PETUUM

Carnegie Mellon University
School of Computer Science

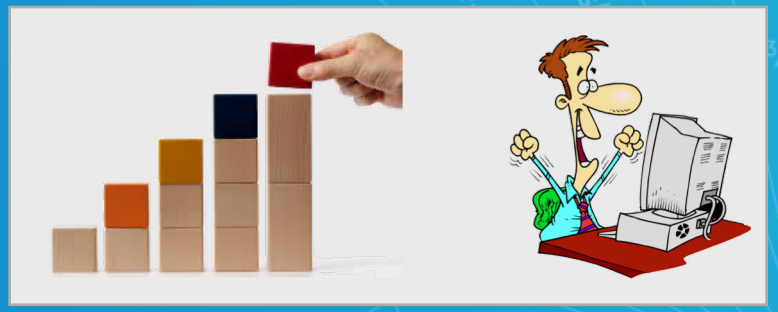
Probabilistic Graphical Models

A Civil Engineering Perspective of AI

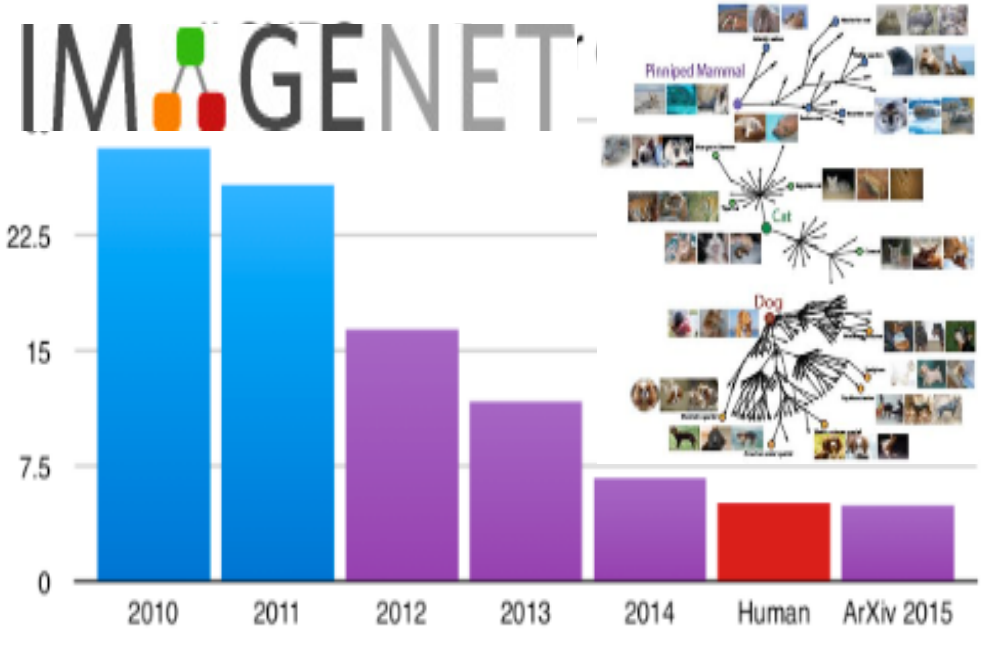
Eric Xing

Lecture 28, April 29, 2019

Reading: see class homepage

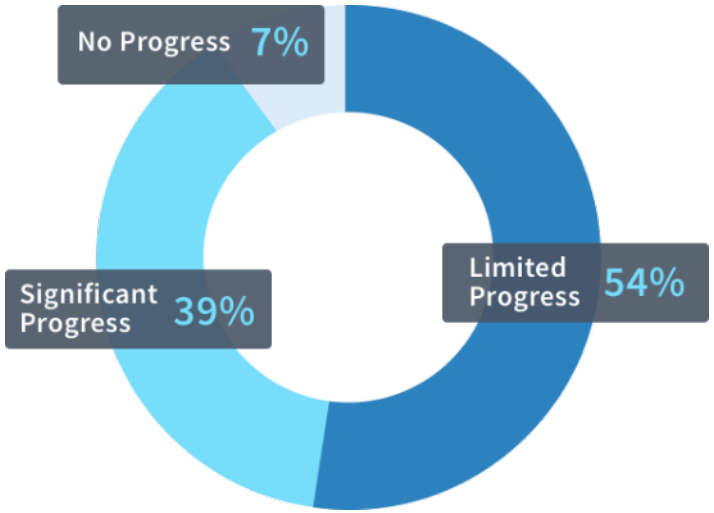


An AI Era?

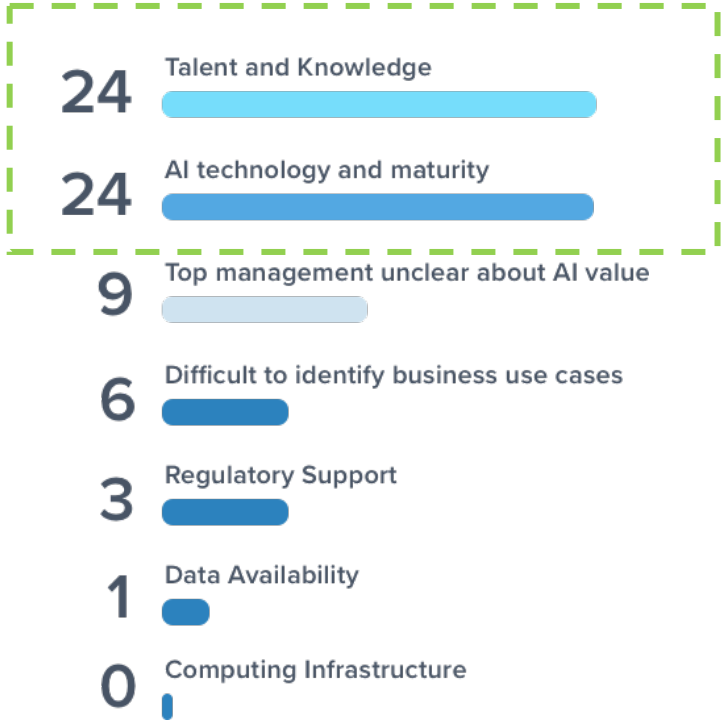


AI Solution Today – major hurdles, and few choices…

How much progress has your company made in the last year implementing AI applications?



What are the top barriers to AI Applications in your company?



Building is infeasible…

- ✗ Talent – Scare resources in data scientists, ML engineers. Sys engineers
- ✗ Support – Little or no enterprise support from open source software
- ✗ Major Infrastructure requirements
- Long development timelines & delivery risks

Lack of viable buy/rent options…

- ✗ Limited to cloud deployment
- ✗ Limited customization
- ✗ Limited service
- ✗ Limited scalability
- Limited capabilities
- … or not available at all!

Source: McKinsey CEO survey

Here might be why …

GIZMODO

VIDEO SPLOID PALEOFUTURE IO9 SCIENCE REVIEW FIELD GUIDE DESIGN

Pretty Much All Tech Demos Are Fake as Hell

Alex Cranz
Yesterday 12:45pm • Filed to: GOOGLE

112.7K 43 4

Screenshot: Andrew Liszewski (Gizmodo)

A real ready-to-use AI solution is extremely complex

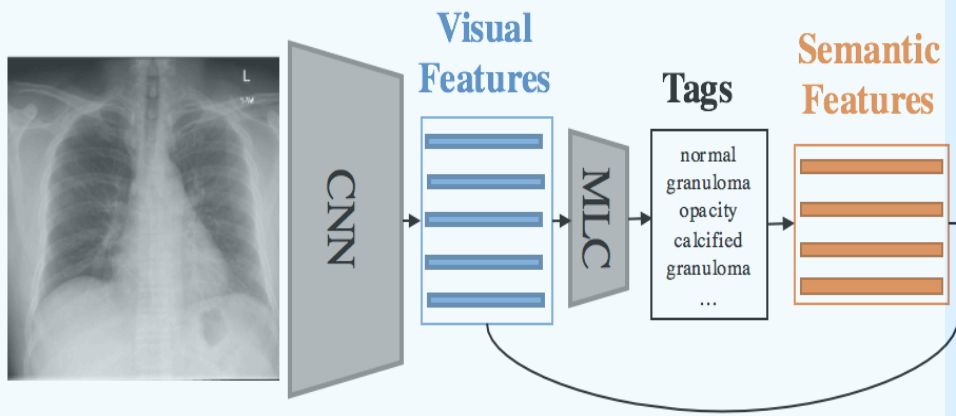
Use Case: Automatic Medical (or other) Report Generation

Findings:
There are no focal areas of consolidation.
No suspicious pulmonary opacities.
Heart size within normal limits.
No pleural effusions.
There is no evidence of pneumothorax.
Degenerative changes of the thoracic spine.

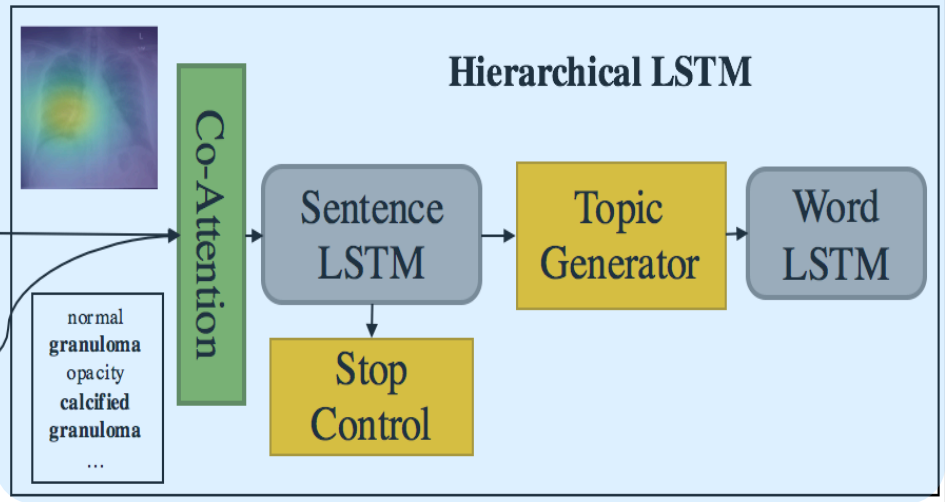
Impression:
No acute cardiopulmonary abnormality.

- Abnormal regions in medical images are difficult to identify.
- How to localize the image regions and tags that are relevant to a sentence?
- How to distribute topics across sentences
- How to make report readable to humans?

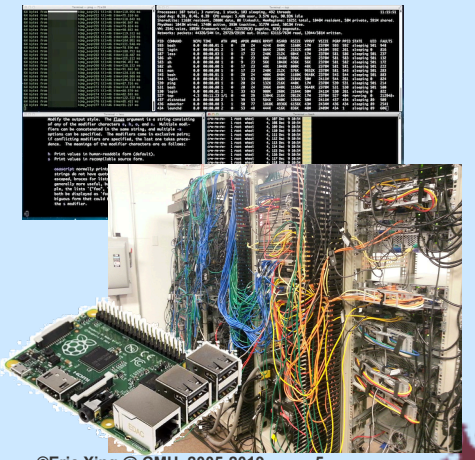
Raw Data Enrichment



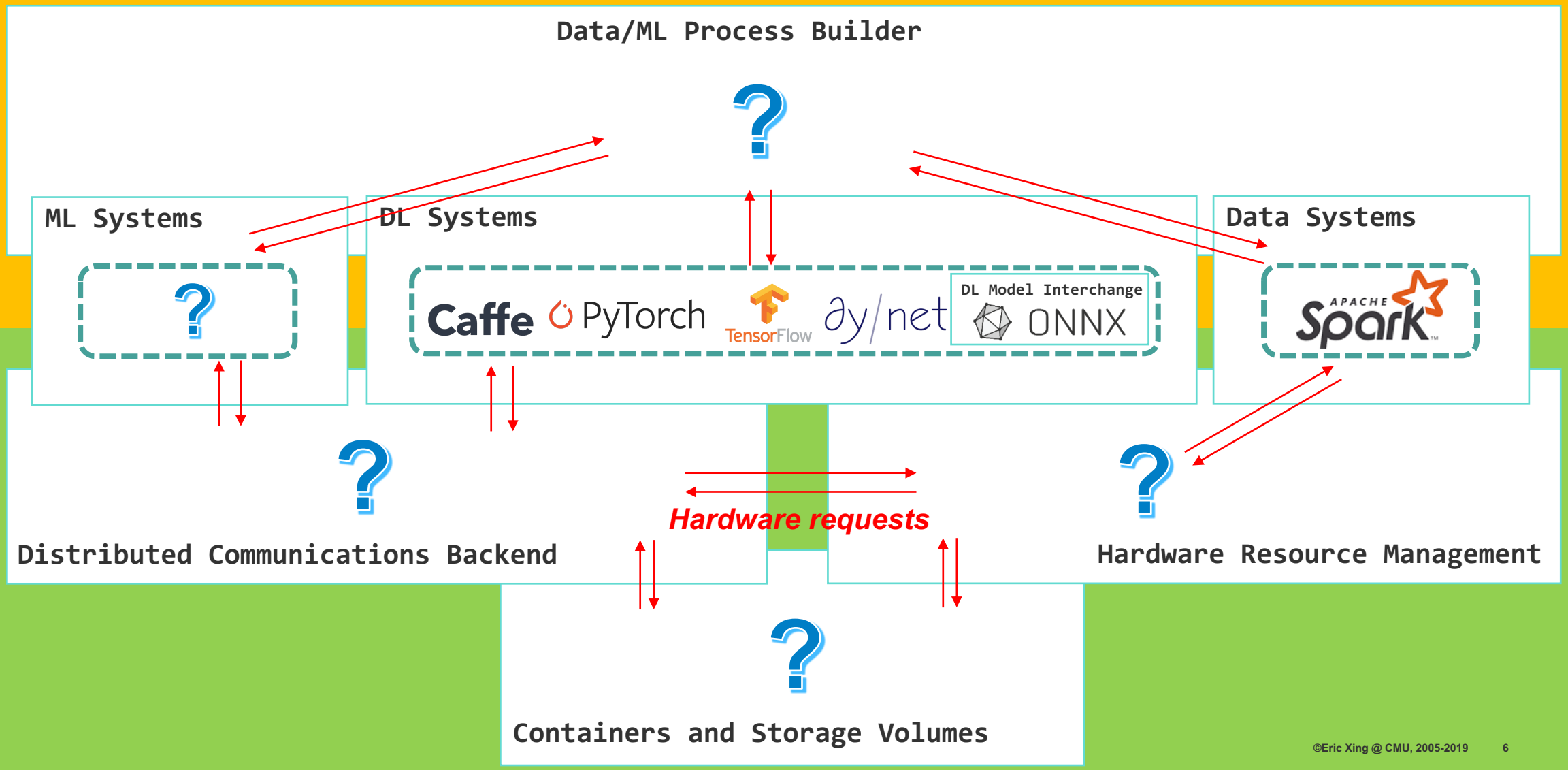
Model/Algorithm



System/Infra



Inter-operability between diverse systems?



Craft versus Build

A builder needs to …

- Build a full, engineering-wise sound solution, not a toy or a demo
- Build many copies
- Build it cheaply
- Build it in any client-specified conditions

Build

- Understand how a solution is built
- Understand what is possible and what is not
- Explain why it works, why not working if undesirable outcome

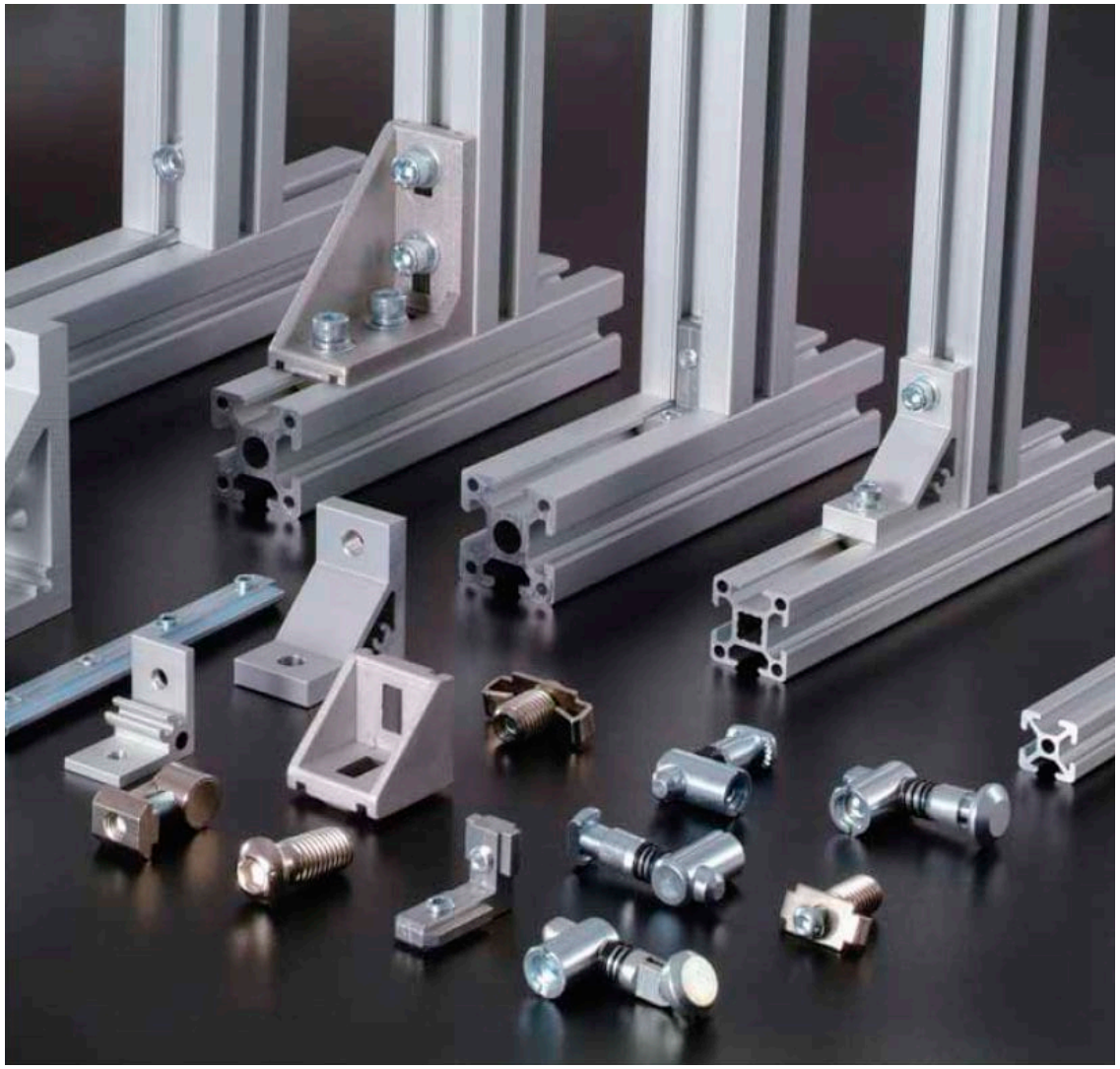
Understand

- Operationalize the solution (deployment, training)
- Repeat the results from the solution
- Teach others to build and allowing others to repeat the results

Sustain

A Need for the “Nuts & Bolts” and “Workbench” of AI – transforming it into “civil engineering”

- Like **Civil Engineering**, via an assembly-line-like building process, the new generation of AI programs should be modular, explainable, debuggable, customizable, and highly accessible
  - **Reusable** pluggable modules allowing enterprises to architect and customize AI rapidly
  - **Standard** instruction sets and pre-built functions ensuring outcomes are repeatable and sound
  - **Comprehensive** inventory of ML/Sys nuts and bolts meeting the needs for any data type, wide range of tasks, commodity hardware, and mission sizes.



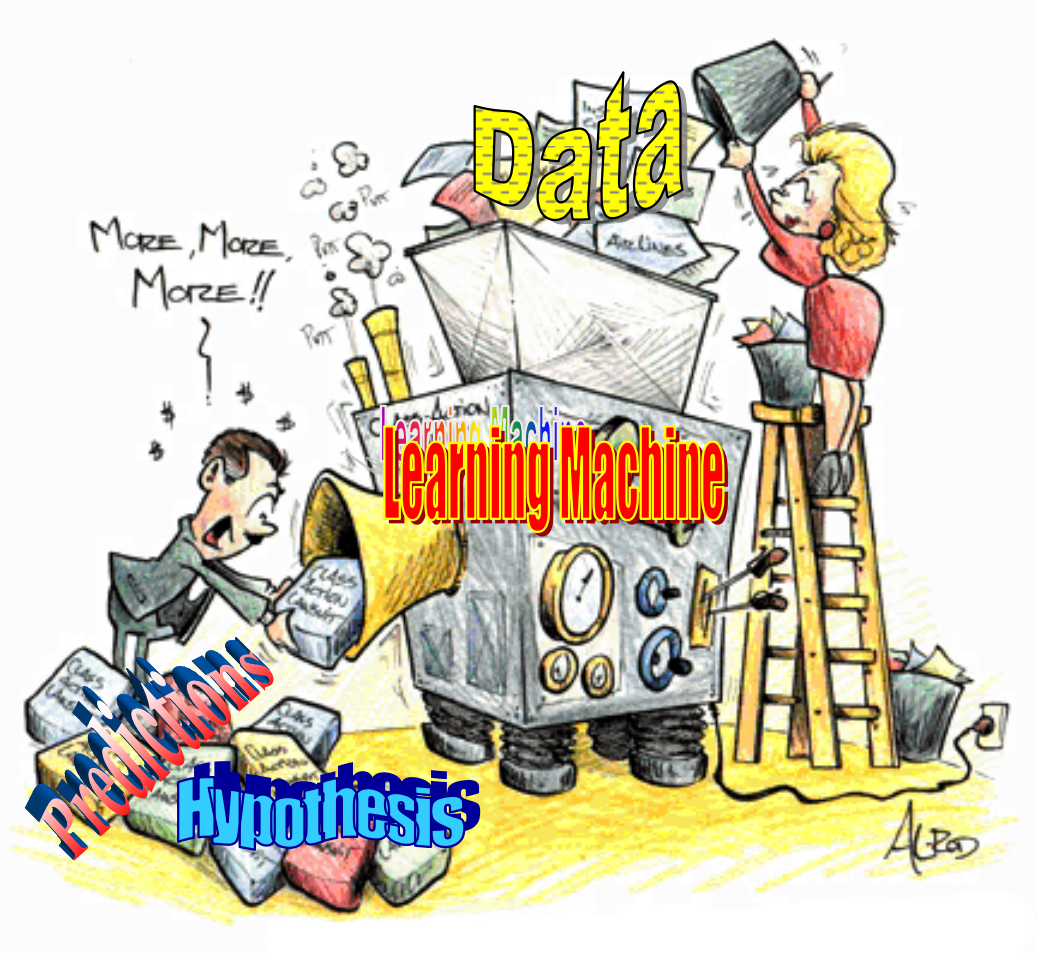
Key issues in enabling such a transformation

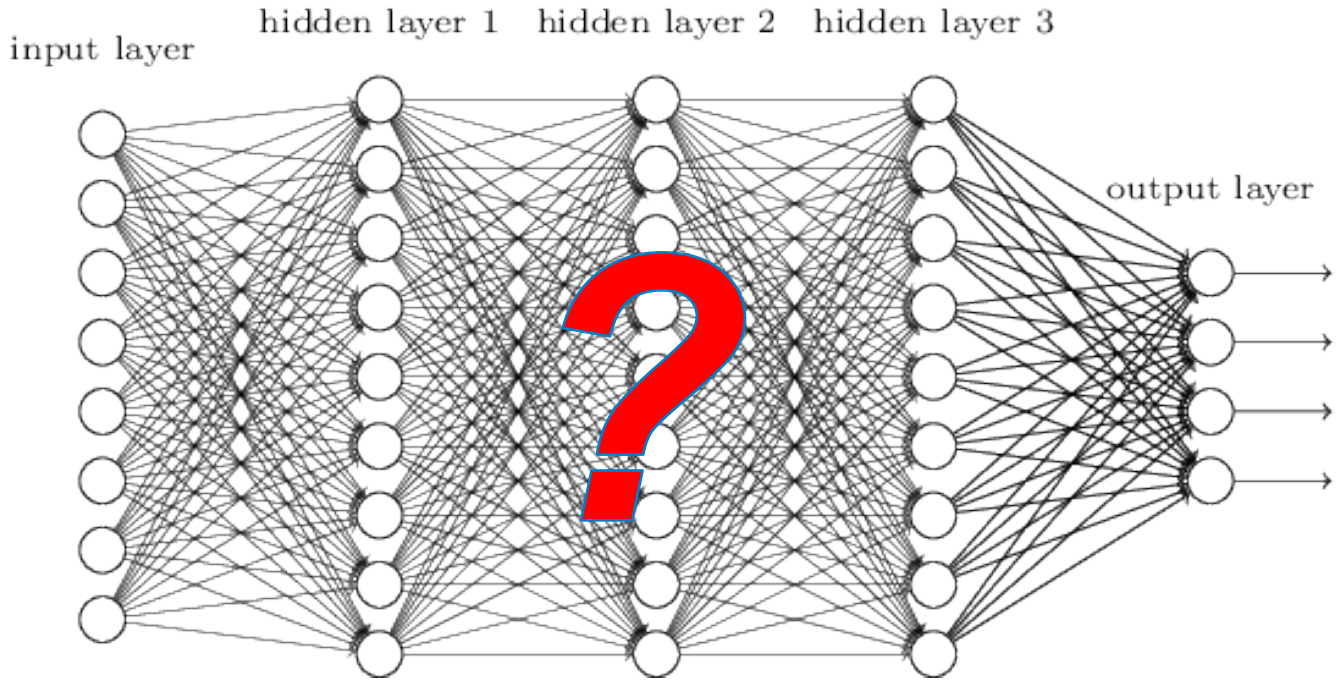
- First Principles
- The “Civil” Engineering
- Explain the process and outcome
- Analysis and safety under real operation
- Standards, mass production, cost amortization

First Principles

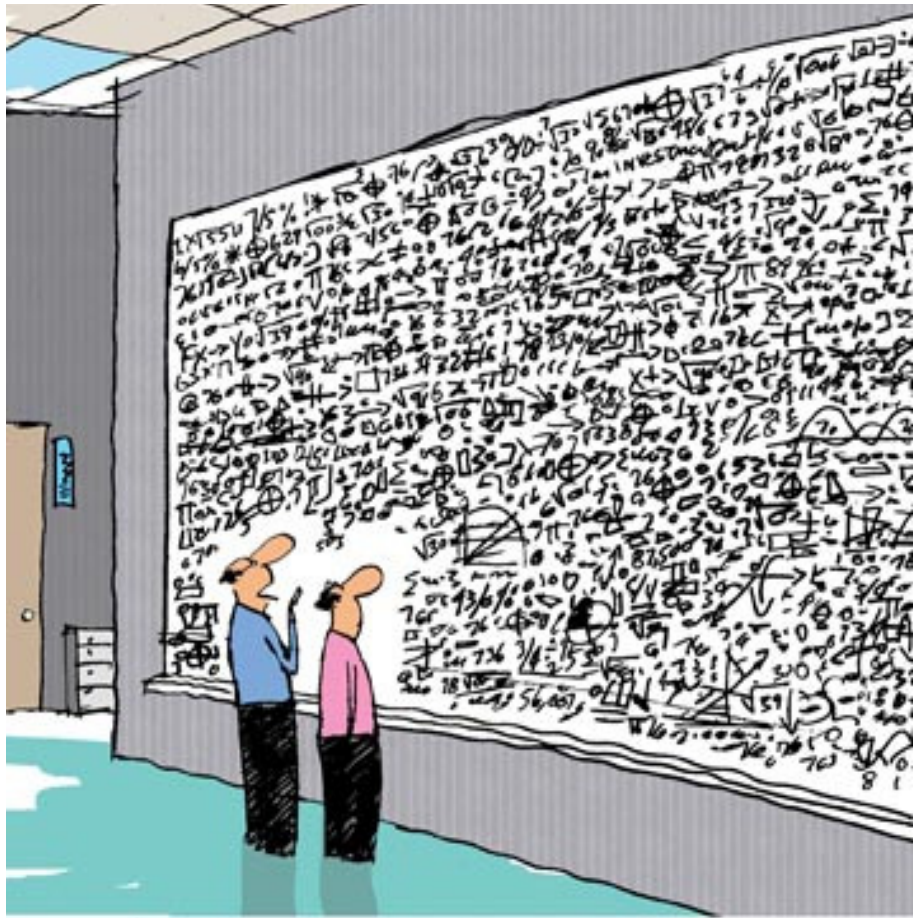
- Models
- Algorithms
- Theories
- Data

Principled ML Process, or ("End-to-End") Blackbox?





First principles: sound, explainable, verifiable



Loss functions
(likelihood, reconstruction, margin, …)

Structures
(Graphical, group, chain, tree, iid, …)

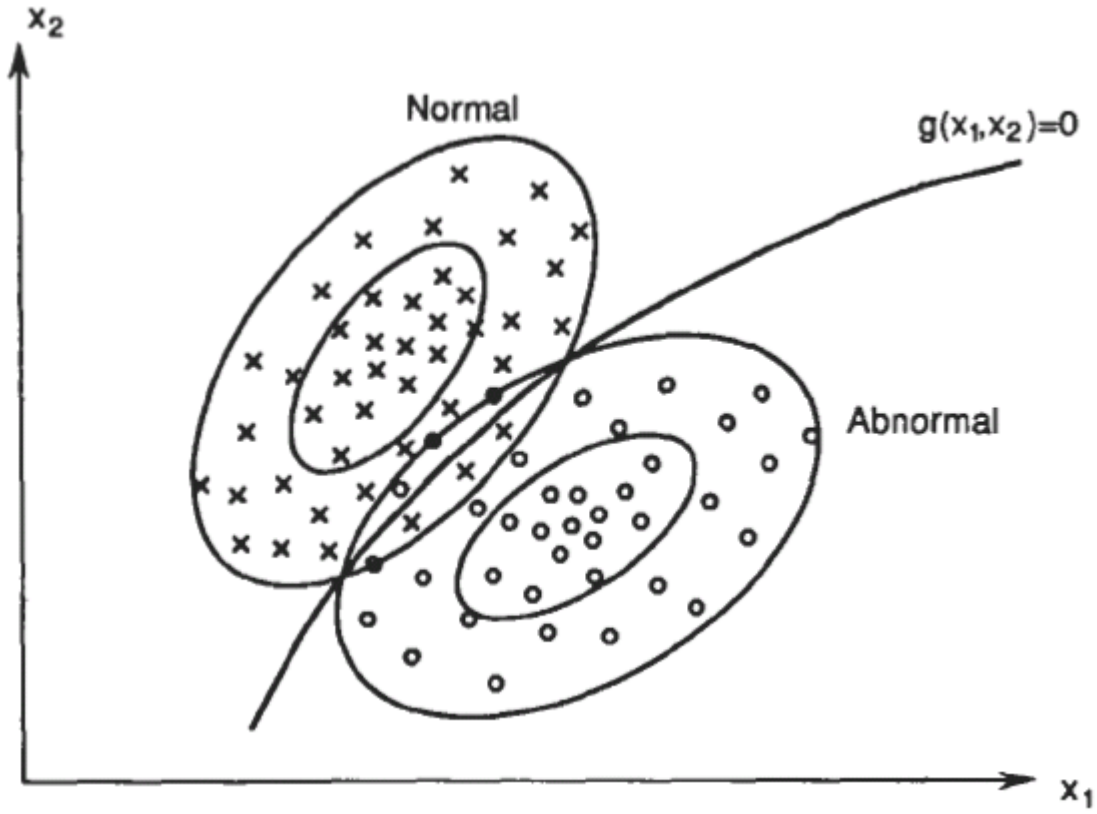
Constraints
(normality, sparsity, label, prior, KL, sum, …)

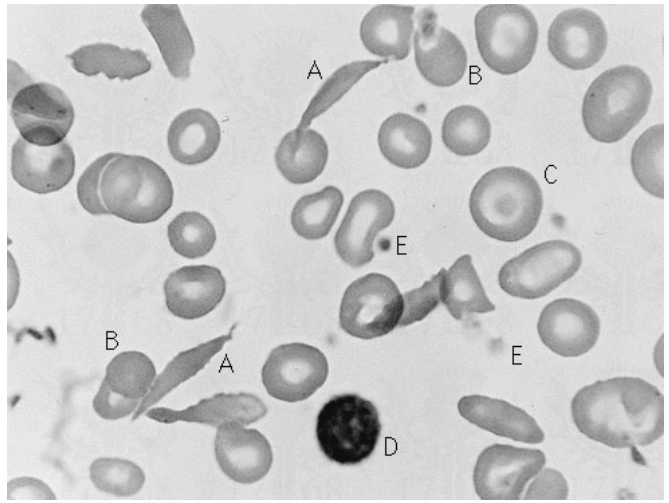
Algorithms
MC (MCMC, Importance), Opt (gradient, IP), …

Stopping criteria
Change in objective, change in update
…

Example: decision-making as high-dimensional inference

- Distributions of samples from normal and abnormal cells





Some ML principles to consider

For the learned F(; θ)

- Consistency (value, pattern, …)
- Bias versus variance
- Sample complexity
- Learning rate
- Convergence
- Error bound
- Confidence
- Stability
- …

PAC Learning Theory

(supervised concept learning)

 # examples (*m*)

representational complexity (*H*)

error rate (ε)

failure probability (δ)

$$m \geq \frac{1}{\epsilon}(\ln |H| + \ln(1/\delta))$$

More principles: e.g., loss augmentation

- Bayesian Learning
  - Treat the distribution parameters θ also as a *random variable*
  - The *a posteriori* distribution of θ after seem the data is:

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{p(D)} = \frac{p(D \mid \theta)p(\theta)}{\int p(D \mid \theta)p(\theta)d\theta}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

- (Frequentist) Regularization
  - Treat the parameters θ as a *unknown constant*
  - The *regularized* estimate of θ after seeing the data is

$$\arg\max_{\vec{\theta}} \equiv \mathcal{L}(\{\mathbf{x}_i, \mathbf{y}i\}_{i=1}^{N} \; ; \; \vec{\theta}) + \Omega(\vec{\theta})$$

The prior p(.) , or the regularizer Ω(.), encodes our prior knowledge about the domain

More principles: e.g., inference methods

Variational Inference:

- Maximize the variational lower bound $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$, or equivalently, minimize free energy

$$F(\boldsymbol{\theta}, \phi; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL(q_{\phi}(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}))$$

- E-step: maximize $\mathcal{L}$ wrt. ϕ with $\boldsymbol{\theta}$ fixed

$$\max_{\phi} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x)||p(z))$$

  - If with closed form solutions:

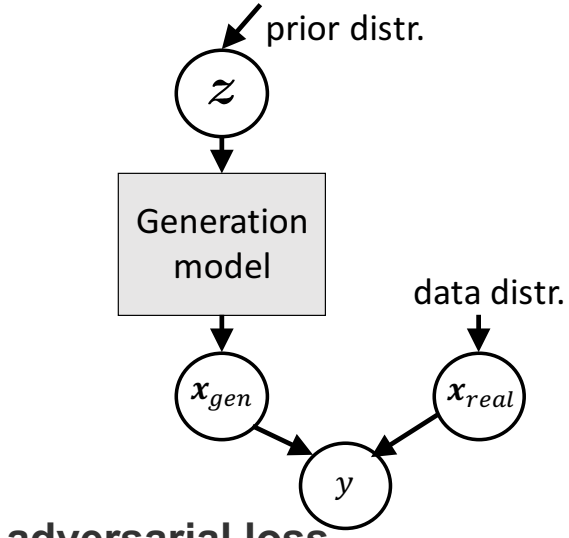
$$q_{\phi}^{*}(z|x) \propto \exp[\log p_{\theta}(x, z)]$$

- M-step: maximize $\mathcal{L}$ wrt. $\boldsymbol{\theta}$ with ϕ fixed

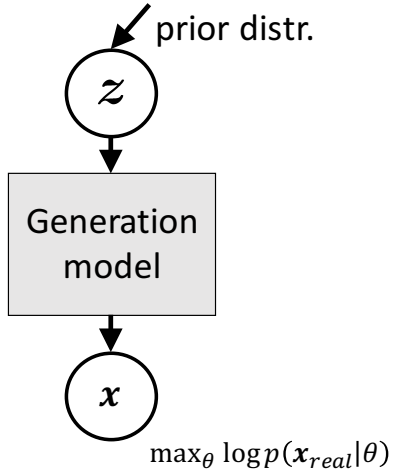
$$\max_{\theta} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x)||p(z))$$

Why useful: demystifying and unifying the zoo of models

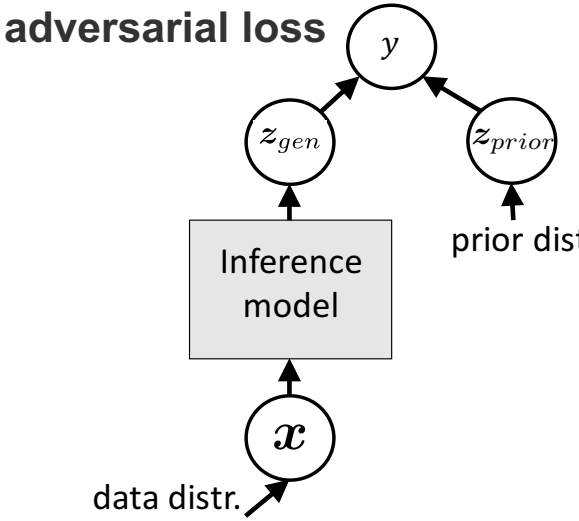
- Deep Generative Models (DGMs): GANs, VAEs, WS, VI …
- No difference in terms of formulations
  - with implicit distributions and black-box NN models
- Difference in terms of space complexity and loss functions
  - depend on the problem at hand
  - choose appropriate tools: implicit/explicit distribution, adversarial/maximum-likelihood optimization, …

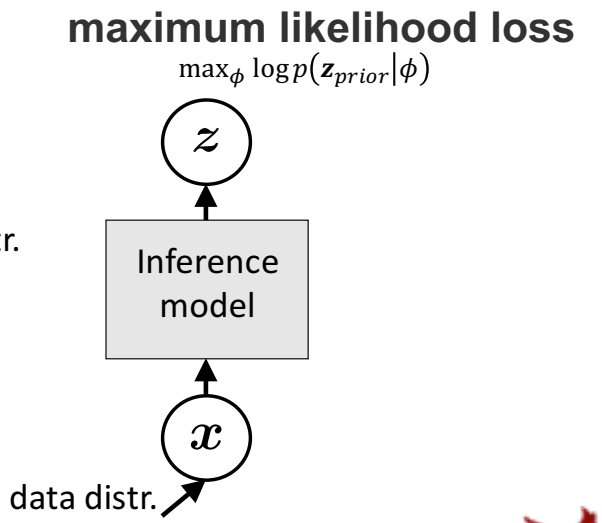


adversarial loss



maximum likelihood loss



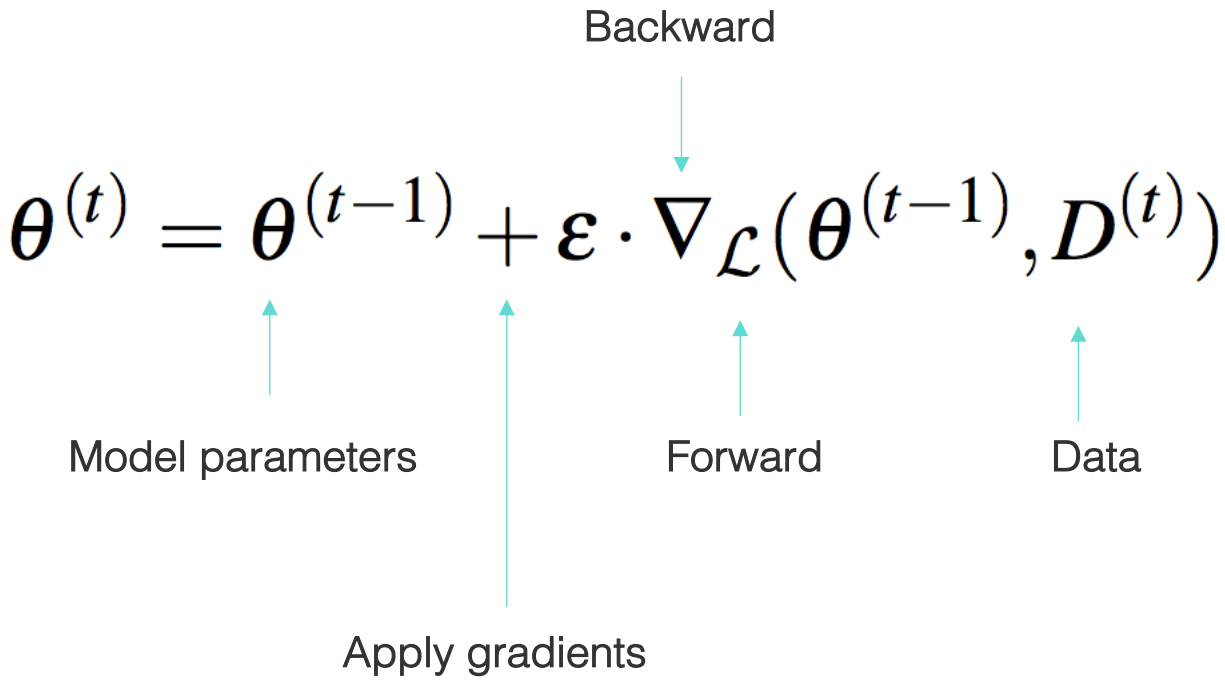


Why useful: design proper model/algorithms with guarantees

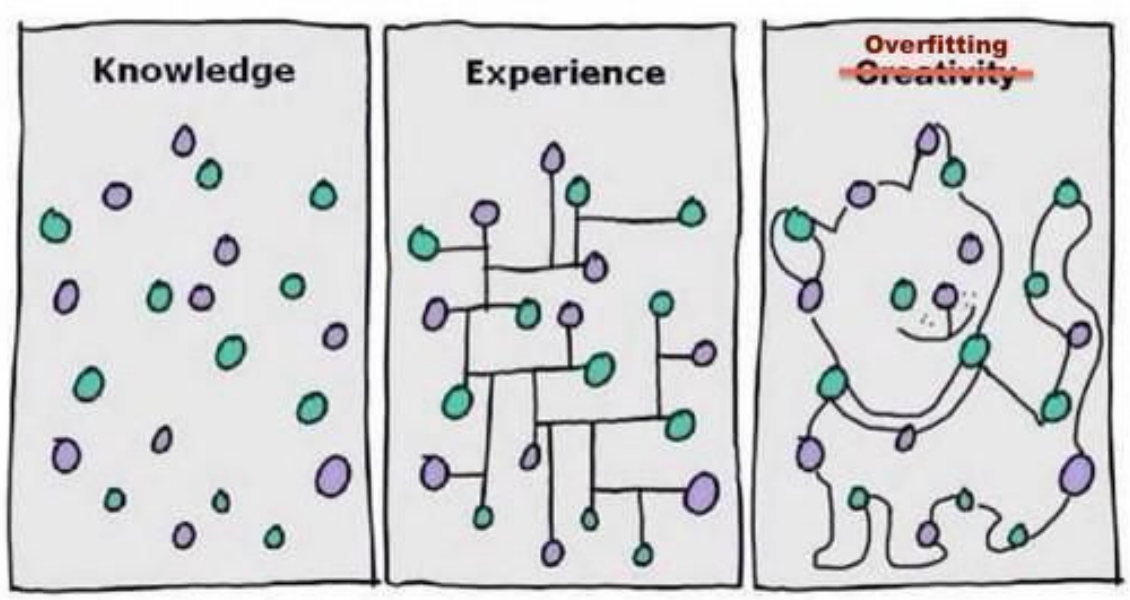
- E.g., A regression model:

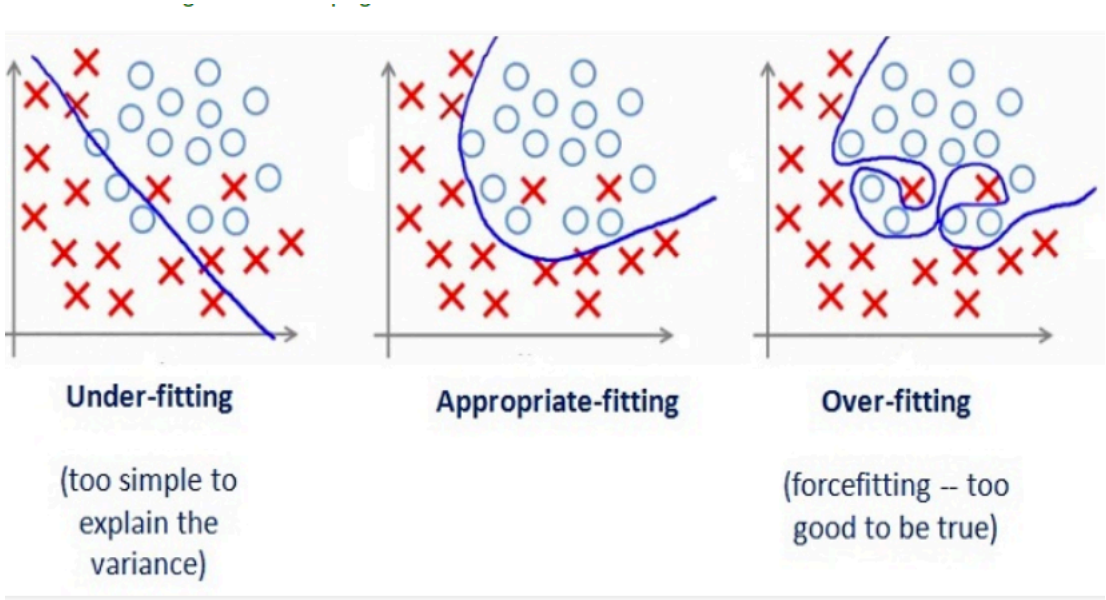
$$\arg\max_{\vec{\theta}} \equiv \mathcal{L}(\{\mathbf{x}_i, \mathbf{y}i\}_{i=1}^{N} \; ; \; \vec{\theta}) + \Omega(\vec{\theta})$$

- E.g., an iterative-convergent gradient algorithm



Why useful: generalizability of your models





Now let's get real

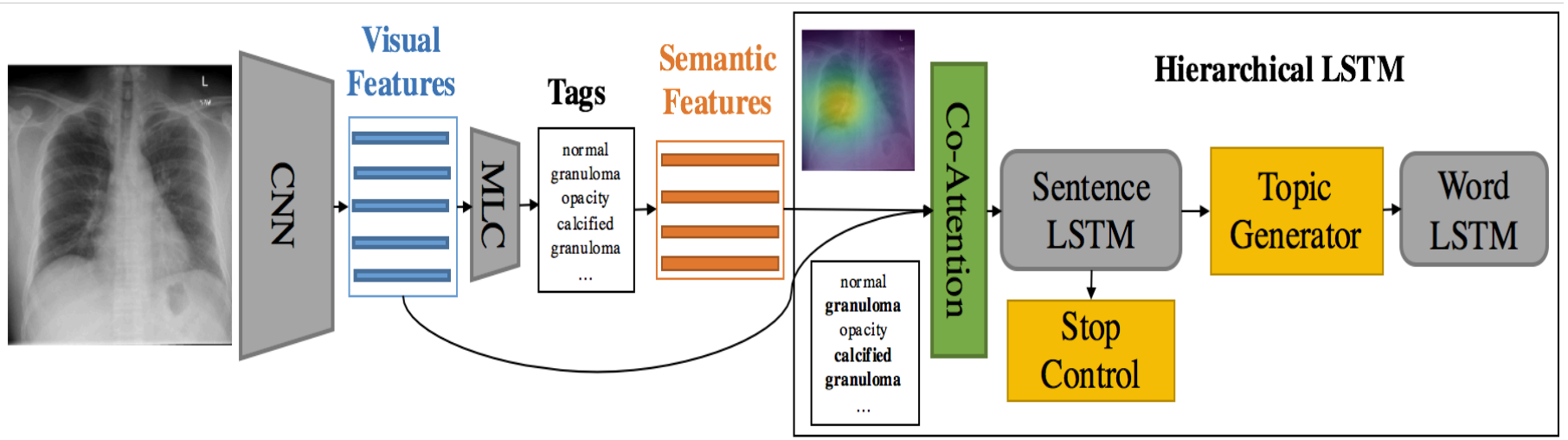
- Chest X-ray Report: multiple sections of information to generate
  - **Findings**: the radiology observations and findings regarding each area of the body examined in the imaging study
  - **Impression**: the radiologist combines the findings, patient clinical history and indication for the imaging study and provides a diagnosis

Findings:
There are no focal areas of consolidation.
No suspicious pulmonary opacities.
Heart size within normal limits.
No pleural effusions.
There is no evidence of pneumothorax.
Degenerative changes of the thoracic spine.

Impression:
No acute cardiopulmonary abnormality.

- **Abnormal regions in medical images are difficult to identify.**
- **The reports are typically long, containing multiple sentences.**
- **Each sentence discusses a specific topic. How to localize the image regions and tags that are relevant to this topic?**

Many methods are needed …



- Semi-supervised learning for lesion tag prediction
- Hierarchical LSTM for long-paragraph generation
- Visual-semantic co-attention to localize the relevant image regions and tags for each sentence to be generated

Still make a delicate craft?

Build – the AI version of “civil engineering”

- Process
- Nuts and Bolts
- Interoperability
- Soundness

- Builder’s tools and training
- Project management: divide workload among multiple builders

Build versus Craft

What does “Build” entail?

- Design
- Plan/Budget
- Materials (data, knowledges, infrastructure, …)
- Specs
- Methods
- Implementation
- Integration
- Refinement/Tuning
- Deployment
- Normal and stress test
- Maintenance and upgrade
- Security
- …

- **Nuts and Bolts** ✓
- **Interoperability** ✓
- **Process**
- **Soundness**

An ML program

$$\arg\max_{\vec{\theta}} = \mathcal{L}(\{\mathbf{x}_i, \mathbf{y}i\}_{i=1}^{N} \; ; \; \vec{\theta}) + \Omega(\vec{\theta})$$

Model Data Parameter

Solved by an iterative convergent algorithm

```
for (t = 1 to T) {
  doThings()
      θ^{t+1} = g(θ^t, Δ_f θ(D))
  doOtherThings()
}
```

This computation needs to be parallelized!

Proximal gradient (a.k.a., ISTA)

$$\min_{\mathbf{w}} \ell(\mathbf{w}) + r(\mathbf{w})$$

- ℓ : loss, for now smooth (continuously differentiable)
- r : regularizer, non-differentiable (e.g. 1-norm)

Projected gradient

- ***r* represents some constraint**

$$r(\mathbf{w}) = \iota_C(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \in C \\ \infty, & \text{otherwise} \end{cases}$$

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \eta \nabla \ell(\mathbf{w}) \\ \mathbf{w} &\leftarrow \arg\min_{\mathbf{z}} \frac{1}{2\eta} \|\mathbf{w} - \mathbf{z}\|^2 + \iota_C(\mathbf{z}) \\ &= \arg\min_{\mathbf{z} \in C} \frac{1}{2} \|\mathbf{w} - \mathbf{z}\|^2 \end{aligned}$$

Proximal gradient

- ***r* represents some simple function**
  - **e.g., 1-norm, constraint C, etc.**

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \eta \nabla \ell(\mathbf{w}) \quad \text{gradient} \\ \mathbf{w} &\leftarrow \underbrace{\arg\min_{\mathbf{z}} \frac{1}{2\eta} \|\mathbf{w} - \mathbf{z}\|^2 + r(\mathbf{z})}_{\text{proximal map}} \end{aligned}$$

Accelerated PG (a.k.a. FISTA)

- PG convergence rate $O(1/(\eta t))$
- Can be boosted to $O(1/(\eta t^2))$
  - Same Lipschitz gradient assumption on f; similar per-step complexity!
  - (Beck & Teboulle'09; Nesterov'13; Tseng'08), lots of follow-up work

Proximal Gradient

$$\mathbf{v}^t \leftarrow \mathbf{w}^t - \eta \nabla \ell(\mathbf{w}^t)$$
$$\mathbf{u}^t \leftarrow \mathsf{P}_r^\eta(\mathbf{v}^t)$$
$$\mathbf{w}^{t+1} \leftarrow \mathbf{u}^t + \underbrace{0}_{no} \cdot \underbrace{(\mathbf{u}^t - \mathbf{u}^{t-1})}_{momentum}$$

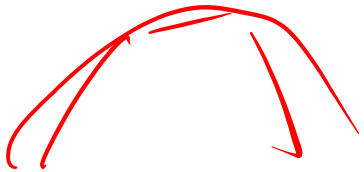
Accelerated Proximal Gradient

$$\mathbf{v}^t \leftarrow \mathbf{w}^t - \eta \nabla \ell(\mathbf{w}^t)$$
$$\mathbf{u}^t \leftarrow \mathsf{P}_r^\eta(\mathbf{v}^t)$$
$$\mathbf{w}^{t+1} \leftarrow \mathbf{u}^t + \underbrace{\frac{t-1}{t+2}}_{\approx 1} \underbrace{(\mathbf{u}^t - \mathbf{u}^{t-1})}_{momentum}$$

$$\mathsf{P}_r^\eta(\mathbf{w}) := \arg\min_{\mathbf{z}} \frac{1}{2\eta}\|\mathbf{w}-\mathbf{z}\|_2^2 + r(\mathbf{z})$$

Smoothing proximal gradient

- Use Moreau envelope as smooth approximation
  - Rich and long history in convex analysis (Moreau'65; Attouch'84)
- Inspired by the proximal point algorithm (Martinet'70; Rockafellar'76)
  - Proximal point alg = PG, when $f \equiv 0$
- Rediscovered in (Nesterov'05), lead to SPG (Chen et al.'12)



$$\min_{\mathbf{w}} f(\mathbf{w}) + g(\mathbf{w}) \quad \Leftarrow \text{original}$$

$$\text{approx.} \Rightarrow \quad \approx \min_{\mathbf{w}} \mathsf{M}_f^{\eta}(\mathbf{w}) + g(\mathbf{w})$$

- With $\eta = O(1/t)$, SPG converges at $O(1/(\eta t^2)) = O(1/t)$
- Improves subgradient $O(1/\sqrt{t})$
- Requires both efficient P_f^{η} and P_g^{η}

Smoothing Proximal Gradient

$$\mathbf{v}^t \leftarrow \overbrace{\mathbf{w}^t - \eta \nabla \mathsf{M}_f^{\eta}(\mathbf{w}^t)}^{=\mathsf{P}_f^{\eta}(\mathbf{w}^t)}$$

$$\mathbf{u}^t \leftarrow \mathsf{P}_g^{\eta}(\mathbf{v}^t)$$

$$\mathbf{w}^{t+1} \leftarrow \mathbf{u}^t + \frac{t-1}{t+2} \underbrace{(\mathbf{u}^t - \mathbf{u}^{t-1})}_{momentum}$$

Data-Parallel for large-scale problems

- Model (e.g. SVM, Lasso …):

$$\min_{\mathbf{a}\in\mathbb{R}^d} \mathcal{L}(\mathbf{a}, D), \quad \text{where} \quad \mathcal{L}(\mathbf{a}, D) = f(\mathbf{a}, D) + g(\mathbf{a})$$

data D, model a

- Algorithm:
  - Update $\mathbf{a}(t) := \text{prox}_g\left(\mathbf{a}^p(t) - \eta(t)\sum_{(p',t')\in Recv^p(t)} \Delta(\mathbf{a}^{p'}(t'), D_{p'})\right)$

sub-update

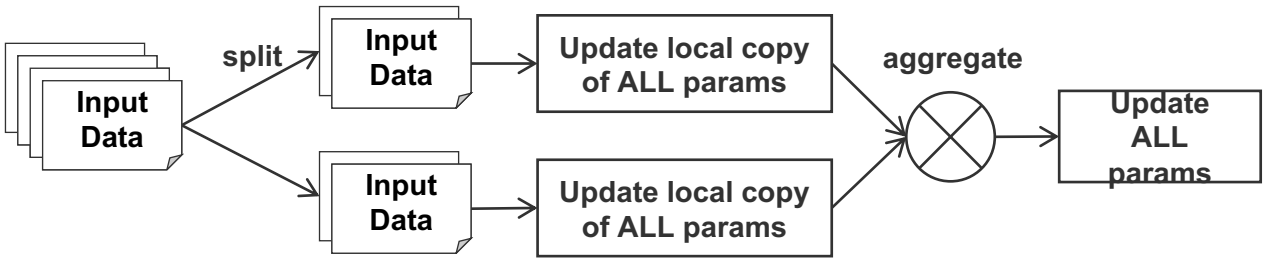
proximal step wrt g

stale sub-updates Δ() received by worker p at iteration t

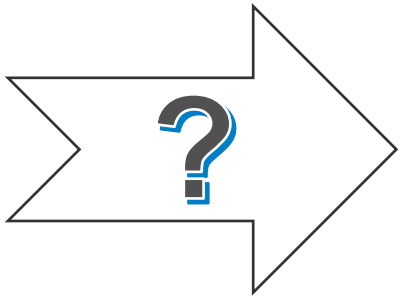
  - sub-update $\Delta(\mathbf{a}^p(t), D_p) := \nabla f(\mathbf{a}^p(t), D_p)$

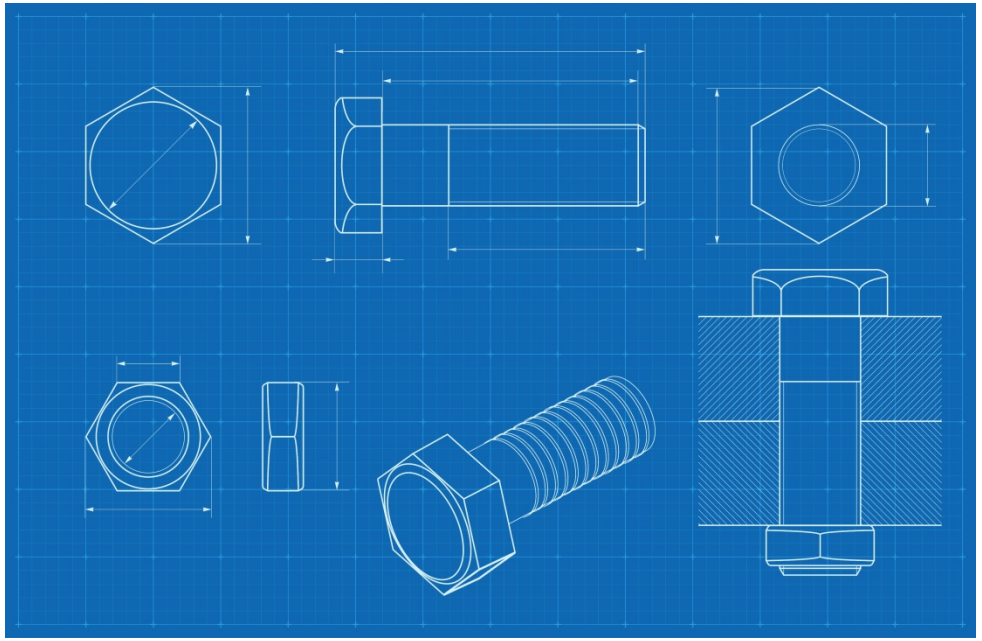
gradient step wrt f

- Data parallel:
  - Data D too large to fit in a single worker, divide among P workers



- How to modularize and standardize these steps/elements?





Nuts and Bolts: complete, reusable, robust

- Data wrangling
- Machine learning
- System harmonization

Examples

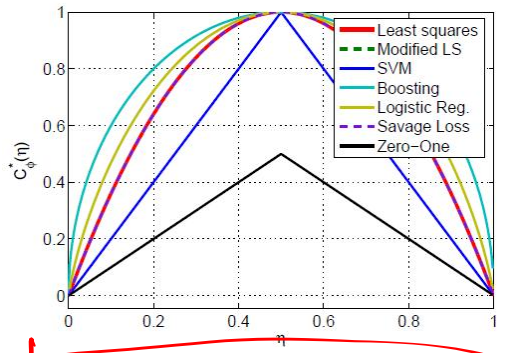
Data Transforms and Feature Engineering

- PCA, Sliding Window, n-th order Derivatives, Discretization, SIFT, Wavelets, Neural Network Embedding, …

Machine Learning

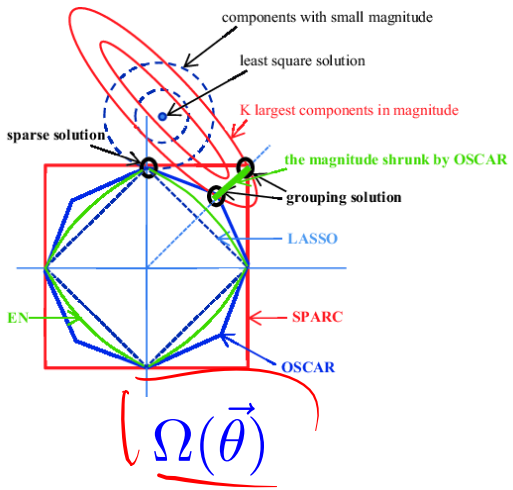
$$\arg\max_{\vec{\theta}} \equiv \mathcal{L}(\{\mathbf{x}_i, \mathbf{y}i\}_{i=1}^{N} \; ; \; \vec{\theta}) + \Omega(\vec{\theta})$$

Loss Functions



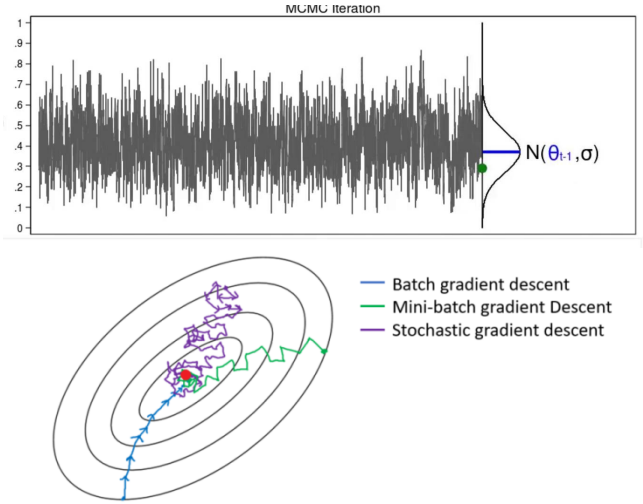
$\mathcal{L}(\{\mathbf{x}_i, \mathbf{y}i\}_{i=1}^{N} \; ; \; \vec{\theta})$

Regularizers/Priors



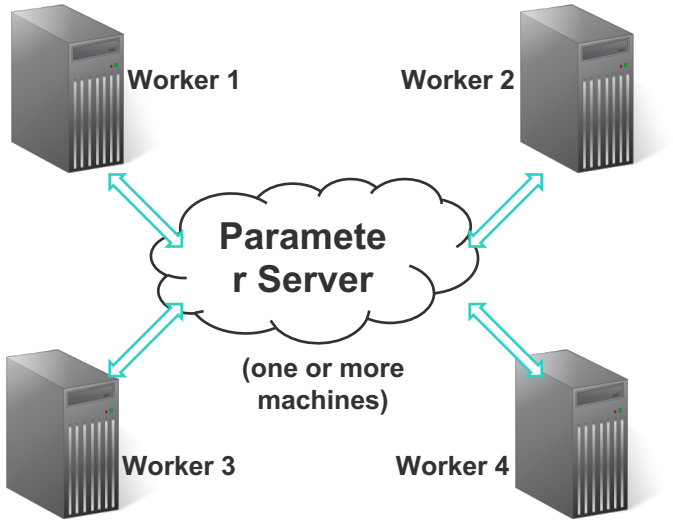
$\Omega(\vec{\theta})$

Training Algorithms



Examples cont.

- Distributed ML via SSP Parameter Server
- Interoperable – no change/massaging of ML algo implementation!
  - Simply call different subroutine/interface and easily switch to distributed operation

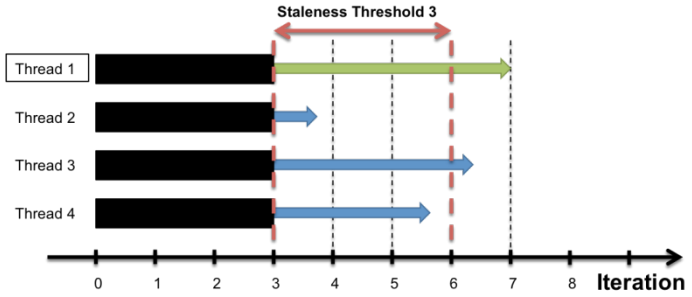


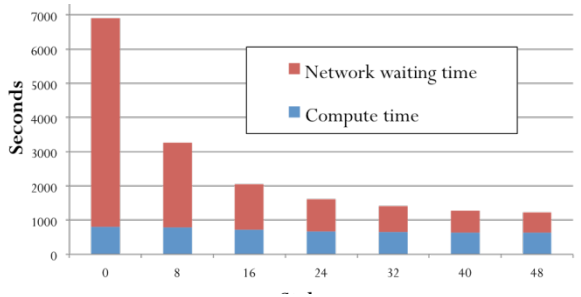
Single Machine Parallel

```
UpdateVar(i) {
  old = y[i]
  delta = f(old)
  y[i] += delta
}
```

Distributed with Bösen PS

```
UpdateVar(i) {
  old = PS.read(y,i)
  delta = f(old)
  PS.inc(y,i,delta)
}
```

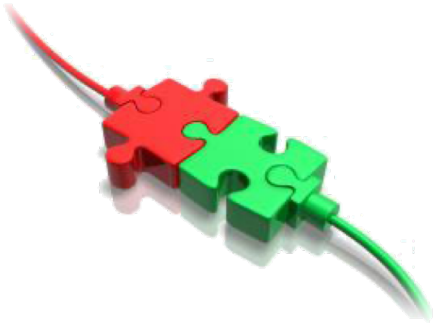




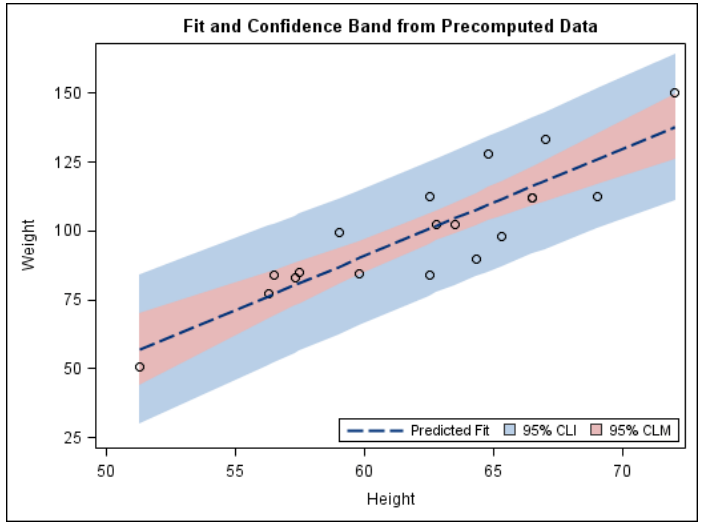
An inventory of ML/Sys nuts and bolts

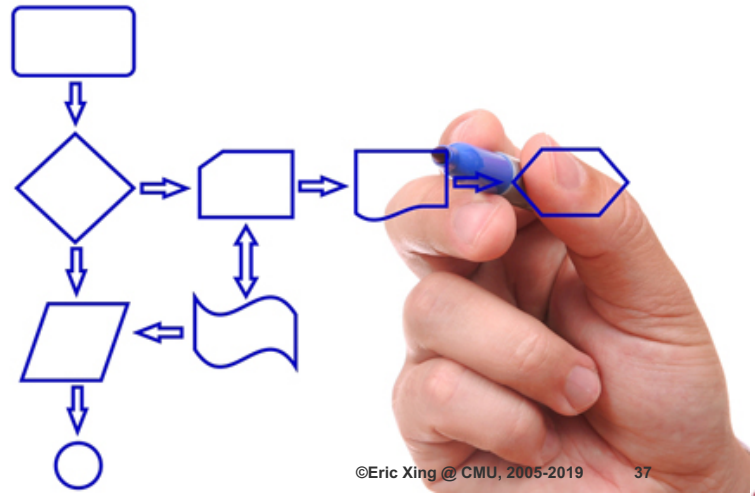
Data Machine	**Ingestion**	**Cleaning & Improvement**	**Feature Engineering**	**Embedding**	**Integration**	**Interpret (DM X-Ray)**	**Dev**	**Resource**
	• Input Source • Data Type Detection • Image, Video, Tabular, Time Series, Graph, …	• Auto Quality Check • Missing Values • Text Parsing, Image Whitening, …	• SIFT Features, Wavelets, Bag-of-Words, TFIDF, N-th order derivatives, …	• PCA, LDA, Autoencoders, RNN, CNN, Locally-Linear Embedding, …	• Join, Split, Intersect, Union • Co-embedding • Export Data	• Real-time visualization and feedback • Point clouds, word maps, heat maps, …	• Command line scripting • Add new Data nuts and bolts • Open source languages	• Multi-machine • Network, Disk, Memory • Job Management
Machine Learning	**Build**	**Augment and Tune**	**Train**	**Score**	**Experiment**	**Interpret (Model X-ray)**	**Dev**	**Resource**
	• Combine, arrange multiple models • Supervised, Unsupervised, Bayesian, Deep, Active, …	• Priors, Regularizers • Hyperparameters • Auto-tuning	• Optimization methods, Sampling methods • Initialization criteria • Termination criteria	• Auto Quality Check • Cross-validation, loss, F1, AUC, … • Business-specific criteria, e.g. factory throughput	• Side-by-side visualization and comparison • Model selection/tuning • Auto-versioning	• Real-time visualization and feedback • Convergence metrics, quality scores, weights	• Command line scripting • Add new ML nuts and bolts • Open source languages	• Multi-machine • Network, Disk, Memory • Job Management
Operating System	**Deploy**	**Storage**	**Compatibility**	**User Accounts and Security**	**Project Management**	**Interpret (Process X-ray)**	**Dev**	**Resource**
	• Launch on different environments • Containerized and Virtualized • Real-time	• Manage content – ML Processes, Nuts and Bolts, Visualizations, Deployed Applications, …	• Datacenter, Cloud, Laptop, Workstation, Mobile, IoT • Linux, Windows or Mac	• Add/remove users • Security and Compliance	• Collaborate on and Share ML Processes • ML Application Development	• Resource Dashboard • Real-time Feedback • System Diagnostics	• Command line scripting • Solutions SDK • Operating System and Filesystem control	• Multi-machine • Network, Disk, Memory • Job Management

Interoperability: interface, standards, specs, tools



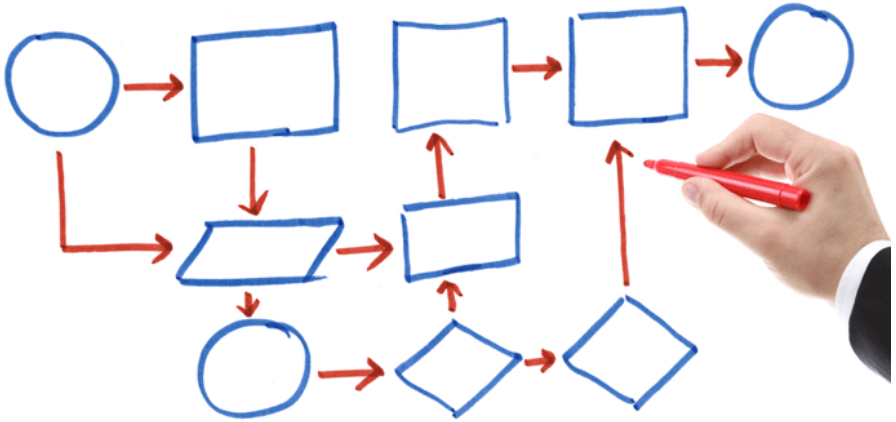
- Standards
  - Interfaces between nuts and bolts
  - Guaranteed behavior for deterministic nuts and bolts (e.g. data transform)
  - Bounded behavior for statistical/random nuts and bolts (e.g. stochastic training)
- Tools
  - Design Tools (e.g. CAD) – "draw" ML process
  - Deployment Tools (e.g. hammers) to compile into ML applications

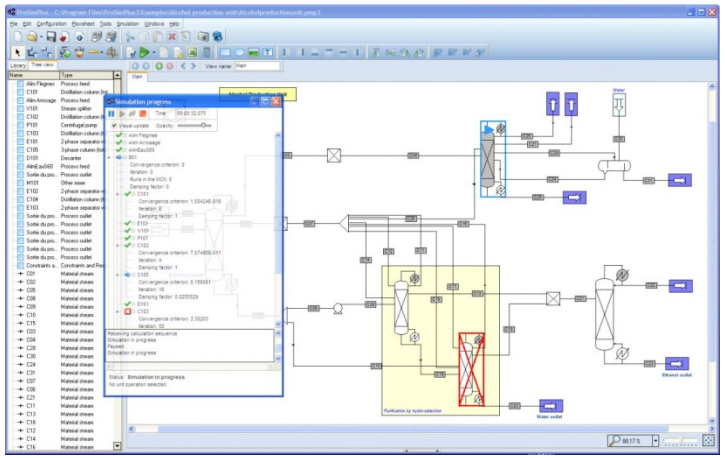


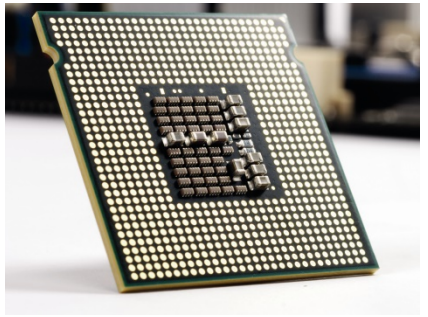


Process: actionable, explainable, and repeatable

- Create
- Simulate
- Experiments
- Debug
- Integrate
- Explain
- Manage
- Track







Soundness: correctness guarantees and theory

- Upon ..
  - Data Transformation
  - Model/Parameter estimation
  - Query/Inference
  - Stochastic Sampling
  - Distribution/Parallelization
  - Augmentation/Refinement
  - Porting
  - Operational stress such as faulty infra
  - Security breach
  - …
- Characterize the error and risk, or allow simulational study

Texar: a toolkit for Text Generation

Text Generation Tasks

- Generates *natural language* from input *data or machine representations*
- Spans a broad set of natural language processing (NLP) tasks:

Task	Input *X*	Output *Y* (Text)
Chatbot / Dialog System	Utterance	Response
Machine Translation	English	Chinese
Summarization	Document	Short paragraph
Description Generation	Structured data	Description
Captioning	Image/video	Description
Speech Recognition	Speech	Transcript

Courtesy: Neubig, 2017

Various (Deep Learning) Techniques

Other Techniques

- Optimization
- Data pre-processing
- Result post-processing
- Evaluation
- …

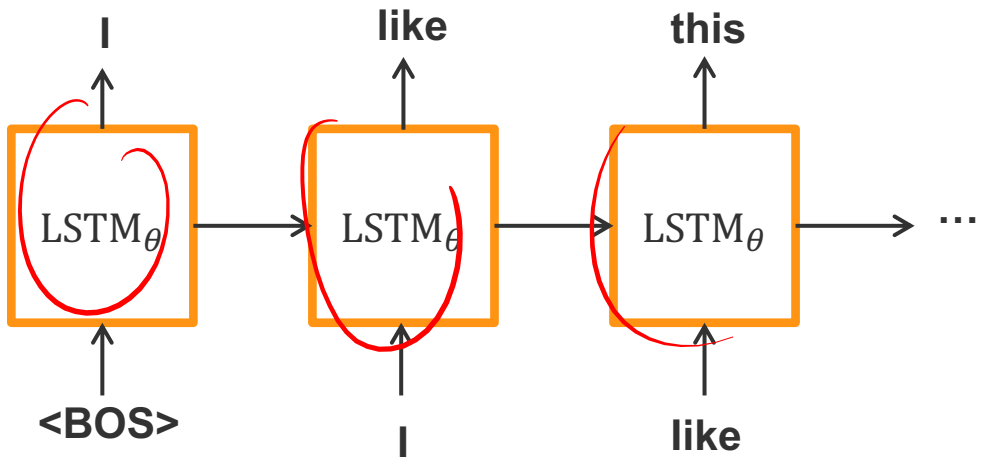
Models / Algorithms

- Neural language models
- Encoder-decoders
- Seq/self-Attentions
- Memory networks
- Adversarial methods
- Reinforcement learning
- Structured supervision
- …

Example: Language Model

- Calculates the probability of a sentence:
  - Sentence: $\boldsymbol{y} = (y_1, y_2 \cdots, y_T)$

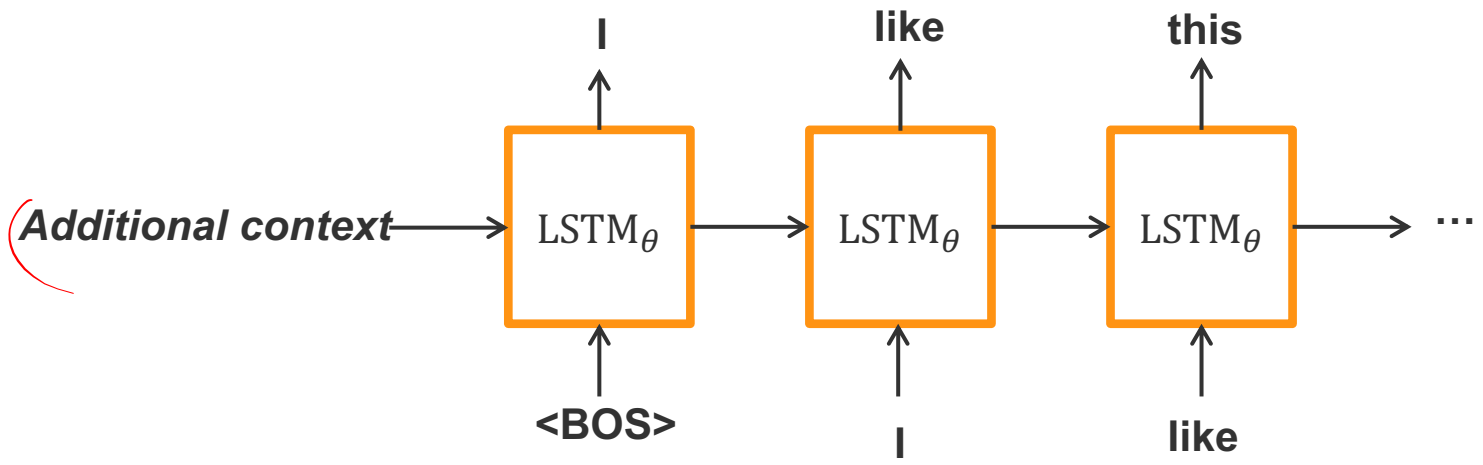
$$p_\theta(\boldsymbol{y}) = \prod_{t=1}^{T} p_\theta(y_t \mid \boldsymbol{y}_{1:t-1})$$



Example: *Conditional* Language Model

- Conditions on additional task-dependent context $\boldsymbol{x}$
  - Machine translation: (representation of) source sentence
  - Medical image report generation: (representation of) medical image

$$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x})$$

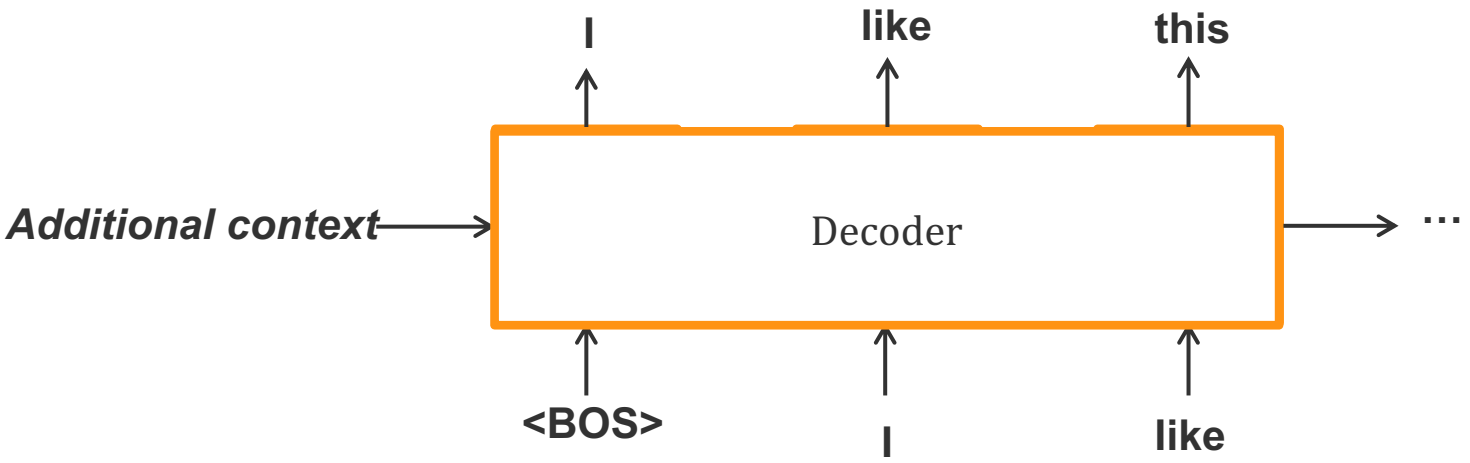


Example: *Conditional* Language Model

- Conditions on additional task-dependent context $\boldsymbol{x}$
  - Machine translation: (representation of) source sentence
  - Medical image report generation: (representation of) medical image

$$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x})$$

- **Language model as a decoder**

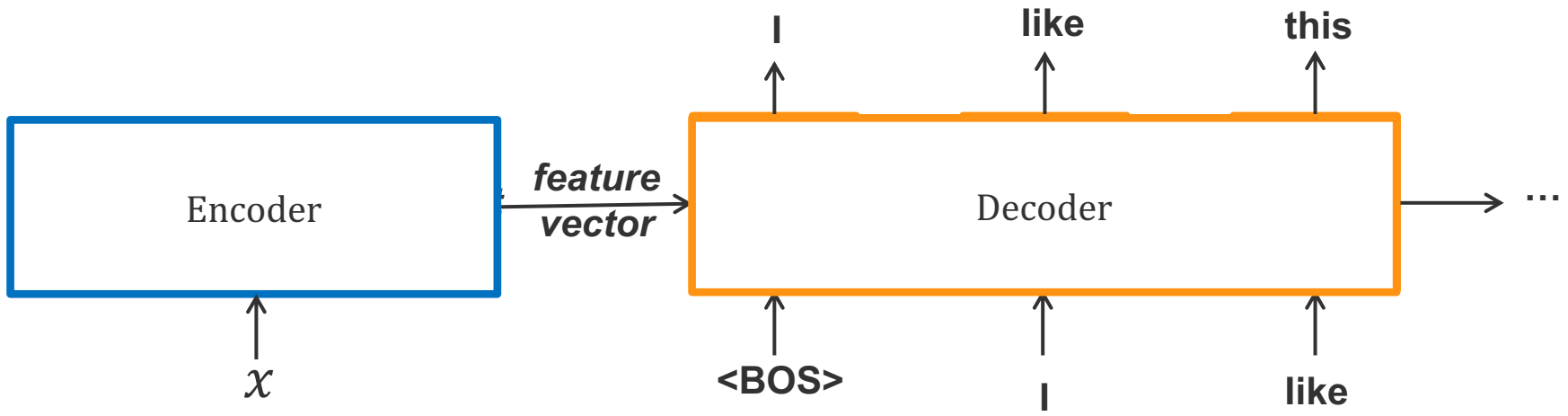


Example: *Conditional* Language Model

- Conditions on additional task-dependent context $\boldsymbol{x}$
  - Machine translation: (representation of) source sentence
  - Medical image report generation: (representation of) medical image

$$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x})$$

- **Language model as a decoder**
- **Encodes context with an encoder**



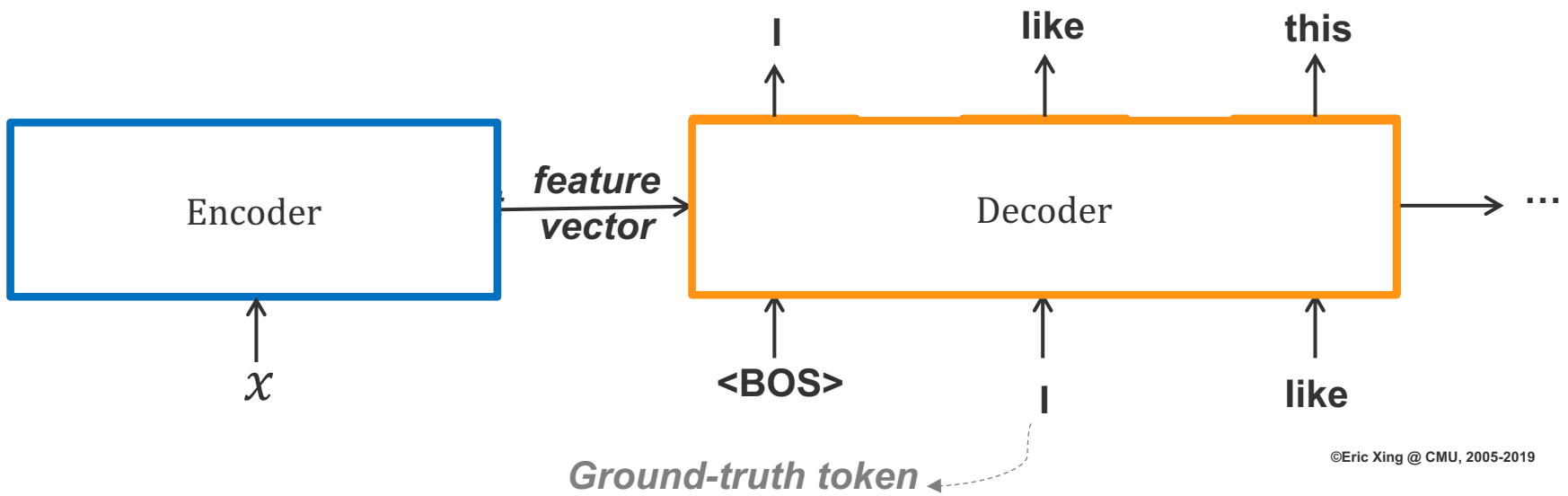
Training: Maximum Likelihood Estimation (MLE)

- Given data example $(\boldsymbol{x}, \boldsymbol{y}^*)$
- Maximizes log-likelihood of the data

$$\max_\theta \mathcal{L}_{\text{MLE}} = \log p_\theta(\boldsymbol{y}^* \mid \boldsymbol{x})$$

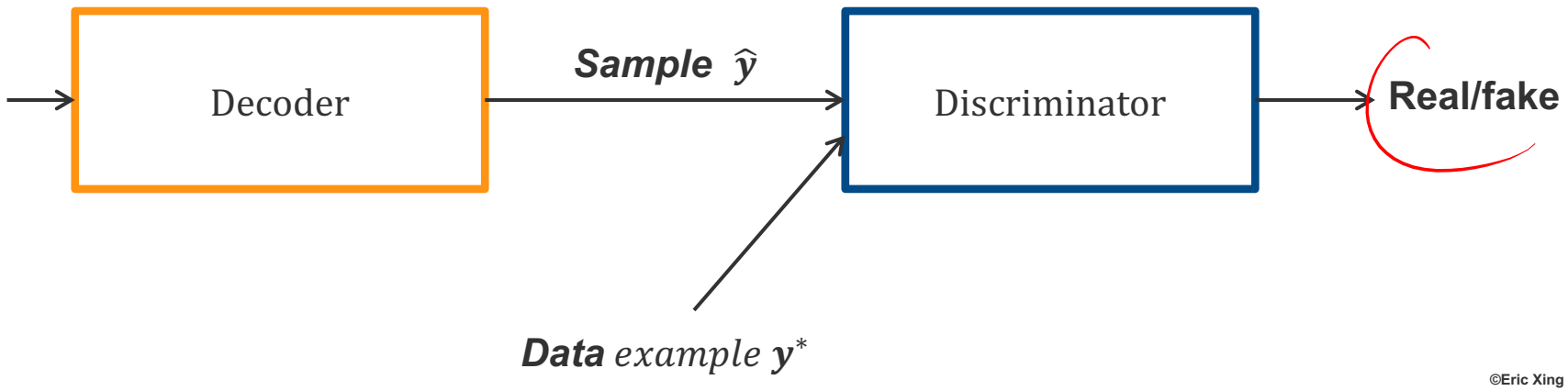
Teacher-forcing decoding:

- **For every step t, feeds in the ground-truth token y_t^* to decode next step**



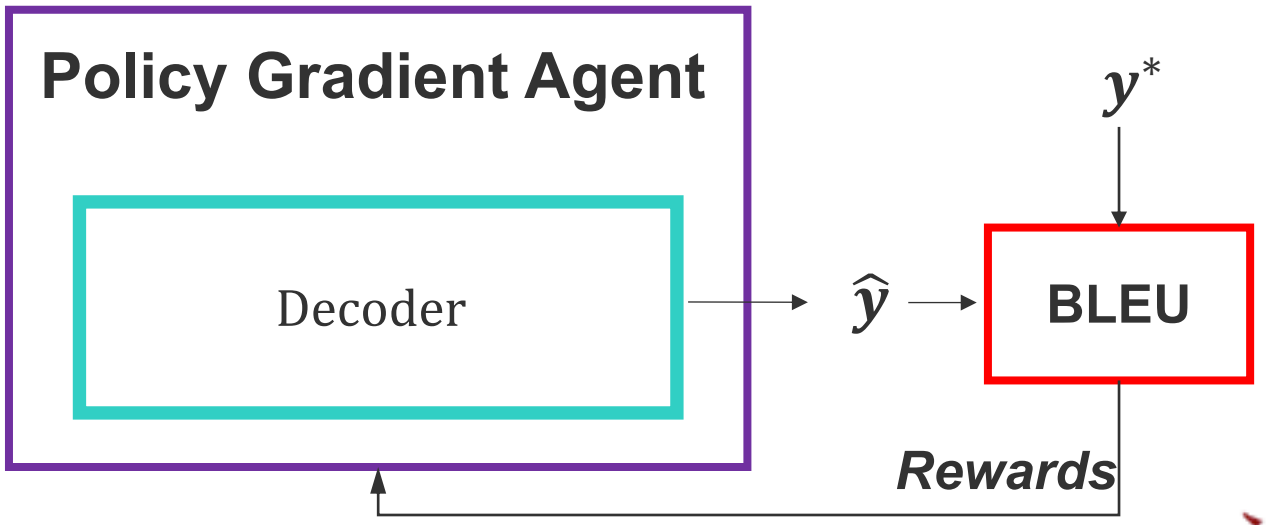
Training: Adversarial Learning

- A discriminator is trained to distinguish between real data examples and fake generated samples
- Decoder is trained to confuse the discriminator
- *Sample* $\hat{\boldsymbol{y}}$ is discrete: not differentiable
  - disables gradient backpropagation from the Discriminator to the Decoder
- Uses a differentiable approximation of $\hat{\boldsymbol{y}}$: *Gumbel-softmax decoding*



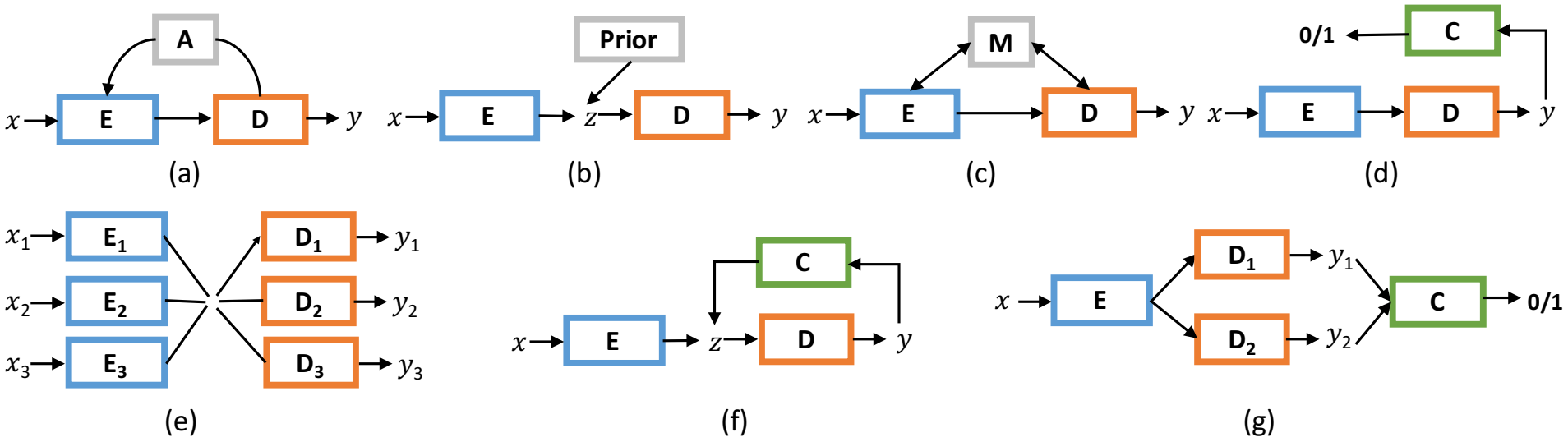
Training: Reinforcement Learning

- Optimizes test metric (e.g., BLEU) directly
- Decoder generates sample $\hat{y}$ which is used to evaluate reward
  - *Greedy decoding / sample decoding / beam search decoding*



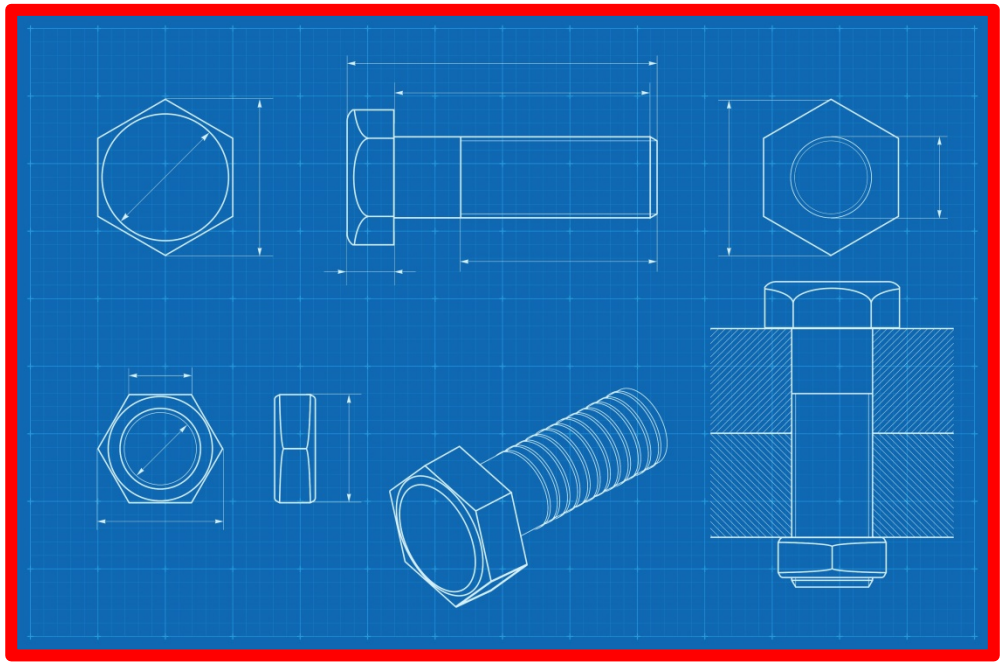
Various (Deep Learning) Techniques (cont'd)

- Techniques are often combined together in various ways to tackle different problems
  - An example of various model architectures



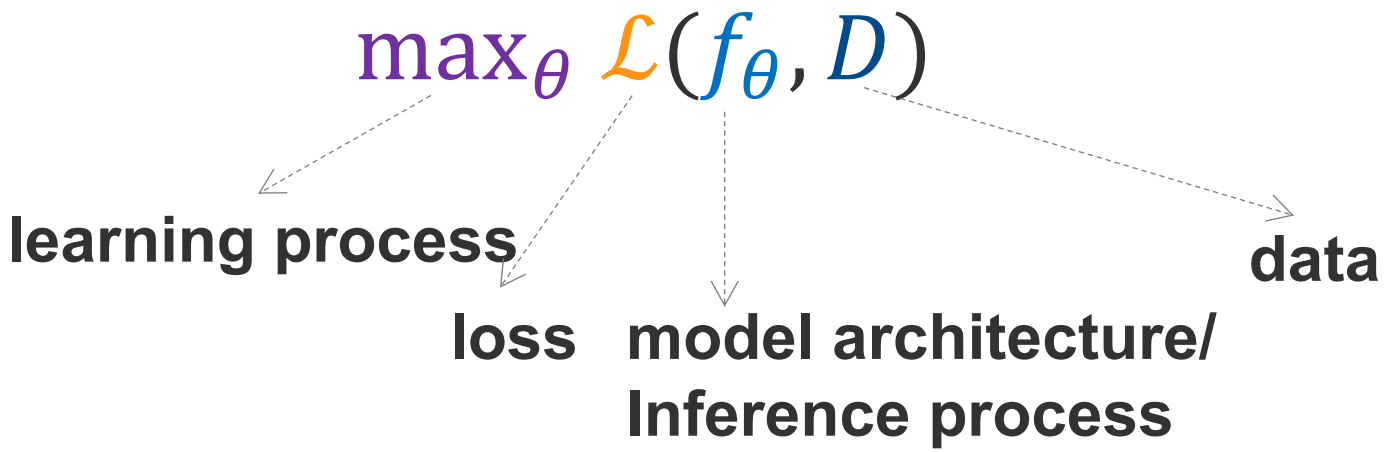
E refers to encoder, D to decoder, C to Classifier, A to attention, Prior to prior distribution, and M to memory

How to modularize and standardize?



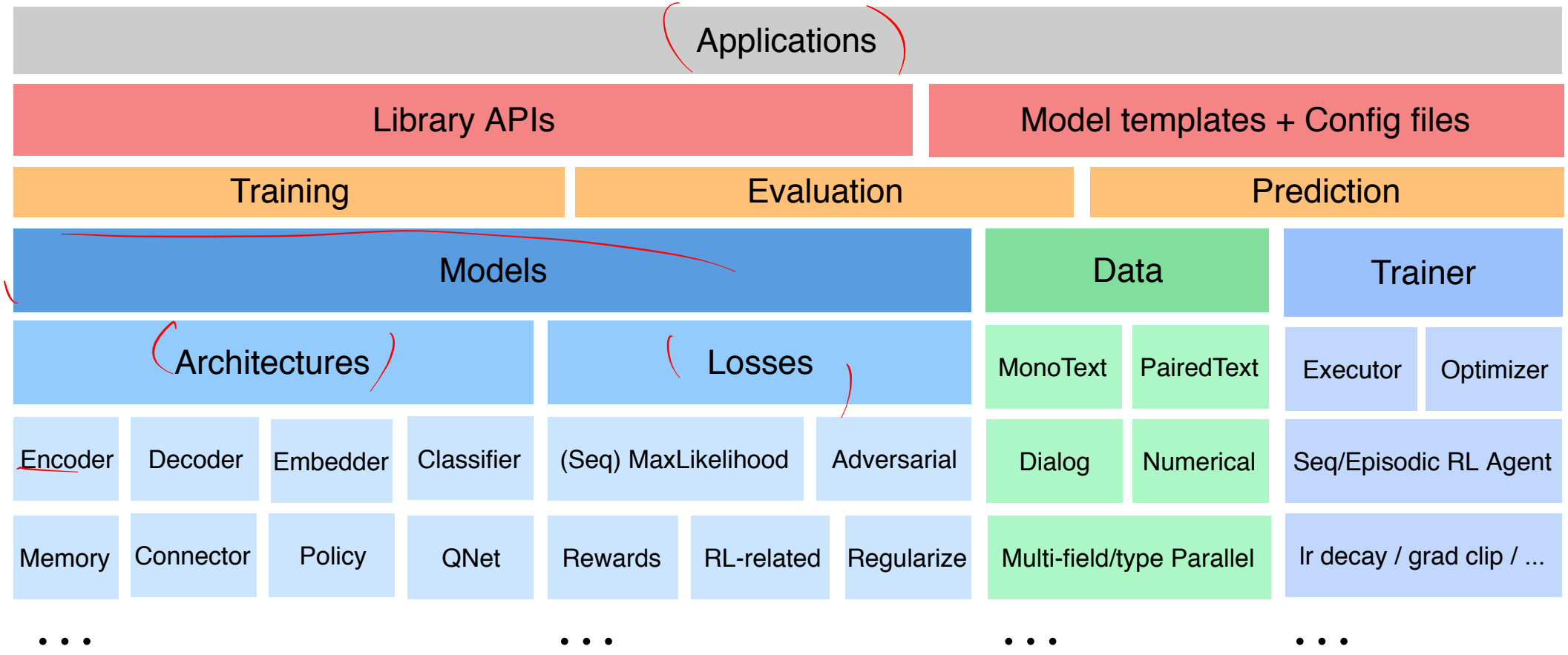
Pipeline Decomposition

- Decomposes ML models/algorithms into highly-reusable model architecture, loss, learning process, and data modules, among others

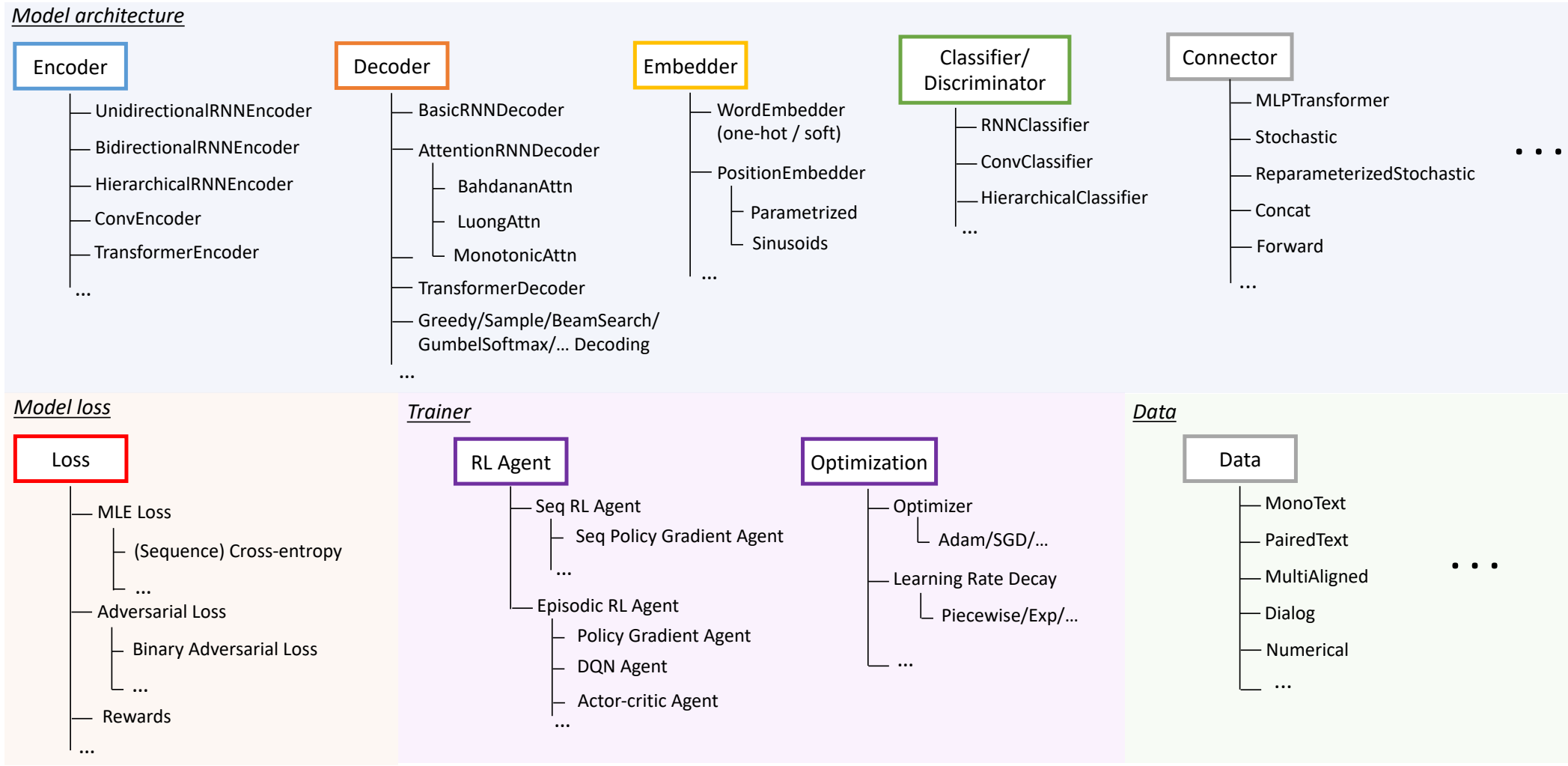


Texar Stack

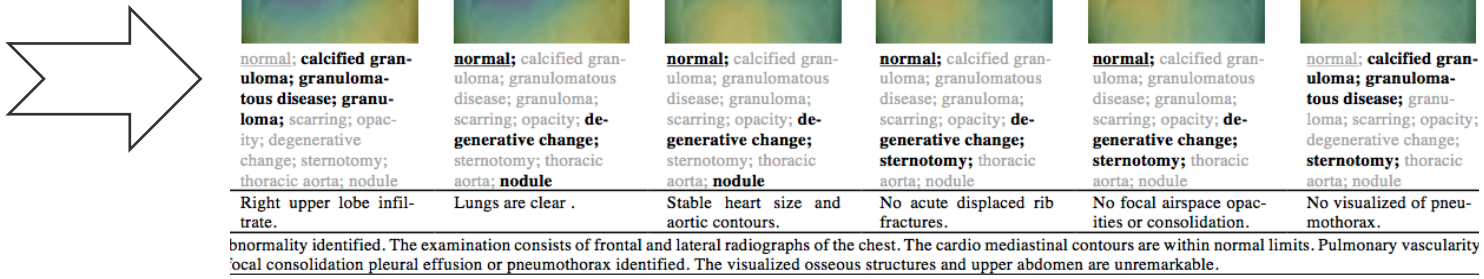
***Texar** stack*

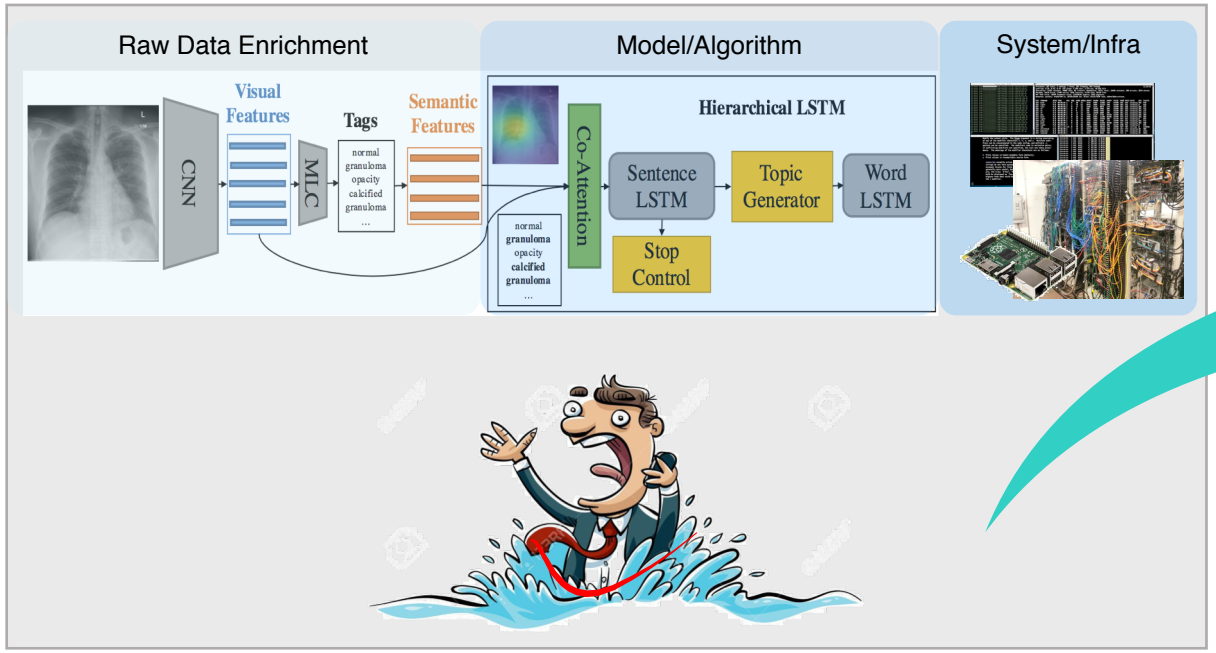


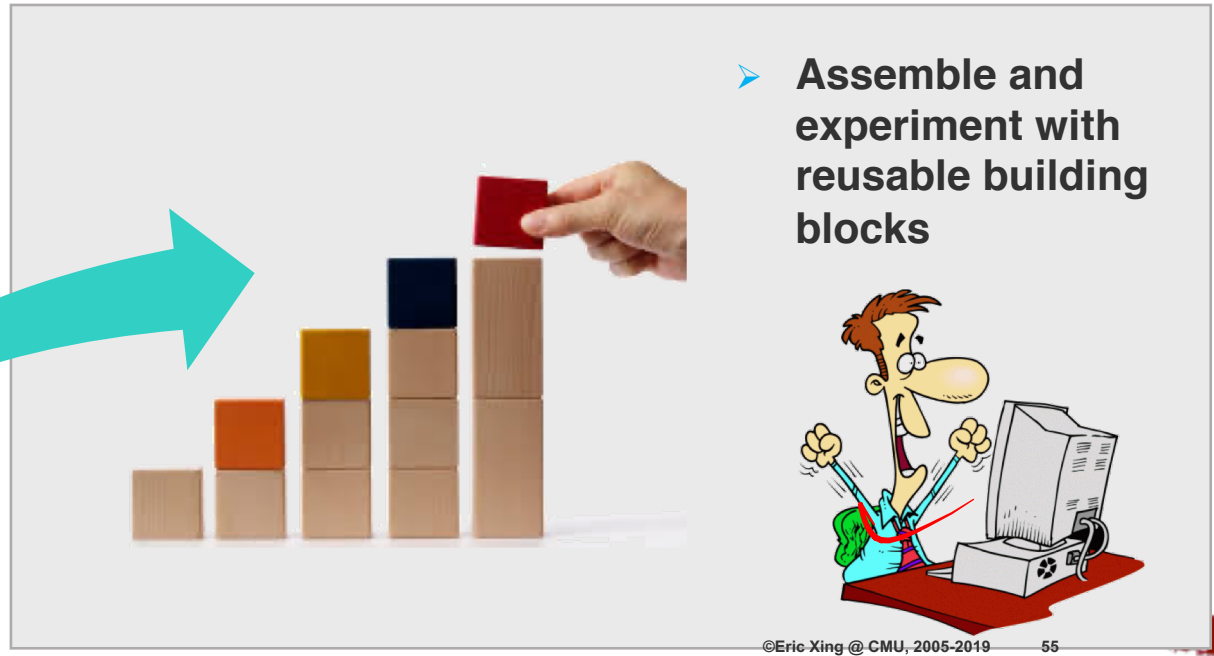
Module Catalog



AI WITH NO TEARS



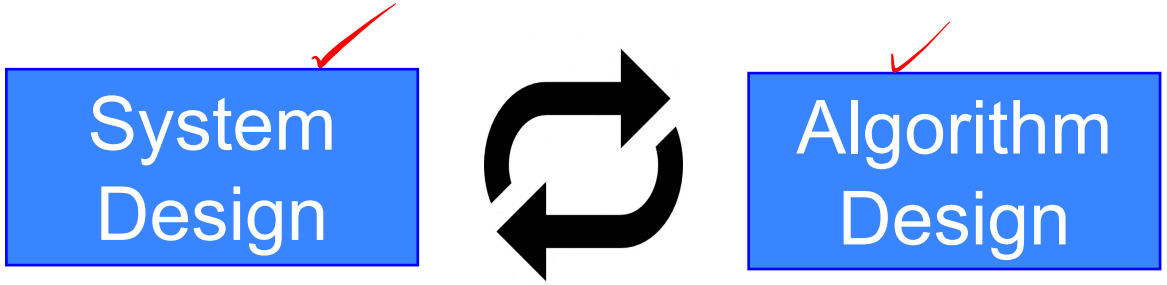




- **Assemble and experiment with reusable building blocks**

PETUUM

The Science of SysML



System/Algorithm Co-design

- System design should be tailored to the unique mathematical properties of ML algorithms
- Algorithms can be re-designed to better exploit the system architectures

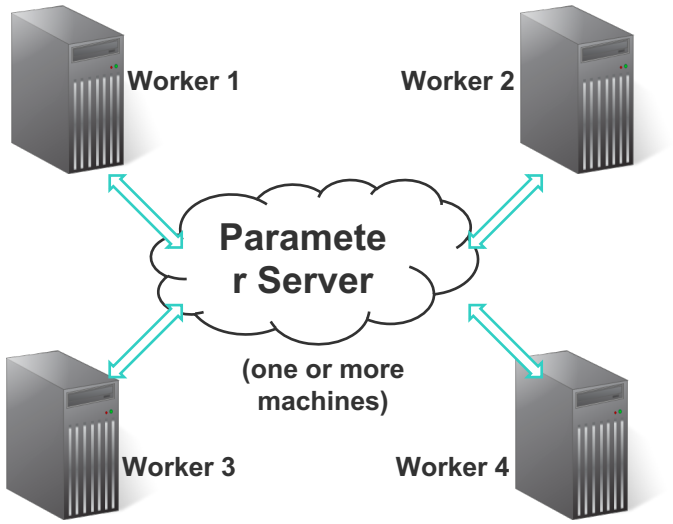
PETUUM

Distributed Machine Learning

1. How to Distribute? (Ho et al NIPS 2013, Jin, et al. EuroSys, 2015, Wei et al. SoCC 2015)
2. How to Bridge Computation and Communication? (Ho et al NIPS 2013, Dai et al, AAAI 2014)
3. What to Communicate? (Xie et al. UAI 2015, Xie et al, SoCC 2018)
4. How to Communicate? (Zhang et al, ATC 2017, Xie et al, SoCC 2018)

Distributed ML via the Parameter Server

- Distributed ML via SSP Parameter Server
- Interoperable – no change/massaging of ML algo implementation!
  - Simply call different subroutine/interface and easily switch to distributed operation

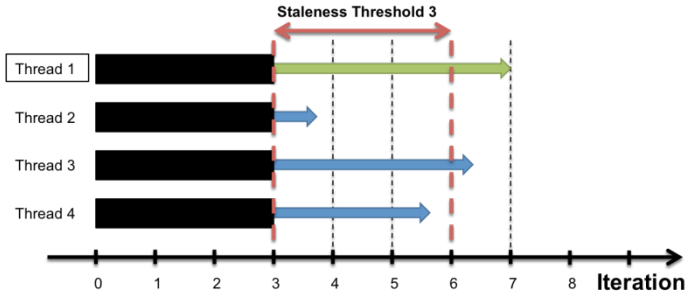


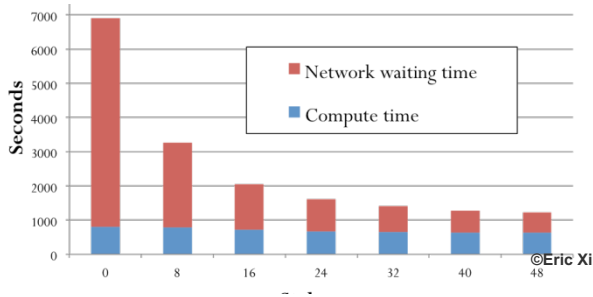
Single Machine Parallel

```
UpdateVar(i) {
  old = y[i]
  delta = f(old)
  y[i] += delta
}
```

Distributed with Bösen PS

```
UpdateVar(i) {
  old = PS.read(y,i)
  delta = f(old)
  PS.inc(y,i,delta)
}
```

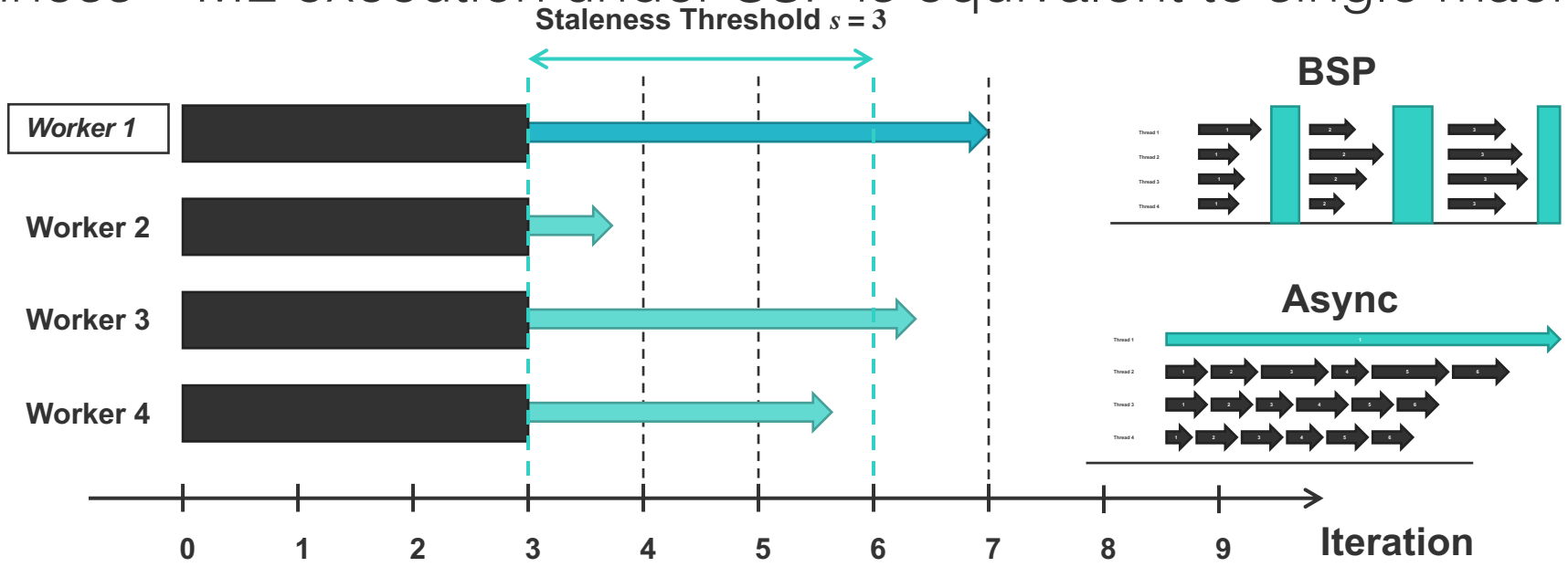




The stale synchronous parallel bridging model for PS

[Ho et al., 2013]

- Soundness – ML execution under SSP is equivalent to single machine ?



Stale Synchronous Parallel (SSP)

- **Fastest/slowest workers not allowed to drift >s iterations apart**

Consequence

- **Fast like async, yet correct like BSP**
- **Why? Workers' local view of model parameters "not too stale" (≤s iterations old)**

SSP data parallel convergence theorem [Ho et al., 2013, Dai et al., 2015]

- Soundness – ML execution under SSP is equivalent to single machine

Let observed staleness be γ_t
Let staleness mean, variance be $\mu_\gamma = \mathbb{E}[\gamma_t] \quad \sigma_\gamma = var(\gamma_t)$

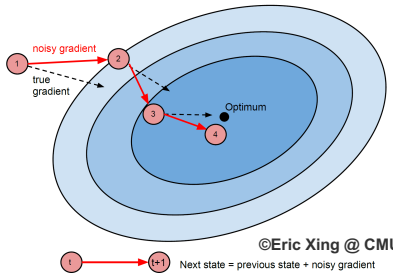
Theorem: Given L-Lipschitz objective f_t and step size h_t,

$$P\left[\frac{R[X]}{T} - \frac{\mathcal{O}(F^2 + \mu_\gamma L^2)}{\sqrt{T}} \geq \tau\right] \leq \exp\left\{\frac{-T\tau^2}{\mathcal{O}(\bar{\eta}_T \sigma_\gamma + L^2 s P \tau)}\right\}$$

whe

$$R[X] := \sum_{t=1}^{T} f_t(\tilde{x}_t) - f(x^*) \qquad \bar{\eta}_T = \frac{\eta^2 L^4 (\ln T + 1)}{T} = o(T)$$

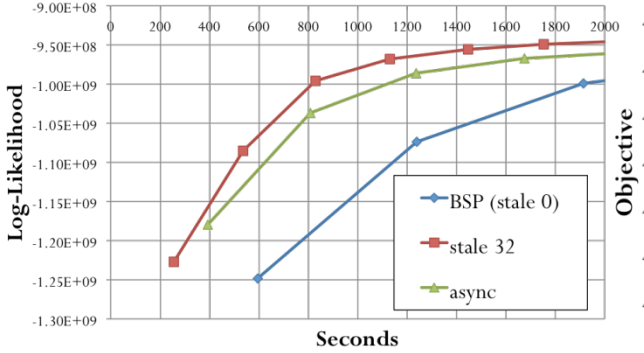
Explanation: the distance between true optima and current estimate decreases exponentially with more SSP iterations.
Lower staleness mean, variance μ_γ , σ_γ improve the convergence rate.



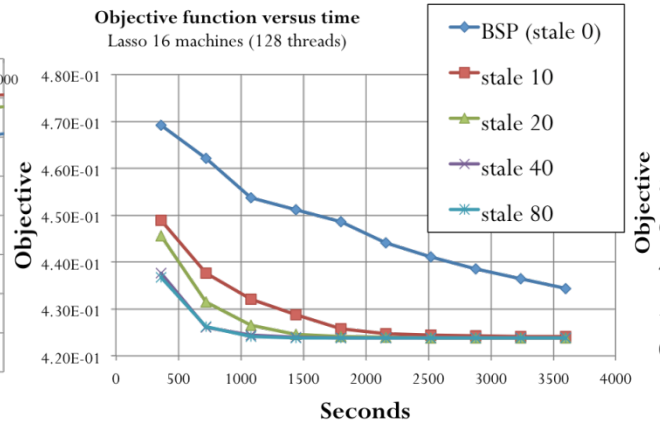
Empirical verification of generalizability

- Massive Data Parallelism
- Effective across different algorithms

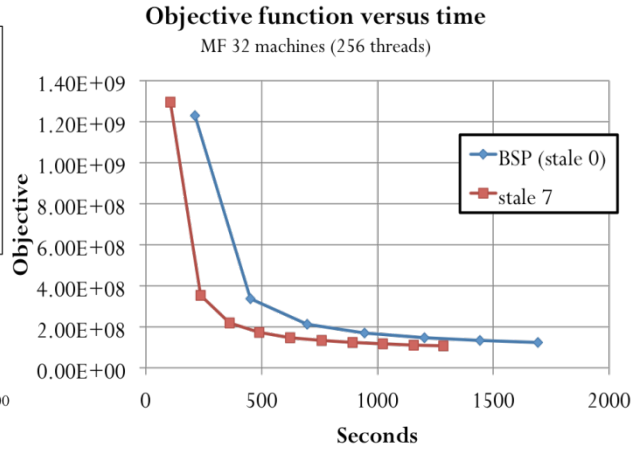
LDA



Lasso

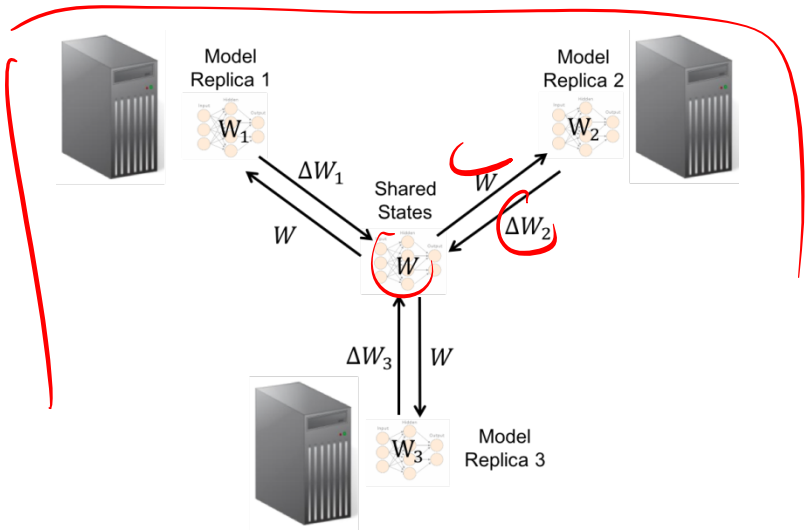


Matrix Fact.

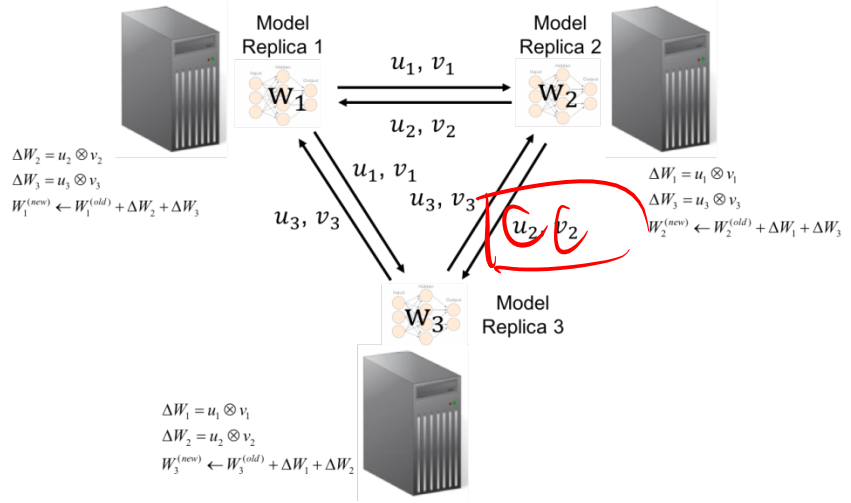


Sufficient Vector Broadcaster vs. Parameter Server

parameter server



Transfer SVs instead of ΔW



- A Cost Comparison

	Size of one message	Number of messages	Network Traffic
P2P SV-Transfer	$O(J+K)$	$O(P^2)$	$O((J+K)P^2)$
Parameter Server	$O(JK)$	$O(P)$	$O(JKP)$

Matrix-parameterized models (MPMs)

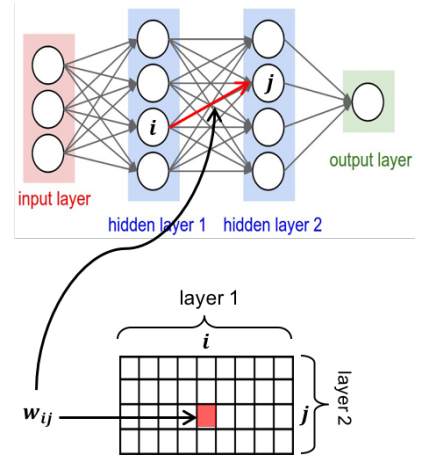
Matrix parameter W

$$\min_{W} \frac{1}{N}\sum_{i=1}^{N} f_i(Wa_i; b_i) + h(W)$$

Loss function

Regularizer

Distance Metric Learning, Topic Models, Sparse Coding, Group Lasso, Neural Network, etc.

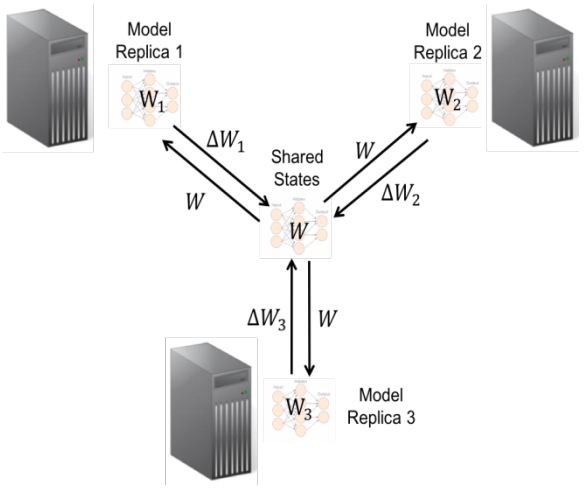


Full updates

- Let matrix parameters be W. Need to send parallel worker updates ΔW to other machines…
  - Primal stochastic gradient descent (SGD)

$$\min_W \frac{1}{N}\sum_{i=1}^{N} f_i(Wa_i;b_i)+h(W)$$

$$\Delta W = \frac{\partial f(Wa_i,b_i)}{\partial W}$$



  - Stochastic dual coordinate ascent (SDCA)

$$\min_Z \frac{1}{N}\sum_{i=1}^{N} f_i^*(-z_i)+h^*\left(\frac{1}{N}ZA^{\mathrm{T}}\right)$$

$$\Delta W = (\Delta z_i)a_i$$

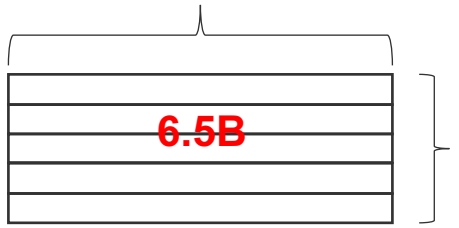
Big MPMs

Billions of params = 10-100 GBs, costly network synchronization

What do we actually need to communicate?

Multiclass Logistic Regression on Wikipedia

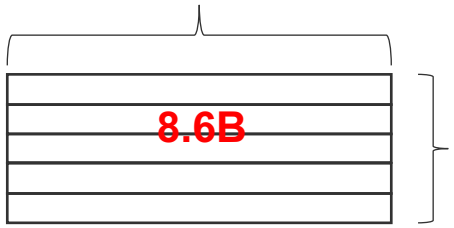
Feature dim. = 20K



#classes=325K

Distance Metric Learning on ImageNet

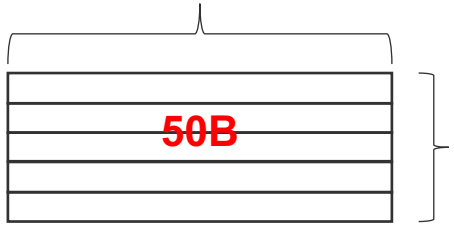
Feature dim. = 172K



Latent dim. = 50K

Topic Model on WWW

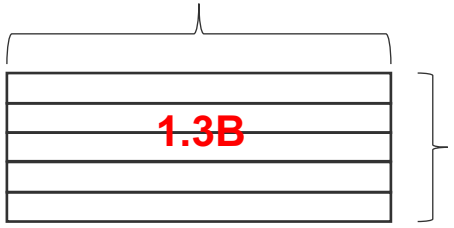
Feature dim. = 1M



Dic. Size= 1M

Neural Network of Google Brain

#neurons in layer 0 = 40K



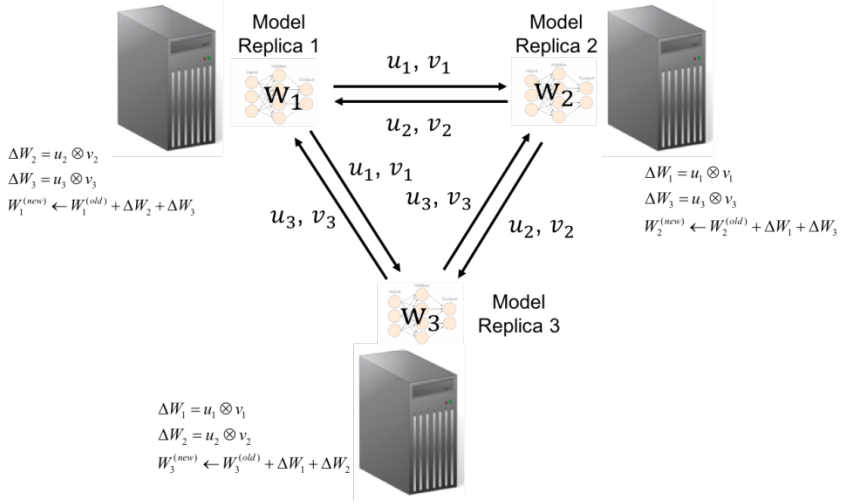
#neurons in layer 1 = 33K

Pre-update: the sufficient vectors [Xie et al., UAI 2015]

- Full parameter matrix update ΔW can be computed as outer product of two vectors uv^T -- the sufficient vectors (SV)
  - Primal stochastic gradient descent (SGD)

$$\min_W \frac{1}{N}\sum_{i=1}^{N} f_i(Wa_i;b_i)+h(W)$$

$$\Delta W = uv^{\mathrm{T}} \quad u=\frac{\partial f(Wa_i,b_i)}{\partial(Wa_i)} \quad v=a_i$$



  - Stochastic dual coordinate ascent (SDCA)

$$\min_Z \frac{1}{N}\sum_{i=1}^{N} f_i^*(-z_i)+h^*\left(\frac{1}{N}ZA^{\mathrm{T}}\right)$$

$$\Delta W = uv^{\mathrm{T}} \quad u=\Delta z_i \quad v=a_i$$

Convergence guarantee

Theorem 1. *Let Assumption 1 hold, and let* $\{\mathbf{W}_p^c\}$, $p = 1, \ldots, P$, $\{\mathbf{W}^c\}$ *be the local sequences and the auxiliary sequence, respectively.*

Under full broadcasting (i.e., $Q = P - 1$*) and set the learning rate* $\eta := \eta_c = O(\sqrt{\frac{1}{L\sigma^2 Psc}})$*, we have*

- $\liminf_{c\to\infty} \mathbb{E}\|\nabla F(\mathbf{W}^c)\| = 0$*, hence there exists a subsequence of* $\nabla F(\mathbf{W}^c)$ *that almost surely vanishes;*
- $\lim_{c\to\infty} \max_p \|\mathbf{W}^c - \mathbf{W}_p^c\| = 0$*, i.e., the maximal disagreement between all local sequences and the auxiliary sequence converges to 0 (almost surely);*
- *There exists a common subsequence of* $\{\mathbf{W}_p^c\}$ *and* $\{\mathbf{W}^c\}$ *that converges almost surely to a stationary point of* F*, with the rate* $\min_{c\le C} \mathbb{E}\|\sum_{p=1}^P \nabla F_p(\mathbf{W}_p^c)\|_2^2 \le O\left(\sqrt{\frac{L\sigma^2 Ps}{C}}\right)$

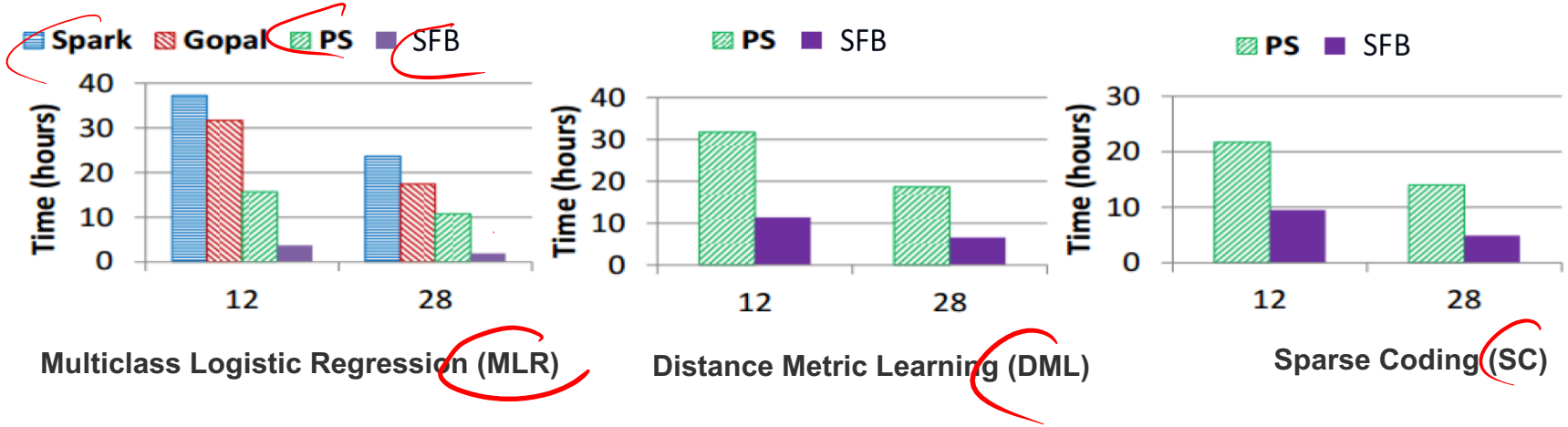
Under partial broadcasting (i.e., $Q < P - 1$*) and set a constant learning rate* $\eta = \frac{1}{CLG(P-Q)}$*, where* C *is the total number of iterations. Then we have*

$$\min_{c\le C} \mathbb{E}\left[\|\sum_{p=1}^P \nabla F_p(\mathbf{W}_p^c)\|_2^2\right] \le O\left(LG(P-Q) + \frac{P(sG+\sigma^2)}{CG(P-Q)}\right).$$

Hence, the algorithm converges to a $O(LG(P-Q))$ *neighbourhood if* $C \to \infty$.

Under partial broadcasting, the algorithm converges to a $O(LG(P-Q))$ neighborhood if $C \longrightarrow \infty$.

Convergence Speedup



- 3 Benchmark ML Programs
  - Big parameter matrices with 6.5-8.6b entries (30+GB), running on 12- & 28-machine clusters
- 28-machine SFB finished in 2-7 hours
  - Up to 5.6x faster than 28-machine PS, 12.3x faster than 28-machine Spark
- PS cannot support SF communication, which requires decentralized storage

Xie et al., UAI 2015

Hybrid Updates: PS + SFB

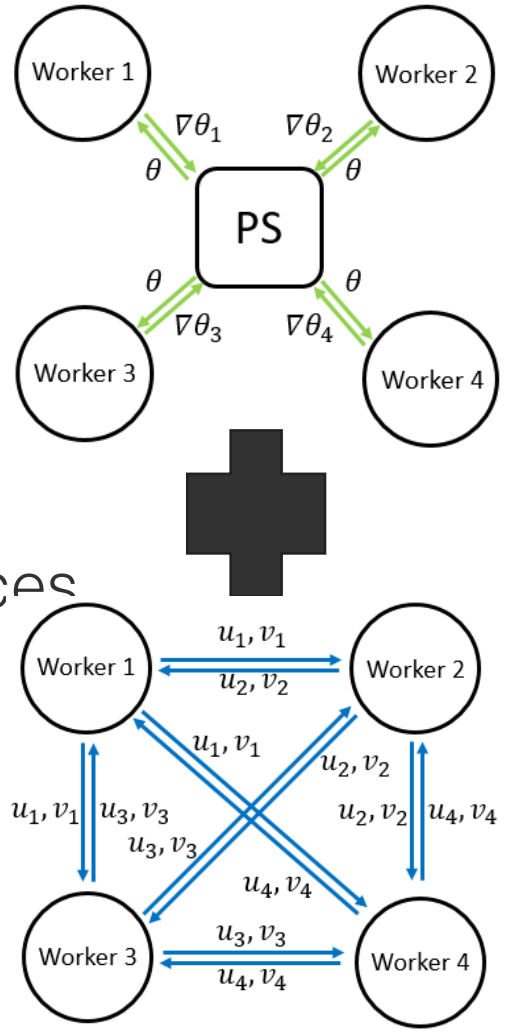
- Hybrid communications:

Parameter Server +

Sufficient Factor Broadcasting

  - Parameter Server: Master-Slave topology
  - Sufficient factor broadcasting: P2P topology

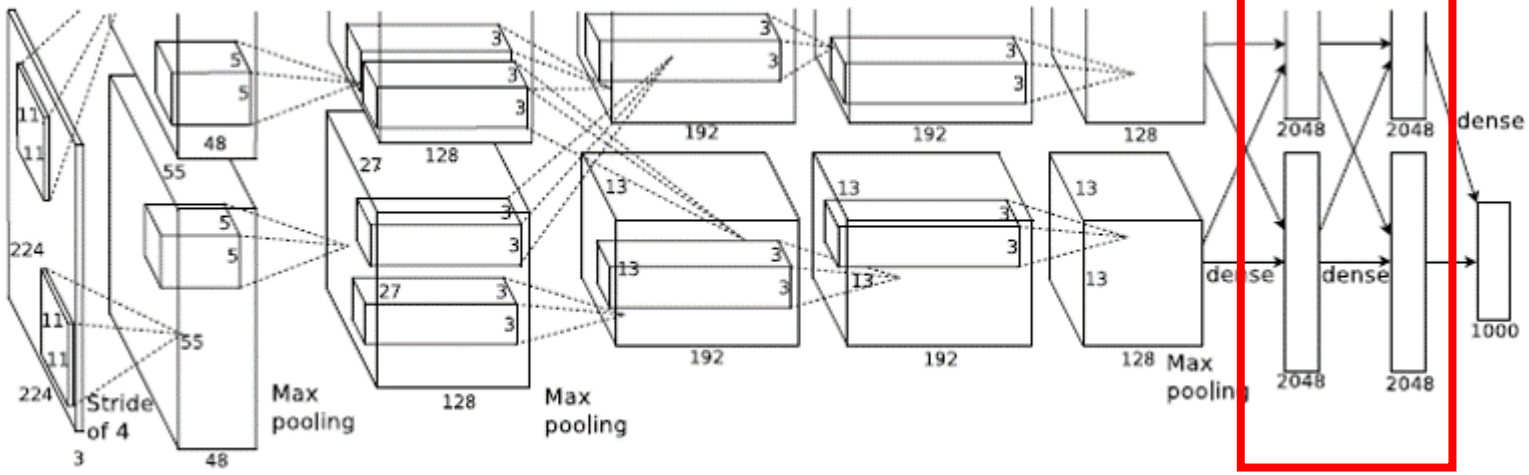
- For problems with a mix of large and small matrices
  - Send small matrices via PS
  - Send large matrices via SFB



Zhang et al., 2015, Zhang et al. 2017

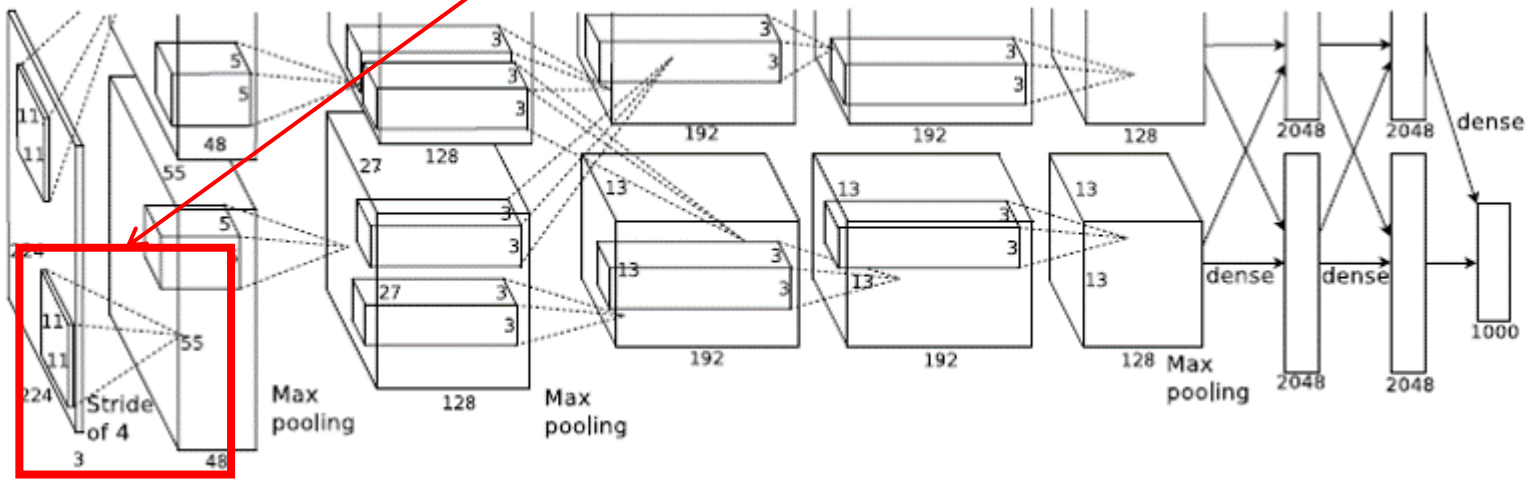
Hybrid example: CNN [Zhang et al., 2015]

- Example: AlexNet CNN model
  - Final layers = 4096 * 4096 matrix (17M parameters)
  - Use SFB to communicate
    - 1. Decouple into two 4096 vectors: u, v
    - 2. Transmit two vectors
    - 3. Reconstruct the gradient matrix



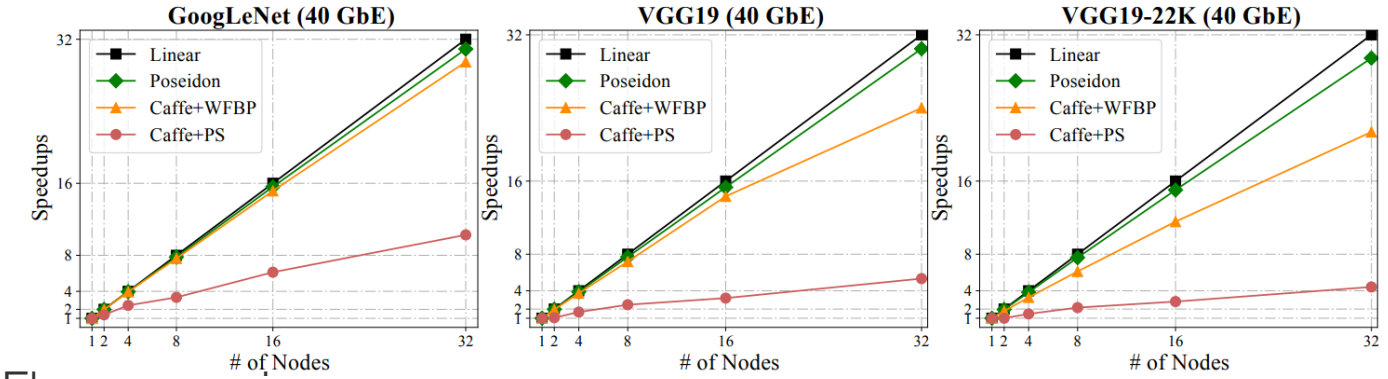
Hybrid example: CNN [Zhang et al., 2015]

- Example: AlexNet CNN model
  - Convolutional layers = e.g. 11 * 11 matrix (121 parameters)
  - Use Full-matrix updates to communicate
    - 1. Send/receive using Master-Slave PS topology

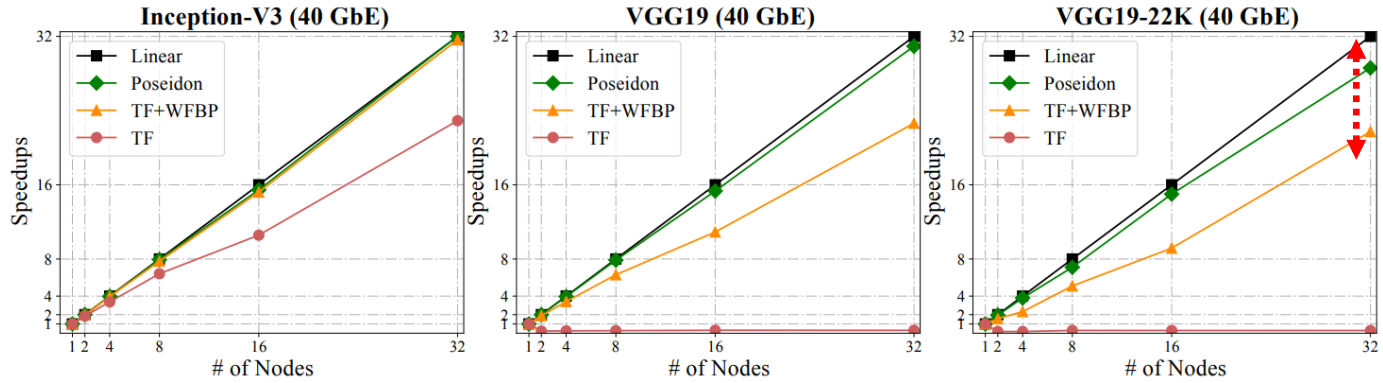


Hybrid Communication

- Results: achieve linear scalability across different models/data with 40GbE bandwidth
  - Using Caffe as an engine:



  - Using TensorFlow as engine



Improve over WFBP

Zhang et al., 2015, Zhang et al. 2017

PETUUM

Programming Interface and Software Frameworks

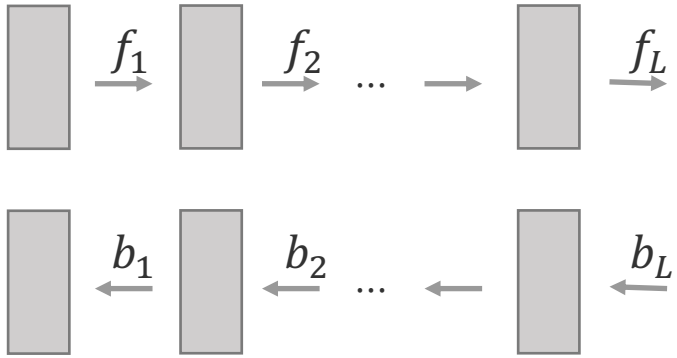
From TensorFlow

to DyNet (Neubig et al, NIPS 2016)

to Cavs (Zhang et al, SoCC 2018)

Deep Learning as Dataflow Graphs

- Gradient Descent via Backpropagation
  - Gradients can be computed by auto differentiation
  - Automatically derive the gradient flow graph from the forward dataflow graph



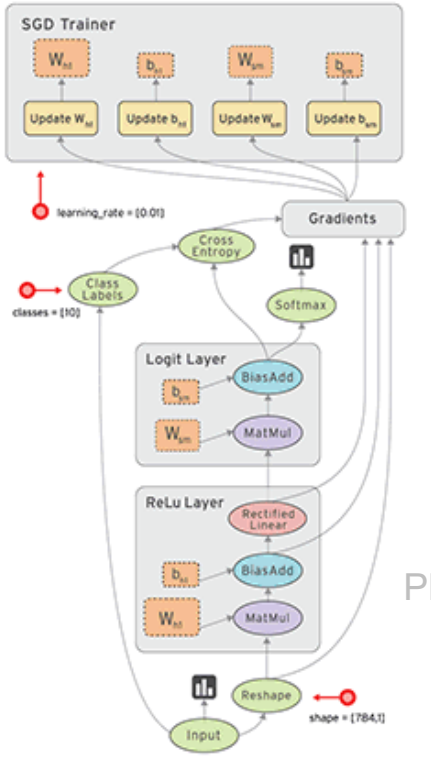
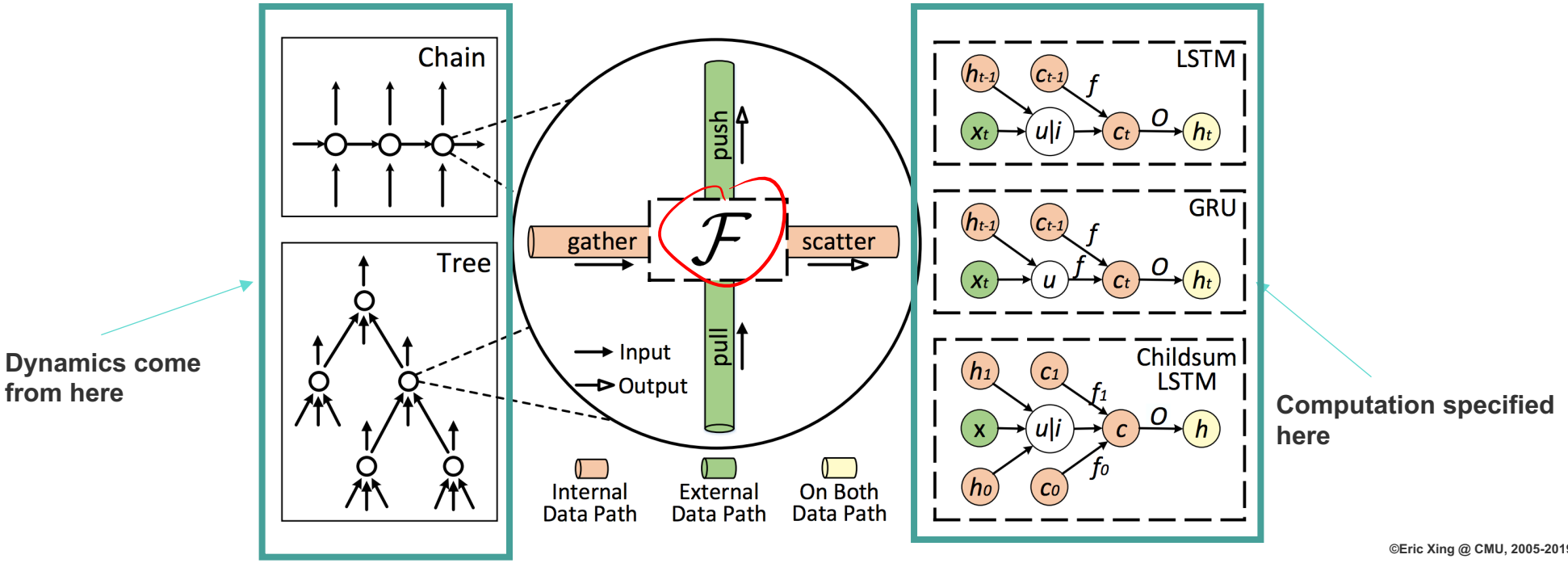


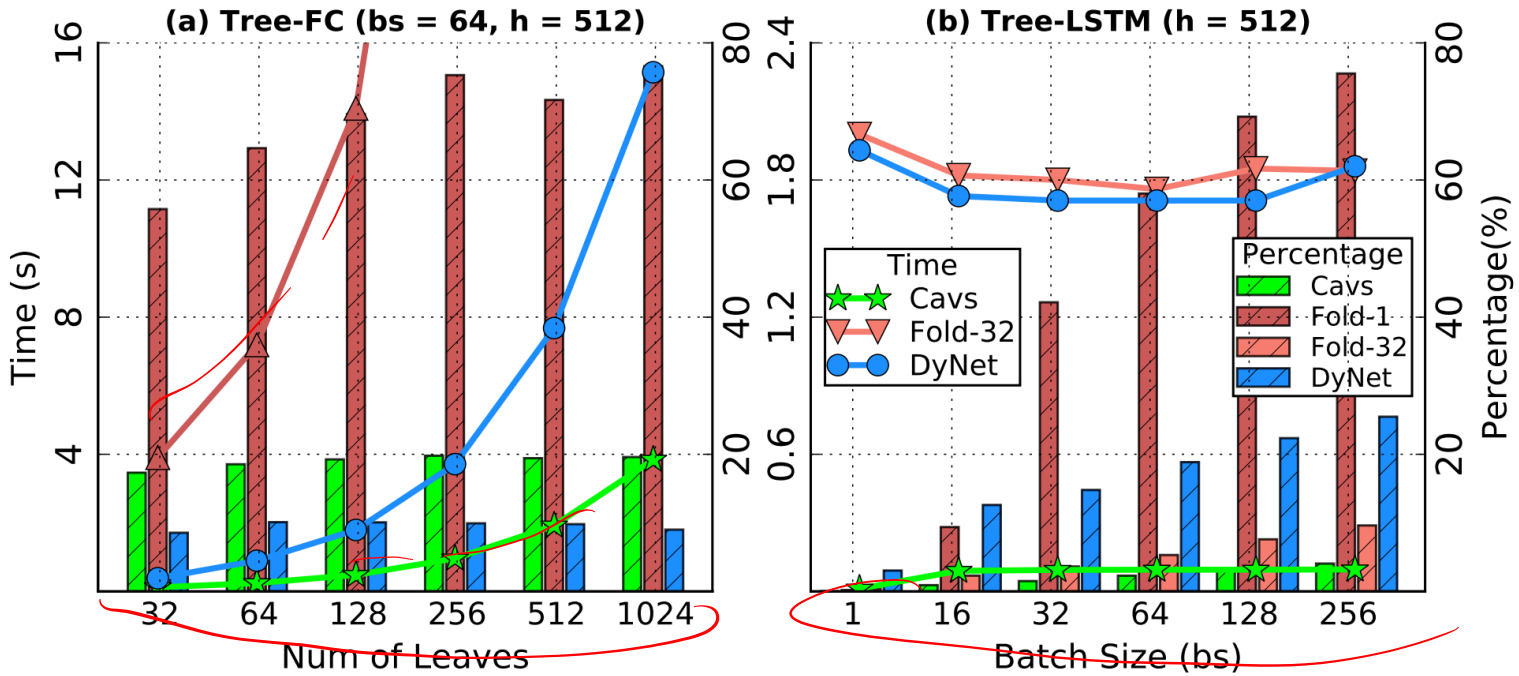
Photo from TensorFlow website

Cavs (Petuum): DL as a Vertex Program (Zhang et al, SoCC 2018)

- Key idea: separate out static ML model declaration from the data-dependent dynamics of input samples
- Vertex-centric representation for DL, decompose a dynamic NN as two modules
  - A vertex function F, which is static;
  - An input graph G, which is data-dependent and dynamic;

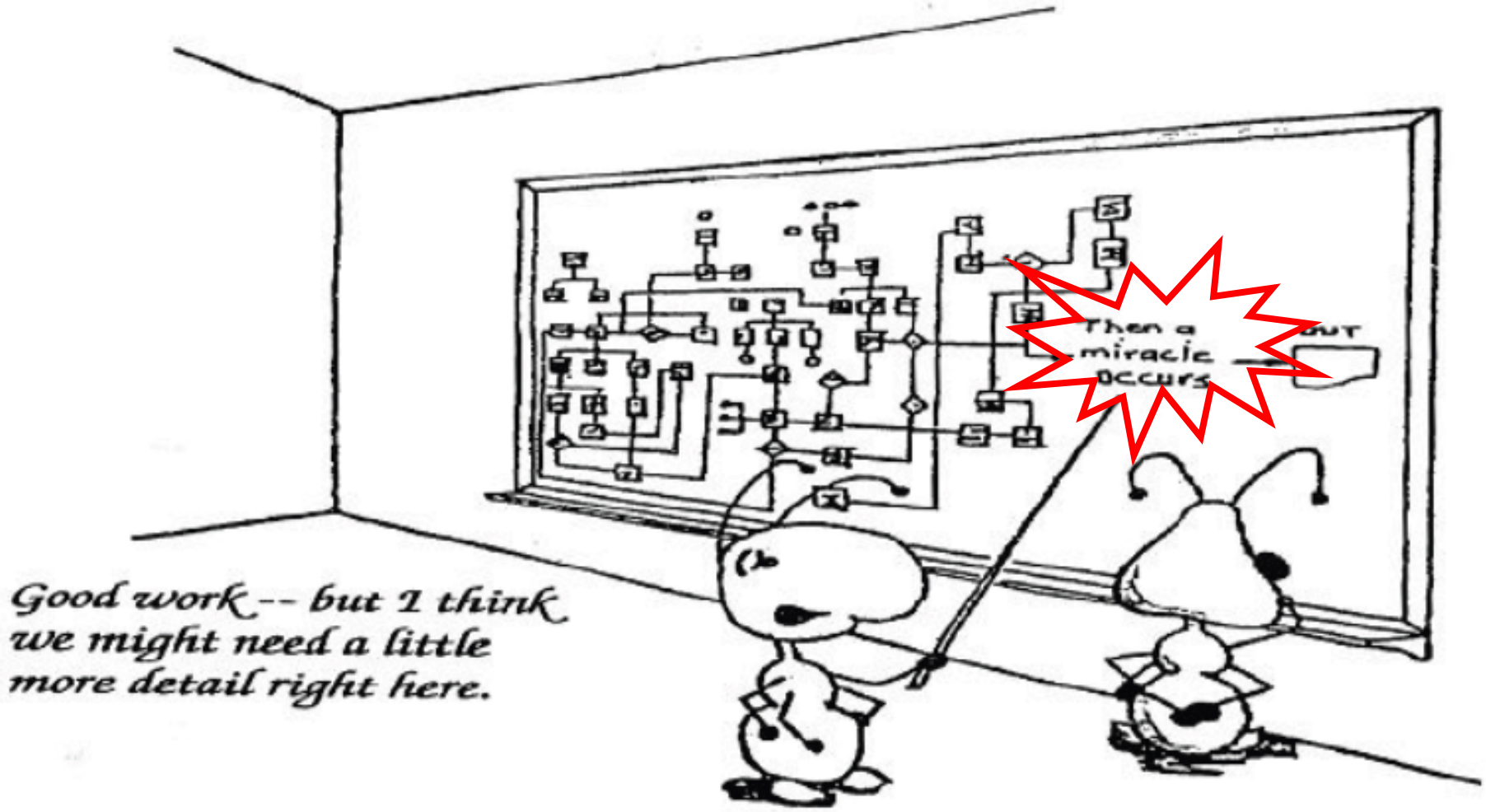


Vertex-Program versus Data-Flow

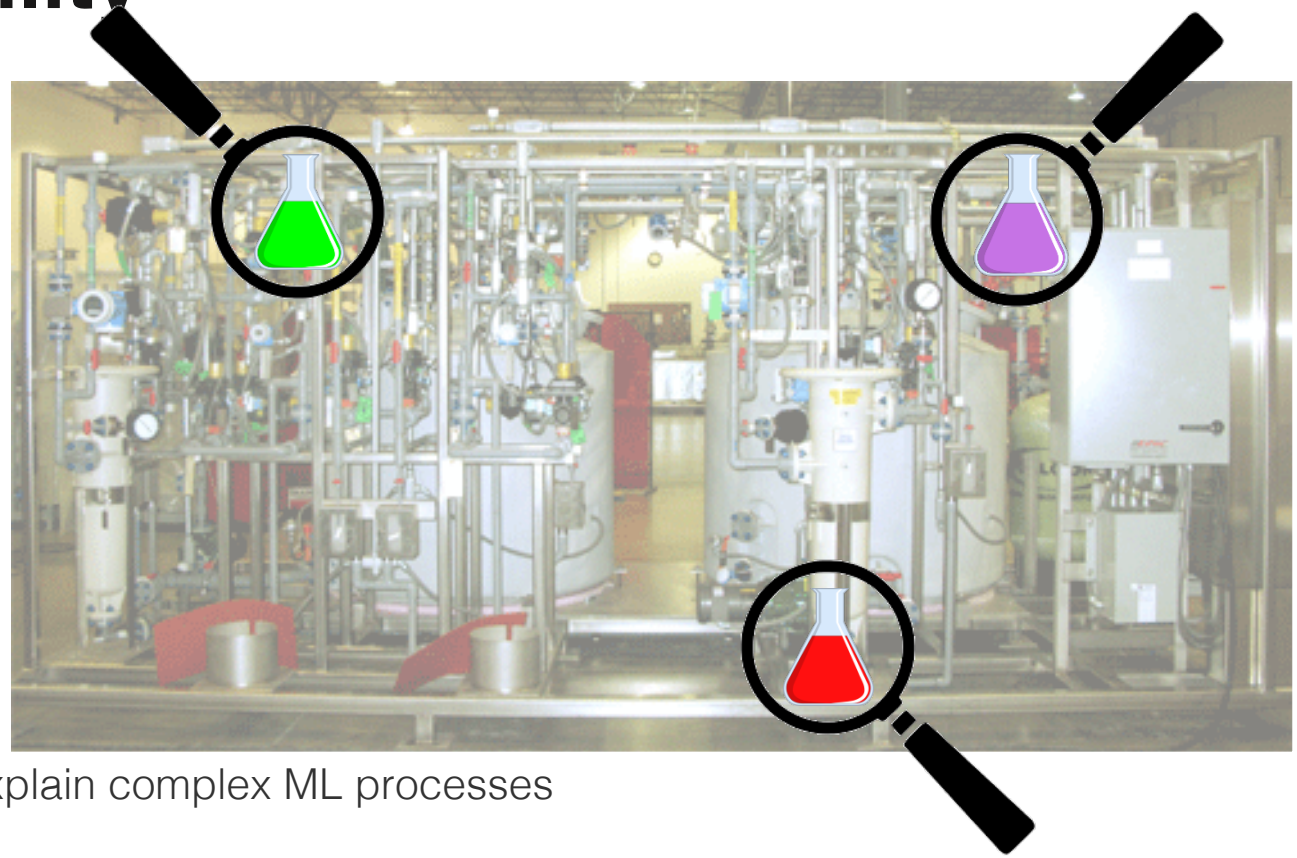


- Graph construction literally takes 80% of time in TensorFlow Fold
- Curve (left axis): absolute time; bar (right): percentage time

On Explainable AI



Explainability/Interpretability



- Data explainability
  - How the data is pre-processed
- Model explainability
  - What you've learned, e.g., feature weights
- Inference explainability
  - How each result is inferred
- Process explainability
  - Factors beyond or complementary to the mechanisms/mathematics of ML

- Post-hoc reason codes may not be sufficient to explain complex ML processes
- "Factory Inspection" (X-ray)
  - Reveal entire ML process – not just code
  - Interactive visualization of every stage in the "AI-build" process
  - Inject data and show how it is turned into results, and vice versa

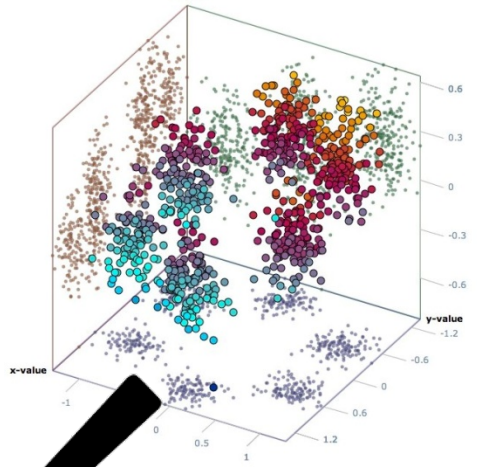
Data explainability

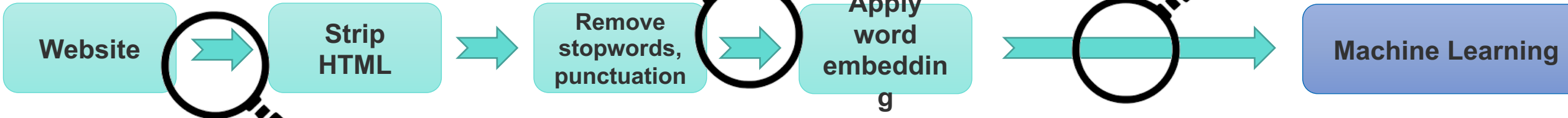
- Example: how is HTML transformed into an “ML-ready” form?
  - Strip HTML tags
  - Remove stopwords, punctuation
  - Apply word embedding (LDA, neural network, etc.)
- Interactively visualize input and output of every stage

Word Cloud


```
bbc culture every story world one …
story story story …
researchers analyzed …
```

Point Cloud

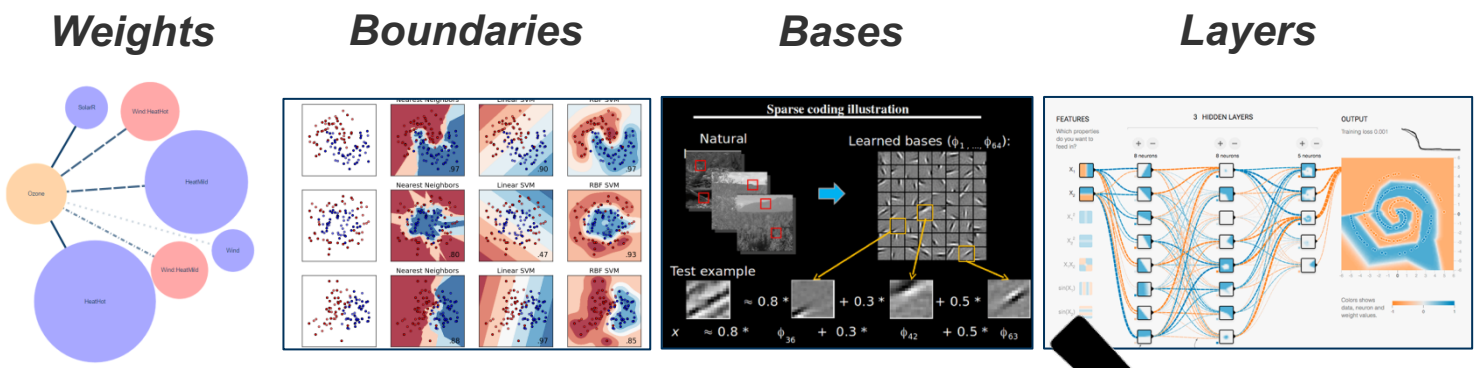





```
<!DOCTYPE html><html lang="en"><head><meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <title>BBC - Culture - Every story in the world has one …
        <meta name="keywords" content="story, STORY, story, …
        <meta name="description" content="Researchers analysed …
">
```

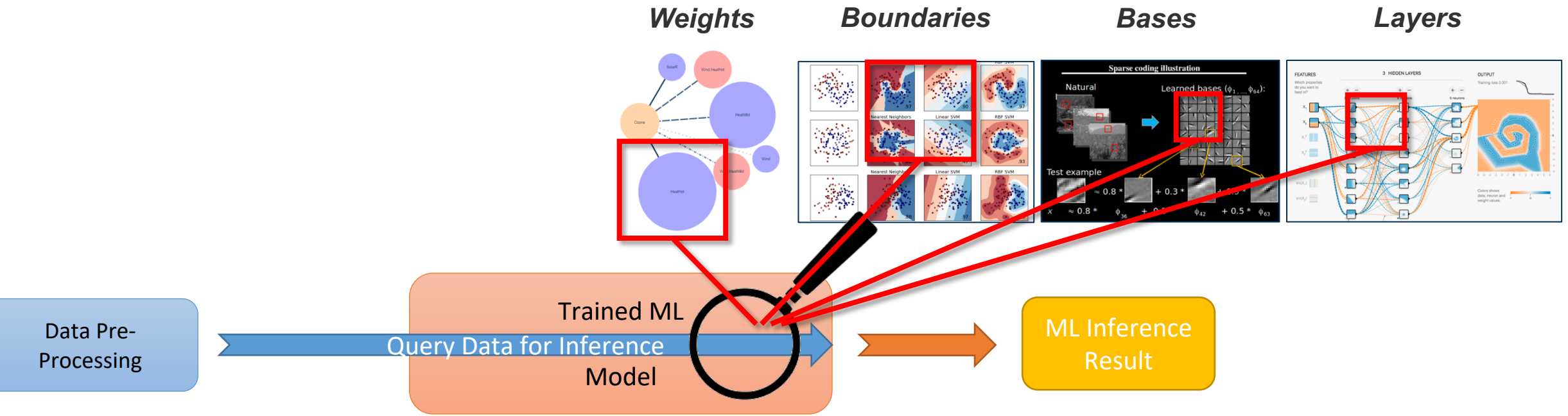
Model explainability

- How can we understand the ML model?
  - Weights – which input features are most important to the result?
  - Boundaries – which input values cause the result to change qualitatively?
  - Bases – what are the concepts, examples that are important to the model?
  - Layers – for hierarchical models (Bayesian, Deep Learning), visualize layer weights/boundaries/bases



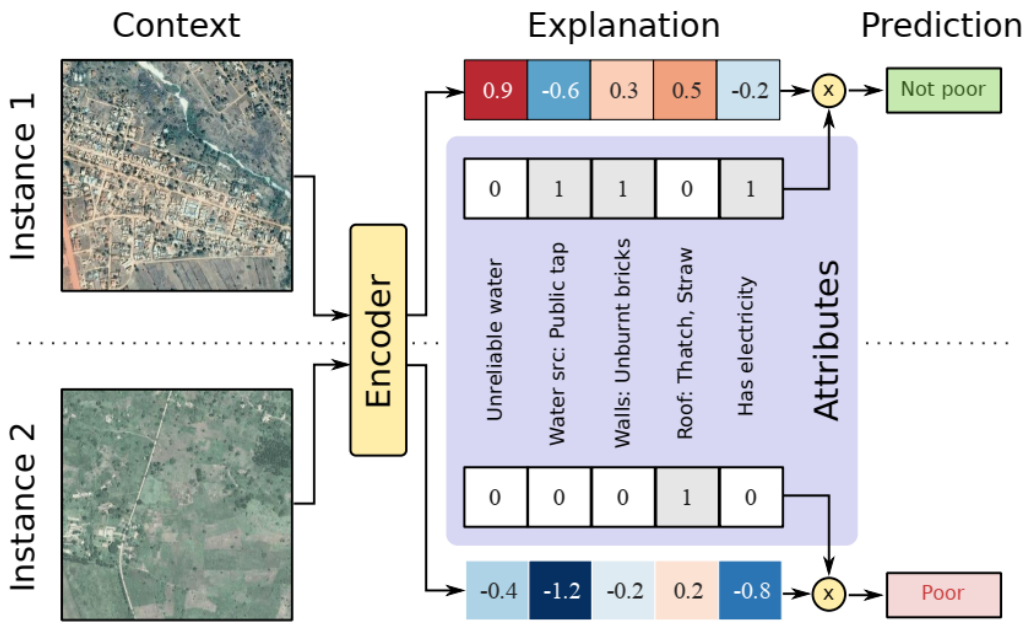
Inference explainability 1/2

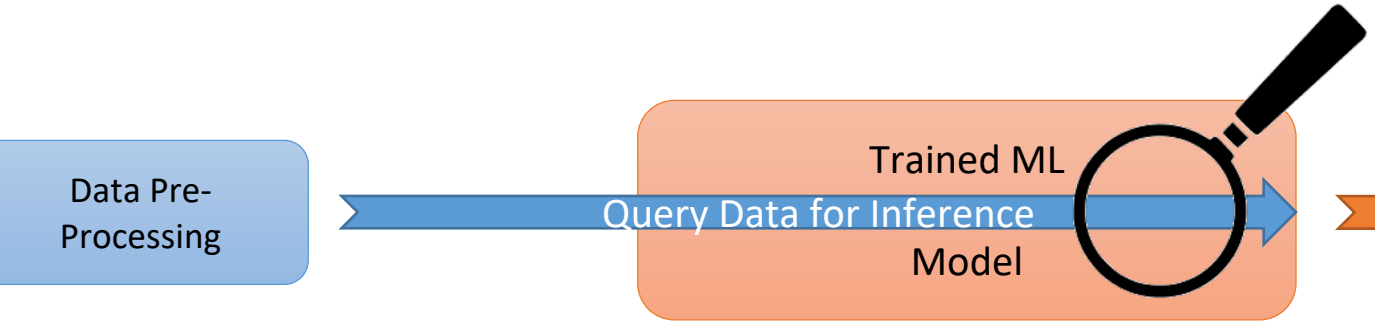
- Understand what happens as data travels through an ML model
  - Weights – which weights are activated by the data?
  - Boundaries – what decision boundaries are close to the data?
  - Bases – what bases are activated by the data?
  - Layers – visualize layer weights/boundaries/bases activation by the data



Inference explainability 2/2

- Understand what happens as data travels through an ML model
  - Incorporate explanations or reason codes directly into the model – Contextual Explanation Networks
  - ML result directly generated from explanations
  - **A "personalized" model for every datum**

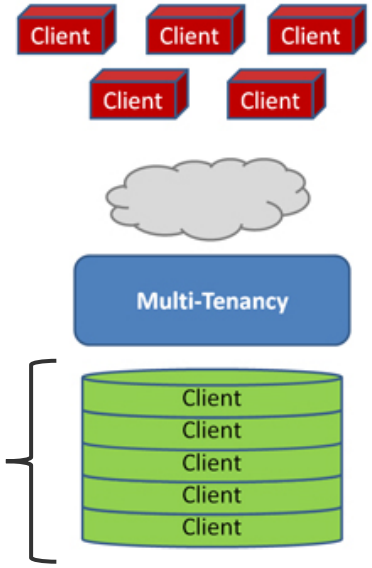




Process explainability

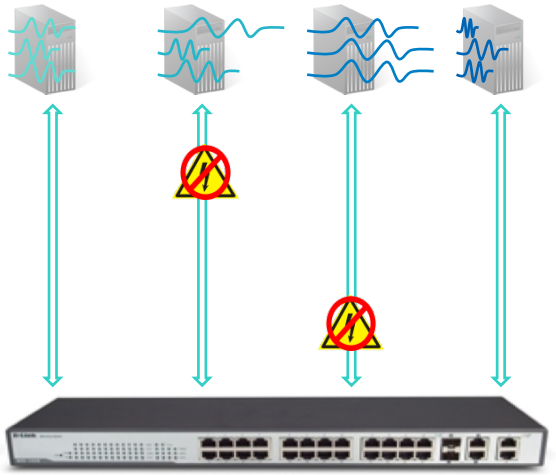
- How do factors beyond Data, Model, Inference affect ML results?
  - Multiple users/tenants sharing the cloud or datacenter resources?
  - One of the computers crashes or becomes slow?
  - Computer network becomes unreliable or slow?
  - Security breaches? Data outage?
- Above are typical in multi-tenant environments – cloud and corporate network/cluster

Different clients/tenants



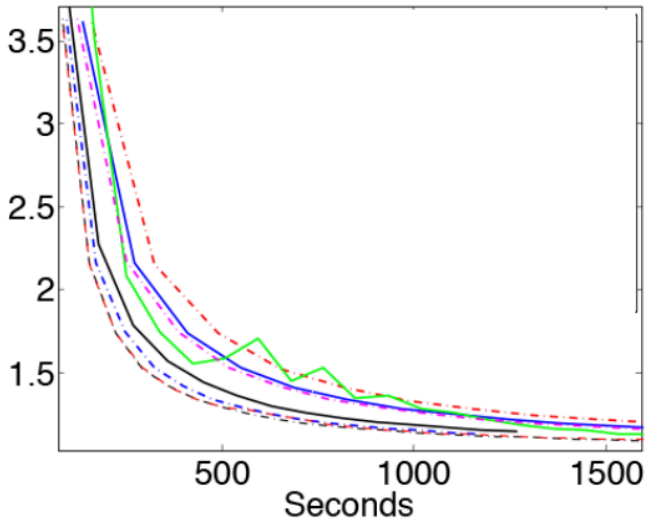
Sharing same computing resources

Computer performance unequal



Network slow, unreliable

Different network, CPU/GPU conditions = different results?



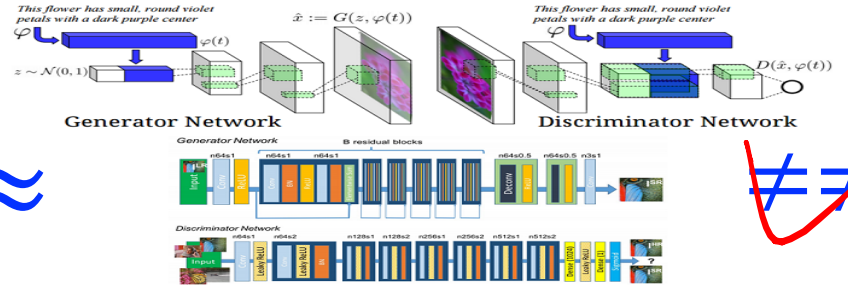
PETUUM

Summary

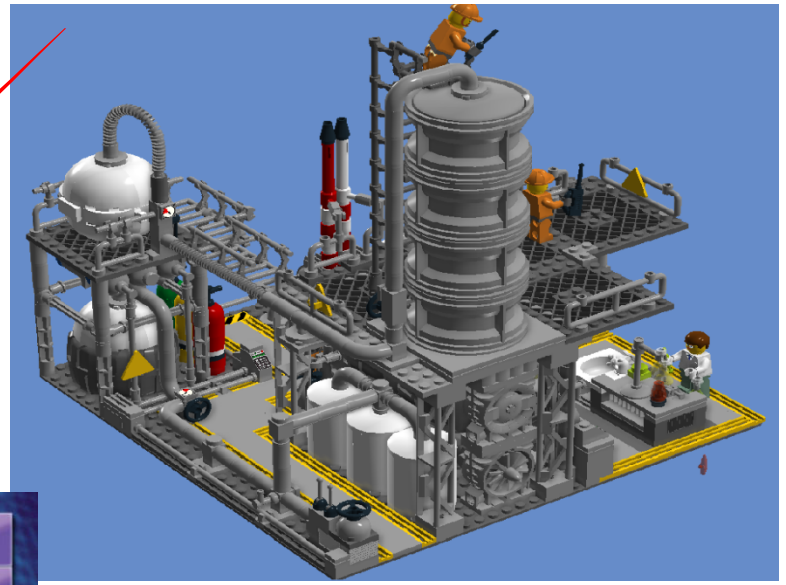
AI as of now: still in medieval age

- Alchemy vs. chemistry vs. chemical engineering

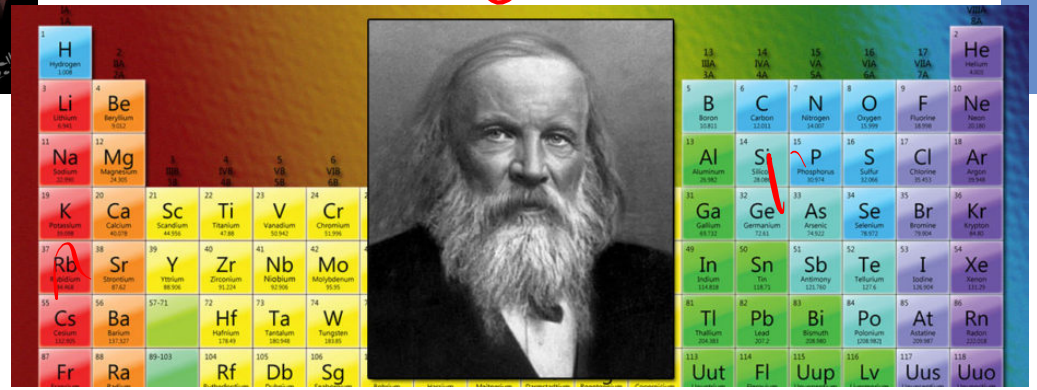
≈



≠≠



≠



AI-Build requires a sound scientific & engineering process
--- not just model/algorithm fiddling

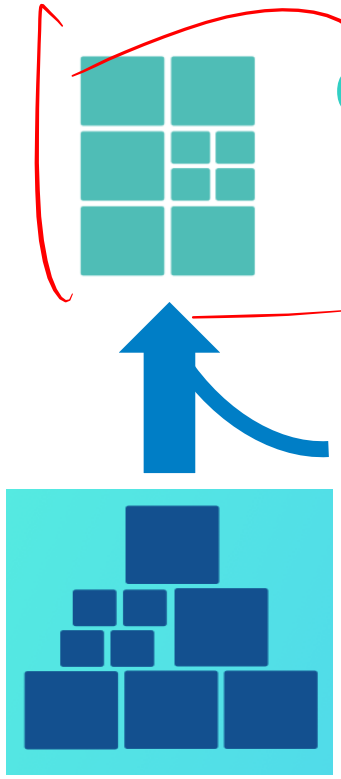
- First Principles
- The “Civil” Engineering
- Explain the process and outcome
- Analysis and safety under real deployment and operation
- Standardize, mass production, cost amortization

AI as "Civil Engineering"

Industry Agnostic

Commercialized to date:
Industrial & Healthcare only

Building AI Like Lego

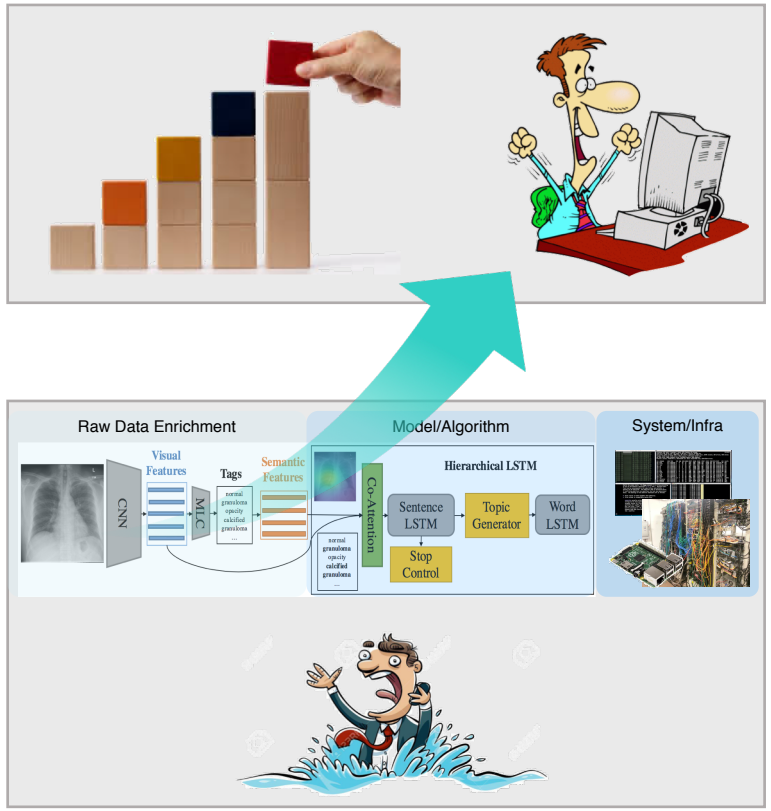


Completed software

10%
White-Glove Assembly

90%
Completed building blocks

AI With No Tears



Petuum OS

An inventory of ML/Sys nuts and bolts

Data Machine	Ingestion	Cleaning & Improvement	Feature Engineering	Embedding	Integration	Interpret (DM X-Ray)	Dev	Resource
Machine Learning	Build	Augment and Tune	Train	Score	Experiment	Interpret (Model X-ray)	Dev	Resource
Operating System	Deploy	Storage	Compatibility	User Accounts and Security	Project Management	Interpret (Process X-ray)	Dev	Resource