

Introduction to Deep Learning in NLP

2/2565: FRA501 Introduction to Natural Language Processing with Deep learning
Week 02

Paisit Khanarsa, Ph.D.

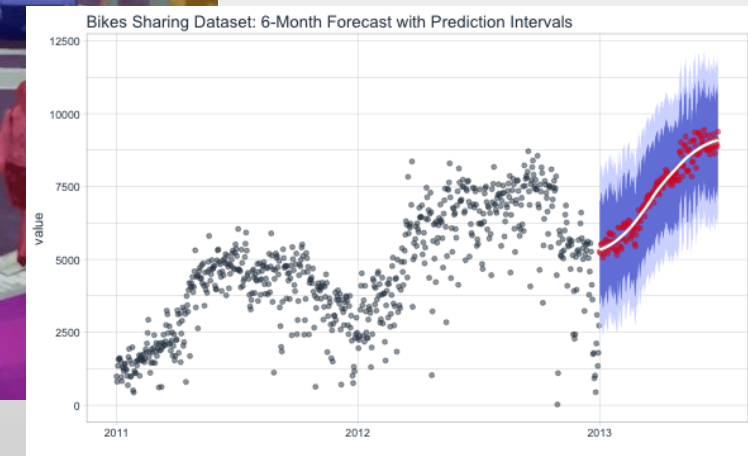
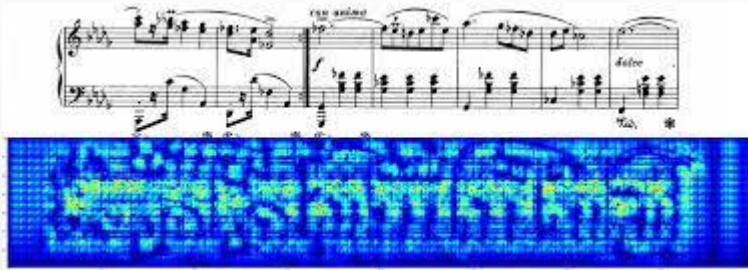
Institute of **Field Robotics** (FIBO), King Mongkut's University of Technology Thonburi

Neural Networks

Deep learning = Deep neural networks = Neural networks

Why deep learning?

- Greatly improved performance in many tasks (Computer Vision, Robotics, Time Series, NLP, etc.)
- Surpassed human performance in many tasks.



Deep learning in NLP

- Easy tasks

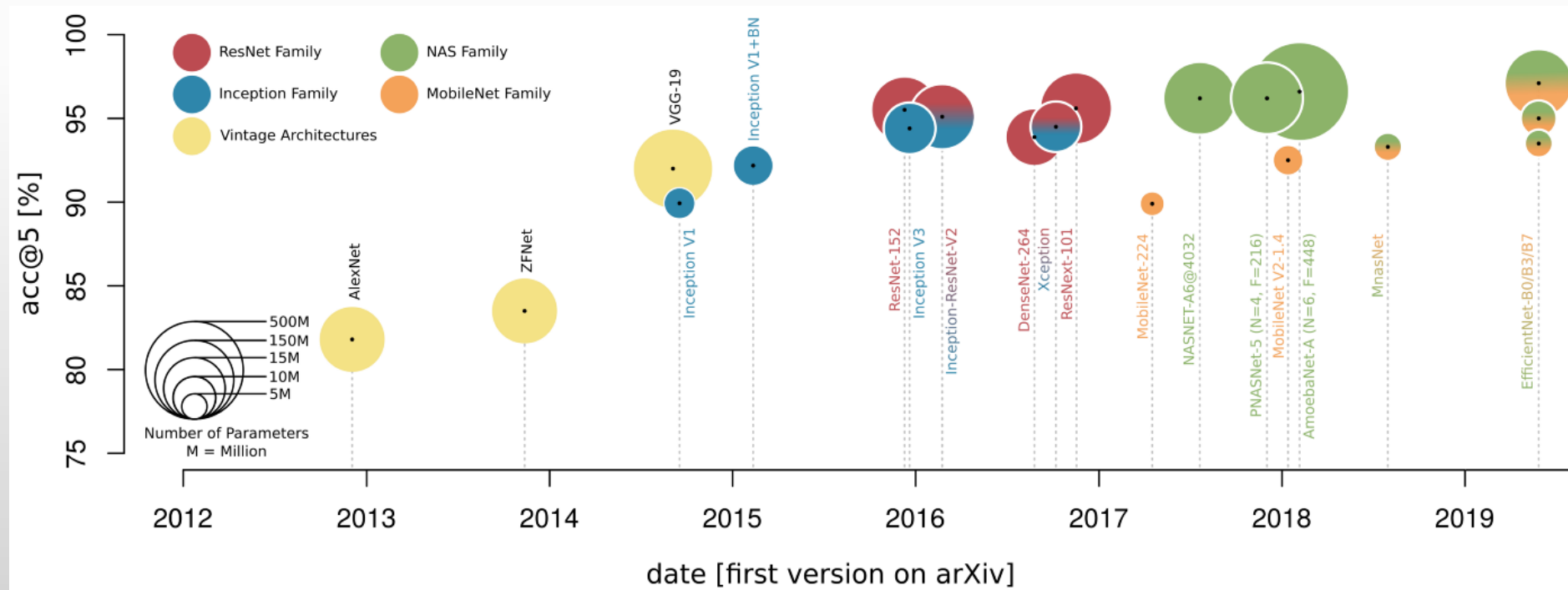
	Traditional ML	Deep learning
Sentiment	72%	76%
Topic classification	67%	70%
PoS	96%	97%

- Hard tasks

	Traditional ML	Deep learning
QA	51%	90%
Images to text or text to image	?	Done

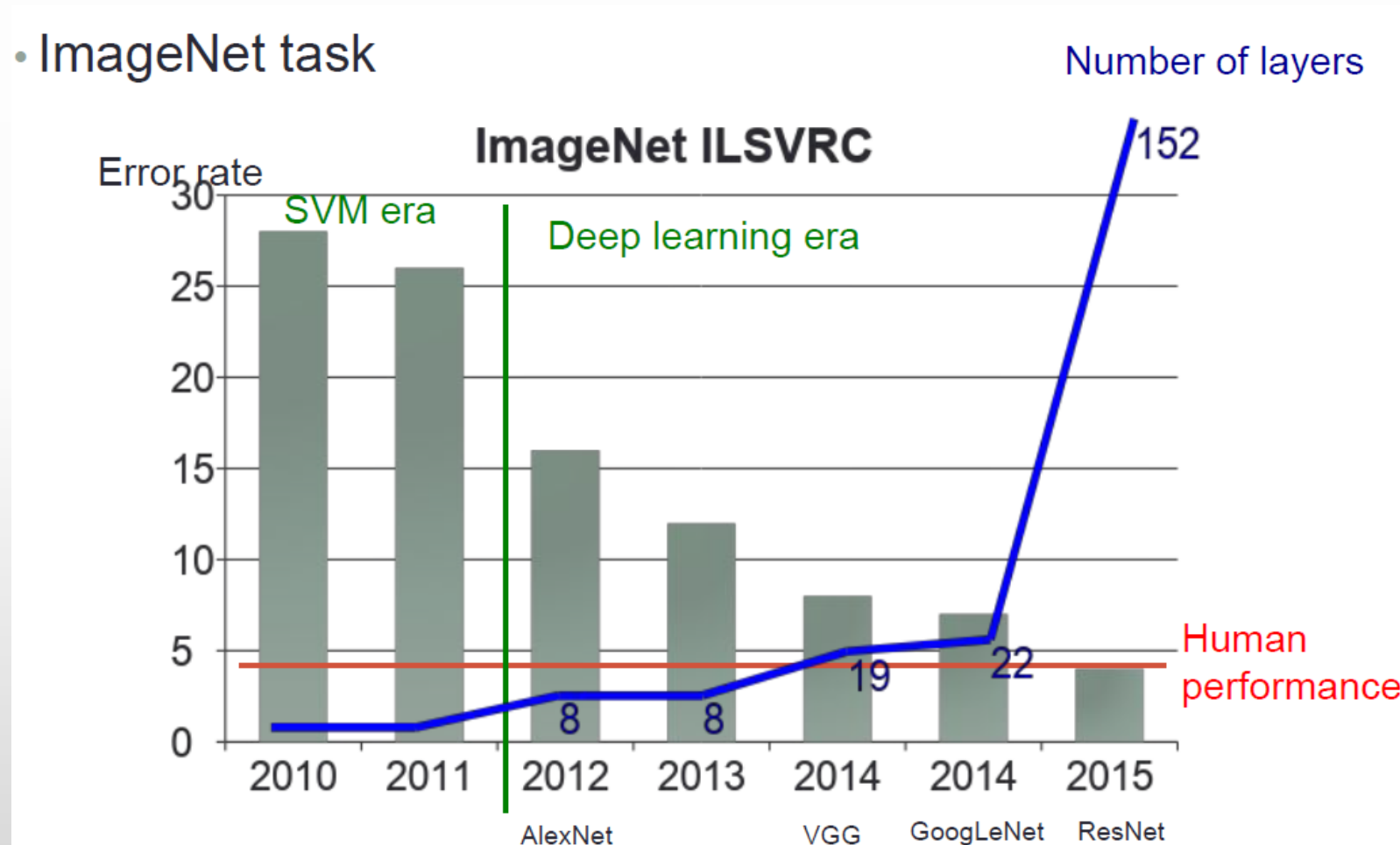
Why deep learning? (cont.)

- **Big data** - Deep neural networks have the ability to utilize large amounts of data more effectively in comparison to other types of models.
- **Graphics processing unit (GPU)** - "Allows for the possibility of training larger models."
- **Deep models** - "A larger model makes it simpler to avoid bad local minimums during training."

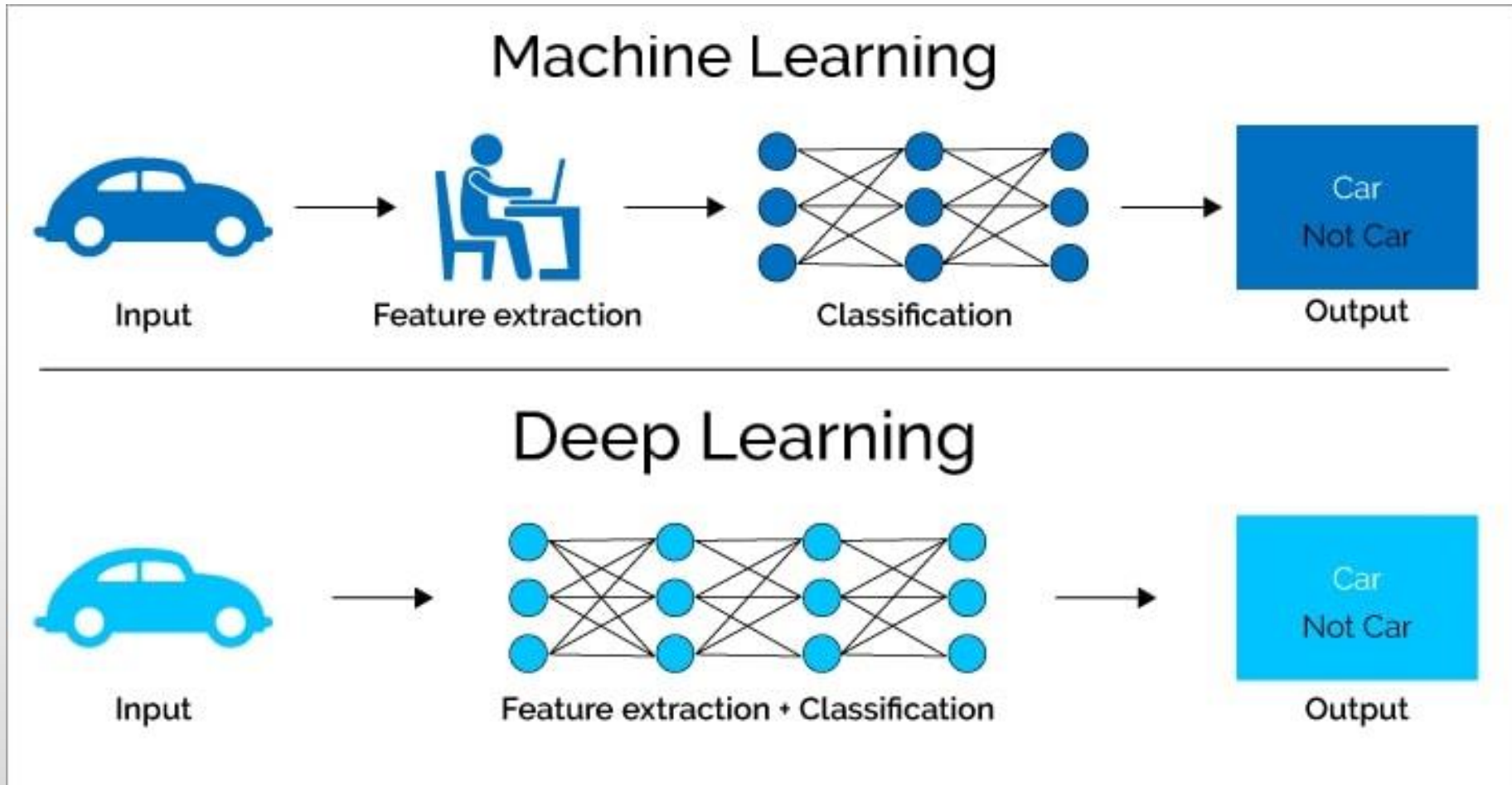


Why deep learning? (cont.)

- ImageNet task



Machine learning vs Deep learning

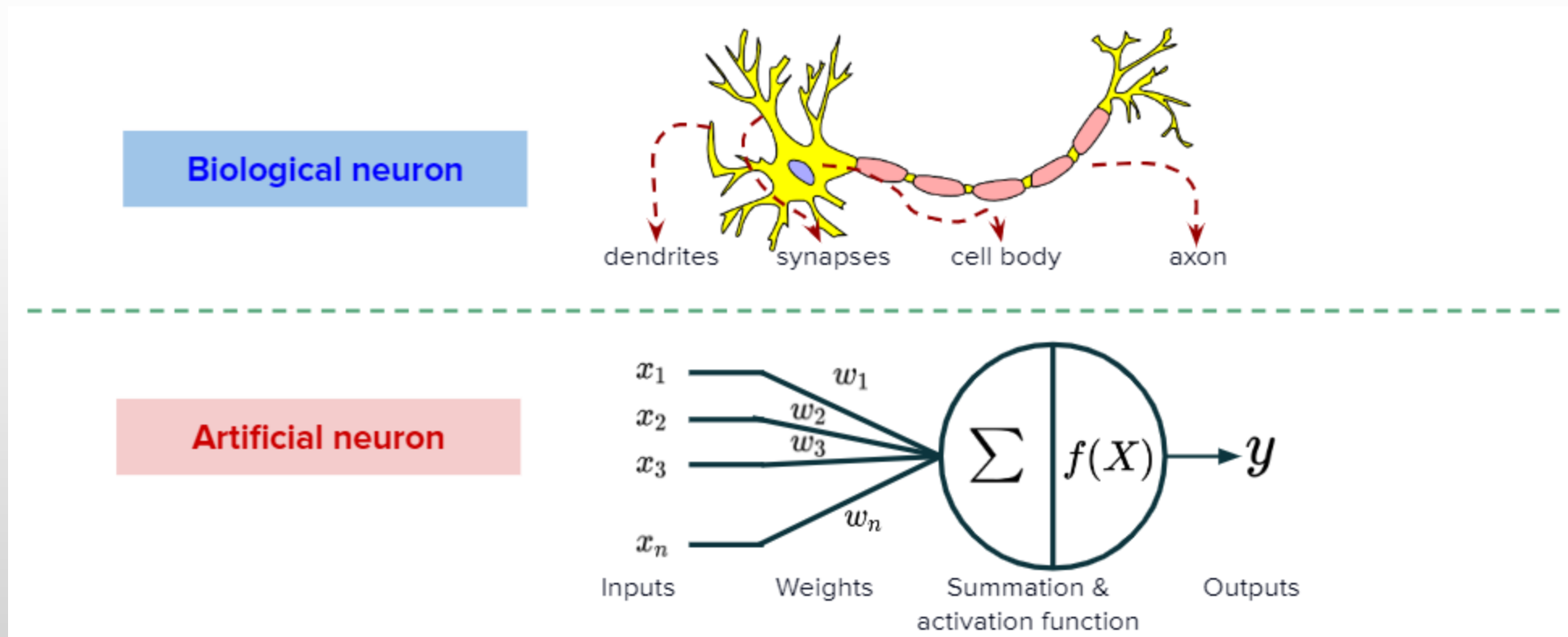


Basic knowledge of Neural Networks

- Fully connected networks
 - Neuron
 - Non-linearity
 - Softmax
- Deep neural network training
 - Loss function
 - Stochastic gradient descent (SGD) and backpropagation
 - Learning rate & Optimizer
 - Overfitting
- CNN, RNN, LSTM, GRU
- Deep learning in NLP

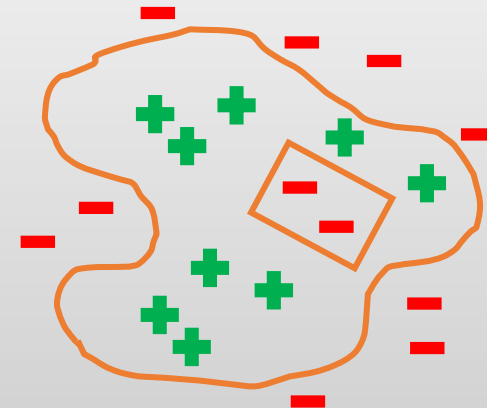
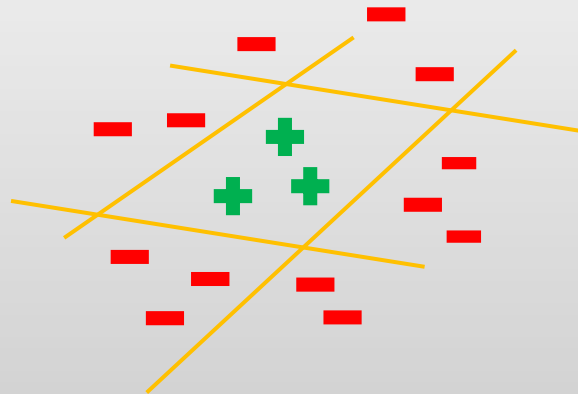
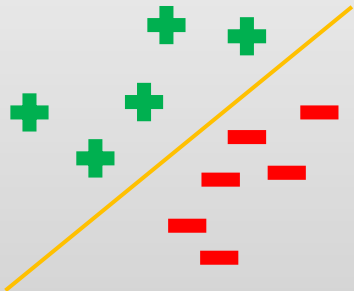
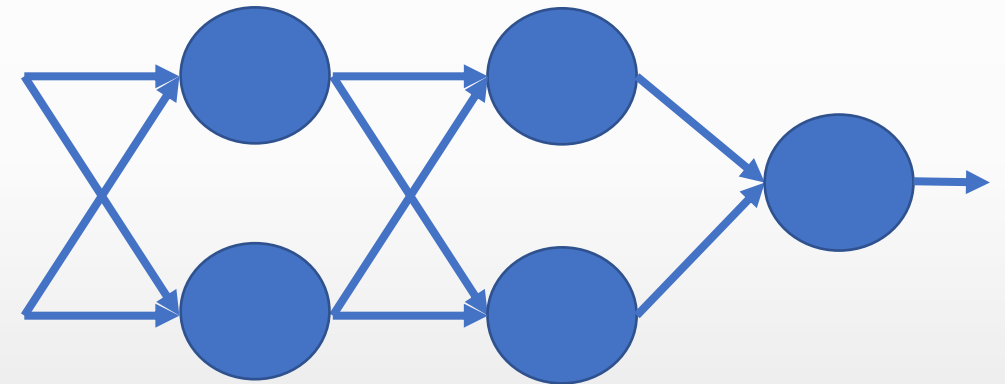
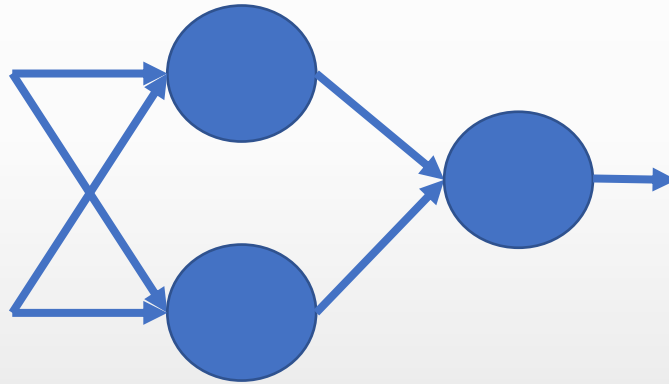
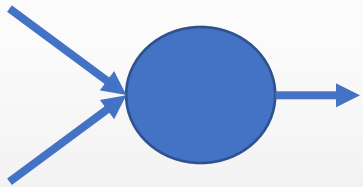
Fully connected networks

- Neural networks are sets of algorithm, modeled loosely after the human brain, that are designed to recognize patterns.

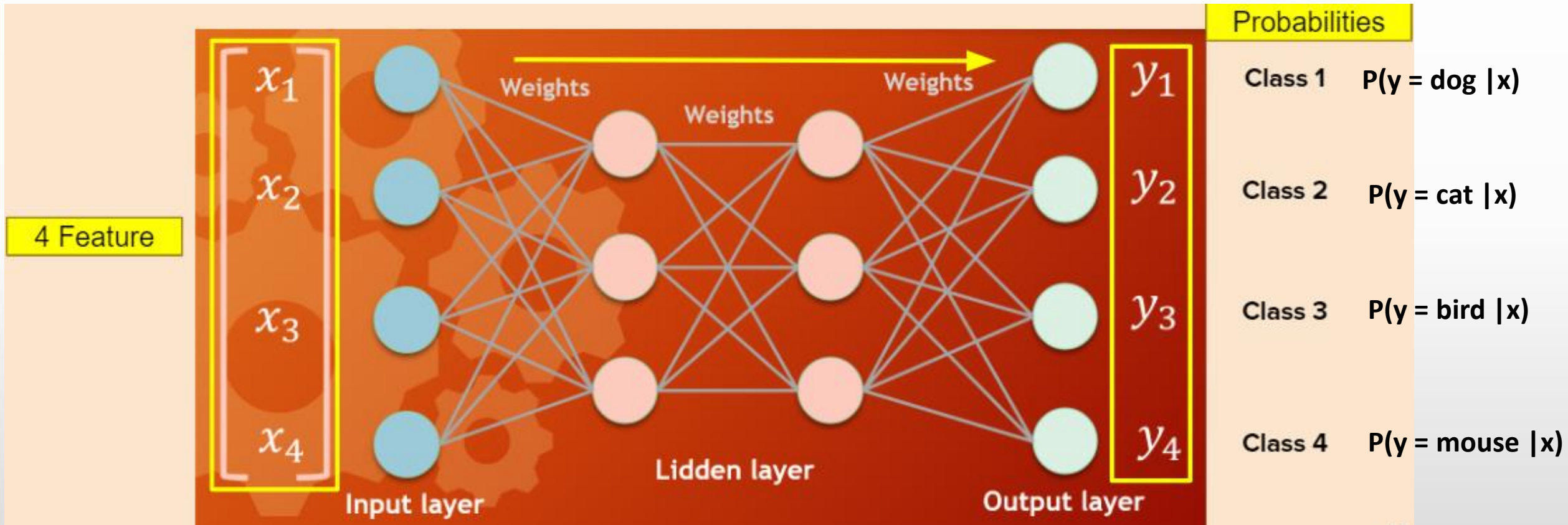


Fully connected networks (cont.)

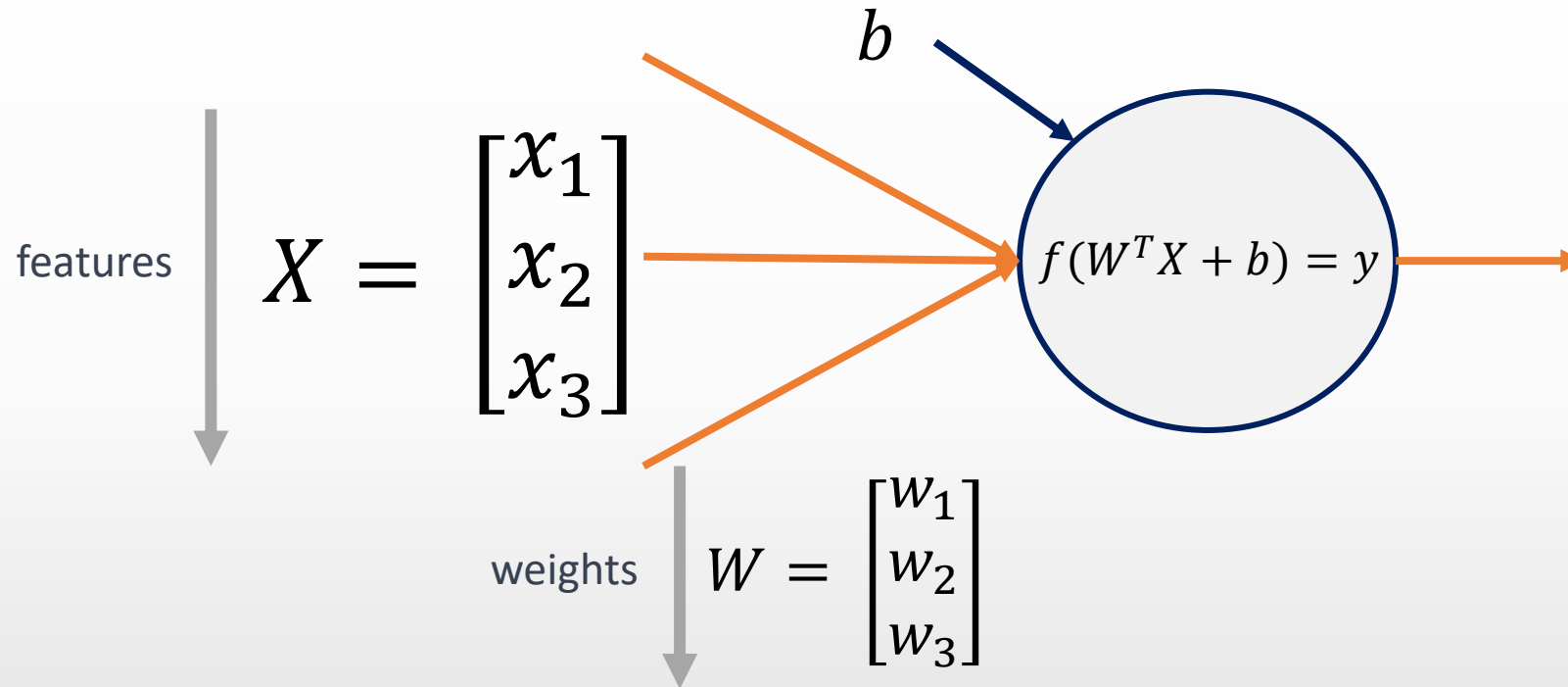
- Each neuron in the model divides the feature space using a hyperplane. As more neurons are added, the decision boundaries become more intricate.



Fully connected networks (cont.)

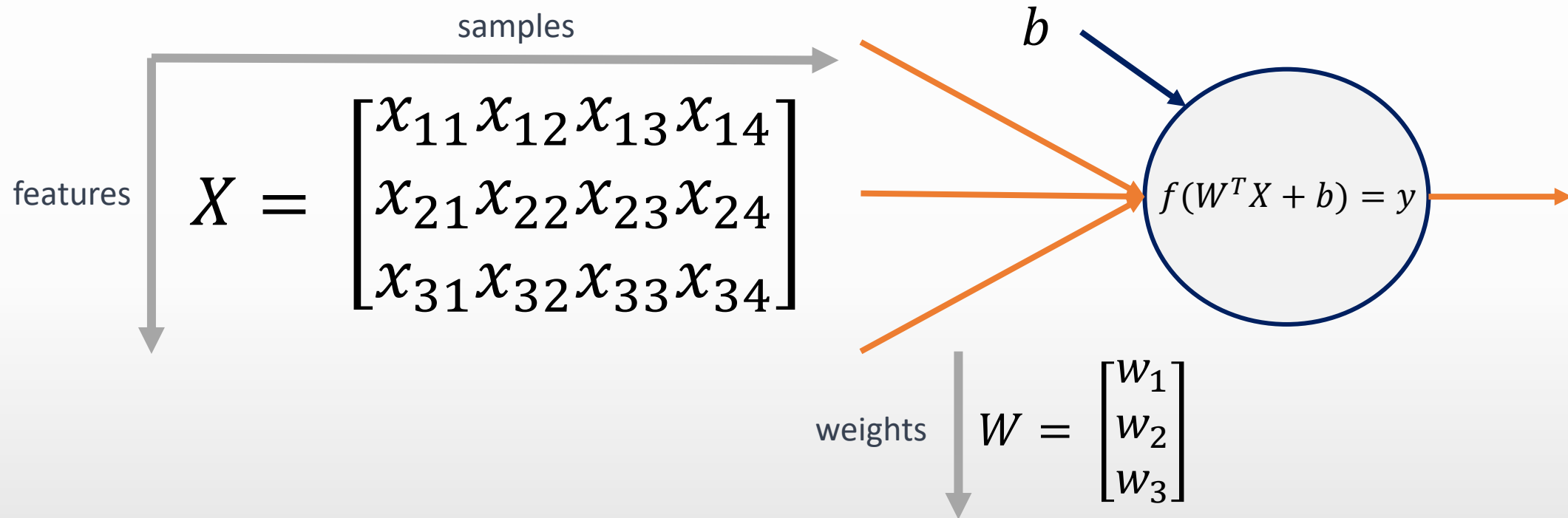


Fully connected networks and Matrices



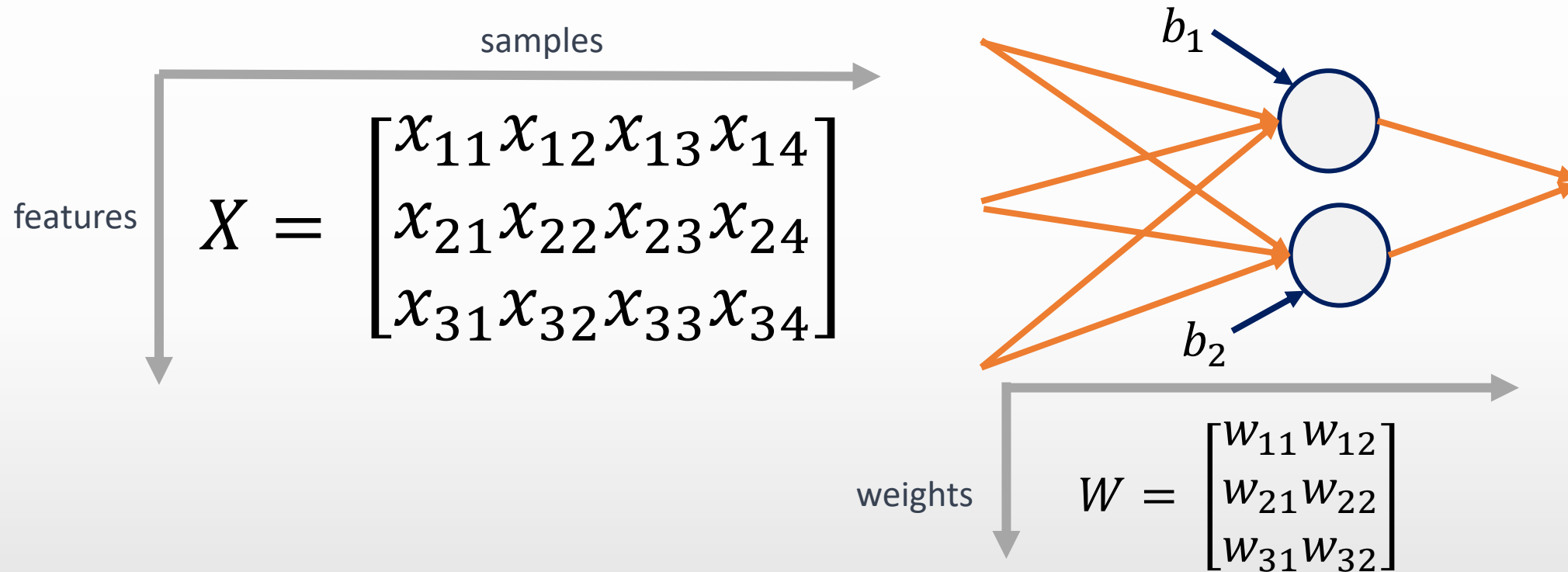
$$W^T X + b = [w_1 \ w_2 \ w_3]_{1 \times 3} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{3 \times 1} + b$$

Fully connected networks and Matrices (cont.)



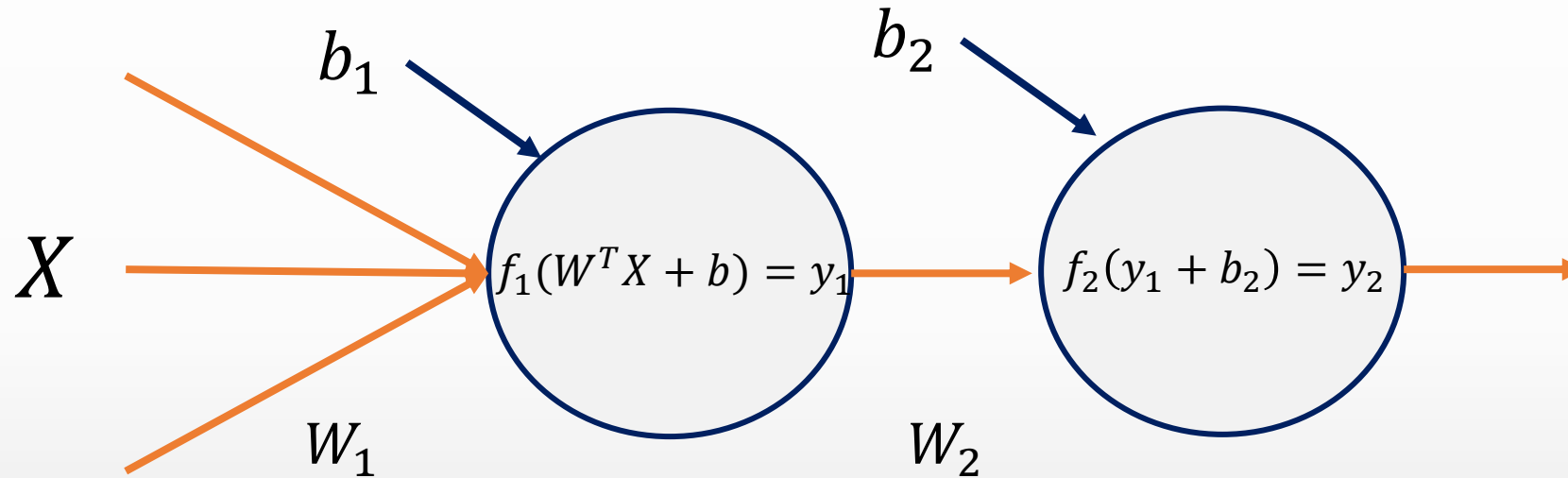
$$W^T X + b = [w_1 \ w_2 \ w_3]_{1 \times 3} \cdot \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix}_{3 \times 4} + [b \ b \ b \ b]$$

Fully connected networks and Matrices (cont.)



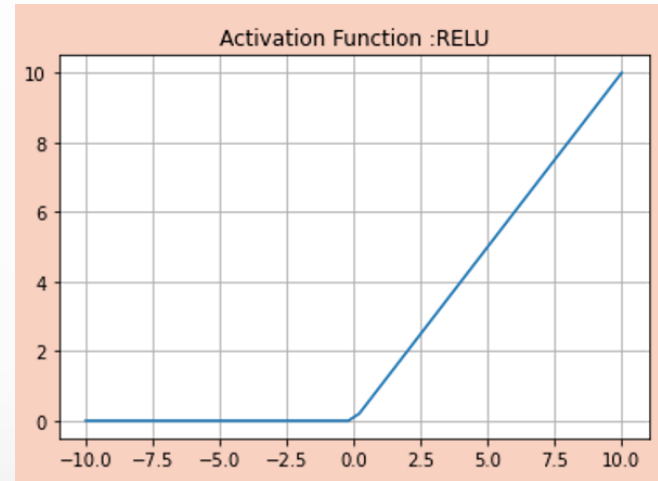
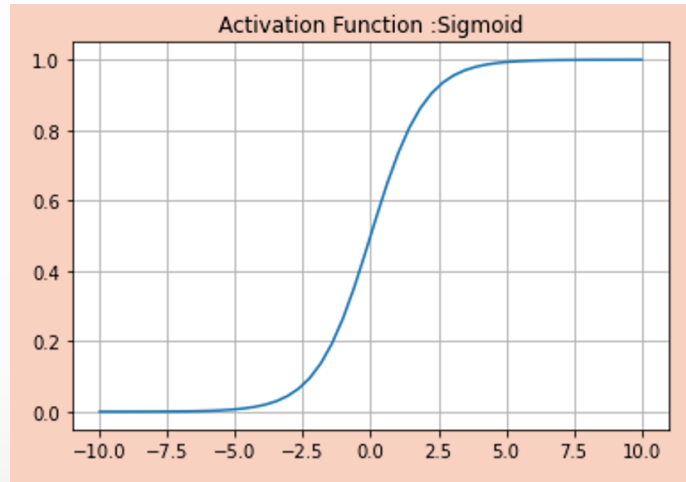
$$W^T X + b = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}_{2 \times 3} \cdot \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix}_{3 \times 4} + \begin{bmatrix} b_1 & b_1 & b_1 & b_1 \\ b_2 & b_2 & b_2 & b_2 \end{bmatrix}$$

Fully connected networks and Non-linearity

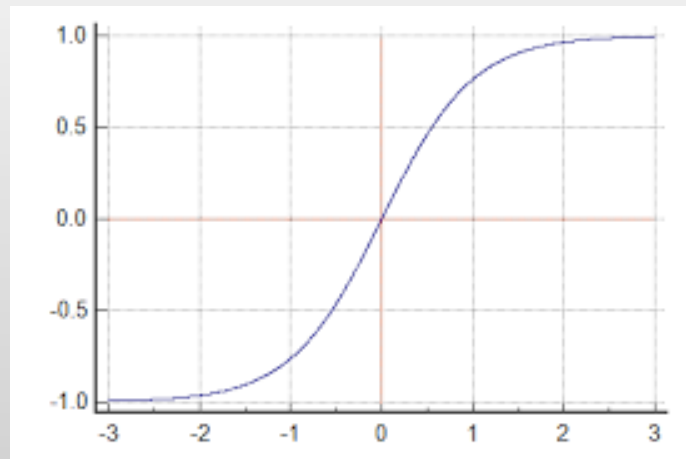


$$f_2(W_2^T f_1(W_1^T X + b_1) + b_2) = y_2$$

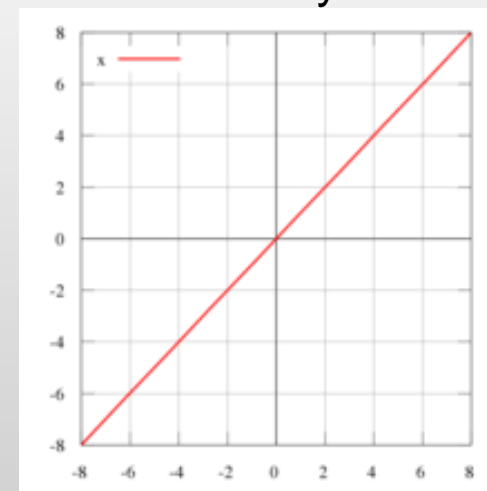
Fully connected networks and Non-linearity (cont.)



tanh



identity



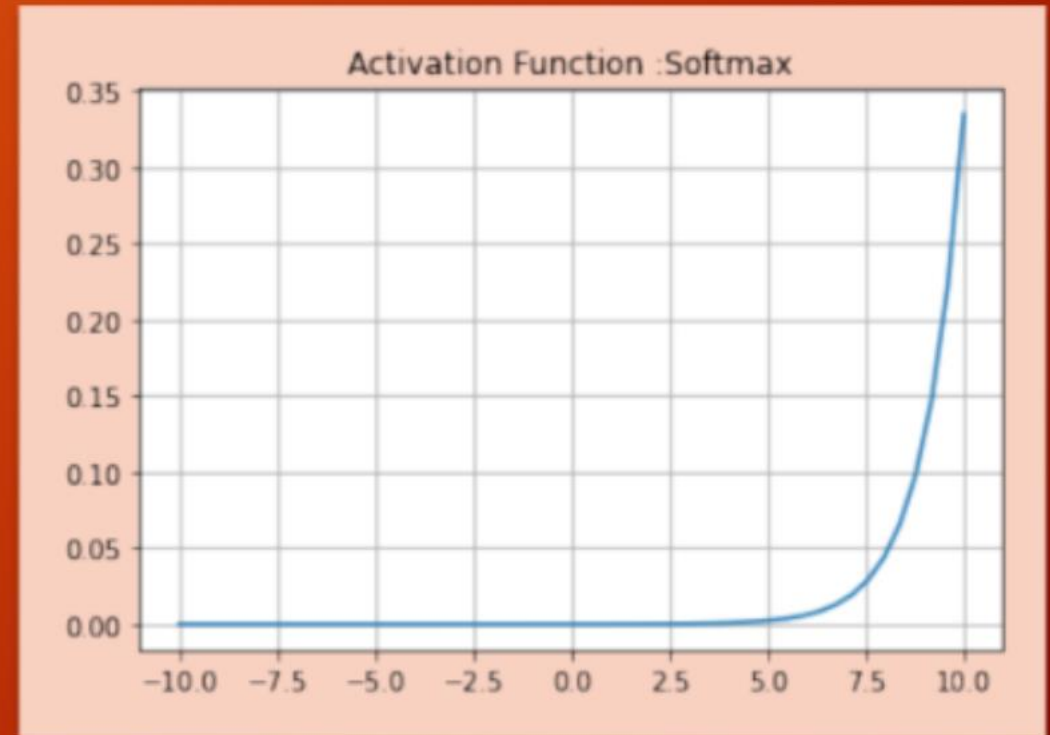
Softmax in Output Layer

$$\text{Softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^n e^{z_k}}$$

where $j = 1, 2, \dots, n$

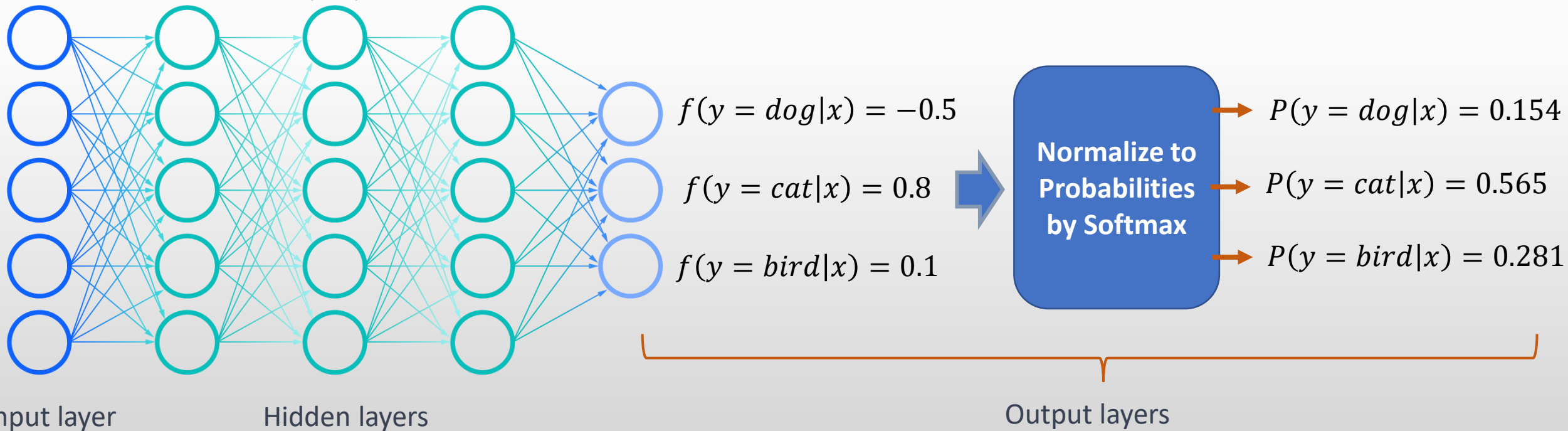
and n is the number of nodes in output layer

- Domain: $z_j \in \mathbb{R}$ (Real Number)
- Range: $\text{Softmax}(z_j) \in (0, 1)$
- $z \rightarrow -\infty$ then $S(z) \rightarrow 0$
- $z \rightarrow \infty$ then $S(z) \rightarrow 1$

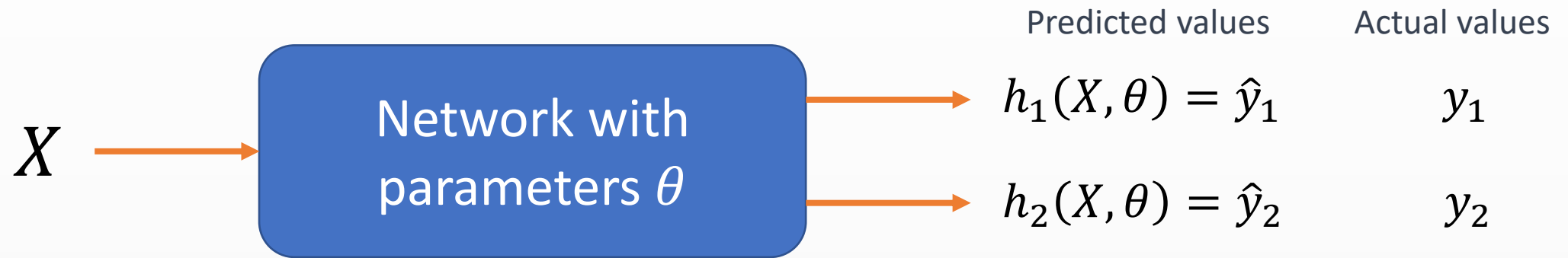


Softmax in Output Layer (cont.)

$$P(y = j|x) = \frac{e^{f(y=j|x)}}{\sum_y e^{f(y|x)}}$$



Loss function (Objective function)



- Cross entropy (Classification problems)
- Sum of square errors (Regression problems)

Loss function (cont.)

- Cross entropy (Classification problems)

$$E(W) = -\frac{1}{m} \sum_{c=1}^k \sum_{i=1}^m [y_c^i \log(h_c(x^i)) + (1 - y_c^i) \log(1 - h_c(x^i))]$$

where m is the number of samples and k is the number of outputs or classes

- Sum of square errors (Regression problems)

$$MSE = E(W) = \frac{1}{m} \sum_{i=1}^m [y^i - h(x^i)]^2$$

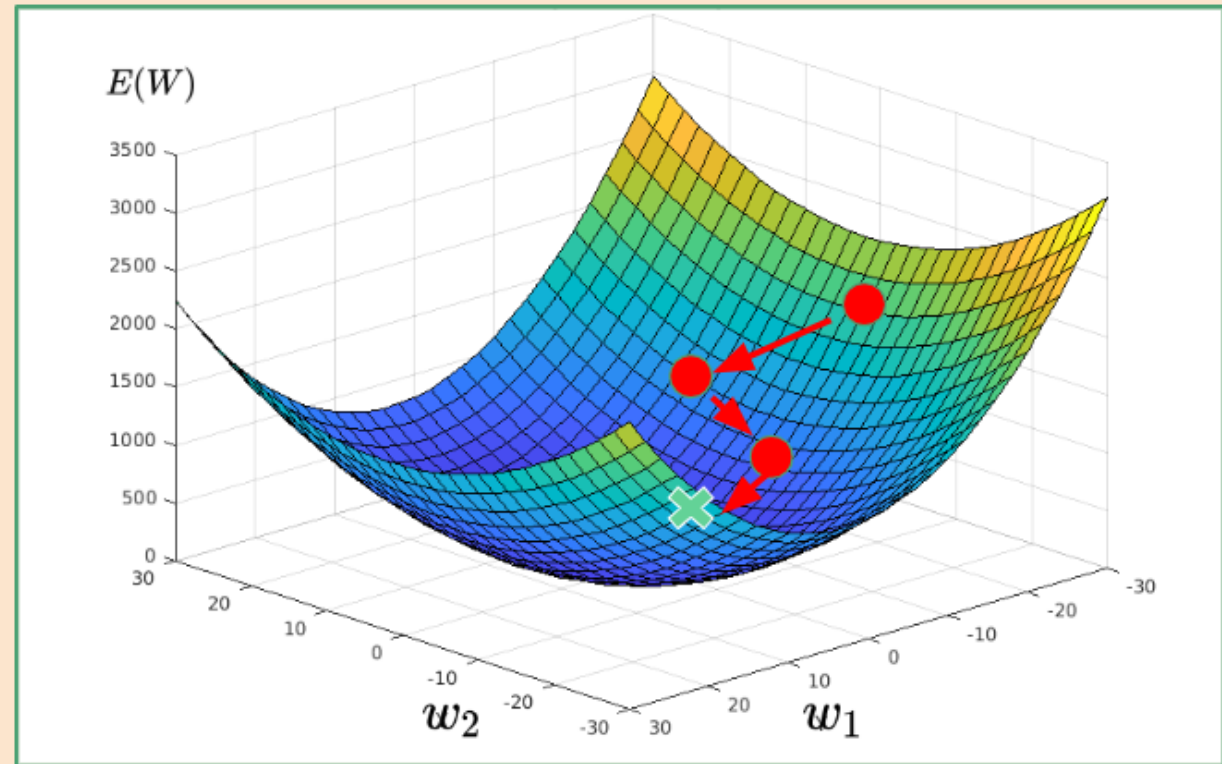
where m is the number of samples

Minimize lost function using Gradient descent

1. Randomly initialize w_1 and w_2 .
2. Calculate the gradient
($\nabla E(w_1, w_2)$)
3. Update the algorithm with
the following formula:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Rightarrow \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \alpha \nabla E(w_1, w_2)$$

4. Monitor the cost function.
Cost should be lower any
time you update weights.
5. When cost function is low
enough, stop updating



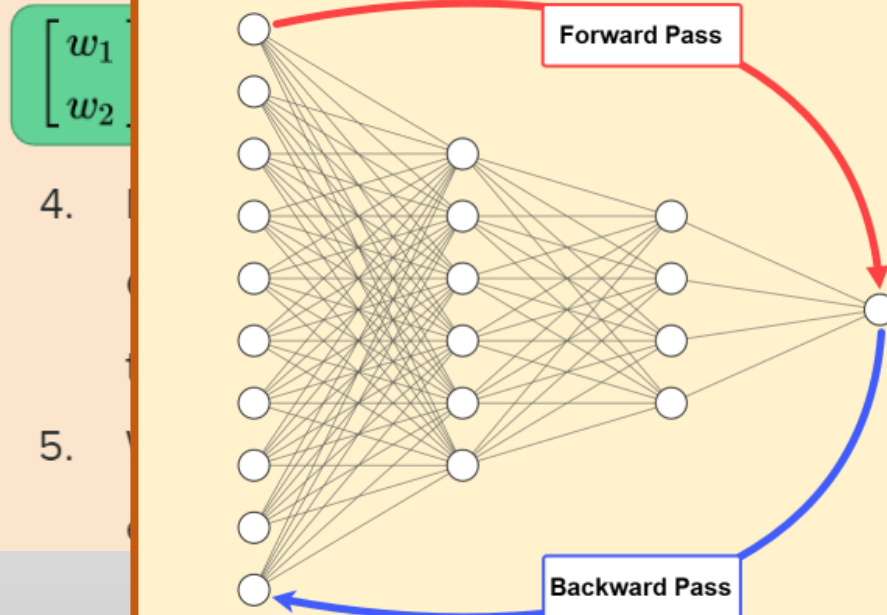
Minimize lost function using Gradient descent

Use Back propagation to compute gradient

Forward pass: pass the value of the input until the end of the network

Backward pass: Compute the gradient starting from the end and passing down gradients using chain rule

the following formula:

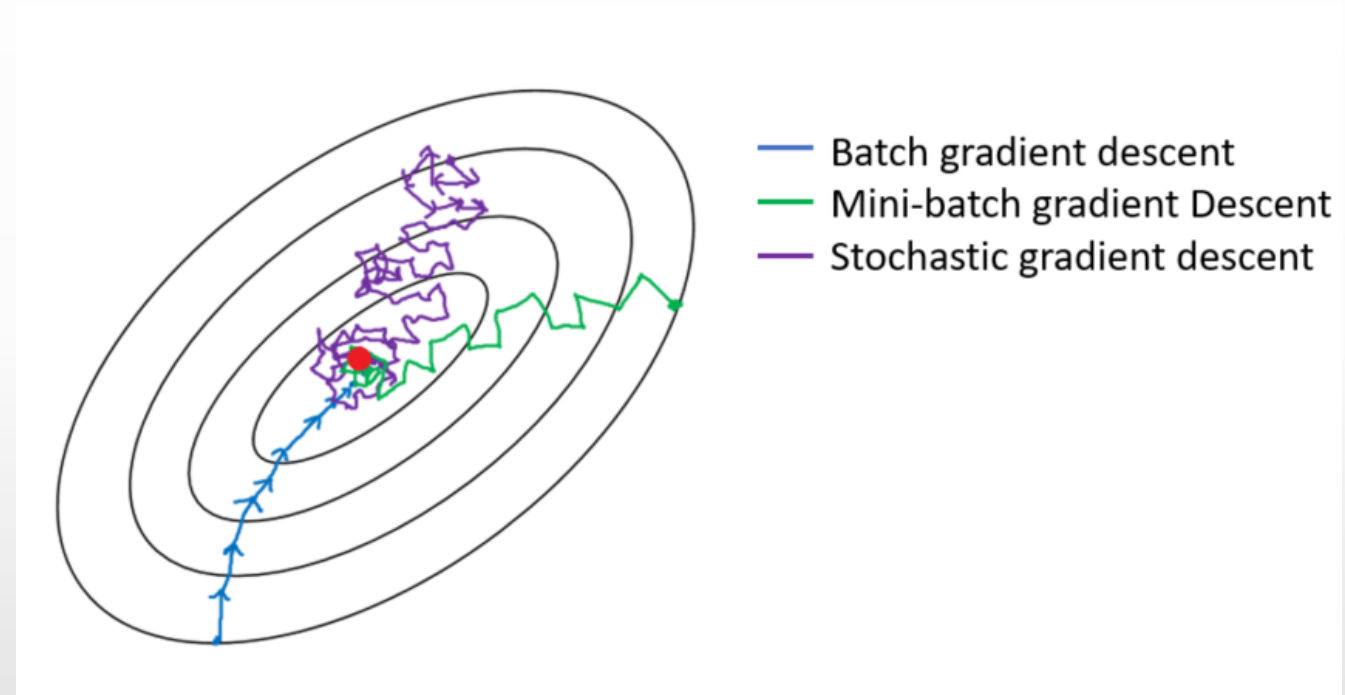


The screenshot shows a YouTube video player with the title 'Backpropagation calculus | Chapter 4, Deep learning'. The video content includes mathematical formulas for the backward pass, such as $\frac{\partial C_0}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L-1} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}}$ and $z_j^{(L)} = \dots + w_{jk}^{(L)} a_k^{(L-1)} + \dots$. It also shows a diagram of a neural network with numerical values for the nodes. The video player interface includes a progress bar, volume control, and a list of recommended videos on the right.

<https://www.youtube.com/watch?v=tIeHLnjs5U8>

Stochastic gradient descent (SGD)

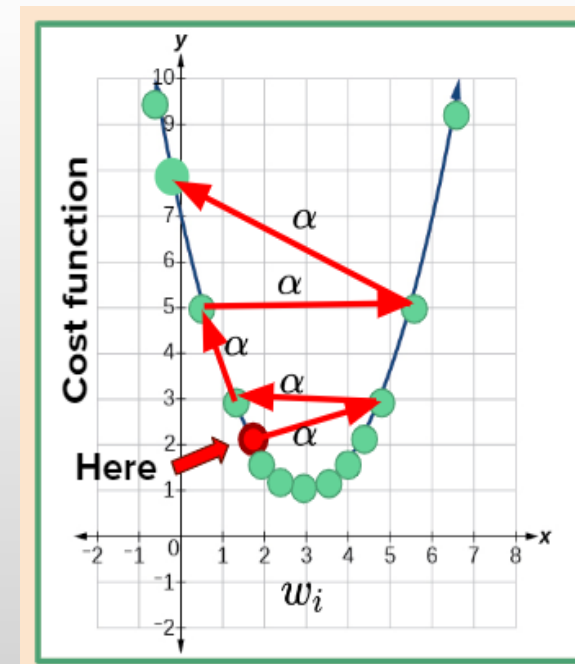
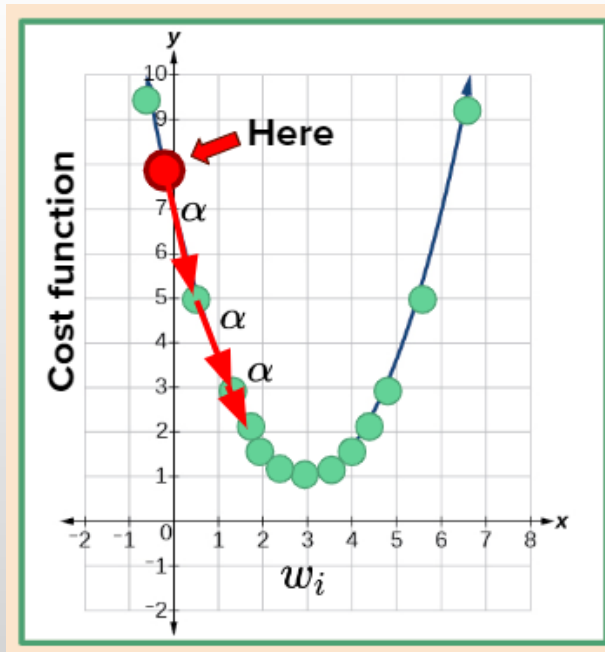
- We have one million training examples
 - Gradient descent compute the loss function of all samples, then decide the direction of the descent
 - ---> Take too long!!!
- SGD compute the loss function on a subset of samples
 - ---> The subset should not be biased and properly randomized to ensure no correlation between samples.
 - The subset is called a mini-batch
 - Size of the mini-batch determines the speed and accuracy



<https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4>

Learning rate (α) & Optimizer

- Learning rate is usually between 0 and 1.
- Large alpha: big step downhill.
- Small alpha: small step.
- You do not want too big or too small alpha

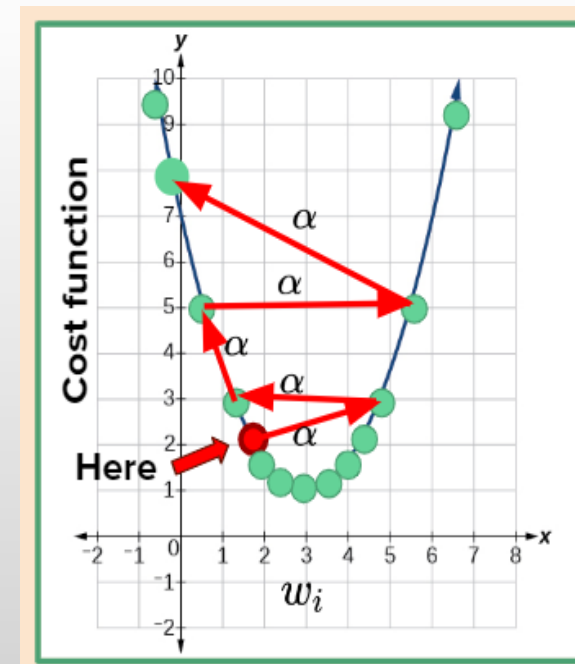
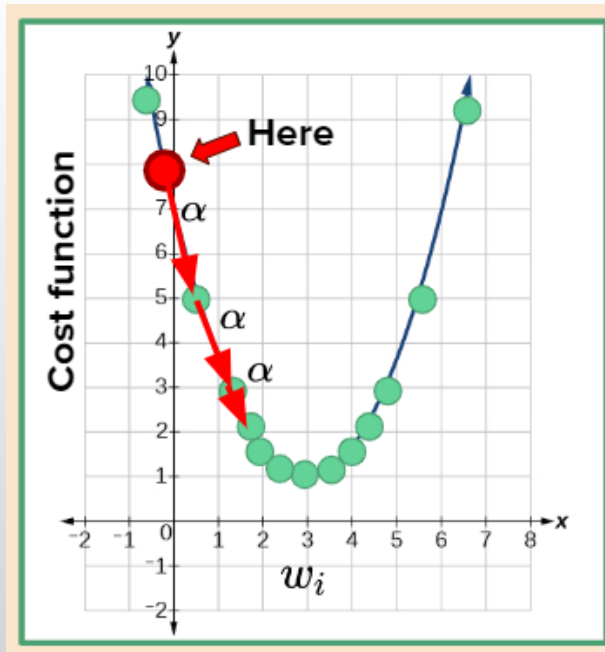


Learning rate (α) & Optimizer (cont.)

- Learning rate is usually between 0 and 1.
- Large alpha: big step downhill.
- Small alpha: small step.
- You do not want too big or too small alpha

Learning rate scheduling

Usually starts with a large learning rate then gets smaller later



Learning rate (α) & Optimizer (cont.)

- Besides learning rate scheduling (coarse grain) we can do finer (and automatic) control of the learning rate
- People find simple SGD with momentum and decay to perform better (with proper tuning)

- ADAM

Most popular for its ease of use

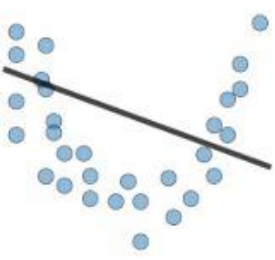


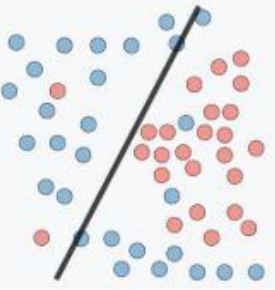
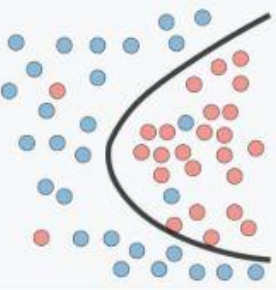
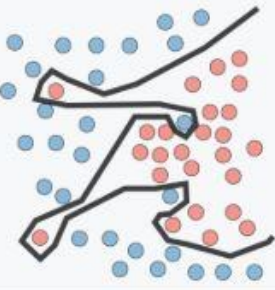

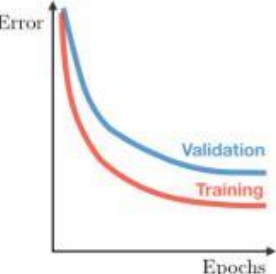
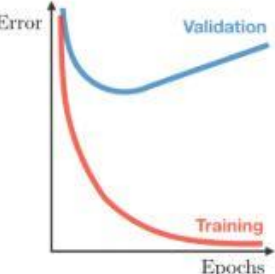
- RMSprop

Faster than SGD but slower than ADAM

More stable than ADAM

Overfitting

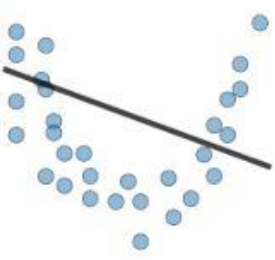


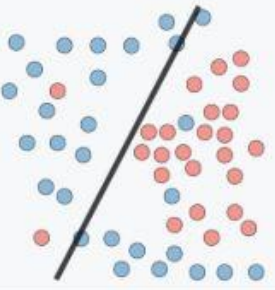
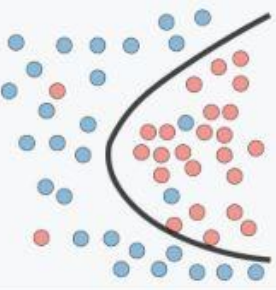
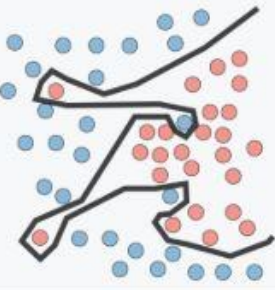

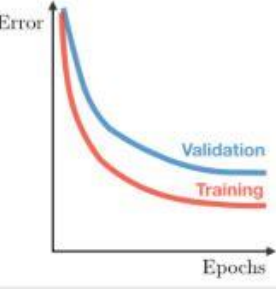
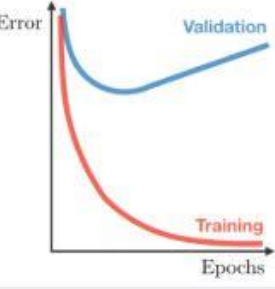
- The training loss will always go down
- But it overfits
- Need to monitor performance on a held-out set
- Stop or decrease learning rate when overfit happens

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none">• High training error• Training error close to test error• High bias	<ul style="list-style-type: none">• Training error slightly lower than test error	<ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	<ul style="list-style-type: none">• Complexify model• Add more features• Train longer		<ul style="list-style-type: none">• Perform regularization• Get more data

Overfitting (cont.)

- The training loss will always go down
- But it overfits
- Need to monitor performance on a held-out set
- Stop or decrease learning rate when overfit happens

- Dropout
- Regularization
- Batch normalization

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none">• High training error• Training error close to test error• High bias	<ul style="list-style-type: none">• Training error slightly lower than test error	<ul style="list-style-type: none">• Very low training error• Training error much lower than test error• High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	<ul style="list-style-type: none">• Complexify model• Add more features• Train longer		<ul style="list-style-type: none">• Perform regularization• Get more data

Convolutional Neural Networks (CNNs)

How we identify things ?

Let's identify key features in each image category



Nose, Eyes,
Mouth



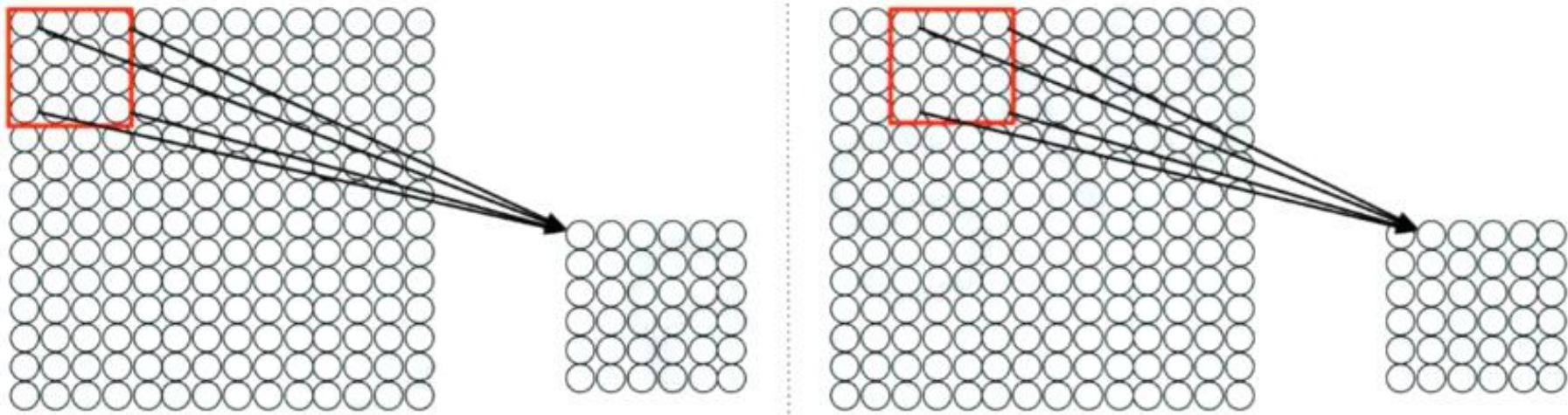
Wheels,
License Plate,
Headlight



Door, Windows,
Steps

Convolutional Neural Networks (CNNs) (cont.)

Using Spatial Structure

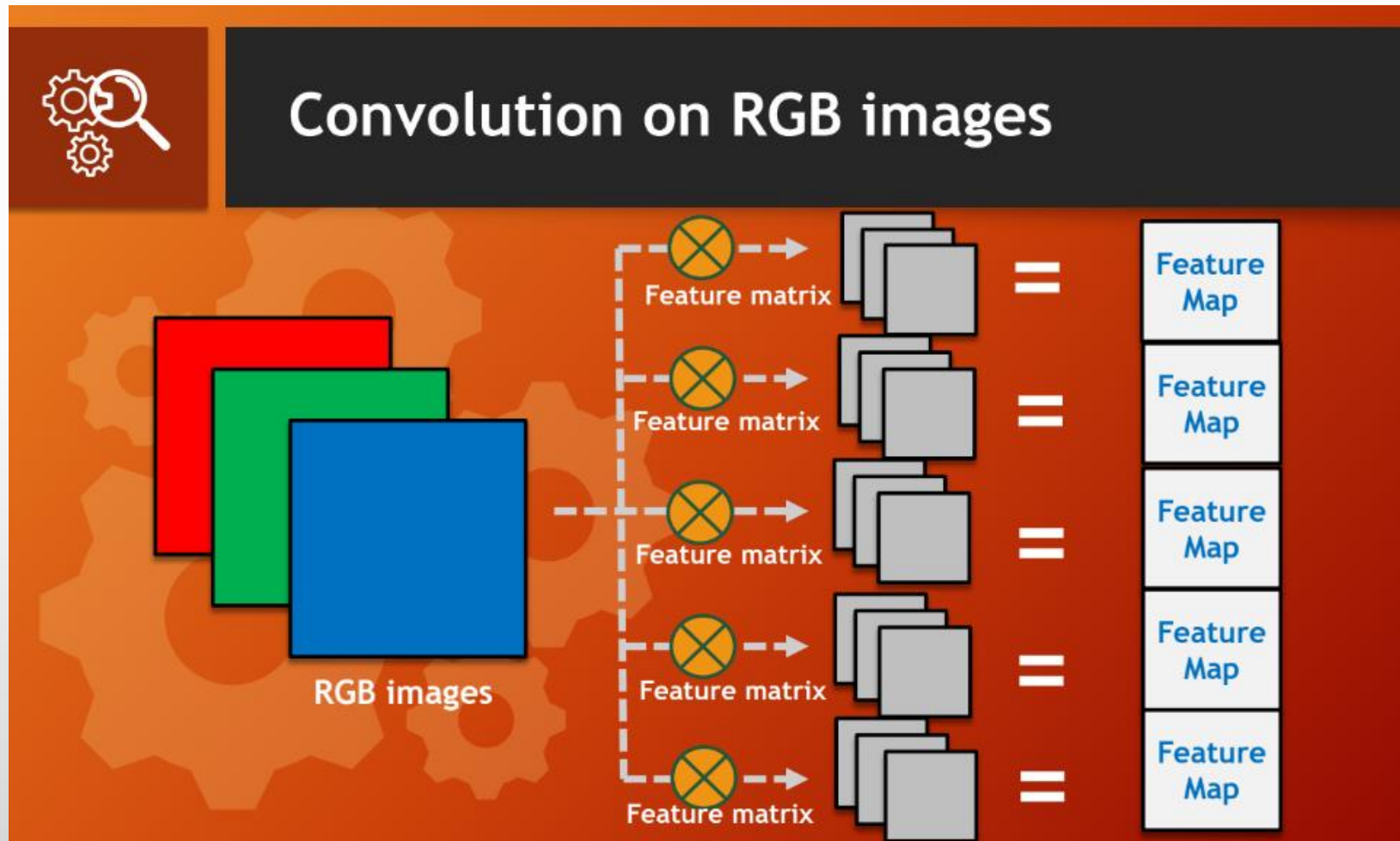


Connect patch in input layer to a single neuron in subsequent layer.

Use a sliding window to define connections.

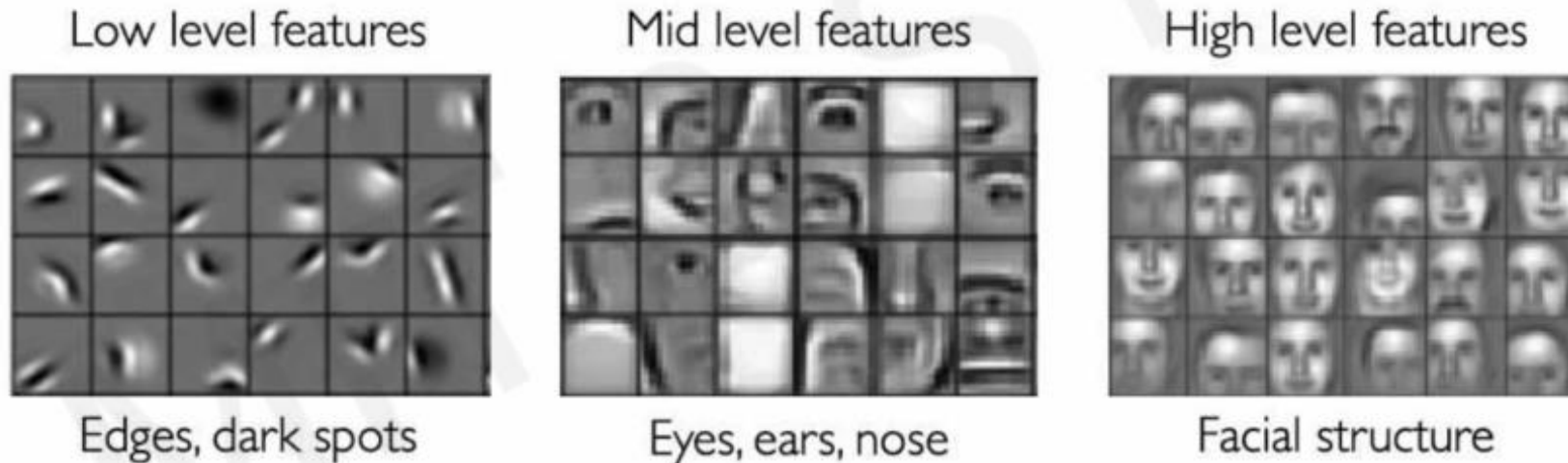
How can we **weight** the patch to detect particular features?

Convolutional Neural Networks (CNNs) (cont.)



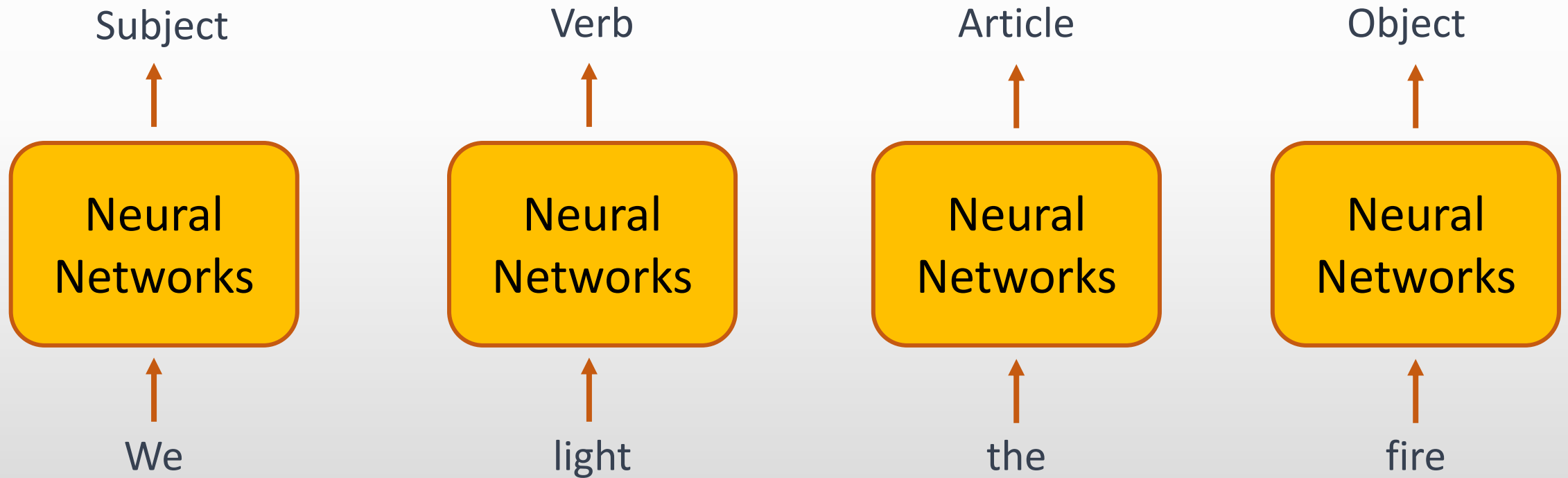
Convolutional Neural Networks (CNNs) (cont.)

Can we learn a hierarchy of features directly from the data instead of hand engineering?



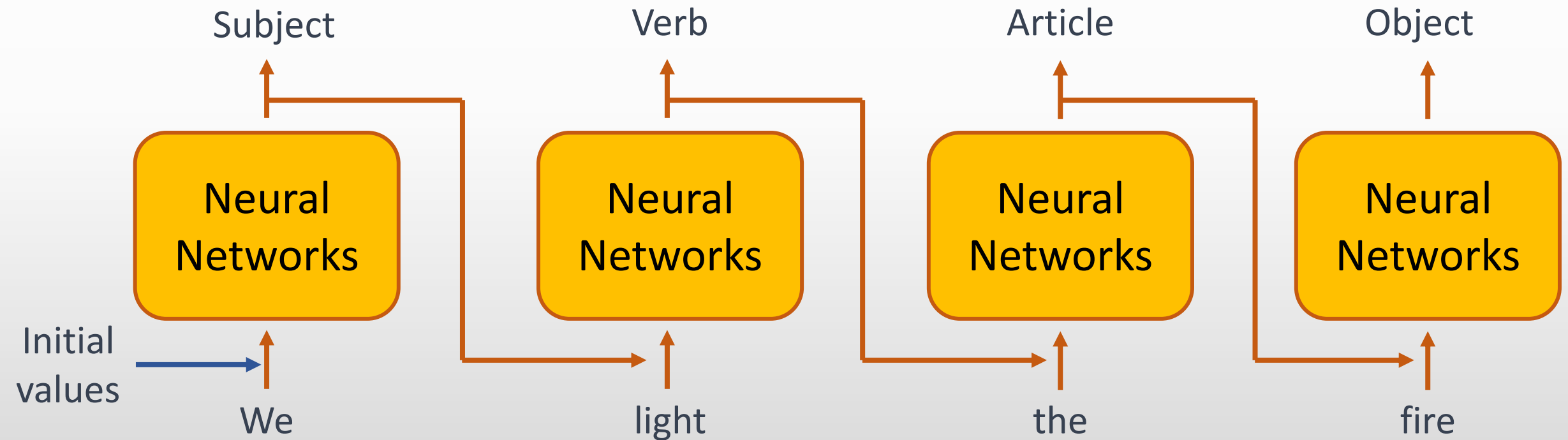
Recurrent Neural Networks (RNNs)

- Neural network concept



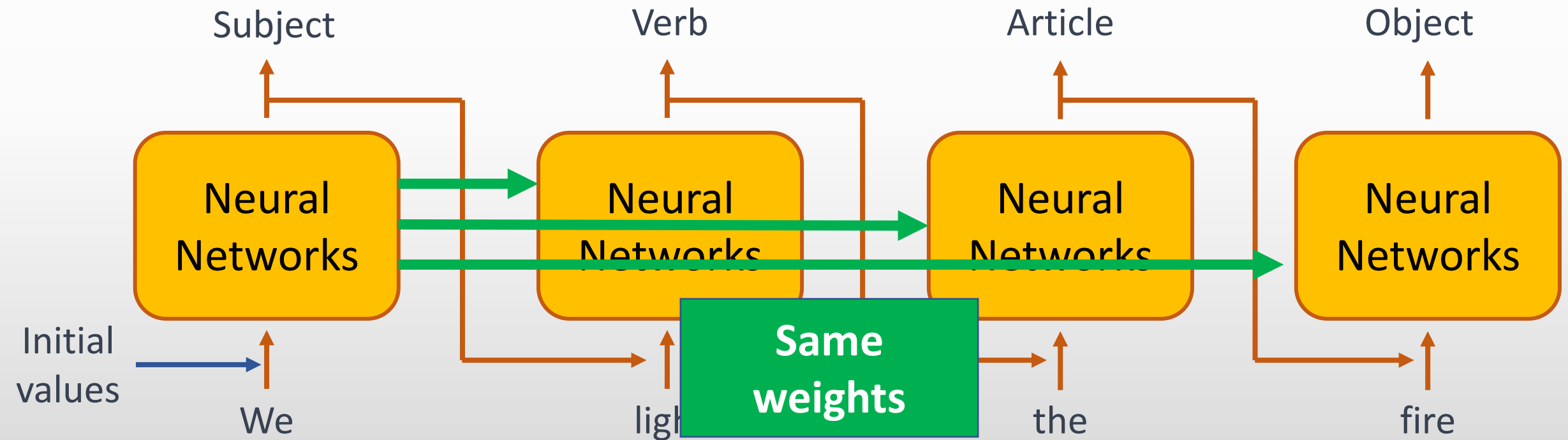
Recurrent Neural Networks (RNNs) (cont.)

- Recurrent Neural network concept

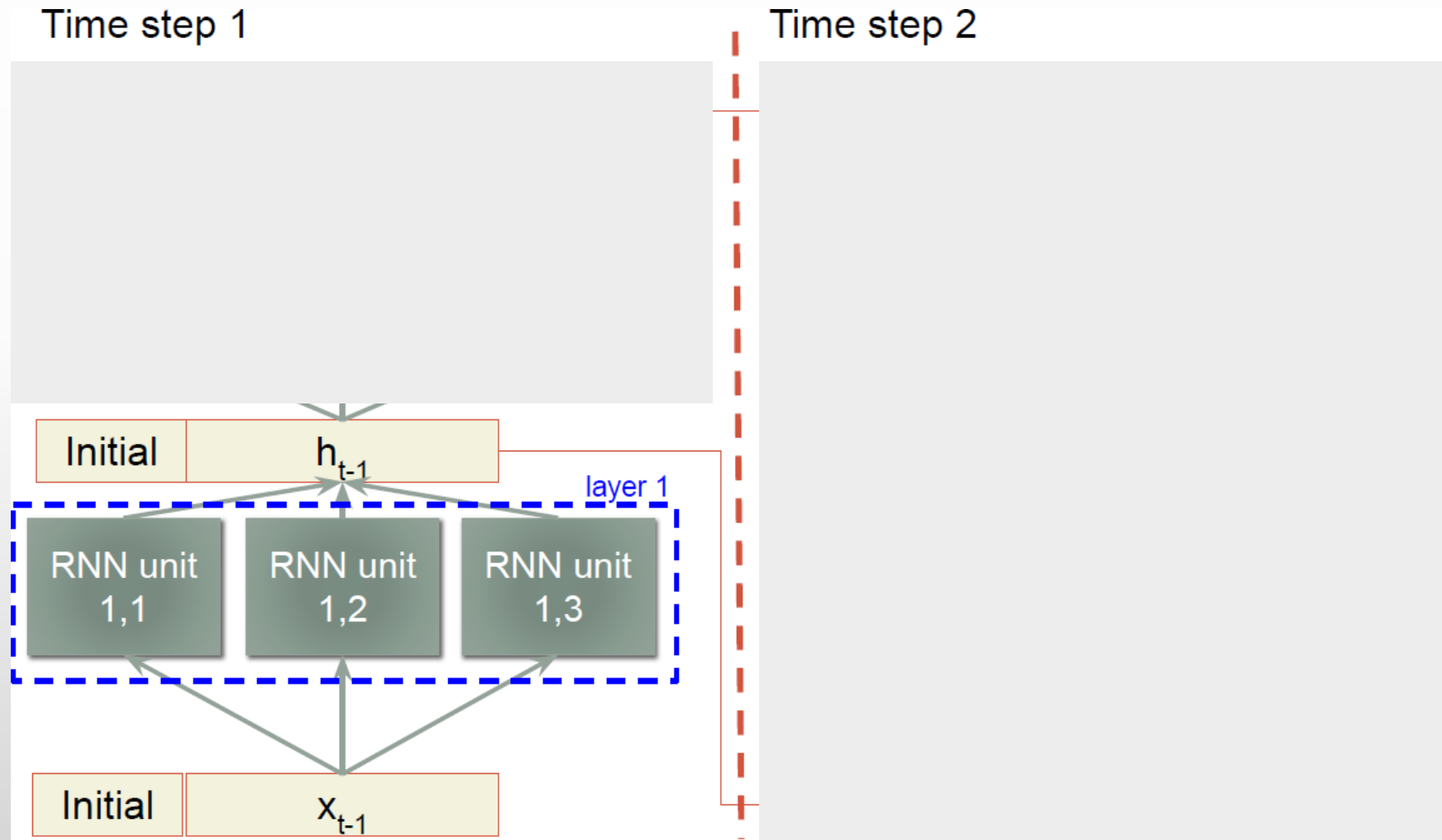


Recurrent Neural Networks (RNNs) (cont.)

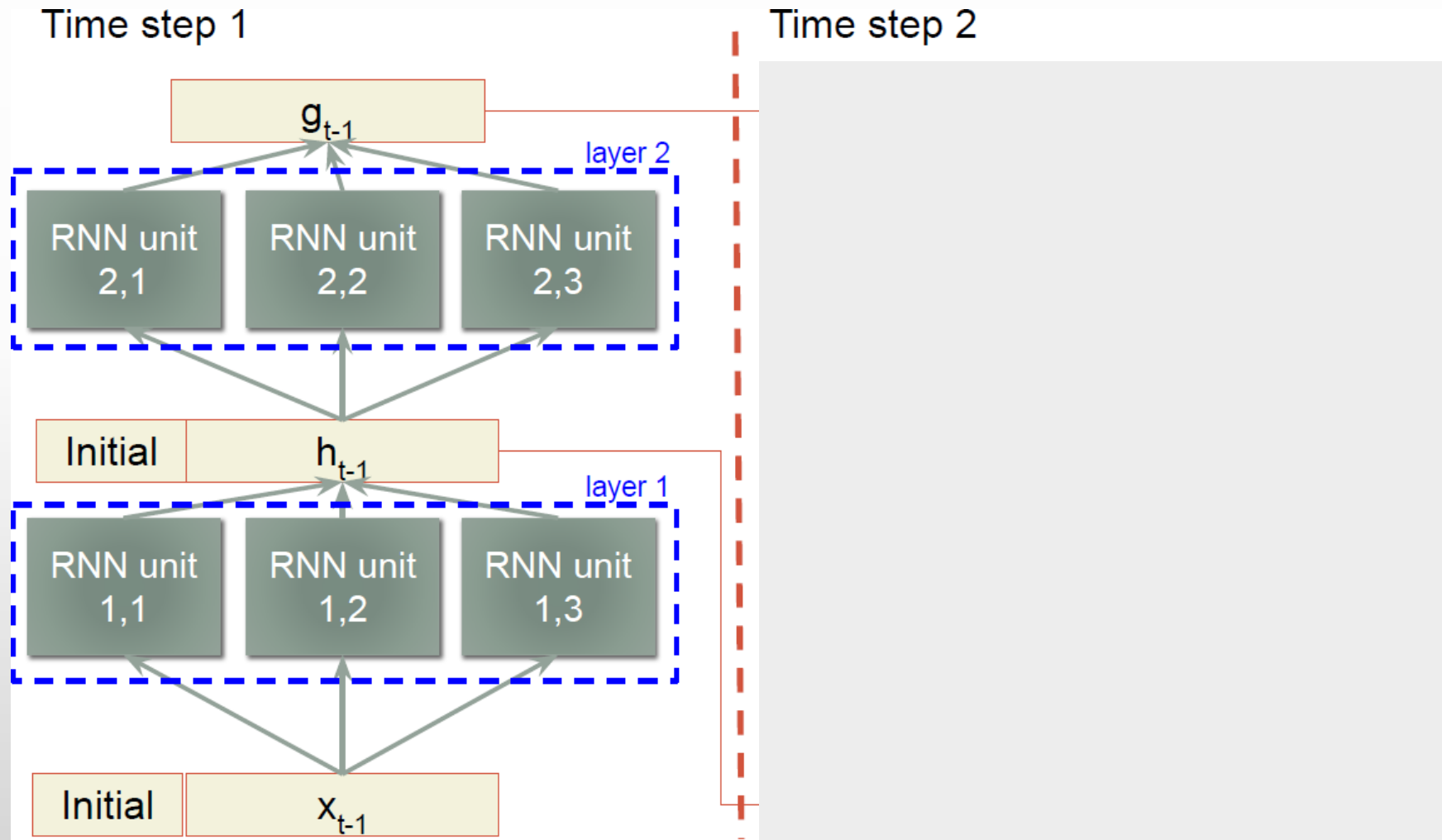
- Recurrent Neural network concept



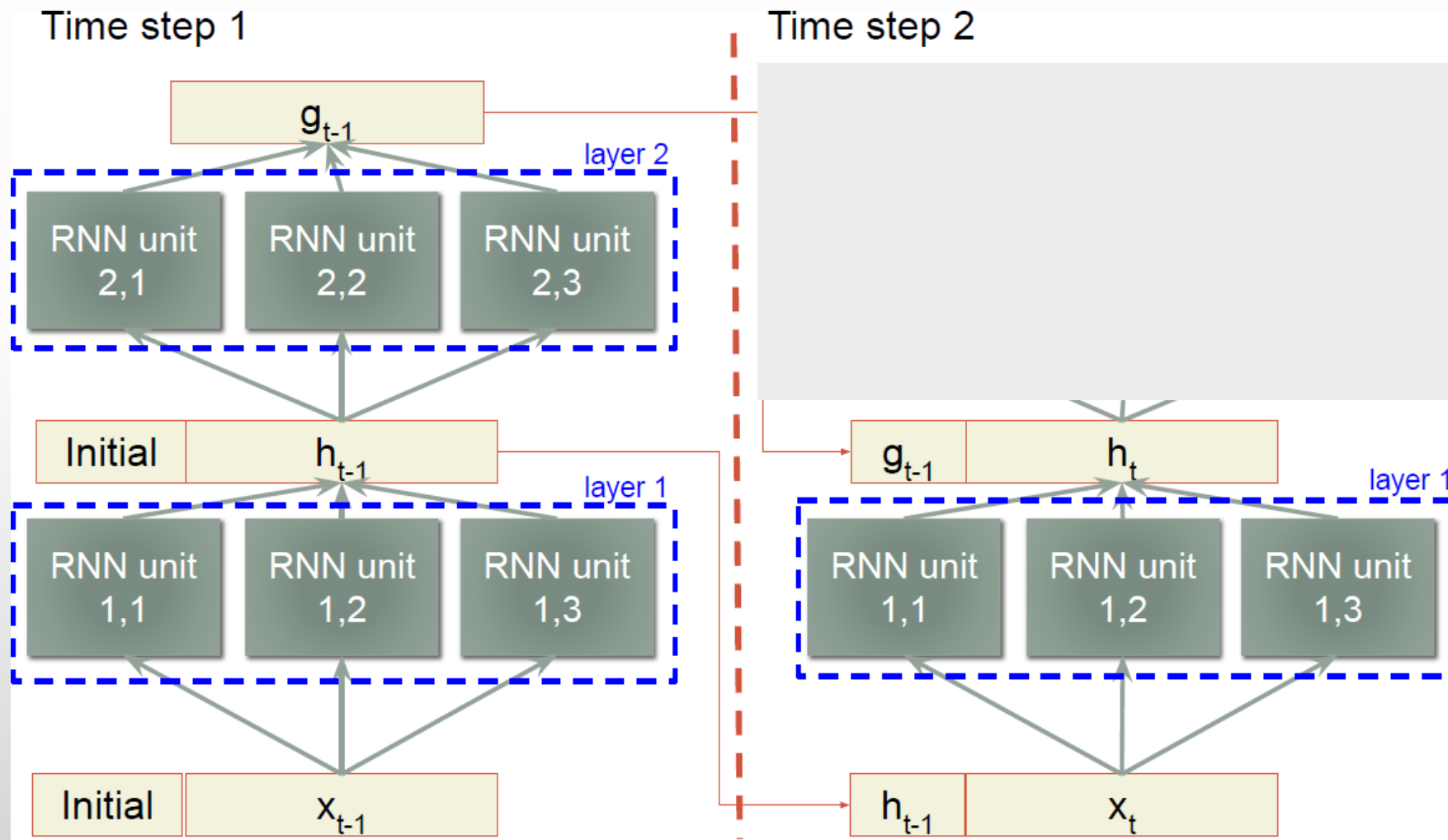
Recurrent Neural Networks (RNNs) (cont.)



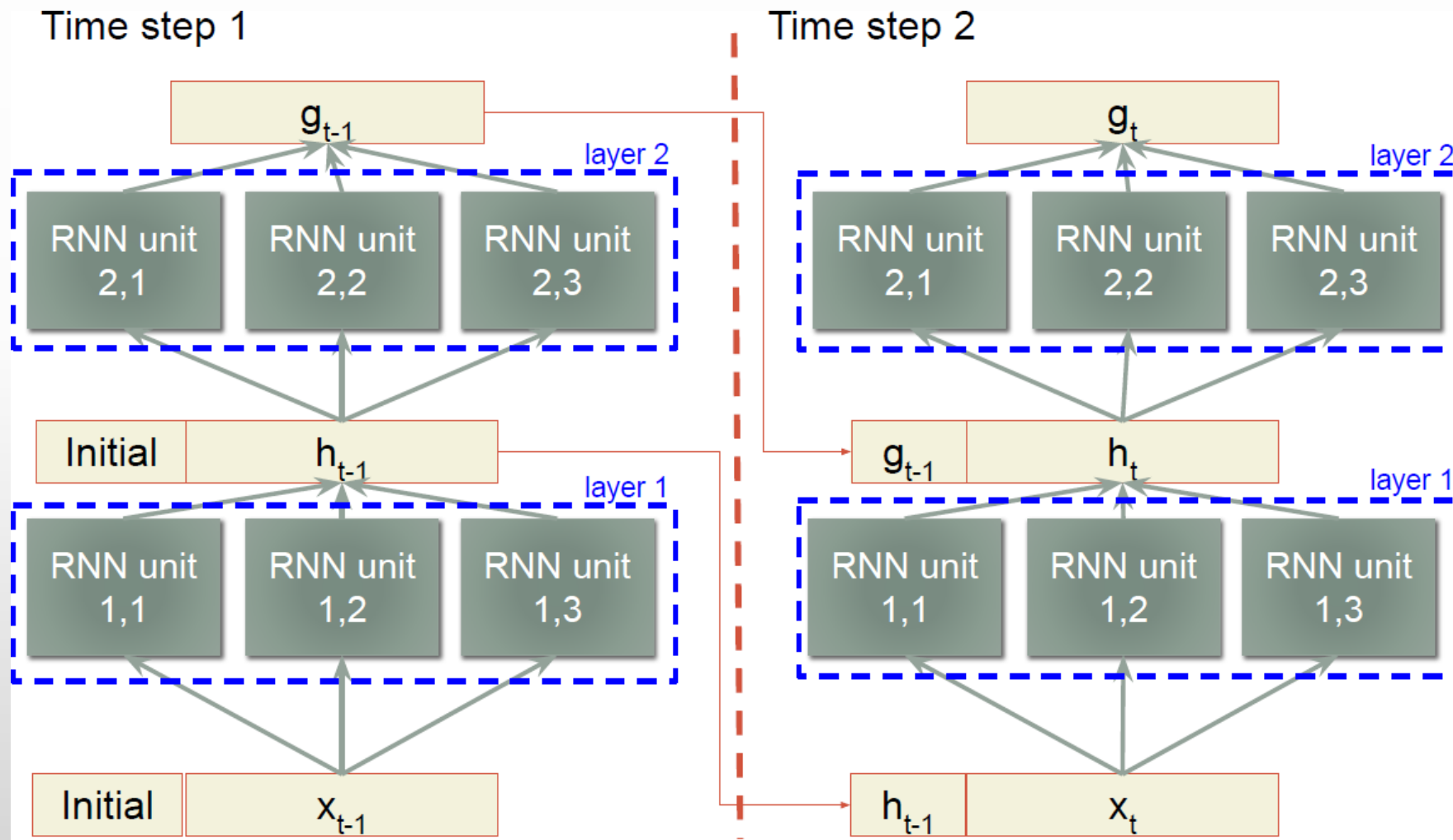
Recurrent Neural Networks (RNNs) (cont.)



Recurrent Neural Networks (RNNs) (cont.)

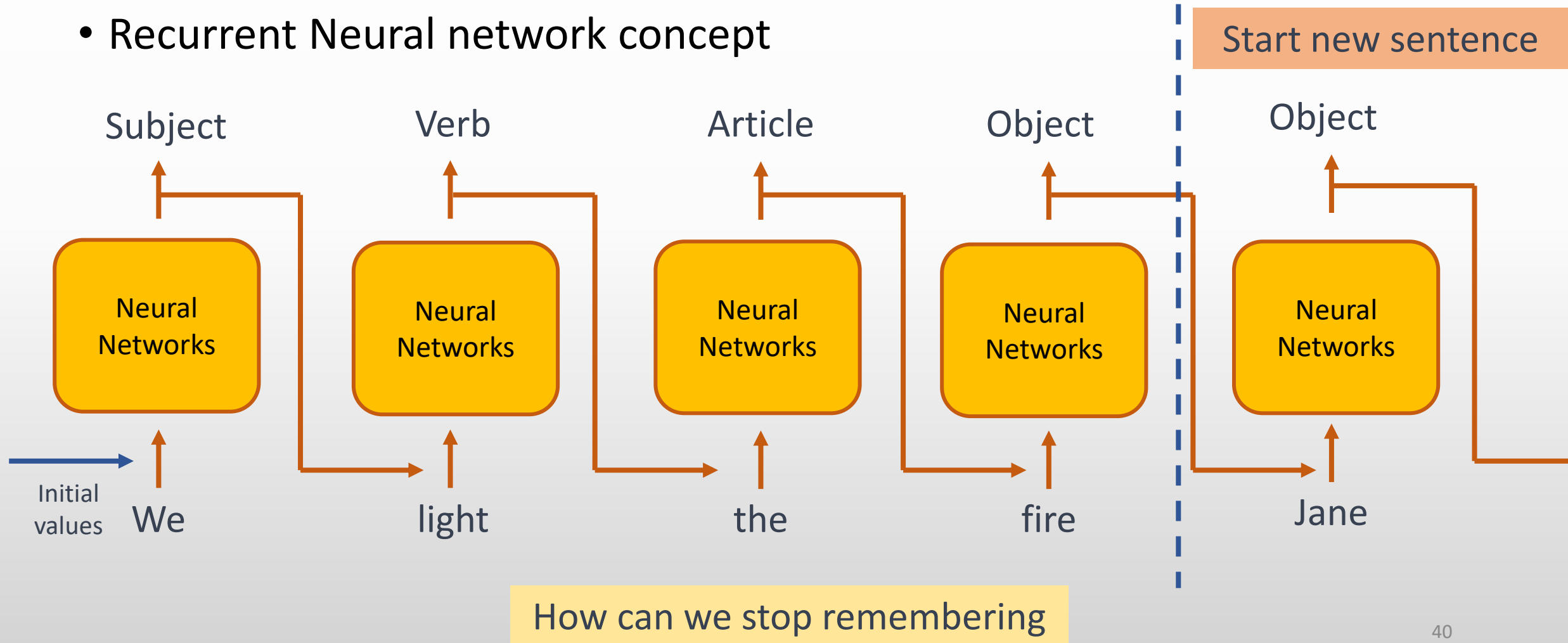


Recurrent Neural Networks (RNNs) (cont.)



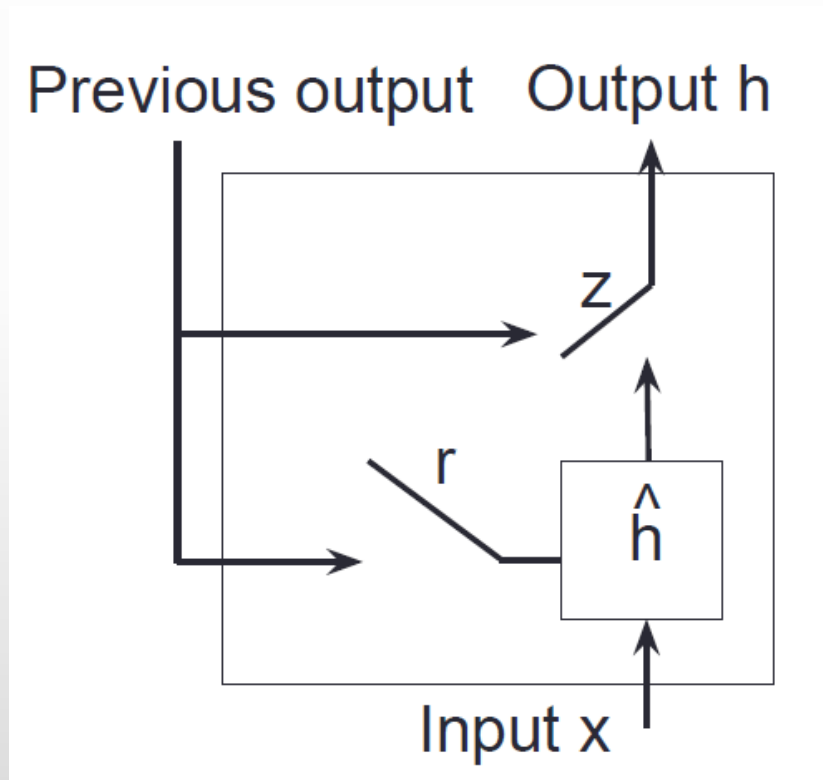
Recurrent Neural Networks (RNNs) (cont.)

- Recurrent Neural network concept



Gated Recurrent Unit (GRU)

- Add gates that can choose to reset (r) or update (z)



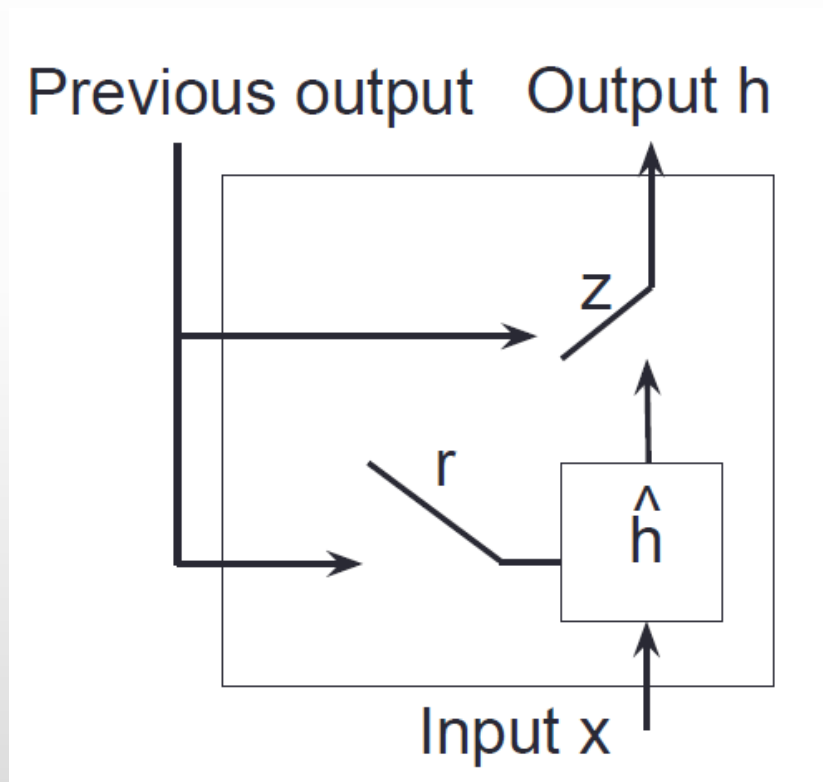
Neuron index

$$h_{\boxed{t}}^{\boxed{j}} = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

time index

Gated Recurrent Unit (GRU)

- Add gates that can choose to reset (r) or update (z)

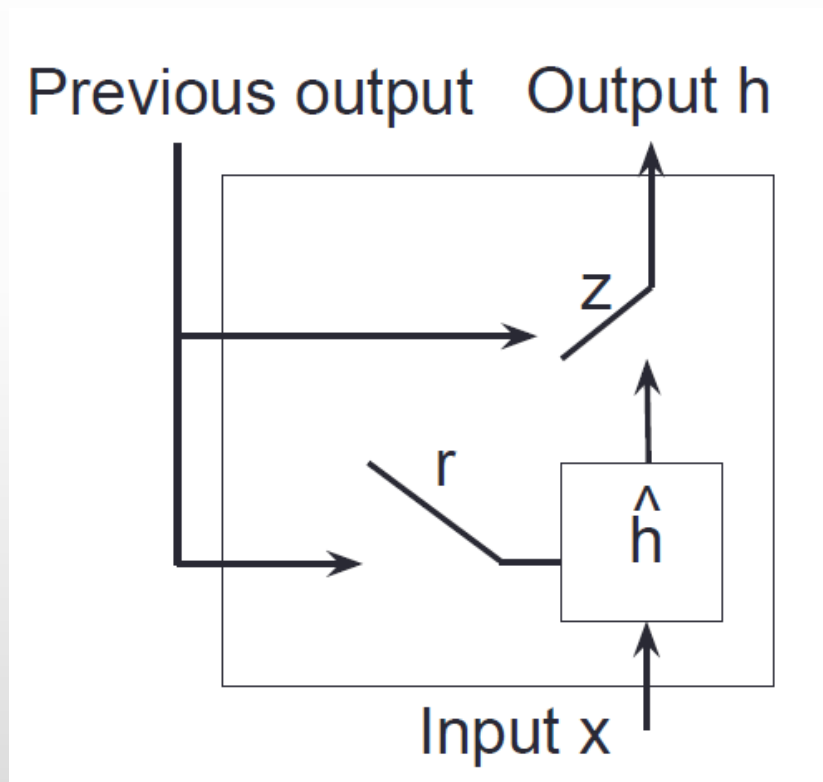


$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \hat{h}_t^j$$

One GRU neuron output (scalar)

Gated Recurrent Unit (GRU) (cont.)

- Add gates that can choose to reset (r) or update (z)



$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j$$

$$\hat{h}_t^j = \tanh^j(W \mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

Element-wise product

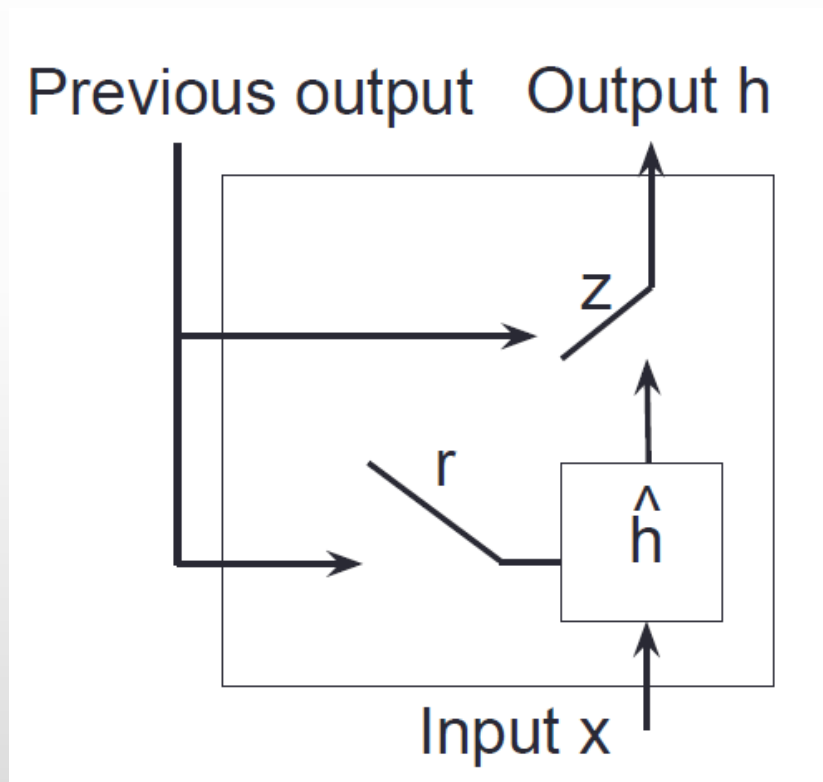
Linear transform with matrix multiply

Vector (each value from each GRU unit in the previous layer)

$$\mathbf{x}_t^j = \mathbf{h}_t^j$$

Gated Recurrent Unit (GRU) (cont.)

- Add gates that can choose to reset (r) or update (z)



$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j\hat{h}_t^j$$

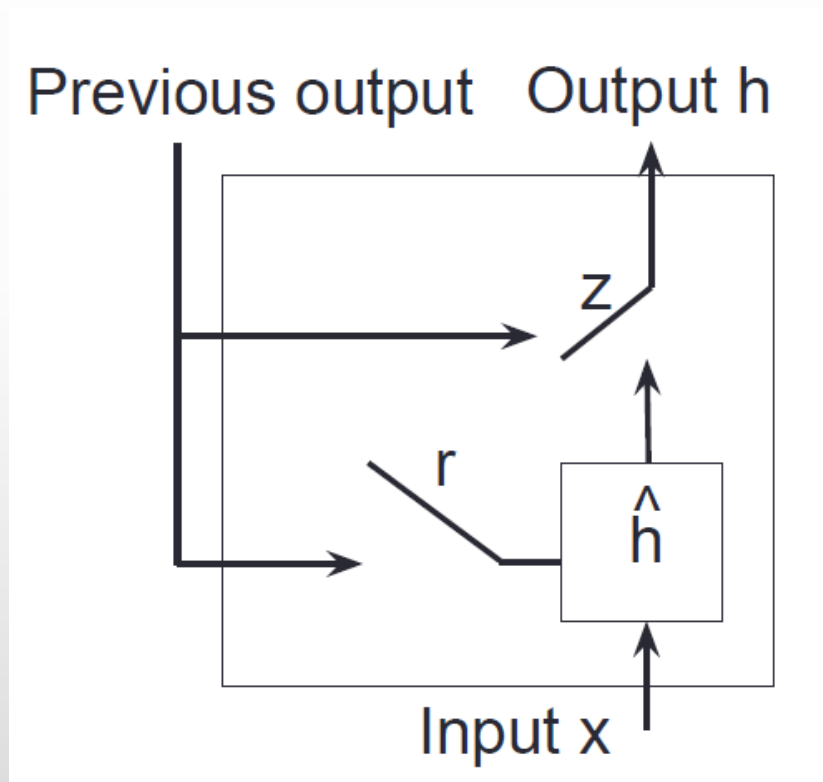
$$\hat{h}_t^j = \tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

$$z_t^j = \text{sigmoid}^j(W_{\boxed{z}}\mathbf{x}_t + U_{\boxed{z}}\mathbf{h}_{t-1})$$

Indicates a different set of weights

Gated Recurrent Unit (GRU) (cont.)

- Add gates that can choose to reset (r) or update (z)

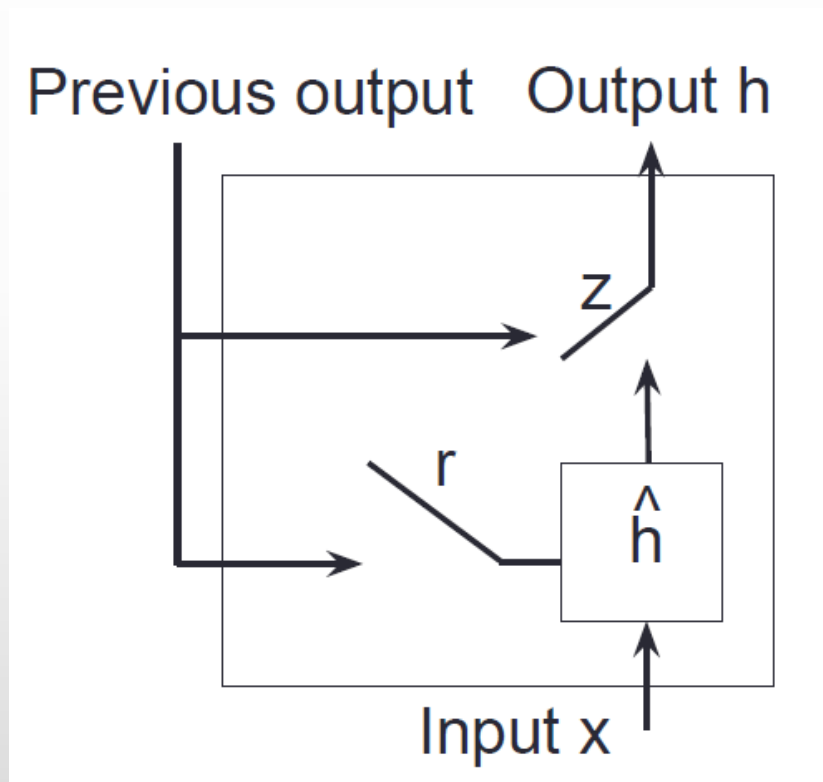


$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \hat{h}_t^j$$
$$\hat{h}_t^j = \tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$
$$z_t^j = \text{sigmoid}^j(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1})$$

Bounds the output to 0 to 1 for interpolation

Gated Recurrent Unit (GRU) (cont.)

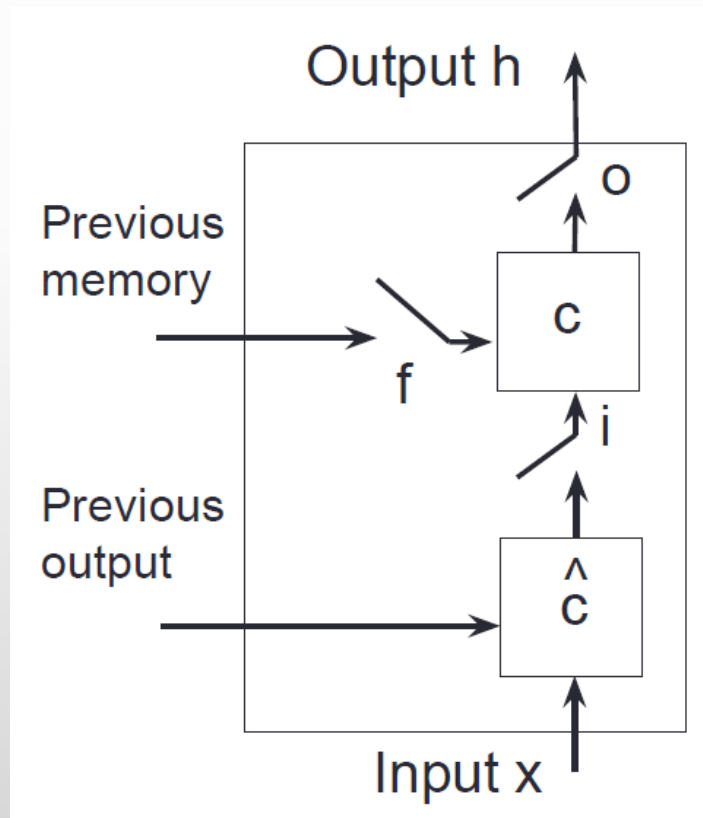
- Add gates that can choose to reset (r) or update (z)



$$\begin{aligned} h_t^j &= (1 - z_t^j)h_{t-1}^j + z_t^j\hat{h}_t^j \\ \hat{h}_t^j &= \tanh^j(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \\ z_t^j &= \text{sigmoid}^j(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1}) \\ r_t^j &= \text{sigmoid}^j(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1}) \end{aligned}$$

Long Short-Term Memory (LSTM)

- Have 3 gates, forget (f), input (i), output (o)
- Has an **explicit memory cell** (c)



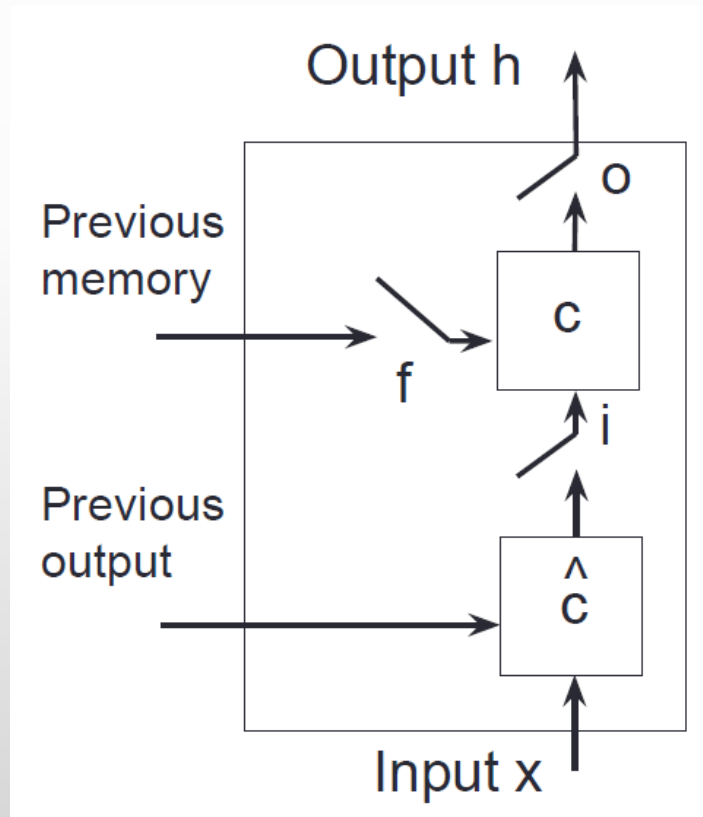
$$\begin{aligned}i_t^j &= F^j(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1}) \\o_t^j &= F^j(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t) \\f_t^j &= F^j(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_j \mathbf{c}_{t-1})\end{aligned}$$

Contribution from memory “Peephole connection”

Vs are diagonal matrices (Each cell can only see its own memory)

Long Short-Term Memory (LSTM) (cont.)

- Have 3 gates, forget (f), input (i), output (o)
- Has an **explicit memory cell** (c)



$$i_t^j = F^j(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1})$$

$$o_t^j = F^j(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t)$$

$$f_t^j = F^j(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_j \mathbf{c}_{t-1})$$

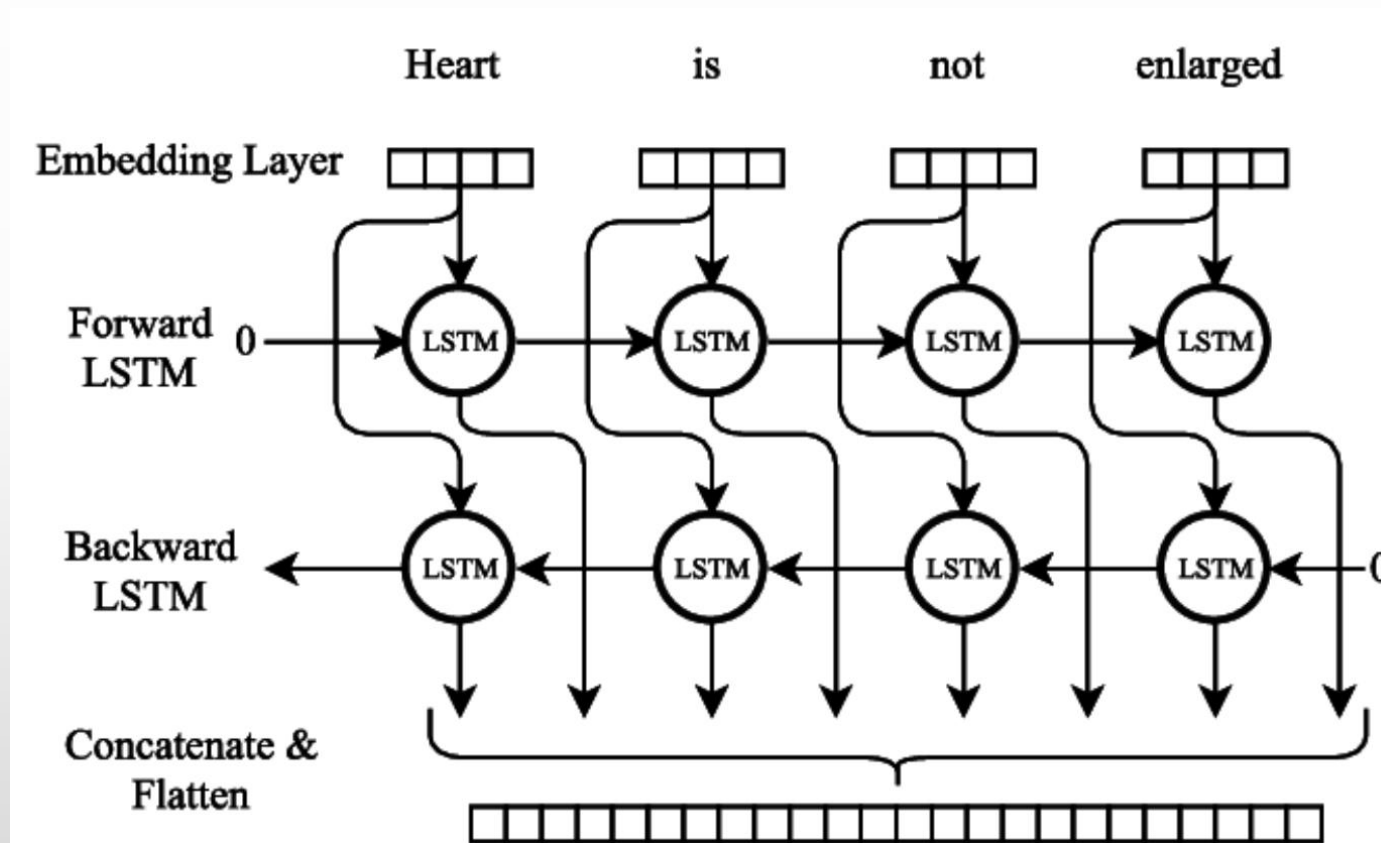
$$h_t^j = o_t^j \tanh(c_t^j)$$

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \hat{c}_t^j$$

$$\hat{c}_t^j = \tanh^j(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1})$$

Bi-directional LSTM

- Most of the time information from the future is useful for predicting the current output



<https://paperswithcode.com/method/bilstm>

External sources

ROBOTICS LEARNING HUB


หน้าหลักบทเรียนบทความyoutube

Teacher
LEARNING ROBOTICS4ALL

Category:
COMPUTER SCIENCE AI

ENROLL COURSE

DescriptionCurriculumReviews



Enrolled: 60 students

Lectures: 8

Level: ระดับกลาง

Overview

Name Course: Introduction to Deep Learning for Sequential Data [Intermediate]

Instructor: Paisit Khanarsa Ph.D.

Course Description

คอร์สนี้ถูกจัดทำขึ้นเพื่อเรียนรู้ความหมายของ Sequential data รวมถึงแนวคิดของ Deep learning model ที่เกี่ยวข้องกับการจัดการข้อมูลประเภท Sequential data ซึ่งประกอบด้วย Recurrent neural networks, Long short term memory และ Gate recurrent unit และพื้นฐานการนำไปประยุกต์ใช้ผ่านการเขียนโปรแกรมด้วยภาษา Python ผ่าน Keras library เพื่อจัดการกับปัญหา Natural language processing และ Time series forecasting

ROBOTICS LEARNING HUB

หน้าหลักบทเรียนบทความyoutube

Introduction to Deep Learning for Sequential Data

Add to Wishlist

ENROLL COURSE

Teacher
LEARNING ROBOTICS4ALL

Category:
COMPUTER SCIENCE AI

Enrolled: 60 students

Lectures: 8

Level: ระดับกลาง

DescriptionCurriculumReviews

Introduction to Deep Learning for Sequential Data

1	EP1: Deep Learning for Sequential Data	19
2	EP2: Recurrent Neural Networks	33
3	EP3: Long Short Term Memory	29
4	EP4: Gate Recurrent Unit	20
5	EP5: Basic Natural Language Processing to Text Classification using RNN – Part 1	18
6	EP6: Basic Natural Language Processing to Text Classification using RNN – Part 2	28
7	EP7: Time Series Forecasting to Predict Stock Market – Part 1	10
8	EP8: Time Series Forecasting to Predict Stock Market – Part 2	19

ติดต่อเรา

126 ถนนประชาธิปไตย แขวงบางมด เขตทุ่งครุ กรุงเทพมหานคร 10140

support@thairobotic



หน่วยงาน

FIBO
KMUTT

สนับสนุนโดย

บพค.
สวช.

SOCIAL MEDIA

<https://learn.thairobotics.org/courses/introduction-to-deep-learning-for-sequential-data-intermediate/>

Deep Learning in NLP

- LSTM remembers meaningful things

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Adding meaning in NLP

- One hot encoding
- Dense representation (embedding)

Adding meaning in NLP

- One hot encoding
 - Categorical representation example:
 - Apple ---> 1 ---> [1,0,0,0,...]
 - Bird ---> 2 ---> [0,1,0,0,...]
 - Cat ---> 3 ---> [0,0,1,0,...]
 - Car ---> 5 ----> [0,0,0,0,1,...]

Adding meaning in NLP

- One hot encoding
 - Categorical representation example:
 - Apple ---> 1 ---> [1,0,0,0,...]
 - Bird ---> 2 ---> [0,1,0,0,...]
 - Cat ---> 3 ---> [0,0,1,0,...]
 - Car ---> 5 ----> [0,0,0,0,1,...]
 - Sparse representation: most values are zero

Adding meaning in NLP

- One hot encoding
 - Categorical representation example:
 - Apple ---> 1 ---> [1,0,0,0,...]
 - Bird ---> 2 ---> [0,1,0,0,...]
 - Cat ---> 3 ---> [0,0,1,0,...]
 - Car ---> 5 ----> [0,0,0,0,1,...]
 - Sparse representation: most values are zero
 - Can not represent meaning : |Apple-Bird| = |Bird-Cat|

Adding meaning in NLP

- One hot encoding
 - Categorical representation example:
 - Apple ---> 1 ---> [1,0,0,0,...]
 - Bird ---> 2 ---> [0,1,0,0,...]
 - Cat ---> 3 ---> [0,0,1,0,...]
 - Car ---> 5 ----> [0,0,0,0,1,...]
 - Sparse representation: most values are zero
 - Can not represent meaning : |Apple-Bird| = |Bird-Cat|
 - Solve by Getting meaning into the feature vectors (Concatenate)
 - Apple ---> 1 ---> [1,0,0,0,..., 1,0]
 - Bird ---> 2 ---> [0,1,0,0,..., 0,1]
 - Cat ---> 3 ---> [0,0,1,0,..., 0,1]
 - Car ---> 5 ----> [0,0,0,0,1,..., 1,0]

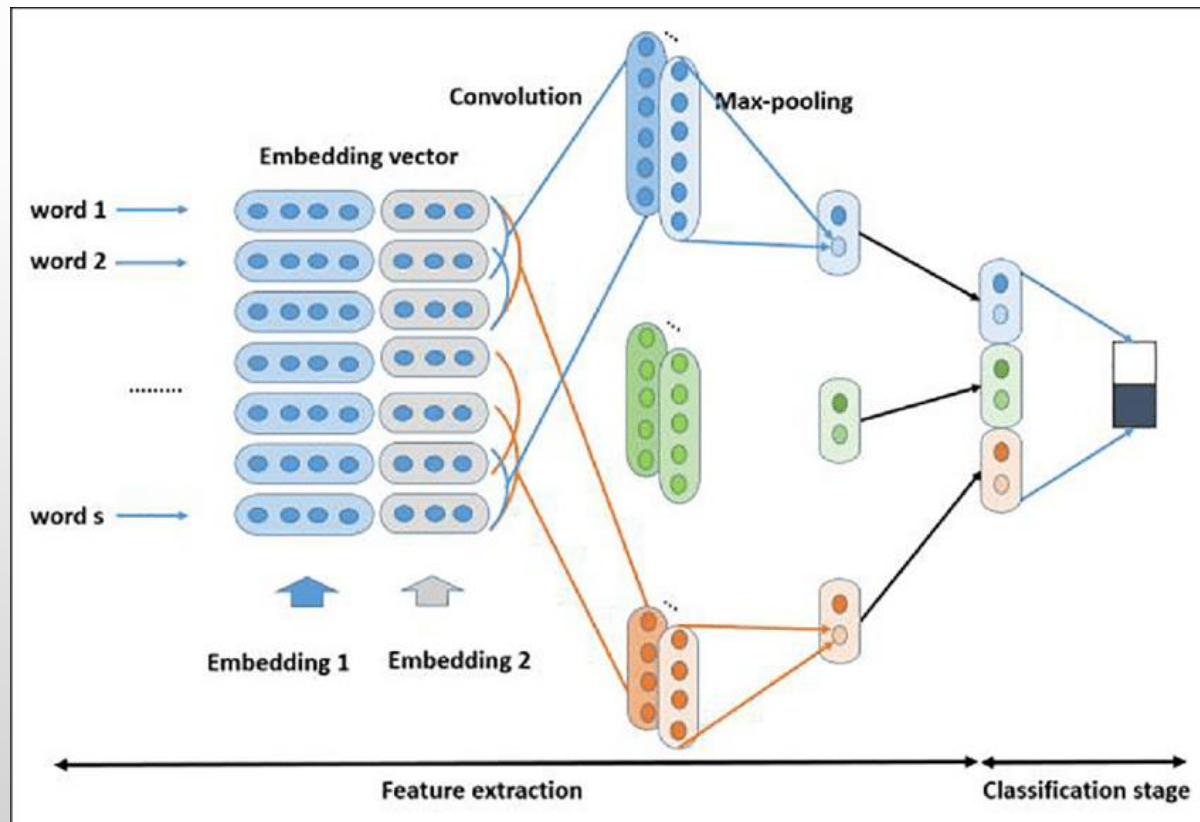
Feature vectors

Adding meaning in NLP

- Dense representation (embedding)
 - Encode sparse representation into a lower dimensional space
 - Apple ---> 1 ---> [1,0,0,0,...] ---> [1.2,2.5]
 - Bird ---> 2 ---> [0,1,0,0,...] ---> [0.3,0.6]
 - Cat ---> 3 ---> [0,0,1,0,...] ---> [0.25,0.71]
 - Car ---> 5 ---> [0,0,0,0,1,...] ---> [1.6,2.1]

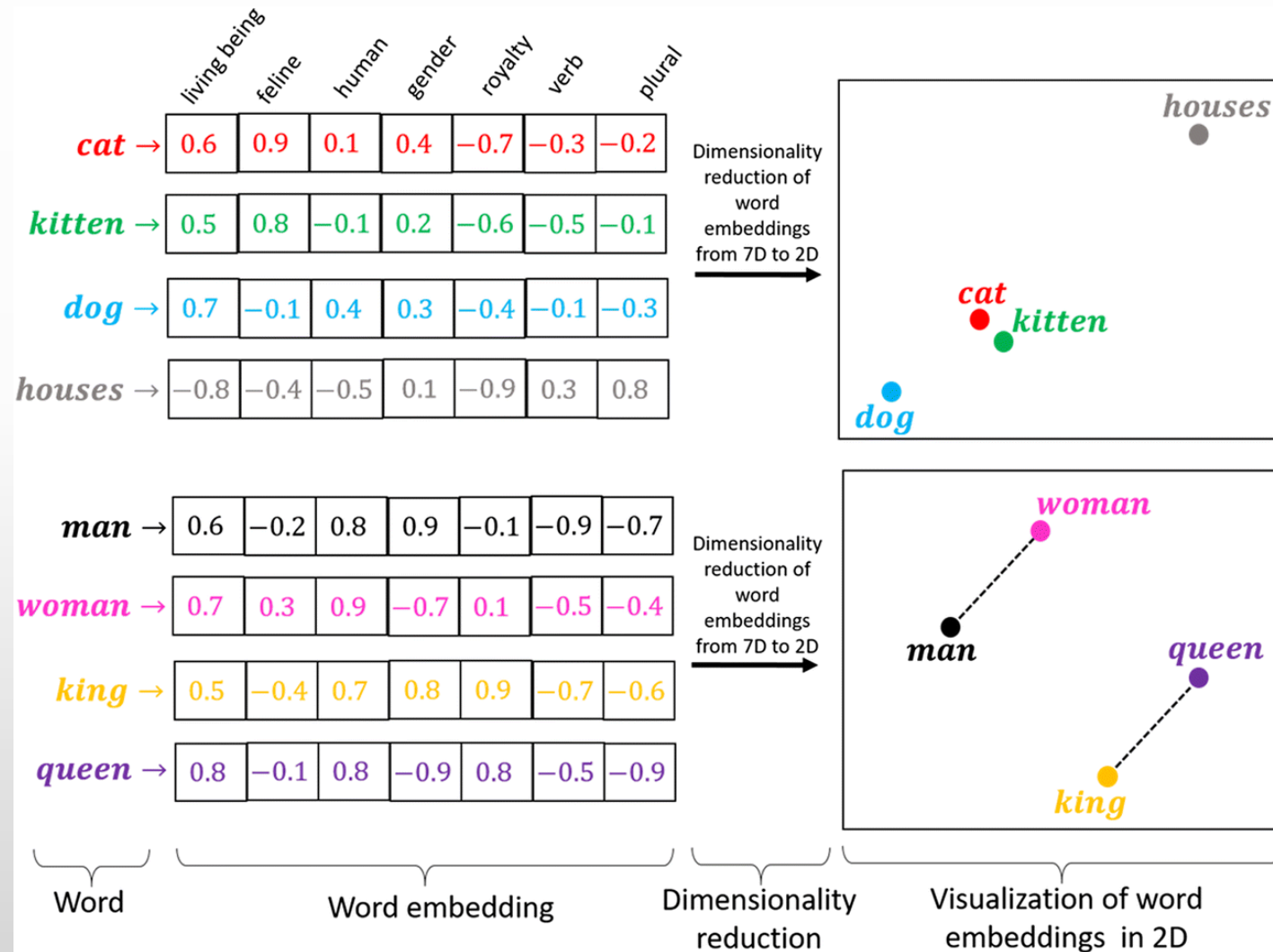
Adding meaning in NLP

- Dense representation (embedding)
 - Encode sparse representation into a lower dimensional space
 - Apple ---> 1 ---> [1,0,0,0,...] ---> [1.2,2.5]



<https://www.researchgate.net/figure/Proposed-CNN-on-multiple-word-embeddings-concatenated-at-embedding-layer-fig3-333752473>

Embeddings



Embeddings and meaning (semantic)

- Character Embedding of 32 dims to 2 dims for visualization

