Artificial Neural Networks and Deep Learning

Daniel Mallory
Saint Michael's College

Outline

Early Ideas

1. Cybernetics: From the greek ′κυβερνήτης′ (cybernetes) meaning 'to steer'
   a. A general formulation of the goal-oriented process; when 'steering' towards some goal, and you get off course, adjust and move back towards the goal
   b. Loop of: Trying → acting → noting difference → adjusting
   c. This 'goal oriented process' is a fundamental property of all 'intelligent' things
   d. This approach to Artificial Intelligence is more similar to the idea of 'Reinforcement Learning'.
2. The Buzzword: Artificial Intelligence
   i. The classic 'I, Robot' perspective (both the Asimov book and the Will Smith movie)
   ii. For all intensive purposes, this doesn't exist

AI Waves and Winters

1. First Wave (1956-1974): The 'perceptron', which is the basis for modern FCNNs, was published by Frank Rosenblatt.
2. First Winter (1974-1980): Financial setbacks combined with lack of computing power and overpromises.
3. Second Wave (1980-1987): The classic "Backpropagation" paper was published, which is now the workhorse of modern nets. The neocognitron (Fukushima et al. 1980) and the fully backpropagated CNN (LeCun et al. 1988) papers were published.
4. Second Winter (1987-1993): Rule-based systems don't generalize too well. Funding low, and hype dies down.
5. Third Wave (1993-Present): Processing power caught up with computational requirements of neural nets. Large datasets and better algorithms (and faster GPUs/CUDA).

Why now?

1. Larger datasets give more accurate pictures (ha!) of tasks in which neural networks are applied
2. Computing power (i.e. GPU/TPU) has increased massively, making processing power no longer a restricting bottleneck (depending on your resources)
3. Architectures now exist that allow people to train sub-architectures of networks in parallel (i.e. on multiple GPUs and CPUs at once)
4. People have just had better ideas; the inception of ideas like Convolutional Networks, Recurrent Networks, and algorithms to improve training time (i.e. Batch normalization, dropout)

16k 485 Share

BEST

u/Skizm • 2mo

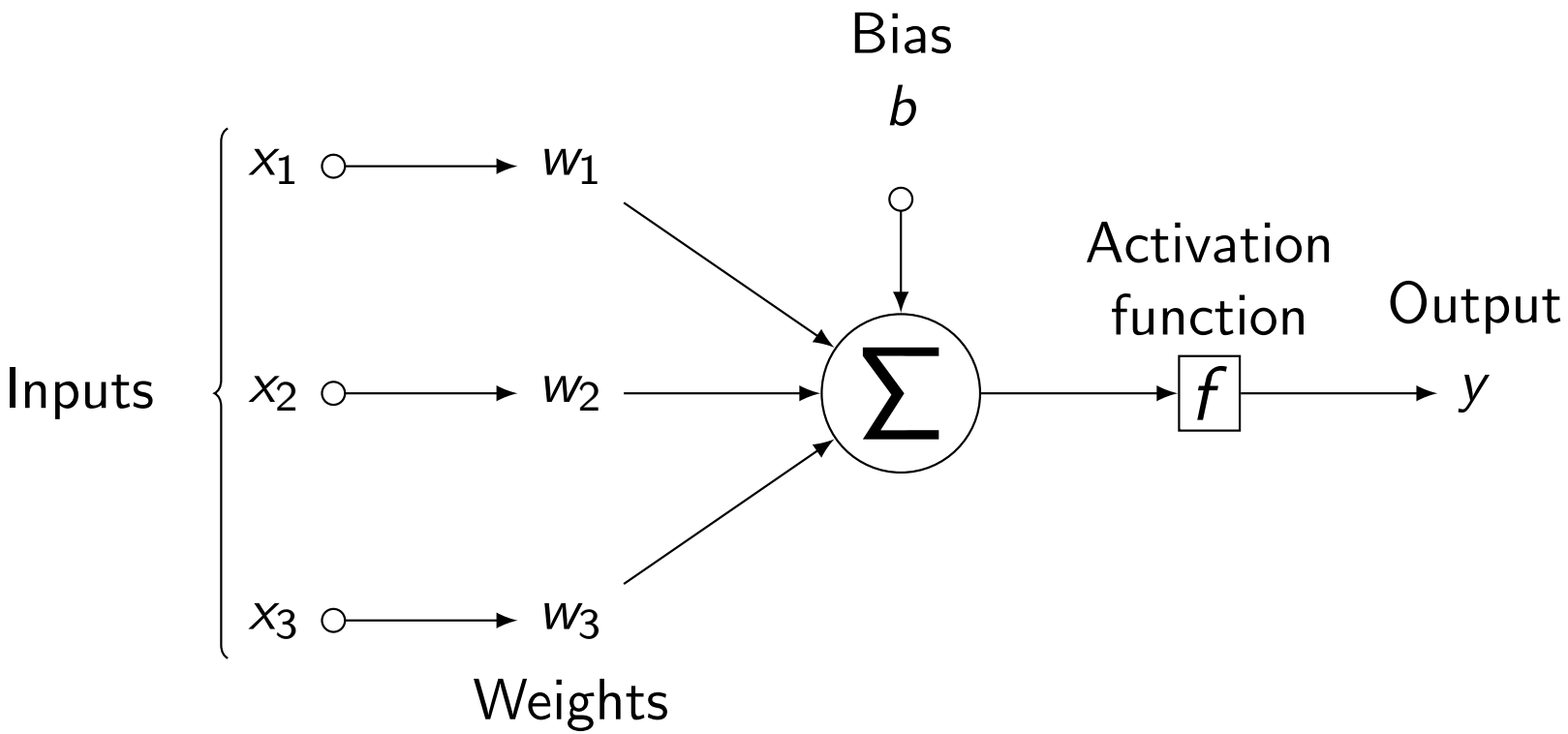
```
if(goingToCrashIntoEachOther)
{ dont(); }
```

Why Neural Networks?

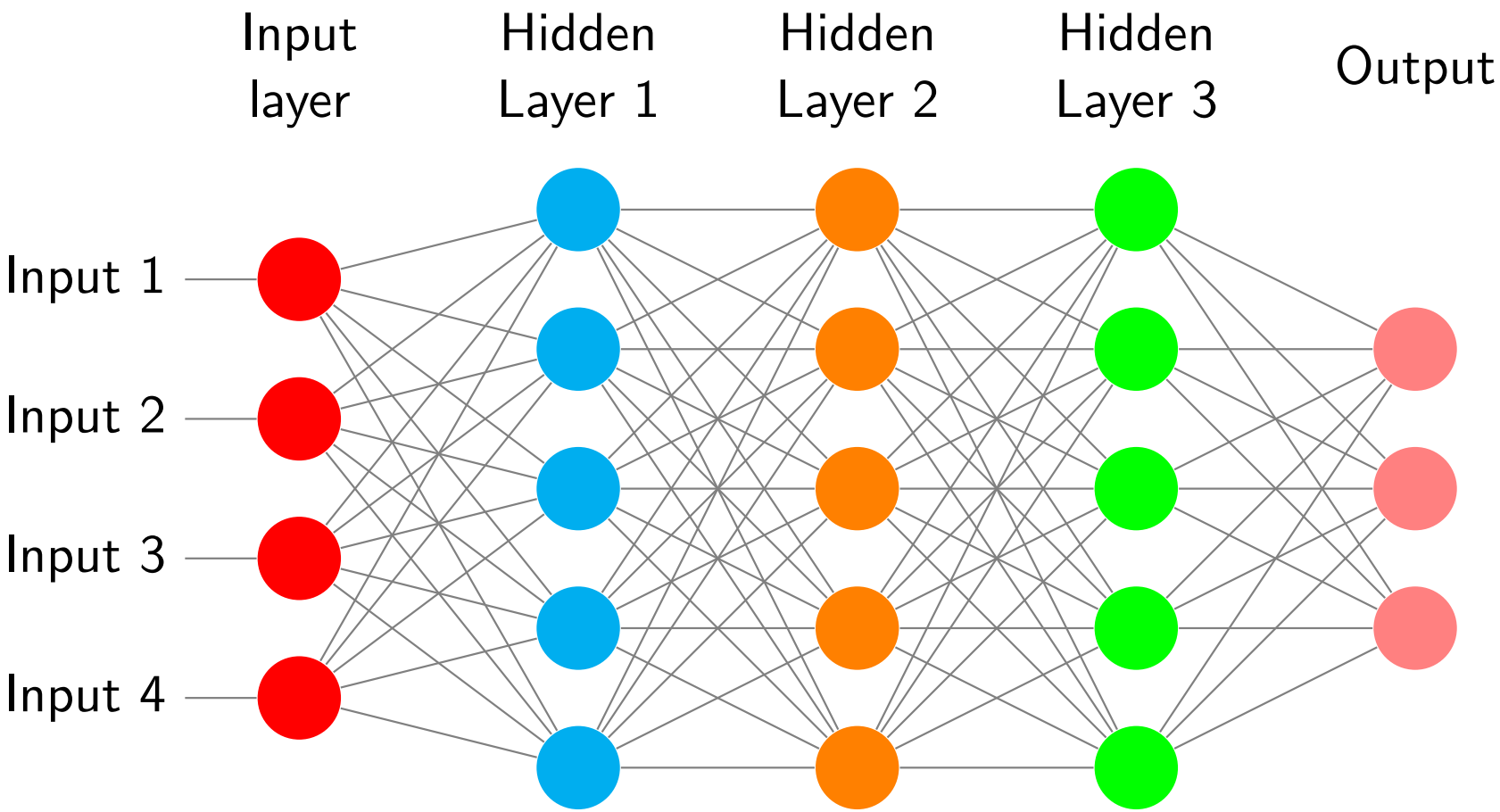
1. Traditionally, computers are much faster than humans at computations
   a. Complex optimization problems
   b. Matrix multiplication
2. But the reverse is also true; it is difficult to program computers to perform tasks at which humans naturally excel
   a. Visual recognition / spatial organization
   b. Natural language: speech and translation
   c. Driving
3. Traditional statistical and computational methods struggle to accurately capture the complexity of these tasks
   a. Too many rules to encode
   b. Encoding varies too much between examples
   c. Asymptotics: more data $\rightarrow$ worse performance?

An example...

The "Neuron"


Bias
b
x_1
x_2
x_3
w_1
w_2
w_3
Inputs
Weights
$\sum$
Activation function
f
Output
y

A 'Deep' Neural Network



Forward Propagation

It's sometimes helpful to think about neural networks as long chains of function compositions. Each 'layer' of the network is vector of 'functions' (or neurons) which do the following:

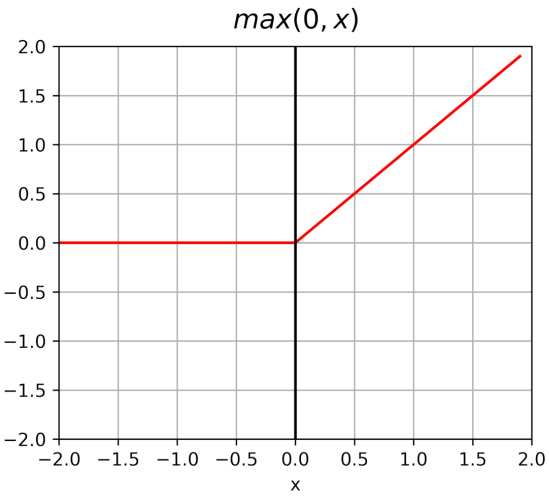
1. Receives the output **x** of the previous layer
2. Computes the dot product/matrix multiplication of x with its weight matrix **W** and adds its bias vector **b**
3. Applies its non-linearity function to obtain its 'output' vector $\mathbf{x}'$ which is fed into the next layer

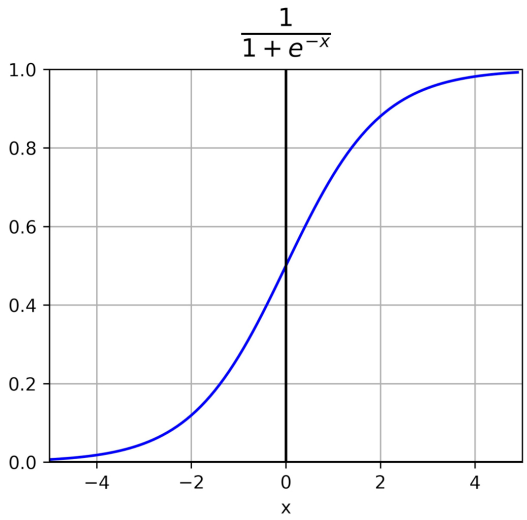
From the previous example:

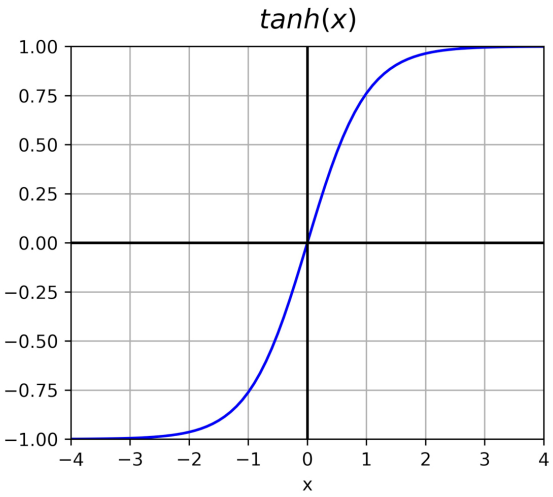
$$x' = \sigma\left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} \\ w_{5,1} & w_{5,2} & w_{5,3} & w_{5,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}\right) = \sigma\left(\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix}\right) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}$$

The Activation Function

1. Why the activation function at all?
   a. Activation functions allow ANNs to model arbitrarily complex non-linear functions
   b. Without activation functions, network depth has no effect on the functions these networks can learn
2. Common activation functions:







Correcting Errors

1. How do we drive our network's weights and biases towards the values that will give us the most correct predictions?

2. It makes sense to push the network's output as close to the desired output as possible via the minimization of some loss function $\mathcal{L}$

3. This allows us to 'back-propagate' the error of the output with respect to the loss function $\mathcal{L}$ to all other weights and biases in the network

Backpropagation and Gradient Descent (Rumelhart et al. 1987)

1. Backpropagation involves adjusting each weight and bias in the network respective to its contribution to the overall error $\mathcal{L}$
2. Weights and biases from nodes with higher activations will contribute greater to the overall 'error' average

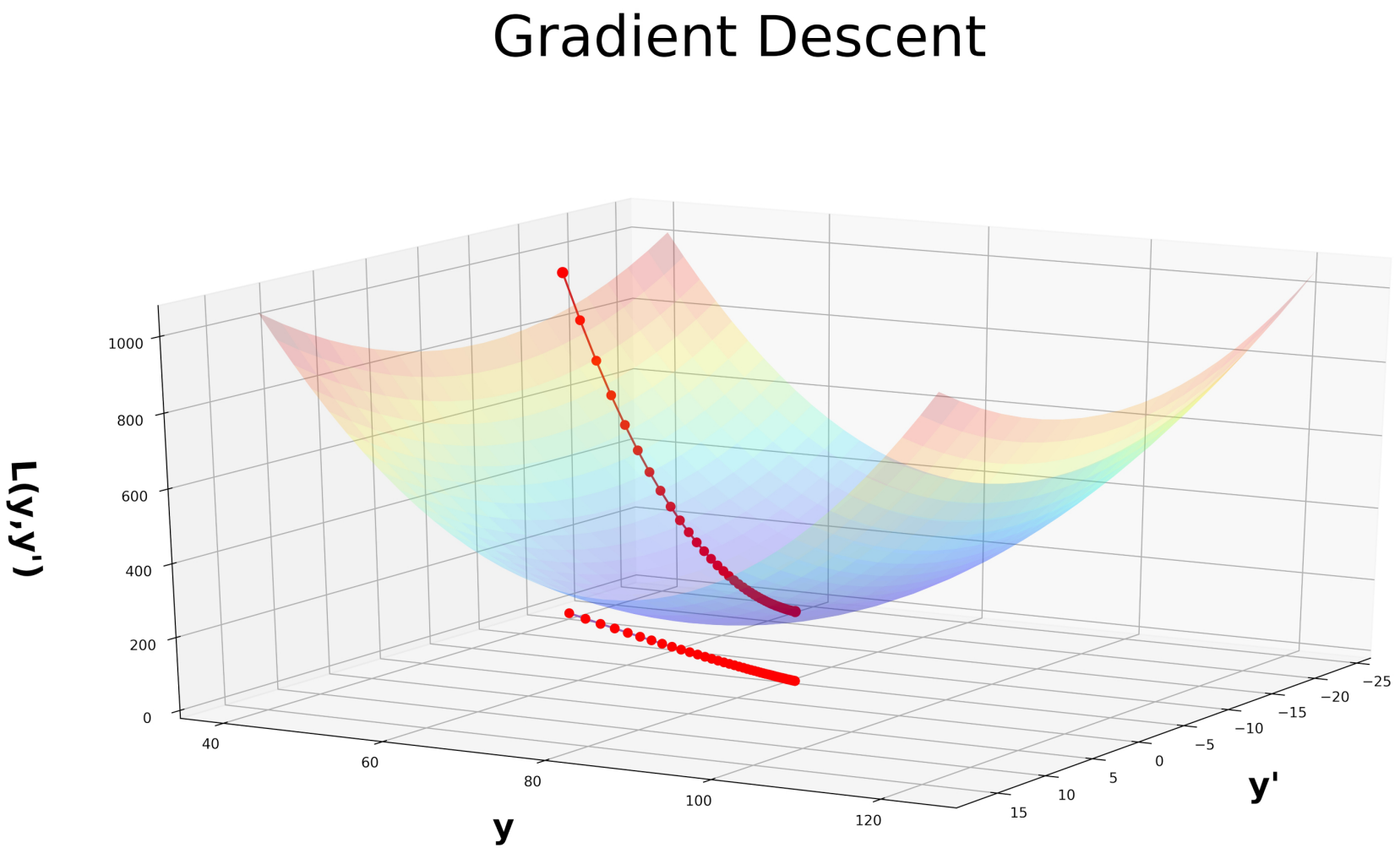
$$\mathcal{L}(y, y') = \frac{1}{2m} \sum_{i=1}^{m} (y' - y)^2$$

3. Weight and bias updates can be made by calculating the gradients of the loss function with respect to the parameters

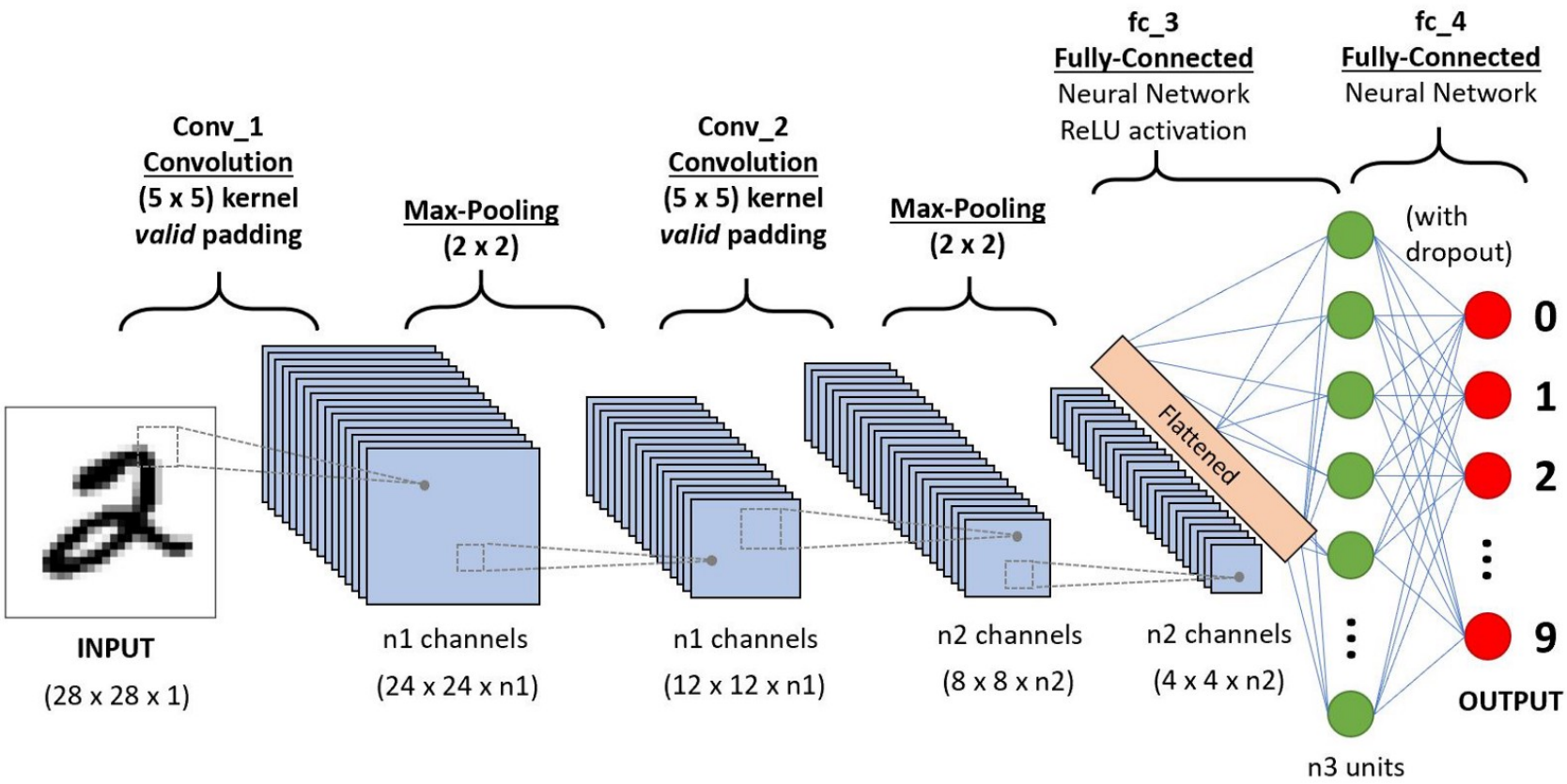
$$\mathbf{W^{[L]}}_{t+1} = W^{[L]}{}_t - \eta \frac{\partial \mathcal{L}}{\partial W^{[L]}{}_t} = W^{[L]} - \eta \nabla_{\theta_w} \mathcal{L}$$

$$\mathbf{b^{[L]}}_{t+1} = b_t^{[L]} - \eta \frac{\partial \mathcal{L}}{\partial b_t^{[L]}} = b_t^{[L]} - \eta \nabla_{\theta_b} \mathcal{L}$$

What does Gradient Descent Look Like?



Convolutional Networks (CNNs)



Why CNNs?

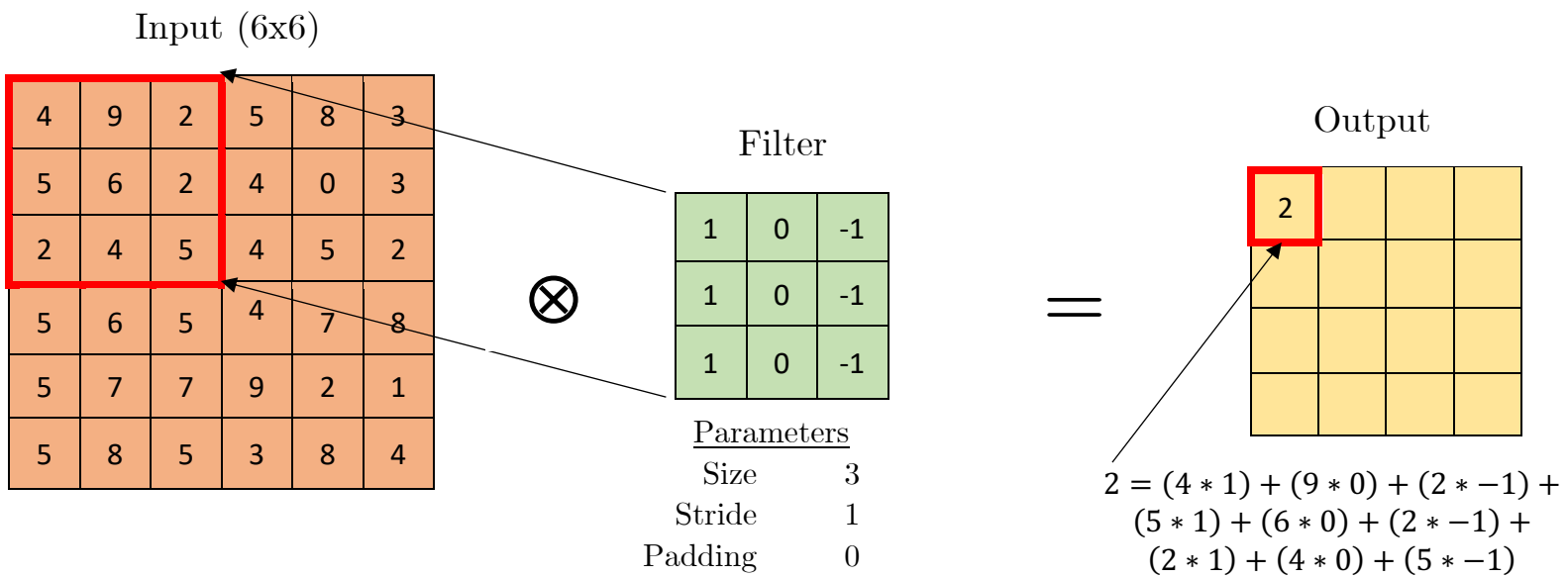
Visualization (Harley 2017)

1. Inspired by visual 'receptive fields' - monkey visual cortexes contain neurons that individually respond to small 'regions' in the visual field (Hubel and Wiesel, 1968)
2. Ex: Two different pictures of cats, while both pictures of cats, may trigger completely different groups of neurons
3. CNNs use image kernels to capture complex patterns in different spatial arrangements of similar output classes
4. The Kernel 'Formula'

$$G[m,n] = (f \circ g)[m,n] = \sum_j \sum_k g[j,k] f[m-j, n-k]$$

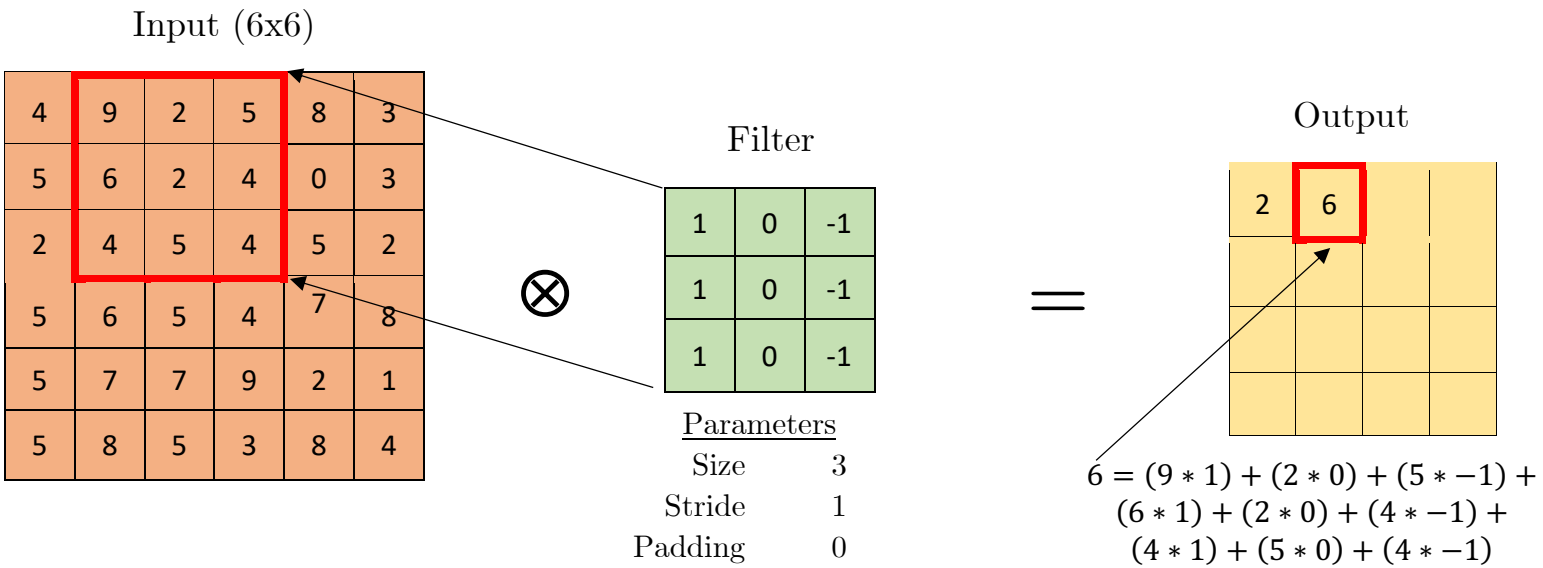
The Convolution Operation

Step 1: Overlay the filter to the input, perform element-wise multiplication, and sum the result

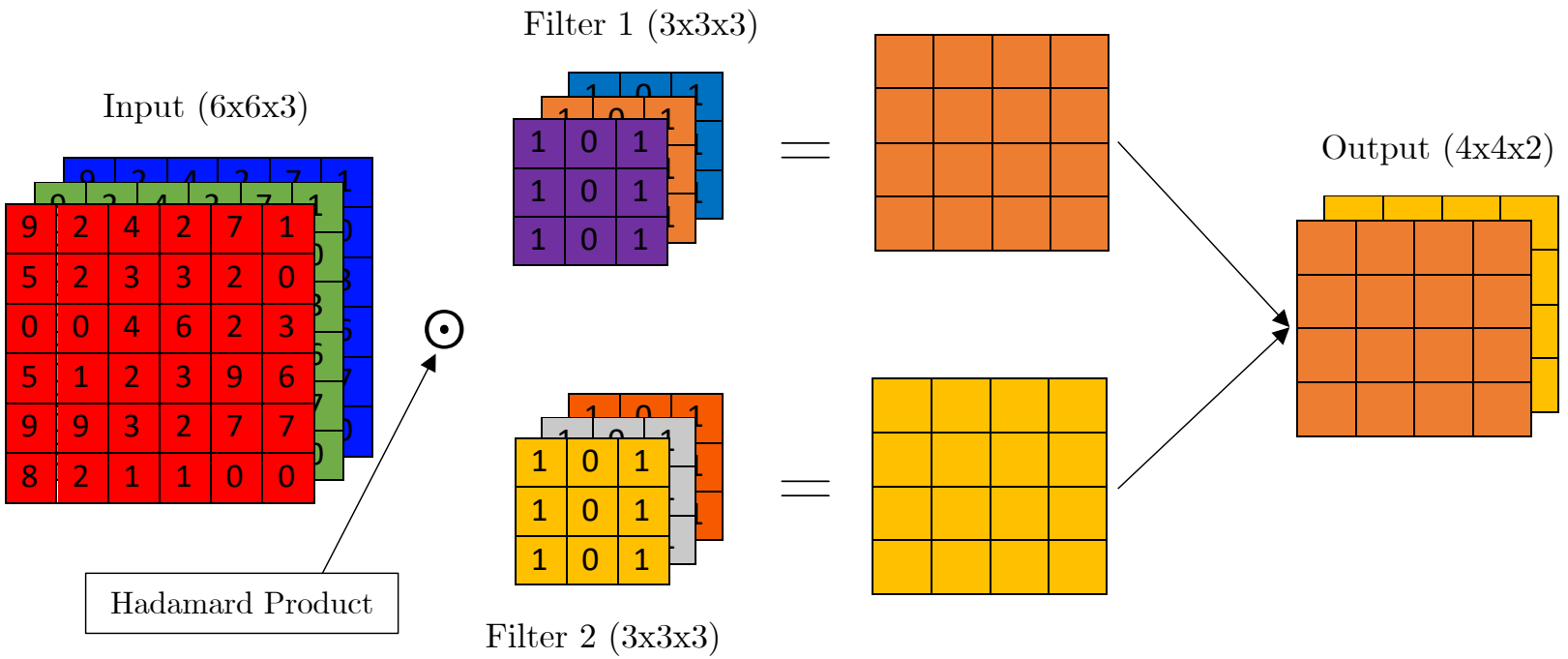


The Convolution Operation (Continued)

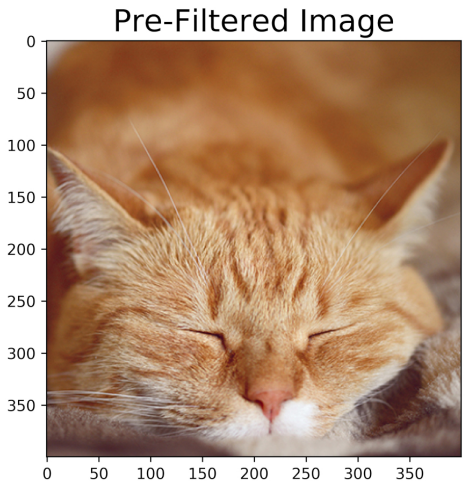
Step 2: Move the overlay right k positions (according to stride parameter) and repeat above calculation for new submatrix

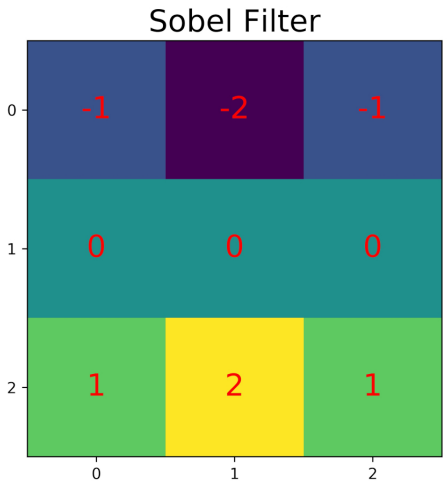


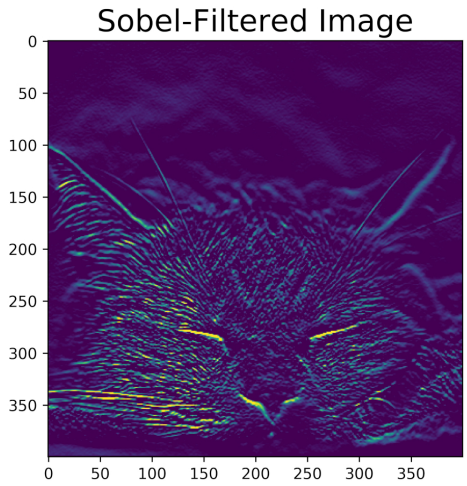
The Convolution Operation (Continued, Continued))



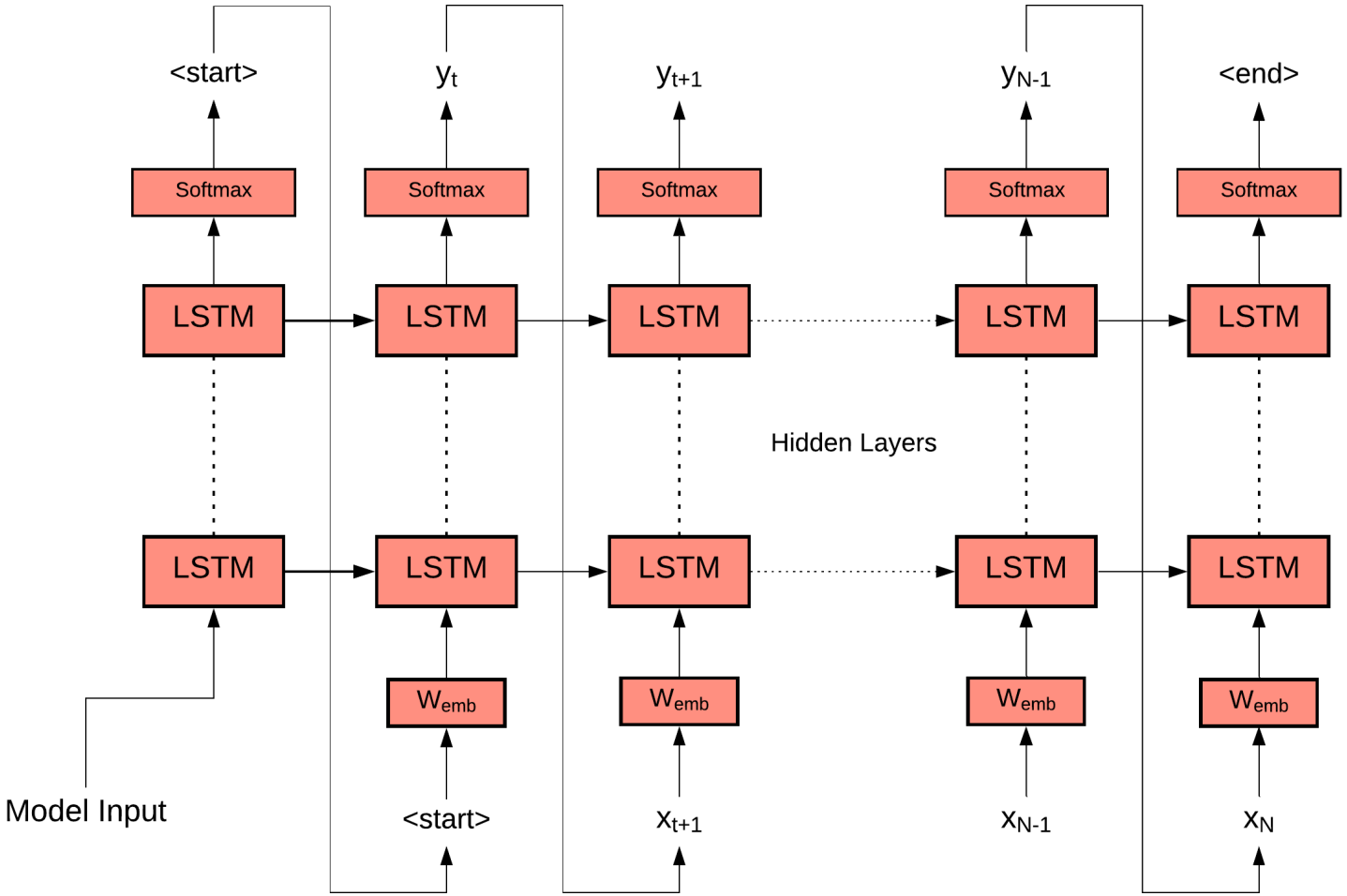
What does that look like?







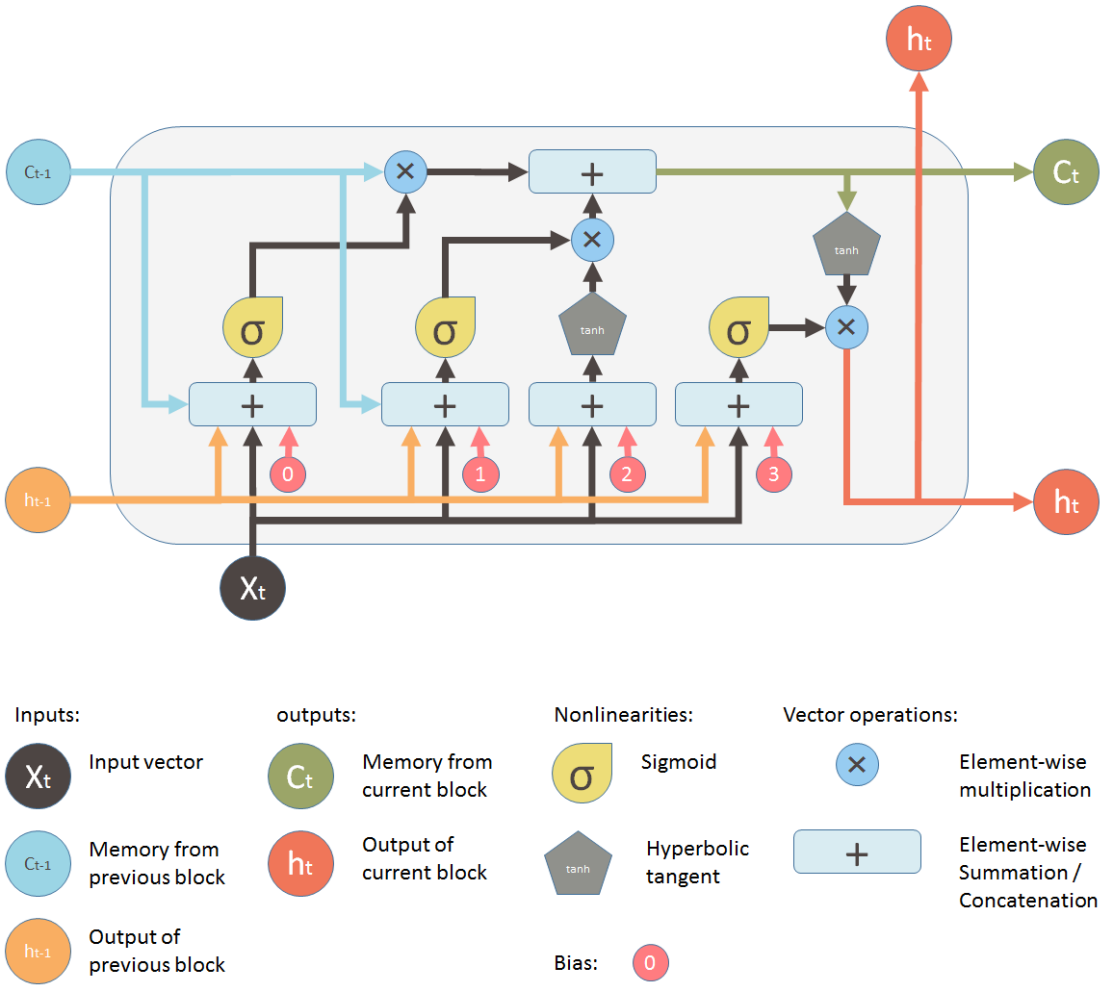
Recurrent Neural Networks



Why Recurrent Networks?

1. FCNNs and CNNs lack *memory*
2. Additionally, not all problems can be converted into ones with fixed-length inputs/outputs
   a. Simple: Determine if a number is even. 1000010100 is even, 100011 is not
3. RNNs were invented for this purpose; each neuron or 'unit' can maintain information about previous inputs with its internal memory
4. Context matters; the structure of natural language, peptide **sequence**
5. Additionally, RNNs can model sequences having variable length, and share weights across time steps; the amount of time an RNN keeps its intermediate values is not decided a priori

LSTM: Long Short-Term Memory



Forward Propagation in an RNN

Algorithm 1: Forward Propagation in an LSTM

Require:

Input Weights: $\mathbf{W}_z, \mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o \in \mathbb{R}^{M \times N}$

Recurrent Weights: $\mathbf{R}_z, \mathbf{R}_i, \mathbf{R}_f, \mathbf{R}_o \in \mathbb{R}^{N \times N}$

Peephole Weights: $\mathbf{p}_i, \mathbf{p}_f, \mathbf{p}_o \in \mathbb{R}^{N}$

Bias Weights: $\mathbf{b}_z, \mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o \in \mathbb{R}^{N}$

procedure FORWARD PASS

$\mathbf{z}_t = g(\mathbf{W}_z \mathbf{x}_t + \mathbf{R}_z \mathbf{y}_{t-1})$ ▷ Block Input

$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{R}_i \mathbf{y}_{t-1} + \mathbf{p}_i \odot \mathbf{c}_{t-1} + \mathbf{b}_i)$ ▷ Input Gate

$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{R}_f \mathbf{y}_{t-1} + \mathbf{p}_f \odot \mathbf{c}_{t-1} + \mathbf{b}_f)$ ▷ Forget Gate

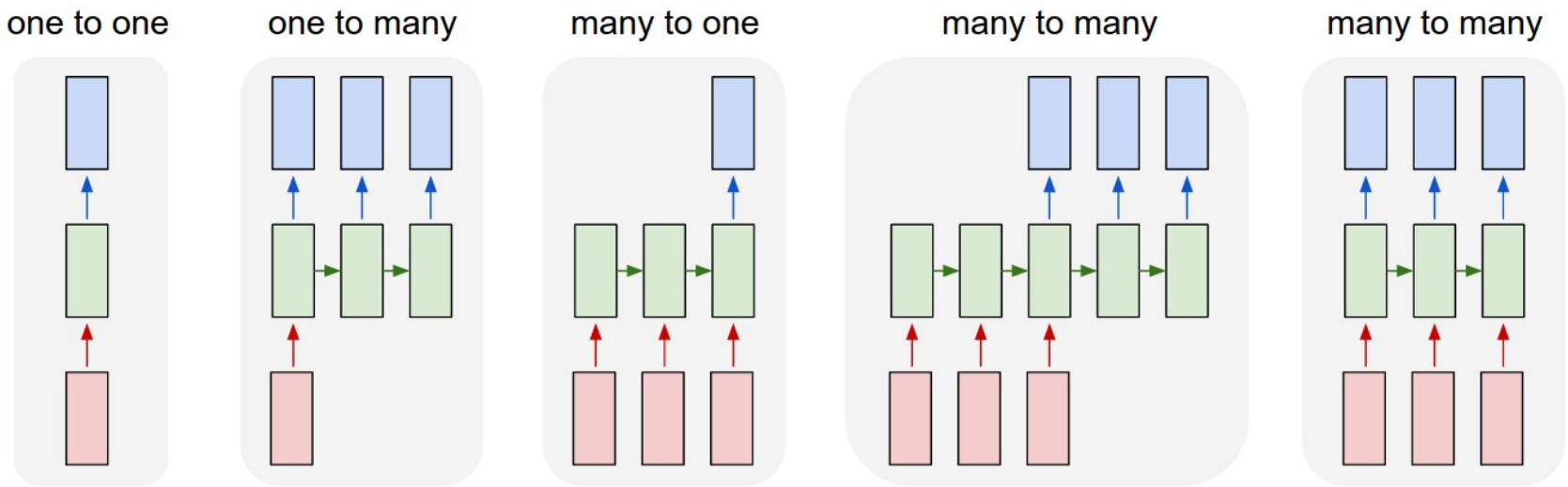
$\mathbf{c}_t = \mathbf{z}_t \cdot \mathbf{i}_t + \mathbf{c}_{t-1} \cdot \mathbf{f}_t$ ▷ Cell

$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{R}_o \mathbf{y}_{t-1} + \mathbf{p}_o \odot \mathbf{c}_t + \mathbf{b}_o)$ ▷ Output Gate

$\mathbf{y}_t = h(\mathbf{c}_t) \cdot \mathbf{o}_t$ ▷ Block Output

end procedure

Different RNNs And Their Purposes

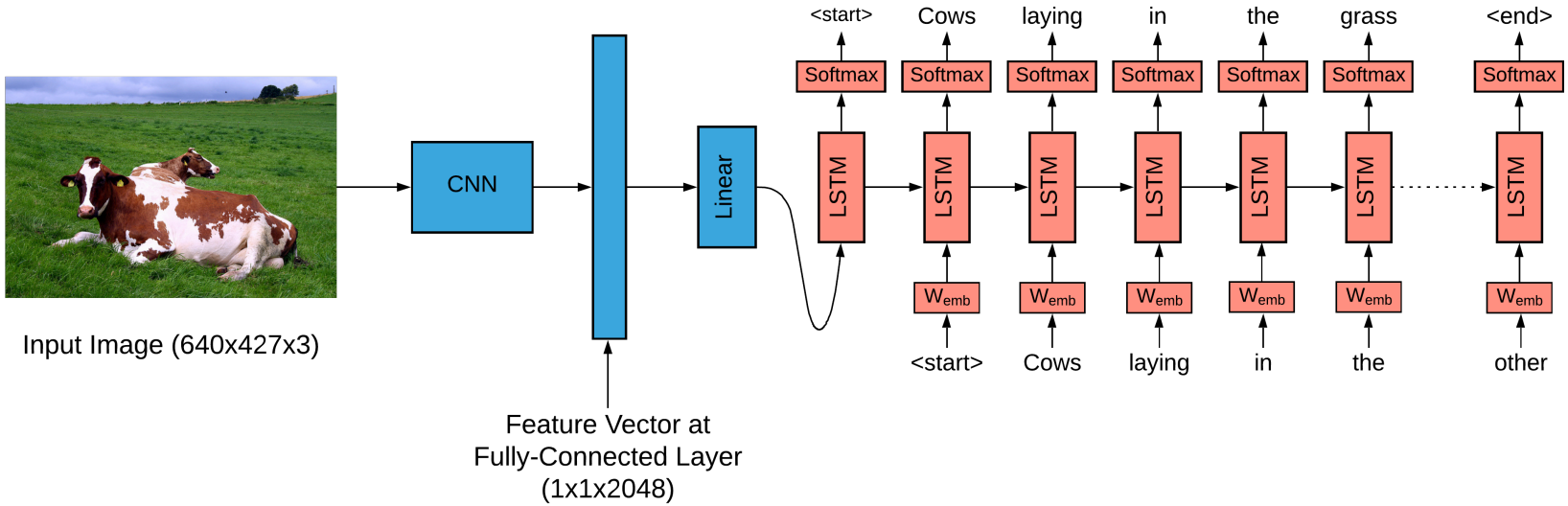


i. One-to-one: Vanilla FCNN
ii. One-to-many: Sequence Output (image captioning)
iii. Many-to-one: Sequence input (sentiment analysis)
iv. Many-to-many: Sequence input & output (machine translation)
v. Many-to-many: Synced sequence input/output (video frame labeling)

How does this help?

1. Image Captioning: take some input image and output a natural language description
2. Finds uses in many different disciplines:
   a. MRI Scans: Ground-truth can be time consuming and varies between physicians. Automated captioning of MRI scans allows for a speed up of this process (augmentation)
   b. Automated diagnosis: arrhythmia etc
   c. Image alternate text: People who are blind benefit from text descriptions of images commonly hard-coded in HTML
3. Current 'state of the art' involves some mix of RNN/CNN or some derivative network

An Input Image → Caption



Caption: Cows laying in the grass

Other Types of Networks and 'AI'

1. Generative Adversarial Networks (Goodfellow et al. 2014): Two networks (a generative network G and a discriminative network D) play a specialized version of minimax:

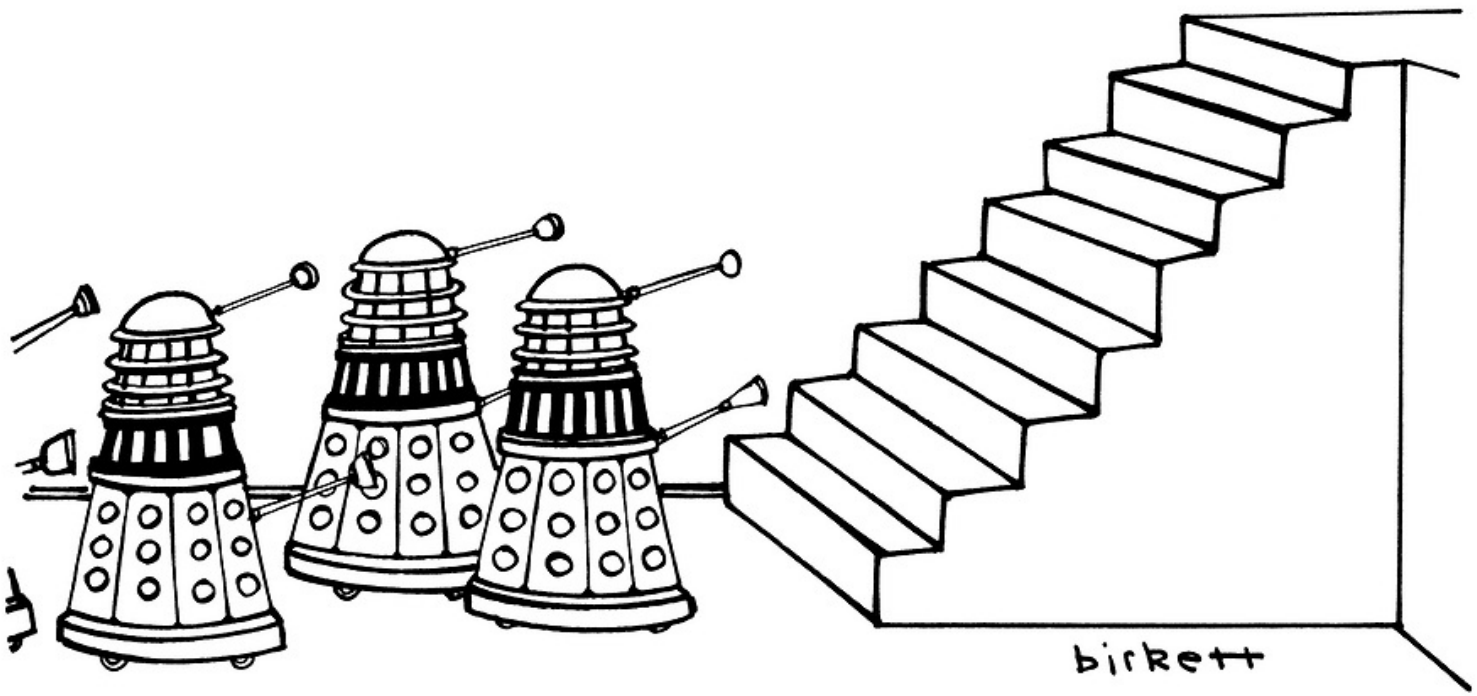
$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[log(D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[log(1 - D(G(\mathbf{z})))]$$

2. Deep Residual Networks (He et al. 2015): Optimize some network output $\mathcal{F}(x) := \mathcal{H}(x) + x$ where $\mathcal{H}(x)$ is the network output of the input x.
2. Reinforcement Learning: Evaluative rather than instructive feedback; a Markov Decision Process in which the 'agent' interacts with the environment to maximize some sort of scalar reward. (Sutton/Barto Book)

Interested in Learning More?

1. Stanford CS230: Deep Learning (Free!)
2. Coursera: Deep Learning Specialization
3. Michael Nielsen: Neural Networks and Deep Learning
4. 3Blue1Brown Youtube Series: Neural Networks
5. Github
6. ....Books!

That's all!



"Well, this certainly buggers our plan to conquer the Universe."