COMS E6998 012/15099

Practical Deep Learning Systems Performance

Lecture 13 12/05/19

Logistics

- Homework 5 is due by Dec 13 11:59 PM
  - Question 2: Work in a group to collect data. At most 5 in a group.
- Requirements for final project/seminar and rubric are posted. Also posted is a template for presentation.
- Dec 10th: Deadline for 10% technical community participation
- Dec 9th and 10th : Project and seminar presentation. Please select a slot in the shared sheet available under Collaborations on Canvas.

Recall from last lecture

- Containers, Docker, Kubernetes
- Fabric for Deep Learning, IBM open source distributed DL platform
- Kubeflow, a machine learning toolkit for Kubernetes
- Amazon Sagemaker, a fully managed machine learning service

Neural Architecture Search

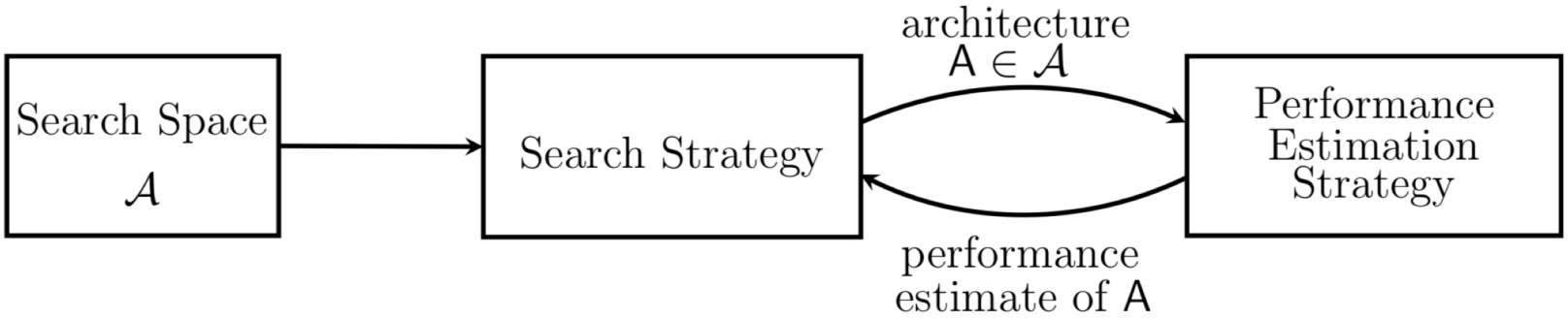


Figure 1: Abstract illustration of Neural Architecture Search methods. A search strategy selects an architecture A from a predefined search space $\mathcal{A}$. The architecture is passed to a performance estimation strategy, which returns the estimated performance of A to the search strategy.

https://arxiv.org/pdf/1808.05377.pdf

NAS Search Space: Layer Based

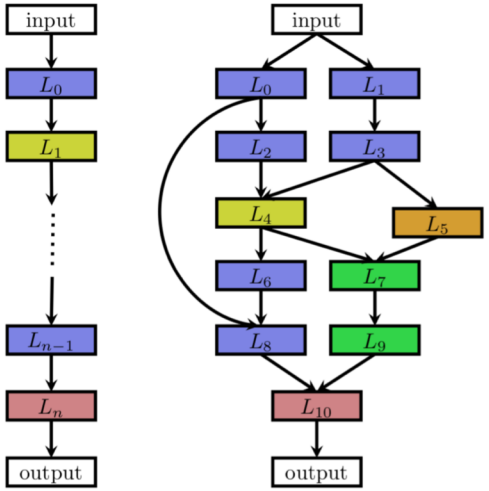


Figure 2: An illustration of different architecture spaces. Each node in the graphs corresponds to a layer in a neural network, e.g., a convolutional or pooling layer. Different layer types are visualized by different colors. An edge from layer L_i to layer L_j denotes that L_j receives the output of L_i as input. Left: an element of a chain-structured space. Right: an element of a more complex search space with additional layer types and multiple branches and skip connections.

NAS Search Space: Cell Based

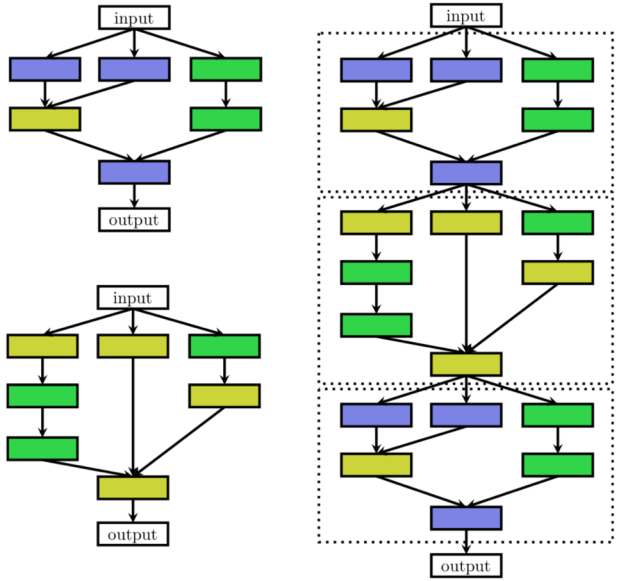


Figure 3: Illustration of the cell search space. Left: Two different cells, e.g., a normal cell (top) and a reduction cell (bottom) (Zoph et al., 2018). Right: an architecture built by stacking the cells sequentially. Note that cells can also be combined in a more complex manner, such as in multi-branch spaces, by simply replacing layers with cells.

NAS Search Strategy

- Random search, Bayesian optimization, Evolutionary methods, Reinforcement learning (RL), and gradient-based methods.

NAS Performance Estimation

Speed-up method	How are speed-ups achieved?	References
Lower fidelity estimates	Training time reduced by training for fewer epochs, on subset of data, downscaled models, downscaled data, ...	Li et al. (2017), Zoph et al. (2018), Zela et al. (2018), Falkner et al. (2018), Real et al. (2019), Runge et al. (2019)
Learning Curve Extrapolation	Training time reduced as performance can be extrapolated after just a few epochs of training.	Swersky et al. (2014), Domhan et al. (2015), Klein et al. (2017a), Baker et al. (2017b)
Weight Inheritance/ Network Morphisms	Instead of training models from scratch, they are warm-started by inheriting weights of, e.g., a parent model.	Real et al. (2017), Elsken et al. (2017), Elsken et al. (2019), Cai et al. (2018a,b)
One-Shot Models/ Weight Sharing	Only the one-shot model needs to be trained; its weights are then shared across different architectures that are just subgraphs of the one-shot model.	Saxena and Verbeek (2016), Pham et al. (2018), Bender et al. (2018), Liu et al. (2019b), Cai et al. (2019), Xie et al. (2019)

One shot architecture

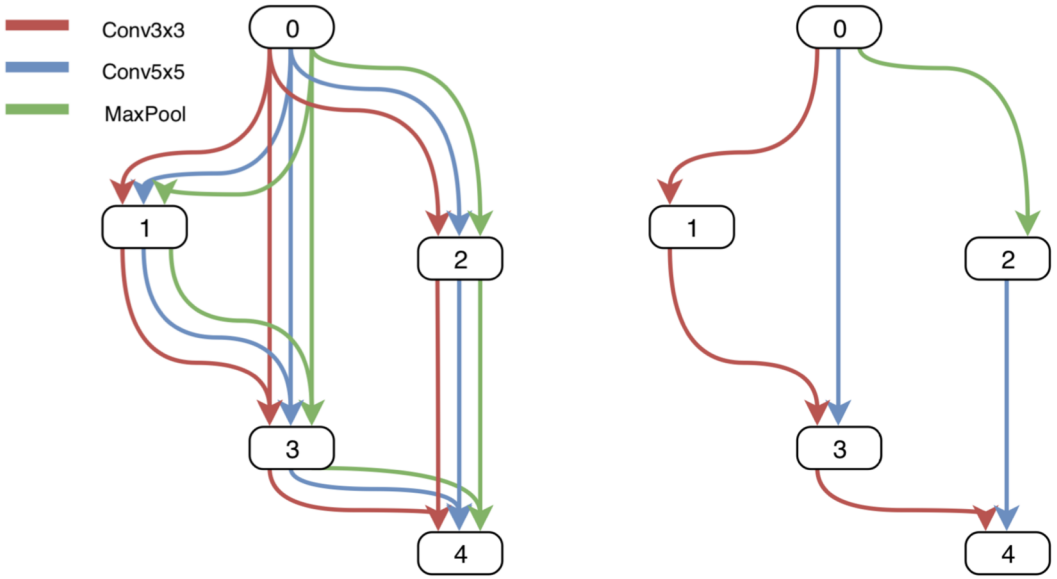
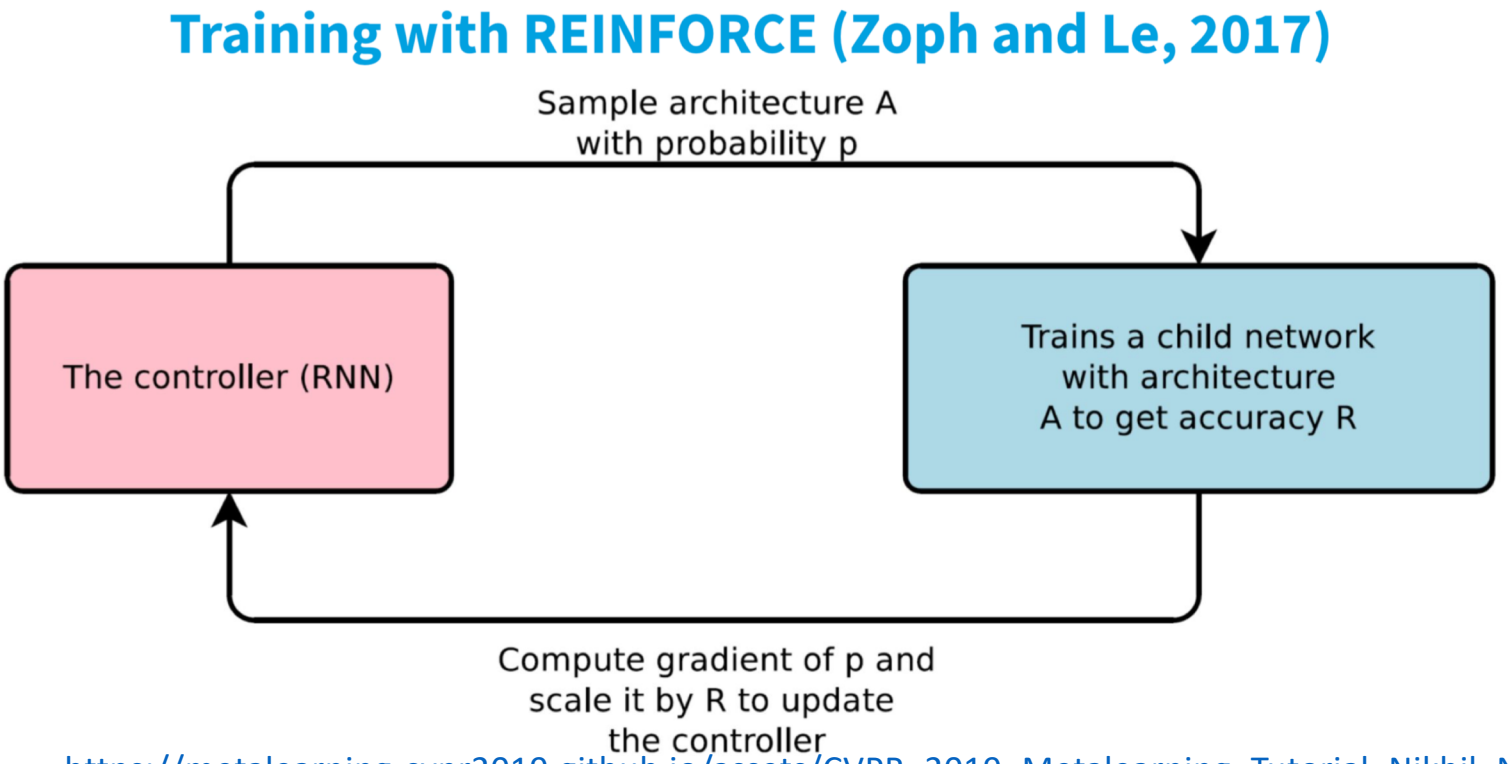


Illustration of one-shot architecture search. Simple network with an input node (denoted as 0), three hidden nodes (denoted as 1,2,3) and one output node (denoted as 4). Instead of applying a single operation (such as a 3x3 convolution) to a node, the one-shot model (left) contains several candidate operations for every node, namely 3x3 convolution (red edges), 5x5 convolution (blue edges) and MaxPooling (green edges) in the above illustration. Once the one-shot model is trained, its weights are shared across different architectures, which are simply subgraphs of the one-shot model (right). Figure inspired by Liu et al. (2019b).

NAS Search



https://metalearning-cvpr2019.github.io/assets/CVPR_2019_Metalearning_Tutorial_Nikhil_Naik.pdf

TAPAS

- Train-less accuracy predictor for architecture search (TAPAS)
- Predicts accuracy of CNN based classifiers on unseen data without training
- Achieved by adapting past predictions to the “difficulty” level of the dataset
- Characterization of the dataset-difficulty
  - Reliable estimation a CNN performance is dependent on the complexity of dataset

Istrate et al. TAPAS: Train-less Accuracy Predictor for Architecture Search. 2018

TAPAS Framework

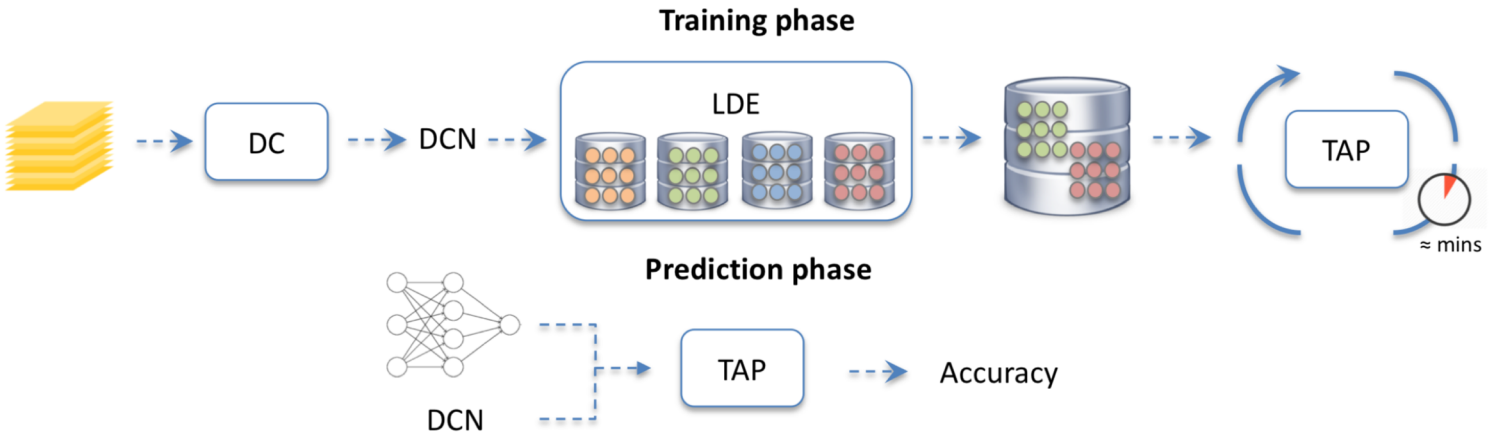


Figure 1: Schematic TAPAS workflow. First row: the Dataset Characterization (DC) takes a new, unseen dataset and characterizes its difficulty by computing the Dataset Characterization Number (DCN). This number is then used to select a subset of experiments executed on similarly difficult datasets from the Lifelong Database of Experiments (LDE). Subsequently, the filtered experiments are used to train the Train-less Accuracy Predictor (TAP), an operation that takes up to a few minutes. Second row: the trained TAP takes the network architecture structure and the dataset DCN and predict the peak accuracy reachable after training. This phase scales very efficiently in a few seconds over a large number of networks.

Istrate et al. TAPAS: Train-less Accuracy Predictor for Architecture Search. 2018

TAPAS Components

1. Dataset Characterization (DC)
   - Receives an unseen dataset and computes a scalar score, namely the Dataset Characterization Number (DCN)
   - DCN is calculated by training Deep Normalized ProbeNet (modest size network) for few epochs
   - DCN is a rough estimation of dataset difficulty, tolerant to approximations
2. Lifelong Database of Experiments (LDE)
   - Ingests training experiments of NNs on a variety of image classification datasets executed inside the TAPAS framework
   - Experiment: CNN architecture description, the training hyper-parameters, the employed dataset (with its DCN), the achieved accuracy
   - A continuously growing DB
   - Returns all experiments performed with datasets with DCN close to the target
3. Train-less Accuracy Predictor (TAP)
   - Given an NN architecture and a DCN, it predicts the accuracy without training the network
   - Employs a layer-by-layer encoding vector of NN architecture
   - 2 stacked LSTM

TAPAS Performance

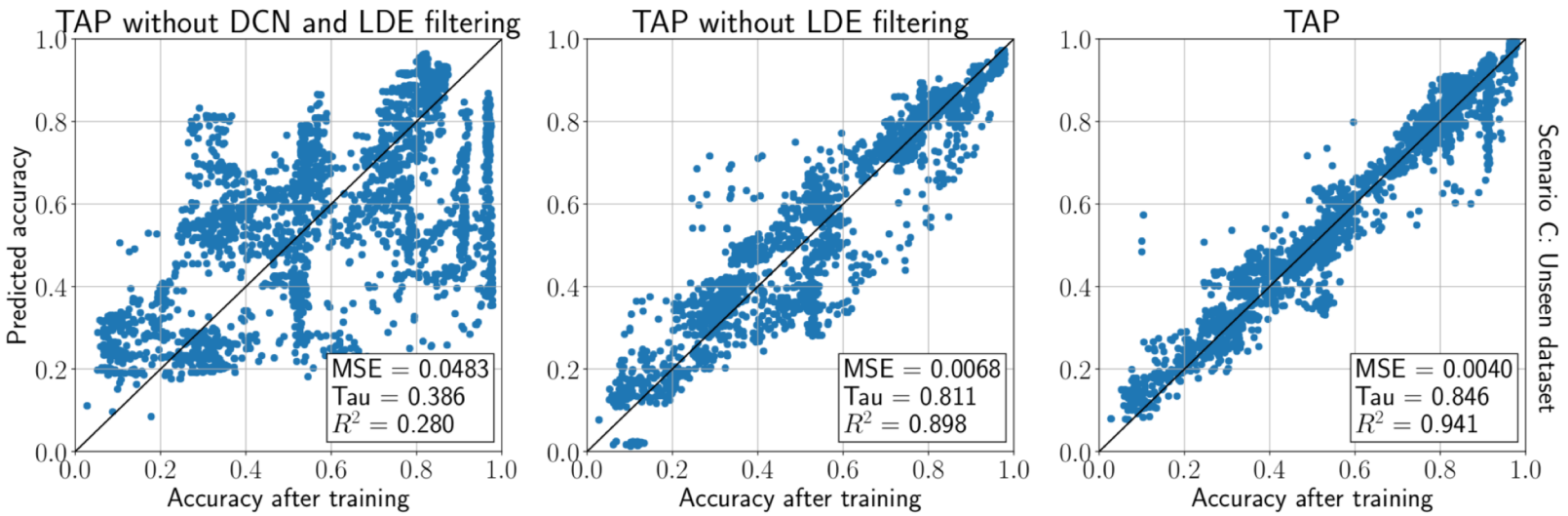


Figure 5: Predicted vs real performance (i.e., after training) for Scenario C. Left plot: TAP trained without DCN or LDE pre-filtering. Middle plot: TAP trained with DCN, but LDE is not pre-filtered. Right plot: TAP trained only on LDE experiments with similar dataset difficulty, according to (1).

Hyper parameter optimization

- Random search
- Bayesian optimization methods
  - Snoek et al., 2012; Hutter et al., 2011; Bergstra et al., 2011; Thornton et al., 2013; Eggensperger et al., 2013; Snoek et al., 2015b
- Hyperband

Neural Network Synthesis

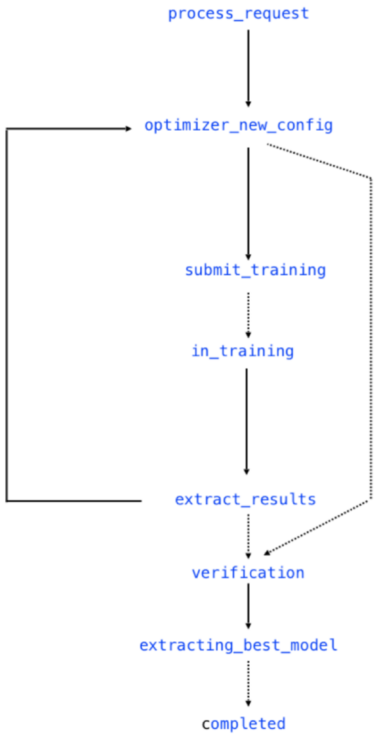


Figure 1: Execution Pipeline Operational States

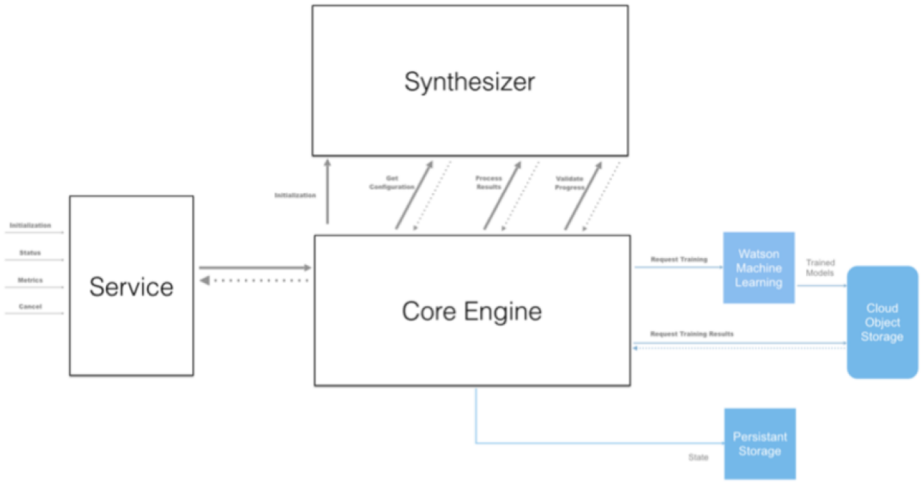


Figure 2: NeuNetS Component Architecture

https://arxiv.org/pdf/1901.06261.pdf

Automated Machine Learning

- IBM Auto AI
- Run a sample AutoAI experiment in IBM WML
- H20 AutoML

Cognito: System for Automated Feature Engineering

- Feature engineering: Finding the minimal set of transforms that maximizes the model performance
- Solution space is unbounded
- *Cognito :* performs automatic feature engineering on a given dataset for supervised learning.
- Adopts a greedy incremental search policy
- Tree based search approach
  - Different search strategies: depth-first traversal, global traversal, balanced traversal
- Feature selection
  - Application of a transform can severely increase the number of columns in the dataset, e.g., log transform on 10 cols. → adds 10 new cols.; 2 column product transform on 10 cols. → adds 45 new cols.
  - Learning algorithms do not scale well with increasing column count
  - Option of performing *feature selection* at each node on the newly generated features post the transform application
  - Can cause premature discarding of features on which additional transformations might have yielded useful features

Cognito Demo and Performance

Look at video: https://www.youtube.com/watch?v=hJlG0mvynDo

Data	#Row	#Col	Before FE Accuracy	After FE Accuracy	%gain
Australian Credit	690	15	0.631	0.675	7%
Svmguide3	1243	22	0.617	0.796	29%
Svmguide1	3089	5	0.951	0.971	2%
Ionosphere	351	35	0.742	0.805	8%
Pima Diabetes	768	9	0.621	0.688	11%
German Credit	1k	25	0.690	0.752	9%
Weather	1m	424	0.660	0.810	23%
Energy	100	6	0.392	0.548	40%

AI Model Lifecycle

- ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI

Reference Papers

- Elsken et al. Neural Architecture Search: A Survey. JMLR 2019
- Istrate et al. TAPAS: Train-less Accuracy Predictor for Architecture Search. 2018
- Li et al. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. 2018
- Sood et al. NEUNETS: AN AUTOMATED SYNTHESIS ENGINE FOR NEURAL NETWORK DESIGN. 2019
- Hummer et al. ModelOps: Cloud-based Lifecycle Management for Reliable and Trusted AI. IC2E 2019
- Khurana et al. Cognito: Automated Feature Engineering for Supervised Learning. ICDM 2017
- Witsuba et al. A Survey on Neural Architecture Search. 2019
- Hummer et al. ModelOps**:** Cloud-Based Lifecycle Management for Reliable and Trusted AI. 2019