Deep Learning

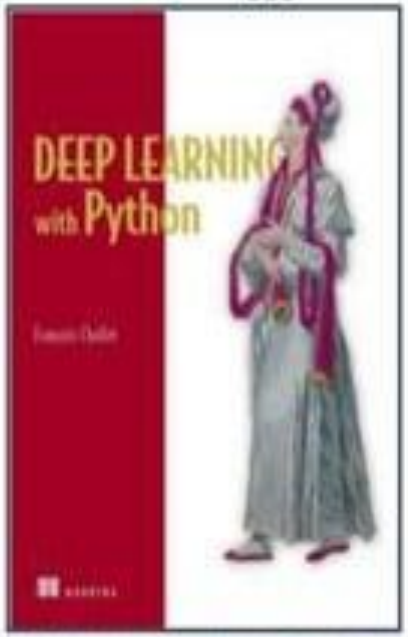
Lecture 1
Introduction to deep learning

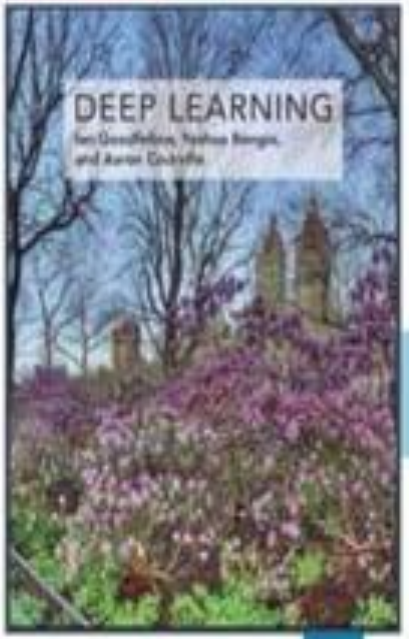
Asst. Lect. Ali Al-khawaja

Class Room

Further resources

- This course is largely "inspired by": "Deep Learning with Python" by François Chollet
- Recommended textbook: "Deep learning" by Goodfellow, Bengio, Courville
- Lots of further material available online, e.g.:
  http://cs231n.stanford.edu/ http://course.fast.ai/
  https://developers.google.com/machine-learning/crash-course/
  www.nvidia.com/dlilabs http://introtodeeplearning.com/
  https://github.com/oxford-cs-deepnlp-2017/lectures,
  https://jalammar.github.io/
- Academic courses





What is artificial intelligence?

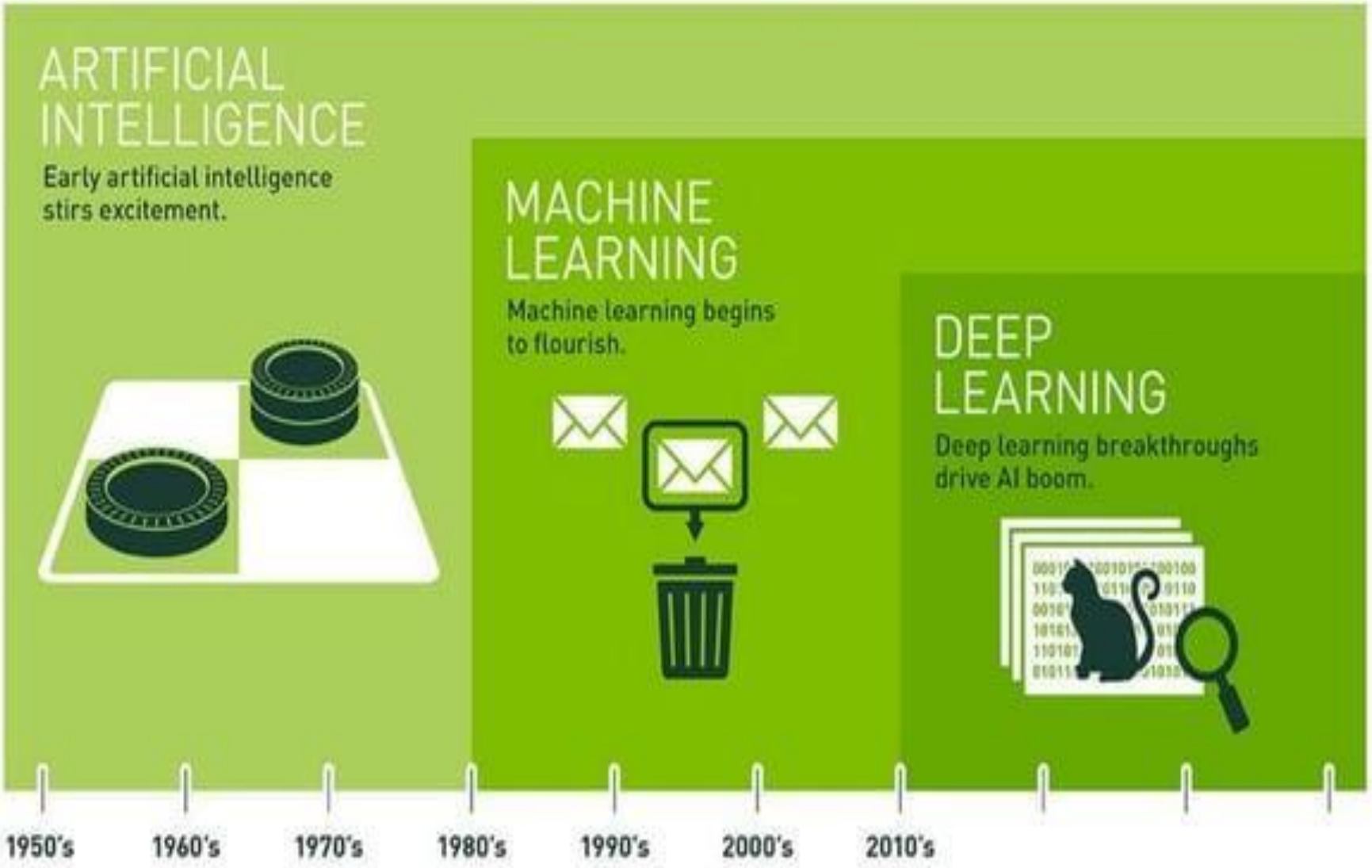
Artificial intelligence is the ability of a computer to perform tasks commonly associated with intelligent beings.

What is machine learning?

Machine learning is the study of algorithms that learn from examples and experience instead of relying on hard-coded rules and make predictions on new data.

What is deep learning?

Deep learning is a subfield of machine learning focusing on learning data representations as successive layers of increasingly meaningful representations.



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.

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

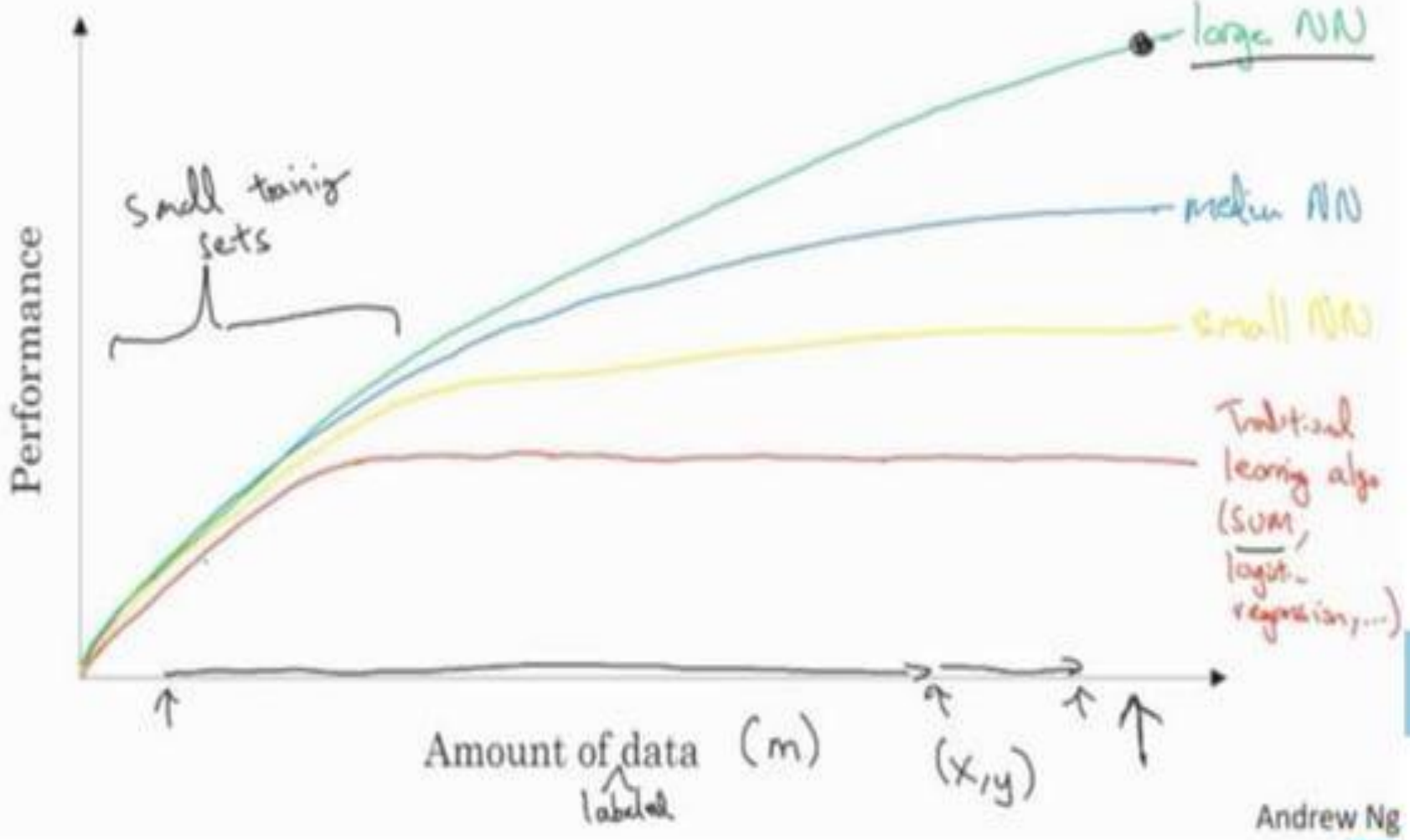
"Traditional" machine learning:

handcrafted features → learned classifier → cat

Deep, "end-to-end" learning:

learned low-level features → learned mid-level features → learned high-level features → learned classifier → cat

Scale drives deep learning progress



Andrew Ng

Table 1: Major milestones that will be covered in this paper

Year	Contributer	Contribution
300 BC	Aristotle	introduced Associationism, started the history of human's attempt to understand brain.
1873	Alexander Bain	introduced Neural Groupings as the earliest models of neural network, inspired Hebbian Learning Rule.
1943	McCulloch & Pitts	introduced MCP Model, which is considered as the ancestor of Artificial Neural Model.
1949	Donald Hebb	considered as the father of neural networks, introduced Hebbian Learning Rule, which lays the foundation of modern neural network.
1958	Frank Rosenblatt	introduced the first perceptron, which highly resembles modern perceptron.
1974	Paul Werbos	introduced Backpropagation
1980	Teuvo Kohonen	introduced Self Organizing Map
	Kunihiko Fukushima	introduced Neocogitron, which inspired Convolutional Neural Network
1982	John Hopfield	introduced Hopfield Network
1985	Hilton & Sejnowski	introduced Boltzmann Machine
1986	Paul Smolensky	introduced Harmonium, which is later known as Restricted Boltzmann Machine
	Michael I. Jordan	defined and introduced Recurrent Neural Network
1990	Yann LeCun	introduced LeNet, showed the possibility of deep neural networks in practice
1997	Schuster & Paliwal	introduced Bidirectional Recurrent Neural Network
	Hochreiter & Schmidhuber	introduced LSTM, solved the problem of vanishing gradient in recurrent neural networks
2006	Geoffrey Hinton	introduced Deep Belief Networks, also introduced layer-wise pretraining technique, opened current deep learning era.
2009	Salakhutdinov & Hinton	introduced Deep Boltzmann Machines
2012	Geoffrey Hinton	introduced Dropout, an efficient way of training neural networks

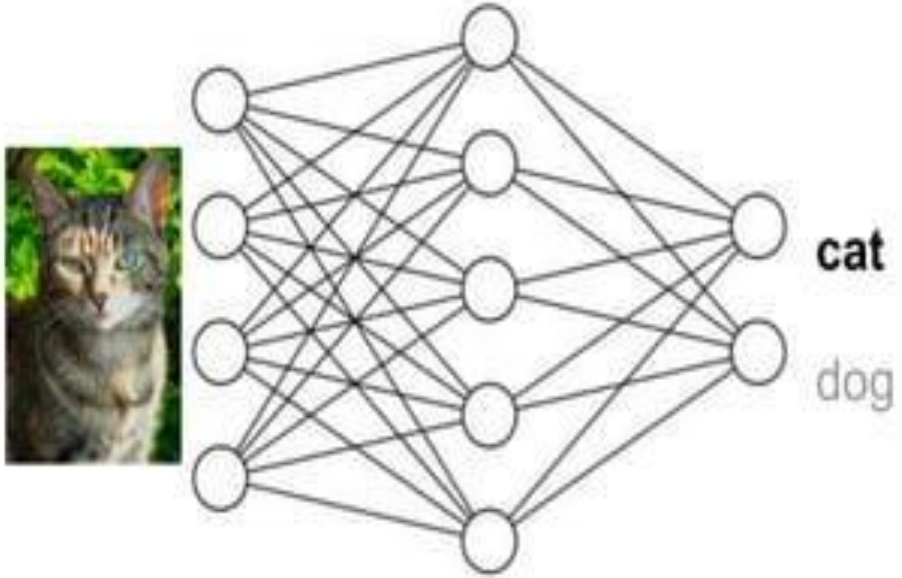
From: Wang & Raj: On the Origin of Deep Learning (2017)

Main types of machine learning

Main types of machine learning

- **Supervised learning**

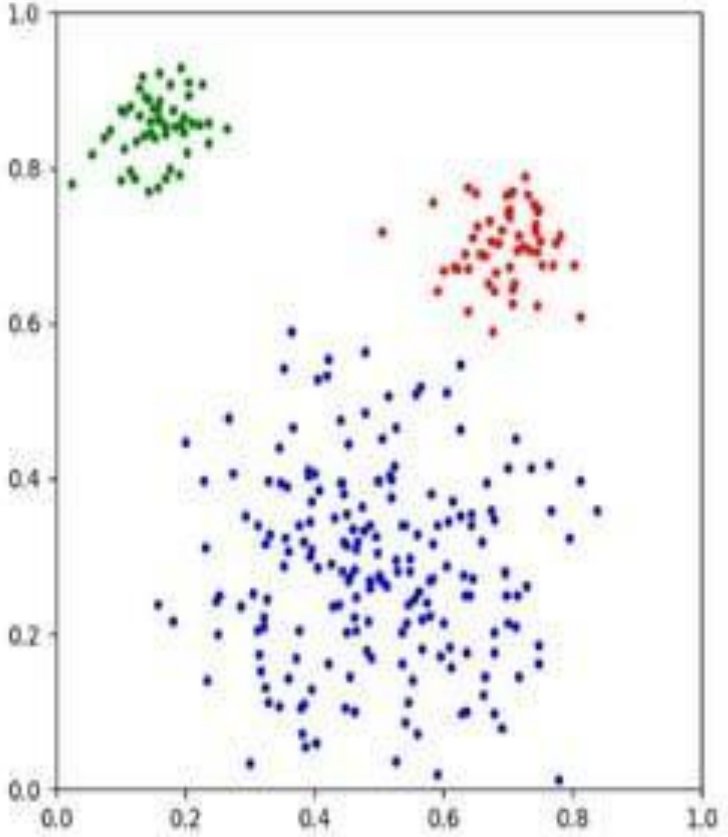
- Unsupervised learning
- Self-supervised learning
- Reinforcement learning



Main types of machine learning

- Supervised learning

- **Unsupervised learning**
- Self-supervised learning
- Reinforcement learning



Main types of machine learning

- Supervised learning

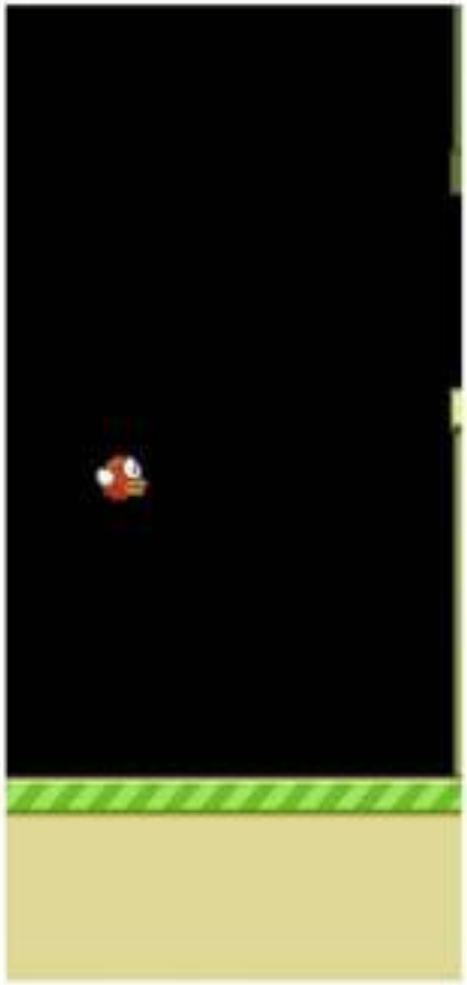
- Unsupervised learning
- **Self-supervised learning**
- Reinforcement learning

Image from https://arxiv.org/abs/1710.10196

Main types of machine learning

- Supervised learning

- Unsupervised learning
- Self-supervised learning
- **Reinforcement learning**

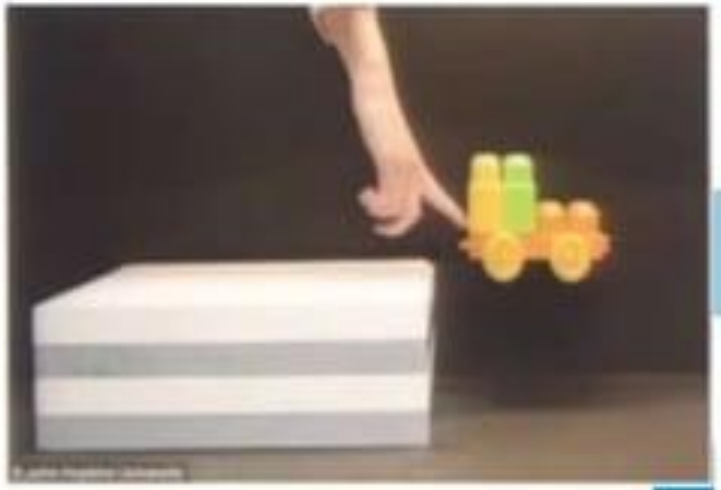


Animation from https://yanpanlau.github.io/2016/07/10/FlappyBird-Keras.html

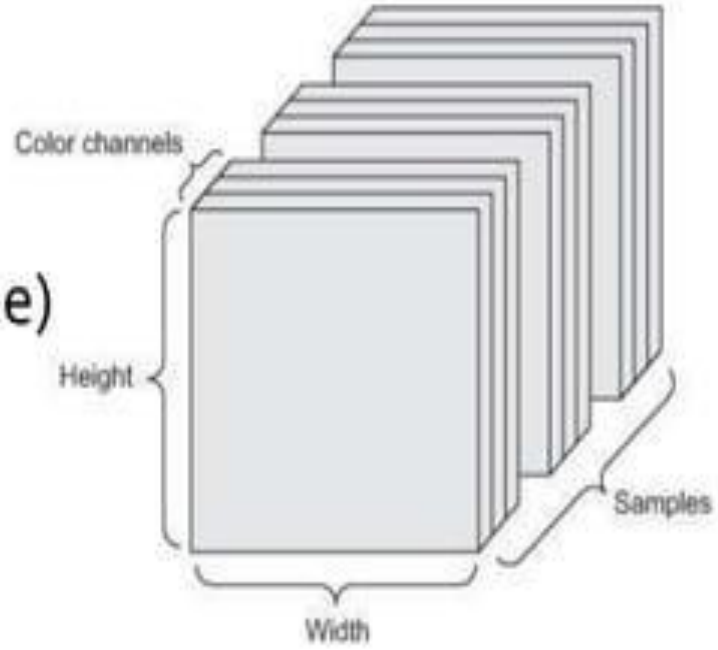
Fundamentals of machine learning

Data

- Humans learn by observation and unsupervised learning
  - model of the world / common sense reasoning
- Machine learning needs lots of (labeled) data to compensate



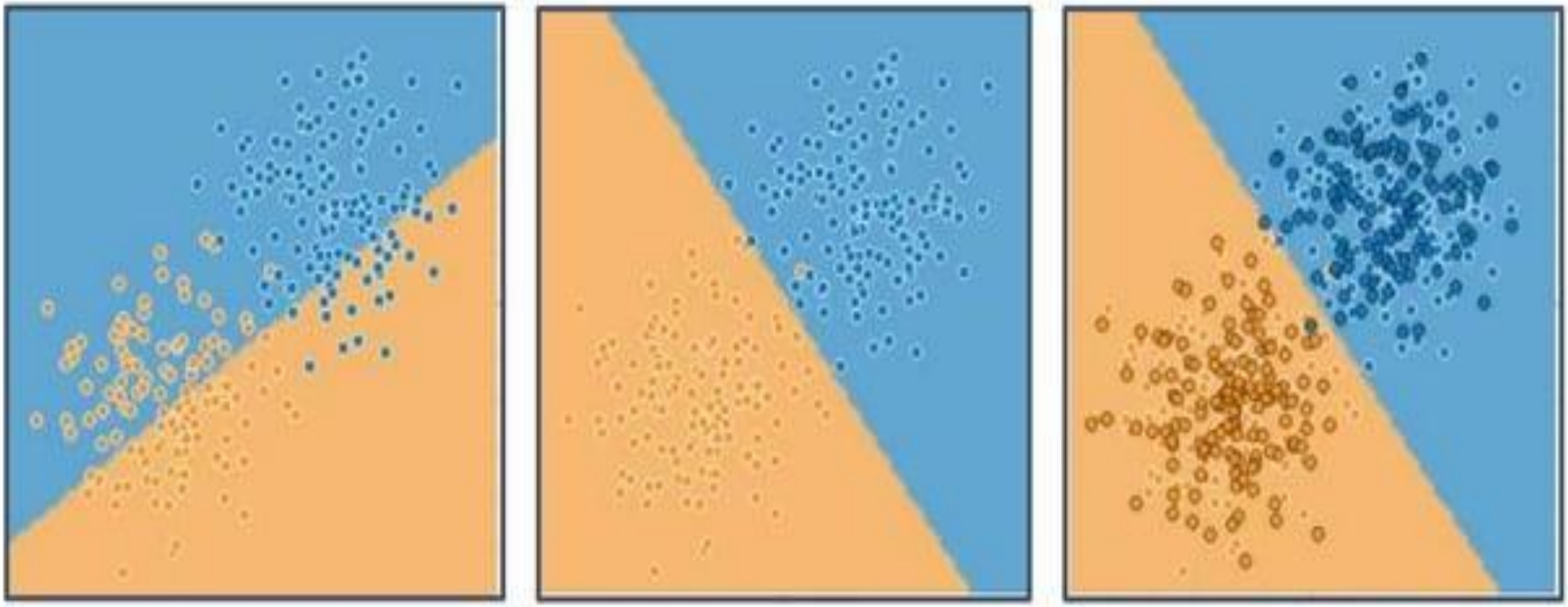
Data



- Tensors: generalization of matrices to *n* dimensions (or rank, order, degree)
  - 1D tensor: vector
  - 2D tensor: matrix
  - 3D, 4D, 5D tensors
  - `numpy.ndarray(shape, dtype)`
- Training – validation – test split (+ adversarial test)
- Minibatches
  - small sets of input data used at a time
  - usually processed independently

Image from: https://arxiv.org/abs/1707.08945

Model – learning/training – inference



http://playground.tensorflow.org/

$$\hat{y} = f(\mathbf{x}; \theta)$$

- parameters θ and hyperparameters

Optimization

- Mathematical optimization:
  "the selection of a best element (with regard to some criterion) from some set of available alternatives" (Wikipedia)
- Main types:
  finite-step, iterative, heuristic
- Learning as an optimization problem
  - cost function:

By Rebecca Wilson (originally posted to Flickr as Vicariously) [CC BY 2.0], via Wikimedia Commons

loss

regularization

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} L(f(\mathbf{x}_i;\theta), y_i) + R(\theta)$$

Optimization

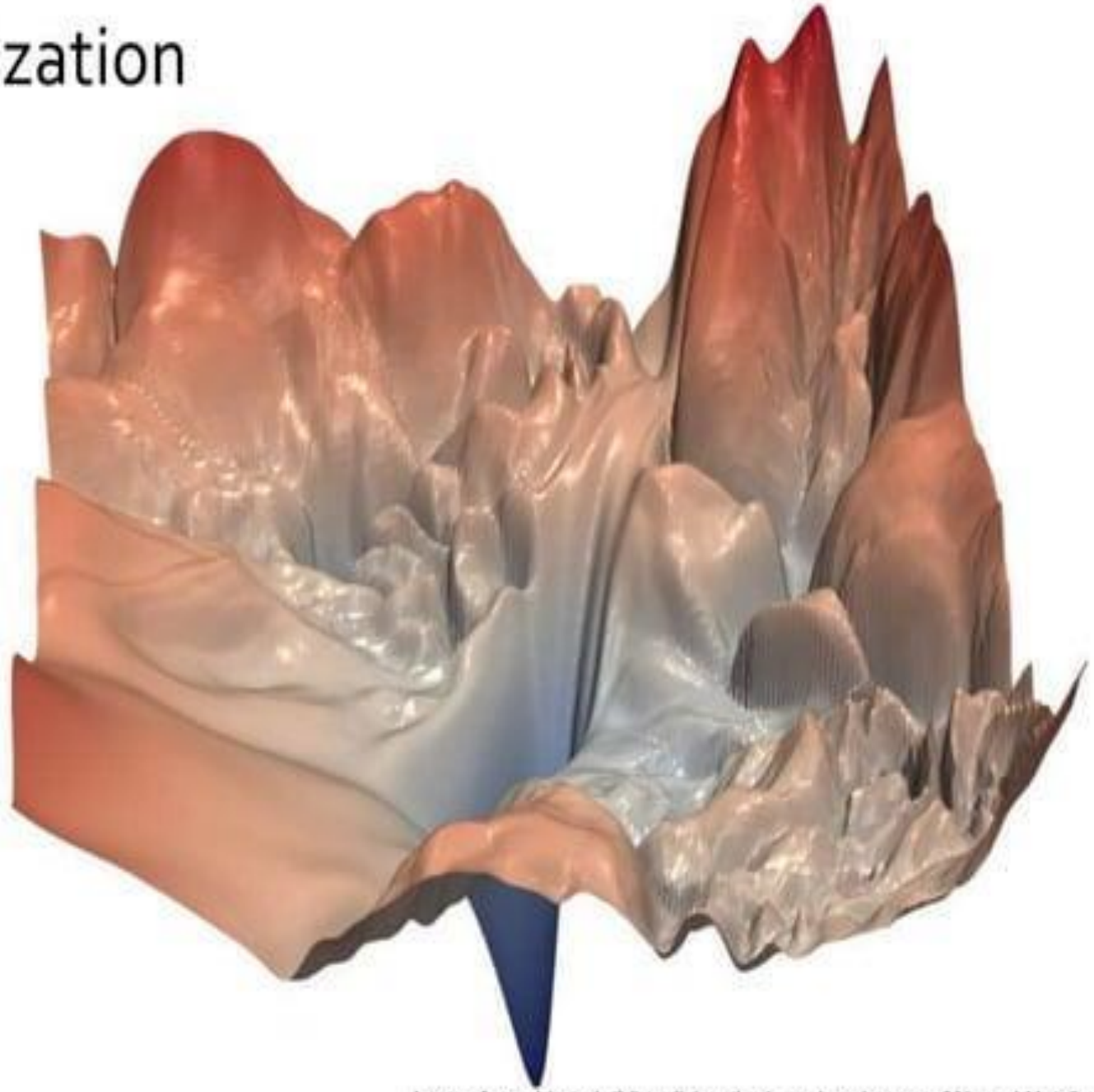


Image from: Li et al. "Visualizing the Loss Landscape of Neural Nets", arXiv:1712.09913

Gradient descent

- Derivative and minima/maxima of functions
- Gradient: the derivative of a multivariable function
- Gradient descent:

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial J(\theta)}{\partial \theta}$$

- (Mini-batch) stochastic gradient descent (and its variants)

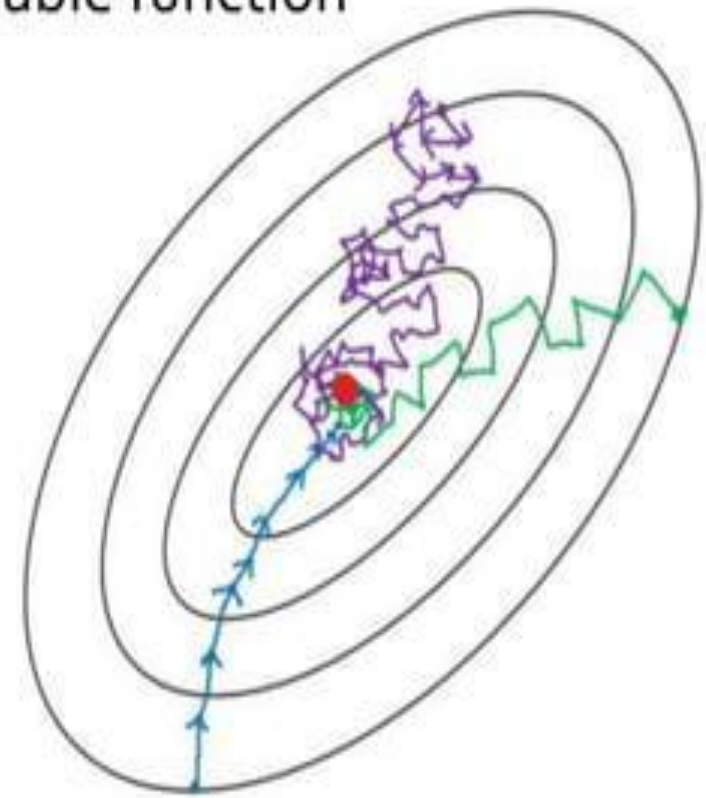
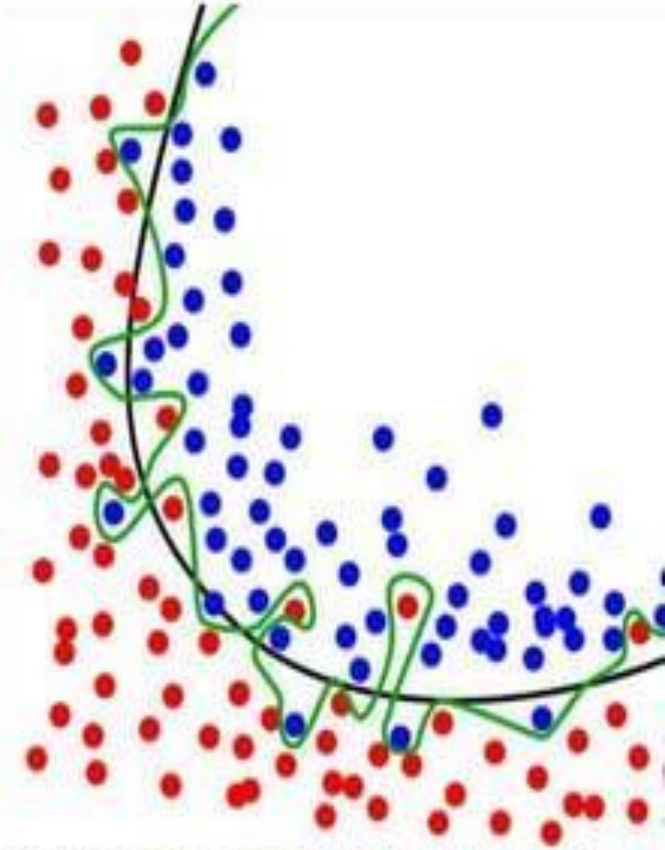


Image from: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Over- and underfitting, generalization, regularization

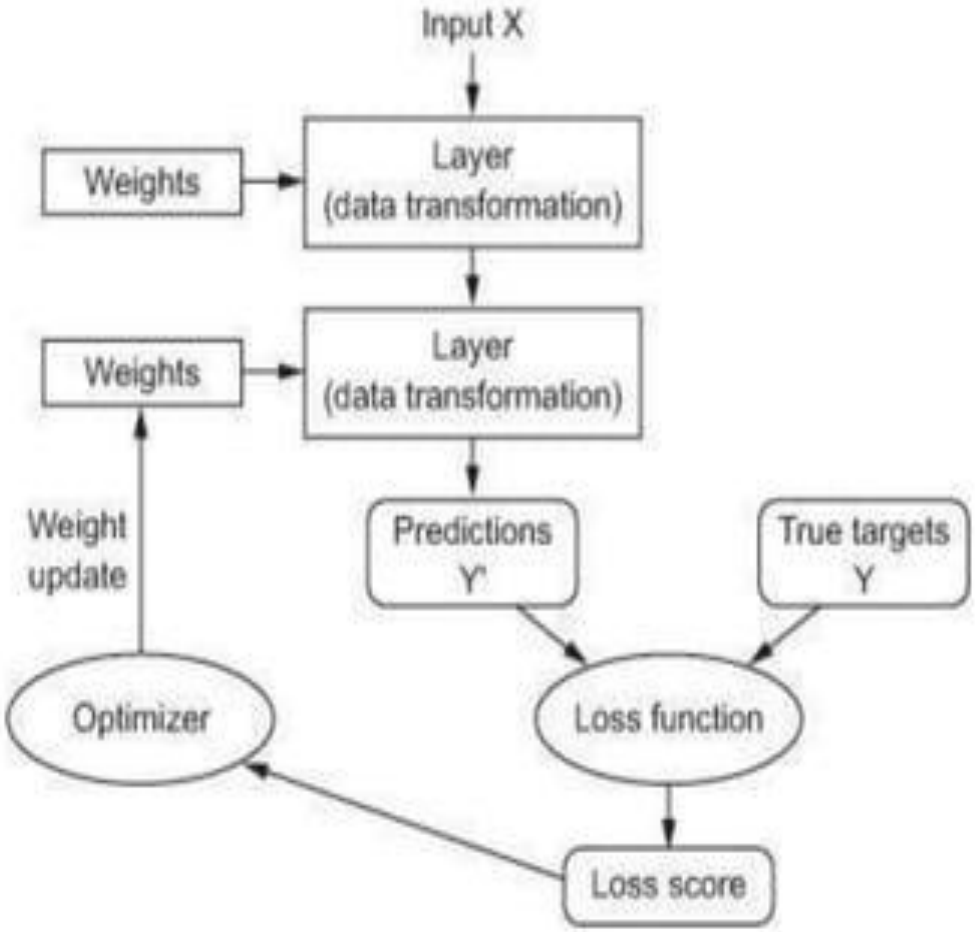
- Models with lots of parameters can easily overfit to training data
- **Generalization:** the quality of ML model is measured on new, unseen samples
- **Regularization:** any method* to prevent overfitting
  - simplicity, sparsity, dropout, early stopping
  - *) other than adding more data



Deep learning

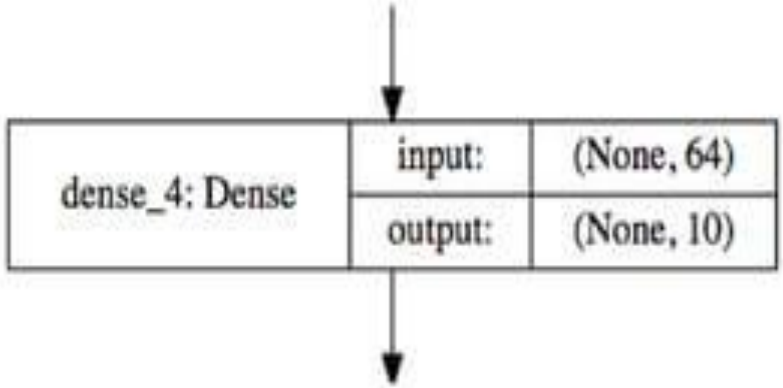
Anatomy of a deep neural network

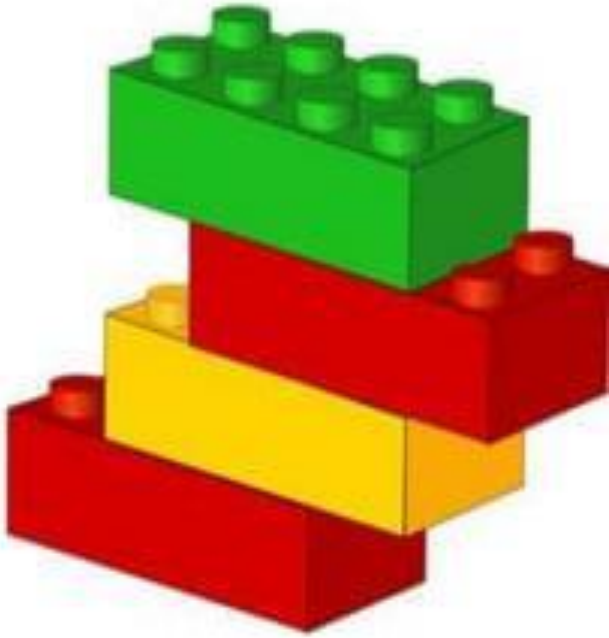
- Layers
- Input data and targets
- Loss function
- Optimizer



Layers

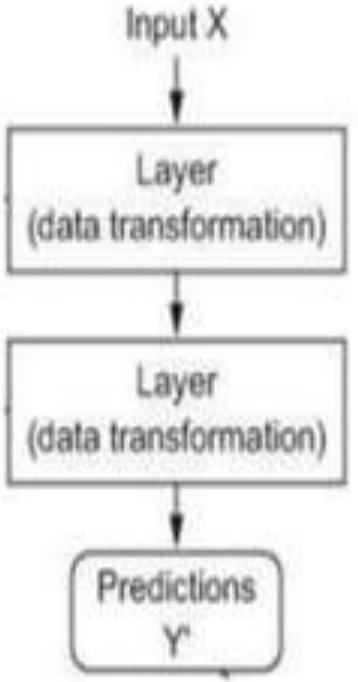
- Data processing modules
- Many different kinds exist
  - densely connected
  - convolutional
  - recurrent
  - pooling, flattening, merging, normalization, etc.
- Input: one or more tensors
  output: one or more tensors
- Usually have a state, encoded as **weights**
  - learned, initially random
- When combined, form a **network** or a **model**

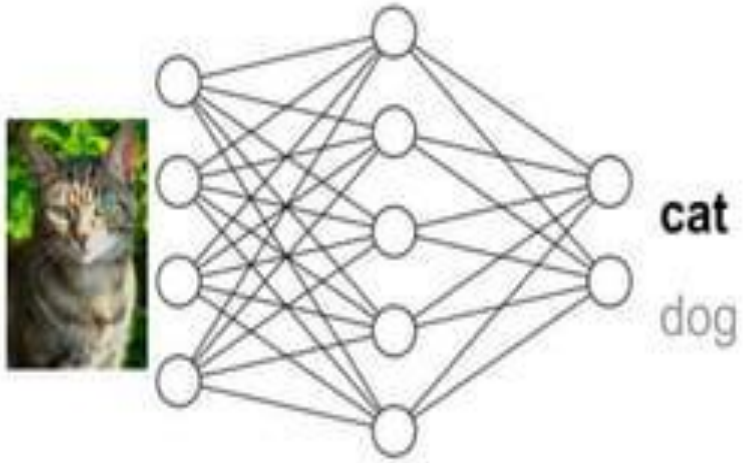




Input data and targets

- The network maps the input data X to predictions Y'
- During training, the predictions Y' are compared to true targets Y using the loss function



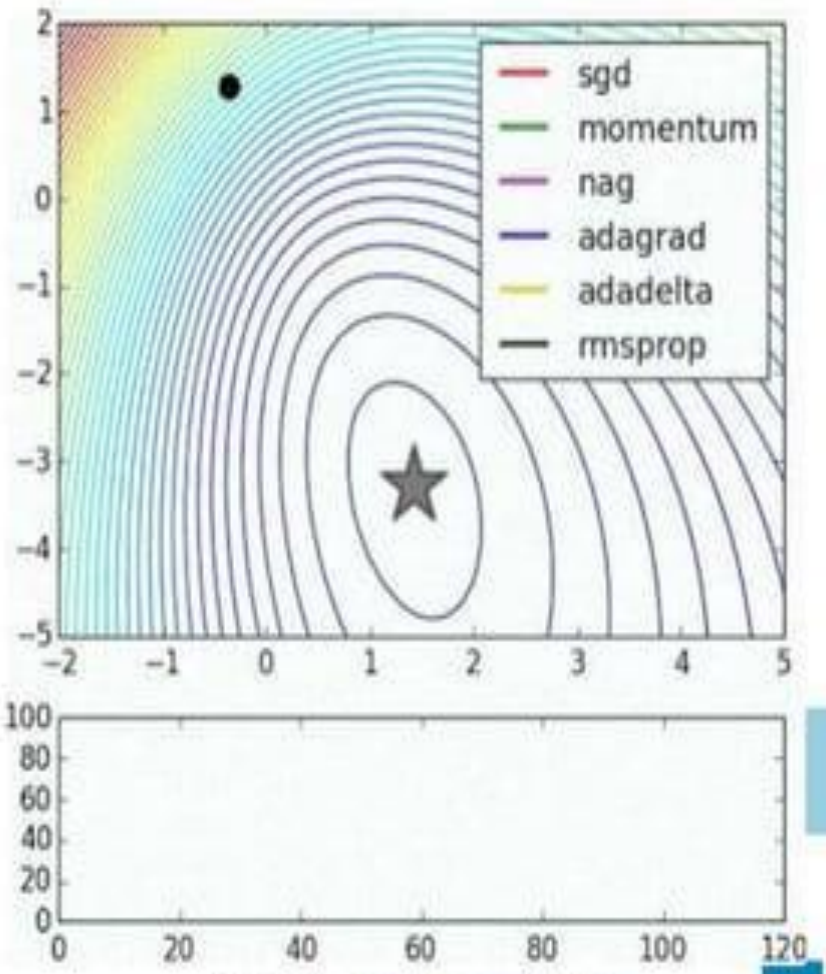


Loss function

- The quantity to be minimized (optimized) during training
  - the only thing **the network** cares about
  - there might also be other metrics **you** care about
- Common tasks have "standard" loss functions:
  - *mean squared error* for regression
  - *binary cross-entropy* for two-class classification
  - *categorical cross-entropy* for multi-class classification
  - etc.
- https://lossfunctions.tumblr.com/

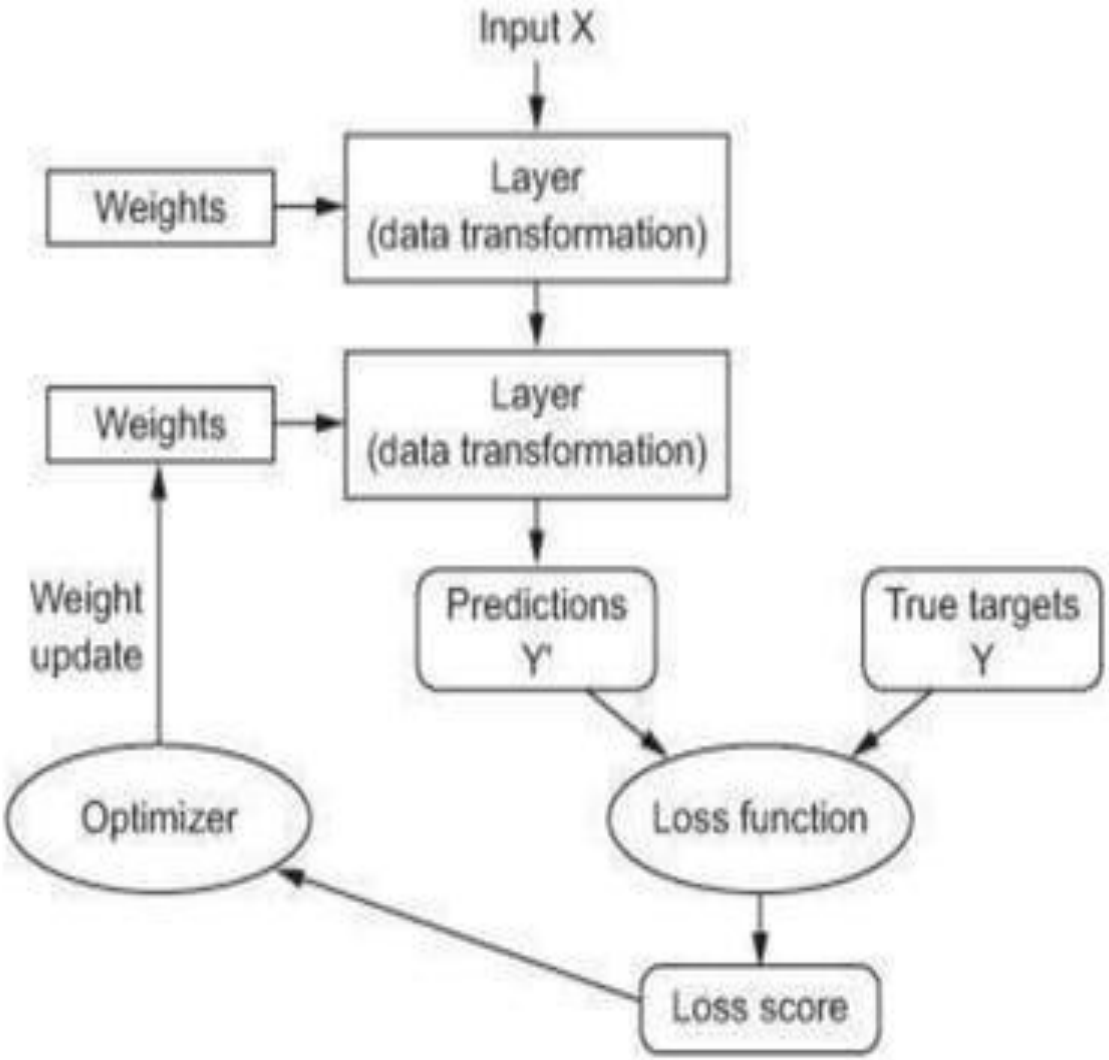
Optimizer

- How to update the weights based on the loss function
- *Learning rate (+scheduling)*
- Stochastic gradient descent, momentum, and their variants
  - RMSProp is usually a good first choice
  - more info: http://ruder.io/optimizing-gradient-descent/



Animation from: https://imgur.com/a/Hqolp

Anatomy of a deep neural network

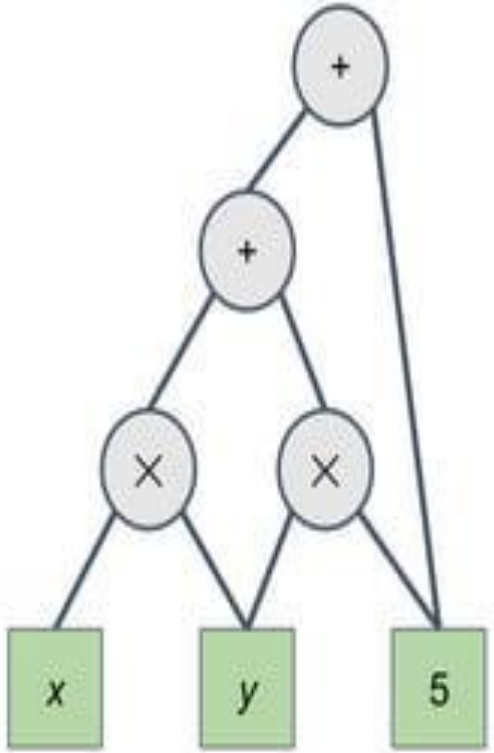


Deep learning frameworks


Caffe
torch
Chainer
PyTorch
DEEPLEARNING4J
Caffe2
The Microsoft Cognitive Toolkit
TensorFlow
K
dmlc
mxnet
theano

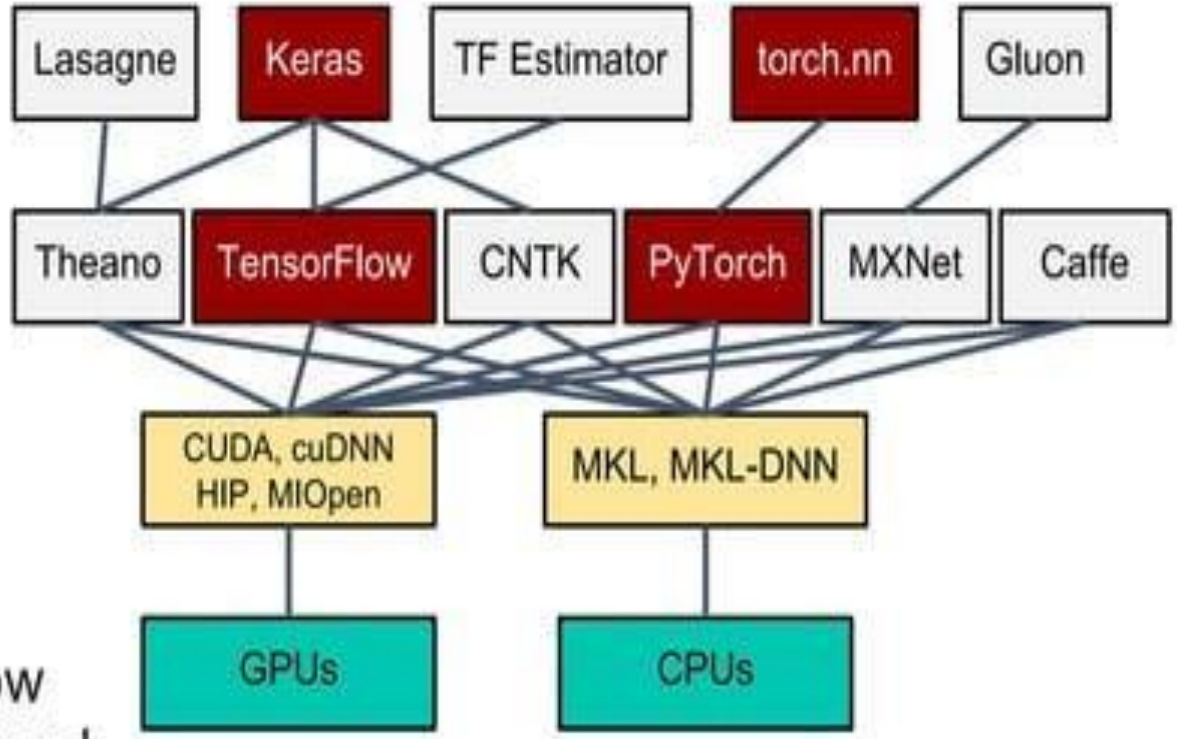
Deep learning frameworks

- Actually tools for defining **static** or **dynamic** general-purpose **computational graphs**
- Automatic differentiation
- Seamless CPU / GPU usage
  - multi-GPU, distributed
- Python/numpy or R interfaces
  - instead of C, C++, CUDA or HIP
- Open source



$$xy + 5y + 5$$

Deep learning frameworks



- Keras is a high-level neural networks API
  - we will use TensorFlow as the compute backend
  - included in TensorFlow 2 as `tf.keras`
  - https://keras.io/ , https://www.tensorflow.org/guide/keras
- PyTorch is:
  - a GPU-based tensor library
  - an efficient library for dynamic neural networks
  - https://pytorch.org/

Thank you…

Any questions??

يرجى مسح رمز الاستجابة السريعة QR Code لتعبئة نموذج التغذية الراجعة حول المحاضرة. ملاحظاتكم مهمة لتحسين المحاضرات القادمة.

My google site