
neuro
dev

LESSON 5:
NEURAL NETWORKS

BY ALEXANDER MATHIS
HARVARD UNIVERSITY

ORGANIZED BY BRUNO BOIVIN
AND RAFAEL GONZALEZ CANO

Today's agenda

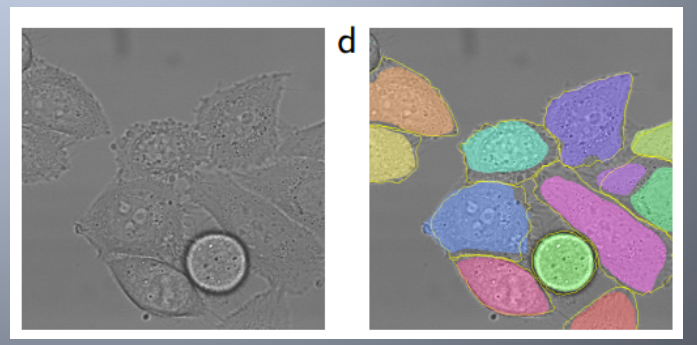
- Intro to Neural Networks: motivation, and basics
- Training and using of a neural network with Keras (TensorFlow) to read handwritten digits (MNIST)

Why neural networks?

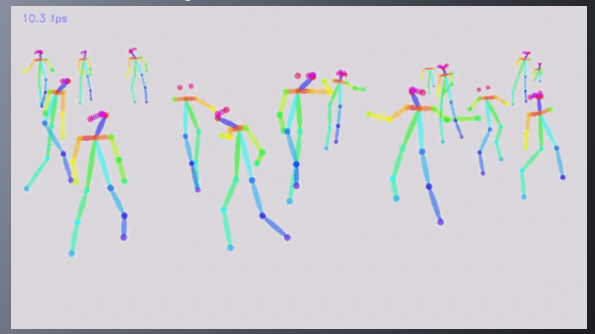
- State-of-the art performance for many challenging tasks….

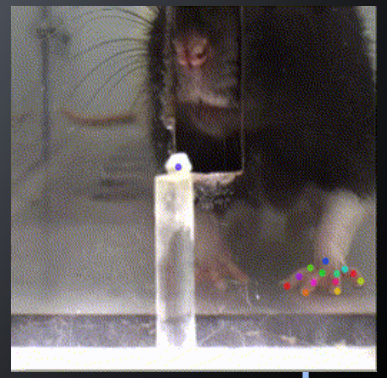
Object recognition,
Alexnet 2012

Cell tracking, U-Net



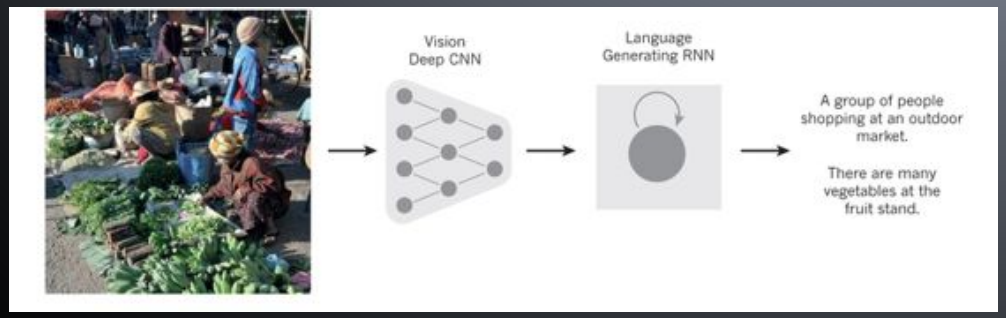
OpenPose





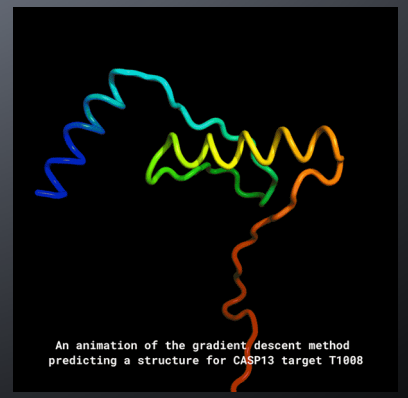
DeepLabCut

Speech recognition,
Image >>> text,



Deep Learning,
LeCun et al.
Nature 2015

Protein folding…
https://deepmind.com/blog/alphafold/

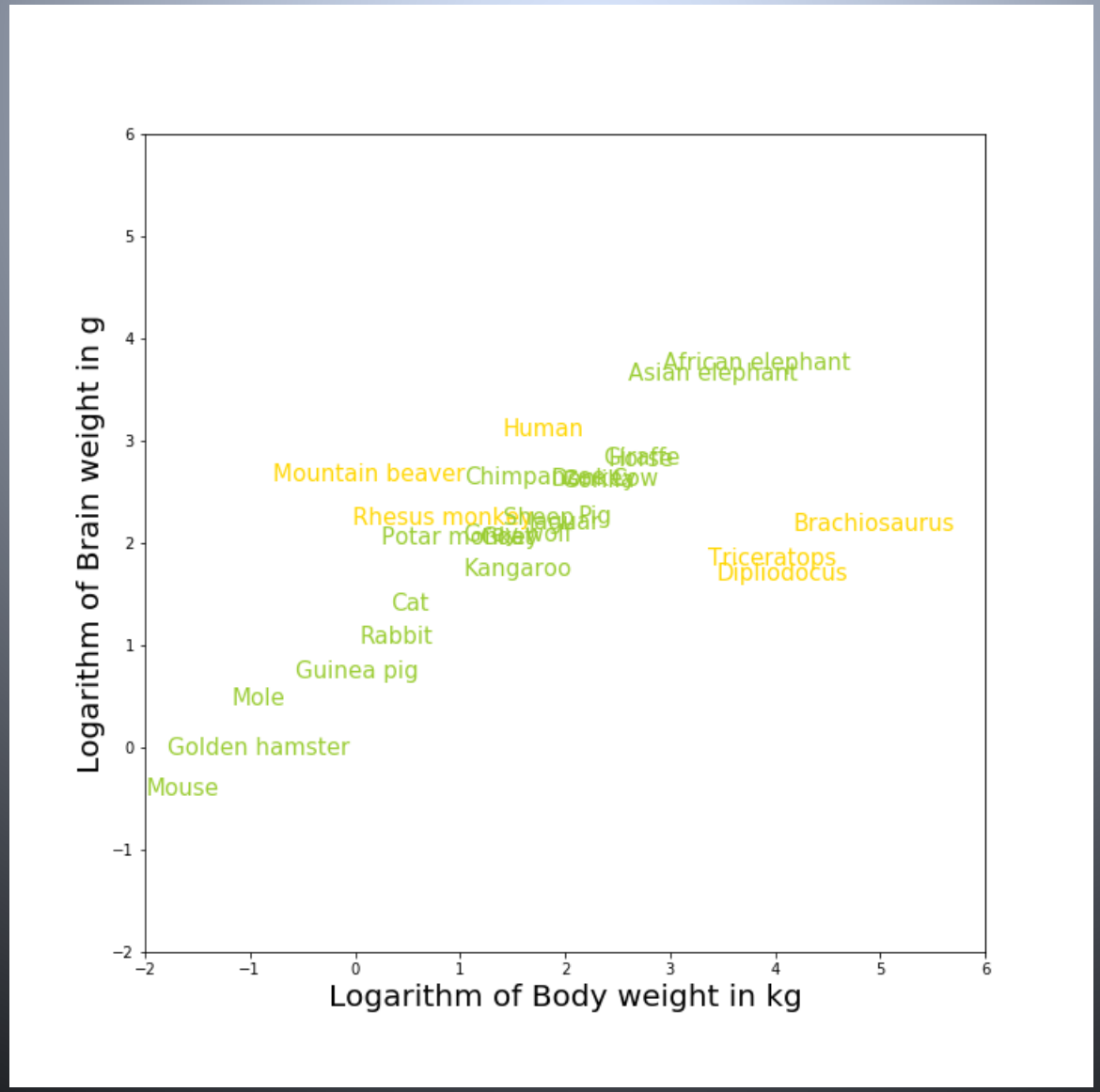


What is learning?

Learning is the process of acquiring new, or
modifying existing, knowledge, behaviors, skills, values, or preferences.
(wikipedia)

For today… learning is curve fitting!

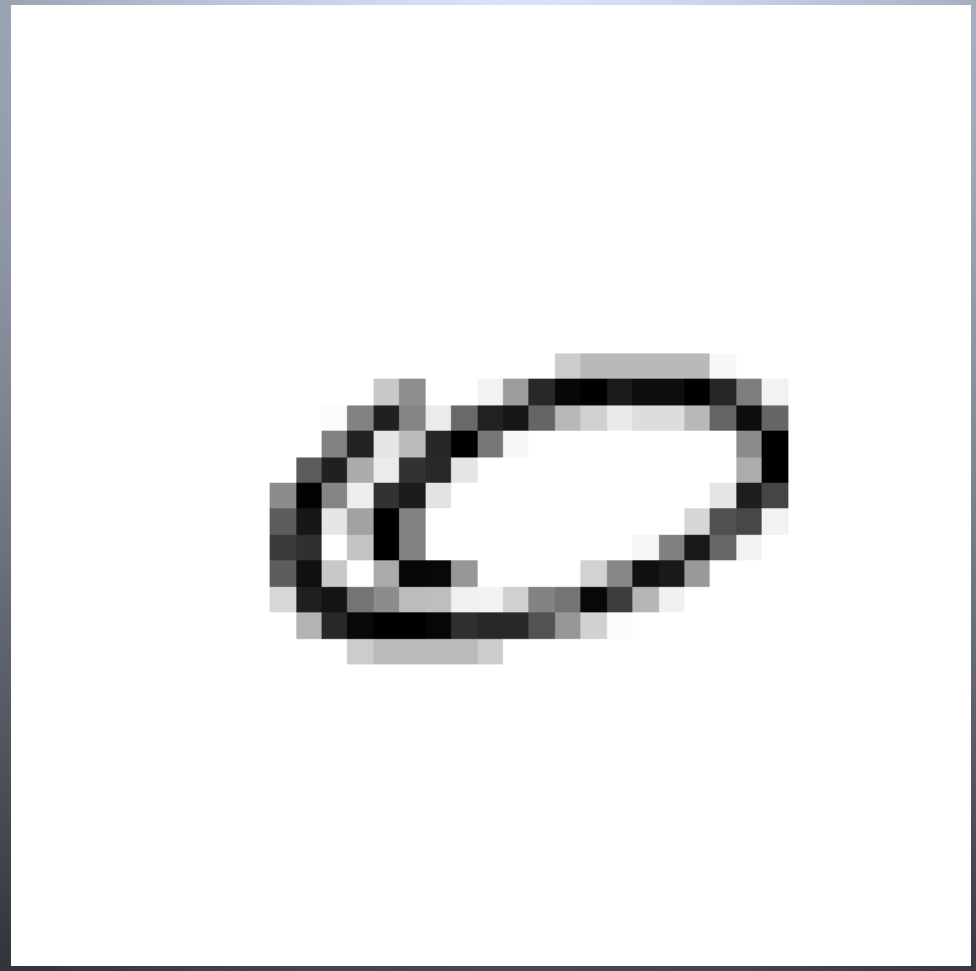
Example 1 – scientific data



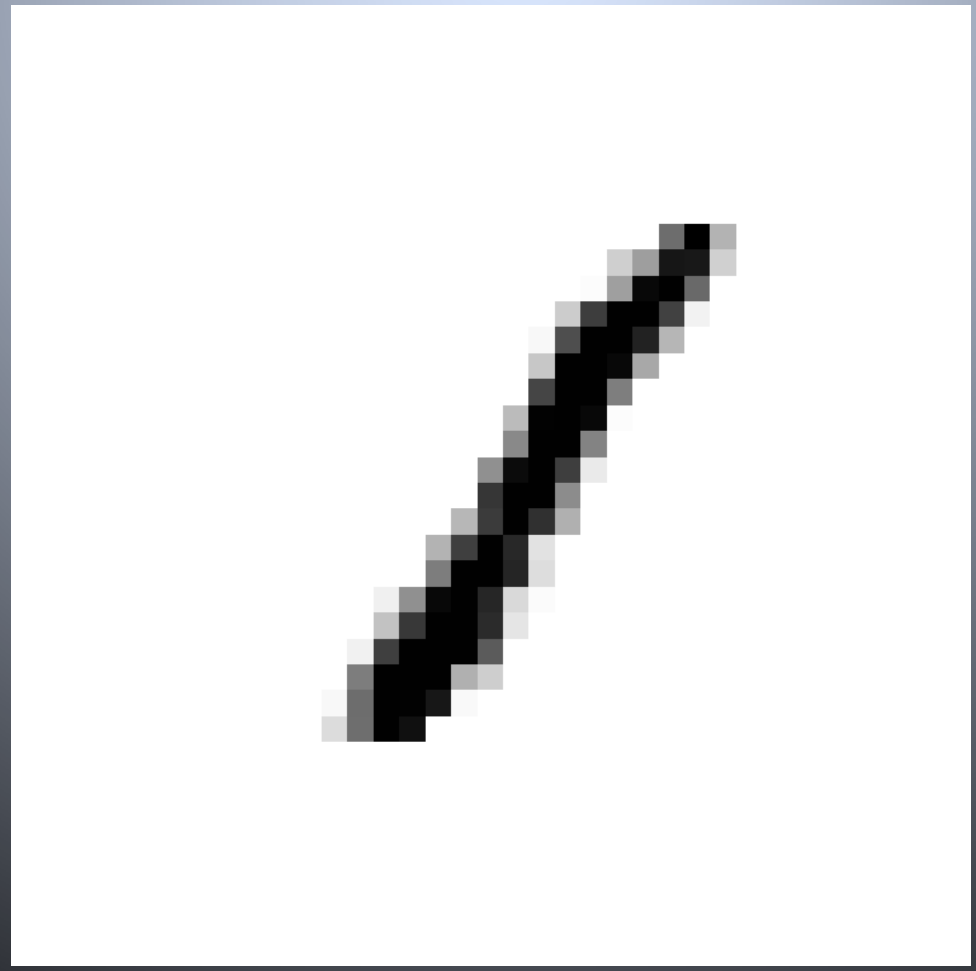
Average brain and body weights for 27 species of land animals.
Source: http://mste.illinois.edu/malcz/DATA/BIOLOGY/Animals.html

Example 1 – scientific data

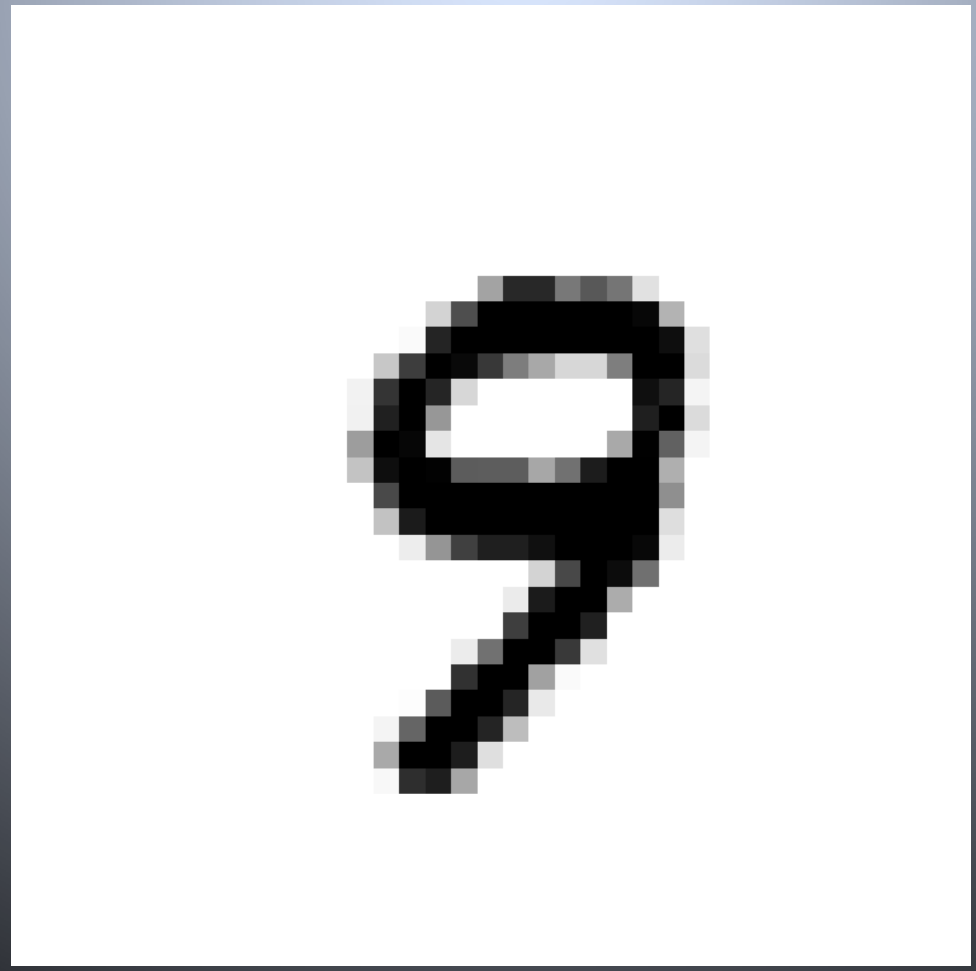
Example 2 – perception



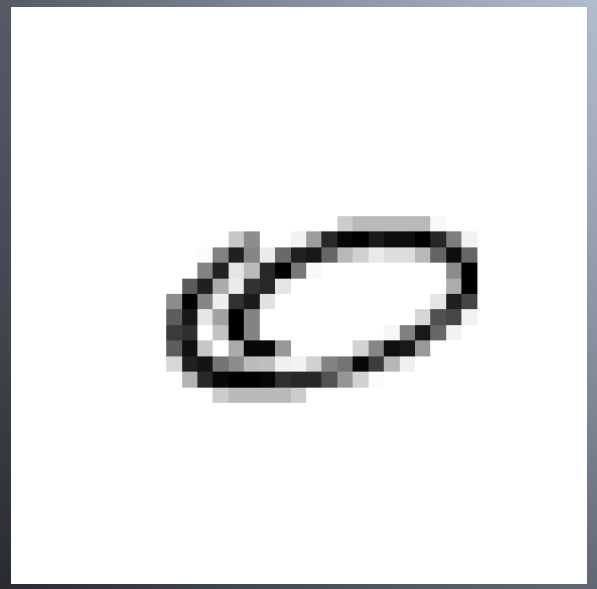
Example 2 – perception

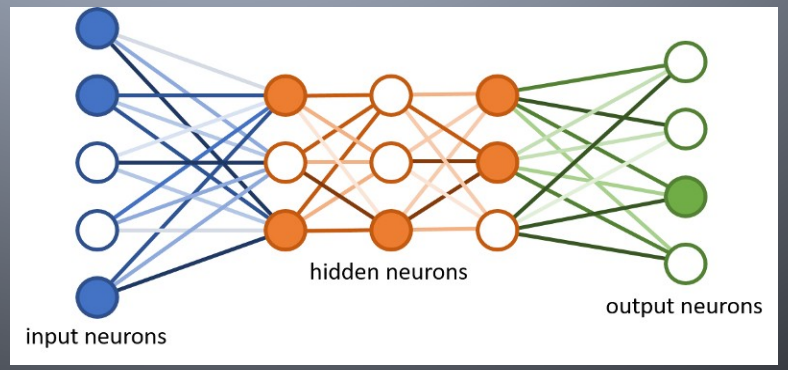


Example 2 – perception



Example 2 – perception





0?
1?
2?
..

The MNIST database of handwritten digits – classical problem for reading handwritten digits
http://yann.lecun.com/exdb/mnist/

Further resources….

The following slides are from
Prof. Russ Salakhutdinov (Carnegie Mellon)

His full lecture slides can be found on the MLSS website!

- Machine Learning Summer School http://mlss.tuebingen.mpg.de/2017/index.html
- Deep Learning Summer School Materials
  https://sites.google.com/site/deeplearningsummerschool2016/

ARTIFICIAL NEURON

- Neuron pre-activation (or input activation):

$$a(\mathbf{x}) = b + \sum_i w_i x_i = b + \mathbf{w}^\top \mathbf{x}$$

- Neuron output activation:

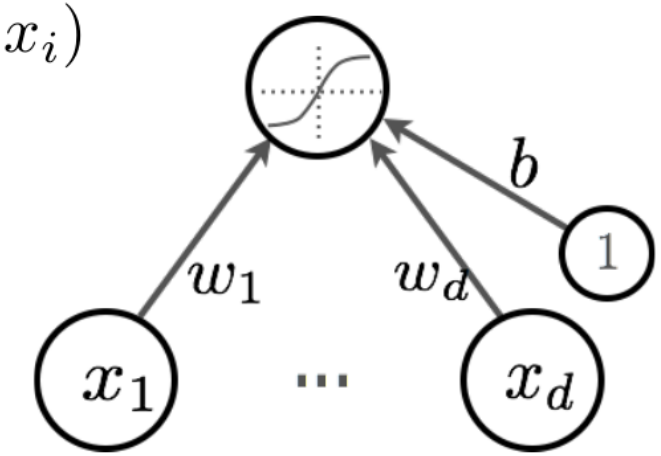
$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$

where

$\mathbf{w}$ are the weights (parameters)

b is the bias term

$g(\cdot)$ is called the activation function



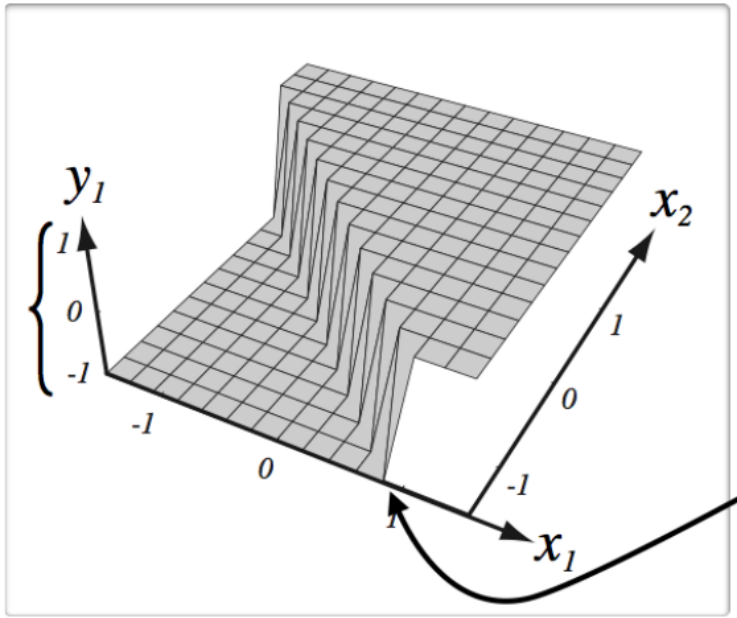
From Prof. Salakhutdinov

ARTIFICIAL NEURON

- Output activation of the neuron:

$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$

Range is determined by $g(\cdot)$



(from Pascal Vincent's slides)

Bias only changes the position of the riff

From Prof. Salakhutdinov

SINGLE HIDDEN LAYER NET

- Hidden layer pre-activation:

$$\mathbf{a}(\mathbf{x}) = \mathbf{b}^{(1)} + \mathbf{W}^{(1)}\mathbf{x}$$

$$\left(a(\mathbf{x})_i = b_i^{(1)} + \sum_j W_{i,j}^{(1)} x_j\right)$$

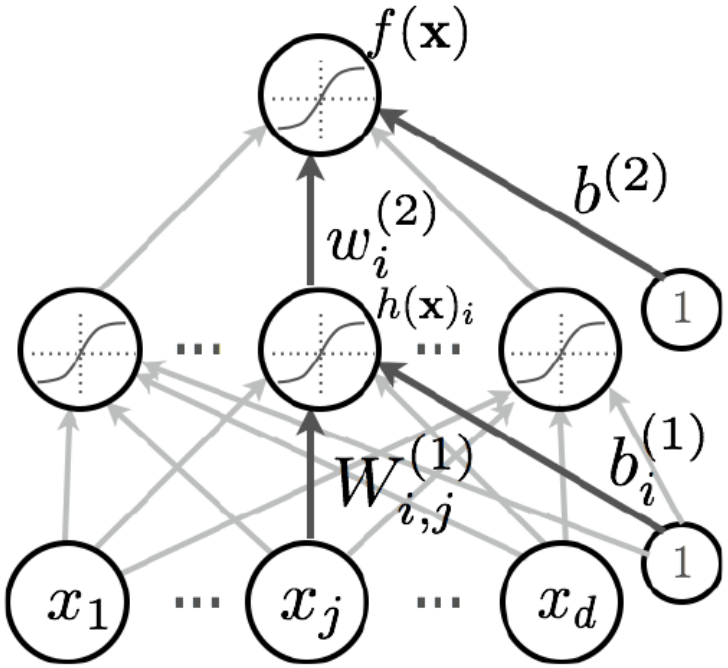
- Hidden layer activation:

$$\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{a}(\mathbf{x}))$$

- Output layer activation:

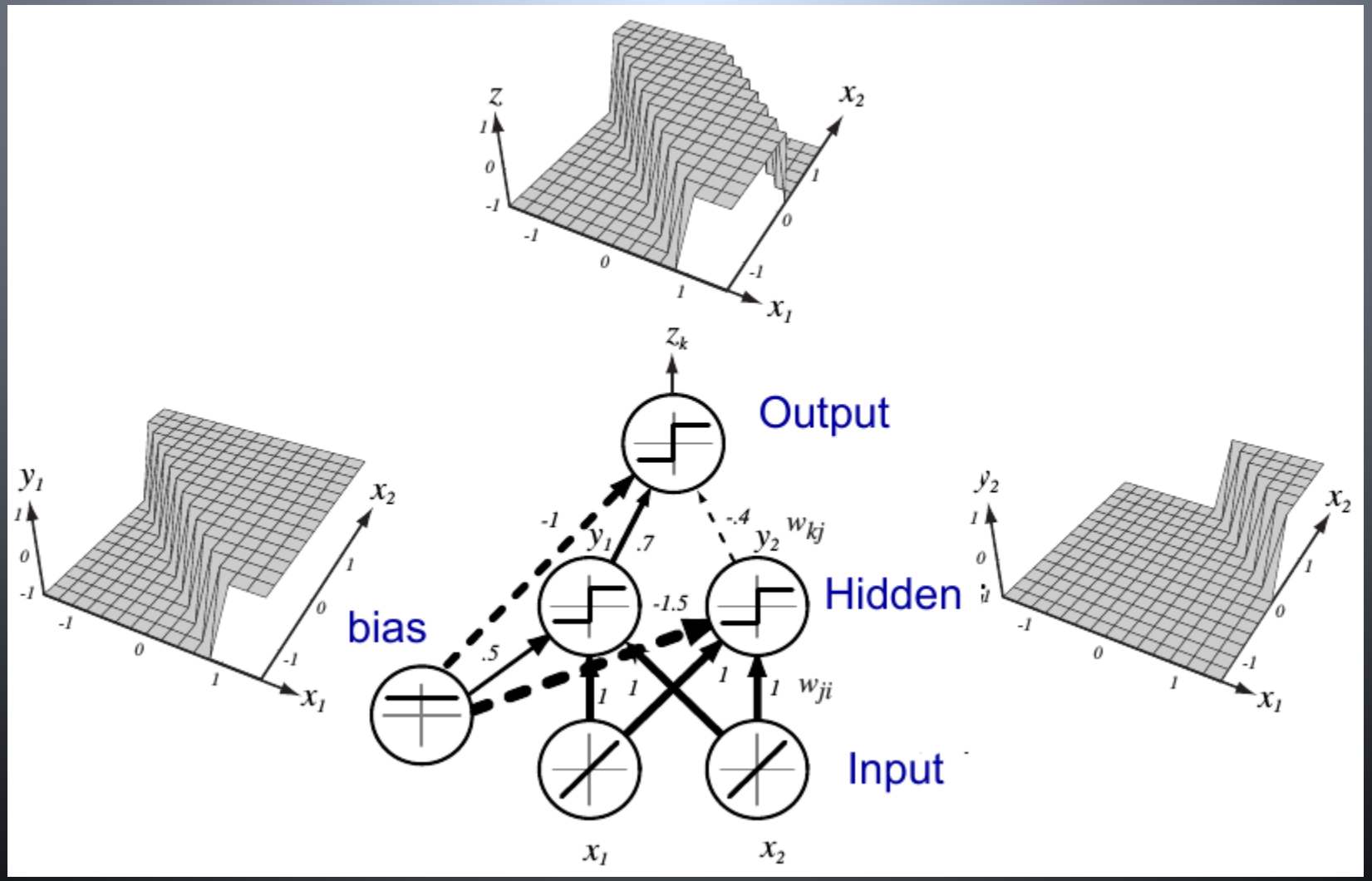
$$f(\mathbf{x}) = o\left(b^{(2)} + \mathbf{w}^{(2)\top}\mathbf{h}^{(1)}\mathbf{x}\right)$$

Output activation function



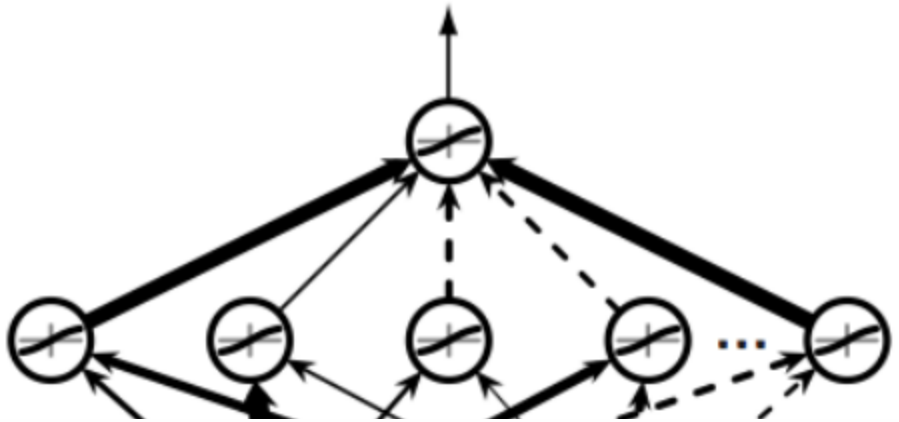
From Prof. Salakhutdinov

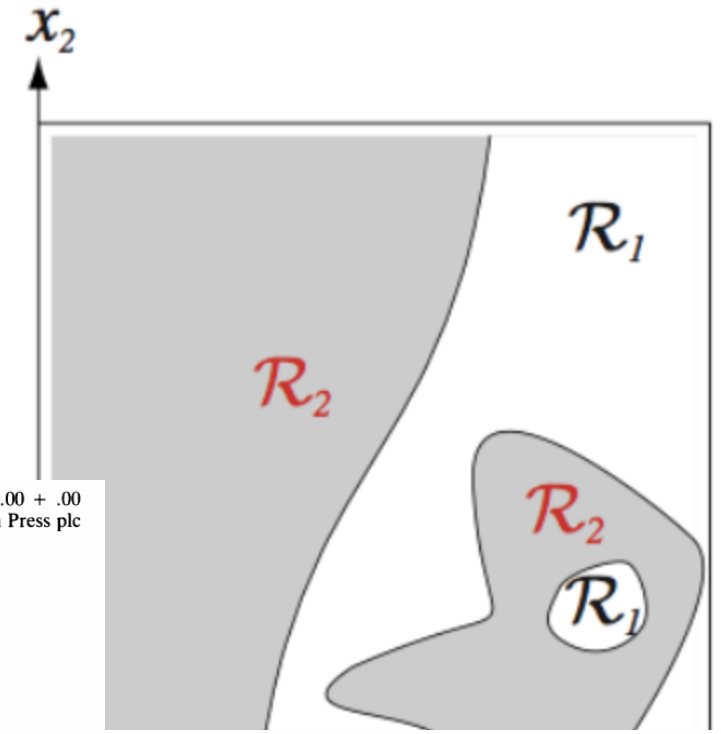
CAPACITY



From Prof. Salakhutdinov

CAPACITY





Neural Networks, Vol. 4, pp. 251–257, 1991
Printed in the USA.

0893-6080/91 $3.00 + .00

ORIGINAL CONTRIBUTION

Approximation Capabilities of Multilayer Feedforward Networks

KURT HORNIK

$$\mathfrak{N}_k^{(n)}(\psi) = \left\{ h: R^k \to R \middle| h(x) = \sum_{j=1}^{n} \beta_j \psi(a_j' x - \theta_j) \right\}$$

Theorem 1: *If ψ is unbounded and nonconstant, then $\mathfrak{N}_k(\psi)$ is dense in $L^p(\mu)$ for all finite measures μ on R^k.*

MULTILAYER NEURAL NET

- Consider a network with L hidden layers.
  - layer pre-activation for k>0

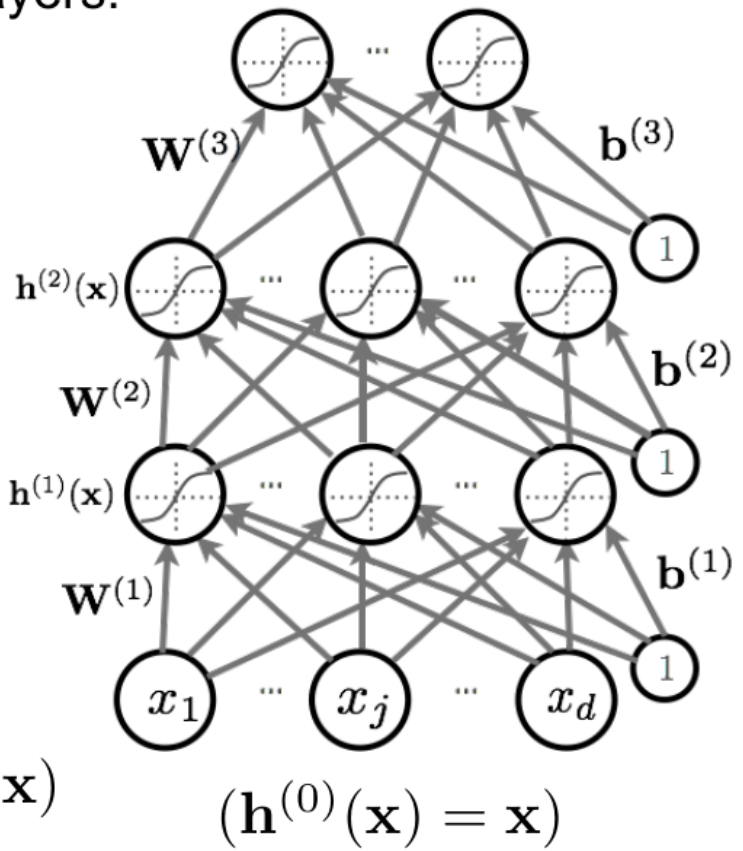
$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation from 1 to L:

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x}))$$

  - output layer activation (k=L+1):

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



From Prof. Salakhutdinov

WELL BUT HOW DO WE LEARN?

- Empirical Risk Minimization:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t} \underbrace{l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})}_{\text{Loss function}} + \underbrace{\lambda \Omega(\boldsymbol{\theta})}_{\text{Regularizer}}$$

- To train a neural net, we need:
  - Loss function: $l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - A procedure to compute gradients: $\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - Regularizer and its gradient: $\Omega(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$

From Prof. Salakhutdinov

WELL BUT HOW DO WE LEARN?

- Perform updates after seeing each example:
  - Initialize: $\boldsymbol{\theta} \equiv \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \ldots, \mathbf{W}^{(L+1)}, \mathbf{b}^{(L+1)}\}$
  - For t=1:T
    - for each training example $(\mathbf{x}^{(t)}, y^{(t)})$

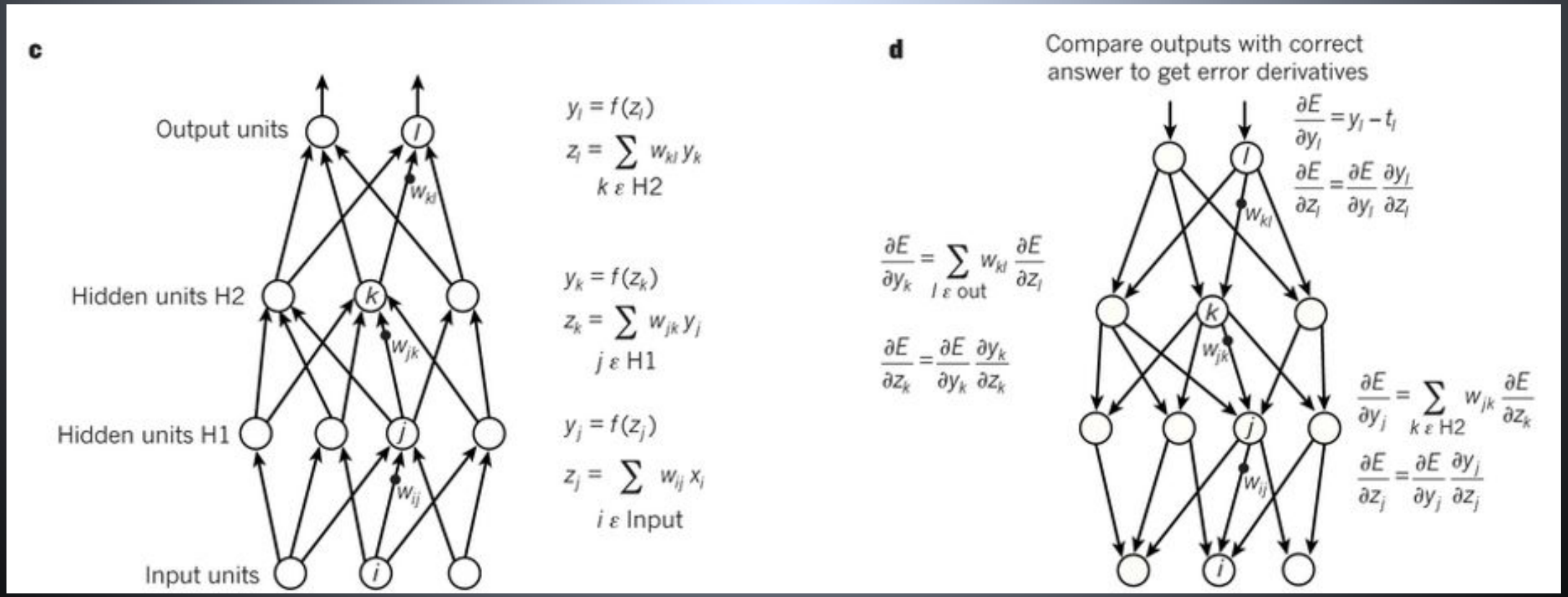
$$\Delta = -\nabla_{\boldsymbol{\theta}} l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) - \lambda \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \, \Delta$$

Training epoch = Iteration of all examples

- To train a neural net, we need:
  - Loss function: $l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - A procedure to compute gradients: $\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - Regularizer and its gradient: $\Omega(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$

From Prof. Salakhutdinov

BACKPROPAGATION (OF ERRORS) ALGORITHM



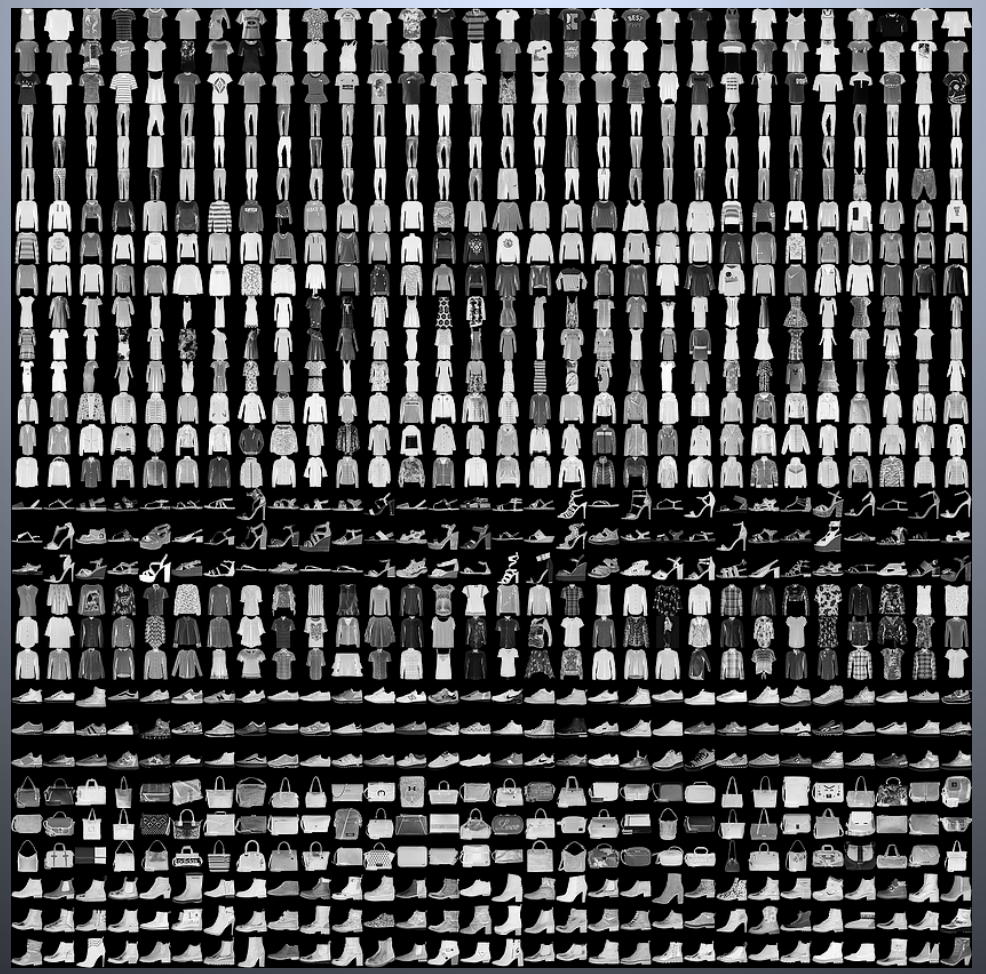
Deep Learning,
LeCun et al.
Nature 2015

Fortunately,

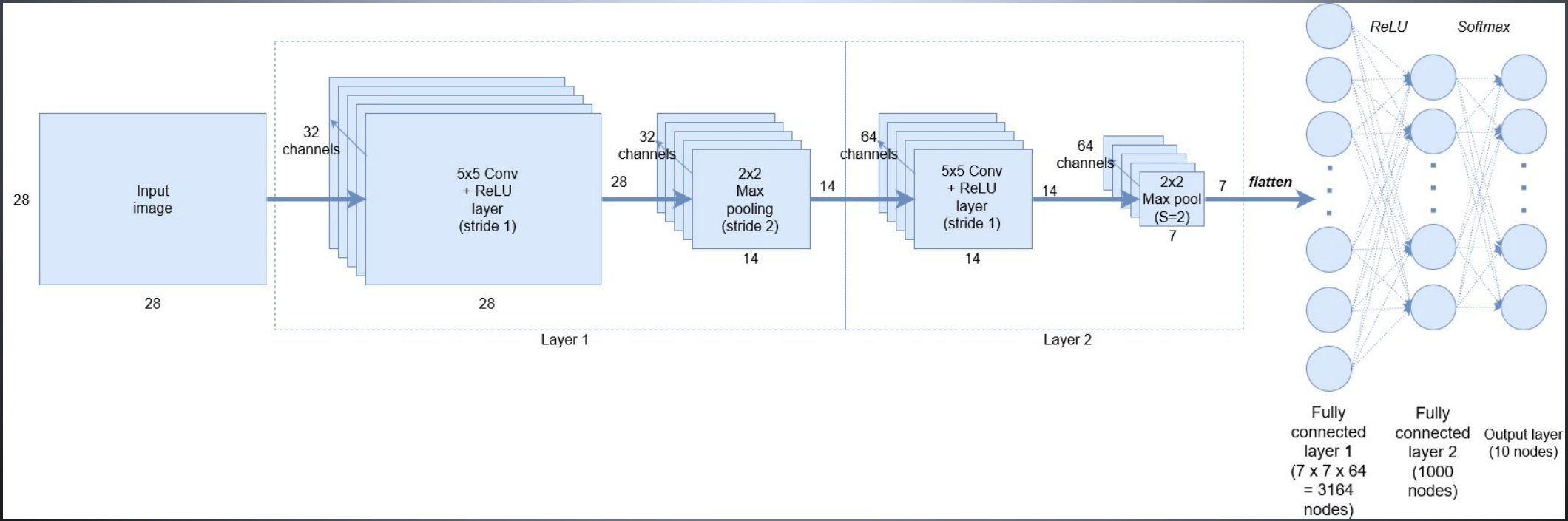
TensorFlow, Keras, Theano, PyTorch…

minimize the loss for us!

Keras demo with MNIST and fashion MNIST….
https://github.com/AlexEMG/Tutorials/blob/master/MNIST-Keras.ipynb



Convolutional Network architecture for Keras model with MNIST performance > 99%

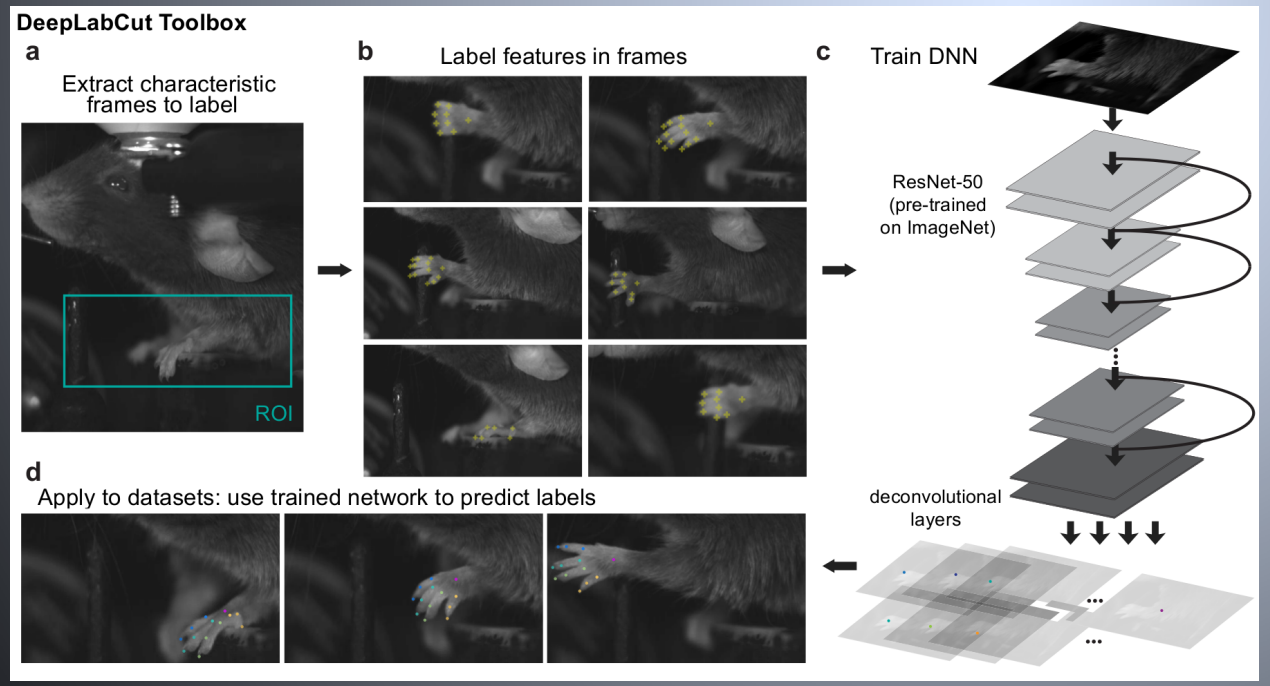


In Keras this is as easy as:
(see notebook for training etc.)

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(tf.keras.layers.Conv2D(64, (5, 5), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1000, activation='relu'))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

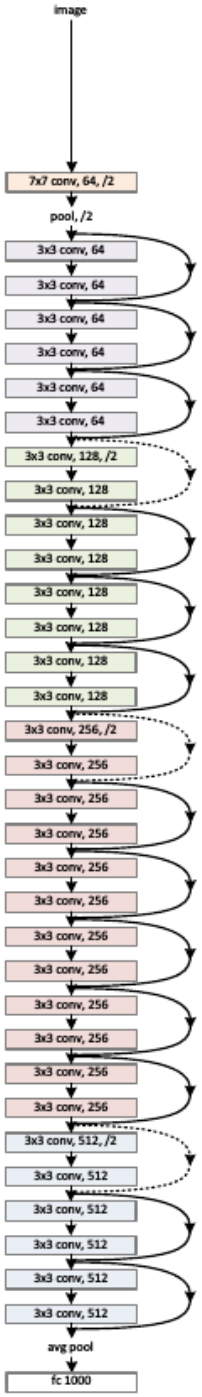
Source: http://adventuresinmachinelearning.com/keras-tutorial-cnn-11-lines/

Transfer learning…



ImageNet (>1Mio images
With 1,000 classes)

ResNet
He et al. 2016



nature neuroscience
TECHNICAL REPORT
https://doi.org/10.1038/s41593-018-0209-y

DeepLabCut: markerless pose estimation of user-defined body parts with deep learning

Alexander Mathis[1,2], Pranav Mamidanna[1], Kevin M. Cury[3], Taiga Abe[3], Venkatesh N. Murthy[2], Mackenzie Weygandt Mathis[1,4,8*] and Matthias Bethge[1,5,6,7,8]

A few links:

- Deep Learning textbook by Goodfellow, Bengio, Courville
- TensorFlow tutorial: https://github.com/chiphuyen/stanford-tensorflow-tutorials
- Awesome interpretability techniques for DeepNets
https://distill.pub/2018/building-blocks/
- Linking neural nets to topology
http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/
https://cs.stanford.edu/people/karpathy/convnetjs//demo/classify2d.html
- Machine Learning Summer School
http://mlss.tuebingen.mpg.de/2017/index.html
- Deep Learning Summer School Materials
https://sites.google.com/site/deeplearningsummerschool2016

info@neurodev.org
amathis@fas.harvard.edu