Machine Learning Model Management and Inference (MLMMI)
04 Model Selection Systems

Dr. Arnab Phani
BIFOLD, TU Berlin
DEEM Lab

06.05.2026

ICDE 2026 Keynote

Model Lake Management

Speaker: Renée J. Miller (University of Waterloo)

Keynote Session: Wednesday, May 6 | 8:30 - 9:30 AM

Room: Place du Canada

Abstract

The concept of data lakes emerged in the early 2010s to mean collections of raw, unstructured data —- as organizations recognized the untapped value of this messy data. The study of data lakes has evolved and matured shaping how we store, manage, and extract insights from these massive heterogeneous information stores. We are now seeing the emergence of model lakes, repositories of large sets of pre-trained AI models. In this talk, I argue that model lakes offer a new transformative paradigm for organizing and understanding the growing ecosystem of AI models. I review our vision for model lakes and what the application of principled data management can do for understanding and using AI models.

Bio

Renée J. Miller is the Canada Excellence Research Chair in Data Intelligence at the University of Waterloo. She is a Fellow of the Royal Society of Canada, Canada's National Academy of Science, Engineering and the Humanities. She received the US Presidential Early Career Award for Scientists and Engineers (PECASE), the highest honor bestowed by the United States government on outstanding scientists and engineers beginning their careers. She received an NSF CAREER Award, the Ontario Premier's Research Excellence Award, and an IBM Faculty Award. She formerly held the Bell Canada Chair of Information Systems at the University of Toronto and a University Distinguished Professorship at Northeastern University. She is a Fellow of the ACM and the AAAS. Her work has focused on the long-standing open problem of data integration and has achieved the goal of building practical data integration systems. She and her colleagues received the ICDT Test-of-Time Award and the 2020 Alonzo Church Award for Outstanding Contributions to Logic and Computation for their influential work establishing the foundations of data exchange. For her body of work, she has received the CS Canada Lifetime Achievement Award in Computer Science. Professor Miller served as president of the non-profit Very Large Data Base (VLDB) Foundation and an Editor-in-Chief of the VLDB Journal. She received her PhD in Computer Science from the University of Wisconsin, Madison and bachelor's degrees in Mathematics and Cognitive Science from MIT.

Announcements

- **No Lecture on June 03**
  - I will be in India to attend SIGMOD
- **Programming Assignment 1**
  - Share your experience
  - MOSES registration done?
- **Open-weight LLM API Access**
- **Invited Talk: Nils Strassenburg**
  - May 13
  - Submit questions by May 11

Chat AI Access

1. **Log into** https://academiccloud.de/services/chatai/ **at least once** with your TU Berlin account
2. **Click on Book** at https://kisski.gwdg.de/en/leistungen/2-02-llm-service
3. **Fill out the form**, here is some required information

   Group leader: Prof. Dr. Sebastian Schelter
   Working group/ department: FG Management of Data Science Processes
   Organisation: Technische Universitaet Berlin
   Website: https://deem.berlin
   Address: Technische Universität Berlin, FG Management of Data Science Processes, Sekr. FR 7-3, Franklinstr. 28 / 29, 10587 Berlin, Germany
   Requirements: <briefly describe that you need LLMs for a course project>
   Critical data in regards to GDPR or ISO27001 is processed: No

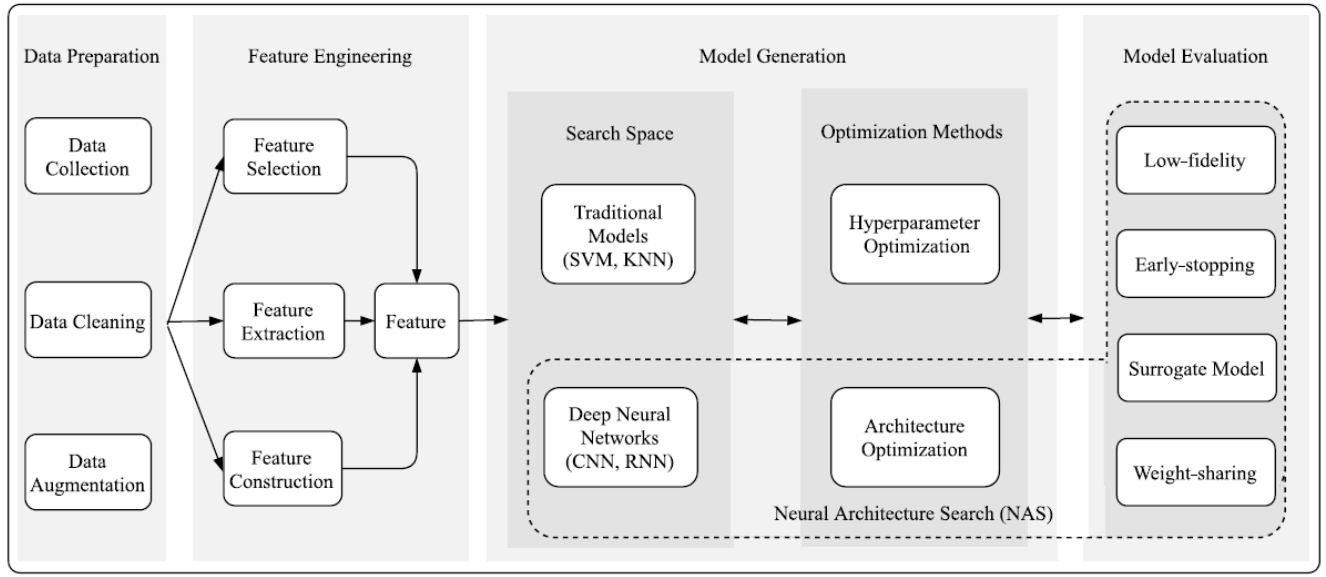
https://chat-ai.academiccloud.de/

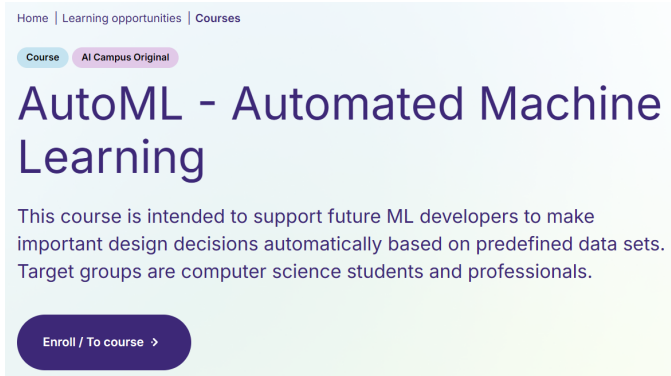
Agenda

- **Model Selection Techniques**
- **Feature Extraction and Transfer learning**
- **Exercise 2: Transfer Learning**

Model Selection Techniques

AutoML





Uni Freiburg AutoML Course

- **Motivation**
  - AutoML steer 3 key activities: feature engineering, model/architecture selection, hyperparameter tuning
  - Maximize performance under budget
  - Challenges: large search space, expensive evaluation, limited resources
  - This is an optimization problem and system scheduling problem

* AutoML: A survey of the state-of-the-art. 2021.

Hyperparameter Tuning

Linear Models

Learning rate,
Regularization

Tree-Based Models

Max depth,
Min samples split,

DNNs

Learning rate,
Batch size,
#Epochs,
Dropout rate

K-NN

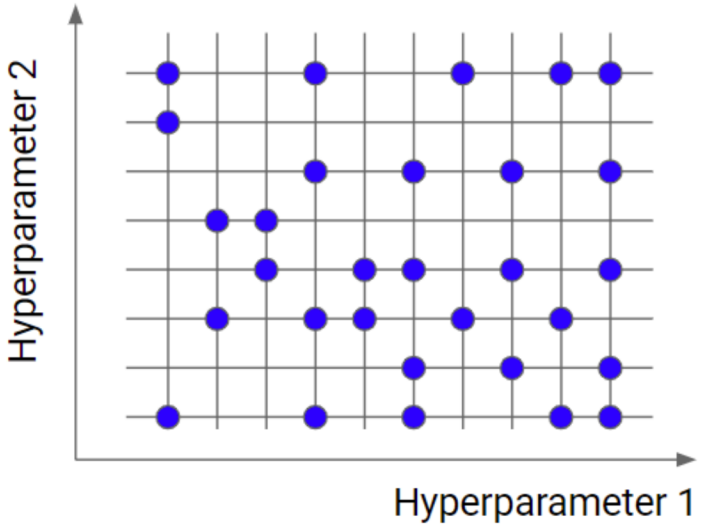
K,
Distance metric

Clustering Models

#Clusters,
Max iterations

- **Hyperparameters**
  - Almost all ML models have hyperparameters knobs
  - Most of them raise bias slightly but reduce variance more
  - No hyperparameter settings universally best for all tasks/data

Grid Search Hyperparameter Tuning



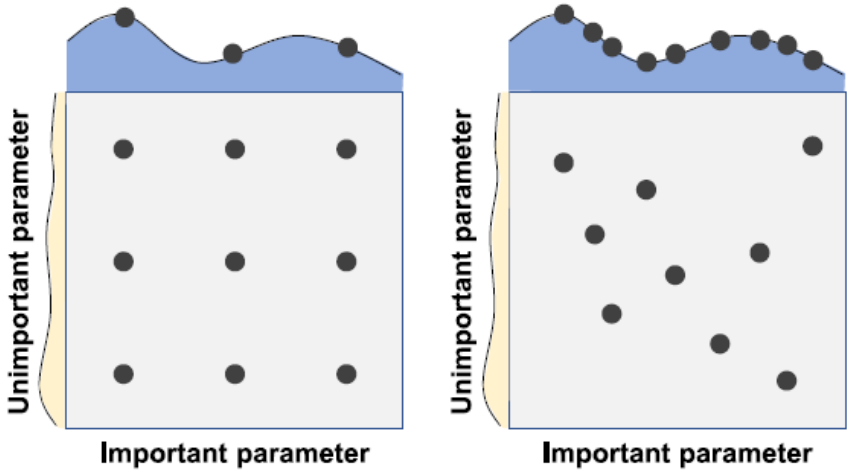
GridSearchCV

```
class sklearn.model_selection.GridSearchCV(estimator, param_grid, *, scoring=None,
n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs',
error_score=nan, return_train_score=False)                                    [source]
```

Exhaustive search over specified parameter values for an estimator.

- **Basic Approach**
  - Given n hyper parameters λ1, …, λn with domains Λ1, …, Λn
  - Enumerate and evaluate parameter space $\Lambda \subseteq \Lambda_1 \times \ldots \times \Lambda_n$
  - Discretize the continuous hyperparameters
  - Exhaustive search; parallelizable but computationally expensive
  - Non-convex or unknown parameter space

Random Search



Grid and random search of nine trials

RandomizedSearchCV

```
class sklearn.model_selection.RandomizedSearchCV(estimator, param_distributions, *,
n_iter=10, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0,
pre_dispatch='2*n_jobs', random_state=None, error_score=nan,
return_train_score=False)                                                [source]
```

Randomized search on hyper parameters.

- **Overview**
  - Randomized search over parameters; each setting is sampled from a distribution of possible values
  - A specified budget that is independent of #parameters and #values
  - Adding parameters that do not influence the performance does not decrease efficiency
  - Computationally less expensive

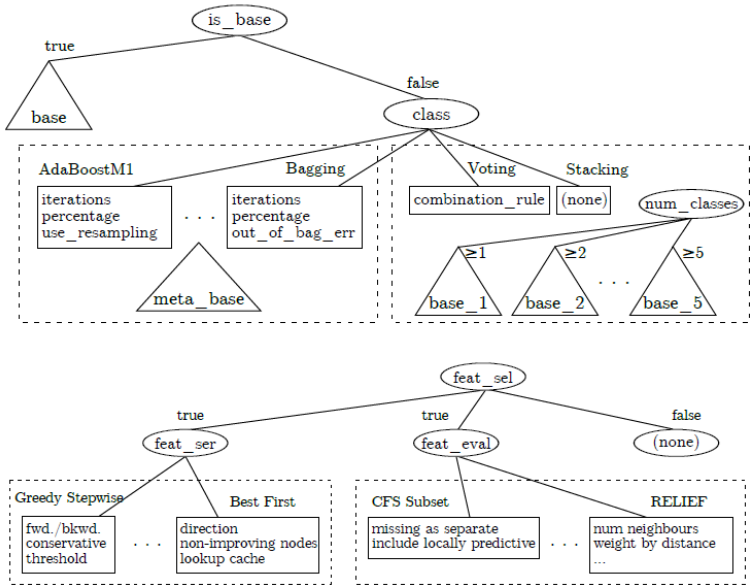
* Random Search for Hyper-Parameter Optimization. JMLR. 2012

Bayesian Optimization and Auto-WEKA

Algorithm 1 SMBO

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

Combined Alg. Selection & Hyp. Par Opt.



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

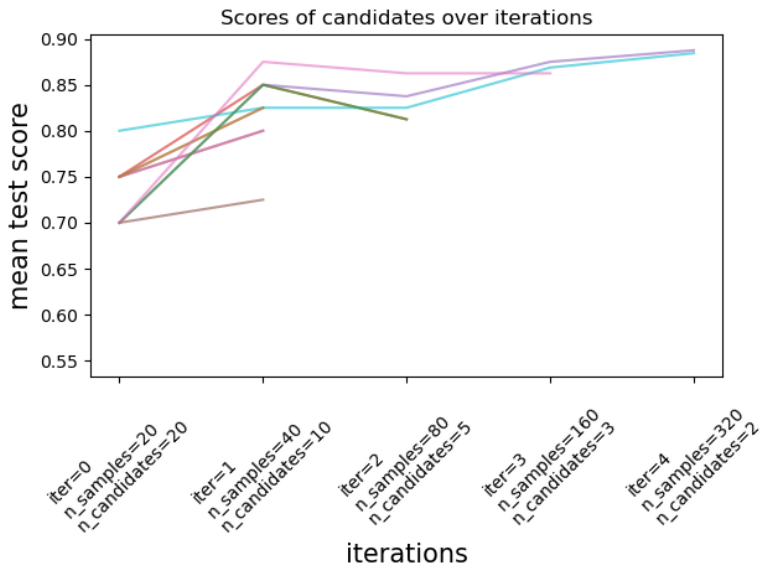
- **Overview**
  - Fit a probabilistic model based on the first n-1 evaluated hyper parameters
  - Use model to select next candidate
  - Gaussian process (GP) models, or tree-based Bayesian Optimization
  - Auto-WEKA: Combined model selection and hyperparameter optimization using Bayesian optimization
  - Choice of model itself is considered a hyperparameter

* Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. KDD. 2013.

Multi-armed Bandits and Hyperband

	$s=4$		$s=3$		$s=2$		$s=1$		$s=0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

n_i: # hyp.par.configs run, r_i: # epochs per config



n_samples --> resource

- **Multi-armed Bandits**
  - Motivation: model types have different quality
  - Select among k model types (k-armed bandit problem)
  - Running score for each arm and repeat
- **Hyperband**
  - Basic Idea: For iterative algorithms (SGD), stop non-promising hyp.par. configs earlier
  - **Successive Halving**: An iterative selection process where all candidates are evaluated with a small amount of resources at the first iteration. Only some of these candidates are selected for the next iteration, which will be allocated more resources.

* Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. JMLR. 2018.

Multi-tenant Model Selection

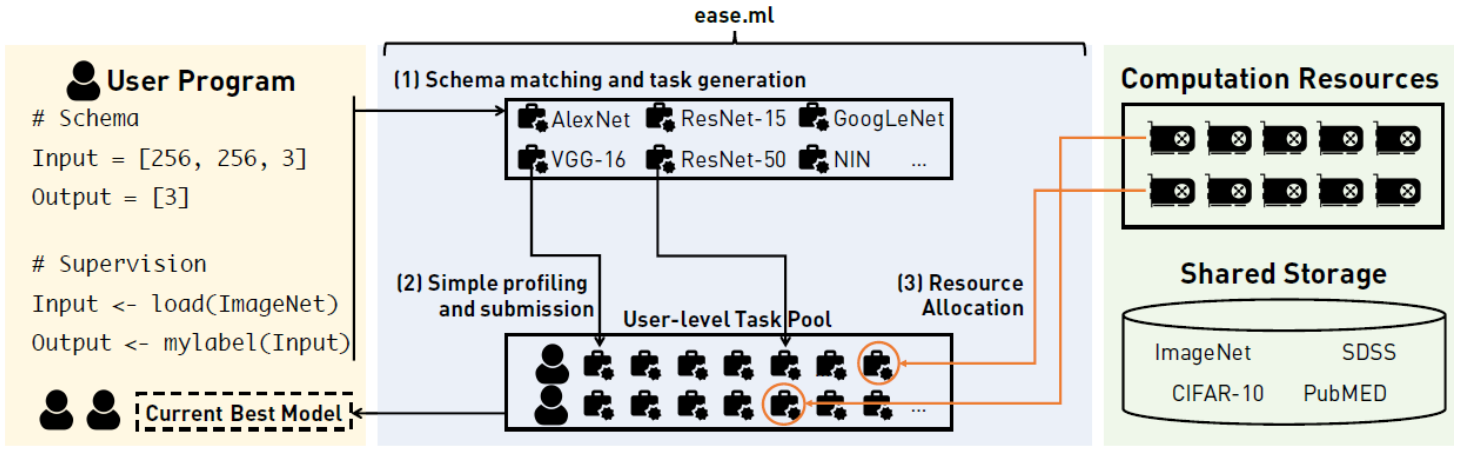


Figure 2: System architecture of `ease.ml`.

Type of Workload	Consistent Models
Image/Tensor Classification	AlexNet, ResNet, GoogLeNet, SqueezeNet, VGG, NIN, BN-AlexNet
Image/Tensor "Recovery"	Auto-encoder, GAN, pix2pix
Time Series Classification	RNN, LSTM, bi-LSTM, GRU

Model Templates

- **Ease.ml**
  - Goal: Select models to maximize avg. accuracy across all uses in a given cluster
  - Challenges: Global optimization objective, cost-aware scheduling
  - First translates user programs into system-data types (IR)
  - Then it generates candidate models via template matching
  - Cost-aware GP-UCB
  - Round robin scheduling policy; each users independently select the next model

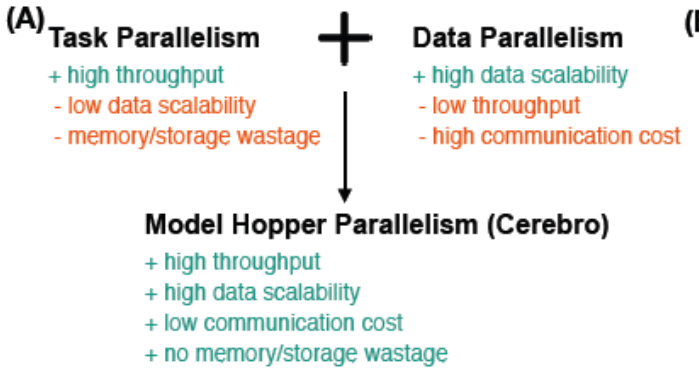
C_k = Execution Time of model k

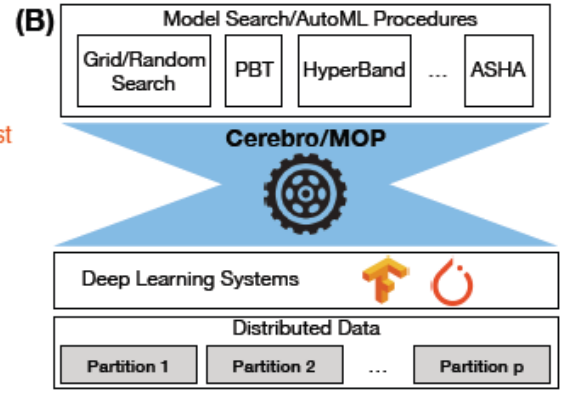
$\sigma_{t-1}(k)$ = Potential reward of model k

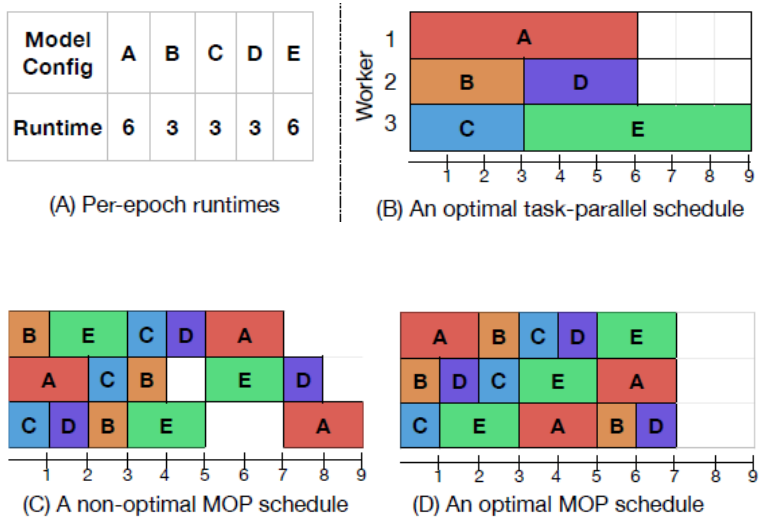
$$a_t \leftarrow \arg\max_{k \in [K]} \mu_{t-1}(k) + \sqrt{\beta_t / c_k}\,\sigma_{t-1}(k),$$

* Ease.ml: Towards Multitenant Resource Sharing for Machine Learning Workloads. PVLDB. 2018.

Cerebro: Deep Learning Model Selection







- **Cerebro**
  - Challenges: large datasets, low throughput (many models), low resource utilization (memory, transfer)
  - Common techniques are **task and data parallelism**, but they have bottlenecks
  - **Model Hopper Parallelism**: Combines data & task parallelism; each worker train a sub-epoch on local partition and a model hops from one worker to another after a sub-epoch; Repeat;
  - Exploit SGD property: any random ordering of examples suffices for convergence
  - MILP problem with randomized solution

* Cerebro: A Data System for Optimized Deep Learning Model Selection. PVLDB. 2020.

Feature Extraction and Transfer Learning

Transfer Learning

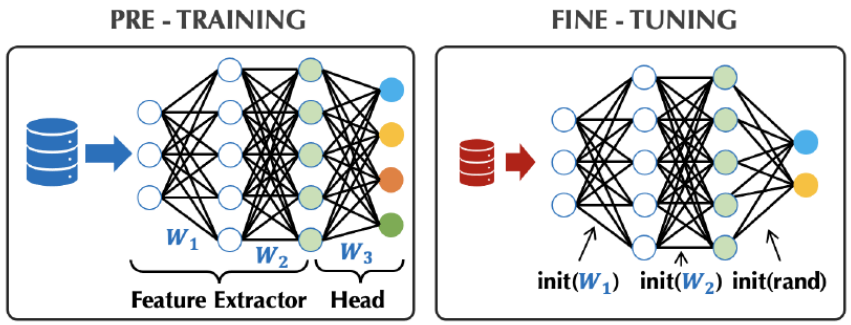
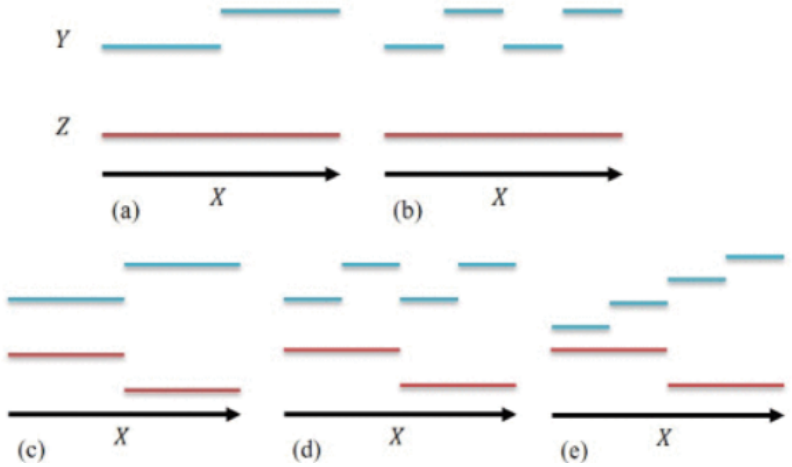


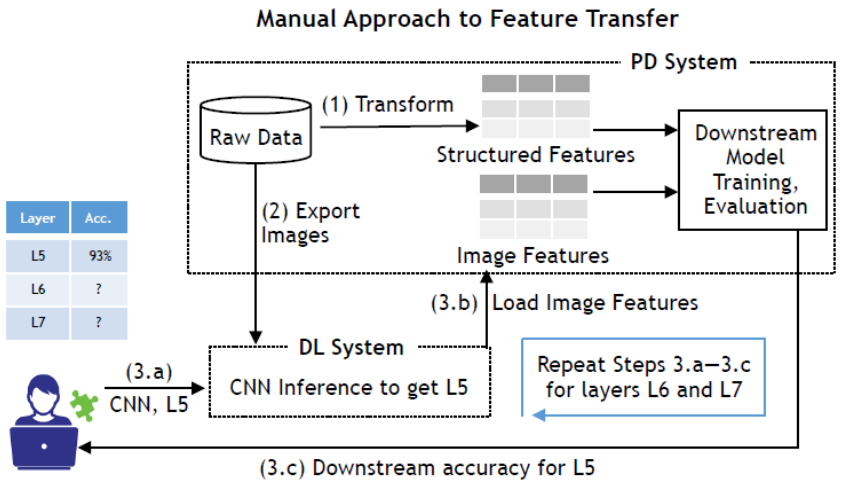
Figure 2: The difference between pre-training (left) and fine-tuning (right). The features stem from the last layer.

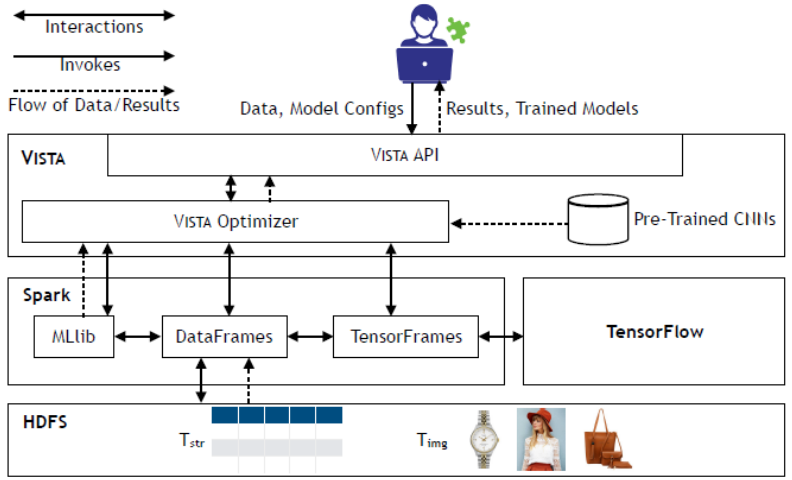


Transferability of a model trained for task T^Z to task T^Y

- **Motivation**
  - High-quality models with smaller dataset and reduced training time/cost
  - Extract features from a pre-trained model; train only the last layers
  - Extracted features can be joined with structured data
  - Challenges: repetitive feature extraction, model search
  - ML-based approach: **transferability estimation** using a proxy of linear classifier
  - Systems approach: optimization, reuse, incremental execution

VISTA: System for Declarative Feature Transfer



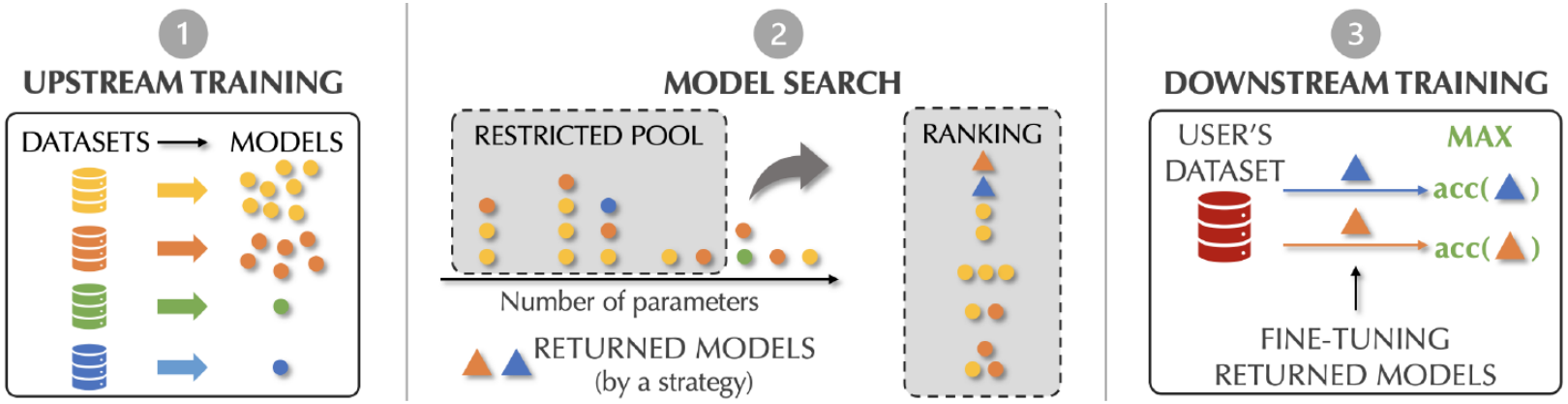


VISTA System Architecture

- **Overview**
  - Critical to try multiple layers for feature transfer because different layers yield different accuracies
  - Materialize all layers (high memory usage) vs. repeat for each layer (high computational redundancy)
  - Components of VISTA: declarative API, model pool with numbered layers, an optimizer
  - Optimization objective: minimize runtime subject to memory constraints

* VISTA: Optimized System for Declarative Feature Transfer from Deep CNNs at Scale. SIGMOD. 2020.

SHiFT: A Search Engine for Transfer Learning



- **Overview**
  - Problem: Finding the optimal pre-trained model for fine-tuning is expensive
  - Need an efficient model search strategy
  - SHiFT-QL query language to declarative model search
  - Successive-Halving: pulling an arm corresponds to running inference on more data
  - Reuse of intermediates and incremental execution for appended data

* SHiFT: An Efficient, Flexible Search Engine for Transfer Learning. VLDB. 2022.

Exercise 2: Transfer Learning

Transfer Learning

- **Motivation**
  - Does pre-trained model on one classification task helps in another task?
- **Data Prep and Training from Scratch**
  - Prepare and train ResNet18 on CIFAR-10 subset
- **Full Fine-tuning from ImageNet Weights**
  - Fine-tune ImageNet on the same dataset

```
2_Transfer_learning/
├── requirements.txt          # or pyproject.toml, uv.lock, etc
├── solution.ipynb            # main notebook orchestrating the pipeline (calling the scripts, producing plots)
├── data/
│   ├── raw/                  # CIFAR-10 as downloaded by torchvision
│   └── subset.json           # reproducible train/test index split
├── models/
│   └── <run_id>.pt           # one checkpoint per training run
├── registry/
│   └── <run_id>.json         # one metadata file per training run (extends Exercise 1's schema)
└── scripts/
    ├── prepare_data.py       # download CIFAR-10 and materialize the subset
    ├── train.py              # parametrized trainer: --init, --freeze, --epochs, --amp, --cache-features, …
    └── evaluate.py           # aggregate registry/*.json into a comparison table
```

- **Tune Transfer Learning Policy**
  - Vary freeze depth (early features/most layers/linear probe)
  - Compare accuracy and execution time
- **Optimizations**
  - Propose at least 2 optimizations to reduce execution time
  - Implement at least one and report speedups

Deadline: 18.05.2026

Summary

- AutoML systems and optimizations
- Hyperparameter tuning techniques and trade-offs
- Model selection techniques
- Feature extraction and redundancy elimination
- Model selection for transfer learning

Please read the recommended papers