Semi-supervised learning with GANs

Scaleway

Olga Petrova, ML Engineer

Semi-supervised learning with GANs

or

WHAT HAPPENS WHEN DOGS AND CATS ARE LEFT UNSUPERVISED

Scaleway

Olga Petrova, ML Engineer

Outline

1. **Theory**
   - A. Semi-supervised learning
   - B. Generative Adversarial Networks
   - C. Semi-supervised learning with GANs
2. **Practice**

A Tale of Cats and Dogs

Outline

1. **Theory**
    A. Semi-supervised learning
    B. Generative Adversarial Networks
    C. Semi-supervised learning with GANs
2. **Practice**

A Tale of Cats and Dogs

Semi-Supervised Learning: what

- Unsupervised learning (unlabelled training data): e.g. clustering, GAN
- Supervised learning (labelled training data): e.g. regression, classification
- Semi-supervised learning:

  - a subset of the training data is labeled
  - the rest is unlabelled
  - supervised learning tasks

Semi-Supervised Learning: why

- In practice, many tasks of interest come from <u>supervised learning</u>
- Deep learning: models <u>need a lot of training data</u>
- Labelling data is <u>expensive</u>
- <u>Unlabelled data</u>: easier to collect
- <u>Semi-supervised learning</u>: use unlabelled data to boost the performance of supervised ML models

Outline

1. **Theory**

   A. Semi-supervised learning

   B. Generative Adversarial Networks

   C. Semi-supervised learning with GANs

2. **Practice**

A Tale of Cats and Dogs

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- *Unsupervised learning*: unlabelled training data (e.g., MNIST images)
- *Goal*: generate data from the same distribution (e.g. new images that look like MNIST)

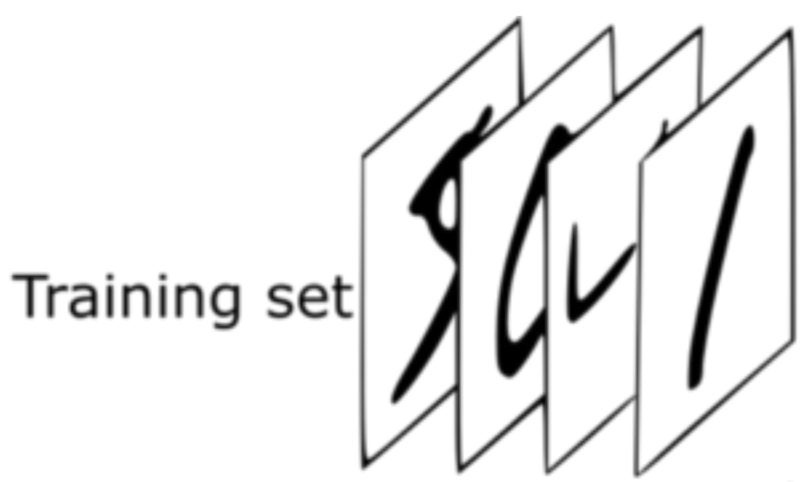


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- *Unsupervised learning*: unlabelled training data (e.g., MNIST images)
- *Goal*: generate data from the same distribution (e.g. new images that look like MNIST)
- **Generator**: deConv network that generates an image from random input noise

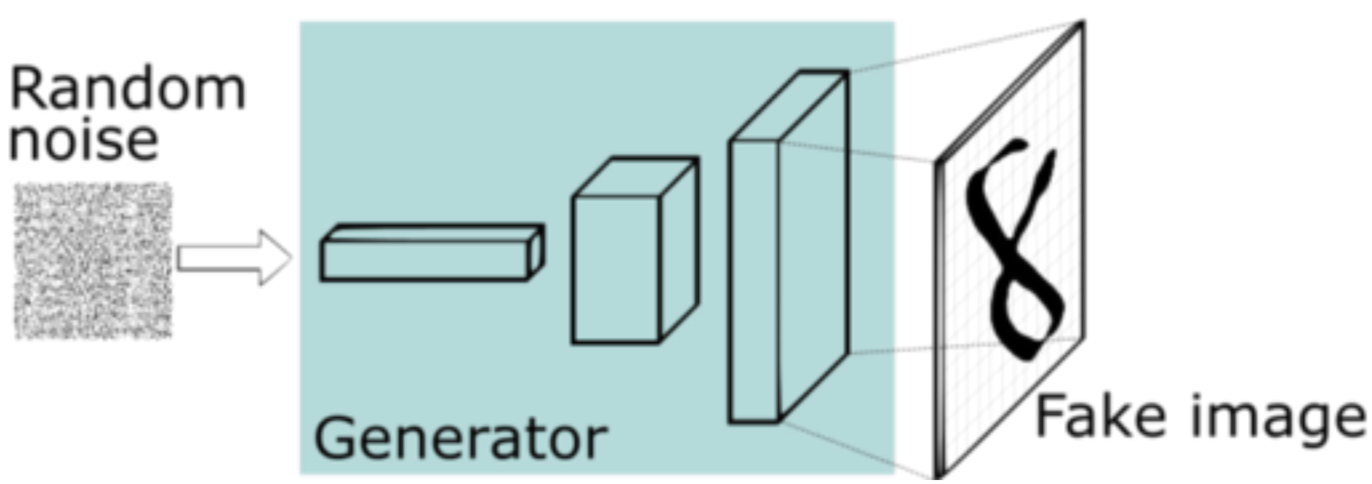


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- *Unsupervised learning*: unlabelled training data (e.g., MNIST images)
- *Goal*: generate data from the same distribution (e.g. new images that look like MNIST)
- **Generator**: deConv network that generates an image from random input noise
- **Discriminator**

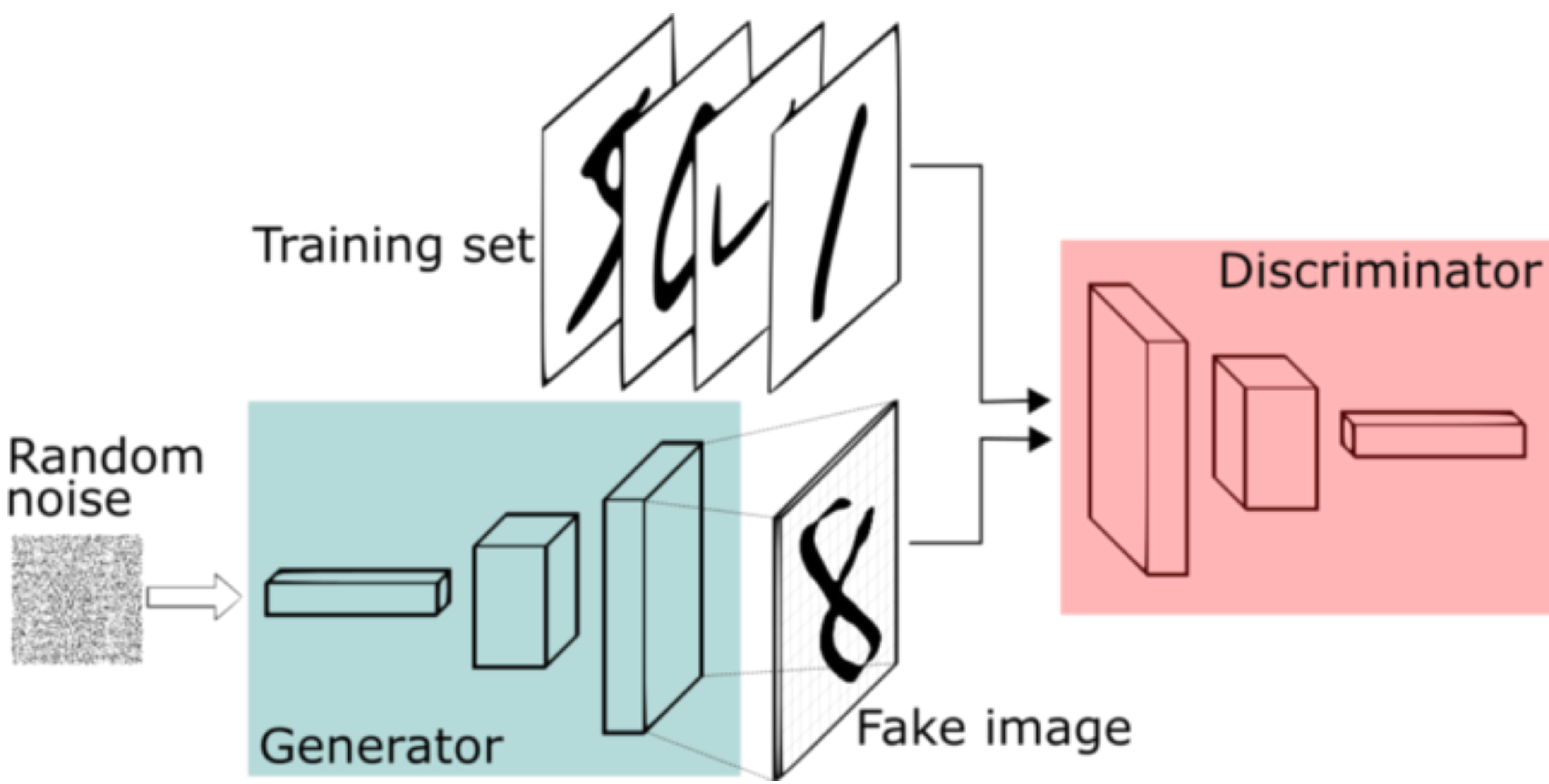


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- *Unsupervised learning*: unlabelled training data (e.g., MNIST images)
- *Goal*: generate data from the same distribution (e.g. new images that look like MNIST)
- **Generator**: deConv network that generates an image from random input noise
- **Discriminator**: binary classifier that is trained to accept Real images and reject Fake ones

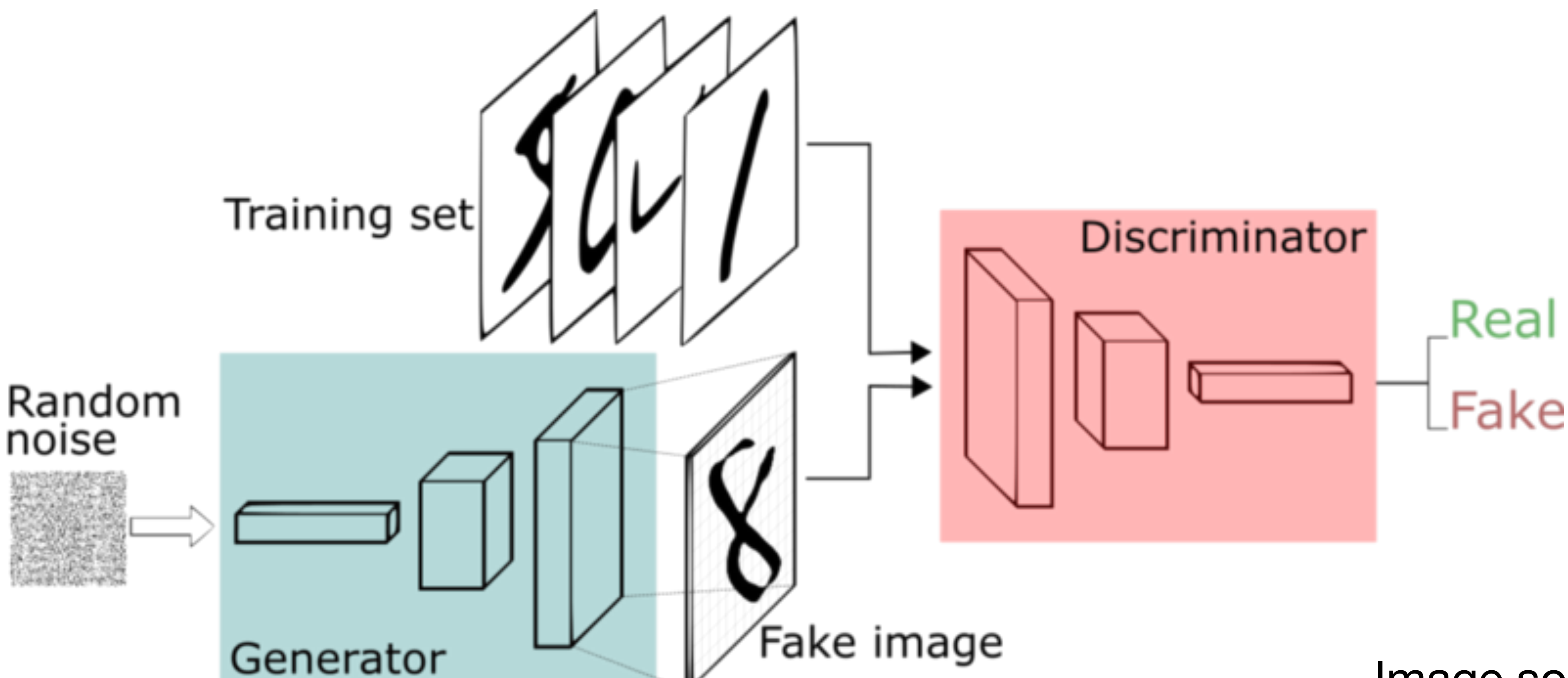


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

Training of the Discriminator

- Straightforward binary classification (*Real* for the training set, *Fake* for the generated images)

Training of the Generator

- We pass the generated image as input to the Discriminator
- If the image is classified as Fake, we adjust the Generator to produce better images

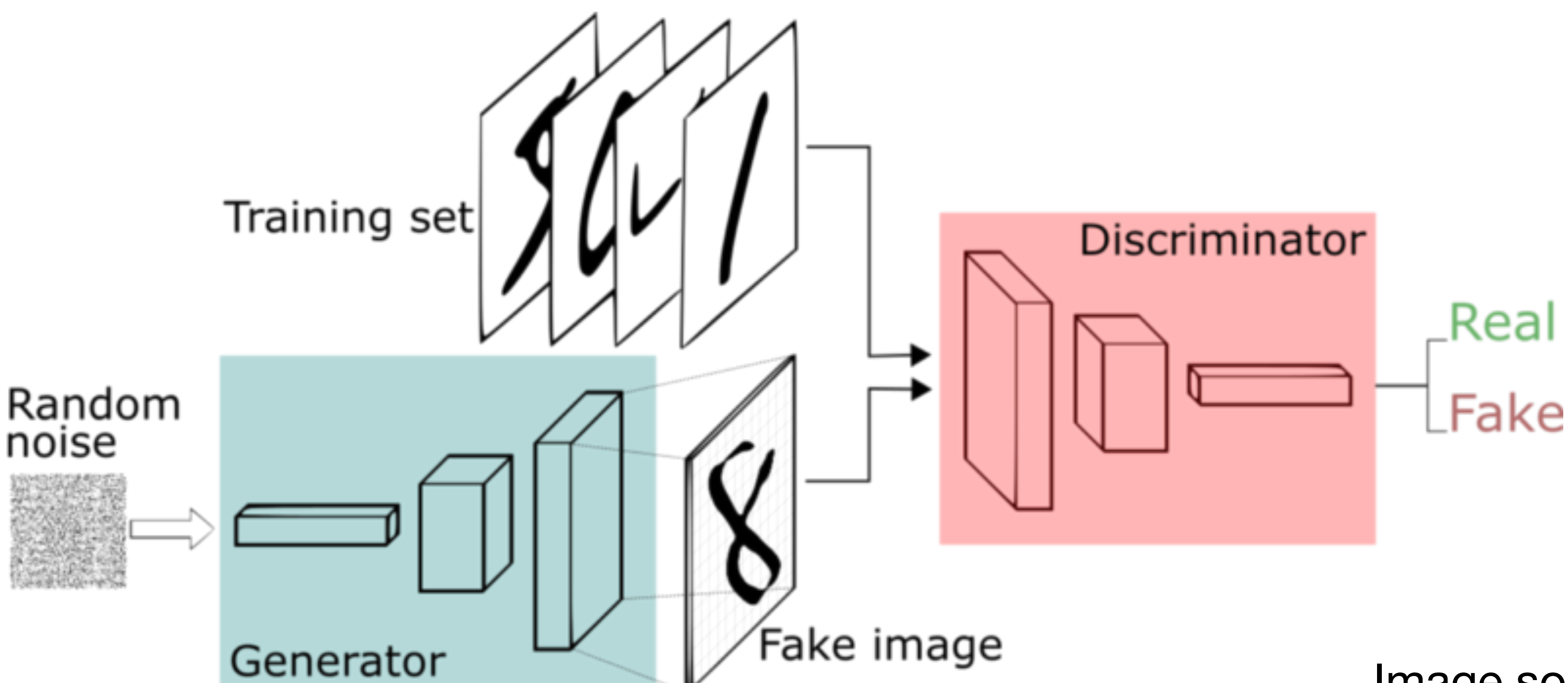


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- In practice: G and D are trained separately, one after another, for each batch:

```
        ## Train D with all-real batch
        netD.zero_grad()
        real = data # a batch of images from the training set
        label = torch.ones(real.size(0))

        output = netD(real).view(-1)
        errD_real = criterion(output, label) #criterion = nn.BCELoss()
        errD_real.backward()


        ## Train D with all-fake batch
        noise = torch.randn(real.size(0), nz, 1, 1) # (generate a batch of inputs z for netG)
        fake = netG(noise)
        label = torch.zeros(real.size(0))
        output = netD(fake.detach()).view(-1) #fake.DETACH() because we are not training netG here
        errD_fake = criterion(output, label)
        errD_fake.backward()

        # Update D
        optimizerD.step()

        ## Train G to fool D
        netG.zero_grad()
        label = torch.ones(real.size(0))
        output = netD(fake).view(-1) # Since D was just updated ;-)
        errG = criterion(output, label)
        errG.backward()

        # Update G
        optimizerG.step()
```

Pytorch GAN tutorial

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Real data: $x \sim p(x)$
- Generated data: $x \sim p_G(x)$ (Actually, $x \sim p_G(x|z)$)
- Goal: $p_G(x) \approx p(x)$
- Discriminator D is trained to assign the correct label (Real vs Fake)
- Generator G is trained to produce images that will fool the Discriminator D

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Outline

1. **Theory**
    A. Semi-supervised learning
    B. Generative Adversarial Networks
    C. Semi-supervised learning with GANs
2. **Practice**

A Tale of Cats and Dogs

Semi-supervised learning with GANs

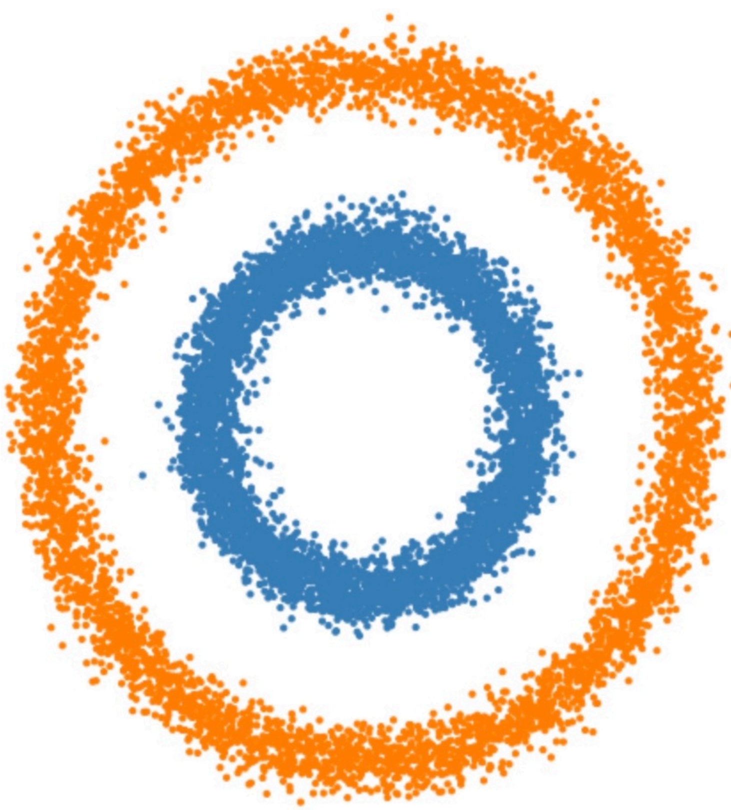
Ref: *Improved Techniques for Training GANs, 2016*

Basic idea:

- Our training data has a certain distribution
- The GAN will attempt to learn that distribution
- ≈ "clustering" + using the labeled samples to assign labels to classes

Example:

- 2D data: two circles = two classes, but only a few of the points are labeled



Semi-supervised learning with GANs

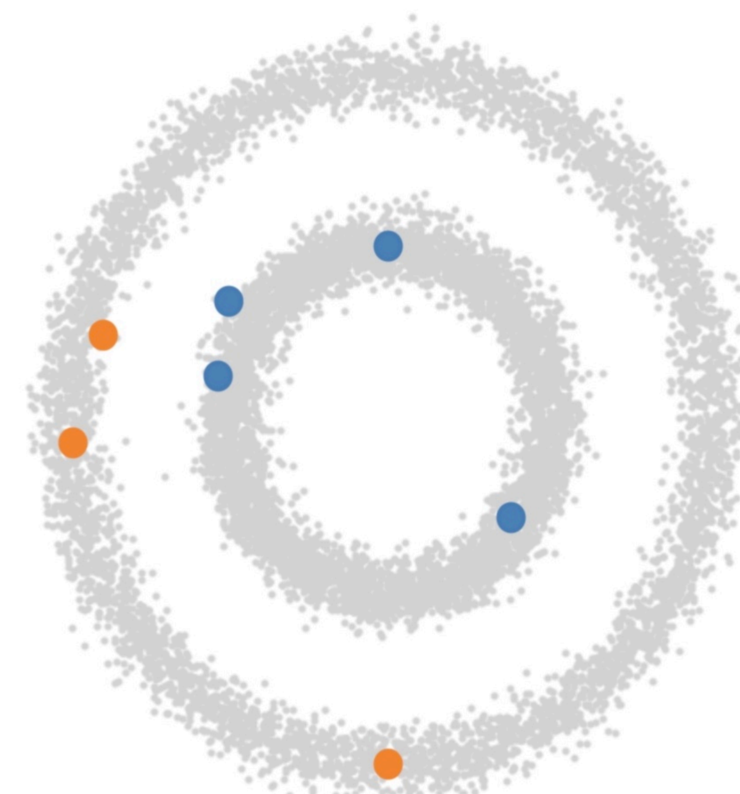
Ref: *Improved Techniques for Training GANs, 2016*

Basic idea:

- Our training data has a certain distribution
- The GAN will attempt to learn that distribution
- ≈ “clustering” + using the labeled samples to assign labels to classes

Example:

- 2D data: two circles = two classes, but only a few of the points are labeled



Semi-supervised learning with GANs

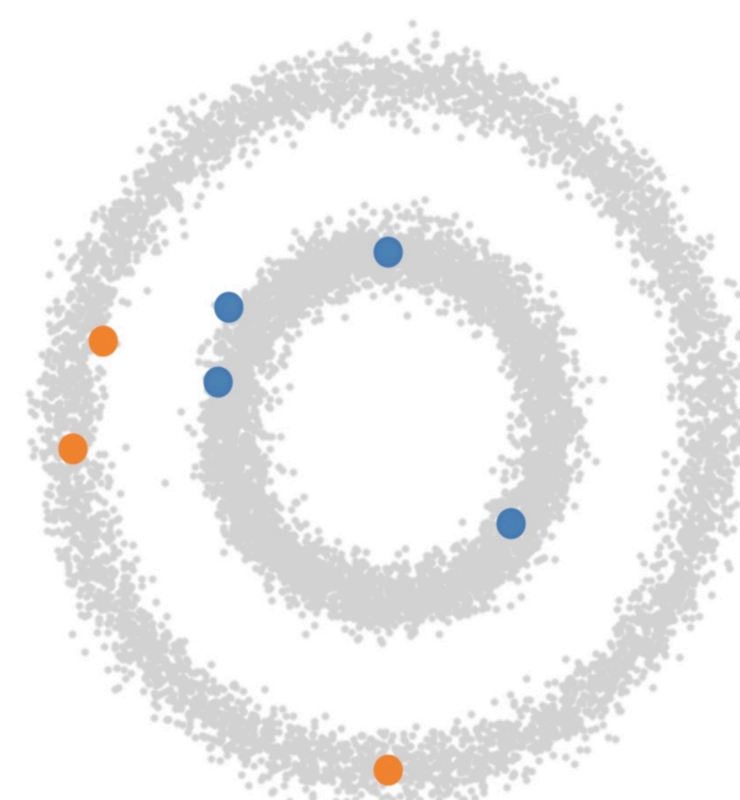
Ref: *Improved Techniques for Training GANs, 2016*

Basic idea:

- Our training data has a certain distribution
- The GAN will attempt to learn that distribution
- ≈ "clustering" + using the labeled samples to assign labels to classes

Example:

- 2D data: two circles = two classes, but only a few of the points are labeled
- GAN will learn that there are two circles



Semi-supervised learning with GANs

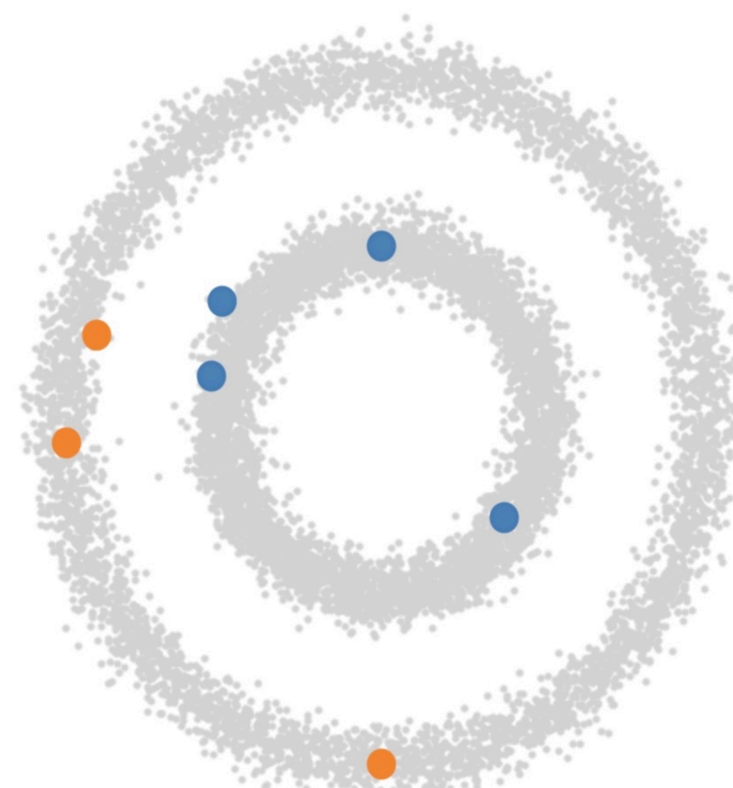
Ref: *Improved Techniques for Training GANs, 2016*

Basic idea:

- Our training data has a certain distribution
- The GAN will attempt to learn that distribution
- ≈ “clustering” + using the labeled samples to assign labels to classes

Example:

- 2D data: two circles = two classes, but only a few of the points are labeled
- GAN will learn that there are two circles and the labeled points will help classify them as two classes



Semi-supervised learning with GANs

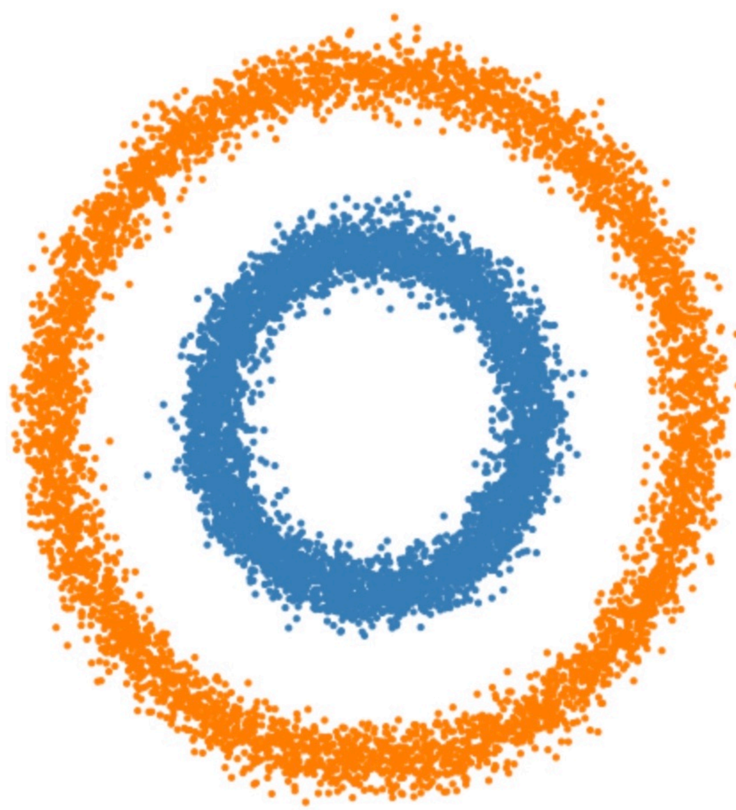
Ref: *Improved Techniques for Training GANs, 2016*

Basic idea:

- Our training data has a certain distribution
- The GAN will attempt to learn that distribution
- ≈ “clustering” + using the labeled samples to assign labels to classes

Example:

- 2D data: two circles = two classes, but only a few of the points are labeled
- GAN will learn that there are two circles and the labeled points will help classify them as two classes



SSL with GANs: how?

Ref: *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

- Unsupervised GAN: **Generator** + **Discriminator** (*Real* vs. *Fake*)
- Supervised classifier: *K classes*
- Semi-supervised: Discriminator = Classifier into *K+1 classes*
- The first *K classes* are *Real*; the added *(K+1) class* is *Fake*
- Semi-supervised GAN: **Generator** + the new **Discriminator/Classifier**

Training

- **Discriminator** is trained to assign:
  - labeled data to correct classes
  - unlabelled data to one of K classes
  - generated data to (K+1) class
- **Generator** is trained to:
  - produce images that the Discriminator would assign to one of K classes

How good is the resulting Generator? 🤔

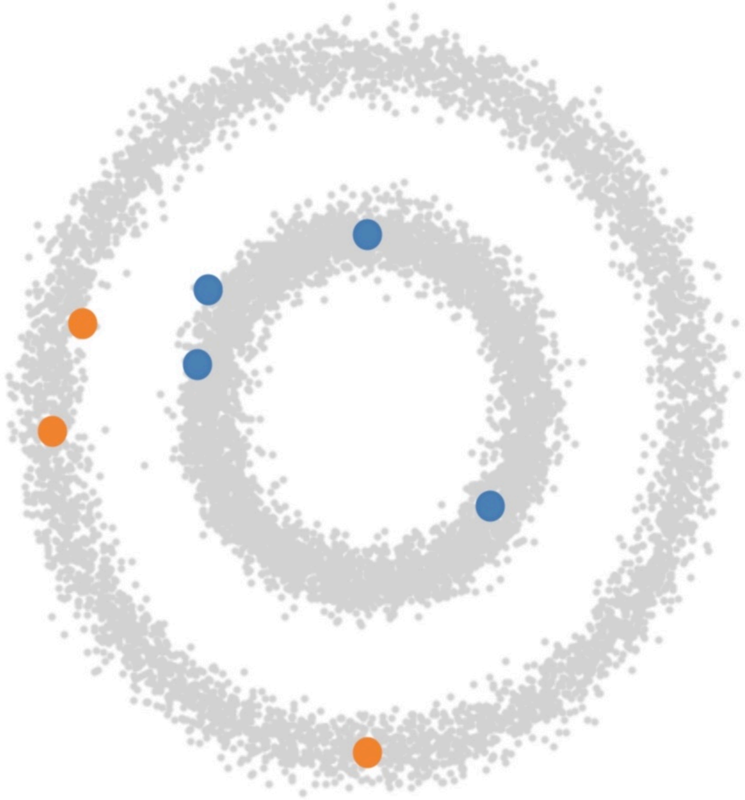
Ref: *Good Semi-supervised Learning that Requires a Bad GAN, 2017*

- State-of-the-art GANs achieve very impressive results (e.g., www.thispersondoesnotexist.com)

- However, the GANs used for semi-supervised learning **do not**

- To get a good classifier, the Generator should be “bad” = **complement**

  (generates complement samples in feature space)



- Generating more points for the two circles would not help us

- Generating *Fake* points between circles

  → better decision boundary between classes

How good is the resulting Generator? 🤔

Ref: *Good Semi-supervised Learning that Requires a Bad GAN, 2017*

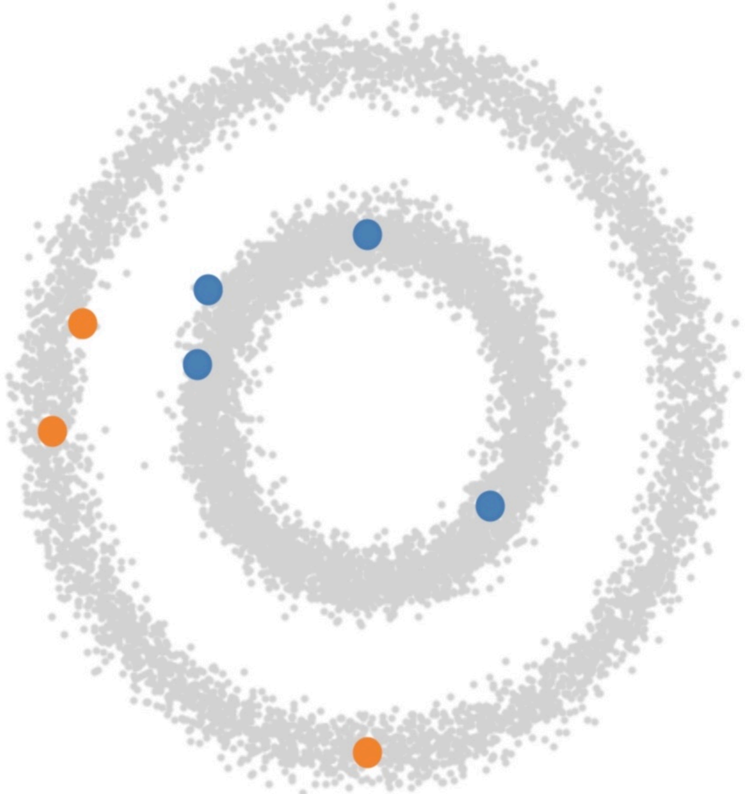
- State-of-the-art GANs achieve very impressive results (e.g., www.thispersondoesnotexist.com)

- However, the … ning **do not**
- To get a good … "bad" = **complement**

  (generates co…

- Generating m… not help us
- Generating *Fa*…

  → better decis…



How good is the resulting Generator? 🤔

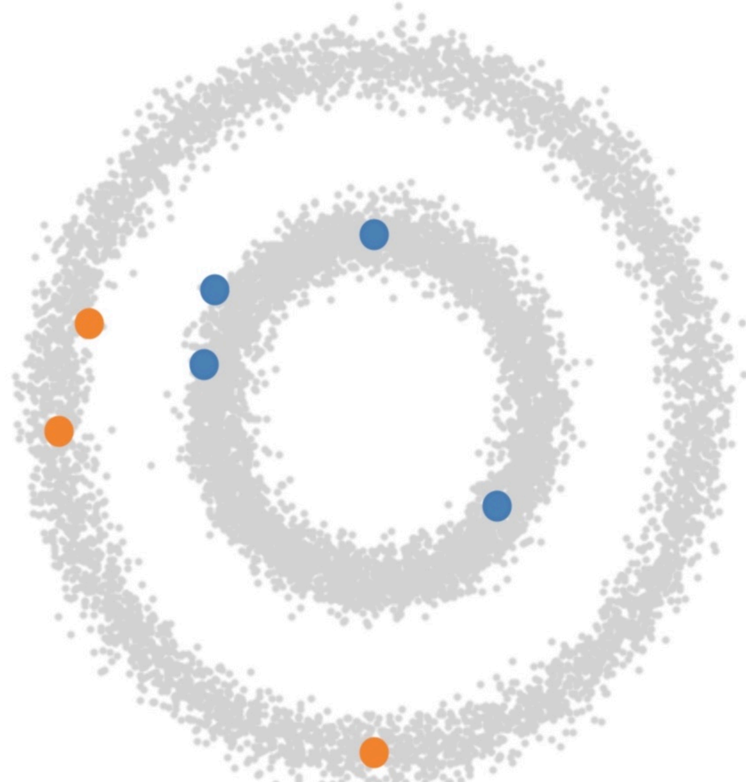
Ref: *Good Semi-supervised Learning that Requires a Bad GAN, 2017*

- State-of-the-art GANs achieve very impressive results (e.g., www.thispersondoesnotexist.com)

- However, the GANs used for semi-supervised learning **do not**

- To get a good classifier, the Generator should be "bad" = **complement**

  (generates complement samples in feature space)



- Generating more points for the two circles would not help us

- Generating *Fake* points between circles

  → better decision boundary between classes

How good is the resulting Generator? 🤔

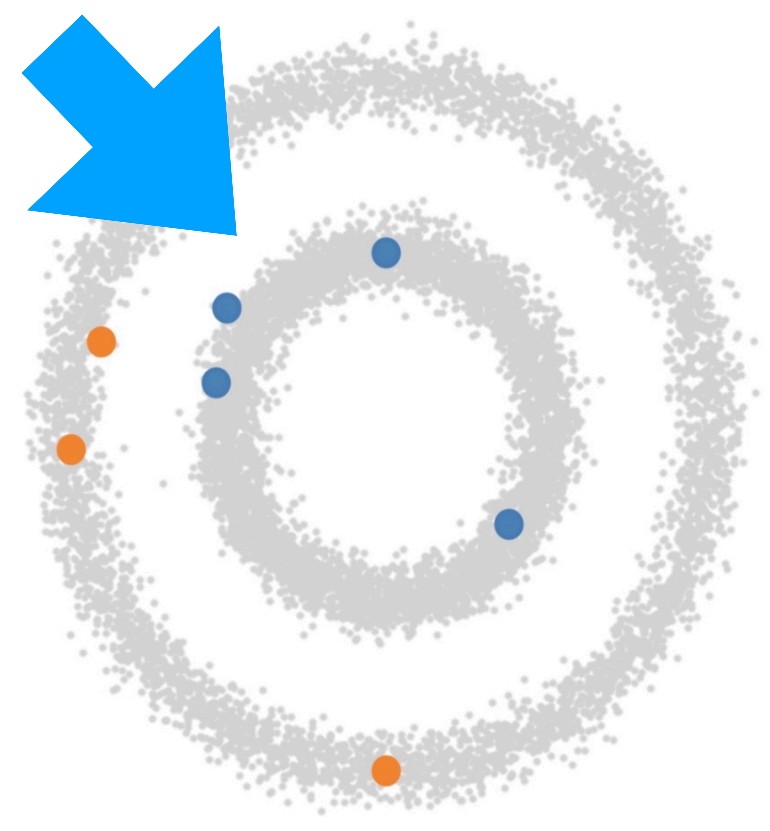
Ref: *Good Semi-supervised Learning that Requires a Bad GAN, 2017*

- State-of-the-art GANs achieve very impressive results (e.g., www.thispersondoesnotexist.com)

- However, the GANs used for semi-supervised learning **do not**

- To get a good classifier, the Generator should be “bad” = **complement**

  (generates complement samples in feature space)



- Generating more points for the two circles would not help us

- Generating *Fake* points between circles

  → better decision boundary between classes

What is next?

- Simple Generator + Classifier GAN has a problem:

  Classifier has two conflicting goals, classifying *Real* and rejecting *Fake*

- Solution: **Triple GAN** = Generator + Discriminator + Classifier

  Ref: *Triple Generative Adversarial Nets, 2017*

- The Good, the Bad, and the GAN:

  get both a **good Classifier** and a **good Generator** (UGAN)

  Ref: *Semi-supervised Learning using Adversarial Training with Good and Bad Samples, 2019*

Results

Table 2: Test accuracy on semi-supervised MNIST. Results are averaged over 10 runs. $*$ denotes hand selection of labeled data. † denotes our implementation of the model.

Model	Test accuracy for a given number of labeled samples			
	20	50	100	200
FM-GAN [32]	$83.23 \pm 4.52\%$	$97.79 \pm 1.36\%$	$99.07 \pm 0.07\%$	$99.10 \pm 0.04\%$
Bad GAN [5]	-	-	$99.21 \pm 0.10\%$	-
Triple-GAN [4]	$95.19 \pm 4.95\%$	$98.44 \pm 0.72\%$	$99.09 \pm 0.58\%$	$99.33 \pm 0.16\%$
Bad GAN†	$88.38 \pm 3.08\%^*$	$96.24 \pm 0.16\%$	$99.17 \pm 0.03\%$	$99.20 \pm 0.03\%$
Triple-GAN†	$95.93 \pm 4.45\%^*$	$98.68 \pm 1.12\%$	$99.07 \pm 0.46\%$	$99.17 \pm 0.08\%$
UGAN	$\mathbf{97.34 \pm 6.86\%}^*$	$\mathbf{98.92 \pm 0.13\%}$	$\mathbf{99.21 \pm 0.08\%}$	$\mathbf{99.35 \pm 0.05\%}$

Ref: *Semi-supervised Learning using Adversarial Training with Good and Bad Samples, 2019*

Outline

1. **Theory**
    A. Semi-supervised learning
    B. Generative Adversarial Networks
    C. Semi-supervised learning with GANs
2. **Practice**

    A Tale of Cats and Dogs

Outline

1. **Theory**
   A. Semi-supervised learning
   B. Generative Adversarial Networks
   C. Semi-supervised learning with GANs
2. **Practice**

A Tale of Cats and Dogs

Moral of the story: read the fine print

- Let’s implement a simple SSL GAN: Generator + Classifier
- Dog vs. Cat binary classification

Moral of the story: read the fine print

- Let's implement a simple SSL GAN: Generator + Classifier
- Dog vs. Cat binary classification
- Option I:

- add a third (Fake) class; increase dim_out of Classifier by one

- Unlabelled data: D wants to minimize probability of 3rd class

- Generated data: D wants to minimize probability of 1 and 2 classes,
G wants to minimize probability of 3rd class

Moral of the story: read the fine print

- Let’s implement a simple SSL GAN: Generator + Classifier
- Dog vs. Cat binary classification
- Option I:

~~- add a third (Fake) class; increase dim. of the classifier by one~~

~~- Unlabelled data: D wants to minimize probability of 3rd class~~

~~- Generated data: D wants to minimize probability of 1 and 2 classes,~~
~~G wants to minimize probability of 3rd class~~

Moral of the story: read the fine print

- Let's implement a simple SSL GAN: Generator + Classifier

- Dog vs. Cat binary classification

- Option I:

~~- add a third (Fake) class; increase dim_out of Classifier by one~~

~~- Unlabelled data: D wants to minimize probability of 3rd class~~

~~- Generated data: D wants to minimize probability of 1 and 2 classes,~~
~~G wants to minimize probability of 3rd class~~

- Option II:

- do not increase dim_out of Classifier

- output a strong class prediction for Real examples, and small class predictions for Fake examples

$$D(\boldsymbol{x}) = \frac{Z(\boldsymbol{x})}{Z(\boldsymbol{x})+1}, \text{ where } Z(\boldsymbol{x}) = \sum_{k=1}^{K} \exp[l_k(\boldsymbol{x})]$$

Ref: *Improved Techniques for Training GANs, 2016*, *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016* https://github.com/opetrova/SemiSupervisedPytorchGAN