5. Visão geral de Aprendizagem Profunda

5.6 - Types of Neural Networks

Wendley S. Silva

Setembro, 2020

Index

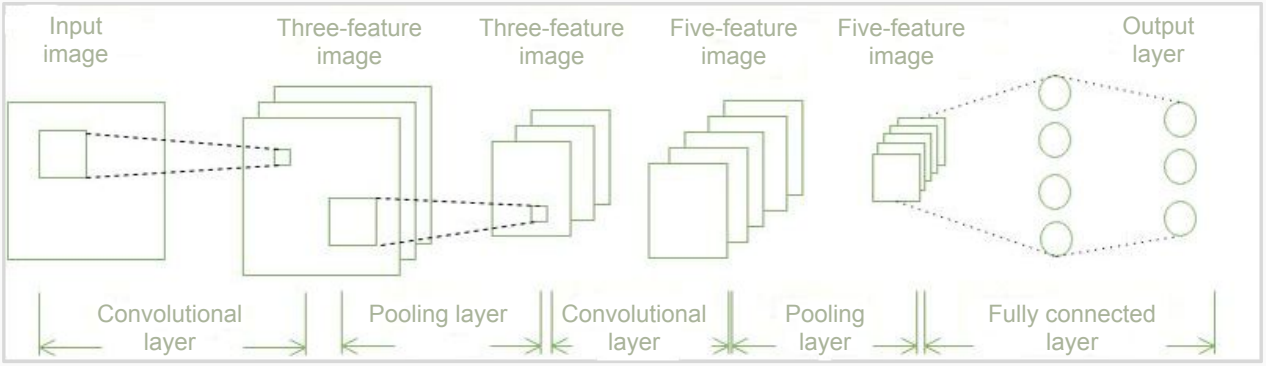
Convolutional Neural Network

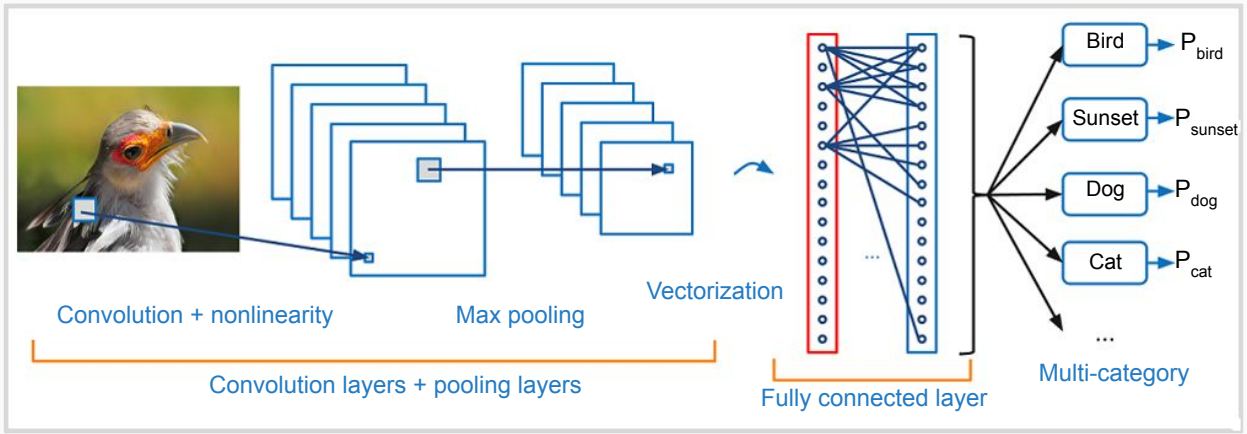
- A convolutional neural network (CNN) is a feedforward neural network. Its artificial neurons may respond to **surrounding units within the coverage range**. CNN excels at image processing. It includes a **convolutional layer**, a **pooling layer**, and a **fully connected layer**.
- In the 1960s, Hubel and Wiesel studied cats' cortex neurons used for local sensitivity and direction selection and found that their unique network structure could simplify feedback neural networks. They then proposed the CNN.
- Now, CNN has become one of the research hotspots in many scientific fields, especially in the pattern classification field. The network is widely used because it can avoid complex pre-processing of images and directly input original images.

Main Concepts of CNN

- **Local receptive field:** It is generally considered that human perception of the outside world is from local to global. **Spatial correlations among local pixels of an image are closer than those among distant pixels**. Therefore, each neuron does not need to know the global image. It only needs to know the local image. The local information is combined at a higher level to generate global information.
- **Parameter sharing:** One or more convolution cores may be used to scan input images. Parameters carried by the convolution cores are weights. In a layer scanned by convolution cores, each core uses the same parameters during weighted computation. Weight sharing means that when each convolution core scans an entire image, parameters of the convolution core are fixed.

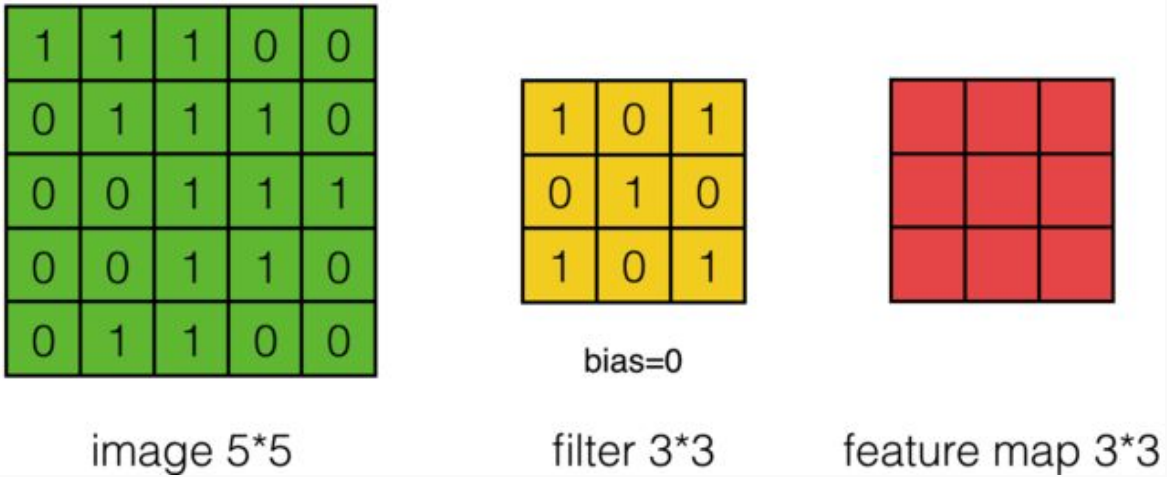
Architecture of Convolutional Neural Network





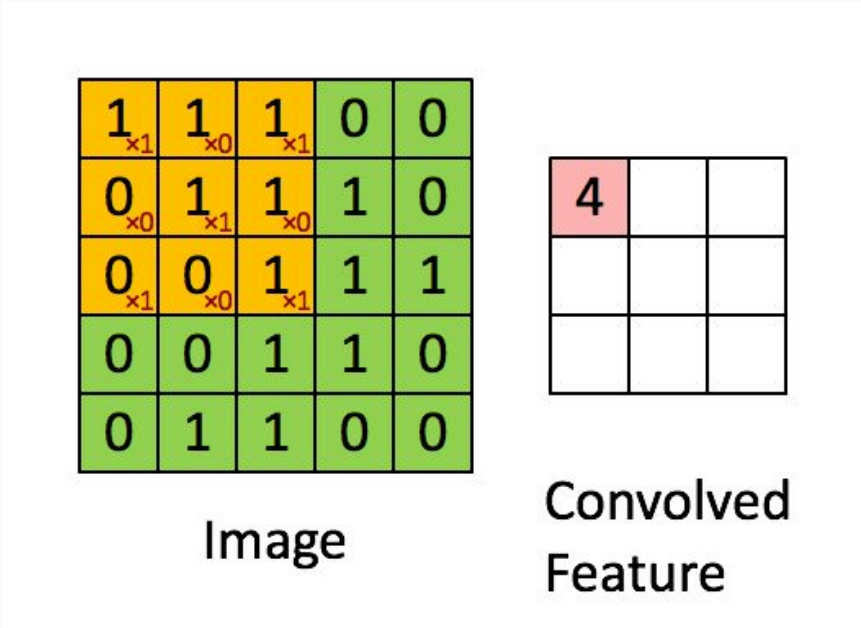
Single-Convolution Core Calculation (1)

- Description of convolution calculation



Single-Convolution Core Calculation (2)

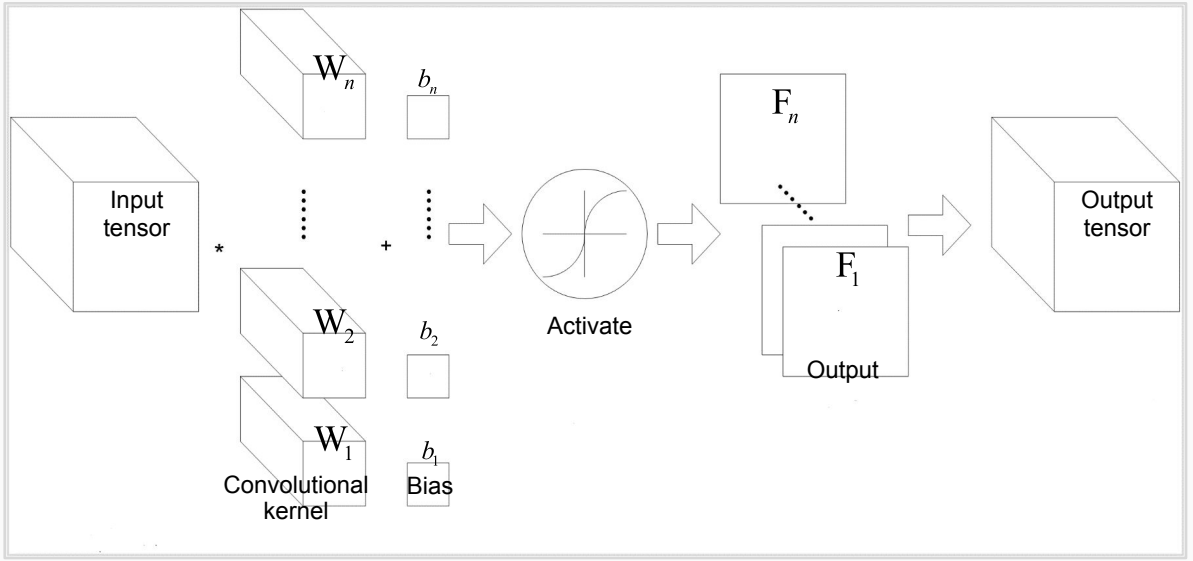
- Demonstration of the convolution calculation



Han Bingtao, 2017, Convolutional Neural Network

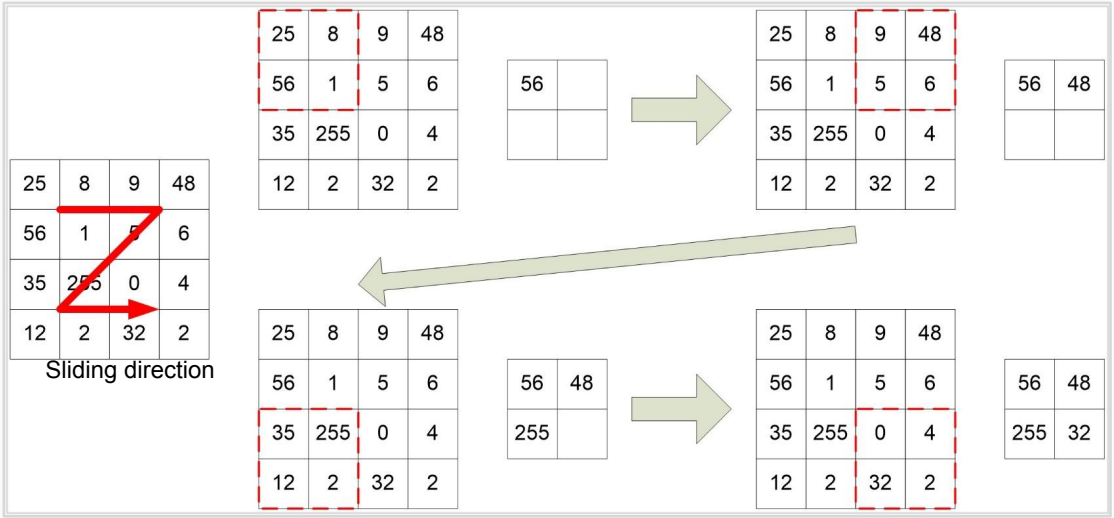
Convolutional Layer

- The basic architecture of a CNN is multi-channel convolution consisting of multiple single convolutions. The output of the previous layer (or the original image of the first layer) is used as the input of the current layer. It is then convolved with the convolution core of the layer and serves as the output of this layer. The convolution kernel of each layer is the weight to be learned. Similar to FCN, after the convolution is complete, the result should be biased and activated through activation functions before being input to the next layer.



Pooling Layer

- Pooling combines nearby units to reduce the size of the input on the next layer, **reducing dimensions**. Common pooling includes max pooling and average pooling. When max pooling is used, the maximum value in a small square area is selected as the representative of this area, while the mean value is selected as the representative when average pooling is used. The side of this small area is the pool window size. The following figure shows the max pooling operation whose pooling window size is 2.



Fully Connected Layer

- The fully connected layer is essentially a classifier. The features extracted on the convolutional layer and pooling layer are straightened and placed at the fully connected layer to output and classify results.
- Generally, the Softmax function is used as the activation function of the final fully connected output layer to combine all local features into global features and calculate the score of each type.

$$\sigma(\mathrm{z})_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Recurrent Neural Network

- The recurrent neural network (RNN) is a neural network that **captures dynamic information in sequential data** through periodical connections of hidden layer nodes. It can classify sequential data.
- Unlike other forward neural networks, the RNN can keep a context state and even store, learn, and express related information in context windows of any length. Different from traditional neural networks, it is not limited to the space boundary, **but also supports time sequences**. In other words, there is a side between the hidden layer of the current moment and the hidden layer of the next moment.
- The RNN is widely used in scenarios related to sequences, such as videos consisting of image frames, audio consisting of clips, and sentences consisting of words.

Recurrent Neural Network Architecture

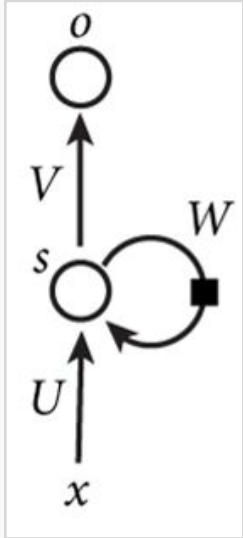
- X_t is the input of the input sequence at time t.
- S_t is the memory unit of the sequence at time t and caches previous information.

$$S_t = tanh(UX_t + WS_{t-1}).$$

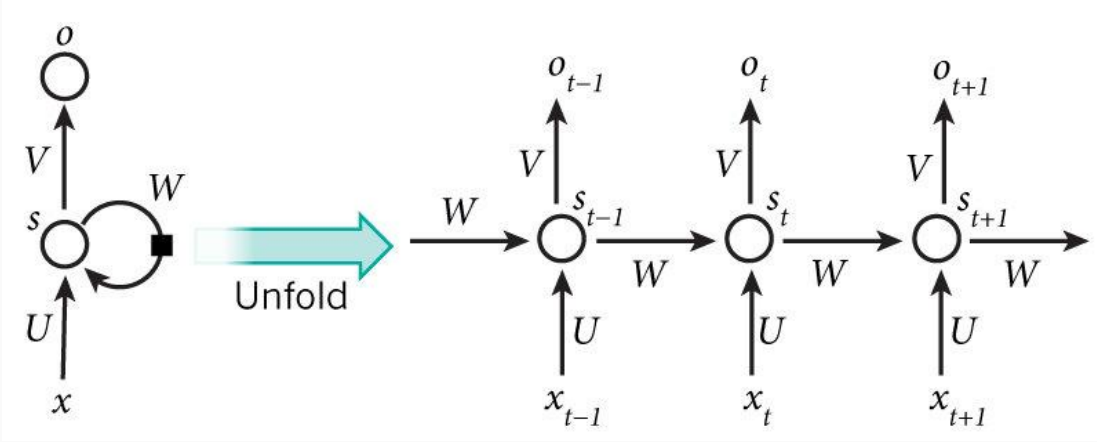
- O_t is the output of the hidden layer of the sequence at time t.

$$O_t = tanh(VS_t)$$

- O_t after through multiple hidden layers, it can get

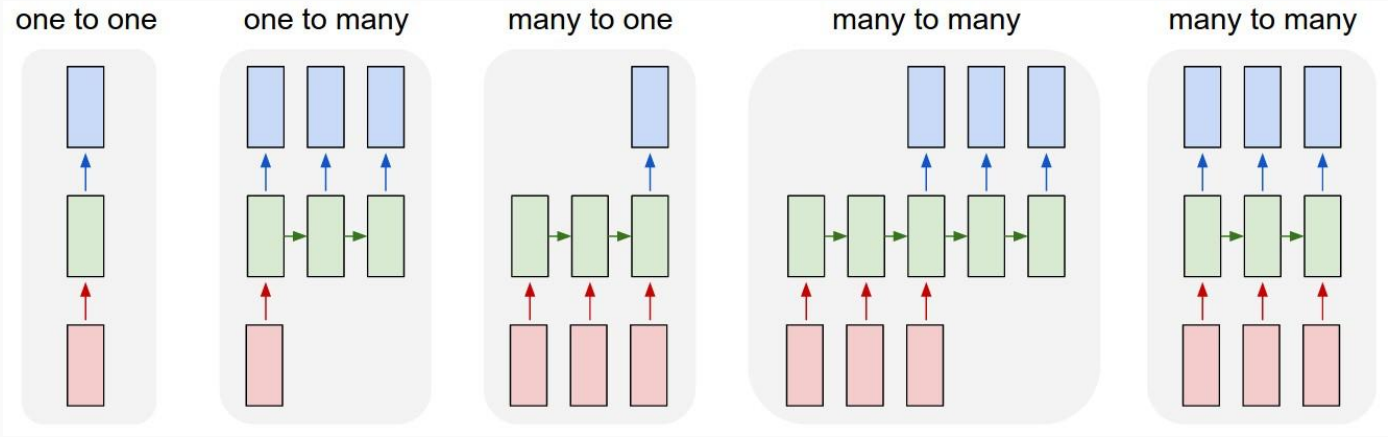


Recurrent Neural Network Architecture



LeCun, Bengio, and G. Hinton, 2015, A Recurrent Neural Network and the Unfolding in Time of the Computation Involved in Its Forward Computation

Types of Recurrent Neural Networks



Andrej Karpathy, 2015, The Unreasonable Effectiveness of Recurrent Neural Networks

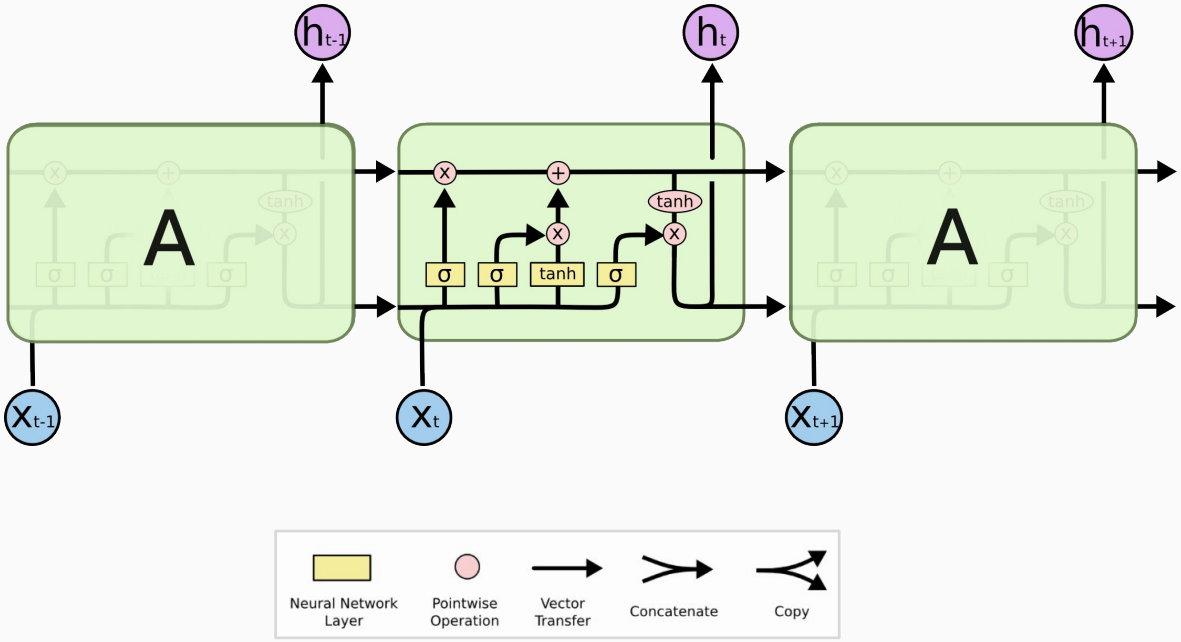
Backpropagation Through Time (BPTT)

- BPTT:
  - Traditional backpropagation is the extension on the time sequence.
  - There are two sources of errors in the sequence at time of memory unit: first is from the hidden layer output error at t time sequence; the second is the error from the memory cell at the next time sequence t + 1.
  - The longer the time sequence, the more likely the loss of the last time sequence to the gradient of w in the first time sequence causes the vanishing gradient or exploding gradient problem.
  - The total gradient of weight w is the accumulation of the gradient of the weight at all time sequence.
- Three steps of BPTT:
  - Computing the output value of each neuron through forward propagation.
  - Computing the error value of each neuron through backpropagation δ_j.
  - Computing the gradient of each weight.

Recurrent Neural Network Problem

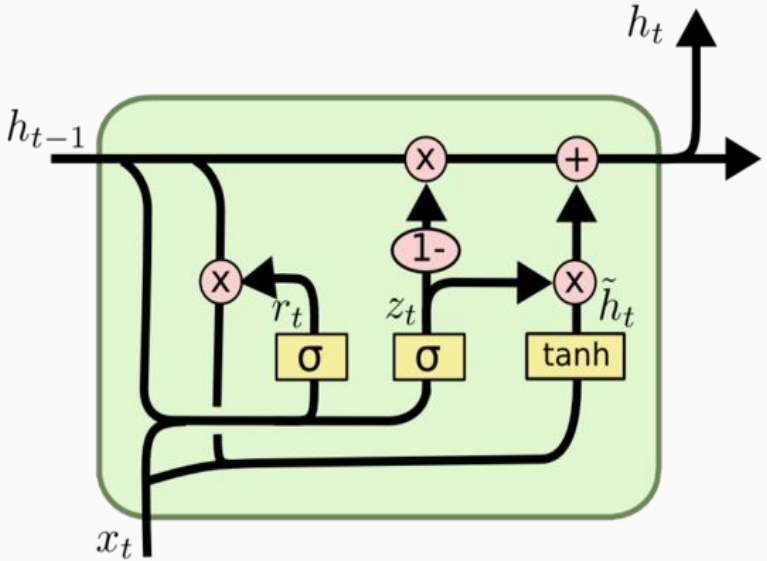
- $S_t = \sigma(UX_t + WS_{t-1})$ is extended on the time sequence.
- $S_t = \sigma\left(UX_t + W\left(\sigma\left(UX_{t-1} + W\left(\sigma\left(UX_{t-2} + W(\dots)\right)\right)\right)\right)\right)$
- Despite that the standard RNN structure solves the problem of information memory, **the information attenuates during long-term memory**.
- Information needs to be saved long time in many tasks. For example, a hint at the beginning of a speculative fiction may not be answered until the end.
- The RNN may not be able to save information for long due to the limited memory unit capacity.
- We expect that memory units can remember key information.

Long Short-term Memory Network



Colah, 2015, Understanding LSTMs Networks

Gated Recurrent Unit (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

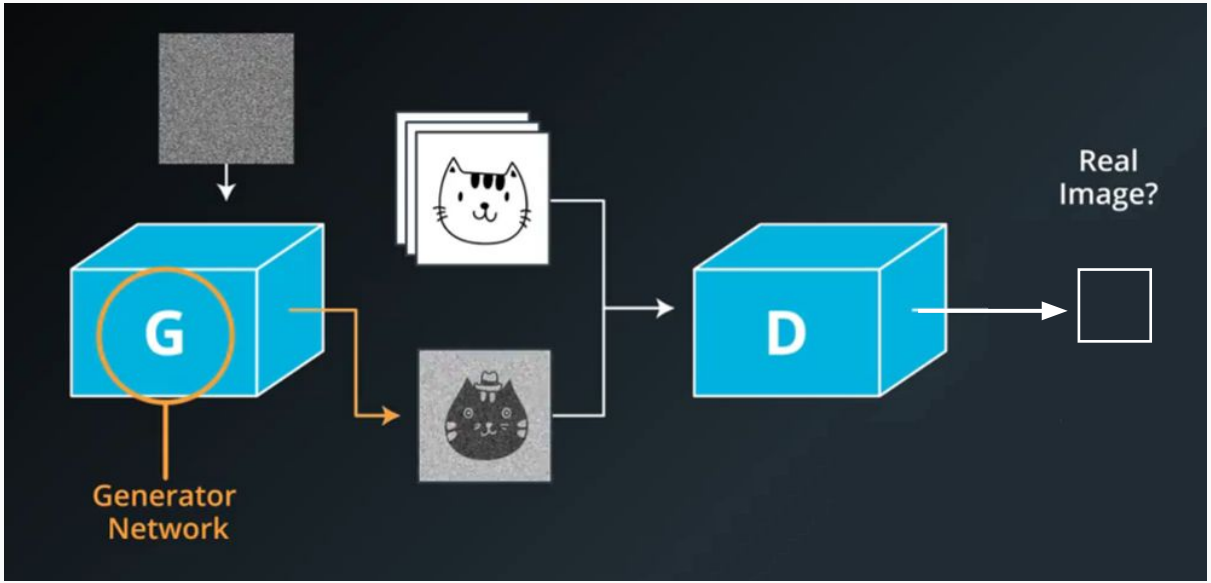
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Generative Adversarial Network (GAN)

- Generative Adversarial Network is a framework that trains generator G and discriminator D through the adversarial process. Through the adversarial process, the discriminator can tell whether the sample from the generator is fake or real. GAN adopts a mature BP algorithm.
- (1) Generator G: The input is noise z, which complies with manually selected prior probability distribution, such as even distribution and Gaussian distribution. The generator adopts the network structure of the multilayer perceptron (MLP), uses maximum likelihood estimation (MLE) parameters to represent the derivable mapping G(z), and maps the input space to the sample space.
- (2) Discriminator D: The input is the real sample x and the fake sample G(z), which are tagged as real and fake respectively. The network of the discriminator can use the MLP carrying parameters. The output is the probability D(G(z)) that determines whether the sample is a real or fake sample.
- GAN can be applied to scenarios such as image generation, text generation, speech enhancement, image super-resolution.

GAN Architecture

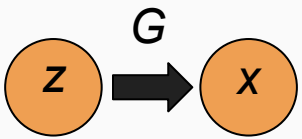
- Generator/Discriminator



Generative Model and Discriminative Model

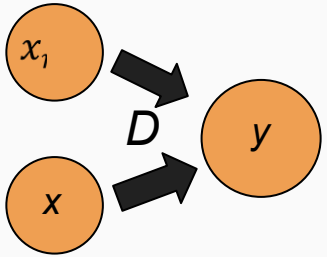
- Generative network
  - Generates sample data
    - Input: Gaussian white noise vector z
    - Output: sample data vector x

$$x = G(z;\theta^G)$$



- Discriminator network
  - Determines whether sample data is real
    - Input: real sample data x_{real} or generated sample data $x = G(z)$
    - Output: probability that determines whether the sample is real

$$y = D(x;\theta^D)$$



Training Rules of GAN

- Optimization objective:
  - Value function

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}} [logD(x)] + E_{z\sim p_{z(z)}} [log(1-D(G(z)))]$$

  - In the early training stage, when the outcome of G is very poor, D determines that the generated sample is fake with high confidence, because the sample is obviously different from training data. In this case, log(1-D(G(z))) is saturated (where the gradient is 0, and iteration cannot be performed). Therefore, we choose to train G only by **minimizing [-log(D(G(z)))]**.

5 - Deep Learning Overview

Next: 5.7 - Common Problems in Deep Learning

Wendley Silva

Setembro, 2020