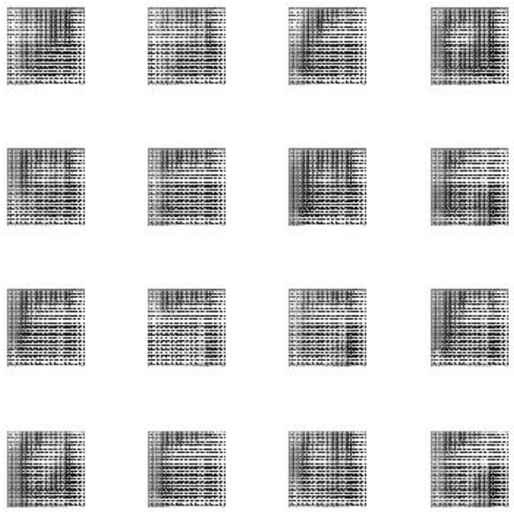
APPLYING GANs IN THE MEDICAL INDUSTRY

ARTURO POLANCO LOZANO
Universidad Surcolombiana

CONTENT

1. *Generative Modelling*
2. Generative Adversarial Networks
3. Data
4. Generating Medical Images with GANs
5. Segmentation of Medical Images with GANs



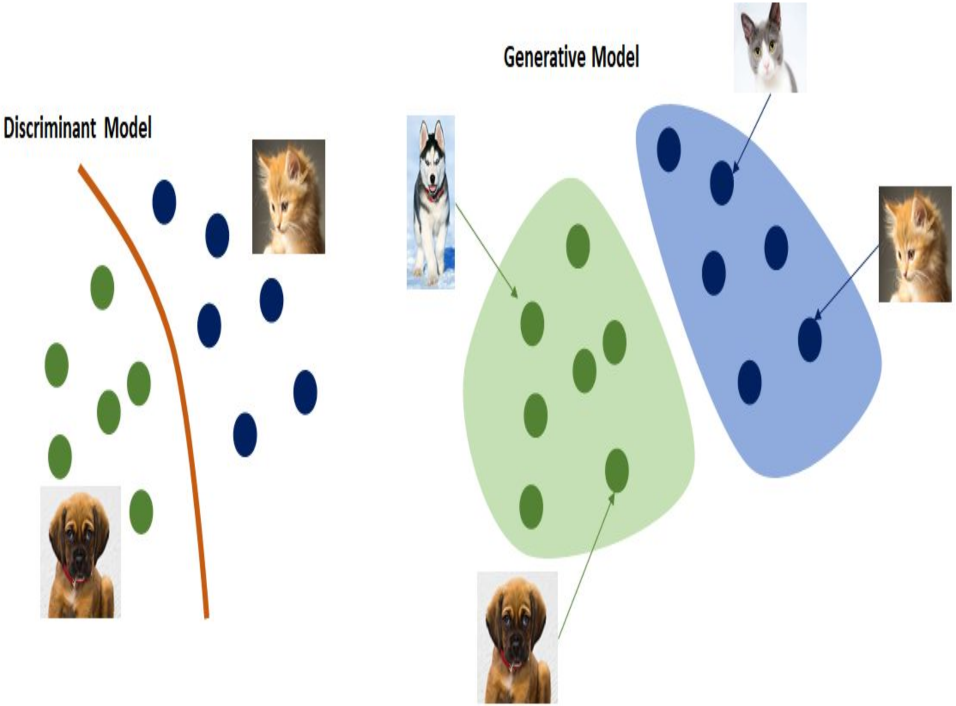
Generative VS Discriminative

Discriminative modeling estimates $p(y|\mathbf{x})$—the probability of a label *y* given observation $\mathbf{x}$.

Generative modeling estimates $p(\mathbf{x})$—the probability of observing observation $\mathbf{x}$.

If the dataset is labeled, we can also build a generative model that estimates the distribution $p(\mathbf{x}|y)$.

Generative VS Discriminative

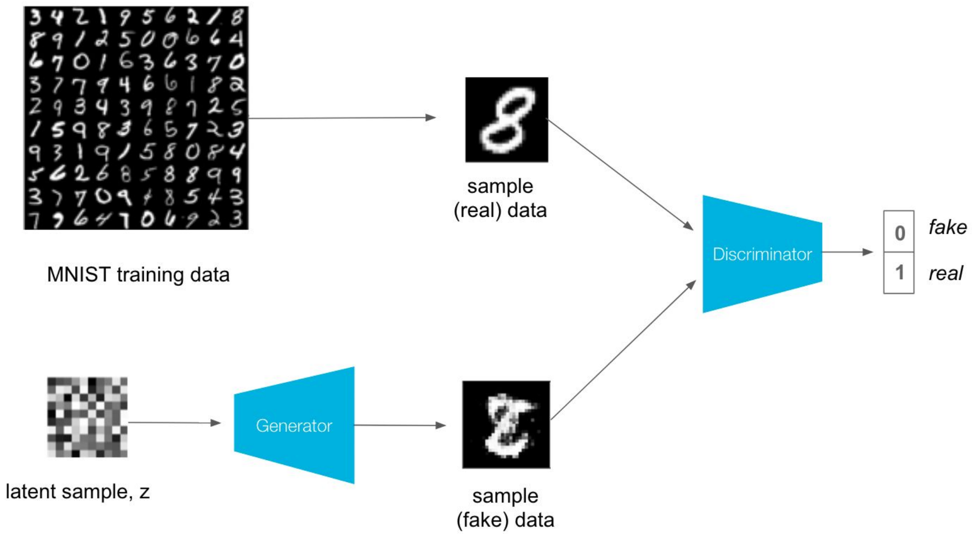


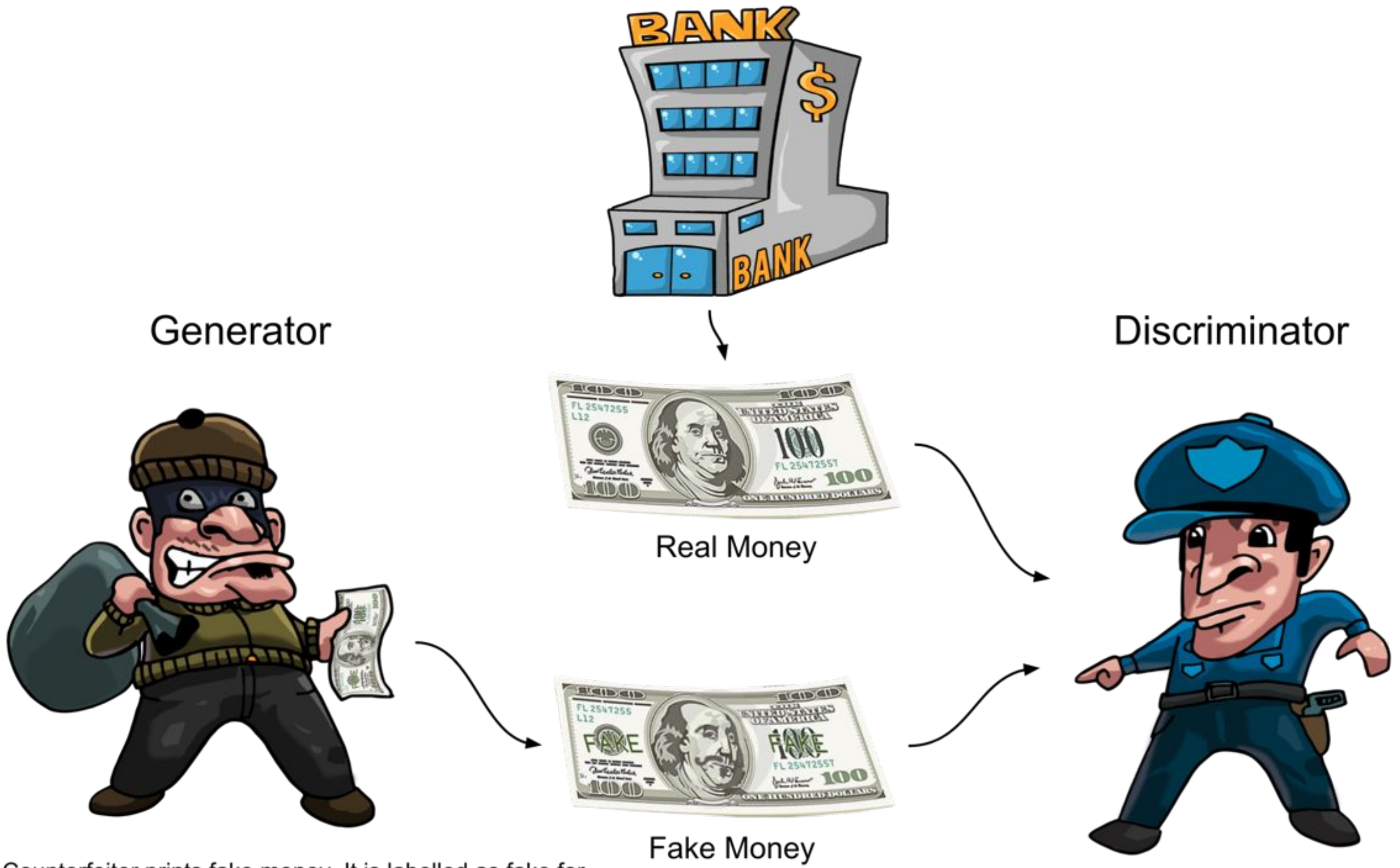
CONTENT

1. Generative Modelling
2. *<u>Generative Adversarial Networks</u>*
3. Data
4. Generating Medical Images with GANs
5. Segmentation of Medical Images with GANs

Generative Adversarial Networks

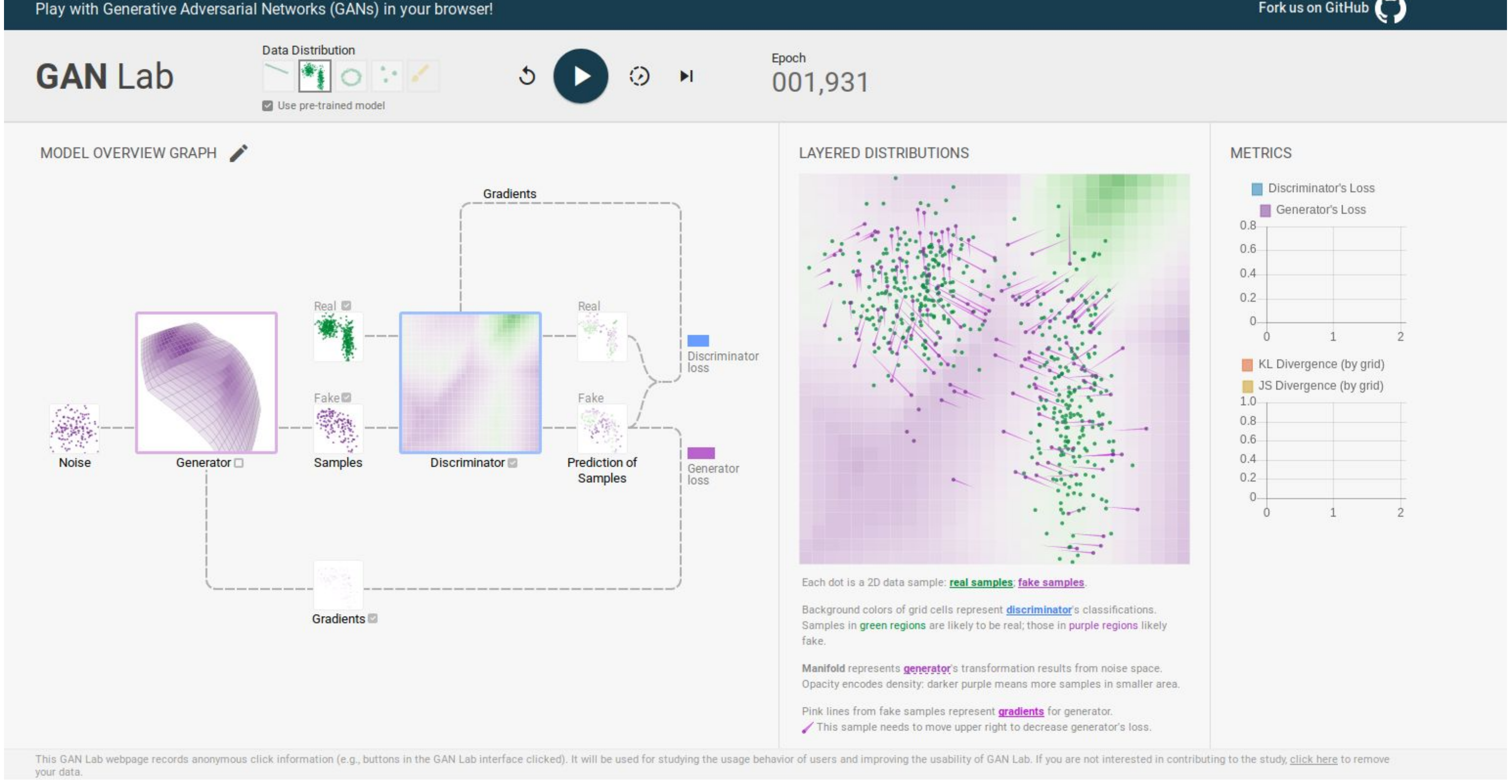
A GAN is a generative model that is trained using two neural network models. One model is called the "*generator*" or "*generative network*" model that learns to generate new plausible samples. The other model is called the "*discriminator*" or "*discriminative network*" and learns to differentiate generated examples from real examples.





Counterfeiter prints fake money. It is labelled as fake for police training. Sometimes, the counterfeiter attempts to fool the police by labelling the fake money as real.

The police are trained to spot real from fake money. Sometimes, the police give feedback to the counterfeiter why the money is fake.



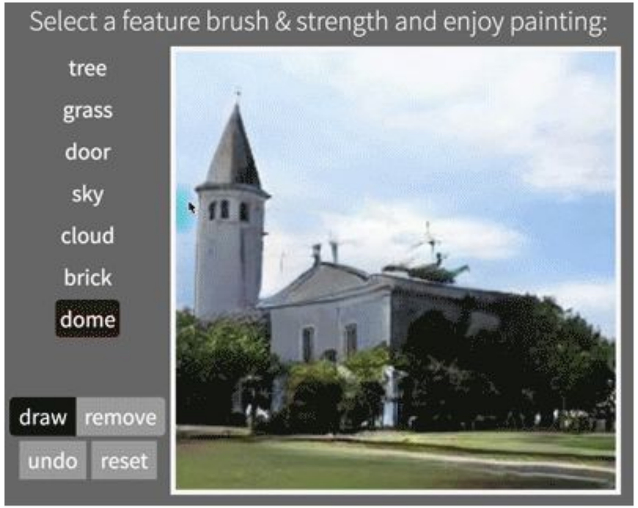
https://poloclub.github.io/ganlab/

This Website Generates AI Portraits of People Who Don’t Exist

FEB 19, 2019 MICHAEL ZHANG

Share 253 Tweet 21 COMMENTS

https://thispersondoesnotexist.com/



The #GANpaint app works by directly activating and deactivating sets of neurons in a deep network trained to generate images. Each button on the left ("door", "brick", etc) corresponds to a set of 20 neurons. The app demonstrates that, by learning to draw, the network also learns about objects such as trees and doors and rooftops. By switching neurons directly, you can observe the structure of the visual world that the network has learned to model. (Try it here.)

https://gandissect.csail.mit.edu/

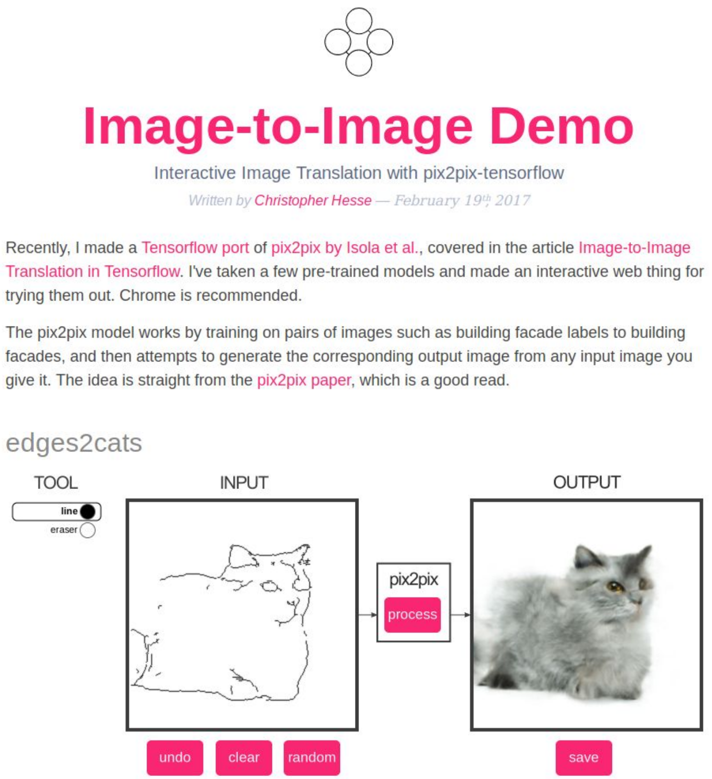


Image-to-Image Demo

Interactive Image Translation with pix2pix-tensorflow

Written by Christopher Hesse — February 19th, 2017

Recently, I made a Tensorflow port of pix2pix by Isola et al., covered in the article Image-to-Image Translation in Tensorflow. I've taken a few pre-trained models and made an interactive web thing for trying them out. Chrome is recommended.

The pix2pix model works by training on pairs of images such as building facade labels to building facades, and then attempts to generate the corresponding output image from any input image you give it. The idea is straight from the pix2pix paper, which is a good read.

edges2cats

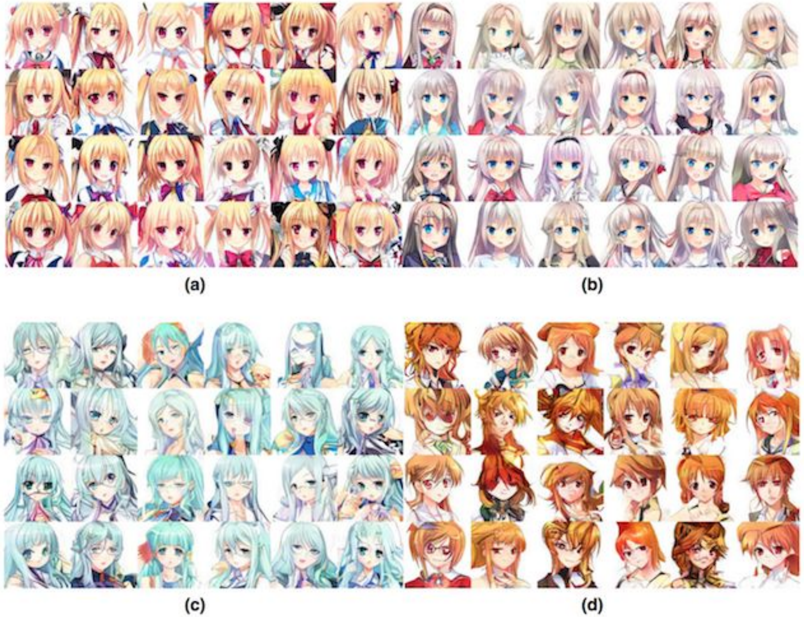
https://affinelayer.com/pixsrv/

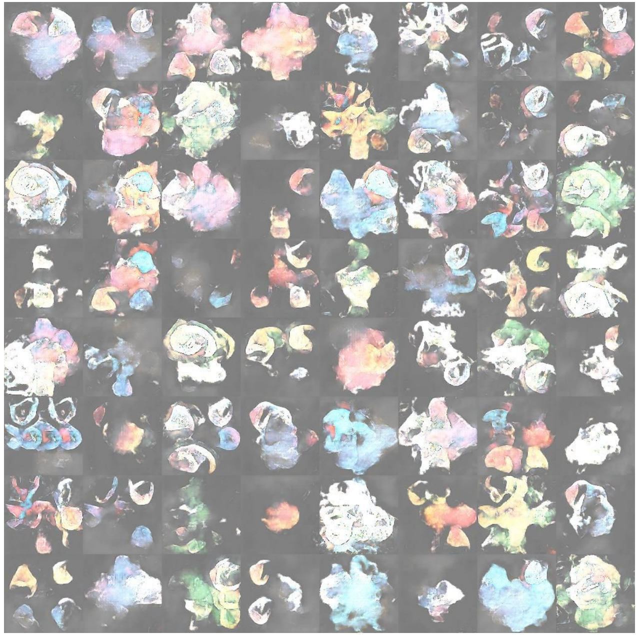
Applications of GANs in Images

- Generate Examples for Image Datasets
- Generate Photographs of Human Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation
- Face Frontal View Generation
- Generate New Human Poses
- Photos to Emojis
- Photograph Editing
- Face Aging
- Photo Blending
- Super Resolution
- Photo Inpainting
- Clothing Translation
- Video Prediction
- 3D Object Generation

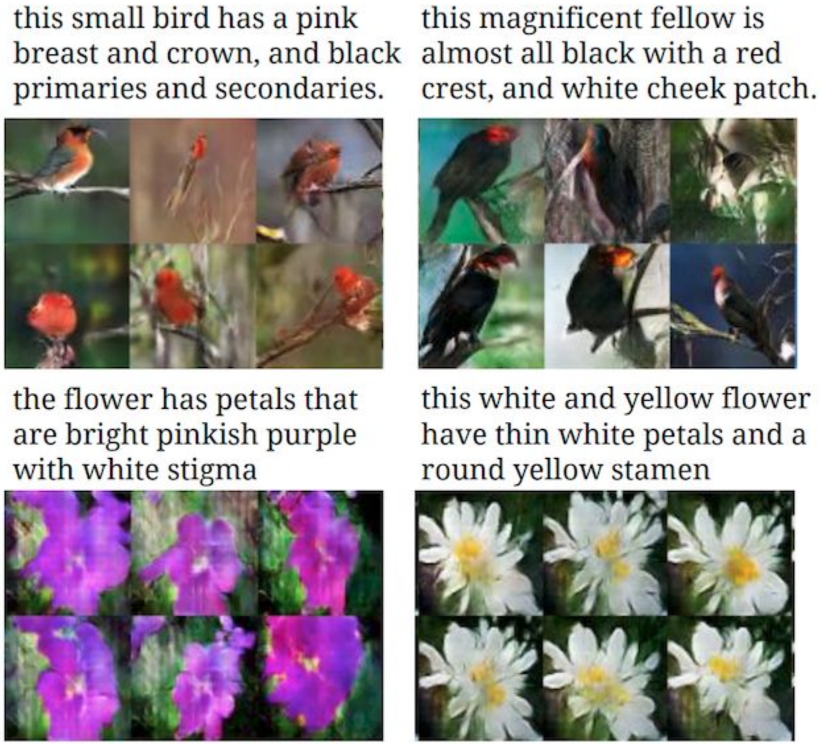
Applications of GANs in Images Conditional AN

Applications of GANs in Images DC GAN

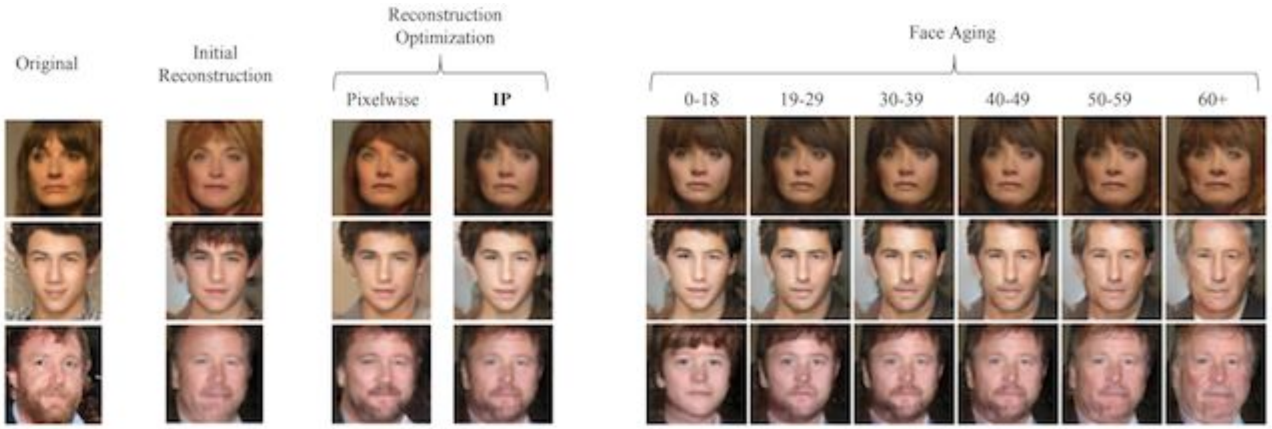




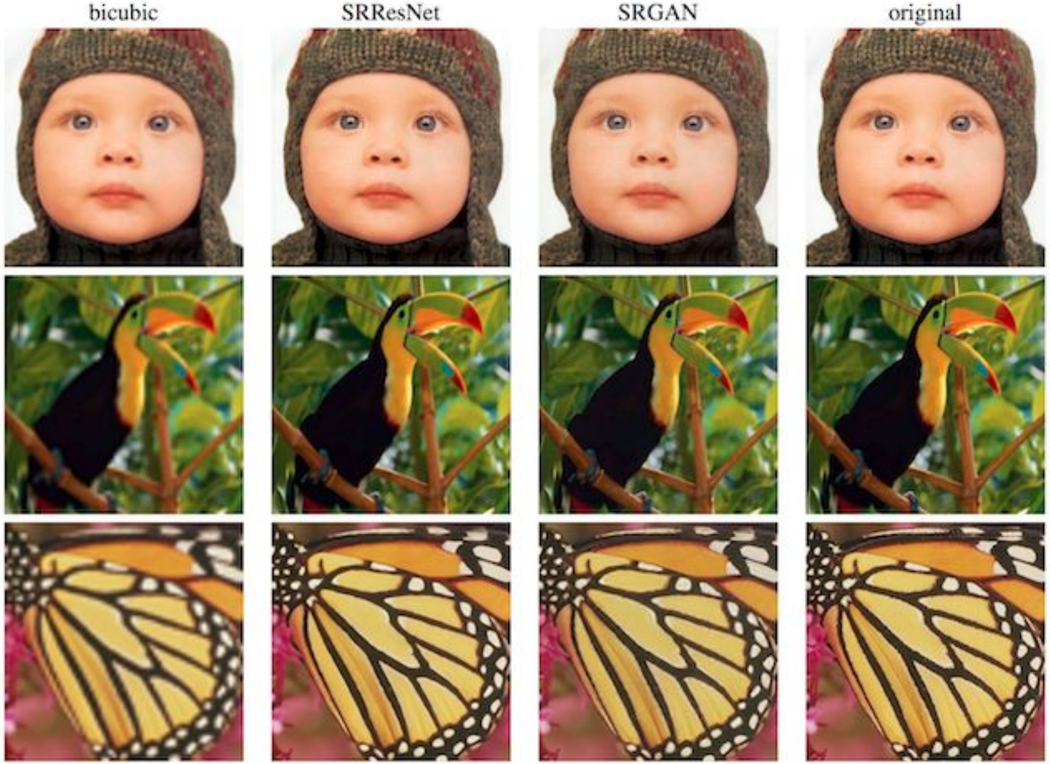
Applications of GANs in Images (StackGan)



Applications of GANs in Images Conditional GAN

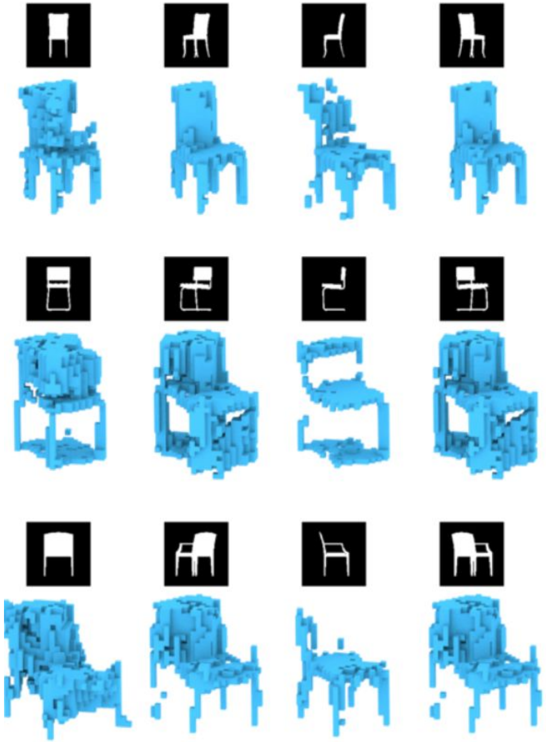


Applications of GANs in Images (SRGAN)

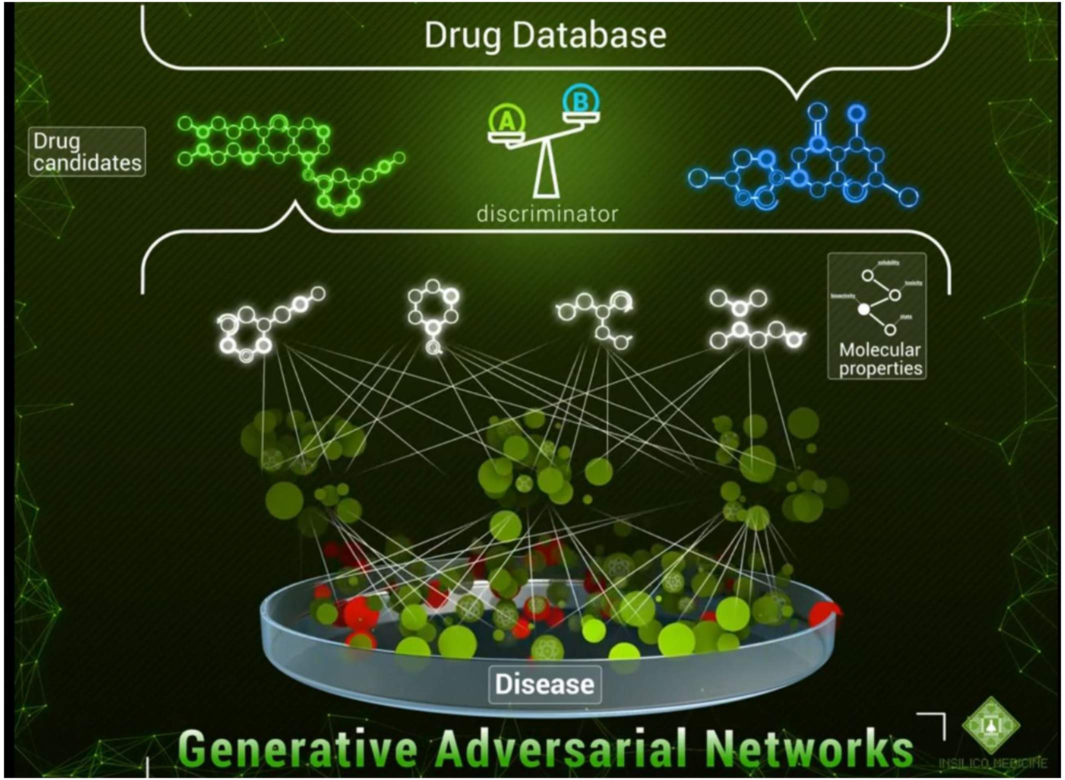


Applications of GANs in Images

Applications of GANs in Images



Other Applications of GANs



Other Applications of GANs

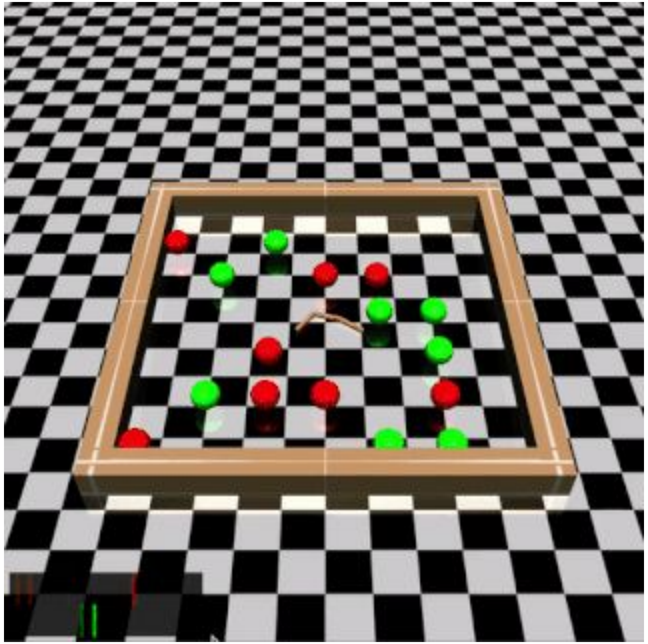
Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis

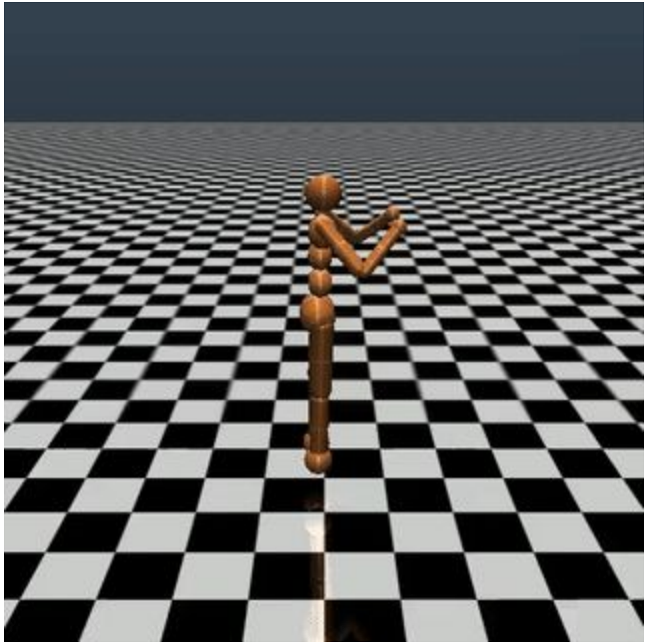
Luke de Oliveira[a], Michela Paganini[a,b], and Benjamin Nachman[a]

[a] *Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA, 94720, USA*

[b] *Department of Physics, Yale University, New Haven, CT 06520, USA*

Other Applications of GANs





Visit The GAN Zoo on Github !

The GAN Zoo

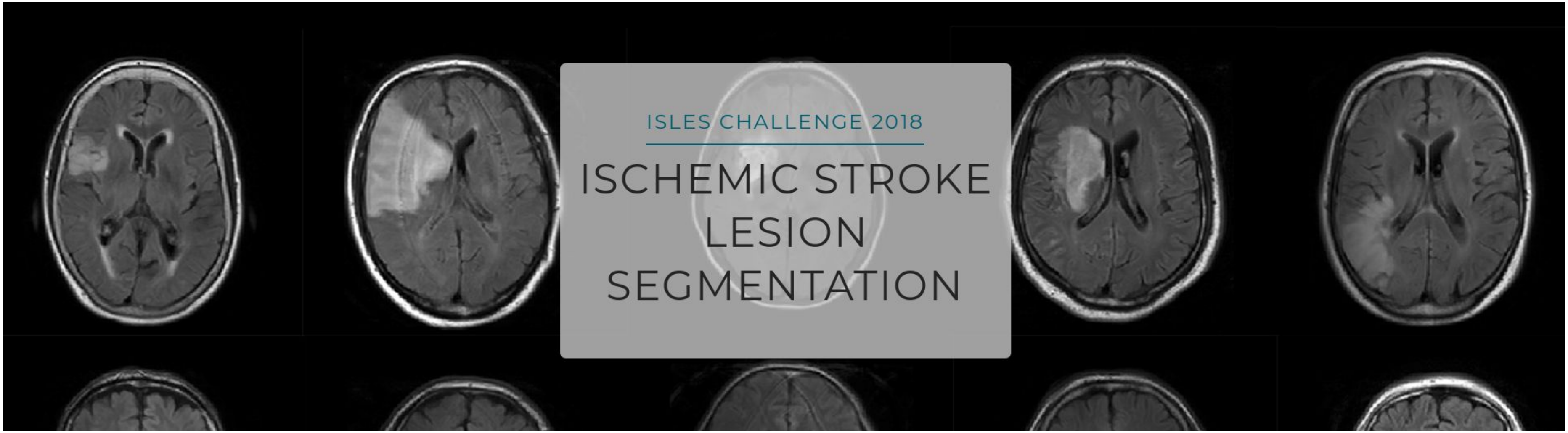
Every week, new GAN papers are coming out and it's hard to keep track of them all, not to mention the incredibly creative ways in which researchers are naming these GANs! So, here's a list of what started as a fun activity compiling all named GANs!

CONTENT

1. Generative Modelling
2. Generative Adversarial Networks
3. *<u>Data</u>*
4. Generating Medical Images with GANs
5. Segmentation of Medical Images with GANs

Multimodal Brain Tumor Segmentation Challenge 2018

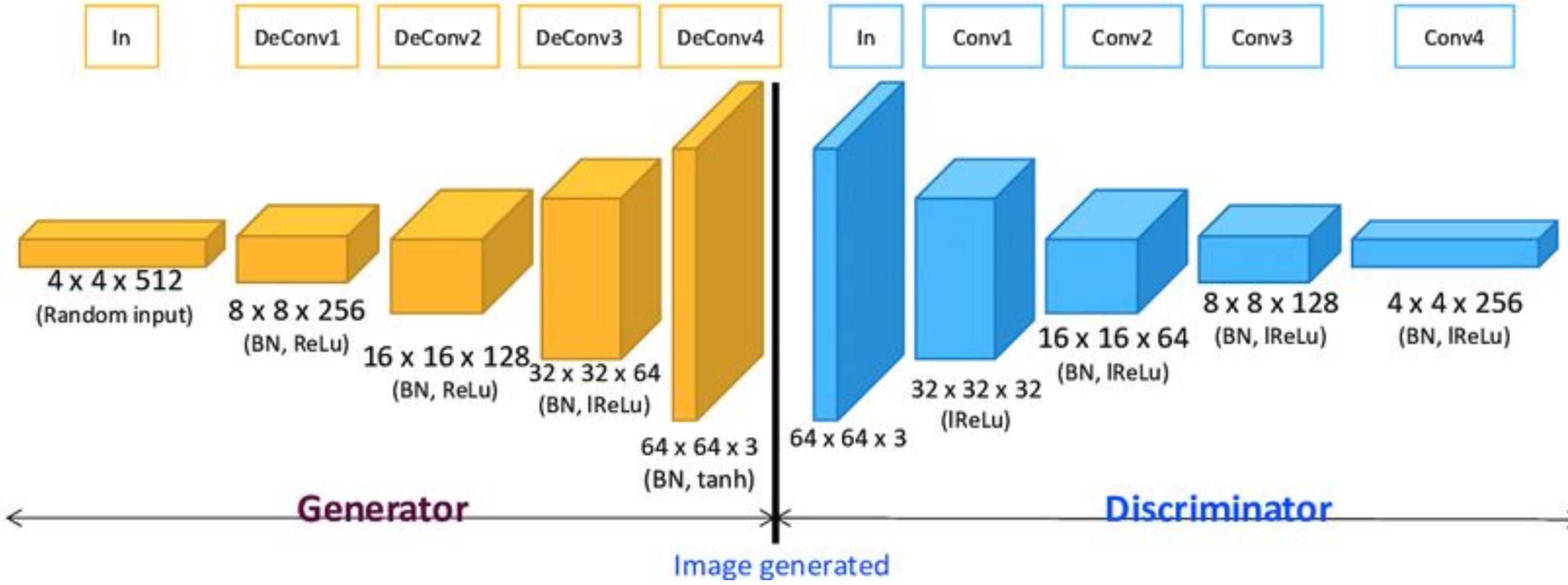
• Scope • Relevance • Tasks • Data • Evaluation • Participation Summary • Data Request • Previous BraTS • People •


ISLES CHALLENGE 2018
ISCHEMIC STROKE
LESION
SEGMENTATION

CONTENT

1. Generative Modelling
2. Generative Adversarial Networks
3. Data
4. *<u>Generating Medical Images with GANs</u>*
5. Segmentation of Medical Images with GANs

Deep Convolutional GAN



Discriminator

```
class Discriminator(tf.keras.Model):
  def __init__(self):
    super(Discriminator, self).__init__()
    self.conv1 = tf.keras.layers.Conv2D(64, (4, 4), strides=(2, 2), padding='same')
    self.conv2 = tf.keras.layers.Conv2D(64 * 2, (4, 4), strides=(2, 2), padding='same')
    self.batchnorm2 = tf.keras.layers.BatchNormalization()
    self.conv3 = tf.keras.layers.Conv2D(64 * 4, (4, 4), strides=(2, 2), padding='same')
    self.batchnorm3 = tf.keras.layers.BatchNormalization()
    self.conv4 = tf.keras.layers.Conv2D(64 * 8, (4, 4), strides=(2, 2), padding='same')
    self.batchnorm4 = tf.keras.layers.BatchNormalization()
    self.conv5 = tf.keras.layers.Conv2D(1, (4, 4), strides=(1, 1), padding='valid')

    self.dropout = tf.keras.layers.Dropout(0.3)
    self.flatten = tf.keras.layers.Flatten()
    self.fc1 = tf.keras.layers.Dense(1)

  def call(self, x, training=True):
    x = tf.nn.leaky_relu(self.conv1(x))

    x = self.conv2(x)
    x = self.batchnorm2(x, training=training)
    x = tf.nn.leaky_relu(x)

    x = self.conv3(x)
    x = self.batchnorm3(x, training=training)
    x = tf.nn.leaky_relu(x)

    x = self.conv4(x)
    x = self.batchnorm4(x, training=training)
    x = tf.nn.leaky_relu(x)

    x = self.conv5(x)
    x = self.fc1(x)
    return x
```

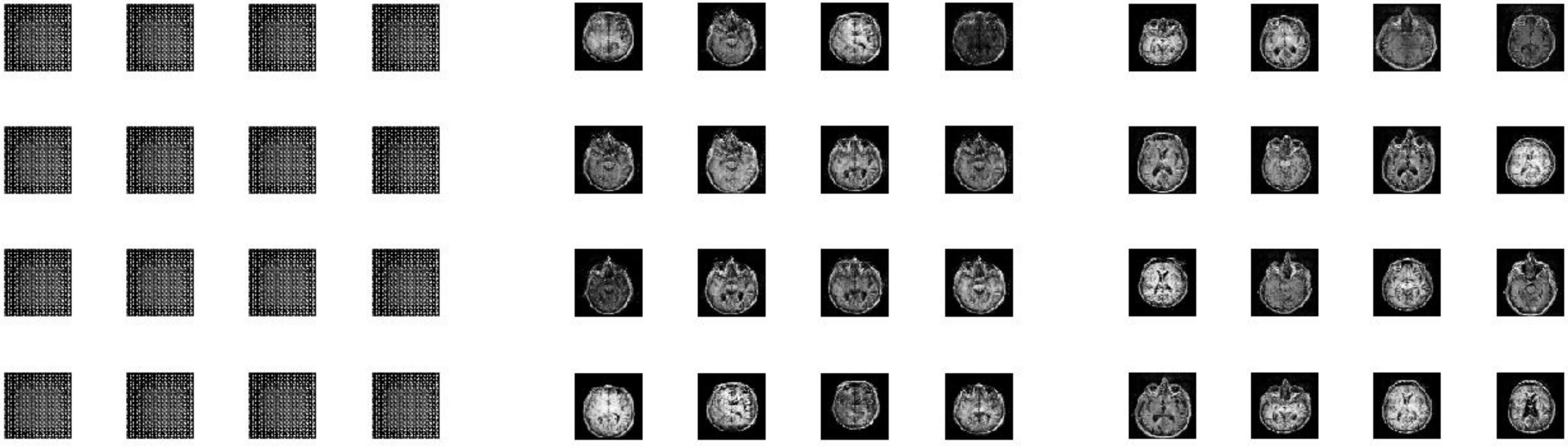
Generator

```
class Generator(tf.keras.Model):
  def __init__(self):
    super(Generator, self).__init__()
    self.fc1 = tf.keras.layers.Dense(8*8*64, use_bias=False)
    self.batchnorm1 = tf.keras.layers.BatchNormalization()

    self.conv1 = tf.keras.layers.Conv2DTranspose(64 * 8, (4, 4), strides=(1, 1), padding='same', use_bias=False)
    self.batchnorm2 = tf.keras.layers.BatchNormalization()

    self.conv2 = tf.keras.layers.Conv2DTranspose(64 * 4, (4, 4), strides=(2, 2), padding='same', use_bias=False)
    self.batchnorm3 = tf.keras.layers.BatchNormalization()

    self.conv3 = tf.keras.layers.Conv2DTranspose(64 * 2, (4, 4), strides=(2, 2), padding='same', use_bias=False)
    self.batchnorm4 = tf.keras.layers.BatchNormalization()

    self.conv4 = tf.keras.layers.Conv2DTranspose(64 * 1, (4, 4), strides=(2, 2), padding='same', use_bias=False)
    self.batchnorm5 = tf.keras.layers.BatchNormalization()

    self.conv5 = tf.keras.layers.Conv2DTranspose(1, (4, 4), strides=(1, 1), padding='same', use_bias=False)

  def call(self, x, training=True):
    x = self.fc1(x)
    x = self.batchnorm1(x, training=training)
    x = tf.nn.relu(x)

    x = tf.reshape(x, shape=(-1, 8, 8, 64))

    x = self.conv1(x)
    x = self.batchnorm2(x, training=training)
    x = tf.nn.relu(x)

    x = self.conv2(x)
    x = self.batchnorm3(x, training=training)
    x = tf.nn.relu(x)

    x = self.conv3(x)
    x = self.batchnorm4(x, training=training)
    x = tf.nn.relu(x)

    x = self.conv4(x)
    x = self.batchnorm5(x, training=training)
    x = tf.nn.relu(x)

    x = tf.nn.tanh(self.conv5(x))
    return x
```

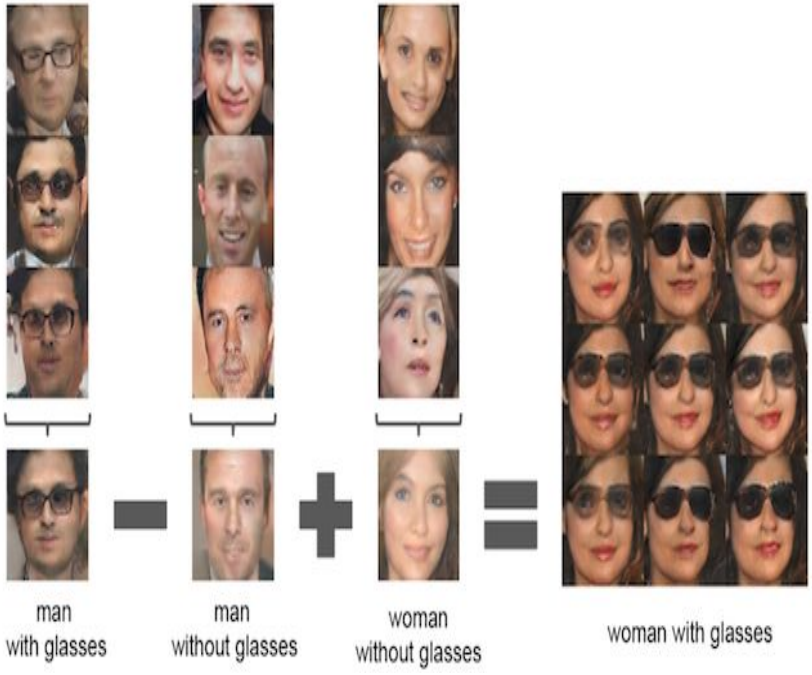
Discriminator and Generator Loss

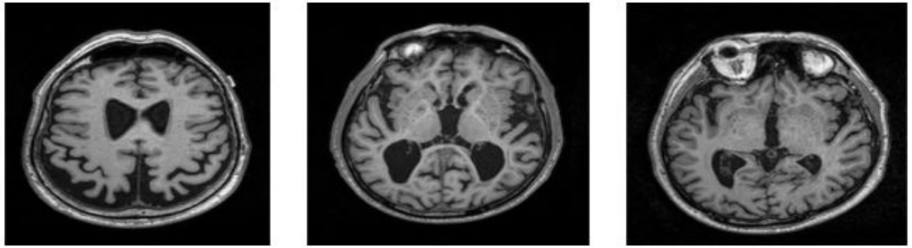
```python
def discriminator_loss(real_output, generated_output):
    # [1,1,...,1] with real output since it is true and we want
    # our generated examples to look like it
    real_loss = tf.losses.sigmoid_cross_entropy(multi_class_labels=tf.ones_like(real_output), logits=real_output)

    # [0,0,...,0] with generated images since they are fake
    generated_loss = tf.losses.sigmoid_cross_entropy(multi_class_labels=tf.zeros_like(generated_output), 
logits=generated_output)

    total_loss = real_loss + generated_loss

    return total_loss


def generator_loss(generated_output):
    return tf.losses.sigmoid_cross_entropy(tf.ones_like(generated_output), generated_output)
```

Output



Explore Latent Space





Goals

- In order to use larger deep learning models is required hughes amounts of data, GANs can be used for data augmentation instead of other methods that involve image cropping, translation, etc.
- Some pathologic findings are rare, GANs can be used for generating images with specific features.
- Medical data needs to remain private, using GANs in order to create synthetic images helps to solve data anonymization problem.

CONTENT

1. Generative Modelling
2. Generative Adversarial Networks
3. Data
4. Generating Medical Images with GANs
5. *Segmentation of Medical Images with GANs*

Im2Im Translation with Conditional AN

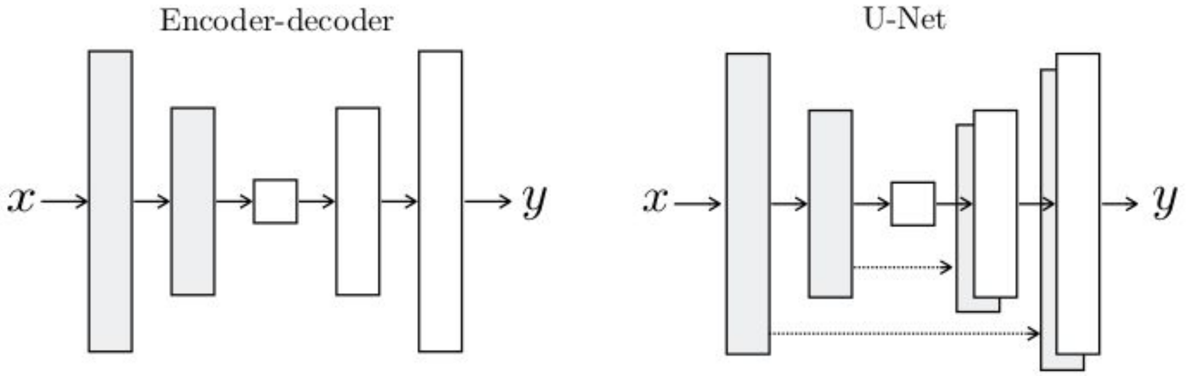
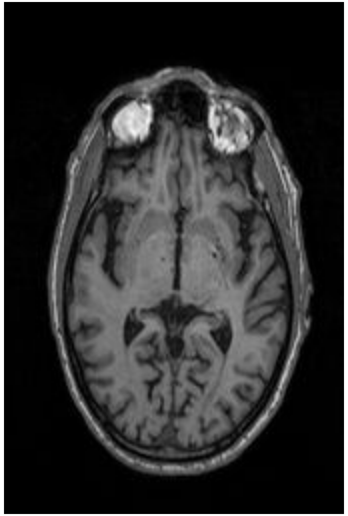
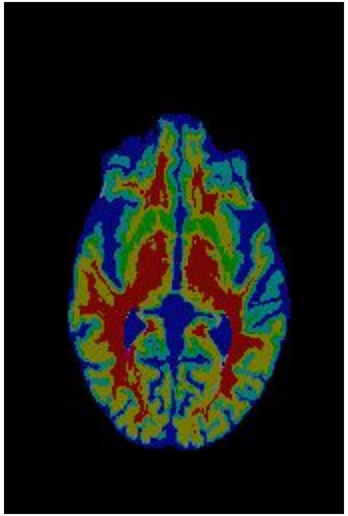


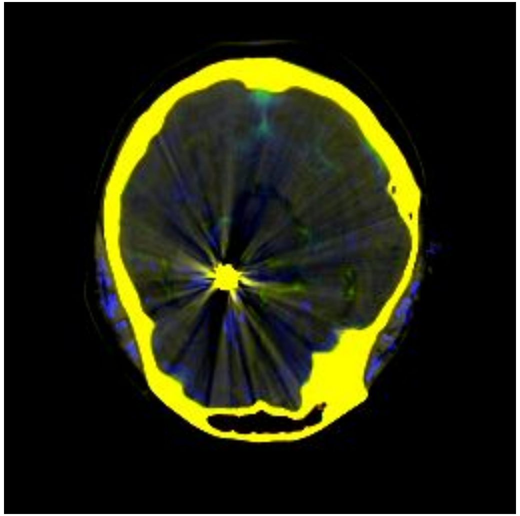
Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

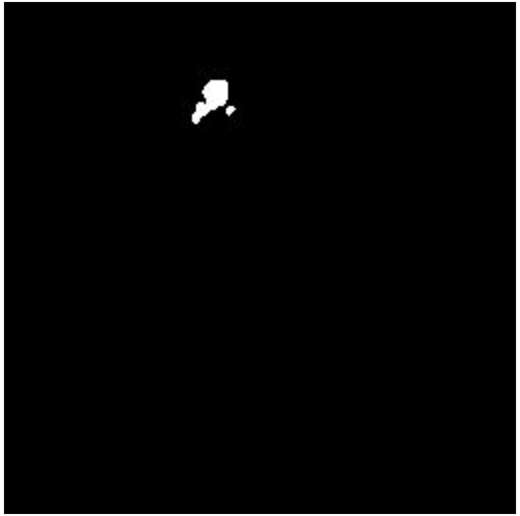
Output





Output





Goals

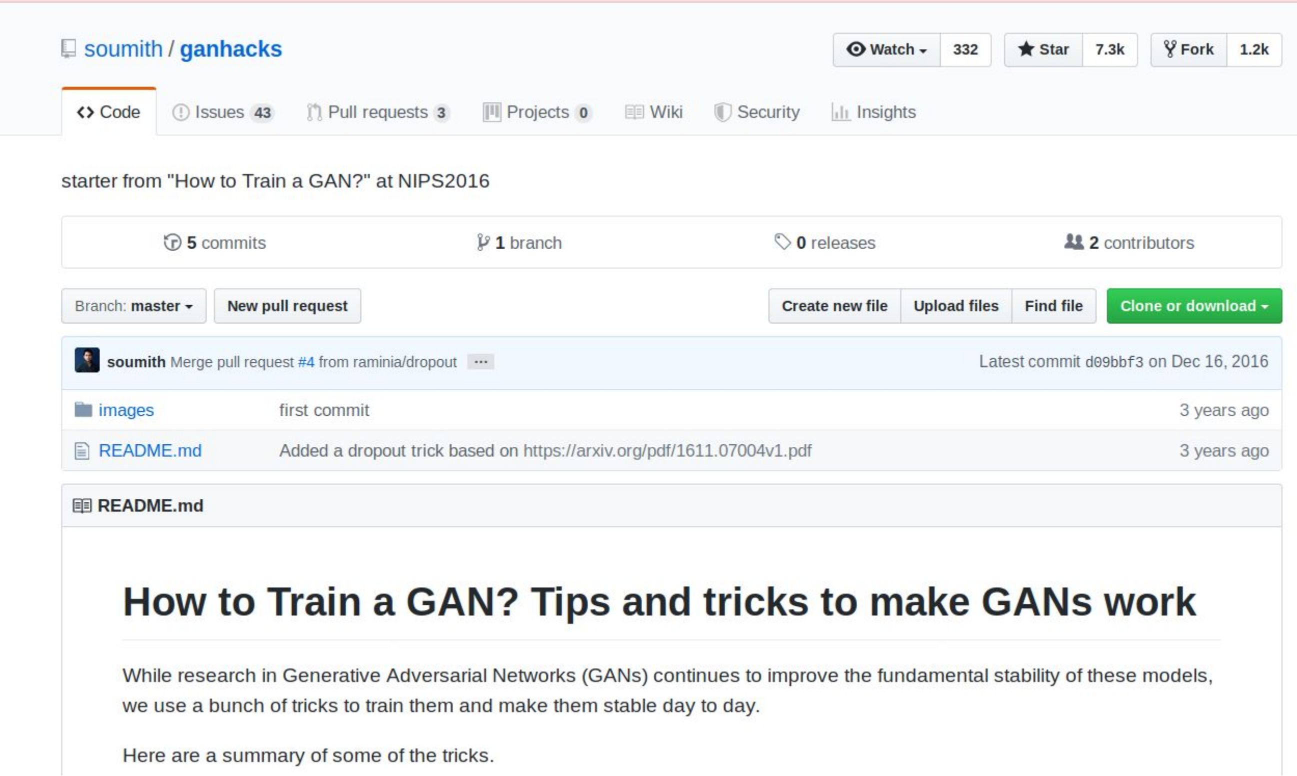
- Comparison between the amount of gray matter, white matter and cerebrospinal fluid are vital for doctors in order to perform diagnostics.
- The diagnostic of ischemic stroke is time sensitive, a delay in the treatment could signify in permanent damage of death of the patient.

GAN Hacks

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

GAN Hacks


soumith / ganhacks
Watch 332
Star 7.3k
Fork 1.2k
Code
Issues 43
Pull requests 3
Projects 0
Wiki
Security
Insights
starter from "How to Train a GAN?" at NIPS2016
5 commits
1 branch
0 releases
2 contributors
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
soumith Merge pull request #4 from raminia/dropout
Latest commit d09bbf3 on Dec 16, 2016
images
first commit
3 years ago
README.md
Added a dropout trick based on https://arxiv.org/pdf/1611.07004v1.pdf
3 years ago
README.md
How to Train a GAN? Tips and tricks to make GANs work
While research in Generative Adversarial Networks (GANs) continues to improve the fundamental stability of these models, we use a bunch of tricks to train them and make them stable day to day.
Here are a summary of some of the tricks.

Thanks :-)