The Future History of Generative Adversarial Networks

mertz@gnosis.cx

Generative Adversarial Networks (GANs)

Two distinct neural networks are placed in competition with each other

One is called a “generator” and tries to produce authentic-seeming data

The second is called a “discriminator” and tries to distinguish genuine from synthetic data

Generative Adversarial Networks (GANs)

Two distinct neural networks are placed in competition with each other

One is called a "generator" and tries to produce authentic-seeming data

The second is called a "discriminator" and tries to distinguish genuine from synthetic data

Generative Adversarial Networks (GANs)

Two distinct neural networks are placed in competition with each other

One is called a “generator” and tries to produce authentic-seeming data

The second is called a “discriminator” and tries to distinguish genuine from synthetic data

The Context of Generative Adversarial Networks

GANs have been used most widely in image generation contexts

Can be applied equally to other domains

When applied to images, GAN's often produce "surreal" and sometimes disturbing resemblances to real images.

While a GAN is technically a kind of unsupervised learning, it cleverly captures much of the power of supervised learning models.

The Context of Generative Adversarial Networks

GANs have been used most widely in image generation contexts

Can be applied equally to other domains

When applied to images, GAN's often produce "surreal" and sometimes disturbing resemblances to real images.

While a GAN is technically a kind of unsupervised learning, it cleverly captures much of the power of supervised learning models.

The Context of Generative Adversarial Networks

GANs have been used most widely in image generation contexts

Can be applied equally to other domains

When applied to images, GAN's often produce "surreal" and sometimes disturbing resemblances to real images.

While a GAN is technically a kind of unsupervised learning, it cleverly captures much of the power of supervised learning models.

The Context of Generative Adversarial Networks

GANs have been used most widely in image generation contexts

Can be applied equally to other domains

When applied to images, GAN's often produce "surreal" and sometimes disturbing resemblances to real images.

While a GAN is technically a kind of unsupervised learning, it cleverly captures much of the power of supervised learning models.

Supervised Learning

Start out with *tagged training data:*

Classifiers predict target in several classes

Regressors predict target in continuous numeric range

Require initial identification of *canonical answers*:

For example, labeled using human judgment

Supervised Learning

Start out with *tagged training data:*

Classifiers predict target in several classes

Regressors predict target in continuous numeric range

Require initial identification of *canonical answers*:

For example, labeled using human judgment

Unsupervised Learning

Data features, but no target per se

No *a priori* to compare to prediction

For example, *clustering*, *decomposition*

Generative Adversarial Network

We only have examples of the positive class

Implicit negative class is "anything else"

The *adversaries* are supervised models

The adversaries provide each other's targets

The Future History of Generative Adversarial Networks

An Uncanny Valley Missed by a Discriminator

Perceptual Verisimilitude (Aspiring Android Actors)

The “Goal” of a Generator

Generate new data of needed shape

Its input is simply random noise (usually uniform)

Once trained, its output hopes to resemble the genuine samples

The reward function is fooling the discriminator

The “Goal” of a Generator

Generate new data of needed shape

Its input is simply random noise (usually uniform)

Once trained, its output hopes to resemble the genuine samples

The reward function is fooling the discriminator

The “Goal” of a Generator

Generate new data of needed shape

Its input is simply random noise (usually uniform)

Once trained, its output hopes to resemble the genuine samples

The reward function is fooling the discriminator

The “Goal” of a Generator

Generate new data of needed shape

Its input is simply random noise (usually uniform)

Once trained, its output hopes to resemble the genuine samples

The reward function is fooling the discriminator

The “Goal” of a Discriminator

Distinguish genuine samples from synthetic samples produced by the generator

The reward function is correctly predicting “real” versus “fake”

A generator is specifically trying to outwit the discriminator

Real world data rarely tries to fool a network

The “Goal” of a Discriminator

Distinguish genuine samples from synthetic samples produced by the generator

The reward function is correctly predicting “real” versus “fake”

A generator is specifically trying to outwit the discriminator

Real world data rarely tries to fool a network

The “Goal” of a Discriminator

Distinguish genuine samples from synthetic samples produced by the generator

The reward function is correctly predicting “real” versus “fake”

A generator is specifically trying to outwit the discriminator

Real world data rarely tries to fool a network

The “Goal” of a Discriminator

Distinguish genuine samples from synthetic samples produced by the generator

The reward function is correctly predicting “real” versus “fake”

A generator is specifically trying to outwit the discriminator

Real world data rarely tries to fool a network

The Future History of Generative Adversarial Networks

However ...

In forgery or fraud, a malicious actor is trying to create currency, or artwork, or some other item

... that can pass inspection by (human or machine) discriminators

In evolution, some organisms use camouflage to appear as something else

... one can think of evolutionary ecology itself as something closely akin to a GAN

The Future History of Generative Adversarial Networks

However ...

In forgery or fraud, a malicious actor is trying to create currency, or artwork, or some other item

... that can pass inspection by (human or machine) discriminators

In evolution, some organisms use camouflage to appear as something else

... one can think of evolutionary ecology itself as something closely akin to a GAN

A Toy Generative Adversarial Network

Let's see a simple GAN that creates synthetic samples of a target Gaussian distribution.

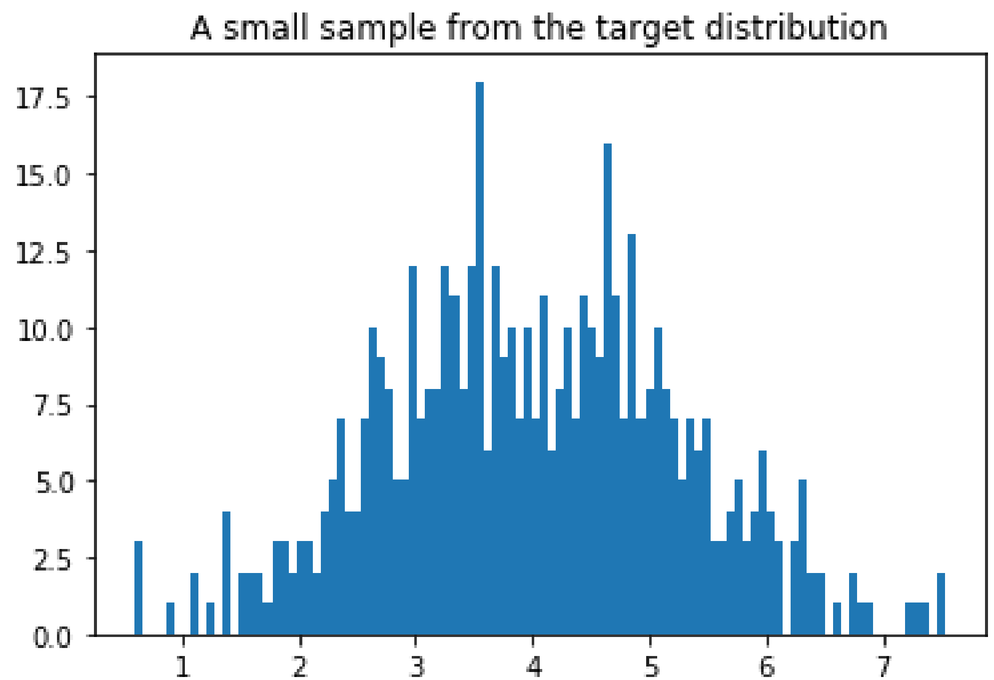
That is, our generator will need to learn what a normal distribution is from scratch

The full code lives at:

```
https://github.com/DavidMertz/GITEX-2021
```

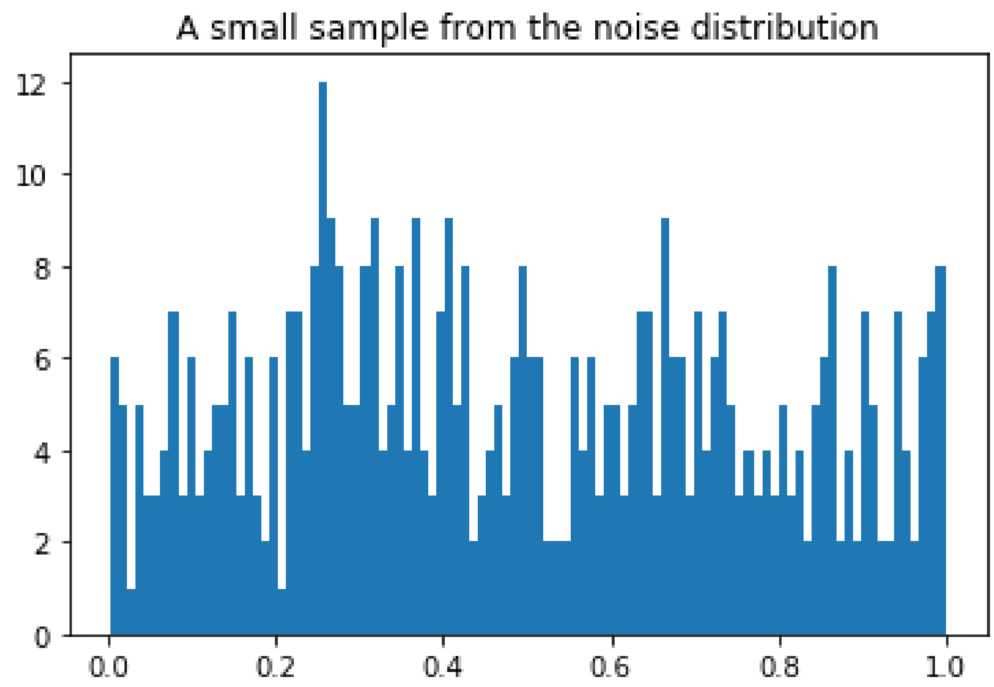
Authentic Samples a Discriminator Sees

```
>>> v = d_sampler()
>>> print("Mean: %.2f | Std: %.2f | "
...       "Skew: %.2f | Kurt: %2f" % stats(v))
Mean: 4.01 | Std: 1.27 | Skew: 0.04 | Kurt: -0.216963
```



Noise Samples a Generator Sees

```
>>> v = gi_sampler()
>>> print("Mean: %.2f | Std: %.2f | "
...       "Skew: %.2f | Kurt: %2f" % stats(v))
Mean: 0.49 | Std: 0.28 | Skew: 0.13 | Kurt: -1.116548
```



The Generator Model

```python
class Generator(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, f):
        super().__init__()
        self.dropout = nn.Dropout(0.25)
        self.map1 = nn.Linear(input_size, hidden_size)
        self.map2 = nn.Linear(hidden_size, hidden_size)
        self.map3 = nn.Linear(hidden_size, output_size)
        self.f = f

    def forward(self, x):
        x = self.map1(x)
        x = self.dropout(x)  # Can we avoid a local trap?
        x = self.f(x)
        x = self.map2(x)
        x = self.dropout(x)  # Can we avoid a local trap?
        x = self.f(x)
        x = self.map3(x)
        return x
```

The Discriminator Model

This model can be (almost) identical to the generator

```
class Discriminator(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, f):
        super().__init__()
        self.map1 = nn.Linear(input_size, hidden_size)
        self.map2 = nn.Linear(hidden_size, hidden_size)
        self.map3 = nn.Linear(hidden_size, output_size)
        self.f = f

    def forward(self, x):
        x = self.map1(x)
        x = self.f(x)
        x = self.map2(x)
        x = self.f(x)
        x = self.map3(x)
        return self.f(x)
```

Training the Models (per epoch/step)

```
# D(iscriminator) and G(enerator) models

# 1A: Train D on real
d_real_data = d_sampler(d_input_size)
d_real_decision = D(d_real_data)
d_real_error = loss_fn(d_real_decision, torch.ones([1]))
d_real_error.backward() # compute grads, don't change params

# 1B: Train D on fake
d_gen_input = gi_sampler(g_input_size)
d_fake_data = G(d_gen_input).detach()  # DO NOT train G here
d_fake_decision = D(d_fake_data.t())
d_fake_error = loss_fn(d_fake_decision, torch.zeros([1]))
d_fake_error.backward()
d_optimizer.step()     # Only optimizes D's parameters;
```

Training the Models (per epoch/step) *... continued*

```
# D(iscriminator) and G(enerator) models

# 2. Train G on D's response
# (but DO NOT train D on these labels)
G.zero_grad()
gen_input = gi_sampler(g_input_size)
g_fake_data = G(gen_input)
dg_fake_decision = D(g_fake_data.t())

# Train G to pretend it's genuine
g_error = loss_fn(dg_fake_decision, torch.ones([1]))
g_error.backward()
g_optimizer.step()  # Only optimizes G's parameters
```

Pitfalls and Guidelines in Training GANs

When you train the discriminator, the generator will remain constant, and vice versa

In a known domain, you might wish to pre-train the discriminator, or utilize a pre-trained model

> ... this gives the generator a more difficult adversary to work against

Pitfalls and Guidelines in Training GANs *... continued*

- One adversary of the GAN can overpower the other
  - Depends on learning rate, optimizers, loss functions, etc.
- If the discriminator is too good, it will return values close to 0 or 1
  - Generator will be unable to find a meaningful gradient
- If the generator is too good, it will exploit weaknesses in the discriminator
  - Simpler patterns than "authenticity" might fool it
    ... the surreal images demonstrate this

Pitfalls and Guidelines in Training GANs *... continued*

- One adversary of the GAN can overpower the other
  - Depends on learning rate, optimizers, loss functions, etc.
- If the discriminator is too good, it will return values close to 0 or 1
  - Generator will be unable to find a meaningful gradient
- If the generator is too good, it will exploit weaknesses in the discriminator
  - Simpler patterns than "authenticity" might fool it
    ... the surreal images demonstrate this

Pitfalls and Guidelines in Training GANs *... continued*

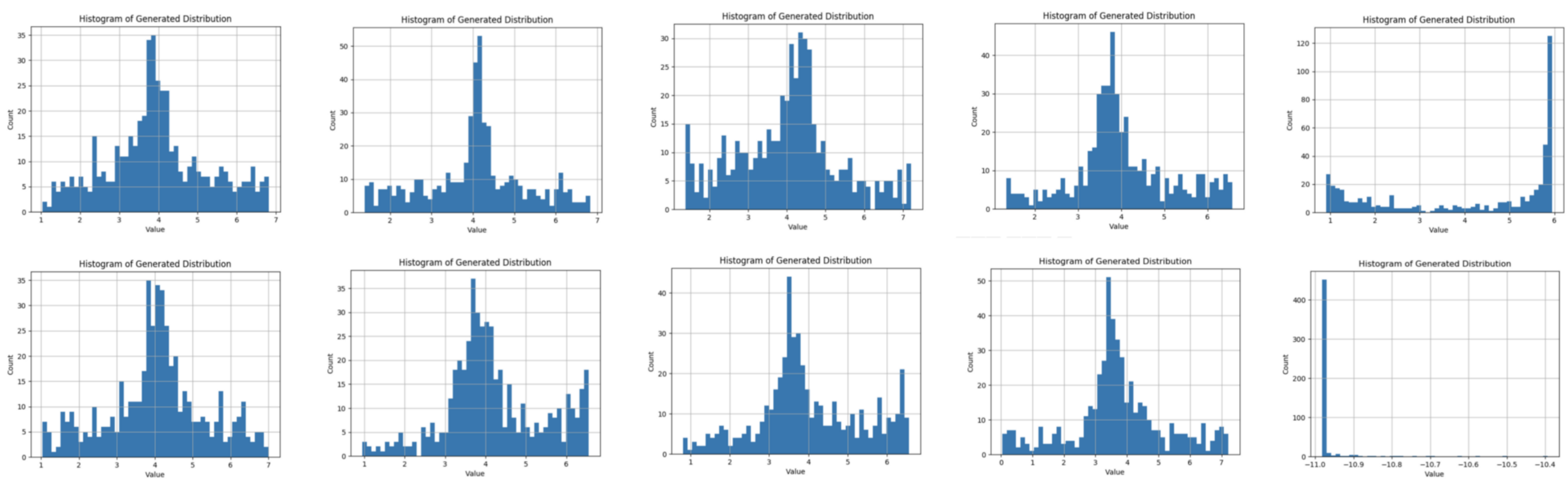
- One adversary of the GAN can overpower the other
  - Depends on learning rate, optimizers, loss functions, etc.
- If the discriminator is too good, it will return values close to 0 or 1
  - Generator will be unable to find a meaningful gradient
- If the generator is too good, it will exploit weaknesses in the discriminator
  - Simpler patterns than "authenticity" might fool it
    ... the surreal images demonstrate this

Pitfalls and Guidelines in Training GANs *... continued*

Randomized initial conditions make a big difference!

Additional training *might* get out of poor local maxima

Imbalance might occur where progress is impossible



Looking Forward and Guessing

- Fraud might be more effective by using GANs
  - “Deep fakes” of images or videos
  - Falsified financial transactions or sensitive communications
  - Scientific or medical research results manipulated
  - Biometric or genomic information simulated

Looking Forward and Guessing **...** *continued*

Discriminators might detect malfeasance better than do naive analyses

Inasmuch as “nature tries to fool us”, discriminators might lead to better science

In some domains, e.g. time-series and NLP, transformers seem on track to supplant recurrent networks

... but GANs might themselves utilize transformers

Simulating confidential data (e.g. medical research)

The Future History of Generative Adversarial Networks