OPEN DATA SCIENCE CONFERENCE

#ODSC

@ODSC

London | September 19 - 22 2018

#ODSC 2018 - London

20.09.2018

The Path to Deep Learning with Tensorflow + Keras

Juliet Moreiro

Technical Evangelist (Microsoft)

Pablo Doval

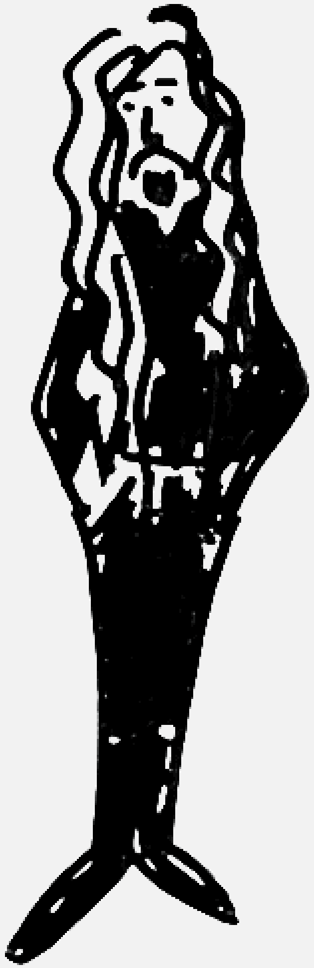
Data Pontifex (Plain Concepts)

Juliet Moreiro

TECHNICAL EVANGELIST @ MICROSOFT

Bio

@julietsvq



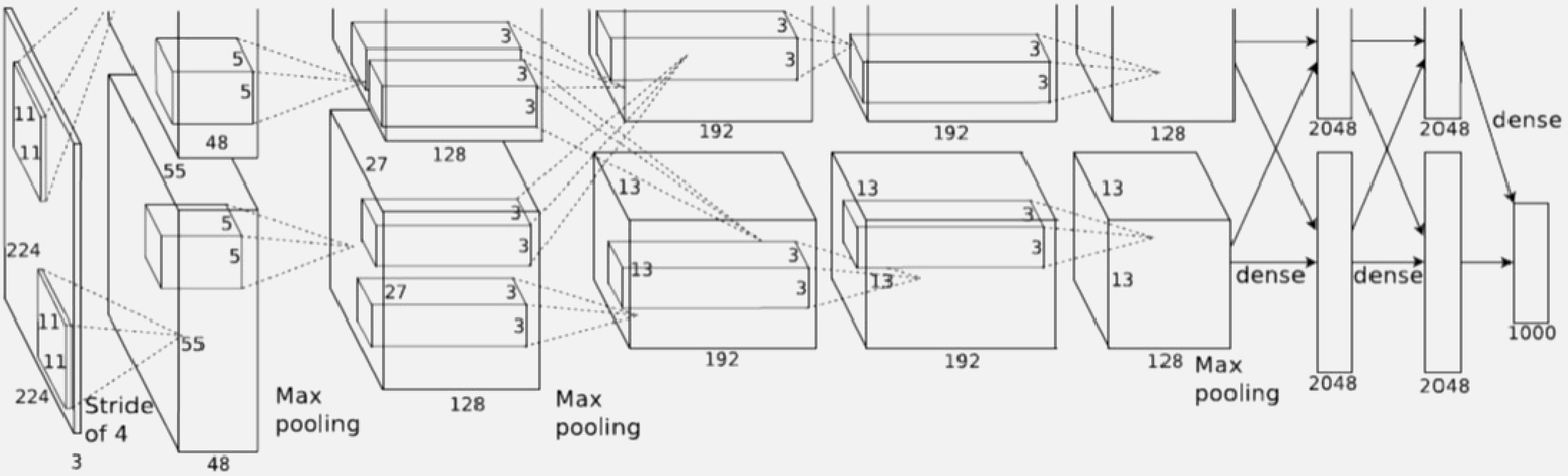
Pablo Doval

DATA PONTIFEX

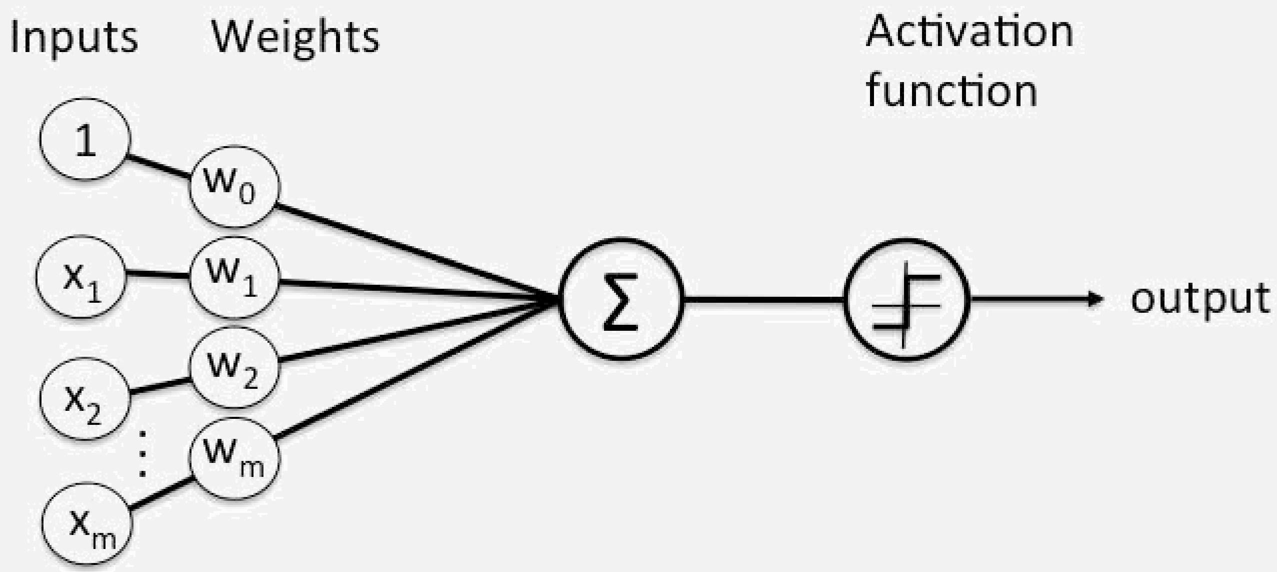
I work with code and data, but don't tell my mom; she thinks I'm a piano player in a whorehouse.

@PabloDoval

WHAT IS THIS THING?

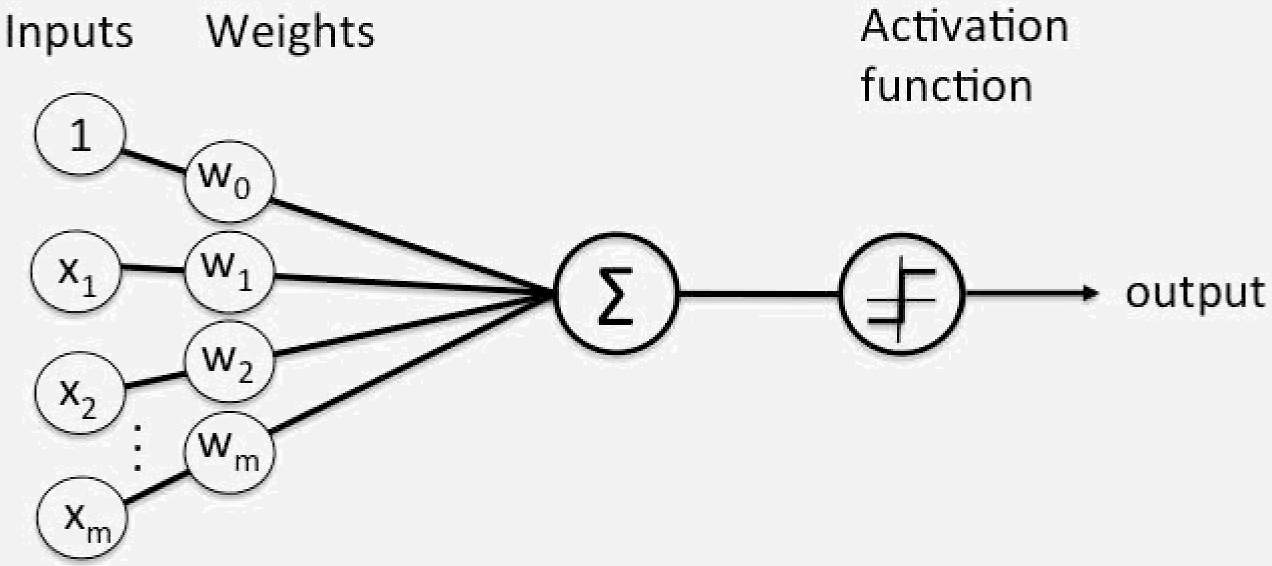


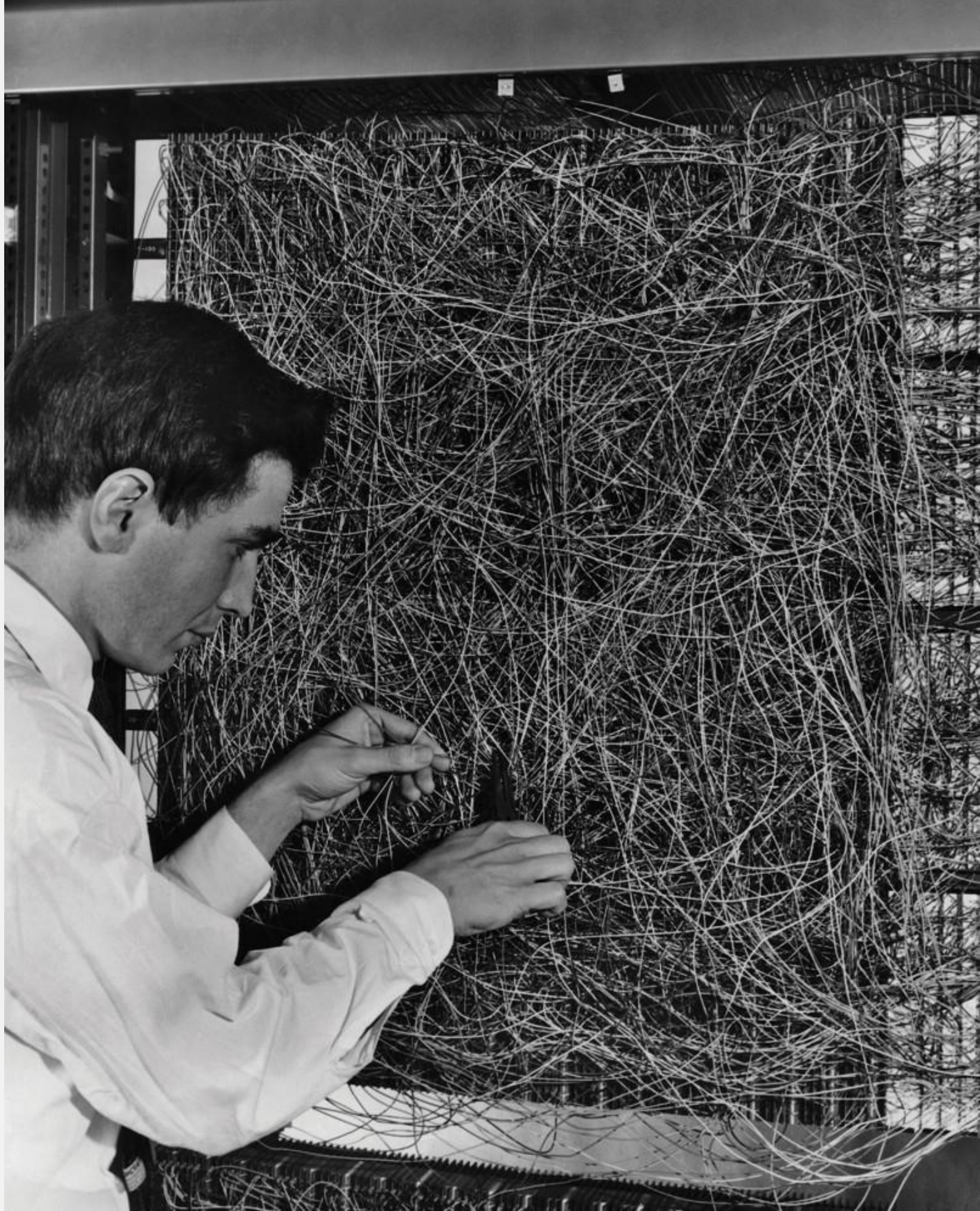
PERCEPTRON



Inputs
Weights
Activation
function
1
x_1
x_2
x_m
w_0
w_1
w_2
w_m
Σ
output

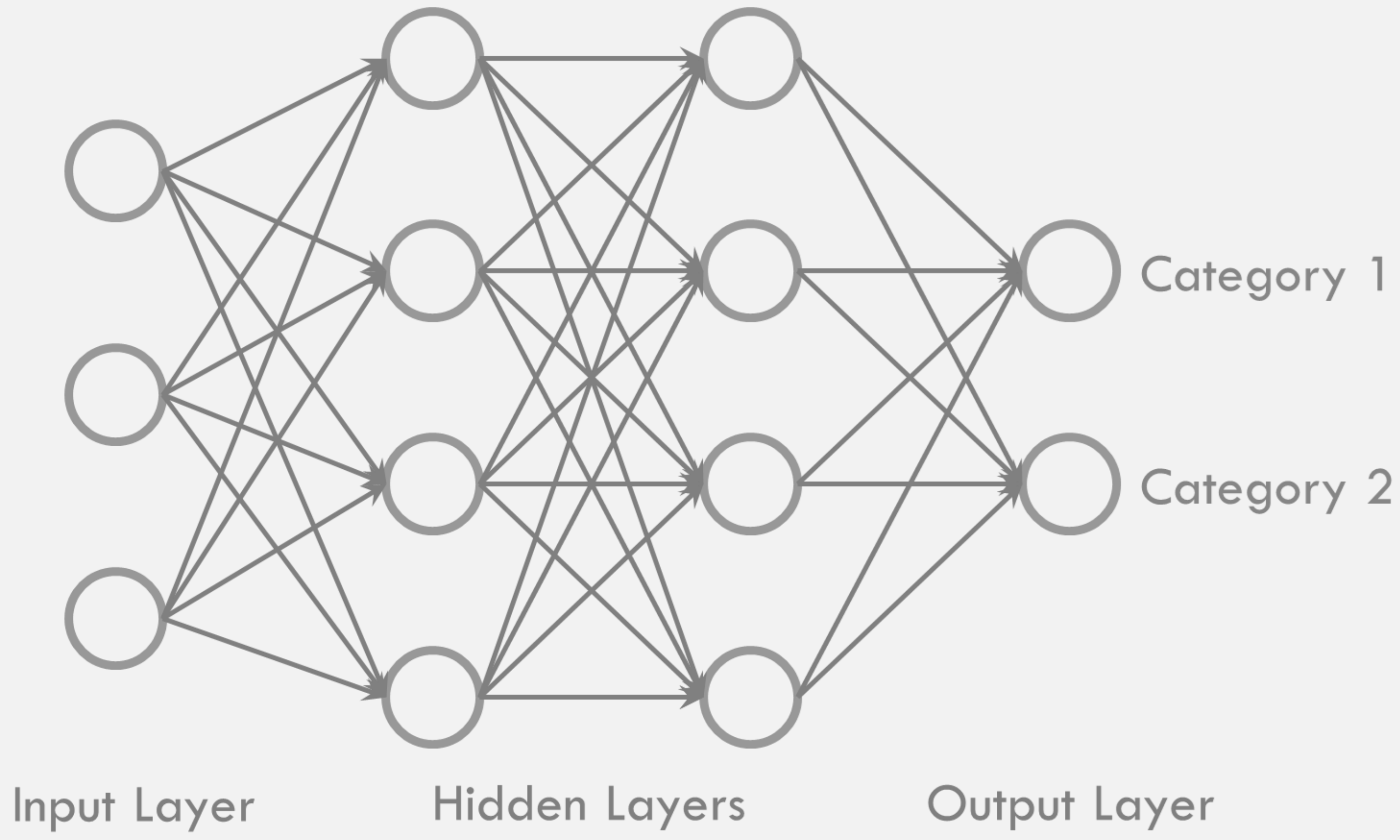
PERCEPTRON


Inputs
Weights
Activation
function
1
x_1
x_2
x_m
w_0
w_1
w_2
w_m
Σ
output

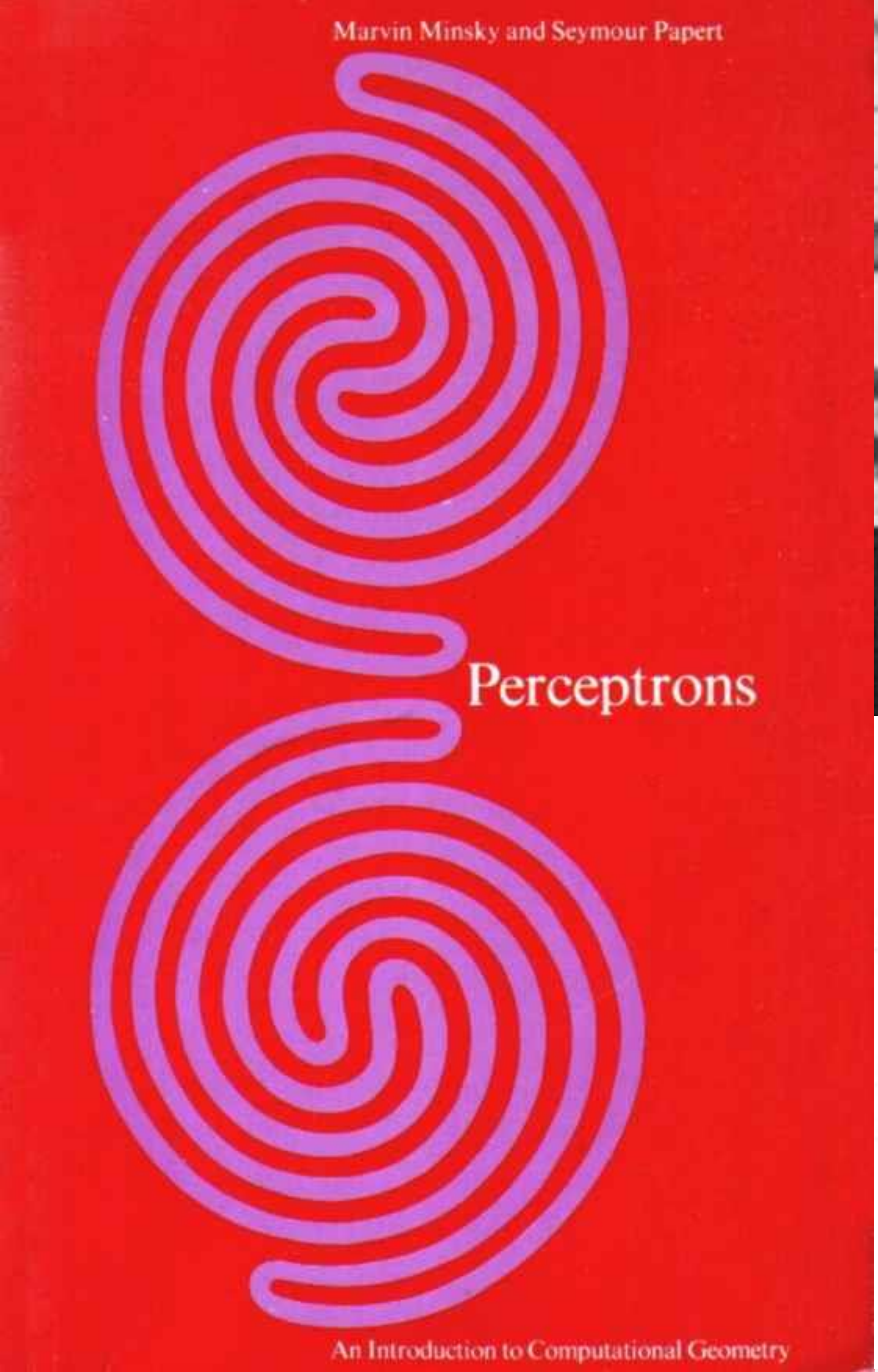


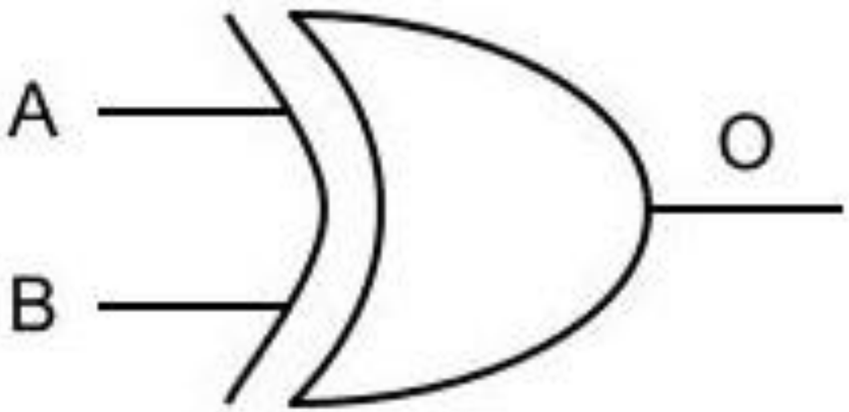
ARTIFICIAL INTELLIGENCE

MULTILAYER PERCEPTRON



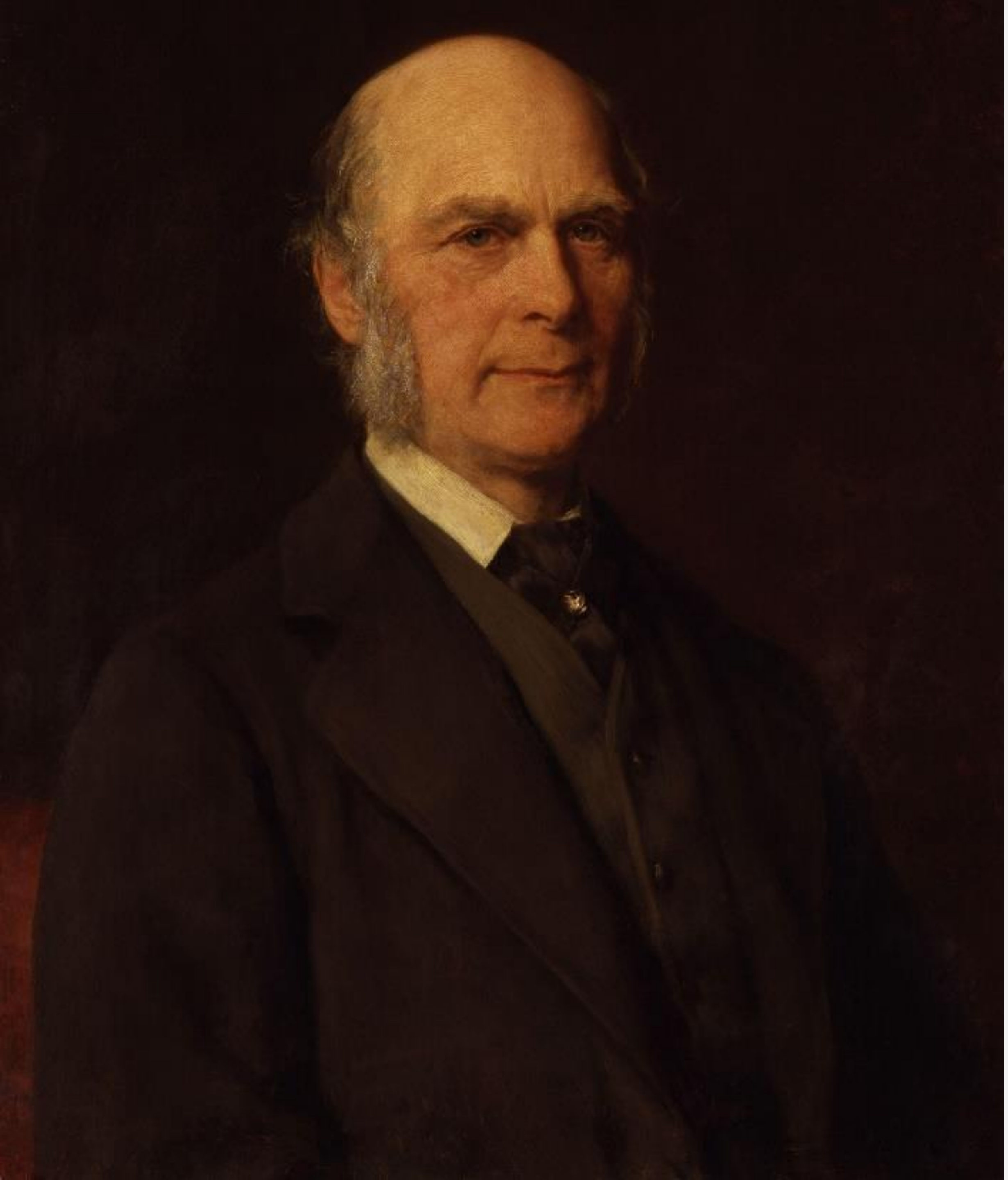
Winter is Coming



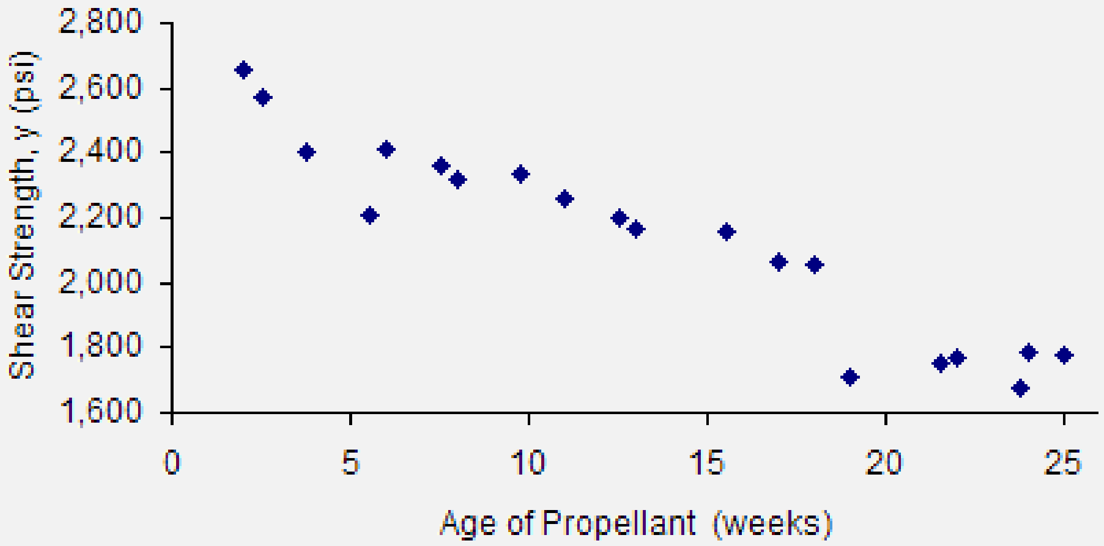


A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

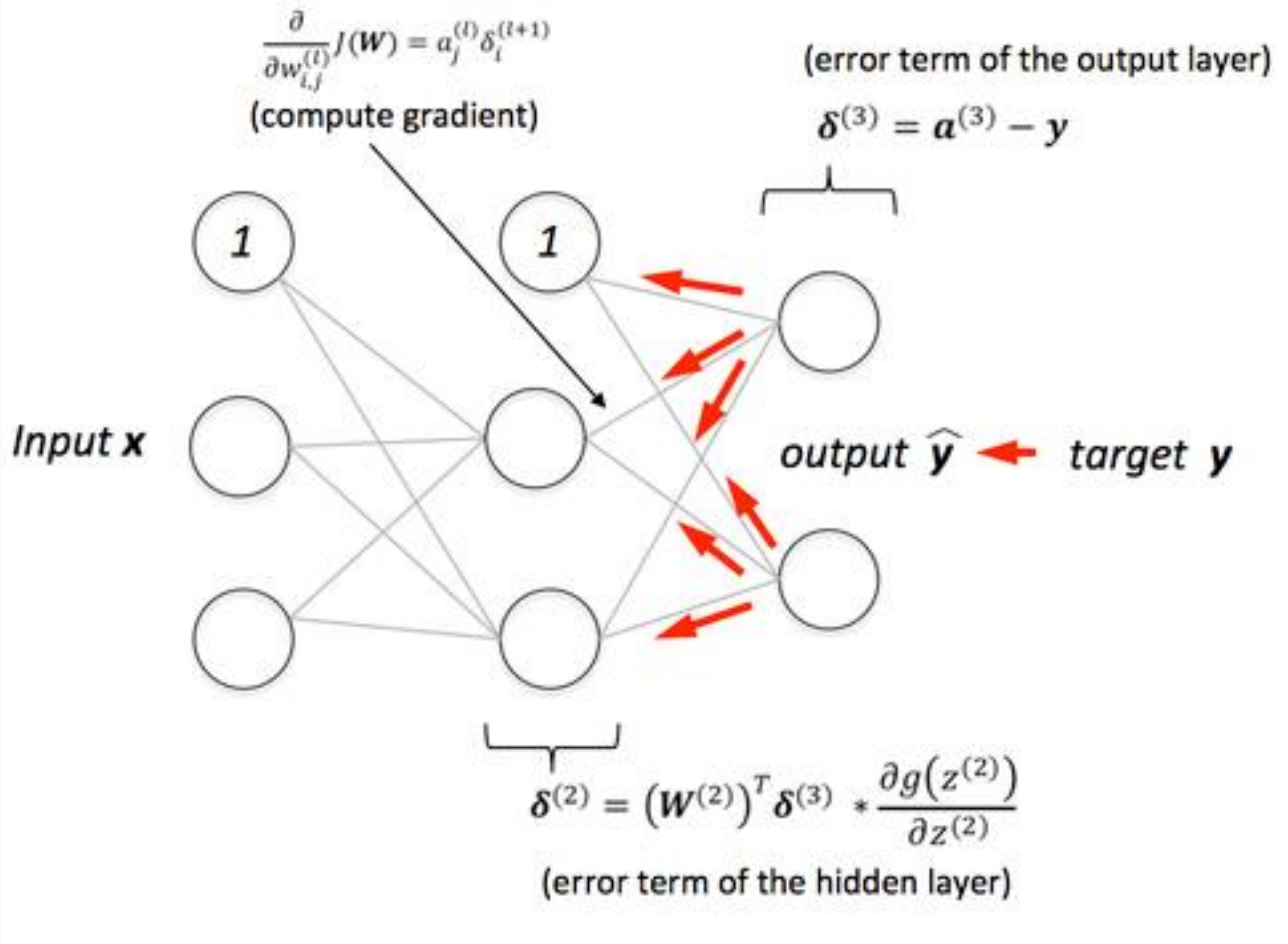
EXPERT SYSTEMS?



MACHINE LEARNING

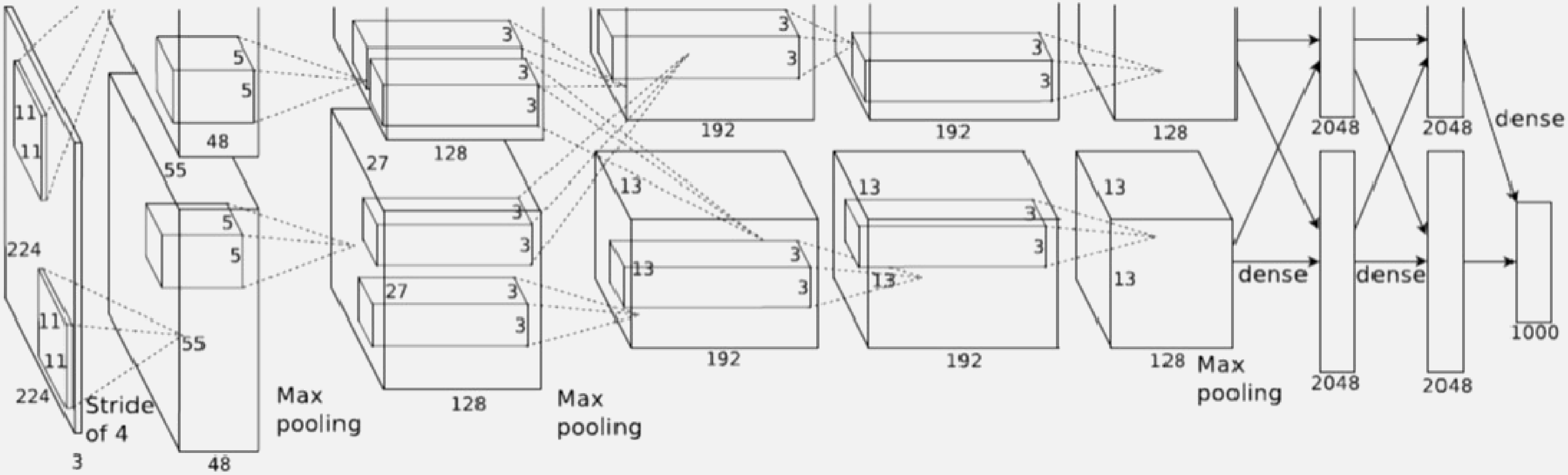


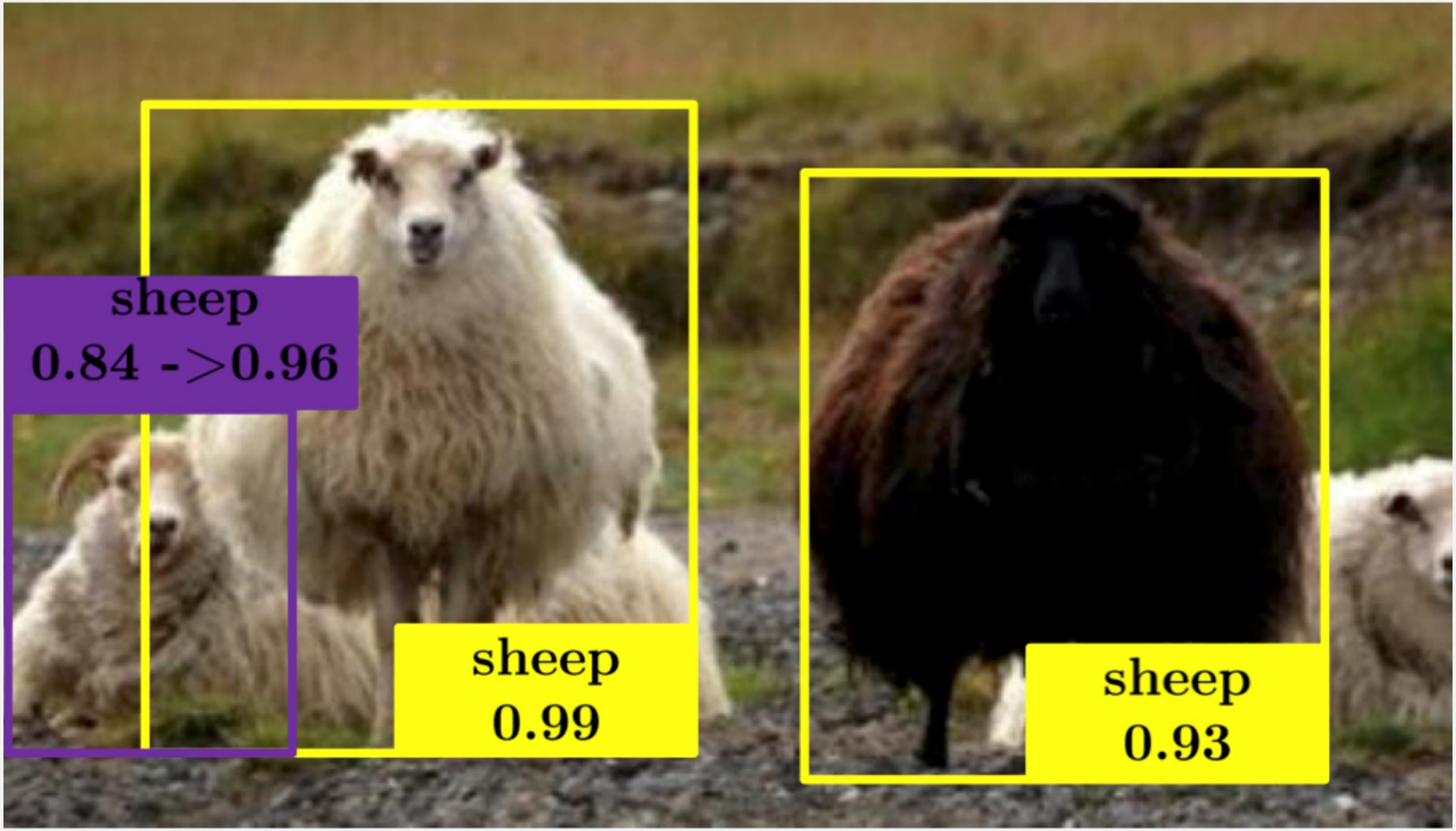
BACKPROPAGATION



DEEP LEARNING

A Machine Learning technique




sheep
0.84 ->0.96
sheep
0.99
sheep
0.93

LAB 0: Setup

CHECKLIST

CUDA
- CUDA 9

Anaconda
- >= 5.2
- Python 3.6

IDE
- VS Code (or Atom, …)
- Jupyter

Repository
https://github.com/PabloDoval/odsc18-london

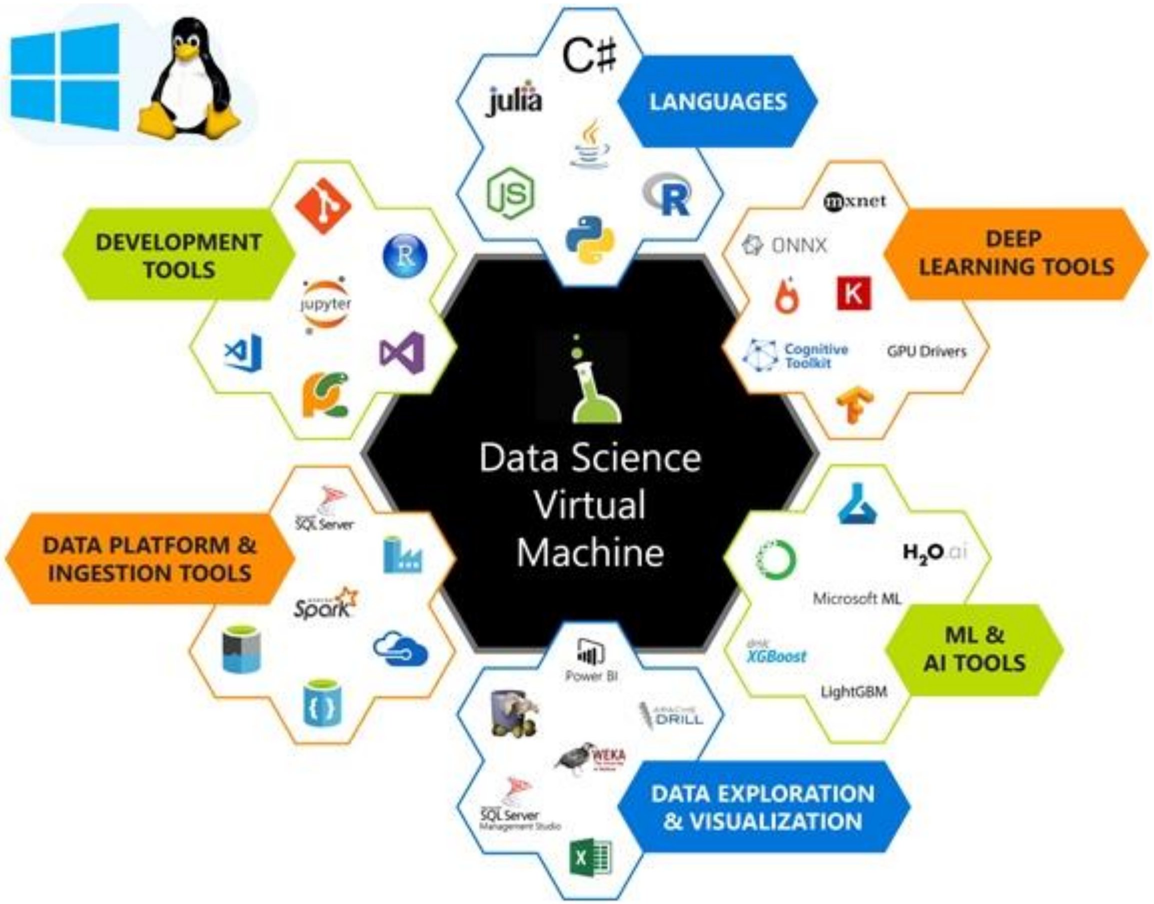
ODSC 2018 Workshop - Deep Learning

Initial Setup

1. First of all, please do install Anaconda 5.2 in your machine
    - Windows version here.
    - Linux version here.
    - MacOS X version here.
2. Go to the .\Scripts folder and run the script that will perform the environment setup:
    - environment.bat (if you are running Windows)
    - ./environment.sh (for you, Linux folks)
3. Once the script finishes running, that environment will be active. The name of the environment is *odsc18-deeplearningworkshop*, in case you want to activate it manually.
4. If you plan on using GPU - and you should! - install the CUDA toolkit from this link to the NVIDIA web page.
5. Run the *main.py* to check that the environment and dependencies have been properly setup. The output will show the version of the Tensorflow and Keras distributions, as well as the number of CPUs and GPUs available in our machine.

Azure Datascience Virtual Machines

Pre-installed installed images, configured and tested with popular tools that are commonly used for data analytics, machine learning and AI.



https://azure.microsoft.com/en-gb/services/virtual-machines/data-science-virtual-machines

Deep Learning with Tensorflow

DEEP LEARNING

Machine Learning

Based on Neural Networks

More than one hidden layer (*Deep Models*)

TensorFlow

Why?

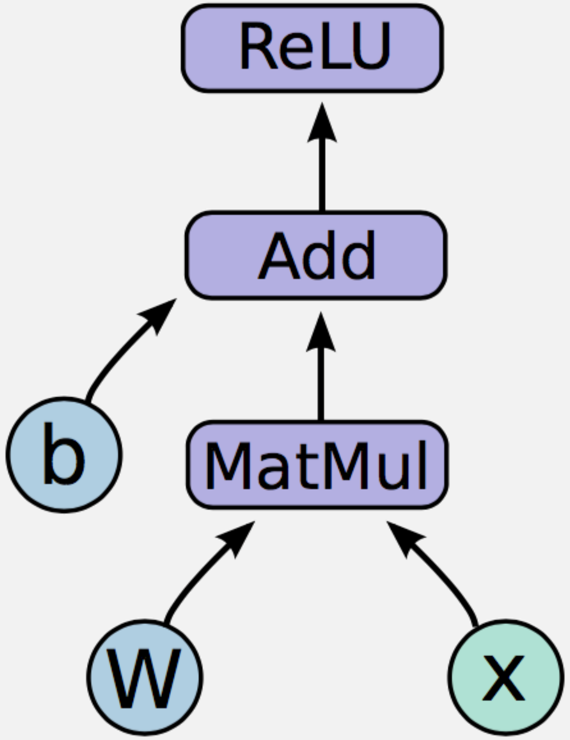
- Interface for expressing machine learning models
- Implementation for executing such models
- Handles multiple CPUs and GPUs
- Multi-platform

TensorFlow

- Express a numeric computation as a graph
- Graph nodes are operations which have any number of inputs and outputs
- Graph edges are tensors which flow between nodes
- Development in Python

TensorFlow

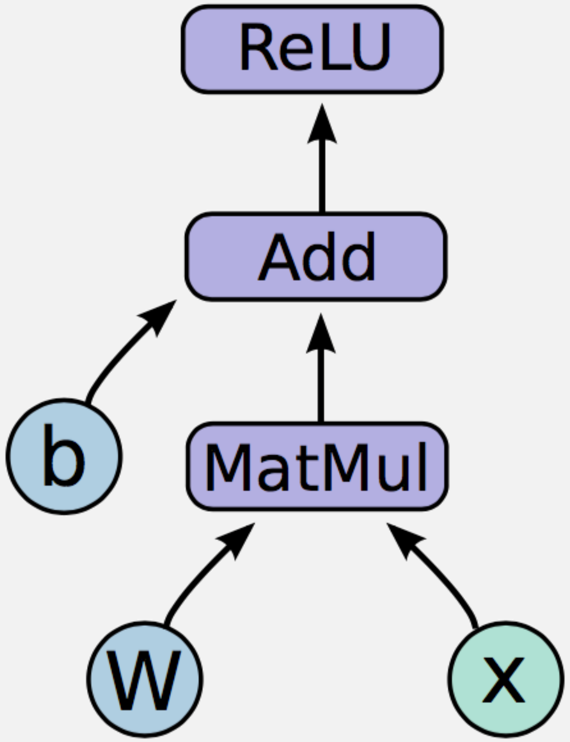
$$h_i = \mathrm{ReLU}(Wx + b)$$



TensorFlow

$$h_i = \mathrm{ReLU}(Wx + b)$$

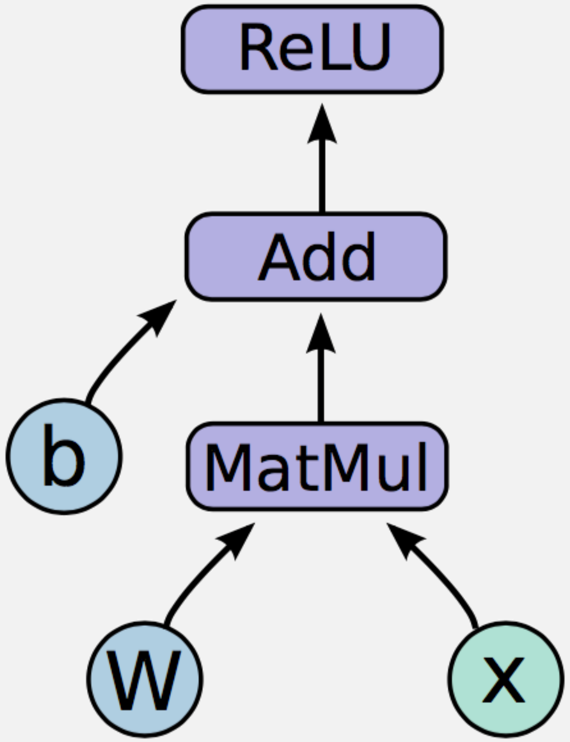
- Variables:
  - Nodes which emit their value as output
  - State is persisted, and retained across multiple executions of the graph
  - Examples: parameters



TensorFlow

$$h_i = \mathrm{ReLU}(Wx + b)$$

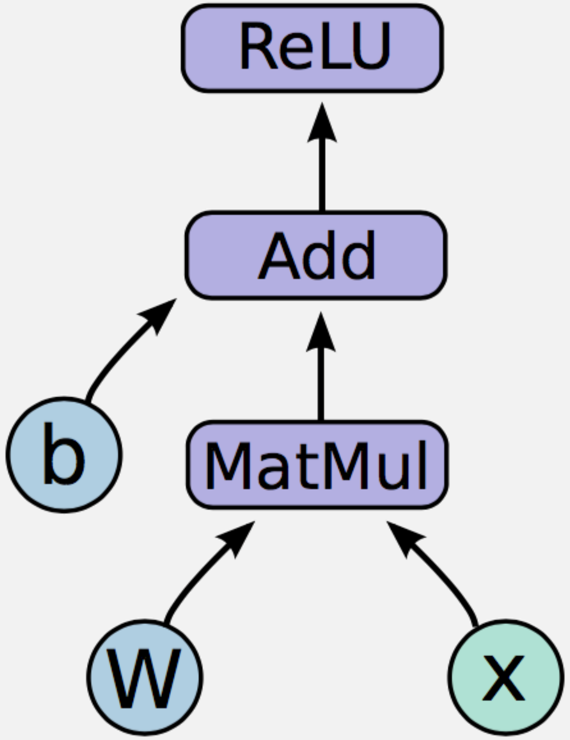
- Placeholders:
  - Values are fed at execution time
  - Examples: inputs, hyper-parameters



TensorFlow

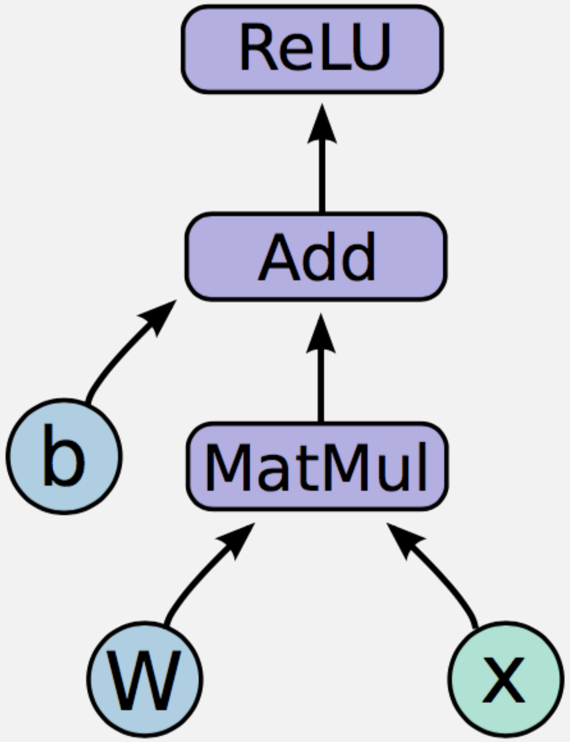
$$h_i = \mathrm{ReLU}(Wx + b)$$

- Operations:
  - Perform the specific operation with the tensor input stream
  - Examples: MatMul, Add, ReLU



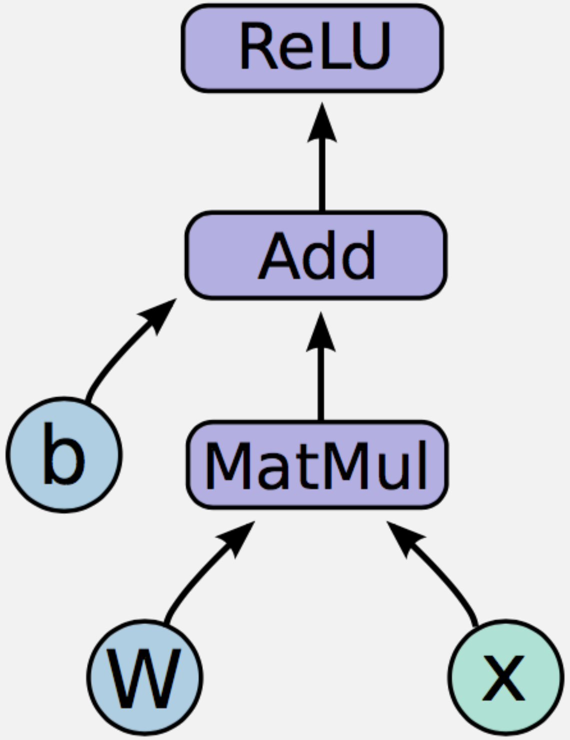
TensorFlow – Graph Code

```
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100), -1, 1))

x = tf.placeholder(tf.float32, (None, 784))
h_i = tf.nn.relu(tf.matmul(x, W) + b)
```



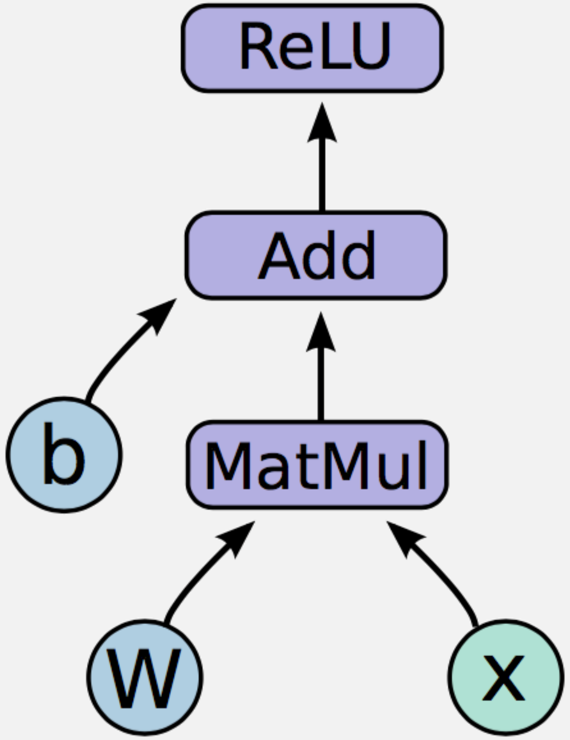
TensorFlow – Graph Execution

- Session:
  - Once we have defined our graph, we need to execute it
  - We can deploy the graph to a session
  - These sessions can be binded to an execution context (GPU, CPU, …)



TensorFlow – Execution Code

```
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100), -1, 1))

x = tf.placeholder(tf.float32, (None, 784))
h_i = tf.nn.relu(tf.matmul(x, W) + b)

sess = tf.Session()
sess.run(tf.initialize_all_variables())
sess.run(h_i, {x: np.random.random(64, 784)})
```

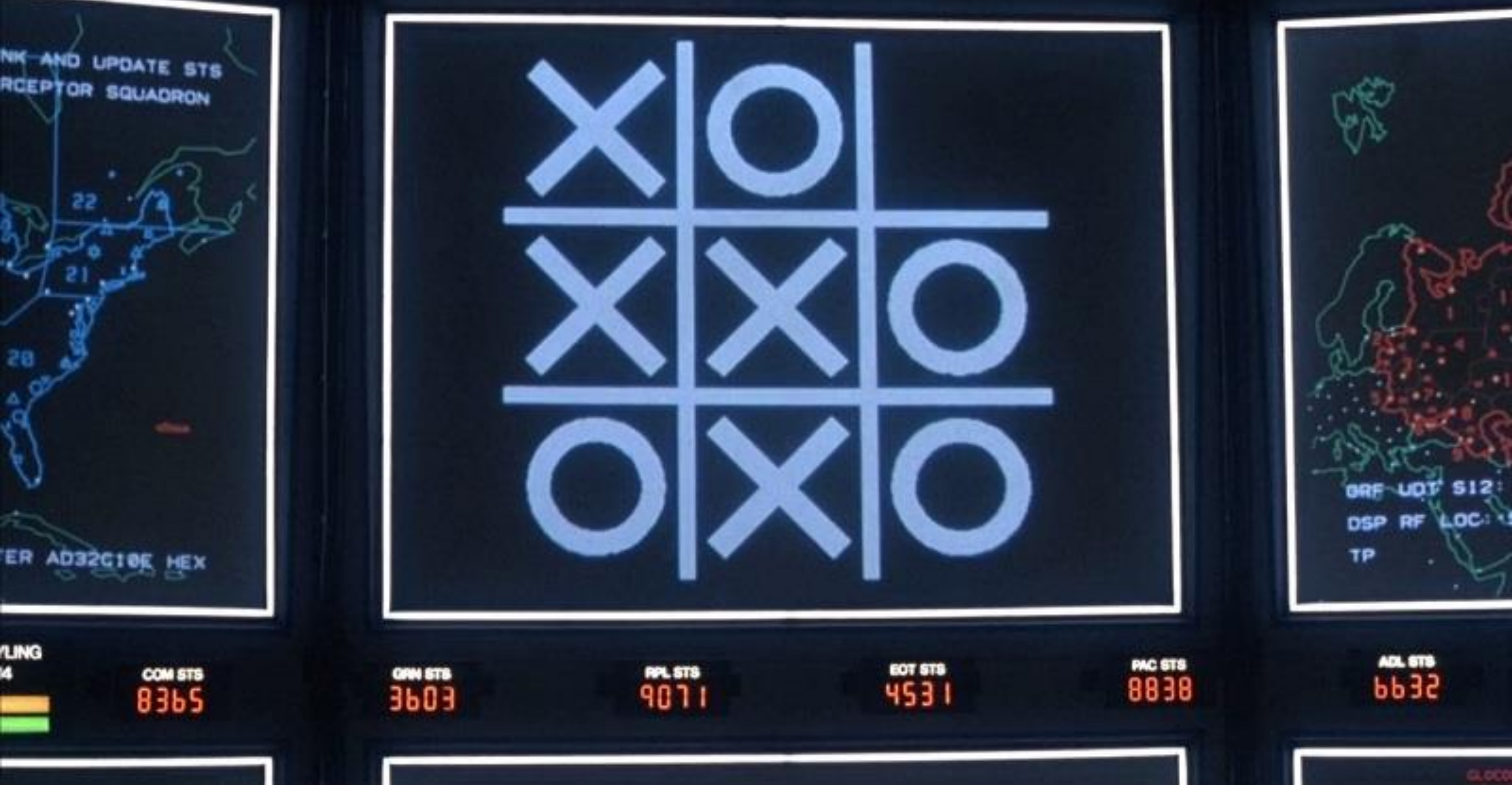


LAB 1: TensorFlow Operations

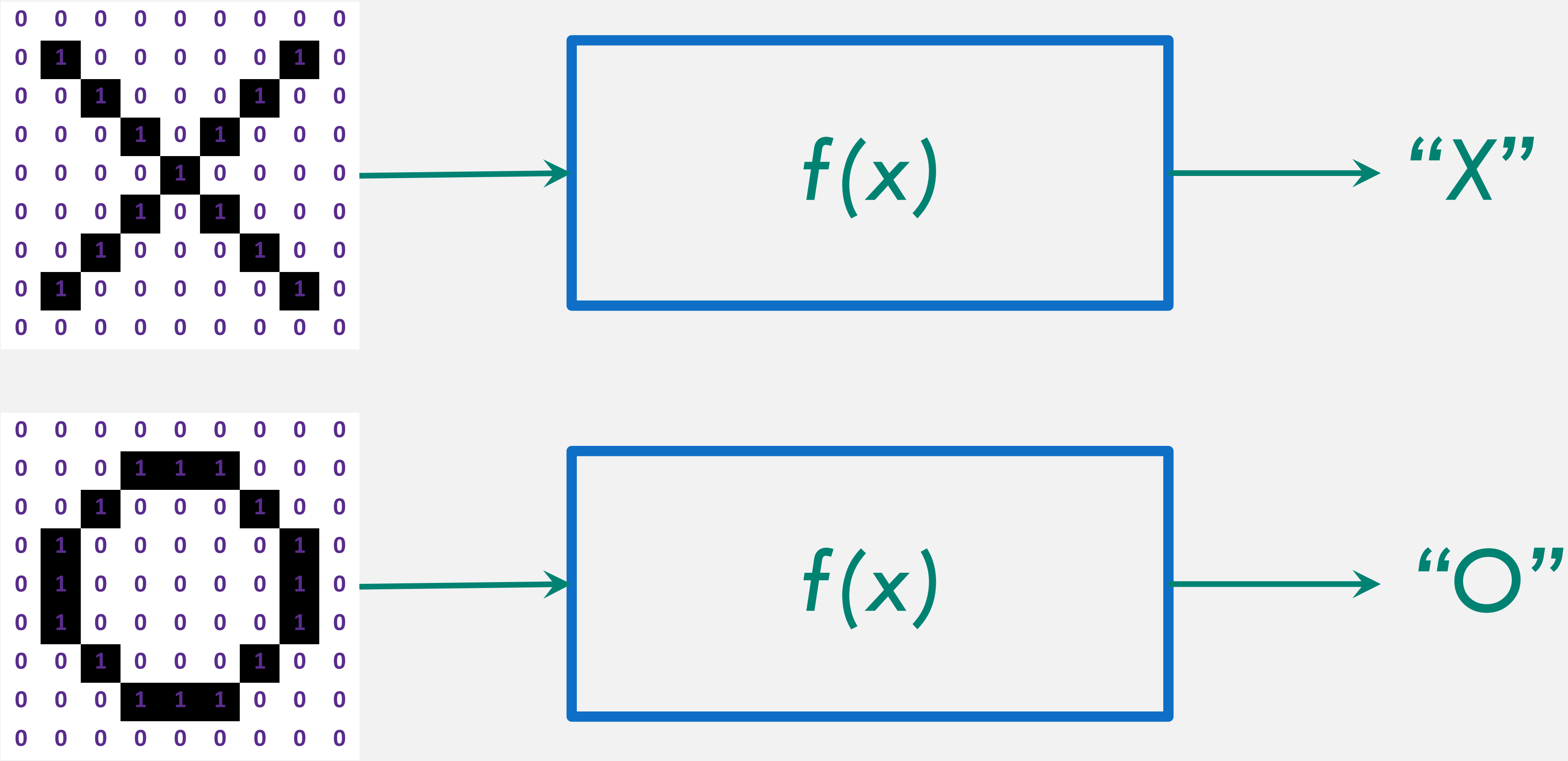
TensorFlow 2.0

- Eager Execution as default
  - No need to add tf.enable_eager_execution()
  - Blocks or Functions can be defined to be executed in graph mode
- Less conventions, more object-oriented and pythonic design
  - Variable_scope removed
- Clean up in libraries and contrib

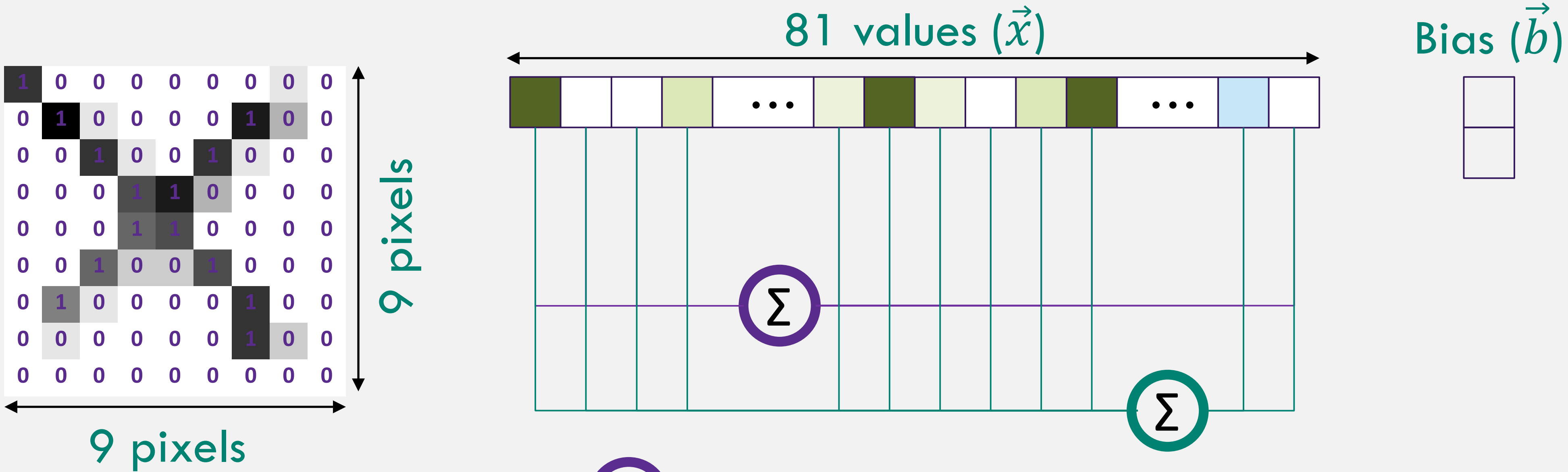
SHALL WE PLAY A LITTLE GAME?



A SIMPLE EXAMPLE



A SIMPLE EXAMPLE



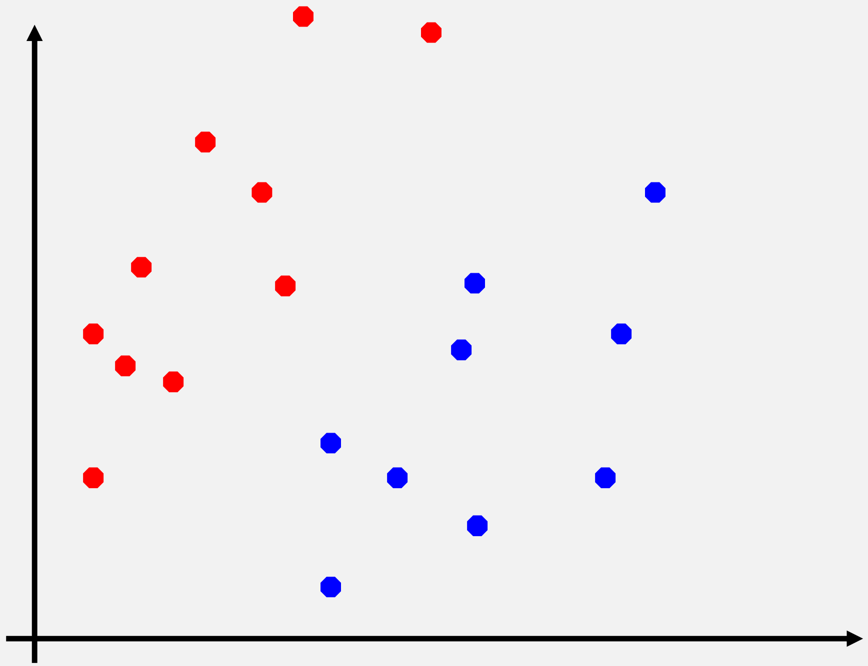
Σ *Sum (weights x pixels)* = $\overrightarrow{w_x} \cdot \vec{x}$

Σ *Sum (weights x pixels)* = $\overrightarrow{w_O} \cdot \vec{x}$

$$z = \vec{w} \cdot \vec{x} + b$$

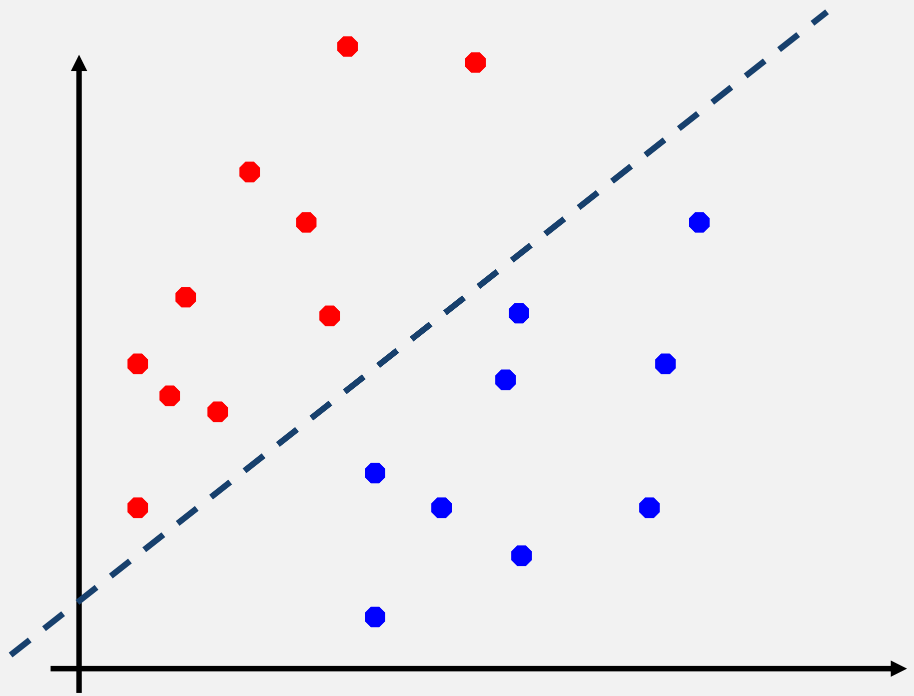
NOW THAT YOU MENTION IT…

$$z = w \cdot \vec{x} + b$$



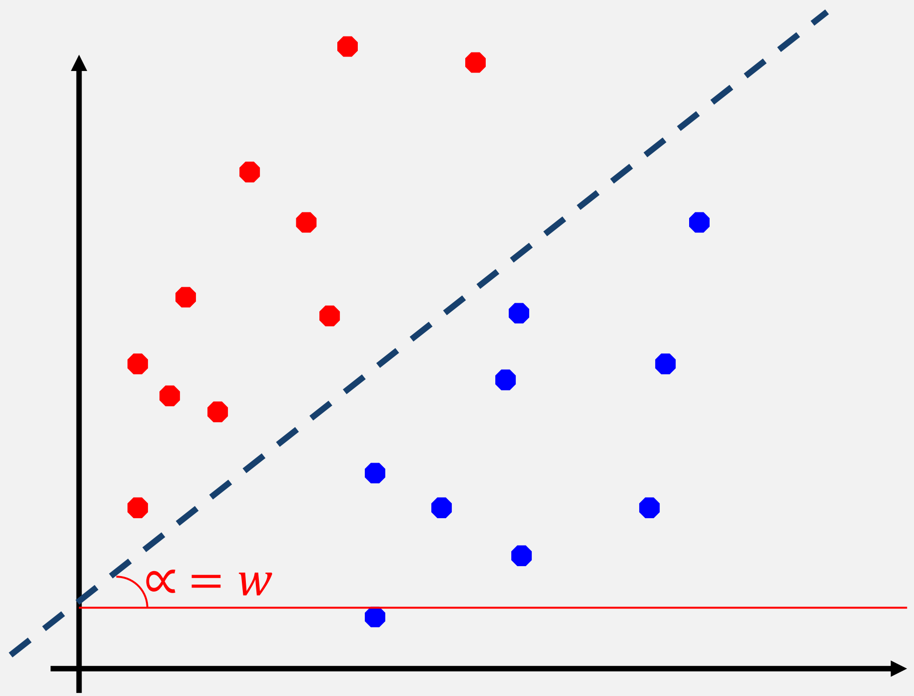
NOW THAT YOU MENTION IT…

$$z = w \cdot \vec{x} + b$$



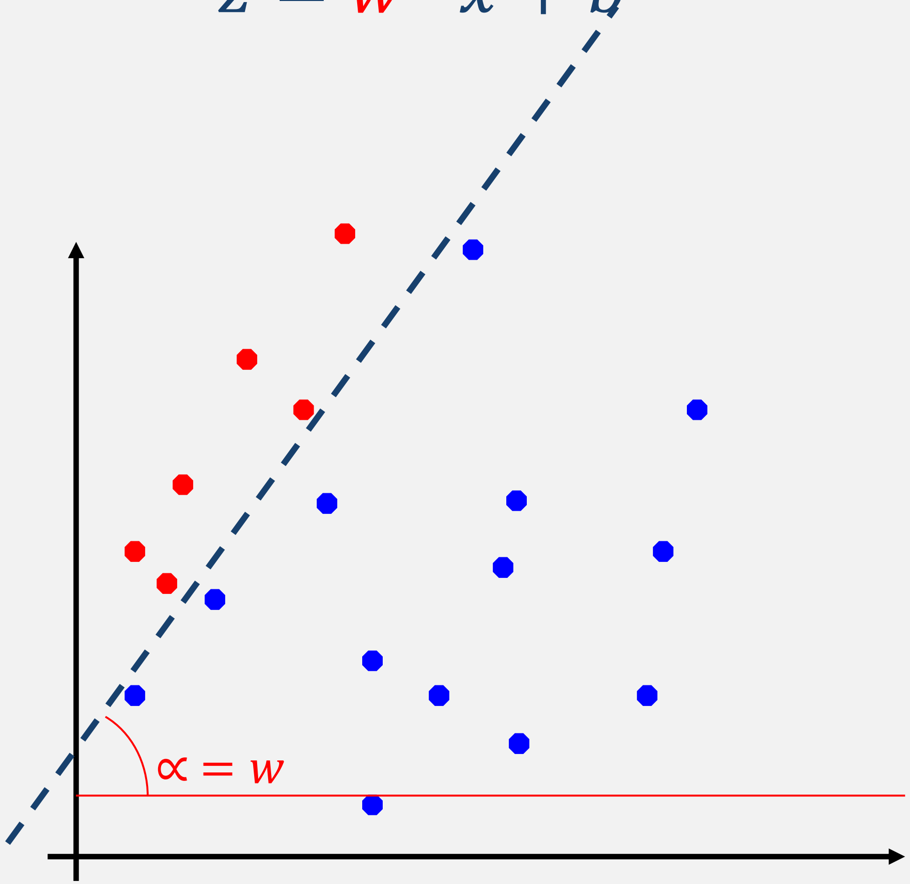
NOW THAT YOU MENTION IT…

$$z = w \cdot \vec{x} + b$$



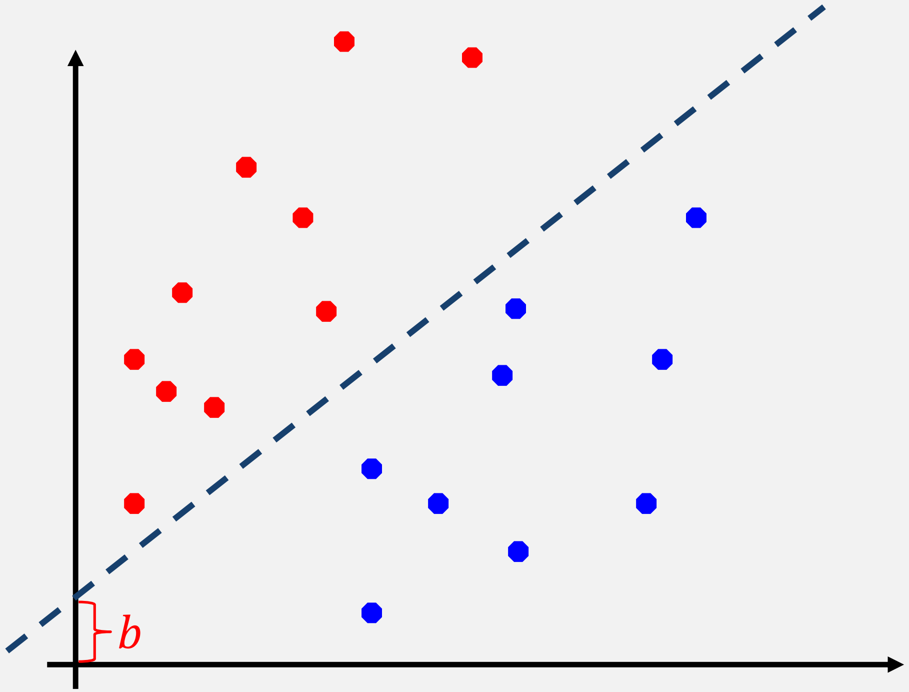
NOW THAT YOU MENTION IT…

$$z = w \cdot \vec{x} + b$$



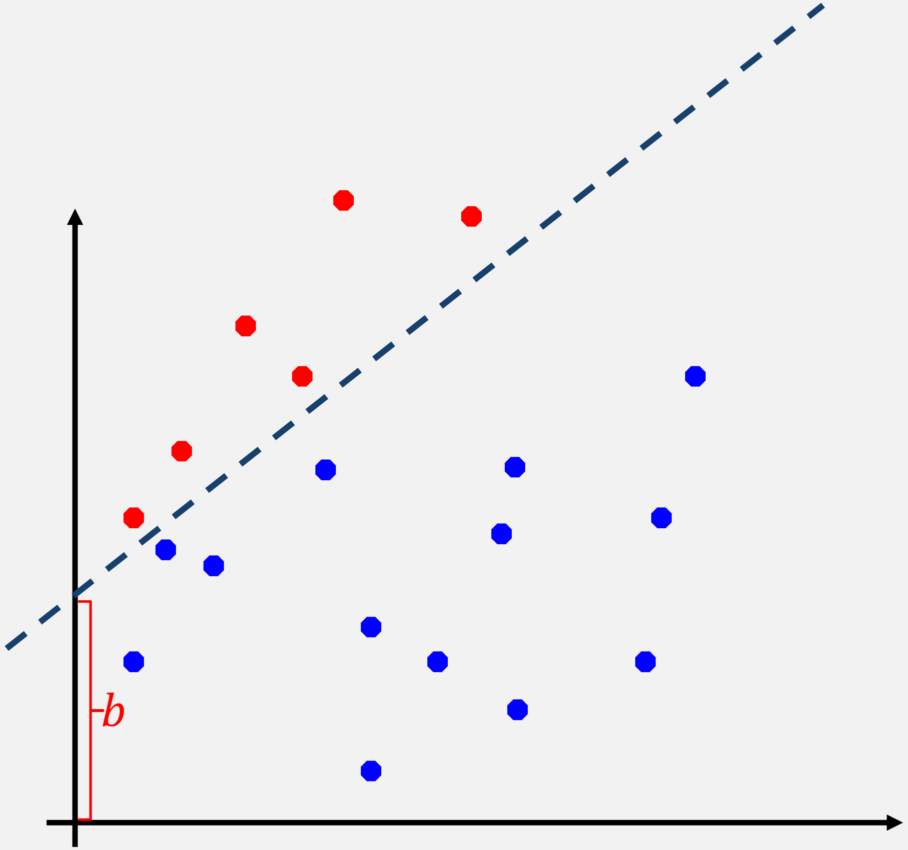
NOW THAT YOU MENTION IT…

$$z = w \cdot \vec{x} + b$$

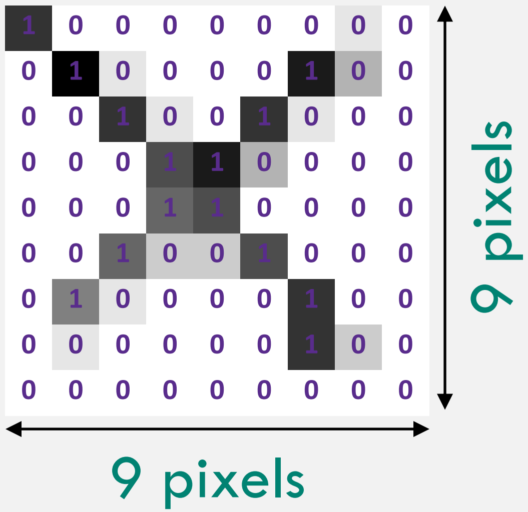


NOW THAT YOU MENTION IT…

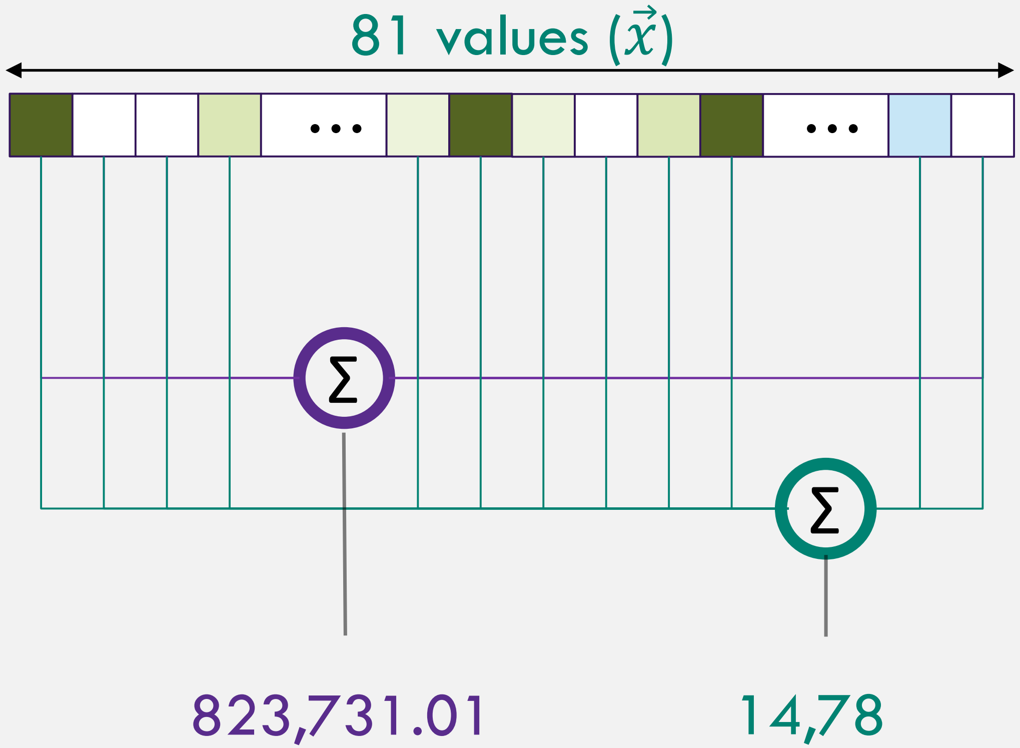
$$z = w \cdot \vec{x} + b$$



SIGMOID

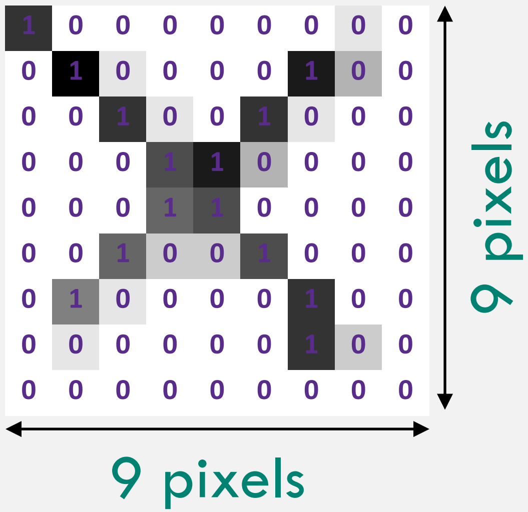


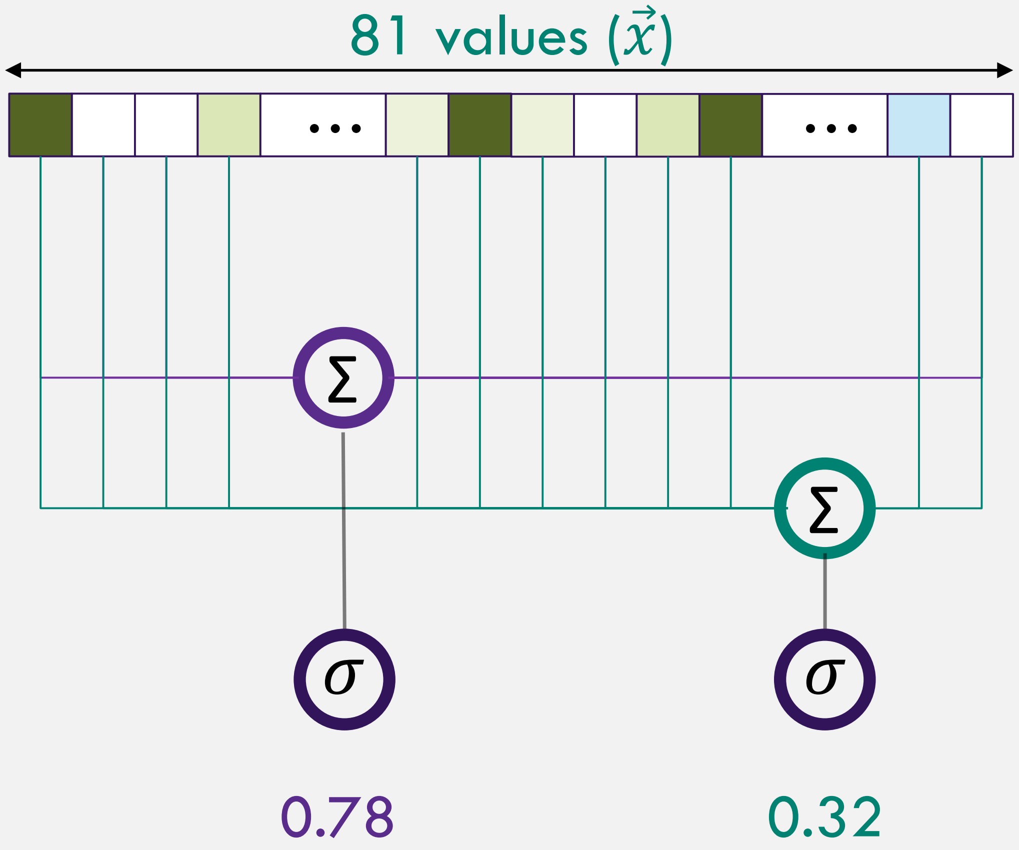
1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 1 0 0
0 0 1 0 0 1 0 0 0
0 0 0 1 1 0 0 0 0
0 0 0 1 1 0 0 0 0
0 0 1 0 0 1 0 0 0
0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0
9 pixels
9 pixels

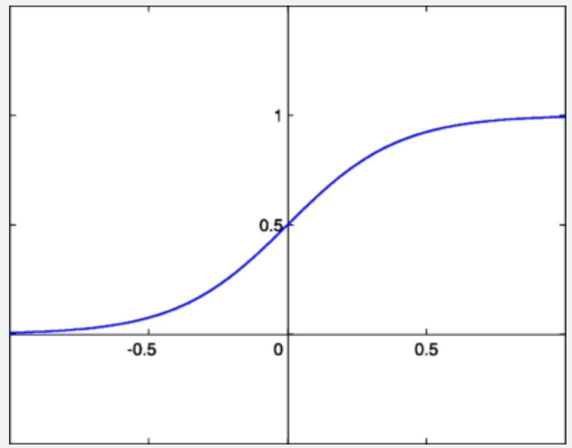


81 values ($\vec{x}$)
...
...
Σ
Σ
823,731.01
14,78

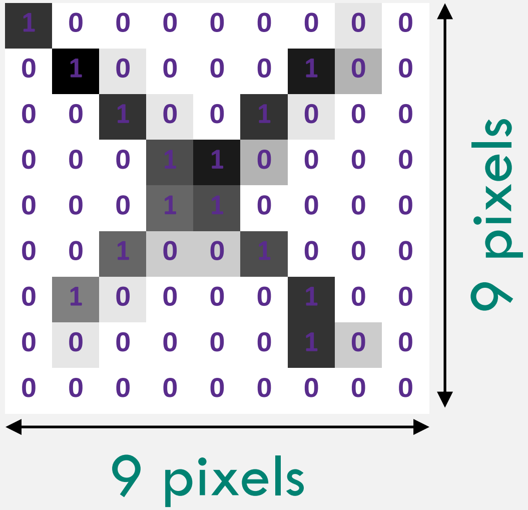
SIGMOID

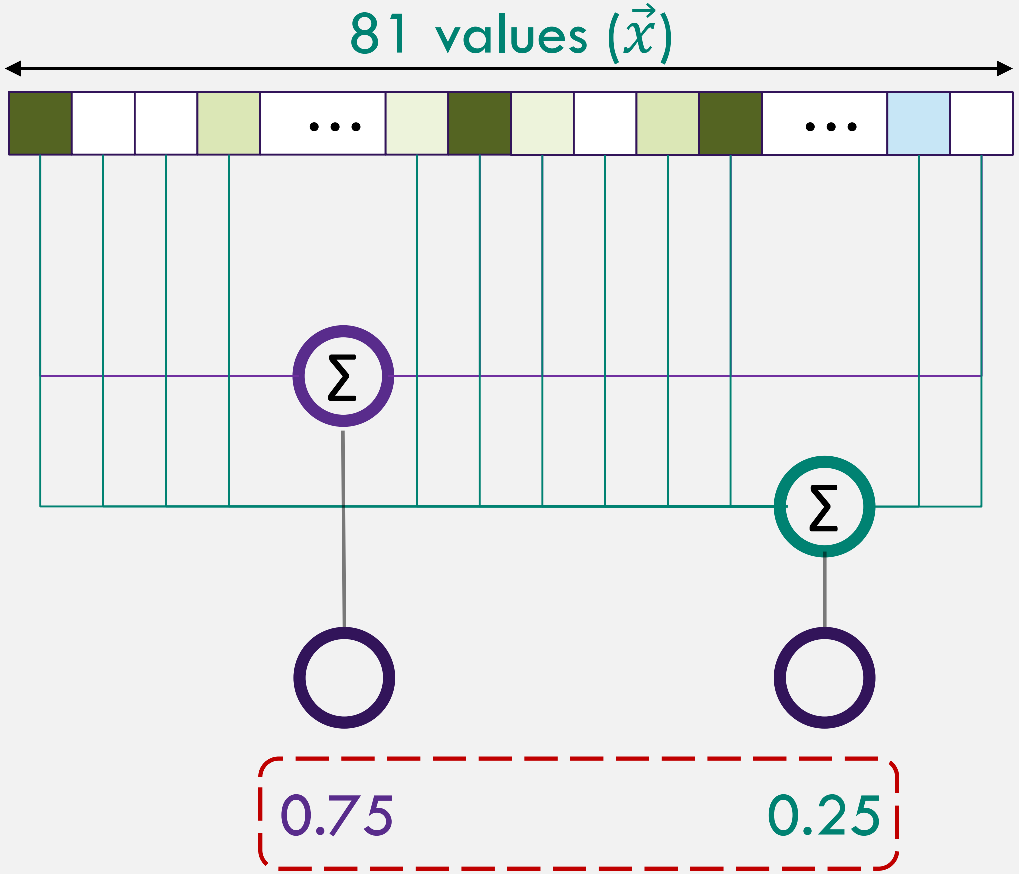

1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 1 0 0
0 0 1 0 0 1 0 0 0
0 0 0 1 1 0 0 0 0
0 0 0 1 1 0 0 0 0
0 0 1 0 0 1 0 0 0
0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0
9 pixels
9 pixels


81 values ($\vec{x}$)
. . .
. . .
Σ
Σ
σ
σ
0.78
0.32


1
0.5
-0.5
0
0.5

SOFTMAX

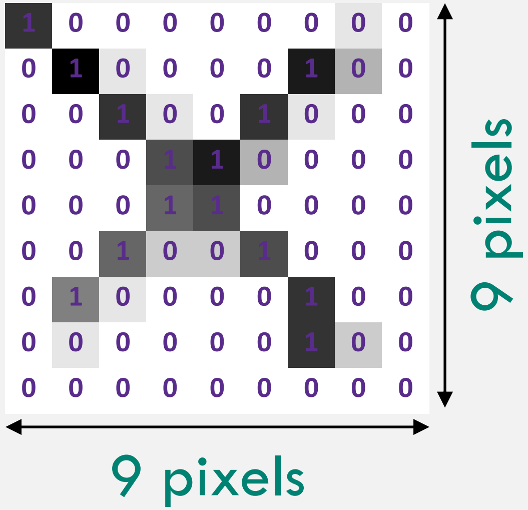




$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

These are actual probabilities

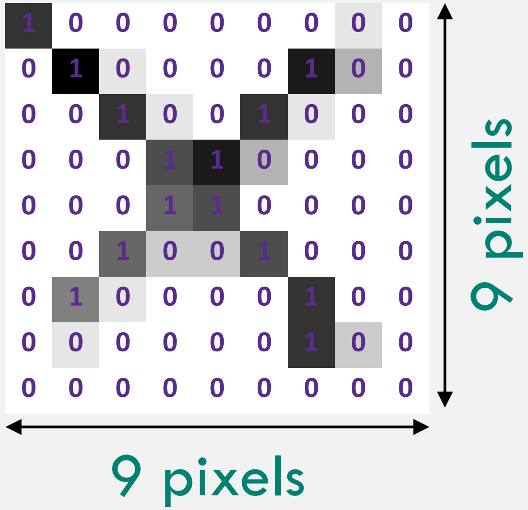
LOSS FUNCTION



One-hot encoding

1	0
0.75	0.25

LOSS FUNCTION



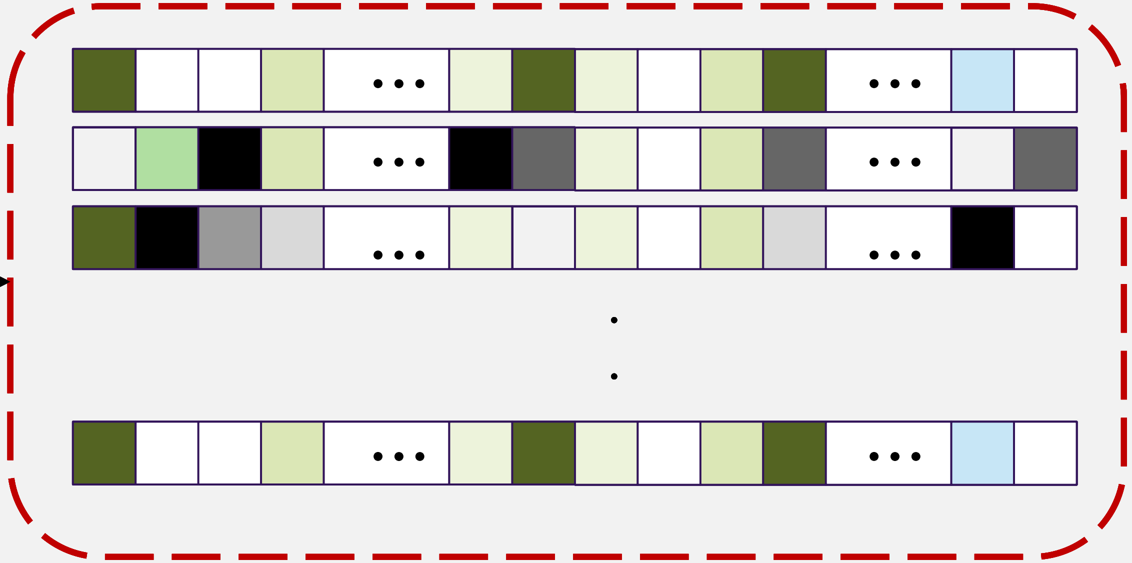
1	0

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n}\left(y_j - p_j\right)^2$$

0.75	0.25

TRAINING

x = vector of pixels (input)



z = times(x, W) + b

cross_entropy_with_softmax(z, y)

classification_error(z, y)

(1, 0)

(0, 1)

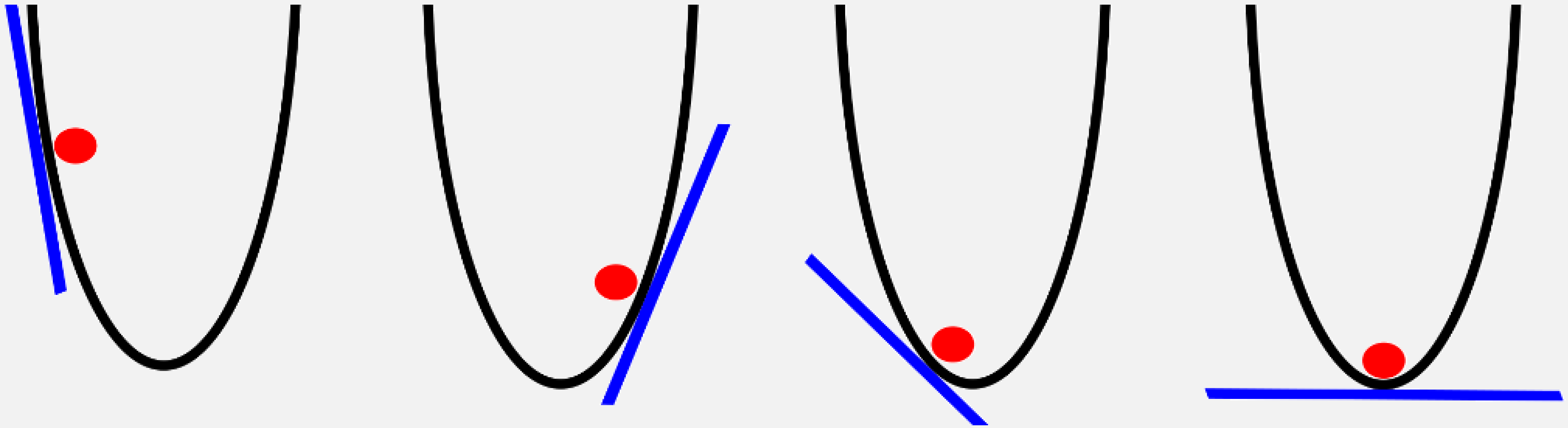
(1, 0)

y = label vector

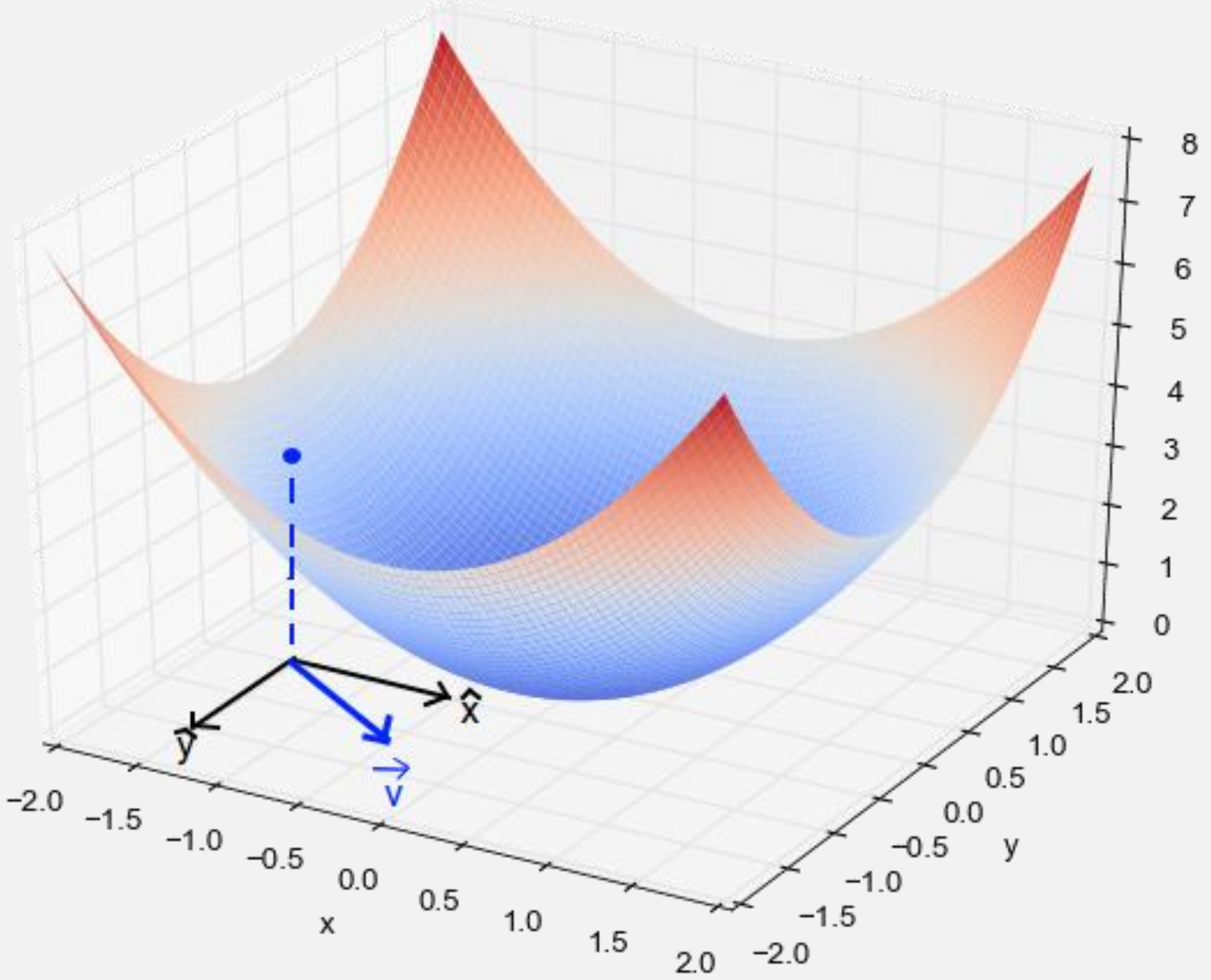
(one-hot)

Training Data

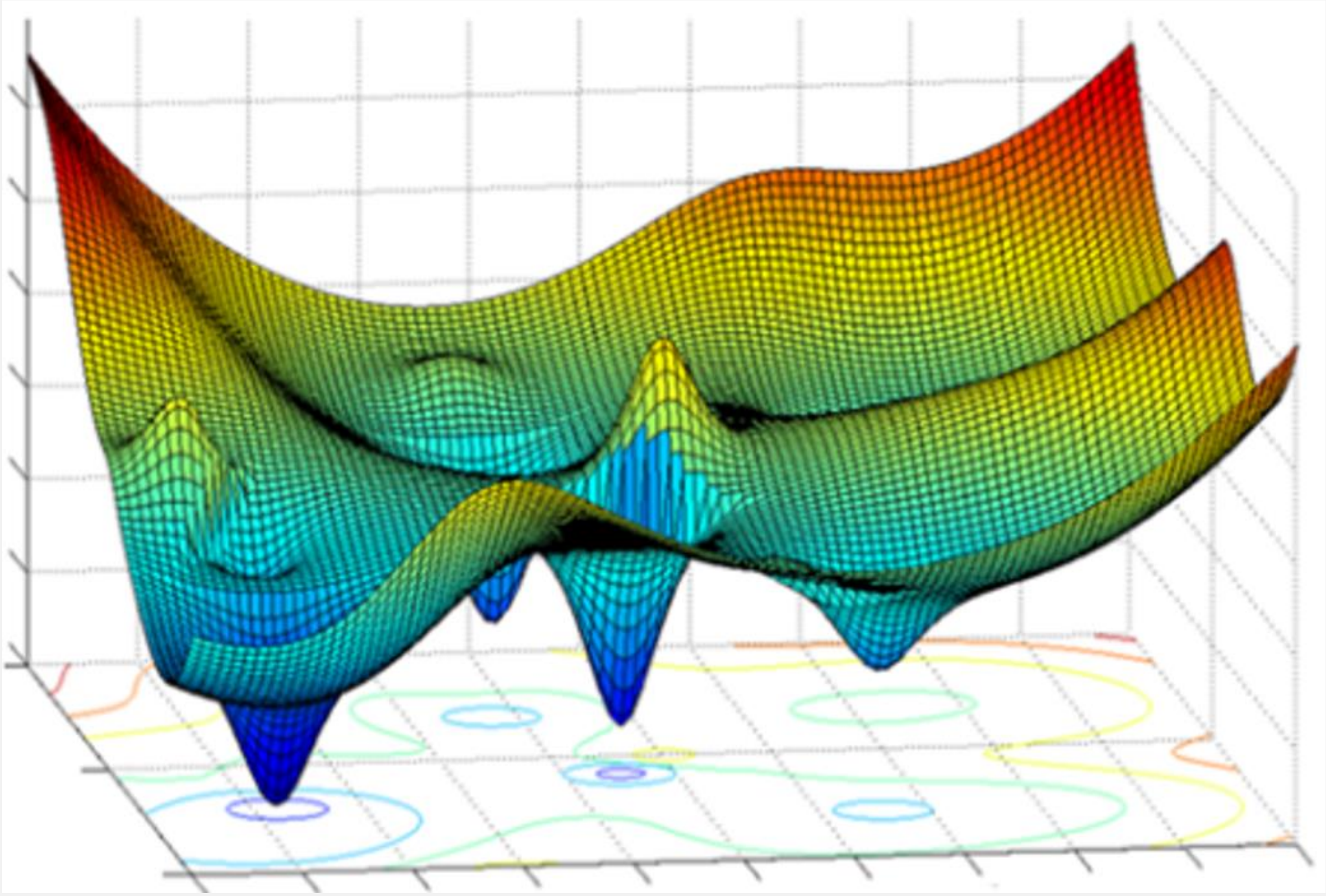
GRADIENT DESCENT



GRADIENT DESCENT



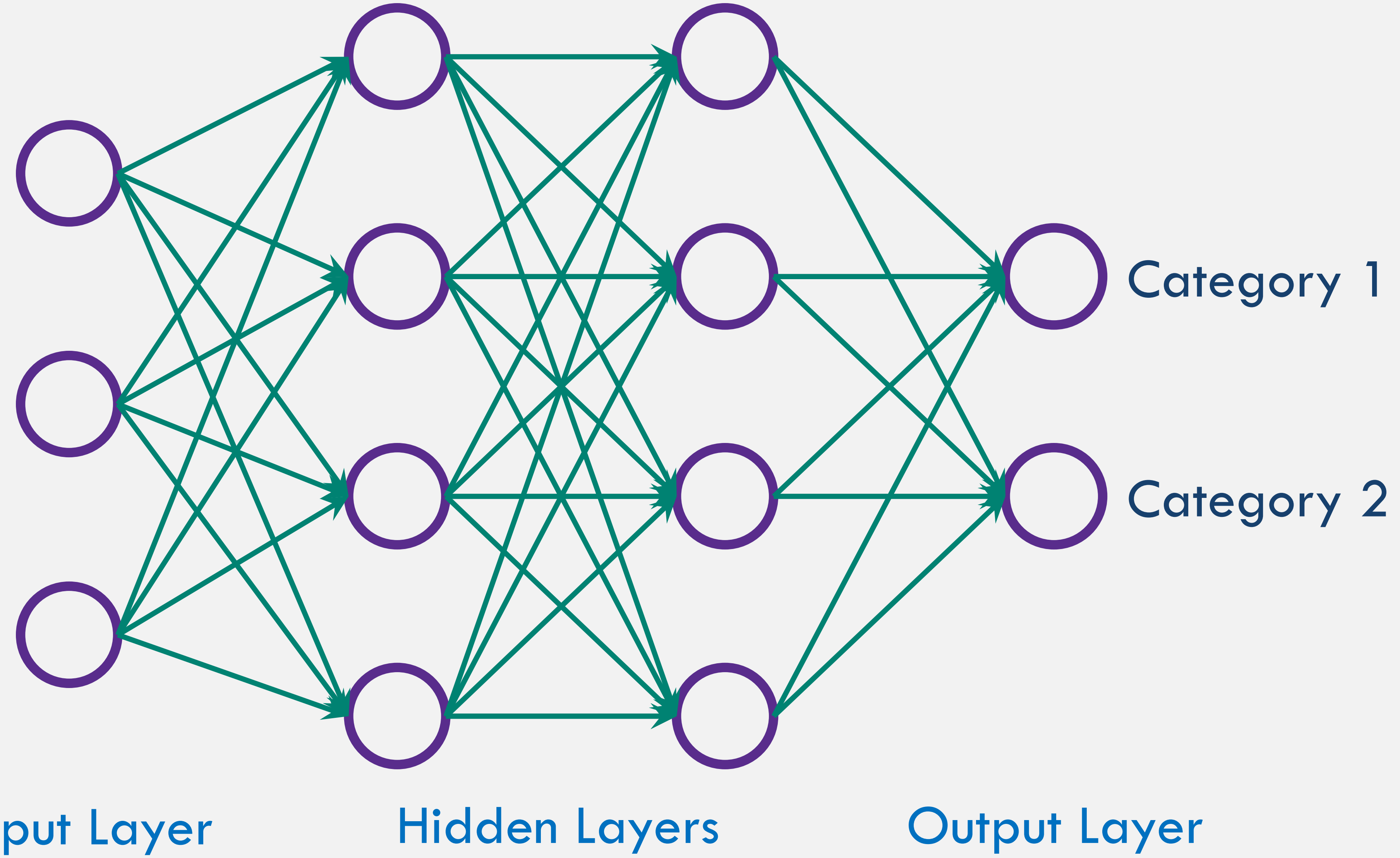
GRADIENT DESCENT



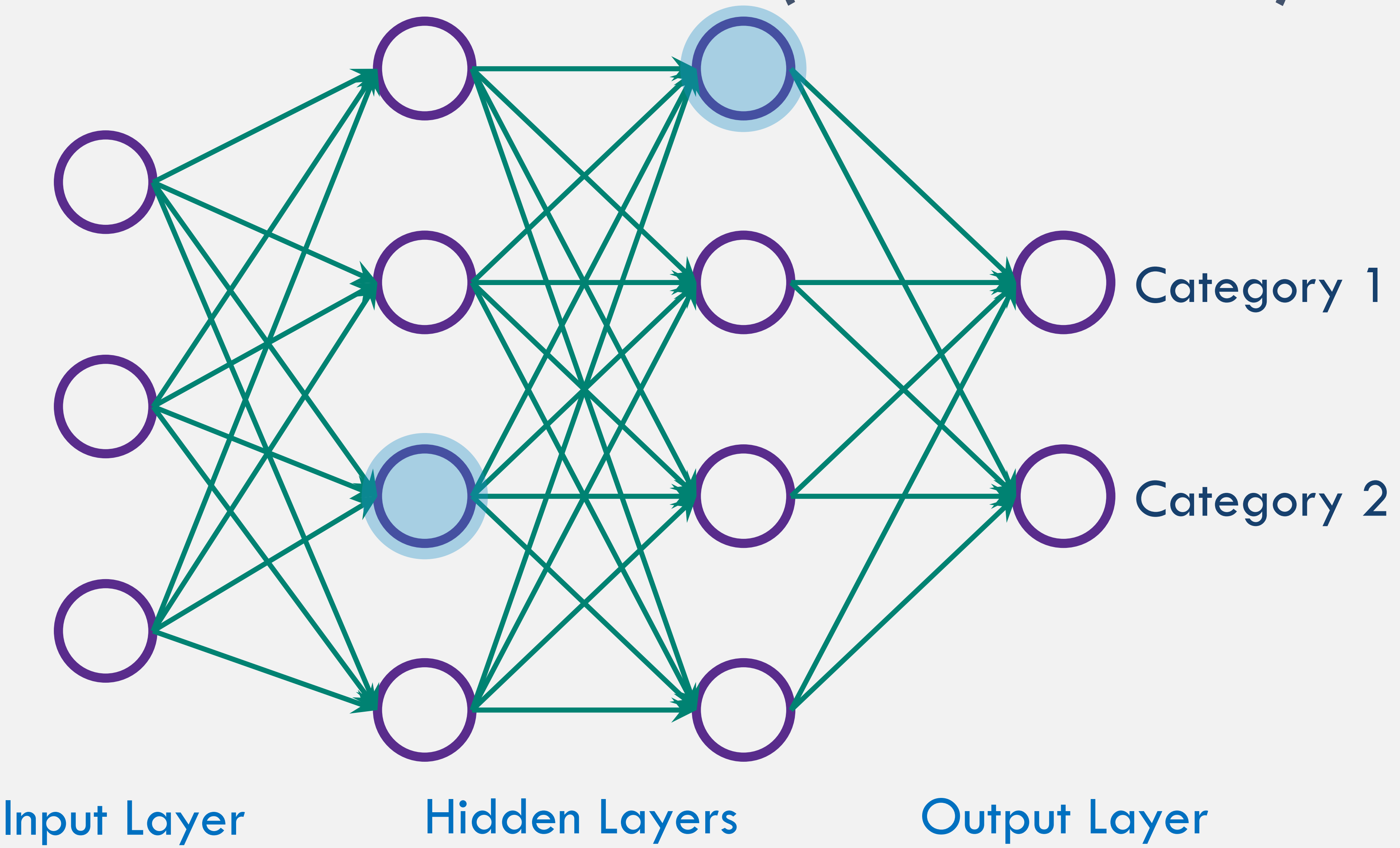
LAB 2: Logistic Regression

Neural Networks

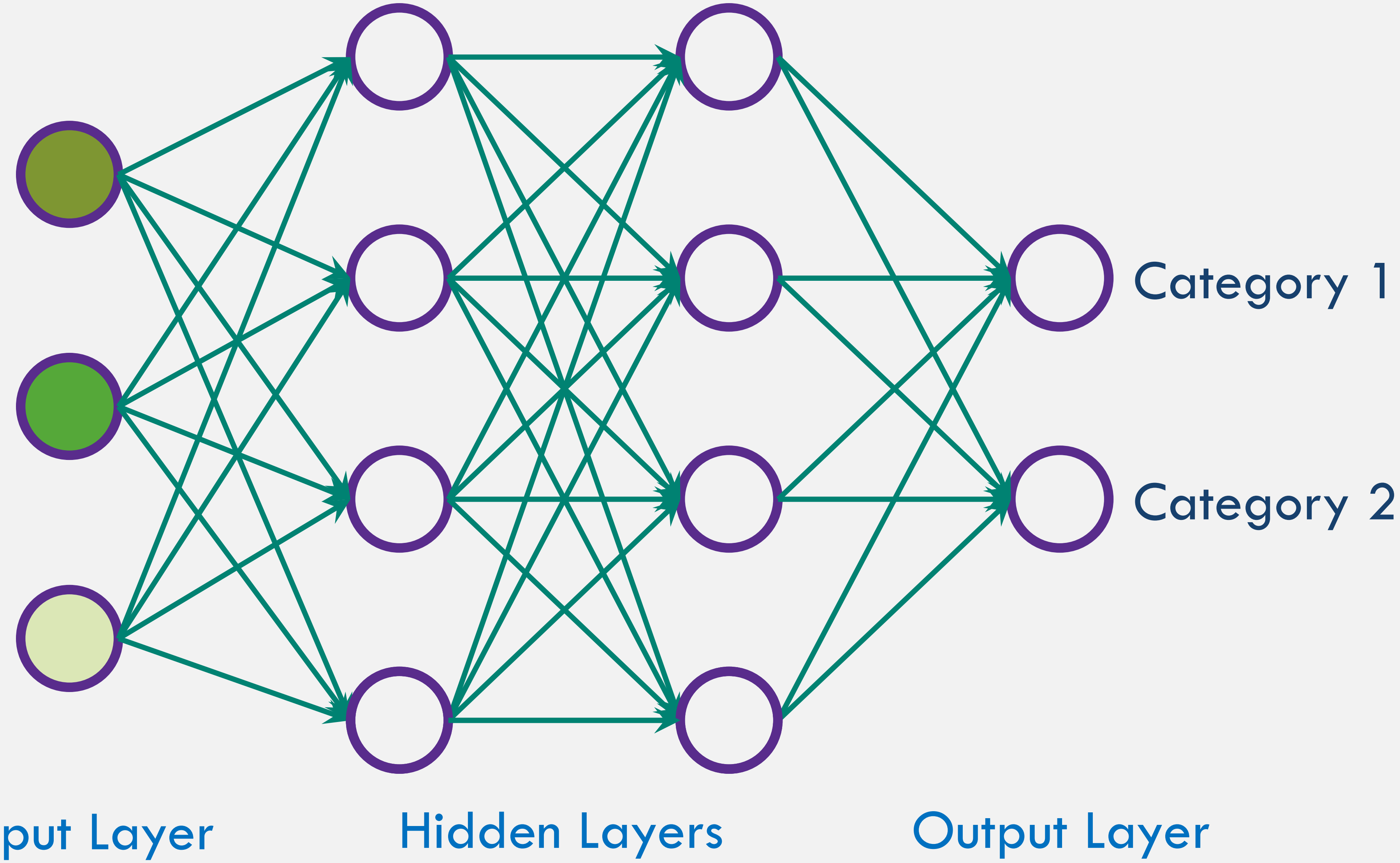
NEURAL NETWORK (CLASSIFIER)



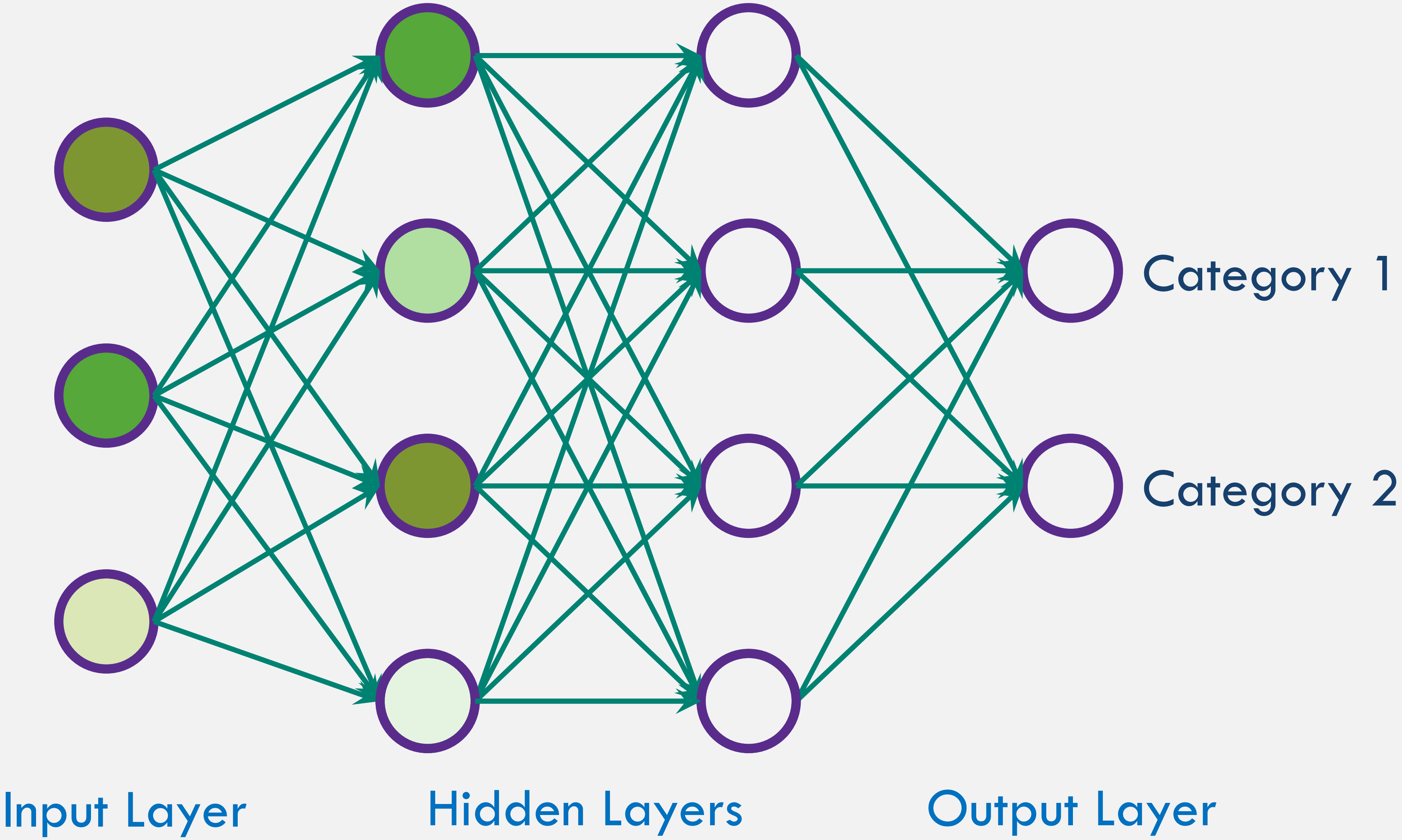
NEURAL NETWORK (CLASSIFIER)



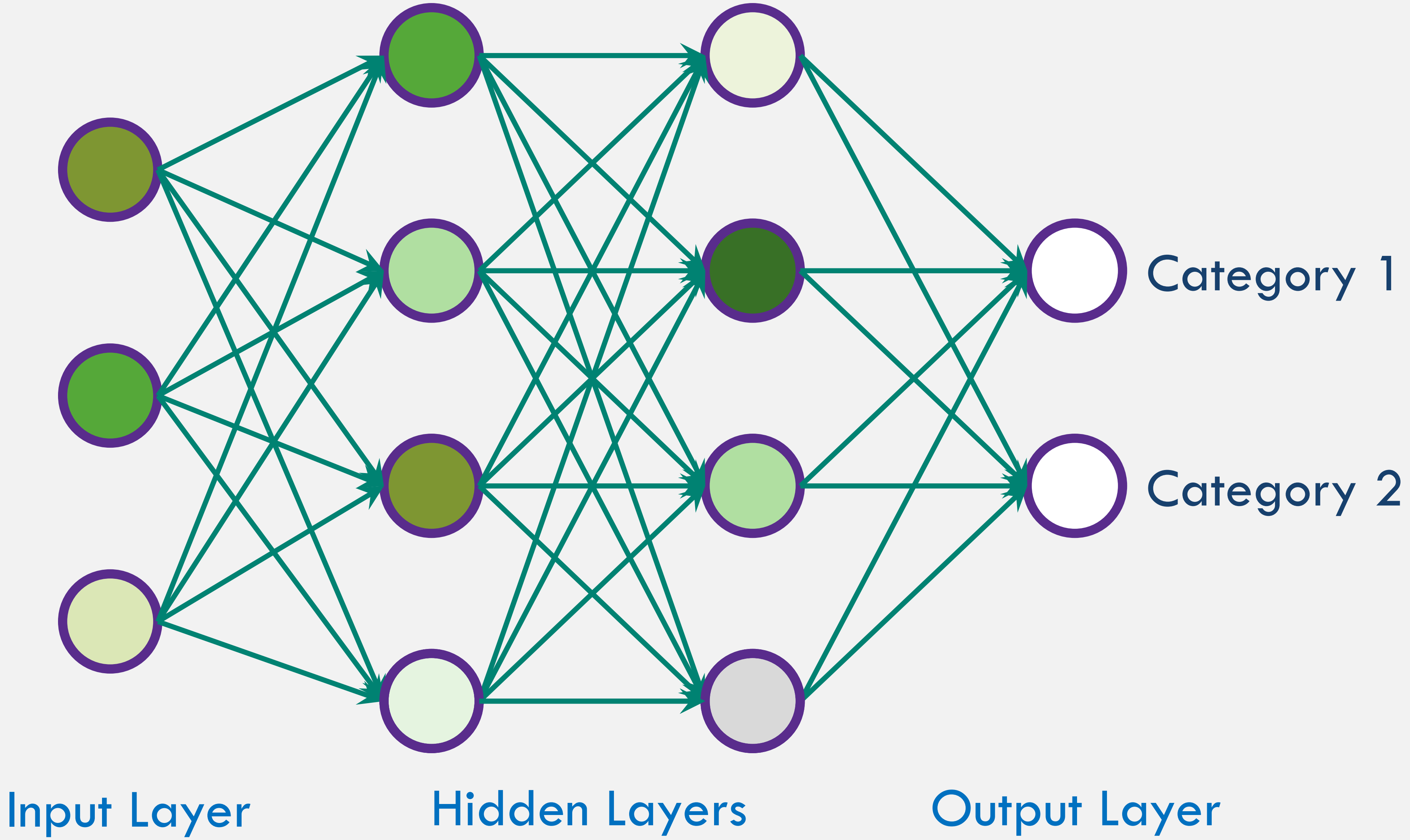
FORWARD PROPAGATION



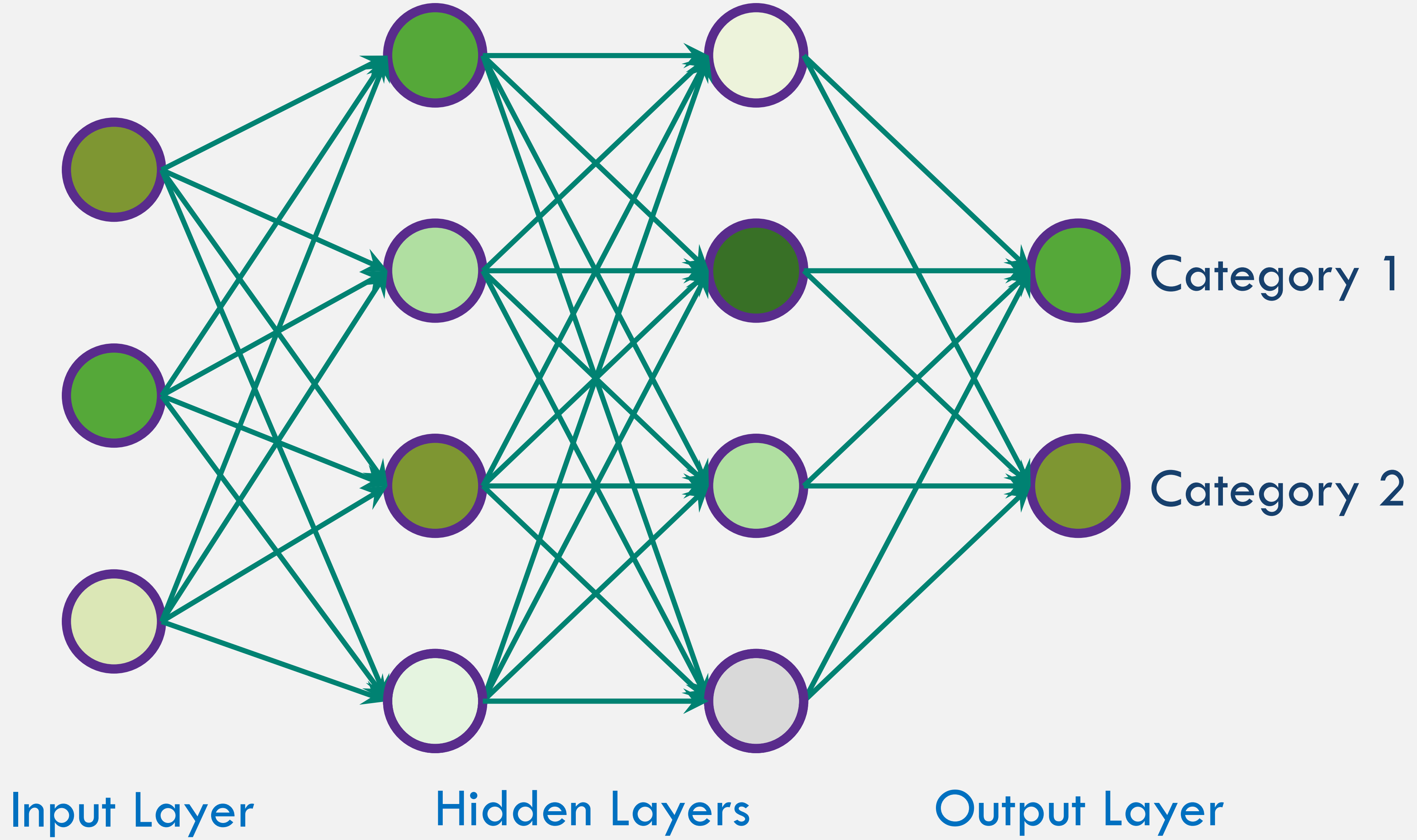
FORWARD PROPAGATION



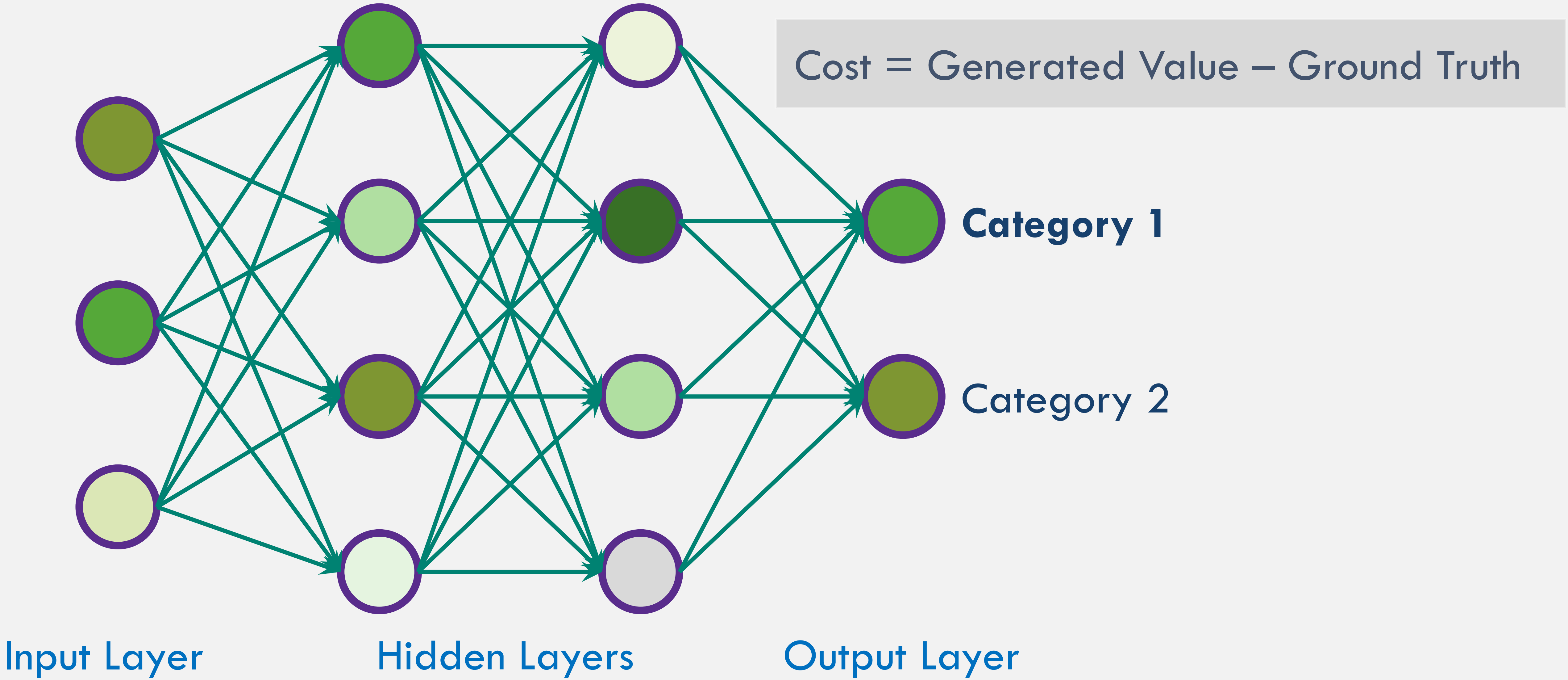
FORWARD PROPAGATION



FORWARD PROPAGATION

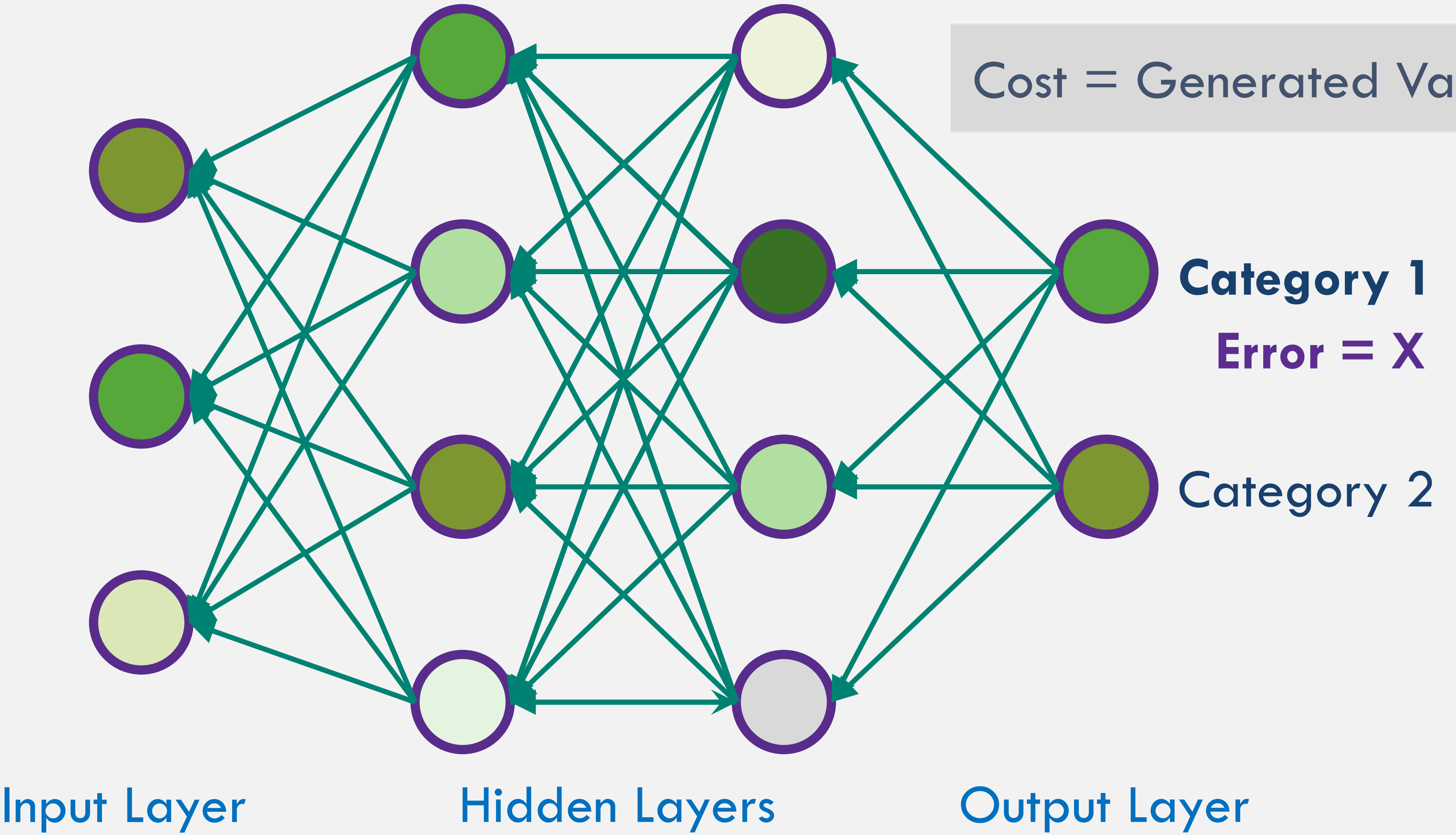


TRAINING


Cost = Generated Value – Ground Truth
Category 1
Category 2
Input Layer
Hidden Layers
Output Layer

BACKPROPAGATION

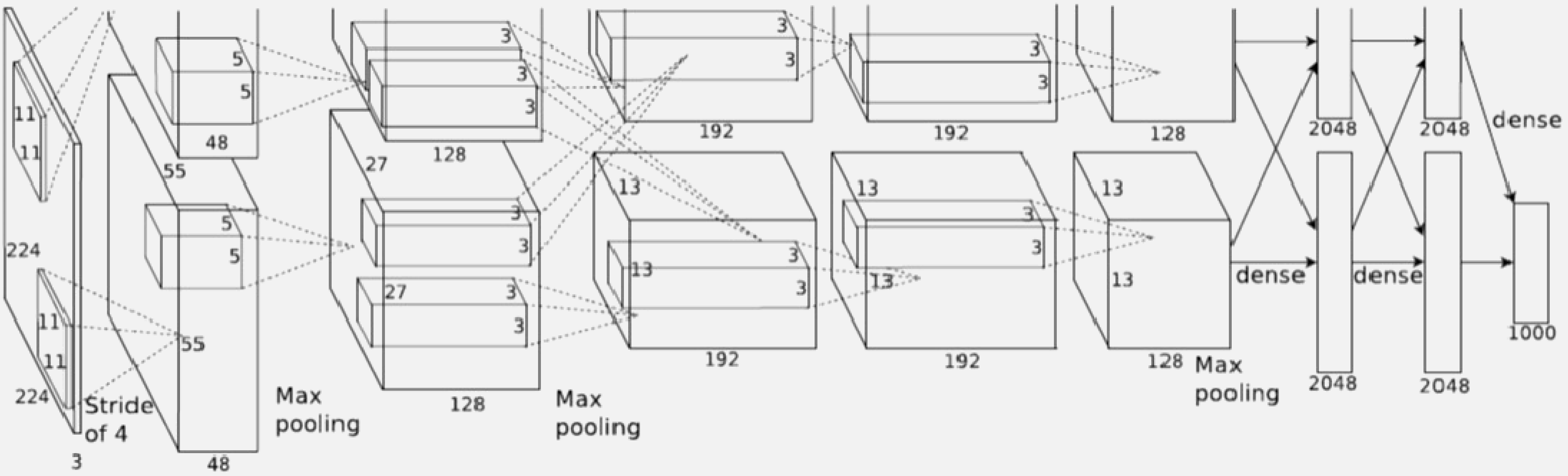
Cost = Generated Value – Ground Truth



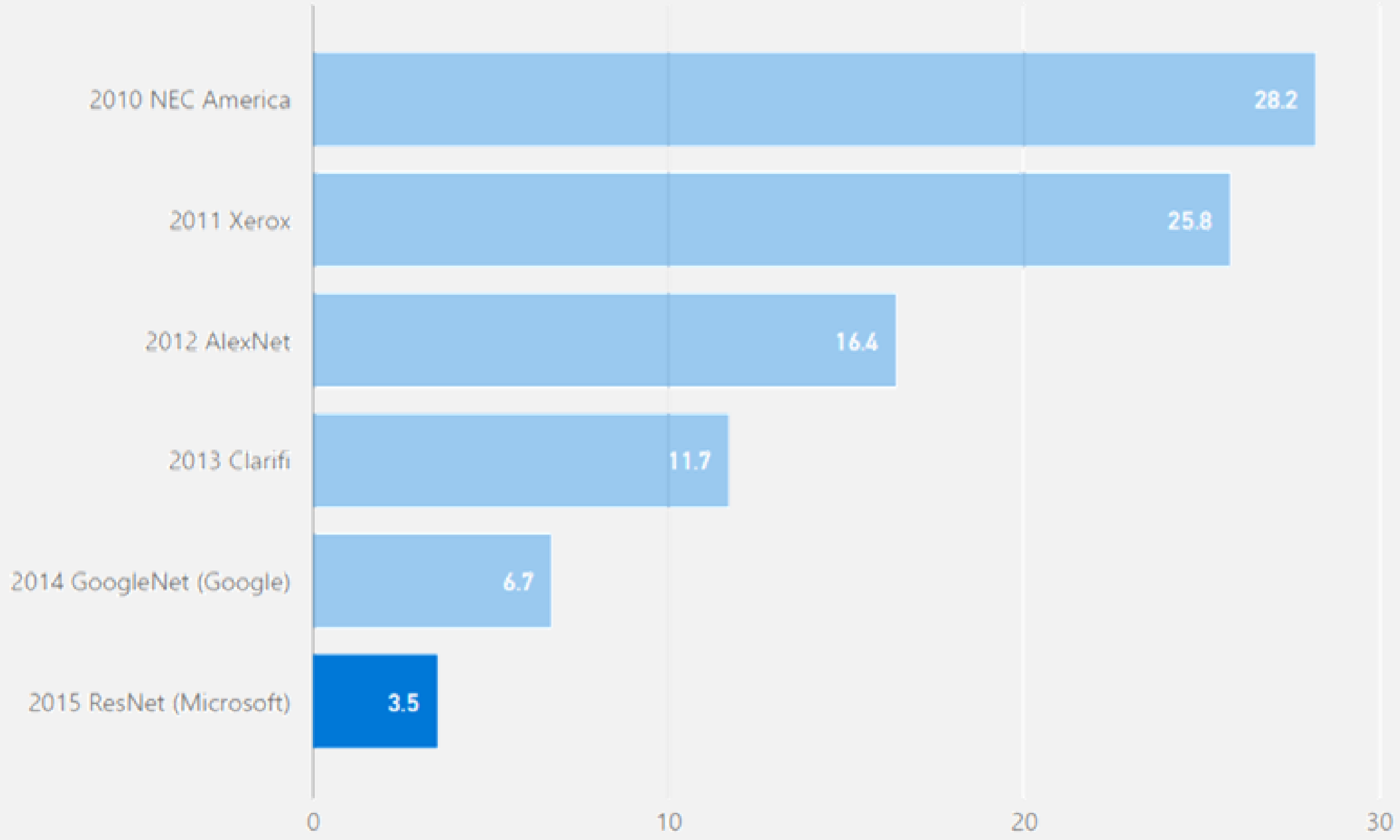
LAB 3: Multi-Layer Perceptron

Convolutional Networks

REMEMBER THIS THING?



IMAGENET CHALLENGE



FILTERING

0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0

						1		
	1				1			
		1	1		1			
				1				
			1		1	1		
			1				1	
		1						

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0
0	1	0	0	0	1	0	0	0
0	0	1	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	1	0	0
0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

FILTERING

…looking for common characteristics

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

1	1	1
1	1	1
1	1	1

1.0

Filter 2

-1	-1	1
-1	1	-1
1	-1	-1

-1	1	-1
1	1	1
-1	1	-1

0.11

Filter 3

1	-1	1
-1	1	-1
1	-1	1

1	1	-1
1	1	1
-1	1	1

0.55

CONVOLUTION

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Filter

1	-1	-1
-1	1	-1
-1	-1	1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

CONVOLUTION

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

$\otimes$

1	-1	-1
-1	1	-1
-1	-1	1

$=$

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

CONVOLUTION

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

$\otimes$

1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

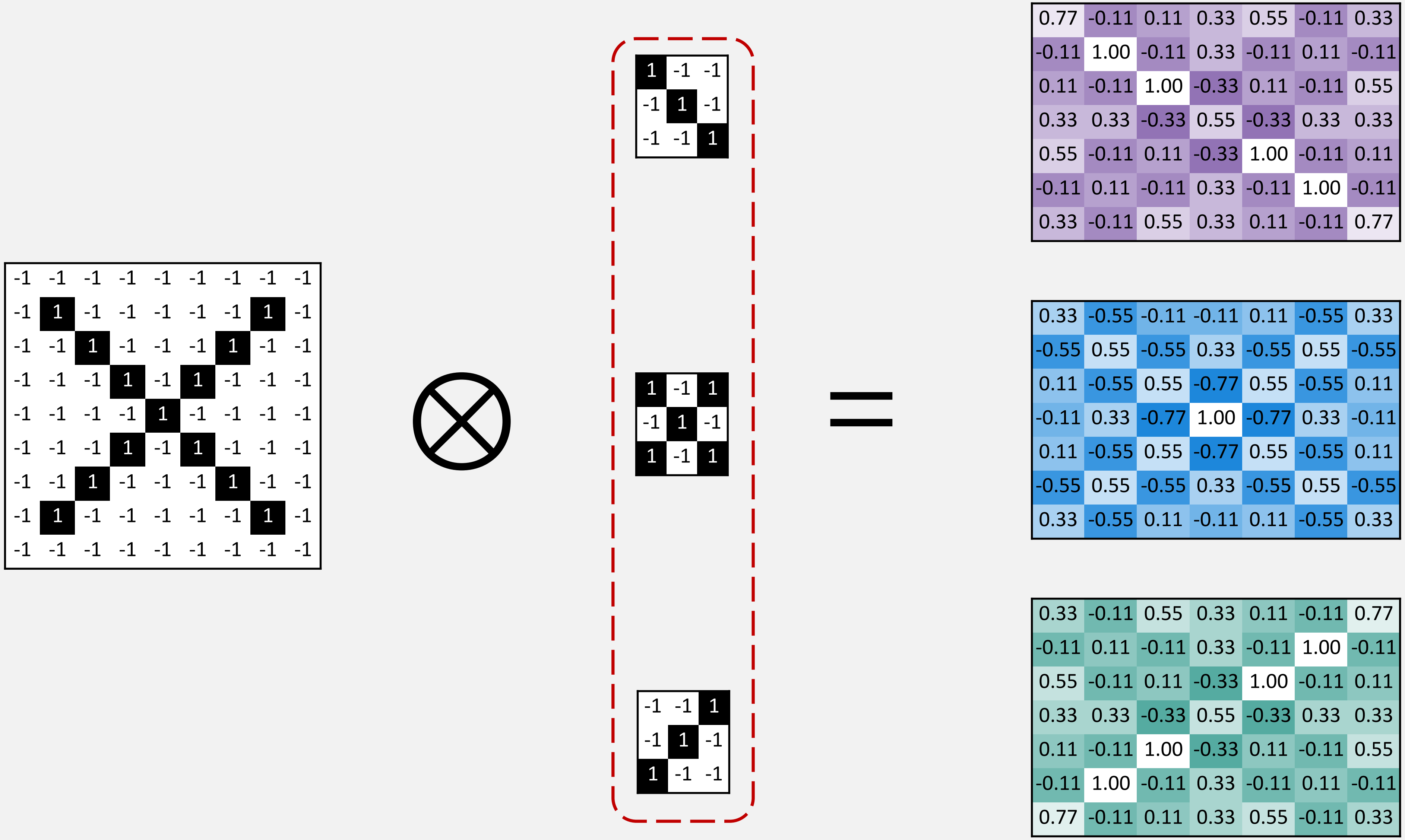
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

$\otimes$

-1	-1	1
-1	1	-1
1	-1	-1

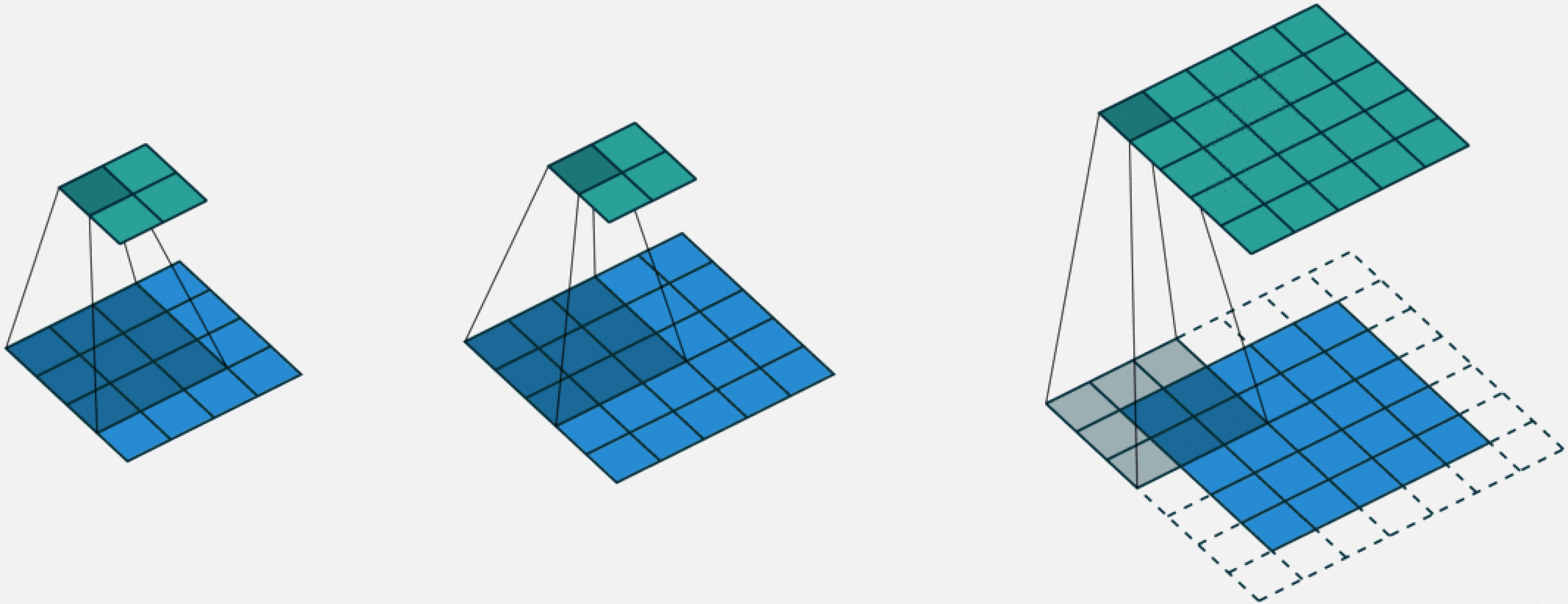
=

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33



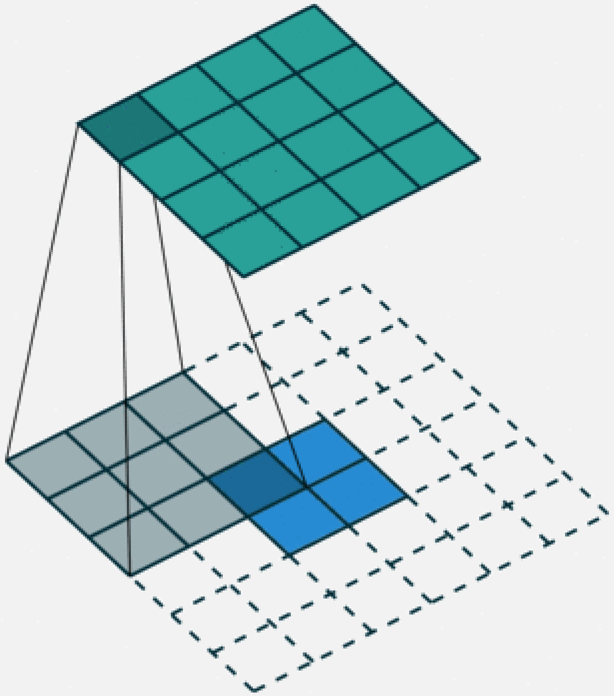
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
0.33 -0.55 -0.11 -0.11 0.11 -0.55 0.33
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.11 0.33 -0.77 1.00 -0.77 0.33 -0.11
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33

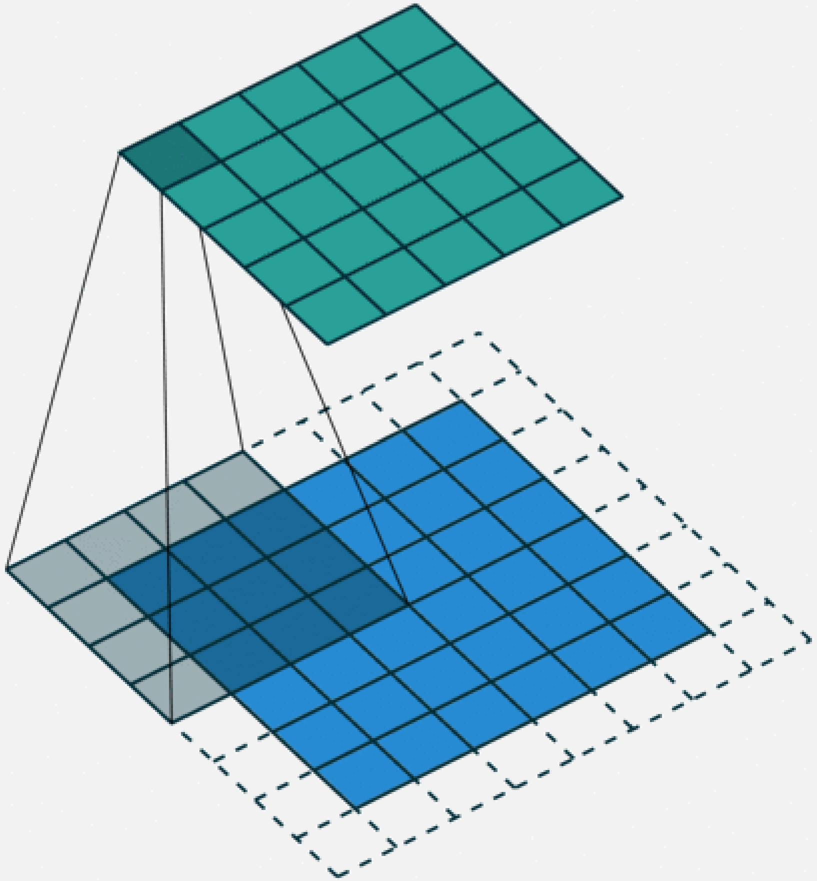
STRIDES AND PADDING



source: https://github.com/vdumoulin/conv_arithmetic

DECONVOLUTION – TRANSPOSED CONVOLUTION





source: https://github.com/vdumoulin/conv_arithmetic

POOLING

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

POOLING

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

AVERAGE POOLING – STRIDE 1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.39	0.22	0.17	0.28	0.11	0.06
0.22	0.45	0.22	0.00	0.00	0.11
0.17	0.22	0.22	0.00	0.00	0.28
0.28	0.00	0.00	0.22	0.22	0.17
0.11	0.00	0.00	0.22	0.45	0.22
0.06	0.11	0.28	0.17	0.22	0.39

MAX POOLING – STRIDE 1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	1.00	0.33	0.55	0.55	0.33
1.00	1.00	1.00	0.33	0.11	0.55
0.33	1.00	1.00	0.55	0.33	0.55
0.55	0.33	0.55	1.00	1.00	0.33
0.55	0.11	0.33	1.00	1.00	1.00
0.33	0.55	0.55	0.33	1.00	1.00

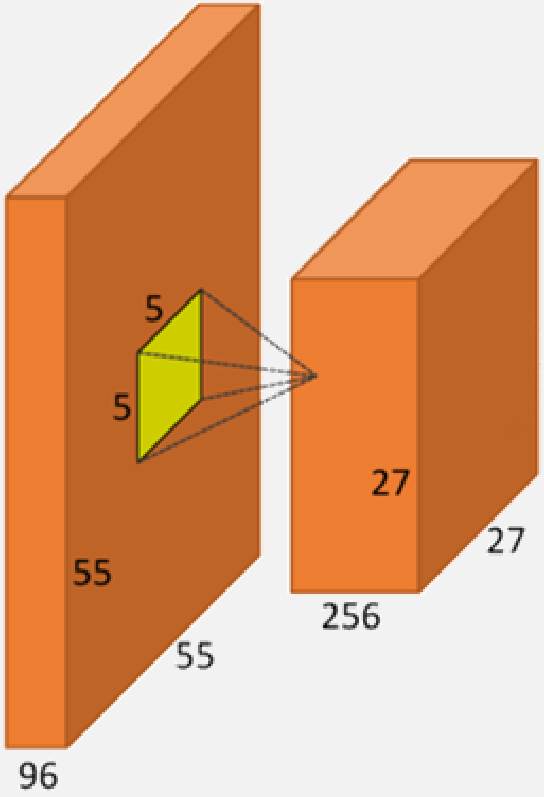
AVERAGE POOLING – STRIDE 2

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

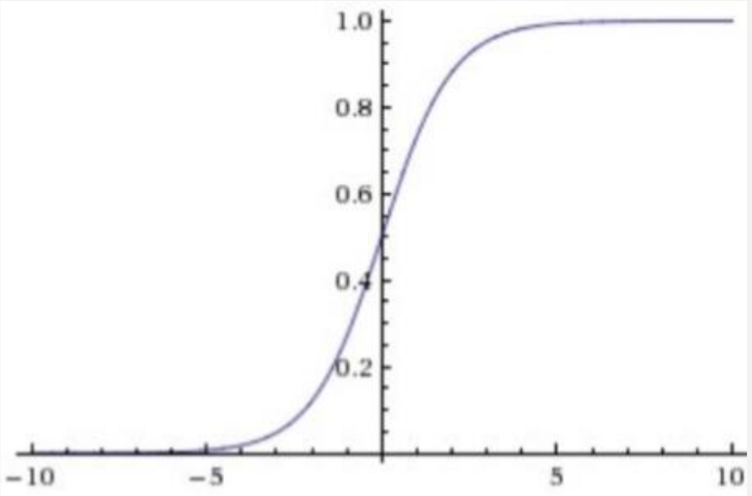
1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

WHY POOLING?

ACTIVATION FUNCTIONS?



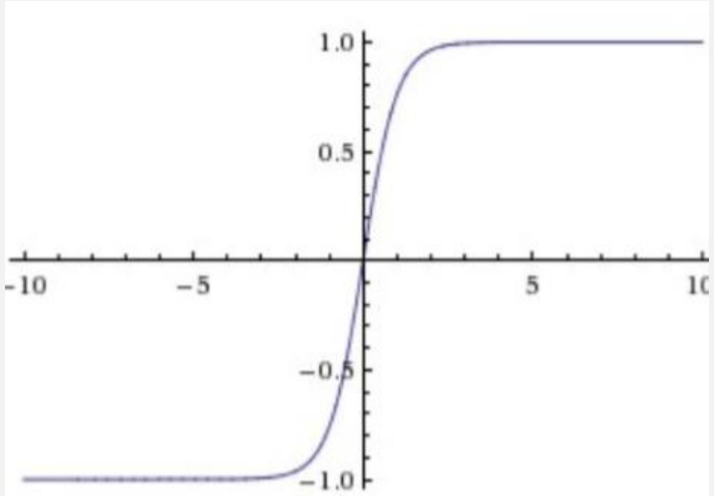
ACTIVATION FUNCTIONS?



LOGISTIC FUNCTION ("*sigmoid*"):

- Simple
- Drawbacks:
  - Saturates
  - Non-centered in zero
- We don't use it anymore, but we thank her for the services provided to the cause :)

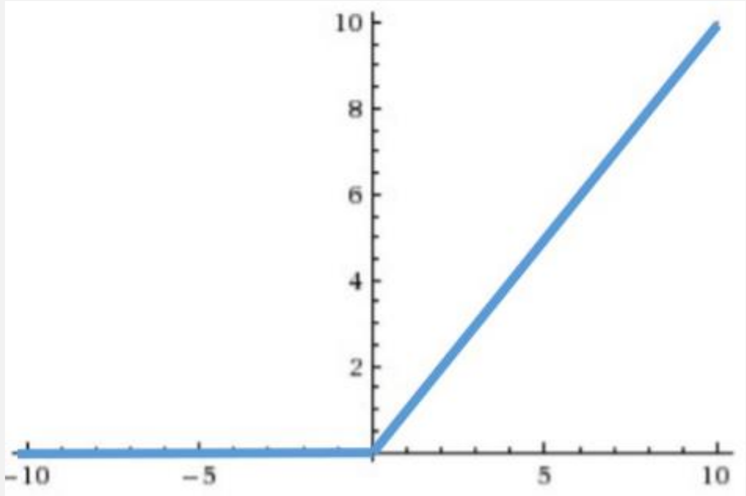
ACTIVATION FUNCTIONS?



HYPERBOLIC TANGENT (*tanh*):

- Solves the issue with not being centered in zero
- Still 'killing gradients' due to saturating behaviouir
- It is always better than the logistic function

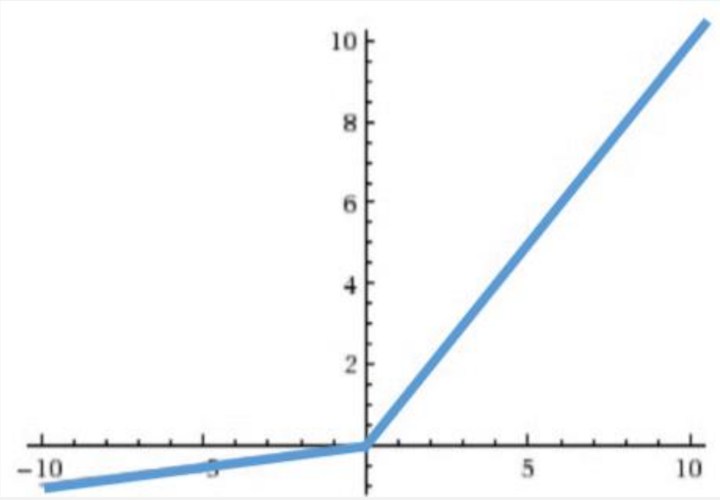
ACTIVATION FUNCTIONS?



RECTIFIED LINEAR UNITS (ReLU):

- Optimization for the SGD convergence
- More simple from a computational point of view
- Fragile

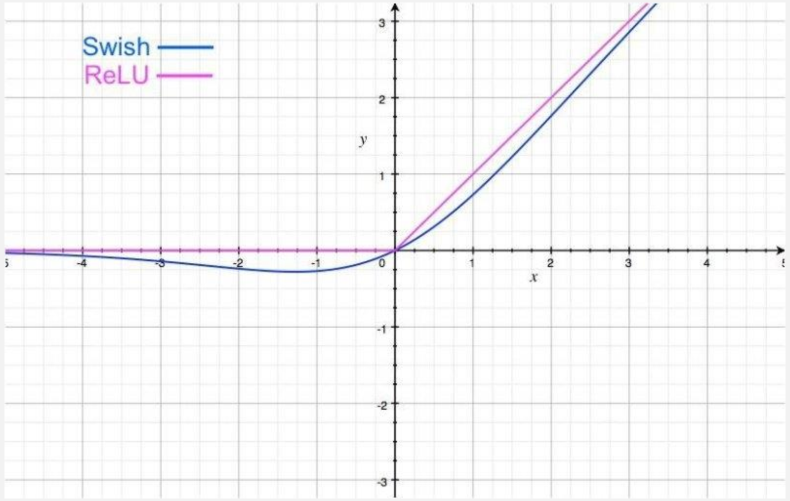
ACTIVATION FUNCTIONS?



LEAKY ReLU:

- More stable than traditional ReLu
- Particular case of *MaxOut*

ACTIVATION FUNCTIONS?



SWISH:

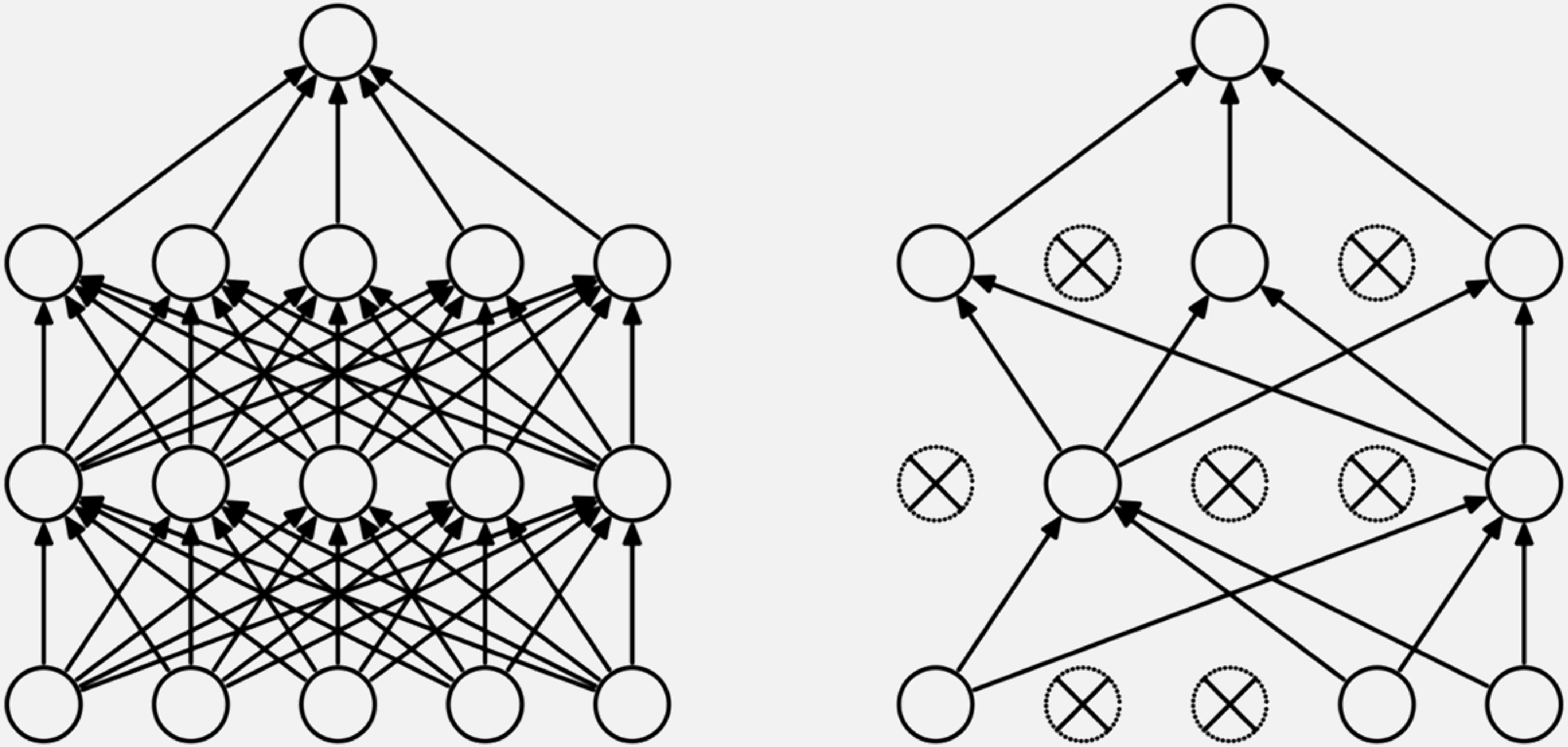
- Self-Gating activation function
- Better performance than ReLu

RECTIFIED LINEAR UNITS (ReLU)

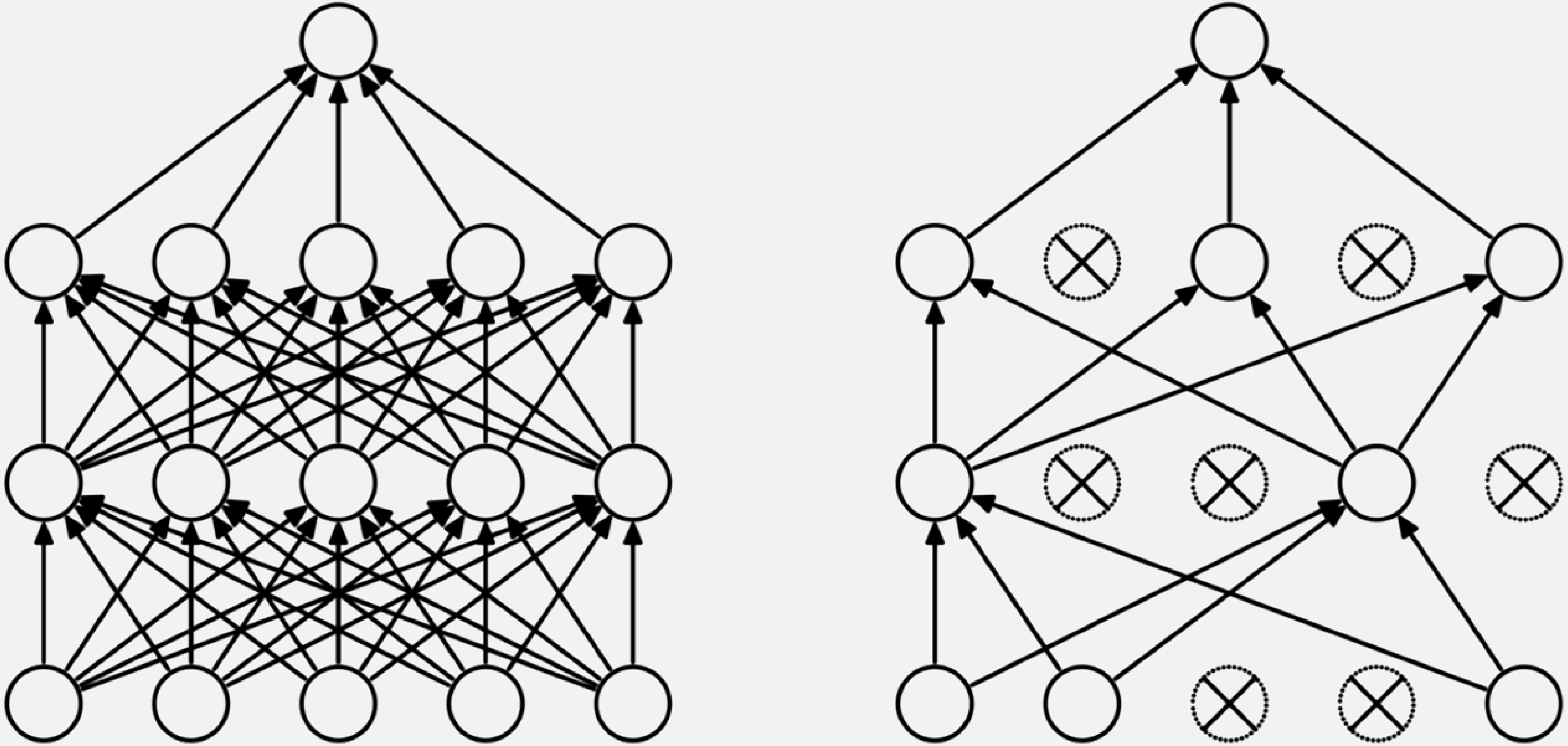
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0.00	0.11	0.33	0.55	0.00	0.33
0.00	1.00	0.00	0.33	0.00	0.11	0.00
0.11	0.00	1.00	0.00	0.11	0.00	0.55
0.33	0.33	0.00	0.55	0.00	0.33	0.33
0.55	0.00	0.11	0.00	1.00	0.00	0.11
0.00	0.11	0.00	0.33	0.00	1.00	0.00
0.33	0.00	0.55	0.33	0.11	0.00	0.77

DROPOUT



DROPOUT



WHAT IS MISSING?

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

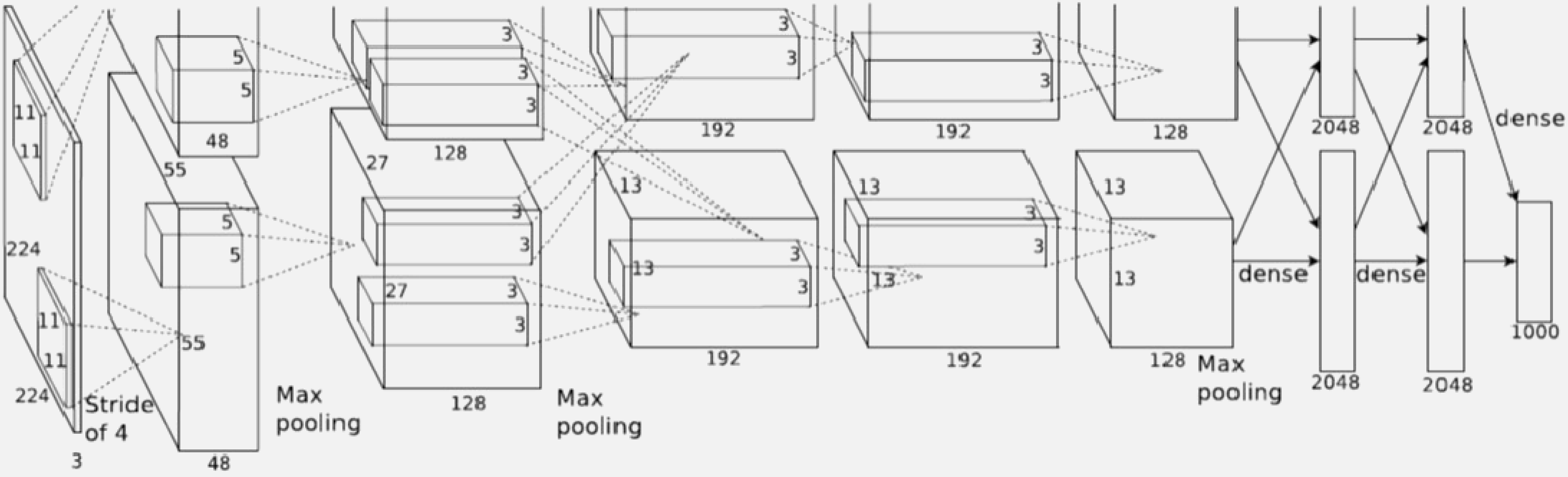
-1	-1	1
-1	1	-1
1	-1	-1

Filter 3

1	-1	1
-1	1	-1
1	-1	1

How have these filters been selected/created?

AlexNet

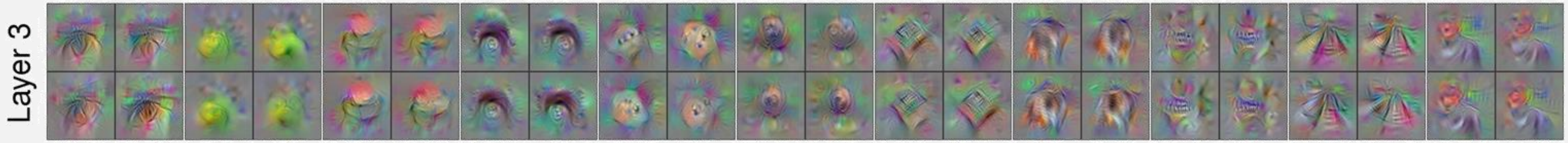


11
11
224
11
11
224
Stride
of 4
3
5
5
48
55
5
5
55
48
Max
pooling
3
3
3
27
128
3
3
27
3
3
128
Max
pooling
3
3
192
13
13
3
3
192
3
3
192
13
13
3
3
192
128
13
13
128
Max
pooling
dense
2048
2048
dense
2048
2048
dense
1000

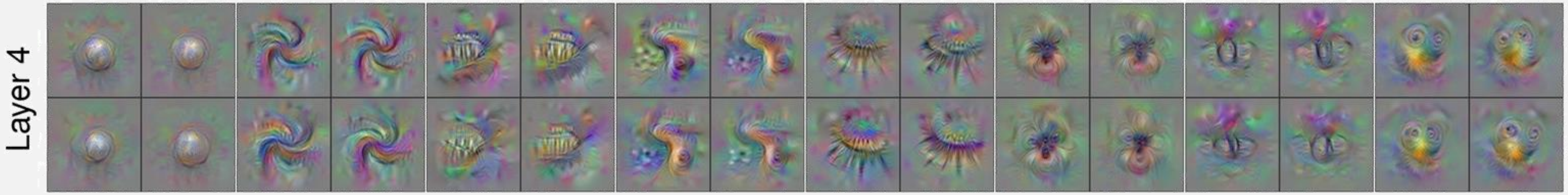
HOW A CNN SEES

Layer 2

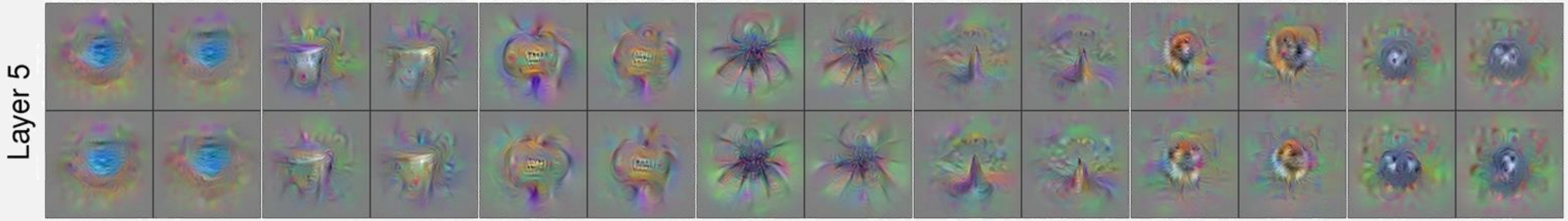
HOW A CNN SEES



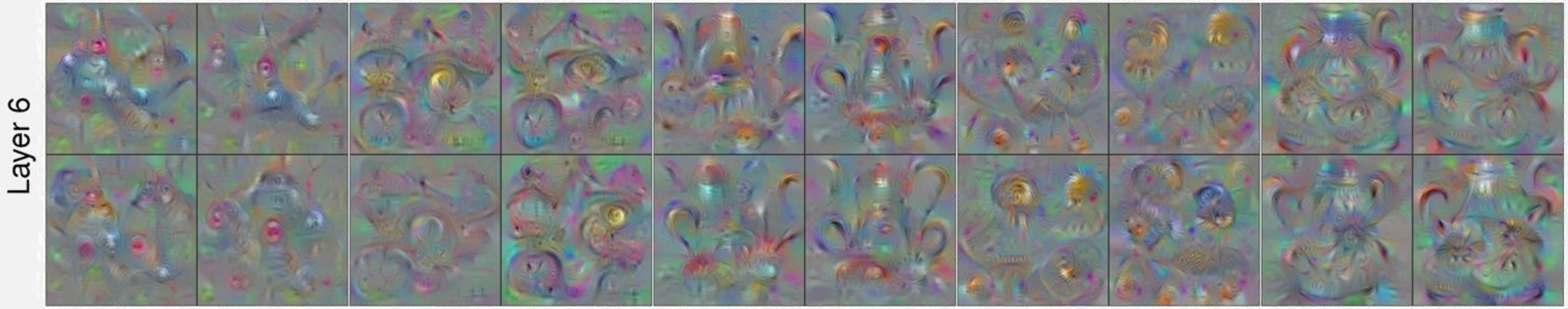
HOW A CNN SEES



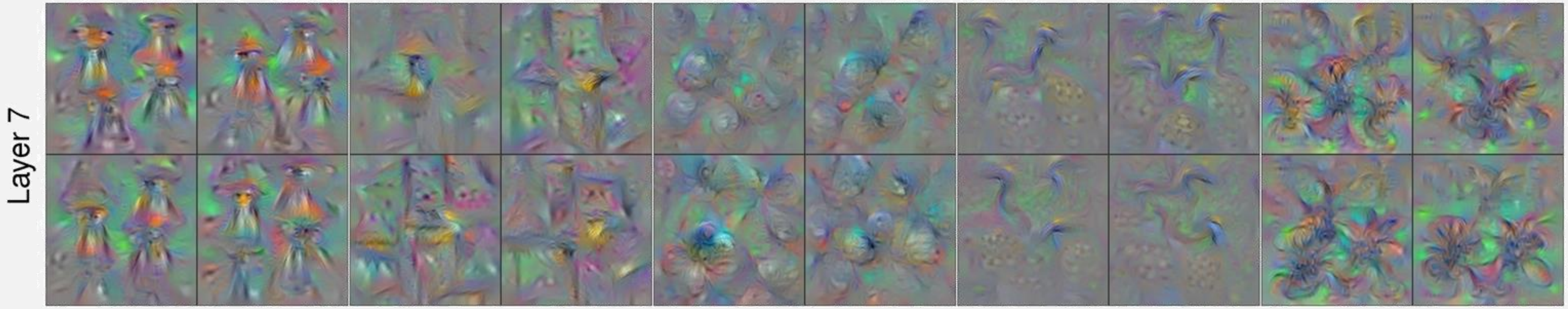
HOW A CNN SEES



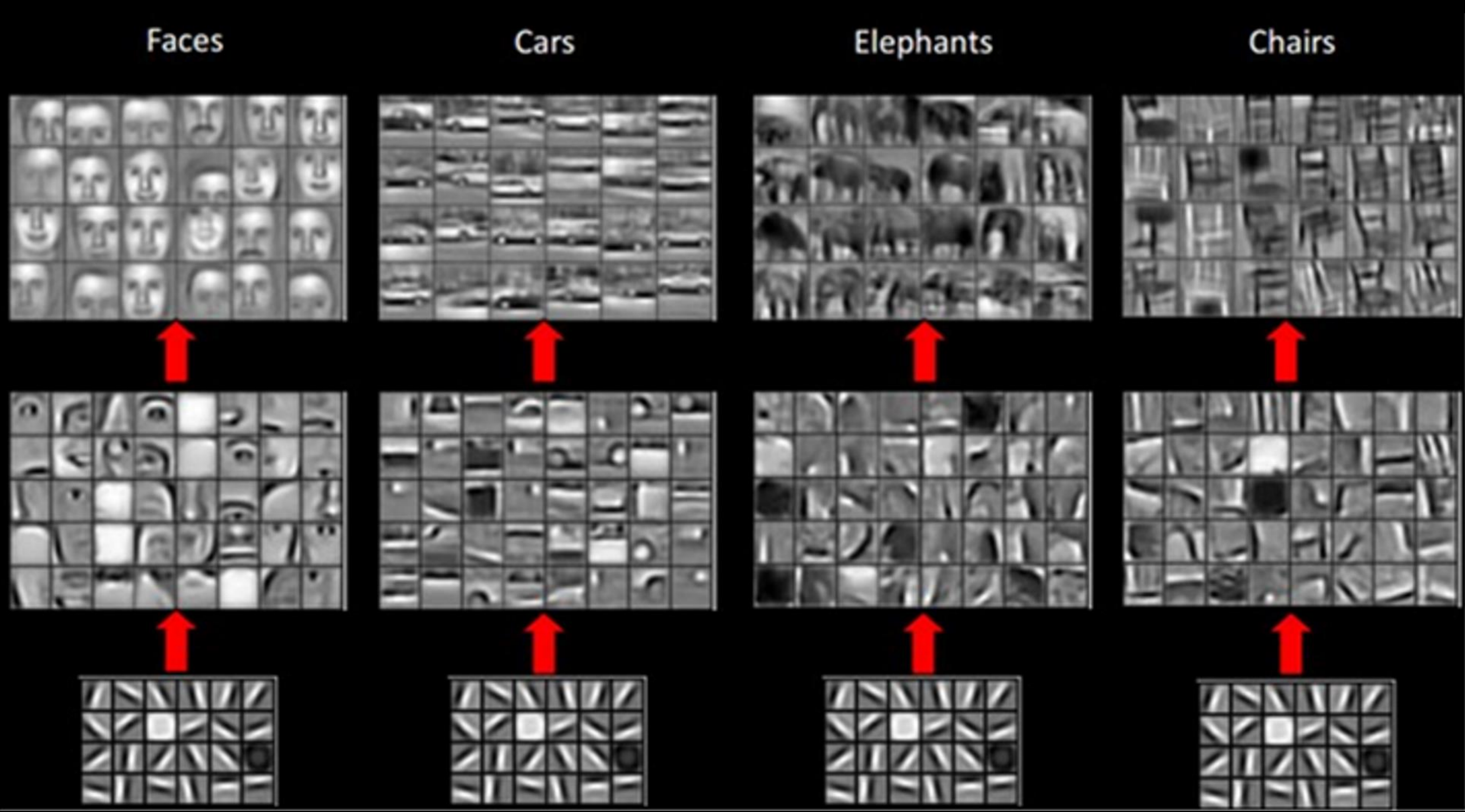
HOW A CNN SEES



HOW A CNN SEES



HOW A CNN SEES

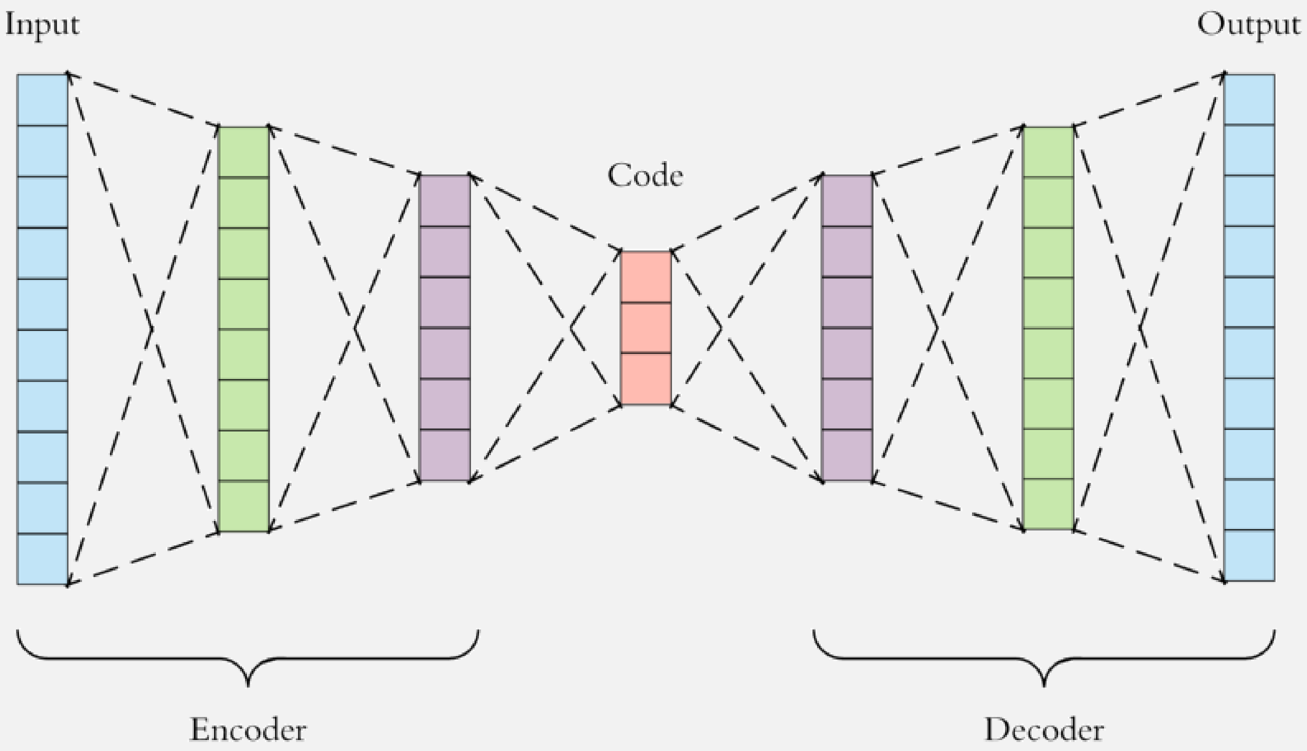

Faces
Cars
Elephants
Chairs

LAB 3: Convolutional Networks

DL: Autoencoders

BASIC AUTOENCODER

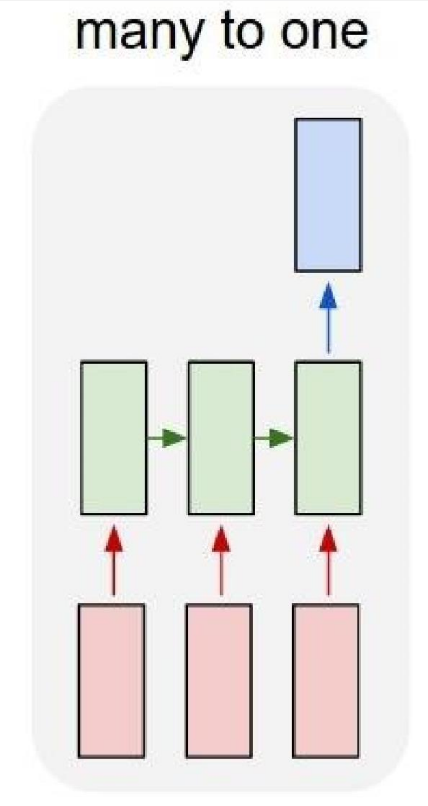
$$\hat{x} = h_{W,b} \approx x$$

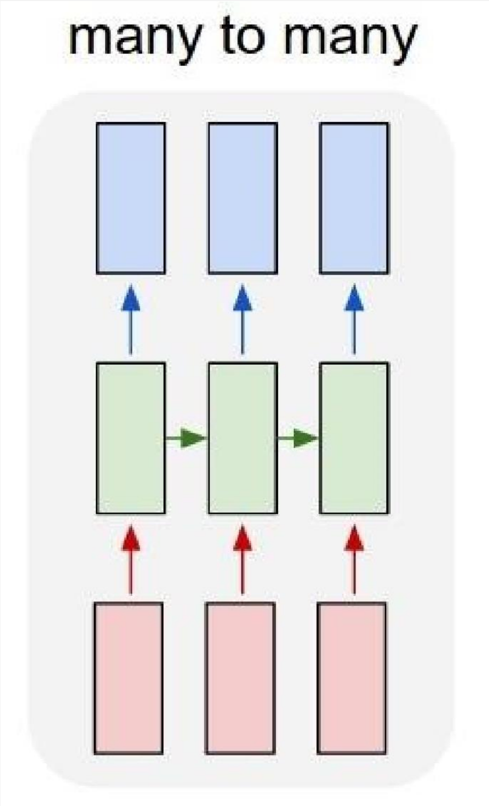


LAB 5: Autoencoder

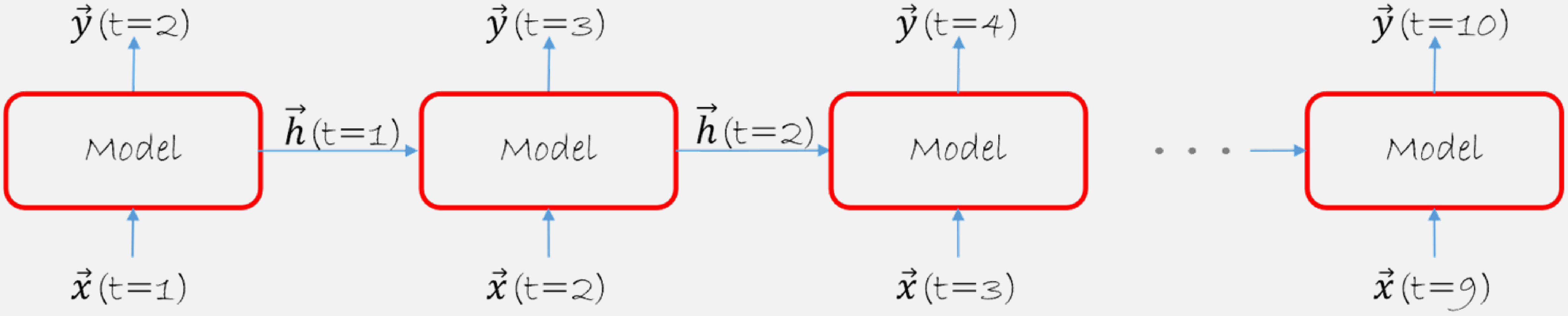
DL: Recurrent Neural Networks

HANDLING SEQUENCES

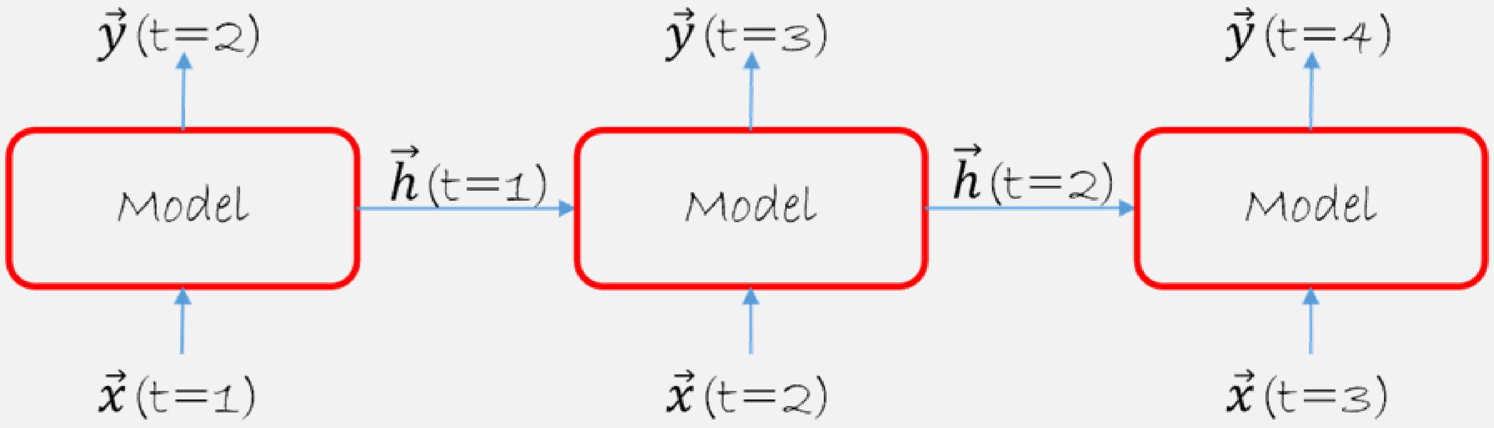




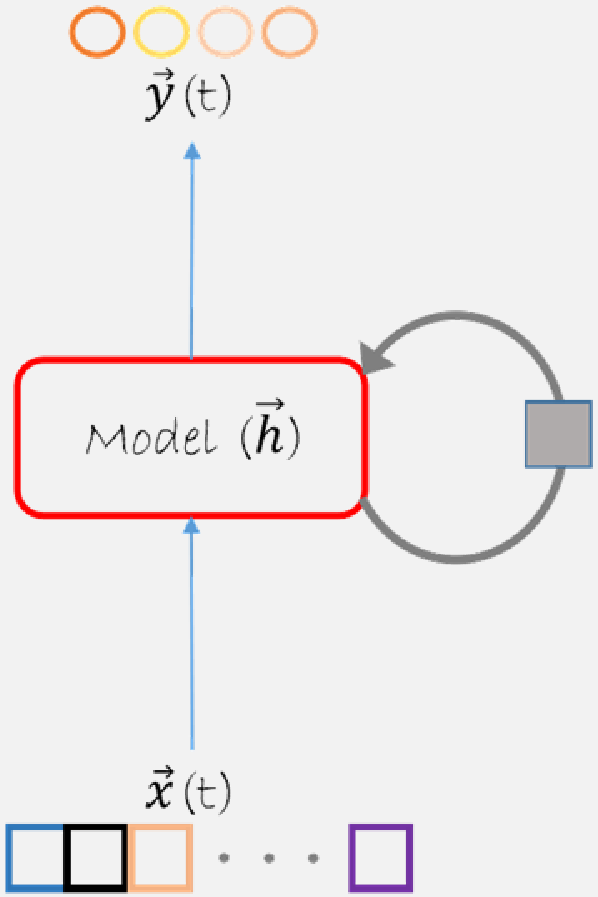
TRACKING THE HISTORY



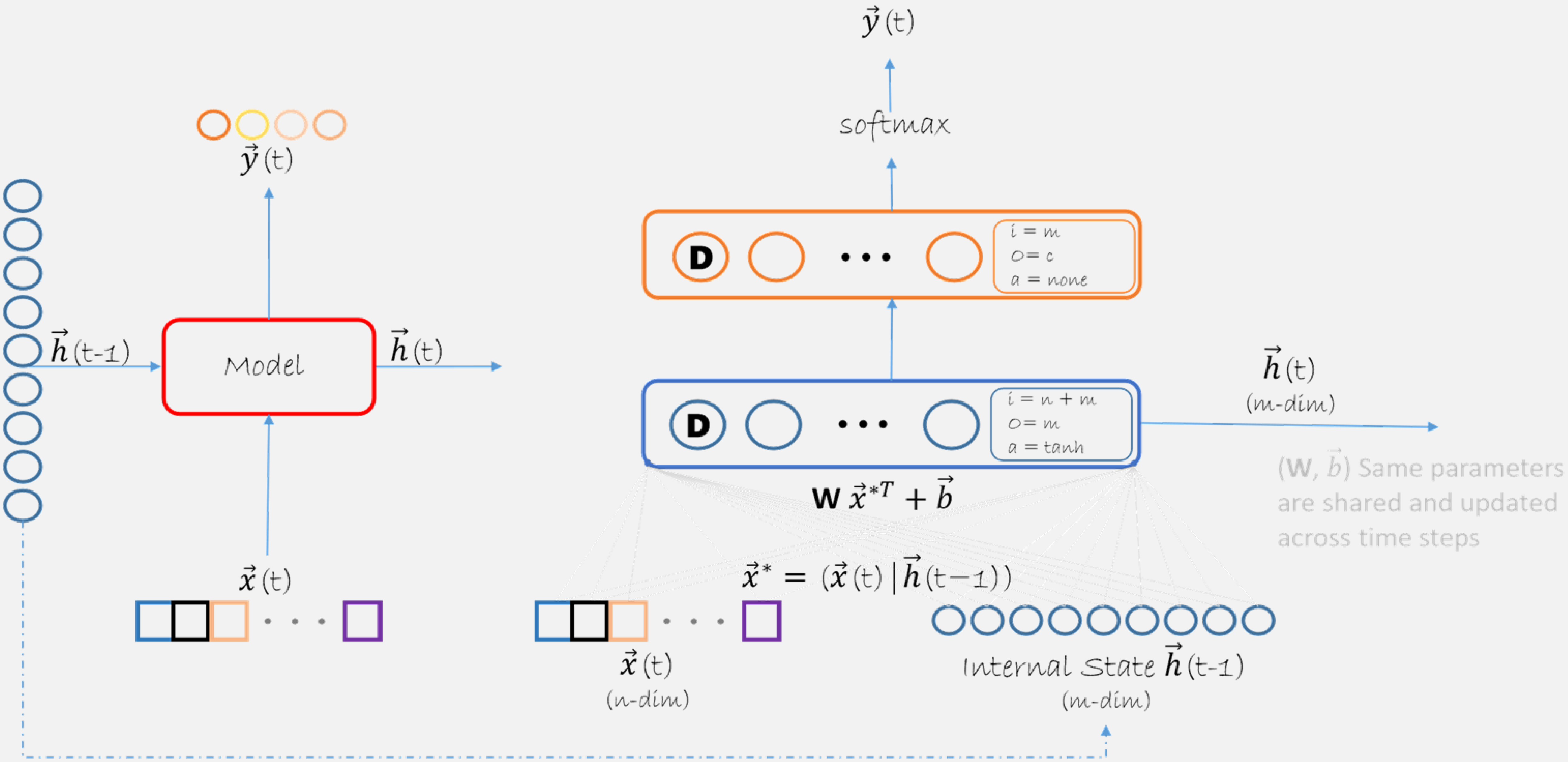
RECURRENCE



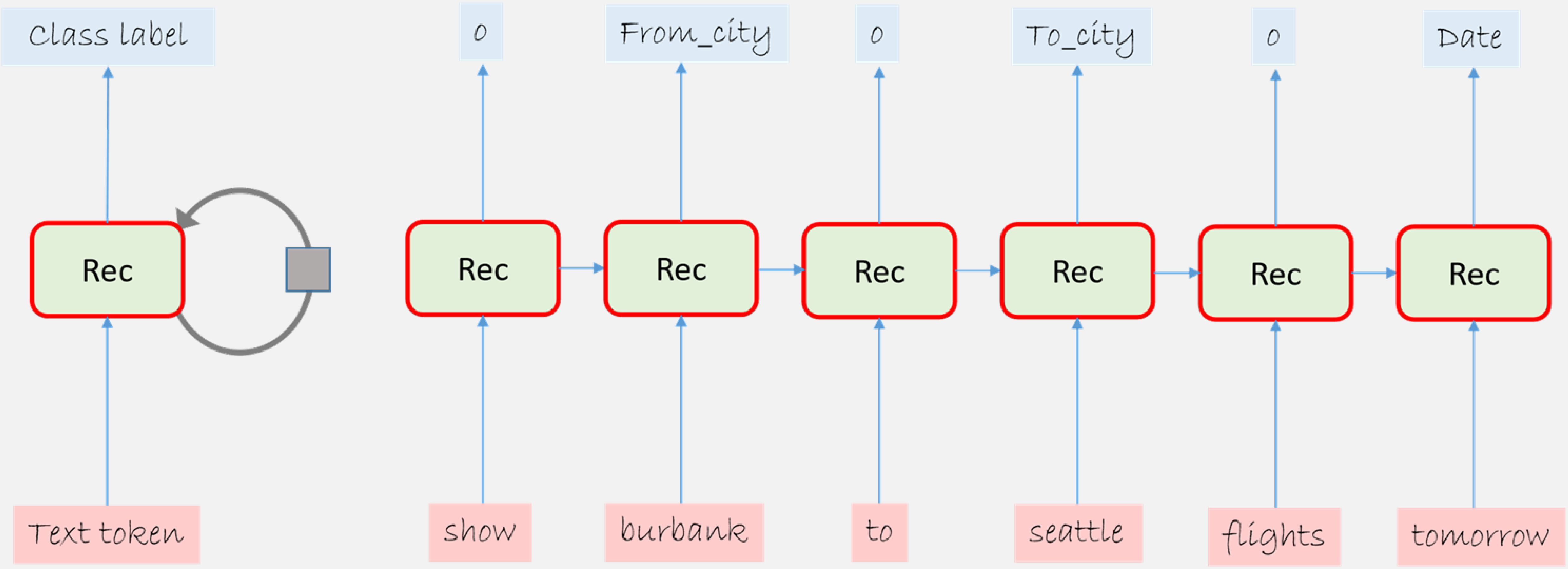
$\vec{y}$(t=2)
$\vec{y}$(t=3)
$\vec{y}$(t=4)
Model
$\vec{h}$(t=1)
Model
$\vec{h}$(t=2)
Model
$\vec{x}$(t=1)
$\vec{x}$(t=2)
$\vec{x}$(t=3)



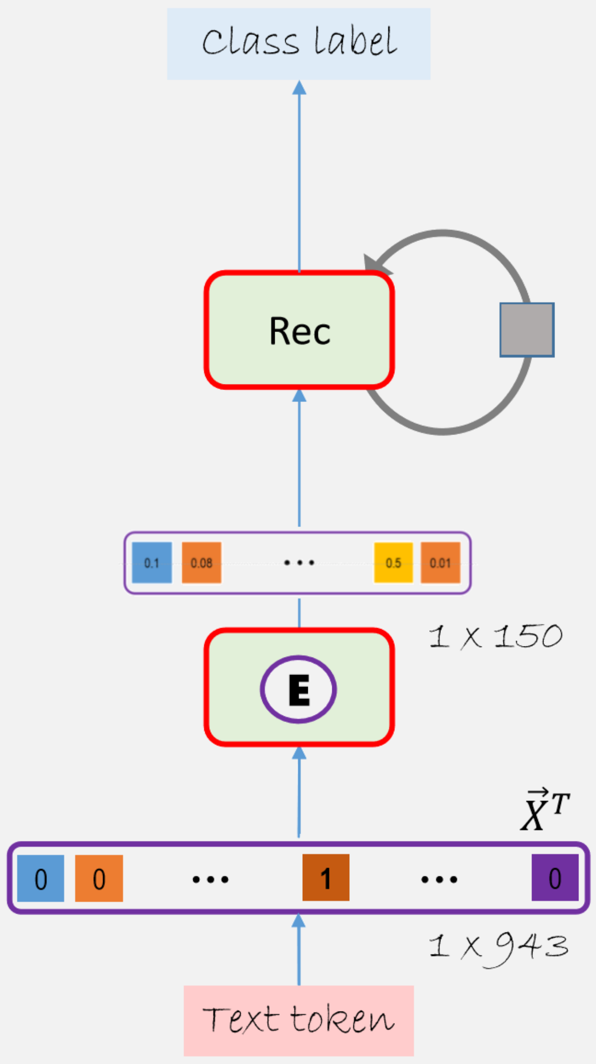
$\vec{y}$(t)
Model ($\vec{h}$)
$\vec{x}$(t)


$\vec{y}$(t)
$\vec{h}$(t-1)
Model
$\vec{h}$(t)
$\vec{x}$(t)
$\vec{y}$(t)
softmax
D
i = m
O= c
a = none
D
i = n + m
O= m
a = tanh
$\vec{h}$(t)
(m-dim)
$\mathbf{W}\,\vec{x}^{*T} + \vec{b}$
(W, $\vec{b}$) Same parameters are shared and updated across time steps
$\vec{x}^* = (\vec{x}(t)\,|\,\vec{h}(t-1))$
$\vec{x}$(t)
(n-dim)
Internal State $\vec{h}$(t-1)
(m-dim)

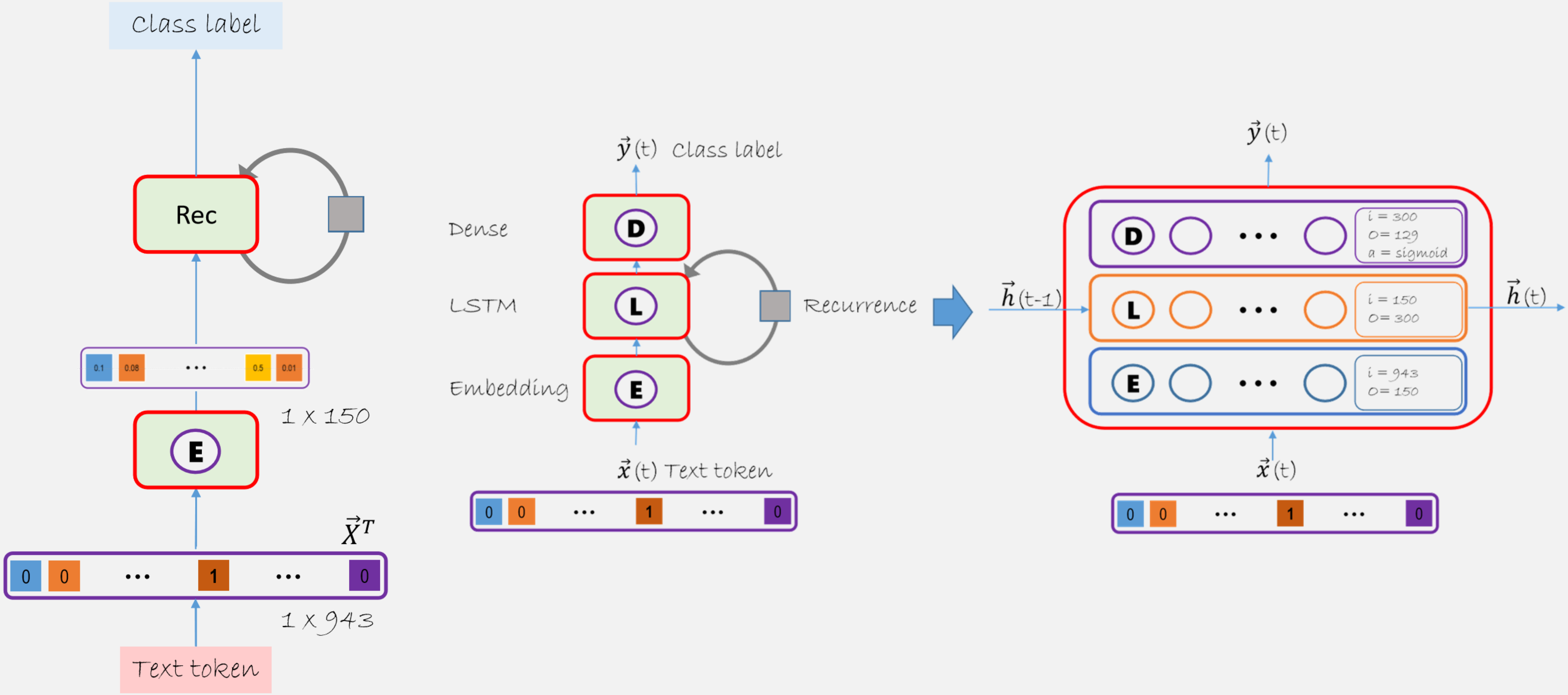
LONG SHORT TERM MEMORY



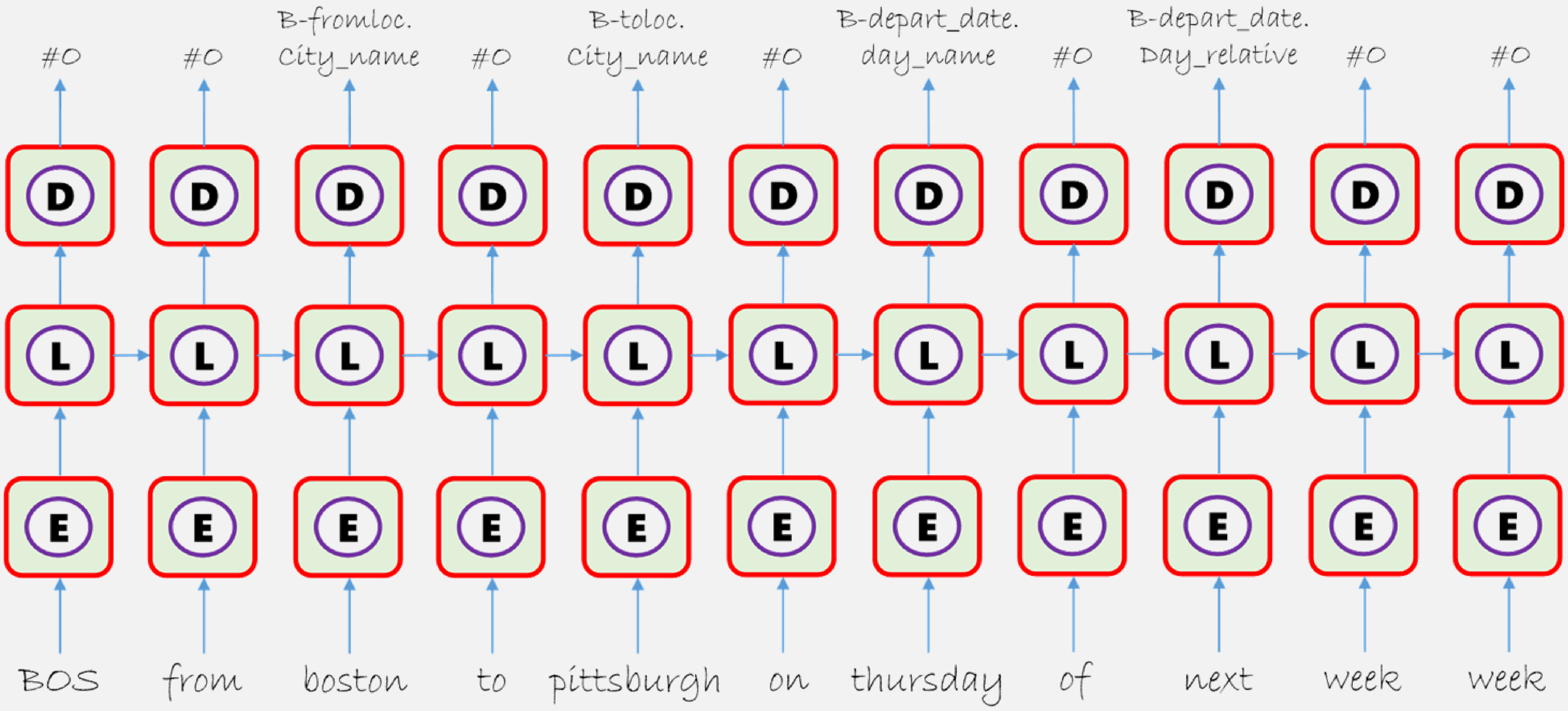
Sequence Id	Input Word (sample)	Word Index (in vocabulary) S0	Word Label	Label Index (S2)
19	# BOS	178:1	# O	128:1
19	# please	688:1	# O	128:1
19	# give	449:1	# O	128:1
19	# me	581:1	# O	128:1
19	# the	827:1	# O	128:1
19	# flights	429:1	# O	128:1
19	# from	444:1	# O	128:1
19	# boston	266:1	# B-fromloc.city_name	48:1
19	# to	851:1	# O	128:1
19	# pittsburgh	682:1	# B-toloc.city_name	78:1
19	# on	654:1	# O	128:1
19	# thursday	845:1	# B-depart_date.day_name	26:1
19	# of	646:1	# O	128:1
19	# next	621:1	# B-depart_date.date_relative	25:1
19	# week	910:1	# O	128:1
19	# EOS	179:1	# O	128:1



Class label
Rec
0.1
0.08
0.5
0.01
1 x 150
E
$\vec{X}^T$
0
0
1
0
1 x 943
Text token



Class label
Rec
0.1
0.08
0.5
0.01
E
1 x 150
$\vec{X}^T$
0
0
1
0
1 x 943
Text token
$\vec{y}$(t) Class label
Dense
D
LSTM
L
Recurrence
Embedding
E
$\vec{x}$(t) Text token
$\vec{y}$(t)
$\vec{h}$(t-1)
$\vec{h}$(t)
i = 300
o= 129
a = sigmoid
i = 150
o= 300
i = 943
o= 150
$\vec{x}$(t)


#O
#O
B-fromloc.
City_name
#O
B-toloc.
City_name
#O
B-depart_date.
day_name
#O
B-depart_date.
Day_relative
#O
#O
D
L
E
BOS
from
boston
to
pittsburgh
on
thursday
of
next
week
week

Some Advanced Ideas

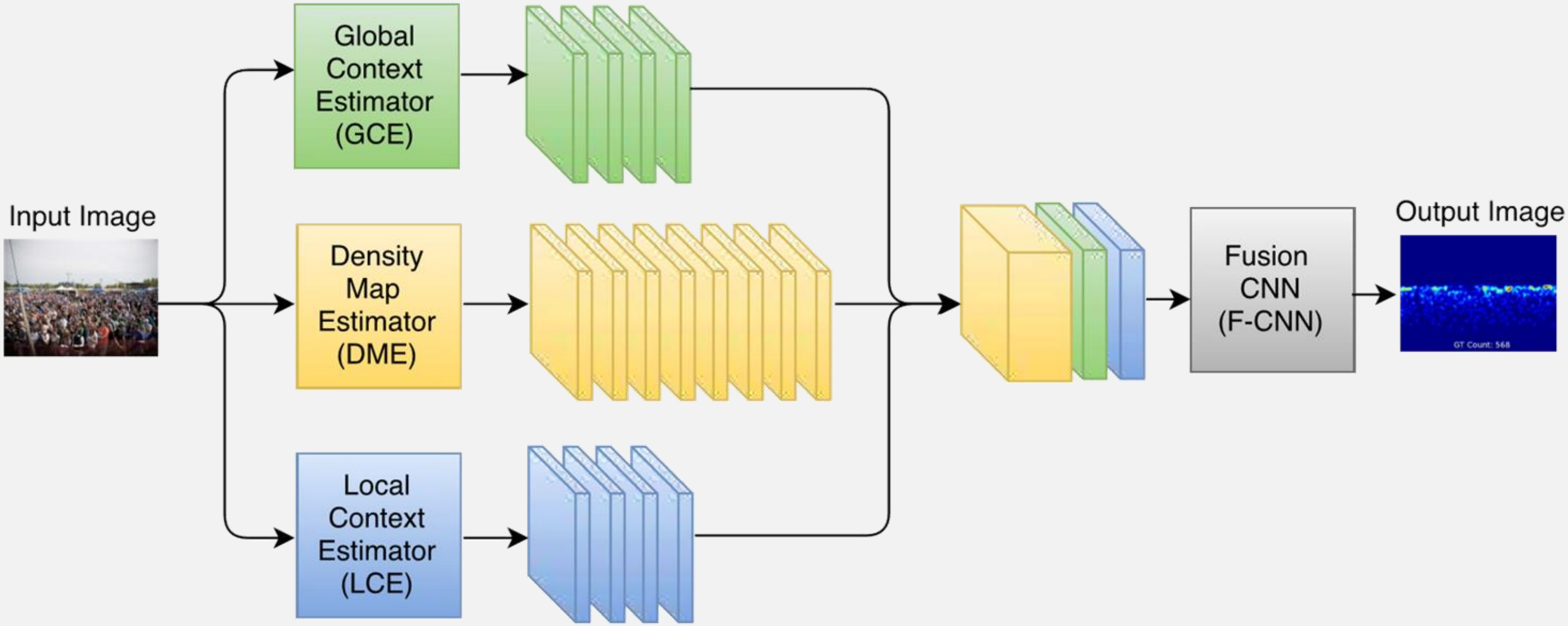
Counting people in a multitude

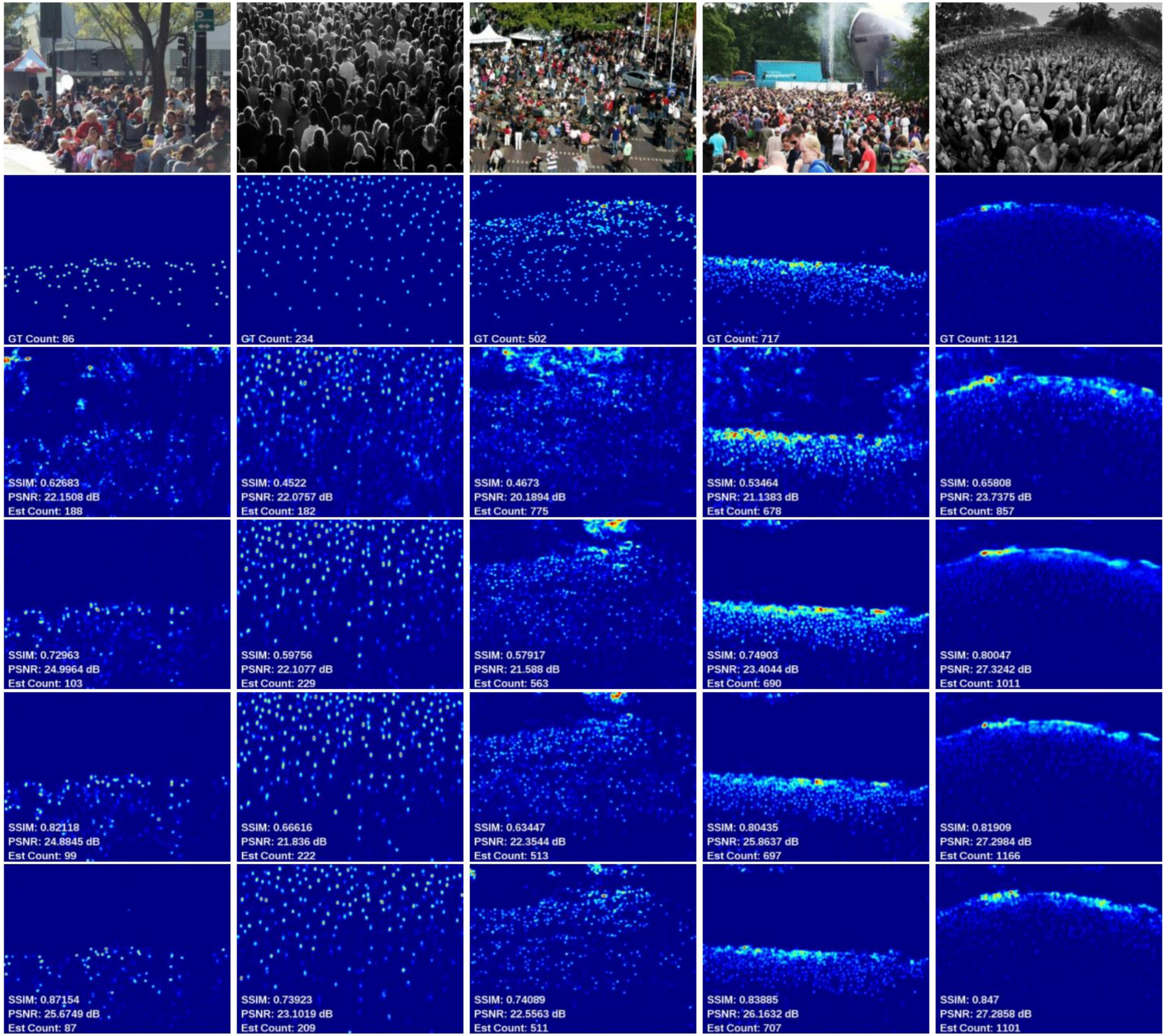
CONTEXTUAL PYRAMID CNN (CP-CNN)

A new method named Contextual Pyramid CNN (CP-CNN) is proposed here to generate density maps and influx estimations, by explicitely incorporating global and local context information. Composed of four modules: Global Context Estimator (GCE), Local Context Estimator (LCE), Density Map Estimator (DME) and a Fusion-CNN (F-CNN) convolutional network.

Vishwanath A. Sindagi, Vishal M. Patel; The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1861-1870

plain concepts


Input Image
Global Context Estimator (GCE)
Density Map Estimator (DME)
Local Context Estimator (LCE)
Fusion CNN (F-CNN)
Output Image
GT Count: 568


GT Count: 86
GT Count: 234
GT Count: 502
GT Count: 717
GT Count: 1121
SSIM: 0.62683
PSNR: 22.1508 dB
Est Count: 188
SSIM: 0.4522
PSNR: 22.0757 dB
Est Count: 182
SSIM: 0.4673
PSNR: 20.1894 dB
Est Count: 775
SSIM: 0.53464
PSNR: 21.1383 dB
Est Count: 678
SSIM: 0.65808
PSNR: 23.7375 dB
Est Count: 857
SSIM: 0.72963
PSNR: 24.9964 dB
Est Count: 103
SSIM: 0.59756
PSNR: 22.1077 dB
Est Count: 229
SSIM: 0.57917
PSNR: 21.588 dB
Est Count: 563
SSIM: 0.74903
PSNR: 23.4044 dB
Est Count: 690
SSIM: 0.80047
PSNR: 27.3242 dB
Est Count: 1011
SSIM: 0.82118
PSNR: 24.8845 dB
Est Count: 99
SSIM: 0.66616
PSNR: 21.836 dB
Est Count: 222
SSIM: 0.63447
PSNR: 22.3544 dB
Est Count: 513
SSIM: 0.80435
PSNR: 25.8637 dB
Est Count: 697
SSIM: 0.81909
PSNR: 27.2984 dB
Est Count: 1166
SSIM: 0.87154
PSNR: 25.6749 dB
Est Count: 87
SSIM: 0.73923
PSNR: 23.1019 dB
Est Count: 209
SSIM: 0.74089
PSNR: 22.5563 dB
Est Count: 511
SSIM: 0.83885
PSNR: 26.1632 dB
Est Count: 707
SSIM: 0.847
PSNR: 27.2858 dB
Est Count: 1101

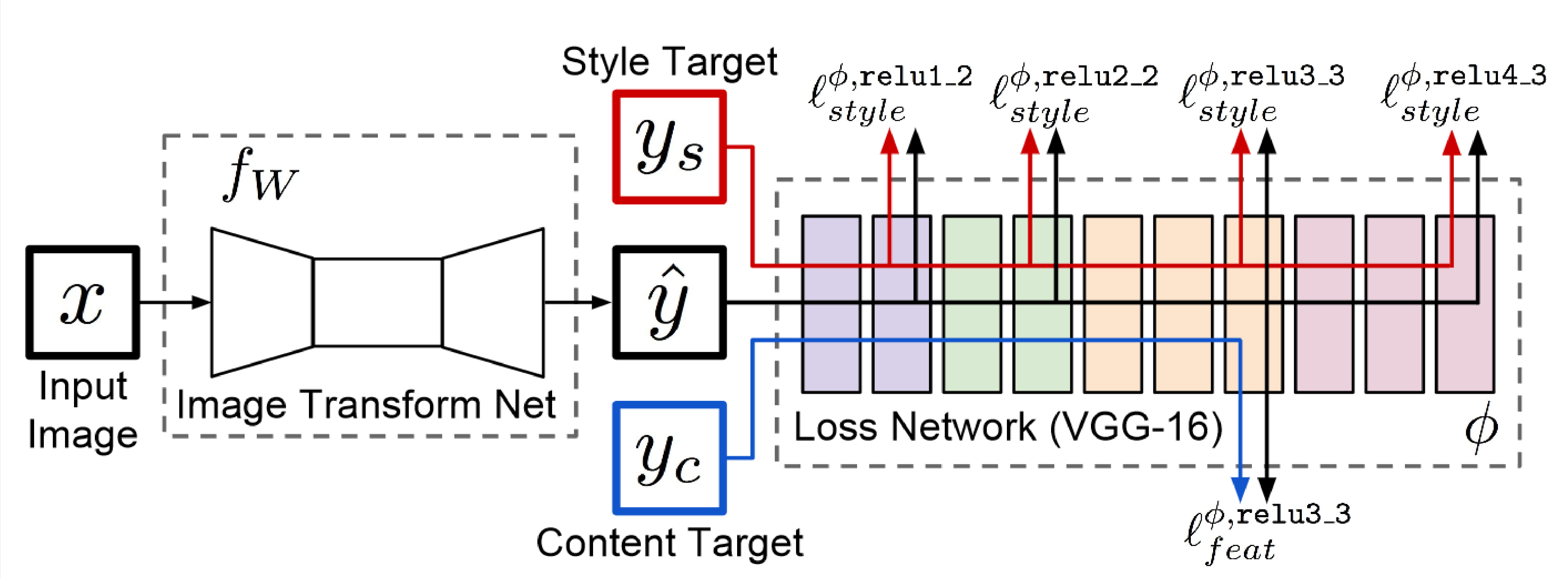
Transferring style between images

CONVOLUTIONAL NEURAL NETWORKS

By using a perceptual loss functions based on high-level features extracted from pretrained networks, networks for image transformation tasks can be trained, and by fine tuning the loss function different features can be kept for the source image and the style image.

Justin Johnson, Alexandre Alahi, Li Fei-Fei; Perceptual Losses for Real-Time Style Transfer and Super-Resolution, 2016

plain concepts



Style Target
$\ell_{style}^{\phi,relu1\_2}$
$\ell_{style}^{\phi,relu2\_2}$
$\ell_{style}^{\phi,relu3\_3}$
$\ell_{style}^{\phi,relu4\_3}$
f_W
y_s
x
$\hat{y}$
y_c
Input Image
Image Transform Net
Loss Network (VGG-16)
ϕ
Content Target
$\ell_{feat}^{\phi,relu3\_3}$


TELIO

Using GANs to drive design decisions

GENERATIVE ADVERSARIAL NETWORKS

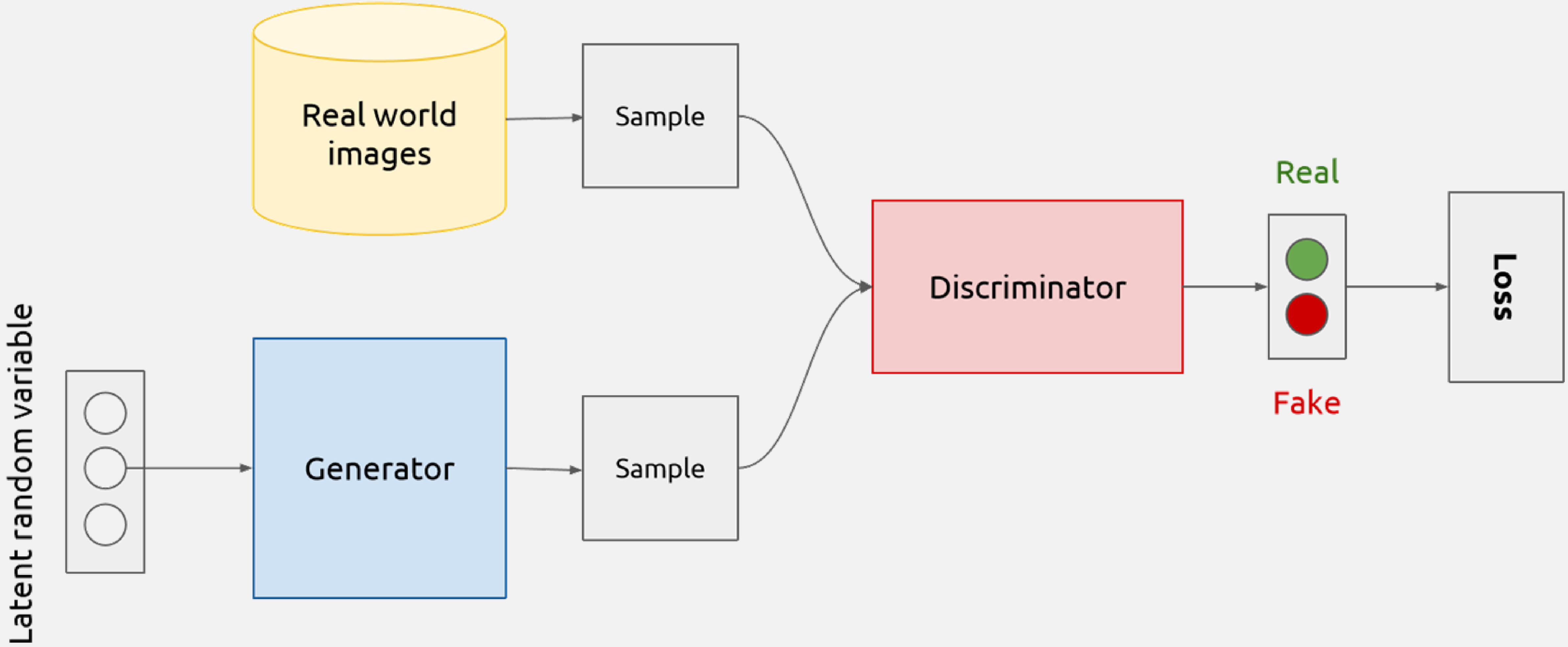
By taking advantage of Generational Adversarial Networks, synthetic images based on the training data can be generated. Including an external array of features, the generated images can be tailored to a specific set of requirements.

Jaime Deverall, Jiwoo Lee, Miguel Ayala; Using Generative Adversarial Networks to Design Shoes

plain concepts


Text description
This bird is blue with white and has a very short beak
This bird has wings that are brown and has a yellow belly
A white bird with a black crown and yellow beak
This bird is white, black, and brown in color, with a brown beak
The bird has small beak, with reddish brown crown and gray belly
This is a small, black bird with a white breast and white on the wingbars.
This bird is white black and yellow in color, with a short black beak
Stage-I images
Stage-II images


Input
Ground truth
Output


Real world images
Sample
Sample
Generator
Latent random variable
Discriminator
Real
Fake
Loss

DEEP LEARNING - MOVING FAST!

- Swish Activation Function
- Capsule Networks
- …

SOME INTERESTING PAPERS

"ImageNet Classification with Deep Convolutional Neural Netowrks"

[Krizhevsky, Sutskever, Hinton]

http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

"Visualizing and Understanding Convolutional Networks"

[Zeiler, Fergus]

arXiv: 1311.2901v3

"Optimization Methods for Large-Scale Machine Learning"

[Bottou, Curtis, Nocedal]

arXiv: 1606.04838v2

SOME INTERESTING PAPERS

"Swish: A Self-Gated Activation Function"

[Ramachandran, Zoph, Quoc V.]

arXiv: 1710.05941v2

"Dynamic Routing Between Capsules"

[Sabour, Frosst, Hinton]

arXiv: 1710.09829