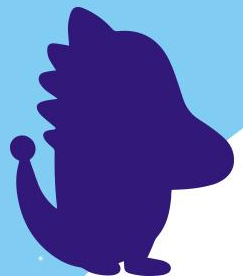
DATAXDAYS

Semi-supervised learning with GANs

Olga Petrova
Machine Learning Engineer @ Scaleway



publicis sapient BY ENGINEERS at PUBLICIS SAPIENT

OUTLINE

THEORY :

- Semi-Supervised Machine Learning
- Generative Adversarial Networks (GANs)
- Semi-supervised Learning with GANs

PRACTICE :

- A Tale of Cats & Dogs

THEORY

Semi-Supervised Machine Learning

WHAT IS SEMI-SUPERVISED LEARNING?

UNSUPERVISED LEARNING (unlabeled training data): e.g. clustering, GANs

SUPERVISED LEARNING (labeled training data): e.g. regression, classification

SEMI-SUPERVISED LEARNING:

- a subset of the training data is labeled
- the rest is unlabeled
- supervised learning task (e.g. classification)

WHAT IS SEMI-SUPERVISED LEARNING?

UNSUPERVISED LEARNING (unlabeled training data): e.g. clustering, GANs

SUPERVISED LEARNING (labeled training data): e.g. regression, classification

SEMI-SUPERVISED LEARNING:

- a subset of the training data is labeled
- the rest is unlabeled
- supervised learning task (e.g. classification)

WHAT IS SEMI-SUPERVISED LEARNING?

UNSUPERVISED LEARNING (unlabeled training data): e.g. clustering, GANs

SUPERVISED LEARNING (labeled training data): e.g. regression, classification

SEMI-SUPERVISED LEARNING:

- a subset of the training data is labeled
- the rest is unlabeled
- supervised learning task (e.g. classification)

WHAT IS SEMI-SUPERVISED LEARNING?

UNSUPERVISED LEARNING (unlabeled training data): e.g. clustering, GANs

SUPERVISED LEARNING (labeled training data): e.g. regression, classification

SEMI-SUPERVISED LEARNING:

- a subset of the training data is labeled
- the rest is unlabeled
- supervised learning task (e.g. classification)

WHY DO SEMI-SUPERVISED LEARNING?

- Most practical tasks: supervised machine learning
  - Good news: today's supervised models are very powerful
  - But: often comes at the expense of very large/deep networks
  - Bad news: the larger the model, the more data it needs to train!
  - Really bad news: labelling data is expensive
- Unlabeled data to the rescue
- Semi-supervised learning: use unlabelled data to boost the performance of supervised ML models

WHY DO SEMI-SUPERVISED LEARNING?

- Most practical tasks: supervised machine learning
  - Good news: today's supervised models are very powerful
  - But: often comes at the expense of very large/deep networks
  - Bad news: the larger the model, the more data it needs to train!
  - Really bad news: labelling data is expensive
- Unlabeled data to the rescue
- Semi-supervised learning: use unlabelled data to boost the performance of supervised ML models

WHY DO SEMI-SUPERVISED LEARNING?

- Most practical tasks: supervised machine learning
- 😸 Good news: today's supervised models are very powerful
- But: often comes at the expense of very large/deep networks
- Bad news: the larger the model, the more data it needs to train!
- Really bad news: labelling data is expensive
- Unlabeled data to the rescue
- Semi-supervised learning: use unlabelled data to boost the performance of supervised ML models

WHY DO SEMI-SUPERVISED LEARNING?

- Most practical tasks: supervised machine learning
- 😸 Good news: today's supervised models are very powerful
- 😸 But: often comes at the expense of very large/deep networks
- Bad news: the larger the model, the more data it needs to train!
- Really bad news: labelling data is expensive
- Unlabeled data to the rescue
- Semi-supervised learning: use unlabelled data to boost the performance of supervised ML models

WHY DO SEMI-SUPERVISED LEARNING?

- Most practical tasks: supervised machine learning
- 😸 Good news: today's supervised models are very powerful
- 😾 But: often comes at the expense of very large/deep networks
- 😿 Bad news: the larger the model, the more data it needs to train!
- Really bad news: labelling data is expensive
- Unlabeled data to the rescue
- Semi-supervised learning: use unlabelled data to boost the performance of supervised ML models

WHY DO SEMI-SUPERVISED LEARNING?

- Most practical tasks: **supervised machine learning**
- 😸 Good news: today's supervised models are **very powerful**
- 😾 But: often comes at the expense of **very large/deep networks**
- 😭 Bad news: the larger the model, **the more data it needs** to train!
- 🙀 Really bad news: **labelling data is expensive**
- Unlabeled data to the rescue
- Semi-supervised learning: use unlabelled data to boost the performance of supervised ML models

WHY DO SEMI-SUPERVISED LEARNING?

- Most practical tasks: **supervised machine learning**
- 😸 Good news: today's supervised models are **very powerful**
- 😿 But: often comes at the expense of **very large/deep networks**
- 😹 Bad news: the larger the model, **the more data it needs** to train!
- 😾 Really bad news: **labelling data is expensive** especially in highly specialized domains e.g. medicine
- Unlabeled data to the rescue
- Semi-supervised learning: use unlabelled data to boost the performance of supervised ML models

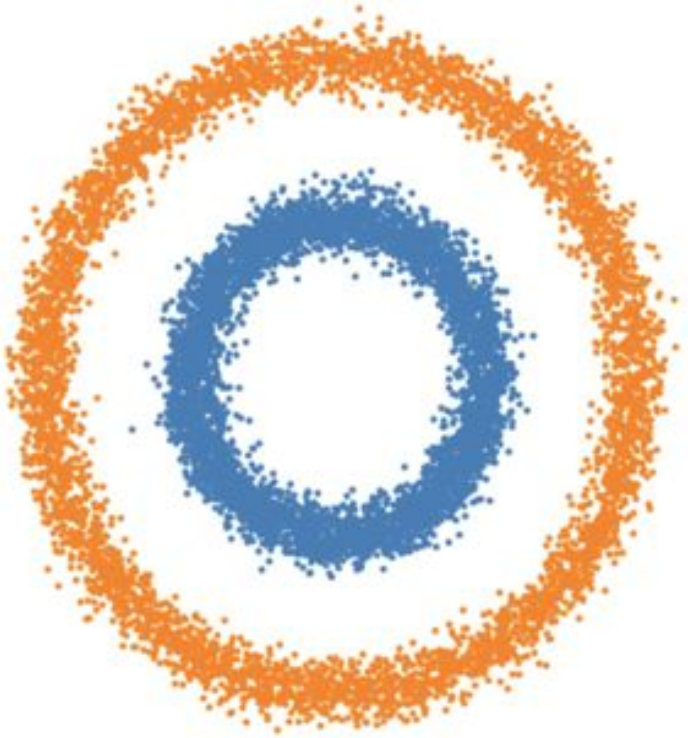
WHY DO SEMI-SUPERVISED LEARNING?

- Most practical tasks: **supervised machine learning**
- 😸 Good news: today's supervised models are **very powerful**
- 😾 But: often comes at the expense of **very large/deep networks**
- 😭 Bad news: the larger the model, **the more data it needs** to train!
- 🙀 Really bad news: **labelling data is expensive**
- **Unlabeled data** to the rescue
- Semi-supervised learning: use unlabelled data to boost the performance of supervised ML models

WHY DO SEMI-SUPERVISED LEARNING?

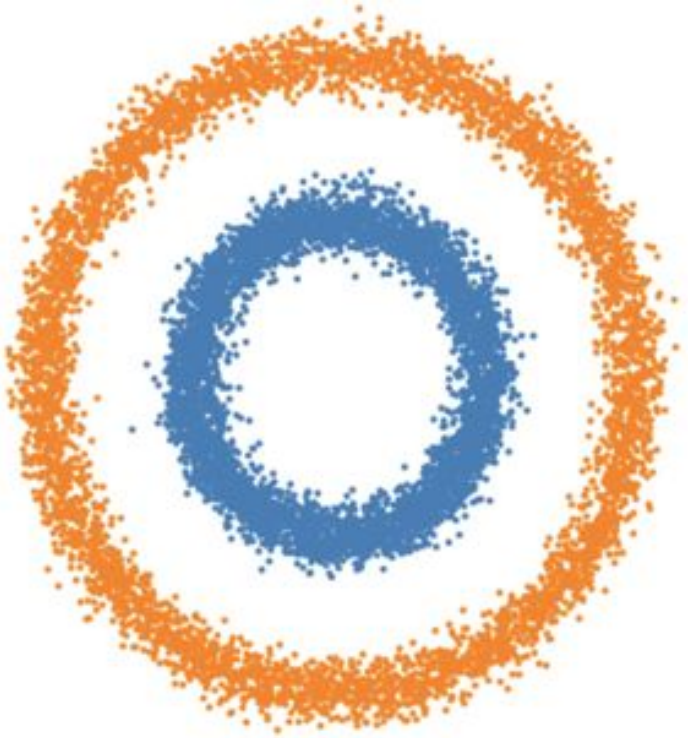
- Most practical tasks: supervised machine learning
- 😸 Good news: today's supervised models are very powerful
- 😿 But: often comes at the expense of very large/deep networks
- 😭 Bad news: the larger the model, the more data it needs to train!
- 🙀 Really bad news: labelling data is expensive
- Unlabeled data to the rescue
- Semi-supervised learning: use unlabelled data to boost the performance of supervised ML models

HOW DOES SSL WORK?



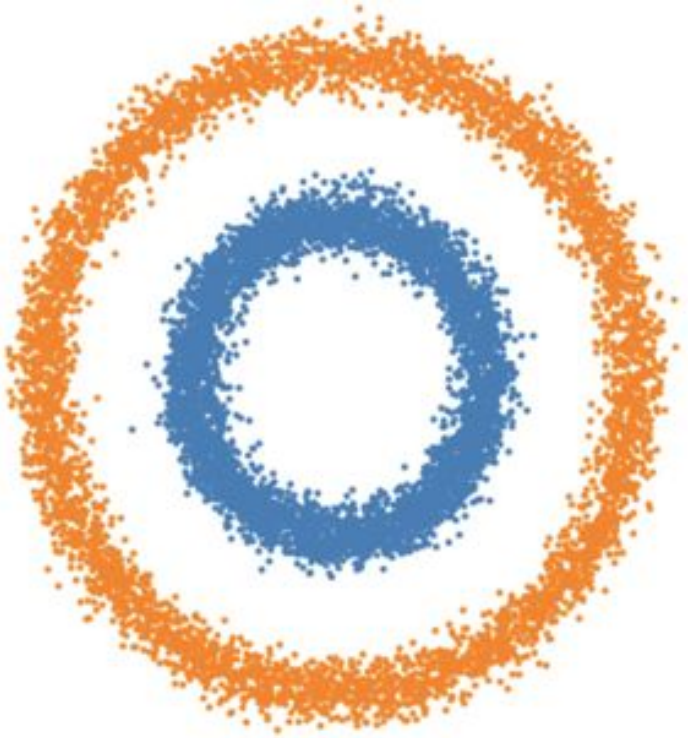
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



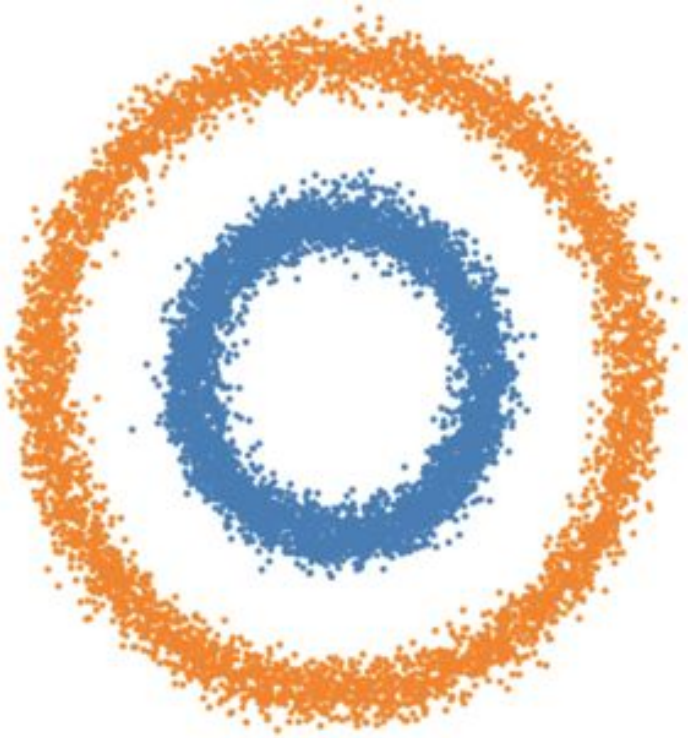
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



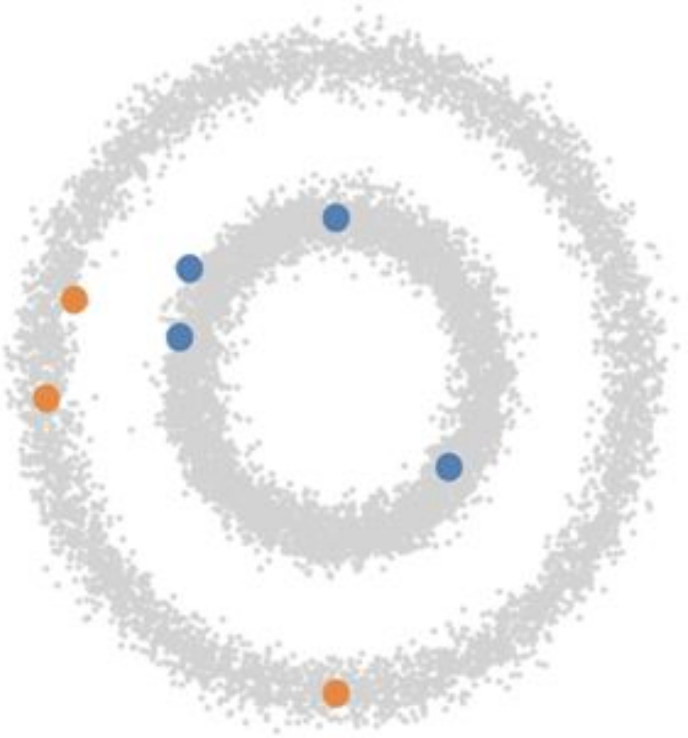
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



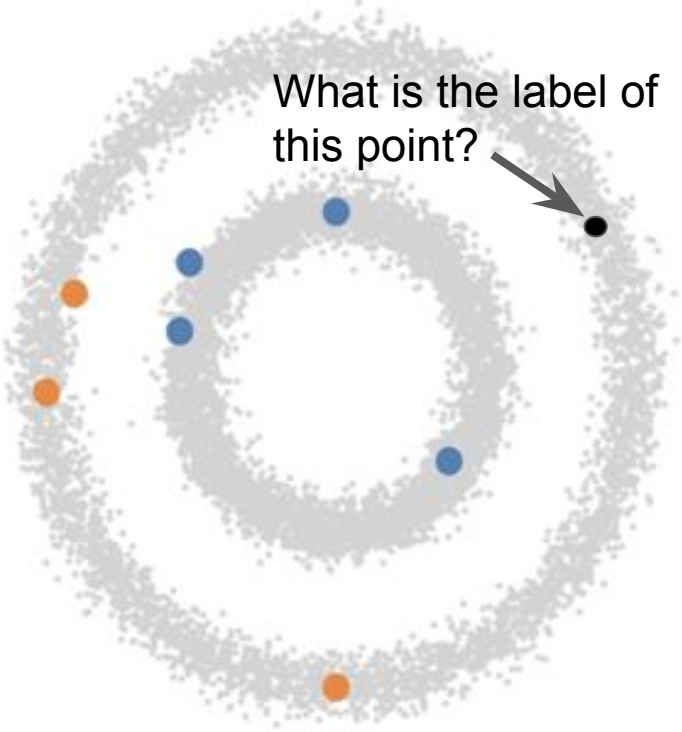
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



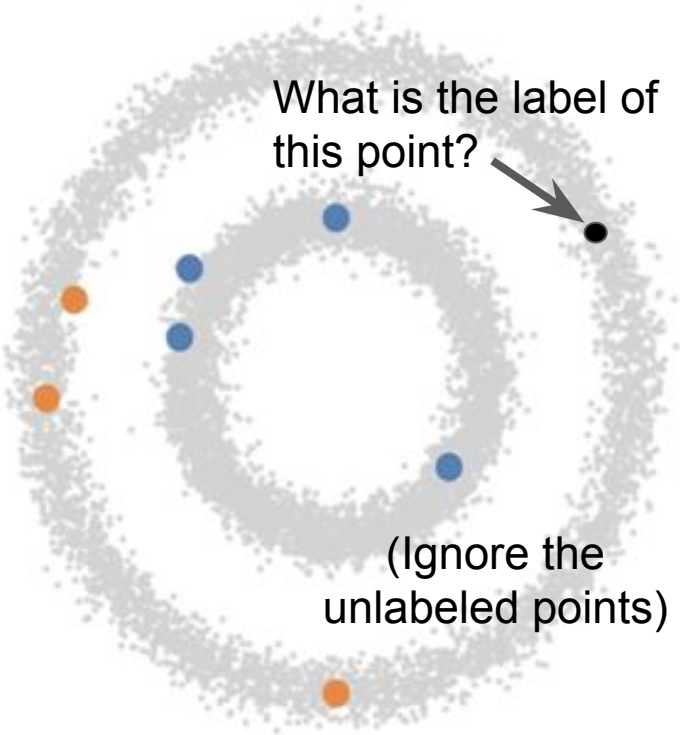
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



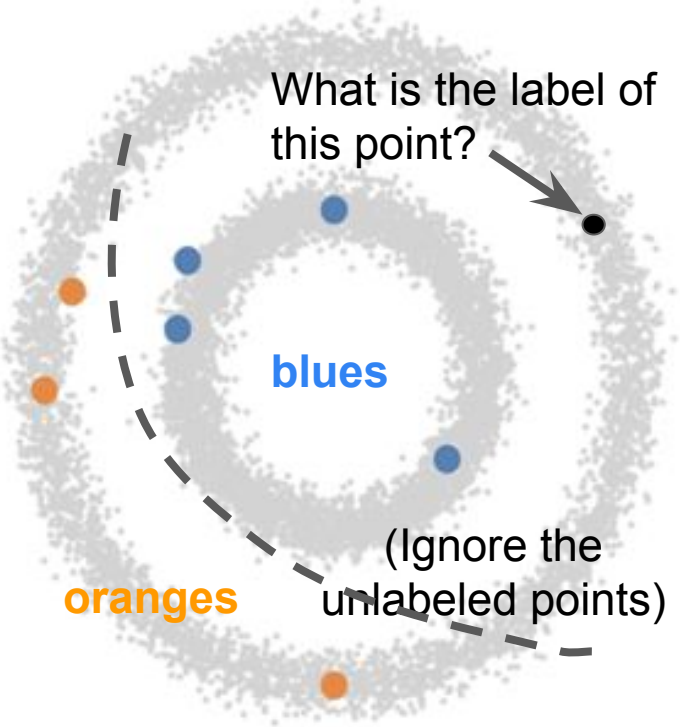
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



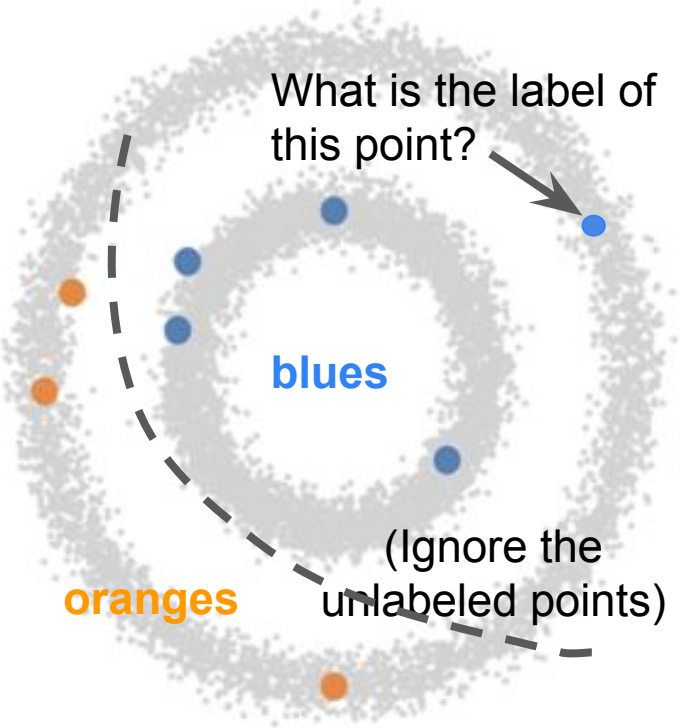
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



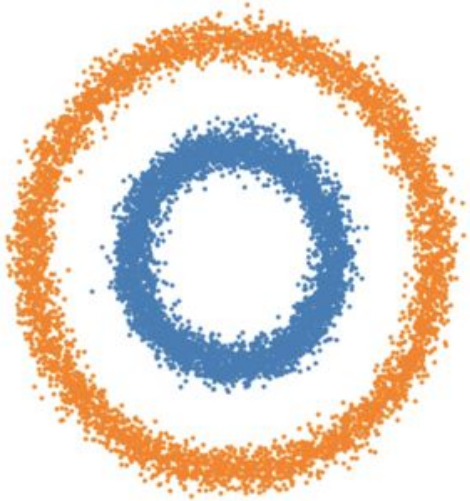
HOW DOES SSL WORK?

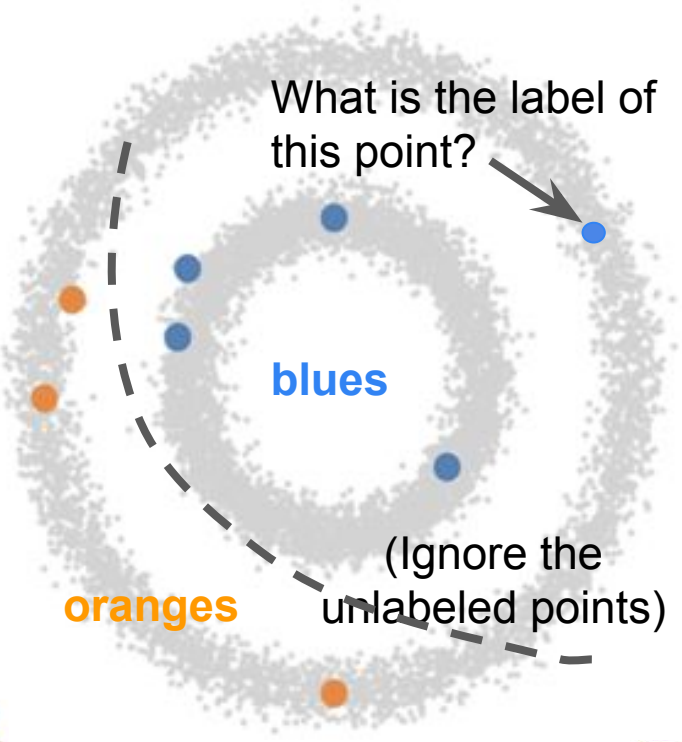
- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



HOW DOES SSL WORK?

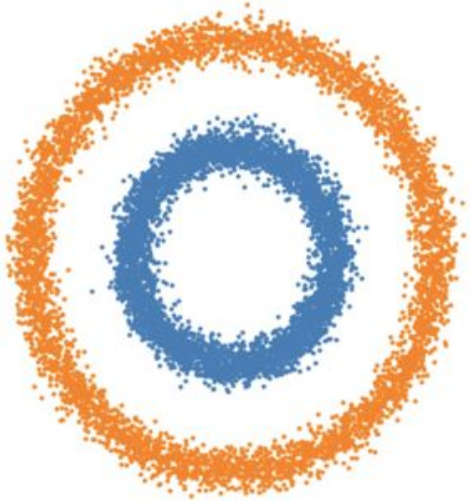
- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is
  the data forms two rings
- With labeled points, we ca
  labels to the rings

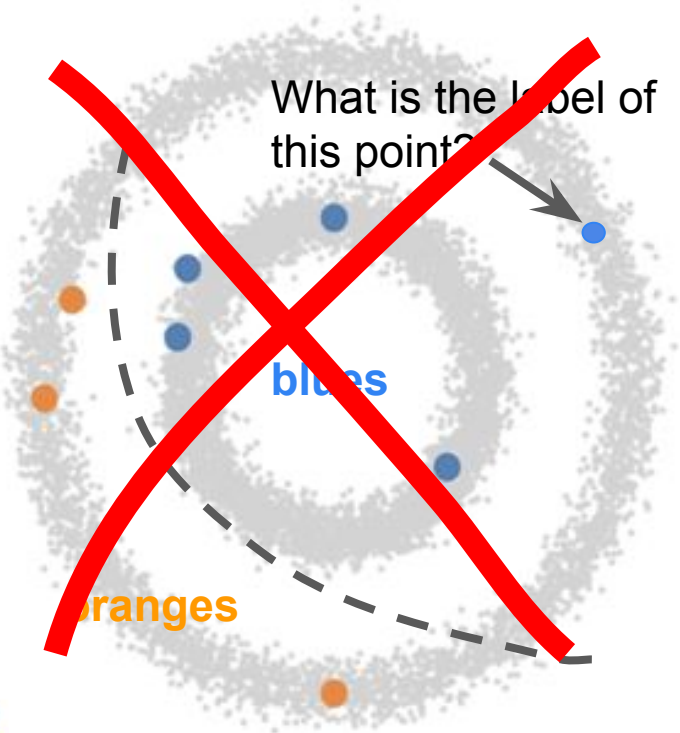




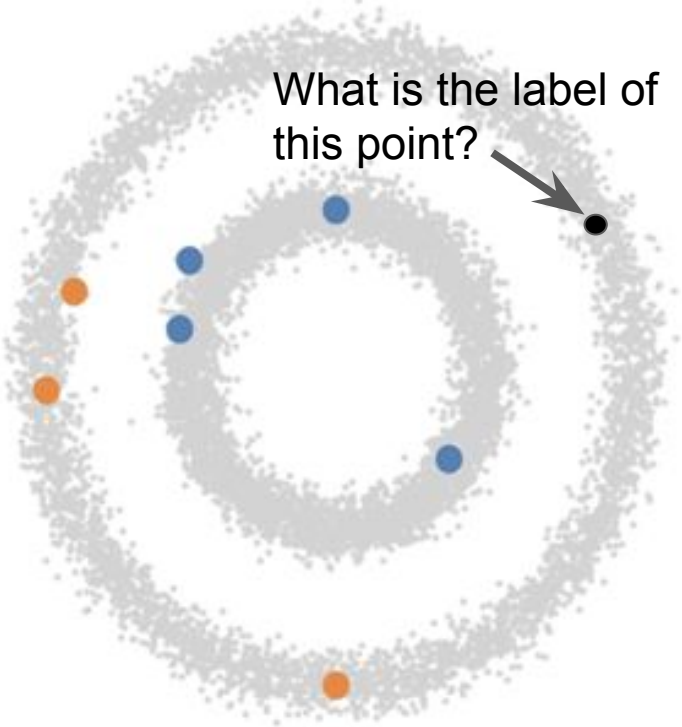
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is
  the data forms two rings
- With labeled points, we ca
  labels to the rings





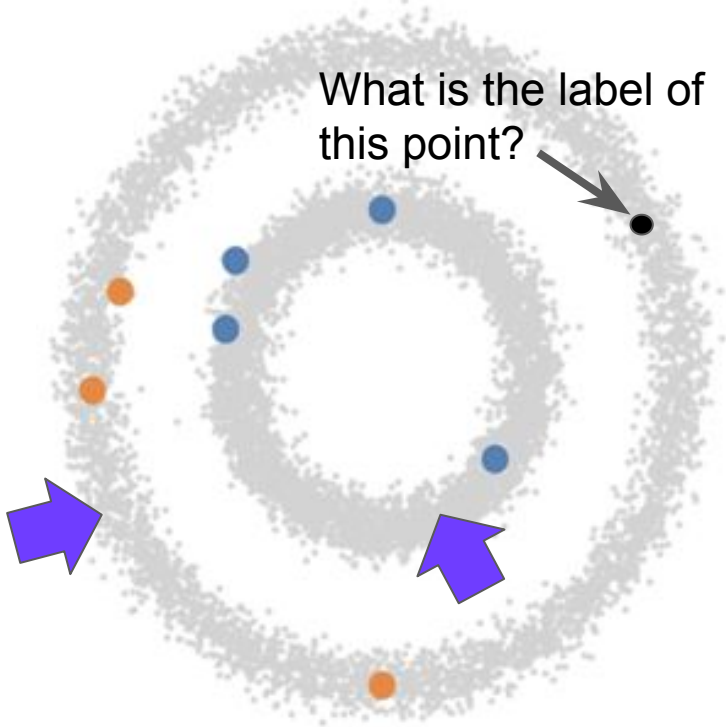
HOW DOES SSL WORK?



- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings

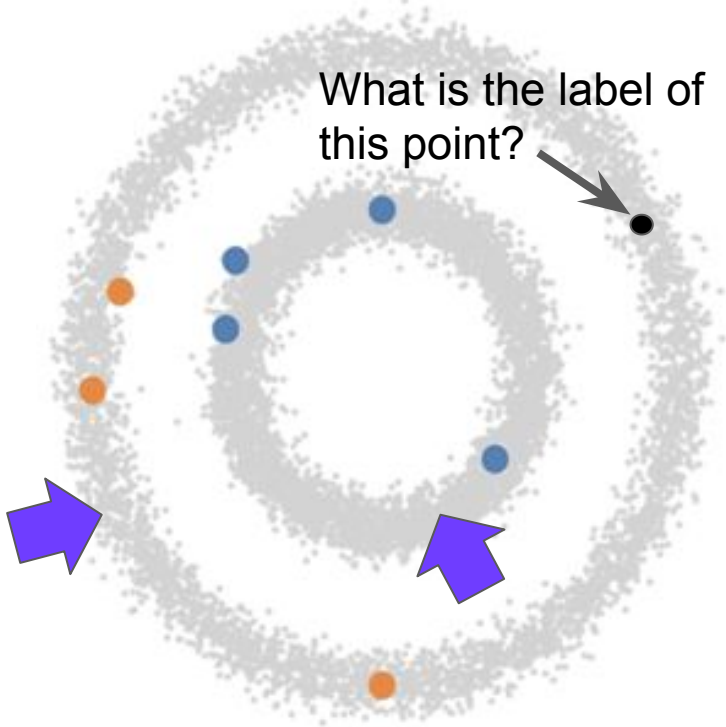
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



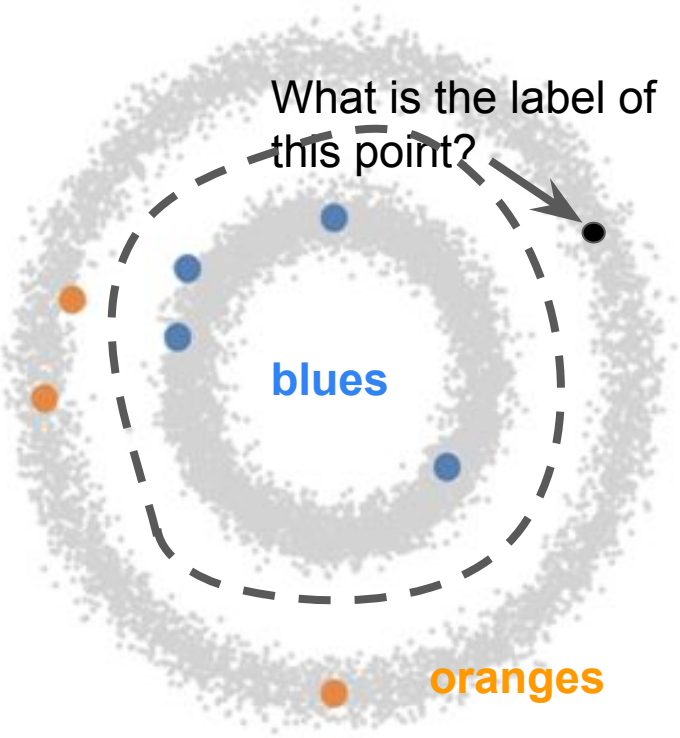
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



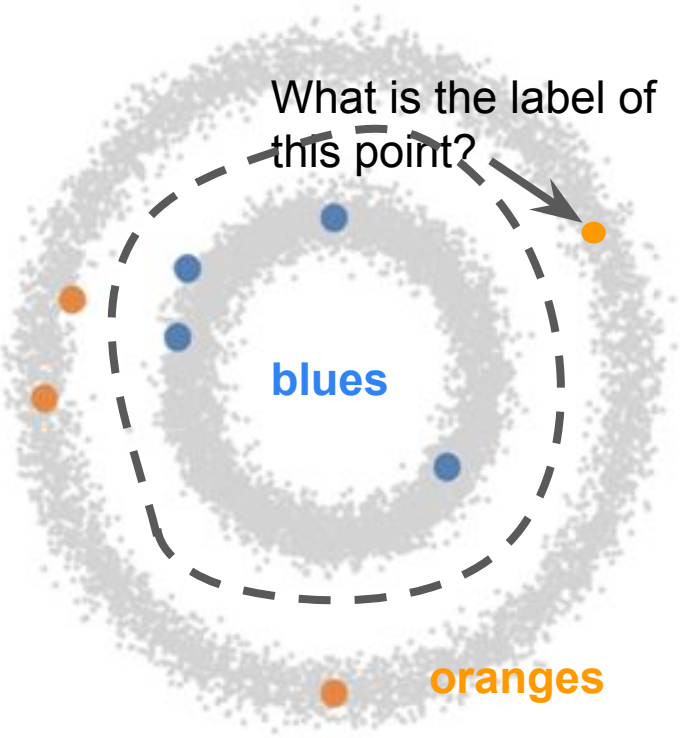
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



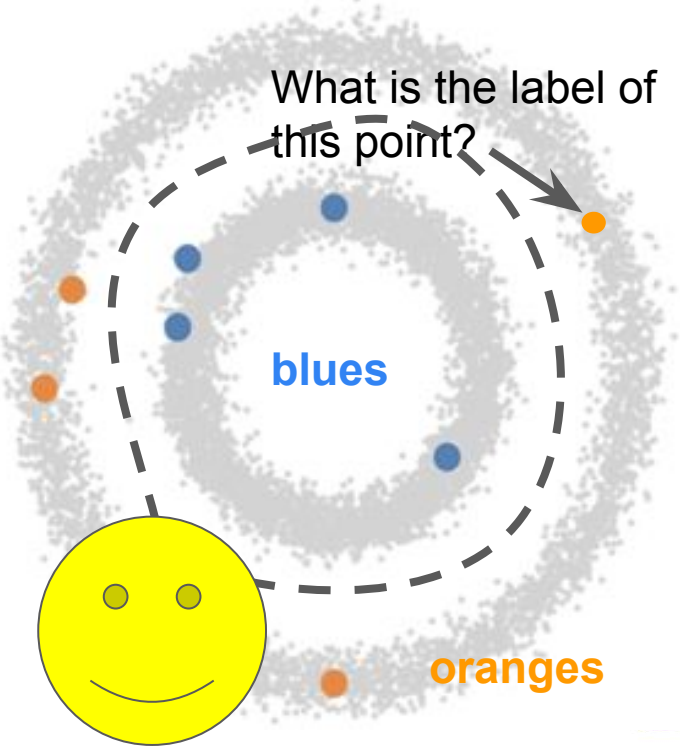
HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings



HOW DOES SSL WORK?

- Consider a simple classification task
- Two classes: blues and oranges
- What if we only had a few training samples?
- With unlabeled points, it is easy to see that the data forms two rings
- With labeled points, we can assign class labels to the rings

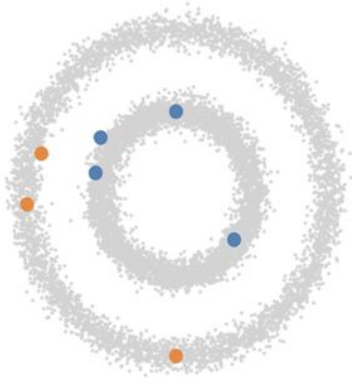


HOW DOES SSL WORK?

The Idea

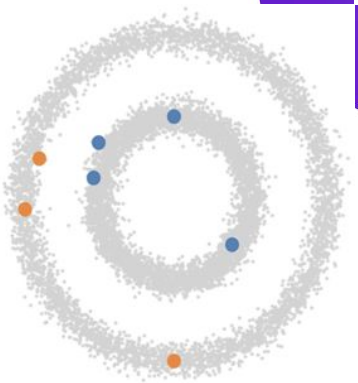
The unlabeled points give us information about the distribution of the data (e.g. through clustering or GANs)

The labeled points can then be used to assign class labels



WHAT ARE SOME OF THE WAYS TO DO SSL?

- TODO



THEORY

Generative Adversarial Networks

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)

Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network generates an image from random input noise

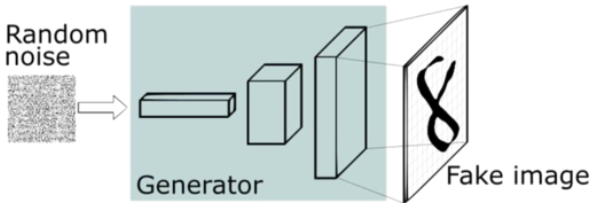


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network
  generates an image from random input noise

Most image classifiers: **convolutional network**
Takes an image, reduces its size at each layer
Image → a 1D array (vector)

Deconvolutional network: the opposite mechanism
Start with a vector, increase the size at each layer
Input vector → an Image

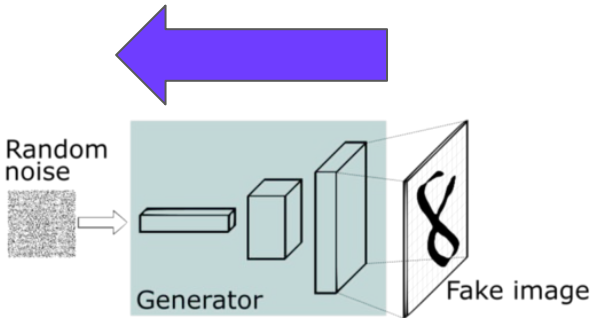


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network
  generates an image from random input noise

Most image classifiers: **convolutional network**
Takes an image, reduces its size at each layer
Image → a 1D array (vector)

Deconvolutional network: the opposite mechanism
Start with a vector, increase the size at each layer
Input vector → an Image

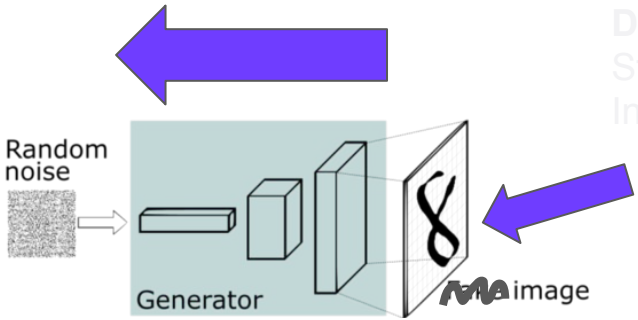


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network
  generates an image from random input noise

Most image classifiers: **convolutional network**
Takes an image, reduces its size at each layer
Image → a 1D array (vector)

Deconvolutional network: the opposite mechanism
Start with a vector, increase the size at each layer
Input vector → an Image

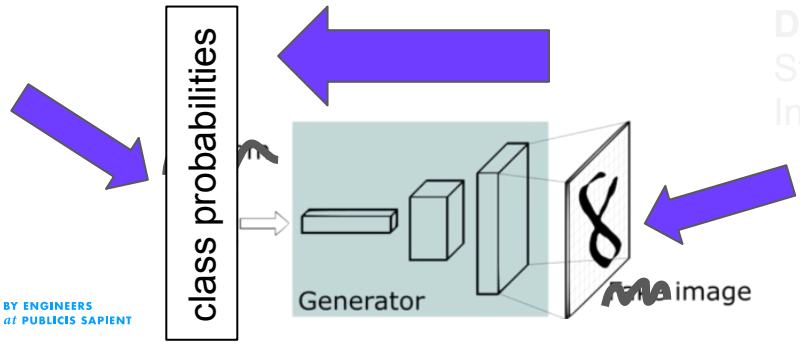
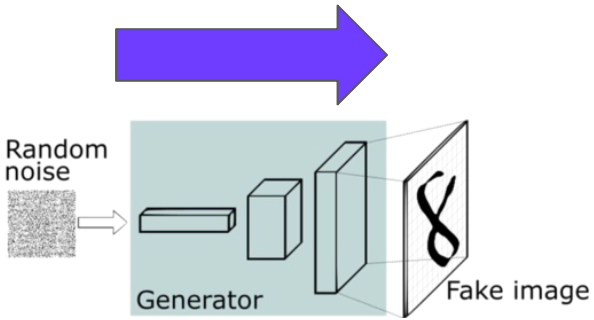


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network
  generates an image from random input noise



Most image classifiers: **convolutional network**
Takes an image, reduces its size at each layer
Image → a 1D array (vector)

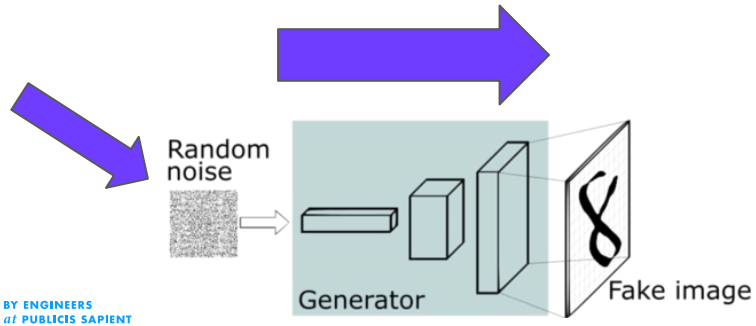
Deconvolutional network: the opposite mechanism
Start with a vector, increase the size at each layer
Input vector → an Image

Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network
  generates an image from random input noise



Most image classifiers: **convolutional network**
Takes an image, reduces its size at each layer
Image → a 1D array (vector)

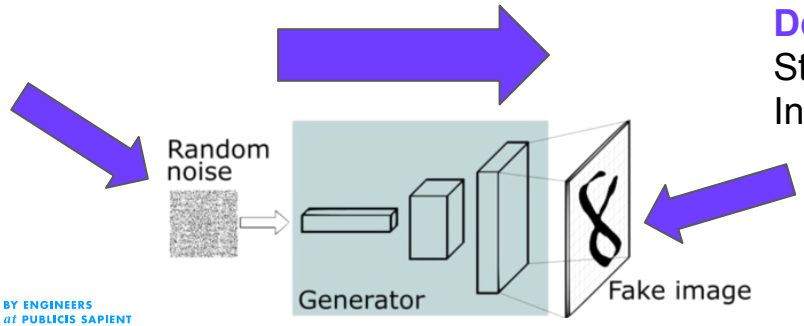
Deconvolutional network: the opposite mechanism
Start with a vector, increase the size at each layer
Input vector → an Image

Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network
  generates an image from random input noise



Most image classifiers: **convolutional network**
Takes an image, reduces its size at each layer
Image → a 1D array (vector)

Deconvolutional network: the opposite mechanism
Start with a vector, increase the size at each layer
Input vector → an Image

Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network generates an image from random input noise

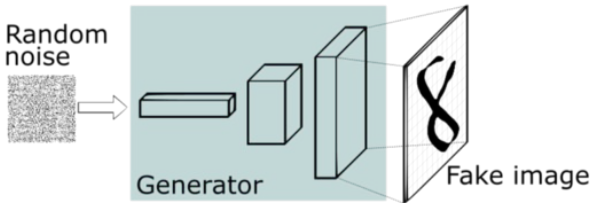


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network generates an image from random input noise
- **Discriminator**

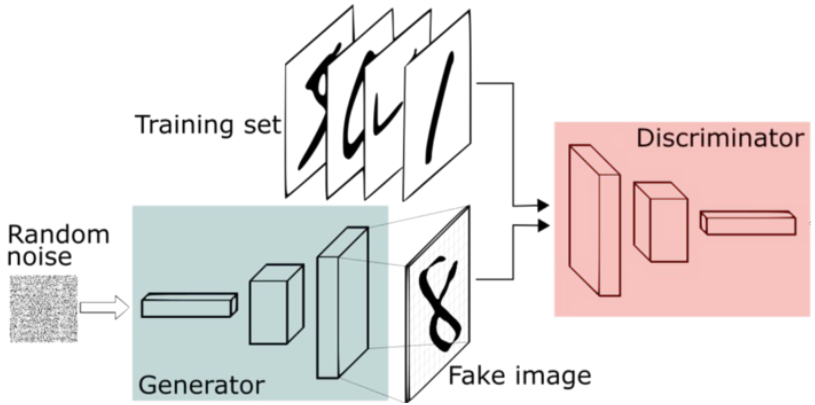


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network generates an image from random input noise
- **Discriminator**

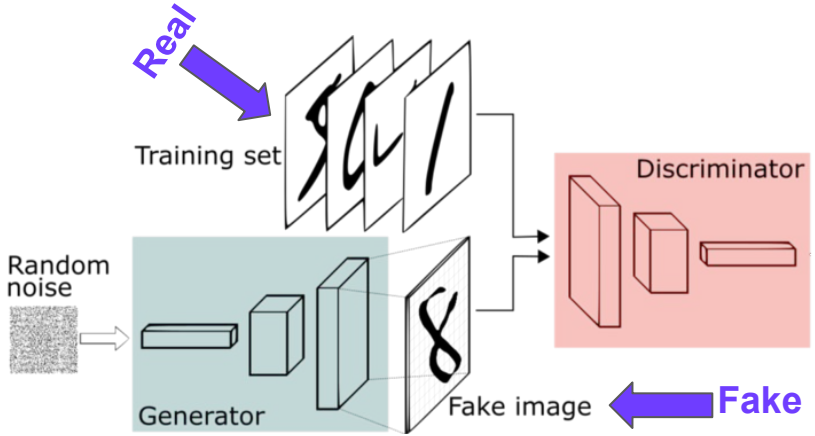


Image source

GANs: Generative Adversarial Networks

Ref: *Ian Goodfellow et al., 2014*

- Two networks: **Generator** + **Discriminator**
- **Generator:** deconvolutional network generates an image from random input noise
- **Discriminator:** binary classifier trained to detect Real (from the training set) vs. Fake (produced by the Generator)

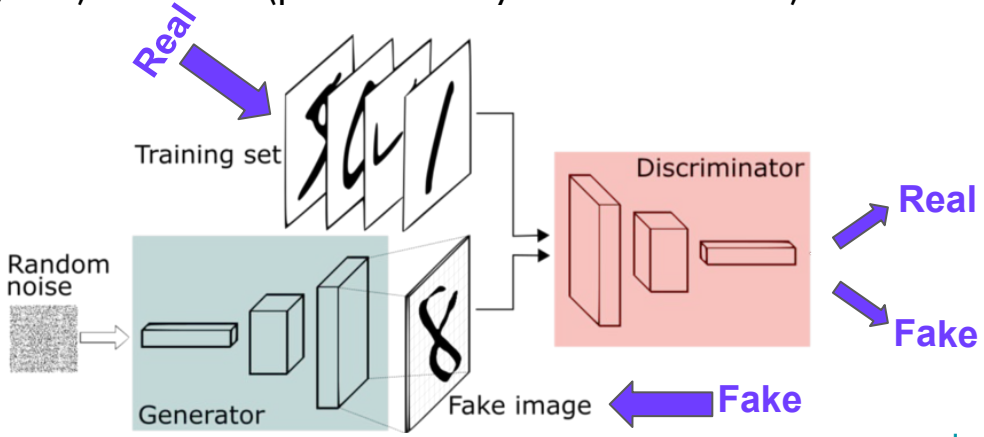


Image source

GANs: Generative Adversarial Networks

Training the Discriminator

- Straightforward binary classification (Real vs. Fake)

Training the Generator

- Generator produces an image → pass it through the Discriminator
- If Discriminator classifies it as Fake → update the Generator to produce better images

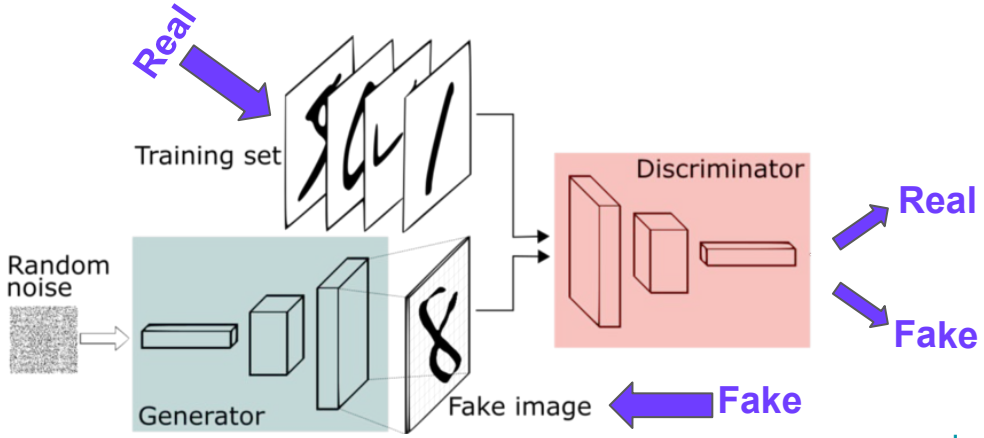


Image source

GANs: Generative Adversarial Networks

Training the Discriminator

- Straightforward binary classification (Real vs. Fake)

Training the Generator

- Generator produces an image → pass it through the Discriminator
- If Discriminator classifies it as Fake → update the Generator to produce better images

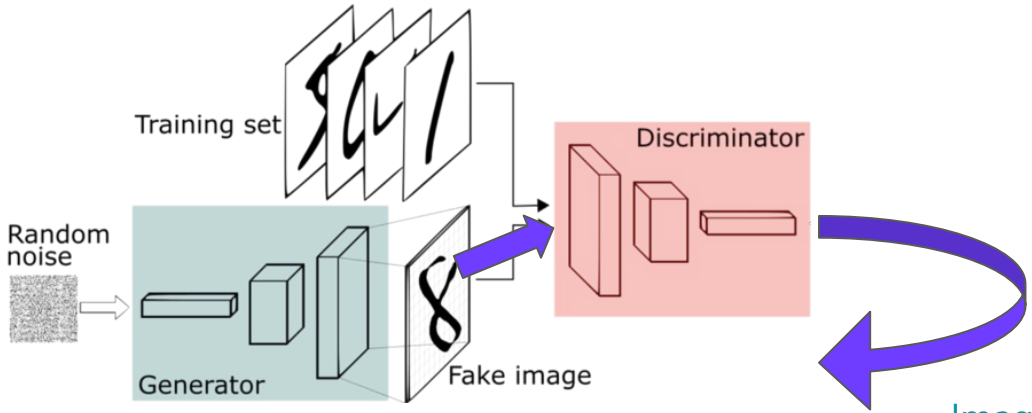


Image source

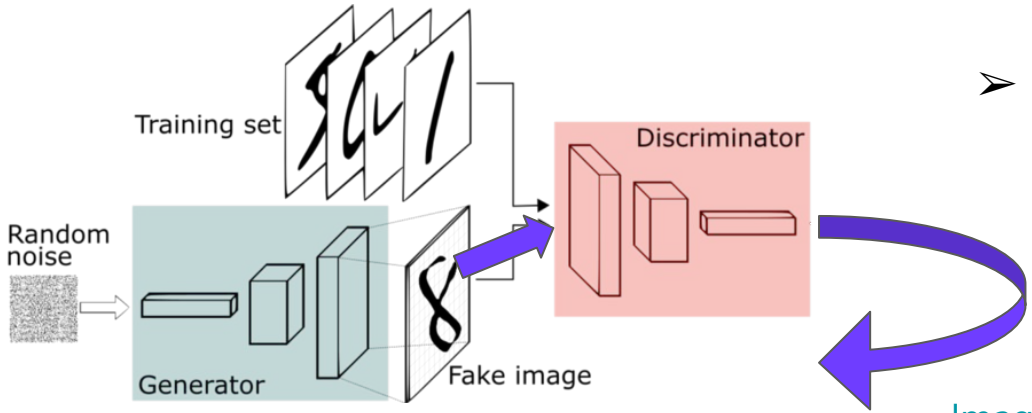
GANs: Generative Adversarial Networks

Training the Discriminator

- Straightforward binary classification (Real vs. Fake)

Training the Generator

- Generator produces an image → pass it through the Discriminator
- If Discriminator classifies it as Fake → update the Generator to produce better images



What's the objective?

- Generator gets good enough that Discriminator can't tell the difference between Real and Fake

Image source

GANs: Generative Adversarial Networks

- 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

GANs: Generative Adversarial Networks

- 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

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

GANs: Generative Adversarial Networks

- 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

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

Binary cross entropy: $-(y \log(p) + (1-y)\log(1-p))$

How good can GANs get?

How good can GANs get?

Really good. So good, it’s freaky!

Example: https://thispersondoesnotexist.com/

How good can GANs get?

Really good. So good, it’s freaky!

Example: https://thispersondoesnotexist.com/

How good can GANs get?

Really good. So good, it’s freaky!

Example: https://thispersondoesnotexist.com/

How good can GANs get?

Really good. So good, it's freaky!

Example: https://thispersondoesnotexist.com/

How good can GANs get?

Really good. So good, it's freaky!

Example: https://thispersondoesnotexist.com/




How good can GANs get?

Really good. So good, it's freaky!

Example: https://thispersondoesnotexist.com/





How good can GANs get?

Really good. So good, it's freaky!

Example: https://thispersondoesnotexist.com/





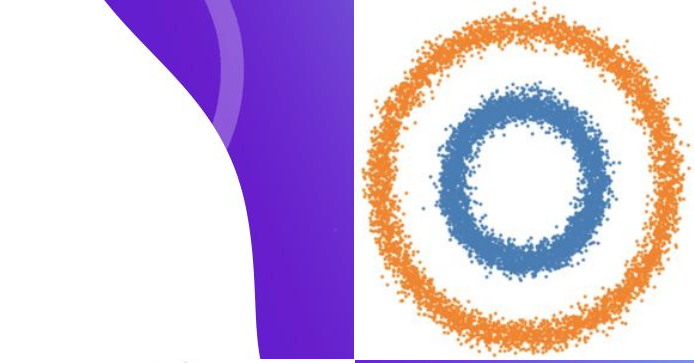
Good for these unsupervised GANs. But, how can they help us with semi-supervised learning?

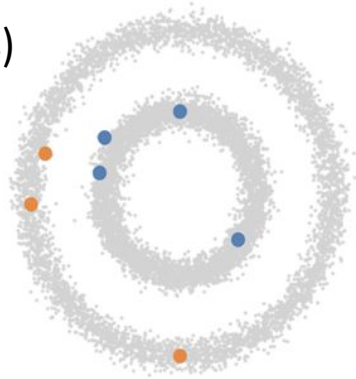
THEORY

Semi-Supervised Learning with GANs

WHAT IS THE BASIC IDEA HERE?

Ref: *Improved Techniques for Training GANs, 2016*



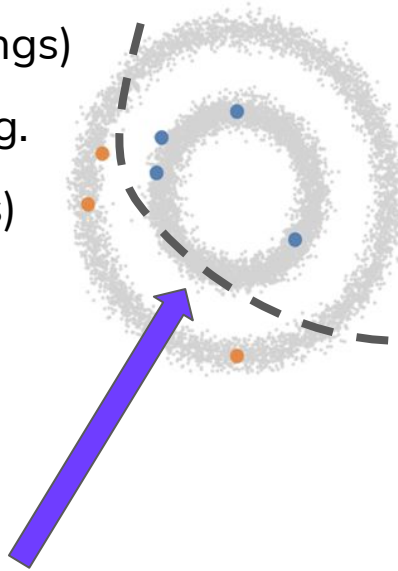


- Our data has a certain distribution (e.g. the two rings)
- The GAN will attempt to learn that distribution (e.g. Generator will be trained to produce the two rings)
- The few labeled datapoints can then be used to assign class labels (e.g. **blue** ring vs. **orange** ring)

WHAT IS THE BASIC IDEA HERE?

Ref: *Improved Techniques for Training GANs, 2016*

- Our data has a certain distribution (e.g. the two rings)
- The GAN will attempt to learn that distribution (e.g. Generator will be trained to produce the two rings)
- The few labeled datapoints can then be used to assign class labels (e.g. **blue** ring vs. **orange** ring)

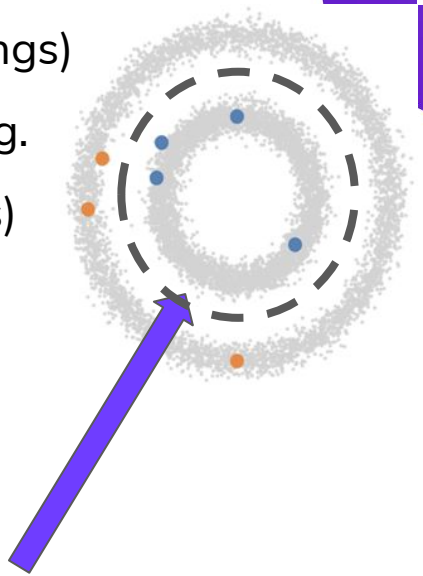


Decision boundary **without** the unlabelled data

WHAT IS THE BASIC IDEA HERE?

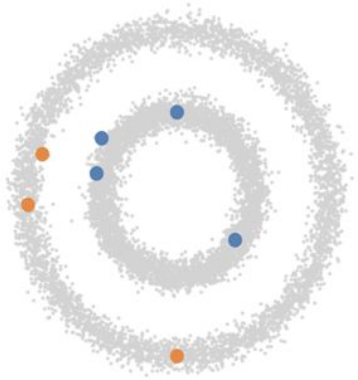
Ref: *Improved Techniques for Training GANs, 2016*

- Our data has a certain distribution (e.g. the two rings)
- The GAN will attempt to learn that distribution (e.g. Generator will be trained to produce the two rings)
- The few labeled datapoints can then be used to assign class labels (e.g. **blue** ring vs. **orange** ring)



Decision boundary **with** the unlabelled data

HOW DOES THIS WORK?



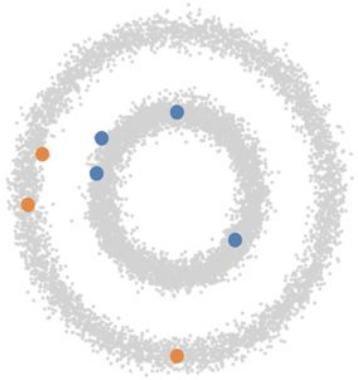
Ref: *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

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

  (**blues**, **oranges** and Fakes)
- Semi-supervised GAN:

  Generator + the new **Discriminator/Classifier**

HOW DOES THIS WORK?



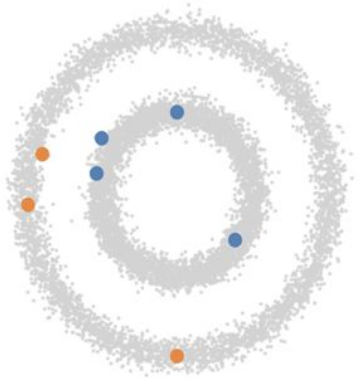
Ref: *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

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

  (**blues**, **oranges** and Fakes)
- Semi-supervised GAN:

  Generator + the new **Discriminator/Classifier**

HOW DOES THIS WORK?



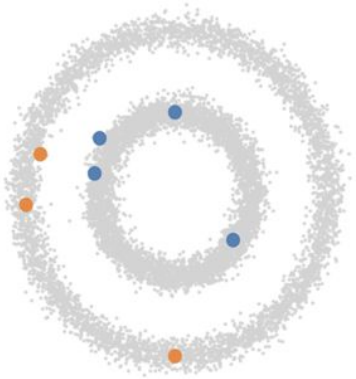
Ref: *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

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

  (**blues**, **oranges** and Fakes)
- Semi-supervised GAN:

  Generator + the new **Discriminator/Classifier**

HOW DOES THIS WORK?



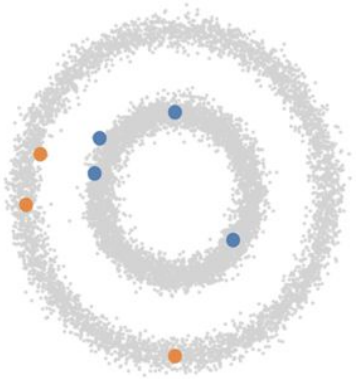
Ref: *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

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

  (**blues**, **oranges** and Fakes)
- Semi-supervised GAN:

  Generator + the new **Discriminator/Classifier**

HOW DOES THIS WORK?



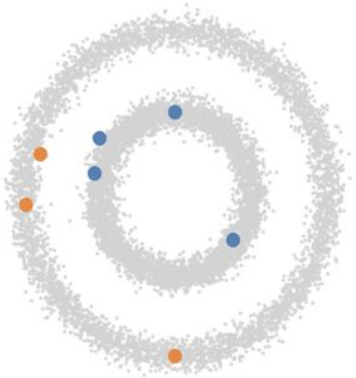
Ref: *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

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

  (**blues**, **oranges** and Fakes)
- Semi-supervised GAN:

  Generator + the new **Discriminator/Classifier**

HOW DOES THIS WORK?



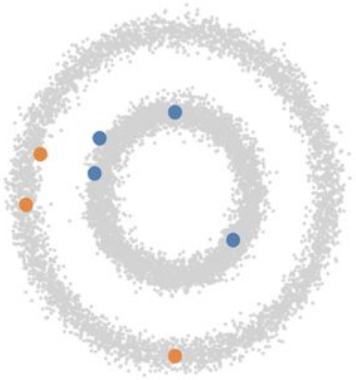
Ref: *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

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

  (**blues**, **oranges** and Fakes)
- Semi-supervised GAN:

  Generator + the new **Discriminator/Classifier**

HOW ARE THE NETWORKS TRAINED?



Three kinds of data: *labeled*, *unlabeled*, and *generated* from noise

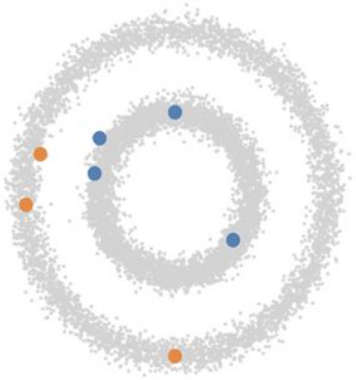
Discriminator is trained to assign:

- *labeled* data to correct classes (**blues** vs. **oranges**)
- *unlabeled* data to one of K classes (**blues** or **oranges**, does not matter which)
- *generated* data to (K+1) class (Fake)

Generator is trained to:

- *generate* images that Discriminator would assign to one of K classes (**blues** or **oranges**, does not matter which)

HOW ARE THE NETWORKS TRAINED?



Three kinds of data: *labeled*, *unlabeled*, and *generated* from noise

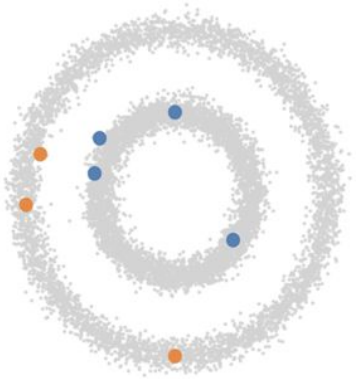
Discriminator is trained to assign:

- *labeled* data to correct classes (**blues** vs. **oranges**)
- *unlabeled* data to one of K classes (**blues** or **oranges**, does not matter which)
- *generated* data to (K+1) class (Fake)

Generator is trained to:

- *generate* images that Discriminator would assign to one of K classes (**blues** or **oranges**, does not matter which)

HOW ARE THE NETWORKS TRAINED?



Three kinds of data: ***labeled***, *unlabeled*, and *generated* from noise

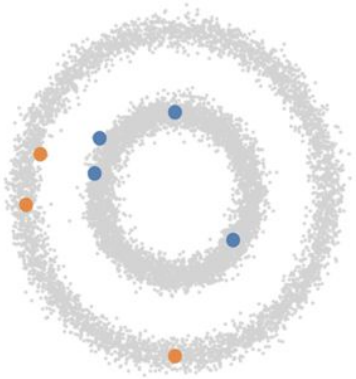
Discriminator is trained to assign:

- *labeled* data to correct classes (**blues** vs. **oranges**)
- *unlabeled* data to one of K classes (**blues** or **oranges**, does not matter which)
- *generated* data to (K+1) class (Fake)

Generator is trained to:

- *generate* images that Discriminator would assign to one of K classes (**blues** or **oranges**, does not matter which)

HOW ARE THE NETWORKS TRAINED?



Three kinds of data: *labeled*, ***unlabeled***, and *generated* from noise

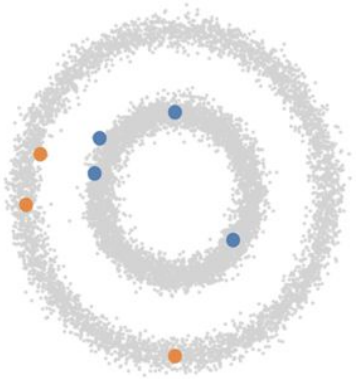
Discriminator is trained to assign:

- *labeled* data to correct classes (**blues** vs. **oranges**)
- *unlabeled* data to one of K classes (**blues** or **oranges**, does not matter which)
- *generated* data to (K+1) class (Fake)

Generator is trained to:

- *generate* images that Discriminator would assign to one of K classes (**blues** or **oranges**, does not matter which)

HOW ARE THE NETWORKS TRAINED?



Three kinds of data: *labeled*, *unlabeled*, and ***generated*** from noise

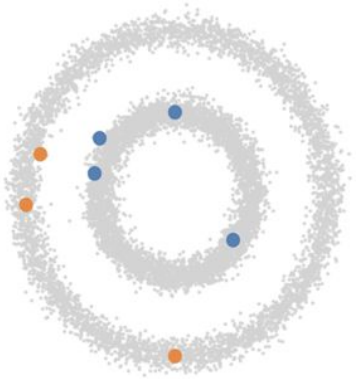
Discriminator is trained to assign:

- *labeled* data to correct classes (**blues** vs. **oranges**)
- *unlabeled* data to one of K classes (**blues** or **oranges**, does not matter which)
- *generated* data to (K+1) class (Fake)

Generator is trained to:

- *generate* images that Discriminator would assign to one of K classes (**blues** or **oranges**, does not matter which)

HOW ARE THE NETWORKS TRAINED?



Three kinds of data: *labeled*, *unlabeled*, and *generated* from noise

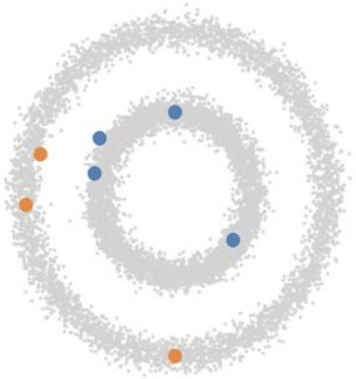
Discriminator is trained to assign:

- *labeled* data to correct classes (**blues** vs. **oranges**)
- *unlabeled* data to one of K classes (**blues** or **oranges**, does not matter which)
- *generated* data to (K+1) class (Fake)

Generator is trained to:

- *generate* images that Discriminator would assign to one of K classes (**blues** or **oranges**, does not matter which)

HOW ARE THE NETWORKS TRAINED?



Three kinds of data: *labeled*, *unlabeled*, and *generated* from noise

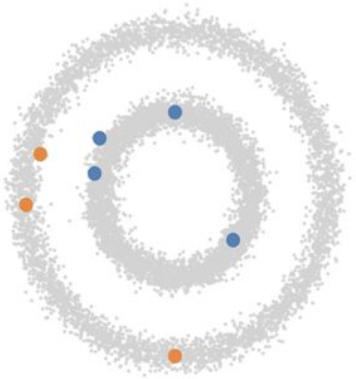
Discriminator is trained to assign:

- *labeled* data to correct classes (**blues** vs. **oranges**)
- *unlabeled* data to one of K classes (**blues** or **oranges**, does not matter which)
- *generated* data to (K+1) class (Fake)

Generator is trained to:

- *generate* images that Discriminator would assign to one of K classes (**blues** or **oranges**, does not matter which)

HOW ARE THE NETWORKS TRAINED?



Three kinds of data: *labeled*, *unlabeled*, and *generated* from noise

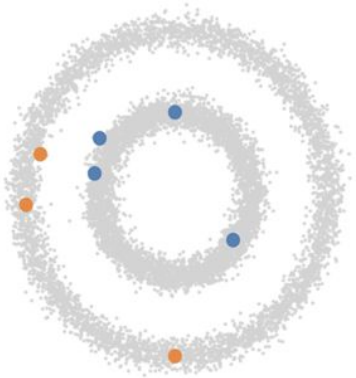
Discriminator is trained to assign:

- *labeled* data to correct classes (**blues** vs. **oranges**)
- *unlabeled* data to one of K classes (**blues** or **oranges**, does not matter which)
- *generated* data to (K+1) class (Fake)

Generator is trained to:

- *generate* images that Discriminator would assign to one of K classes (**blues** or **oranges**, does not matter which)

HOW ARE THE NETWORKS TRAINED?



Three kinds of data: *labeled*, *unlabeled*, and *generated* from noise

Discriminator is trained to assign:

- *labeled* data to correct classes (**blues** vs. **oranges**)
- *unlabeled* data to one of K classes (**blues** or **oranges**, does not matter which)
- *generated* data to (K+1) class (Fake)

Generator is trained to:

- *generate* images that Discriminator would assign to one of K classes (**blues** or **oranges**, does not matter which)

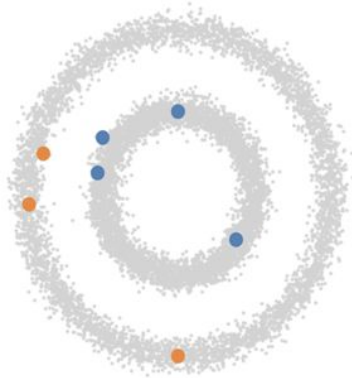
HOW GOOD IS THE TRAINED GENERATOR?

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

- Unsupervised GANs can produce very impressive images
- 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)

The two rings example:

- Generating more points for the two circles would not help us
- Generating *Fake* points between circles
  → better decision boundary between classes



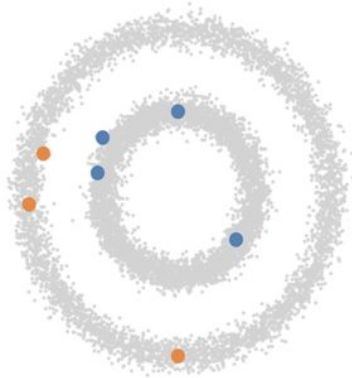
HOW GOOD IS THE TRAINED GENERATOR?

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

- Unsupervised GANs can produce very impressive images
- 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)

The two rings example:

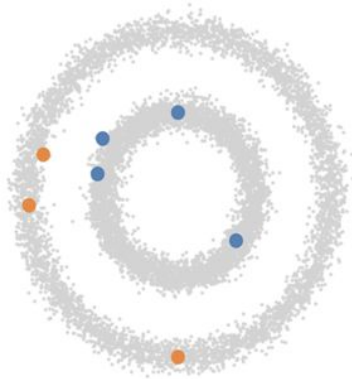
- Generating more points for the two circles would not help us
- Generating *Fake* points between circles
  → better decision boundary between classes



HOW GOOD IS THE TRAINED GENERATOR?

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

- Unsupervised GANs can produce very impressive images
- 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)



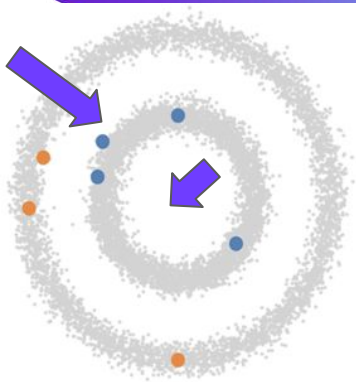
The two rings example:

- 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 TRAINED GENERATOR?

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

- Unsupervised GANs can produce very impressive images
- 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)



The two rings example:

- Generating more points for the two circles would not help us
- Generating *Fake* points between circles
  → better decision boundary between classes

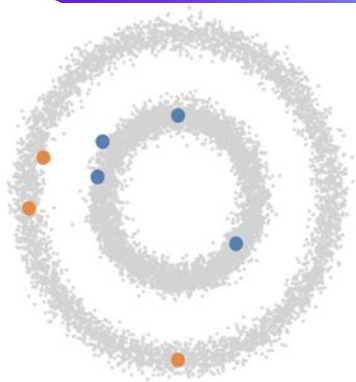
HOW GOOD IS THE TRAINED GENERATOR?

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

- Unsupervised GANs can produce very impressive images
- 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)

The two rings example:

- Generating more points for the two circles would not help us
- Generating *Fake* points between circles
  → better decision boundary between classes



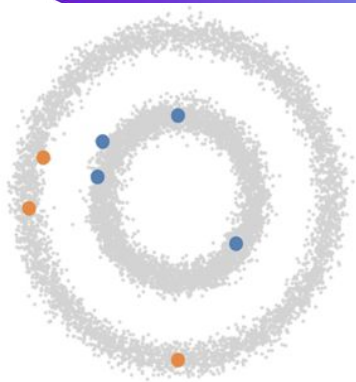
HOW GOOD IS THE TRAINED GENERATOR?

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

- Unsupervised GANs can produce very impressive images
- 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)

The two rings example:

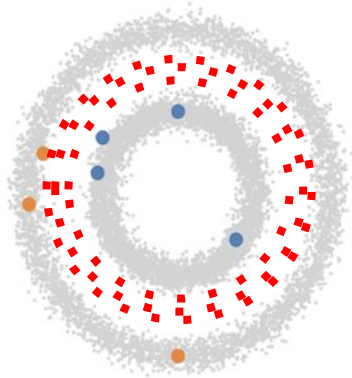
- Generating more points for the two circles would not help us
- Generating *Fake* points between circles
  → better decision boundary between classes



HOW GOOD IS THE TRAINED GENERATOR?

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

- Unsupervised GANs can produce very impressive images
- 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)



The two rings example:

- Generating more points for the two circles would not help us
- Generating *Fake* points between circles
  → better decision boundary between classes

PRACTICE

A Tale of Cats & Dogs

PRACTICE

A Tale of Cats & Dogs

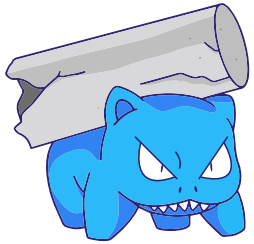
github.com/opetrova/SemiSupervisedPytorchGAN

PRACTICE

A Tale of Cats & Dogs

github.com/opetrova/SemiSupervisedPytorchGAN

A Cautionary



MORAL(S) OF THE STORY

MORAL(S) OF THE STORY

1. Your labelled data matters **A LOT** 🐈 🐕 🐩
   10% average rise in accuracy for randomly chosen cats+dogs
   vs. 20%+ for a curated labeled dataset (“representative” images)

2. Naively adding a new Fake class to the Classifier does **NOT** work
   Instead:
   - Keep number of output logits at K (K=2 for binary classification)
   - Change your loss function:

Ref: *Improved Techniques for Training GANs, 2016*, *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

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

MORAL(S) OF THE STORY

1. Your labelled data matters **A LOT** 🐈 🐕 🐩
   10% average rise in accuracy for randomly chosen cats+dogs
   vs. 20%+ for a curated labeled dataset (“representative” images)
2. Naively adding a new Fake class to the Classifier does **NOT** work 😦
   Instead:
   - Keep number of output logits at K (K=2 for binary classification)
   - Change your loss function:

Ref: *Improved Techniques for Training GANs, 2016*, *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

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

MORAL(S) OF THE STORY

1. Your labelled data matters **A LOT** 🐈 🐕 🐩
   10% average rise in accuracy for randomly chosen cats+dogs
   vs. 20%+ for a curated labeled dataset (“representative” images)

2. Naively adding a new Fake class to the Classifier does **NOT** work 😦
   Instead:

- Keep number of output logits at K (K=2 for binary classification)
- Change your loss function:

Ref: *Improved Techniques for Training GANs, 2016*, *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks, 2016*

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

MORAL(S) OF THE STORY

1. Your labelled data matters **A LOT** 🐈 🐕 🐩
   10% average rise in accuracy for randomly chosen cats+dogs
   vs. 20%+ for a curated labeled dataset (“representative” images)

2. Naively adding a new Fake class to the Classifier does **NOT** work 😮
   Instead:
   - Keep number of output logits at K (K=2 for binary classification)
   - Change your loss function: $D(\boldsymbol{x}) = \frac{Z(\boldsymbol{x})}{Z(\boldsymbol{x})+1}$, 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*

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

MORAL(S) OF THE STORY

1. Your labelled data matters **A LOT** 🐈 🐕 🐩
   10% average rise in accuracy for randomly chosen cats+dogs
   vs. 20%+ for a curated labeled dataset (“representative” images)

2. Naively adding a new Fake class to the Classifier does **NOT** work 😮
   Instead:

- Keep number of output logits at K (K=2 for binary classification)
- Change your loss function: $D(\boldsymbol{x}) = \frac{Z(\boldsymbol{x})}{Z(\boldsymbol{x})+1}$, 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*

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

MERCI

CONTACT
Olga PETROVA

https://www.linkedin.com/in/olga-p-petrova/