

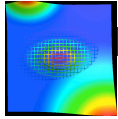
# Data Mining algorithms

2017-2018 spring

03.28.2018

1.

NN and BN



# Neural networks, briefly deeply

Hypothesis: deep, hierarchical models can be exponentially more efficient than a shallow one [Bengio et al. 2009, Le Roux and Bengio, 2010, Delalleau and Bengio, 2011 etc. ]

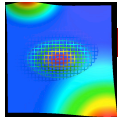
[Delalleau and Bengio, 2011]: deep sum-product network may require exponentially less units to represent the same function compared to a shallow sum-product network.

What is the depth of a Neural Network?

In case of feed forward networks, the number of multiple nonlinear layers between the input and the output layer.

We will see, in case of recurrent NN this definition does not apply.

So Q1: What is NN?

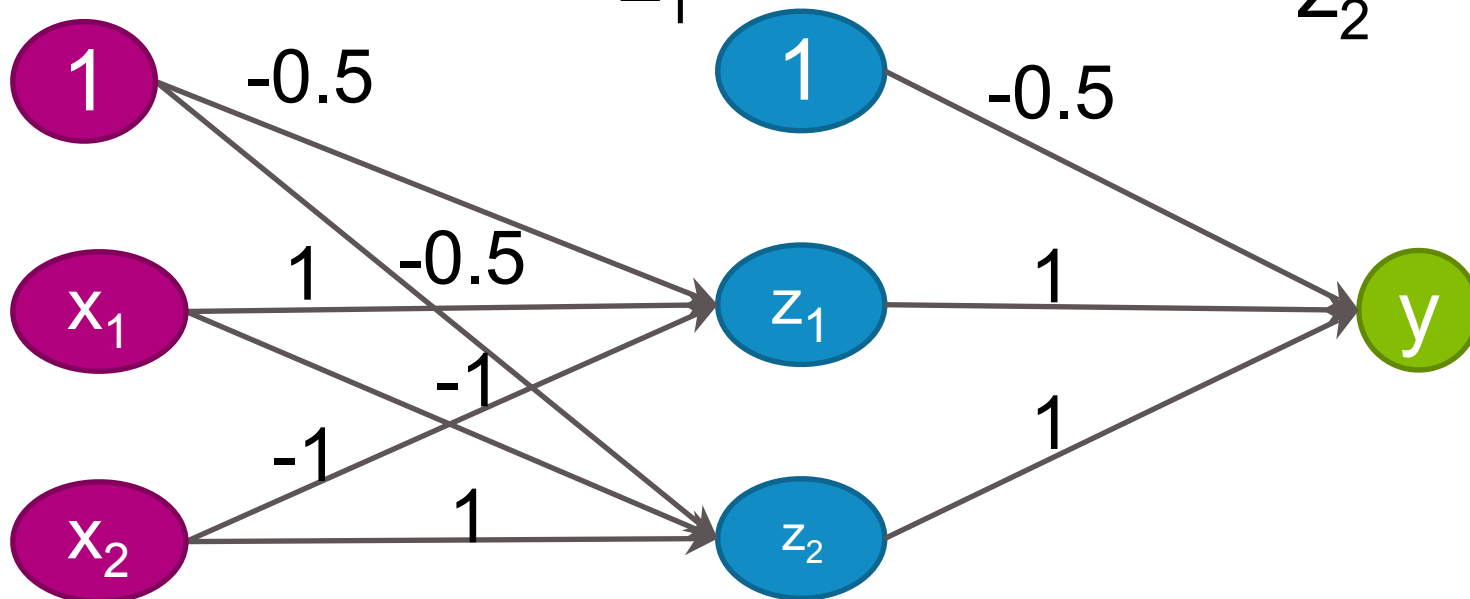


# Neural Networks

Key ingredients:

- Wiring: units and connections

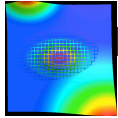
$$\text{XOR} = \underbrace{x_1 \text{ AND NOT } x_2}_{z_1} \text{ OR } \underbrace{\text{NOT } x_1 \text{ AND } x_2}_{z_2}$$

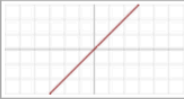

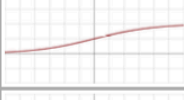
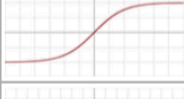
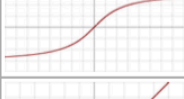
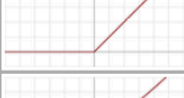
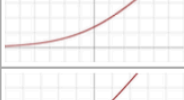
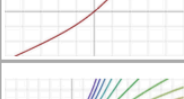
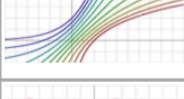
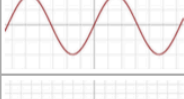
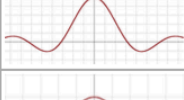



29/03/2017

Fig.: Danny Bickson

# Activation functions



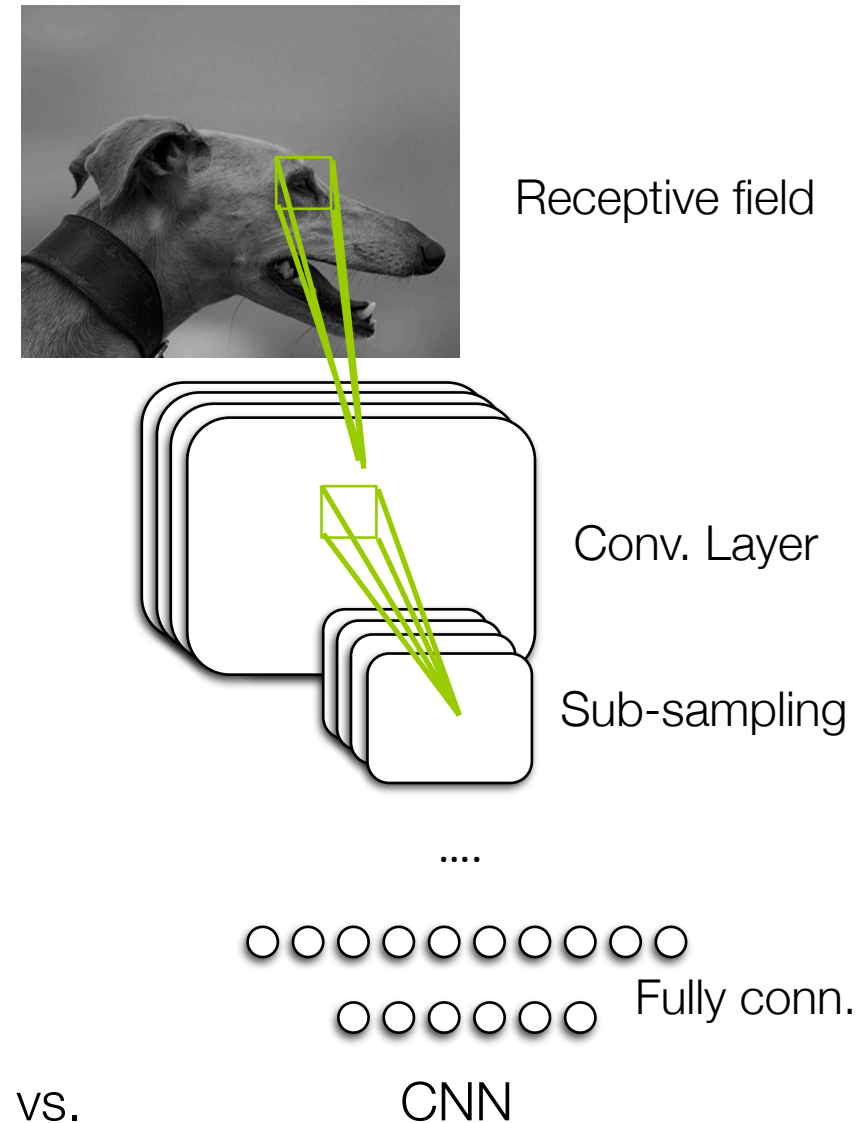
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$
Bent identity		$f(x) = \frac{\sqrt{x^2 + 1} - 1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2 + 1}} + 1$
SoftExponential		$f(\alpha, x) = \begin{cases} -\frac{\log_e(1 - \alpha(x + \alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1 - \alpha(x + \alpha)} & \text{for } \alpha < 0 \\ 1 & \text{for } \alpha = 0 \\ e^{\alpha x} & \text{for } \alpha > 0 \end{cases}$
Sinusoid		$f(x) = \sin(x)$	$f'(x) = \cos(x)$
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{for } x \neq 0 \end{cases}$
Gaussian		$f(x) = e^{-x^2}$	$f'(x) = -2xe^{-x^2}$

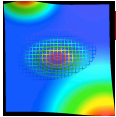
Output of a unit

- linear/  
non-linear
- bounded/  
non-bounded
- usually monotonic,  
but not all

Why so rigid?

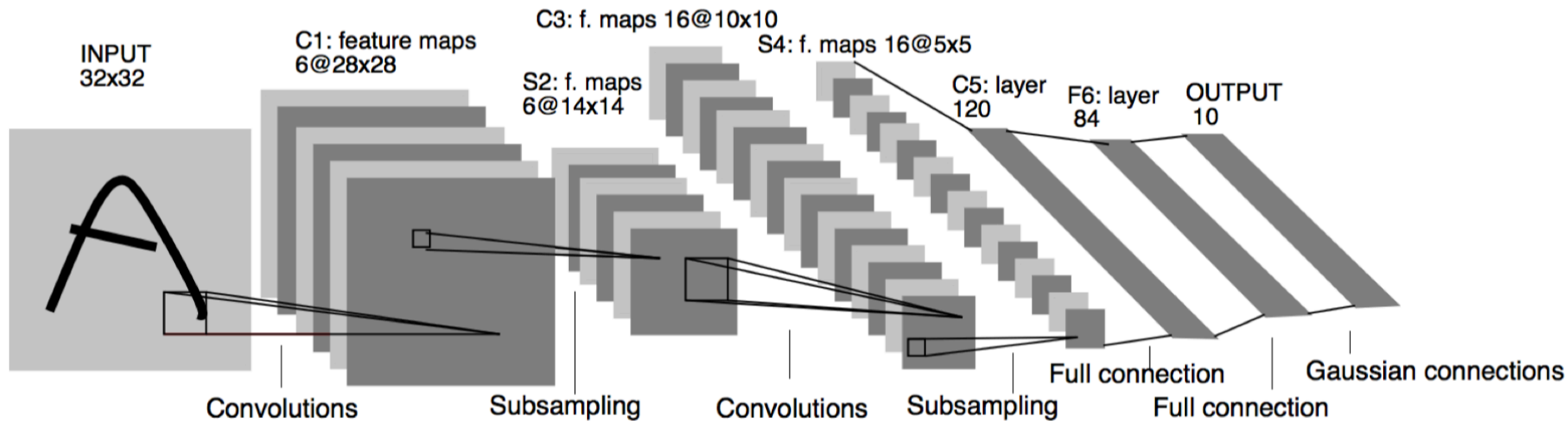
Fig.: wikipedia





# LeNet-5

LeNet-5 for handwriting recognition in [LeCun et al. 1998]

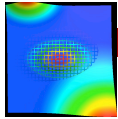


Key advantages:

- Fixed feature extraction vs. learning the kernel functions
- Spatial structure through sampling
- “Easier to train” due much lesser connection than fully connected

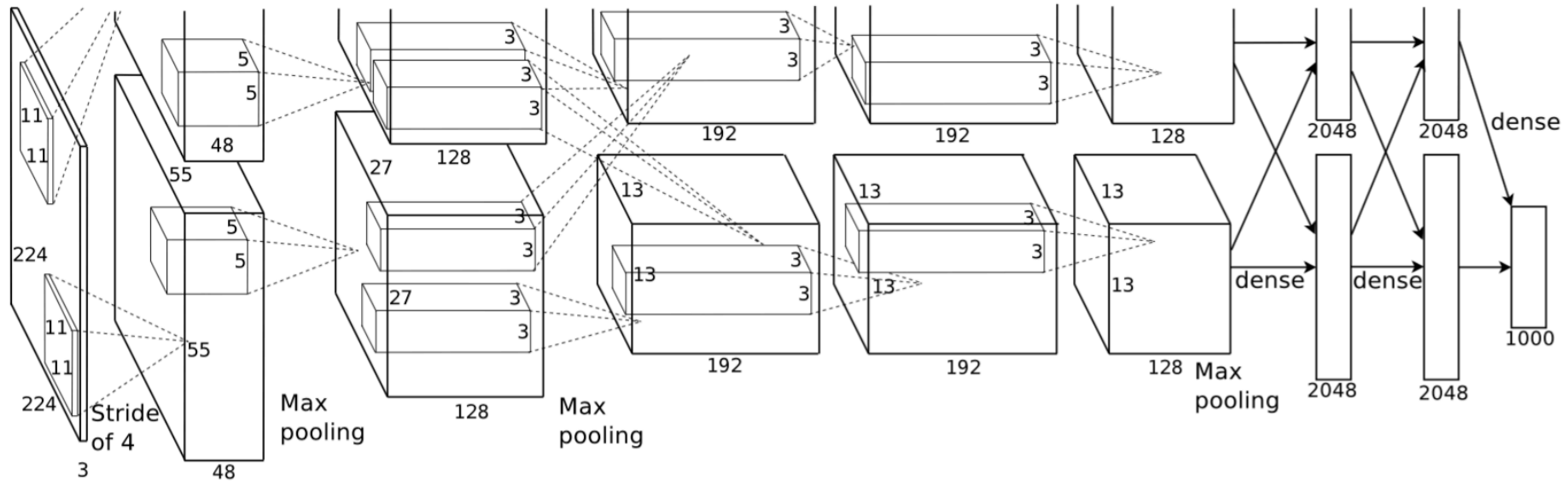
Training: back propagation

By definition it is a feed forward deep neural network.



# Image classification with CNN

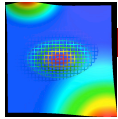
[Krizhevsky et al. 2012]



Advantages over LeNet:

- **Local response normalization** (normalize over the kernel maps at the same position) over ReLU (-1.2%..1.4% in error rate)
- Overlapping pooling (-0.3%..-0.4% in error rate)
- traditional image tricks: augmentation as horizontal flipping, subsampling, PCA over the RGB and noise (-1% in error rate)
- **Dropout** (we will discuss it later)





# Image classification with CNN

[Krizhevsky et al. 2012]

ImageNet: 150k test set and 1.2 million training images with 1000 labels.

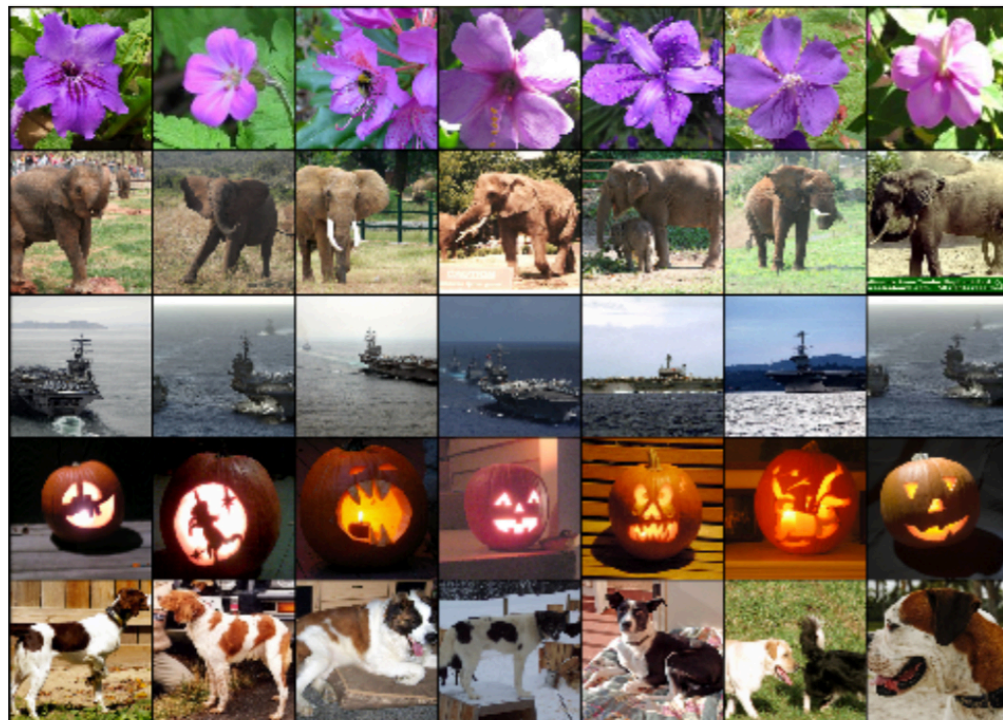
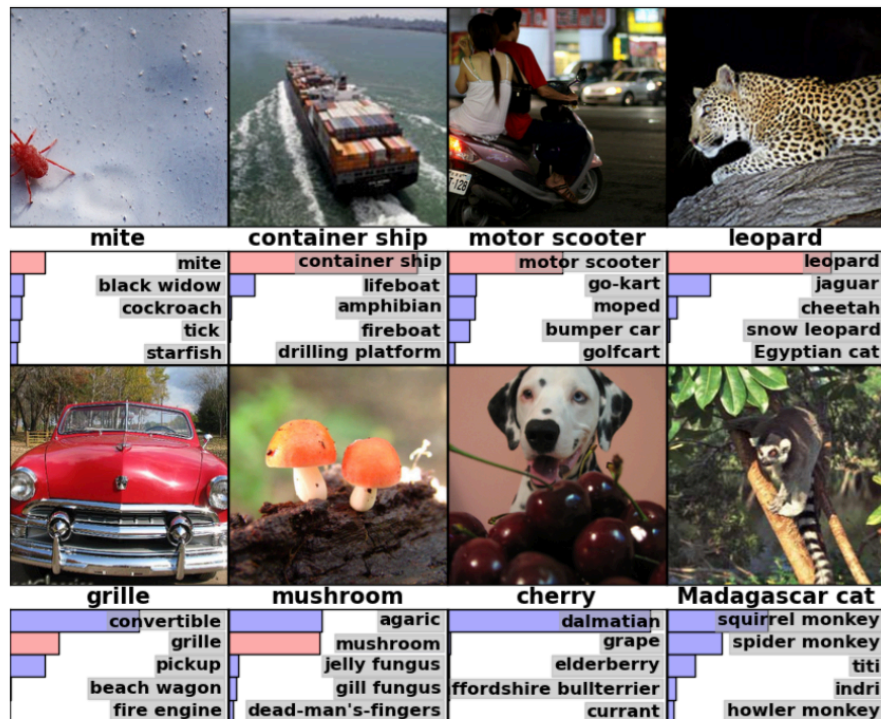
Evaluation: top-1 and top-5 error rate

\* - additional data

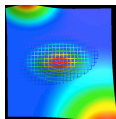
4096 dim. representation per image

5-6 days with 2 Nvidia GTX 580 3GB

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
<i>SIFT + FVs [7]</i>	—	—	26.2%
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	<b>16.4%</b>
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	<b>15.3%</b>



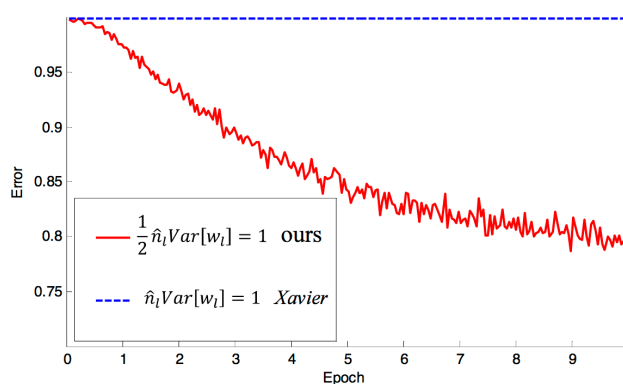




# Recent results

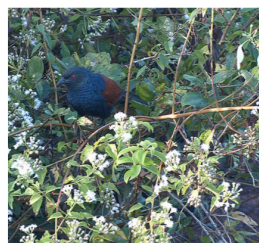
[He et al. 2015]: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

**Parametric ReLU** + zero mean Gaussian init + extreme (at the time...) deep network:  
A: 19 layers, B: 22 layers, C: 22 layers with more filters



Training of model C: 8xK40 Nvidia GPU 3..4 weeks (!)

model	top-1	top-5
MSRA [11]	29.68	10.95
VGG-16 [25]	28.07 <sup>†</sup>	9.33 <sup>†</sup>
GoogLeNet [29]	-	9.15
A, ReLU	26.48	8.59
A, PReLU	25.59	8.23
B, PReLU	25.53	8.13
C, PReLU	<b>24.27</b>	<b>7.38</b>



GT: coucal  
1: coucal  
2: indigo bunting  
3: lorikeet  
4: walking stick  
5: custard apple



GT: komondor  
1: komondor  
2: patio  
3: llama  
4: mobile home  
5: Old English sheepdog



GT: yellow lady's slipper  
1: yellow lady's slipper  
2: slug  
3: hen-of-the-woods  
4: stinkhorn  
5: coral fungus



GT: torch  
1: stage  
2: spotlight  
3: torch  
4: microphone  
5: feather boa

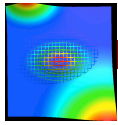


GT: banjo  
1: acoustic guitar  
2: shoji  
3: bow tie  
4: cowboy hat  
5: banjo

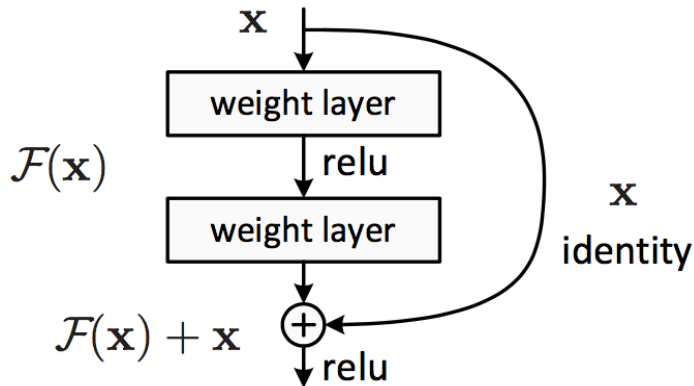


GT: go-kart  
1: go-kart  
2: crash helmet  
3: racer  
4: sports car  
5: motor scooter

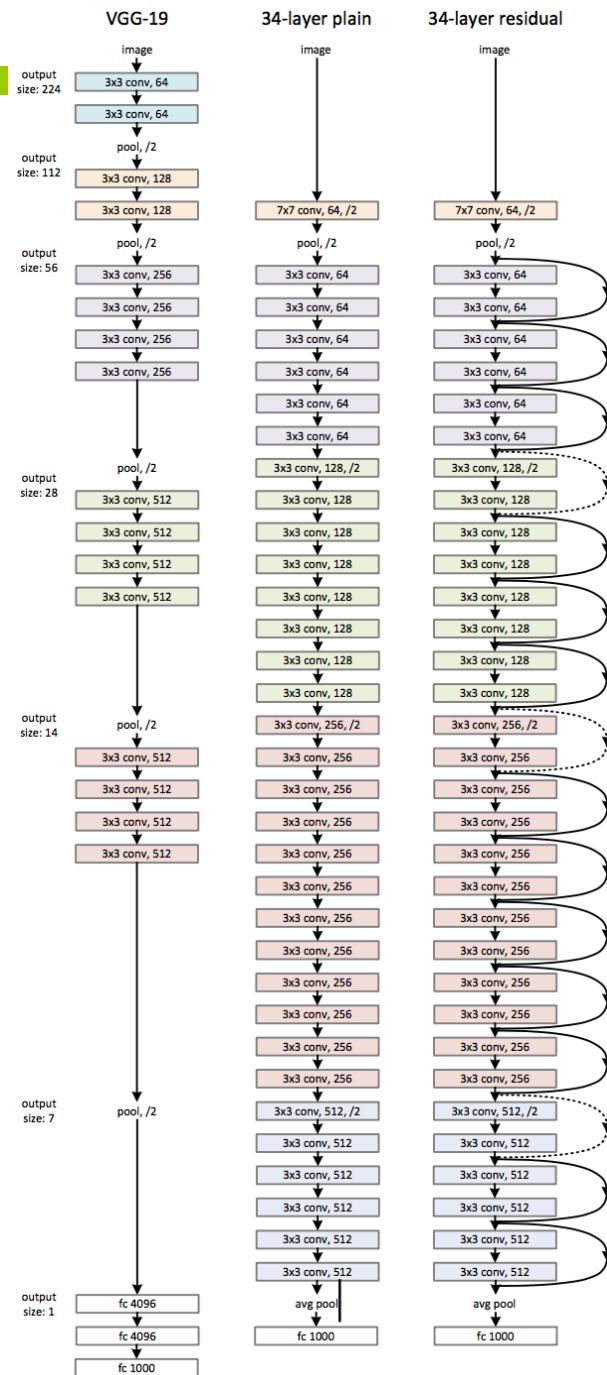
# Recent results

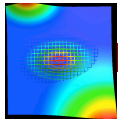


[He et al. 2015]: ResNet: “Is learning better networks as easy as stacking more layers?”



model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PRelu-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	<b>21.43</b>	<b>5.71</b>





# Several implementations

	Restrictions	Wrapper	Architectures	Notes
Theano	Both feed forward and recurrent nets	Python	Multi core/CUDA	Multiple optimization
Chainer	Both feed forward and recurrent nets	Python	Multi core/CUDA	Multiple optimization
GraphLab	Feed Forward: CNN, DBN	Python/C++	Multi core/CUDA/distributed	Compact, Hadoop
TensorFlow	Both feed forward and recurrent nets	Python/C++	Multi core/CUDA/distributed	Graphical interface and multiple optimization
Caffe	Feed Forward: CNN, DBN	Python/Matlab	Multi core/CUDA	
Torch	Both feed forward and recurrent nets	Lua	Multi core/CUDA	Developed for vision

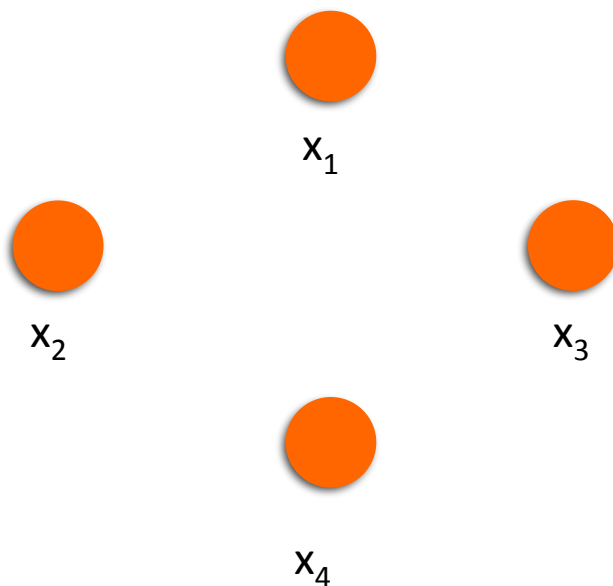
# Bayes Networks

Ok, step back a little and ... 😊

Probabilistic Graphical models (hmmm, RF?)

Set of random variables:  $X = \{x_1, \dots, x_T\}$

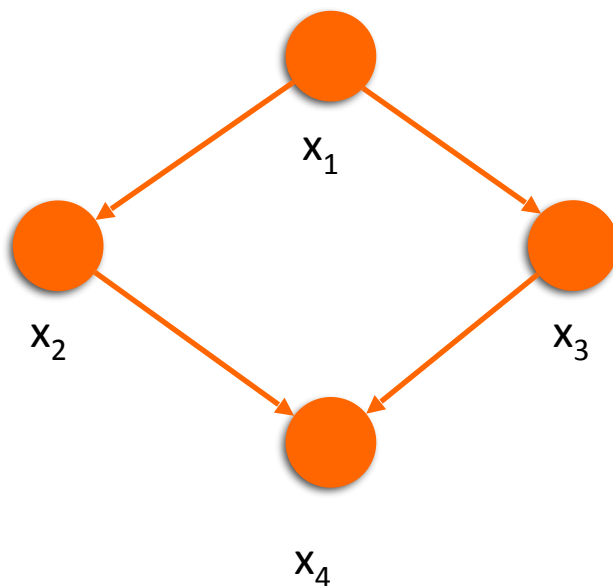
Visualize connections with edges



# Bayes Networks

Probabilistic Graphical models

Visualize connections with edges (with directed edges!!!)  
Conditional dependencies (A “causes” B) vs. MRF?

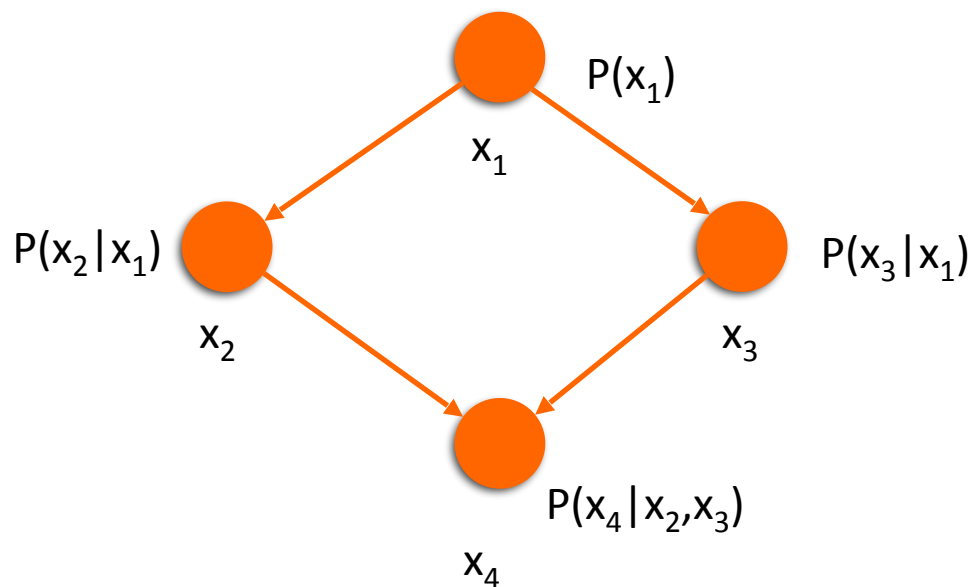


# Bayes Networks

Probabilistic Graphical models

Set of random variables:  $X = \{x_1, \dots, x_T\}$

Visualize connections with edges (with directed edges!!!)



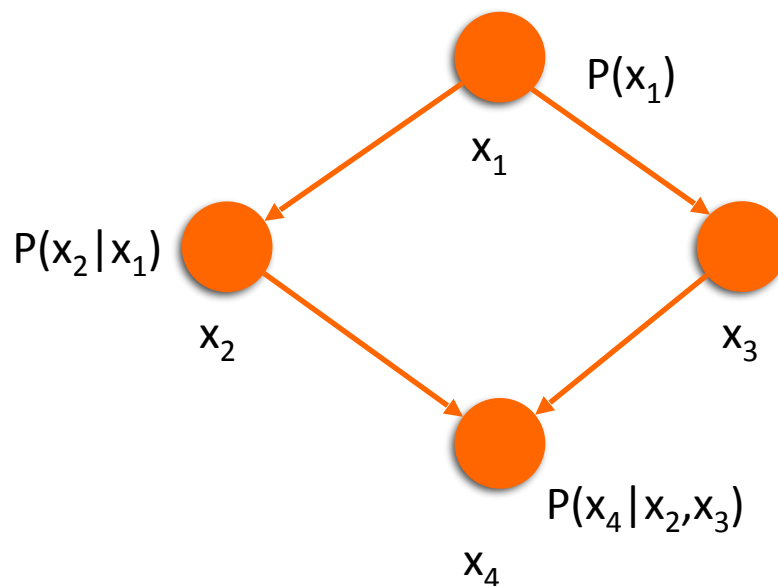


# Bayes Networks

Probabilistic Graphical models

Set of random variables:  $X = \{x_1, \dots, x_T\}$

Visualize connections with edges (with directed edges!!!)



$$P(x_1)P(x_2|x_1)P(x_3|x_2, x_1) P(x_4|x_3, x_2, x_1) \\ = P(x_1)P(x_2|x_1)P(x_3|x_1) P(x_4|x_3, x_2)$$

# Bayes Networks

Learning:

I) We know the structure (the dependencies)

Parameter estimation (prior, posterior)  
Analytically or via optimization (EM, GD etc.)

II) We do not know the structure

optimize over the space of the possible trees...  
and estimate parameters, optimization etc.

What kind of BN is a feed—forward network?