Introduction to RNNs

Part I

Victoria Zhang

10.4 2022

A woman is throwing a **frisbee** in a park.

A **dog** is standing on a hardwood floor.

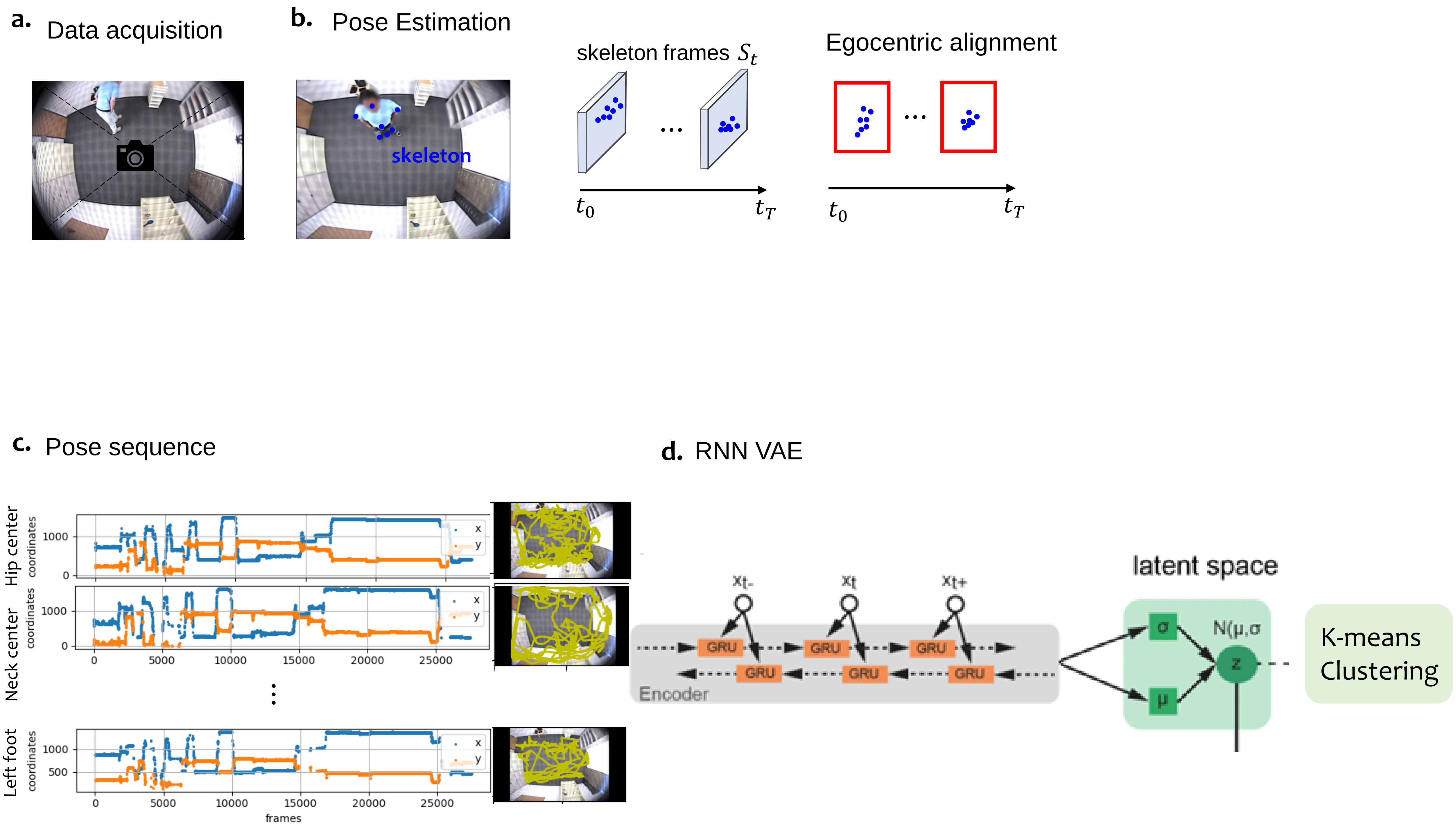
A **stop** sign is on a road with a mountain in the background

A little **girl** sitting on a bed with a teddy bear.

A group of **people** sitting on a boat in the water.

A giraffe standing in a forest with **trees** in the background.

Yann LeCun, Yoshua Bengio & Geoffrey Hinton, Deep Learning, Nature 2015


a. Data acquisition
b. Pose Estimation
skeleton
skeleton frames S_t
t_0
t_T
Egocentric alignment
t_0
t_T
c. Pose sequence
Hip center coordinates
Neck center coordinates
Left foot coordinates
x
y
frames
d. RNN VAE
latent space
Encoder
GRU
N(μ,σ)
σ
μ
z
K-means Clustering

Outline

- Before RNNs: Perceptron and ConvNets
- RNNs, and Why?
- Some Math
  - Forward pass
  - Backpropagation refresher
  - The RNN backward pass
- Some pros and cons
  - On the difficulty of training RNNs
  - Applications

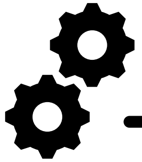
Supervised Learning

Labeled data

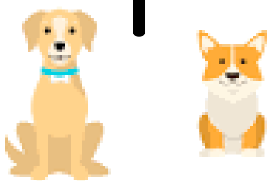
Labels

Pug, Corgi, Golden retriever…

Train model



Prediction



Test data

Golden retriever

Corgi

Supervised Learning

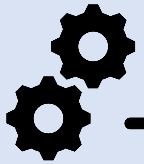
- Compute objective function
- Measure the error (or distance)
- Adjust internal parameters (weights) to reduce the error

Labeled data

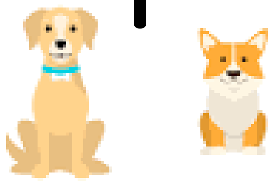
Labels

Pug, Corgi, Golden retriever…

Train model



Prediction

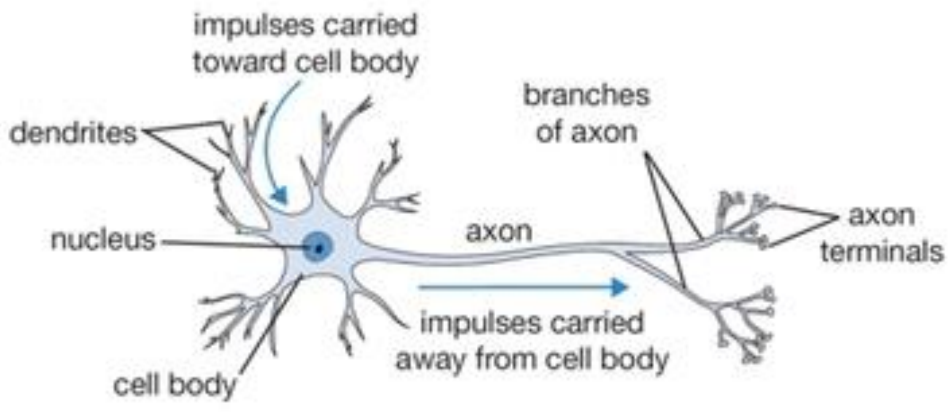


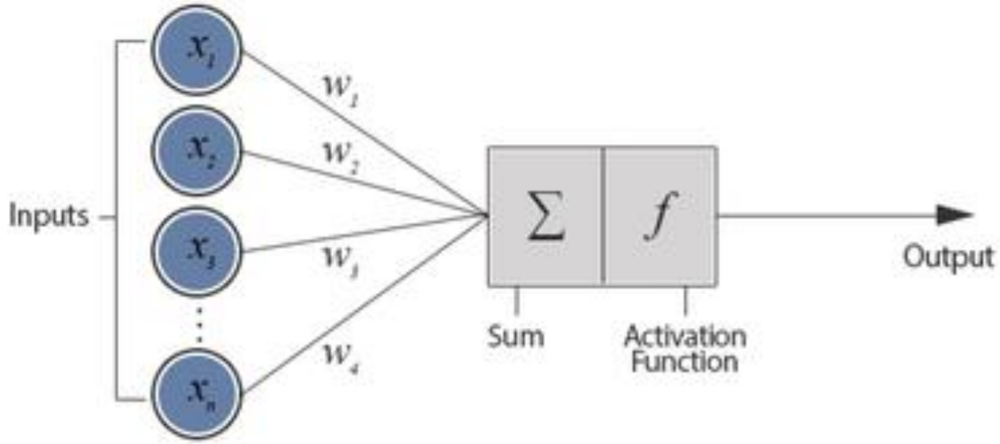
Test data

Golden retriever

Corgi

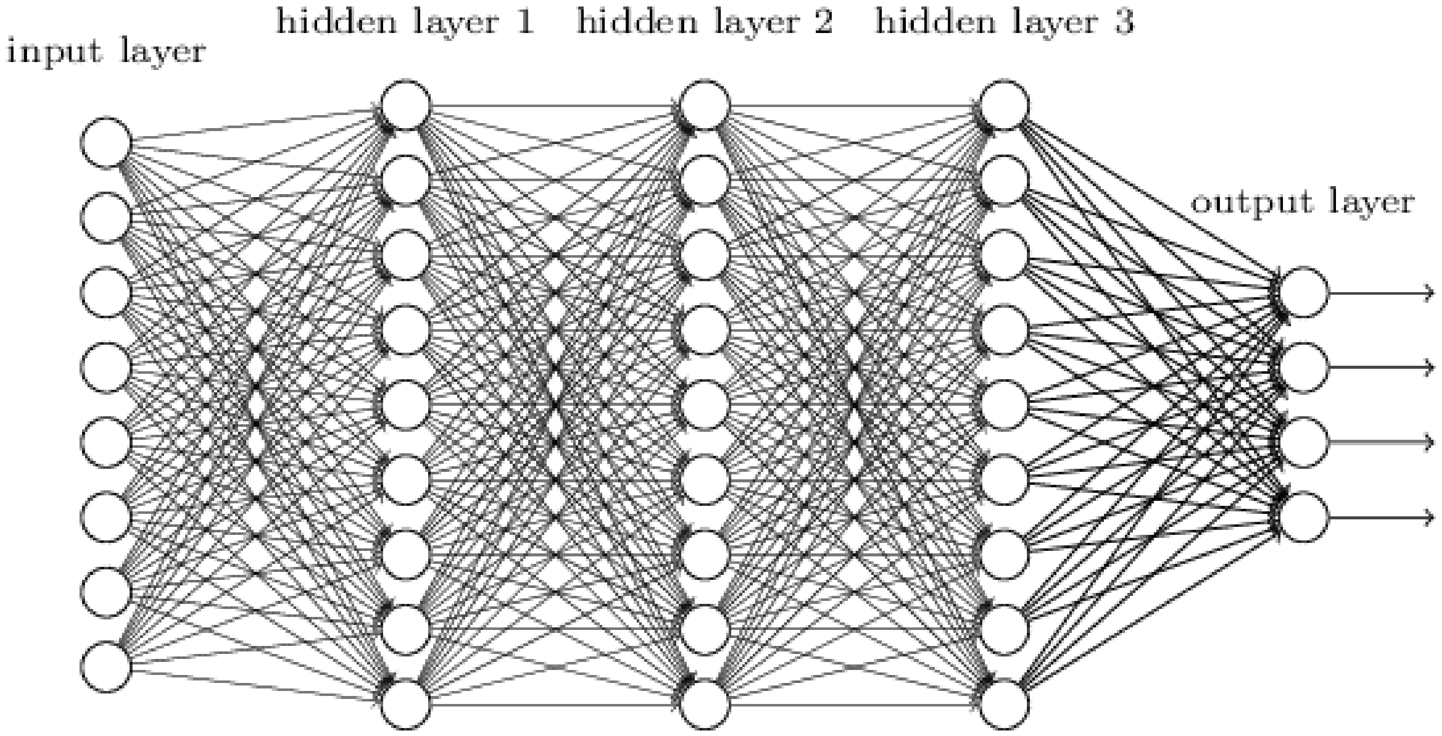
Perceptrons



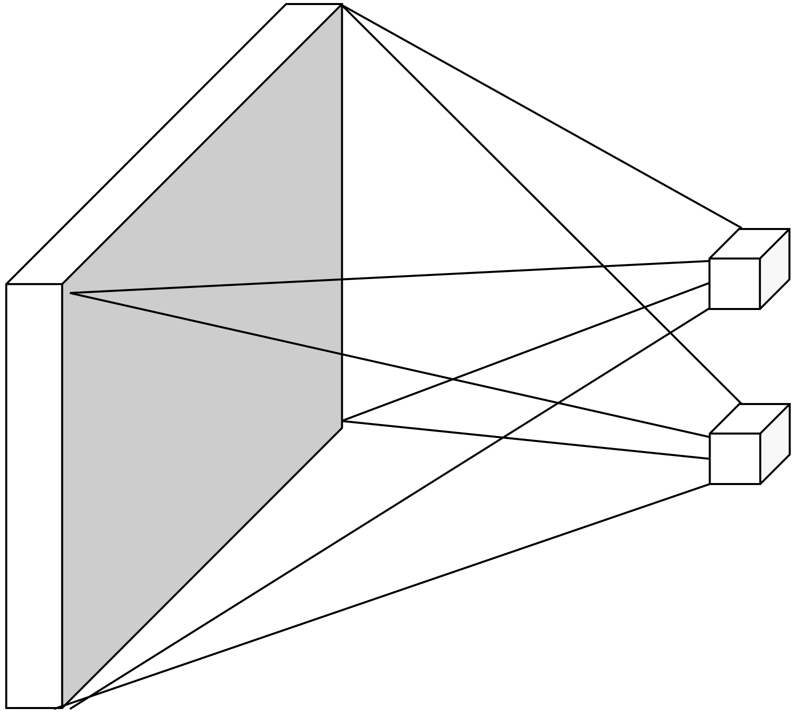


https://www.datacamp.com/tutorial/deep-learning-python

Multi-layer Perceptrons

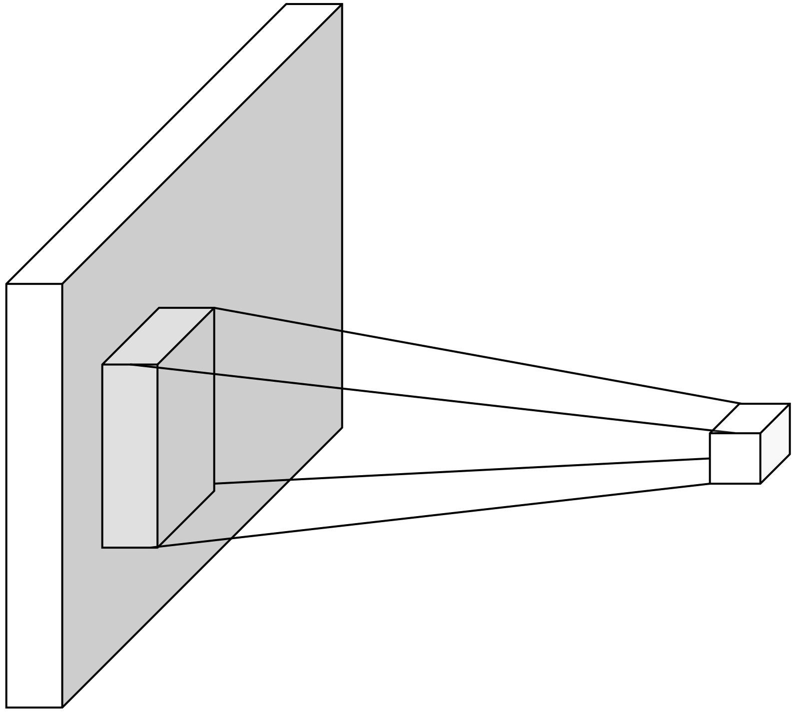


From fully connected to convolution



image

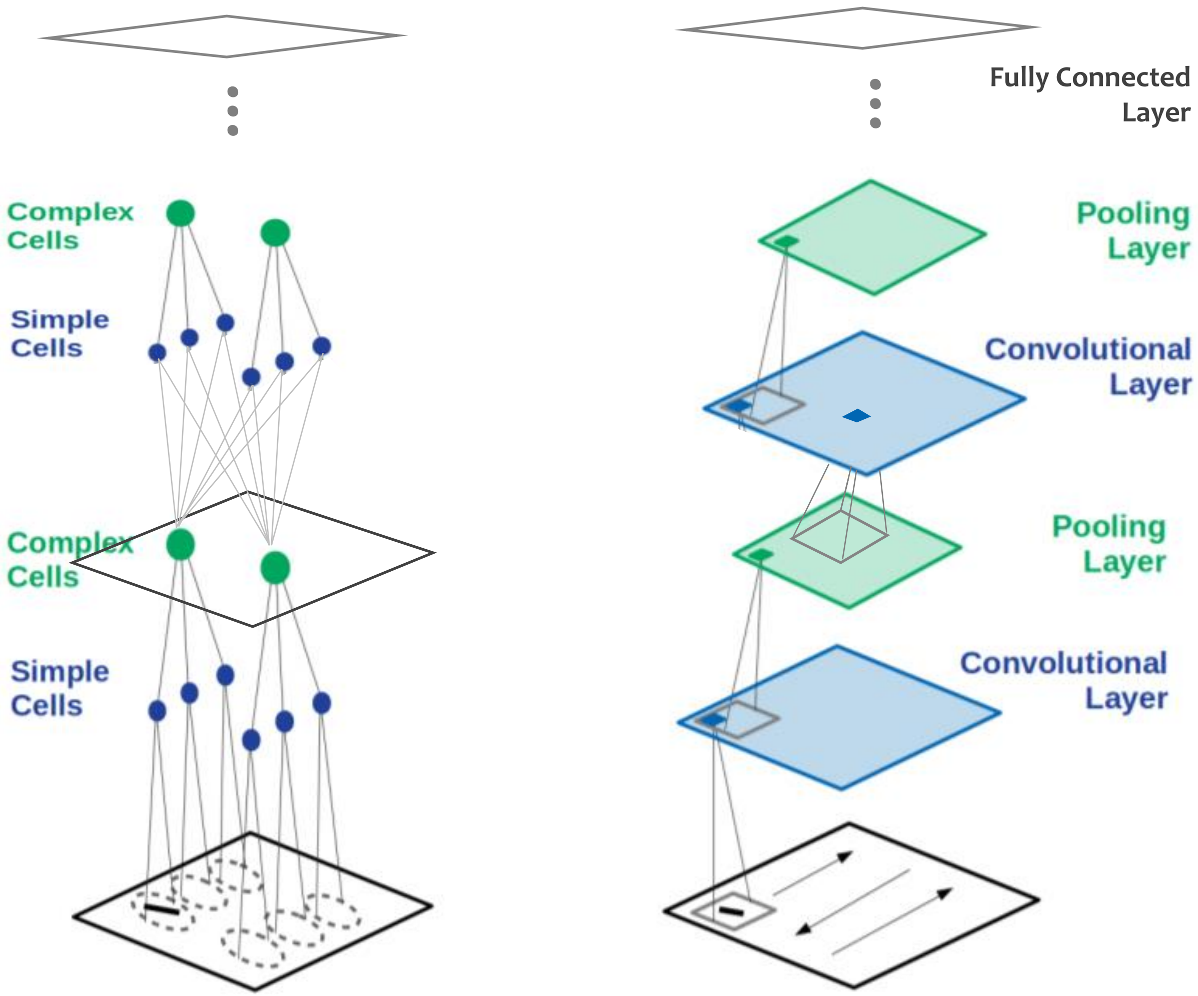
Fully connected layer



image

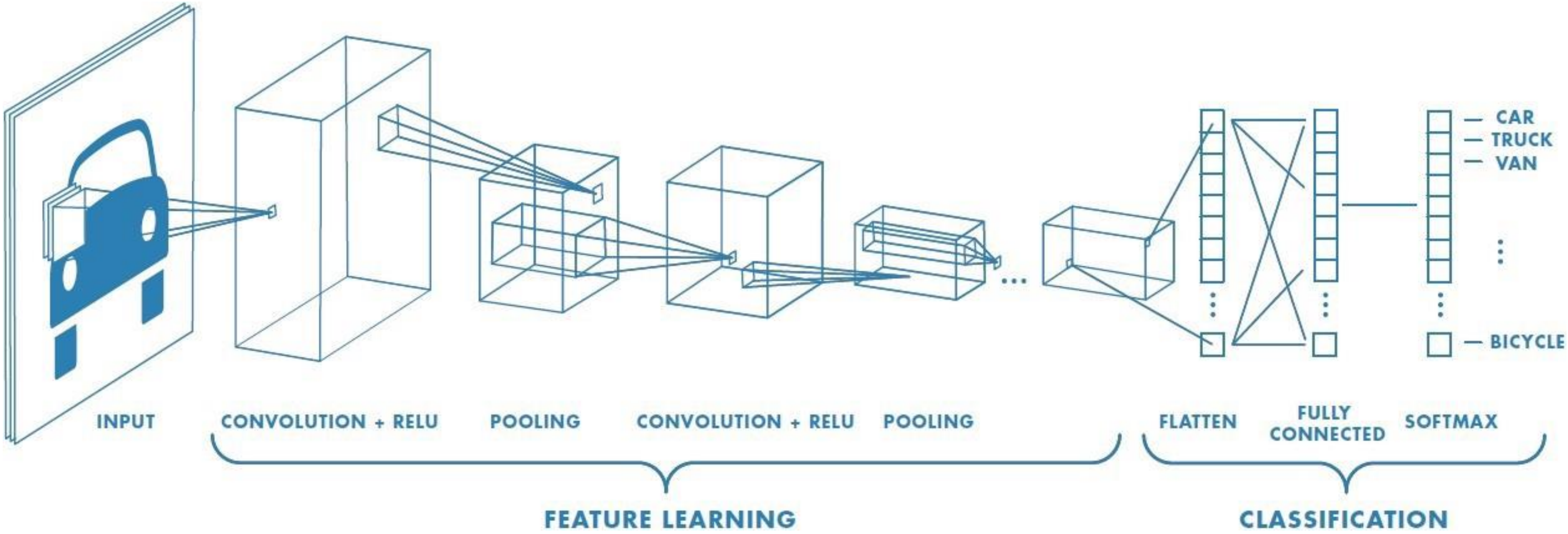
Convolutional layer

Convolutional neural networks (CNNs)



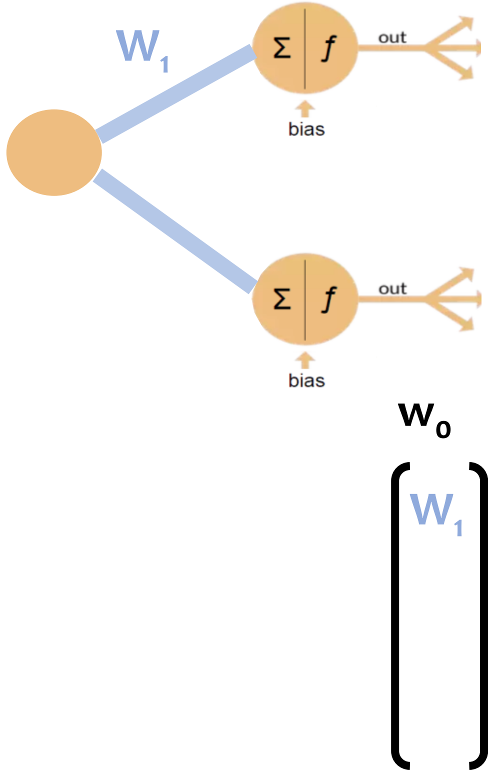
Lindsay, Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future, 2020

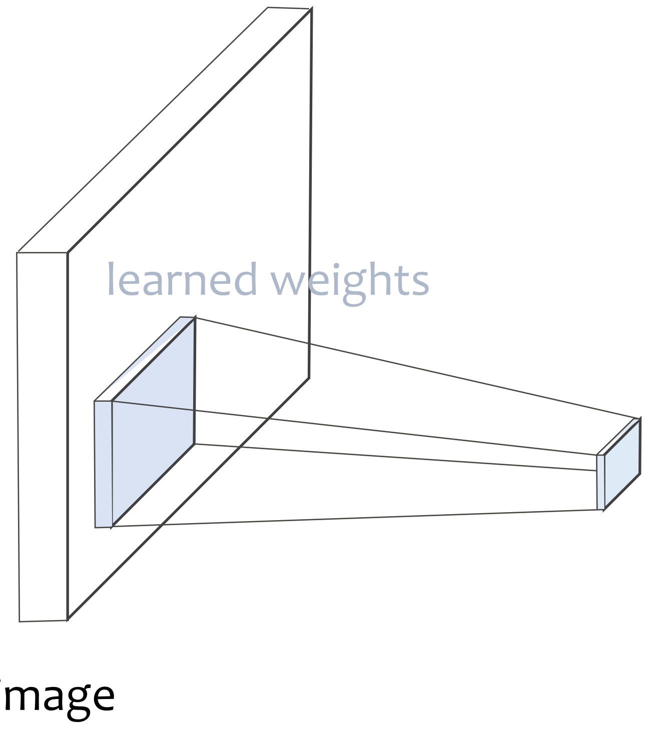
Convolutional neural networks (CNNs)



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks

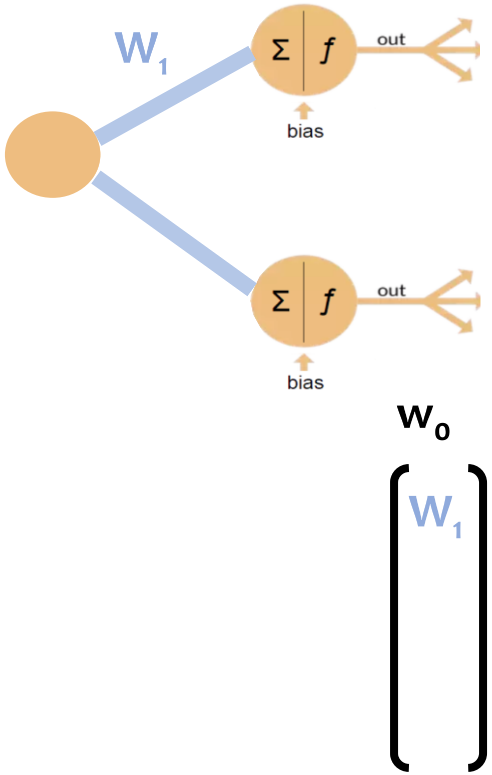
Convolutional neural networks

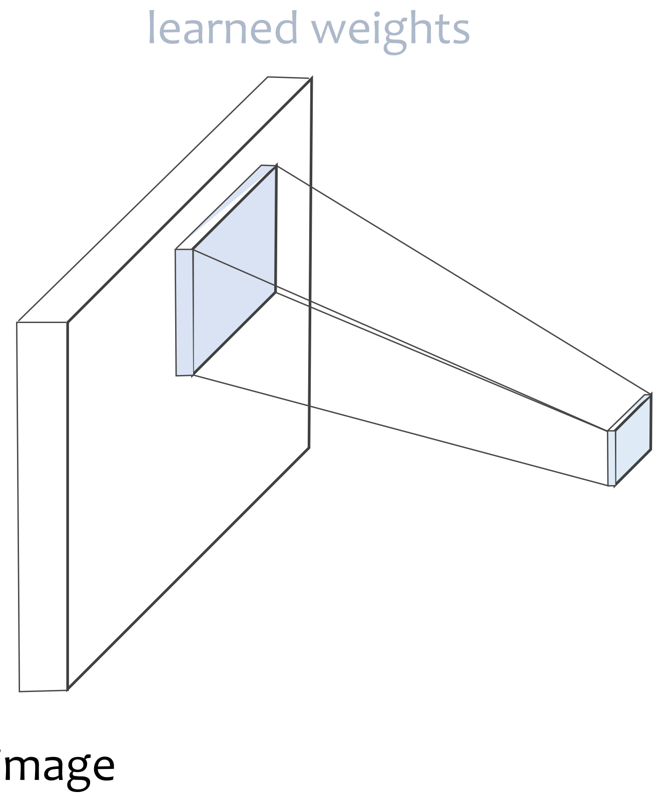




https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/

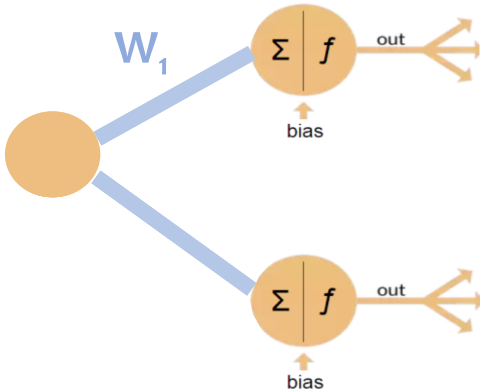
Convolutional neural networks





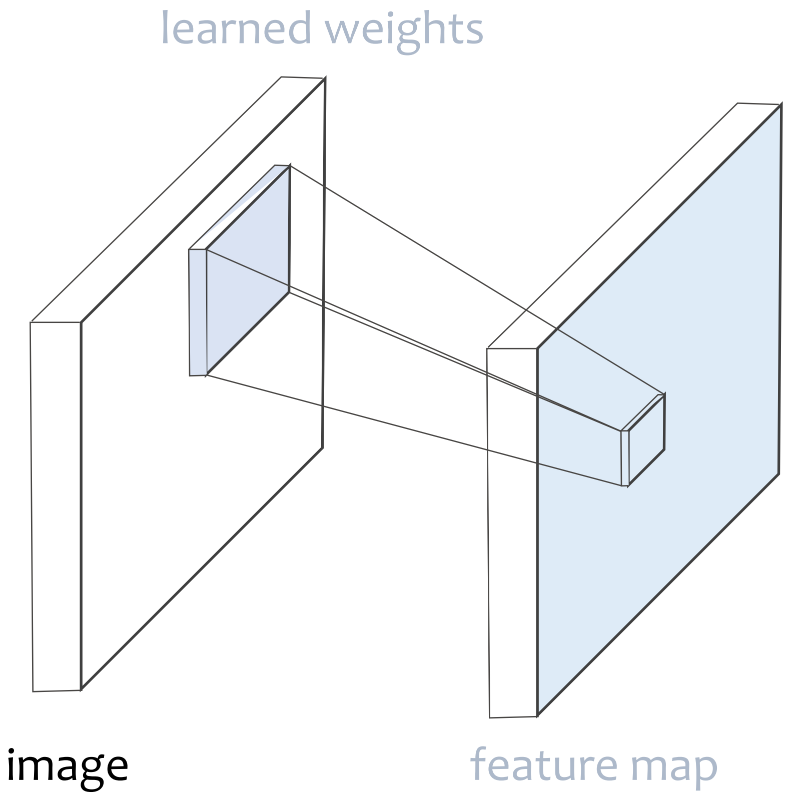
https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/

Convolutional neural networks

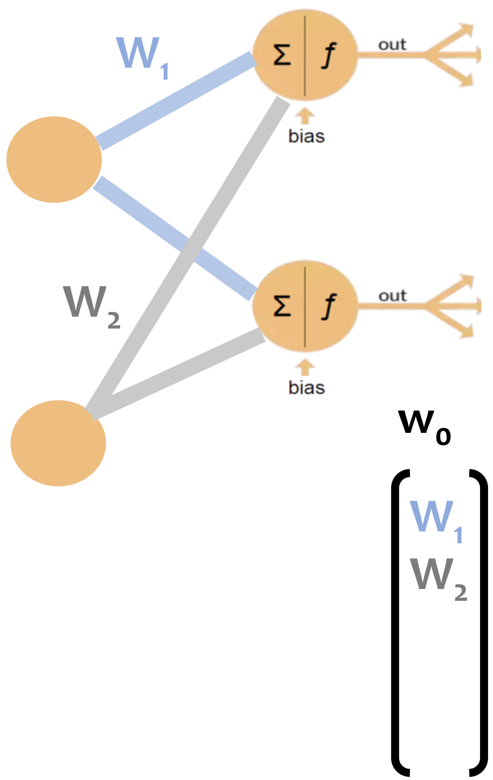


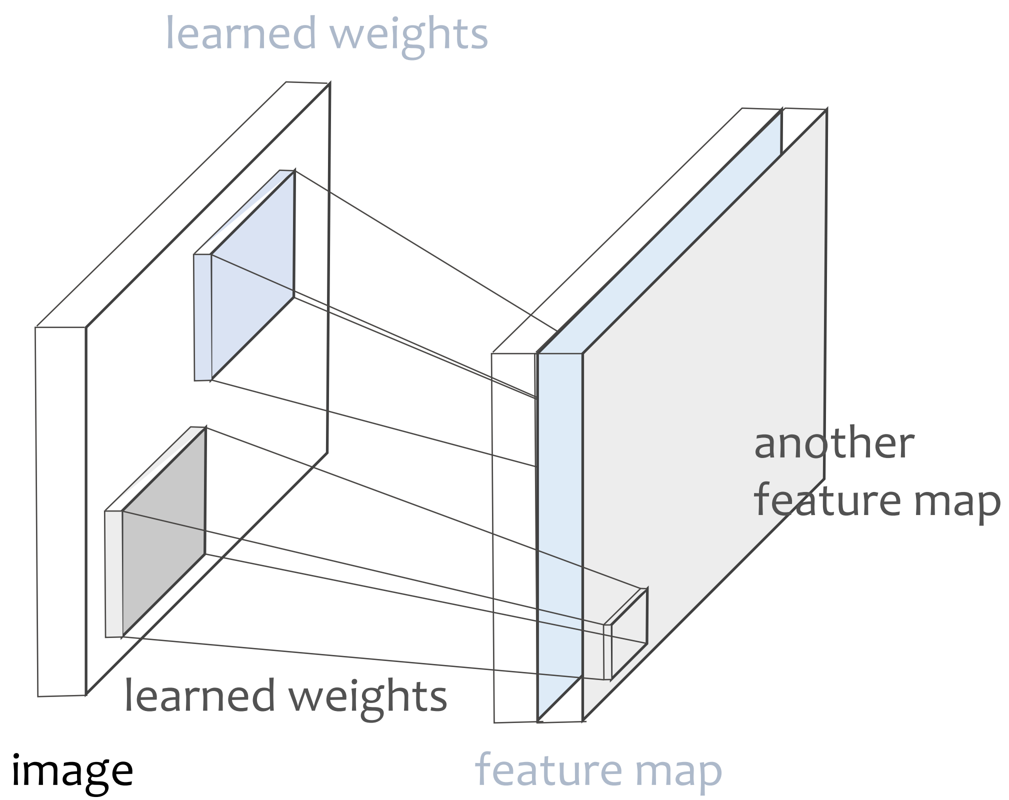
$\mathbf{w}_0$

$$\begin{bmatrix} W_1 \\ \\ \\ \end{bmatrix}$$

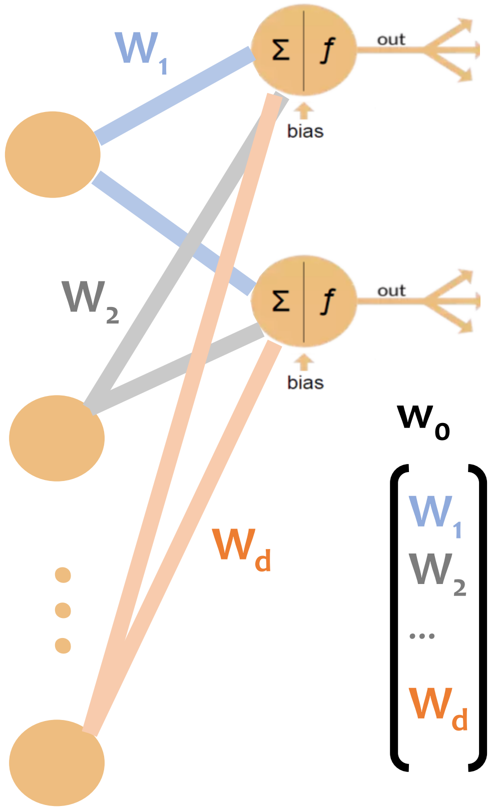


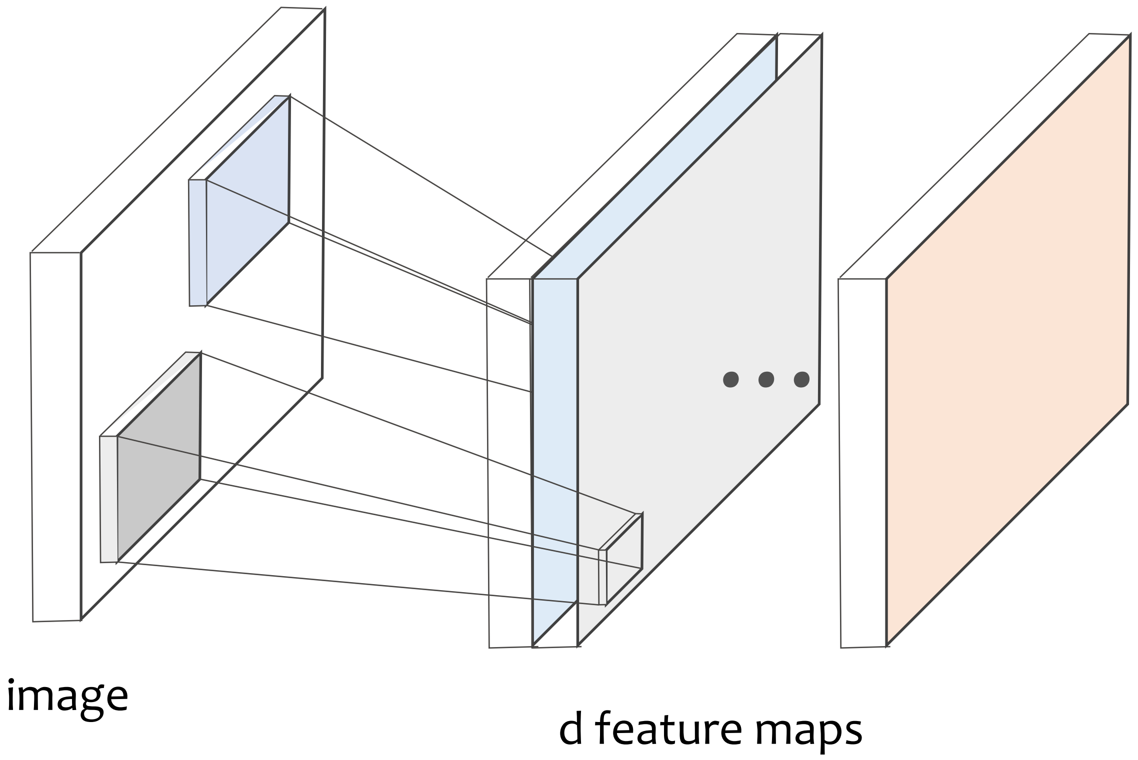
Convolutional neural networks





Convolutional neural networks

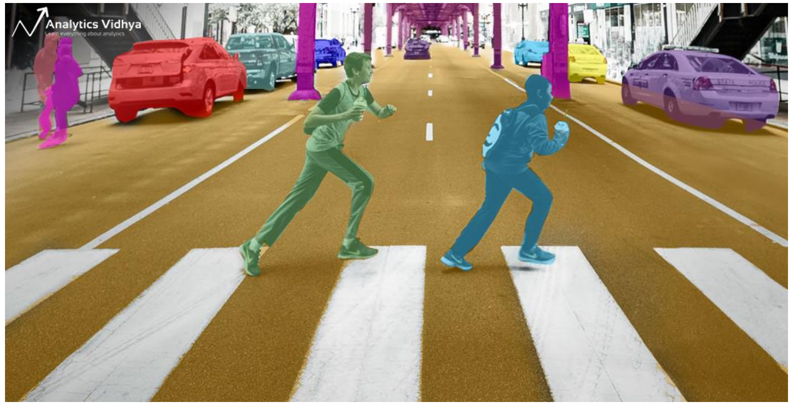




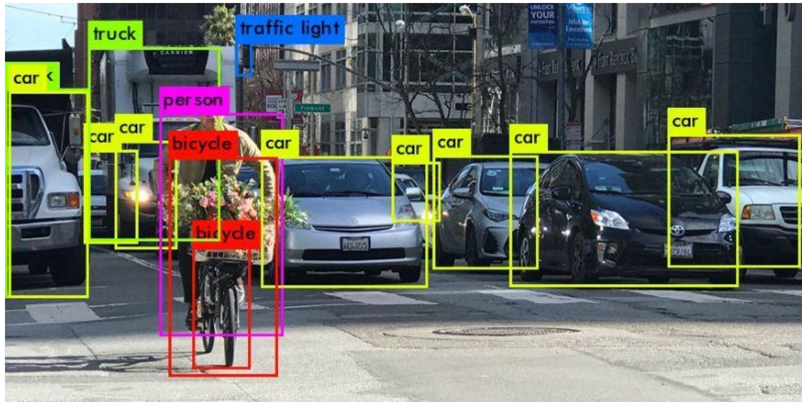
https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/

Image understanding with deep CNNs

Detection



Segmentation



Recognition

What if the input/output is ***speech, texts*** or ***time-series***?
Not all problems can be converted into one with **fixed-length** inputs and outputs

Outline

- Perceptron and ConvNets
- RNNs, and Why RNNs
- Some Math
  - Forward pass
  - On the difficulty of training RNNs

Finding Structure in Time

JEFFREY L. ELMAN

University of California, San Diego

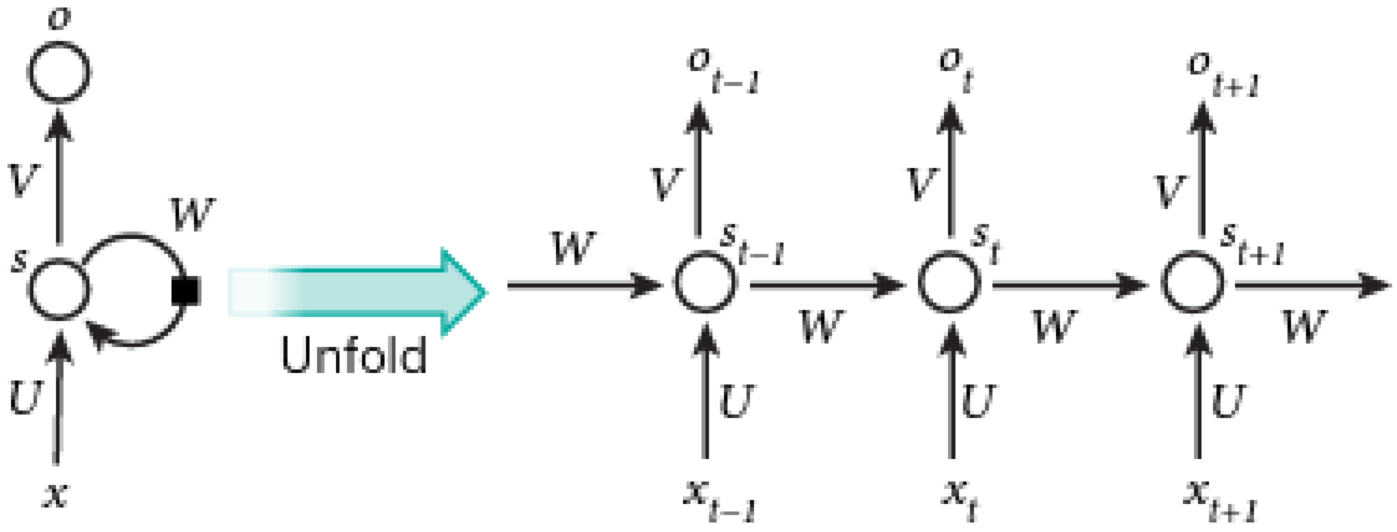
The question of how to represent time might seem to arise as a special problem unique to parallel-processing models, if only because the parallel nature of computation appears to be at odds with the serial nature of temporal events.

The recurrent connections allow the network's hidden units to see its own previous output, so that the subsequent behavior can be shaped by previous responses. These recurrent connections are what give the network memory.

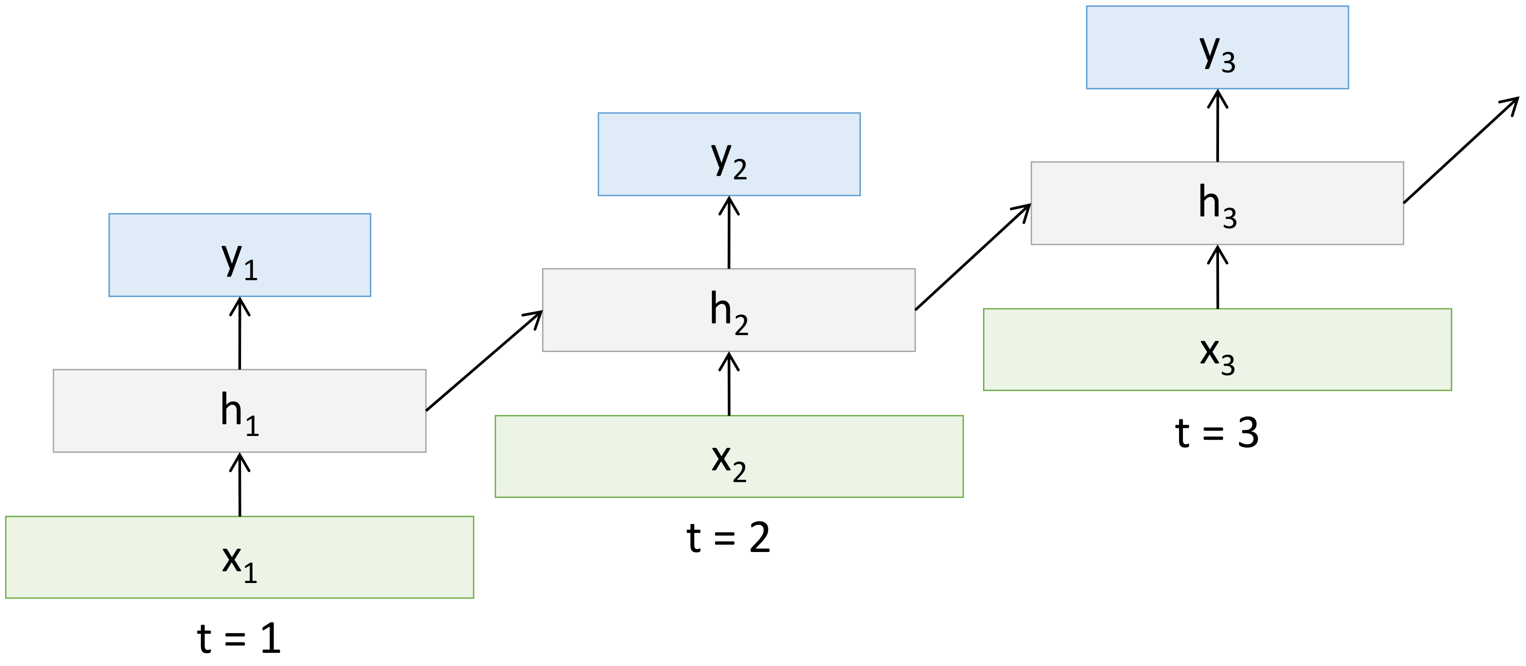
Recurrent Neural Networks (RNNs)

- RNNS take the previous output or hidden states as inputs.
- The composite input at time t has some historical information about the happenings at time T < t

- RNNs are useful as their intermediate values (state) can store information about past inputs for a time that is not fixed a priori



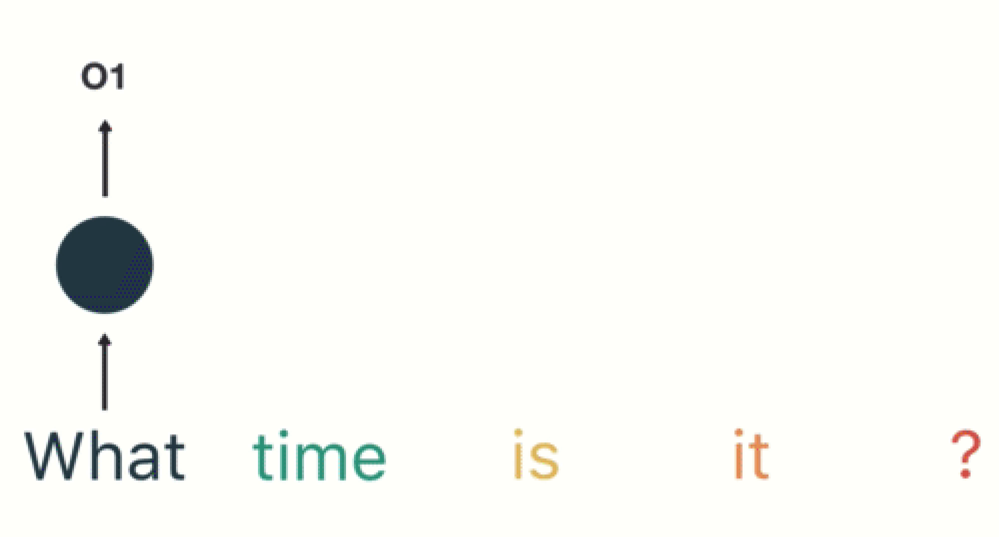
Sample RNN



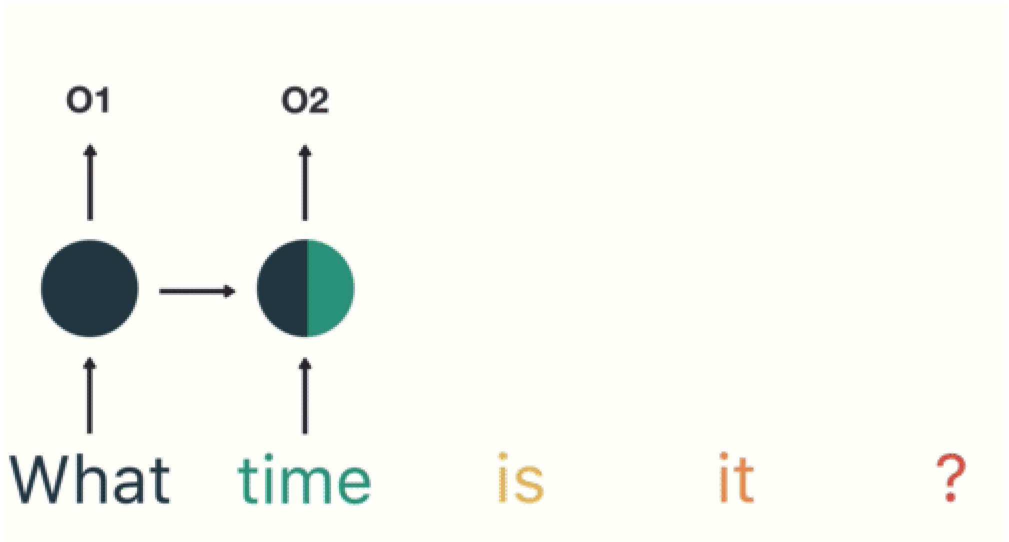
What time is it?

https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

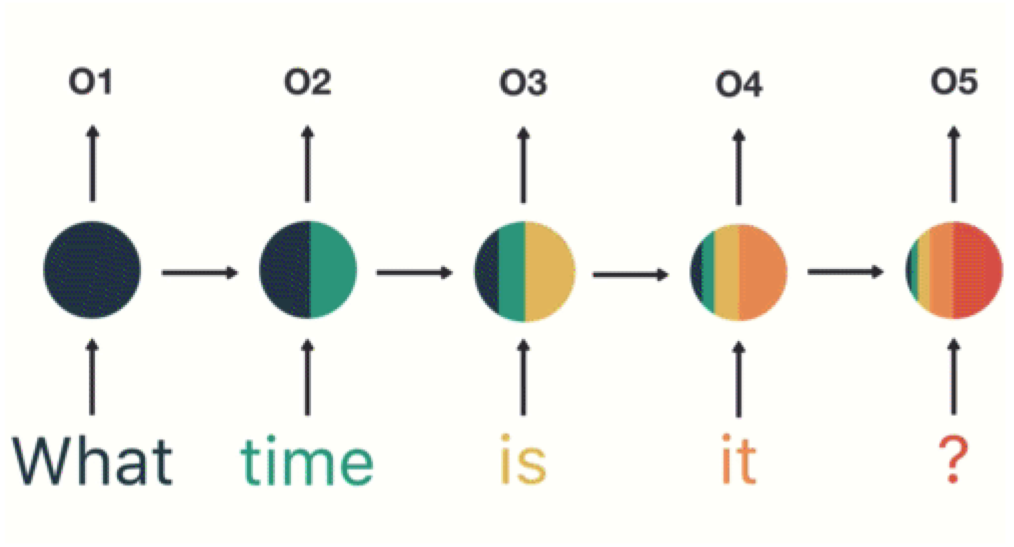
https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9



https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9



https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

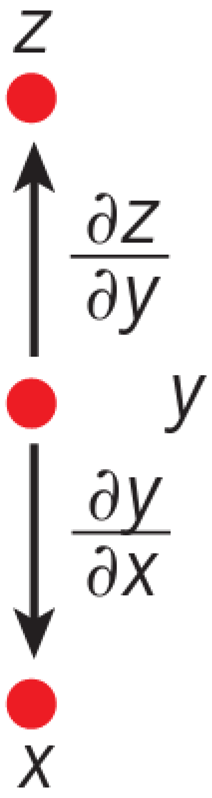


https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

Outline

- Perceptron and ConvNets
- Why RNNs?
- Some Math
  - Forward pass
  - Backpropagation refresher
  - The RNN backward pass
- Some pros and cons
  - On the difficulty of training RNNs
  - Applications

Math time: the chain rule



$$\Delta z = \frac{\partial z}{\partial y} \Delta y$$

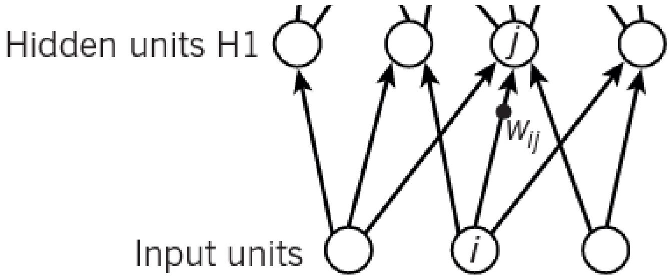
$$\Delta y = \frac{\partial y}{\partial x} \Delta x$$

$$\Delta z = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \Delta x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

Yann LeCun, Yoshua Bengio & Geoffrey Hinton, Deep Learning, Nature 2015

Feedforward

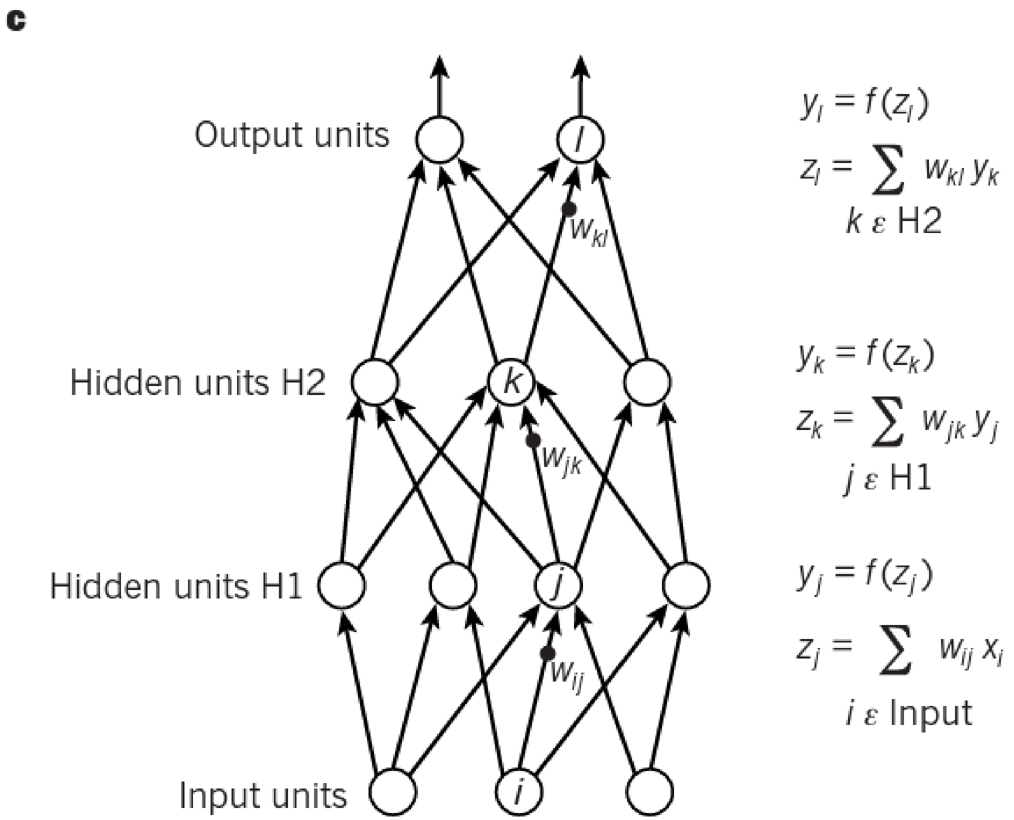


$$y_j = f(z_j)$$

$$z_j = \sum_{i \,\varepsilon\, \text{Input}} w_{ij} x_i$$

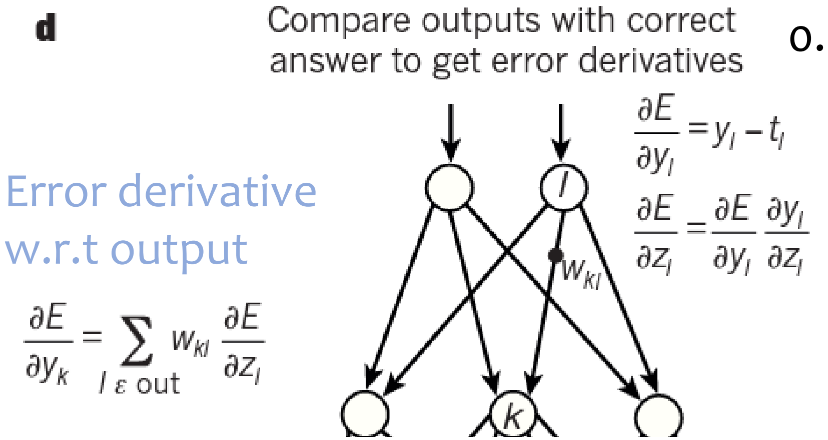
Yann LeCun, Yoshua Bengio & Geoffrey Hinton, Deep Learning, Nature 2015

Feedforward vs Backpropagation

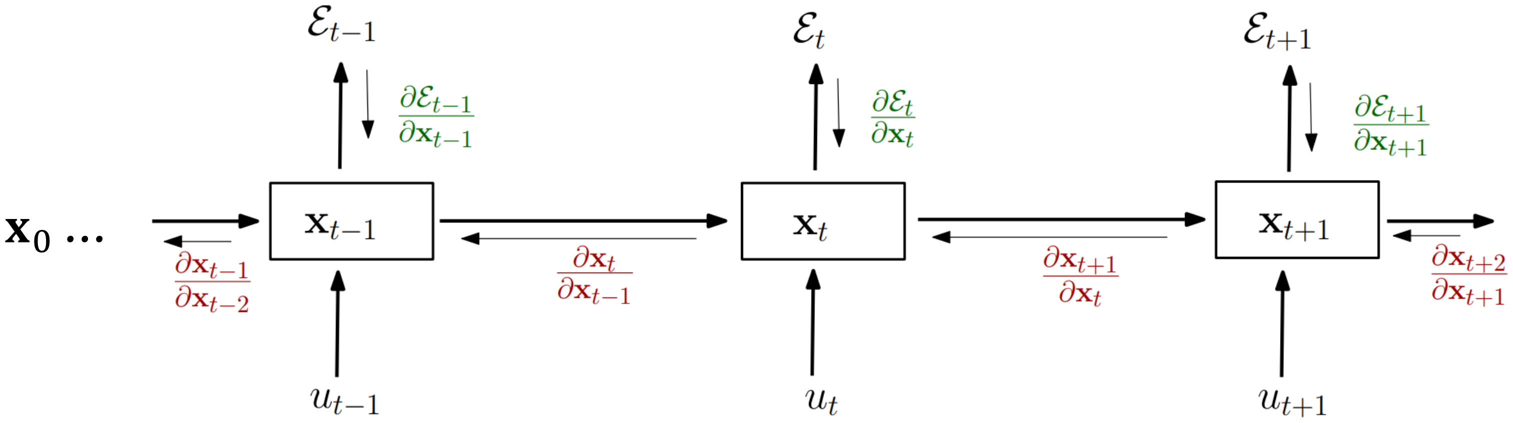


cost function for unit l
0.5(yl – tl)^2

Error derivative
w.r.t output



The RNN backward pass



Hidden state

$$\mathbf{x}_t = F(\mathbf{x}_{t-1}, \mathbf{u}_t, \theta)$$

$$\mathbf{x}_t = \sigma(\mathbf{W}_{rec}\mathbf{x}_{t-1} + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b})$$

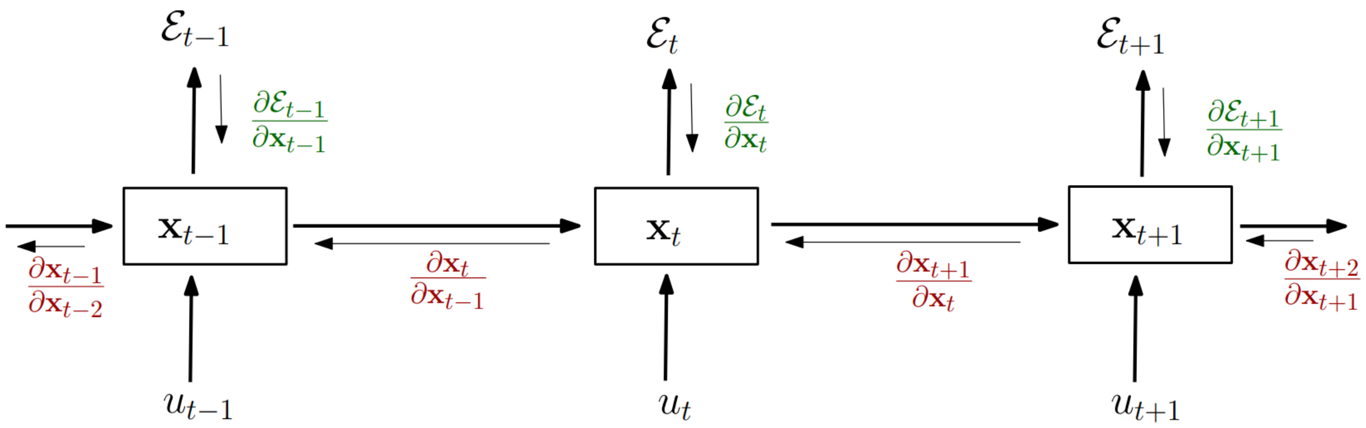
Cost

$$\mathcal{E} = \sum_{1 \le t \le T} \mathcal{E}_t$$

$$\mathcal{E}_t = \mathcal{L}(\mathbf{x}_t)$$

R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, ICML 2013

Back Propagation Through Time (BPTT)



$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \leq t \leq T} \frac{\partial \mathcal{E}_t}{\partial \theta}$$

Temporal contribution:
how θ at step k affects the cost at step t > k.

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \leq k \leq t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right)$$

Long -and short- term contributions:
transport the error “in time“ from step t back to step k.

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}}$$

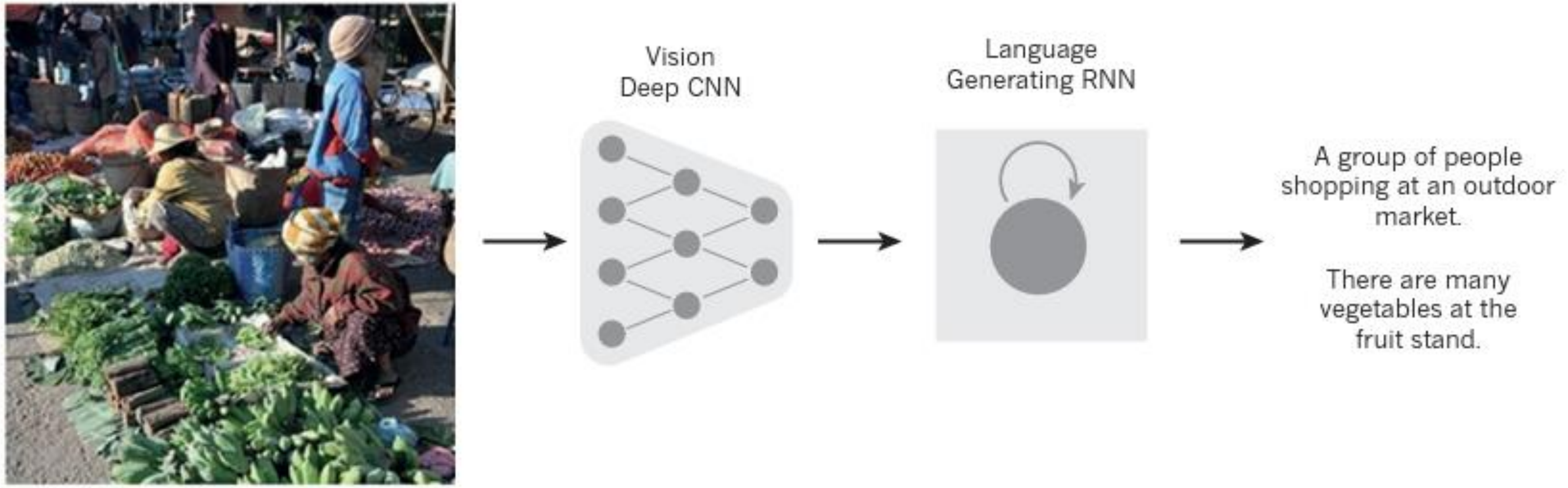
R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, ICML 2013

Outline

- Perceptron and ConvNets
- Why RNNs?
- Some Math
  - Forward pass
  - Backpropagation refresher
  - The RNN backward pass
- Some pros and cons
  - On the difficulty of training RNNs
  - Applications

RNN applications

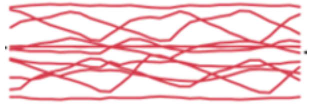
- English sentence -> French sentence
- Image Captioning



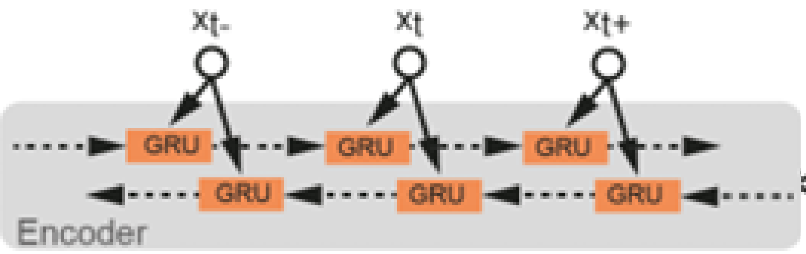
Sutskever, I. Vinyals, O. & Le. Q. V. Sequence to sequence learning with neural networks. In Proc. Advances in Neural Information Processing Systems 27 3104–3112 (2014).

VAME model:
bidirectional RNN VAE
(time window: 30)

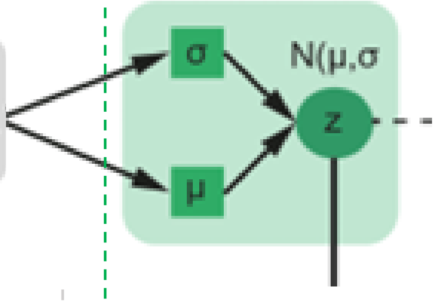
time sequence
(D: 2k x T)



$X = \{x_1, x_2, \dots x_{2k}\}$



latent space (D: m x T)



Internal state at each time step h_t

on updates:

$$\mathbf{h}_t^f = tanh(f_\phi(\mathbf{x}_t, \mathbf{h}_{t-1}))$$

$$\mathbf{h}_c = \mathbf{h}_t^f + \mathbf{h}_t^b$$

$$\mathbf{h}_t^b = tanh(f_\phi(\mathbf{x}_t, \mathbf{h}_{t+1})),$$

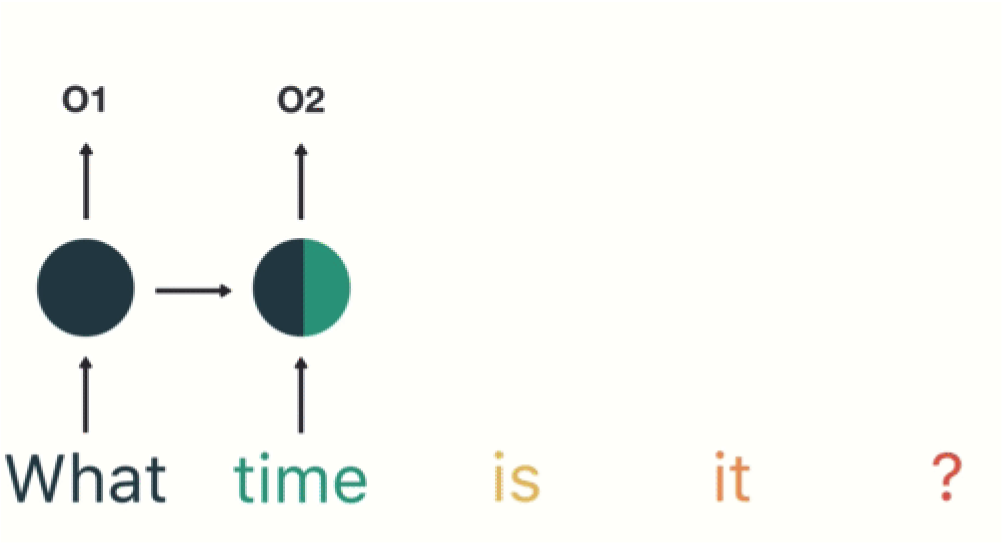
h_t^f: hidden info of the forward pass
h_t^b: hidden info of the backward pass
f: gated recurrent units as transition func

$$h_i = h_i^f + h_i^b$$

Prior: $p_\theta(\boldsymbol{z}_i) \sim N(\boldsymbol{z}_i; \boldsymbol{0}, \boldsymbol{I})$
Approximate posterior: $q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i)$ μ_z, Σ_z
$\boldsymbol{z}_i = \mu_z + \sigma_z \odot \varepsilon$

$Z = \{z_1, z_2, \dots z_m\}$
$m \approx 10$

Vanishing and the Exploding Gradient



Recall:

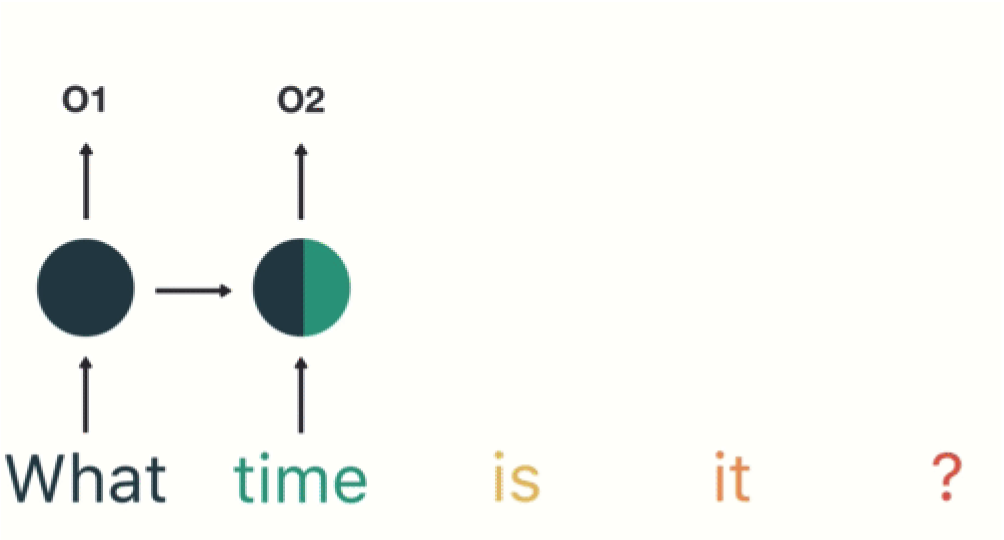
Long -and short- term contributions:
transport the error "in time" from step t back to step k.

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \geq i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1}))$$

Shrink to **zero** or Explode to **infinity**

https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9
R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, ICML 2013

Vanishing and the Exploding Gradient



Recall:

Long -and short- term contributions:
transport the error "in time" from step t back to step k.

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \geq i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1}))$$

1. Small gradients
2. Internal weights barely change
3. The earlier layers fail to do any learning
4. RNN doesn't learn the long-range dependencies across time steps

https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9
R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, ICML 2013

Vanishing and the Exploding Gradient

Long -and short- term contributions:
transport the error "in time" from step t back to step k.

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \geq i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1}))$$

It is *sufficient* for the largest eigenvalue λ1 of the $\mathbf{W}_{recc}$ to be < 1 for long term components to **vanish** (as t ⟶ ∞),

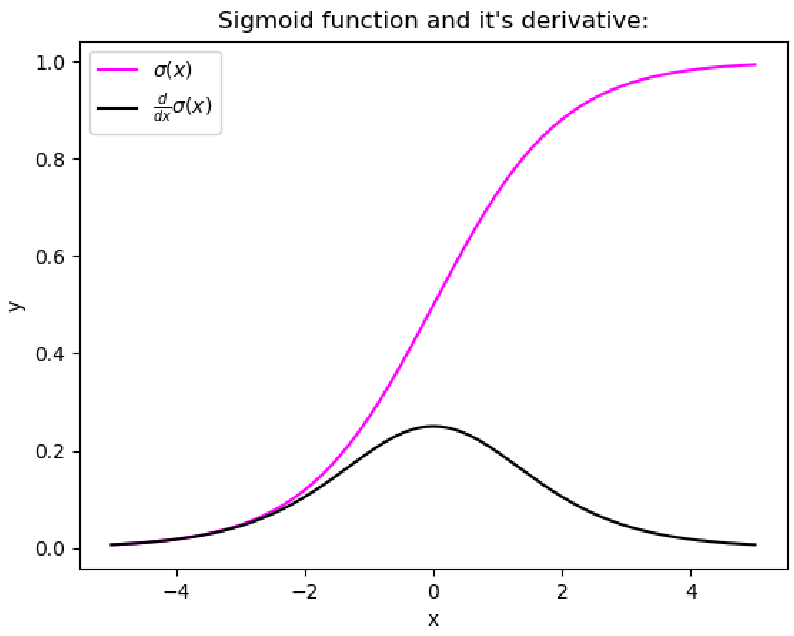
and *necessary* for it to be > 1 for gradients to **explode.**

https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9
R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, ICML 2013

Vanishing and the Exploding Gradient

- **Activation functions** like sigmoid. For larger inputs, it saturates at 0 or 1 with a derivative very close to 0, leading to ~ no gradient at back prob

- **Initial weights** assigned to the network generate some large loss. Gradients accumulate and eventually result in large updates to the network weights. Overflow and NaN values



Solutions

- Proper Weight Initialization
  - The variance of outputs of each layer should = the variance of its inputs.
  - The gradients should have equal variance before and after flowing through a layer in the reverse direction.
- Using Non-saturating Activation Functions
  - e.g. ReLU, Leaky ReLU
- Batch Normalization
  - let the model learn the optimal scale and mean of each of the layer's inputs.
- Gradient Clipping
  - The threshold is a hyperparameter we can tune

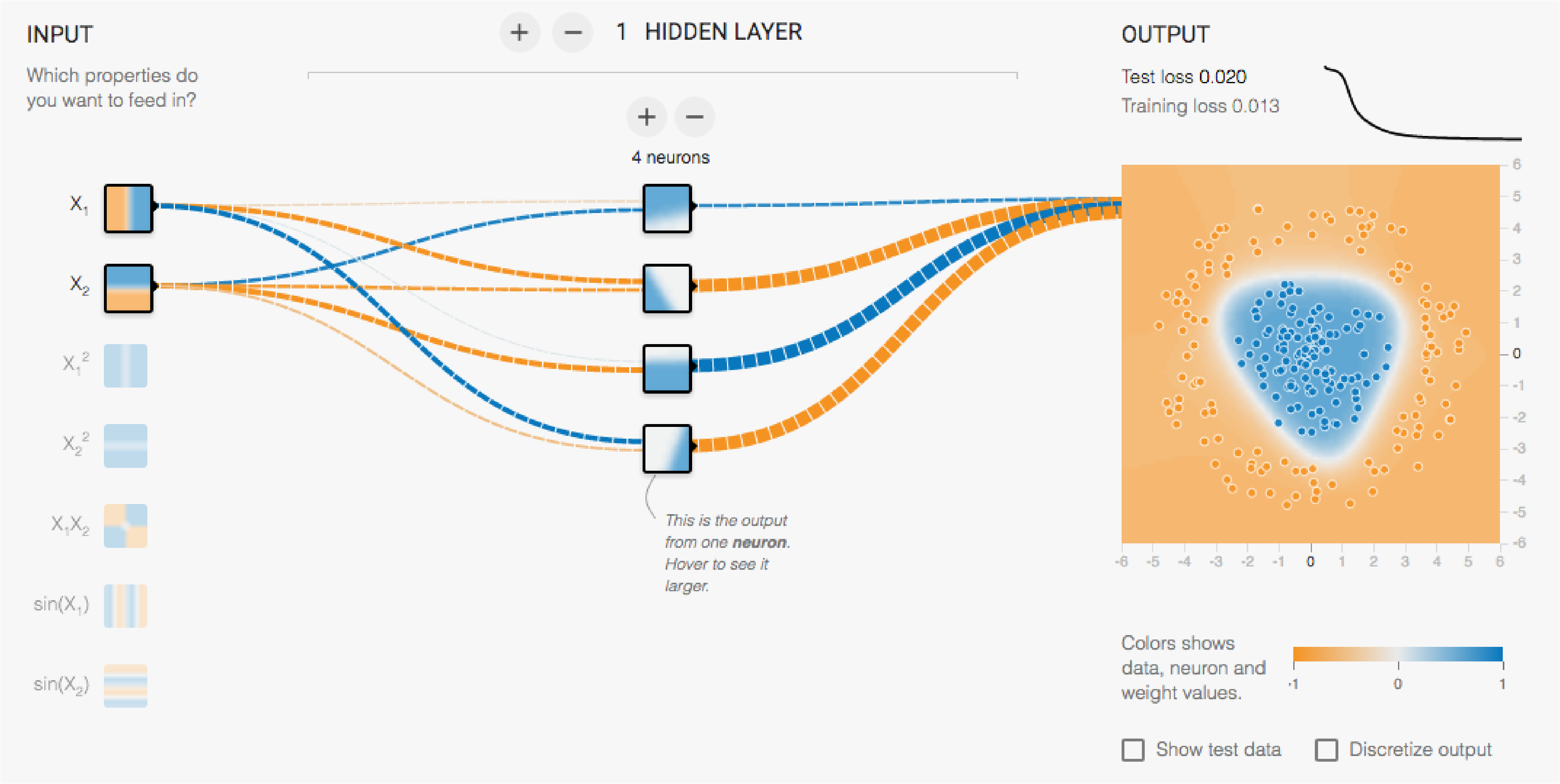
Loffe ,Szegedy, Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift, ArXiv 2015
R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, ICML 2013

Solutions & more

- Gated Recurrent Units (GRUs)
- Long Short-Term Memory (LSTMs)
- Residual/skip connections
- RNN VAE
- Bidirectional

Thanks

Multi-Layer Network Demo



http://playground.tensorflow.org/

How do error signals backpropagate in brains?