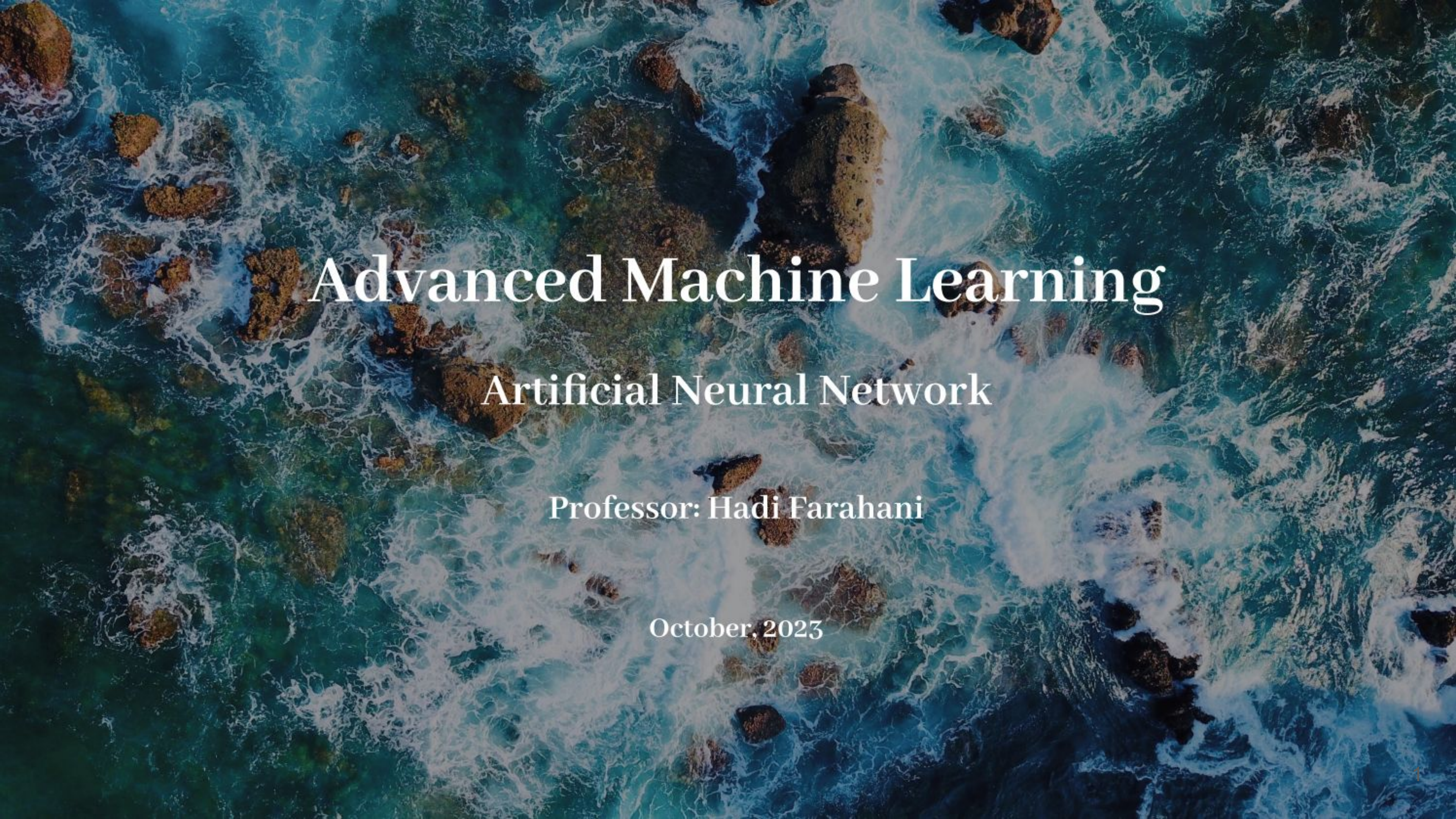


Advanced Machine Learning

Artificial Neural Network

Professor: Hadi Farahani

October, 2023

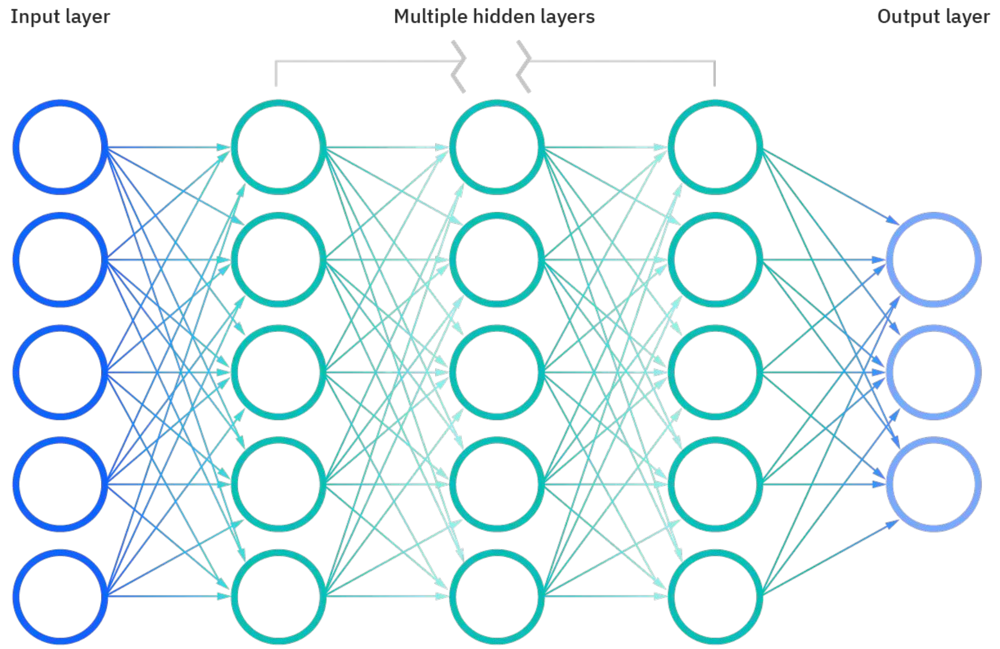
Content

In this module, we'll be covering the following topics:

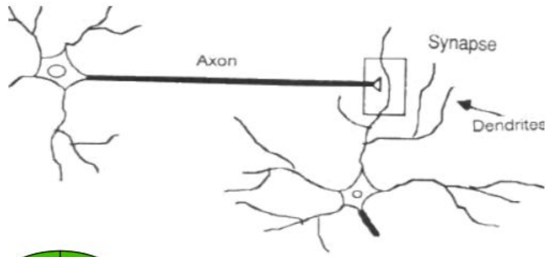
- What is neural network?
- Different variant of neural network
- Autoencoders
- Variational Autoencoders
- GANs

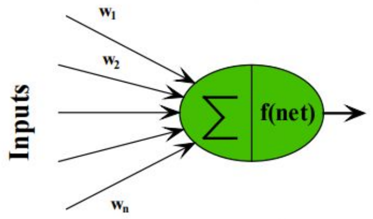
Neural Network

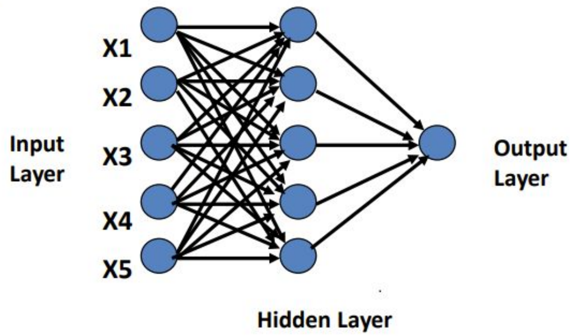
A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain.



Neural Network







Learning Process in Neural Network

- **Forward pass:**

  Compute the output of the network given the input data. Forward propagation (or forward pass) refers to the calculation and storage of intermediate variables (including outputs) for a neural network in order from the input layer to the output layer.

- **Backward pass:**

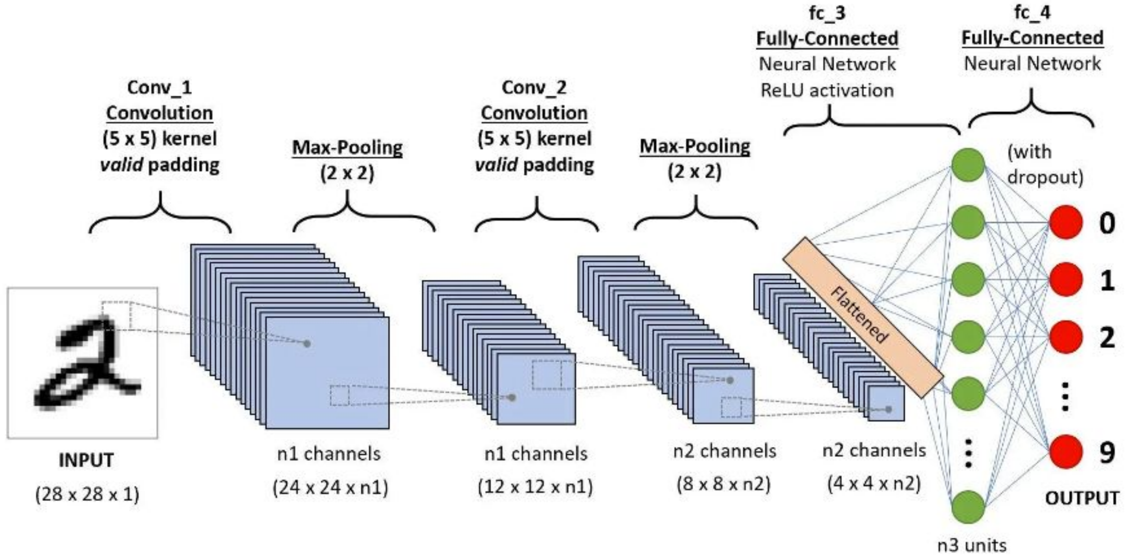
  Refers to process of counting changes in weights, using gradient descent algorithm. Computation is made from last layer, backward to the first layer.

Different variant of neural network

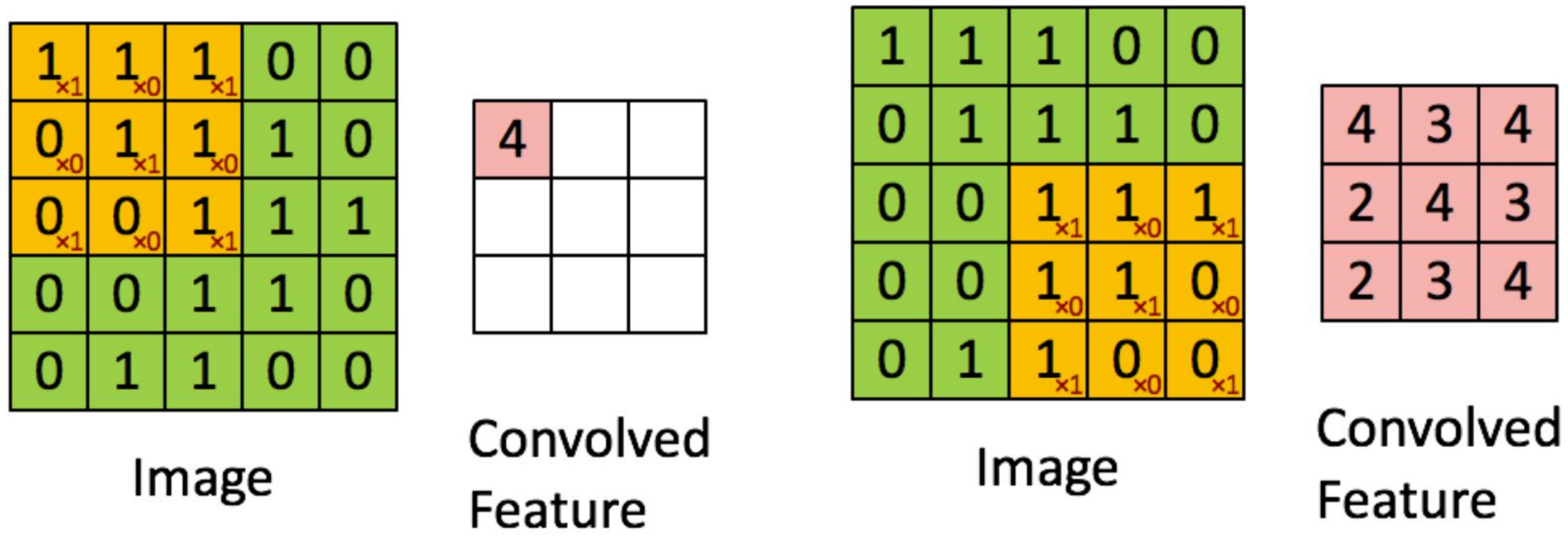
- **Convolutional Neural Network**
  The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.

- **Recurrent Neural Network**
  Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words.

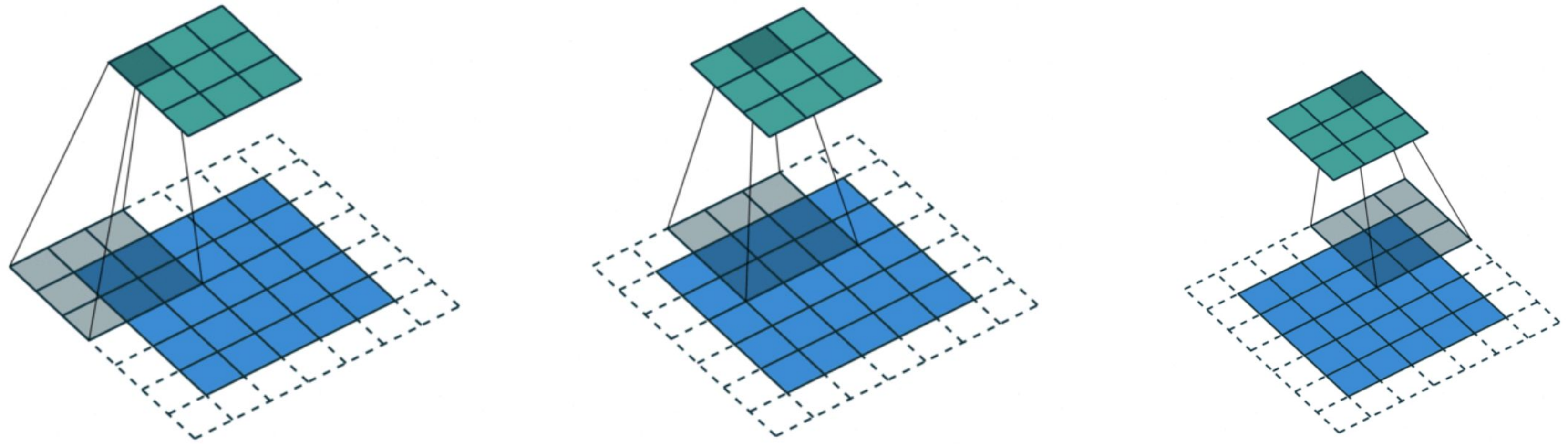
Convolutional Neural Network



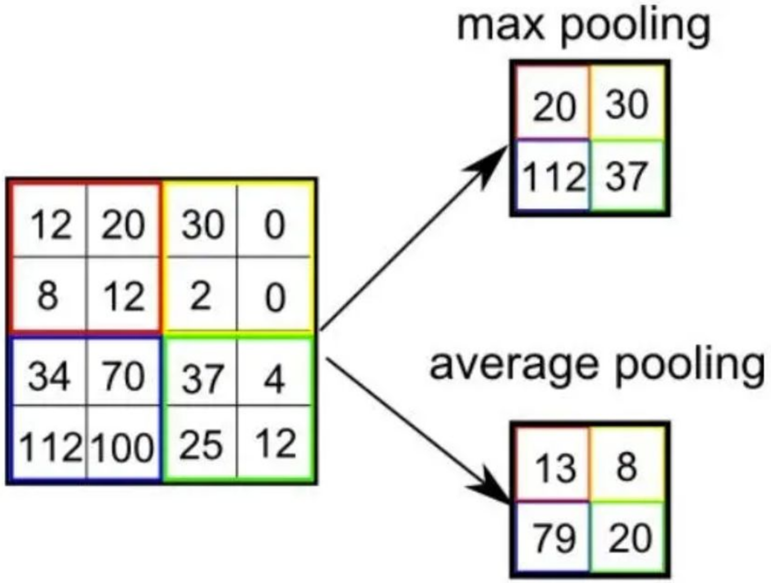
Convolutional Neural Network



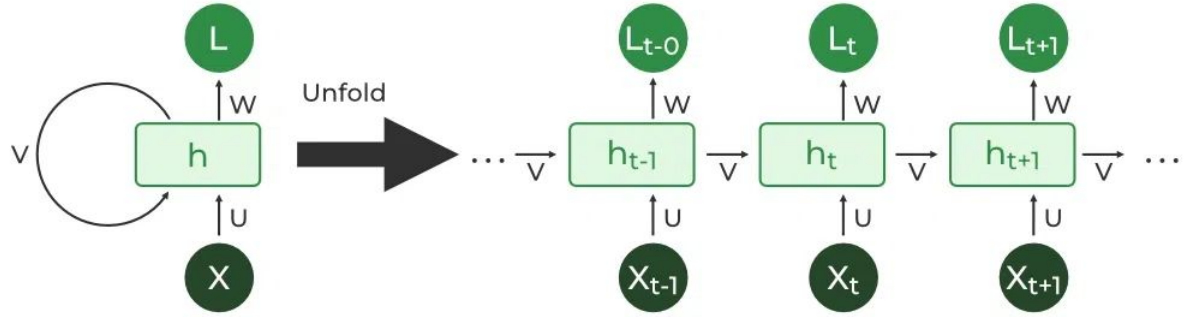
Convolutional Neural Network



Convolutional Neural Network



Recurrent Neural Network

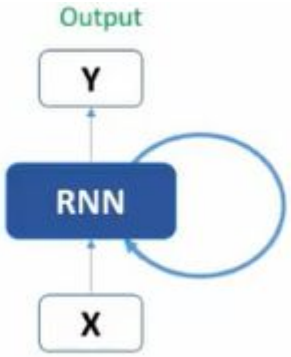


Recurrent Neural Network

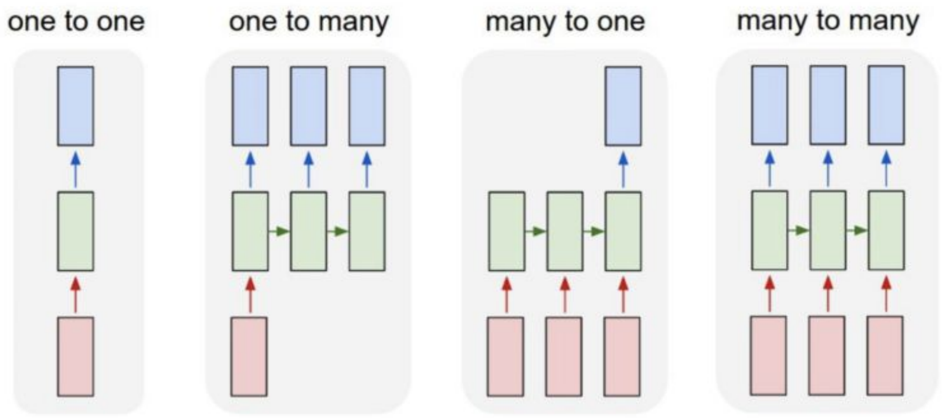
$$h_t = f(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

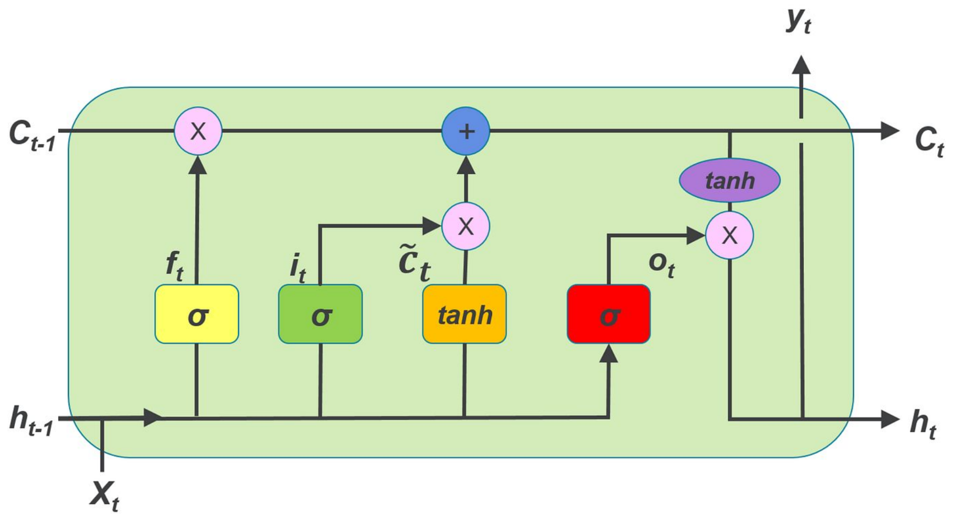


Recurrent Neural Network

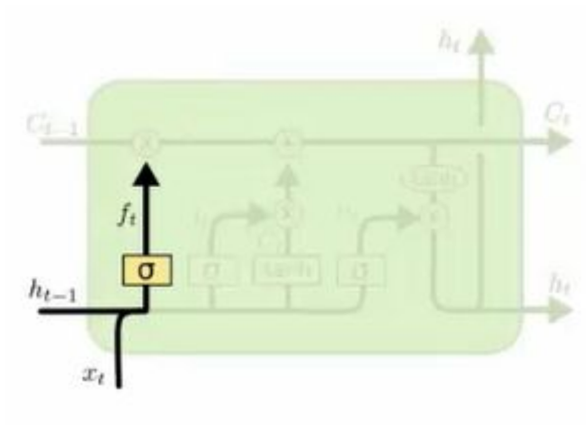


LSTM

Long Short Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. LSTM was designed by Hochreiter & Schmidhuber. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying on the basis of time-series data.

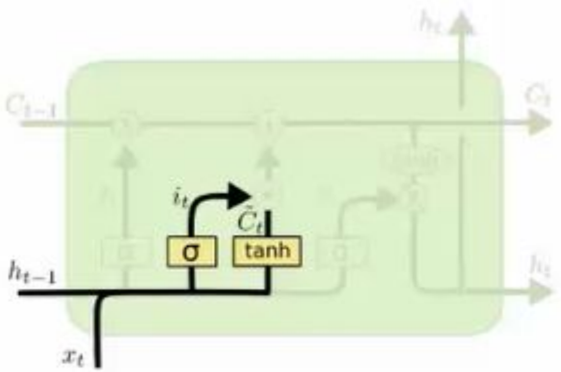


LSTM



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

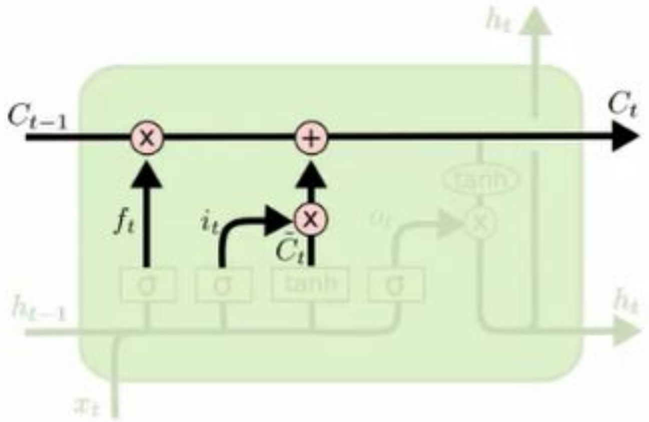
LSTM



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

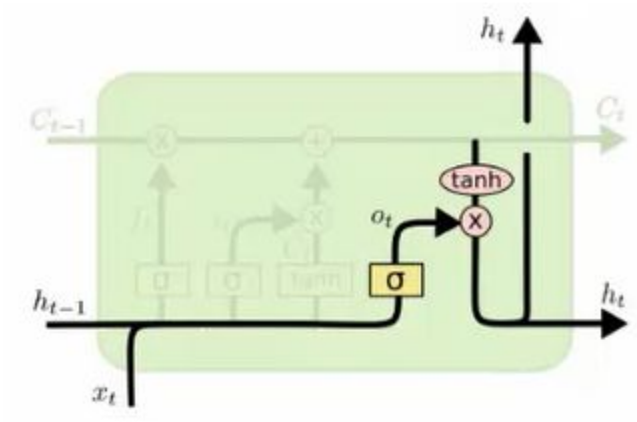
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM



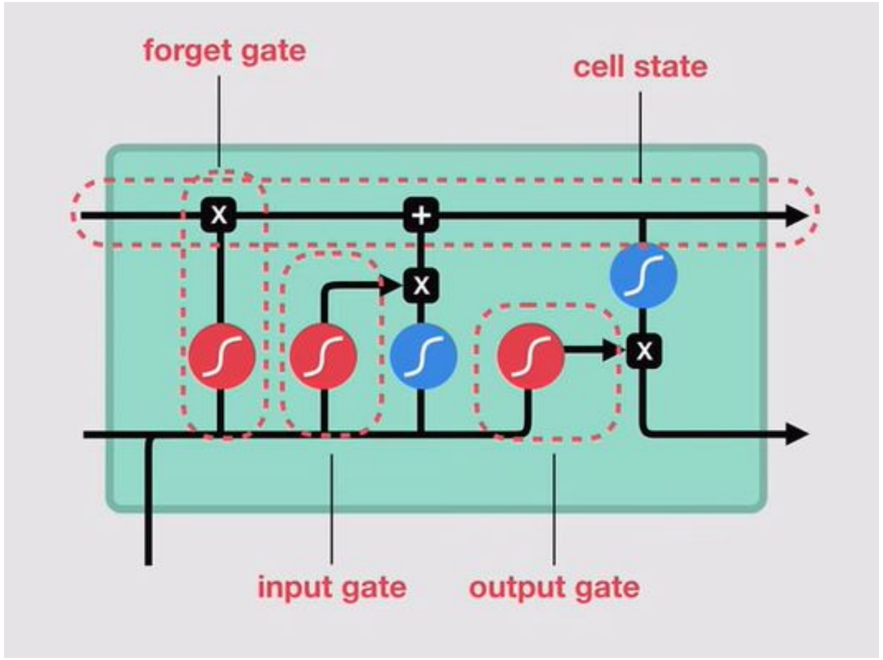
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM



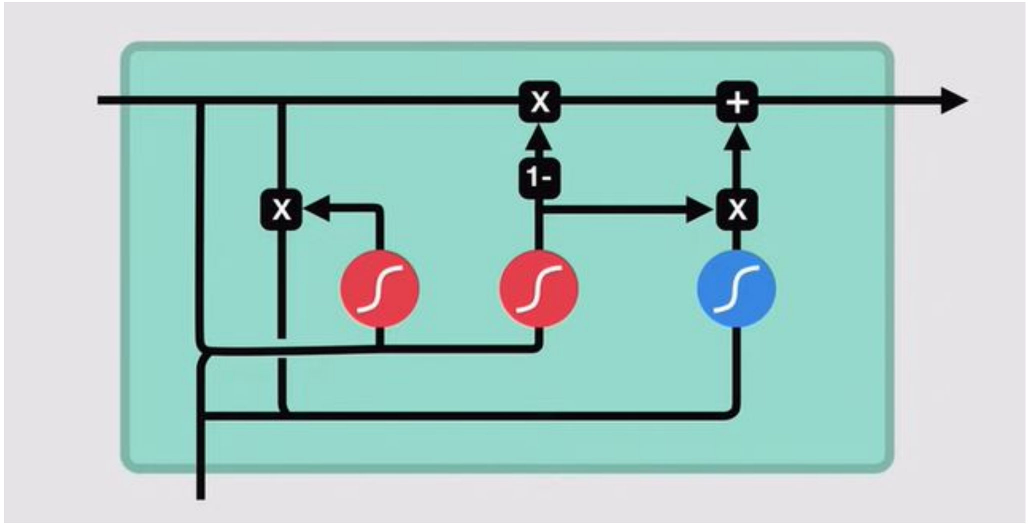
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM



GRU

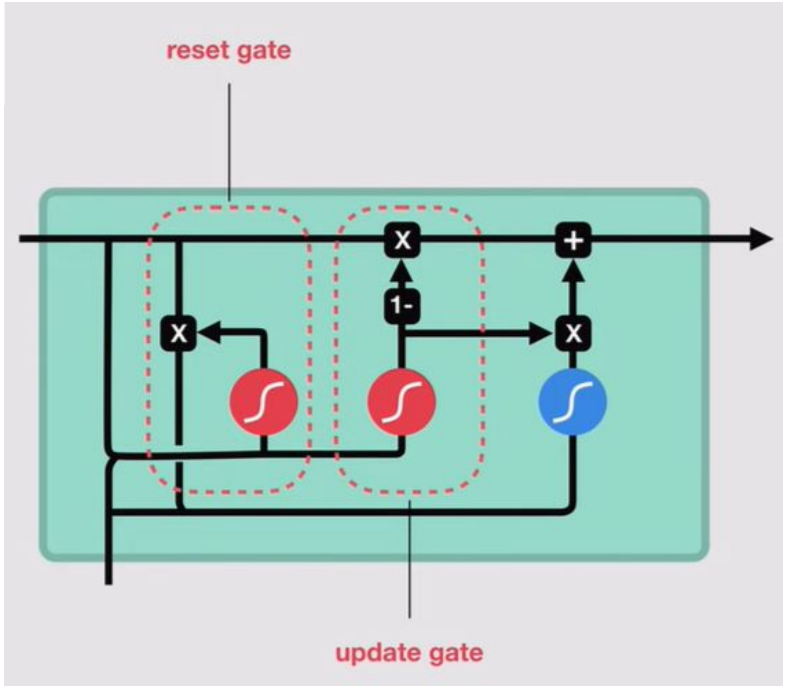
To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, update gate and reset gate.



GRU

$$gate_{reset} = \sigma(W_{input_{reset}} \cdot x_t + W_{hidden_{reset}} \cdot h_{t-1})$$

$$r = tanh(gate_{reset} \odot (W_{h_1} \cdot h_{t-1}) + W_{x_1} \cdot x_t)$$

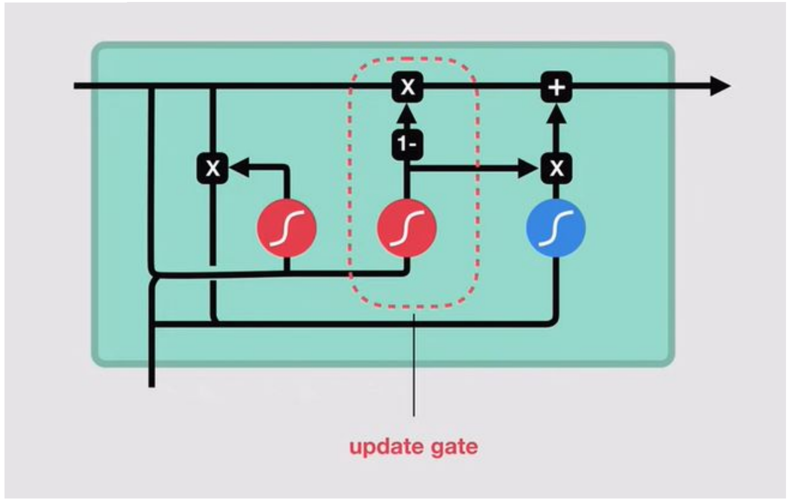


GRU

$$gate_{update} = \sigma(W_{input_{update}} \cdot x_t + W_{hidden_{update}} \cdot h_{t-1})$$

$$u = gate_{update} \odot h_{t-1}$$

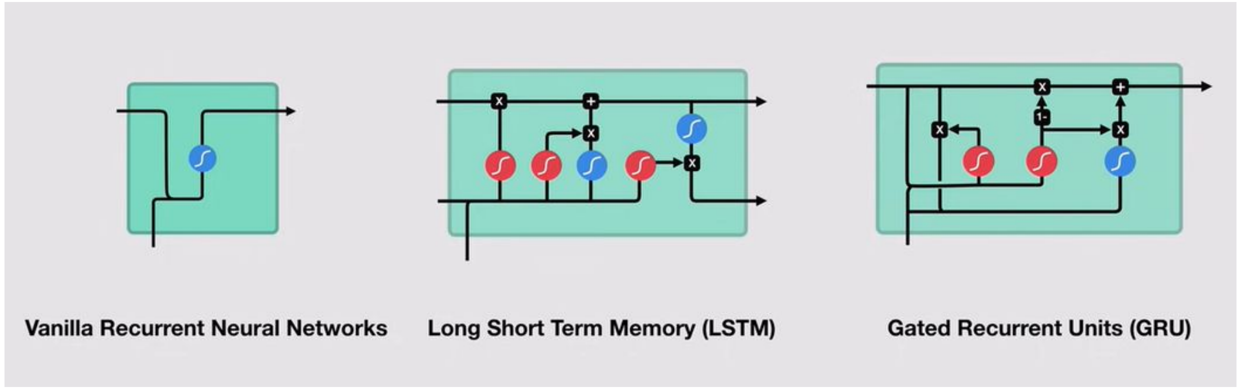
$$h_t = r \odot (1 - gate_{update}) + u$$



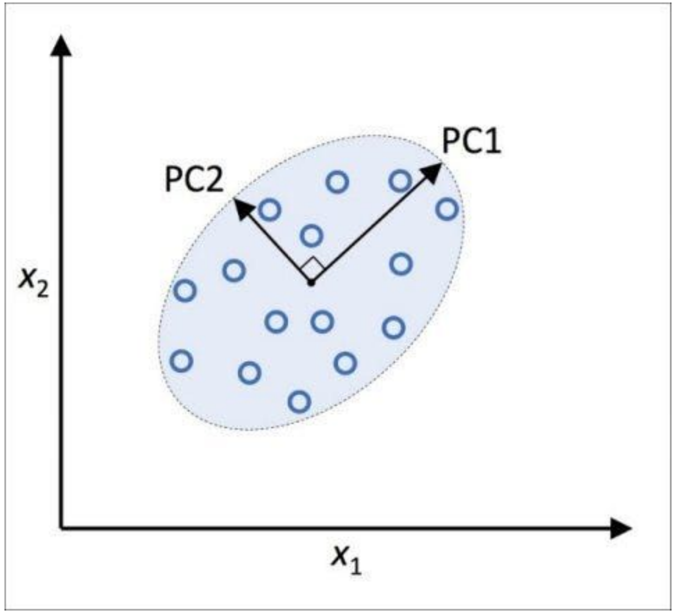
GRU

- As we’ve seen in the mechanisms above, the *Reset gate* is responsible for deciding which portions of the previous hidden state are to be combined with the current input to propose a new hidden state.
- And the *Update gate* is responsible for determining how much of the previous hidden state is to be retained and what portion of the new proposed hidden state (derived from the *Reset gate*) is to be added to the final hidden state. When the *Update gate* is first multiplied with the previous hidden state, the network is picking which parts of the previous hidden state it is going to keep in its memory while discarding the rest. Subsequently, it is patching up the missing parts of information when it uses the inverse of the *Update gate* to filter the proposed new hidden state from the *Reset gate*.

GRU


1-
Vanilla Recurrent Neural Networks
Long Short Term Memory (LSTM)
Gated Recurrent Units (GRU)

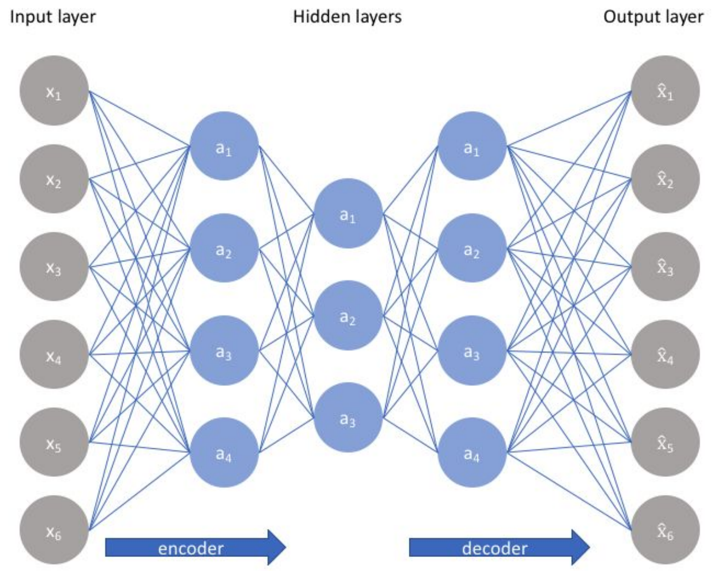
Autoencoders



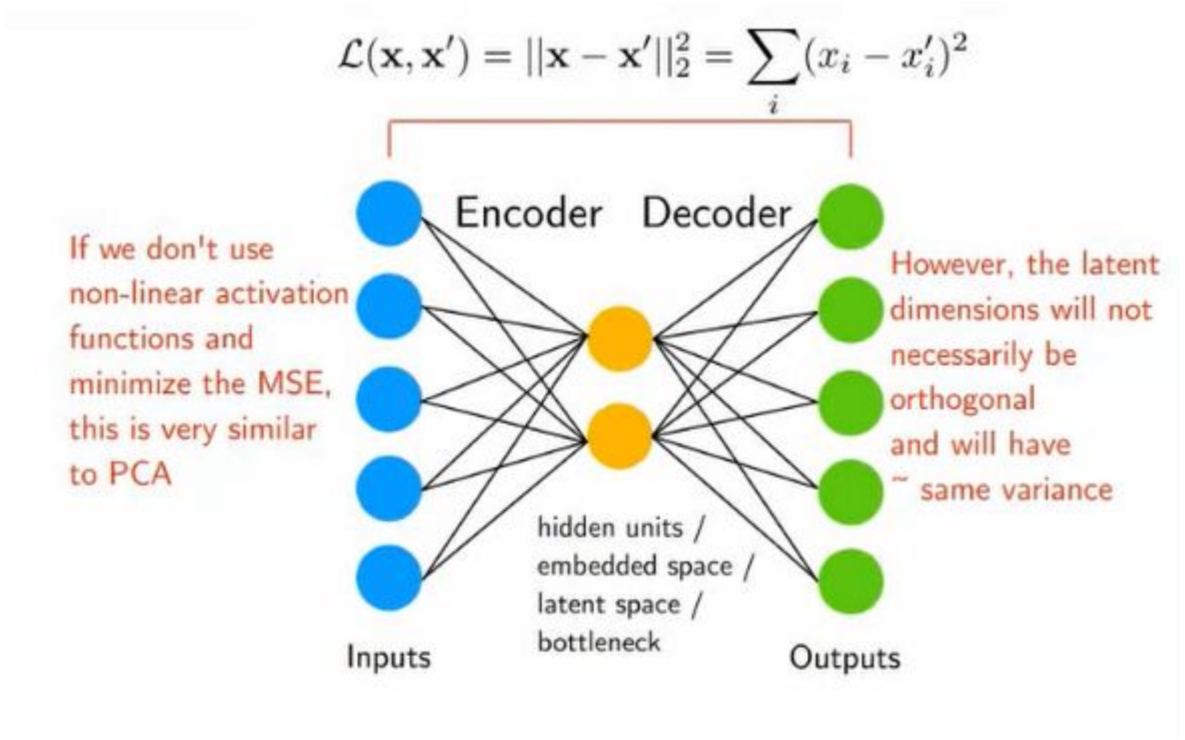
Autoencoders

Autoencoders are an unsupervised learning technique in which we leverage neural networks for the task of representation learning.

$$\text{reconstruction loss} = \|x - \hat{x}\|_2$$



Autoencoders


$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_2^2 = \sum_i (x_i - x_i')^2$$

If we don't use non-linear activation functions and minimize the MSE, this is very similar to PCA

However, the latent dimensions will not necessarily be orthogonal and will have ~ same variance

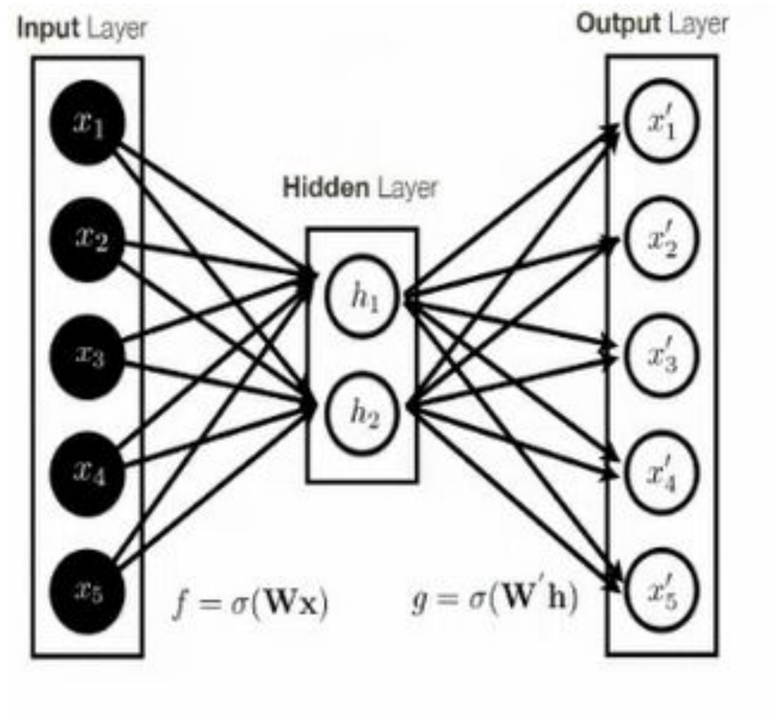
Autoencoders

- Training autoencoder using untied weight

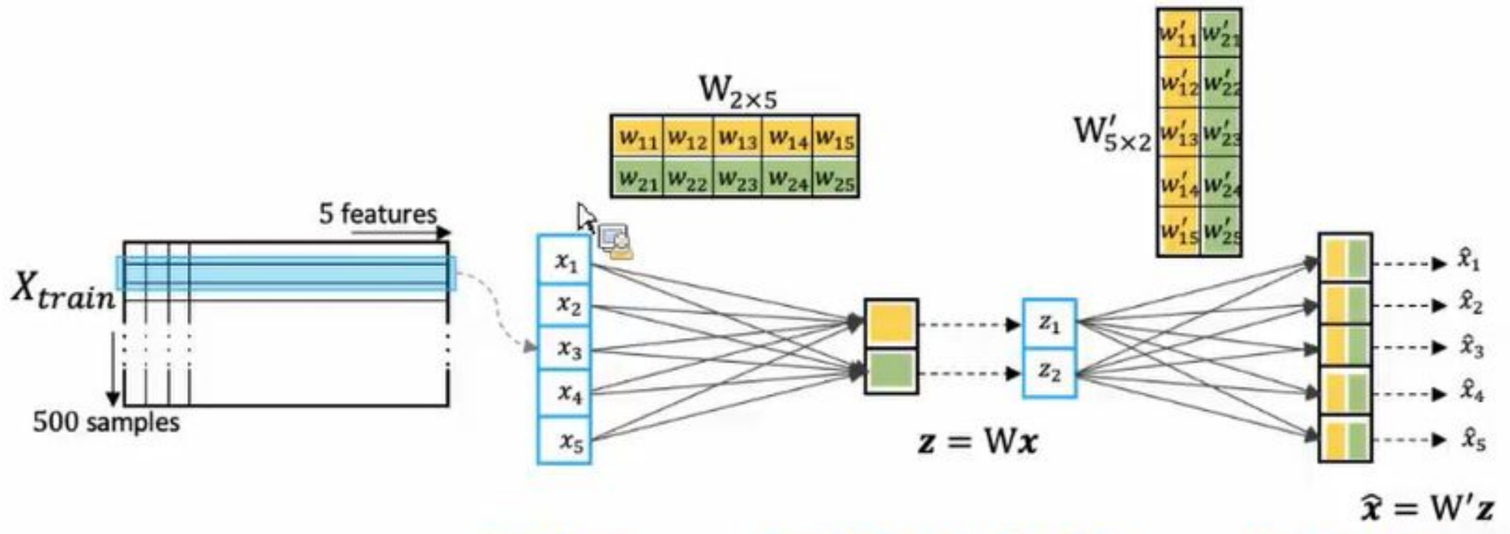
$$f(x)=\sigma_2(\mathbf{b_2}+\mathbf{W_2}*\sigma_1(\mathbf{b_1}+\mathbf{W_1}*x))$$

- Training autoencoder using tied weight

$$f(x)=\sigma_2(\mathbf{b_2}+\mathbf{W_1}^{T}*\sigma_1(\mathbf{b_1}+\mathbf{W_1}*x))$$



Autoencoders


$W_{2\times5}$
w_{11} w_{12} w_{13} w_{14} w_{15}
w_{21} w_{22} w_{23} w_{24} w_{25}
$W'_{5\times2}$
w'_{11} w'_{21}
w'_{12} w'_{22}
w'_{13} w'_{23}
w'_{14} w'_{24}
w'_{15} w'_{25}
5 features
X_{train}
500 samples
x_1 x_2 x_3 x_4 x_5
z_1 z_2
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$
$\boldsymbol{z} = \mathrm{W}\boldsymbol{x}$
$\hat{\boldsymbol{x}} = \mathrm{W}'\boldsymbol{z}$

Required Constraints.

1. Tied weights, $W' = W^T$.
2. Orthogonal weights, $W^T W = I$.
3. Uncorrelated Encodings, $\mathrm{cor}(z_i, z_j) = 0$, if $i \neq j$.
4. Weights are Unit Norm, $\sum_{j=1}^{p} w_{ij}^2 = 1, i = 1, \ldots, k$.

Input Layer.
Size 5 x 1.

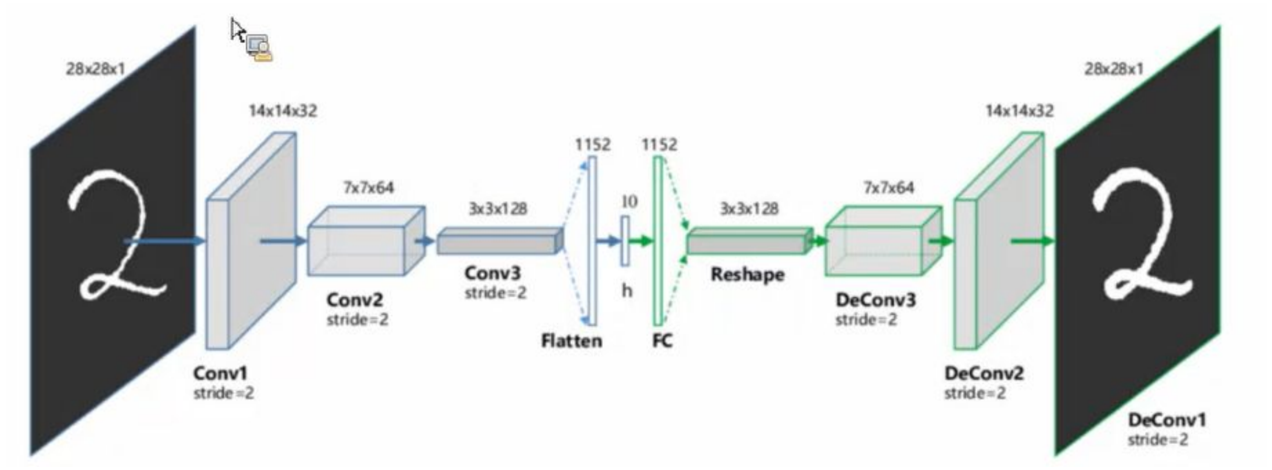
Encoding Layer.
Input 5 x 1.
Output 2 x 1.
Weights $W_{2\times5}$

Encodings,
output of the
Encoding Layer.
Size 2 x 1.

Decoding Layer.
Input 2 x 1.
Output 5 x 1.
Weights $W'_{5\times2}$

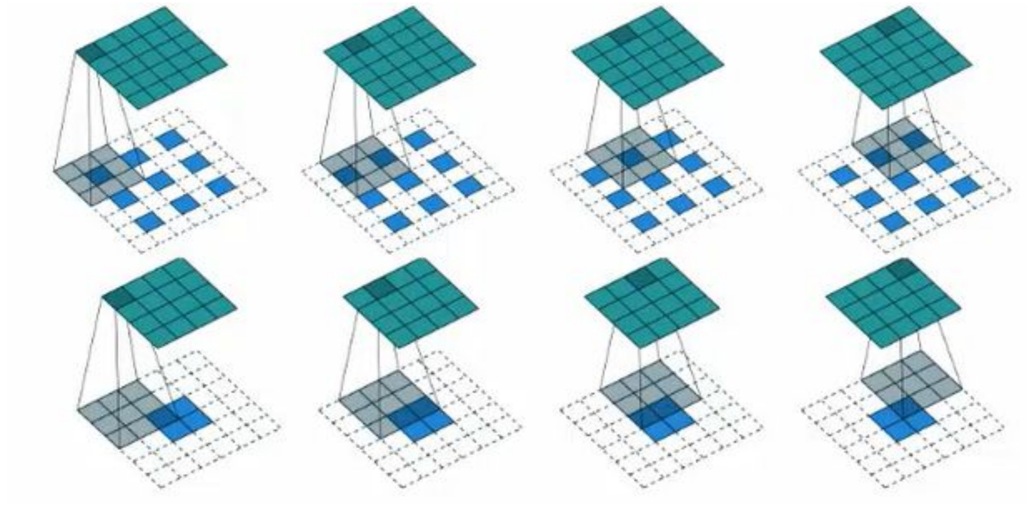
Autoencoders

Convolutional Autoencoders



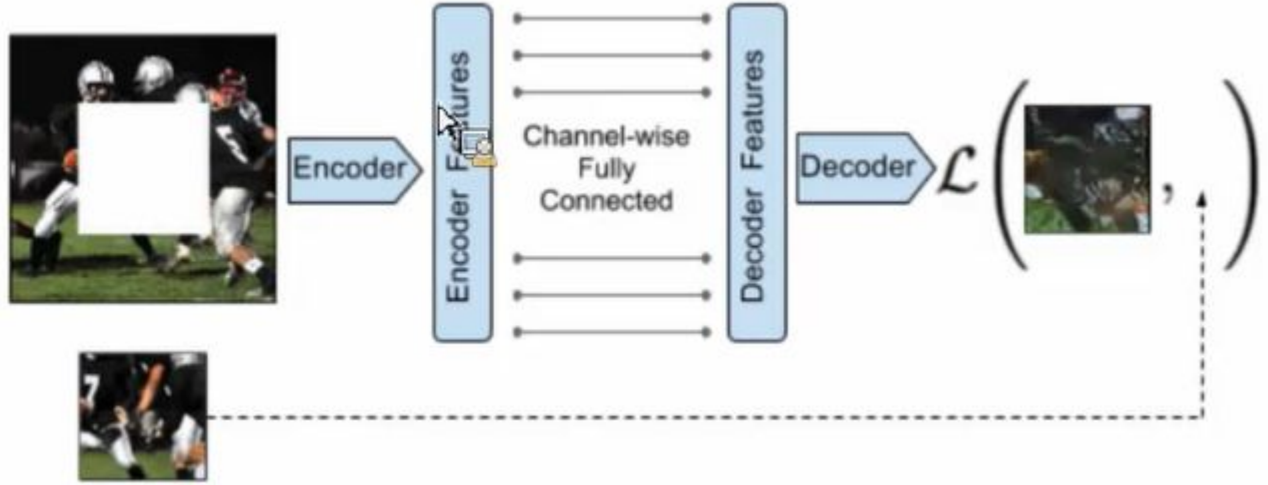
Autoencoders

Upsampling + padding

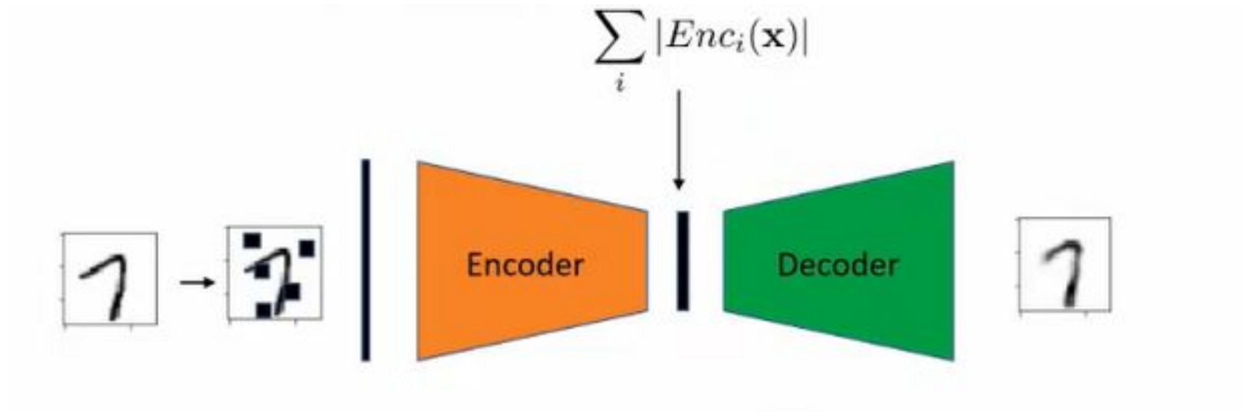


Autoencoders

Context Encoder

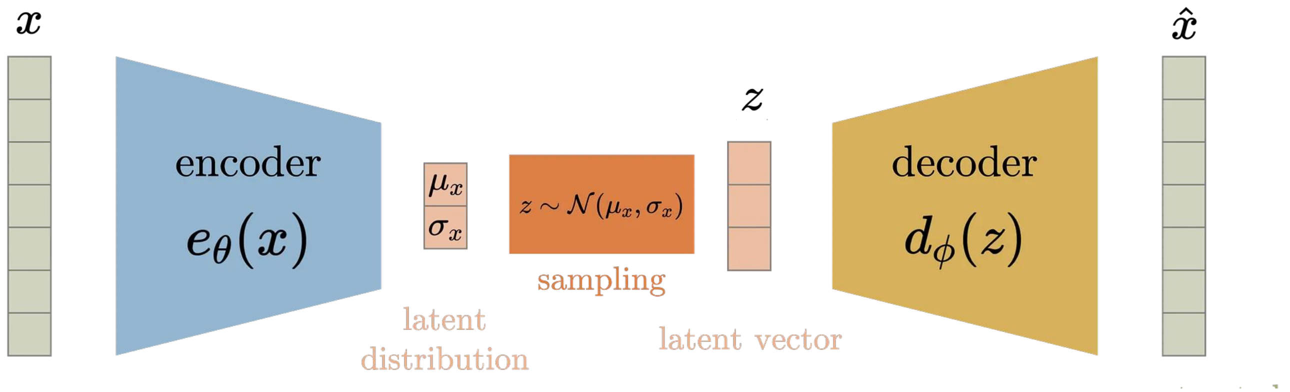


Autoencoders

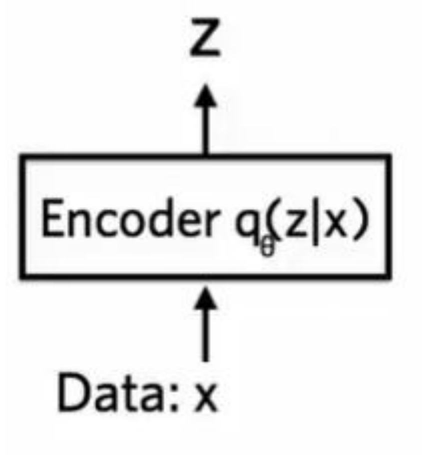


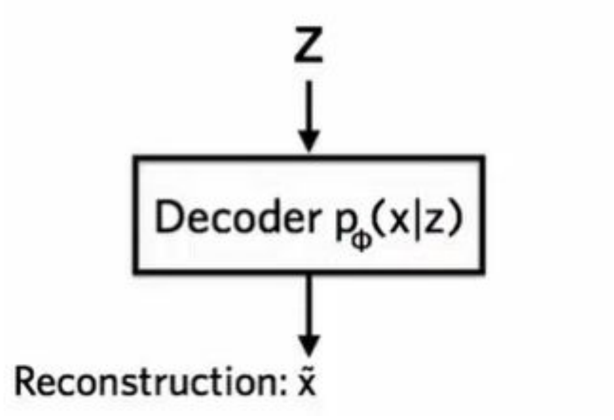
Variational Autoencoders

Variational autoencoder was proposed in 2013 by Knigma and Welling at Google and Qualcomm. A variational autoencoder (VAE) provides a probabilistic manner for describing an observation in latent space. Thus, rather than building an encoder that outputs a single value to describe each latent state attribute, we'll formulate our encoder to describe a probability distribution for each latent attribute.



Variational Autoencoders





Variational Autoencoders

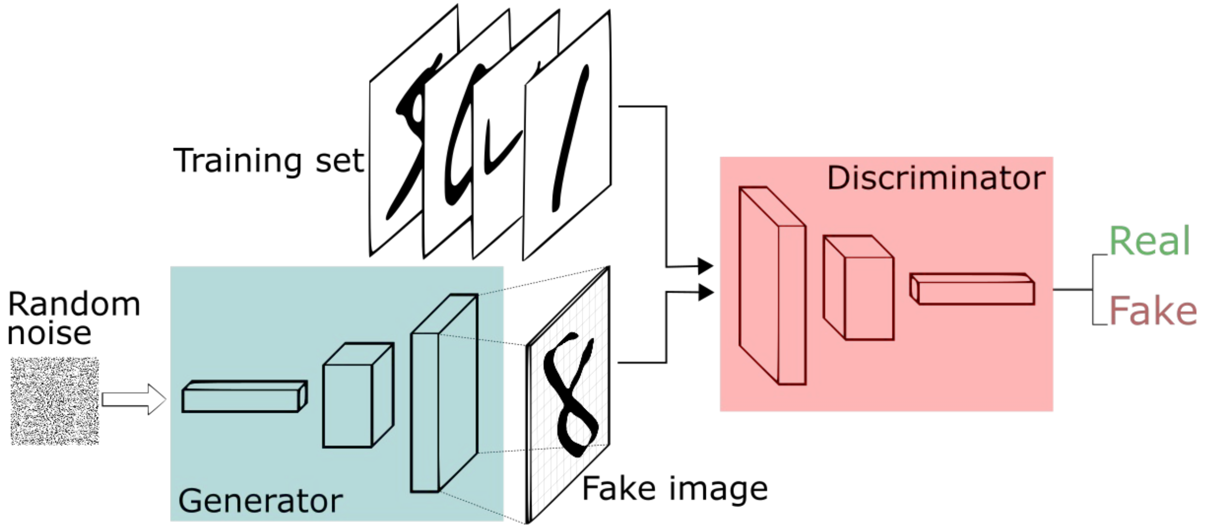
- The first term is the reconstruction loss, or expected negative log-likelihood of the *i*-th datapoint. The expectation is taken with respect to the encoder's distribution over the representations. This term encourages the decoder to learn to reconstruct the data.
- The second term is a regularizer.

$$L(\theta, \phi) = \sum_{i=1}^{N} l_i(\theta, \phi)$$

$$l_i(\theta, \phi) = -\mathbb{E}_{z\sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] + \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

GANs

In 2014, a paper on generative adversarial networks (GANs) was published by Ian Goodfellow and his colleagues. This research paper proposed a new framework for unsupervised learning, in which two neural networks are trained to compete against each other.



GANs

In simple words, the idea behind GANs can be summarized like this:

- Two Neural Networks are involved.
- One of the networks, the Generator, starts off with a random data distribution and tries to replicate a particular type of distribution.
- The other network, the Discriminator, through subsequent training, gets better at classifying a forged distribution from a real one.
- Both of these networks play a min-max game where one is trying to outsmart the other.

GANs

The generator tries to minimize this function while the discriminator tries to maximize it.

$$\min_{\theta_g} \max_{\theta_d} \underbrace{\mathbb{E}_{p_{\text{data}}} [\log D_{\theta_d}(\mathbf{x})]}_{\text{likelihood of true data}} + \underbrace{\mathbb{E}_{p_z(\mathbf{z})} \left[\log \left(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z}))\right)\right]}_{\text{likelihood of generated data}}$$

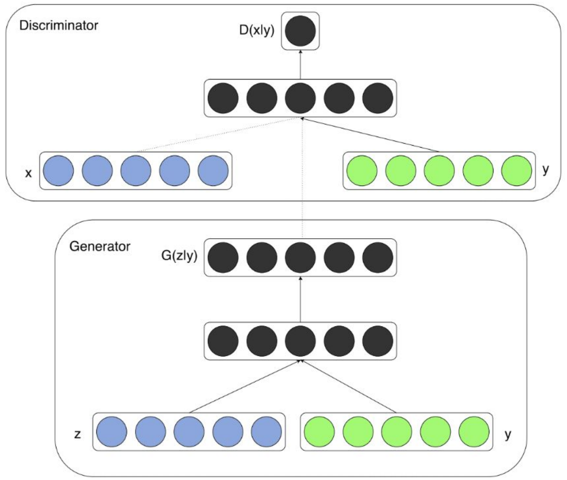
GANs

some of the most popular GANs include:

- Conditional GAN (CGAN)
- Adversarial Autoencoder (AAE)
- Dual GAN (DGAN)
- Stack GAN (StackGAN)
- Cycle GAN (CycleGAN)

GANs

- **The cGAN was first described by Mehdi Mirza and Simon Osindero in their 2014 paper titled "Conditional Generative Adversarial Nets."**
- Their approach is demonstrated in the MNIST handwritten digit dataset where the class labels are one hot encoded and concatenated with the input to both the generator and discriminator models.



GANs

Conditional GAN (cGAN) allows us to condition the network with additional information such as class labels. It means that during the training, we pass images to the network with their actual labels for it to learn the difference between them. That way, we gain the ability to ask our model to generate images of specific flowers.

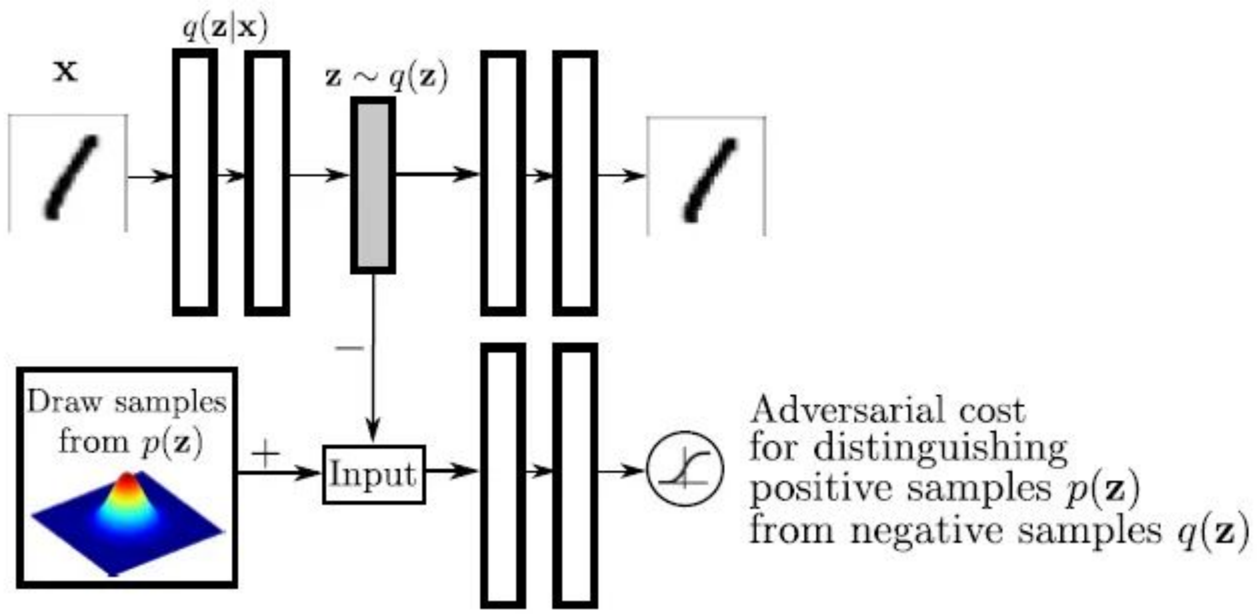
$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

GANs

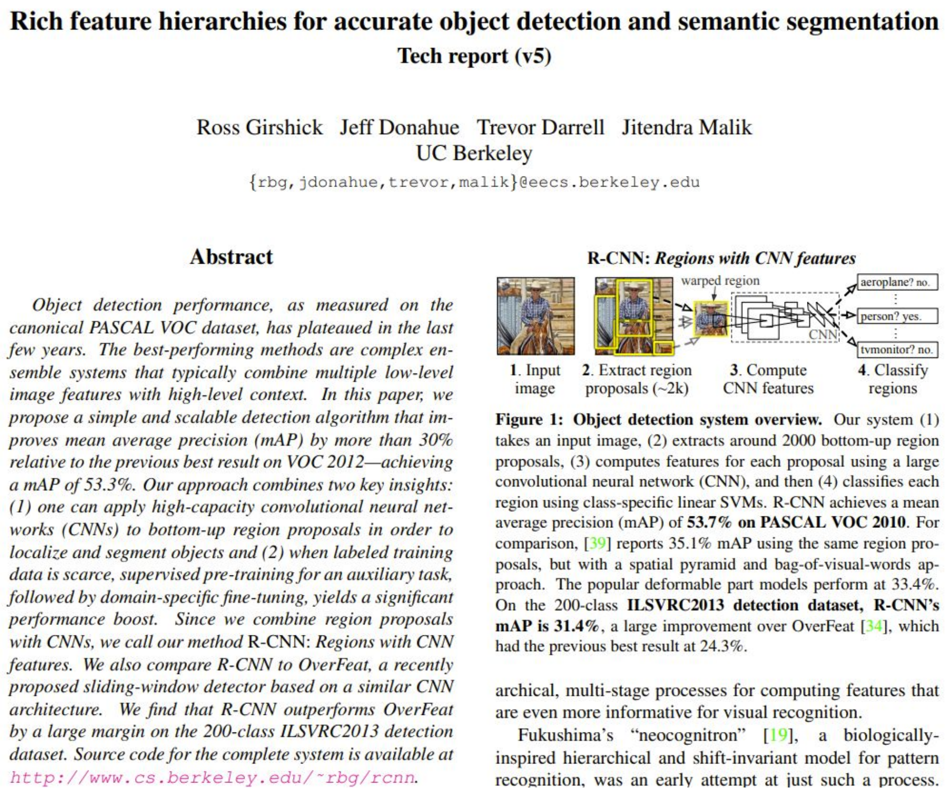
What is an adversarial autoencoder?

An adversarial autoencoder (AAE) is a deep learning model that combines the strengths of autoencoders and generative adversarial networks (GANs). Autoencoders are neural networks that learn to compress and reconstruct data, while GANs consist of two networks, a generator and a discriminator, that compete against each other to generate realistic samples from a given data distribution. AAEs use the adversarial training process from GANs to impose a specific prior distribution on the latent space of the autoencoder, resulting in a more expressive generative model.

GANs


$\mathbf{x}$
$q(\mathbf{z}|\mathbf{x})$
$\mathbf{z} \sim q(\mathbf{z})$
$-$
Draw samples from $p(\mathbf{z})$
$+$
Input
Adversarial cost for distinguishing positive samples $p(\mathbf{z})$ from negative samples $q(\mathbf{z})$

Paper



Rich feature hierarchies for accurate object detection and semantic segmentation

Tech report (v5)

Ross Girshick   Jeff Donahue   Trevor Darrell   Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

Abstract

Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, we propose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012—achieving a mAP of 53.3%. Our approach combines two key insights: (1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and (2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. Since we combine region proposals with CNNs, we call our method R-CNN: *Regions with CNN features. We also compare R-CNN to OverFeat, a recently proposed sliding-window detector based on a similar CNN architecture. We find that R-CNN outperforms OverFeat by a large margin on the 200-class ILSVRC2013 detection dataset. Source code for the complete system is available at* http://www.cs.berkeley.edu/~rbg/rcnn.

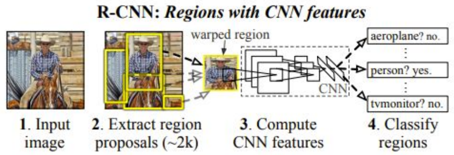


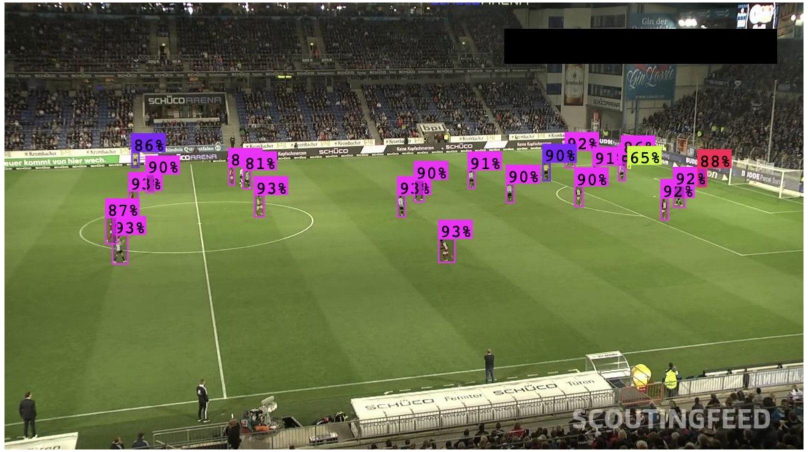
Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class **ILSVRC2013 detection dataset, R-CNN's mAP is 31.4%**, a large improvement over OverFeat [34], which had the previous best result at 24.3%.

archical, multi-stage processes for computing features that are even more informative for visual recognition.

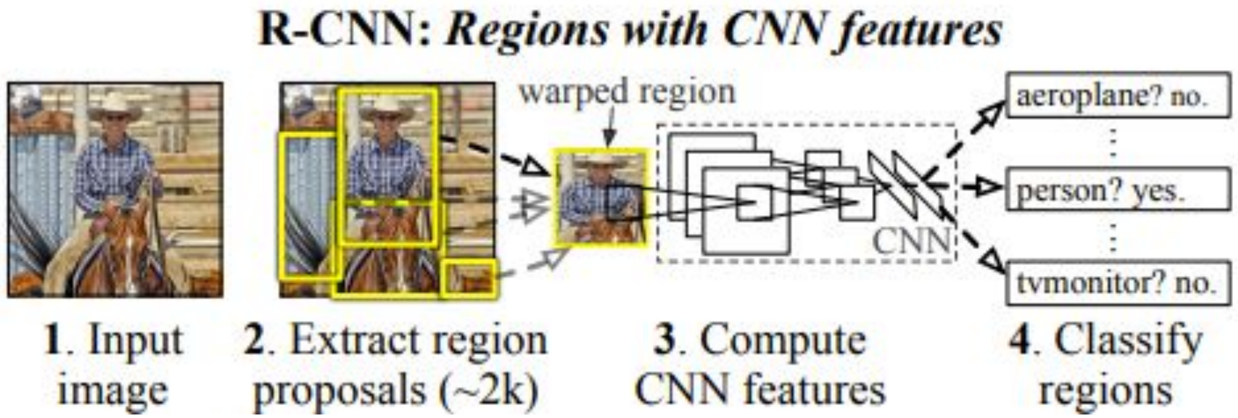
Fukushima's "neocognitron" [19], a biologically-inspired hierarchical and shift-invariant model for pattern recognition, was an early attempt at just such a process.

Paper

- Object detection is a computer vision solution that identifies objects, and their locations, in an image. An object detection system will return the coordinates of the objects in an image that it has been trained to recognize. The system will also return a confidence level, which shows how confident the system is that a prediction is accurate.



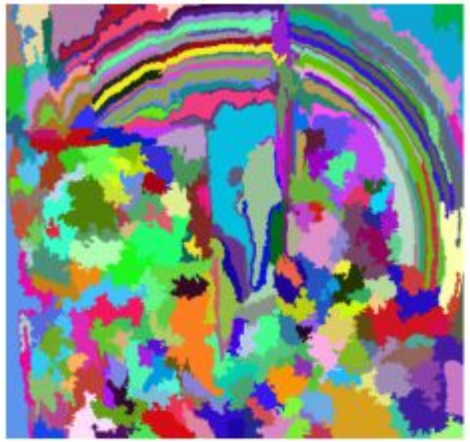
Paper


R-CNN: *Regions with CNN features*
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

Paper

Algorithm Of Selective Search :

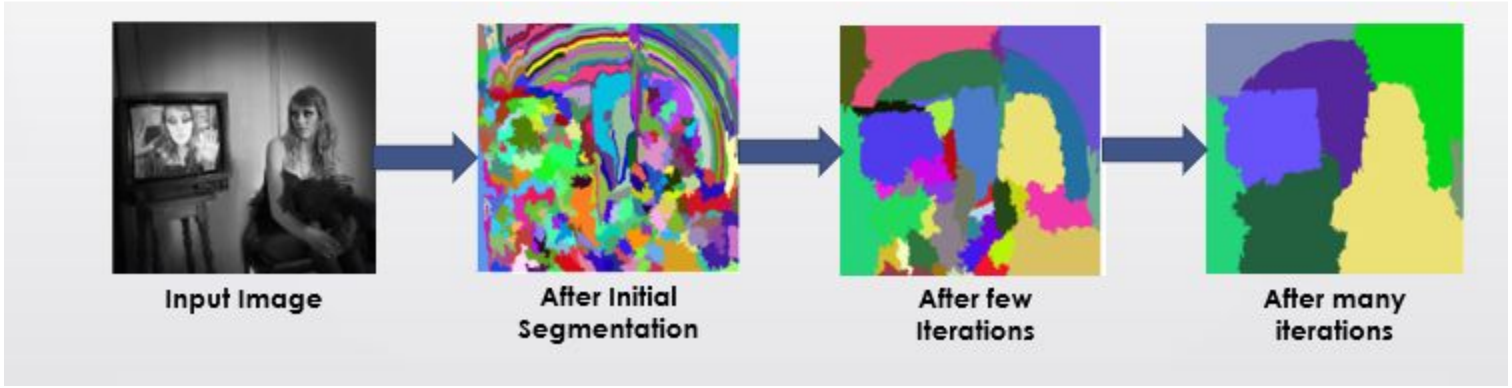
Generate initial sub-segmentation of input image using the method describe by *Felzenszwalb et al* in his paper “Efficient Graph-Based Image Segmentation “.



Paper

Algorithm Of Selective Search :

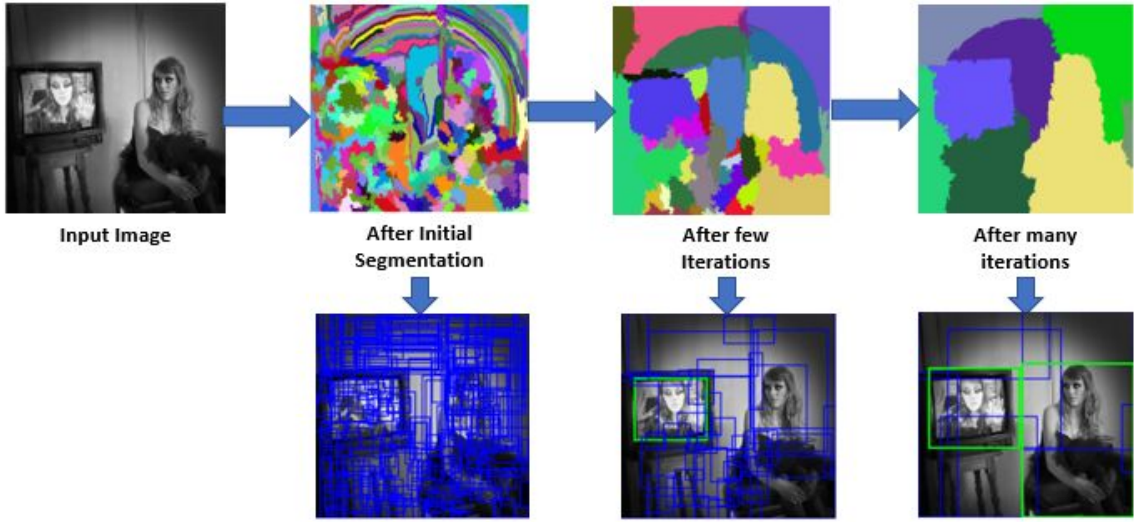
Recursively combine the smaller similar regions into larger ones.



Paper

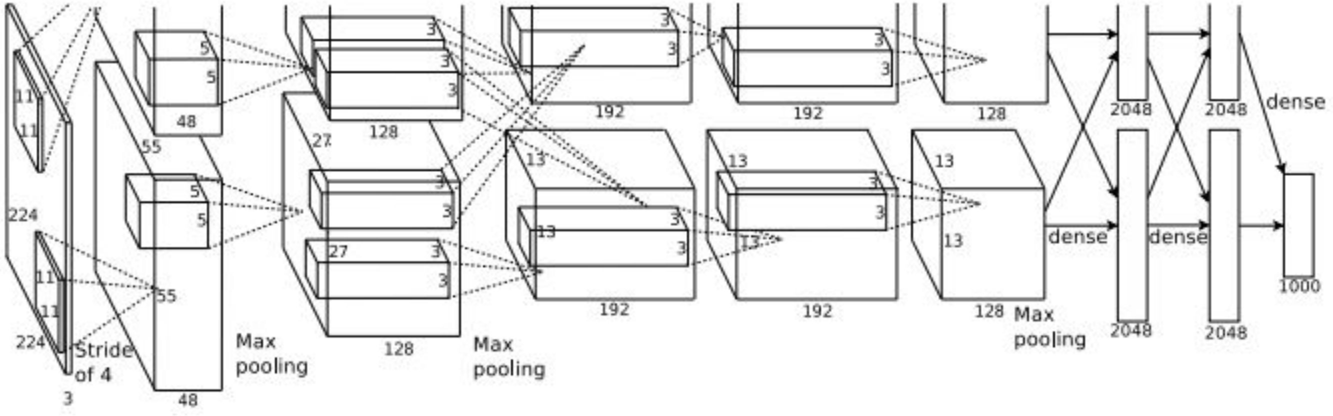
Algorithm Of Selective Search :

Use the segmented region proposals to generate candidate object locations

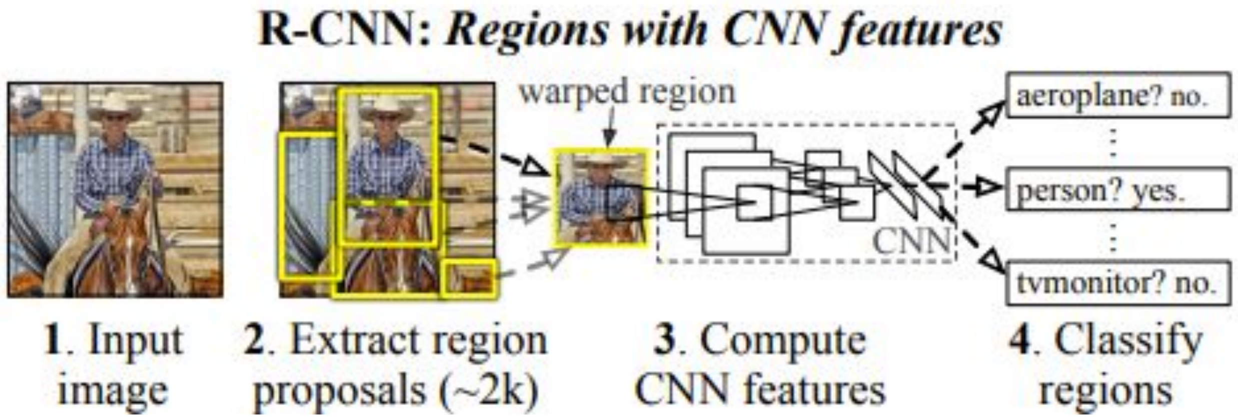


Paper

They extract a 4096-dimensional feature vector from each region proposal



Paper

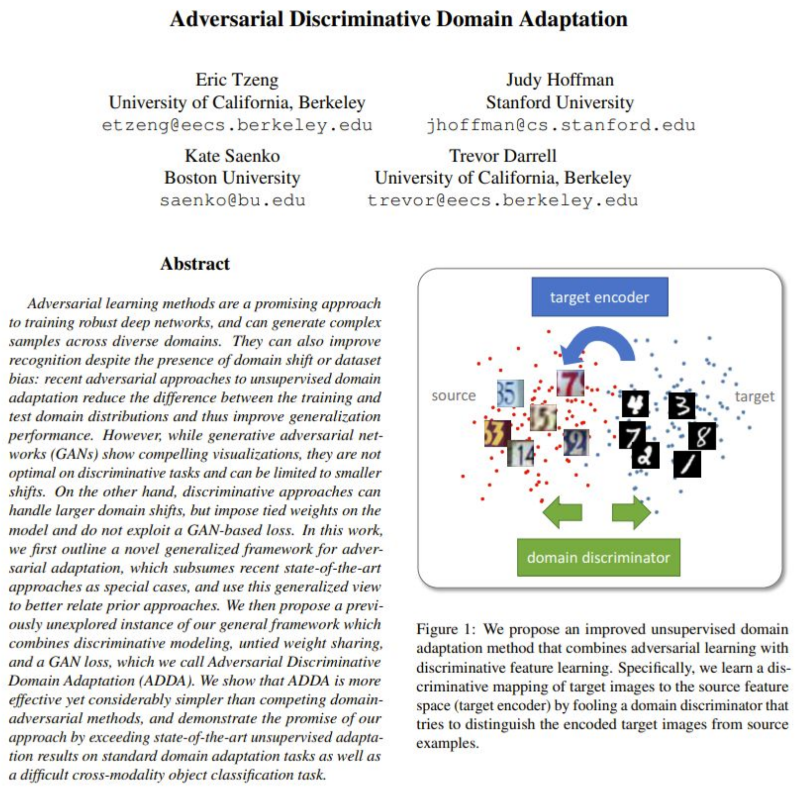

R-CNN: *Regions with CNN features*
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

Paper

- Consider training a binary classifier to detect cars. It’s clear that an image region tightly enclosing a car should be a positive example. Similarly, it’s clear that a background region, which has nothing to do with cars, should be a negative example. Less clear is how to label a region that partially overlaps a car.

- Once features are extracted and training labels are applied, we optimize one linear SVM per class.

Paper



Adversarial Discriminative Domain Adaptation

Eric Tzeng
University of California, Berkeley
etzeng@eecs.berkeley.edu

Judy Hoffman
Stanford University
jhoffman@cs.stanford.edu

Kate Saenko
Boston University
saenko@bu.edu

Trevor Darrell
University of California, Berkeley
trevor@eecs.berkeley.edu

Abstract

Adversarial learning methods are a promising approach to training robust deep networks, and can generate complex samples across diverse domains. They can also improve recognition despite the presence of domain shift or dataset bias: recent adversarial approaches to unsupervised domain adaptation reduce the difference between the training and test domain distributions and thus improve generalization performance. However, while generative adversarial networks (GANs) show compelling visualizations, they are not optimal on discriminative tasks and can be limited to smaller shifts. On the other hand, discriminative approaches can handle larger domain shifts, but impose tied weights on the model and do not exploit a GAN-based loss. In this work, we first outline a novel generalized framework for adversarial adaptation, which subsumes recent state-of-the-art approaches as special cases, and use this generalized view to better relate prior approaches. We then propose a previously unexplored instance of our general framework which combines discriminative modeling, untied weight sharing, and a GAN loss, which we call Adversarial Discriminative Domain Adaptation (ADDA). We show that ADDA is more effective yet considerably simpler than competing domain-adversarial methods, and demonstrate the promise of our approach by exceeding state-of-the-art unsupervised adaptation results on standard domain adaptation tasks as well as a difficult cross-modality object classification task.

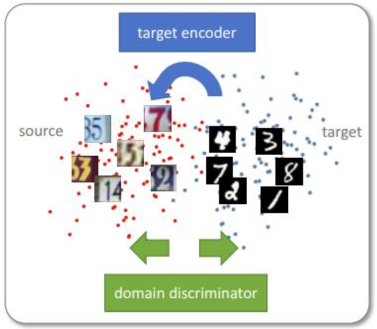
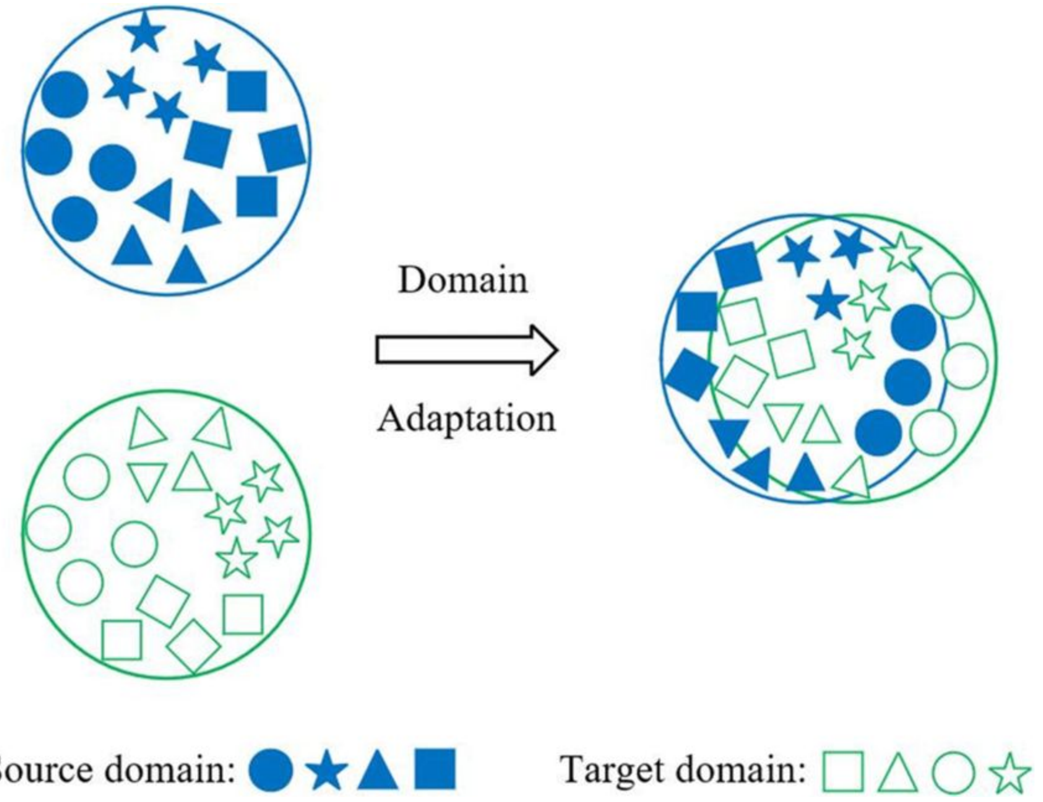
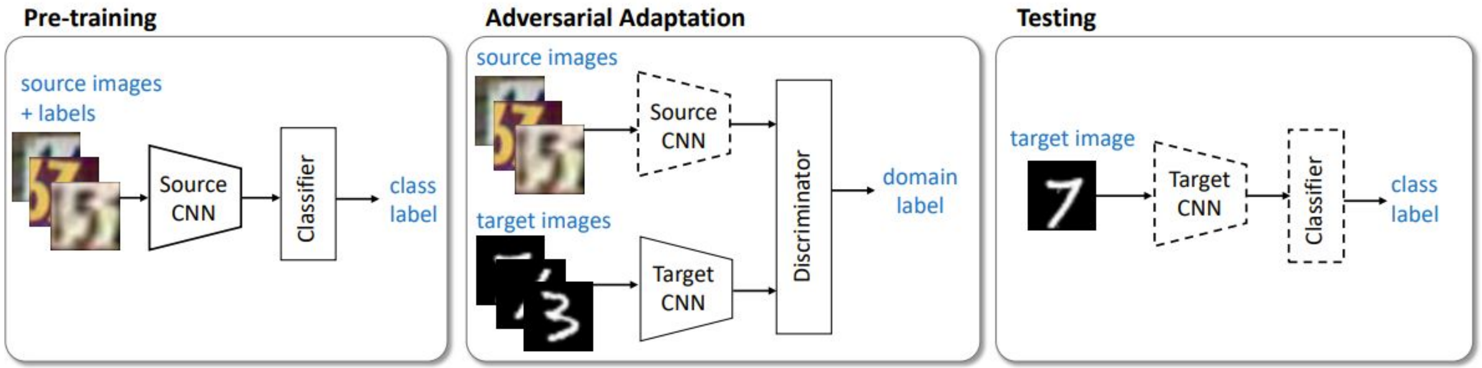


Figure 1: We propose an improved unsupervised domain adaptation method that combines adversarial learning with discriminative feature learning. Specifically, we learn a discriminative mapping of target images to the source feature space (target encoder) by fooling a domain discriminator that tries to distinguish the encoded target images from source examples.

Paper



Paper


Pre-training
source images
+ labels
Source CNN
Classifier
class label
Adversarial Adaptation
source images
Source CNN
target images
Target CNN
Discriminator
domain label
Testing
target image
Target CNN
Classifier
class label