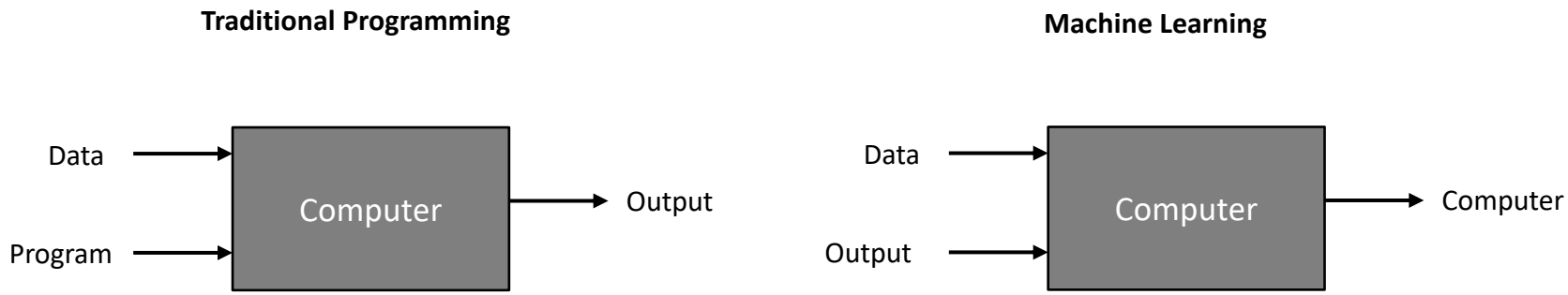
Machine Learning.

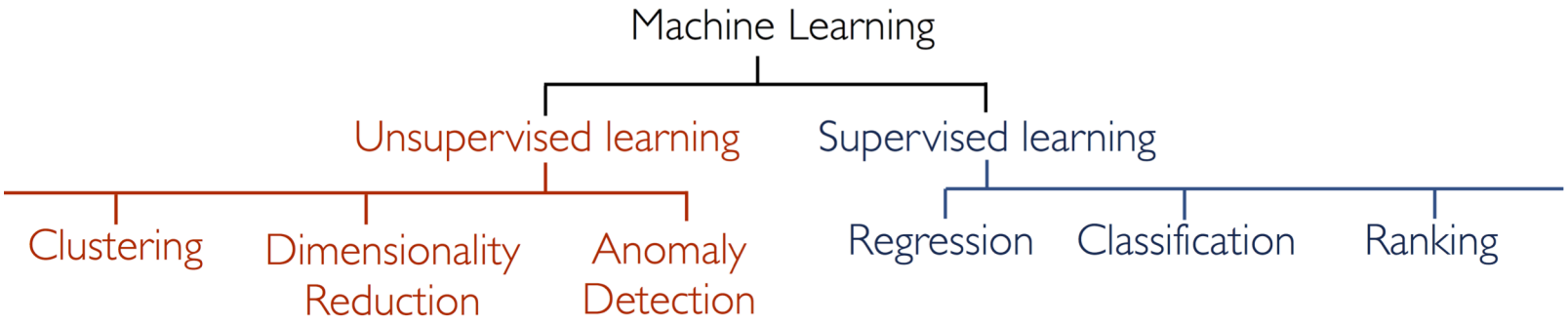
Machine Learning (1)

- Machine learning is a set of algorithms that can take a set of inputs (data) and return a prediction.



Machine Learning (2)

- **Supervised:**
  - **Supervised learning:** using a collection of predictors and some observed outcomes to build an algorithm to predict the outcome when it is not observed; random forests, boosting, SVMs
  - A human assigns a topic label to each document in a corpus. The algorithm learns how to predict the label.
  - **Classification:** the learned attribute is categorical
  - **Regression:** the learned attribute is numeric
- **Unsupervised**
  - **Unsupervised learning:** trying to uncover unobserved factors; clustering, mixture models, principal components
  - Input is given (without label). ML discovers groups and patterns


Machine Learning
Unsupervised learning
Supervised learning
Clustering
Dimensionality Reduction
Anomaly Detection
Regression
Classification
Ranking

- **Representation.**
  - How to represent the input, i.e., what features to use.
  - A classifier must be represented in some formal language that the computer can handle.
- **Loss** is the penalty for a bad prediction. That is, loss is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater.
- **Evaluation.** An evaluation function (also called objective function or scoring function) is needed to distinguish good classifiers from bad ones.

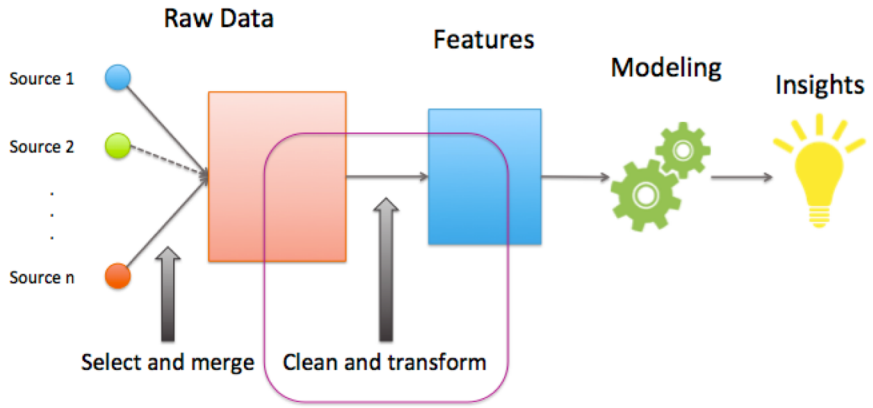
Machine Learning/ Modeling Terms

- **Training** a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss;
- **Optimization.** Finally, we need a method to search among the classifiers in the language for the highest-scoring one. The choice of optimization technique is key to the efficiency of the learner, and also helps determine the classifier produced if the evaluation function has more than one optimum. It is common for new learners to start out using off-the-shelf optimizers, which are later replaced by custom-designed ones.

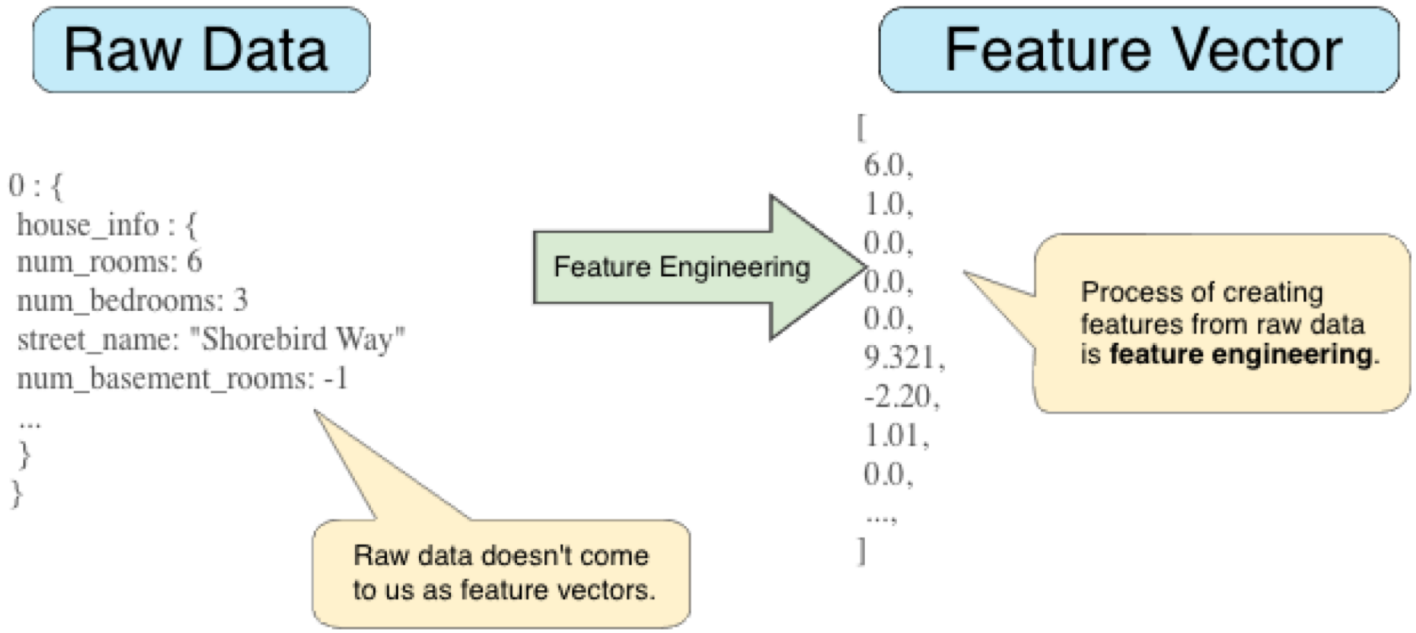
https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss

Representation and Feature Engineering

- Machine Learning require numeric data as input
- A *feature* is a numeric representation of an aspect of raw data.
- Features sit between data and models in the machine learning pipeline.
- *Feature engineering* is the act of extracting features from raw data, and transforming them into formats that is suitable for the machine learning model

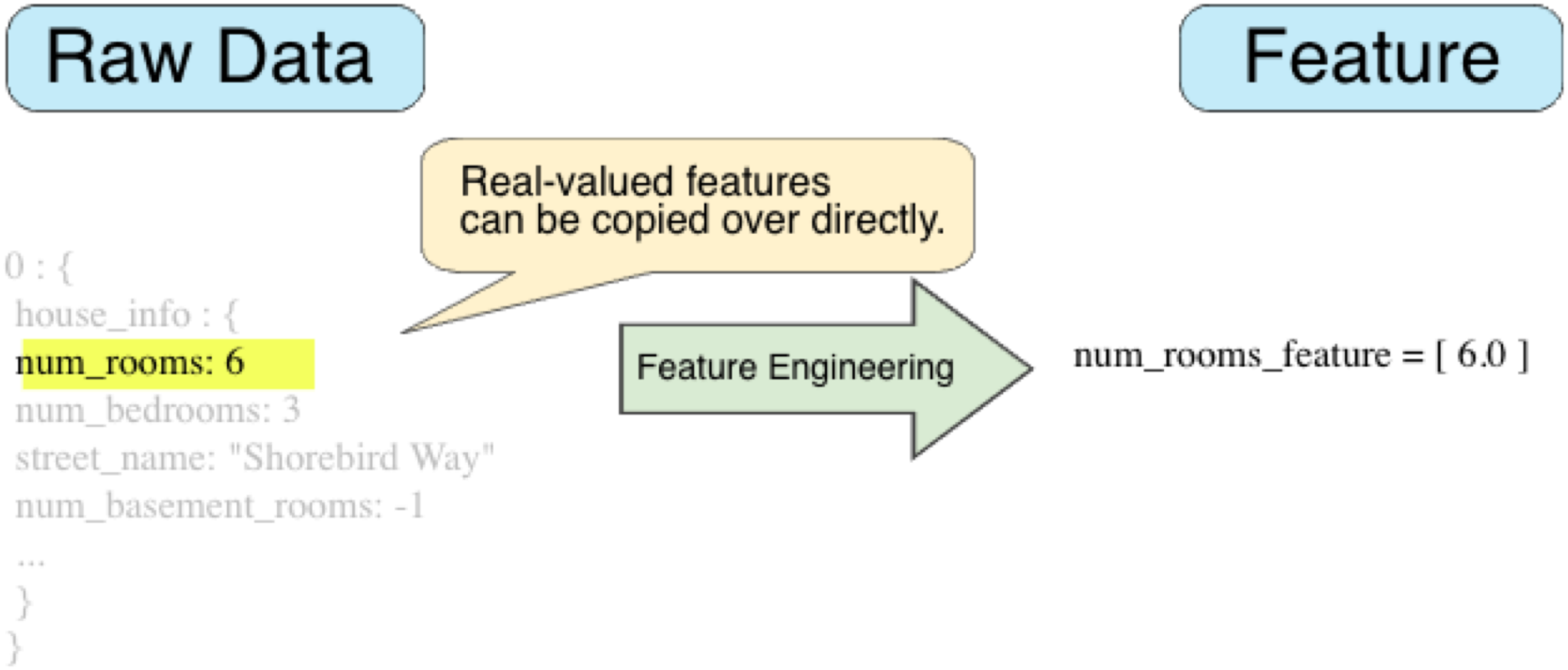


Feature Engineering



https://developers.google.com/machine-learning/crash-course/representation/feature-engineering

Feature Engineering



https://developers.google.com/machine-learning/crash-course/representation/feature-engineering

Feature Engineering

- Binning:
  - Raw measures/counts are often not a robust enough measure
  - Types of Binning: Fixed, quantile
- Feature Standardization:
  - Min-max scaling
  - Normalization
- Unstructured Data:
  - Images: HOG (Histogram of Oriented Gradients), DL
  - Test: Bag-of-Words, TF-IDF, Word Embeddings

Regression Models

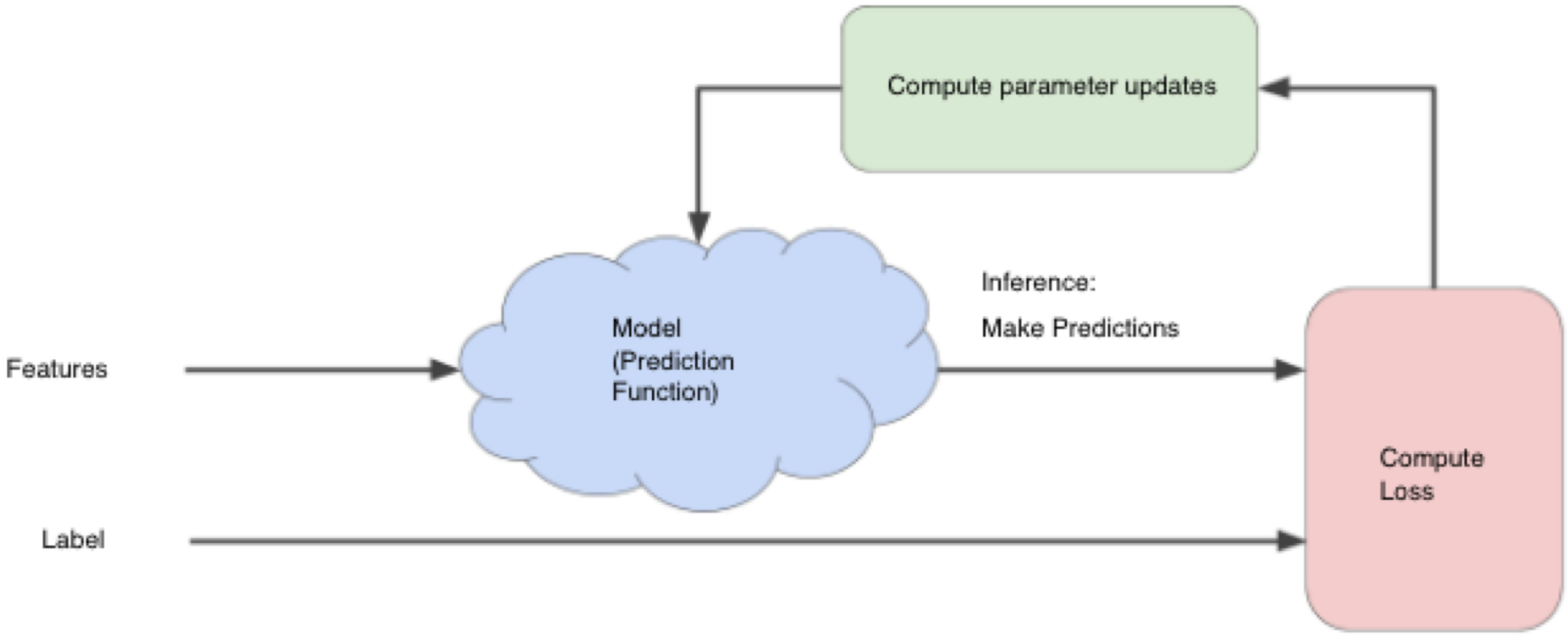
- Regression models are used for continuous target variables
- Assumes linear relationship between predictor variable and outcome
- Easy to use
- Formula:

$$Y = \beta_0 + \beta_1 * X + \epsilon$$

Outcome — Intercept — slope — Predictor Variable / Feature — Error

- β_0 and β_1 are parameters, weights or coefficients
- Objective: Minimize error, i.e. the difference between predictions and obsverations

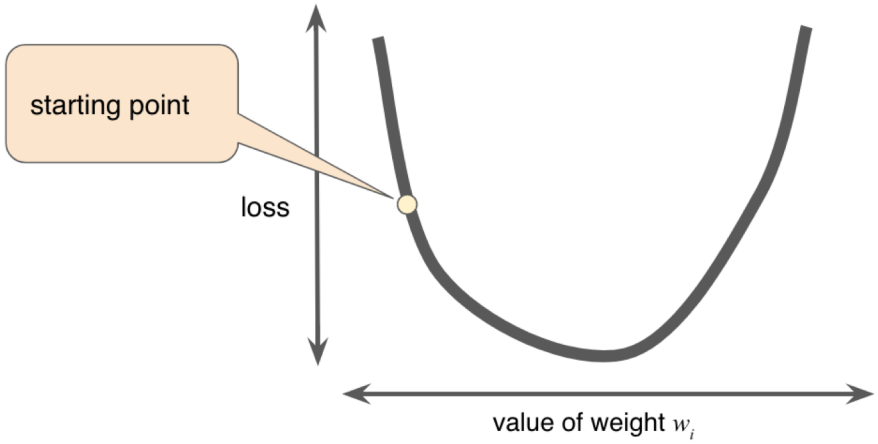
Loss Reduction

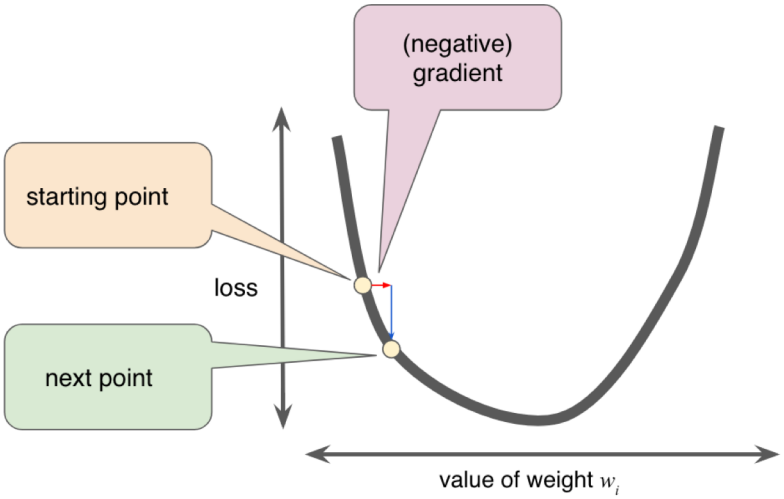


https://developers.google.com/machine-learning/crash-course/reducing-loss/an-iterative-approach

Gradient Decent

- Calculating the loss function for every conceivable value β_0 and β_1 inefficient
- Gradient Decent is a common optimization approach used to compute parameters





- Learning rate determines the size of step
- Mini-batch Stochastic Gradient Decent: Only a sample of examples used to compute gradient in a single iteration

https://developers.google.com/machine-learning/crash-course/reducing-loss/an-iterative-approach

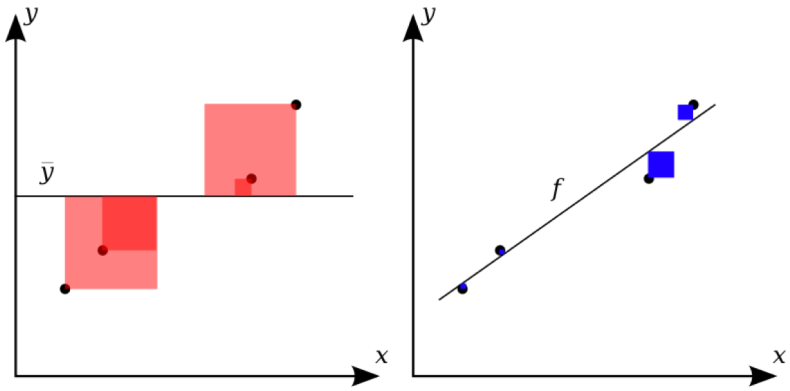
Model Quality

- Mean Square Error: measures the average of the squares of the errors or deviations

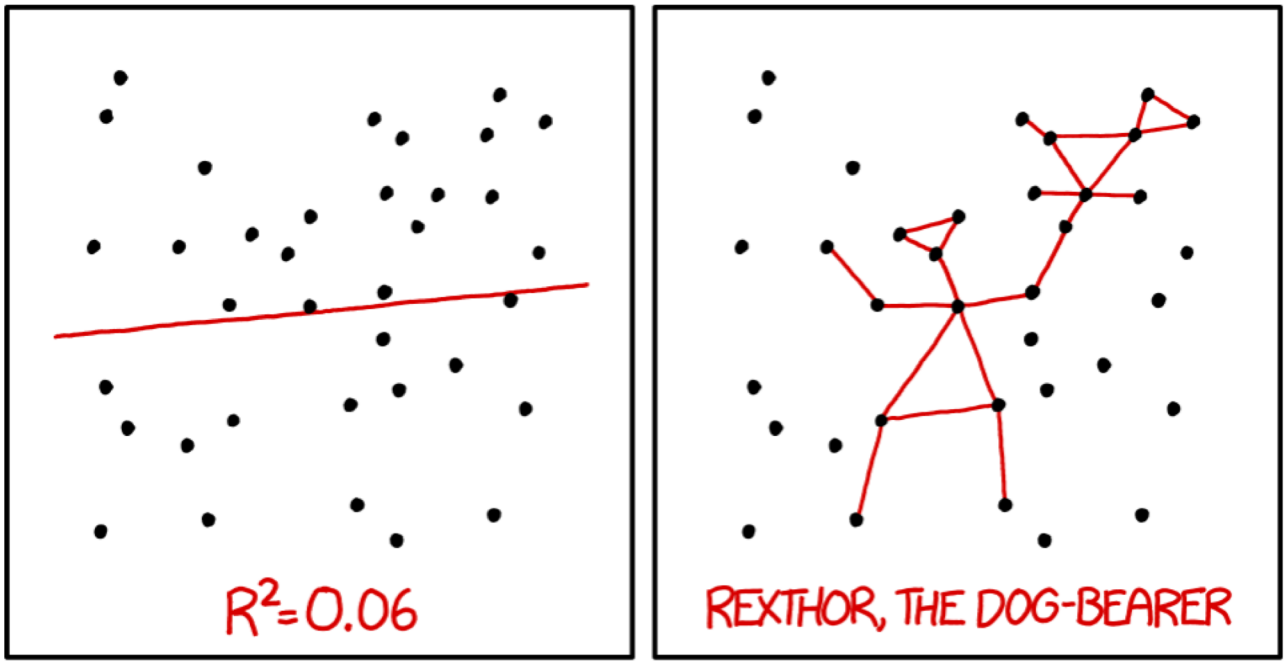
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

- $R^{2:}$: Coefficient of determination:
  - the proportion of total variation of outcomes explained by the model
  - Explained variation / Total variation
  - $R^2 = 1$: regression line fits perfectly
  - $R^2 = 0$: regression line does not fit at all

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$



Linear Regressions

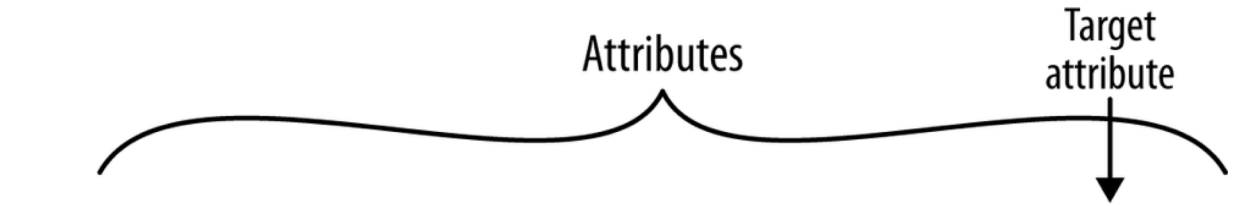


I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER TO GUESS THE DIRECTION OF THE CORRELATION FROM THE SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.

https://xkcd.com/1725/

Supervised Learning: Classification

- Predict that an item belongs to a certain class



Name	Balance	Age	Employed	Write-off
Mike	$200,000	42	no	yes
Mary	$35,000	33	yes	no
Claudio	$115,000	40	no	no
Robert	$29,000	23	yes	yes
Dora	$72,000	31	no	no

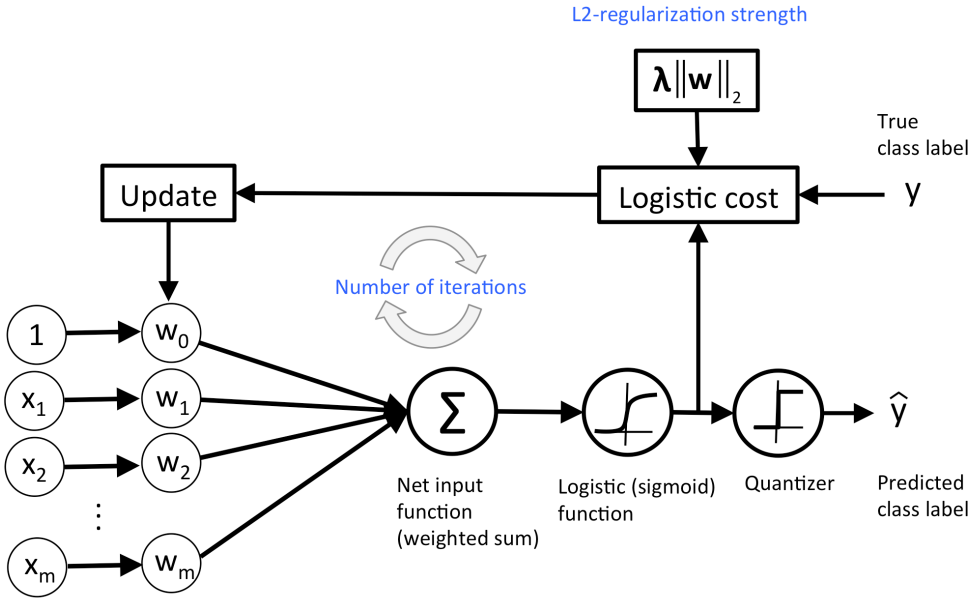
This is one row (example).
Feature vector is: **<Claudio,115000,40,no>**
Class label (value of Target attribute) is **no**

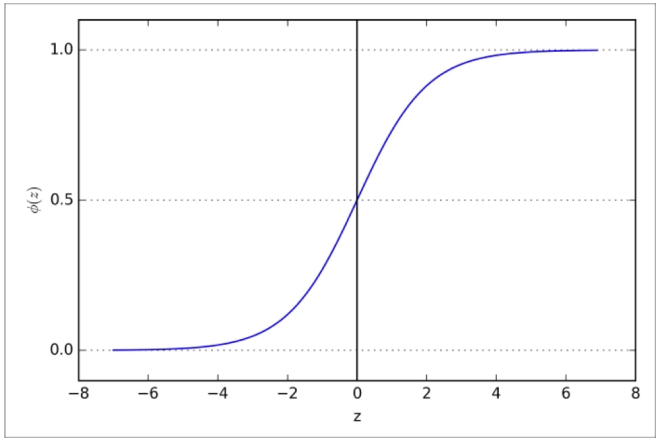
Foster Provost, Tom Fawcett, Data Science for Business, 2013

Logistic Regression

Logistic regression is a **classification model** (despite the name "regression") that is very easy to implement but performs very well on linearly separable classes.

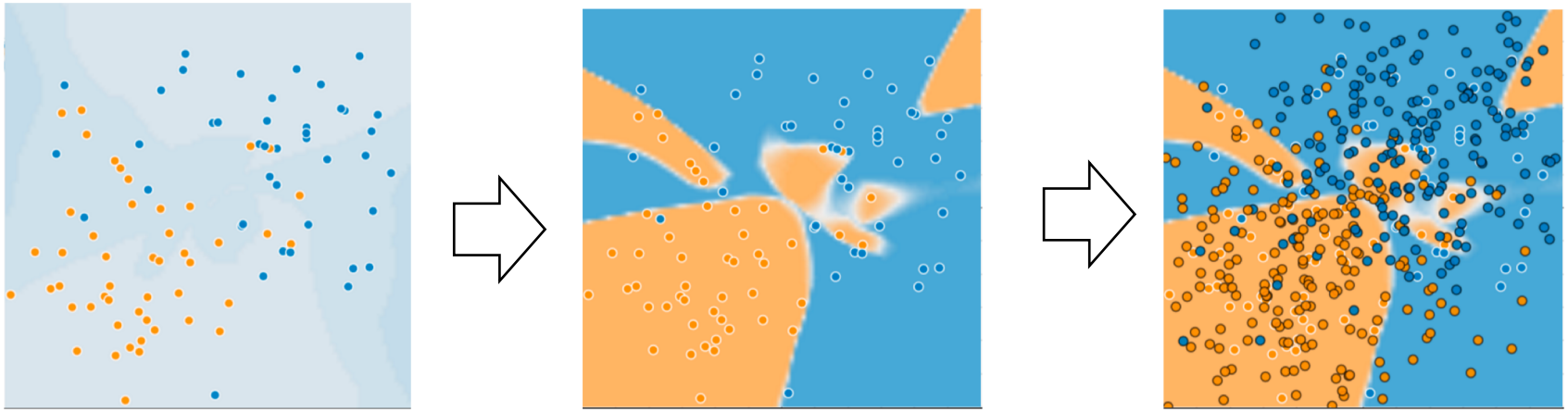
Utilization of logarithmic cost function that is designed to separate two classes





How do we know that the model works?

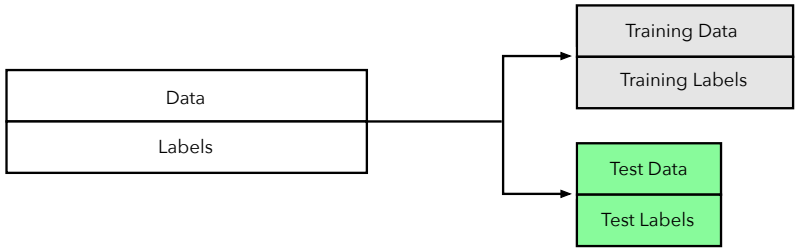
Generalization: Refers to your model's ability to make correct predictions on new, previously unseen data as opposed to the data used to train the model.



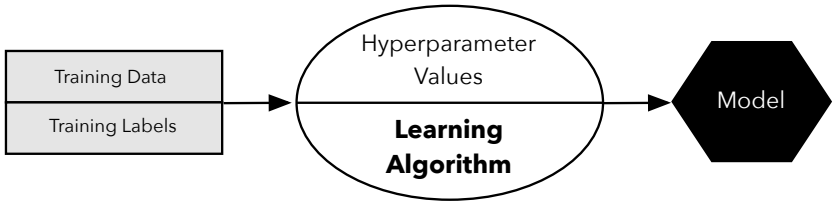
https://developers.google.com/machine-learning/crash-course/generalization/peril-of-overfitting

Training Data and Test Data.

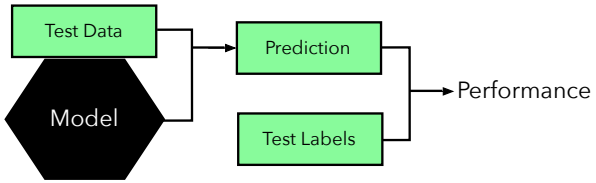
1



2



3



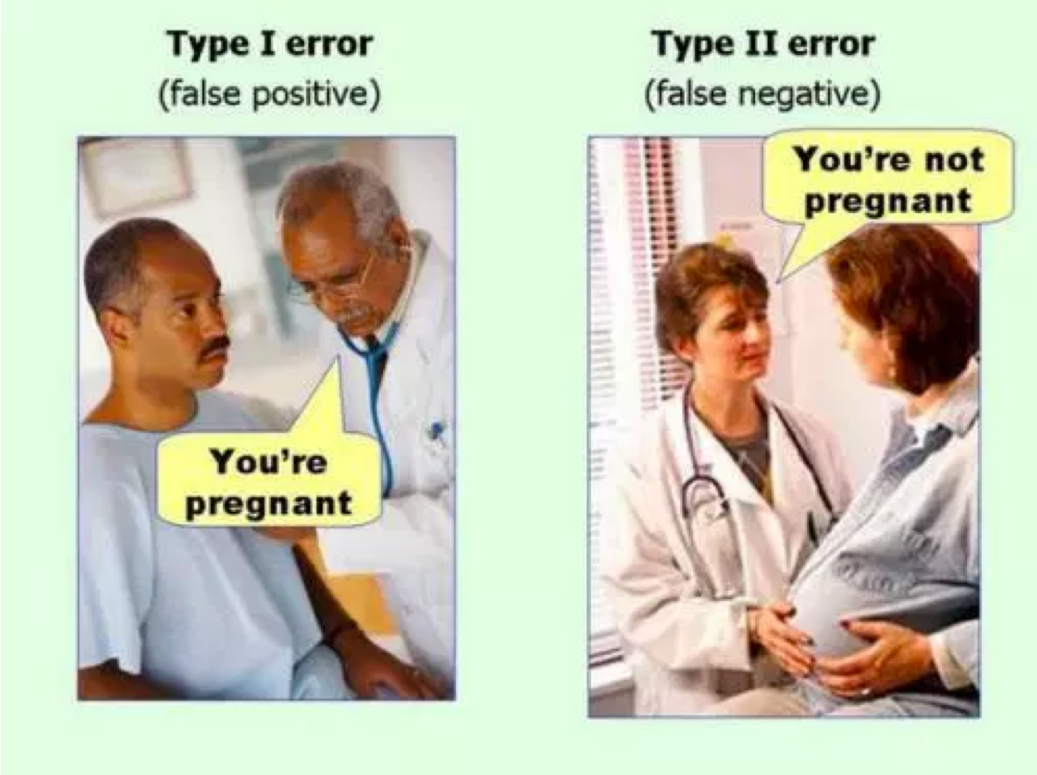
We want to estimate the generalization performance, the predictive performance of our model on future (unseen) data.

It is important to not train and evaluate a model on the same training dataset, since it would typically introduce a very optimistic bias due to overfitting.

If extensive hyper-parameter tuning is done using the testing dataset, a separate validation dataset is recommended.

Keep some data to evaluate your supplier/vendor!

What can go Wrong?



https://chemicalstatistician.wordpress.com/2014/05/12/applied-statistics-lesson-of-the-day-type-i-error-false-positive-and-type-2-error-false-negative/

Model Quality Assessment

Confusion Matrix: A matrix showing the predicted and actual classifications. A confusion matrix is of size $l \times l$, where l is the number of different label values. A variety of classifier evaluation metrics are defined based on the contents of the confusion matrix, including *accuracy, precision, recall*.

	Predicted = FALSE	**Predicted = TRUE**
Actual = FALSE	TN (True Negative)	FP (False Positive)
Actual = TRUE	FN (False Negative)	TP (True Positive)

Alice Zheng, Evaluating Machine Learning Model, https://www.oreilly.com/ideas/evaluating-machine-learning-models, 2015.

Accuracy, Precision, Recall

Accuracy = (TP + TN) / total data point

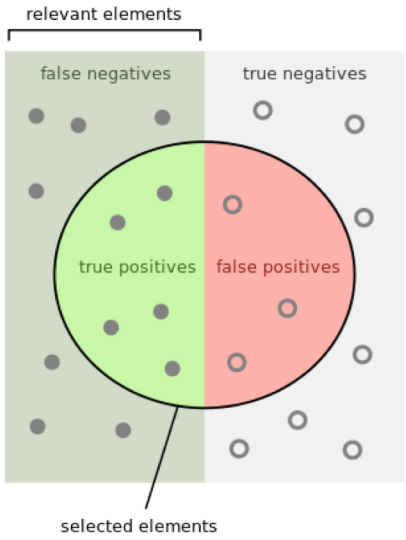
- A good estimate of the entire model

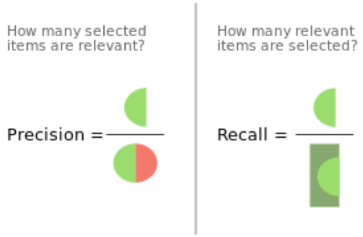
Precision = TP / positive data points (FP and TP)

- What fraction of positively predicted data points is correct
- Out of the items that the ranker/classifier predicted to be truly relevant?

Recall = (TP) / (TP + FN)

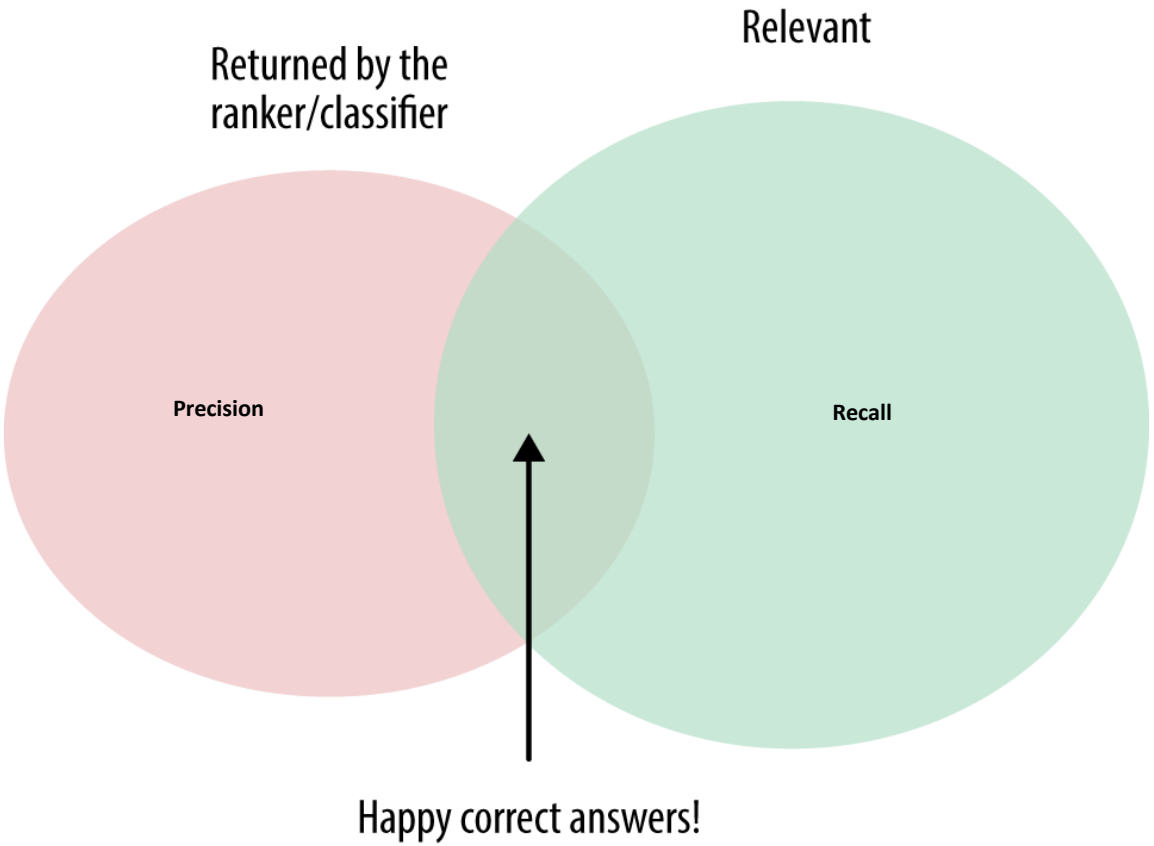
- Out of all the items that are truly relevant, how many are ranker/classifier?
- Measures how many items a classifier is missing!





Alice Zheng, Evaluating Machine Learning Model, https://www.oreilly.com/ideas/evaluating-machine-learning-models, 2015.
https://en.wikipedia.org/wiki/Precision_and_recall

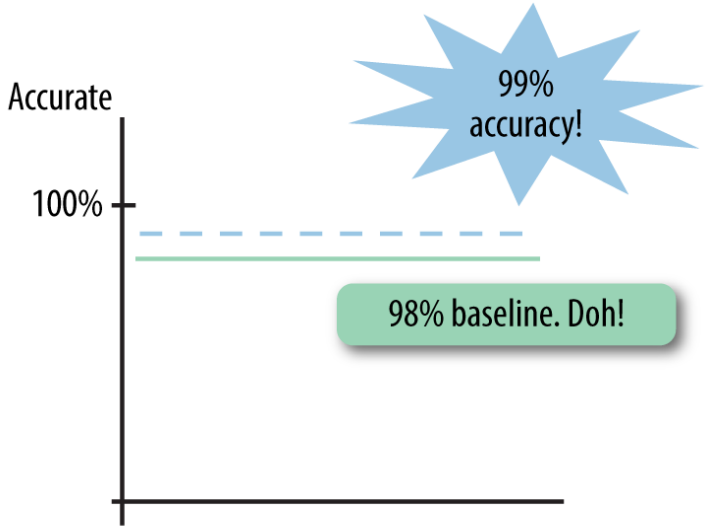
Accuracy, Precision, Recall



Imbalanced Data.

An IT System is down on overage 1% of the time. A vendor proposes a model that can predict downtimes with an accuracy of 99%!

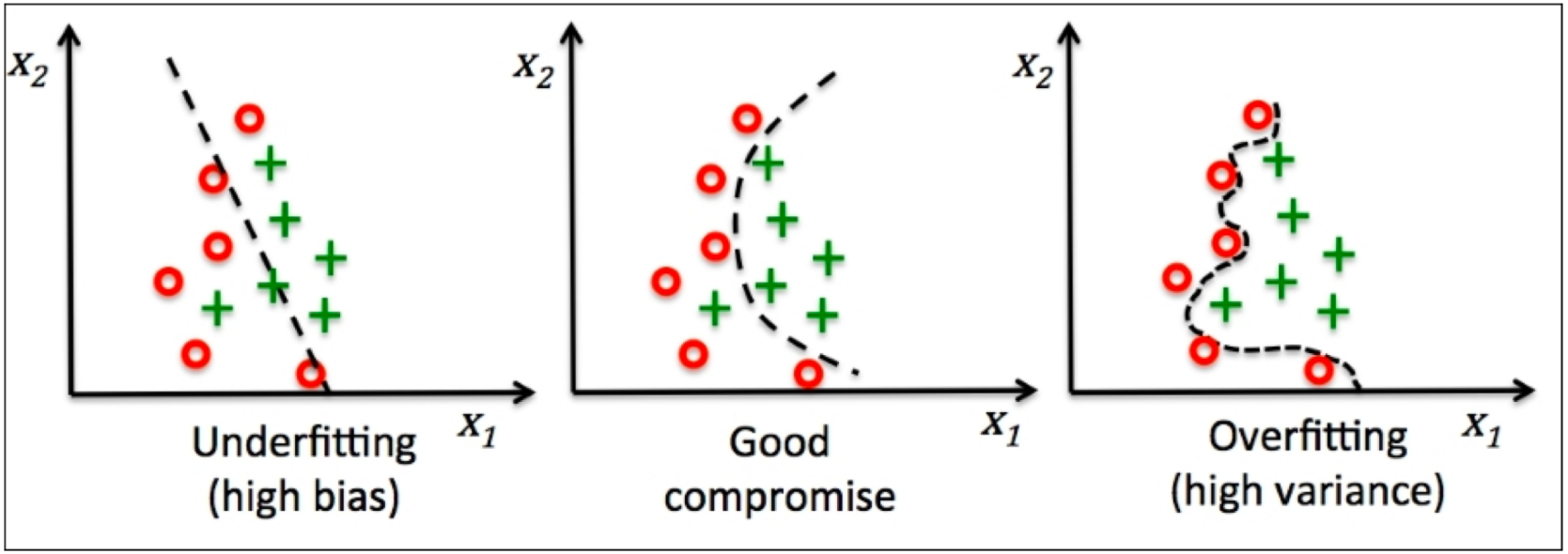
Should you trust the model?



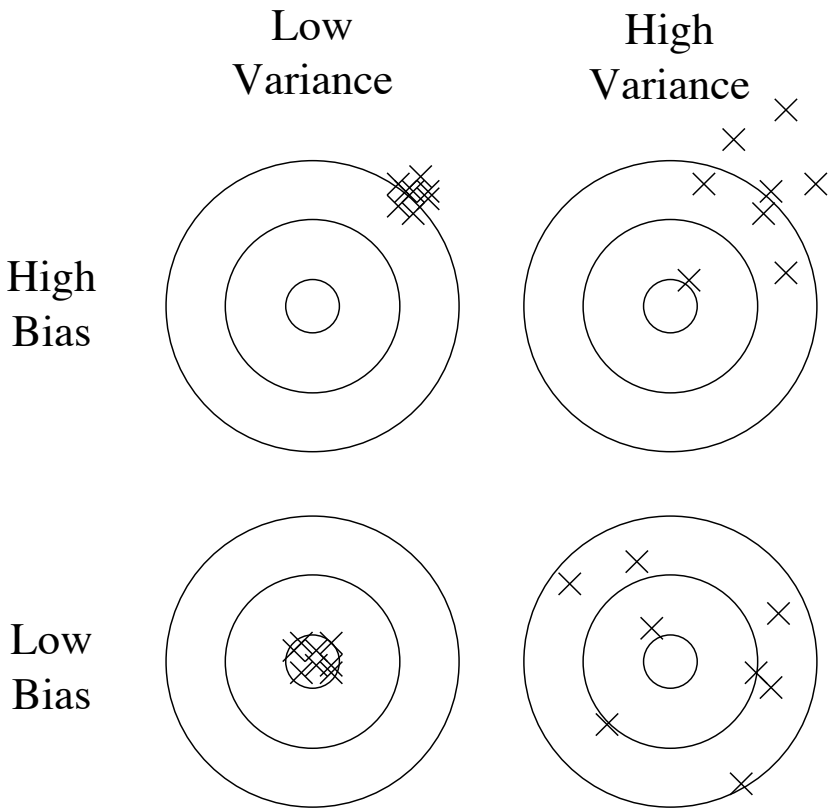
Understand the metrics that your vendor is using to sell you a model!

Alice Zheng, Evaluating Machine Learning Model, https://www.oreilly.com/ideas/evaluating-machine-learning-models, 2015.

Overfitting



Overfitting: Bias - Variance Trade-Off

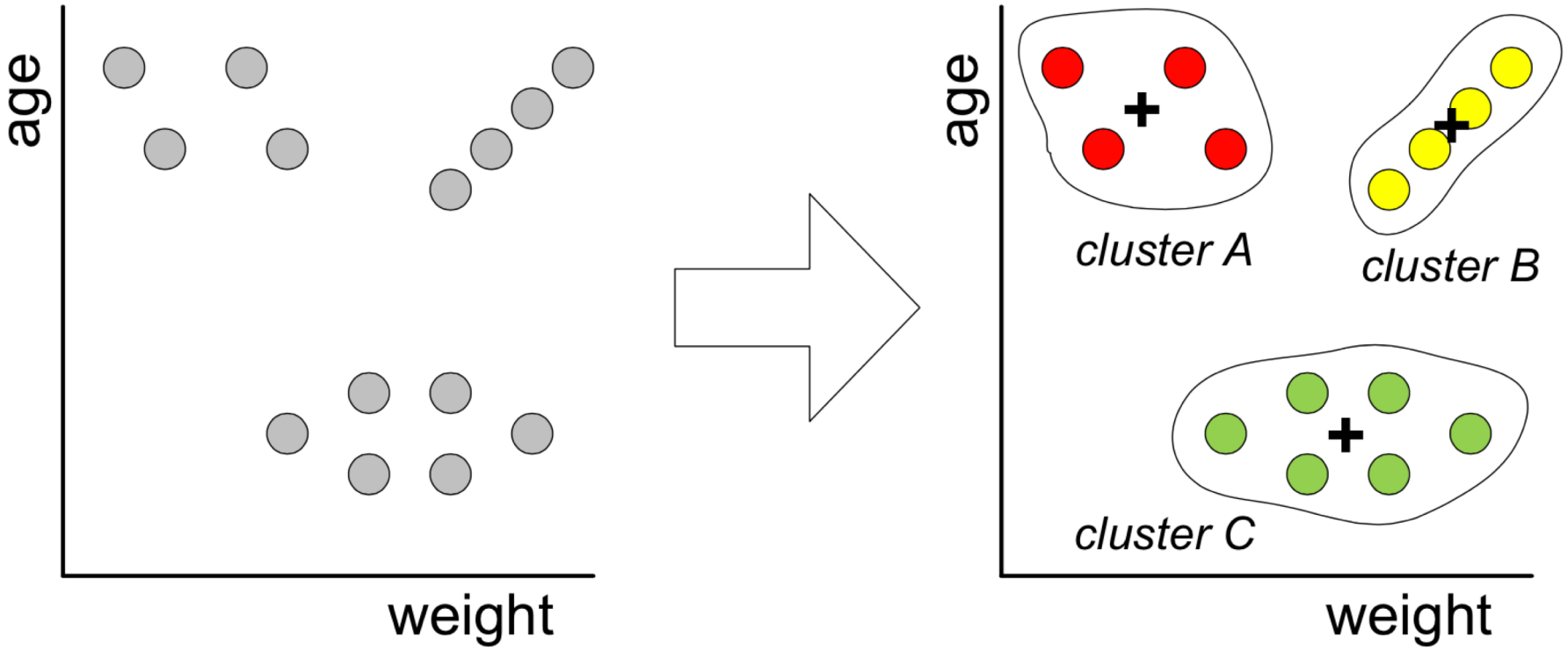


- **Bias:** Tendency of a Classifier to learn the same wrong thing
- **Variance:** Measure of the variability of a model's predictions if we repeat the learning process multiple times with small fluctuations in the training set.

Domingos, A Few Useful Things to Know about Machine Learning, https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf, 2012
Raschka, Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning, 2018

Unsupervised Learning: Clustering

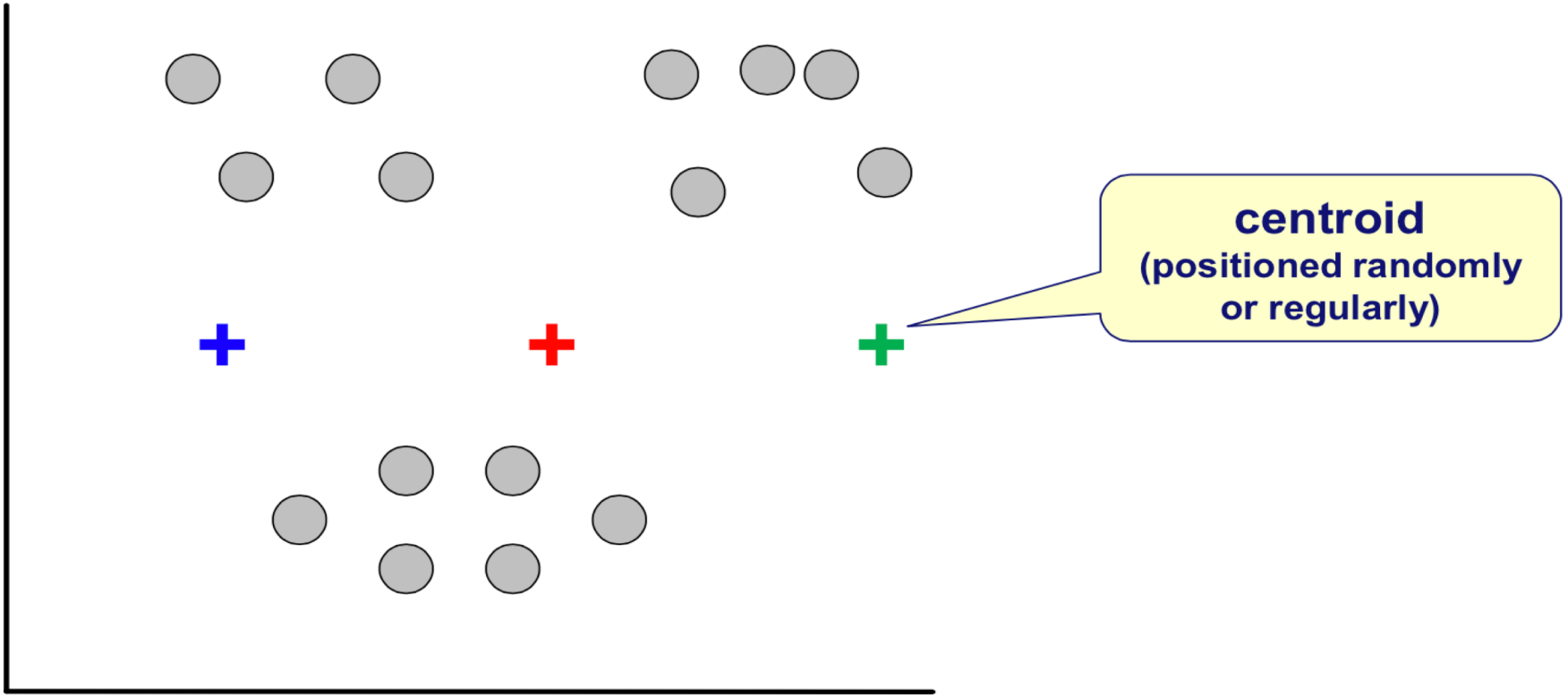
- Group data into “similar” groups
- Best known algorithm: kmeans



Wil van der Aalst, Process Mining: Data science in Action, Coursera, 2014

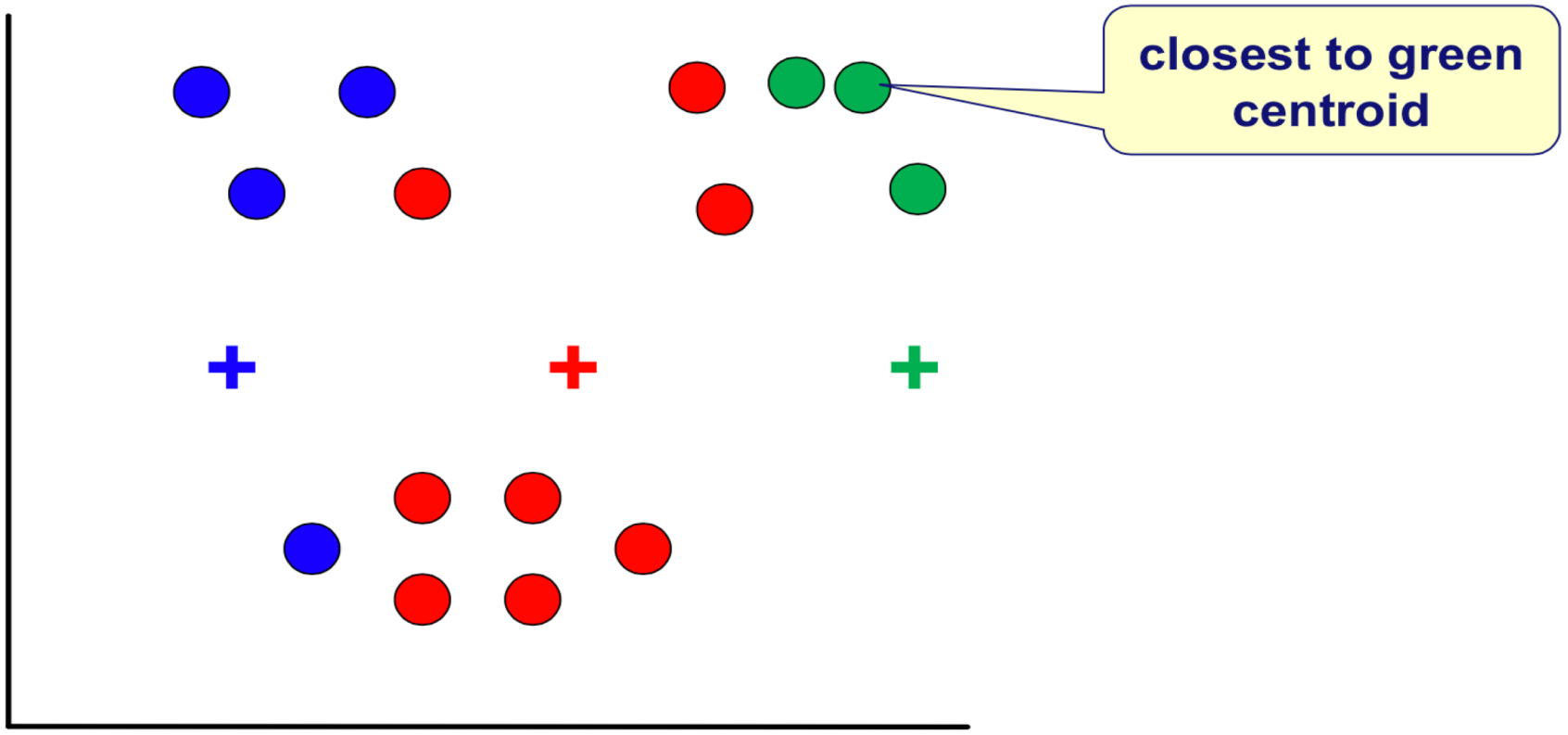
- Kmeans:
  - Initially, you randomly pick k centroids (or points that will be the center of your clusters) in d-space. Try to make them near the data but different from one another.
  - Then assign each data point to the closest centroid.
  - Move the centroids to the average location of the data points (which correspond to users in this example) assigned to it.
  - Repeat the preceding two steps until the assignments don't change, or change very little.

Clustering: KMeans



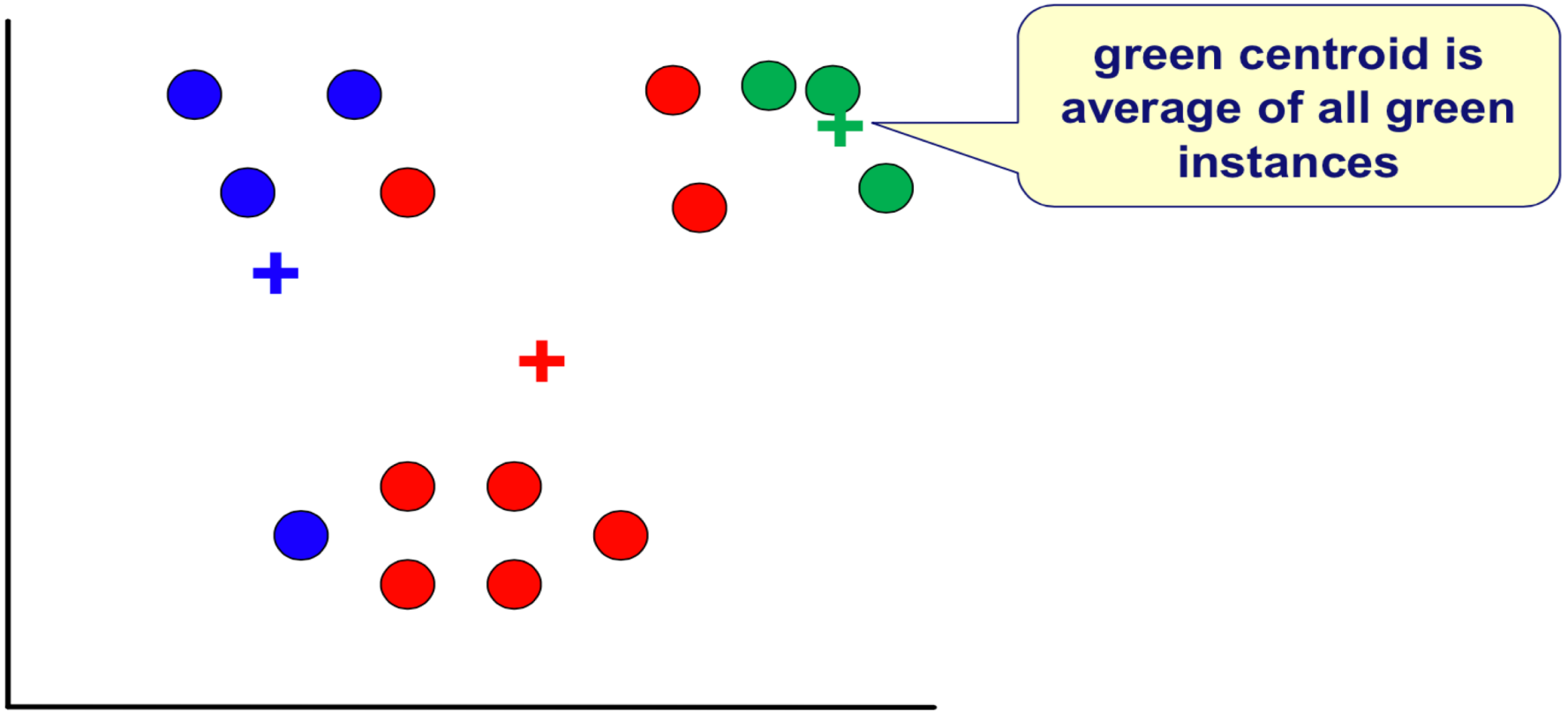
Wil van der Aalst, Process Mining: Data science in Action, Coursera, 2014

Clustering: KMeans



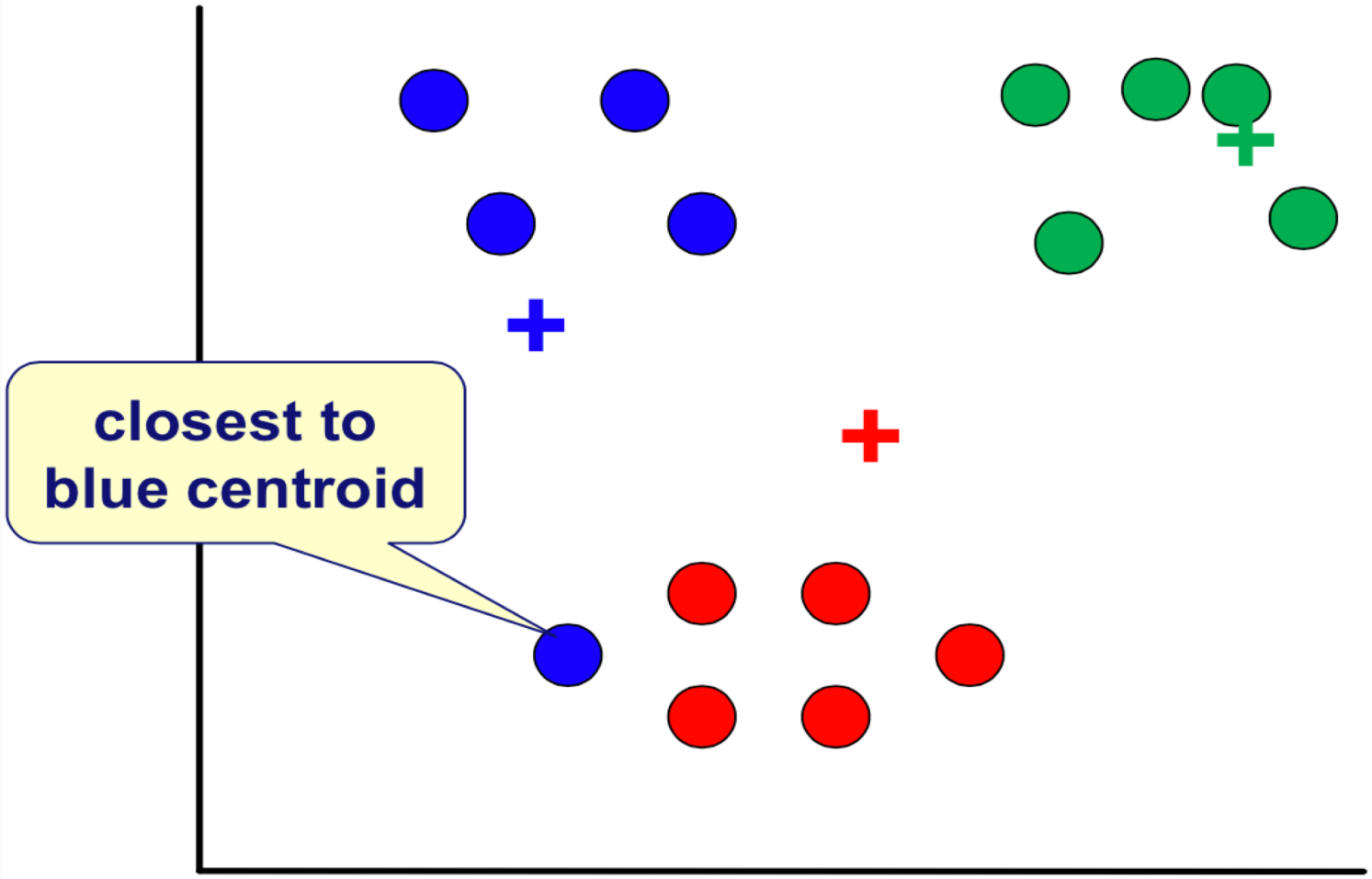
Wil van der Aalst, Process Mining: Data science in Action, Coursera, 2014

Clustering: KMeans



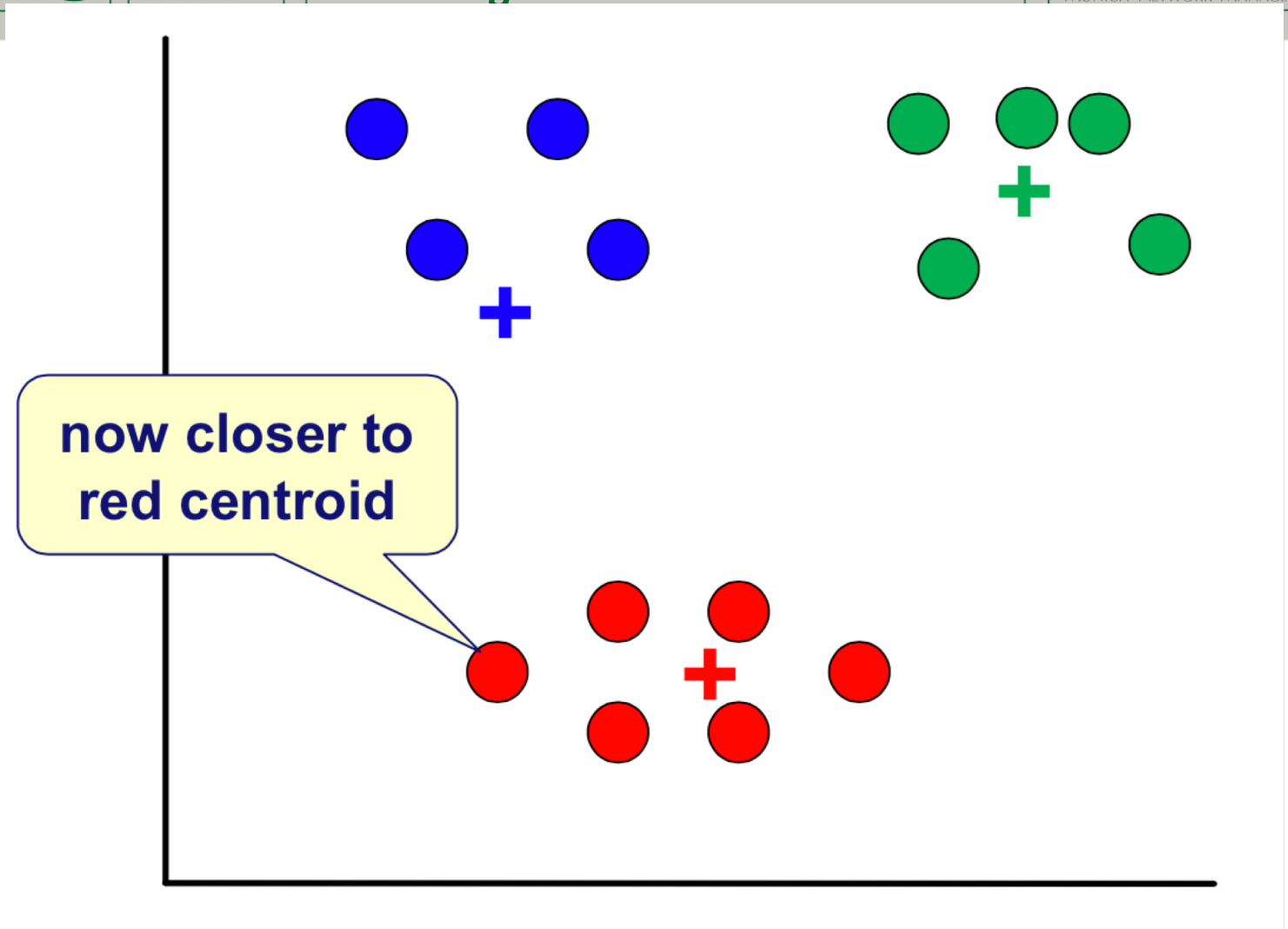
Wil van der Aalst, Process Mining: Data science in Action, Coursera, 2014

Clustering: KMeans



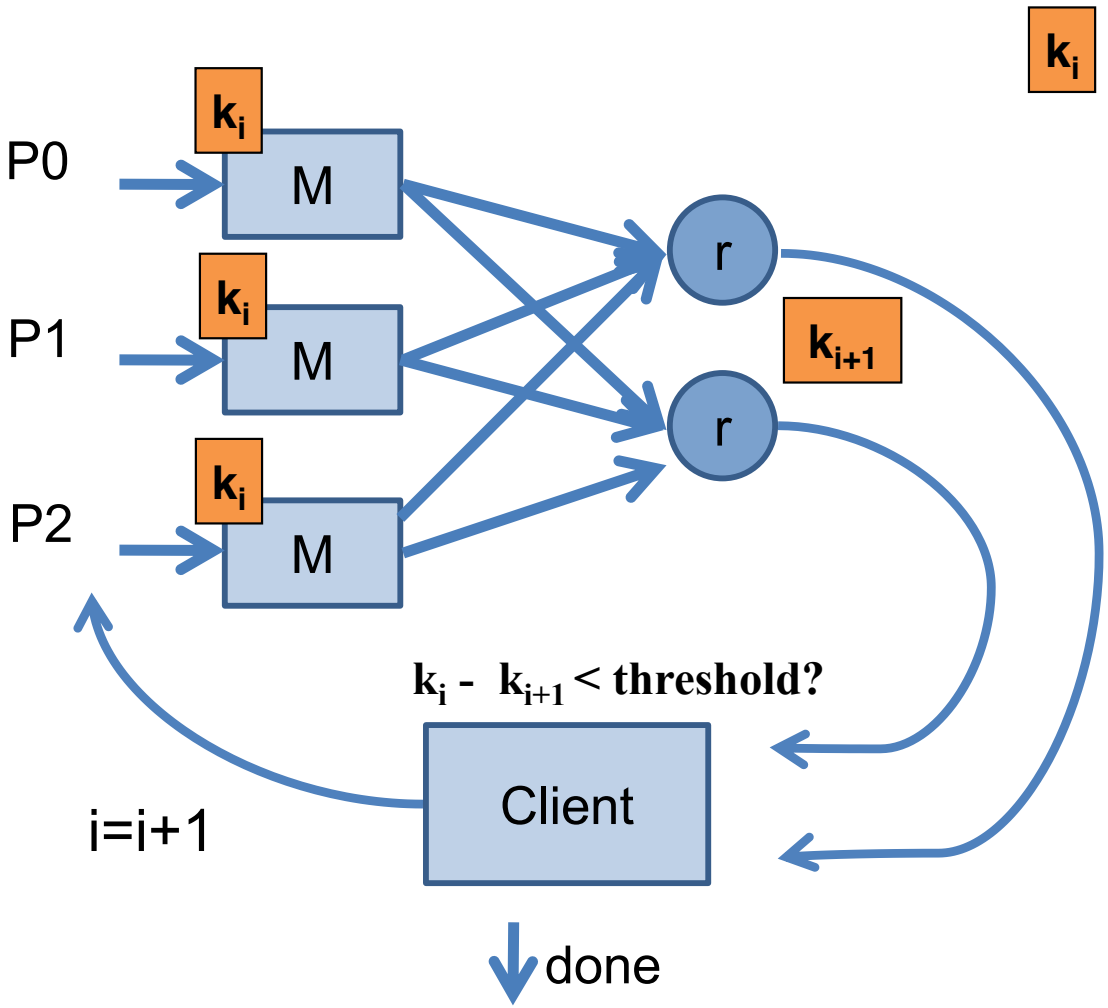
Wil van der Aalst, Process Mining: Data science in Action, Coursera, 2014

Clustering: KMeans



Wil van der Aalst, Process Mining: Data science in Action, Coursera, 2014

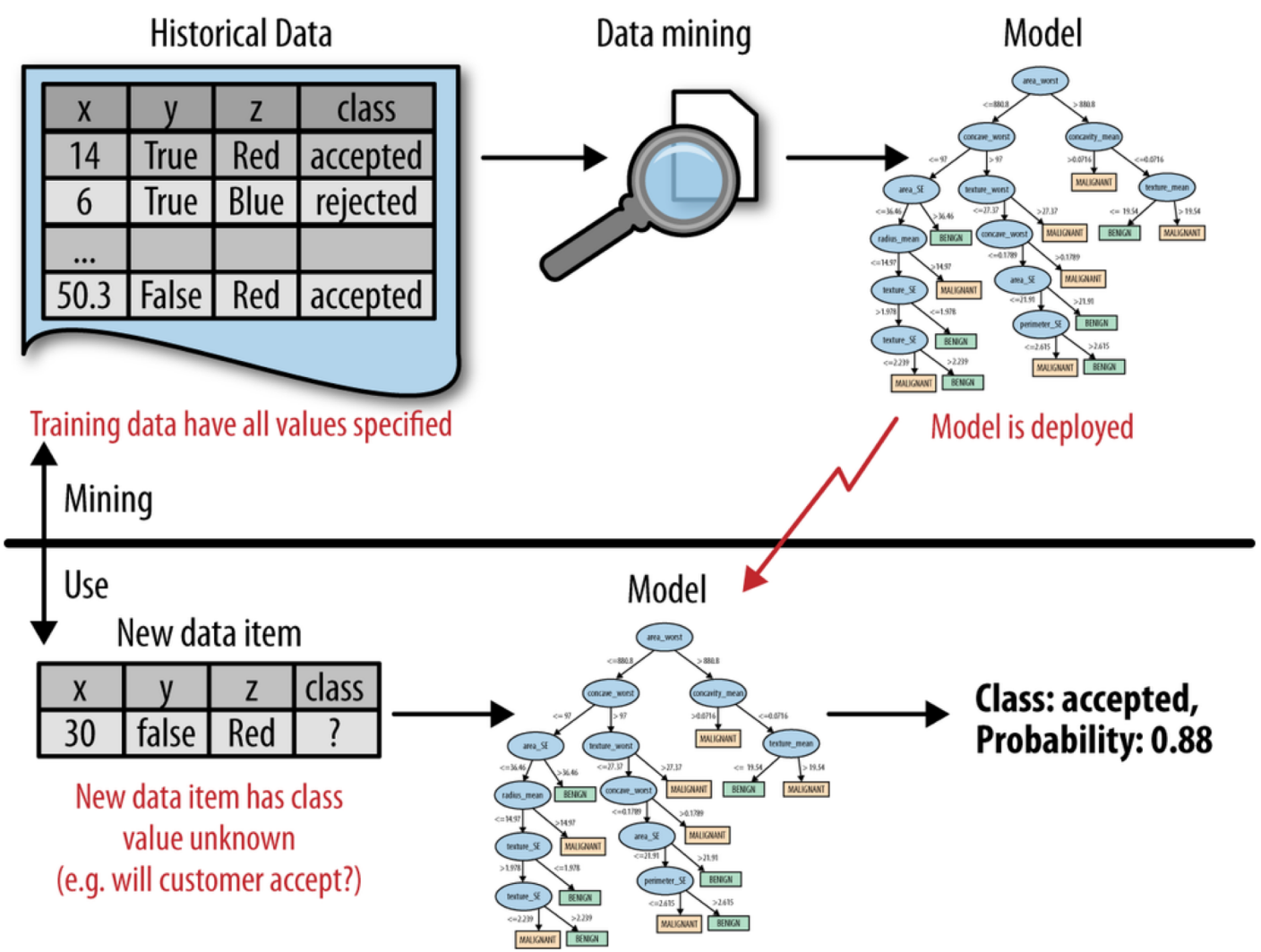
Unsupervised Learning: Clustering



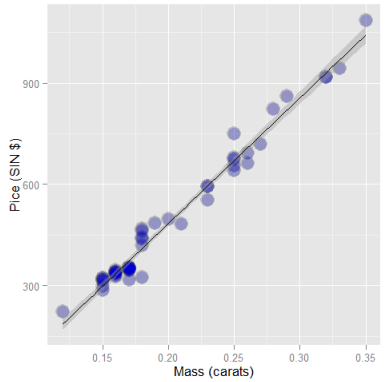
Map Phase: Compute distance of each point to centroid and assign each point to a centroid

Reduce Phase: Compute new centroid as average of all points assigned to one cluster

Model Building and Deployment



- Successor of S language first released in 1996:
  - Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics, 5(3):299–314, 1996
- Objectives: “We wanted users to be able to begin in an interactive environment, where they did not consciously think of themselves as programming. Then as their needs became clearer and their sophistication increased, they should be able to slide gradually into programming, when the language and system aspects would become more important.”
- Introduced DataFrame Abstraction
- Standard build-in syntax for statistical modeling and prediction




```
> fit <- lm(price ~ carat, data = diamond)
> coef(fit)
(Intercept)       carat
     -259.6      3721.0
```

- NumPy: Matrix representations and linear algebra
- Pandas (https://pandas.pydata.org/):
  - Higher-Level Dataframe library on top of NumPy
  - Objective: “Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.”
  - Inspired by R
- Scikit-Learn:
  - Machine Learning algorithm implementations
- Introduces Estimator API for unified training, evaluation, prediction, export for serving
- Matplotlib, Seaborn for Plotting

Scikit-Learn

In [1]:
```
%matplotlib inline
from sklearn import datasets
import pandas as pd
import numpy as np
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
iris_df["target_name"]=iris['target_names'][iris_df['target']]
iris_df.head()
```

Out[2]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	target_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

- Scikit-Learn provides access to different test dataset, e.g. IRIS Dataset.
- Datasets are encoded as NumPy multidimensional arrays for dense data.
- Interoperability with Pandas Dataframe (also based on NumPy)

Scikit-Learn: Estimator

The estimator interface is at the core of the library. It defines instantiation mechanisms of objects and exposes a fit method for learning a model from training data. All supervised and unsupervised learning algorithms (e.g., for classification, regression or clustering) are offered as objects implementing this interface. Machine learning tasks like feature extraction, feature selection or dimensionality reduction are also provided as estimators.

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(penalty="l1")
clf.fit(X_train, y_train)
```

```
from sklearn.cluster import KMeans

km = KMeans(n_clusters=10)
km.fit(X_train)
clust_pred = km.predict(X_test)
```

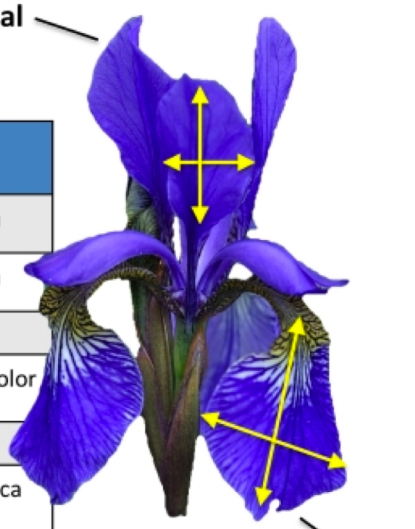
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
```

Buitinck et al., API design for machine learning software: experiences from the scikit-learn project, https://arxiv.org/pdf/1309.0238.pdf, 2013

Scikit-Learn Exercise

Samples
(instances, observations)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica



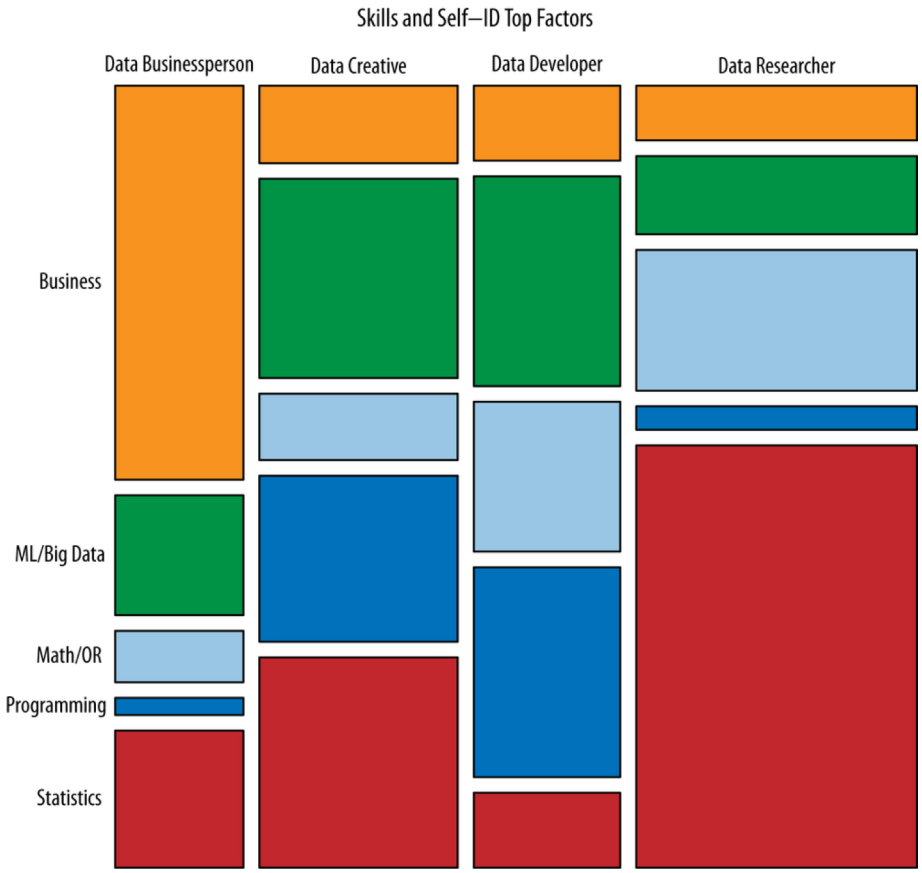
Features
(attributes, measurements, dimensions)

Class labels
(targets)

The IRIS dataset contains 150 examples with 4 features each

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}$$

Data Science with different focus areas



Business	ML / Big Data	Math / OR	Programming	Statistics
Product Developement	Unstructured Data	Optimization	Systems Administration	Visualization
Business	Structured Data	Math	Back End Programming	Temporal Statistics
	Machine Learning	Graphical Models	Front End Programming	Surveys and Marketing
	Big and Distributed Data	Bayesian / Monte Carlo Statistics		Spatial Statistics
		Algorithms		Science
		Simulation		Data Manipulation
				Classical Statistics

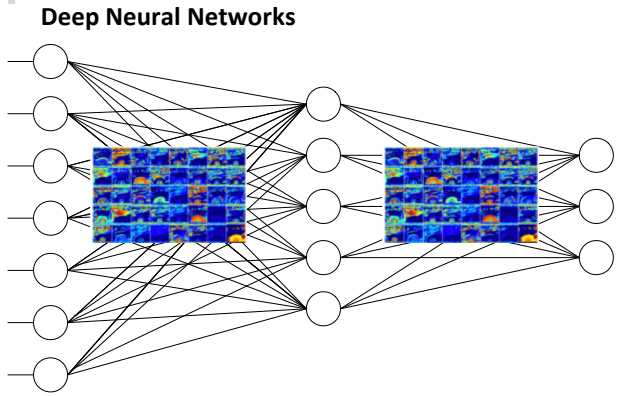
Harlan Harris, Sean Murphy and Marck Vaisman, Analyzing the Analyzers, O'Reilly, 2013

- Peng, Matsui, The Art of Data Science, 2017
- Hey, The Fourth Paradigm of Scientific Discovery, 2009
- Cao, Data Science: Challenges and Directions, Communications of the ACM, 2017
- Donoho, 50 Years of Data Science, 2015
- Machine Learning Frameworks (Github), 2017

Deep Learning

What is Deep Learning?

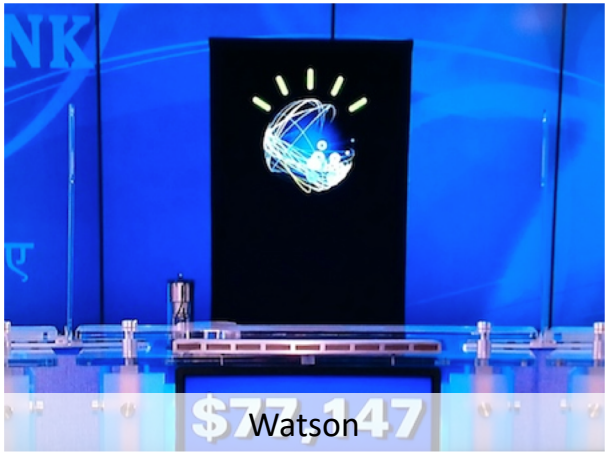
- Deep learning are techniques that use complex neural networks that have the ability to learn abstract concepts. (Bengio, Machines Who Learn, 2016, doi:10.1038/scientificamerican0616-46).
- Key Enablers:
  - Big data
  - Computing
  - Better scalable algorithms & tools



Challenge and Objective: Landscape of infrastructures and tools for deep learning is evolving rapidly. Systematic approach for evaluating deep learning use cases and tools required.

History of Smart Machines: From Deep Blue, Watson to Alpha Go.

Deep Blue



Watson

Deep Mind Alpha Go

Computer Vision Applications.

Image classification and search

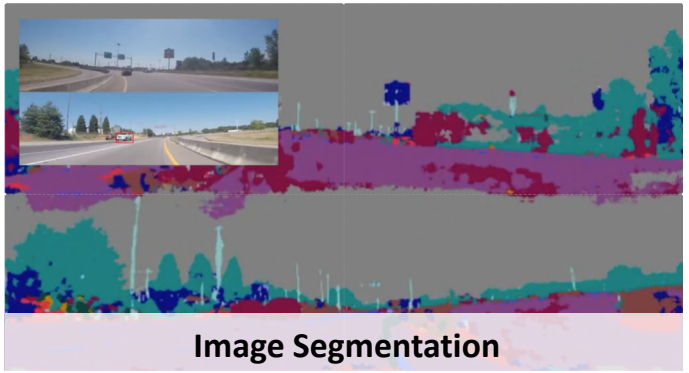
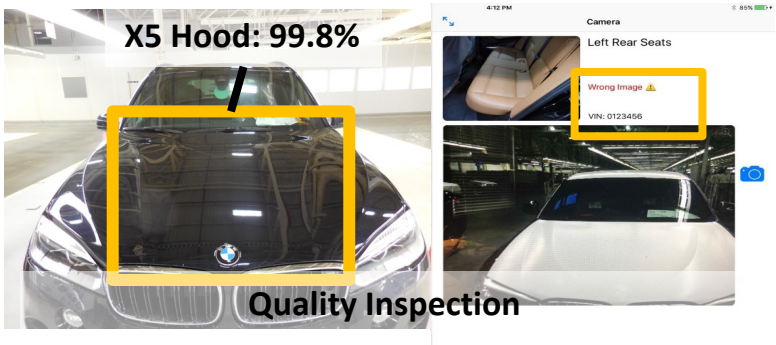
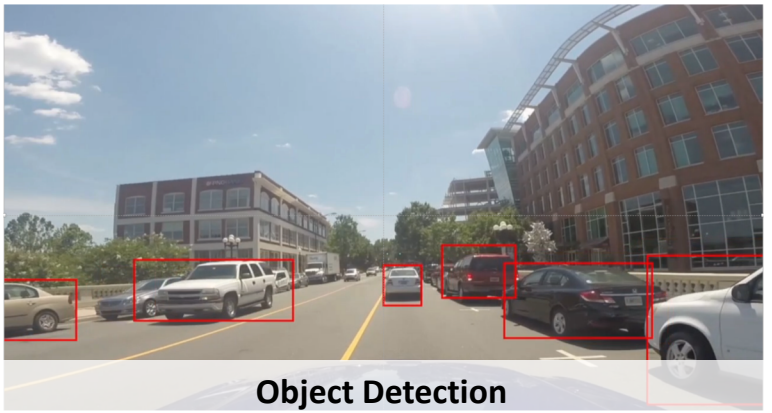


Image Segmentation

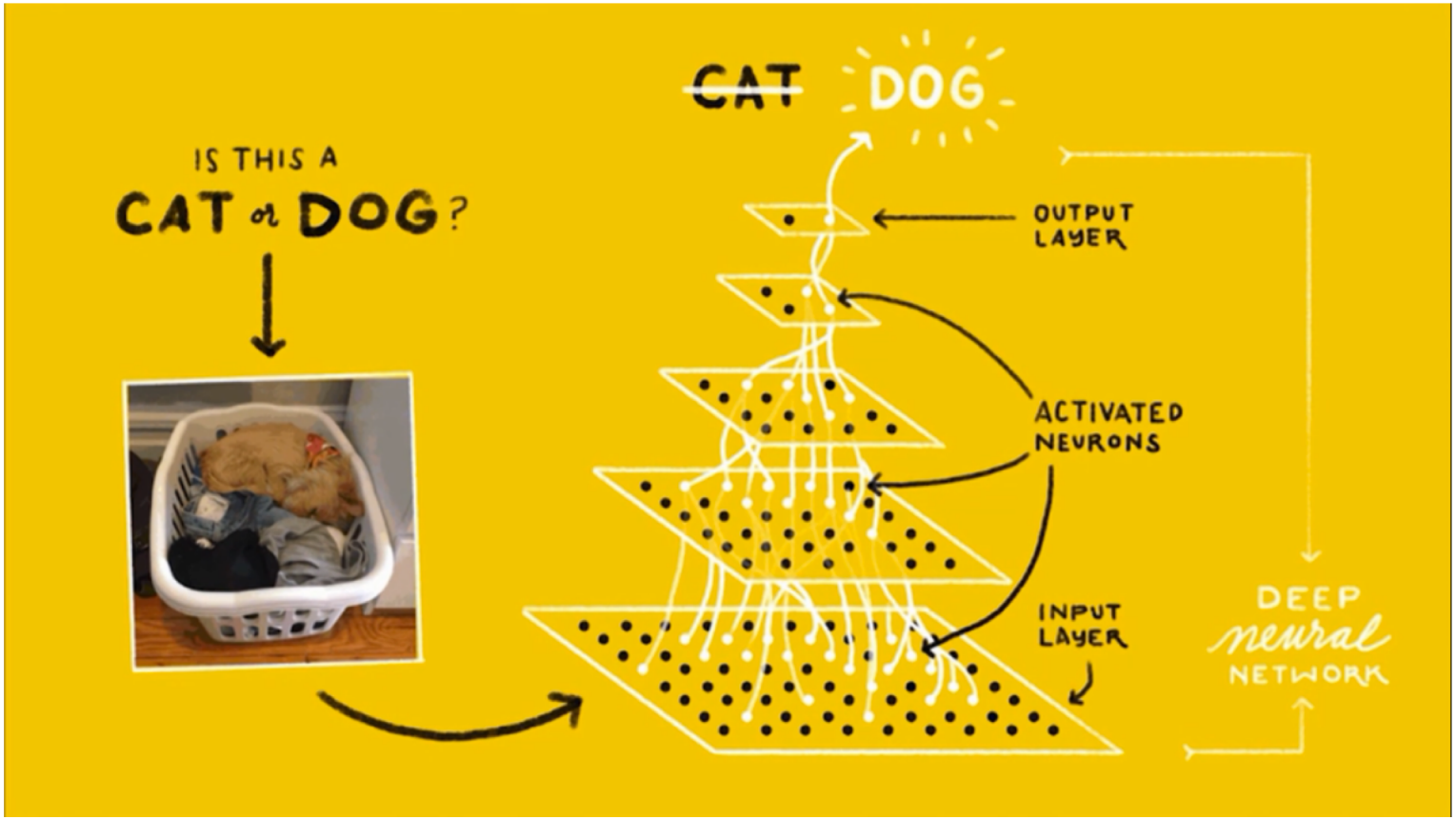


Quality Inspection



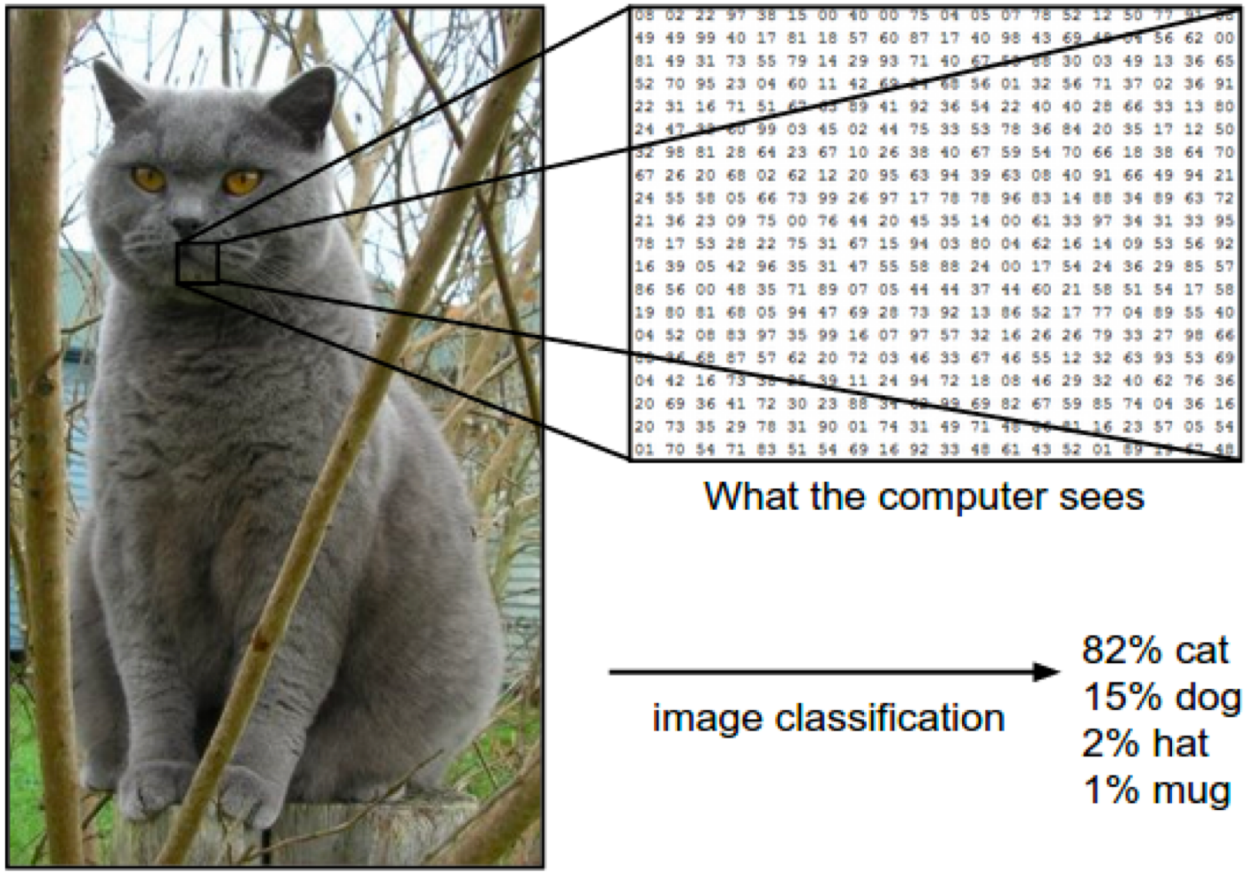
Object Detection

Deep Learning.



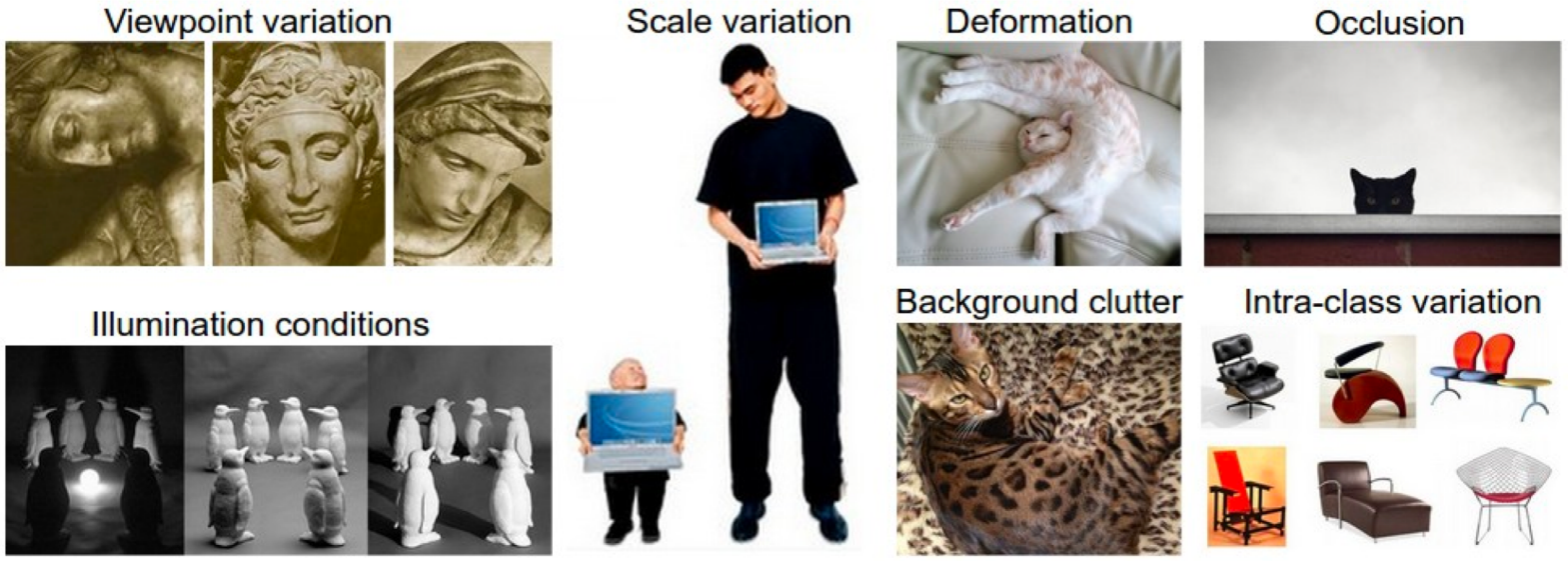
https://www.google.com/about/main/machine-learning-qa/

Image Classification



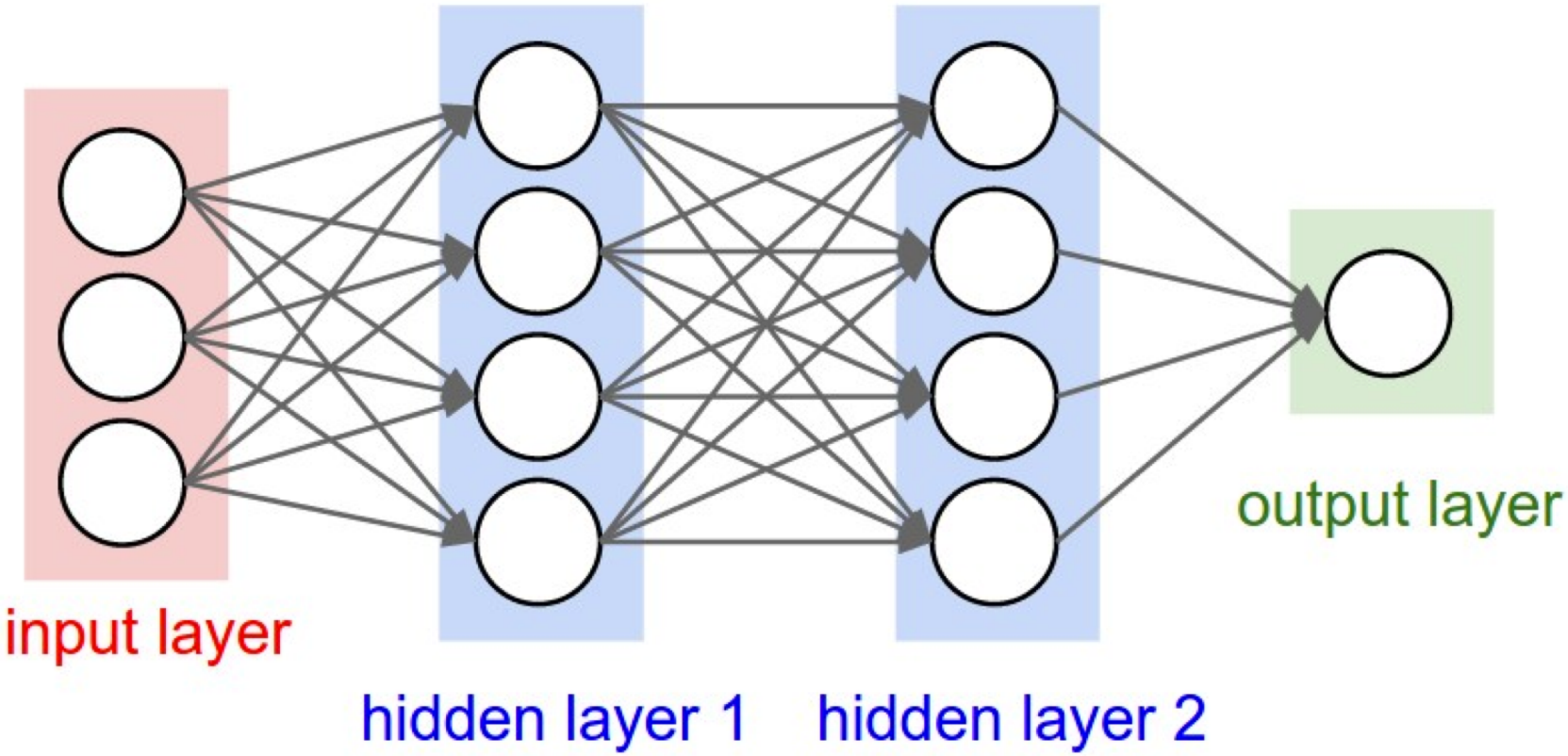
http://cs231n.github.io/classification/

Image Classification



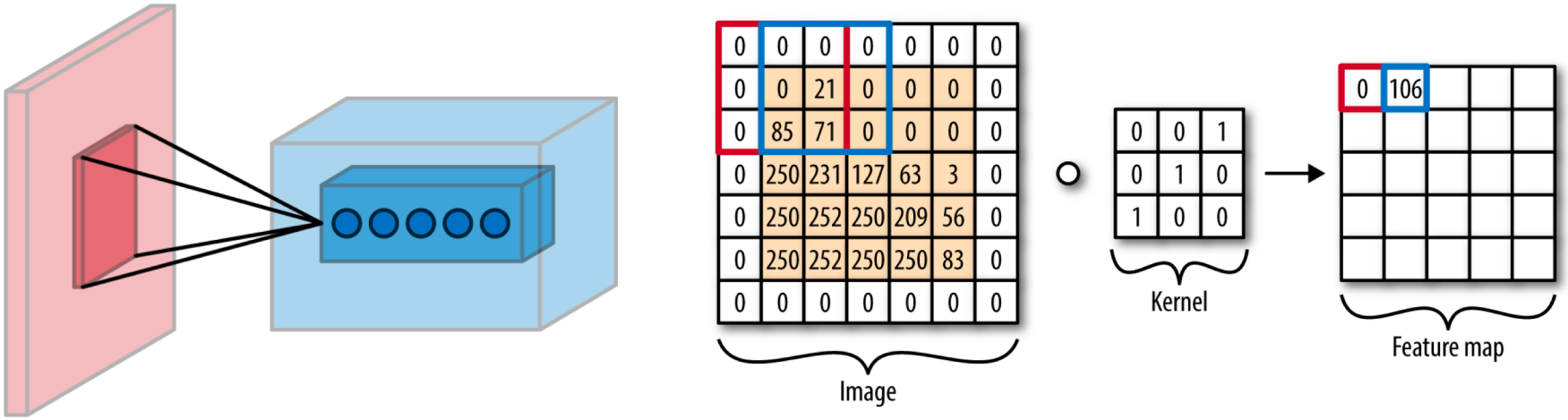
http://cs231n.github.io/classification/

Neural Networks vs. Convolutional Neural Networks



Fei-Fei Li et al., CS231n, Stanford, http://cs231n.github.io/convolutional-networks/

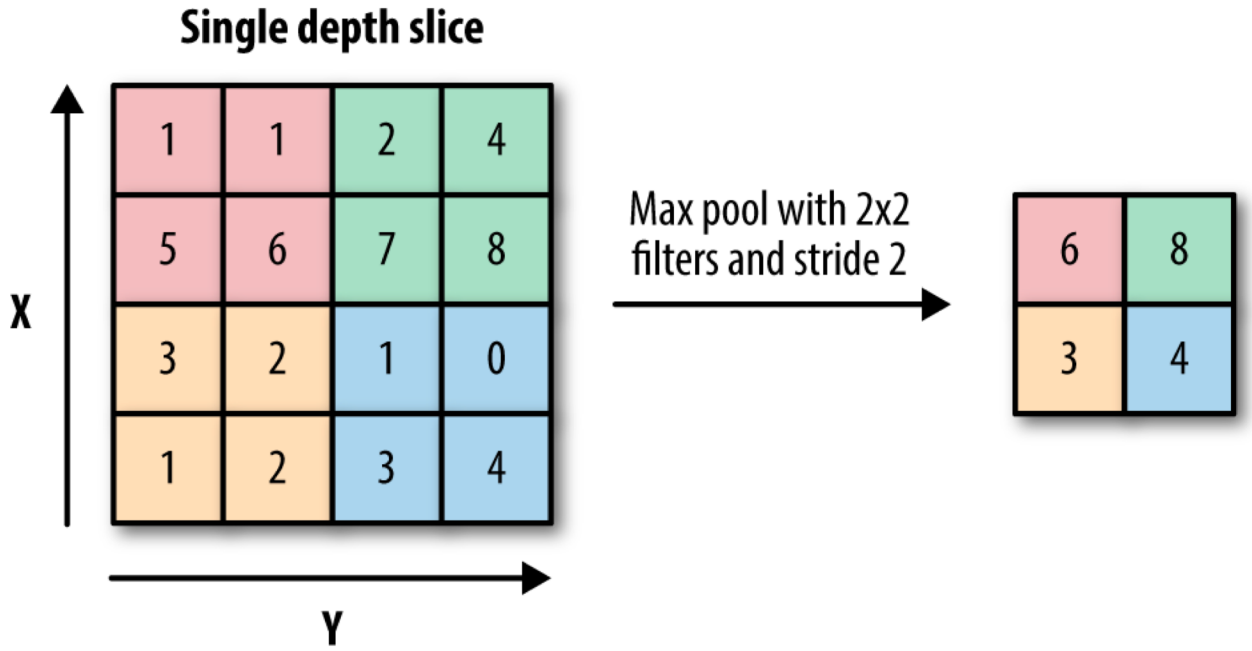
Convolutional Neural Networks: Convolutional Layer.



Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations.
ConvNet is a sequence of Convolution Layers, interspersed with activation functions

Zadeh, Tensorflow for Deep Learning, O'Reilly, 2018.

Convoluational Neural Networks: Pooling

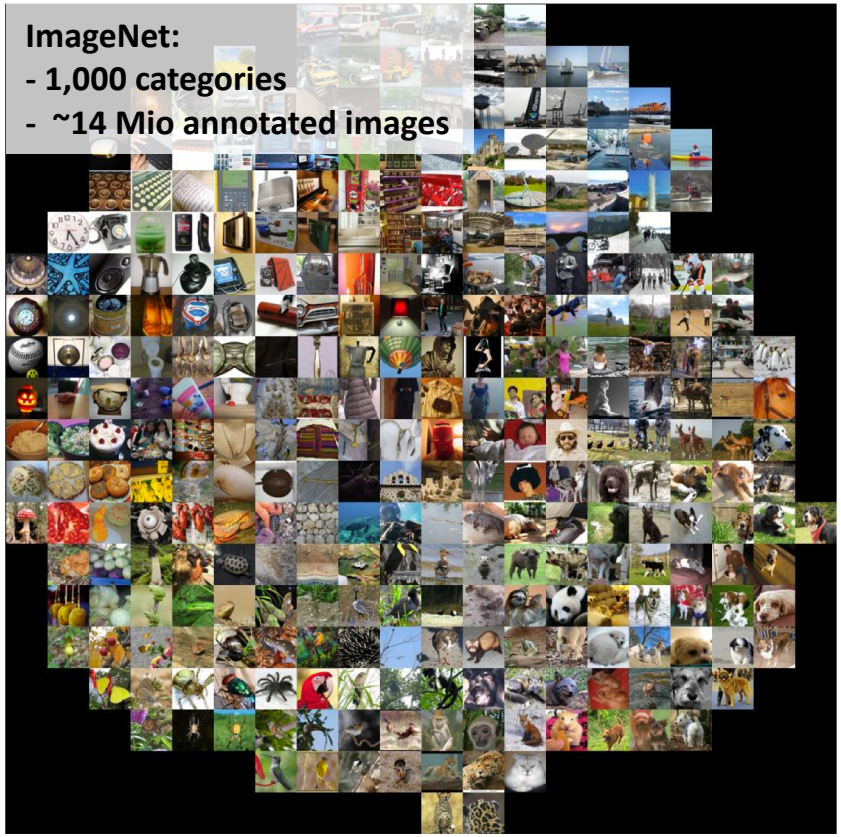


Zadeh, Tensorflow for Deep Learning, O'Reilly, 2018.

Convolutional Neural Networks: Further Layers

- **Fully-connected layer:** Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.
- **RELU layer** will apply an elementwise activation function, such as the max(0,x)max(0,x) thresholding at zero.

ImageNet Challenge

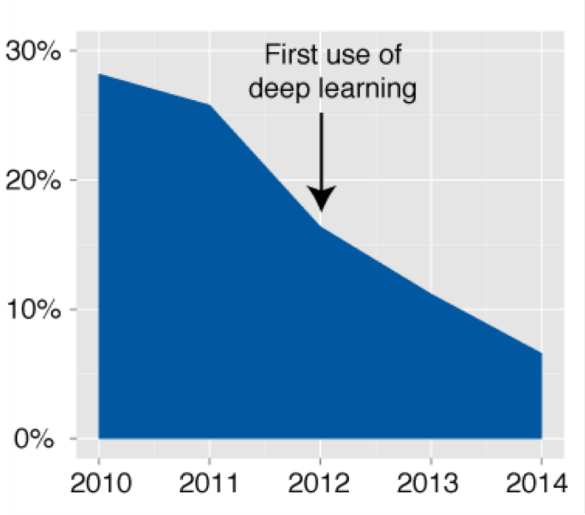


Delving Deep into Rectifiers:
Surpassing Human-Level Performance on ImageNet Classification

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun

Microsoft Research

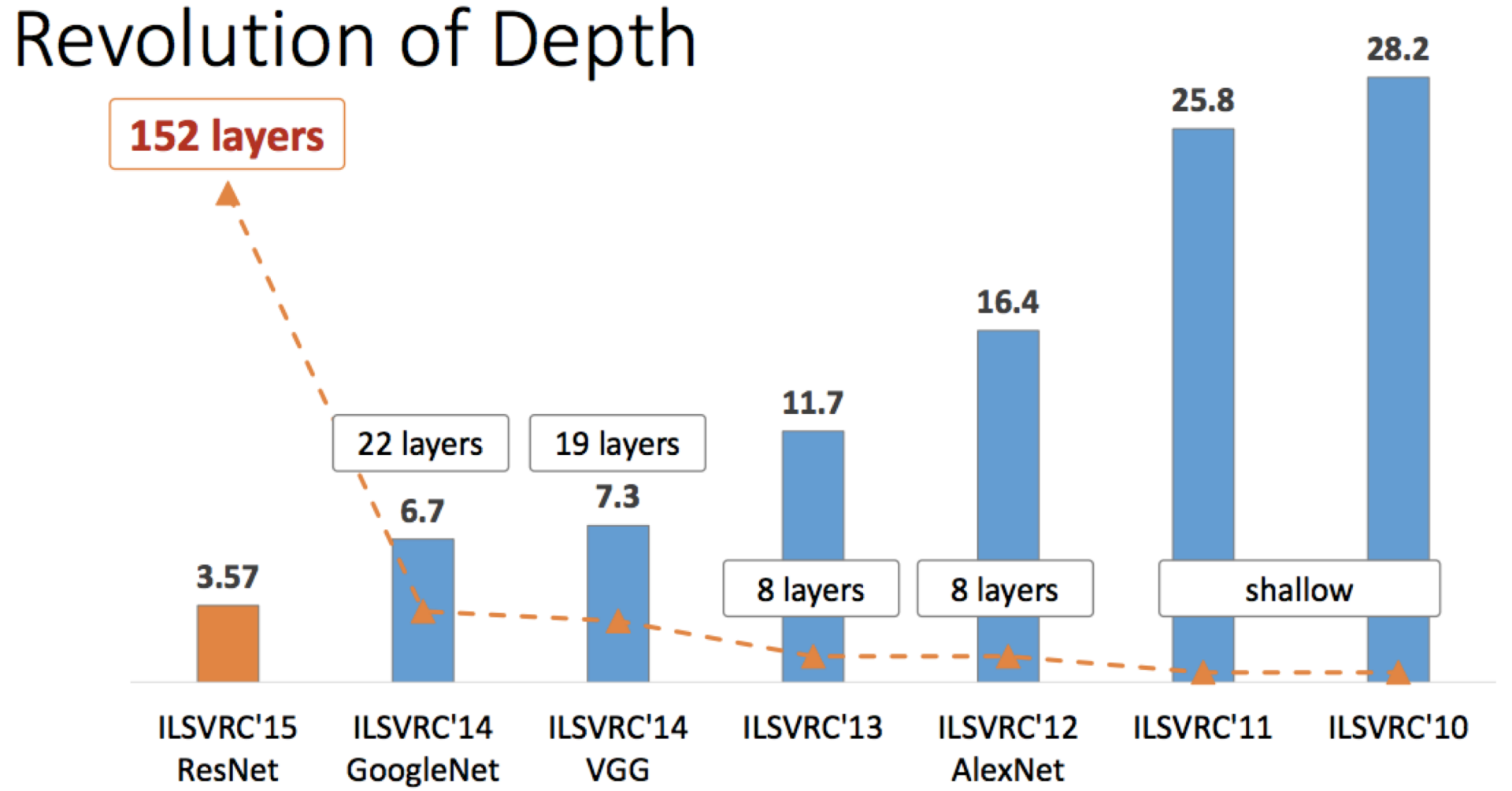
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com



Microsoft in 2015:
4.9% Error Rate –
Better than human

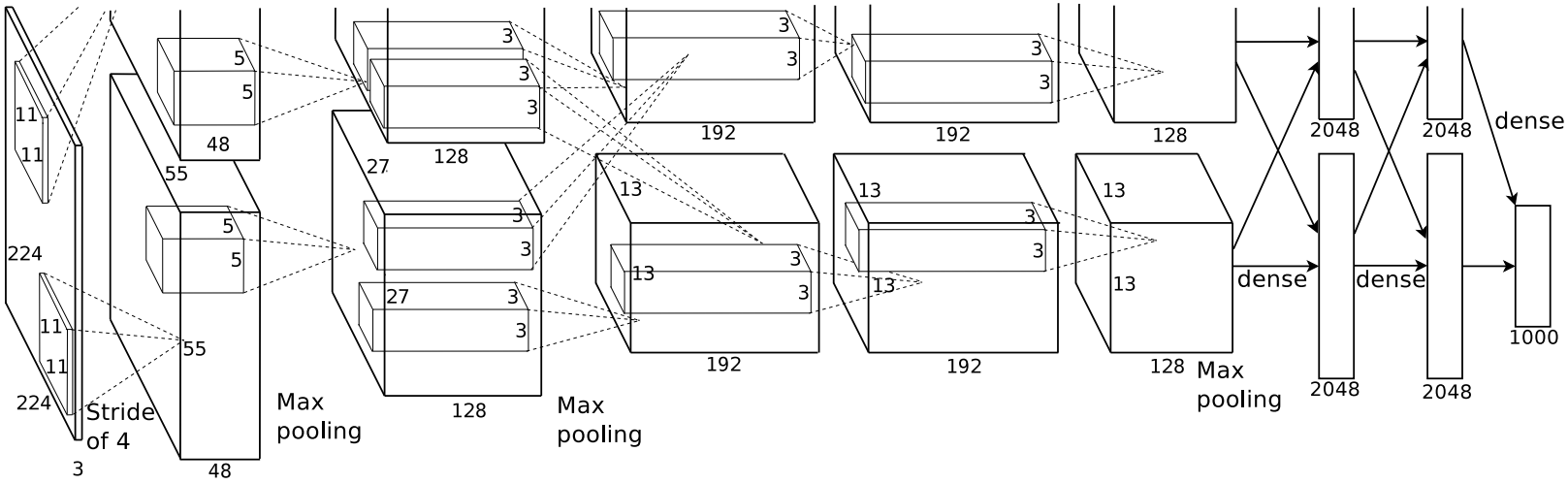
http://arxiv.org/pdf/1502.01852.pdf, 2015.

ImageNet Classification Accuracy



Kaiming He, https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf, 2016.

AlexNet



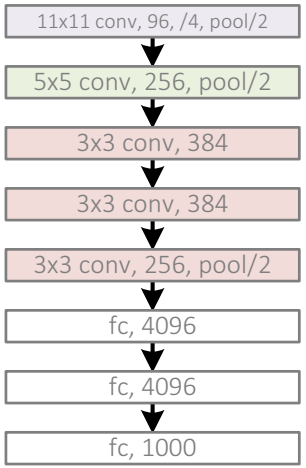
8 Parameter Layer

Input: 227x227x3 images

FC6: 4096 neurons [4096]

FC7: 4096 neurons [1000]

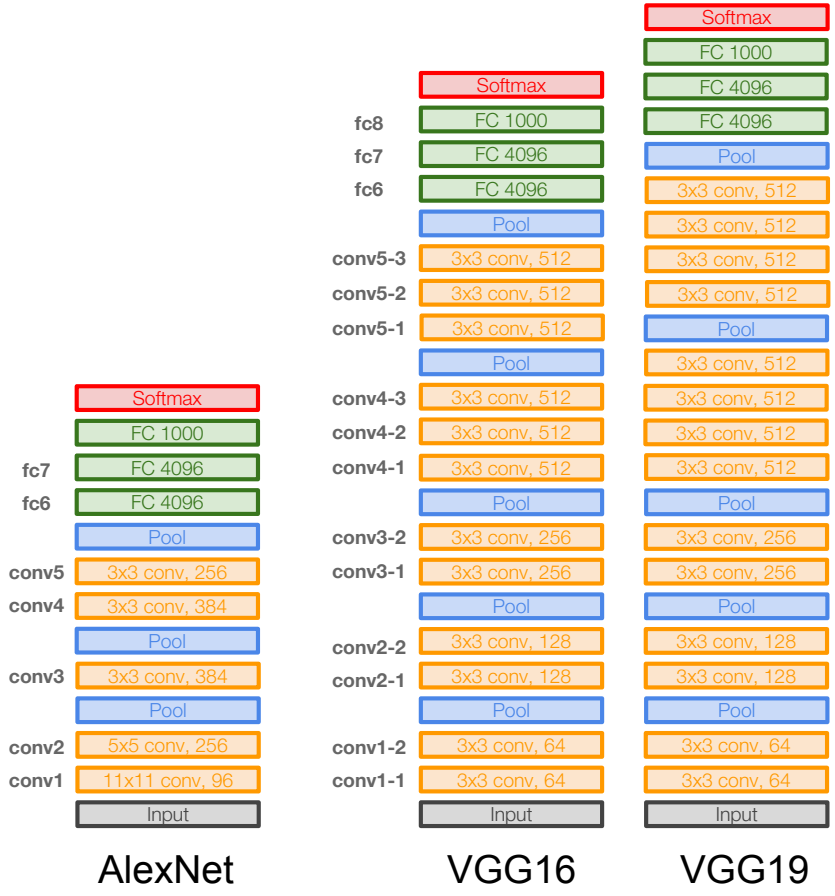
FC8: 1000 neurons (class scores)



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

VGGNet

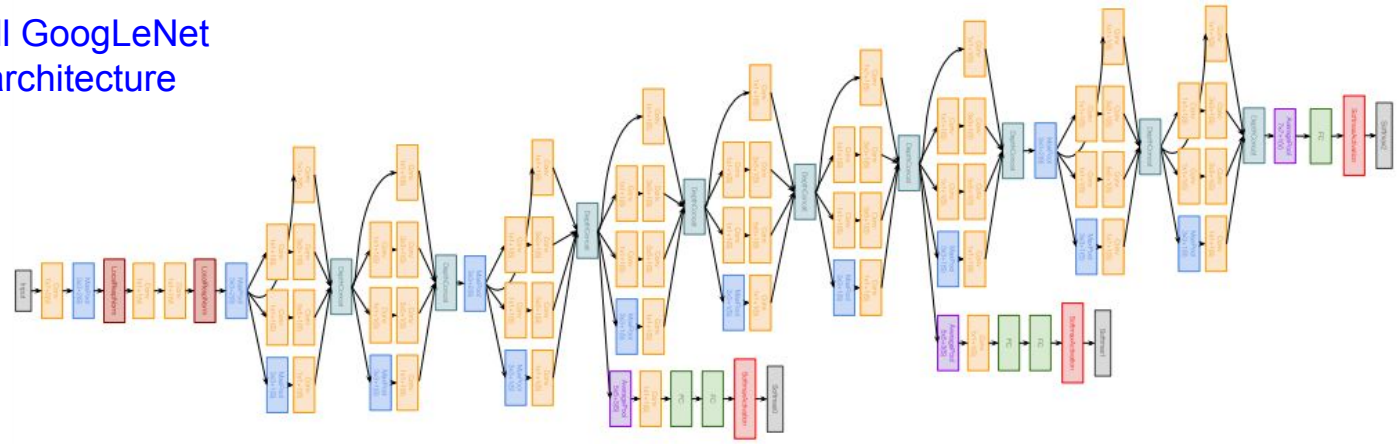
- Small filters, Deeper networks
- 138 mio parameters



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

GoogleNet/Inception

Full GoogLeNet architecture

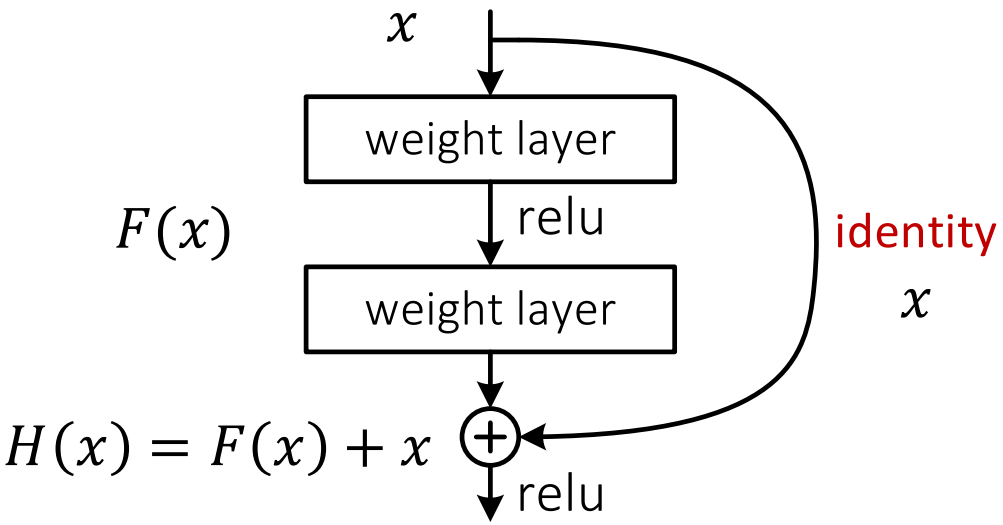


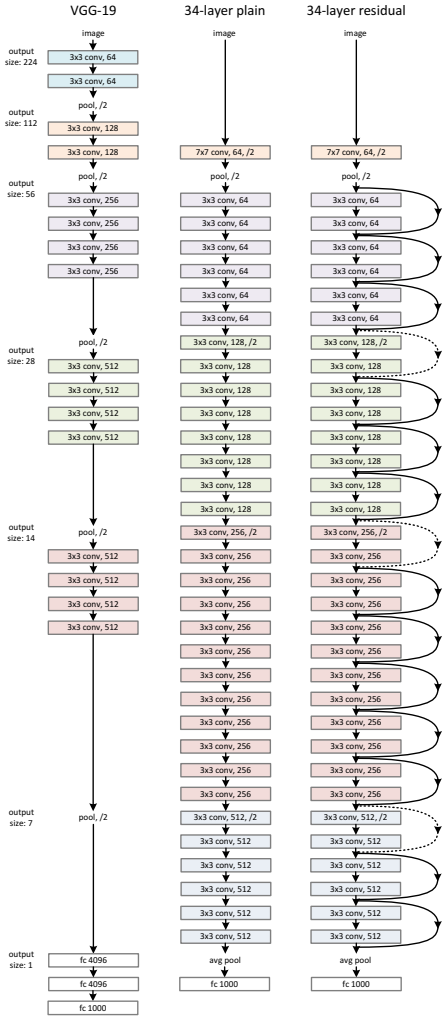
- Deeper networks, with computational efficiency
- 22 layers - Efficient “Inception” module
- No fully connected layers layers
- Only 5 million parameters!

ResNet

The identity connection permits an unmodified version of the input to pass through the cell. This modification allows for the effective training of very deep convolutional architectures.

Only 1 Fully Connected Layer for class output at end





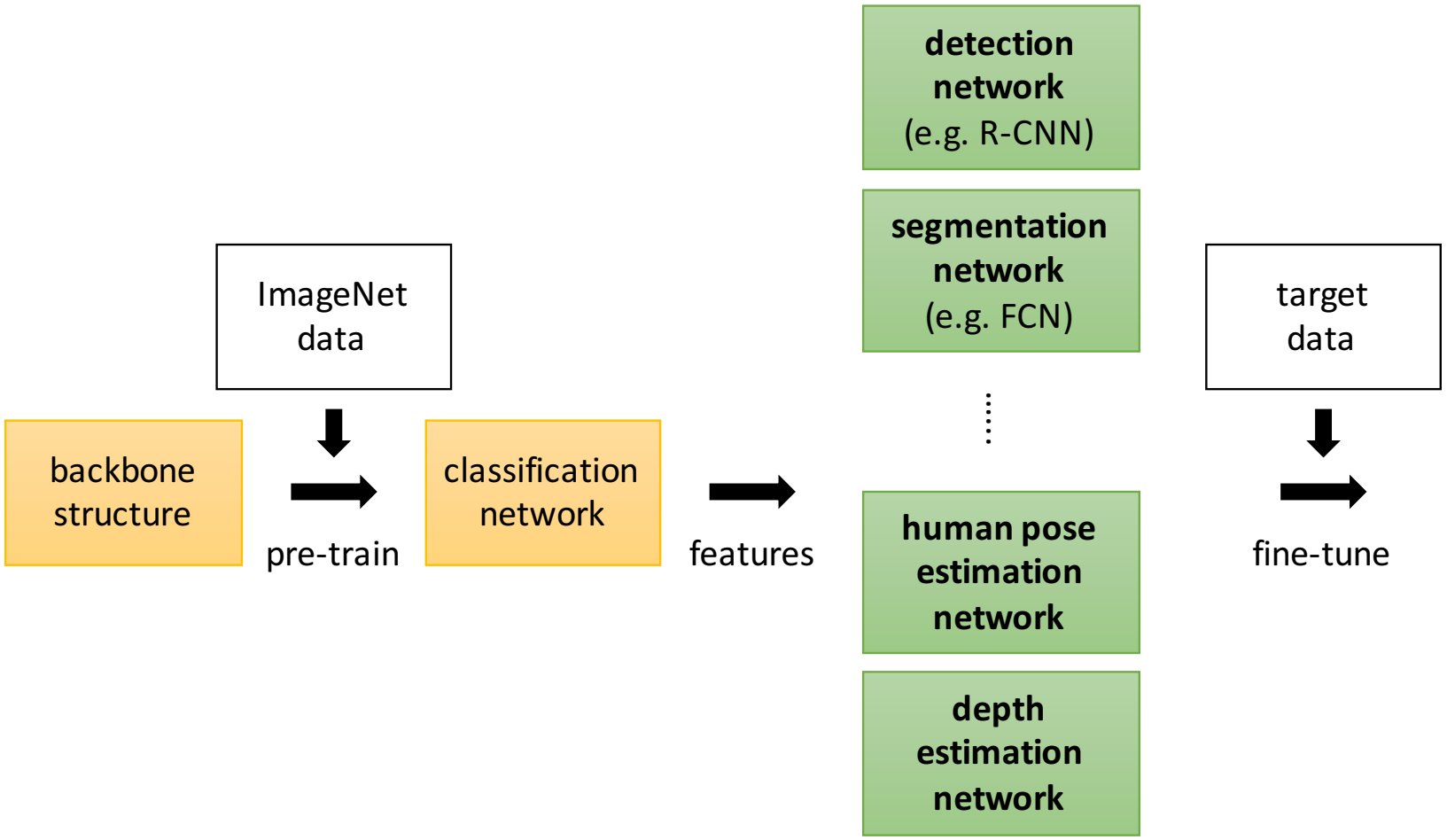
Kaiming He, Deep Residual Learning for Image Recognition, https://arxiv.org/pdf/1512.03385.pdf, 2015.
Zadeh, Tensorflow for Deep Learning, O'Reilly, 2018.

Convolutional Neural Network Architectures.

Network	Number Parameters	Number Layers	ImageNet Top 5 Error
AlexNet (2012) [7]	60 mio.	8 (5 convolutional, 3 fully connected)	15.3 %
GoogLeNet (2014) [65], [66]	5 mio.	22	6.7 %
VGG (2014) [67]	∼140 mio.	19 (16 convolutional, 3 fully connected)	7.3 %
Inception v3 (2015) [68]	25 mio.	42	3.58 %
Deep Residual Learning (2015) [12]	∼60 mio.	152	3.57 %

Luckow et al, Deep Learning in Automotive Industry, https://arxiv.org/abs/1705.00346, 2017.

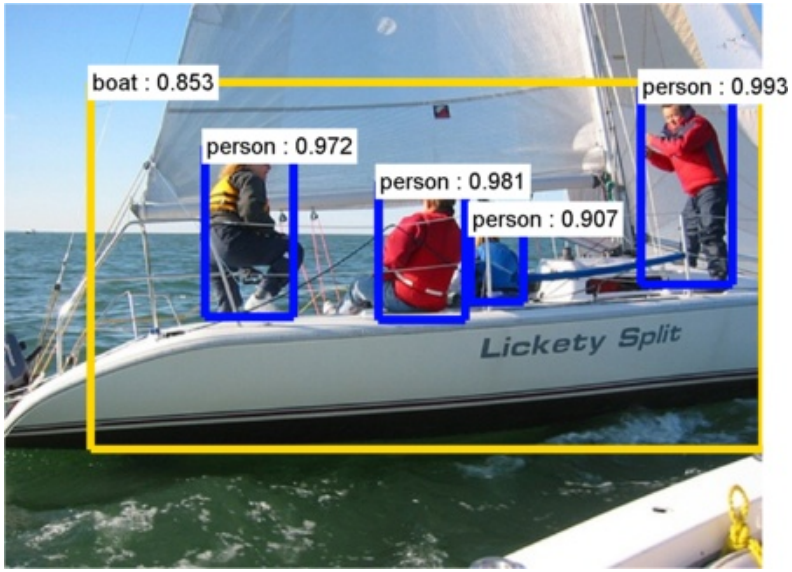
Further Computer Vision Tasks



Kaiming He, https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf, 2016.

Object Detection

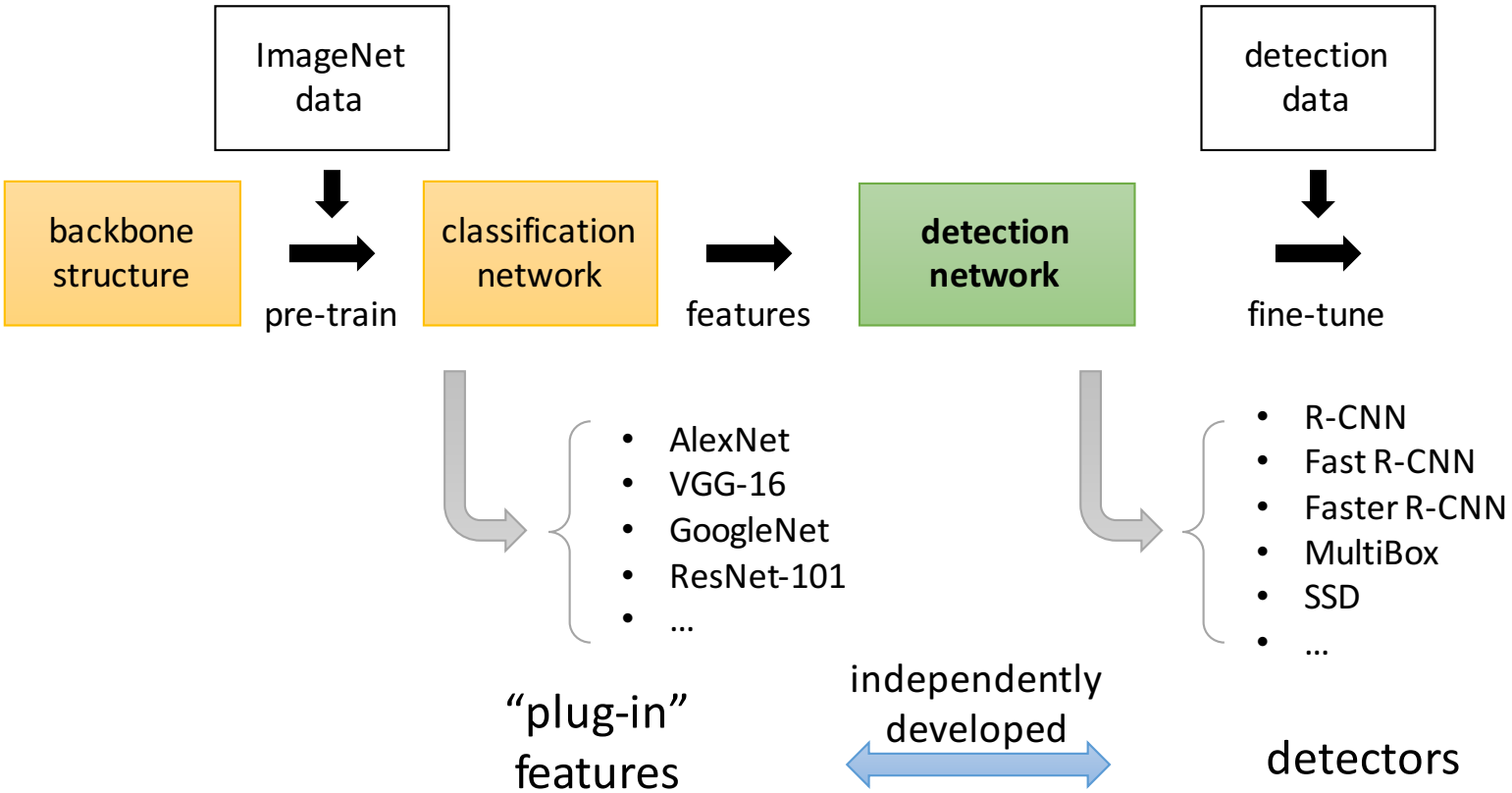
Image Classification
(what?)



Object Detection
(what + where?)

Kaiming He, https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf, 2016.

Object Detection



Kaiming He, https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf, 2016.

Object Detection Networks.

Network	Frameworks	Architecture
DetectNet	Caffe	Based on GoogLeNet. Two loss functions for predictions: object class and bounding box. Clustering of proposed bounding boxes.
Faster-RCNN	Matlab, Python, MXNet, CNTK	Based on VGG. Multiple components: Regional proposal network (class agnostic), classification and regression network. End-to-end trained using a single CNN.
Single Shot Detector (SSD)	Caffe, Tensorflow	Based on VGG. Eliminates bounding box proposal. Predicts for a fixed set of boxes: object class and offset to box.
You Only Look Once (Yolo9000)	DarkNet, Turi-Create, Tensorflow	Based on a high-resolution AlexNet. Direct prediction of bounding boxes (predict offsets). Clustering of boxes using K-Means.

Table 1: Object Detection: Architectures and Frameworks

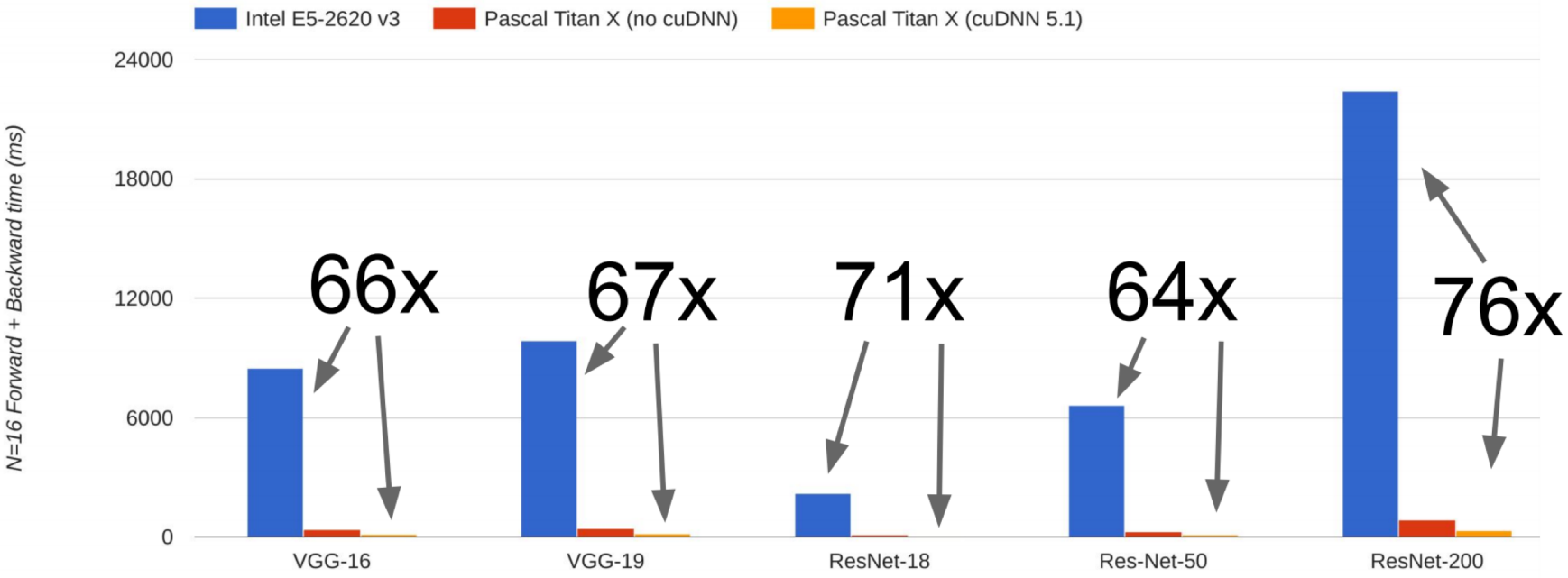
Training Deep Neural Networks: CPU vs. GPU

	# Cores	Clock Speed	Memory	Price
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.4 GHz	Shared with system	$339
CPU (Intel Core i7-6950X)	10 (20 threads with hyperthreading)	3.5 GHz	Shared with system	$1723
GPU (NVIDIA Titan Xp)	3840	1.6 GHz	12 GB GDDR5X	$1200
GPU (NVIDIA GTX 1070)	1920	1.68 GHz	8 GB GDDR5	$399

- CPU: Fewer cores, but each core is much faster and much more capable; great at sequential tasks
- GPU: More cores, but each core is much slower and “dumber”; great for parallel tasks
- Deep Learning Frameworks can run computational graphs on GPUs
- Compute-intensive linear algebra operations can easily be pushed to GPUs

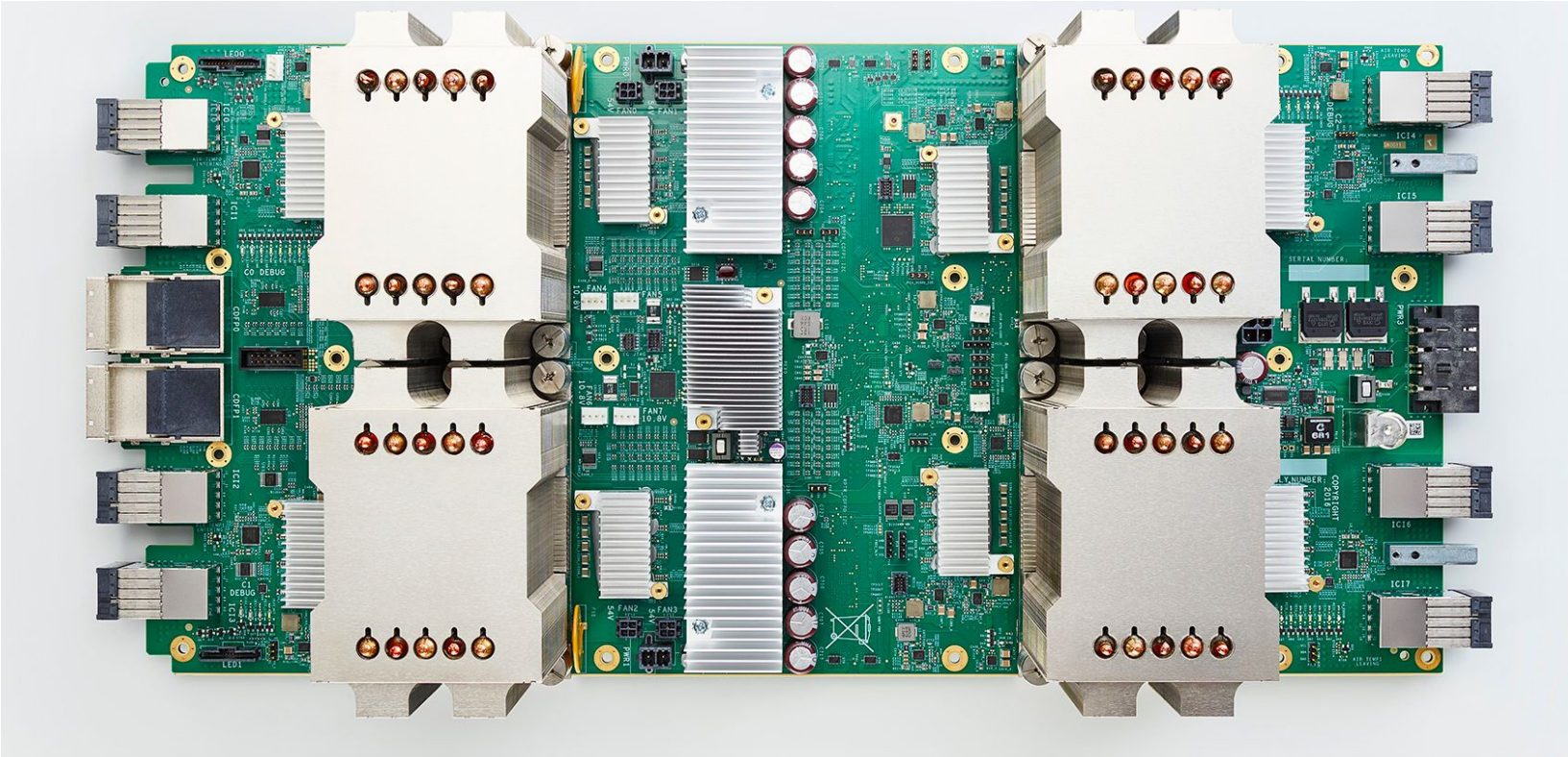
CPU vs GPU in practice

(CPU performance not well-optimized, a little unfair)



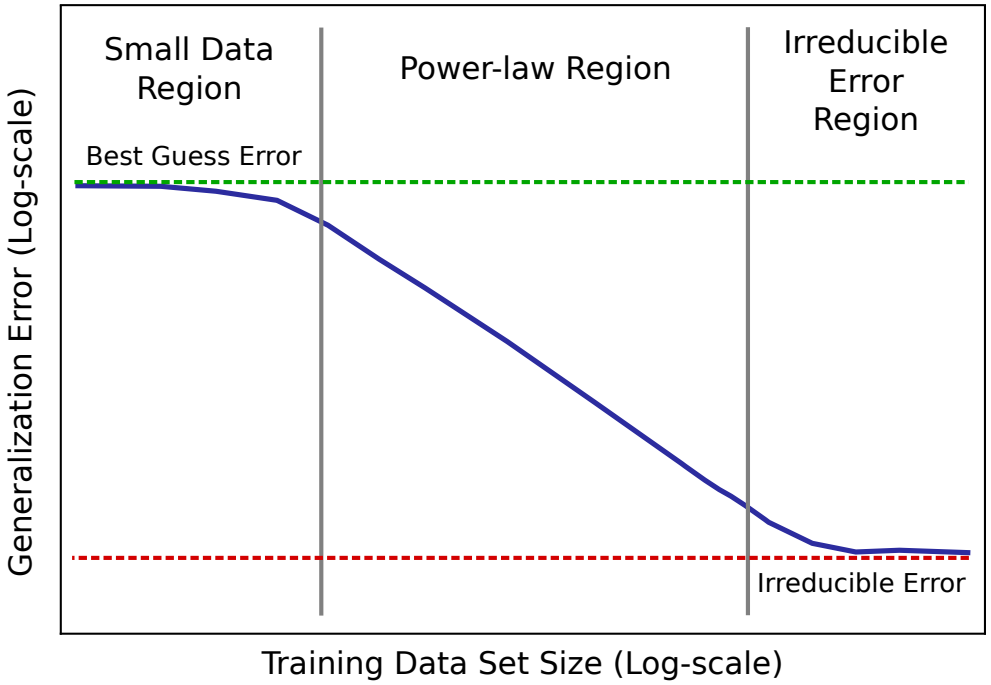
Data from https://github.com/jcjohnson/cnn-benchmarks

Training Deep Neural Networks: Tensor Processing Unit



- TPU V1: Only inference
- TPU V2: 180 teraflops of computation, 64 GB memory, 2400 GB/s mem BW
- Tensor Processing Unit only available for rent in Google Cloud

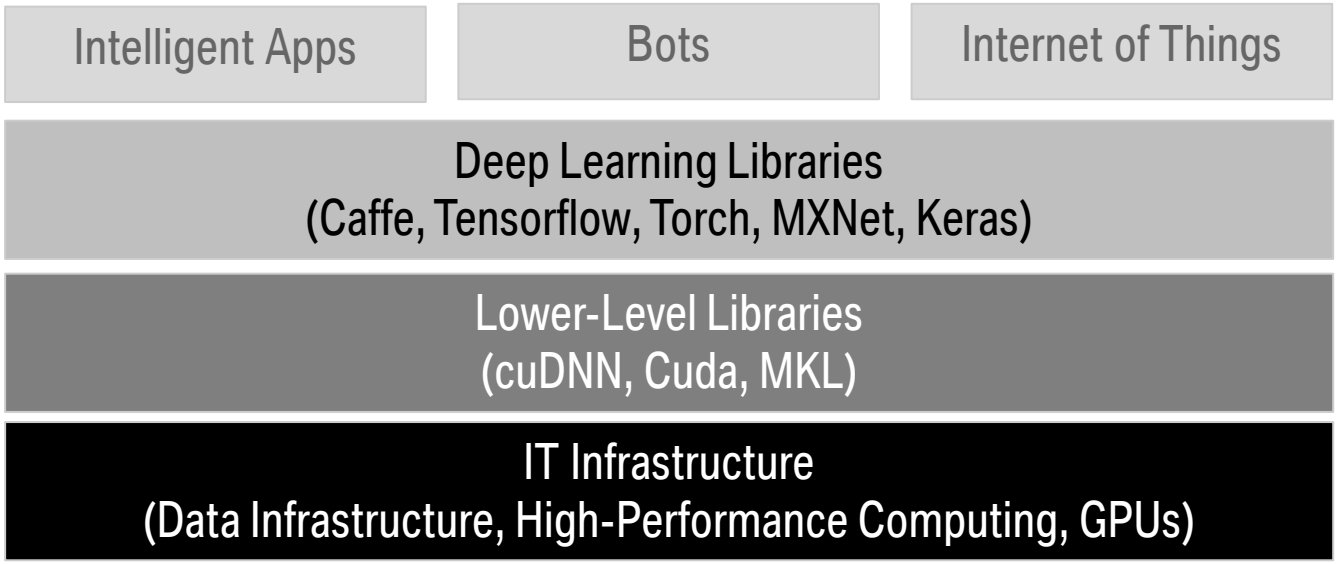
Training Deep Neural Networks: Data



More Data improves accuracy better than trial-and-error model tuning.

Joel Hestness, Deep Learning Scaling is Predictable, Empirically, https://arxiv.org/pdf/1712.00409.pdf, 2017

Scalable Infrastructure for Deep Learning.

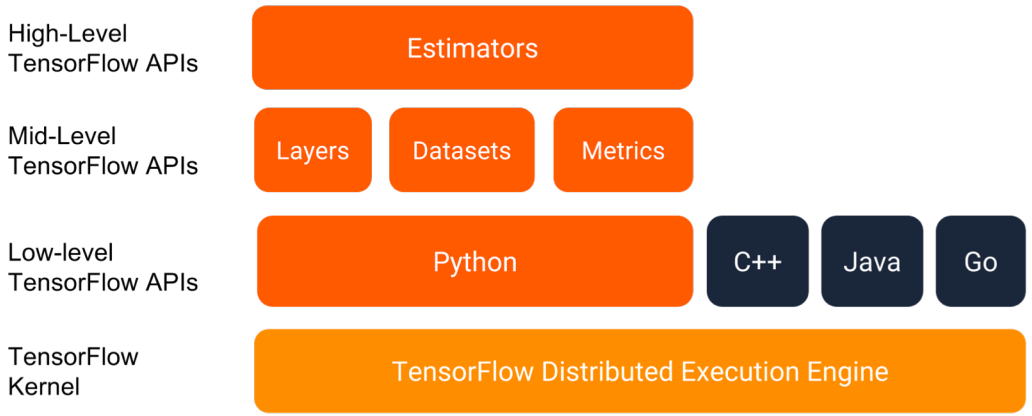


Deep Learning Frameworks

- Main Objectives:
  - Build computational graphs
  - Run on GPUs
  - Compute gradients
- Examples:
  - Caffe
  - Tensorflow
  - PyTorch
  - MXNet

Tensorflow

- Open Source Deep Learning library developed by Google and released in Nov. 2015.
- Provides the ability to define tensor and functions on these tensors
- Provides abstractions on different levels! In addition high-level libraries, such as Keras use Tensorflow



https://www.tensorflow.org/get_started/get_started_for_beginners

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

```
res50 = keras.applications.resnet50.ResNet50(include_top=True,
weights='imagenet', input_tensor=None, input_shape=None, pooling=None,
classes=1000)
img_path = 'sky.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
preds = res50.predict(x)
```

https://keras.io/

- Imperative abstraction based on dynamic computational graphs allows easier development.
- PyTorch Tensors are just like numpy arrays, but they can run on GPU.

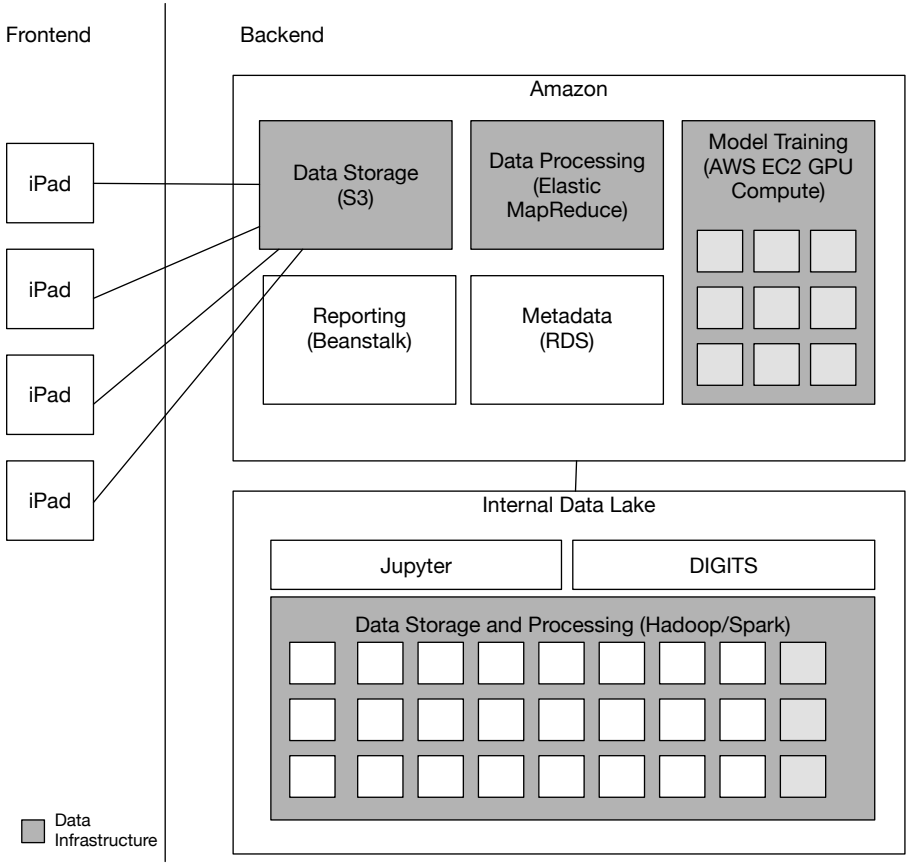
```
In [53]: %%time
         # Initialize the pre-trained model
         vgg19 = models.vgg19(pretrained=True)

CPU times: user 53.9 s, sys: 352 ms, total: 54.3 s
Wall time: 54.4 s
```

```
In [48]: %%time
         # Initialize the pre-trained model
         resnet50 = models.resnet50(pretrained=True)

CPU times: user 9.26 s, sys: 475 ms, total: 9.73 s
Wall time: 13.6 s
```

Characteristic	Description
Problem Type	Object Classification, Detection, Segmentation
Training Data	Real data, synthetic data, GAN, online/human-in-the-loop, public/private
Continuous Data	Process for improving initial model using data generated by the running systems
Off-the-shelf	Availability of integrated solutions
Custom Implementation	Framework: Tensorflow, Pytorch, CNTK; Model Architecture, Re-use of standard networks (ResNet, Inception) and pre-trained models (transfer learning).
Model Deployment	Cloud, edge or hybrid. Required model, inference latencies
Model Management	adhoc, model management system (e.g. ModelDB)
Realtime Training	adhoc training required. Edge vs. cloud

Table 2: Framework for Characterizing AI Application

Data Collection



Deep Learning Tool: DIGITS

References.

- LeCun, Bengio, Hinton, Deep Learning, Nature, 2015
- Krizhevsky et al., ImageNet Classification with Deep Convolutional Neural Networks, 2012
- Ujjwal Karn, An intuitive explanation of convolutional neural networks, 2016
- Alex Krizhevsky, One weird trick for parallelizing convolutional neural networks, 2014
- Abadi et al., TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, White Paper, 2015.
- Seide et al., CNTK: Microsoft's Open Source Deep-Learning Toolkit, White Paper, 2016.
- Chen et al., MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems, 2015.

There are only two kinds of languages: the ones people complain about and the ones nobody uses —Bjarne Stroustrup

Scalable Machine Learning.

Parallelization Approaches

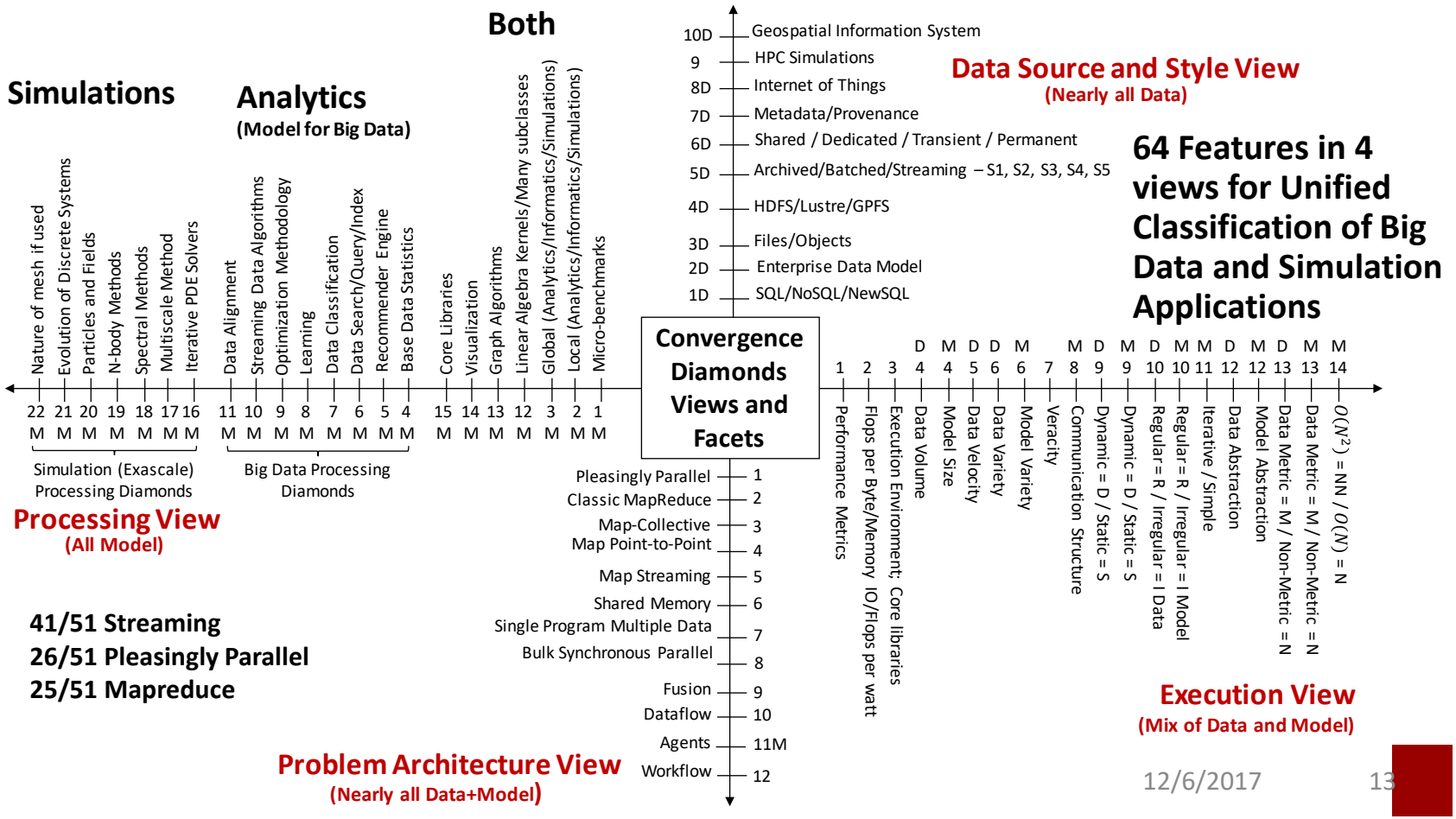
Distributed Deep Learning

Limitations of Machine Learning Libraries.

- Limitations of Python and R standard libraries:
  - Model has to fit into memory of a single machine
  - Computational resources on a single machine constrained
- Mitigations:
  - Some libraries provide disk-based extensions for datasets that cannot be fitted into memory
  - Some algorithms provide thread-based parallelism to exploit multiple cores on a single machines
- Nevertheless, often long training times due to limitations in available compute resources
- Distributed libraries needed for :
  - To store data into memory of multiple machines
  - Provide more compute resources for training

- Machine Learning (ML) needs high performance computing: Big Data and Big Model
- **All the different programming models** (Spark, Flink, Storm, Naiad, MPI/OpenMP) have the **same high level approach:**
  - First: Break Problem **Data** and/or **Model-parameters** into parts assigned to separate nodes, processes, threads
  - Then: In parallel, do computations typically leaving data untouched but changing model-parameters.
- Called **Maps** in MapReduce parlance; typically **owner computes rule**.

Big Data Ogres and Facets

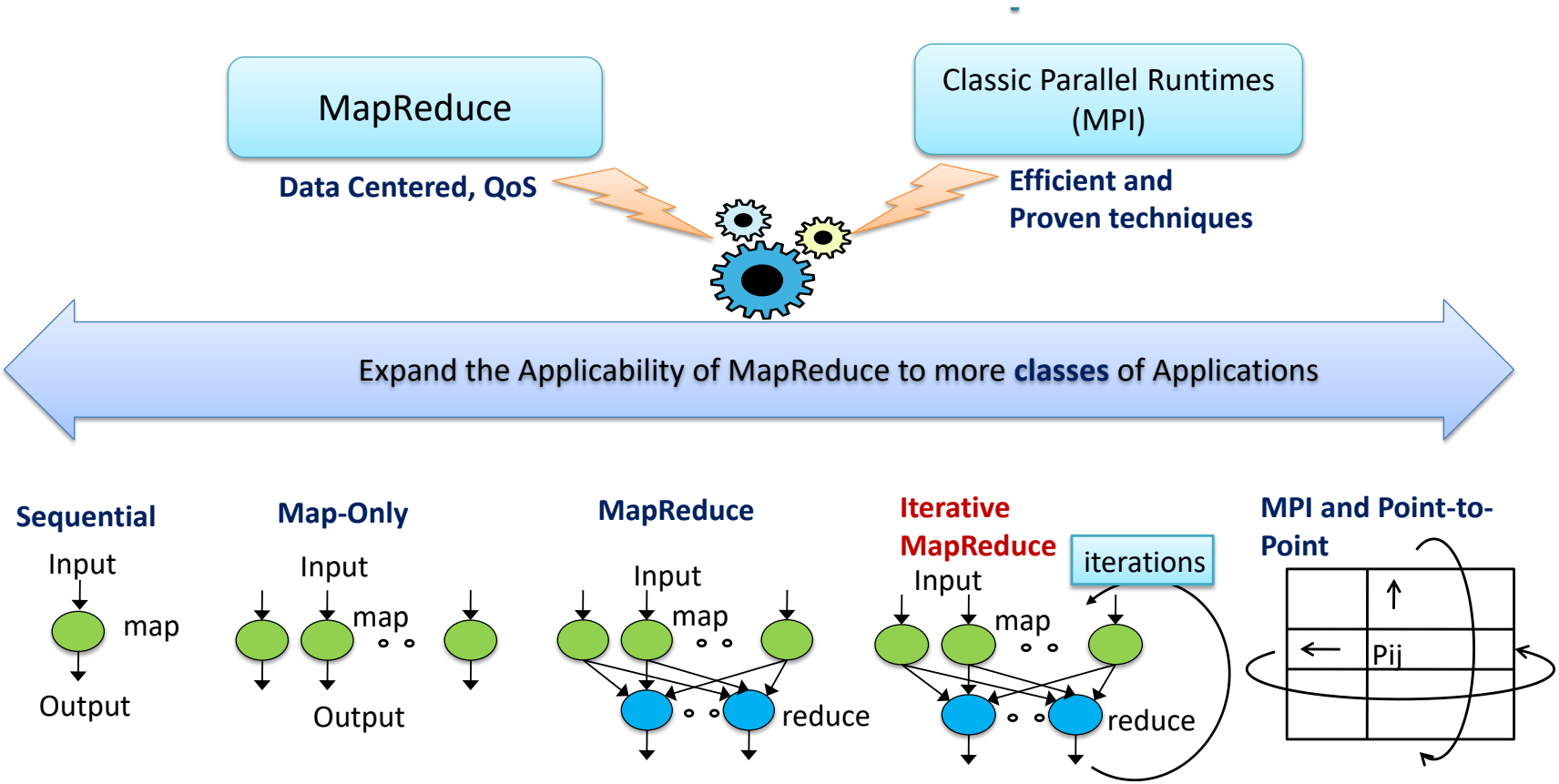


Problem Architectures

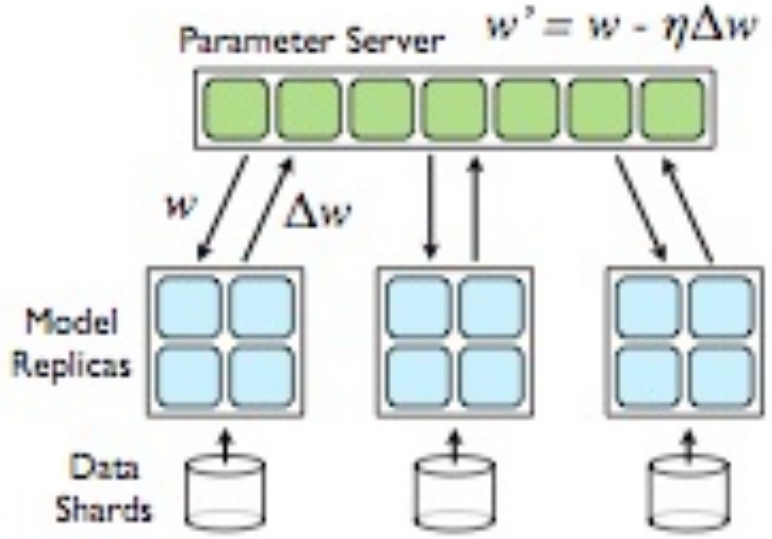
- **Pleasingly Parallel** – as in BLAST, Protein docking, some (bio-)imagery including Local Analytics or Machine Learning – ML or filtering pleasingly parallel, as in bio-imagery, radar images (pleasingly parallel but sophisticated local analytics)
- **Classic MapReduce:** Search, Index and Query and Classification algorithms like collaborative filtering
- **Map-Collective:** Iterative maps + communication dominated by "collective" operations as in reduction, broadcast, gather, scatter. Common datamining pattern
- **Map-Point to Point:** Iterative maps + communication dominated by many small point to point messages as in graph algorithms
- **Map-Streaming:** Describes streaming, steering and assimilation problems
- **Shared Memory:** Some problems are asynchronous and are easier to parallelize on shared rather than distributed memory
- **SPMD:** Single Program Multiple Data, common parallel programming feature
- **BSP or Bulk Synchronous Processing:** well-defined compute-communication phases
- **Fusion:** Knowledge discovery often involves fusion of multiple methods.
- **Dataflow:** Important application features often occurring in composite Ogres
- **Workflow:** All applications often involve orchestration (workflow) of multiple components

- Shortcomings of Hadoop/MapReduce for scalable Machine Learning:
  - Hadoop writes to disk frequently (e.g. during shuffle and between multi-stage jobs)
  - Spark and Flink spawn many processes and do not support allreduce directly;
  - MPI does in-place combined reduce/broadcast
- **Iterative algorithms** are fundamental in learning a non-trivial model
- **Model training** and **Hyper-parameter tuning** steps run the iterative algorithms many times

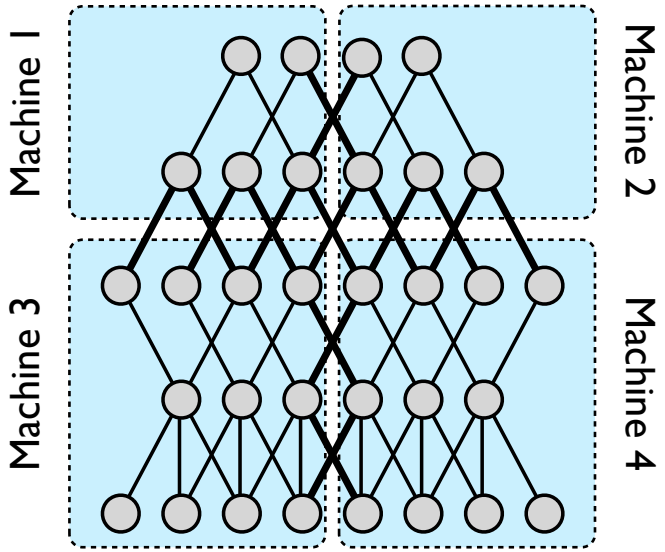
Iterative MapReduce.



Judy Qui, Scalable High Performance Data Analytics: Harp and Harp-DAAL, http://dsc.soic.indiana.edu/presentations/Qiu_Harp_DAAL_June_27_2017.pdf, 2017



Data Parallelism

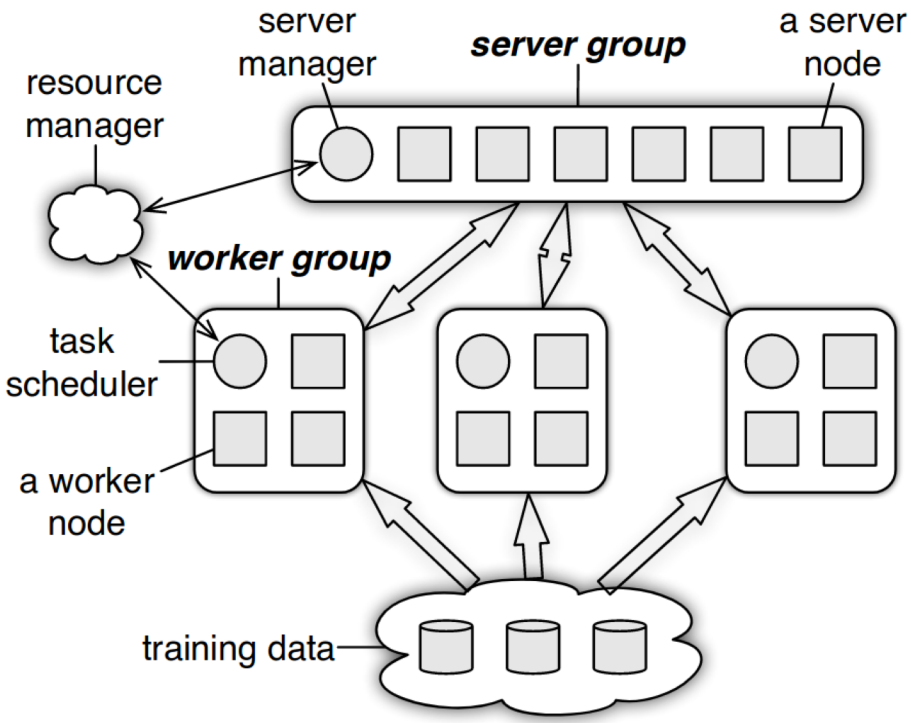


Model Parallelism

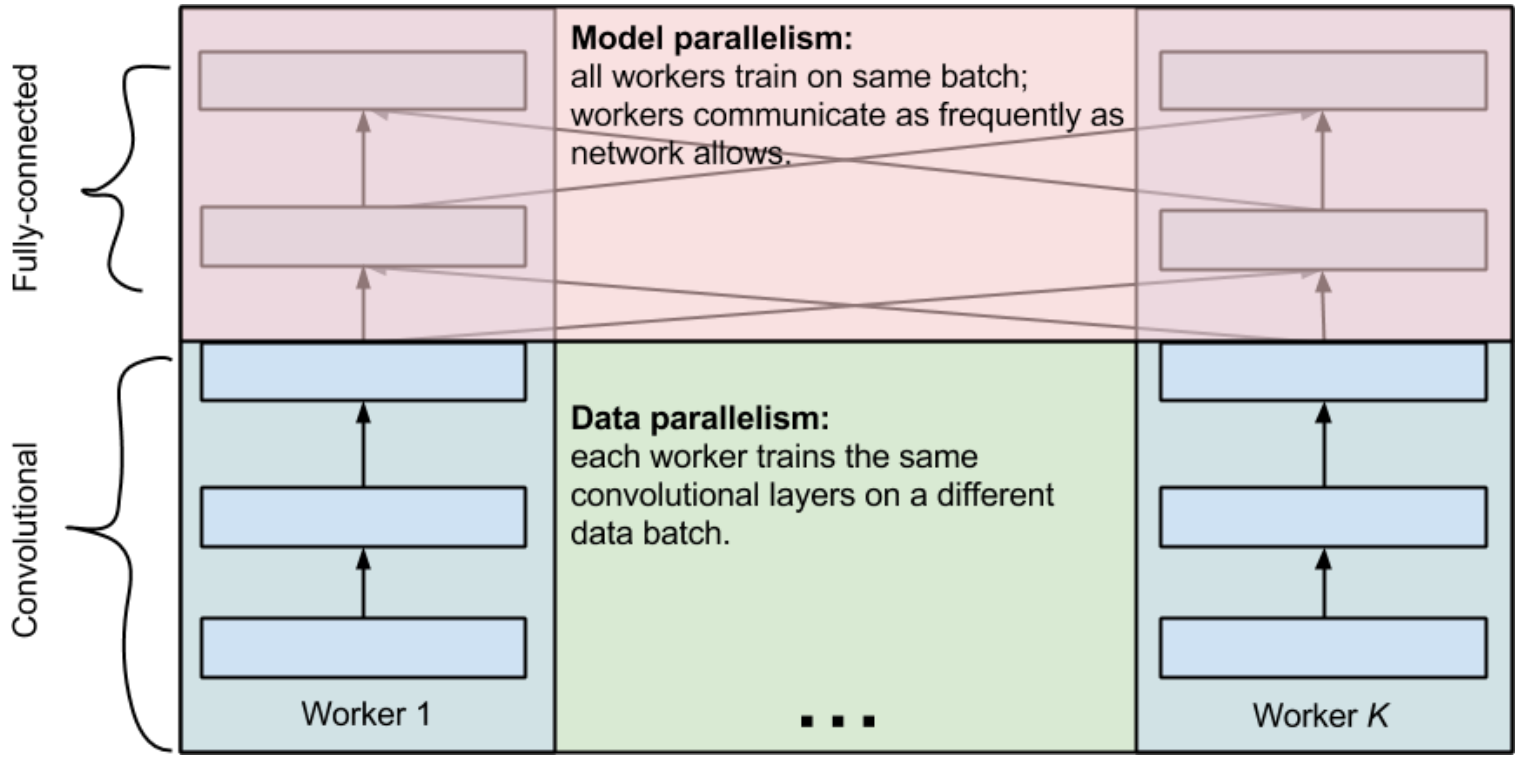
Source: Dean et al., Large Scale Distributed Deep Networks, https://static.googleusercontent.com/media/research.google.com/en//archive/large_deep_networks_nips2012.pdf, 2012

Parameter Server

- Abstraction used for managing models across multiple worker processes
- Used for exploring data parallelism
- Properties:
  - Management of Worker nodes
  - Synchron vs. Asynchronous mode
  - Distributed vs. Non-Distributed



Example AlexNet



MLLib

- Spark-based Machine Learning Library
  - Classification: logistic regression, naive Bayes, random forest, …
  - Collaborative filtering: ALS, …
  - clustering: k-means
  - Principal Component Analysis: SVD
- Spark In-Memory capabilities ideally suited for iterative machine learning implementation.

References

- Li et al., Scaling Distributed Machine Learning with the Parameter Server, OSDI, 2014
- Xing et al., Petuum: A New Platform for Distributed Machine Learning on Big Data, KDD, 2015
- Meng et al., MLlib: Machine Learning in Apache Spark, Journal of Machine Learning Research, 2016
- H2O, H2O version 3, 2017.