Faculty of Computer Science
Database and Software Engineering Group

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

AI Techniques for Database Management (AI4DB)

Gabriel Campero Durand

Advanced Topics in Databases, 2019/June/28
Otto-von-Guericke University of Magdeburg

We will talk about

- Emerging configurations for using AI in production systems
- General overview of recent AI areas of application in data management
  - In-database machine learning
  - Examples of implementing database internals with AI
    - Learned index structures and deep hashing
  - Self-managing data solutions with AI techniques

Goals :

- By the end of this lecture, you:
  - Will have basic knowledge about considerations for deploying AI in production systems
  - Will have a better understanding on existing AI areas of application in data management, facilitating your entry to collaborating with us in this research field
  - Will know the state of the art in using AI for database internals, and for self-managing database components
  - Will hopefully be encouraged to research with us, in the area of AI4DB.

Agenda

- Part 1: Introduction (1st lecture)
- Part 2: Recent Applications of AI Techniques in Data Management (1st and 2nd lecture)
- Part 3: Open Directions/Research Interests (2nd lecture)

Part 1. Introduction

Part 1. Introduction

Sec 1. Recap and Driving factors for the adoption of AI

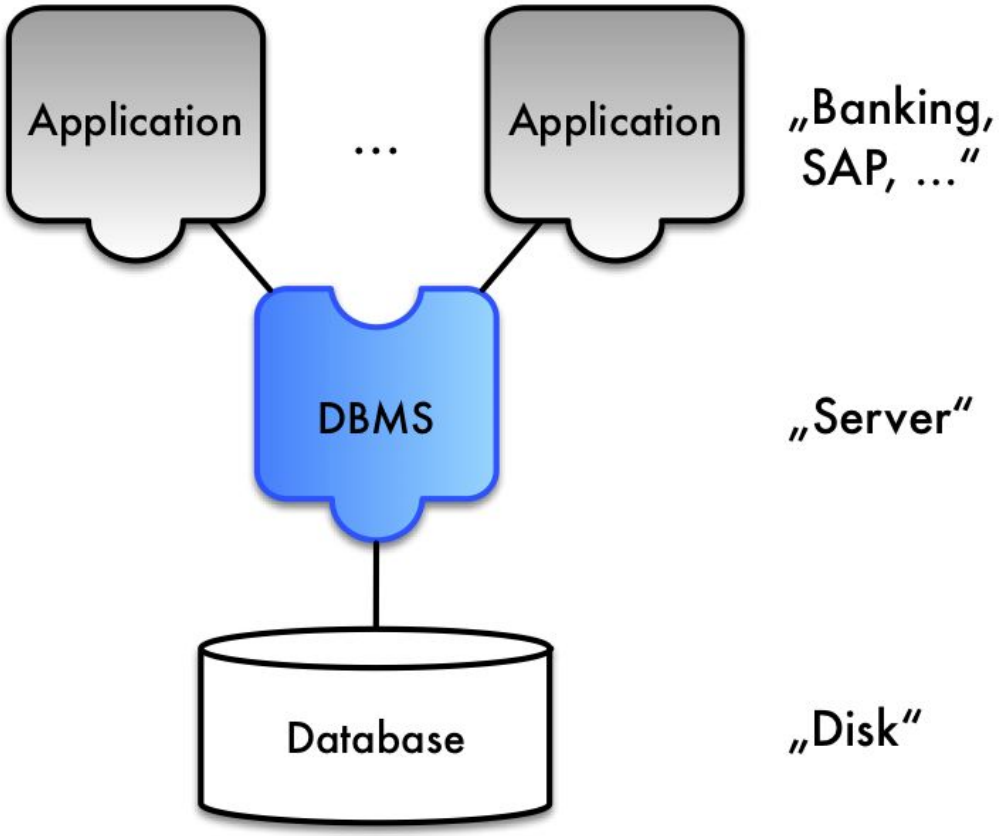
Yesterday's DBMS Landscape

ORACLE

IBM DB2

Microsoft SQL Server

TERADATA



Yesterday's DBMS Hardware

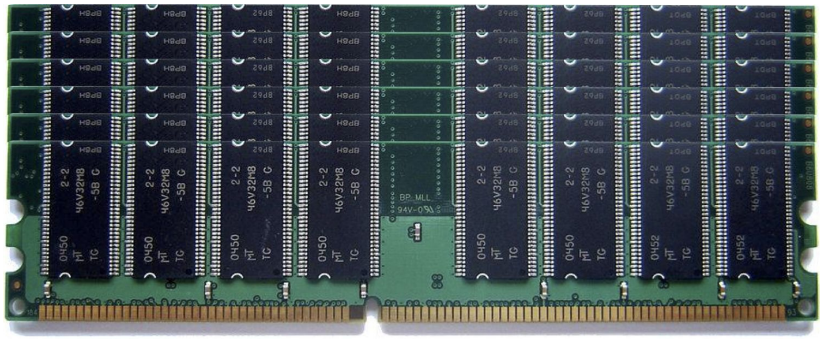
Picture taken from [1]

Small main memory

Picture taken from [2]

Disk-based systems

Today's DBMS Hardware



Picture taken from [1]

Large main memory

Picture taken from [3]

Multi-core CPUs

Picture taken from [4]

Solid state disks

Picture taken from [5]

Co-processors

Today's DBMS Hardware

And there is much more:

- Networking (RDMA on InfiniBand)
- Memory (NVM, Far/disaggregated memory)
- Processors (TPUs)
- Custom hardware
- …

Today's DBMS Infrastructure

- Large-scale query/data flow engines
- Stream-based query engines
- **In-Memory Storage**
- MPP DBs, cloud EDWs, **GPU DBs**
- NewSQL: Large-scale OLTP and **HTAP** DBs
- NoSQL: Column-families, graph data, key-value stores, **documents**, time series, etc.
- Specialized data transformation

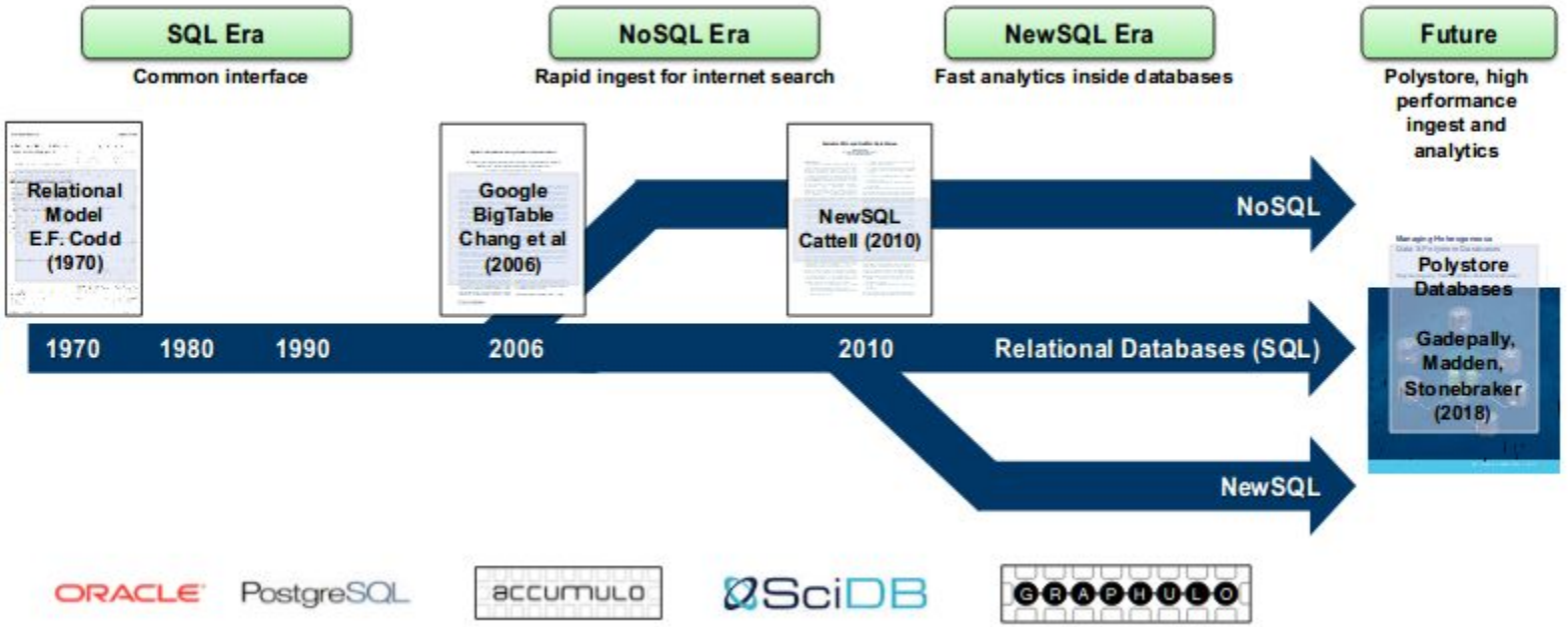
Big Data Landscape 2019- Pictures taken from [14]

Today's DBMS Analytics

- Statistical analysis and Data science workloads backed by DBs
- Interactive visual data exploration & BI tools
- Specialized ML systems with their own data solutions
- Search engines
- Web, Commerce, Social and Log analytics
- Speech and NLP

Big Data Landscape 2019- Pictures taken from [14]

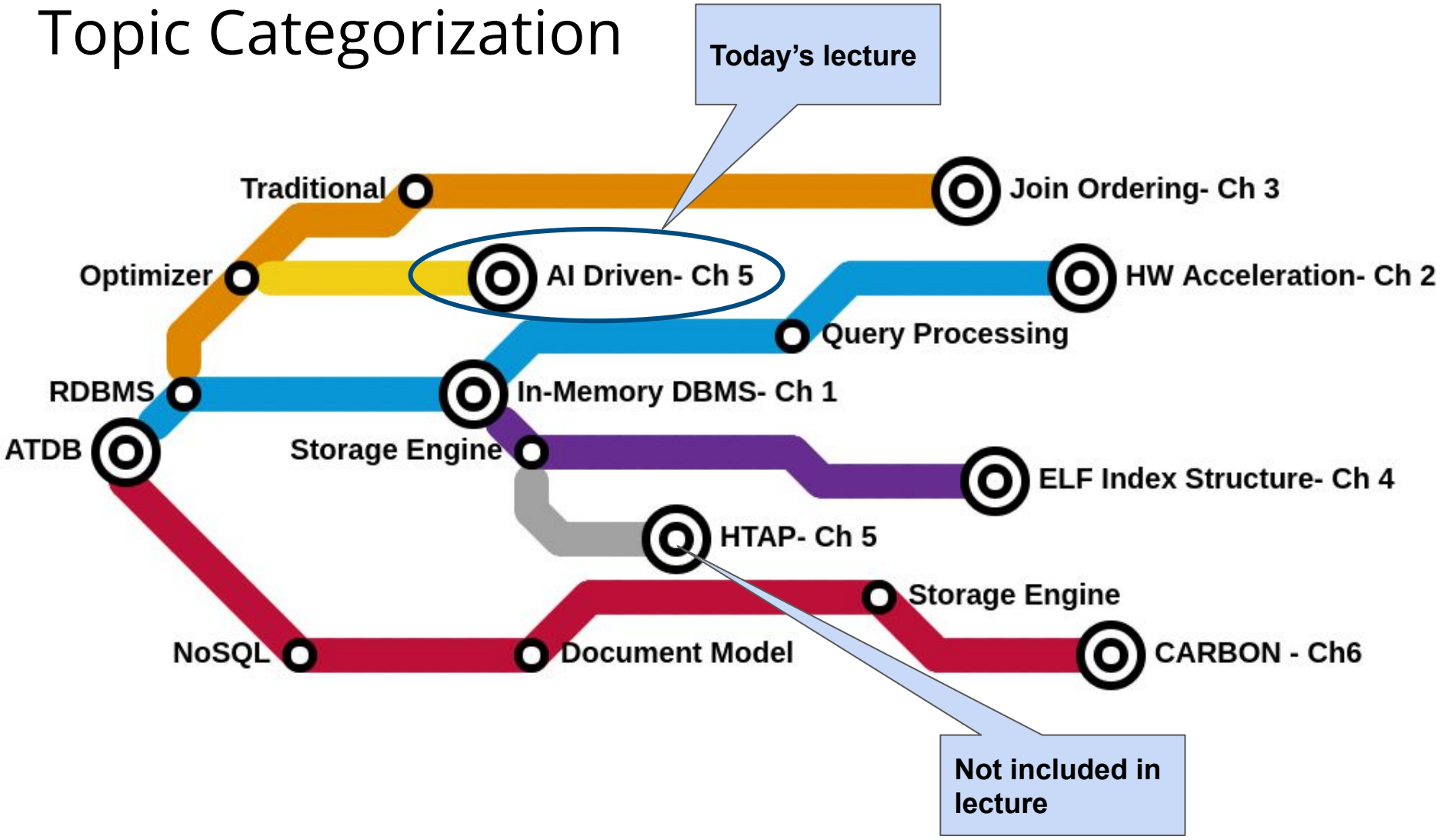
Database Trends



[GGK19]

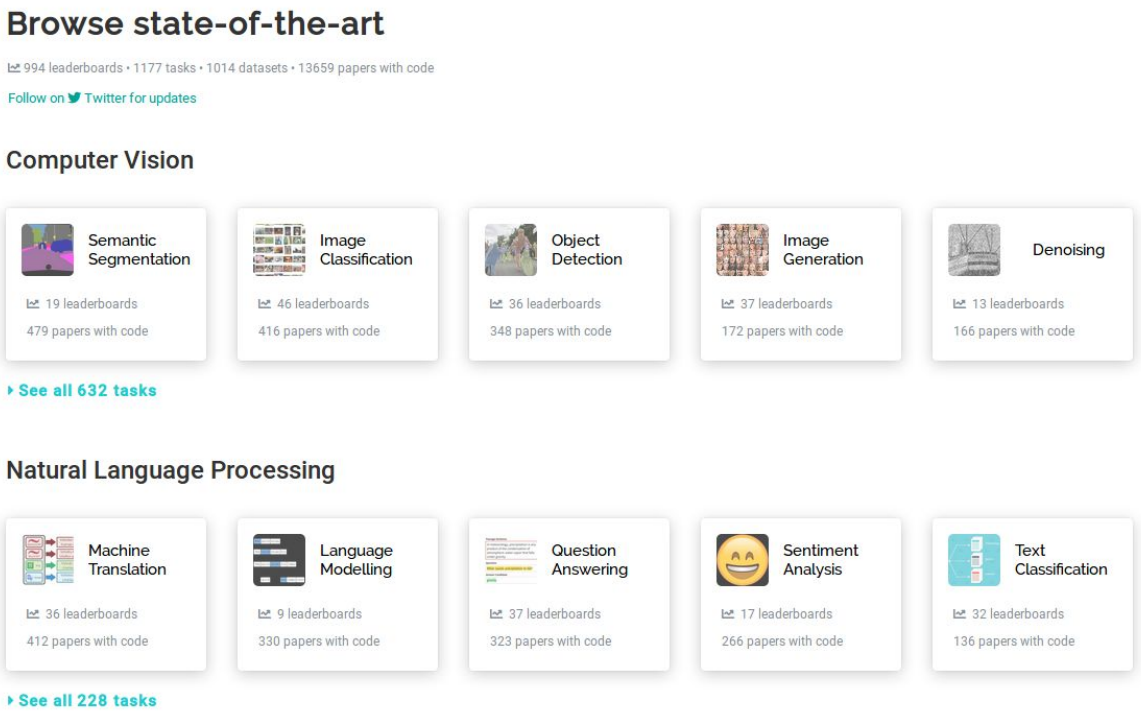
Databases have evolved towards more scalability and data variety

Topic Categorization



Driving factors for the increased adoption of AI

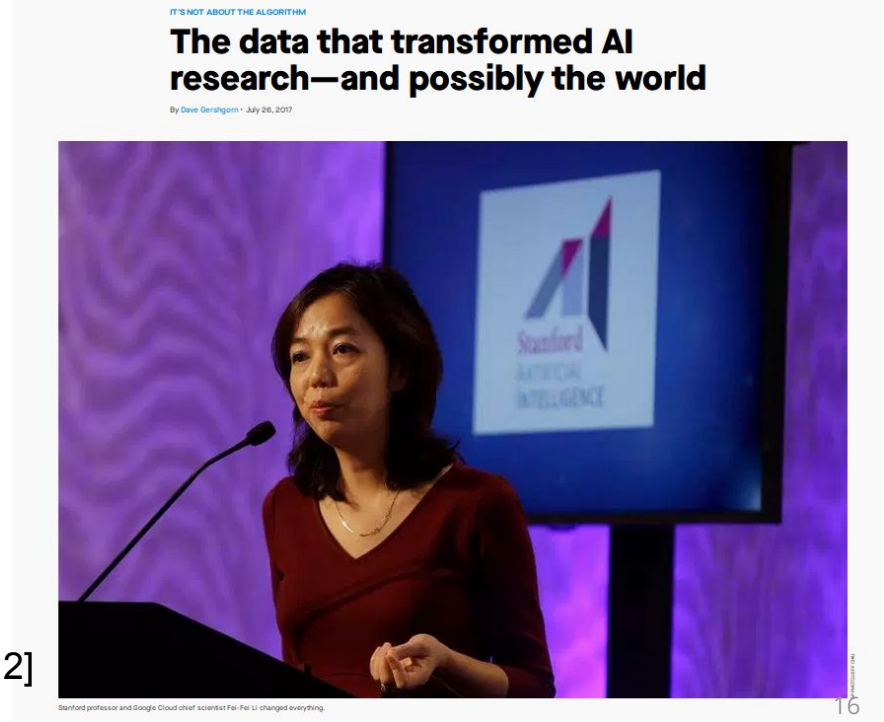
- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks, and in commercial products (e.g. voice assistants, recommenders)



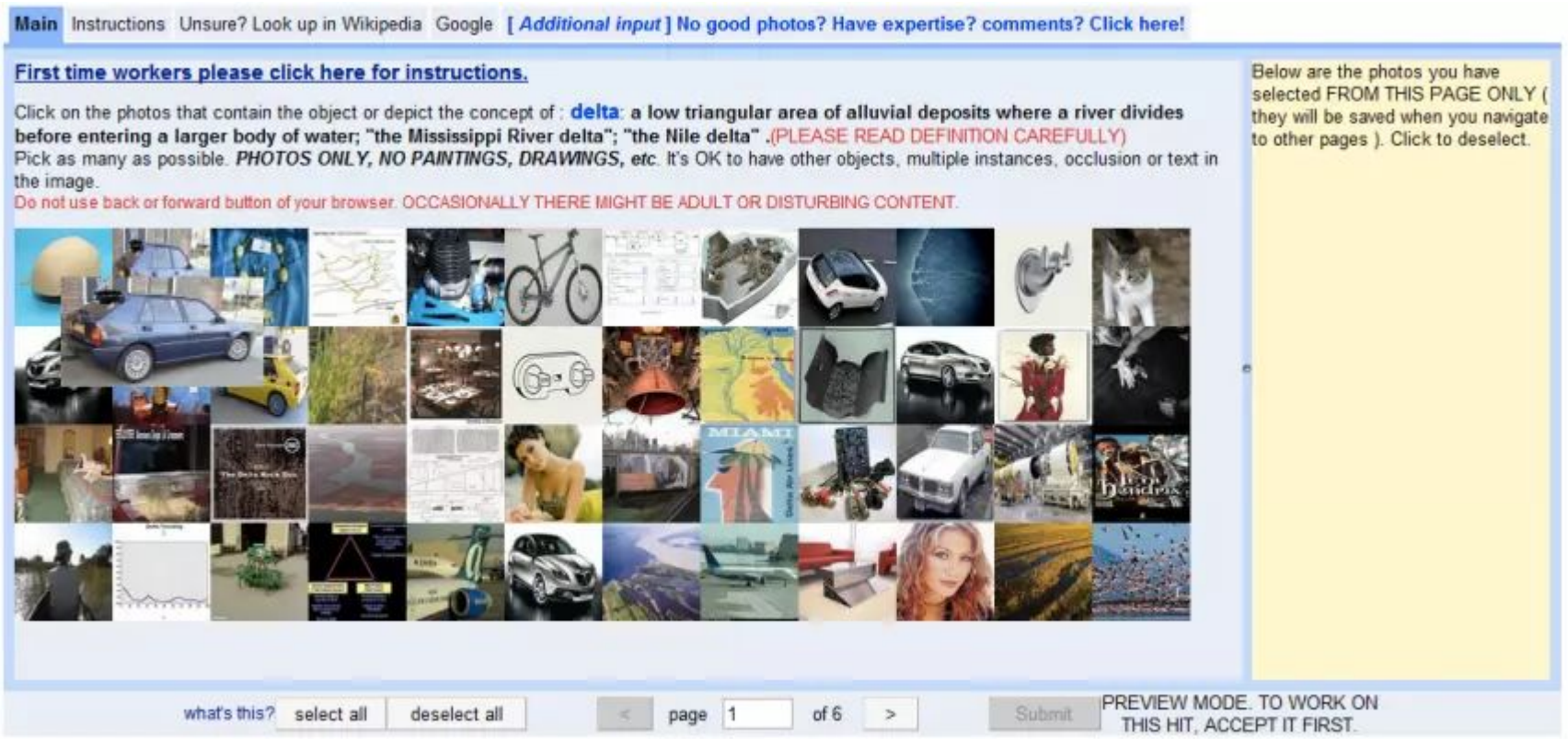
Leaderboards of models, at different tasks [6]
Similar repository in [7]

Driving factors for the increased adoption of AI

- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks
  2. **Technical progress** in data and algorithms
     a. Some examples:
        Large-scale challenging datasets and competitions

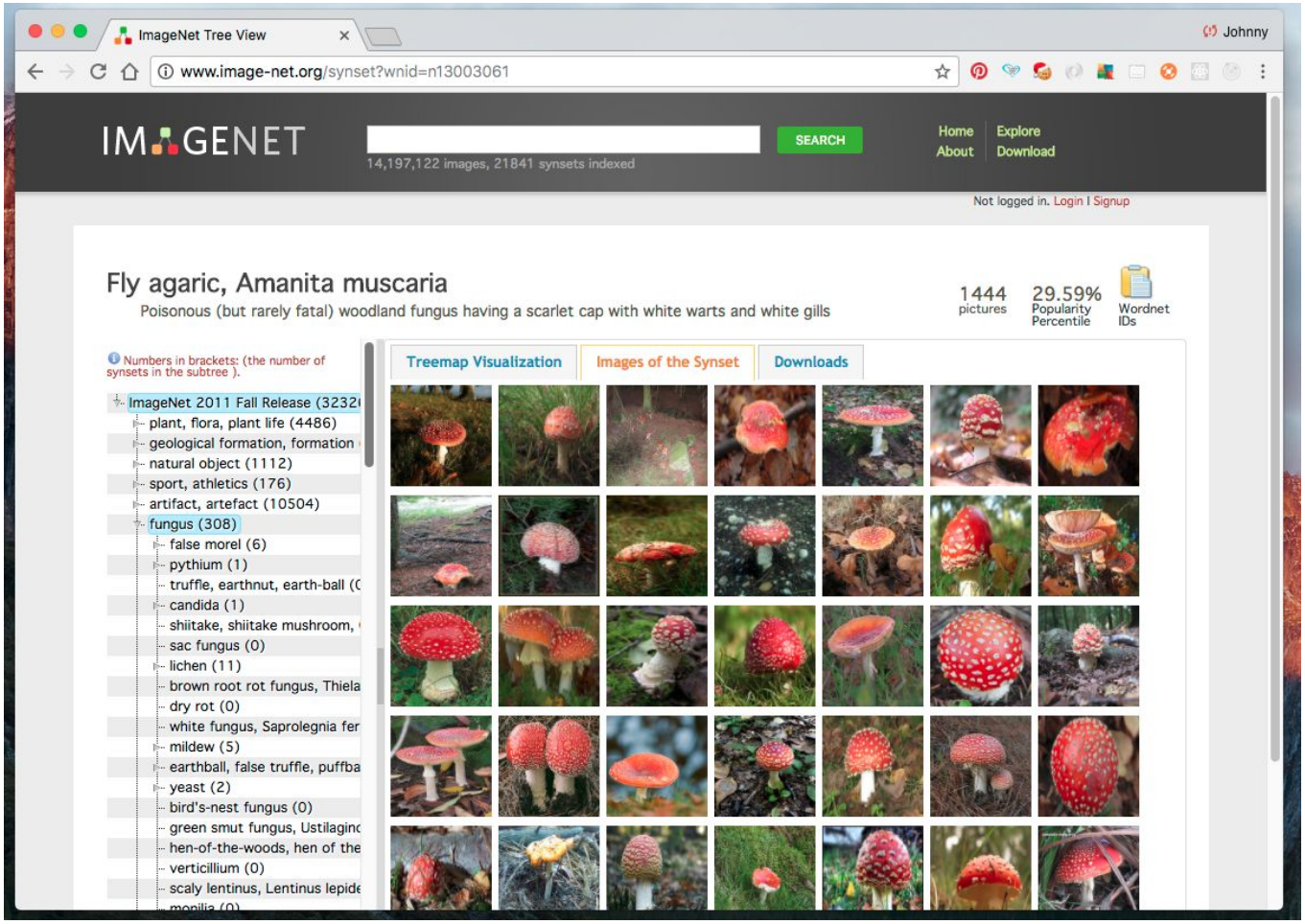


Picture taken from [12]



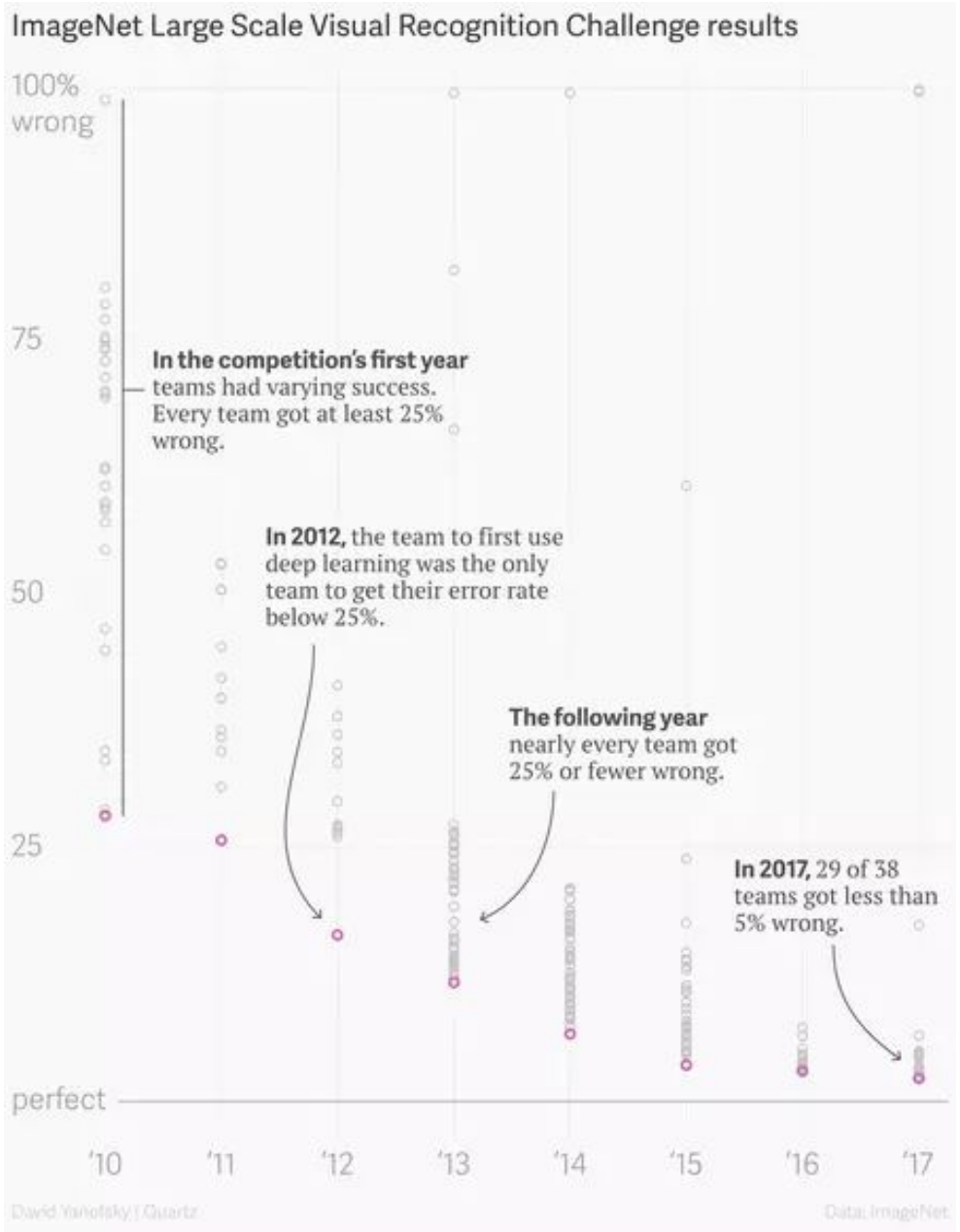
Picture taken from [12]

ImageNet crowd-sourced through Amazon's Mechanical Turk the task of image labeling.



Picture taken from [12]

ImageNet currently offers verified labels for around 14M images in 20k categories



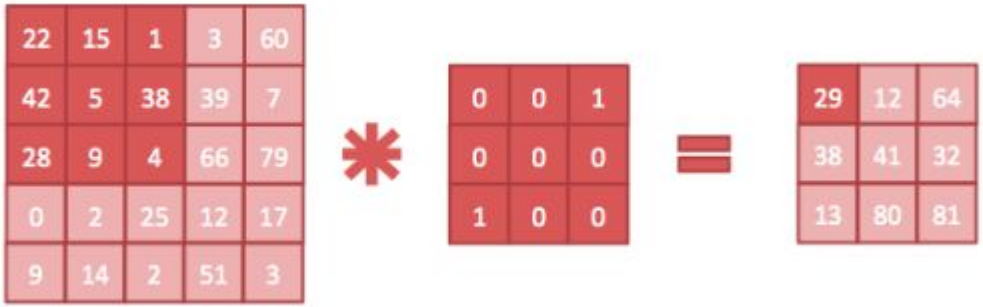
Challenging datasets and competitions are essential to the progress of fields

Throughout the years ImageNet competitors have advanced the state of the art (SOTA) in visual recognition.

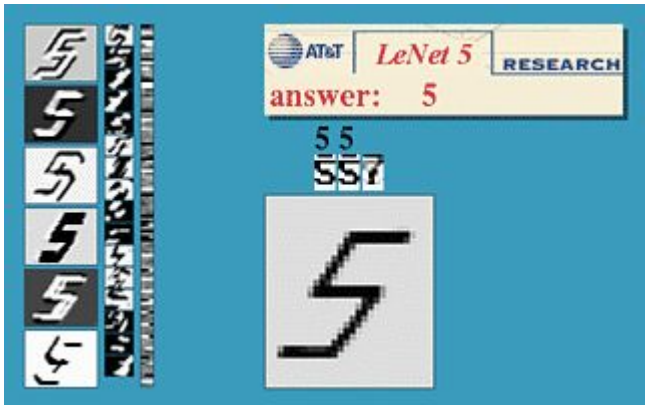
Picture taken from [12]

Driving factors for the increased adoption of AI

- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks
  2. **Technical progress** in data and algorithms
     a. Some examples:
        GPU-accelerated convolutional neural networks [S15]



A simple 2D convolution operation [WR17]

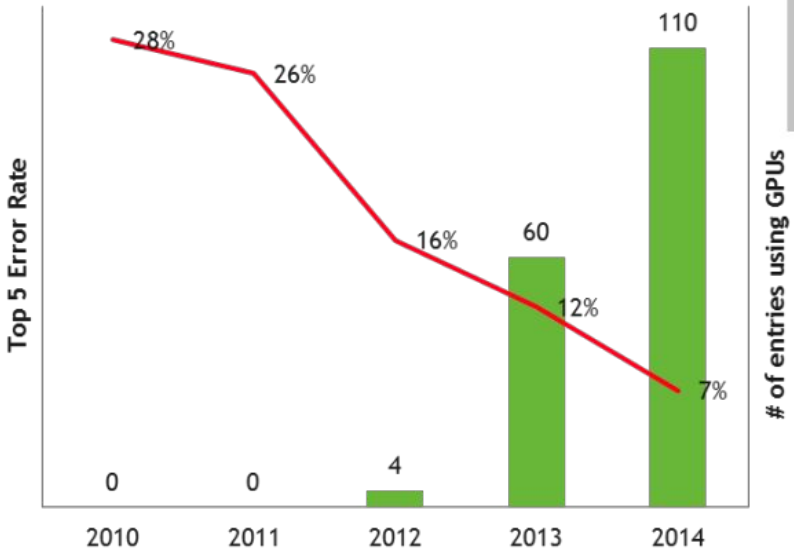


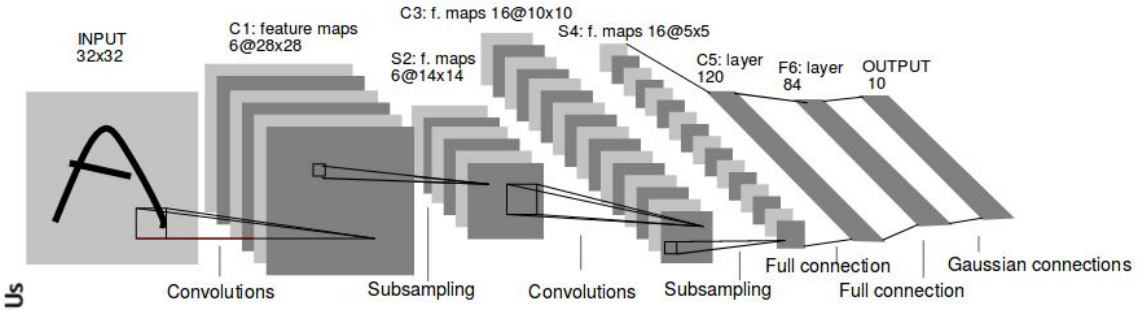
MNIST recognition [11]

Driving factors for the increased adoption of AI

- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks
  2. **Technical progress** in data and algorithms
     a. Some examples:
        GPU-accelerated convolutional neural networks [S15]

IMAGENET



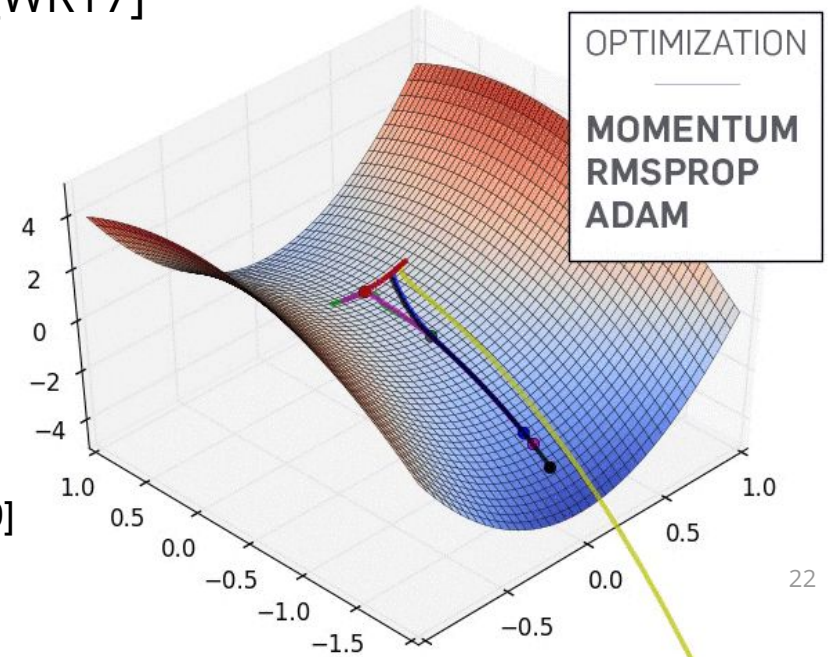


LeNet-5 One classical convolutional architecture, capable of GPU implementations [LBB98]

Throughout the years there has been a steady increase in ImageNET entries using GPUs [13]

Driving factors for the increased adoption of AI

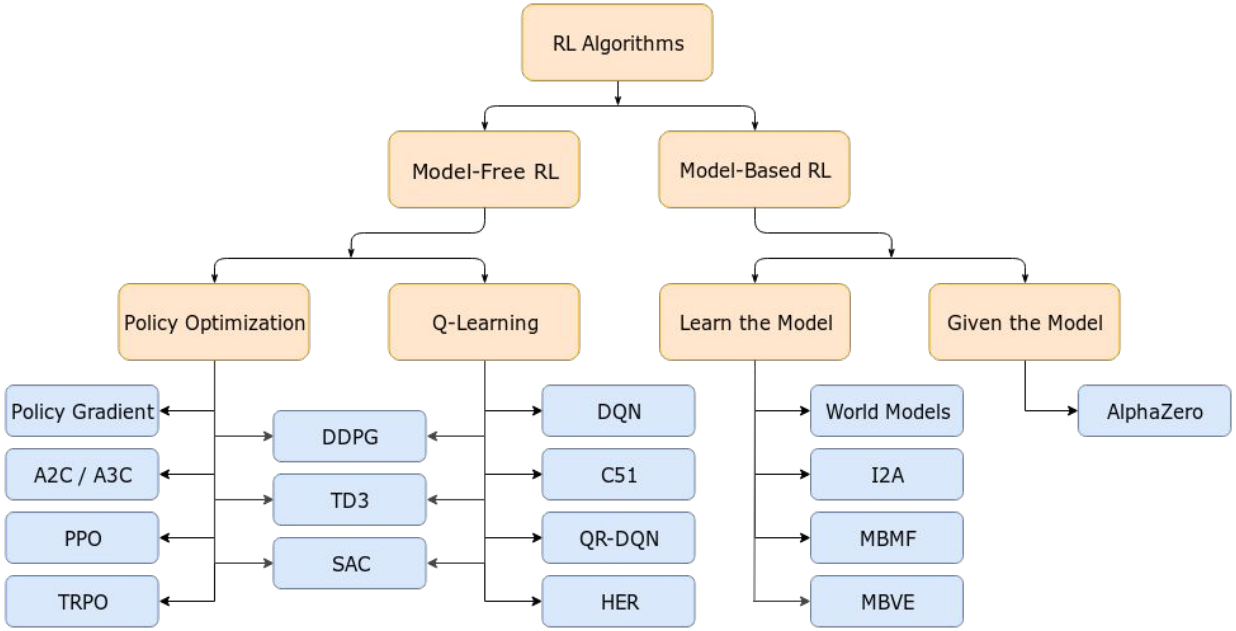
- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks
  2. **Technical progress** in data and algorithms
     a. Some examples:
        Improvements in deep learning techniques (e.g. dropout, ReLU activation functions), and optimization algorithms (e.g RMSProp, Adagrad, Adadelta) [S15][WR17]



Different optimizers have been proposed to improve stochastic gradient descent. They have different convergence properties [9]

Driving factors for the increased adoption of AI

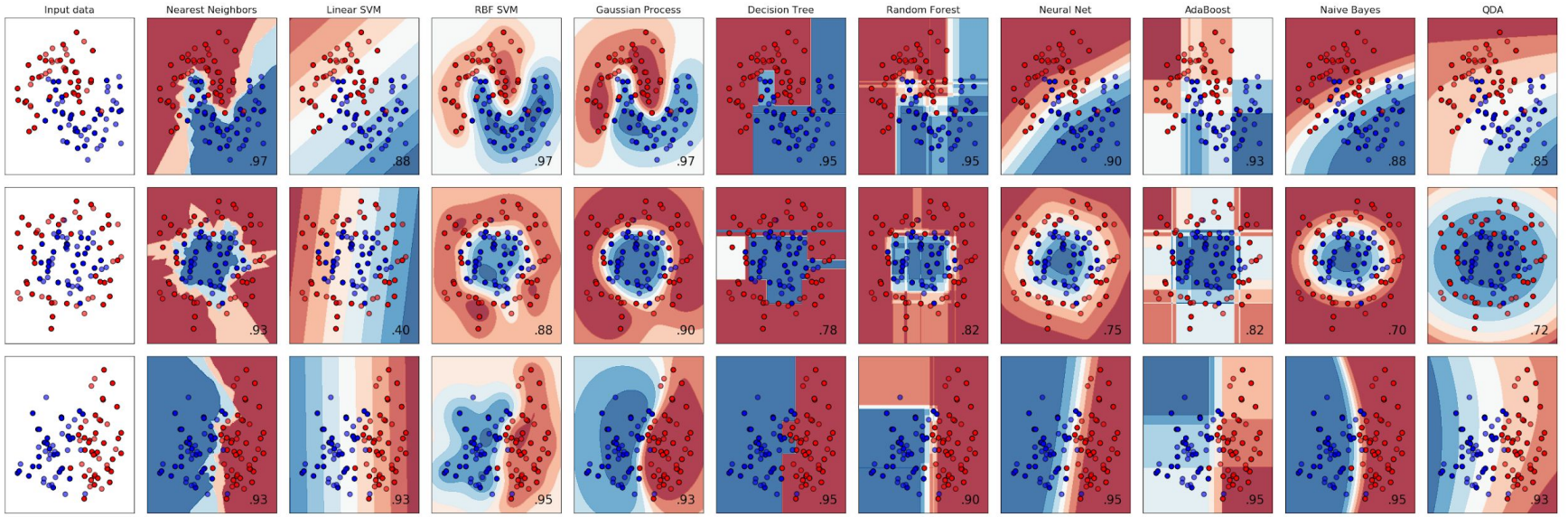
- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks
  2. **Technical progress** in data and algorithms
     a. Some examples:
        Novel deep reinforcement learning models and training methods [8][LHI18]



Picture taken from [8]

Driving factors for the increased adoption of AI

- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks
  2. **Technical progress** in data and algorithms
  3. Standardization and efficiency of **tools**



Picture taken from [10]

We live in a golden age for ML!
With an abundance of efficient model implementations at our fingertips.

Driving factors for the increased adoption of AI

- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks
  2. **Technical progress** in data and algorithms
  3. Standardization and efficiency of **tools**
  4. **Modern hardware**
     GPUs and specialized processors

(We cover examples of this later)

Big Data Landscape 2019- Pictures taken from [14]

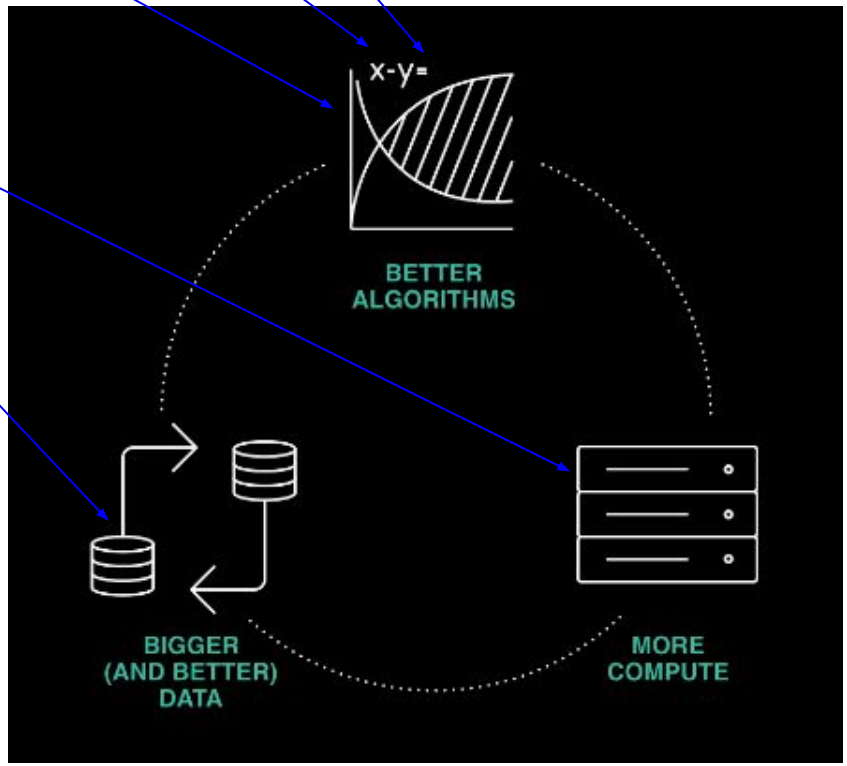
Driving factors for the increased adoption of AI

- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks
  2. **Technical progress** in data and algorithms
  3. Standardization and efficiency of **tools**
  4. **Modern hardware**
  5. Abundance of **data,** from many sources

     Existing internal data, missing internal data (not indexed or being considered), external data sources (that need to be integrated).

Driving factors for the increased adoption of AI

- The current interest on applying AI to tasks in all walks of life stems from:
  1. **Ongoing successes** in achieving human-level performance at grand challenge tasks
  2. **Technical progress** in data and algorithms
  3. Standardization and efficiency of **tools**
  4. **Modern hardware**
  5. Abundance of **data**



The virtuous cycle for AI's increased adoption. Picture taken from [24]

- Building on the success of approaches like deep learning, and of machine learning (ML) frameworks, coupled with progress in hardware and abundance of data, there's excitement on adopting AI for a large amount of applications.

- To better understand the components involved, we will consider in this first part of the lecture (Part 1- Introduction):
  - A canonical architecture + the field of SysML
  - Best/practices & configurations for AI in production

- However, before starting, we review some basic ways of categorizing ML applications

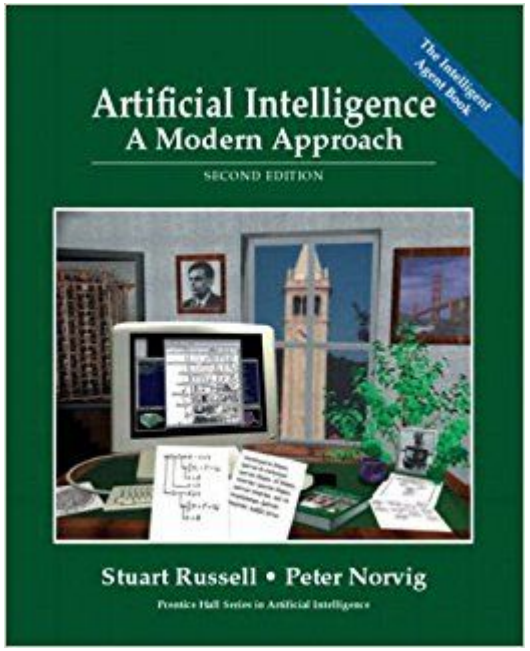
Part 1. Introduction

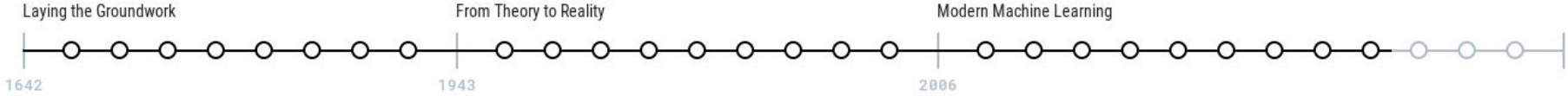
Sec 2. Basic Ways of Categorizing ML

Artificial Intelligence

Thinking Humanly	Thinking Rationally
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

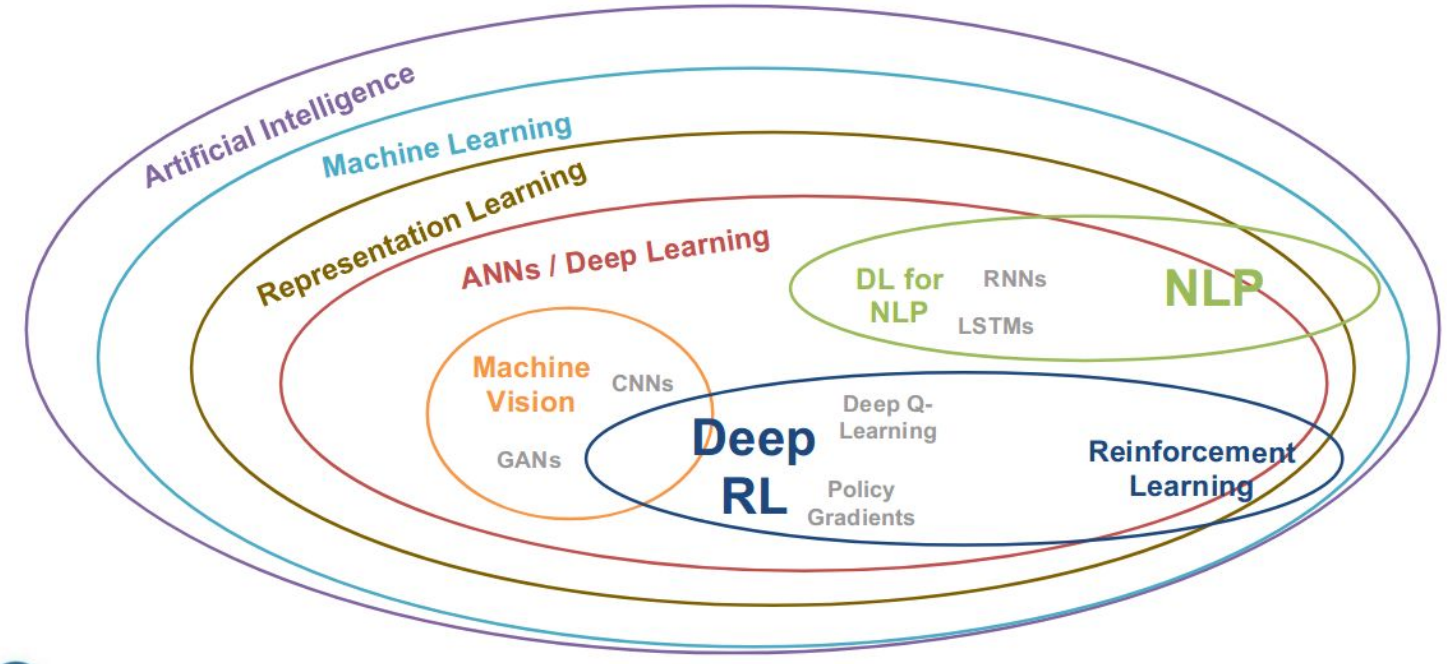
Figure 1.1 Some definitions of artificial intelligence, organized into four categories.





History highlights: https://cloud.withgoogle.com/build/data-analytics/explore-history-machine-learning/

Artificial Intelligence is a superset of Machine Learning



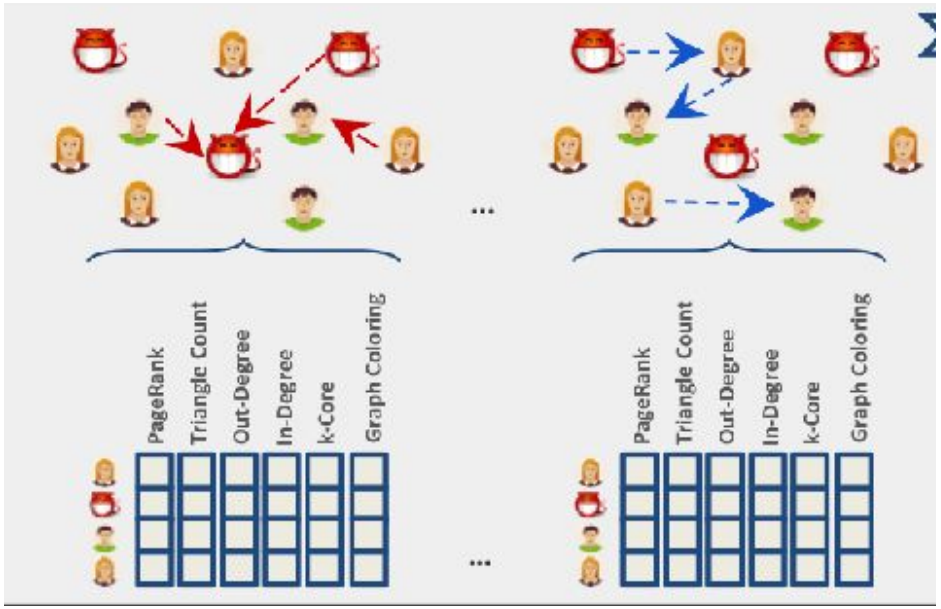
Picture taken from [26]

Machine learning involves statistical modeling of data, whereas AI is a broader concept about machines appearing smart.
AI can include hard-coded knowledge in formal languages, forming knowledge bases.

Our working definition of ML

- "**Machine learning** is the process of solving a practical problem by gathering a dataset, and algorithmically building a *statistical* model based on that dataset." [Bur19]

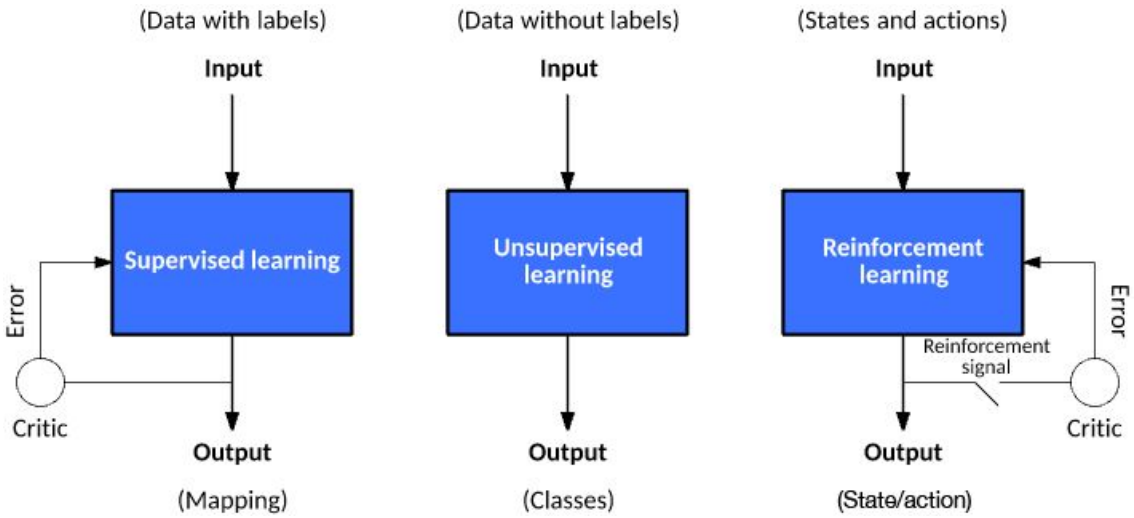
- Key terms[Zin19]:
  - Instance
  - Label
  - Feature
  - Feature Column
  - Example
  - Model
  - Metric
  - Objective
  - Pipeline



Picture taken from [FFS15]

The many categorizations of ML

- **First Categorization:** *Based on the application*

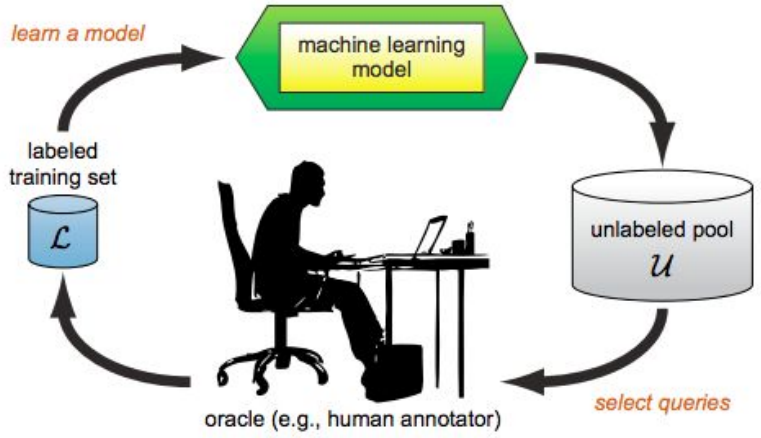


Picture taken from [28]

Supervised: Requires labels (e.g regression, classification)
Unsupervised: No labels (e.g. clustering, frequent pattern mining, anomaly detection)
Semi-supervised: Supervised learning, but also using unlabeled data
Reinforcement Learning: Agents learn based on interaction, under a *Markov Decision Process*

The many categorizations of ML

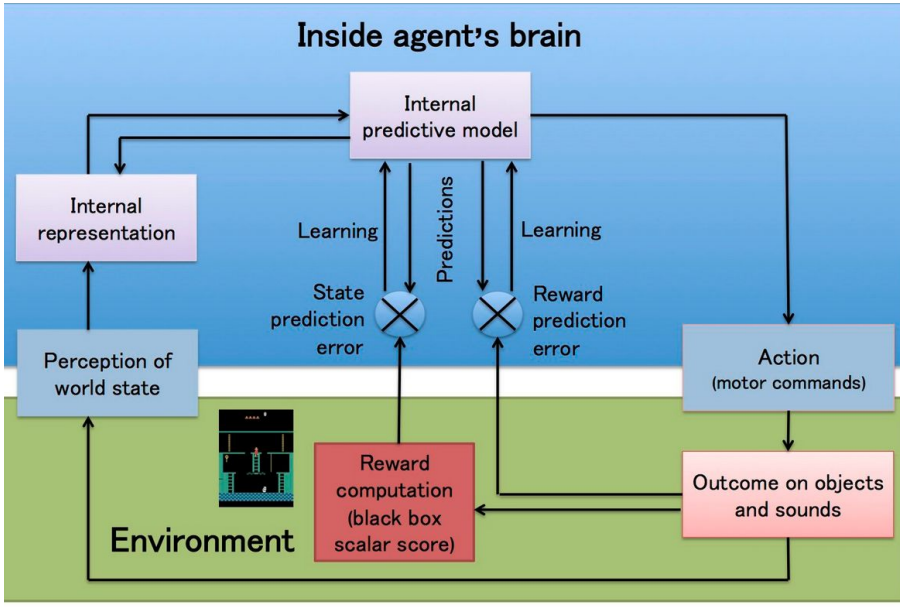
- **Second Categorization:** *Based on the involvement of the model in selecting the data used for training*
  - Passive learning
    - The model learns on data given, and is not involved in the data selection process.
  - Active learning
    - The model selects, from a pool of unlabeled data, which data should be labeled to improve the model.



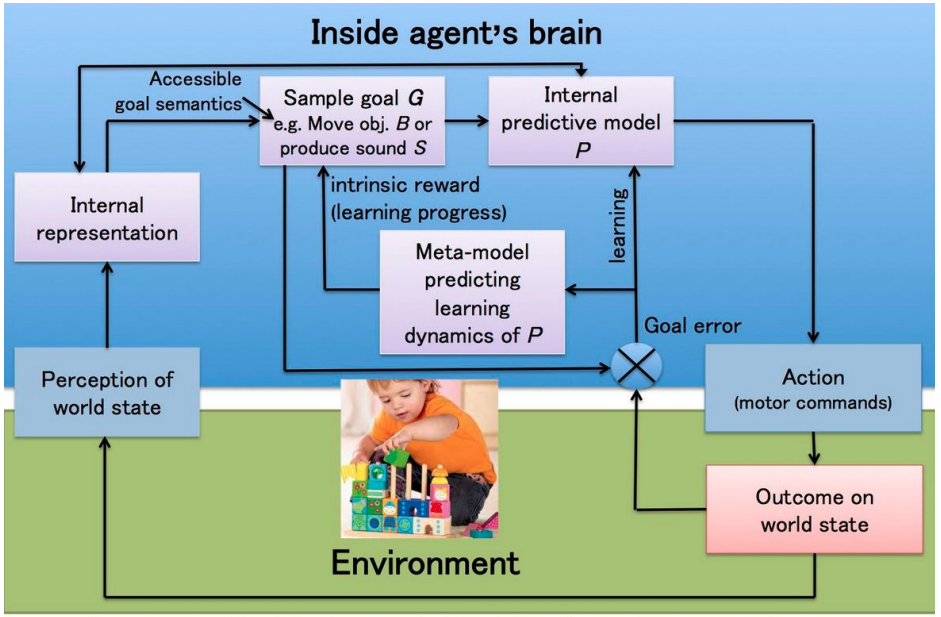
Picture taken from [29]

The many categorizations of ML

- **Second Categorization:** *Based on the involvement of the model in selecting the data used for training*
  - Reinforcement learning is a form of Active learning
    - Either by being created for a single goal, or by generating goals with intrinsic motivation, in RL the model is also actively involved in the training data generation.



The classical (Deep) Reinforcement Learning paradigm

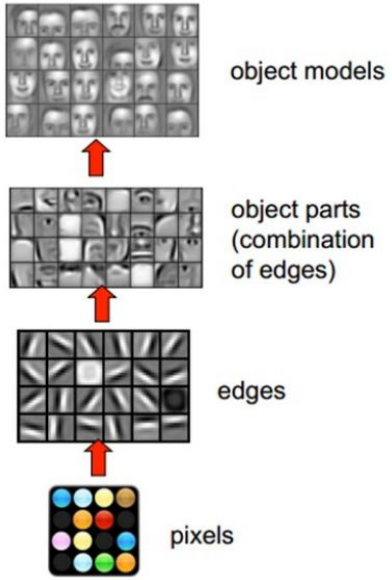


Intrinsically Motivated Goal Exploration paradigm

Picture taken from [27]

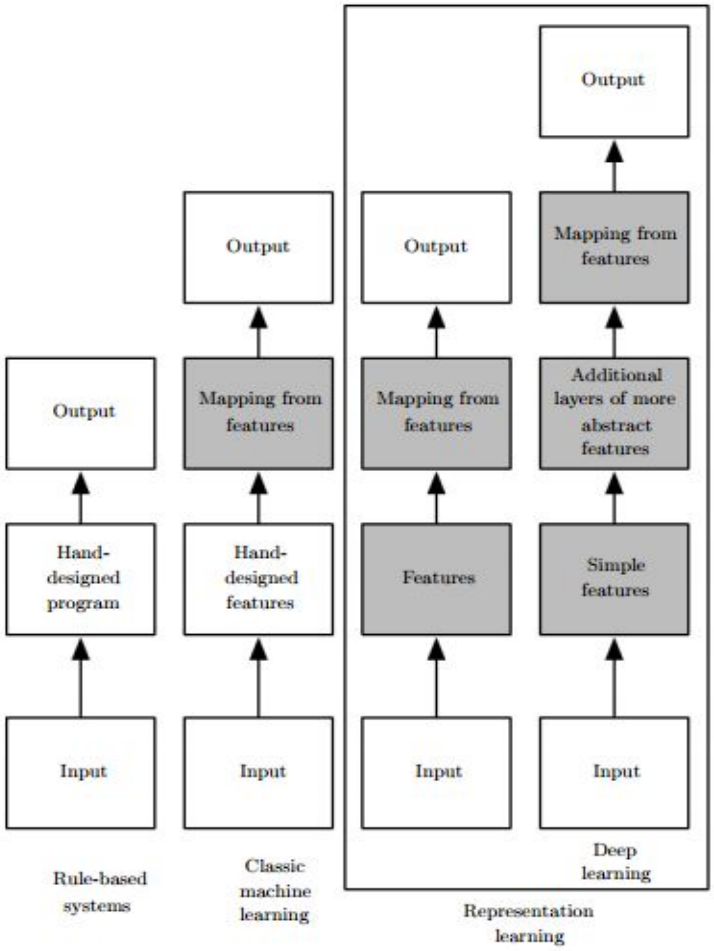
The many categorizations of ML

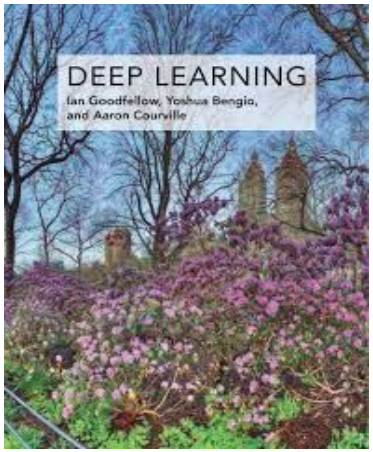
- **Third Categorization:** *Based on the amount of human involvement in the feature engineering*
  - Feature Engineered: Traditionally features are provided for the models.
  - Representation Learning: Models are provided with raw data and while solving their task, they learn an internal representation (featurization) that can be used also by other models.



Representation Learning [GBC16]

The many categorizations of ML

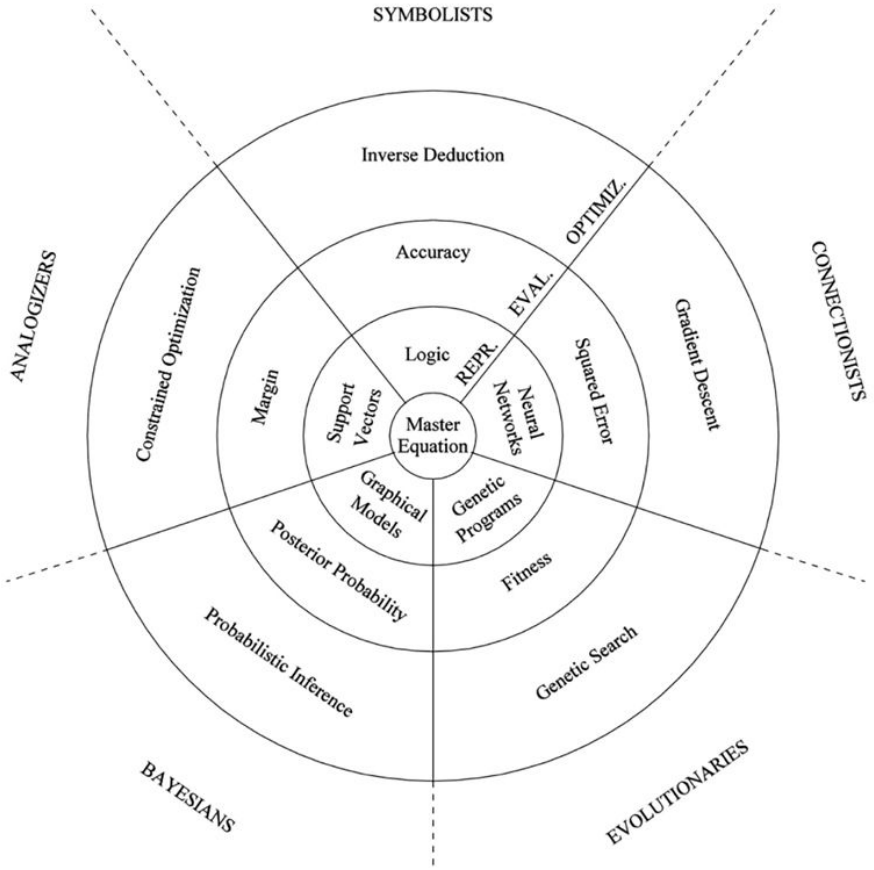




Representation Learning [GBC16]

The many categorizations of ML

- **Fourth categorization:** *Based on the kind of model used*



Picture taken from [Dom15]

  - These are the so-called “ 5 Tribes of ML” [Dom15]

The many categorizations of ML

- **Fourth categorization:** *Based on the kind of model used*
  - *1. Bayesian models:*
    - These methods are based on Bayes rule on conditional probability.
    - They focus on reducing the uncertainty about a prediction, usually assuming a parameterized distribution function.
    - The Naive Bayes classifier is the simplest introductory example on how the Bayes rule can be applied for predictions.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

P(Yes or No | Sunny) =?
P(Yes| Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)
P (Sunny |Yes) = 3/9, P(Sunny) = 5/14, P(Yes)= 9/14 => P(Yes or No | Sunny) = 0.6
P(No|Sunny)=0.05
So the most likely prediction for Sunny would be Yes.

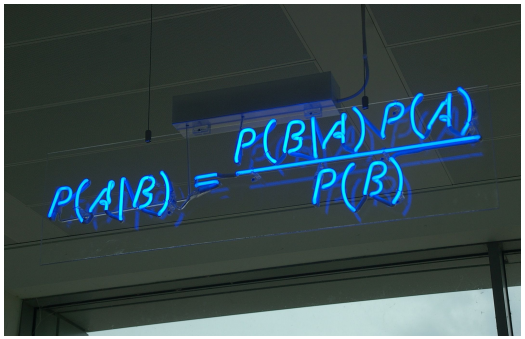
Picture taken from [30]

The many categorizations of ML

- **Fourth categorization:** *Based on the kind of model used*
  - *1. Bayesian models:*
    - Models based on the Bayes rule can be extended to:
      - Include continuous variables (by fitting the parameters of a distribution function, to be the most likely given the training data).
      - Break the naive independence assumption of events (with the use of probabilistic graph models)

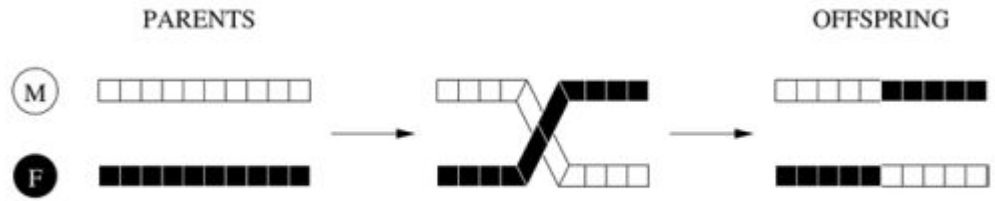
The many categorizations of ML

- **Fourth categorization:** *Based on the kind of model used*
  - *1. Bayesian models:*
    - They work in the following way:
      - Given:
        - an a priori distribution function <= highly important hyper parameter
        - a likelihood distribution that summarizes the data
      - We try to find an a posteriori function that attempts combines the likelihood and the a priori assumptions
      - Methods like MCMC can be employed to find the best parameters of the a posteriori function.
    - *Probabilistic programming* tools exist to support these kind of models (e.g PyMC).



The many categorizations of ML

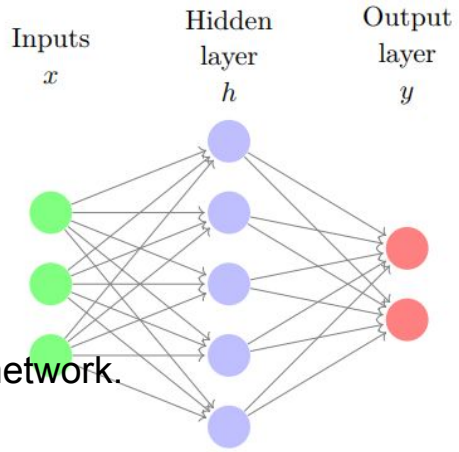
- **Fourth categorization:** *Based on the kind of model used*
  - *2. Genetic algorithms*
    - Given a population of candidates, the population is grown through:
      - Crossover
      - Mutation
    - Candidates are evaluated through fitness functions, and they propagate to the following generation.
    - Can be applied to classification, for example by growing rules codified as bits.
    - These methods explore well, when contrasted to neural networks.



Crossover
Picture taken
from [Dom15]

The many categorizations of ML

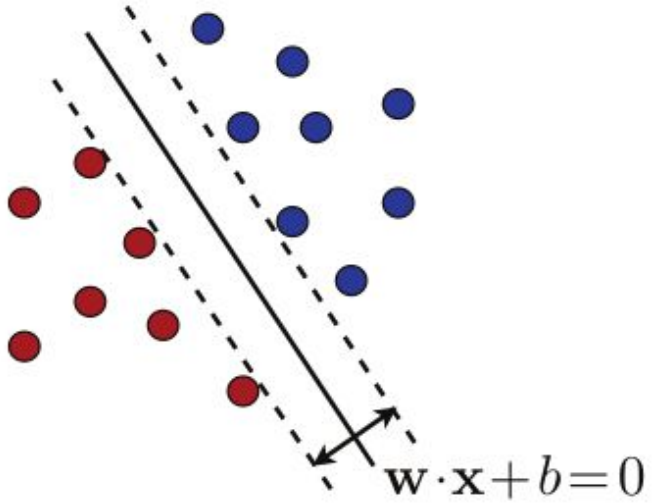
- **Fourth categorization:** *Based on the kind of model used*
  - *3. Neural networks (a.k.a Deep learning)*
    - Given a numerical input, this approach maps it, through a series of layers where each layer can perform several functions, such as adding biases, or activation functions, to an output signal.
    - The connections at each layer carry trainable weights, which, through a method like gradient descent with the backpropagation algorithm, can be fitted to minimize a loss function between the output and the expected output for a training dataset.
    - Convolutional and recurrent layers are especially useful for image and sequential data.
      GANs are highly relevant too.



Simple schema of a neural network.
Picture taken from [LHI18]

The many categorizations of ML

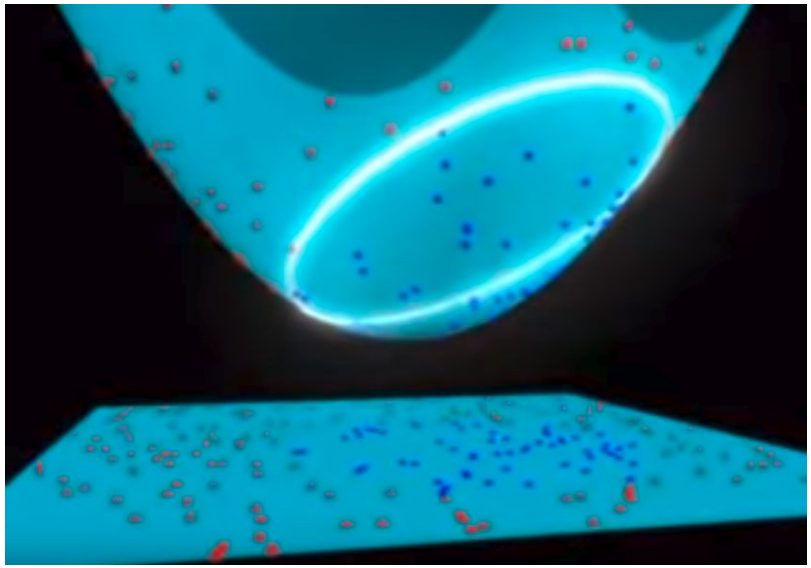
- **Fourth categorization:** *Based on the kind of model used*
  - *4. Support Vector Methods*
    - Given support vectors: the minimum of points that need to be kept from the data, to draw class boundaries, for a classification task.
    - The goal is to find a hyperplane that divides these vectors with the maximum margin.



Picture taken from [MTR18]

The many categorizations of ML

- **Fourth categorization:** *Based on the kind of model used*
  - *4. Support Vector Methods*
    - For complex cases, it is possible to map the points to a high-dimensional space where they are more easily separable. The distance between points can be redefined to a kernel function.
    - Solving this problem is a constrained optimization problem.



Visualization of moving to the high-dimensional space and the hyper-plane dividing classes found there
Picture taken from [31]

The many categorizations of ML

- **Fourth categorization:** *Based on the kind of model used*
  - *5. Symbolic AI*
    - Often called good old fashioned AI.
    - One example is the building of a network of production rules, over which reasoning can be made.
      - If X- then do->Y
  - Apart from the tribes, there are also other models commonly used, such as tree-based models.

The many categorizations of ML

- We covered 4 categorizations.
- Other categorizations could also be suggested…
  - Single agent learning vs. Multi-agent learning, Population learning, Swarm intelligence or Adaptive mechanism design.
  - Learning with a model vs. learning …
  -
- But the categories that we just considered, are sufficient for an introductory level.

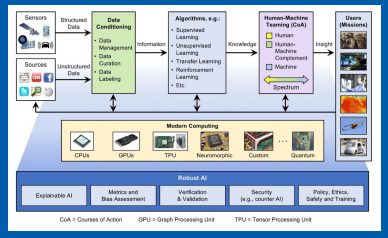
Summary

- AI is a superset of ML
- ML can have many categorizations
  1. Based on the application
  2. Based on the involvement of the model in selecting the data used for training
  3. Based on the amount of human involvement in the feature engineering
  4. Based on the kind of model used

When searching through papers, it is useful to recognize quickly how the solution can be categorized.

Part 1. Introduction

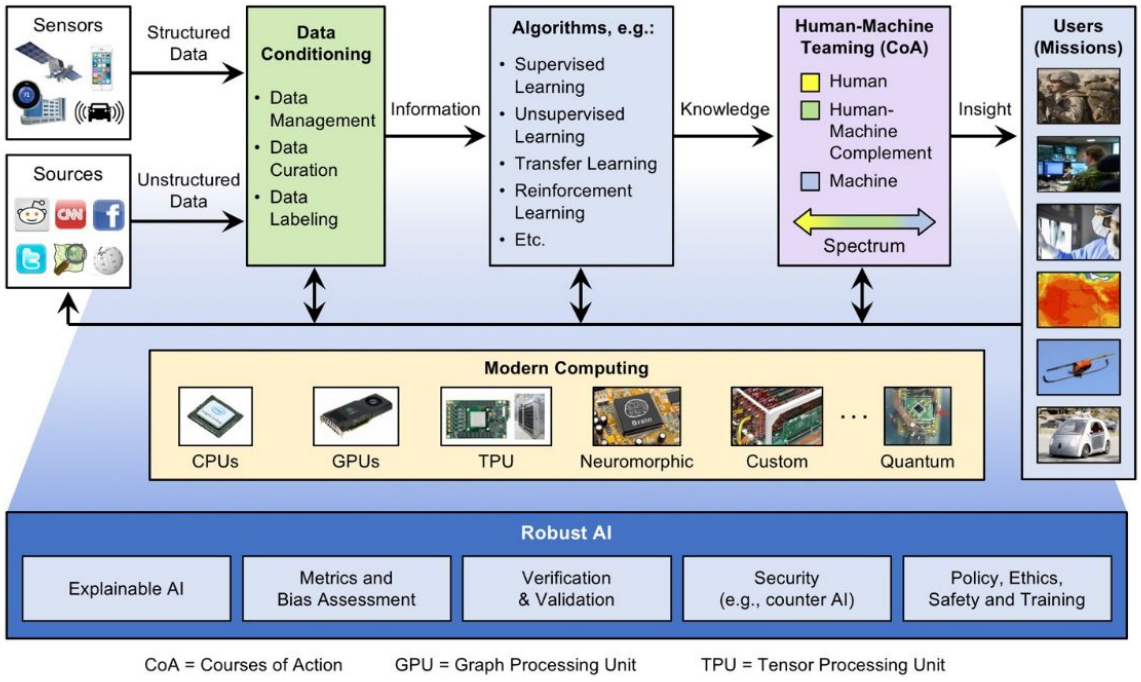
Sec 3. A Canonical Architecture for AI-enabled Applications & The New Field of SysML



SysML

A Canonical Architecture for AI-enhanced Applications

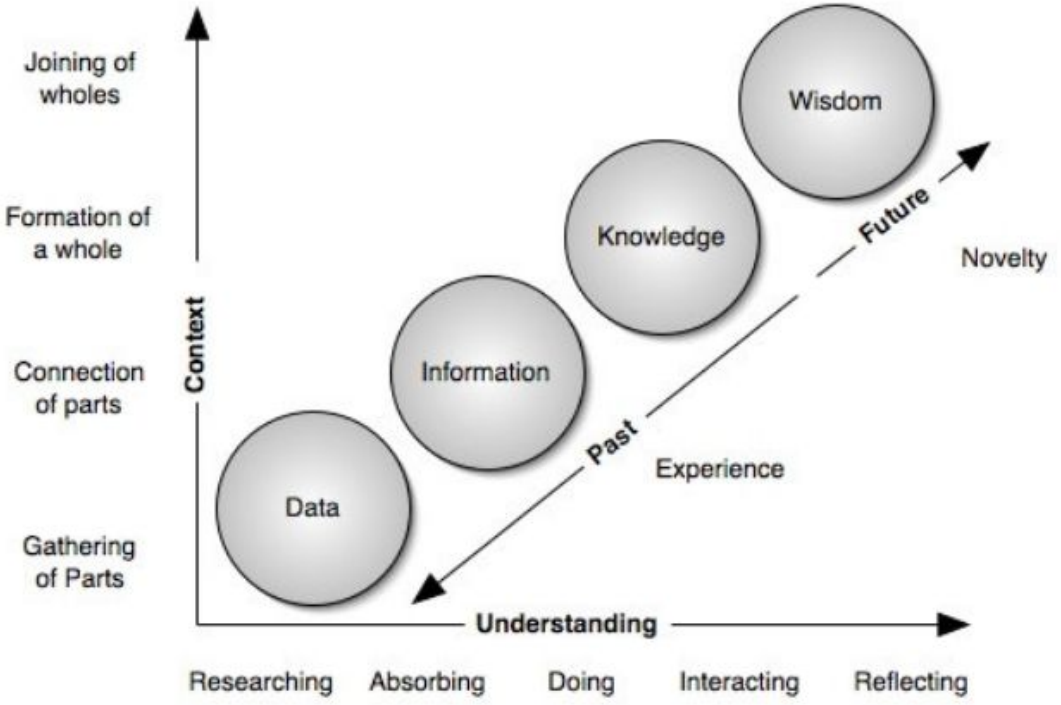
- Researchers have proposed a canonical architecture.
- Let's consider the *overall process* and *components* involved.



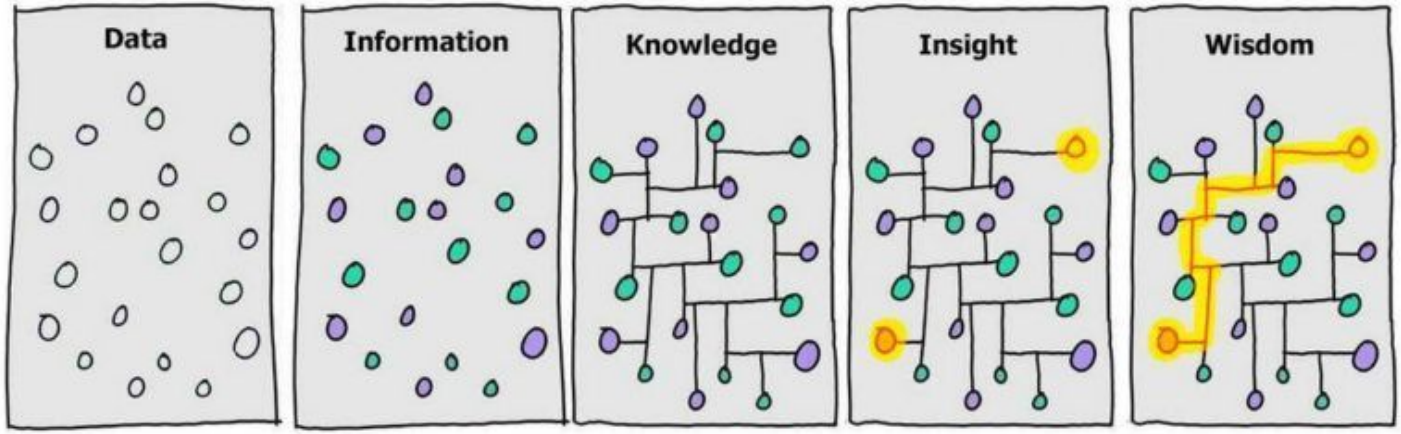
This architecture, shows **robustness** as an overall concern [GGK19]

A Canonical Architecture for AI-enhanced Applications: *Overall process*

- Transforming ***data*** to ***information*** to ***insight*** that can then be applied to end-user missions is the overall process.
- The additional skill for using insights to perform actions that would improve end-user missions, can be considered ***wisdom***.



The DIKW framework [F18]

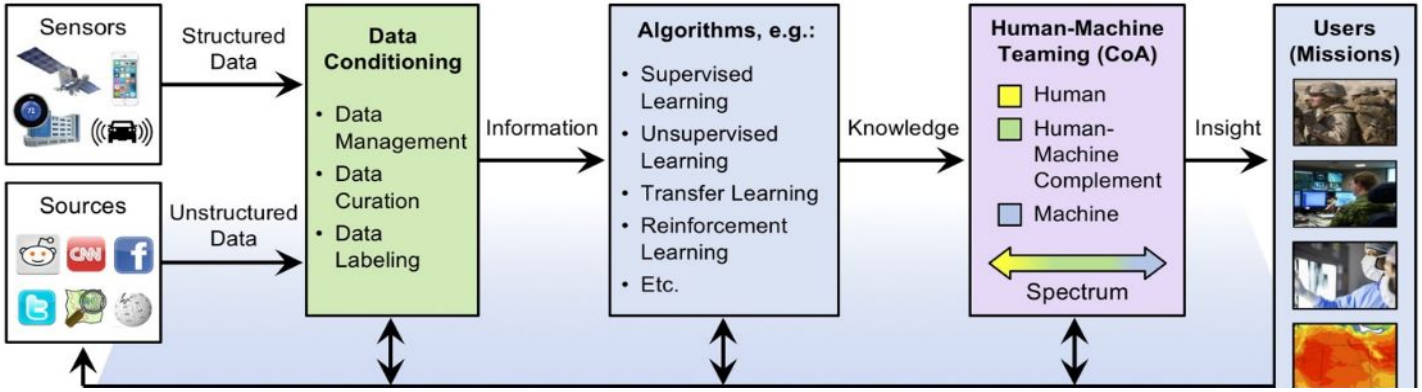


Picture taken from [15]

- As an example, sensor measurements related to air quality can be our ***data*** (e.g. amount of a given chemical detected in the air, etc).
- ***Information*** can be human-assigned labels indicating what cases were acceptable in air quality
- ***Knowledge*** can be built in the form of a trained classification model to predict for new measurements if the air quality will be acceptable.
- A use of this knowledge could be to understand (***insight***) what changes in the data producing process might ensure a sustained high air quality.
- Applying these changes and fine-tuning them, based on new observations, shows some ***wisdom*** from the system.

A Canonical Architecture for AI-enhanced Applications: *Overall process*

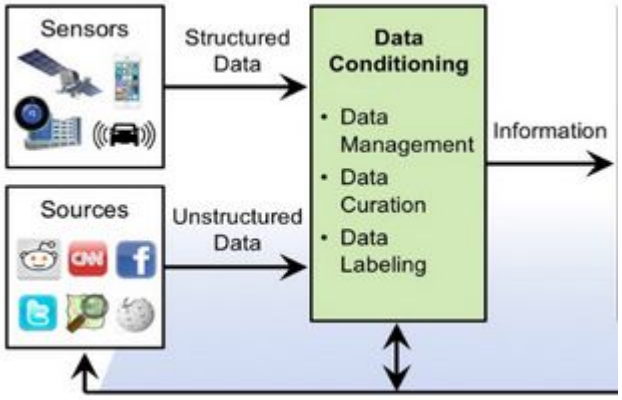
- **Data->Information->Insight->Wisdom**
- This process also assumes ***continuous evaluation***, of the components in the form of an abstract component called Robust AI.



[GGK19]

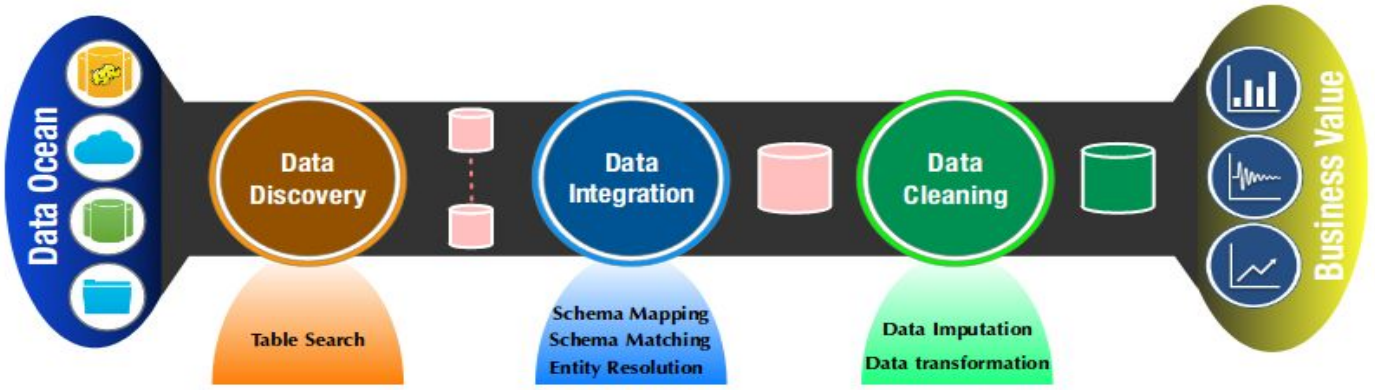
A Canonical Architecture for AI-enhanced Applications: *Components*

- *Sensors and Data Sources*
  - Structured data includes metadata that allows to infer its structure
  - Unstructured data does not include such metadata
- *Data Conditioning (a.k.a data wrangling)*
  - Some practitioner surveys claim that this process takes 80% of the time to prepare data [16]
  - A large number of AI advances have occurred where conditioned data, in the form of challenging datasets was made available.



A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning



[TTO18]

- Data conditioning involves:
  - Data Management (i.e., reliable storage, tracking lineage)
  - Data Discovery (i.e., finding possible data structures in data lakes, identifying relations that could enhance the current data)
  - Data Integration/Linkage (e.g., creating an entity from several sources, de-duplication)
  - Data Cleaning (e.g., imputation of missing values, outlier detection, standardization) and Augmentation
  - Data Labeling

A Canonical Architecture for AI-enhanced Applications: *Components*

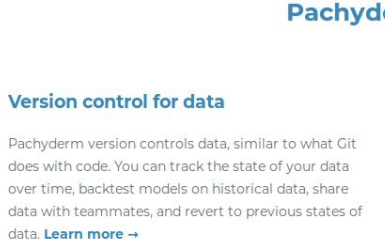
Data Conditioning

- *Discovery*
  - The goal is to enhance a known dataset, by adding new data to it
  - One representative goal is to have a 360 vision of users, compiling their data from many sources

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

- *Discovery*
  - *Two approaches: Data sharing and Searching on data lakes*
    1) Data sharing
       - Data scientists collaborate by searching/creating versioned and well-documented datasets and coordinating in identifying potential sources to merge
       - Hosted ML notebooks are also used in the process
       - Web sources can be scraped, to create data
       - Data markets are available too, to purchase datasets



Determined AI

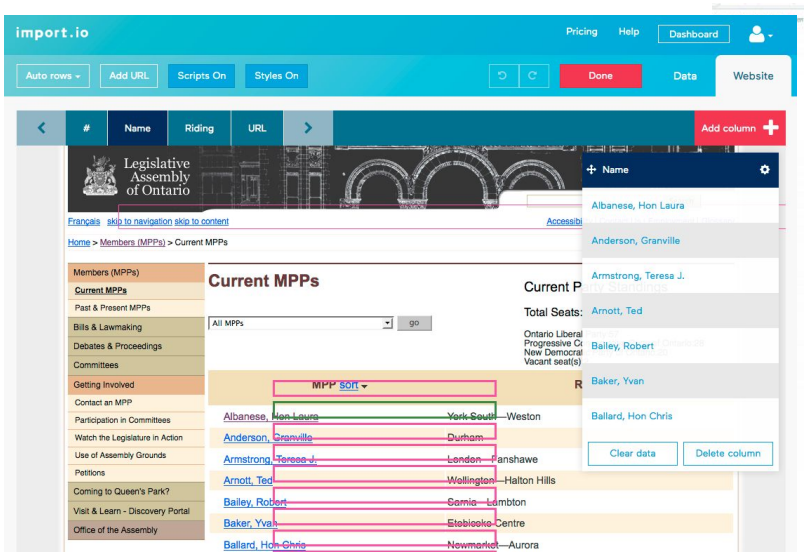
FLOYDHUB

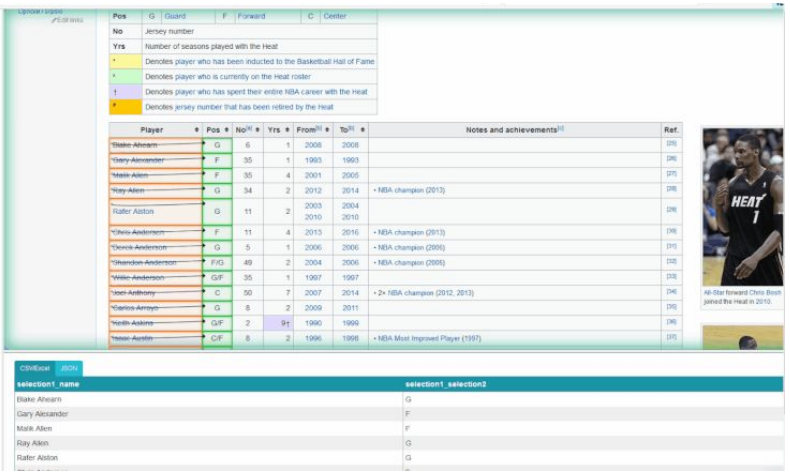
A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

Example Tools

- *Discovery- Data sharing*
  Example tools for web-scraping, based on rules and instances, helping data sharing
  - *Import.io*
  - *ParseHub*





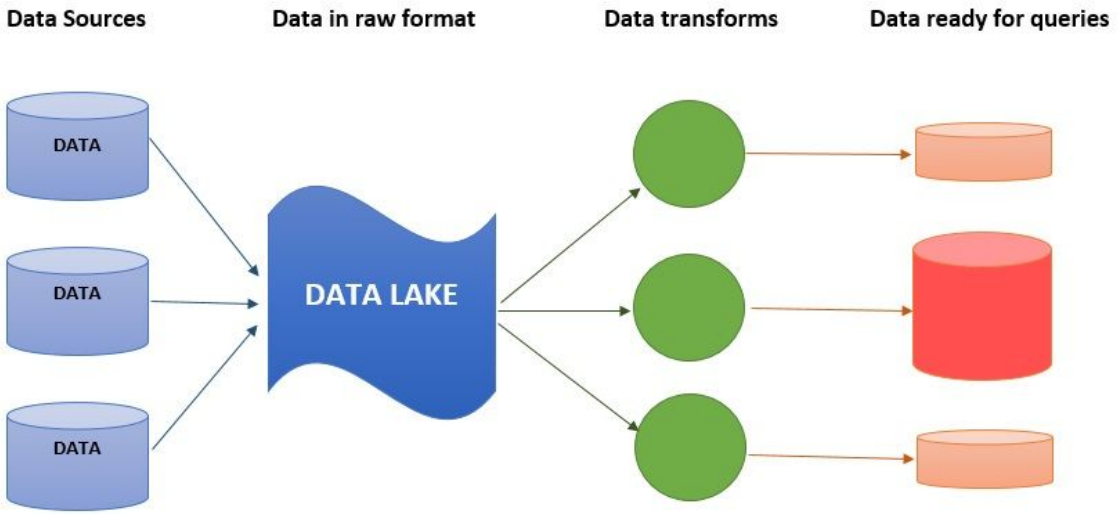
Tools for screen scraping and creating tables from the scraping [17,18]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

- *Discovery*
  - *Two approaches:*

    2) Searching on data lakes
    - Data lakes are the unstructured collection of data available in a company, without a clear indexing.
    - By keeping data lakes ingestion can be fast, at the cost of overheads in querying.



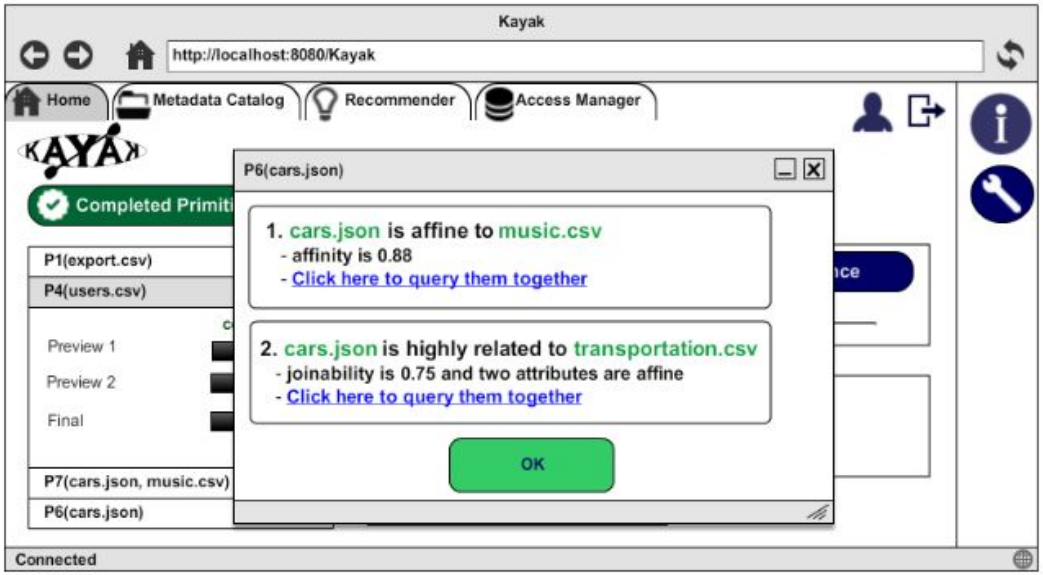
Picture taken from [19]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

Example Tools

- *Discovery- Data searching over raw data or data lakes*
  Example tools creating data structure in data lakes during querying
  - *Kayak* [MT17]



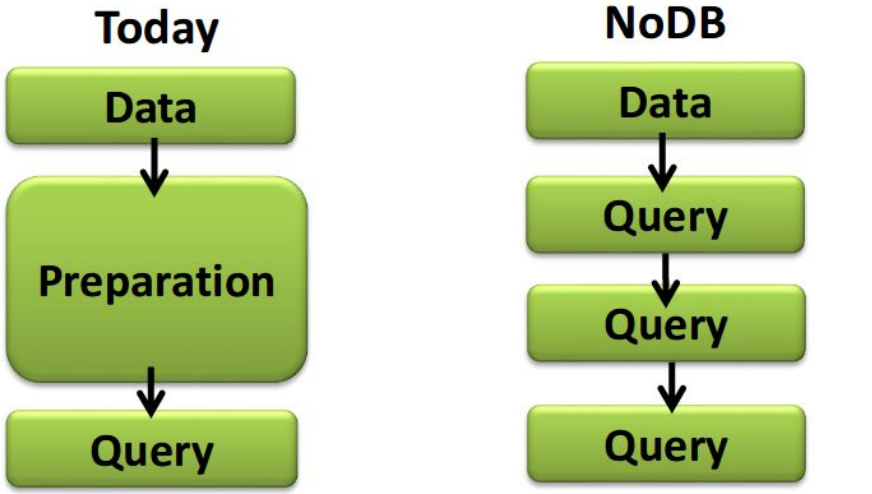
Kayak helps users to consider possible links between datasets from a data lake by computing affinity and joinability based on data profiles [MT17]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

Example Tools

- *Discovery- Data searching over raw data or data lakes*
  Example tools creating data structure in data lakes during querying
  - *NoDB* [ABB12]- Proposes to create query-engines on the fly, that learn about data structure, based on queries.

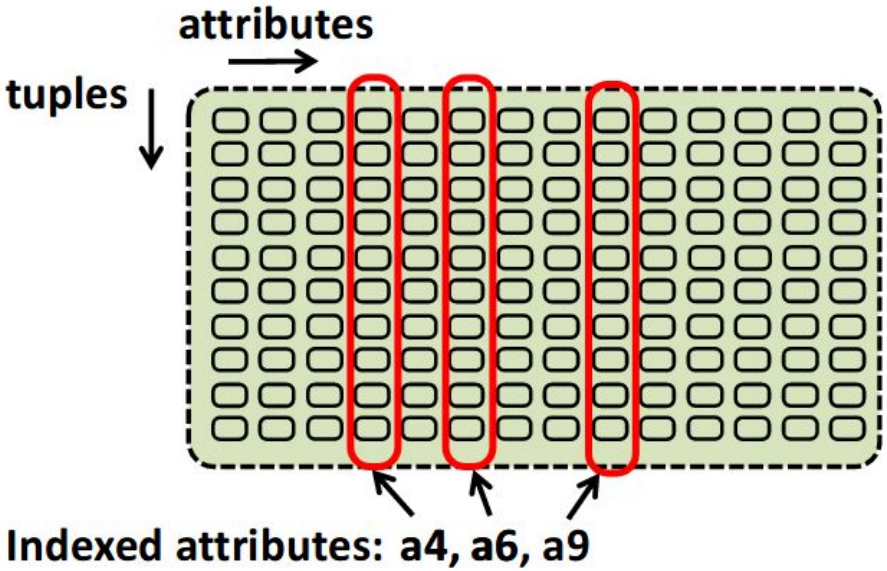


[ABB12]

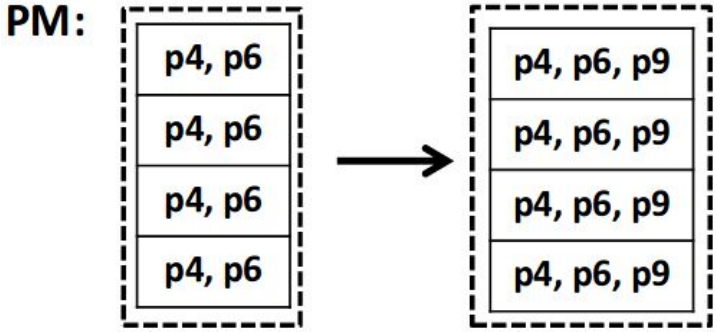
A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

Example Tools



1. Positional map is empty
2. Q1 accesses *a4* and *a6*
3. Q2 accesses *a4* and *a9*



[ABB12]

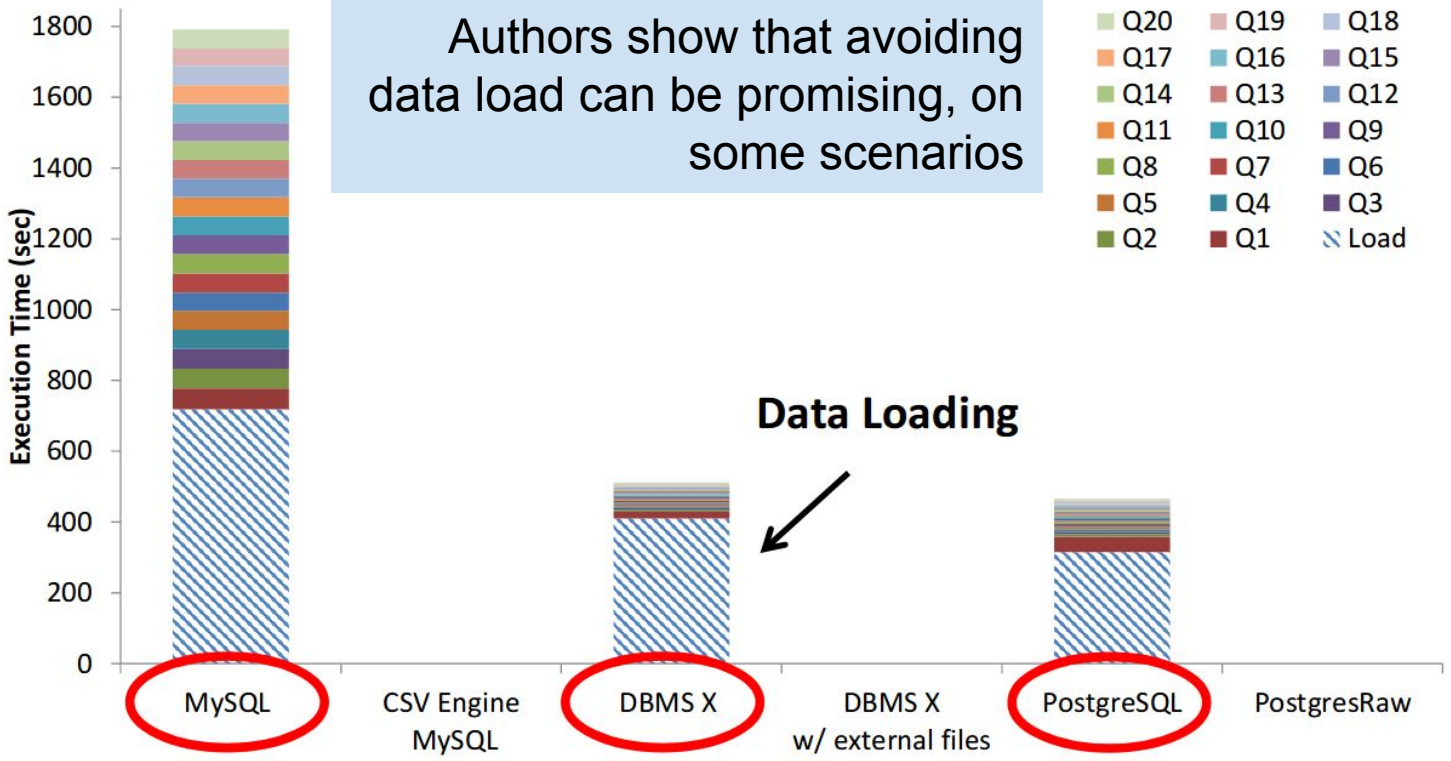
A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

Example Tools

PostgresRaw vs. other DBMS

Authors show that avoiding data load can be promising, on some scenarios



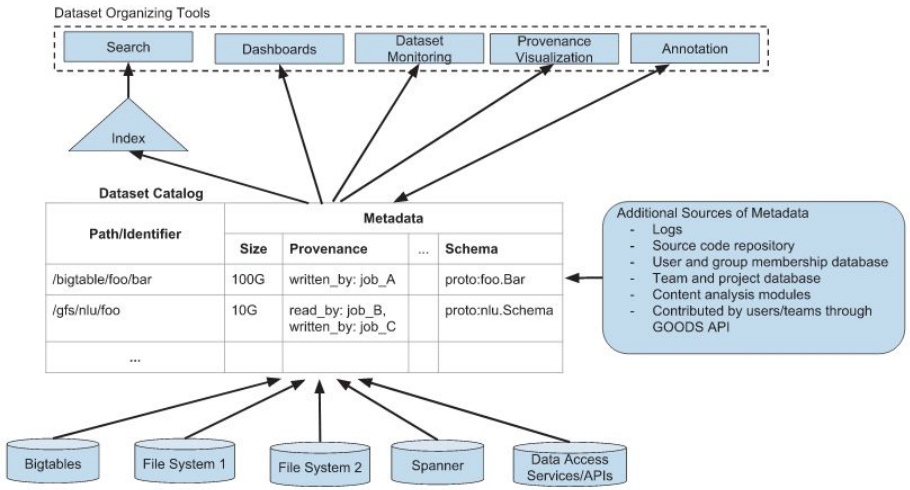
Authors evaluate NoDB over Postgres, called PostgresRaw [ABB12]

A Canonical Architecture for AI-enhanced Applications: *Components*

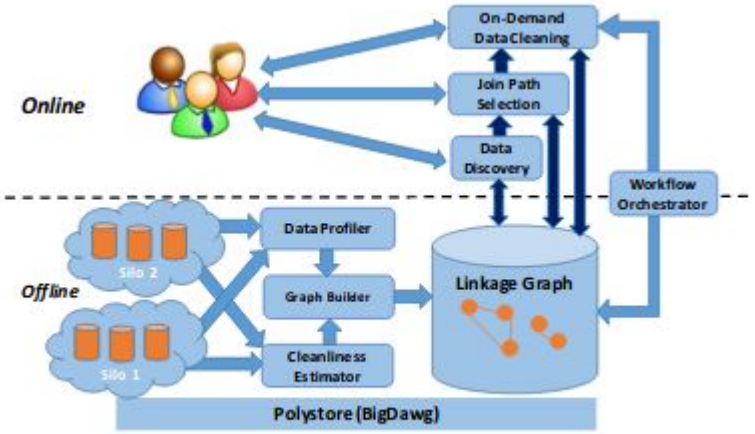
Data Conditioning

Example Tools

- *Discovery- Data searching over raw data or data lakes*
  Example tools creating data structure in data lakes during querying
  Others:
  - *Data civilizer [*DCA17]
  - *GOODS* [HCN17]: Metadata collection



[HCN17]



[DCA17]

Figure 1: Data Civilizer Architecture

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

- *Data Integration*
  - Creating single entities by combining their versions available in different data sources.
  - *Entity resolution* is one such task.

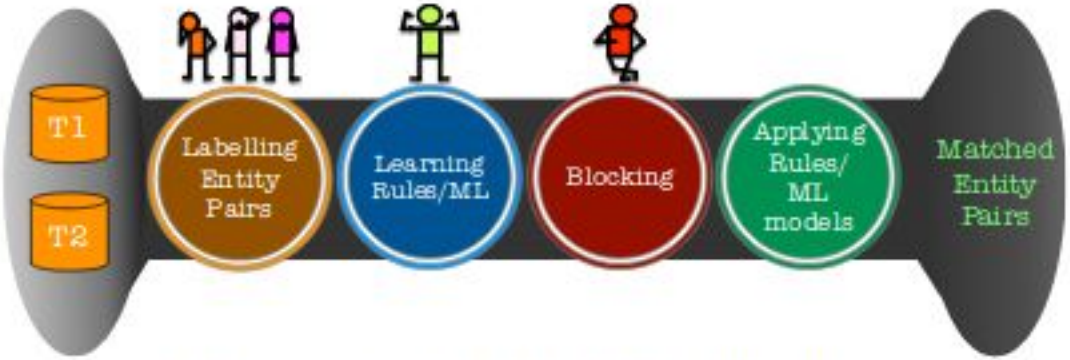


Figure 1: A Typical ER Pipeline

[TTO18]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

Example Tools

- *Data Integration*
  Example tools:
  - *Unify- Tamr [20]*
    - Based on the Data Tamer, developed at the MIT.
    - Suggests schema mappings between sources, using ML.
    - Employs Spark for data normalization scripts.
    - Offers a simplified interface that guides the resolution process with yes/no questions.

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

Example Tools

- *Data Integration*
  Example tools:
  - *DeepER* [TTO18]:
    - Authors develop a system that creates a *distributed representation* (often called embeddings) for the words in each tuple.
    - Authors show that this helps in the ML classification task of determining if two tuples from different sources are a match or not.

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

(a) Local Representations

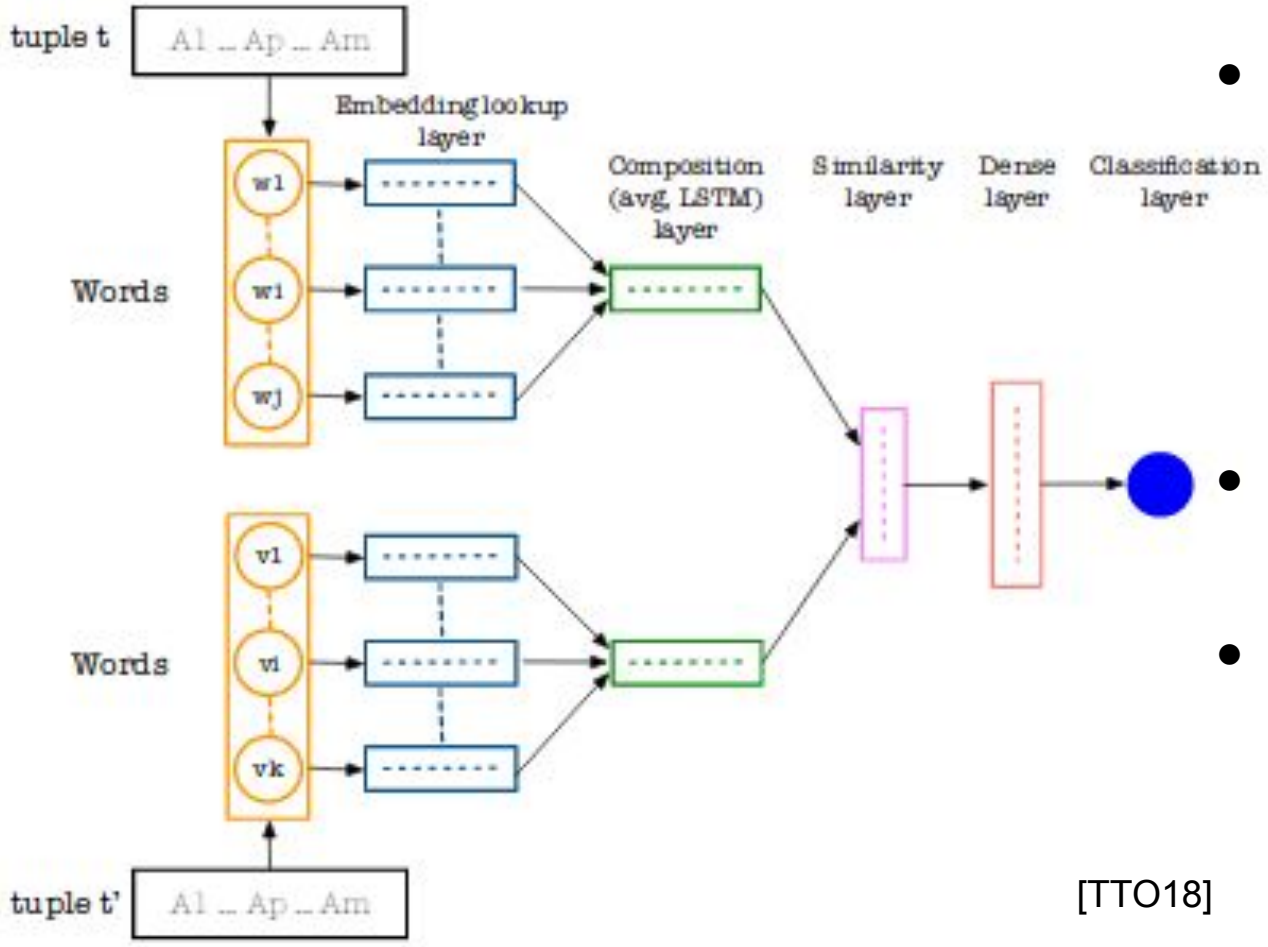
	Femininity	Youth	Royalty
man	0	0	0
woman	1	0	0
boy	0	1	0
girl	1	1	0
prince	0	1	1
princess	1	1	1
queen	1	0	1
king	0	0	1
monarch	0.5	0.5	1

(b) Distributed Representations

[TTO18]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning



Example Tools

- After embedding, words are composed into a complete representation for the tuple, such that classification can take place.
- Authors perform all steps of the process with deep learning
- This is the SOTA on automated entity resolution.

[TTO18]

A Canonical Architecture for AI-enhanced Applications: *Components*

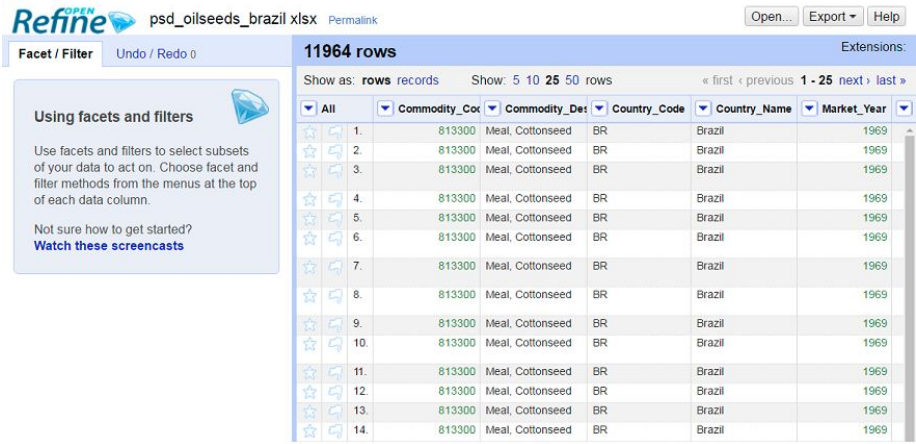
Data Conditioning

- *Data Cleaning*
  - Normalization, standardization, imputation of missing values, etc...

Example tools:

Example Tools

  - *OpenRefine:*
    - Helps users to define rules and identify their impact, during data cleaning.
    - Uses a standard refinement language, called Google Refine Expression Language



Picture taken from [21]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

- *Labeling*
  - For supervised learning algorithms, labels are necessary.

Picture taken from [23]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

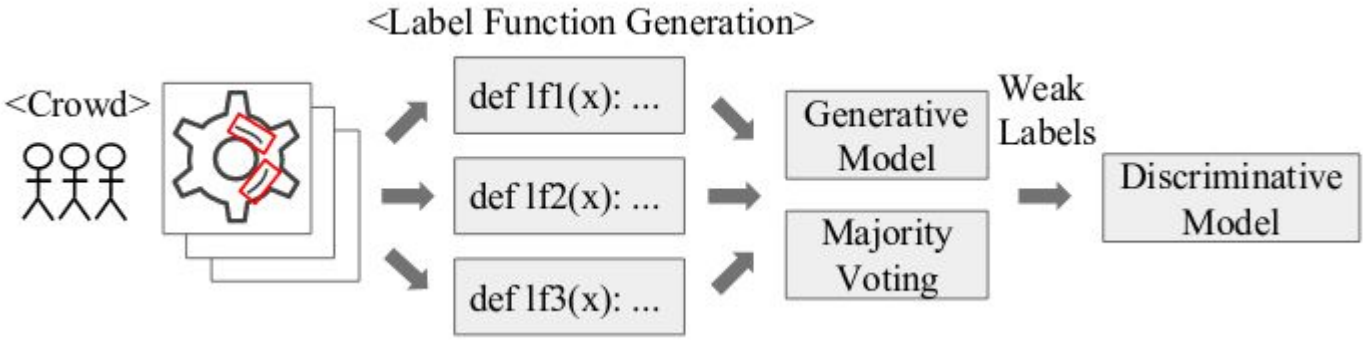
- *Labeling*
  - For supervised learning algorithms, labels are necessary.

Picture taken from [23]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

- *Labeling*
  - For supervised learning algorithms, labels are necessary.
  - When there are limited labels, semi-supervised learning can be chosen.
  - Manual labeling is also applicable:
    - Crowdsourcing <= Easy, but labelers might be unreliable
    - Weak supervision<=Improves over crowd-sourcing and can be combined with it: A better approach



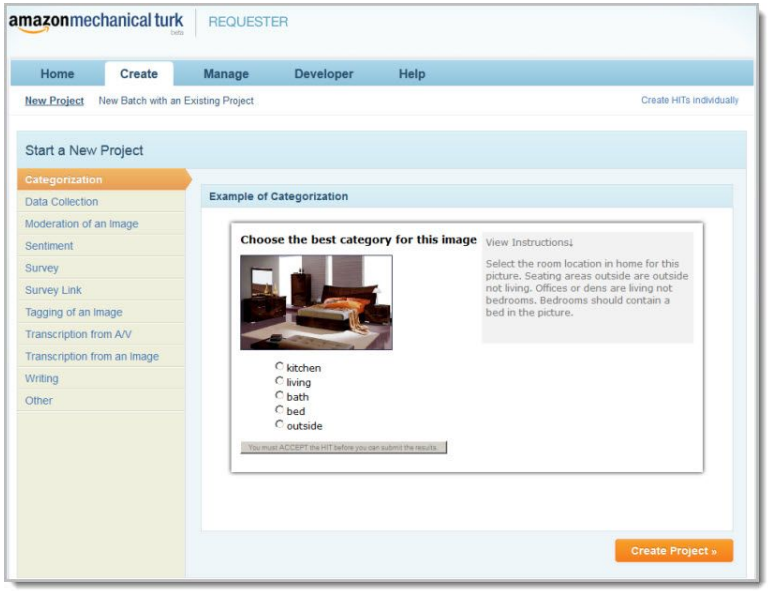
Labeling with weak supervision: Possible labels are collected as labeling functions, which are combined to form weak labels. A discriminative model learns how to pick [RGS18]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

- *Labeling*
  Examples of tools
  - *Amazon Mechanical Turk and Amazon SageMaker Ground Truth*
    Offers a marketplace that can be employed for labeling tasks
    Also specialized features for labeling are considered, with an active learning tool guiding the process.

Example Tools



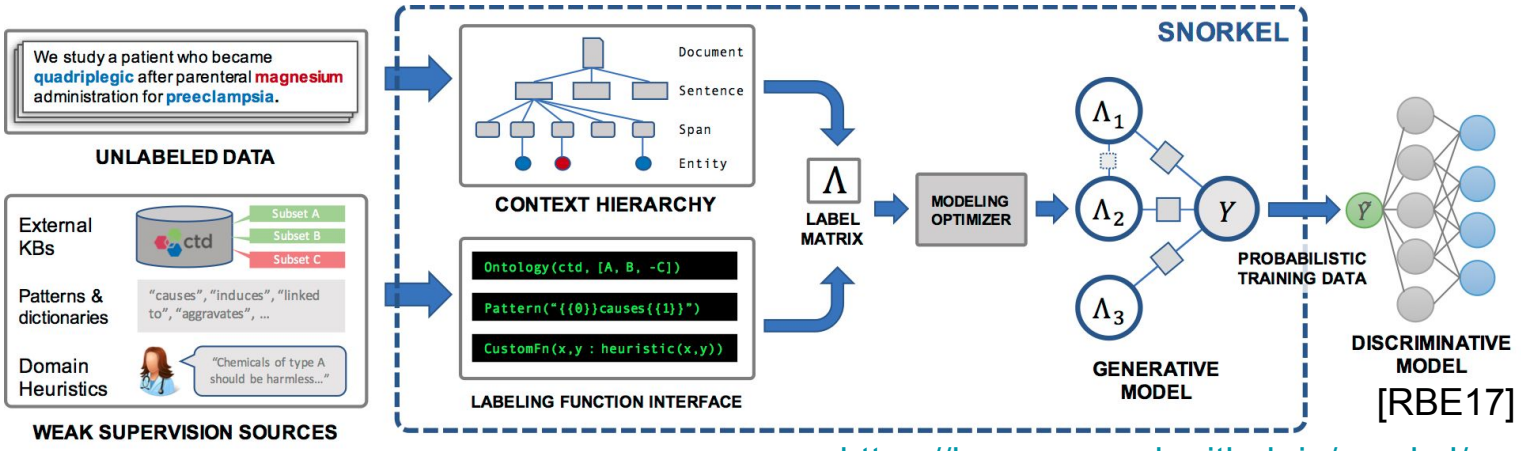
Picture taken from [22]

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

- *Labeling*

  Examples of tools
  - *Snorkel*
    - Standardizes labeling functions, working with alternative inputs
    - Collects propensity, accuracy and pairwise correlations between labeling functions, training a generative model that, in the end, creates probabilistic training labels.



[RBE17]

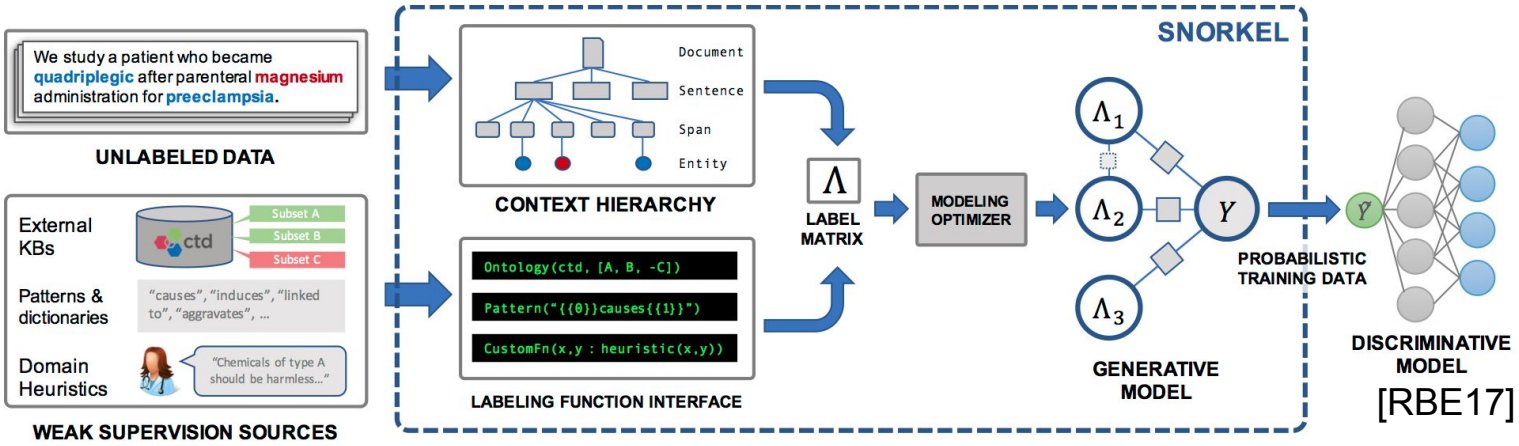
https://hazyresearch.github.io/snorkel/

A Canonical Architecture for AI-enhanced Applications: *Components*

Data Conditioning

- *Labeling*

  Examples of tools
  - *Snorkel*
    - The probabilistic labels are fed to a discriminative model that maps from features to labels, and hence, generalizes

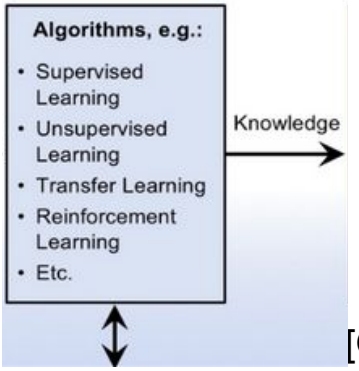


[RBE17]

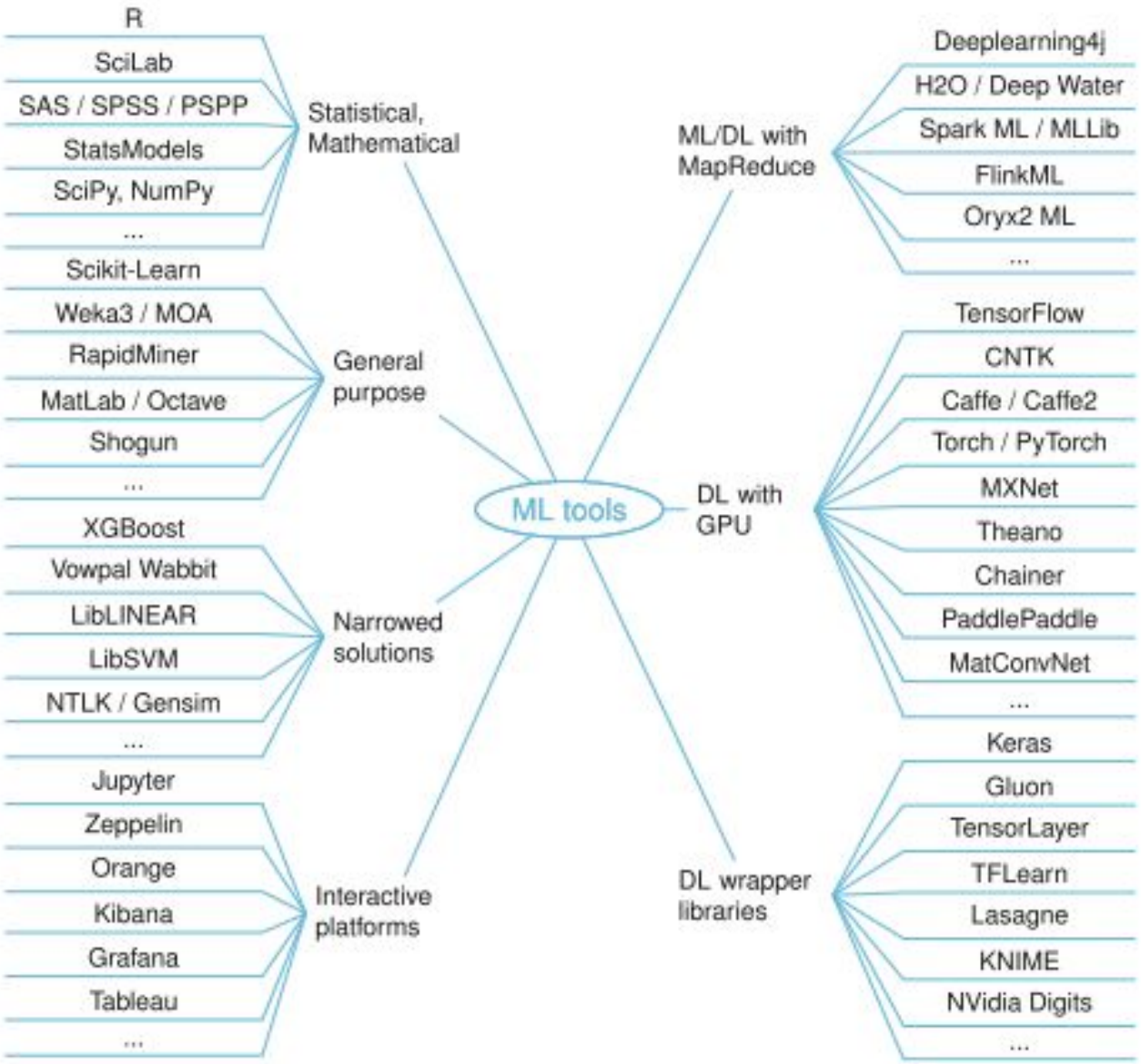
https://hazyresearch.github.io/snorkel/

A Canonical Architecture for AI-enhanced Applications: *Components* Algorithms,

- *Algorithms (ML Tools)*
  - Once the infrastructure is prepared, the core learning process can be framed.
  - Many tools available for the core learning…
    1. General purpose ML tools
    2. Narrow application ML tools
    3. ML/DL with Map Reduce
    4. DL wrapper libraries
    5. DL frameworks accelerated with GPUs and scale-out



[GGK19]



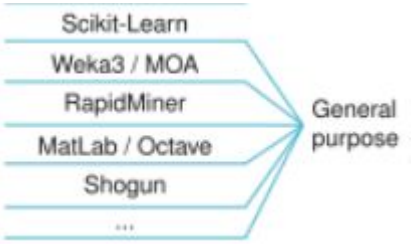
Others, like Deep Reinforcement Learning frameworks, or In-database ML are not shown in the picture.

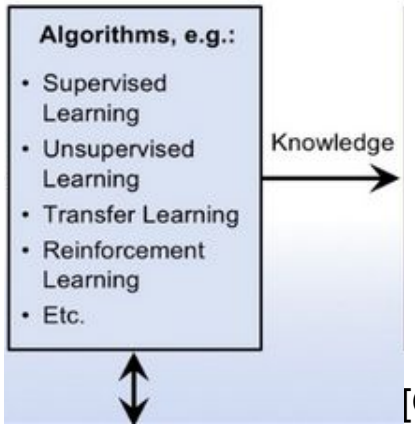
Picture taken from [NDB19]

A Canonical Architecture for AI-enhanced Applications: *Components*

Algorithms,

- *Algorithms (ML Tools)*
  - Many tools available for the core learning:
    1. General purpose ML tools
       - Goals:
         - Easy-to-use API-oriented offering of ML algorithms
       - Key features:
         - Extensible libraries
         - Some offer a desktop GUI
         - Efficient implementations
         - Exportable models

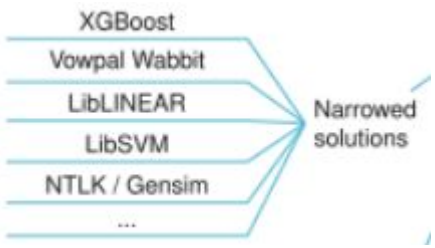




[GGK19]

A Canonical Architecture for AI-enhanced Applications: *Components* Algorithms,

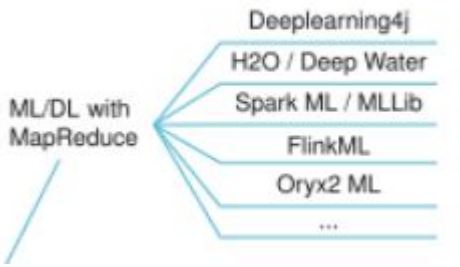
- *Algorithms (ML Tools)*
  - Many tools available for the core learning:
    2. Narrow application ML tools
       - Goals:
         Specialized tools that do not seek to be general, covering easy to use interfaces for a given task/model (e.g. graph embedding, topic models, NLP functionality, boosted tree models, contextual multi-arm bandits)
       - Key features:
         - Efficient support for specialized domains
         - Some offer MPI support for multi-CPU execution

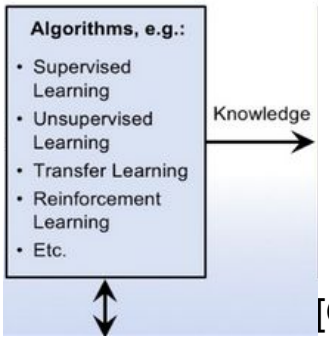


A Canonical Architecture for AI-enhanced Applications: *Components*

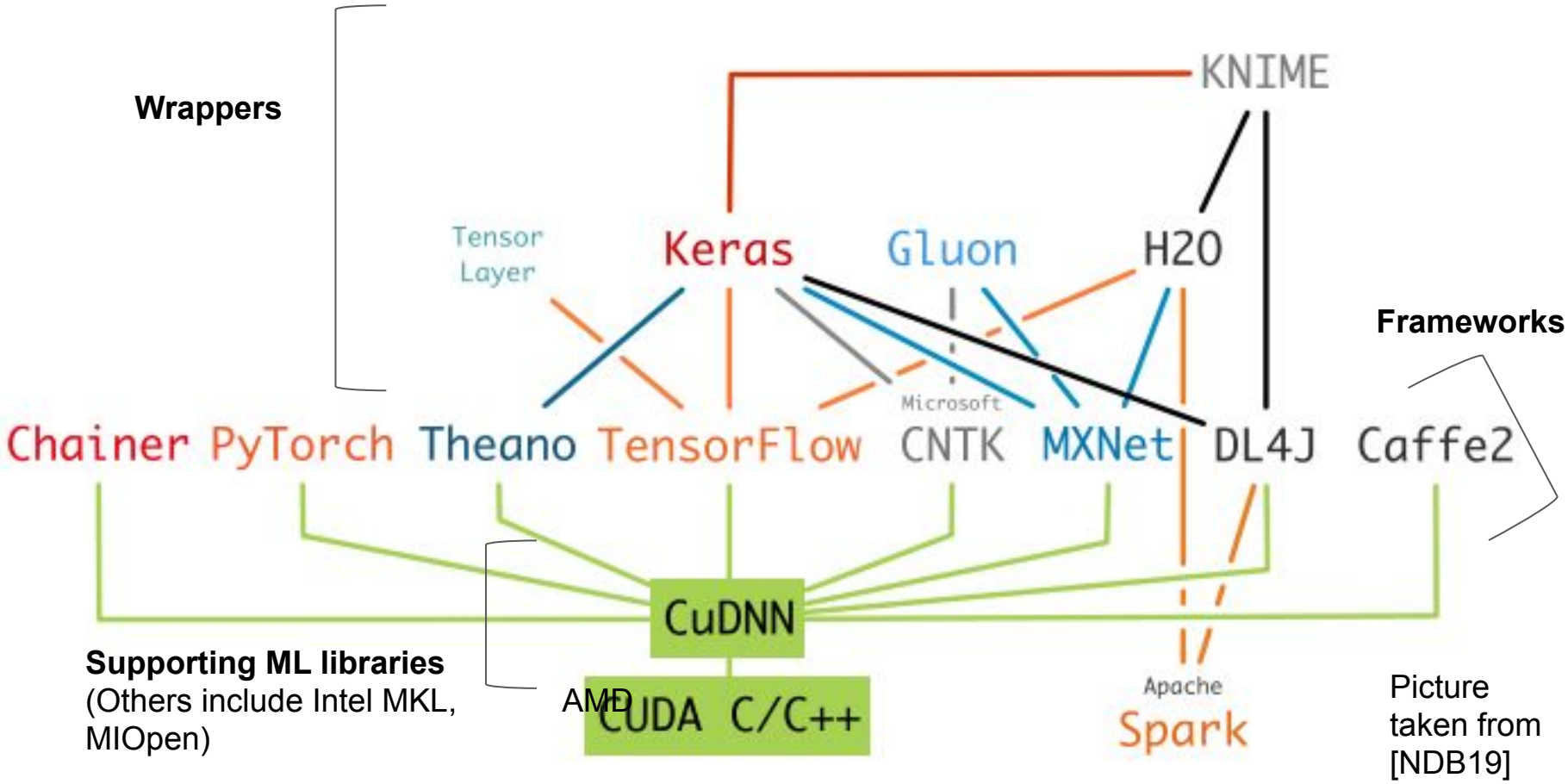
Algorithms,

- *Algorithms (ML Tools)*
  - Many tools available for the core learning:
    3. ML/DL with Map Reduce
       - Goals:
         Ease of large-scale processing for ML/DL
       - Key features:
       - Leverage the elasticity of the underlying frameworks
         Most commonly with Spark APIs





[GGK19]

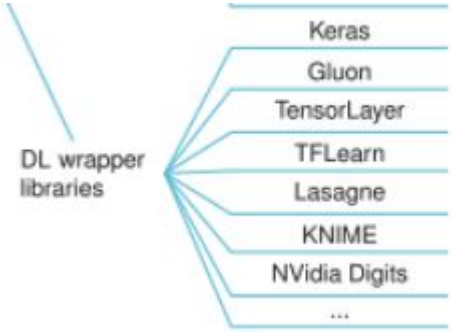


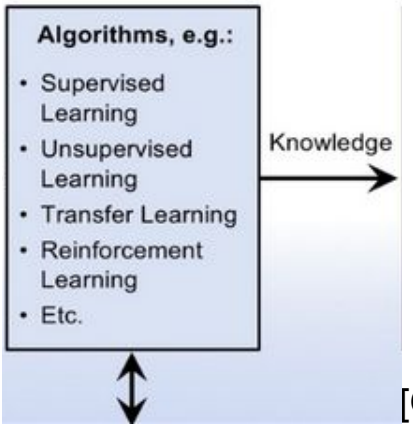
Regarding DL, most tools nowadays should be understood according to their level of abstraction and stack.

A Canonical Architecture for AI-enhanced Applications: *Components*

Algorithms,

- *Algorithms (ML Tools)*
  - Many tools available for the core learning:
    4. DL wrapper libraries
       - Goals:
         High-level abstractions for working with neural networks.
         They rely on a core DL framework.
         Key features:
         Some optimizations of the computation graph are introduced.

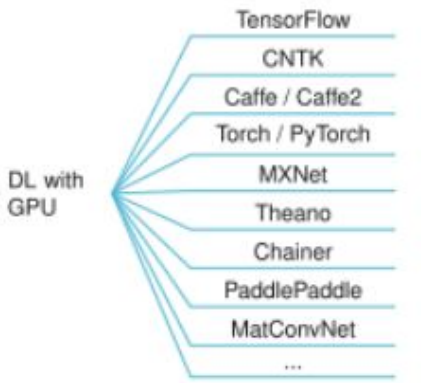


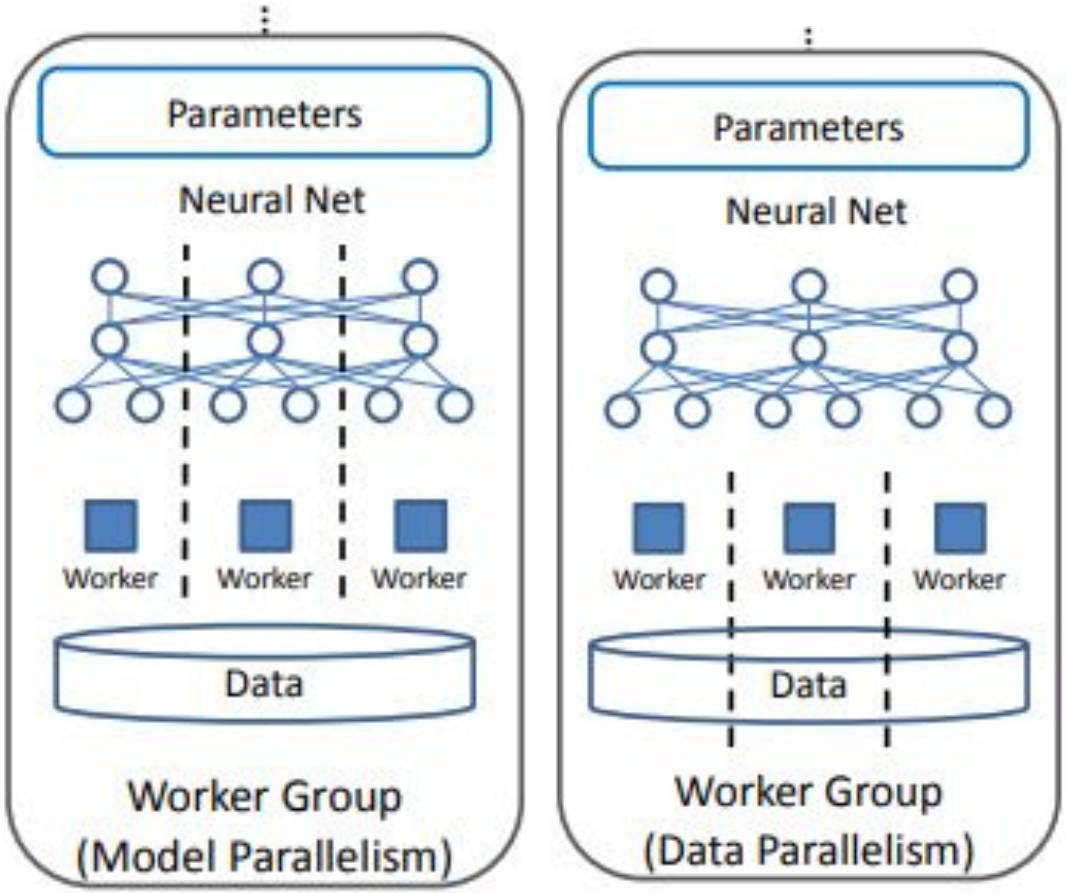


[GGK19]

A Canonical Architecture for AI-enhanced Applications: *Components* Algorithms,

- *Algorithms (ML Tools)*
  - Many tools available for the core learning:
    5. DL frameworks accelerated with GPUs and scale-out
       - Goals:
         Ease of use and efficient implementation of building blocks for using neural networks.
         The core abstractions that they offer are the types of layers, the activation functions and the optimizers. They can also offer different distribution mechanisms.
       - Key features:
         They are optimized for training, not inference.
         Some offer scale-out parallelism
         Most use GPUs for training acceleration





Data and model parallelism are the 2 approaches for scaling-out/distributing the training of very large neural networks.

In data parallelism, the training data is split among workers, each with a copy of the original model.

In model parallelism, partitions of the model are distributed to workers, each seeing all the data.

Workers synchronize (most of the times) for consistent updates.

Picture taken from [WCT15]

A Canonical Architecture for AI-enhanced Applications: *Components* Algorithms,

- *Algorithms (ML Tools)*
  - Many tools available for the core learning:
    6. Deep Reinforcement Learning frameworks
       - Goals:
         Offerings of deep reinforcement learning agents, their data management and training.
         At their core they are build with the idea to optimize the commonality between RL processes, including the management of the experience replay, the training process and the model checkpointing.
         This facilitates DRL, helping practitioners to not have to build models from scratch.

A Canonical Architecture for AI-enhanced Applications: *Components* Algorithms,

- *Algorithms (ML Tools)*
  - Many tools available for the core learning:
    6. Deep Reinforcement Learning frameworks
       - Key features:
         Based on DL frameworks
         Some offer offline learning,
         multi-agents, imitation learning.

Examples: Ray-RLlib, Google Dopamine, Intel Nervana Coach (AWS-SagemakerRL), Facebook Horizon

Algorithm Family	Policy Evaluation	Replay Buffer	Gradient-Based Optimizer	Other Distributed Components
DQNs	X	X	X	
Policy Gradient	X		X	
Off-policy PG	X	X	X	
Model-Based/Hybrid	X		X	Model-Based Planning
Multi-Agent	X	X	X	
Evolutionary Methods	X			Derivative-Free Optimization
AlphaGo	X	X	X	MCTS, Derivative-Free Optimization

Common abstractions for Scalable Reinforcement Learning [LLM17]

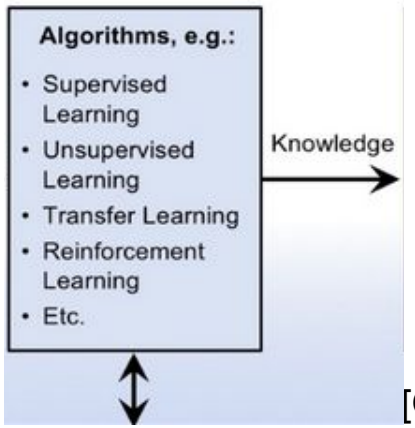
A Canonical Architecture for AI-enhanced Applications: *Components* Algorithms,

- *Algorithms (ML Tools)*
  - Many tools available for the core learning:
    7. In-database machine learning
       - Goal: These are tools that aim to perform machine learning tasks, covering the steps of model creation, model exploration and model inference, inside a database system. The key idea is to avoid data transfer to-and-from an external ML tool.
       - Key features:
         The model is stored in the database (usually relational), and the training process happens there.
         Examples include: MADLib, Riot-DB, etc.

    We talk about this later.

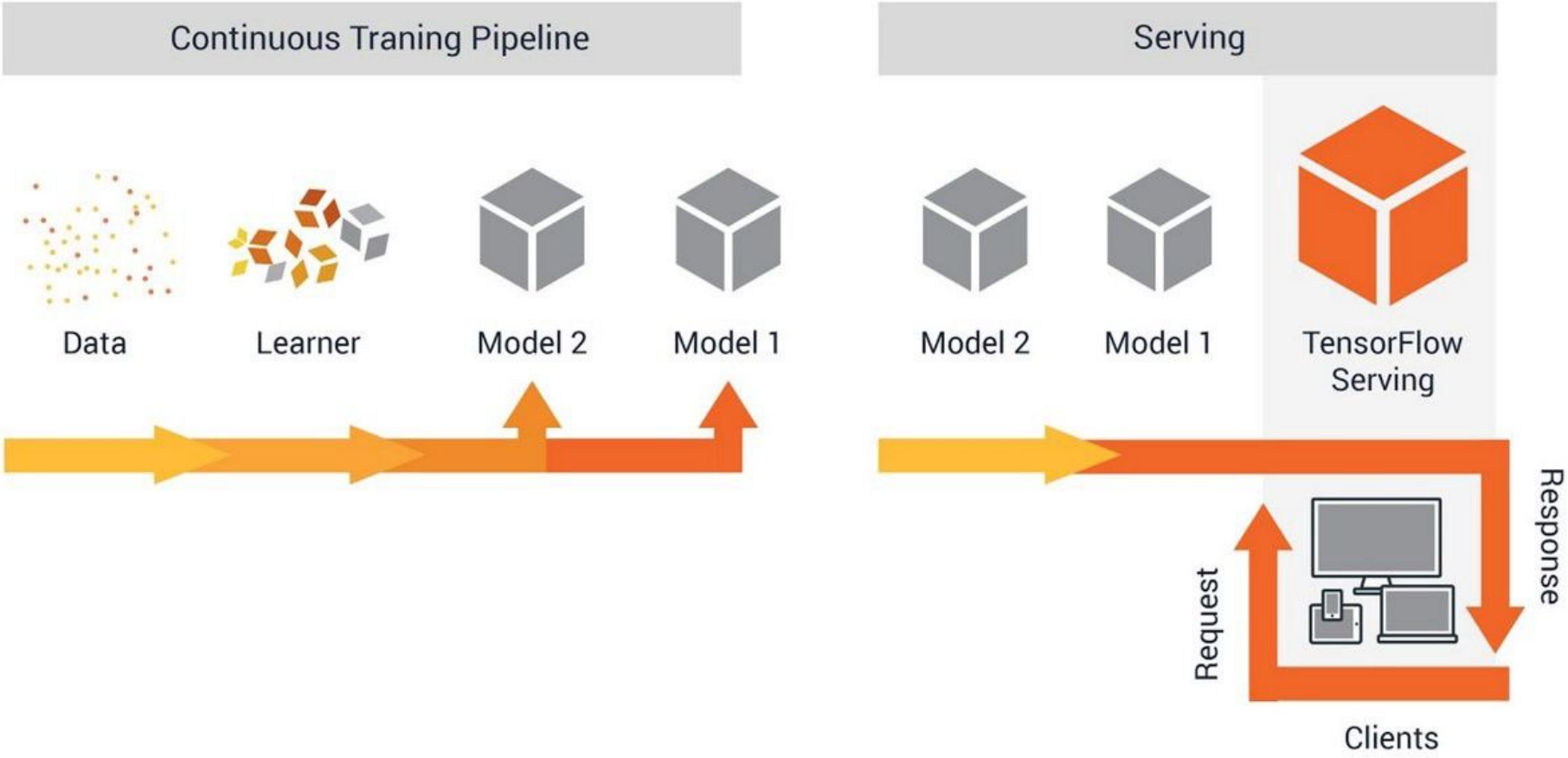
A Canonical Architecture for AI-enhanced Applications: *Components* Algorithms,

- *Algorithms (ML Tools)*
  - Apart from core tools, to support the process, the following tools exist:
    1. Model serving systems
       - Goal: Once a model is trained, these systems prepare a runtime for the model, such that requests for predictions can be served efficiently.
       - Key features:
         Exchange of models is supported
         Support for containerization
         Examples: TF Serving, Clipper, MLFlowModels, Velox.



[GGK19]

Serve models in production with TensorFlow Serving



Picture taken from [32]

A Canonical Architecture for AI-enhanced Applications: *Components* Algorithms,

- *Algorithms (ML Tools)*
  - Apart from core tools, to support the process, the following tools exist:
    2. ML model management systems
       - Goal: These are novel systems that aim to help in tracking, storing, and indexing a large number of ML models for later sharing, querying and analysis. They help to monitor model decay.
       - Key features:
       - Logging the metrics achieved after training (e.g. loss, F1-score) or in provided tests; and the hyper-parameters.
       - Pipeline and model search:
         The first one allows to visualize the process of a ML application, with all the steps of data creation, and changes in the models. The latter allows to zoom into models.

A Canonical Architecture for AI-enhanced Applications: *Components*

Algorithms,

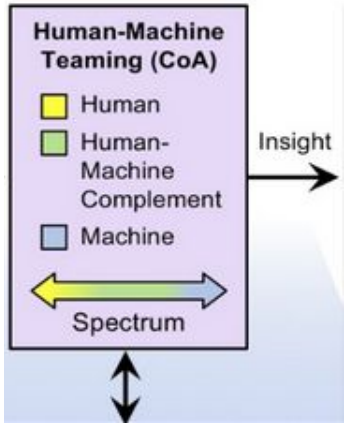
- *Algorithms (ML Tools)*
  - Apart from core tools, to support the process, the following tools exist:
    2. ML model management systems

       Examples: ModelDB, Tensorflow TFX, Weights and Biases, SAS Model Manager

A Canonical Architecture for AI-enhanced Applications: *Components*

Human-Machine Teaming (CoA)

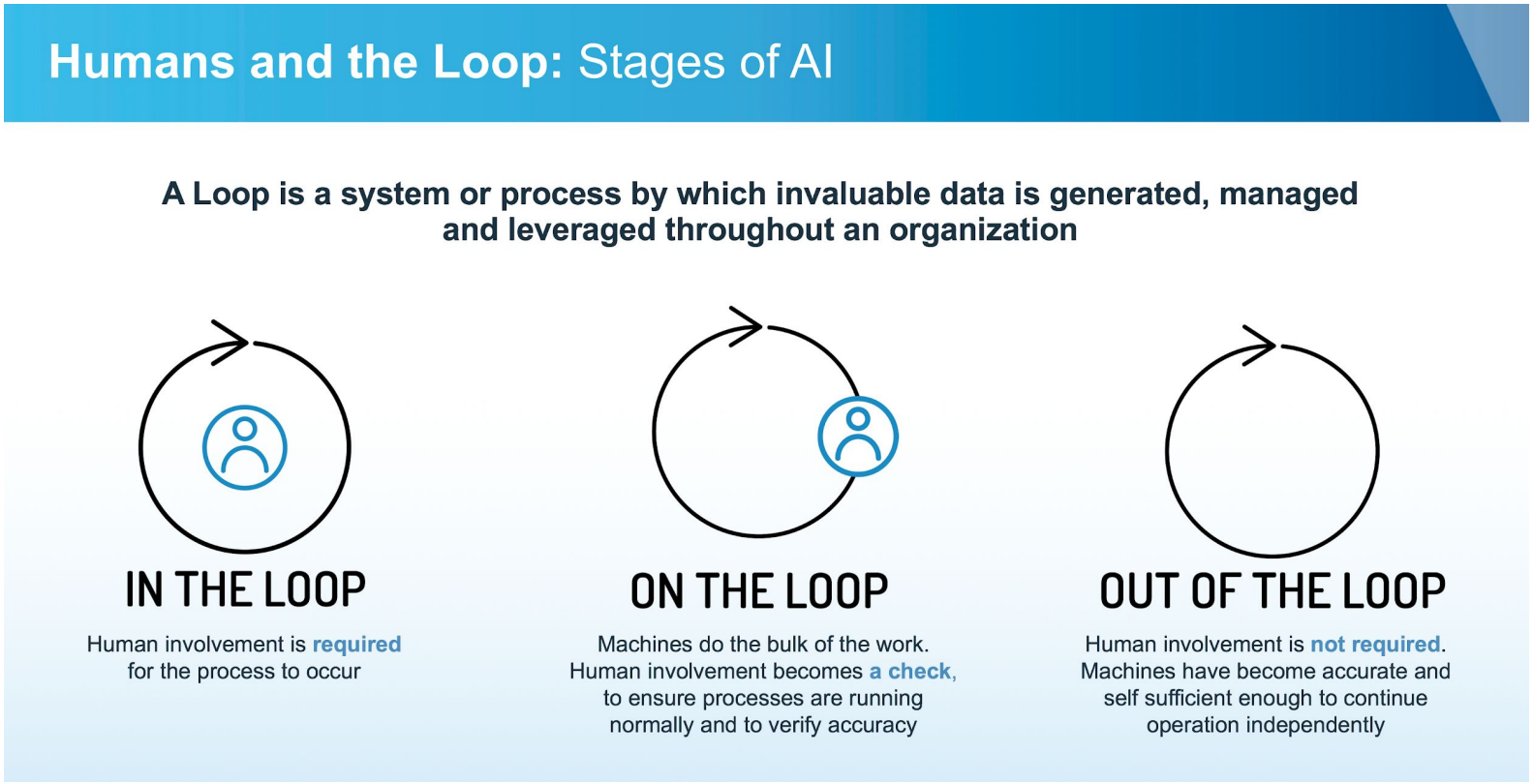
- *Human-Machine Teaming*
  - Humans can be involved in enabling the systems to make use of the insights, or on evaluation.
  - Different approaches: in, on and out of the loop.
  - Interface design is essential for this component.



A Canonical Architecture for AI-enhanced Applications: *Components*

Human-Machine Teaming (CoA)

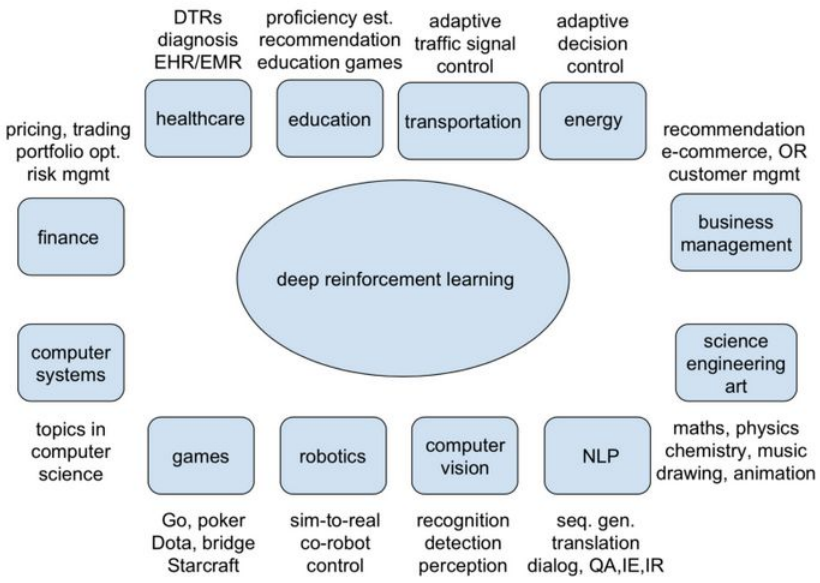
- *Human-Machine Teaming*



Different degrees of human involvement in AI-enabled applications. Source: [33]

A Canonical Architecture for AI-enhanced Applications: *Components*

- *Components:*
  - *Users (Missions)*
    - Missions exist in many domains. They determine the actual objectives of learned models.

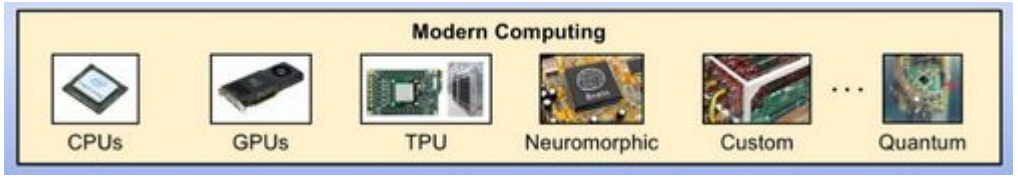


Applications of Deep Reinforcement Learning [Li17]

[GGK19]

A Canonical Architecture for AI-enhanced Applications: *Components*

- *Components:*
  - *Modern computing*
    - GPUs: Offer massive parallelism for deep learning.
    - TPUs: Tensor Processing Units: These are AI accelerator application-specific integrated circuits, developed by Google specifically for deep learning. TPUs are designed for high volume low precision computation, with higher input/output operations per Joule, and without hardware for rasterisation/texture mapping.
      - Cloud version
      - Edge version



[GGK19]

CoA = Courses of Action GPU = Graph Processing Unit TPU = Tensor Processing Unit

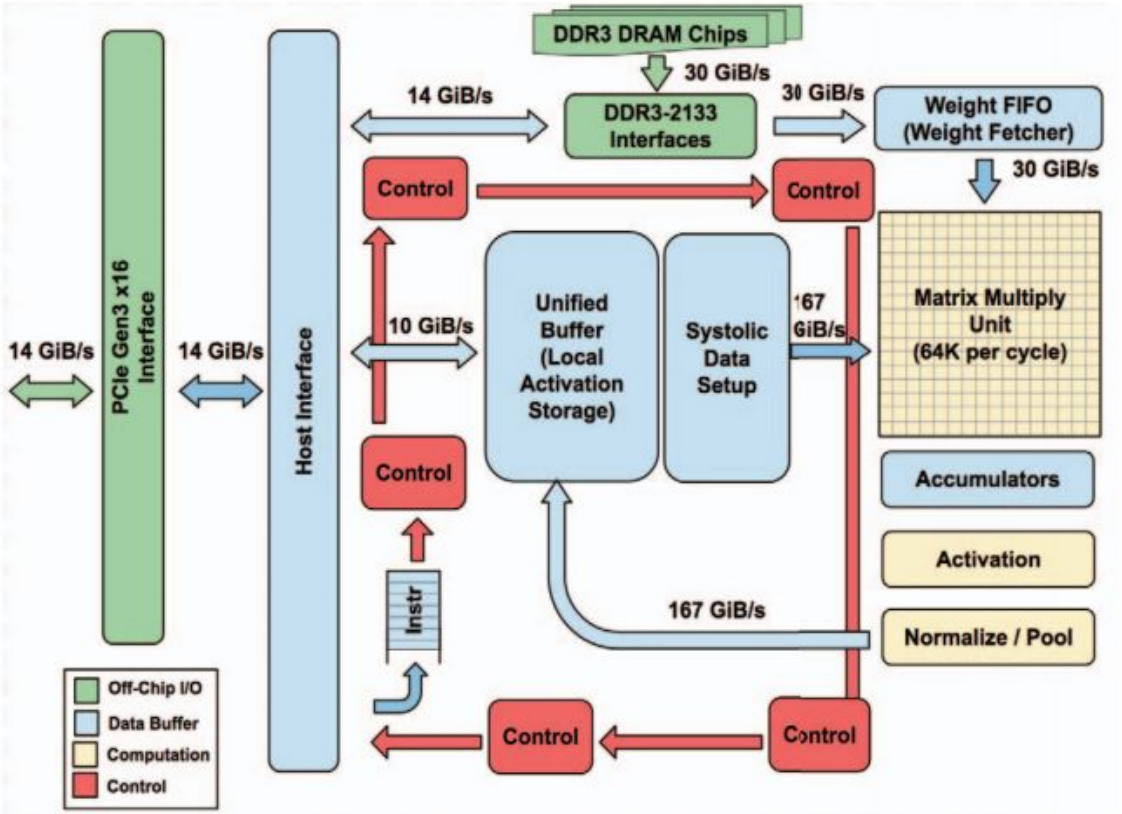


Figure 1. TPU Block Diagram. The main computation is the yellow Matrix Multiply unit. Its inputs are the blue Weight FIFO and the blue Unified Buffer and its output is the blue Accumulators. The yellow Activation Unit performs the nonlinear functions on the Accumulators, which go to the Unified Buffer.

The architecture of TPUs is adapted to the use case.

TPUs seek to serve FCN, CNN and RNNs, while other designs focus on CNNs.

Some applications in benchmarks are shown to be memory bound.

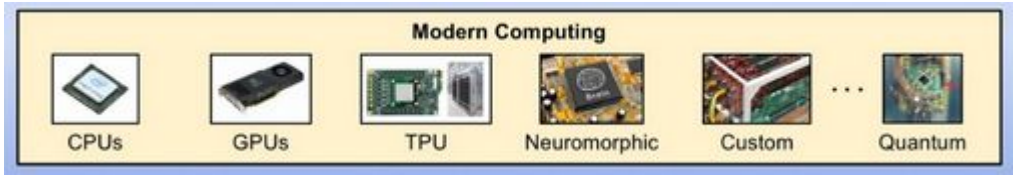
Efficient energy usage.

Other custom processors similar to TPUs include, Microsoft's Brainwave and Intel Nervana Neural Network Processors

Picture taken from [JYP19]

A Canonical Architecture for AI-enhanced Applications: *Components*

- *Components:*
  - *Modern computing*
    - Neuromorphic Computing: This is an approach to processor design based on the attempt to mimic neural networks in the human brain.
      - Processors consist of neurons and synapses.
    - Massive parallelism, large amount of connections.
    - Some examples: Intel Loihi, IBM TrueNorth, SpiNNaker



[GGK19]

CoA = Courses of Action GPU = Graph Processing Unit TPU = Tensor Processing Unit

A Canonical Architecture for AI-enhanced Applications: *Components*

- *Components:*
  - *Robustness Evaluation*
    - ML model management systems should help users in evaluating the robustness of ML applications.
      Dedicated tools and standards for evaluating ML models are still emerging.

[GGK19]

From the tools that we have mentioned:

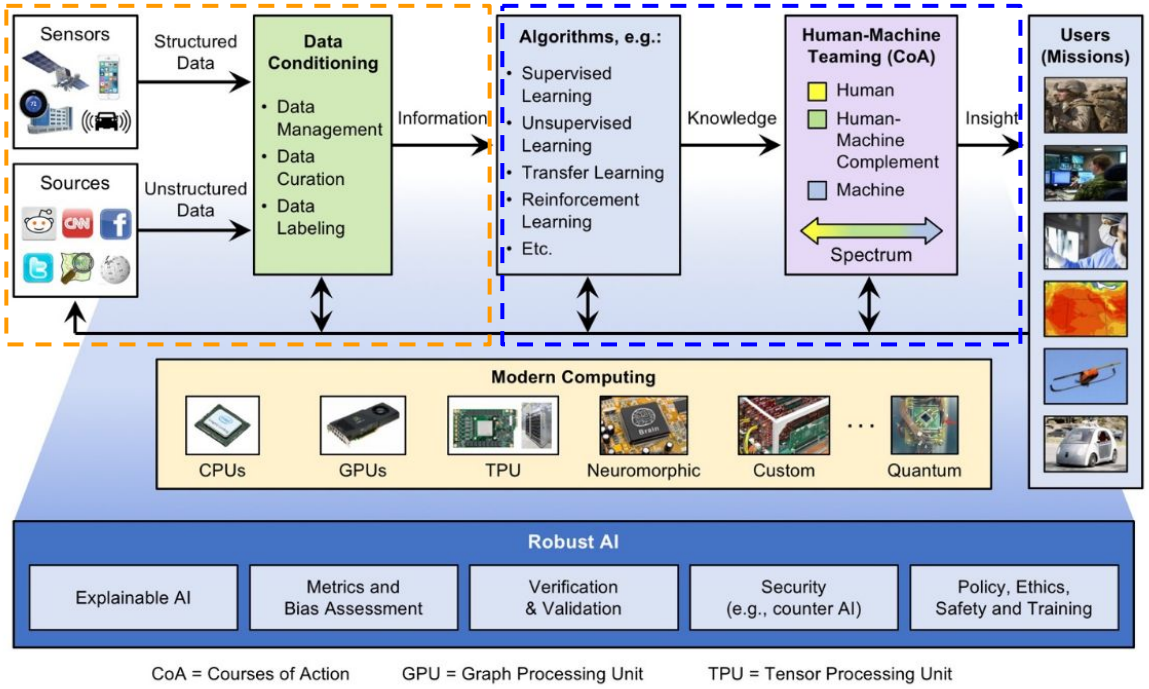
What tools have you used?

What tools do you find missing from the list?

A Canonical Architecture for AI-enhanced Applications

- From the components discussed we can see that some of those are traditionally covered by the ML community, and some by the data management community.

Traditionally Studied by the ML Community



Traditionally Studied by the Systems/Data Management Community

This architecture, shows **robustness** as an overall concern [GGK19]

Systems + ML = SysML

- In the canonical AI architecture there are many cross-cutting concerns:
  - **Efficiency** of ML tools
  - **Accessibility** of ML tools
  - **Robustness** for AI solutions
  - **How can data management** be improved to serve ML better **(DB4AI)**?
  - **How to apply ML in data management (AI4DB)?**

- These are propiciating a growing amount of research at the intersection of data management and ML (**SysML**)

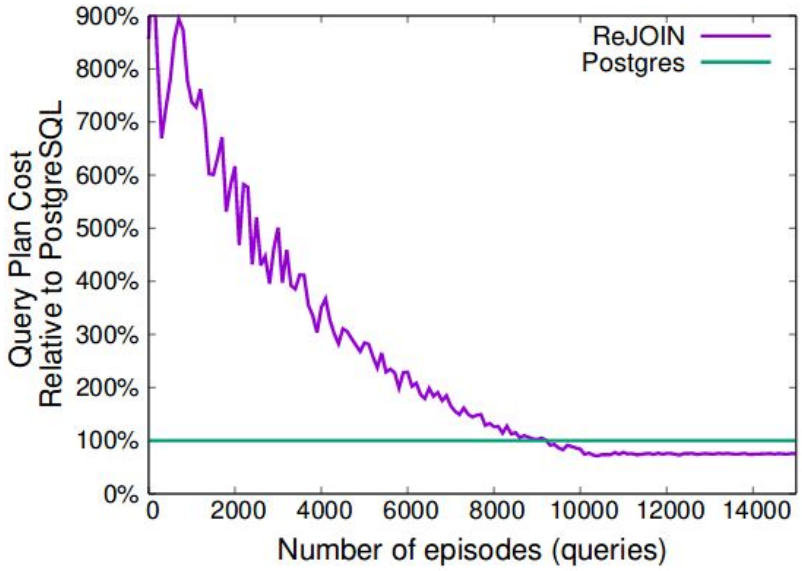
- In this lecture we will cover some ongoing developments.

So far we have motivated the importance of DB4AI,
but what about AI4DB?

What can ML bring to data management? (AI4DB)

- Database tools, in their implementation often rely on heuristics
  Heuristic: "any approach to problem solving or self-discovery that employs a practical method, not guaranteed to be optimal, perfect, logical, or rational, but instead sufficient for reaching an immediate goal" Source: Wikipedia.
- Heuristics can grow complex when trying to change them from specific to general cases
- Heuristics that attempt to be general (e.g. avoiding worst-case scenarios) can be un-optimal when the assumptions change.
  - Example: Search on a sorted array

Why can ML bring to data management?



Results on the Join Order Benchmark PostgreSQL vs. Learning from experience with ReJoin (DRL)

Picture taken from [MP18]

=> *"Since **real-world systems are difficult to model** accurately, state-of-the-art systems often rely on human-engineered **heuristic algorithms** that can **leave significant room for improvement"*** [MPP19]

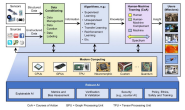
Summary

- Several factors are driving the **interest in adopting AI for optimizing applications**: success at grand challenges, modern hardware, technical and tool developments.
- To better understand the complete process, a **canonical architecture** is useful.
  - The **overall process** is to go from data to knowledge, to insights, and to act wisely in user missions.
  - **Components** include:
    - Sources and sensors,
    - Data conditioning
    - AI-algorithms
    - Human-machine teaming
    - User missions
    - Modern hardware
    - Tools for robust AI.
  - We reviewed the series of components, giving some examples of current tools for each component.

Summary

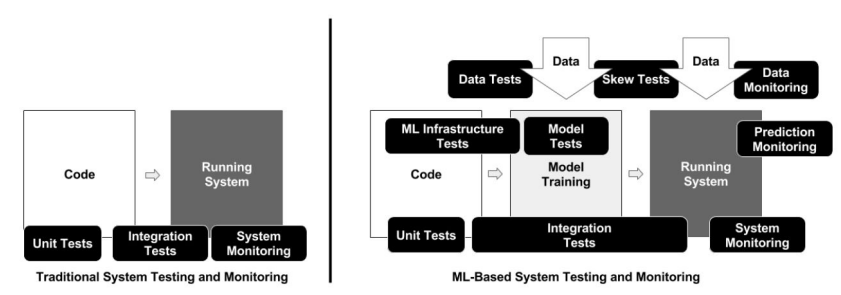
- **SysML:** Novel AI-based applications require research that brings together ML and Systems communities.
  - DB4AI
  - AI4DB
    - ML can bring benefits to data management, by making the process improve over heuristics.

- So far we have considered:
  - The motivating factors for the increased adoption of AI
  - Basic categories of ML
  - A canonical AI-enabled architecture
  - The new field of SysML
- In the next slides we consider (very briefly), some emerging configurations and guidelines for AI in production



Part 1. Introduction

4. Emerging Configurations and Guidelines for AI in production systems



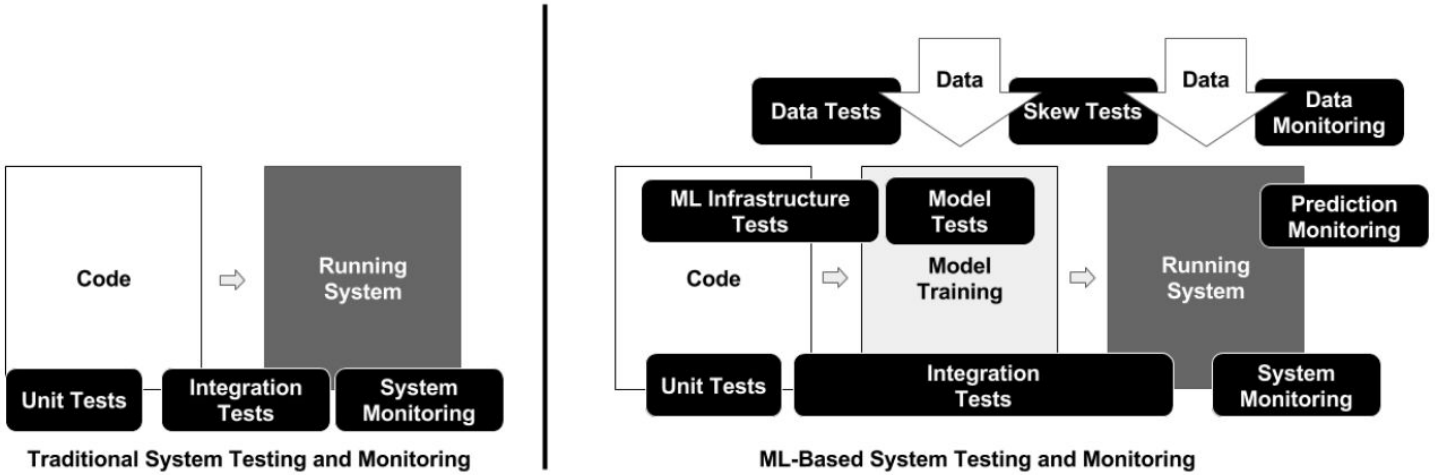
ML in production

- The adoption of ML in production faces many challenges
  - Replacing hard-coded program logics with adaptive code is a big shift!

Picture taken from [24]

    - The behavior of ML tools is not known in advance
- Tools and guidelines to support the complete lifecycle of ML in production are slowly emerging

ML in production



Assessing systems with the ML test score [BSN17]

- Models need to be monitored to avoid *Training-serving skew.*
- *Training-serving skew* is a difference between performance during training and performance during serving (i.e., when the model is already built). This skew can be caused by:
  - differences in how data is managed
  - changes in the data
  - some impact between the new model and the process that generates input data for it.

ML in production

- Some example for conditions that might cause training-serving skew:
  - A forecasting model is trained on data that does not consider months with holidays, but is deployed during Christmas
  - A model trained to predict the runtime of a given operation, on a single-user system, over a chosen dataset and a specific hardware; is finally deployed on the same dataset and hardware, but a varying number of concurrent users

Rules for ML

- Practitioners [Zin19] have proposed a series of rules that capture the domain expectations on what is a good practice for solving problems with ML.
- The basic development approach is:
  - Make sure your pipeline is solid end to end
  - Start with a reasonable objective
  - Add common-sense features in a simple way
  - Make sure that your pipeline stays solid
- Some of these rules include:
  - #2 Start with metrics, making sure they are close to the objectives
    - Objective: Something that you care about, but that you might not be able to directly optimize with your model.
    - Metric: The measurement that the model optimizes when training.
  - #3 Choose ML over a complex heuristic
  - #16 Plan to launch and iterate

The ML Test Score [BSN17]

1. Feature and data tests
   - Costs and reliability of features
   - Data distribution tests (features vs. target), incl. feature correlation
   - Tests for feature-creation process and code
2. Model tests
   - Model repository
   - Relationship between proxy (offline) and actual metrics
   - Optimal hyper-parameter evaluation
   - Model staleness
   - Test against simpler models
3. ML infrastructure tests
   - Reproducibility of training
   - Efficiency at model rollbacks
4. Monitoring tests
   - Test for alerts on unstable cases
   - Test for training/production skew

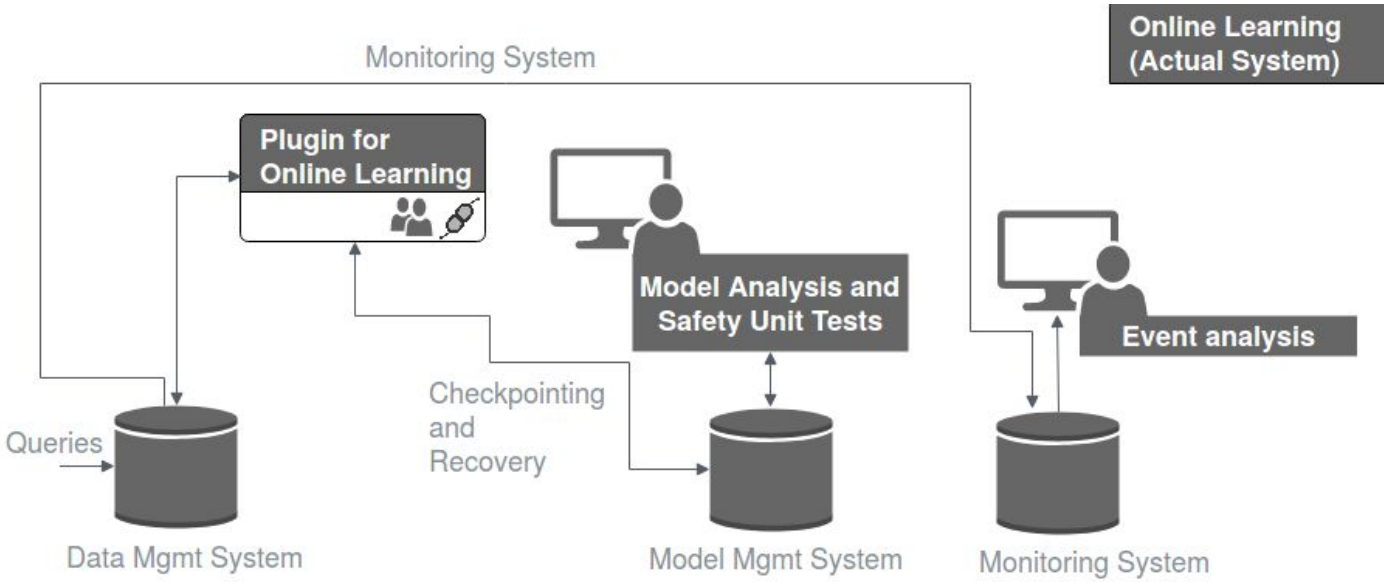
Emerging configurations

After discussing about guidelines for ML in production, we consider now some generic emerging configurations that could be adopted in applying AI with databases.

Our focus is on self-driving tools, but the configurations can be adapted to other cases.

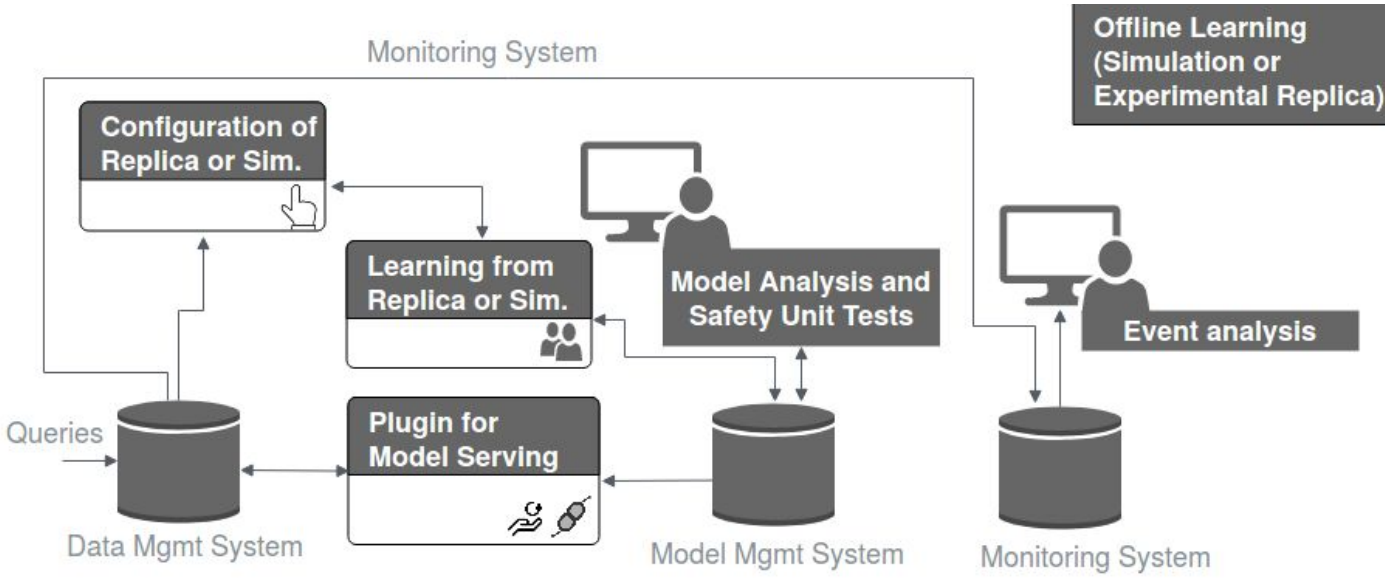
Emerging configuration #1

Online learning



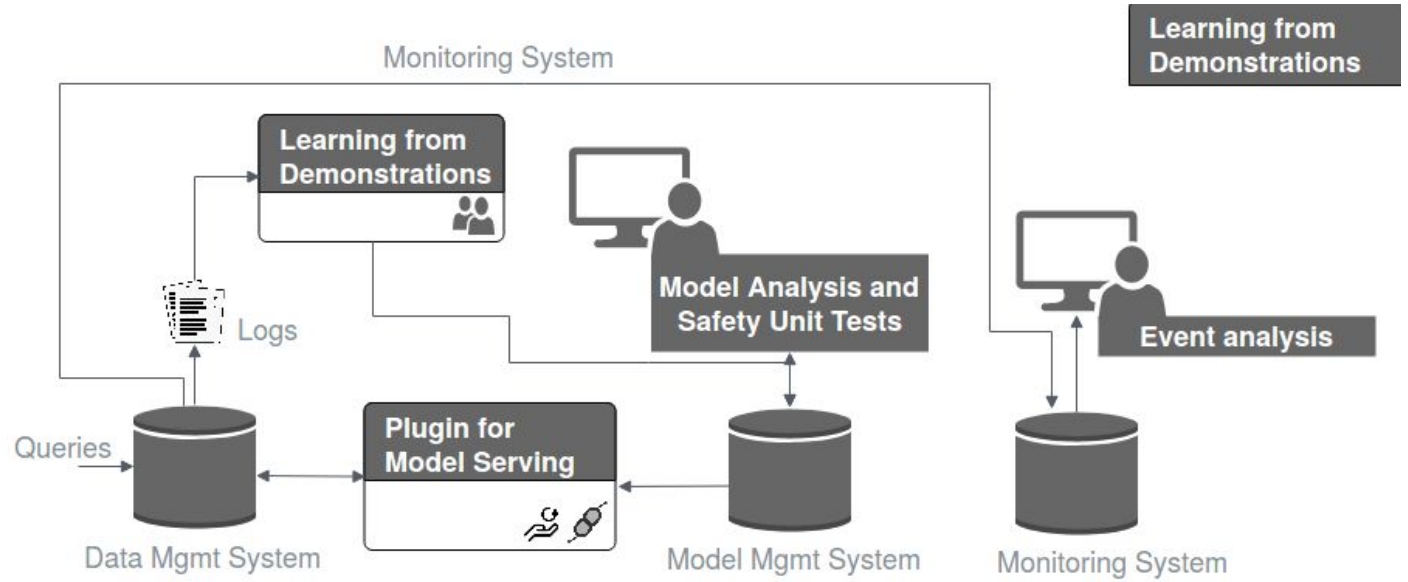
Emerging configuration #2

Offline learning



Emerging configuration #3

Learning from demonstrations

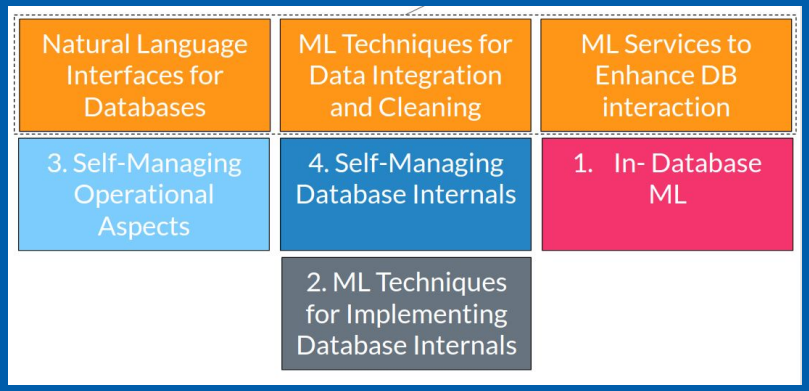


Summary

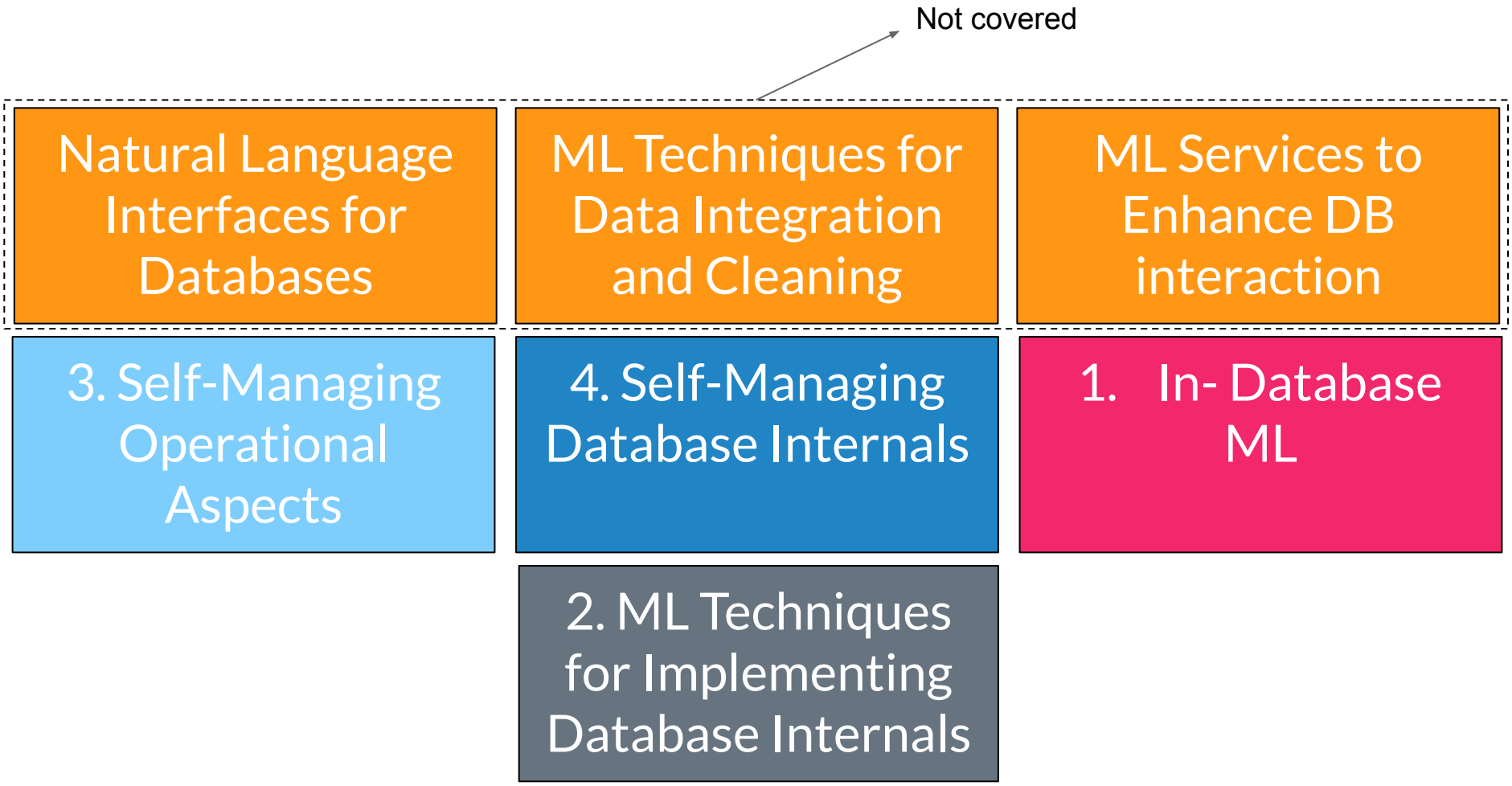
- ML in production differs from traditional systems
- The core common-sense approach for development is: start simple, be certain about your infrastructure, test and iterate.
- An ML test score can be adopted to quantify the goodness of an implementation
- 3 emerging configurations can be considered (with a focus on self-driving components): offline, online and learning from demonstrations.
- To date, they have not been fully developed for data management applications.

Part 2.

Recent Applications of AI techniques in Data Management



Overview


Not covered
Natural Language Interfaces for Databases
ML Techniques for Data Integration and Cleaning
ML Services to Enhance DB interaction
3. Self-Managing Operational Aspects
4. Self-Managing Database Internals
1. In- Database ML
2. ML Techniques for Implementing Database Internals

Part 2. Recent Applications of AI techniques in Data Management

Sec 1. In-Database Machine Learning (DB4AI)

```
SELECT madlib.logregr_train(
    'patients',                         -- source table
    'patients_logregr',                 -- output table
    'second_attack',                    -- labels
    'ARRAY[1, treatment, trait_anxiety]',  -- features
    NULL,                               -- grouping columns
    20,                                 -- max number of iteration
    'irls'                              -- optimizer
    );
```

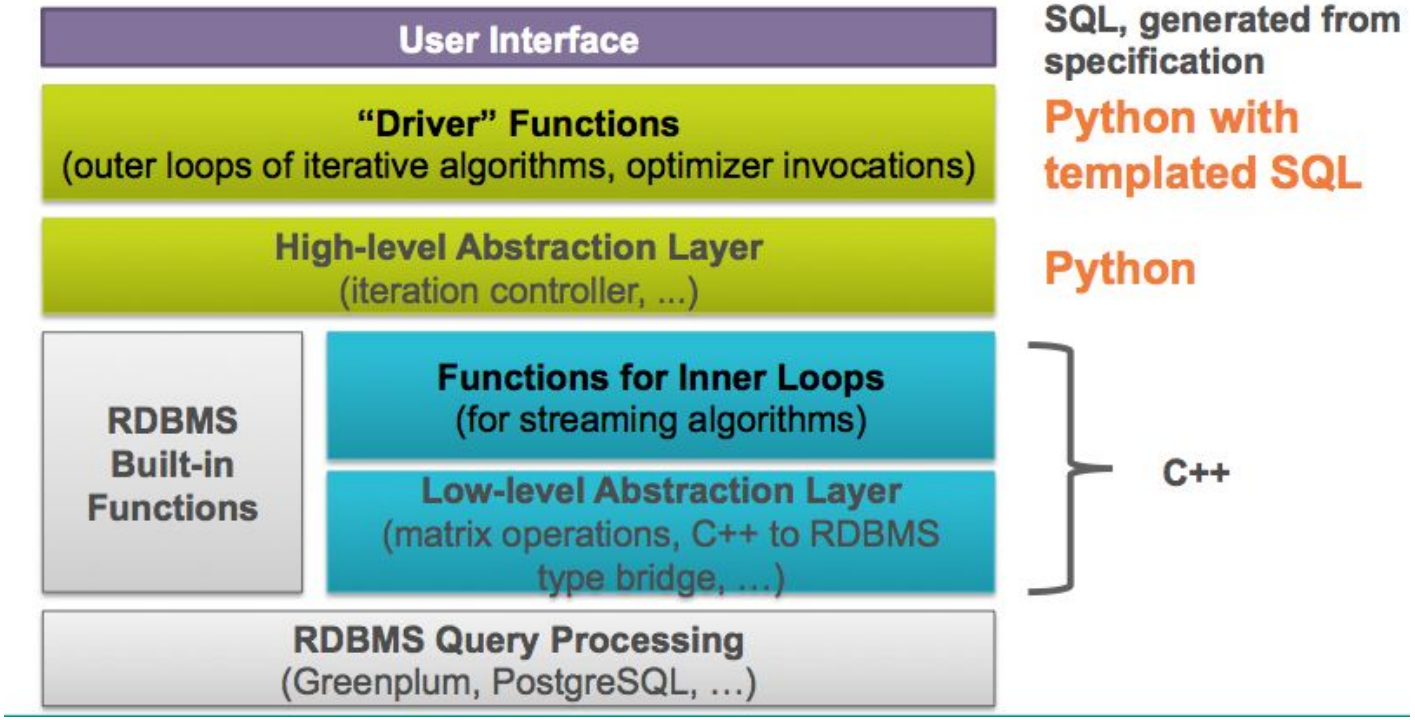
In-Database ML

- Enterprises have large amounts of labeled data, stored in databases
- Analyzing it with ML brings business potential
- In-Database ML
  - Brings the computation to the data
  - Leverages optimizations from the DBMS
  - Could ease the work for larger-than-memory data

1. In- Database ML

In-Database ML

- Some examples: MADLib, SAP PAL from Leonardo
- MADLib: Magnetic, Agile, Deep

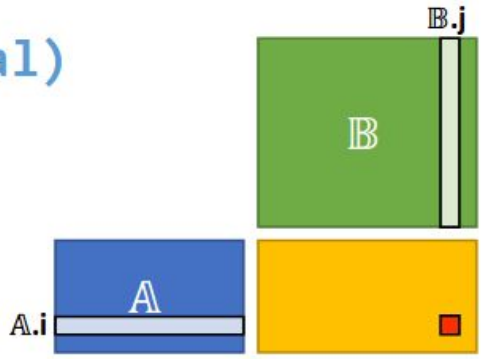


Picture taken from [34]

In-Database ML: Under the hood

- Naive Matrix Multiplication in SQL

```
SELECT A.i, B.j, SUM(A.val*B.val)
FROM A, B
WHERE A.j = B.i
GROUP BY A.i, B.j;
```



i: row number, j: column number

0 vals are not stored, so it is a sparse representation

Picture taken from [KBY17].

Related material in [CDD09]

In-Database ML

- Naive Matrix Multiplication in SQL

  An example:

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} \quad A \times B = \begin{pmatrix} 4 & 5 \\ 6 & 8 \end{pmatrix}$$

A

i	*j*	*val*
0	1	1
1	0	1
1	1	1

B

i	*j*	*val*
0	0	2
0	1	3
1	0	4
1	1	5

```
SELECT A.i, B.j, SUM(A.val*B.val)
FROM A, B
WHERE A.j = B.i
GROUP BY A.i, B.j;
```

i: row number, j: column number

0 vals are not stored, so it is a sparse representation

A x B

A.j=B.i	*A.i*	*B.j*	*A.val*B.val*
1	0	0	1*4
1	0	1	1*5
0	1	0	1*2
0	1	1	1*3
1	1	0	1*4
1	1	1	1*5

In-Database ML

- Naive Matrix Multiplication in SQL

  An example:

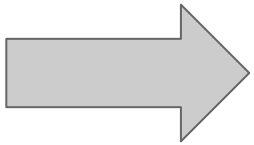
$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} \quad A \times B = \begin{pmatrix} 4 & 5 \\ 6 & 8 \end{pmatrix}$$

```
SELECT A.i, B.j, SUM(A.val*B.val)
FROM A, B
WHERE A.j = B.i
GROUP BY A.i, B.j;
```

i: row number, j: column number

0 vals are not stored, so it is a sparse representation

A.j=B.i	A.i	B.j	A.val*B.val
1	0	0	1*4
1	0	1	1*5
0	1	0	1*2
0	1	1	1*3
1	1	0	1*4
1	1	1	1*5



GROUP BY A.i, B.j

And SUM

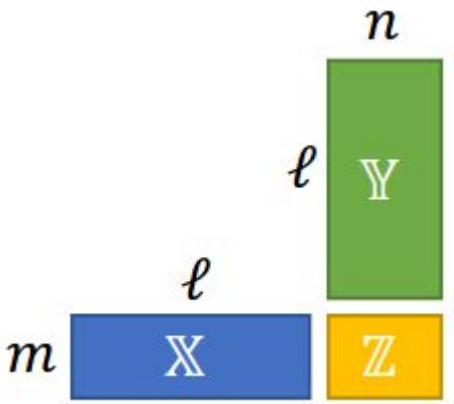
A.j=B.i	A.i	B.j	SUM(A.val*B.val)
1	0	0	4
1	0	1	5
0,1	1	0	1*2+1*4=6
0,1	1	1	1*3+1*5=8

In-Database ML

- Improved Matrix Multiplication
  - Better representation
  - UDFs

```
A(i,row), B(j,col)
```

```
SELECT A.i, B.j, dotproduct(A.row, B.col)
FROM A, B;
```

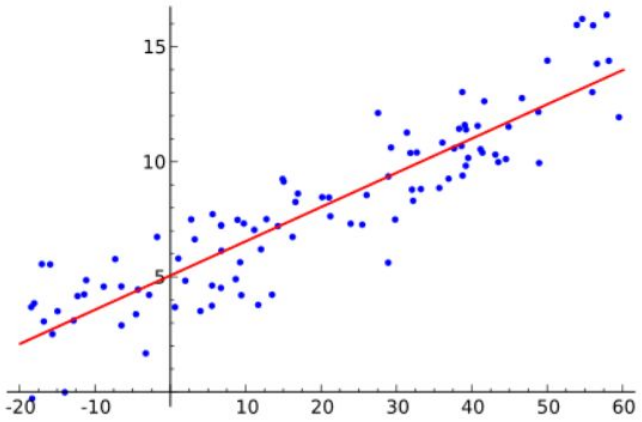


Picture taken from [KBY17].

Related material in [CDD09]

In-Database ML

- Authors show that some building blocks of ML can be computed using such UDF-based approaches:
  - Ordinary Least Squares
  - Gradient Descent
  - ...
- 3 core components building on the former ones:
  - Matrix operations
  - Pipeline operations
  - Algorithms



Buildinb blocks presented in [CDD09]

In-Database ML

- MADLib: Interface

```
CREATE TABLE patients( id INTEGER NOT NULL,
                       second_attack INTEGER,
                       treatment INTEGER,
                       trait_anxiety INTEGER);

INSERT INTO patients VALUES
(1,     1,      1,      70),
(3,     1,      1,      50),
(5,     1,      0,      40),
(7,     1,      0,      75),
(9,     1,      0,      70),
(11,    0,      1,      65),
(13,    0,      1,      45),
(15,    0,      1,      40),
(17,    0,      0,      55),
(19,    0,      0,      50),
(2,     1,      1,      80),
(4,     1,      0,      60),
(6,     1,      0,      65),
(8,     1,      0,      80),
(10,    1,      0,      60),
(12,    0,      1,      50),
(14,    0,      1,      35),
(16,    0,      1,      50),
(18,    0,      0,      45),
(20,    0,      0,      60);
```

Model Creation

```
SELECT madlib.logregr_train(
    'patients',                                 -- source table
    'patients_logregr',                         -- output table
    'second_attack',                            -- labels
    'ARRAY[1, treatment, trait_anxiety]',       -- features
    NULL,                                       -- grouping columns
    20,                                         -- max number of iteration
    'irls'                                      -- optimizer
    );
```

Picture taken from [34]

In-Database ML

- MADLib: Interface

```
-- Set extended display on for easier reading of output (\x is for psql only)
\x on
SELECT * from patients_logregr;

-- ************ --
--    Result    --
-- ************ --
coef                    | [-6.36346994178187, -1.02410605239327, 0.119044916668606]
log_likelihood          | -9.41018298389
std_err                 | [3.21389766375094, 1.17107844860319, 0.0549790458269309]
z_stats                 | [-1.97998524145759, -0.874498248699549, 2.16527796868918]
p_values                | [0.0477051870698128, 0.38184697353045, 0.0303664045046168]
odds_ratios             | [0.0017233763092323, 0.359117354054954, 1.12642051220895]
condition_no            | 326.081922792
num_rows_processed      | 20
num_missing_rows_skipped | 0
num_iterations          | 5
variance_covariance     | [[10.3291381930637, -0.47430466519573, -0.171995901260052], [-0.47430466519573, 1.37142473278285, -0.00119520703381598], [-0.1719959

-- Alternatively, unnest the arrays in the results for easier reading of output (\x is for psql only)
\x off
SELECT unnest(array['intercept', 'treatment', 'trait_anxiety']) as attribute,
       unnest(coef) as coefficient,
       unnest(std_err) as standard_error,
       unnest(z_stats) as z_stat,
       unnest(p_values) as pvalue,
       unnest(odds_ratios) as odds_ratio
FROM patients_logregr;

-- ************ --
--    Result    --
-- ************ --
+--------------+--------------+----------------+-----------+-----------+--------------+
| attribute    |   coefficient |   standard_error |    z_stat |    pvalue |   odds_ratio |
|--------------+--------------+----------------+-----------+-----------+--------------|
| intercept    |     -6.36347 |         3.2139 | -1.97999  | 0.0477052 |   0.00172338 |
| treatment    |     -1.02411 |         1.17108 | -0.874498 | 0.381847  |   0.359117   |
| trait_anxiety |      0.119045 |         0.054979 |  2.16528  | 0.0303664 |   1.12642    |
+--------------+--------------+----------------+-----------+-----------+--------------+
```

Model Exploration

Picture taken from [34]

In-Database ML

- MADLib: Interface

```
-- Display prediction value along with the original value
SELECT p.id, madlib.logregr_predict(coef, ARRAY[1, treatment, trait_anxiety]),
       p.second_attack
FROM patients p, patients_logregr m
ORDER BY p.id;

-- ************ --
--    Result    --
-- ************ --
+------+------------------+-----------------+
|   id | logregr_predict  |   second_attack |
|------+------------------+-----------------|
|    1 | True             |               1 |
|    2 | True             |               1 |
|    3 | False            |               1 |
|    4 | True             |               1 |
|    5 | False            |               1 |
|    6 | True             |               1 |
|    7 | True             |               1 |
|    8 | True             |               1 |
|    9 | True             |               1 |
|   10 | True             |               1 |
|   11 | True             |               0 |
|   12 | False            |               0 |
|   13 | False            |               0 |
|   14 | False            |               0 |
|   15 | False            |               0 |
|   16 | False            |               0 |
|   17 | True             |               0 |
|   18 | False            |               0 |
|   19 | False            |               0 |
|   20 | True             |               0 |
+------+------------------+-----------------+
```

Model-based Inference

Picture taken from [34]

In-Database ML

Some alternatives:

- Backend choice
  - On top of DBMSs: MADLib, RIOT-DB
- Inside of DBMSs: SimSQL
- Alternative to DBMSs, but with some similarities: SystemML, RIOT, SciDB
- Interface choice
  - SQL or extension: MADLib
  - ML-Oriented Language on top of SQL: Oracle R Enterprise, RIOT-DB

In-Database ML

Evaluation

- **Table 1: Tests/settings for Axis 1 (task complexity). Upper case symbols in bold font (e.g., X) represent matrices; lower case symbols in bold font (e.g., w) represent vectors. $\mathbf{X}_{i,j}$ is cell (i,j) of X. ϵ is the residual vector from OLS.**

Test Description	Test Semantics
MAT: Matrix Operators ∼ **Vary**: Nodes, Cores, Rows, Sparsity	
Category (1): Matrix Transpose (TRANS)	$\mathbf{X}^T; \mathbf{X}^T_{i,j} = \mathbf{X}_{j,i}$
Category (2): Frobenius Norm (NORM)	$\|\mathbf{X}\|_F = \sqrt{\sum_i \sum_j \|\mathbf{X}^2_{i,j}\|}$
Category (3): Gram Matrix (GRM)	$\mathbf{X}^T\mathbf{X}; (\mathbf{X}^T\mathbf{X})_{i,j} = \sum_k \mathbf{X}_{k,i} \cdot \mathbf{X}_{k,j}$
Category (4): Matrix-Vector Multiplication (MVM)	$\mathbf{Xw}; (\mathbf{Xw})_i = \sum_k \mathbf{X}_{ik} \cdot \mathbf{w}_k$
Category (5): Matrix Addition (ADD)	$\mathbf{M} + \mathbf{N}; (\mathbf{M} + \mathbf{N})_{i,j} = \mathbf{M}_{i,j} + \mathbf{N}_{i,j}$
Category (6): Matrix Multiplication (GMM)	$\mathbf{MN}; (\mathbf{MN})_{i,j} = \sum_k \mathbf{M}_{i,k} \cdot \mathbf{N}_{k,j}$
PIPE: Pipelines and Decompositions ∼ **Vary**: Rows	
Multiplication Chain (MMC)	$\mathbf{X_1X_2X_3}$
Singular Value Decomposition (SVD)	$\mathbf{U\Sigma V}^T \leftarrow \mathbf{X}$
ALG: Bulk LA-based ML Algorithms ∼ **Vary**: Nodes, Cores, Rows, Sparsity	
Ordinary Least Squares Regression (OLS)	$(\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{y})$
Logistic Regression (LR)	See Algorithm 2
Non-negative Matrix Factorization (NMF)	See Algorithm 3
Heteroscedasticity-Robust Standard Errors (HRSE)	$(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\text{diag}(\boldsymbol{\epsilon}^2)\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}$

Picture taken from [TK18].

In-Database ML

Evaluation

- MADLib or SciDB do not outperform specialized systems.
- In-DB performs better for sparse data

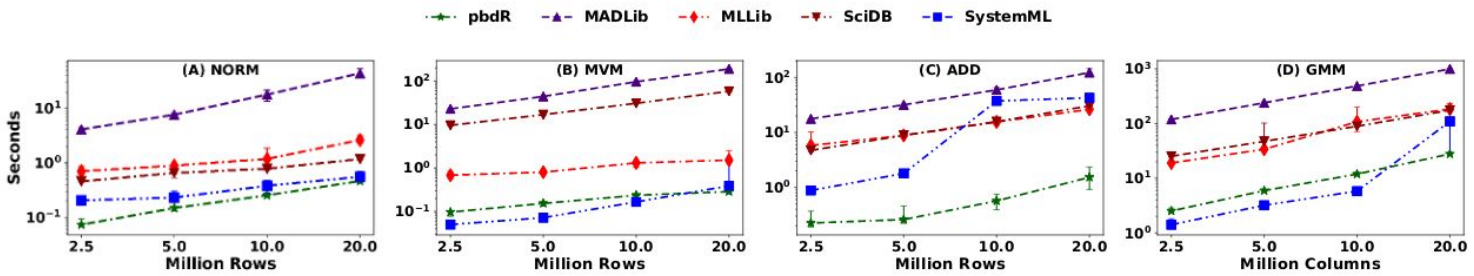


Figure 1: Multi-Node Dense Data for MAT Varying Rows

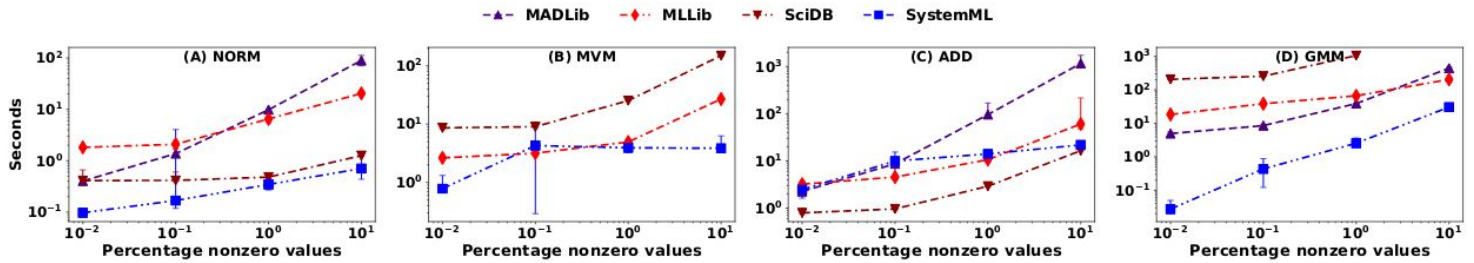


Figure 2: Multi-Node Sparse Data for MAT with Varying Sparsity

Picture taken from [TK18].

In-Database ML

Evaluation

- MADLib or SciDB do not outperform specialized systems.
- In-DB performs better for sparse data

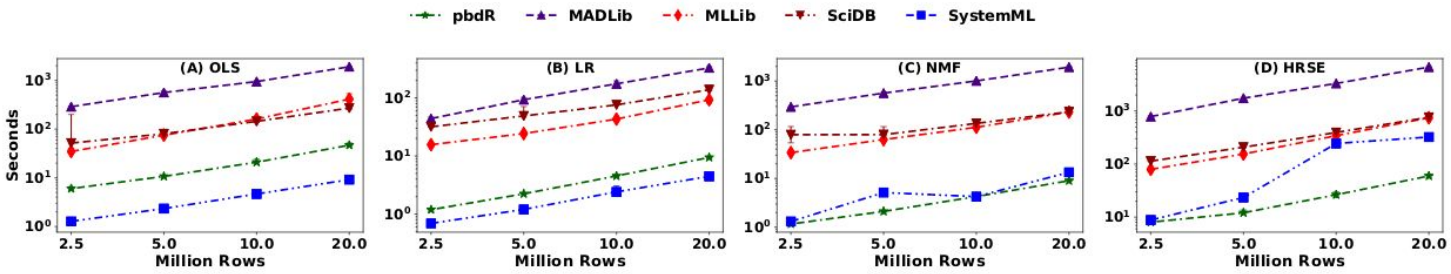


Figure 5: Multi-Node Dense Data for ALG with varying Rows

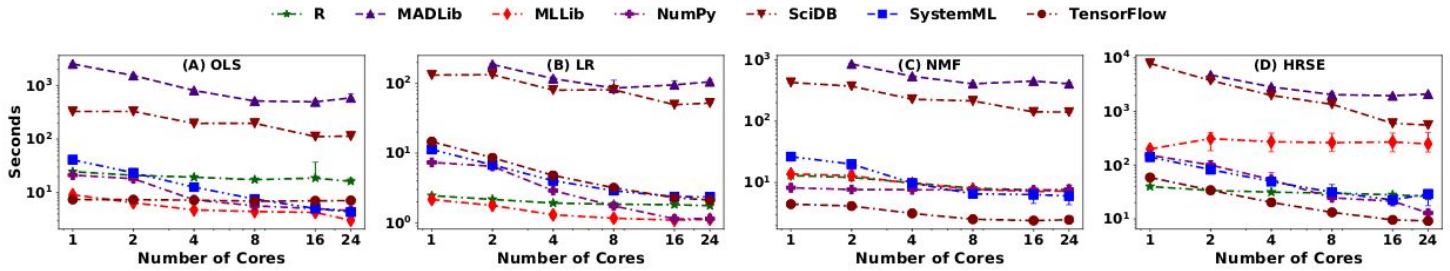


Figure 6: Single-Node Dense Data for ALG with varying Cores

Picture taken from [TK18].

In-Database ML

Evaluation

- In terms of ease of configuration SystemML is currently at the top of the pack.
- MLLib remains unstable, hard to configure and to work with
  - Many matrix types and operations not supported by them
  - User effort is required
- Intermediate results are the core bottleneck in systems like MADLib
  - By using a disk-based system, results are written to disk before being read again
- SystemML deals well with automating pipeline optimizations (ordering of matrix operations).

Summary

- In-database ML seeks to reduce the data movement by doing ML inside the system
- It seeks to support:
  - Model creation
  - Model exploration
  - Model-based inference
- In-DB ML relies on operators and data configurations to support linear-algebra operations efficiently
- Though practical, the performance of these solutions still lags behind those of specialized ML systems

2. Implementing Database Internals with AI

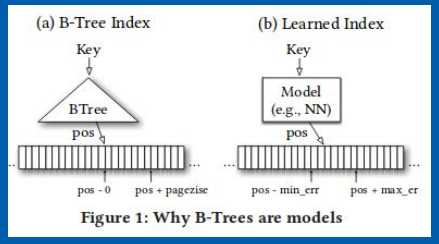
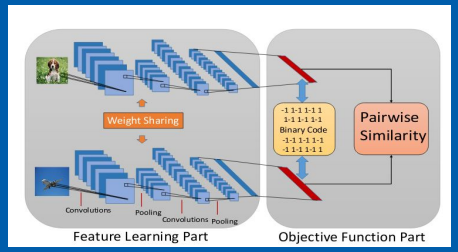


Figure 1: Why B-Trees are models



Storage Engine Components can be implemented with ML too!

- We consider two examples:
  - Learned Index Structures
  - Deep hashing

2. ML Techniques for Implementing Database Internals

Learned Index Structures [KBC18]

- Indexes are essential for efficient data access
- Range queries(B-Trees), key lookups (Hash Maps), existence queries (Bloom Filters) rely on tuned index structures
- **These indexes are general purpose and don't fit for the actual distribution of the data.**

Learned Index Structures

- The core idea of the paper:
  - All index structures can be replaced with other type of models, the authors call these LEARNED INDEXES.
  - They can be neural nets, or linear regressors, etc.
    - Reasoning on these models is a tradeoff between their accuracy at the task, their memory footprint and inference runtime
  - These models reconstruct query results with low cost, acting as a form of compression
  - Since regressors can be inaccurate, we search for positions + max_error

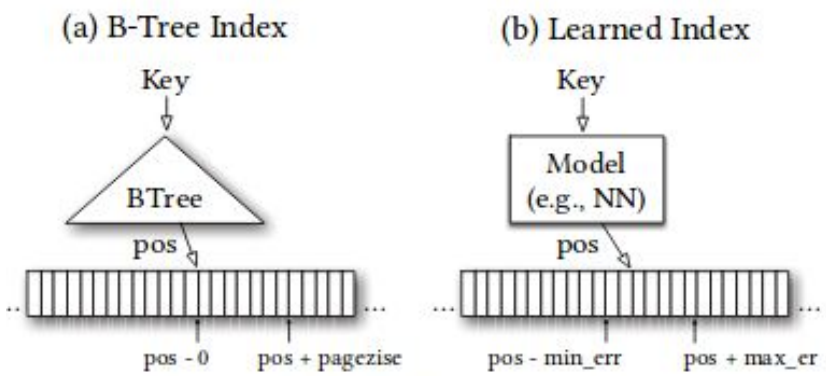


Figure 1: Why B-Trees are models

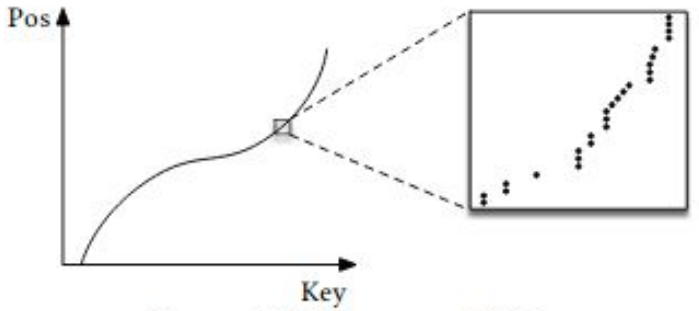
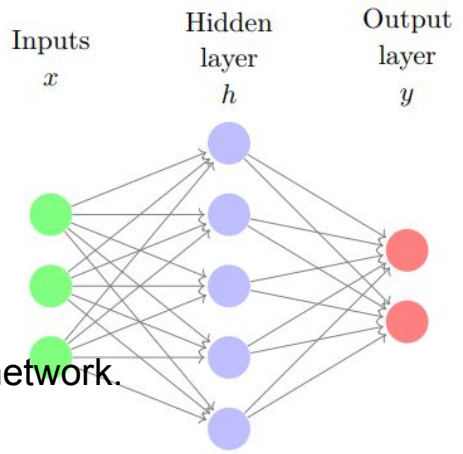


Figure 2: Indexes as CDFs

Pictures taken from [KBC18]

Learned Index Structures

- Neural networks are a simple model that has a good computational performance at inference.
  - Limited memory (unlike KNN, for example)
  - High parallelism
  - Already tools available for accelerating their serving and their training



Simple schema of a neural network.

Picture taken from [LHI18]

Learned Index Structures

- Two general expectations when comparing indexes at the key search task:

Index Search	NN Inference
- Accesses complete data, with cache misses	+Reduced data used, less or no cache misses
- Mostly sequential search	+More parallelism available per search (since matrix multiplications are embarrassingly parallel)

Learned Index Structures

- Recursive Model Index (RMI)
- RMI: **hierarchy of models**, where at each stage the model takes the key as an input and based on it picks another model, until the final stage predicts the position. End result will be a page that can answer the lookup query.
  Benefits:
  - Easy to learn the overall data distribution, divides spaces into smaller sub ranges (like B-Trees), no search process required between stages.
  - Hybridity.
- Inner nodes are classifers
- Leaves are linear regressors

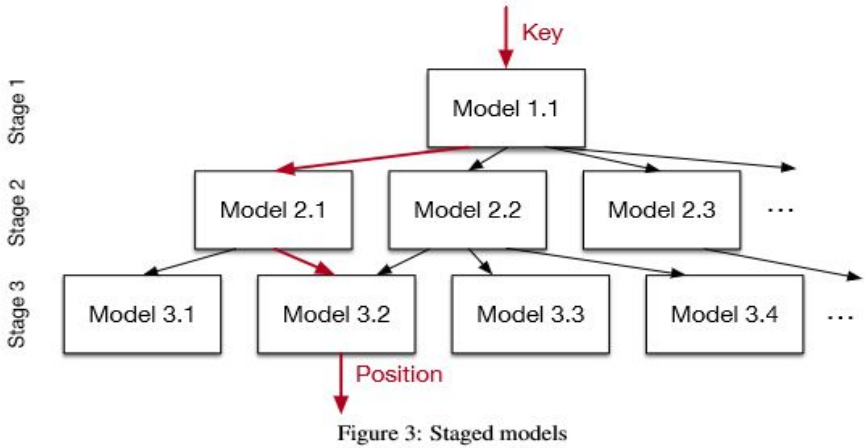


Figure 3: Staged models

Picture taken from [KBC18]

Learned Index Structures

- Hybrid End to End Training

  This model starts of by training the top nodes first

  It then predicts the next models in line 9 and 10 and stores all the keys that fall into that particular model (for training it later).

  Also in case of hybrid index they calculate the error in line 12. If the error is higher than the threshold, the model is substituted by a B-tree.

  Hybrid indexes bound the worst case performance of learned indexes to the performance of B-Trees

Algorithm 1: Hybrid End-To-End Training

```
   Input: int threshold, int stages[], NN_complexity
   Data: record data[], Model index[][]
   Result: trained index
1  M = stages.size;
2  tmp_records[][];
3  tmp_records[1][1] = all_data;
4  for i ← 1 to M do
5  |  for j ← 1 to stages[i] do
6  |  |  index[i][j] = new NN trained on tmp_records[i][j];
7  |  |  if i < M then
8  |  |  |  for r ∈ tmp_records[i][j] do
9  |  |  |  |  p = f(r.key) / stages[i + 1];
10 |  |  |  |  tmp_records[i + 1][p].add(r);
11 for j ← 1 to index[M].size do
12 |  index[M][j].calc_err(tmp_records[M][j]);
13 |  if index[M][j].max_abs_err > threshold then
14 |  |  index[M][j] = new B-Tree trained on tmp_records[M][j];
15 return index;
```

Picture taken from [KBC18]

Learned Index Structures

- Results comparing with B-trees on Integer data

		Map Data			Web Data			Log-Normal Data		
Type	**Config**	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)
Btree	page size: 32	52.45 (4.00x)	274 (0.97x)	198 (72.3%)	51.93 (4.00x)	276 (0.94x)	201 (72.7%)	49.83 (4.00x)	274 (0.96x)	198 (72.1%)
	page size: 64	26.23 (2.00x)	277 (0.96x)	172 (62.0%)	25.97 (2.00x)	274 (0.95x)	171 (62.4%)	24.92 (2.00x)	274 (0.96x)	169 (61.7%)
	page size: 128	13.11 (1.00x)	265 (1.00x)	134 (50.8%)	12.98 (1.00x)	260 (1.00x)	132 (50.8%)	12.46 (1.00x)	263 (1.00x)	131 (50.0%)
	page size: 256	6.56 (0.50x)	267 (0.99x)	114 (42.7%)	6.49 (0.50x)	266 (0.98x)	114 (42.9%)	6.23 (0.50x)	271 (0.97x)	117 (43.2%)
	page size: 512	3.28 (0.25x)	286 (0.93x)	101 (35.3%)	3.25 (0.25x)	291 (0.89x)	100 (34.3%)	3.11 (0.25x)	293 (0.90x)	101 (34.5%)
Learned Index	2nd stage models: 10k	0.15 (0.01x)	98 (2.70x)	31 (31.6%)	0.15 (0.01x)	222 (1.17x)	29 (13.1%)	0.15 (0.01x)	178 (1.47x)	26 (14.6%)
	2nd stage models: 50k	0.76 (0.06x)	85 (3.11x)	39 (45.9%)	0.76 (0.06x)	162 (1.60x)	36 (22.2%)	0.76 (0.06x)	162 (1.62x)	35 (21.6%)
	2nd stage models: 100k	1.53 (0.12x)	82 (3.21x)	41 (50.2%)	1.53 (0.12x)	144 (1.81x)	39 (26.9%)	1.53 (0.12x)	152 (1.73x)	36 (23.7%)
	2nd stage models: 200k	3.05 (0.23x)	86 (3.08x)	50 (58.1%)	3.05 (0.24x)	126 (2.07x)	41 (32.5%)	3.05 (0.24x)	146 (1.79x)	40 (27.6%)

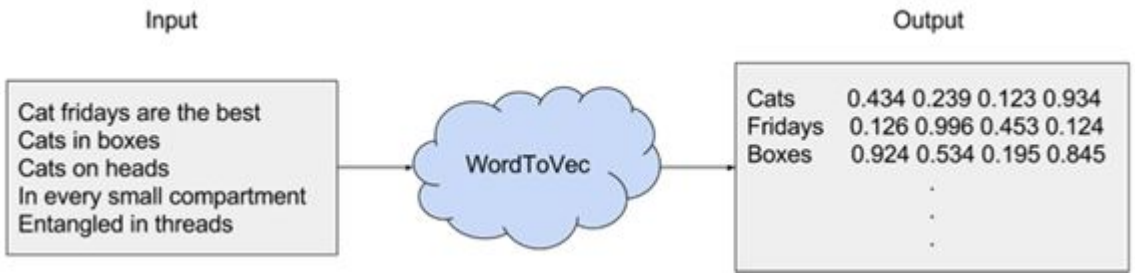
Picture taken from [KBC18]

Learned index structures perform consistently better than the baseline (B-tree with page size=128). These results do not consider out-of-the-box opportunities for co-processor acceleration.

There is follow-up work for improving them (e.g. the updates and the search process), for extending their designs to multi-dimensional cases, in using the learned approach to accelerate operations like sorting, among others.

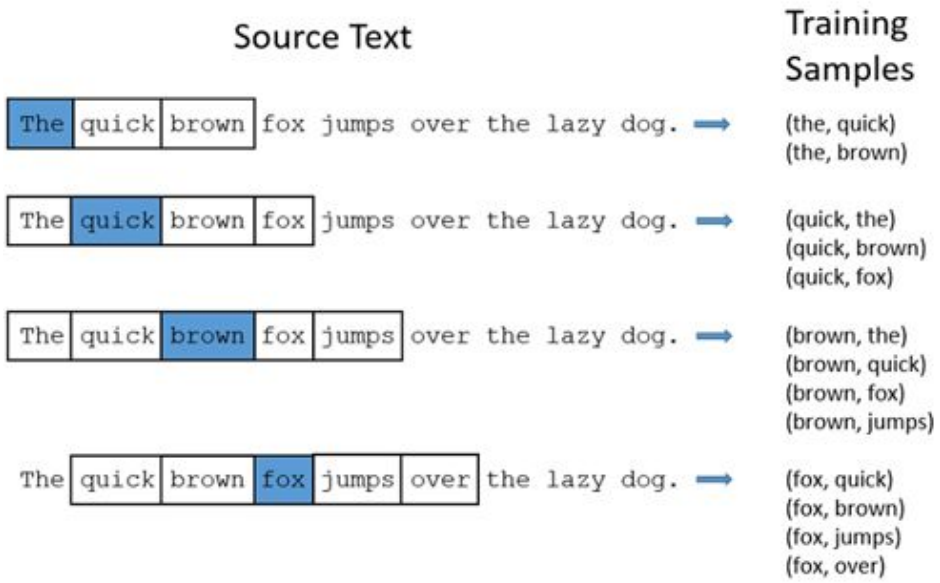
Deep Hashing: Motivation

- Many real world problems involve the search of nearest neighbours over data that is non euclidian:
  - Similar tweets
  - Similar users in a network
- One approach that has proven to be useful involves a process called *embedding.*
  - *Embedding* consists of learning an Euclidean (vector) representation for data that is non-euclidian in their input.



Picture taken from [35]

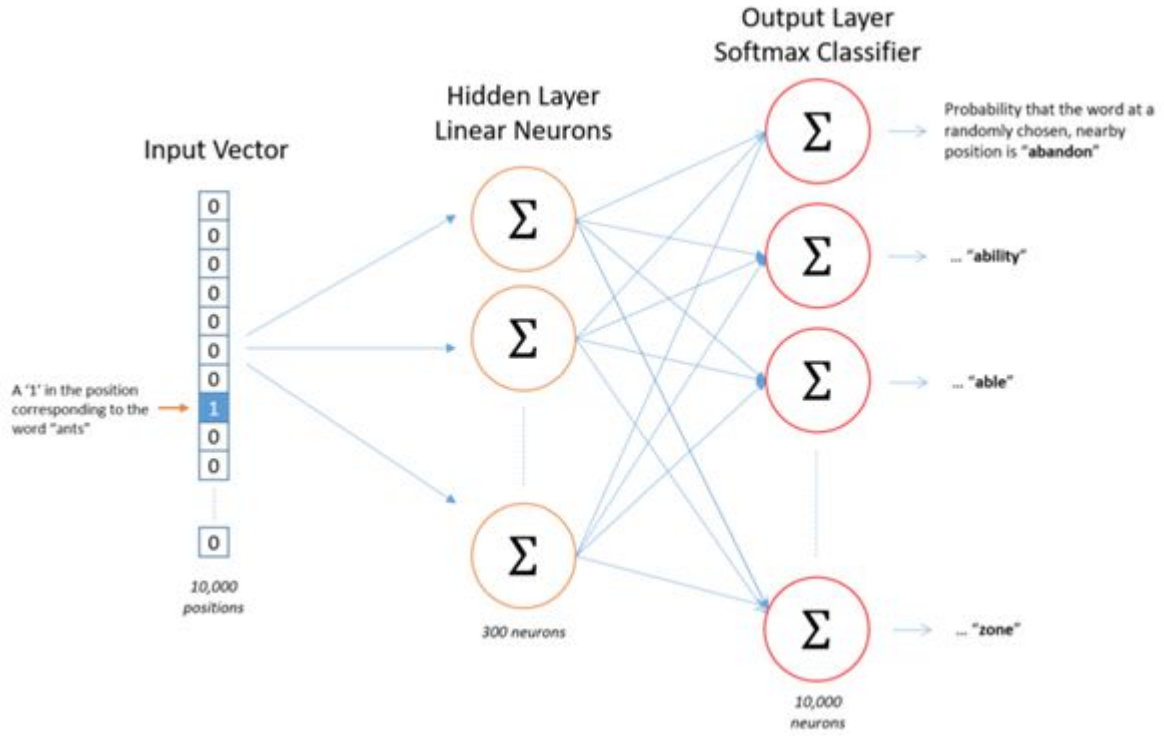
Deep Hashing: Motivation



Word2Vec embedding: Generating training examples

Picture taken from [35]

Deep Hashing: Motivation

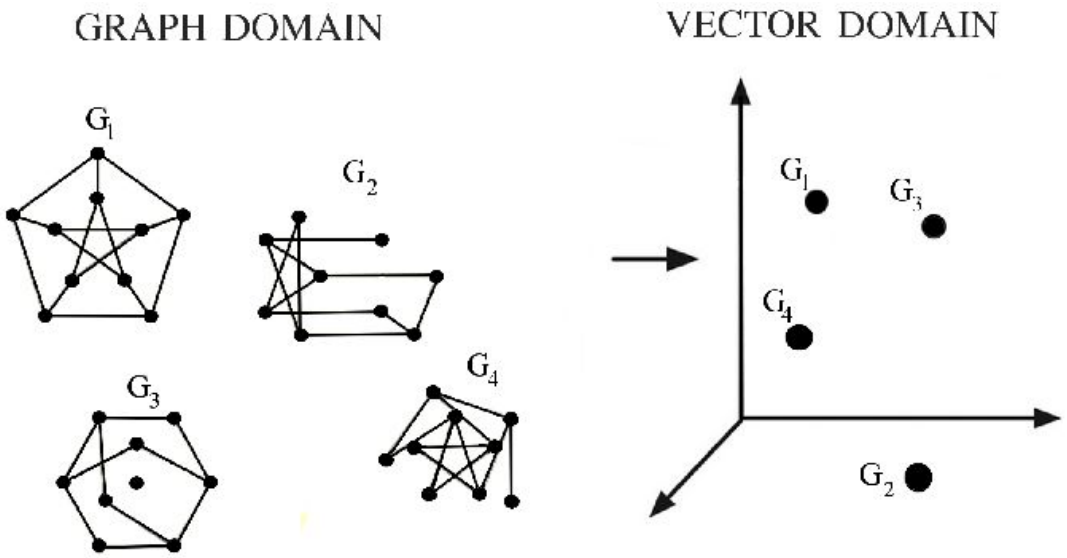


Training is done on the surrogate task of learning the proximity distribution probability of a word

After training, the hidden layer contains the embeddings

Picture taken from [35]

Deep Hashing: Motivation



This approach is also applicable in graphs
Tools like PyTorch Graph help in this

Deep Hashing

- The management of such data represents a problem (tools like Facebook's FAISS, consider some solutions)
- Recently learned hashing has been proposed
  - The idea:
    - Given:
      - an instance
      - pairwise similarity labels
    - Learn a binary code that is similar for similar instances and disimilar for disimilar instances.
- Supervised and unsupervised approaches exist. More information in [36]
- We consider today, as an example, a deep learning supervised approach.

Deep Hashing

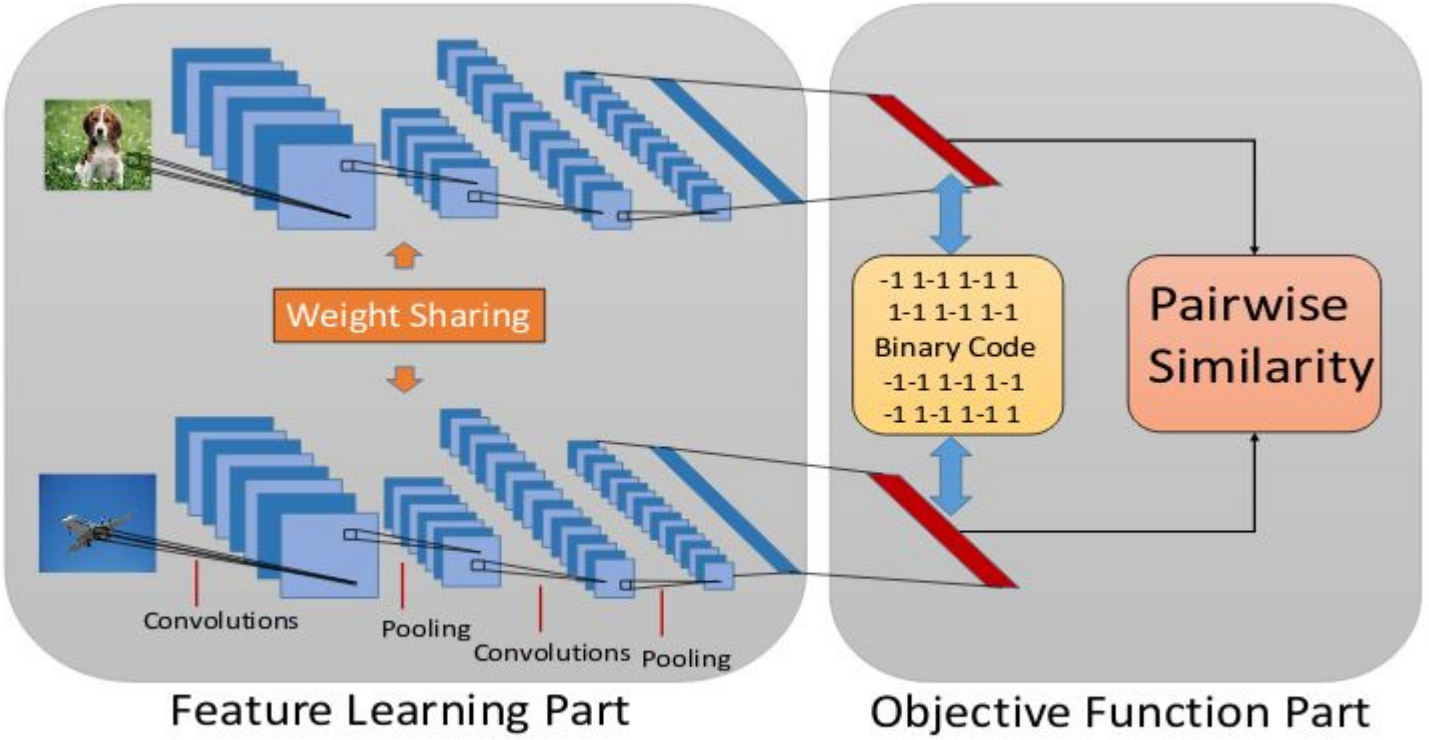


Figure 1: The end-to-end deep learning architecture for DPSH.

Source: [LWK15]

Deep Hashing

- For training, a batch of pairs, with their similarities are selected.
- No activation function is given to the output of the neural network (though clipping is good)
- The network is trained to minimize the follwing loss, by moving the weights to map input (raw data) to codes that minimize the loss.
- Since the values are in the range [-1,1], the dot product can only go from - length of code, to + length of code.
- This loss naturally maximizes the distance in the codes that should be disimilar, and maximizes that in codes that should be similar.

$$\min_{\mathcal{B},\mathcal{U}} \mathcal{J}_3 = -\sum_{s_{ij}\in\mathcal{S}} (s_{ij}\Theta_{ij} - \log(1+e^{\Theta_{ij}}))$$

Source: [LWK15]

Deep Hashing

- An emergent property during the training process is that the most similar codes are to be found at no-bit distance, or 1-2 bit distance...

Deep Hashing: Evaluation

Table 1: Mean Average Precision (MAP) of Hamming Ranking for Different Number of Bits on Three Image Datasets

Method	NUS-WIDE (MAP)				CIFAR-10 (MAP)				Flickr (MAP)			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
DHN	**0.708**	**0.735**	**0.748**	**0.758**	**0.555**	**0.594**	**0.603**	**0.621**	**0.810**	**0.828**	**0.829**	**0.841**
DNNH	0.674	0.697	0.713	0.715	0.552	0.566	0.558	0.581	0.783	0.789	0.791	0.802
CNNH⋆	0.617	0.663	0.657	0.688	0.484	0.476	0.472	0.489	0.749	0.761	0.768	0.776
CNNH	0.611	0.618	0.625	0.608	0.429	0.511	0.509	0.522	0.732	0.734	0.741	0.740
KSH	0.556	0.572	0.581	0.588	0.303	0.337	0.346	0.356	0.690	0.702	0.702	0.706
ITQ-CCA	0.435	0.435	0.435	0.435	0.264	0.282	0.288	0.295	0.513	0.531	0.540	0.555
MLH	0.500	0.514	0.520	0.522	0.182	0.195	0.207	0.211	0.610	0.618	0.629	0.634
BRE	0.485	0.525	0.530	0.544	0.159	0.181	0.193	0.196	0.571	0.592	0.599	0.604
SH	0.433	0.426	0.426	0.423	0.131	0.135	0.133	0.130	0.531	0.533	0.531	0.529
ITQ	0.452	0.468	0.472	0.477	0.162	0.169	0.172	0.175	0.544	0.555	0.560	0.570
LSH	0.403	0.421	0.426	0.441	0.121	0.126	0.120	0.120	0.499	0.513	0.521	0.548

Considering top-k searches for similarity, DHNs perform well, in their precision.

Source: [LWK15]

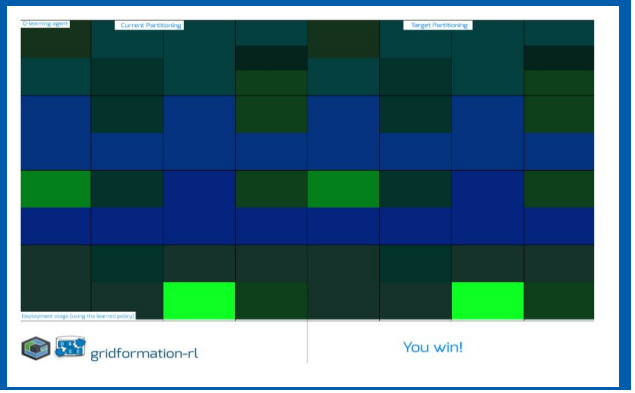
Summary

- Database storage structures can be improved with machine learning models.
  - Learned index structures are, to date, an interesting idea for accelerating OLAP queries, on data without many changes.
  - LIS work on 2 simplifications: CDFs and the Recursive Index Model.
  - Learned hashing can help high-dimensional nearest neighbor search, it relies on labeled data
  - Though it is early work, in both cases, results are promising

End of First Part of Lecture

Thanks for the attention!

Questions?



Web Resources

- [1] http://commons.wikimedia.org/wiki/File:RAM_module_SDRAM_1GiB.jpg
- [2] http://commons.wikimedia.org/wiki/File:Hard_disks.jpg
- [3] http://www.flickr.com/photos/25757823@N07/2719552544
- [4] http://commons.wikimedia.org/wiki/File:Super_Talent_2.5in_SATA_SSD_SAM64GM25S.jpg
- [5] http://commons.wikimedia.org/wiki/File:Gtx260.jpg
- [6] https://paperswithcode.com/sota
- [7] https://www.groundai.com/
- [8] https://spinningup.openai.com/en/latest/
- [9] https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/
- [10] https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
- [11] http://yann.lecun.com/exdb/lenet/
- [12] https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/
- [13] https://devblogs.nvidia.com/nvidia-ibm-cloud-support-imagenet-large-scale-visual-recognition-challenge/

Web Resources

- [14] http://mattturck.com
- [15] https://towardsdatascience.com/rootstrap-dikw-model-32cef9ae6dfb
- [16] https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/
- [17] https://www.parsehub.com/
- [18] https://www.import.io/
- [19] https://databricks.com/glossary/data-lake
- [20] https://www.tamr.com/
- [21] http://openrefine.org/
- [22] https://www.mturk.com/
- [23] https://www.slideshare.net/databricks/snorkel-dark-data-and-machine-learning-with-christopher-r-76732545
- [24] https://www.oreilly.com/ideas/rethinking-software-engineering-in-the-ai-era

Web Resources

- [26] https://www.oreilly.com/library/view/deep-reinforcement-learning/9780135171233/
- [27] https://twitter.com/pyoudeyer/status/1051792302430130176
- [28] https://developer.ibm.com/articles/cc-models-machine-learning/
- [29] https://medium.com/@ODSC/active-learning-your-models-new-personal-trainer-a89722c0db5a
- [30] https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
- [31] https://www.youtube.com/watch?v=3liCbRZPrZA
- [32] https://twitter.com/tensorflow/status/832008382408126464
- [33] https://www.datacenterdynamics.com/opinions/path-ai-connected-government/
- [34] https://madlib.apache.org/
- [35] https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1
- [36] http://cs.nju.edu.cn/lwj/L2H.html
- [37] http://cidrdb.org/cidr2017/slides/p42-pavlo-cidr17-slides.pdf
- [38] http://dsg.uwaterloo.ca/tcde-smdb/#workshops

References

- [ABB12] Alagiannis, Ioannis, Renata Borovica, Miguel Branco, Stratos Idreos, and Anastasia Ailamaki. "NoDB: efficient query execution on raw data files." In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 241-252. ACM, 2012.
  - NoDB system, for building data structure as queries come, over raw data.
- [CDD09] Cohen, Jeffrey, Brian Dolan, Mark Dunlap, Joseph M. Hellerstein, and Caleb Welton. "MAD skills: new analysis practices for big data." Proceedings of the VLDB Endowment2, no. 2 (2009): 1481-1492.
- [DCA17] Deng, Dong, Raul Castro Fernandez, Ziawasch Abedjan, Sibo Wang, Michael Stonebraker, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, and Nan Tang. "The Data Civilizer System." In Cidr. 2017.
- [Dom15] Domingos, Pedro. The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books, 2015.
  - An easy read presenting the 5 tribes of ML, the core approaches for each case, some historical anecdotes and attempts at integrating the approaches.
- [F18] Frické M.H. (2018) Data-Information-Knowledge-Wisdom (DIKW) Pyramid, Framework, Continuum. In: Schintler L., McNeely C. (eds) Encyclopedia of Big Data. Springer, Cham
  - Encyclopedia entry on the DIKW pyramid

References

- [FFS15] Fakhraei, Shobeir, James Foulds, Madhusudana Shashanka, and Lise Getoor. "Collective spammer detection in evolving multi-relational social networks." In Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining, pp. 1769-1778. ACM, 2015.
    - An example of ML on graphs. Node classification using graph features.
- [GBC16] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.
    - Textbook on deep learning
- [GCK19] Gadepally, Vijay, Justin Goodwin, Jeremy Kepner, Albert Reuther, Hayley Reynolds, Siddharth Samsi, Jonathan Su, and David Martinez. "AI Enabling Technologies: A Survey." arXiv preprint arXiv:1905.03592 (2019).
    - Comprehensive discussion on a canonical AI architecture, presenting components, process and robustness conce
- [HCN16] Halevy, Alon, Flip Korn, Natalya F. Noy, Christopher Olston, Neoklis Polyzotis, Sudip Roy, and Steven Euijong Whang. "Goods: Organizing google's datasets." In Proceedings of the 2016 International Conference on Management of Data, pp. 795-806. ACM, 2016.
- [JYP19] Jouppi, Norman P., Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates et al. "In-datacenter performance analysis of a tensor processing unit." In 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), pp. 1-12. IEEE, 2017.

References

- [KBC18] Kraska, Tim, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. "The case for learned index structures." In Proceedings of the 2018 International Conference on Management of Data, pp. 489-504. ACM, 2018.
- [KBY17] Kumar, Arun, Matthias Boehm, and Jun Yang. "Data management in machine learning: Challenges, techniques, and systems." In Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1717-1722. ACM, 2017.
- [LBB98] LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86, no. 11 (1998): 2278-2324.
    - LeNet-5 paper
- [LHI18] François-Lavet, Vincent, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. "An introduction to deep reinforcement learning." Foundations and Trends® in Machine Learning 11, no. 3-4 (2018): 219-354.
    - Textbook on deep reinforcement learning.
- [Li17] Li, Yuxi. "Deep reinforcement learning: An overview." arXiv preprint arXiv:1701.07274 (2017).

References

- [LLM17] Liang, Eric, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. "Rllib: Abstractions for distributed reinforcement learning." arXiv preprint arXiv:1712.09381(2017).
  - Introduces abstractions for parallel computation, as used in the Ray-RLLib framework
- [LWK15] Li, Wu-Jun, Sheng Wang, and Wang-Cheng Kang. "Feature learning based deep supervised hashing with pairwise labels." arXiv preprint arXiv:1511.03855 (2015).
- [MNN19] Mao, Hongzi, Parimarjan Negi, Akshay Narayan, Hanrui Wang, Jiacheng Yang, Haonan Wang, Ryan Marcus et al. "Park: An Open Platform for Learning Augmented Computer Systems." (2019).
  - Park platform. Insightful presentation of the challenges for applying reinforcement learning to computer systems.
- [MP18] Marcus, Ryan, and Olga Papaemmanouil. "Deep reinforcement learning for join order enumeration." In Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, p. 3. ACM, 2018.
  - ReJoin: First proposal for a DRL-based join order optimizer

References

- [MRT18] Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar. Foundations of machine learning. MIT press, 2018.
  - Comprehensive textbook on machine learning, without deep learning.
- [MT17] Maccioni, Antonio, and Riccardo Torlone. "Crossing the finish line faster when paddling the data lake with kayak." Proceedings of the VLDB Endowment 10, no. 12 (2017): 1853-1856.
  - Kayak system for exploring data lakes
- [NDB19] Nguyen, Giang, Stefan Dlugolinsky, Martin Bobák, Viet Tran, Álvaro López García, Ignacio Heredia, Peter Malík, and Ladislav Hluchý. "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey." Artificial Intelligence Review 52, no. 1 (2019): 77-124.
  - Overview of ML frameworks
- [RBE17] Ratner, Alexander, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. "Snorkel: Rapid training data creation with weak supervision." Proceedings of the VLDB Endowment 11, no. 3 (2017): 269-282.
  - Snorkel system to improve the labeling process.

References

- [RGS18] Roh, Yuji, Geon Heo, and Steven Euijong Whang. "A Survey on Data Collection for Machine Learning: a Big Data-AI Integration Perspective." arXiv preprint arXiv:1811.03402 (2018).
  - A comprehensive review on approaches for data collection for machine learning. Serves as a comprehensive resource for understanding data curation.
- [S15] Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.
  - Survey with a broad historical overview on deep learning, discussing regularization methods and optimization.
- [TK18] Thomas, Anthony, and Arun Kumar. "A comparative evaluation of systems for scalable linear algebra-based analytics." Proceedings of the VLDB Endowment 11, no. 13 (2018): 2168-2182.
- [TTO18]Thirumuruganathan, Saravanan, Nan Tang, and Mourad Ouzzani. "Data Curation with Deep Learning [Vision]: Towards Self Driving Data Curation." arXiv preprint arXiv:1803.01384 (2018).
  - Establishes a comprehensive vision for automated data curation, using deep learning. Authors highlight the usefulness of distributed representations, and summarize recent successes from their group (specially in entity resolution).

References

- [VM18] Vartak, Manasi, and Samuel Madden. "MODELDB: Opportunities and Challenges in Managing Machine Learning Models." IEEE Data Eng. Bull. 41, no. 4 (2018): 16-25.
  - ModelDB presentation and discussion on ML model management systems
- [WCT15] Wang, Wei, Gang Chen, Anh Tien Tuan Dinh, Jinyang Gao, Beng Chin Ooi, Kian-Lee Tan, and Sheng Wang. "SINGA: Putting deep learning in the hands of multimedia users." In Proceedings of the 23rd ACM international conference on Multimedia, pp. 25-34. ACM, 2015.
- [WR17] Wang, Haohan, and Bhiksha Raj. "On the origin of deep learning." arXiv preprint arXiv:1702.07800 (2017).
  - Historical presentation on the development of deep learning, with a focus on tracing precedents for deep belief networks and convolutional neural networks.