ERHVERVS AKADEMI AARHUS

DC GAN & Google Colab.

Sila. Pba, ML. Nov. 2021.

From a GAN website:

*DCGAN, or **Deep Convolutional GAN**, is a generative adversarial network architecture. It uses a couple of guidelines.*

- Replacing any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Removing fully connected hidden layers for deeper architectures.
- Using LeakyReLU activation in the discriminator for all layer.

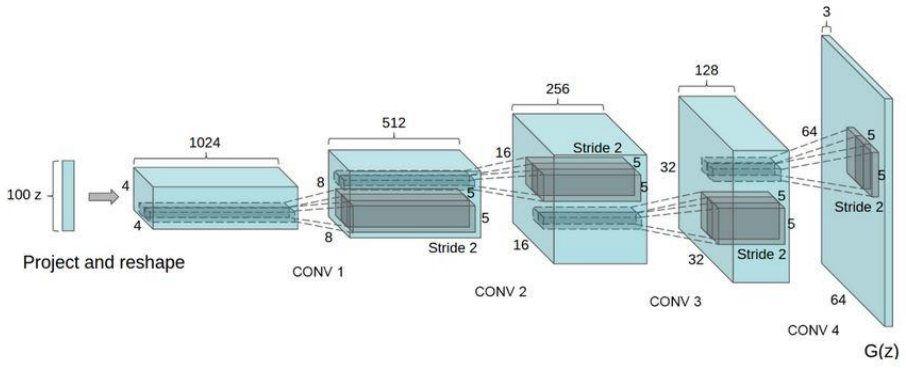


Fig: Convolution transpose layers in a DC GAN.

But, well, most GANs contain convolutional layers, so the DC is implied when we usually talk about GANs. Batch normalization layers is also quite normal (in the discriminator at least).

Here we will use the Celeba face dataset as basis for creating new faces. The size of this dataset is 1.3GB, and is available here:
https://drive.google.com/uc?id=1O7m1010EJjLE5QxLZiM9Fpjs7Oj6e684

A description (and download) of the dataset can be found here:

https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

One of the faces in the dataset:

```
for x in dataset:
    plt.axis("off")
    plt.imshow((x.numpy() * 255).astype("int32")[0])
    break
```

Discriminator:

```
discriminator = keras.Sequential(
    [
        keras.Input(shape=(64, 64, 3)),
        layers.Conv2D(64, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Flatten(),
        layers.Dropout(0.2),
        layers.Dense(1, activation="sigmoid"),
    ],
    name="discriminator",
)
```

Generator:

```
latent_dim = 128

generator = keras.Sequential(
    [
        keras.Input(shape=(latent_dim,)),
        layers.Dense(8 * 8 * 128),
        layers.Reshape((8, 8, 128)),
        layers.Conv2DTranspose(128, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2DTranspose(256, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2DTranspose(512, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2D(3, kernel_size=5, padding="same", activation="sigmoid"),
    ],
    name="generator",
)
```

The full DC GAN by Francois Chollet is available here:

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

Press view in Colab, and you are good to go (i.e. run it).

Except, of course, you have to be patient.

One epoch of training will probably take something like an hour and 20 minutes on colab.

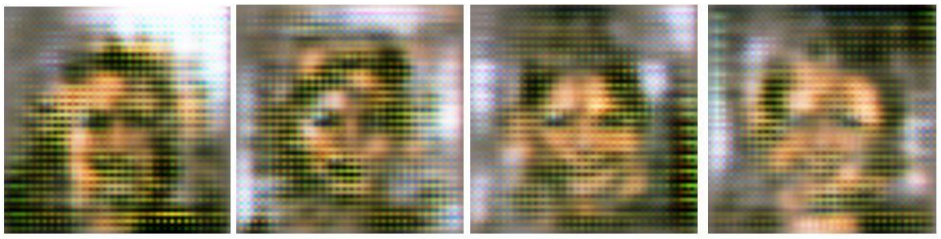
```
)

gan.fit(
    dataset, epochs=epochs, callbacks=[GANMonitor(num_img=10, latent_dim=latent_dim)]
)
```

```
1988/6332 [========>.....................] - ETA: 59:44 - d_loss: 0.5392 - g_loss: 1.5540
```

Here are some results after the first 1 epoch (faces generated by the GAN).

After 1 epoch.



After 1 epoch.

After 1 epoch.

// Run on Colab Nov 30th 2021 //

After 30 epoch (faces generated by the GAN, according to Chollet).

Some of the last generated images around epoch 30 (results keep improving after that):

According to Chollet:

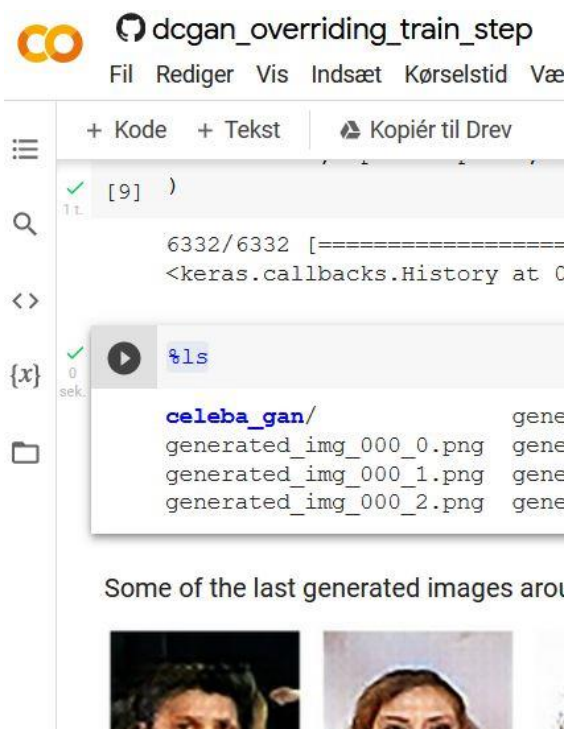
In practice, use ~100 epochs

To get good results ….

(unlike the results from one epoch…)

And now it is your turn ….

Your output can be seen by pressing



the files (folder) icon on the left in Colab. Press download to see generated output.

Make a run with at least 1 epoch … (or 100 epochs if you are up for it…).

And make sure to share your results… ☺