SCIENCE
PASSION
TECHNOLOGY

Architecture of ML Systems
10 Model Selection & Management

Matthias Boehm

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMVIT endowed chair for Data Management

Last update: June 05, 2020

ISDS

Announcements/Org

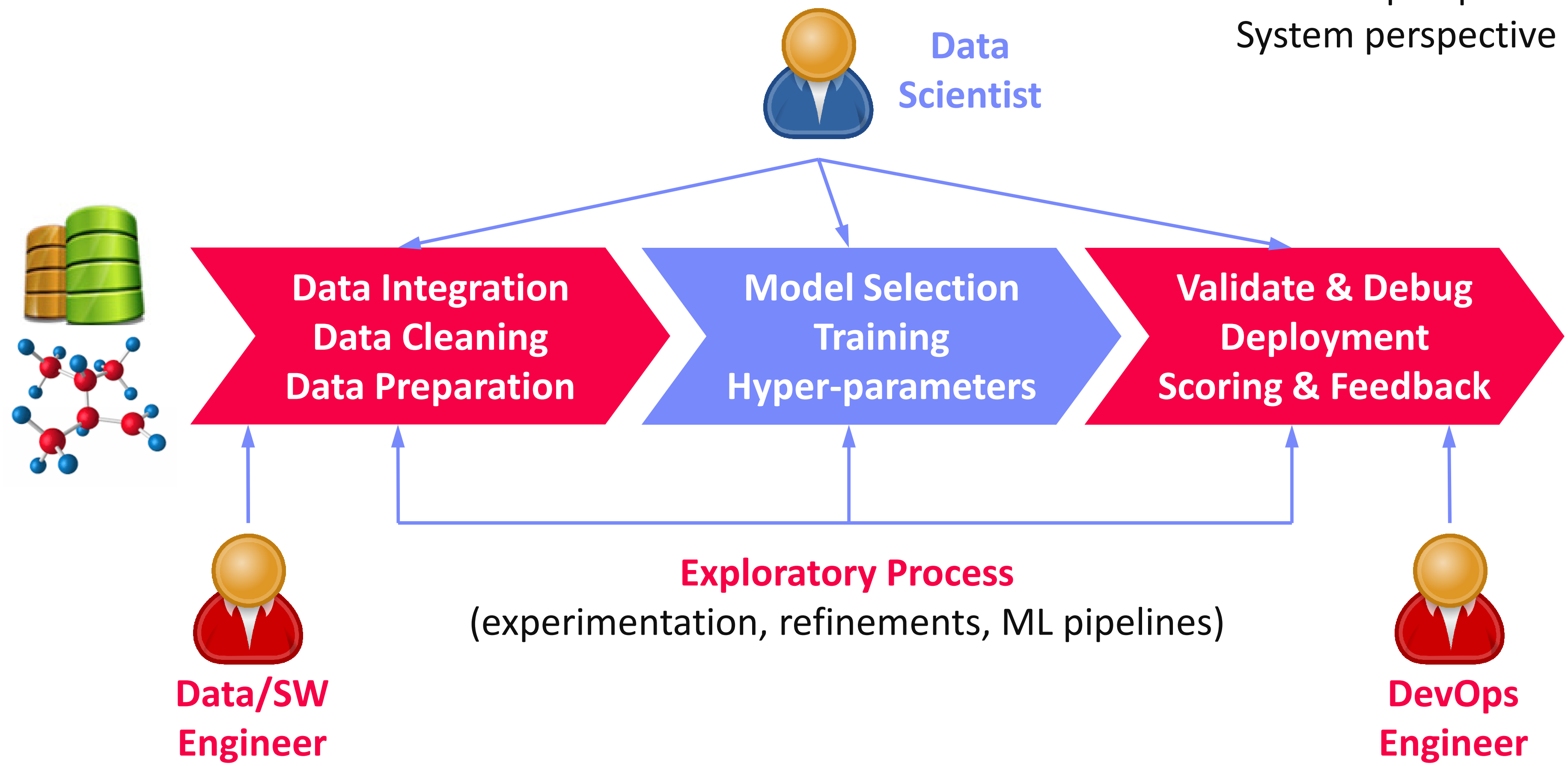
- **#1 Video Recording**
  - Link in **TeachCenter** & **TUbe** (lectures will be public)
  - **Live streaming through TUbe**, starting May 08
  - Questions: https://tugraz.webex.com/meet/m.boehm

- **#2 AMLS Programming Projects**
  - **Status:** all project discussions w/ **15 students** (~**8 PRs**)
  - Awesome mix of projects (algorithms, compiler, runtime)
  - Soft deadline: **June 30**
  - If unable to complete: email to m.boehm@tugraz.at

- **#3 Course Evaluation**
  - Please participate; open period: **June 1 – July 15**

Recap: The Data Science Lifecycle

Data-centric View:
Application perspective
Workload perspective
System perspective



Agenda

- **Data Augmentation**
- **Model Selection Techniques**
- **Model Management**

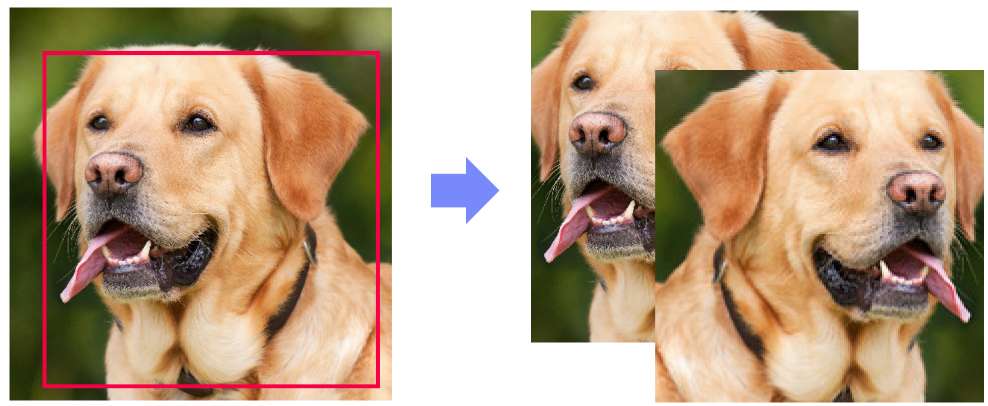
Data Augmentation

Motivation and Basic Data Augmentation

AlexNet

[Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks. **NIPS 2012**]

- **Motivation Data Augmentation**
  - Complex ML models / deep NNs need lots of labeled data to avoid overfitting → **expensive**
  - Augment training data by synthetic labeled data
- **Translations & Reflections**
  - Random 224x224 patches and their reflections (from 256x256 images with **known labels**)
  - Increased data by **2048x**
  - Test: corner/center patches + reflections → prediction



- **Alternating Intensities**
  - **Intuition:** object identity is invariant to illumination and color intensity
  - PCA on dataset → add eigenvalues times a random variable N(0,0.1)

Basic Data Augmentation

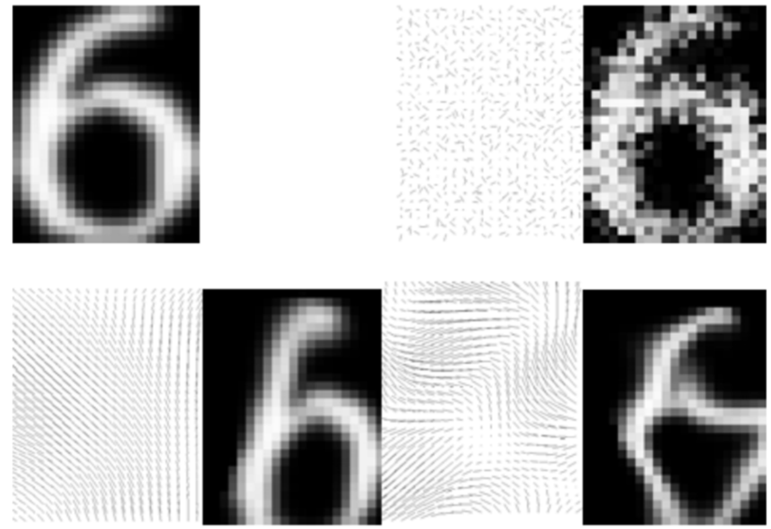
- **Scaling and Normalization**
  - Standardization: subtract per-channel global pixel means
  - Normalization: normalized to range [-1,1] (see min-max)
- **General Principles**
  - **#1:** **Movement/selection** (translation, rotation, reflection, cropping)
  - **#2:** **Distortions** (stretching, shearing, lens distortions, color)
  - In many different combinations ➔ often trial & error / domain expertise
- **Excursus: Reducing Training Time**
  - **Transfer learning:** Use pre-trained model on ImageNet; freeze lower NN layers, fine-tune last layers w/ domain-specific data
  - **Multi-scale learning:** Use cropping and scaling to train 256 x 256 model as starting point for a more compute-intensive 384x384 model

[Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition. **ICLR 2015**]

Basic Data Augmentation, cont.

- **Distortions**
  - Translations, rotations, skewing
  - Compute for every pixel a new target location via rand displacement fields)

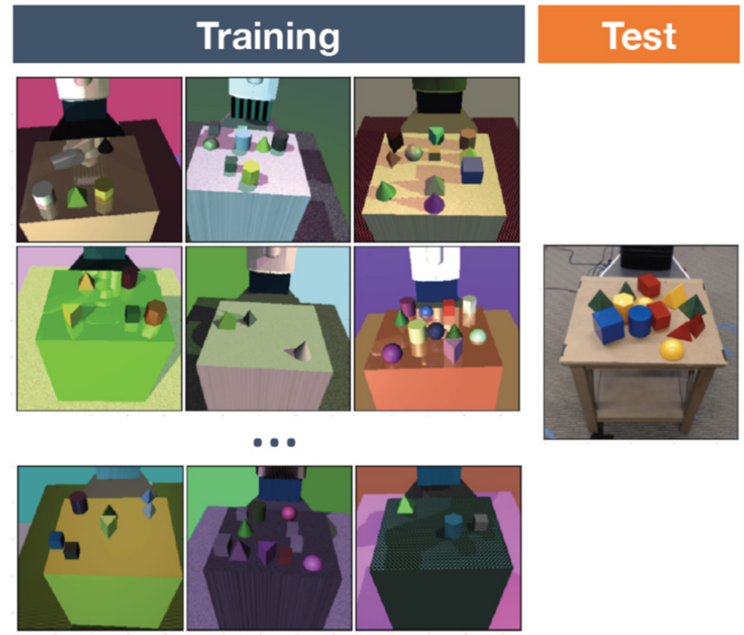
[Patrice Y. Simard, David Steinkraus, John C. Platt: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. **ICDAR 2003**]



- **Cutout**
  - Randomly masking out square regions of input images
  - Size more important than shape

[Terrance Devries, Graham W. Taylor: Improved Regularization of Convolutional Neural Networks with Cutout. **CoRR 2017**]

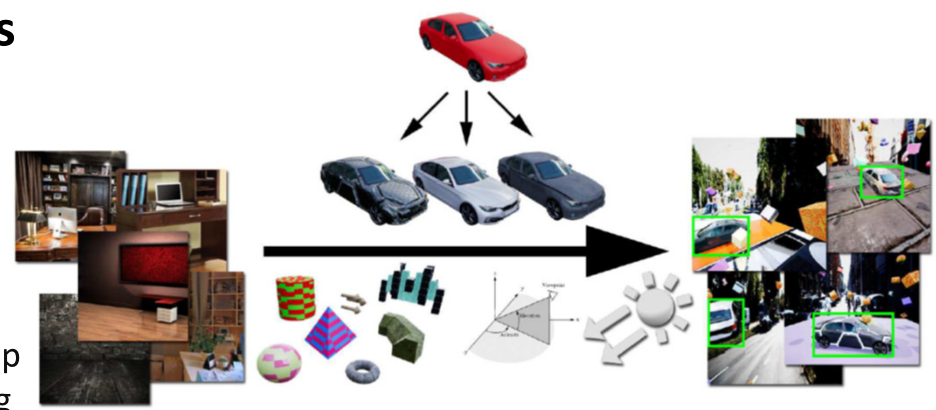
Domain Randomization



- **Training on Simulated Images**
  - Random rendering of objects with non-realistic textures
  - Large variability for generalization to real world objects

[Josh Tobin et al.: Domain randomization for transferring deep neural networks from simulation to the real world. **IROS 2017**]

- **Pre-Training on Simulated Images**
  - Random 3D objects and flying distractors w/ random textures
  - Random lights and rendered onto random background



[Jonathan Tremblay et al.: Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization. **CVPR Workshops 2018**]

Learning Data Augmentation Policies

- **AutoAugment**
  - Search space of augmentation policies
  - Goal: **Find best augmentation policy** (e.g., via reinforcement learning)
  - **#1: Image processing functions** (translation, rotation, color normalization)
  - **#2: Probabilities of applying these functions**

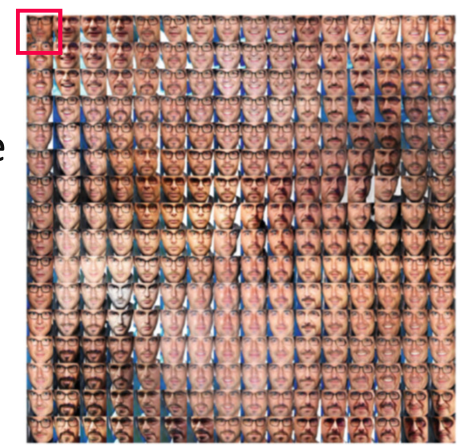
[Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, Quoc V. Le: AutoAugment: Learning Augmentation Policies from Data. **CVPR 2019**]

➔ **New state-of-the art top-1 error** on ImageNet and CIFAR10

- **Data Augmentation GAN (DAGAN)**
  - Image-conditional generative model for creating within-class images from inputs
  - No need for known invariants

[Antreas Antoniou, Amos J. Storkey, Harrison Edwards: Augmenting Image Classifiers Using Data Augmentation Generative Adversarial Networks. **ICANN 2018**]

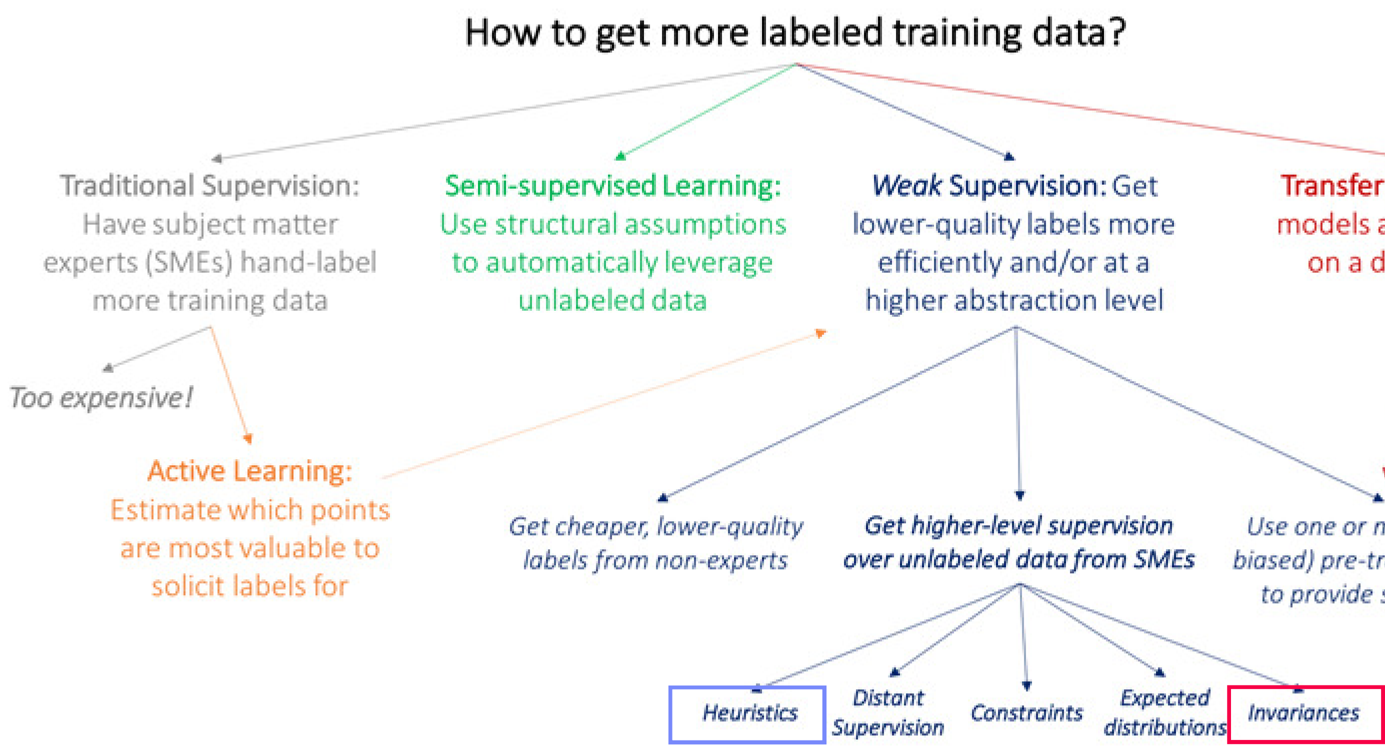
Real input image



Weak Supervision

[Alex Ratner, Paroma Varma, Braden Hancock, Chris Ré, and others: Weak Supervision: A New Programming Paradigm for Machine Learning, ai.stanford.edu/blog/weak-supervision/, 2019]

- **Heuristically Generated Training Data**
  - Hand labeling expensive and time consuming, but abundant unlabeled data
  - Changing labeling guidelines → labeling heuristics

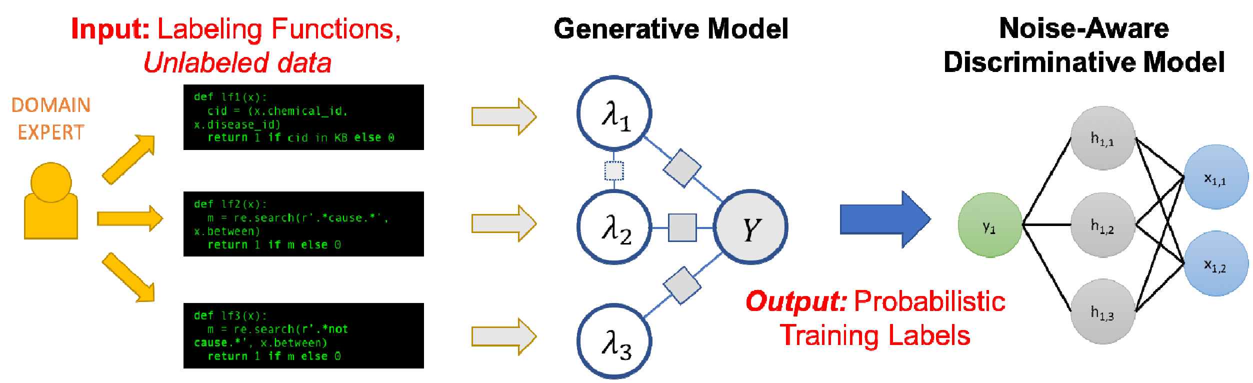


basic data augmentation

Weak Supervision, cont.

(coverage α_i, accuracy β_i)

- **Data Programming Overview**



[Alexander J. Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, Christopher Ré: **Data Programming:** Creating Large Training Sets, Quickly. **NIPS 2016**]

[Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, Christopher Ré: **Snorkel:** Rapid Training Data Creation with Weak Supervision. **PVLDB 2017**]

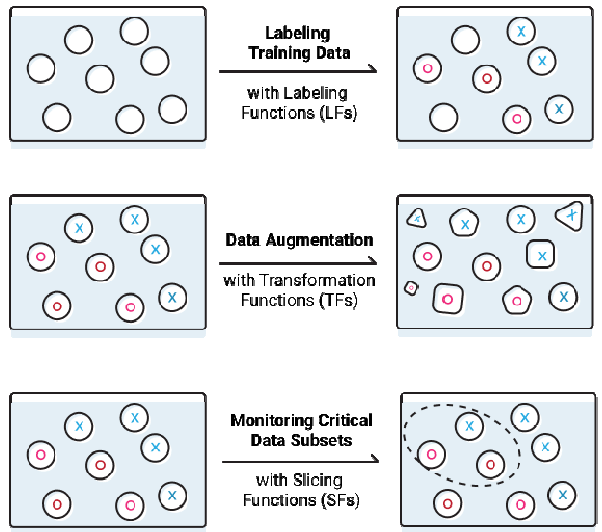
[Paroma Varma, Christopher Ré: **Snuba:** Automating Weak Supervision to Label Training Data. **PVLDB 2018**]

[Stephen H. Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alexander Ratner, Braden Hancock, Houman Alborzi, Rahul Kuchhal, Christopher Ré, Rob Malkin: **Snorkel DryBell:** A Case Study in Deploying Weak Supervision at Industrial Scale. **SIGMOD 2019**]

Weak Supervision, cont.

- **Excursus: Snorkel** [https://www.snorkel.org/]
  - Programmatically Building and Managing Training Data

snorkel



10 Model Selection & Management

11 Model Debugging Techniques

- **Effects of Augmentation**
  - **#1** Regularization for reduced generalization error, not always training error (penalization of model complexity)
  - **#2** Invariance increase by averaging features of augmented data points

  → **Data Augmentation as a Kernel**
  - Kernel metric for augmentation selection
  - Affine transforms on approx. kernel features

[Tri Dao et al: A Kernel Theory of Modern Data Augmentation. **ICML 2019**]

Model Selection Techniques

AutoML Overview

[Chris Thornton, Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. **KDD 2013**]

- **Model Selection**
  - Given a dataset and ML task (e.g., classification or regression)
  - Select the model (type) that performs best (e.g.: LogReg, Naïve Bayes, SVM, Decision Tree, Random Forest, DNN)

$$A^* \in \operatorname*{argmin}_{A \in \mathcal{A}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A, \mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{valid}}),$$

- **Hyper Parameter Tuning**
  - Given a model and dataset, find best hyper parameter values (e.g., learning rate, regularization, kernels, kernel parameters, tree params)

$$A^*{}_{\boldsymbol{\lambda}^*} \in \operatorname*{argmin}_{A^{(j)} \in \mathcal{A}, \boldsymbol{\lambda} \in \boldsymbol{\Lambda}^{(j)}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A^{(j)}_{\boldsymbol{\lambda}}, \mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{valid}}).$$

- **Validation: Generalization Error**
  - Goodness of fit to held-out data (e.g., 80-20 train/test)
  - Cross validation (e.g., leave one out → k=5 runs w/ 80-20 train/test)

→ **AutoML Systems/Services**
  - Often providing both **model selection and hyper parameter search**
  - Integrated ML system, often in distributed/cloud environments

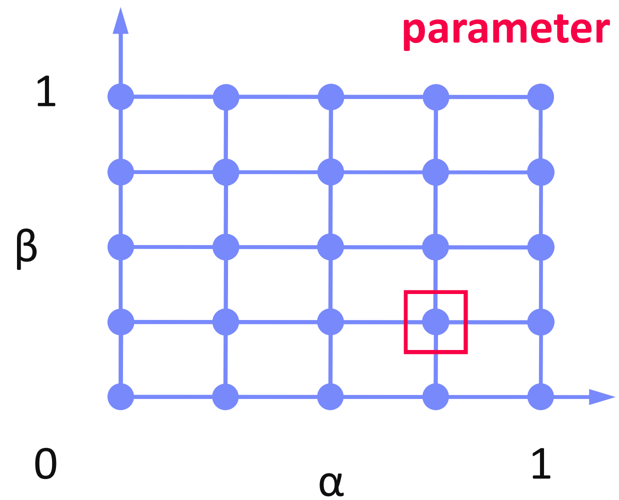
Basic Grid Search

Apache SystemML™ `gridSearch()`

scikit learn `GridSearchCV()`

- **Basic Approach**
  - Given n hyper parameters λ1, …, λn with domains Λ1, …, Λn
  - Enumerate and evaluate parameter space $\Lambda \subseteq \Lambda_1 \times \ldots \times \Lambda_n$ (often strict subset due to dependency structure of parameters)
  - Continuous hyper parameters → discretization
    - Equi-width
    - Exponential (e.g., regularization 0.1, 0.01, 0.001, etc)
  - **Problem:** Only applicable with small domains
- **Heuristic: Monte-Carlo (random search)**

Non-convex or unknown parameter space



Basic Iterative Algorithms

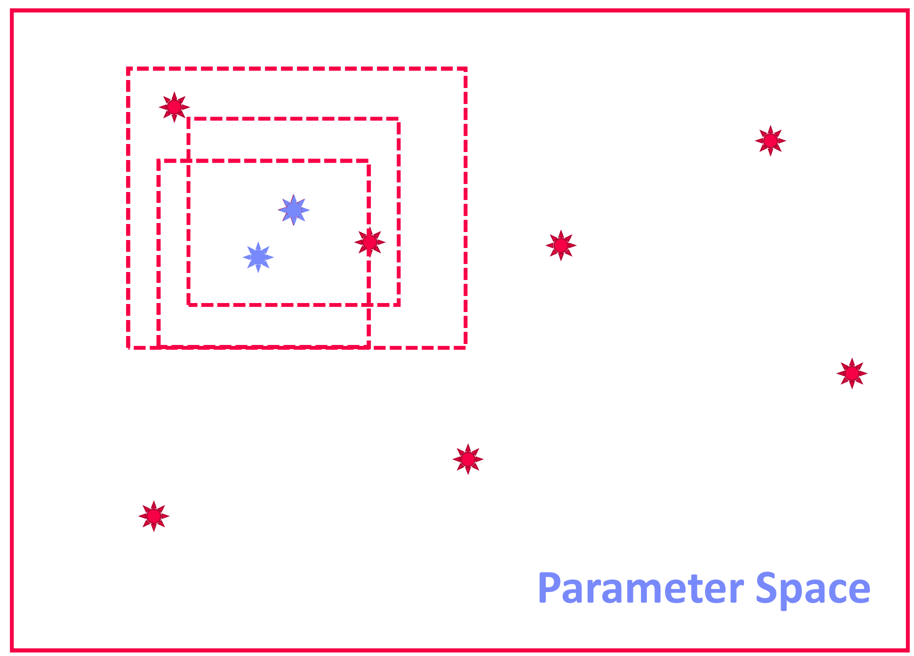
- **Simulated Annealing**
  - Decaying temperature schedules: $T_{k+1} = \alpha \cdot T_k$
  - **#1** Generate neighbor in **ε-env** of old point
  - **#2** Accept better points and worse points w/

Exploration vs exploitation

$$P(T_k) = \frac{1}{1 + \exp((f' - f)/T_k)}$$

- **Recursive Random Search**
  - Repeated restart
  - Sample and evaluate points
  - Determine best and shrink area if optimum unchanged
  - Realign area if new optimum found

[Tao Ye, Shivkumar Kalyanaraman: A recursive random search algorithm for large-scale network parameter configuration. **SIGMETRICS 2003**]



Bayesian Optimization

[Chris Thornton, Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. **KDD 2013**]

- **Overview BO**
  - Sequential Model-Based Optimization
  - Fit a probabilistic model based on the first n-1 evaluated hyper parameters
  - Use model to select next candidate
  - **Gaussian process (GP)** models, or tree-based Bayesian Optimization

Algorithm 1 SMBO

initialise model $\mathcal{M}_L$; $\mathcal{H} \leftarrow \emptyset$
while time budget for optimization has not been exhausted **do**
  $\boldsymbol{\lambda} \leftarrow$ candidate configuration from $\mathcal{M}_L$
  Compute $c = \mathcal{L}(A_{\boldsymbol{\lambda}}, \mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{valid}})$
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{\lambda}, c)\}$
  Update $\mathcal{M}_L$ given $\mathcal{H}$
end while
return $\boldsymbol{\lambda}$ from $\mathcal{H}$ with minimal c

- **Underlying Foundations**
  - The posterior probability of a model M given evidence E is proportional to the likelihood of E given M multiplied by prior probability of M
  - **Prior knowledge:** e.g., smoothness, noise-free
  - **Maximize acquisition function:**
    GP high objective (exploitation) and high prediction uncertainty (exploration)

$$P(M|E) = P(E|M)P(M)/P(E)$$

$$\rightarrow$$

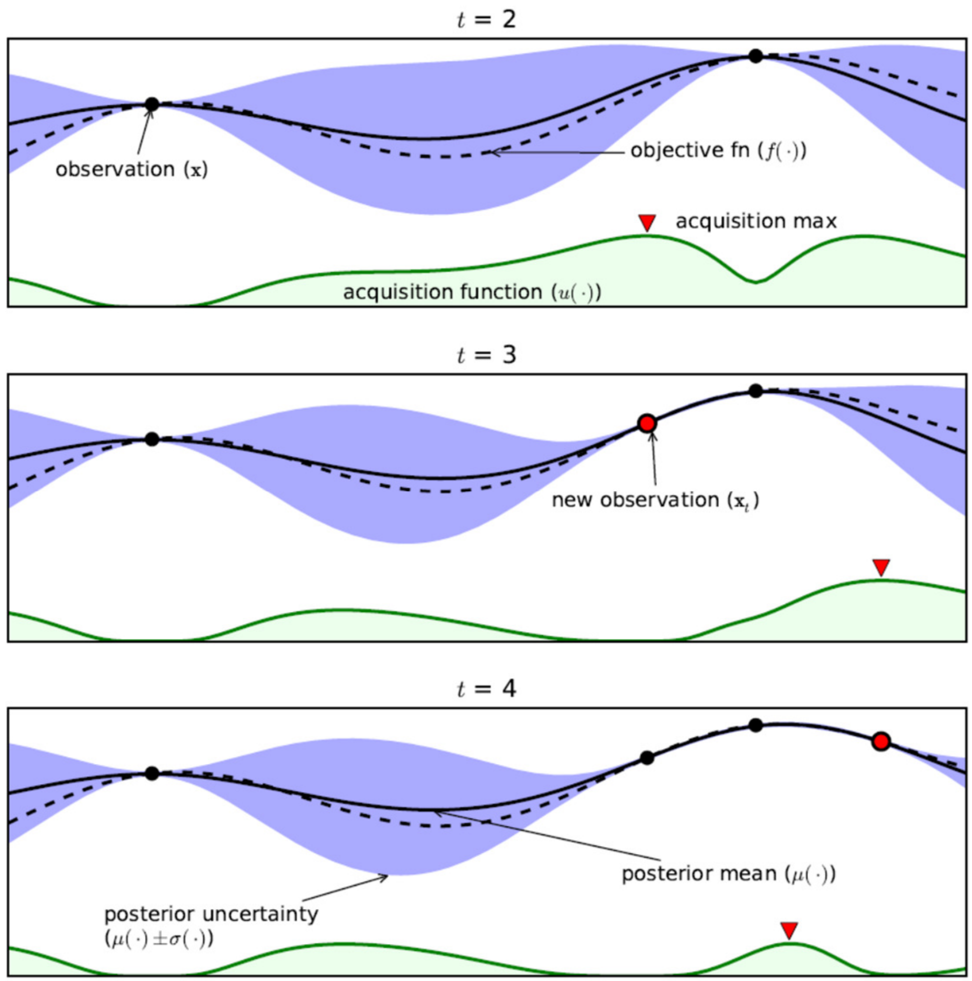
$$P(M|E) \propto P(E|M)P(M)$$

after next before

Bayesian Optimization, cont

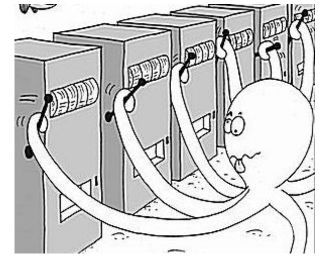
- **Example 1D Problem**
  - Gaussian Process
  - 4 iterations

[Eric Brochu, Vlad M. Cora, Nando de Freitas: A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. **CoRR 2010**]



Multi-armed Bandits and Hyperband

[**Credit:** blogs.mathworks.com]



- **Overview Multi-armed Bandits**
  - Motivation: model types have different quality
  - Select among k model types → **k-armed bandit problem**
  - Running score for each arm → **scheduling policy**

[Sébastien Bubeck, Nicolò Cesa-Bianchi: Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. **Foundations and Trends in Machine Learning 2012**]

- **Hyperband**
  - Non-stochastic setting, without parametric assumptions
  - Pure exploration algorithm for **infinite-armed bandits**
  - Based on **Successive Halving**
    - Successively discarding the worst-performing half of arms
    - Extended by doubling budget of arms in each iteration (no need to configure k, random search included)

[Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, Ameet Talwalkar: Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. **JMLR 2017**]

Selected AutoML Systems

- **Auto Weka**
  - Bayesian optimization with 28 learners, 11 ensemble/meta methods
- **Auto Sklearn**
  - Bayesian optimization with 15 classifiers, 14 feature prep, 4 data prep
- **TuPaQ**
  - **Multi-armed bandit** and large-scale
- **TPOT**
  - Genetic programming
- **Other Services**
  - Azure ML, Amazon ML
  - Google AutoML, H20 AutoML

[Chris Thornton et al: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. **KDD 2013**]

[Lars Kotthoffet al: Auto-WEKA 2.0: Automatic model selection and hyper-parameter optimization in WEKA. **JMLR 2017**]

[Matthias Feurer et al: Auto-sklearn: Efficient and Robust Automated Machine Learning. **Automated Machine Learning 2019**]

[Evan R. Sparks, Ameet Talwalkar, Daniel Haas, Michael J. Franklin, Michael I. Jordan, Tim Kraska: Automating model search for large scale machine learning. **SoCC 2015**]

[Randal S. Olson, Jason H. Moore: TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning. **Automated Machine Learning 2019**]

[Hantian Zhang, Luyuan Zeng, Wentao Wu, Ce Zhang: How Good Are Machine Learning Clouds for Binary Classification with Good Features? **CoRR 2017**]

Selected AutoML Systems, cont.

- **Alpine Meadow**
  - Logical and physical ML pipelines
  - **Multi-armed bandit** for pipeline selection
  - **Bayesian optimization** for hyper-parameters

[Zeyuan Shang et al: Democratizing Data Science through Interactive Curation of ML Pipelines. **SIGMOD 2019**]

- **Dabl** (Data Analysis Baseline Library)
  - Tools for simple data preparation and ML training
  - **Hyperband** (successive halving) for optimization

[https://amueller.github.io/dabl/dev/user_guide.html]

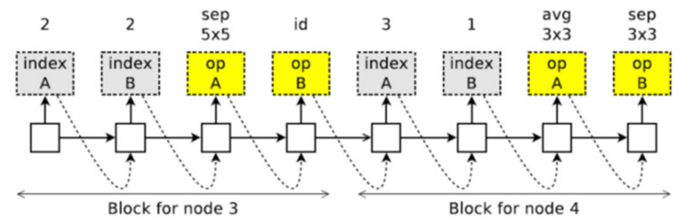
- **BOHB**
  - **Bayesian optimization** & **hyperband**
  - Queue-based **parallelization** of successive halving

[Stefan Falkner, Aaron Klein, Frank Hutter: BOHB: Robust and Efficient Hyper-parameter Optimization at Scale. **ICML 2018**]

- **Curated AutoML Paper Collections**

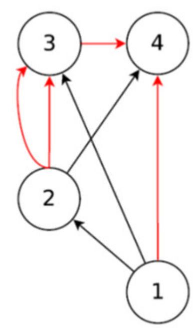
Neural Architecture Search

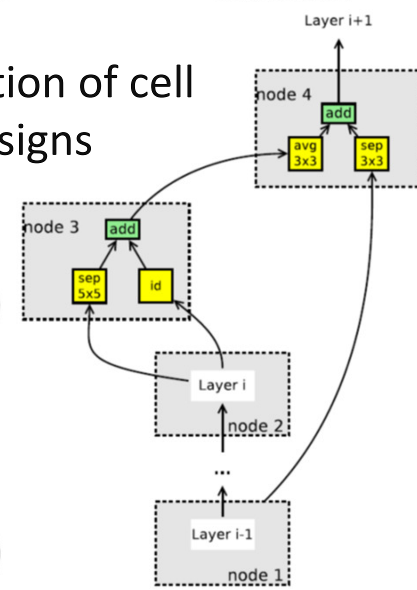
- **Motivation**
  - Design neural networks (type of layers / network) is often trial & error process
  - Accuracy vs necessary computation characterizes an architecture
  - ➔ **Automatic neural architecture search**



- **#1 Search Space of Building Blocks**
  - Define possible operations (e.g., identity, 3x3/5x5 separable convolution, avg/max pooling)
  - Define approach for connecting operations (pick 2 inputs, apply op, and add results)

Exploration of cell designs





[Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, Jeff Dean: Efficient Neural Architecture Search via Parameter Sharing. **ICML 2018**]

Neural Architecture Search, cont.

- **#2 Search Strategy**
  - Classical evolutionary algorithms
  - Recurrent neural networks (e.g., LSTM)
  - Bayesian optimization (with special distance metric)

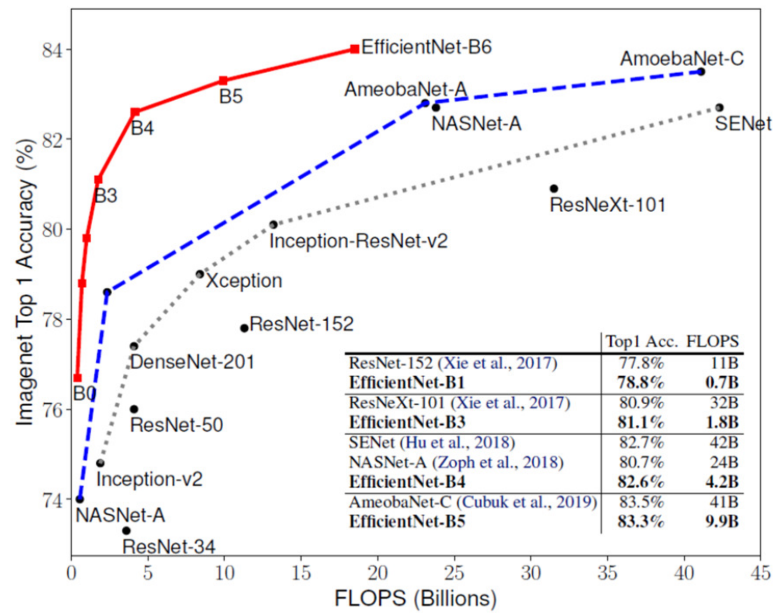
[Barret Zoph, Quoc V. Le: Neural Architecture Search with Reinforcement Learning. **ICLR 2017**]

[Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabás Póczos, Eric P. Xing: Neural Architecture Search with Bayesian Optimisation and Optimal Transport. **NeurIPS 2018**]

- **#3 Optimization Objective**
  - Max accuracy (min error)
  - Multi-objective (accuracy and runtime)

- **Excursus: Model Scaling**
  - Automatically scale-up small model for better accuracy
  - EfficientNet

[Mingxing Tan, Quoc V. Le: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. **ICML 2019**]



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**78.8%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.1%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.6%**	**4.2B**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.3%**	**9.9B**

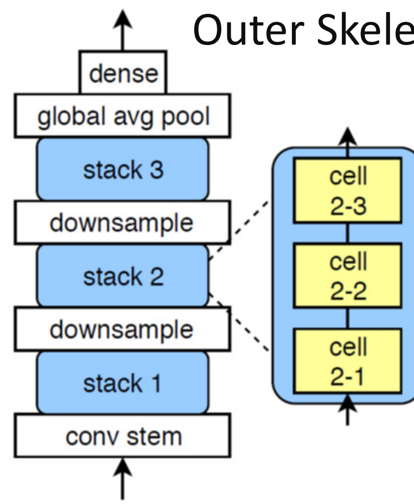
Neural Architecture Search, cont.

- **Problem: Computational Resources**
  - Huge computational requirements for NAS (even on small datasets)
  - ➔ **#1 Difficult to reproduce**, and **#2 barrier-to-entry**

- **Excursus: NAS-Bench-101**
  - 423K unique convolutional architectures
  - Training and evaluated ALL architectures, multiple times on **CIFAR-10**
  - Shared dataset: **5M trained models**

[Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, Frank Hutter: NAS-Bench-101: Towards Reproducible Neural Architecture Search. **ICML 2019**]

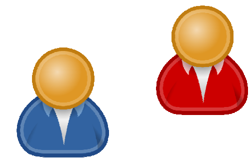
Outer Skeleton



Model Management

Overview Model Management

How did you create that model?
Did you consider X?



- **Motivation**
  - **Exploratory data science process** → trial and error (preparation, feature engineering, models, model selection)
  - **Different personas** (data engineer, ML expert, devops)
- **Problems**
  - No record of experiments, insights lost along the way
  - Difficult to reproduce results
  - Cannot search for or query models
  - Difficult to collaborate

[Manasi Vartak: ModelDB: A system to manage machine learning models, **Spark Summit 2017**]

- **Overview**
  - Experiment tracking and visualization
  - Coarse-grained ML pipeline provenance and versioning
  - Fine-grained data provenance (data-/ops-oriented)

Background: Data Provenance and Lineage

- **Overview**
  - Base query Q(D) = O with database D = $\{R_1, \ldots, R_n\}$
  - **Forward lineage query:** $L_f(R_i'', O')$ from subset of input relation to output
  - **Backward lineage query:** $L_b(O', R_i)$ from subset of outputs to base tables
- **#1 Lazy Lineage Query Evaluation**
  - Rewrite (**invert**) lineage queries as relational queries over input relations
  - No runtime overhead but slow lineage query processing
- **#2 Eager Lineage Query Evaluation**
  - Materialize **annotations** (data/transforms) during base query evaluation
  - Runtime overhead but fast lineage query processing
  - Lineage capture: **Logical** (relational) vs **physical** (instrumented physical ops)

[Fotis Psallidas, Eugene Wu: Smoke: Fine-grained Lineage at Interactive Speed. **PVLDB 2018**]

Model Management Systems

- **ModelHub**
  - Versioning system for DNN models, including provenance tracking
  - DSL for model exploration and enumeration queries (model selection + hyper parameters)
  - Model versions stored as deltas

[Hui Miao, Ang Li, Larry S. Davis, Amol Deshpande: ModelHub: Deep Learning Lifecycle Management. **ICDE 2017**]

- **ModelDB**
  - Model and provenance logging for ML pipelines via programmatic APIs
  - Support for different ML systems (e.g., spark.ml, scikit-learn, others)
  - GUIs for capturing meta data and metric visualization

[Manasi Vartak, Samuel Madden: MODELDB: Opportunities and Challenges in Managing Machine Learning Models. **IEEE Data Eng. Bull. 2018**]

Model Management Systems, cont.

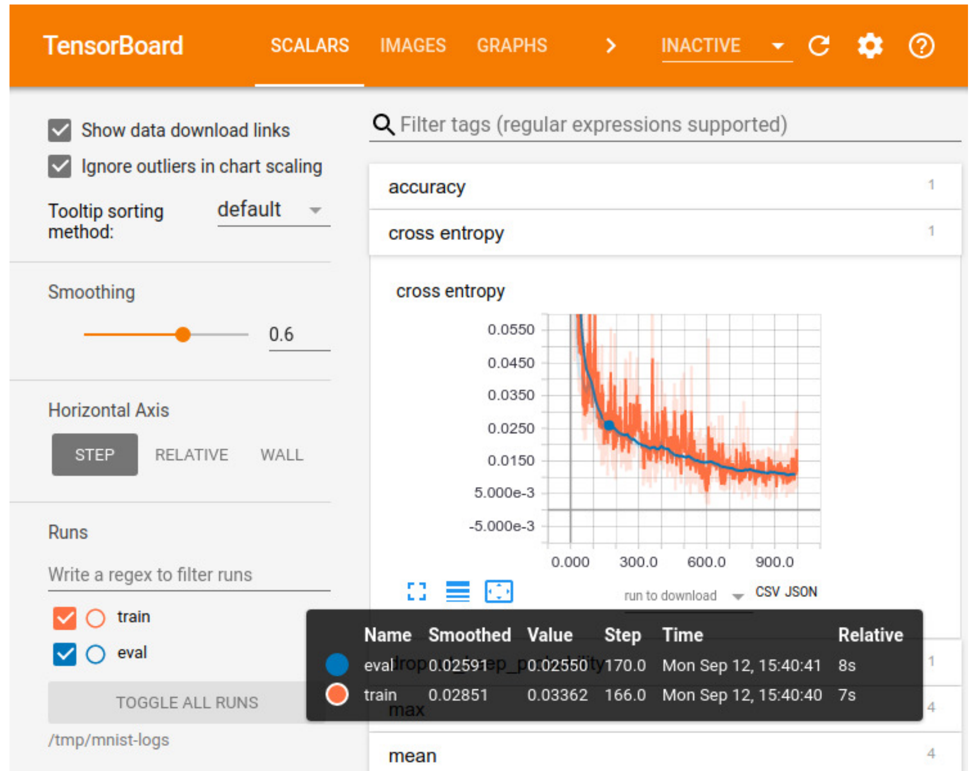
- **MLflow**
  - An open source platform for the machine learning lifecycle
  - Use of existing ML systems and various language bindings
  - **MLflow Tracking:** logging and querying experiments
  - **Mlflow Projects:** packaging/reproduction of ML pipeline results
  - **MLflow Models:** deployment of models in various services/tools

[Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, Corey Zumar: Accelerating the Machine Learning Lifecycle with MLflow. IEEE Data Eng. Bull. 41(4) 2018]

Experiment Tracking

- **TensorFlow: TensorBoard**
  - Suite of visualization tools
  - Explicitly track and write summary statistics
  - Visualize behavior over time and across experiments
  - Different folders for model versioning?

- **Other Tools:**
  - Integration w/ TensorBoard
  - Lots of custom logging and plotting tools



[**Credit:** https://www.tensorflow.org/guide/summaries_and_tensorboard]

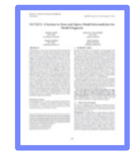
Provenance for ML Pipelines (fine-grained)

- **DEX: Dataset Versioning**
  - **Versioning of datasets, stored with delta encoding**
  - Checkout, intersection, union queries over deltas
  - Query optimization for finding efficient plans

[Amit Chavan, Amol Deshpande: DEX: Query Execution in a Delta-based Storage System. **SIGMOD 2017**]

- **MISTIQUE: Intermediates of ML Pipelines**
  - Capturing, storage, querying of intermediates
  - **Lossy deduplication and compression**
  - Adaptive querying/materialization for finding efficient plans

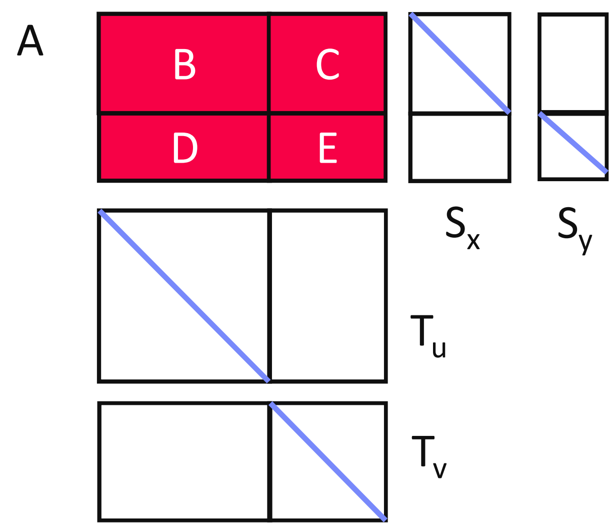
[Manasi Vartak et al: MISTIQUE: A System to Store and Query Model Intermediates for Model Diagnosis. **SIGMOD 2018**]



- **Linear Algebra Provenance**
  - Provenance propagation by decomposition
  - Annotate parts w/ provenance polynomials (identifiers of contributing inputs + impact)

$$A = S_x B T_u + S_x C T_v + S_y D T_u + S_y E T_v$$

[Zhepeng Yan, Val Tannen, Zachary G. Ives: Fine-grained Provenance for Linear Algebra Operators. **TaPP 2016**]



Provenance for ML Pipelines (coarse-grained)

- **MLflow**
  - Programmatic API for tracking parameters, experiments, and results
  - **autolog()** for specific params

[**Credit:** https://databricks.com/blog/2018/06/05]

```
import mlflow
mlflow.log_param("num_dimensions", 8)
mlflow.log_param("regularization", 0.1)
mlflow.log_metric("accuracy", 0.1)
mlflow.log_artifact("roc.png")
```

- **Flor (on Ground)**
  - DSL embedded in python for managing the workflow development phase of the ML lifecycle
  - DAGs of actions, artifacts, and literals
  - Data context generated by activities in Ground

[Credit: https://rise.cs.berkeley.edu/projects/jarvis/]

[Joseph M. Hellerstein et al: Ground: A Data Context Service. **CIDR 2017**]

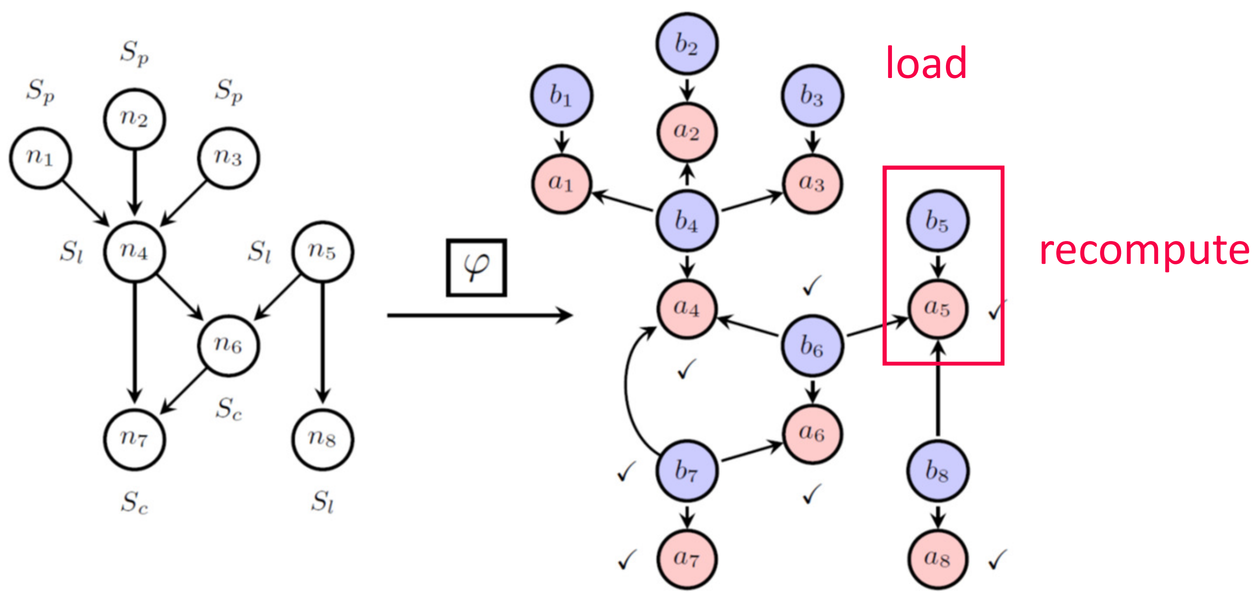
- **Dataset Relationship Management**
  - **Reuse**, **reveal**, **revise**, **retarget**, **reward**
  - Code-to-data relationships (data provenance)
  - Data-to-code relationships (potential transforms)

[Zachary G. Ives, Yi Zhang, Soonbo Han, Nan Zheng,: Dataset Relationship Management. **CIDR 2019**]

Provenance for ML Pipelines (coarse-grained), cont.

- **HELIX**
  - Goal: focus on iterative development w/ small modifications (trial & error)
  - Caching, reuse, and recomputation
  - Reuse as Max-Flow problem → NP-hard → heuristics
  - Materialization to disk for future reuse

[Doris Xin, Stephen Macke, Litian Ma, Jialin Liu, Shuchen Song, Aditya G. Parameswaran: Helix: Holistic Optimization for Accelerating Iterative Machine Learning. **PVLDB 2018**]



Fine-grained Lineage in SystemDS

- **Problem**
  - **Exploratory data science** (data preprocessing, model configurations)
  - **Reproducibility** and **explainability** of trained models (data, parameters, prep)

  ➔ **Lineage/Provenance as Key Enabling Technique:**
  Model versioning, reuse of intermediates, incremental maintenance, auto differentiation, and debugging (query processing over lineage)

- **Efficient Lineage Tracing**
  - Tracing of inputs, literals, and **non-determinism**
  - **Trace lineage of logical operations** for all live variables, store along outputs, program/output reconstruction possible:

```
X = eval(deserialize(serialize(lineage(X))))
```

  - **Proactive deduplication** of lineage traces for loops

Fine-grained Lineage in SystemDS, cont.

$O(k(mn^2+n^3)) \rightarrow O(mn^2+kn^3)$

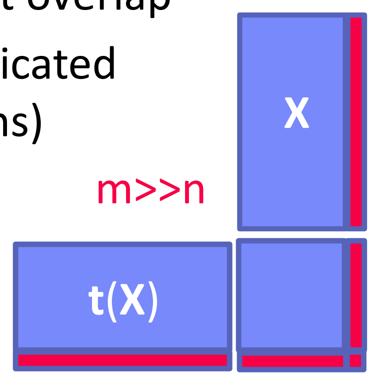
```
for( i in 1:numModels )
  R[,i] = lm(X, y, lambda[i,], ...)
```

- **Full Reuse of Intermediates**
  - Before executing instruction, probe output lineage in cache `Map<Lineage, MatrixBlock>`
  - Cost-based/heuristic caching and eviction decisions (compiler-assisted)

```
m_lmDS = function(...) {
  l = matrix(reg,ncol(X),1)
  A = t(X) %*% X + diag(l)
  b = t(X) %*% y
  beta = solve(A, b) ...}
```

- **Partial Reuse of Intermediates**
  - **Problem:** Often partial result overlap
  - Reuse partial results via dedicated rewrites (compensation plans)
  - Example: steplm

$m >> n$




```
m_steplm = function(...) {
  while( continue ) {
    parfor( i in 1:n ) {
      if( !fixed[1,i] ) {
        Xi = cbind(Xg, X[,i])
        B[,i] = lm(Xi, y, ...)
    } }
    # add best to Xg
    # (AIC)
} }
```

$O(n^2(mn^2+n^3)) \rightarrow O(n^2(mn+n^3))$

Summary and Q&A

- **Data Augmentation**
- **Model Selection Techniques**
- **Model Management**

- **Next Lectures**
  - **June 11/12:** Corpus Christi (Fronleichnam)
  - **11 Model Debugging Techniques** [Jun 19]
  - **12 Model Serving Systems and Techniques** [Jun 26]