Portable Fast Platform-Aware Neural Architecture Search for Edge/Mobile Computing AI Applications

Presenter: Kuo-Teng Ding (NCHC, Taiwan, ROC)

Authors: Kuo-Teng Ding, Hui-Shan Chen, Yi-Lun Pan, Hung-Hsin Chen, Yuan-Ching Lin and Shih-Hao Hung

Outline

- Background & Motivation
- Methodology
  - Just-In-Time (JIT) Performance Prediction Module
  - Once-for-All Pre-training Module
  - Neural Architecture Search (NAS) Module
  - Neural Architecture Search (NAS) Agent Module
  - Kernel Service Module
- System Design
- Experiments
- Conclusion

Motivation

Goal

- Split the model training process, let the data center do the steps that require time and resources, and let the user terminal do the platform testing that does not require time.

- Protect end users' private datasets.

- Leverage multiple hyperparameter tunning methods to recommend hyperparameters and speed up the whole process.

- The proposed framework can be applied to various public cloud services.

Contribution

- End-users can use the service to train their own neural networks without leaking their own datasets.

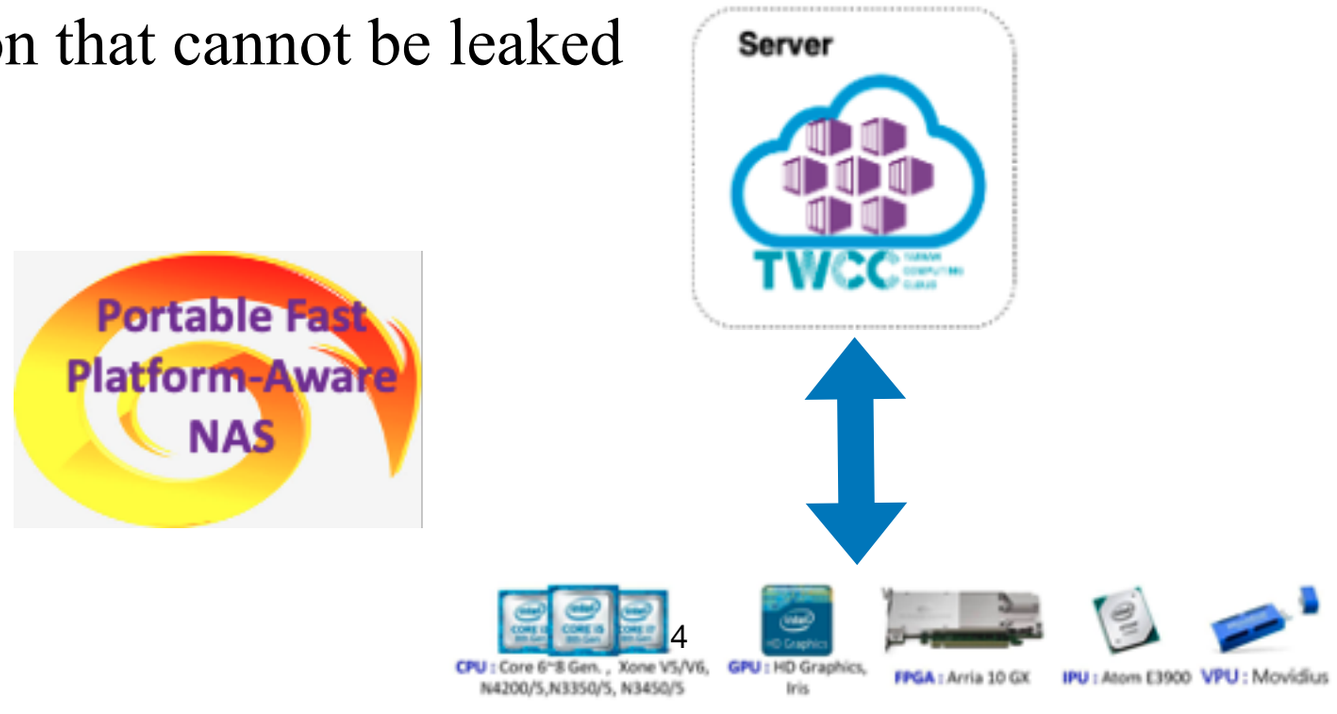
- Multiple hyperparameter tunning methods can be chosen.

- Intuitive and convenient user interface, providing users with easier use.

Motivation

Problems

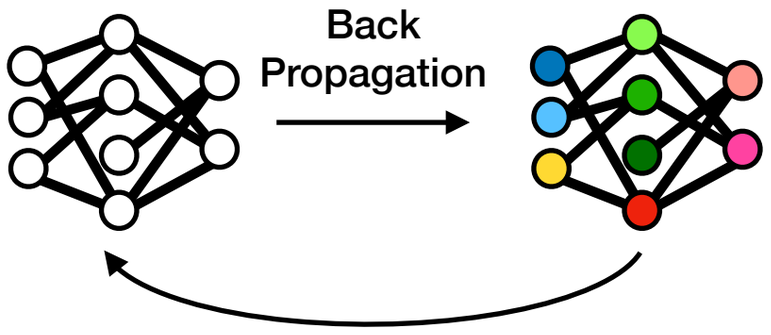
- Users' pain points - **Edge AI System Developers**
    - Pain point I: NAS and training neural networks require a lot of computing power
    - Pain point II: All kinds of AI accelerators are very different, and traditional search methods (NAS) are difficult to cover
    - Pain point III: The data set and accelerator structure are confidential information that cannot be leaked



Motivation

Current State

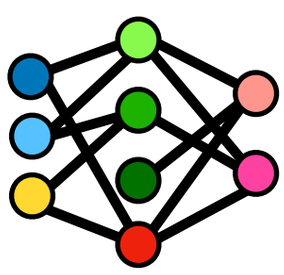
- "How to offload the computing resource from DC to end user along with privacy preserving?"



= Time- and Resource-Consuming Training

Back Propagations For Many Times

- A deep learning network training process



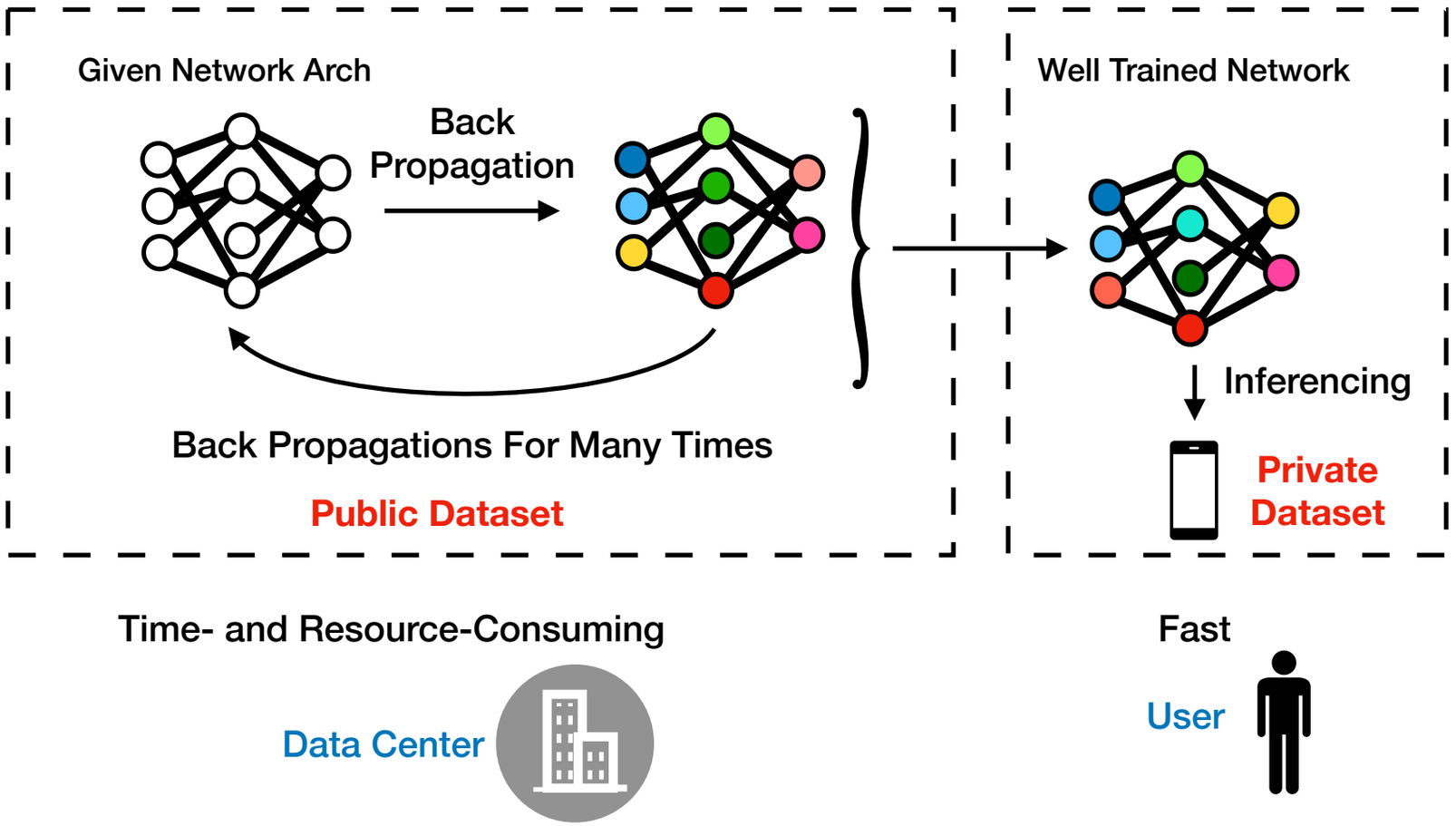
= Fast

Motivation

The Proposed Framework

- Time- and resource-consuming tasks remain to DC

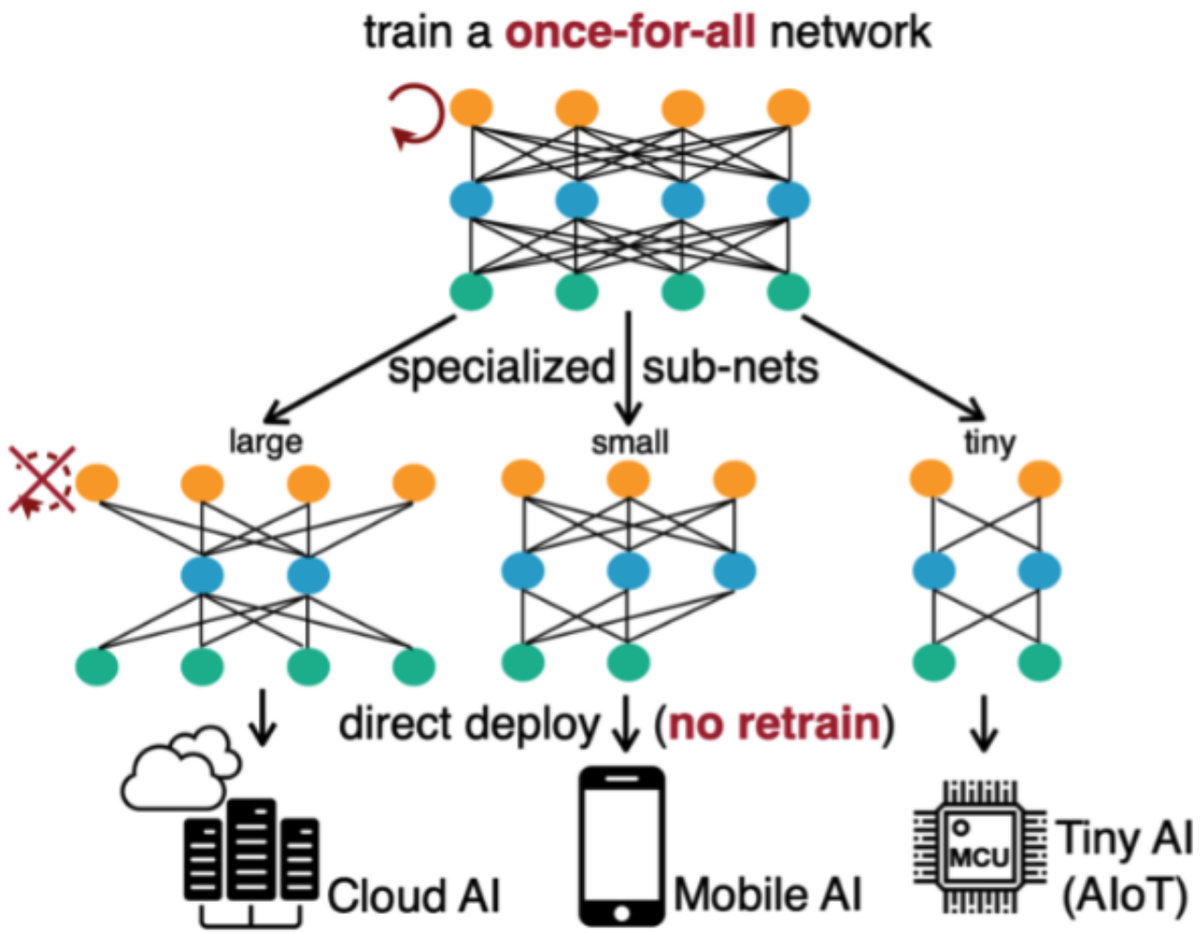
- End users provide their devices to execute (inference) network with their own dataset



Related Works

Once-for-all Network

- Once-For-All network (ICLR'2020)



Background

Inspiration

- Q: How to deal with the difference between private and public dataset?

- A: Backpropagation according to performance prediction (on private dataset)

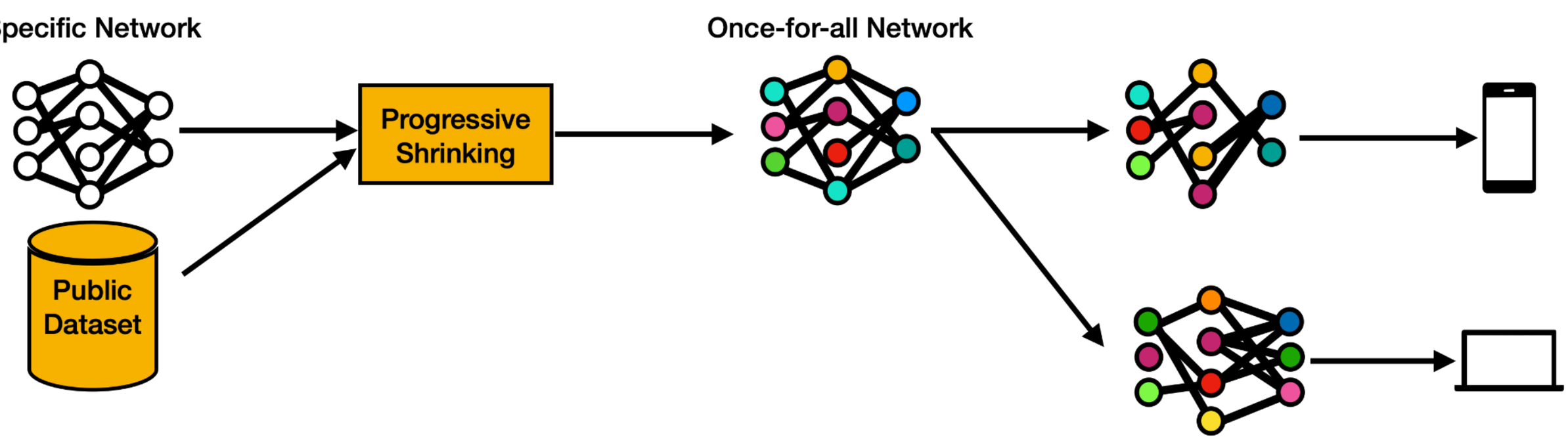
- Q: How to decrease prediction times efficiently?

- A: Hyperparameter tuning

Methodology

Once-for-All Pre-training

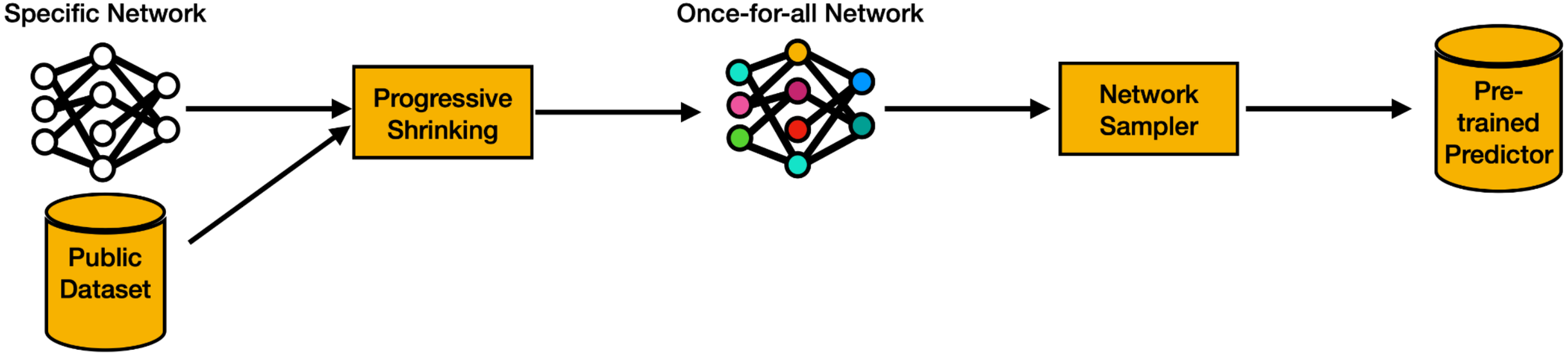
- A task-aware OFA network will be pre-trained.



Methodology

Once-for-All Pre-training

- A task-aware OFA network will be pre-trained.



Methodology

Performance Prediction Module

- 3 layers MLP predictors for latency and accuracy

- Joint score algorithm

Algorithm 1 The designed joint latency and accuracy score

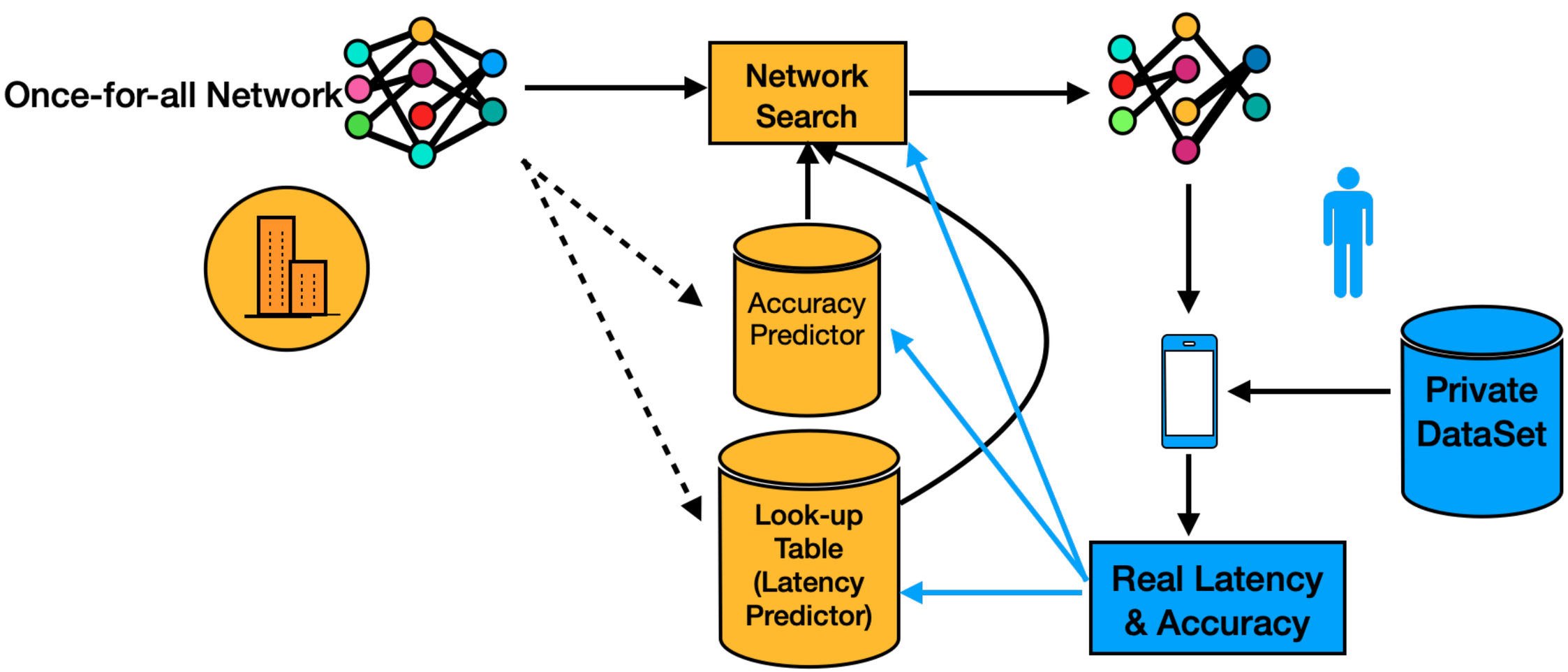
Input: latency and accuracy
Output: score

```
if latency > LATENCY_THRES then
    score ← accuracy - (latency / LATENCY_THRES ) * PENALTY_COEFF
else
    score ← accuracy
end if
```

Methodology

Once-for-All with Performance Prediction

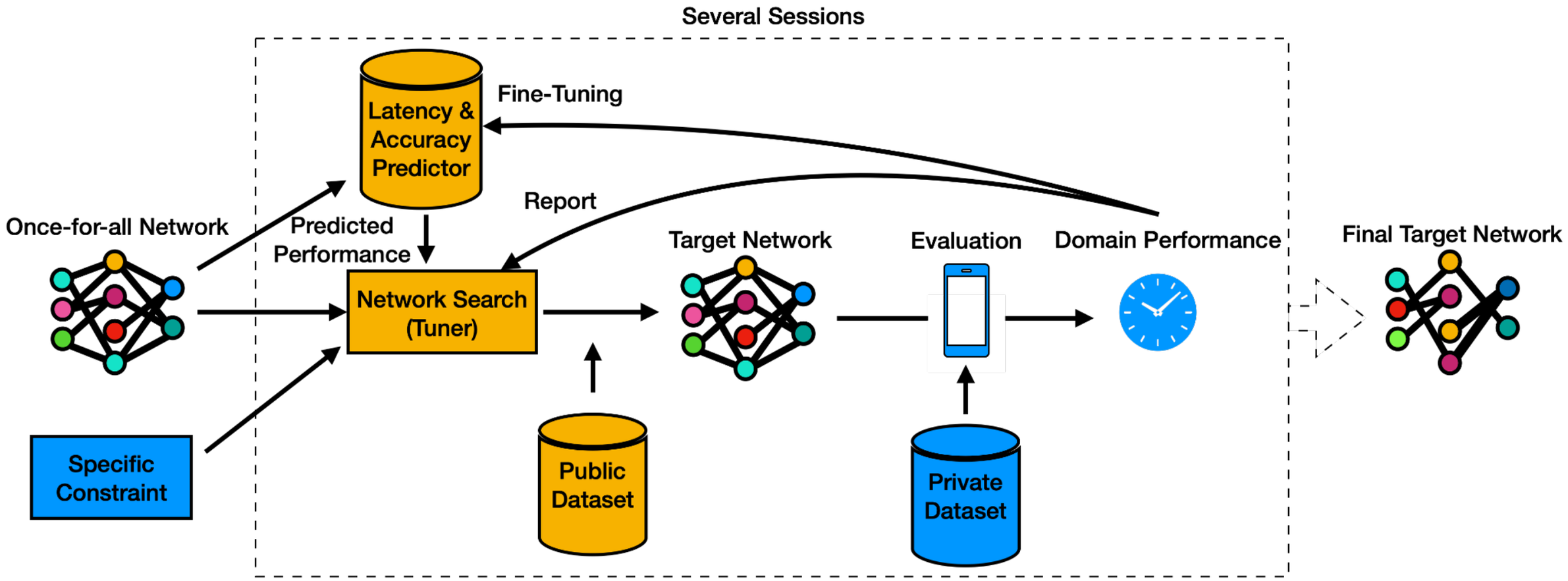
- With pre-trained OFA networks, DC can provide subnet according to inferencing latency and accuracy



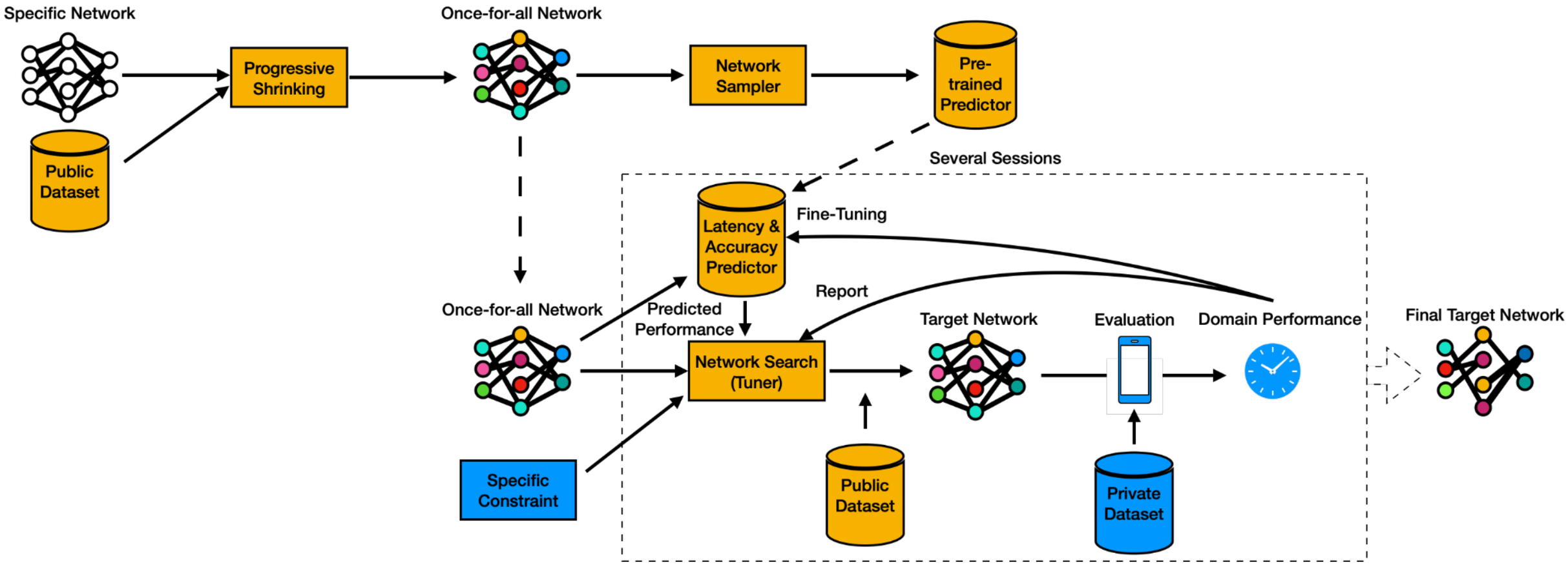
Methodology

Hyper-Parameter Tuning

- For accelerating the NAS process, we will use a hyperparameter tuner to recommend the hyperparameters in each training epoch.



Methodology

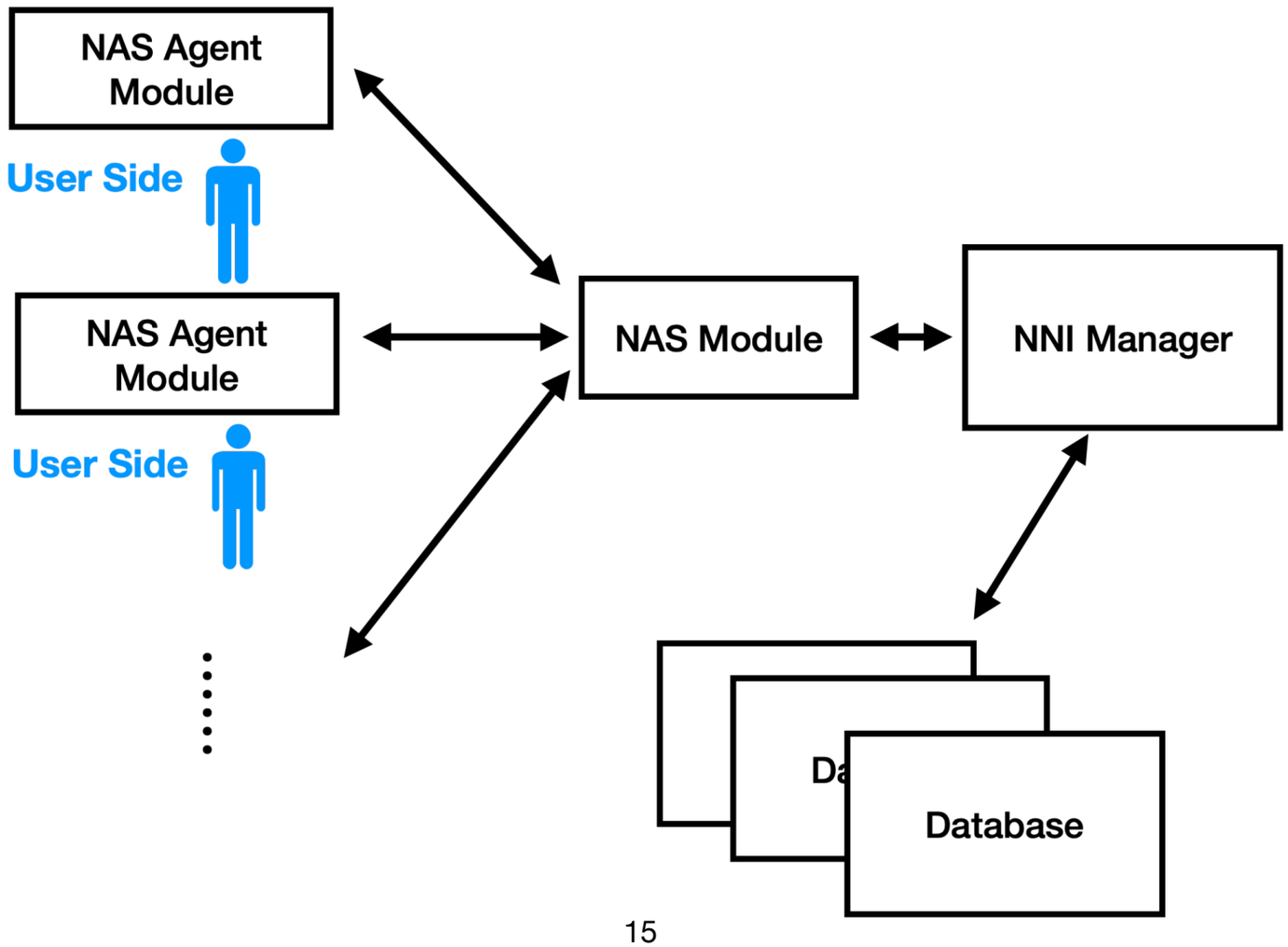


System Design

NAS and NAS Agent Module

- NAS agent is a daemon process deployed on user side, it performs a network inference and gather the hardware metrics.

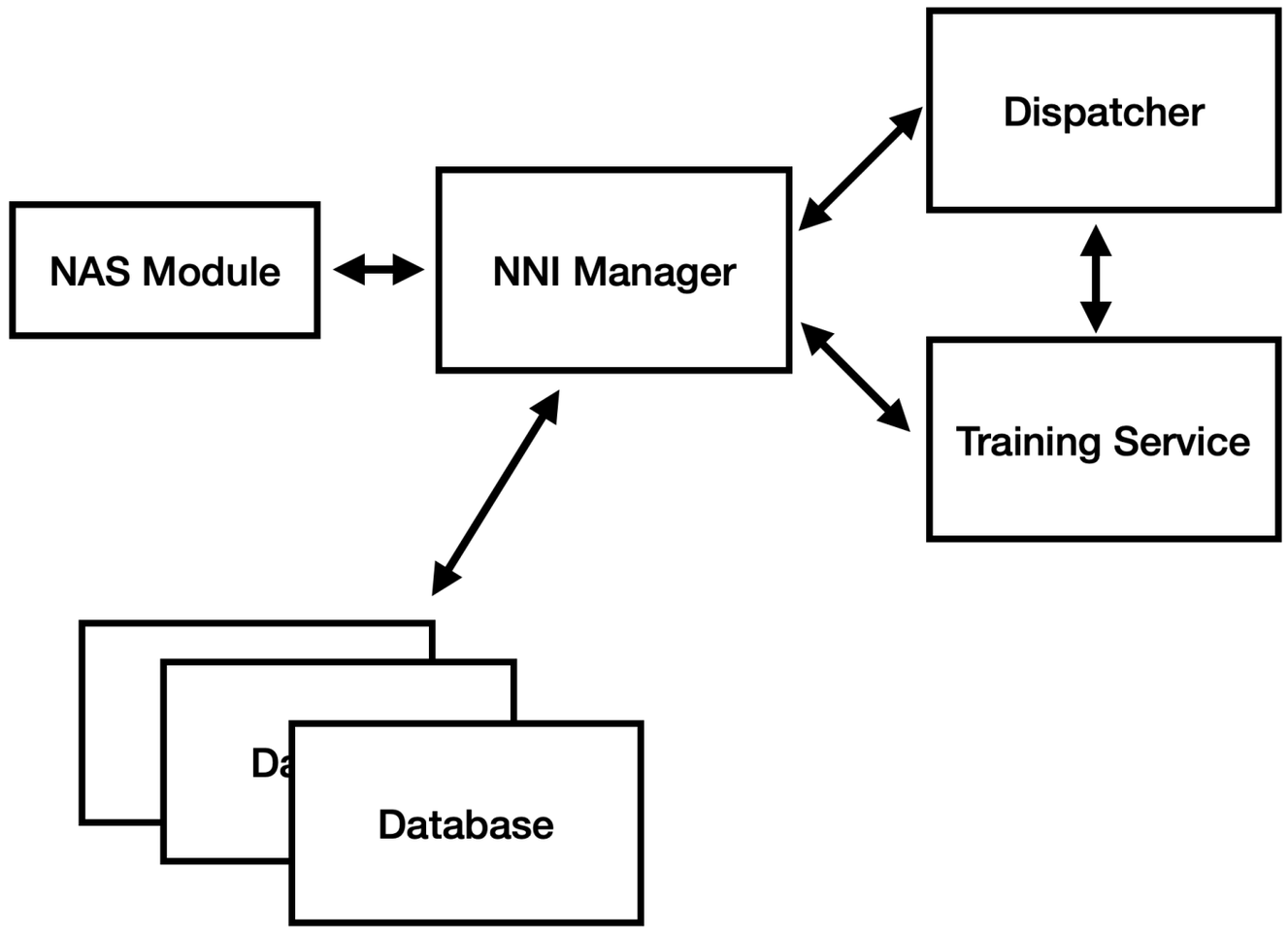
- NAS module communicates with NAS agent.



System Design

Kernel Service Module

- Kernel Service is a modified Microsoft NNI kernel, it's a control interface for Dispatcher (for hyperparameter tuning), Training Service (for OFA) and NAS module (for performance prediction)



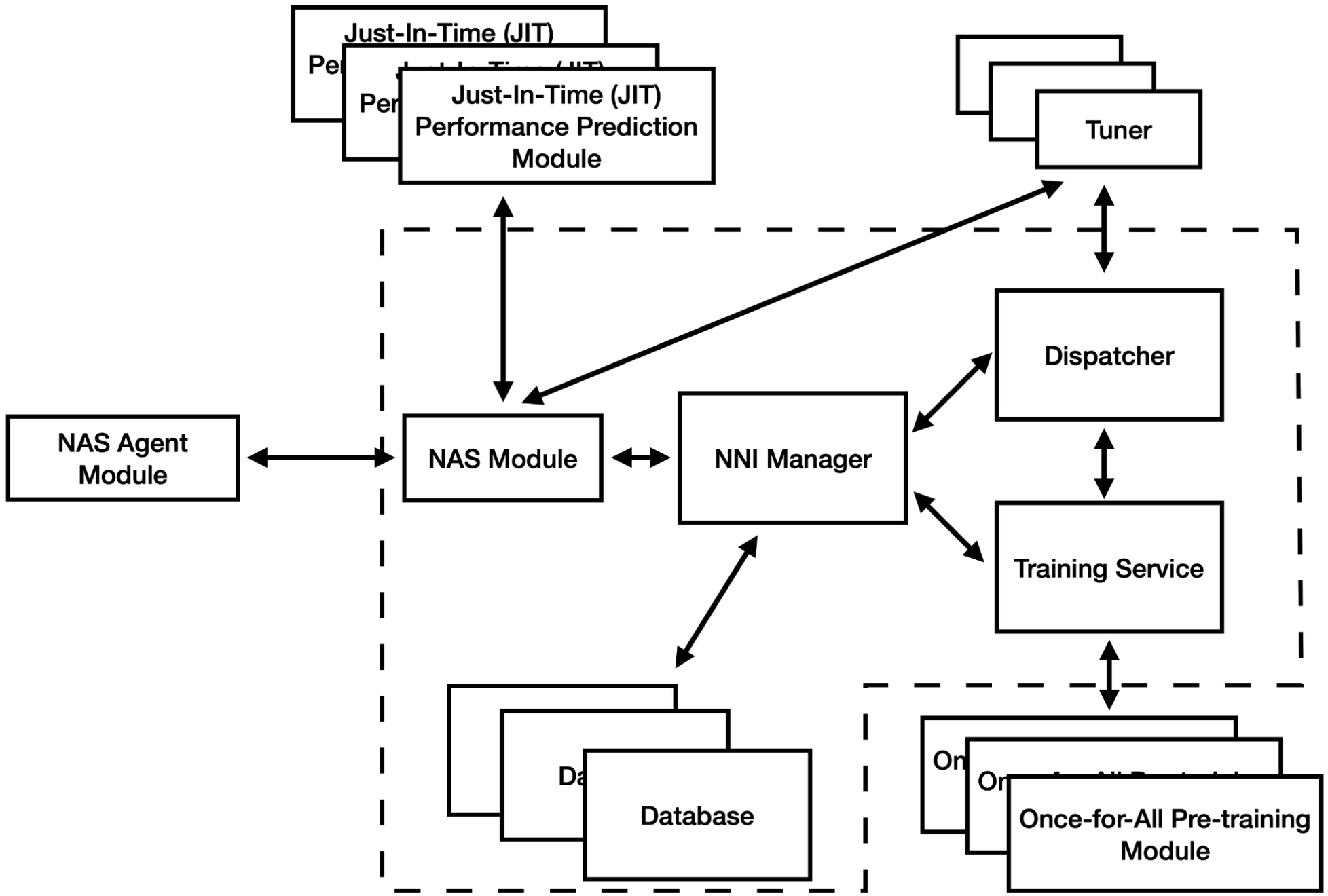
System Design

Components

Task	Control Component	Component
OFA Pretraining	OFA Pretraining Module	Training Service
Performance Predict	JIT PP Module	NAS Agent
Hyperparameter Tuning	Tuner	Dispatcher

System Design

Components

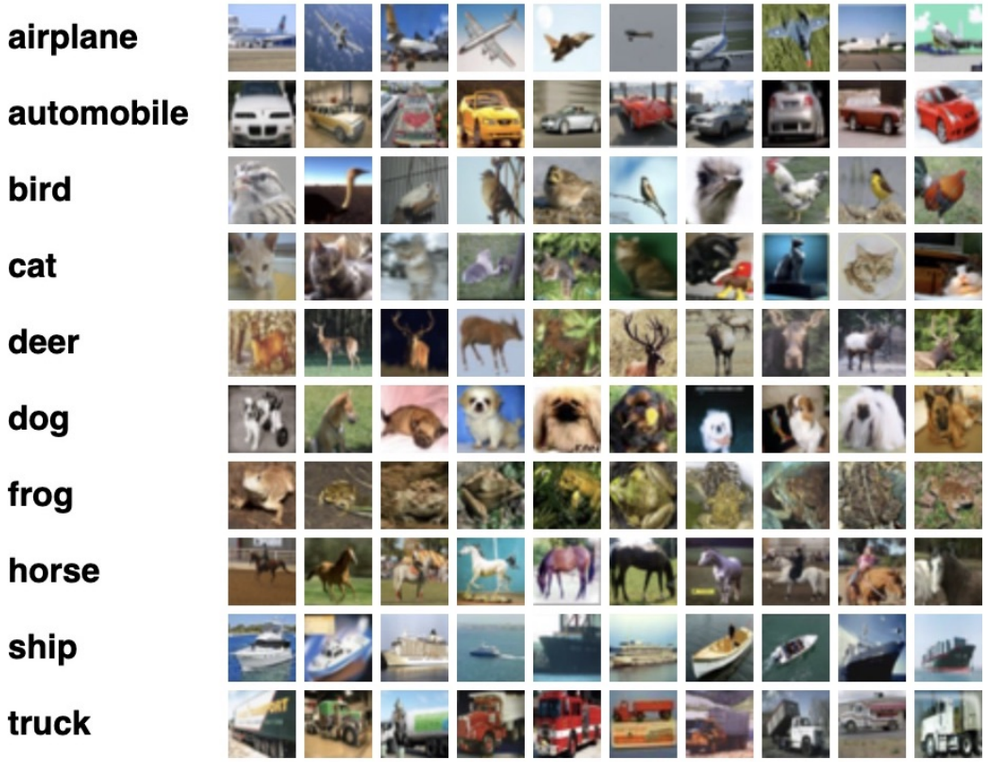


Experiments

Components

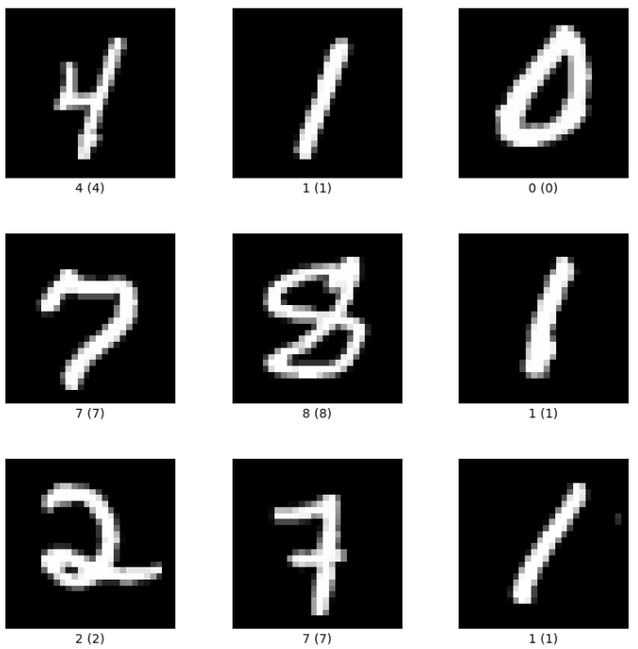
- Experiments apply on five workstations with Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz and 24 NVidia GeForce GTX 1080 GPUs.

- The client task is MNIST, and the pre-trained once-for-all network is based on MobileNet V2 and ProxylessNAS with Cifar-10



Cifar-10

Data Center



MNIST

User

Experiments

TABLE I
PFP-NAS PERFORMANCE IN SINGLE MACHINE AND MULTIPLE MACHINES

machine platform	dispatched process	tuner algorithm	psuedo trial	latency mape	avg_corr	max_latency	top-1 (%)	top-5(%)
local	2	anneal	0	0	1	85	94.22	94.16
local	4						94.39	94.18
controller	12						94.49	94.41

TABLE III
PFP-NAS PERFORMANCE IN DIFFERENCT HYPERPARAMETER SEARCHING METHODS.

dispatched process	tuner algorithm	psuedo trial	latency MAPE	avg_corr	max_latency	top-1 (%)	top-5 (%)
6	tpe	300	0.05	0.92	85	94.08	94.026
	random_search					93.99	93.762
	anneal					94.24	94.16

Experiments

TABLE II
PFP-NAS PERFORMANCE IN DIFFERENT TUNER TRIAL REQUIREMENTS

dispatched process	tuner algorithm	psuedo trial	latency MAPE	avg_corr	max_latency	top-1(%)	top-5 (%)
12	anneal	0	0	1	85	94.49	94.41
		10	0.05	0.92		94.24	94.154
		150				94.47	94.418
		300				94.44	94.24
		1000				94.11	94.088
24		300				94.49	94.358
6					125	94.01	93.822
					165	93.96	93.886
					205	94.18	93.89
					245	94.08	94.036
					285	94.14	93.918

Conclusion

We have proposed a framework that integrates NAS and hyperparameter recommendations so that the party with more computing power can share the results of the calculation with users with less computing power.

Users can use this framework without giving out their own datasets to achieve privacy protection

Next stage, we will leverage our proposed method into diversity experiments mainly to extend the Once-for-All module with other popular networks (not only the MobileNet V2), and apply the PFP-NAS to other application scenarios.

Thank you!