

# From Neural Networks To ...

(STATE-OF-ART NEURAL NETWORKS MODEL AND ALGORITHMS2019)



Shen Hengheng

April 20, 2019

University of Chinese Academy of Sciences

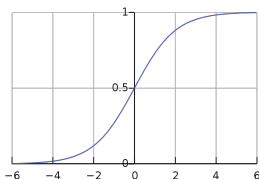
shenhengheng17g@ict.ac.cn



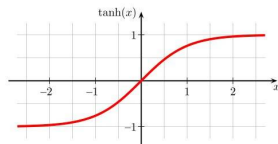
# 在我们讨论深度学习之前...

Check the **regression** notes on:

- revise on sigmoid and tanh function
- revise on logistic and softmax regression
- Then we talk about neural networks and multilayer perceptron



$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



$$\tanh u = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$

# Feedforward Neural Network in a nutshell

We begin Feedforward Neural networks, in a nutshell, comprised of the following steps:

- Feedforward

$$f(\mathbf{x}) = f_L(\dots f_2(f_1(\mathbf{x}; \theta_1); \theta_2) \dots), \theta_L)$$

- The objective is to minimize the overall cost:

$$L_\theta = \frac{1}{n} \sum_{i=1}^n l(\mathbf{y}_i, f(\mathbf{x}_i; \theta))$$

- To put it in a gradient descent framework, i.e.,

$$\theta_{t+1} = \theta_t - \eta \frac{\partial f}{\partial \theta}(\theta_t)$$

# Something about $\frac{\partial f}{\partial \theta_t}$

We know all about it already from high school mathematics:

- Product rule:

$$\frac{\partial}{\partial \theta} (f(\theta)g(\theta)) = f(\theta) \frac{\partial}{\partial \theta} g(\theta) + \frac{\partial}{\partial \theta} g(\theta) f(\theta)$$

- Derivative of sums:

$$\frac{\partial}{\partial \theta} (f(\theta) + g(\theta)) = \frac{\partial f(\theta)}{\partial \theta} + \frac{\partial g(\theta)}{\partial \theta}$$

- Chain rule

$$\begin{aligned} \frac{\partial}{\partial \theta_l} &= \frac{\partial}{\partial \theta_l} f_L(\dots f_2(f_1(\mathbf{x}; \theta_1); \theta_2) \dots), \theta_L) \\ &= \frac{\partial f_L}{\partial f_{L-1}^T} \frac{\partial f_{L-1}}{\partial f_{L-2}^T} \dots \frac{\partial f_{l+1}}{\partial f_l^T} \frac{\partial f_l}{\partial \theta_l} \end{aligned}$$

# Multivariable Chain Rule

We know all about it already from high school mathematics:

- Product rule:

$$\frac{\partial}{\partial \theta} (f(\theta)g(\theta)) = f(\theta) \frac{\partial}{\partial \theta} g(\theta) + \frac{\partial}{\partial \theta} g(\theta) f(\theta)$$

- Derivative of sums:

$$\frac{\partial}{\partial \theta} (f(\theta) + g(\theta)) = \frac{\partial f(\theta)}{\partial \theta} + \frac{\partial g(\theta)}{\partial \theta}$$

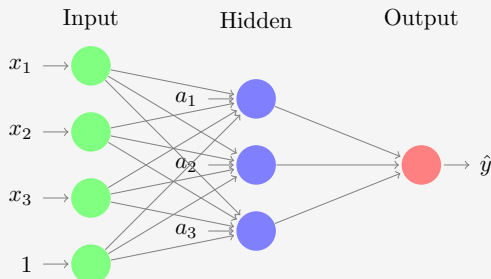
- Chain rule

$$\begin{aligned} \frac{\partial}{\partial \theta_l} &= \frac{\partial}{\partial \theta_l} f_L(\dots f_2(f_1(\mathbf{x}; \theta_1); \theta_2) \dots), \theta_L) \\ &= \frac{\partial f_L}{\partial f_{L-1}^T} \frac{\partial f_{L-1}}{\partial f_{L-2}^T} \dots \frac{\partial f_{l+1}}{\partial f_l^T} \frac{\partial f_l}{\partial \theta_l} \end{aligned}$$

# Look at Neural network systematically: Feed Forward (1)

$$\begin{aligned}\hat{y} &= U^T f(Wx + b) \\ &= U^T \underbrace{f(Wx + b)}_z = U^T \underbrace{a}_a\end{aligned}$$

let  $\hat{y}$  be a **scalar** score instead of a **softmax** this time



$$\begin{aligned}\hat{y} &= U^T f(Wx + b) \\ &= U^T f\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}\right)\end{aligned}$$

# Look at Neural network systematically: Feed Forward (2)

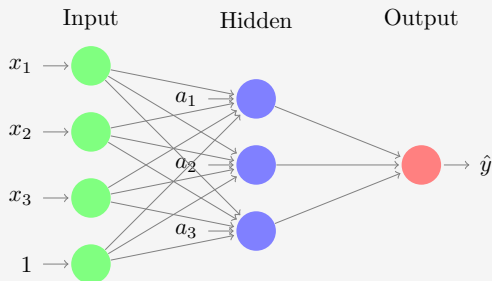
$$z_1 = w_{1,:}^T X + b_1 = \sum_i w_{1,i}^T x_i + b_1$$

$$z_2 = w_{2,:}^T X + b_2 = \sum_i w_{2,i}^T x_i + b_2$$

$$z_3 = w_{3,:}^T X + b_3 = \sum_i w_{3,i}^T x_i + b_3$$

Therefore

$$W_{(\text{index of } a, \text{index of } X)}$$



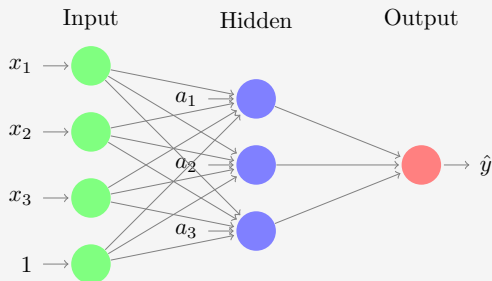
$$\begin{aligned} \hat{y} &= U^T f(Wx + b) \\ &= U^T f \left( \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right) \end{aligned}$$

# Neural Networks: Backpropagation

$$\begin{aligned}\hat{y} &= U^T f(Wx + b) \\ &= U^T \underbrace{f(Wx + b)}_z = U^T a\end{aligned}$$

Careful of their dimensions:

$$\begin{aligned}\frac{\partial \hat{y}}{\partial W} &= \underbrace{\frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial z}}_{\text{columnvector}} \times \underbrace{\frac{\partial z}{\partial W}}_{\text{rowvector}} \\ &= \underbrace{(U \odot f'(z))}_{\text{columnvector}} \times \underbrace{x}_{\text{rowvector}}\end{aligned}$$



$$\begin{aligned}\hat{y} &= U^T f(Wx + b) \\ &= U^T f\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}\right)\end{aligned}$$



# Backpropagation for $W_{i,j}$

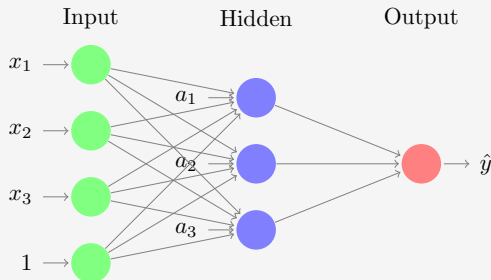
$$\begin{aligned}\hat{y} &= U^T f(Wx + b) \\ &= U^T \underbrace{f(Wx + b)}_a = U^T a\end{aligned}$$

If dimensionality of derivative of  $W$  is too hard to see, then we perform derivative one element  $W_{i,j}$  at the time:

$$W_{(\text{index of } a, \text{index of } X)}$$

If we were to compute  $\frac{\partial S}{\partial W_{i,j}}$ :

$$\begin{aligned}\frac{\partial \hat{y}}{\partial W} &= \frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial W} \\ \Rightarrow \frac{\partial \hat{y}}{\partial W_{i,j}} &= \frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial W_{i,j}} \\ &= \underbrace{U_j^T(Z_i)}_{\delta_i} X_j \\ &= \underbrace{U_j^T(W_{i,:}^T X + b_i)}_{\delta_i} X_j\end{aligned}$$



$$\begin{aligned}\hat{y} &= U^T f(Wx + b) \\ &= U^T f \left( \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)\end{aligned}$$

important to remember:

$$Z_i = W_{i,:}^T X + b_i = (WX + b)_i$$

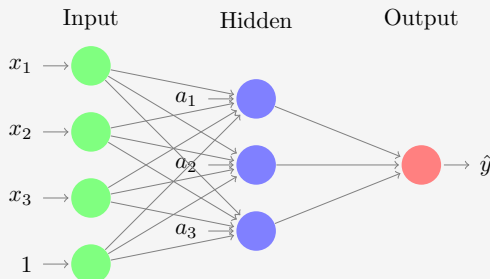
# Backpropagation for $W$

If we were to compute  $\frac{\partial S}{\partial W_{i,j}}$ :

$$\frac{\partial \hat{y}}{\partial W} = \underbrace{u_j f'(w_{i,:}^T X + b_i)}_{\delta_i} x_j$$

$$\begin{aligned} \delta &= \begin{bmatrix} u_1 f'(w_{1,:}^T X + b_1) \\ u_2 f'(w_{2,:}^T X + b_2) \\ u_3 f'(w_{3,:}^T X + b_3) \end{bmatrix} \\ &= U \odot f'(WX + B) \end{aligned}$$

$$\begin{aligned} \frac{\partial \hat{y}}{\partial W} &= \begin{bmatrix} u_1 f'(w_{1,:}^T X + b_1) \\ u_2 f'(w_{2,:}^T X + b_2) \\ u_3 f'(w_{3,:}^T X + b_3) \end{bmatrix} \\ &= \delta X^T \end{aligned}$$



$$\begin{aligned} \hat{y} &= U^T f(Wx + b) \\ &= U^T f \left( \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right) \end{aligned}$$

# Something about $\delta$

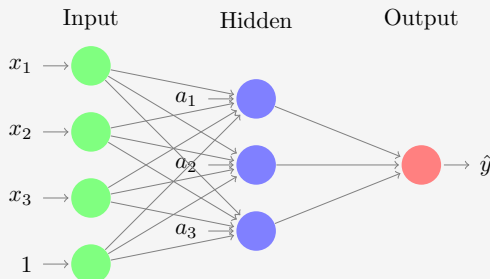
$$\delta = \begin{bmatrix} u_1 f'(w_{1,:}^T x + b_1) \\ u_2 f'(w_{2,:}^T x + b_2) \\ u_3 f'(w_{3,:}^T x + b_3) \end{bmatrix}$$

$$= U \odot f'(WX + B)$$

$$\frac{\partial \hat{y}}{\partial W} = \begin{bmatrix} u_1 f'(w_{1,:}^T x + b_1) \\ u_2 f'(w_{2,:}^T x + b_2) \\ u_3 f'(w_{3,:}^T x + b_3) \end{bmatrix}$$

$$= \delta X^T$$

- $\delta$  is the error signal, i.e.,  $\frac{\partial \hat{y}}{\partial z}$
- $\delta$  involves the derivatives of all the activationfunction  $\{a_i = f(z_i)\}$

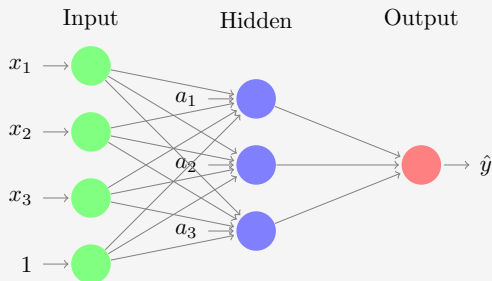


$$\hat{y} = U^T f(Wx + b)$$

$$= U^T f \left( \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

# Backpropagation for $b$

$$\begin{aligned}\frac{\partial \hat{y}}{\partial b_i} &= \underbrace{U_i f'(W_{i,:}^T X + b_i)}_{\delta_i} 1 \\ &= \delta_i\end{aligned}$$

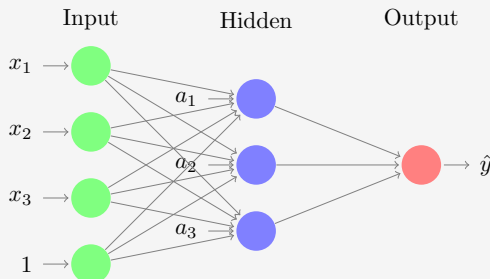


$$\begin{aligned}\hat{y} &= U^T f(Wx + b) \\ &= U^T f\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}\right)\end{aligned}$$

# Backpropagation for $x_j$

Note each  $x_j$  is contributed by all  $\{a_i\}$

$$\begin{aligned}
 \frac{\partial \hat{y}}{\partial x_j} &= \sum_{i=1}^3 \frac{\partial \hat{y}}{\partial a_i} \frac{\partial a_i}{\partial x_j} \\
 &= \sum_{i=1}^3 \frac{\partial \hat{U}^T a}{\partial a_i} \frac{\partial a_i}{\partial x_j} \\
 &= \sum_{i=1}^3 U_i \frac{\partial f(W_{i,:}x + b)}{\partial x_j} \\
 &= \sum_{i=1}^3 U_i \underbrace{f'(W_{i,:}x + b)}_{\delta_i} \frac{\partial W_{i,:}x}{\partial x_j} \\
 &= \sum_{i=1}^3 \delta_i W_{i,j} \\
 &= \delta^T W_{:,j}
 \end{aligned}$$



$$\begin{aligned}
 \hat{y} &= U^T f(Wx + b) \\
 &= U^T f \left( \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)
 \end{aligned}$$

# Backpropagation for two layers with respect to $W^{(2)}$

$$a^{(0)} = x \quad (\text{input})$$

$$z^{(1)} = W^{(1)} a^{(0)} + b^{(1)} \quad (\text{linear})$$

$$a^{(1)} = f(z^{(1)}) \quad (\text{non-linear})$$

$$z^{(2)} = W^{(2)} a^{(1)} + b^{(2)} \quad (\text{linear})$$

$$a^{(2)} = f(z^{(2)}) \quad (\text{non-linear})$$

$$\hat{y} = U^T a^{(2)} \quad (\text{output})$$

$$\hat{y} = U^T f(W^{(2)} f(W^{(1)} x + b^{(1)}) + b^{(2)})$$

$$\frac{\partial \hat{y}}{\partial W_{i,j}} = \underbrace{U_j f'(z_i) x_j}_{\delta_i} \quad (\text{one layer case})$$

$$\Rightarrow \frac{\partial \hat{y}}{\partial W_{i,j}^{(2)}} = \underbrace{U_j f'(z_i^{(2)}) a_j^{(2)}}_{\delta_i^{(2)}}$$

$$\Rightarrow \frac{\partial \hat{y}}{\partial W^{(2)}} = \delta_i^{(2)} a^{(2)T} \quad (\text{where } \delta^{(2)} = U \odot f'(z^{(2)}))$$

# Backpropagation for two layers with respect to $W^{(1)}$

$$a^{(0)} = x \quad (\text{input})$$

$$z^{(1)} = W^{(1)}a^{(0)} + b^{(1)} \quad (\text{linear})$$

$$a^{(1)} = f(z^{(1)}) \quad (\text{non-linear})$$

$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)} \quad (\text{linear})$$

$$a^{(2)} = f(z^{(2)}) \quad (\text{non-linear})$$

$$\hat{y} = U^T a^{(2)} \quad (\text{output})$$

$$\begin{aligned} \hat{y} &= U^T f(W^{(2)} f(W^{(1)} x + b^{(1)}) + b^{(2)}) \\ &= U^T f(W^{(2)} \underbrace{f(W^{(1)} x + b^{(1)})}_{z^{(1)}} + b^{(2)}) \\ &\quad \underbrace{\hspace{10em}}_{z^{(2)}} \end{aligned}$$

$$\begin{aligned} \frac{\partial \hat{y}}{\partial W^{(1)}} &= \underbrace{U^T f'(W^{(2)} f(W^{(1)} x + b^{(1)}))}_{\frac{\partial \hat{y}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}}} \underbrace{W^{(2)}}_{\frac{\partial z^{(2)}}{\partial a^{(1)}}} \underbrace{f'(W^{(1)} x + b^{(1)})}_{\frac{\partial a^{(1)}}{\partial z^{(1)}}} \underbrace{x}_{\frac{\partial z^{(1)}}{\partial W^{(1)}}} \\ &= \underbrace{\frac{\partial \hat{y}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial W^{(1)}}}_{\delta^{(2)}} \\ &\quad \underbrace{\hspace{10em}}_{\delta^{(1)}} \\ &= \underbrace{\delta^{(2)} W^{(2)} f'(z^{(2)})}_{\delta^{(1)}} x \end{aligned}$$

# Generalisation

Putting them in the "correct" form of the matrix operations and generalise:

$$\begin{aligned}
 &= \underbrace{(W^{(2)T} \delta^{(2)}) \odot f'(z^{(2)})}_{\delta^{(1)}} X^T \\
 \Rightarrow \delta^{(2)} &= W^{(2)T} \delta^{(2)} \odot f'(z^{(2)}) \\
 \Rightarrow \delta^{(l)} &= W^{(l)T} \delta^{(l)} \odot f'(z^{(l)})
 \end{aligned}$$



# Backpropagation in action!

$$a^{(0)} = x \quad (\text{input})$$

$$z^{(1)} = w^{(1)} a^{(0)} + b^{(1)} \quad (\text{linear})$$

$$a^{(1)} = f(z^{(1)}) \quad (\text{non-linear})$$

$$z^{(2)} = w^{(2)} a^{(1)} + b^{(2)} \quad (\text{linear})$$

$$a^{(2)} = f(z^{(2)}) \quad (\text{non-linear})$$

$$\hat{y} = U^T a^{(2)} \quad (\text{output})$$

To generalise this:

$$\delta^{(l)} = w^{(l)T} \delta^{(l+1)} \odot f'(z^{(l)})$$

- step 1: compute  $\delta$  of the last layer  $L$ :

$$\delta^{(L)} = U \odot \underbrace{f'(z^{(L)})}_{\text{from feed-forward}}$$

- step 2: Generate the whole sequence of  $\{\delta^{(l)}\}_1^L$

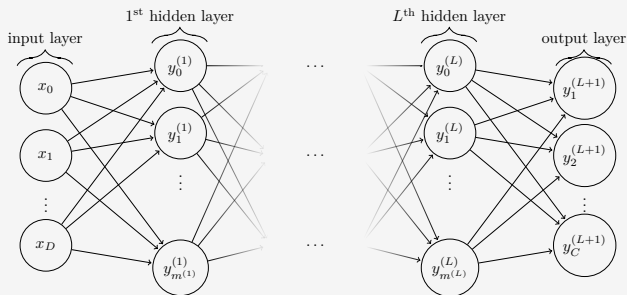
$$\delta^{(l)} = w^{(l)T} \delta^{(l+1)} \odot f'(z^{(l)})$$

- step 3: compute gradients at each  $\frac{\partial \hat{y}}{\partial w^{(l)}}_1^{(L-1)}$ :

$$\frac{\partial \hat{y}}{\partial w^{(l)}} = \delta^{(l+1)} a^{(l)T}$$

Note that  $\frac{\partial \hat{y}}{\partial w^{(l)}}$  can be obtained as soon as  $\delta^{(l+1)}$  becomes available.

# Good old Feedforward Neural Network with Many layers

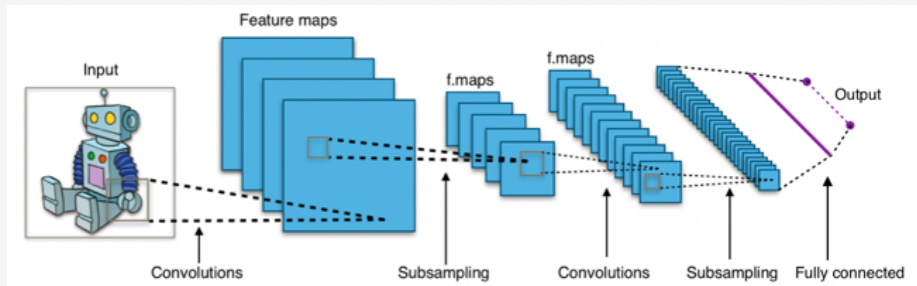


**Figure:** Network graph of a  $(L+1)$ -layer perceptron with  $D$  input units and  $C$  output units. The  $l^{\text{th}}$  hidden layer contains  $m^{(l)}$  hidden units.

# Feedforward Neural Network with Many layers

- For complicated problem, one hidden layer may NOT be enough.
- $\sigma$  Layers can be flexibly designed: linear, Softmax, ReLU or any other
- Each layers may have their own parameters  $\theta(l)$
- There are many such layers, hence many parameters
- Think about the case of **one meg pixel image**.
- How may we reduce the number of parameters from the fully connected network?
- keyword of the day: **parameter sharing**

# A high level view of Convolution Neural Network (CNN)



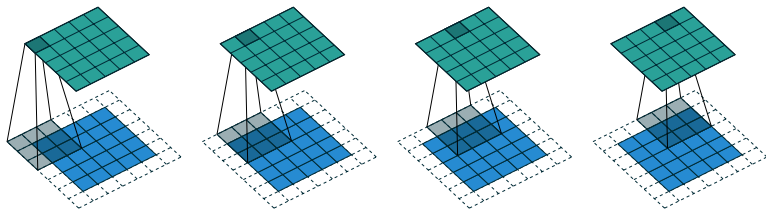
# Convolution Neural Networks: What is a convolution?

- from a signal processing perspective:

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau = \int_{-\infty}^{\infty} f(t - \tau)g(\tau)d\tau$$

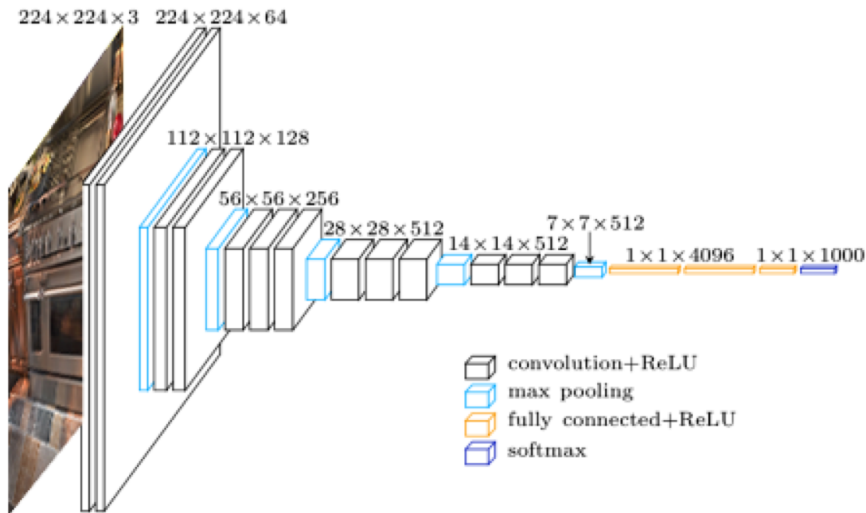
# What is a 2D Convolution in Discrete Imaging?

An example of Convoluting a  $3 \times 3$  kernel over a  $5 \times 5$  input with padding1.

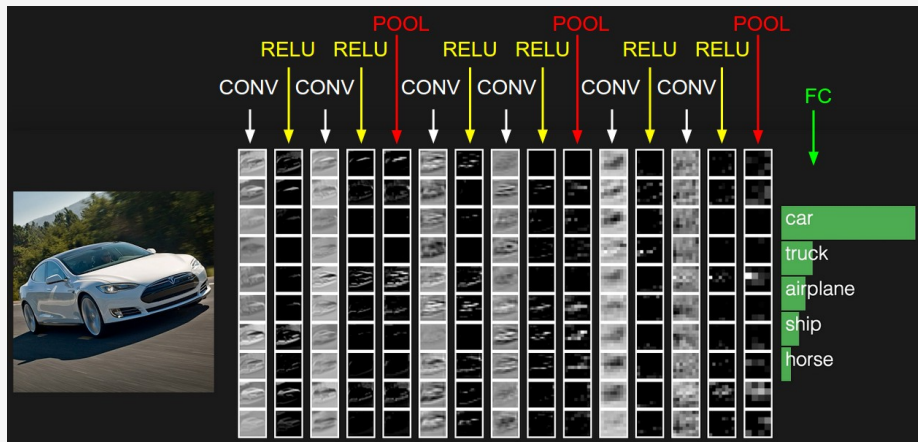


- ▶ Exploiting the strong spatially local correlation present in natural images.
- ▶ In term of number of parameters, assume in a layer, you have 128 convolution Kernels (the grey square), you create 64 “copies” of the image.
- ▶ Assume in a layer, if each kernel is  $5 \times 5$  in size, we have something like  $128 \times 5 \times 5 = 3200$  parameters.
- ▶ A lot less than fully connected neural networks

# CNN for image classification: look at it again



# What does the "learnt" image feature represent?





# CNN Feed-forward: convolution as a linear operator

- Let's forget about only considers:

$$x_{i,j}^{(l)} = W * (y^{(l-1)}) \quad y_{ij}^{(l)} = \sigma(x_{ij}^{(l)})$$

- as matter of fact, in 1 - D convolution, one may replace convolution by matrix multiplication:

$$x^{(l)} = W * (y^{(l-1)}) = \begin{bmatrix} w_1 & 0 & \cdots & 0 & 0 \\ w_2 & w_1 & \cdots & \vdots & \vdots \\ w_3 & w_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \vdots & w_3 & \cdots & w_1 & 0 \\ w_{m-1} & \vdots & \cdots & w_2 & w_1 \\ w_m & w_{m-1} & \vdots & \vdots & w_2 \\ 0 & w_m & \cdots & w_{m-2} & \vdots \\ 0 & 0 & \cdots & w_{m-1} & w_{m-2} \\ \vdots & \vdots & \vdots & w_m & w_{m-1} \\ 0 & 0 & 0 & \cdots & w_m \end{bmatrix} \begin{bmatrix} y_1^{(l-1)} \\ y_2^{(l-1)} \\ y_3^{(l-1)} \\ \vdots \\ y_n^{(l-1)} \end{bmatrix}$$

# CNN Feed-forward: looking at one layer only

- say at layer  $(l - 1)$ :  $y^{(l-1)}$  has  $N \times N$  square neurons
- use  $m \times m$  filter  $\omega$
- when stride 0 used,  $x^{(l)}$  will be of size  $(N - m + 1) \times (N - m + 1)$
- for simplicity - shift the centre by  $(\frac{m}{2}, \frac{m}{2})$

$$x_{ij}^{(l)} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab} y_{(i+a)(j+b)}^{(l-1)}$$

$$y_{ij}^{(l)} = \delta(x_{ij}^{(l)})$$

# CNN back-propagation

- we need four derivative quantities:

- $\frac{\partial y_{ij}^{(l)}}{\partial x_{ij}^{(l)}} = \sigma'(x_{ij}^{(l)})$

- $\frac{\partial x_{ij}^{(l)}}{\partial w_{a,b}^{(l)}} = y_{i+a,j+b}^{(l-1)}$

- $\frac{\partial x_{(i-a)(j-b)}^{(l)}}{\partial y_{ij}^{(l-1)}} = \omega_{a,b}$

- say we already knew  $\frac{\partial E}{\partial y_{ij}^{(l)}}$

find  $\frac{\partial E}{\partial y_{ij}^{(l-1)}}$ : it involves a "filter-window" of  $x^{(l)}$

$$\begin{aligned} \frac{\partial E}{\partial y_{ij}^{(l-1)}} &= \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{(l)}} \frac{\partial x_{(i-a)(j-b)}^{(l)}}{\partial y_{ij}^{(l-1)}} \\ &= \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial E}{\partial y_{ij}^{(l)}} \sigma'(x_{(i-a)(j-b)}^{(l)}) \omega_{a,b} \end{aligned}$$

sum over a filter window

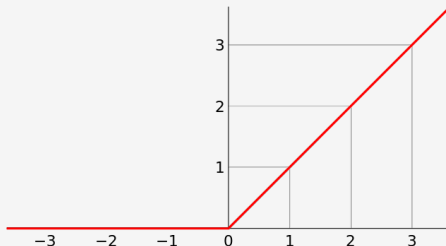
find  $\frac{\partial E}{\partial w_{a,b}^{(l)}}$ : it involves entire  $x^{(l)}$

$$\begin{aligned} \frac{\partial E}{\partial y_{ij}^{(l-1)}} &= \sum_{i=0}^{N-m+1} \sum_{j=0}^{N-m+1} \frac{\partial E}{\partial x_{ij}^{(l)}} \frac{\partial x_{ij}^{(l)}}{\partial w_{a,b}^{(l)}} \\ &= \sum_{i=0}^{N-m+1} \sum_{j=0}^{N-m+1} \frac{\partial E}{\partial y_{ij}^{(l)}} \sigma'(x_{i,j}^{(l)}) y_{i+a,j+b}^{(l-1)} \end{aligned}$$

sum over an entire image

# The winning ingredient: Rectified Linear Unit

- How to combine the different "convolved" images together?
- A Linear + **Activation function**  $f(x) = \max(0, x)$



- High school mathematics is useful
- its derivative can only be  $\{0, 1\}$ . This is useful to prevent gradient vanishing and exploding
- They do not require any exponential computation (such as those required in sigmoid)
- reported around 6X speed over existing activation functions