

Introduction to Machine Learning Artificial Neural Networks

Prof. Chang-Chieh Cheng

Information Technology Service Center

National Chiao Tung University

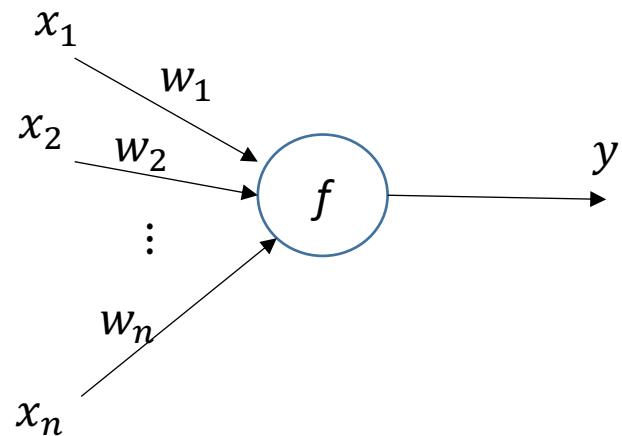
Artificial Neural Network

- Artificial neuron

- Also called "Perceptron"
- A set of inputs $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_n]$ and weights $\mathbf{w} = [w_1 \ w_2 \ \cdots \ w_n]$
- output $y = f(\mathbf{x}, \mathbf{w})$
- For example:

-

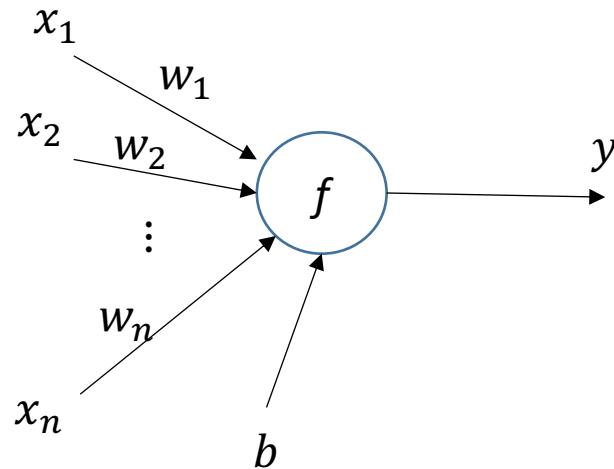
$$f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^n x_i w_i$$



Artificial Neural Network

- Artificial neuron
 - Biased perceptron

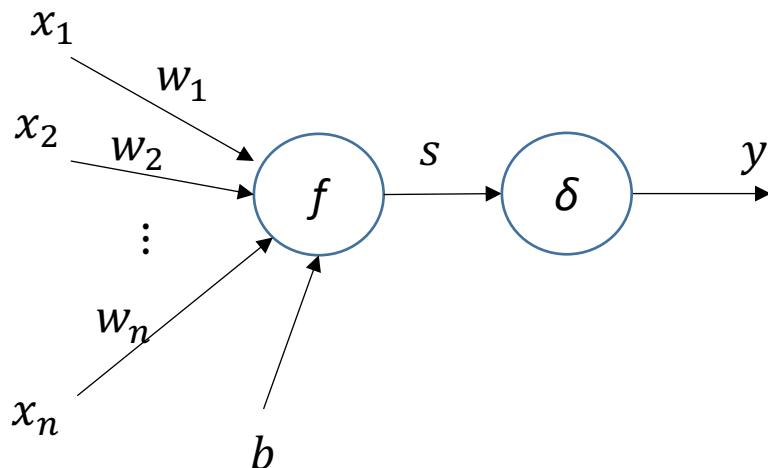
$$f(\mathbf{x}, \mathbf{w}, b) = \sum_{i=1}^n x_i w_i + b$$



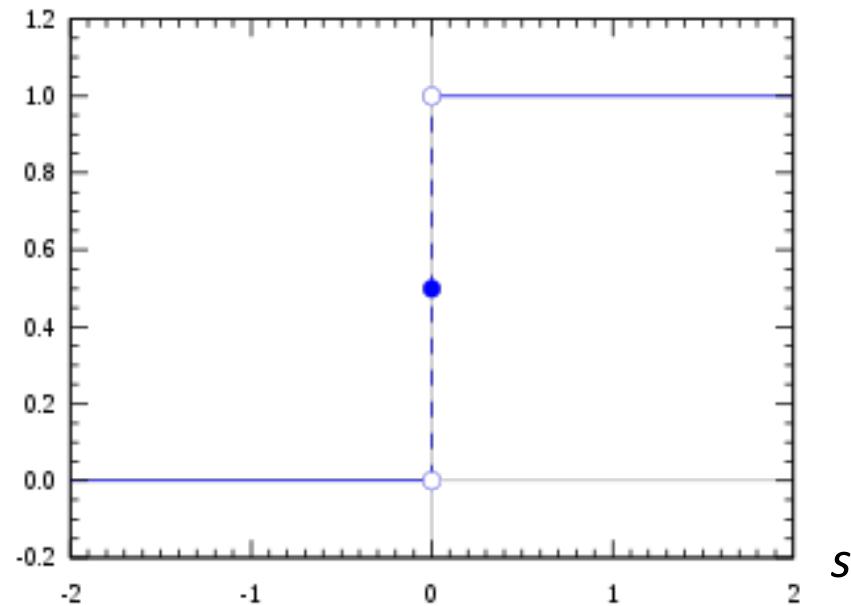
Artificial Neural Network

- Activation functions

- Step activation



$$y = \delta(s) = \begin{cases} 0 & \text{if } s \leq 0 \\ 1 & \text{if } s > 0 \end{cases}$$

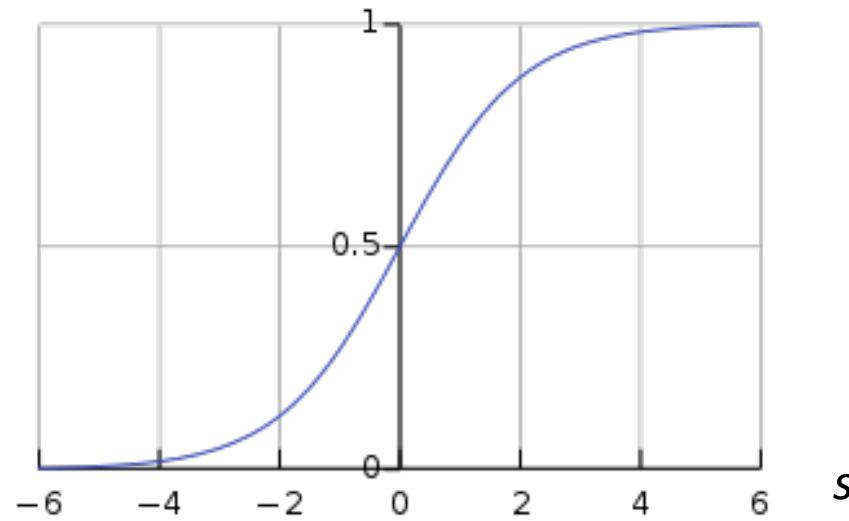
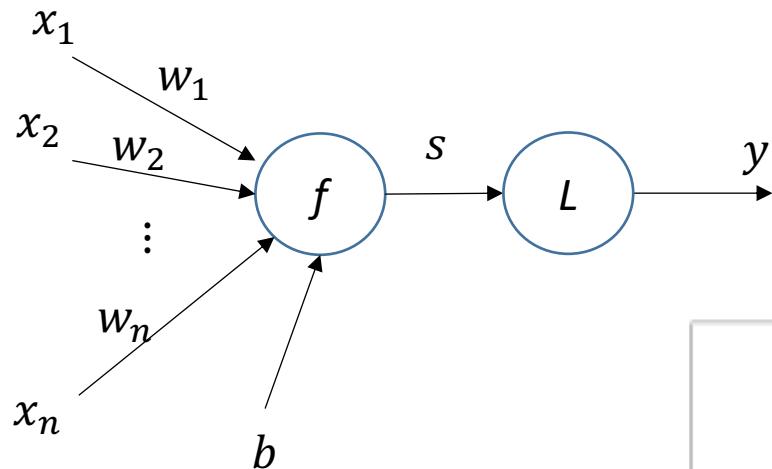


- where $\delta(y)$ is the step function

Artificial Neural Network

- Activation functions
 - Sigmoid activation

$$y = L(s) = \frac{1}{1 + e^{-s}}$$



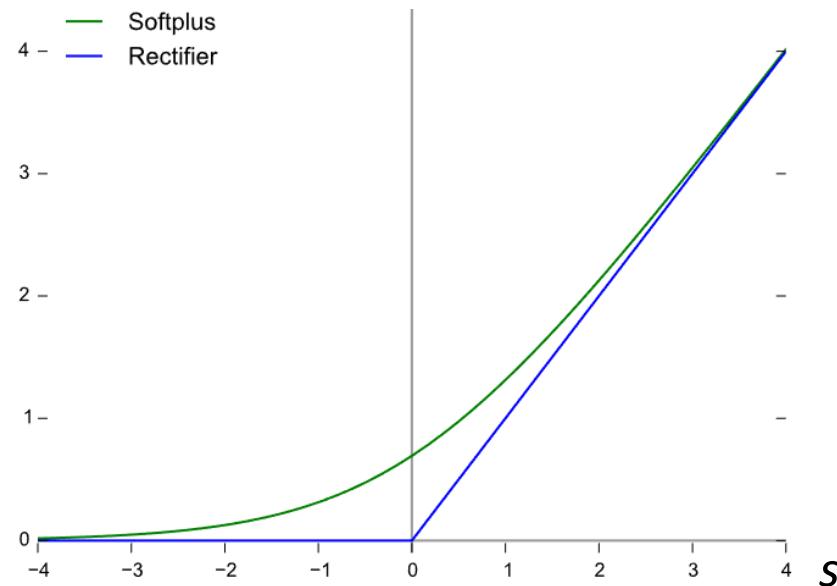
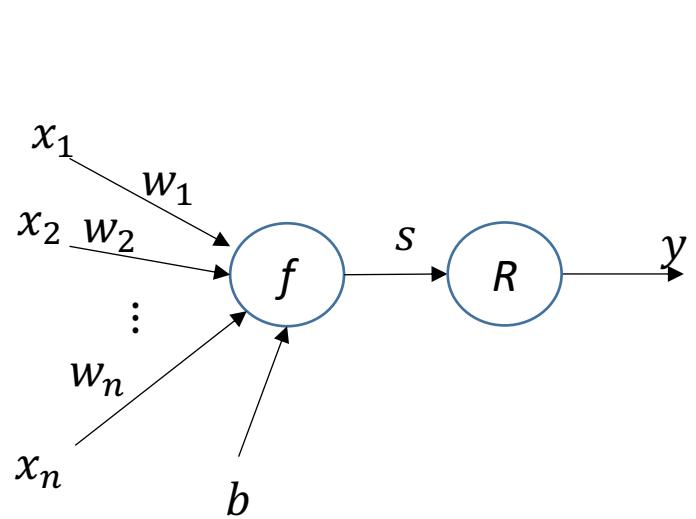
Artificial Neural Network

- Activation functions
 - Rectified linear unit, ReLU (or rectifier):

$$y = R(s) = \max(0, s)$$

- Softplus, a smooth approximation to the ReLU

$$y = SR(s) = \ln(1 + e^s)$$



Artificial Neural Network

- Activation functions
 - Parametric ReLU, PReLU :

$$y = R(s) = \max(\alpha s, s)$$

- where $\alpha < 1$,
- Leaky ReLU, $\alpha = 0.1$

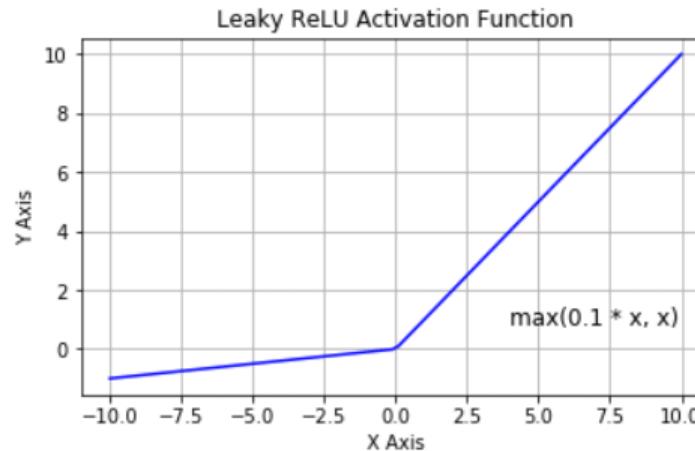
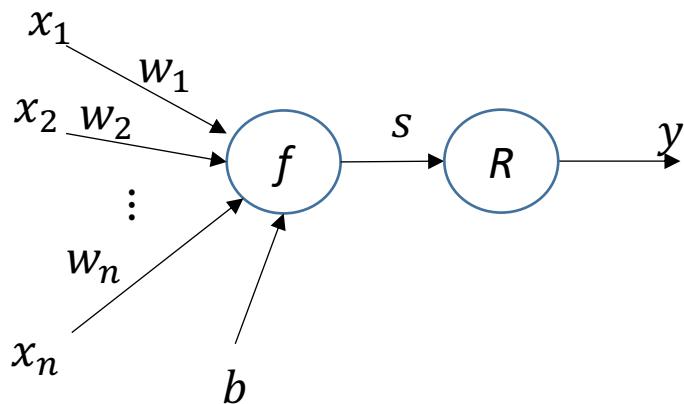


Figure designed by Stefano Romanazzi

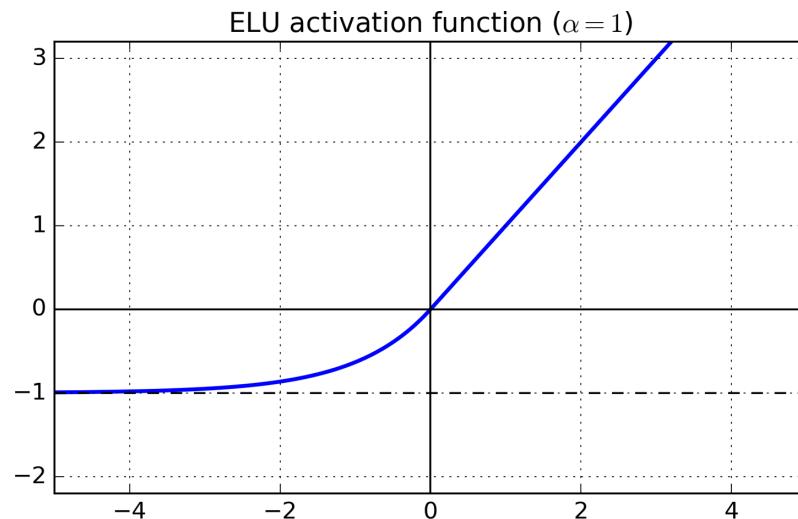
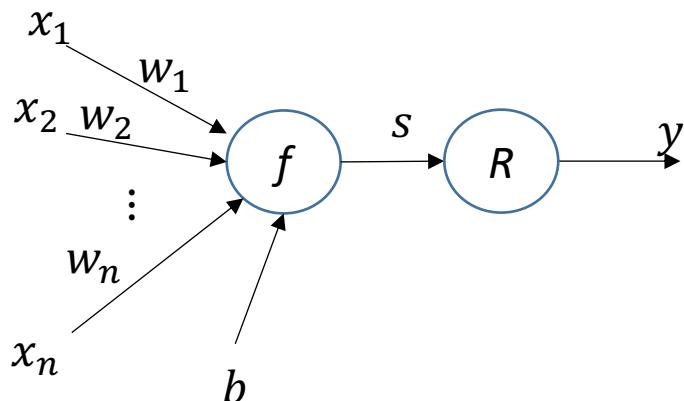
Artificial Neural Network

- Activation functions

- ELU :

$$y = R(s) = \begin{cases} \alpha(e^s - 1) & \text{if } s < 0 \\ s & \text{if } s \geq 0 \end{cases}$$

- typically, $\alpha = 1$



Artificial Neural Network

- Activation functions
 - Concatenated ReLU, CReLU :

$$y = R(s) = [\max(0, s) \quad \max(0, -s)]$$

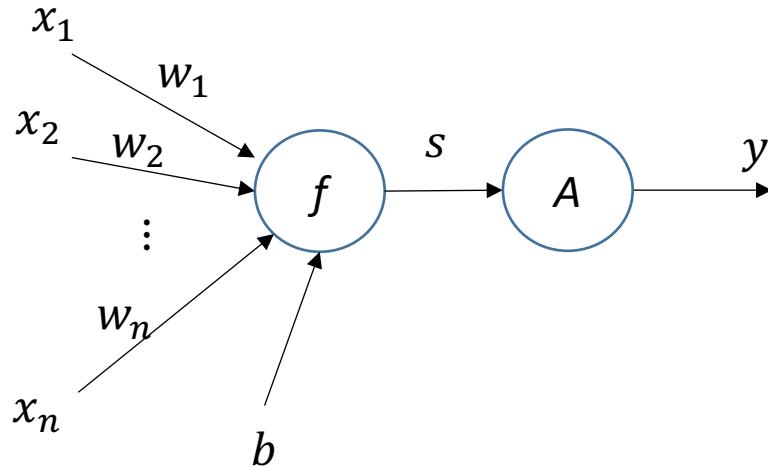
- For example,
 - $s = 1 \rightarrow y = [1, 0]$
 - $s = -1 \rightarrow y = [0, 1]$

Artificial Neural Network

- Finding the best weights and bias so that the error of output is minimum.
- Error-based machine learning model (logistic model or SVM)
- $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, each $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ has a target result t
- $\mathbf{w} = \{w_1, w_2, \dots, w_n\}$

$$\mathbf{w} = \arg \min_{\mathbf{w}'} \sum_{i=1}^m E(t_i, y_i)$$

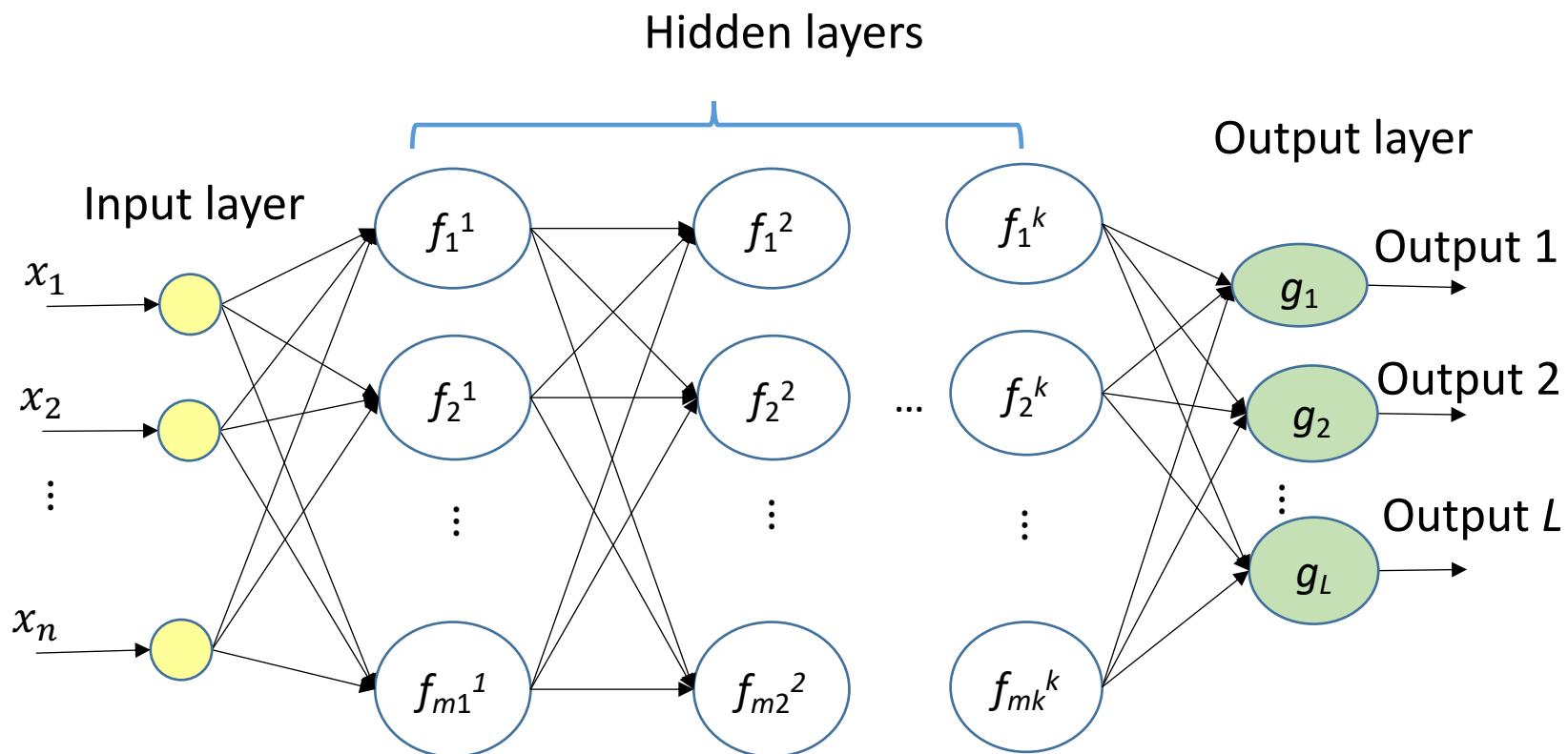
where $E(t, y)$ is the error of t with y .



Artificial Neural Network

- Artificial Neural network, ANN

- Each circle in the hidden layers and output layer is a neuron
- Each edge that goes to a neuron represents an input with a weight
- W. McCulloch and P. Walter, "**A Logical Calculus of Ideas Immanent in Nervous Activity**". *Bulletin of Mathematical Biophysics*. 5 (4): 115–133, 1943.



Artificial Neural Network

- Softmax function
 - To deal with the classification of multiclass

$$P(y_i|\mathbf{x}) = \frac{e^{\mathbf{x}^T \mathbf{w}}}{\sum_{l=1}^L e^{\mathbf{x}^T \mathbf{w}_l}}$$

$$y = \arg \max_{y_i} P(y_i|\mathbf{x})$$

Artificial Neural Network

- Deep neural network, DNN
 - Multiple hidden layers
 - The number of hidden layers ≥ 1
 - Also called deep learning neural network
 - Some problem must be solved by DNN, for example XOR problem
 - Marvin Lee Minsky, Seymour Papert, "**Perceptrons: An Introduction to Computational Geometry**," Mit Press, 1972.

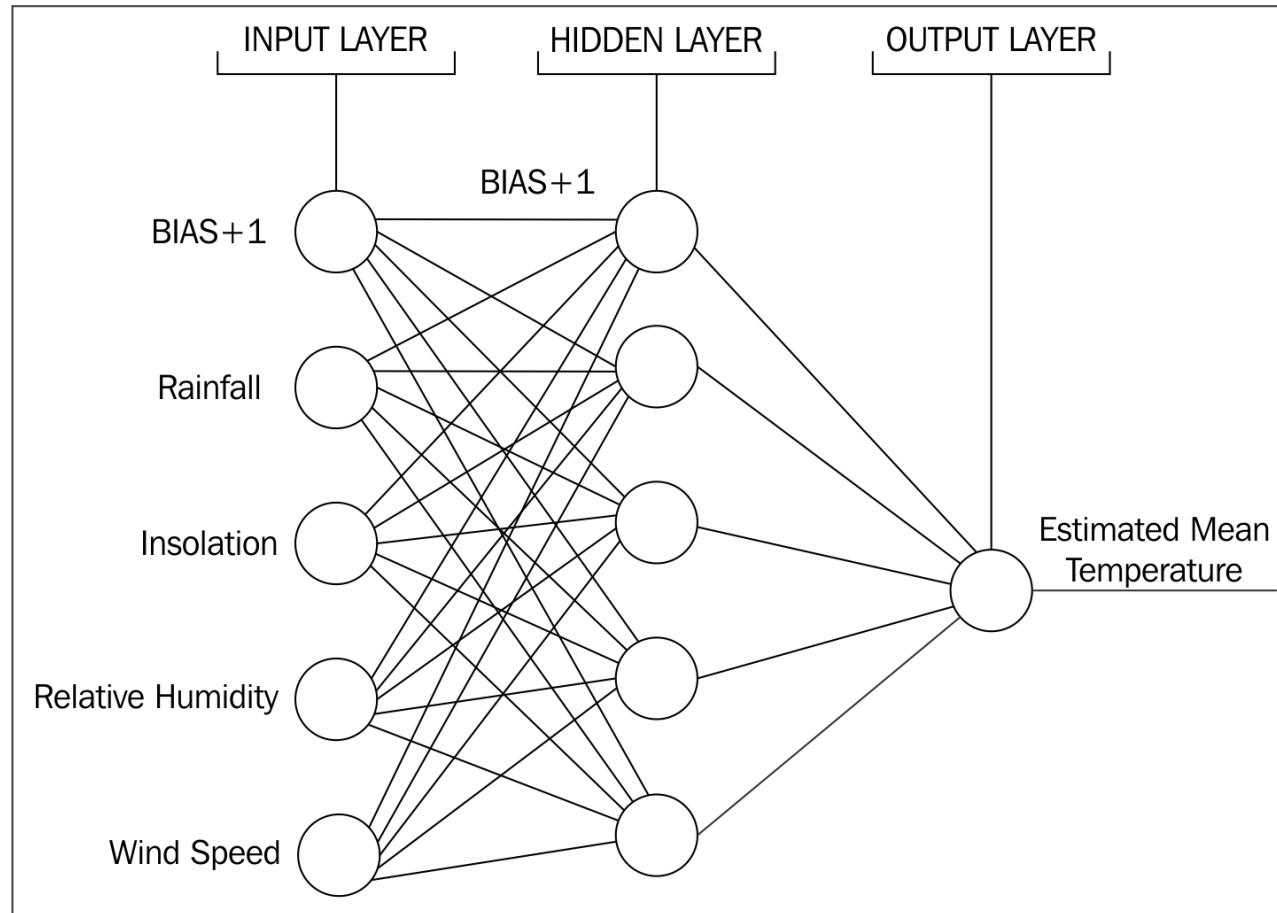
input1	input2	output
1	1	0
1	0	1
0	1	1
0	0	0



Marvin Lee Minsky
1927-2016,
1969 Turing Award winner

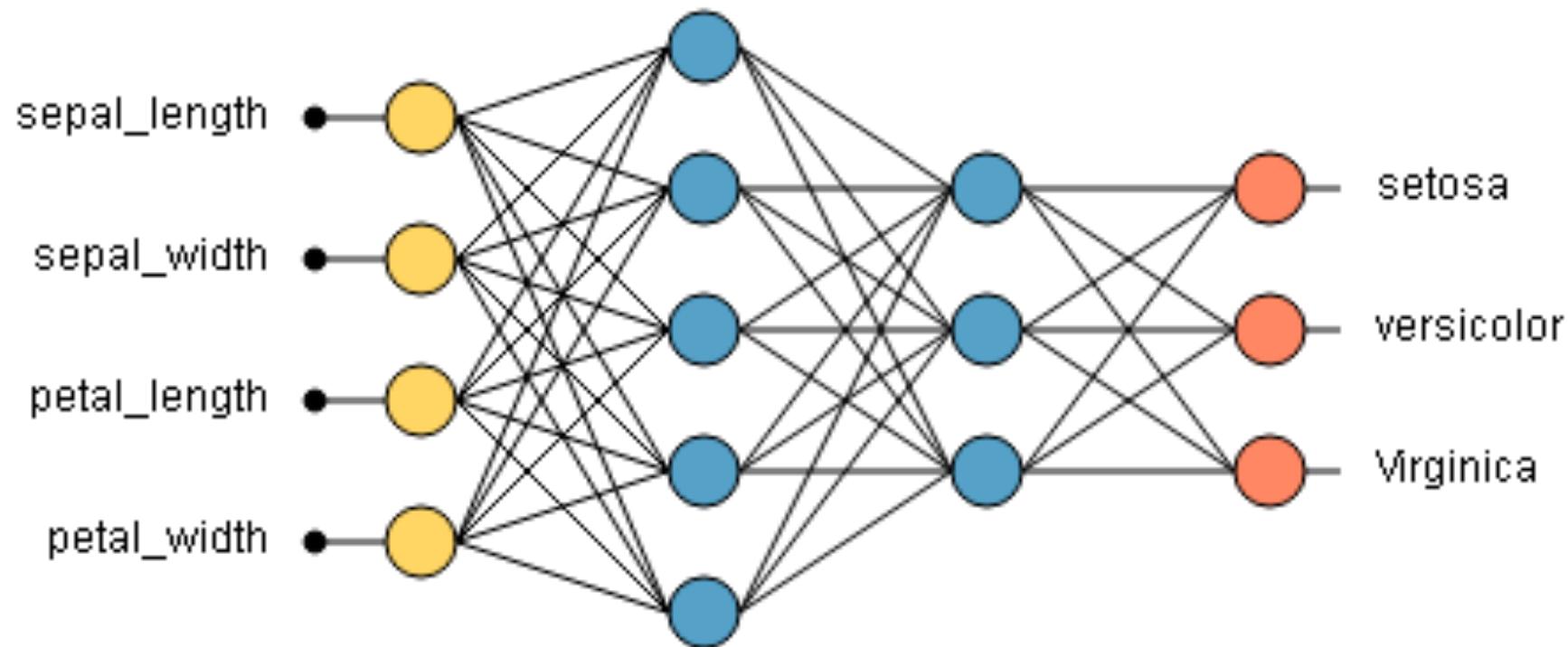
Artificial Neural Network

- Example: Forecasting weather via neural network
 - Fabio Soares and Alan Souza, "Neural Network Programming with Java", Packt, 2016.



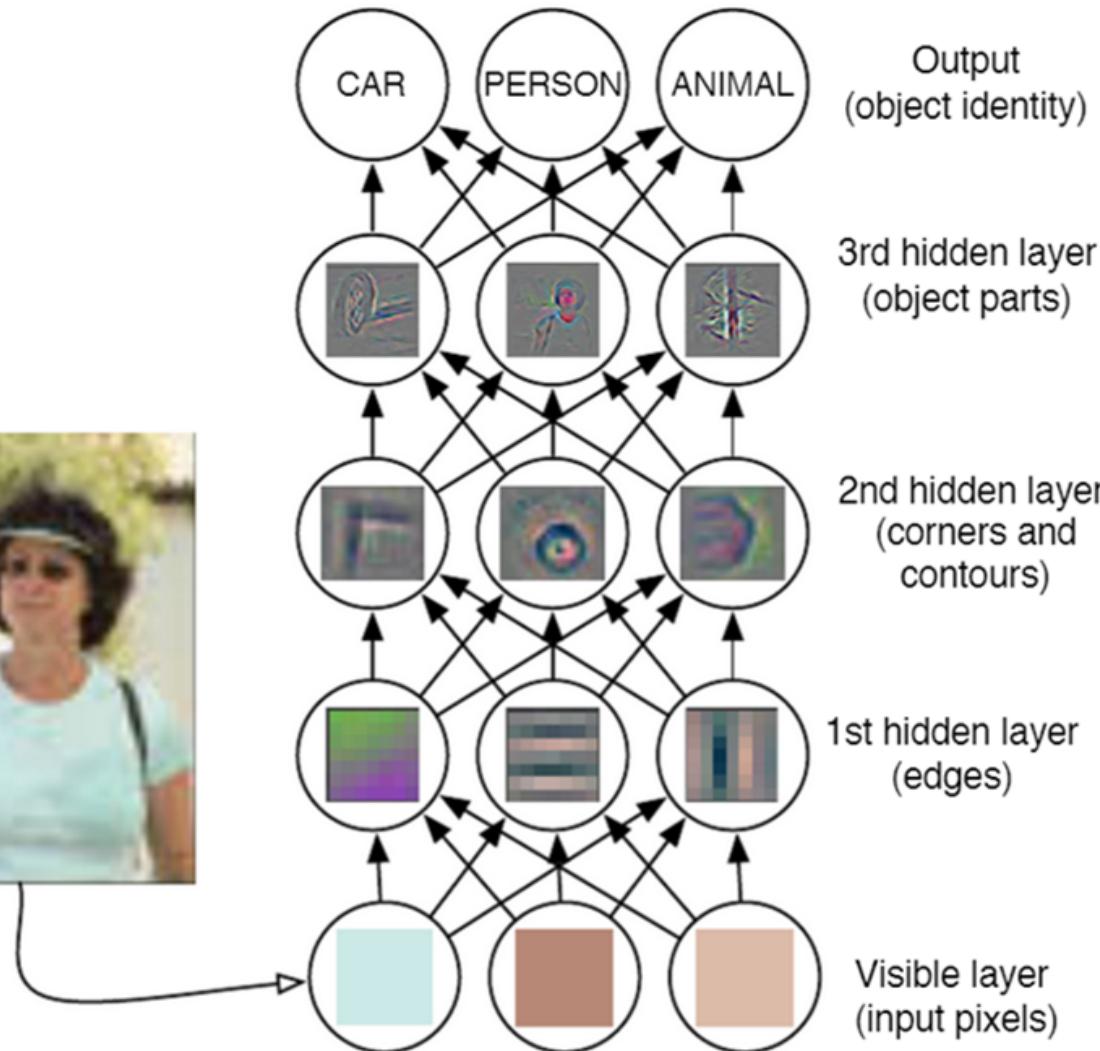
Artificial Neural Network

- Example: Iris flowers
 - Roberto Lopez, "Introduction to neural networks, " Neural Designer.



Artificial Neural Network

- Example: Object recognizing from images
 - I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," *MIT Press*, 2016.

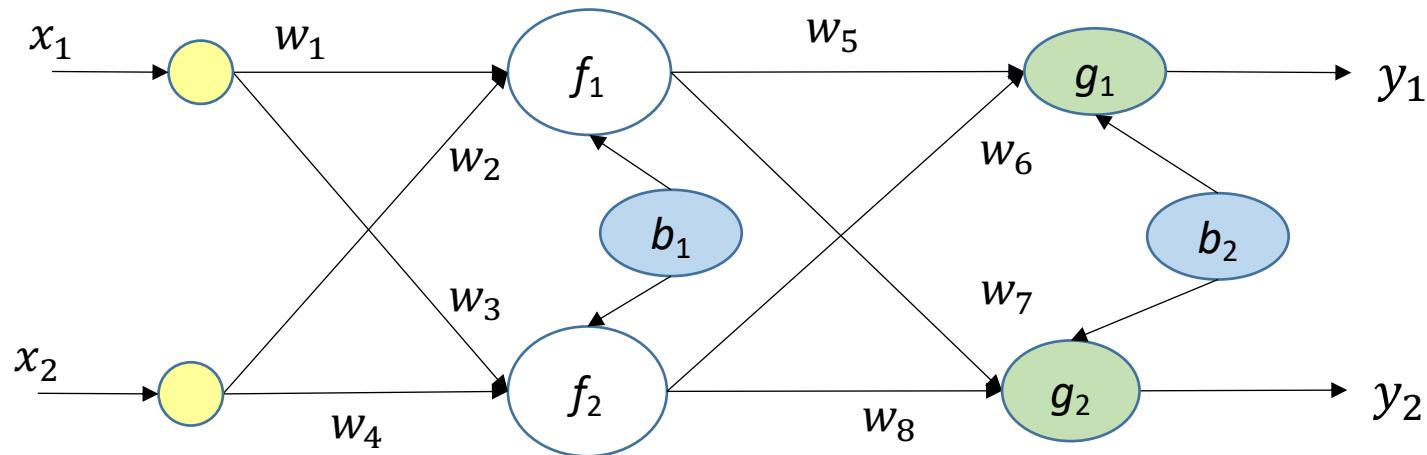


Backpropagation

- How to decide all weights in a neural network?
- We have a training dataset and each training instance has a target result.
- According the errors of the target results with the outputs of neural network to adjust the weights from the last layer to the first layer.

Backpropagation

- Error calculation



$$y_1 = g_1(f_1 w_5 + f_2 w_6 + b_2)$$

$$= g_1((x_1 w_1 + x_2 w_2 + b_1) w_5 + (x_1 w_3 + x_2 w_4 + b_1) w_6 + b_2)$$

$$E = \frac{1}{2} \sum_{i=1}^2 (t_i - y_i)^2 = \sum_{i=1}^2 E(t_i, y_i)$$

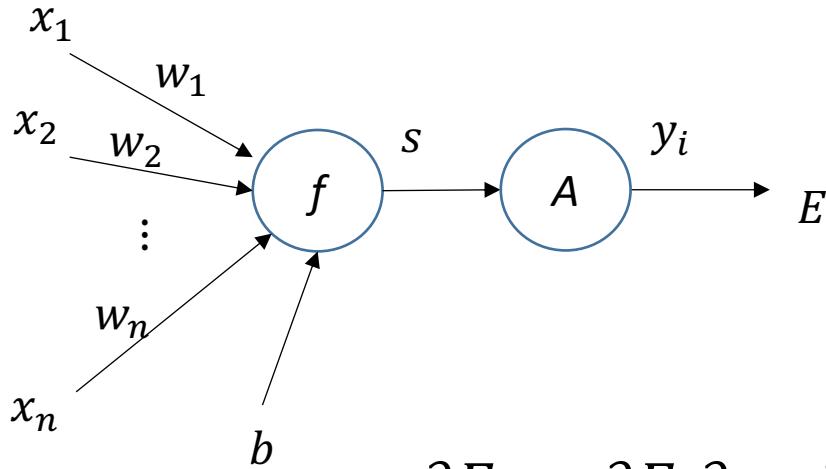
Backpropagation

- From the last layer to the first layer
 1. Estimate the partial derivative of E with respect to w_i .
 2. Then apply gradient descent to update the weight w_i
 3. Repeat step 1.

$$\begin{aligned}\frac{\partial E}{\partial w_5} \Rightarrow & \text{ update } w_5 \Rightarrow \frac{\partial E}{\partial w_6} \Rightarrow \text{ update } w_6 \Rightarrow \\ \frac{\partial E}{\partial w_7} \Rightarrow & \text{ update } w_7 \Rightarrow \frac{\partial E}{\partial w_8} \Rightarrow \text{ update } w_8 \Rightarrow \\ \frac{\partial E}{\partial w_1} \Rightarrow & \text{ update } w_1 \Rightarrow \frac{\partial E}{\partial w_2} \Rightarrow \text{ update } w_2 \Rightarrow \\ \frac{\partial E}{\partial w_3} \Rightarrow & \text{ update } w_3 \Rightarrow \frac{\partial E}{\partial w_4} \Rightarrow \text{ update } w_4\end{aligned}$$

Backpropagation

- For the output layer



$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial s} \frac{\partial s}{\partial w_j}$$

- Applying the gradient descent

$$\Delta w_j = - \frac{\partial E}{\partial w_j}$$

Backpropagation

- Since

$$\frac{\partial E}{\partial y_i} = \frac{\partial(\frac{1}{2}\sum_{i=1}^h(t_i - y_i)^2)}{\partial y_i} = \frac{\partial E(t_i, y_i)}{\partial y_i} = -(t_i - y_i) = y_i - t_i$$

- Assuming that A is a logistic function = $L(s)$

$$\frac{\partial y_i}{\partial s} = \frac{\partial L(s)}{\partial s} = L(s)(1 - L(s))$$

- And

$$\frac{\partial s}{\partial w_j} = \frac{\partial(\sum_{i=1}^n x_i w_i + b)}{\partial w_j} = x_j$$

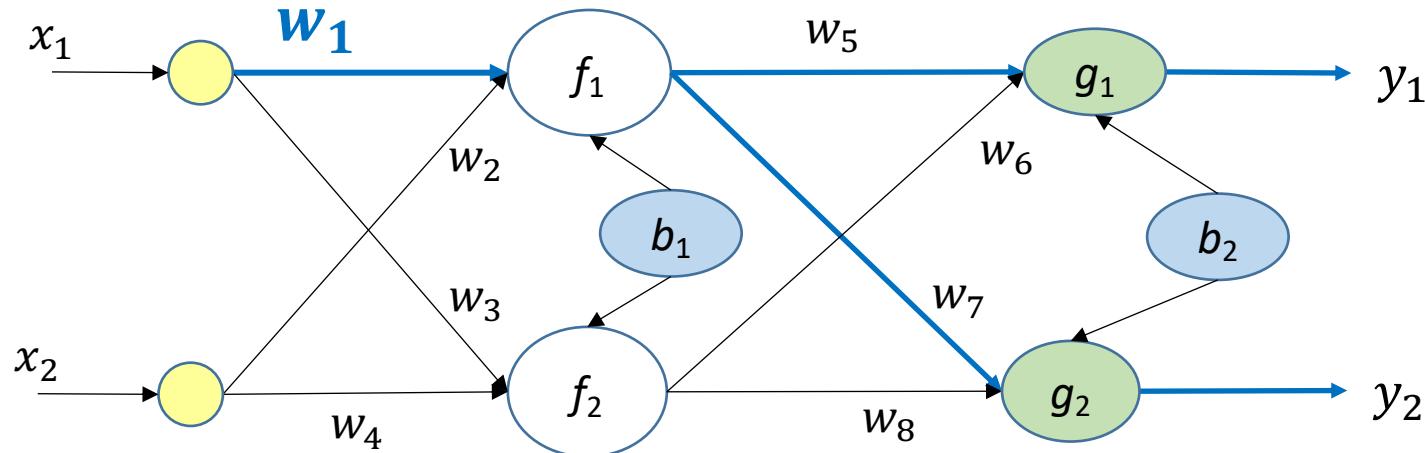
Backpropagation

- Therefore,

$$\Delta w_j = -\frac{\partial E}{\partial w_j} = -(y_i - t_i)L(s)(1 - L(s))x_j$$

Backpropagation

- For hidden layers



$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial f_1} \frac{\partial f_1}{\partial w_1}$$

Chain rule

$$= \left(\frac{\partial E(t_1, y_1)}{\partial f_1} + \frac{\partial E(t_2, y_2)}{\partial f_1} \right) \frac{\partial f_1}{\partial w_1}$$

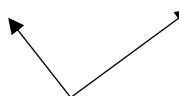
Backpropagation

- For hidden layers

$$\frac{\partial E(t_1, y_1)}{\partial f_1} = \frac{\partial s_1}{\partial f_1} \frac{\partial E(t_1, y_1)}{\partial s_1}$$

$$= \frac{\partial(f_1 w_5 + f_2 w_6 + b_2)}{\partial f_1} \frac{\partial E(t_1, y_1)}{\partial s_1}$$

$$= w_5 \frac{\partial E(t_1, y_1)}{\partial y_1} \frac{\partial y_1}{\partial s_1}$$



These are calculated before

Backpropagation

- A neuron in hidden layer may has the logistic activation

$$\frac{\partial f_1}{\partial w_1} = \frac{\partial f_1}{\partial r_1} \frac{\partial r_1}{\partial w_1} = L(r_1)(1 - L(r_1))x_1$$

- Therefore,

$$\begin{aligned}\Delta w_1 &= -\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial f_1} \frac{\partial f_1}{\partial w_1} \\ &= \left(\frac{\partial E(t_1, y_1)}{\partial f_1} + \frac{\partial E(t_2, y_2)}{\partial f_1} \right) L(r_1)(1 - L(r_1))x_1\end{aligned}$$

- where

$$\begin{aligned}\frac{\partial E(t_1, y_1)}{\partial f_1} &= w_5 \frac{\partial E(t_1, y_1)}{\partial y_1} \frac{\partial y_1}{\partial s_1} \\ \frac{\partial E(t_2, y_2)}{\partial f_1} &= w_7 \frac{\partial E(t_2, y_2)}{\partial y_2} \frac{\partial y_2}{\partial s_1}\end{aligned}$$

Backpropagation

- Vanishing gradient problem
 - In the final layer k , $\frac{\partial E}{\partial w_j^k}$ is large \rightarrow OK!
 - In $(k - 1)$ th layer, $\frac{\partial E}{\partial w_j^{k-1}}$ is smaller than $\frac{\partial E}{\partial w_j^k}$ \rightarrow Still OK!
 - ...
 - In the first layer, $\frac{\partial E}{\partial w_j^1}$ is closed to zero \rightarrow Not OK!
 - The sigmoid activation may cause this problem

$$0 \leq \frac{\partial L(s)}{\partial s} = L(s)(1 - L(s)) \leq 0.25$$

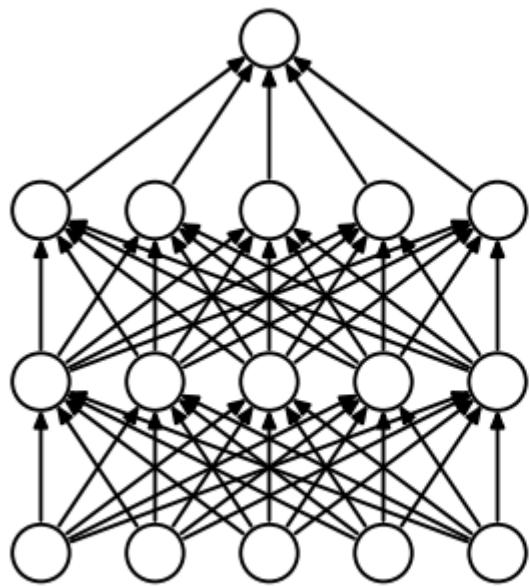
- Using ReLU can solve this problem

$$\frac{\partial R(s)}{\partial s} = \begin{cases} 1 & s \geq 0 \\ 0 & s < 0 \end{cases}$$

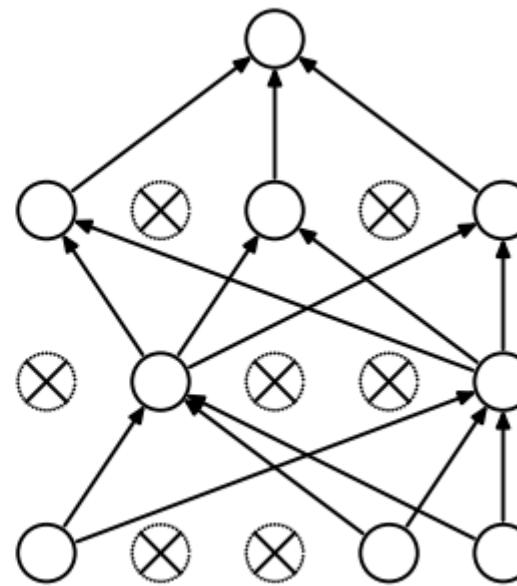
Dropout

- N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," JMLR, vol. 15, pp. 1929-1958, 2014.
- Some problems of a neural network
 - High computational cost
 - Overfitting
- At each round of training
 - Each unit is retained with a probability p .
 - Typically, p is 0.5
- At test time
 - The network is used as a whole.
 - The weights are multiplied by a factor of p .

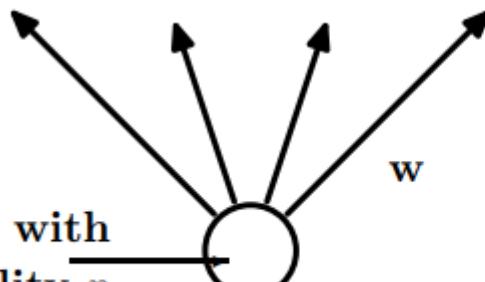
Dropout



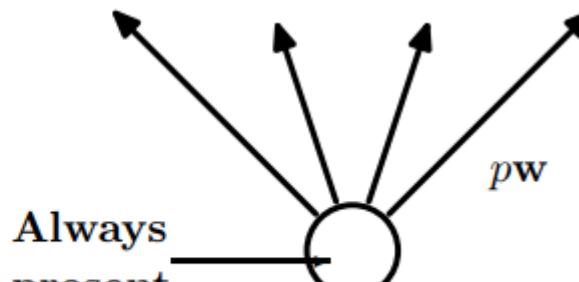
(a) Standard Neural Net



(b) After applying dropout.



(a) At training time



(b) At test time

Early Stopping

- When training a learner with a iterative method
 - Too few epochs → underfitting
 - Too many epochs → overfitting
- Stop the learner from overfitting
- Validation-based early stopping
 1. Split the training data into a training set and a validation set, e.g. in a 2-to-1 proportion.
 2. Train only on the training set and evaluate the per-example error on the validation set once in a while, e.g. after every fifth epoch.
 3. Stop training as soon as the error on the validation set is higher than it was the last time it was checked.
 4. Use the weights the network had in that previous step as the result of the training run.

DNN in TensorFlow

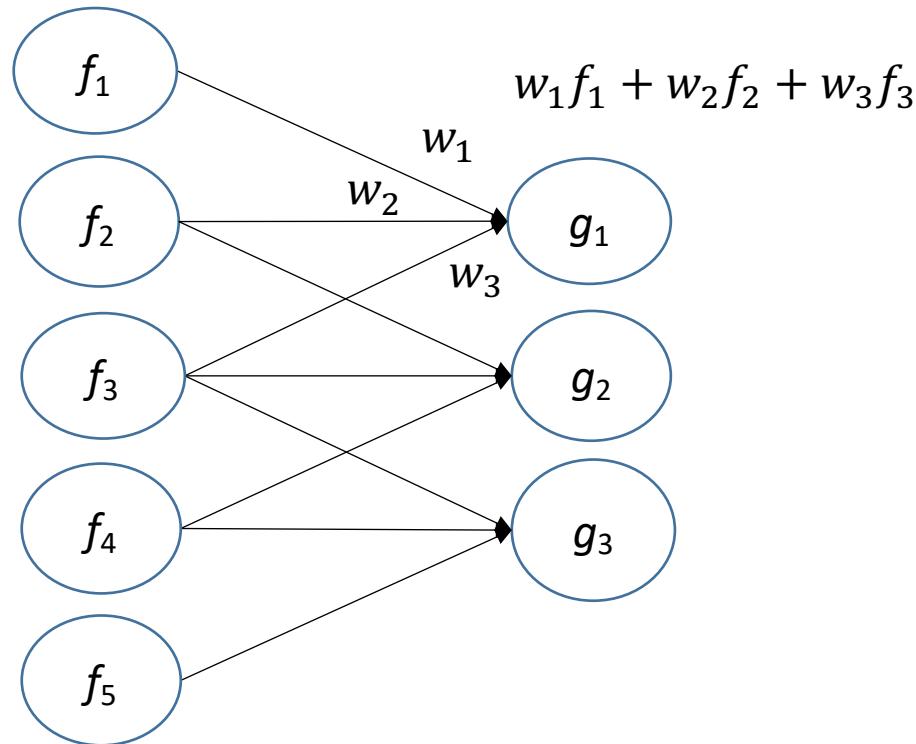
- An example of MNIST:
 - https://github.com/jameschengcs/ml/blob/master/dnn_mnist.py

Convolutional Neural Network, CNN

- Fully-connected neural network
 - In a fully connected layer, each neuron is connected to every neuron in the previous layer, and each connection has its own weight.
 - General purpose connection pattern
 - No assumptions about the features
 - High consumption of memory (weights) and computation (connections).

Convolutional Neural Network, CNN

- In a convolutional layer
 - Each neuron only connected from a set of related neurons in the previous layer
 - Locality relation
 - Shared weights
 - Each neuron using the same weights for the input

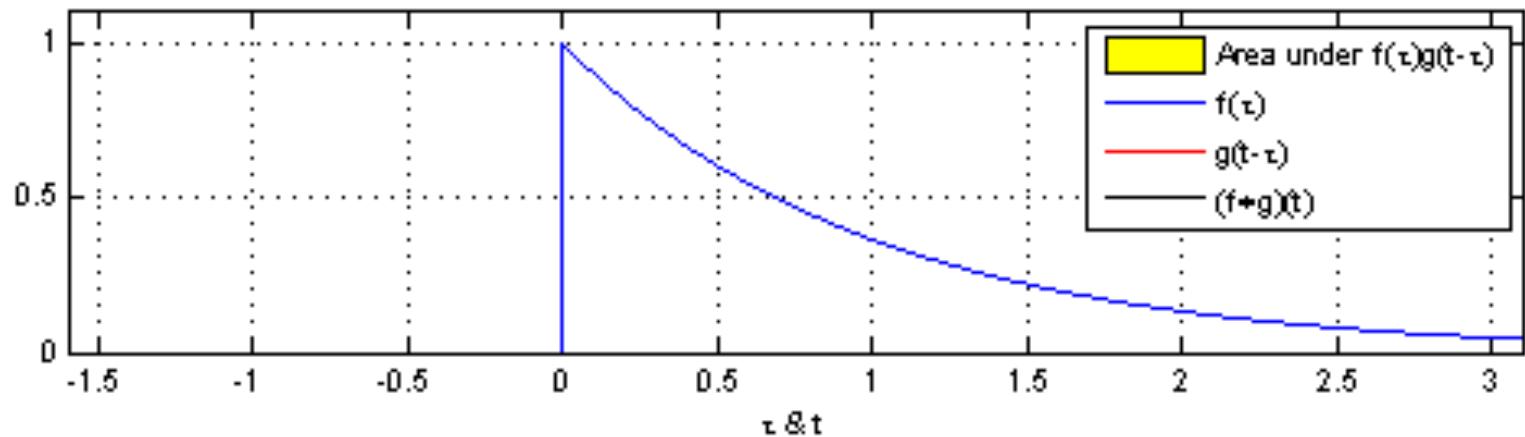


Convolutional Neural Network, CNN

- Convolution

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

- g is called **convolution kernel, operator, or mask**.



Convolutional Neural Network, CNN

- Discrete convolution

$$(f * g)[n] = \sum_{m=-M}^{M} f[n - m]g[m]$$

- where M is the index range of g
- example:
 - $f[-3: 3] = \{1, 9, 0, 5, 2, 4, 3\}$
 - $g [-1: 1] = \{-2, 0, 2\}$
 - $(f * g)[2] = f[2 - (-1)]g[-1] + f[2 - 0]g[0] + f[2 - 1]g[1]$
 $= f[3]g[-1] + f[2]g[0] + f[1]g[1]$
 $= 3(-2) + 4(0) + 2(2)$
 $= -2$

Convolutional Neural Network, CNN

- 2D discrete convolution

$$(f * g)[m][n] = \sum_{y=-H}^H \sum_{x=-W}^W f[m - y][n - x]g[y][x]$$

$f[-3:2][-3:2]$

A 6x6 grid representing the input matrix $f[-3:2][-3:2]$. A 3x3 subgrid in the center is highlighted in yellow, representing the receptive field of the central element in the kernel. The grid contains the following values:

1	4	5	8	2	1
0	3	1	5	7	2
1	2	1	4	7	1
2	2	0	3	6	2
0	1	7	3	5	3
1	3	4	1	2	4

The highlighted 3x3 subgrid is:
Row 2, Columns 2-4: 1, 5, 7
Row 3, Columns 2-4: 1, 4, 7
Row 4, Columns 2-4: 0, 3, 6

$g[-1:1][-1:1]$

A 3x3 grid representing the kernel matrix $g[-1:1][-1:1]$. The grid contains the following values:

-1	0	1
-2	0	2
-3	0	3

$$(f * g)[0][0] = -45$$

A 3x3 grid representing the output of the convolution step. The grid contains the following values, with the original input values from f enclosed in parentheses:

1(3)	5(0)	7(-3)
1(2)	4(0)	7(-2)
0(1)	3(0)	6(-1)

Convolutional Neural Network, CNN

- Zero padding

$$f[-3:2][-3:2] \rightarrow f[-4:3][-4:3]$$

0	0	0	0	0	0	0	0	0
0	1	4	5	8	2	1	0	
0	0	3	1	5	7	2	0	
0	1	2	1	4	7	1	0	
0	2	2	0	3	6	2	0	
0	0	1	7	3	5	3	0	
0	1	3	4	1	2	4	0	
0	0	0	0	0	0	0	0	

$$g[-1:1][-1:1]$$

-1	0	1
-2	0	2
-3	0	3

$$(f * g)[2][2] = 11$$

0(3)	0(0)	0(-3)
2(2)	1(0)	0(-2)
7(1)	2(0)	0(-1)

Convolutional Neural Network, CNN

- Commonly-used 2D kernels

- Mean $g(x, y) = \frac{1}{(2H+1) \times (2W+1)}$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

- Gaussian $g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Convolutional Neural Network, CNN

- Commonly-used 2D kernels

- X derivative, $g_x =$

a	0	-a
b	0	-b
a	0	-a

$$a = b = 1$$

1	0	-1
1	0	-1
1	0	-1

- Y derivative , $g_y =$

-a	-b	-a
0	0	0
a	b	a

$$a = 2, b = 5$$

-2	-5	-2
0	0	0
2	5	2

- Edge detection: $| (f * g_x), (f * g_y) |$
 - Sobel operator

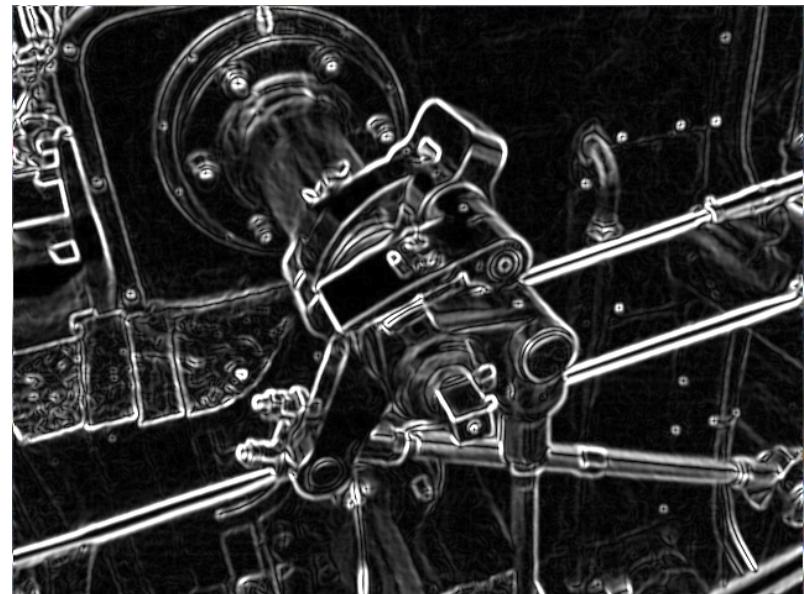
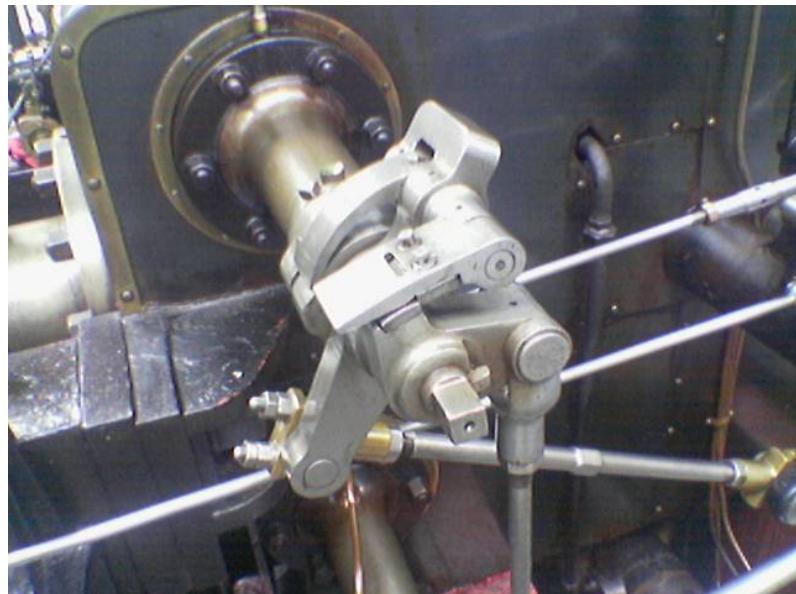
Convolutional Neural Network, CNN

- Convolution in image processing
 - Filtering
 - Gaussian filtering



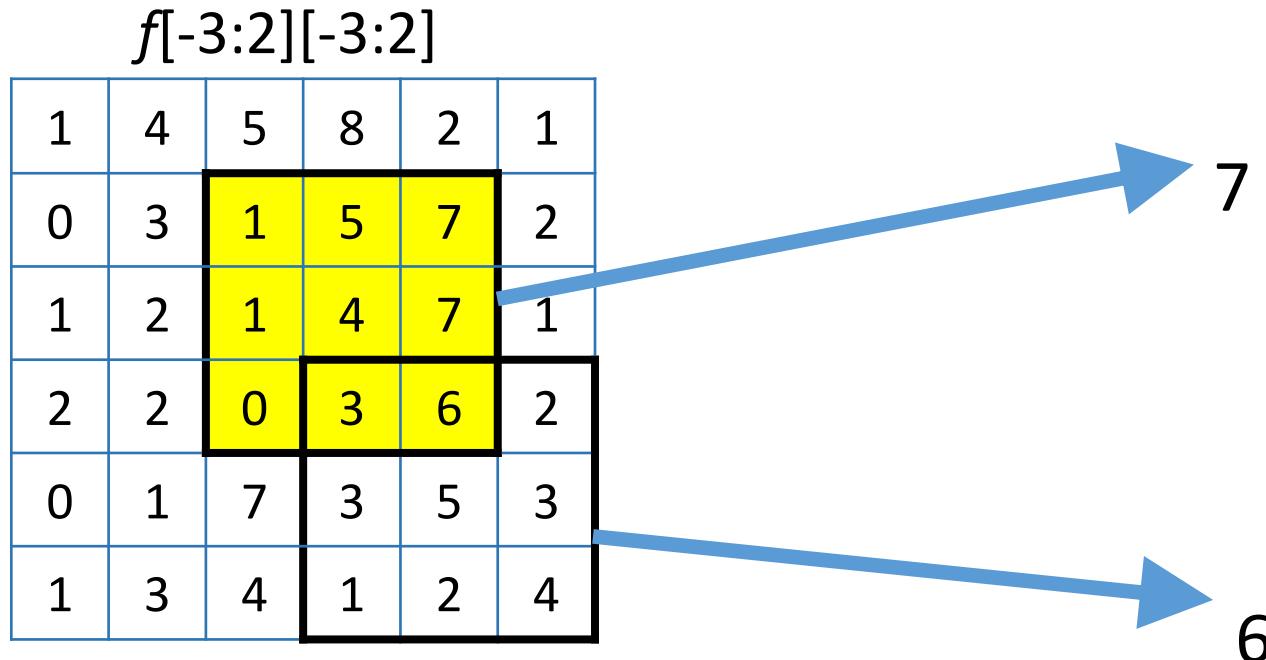
Convolutional Neural Network, CNN

- Convolution in image processing
 - Edge detection (Sobel operator)



Convolutional Neural Network, CNN

- Max pooling
 - Finding the maximum from a masked area



Convolutional Neural Network, CNN

- Stride
 - The step of the convolution operation
 - It affects the output size

Input size: 6 x 6

Kernel size: 3 x 3 (max pooling)

Stride: 1

1	4	5	8	2	1
0	3	1	5	7	2
1	2	1	4	7	1
2	2	0	3	6	2
0	1	7	3	5	3
1	3	4	1	2	4

Output size: 4 x 4

5	8	8	8
3	5	7	7
7	7	7	7
7	7	7	6

Convolutional Neural Network, CNN

- Stride

Input size: 6×6

Kernel size: 3×3 (max pooling)

Stride: 3

1	4	5		8	2	1
0	3	1		5	7	2
1	2	1		4	7	1
2	2	0		3	6	2
0	1	7		3	5	3
1	3	4		1	2	4

Output size: 2×2

5	8
7	6

Convolutional Neural Network, CNN

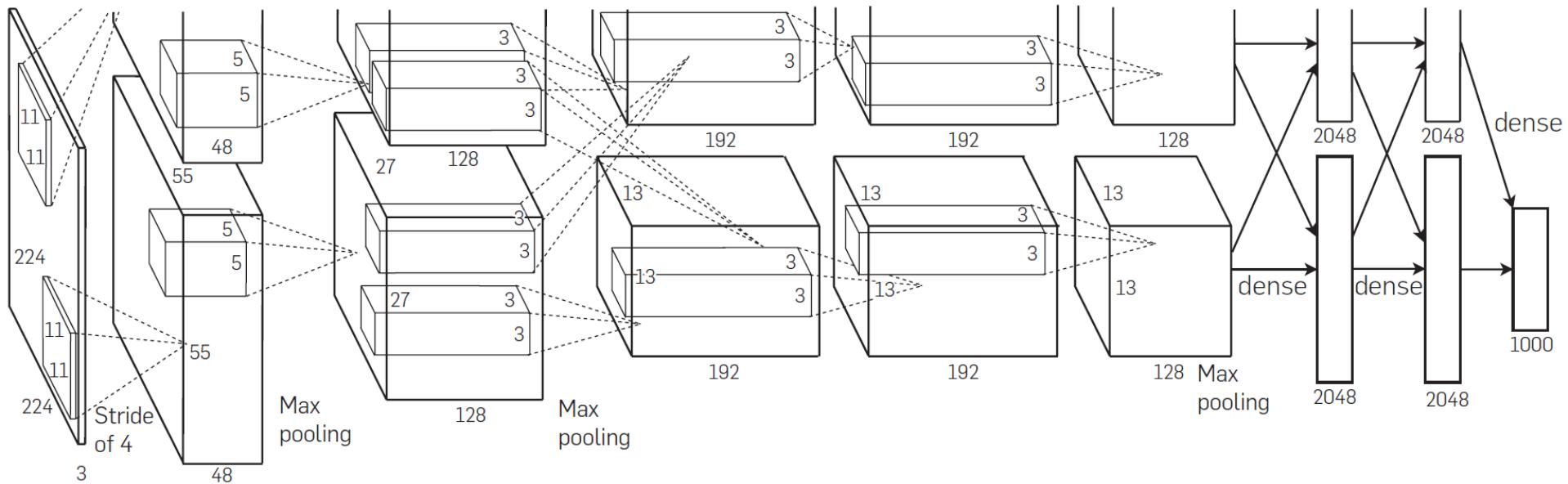
- Example 1: Image classification
 - Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. **ImageNet classification with deep convolutional neural networks.** *Commun. ACM* 60, 6 (May 2017), 84-90.
 - Using CNN to classify the 1.2 million high-resolution images



koala	tiger	container ship	motor scooter
wombat	tiger	container ship	motor scooter
Norwegian elkhound	tiger cat	container ship	motor scooter
wild boar	tiger cat	lifeboat	go-kart
wallaby	jaguar	amphibian	moped
koala	lynx	fireboat	bumper car
	leopard	drilling platform	golfcart

Convolutional Neural Network, CNN

- Example 1: Image classification



- In the first convolution layer
 - Number of kernels: 96
 - Kernel size: 11 x 11
 - Stride: 4 →(0, 4, 8, ..., 220)
 - Zero padding
 - output size: 55 x 55 x 96

Convolutional Neural Network, CNN

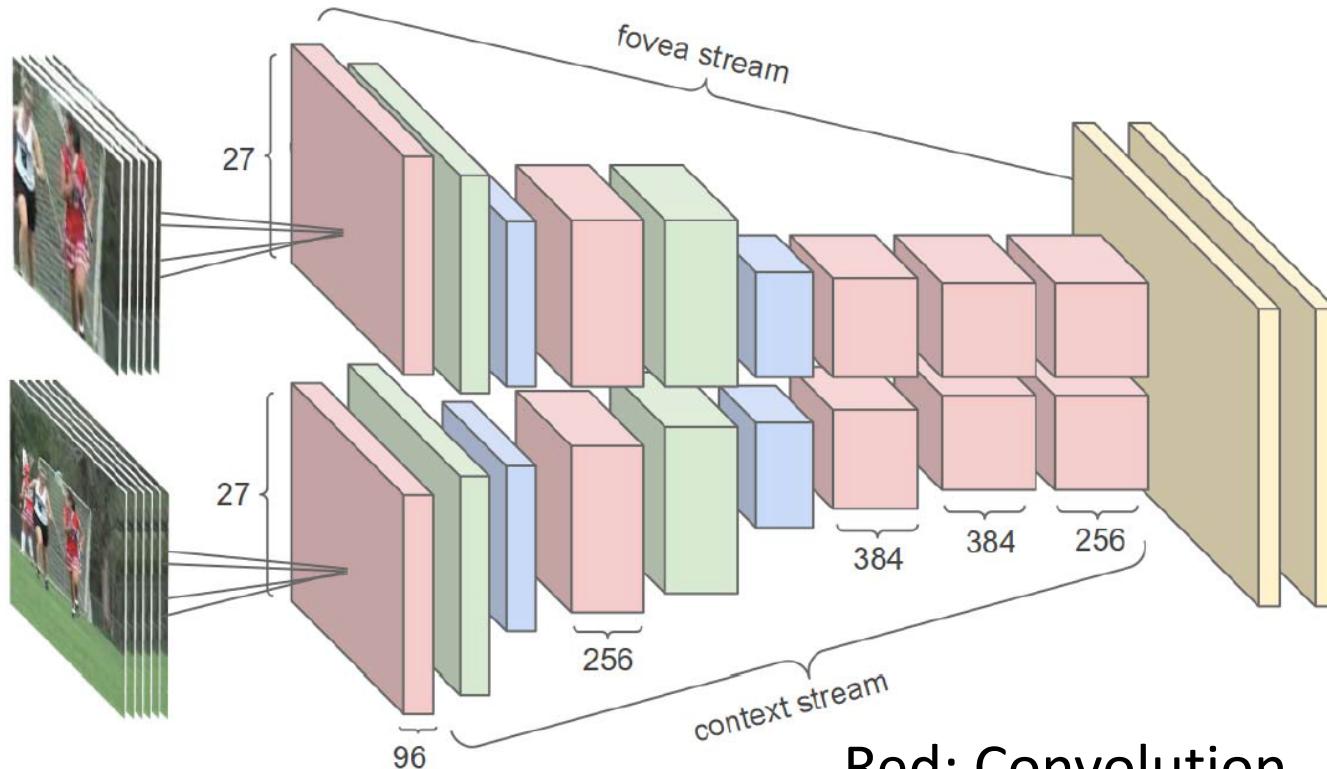
- Example 2: Video classification
 - A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, "**Large-Scale Video Classification with Convolutional Neural Networks**," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 1725-1732.
 - 1 million videos → 487 categories



short track motor racing
single frame predictions:
short track motor racing
touring car racing
drifting (motorsport)
motorcycle racing
time attack
motion-aware predictions:
dirt track racing
drifting (motorsport)
stock car racing
rallycross
auto racing

Convolutional Neural Network, CNN

- Example 2: Video classification



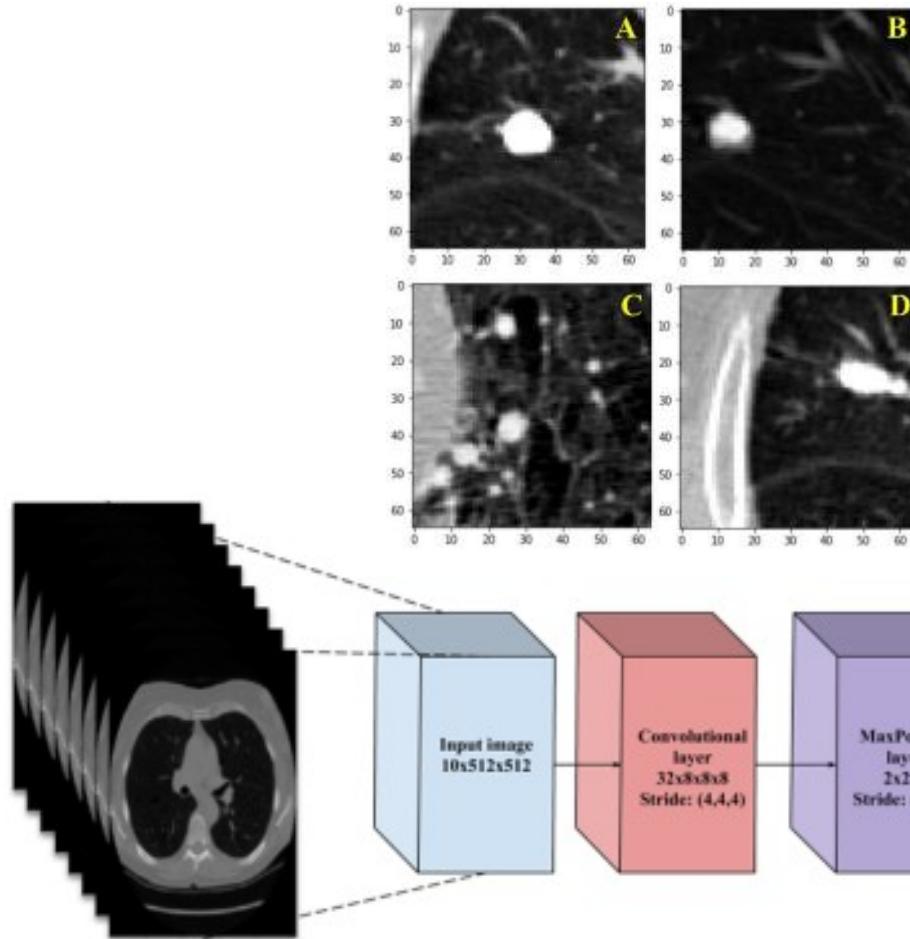
Red: Convolution
Green: Normalization
Blue: Pooling
Yellow: Fully connected layers

Convolutional Neural Network, CNN

- Example 2: Video classification
 - Input size: 178 x 178 pixels per frame
 - Input frames are fed into two separate streams
 - Context stream that models low-resolution image
 - Downsampled to 89 x 89 pixels
 - Fovea stream that processes high-resolution center crop.
 - the middle portion of a frame.
 - Both streams consist of alternating convolution (red), normalization(green), and pooling layers(blue).
 - Both streams converge to two fully connected layers (yellow).

Convolutional Neural Network, CNN

- Example 3. Lung nodule detection from low-dose CT
 - Ali, Issa et al. "Lung Nodule Detection via Deep Reinforcement Learning." *Frontiers in Oncology* 8 (2018): 108. PMC. Web. 23 Aug. 2018.



Dataset:

LUNA

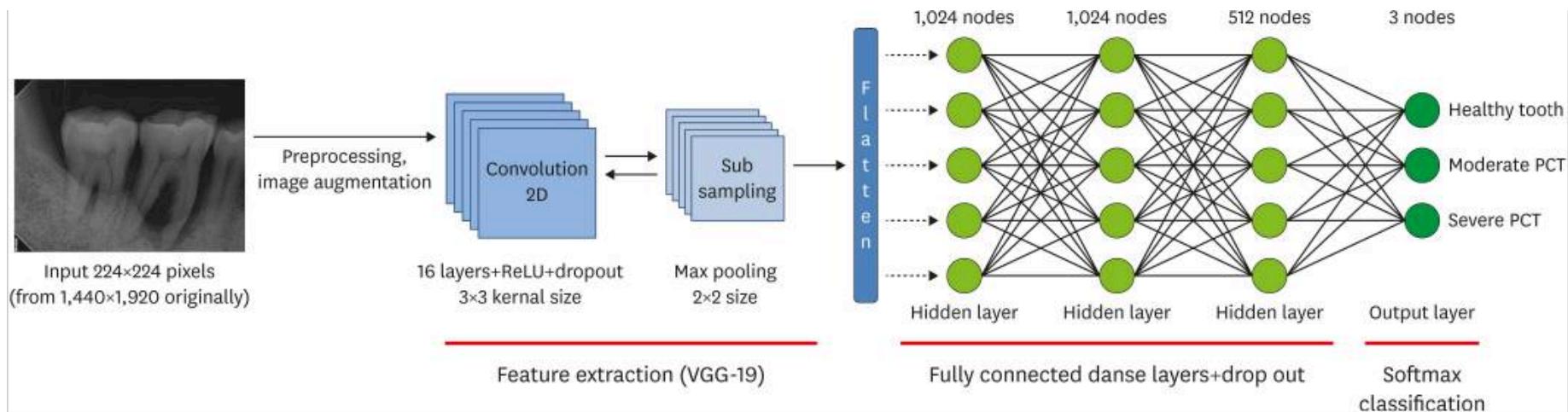
<https://luna16.grand-challenge.org/>

RIDER

<https://wiki.nci.nih.gov/display/CIP/RIDER>

Convolutional Neural Network, CNN

- Example 4
- J.H. Lee, et al., "Diagnosis and prediction of periodontally compromised teeth using a deep learning-based convolutional neural network algorithm," *J Periodontal Implant Sci*, 48(2): pp. 114 - 123, Apr. 2018.



Convolutional Neural Network, CNN

- Example 5: Speech recognition
 - O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn and D. Yu, "**Convolutional Neural Networks for Speech Recognition**," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533-1545, Oct. 2014.

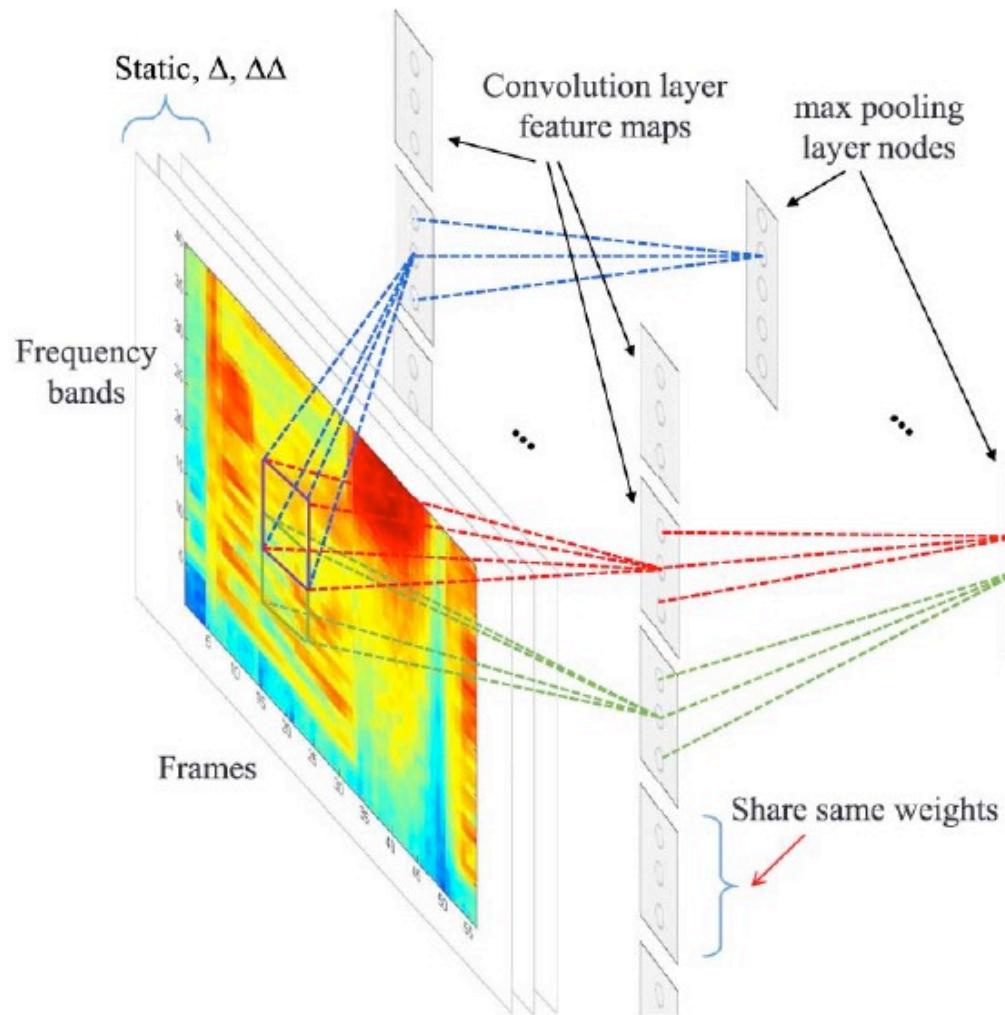
	No PT	With PT
DNN	37.1%	35.4%
CNN	34.2%	33.4%

WER: Word error rate

PT: Pretraining

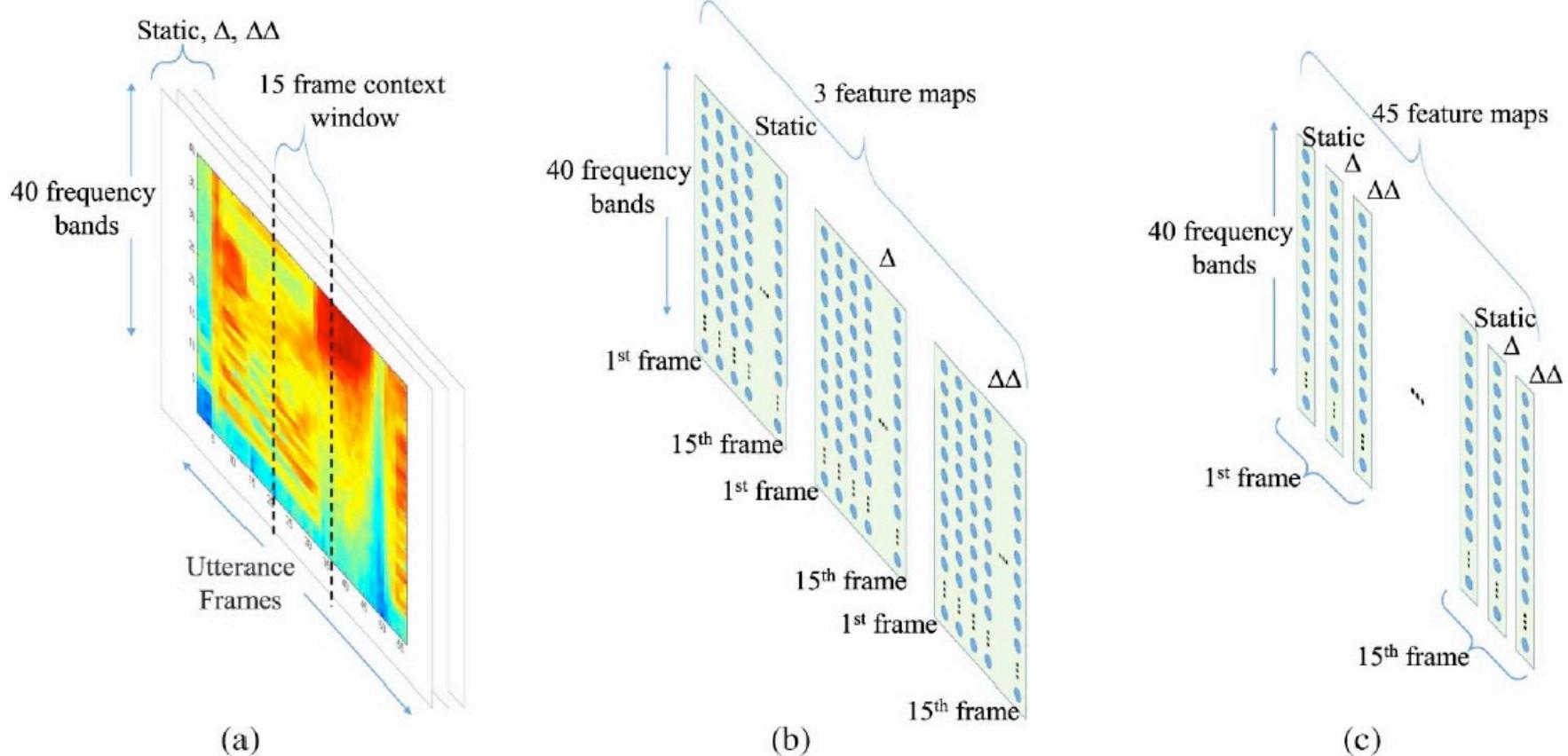
Convolutional Neural Network, CNN

- Example 5: Speech recognition
 - Structure



Convolutional Neural Network, CNN

- Example 5: Speech recognition
 - Input data



Fourier Transform

- 1D FT

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i ux} dx$$

$$f(x) = \int_{-\infty}^{\infty} F(u)e^{2\pi i ux} du$$

- Discrete 1D FT

- For N numbers:

$$F[u] = \sum_{x=0}^{N-1} f[x]e^{-2\pi i \frac{ux}{n}}$$

$$f[x] = \sum_{u=0}^{N-1} F[u]e^{2\pi i \frac{ux}{n}}$$

$$i = \sqrt{-1}$$

Euler's formula:

$$e^{ix} = \cos x + i \sin x$$

Euler's identity:

$$e^{i\pi} + 1 = 0$$

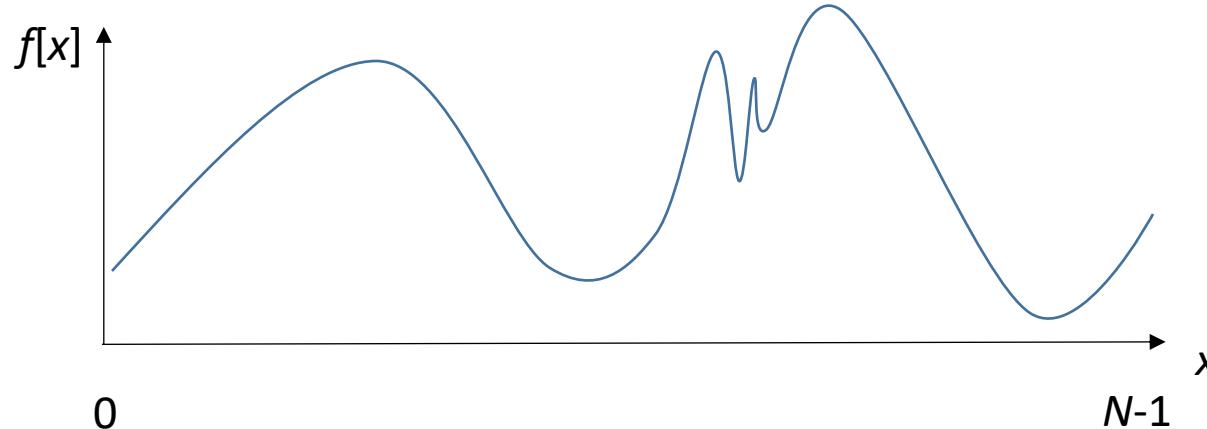
Fourier Transform

- The purpose of FT
 - Any periodic signal can be decomposed by many sine and cosine waves with different frequencies and amplitudes.
 - The frequency is called **angular frequency**
 - The unit is **radian** rather than Hz
 - The range of angular freq. is $[-\pi, \pi]$

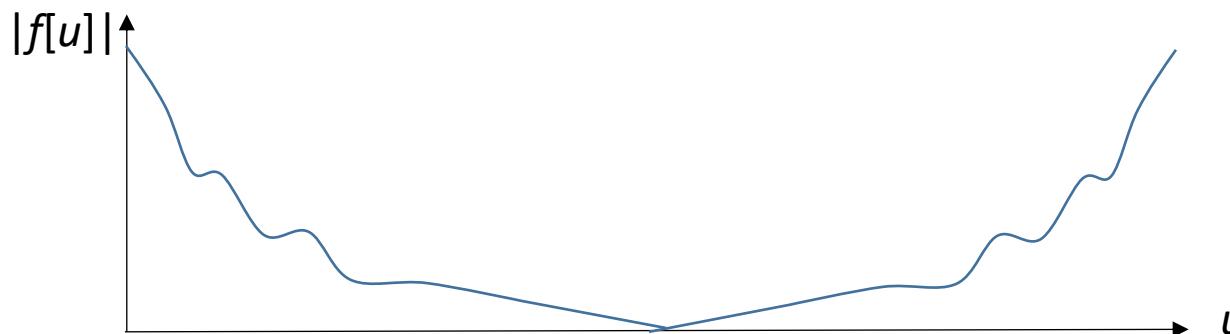


Fourier Transform

- Time domain $f[x]$



- Frequency domain $F[u]$



0
Lowest freq.

$N/2$
 $= \pi$
Highest freq.

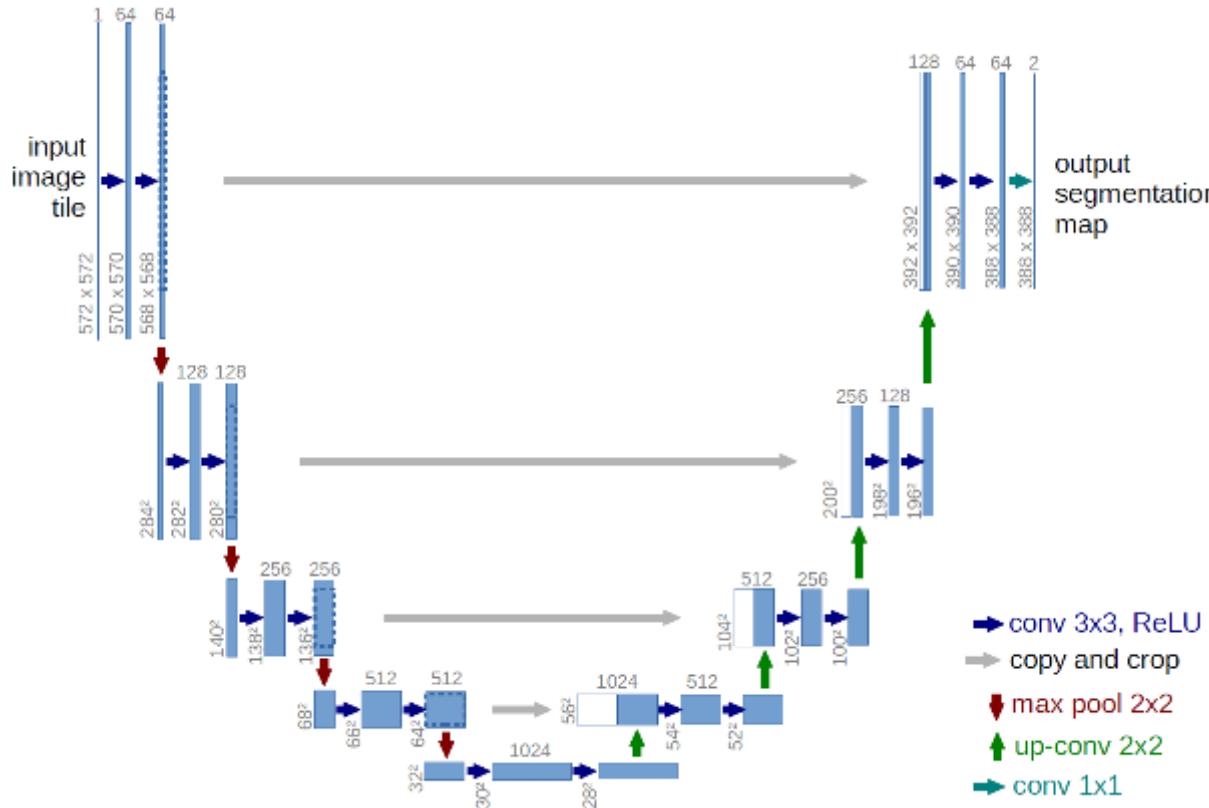
$N - 1$
 $= 2\pi - 2\pi/N$
Lowest freq.

CNN in Tensorflow

- An example of MNIST
 - https://github.com/jameschengcs/ml/blob/master/cnn_mnist.py

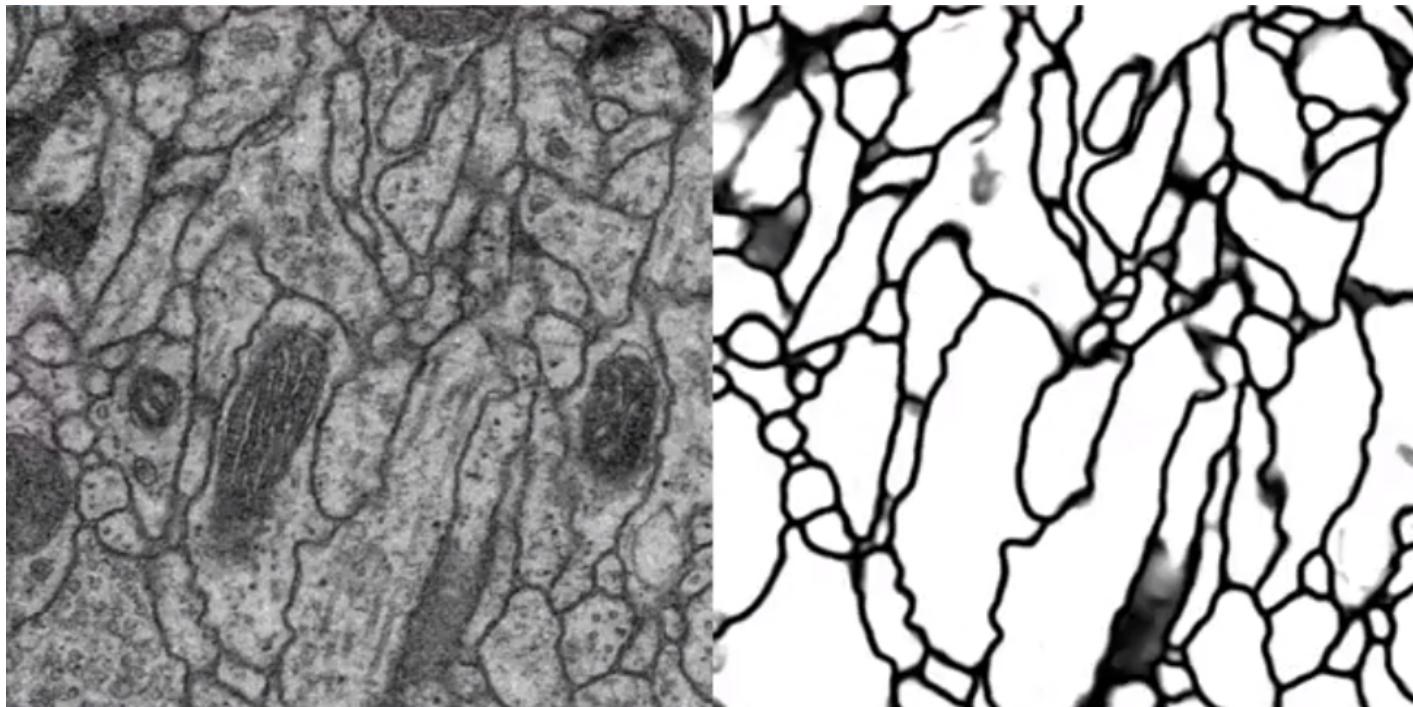
U-Net

- The u-net is a CNN for fast and precise segmentation of images.
 - O. Ronneberger, et al., "**U-Net: Convolutional Networks for Biomedical Image Segmentation**," *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Springer, LNCS, Vol.9351: pp. 234-241, 2015.



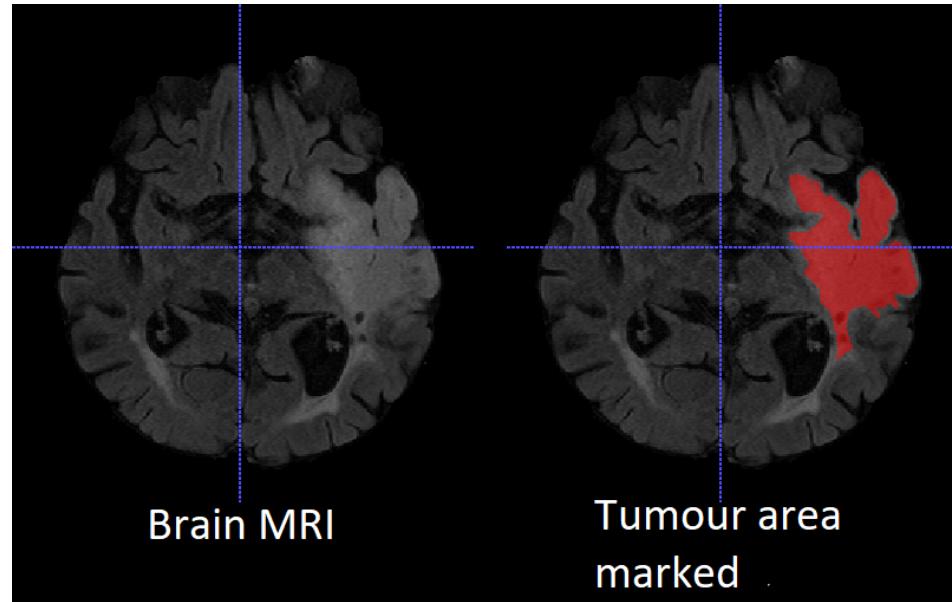
U-Net

- Segmentation of neuronal structure in EM



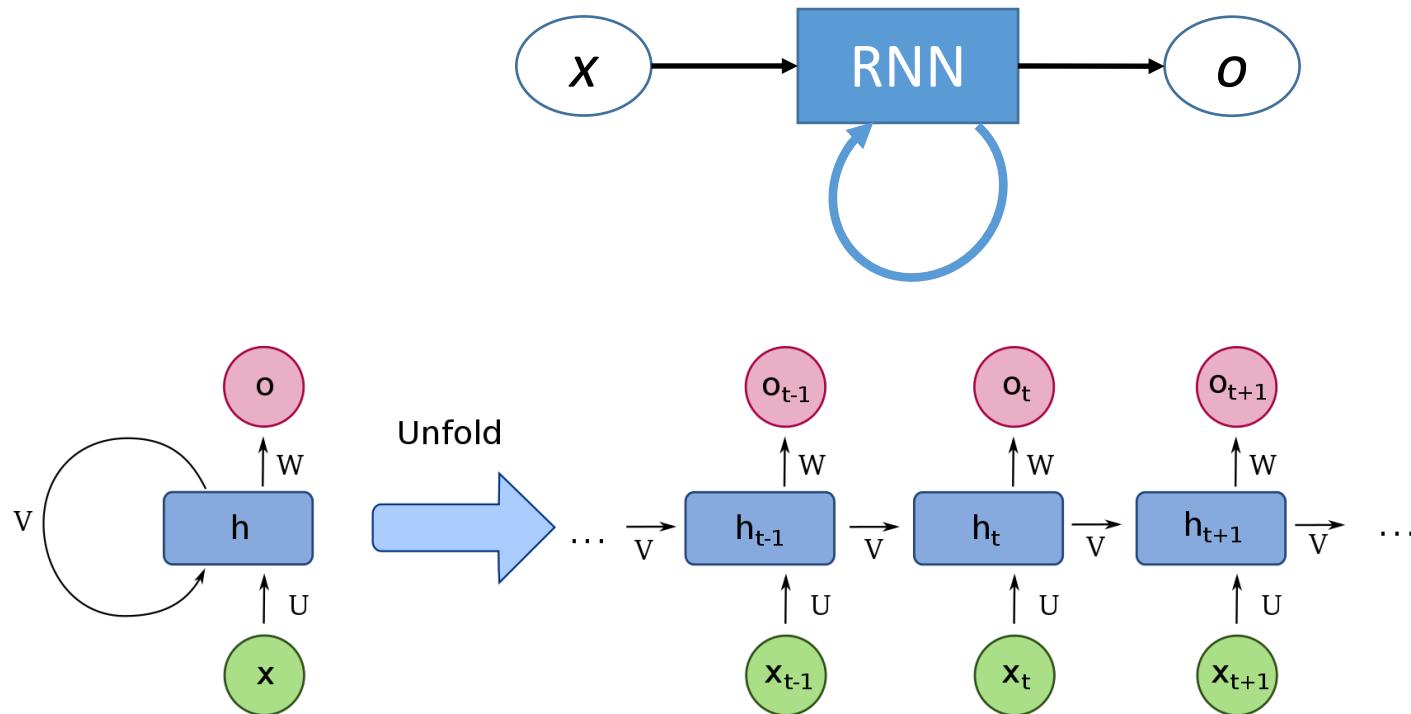
U-Net

- Philipp Schnell, "Automated Brain Tumour Segmentation based on MR Images using U-Net and its application for Mutation Status Prediction"



Recurrent Neural Network, RNN

- RNNs perform the same task for every element of a sequence, with the output being depended on the **previous computations**.
- J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences of the USA*, vol. 79 no. 8 pp. 2554–2558, April 1982.

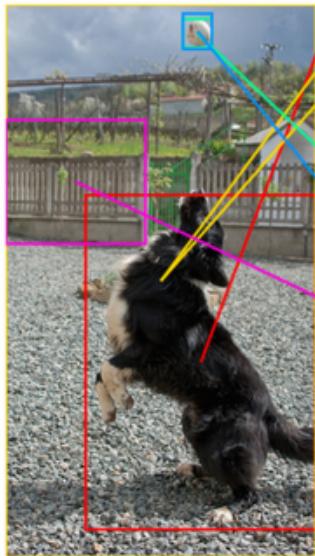


Recurrent Neural Network, RNN

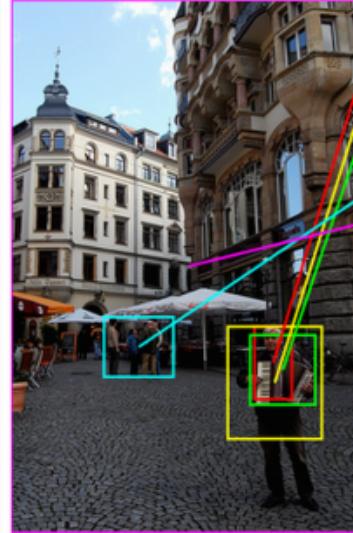
- Training RNNs
 - The backpropagation algorithm
 - The gradient at each output depends not only on the calculations of the current time step, but also the previous time steps.
 - For example, in order to calculate the gradient at $t = 4$ we would need to backpropagate 3 steps and sum up the gradients.

Recurrent Neural Network, RNN

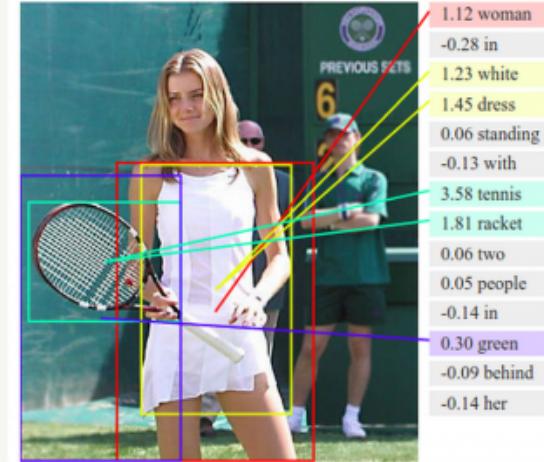
- Applications:
 - Speech Recognition
 - Generating Image Descriptions



1.31 dog
0.31 plays
0.45 catch
-0.02 with
0.25 white
1.62 ball
-0.10 near
-0.07 wooden
0.22 fence



0.26 man
0.31 playing
1.51 accordion
-0.07 among
-0.08 in
0.42 public
0.30 area



1.12 woman
-0.28 in
1.23 white
1.45 dress
0.06 standing
-0.13 with
3.58 tennis
1.81 racket
0.06 two
0.05 people
-0.14 in
0.30 green
-0.09 behind
-0.14 her

Deep Visual-Semantic Alignments for Generating Image Descriptions. Source:
<http://cs.stanford.edu/people/karpathy/deepimagesent/>

Recurrent Neural Network, RNN

- RNN Extensions
 - **Bidirectional RNNs**

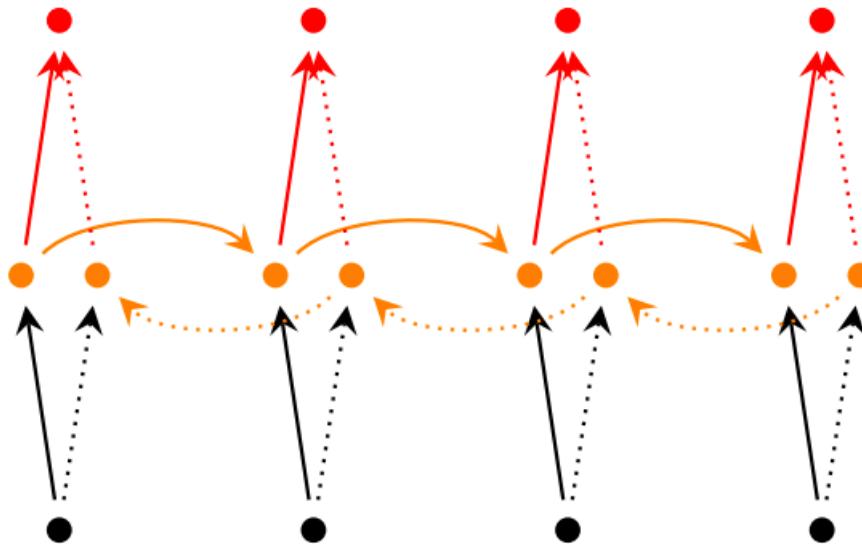


Fig from:

<http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>

Recurrent Neural Network, RNN

- RNN Extensions
 - Deep (Bidirectional) RNNs

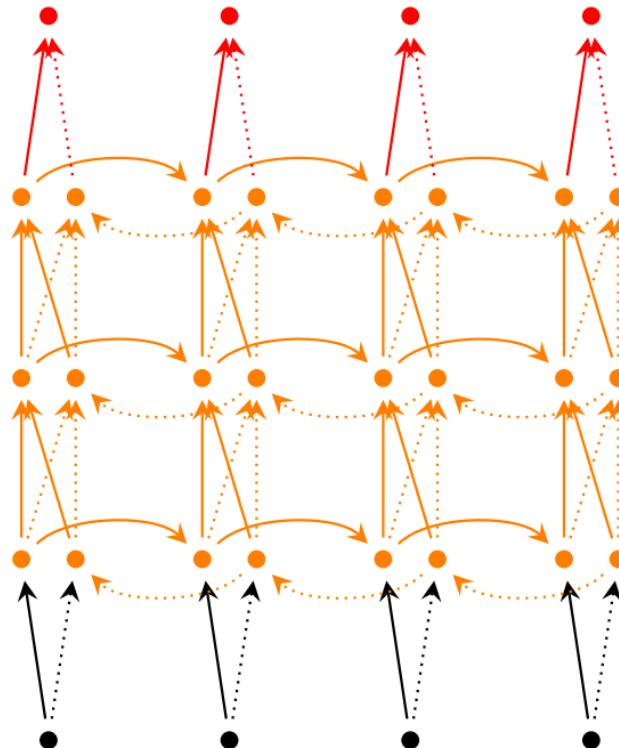


Fig from:

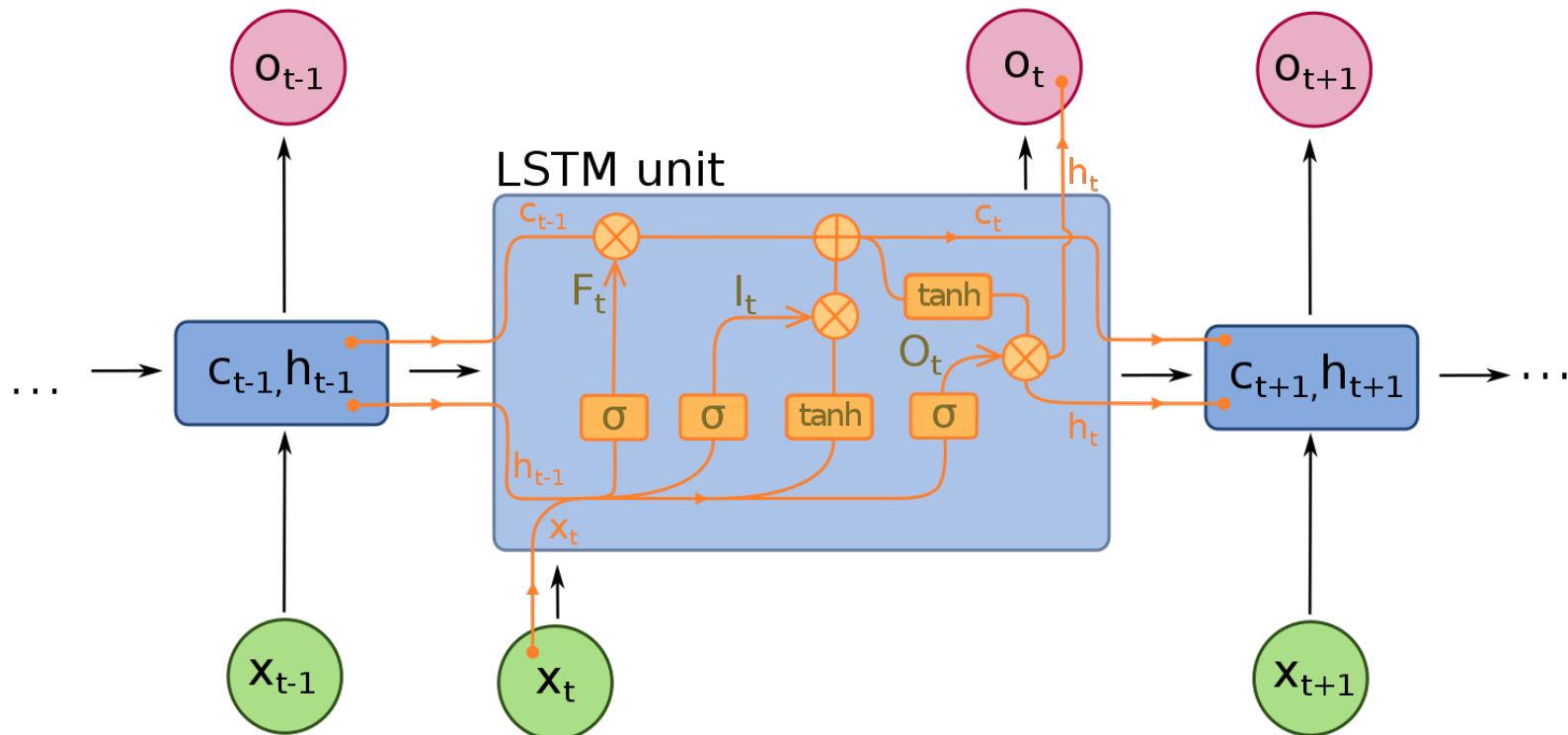
<http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>

Recurrent Neural Network, RNN

- A problem:
 - In text recognition
 - "the clouds are in the *sky*"
 - → "in the *sky*"
 - → "sky"
 - *sky* means sky, there is no other meaning
 - "I grew up in France... I speak fluent *French*"
 - → " I speak fluent *French* "
 - → " *French* "
 - *French* means people? or language?
 - It needs a memory to store some information of previous states.

Recurrent Neural Network, RNN

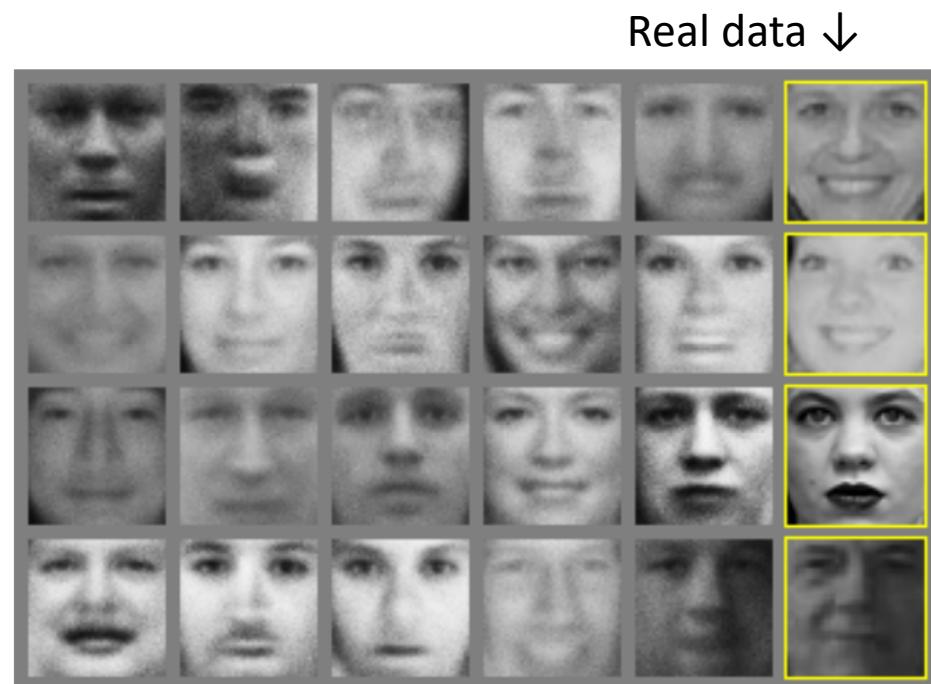
- Long short-term memory, **LSTM**
- LSTM networks
 - C_t : The cell state at time t . It store the necessary input of $t, t-1, t-2, \dots$
 - F_t : Forget weight. It can remove some old data at time t .
 - I_t : Update weight. It can add some new data at time t .



GAN

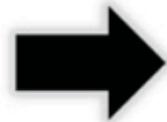
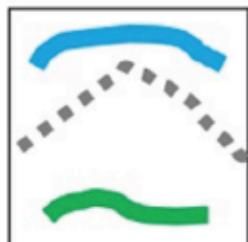
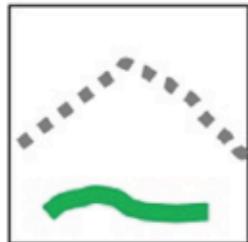
- GAN, Generative Adversarial Network

- Ian J. Goodfellow et. al, "Generative Adversarial Networks," arXiv, Jun. 2014.



GAN

User edits



Generated images



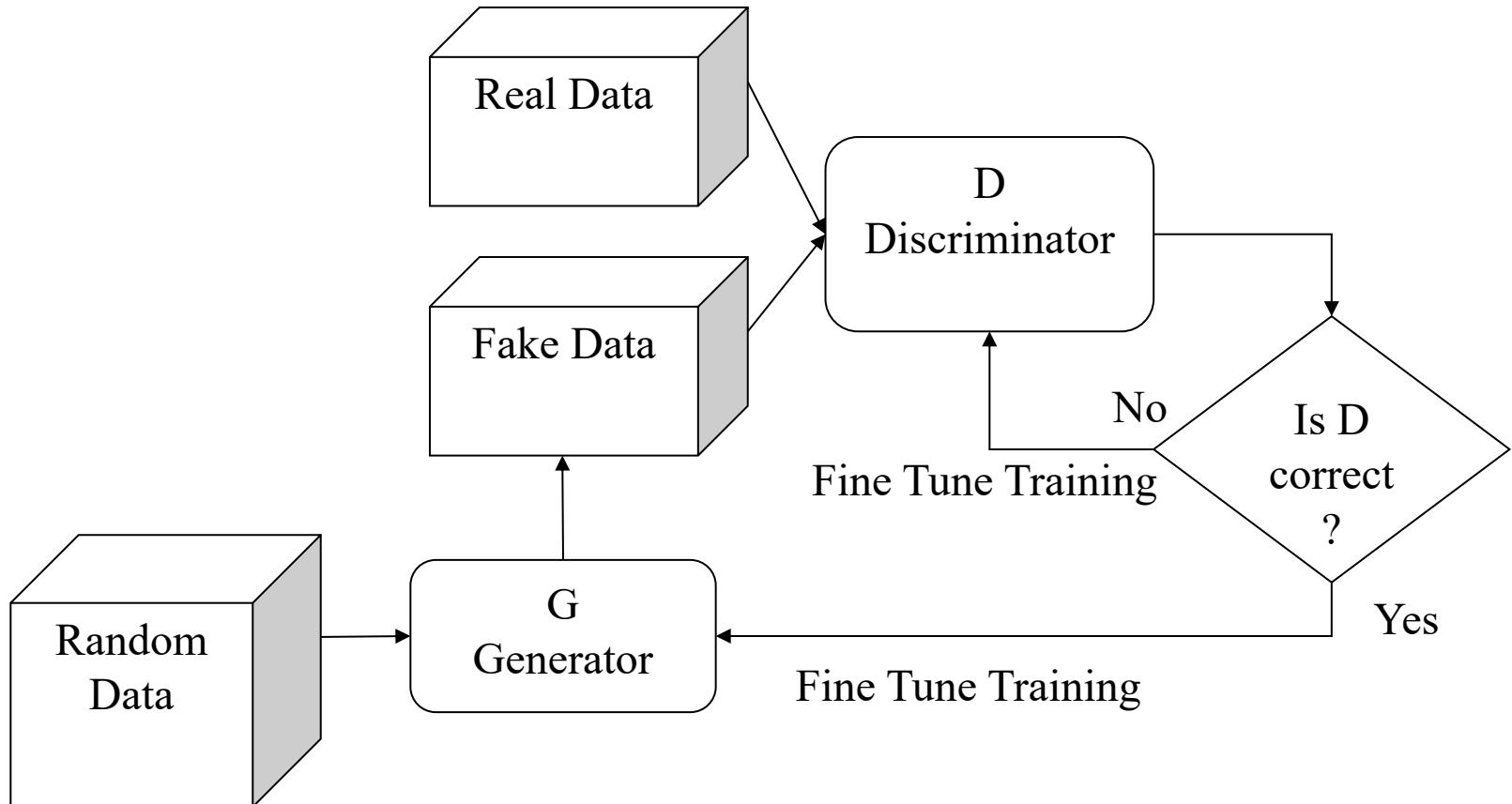
<http://people.eecs.berkeley.edu/~junyanz/projects/gvm/>

GAN

- Generative
 - Learn a generative model
- Adversarial
 - Trained in an adversarial setting

GAN

- Architecture



GAN

- Model

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

Image Synthesis by AI

- C. Li and M. Wand, "Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis," *arXiv*, 2016.



Input A



Input B



Content A + Style B



Content B + Style A



Input style



Input content



Image Synthesis by AI

- A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," *ICLR 2019*.



Image Synthesis by AI

- J.Y. Z, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," ICCV 2017.

Monet \curvearrowright Photos



Monet \rightarrow photo

Zebras \curvearrowright Horses



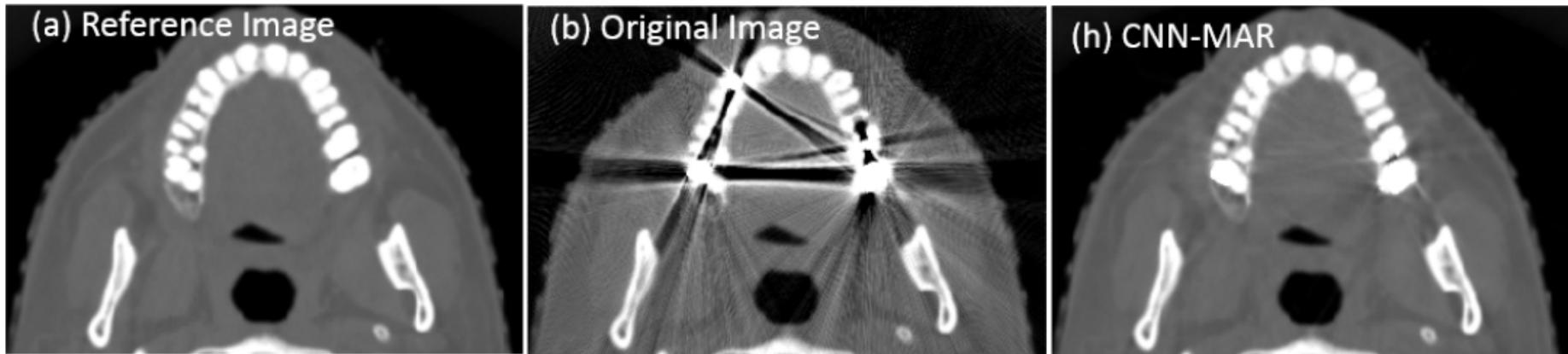
zebra \rightarrow horse

photo \rightarrow Monet

horse \rightarrow zebra

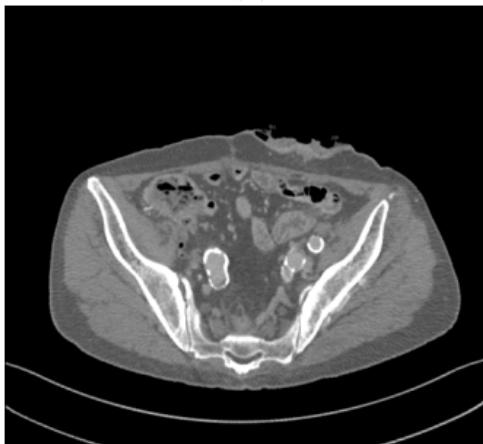
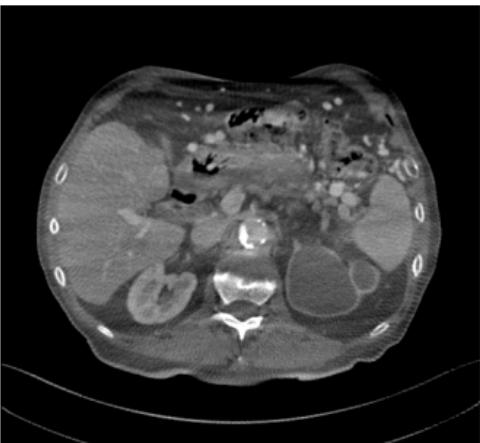
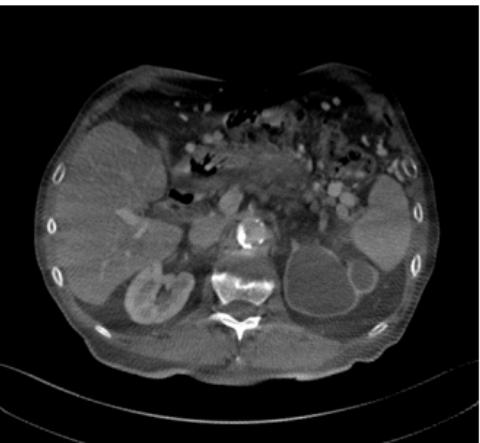
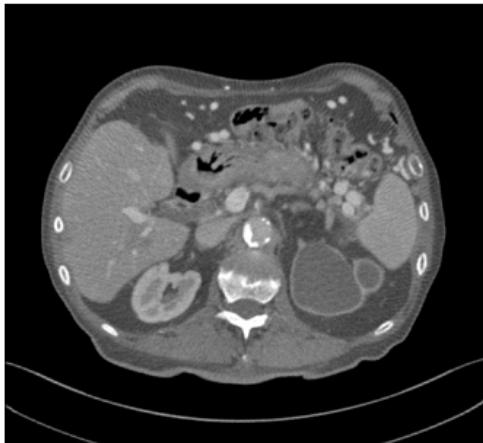
CT and AI

- Y. Zhang and H. Yu, "Convolutional Neural Network Based Metal Artifact Reduction in X-Ray Computed Tomography," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1370-1381, Jun. 2018.



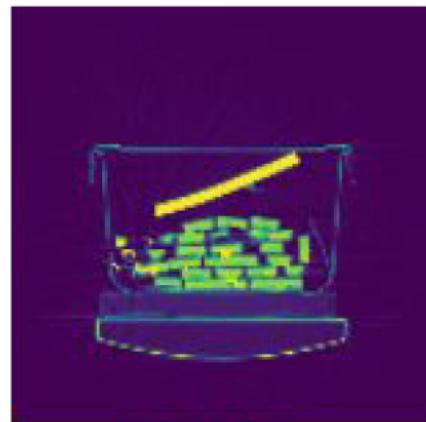
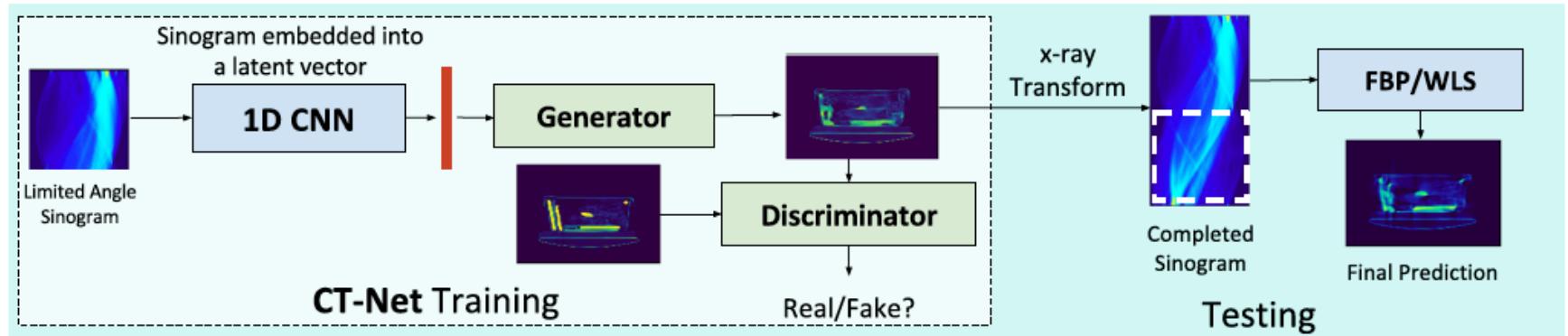
CT and AI

- T. Würfl et al., "Deep Learning Computed Tomography: Learning Projection-Domain Weights From Image Domain in Limited Angle Problems," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1454-1463, Jun. 2018.

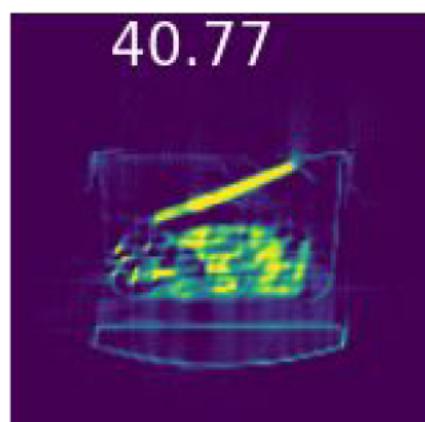


CT and AI

- R. Anirudh, et al., "Lose The Views: Limited Angle CT Reconstruction via Implicit Sinogram Completion," *arXiv*, Jul. 2018.



Ground truth
128*128



CT-Net Result
sonogram: 0°-90°

3D Modeling and AI

- D. Kudinov and D. Hedges, "Reconstructing 3D buildings from aerial LiDAR with AI"



RGB channels



Rasterized Aerial LiDAR



Manually digitized Hip
(purple) and Gable
(orange) segments

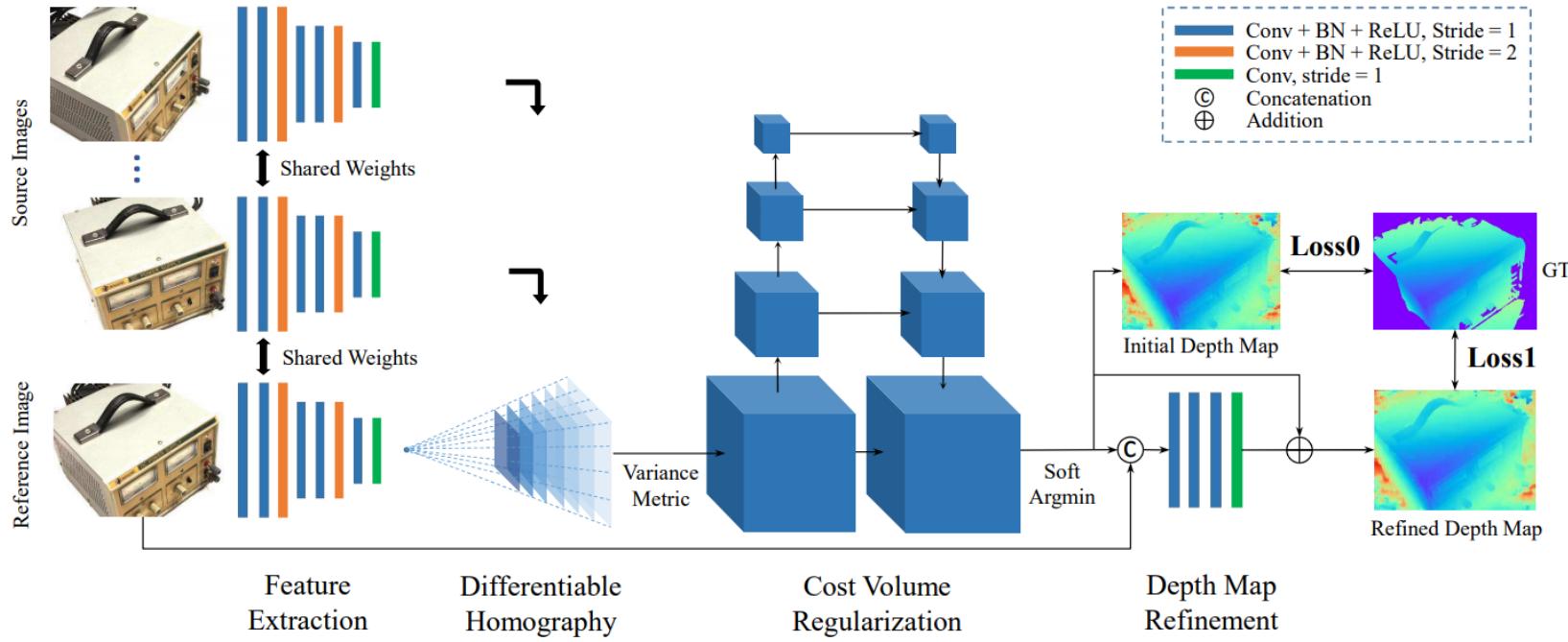


3D reconstruction of
building using manually
digitized segments



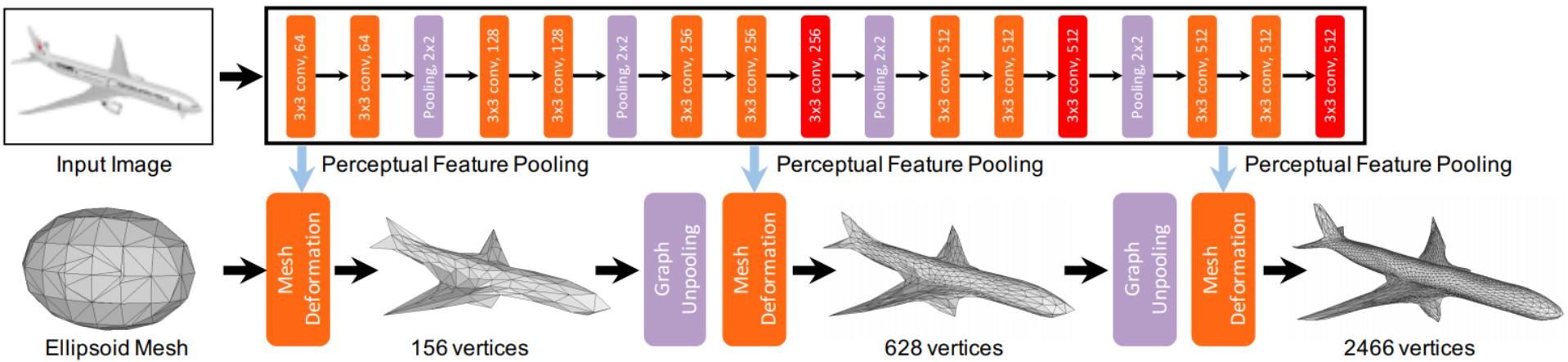
3D Modeling and AI

- Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "MVSNet: Depth Inference for Unstructured Multi-view Stereo," ECCV 2018.



3D Modeling and AI

- N. Wang, et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images," ECCV 2018.



Poincaré Conjecture

- In 1904, a French mathematician, **Jules Henri Poincaré** (1854 – 1912), conjectured a theorem.

Every simply connected, closed 3-manifold is homeomorphic to the 3-sphere.



Henri Poincaré

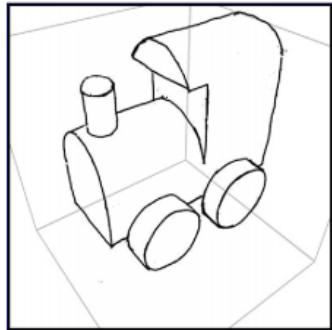


Grigori Perelman

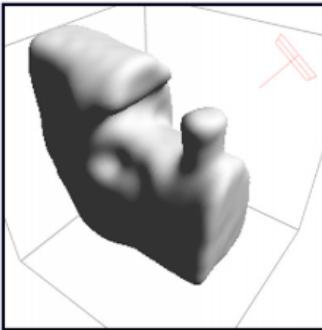
- Nov. 13, 2002, a Russian mathematician, **Grigori Perelman**, proved the Poincaré conjecture.
 - Poincaré Conjecture is one of Millennium Prize Problems
 - Prize: \$1 million
 - But Perelman rejected that prize

3D Modeling and AI

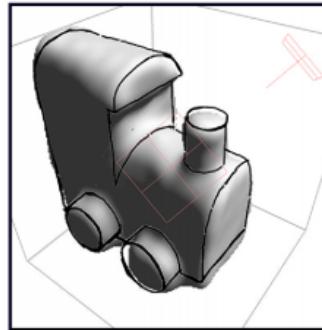
- J. Delanoy, M. Aubry, P. Isola, A. A. Efros, and A. Bousseau, **"3D Sketching using Multi-View Deep Volumetric Prediction,"** *Proc. ACM Comput. Graph. Interact. Tech.* 1, 1, Article 21, July 2018.



a) Initial drawing



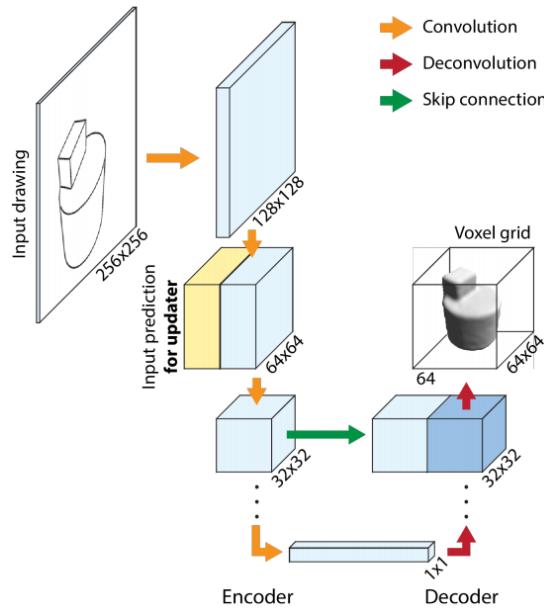
b) 3D prediction
seen from another viewpoint



c) New drawing
and updated prediction



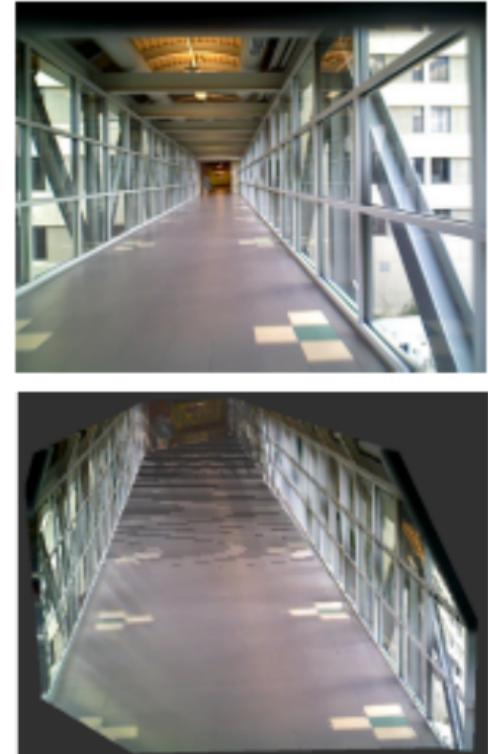
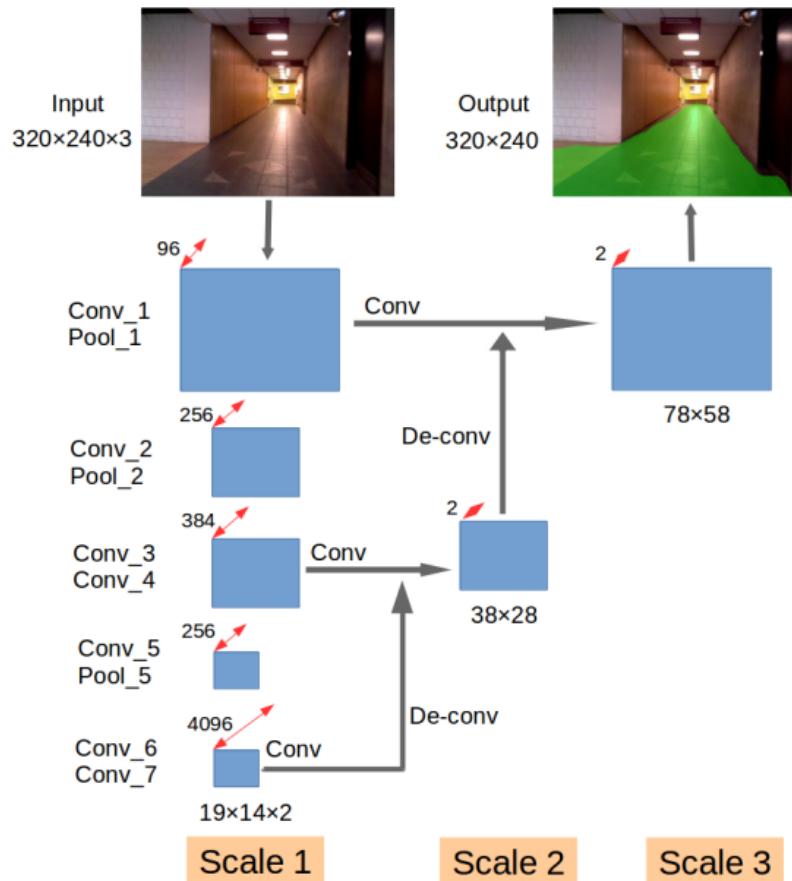
d) 3D printed objects



https://www.youtube.com/watch?time_continue=354&v=DGIYzmlm2pQ

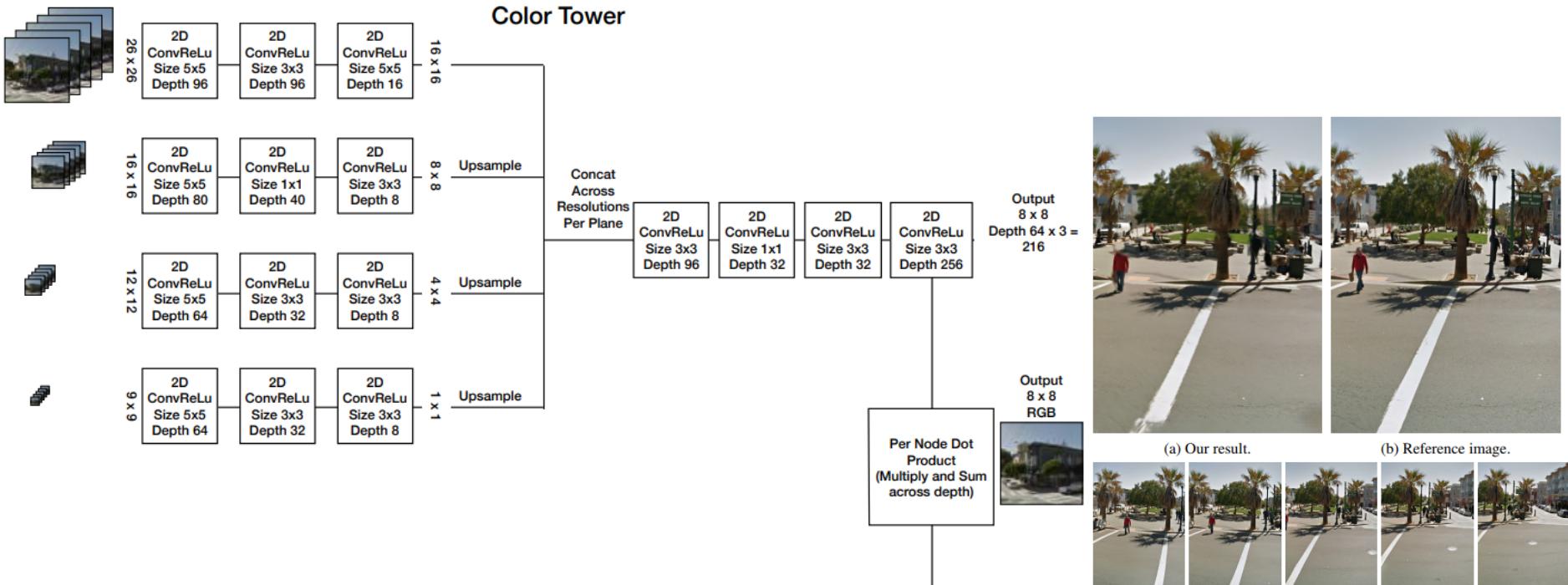
3D Modeling and AI

- S. Yang, D. Maturana and S. Scherer, "Real-time 3D scene layout from a single image using Convolutional Neural Networks," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, 2016, pp. 2183-2189.
- <https://www.youtube.com/watch?v=2CvFHy5jk1c>



3D Modeling and AI

- J. Flynn, I. Neulander, J. Philbin and N. Snavely, "Deep Stereo: Learning to Predict New Views from the World's Imagery," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 5515-5524.
- <https://www.youtube.com/watch?v=cizgVZ8rjKA>



Same Network structure As above,
but different learned parameters

