MaGIC
MALAYSIAN GLOBAL INNOVATION AND CREATIVITY CENTRE

Introductory Session: Applied Deep Learning

18th November 2017

Dr. John See Su Yang (Multimedia University, MMU)
Dr. Poo Kuan Hoong (ASEAN Data Analytics Exchange, ADAX)

Agenda

- Introduction
  - Machine Learning
- Deep Learning
  - Artificial Neural Network
  - Convolution Neural Network (CNN)
  - Recurrent Neural Network (RNN)
  - Generative Adversarial Network (GAN)
  - Use Cases
  - Deep Learning Libraries/Tools
- Demo - TensorFlow & Keras
- Course Preview

About “Applied Deep Learning”

- This is an introductory preview towards a course aimed at
  - Graduate students,
  - Researchers,
  - IT professionals
  - Industrial engineers
- who already possess basic knowledge in Python programming and machine learning (and possibly, but not necessarily of deep learning) - who wish to learn more about this rapidly growing field of research and its application in the commercial world.

About us:

Poo Kuan Hoong, Ph.D http://www.linkedin.com/in/kuanhoong

COQNITICS
EMPOWERING INTELLIGENCE

- Founder

ADAX

- Senior Data Scientist

nielsen

- Senior Manager Data Science

MDEC
Driving Transformation

- Coursera Facilitator
- Consultant
- Funding mentor

MMU
MULTIMEDIA UNIVERSITY

- Senior Lecturer
- Chairperson Data Science Institute

- Founder R User Group & TensorFlow User Group
- Speaker/Trainer

About us:

- John See, Ph.D http://pesona.mmu.edu.my/~johnsee

MMU
MULTIMEDIA UNIVERSITY

- Senior Lecturer, Faculty of Computing and Informatics
- Head, Visual Processing Lab (http://viprlab.github.io)
- Theme Lead, Smart Cities & Digital Services
- Co-Organizer,
- TensorFlow and Deep Learning Malaysia User Group

Coqnitics Sdn Bhd

COQNITICS
EMPOWERING INTELLIGENCE

- Video surveillance with deep learning
- Fashion analytics with deep learning
- Deep Learning trainings and consultations

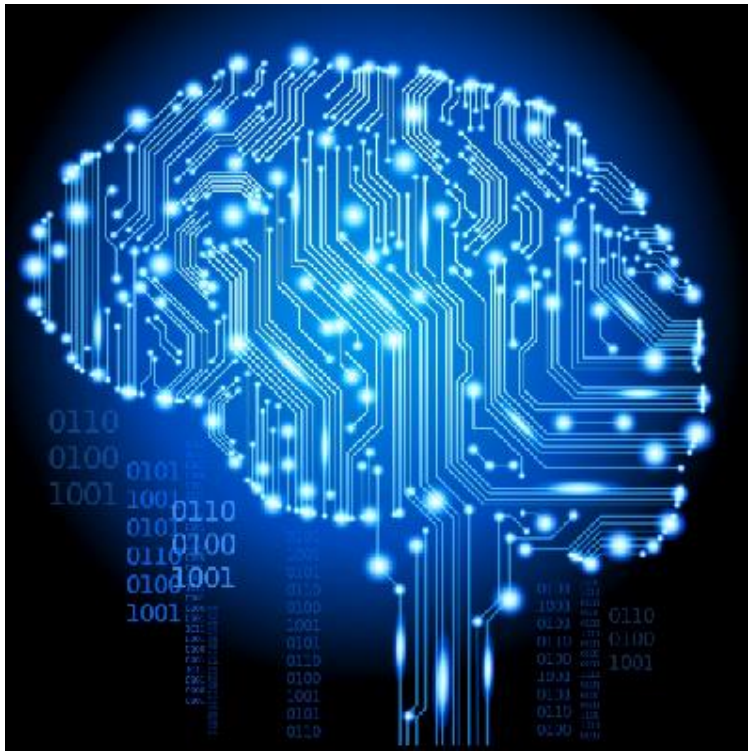
We are hiring…
www.coqnitics.com

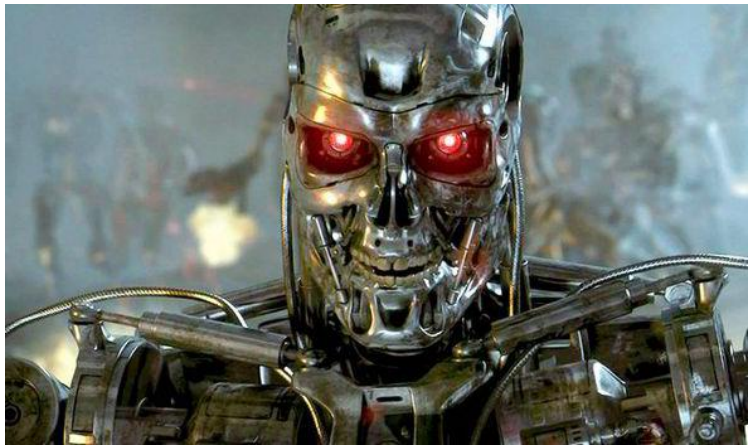
Where are we now..

Where are we now..

Introduction

- **Deep Learning**: The fastest growing sub-field in A.I.
  - Computers can now make sense of difficult tasks by crunching large amounts of data
- **See, Hear, Learn, React**
- Many industries and businesses have been impacted by DL, spurring new products and services that are data-driven
- **Machine Learning:** Machines that can learn!

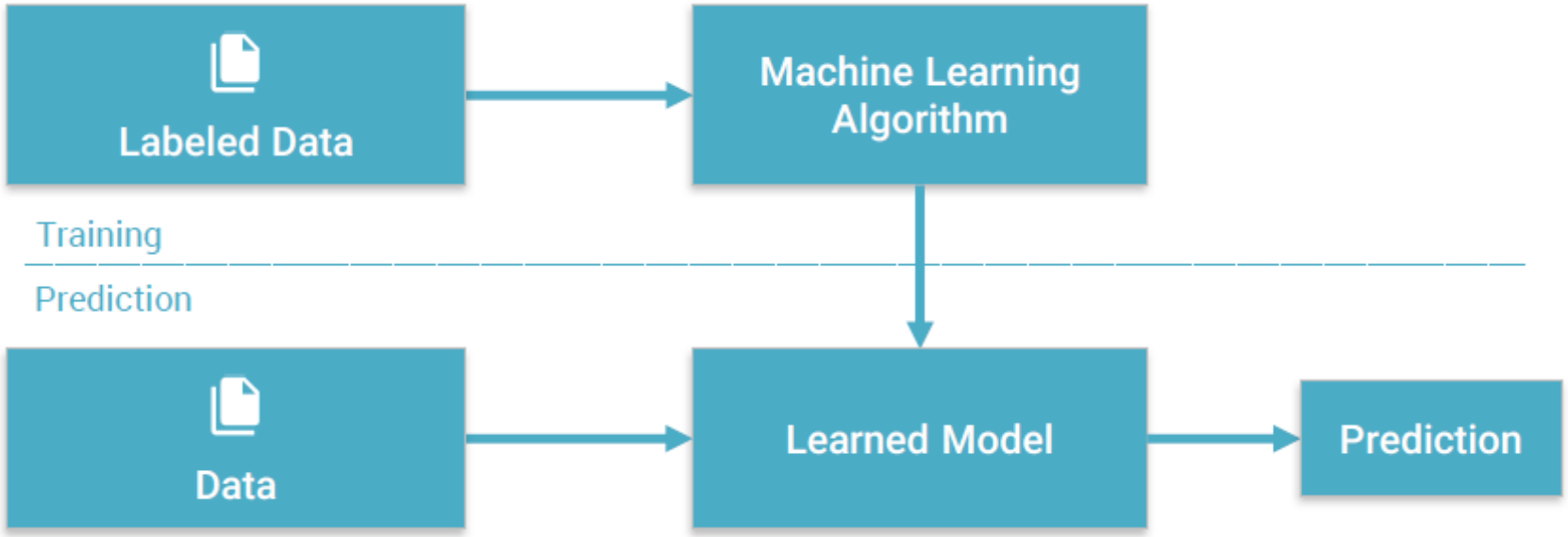




How do we learn?

Machine Learning

A type of Artificial Intelligence that provides computers with the ability to learn without being explicitly programmed



Machine Learning - Approaches

Supervised Learning

Learning from a labeled training set

Unsupervised Learning

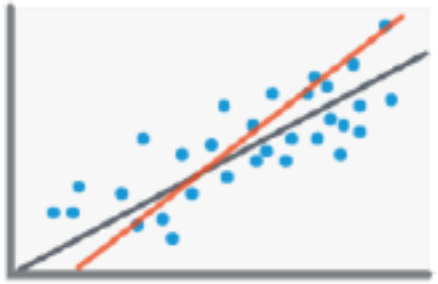
Discovering patterns in unlabeled data

Reinforcement Learning

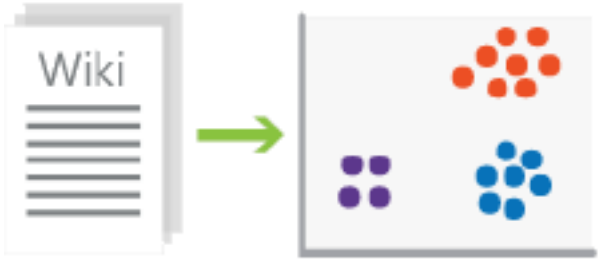
Learning based on feedback or reward

Machine Learning - Types of Problems

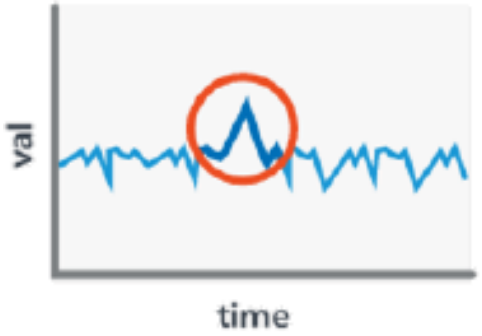
Classification



Regression



Clustering



Anomaly Detection

Neural Network

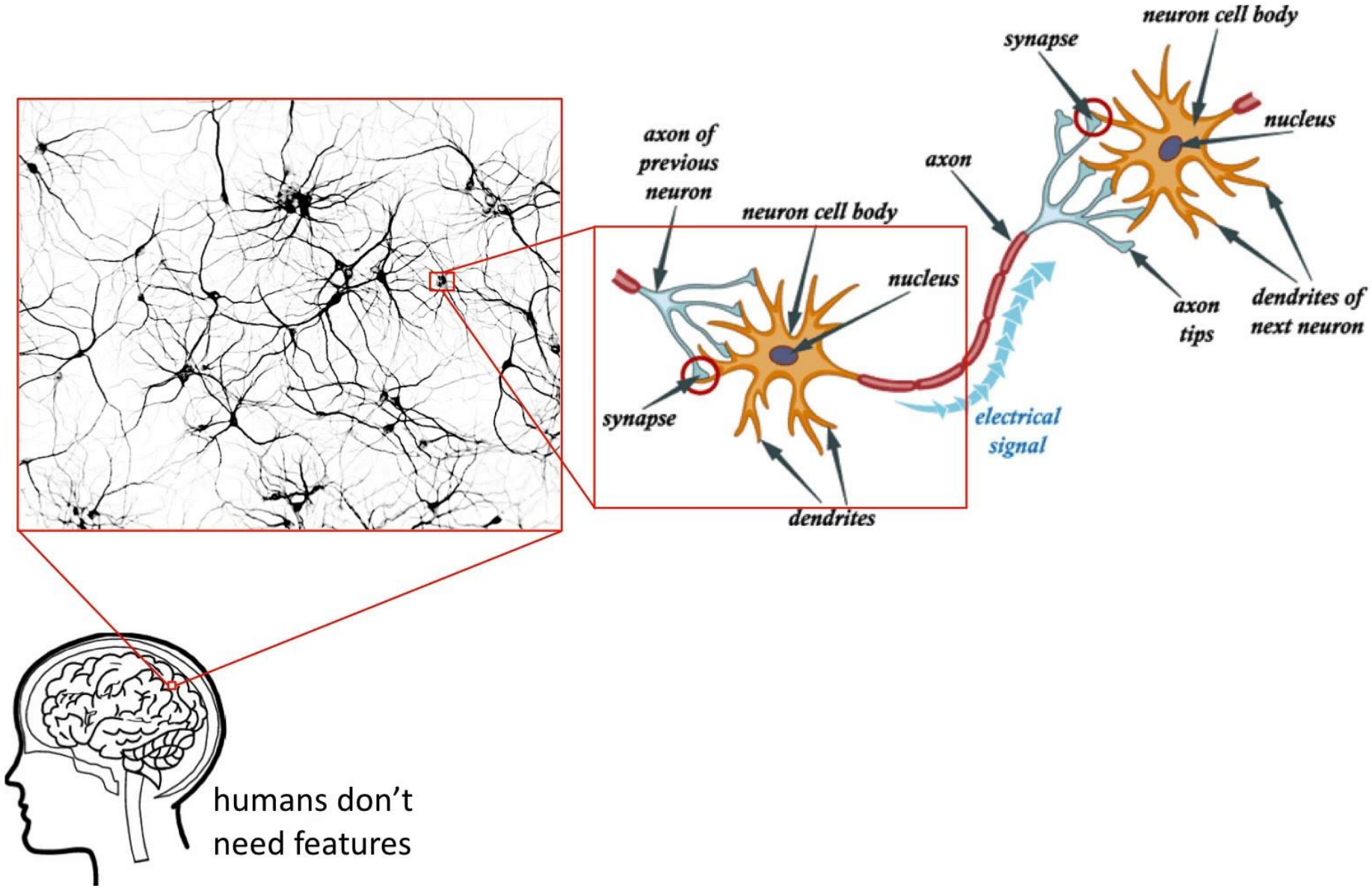
Biological Inspiration

- Animals are able to react adaptively to changes in their external and internal environment, and they use their nervous system to perform these behaviours.

- An appropriate model/simulation of the nervous system should be able to produce similar responses and behaviours in artificial systems.

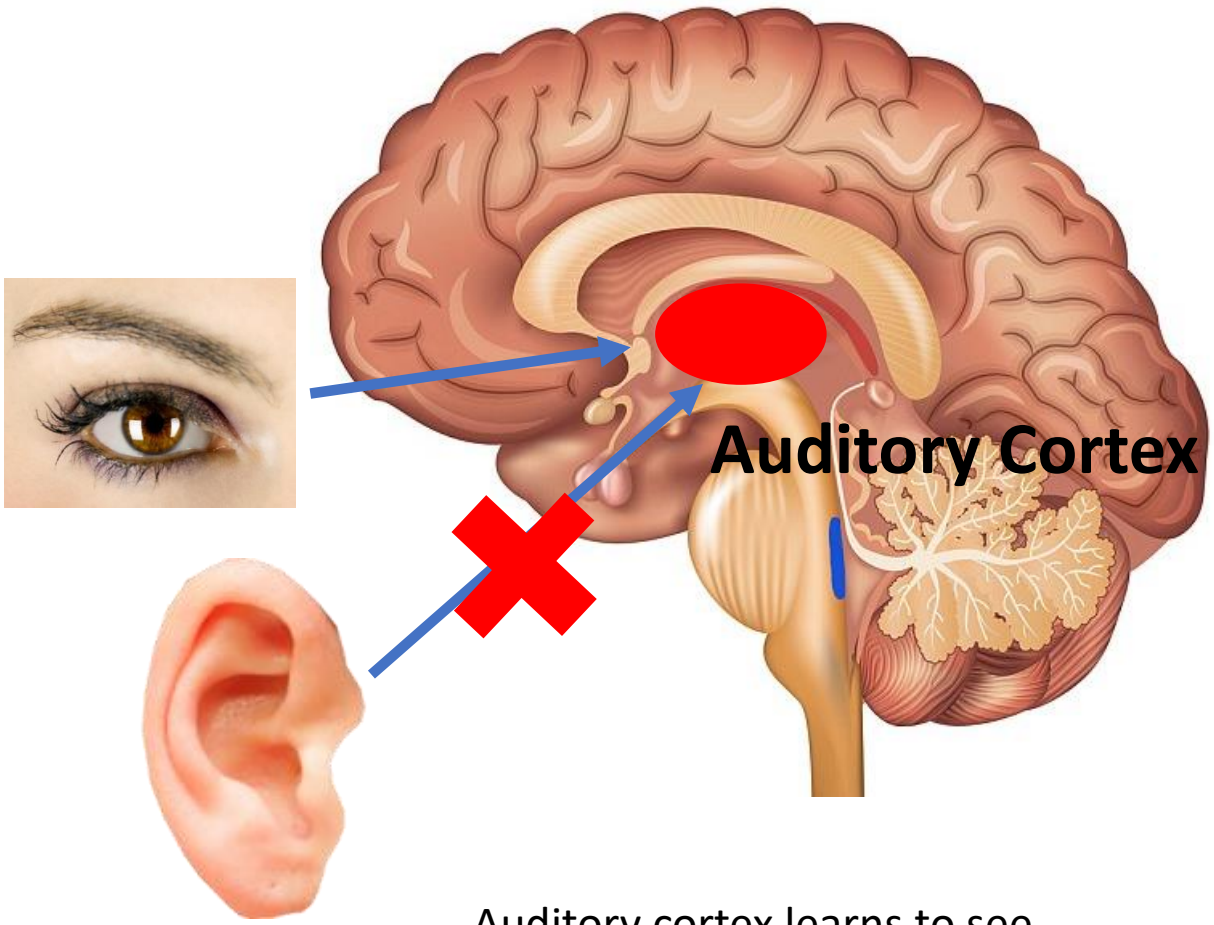
- The nervous system is build by relatively simple units, the neurons, so copying their behaviour and functionality should be the solution.

Neurons and the brain

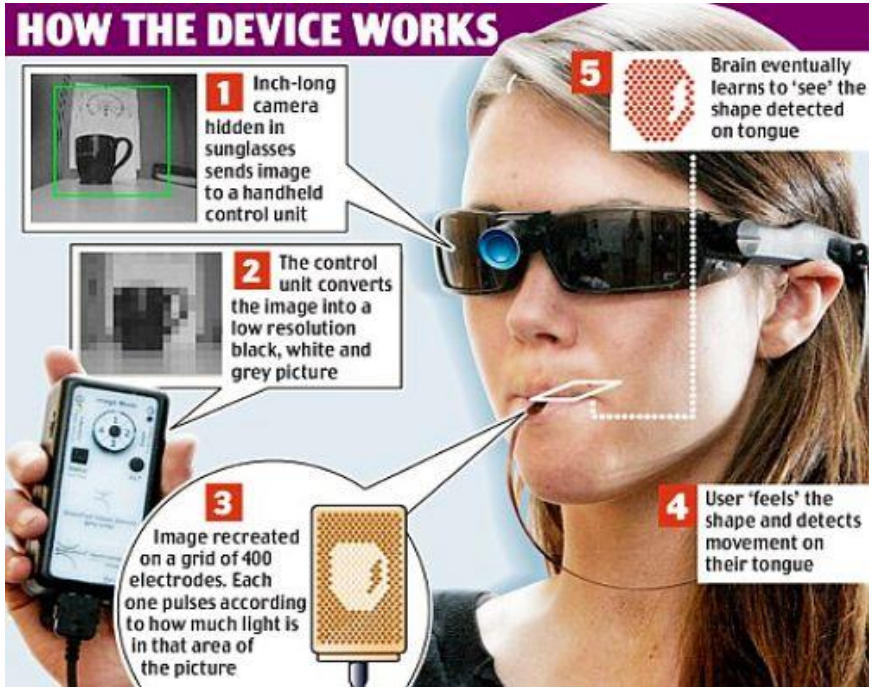


humans don't need features

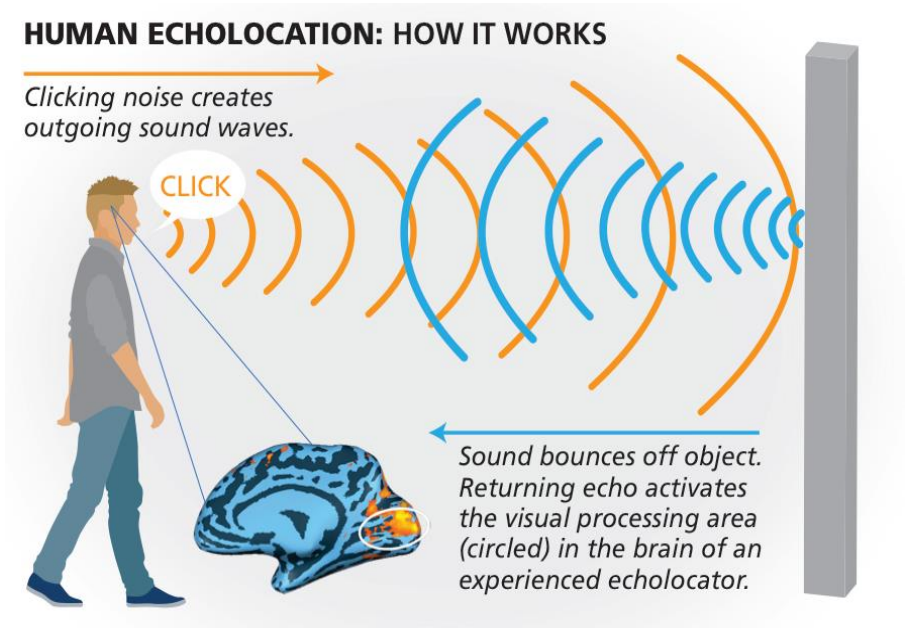
Human Brain



Auditory cortex learns to see. (Same rewiring process also works for touch/ somatosensory cortex.)

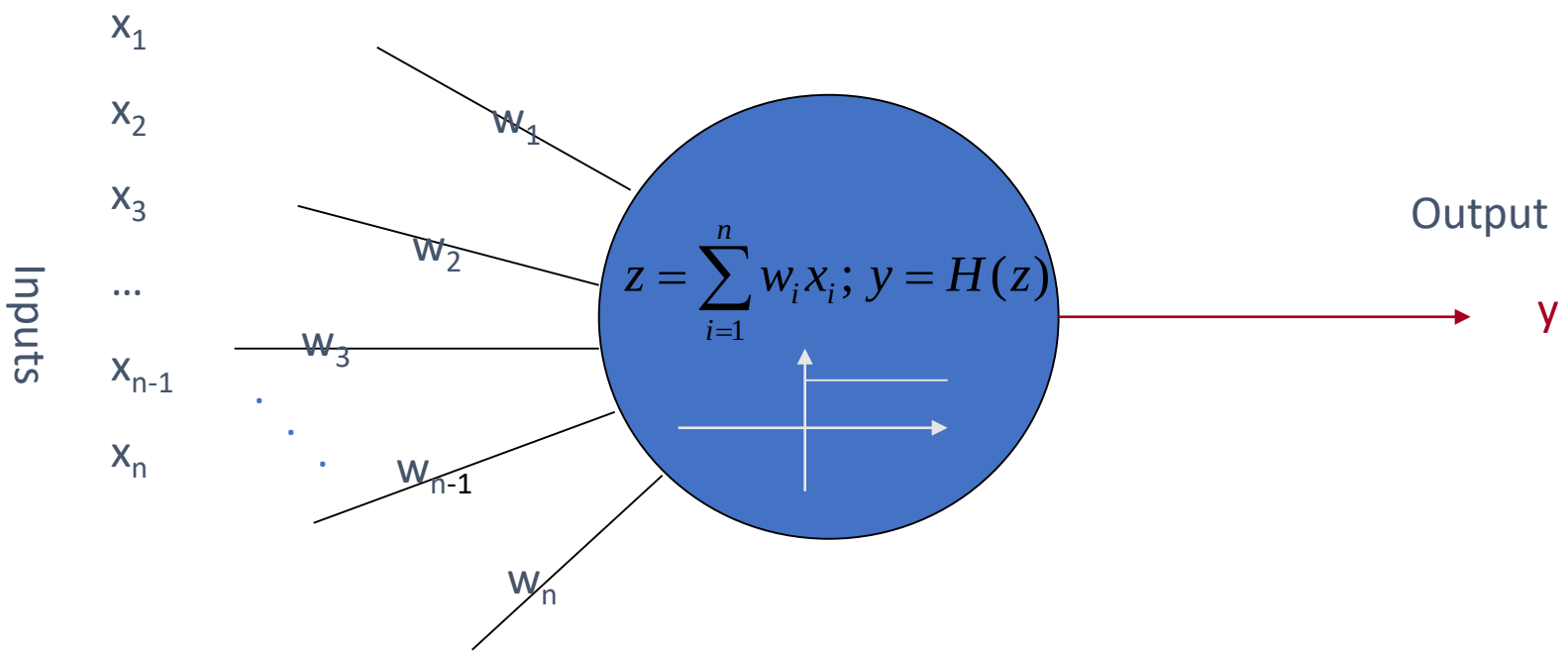


Seeing with tongue



Artificial Neurons

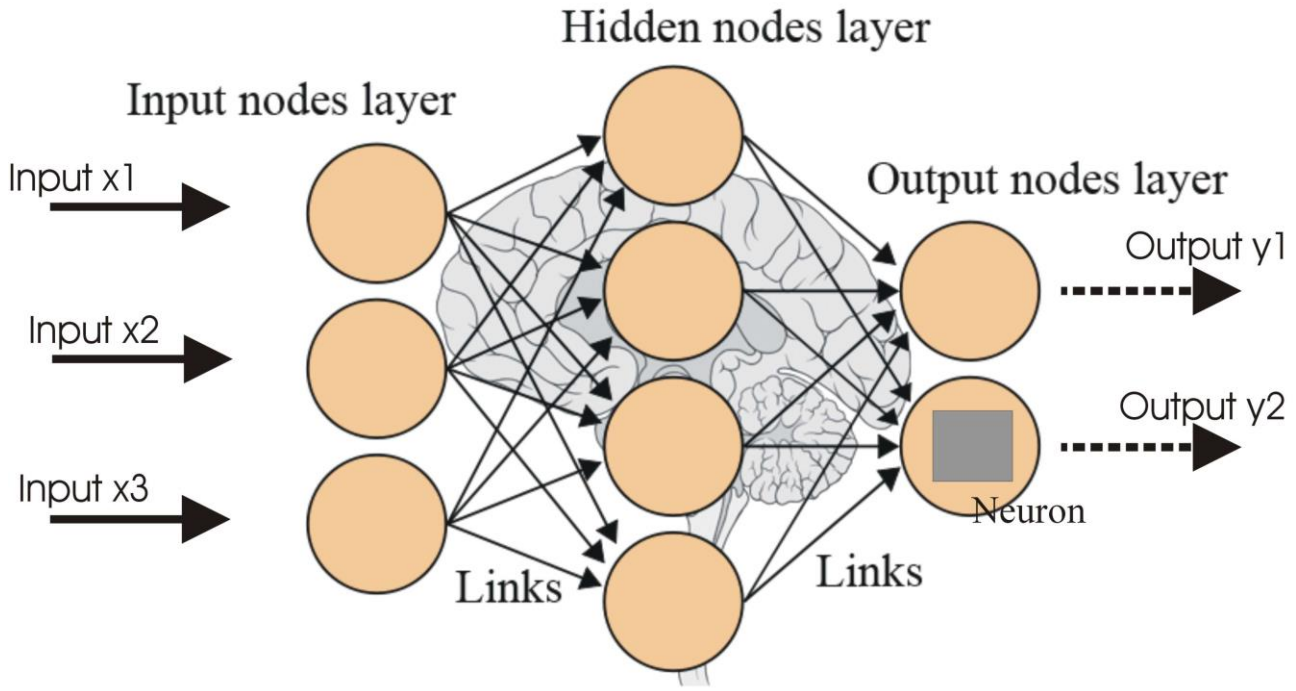
- Neurons work by processing information. They receive and provide information in form of spikes.



The McCullogh-Pitts model

Artificial Neural Networks

- An artificial neural network is composed of many artificial neurons that are linked together according to a specific network architecture. The objective of the neural network is to transform the inputs into meaningful outputs.



Artificial neural networks

- Tasks to be solved by artificial neural networks:
  - controlling the movements of a robot based on self-perception and other information (e.g., visual information);
  - deciding the category of potential food items (e.g., edible or non-edible) in an artificial world;
  - recognizing a visual object (e.g., a familiar face);
  - predicting where a moving object goes, when a robot wants to catch it.

Learning in biological neural networks

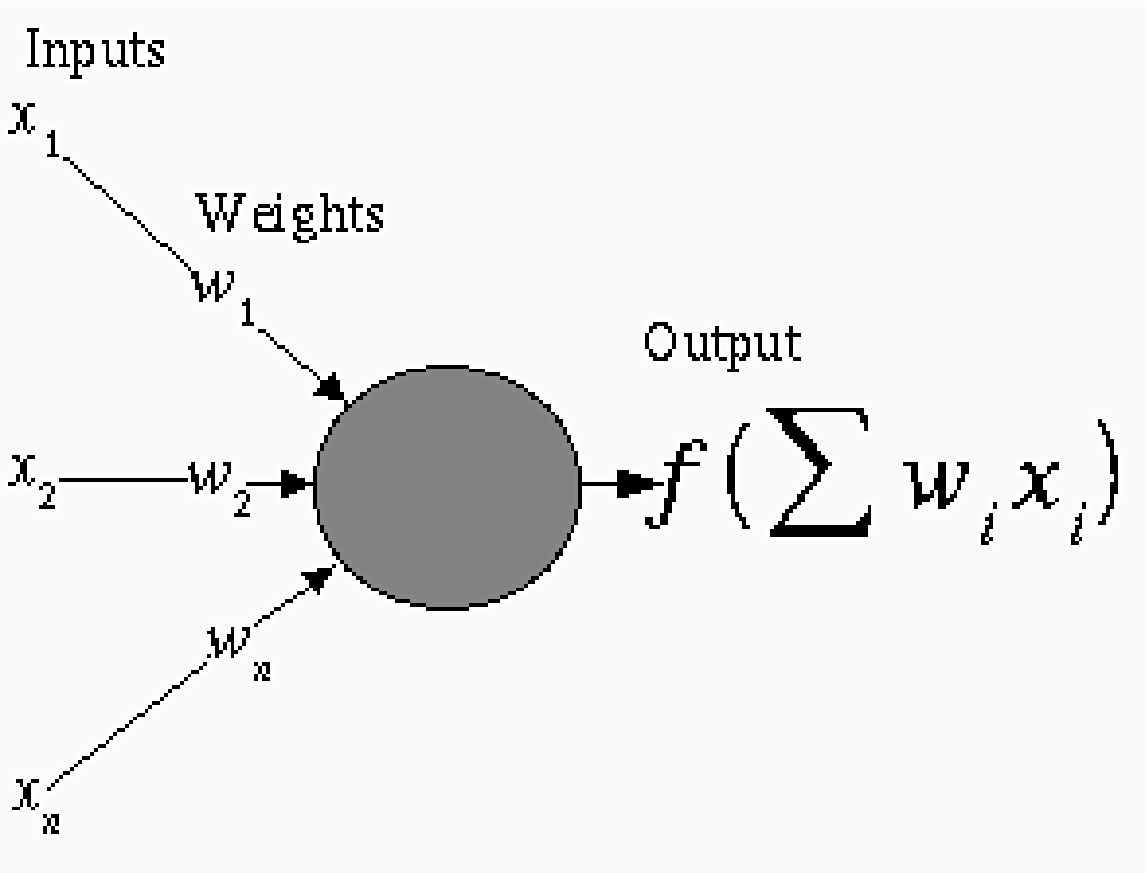
- The learning rules of Hebb:
  - synchronous activation increases the synaptic strength;
  - asynchronous activation decreases the synaptic strength.
- These rules fit with energy minimization principles.
- Maintaining synaptic strength needs energy, it should be maintained at those places where it is needed, and it shouldn’t be maintained at places where it’s not needed.

Backpropagation

- Solution of the complicated learning:
  - calculate first the changes for the synaptic weights of the output neuron;
  - calculate the changes backward starting from layer p-1, and propagate backward the local error terms.
- The method is still relatively complicated but it is much simpler than the original optimisation problem.

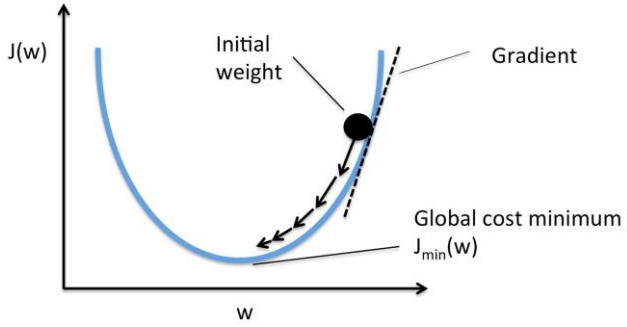
Backpropagation

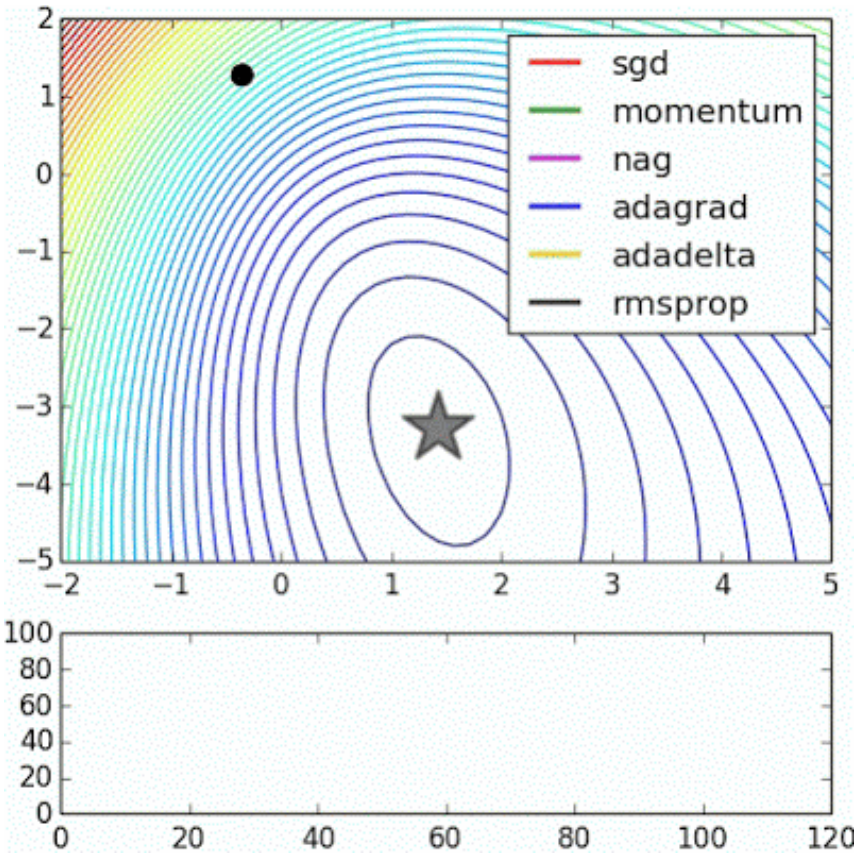
- **Inputs x_i** arrive through pre-synaptic connections
- Synaptic efficacy is modeled using real **weights w_i**
- The response of the neuron is a **nonlinear function *f*** of its weighted inputs



NN Basics: Optimizing the Cost Function

- Re-adjusting the weights and biases is an optimization problem
- A cost function is defined to measure overall error which needs to be minimized
- Gradients are used to update weight parameters





A Pseudo-Code Algorithm

- Randomly choose the initial weights
- While error is too large
  - For each training pattern (presented in random order)
    - Apply the inputs to the network
    - Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer
    - Calculate the error at the outputs
    - Use the output error to compute error signals for pre-output layers
    - Use the error signals to compute weight adjustments
    - Apply the weight adjustments
  - Periodically evaluate the network performance

Feedforward Network Training by Backpropagation: Process Summary

- Select an architecture
- Randomly initialize weights
- While error is too large
  - Select training pattern and feedforward to find actual network output
  - Calculate errors and backpropagate error signals
  - Adjust weights
- Evaluate performance using the test set

What's wrong with NN and back-propagation?

- It requires a lot of labelled training data
  - Almost all data is unlabelled
- The learning time does not scale well
  - It is very slow in networks with multiple hidden layers
- It can get stuck in poor local optimas
  - Works well most of the time, but suffers when complexi

Use unsupervised generative methods

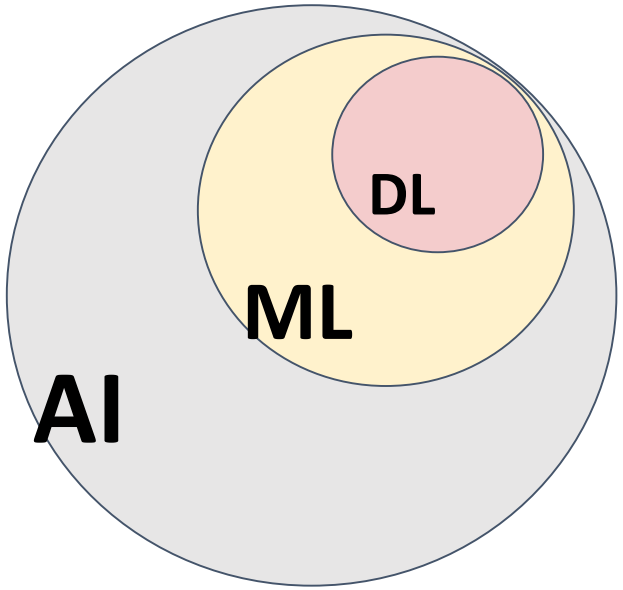
Bring out the GPUs!

Use more robust activation functions like ReLU, dropout layers, etc.

Deep Learning

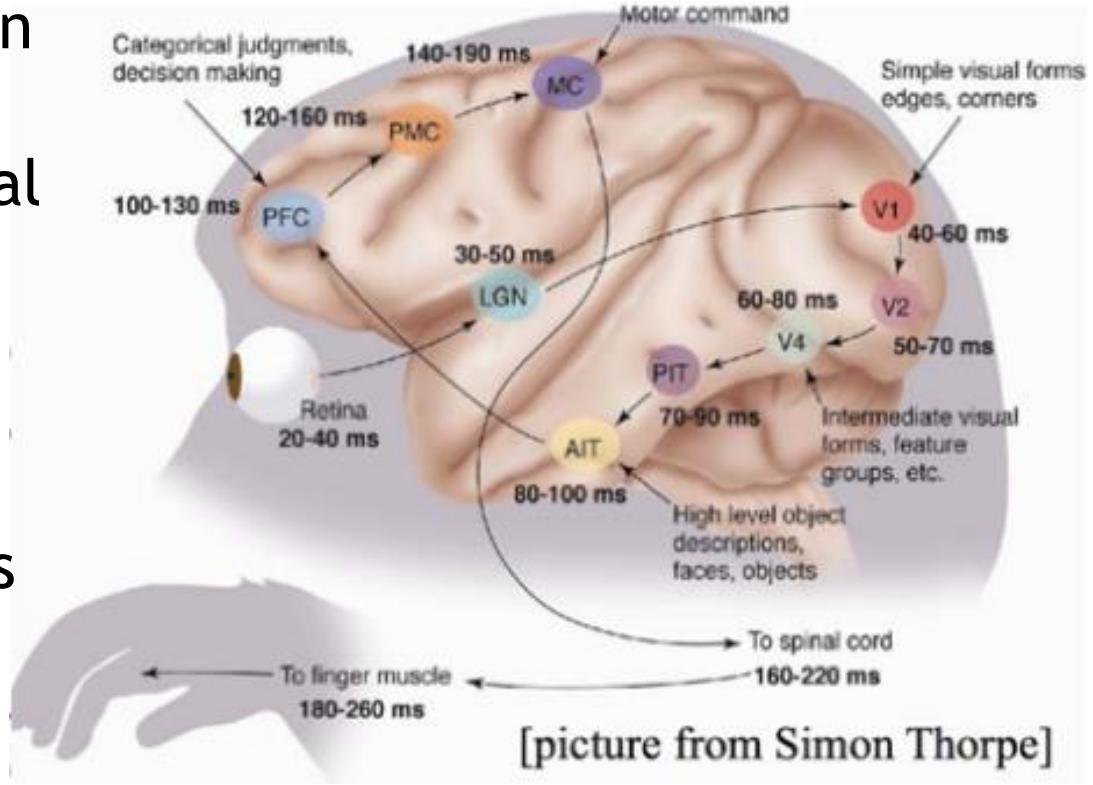
Deep Learning

- In the past 10 years, Artificial Intelligence (AI) particularly ML have shown tremendous progress.
- **Deep Learning:** part of Machine Learning (ML) field of learning representations of data
  - Exceptional effectiveness
  - Hierarchy of multiple “learning” layers that mimic the neurons in our brain
  - Thrives on big amounts of data



Why learn “deeply”?

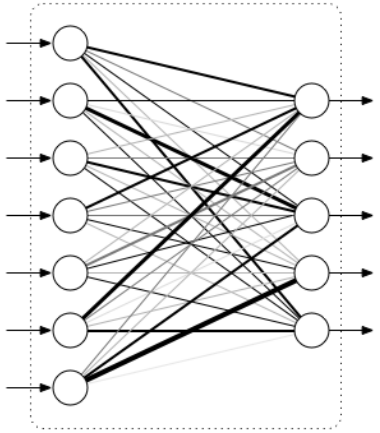
- Inspired by architectural depth of brain
- Neuroscience findings: Information becomes more complex down the visual pipeline
- No successful attempts were reported before 2006
    - SVM: Vapnik and co. developed Support Vector Machine which was pretty good, but shallow learning



2006 Breakthrough: What made Deep Learning possible?

- Explosion of data
- Faster and cheaper computing -- multi-core CPUs and GPUs
- Improvement of ML models -- more complex, more scalable

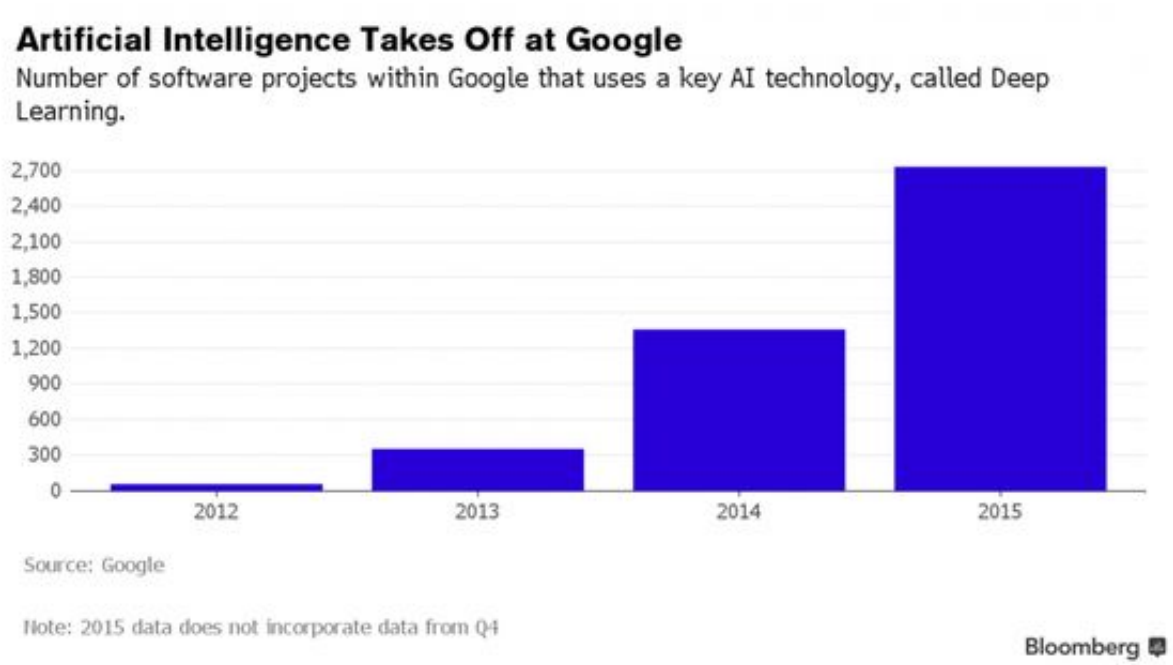
Stacked Restricted Boltzmann Machines (RBM) or **Deep Belief Nets** (DBN), Hinton, 2006
Stacked Autoencoders (AE), Bengio, 2007

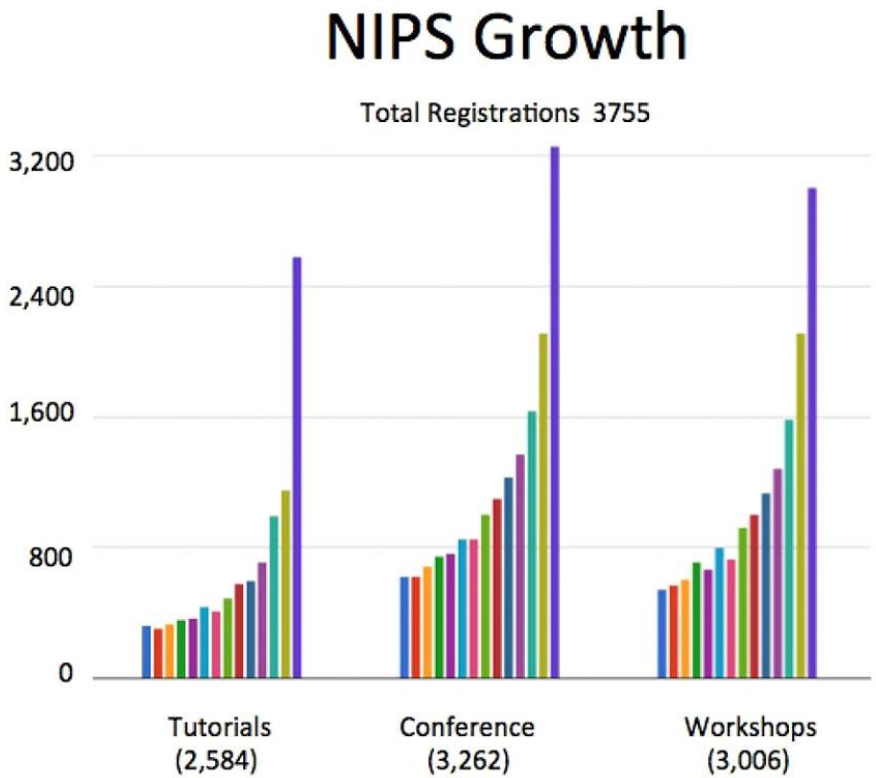


"I've worked all my life in Machine Learning, and I've never seen one algorithm knock over benchmarks like Deep Learning"

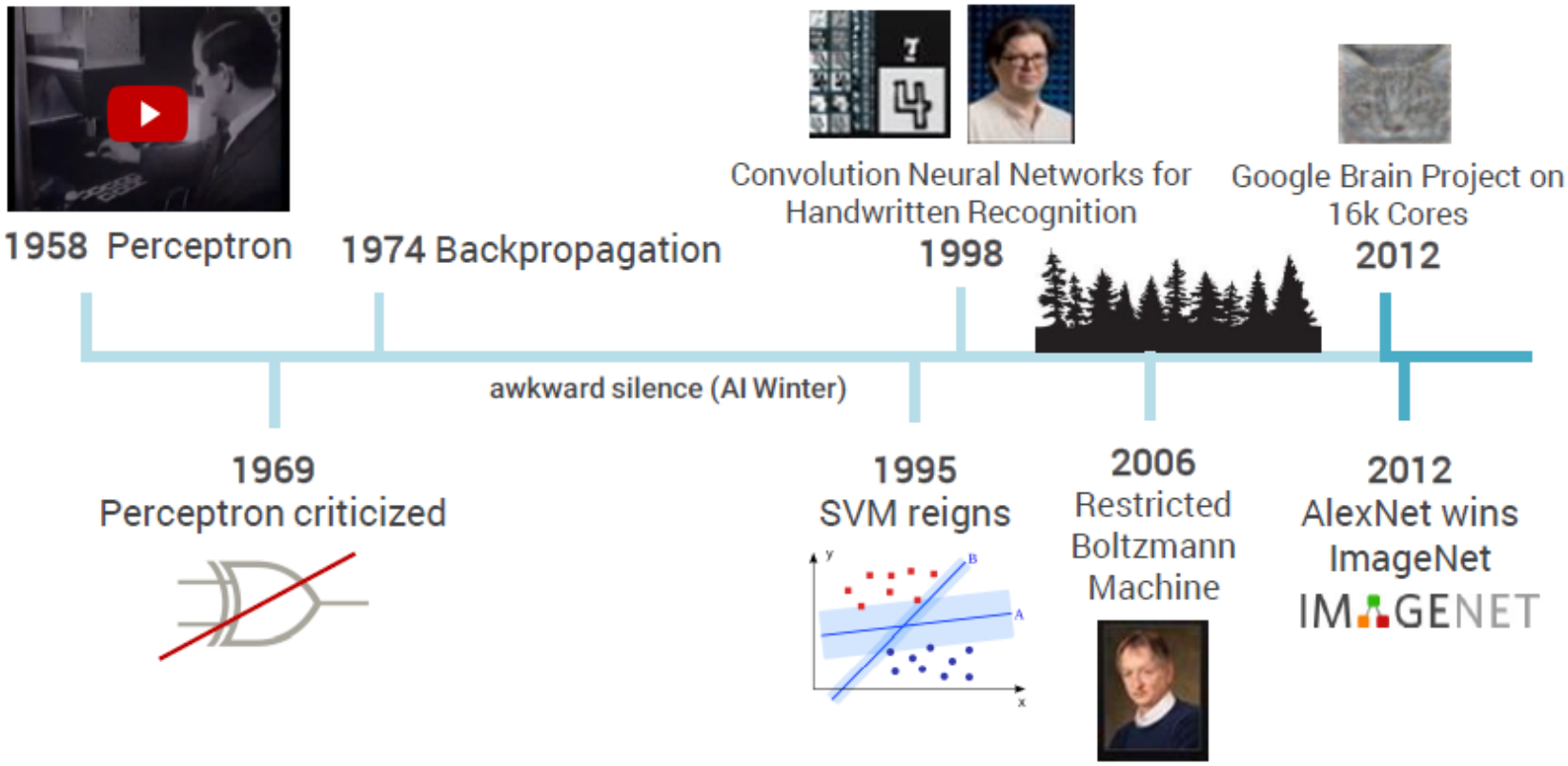
Andrew Ng
Stanford Univ, Baidu (prev.)
Co-Founder of Coursera

Deep Learning: Hype or Reality?

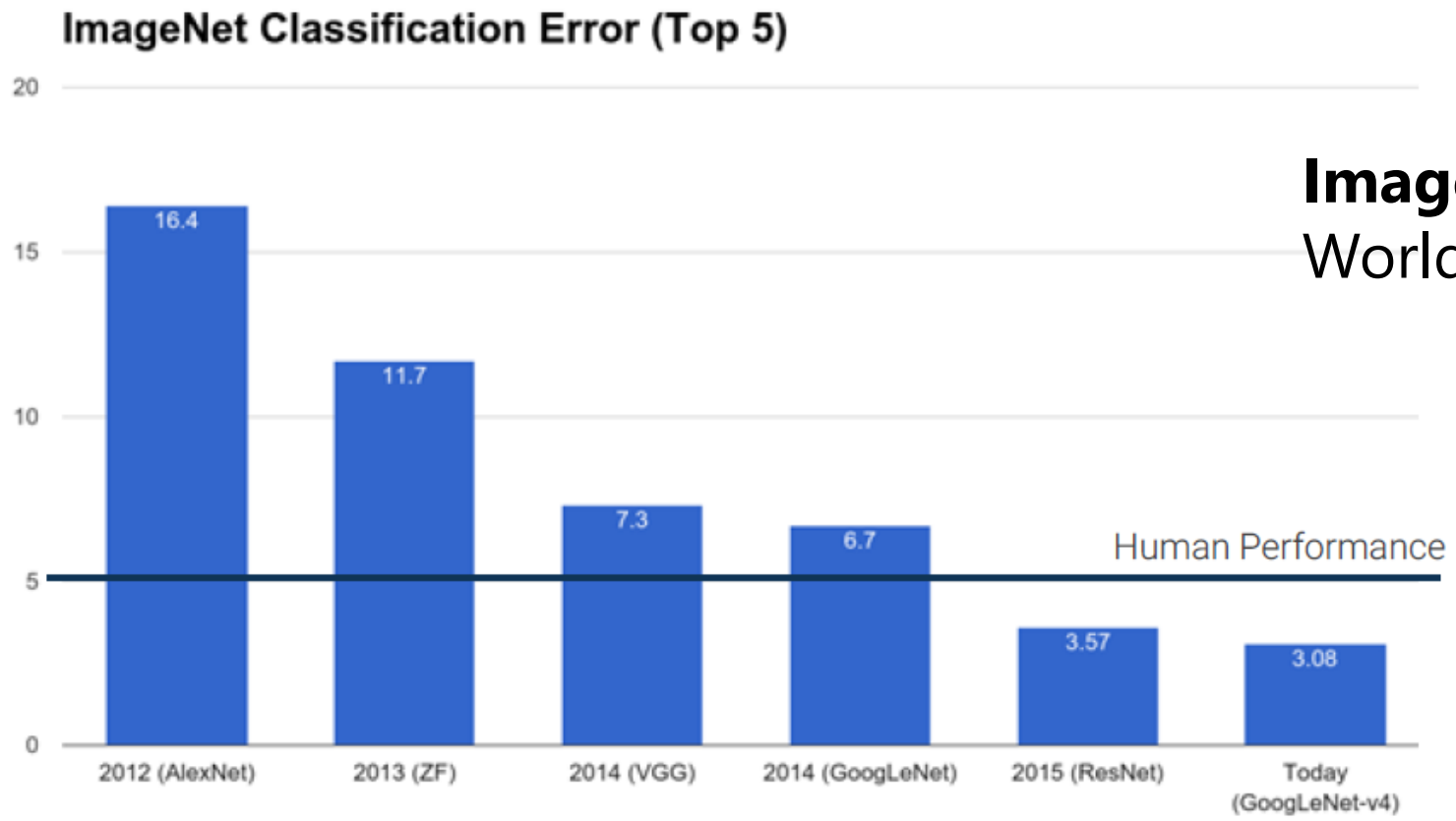




A very brief history...

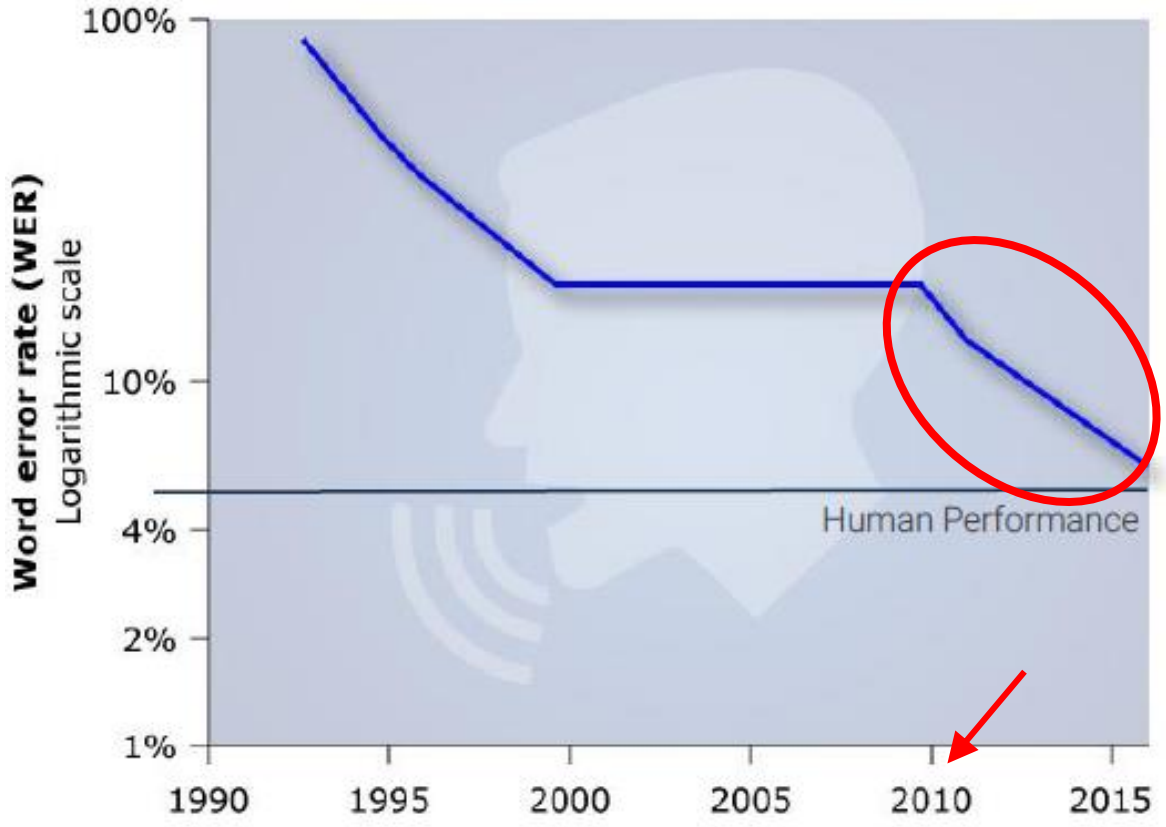


“One algorithm to rule them all”



ImageNet: The “computer vision World Cup”

“One algorithm to rule them all”

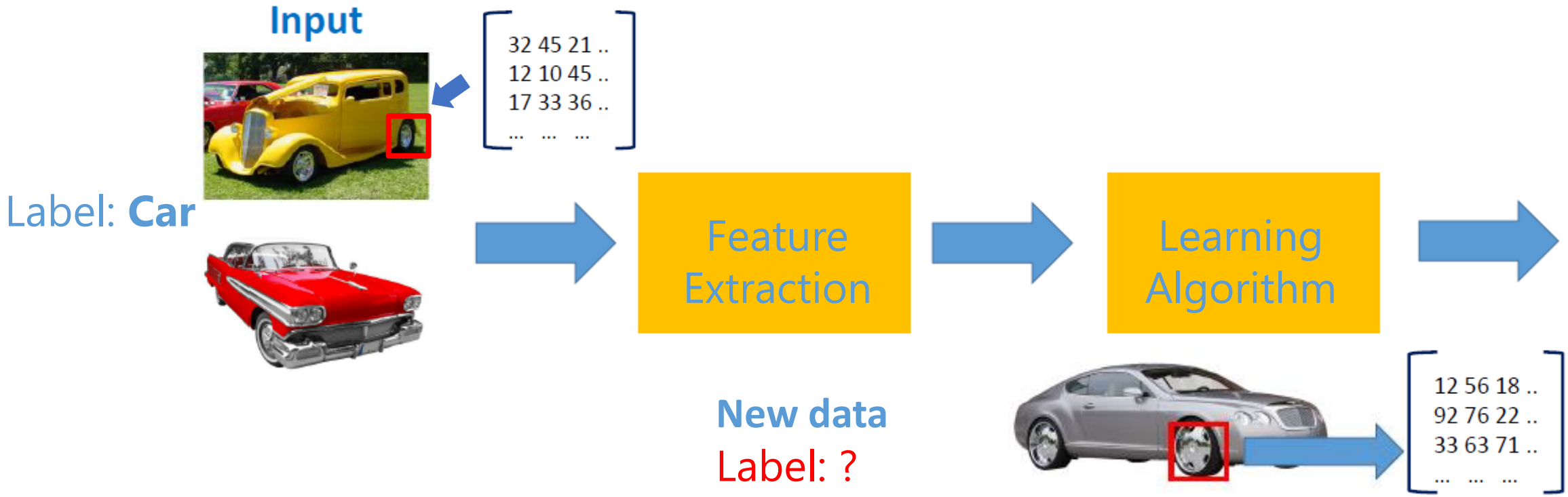


State-of-the-art Performance for Speech Recognition

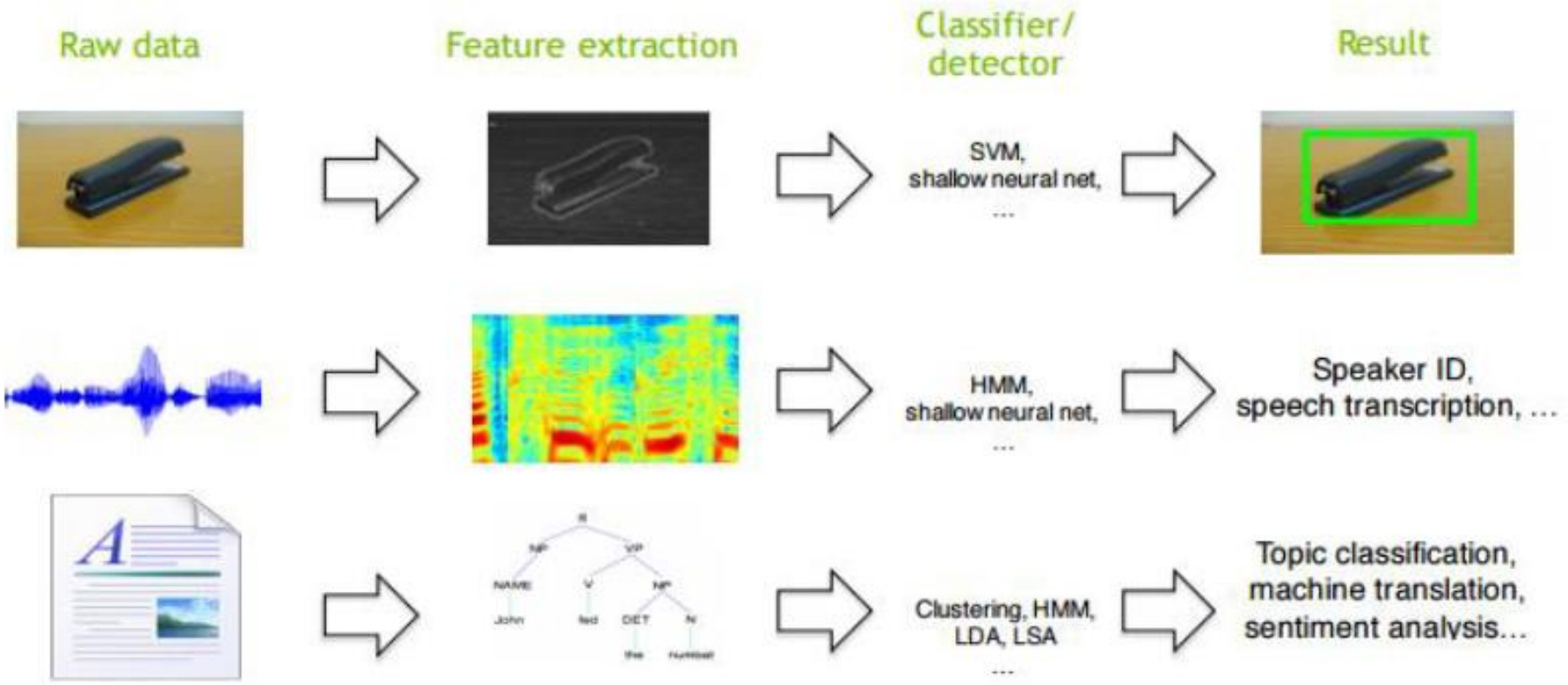
Thanks to DL!

Revisiting Traditional Machine Learning

- **"Feature Engineering":** Relates to extracting features by "hand-crafted" means. A simple classifier can be trained for recognition.

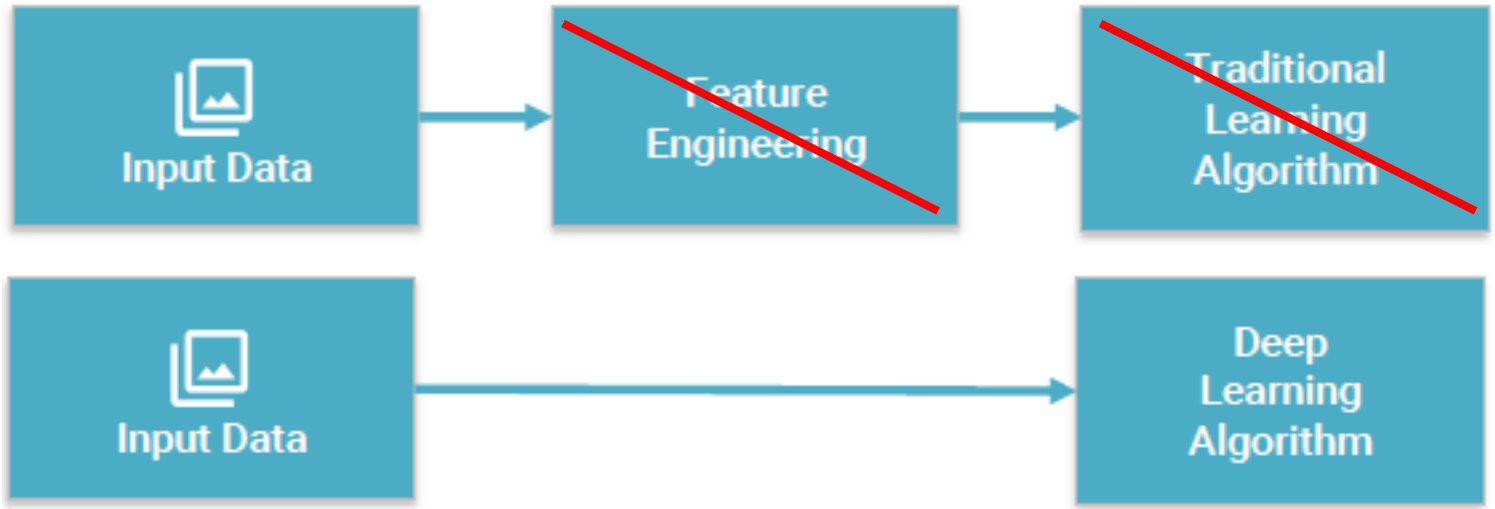


Revisiting Traditional Machine Learning

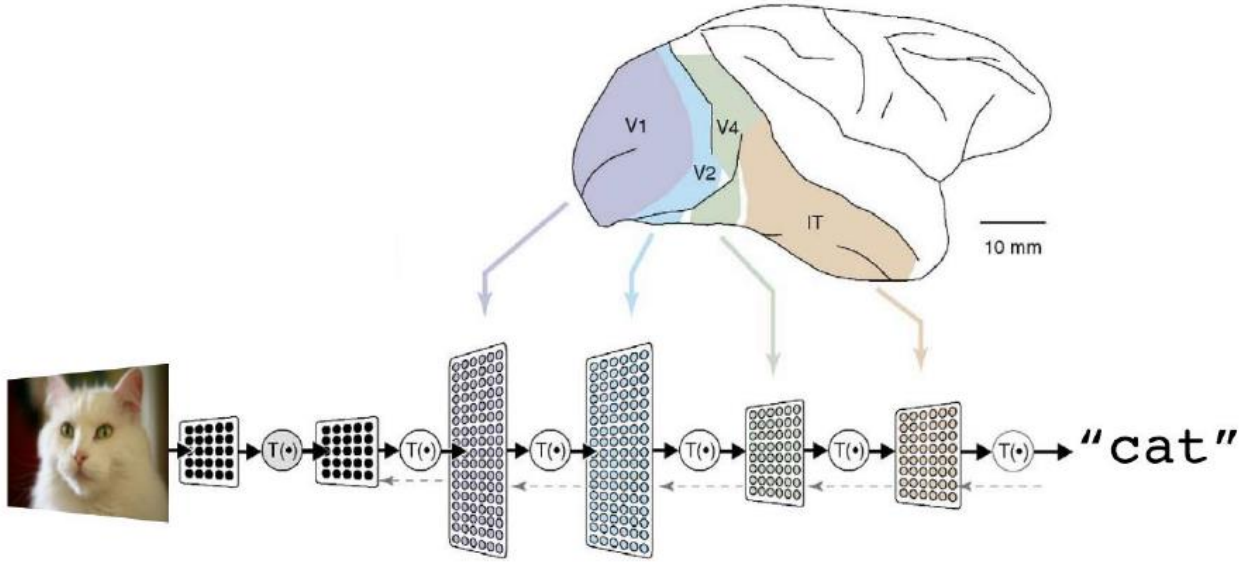


Deep Learning - All-in-one

- Removes the need to manually craft the feature (DL “finds” them instead)
- Removes the need to perform separate learning (DL “learns” them directly)



Why Go Deeper in Neural Networks?

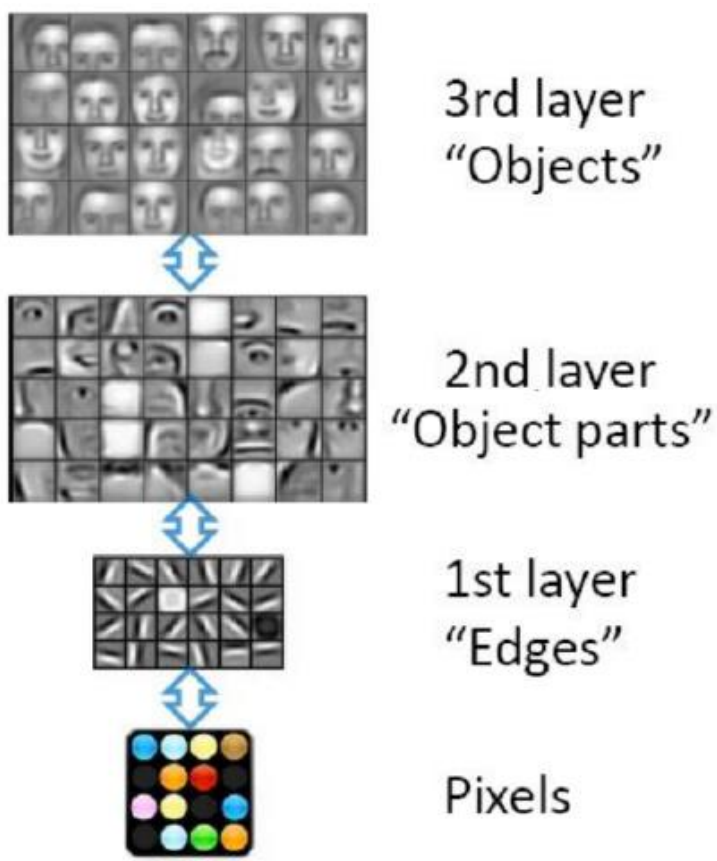


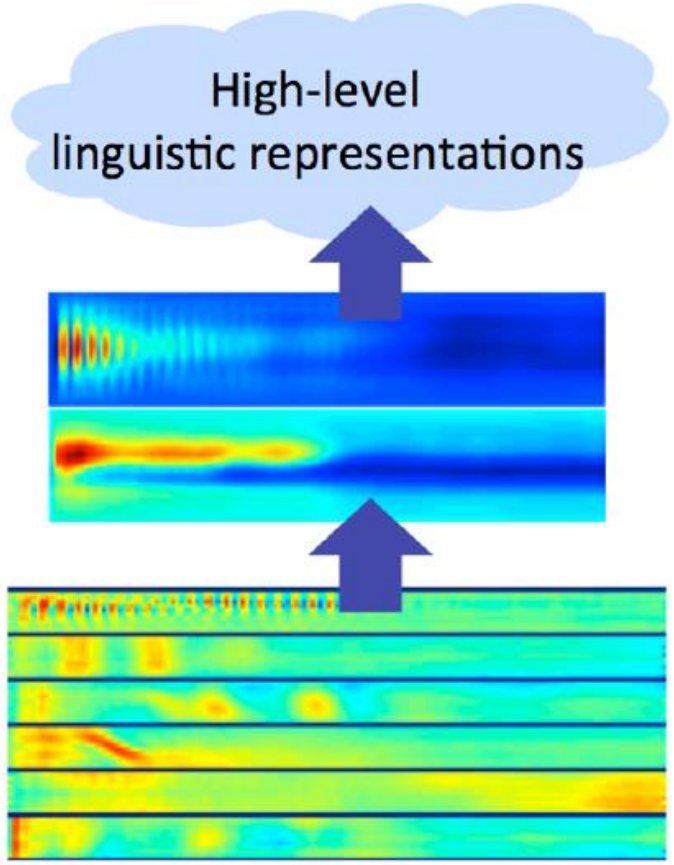
1. Biological Justification -- Hierarchical learning

   - A deep neural network consists of a hierarchy of layers, whereby each layer transforms input data into more abstract representations. The output layer combines these features to make predictions

Why Go Deeper in Neural Networks?

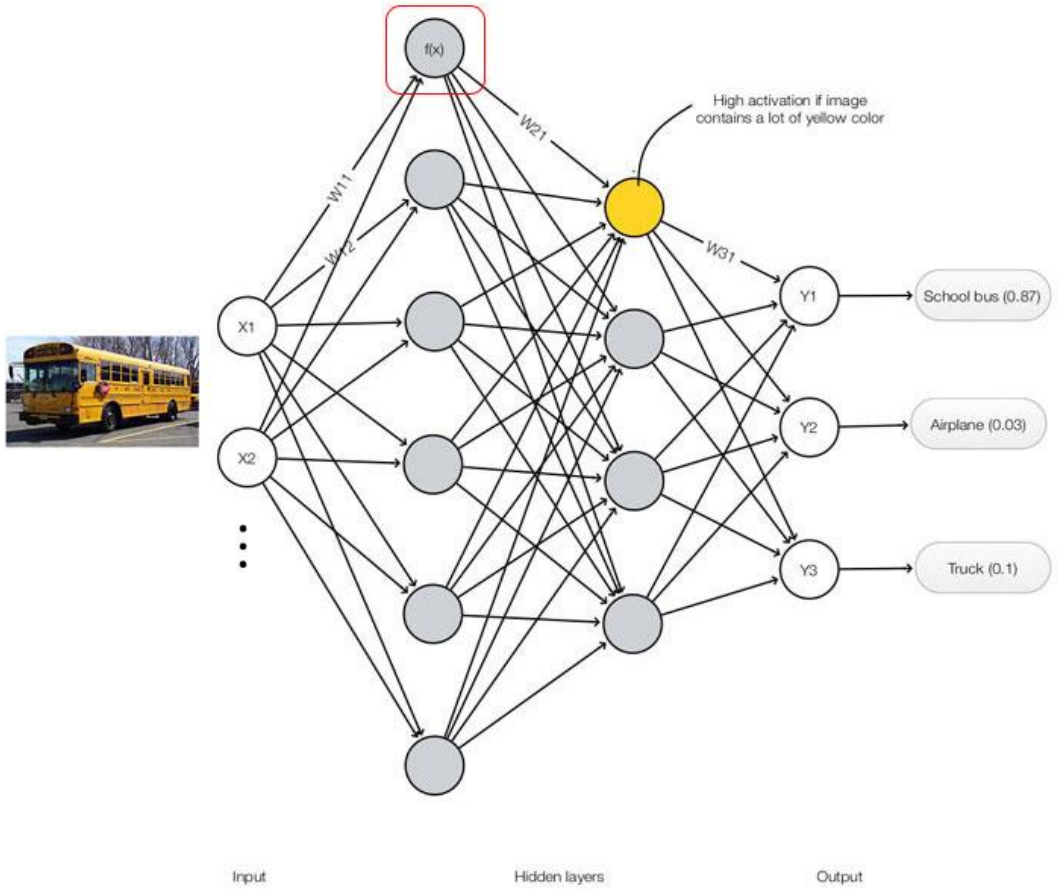
2. Different levels of abstraction
    - Natural progression from low to high level structure as seen in natural complexity





How “deep” is a deep neural network?

- Credit Assignment Path (**CAP**) or (number of hidden layers + 1) more than 2
- CAP > 10 : Very deep



“There is no universally agreed upon threshold of depth dividing shallow learning from deep learning, but most researchers in the field agree that deep learning has multiple nonlinear layers (CAP > 2) and Schmidhuber considers CAP > 10 to be very deep learning”

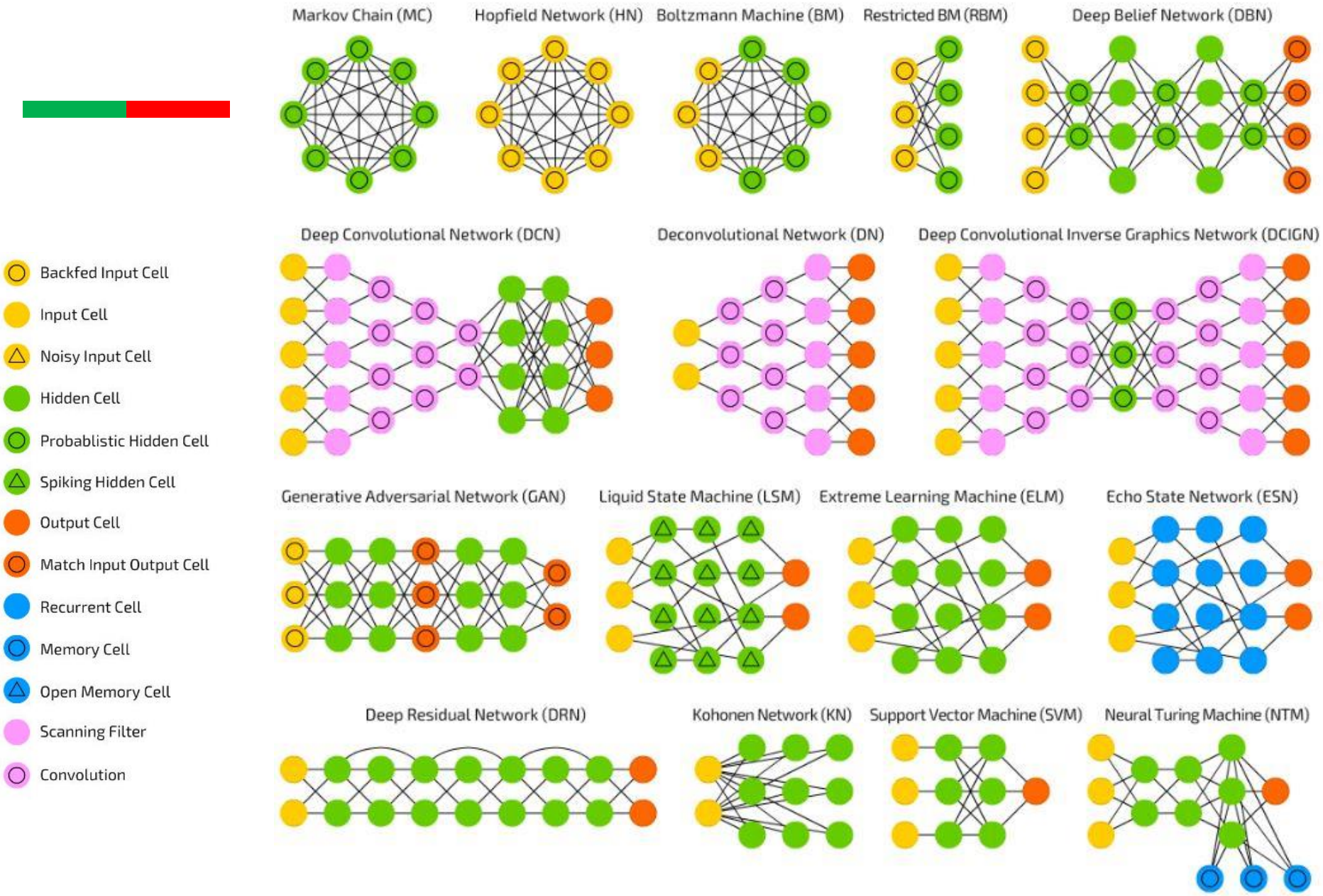
Deep Learning: Strengths

- **Robust**
  - Features are automatically learned to be optimal for the task at hand
- **Generalizable**
  - The same DL approach/architecture can be used for many different applications and data types
- **Scalable**
  - Performance improves with more data, method is massively parallelizable

Deep Learning: Weaknesses

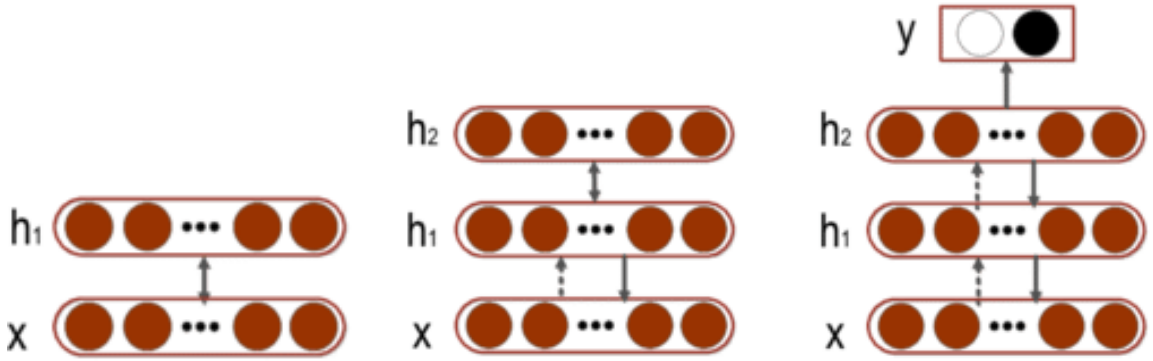
- **Requires large datasets** (typically), also means long training periods.
- **Not necessarily > traditional ML methods** like SVMs, decision trees
- Learned features are often **difficult to understand or explain**.
  - Many vision features are also not really human-understandable (e.g, concatenations/combinations of different features).
- Requires **a good understanding of how to design new models** for new tasks, or using new forms of data, etc.

Deep Learning Flavours


Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Open Memory Cell
Scanning Filter
Convolution

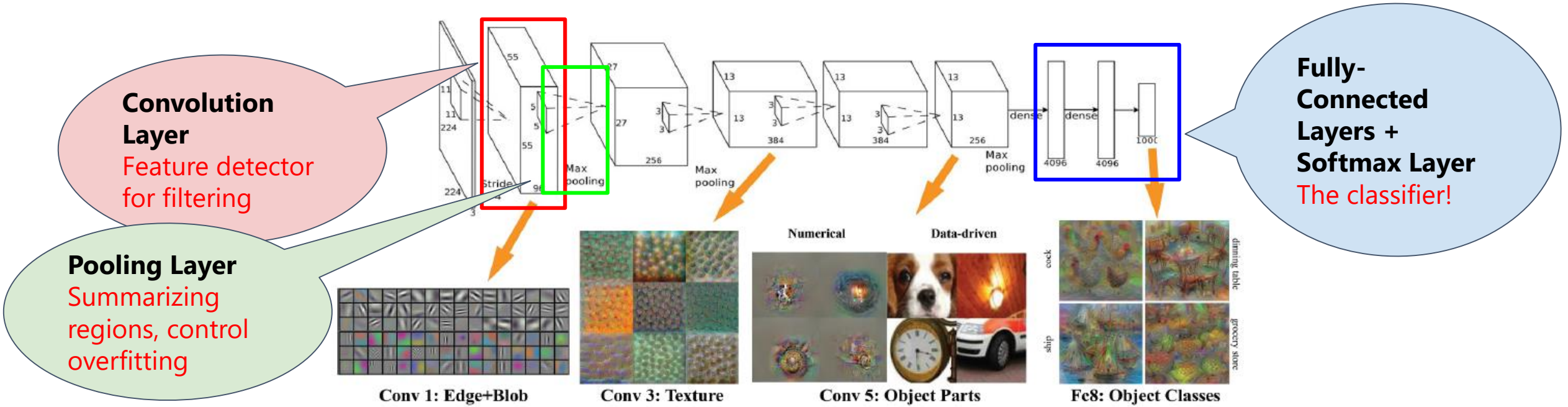
Stacking Neural Networks

- Early attempts, pioneered by Geoffrey Hinton:
  - Deep Belief Networks (DBN) or Stacked Restricted Boltzmann Machines (RBM),
  - Stacked AutoEncoders (SAE)
- Idea:
  - Layer-wise Unsupervised Pre-training
  - Train last layer in supervised mode



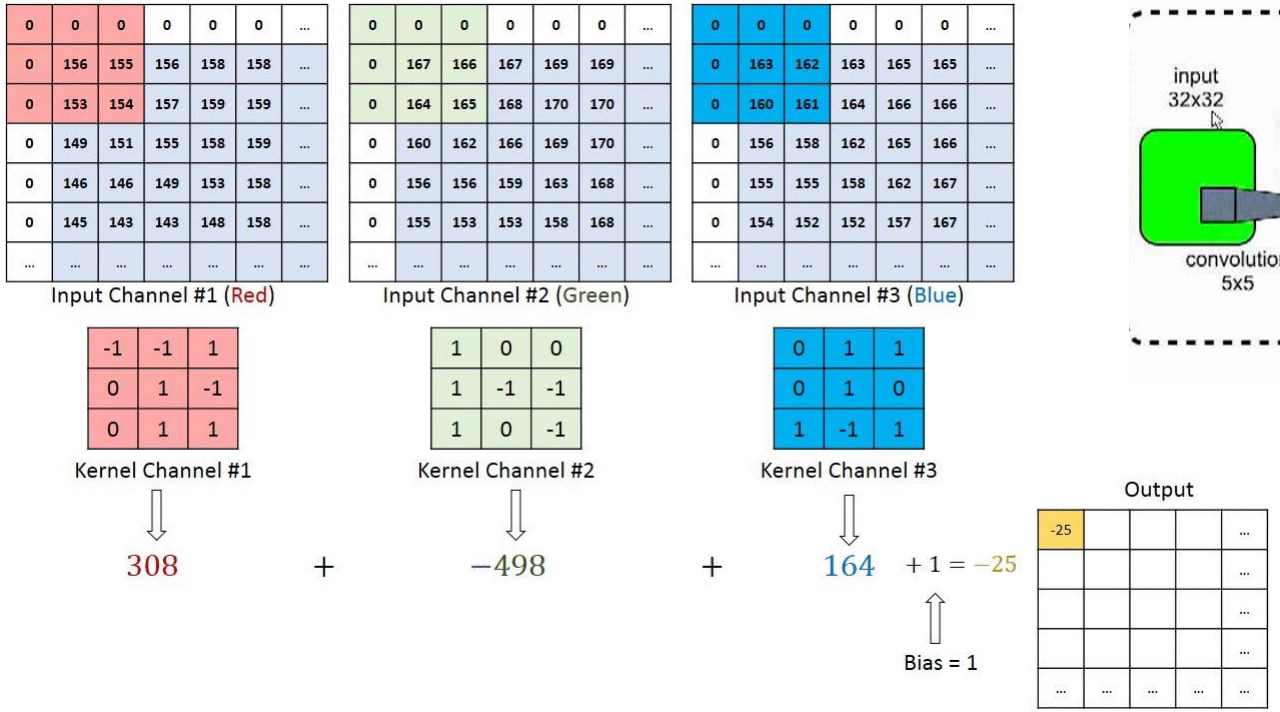
Convolutional Neural Networks (CNN)

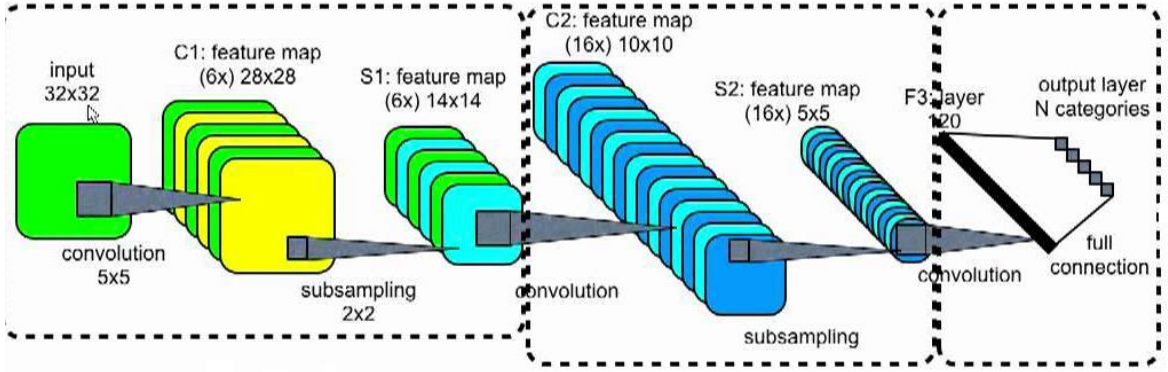
- Learn multiple layers of transformations, stacked to extract a progressively more sophisticated representation of the input



Convolutional Neural Networks (CNN)

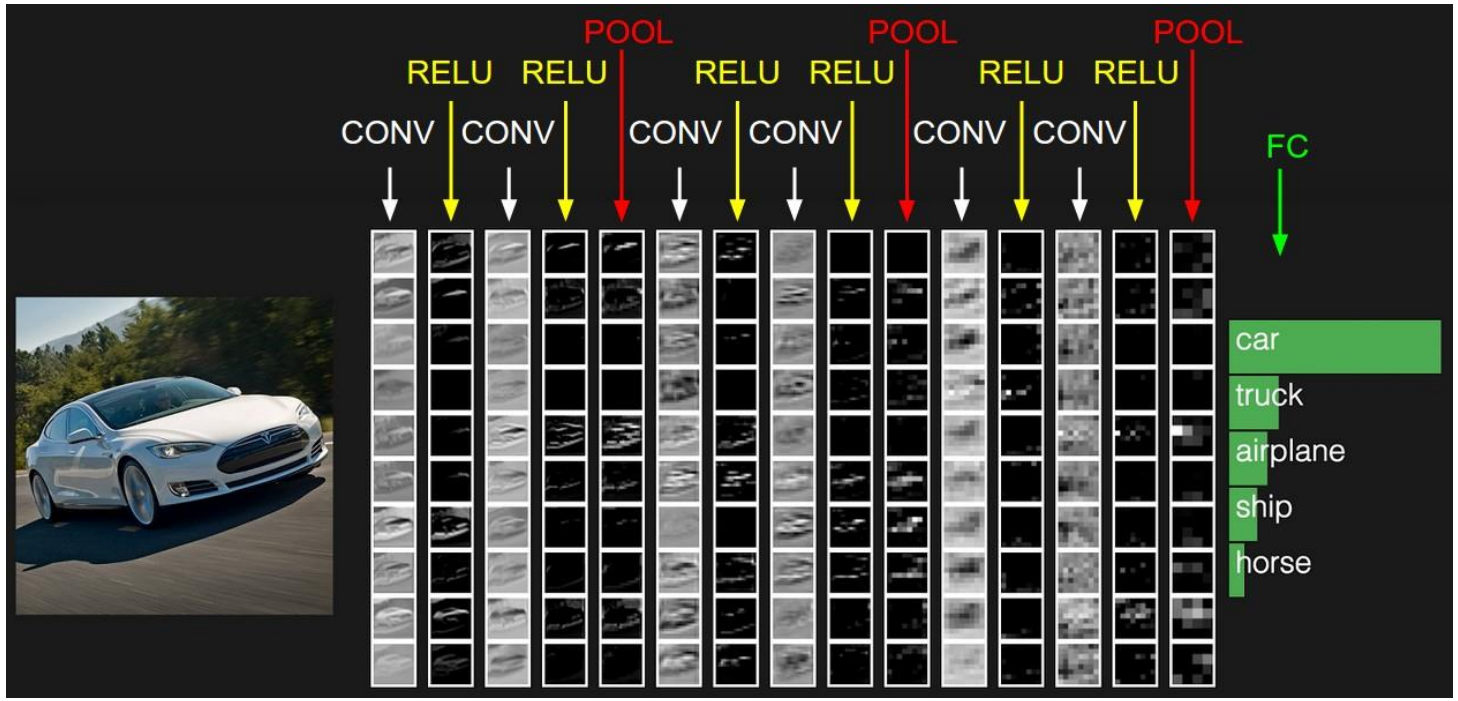
- Each neuron in the Convolutional layer is mapped to a receptive field



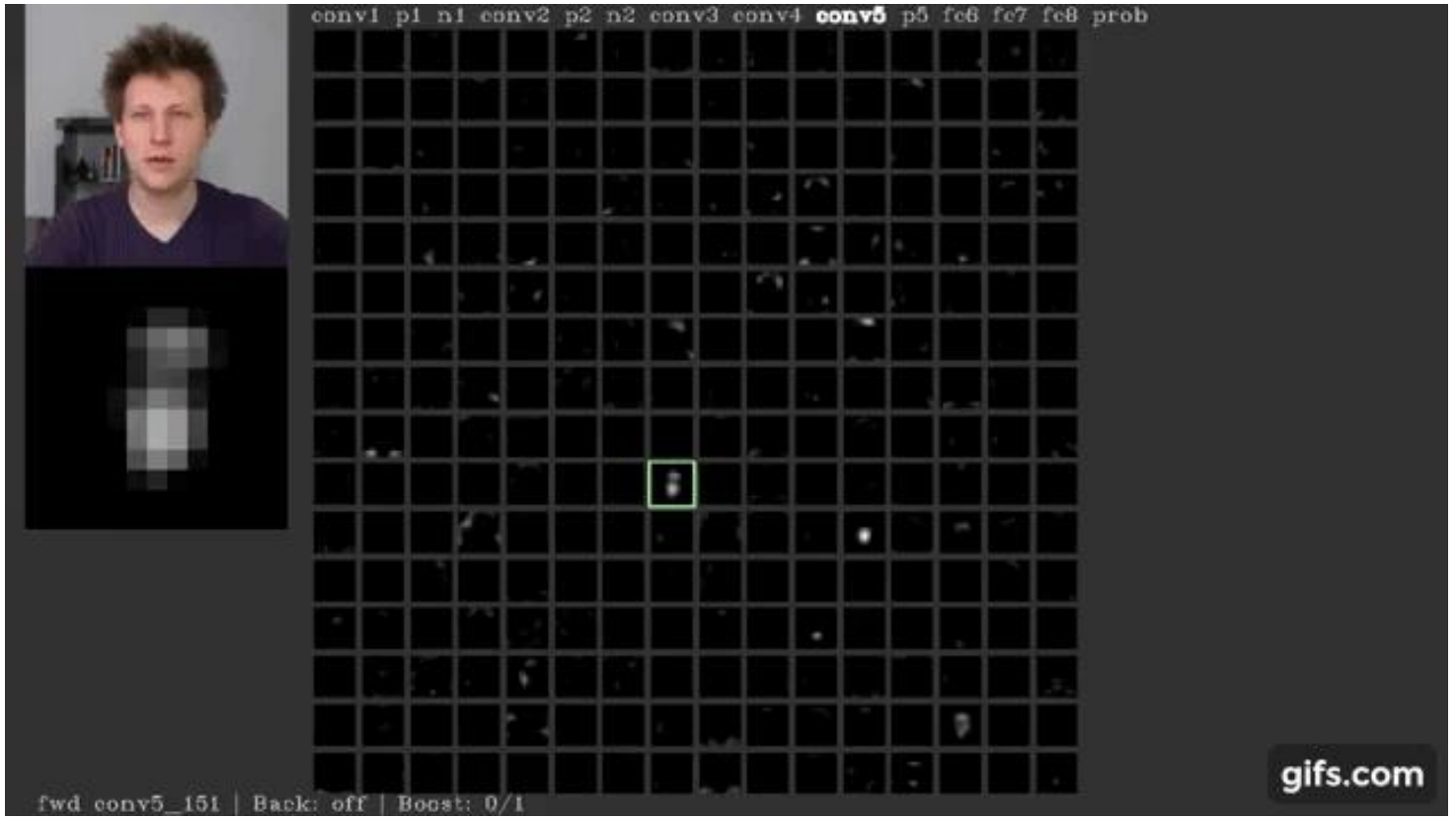


Convolutional Neural Networks (CNN)

- Pioneered by Yann LeCun back in 1989, why didn't it take off then?



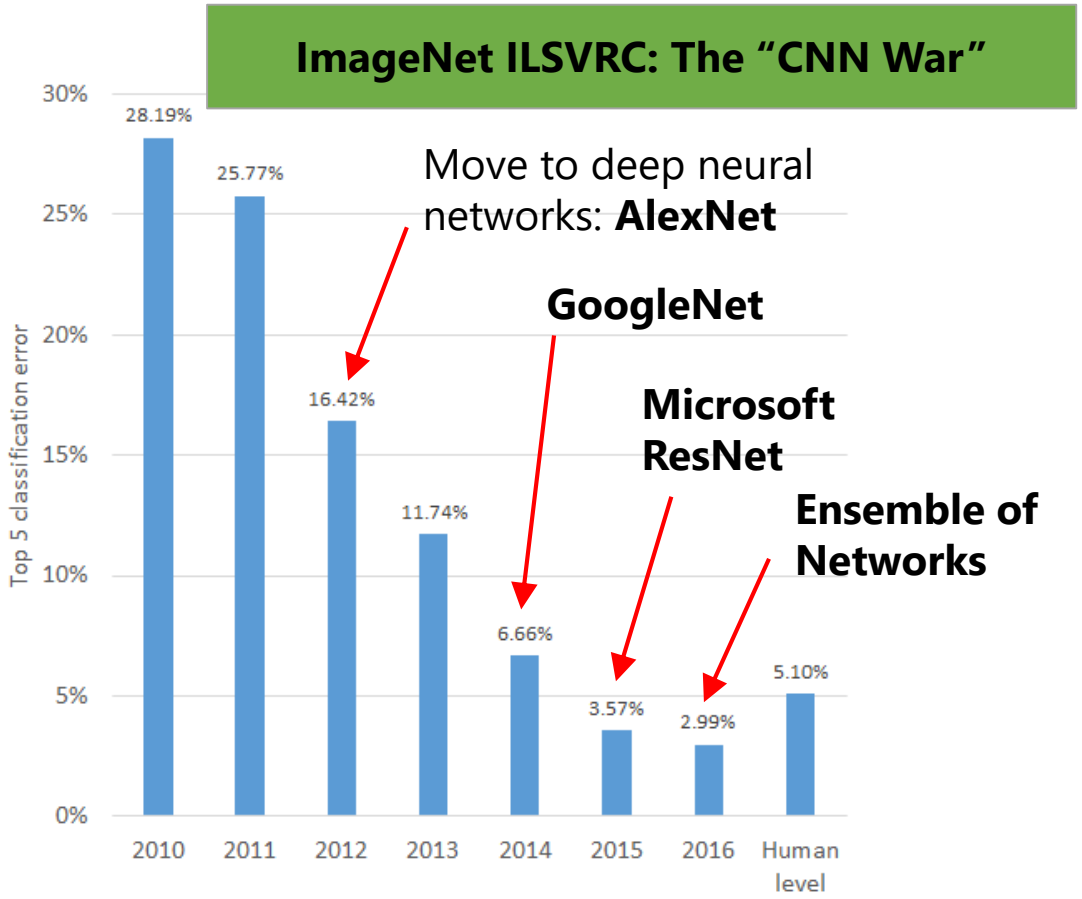
Convolutional Neural Networks (CNN)

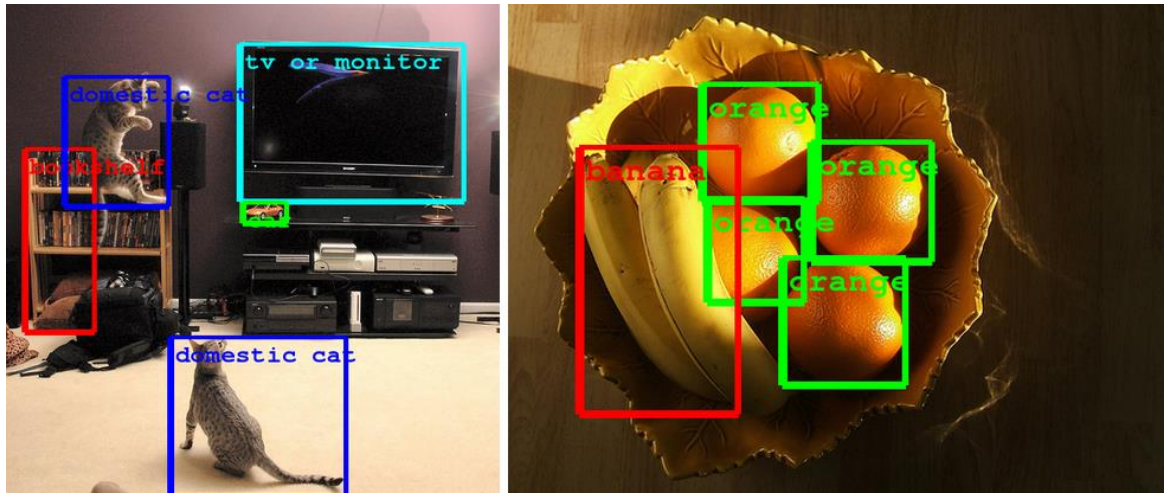


- CNNs can be visualized to gain better understanding of what the network is learning!

https://github.com/yosinski/deep-visualization-toolbox

Object Recognition and Detection/Localization





For Detecting objects:
R-CNN, Fast R-CNN, YOLO

Use Case: Visual Search @ Pinterest

Powered by CNNs!

What was used?

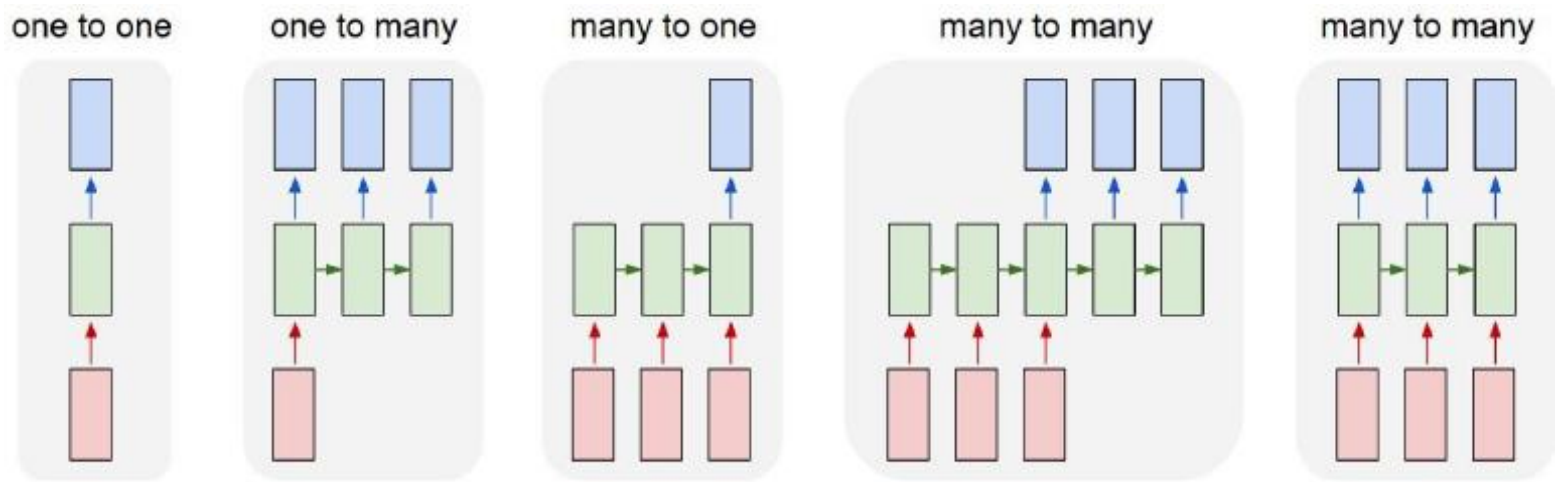
Object detection

Object classification

Recommendation engine:
"Related Pins"

Recurrent Neural Networks (RNN)

- Pioneered by Juergen Schmidhuber and Sepp Hochreiter
- Great choice if your data changes over time, or patterns are sequential
- RNNs can map input sequences to output sequences!



Recurrent Neural Networks (RNN)

- Suitable for: Time series prediction, handwriting recognition, speech recognition, stock market forecasting, text translation, sentiment analysis

Single input, Sequential output

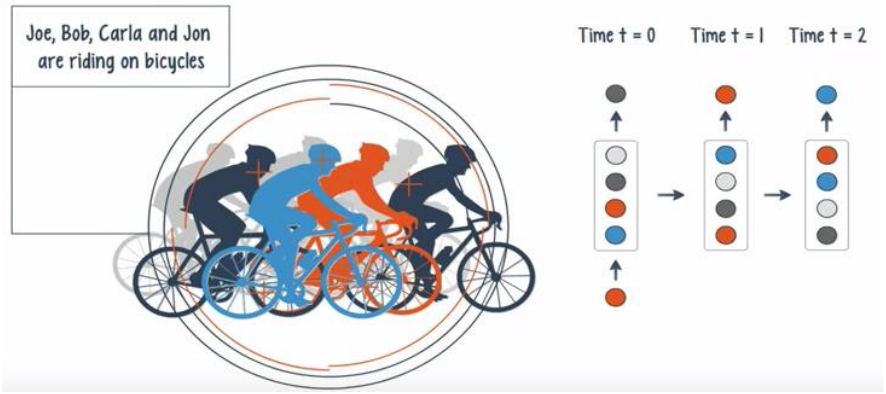
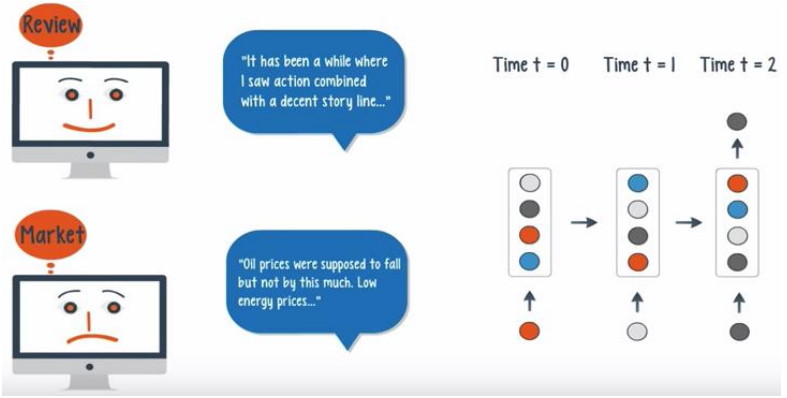


Image captioning

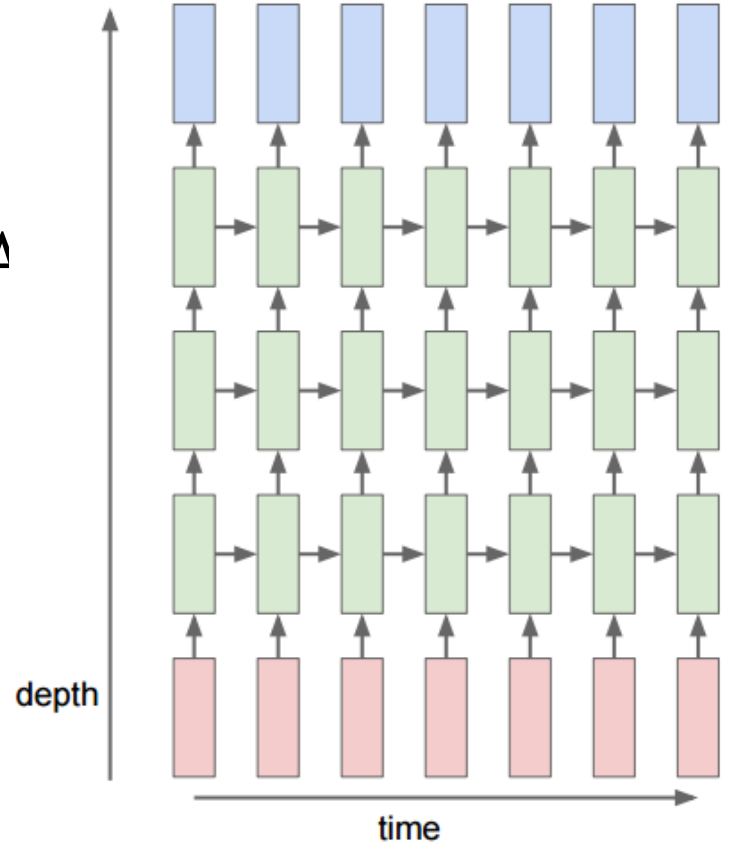
Sequential input, Single output



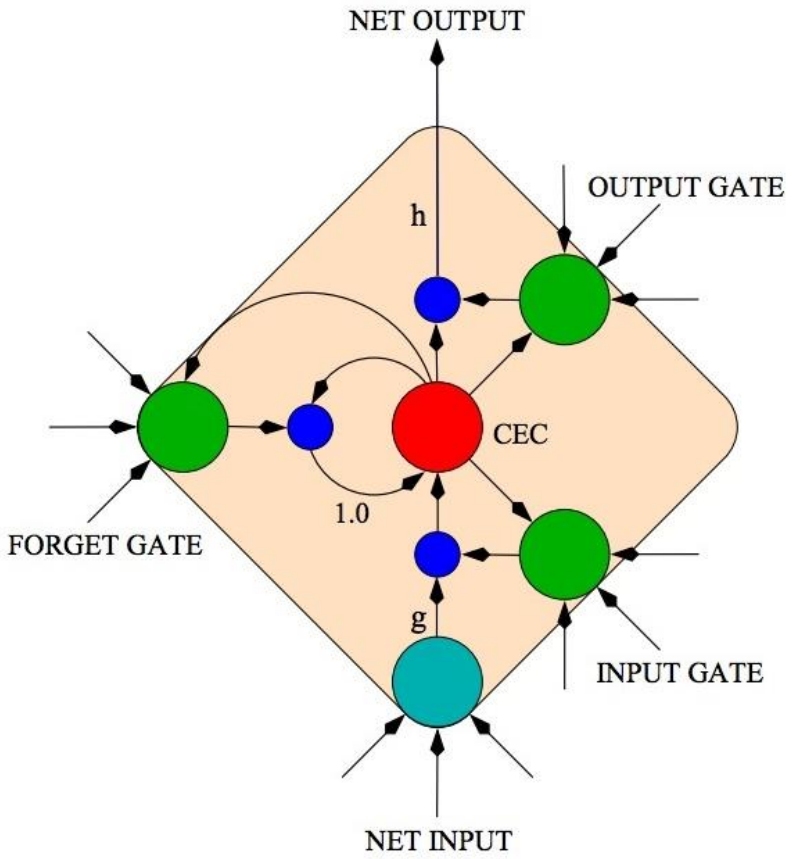
Sentiment analysis

Recurrent Neural Networks (RNN)

- Challenges:
  - Difficult to train, sensitive to parameters
  - Vanishing gradient problem just got worse! (A 100 time step RNN = 100 layer MLP!)
- RNNs can also be stacked into multiple layers to create more complex representations



Long Short-Term Memory (LSTM) RNN



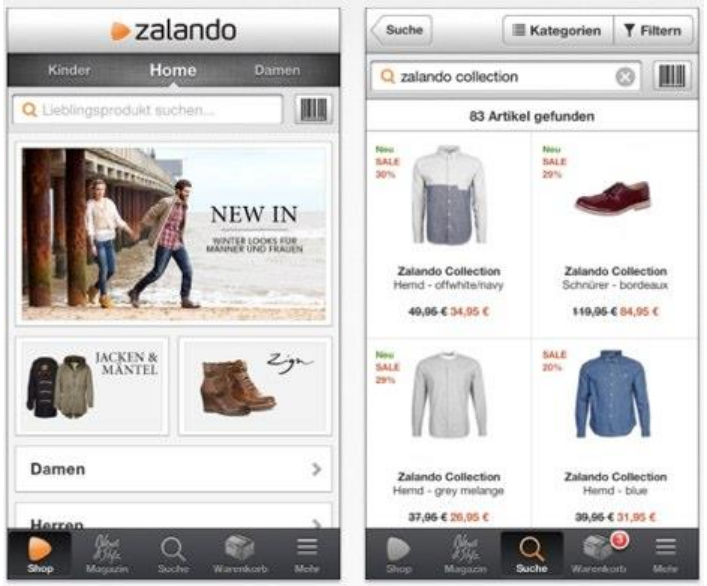
- **LSTM** is a variant of RNN that provides better back-propagation dynamics,
- LSTM units give the network memory cells to read, write and reset operations. During training, the network can learn when it should "remember" and when it should "forget"

LSTM: Modeling Text Data

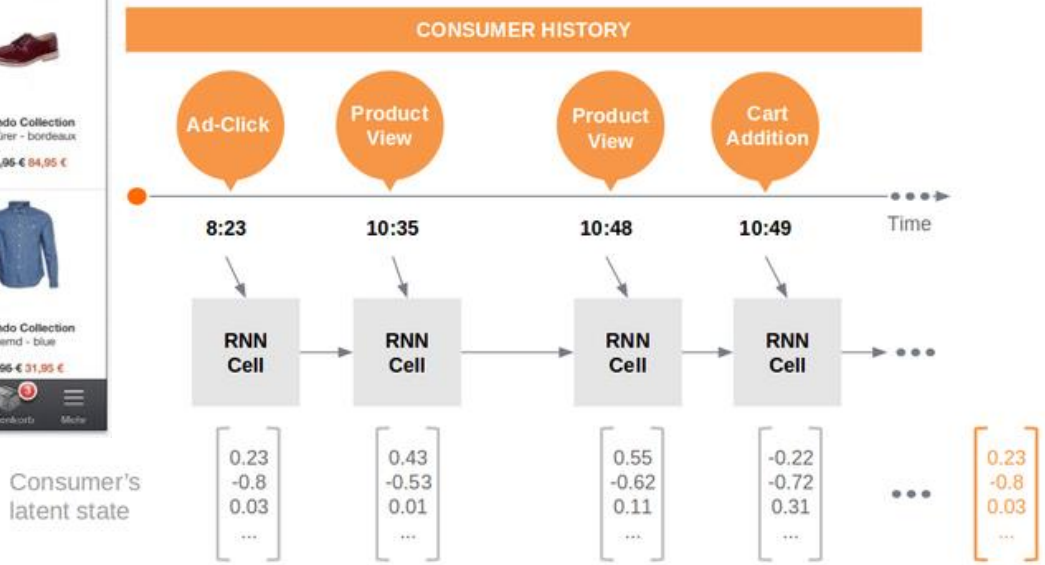
```
This is one of Crichton's best books. The characters of Karen Ross, Peter Elliot,
Munro, and Amy are beautifully developed and their interactions are exciting,
complex, and fast-paced throughout this impressive novel. And about 99.8 percent of
that got lost in the film. Seriously, the screenplay AND the directing were
horrendous and clearly done by people who could not fathom what was good about the
novel. I can't fault the actors because frankly, they never had a chance to make this
turkey live up to Crichton's original work. I know good novels, especially those with
a science fiction edge, are hard to bring to the screen in a way that lives up to the
original. But this may be the absolute worst disparity in quality between novel and
screen adaptation ever. The book is really, really good. The movie is just dreadful.
```

- Text is sequential: Insert characters into LSTM to predict probabilities of the next letter
- Text models can be turned into sentiment classifiers: Cater to longer bunch of sentences, like product/book/movie reviews

Use Case: Forecasting Consumer Preferences @Zalando



Powered by RNNs!

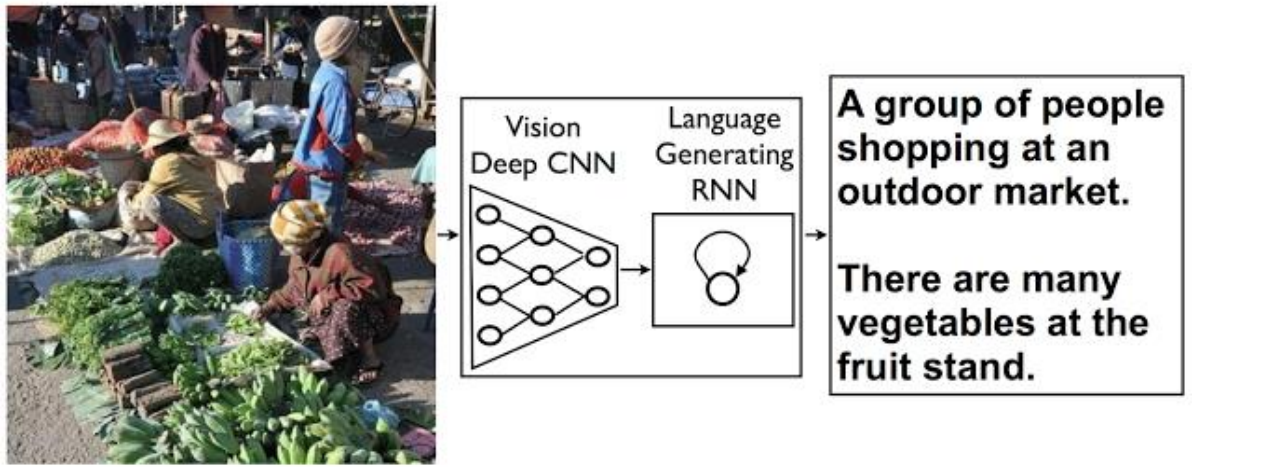


What was achieved?

Track consumer sessions

"Memorize" patterns in consumer behaviour to predict probability of orders

Hybrid Networks

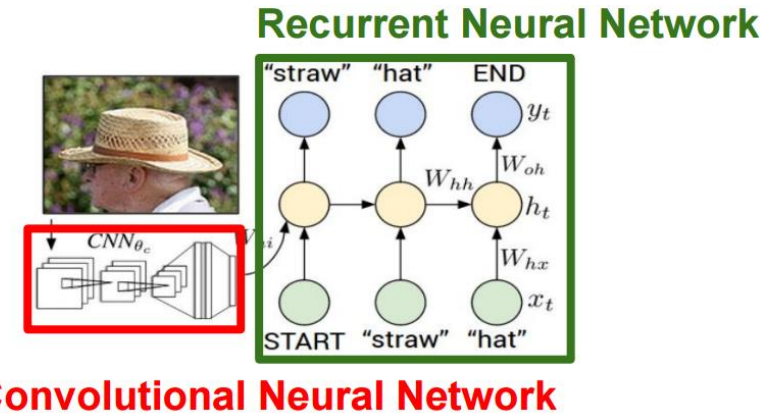


A close up of a child holding a stuffed animal

Two pizzas sitting on top of a stove top oven

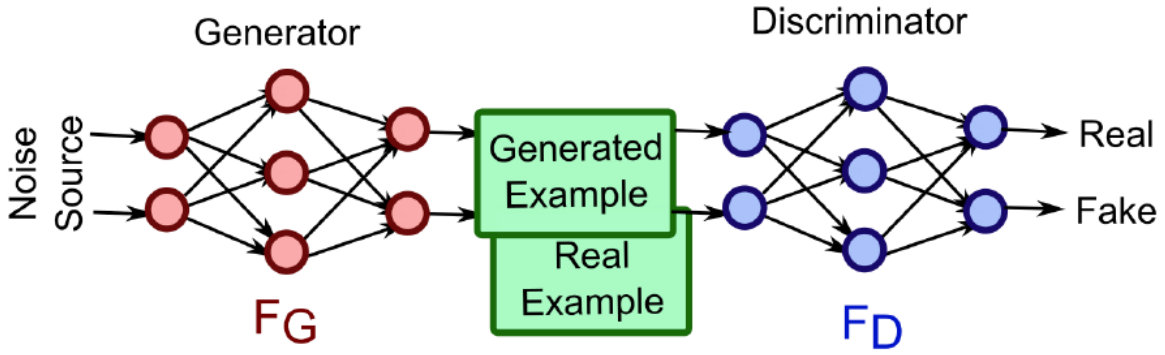
A man flying through the air while riding a skateboard

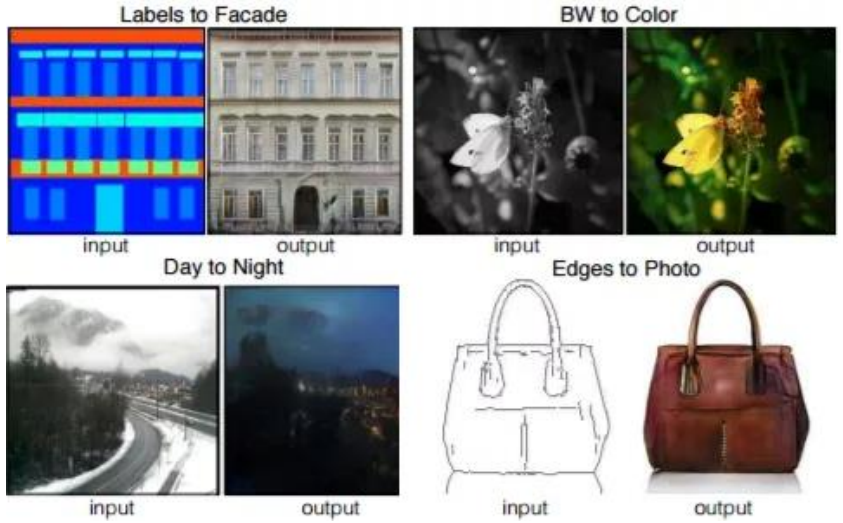
- **Image Captioning:** Combines CNN and RNN
  - Generates fitting natural language captions only based on the pixel data



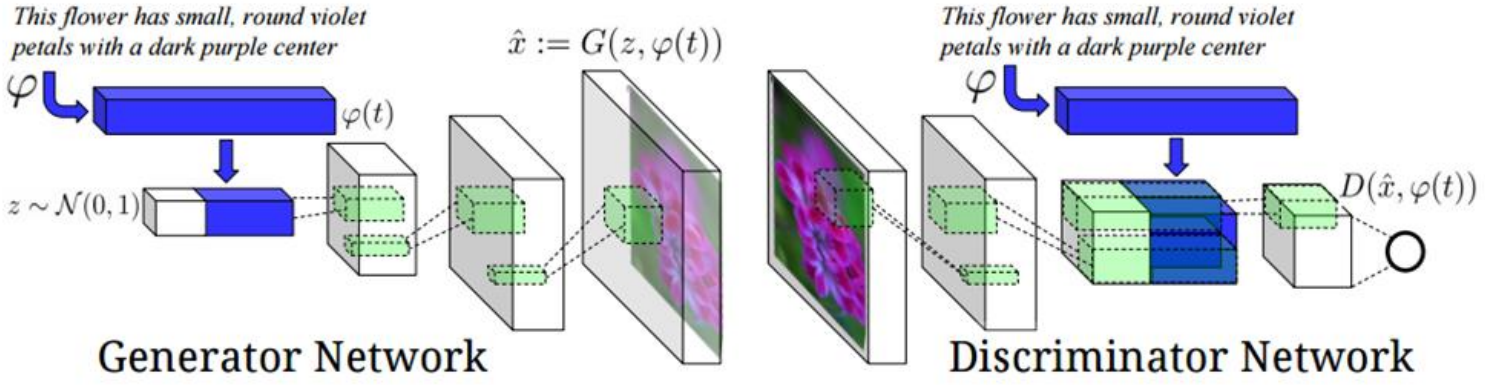
Generative Adversarial Networks (GAN)

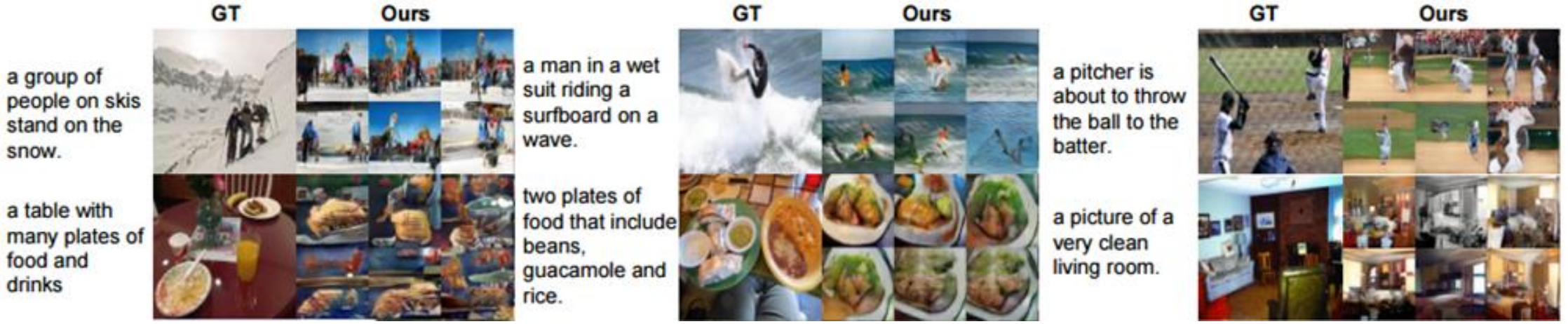
- Inspired by adversarial competition in Game Theory
  - **Two networks:** One tasked to generate content, the other to judge content



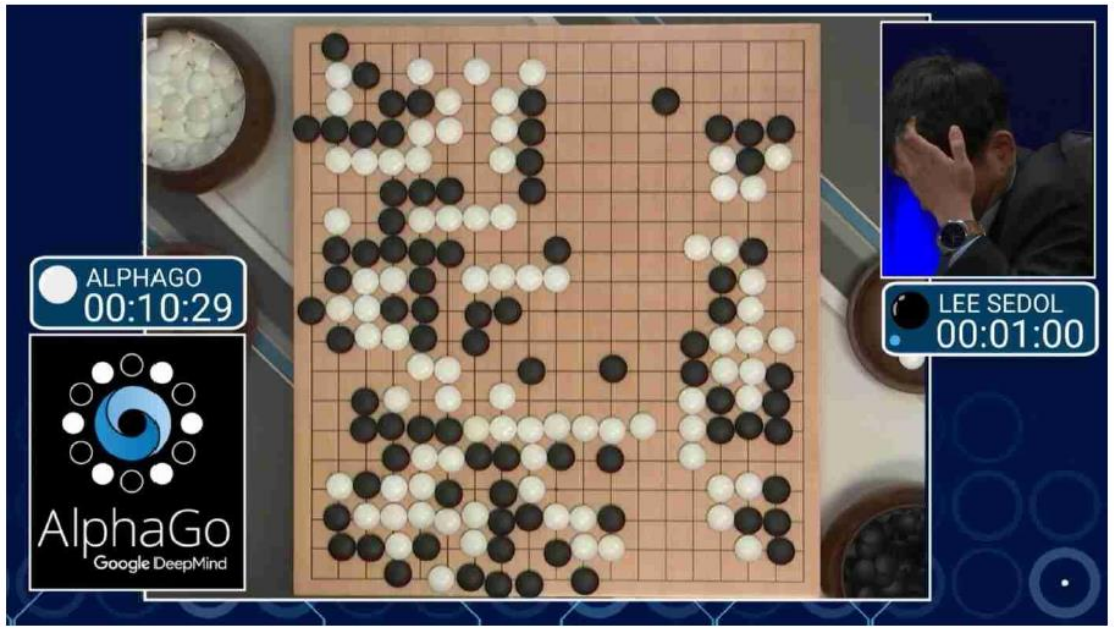


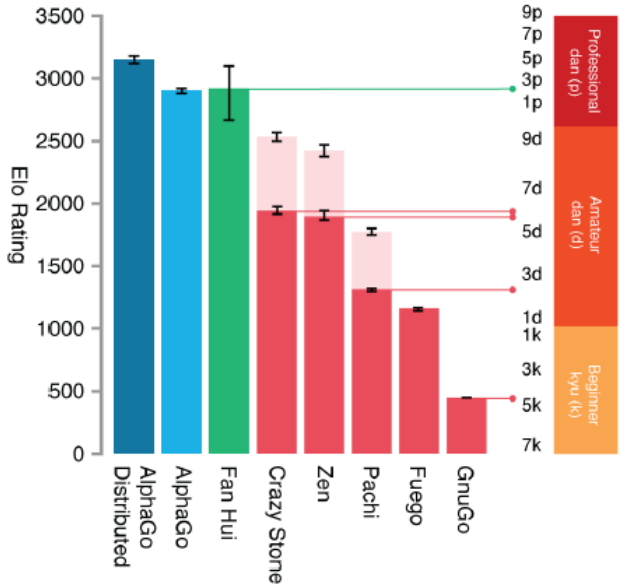
GAN: Text-to-Image Synthesis





Deep Learning for Games

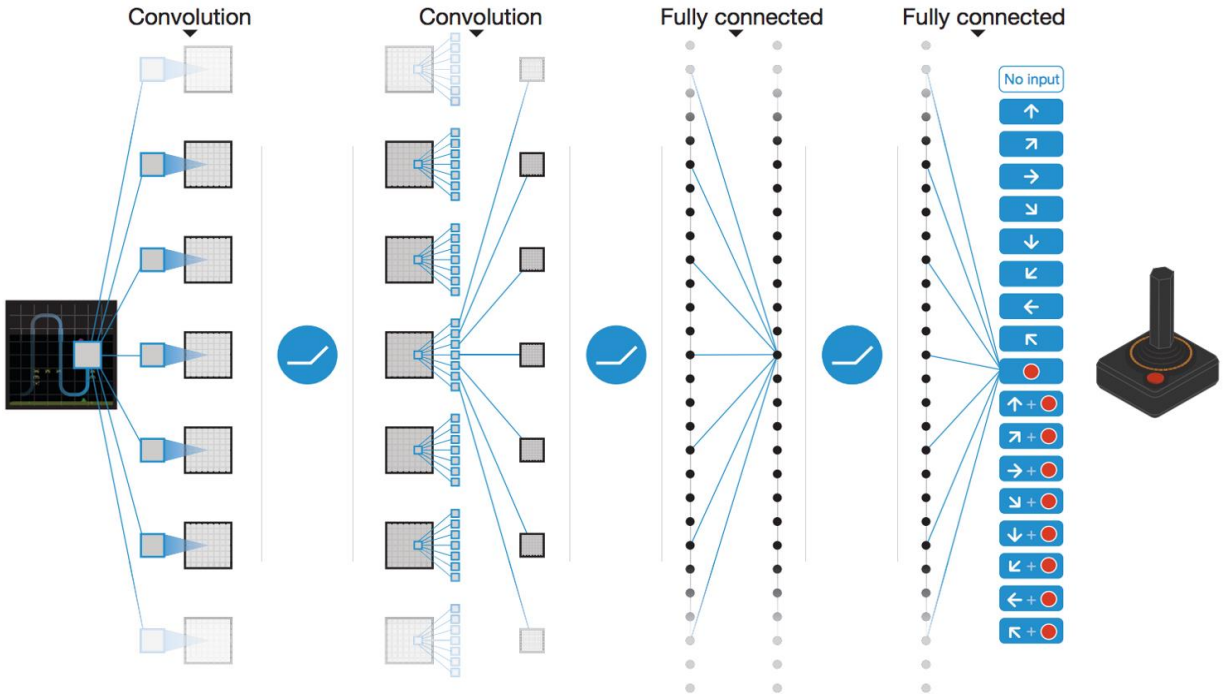




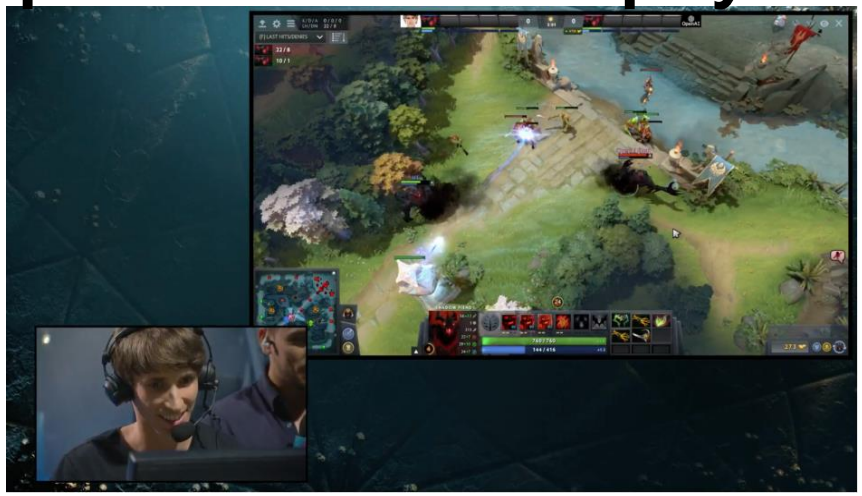
Examines thousands of human GO moves, and then masters it by playing with itself over and over again.

History was made: Google DeepMind's AlphaGo beats Go champion Lee Sedol

Deep Learning for Games



Just In: OpenAI's bot beats professional Dota 2 players

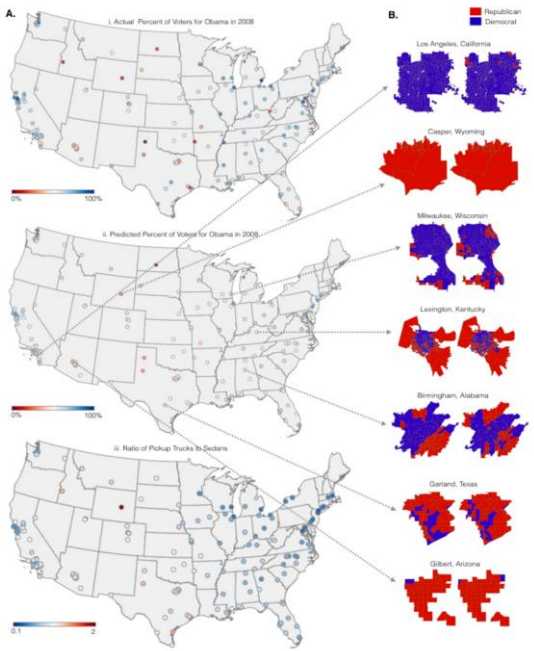


- Deep Q-Network (DQN) is a model-free approach to reinforcement learning using deep networks in environments with discrete action choices

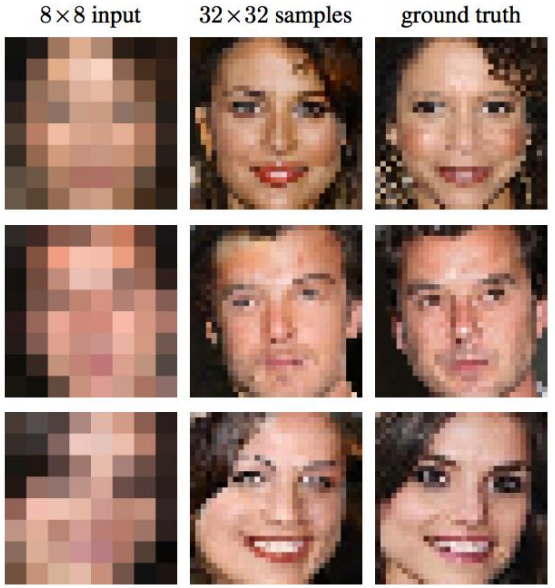
What Else Can Deep Learning Generate?

Generate Expressive Music

Learning Demographics and Predicting Election Results based on Street View Images

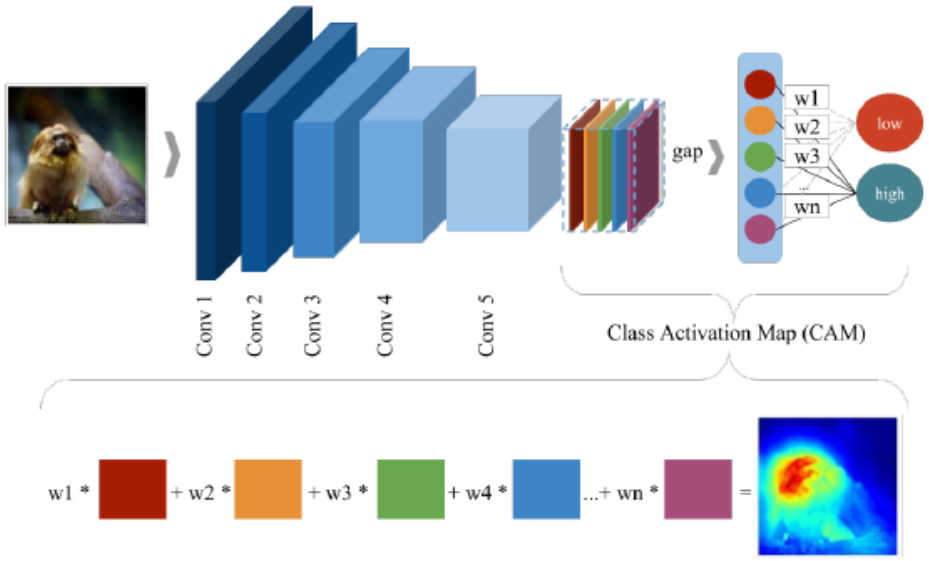


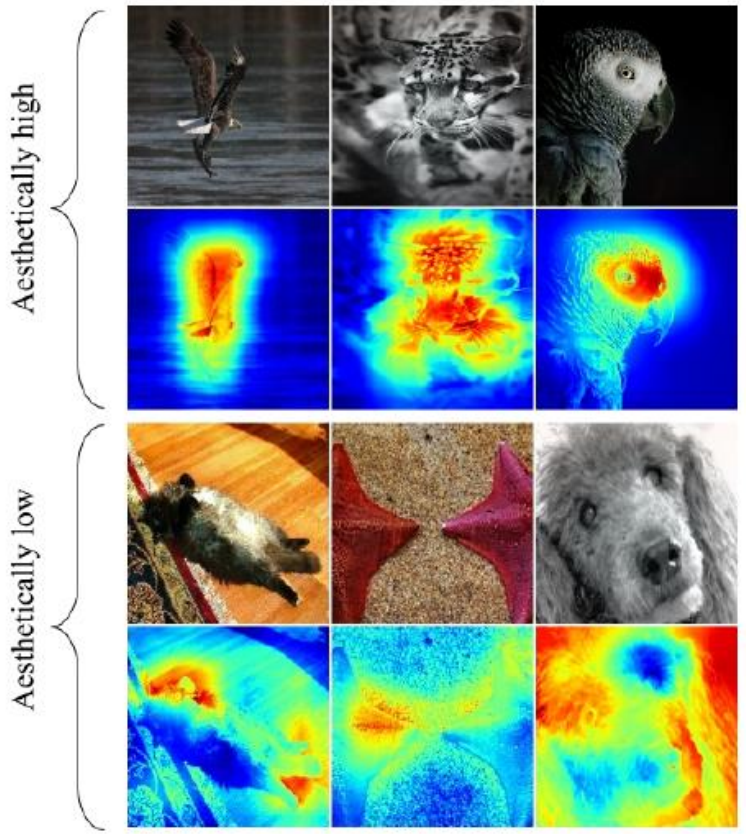
Pixel Restoration “CSI-style”



Some Homegrown Research

Predicting and Localizing Aesthetics in Images



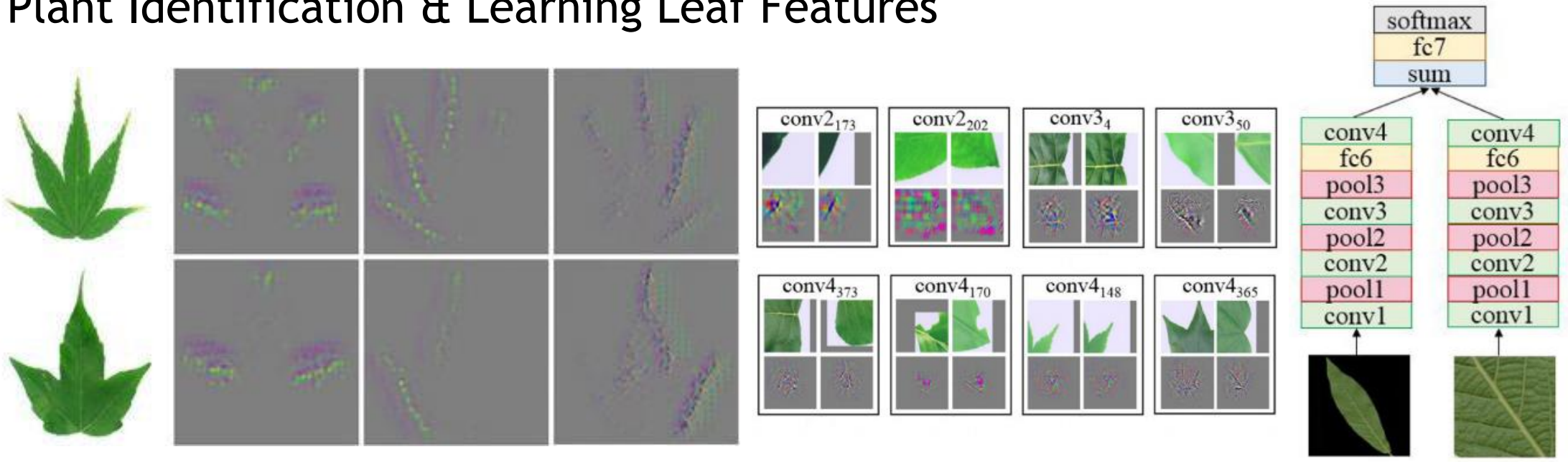


State-of-the-art:
82.27% (Text-augmented MultGAP Network, ICIP 2017)

Some Homegrown Research

UNIVERSITY OF MALAYA

Plant Identification & Learning Leaf Features



MalayaKew and Flavia datasets

Top-1 accuracy: 96.3% (Early Fusion conv-sum, PR 2017)

Deep Learning Tools - Open Source!

Is Deep Learning over-hyped?

Is Deep Learning a black box that is unexplainable?

What should I do if I want to get started?

Applied Deep Learning

Course Preview

Applied Deep Learning: 5-day short course

- Fundamental concepts, Hands-on practical
- Basic knowledge in Python programming & ML (added advantage)
- Course outline:
  1. Machine Learning Primer - Classification & Regression
  2. Fundamentals of Neural Networks (MLPs, Shallow Networks)
  3. Deep Networks I: Convolutional Neural Networks
  4. Deep Networks II: Recurrent Neural Networks
  5. Tricks and Techniques in Deep Learning (Regularization, Data Augmentation, Ensembles)

Contact us

John See
johnsee@gmail.com

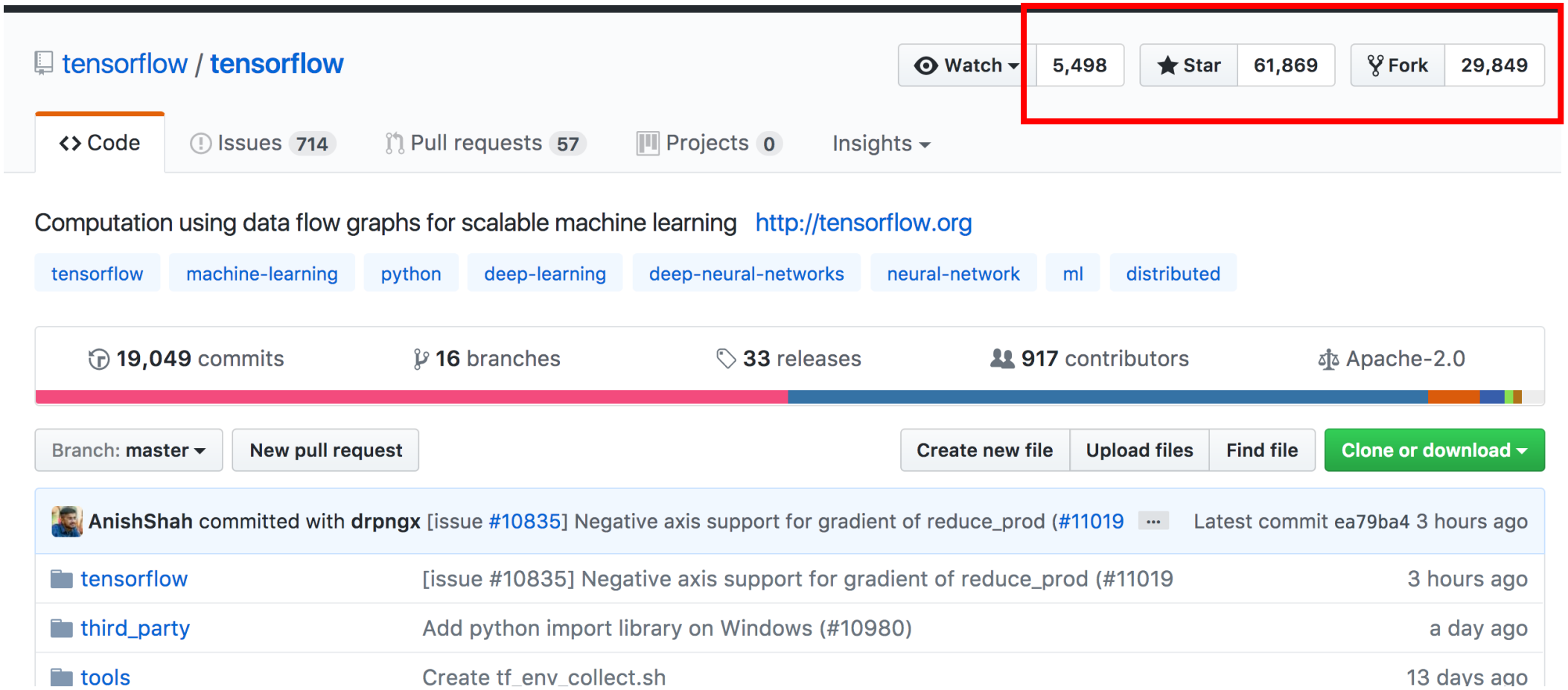
Poo Kuan Hoong
kuanhoong@gmail.com

TensorFlow

What is TensorFlow?

- URL: https://www.tensorflow.org/
- Released under the open source license on November 9, 2015
- Current version 1.2
- Open source software library for numerical computation using data flow graphs
- Originally developed by Google Brain Team to conduct machine learning and deep neural networks research
- General enough to be applicable in a wide variety of other domains as well
- TensorFlow provides an extensive suite of functions and classes that allow users to build various models from scratch.

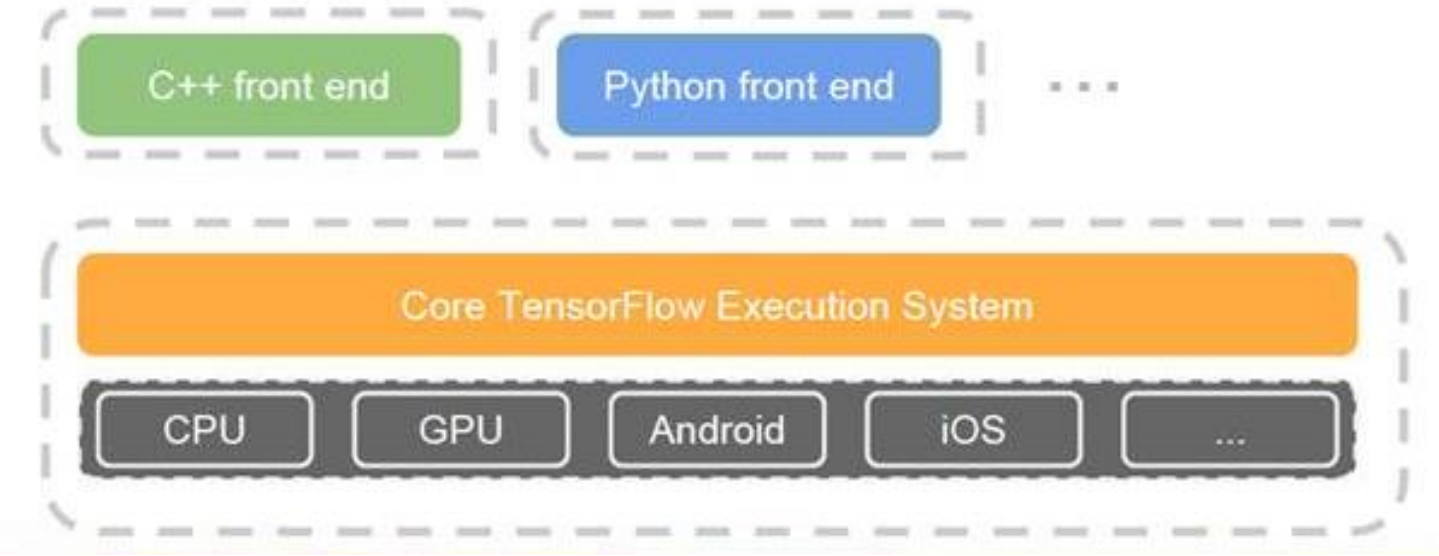
Most popular on Github



https://github.com/tensorflow/tensorflow

TensorFlow architecture

- Core in C++
  - Low overhead
- Different front ends for specifying/driving the computation
  - Python, C++, R and many more



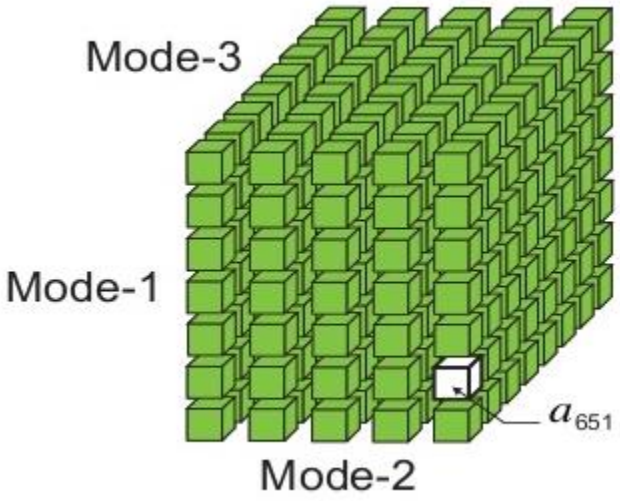
CPU - GPU

- In TensorFlow, the supported device types are CPU and GPU. They are represented as strings. For example:
  - "/cpu:0": The CPU of your machine.
  - "/gpu:0": The GPU of your machine, if you have one.
  - "/gpu:1": The second GPU of your machine, etc.

```
# Creates a graph.
with tf.device('/gpu:2'):
  a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
  b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
  c = tf.matmul(a, b)
# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
# Runs the op.
print(sess.run(c))
```

Why it is called TensorFlow?

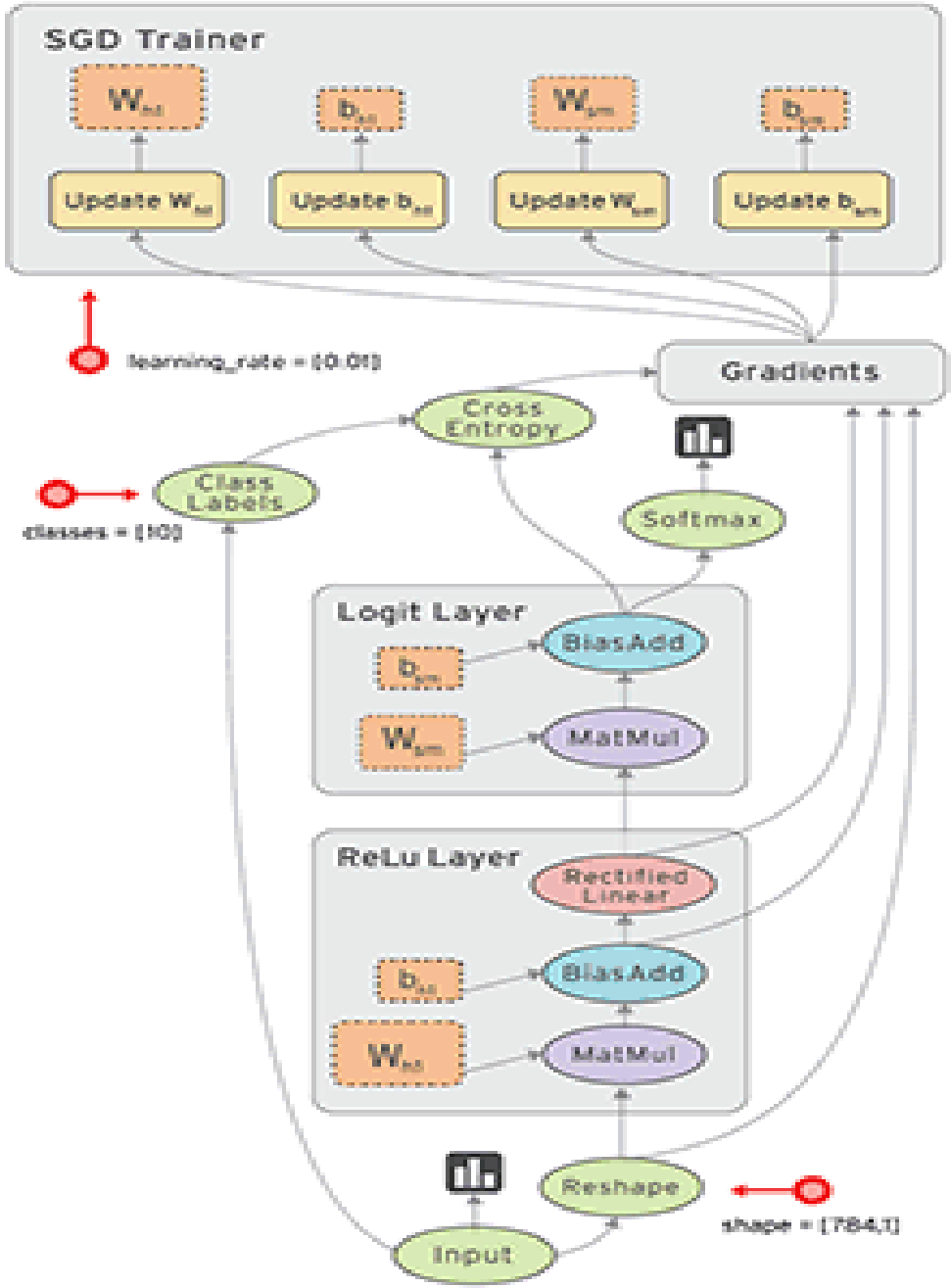
- What is a Tensor?



Dimensions	Example	Terminology
1	0 1 2	Vector
2	0 1 2 3 4 5 6 7 8	Matrix
3	0 1 2 3 4 5 6 7 8	3D Array (3rd order Tensor)
N	... ...	ND Array

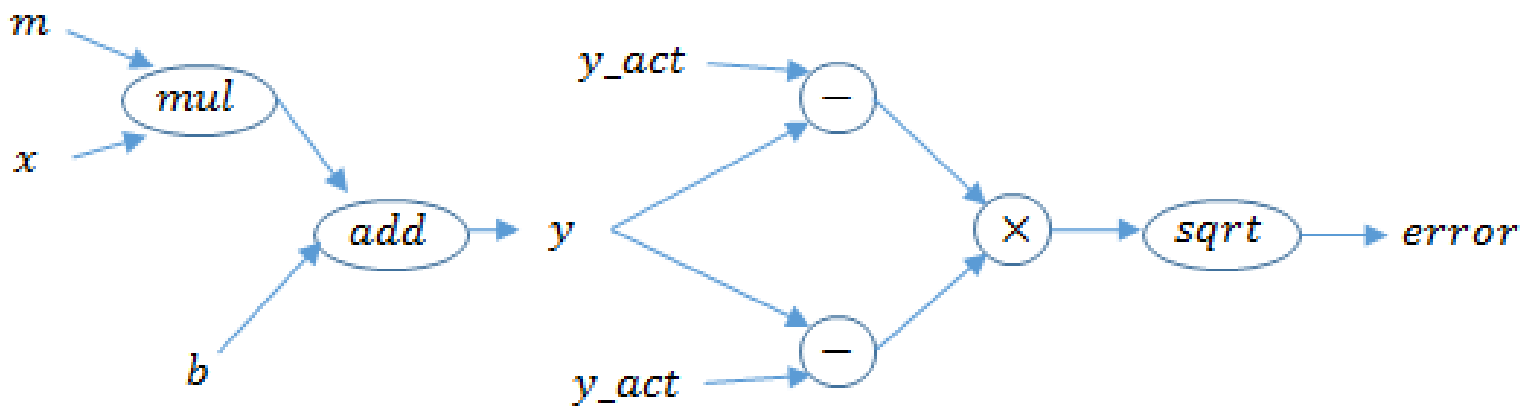
Why it is called TensorFlow?

- TensorFlow is based on computation data flow graph
- TensorFlow separates definition of computations from their execution



Steps to start with TensorFlow

- **Step 1:** Assemble a computational graph
- **Step 2:** Use a session to execute operations in the graph



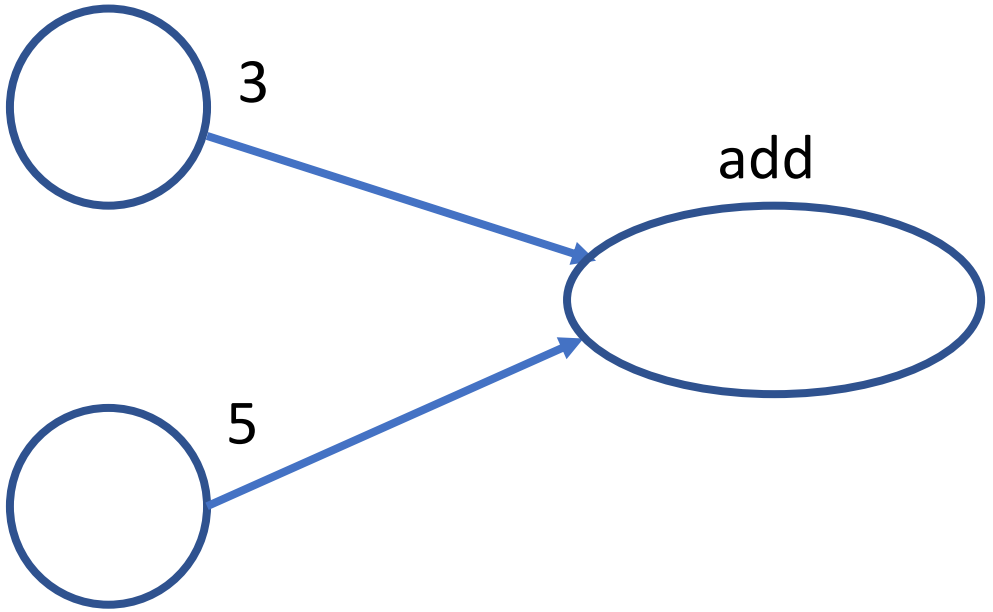
TensorFlow Graph

```
import tensorflow as tf
a = tf.add(3,5)
print(a)
```

```
Tensor("Add:0", shape=(), dtype=int32)
```

```
import numpy as np
a = np.add(3,5)
print(a)
```

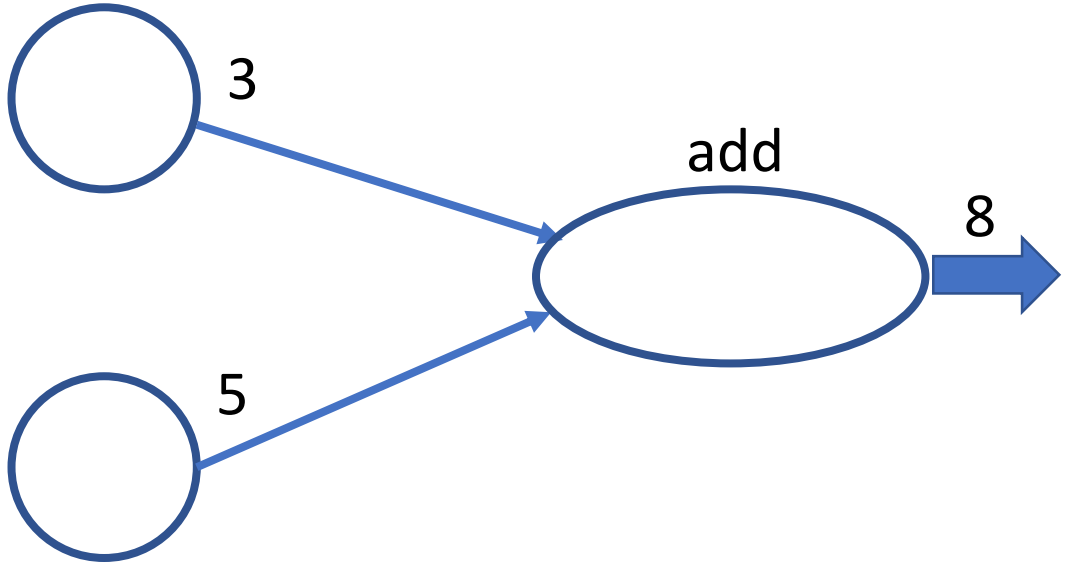
```
8
```



TensorFlow Graph

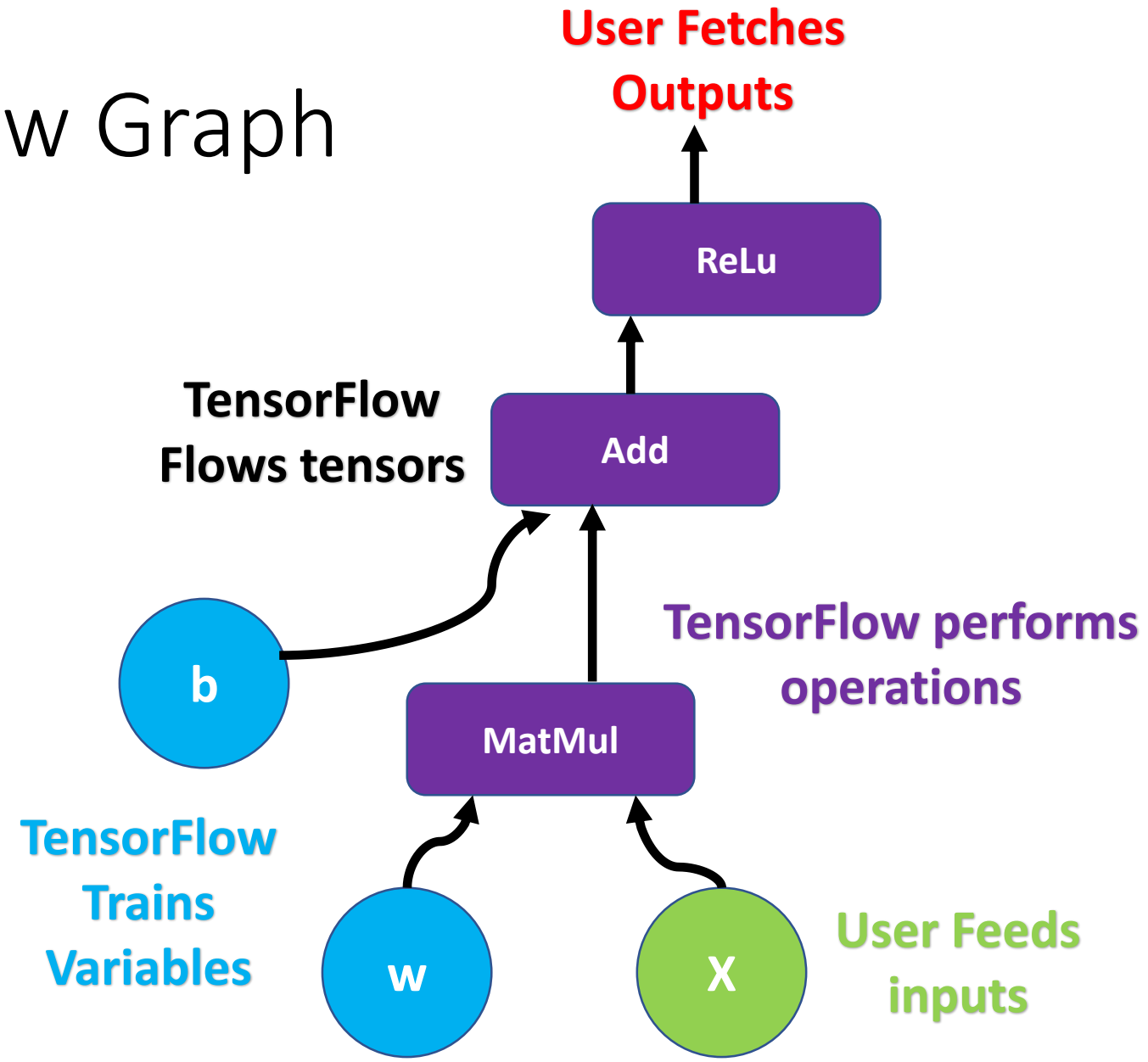
- To get the value of a
- Create a **session**, assign it to variable ‘sess’ so we can call it later
- Within the session, evaluate the graph to fetch the value of *a*

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
print (sess.run(a))
sess.close()
```



- A Session object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

TensorFlow Graph



Variables

- Variables to hold and update parameters.
- Variables are in-memory buffers containing tensors.
- Must be explicitly initialized and can be saved to disk during and after training

```
# create variable a with scalar value
a = tf.Variable(2, name="scalar")

# create variable b as a vector
b = tf.Variable([2, 3], name="vector")

# create variable c as a 2x2 matrix
c = tf.Variable([[0, 1], [2, 3]], name="matrix")

init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)
```

Placeholder

- A placeholder is simply a variable that will be assigned data to at a later date.
- It allows operations to be created and computation graph to be built, without needing the data.

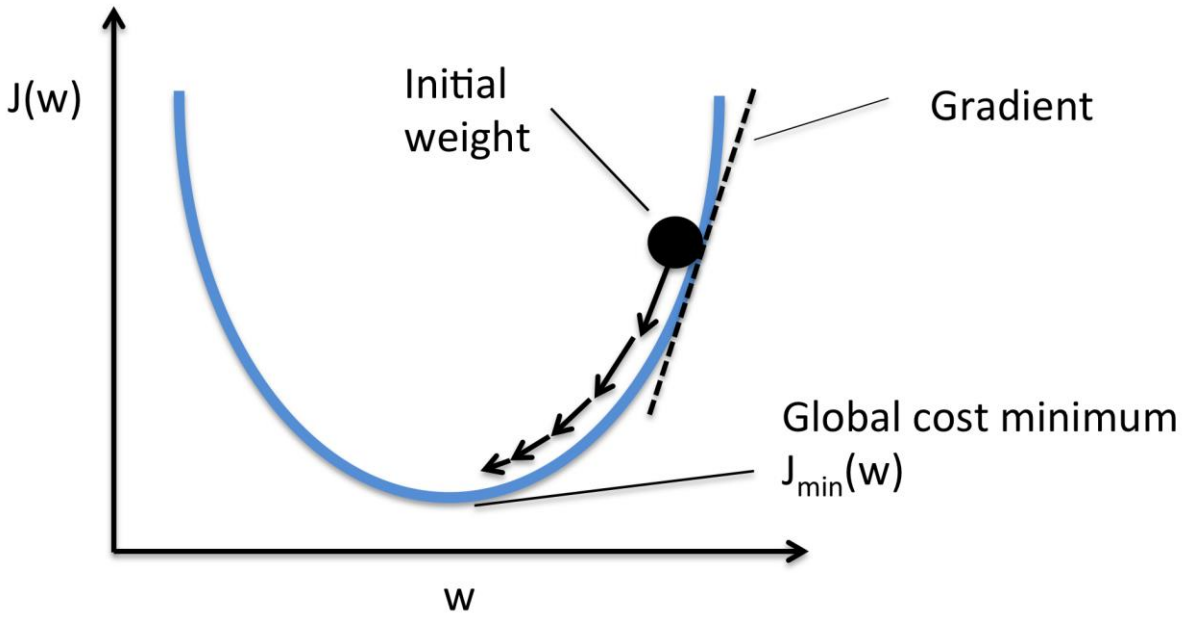
```
x = tf.placeholder("float", None)
y = x * 2

with tf.Session() as session:
    result = session.run(y, feed_dict={x: [1, 2, 3]})
    print(result)
```

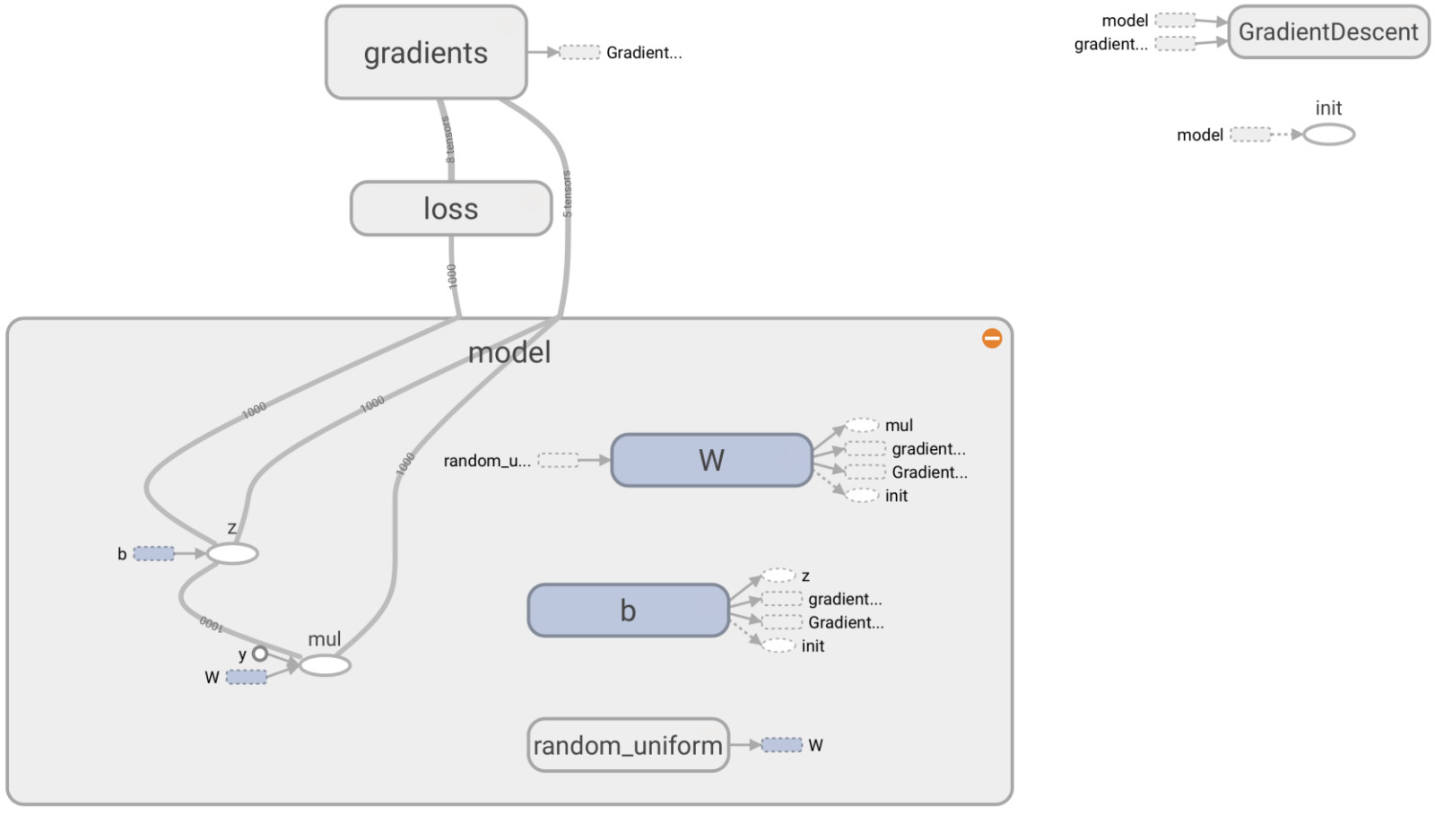
```
[ 2.  4.  6.]
```

Learn Parameters: Optimization

- The Optimizer base class provides methods to compute gradients for a loss and apply gradients to variables.
- A collection of subclasses implement classic optimization algorithms such as GradientDescent and Adagrad.
- TensorFlow provides functions to compute the derivatives for a given TensorFlow computation graph, adding operations to the graph.

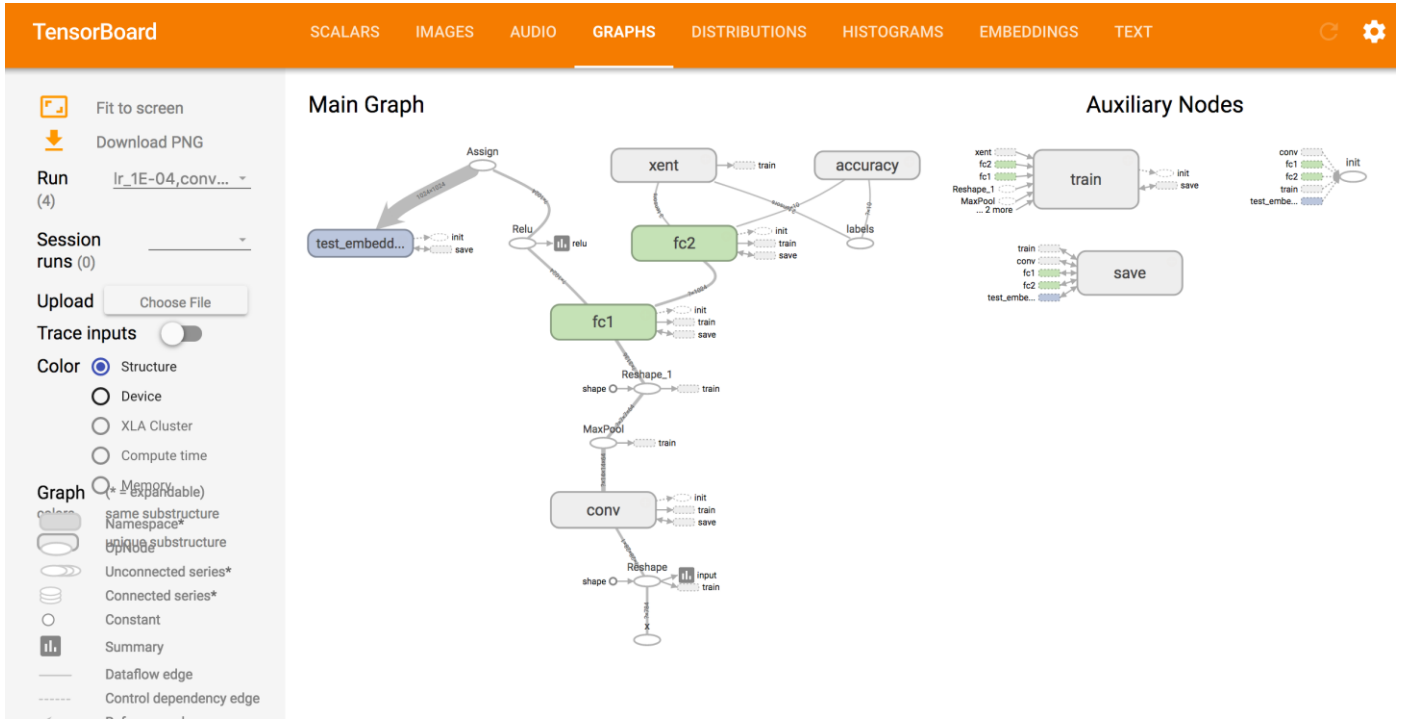


Learn parameters: Optimization



TensorBoard

- Visualize your TensorFlow graph
- Plot quantitative metrics about the execution of your graph
- Show additional data like images that pass through it



TensorFlow Models

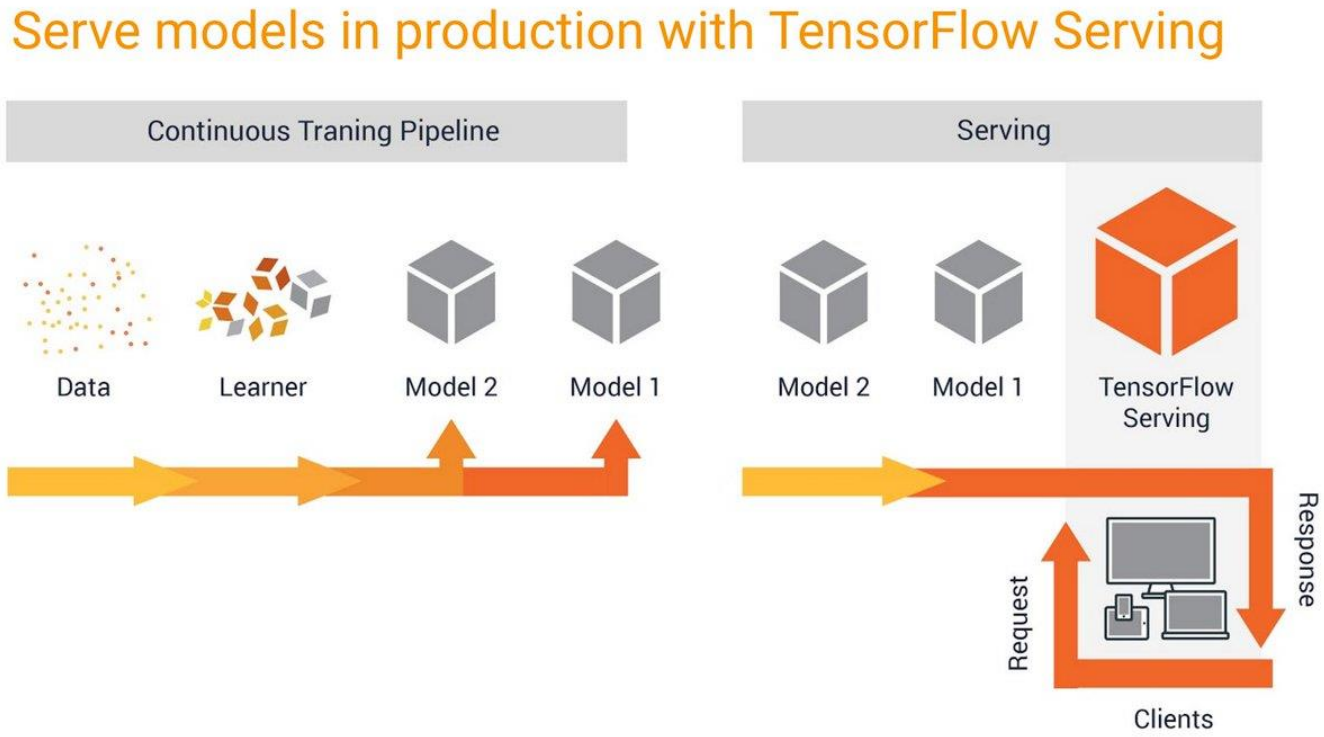
https://github.com/tensorflow/models

Models

- adversarial_crypto: protecting communications with adversarial neural cryptography.
- adversarial_text: semi-supervised sequence learning with adversarial training.
- attention_ocr: a model for real-world image text extraction.
- autoencoder: various autoencoders.
- cognitive_mapping_and_planning: implementation of a spatial memory based mapping and planning architecture for visual navigation.
- compression: compressing and decompressing images using a pre-trained Residual GRU network.
- differential_privacy: privacy-preserving student models from multiple teachers.
- domain_adaptation: domain separation networks.
- im2txt: image-to-text neural network for image captioning.
- inception: deep convolutional networks for computer vision.

TensorFlow Serving

- Flexible, high-performance serving system for machine learning models, designed for production environments.
- Easy to deploy new algorithms and experiments, while keeping the same server architecture and APIs



Code snippets

```
import tensorflow as tf

# build a linear model where y = w * x + b

w = tf.Variable([0.2], tf.float32, name='weight')
b = tf.Variable([0.3], tf.float32, name='bias')

X = tf.placeholder(tf.float32, name="X")
Y = tf.placeholder(tf.float32, name='Y')

# the training values for x and y
x = ([2.,3.,4.,5.])
y = ([-1.,-2.,-3.,-4.])

# define the linear model
linear_model = w*X+b

# define the loss function
square_delta = tf.square(linear_model - Y)
loss = tf.reduce_sum(square_delta)

#set the learning rate and training epoch
learning_rate = 0.01
training_epoch = 1000
```

•

```
# optimizer
optimizer = tf.train.GradientDescentOptimizer(learning_rate)
train = optimizer.minimize(loss)

# start a session
init = tf.global_variables_initializer()

with tf.Session() as sess:
  sess.run(init)

  for i in range(training_epoch):
     sess.run(train, feed_dict={X:x,Y:y})

  # evaluate training accuracy
  curr_w, curr_b, curr_loss  = sess.run([w, b, loss], {X:x,Y:y})
  print('w: %f b: %f loss: %f '%(curr_w, curr_b, curr_loss))
```

Install TensorFlow

Installation

- Determine which TensorFlow to install
- You must choose one of the following types of TensorFlow to install:
  - TensorFlow with CPU support only. If your system does not have a NVIDIA® GPU, you must install this version. Note that this version of TensorFlow is typically much easier to install (typically, in 5 or 10 minutes), so even if you have an NVIDIA GPU, we recommend installing this version first.
  - TensorFlow with GPU support. TensorFlow programs typically run significantly faster on a GPU than on a CPU. Therefore, if your system has a NVIDIA® GPU meeting the prerequisites shown below and you need to run performance-critical applications, you should ultimately install this version.

- `pip3 install --upgrade tensorflow`

Overview: Keras

Keras

- https://keras.io/
- Minimalist, highly modular neural networks library
- Written in Python
- Capable of running on top of either TensorFlow/Theano and CNTK
- Developed with a focus on enabling fast experimentation

Keras: Guiding Principles

- Modularity
  - A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible
- Minimalism
  - Each module should be kept short and simple
- Easy extensibility
  - New modules can be easily added and extended
- Python
  - Supports Python

General Design

- General idea is to based on layers and their input/output
  - Prepare your inputs and output tensors
  - Create first layer to handle input tensor
  - Create output layer to handle targets
  - Build virtually any model you like in between

Keras: Layers

- Keras has a number of pre-built layers. Notable examples include:
  - Regular dense, MLP type
- keras.layers.core.Dense(units, activation=**None**, use_bias=**True**, kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=**None**, bias_regularizer=**None**, activity_regularizer=**None**, kernel_constraint=**None**, bias_constraint=**None**)
  - Recurrent layers, LTSM, GRU, etc
- keras.layers.recurrent.Recurrent(return_sequences=**False**, return_state=**False**, go_backwards=**False**, stateful=**False**, unroll=**False**, implementation=0)
-

Keras: :Layers

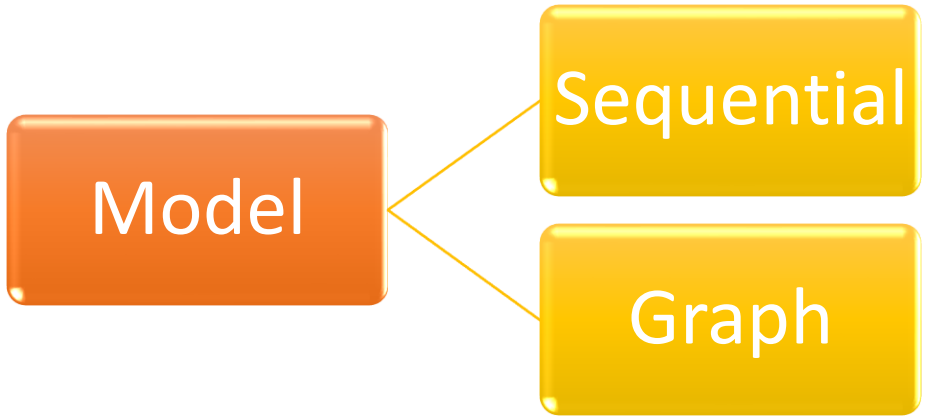
- 1D Convolutional layers
  - keras.layers.convolutional.Conv1D(filters, kernel_size, strides=1, padding='valid', dilation_rate=1, activation=None, use_bias=True, kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, bias_constraint=None)
- 2D Convolutional layers
  - keras.layers.convolutional.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', data_format=None, dilation_rate=(1, 1), activation=None, use_bias=True, kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, bias_constraint=None)

Why use keras?

- Easy and fast prototyping (through total modularity, minimalism, and extensibility).
- Supports both convolutional networks and recurrent networks and combinations of the two.
- Supports arbitrary connectivity schemes (including multi-input and multi-output training).
- Runs seamlessly on CPU and GPU.

Keras Code examples

- The core data structure of Keras is a **model**
- Model → a way to organize layers



Keras: Code Snippets

```
import numpy as np
from keras.models import Sequential
from keras.layers.core import Activation, Dense
from keras.optimizers import SGD

X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])

model = Sequential()
model.add(Dense(2, input_dim=2, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))

sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)

history = model.fit(X, y, nb_epoch=10000, batch_size=4)

print(model.predict(X))
```

Keras: Installation

- **Theano :**
  - pip3 install Theano
- **CNTK**
  - $ export CNTK_BINARY_URL=... [choose link here]
  - pip3 install $CNTK_BINARY_URL
- **TensorFlow :**
  - $ export TF_BINARY_URL=... [choose link here]
  - pip3 install $TF_BINARY_URL
- **Keras :**
  - pip3 install keras
- **To enable GPU computing :**
  - NVidia Cuda
  - CuDNN
  - CNMeM

Slides and codes

Slides and codes will be available from github:

http://bit.ly/googledevfest-penang