Generative Adversarial Networks and AutoEncoders

ECEN 478

Dr. Mahmoud Nabil Mahmoud
mnmahmoud@ncat.edu

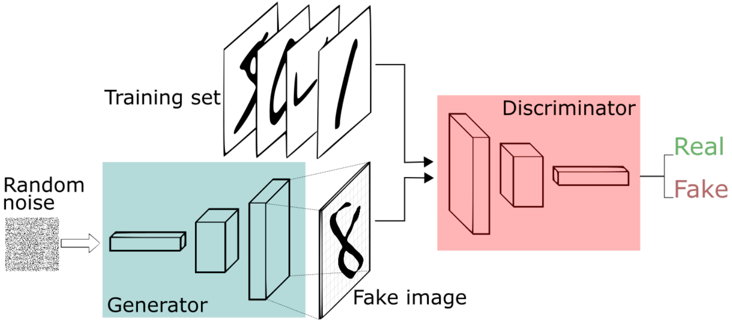
North Carolina A & T State University

April 5, 2022

Outline

Generative Adversarial Networks

Generative adversarial networks (GANs) are architectures that use two neural networks, pitting one against the other (thus the "adversarial") in order to generate new, synthetic instances of data that can pass for real data.



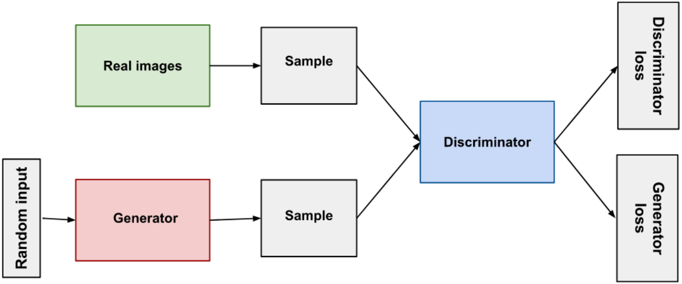
GAN Example[1]

[1] https://www.thispersondoesnotexist.com/

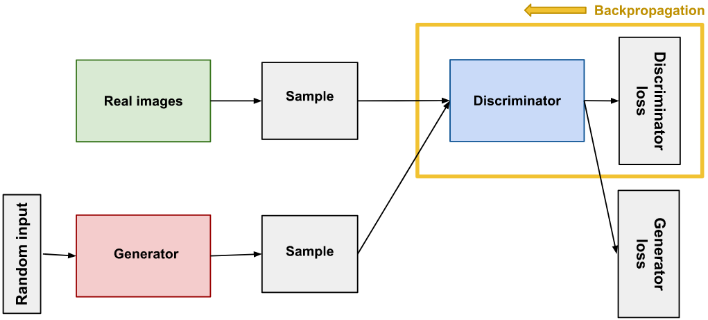
GANs steps

- When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake.
- As training progresses, the generator gets closer to producing output that can fool the discriminator.
- Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases.

GANs Loss


Real images
Sample
Random input
Generator
Sample
Discriminator
Discriminator loss
Generator loss

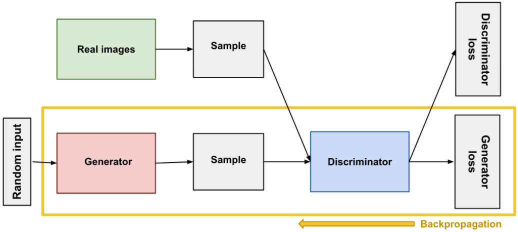
The Discriminator Training



The discriminator's training data comes from two sources:

- Real data instances, such as real pictures of people.
- Fake data instances created by the generator.

The Generator Training



Generator Training Steps:

1. Sample random noise.
2. Produce generator output from sampled random noise.
3. Get discriminator "Real" or "Fake" classification for generator output.
4. Calculate loss from discriminator classification.
5. Back-propagate through both the discriminator and generator to obtain gradients.
6. Use gradients to change only the generator weights.

Training GAN

Alternating Training

1. GAN training proceeds in alternating periods:
2. The discriminator trains for one or more epochs. The generator trains for one or more epochs. Repeat steps 1 and 2 to continue to train the generator and discriminator networks.

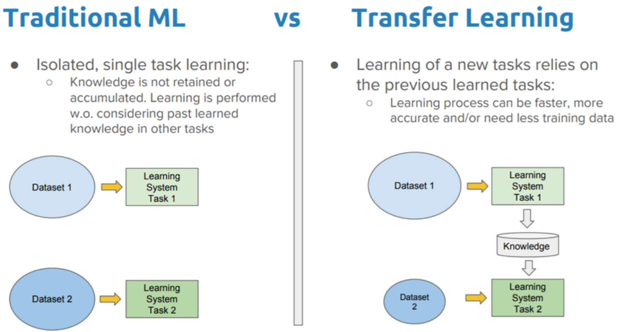
Outline

What is Transfer Learning?

Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is re-used on a second related task.

- Only works in deep learning if the model features learned from the first task are general.

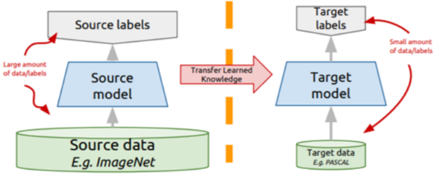


Transfer Learning Idea

- Take a network trained on different domain for a different Source Task.
- Adapt it to for your domain and Target Task

Variations

- Same domain, differnet task.
- Different domain, same task.

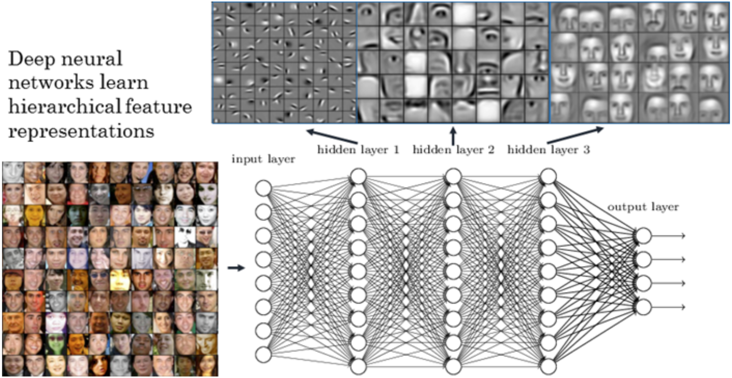


Why Transfer Learning?

- Alternative is to build model from scratch
  - Time cosuming
  - Hard feature engineering process.
- Can Lessen the data demands.
- Transfer Learning can be used for privacy.
- Can be used to improve a model performance.

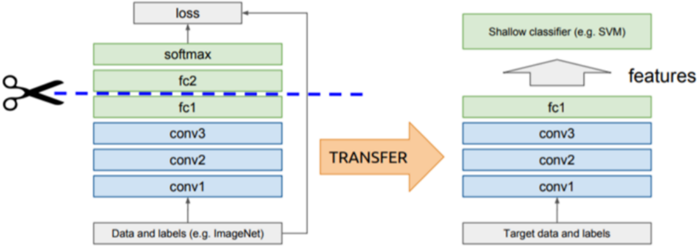
How does it work?

- Deep learning systems and models are layered architectures that learn different features at different layers.
- The initial layers have been seen to capture generic features, while the later ones focus more on the specific task at hand.



How does it work?

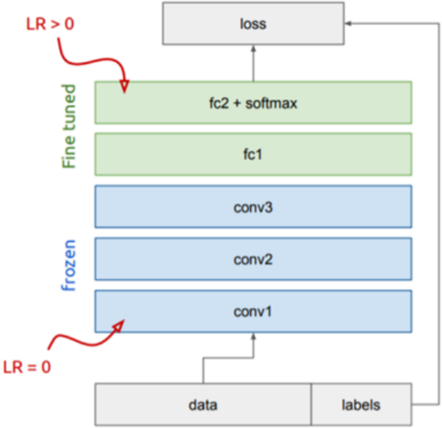
- The key idea is to leverage the pre-trained model's weighted layers to extract features but not to update the weights of the model's layers during training with new data for the new task.



Freeze or Fine-tune

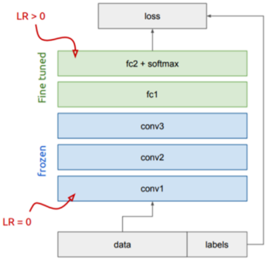
- **Freezed layers:** not updated during backprop.
- **Fine-tuned layer:** updated during backprop.

In general, we can set learning rates to be different for each layer. Our goal is retraining the head of a network to recognize classes it was not originally intended for.



Freeze and Fine-tune

1. If you have very small dataset you may freeze all layers except softmax in the source model and replace it and train only softmax of target model.
2. If you have small dataset you may freeze large number of layers in the source model while train low number of layers.
3. If you have large dataset you may freeze low number of layers while train more number of layers.



Outline

Unsupervised Learning

We have access to $\{x_1, x_2, \ldots, x_N\}$ but not $\{y_1, y_2, \ldots, y_N\}$ (where x_i is the data sample i and y_i is the target)

Why would we want to tackle such a task:

1. Extracting interesting information from data
    - Clustering
    - Discovering interesting trend
    - Data compression
2. Learn better representations

Unsupervised Representation Learning

Force our **representations** to better model input distribution

- Not just extracting features for classification
- Asking the model to be good at representing the data and not overfitting to a particular task (we get this with ImageNet, but maybe we can do better)
- Potentially allowing for better generalization

Use for **initialization of supervised task**, especially when we have a lot of unlabeled data and much less labeled examples

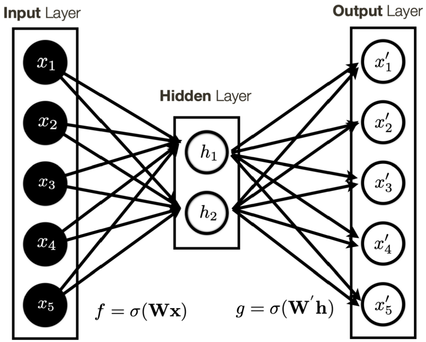
Autoencoders

Self (i.e. self-encoding)

Autoencoders

Self (i.e. self-encoding)

- Feed forward network intended to reproduce the input

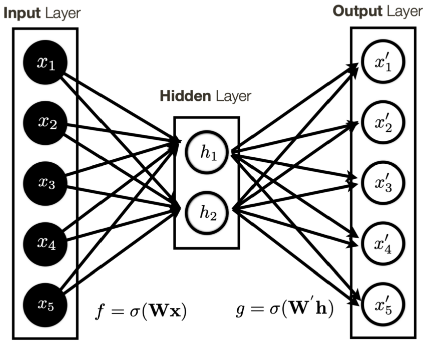


Autoencoders

Self (i.e. self-encoding)

- Feed forward network intended to reproduce the input
- Encoder/Decoder architecture

Encoder: $\mathbf{h} = \sigma(\mathbf{Wx})$
Decoder $\mathbf{f} = \sigma(\mathbf{W'h})$



Autoencoders

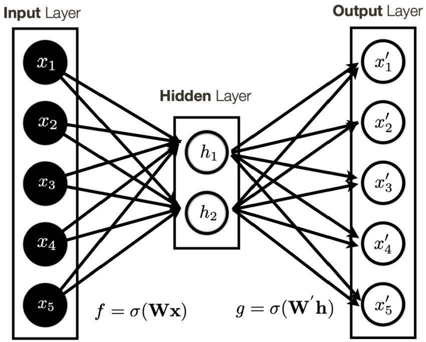
Self (i.e. self-encoding)

- Feed forward network intended to reproduce the input
- Encoder/Decoder architecture

  Encoder: $\mathbf{h} = \sigma(\mathbf{Wx})$
  Decoder $\mathbf{f} = \sigma(\mathbf{W'h})$

- Score function

$$\mathbf{x'} = h(f(\mathbf{x}))$$
$$\ell(\mathbf{x'},\mathbf{x})$$



Autoencoders: Hidden Layer Dimensionality

Smaller than the input

- Will compress the data, reconstruction of the data is difficult unless you have the decoder
- Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (Principal Component Analysis)

Larger than the input

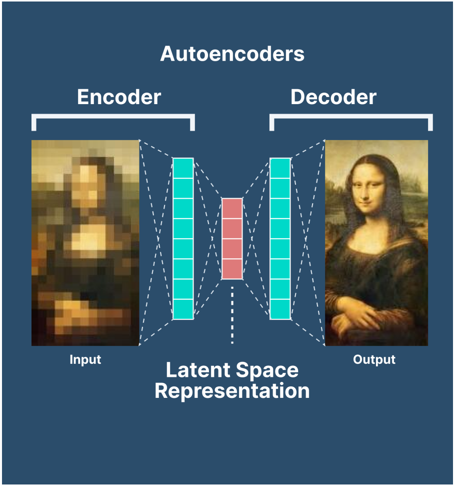
- Can trivially learn to just copy, no structure is learned (unless you regularize)
- Does not encourage learning of meaningful features

De-noising Autoencoder

Idea: add noise to input but learn to reconstruct the original

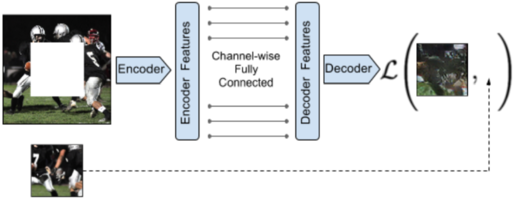
- Leads to better representations
- Prevents copying

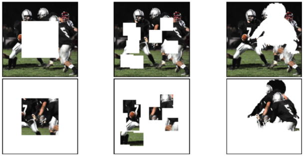
Note: different noise is added during each epoch



Context Encoders

[Pathak et al., 2016]

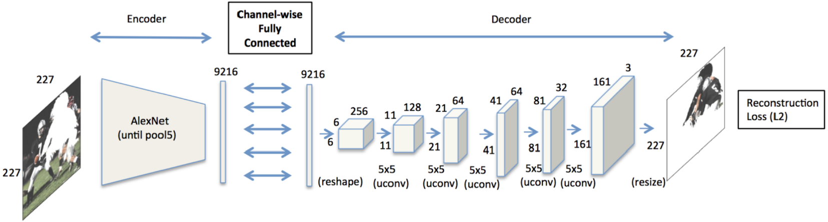




(a) Central region (b) Random block (c) Random region

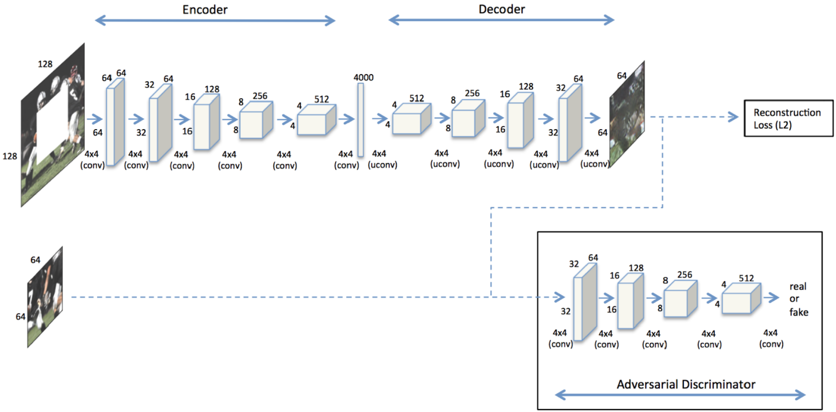
Context Encoders

[Pathak et al., 2016]



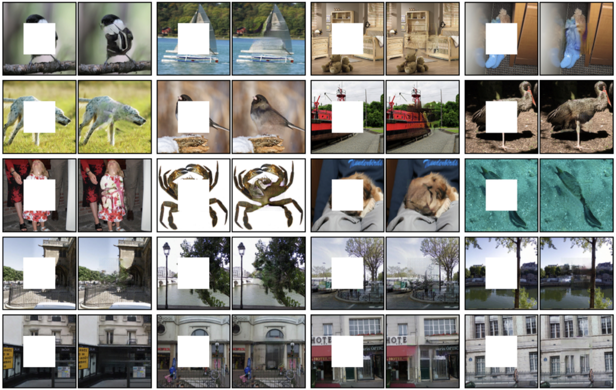
Context Encoders

[Pathak et al., 2016]



Context Encoders

[Pathak et al., 2016]



Outline

Deep Learning Frameworks (1/2)

- TensorFlow

Deep Learning Frameworks (1/2)

- TensorFlow
  - Google's Tensorflow the most popular Deep Learning.

Deep Learning Frameworks (1/2)

- TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.

Deep Learning Frameworks (1/2)

- TensorFlow TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch PYTORCH

Deep Learning Frameworks (1/2)

- TensorFlow TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch PYTORCH
  - Was developed for Facebook services

Deep Learning Frameworks (1/2)

- TensorFlow TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch PYTORCH
  - Was developed for Facebook services
  - In PyTorch, you can use standard debuggers, for example, pdb or PyCharm.

Deep Learning Frameworks (1/2)

- TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch
  - Was developed for Facebook services
  - In PyTorch, you can use standard debuggers, for example, pdb or PyCharm.
- Sonnet

Deep Learning Frameworks (1/2)

- TensorFlow TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch PYTORCH
  - Was developed for Facebook services
  - In PyTorch, you can use standard debuggers, for example, pdb or PyCharm.
- Sonnet Sonnet
  - Built on top of TensorFlow.

Deep Learning Frameworks (1/2)

- TensorFlow TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch PYTORCH
  - Was developed for Facebook services
  - In PyTorch, you can use standard debuggers, for example, pdb or PyCharm.
- Sonnet Sonnet
  - Built on top of TensorFlow.
  - High-level object-oriented libraries.

Deep Learning Frameworks (1/2)

- TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch
  - Was developed for Facebook services
  - In PyTorch, you can use standard debuggers, for example, pdb or PyCharm.
- Sonnet
  - Built on top of TensorFlow.
  - High-level object-oriented libraries.
- Keras

Deep Learning Frameworks (1/2)

- TensorFlow TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch PYTORCH
  - Was developed for Facebook services
  - In PyTorch, you can use standard debuggers, for example, pdb or PyCharm.
- Sonnet Sonnet
  - Built on top of TensorFlow.
  - High-level object-oriented libraries.
- Keras K Keras
  - The best Deep Learning framework for those who are just starting out.

Deep Learning Frameworks (1/2)

- TensorFlow TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch PYTORCH
  - Was developed for Facebook services
  - In PyTorch, you can use standard debuggers, for example, pdb or PyCharm.
- Sonnet Sonnet
  - Built on top of TensorFlow.
  - High-level object-oriented libraries.
- Keras K Keras
  - The best Deep Learning framework for those who are just starting out.
  - High level like Sonnet.

Deep Learning Frameworks (1/2)

- TensorFlow TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch PYTORCH
  - Was developed for Facebook services
  - In PyTorch, you can use standard debuggers, for example, pdb or PyCharm.
- Sonnet Sonnet
  - Built on top of TensorFlow.
  - High-level object-oriented libraries.
- Keras K Keras
  - The best Deep Learning framework for those who are just starting out.
  - High level like Sonnet.
- MXNet mxnet

Deep Learning Frameworks (1/2)

- TensorFlow TensorFlow
  - Google's Tensorflow the most popular Deep Learning.
  - TF needs a lot of coding.
- PyTorch PYTORCH
  - Was developed for Facebook services
  - In PyTorch, you can use standard debuggers, for example, pdb or PyCharm.
- Sonnet Sonnet
  - Built on top of TensorFlow.
  - High-level object-oriented libraries.
- Keras K Keras
  - The best Deep Learning framework for those who are just starting out.
  - High level like Sonnet.
- MXNet mxnet
  - Effectively parallel on multiple GPUs and many machines..

Deep Learning Frameworks (2/2)

- Gluon GLUON

Deep Learning Frameworks (2/2)

- Gluon GLUON
  - Gluon is based on MXNet and offers a simple API that simplifies the creation of deep learning models.

Deep Learning Frameworks (2/2)

- Gluon GLUON
    - Gluon is based on MXNet and offers a simple API that simplifies the creation of deep learning models.
- Swift SWIFT

Deep Learning Frameworks (2/2)

- Gluon GLUON
  - Gluon is based on MXNet and offers a simple API that simplifies the creation of deep learning models.
- Swift SWIFT
  - Swift for Tensorflow.

Deep Learning Frameworks (2/2)

- Gluon GLUON
    - Gluon is based on MXNet and offers a simple API that simplifies the creation of deep learning models.
- Swift SWIFT
    - Swift for Tensorflow.
- Chainer Chainer

Deep Learning Frameworks (2/2)

- Gluon GLUON
  - Gluon is based on MXNet and offers a simple API that simplifies the creation of deep learning models.
- Swift SWIFT
  - Swift for Tensorflow.
- Chainer Chainer
  - The code is written in pure Python on top of the Numpy and CuPy libraries. (fast)

Deep Learning Frameworks (2/2)

- Gluon GLUON
    - Gluon is based on MXNet and offers a simple API that simplifies the creation of deep learning models.
- Swift SWIFT
    - Swift for Tensorflow.
- Chainer Chainer
    - The code is written in pure Python on top of the Numpy and CuPy libraries. (fast)
- DL4J DL4J

Deep Learning Frameworks (2/2)

- Gluon GLUON
  - Gluon is based on MXNet and offers a simple API that simplifies the creation of deep learning models.
- Swift SWIFT
  - Swift for Tensorflow.
- Chainer Chainer
  - The code is written in pure Python on top of the Numpy and CuPy libraries. (fast)
- DL4J DL4J
  - Deep Learning for Java.

Deep Learning Frameworks (2/2)

- Gluon GLUON
  - Gluon is based on MXNet and offers a simple API that simplifies the creation of deep learning models.
- Swift SWIFT
  - Swift for Tensorflow.
- Chainer Chainer
  - The code is written in pure Python on top of the Numpy and CuPy libraries. (fast)
- DL4J DL4J
  - Deep Learning for Java.
- ONNX ONNX

Deep Learning Frameworks (2/2)

- Gluon GLUON
  - Gluon is based on MXNet and offers a simple API that simplifies the creation of deep learning models.
- Swift SWIFT
  - Swift for Tensorflow.
- Chainer Chainer
  - The code is written in pure Python on top of the Numpy and CuPy libraries. (fast)
- DL4J DL4J
  - Deep Learning for Java.
- ONNX ONNX
  - Enables models to be trained in one framework and transferred to another for inference.

Questions