Generative Adversarial Networks

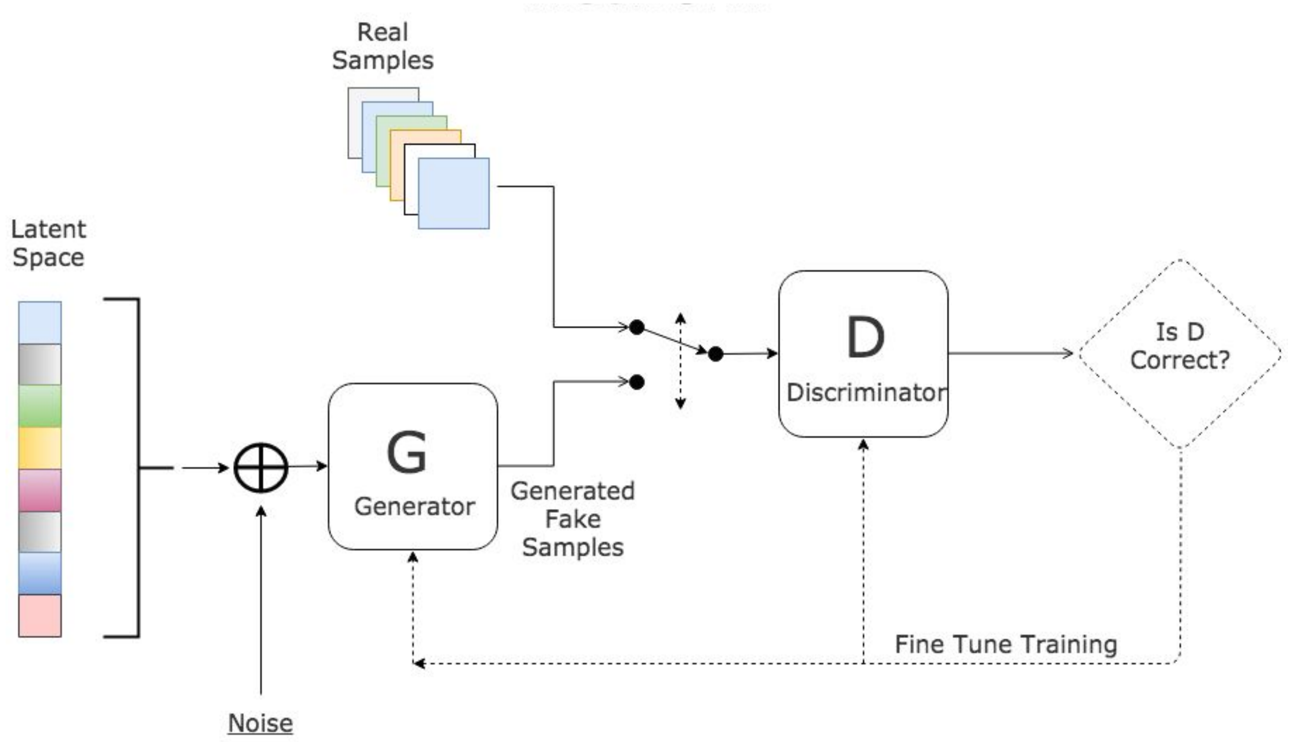
by Manas Bedmutha

PyData Gandhinagar

Intro

- ❖ Proposed by Prof. Ian Goodfellow in 2014.
- ❖ A Generative Adversarial Network (GAN) is used to create data that looks at least to superficially similar to original to human observers, having many realistic characteristics (though in tests people can tell real from generated in many cases).
- ❖ They produce photorealistic images and find application in data augmentation, reconstruction as well as feature mapping.
- ❖ Common applications include Medical Imaging, Superresolution and Domain Transfer

Visualizing the Structure

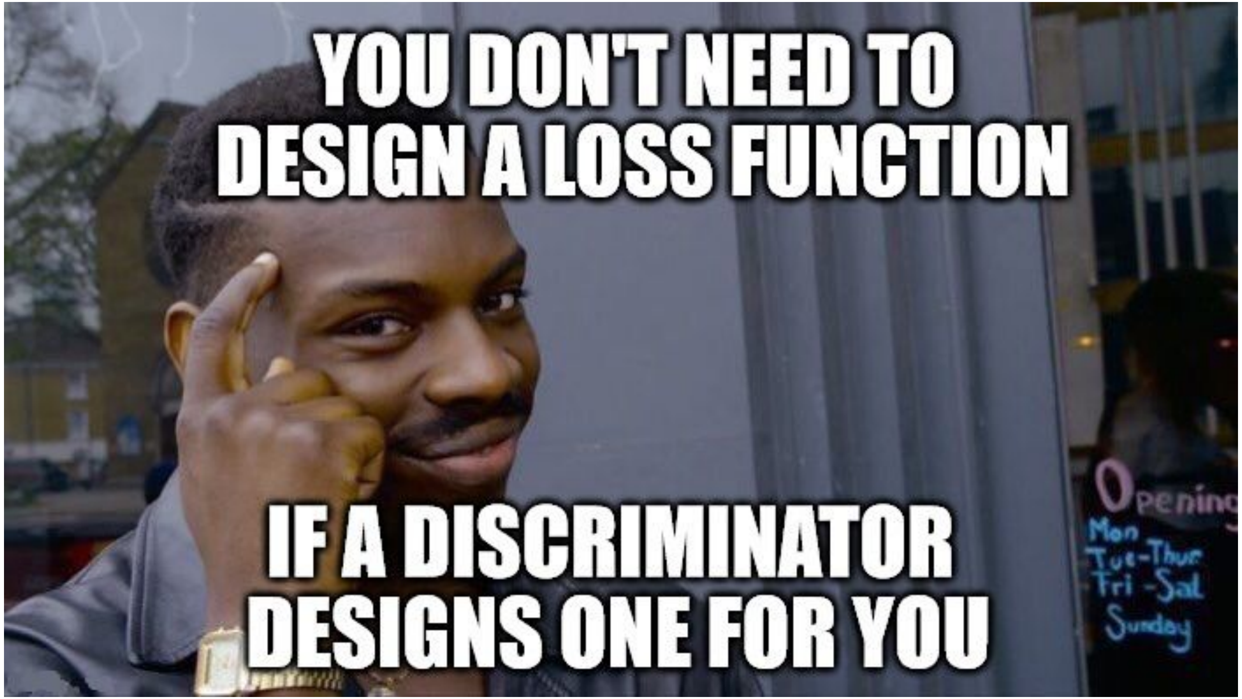


Idea of Adversarial Training

1. Generator - generates fake data based on the training set.
2. Discriminator - tells if it is fake or not.
3. If fake, modify the Generator
4. Else, your network is ready!

Implemented by a Minimax loss function where the change in gradient is determined by how badly the discriminator accepts or rejects generated data.

Idea of Adversarial Training



http://guimperarnau.com/blog/2017/11/Fantastic-GANs-and-where-to-find-them-II

Idea

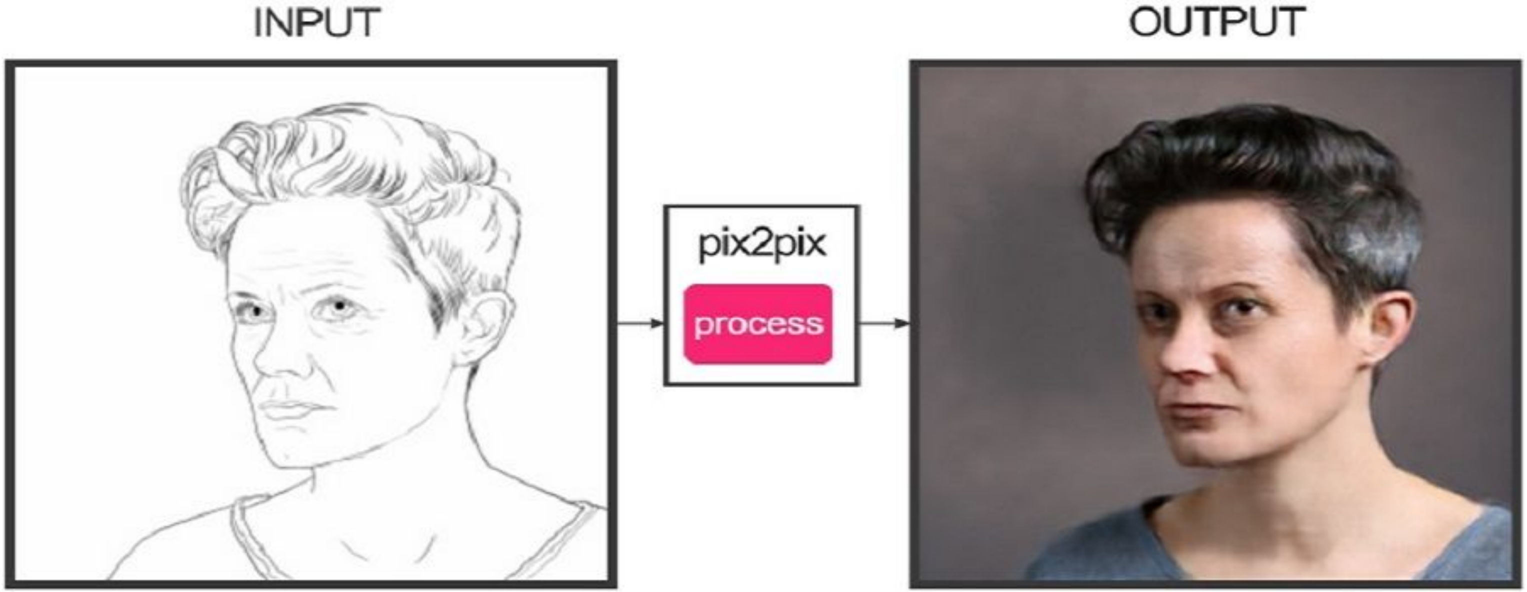
https://www.youtube.com/watch?v=ZRgwcMqxhPw

Visualizing GANs

IDEA

https://creativemarket.com/Meilun/2297571-Caught-thief-loot-policeman

Pix2Pix



http://chattr.com.au/2017/06/19/pix2pix-ai-drawing/

Super Resolution (SRGAN)

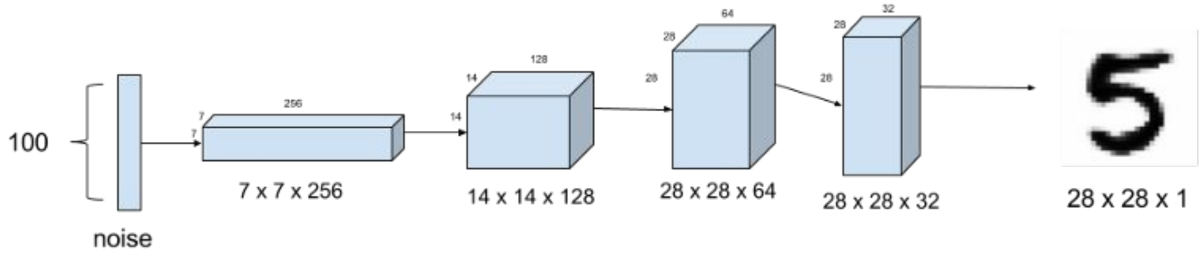
https://github.com/tadax/srgan pic.twitter.com/9wzE2jBxmj

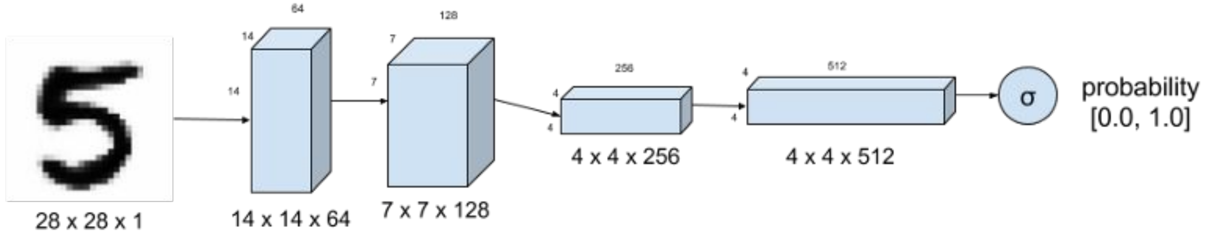
Dense Pose Transfer

https://www.youtube.com/watch?time_continue=141&v=Kmaf8wLqljU

Scene Generation

Generating a number from Noise





https://towardsdatascience.com/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0

Code - Discriminator

```
self.D = Sequential()

depth = 64

dropout = 0.4

# In: 28 x 28 x 1, depth = 1

# Out: 14 x 14 x 1, depth=64

input_shape = (self.img_rows, self.img_cols, self.channel)

self.D.add(Conv2D(depth*1, 5, strides=2, input_shape=input_shape,\

padding='same', activation=LeakyReLU(alpha=0.2)))

self.D.add(Dropout(dropout))

self.D.add(Conv2D(depth*2, 5, strides=2, padding='same',\

activation=LeakyReLU(alpha=0.2)))

self.D.add(Dropout(dropout))

self.D.add(Conv2D(depth*4, 5, strides=2, padding='same',\

activation=LeakyReLU(alpha=0.2)))

self.D.add(Dropout(dropout))

self.D.add(Conv2D(depth*8, 5, strides=1, padding='same',\

activation=LeakyReLU(alpha=0.2)))

self.D.add(Dropout(dropout))

# Out: 1-dim probability

self.D.add(Flatten())

self.D.add(Dense(1))

self.D.add(Activation('sigmoid'))
```

Code - Generator

```
self.G = Sequential()

dropout = 0.4

depth = 64+64+64+64

dim = 7

# In: 100

# Out: dim x dim x depth

self.G.add(Dense(dim*dim*depth, input_dim=100))

self.G.add(BatchNormalization(momentum=0.9))

self.G.add(Activation('relu'))

self.G.add(Reshape((dim, dim, depth)))

self.G.add(Dropout(dropout))

# In: dim x dim x depth

# Out: 2*dim x 2*dim x depth/2
```

```
self.G.add(UpSampling2D())

self.G.add(Conv2DTranspose(int(depth/2), 5, padding='same'))

self.G.add(BatchNormalization(momentum=0.9))

self.G.add(Activation('relu'))

self.G.add(UpSampling2D())

self.G.add(Conv2DTranspose(int(depth/4), 5, padding='same'))

self.G.add(BatchNormalization(momentum=0.9))

self.G.add(Activation('relu'))

self.G.add(Conv2DTranspose(int(depth/8), 5, padding='same'))

self.G.add(BatchNormalization(momentum=0.9))

self.G.add(Activation('relu'))

# Out: 28 x 28 x 1 grayscale image [0.0,1.0] per pix

self.G.add(Conv2DTranspose(1, 5, padding='same'))

self.G.add(Activation('sigmoid'))

self.G.summary()

return self.G
```

Code - Combining

- Compile individual models of Generator and Discriminator with hyperparameters
- Training can can be done individually initially or the entire stack can be combined as well!
- Combining the models to make a complete GAN

```
optimizer = RMSprop(lr=0.0004, clipvalue=1.0, decay=3e-8)

self.AM = Sequential()

self.AM.add(self.generator())

self.AM.add(self.discriminator())

self.AM.compile(loss='binary_crossentropy', optimizer=optimizer,\

metrics=['accuracy'])
```

Learning Resources

- Mother Paper - https://arxiv.org/abs/1406.2661
- Talk by Ian Goodfellow - https://www.youtube.com/watch?v=9JpdAg6uMXs
- http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf
- https://www.quora.com/What-are-good-resources-to-learn-about-Generative-adversarial-networks

Fun Links

- https://reiinakano.github.io/gan-playground/
- https://dena.com/intl/anime-generation/
- https://cs.stanford.edu/people/karpathy/gan/

And many more ...

Thank You