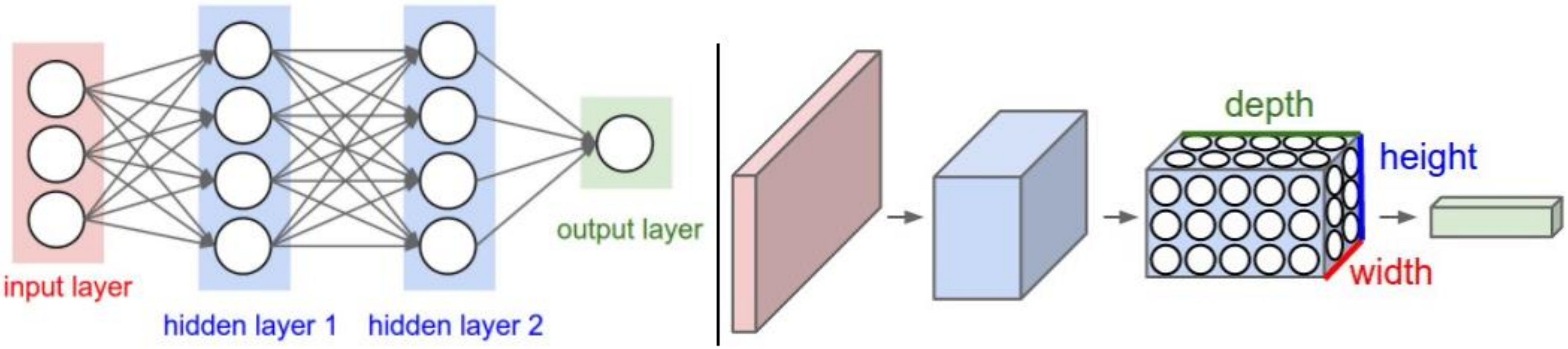
Session 6

- Recap of Session 5 - CNN
- Recurrent Neural Networks – RNN
- General Recap

Session 6

- Recap of Session 5 - CNN
- Recurrent Neural Networks – RNN
- General Recap

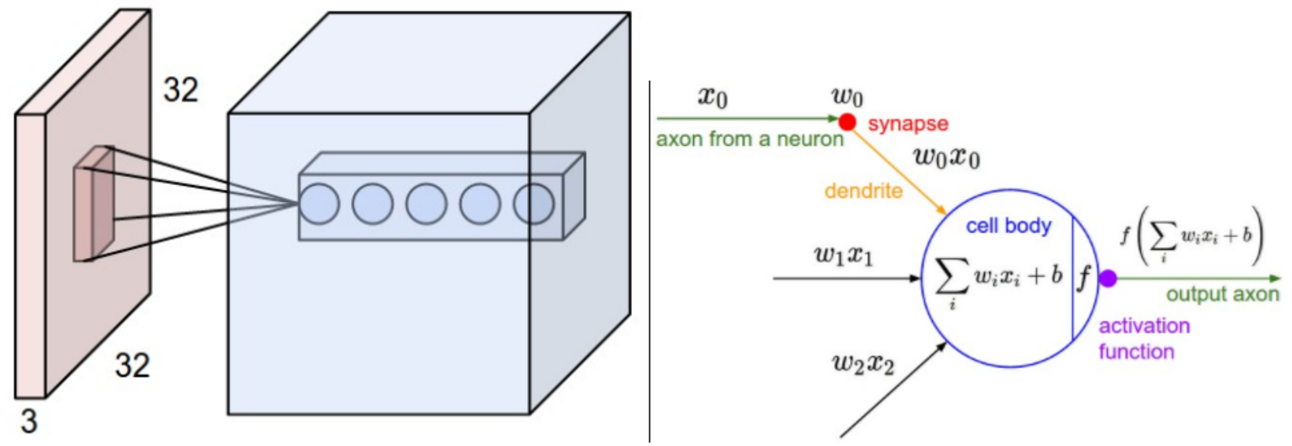
What is Convolutional NN



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

Demo

- http://cs231n.github.io/convolutional-networks/

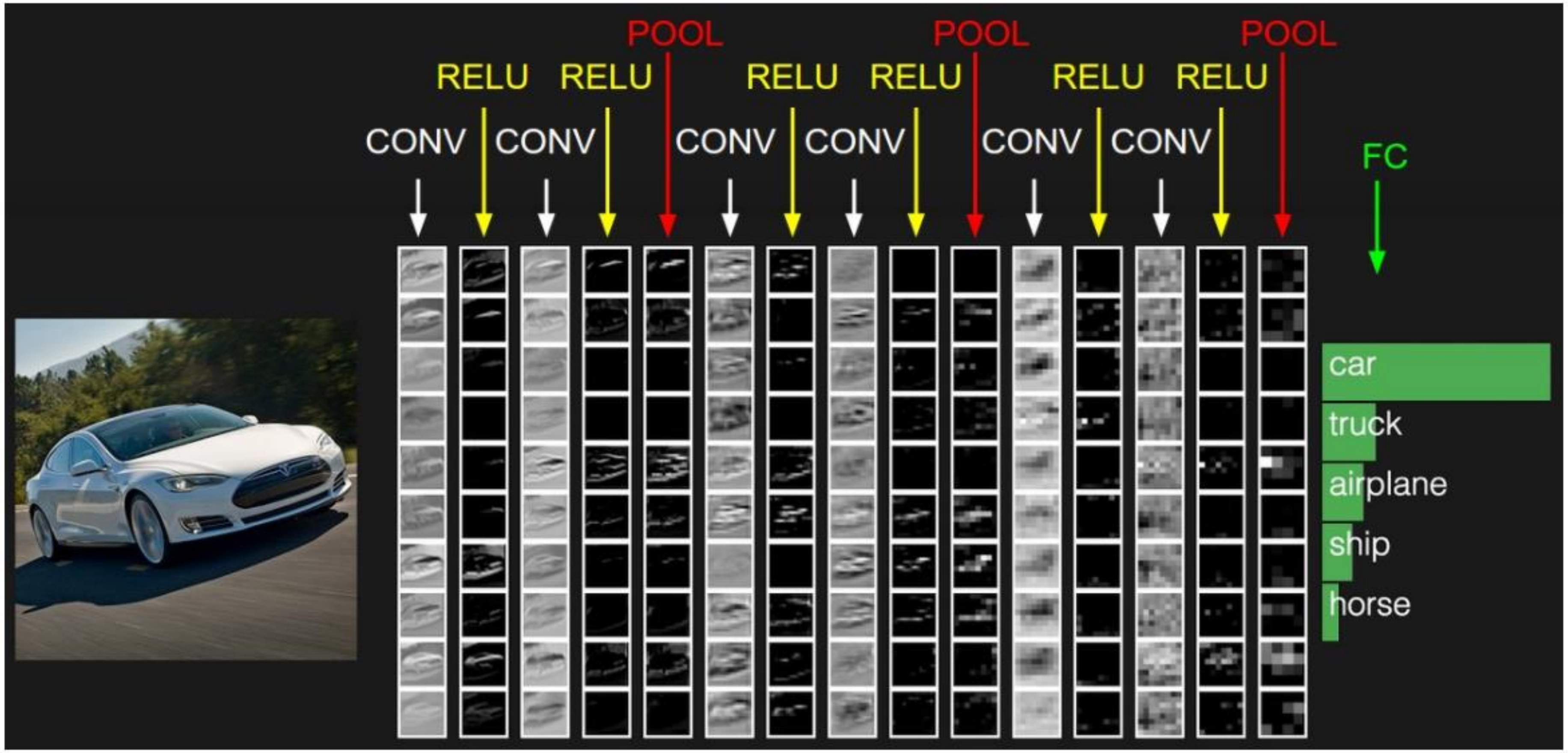


Left: An example input volume in red (e.g. a 32x32x3 CIFAR-10 image), and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels). Note, there are multiple neurons (5 in this example) along the depth, all looking at the same region in the input - see discussion of depth columns in text below. **Right:** The neurons from the Neural Network chapter remain unchanged: They still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially.

Demo

- http://cs231n.github.io/convolutional-networks/
  - Parameter sharing
  - Filters
    - Size
    - Depth
    - Strides
    - Number of filters
    - Strategy to handle edges (same/valid)

What is each layer doing?

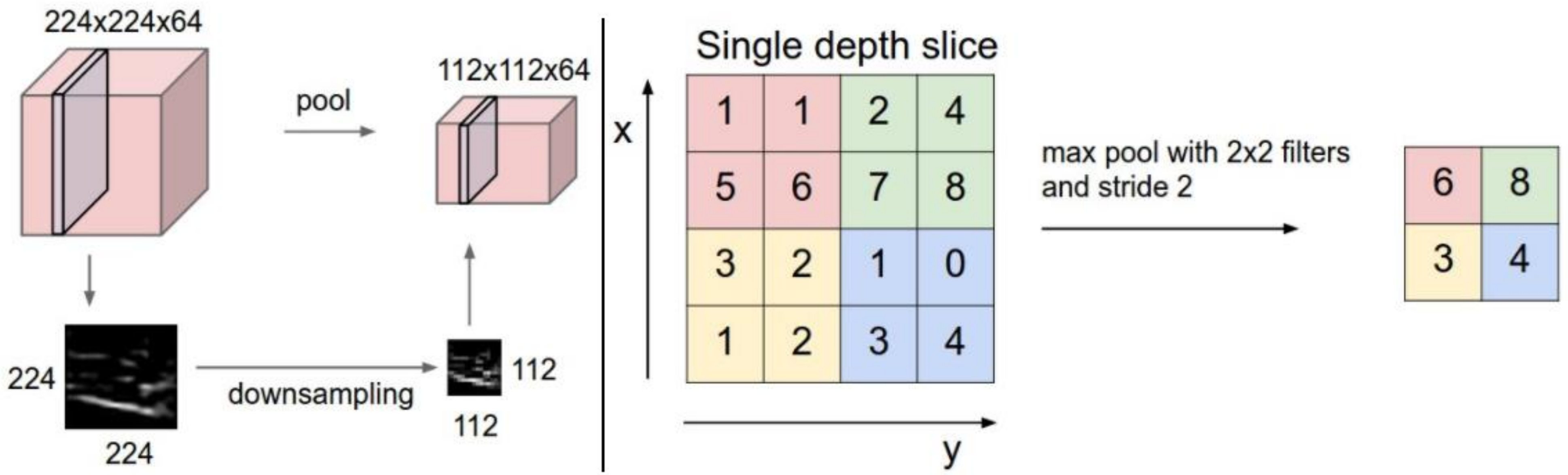


The activations of an example ConvNet architecture. The initial volume stores the raw image pixels (left) and the last volume stores the class scores (right). Each volume of activations along the processing path is shown as a column. Since it's difficult to visualize 3D volumes, we lay out each volume's slices in rows. The last layer volume holds the scores for each class, but here we only visualize the sorted top 5 scores, and print the labels of each one. The full web-based demo is shown in the header of our website. The architecture shown here is a tiny VGG Net, which we will discuss later.

New types of layers

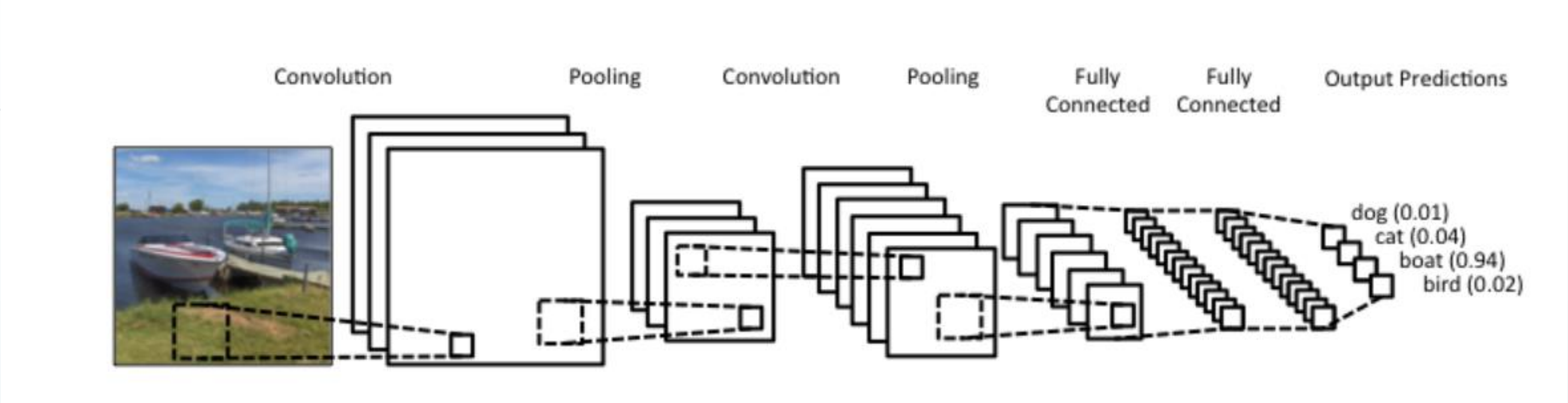
- Convolutional Layer – we just saw
- Dropout layers – we talked in previous section
- Activation functions/layers
  - RELU – most common for hidden layers
  - Softmax – as output activation - most of classification problems
- Polling layers
- Flatten and then FC (fully connected layers)

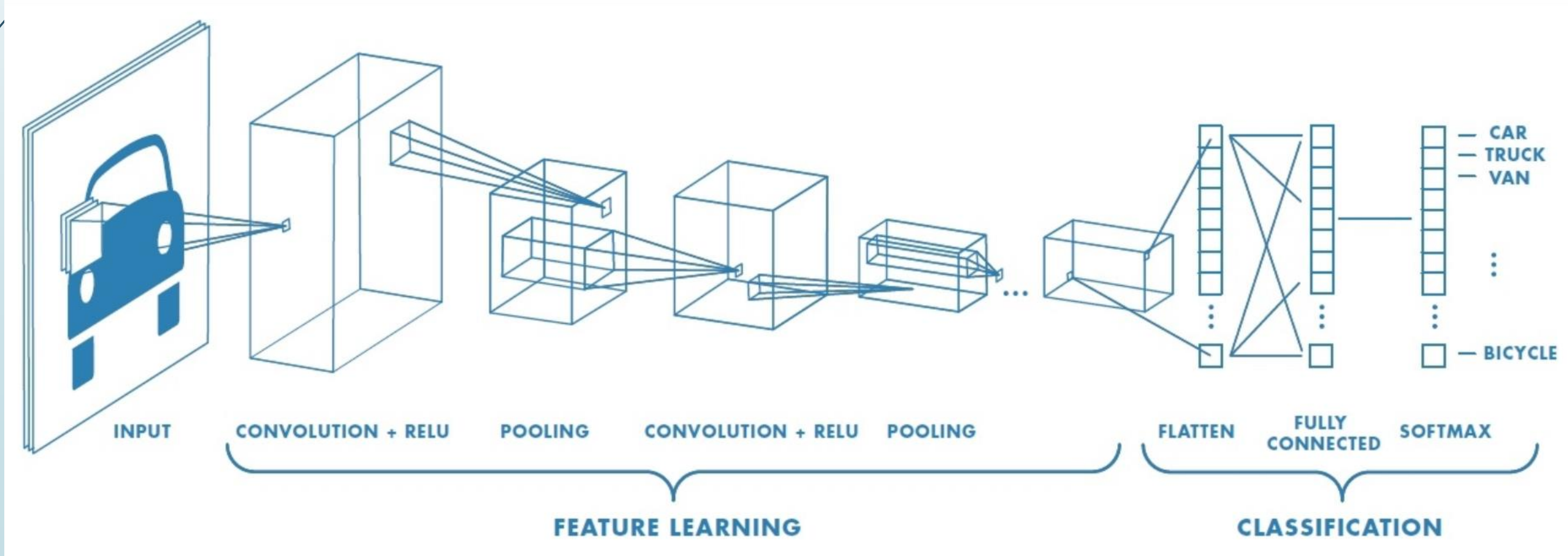
Pooling layers



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

Flatten followed by FC





Why are NN so popular and powerful?

They learn features on their own.

No need for extensive feature engineering that was required earlier

Session 6

- Recap of Session 5 - CNN
- Recurrent Neural Networks – RNN
- General Recap

Examples of sequence data

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	∅	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

Basic RNN Model

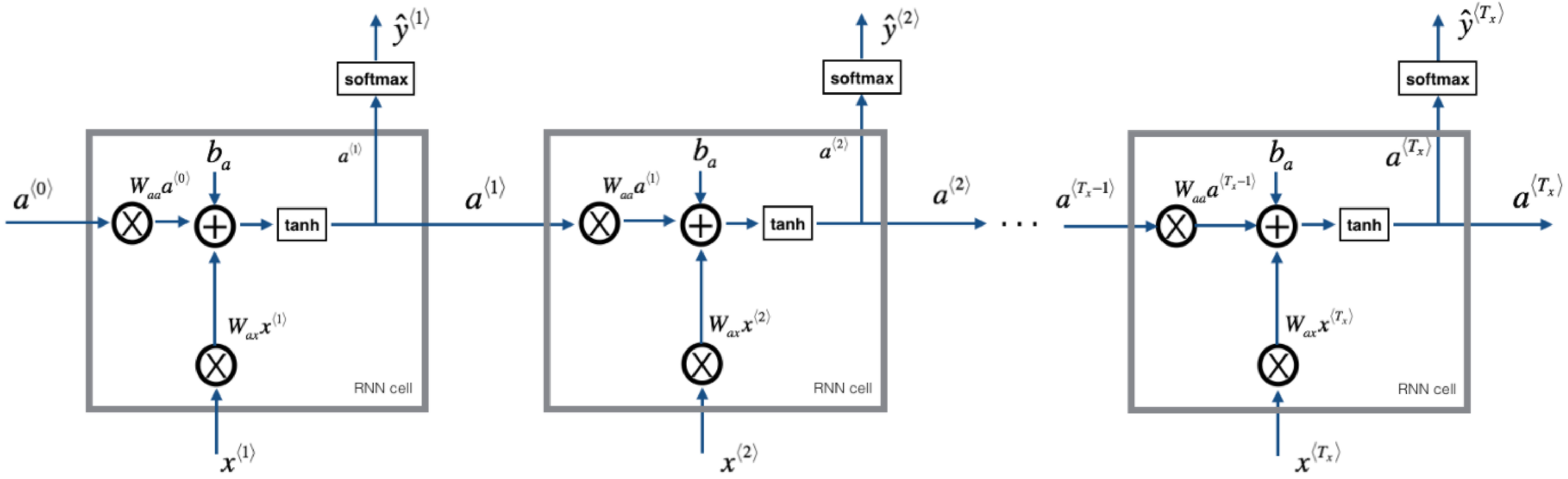
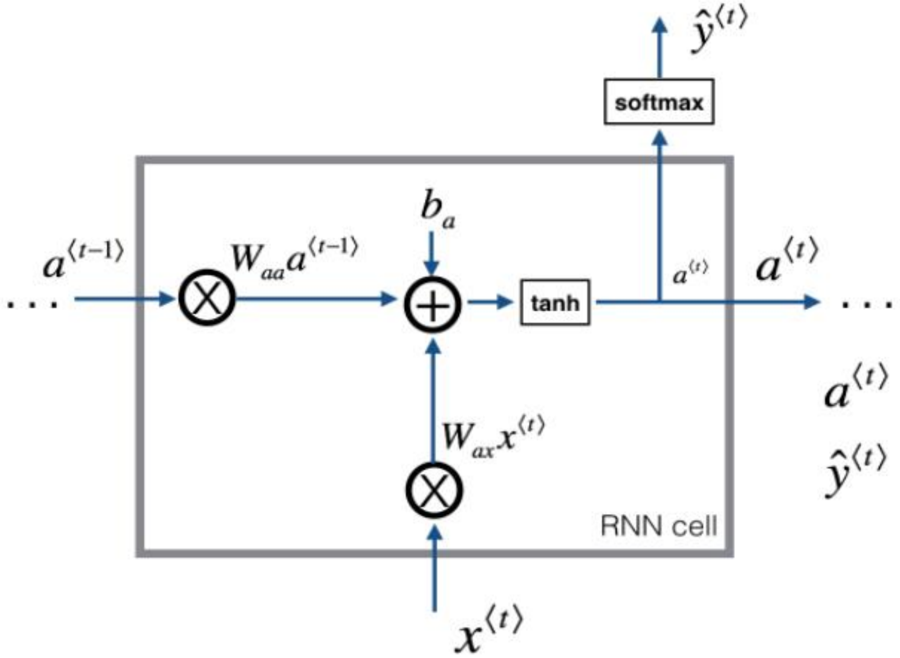


Figure 1: Basic RNN model

Basic RNN Cell

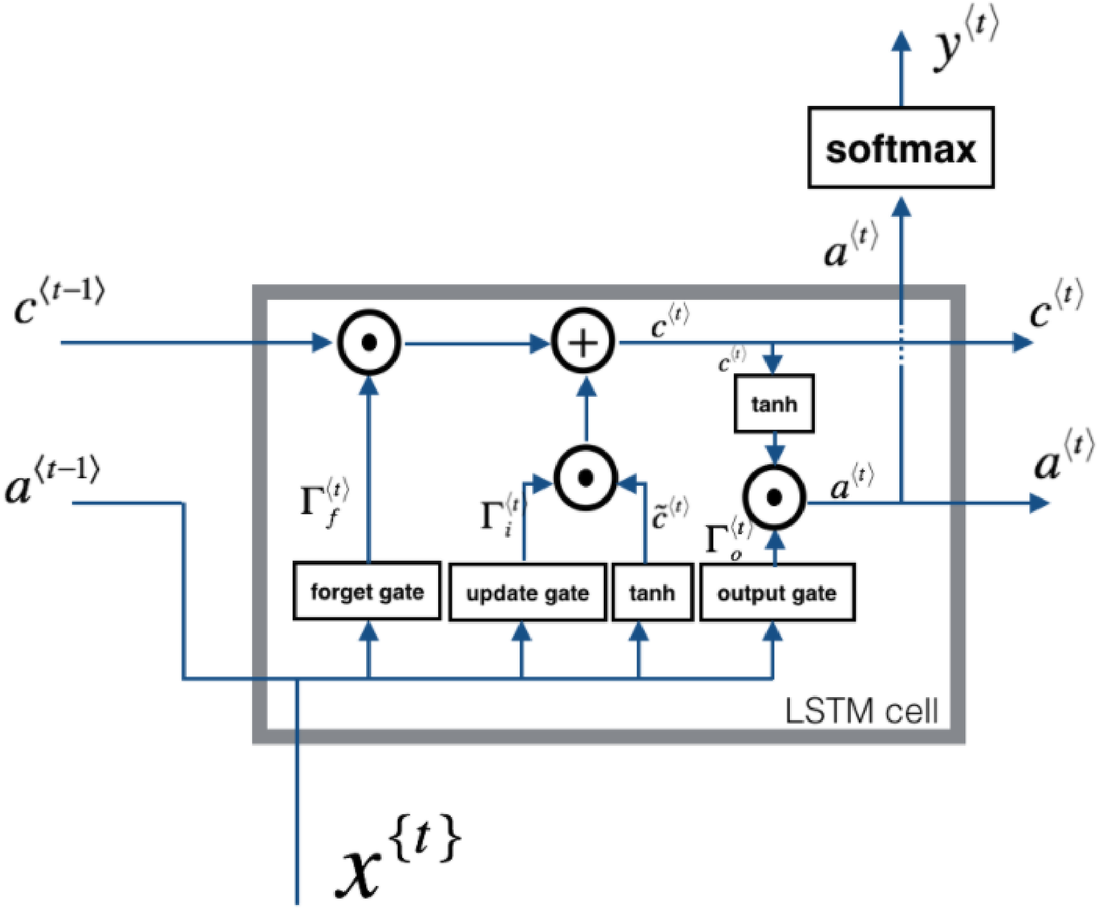


$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = softmax(W_{ya}a^{\langle t \rangle} + b_y)$$

Figure 2: Basic RNN cell. Takes as input $x^{\langle t \rangle}$ (current input) and $a^{\langle t-1 \rangle}$ (previous hidden state containing information from the past), and outputs $a^{\langle t \rangle}$ which is given to the next RNN cell and also used to predict $y^{\langle t \rangle}$

LSTM Cell



. . .

$$\Gamma_f^{\langle t \rangle} = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$$

$$\Gamma_u^{\langle t \rangle} = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$

$$\tilde{c}^{\{t\}} = \tanh(W_C[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_C)$$

$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle}$$

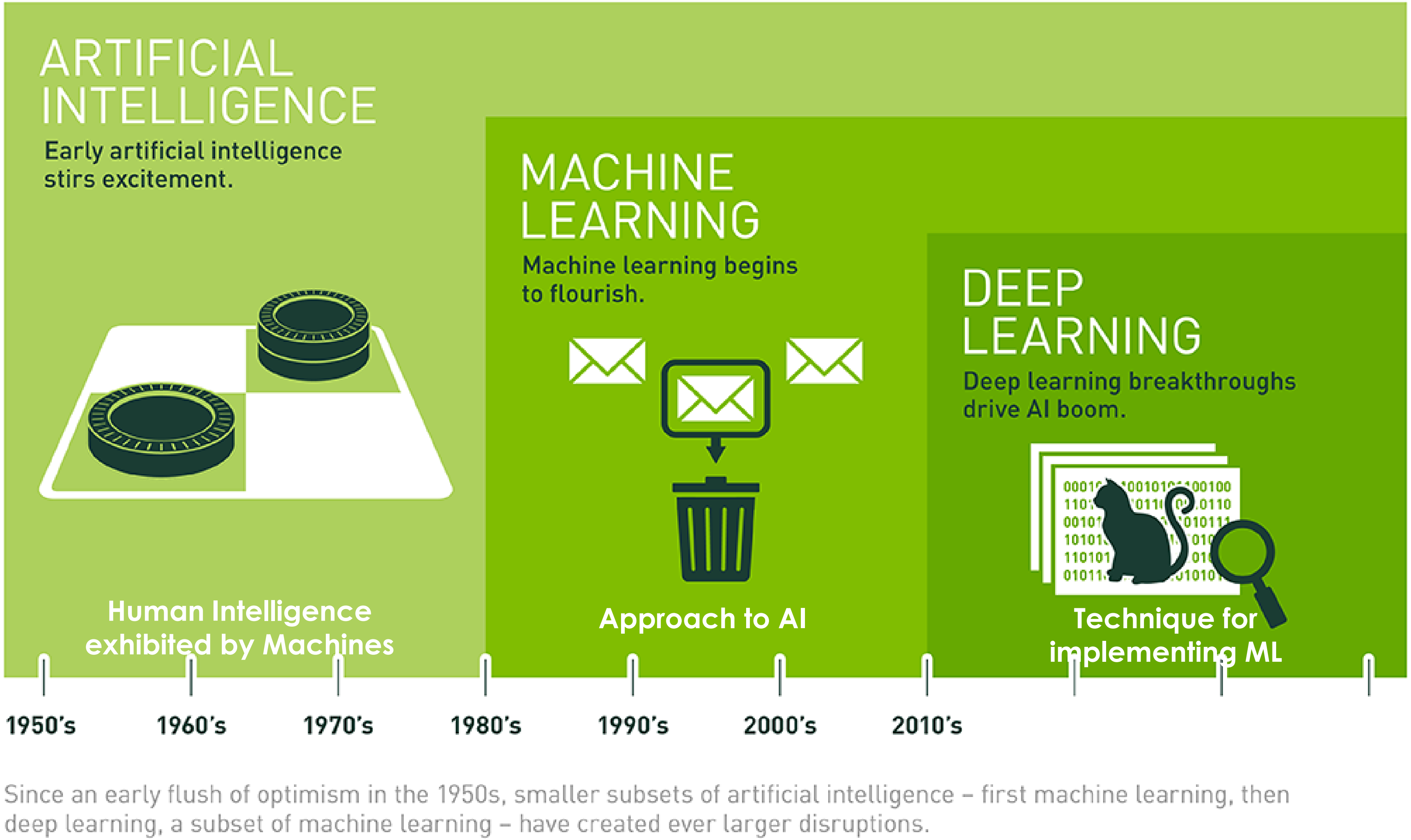
$$\Gamma_o^{\langle t \rangle} = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$$

$$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} \circ \tanh(c^{\langle t \rangle})$$

Figure 4: LSTM-cell. This tracks and updates a "cell state" or memory variable $c^{\langle t \rangle}$ at every time-step, which can be different from $a^{\langle t \rangle}$.

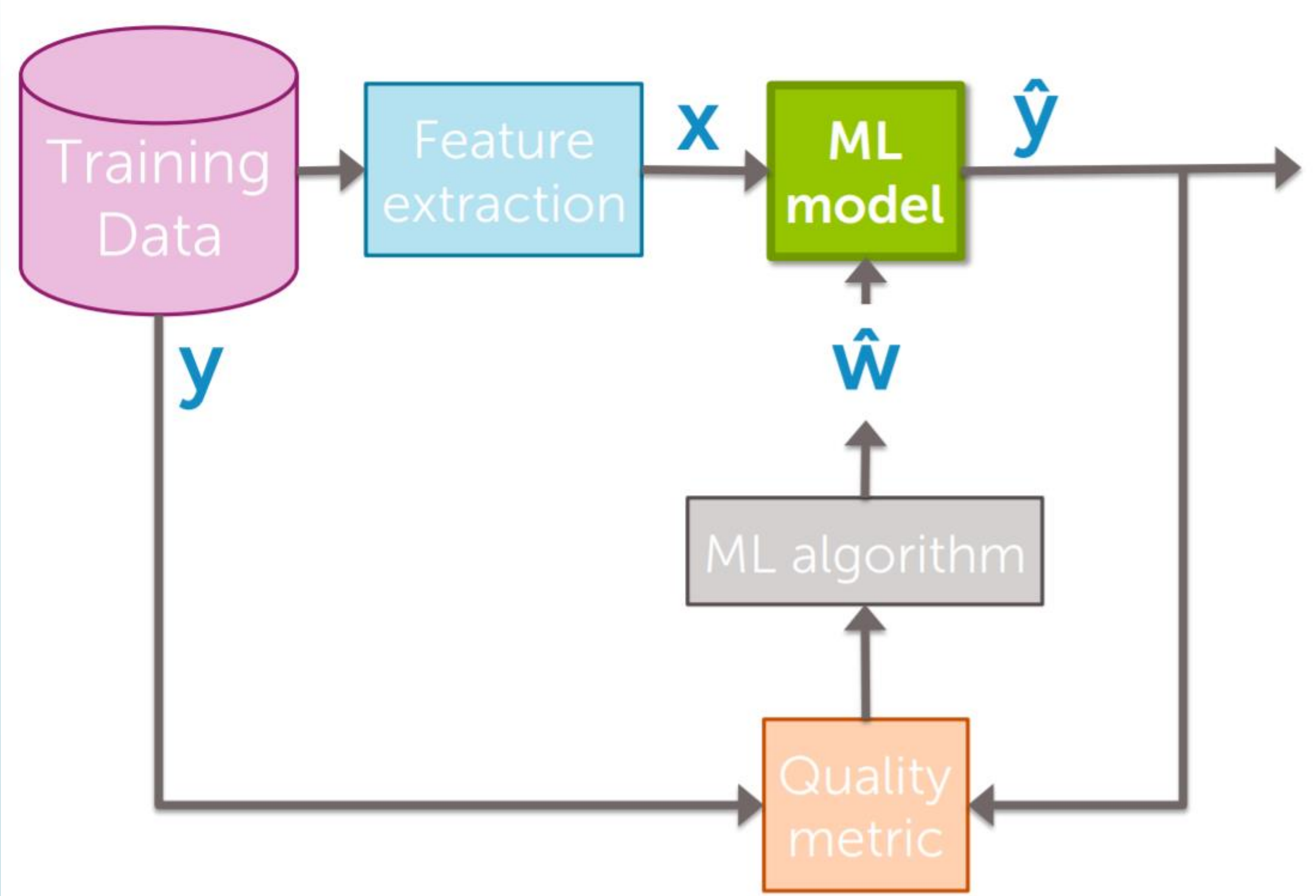
Session 6

- Recap of Session 5 - CNN
- Recurrent Neural Networks – RNN
- General Recap



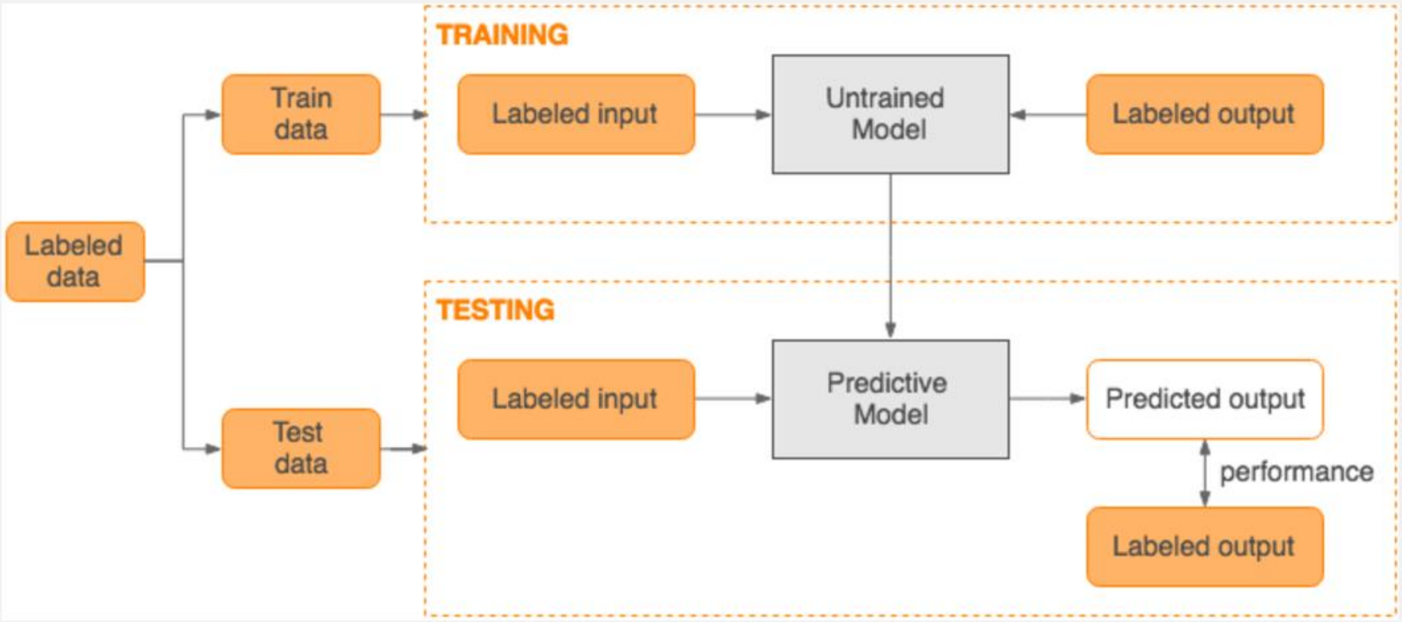
Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

So what are we trying to do?



What is ML – Step 1

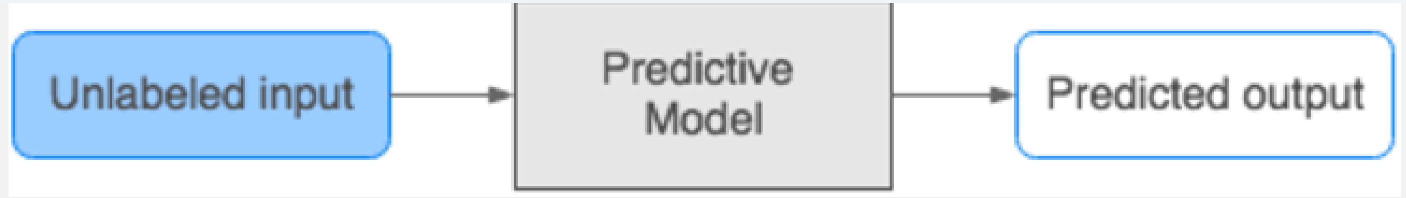
- We take a sample of data and split it into training and testing set
- Use training data to build predictive model
- Use testing data to check the quality of model



Splitting labeled data into training and testing sets

What is ML – Step 2

- Use predictive model to future data

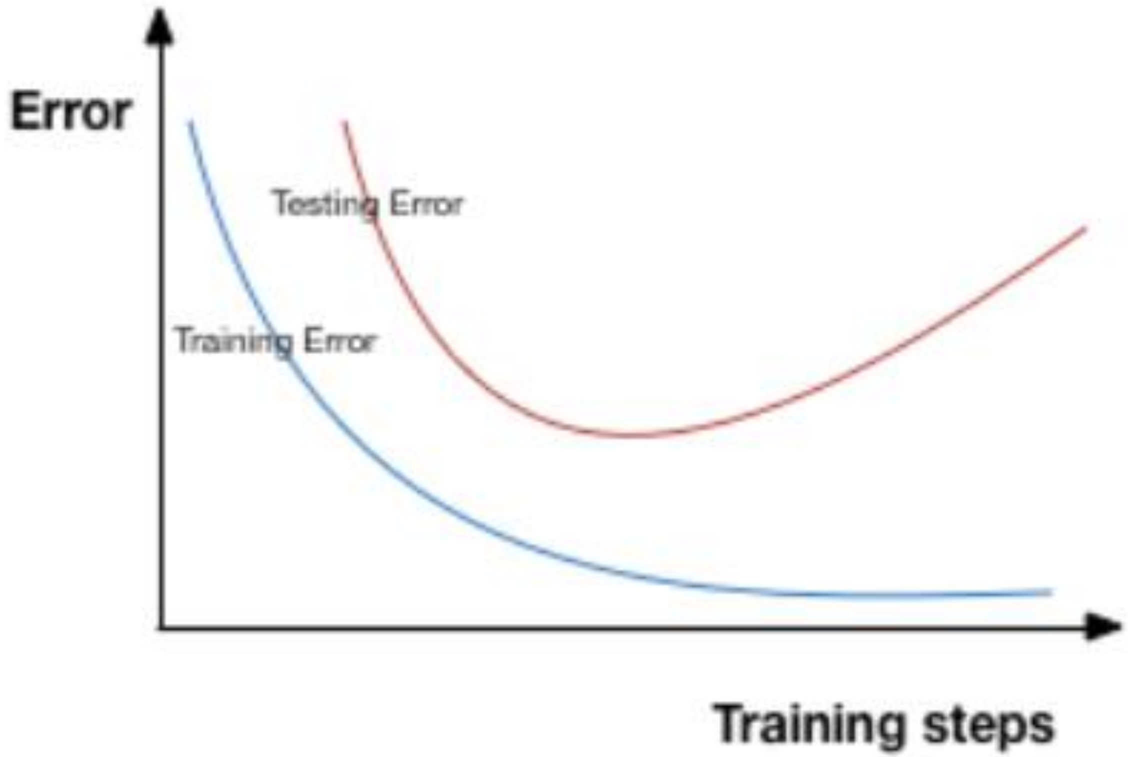


- We do not see complete data during training – only a sample
  - We want to have a way so that "training error" stays close to "testing error"
  - And "testing error" close to "actual error" of the model if it was evaluated on complete data

Theory provided by Probability

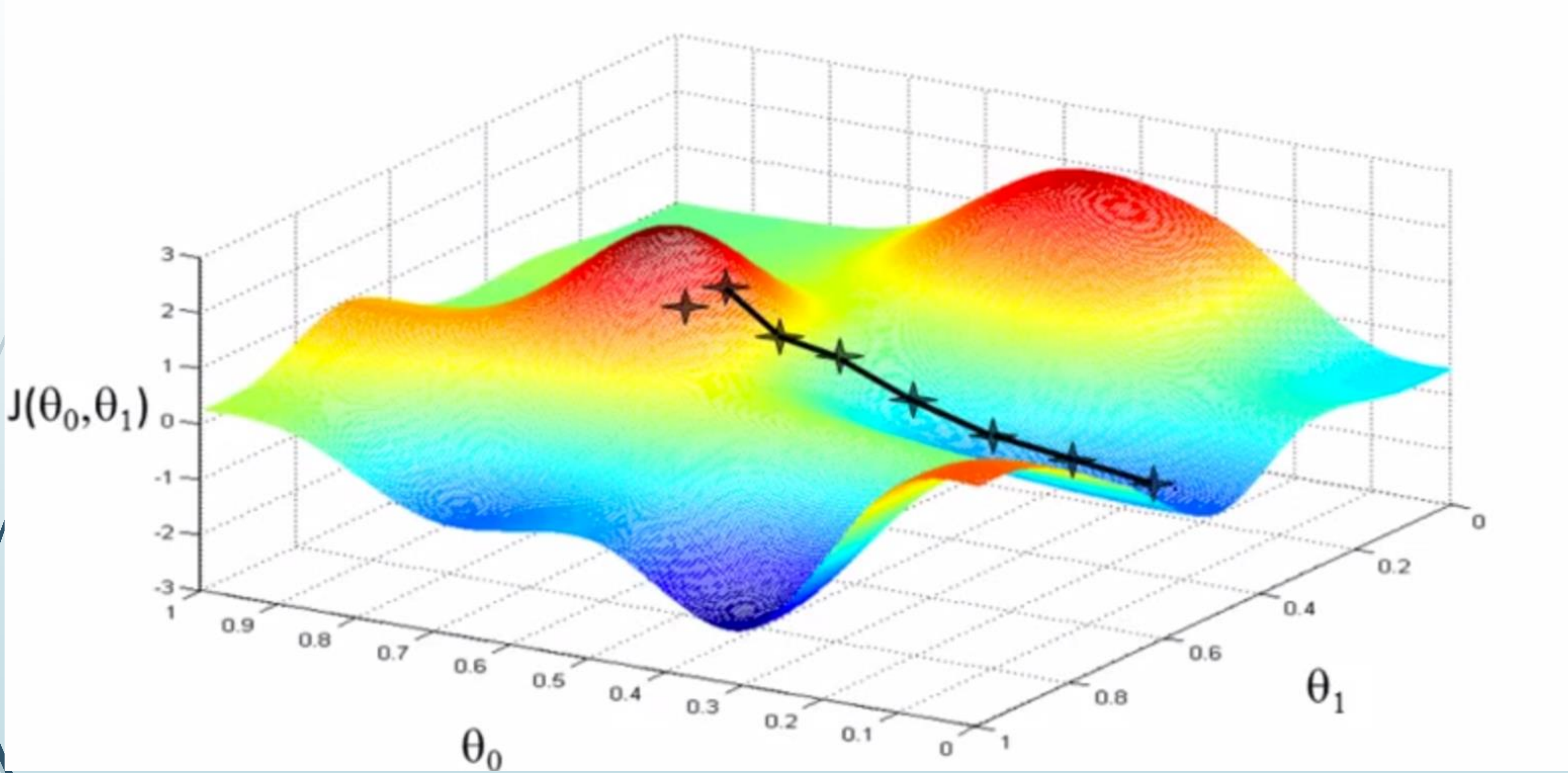
Please refer Appendix A for a short refresher on Probability

Training and Testing Error



Overfitting symptom: Testing error >> Training error

Gradient Descent



Gradient Descent - ALgorithm

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

}

Correct: Simultaneous update

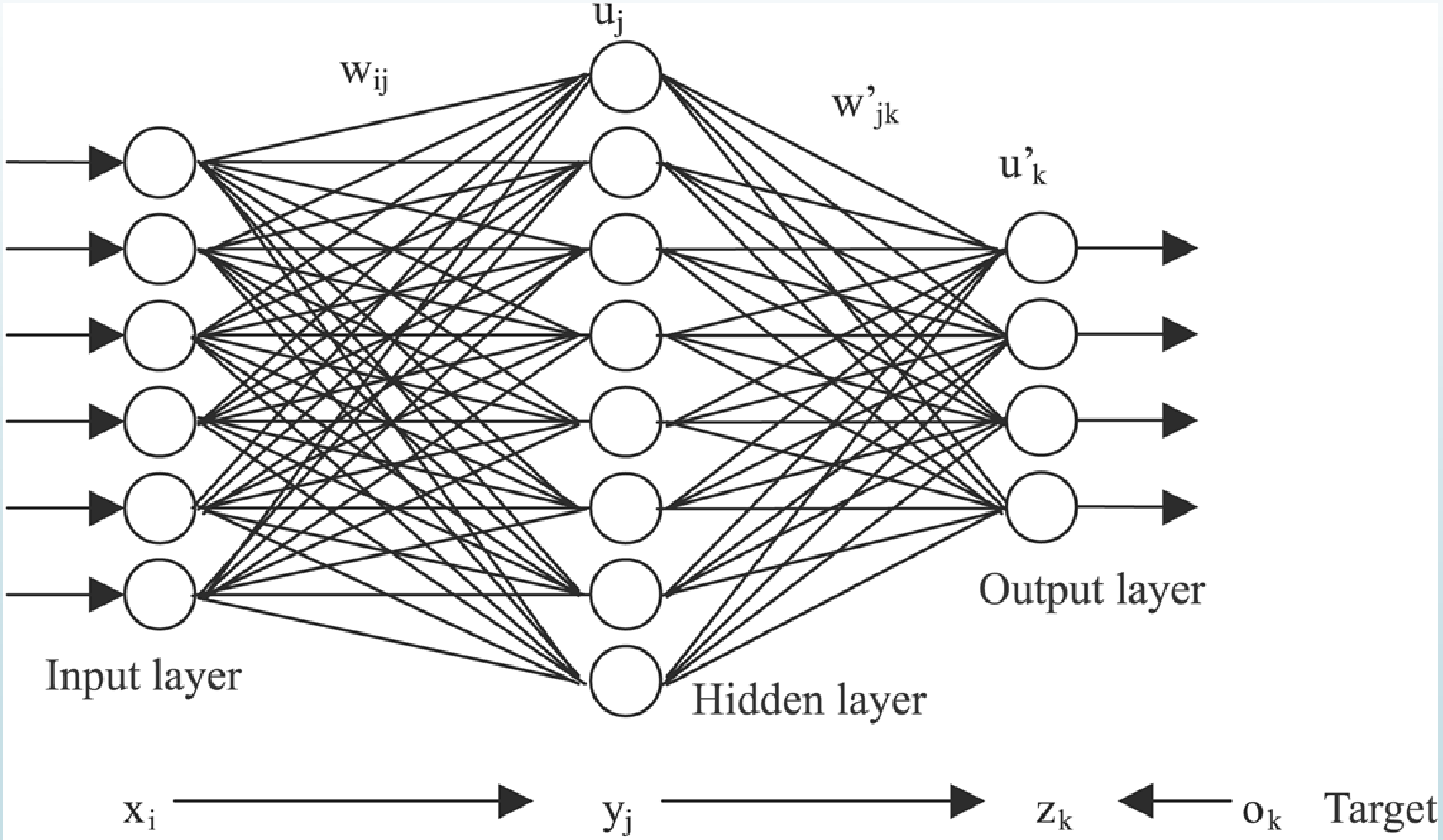
$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

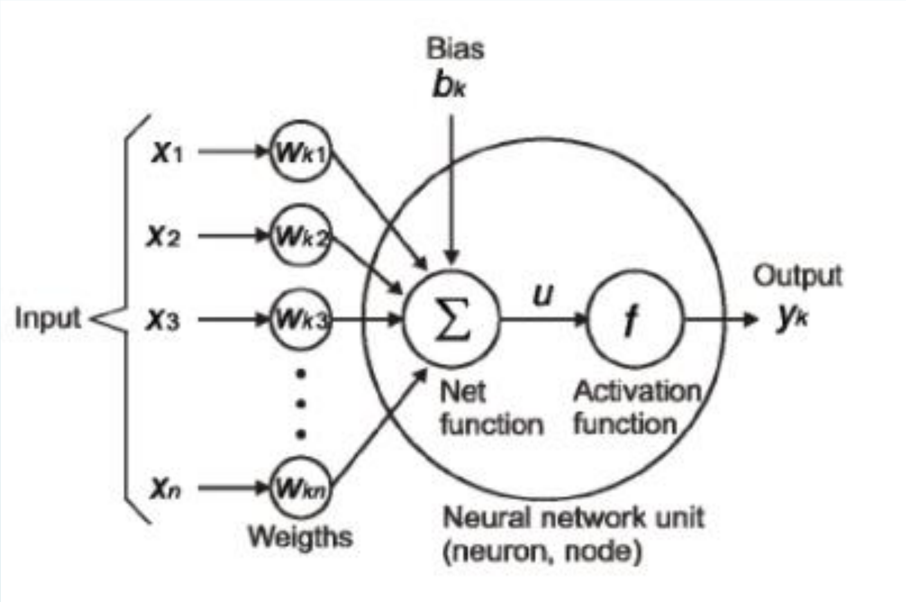
$\theta_0 := \text{temp0}$

$\theta_1 := \text{temp1}$

What is a neural network

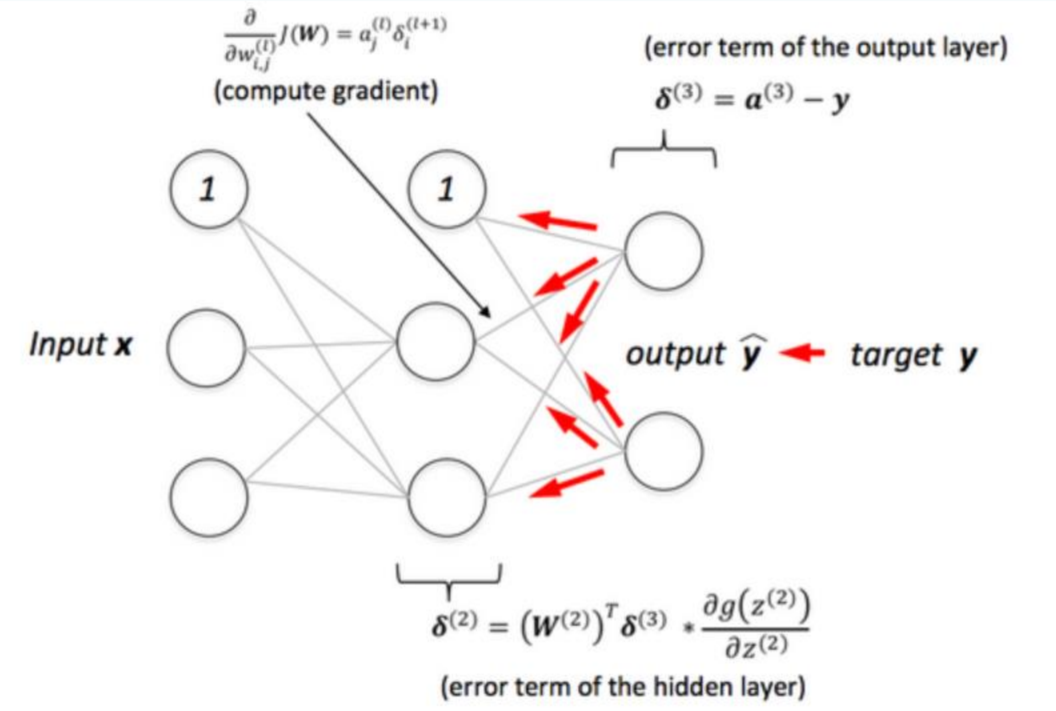


What is a neural network



Back Propagation

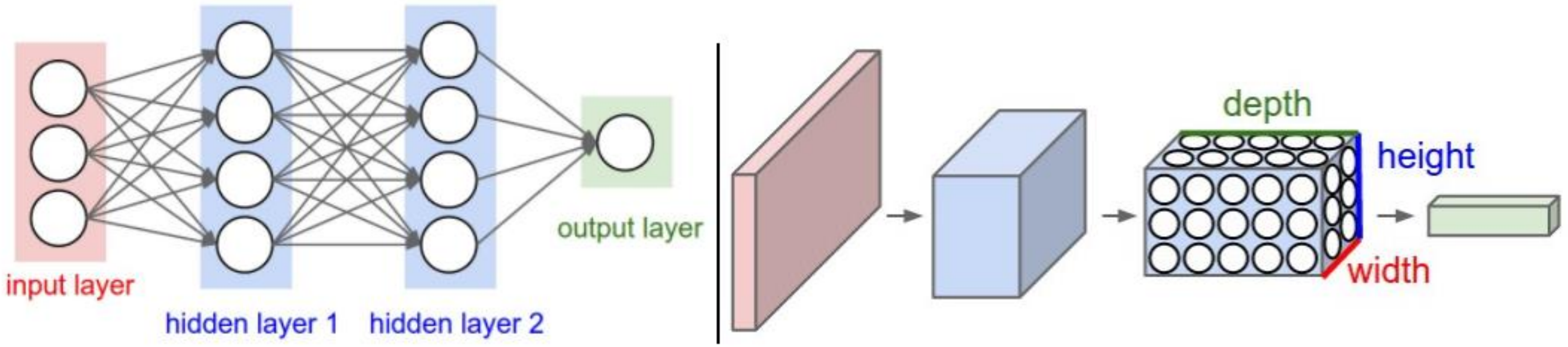
- Apply chain rule of calculus and propagate error from output layer to input layer.
- It is like a flow graph you flow the error back and try to find the impact of each variable (weight, bias) on the total error
- The most popular algorithm to train neural networks
- "Tensorflow" from Google allows you to build the graph of neural network and apply chain rule with partial derivatives in a systematic and efficient manner.
- This field is also known as "Automatic differentiation"



Tensorflow and Keras

- Tensorflow – open source framework from Google to build and train Neural network models.
- Caffe from Facebook is another very popular framework
- MSFT: Microsoft Cognitive Toolkit—previously known as CNTK
- And many more
  - They are try to do the same – build, train deep learning models but vary in the approach they take.
- Keras: is a higher level framework that can sit on top of Caffe, Tensorflow. It abstarcts the nuts and bolts of underlying framework. Same Keras code will work even if you swap the lower level framework.

What is Convolutional NN



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

Basic RNN Model

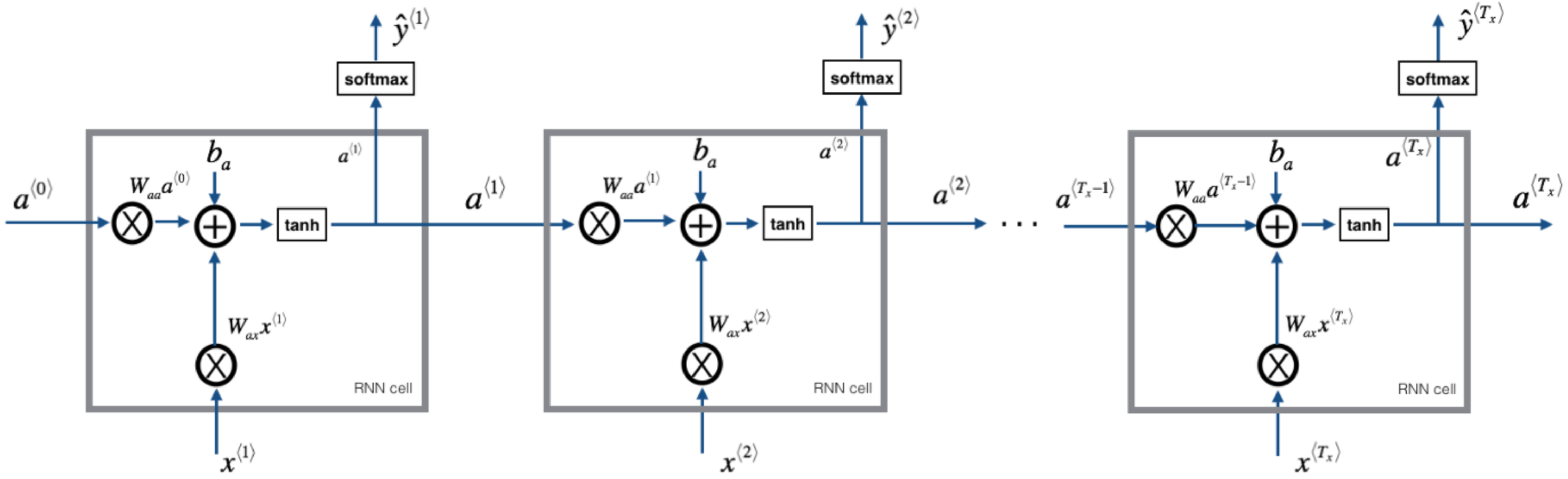


Figure 1: Basic RNN model

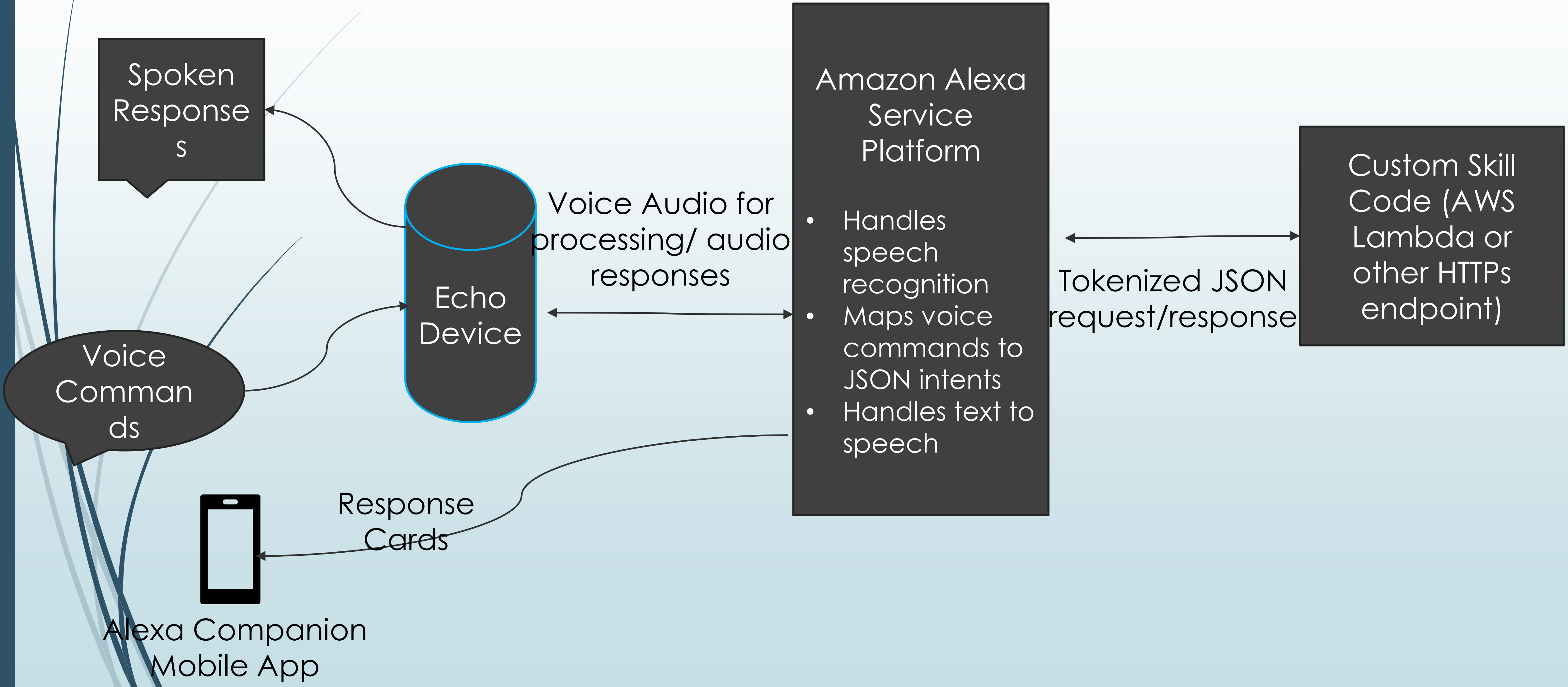
ChatBots

UI
slack, custom, Google, Messenger

Dialogflow
Entities, intents, context

Services
Fulfillment Application

Anatomy of Alexa Custom Skill



Resources - ML

- Machine Learning - MOOCs
  - https://www.coursera.org/learn/machine-learning - One of the best courses to get to know about machine learning
  - https://www.coursera.org/specializations/machine-learning - 4 course specialization. I really like the engaging style of lectures. Exercises would need you to install Anaconda as well as some additional framework (graphlib)
  - https://in.udacity.com/nanodegree - You can go for paid Machine Learning Basics and Machine Learning Foundations Nano degrees. Udacity model is a great and helps you stay invested in learning with well defined deadlines plus community support
  - https://in.udacity.com/course/machine-learning--ud262 - Free course by Gerogia Tech – great course and a very conversational style of lectures. Downside not enough projects
  - https://in.udacity.com/course/machine-learning-unsupervised-learning--ud741 - by same profs – but Unsupervised learning
  - https://work.caltech.edu/telecourse.html - one of the best if you want to understand the theory and math behind machine learning. By CalTech

Resources – DL (Deep Learning)

- https://www.coursera.org/specializations/deep-learning - 5 course specialization by Andrew Ng – one of the best – covers some theory and lots of hands on exercise – he is really good at making complex things look simple
- https://www.coursera.org/specializations/aml - 7 course specialization with lots of math and tough hands-on. Also goes into the new exciting area of Bayesian approach to Learning. i.e. extension of basic ML to moe complex models.
- https://in.udacity.com/course/deep-learning-nanodegree-foundation--nd101 - a well structured course spread over 4-6 months. Udacity style – real deep dive on code and lots of community support

Resources – RL (Reinforcement Learning) and Deep RL

- https://in.udacity.com/course/reinforcement-learning--ud600 - By Georgia Tech. Have not done this course but by good profs
- http://ai.berkeley.edu/home.html - one the best courses on RL. It has not been updated some time but still very current
- https://sites.google.com/view/deep-rl-bootcamp/ - a very comprehensive hands-on course with video lectures and labs

Resources - Books

- https://work.caltech.edu/textbook.html - a short but very packed book from CalTech Prof
- http://www.deeplearningbook.org/ - one of the best book on deep learning – specially first 10 chapters – you will lot wiser even if you grasp 40-50%. Avoids complex maths
- ML – A probabilistic Perspective – Kevin Murphy - https://www.amazon.in/dp/B00AF1AYTQ/ref=dp-kindle-redirect?_encoding=UTF8&btkr=1 – great book to read for a very mathematical exploration of ML. Even first 3-4 chapters are good enough. No Indian edition and foreign edition is expensive.
- Bishop: Pattern Recognition and Machine Learning – another great book to read – similar to Kevin Murphy. No indian edition - https://www.amazon.in/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738/ref=sr_1_1?ie=UTF8&qid=1518960696&sr=8-1&keywords=bishop+machine+learning
- Two other good books on theory with free legitimate online versions
  - Elements of Statistical Learning - https://web.stanford.edu/~hastie/ElemStatLearn/
  - Bayesian Reasoning and Machine Learning - http://web4.cs.ucl.ac.uk/staff/D.Barber/pmwiki/pmwiki.php?n=Brml.HomePage