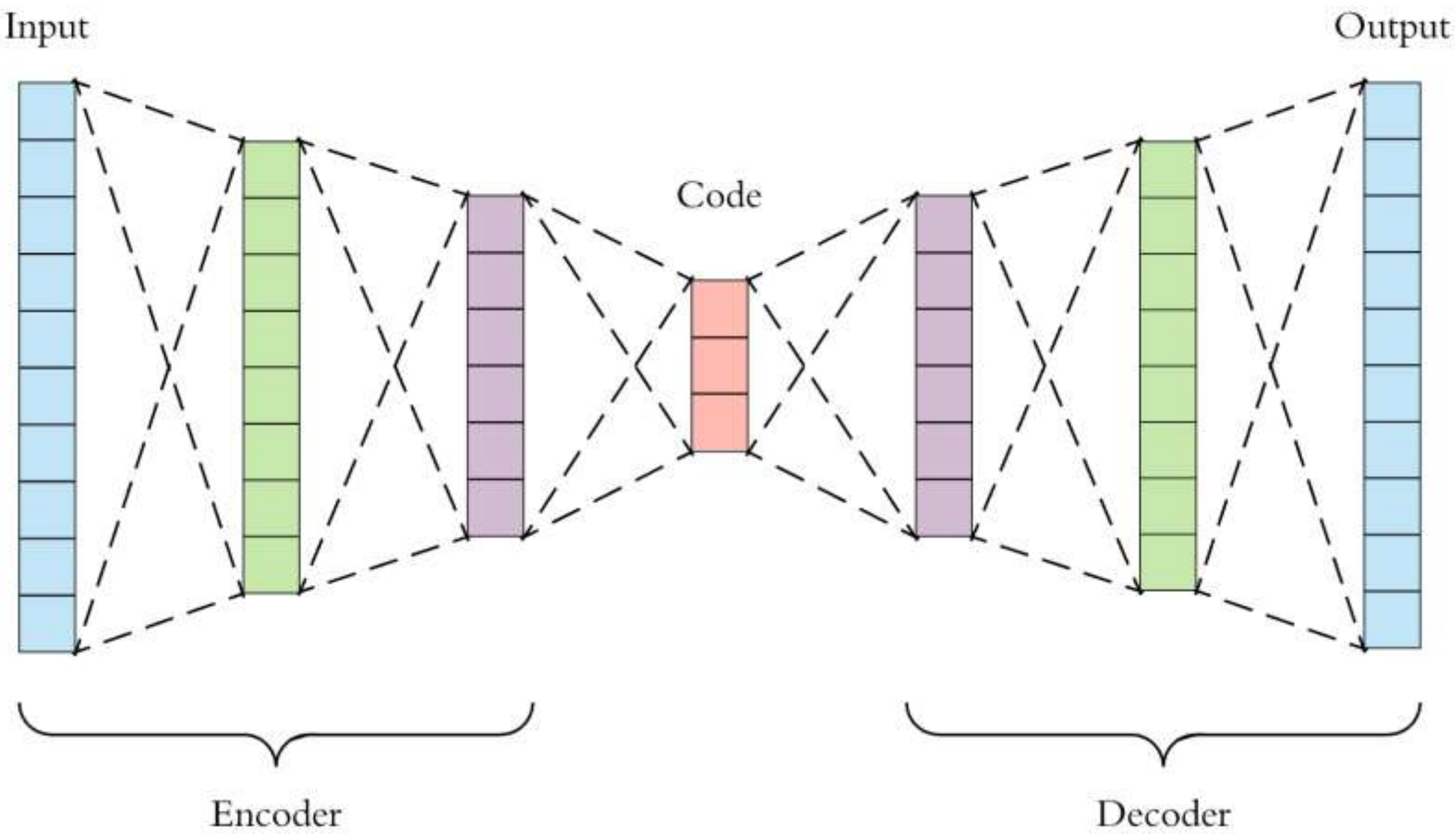
Agenda

- Simple GAN
- Transposed convolution & DCGAN
- TF2 for researchers! (pre-requirement)
- **Auto-Encoders**
- U-Net & Segmentation
- GAN overriding Model.train_step
- Pix2pix
- Cycle-GAN

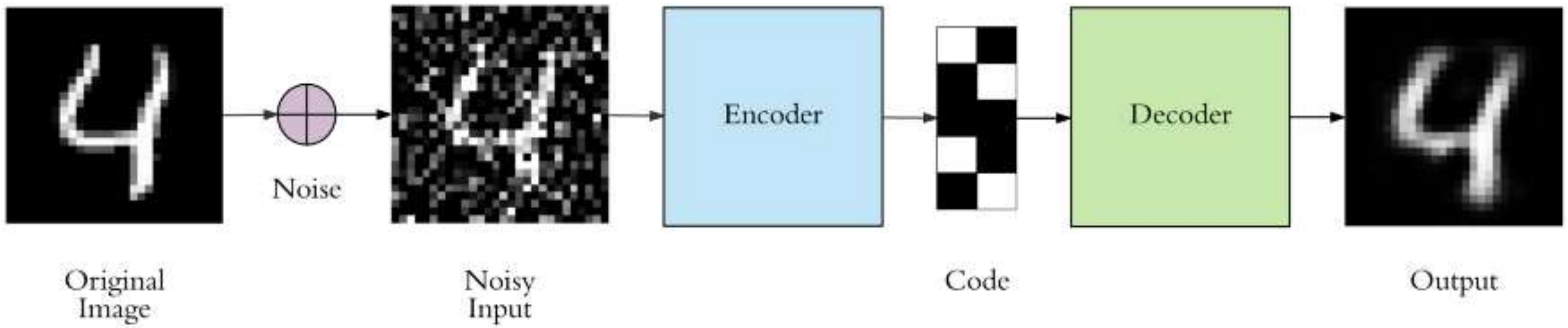
What is auto encoder?



What is auto encoder?



Denoising Autoencoders



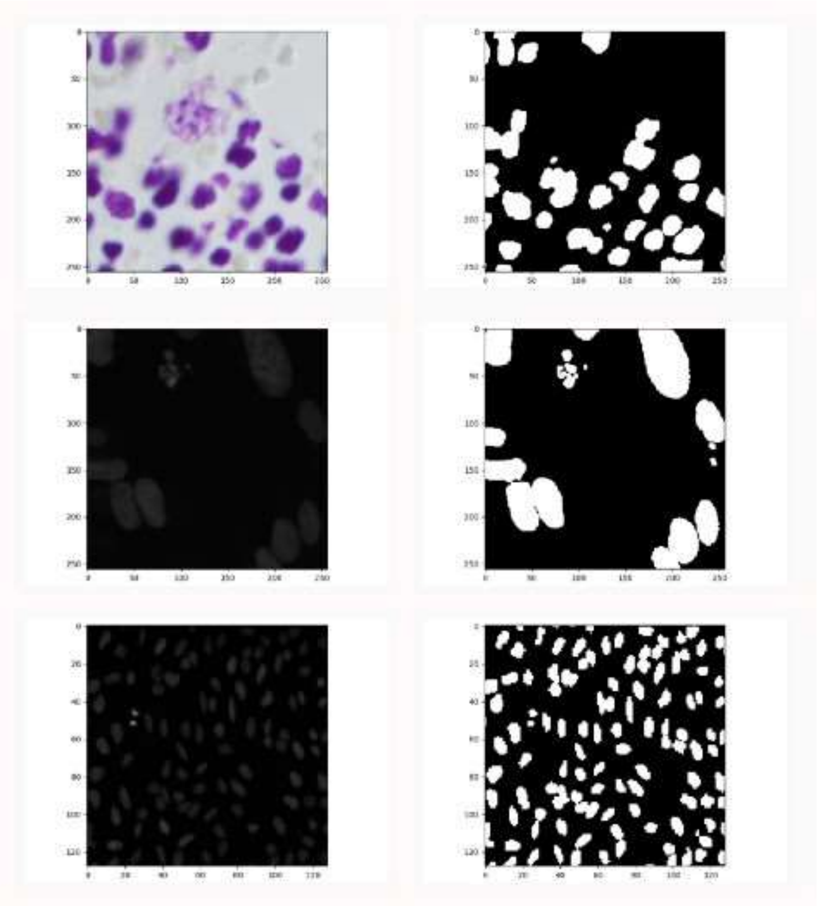
پیاده سازی

- **Intro to Autoencoders**
  - https://www.tensorflow.org/tutorials/generative/autoencoder
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/autoencoder.ipynb

Agenda

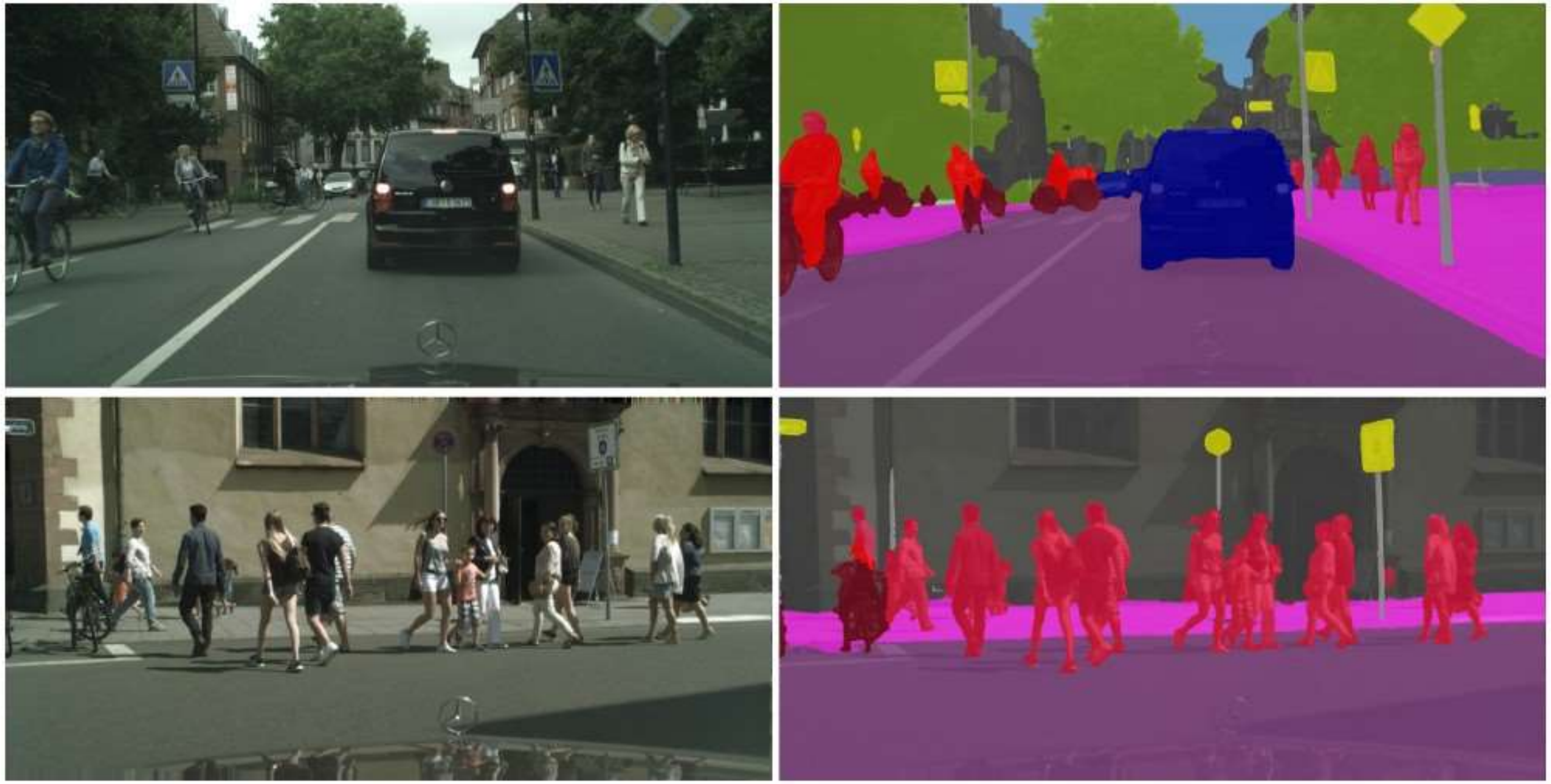
- Simple GAN
- Transposed convolution & DCGAN
- TF2 for researchers! (pre-requirement)
- Auto-Encoders
- **U-Net & Segmentation**
- GAN overriding Model.train_step
- Pix2pix
- Cycle-GAN

Segmentation



Human Segmentation

Semantic segmentation



Semantic segmentation of Cityscapes, with input on left and output on right. (source)

128x128x3

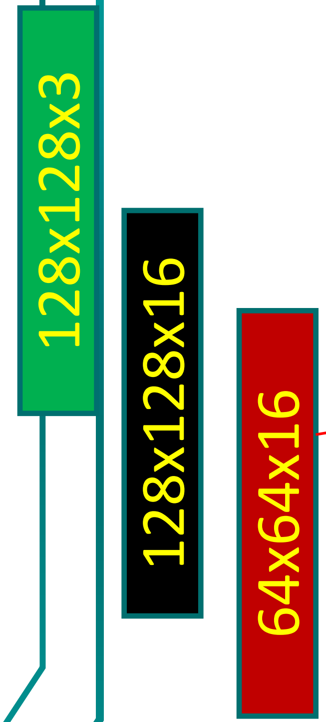
```
inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
s = Lambda(lambda x: x / 255) (inputs)
```

128x128x3

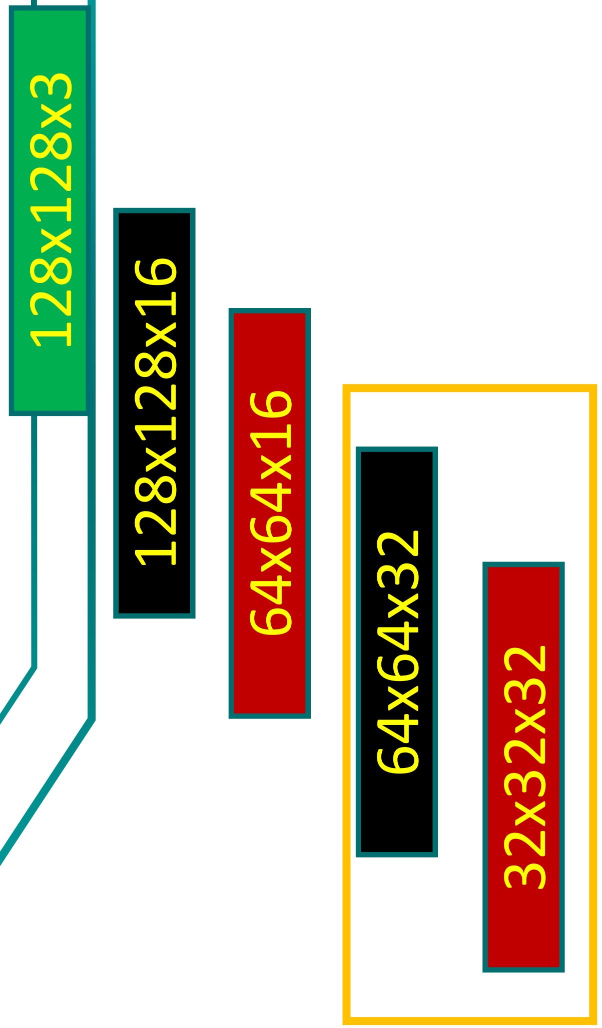
128x128x16

```
inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
s = Lambda(lambda x: x / 255) (inputs)
```

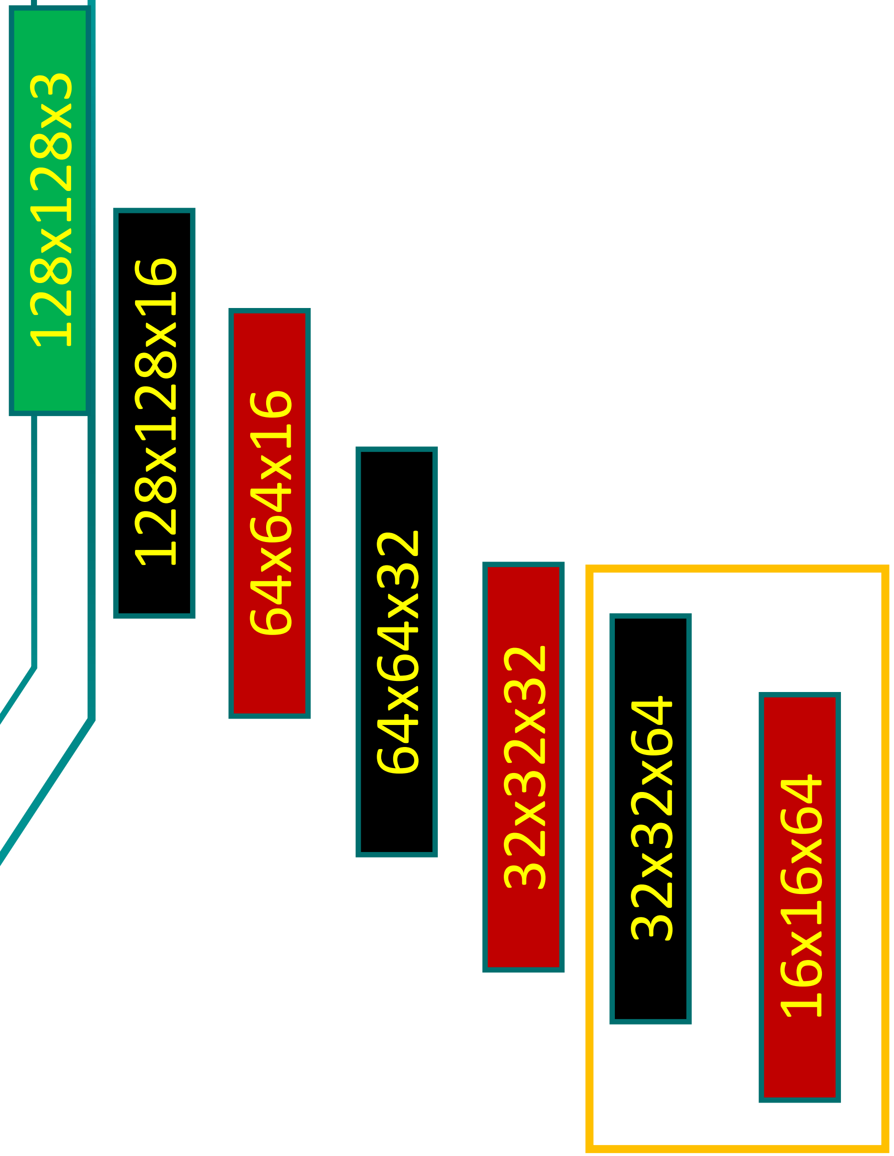
```
c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (c1)
```



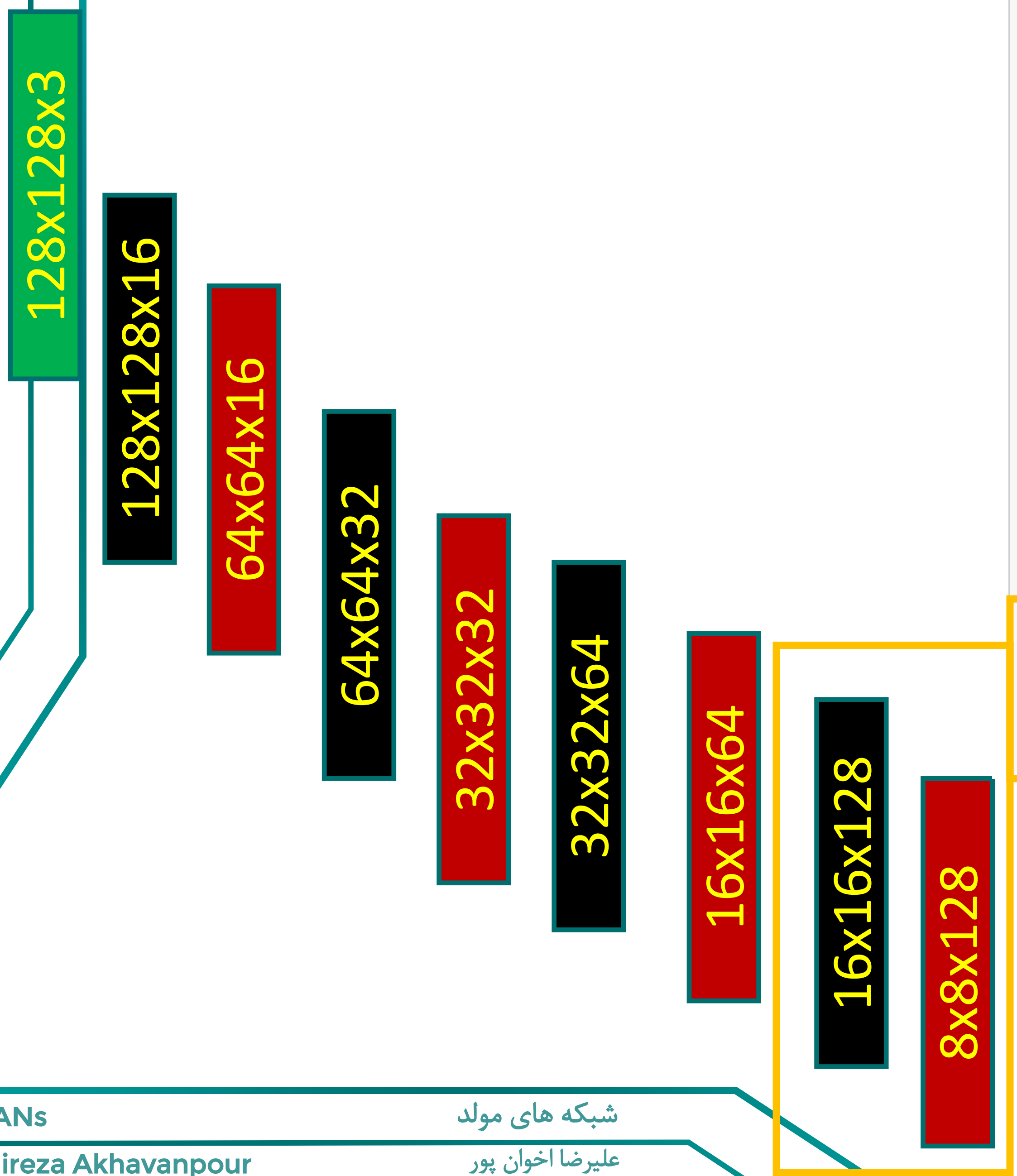

```
inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
s = Lambda(lambda x: x / 255) (inputs)

c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)
```



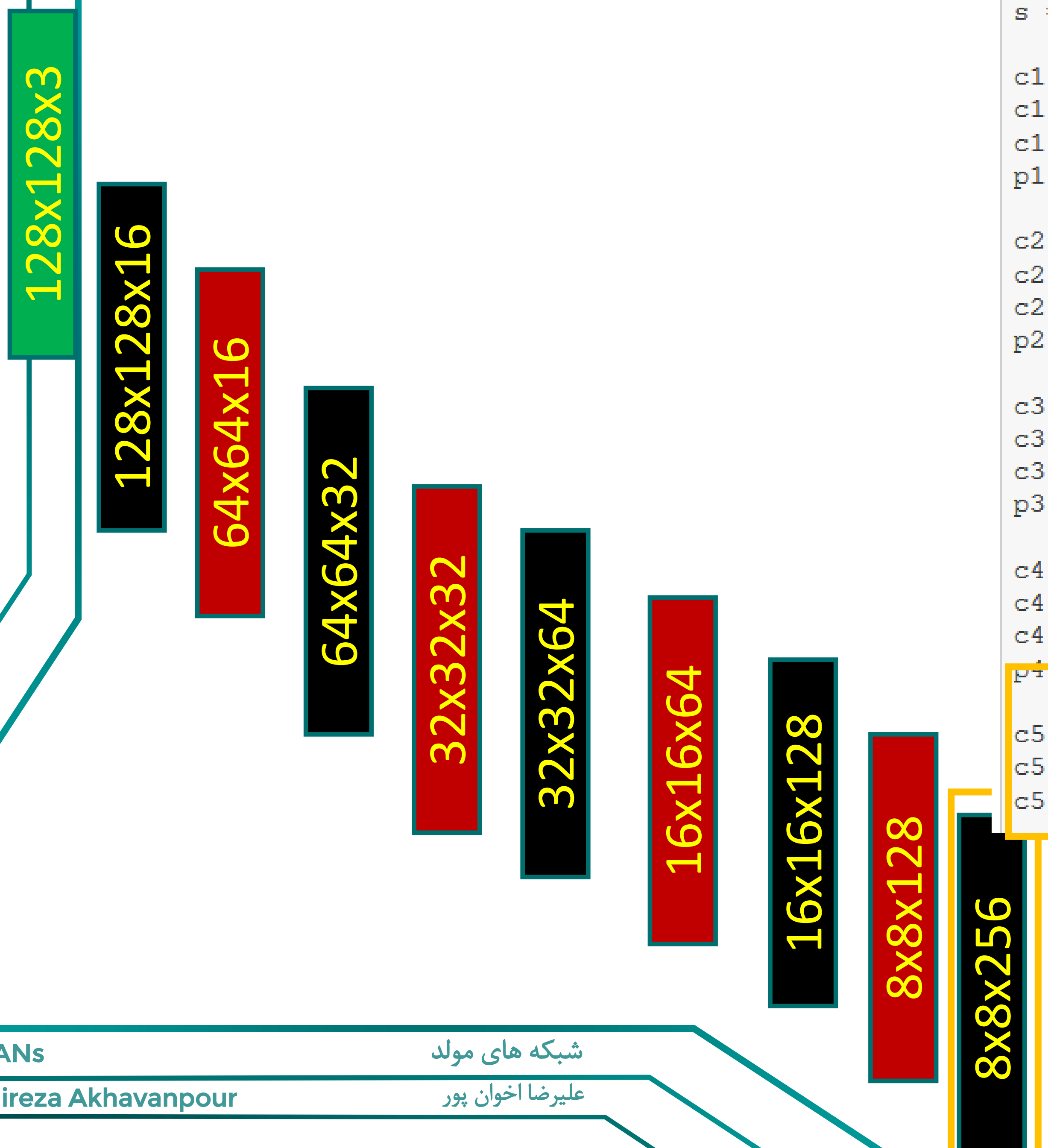

```
inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
s = Lambda(lambda x: x / 255) (inputs)

c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)

c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (p1)
c2 = Dropout(0.1) (c2)
c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (c2)
p2 = MaxPooling2D((2, 2)) (c2)
```



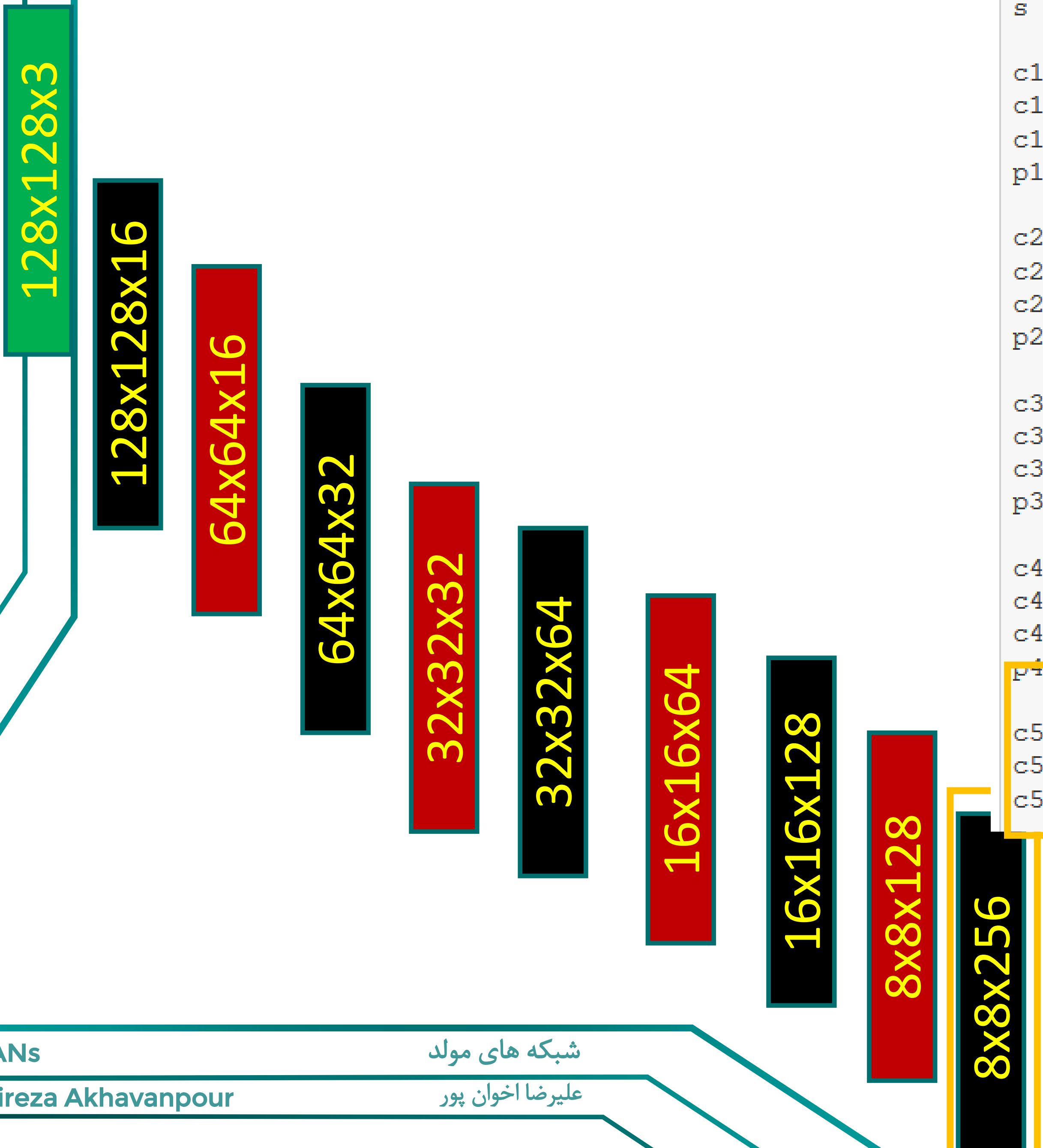

```
inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
s = Lambda(lambda x: x / 255) (inputs)

c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)

c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (p1)
c2 = Dropout(0.1) (c2)
c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (c2)
p2 = MaxPooling2D((2, 2)) (c2)

c3 = Conv2D(64, (3, 3), activation='relu', padding='same') (p2)
c3 = Dropout(0.2) (c3)
c3 = Conv2D(64, (3, 3), activation='relu', padding='same') (c3)
p3 = MaxPooling2D((2, 2)) (c3)
```




```
inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
s = Lambda(lambda x: x / 255) (inputs)

c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)

c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (p1)
c2 = Dropout(0.1) (c2)
c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (c2)
p2 = MaxPooling2D((2, 2)) (c2)

c3 = Conv2D(64, (3, 3), activation='relu', padding='same') (p2)
c3 = Dropout(0.2) (c3)
c3 = Conv2D(64, (3, 3), activation='relu', padding='same') (c3)
p3 = MaxPooling2D((2, 2)) (c3)

c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)
```

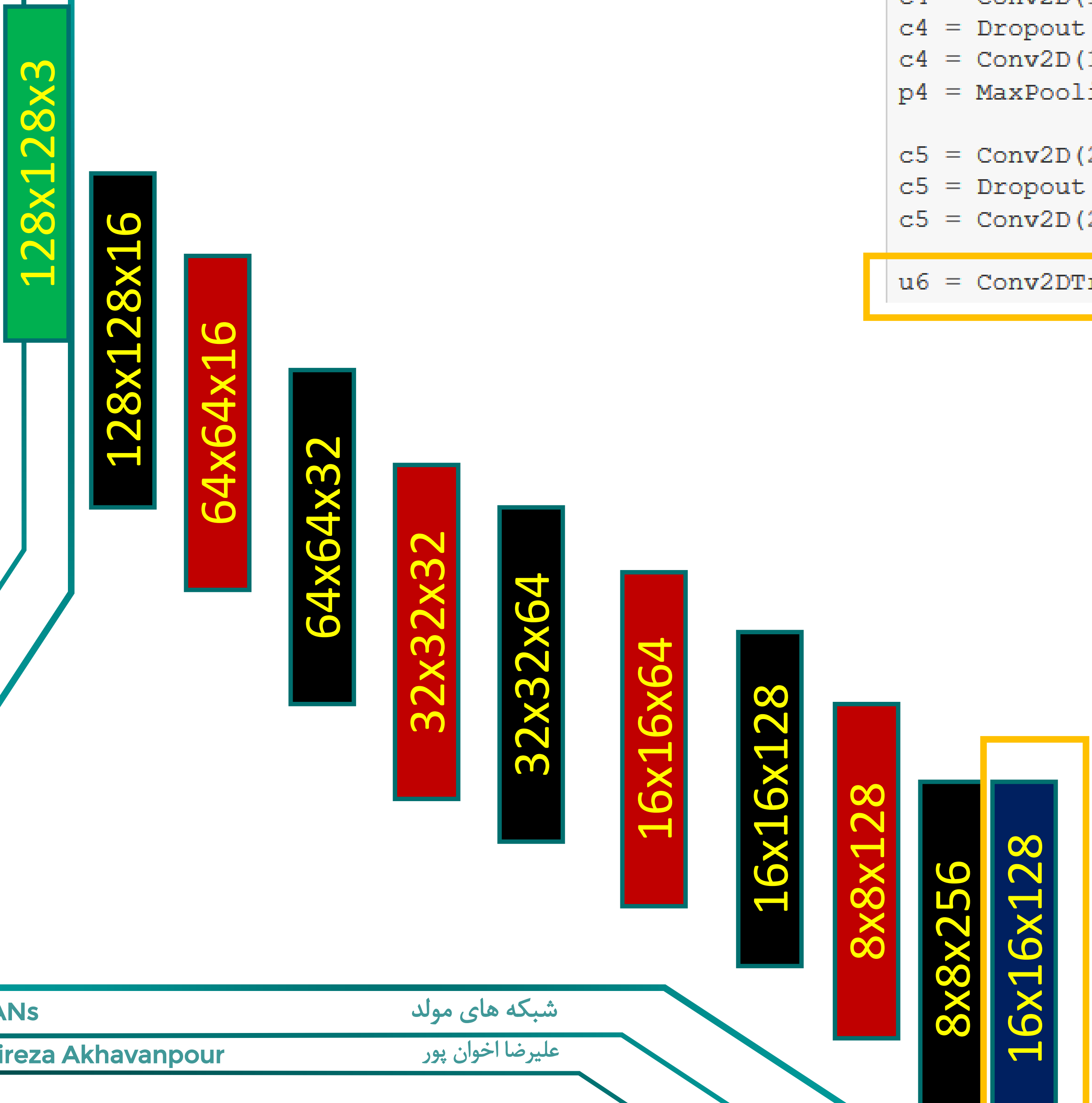



```python
inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
s = Lambda(lambda x: x / 255) (inputs)

c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)

c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (p1)
c2 = Dropout(0.1) (c2)
c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (c2)
p2 = MaxPooling2D((2, 2)) (c2)

c3 = Conv2D(64, (3, 3), activation='relu', padding='same') (p2)
c3 = Dropout(0.2) (c3)
c3 = Conv2D(64, (3, 3), activation='relu', padding='same') (c3)
p3 = MaxPooling2D((2, 2)) (c3)

c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (p4)
c5 = Dropout(0.3) (c5)
c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (c5)
```

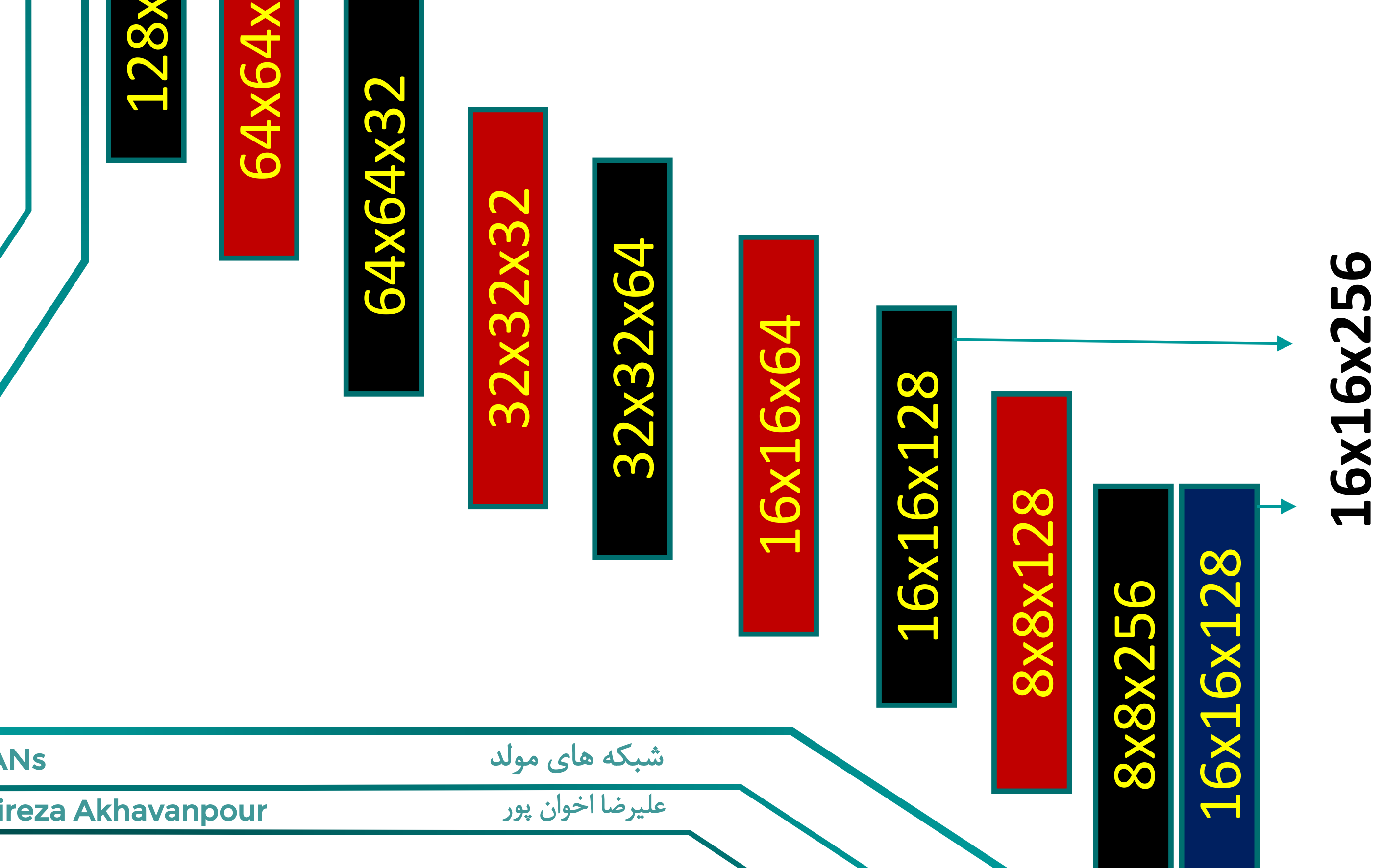



```
inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
s = Lambda(lambda x: x / 255) (inputs)

c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='relu', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)

c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (p1)
c2 = Dropout(0.1) (c2)
c2 = Conv2D(32, (3, 3), activation='relu', padding='same') (c2)
p2 = MaxPooling2D((2, 2)) (c2)

c3 = Conv2D(64, (3, 3), activation='relu', padding='same') (p2)
c3 = Dropout(0.2) (c3)
c3 = Conv2D(64, (3, 3), activation='relu', padding='same') (c3)
p3 = MaxPooling2D((2, 2)) (c3)

c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (p4)
c5 = Dropout(0.3) (c5)
c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (c5)
```

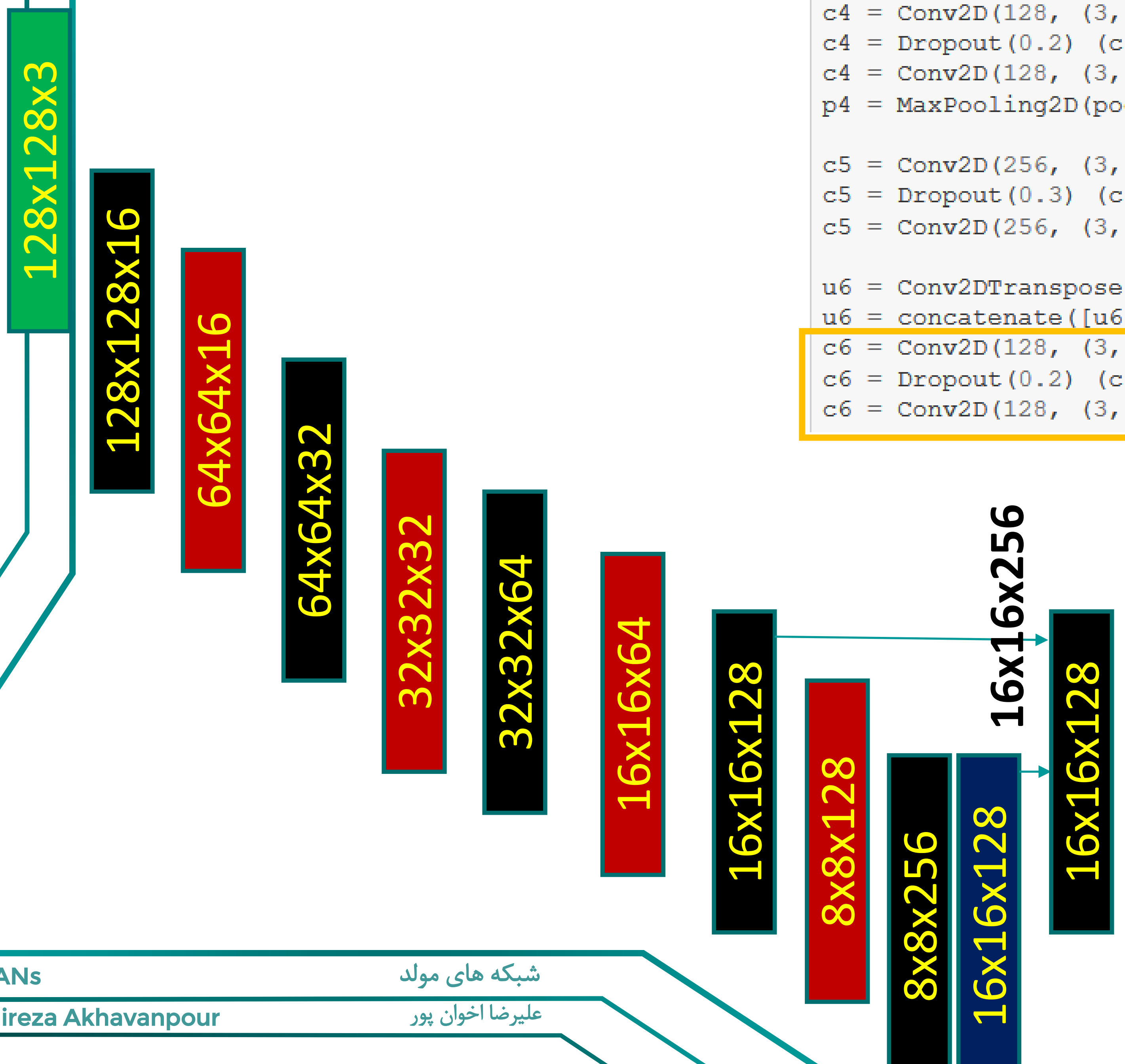
```
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (p4)
c5 = Dropout(0.3) (c5)
c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (c5)

u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (c5)
```

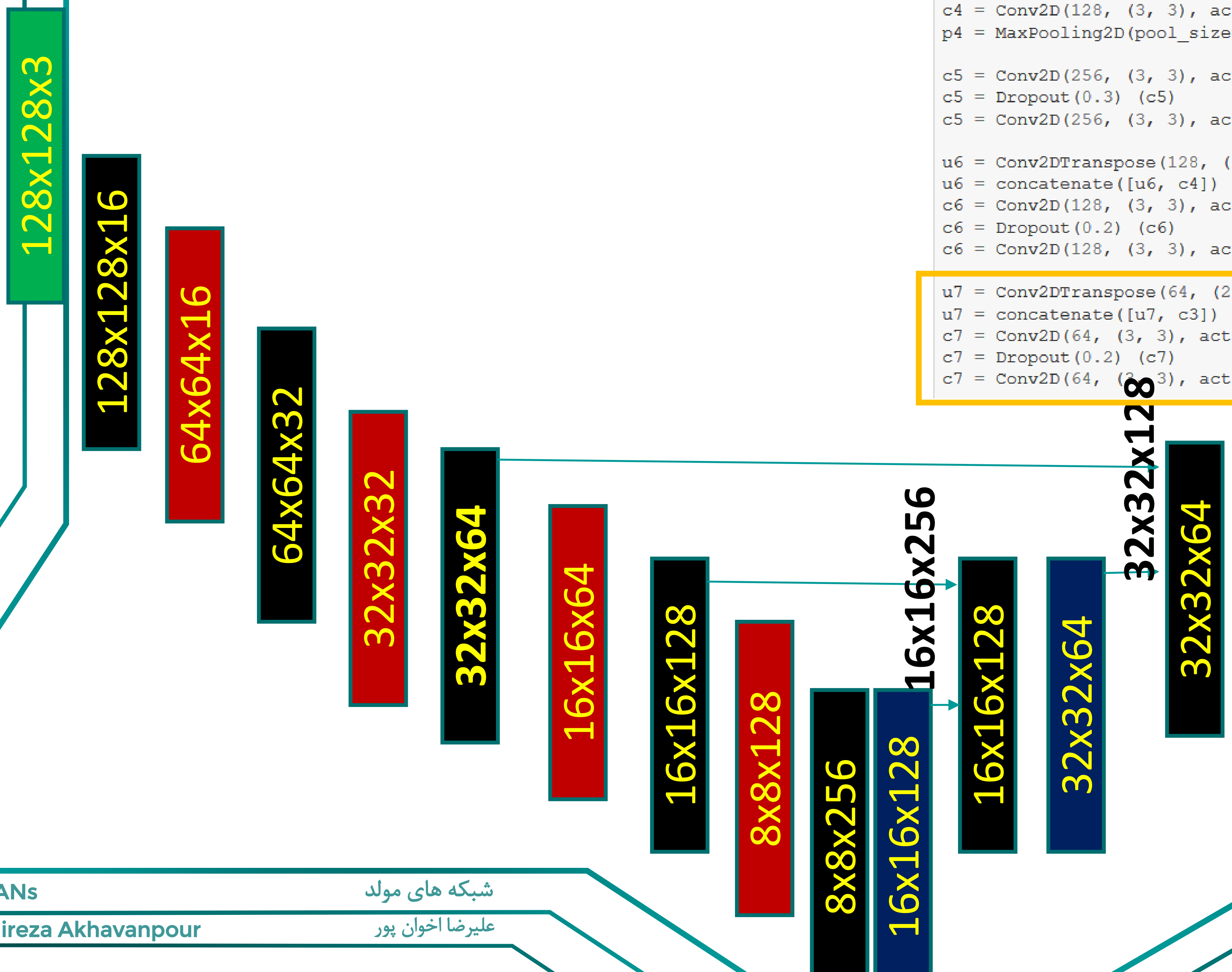


```
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (p4)
c5 = Dropout(0.3) (c5)
c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (c5)

u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (c5)
u6 = concatenate([u6, c4])
```

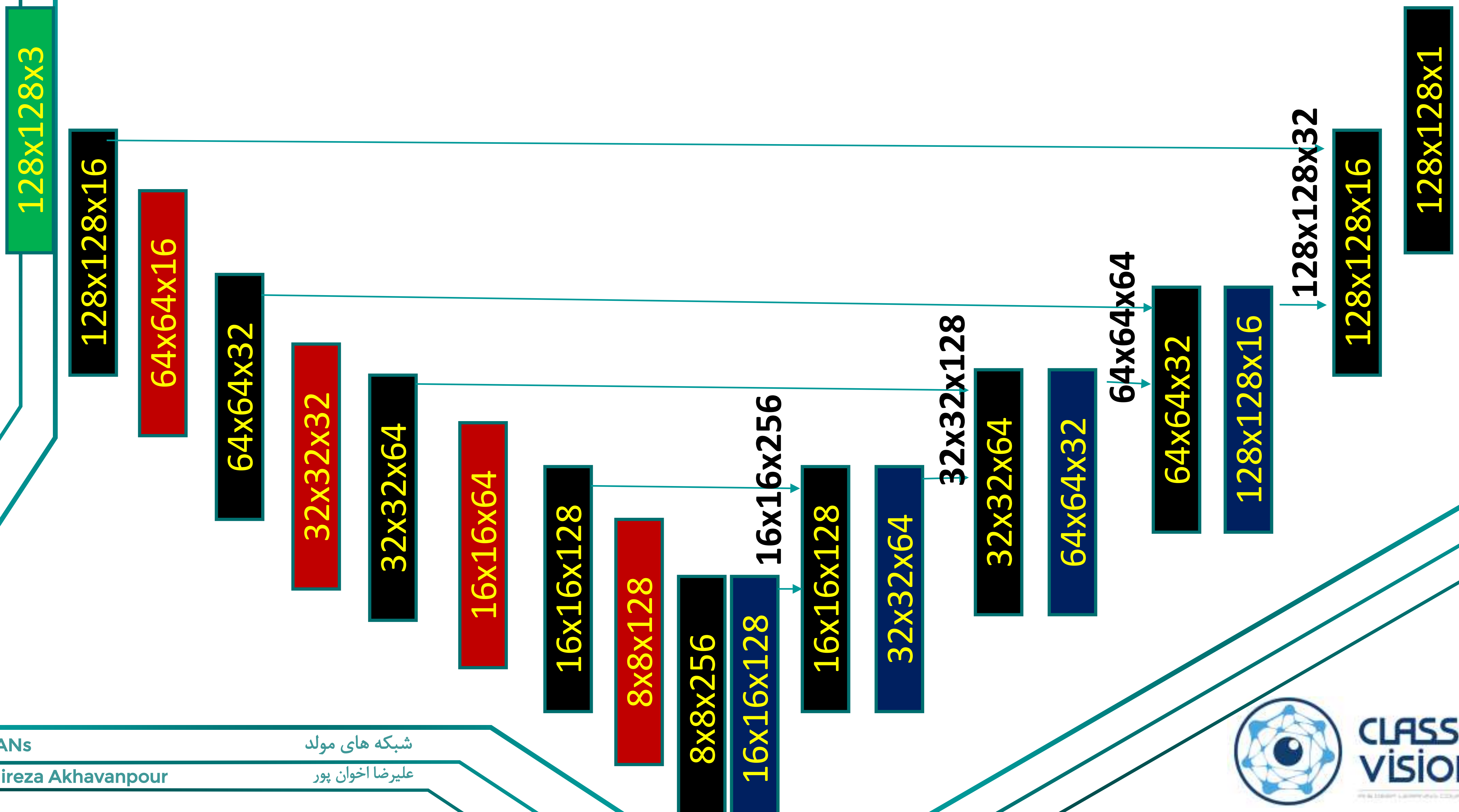


```python
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (p4)
c5 = Dropout(0.3) (c5)
c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (c5)

u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (c5)
u6 = concatenate([u6, c4])
c6 = Conv2D(128, (3, 3), activation='relu', padding='same') (u6)
c6 = Dropout(0.2) (c6)
c6 = Conv2D(128, (3, 3), activation='relu', padding='same') (c6)
```






```
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='relu', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (p4)
c5 = Dropout(0.3) (c5)
c5 = Conv2D(256, (3, 3), activation='relu', padding='same') (c5)

u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (c5)
u6 = concatenate([u6, c4])
c6 = Conv2D(128, (3, 3), activation='relu', padding='same') (u6)
c6 = Dropout(0.2) (c6)
c6 = Conv2D(128, (3, 3), activation='relu', padding='same') (c6)

u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same') (c6)
u7 = concatenate([u7, c3])
c7 = Conv2D(64, (3, 3), activation='relu', padding='same') (u7)
c7 = Dropout(0.2) (c7)
c7 = Conv2D(64, (3, 3), activation='relu', padding='same') (c7)
```


128x128x3
128x128x16
64x64x16
64x64x32
32x32x32
32x32x64
16x16x64
16x16x128
8x8x128
8x8x256
16x16x128
16x16x256
16x16x128
32x32x64
32x32x128
32x32x64
64x64x32
64x64x64
64x64x32
128x128x16
128x128x32
128x128x16
128x128x1

پیاده سازی

- https://nbviewer.jupyter.org/github/Alireza-Akhavan/class.vision/blob/master/U-Net.ipynb

Agenda

- Simple GAN
- Transposed convolution & DCGAN
- TF2 for researchers! (pre-requirement)
- Auto-Encoders
- U-Net & Segmentation
- **GAN overriding Model.train_step**
- Pix2pix
- Cycle-GAN

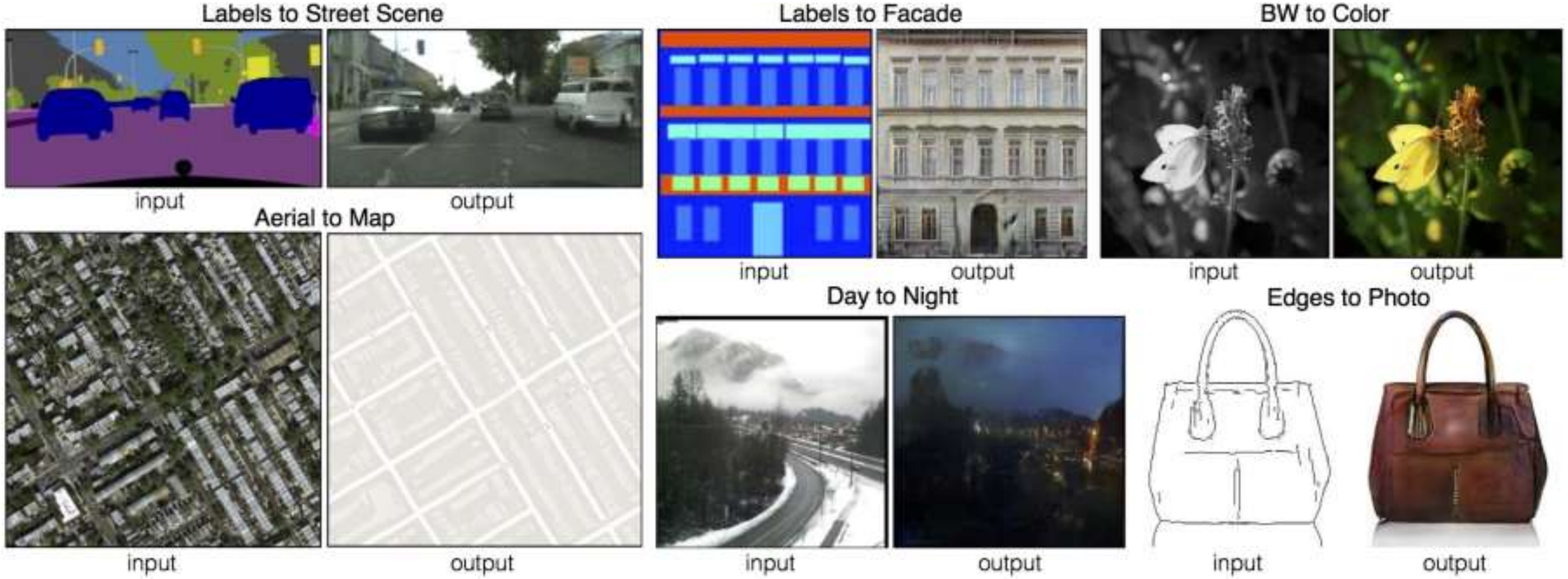
Official documentation

- https://keras.io/examples/generative/dcgan_overriding_train_step/

Agenda

- Simple GAN
- Transposed convolution & DCGAN
- TF2 for researchers! (pre-requirement)
- Auto-Encoders
- U-Net & Segmentation
- GAN overriding Model.train_step
- **Pix2pix**
- Cycle-GAN

pix2pix by Isola et al.



Try Online: https://affinelayer.com/pixsrv/

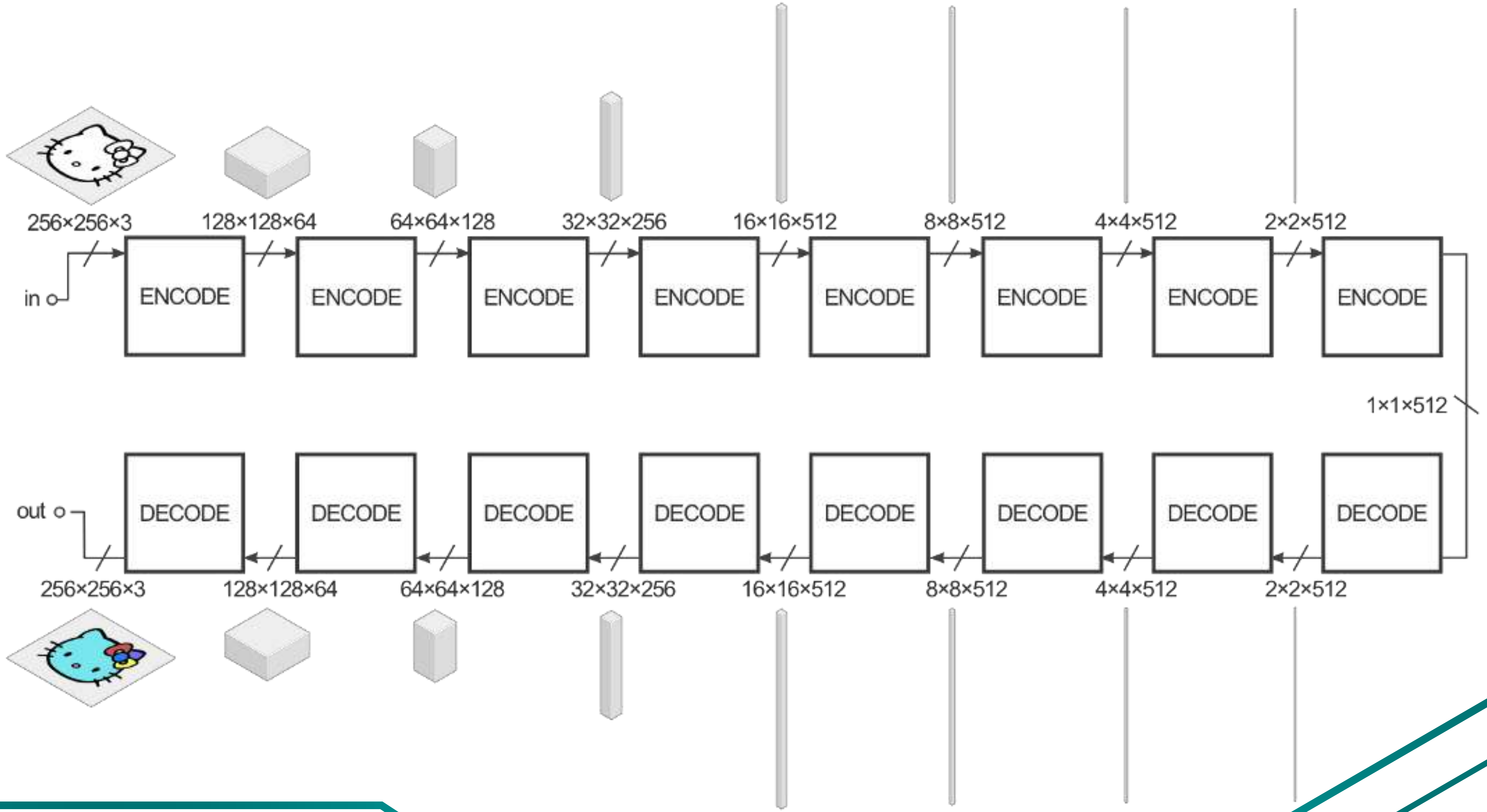
Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks."(2017).

Pix2Pix

- pix2pix uses a **conditional** generative adversarial network (**cGAN**) to learn a mapping from an input image to an output image.

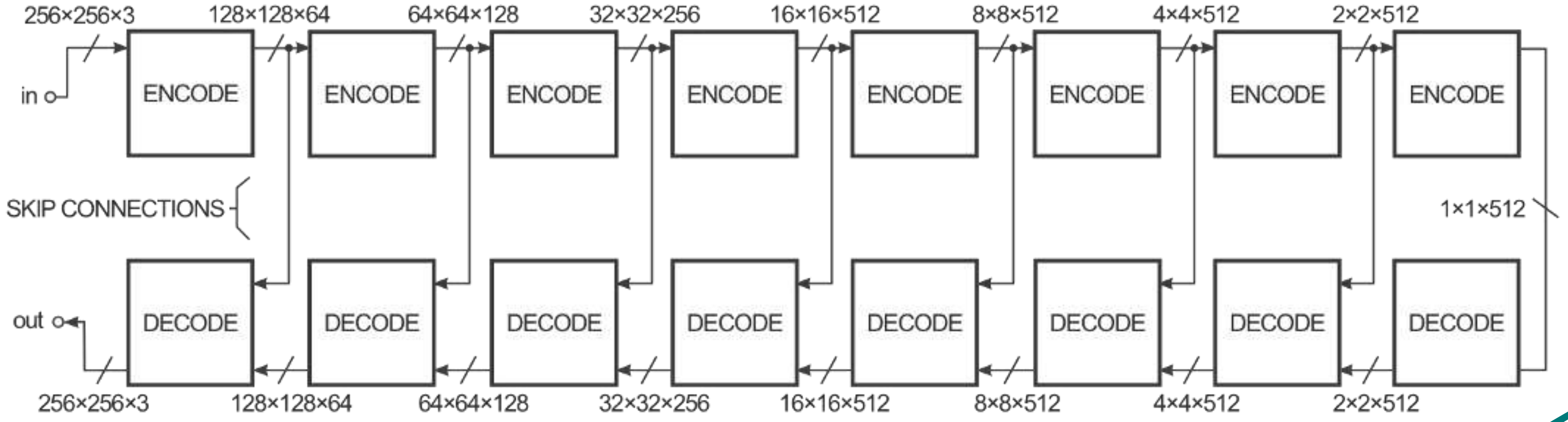
Generator

- The structure of the **generator** is called an “**encoder-decoder**” and in pix2pix the encoder-decoder looks more or less like this:



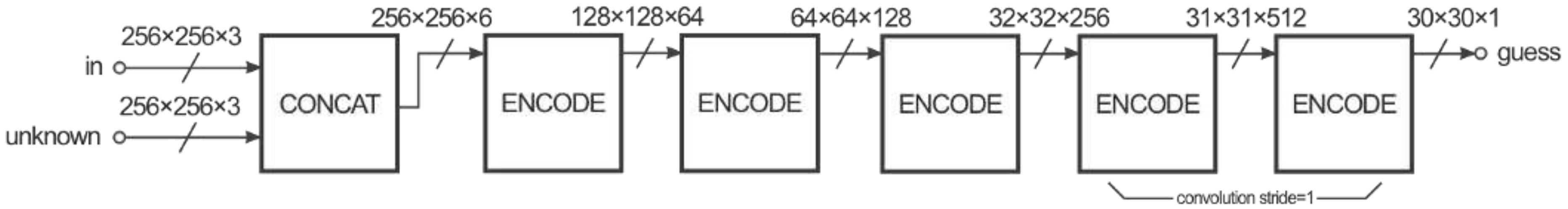
Generator: **U-Net**

- The authors used a “U-Net” instead of an encoder-decoder.
- skip connections



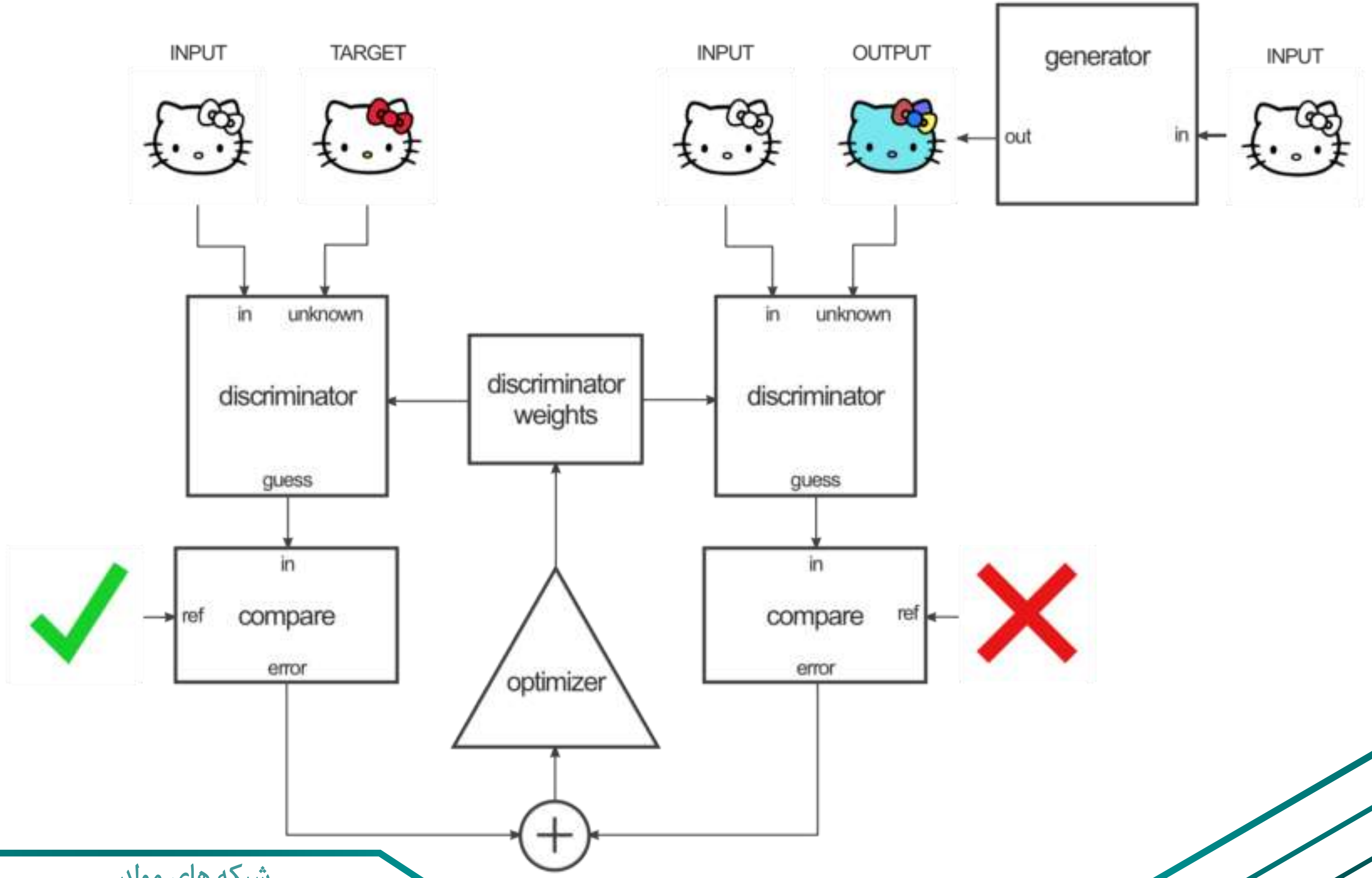
Discriminator - **PatchGAN**

The Discriminator has the job of **taking two images**, an **input** image and an **unknown** image (which will be either a **target or output image from the generator**), and deciding if the second image was produced by the generator or not.



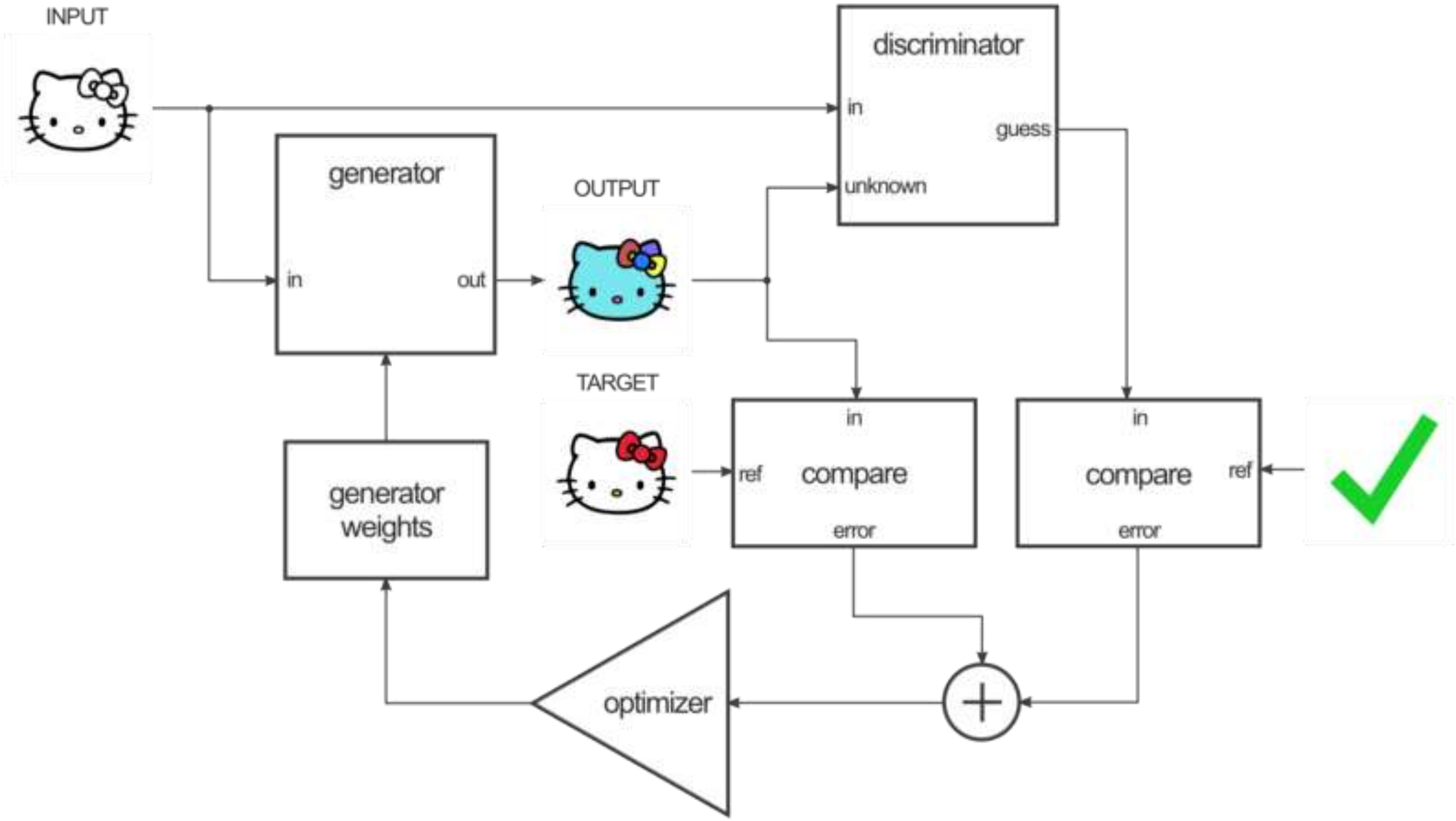
The **output** is a **30x30** image where each pixel value (0 to 1) represents how believable the corresponding section of the unknown image is. In the pix2pix implementation, each pixel from this 30x30 image corresponds to **the believability of a 70x70 patch** of the input image (the **patches overlap** a lot since the input images are 256x256)

Training - D



INPUT
TARGET
INPUT
OUTPUT
generator
out
in
INPUT
in
unknown
discriminator
guess
discriminator
weights
in
unknown
discriminator
guess
in
ref
compare
error
optimizer
in
compare
ref
error

Training - G



INPUT
discriminator
in
guess
unknown
generator
in
out
OUTPUT
TARGET
in
ref
compare
error
in
compare
ref
error
generator
weights
optimizer

Tutorial

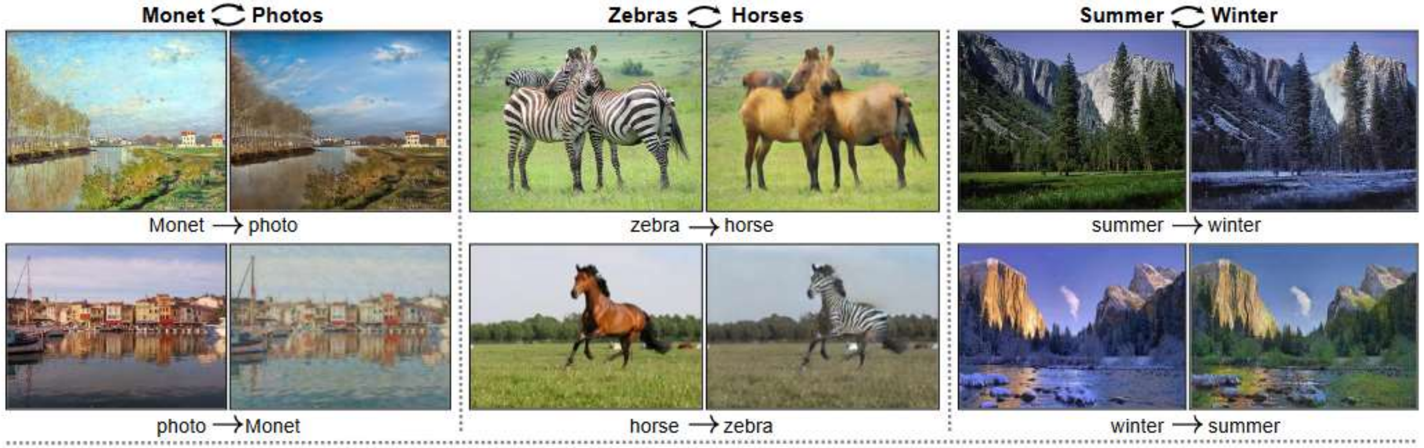
- https://www.tensorflow.org/tutorials/generative/pix2pix
- https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/pix2pix.ipynb

Agenda

- Simple GAN
- Transposed convolution & DCGAN
- TF2 for researchers! (pre-requirement)
- Auto-Encoders
- U-Net & Segmentation
- GAN overriding Model.train_step
- Pix2pix
- **Cycle-GAN**

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

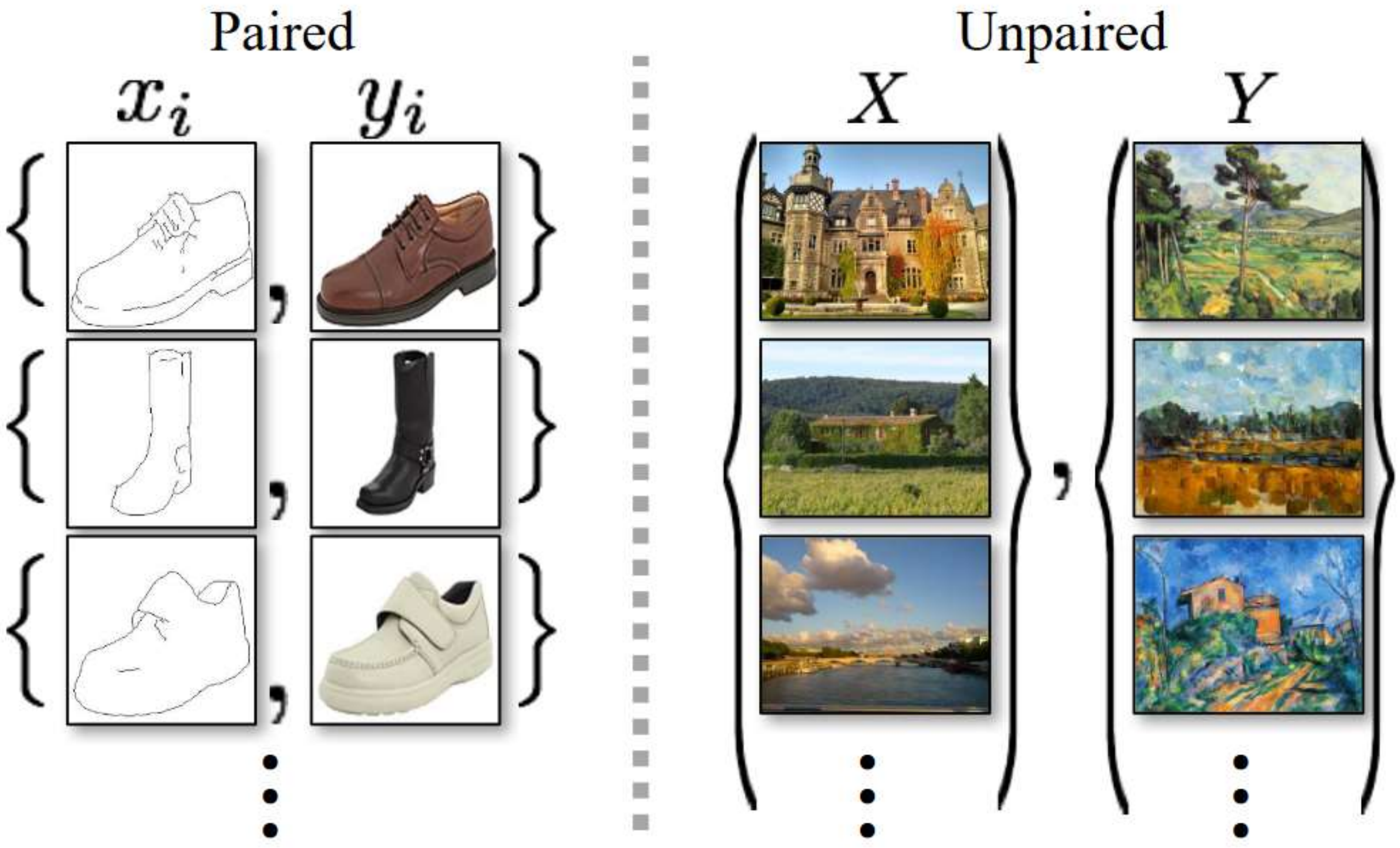
Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley



https://arxiv.org/pdf/1703.10593.pdf

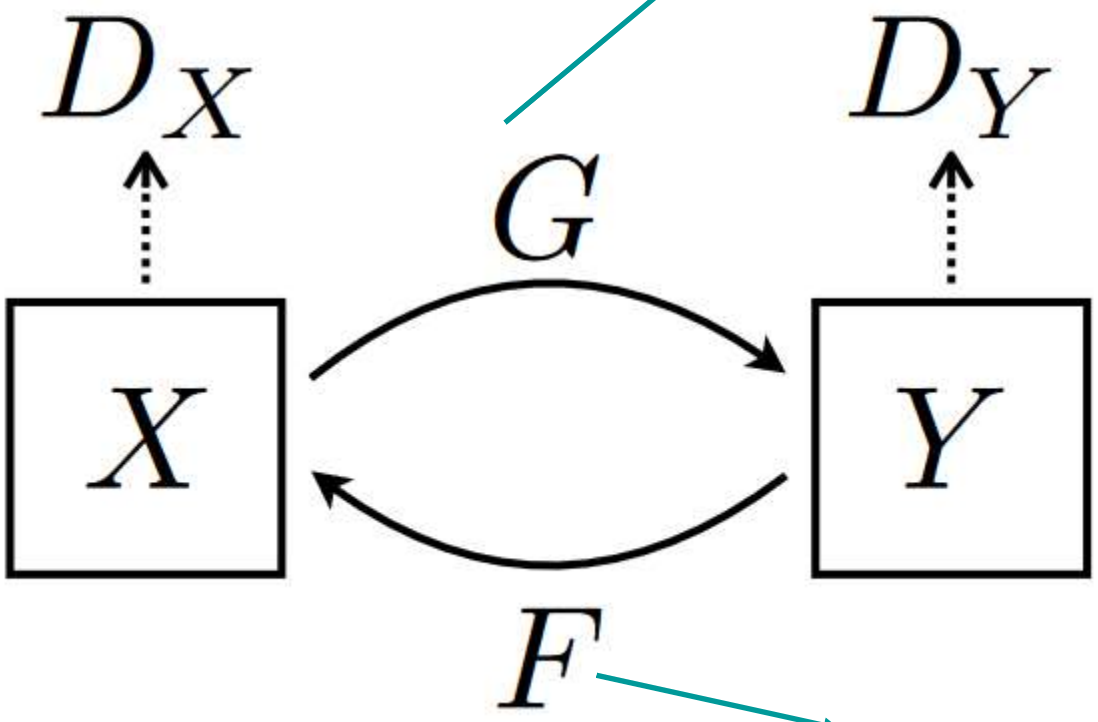
https://t.me/cvision/349 https://youtu.be/Fea4kZq0oFQ

The challenge



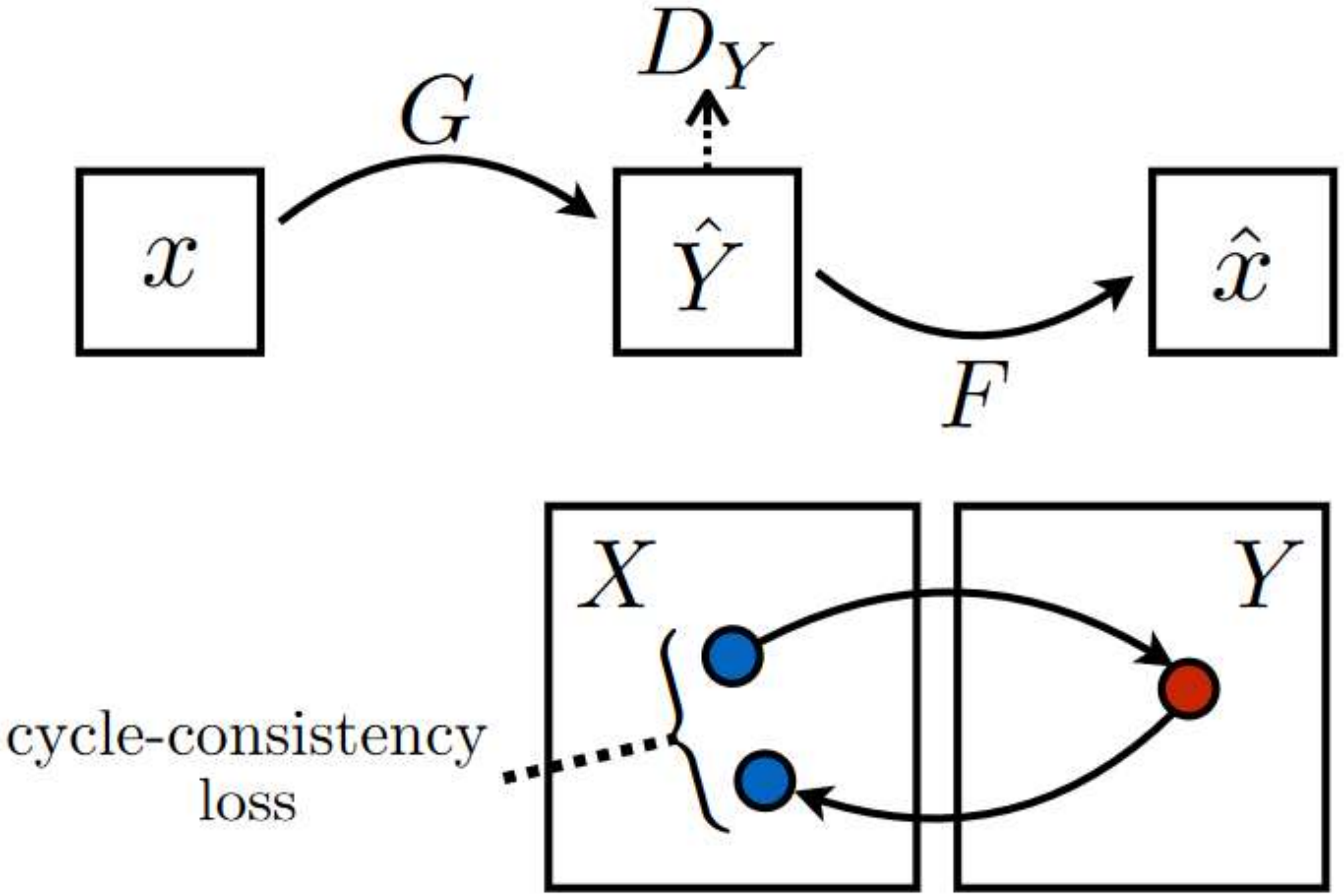
Full translation cycle

G: horse to zebra

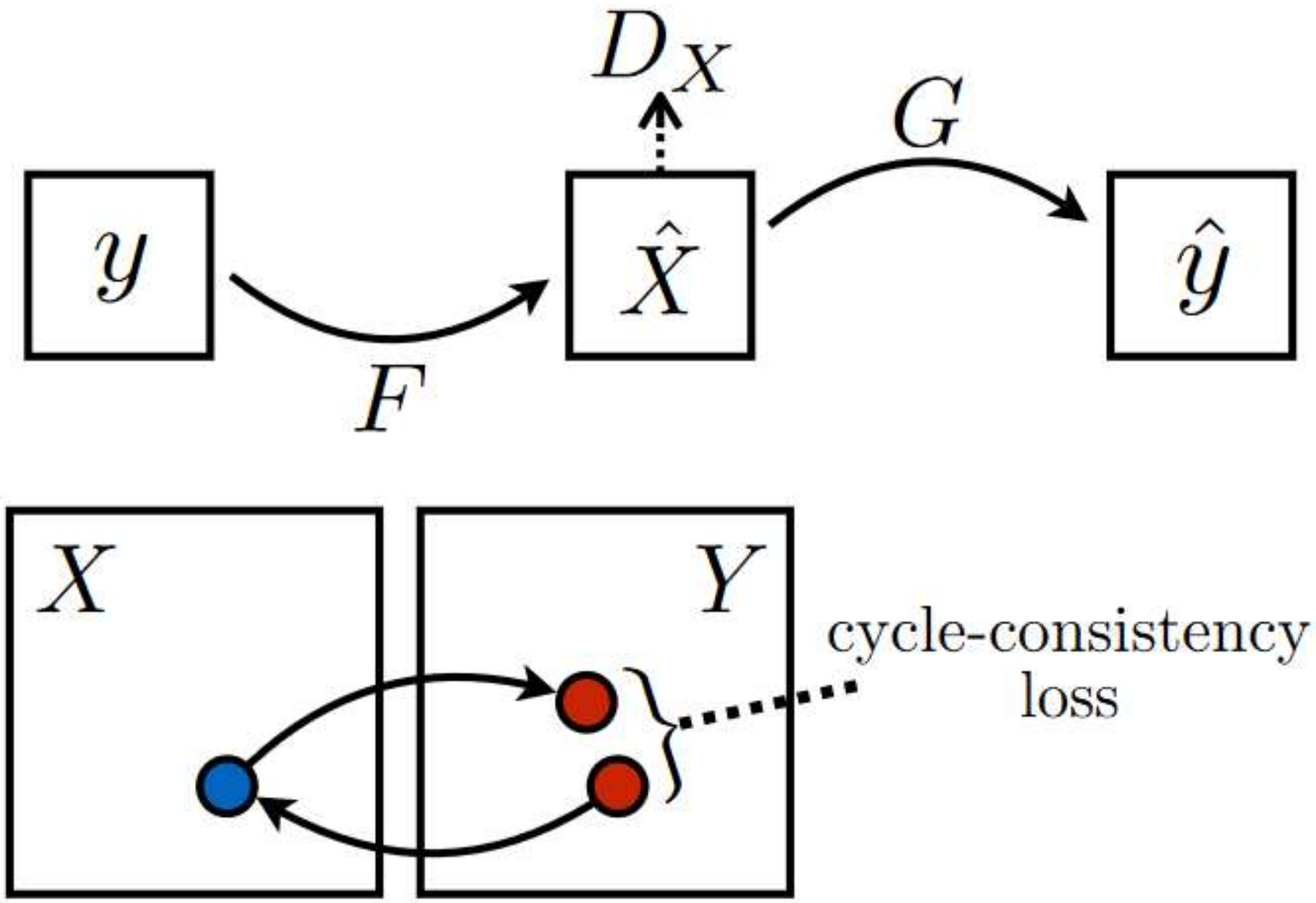


F: zebra to horse

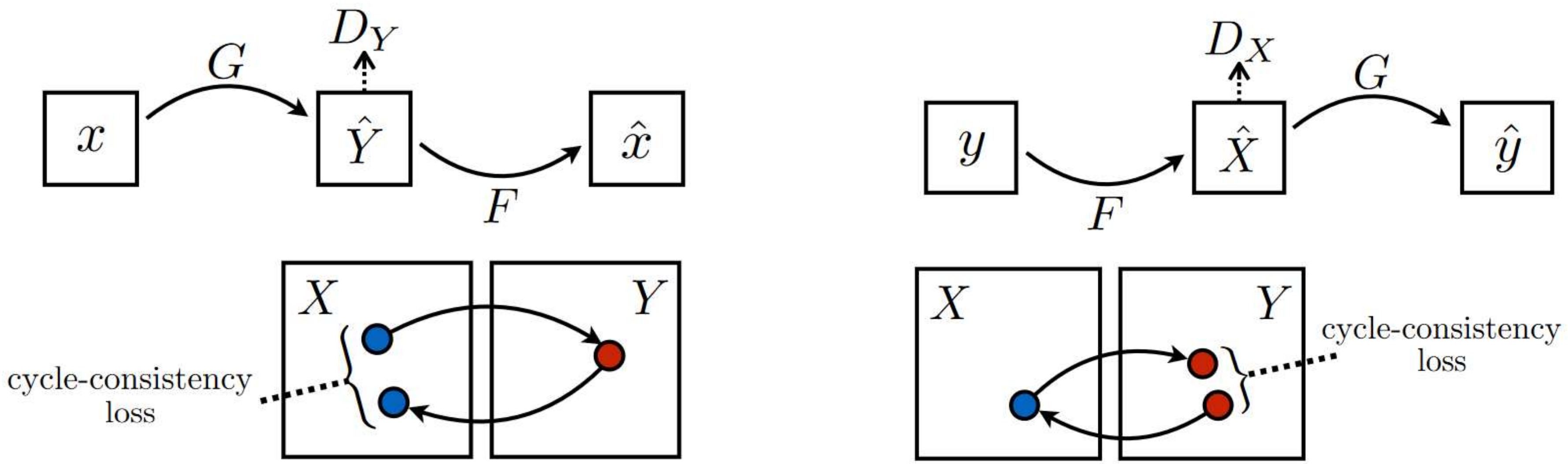
Cycle consistency losses



Cycle consistency losses



Cycle Loss Formulation



$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Full objective

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda\mathcal{L}_{\text{cyc}}(G, F),$$

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$


Input
Monet
Van Gogh
Cezanne
Ukiyo-e

winter Yosemite → summer Yosemite

summer Yosemite → winter Yosemite

apple → orange

orange → apple

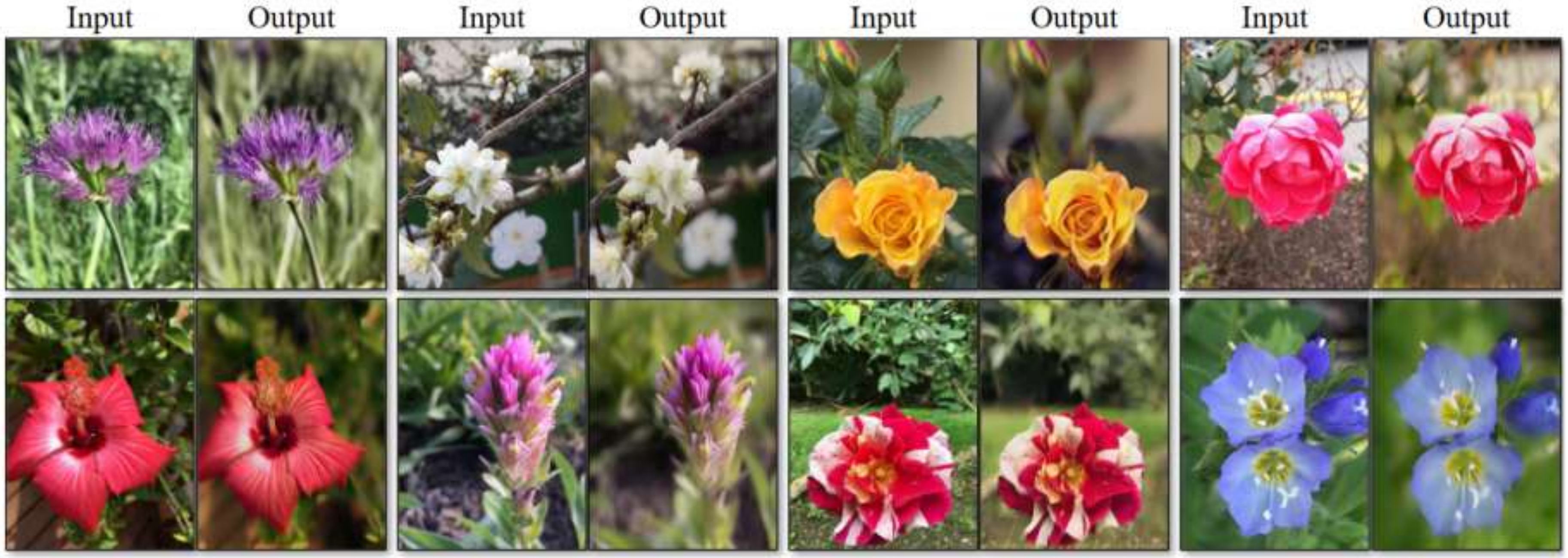
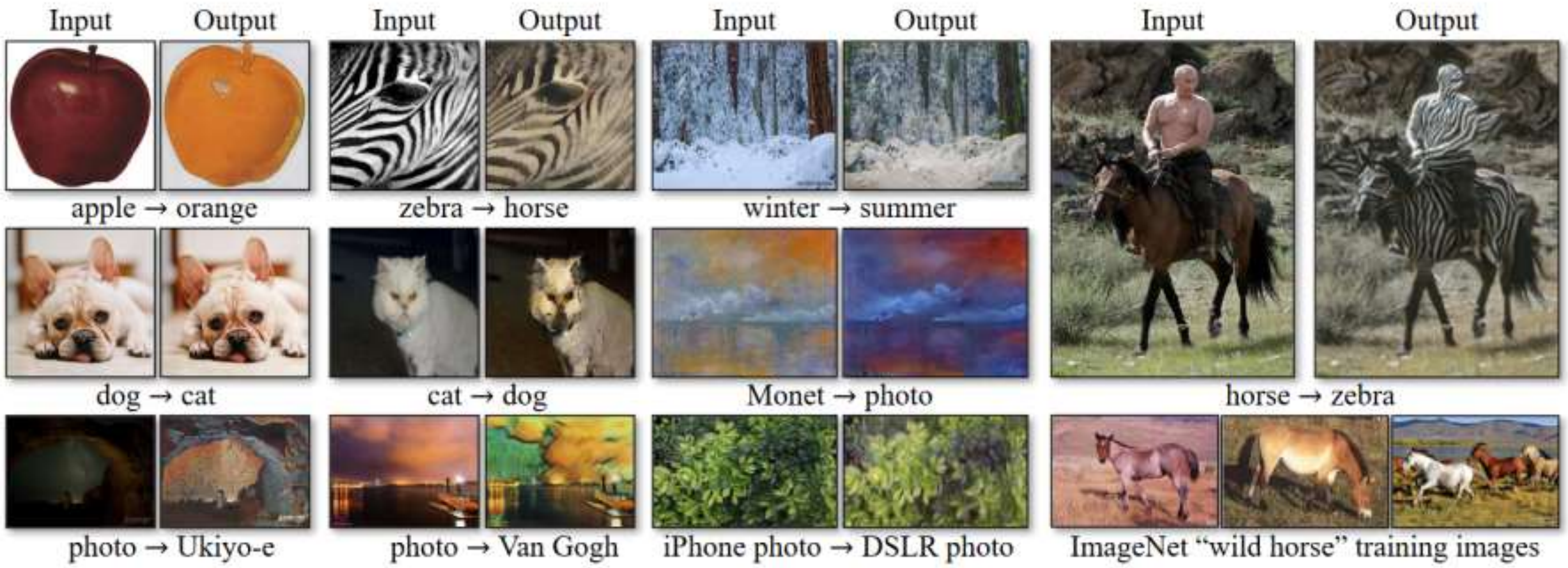


Figure 14: Photo enhancement: mapping from a set of smartphone snaps to professional DSLR photographs, the system often learns to produce shallow focus. Here we show some of the most successful results in our test set – average performance is considerably worse. Please see our website for more comprehensive and random examples.

Failure


Input
Output
apple → orange
zebra → horse
winter → summer
dog → cat
cat → dog
Monet → photo
horse → zebra
photo → Ukiyo-e
photo → Van Gogh
iPhone photo → DSLR photo
ImageNet "wild horse" training images

منابع

https://livebook.manning.com/#!/book/gans-in-action/chapter-4/v-4/54
https://filebox.ece.vt.edu/~jbhuang/teaching/ece6554/sp17/lectures/GAN-topic.pptx
https://www.slideshare.net/datamics/slideshare-gan
https://mc.ai/conditional%E2%80%8A-%E2%80%8Adcgan-in-tensorflow/
https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8
https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html
https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798
http://openaccess.thecvf.com/content_cvpr_2017/papers/Isola_Image-To-Image_Translation_With_CVPR_2017_paper.pdf
https://www.microsoft.com/developerblog/2017/06/12/learning-image-image-translation-cyclegans/
https://arxiv.org/pdf/1703.10593.pdf