Pretrained Models + Advanced Architectures

CMSC 389A: Lecture 9

Sujith Vishwajith

University of Maryland

Agenda

1. Pretrained Models
2. Autoencoders
3. R-CNN Model
4. Siamese Networks
5. Seq-Seq Models
6. Generative Adversarial Network
7. Announcements

Pretrained Models

Inspiration

Training models on large datasets is computationally intensive.

Often takes multiple hours or even days to train a single model.

This doesn't even include the time it takes to find the hyper parameters.

Solution: Use a model already trained on the data (Case 1) or similar data (Case 2).

Pretrained Models (Case 1)

If a model is already trained on our data, we can simply load it in.

Chances are it performs better than if we train it manually.

What we need is the architecture of the model and it's weights.

Typically the architecture is saved as a JSON file and the weights are saved in an H5 file.

Examples of pre-trained models commonly used:

- Word2Vec trained on Google News or Wikipedia
- VGG-16, Resnet152, etc. trained on ImageNet

Pretrained Models (Case 2)

If a model is already trained on similar data, we can still use it!

Transfer Learning: Encoding knowledge gained while solving one problem and applying it to a different but related problem.

If we have a model that knows how to identify a variety of objects (ImageNet), we can leverage the features it has learnt for our own problem.

We just need to slightly retrain the last fully connected dense layers to output our own classes instead of the classes the model used.

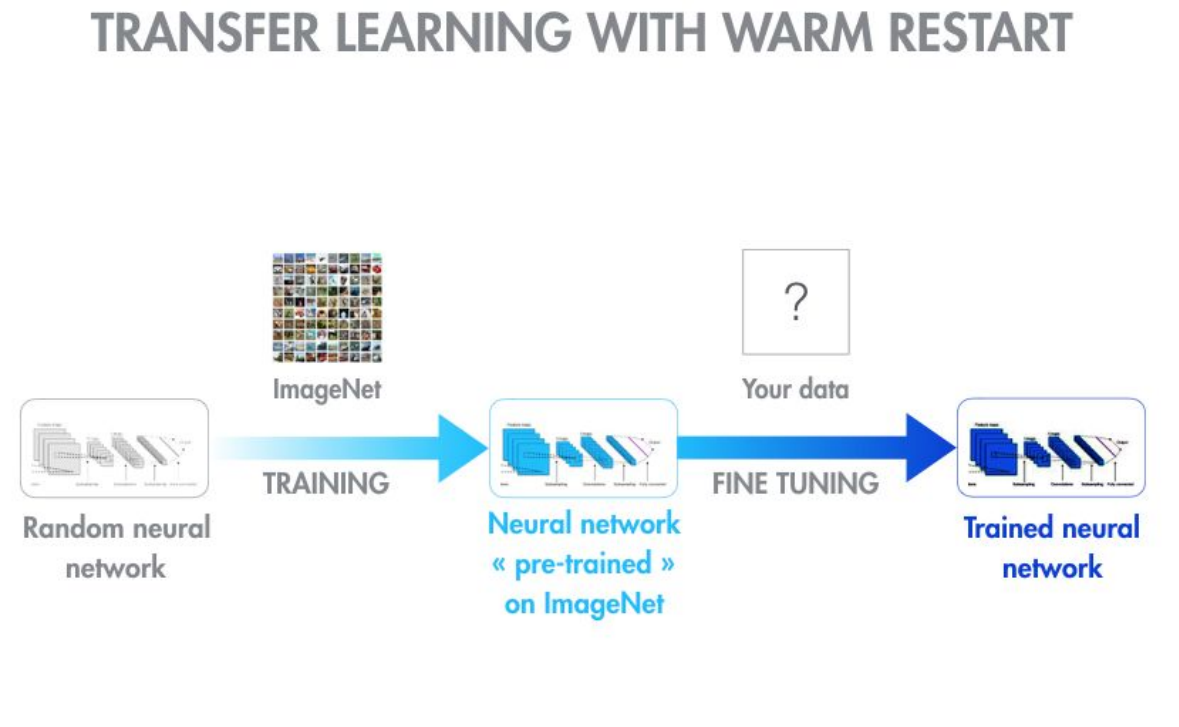
Since we aren't retraining the entire model nor the convolutional layers, this is incredibly fast to run.

Pretrained Models (Case 2)

Transfer Learning Steps for CNNs:

1. Load the pretrained model but exclude the top layers of the model (dense layers).
    - Essentially we are just extracting the CNN layers.
2. Add the dense layers of your choice (final one should contain the number of classes you have to predict).
    - We change this because their task was different from ours.
3. Train your new model again on your data.
    - Make sure your input images are in the same format as the pretrained model.
    - The reason it is fast is because it is already reasonably close to an optimum and doesn't have to find it from scratch.

Pretrained Model (Case 2)



From Giles Wainrib

Autoencoders

Autoencoders

Unsupervised model that aims to learn a small encoding of some data.

For example learning an encoding vector of a human face.

Think of the encoding vector as a representational list of features zipping up a file.

Concept: The encoding vector should be informative enough to get the original data back by decoding it.

Autoencoders (cont.)

$$f_{w,b}(x) = x$$

The goal is to learn a function that maps from the input to an output equal to the input.

Forcing the dimensions to reduce, the model is forced to learn representational features.

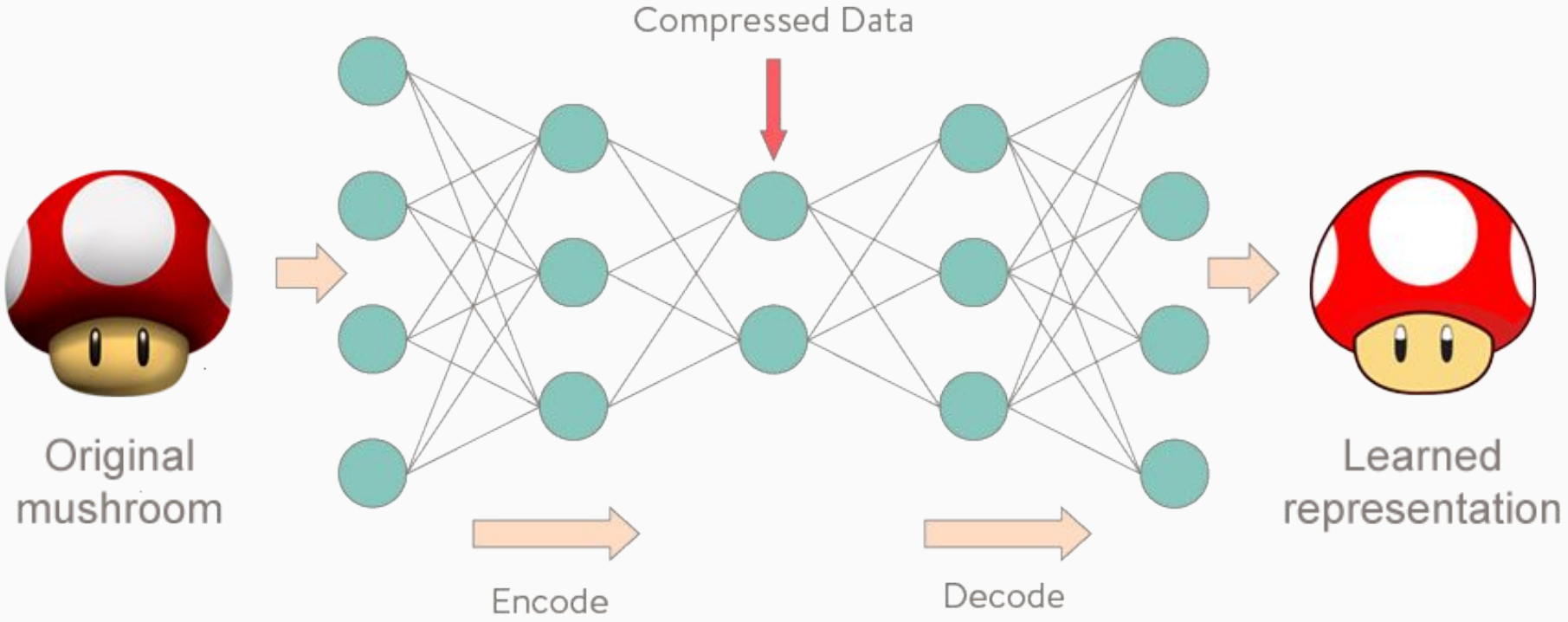
$$L(x,x') = || x - x' ||^2$$

The loss function typically used to train these is simply the distance from the original data.

The intermediate layer in the middle is our feature set (aka encoding vector).

We can find the feature vectors of multiple data points such as images and use them to compare properties of each other (similarity, etc.)

Autoencoders (cont.)



From Curiosily.

Cosine Similarity

Comparing the similarity between two vectors.

We can use this in many ways (e.g. how similar two words, are two faces the same person).

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

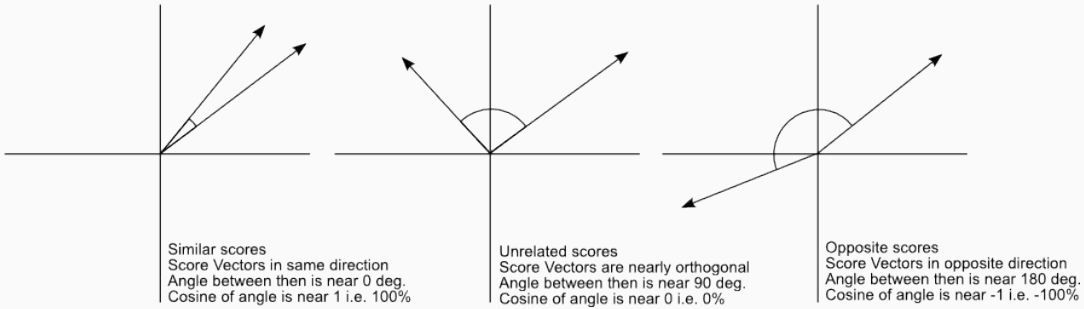


Image Captioning

If we have a vector encoding everything in an image, can we use it to understand and even caption the image?

We can do this based off of a encoder-decoder framework.

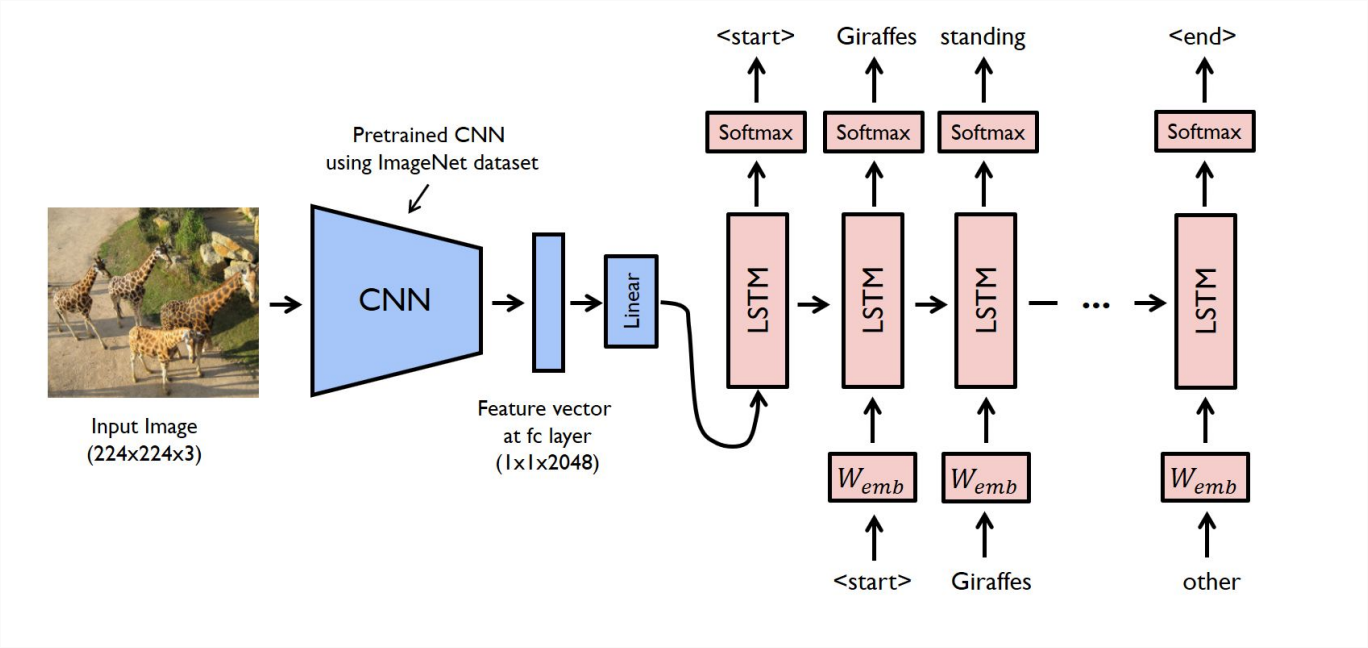
However the output is a sequence of words that should describe the original image.

Encoder: A convolutional neural network that extracts the feature representation.

Decoder: A LSTM language model conditioned on the feature vector.

Minimize the ambiguity between the caption and what is in the image.

Image Captioning



From PyTorch.

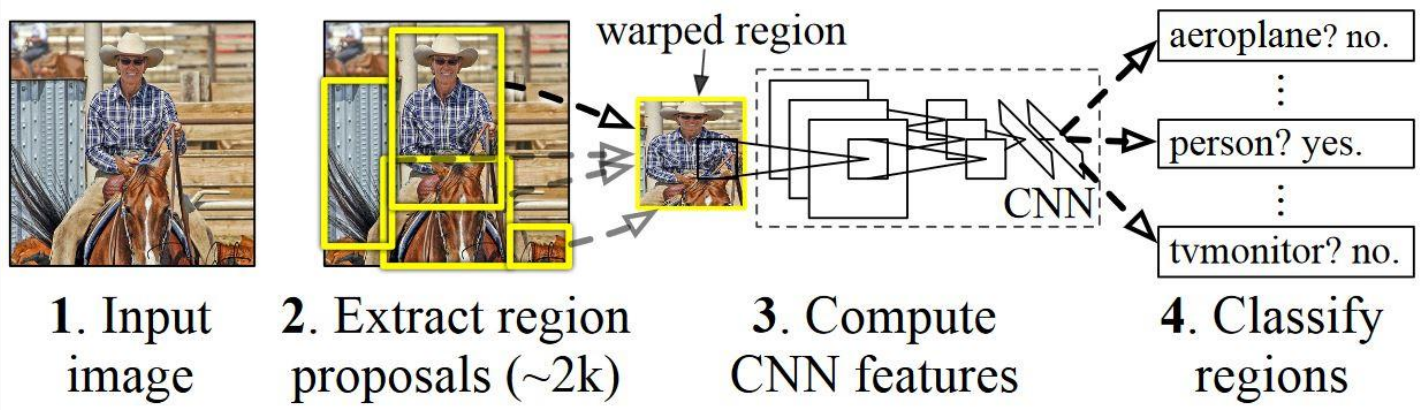
R-CNN Model

R-CNN Model

Regions with Convolutional Neural Network Features.

Two Main Parts:

1. Selective Search (finding potential objects in an image)
2. Object Classification (for each potential object, classify what it is)



From Microsoft.

Siamese Networks

Siamese Networks

Goal is to model relationships between two inputs.

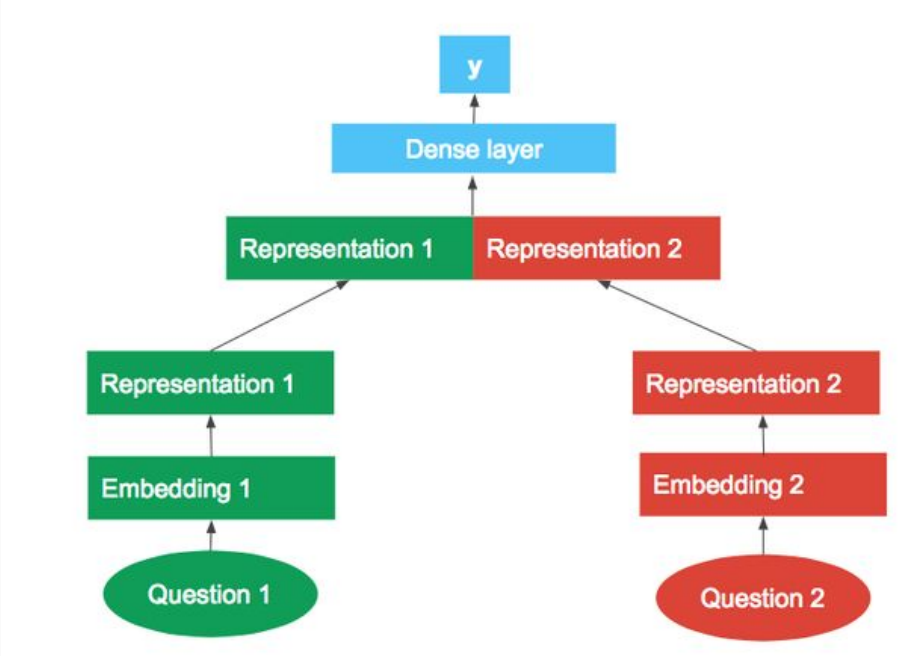
Consists of two identical neural networks which each take in their own input.

Goal is to differentiate between two images rather than classify them.

Intuitively to decide whether two images are from the same class or not.

Useful for one shot classification (e.g. learning a class based off a single example).

Siamese Networks



From Quora Engineering.

Seq-Seq Models

Seq-Seq Models

Goal is to convert one sequence to another sequence. (e.g. english to french)

Input and output sequences have different lengths and the entire input sequence is required in order to start predicting the target.

Based off of an encoder-decoder framework.

Seq-Seq Models

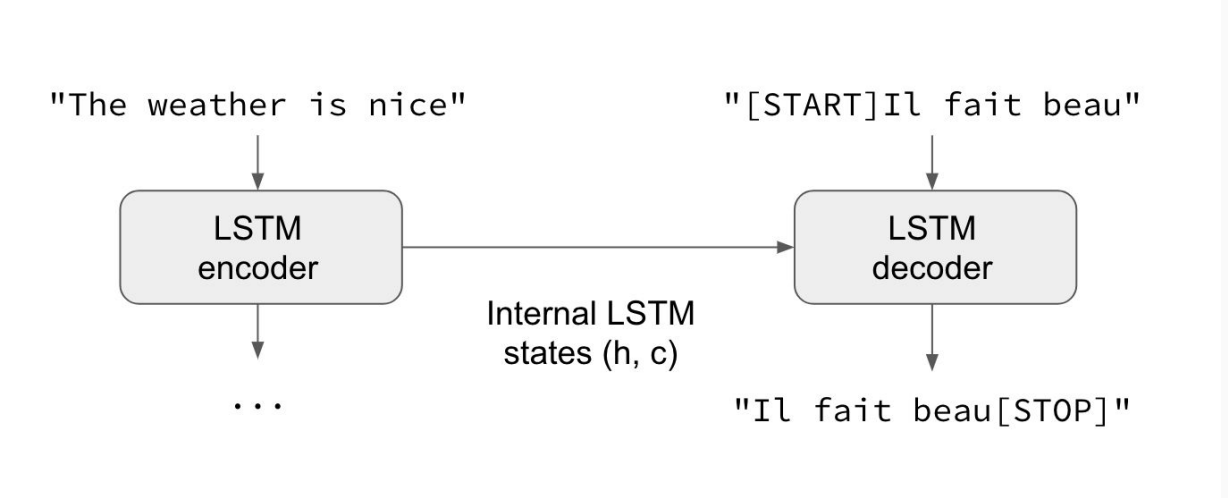
Encoder:

- RNN layer that processes the input sequence and returns its own internal state.
- Note that we discard the output and only care about the state (gives us context).

Decoder:

- RNN layer trained to predict next characters of the target sequence, given previous characters of the target sequence.
- The decoder learns to generate targets ***[t+1...]*** given targets ***[...t]***, conditioned on the input sequence.

Seq-Seq Models



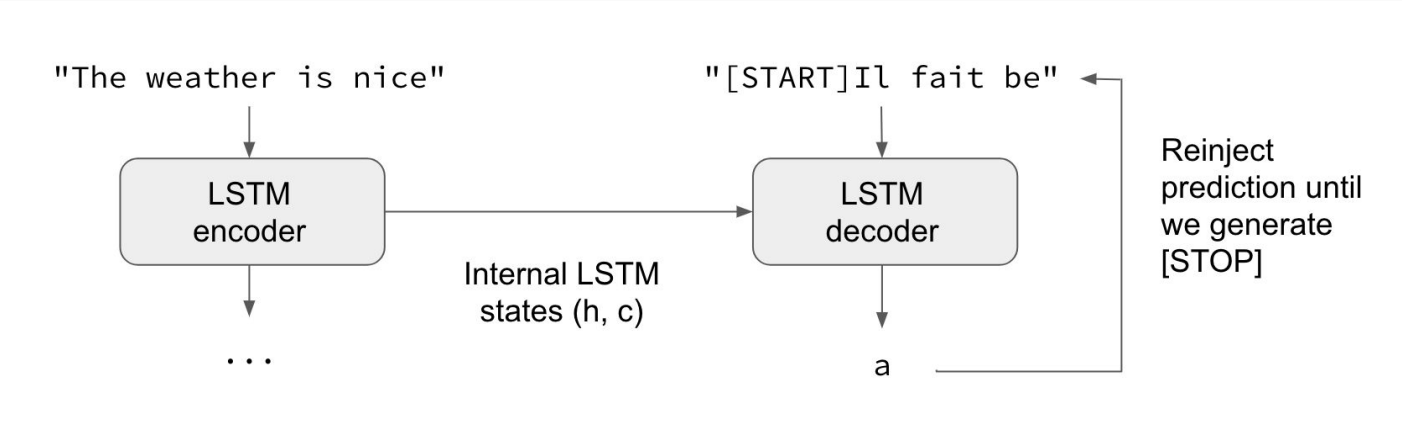
From Keras Blog.

Seq-Seq Models

However for unknown input sequences, we have to use inference process:

1. Encode the input sequence into state vectors.
2. Start with a target sequence of size 1 (just the start-of-sequence character).
3. Feed the state vectors and 1-char target sequence to the decoder to produce predictions for the next character.
4. Sample the next character using these predictions (we simply use argmax).
5. Append the sampled character to the target sequence
6. Repeat until we generate the end-of-sequence character or we hit the character limit.

Seq-Seq Models



From Keras Blog.

Generative Adversarial Networks

Generative Adversarial Network

Commonly referred to as GAN.

Network learns to model any distribution of data and create new examples that don't exist in the original data. (e.g. draw new images, make up new music, etc.)

Discriminative models try to classify input data to a category (y given x) but generative models try to model the distribution of classes (x given y).

Learning a boundary vs a distribution.

You may have heard of Google's model which paints its own images and composes music.

Intuition

Based off of an adversarial game between two networks (generator and discriminator).

Generator network is a counterfeiter and the discriminator network is a cop.

The generator network's goal is to generate new data instances similar to training data.

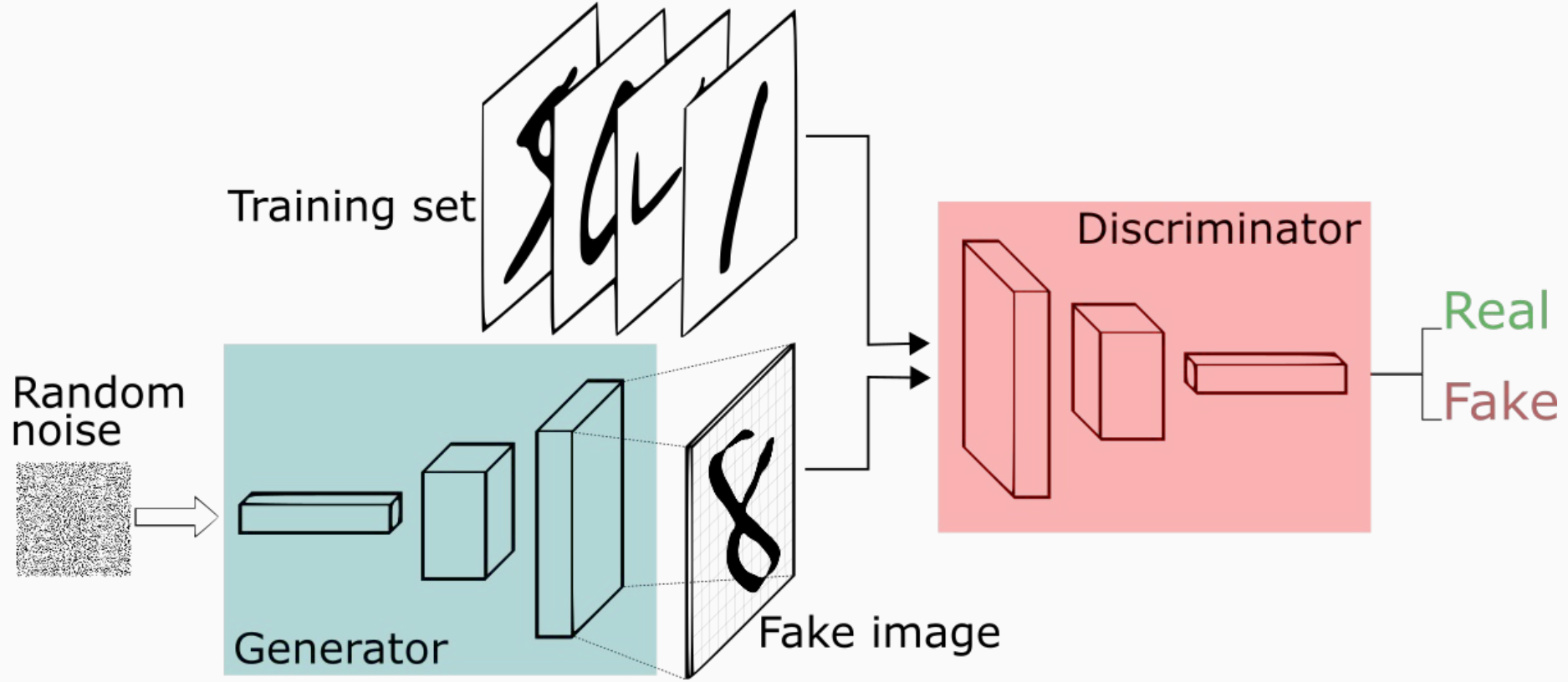
The discriminator network's goal is to decide if the example came from the generator (counterfeit) or the training data.

Optimized over time so that the discriminator is really good at catching fake data and the generator is really good at generating fake data that it doesn't get caught.

Generator takes in random noise and up-samples an image based on distribution.

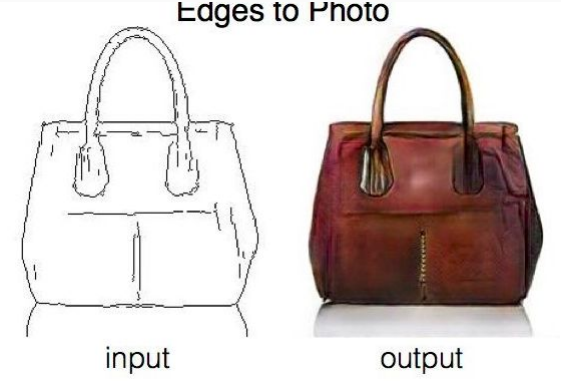
Discriminator takes in both real images and fake images.

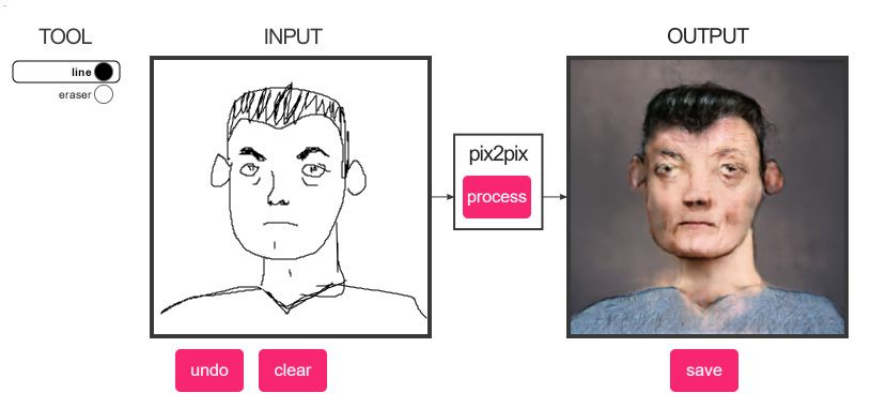
Generative Adversarial Network



From DeepLearning4J.

Examples


Edges to Photo
input
output


TOOL
line
eraser
INPUT
pix2pix
process
OUTPUT
undo
clear
save

Announcements

Announcements

Final project proposals due **April 8th**.

Practical 3 due **April 8th**.

Midterm will take place on **April 13th** and cover everything up till today.

Please submit weekly feedbacks.

Questions?