Morphling: Fast, Near-Optimal Auto-Configuration for Cloud-Native Model Serving

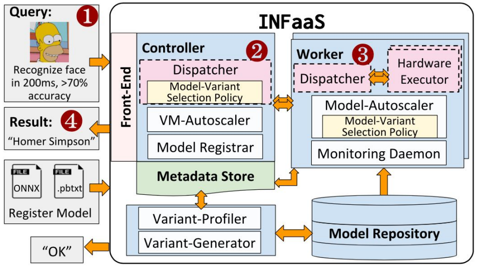
Alibaba Group

발표자: 최재강

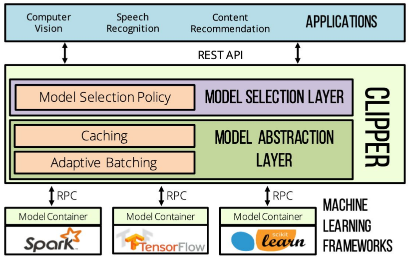
To optimize real-time inference system

Without violating the response-time SLO(Service Level Objective)

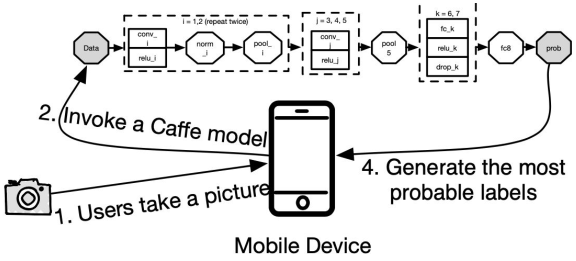
InFaaS



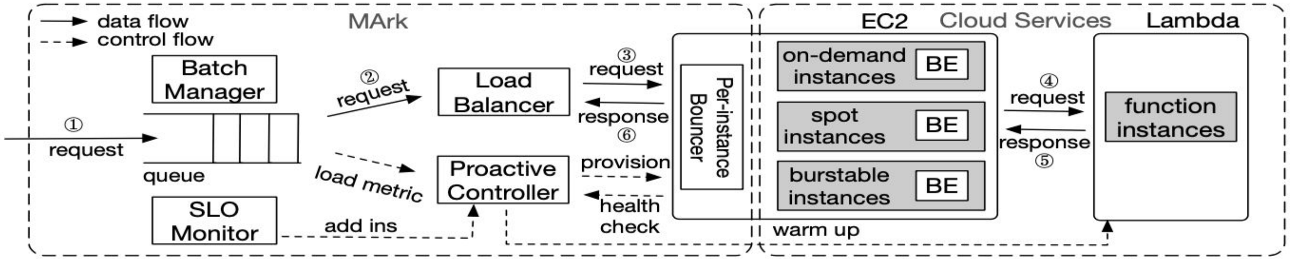
Clipper



In Mobile Application



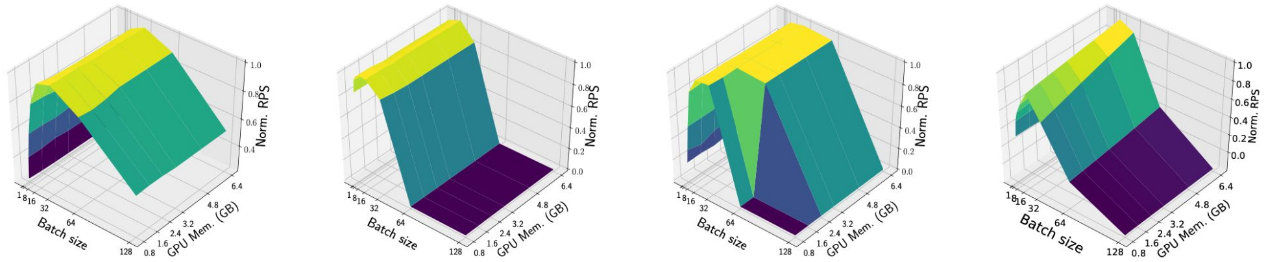
MArk



Configuration factors

Resource configuration

- CPU Cores
- GPU Memory
- GPU Timeshare
- GPU Type

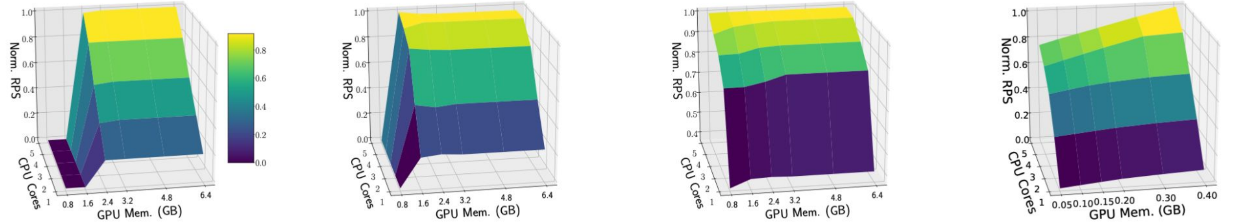


(a) ResNet101 (image class.) (b) ALBERT (language proc.) (c) VGG16 (image class.) (d) Meta model θ^m.

Figure 3. Normalized RPS under different configurations of batch size and GPU memory.

Runtime Parameters

- Batch Size



(a) Uni.Sen.Enc. (language proc.) (b) Word2vec-250 (language proc.) (c) ResNet50 (image class.) (d) Meta model θ^m.

Motivation: resembling configuration-RPS planes -> **Morphling**

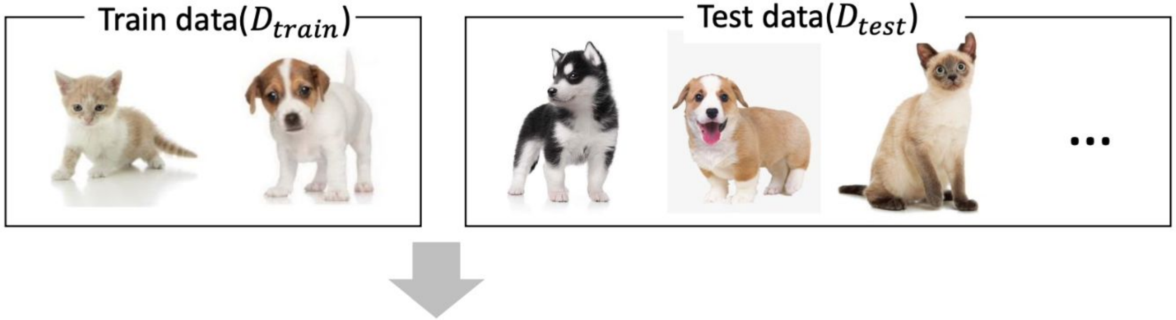
Meta-Learning

- 적은 데이터의 정의

  ✓ **N-way, k-shot**

  Classes : N 개

  Examples : k 개



2-way 1-shot classification

Train data는 총 n x k 개의 data point(x, y)로 이루어져 있음

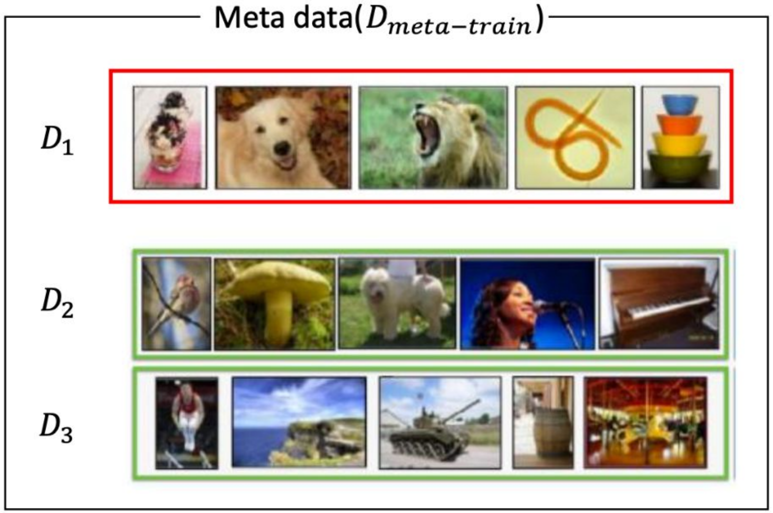
Test data의 경우 개수는 크게 상관이 없으며 test의 label은 학습에 사용하지 않음

Meta-Learning

- Meta-learning에 사용되는 데이터

✓ $\boldsymbol{D_{meta-train}}$는 train data와 비슷한 Task를 할 수 있는 다양한 데이터셋들로 이루어져 있음

$$D_{meta-train} = (D_1, D_2, D_3, \dots)$$



✓ 다양한 class들의 데이터가 가능

Train data의 class가 개, 고양이 라도

$D_{meta-train}$의 class는 사자, 사람, 전차 등 가능

✓ 비슷한 task의 예시

기존 task(T) : Train data를 이용하여 개와 고양이를 구분할 수 있는 파라미터(ø)를 학습시키는 것

Meta-task(T_1) : D_1을 이용하여 그릇과 사자를 구분할 수 있는 파라미터(ø)를 학습시키는 것

http://dmqm.korea.ac.kr/activity/seminar/265

Meta-Learning

Optimization-based Approach

- meta-train dataset D를 이용하여 $\boldsymbol{\theta}$ 를 구함
- 새로운 D와 기존 $\boldsymbol{\theta}$를 이용하여 새로운 Task의 최적화된 $\boldsymbol{\theta}$를 구함

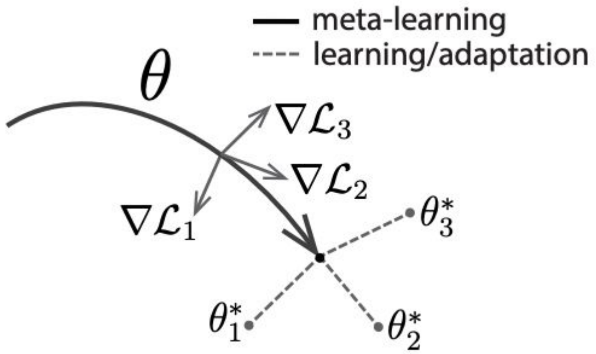


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

Reference: Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Meta-Learning

Optimization-based Approach

- ✓ $\emptyset_i \leftarrow \theta - \alpha\nabla_\theta L(\theta, D_i^{tr})$

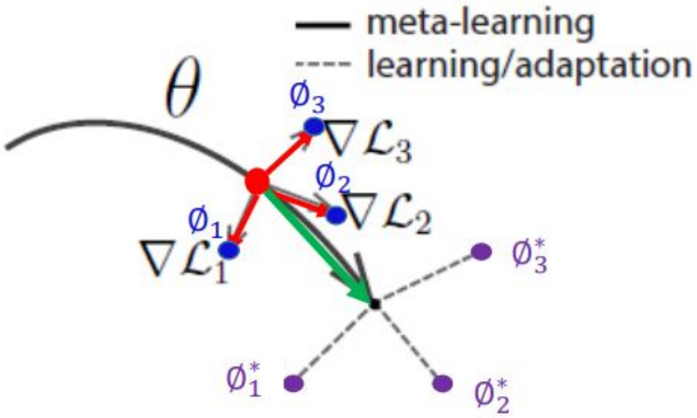
  θ를 $\emptyset_i$의 weight initialization으로 사용

  D_i^{tr}의 양이 적기 때문에 적은 update 만으로 θ -> $\emptyset_i$

- ✓ $\theta \leftarrow \theta - \beta\nabla_\theta \sum L(\emptyset_i, D_i^{test})$

  $L(\emptyset_i, D_i^{test})$ 가 최소인 경우는 $L(\emptyset_i^*, D_i^{test})$

  즉 $\emptyset_i$ = $\emptyset_i^*$가 되는 방향으로 θ를 업데이트



- ✓ 즉, 적은 update 만으로 $\emptyset_i^*$를 구할 수 있는 θ를 찾는 것이 meta-learning의 목적

http://dmqm.korea.ac.kr/activity/seminar/265

Meta-Model Training

Auto-Configuration Using Search

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{A}} \quad f(\boldsymbol{x}),$$

1. Black-box search - SMBO(Sequential Model-Based-Optimization)
   : 하나하나 회귀해 가며 최적의 구성을 찾는 방식

2. White-box prediction
   : 특성 구성을 성능을 예측하고 이를 사용하여 검색 프로세서 구동하는 접근 방식

3. Similarity-based search
   : 워크로드간 유사도 측정을 통해 다음 검색에 유사한 구성으로 선택

Meta-Model Training

Few-shot Regression

Goal: small number to minimize the training overhead

- Symbol

T_i : regression task

$\mathcal{L}_{T_i}$: loss function

$f_{\theta_i}(\boldsymbol{x})$: mapping function

$\mathcal{D} = \{\boldsymbol{x}^j, y^j | j = 1, 2, \ldots, K\}$

$\mathcal{T} = \{T_1, T_2, \ldots T_N\}$: set of regression tasks

Meta-Model Training

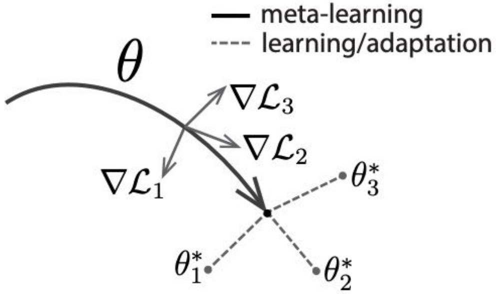
Model-Agnostic Meta-Learning (MAML)

Algorithm 1: MAML for Few-Shot Regression

Input : Regression task set of N inference models: $\mathcal{T}$, learning rates α and β, sampling budget K, meta training episodes E

Output : A well-trained meta model with parameters θ^m

```
Randomly initialize θ^m
for episode = 1, 2, ..., E do
    for all T_i ∈ 𝒯 do
        Sample K datapoints 𝒟_{||K||} = {x^j, y^j} from T_i
        Evaluate ∇_{θ^m} ℒ_{T_i}(f_{θ^m}) using 𝒟 and ℒ_{T_i} in Eq. (2)
        Compute adapted model θ_i with SGD using Eq. (3)
    Update meta model θ^m:
      θ_m ← θ_m − β∇_{θ_m} Σ_{T_i∼𝒯} ℒ_{T_i}(f_{θ_i})
```



$$\mathcal{L}_{T_i}(f_{\theta_i}) = \textstyle\sum_{\boldsymbol{x}^j, y^j \sim T_i} ||f_{\theta_i}(\boldsymbol{x}^j) - y^j||_2^2. \tag{2}$$

$$\theta_i = \theta^m - \alpha \nabla_{\theta^m} \mathcal{L}_{T_i}(f_{\theta^m}), \tag{3}$$

Directing SMBO Search with Meta-Model

Meta-Model as an Initial Regression Model

Algorithm 2: SMBO with Meta Model

Input : A new regression task T_i, learning rates α, sampling budget K, meta model θ^m

Output : The optimal configuration $\boldsymbol{x}^*$

```
Initialize θ_i ← θ^m, and newly-sampled data set D ← {}
for k = 1, 2, ..., K do
    Update regression model θ'_i = θ_i − α∇_{θ_i} L_{T_i}(f_{θ_i})
    for all x ∈ A do
        Calculate the f_{θ_i}(x) and Acq(f_{θ_i}(x)) using Eq. (6)
    x^k ← arg max_{x ∈ A, x ∉ D} Acq(f_{θ_i}(x))
    Estimate y^k for x^k          ▷ Real-world evaluation
    D ← D ∪ (x^k, y^k), θ_i ← θ'_i
x* ← arg max_{x^k ∈ D} y^k
```

Directing SMBO Search with Meta-Model

Algorithm 2: SMBO with Meta Model

Input : A new regression task T_i, learning rates α, sampling budget K, meta model θ^m

Output : The optimal configuration $\boldsymbol{x}^*$

Initialize $\theta_i \leftarrow \theta^m$, and newly-sampled data set $\mathcal{D} \leftarrow \{\}$

for $k = 1, 2, \ldots, K$ **do**

  Update regression model $\theta_i' = \theta_i - \alpha \nabla_{\theta_i} \mathcal{L}_{T_i}(f_{\theta_i})$

  for *all* $\boldsymbol{x} \in \mathcal{A}$ **do**

    Calculate the $f_{\theta_i}(\boldsymbol{x})$ and $\mathrm{Acq}(f_{\theta_i}(\boldsymbol{x}))$ using Eq. (6)

  $\boldsymbol{x}^k \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{A}, \boldsymbol{x} \notin \mathcal{D}} \mathrm{Acq}(f_{\theta_i}(\boldsymbol{x}))$

  Estimate y^k for $\boldsymbol{x}^k$ ▷ Real-world evaluation

  $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{x}^k, y^k)$, $\theta_i \leftarrow \theta_i'$

$\boldsymbol{x}^* \leftarrow \arg\max_{\boldsymbol{x}^k \in \mathcal{D}} y^k$

$$\mathrm{Conf}(f_{\theta_i'}(\boldsymbol{x})) = |f_{\theta_i}(\boldsymbol{x}) - f_{\theta_i'}(\boldsymbol{x})|.$$

$$\mathrm{Acq}(f_{\theta_i}(\boldsymbol{x})) = f_{\theta_i}(\boldsymbol{x}) + \delta \mathrm{Conf}(f_{\theta_i}(\boldsymbol{x})),$$

Directing SMBO Search with Meta-Model

Algorithm 2: SMBO with Meta Model

Input : A new regression task T_i, learning rates α, sampling budget K, meta model θ^m

Output : The optimal configuration $\boldsymbol{x}^*$

```
Initialize θ_i ← θ^m, and newly-sampled data set D ← {}
for k = 1, 2, ..., K do
    Update regression model θ'_i = θ_i − α∇_{θ_i} L_{T_i}(f_{θ_i})
    for all x ∈ A do
        Calculate the f_{θ_i}(x) and Acq(f_{θ_i}(x)) using Eq. (6)
    x^k ← arg max_{x∈A, x∉D} Acq(f_{θ_i}(x))
    Estimate y^k for x^k        ▷ Real-world evaluation
    D ← D ∪ (x^k, y^k), θ_i ← θ'_i
x* ← arg max_{x^k ∈ D} y^k
```

$$\mathrm{Conf}(f_{\theta'_i}(\boldsymbol{x})) = |f_{\theta_i}(\boldsymbol{x}) - f_{\theta'_i}(\boldsymbol{x})|.$$

$$\mathrm{Acq}(f_{\theta_i}(\boldsymbol{x})) = f_{\theta_i}(\boldsymbol{x}) + \delta \mathrm{Conf}(f_{\theta_i}(\boldsymbol{x})),$$

Evaluation with...

Traditional Auto-Configuration Methods

- Bayesian optimization (BO)
- Ernest : build a dedicated regression model
- Google Vizier : Only Black box optimization
- Fine-Tuning : Similarity-based search

Evaluation

Objective

- Cost = base price + GPU prices * GPU memory + CPU prices * CPU cores

$$\max_{x \in \mathcal{A}} \quad \text{RPS/Cost.}$$

Instance Type	# of CPUs	GPU Type	$/hour
g4dn.2xlarge	8	T4	0.75
p3.2xlarge	8	Tesla V100	3.06
g4ad.4xlarge	16	Tesla M60	0.87
c6g.4xlarge	16	None	0.54

Search Space

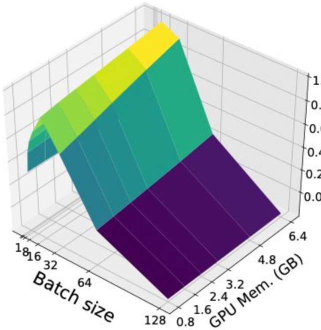
- Image Classification & NLP provided Tensorflow model zoo

Model Type	Model Families (# of models)
Img. Class.	ResNet (5), NASNet (2), VGG (2), Inception (2), DenseNet (1), MobileNet (2), EfficientNet (7),
Lang. Mod.	BERT (2), ALBERT (4), ELMo (1), NNLM (2), Small BERT (4), Word2vec (2), ELECTRA (2), Universal Sentence Encoder (4)

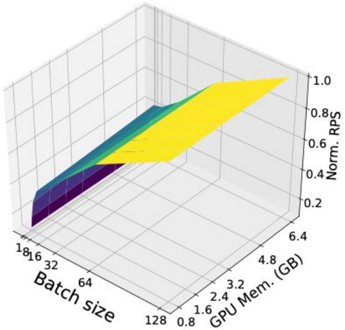
Configuration	Candidate choices
CPU cores	1, 2, 3, 4, 5
GPU memory	5%, 10%, 15%, 20%, 30%, 40%
Batch size	1, 2, 4, 8, 16, 32, 64, 128
GPU type	T4, Tesla V100, Tesla M60

Evaluation

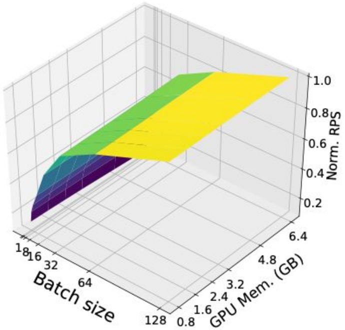
Fast adaptation for New Regression Tasks



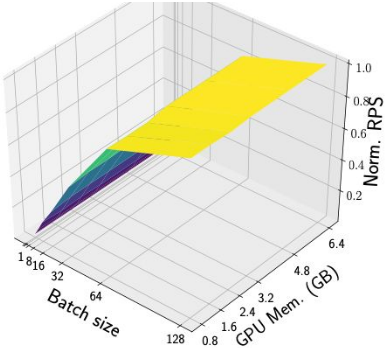
(a) Morphling: meta-model.



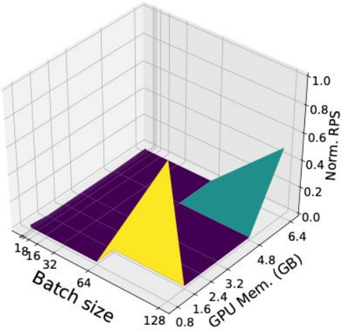
(b) Morphling: 8 initial samplings.



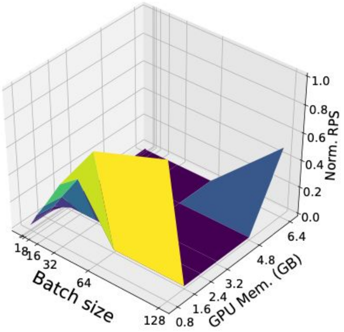
(c) Morphling: 8 + 20 samplings.



(d) Ground truth.

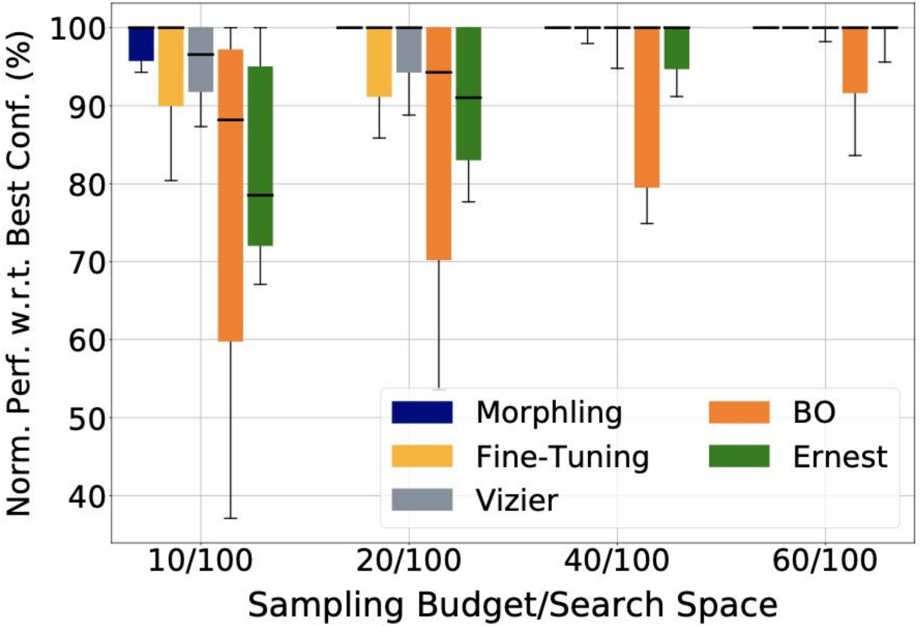


(a) BO: 8 initial samplings.

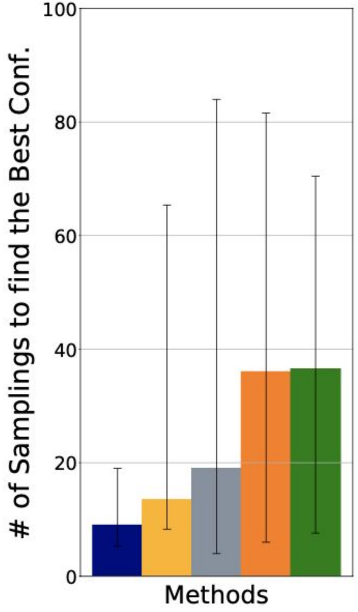


(b) BO: 8 + 20 samplings.

Evaluations of search quality and costs

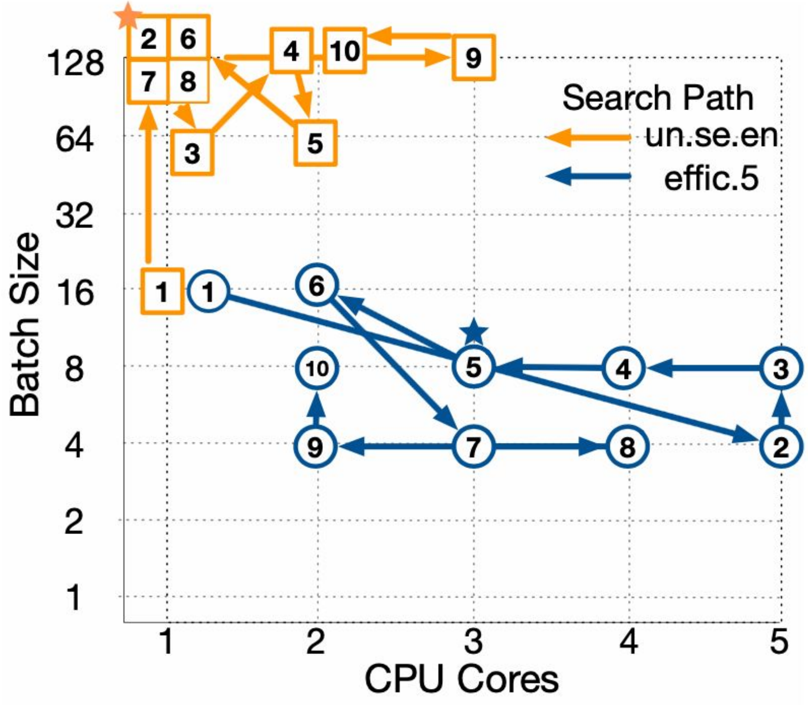


(a) The normalized performance of the identified configurations for the evaluated inference services under varying sampling budgets.



(b) Search costs to find the optimal configurations.

Search Path



(a) CPU cores and batch size.

Step	un.se.en	effic.5
1	Tesla M60	T4
2	Tesla M60	Tesla V100*
3	T4*	Tesla V100
4	T4	Tesla V100
5	T4	Tesla V100
6	T4	Tesla M60
7	T4	Tesla V100
8	Tesla M60	Tesla V100
9	T4	Tesla V100
10	T4	Tesla V100

(b) GPU type.

Conclusion

Summary

1. 메타 모델을 오프라인으로 훈련하여 다양한 구성에서 일반적인 성능 추세를 포착

2. 새로운 추론 서비스에 대한 구성 검색을 지시하는 초기 회귀 모델로 사용

3. 샘플링 예산이 소진될 때까지 모델을 fitting하고 이를 사용하여 탐색할 구성을 결정하는 작업을 반복

Merits

1. 기존 auto-configuration 방법보다 적은 샘플링으로 좋은 구성을 찾아냄

2. 검색에 드는 비용이 적어 비용적으로 효율적임

More ?

https://github.com/jaeriver/Paper_Review_and_Practice/tree/main/Morphling