Hasso Plattner Institut
Digital Engineering · Universität Potsdam

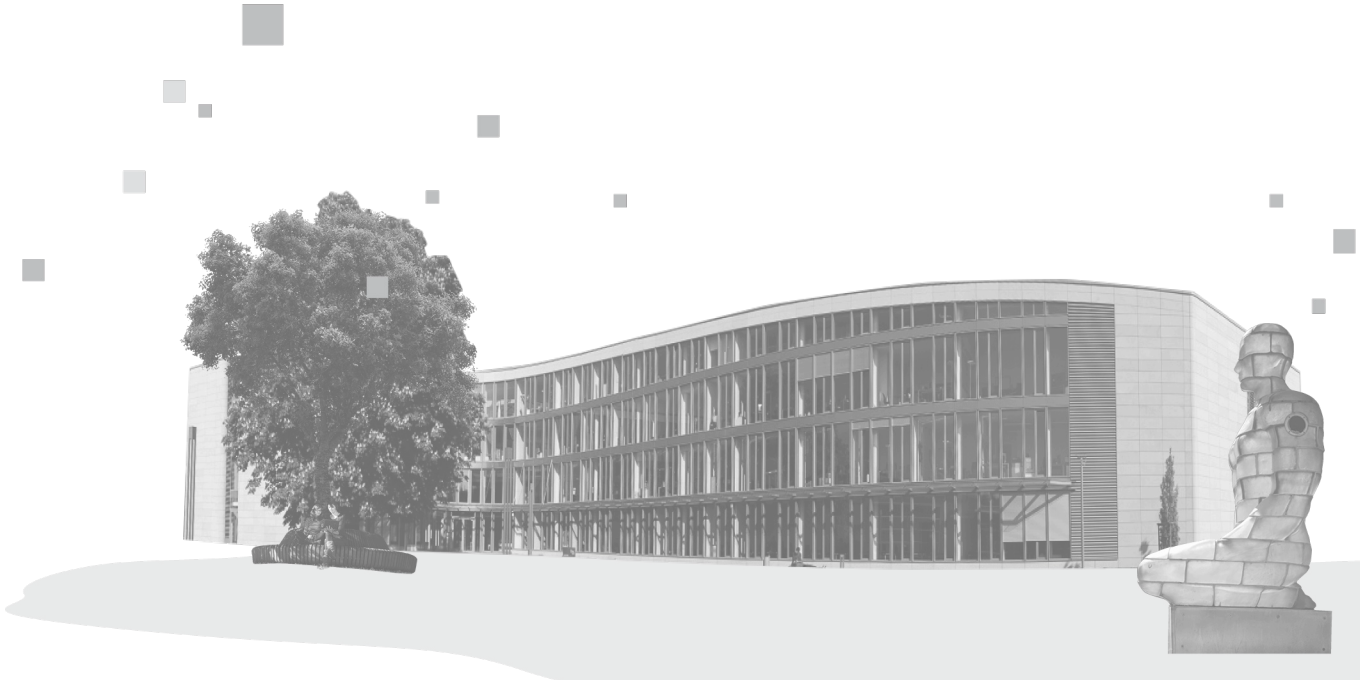
Machine Learning Model Management

Nils Strassenburg

Lecture 5 - Machine Learning Model Management and Inference

Design IT.
Create Knowledge.

www.hpi.de



Nils Strassenburg

Born and Raised in Lüneburg

Bachelor's in Computer Science @ Uni Hamburg

Master's in IT-Systems Engineering @ HPI

- Internship
- Visiting semester

PhD in Computer Science @ HPI

- Focus on Data Management for ML Systems
- Supervisor Tilmann Rabl

Part 1: ML Deployment

Lecture 2: Model Management Systems [April 22]

- Model and data versioning
- Lineage tracing
- Feature stores
- Readings: ModelDB, ModelHub, DataHub, LIMA, FeathrPO
- Slides

Lecture 3: ML Platforms and Deployment [April 29]

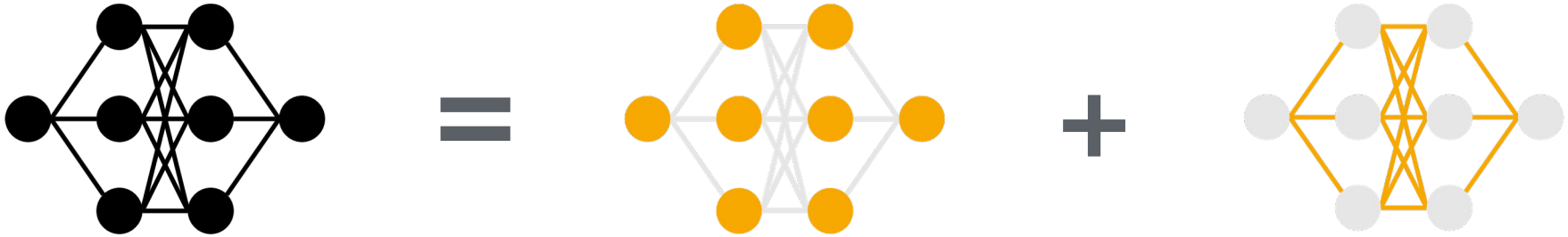
- MLOps
- End-to-end ML platforms and lifecycle management
- Continuous retraining
- Data validation
- Readings: TFX, MODYN, AWS Deequ
- Slides

Lecture 4: Model Selection Systems [May 06]

- Model selection systems
- Transfer learning
- Readings: Hyperband, Cerebro, VISTA, SHiFT
- Slides

Definition of a Model

In this talk: **model == neural network**



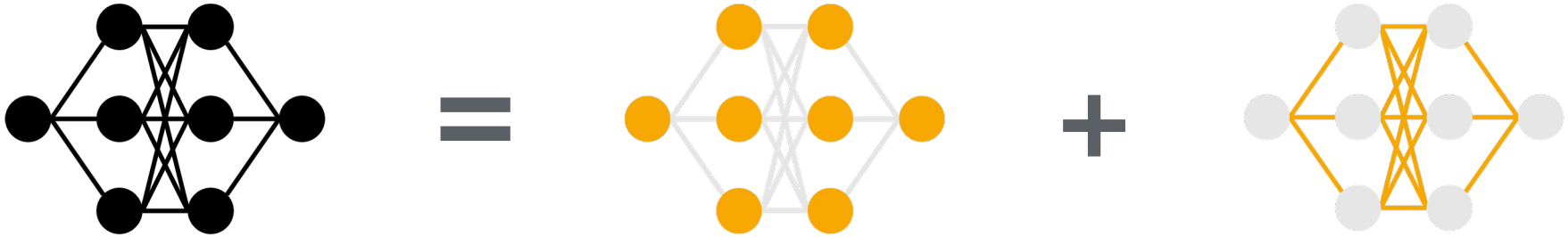
Model
Representation of learned

Architecture
Computational structure

Parameters
Weights and biases

Definition of a Model

In this talk: **model == neural network**



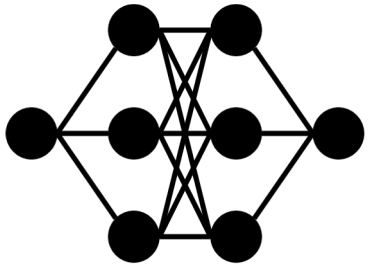
Model
Representation of learned

Architecture
Computational structure

Parameters
Weights and biases

Definition of a Model

In this talk: **model == neural network**



Model
Representation of learned

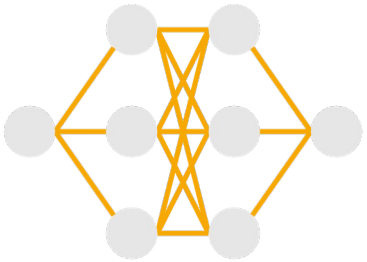
```python
import torch.nn as nn

class SimpleNet(nn.Module):

def __init__(self):
super().__init__()
# "fc1.weight", "fc1.bias"
self.fc1 = nn.Linear(784, 128)
# "fc2.weight", "fc2.bias"
self.fc2 = nn.Linear(128, 10)

def forward(self, x):
x = nn.functional.relu(self.fc1(x))
return self.fc2(x)
```

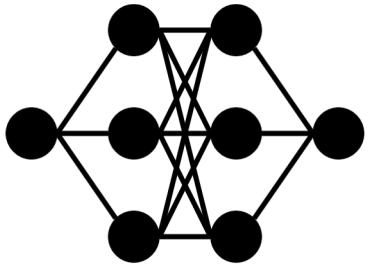
+



Parameters
Weights and biases

Definition of a Model

In this talk: **model == neural network**



Model
Representation of learned

```python
import torch.nn as nn

class SimpleNet(nn.Module):

def __init__(self):
super().__init__()
# "fc1.weight", "fc1.bias"
self.fc1 = nn.Linear(784, 128)
# "fc2.weight", "fc2.bias"
self.fc2 = nn.Linear(128, 10)

def forward(self, x):
x = nn.functional.relu(self.fc1(x))
return self.fc2(x)
```

+

state_dict = OrderedDict({

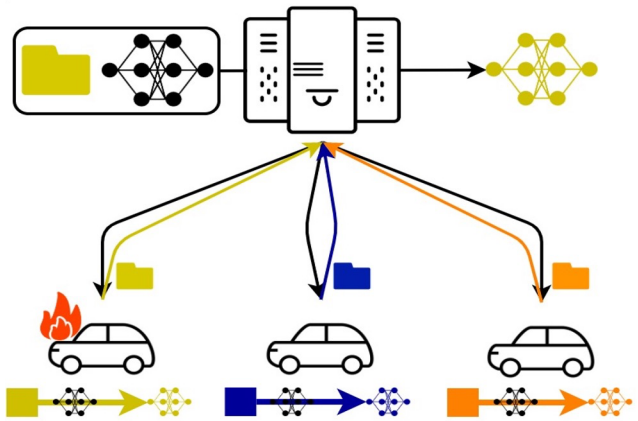
key (str)	value (torch.Tensor)
"fc1.weight" Linear layer 1 weights	tensor([[...]]) shape: [128, 784] dtype: float32
"fc1.bias" Linear layer 1 bias	tensor([...]) shape: [128] dtype: float32
"fc2.weight" Linear layer 2 weights	tensor([[...]]) shape: [10, 128] dtype: float32
"fc2.bias" Linear layer 2 bias	tensor([...]) shape: [10] dtype: float32

})

"module.parameter" naming convention — actual parameter values stored on CPU or GPU

Model Management

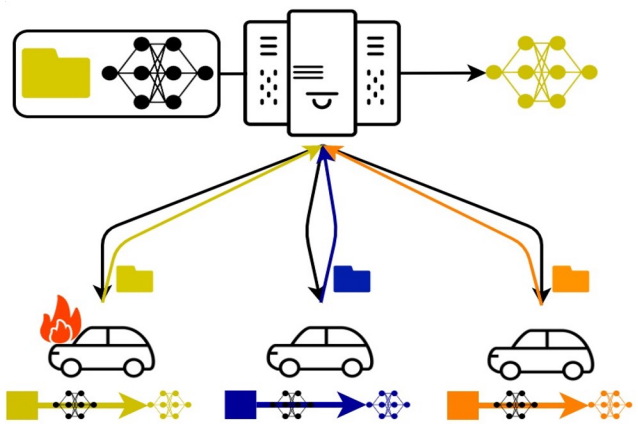
Model Management for Archival



- Efficiently manage many DL models in a distributed environment
- Assumption: save often, read rarely
- MMlib and M3lib

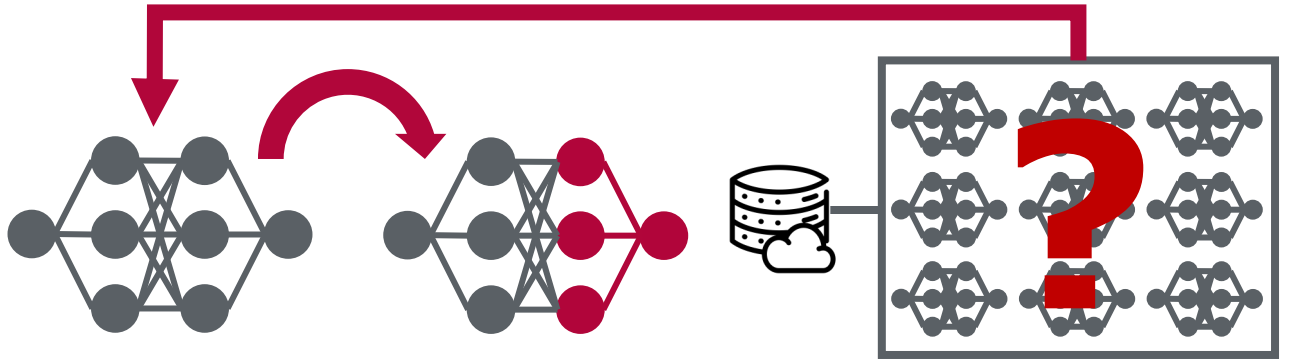
Model Management

Model Management for Archival



- Efficiently manage many DL models in a distributed environment
- Assumption: save often, read rarely
- MMlib and M3lib

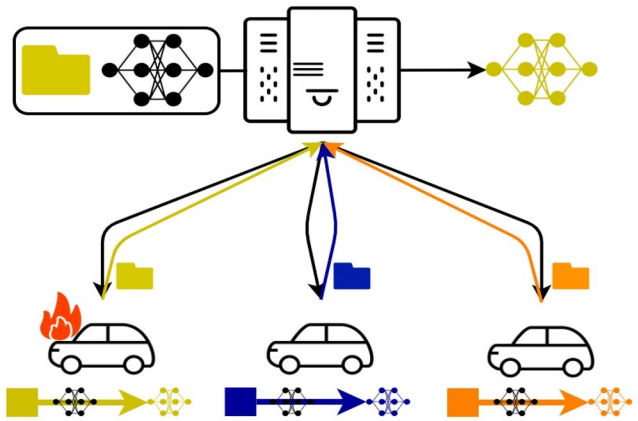
Model Management for Reuse



- Efficiently access models to find the best candidate for transfer learning
- Assumption: read often, models overlap
- Poodle and Alsatian

Model Management

Model Management for Archival



- Efficiently manage many DL models in a distributed environment
- Assumption: save often, read rarely
- MMlib and M3lib

Authors

Nils Strassenburg[1]

Ilin Tolovski[1]

Tilmann Rabl[1]

Dominic Kupfer[2]

Julia Kowal[2]

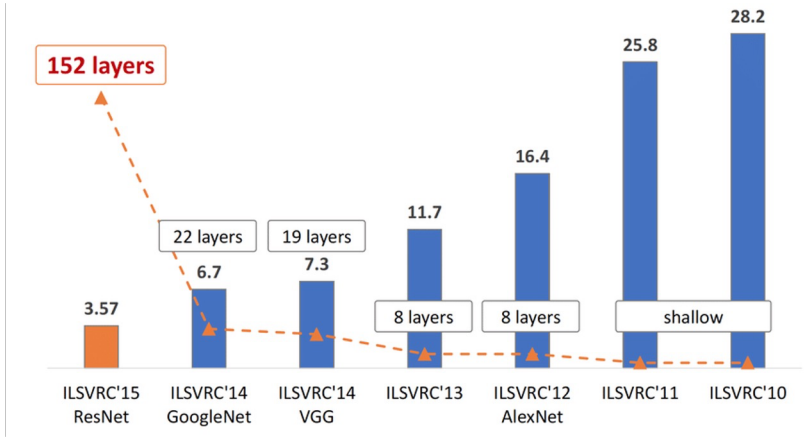
1 HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

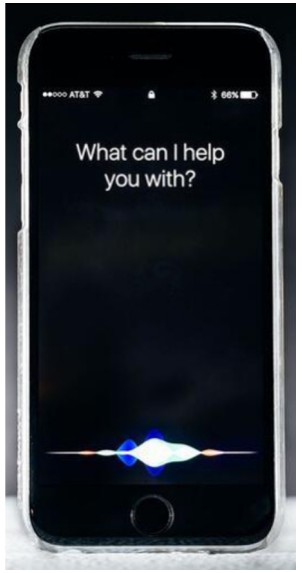
firstname.lastname@hpi.de

2 TECHNISCHE UNIVERSITÄT BERLIN

firstname.lastname@tu-berlin.de

ML Use Cases





[The evolution of the winning entries on the ImageNet Large Scale Visual Recognition Challenge]

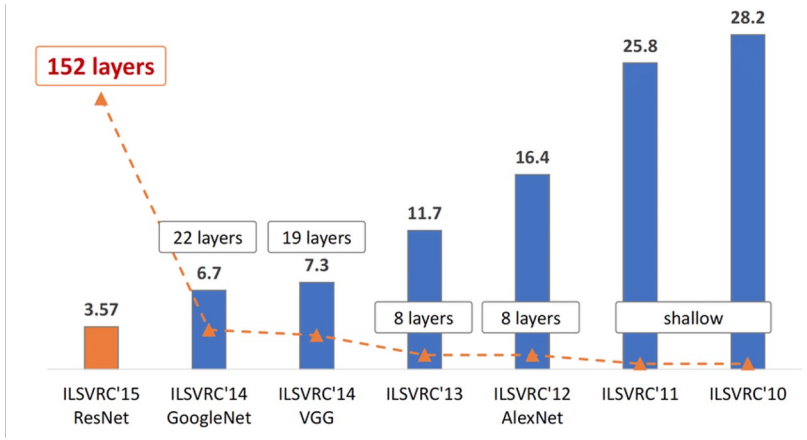
[Apple's Siri: A cheat sheet]

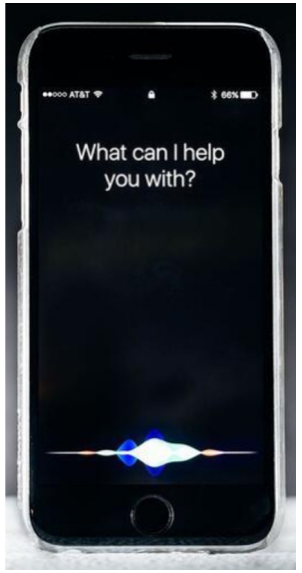
[Eyes on the Road: How Autonomous Cars Understand What They're Seeing]

[The limitations of deep learning]

[Opinion: Why driverless cars will force an insurance u-turn]

ML Use Cases





The boy is holding a baseball bat.

[The evolution of the winning entries on the ImageNet Large Scale Visual Recognition Challenge]

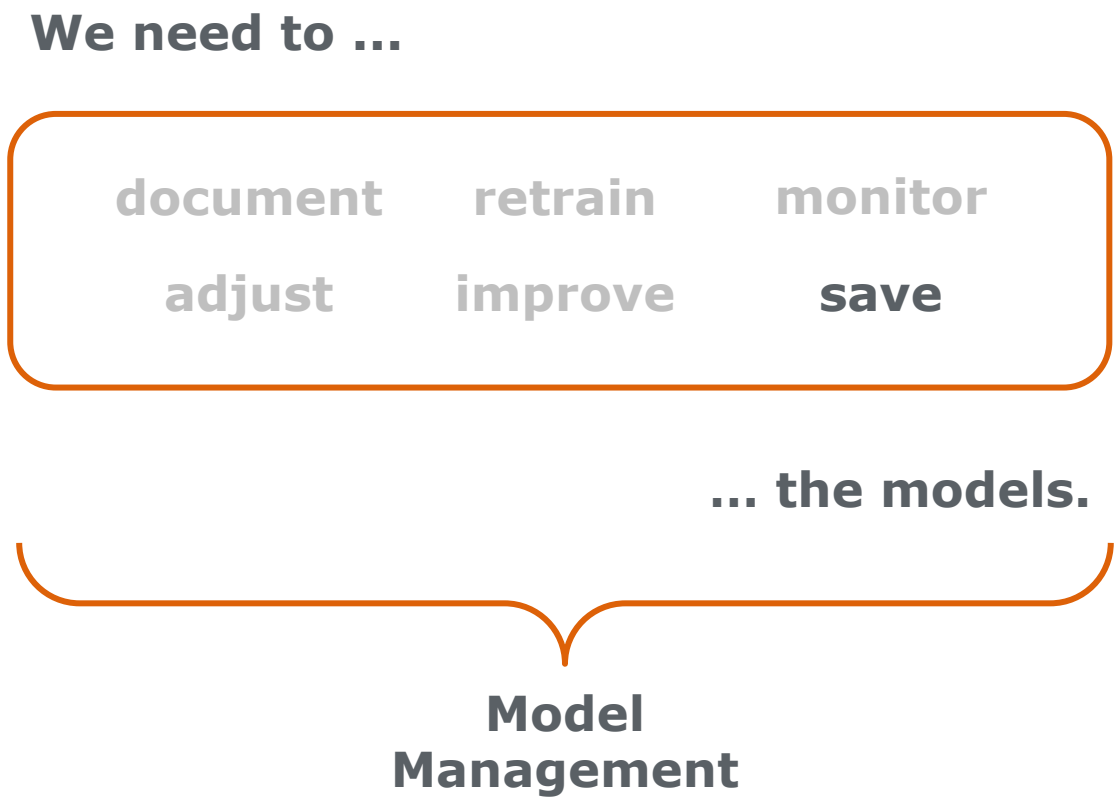
[Apple's Siri: A cheat sheet]

[Eyes on the Road: How Autonomous Cars Understand What They're Seeing]

[The limitations of deep learning]

[Opinion: Why driverless cars will force an insurance u-turn]

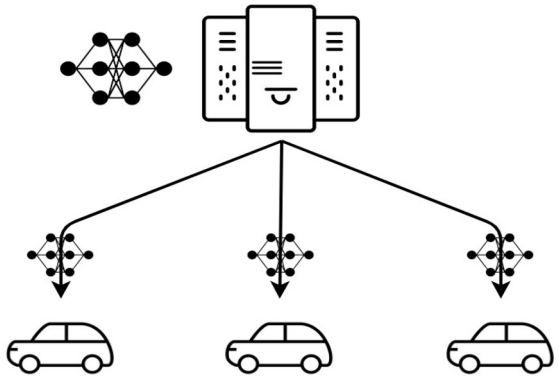
Model Management



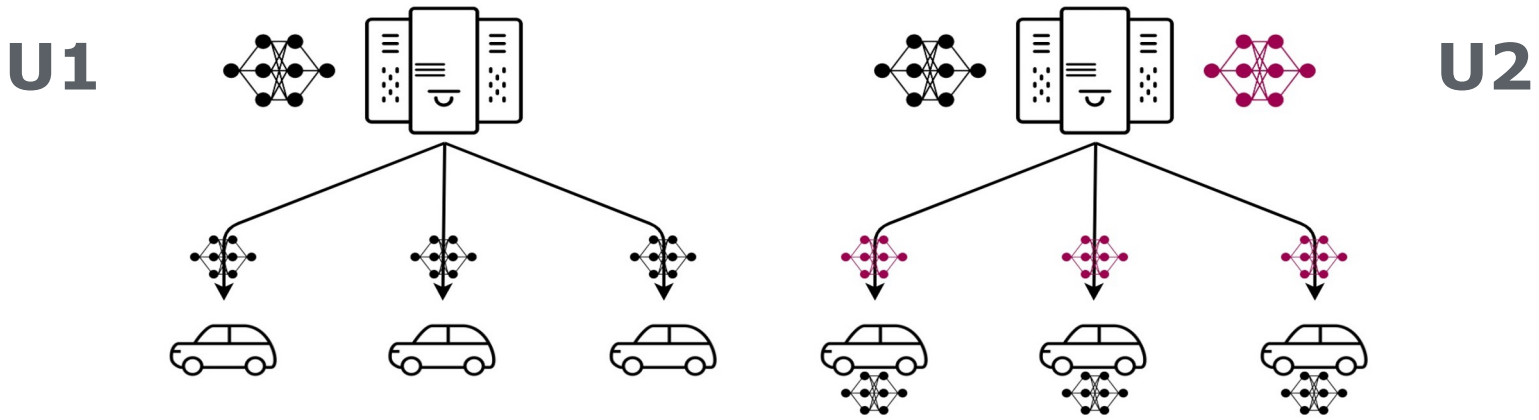
Schelter et al.: On challenges in machine learning model management

Running Example

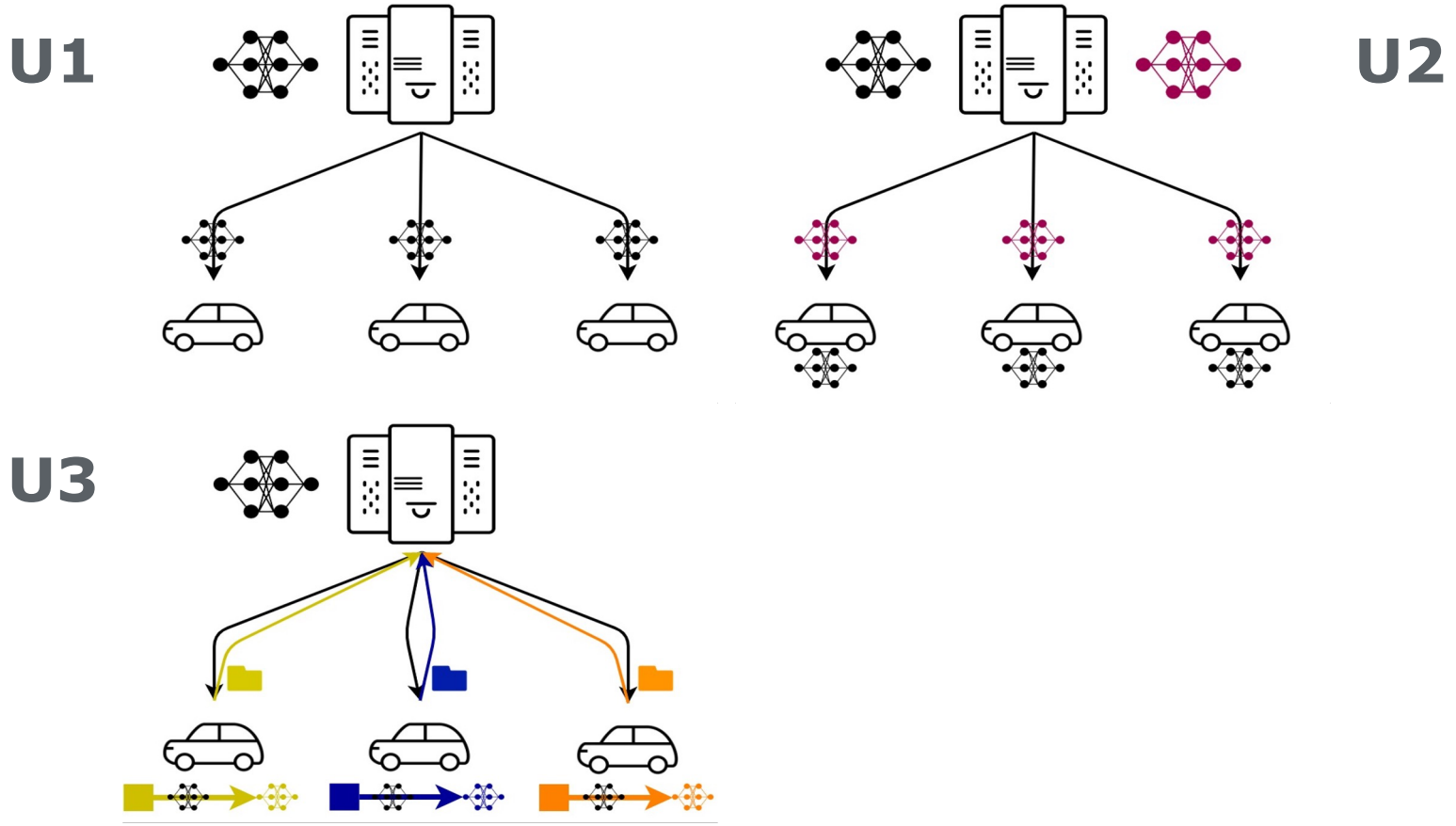
U1



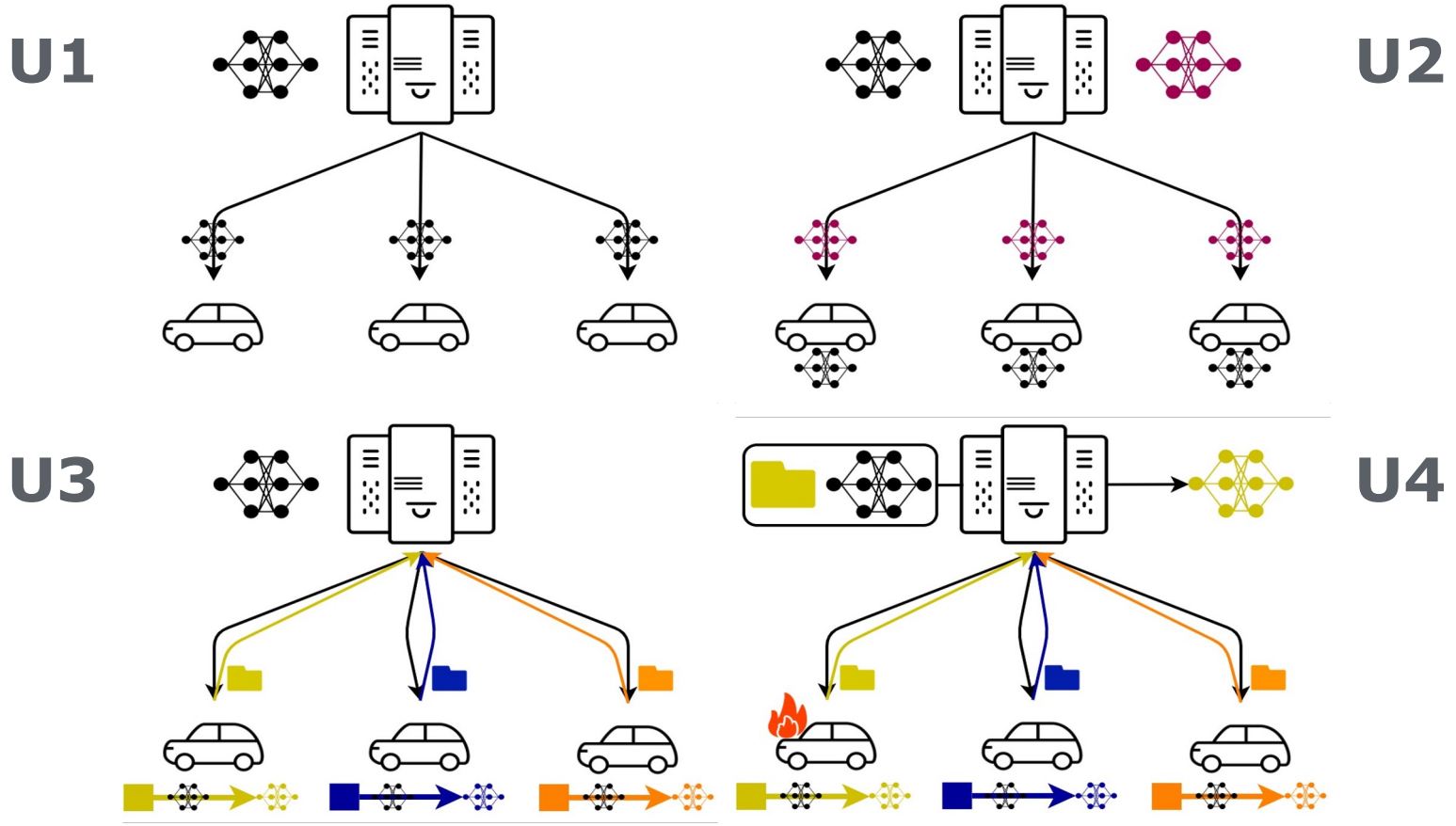
Running Example



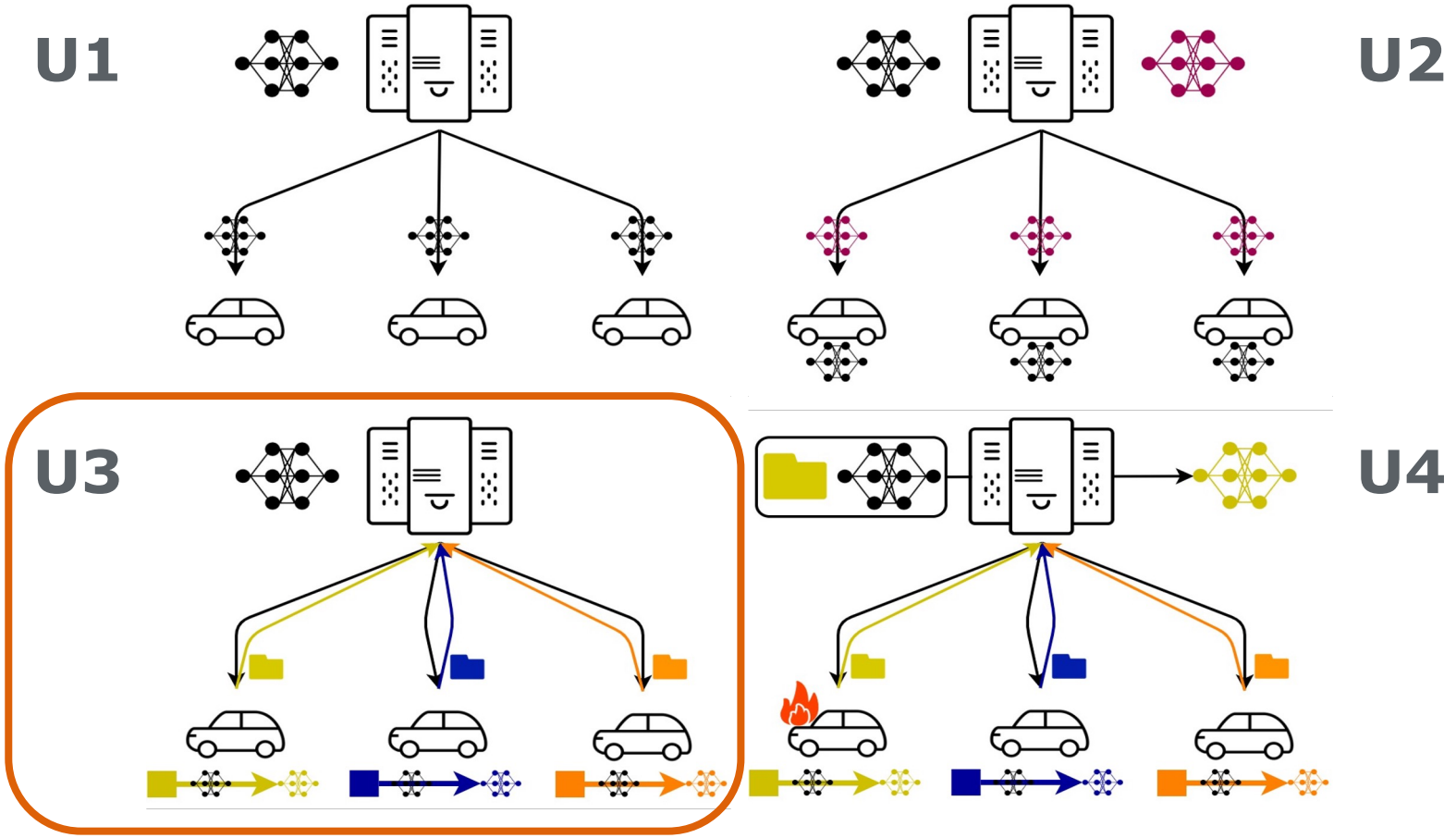
Running Example



Running Example



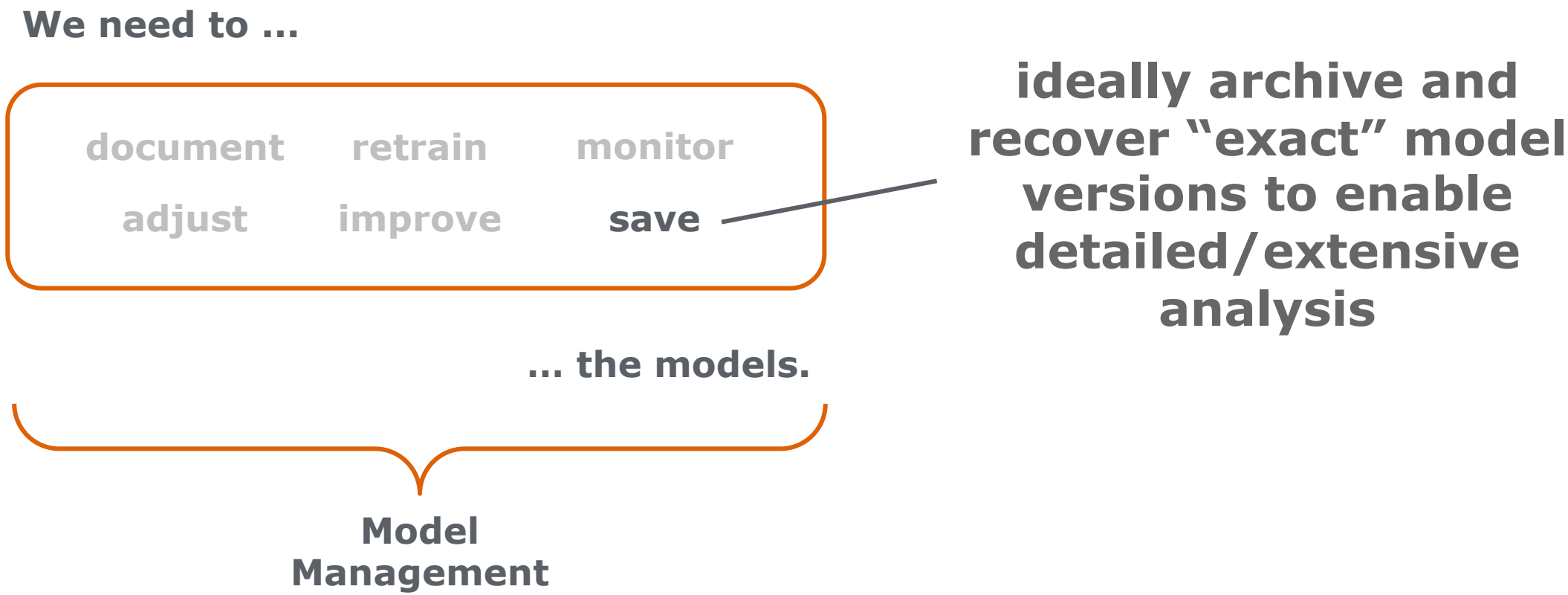
Running Example



Assumption:

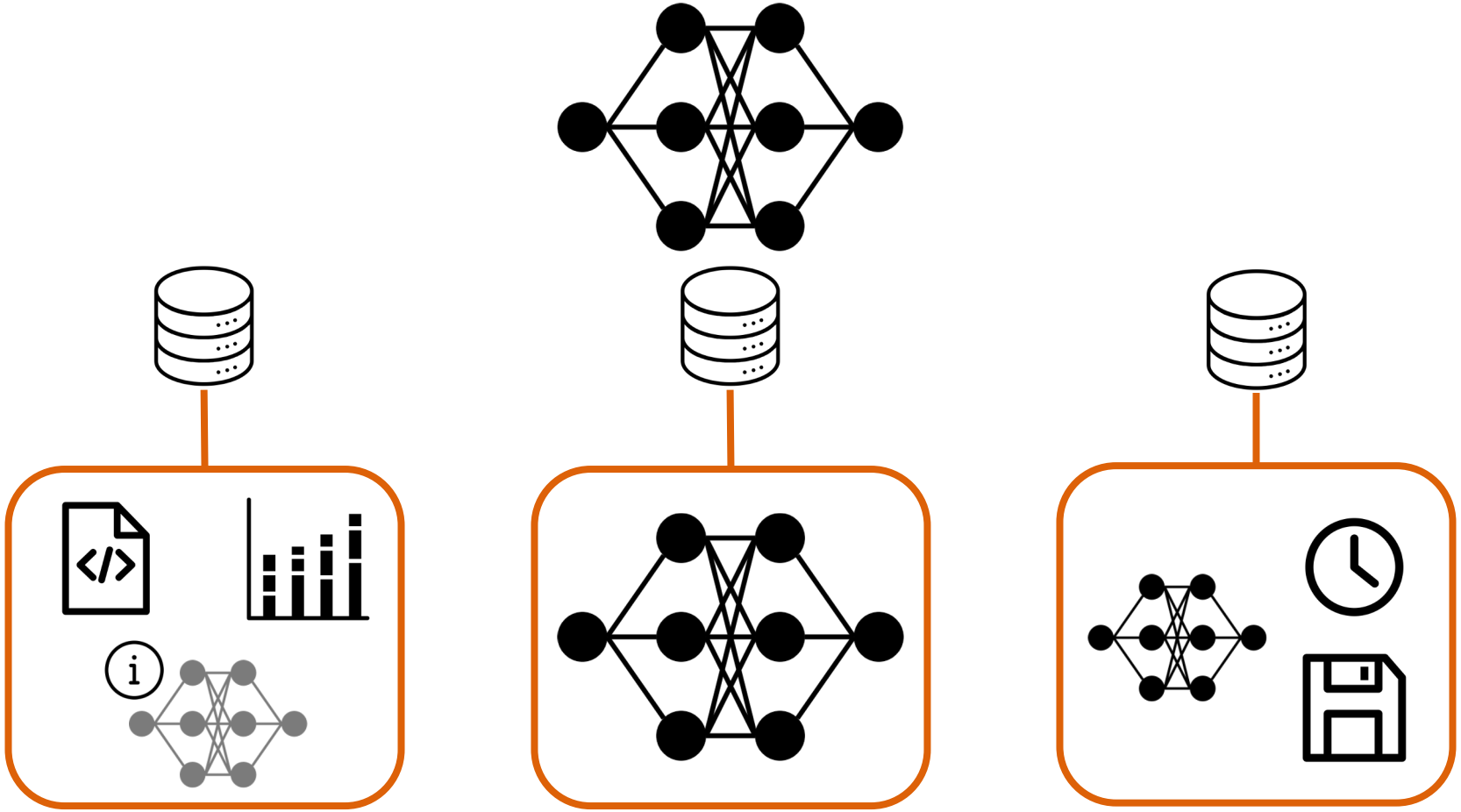
- Save models often
- Recover models rarely

Model Management

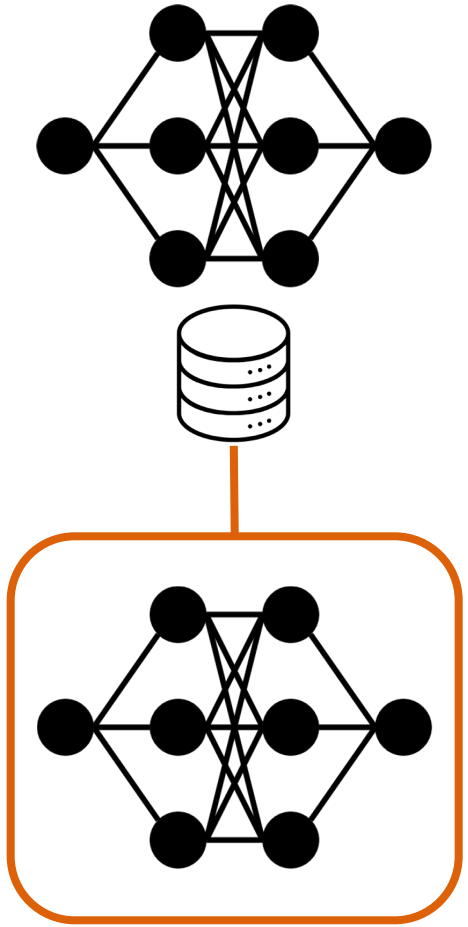


Schelter et al.: On challenges in machine learning model management

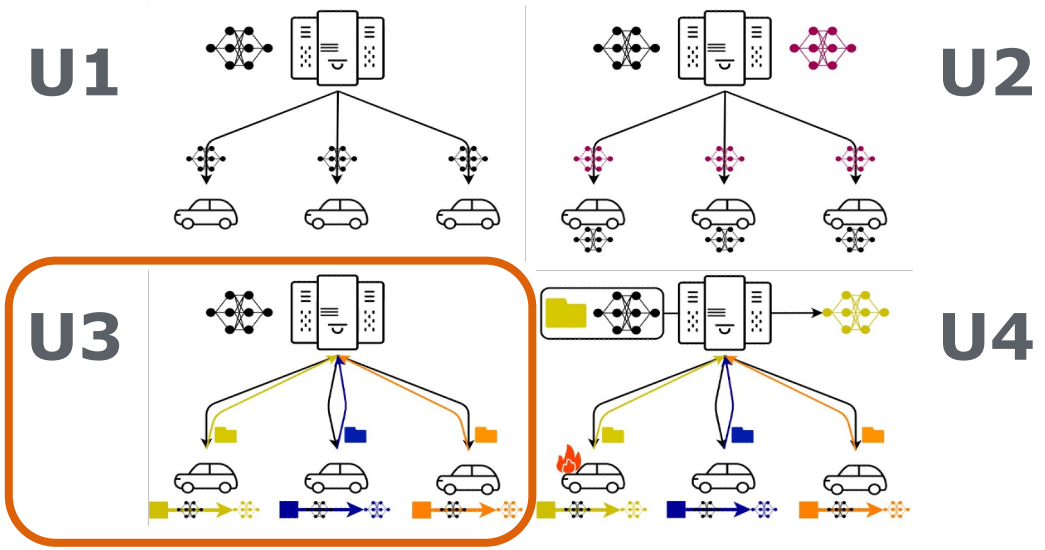
Related Work



Research Question

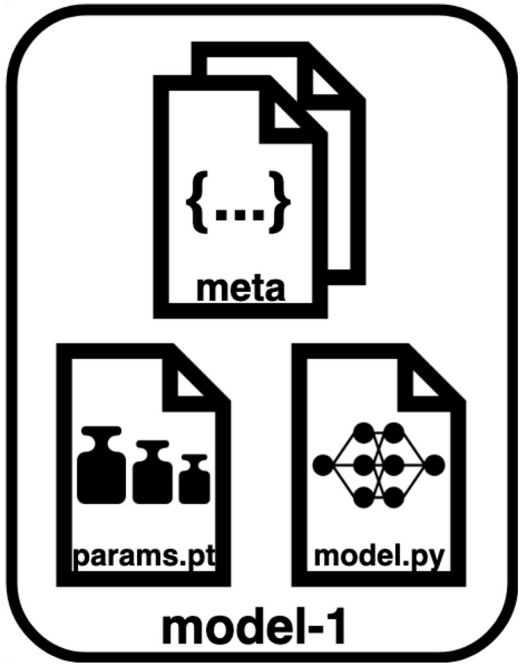


Complete Snapshot

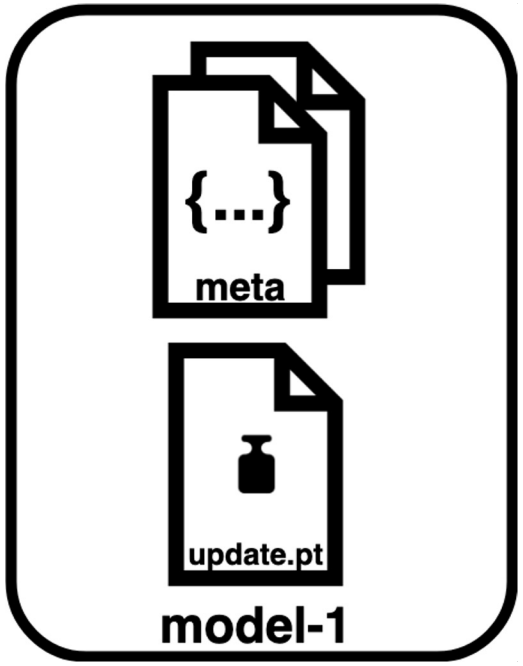


(How) Can we do better than a baseline approach that saves every model individually?

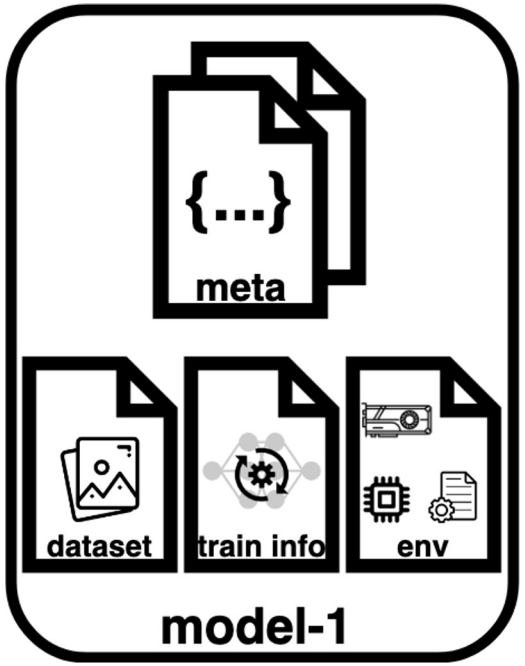
MMlib Approaches



Baseline

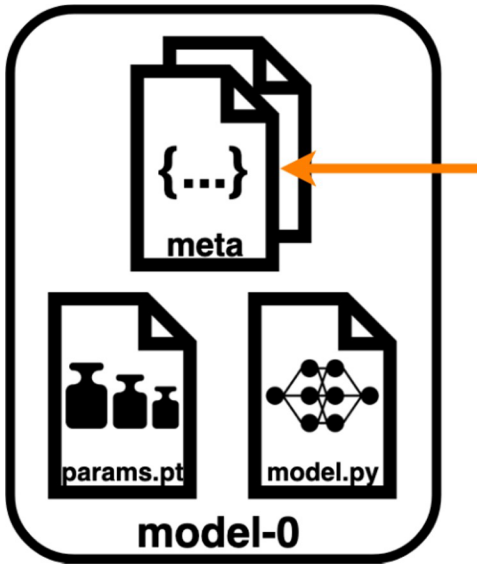


Parameter Update

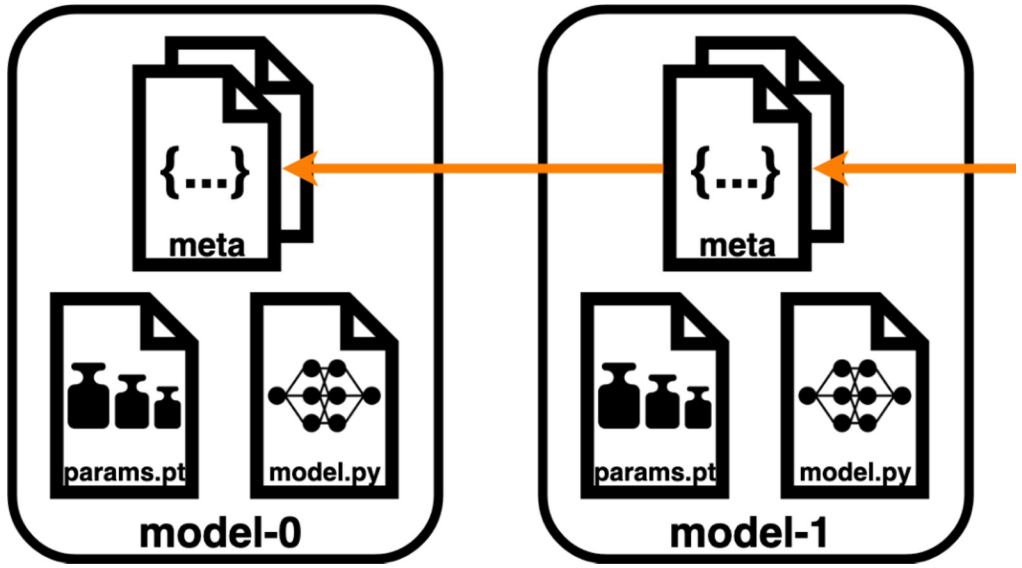


Provenance

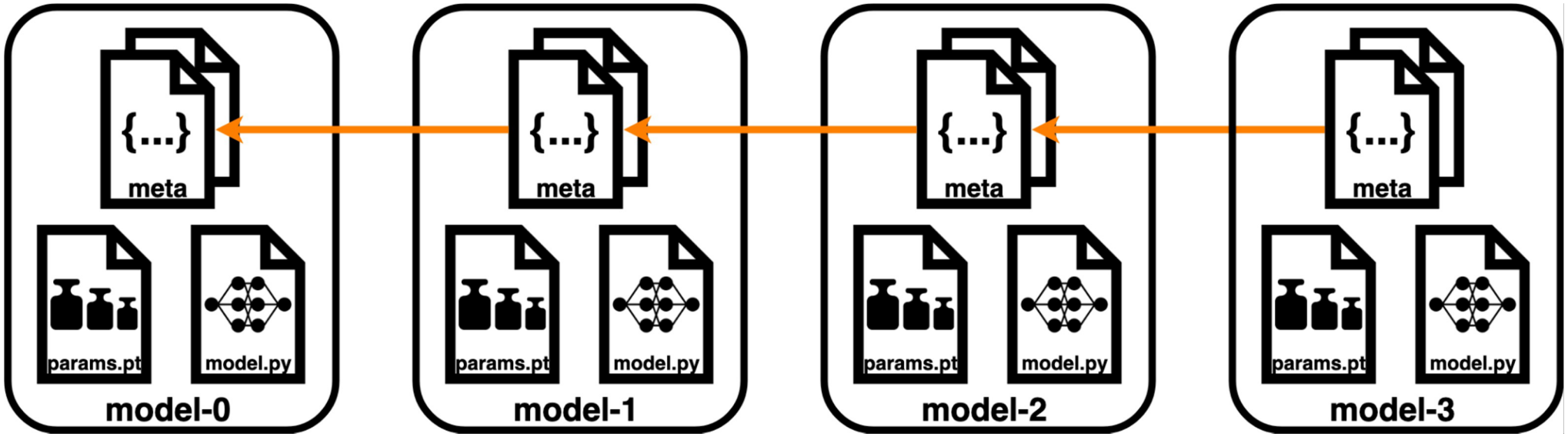
MMlib: Baseline



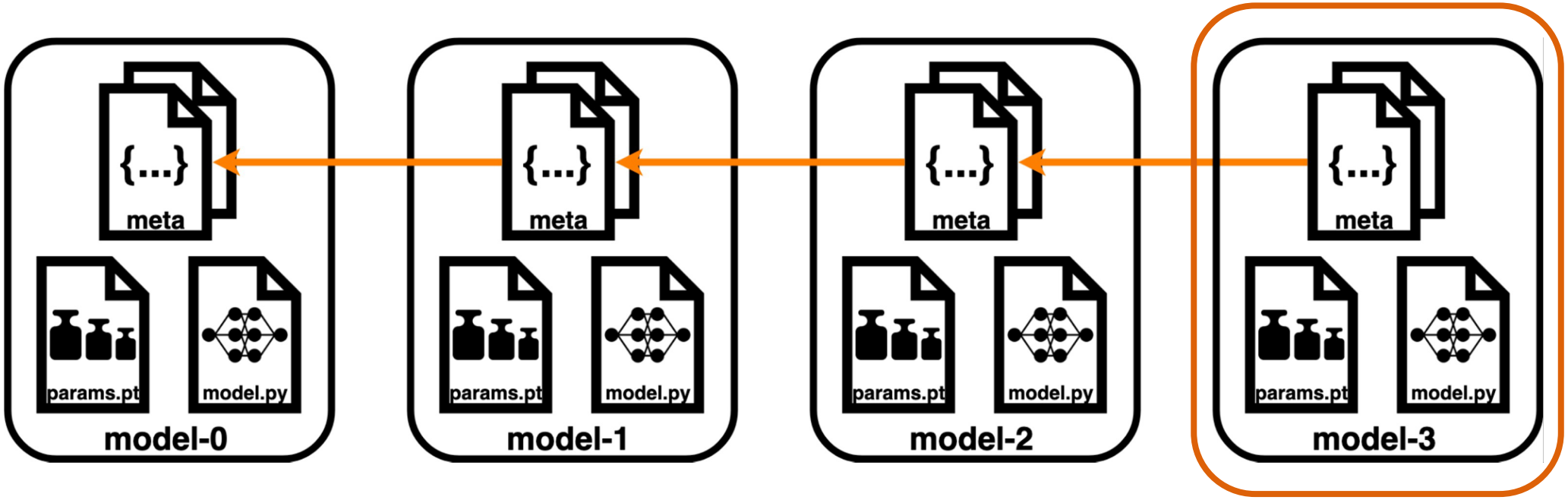
MMlib: Baseline



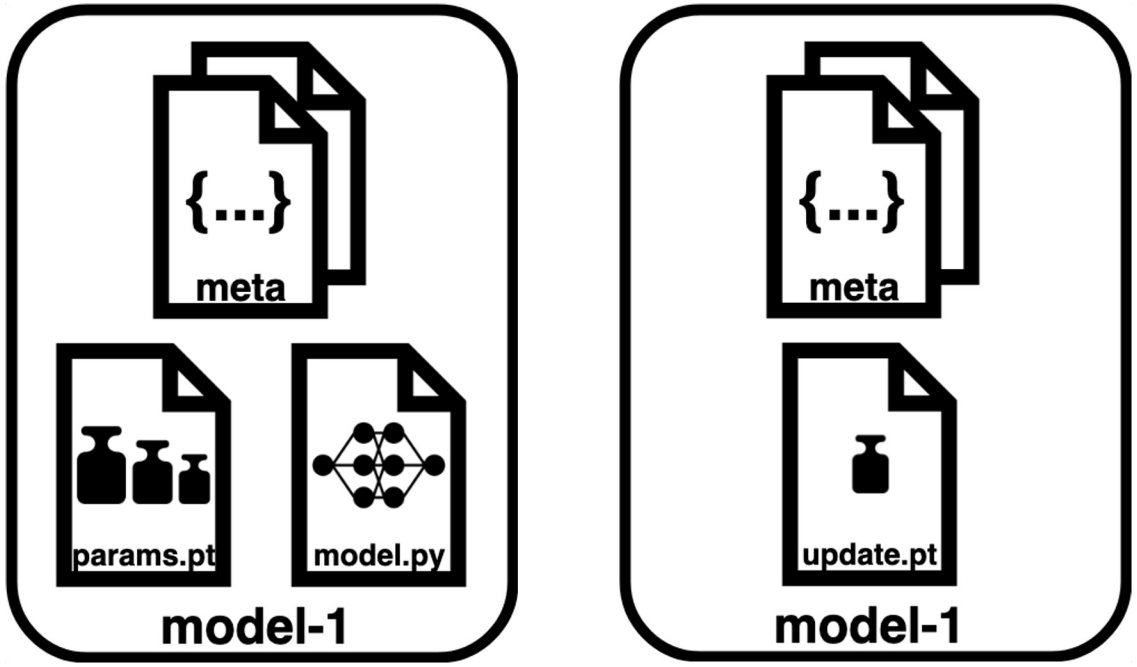
MMlib: Baseline



MMlib: Baseline



MMlib Approaches



Baseline

Parameter Update

How Models Are Updated

19

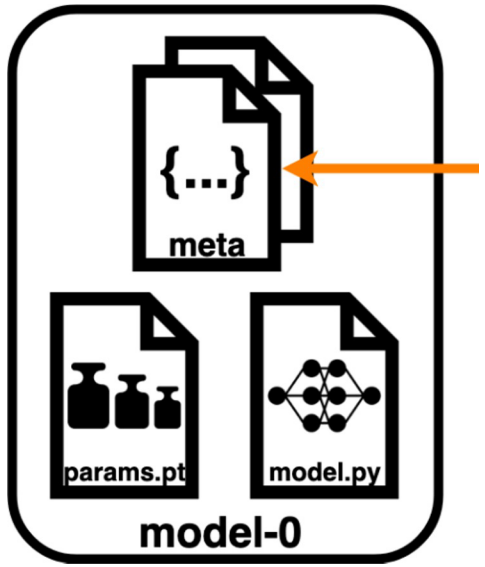
Transfer Learning

- **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
- **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

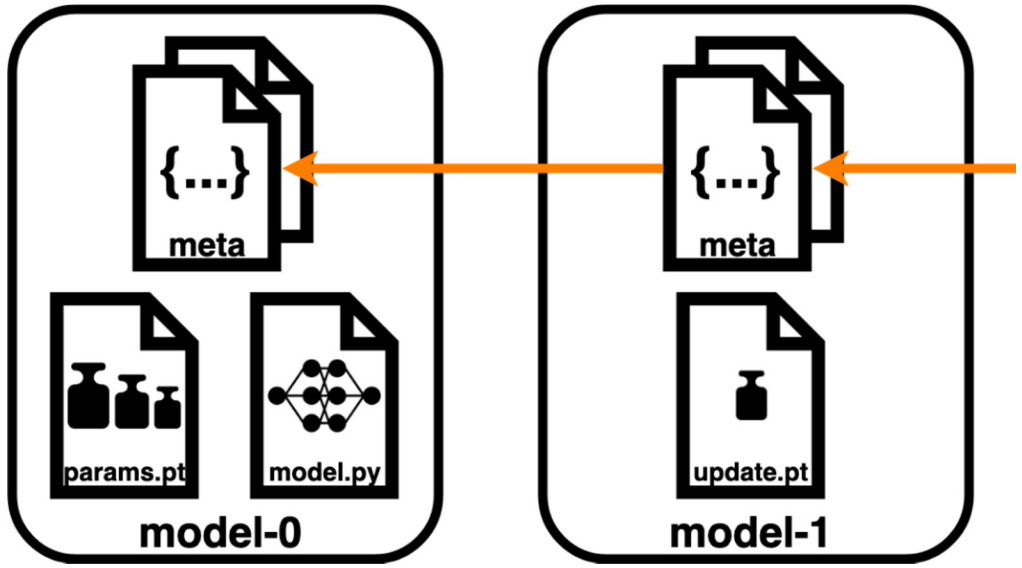
```
2_Transfer_learning/
├── requirements.txt          # or pyproject.toml, uv.lock, etc
├── solution.ipynb            # main notebook orchestrating the pipeline (calling the scripts, producing plots)
├── data/
│   ├── raw/                  # CIFAR-10 as downloaded by torchvision
│   └── subset.json           # 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
```

Deadline: 18.05.2026

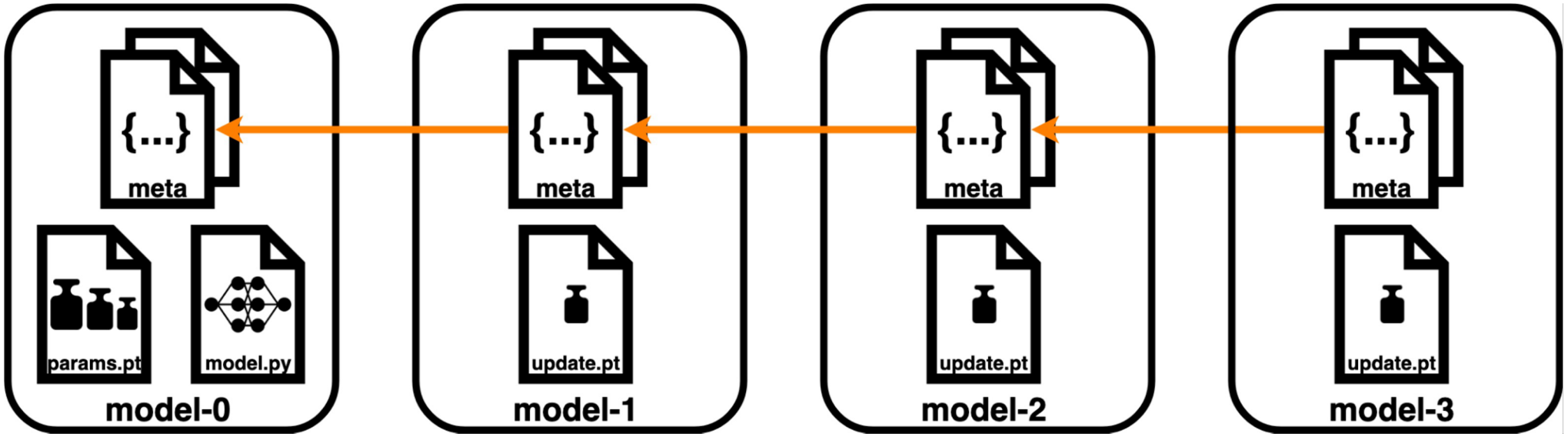
MMlib: Parameter Update



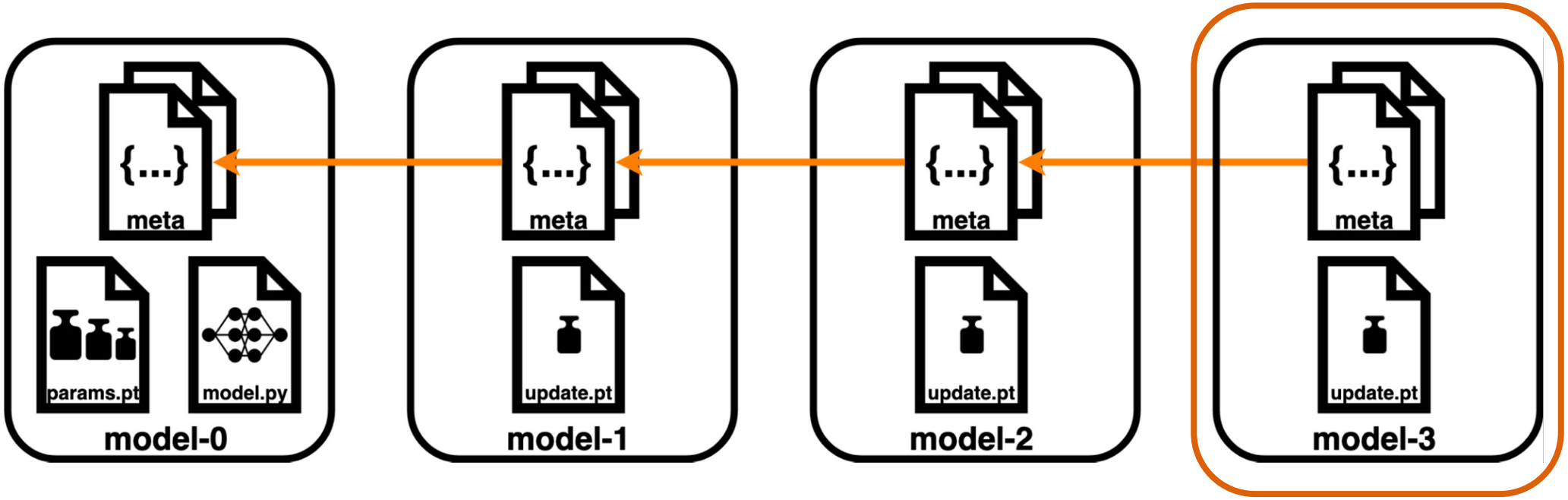
MMlib: Parameter Update



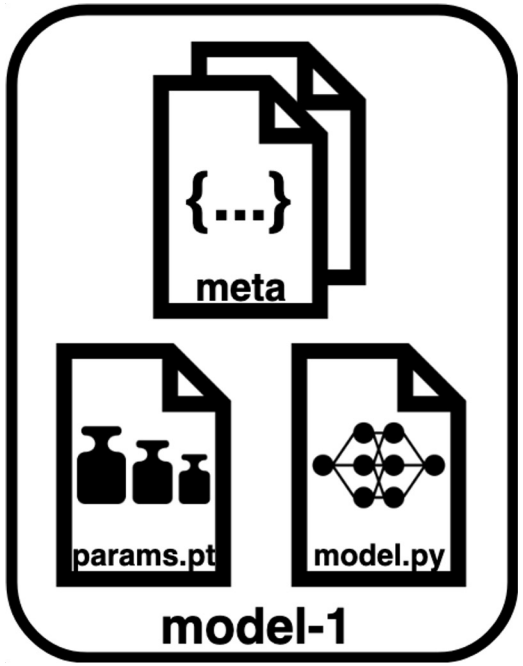
MMlib: Parameter Update



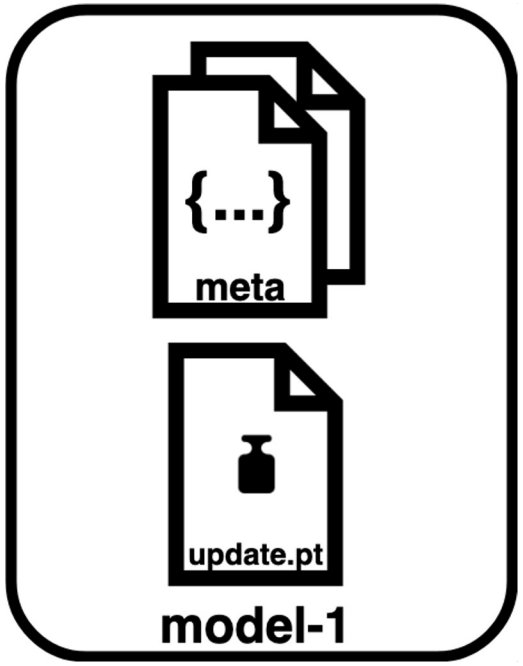
MMlib: Parameter Update



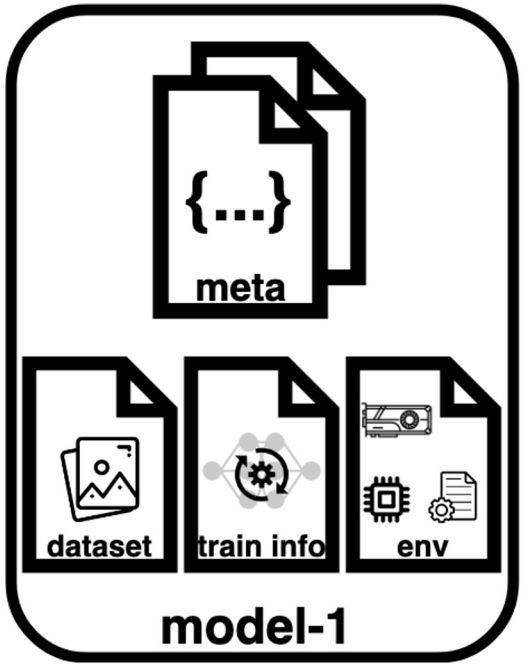
MMlib Approaches



Baseline

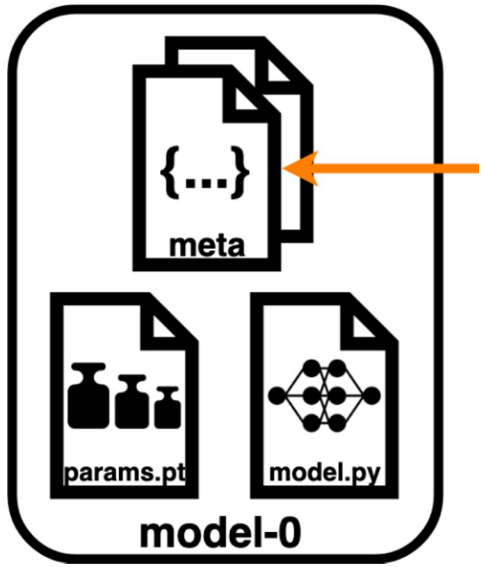


Parameter Update

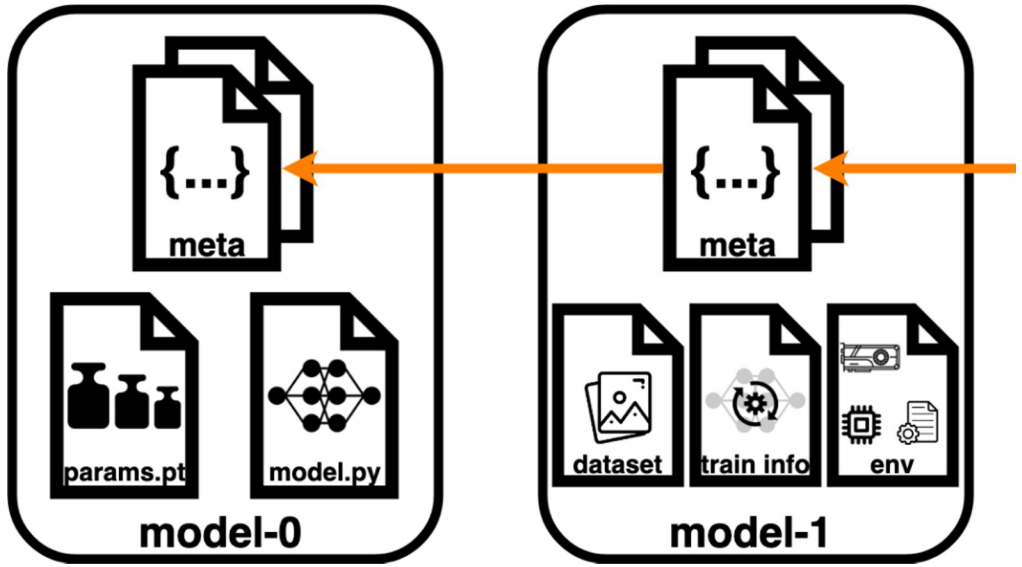


Provenance

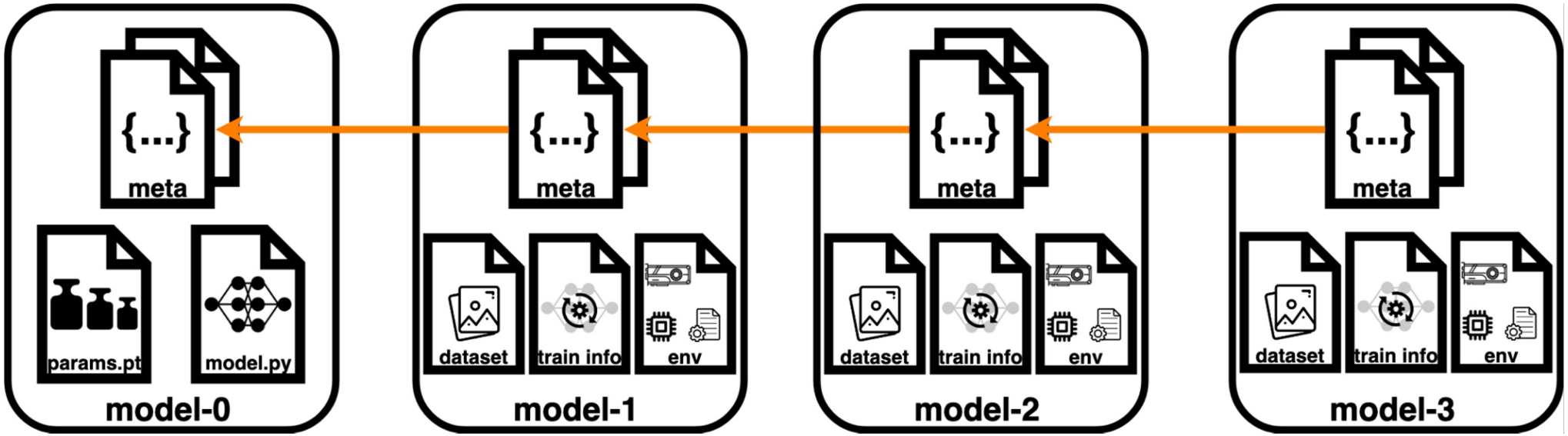
MMlib: Provenance



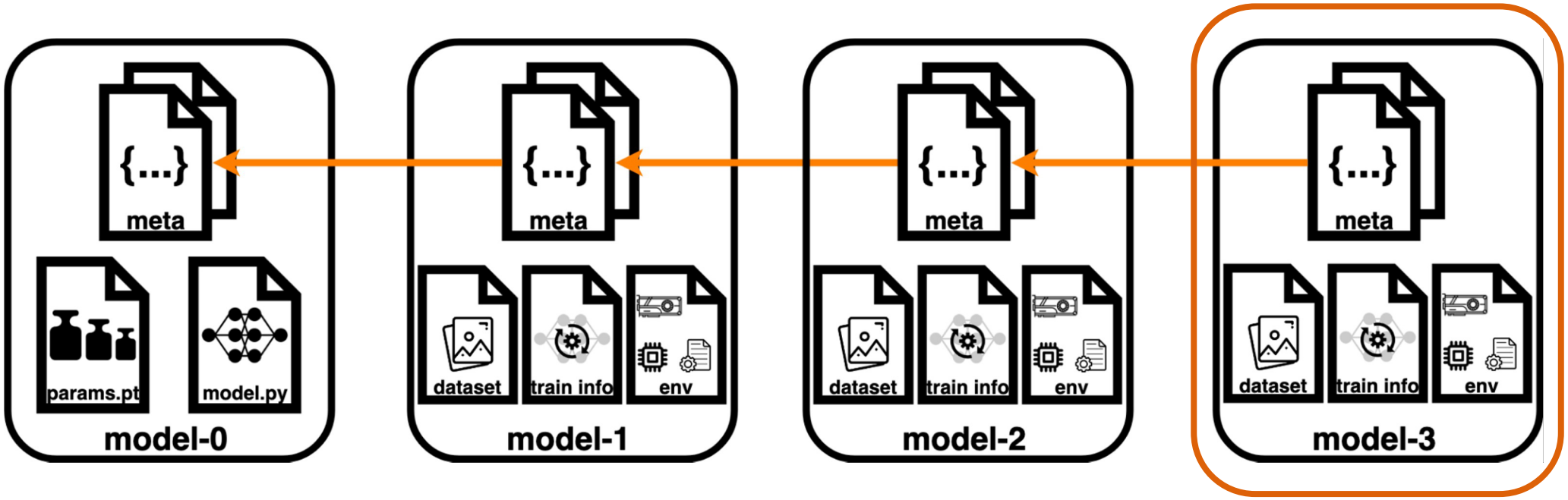
MMlib: Provenance



MMlib: Provenance



MMlib: Provenance



recover exact model ?!?

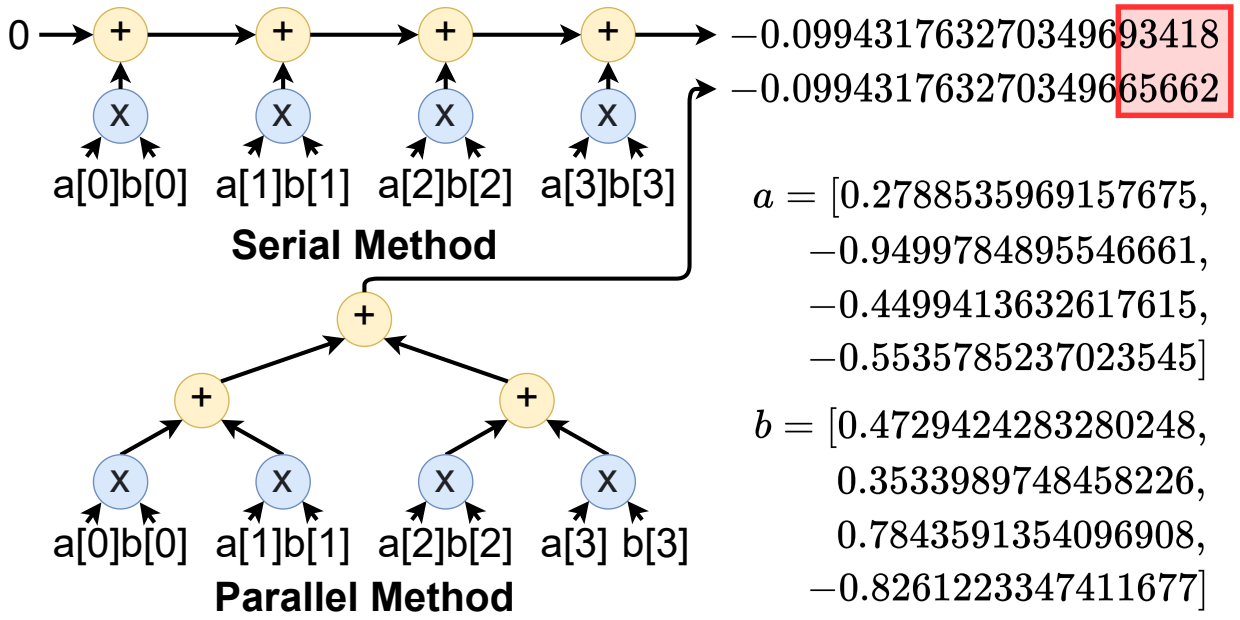
MMlib: Provenance

Other summary is: diff

l_name	fwd_idx	inp	out	grad_inp	grad_out
Conv2d	186	same	same	diff	diff
BNorm2d	187	same	same	diff	diff
BcConv2d	188	same	same	diff	diff
MaxPool2d	189	same	same	diff	diff
Conv2d	190	same	same	diff	diff
BNorm2d	191	same	same	diff	diff
BcConv2d	192	same	same	diff	diff
Inception	193	same	same	diff	diff
AAvgP2d	194	same	same	diff	diff
Dropout	195	same	diff	diff	diff
Linear	196	diff	diff	diff	diff

- Same data, model, and hyperparameters
- Intentional Randomness (e.g., Dropout)

  → set seeds
- Floating Point Arithmetic



MMlib: Provenance

Other summary is: diff

l_name	fwd_idx	inp	out	grad_inp	grad_out
Conv2d	186	same	same	diff	diff
BNorm2d	187	same	same	diff	diff
BcConv2d	188	same	same	diff	diff
MaxPool2d	189	same	same	diff	diff
Conv2d	190	same	same	diff	diff
BNorm2d	191	same	same	diff	diff
BcConv2d	192	same	same	diff	diff
Inception	193	same	same	diff	diff
AAvgP2d	194	same	same	diff	diff
Dropout	195	same	diff	diff	diff
Linear	196	diff	diff	diff	diff

- Same data, model, and hyperparameters
- Intentional Randomness (e.g., Dropout)

  → set seeds
- Floating Point Arithmetic
  - Track environment, make sure it is the same/equivalent (hardware and software)
  - torch.backends.cudnn.benchmark = False
  - torch.set_deterministic(True)

    now torch.use_deterministic_algorithms(True)

MMlib: Provenance

Other summary is: diff

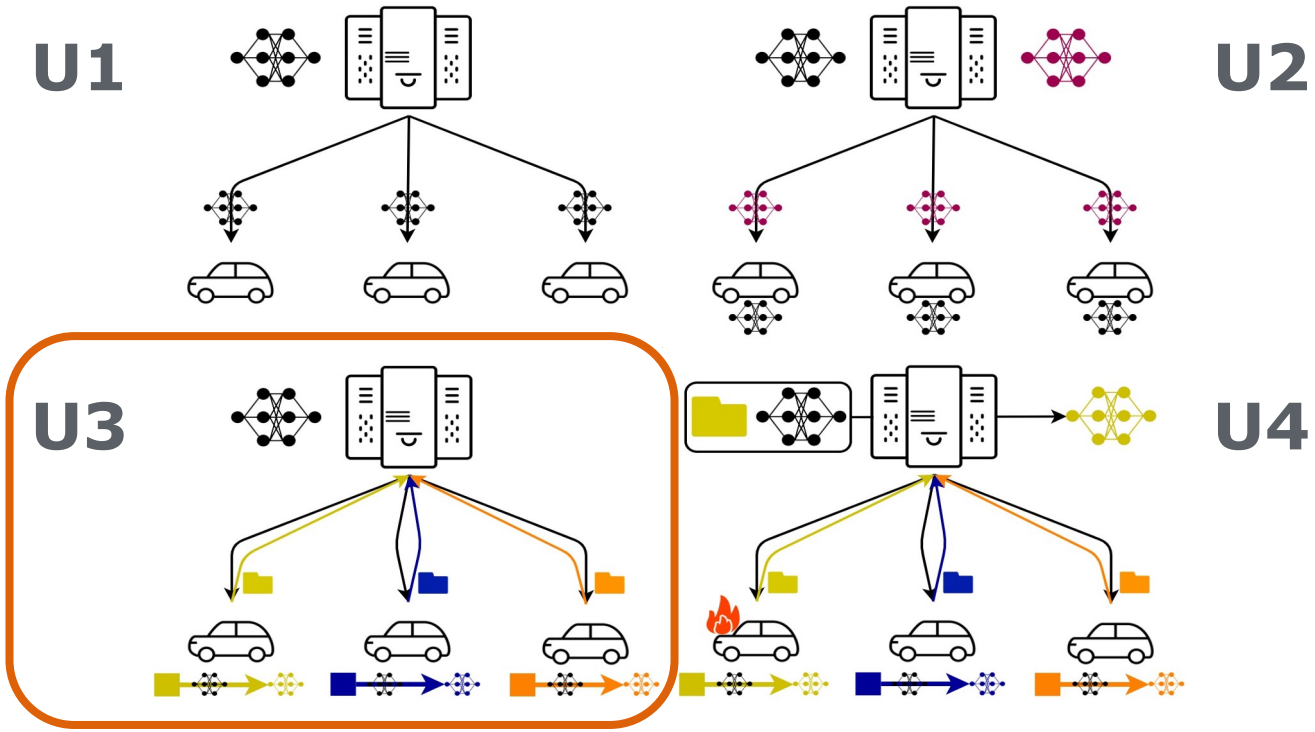
l_name	fwd_idx	inp	out	grad_inp	grad_out
Conv2d	186	same	same	diff	diff
BNorm2d	187	same	same	diff	diff
BcConv2d	188	same	same	diff	diff
MaxPool2d	189	same	same	diff	diff
Conv2d	190	same	same	diff	diff
BNorm2d	191	same	same	diff	diff
BcConv2d	192	same	same	diff	diff
Inception	193	same	same	diff	diff
AAvgP2d	194	same	same	diff	diff
Dropout	195	same	diff	diff	diff
Linear	196	diff	diff	diff	diff

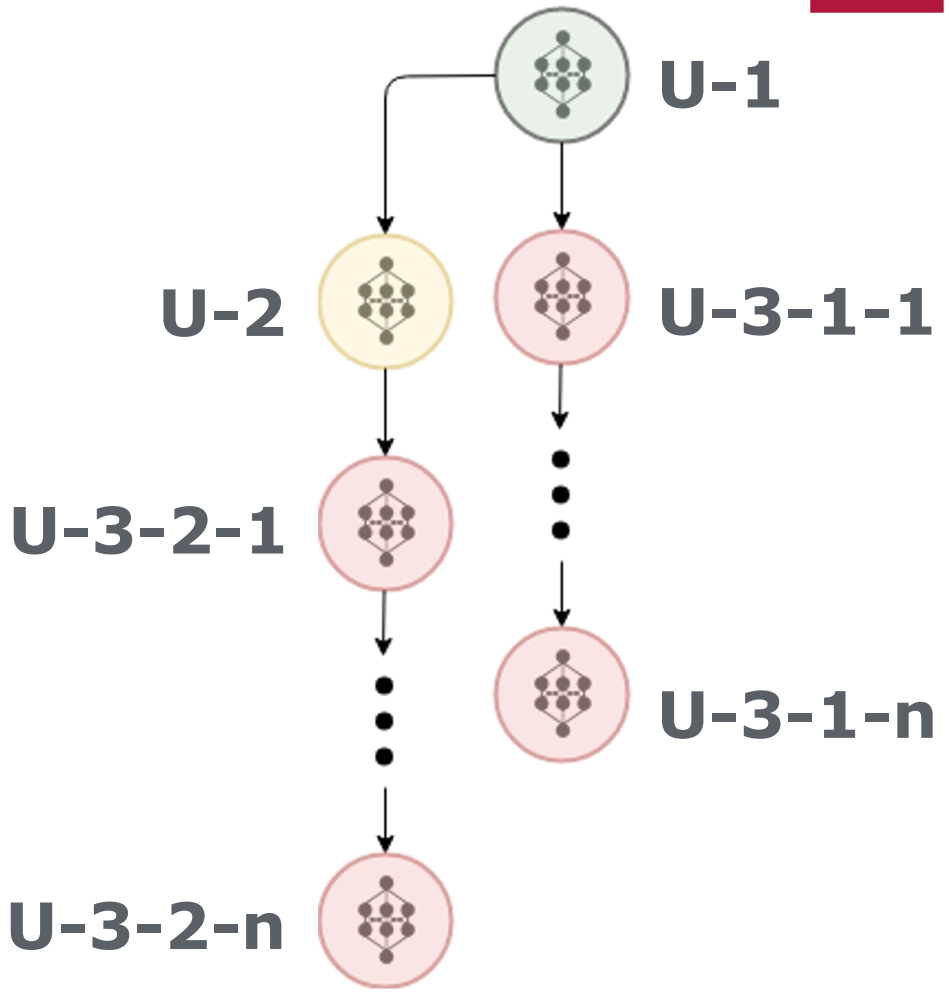
Other summary is: same

l_name	fwd_idx	inp	out	grad_inp	grad_out
Conv2d	186	same	same	same	same
BNorm2d	187	same	same	same	same
BcConv2d	188	same	same	same	same
MaxPool2d	189	same	same	same	same
Conv2d	190	same	same	same	same
BNorm2d	191	same	same	same	same
BcConv2d	192	same	same	same	same
Inception	193	same	same	same	same
AAvgP2d	194	same	same	same	same
Dropout	195	same	same	same	same
Linear	196	same	same	same	same

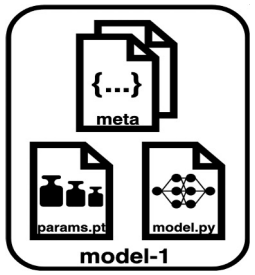
apply measures to make execution deterministic

Evaluation

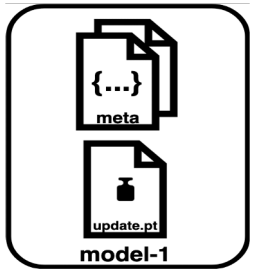




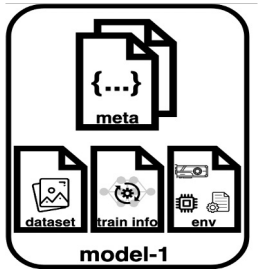
Evaluation



Baseline

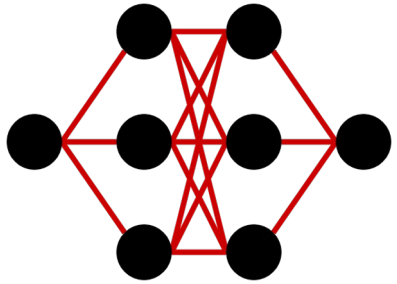


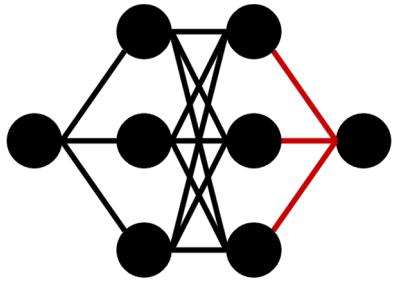
Parameter Update



Provenance

fully updated



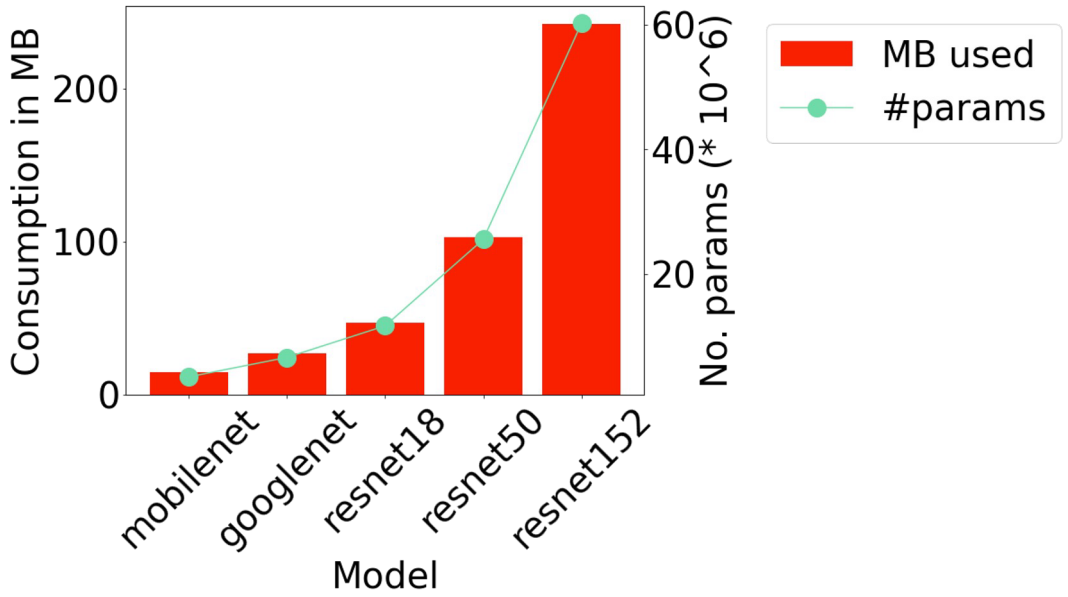


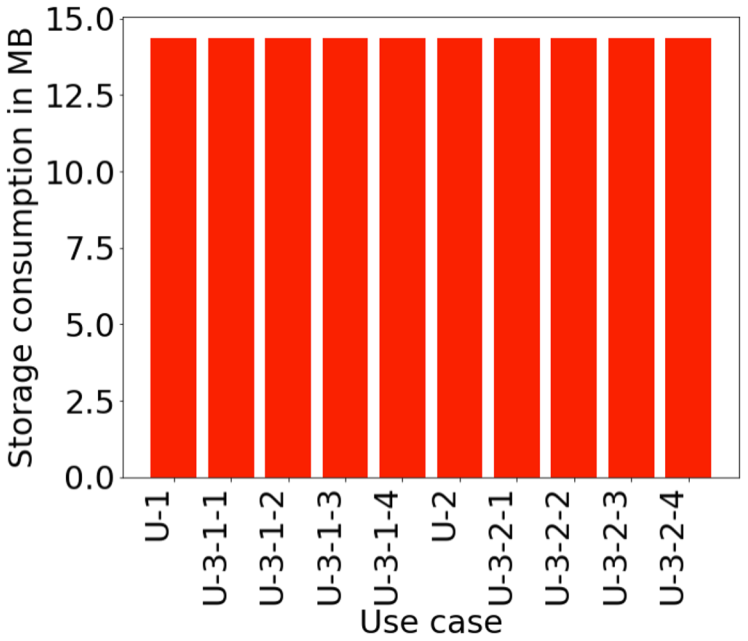
partially updated

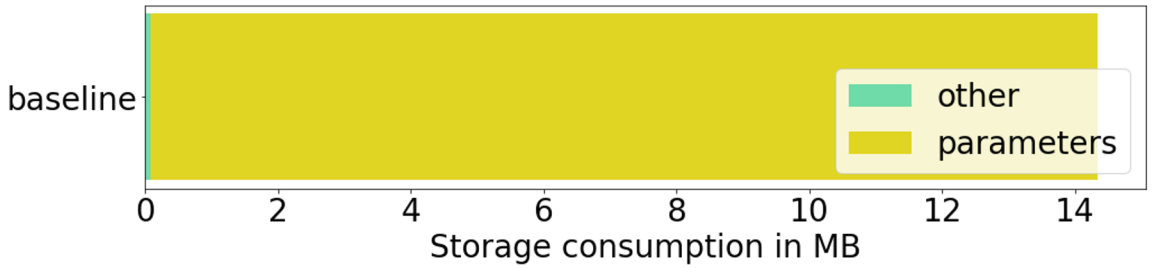
Name	#Params	part. updated	Size
MobileNetV2	3,504,872	1,281,000	14.3 MB
GoogLeNet	6,624,904	1,025,000	26.7 MB
ResNet-18	11,689,512	513,000	46.8 MB
ResNet-50	25,557,032	2,049,000	102.5 MB
ResNet-152	60,192,808	2,049,000	241.7 MB

Short name	Images	Size	Use case
$INet_{val}$	50,000	6.3 GB	U_2
$mINet_{val}$	1,400	200 MB	U_2
CF-512	512	94.3 MB	U_3
CO-512	512	71.6 MB	U_3

Evaluation: Storage Consumption – Baseline



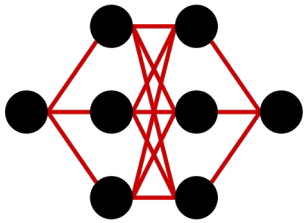




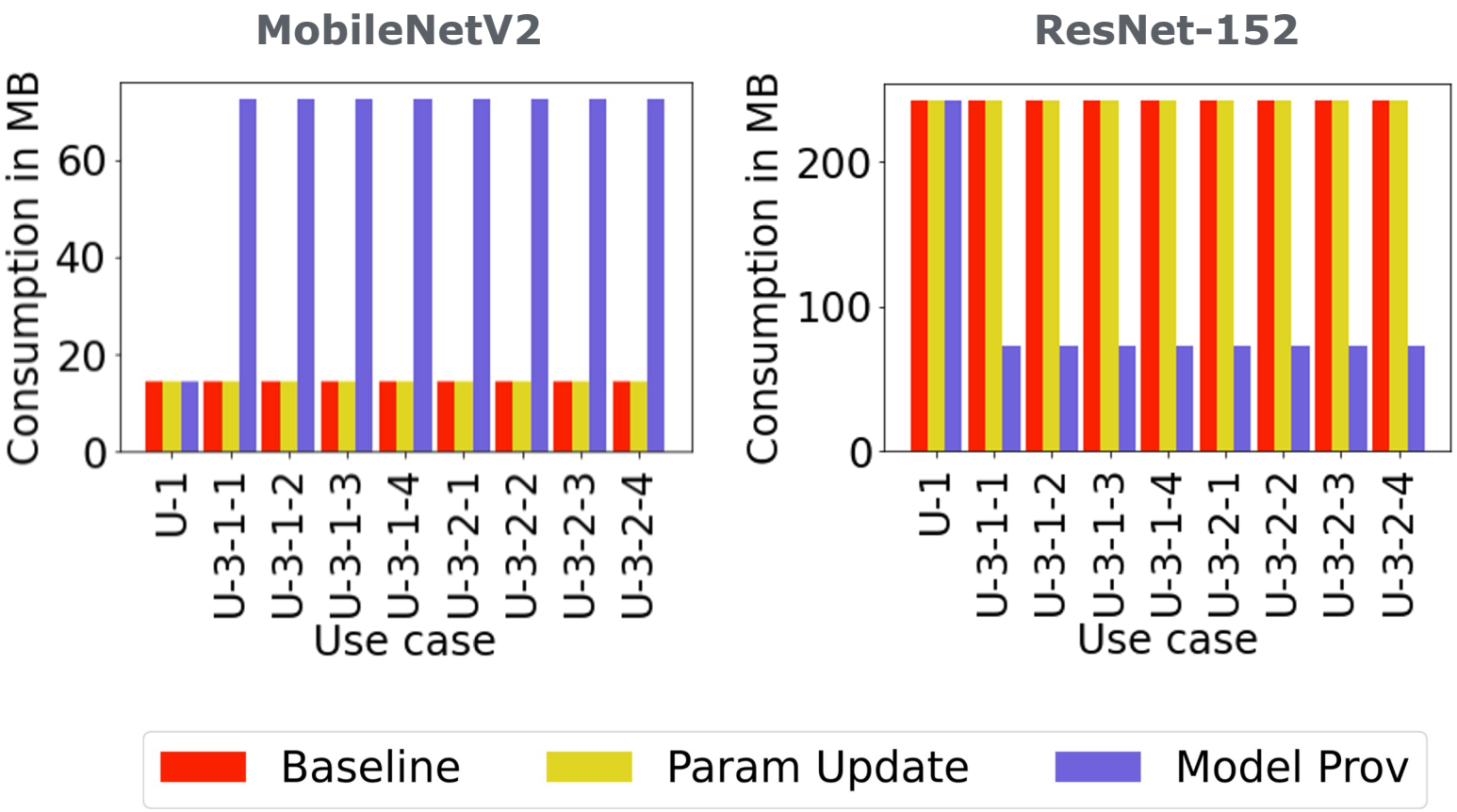
Storage consumption…

- … is mainly depends on **parameters**
- … is **independent** of the use case, dataset, and model relation

Evaluation: Storage Consumption

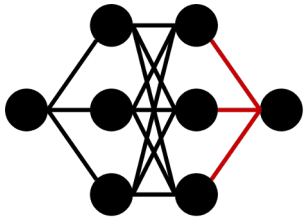


fully updated

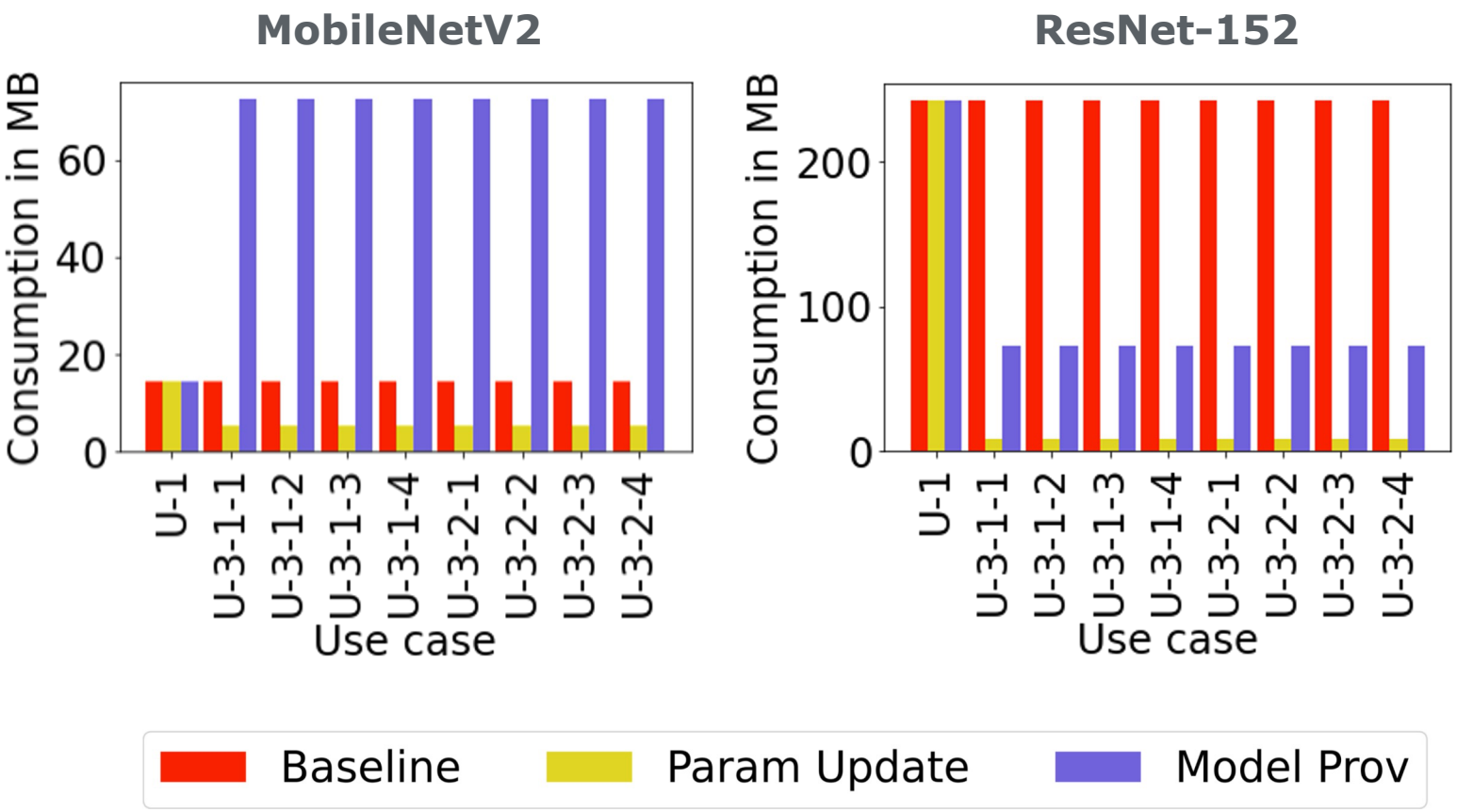


- **No significant improvement** with parameter update approach
- For provenance approach **storage consumption** depends on the **dataset**

Evaluation: Storage Consumption

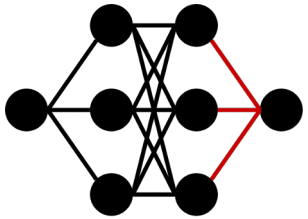


partially updated

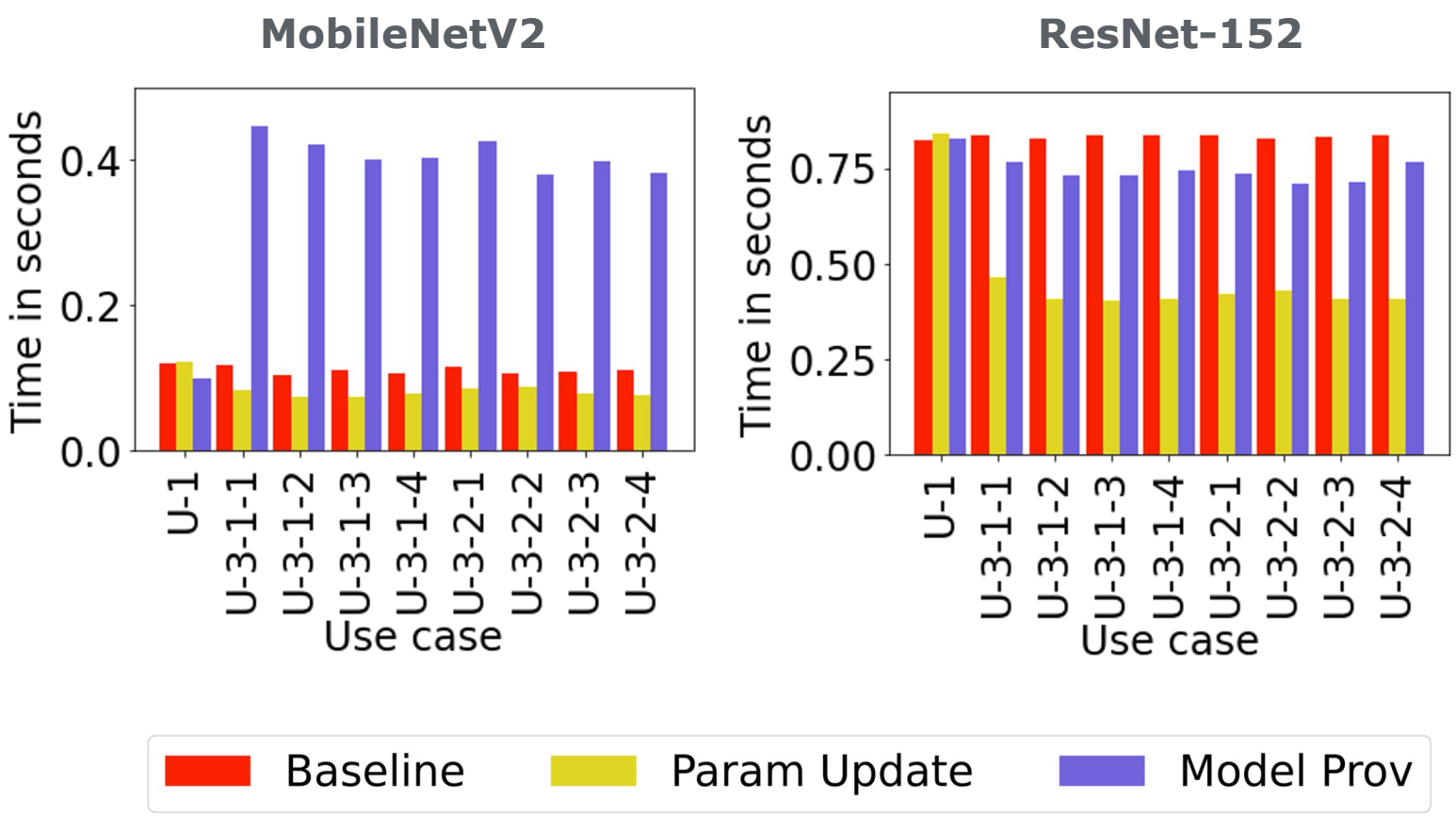


- **Significant improvement** with parameter update approach
- For provenance approach **storage consumption** depends on the **dataset**

Evaluation: Time-to-Save

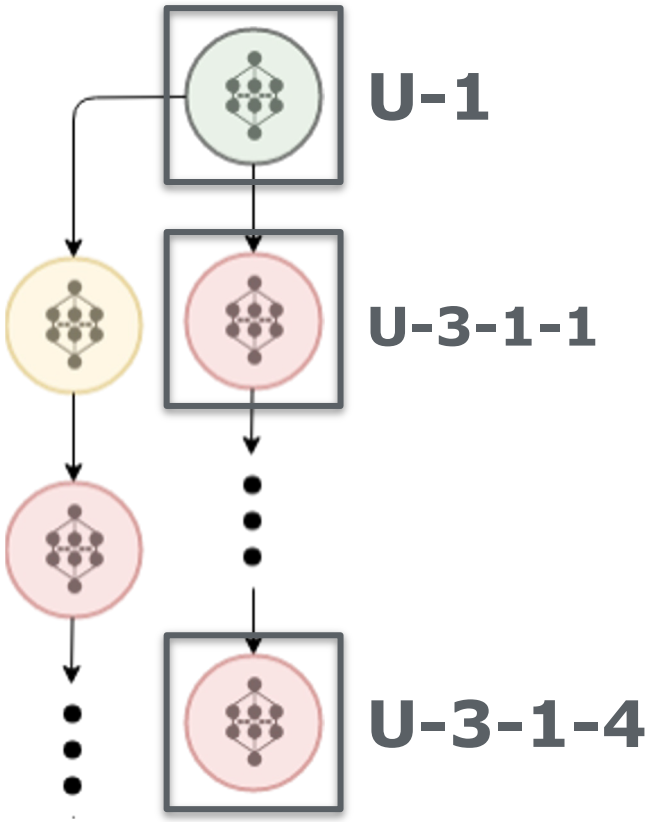


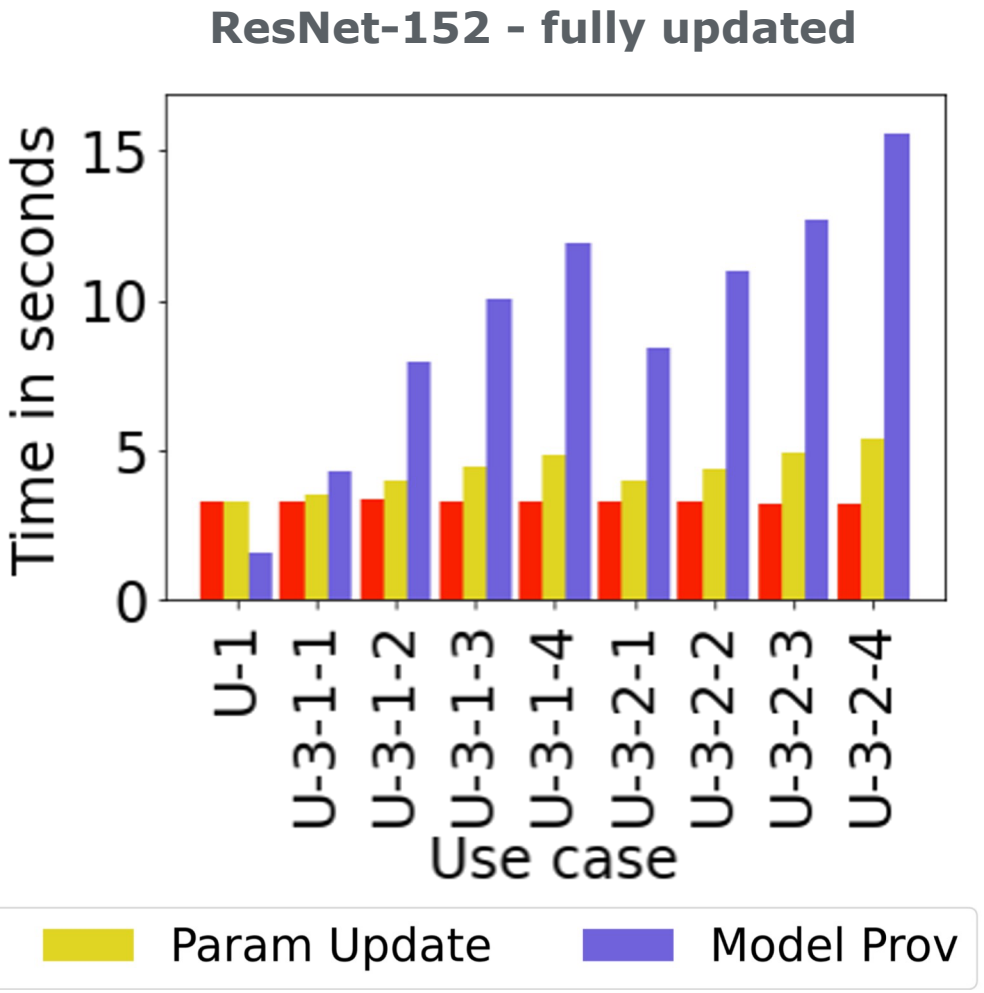
partially updated



- For baseline and parameter update: depends on **size of parameters**
- For provenance: depends mostly on **dataset**
- **Both advanced can outperform baseline and each other**

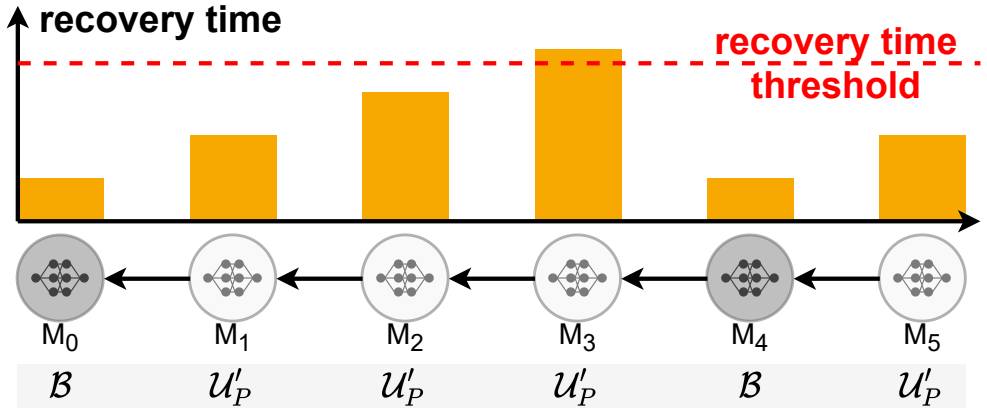
Evaluation: Time-to-Recover





- **Baseline has shortest time-to- recover**
- **Marginally** increasing for parameter update
- **Long** for provenance approach
- Advanced approaches show **staircase pattern**

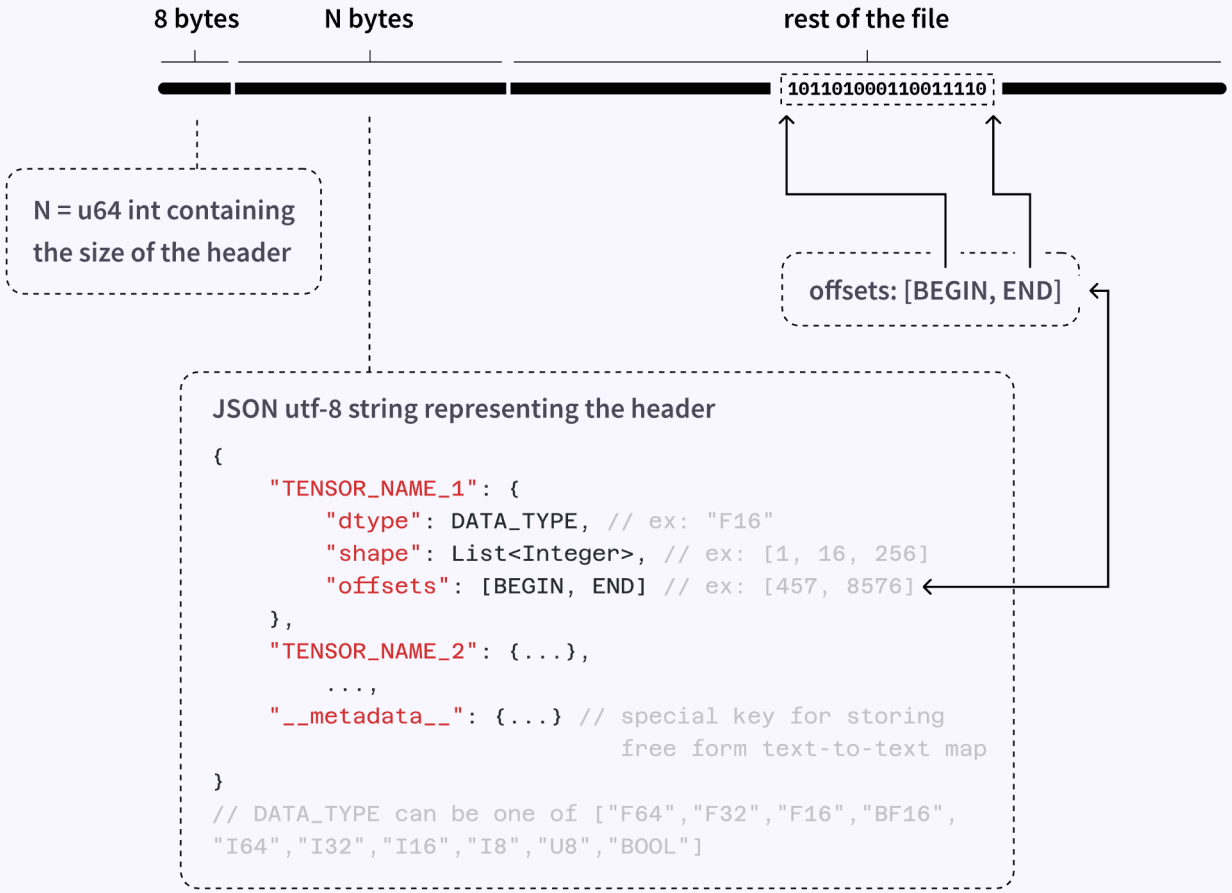
Future Work



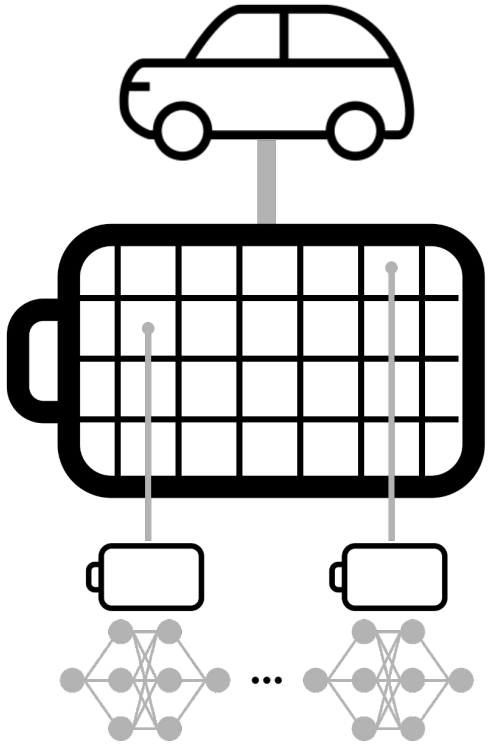
Hybrid Approaches

Safetensors Format

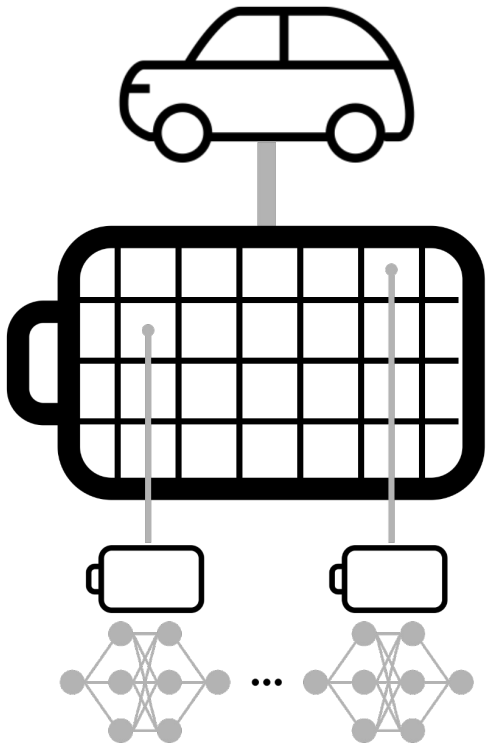
File name **model.safetensors**

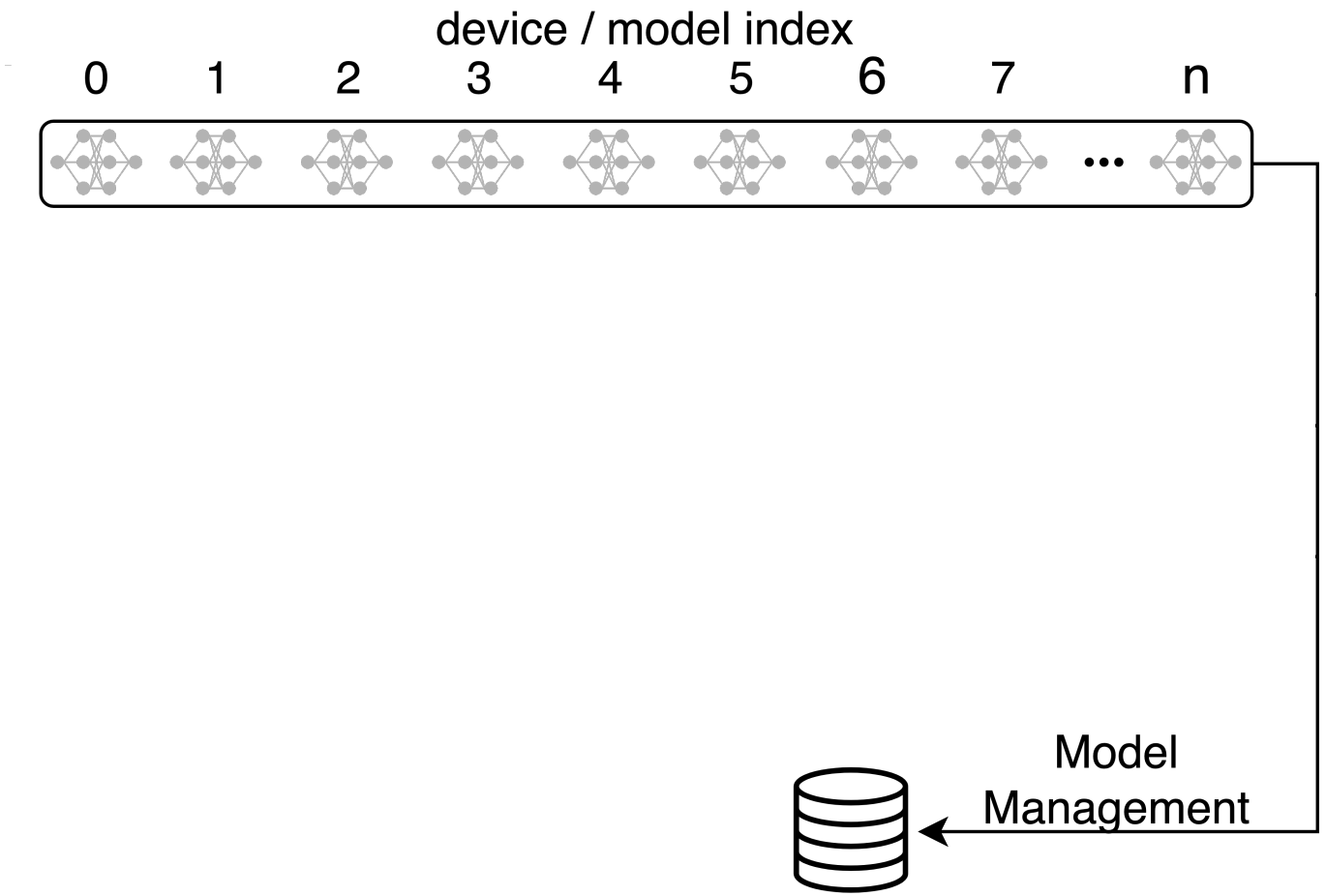


Future Work

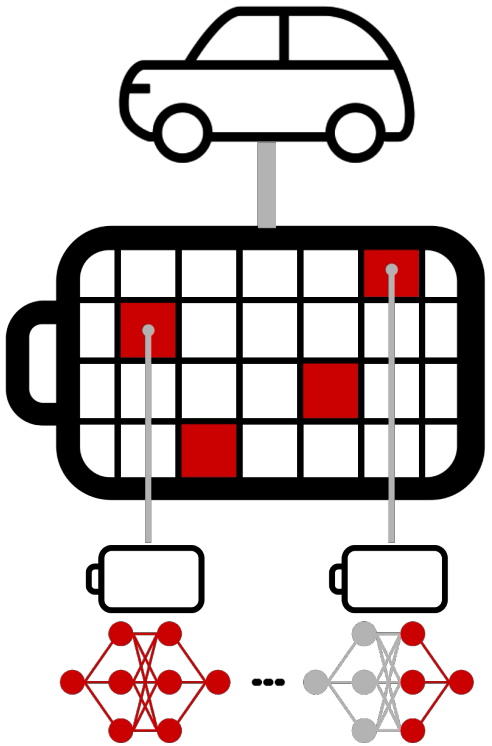


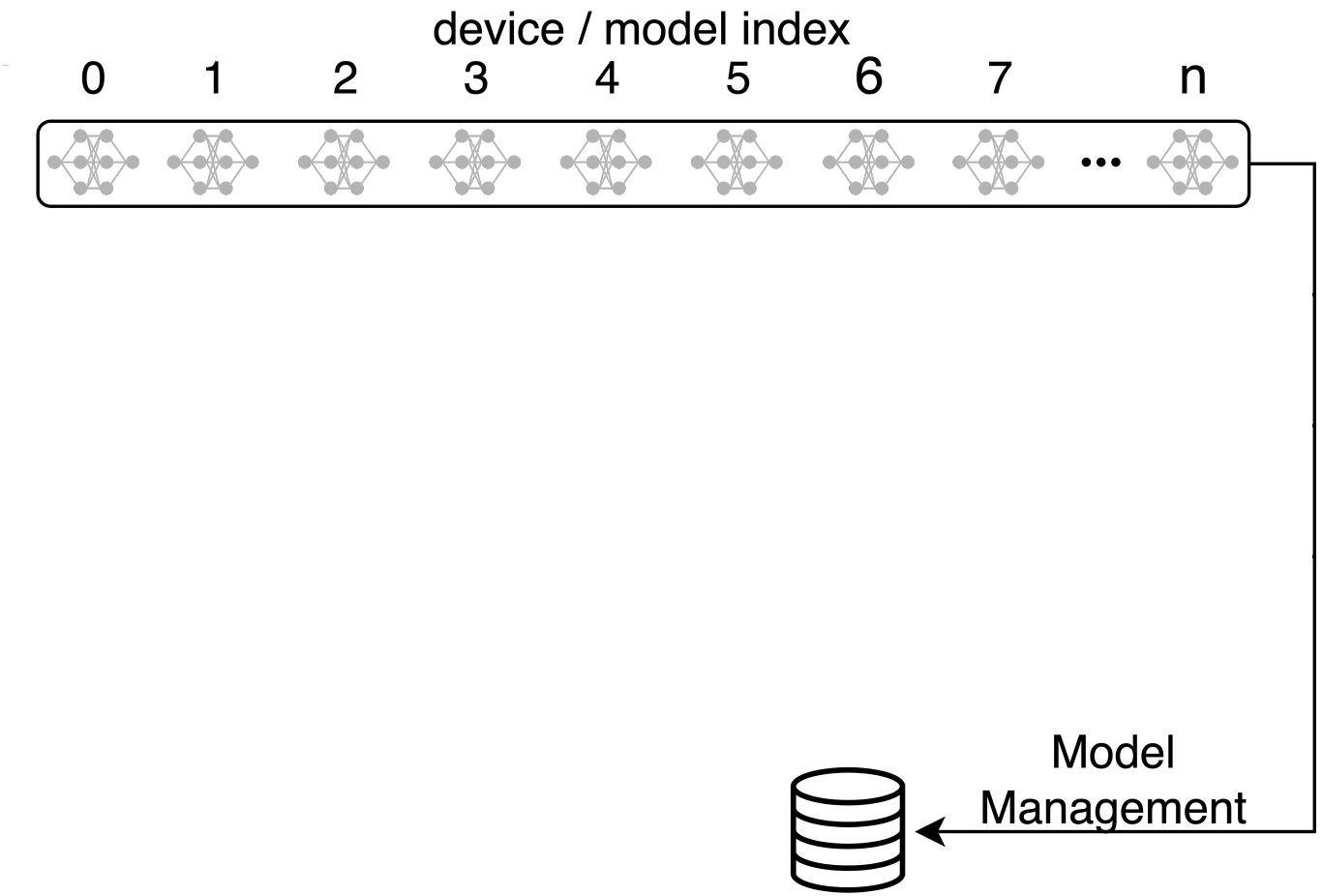
Future Work



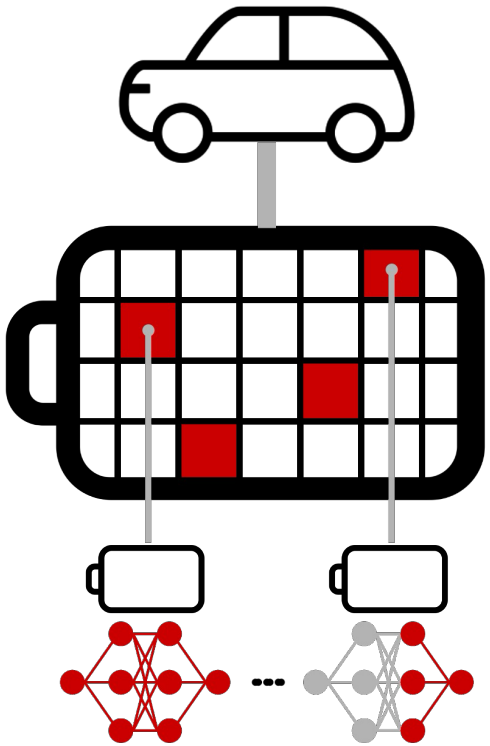


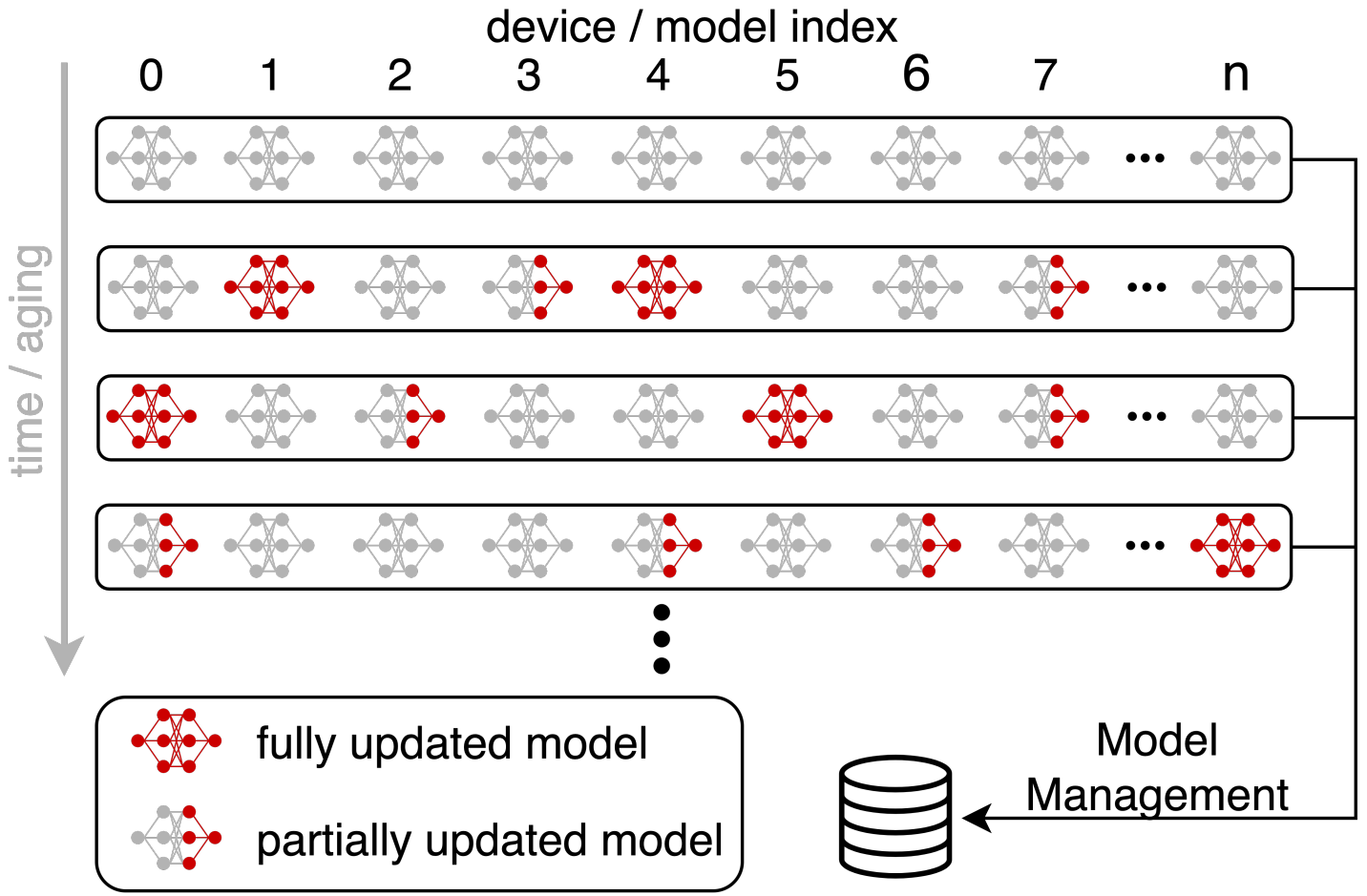
Future Work





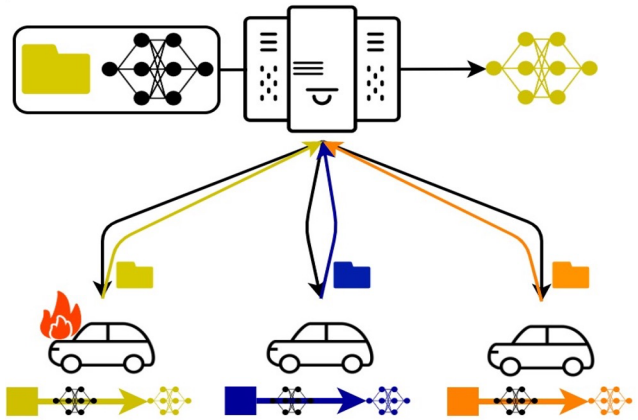
Future Work




device / model index
0
1
2
3
4
5
6
7
n
time / aging
fully updated model
partially updated model
Model Management

Model Management

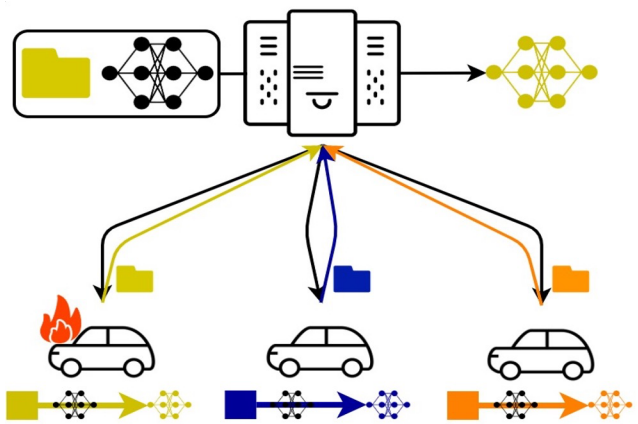
Model Management for Archival



- Efficiently manage many DL models in a distributed environment
- Assumption: save often, read rarely
- MMlib and M3lib

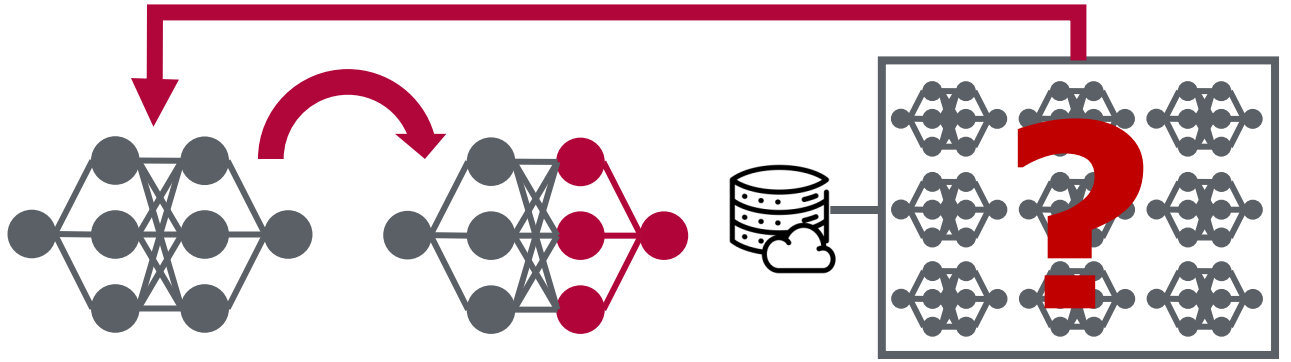
Model Management

Model Management for Archival



- Efficiently manage many DL models in a distributed environment
- Assumption: save often, read rarely
- MMlib and M3lib

Model Management for Reuse



- Efficiently access models to find the best candidate for transfer learning
- Assumption: read often, models overlap
- Poodle and Alsatian

Authors

Nils Strassenburg

nils.strassenburg@hpi.de
Hasso Plattner Institute
University of Potsdam

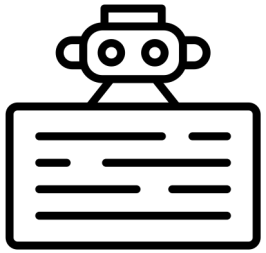
Boris Glavic

bglavic@uic.edu
University of Illinois,
Chicago

Tilmann Rabl

tilmann.rabl@hpi.de
Hasso Plattner Institute
University of Potsdam

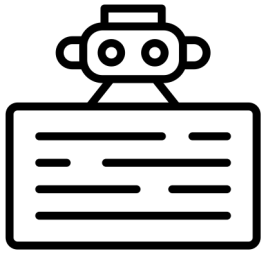
LLMs



LLMs (ML-as-a-Service)

- State-of-the-art capabilities
- Simple API integration
- No AI expertise
- Zero model development cost
- No data collection
- No data labeling

LLMs

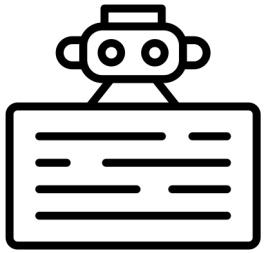


LLMs (ML-as-a-Service)

p_id	u_id	review	sentiment
1	1	Quality is fine ...	? ➔ 1
2	1	Broken after ...	? ➔ 0
3	1	Super happy ...	? ➔ 1

- State-of-the-art capabilities
- Simple API integration
- No AI expertise
- Zero model development cost
- No data collection
- No data labeling

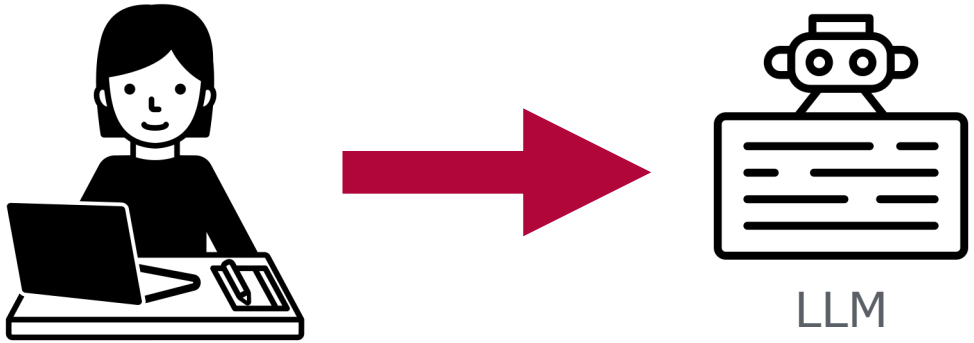
LLMs



LLMs (ML-as-a-Service)

- State-of-the-art capabilities
- Simple API integration
- No AI expertise
- Zero model development cost
- No data collection
- No data labeling

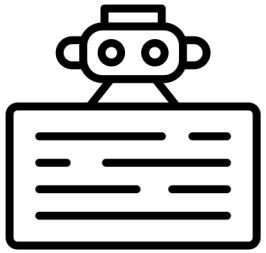
p_id	u_id	review	sentiment
1	1	Quality is fine ...	? → 1
2	1	Broken after ...	? → 0
3	1	Super happy ...	? → 1



Offload simple recurring tasks to LLM

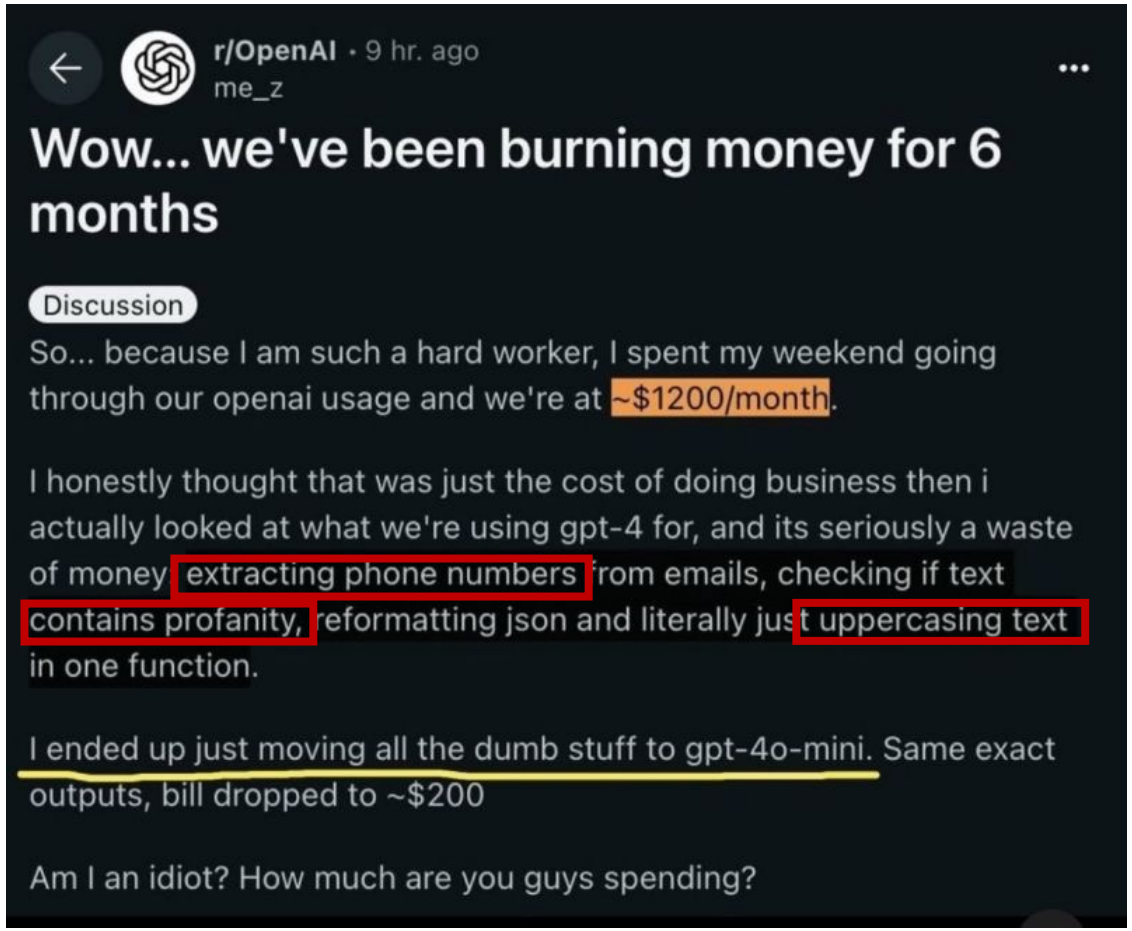
As long as it runs → focus on something else

LLMs



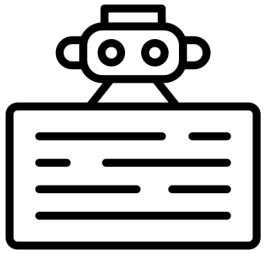
LLMs (ML-as-a-Service)

- ⊖ High inference costs
- ⊕ State-of-the-art capabilities
- ⊕ Simple API integration
- ⊕ No AI expertise
- ⊕ Zero model development cost
- ⊕ No data collection
- ⊕ No data labeling



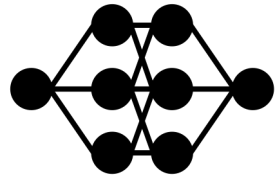
https://www.linkedin.com/posts/... (accessed 03.02.2026)

LLMs vs Custom Models



LLMs (ML-as-a-Service)

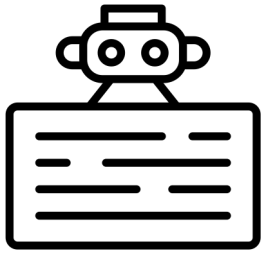
- ⊖ High inference costs
- ⊕ State-of-the-art capabilities
- ⊕ Simple API integration
- ⊕ No AI expertise
- ⊕ Zero model development cost
- ⊕ No data collection
- ⊕ No data labeling



Custom Models

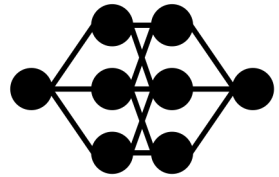
- ⊕ Low inference costs

LLMs vs Custom Models



LLMs (ML-as-a-Service)

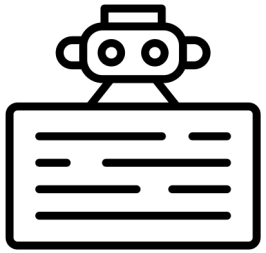
- ➖ High inference costs
- ➕ State-of-the-art capabilities
- ➕ Simple API integration
- ➕ No AI expertise
- ➕ Zero model development cost
- ➕ No data collection
- ➕ No data labeling



Custom Models

- ➕ Low inference costs
- ➖ (?) State-of-the-art capabilities
- ➖ (?) Simple API integration
- ➖ AI expertise needed
- ➖ High model development cost
- ➖ Data collection needed
- ➖ Data labeling needed

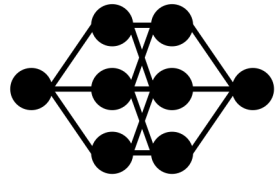
LLMs vs Custom Models



LLMs (ML-as-a-Service)

- − High inference costs
- \+ State-of-the-art capabilities
- \+ Simple API integration
- \+ No AI expertise
- \+ Zero model development cost
- \+ No data collection
- \+ No data labeling

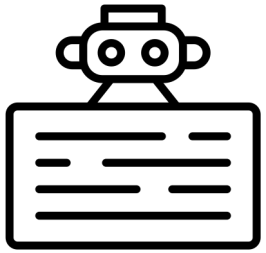
Can we get the best of both worlds?



Custom Models

- \+ Low inference costs
- − (?) State-of-the-art capabilities
- − (?) Simple API integration
- − AI expertise needed
- − High model development cost
- − Data collection needed
- − Data labeling needed

LLMs vs Custom Models

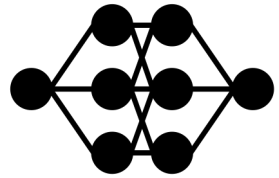


LLMs (ML-as-a-Service)

- − High inference costs
- \+ State-of-the-art capabilities
- \+ Simple API integration
- \+ No AI expertise
- \+ Zero model development cost
- \+ No data collection
- \+ No data labeling

Just-in-Time Model Replacement

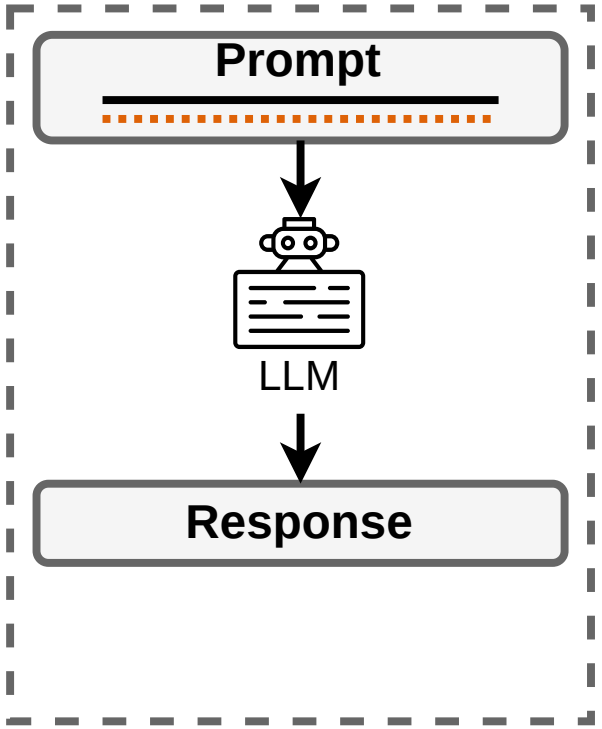
Can we get the best of both worlds?



Custom Models

- \+ Low inference costs
- − (?) State-of-the-art capabilities
- − (?) Simple API integration
- − AI expertise needed
- − High model development cost
- − Data collection needed
- − Data labeling needed

Baseline LLM Usage

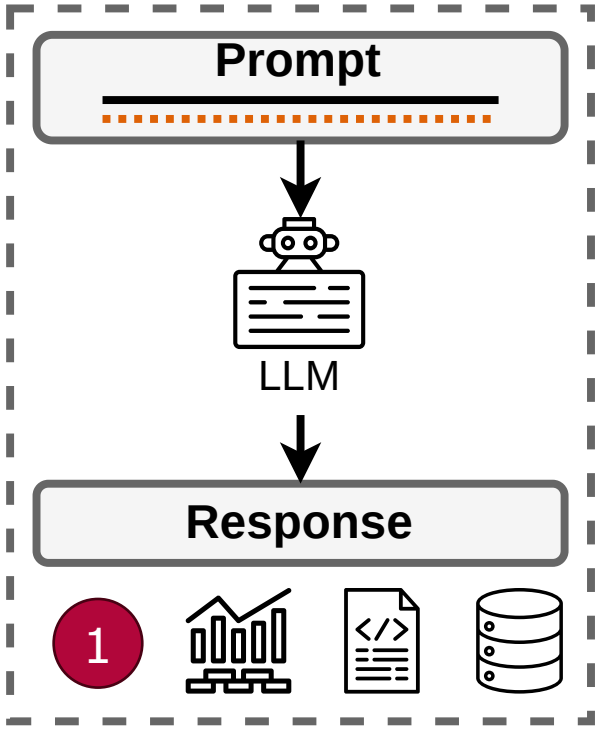


p_id	u_id	review	sentiment
1	1	Quality is fine ...	? ➔ 1
2	1	Broken after ...	? ➔ 0
3	1	Super happy ...	? ➔ 1

Baseline

- Embed relevant data in prompt
- Send to LLM
- Write response back to table

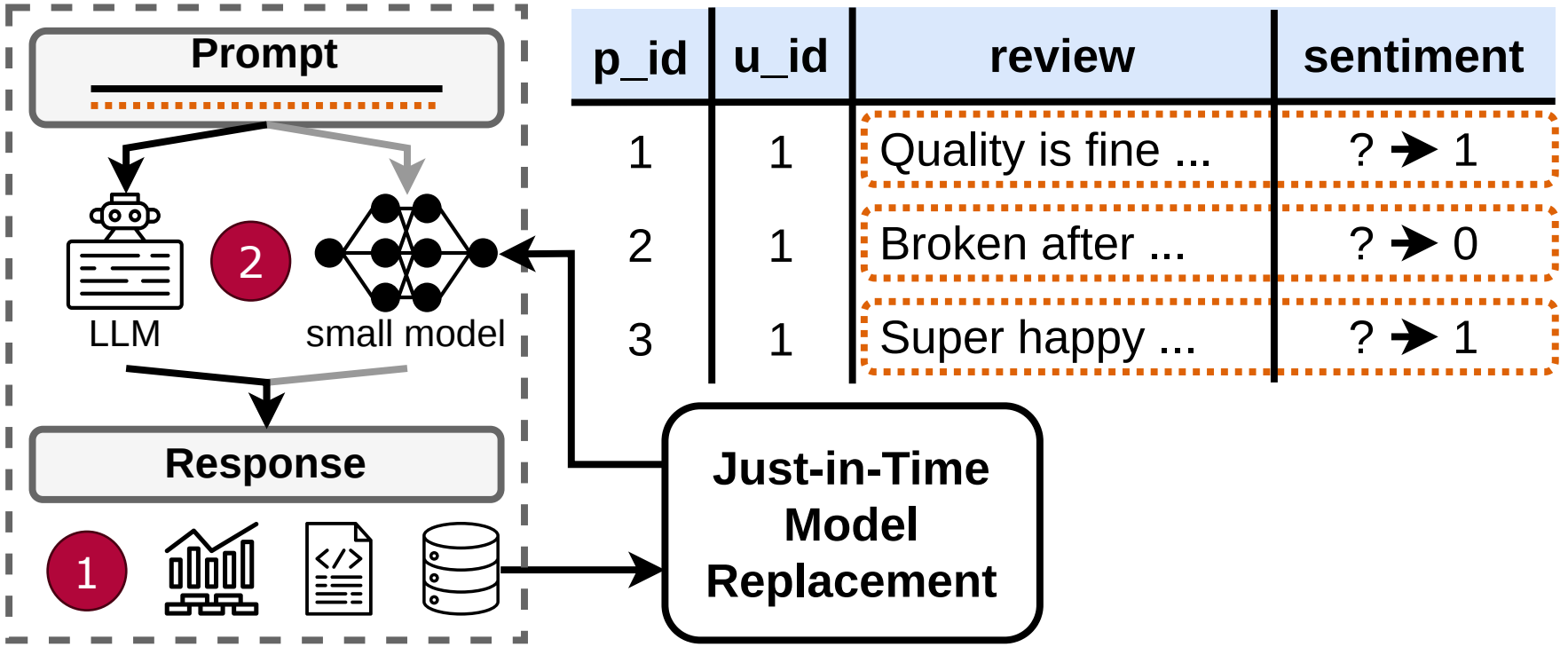
Just-in-Time Model Replacement



p_id	u_id	review	sentiment
1	1	Quality is fine ...	? ➔ 1
2	1	Broken after ...	? ➔ 0
3	1	Super happy ...	? ➔ 1

1 Collect LLM labels, task type, task metadata

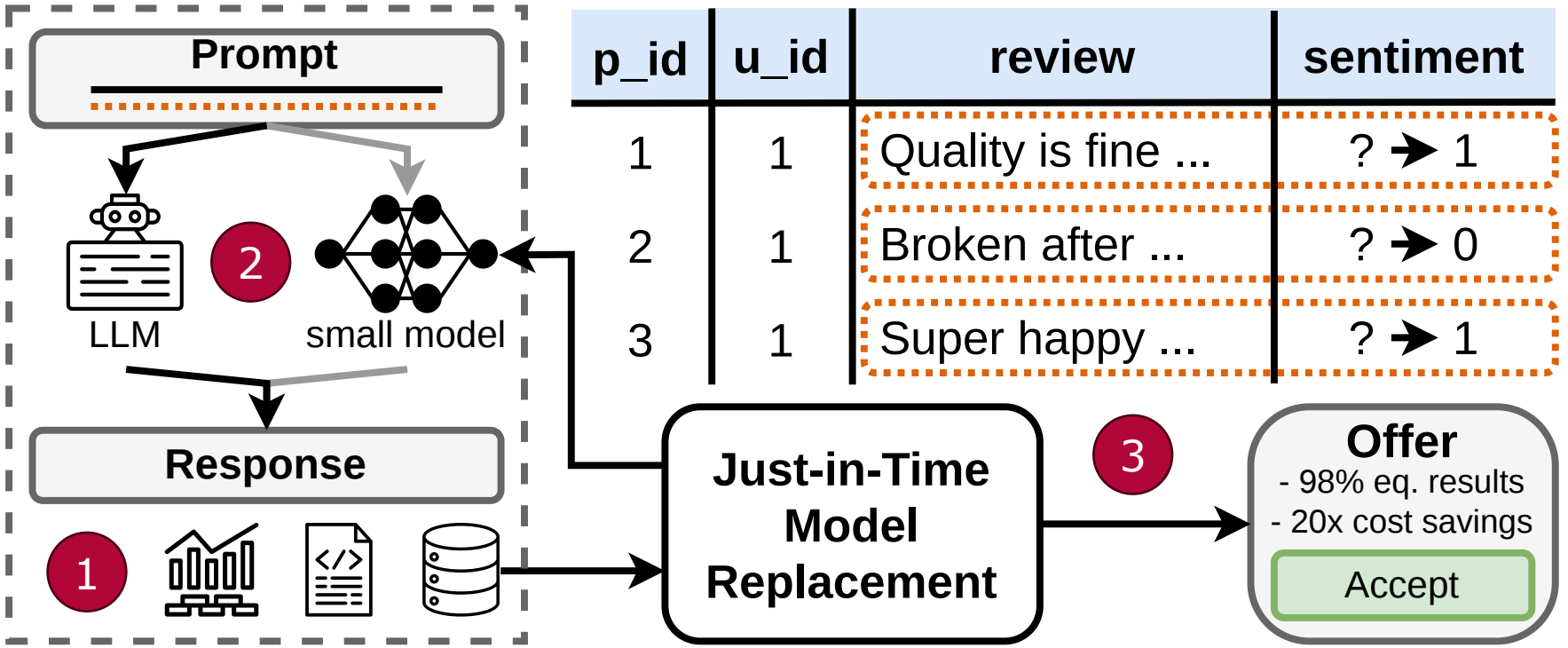
Just-in-Time Model Replacement



p_id	u_id	review	sentiment
1	1	Quality is fine ...	? → 1
2	1	Broken after ...	? → 0
3	1	Super happy ...	? → 1

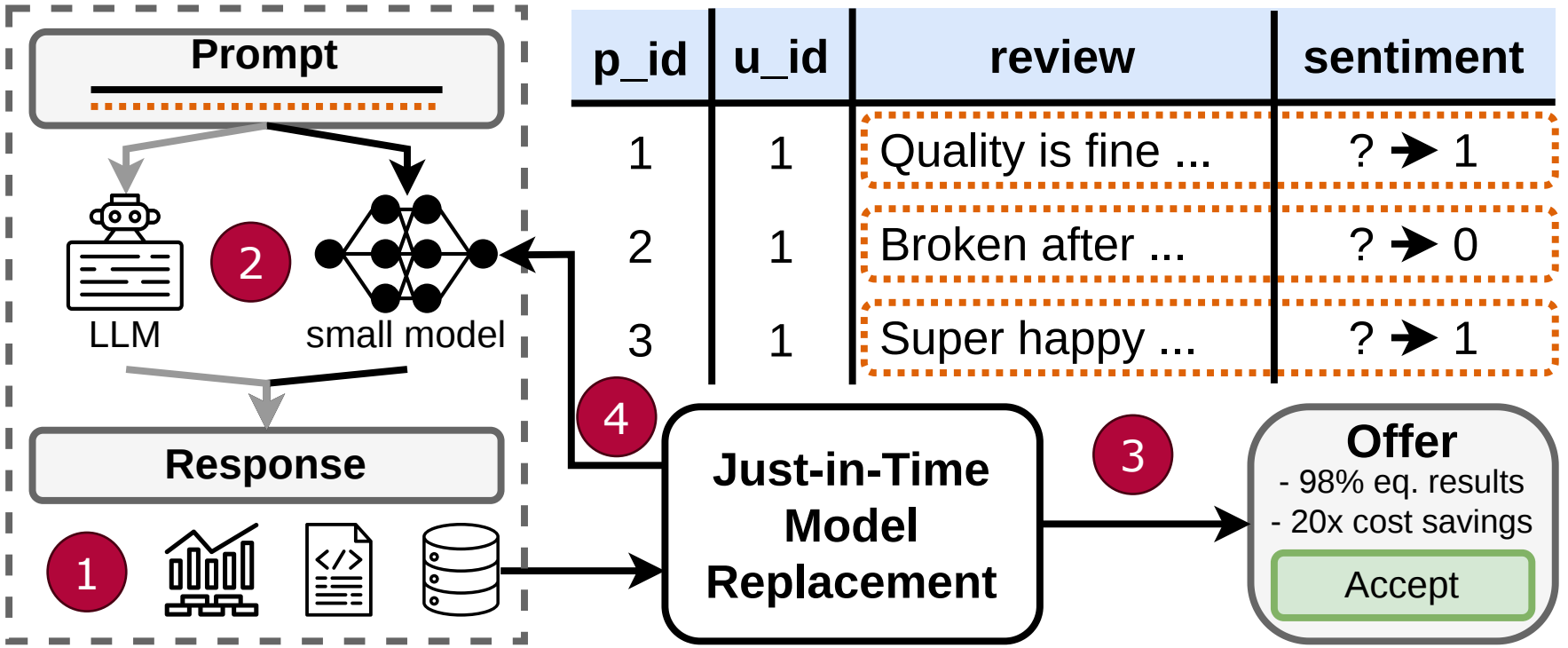
1. Collect LLM labels, task type, task metadata
2. Develop custom model → monitor model

Just-in-Time Model Replacement



1. Collect LLM labels, task type, task metadata
2. Develop custom model → monitor model
3. (Ask user to accept switch)

Just-in-Time Model Replacement



1. Collect LLM labels, task type, task metadata
2. Develop custom model → monitor model
3. (Ask user to accept switch)
4. Replace LLM with custom model

Conditions to Meet For JITR to Be Effective

1) Tasks can be identified

```
You are a sentiment analysis assistant. Your job is to read movie reviews and
classify their sentiment as either "positive" or "negative". Only respond in
this exact JSON format: "sentiment": "positive" or "sentiment": "negative".
Do not provide any explanation or additional text. Review: <REVIEW TEXT>
```

Figure 3: Base Prompt

```
You are a powerful language model. You are given a user request consisting of
a system prompt and a user message. Your task is as follows:
1. Identify the type of input the user is providing: one of ["text", "image",
"table", "other"]
2. Infer what task you are expected to perform, choosing from: ["sentiment
classification", "summarization", "translation", "question answering",
"information extraction", "topic modeling", "other"]
3. Solve the task that the user is giving you and put your response in a
separate field called "user_response" which is in JSON format.
Respond with a JSON object containing: "input_type", "task_type",
"user_response" (JSON describing user response)
USER REQUEST: <USER REQUEST>
```

Figure 4: Wrapper Prompt

```
{ "input_type": "text", "task_type": "sentiment classification",
"user_response": "sentiment": "negative" }
```

Figure 5: Wrapped Response

Conditions to Meet For JITR to Be Effective

1) Tasks can be identified

2) Logged requests and LLM responses provide sufficient training data

Conditions to Meet For JITR to Be Effective

1) Tasks can be identified

2) Logged requests and LLM responses provide sufficient training data

3) Surrogate model has satisfactory performance

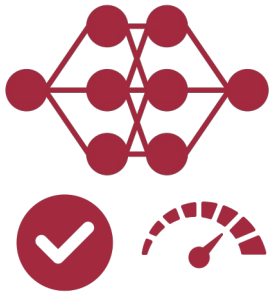
Conditions to Meet For JITR to Be Effective

1) Tasks can be identified

2) Logged requests and LLM responses provide sufficient training data

3) Surrogate model has satisfactory performance

4) Monitoring can detect performance degradation



Conditions to Meet For JITR to Be Effective

1) Tasks can be identified

2) Logged requests and LLM responses provide sufficient training data

3) Surrogate model has satisfactory performance

4) Monitoring can detect performance degradation

5) Cost of task identification, model development, and monitoring can be amortized

Conditions to Meet For JITR to Be Effective

1) Tasks can be identified

2) Logged requests and LLM responses provide sufficient training data

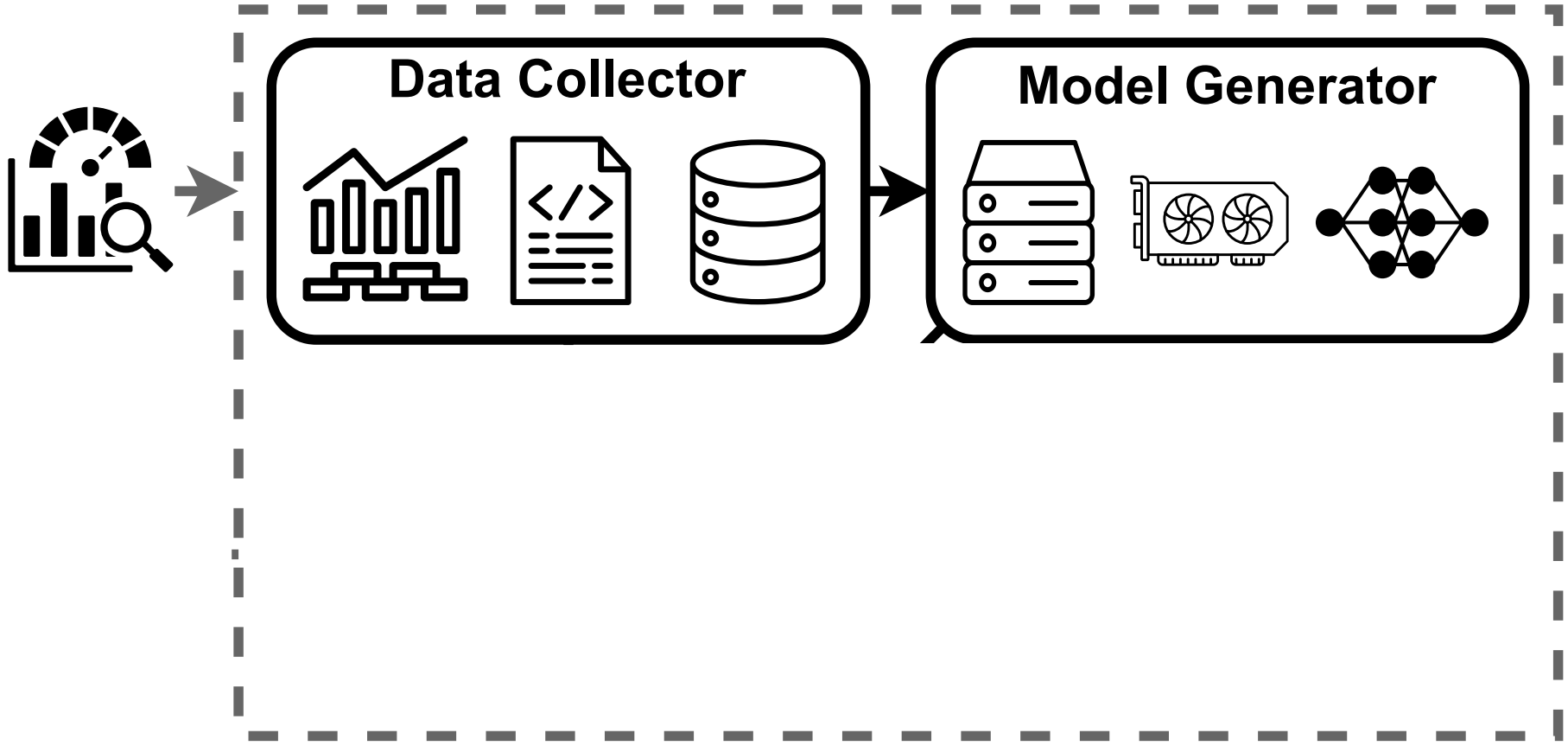
3) Surrogate model has satisfactory performance

4) Monitoring can detect performance degradation

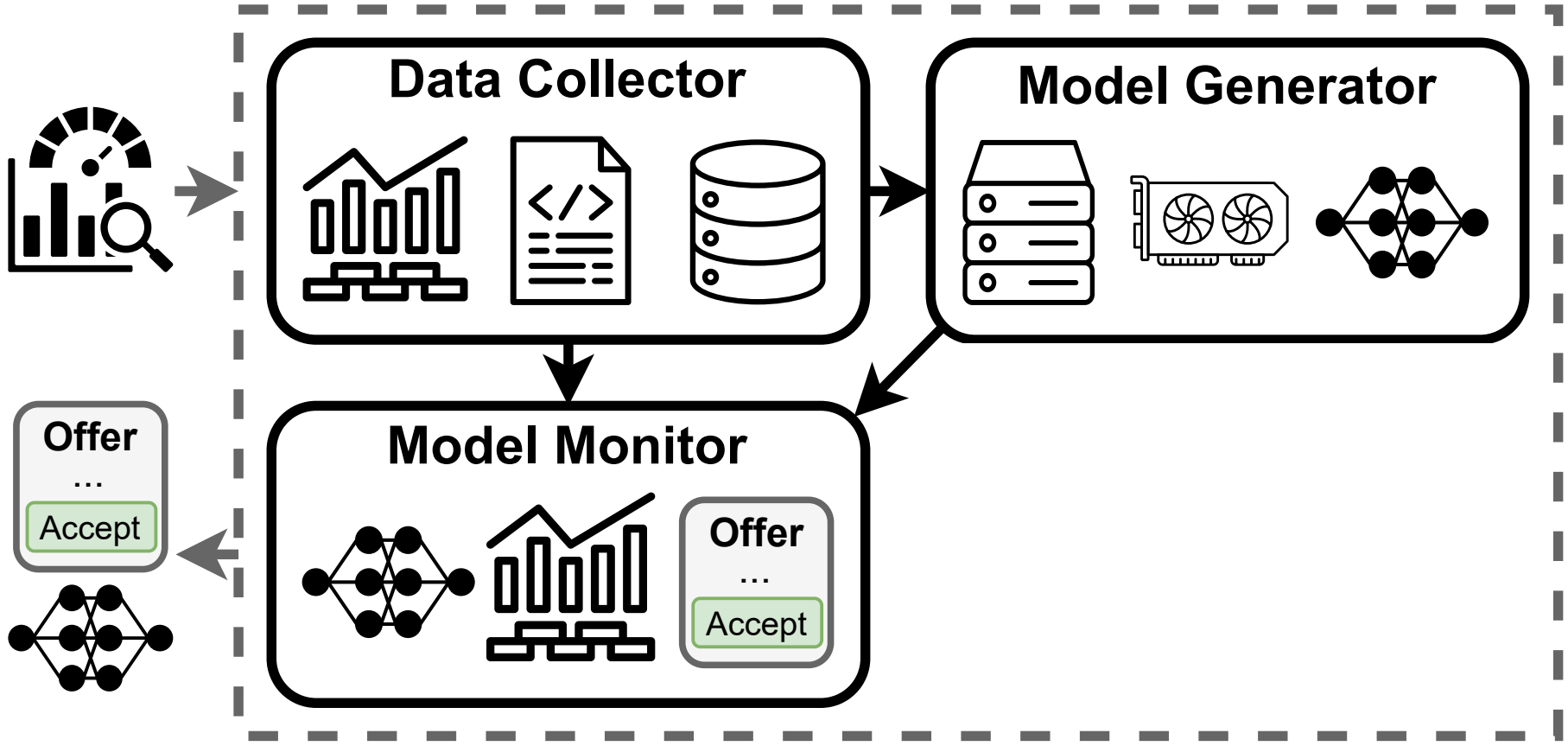
5) Cost of task identification, model development, and monitoring **can be amortized**

Poodle: Just-in-Time Model Replacement Prototype

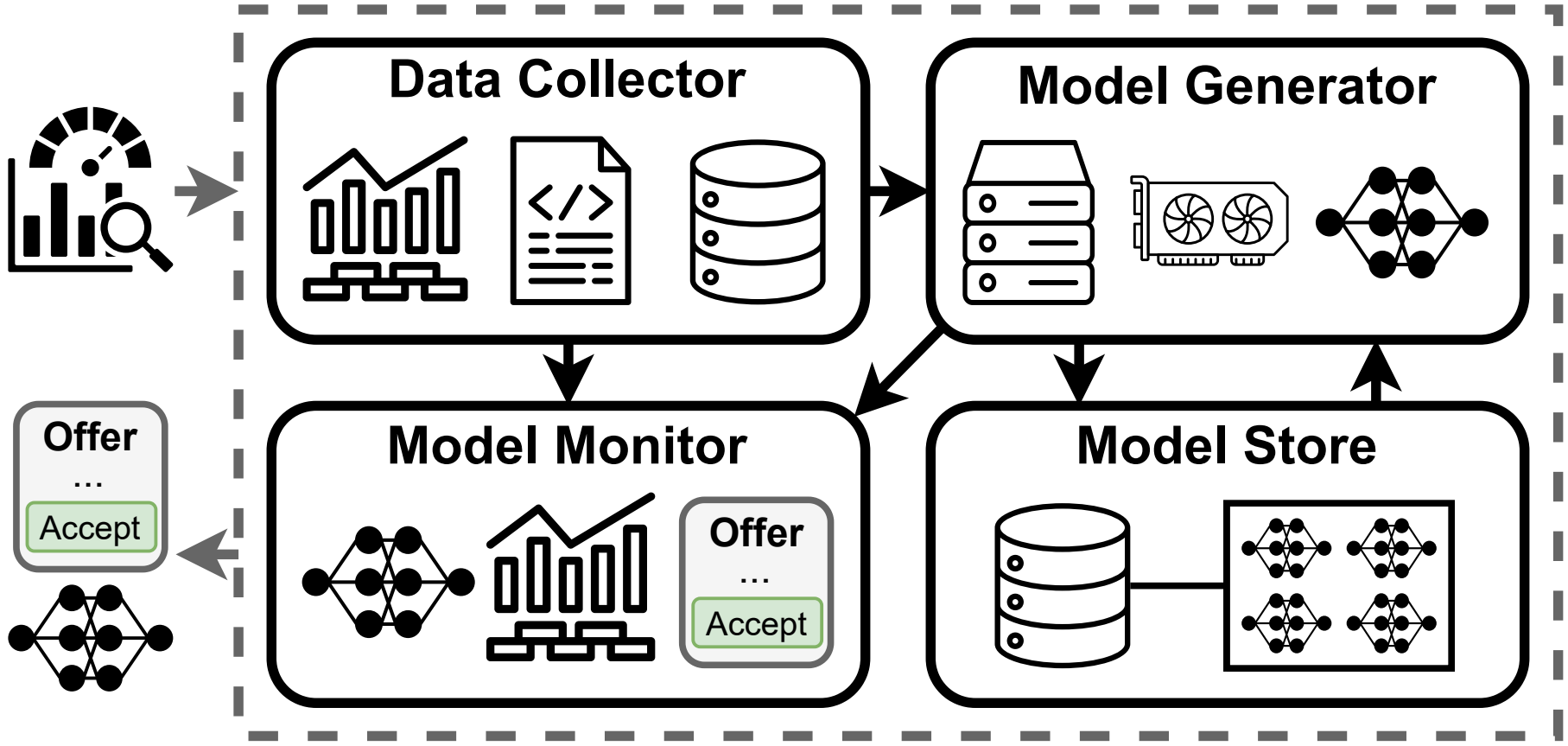
Poodle: Just-in-Time Model Replacement Prototype



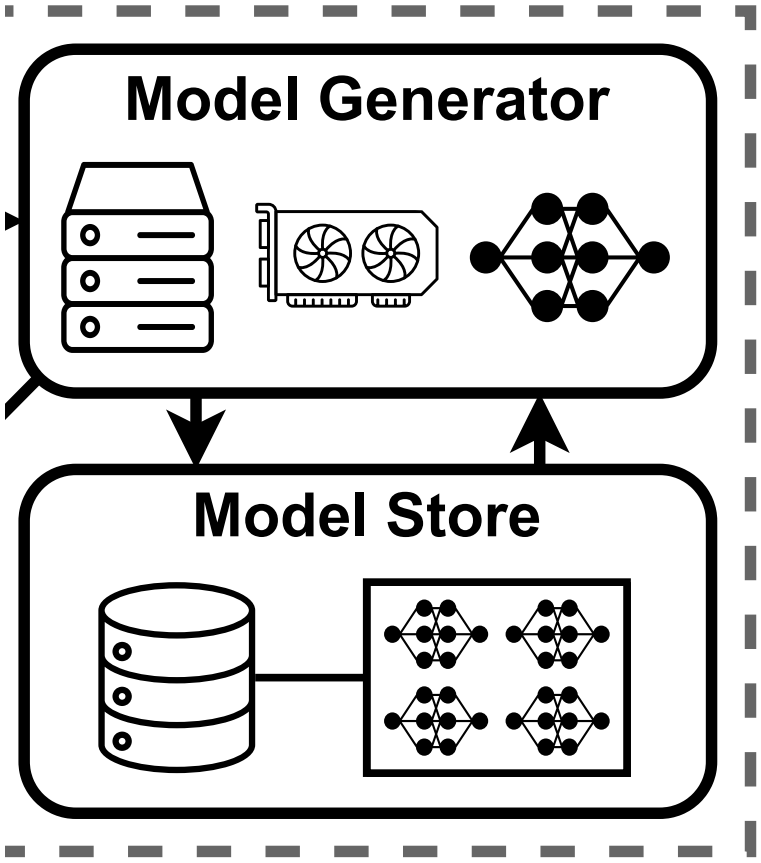
Poodle: Just-in-Time Model Replacement Prototype



Poodle: Just-in-Time Model Replacement Prototype

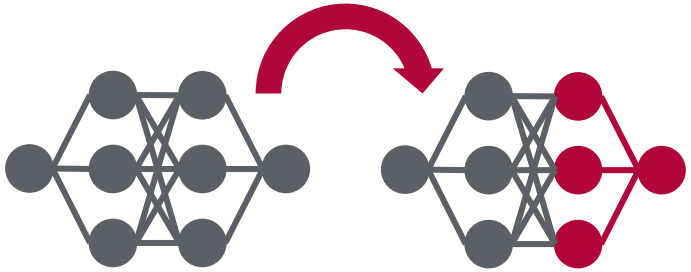


Poodle: Just-in-Time Model Replacement Prototype



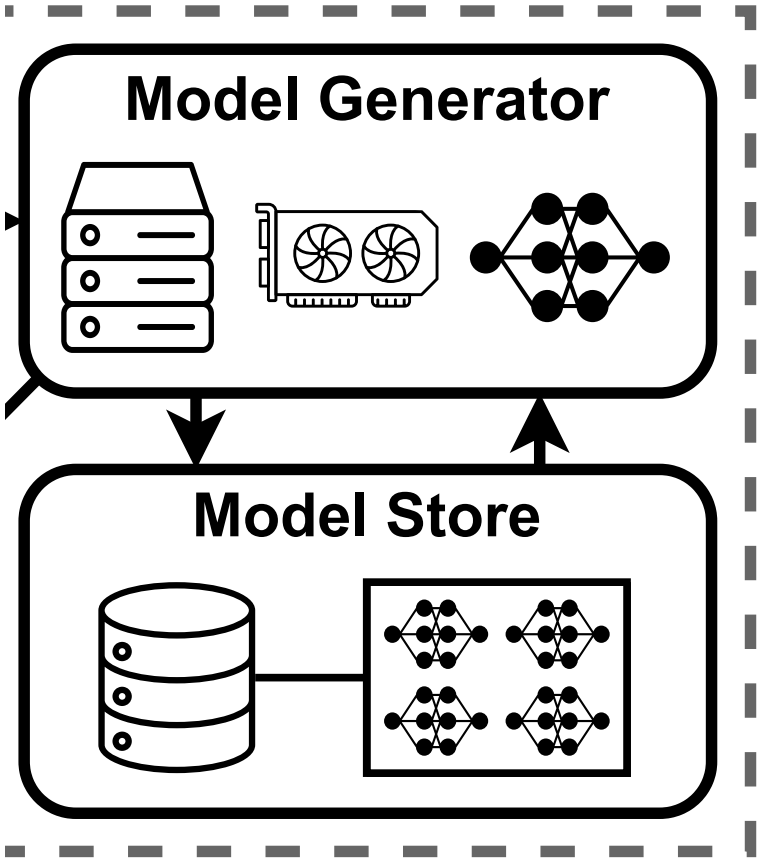
Baseline Model Development

- Label or collect training data
- Fine-tune a “standard” model



Transfer Learning
(fine-tuning)

Poodle: Just-in-Time Model Replacement Prototype

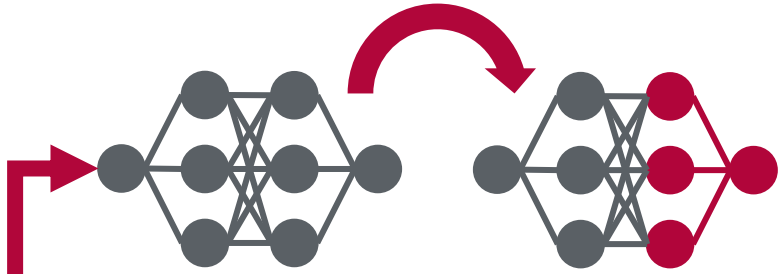


Baseline Model Development

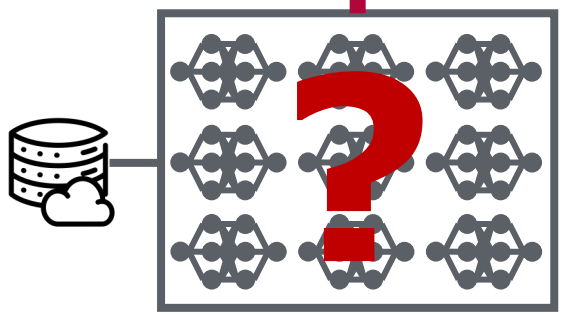
- Label or collect training data
- Fine-tune a “standard” model
- **Model Search**

→ **Fine-Tuning [25, 26]**

- ⊕ Less training data
- ⊕ Faster convergence
- ⊕ Higher accuracy



Transfer Learning
(fine-tuning)



Models in a model store

Poodle – Preliminary Results

Costs 

Inference Time 

Accuracy 

Model Development 

Poodle – Preliminary Results

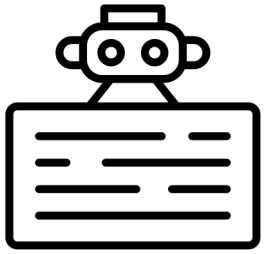
Costs 

- JITR amortizes overhead
- Significant cost savings

Poodle – Preliminary Results

Costs

- JITR amortizes overhead
- Significant cost savings



VS

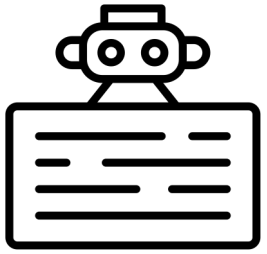
Just-in-Time Model Replacement

- Wrap requests in an additional prompt → LLM

Poodle – Preliminary Results

Costs

- JITR amortizes overhead
- Significant cost savings



VS

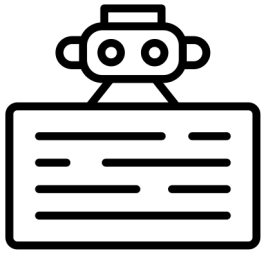
Just-in-Time Model Replacement

- Wrap requests in an additional prompt → LLM
- Collect 5,000 labels
- Training of a small model
- Use a small model instead

Poodle – Preliminary Results

Costs $

- JITR amortizes overhead
- Significant cost savings



VS

Just-in-Time Model Replacement

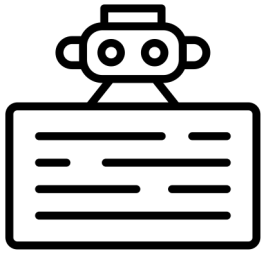
- Wrap requests in an additional prompt → LLM
- Collect 5,000 labels
- Training of a small model
- Use a small model instead

Model	Input	Output	Provider
GPT-4.1	$2.00	$8.00	OpenAI
GPT-4.1-nano	$0.10	$0.40	OpenAI
Llama 405B Turbo	$3.50	$3.50	TogetherAI
Llama 8B	$0.20	$0.20	TogetherAI
BERT 80M	$0.01	$0.01	TogetherAI

Poodle – Preliminary Results

Costs $

- JITR amortizes overhead
- Significant cost savings



VS

Just-in-Time Model Replacement

- Wrap requests in an additional prompt → LLM
- Collect 5,000 labels
- Training of a small model
- Use a small model instead

Model	Input	Output	Provider
GPT-4.1	$2.00	$8.00	OpenAI
GPT-4.1-nano	$0.10	$0.40	OpenAI
Llama 405B Turbo	$3.50	$3.50	TogetherAI
Llama 8B	$0.20	$0.20	TogetherAI
BERT 80M	$0.01	$0.01	TogetherAI

GPT-4.1 nano

GPT-4.1

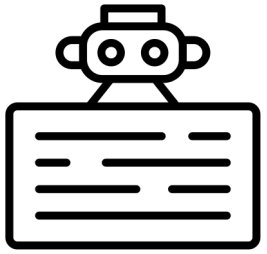
Break-even

Savings 1M requests

Poodle – Preliminary Results

Costs $

- JITR amortizes overhead
- Significant cost savings

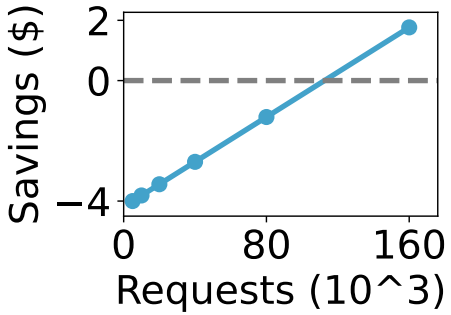


VS

Just-in-Time Model Replacement

- Wrap requests in an additional prompt → LLM
- Collect 5,000 labels
- Training of a small model
- Use a small model instead

Model	Input	Output	Provider
GPT-4.1	$2.00	$8.00	OpenAI
GPT-4.1-nano	$0.10	$0.40	OpenAI
Llama 405B Turbo	$3.50	$3.50	TogetherAI
Llama 8B	$0.20	$0.20	TogetherAI
BERT 80M	$0.01	$0.01	TogetherAI

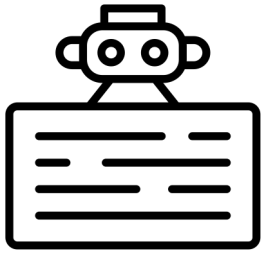


	GPT-4.1 nano	GPT-4.1
Break-even	100,000	
Savings 1M requests	$33	

Poodle – Preliminary Results

Costs $

- JITR amortizes overhead
- Significant cost savings

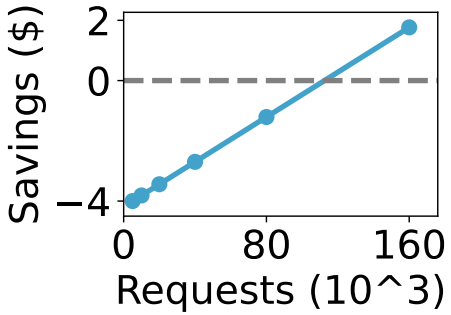


VS

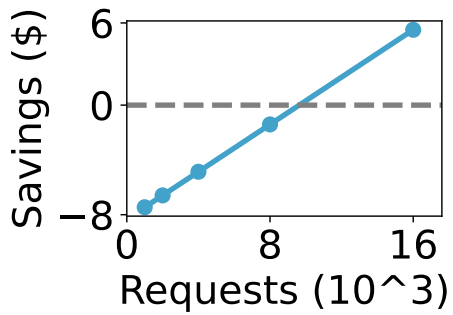
Just-in-Time Model Replacement

- Wrap requests in an additional prompt → LLM
- Collect 5,000 labels
- Training of a small model
- Use a small model instead

Model	Input	Output	Provider
GPT-4.1	$2.00	$8.00	OpenAI
GPT-4.1-nano	$0.10	$0.40	OpenAI
Llama 405B Turbo	$3.50	$3.50	TogetherAI
Llama 8B	$0.20	$0.20	TogetherAI
BERT 80M	$0.01	$0.01	TogetherAI



GPT-4.1 nano



GPT-4.1

	GPT-4.1 nano	GPT-4.1
Break-even	100,000	<10,000
Savings 1M requests	$33	$850

Poodle – Preliminary Results

Costs

- JITR amortizes overhead
- Significant cost savings

Inference Time

- Even for small LLMs JITR reduces inference time

Setup

Evaluate break-even (number of requests):
LLM (Llama-2-7B) VS **JITR** with Poodle (BERT)

IMDB dataset; NVIDIA RTX A5000 GPU; switch after 1K requests

(1) **LLM** and **JITR** break-even in less than 100,000 requests.

(2) BERT processes 240 more items per second.

(3) For 1M requests Poodle takes 15% less time; For 2M requests more than 1 hr.

(4) Real LLMs orders of magnitude larger than Llama-2-7B

Poodle – Preliminary Results

Costs

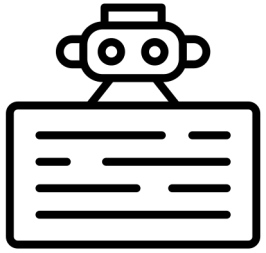
- JITR amortizes overhead
- Significant cost savings

Inference Time

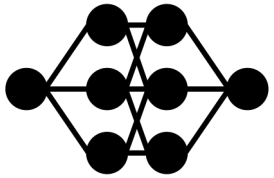
- Even for small LLMs JITR reduces inference time

Accuracy

- Small, specialized models reach competitive accuracy



Llama 2 7B
Accuracy: 0.93
1024 Tokens



BERT, no tuning
Accuracy: 0.89 (0.92)
256 Tokens

	Ground Truth		LLM-Generated	
# Items	Accuracy	Epochs	Accuracy	Epochs
500	0.86	9	0.88	6
1,000	0.88	5	0.88	7
2,000	0.89	3	0.88	5
5,000	0.90	3	0.90	2

Related work [6, 7, 10, 12, 23]
Across multiple studies, fine-tuned small models outperform LLMs for text classification tasks

Poodle – Preliminary Results

Costs 

- JITR amortizes overhead
- Significant cost savings

Inference Time 

- Even for small LLMs JITR reduces inference time

Accuracy 

- Small, specialized models reach competitive accuracy

Model Development 

- Model Search → Fine-Tuning outperforms alternative approaches in dev. time, accuracy, and required data

Poodle – Preliminary Results

Llama 2 7B
Accuracy: 0.93

Costs

- JITR amortizes overhead
- Significant cost savings

Inference Time

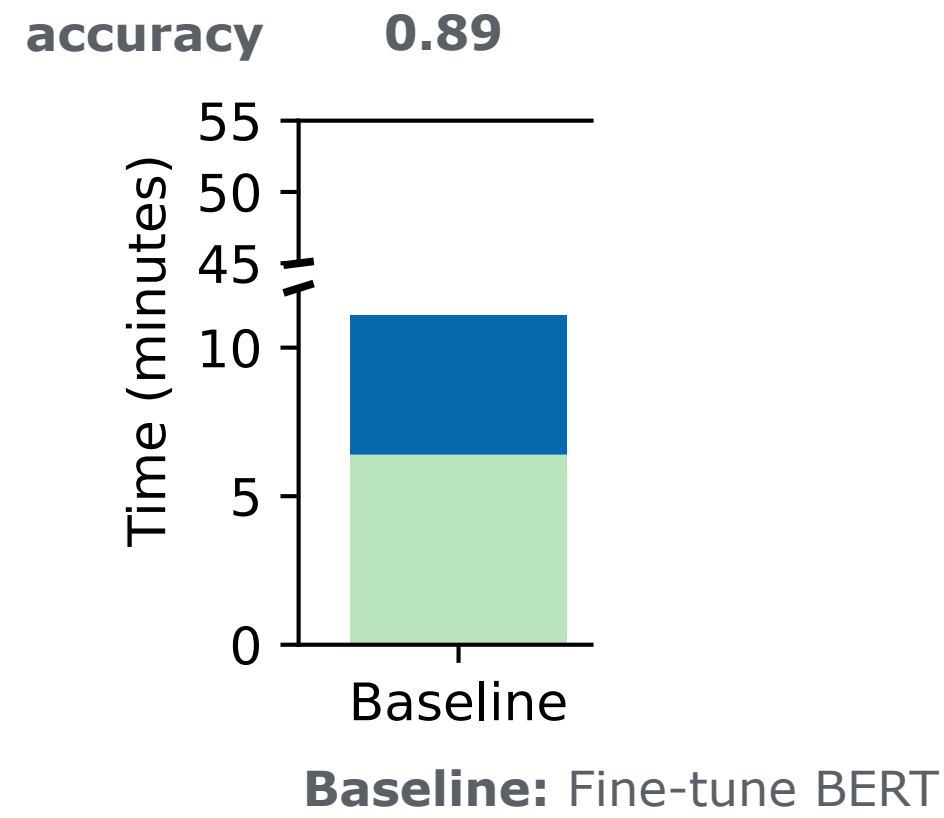
- Even for small LLMs JITR reduces inference time

Accuracy

- Small, specialized models reach competitive accuracy

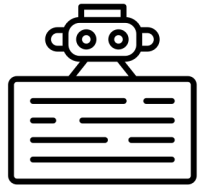
Model Development

- Model Search → Fine-Tuning outperforms alternative approaches in dev. time, accuracy, and required data



Baseline: Fine-tune BERT

Poodle – Preliminary Results



Llama 2 7B
Accuracy: 0.93

Costs

- JITR amortizes overhead
- Significant cost savings

Inference Time

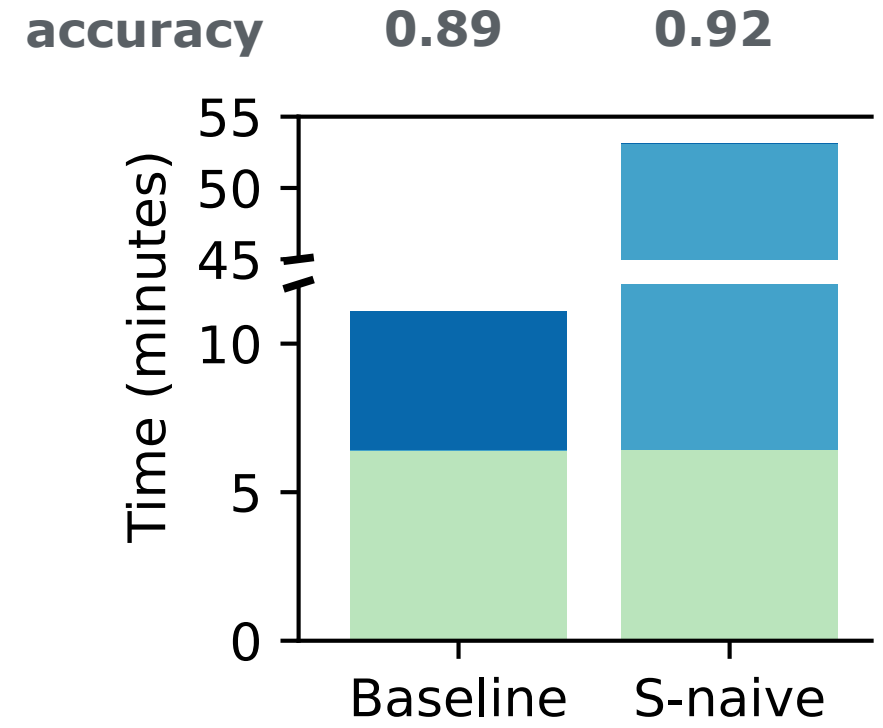
- Even for small LLMs JITR reduces inference time

Accuracy

- Small, specialized models reach competitive accuracy

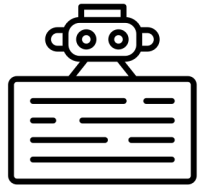
Model Development

- Model Search → Fine-Tuning outperforms alternative approaches in dev. time, accuracy, and required data



Baseline: Fine-tune BERT
S-naïve: Exhaustive search over 10 hand-selected models from HuggingFace

Poodle – Preliminary Results



Llama 2 7B
Accuracy: 0.93

Costs

- JITR amortizes overhead
- Significant cost savings

Inference Time

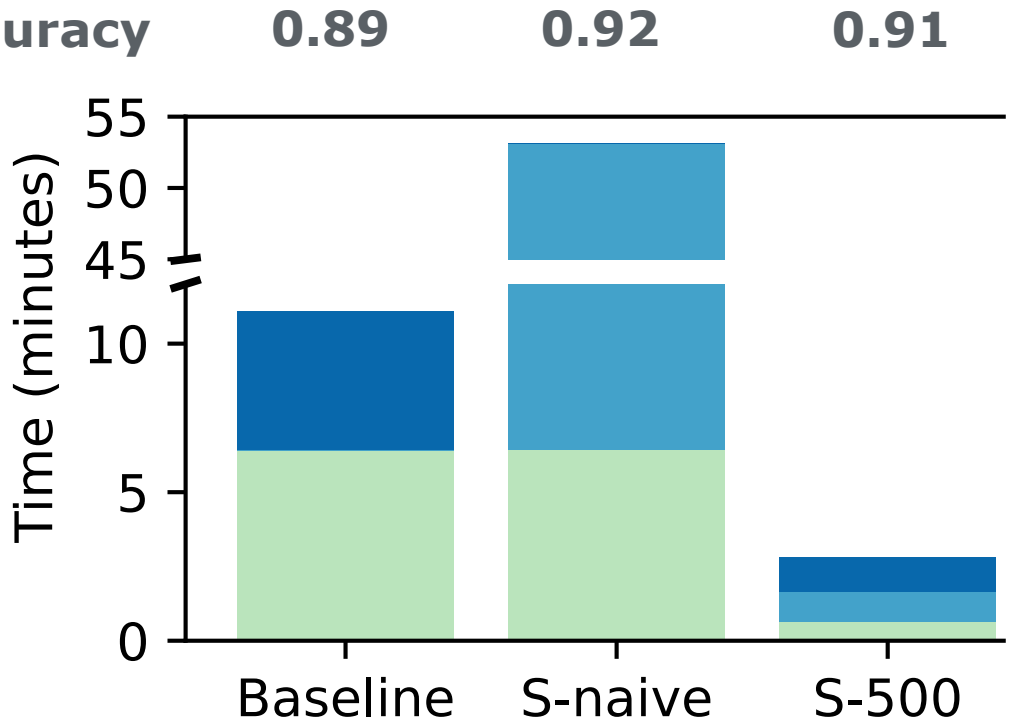
- Even for small LLMs JITR reduces inference time

Accuracy

- Small, specialized models reach competitive accuracy

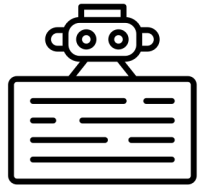
Model Development

- Model Search → Fine-Tuning outperforms alternative approaches in dev. time, accuracy, and required data



Baseline: Fine-tune BERT
S-naïve: Exhaustive search over 10 hand-selected models from HuggingFace
S-500: Search + fine-tune on 500 items

Poodle – Preliminary Results



Llama 2 7B
Accuracy: 0.93

Costs

- JITR amortizes overhead
- Significant cost savings

Inference Time

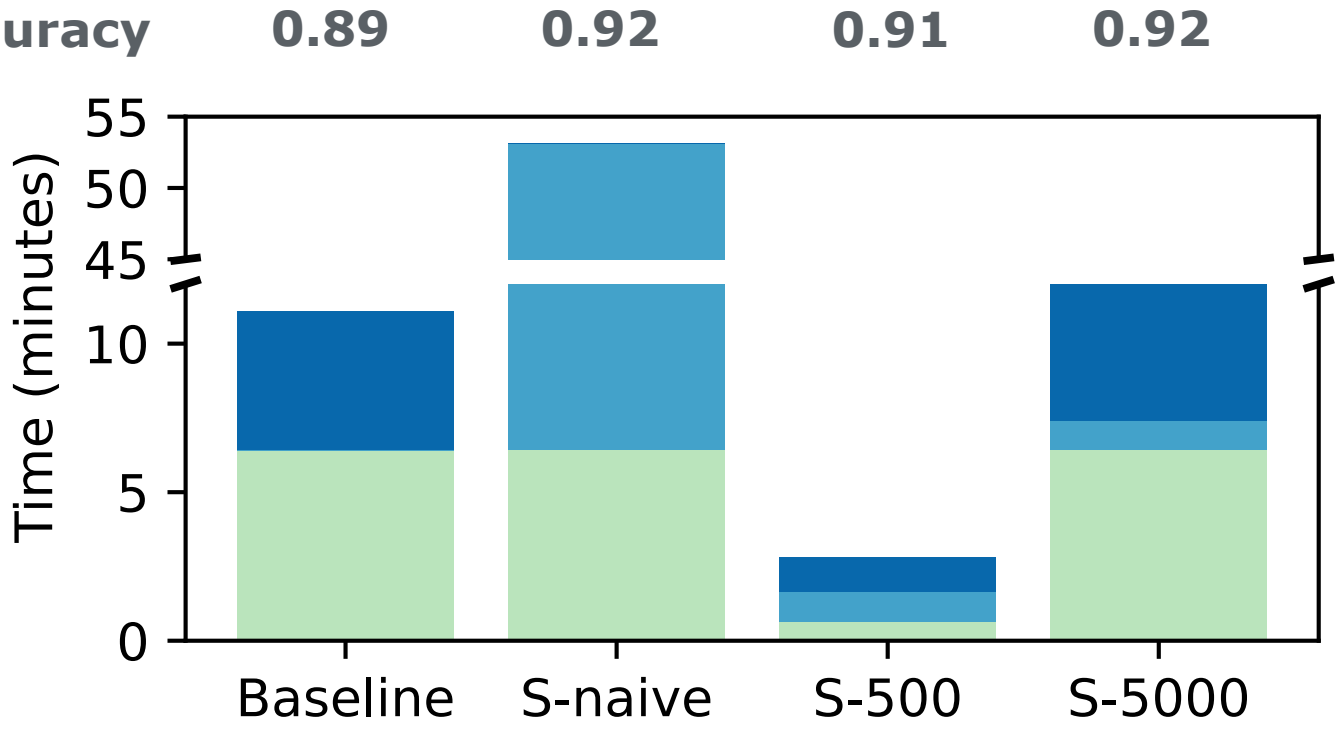
- Even for small LLMs JITR reduces inference time

Accuracy

- Small, specialized models reach competitive accuracy

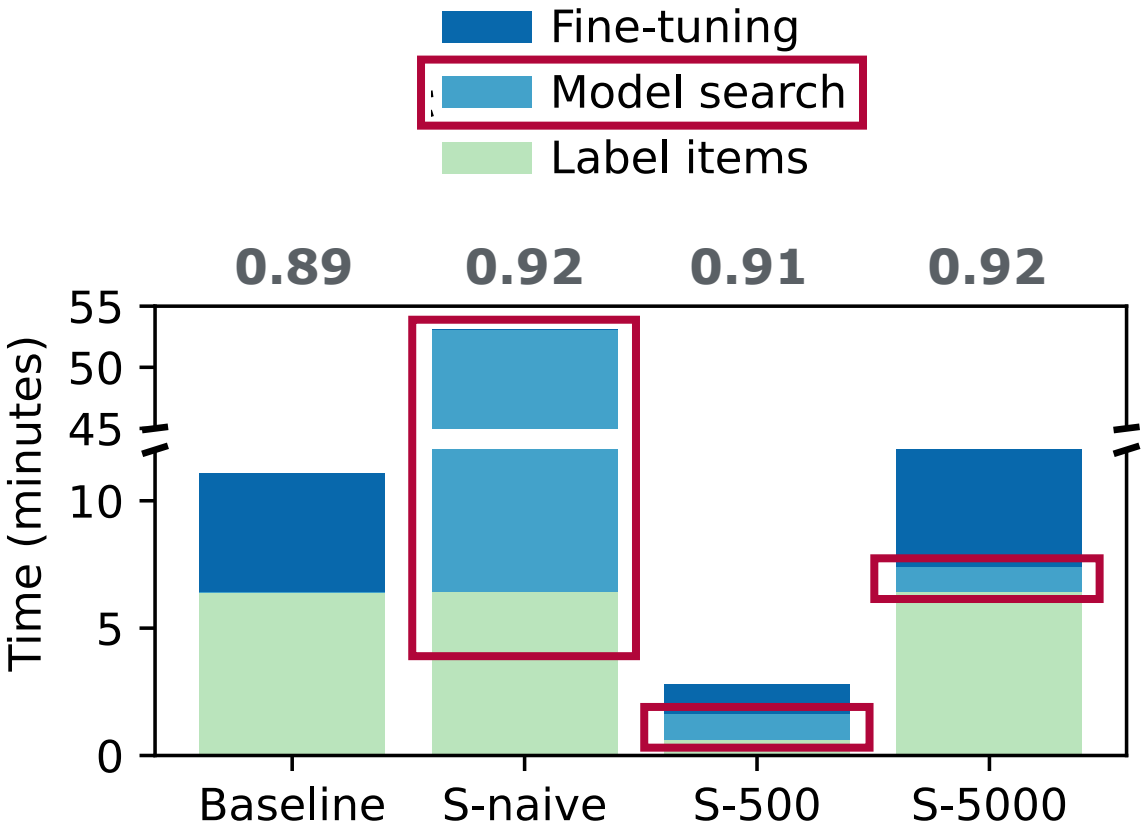
Model Development

- Model Search → Fine-Tuning outperforms alternative approaches in dev. time, accuracy, and required data



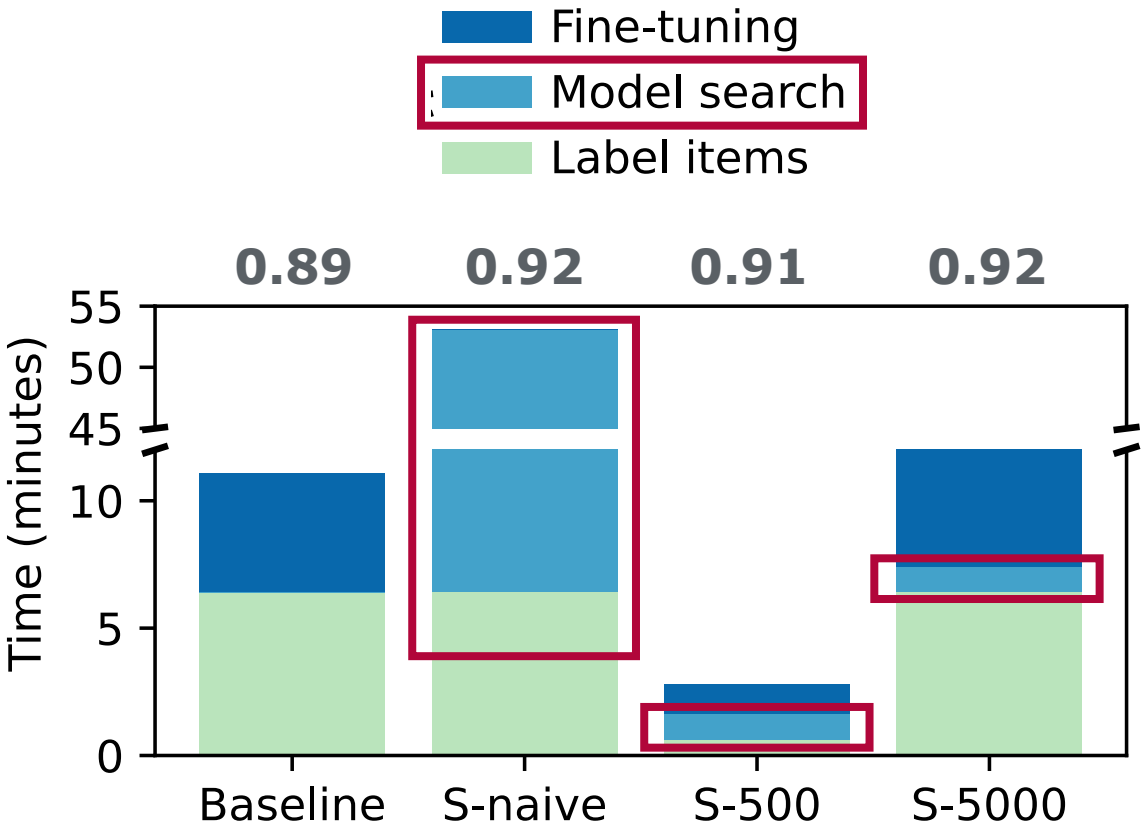
Baseline: Fine-tune BERT
S-naïve: Exhaustive search over 10 hand-selected models from HuggingFace
S-500: Search + fine-tune on 500 items
S-5000: Search on 500 items
+ fine-tune on 5,000 items

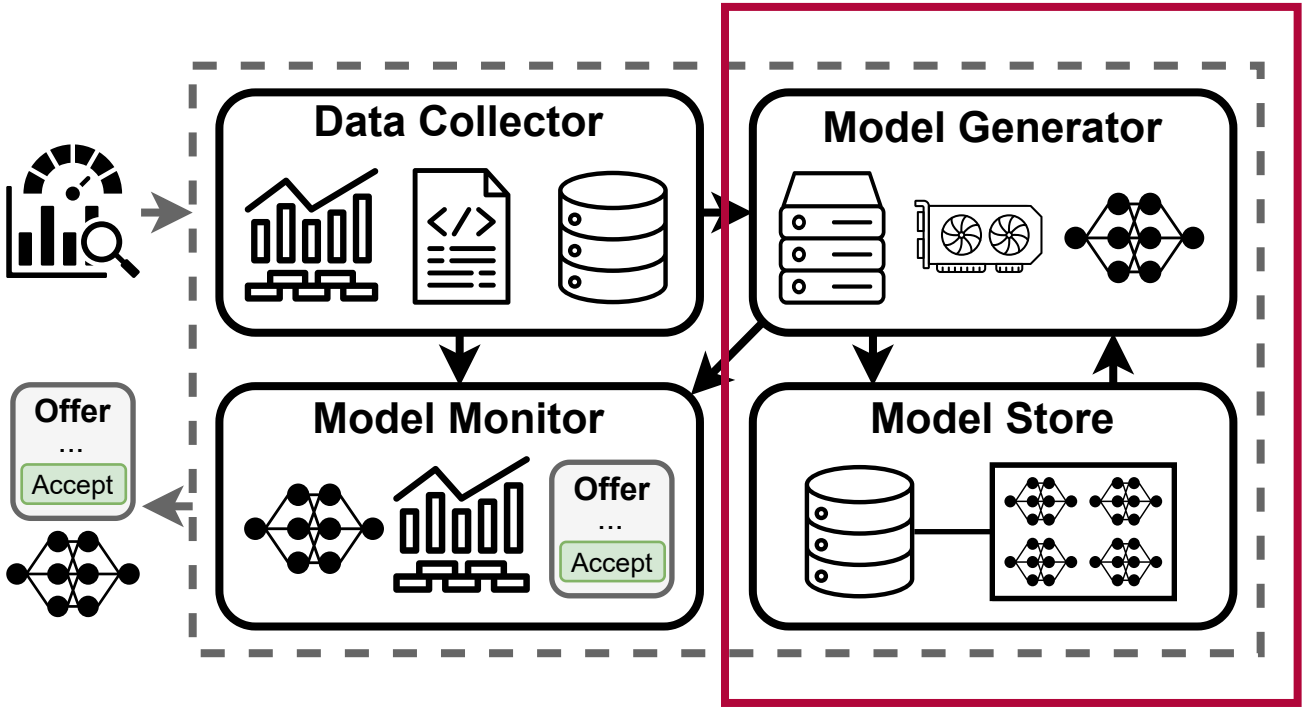
The Road Ahead



Baseline: Fine-tune BERT
S-naïve: Exhaustive search over **10 hand-selected models** from HuggingFace
S-500: Search + fine-tune on 500 items
S-5000: Search on 500 items
+ fine-tune on 5,000 items

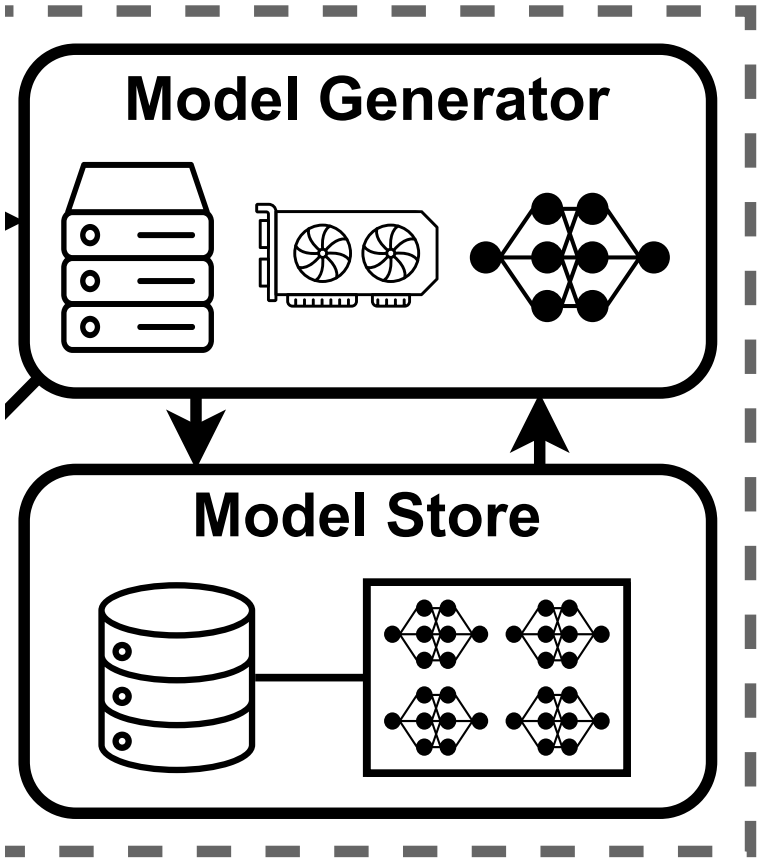
The Road Ahead





Baseline: Fine-tune BERT
S-naïve: Exhaustive search over **10 hand-selected models** from HuggingFace
S-500: Search + fine-tune on 500 items
S-5000: Search on 500 items
+ fine-tune on 5,000 items

Co-Design Model Search and Model Store



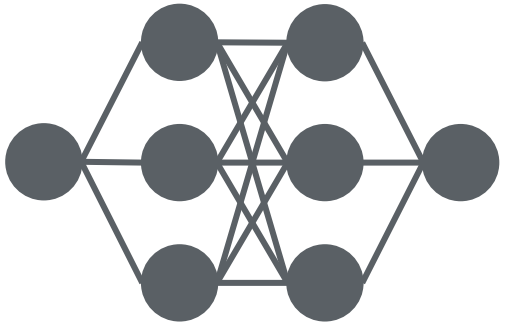
Two main directions

(1) Co-design model store and model search techniques

(2) Approximate model search

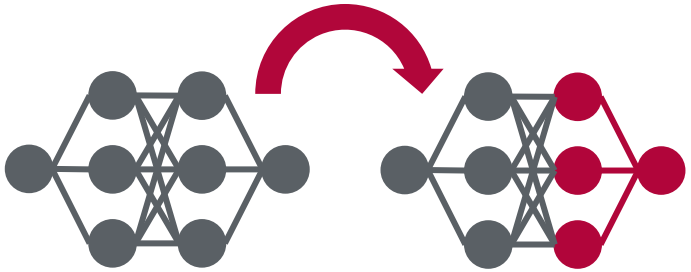
Overall Problem

ML Task
(in the form of data)



Overall Problem

ML Task
(in the form of data)

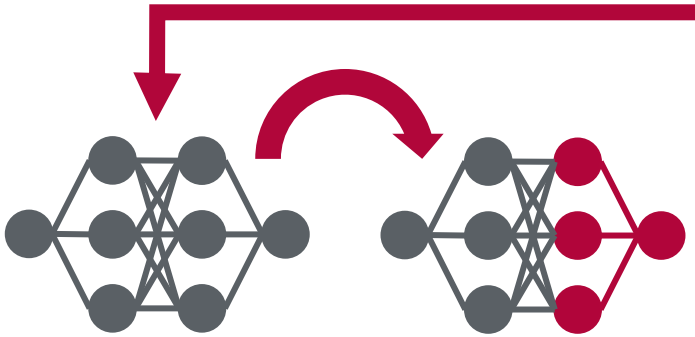


Transfer Learning
(fine-tuning)

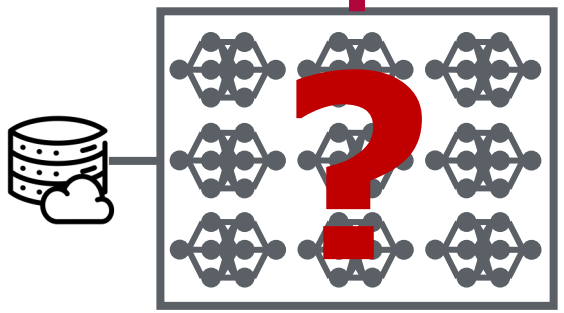
- Higher accuracy
- Less data
- Faster convergence

Overall Problem

ML Task
(in the form of data)



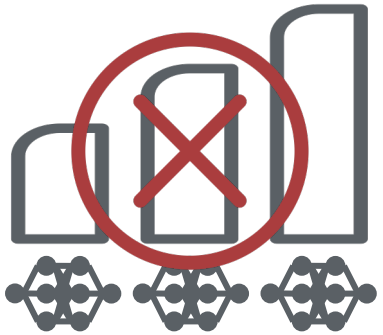
Transfer Learning
(fine-tuning)



Set of Models
(in a model store)

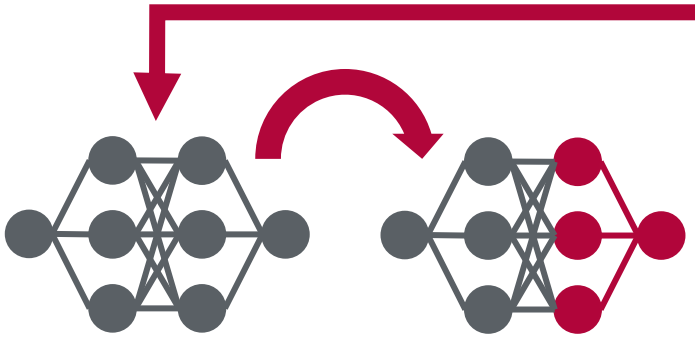
Model Search
(find best model to start with)

- Google (brain) [5, 11, 12]
- AWS [13, 14, 15, 16]
- Facebook AI [16]
- ML community [9, 17, …]
- Database community [6, 18]

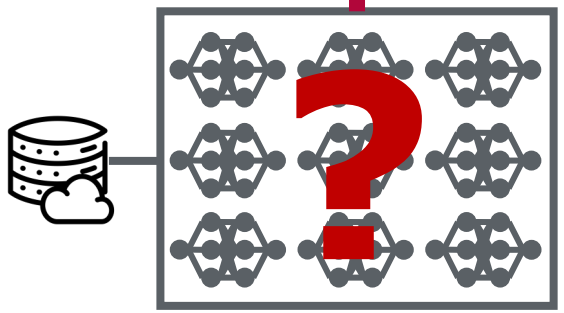


Overall Problem

ML Task
(in the form of data)



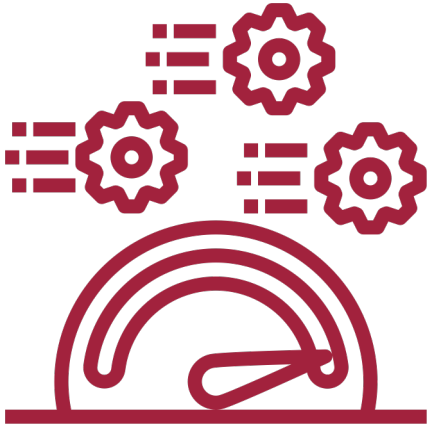
Transfer Learning
(fine-tuning)



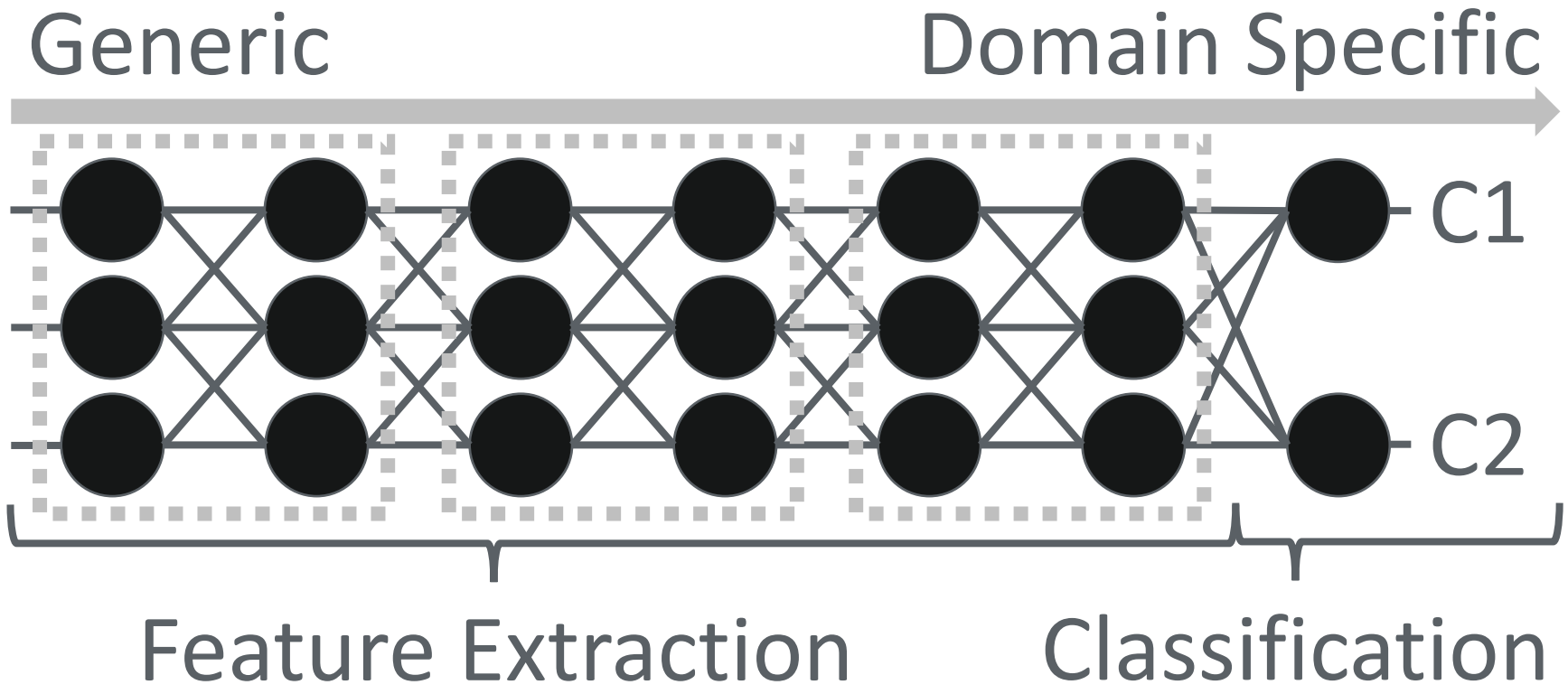
Set of Models
(in a model store)

Model Search
(find best model to start with)

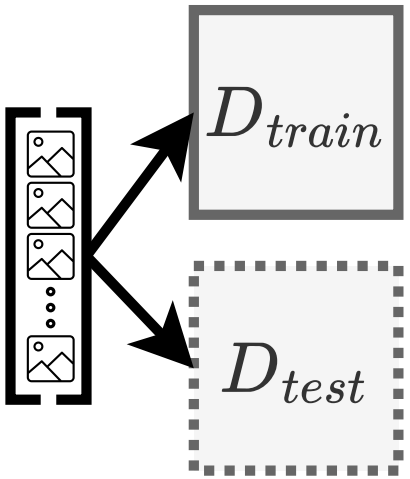
- Google (brain) [5, 11, 12]
- AWS [13, 14, 15, 16]
- Facebook AI [16]
- ML community [9, 17, …]
- Database community [6, 18]



Background – Structure of DL Model – [1, 2, 3]

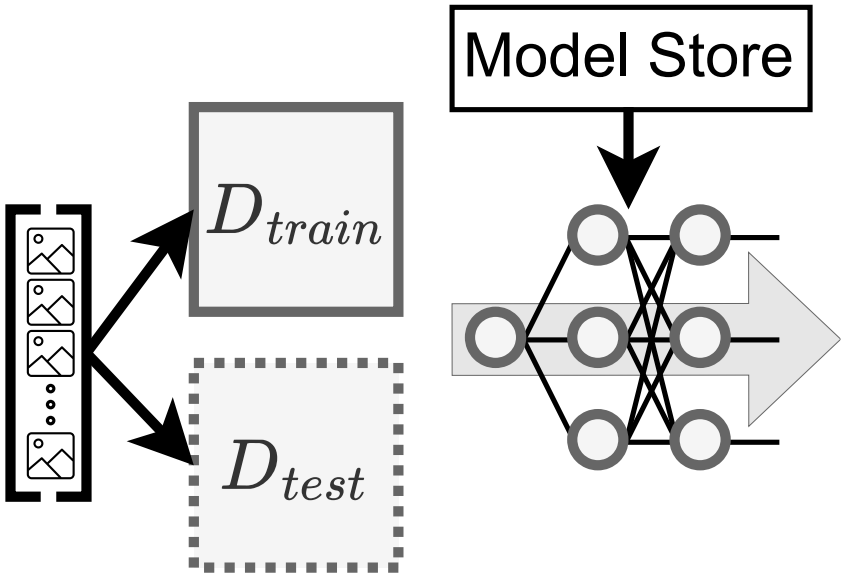


Background – Feature Based Model Search – [5, 6]



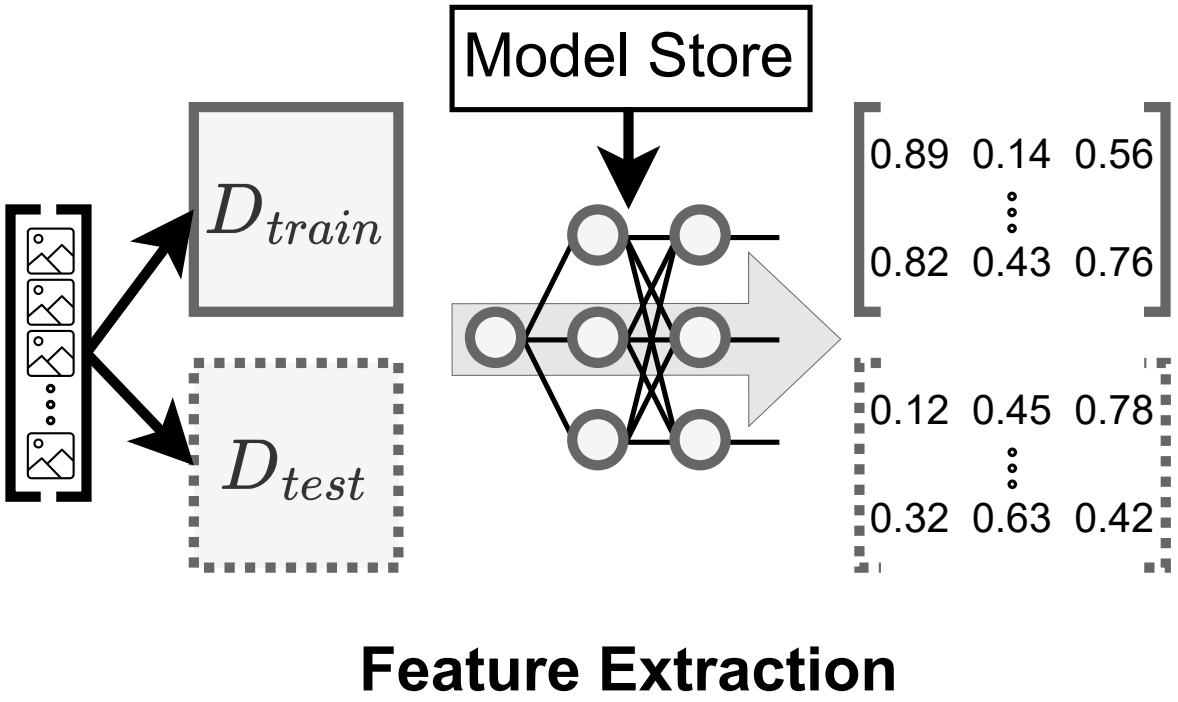
Steps: **1:** prepare data,

Background – Feature Based Model Search – [5, 6]



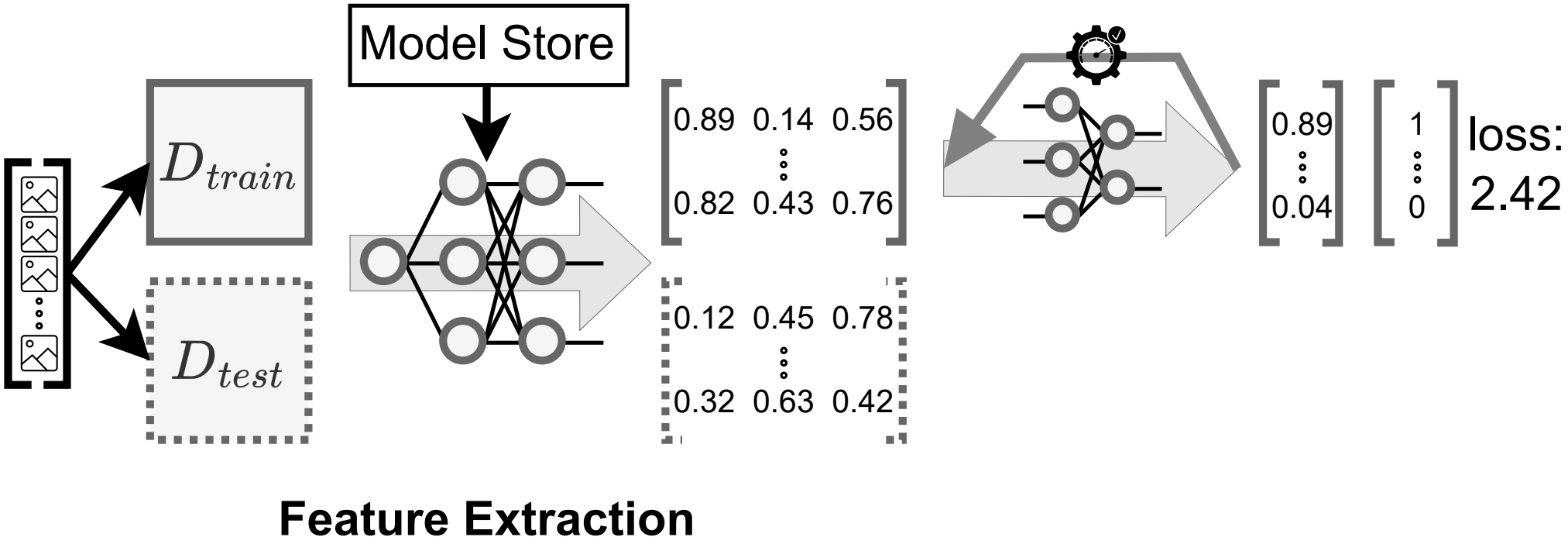
Steps: **1:** prepare data, **2:** prepare model,

Background – Feature Based Model Search – [5, 6]



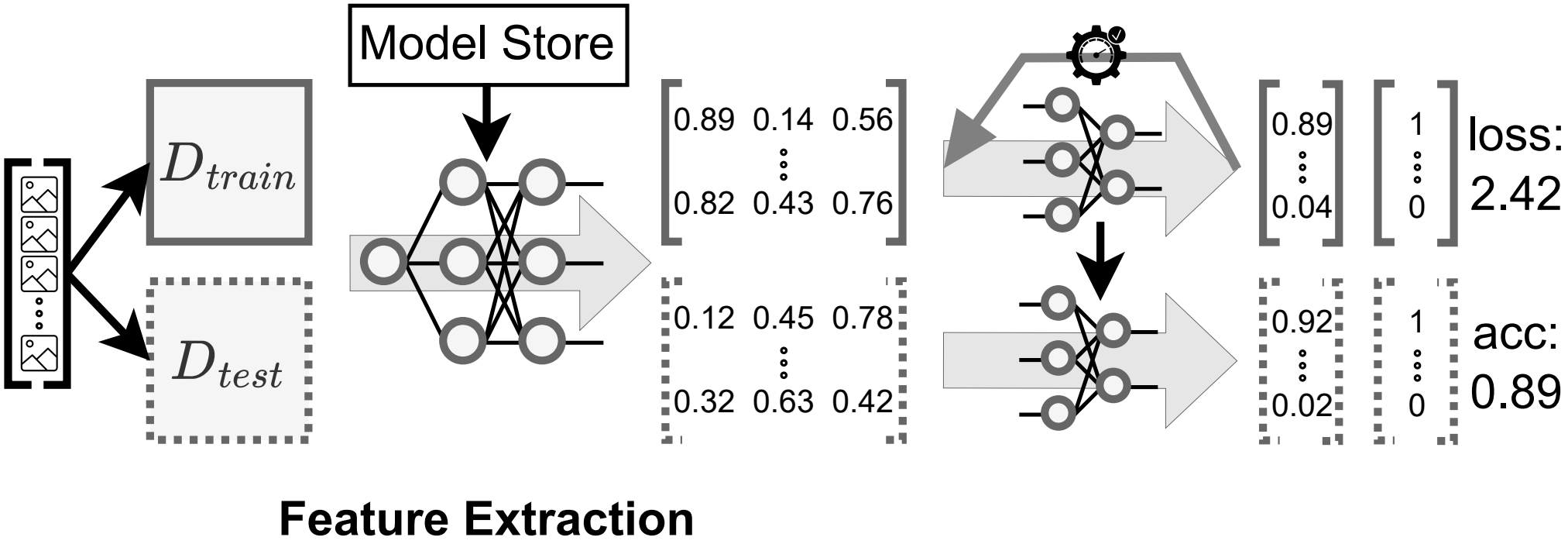
Steps: **1:** prepare data, **2:** prepare model, **3:** inference,

Background – Feature Based Model Search – [5, 6]



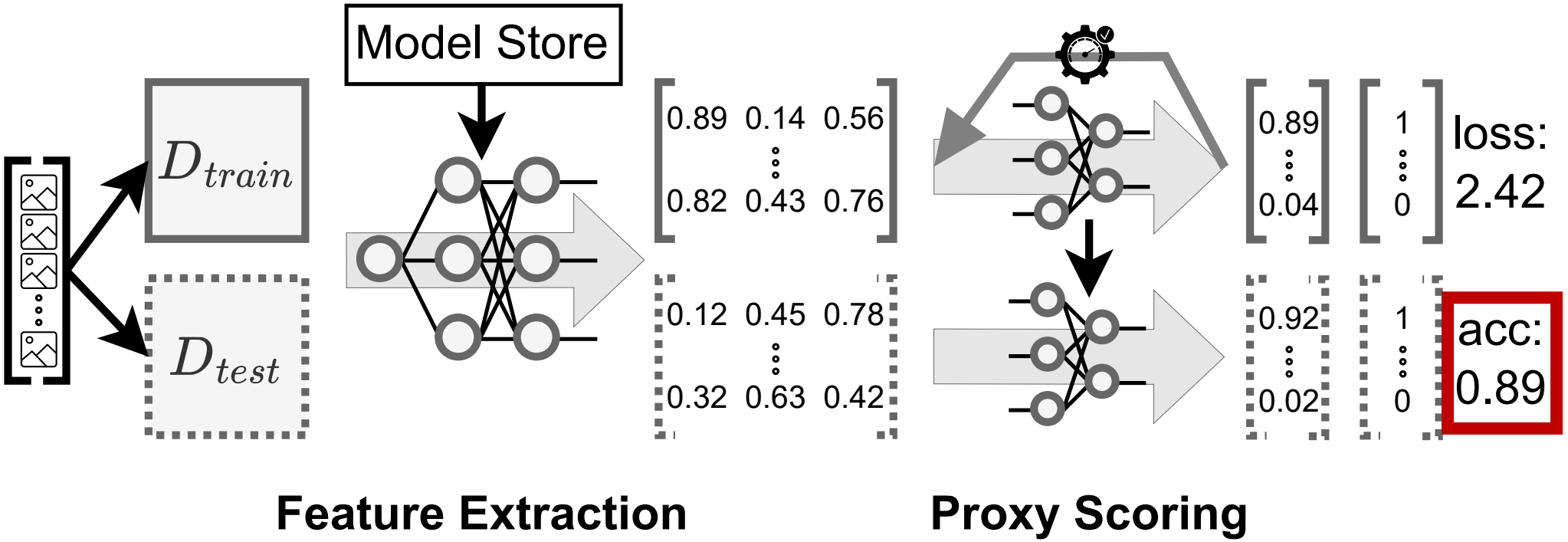
Steps: **1:** prepare data, **2:** prepare model, **3:** inference, **4:** proxy scoring

Background – Feature Based Model Search – [5, 6]



Steps: **1:** prepare data, **2:** prepare model, **3:** inference, **4:** proxy scoring

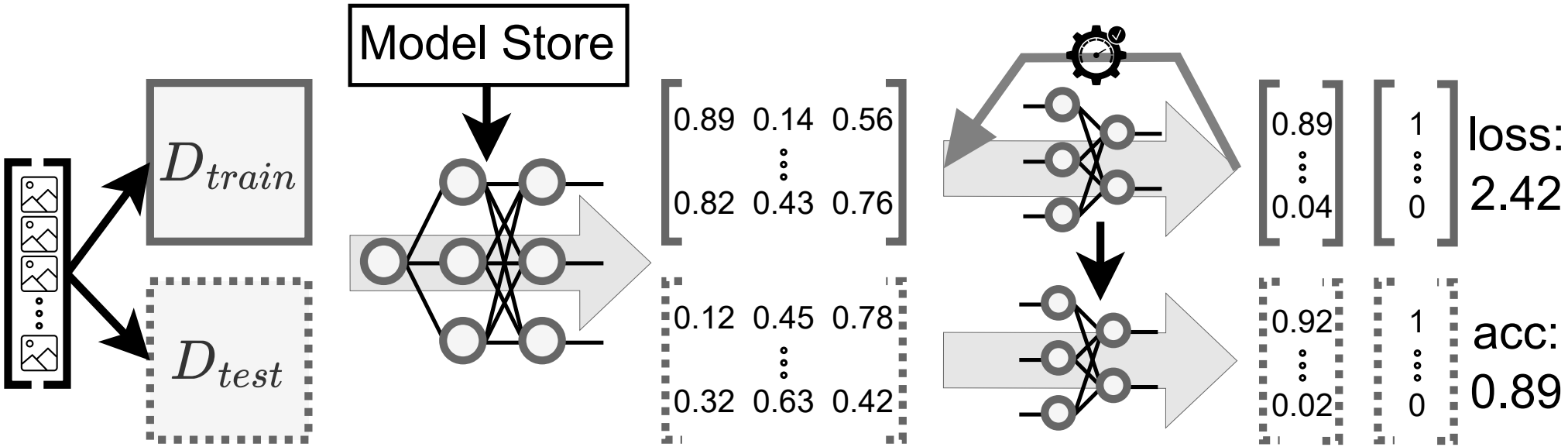
Background – Feature Based Model Search – [5, 6]



Rank models by proxy score (e.g. accuracy)

Steps: **1:** prepare data, **2:** prepare model, **3:** inference, **4:** proxy scoring

Background – Optimizations



Feature Extraction

SHiFT [6]

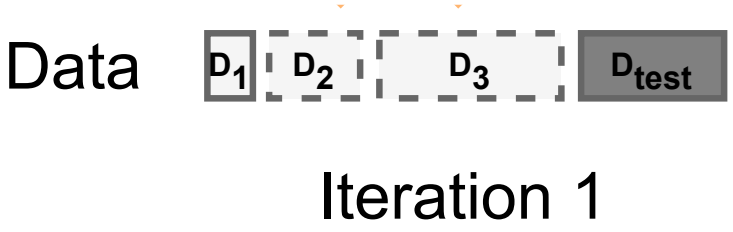
Nautilus [7]

…

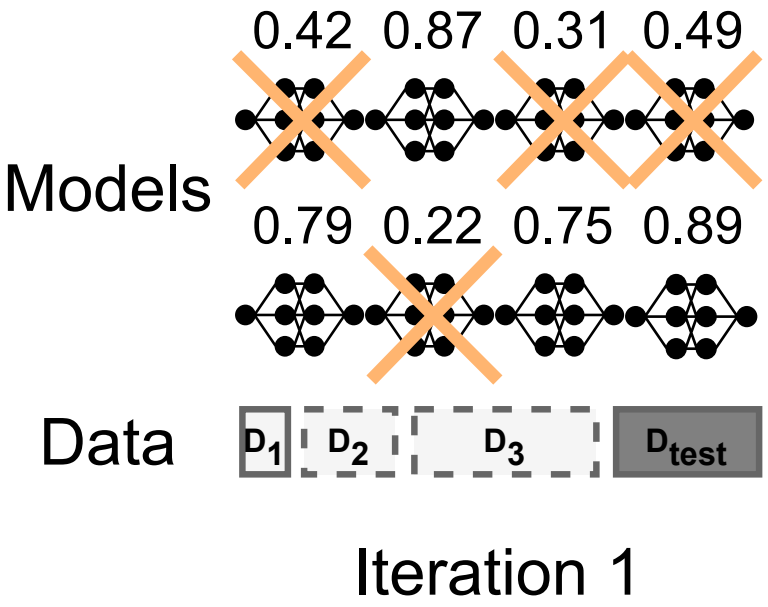
Proxy Scoring

Renggli et al. [5]
Yandong Li et al. [8]
Bolya et al. [9]
Hao Li et al. [10]
…

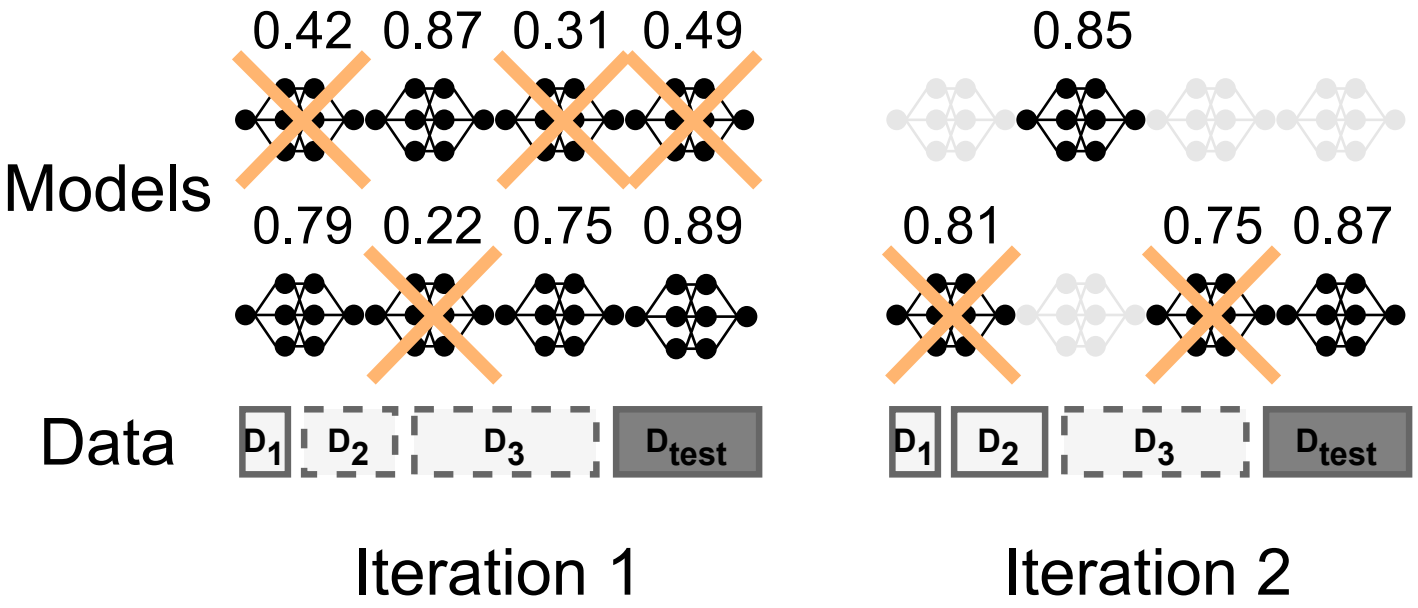
Background – Successive Halving – [6]



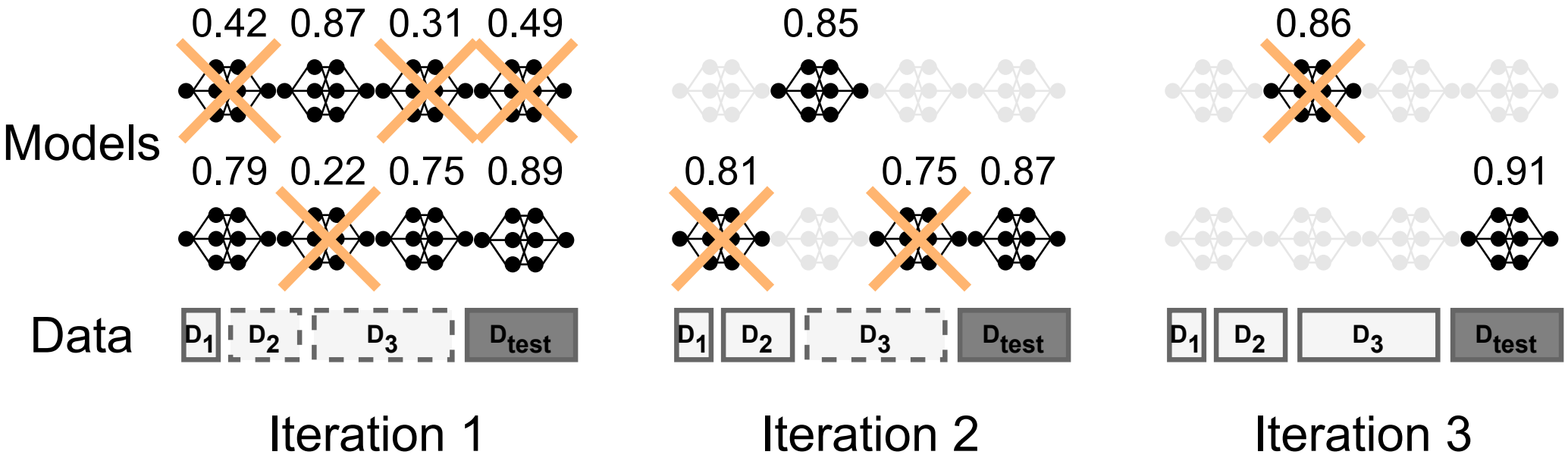
Background – Successive Halving – [6]



Background – Successive Halving – [6]

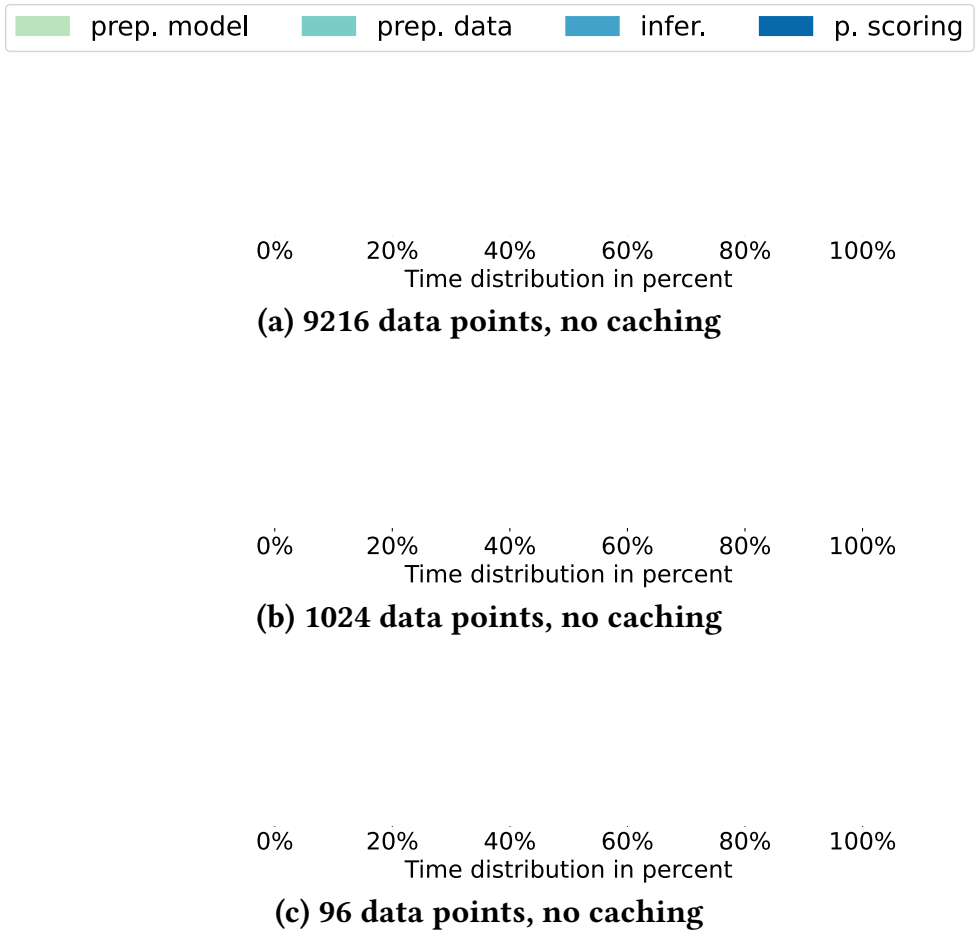


Background – Successive Halving – [6]



Approach – Bottlenecks

Approach – Bottlenecks

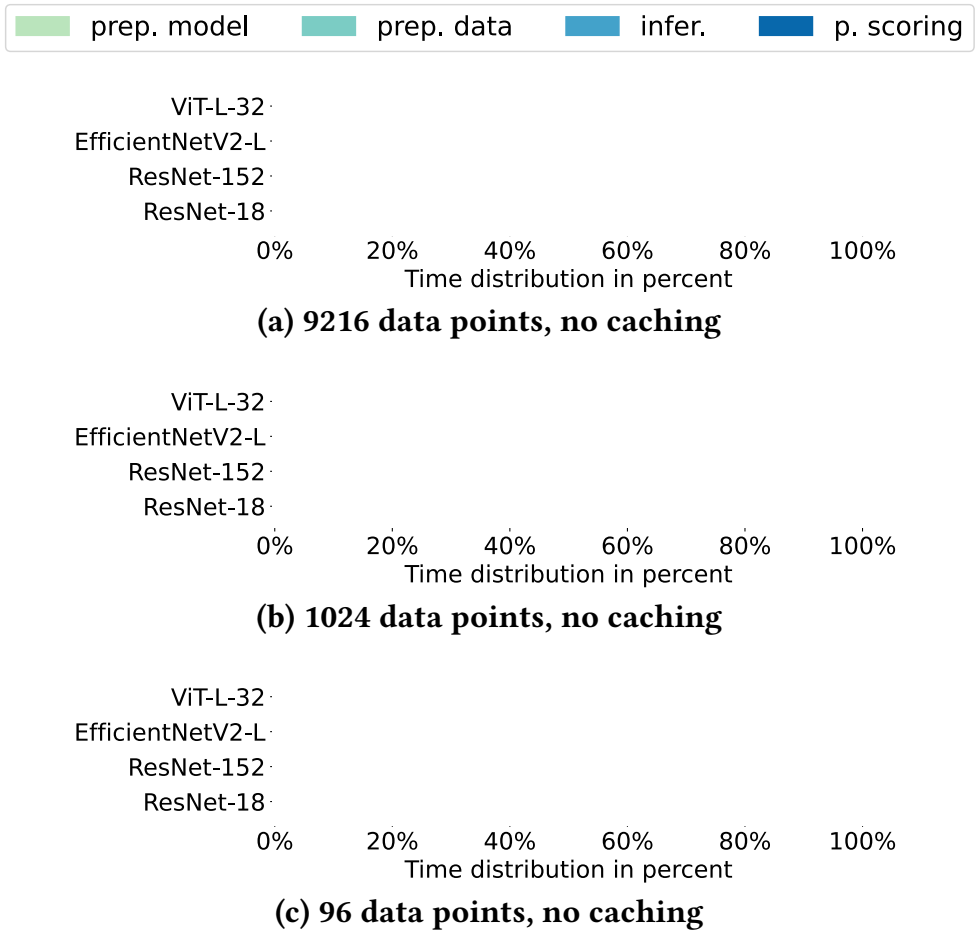


(a) 9216 data points, no caching

(b) 1024 data points, no caching

(c) 96 data points, no caching

Approach – Bottlenecks

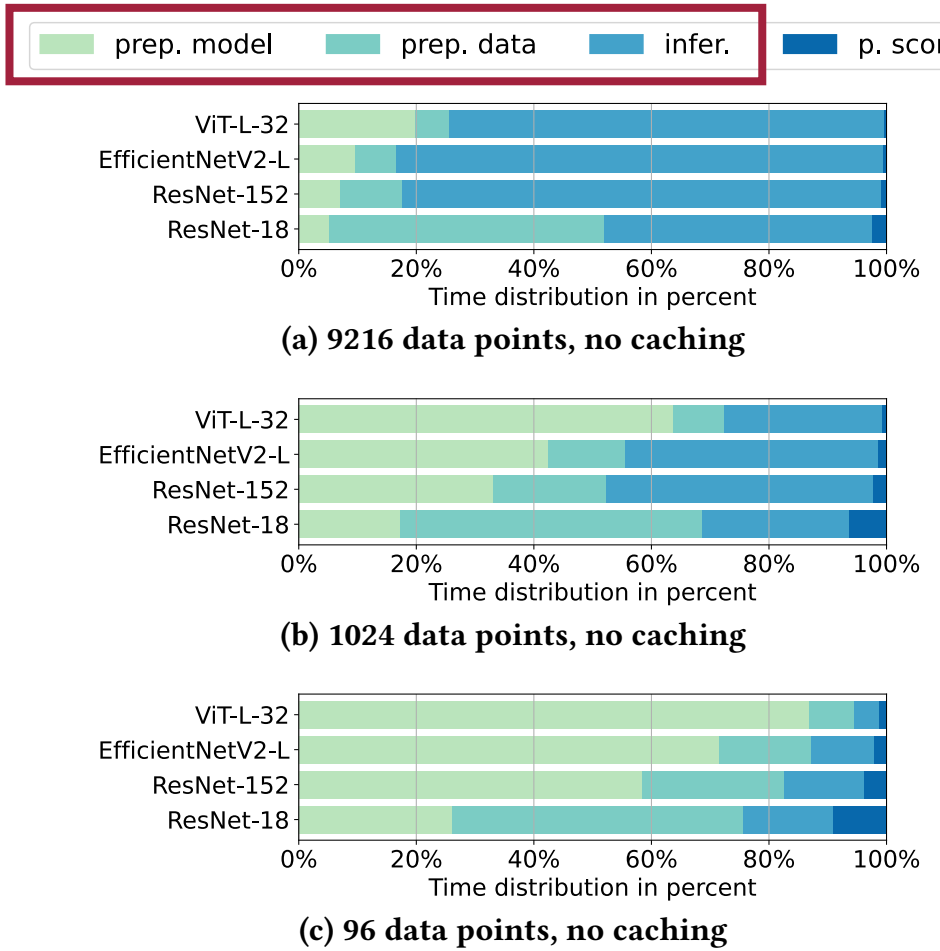


(a) 9216 data points, no caching

(b) 1024 data points, no caching

(c) 96 data points, no caching

Approach – Bottlenecks



(a) 9216 data points, no caching

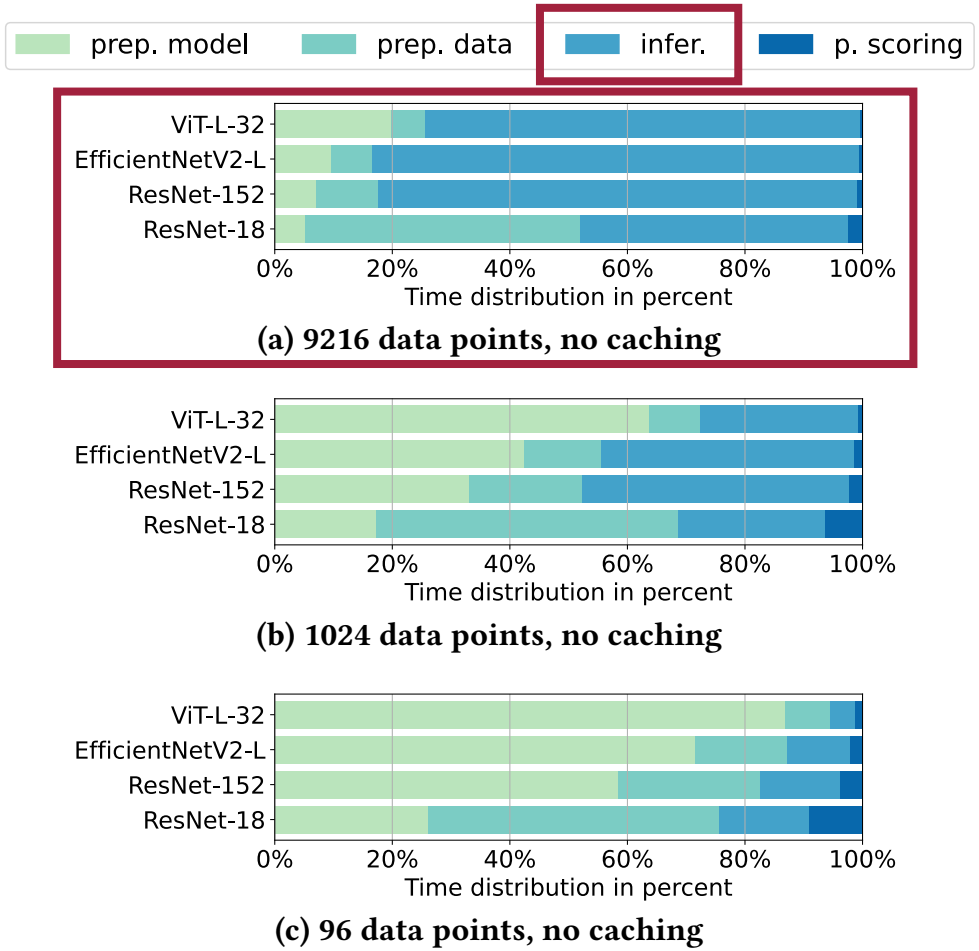
(b) 1024 data points, no caching

(c) 96 data points, no caching

Bottlenecks

- Overall: **feature extraction**

Approach – Bottlenecks



(a) 9216 data points, no caching

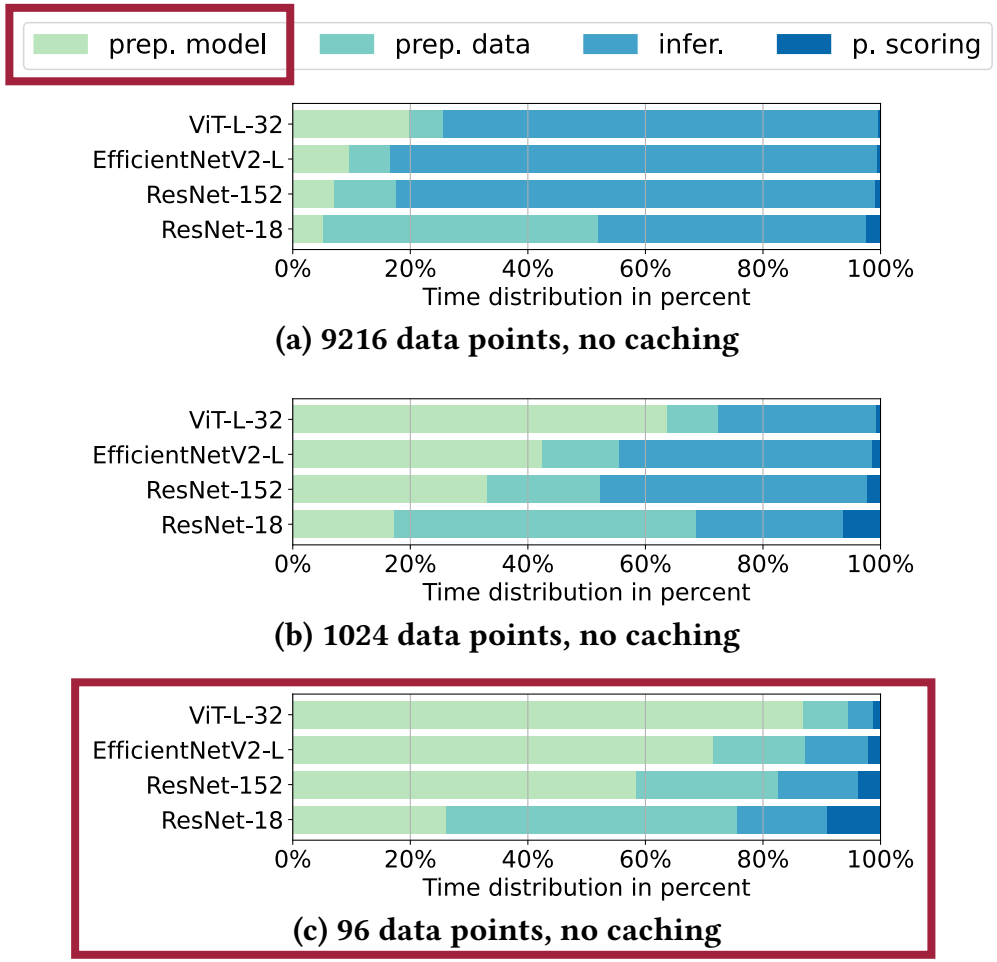
(b) 1024 data points, no caching

(c) 96 data points, no caching

Bottlenecks

- Overall: **feature extraction**
- For large datasets: **inference**

Approach – Bottlenecks



(a) 9216 data points, no caching

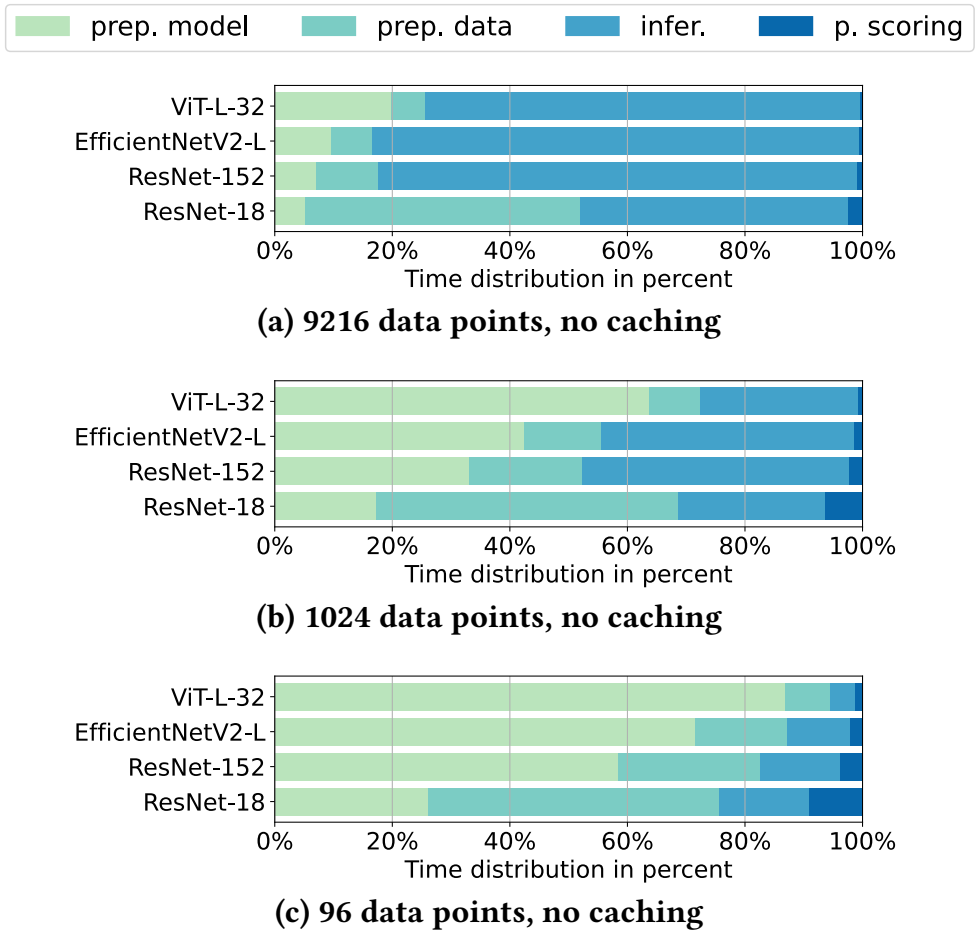
(b) 1024 data points, no caching

(c) 96 data points, no caching

Bottlenecks

- Overall: **feature extraction**
- For large datasets: **inference**
- For small datasets: **model preparation**

Approach – Bottlenecks



(a) 9216 data points, no caching

(b) 1024 data points, no caching

(c) 96 data points, no caching

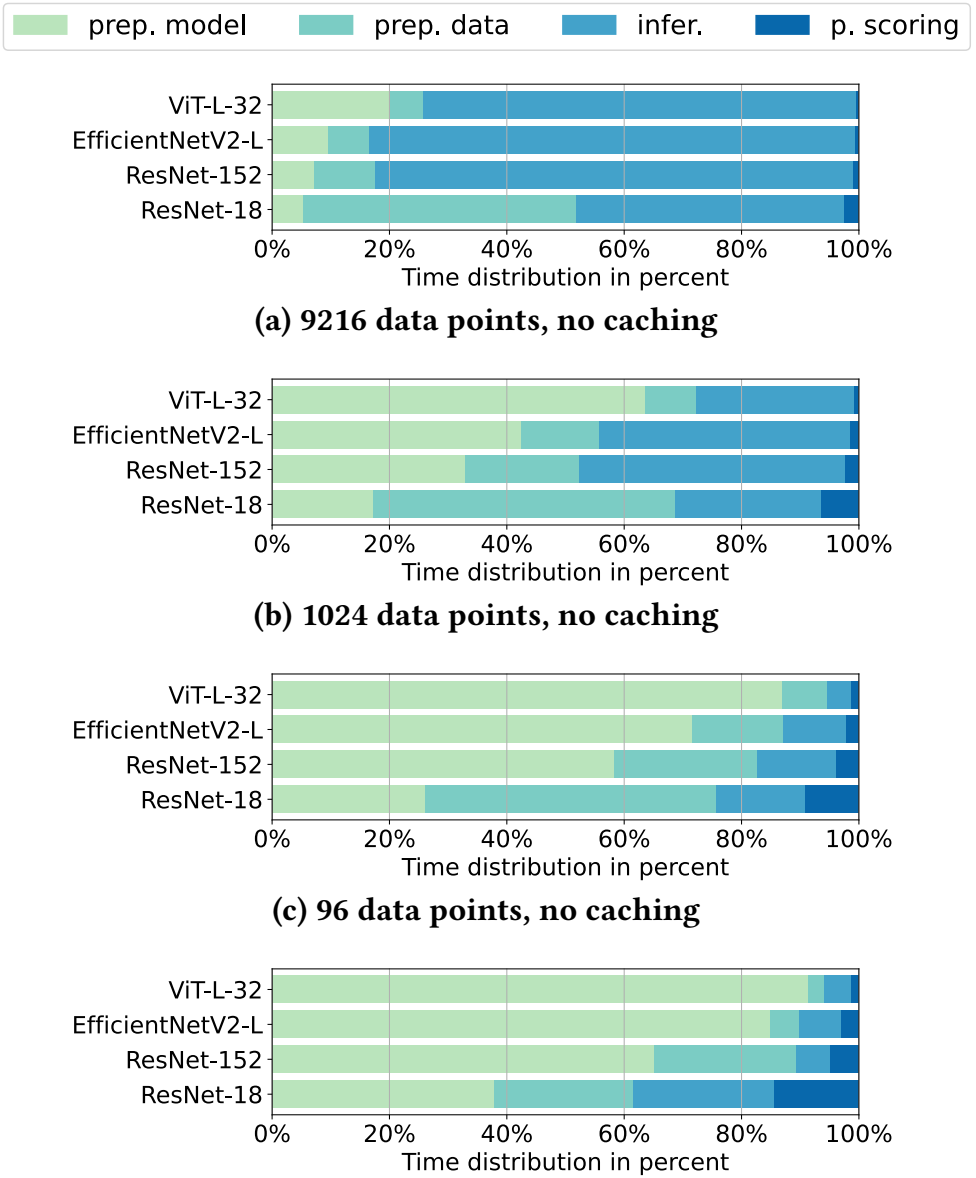
Bottlenecks

- Overall: **feature extraction**
- For large datasets: **inference**
- For small datasets: **model preparation**

Model Overlap

- Number of frozen layers is hyperparameter
- Patterns: last few, 25% of layers, and 50% of layers [19, 20, 21]
- Own analysis of over ~2800 models on HF

Approach – Bottlenecks



(a) 9216 data points, no caching

(b) 1024 data points, no caching

(c) 96 data points, no caching

(d) 1024 data points, reusing cached input for the 3rd last block

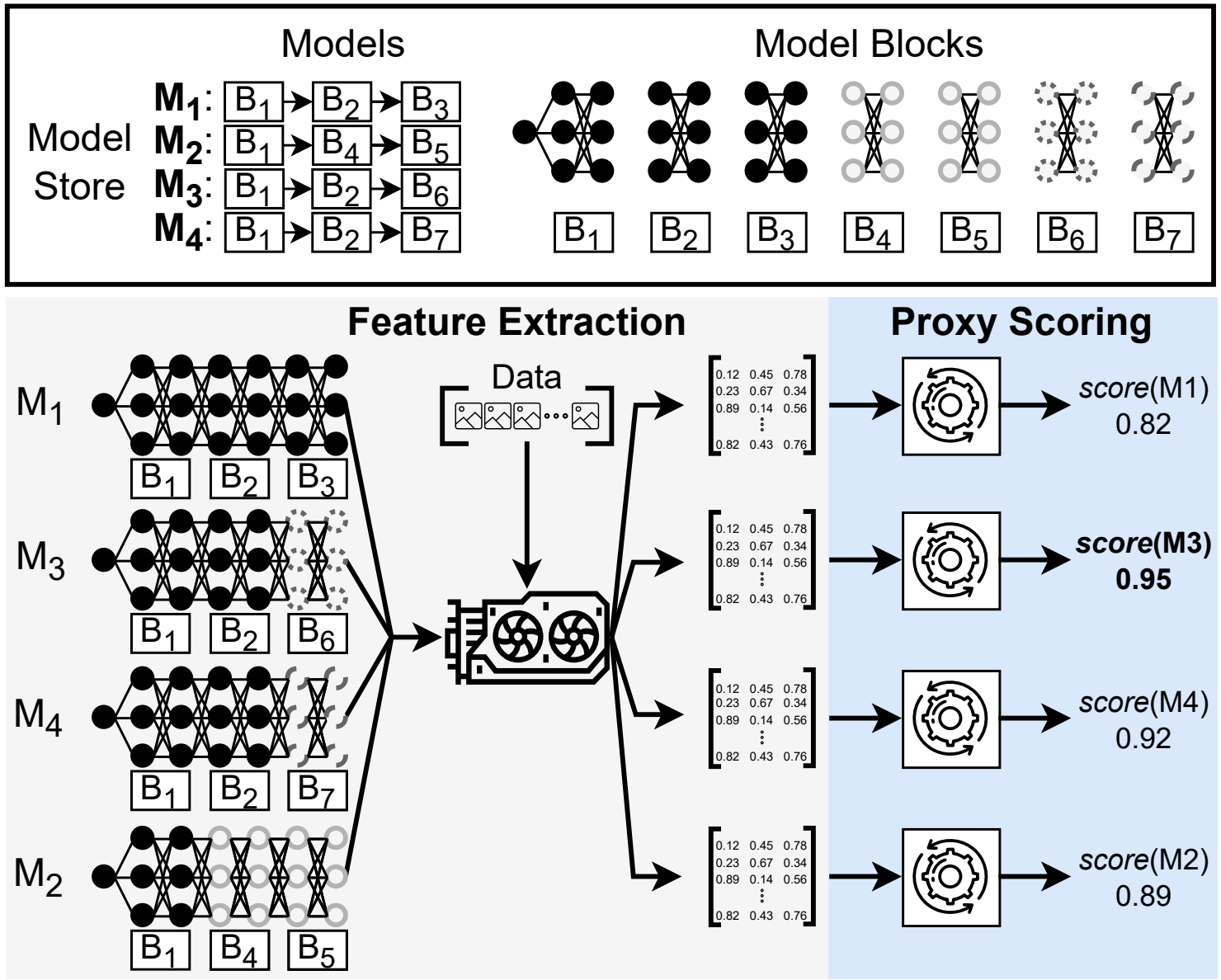
Bottlenecks

- Overall: **feature extraction**
- For large datasets: **inference**
- For small datasets: **model preparation**
- For partial models: **model preparation**

Model Overlap

- Number of frozen layers is hyperparameter
- Patterns: last few, 25% of layers, and 50% of layers [19, 20, 21]
- Own analysis of over ~2800 models on HF

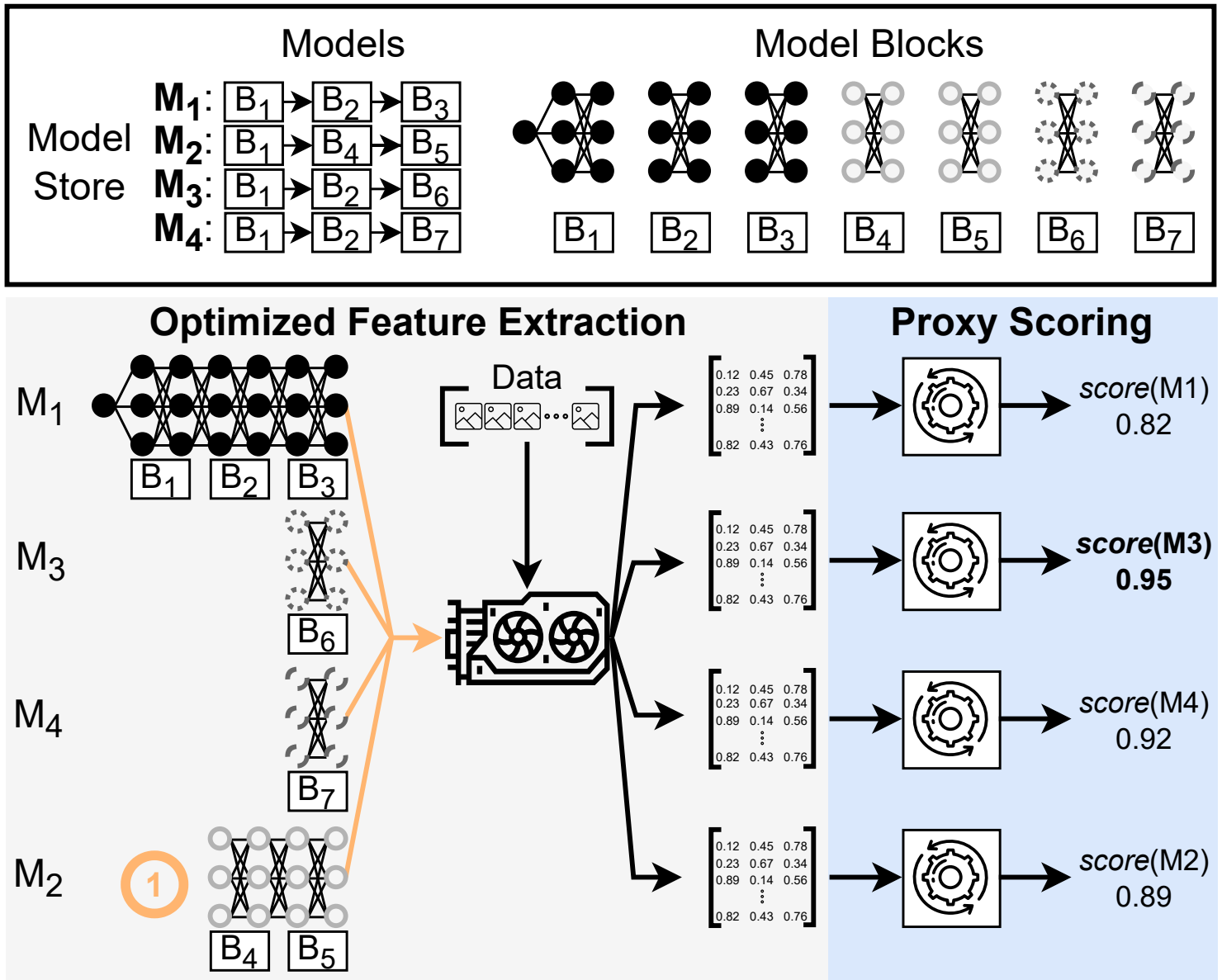
Alsatian – Optimizations



Approach – Optimizations

(1) Partial model access

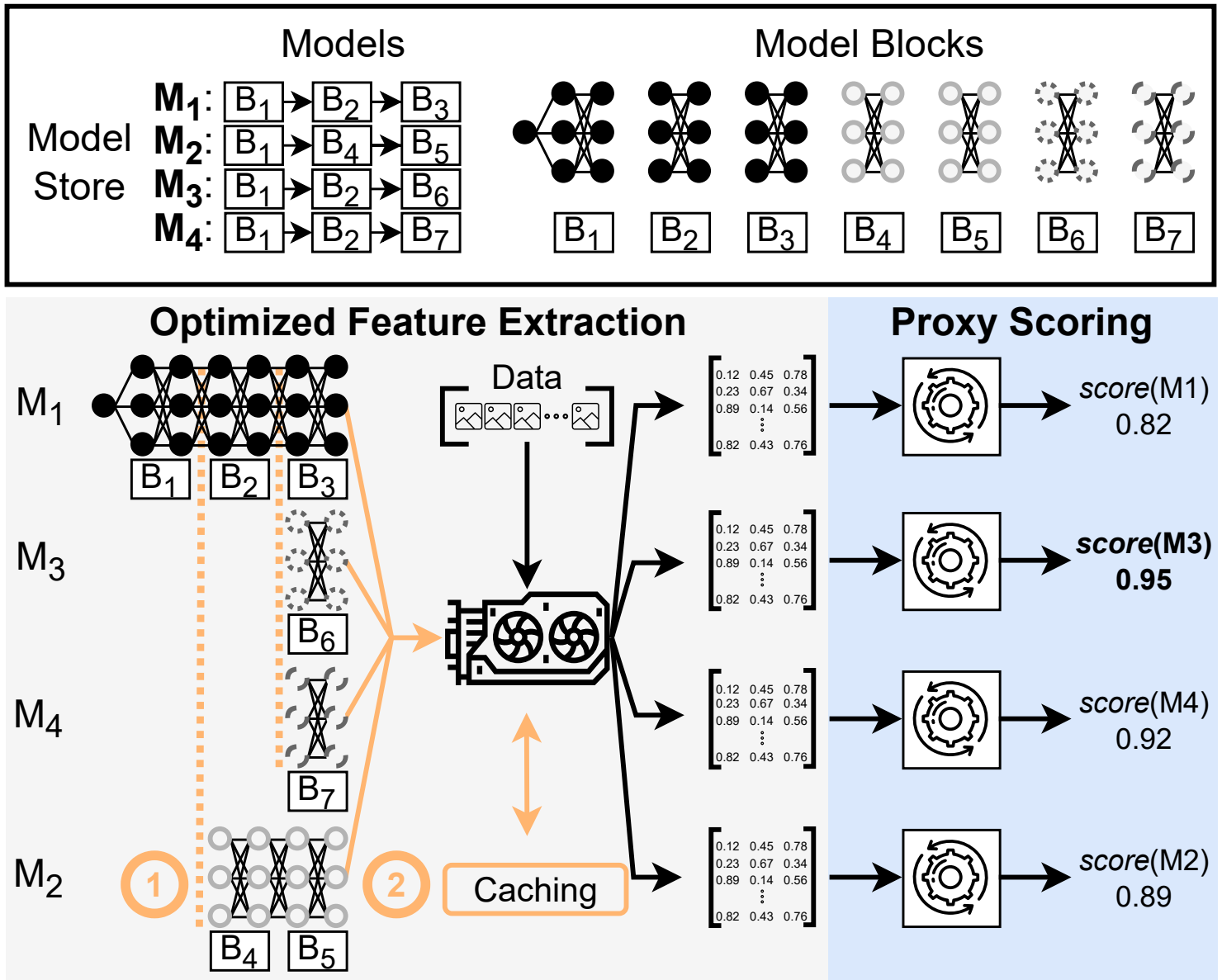
- shorten model preparation time



Approach – Optimizations

(1) Partial model access
- shorten model preparation time

(2) Caching
- avoid redundant inference
  speeds up model preparation

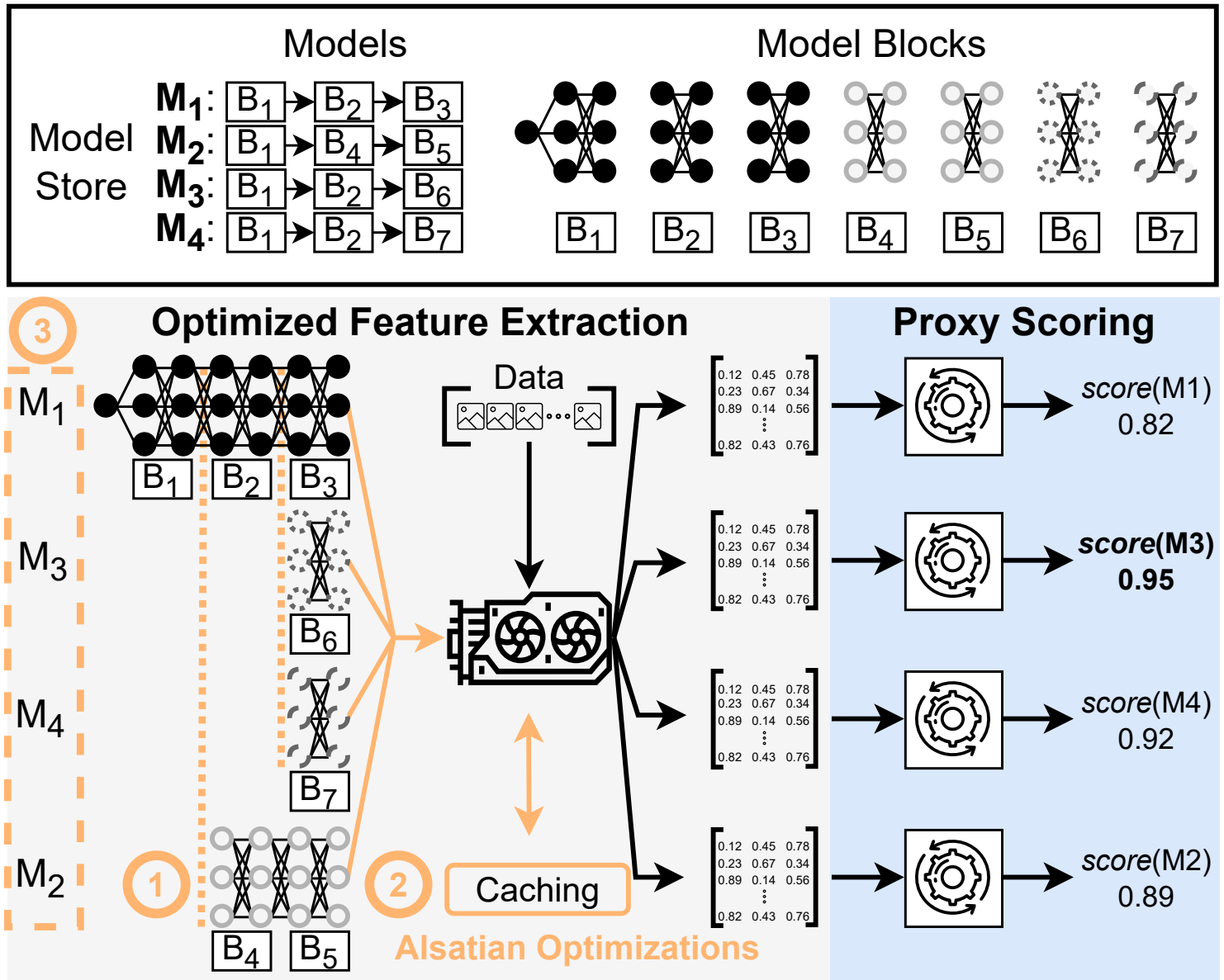


Approach – Optimizations

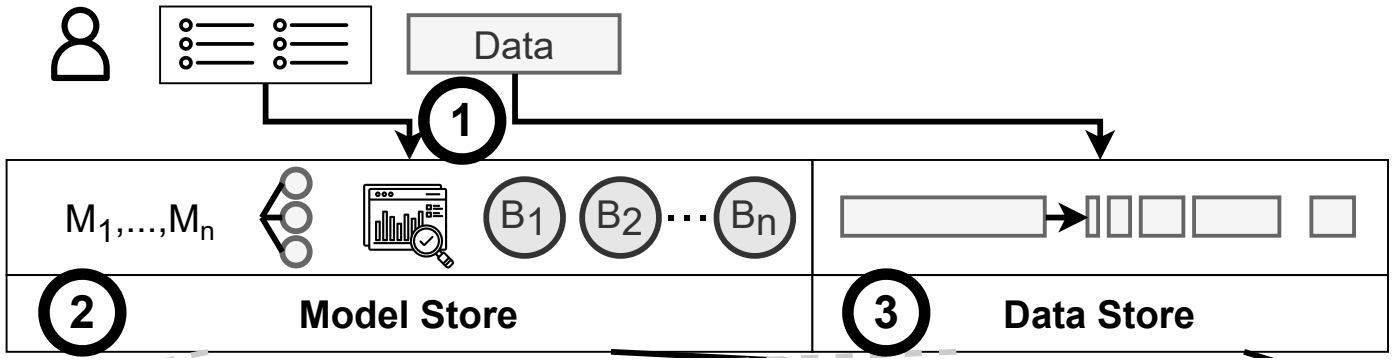
(1) Partial model access
- shorten model preparation time

(2) Caching
- avoid redundant inference
  speeds up model preparation

(3) Execution planning
- maximizes positive effects
  of caching

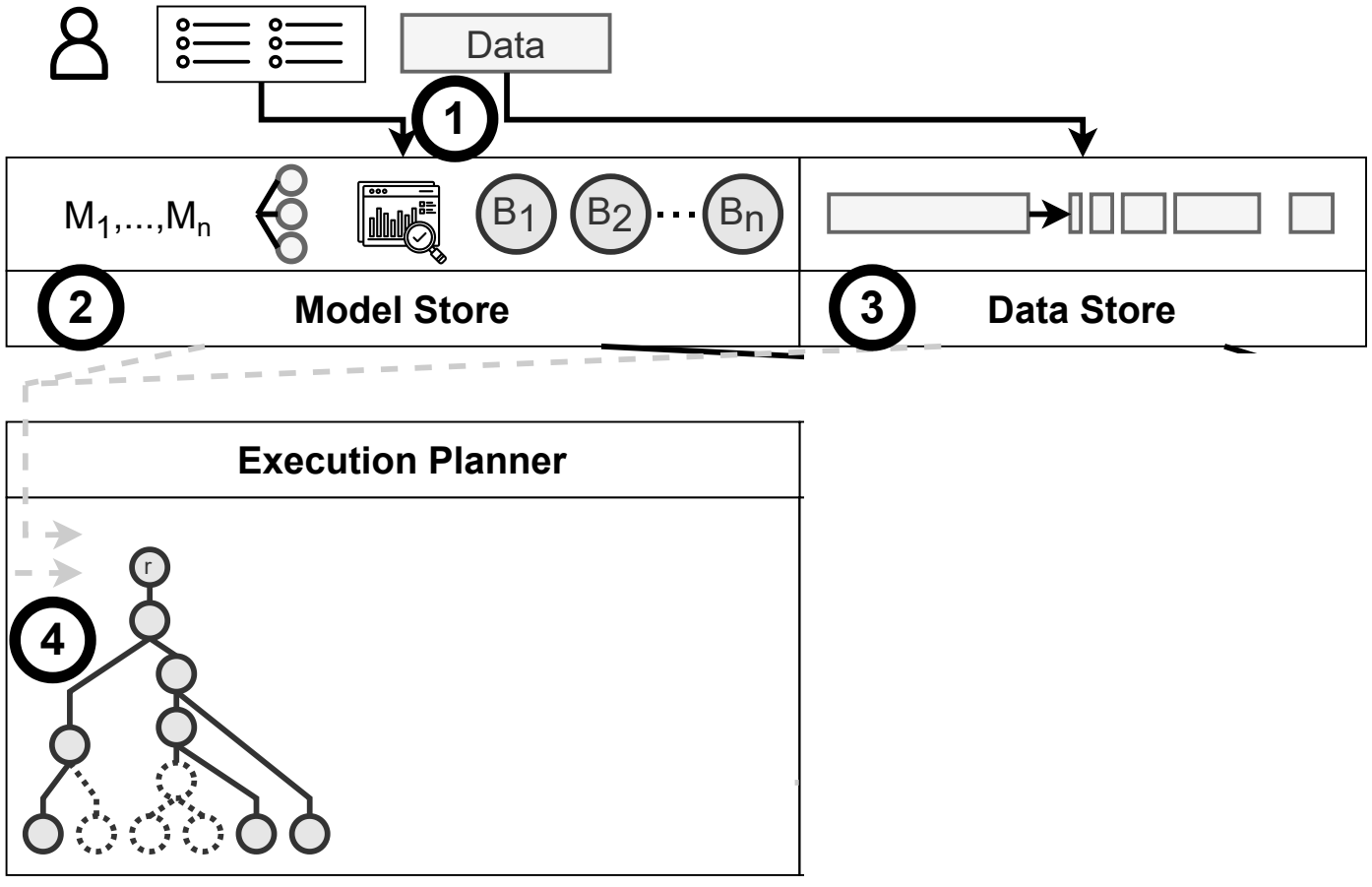


Approach – Alsatian



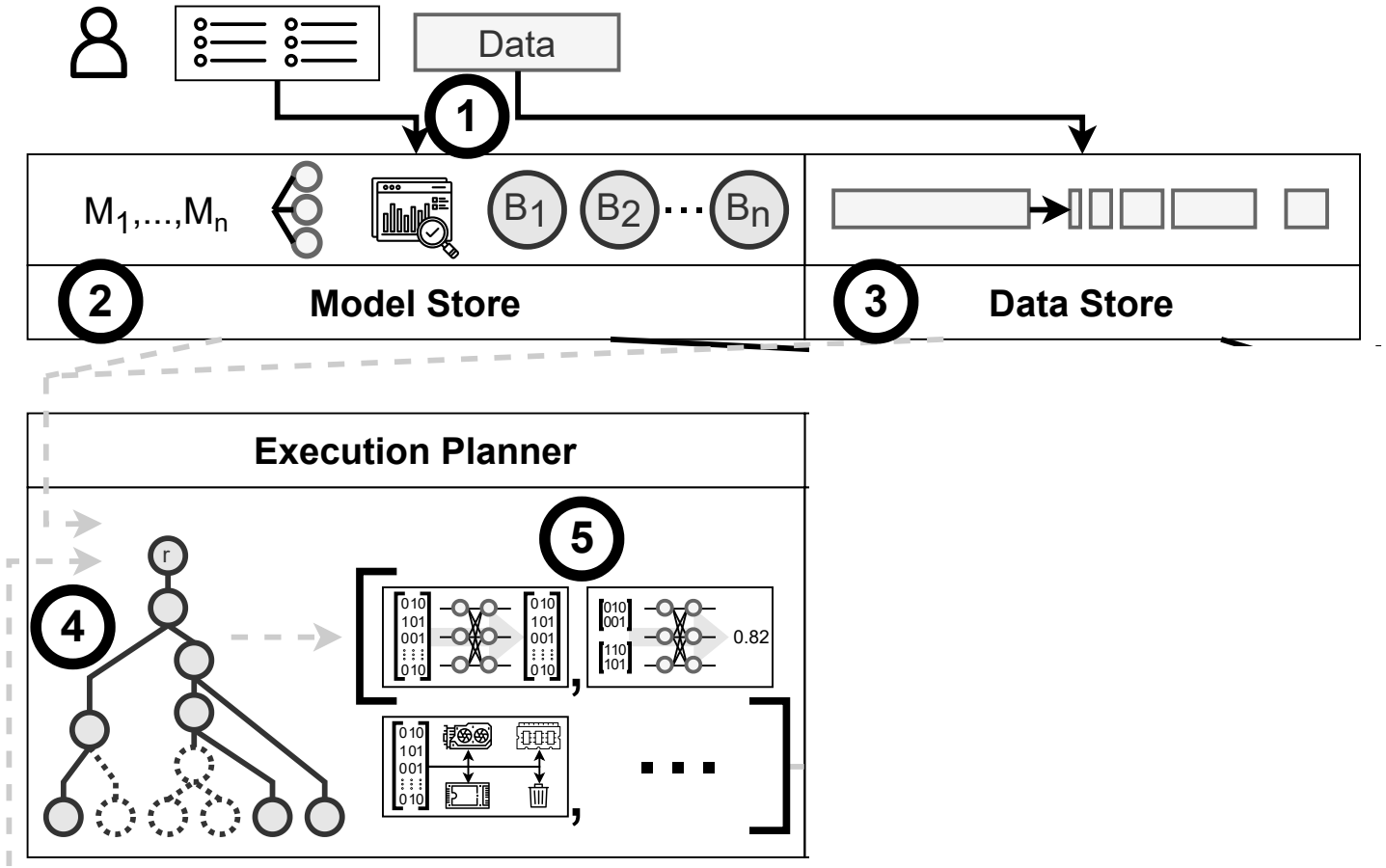
1) Query: dataset & model constraints
2) Identify matching models
3) Split data for successive halving

Approach – Alsatian



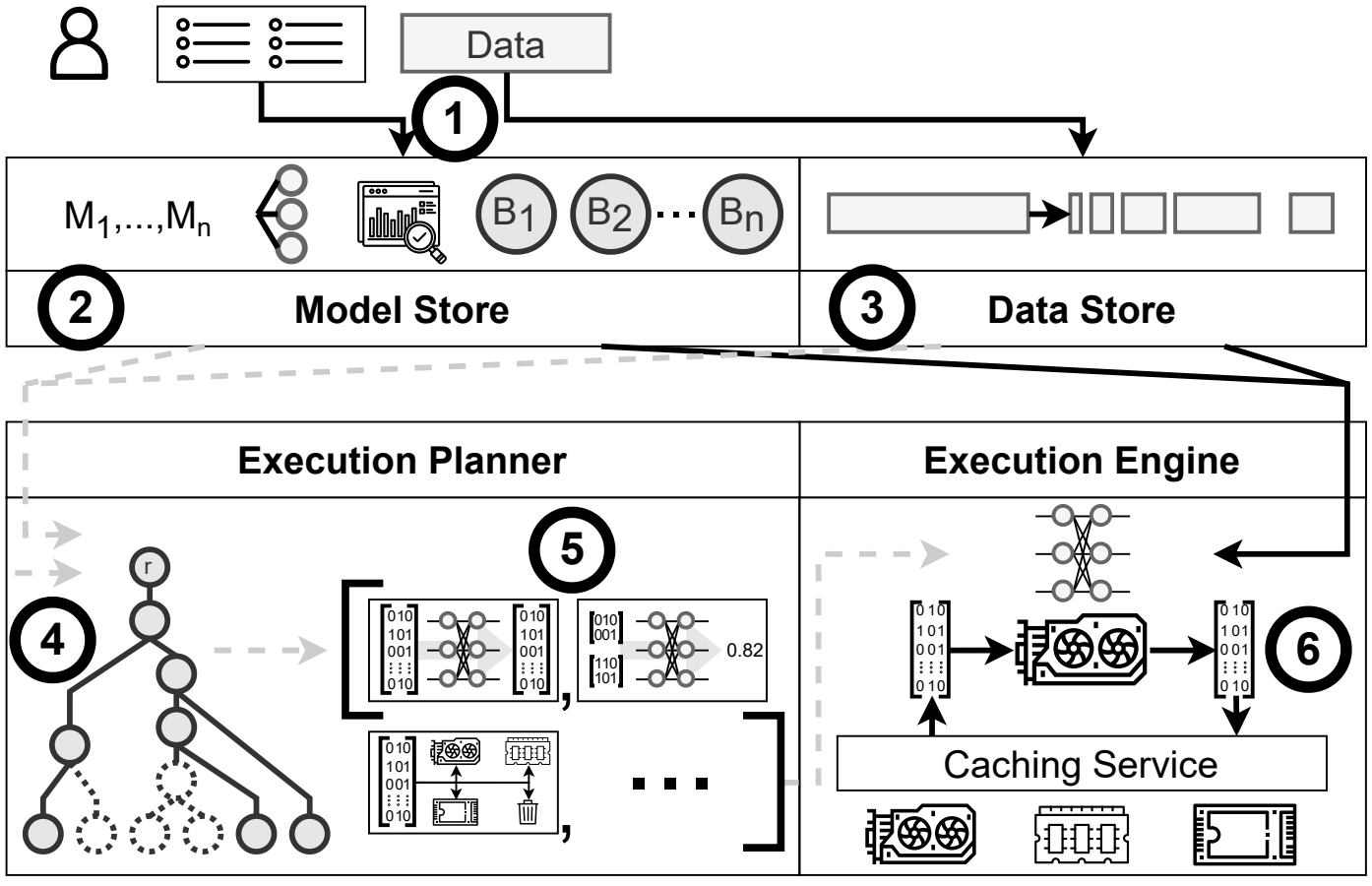
1) Query: dataset & model constraints
2) Identify matching models
3) Split data for successive halving
4) Generate a task tree

Approach – Alsatian



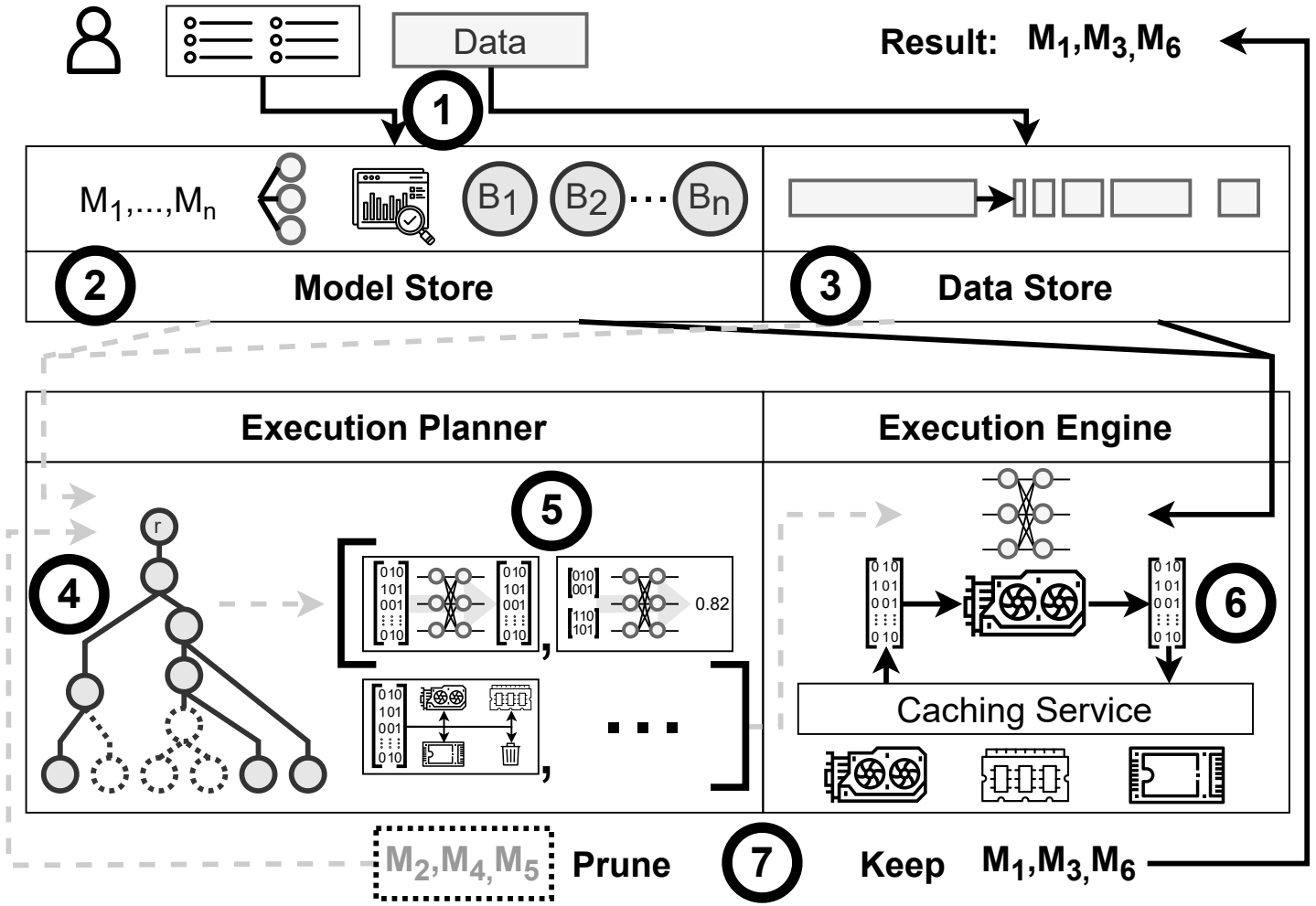
1) Query: dataset & model constraints
2) Identify matching models
3) Split data for successive halving
4) Generate a task tree
5) Traverse task tree to generate an execution plan (list of steps):
   - Inference
   - Proxy Scoring
   - Cache operation

Approach – Alsatian



1) Query: dataset & model constraints
2) Identify matching models
3) Split data for successive halving
4) Generate a task tree
5) Traverse task tree to generate an execution plan (list of steps):
   - Inference
   - Proxy Scoring
   - Cache operation
6) Execute execution plan

Approach – Alsatian



1) Query: dataset & model constraints
2) Identify matching models
3) Split data for successive halving
4) Generate a task tree
5) Traverse task tree to generate an execution plan (list of steps):
   - Inference
   - Proxy Scoring
   - Cache operation
6) Execute execution plan
7) Prune half of the models → repeat

Evaluation – Search Time

Search over 35 models, find the best model

Evaluation – Search Time

Search over 35 models, find the best model

Three dimensions: **model similarity**

Evaluation – Search Time

Search over 35 models, find the best model

Three dimensions: model similarity, **dataset size**

Evaluation – Search Time

Search over 35 models, find the best model

Three dimensions: model similarity, dataset size, **model architecture**

Evaluation – Search Time

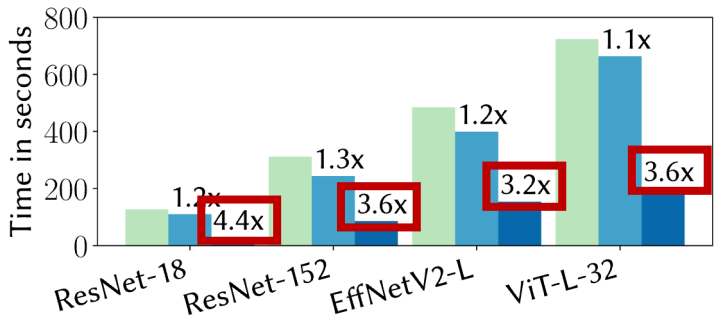
Search over 35 models, find the best model

Three dimensions: model similarity, dataset size, model architecture

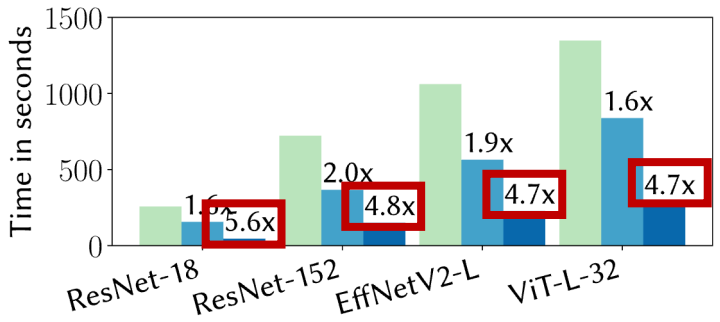
Base SHiFT Alsatian

~50% retrained **~25% retrained** **top few retrained**

2000 items



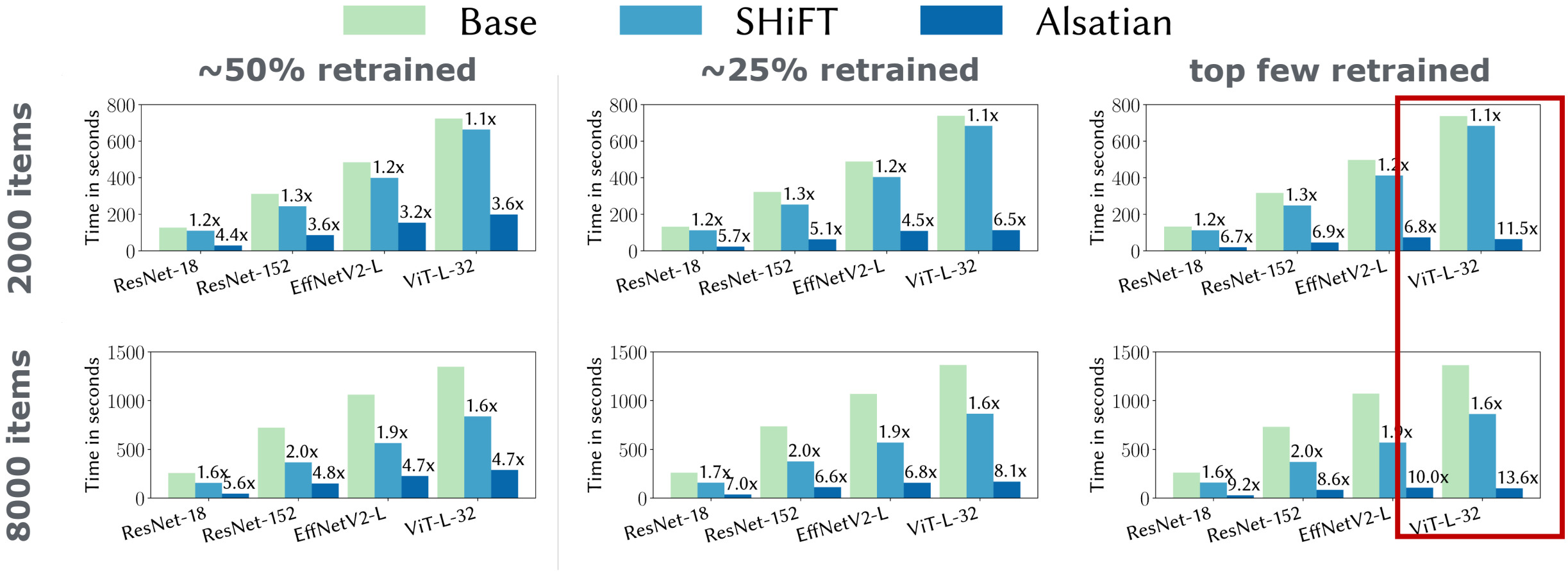
8000 items



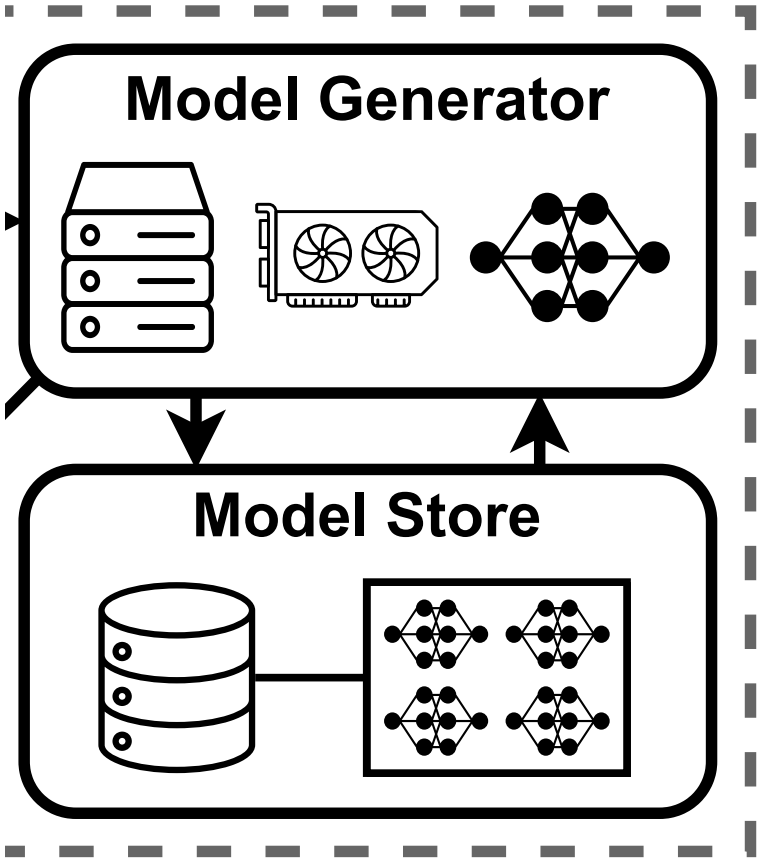
Evaluation – Search Time

Search over 35 models, find the best model

Three dimensions: model similarity, dataset size, model architecture



Co-Design Model Search and Model Store

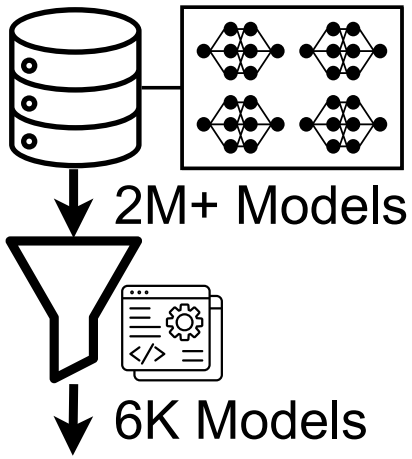


Two main directions

(1) Co-design model store and model search techniques

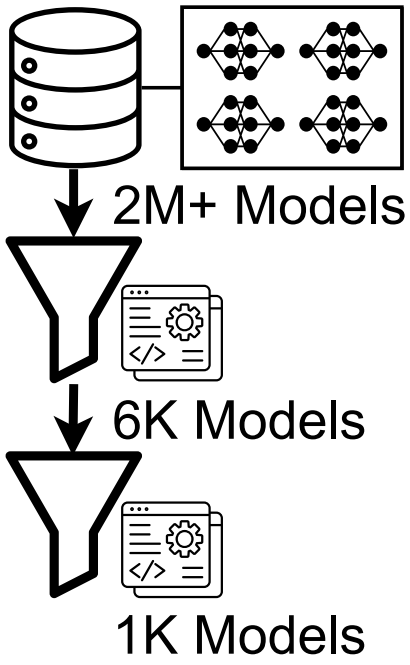
(2) Approximate model search

Model Search



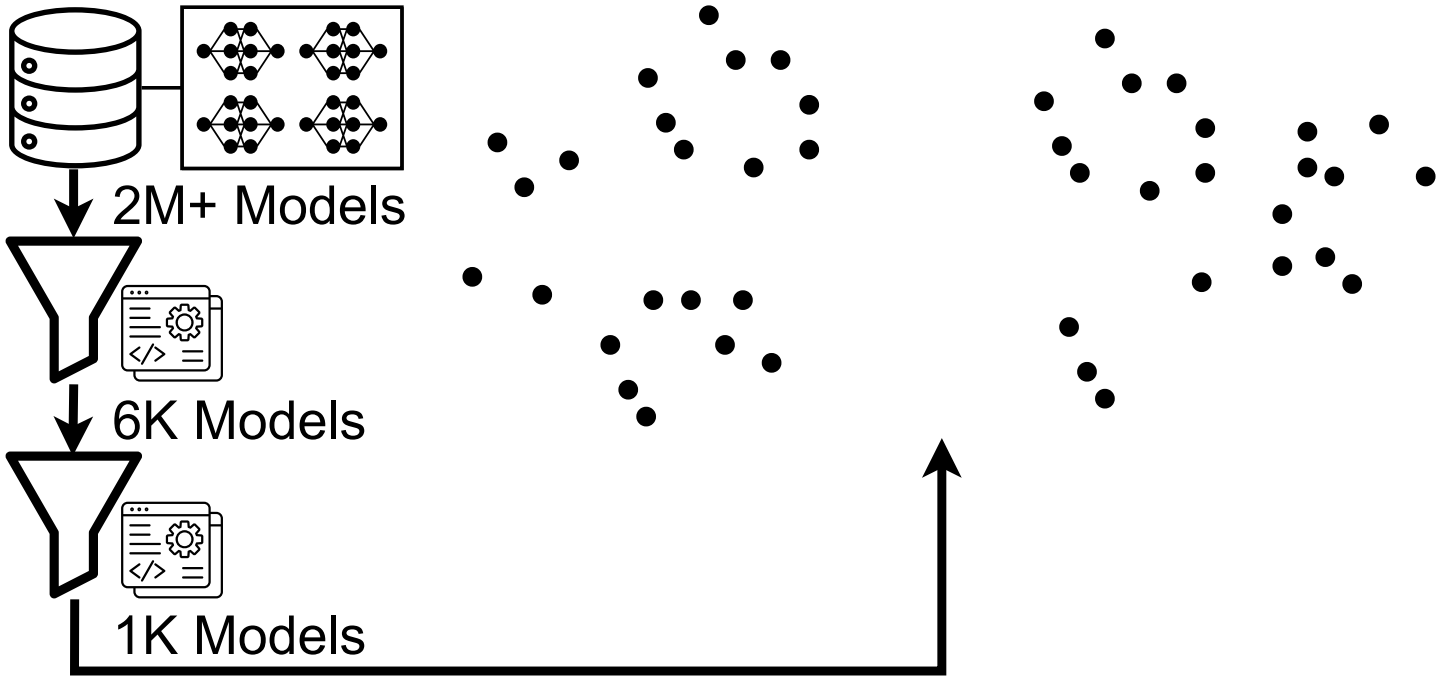
1) Filter by architecture

Model Search



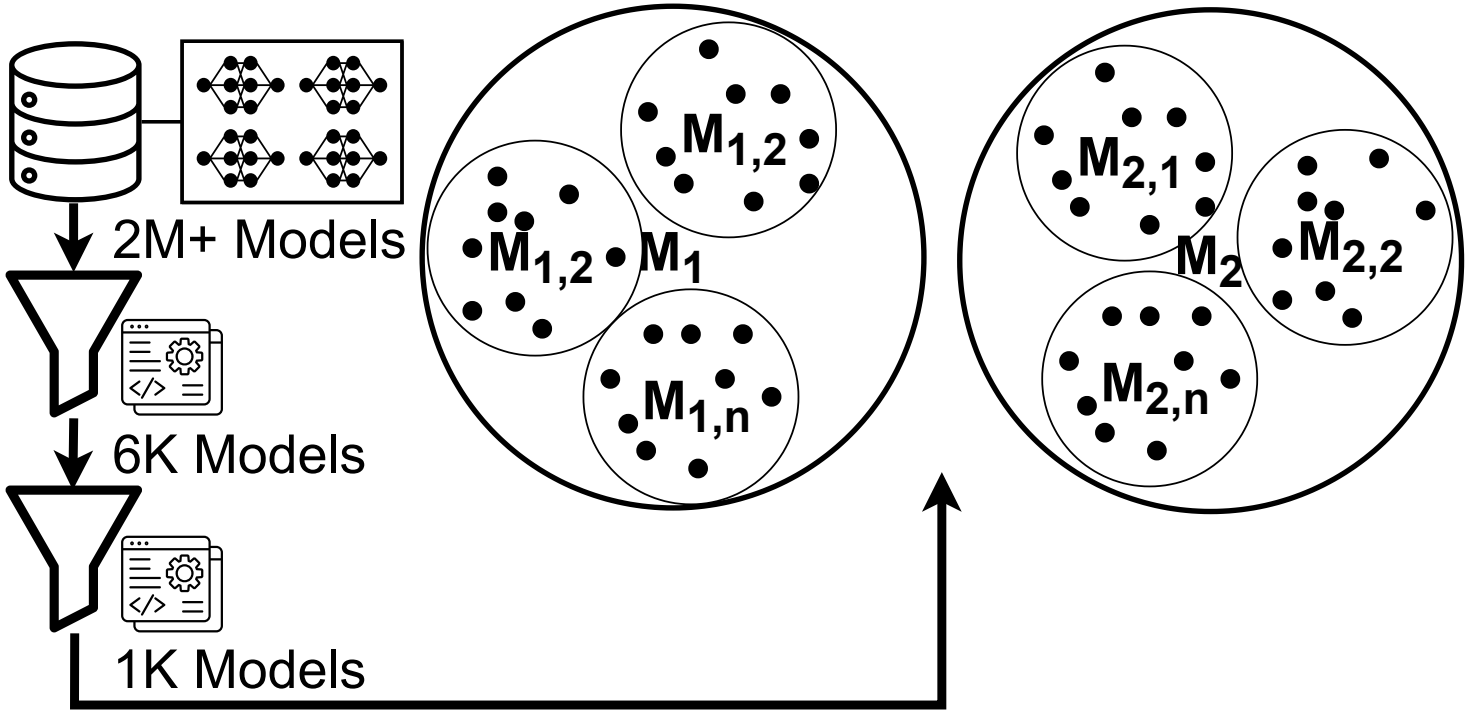
1) Filter by architecture
2) Filter by performance

Model Search



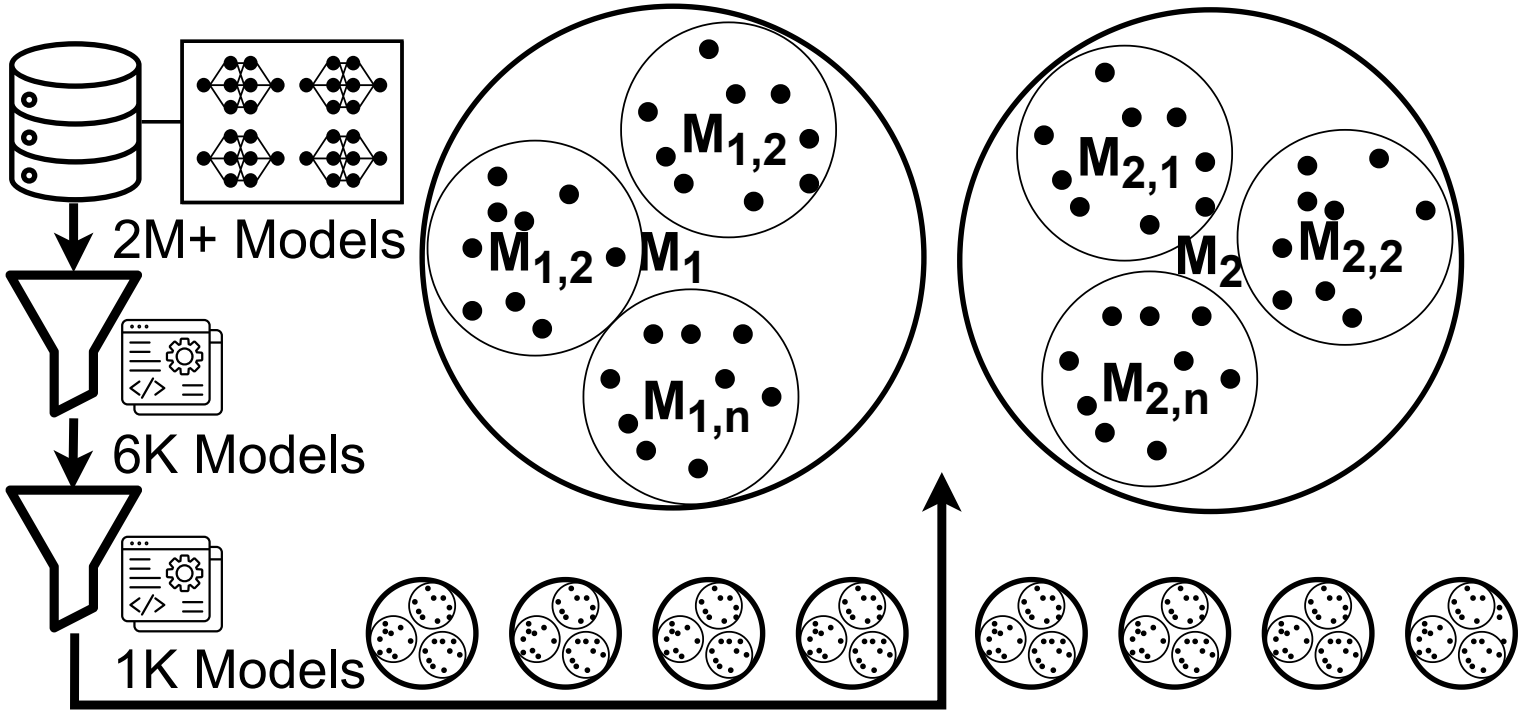
1) Filter by architecture
2) Filter by performance
3) Embed models

Model Search



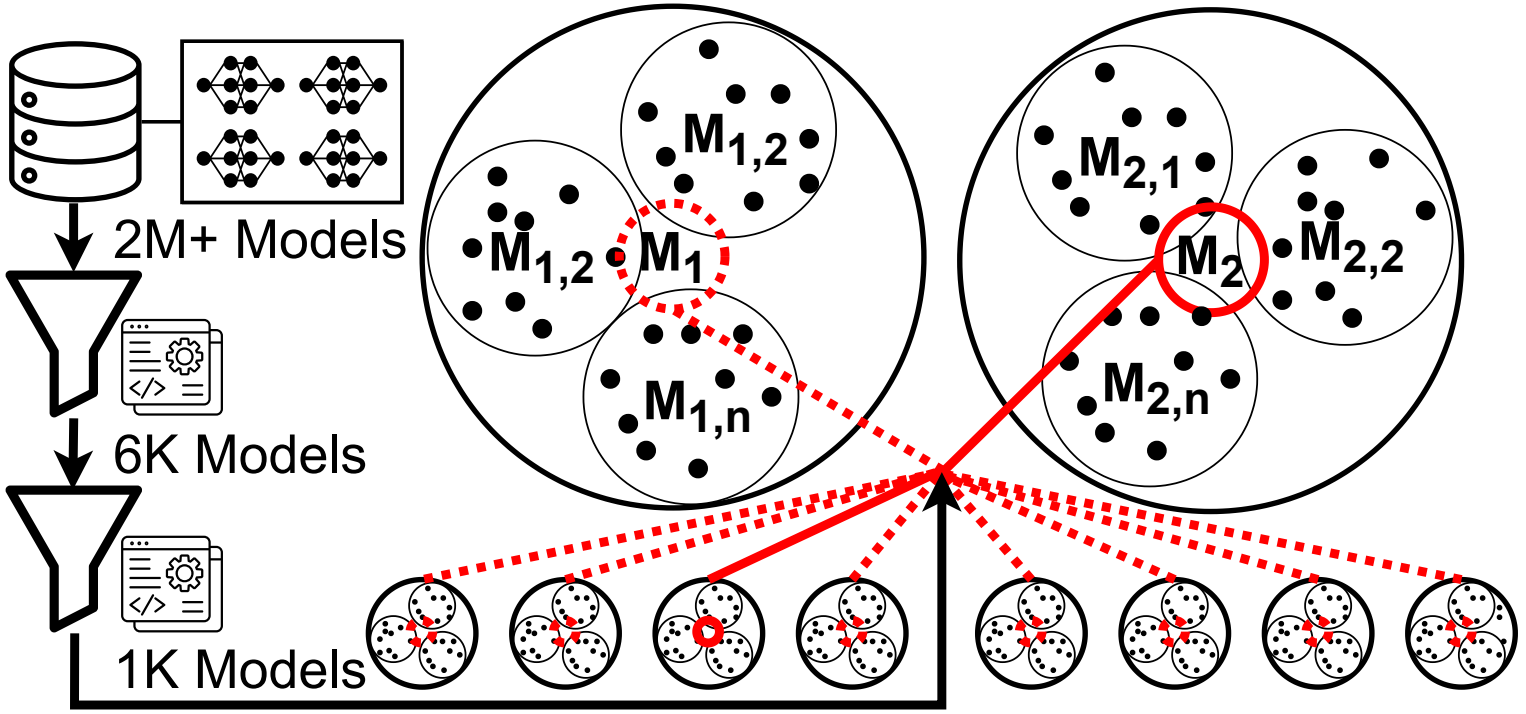
1) Filter by architecture
2) Filter by performance
3) Embed models
4) Hierarchical clustering

Model Search



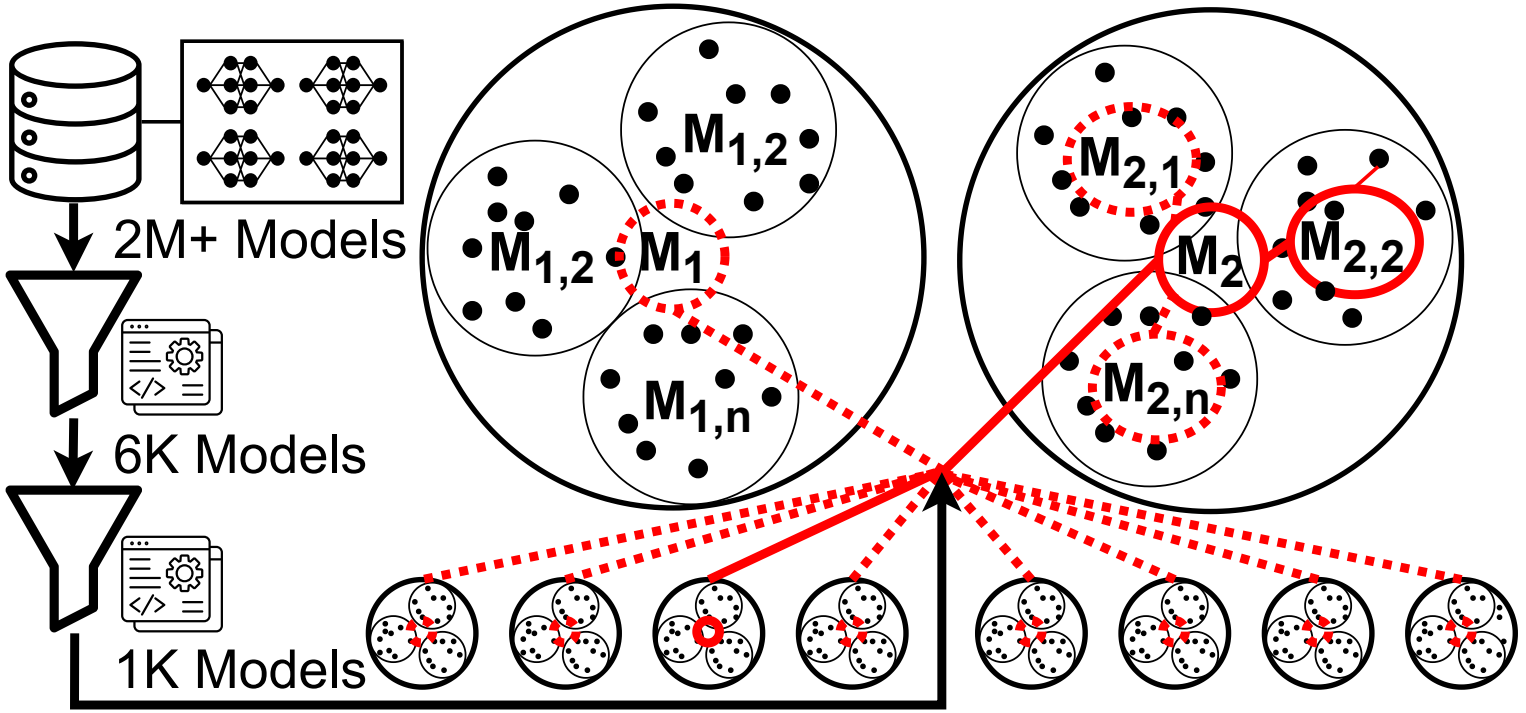
1) Filter by architecture
2) Filter by performance
3) Embed models
4) Hierarchical clustering

Model Search



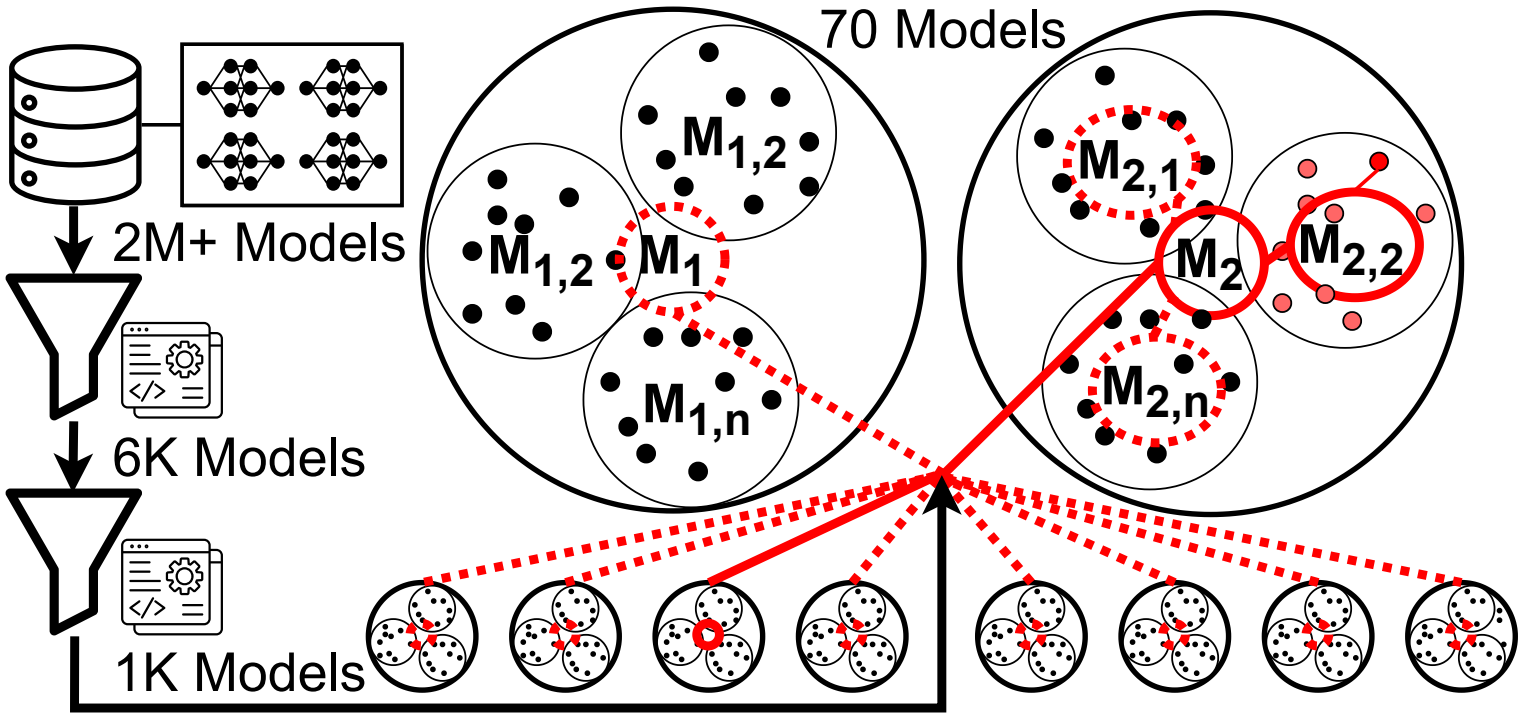
1) Filter by architecture
2) Filter by performance
3) Embed models
4) Hierarchical clustering
5) Search approx. cluster centroids

Model Search



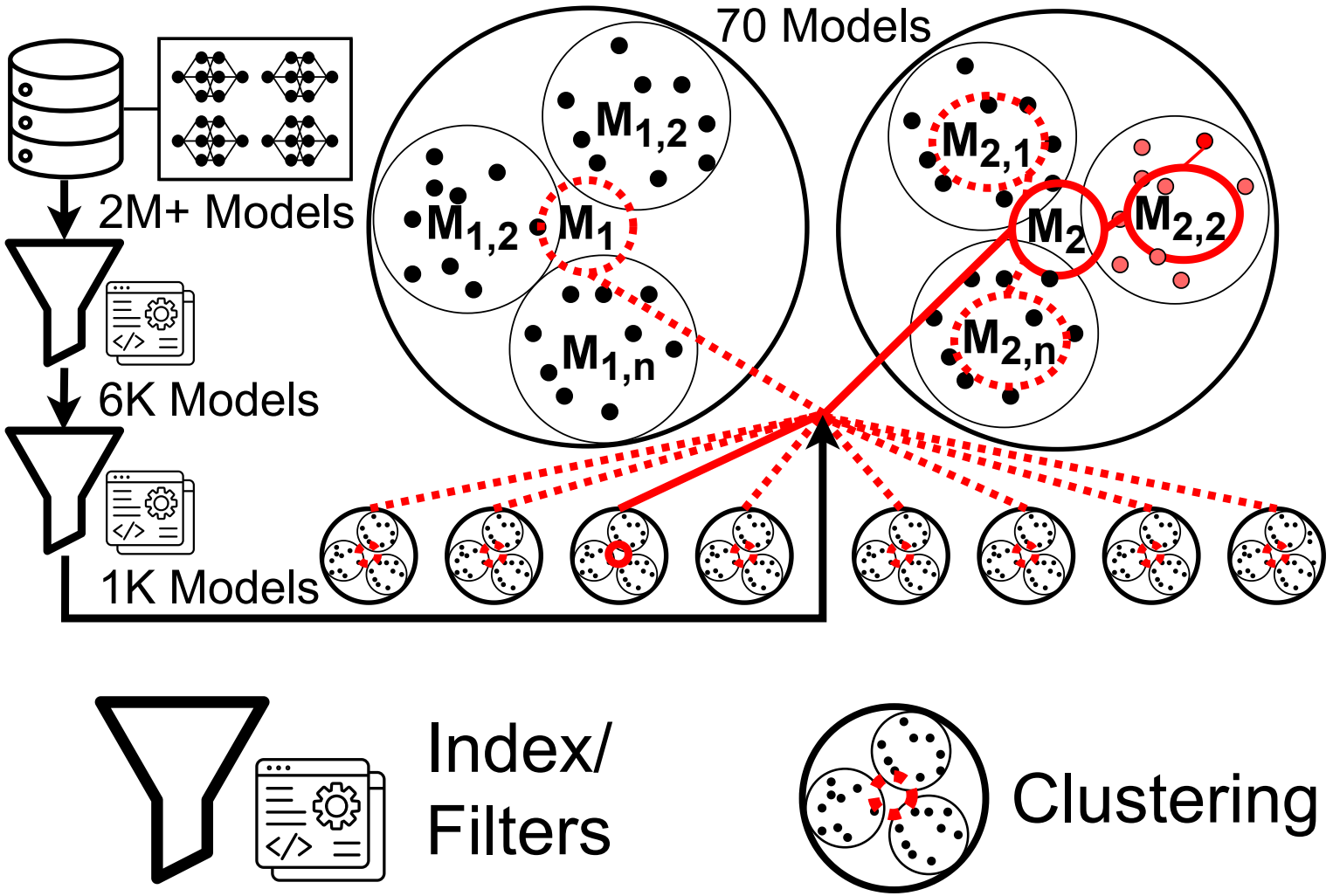
1) Filter by architecture
2) Filter by performance
3) Embed models
4) Hierarchical clustering
5) Search approx. cluster centroids
6) Search approx. within cluster

Model Search



1) Filter by architecture
2) Filter by performance
3) Embed models
4) Hierarchical clustering
5) Search approx. cluster centroids
6) Search approx. within cluster
7) Search approx. within cluster

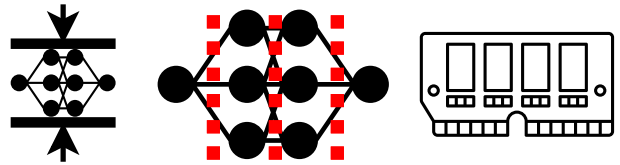
Model Search



Index/
Filters

Clustering

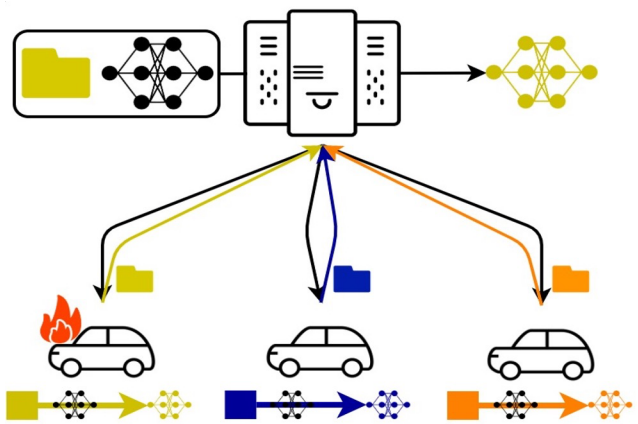
1) Filter by architecture
2) Filter by performance
3) Embed models
4) Hierarchical clustering
5) Search approx. cluster centroids
6) Search approx. within cluster
7) Search approx. within cluster



Compression,
Partial Access,
Caching

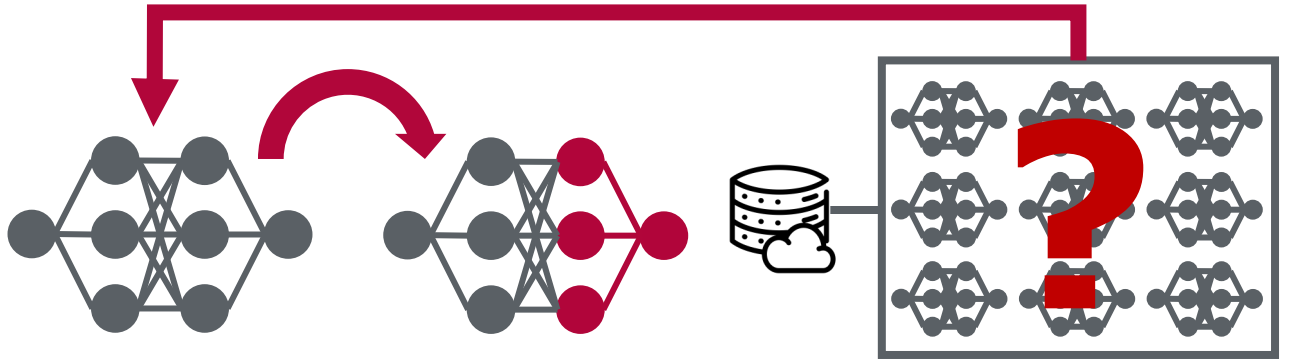
Model Management

Model Management for Archival



- Efficiently manage many DL models in a distributed environment
- Assumption: save often, read rarely
- MMlib and M3lib

Model Management for Reuse



- Efficiently access models to find the best candidate for transfer learning
- Assumption: read often, models overlap
- Poodle and Alsatian