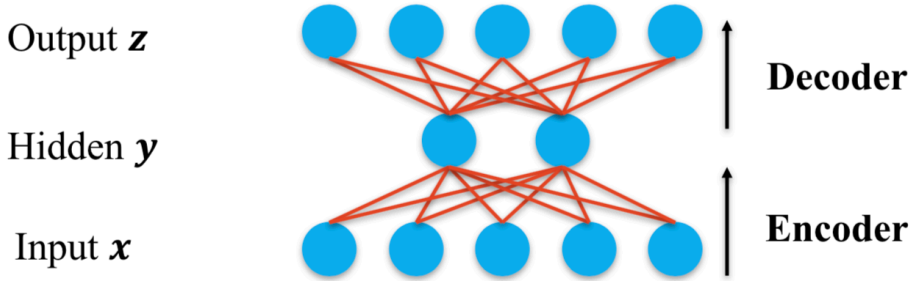
Week 12
Deep Learning

Auto-encoders



Encoder: $y = f(w_1 x + b_1)$ mapping x => y

Decoder: $z = f(w_2 y + b_2)$ mapping hidden representation => reconstruction z of the same shape as x

Deconstruction error:

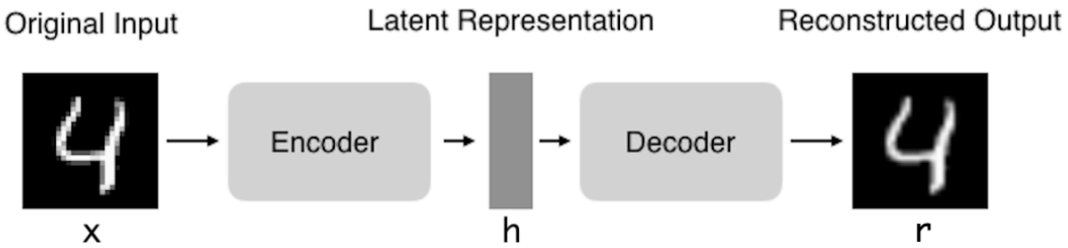
1. square loss: $J(\theta) = ||x - z||^2$
2. cross-entropy: $J(\theta) = -\sum_{i=1}^{d} x_i log(z_i) + (1 - x_i) log(1 - z_i)$

We can use back propagation to optimise AE.

Applications:

1. Text retrieval
2. Similar image search
3. Pre-training DNN
4. De-noising auto-encoder (adds noises to input data, uses the corrupted data as input of AE, then reconstructs the original input x)

Auto-encoders



AE is aim to obtain main and useful features.

Difference between AE and PCA?

AE: nonlinearity
PCA: orthogonality

AE with Regularisation

Without regularisation, AE tends to learn identity function

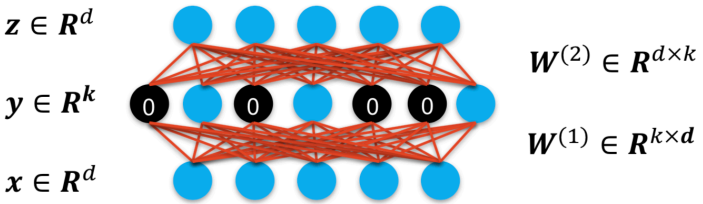
Two main types regularised AE:

1. Sparse AE
2. Denoising AE

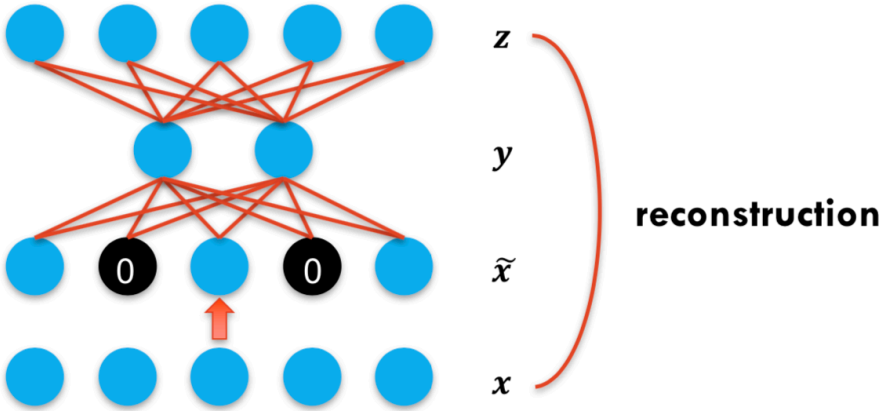
Over-complete, we want to learn sparse representation.

Why sparse representation?

More numerous the sparse representation will need to be to achieve the same reconstruction error rate on a training data set.

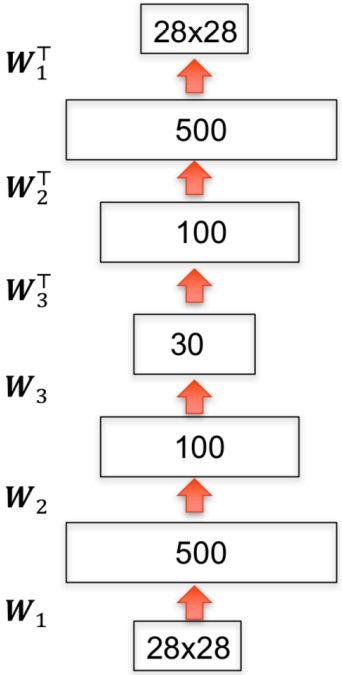


AE with Regularisation

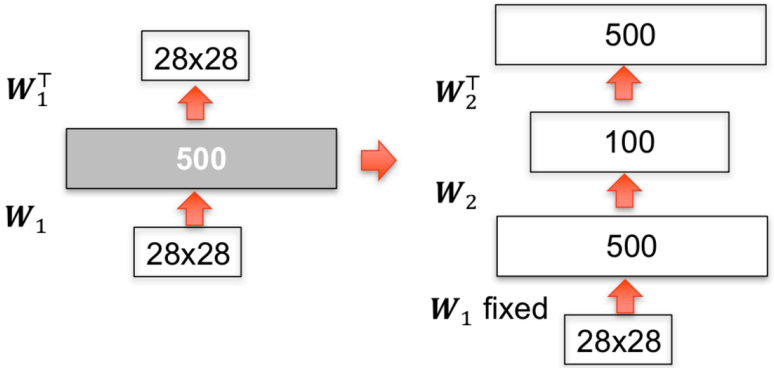


Denoising AE : Adding some noise of the input image and make the AE learn to remove it. By this means, the encoder will extract the most important features and learn a robust representation of the data.

Stacked AE

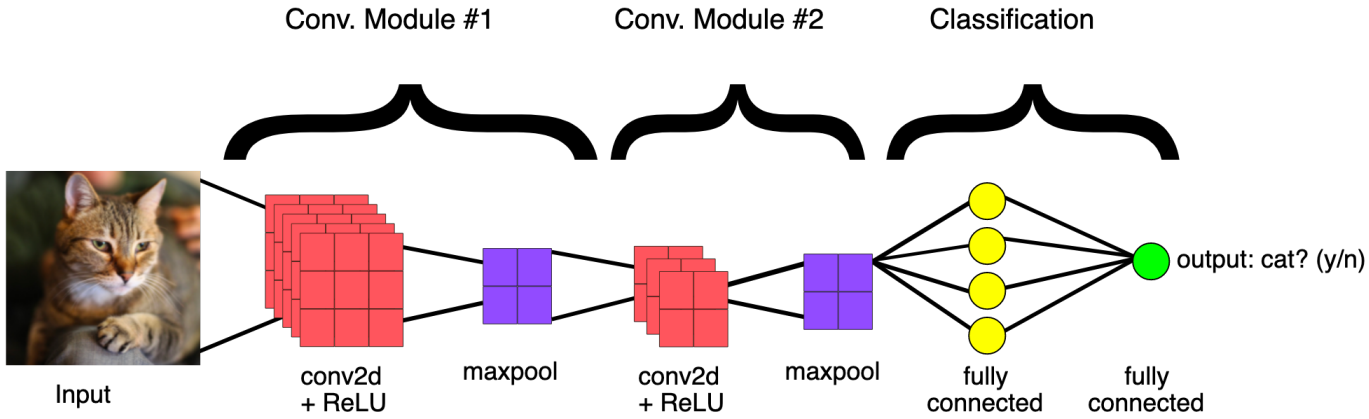


Method: Greedy layer-wise training + fine-tuning using BP.



Stacked AE tends to learn higher-order features. When add more hidden layers, it helps to reduce a high dimensional data to a smaller code representing important features. Each hidden layer is a more compact representation than the last hidden layer.

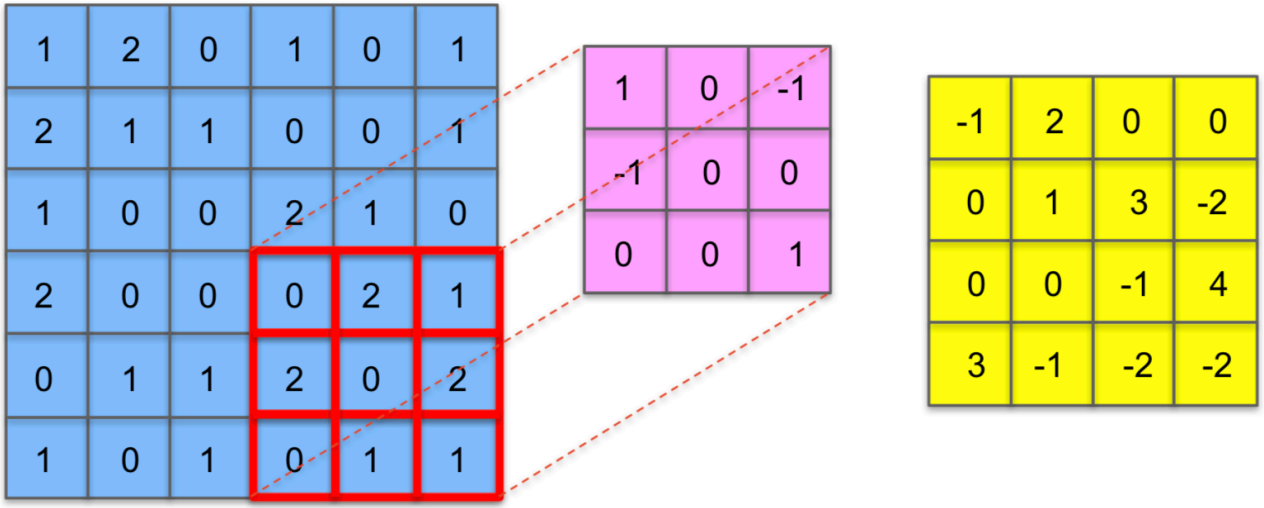
CNN



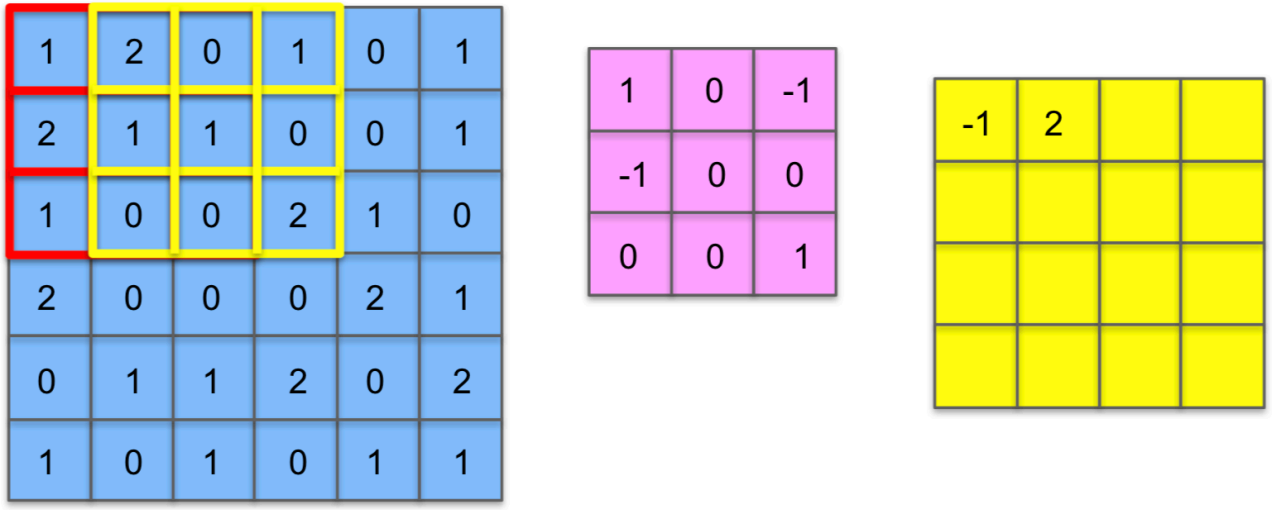
Basic CNN structure:

- Convolutional Layer
- Pooling
- Fully-connected Layer

CNN – Convolutional Layer



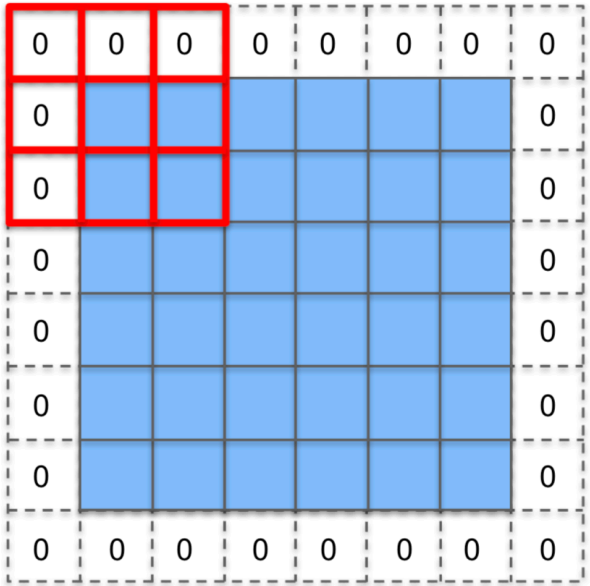
CNN – Convolutional Layer



Stride = 1

CNN – Convolutional Layer

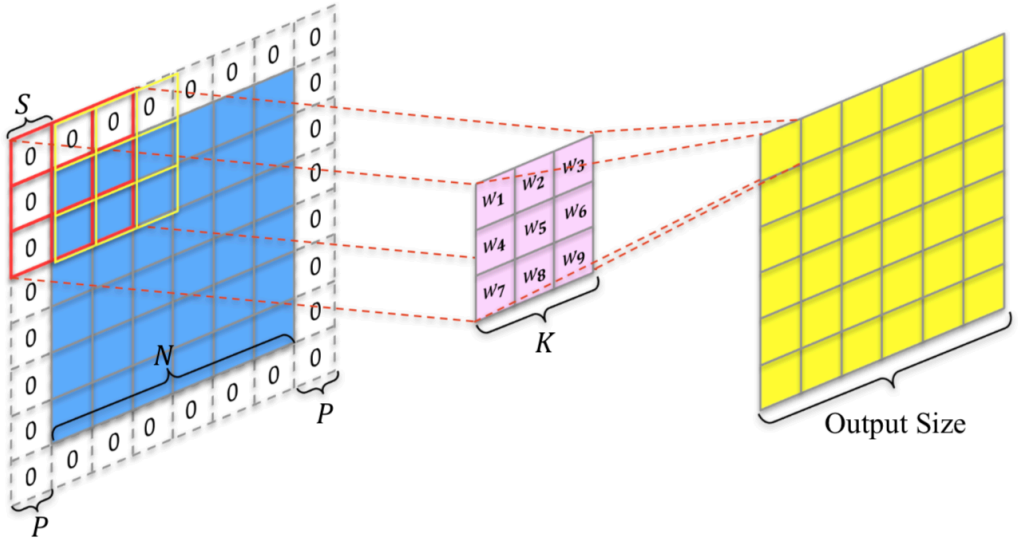
Zero padding (pad = 1)



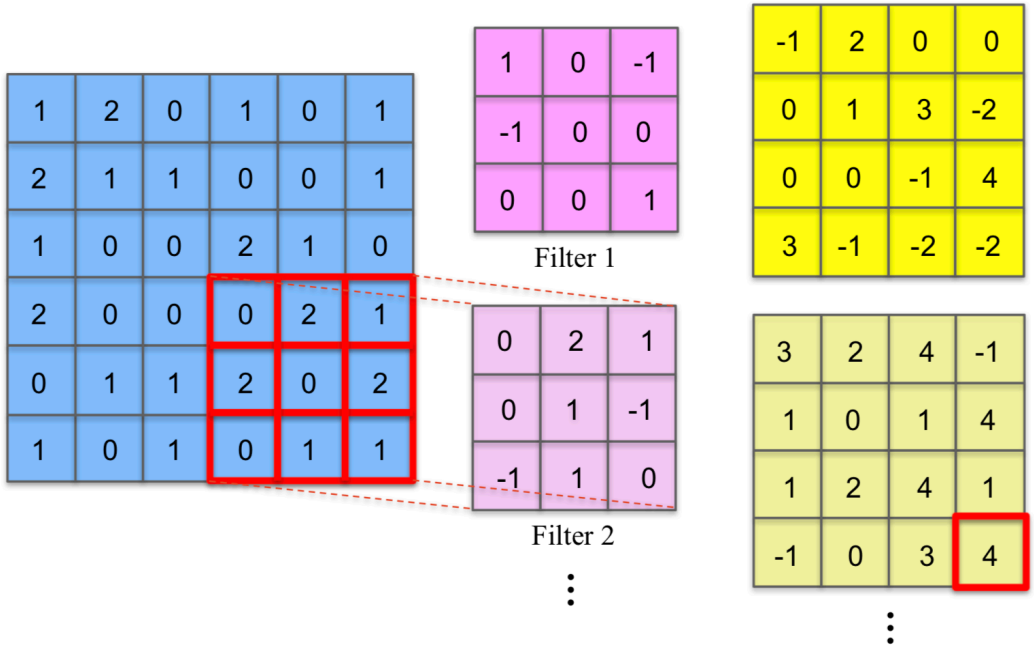
improves performance by keeping information at the borders

CNN – Convolutional Layer

$$\text{Output Size} = \frac{N+2P-K}{S} + 1$$



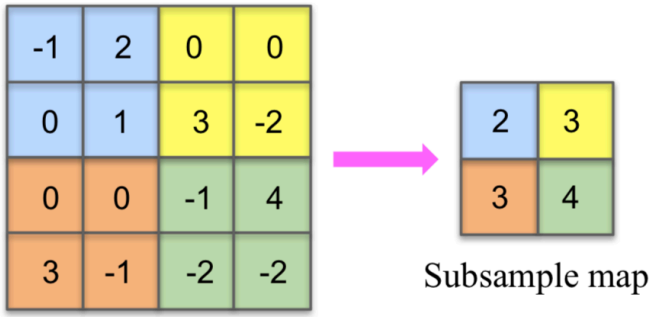
CNN – Convolutional Layer



CNN – Pooling

Max pooling

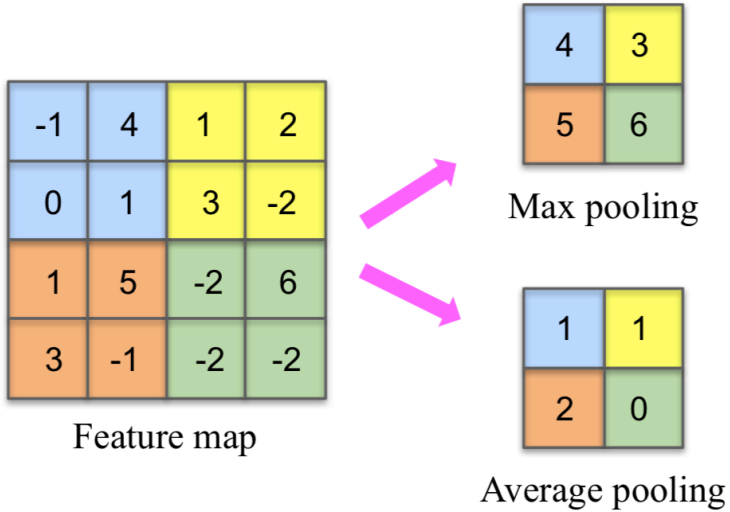
- Filter size: (2,2)
- Stride: (2,2)
- Pooling ops: max(·)



CNN – Pooling

Average pooling

- Filter size: (2,2)
- Stride: (2,2)
- Pooling ops: mean(·)



CNN – Pooling

L_2 norm pooling

- Filter size (Gaussian kernel size): (2,2)
- Stride: (2,2)
- Pooling ops: $y_i = \sqrt{\sum_j w_j x_{i,j}{}^2}$

Feature map

Gaussian window

Output

CNN – Pooling

L_2 norm pooling

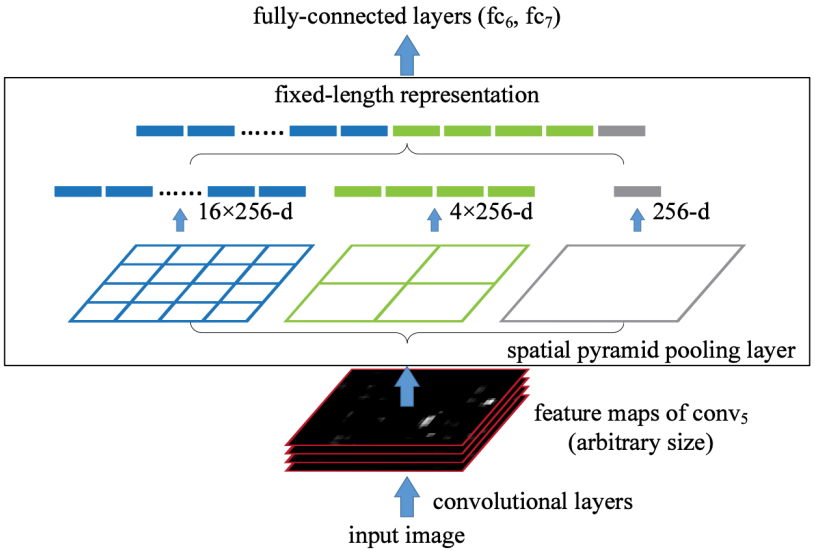
- Filter size (Gaussian kernel size): (2,2)
- Stride: (2,2)
- Pooling ops: $y_i = \sqrt{\sum_j w_j x_{i,j}{}^2}$

Feature map

Gaussian window

Output

CNN – Spatial Pyramid Pooling (SPP)



1. SPP allows CNN to use input images of any size (because FC layer need fixed-size input)
2. Add SPP on the last convolutional layer, before FC layer.
3. [EX] Use 4*4, 2*2 and 1*1 filters to get 21 bins.

Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition
Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun

https://arxiv.org/pdf/1406.4729.pdf

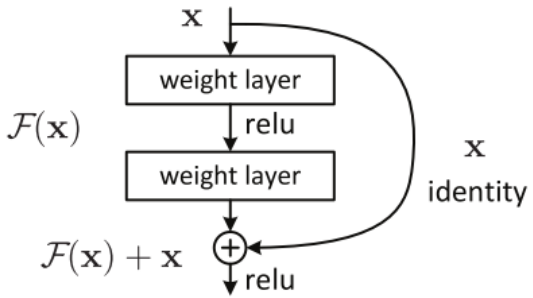
CNN Architecture

LeNet => AlexNet => ZFNet => VGGNet => GoogLeNet => ResNet

Since AlexNet, the state-of-the-art CNN architecture is going deeper and deeper

Problem with very deep CNN architecture:

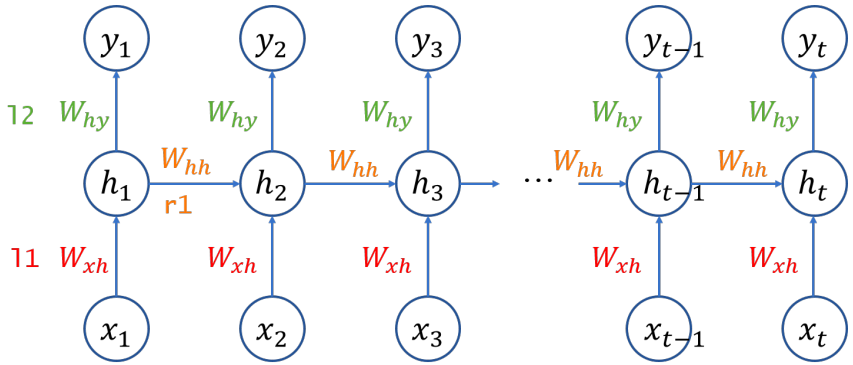
1. vanishing/exploding gradient problem.
2. degradation problem: as the network goes deeper, its performance gets saturated or even starts degrading rapidly.



Deep Residual Learning for Image Recognition
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

https://arxiv.org/abs/1512.03385

RNN and LSTM



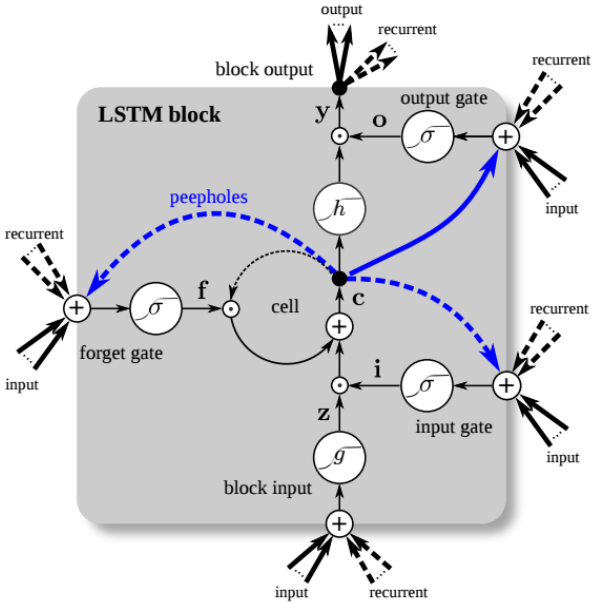
$$h_t = \tanh(\, l1(x_t) + r1(h_{t-1})\,)$$
$$y_t = l2(h_t)$$

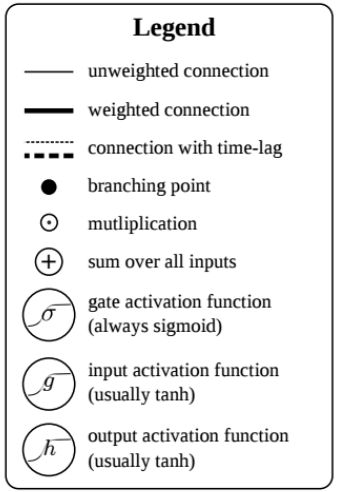
The output of hidden layer are stored in memory. Memory can be considered as another input.

So information cycles through a loop.

1. current input
2. what it has leaned from the inputs it received previously.

RNN and LSTM





LSTM is an extension for RNN.

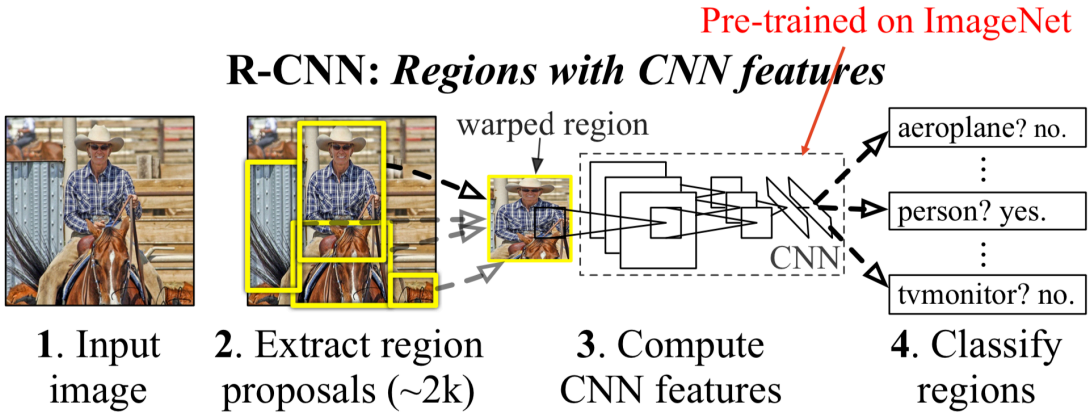
The memory in LSTM can be seen as a gated cell (store or delete based on the importance it assigns to the information).

Three gates:

1. input gate: whether or not let new input in
2. forget gate: delete information
3. output gate: output at the current time step

R-CNN

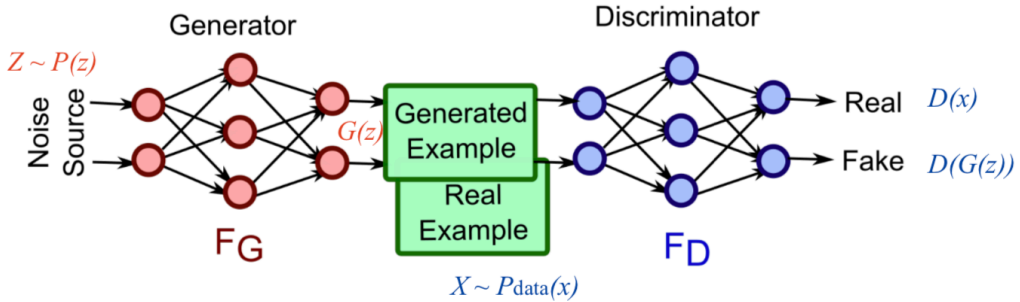
Object detection: detect semantic objects of certain classes in images.



Region proposal:
Input image => find all possible places where objects can be located. The output of this stage should be a list of bounding boxes of likely positions of objects. These are often called region proposals or regions of interest (ROI).

Final classification:
for every region proposal from the previous stage, decide whether it belongs to one of the target classes or to the background. Here we could use a deep convolutional network.

GAN



A two-player game between a generator G and a discriminator D

G: try to fool the discriminator by generating real-looking images
D: try to distinguish between real and fake images

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

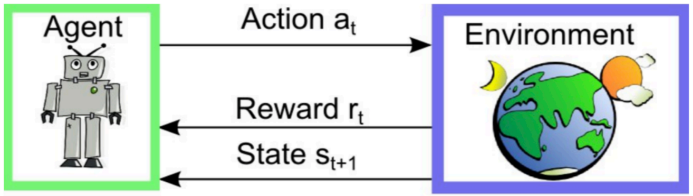
Objective: max the score of real image, minimise the score of fake image

z is noise, G(z) is also a image.
Lower score for fake image, so use 1 – D(G(z))

We want to max D, if D is maximised, we will get higher score for real image and lower score for fake image
But for G, we want to minimise it so D(G(z)) will get higher score (since G want to fool D)

D's goal: maximise objective such that D(x) is close to 1 (real) and D(G)) is close to 0 (fake)
G's goal: minimise objective such that D(G(z)) is close to 1 (discriminator) is fooled into thinking generated G(z) is real)

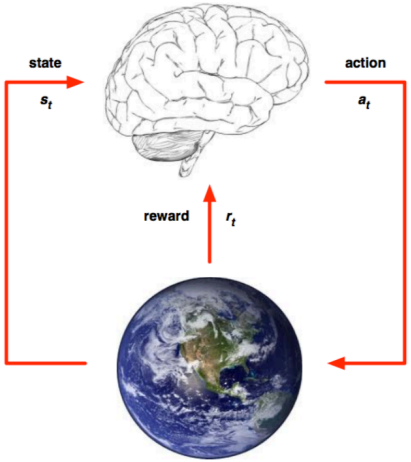
Reinforcement Learning



RL is a general-purpose framework for decision making:

1. Agent to act with an environment
2. Each action influences the agent's future state
3. Feedback is given by reward signal
4. Goal: select actions to maximise future reward

Markov Decision Processes



- At each step t the agent:
  - Receives state s_t
  - Receives scalar reward r_t
  - Executes action a_t
- The environment:
  - Receives action a_t
  - Emits state s_t
  - Emits scalar reward r_t

Some theory

Gradient Descent Finds Global Minima of Deep Neural Networks

https://arxiv.org/pdf/1811.03804.pdf