# INTRO TO Deep Learning Focusing on Tools

Facebook.com/knowlexon
https://www.linkedin.com/company/knowlexon
info@knowlexon.com

Biswa G Singh
Acknowledgement: Subrat Panda

# SPEAKER SHORT BIO

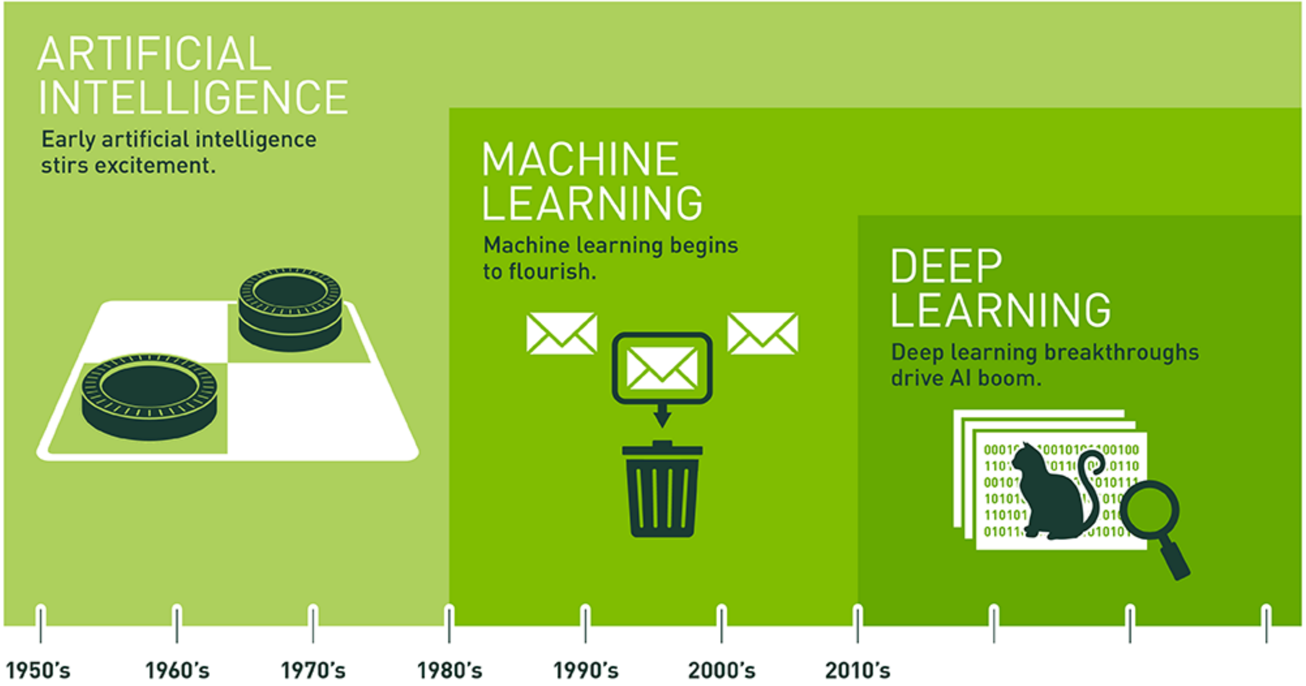
- MS (2010) – Clemson University
- Synopsys (EDA), IBM (CPU), ARM, Qualcomm, HP, AMD
- Lead ML Engineer at Capillary Technologies (We are Hiring)
- Deep Learning Performance on CPU/GPU
- Co-Founded IDLI (for social good) with Prof. Amit Sethi (IIT Bombay), **Subrat Panda (Capillary AI head, IIT KGP) and** Jacob Minz (Synopsys)
- https://www.facebook.com/groups/idliai/ (Consider Joining)
- Linked In - https://www.linkedin.com/in/biswagsingh/
- **Mentor** to knowlexon Innovation and technology Pvt Ltd(knowlexon.com)

info@knowlexon.com

info@knowlexon.com

# Preface

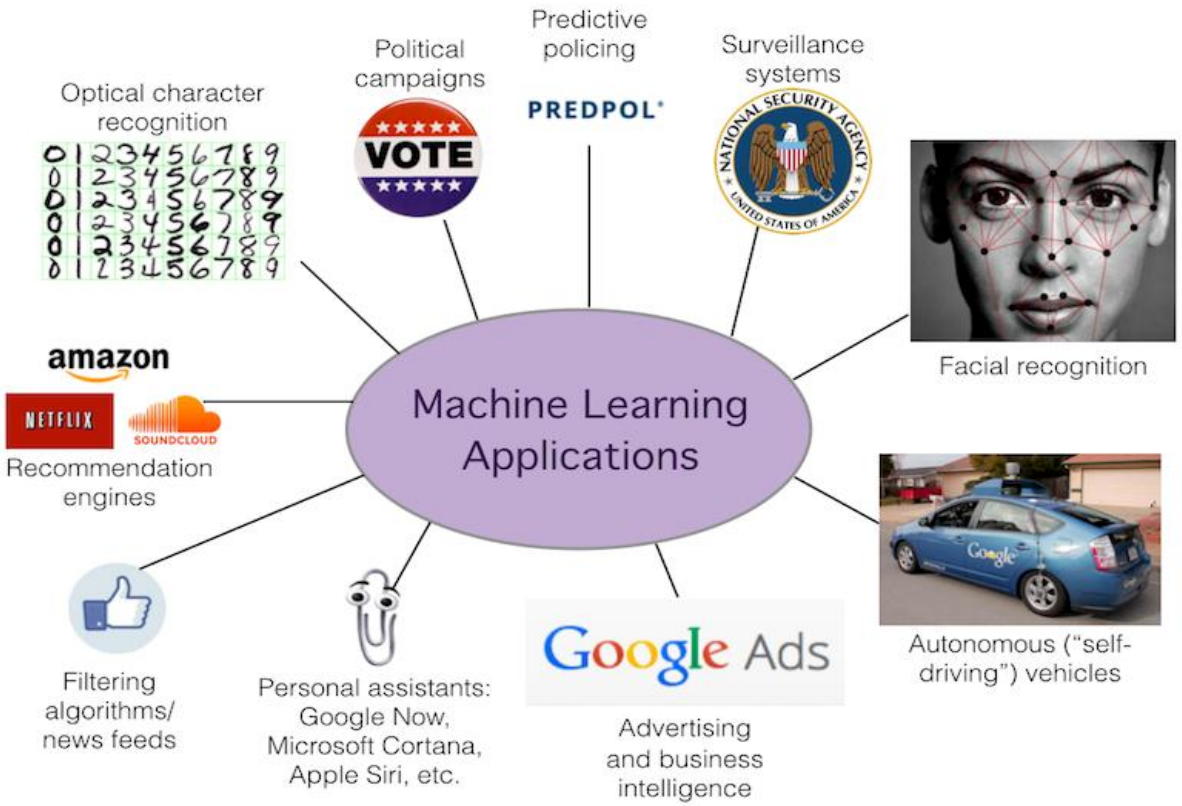
- Artificial intelligence is already part of our everyday lives.




Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

info@knowlexon.com

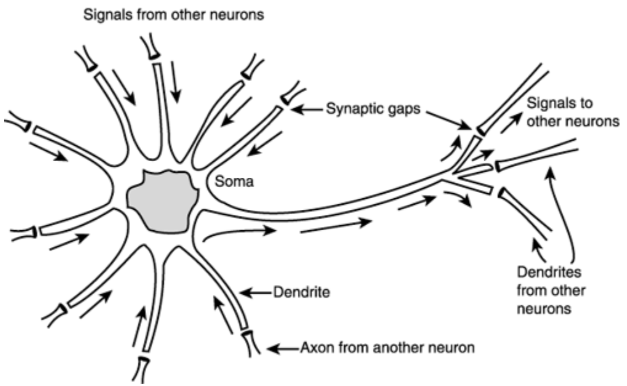
# Application of AI, Machine Learning and Deep Learning

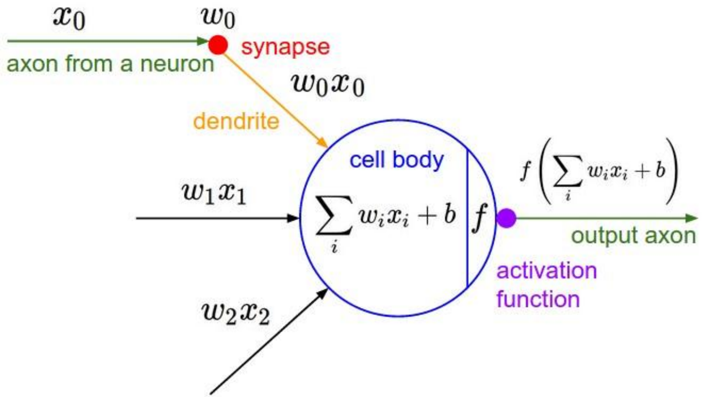


# Artificial Neural Networks

info@knowlexon.com

- The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living **neurons and dendrites.**
- **A Single Neuron:** The basic unit of computation in a neural network is the **neuron**, often called a **node** or **unit**.




- The **function f** is non-linear and is called the **Activation Function**.

info@knowlexon.com

# Activation Function

- **Sigmoid:** takes a real-valued input and squashes it to range between 0 and 1.

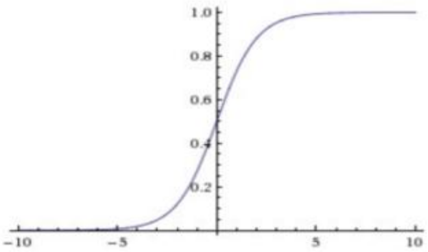
  $\sigma(x) = 1 / (1 + \exp(-x))$

- **tanh:** takes a real-valued input and squashes it to the range [-1, 1]

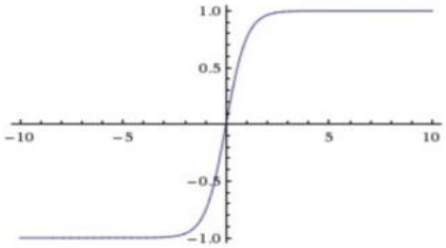
  $\tanh(x) = 2\sigma(2x) - 1$

- **ReLU**: ReLU stands for Rectified Linear Unit. It takes a real-valued input and thresholds it at zero (replaces negative values with zero)

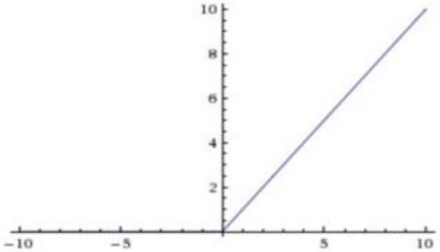
  $f(x) = \max(0, x)$




Sigmoid



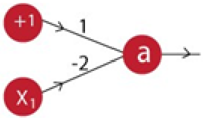

tanh



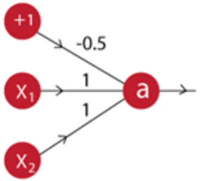

ReLU

# Neural Network Intuition (single layer)

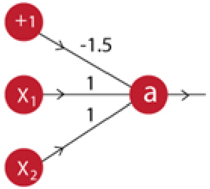
info@knowlexon.com



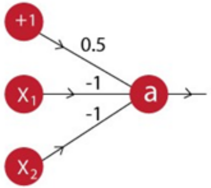

| X1 | NOT X1 | (1-2*X1) | a |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 0 | -1 | 0 |




| X1 | X2 | X1 OR X2 | (-0.5+X1+X2) | a |
|---|---|---|---|---|
| 0 | 0 | 0 | -0.5 | 0 |
| 0 | 1 | 1 | 0.5 | 1 |
| 1 | 0 | 1 | 0.5 | 1 |
| 1 | 1 | 1 | 1.5 | 1 |




| X1 | X2 | X1 AND X2 | (-1.5+X1+X2) | a |
|---|---|---|---|---|
| 0 | 0 | 0 | -1.5 | 0 |
| 0 | 1 | 0 | -0.5 | 0 |
| 1 | 0 | 0 | -0.5 | 0 |
| 1 | 1 | 1 | 0.5 | 1 |

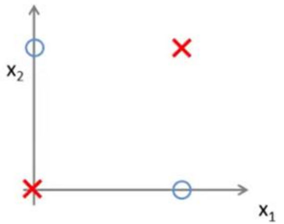


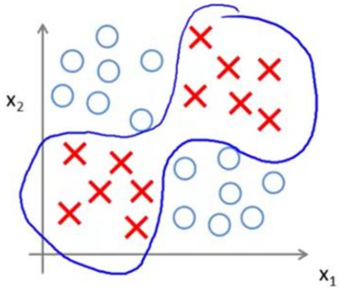

| X1 | X2 | X1' AND X2' | (0.5-X1-X2) | a |
|---|---|---|---|---|
| 0 | 0 | 1 | 0.5 | 1 |
| 0 | 1 | 0 | -0.5 | 0 |
| 1 | 0 | 0 | -0.5 | 0 |
| 1 | 1 | 0 | -1.5 | 0 |

info@knowlexon.com

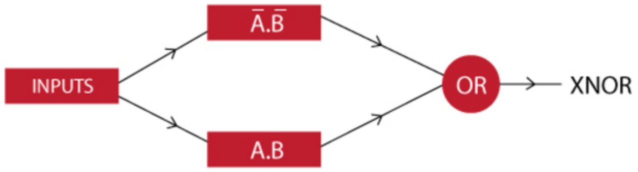
# Neural Network Intuition (Multiple Layer layer)

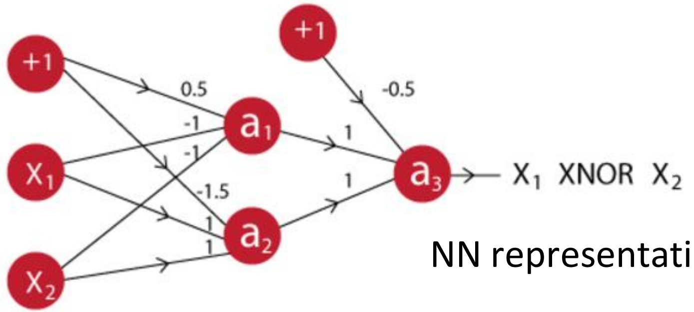
- Multi Layer Neural network is capable of learning complex functions.
- Lets consider XNOR operation.



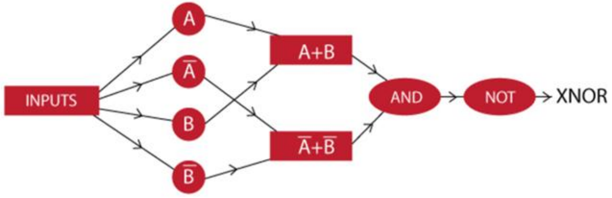

- CASE1: X1 XNOR X2 = (A’.B’) + (A.B)



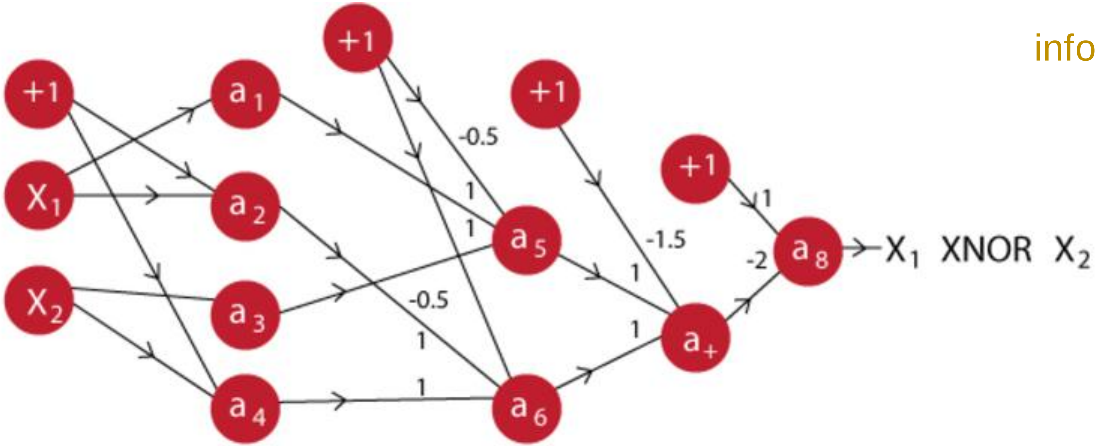

NN representation

- CASE2: X1 XNOR X2 = NOT [ (A+B).(A’+B’)




NN representation = ?

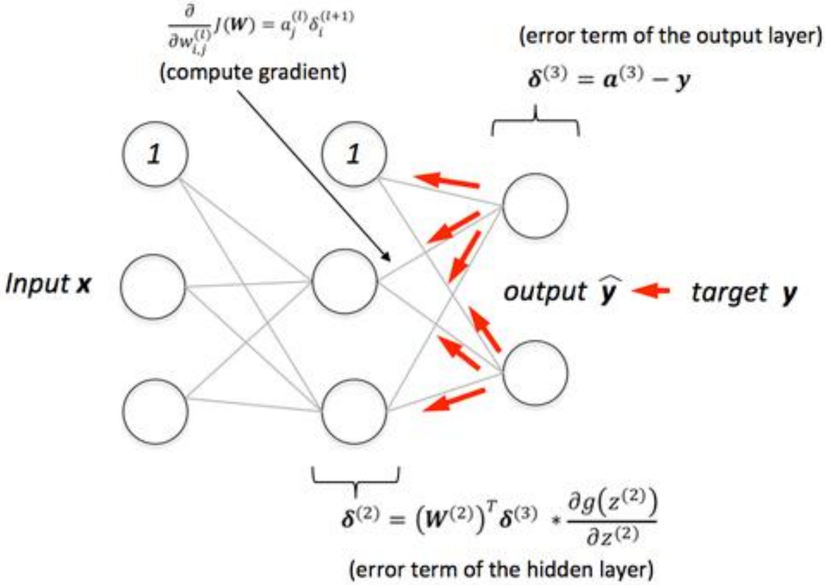
info@knowlexon.com




# Back-Propagation

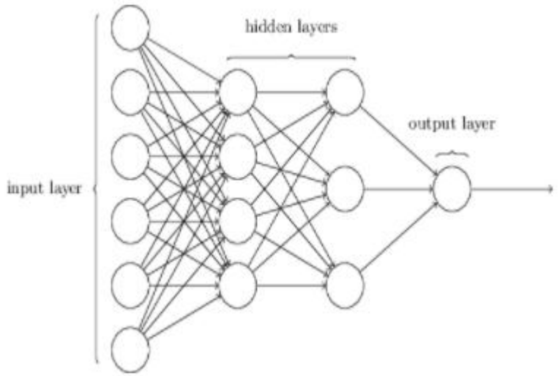
- Back-propagation (BP) algorithms works by determining the loss (or error) at the output and then propagating it back into the network.

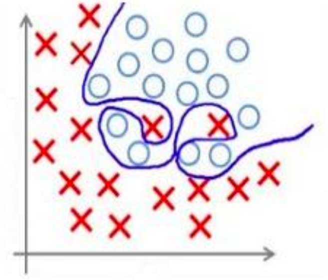
- The weights are updated to minimize the error resulting from each neuron.



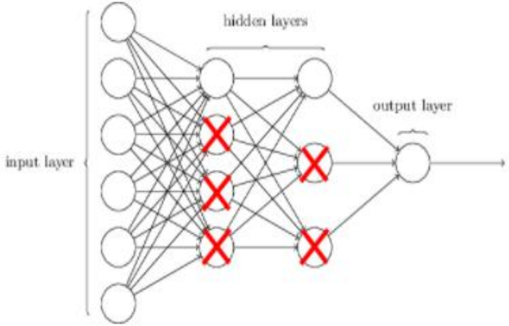
# Regularization: Dropout

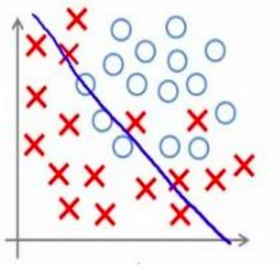
info@knowlexon.com



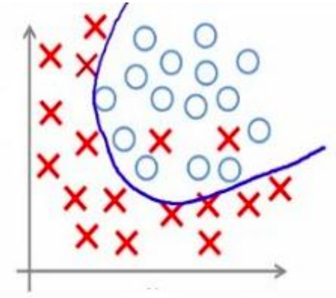
Over-fitting



Under-fitting

- At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections
- We need to choose the dropout parameter such that we get the appropriate fitting



Appropriate-fitting

info@knowlexon.com

# Building a Deep Learning Application

- Make sure that you actually need Deep Learning - don’t just jump into it.
- You need to have DATA (Labelled) - curated.
- GIGO is the rule on using data.
- Choose the right algorithm/network/framework for your Use case.
- Find out if you can use **Transfer Learning** - if yes go ahead try it out.
- If you don’t have data - Prepare a data capture strategy (buy it. Collect it etc) - and use API based service of known providers if you can use it.
- Do not reinvent the wheel of trying to build a network from scratch.
- Play with hyper-parameters of known models/framework etc.
- Some cases an ensemble is good enough for the problem.
- Look at Kaggle, Arxiv and other data science competition platforms

info@knowlexon.com

# Different DL Frameworks

Chainer

Caffe2

mxnet

TensorFlow

PYTORCH

Microsoft CNTK

theano

K

# Underneath

info@knowlexon.com

# Basic Components: DL Frameworks

- Tensors
- Operations
- Computational Graph: Combine multiple operations
- Auto-differentiation Module
- GPU support with FP operation (cuBLAS etc)

info@knowlexon.com

# What is a Tensor?

- Tensors are multidimensional arrays
- This is the standard way of representing data in Tensorflow

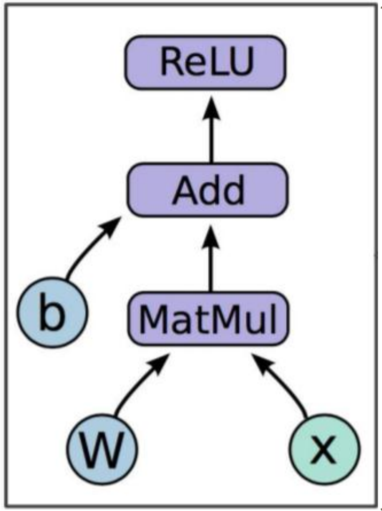

*Tensor of dimension[1]*  *Tensor of dimensions[2]*  *Tensor of dimensions[3]*

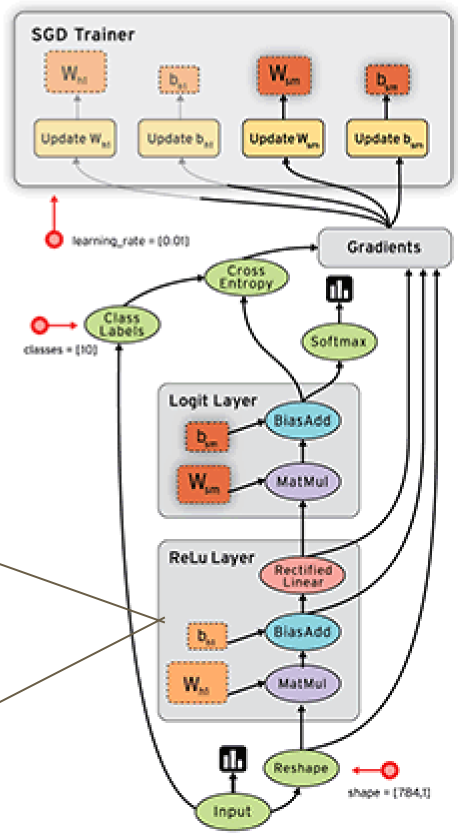
info@knowlexon.com

# TensorFlow, a DL Library

1) A deep learning library open sourced by google
2) Created for task with heavy numeric computation
3) Based on dataflow graph

$$h = ReLU(Wx + b)$$



info@knowlexon.com

# Tensorflow code

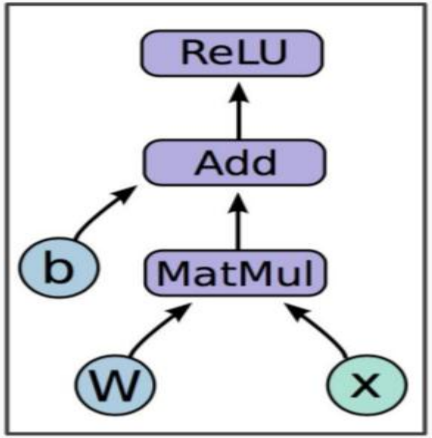
```
import tensorflow as tf

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

x = tf.placeholder(tf.float32, (100, 784))

h = tf.nn.relu(tf.matmul(x, W) + b)
```

$$h = ReLU(Wx + b)$$



info@knowlexon.com

# Tensorflow code: Getting the output

sess.run(fetches, feeds)

**Fetches:** List of graph nodes.

To fetch the the output of the operation execute the graph in run().

**Feeds:**
Feed data to the graph

```
import numpy as np
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, (100, 784))
h = tf.nn.relu(tf.matmul(x, W) + b)


sess = tf.Session()
sess.run(tf.initialize_all_variables())
sess.run(h, {x: np.random.random(100, 784)})
```

# So What we covered so far?

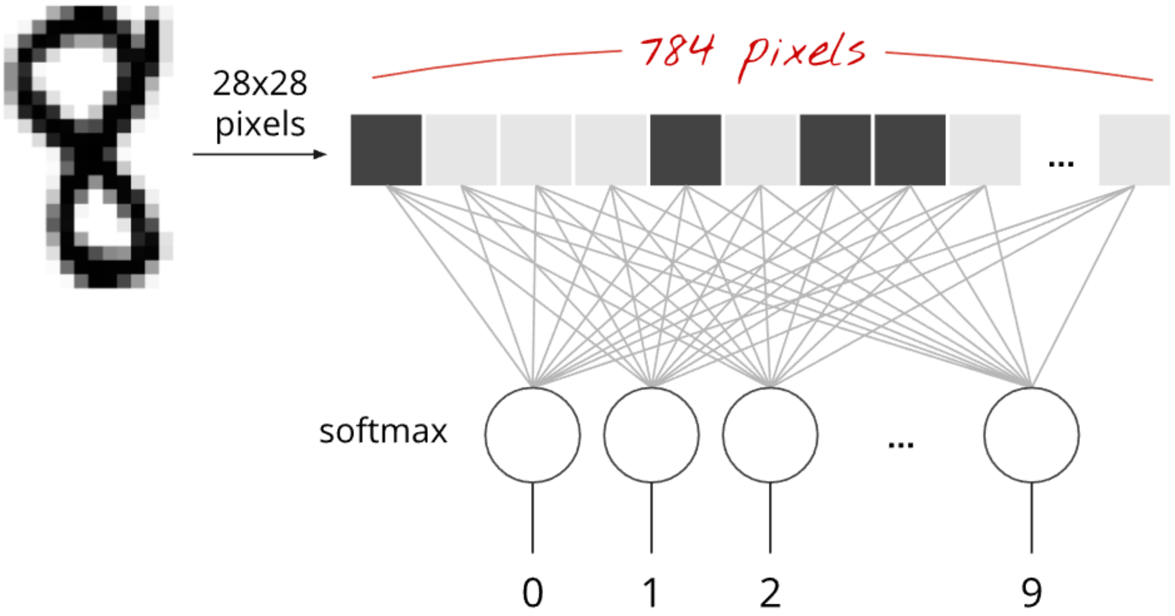
We first built a **graph** using **variables**, **placeholders** and **operations** on them

We then deployed the graph **onto a session**, which is the **execution environment**

Next we will see **how to train the model**

info@knowlexon.com

# Handwritten Digit classification MNIST



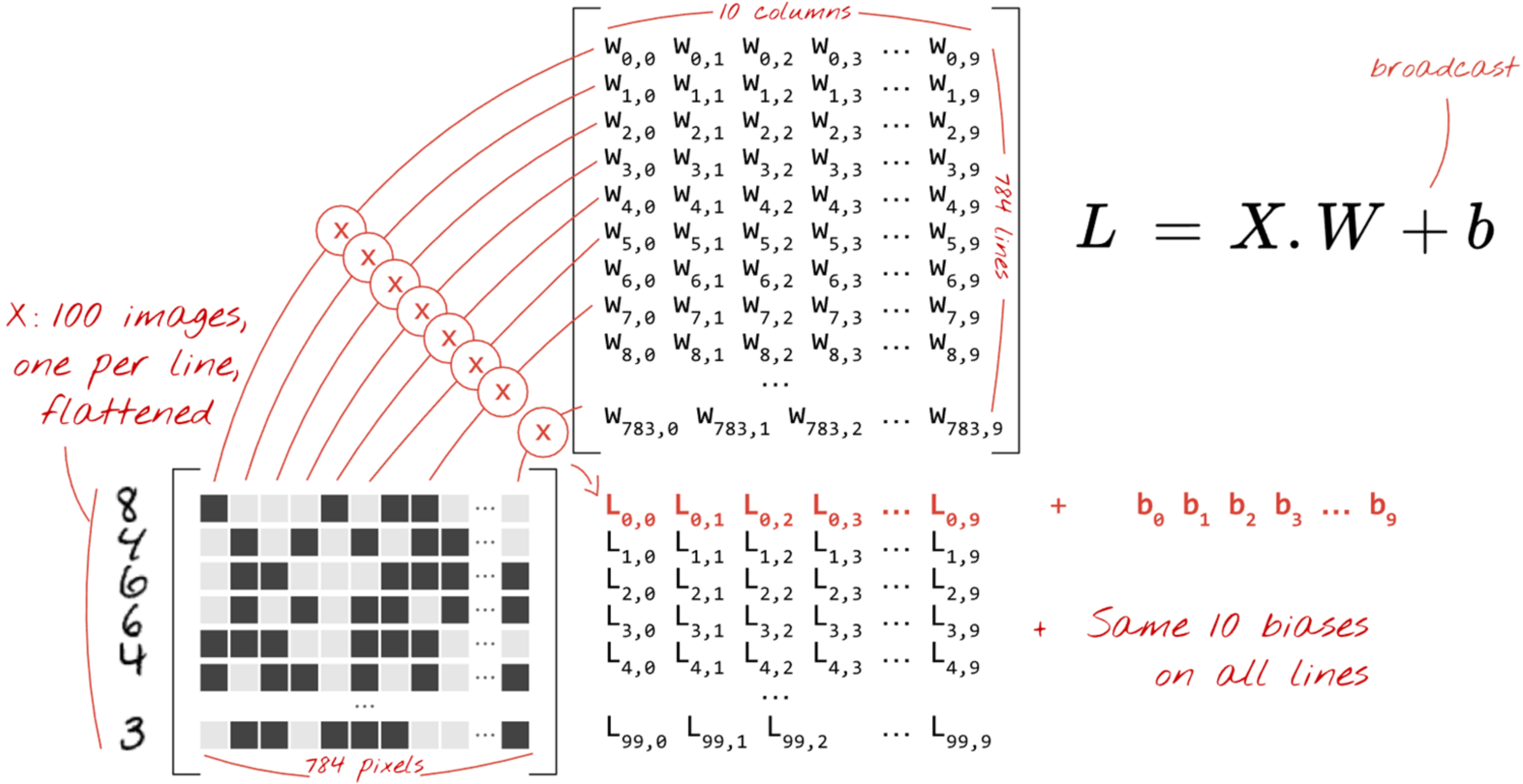

weighted sum of all pixels + bias

$$softmax(L_n) = \frac{e^{L_n}}{||e^{L}||}$$

neuron outputs

# Matrix Notation

info@knowlexon.com




$$L = X.W + b$$

info@knowlexon.com

# Error Metric

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

actual probabilities, "one-hot" encoded

Cross entropy: $$-\sum Y_i' . \log(Y_i)$$

this is a "6"

computed probabilities

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.1 | 0.2 | 0.1 | 0.3 | 0.2 | 0.1 | 0.9 | 0.2 | 0.1 | 0.1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

info@knowlexon.com

# Training

```
optimizer = tf.train.GradientDescentOptimizer(0.003)
train_step = optimizer.minimize(cross_entropy)
```

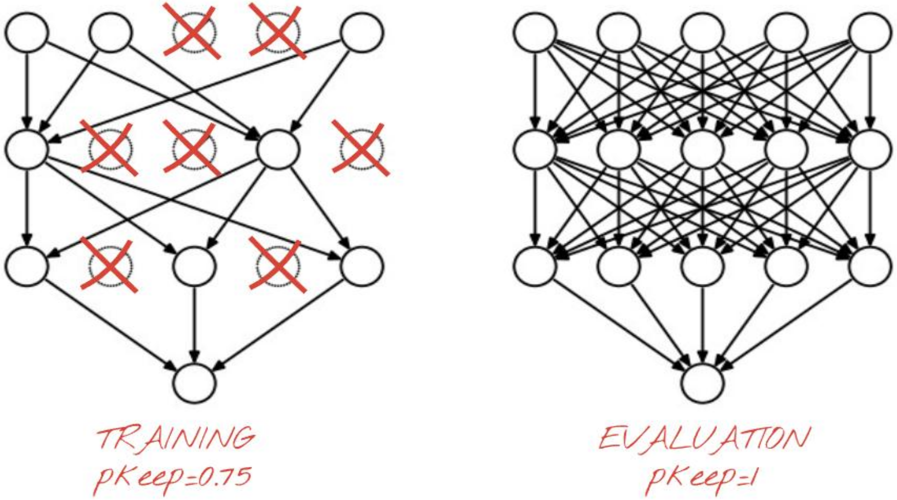
info@knowlexon.com

# Dropout

A regularization technique works great in Deep Neural network

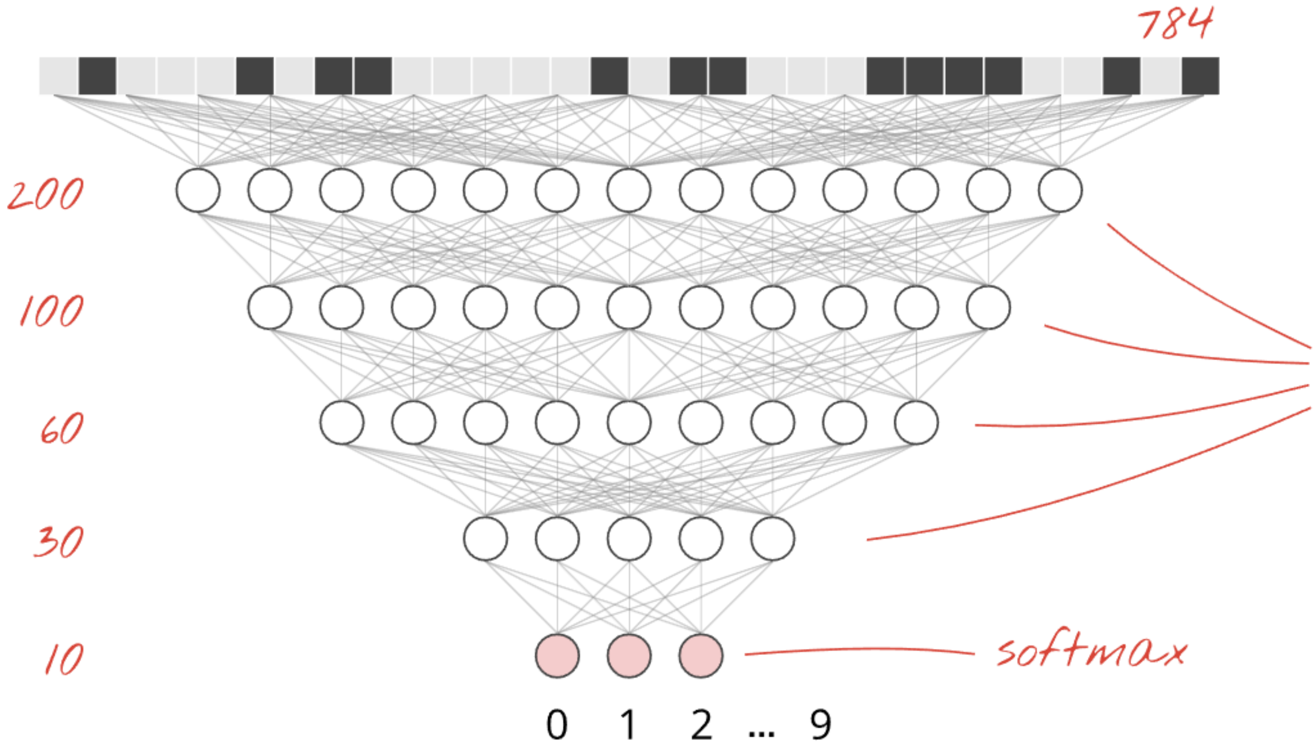
Randomly Drops certain % of nodes in training to reduce overfitting

An efficient way of doing model averaging as we randomly drop nodes for each training example



info@knowlexon.com

# DEMO




**RELU** and
**DROPOUT**

info@knowlexon.com

# Convolutional Neural Network (CNN)

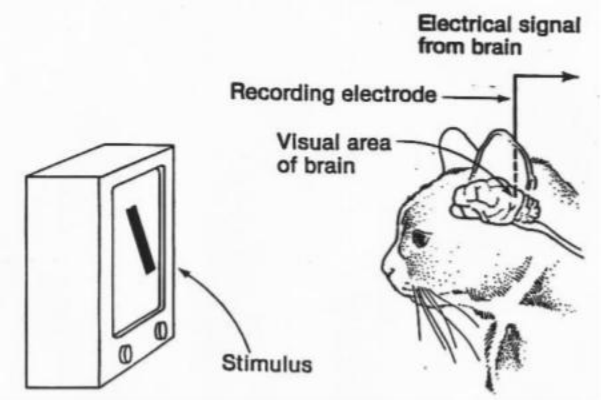
Problem With Fully connected network on images:

Losing shape information when we are flattening the image into a single array

Visual cortex system of brain doesn't work like fully connected network.

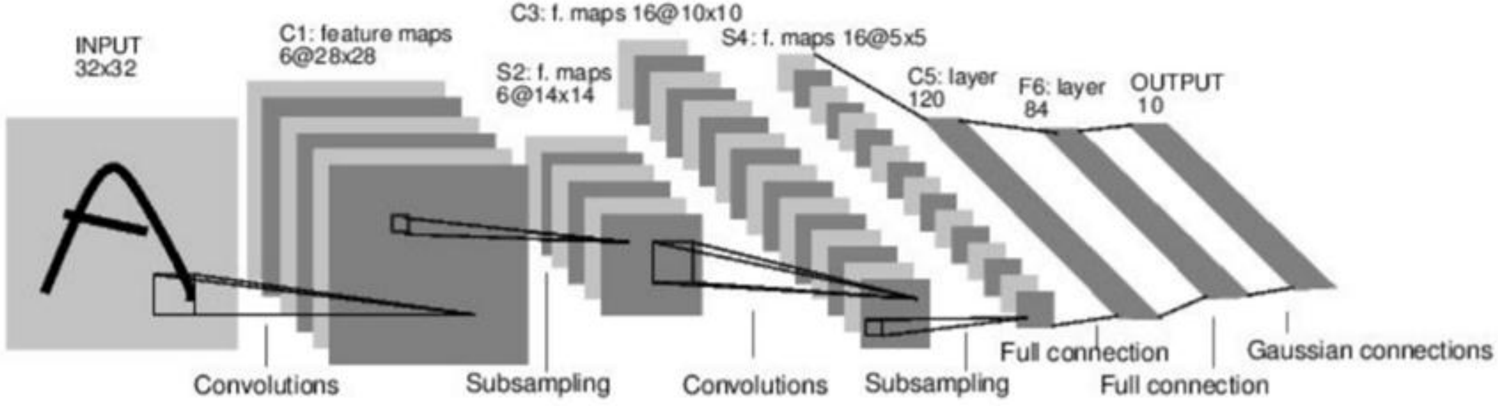
**Huble and Wiesel** experiment on cat reveals

1) **Local connection**
2) **Hierarchical layer**
3) **Spatial Invariance : Any size, rotation, shift**



info@knowlexon.com

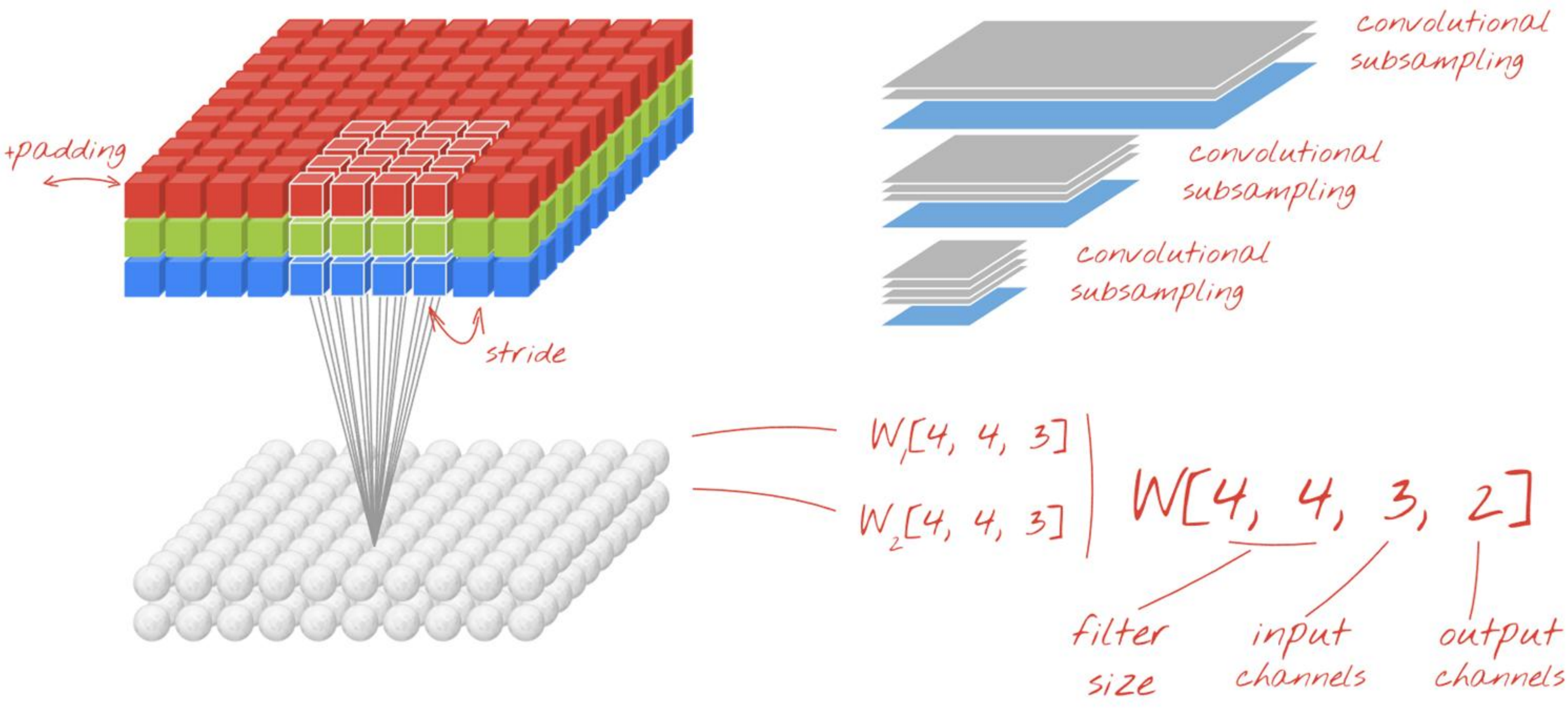
# CNN to rescue




*[LeNet-5, LeCun 1998 ]*

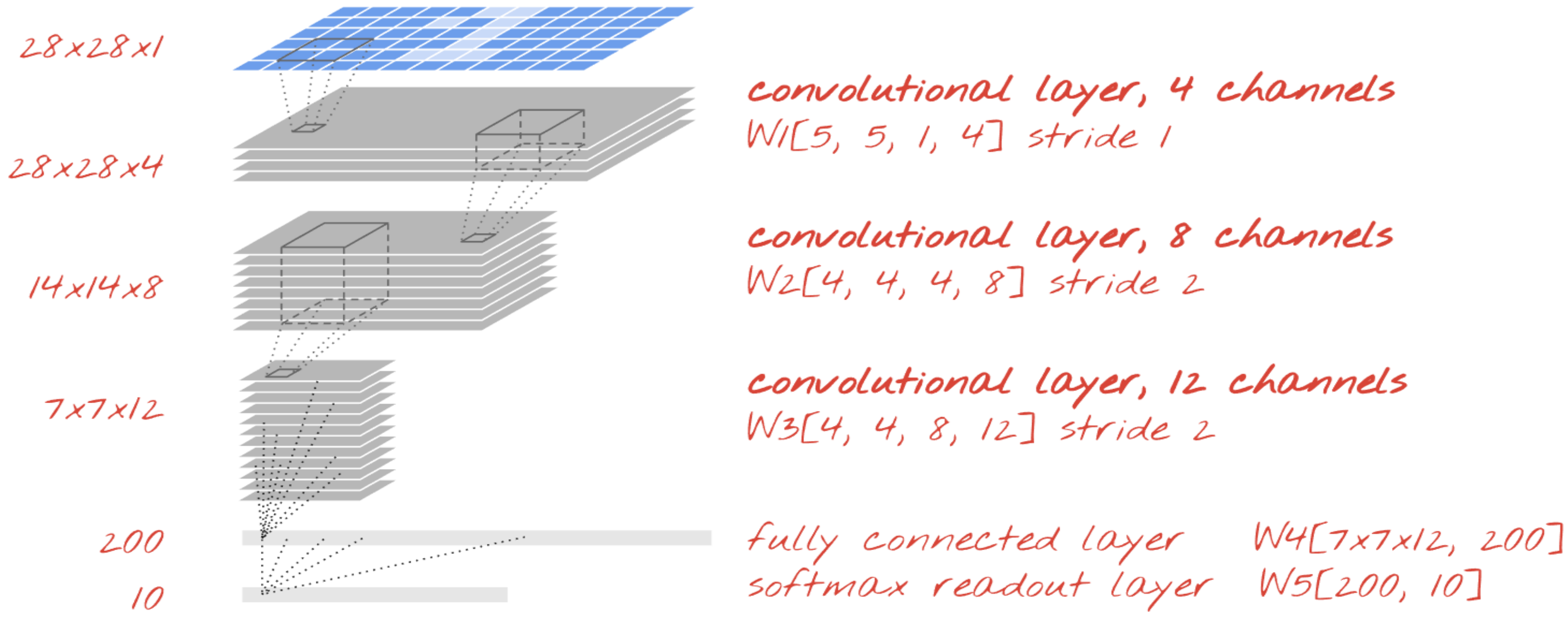
info@knowlexon.com

# Convolution layer



# Conv Net on MNIST

info@knowlexon.com



info@knowlexon.com

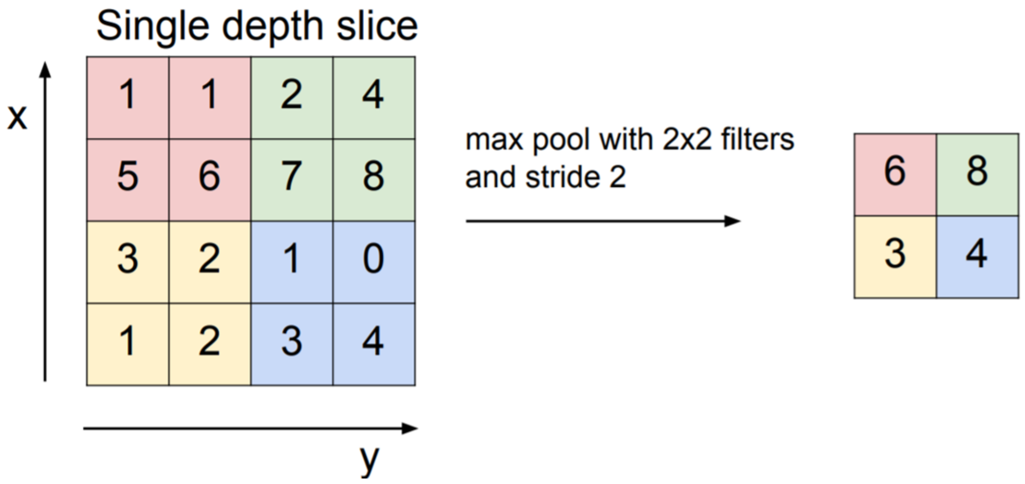
# Pooling

The idea of pooling in convolutional neural networks is to do two things:

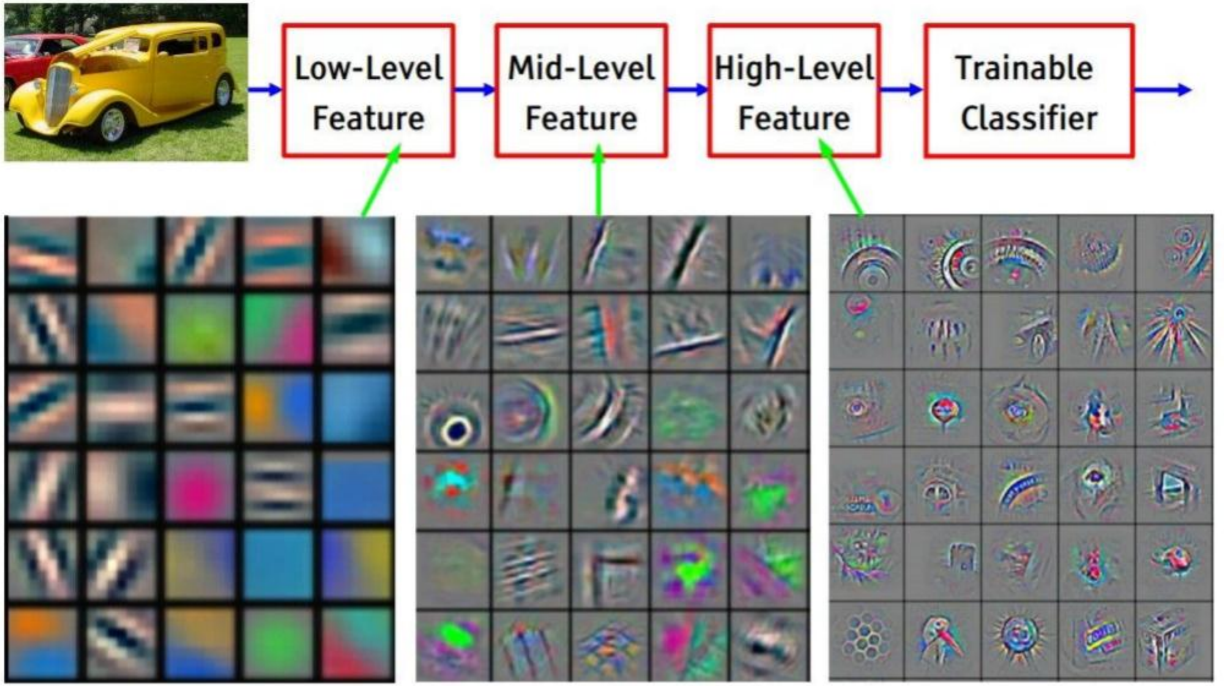
- Reduce the number of parameters in your network (pooling is also called “down-sampling” for this reason)
- To make feature detection more robust by making it more impervious to scale and orientation changes

Max Pooling

Average Pooling



# Feature extraction




Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

# Pytorch vs Tensorflow

- Tensorflow is widely adopted but pytorch picking up
- Dynamic vs static graph
- Tensorboard is better than pytorch visualization
- Plain tensorflow looks pretty much like a library
- Abstraction is better in pytorch, even data parallelism
- Tf.contrib, keras to rescue

info@knowlexon.com

# PYTORCH Framework Pieces

1. torch: a general purpose array library similar to Numpy that can do computations on GPU when the **tensor type is cast to (torch.cuda.TensorFloat)**
2. torch.autograd: a package for building a computational graph and automatically obtaining gradients
3. torch.nn: a neural net library with common layers and cost functions
4. torch.optim: an optimization package with common optimization algorithms like SGD,Adam, etc

# DEMO

# Acknowledge


- Materials have been taken from
- Martin Gorner Tensorflow Tutorial
- Stanford CS224d Tensorflow Tutorial
- Stanford CS231n course material
- Analytics Vidya article
- Many more PPTs available on the internet from Universities and Good Samaritans
- www.tensorflow.org

# For more details -

www.Facebook.com/knowlexon

https://www.linkedin.com/company/knowlexon

www.Knowlexon.com

info@knowlexon.com