Transfer Learning for Performance Analysis of Highly-Configurable Systems

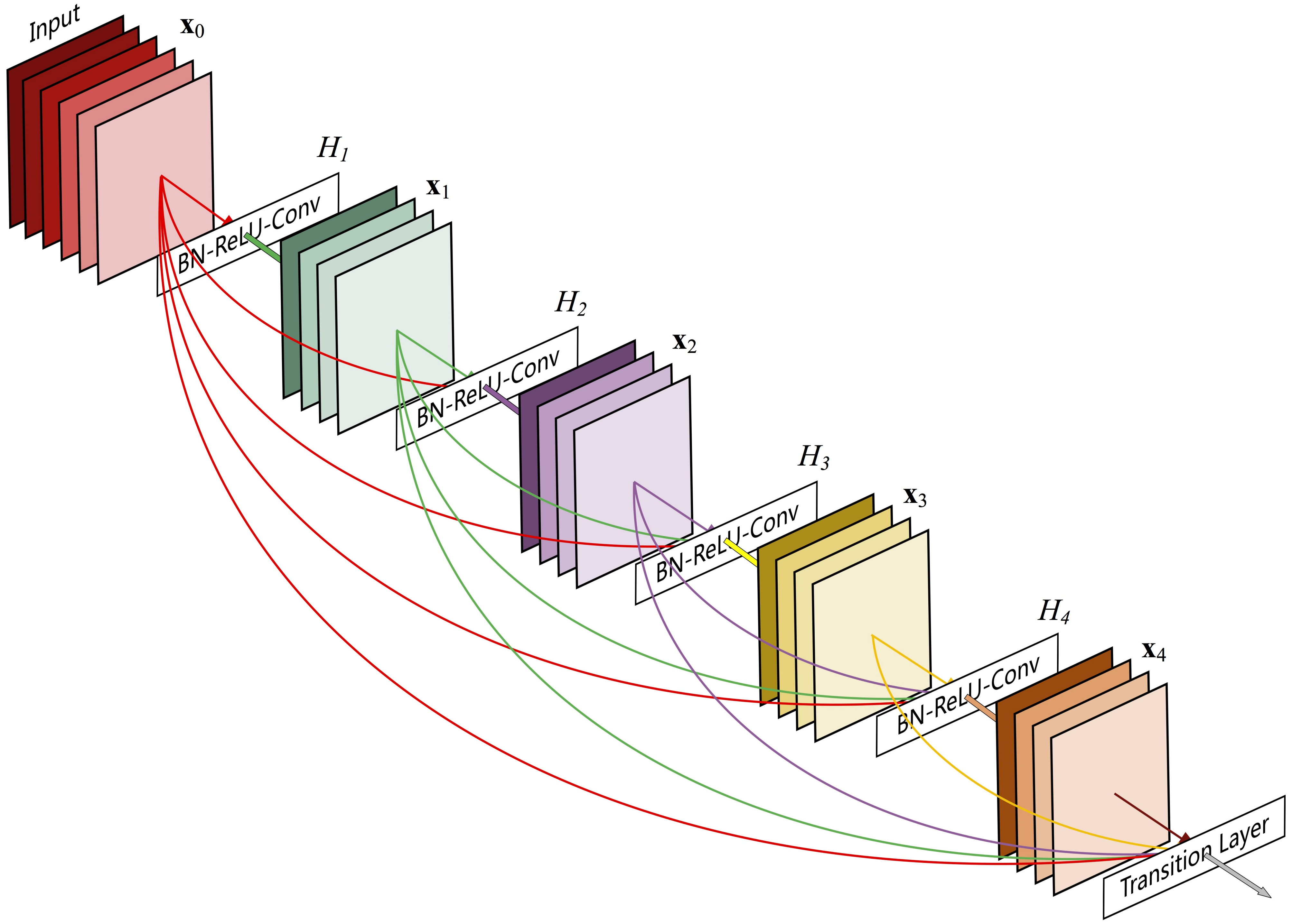
Pooyan Jamshidi
University of South Carolina
pooyanjamshidi


Snap It BETA suggestions from photo
Almonds
View All
Add foods in this photo

AT&T LTE
4:29 PM
65%
bitesnap
MACRO
PERCENT
TARGET
Carbs
47%
50%
Fat
29%
30%
Protein
23%
20%
Banana, Strawberries, Kiwi Fruit, Blackberries, Chia Seeds
232 calories
10:48 AM

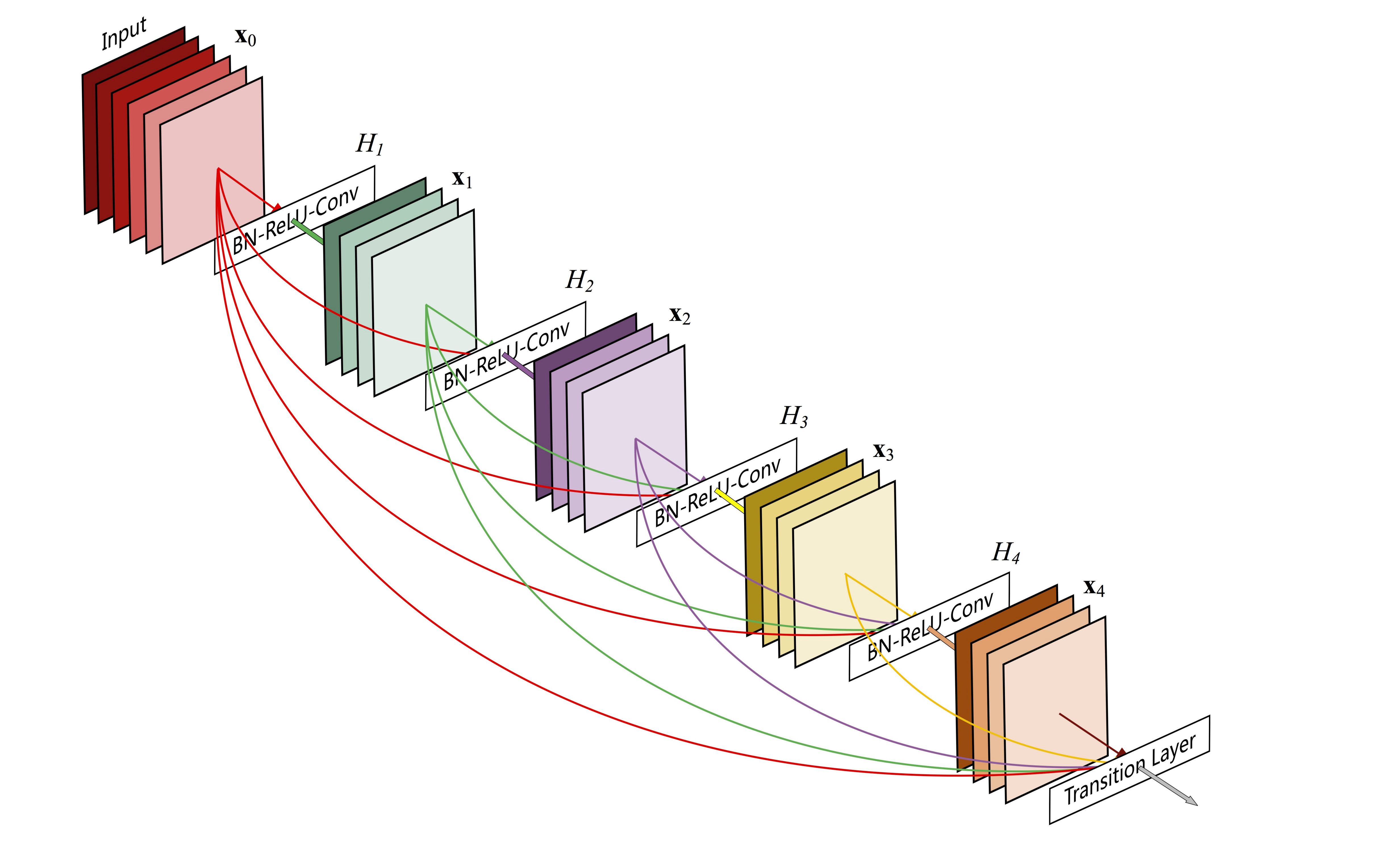

Snap It BETA suggestions from photo
Almonds
View All
Add foods in this photo

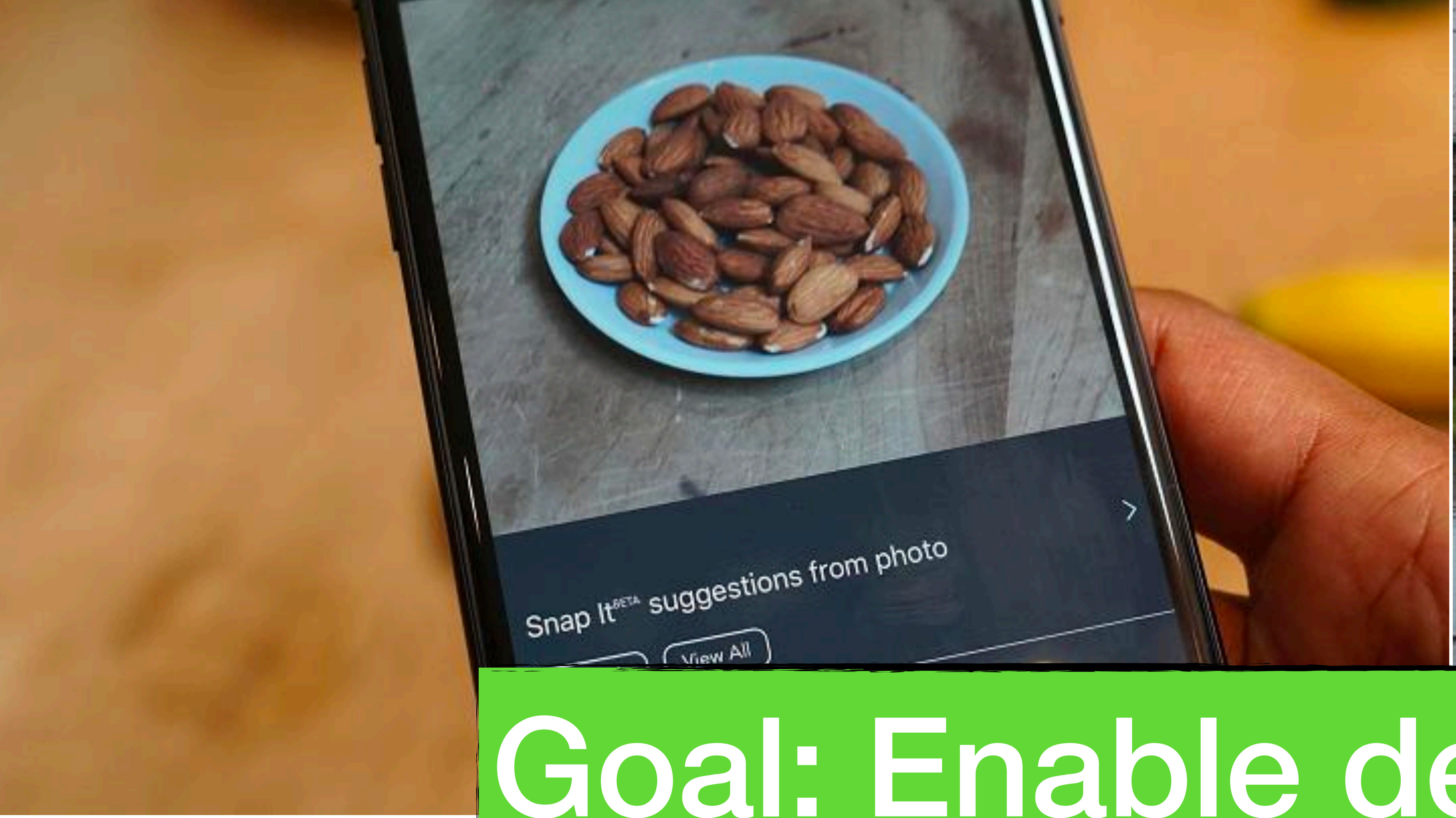

AT&T LTE
4:29 PM
65%
bitesnap
MACRO
PERCENT
TARGET
Carbs 47% 50%
Fat 29% 30%
Protein 23% 20%
Banana, Strawberries, Kiwi Fruit, Blackberries, Chia Seeds
232 calories
10:48 AM


Input
$\mathbf{x}_0$
H_1
BN-ReLU-Conv
$\mathbf{x}_1$
H_2
BN-ReLU-Conv
$\mathbf{x}_2$
H_3
BN-ReLU-Conv
$\mathbf{x}_3$
H_4
BN-ReLU-Conv
$\mathbf{x}_4$
Transition Layer

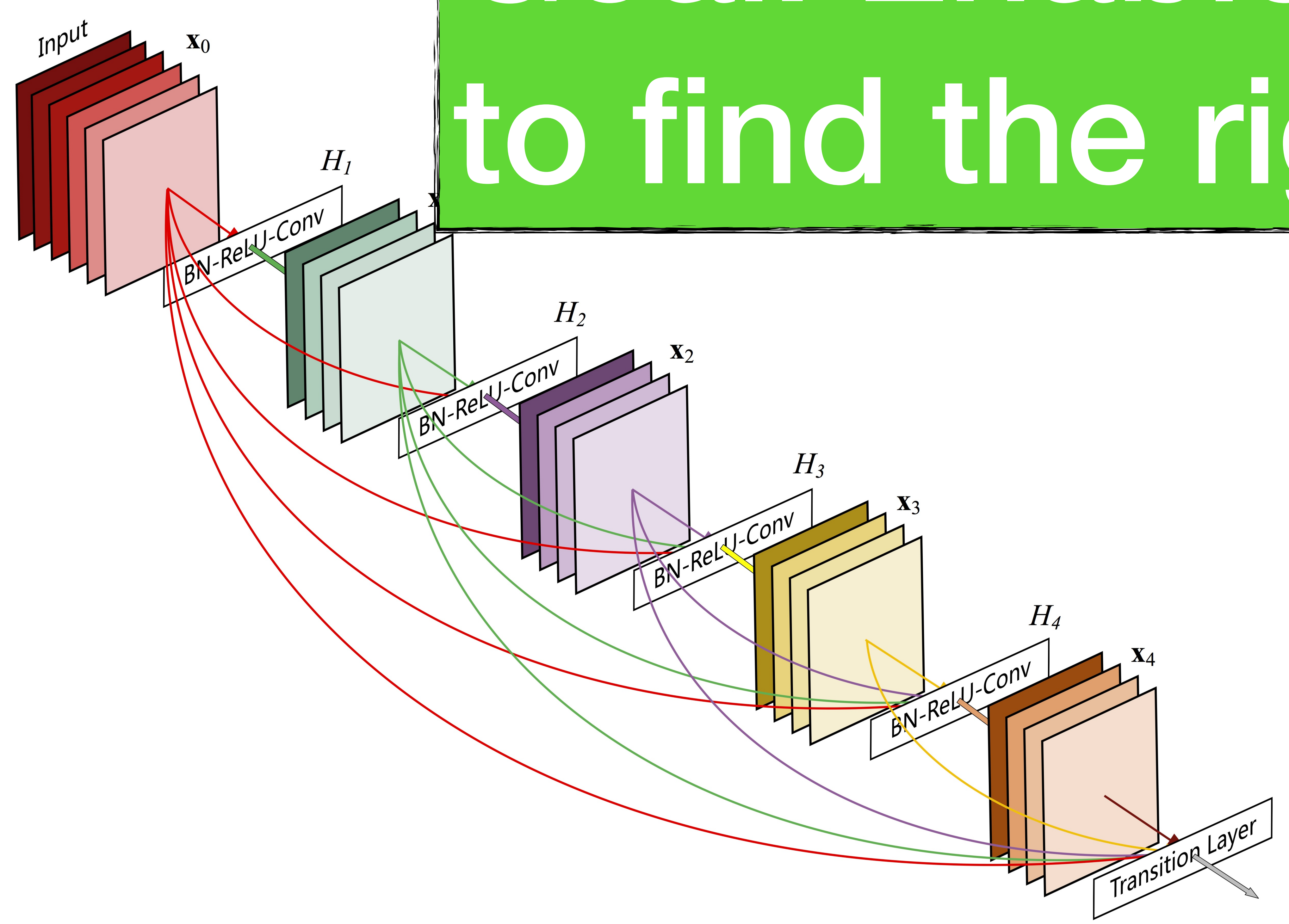

Snap It BETA suggestions from photo
Almonds
View All
Add foods in this photo


AT&T LTE
4:29 PM
65%
bitesnap
MACRO
PERCENT
TARGET
Carbs 47% 50%
Fat 29% 30%
Protein 23% 20%
Banana, Strawberries, Kiwi Fruit, Blackberries, Chia Seeds
232 calories
10:48 AM


Input
$\mathbf{x}_0$
H_1
BN-ReLU-Conv
$\mathbf{x}_1$
H_2
BN-ReLU-Conv
$\mathbf{x}_2$
H_3
BN-ReLU-Conv
$\mathbf{x}_3$
H_4
BN-ReLU-Conv
$\mathbf{x}_4$
Transition Layer



Goal: Enable developers/users to find the right quality tradeoff



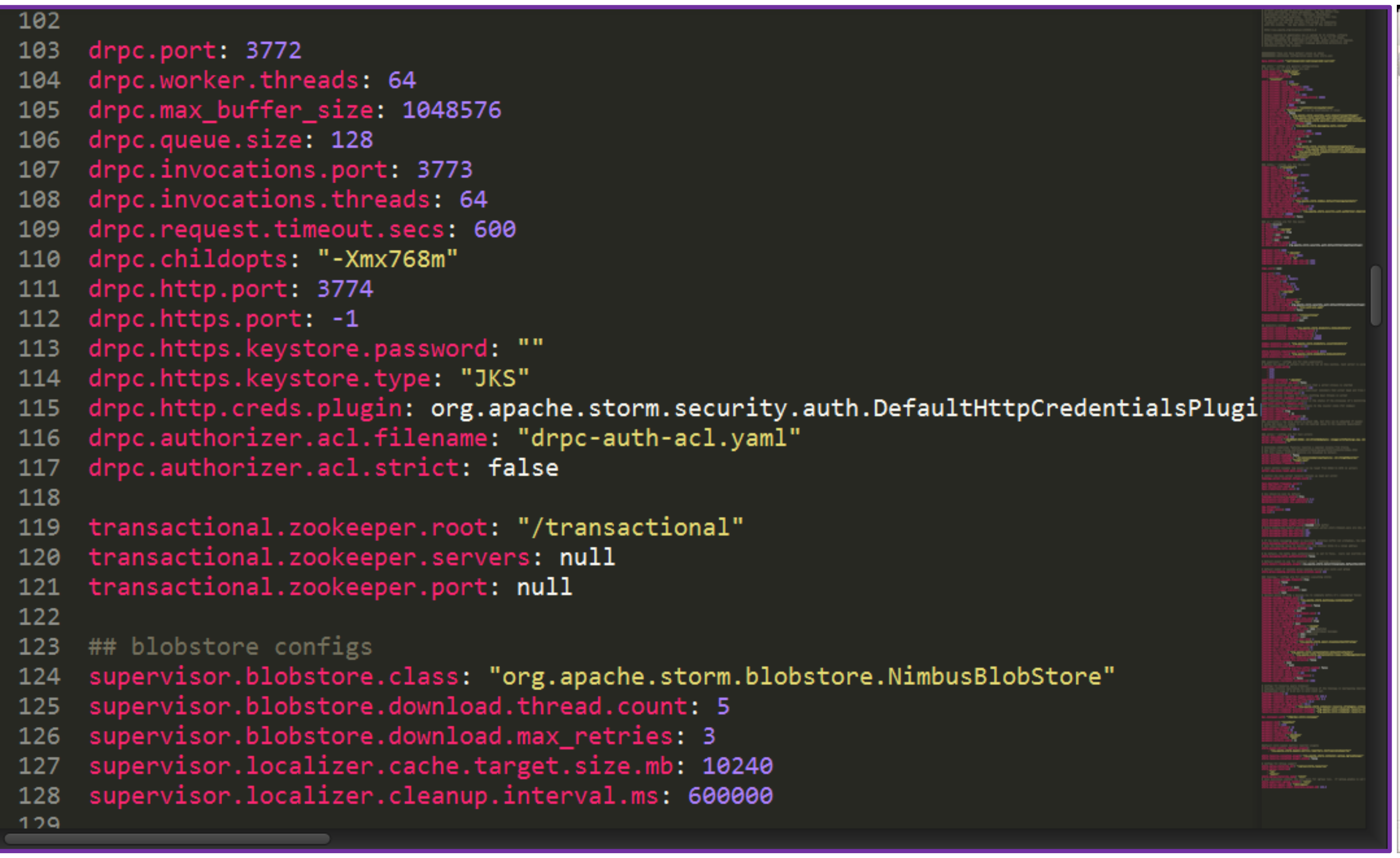
Today’s most popular systems are built configurable

Today's most popular systems are built configurable

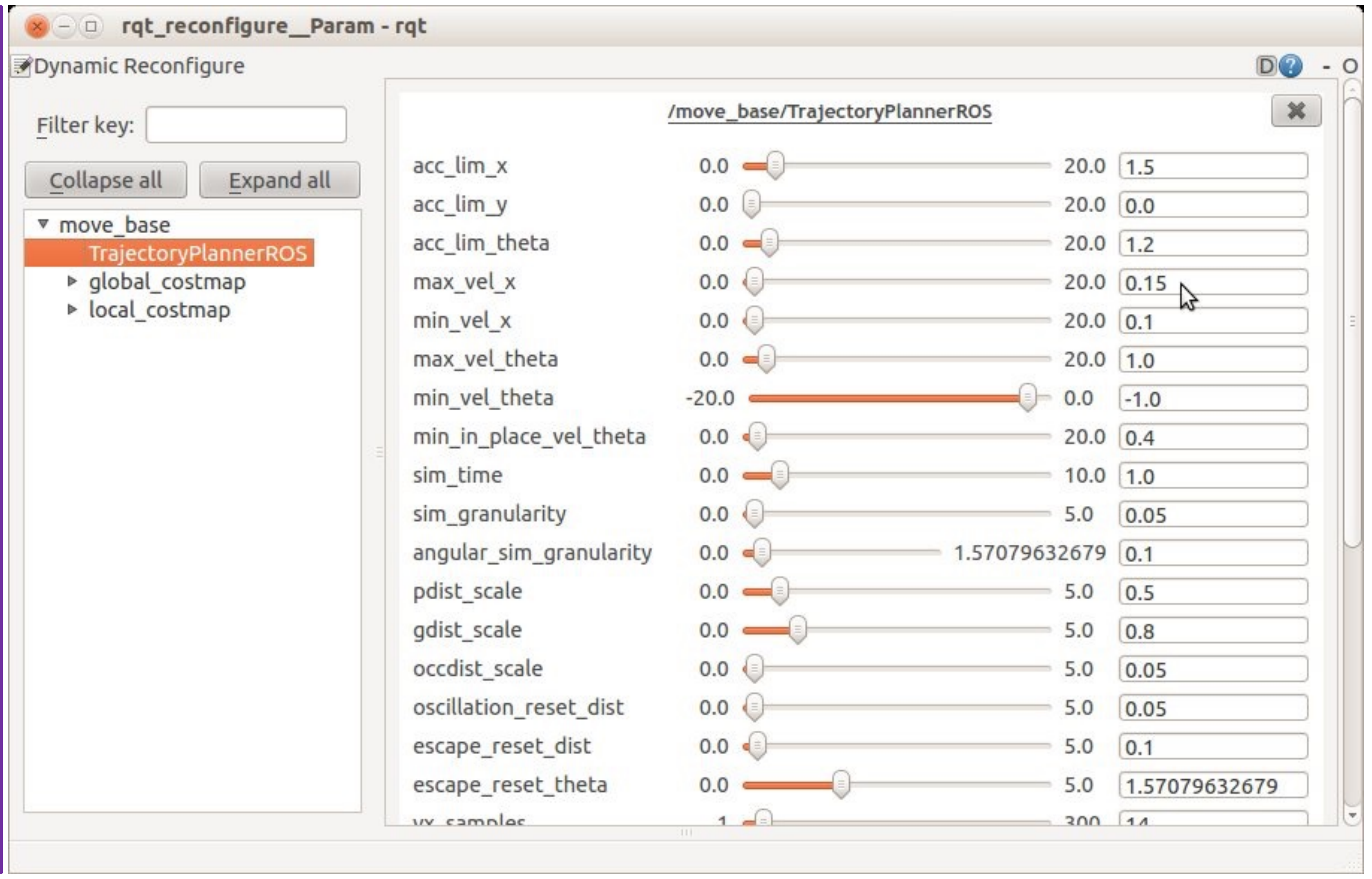
```
drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "JKS"
drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
drpc.authorizer.acl.strict: false

transactional.zookeeper.root: "/transactional"
transactional.zookeeper.servers: null
transactional.zookeeper.port: null

## blobstore configs
supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
supervisor.blobstore.download.thread.count: 5
supervisor.blobstore.download.max_retries: 3
supervisor.localizer.cache.target.size.mb: 10240
supervisor.localizer.cleanup.interval.ms: 600000
```

hadoop

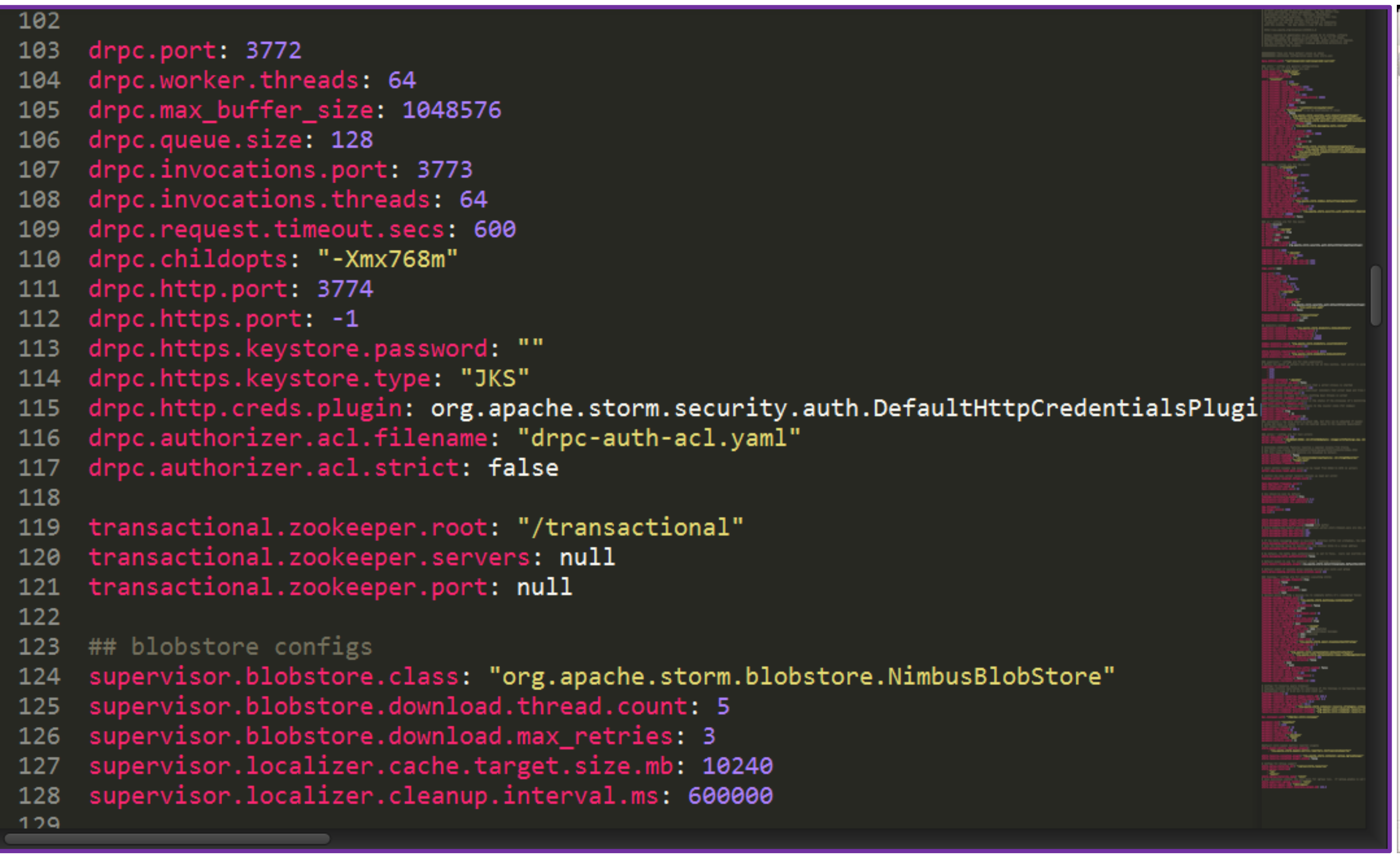
Today’s most popular systems are built configurable

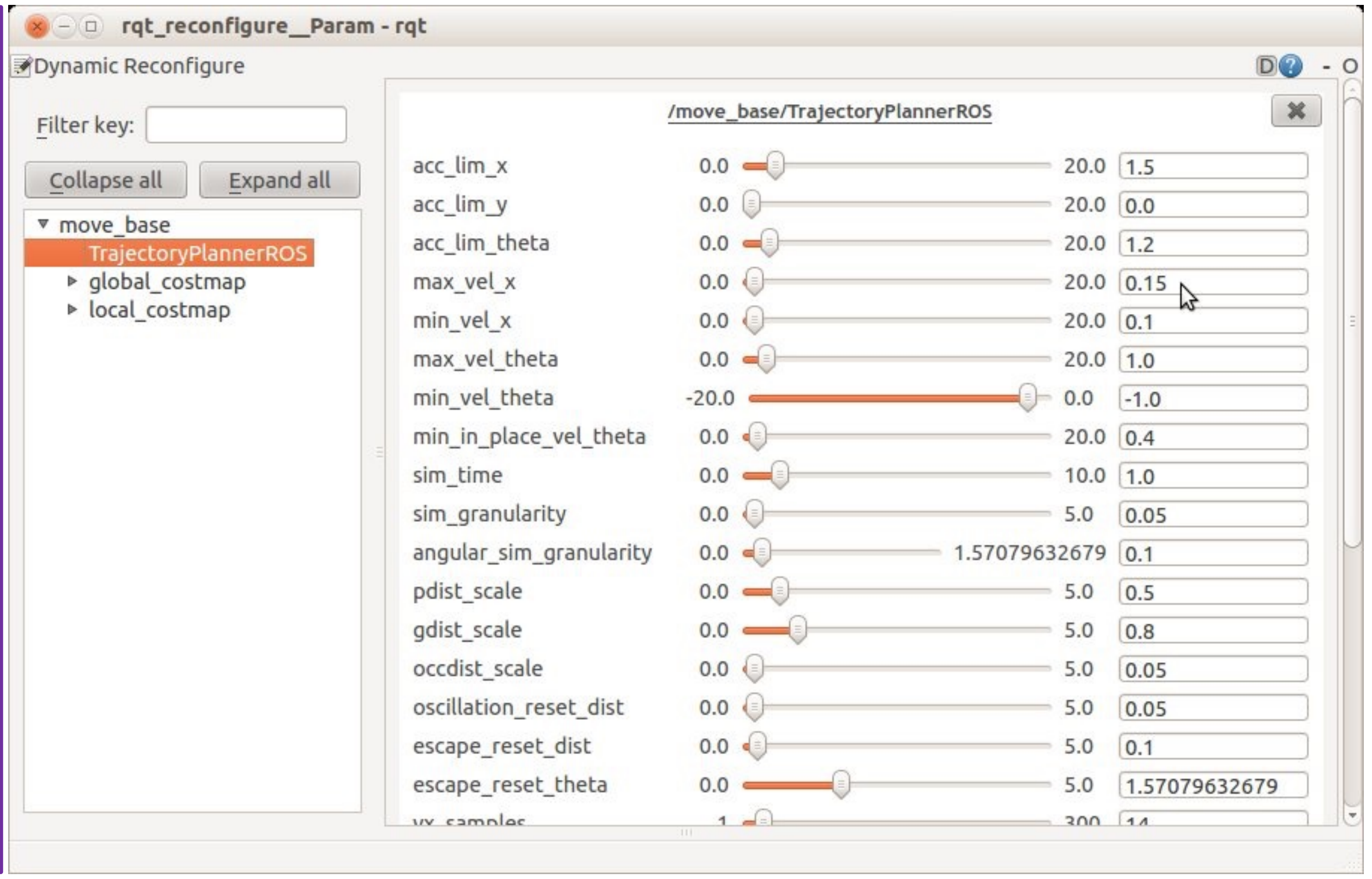


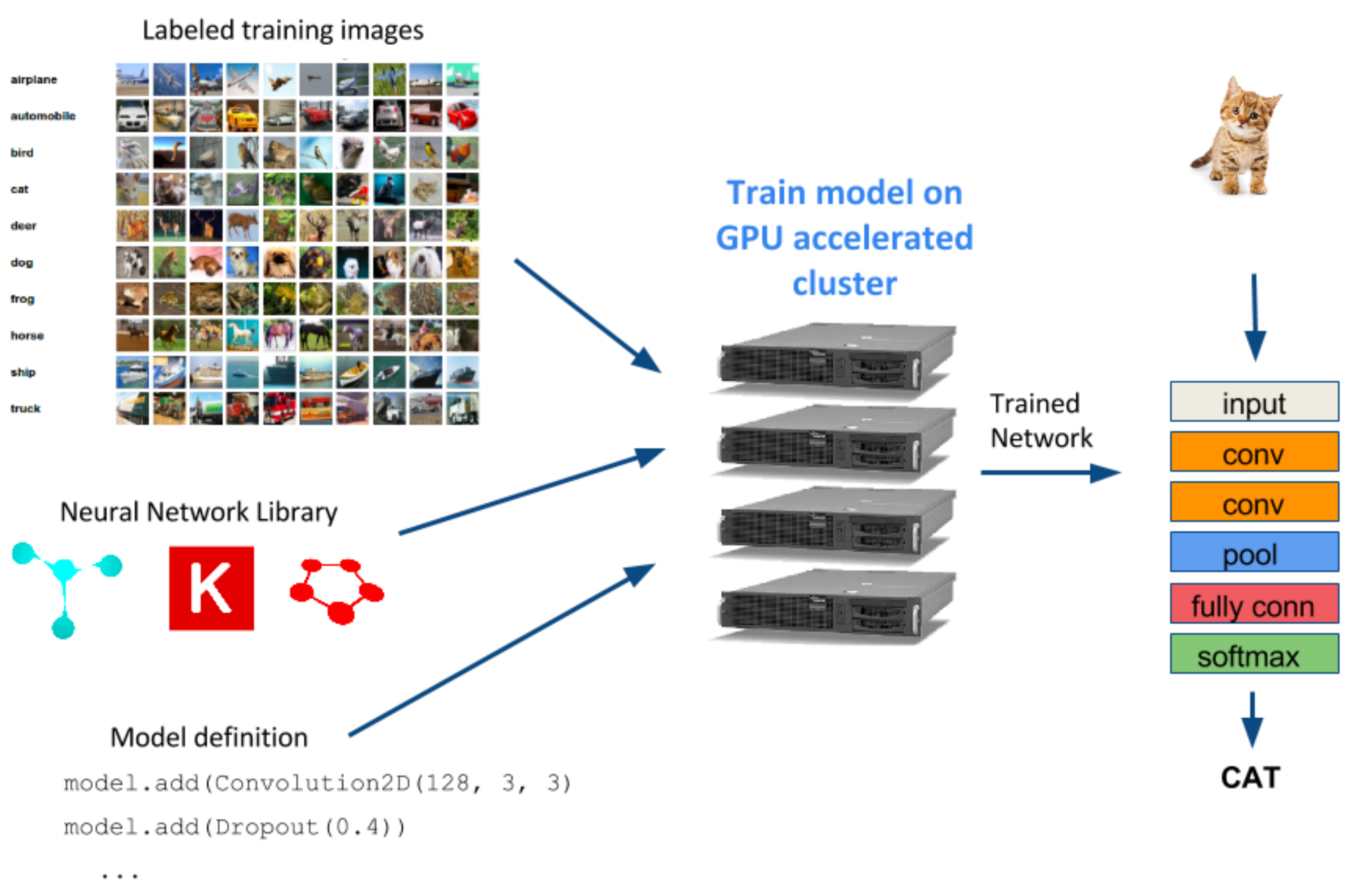
```
drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "JKS"
drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
drpc.authorizer.acl.strict: false

transactional.zookeeper.root: "/transactional"
transactional.zookeeper.servers: null
transactional.zookeeper.port: null

## blobstore configs
supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
supervisor.blobstore.download.thread.count: 5
supervisor.blobstore.download.max_retries: 3
supervisor.localizer.cache.target.size.mb: 10240
supervisor.localizer.cleanup.interval.ms: 600000
```

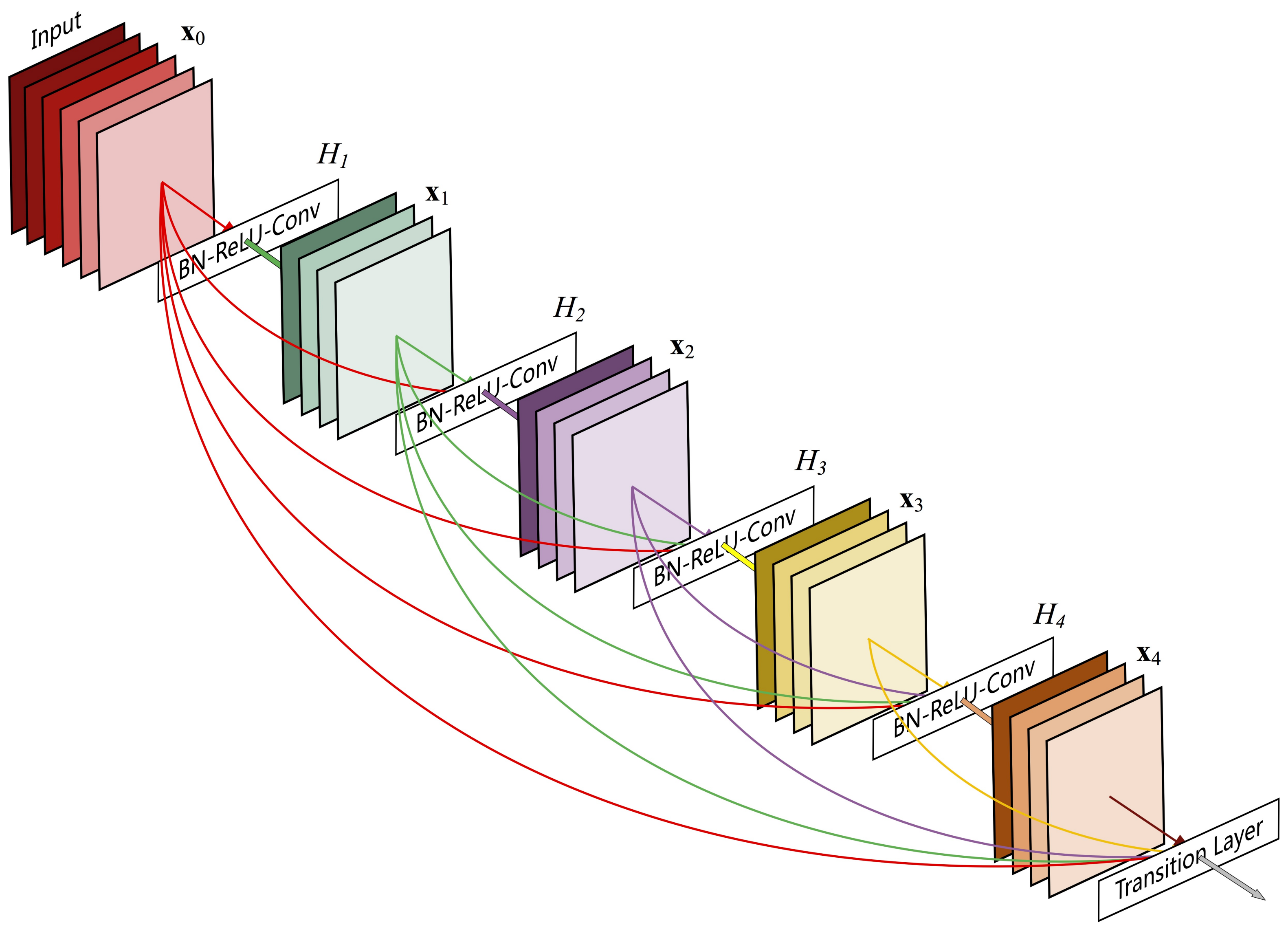


Today’s most popular systems are built configurable

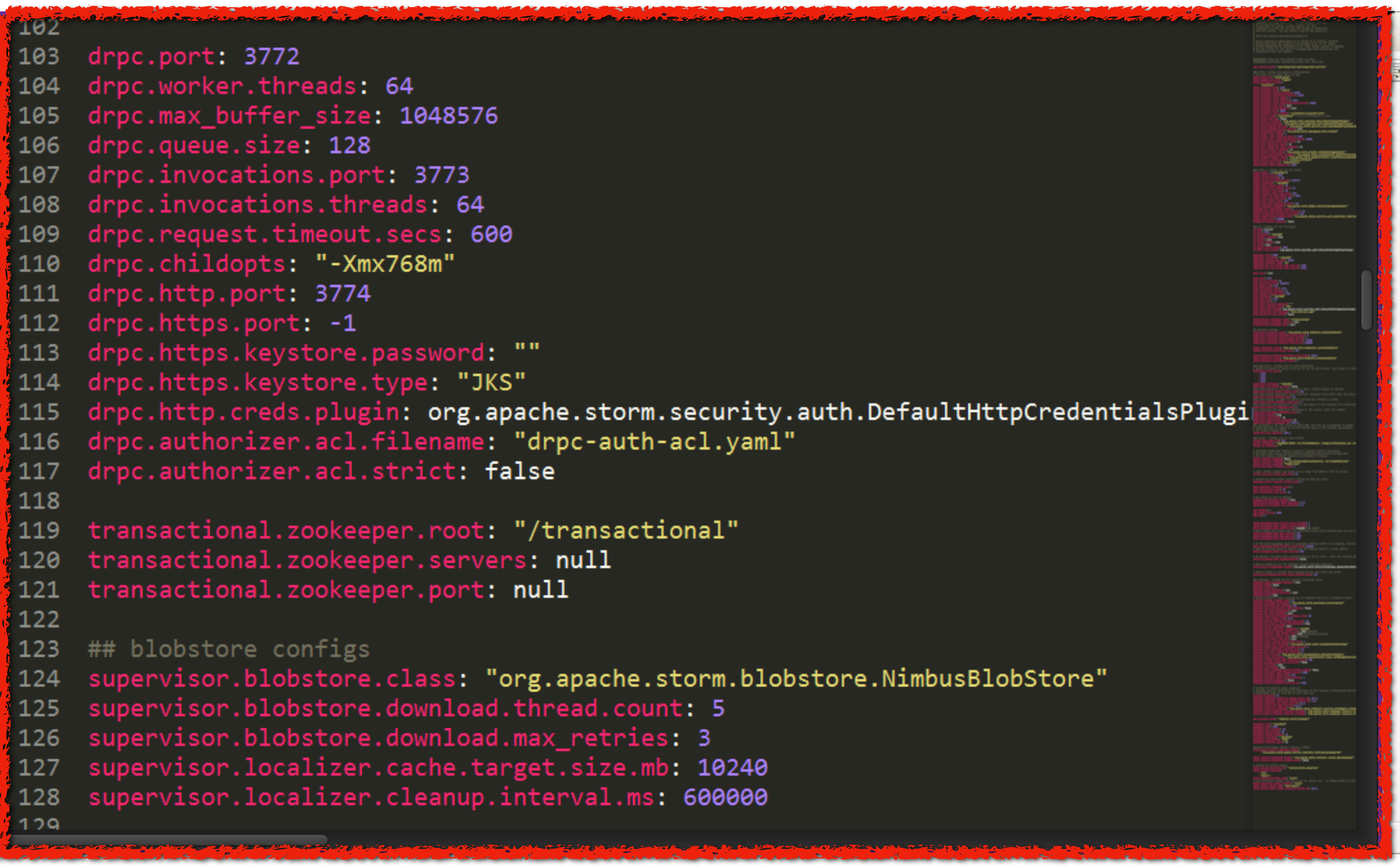


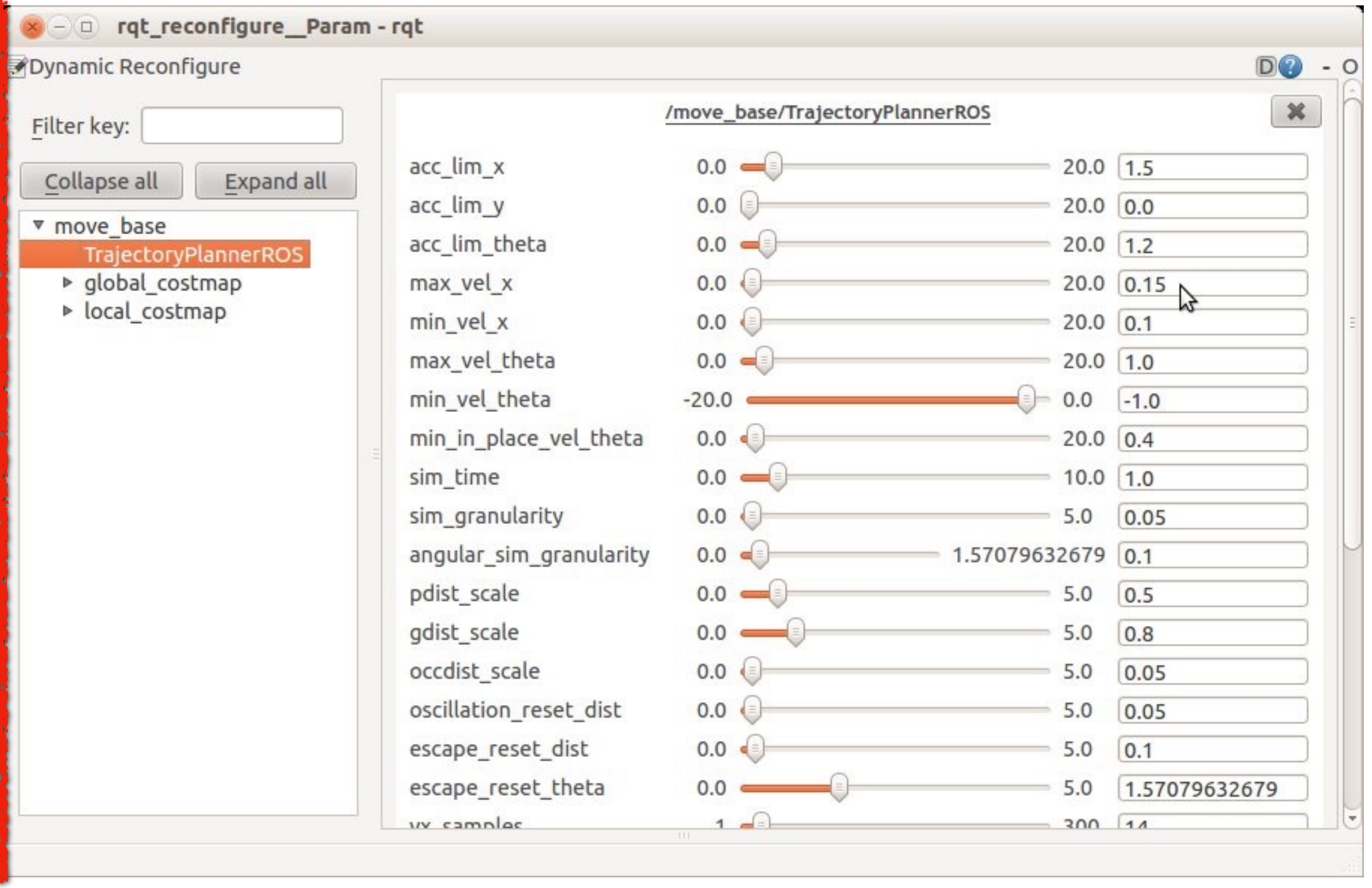


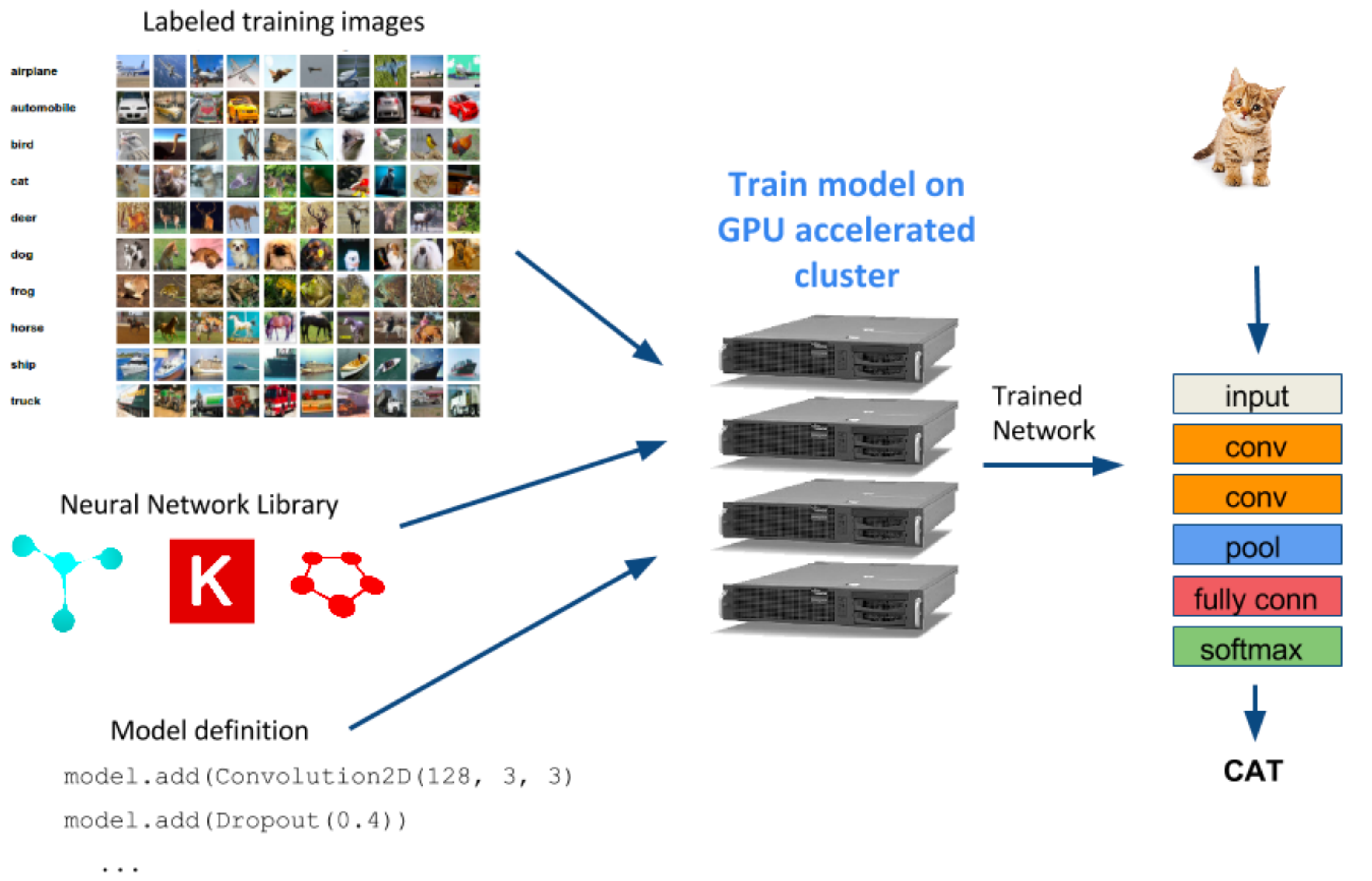


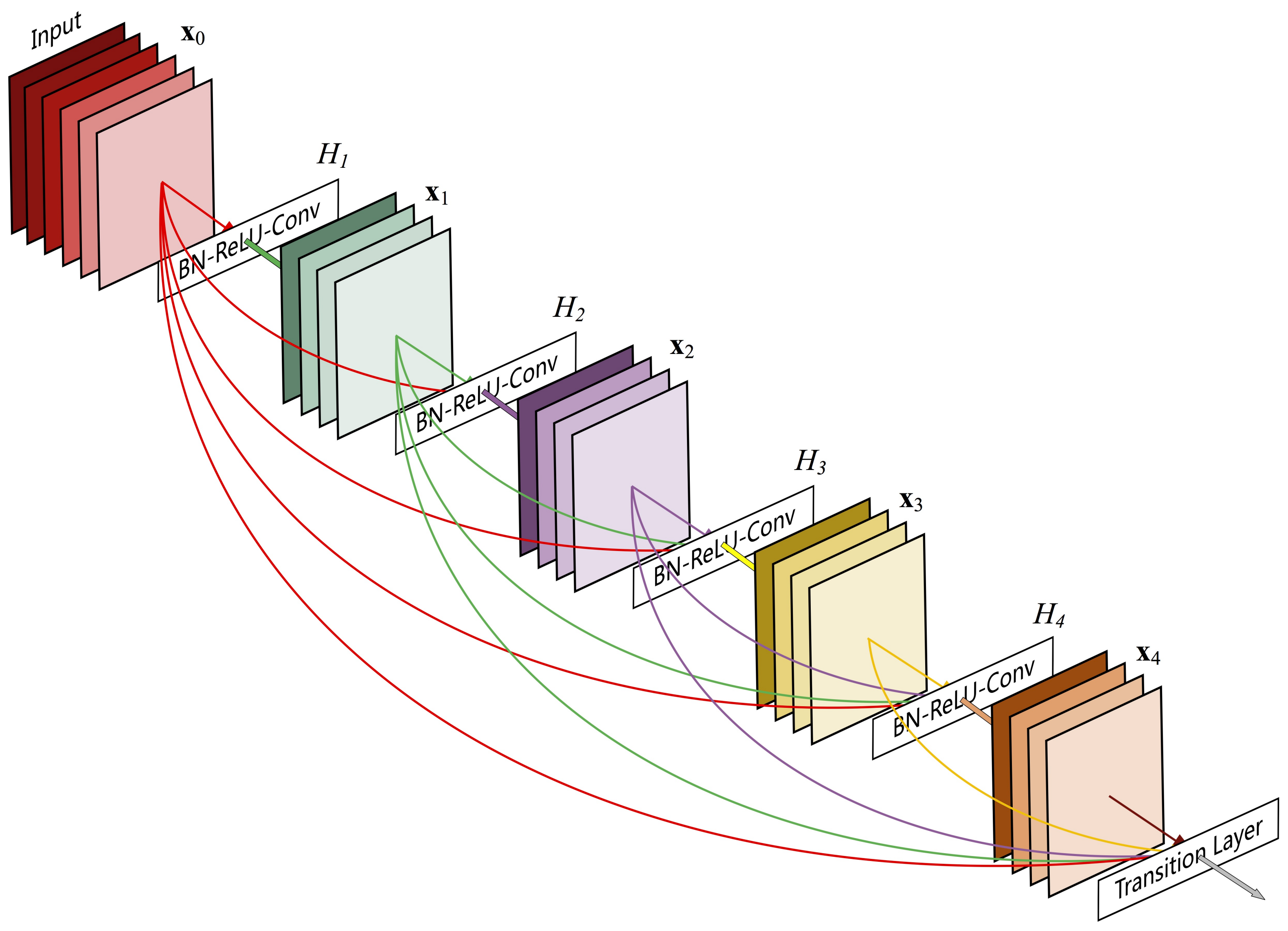


Today’s most popular systems are built configurable









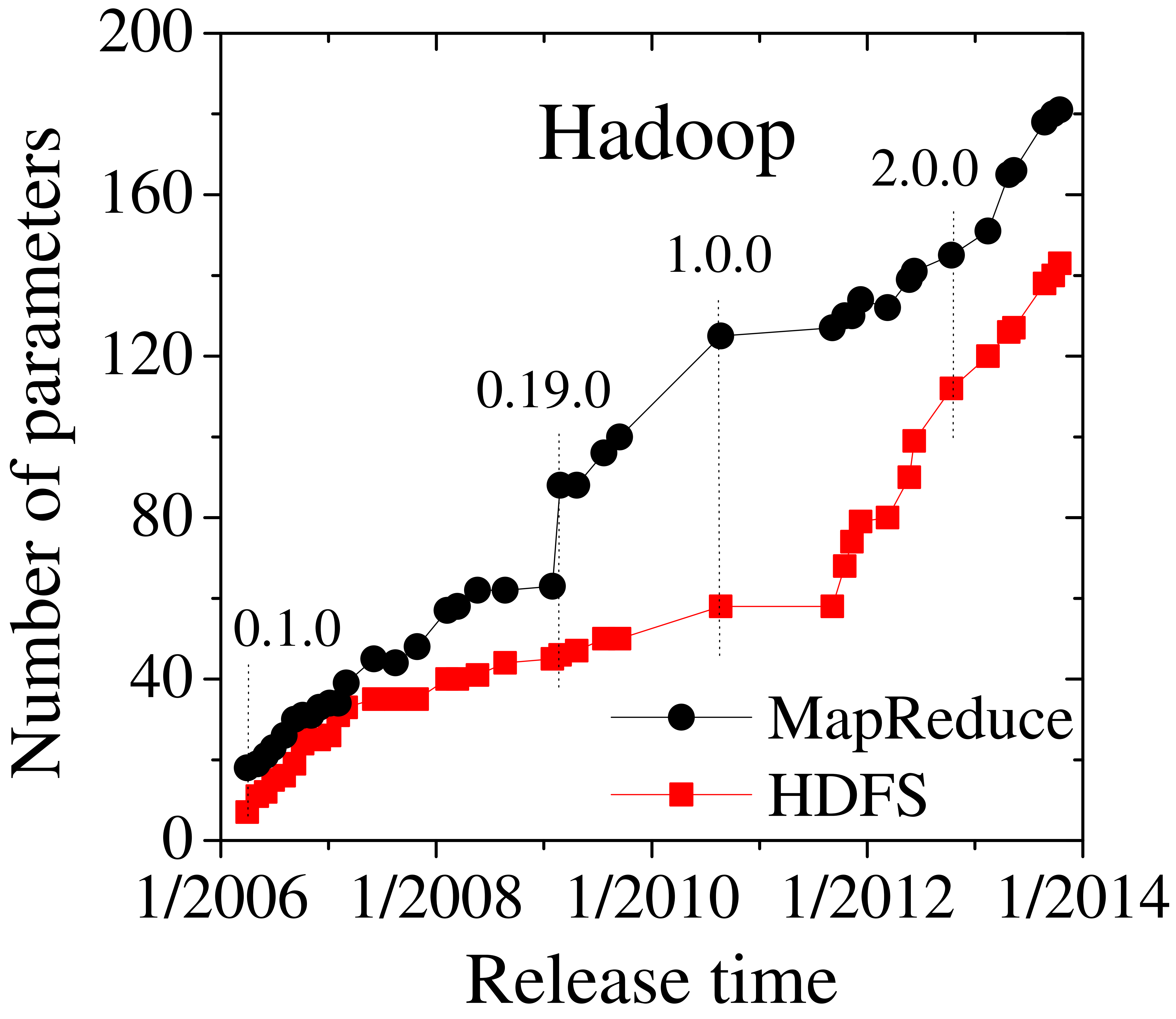
```yaml
drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "JKS"
drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
drpc.authorizer.acl.strict: false

transactional.zookeeper.root: "/transactional"
transactional.zookeeper.servers: null
transactional.zookeeper.port: null

## blobstore configs
supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
supervisor.blobstore.download.thread.count: 5
supervisor.blobstore.download.max_retries: 3
supervisor.localizer.cache.target.size.mb: 10240
supervisor.localizer.cleanup.interval.ms: 600000
```

```yaml

drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "JKS"
drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
drpc.authorizer.acl.strict: false

transactional.zookeeper.root: "/transactional"
transactional.zookeeper.servers: null
transactional.zookeeper.port: null

## blobstore configs
supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
supervisor.blobstore.download.thread.count: 5
supervisor.blobstore.download.max_retries: 3
supervisor.localizer.cache.target.size.mb: 10240
supervisor.localizer.cleanup.interval.ms: 600000
```

```yaml

drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "JKS"
drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
drpc.authorizer.acl.strict: false

transactional.zookeeper.root: "/transactional"
transactional.zookeeper.servers: null
transactional.zookeeper.port: null

## blobstore configs
supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
supervisor.blobstore.download.thread.count: 5
supervisor.blobstore.download.max_retries: 3
supervisor.localizer.cache.target.size.mb: 10240
supervisor.localizer.cleanup.interval.ms: 600000
```

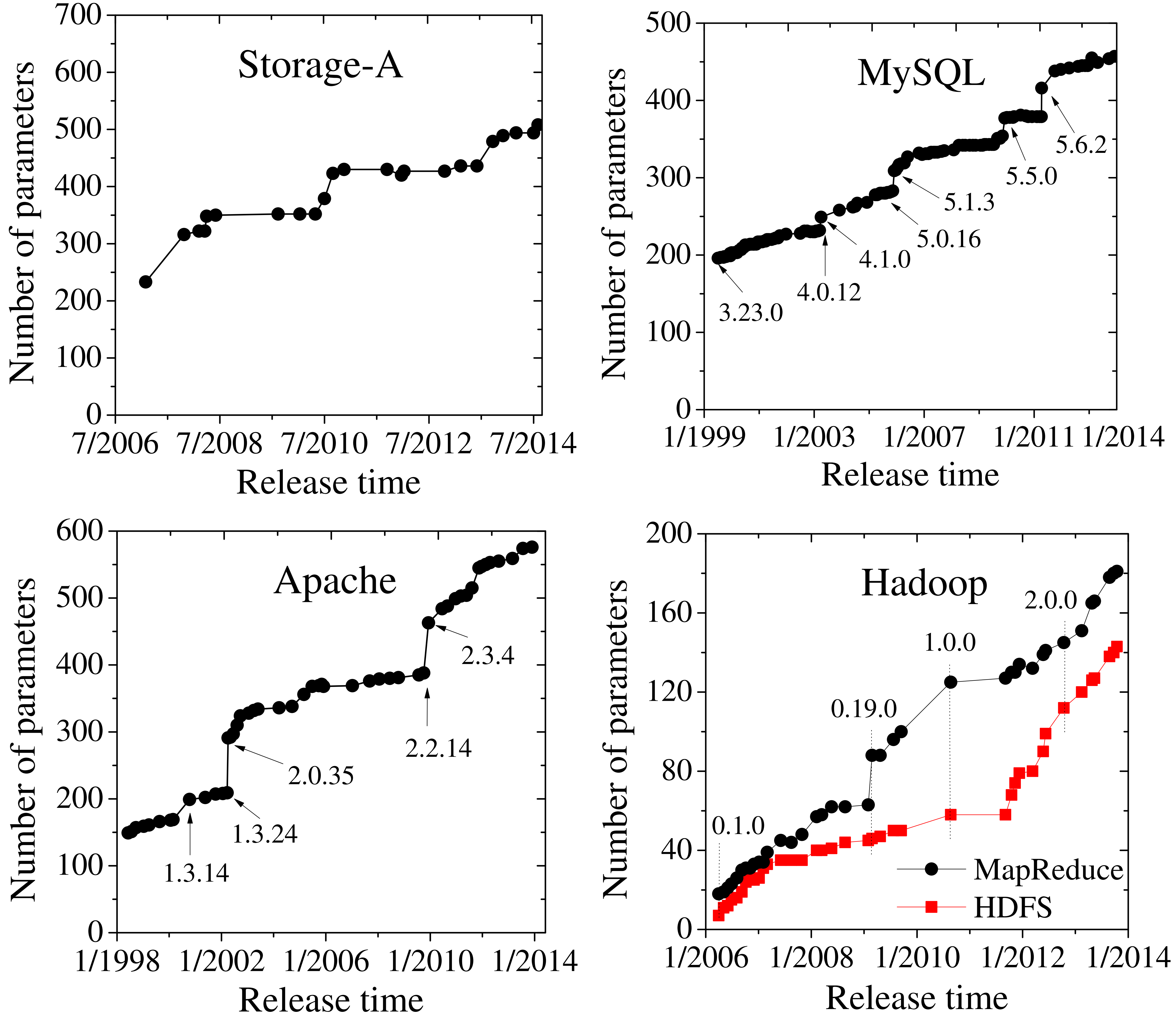
```
drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "JKS"
drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
drpc.authorizer.acl.strict: false

transactional.zookeeper.root: "/transactional"
transactional.zookeeper.servers: null
transactional.zookeeper.port: null

## blobstore configs
supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
supervisor.blobstore.download.thread.count: 5
supervisor.blobstore.download.max_retries: 3
supervisor.localizer.cache.target.size.mb: 10240
supervisor.localizer.cleanup.interval.ms: 600000
```

Empirical observations confirm that systems are becoming increasingly configurable



[Tianyin Xu, et al., "**Too Many Knobs**…", FSE'15]

Empirical observations confirm that systems are becoming increasingly configurable



[Tianyin Xu, et al., “**Too Many Knobs**…”, FSE’15]

Configurations determine the performance behavior

```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```

Configurations determine the performance behavior

```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```

Configurations determine the performance behavior

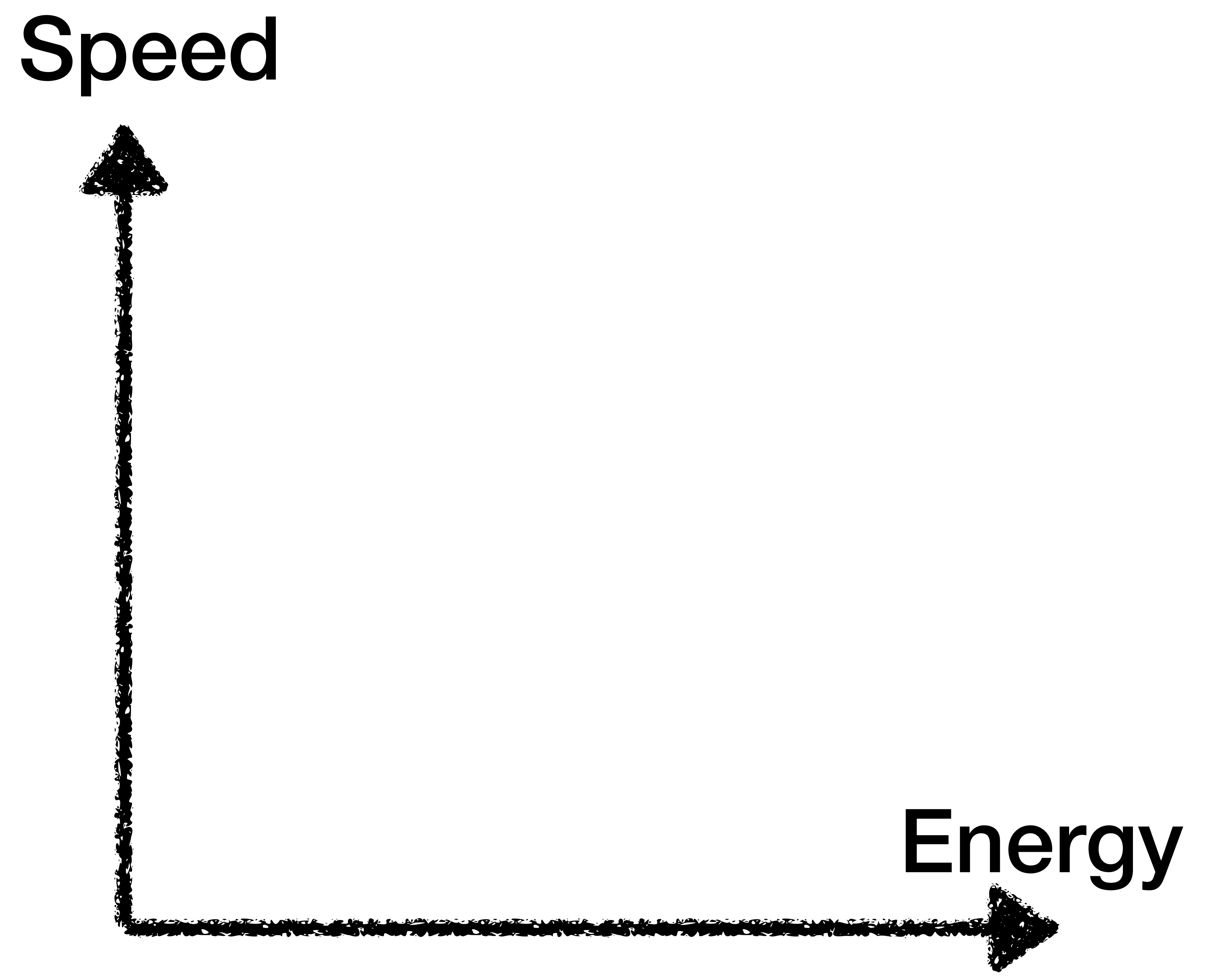
```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```

Configurations determine the performance behavior

```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```

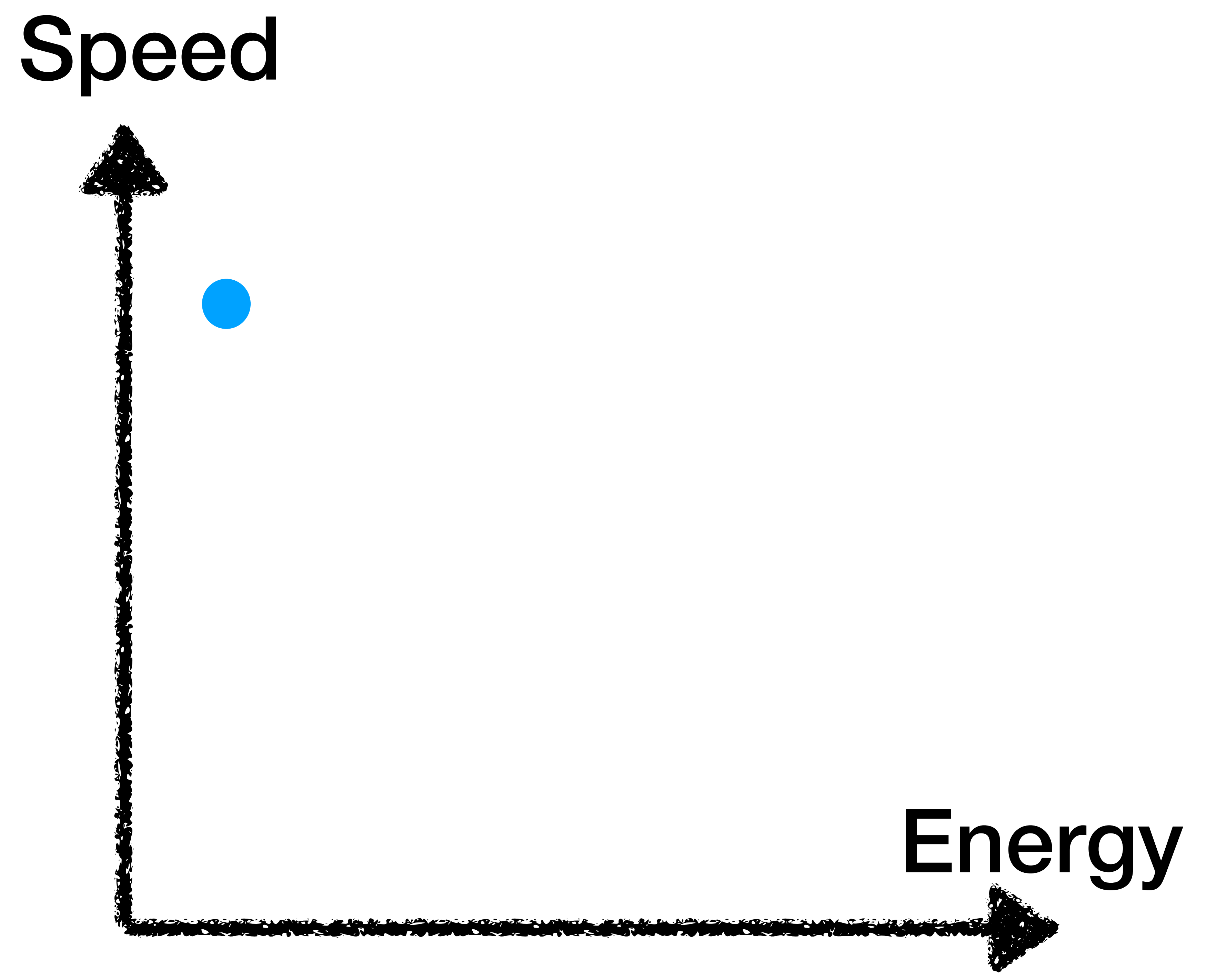
Configurations determine the performance behavior

```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```



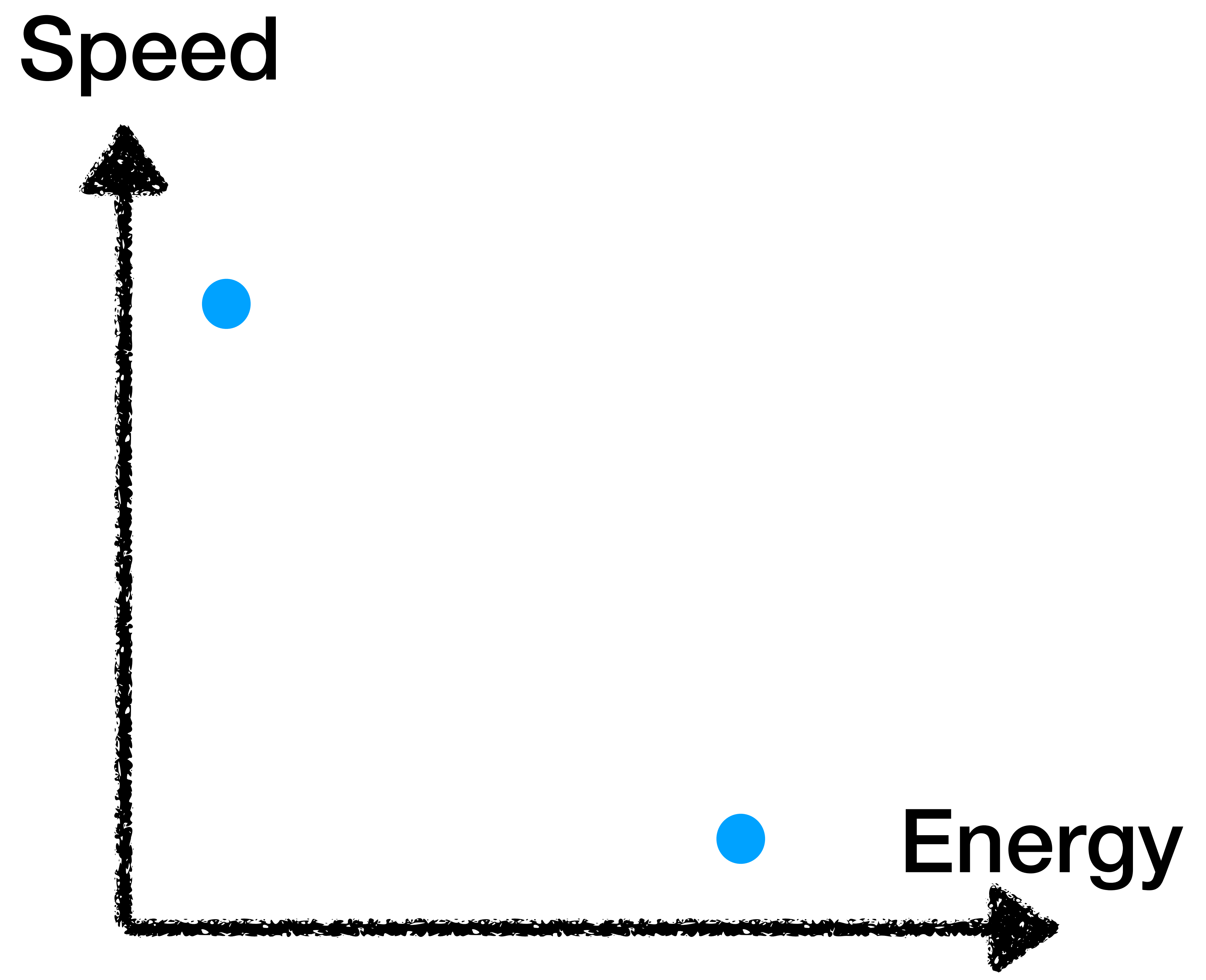
Configurations determine the performance behavior

```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```



Configurations determine the performance behavior

```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```



How do we understand performance behavior of real-world highly-configurable systems that scale well…

How do we understand performance behavior of real-world highly-configurable systems that scale well…

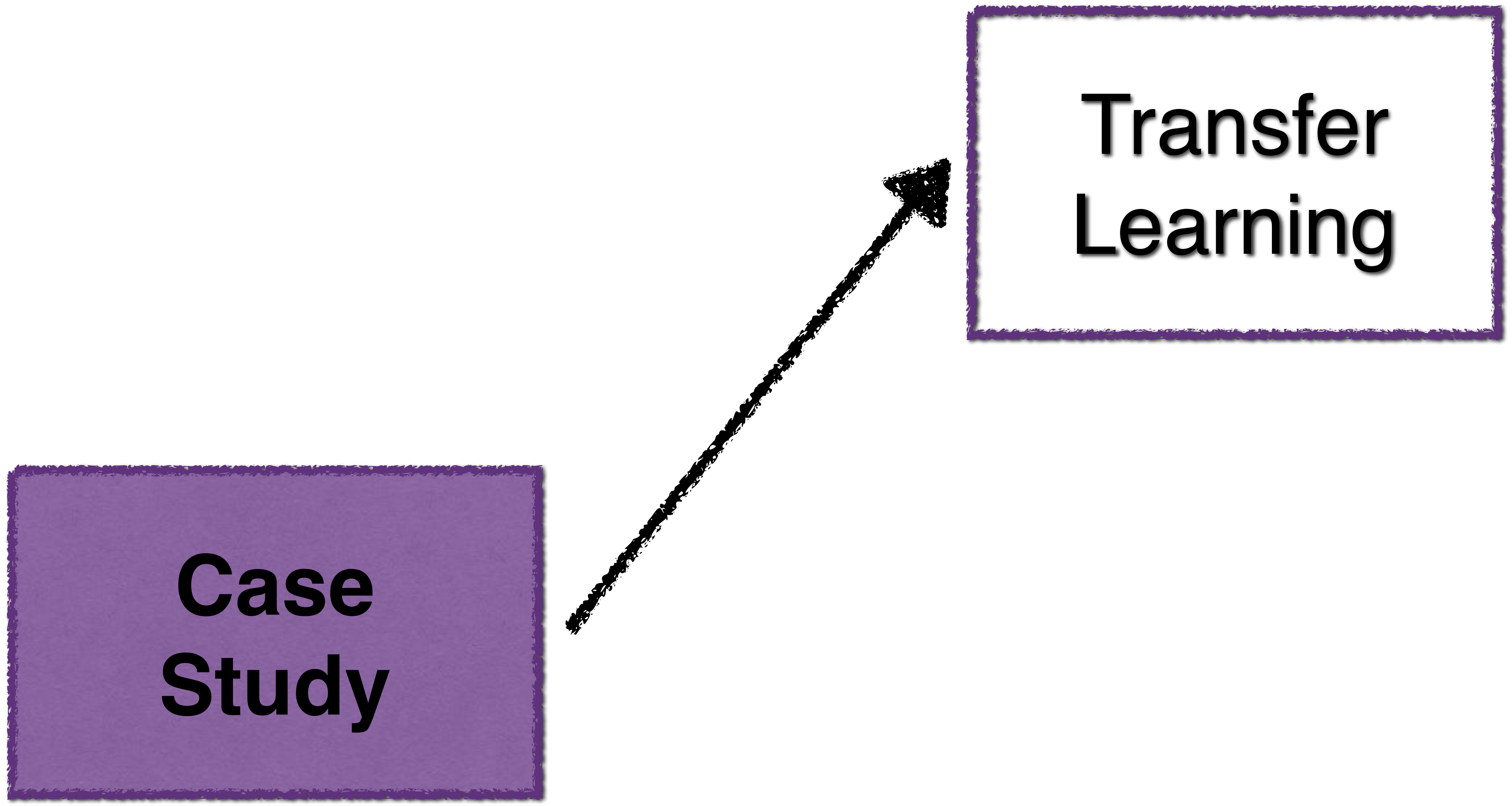
… and enable developers/users to reason about qualities (performance, energy) and to make tradeoff?

Outline

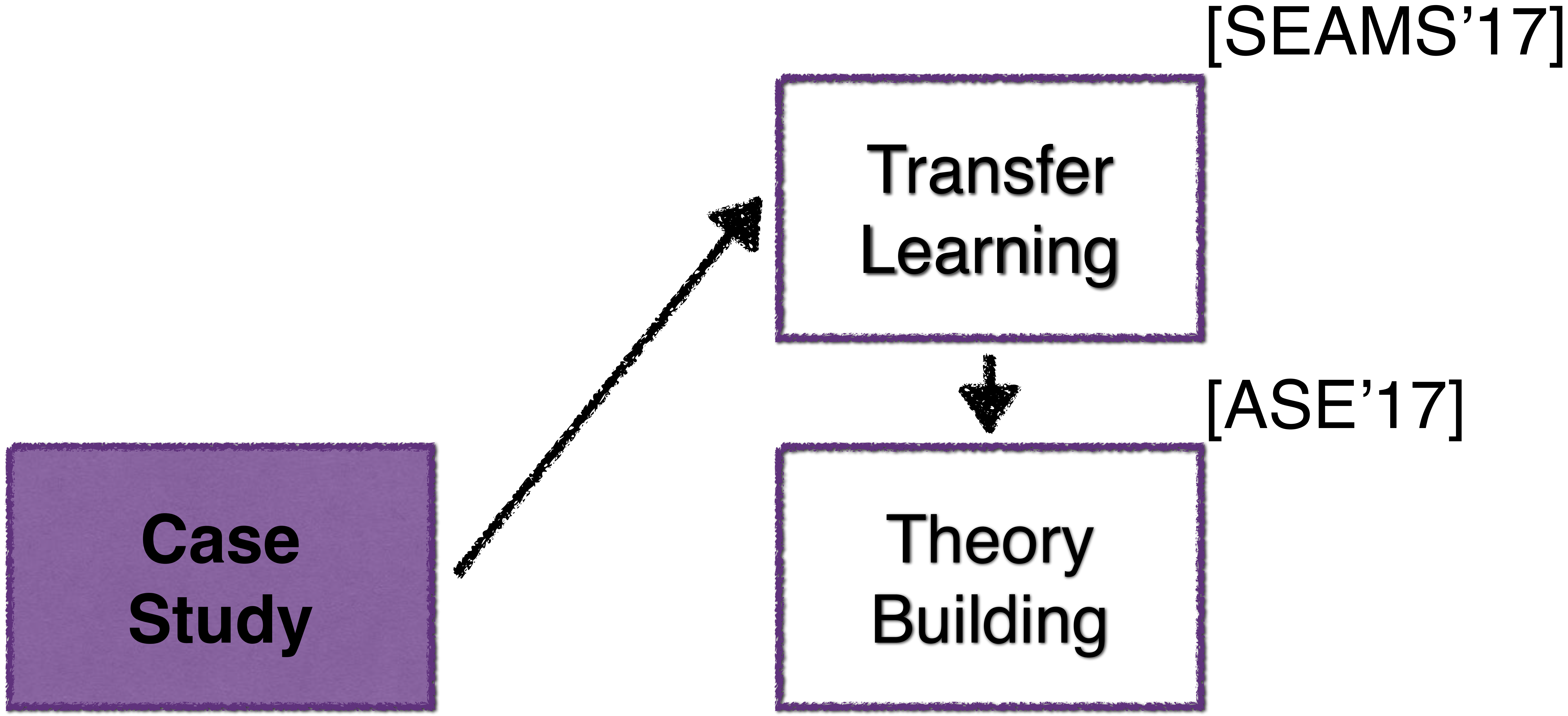
Case Study

Outline

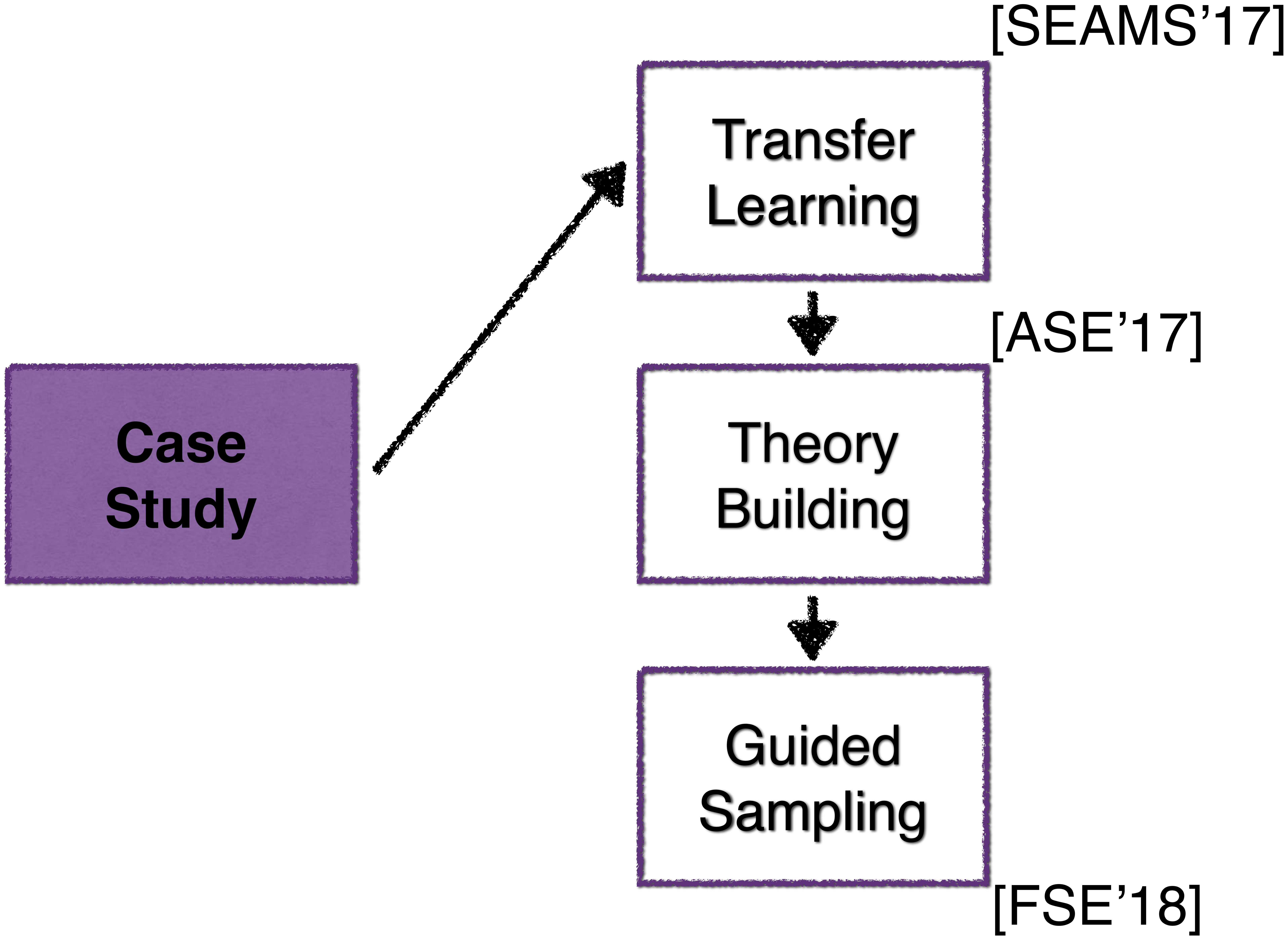
[SEAMS'17]



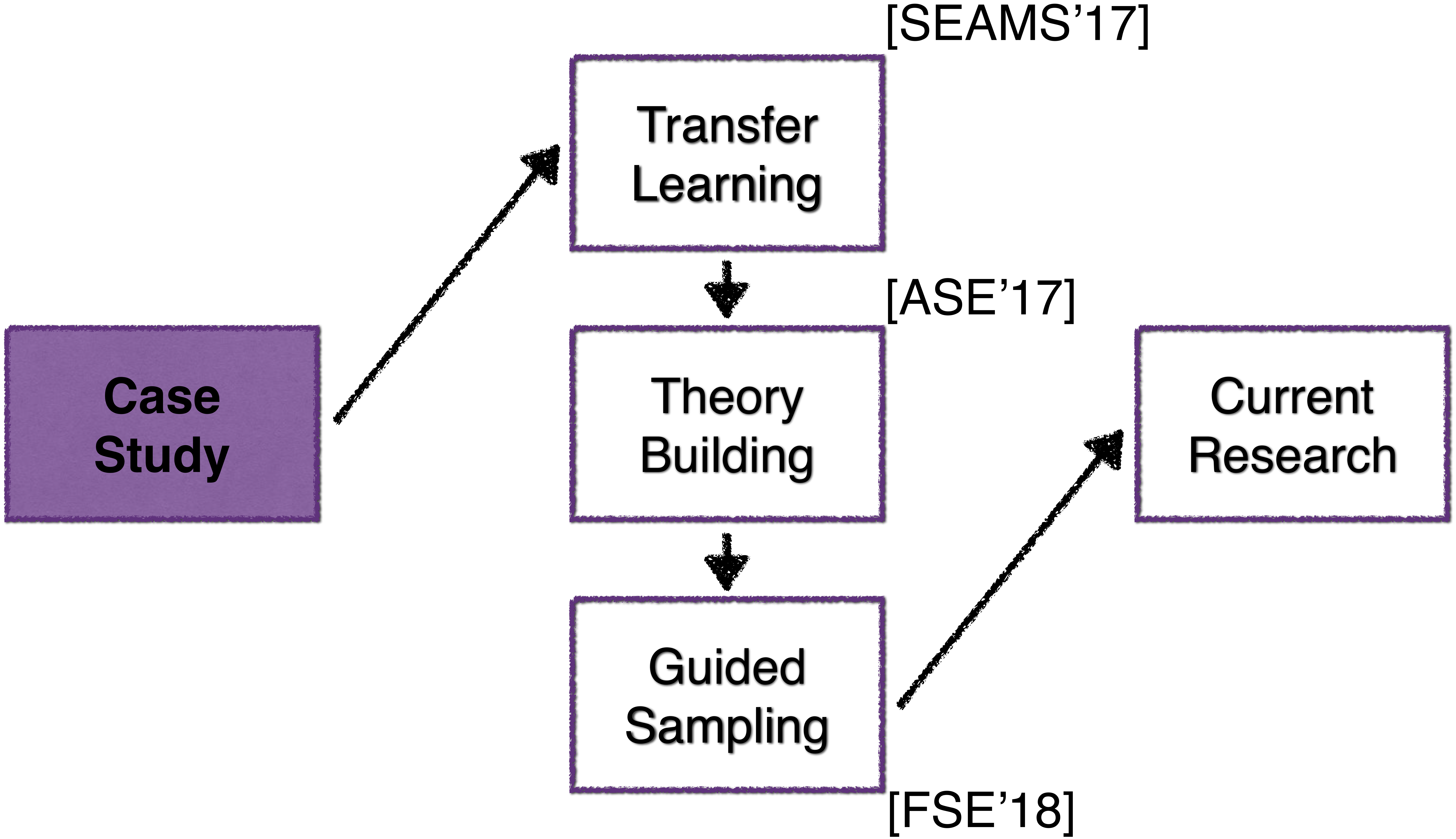
Outline


[SEAMS’17]
Transfer Learning
[ASE’17]
Case Study
Theory Building

Outline


[SEAMS'17]
Transfer Learning
[ASE'17]
Case Study
Theory Building
Guided Sampling
[FSE'18]

Outline


[SEAMS'17]
Transfer Learning
[ASE'17]
Case Study
Theory Building
Current Research
Guided Sampling
[FSE'18]

SocialSensor

- Identifying trending topics
- Identifying user defined topics
- Social media search

SocialSensor

Internet

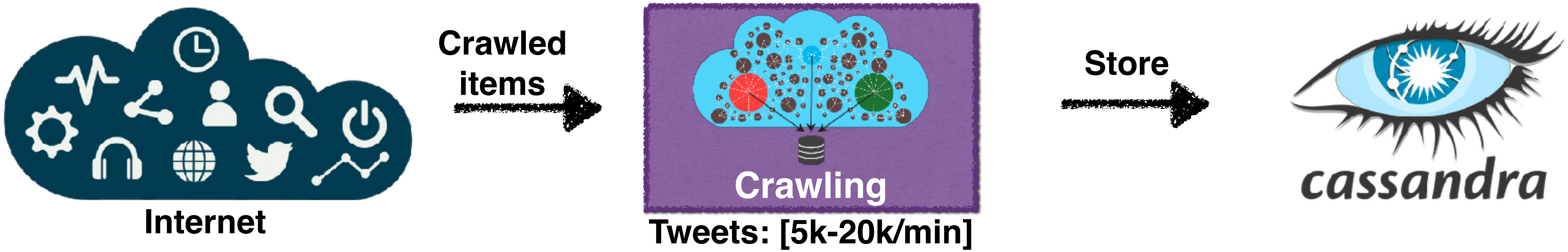
SocialSensor

Internet

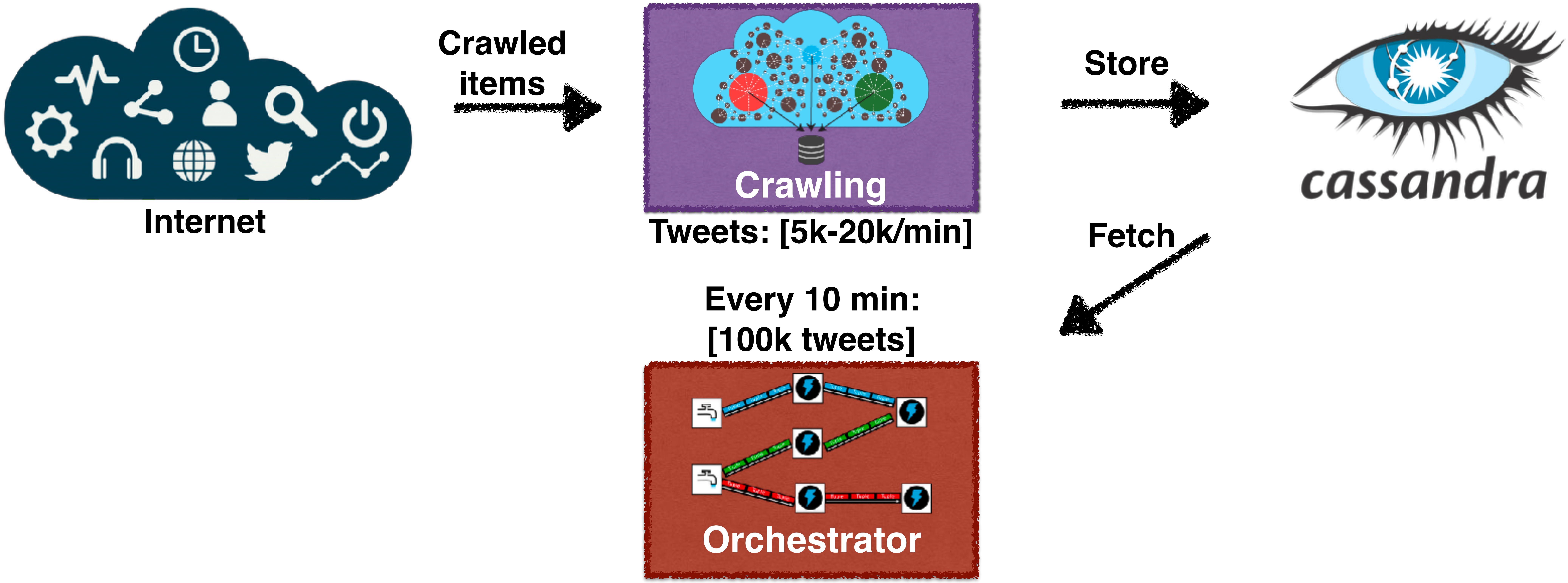
Crawled
items

Tweets: [5k-20k/min]

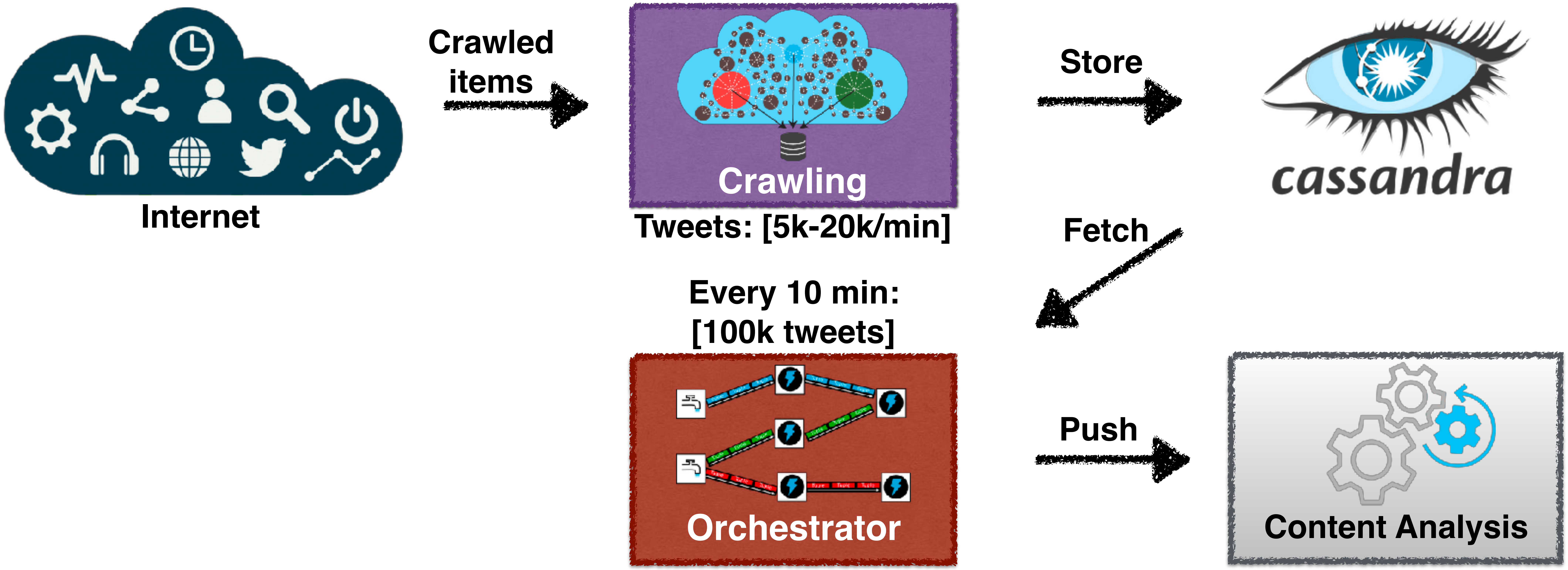
SocialSensor



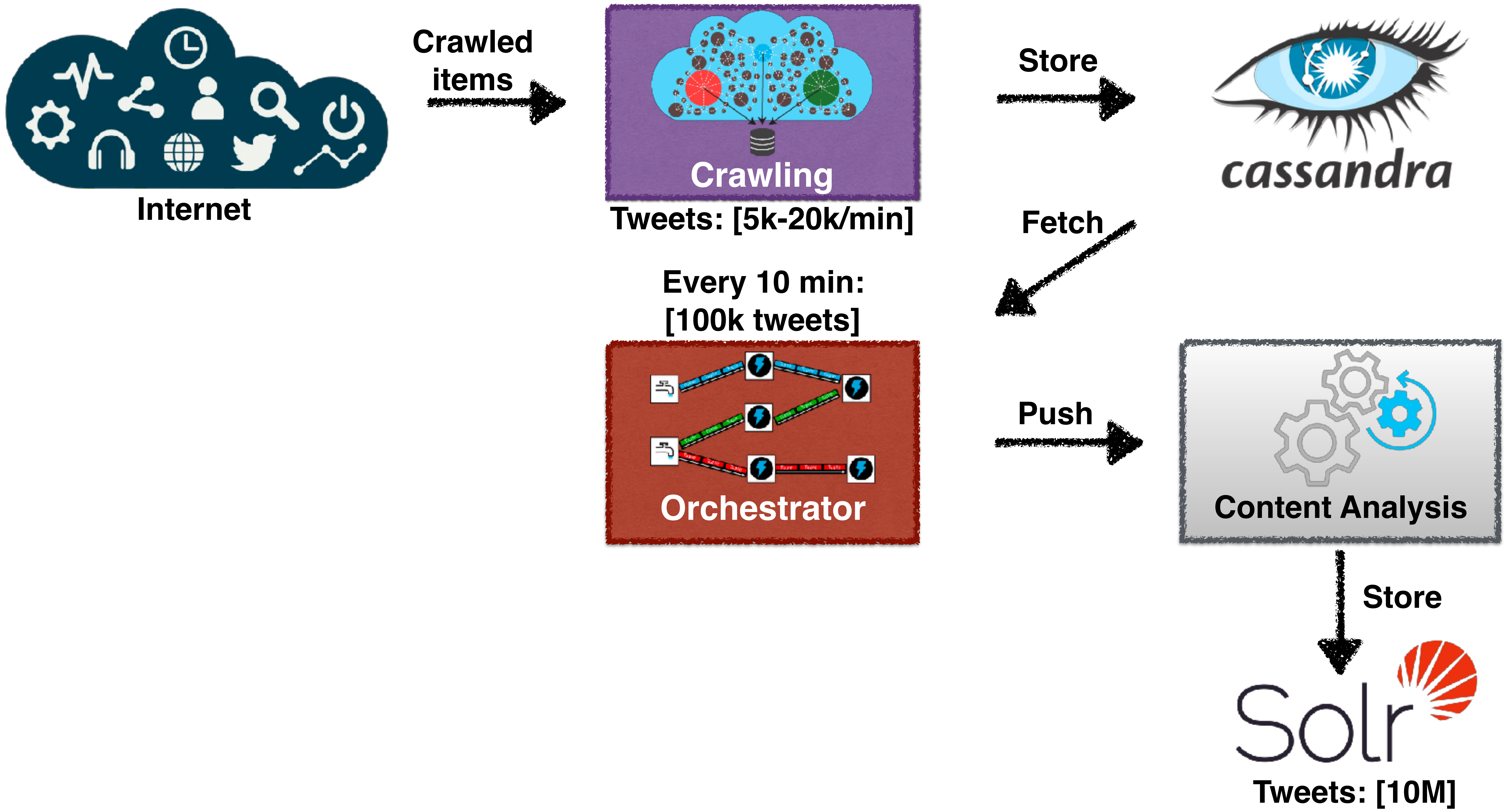
SocialSensor



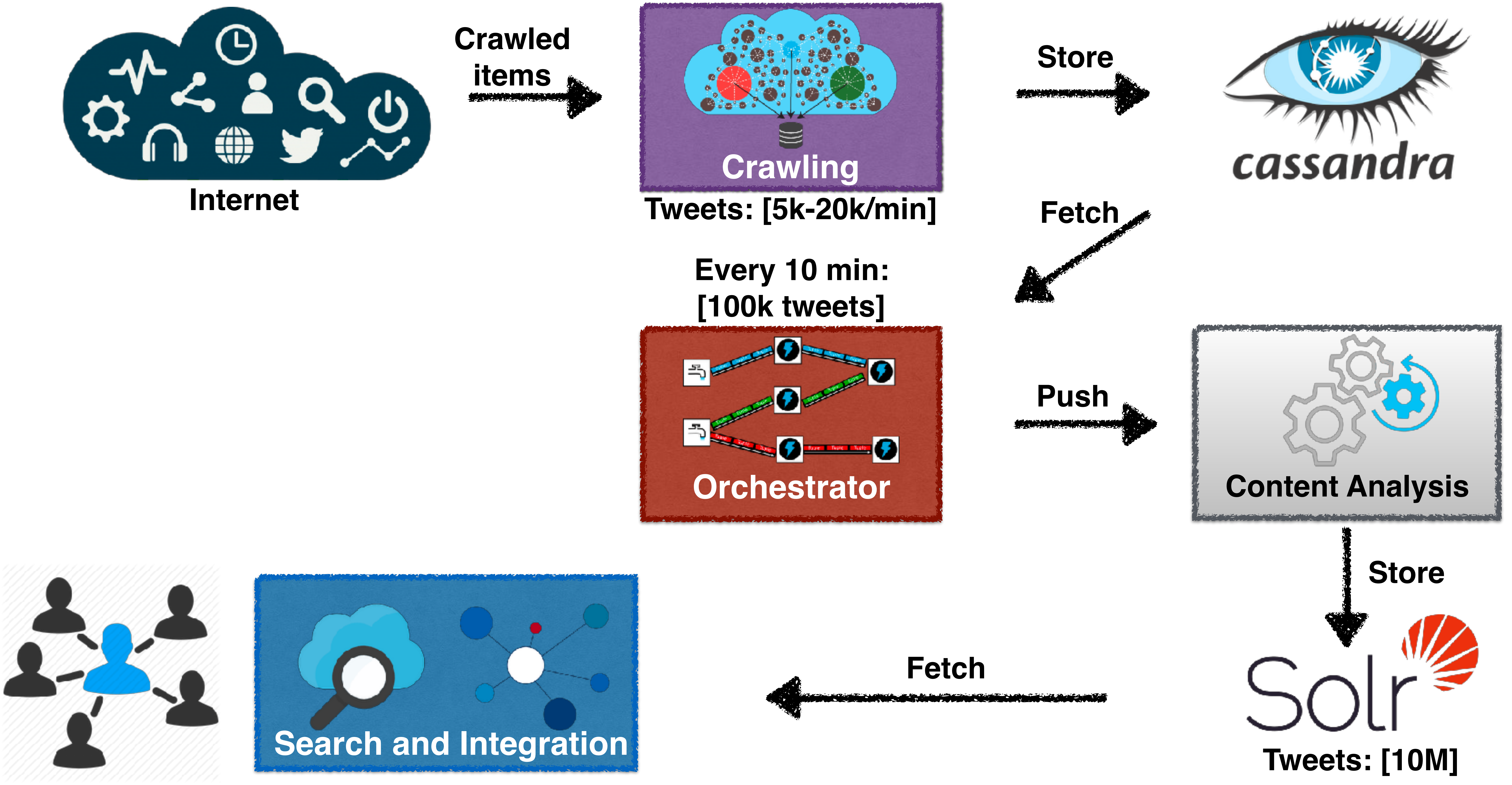
SocialSensor



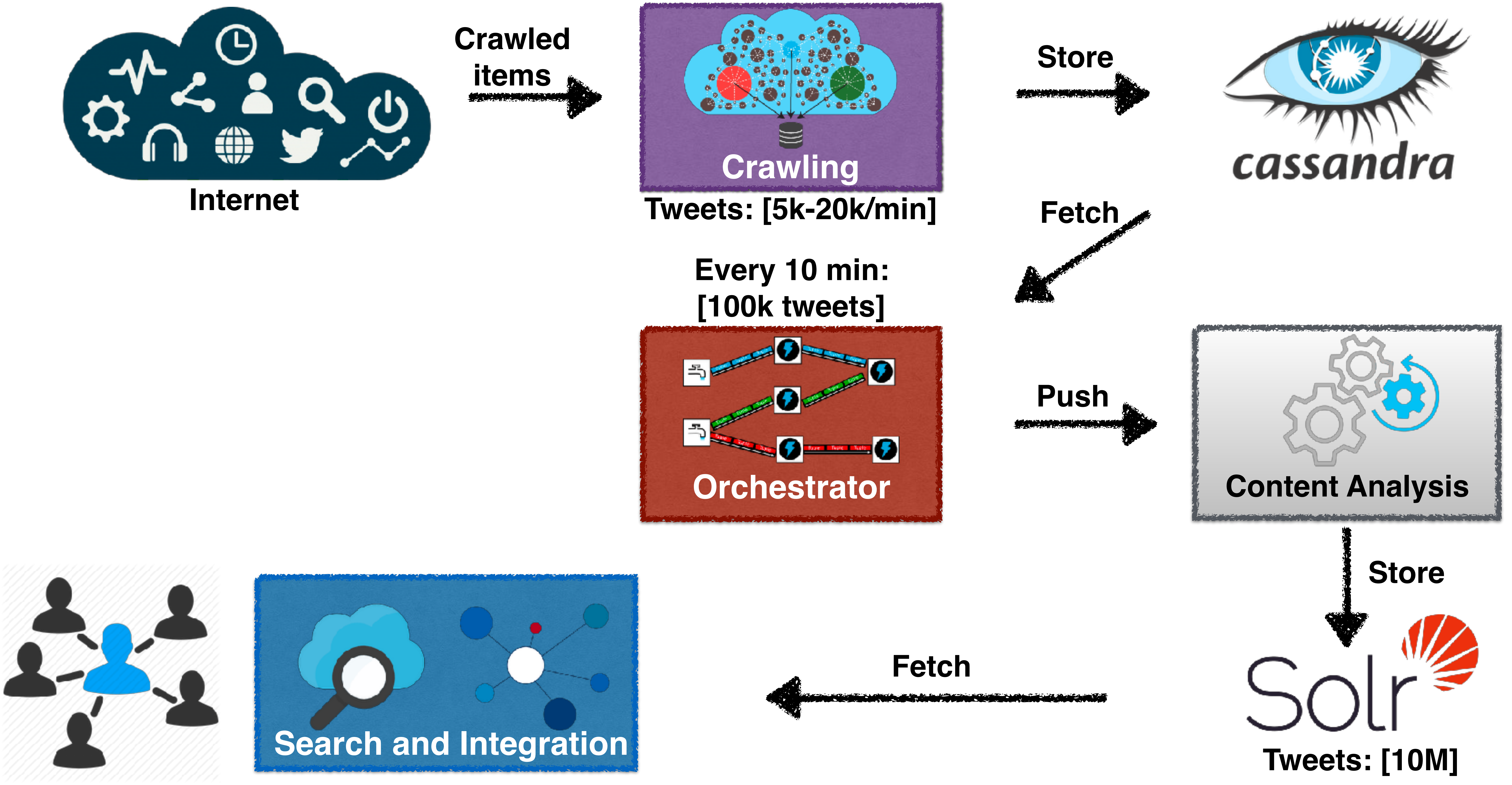
SocialSensor



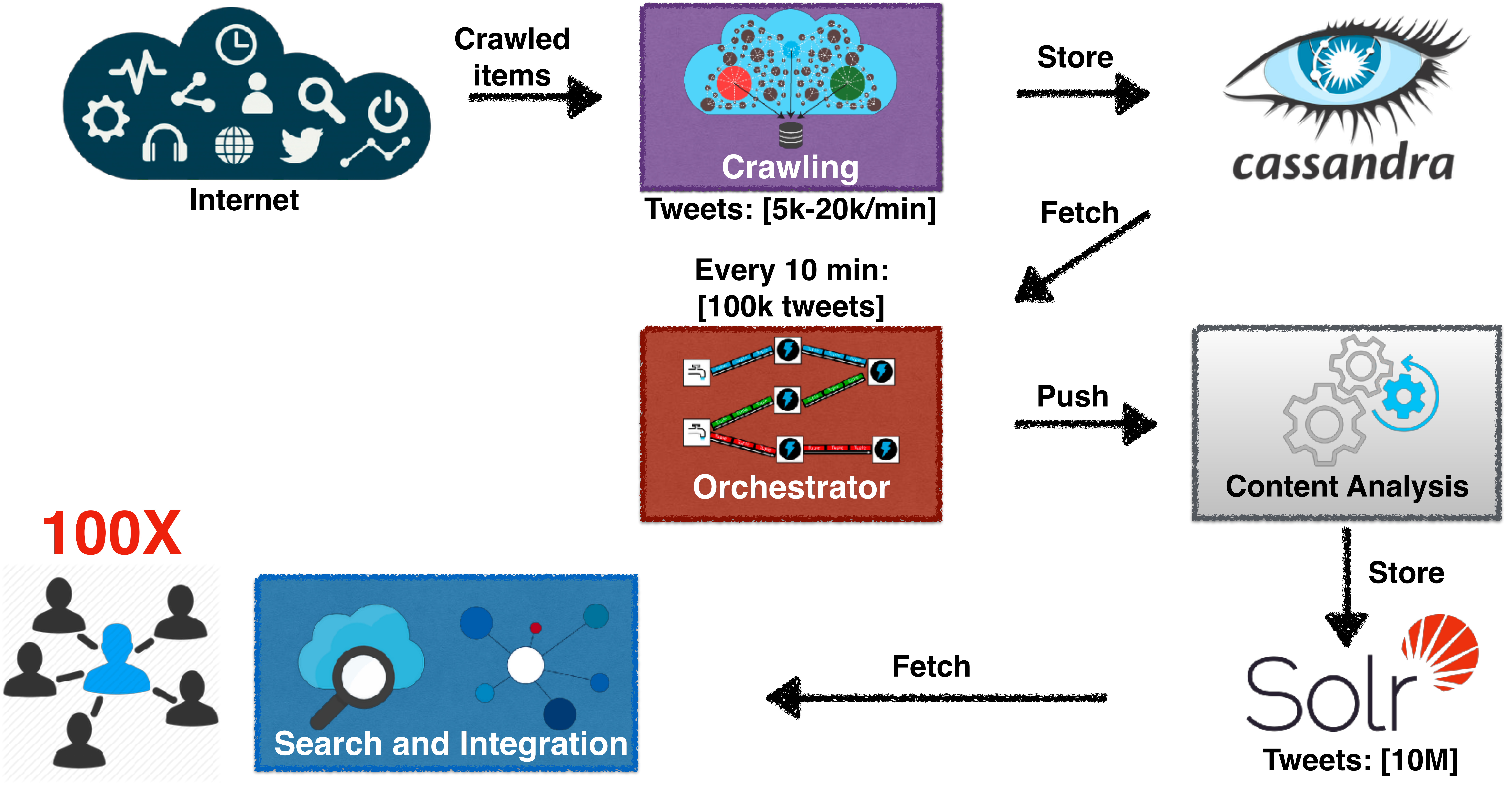
SocialSensor



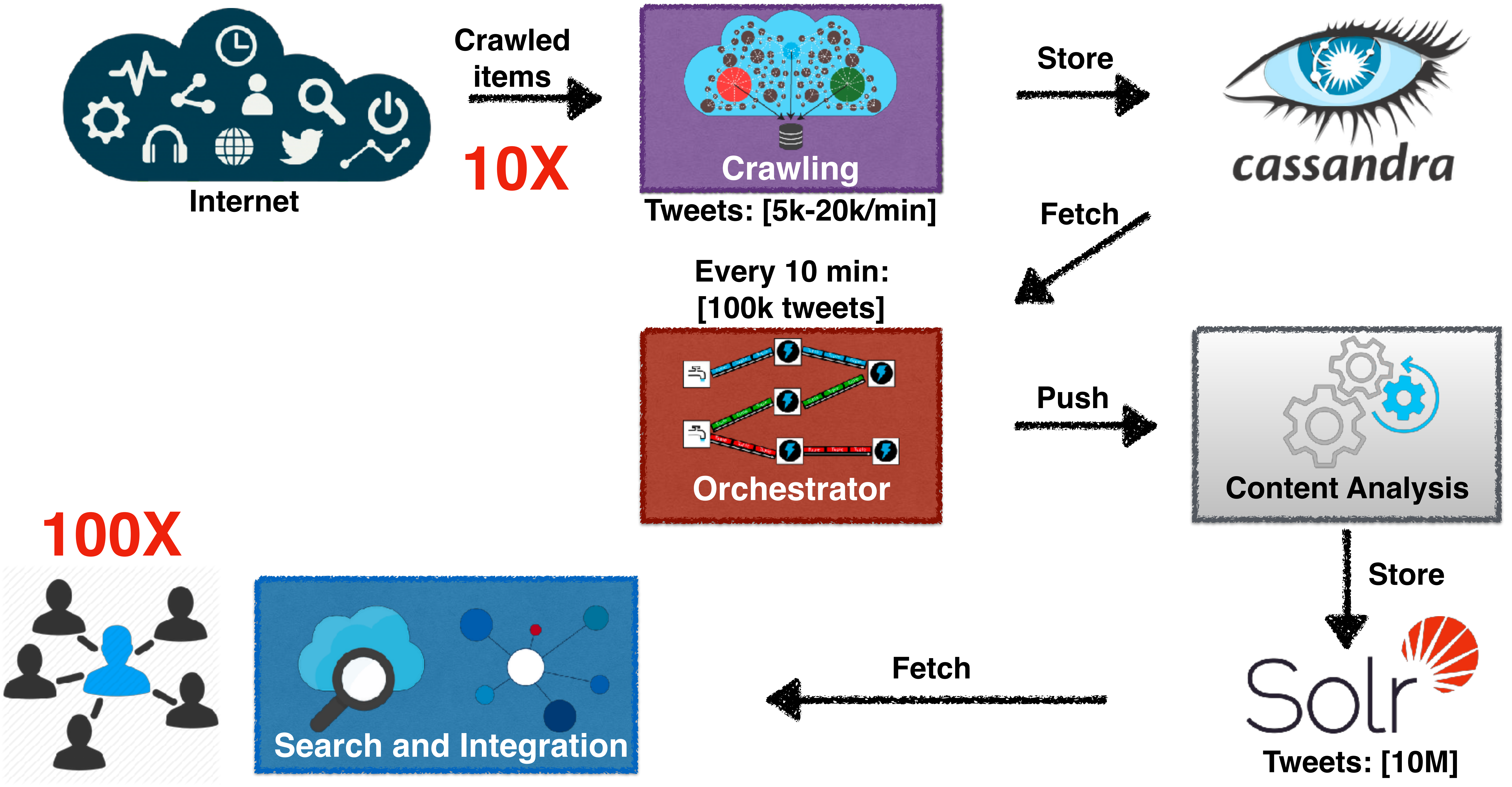
Challenges


Internet
Crawled items
Crawling
Tweets: [5k-20k/min]
Store
cassandra
Fetch
Every 10 min: [100k tweets]
Orchestrator
Push
Content Analysis
Store
Solr
Tweets: [10M]
Fetch
Search and Integration

Challenges

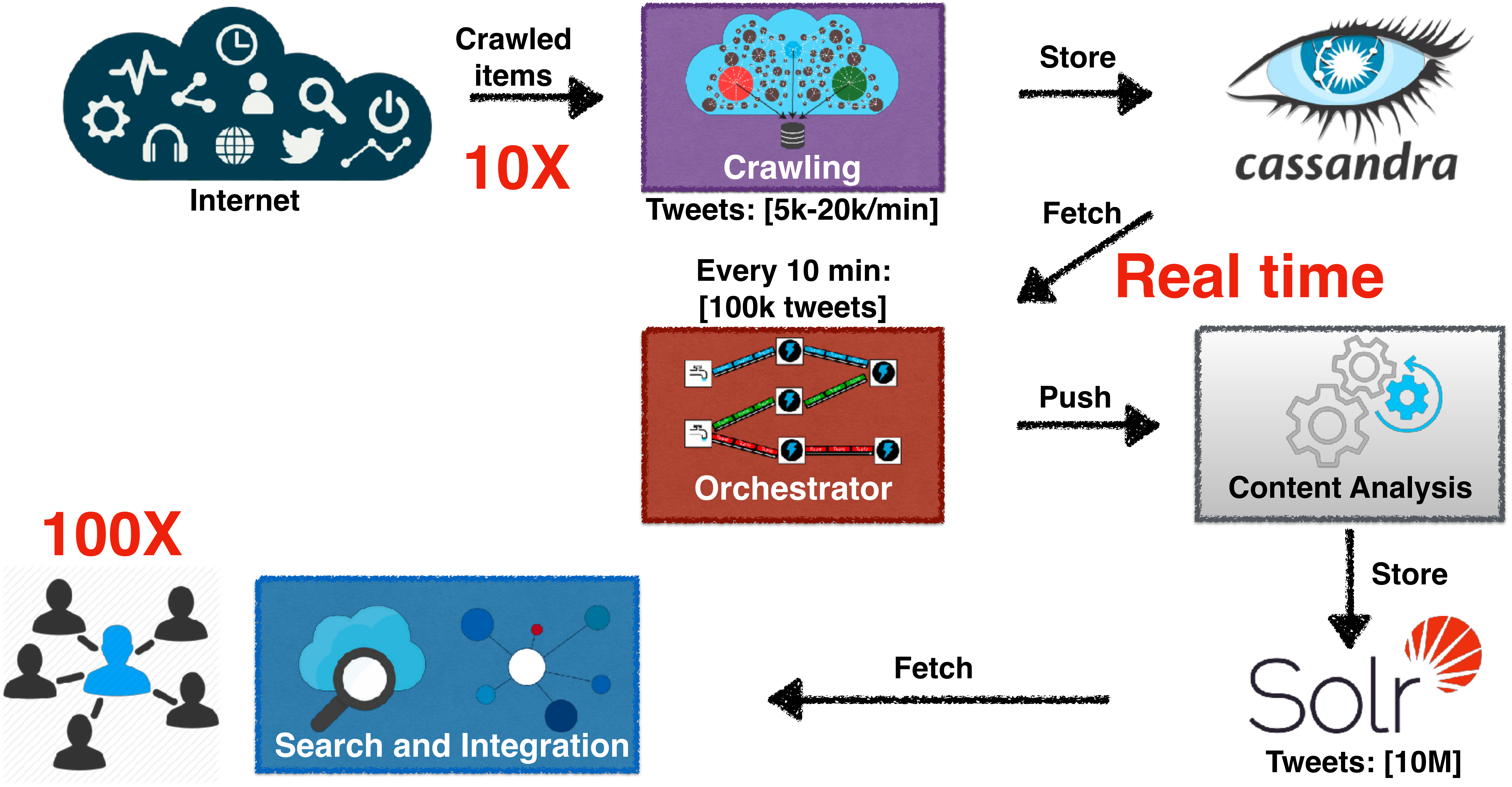

Crawled items
Crawling
Tweets: [5k-20k/min]
Store
cassandra
Fetch
Every 10 min: [100k tweets]
Orchestrator
Push
Content Analysis
Store
Solr
Tweets: [10M]
Fetch
Search and Integration
Internet
100X

Challenges



Crawled items
10X
Crawling
Tweets: [5k-20k/min]
Store
cassandra
Internet
Fetch
Every 10 min:
[100k tweets]
Orchestrator
Push
Content Analysis
Store
100X
Search and Integration
Fetch
Solr
Tweets: [10M]

Challenges



Internet
Crawled
items
10X
Crawling
Tweets: [5k-20k/min]
Store
cassandra
Fetch
Real time
Every 10 min:
[100k tweets]
Orchestrator
Push
Content Analysis
Store
Solr
Tweets: [10M]
Fetch
100X
Search and Integration

How can we gain a better performance without using more resources?

Let’s try out different system configurations!

Opportunity: Data processing engines in the pipeline were all configurable

Opportunity: Data processing engines in the pipeline were all configurable

Opportunity: Data processing engines in the pipeline were all configurable

cassandra

STORM

Opportunity: Data processing engines in the pipeline were all configurable

cassandra

STORM

Solr

Opportunity: Data processing engines in the pipeline were all configurable

cassandra

> 100

STORM

> 100

Solr

> 100

Opportunity: Data processing engines in the pipeline were all configurable

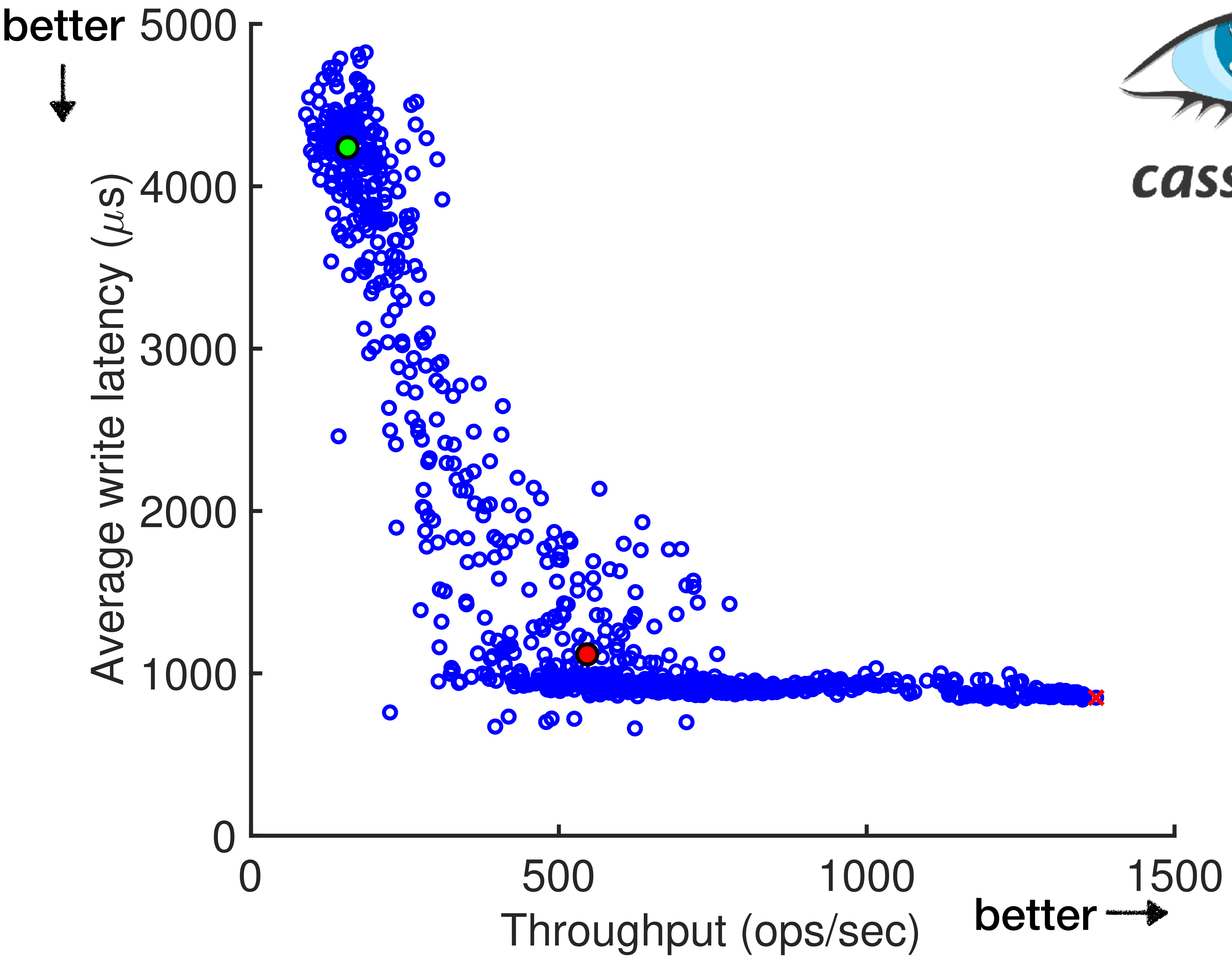
cassandra > 100

STORM > 100

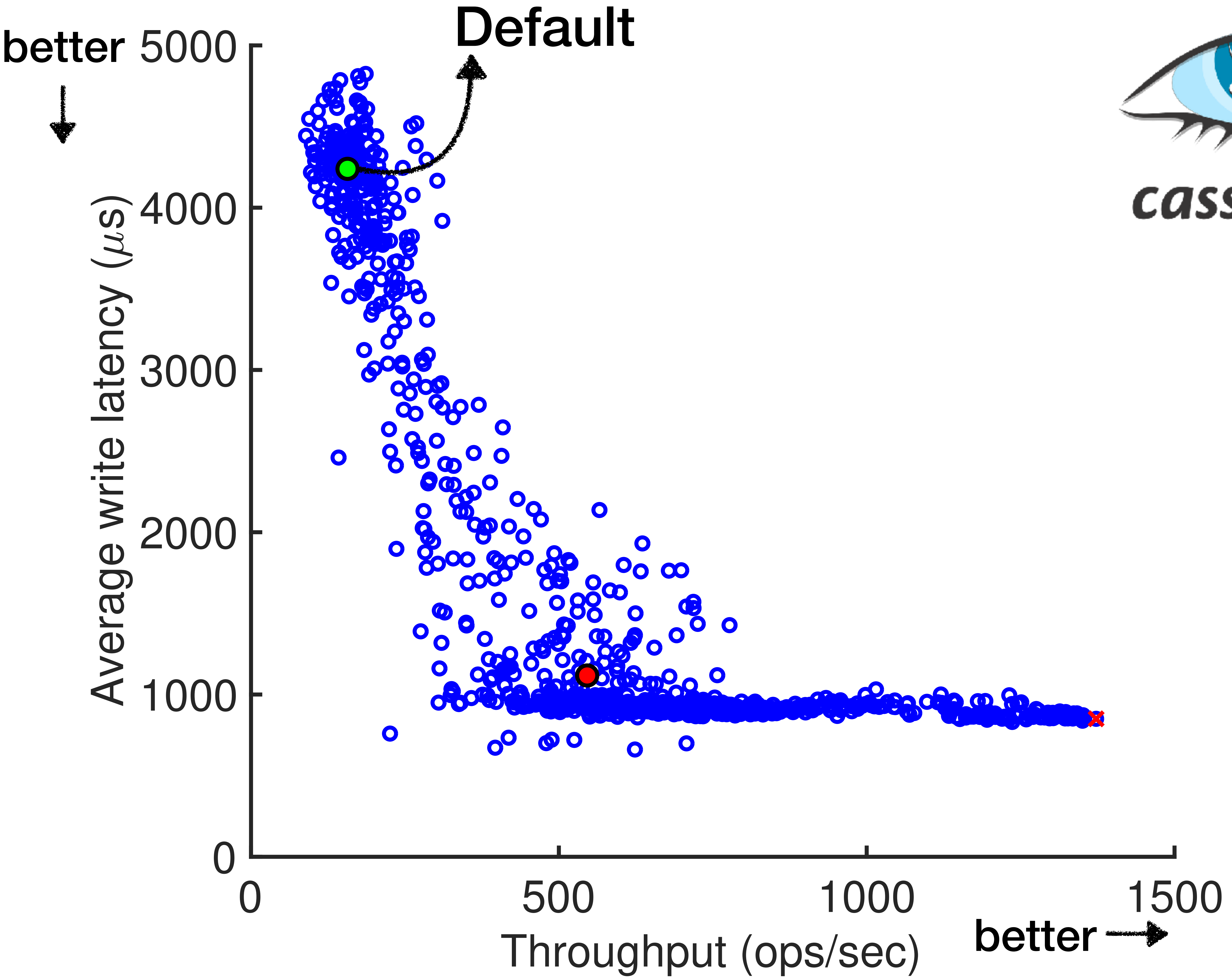
Solr > 100

$$2^{300}$$

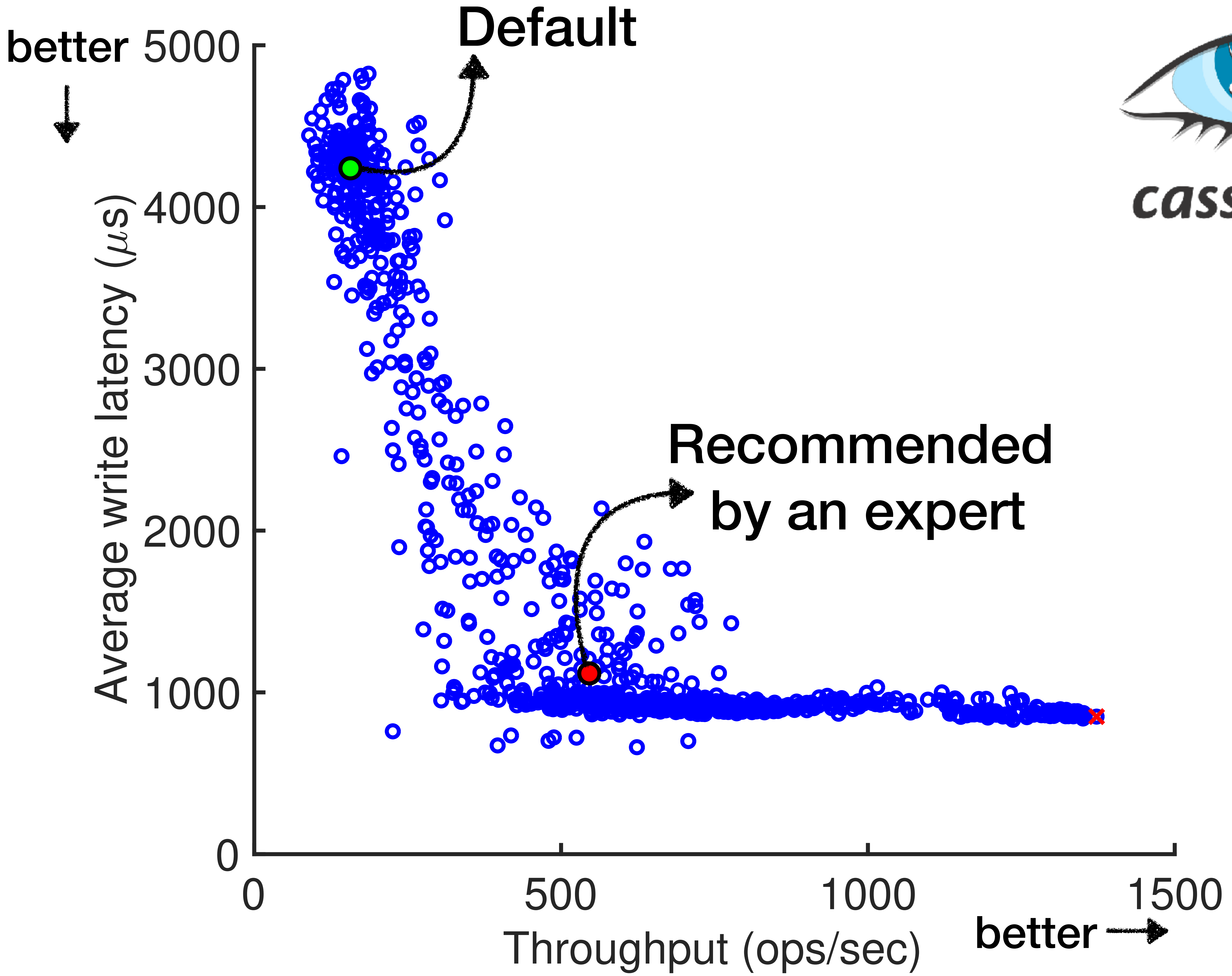
Default configuration was bad, so was the expert'



Default configuration was bad, so was the expert'

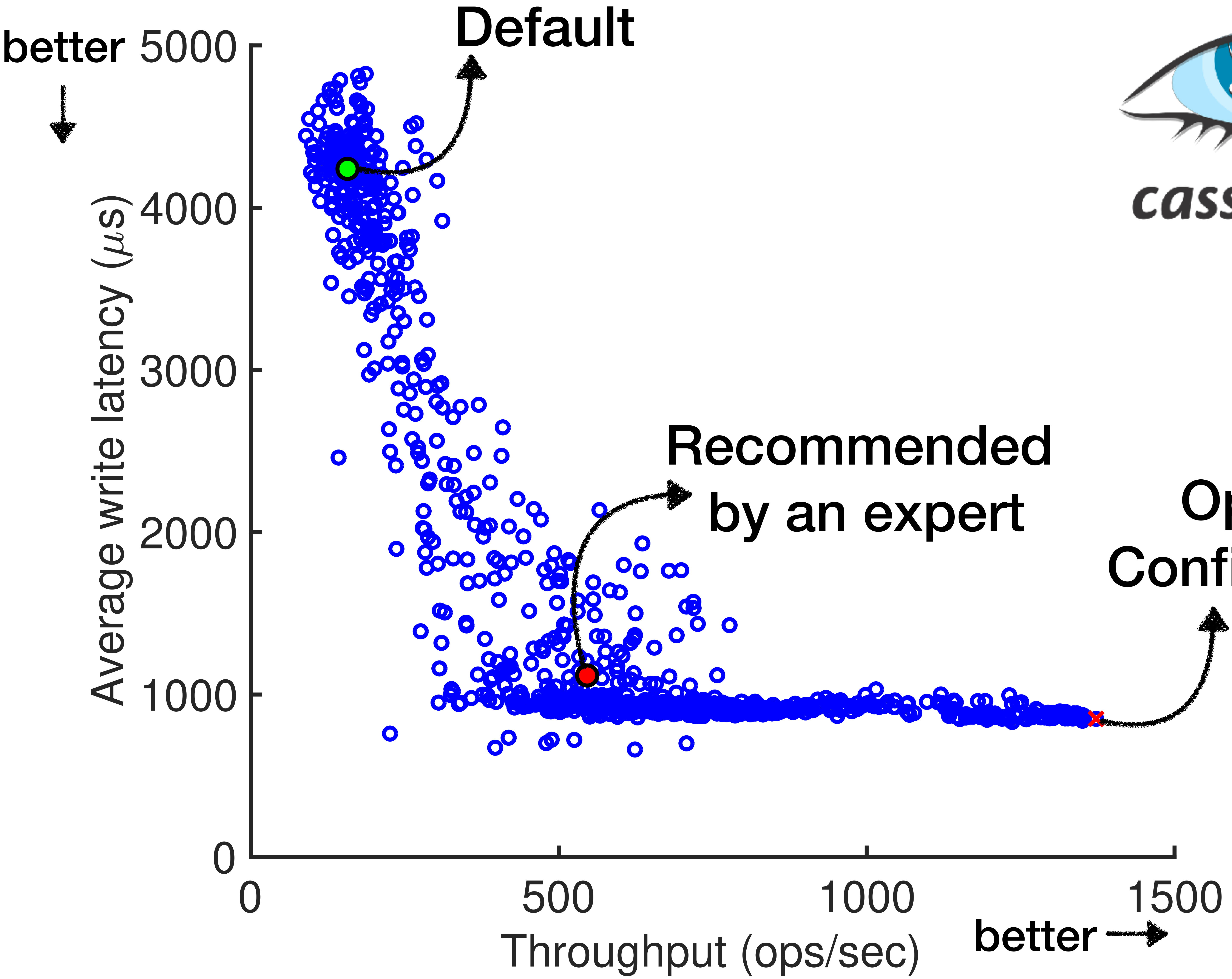


Default configuration was bad, so was the expert’

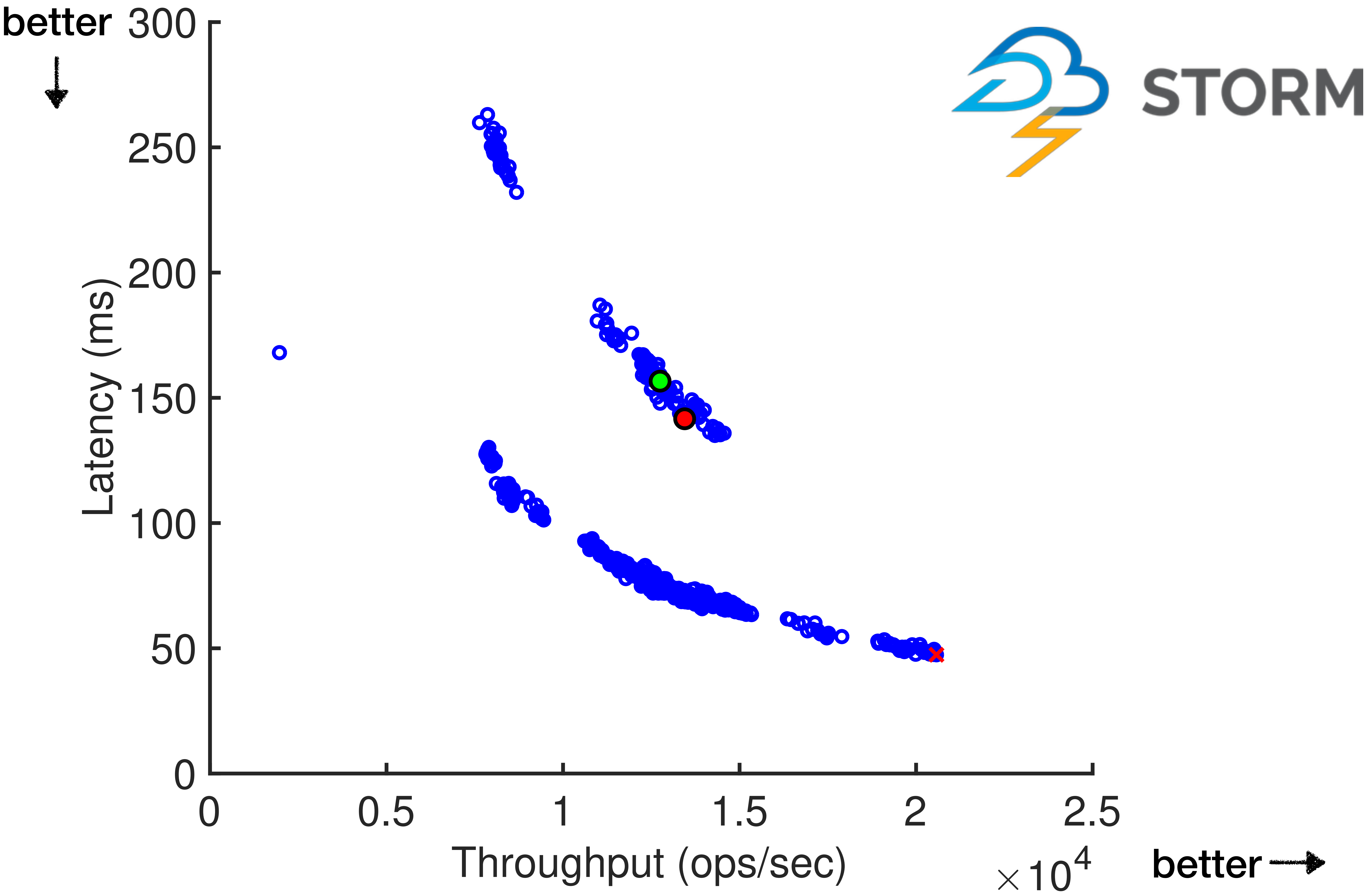


cassandra

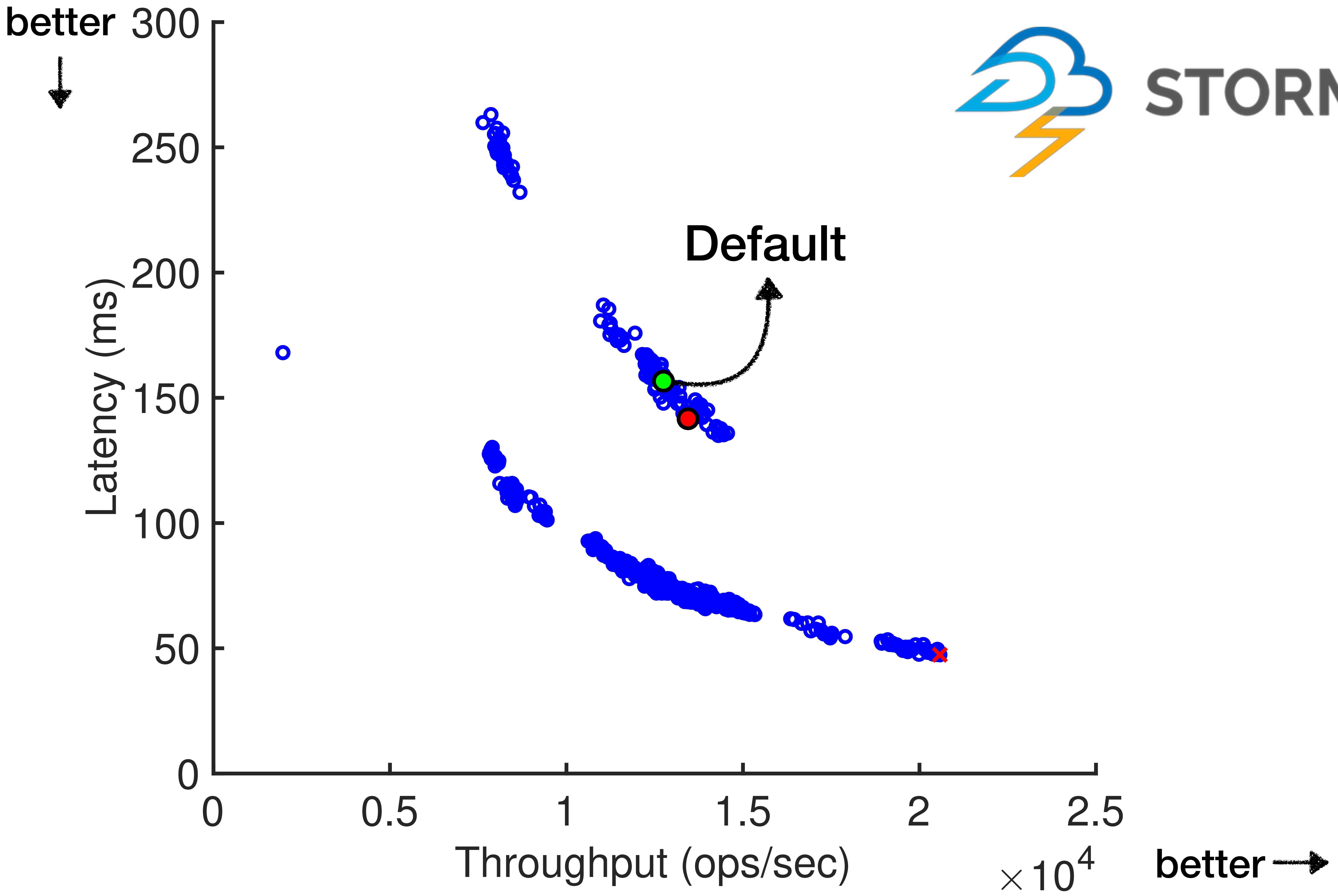
Default configuration was bad, so was the expert'



Default configuration was bad, so was the expert'

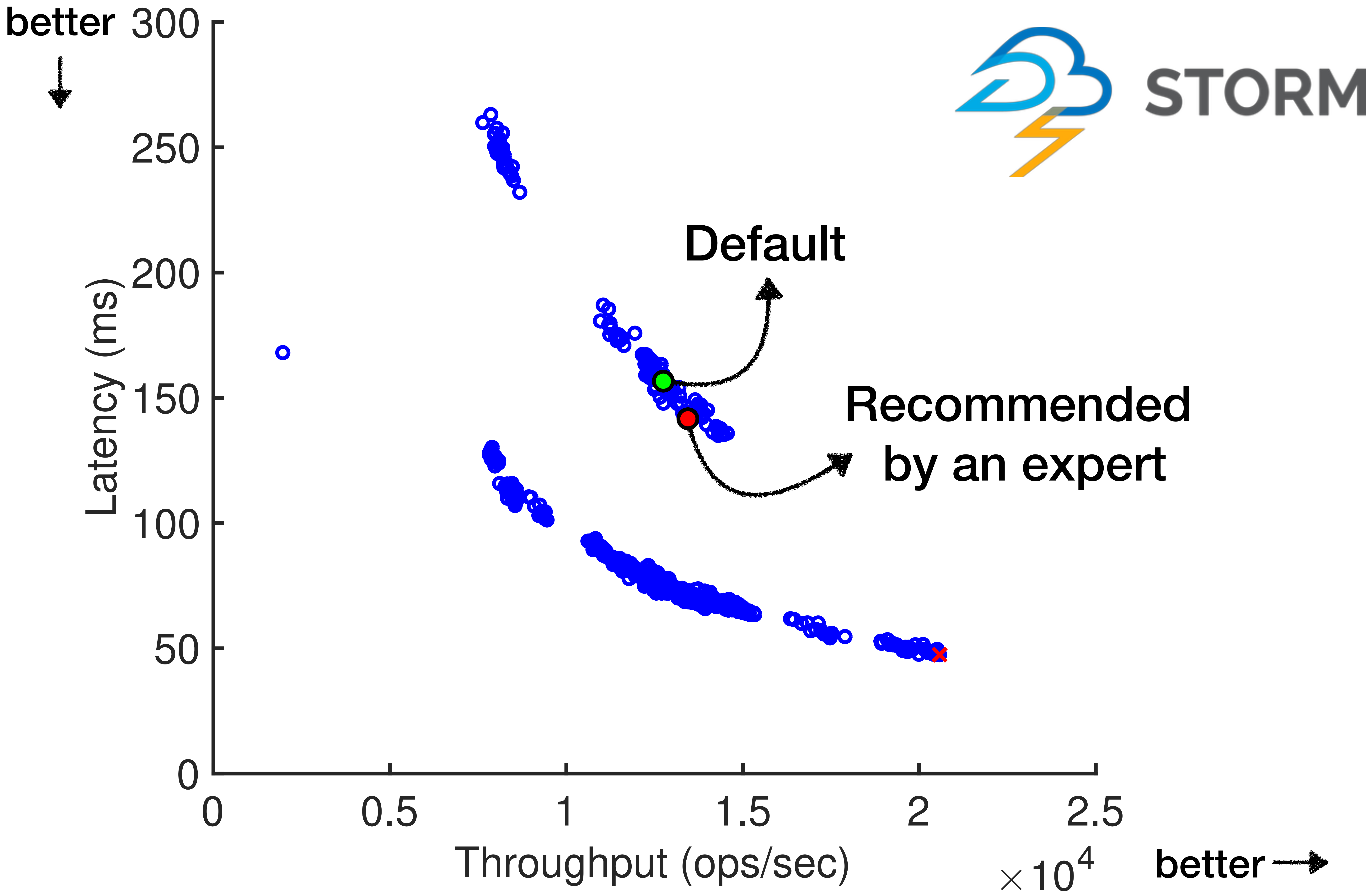


Default configuration was bad, so was the expert'

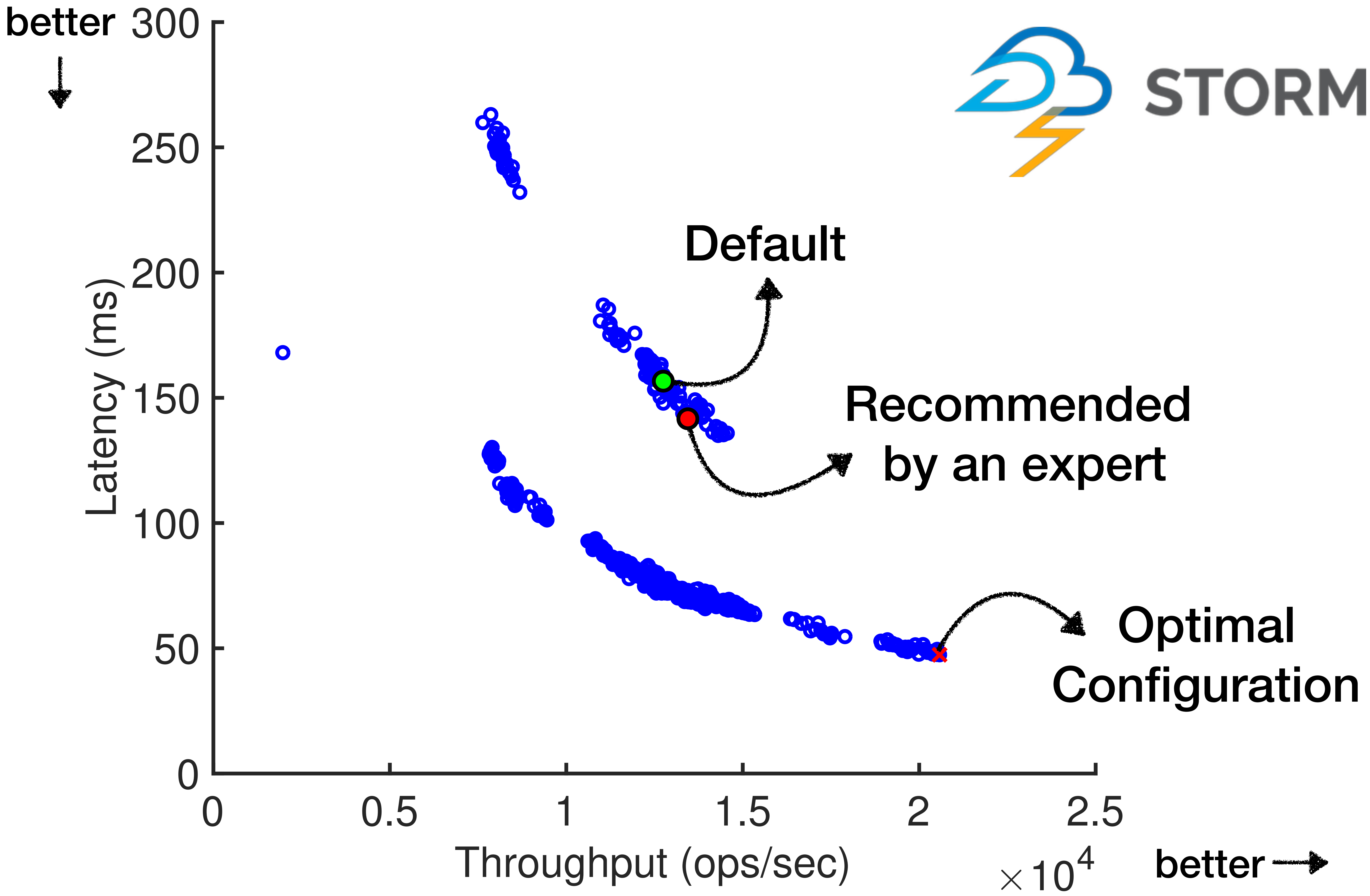


STORM

Default configuration was bad, so was the expert'

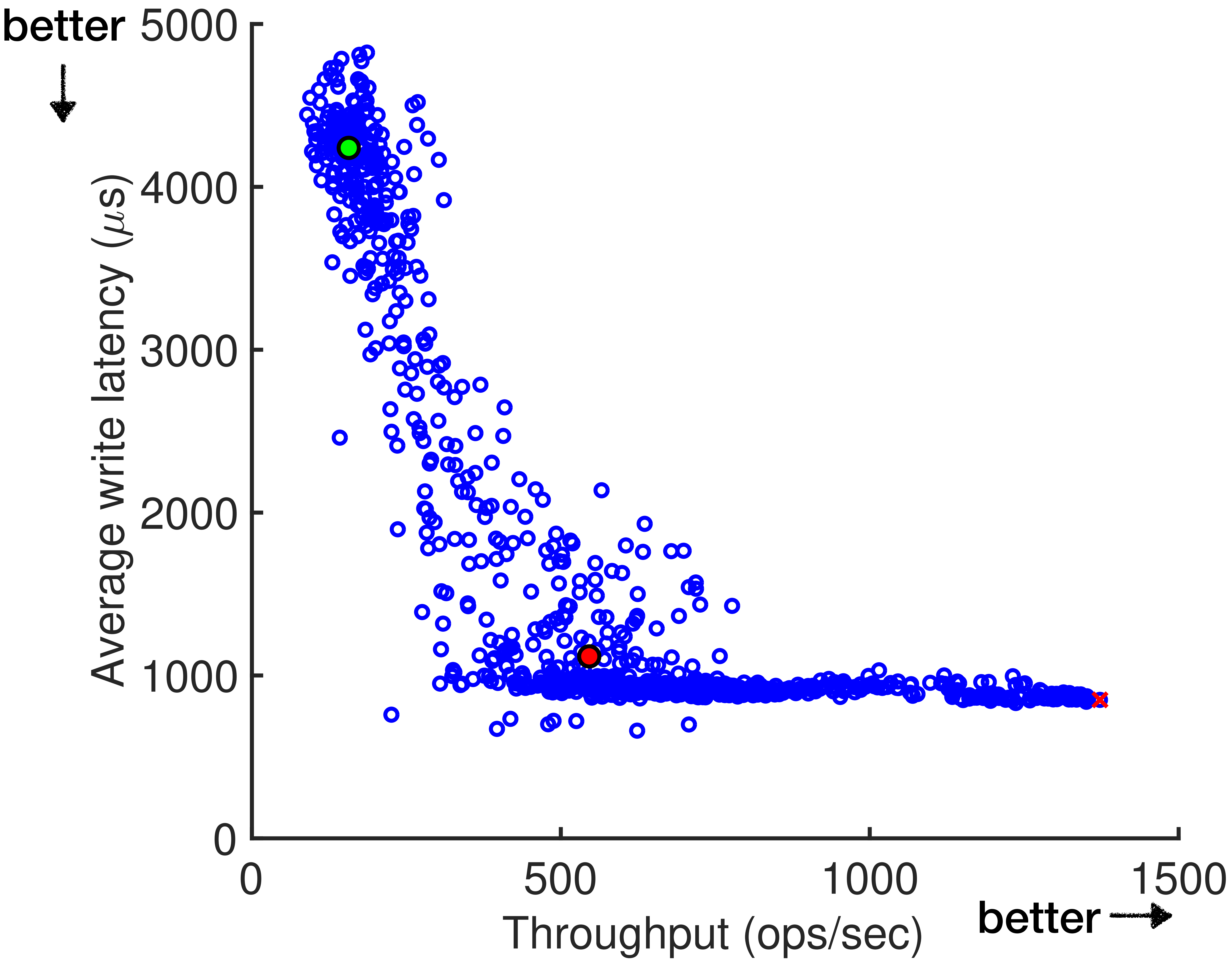


Default configuration was bad, so was the expert'

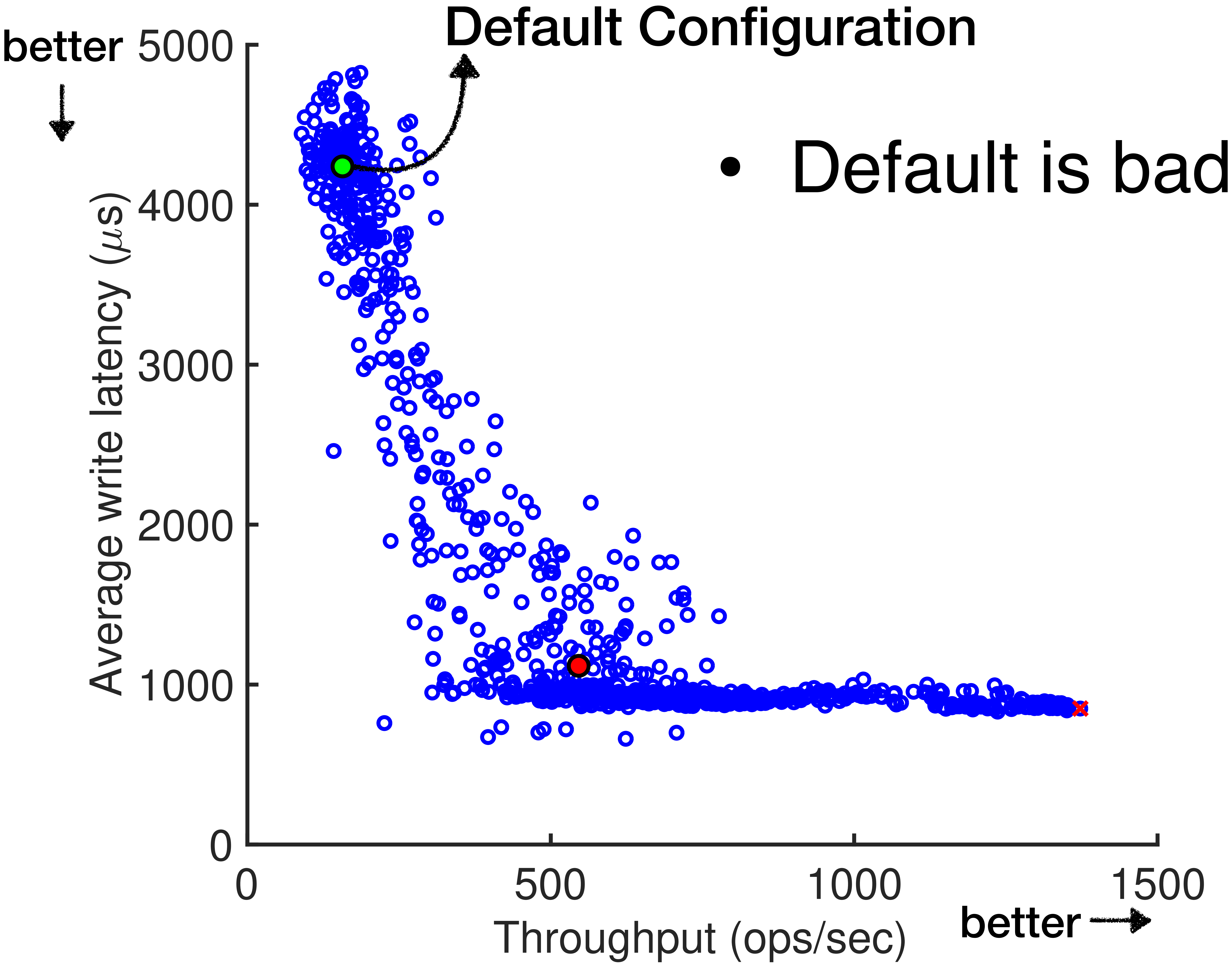


STORM

The default configuration is typically bad and the optimal configuration is noticeably better than median

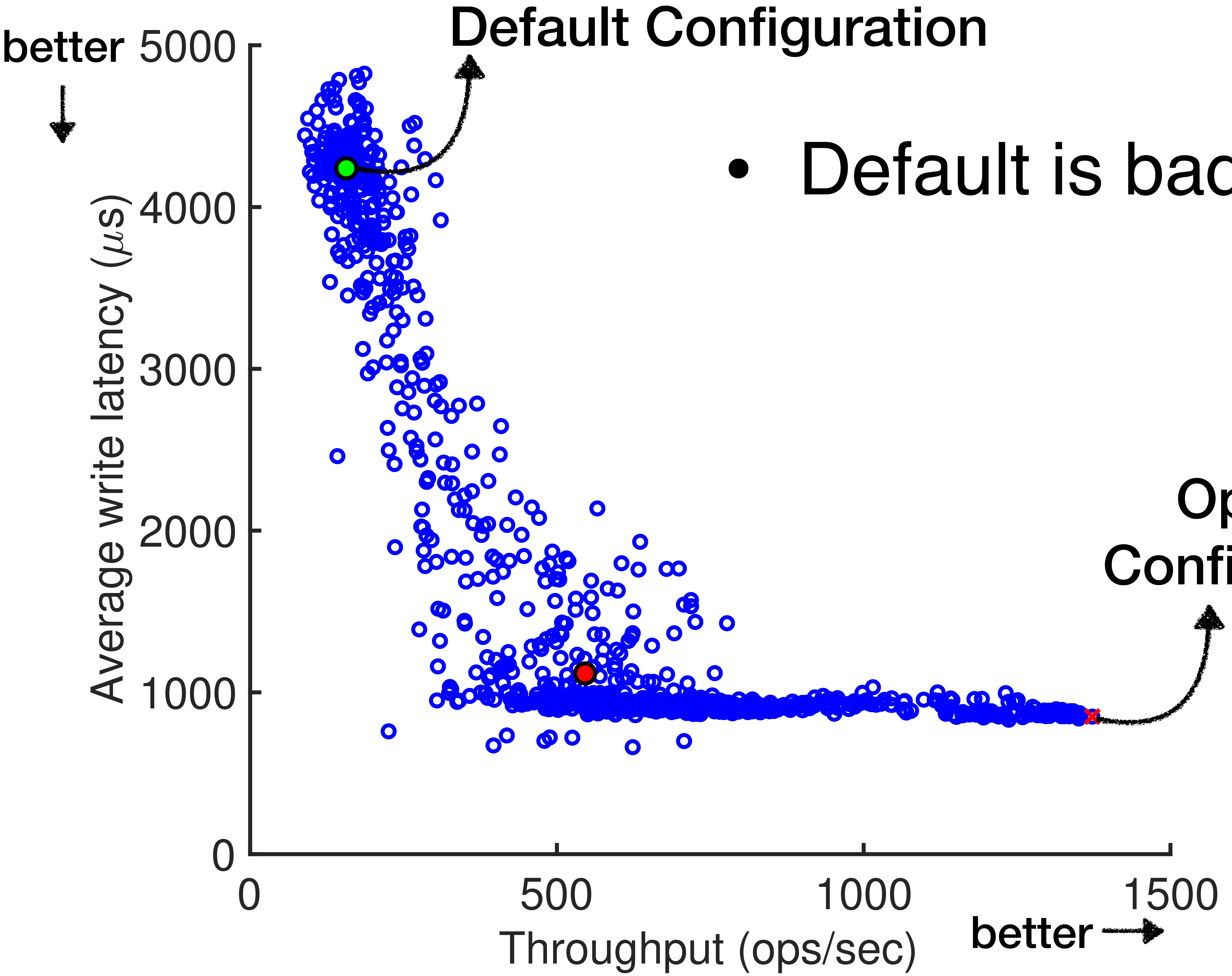


The default configuration is typically bad and the optimal configuration is noticeably better than median



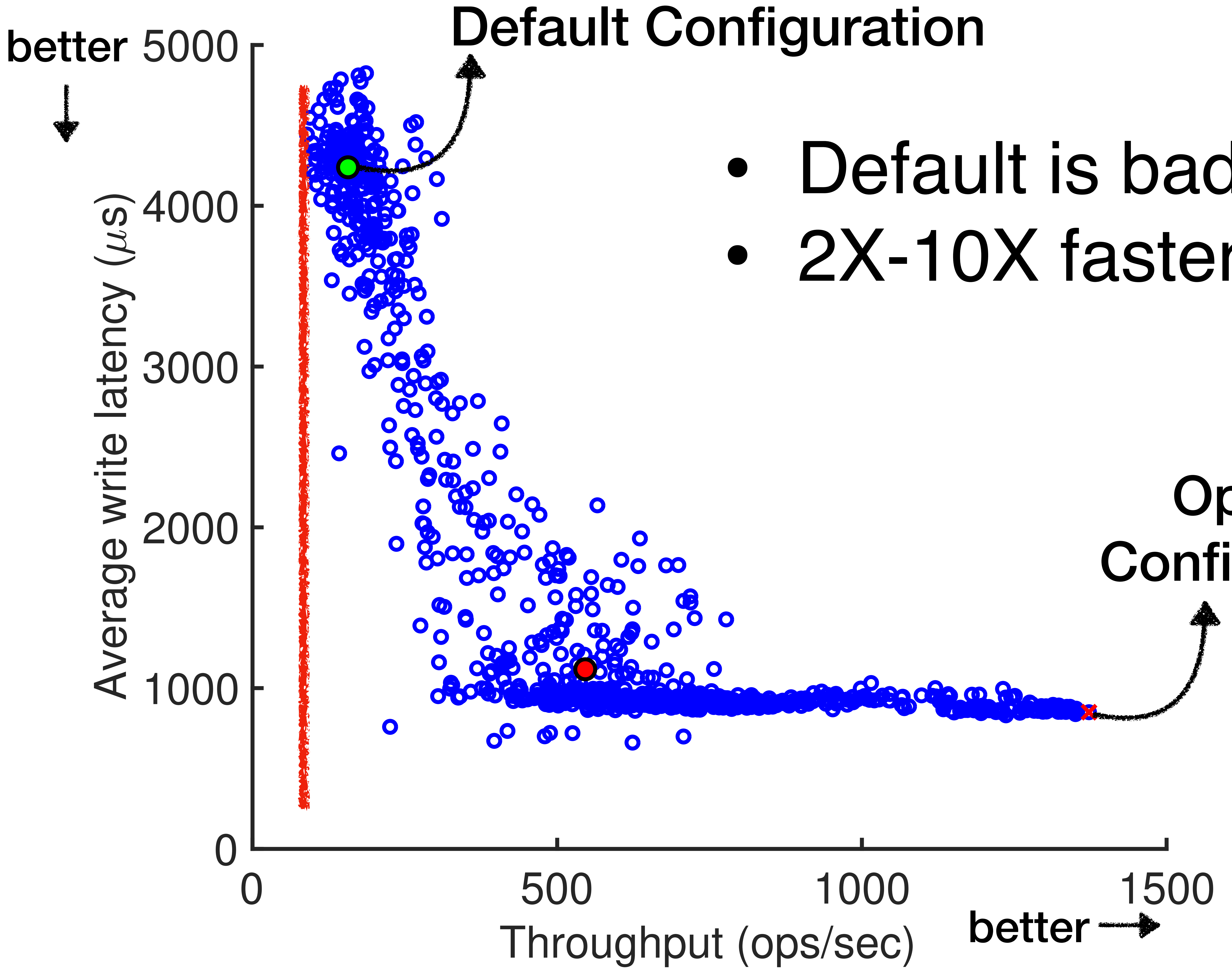
- Default is bad

The default configuration is typically bad and the optimal configuration is noticeably better than median



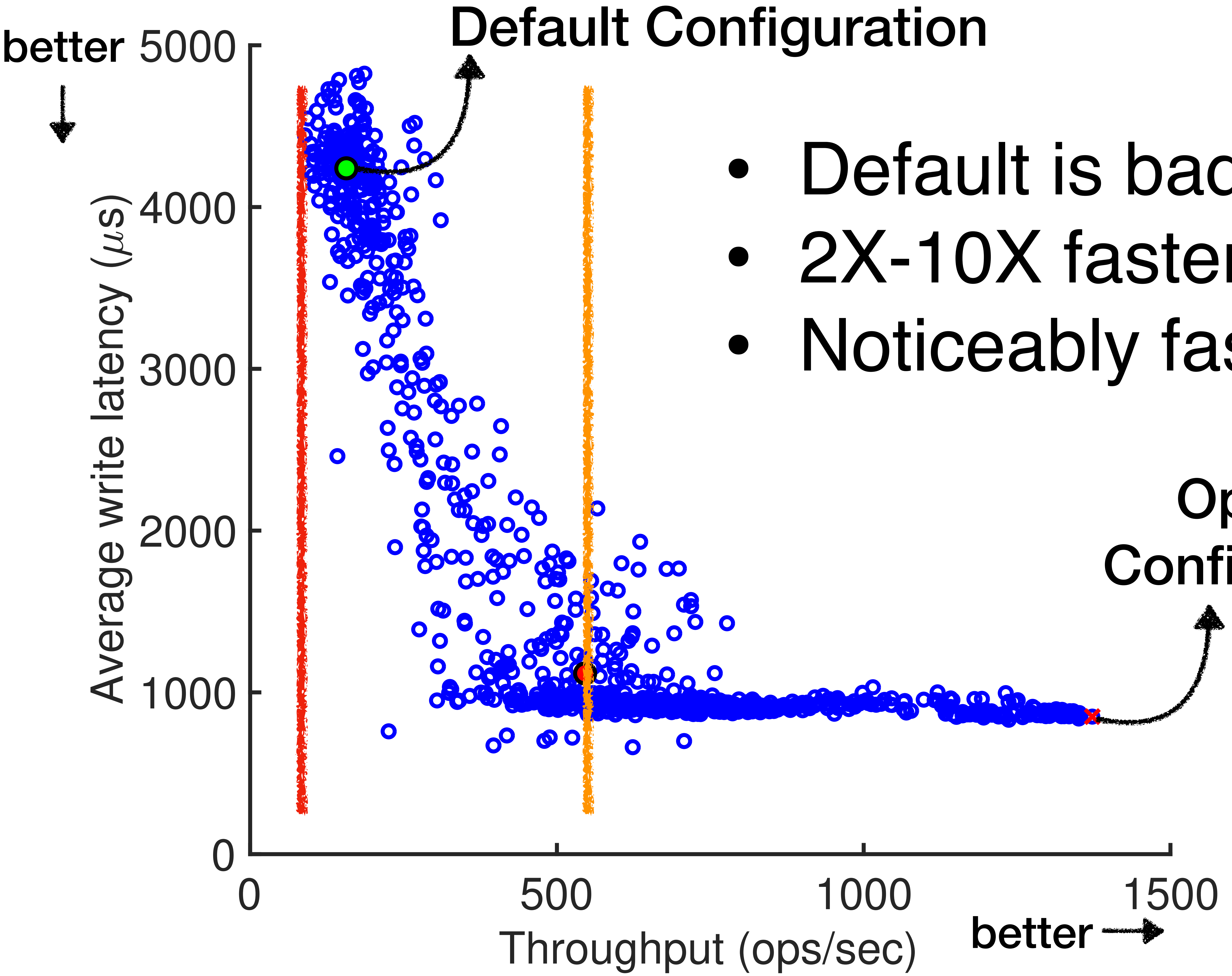
- Default is bad

The default configuration is typically bad and the optimal configuration is noticeably better than median



- Default is bad
- 2X-10X faster than worst

The default configuration is typically bad and the optimal configuration is noticeably better than median



- Default is bad
- 2X-10X faster than worst
- Noticeably faster than median

cassandra

What did happen at the end?

- Achieved the objectives (**100X** user, same experience)
- Saved money by reducing cloud resources up to **20%**
- Our tool was able to identify configurations that was consistently better than expert recommendation

Setting the scene

Setting the scene

Compiler
(e.f., SaC, LLVM)

Setting the scene

Compiler
(e.f., SaC, LLVM)

$$\mathbb{C} = O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$$

Setting the scene

Dead code removal

Compiler
(e.f., SaC, LLVM)

$$\mathbb{C} = \boxed{O_1} \times O_2 \times \cdots \times O_{19} \times O_{20}$$

Setting the scene

Compiler
(e.f., SaC, LLVM)

Dead code removal

Constant folding

Loop unrolling

Function inlining

$$\mathbb{C} = \boxed{O_1} \times \boxed{O_2} \times \cdots \times \boxed{O_{19}} \times \boxed{O_{20}}$$

Setting the scene

Dead code removal **Loop unrolling**

Constant folding **Function inlining**

Compiler
(e.f., SaC, LLVM)

Configuration Space

$$\mathbb{C} = \boxed{O_1} \times \boxed{O_2} \times \cdots \times \boxed{O_{19}} \times \boxed{O_{20}}$$

Setting the scene

Compiler
(e.f., SaC, LLVM)

Configuration
Space

$$\mathbb{C} = \boxed{O_1} \times \boxed{O_2} \times \cdots \times \boxed{O_{19}} \times \boxed{O_{20}}$$

Dead code removal

Constant folding

Loop unrolling

Function inlining

Program

Setting the scene

Dead code removal

Loop unrolling

Constant folding

Function inlining

Compiler
(e.f., SaC, LLVM)

Configuration Space

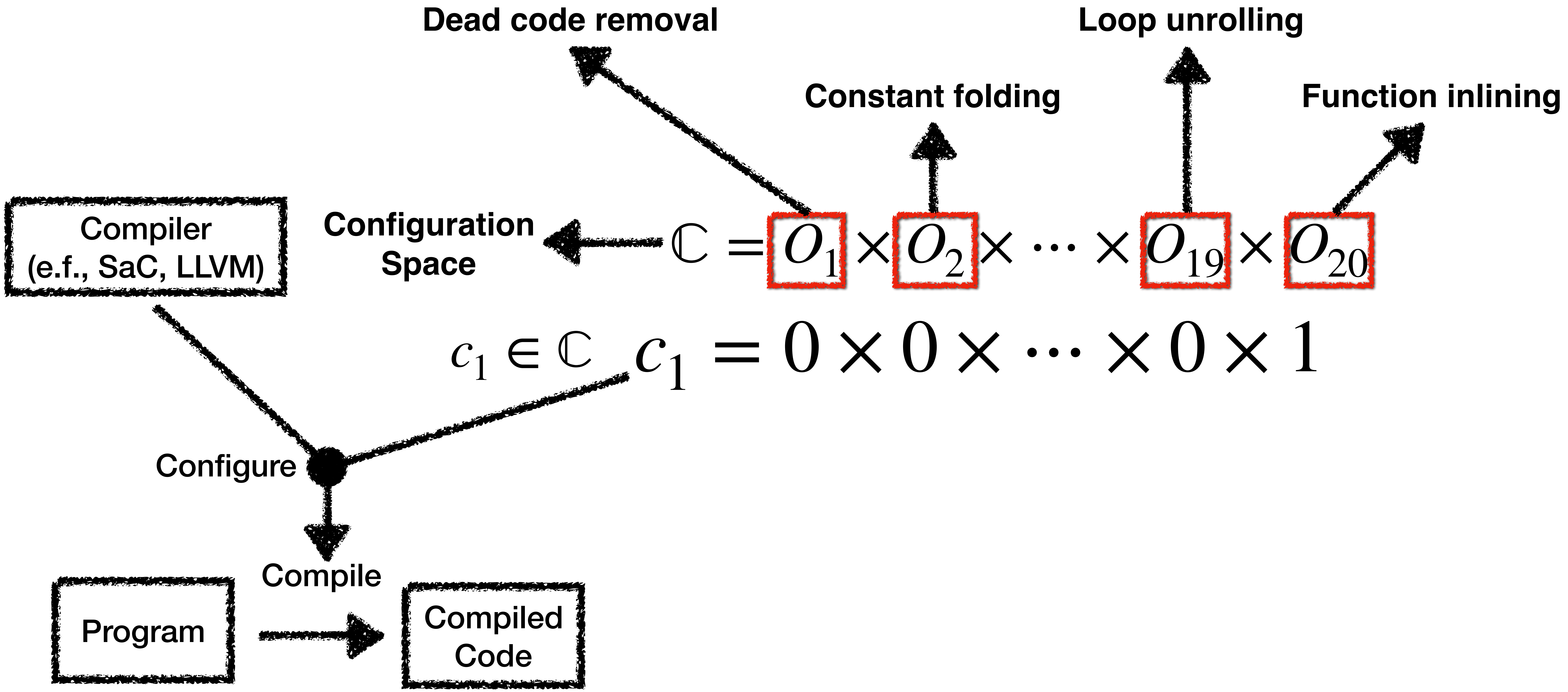
$$\mathbb{C} = \boxed{O_1} \times \boxed{O_2} \times \cdots \times \boxed{O_{19}} \times \boxed{O_{20}}$$

$c_1 \in \mathbb{C}$

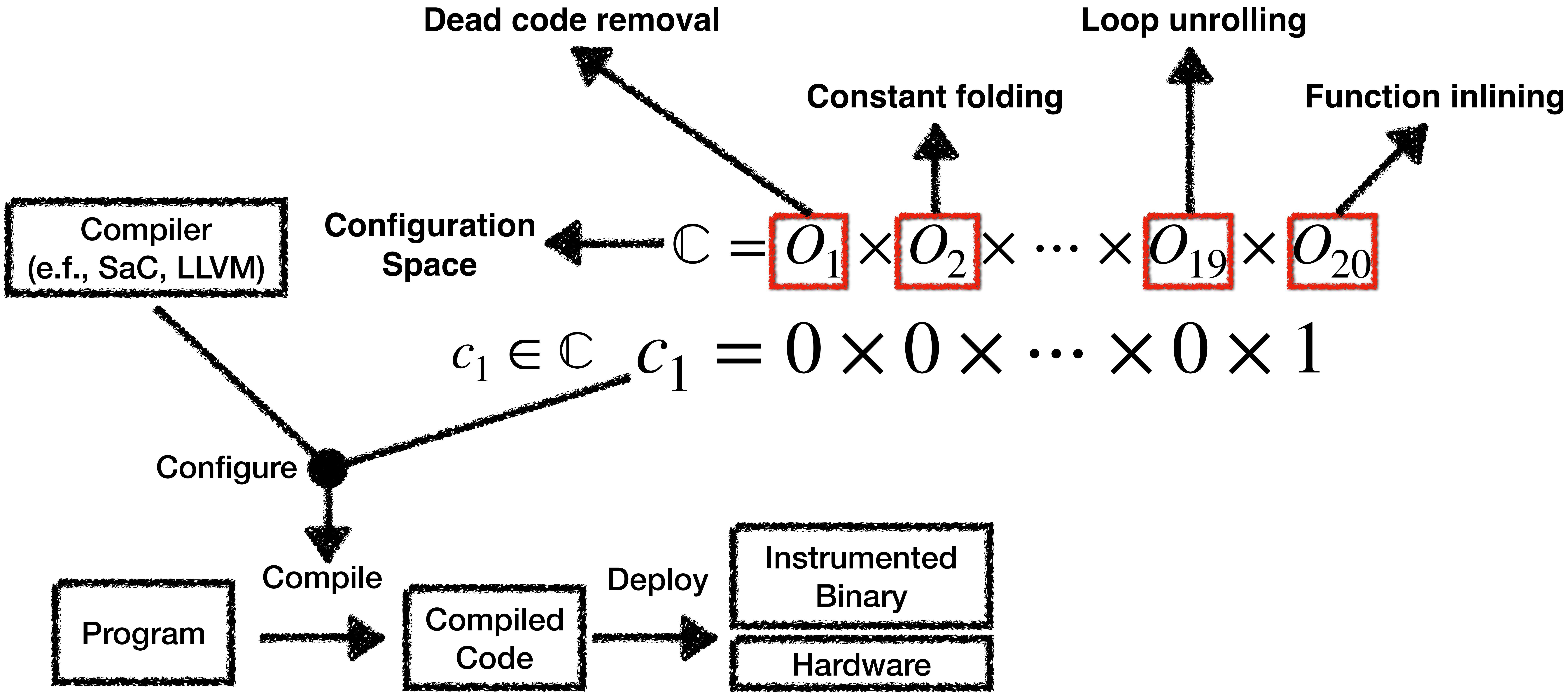
$$c_1 = 0 \times 0 \times \cdots \times 0 \times 1$$

Program

Setting the scene



Setting the scene



Setting the scene

A typical approach for understanding the performance behavior is sensitivity analysis

$$
\begin{array}{ll|c|l}
 & & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} & \\
 & c_1 & 0 \times 0 \times \cdots \times 0 \times 1 & y_1 = f(c_1) \\
 & c_2 & 0 \times 0 \times \cdots \times 1 \times 0 & y_2 = f(c_2) \\
 & c_3 & 0 \times 0 \times \cdots \times 1 \times 1 & y_3 = f(c_3) \\
 & & \cdots & \cdots \\
 & & 1 \times 1 \times \cdots \times 1 \times 0 & \\
 & c_n & 1 \times 1 \times \cdots \times 1 \times 1 & y_n = f(c_n)
\end{array}
$$

A typical approach for understanding the performance behavior is sensitivity analysis

$$
\begin{array}{lcl}
 & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} & \\
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 & y_1 = f(c_1) \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 & y_2 = f(c_2) \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 & y_3 = f(c_3) \\
 & \cdots & \cdots \\
 & 1 \times 1 \times \cdots \times 1 \times 0 & \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1 & y_n = f(c_n)
\end{array}
$$

Training/Sample Set

A typical approach for understanding the performance behavior is sensitivity analysis

$$
\begin{array}{c|c|}
 & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} \\
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 \\
 & \cdots \\
 & 1 \times 1 \times \cdots \times 1 \times 0 \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1
\end{array}
$$

$y_1 = f(c_1)$

$y_2 = f(c_2)$

$y_3 = f(c_3)$

$\cdots$

$y_n = f(c_n)$

$$\boxed{\text{Training/Sample Set}} \xrightarrow{\text{Learn}} \hat{f} \sim f(\cdot)$$

Performance model could be in any appropriate form of black-box models

$$
\begin{array}{l|l|}
 & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} \\
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 \\
 & \cdots \\
 & 1 \times 1 \times \cdots \times 1 \times 0 \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1
\end{array}
$$

$y_1 = f(c_1)$

$y_2 = f(c_2)$

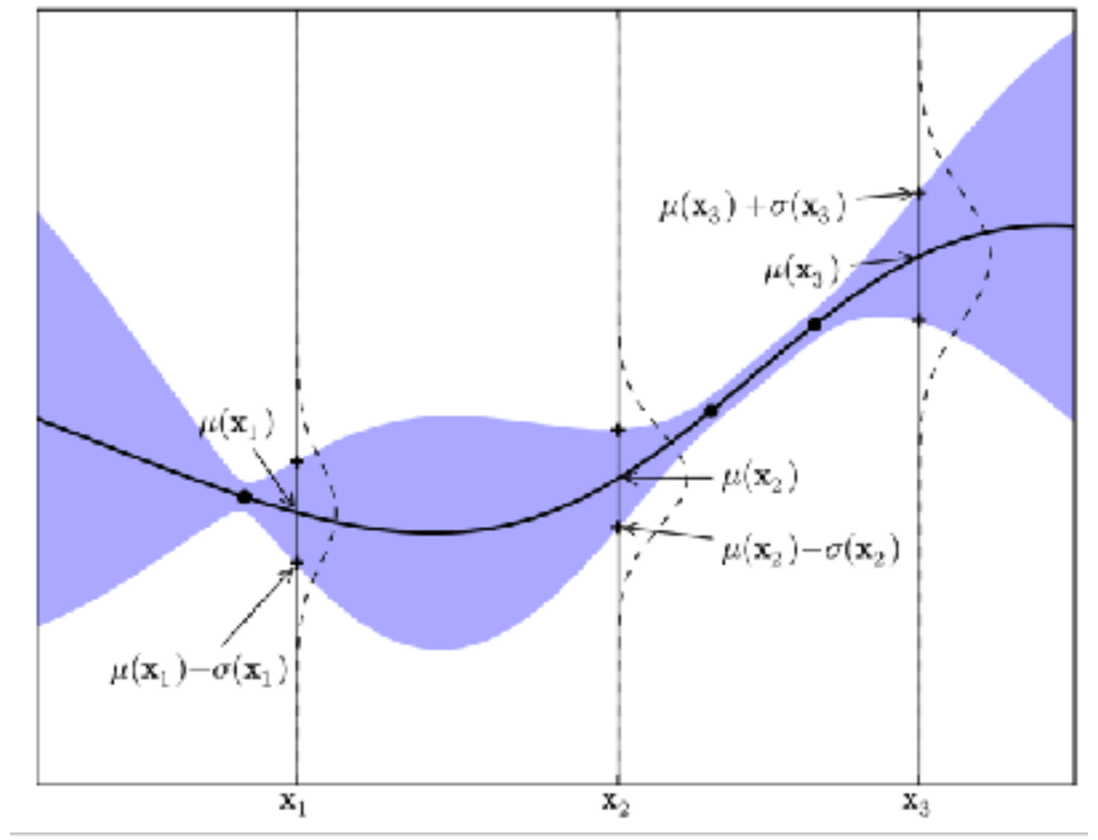
$y_3 = f(c_3)$

$\cdots$

$\boxed{\text{Training/Sample Set}}$ $\hat{f} \sim f(\cdot)$

$y_n = f(c_n)$

Performance model could be in any appropriate form of black-box models



$$
\begin{array}{cc}
 & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} \\
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 \\
 & \cdots \\
 & 1 \times 1 \times \cdots \times 1 \times 0 \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1
\end{array}
$$

$$y_1 = f(c_1)$$

$$y_2 = f(c_2)$$

$$y_3 = f(c_3)$$

$$\cdots$$

Training/Sample Set

$$\hat{f} \sim f(\,\cdot\,)$$

$$y_n = f(c_n)$$

Performance model could be in any appropriate form of black-box models

$$
\begin{array}{ll}
 & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} \\
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 \\
 & \cdots \\
 & 1 \times 1 \times \cdots \times 1 \times 0 \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1
\end{array}
$$

$y_1 = f(c_1)$

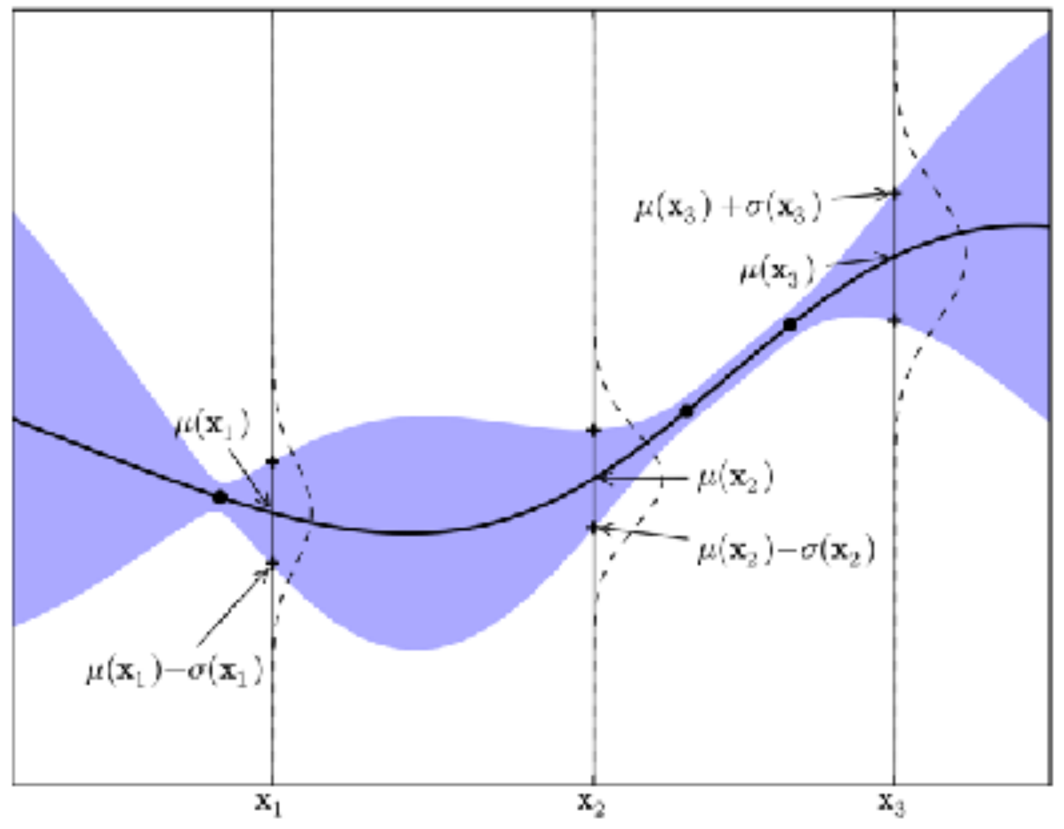
$y_2 = f(c_2)$

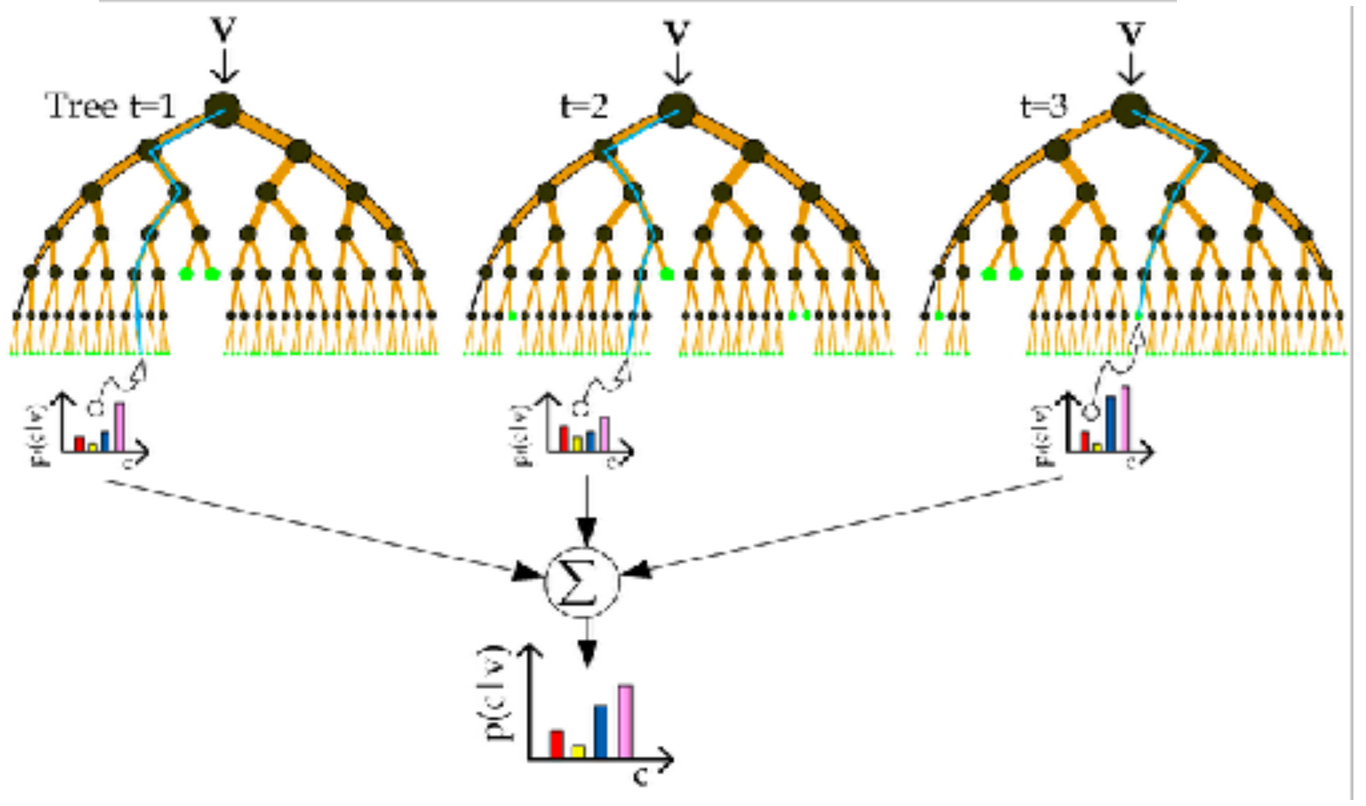
$y_3 = f(c_3)$

$\cdots$

Training/Sample Set $\hat{f} \sim f(\cdot)$

$y_n = f(c_n)$



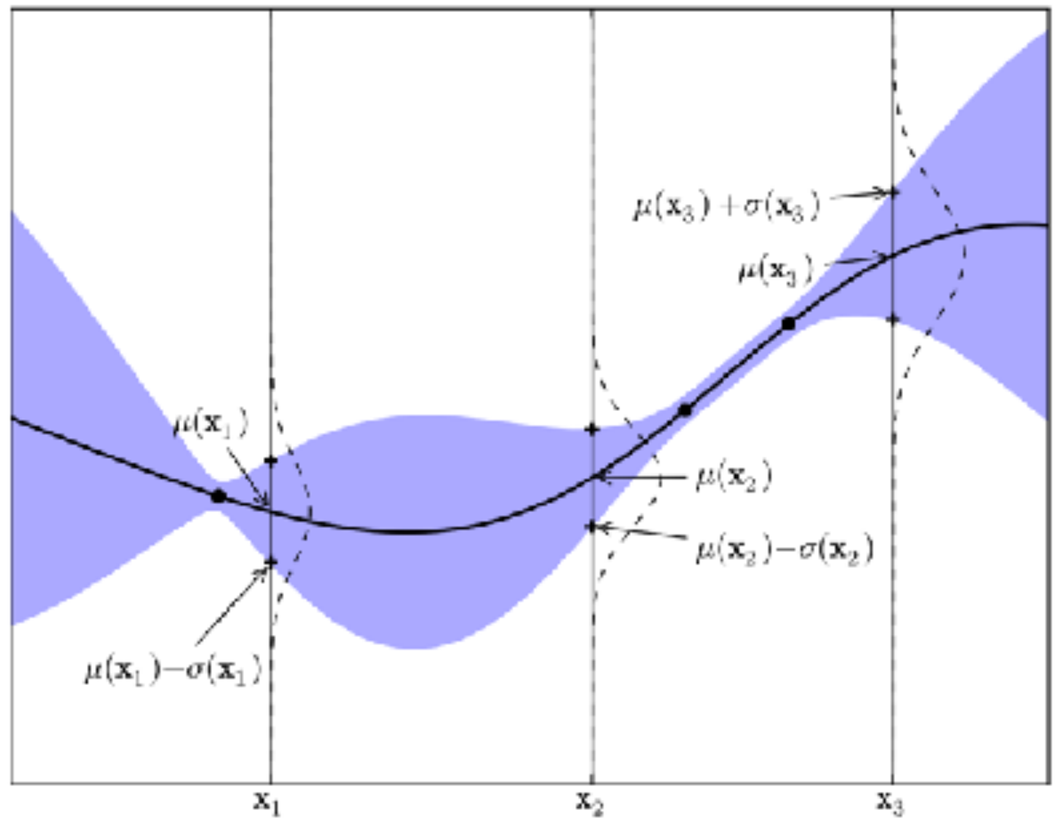


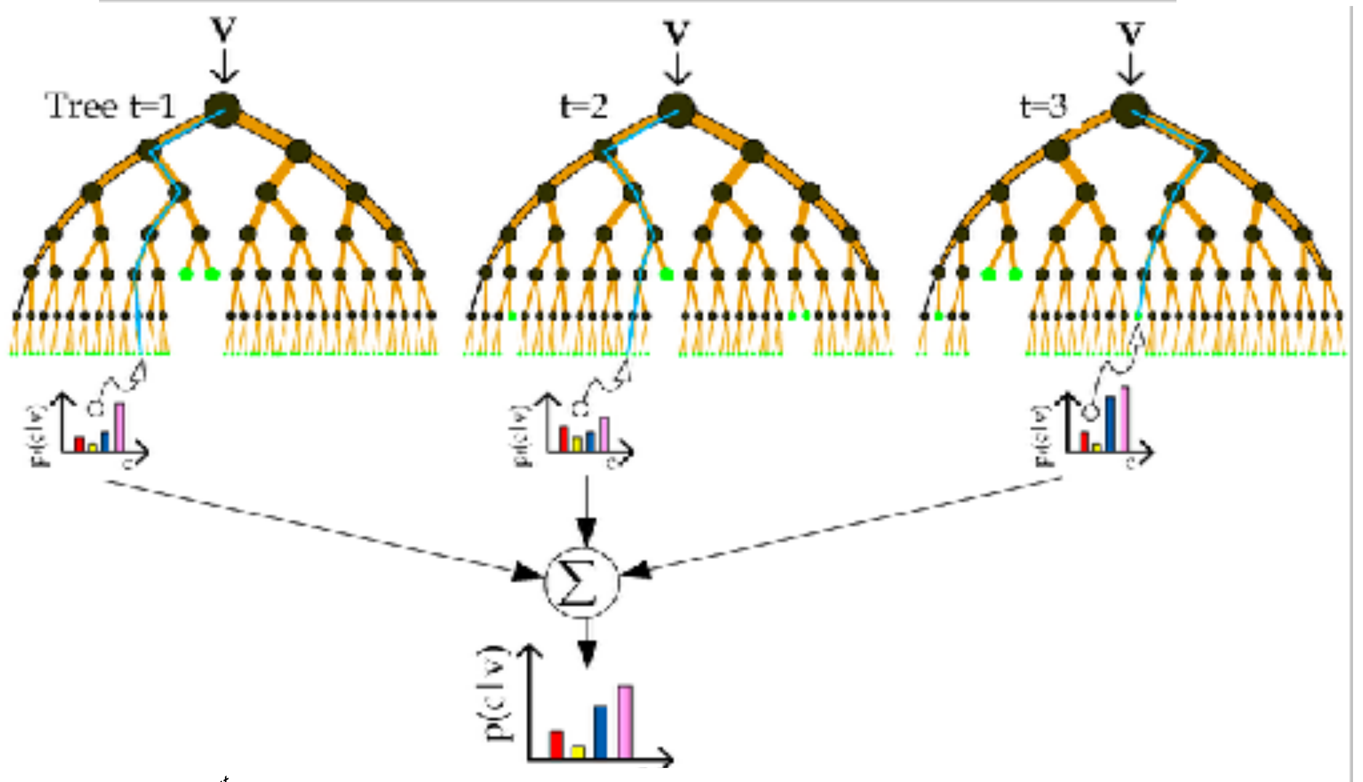
Performance model could be in any appropriate form of black-box models

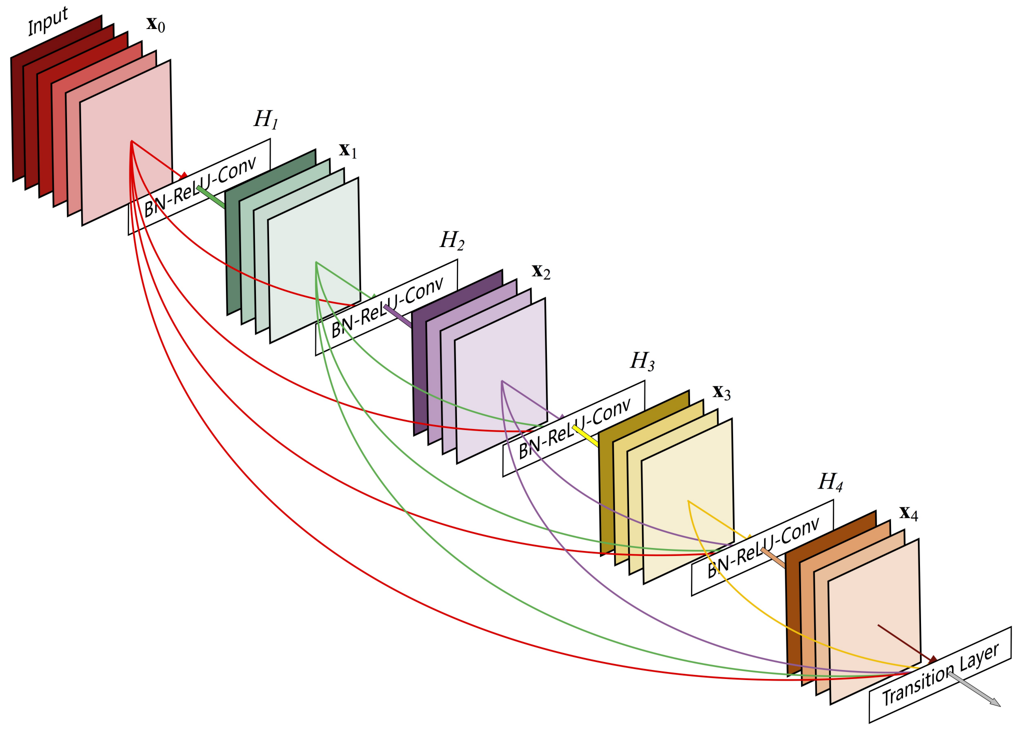
$$
\begin{array}{ll}
 & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} \\
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 \\
 & \cdots \\
 & 1 \times 1 \times \cdots \times 1 \times 0 \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1
\end{array}
\qquad
\begin{array}{l}
y_1 = f(c_1) \\
y_2 = f(c_2) \\
y_3 = f(c_3) \\
\cdots \\
\\
y_n = f(c_n)
\end{array}
$$

Training/Sample Set

$$\hat{f} \sim f(\cdot)$$







Evaluating a performance model

$$\begin{array}{lll}
 & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} & \\
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 & y_1 = f(c_1) \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 & y_2 = f(c_2) \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 & y_3 = f(c_3) \\
 & \cdots & \cdots \\
 & 1 \times 1 \times \cdots \times 1 \times 0 & \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1 & y_n = f(c_n)
\end{array}$$

Training/Sample Set $\xrightarrow{\text{Learn}}$ $\hat{f} \sim f(\cdot)$

Evaluating a performance model

$$
\begin{array}{ll}
 & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} \\
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 \\
 & \cdots \\
 & 1 \times 1 \times \cdots \times 1 \times 0 \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1
\end{array}
$$

$$
\begin{array}{l}
y_1 = f(c_1) \\
y_2 = f(c_2) \\
y_3 = f(c_3) \\
\cdots \\
\text{Training/Sample Set} \xrightarrow{\text{Learn}} \hat{f} \sim f(\cdot) \\
y_n = f(c_n)
\end{array}
$$

Evaluating a performance model

$$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$$

$$\begin{array}{ll} c_1 & 0 \times 0 \times \cdots \times 0 \times 1 \\ c_2 & 0 \times 0 \times \cdots \times 1 \times 0 \\ c_3 & 0 \times 0 \times \cdots \times 1 \times 1 \\ & \cdots \\ & 1 \times 1 \times \cdots \times 1 \times 0 \\ c_n & 1 \times 1 \times \cdots \times 1 \times 1 \end{array}$$

$$y_1 = f(c_1)$$
$$y_2 = f(c_2)$$
$$y_3 = f(c_3)$$
$$\cdots$$

Training/Sample Set

$$y_n = f(c_n)$$

Evaluate →

Accuracy

$$APE(\hat{f}, f) = \frac{|\hat{f}(c) - f(c)|}{f(c)} \times 100$$

Learn →

$$\hat{f} \sim f(\cdot)$$

A performance model contain useful information about influential options and interactions

$$f : \mathbb{C} \to \mathbb{R}$$

$$f(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

A performance model contain useful information about influential options and interactions

$$f : \mathbb{C} \to \mathbb{R}$$

$$f(\cdot) = 1.2 + 3\boxed{o_1} + 5\boxed{o_3} + 0.9\boxed{o_7} + 0.8 o_3 o_7 + 4 o_1 o_3 o_7$$

A performance model contain useful information about influential options and interactions

$$f : \mathbb{C} \to \mathbb{R}$$

$$f(\cdot) = 1.2 + 3\boxed{o_1} + 5\boxed{o_3} + 0.9\boxed{o_7} + 0.8\boxed{o_3 o_7} + 4\boxed{o_1 o_3 o_7}$$

Performance model can then be used to reason about qualities

```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```

Execution time (s)

$$f(\cdot) = 5 + 3 \times o_1$$

Performance model can then be used to reason about qualities

```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```

Execution time (s)

$$f(\cdot) = 5 + 3 \times o_1$$

$$\boxed{f(o_1 := 1) = 8}$$

Performance model can then be used to reason about qualities

```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);
  putenv(envs);
#endif
}
```

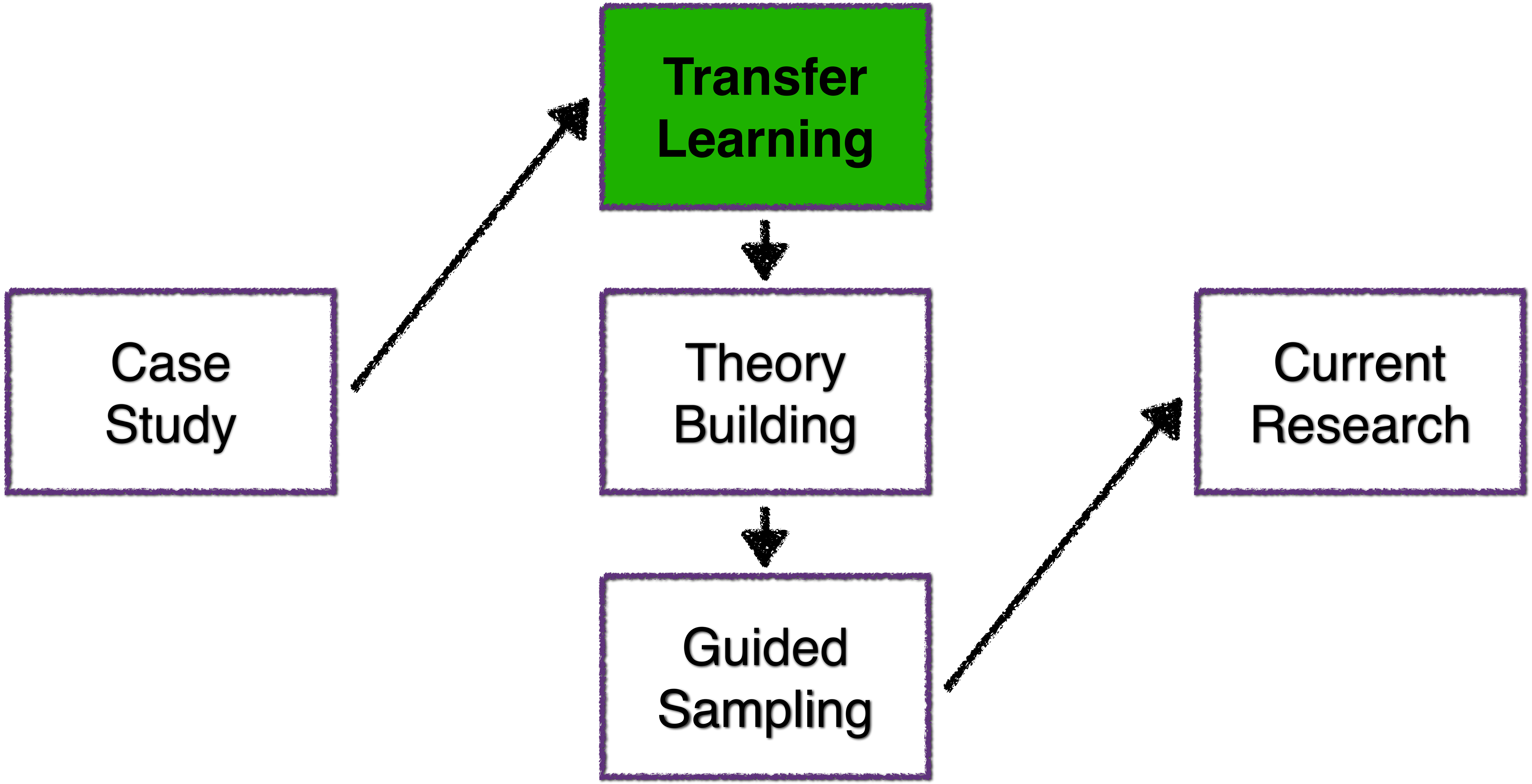
Execution time (s)

$$f(\cdot) = 5 + 3 \times o_1$$

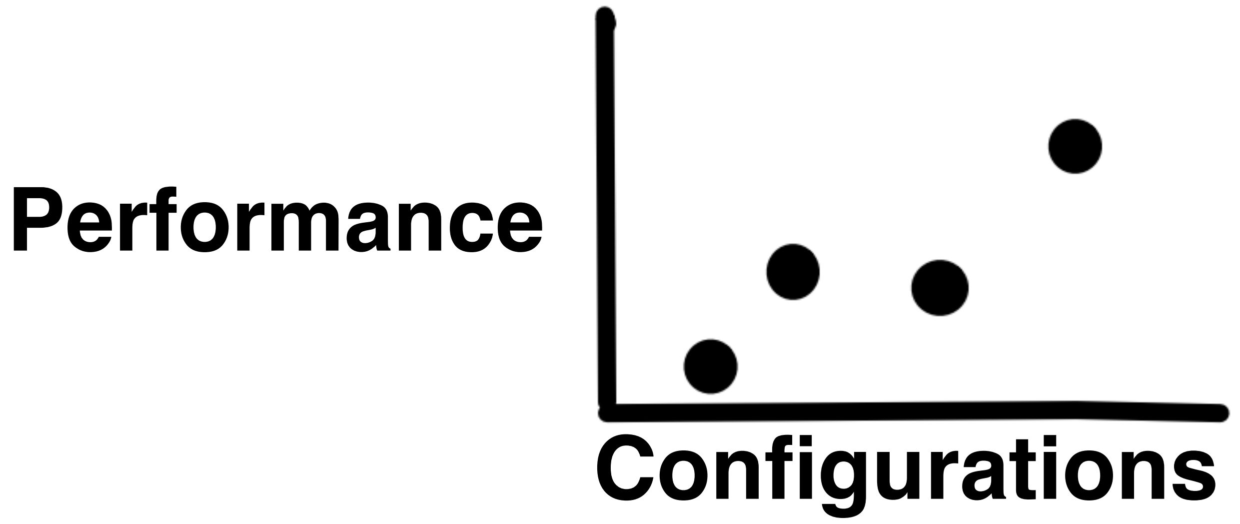
$$\boxed{f(o_1 := 1) = 8}$$

$$\boxed{f(o_1 := 0) = 5}$$

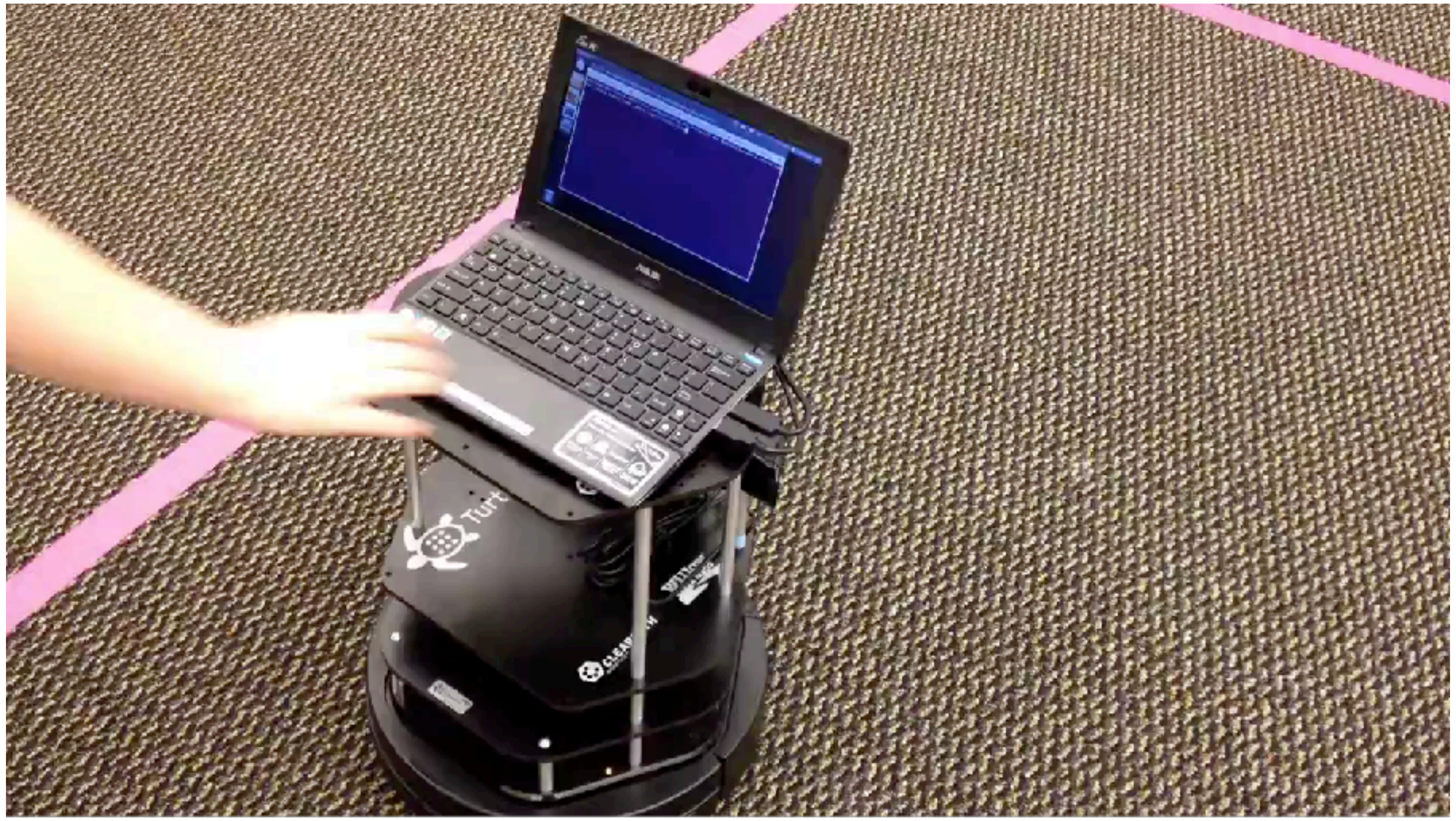
Outline


Transfer Learning
Case Study
Theory Building
Current Research
Guided Sampling

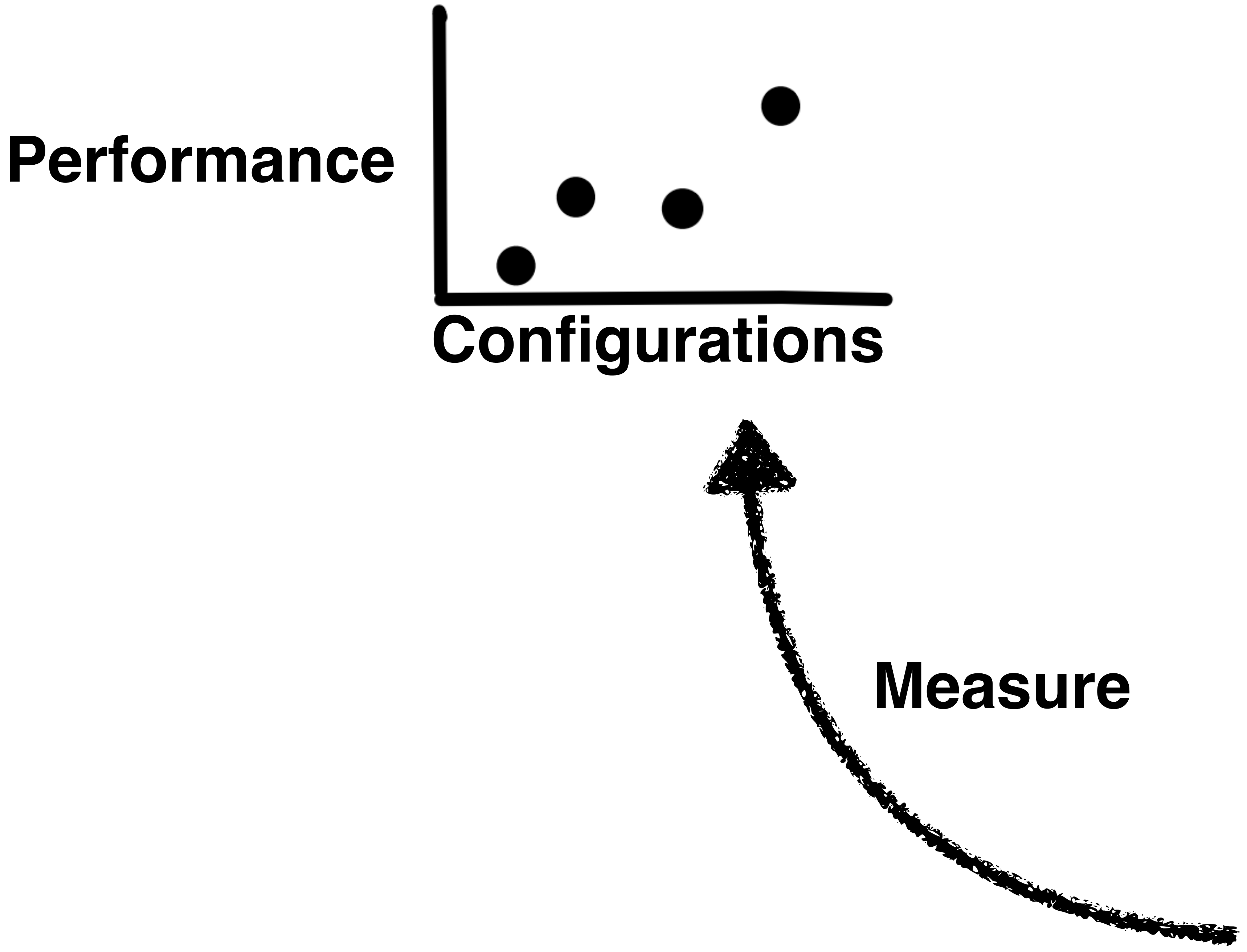
Insight: Performance measurements of the real system is “**similar**” to the ones from the simulators

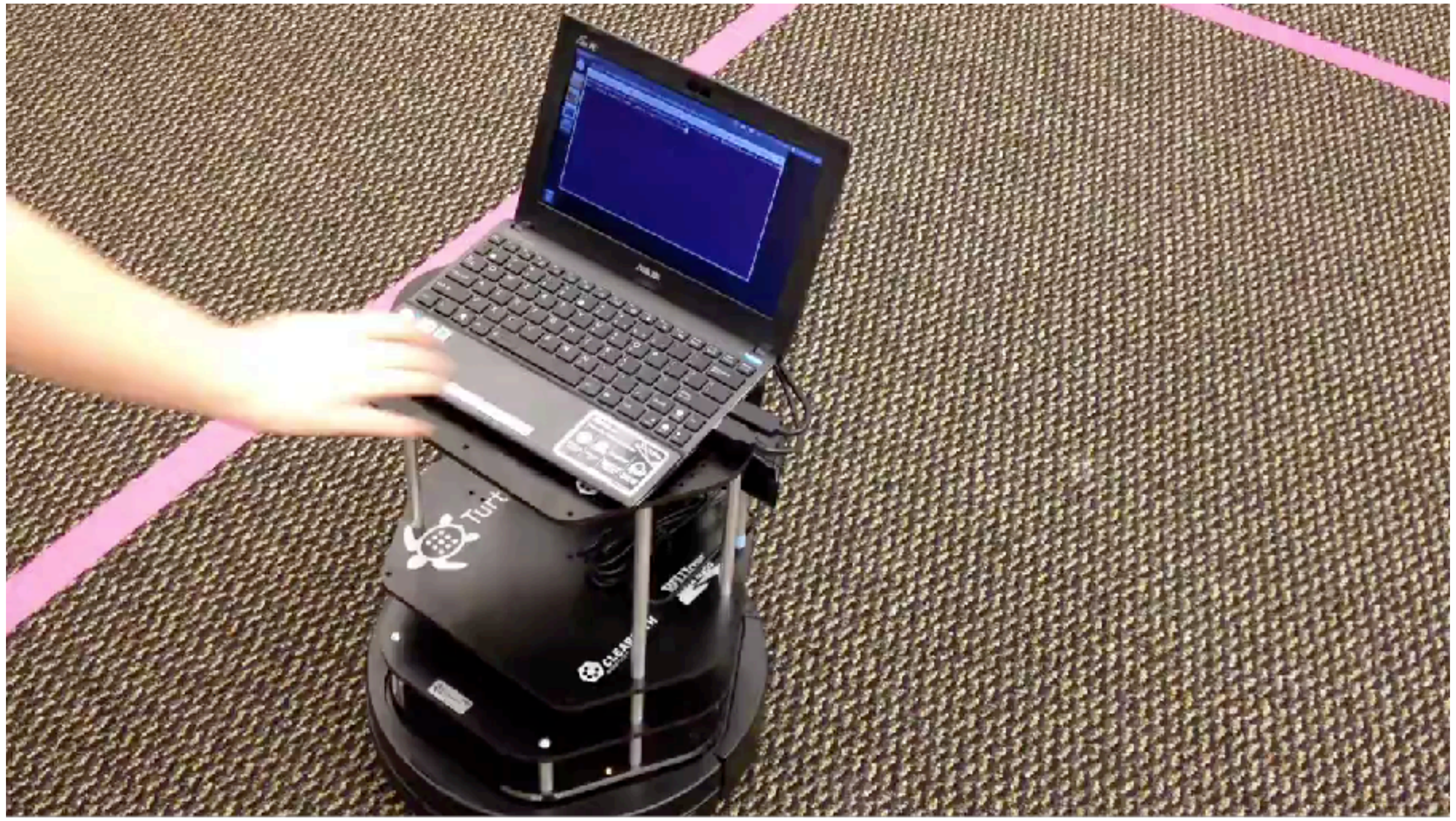


Measure



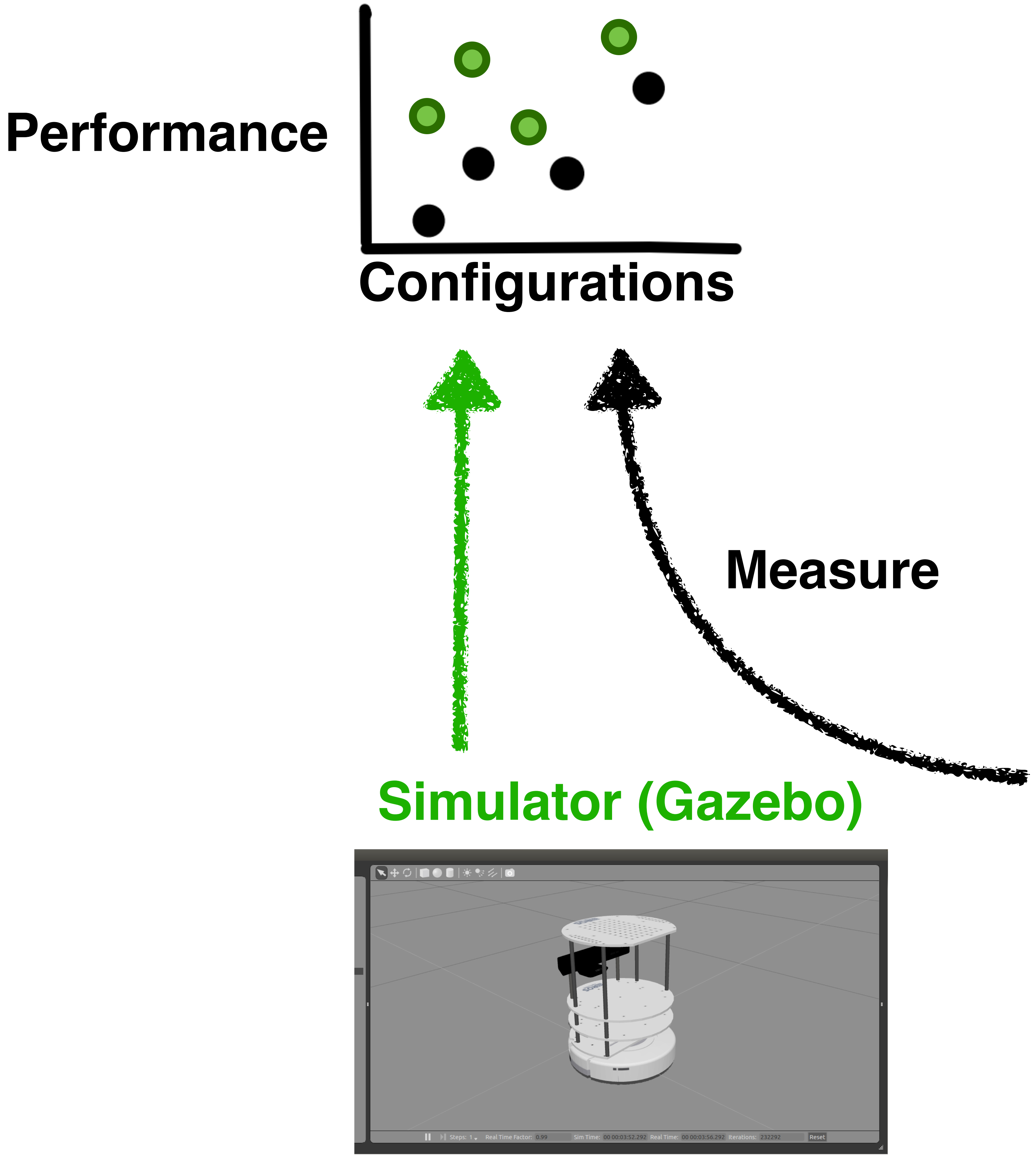
Insight: Performance measurements of the real system is "**similar**" to the ones from the simulators

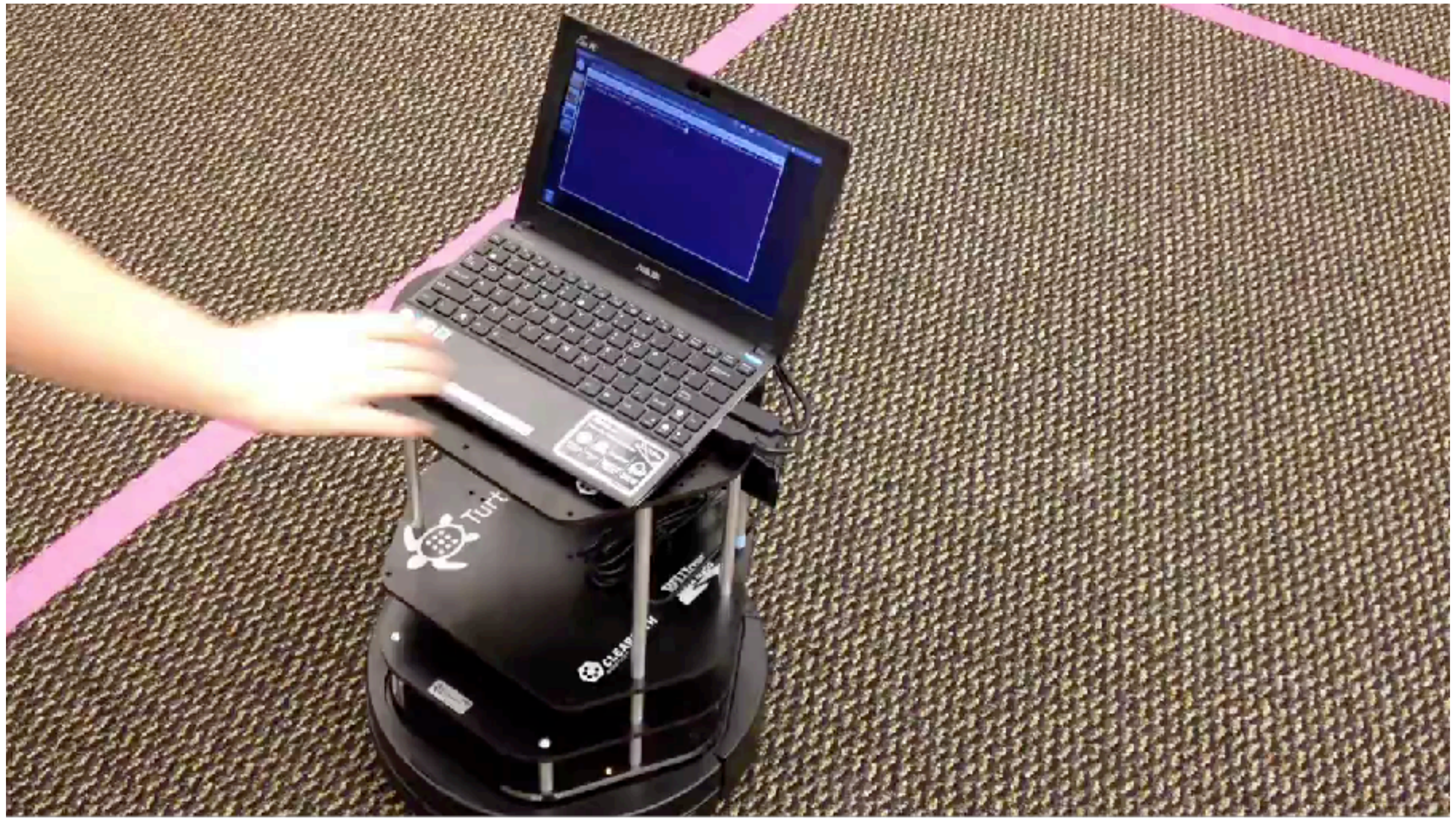




Insight: Performance measurements of the real system is “**similar**” to the ones from the simulators

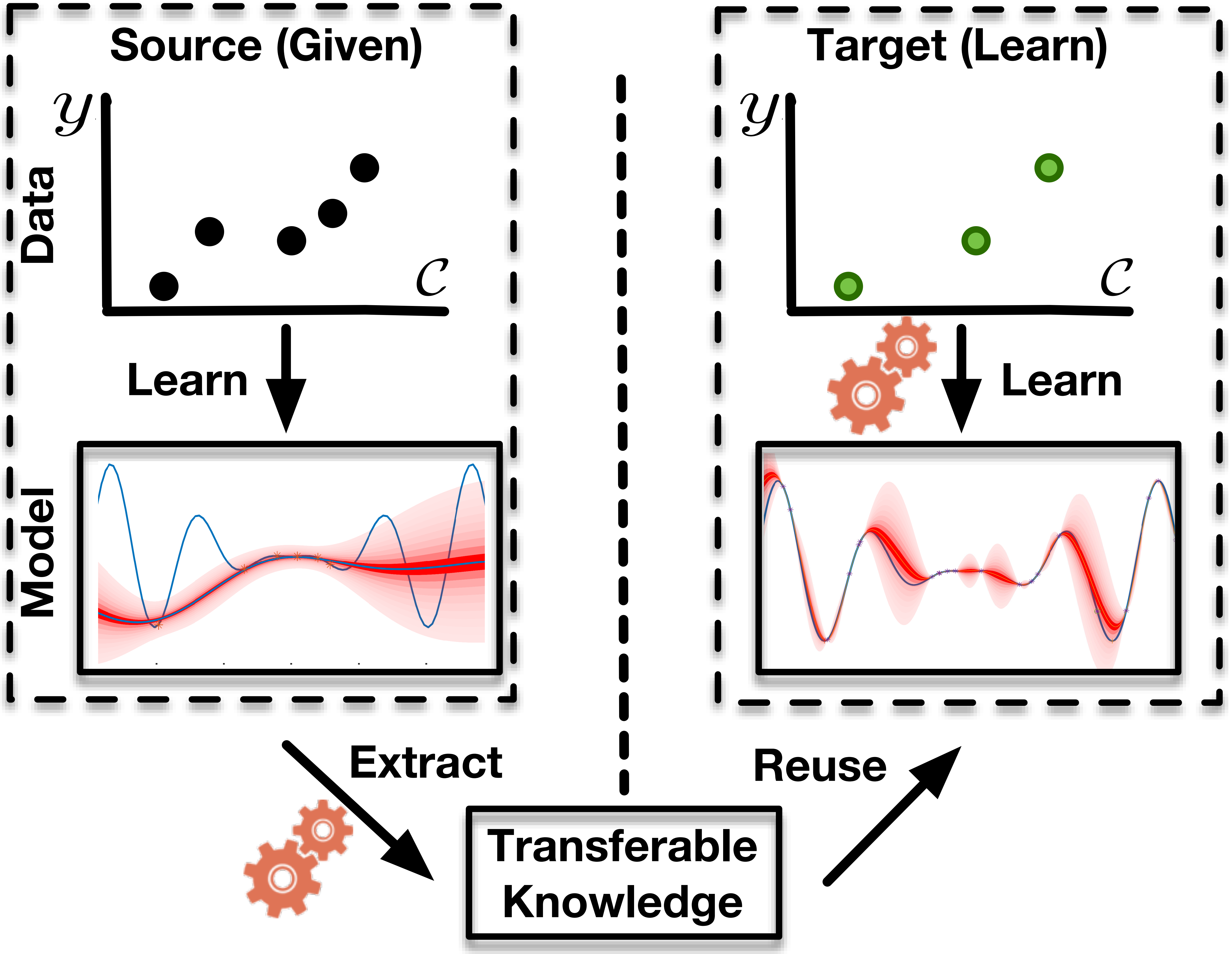
So why not reuse these data, instead of measuring on real robot?



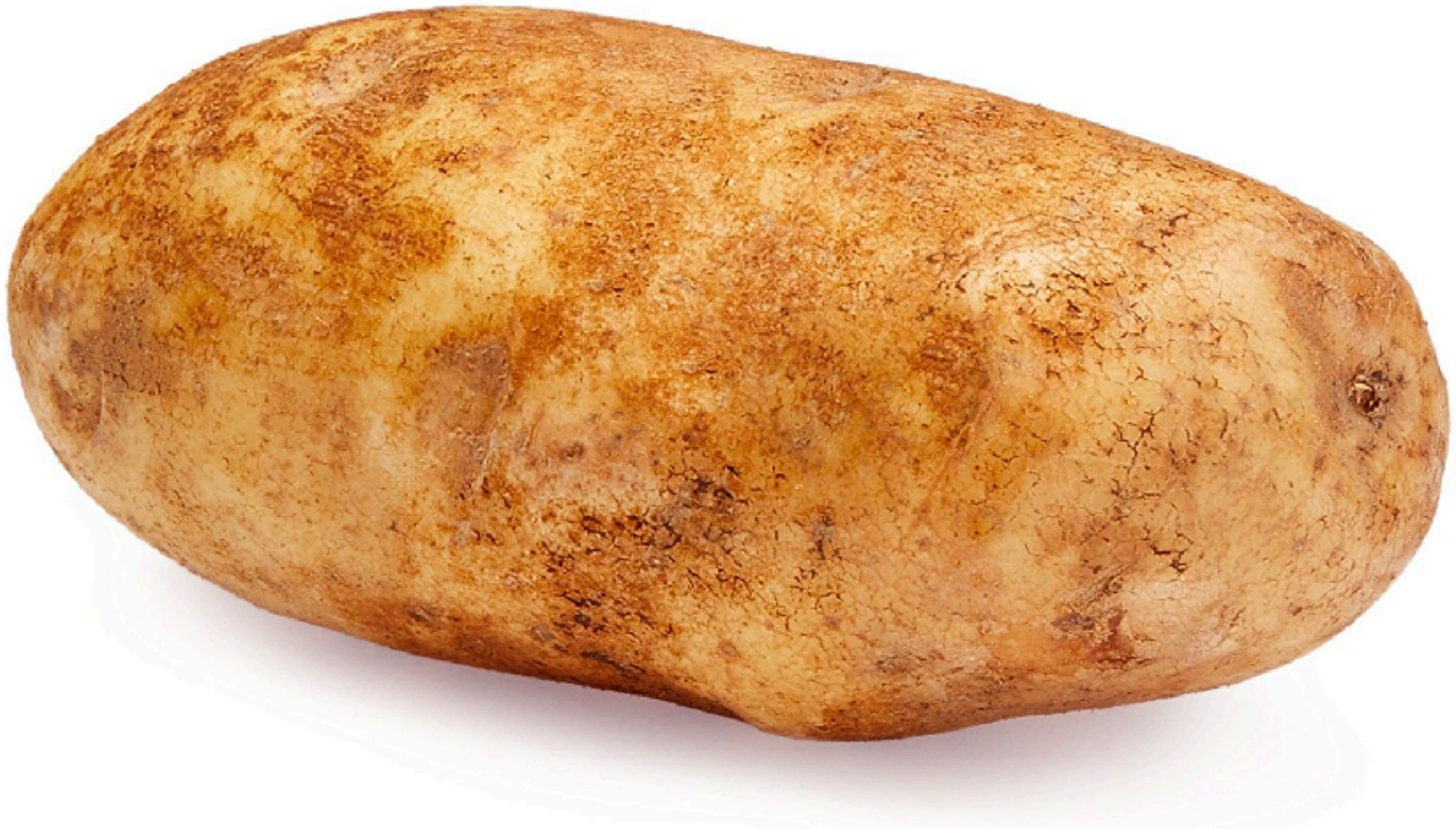


We developed methods to make learning cheaper via **transfer learning**

Goal: Gain strength by transferring information across environments



What is transfer learning?



What is transfer learning?

What is the advantage of transfer learning?

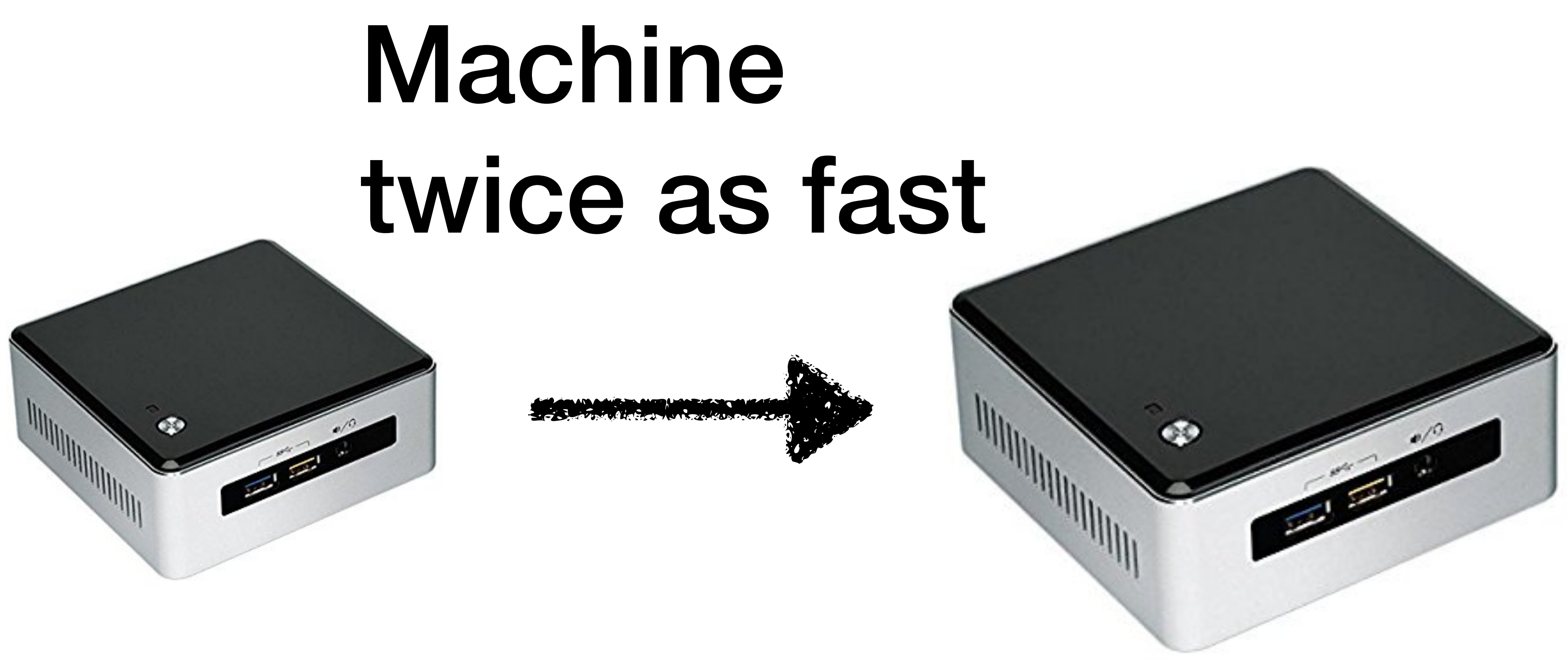
- During learning you may need thousands of rotten and fresh potato and hours of training to learn.

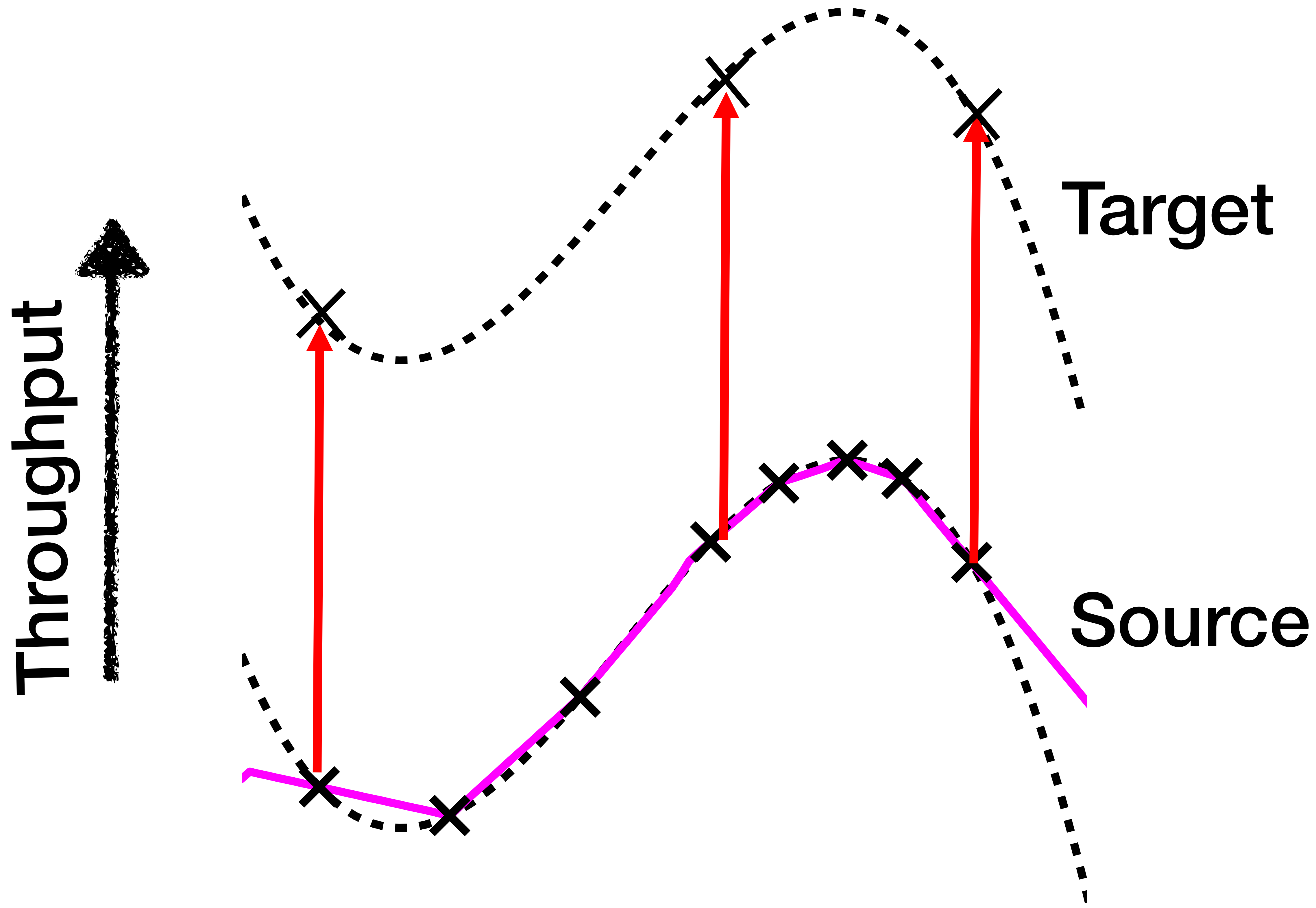
- But now using the same knowledge of rotten features you can identify rotten tomato with less samples and training time.

- You may have learned during daytime with enough light and exposure; but your present tomato identification job is at night.

- You may have learned sitting very close, just beside the box of potato; but now for tomato identification you are in the other side of the glass.

A simple transfer learning via model shift





[Pavel Valov, et al. “**Transferring performance prediction models**…”, ICPE’17]

Our transfer learning solution

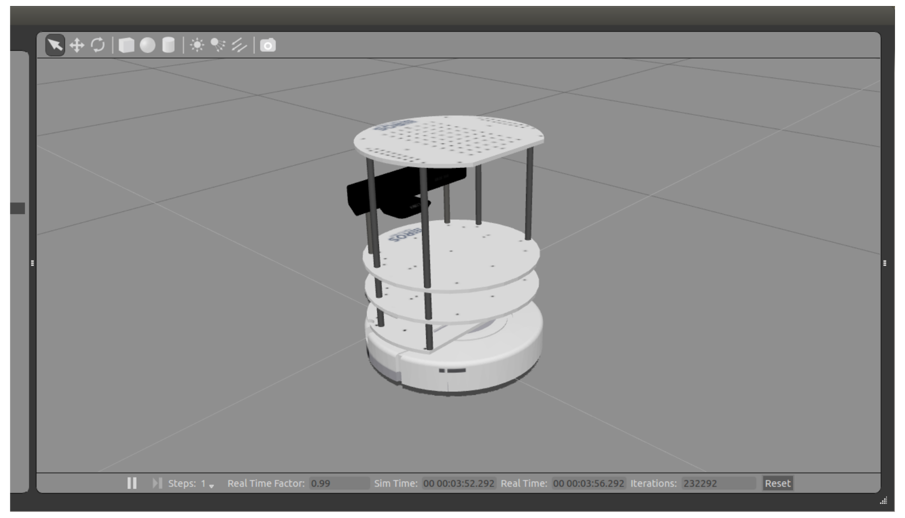
TurtleBot

[P. Jamshidi, et al., "**Transfer learning for improving model predictions** ….", SEAMS'17]

Our transfer learning solution

TurtleBot

Simulator (Gazebo)

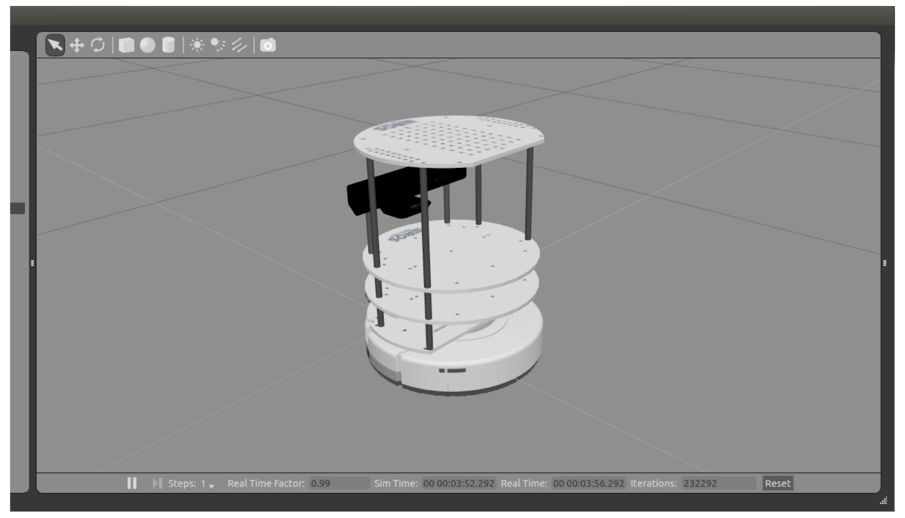


[P. Jamshidi, et al., “**Transfer learning for improving model predictions** ….”, SEAMS’17]

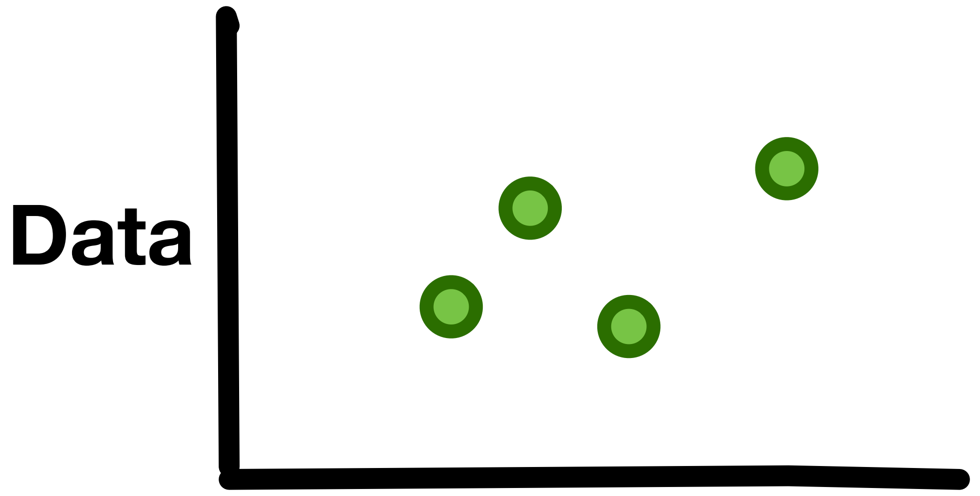
Our transfer learning solution

TurtleBot

Simulator (Gazebo)



Measure

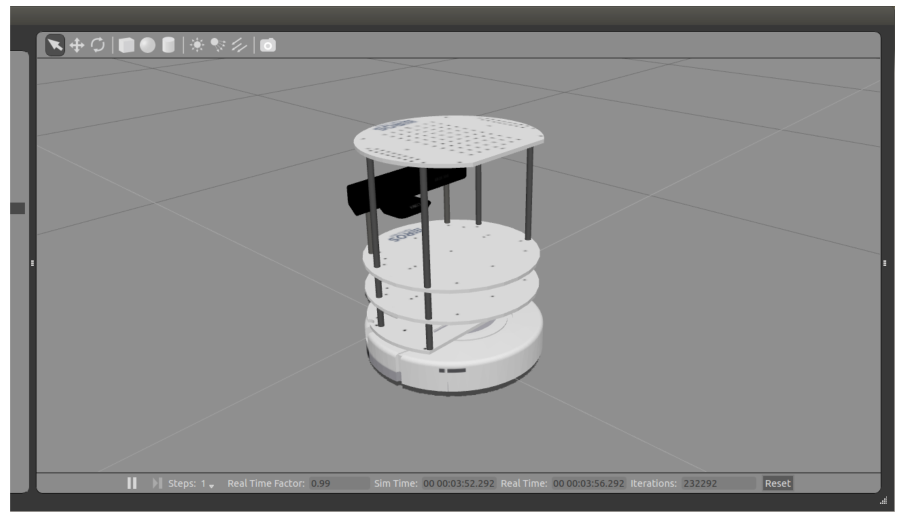


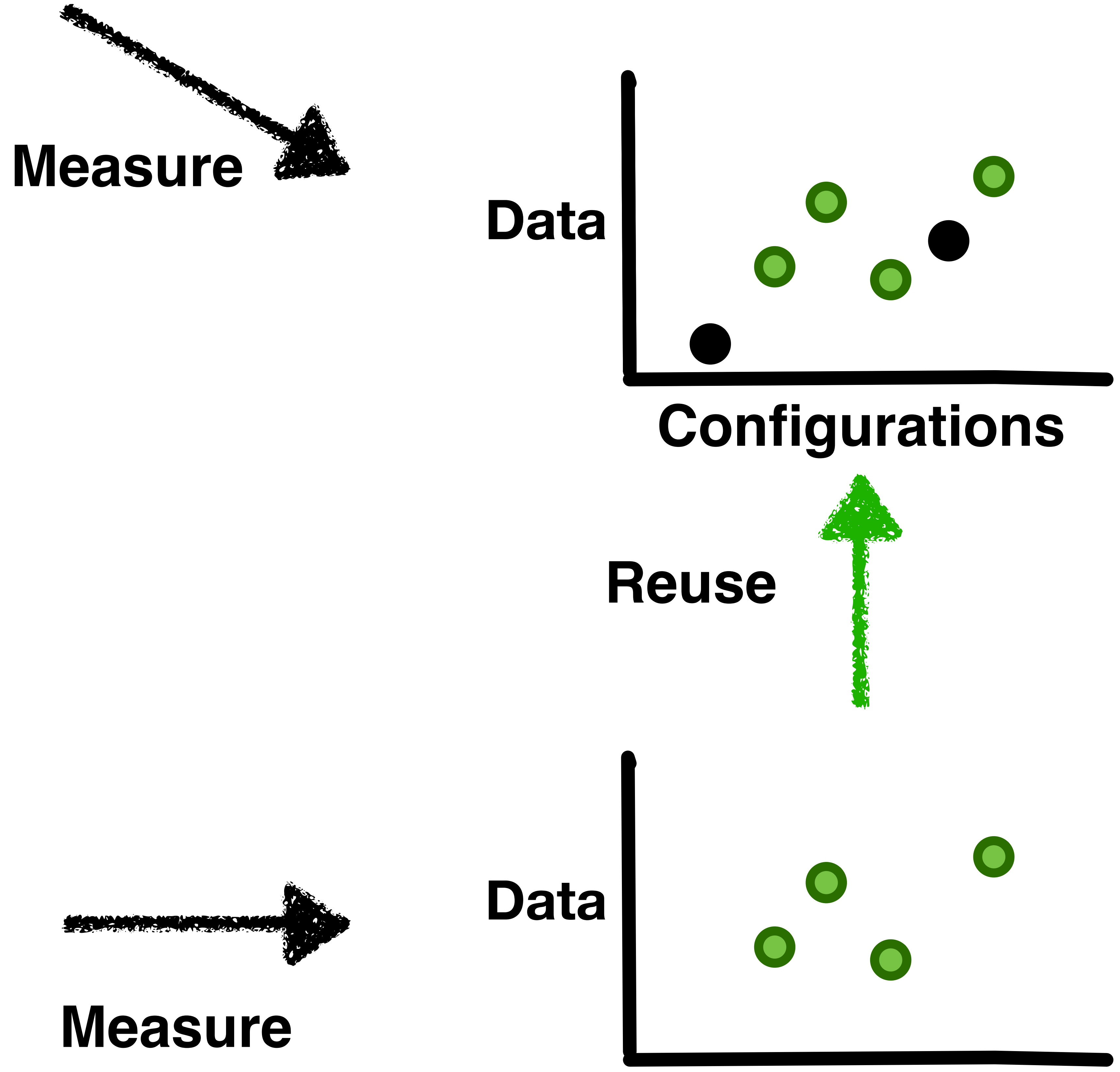
[P. Jamshidi, et al., "**Transfer learning for improving model predictions** ….", SEAMS'17]

Our transfer learning solution

TurtleBot

Simulator (Gazebo)





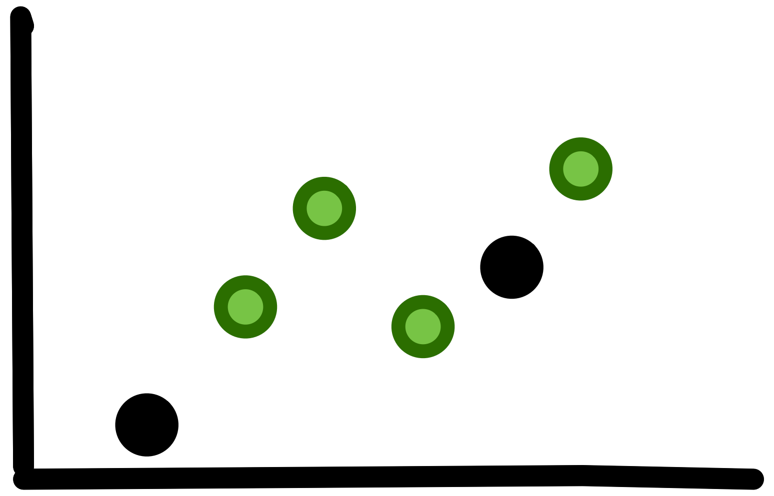
[P. Jamshidi, et al., “**Transfer learning for improving model predictions** ….”, SEAMS’17]

Our transfer learning solution

TurtleBot

Measure

Data

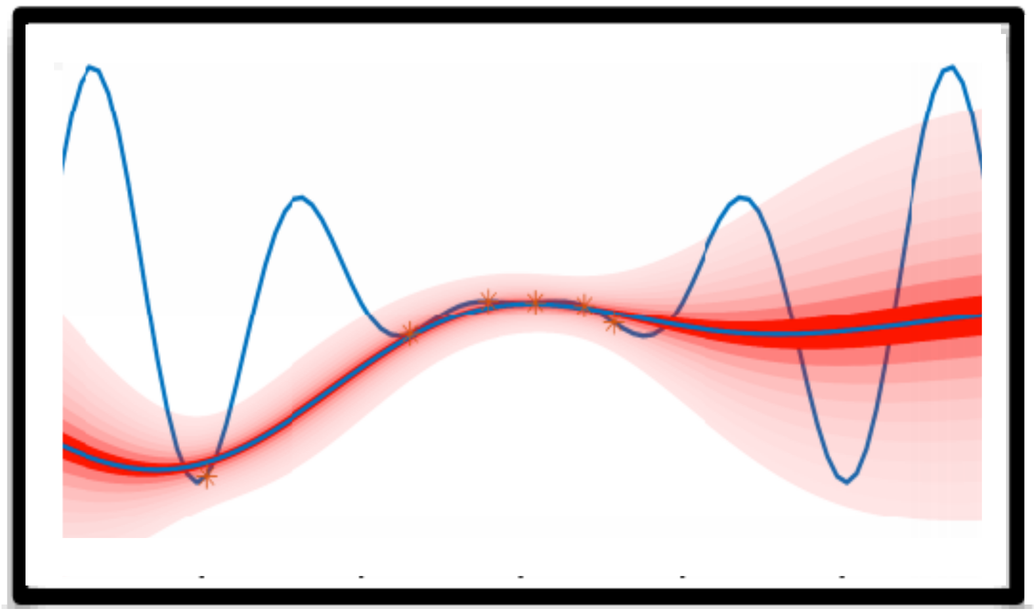


Configurations

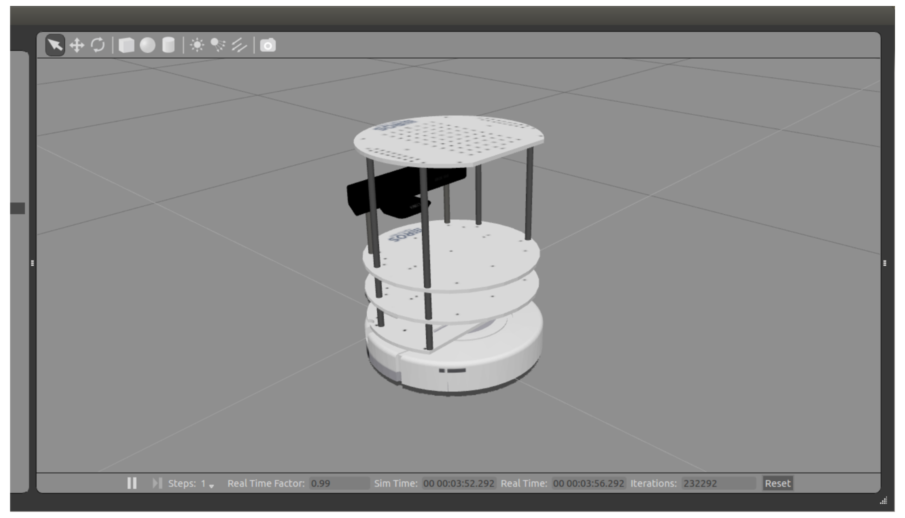
Reuse

Learn

$$f(o_1, o_2) = 5 + 3o_1 + 15o_2 - 7o_1 \times o_2$$

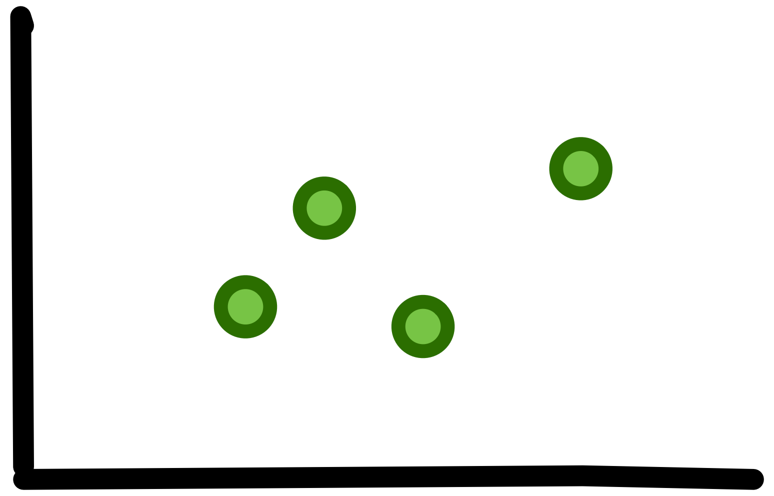


Simulator (Gazebo)



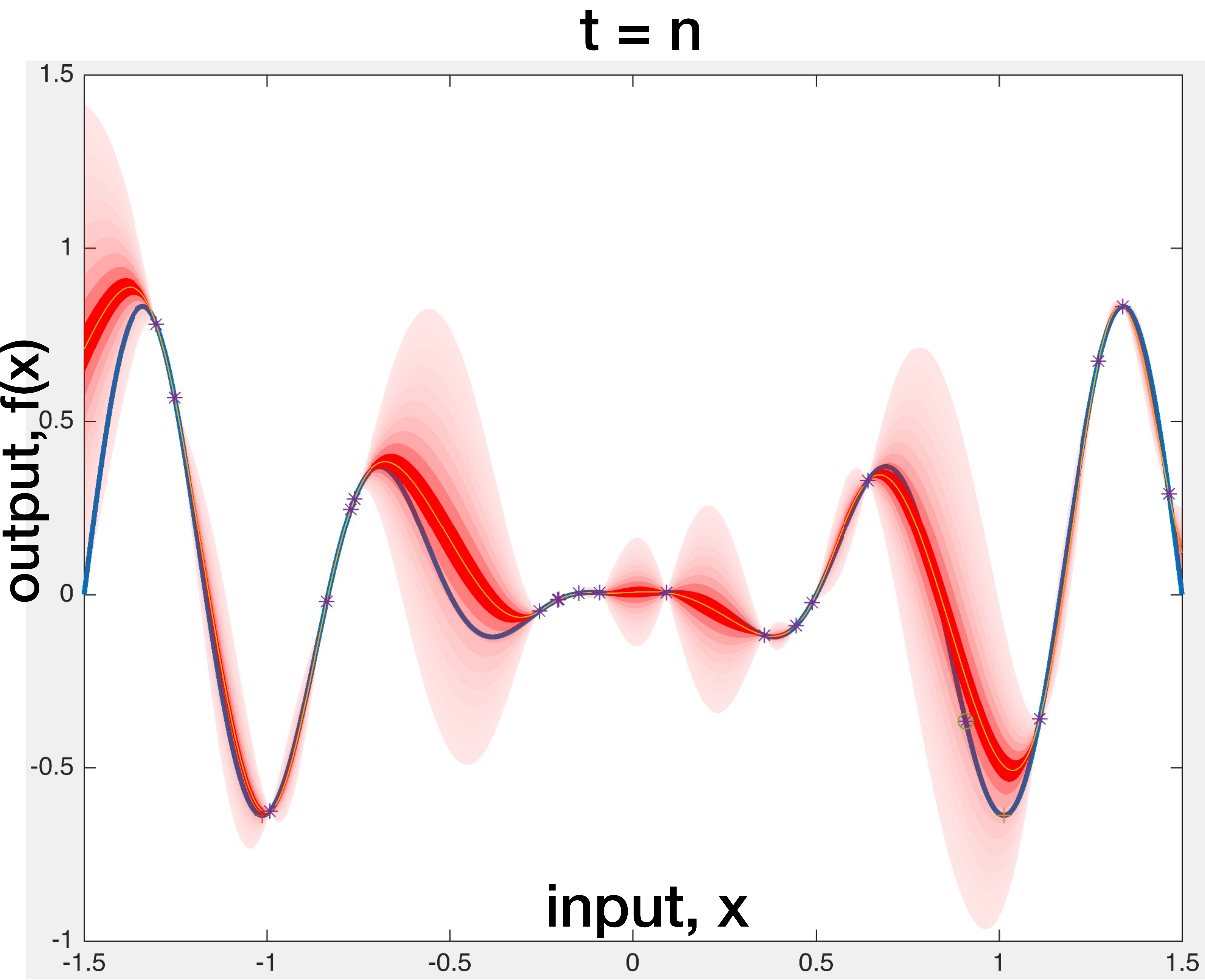
Measure

Data

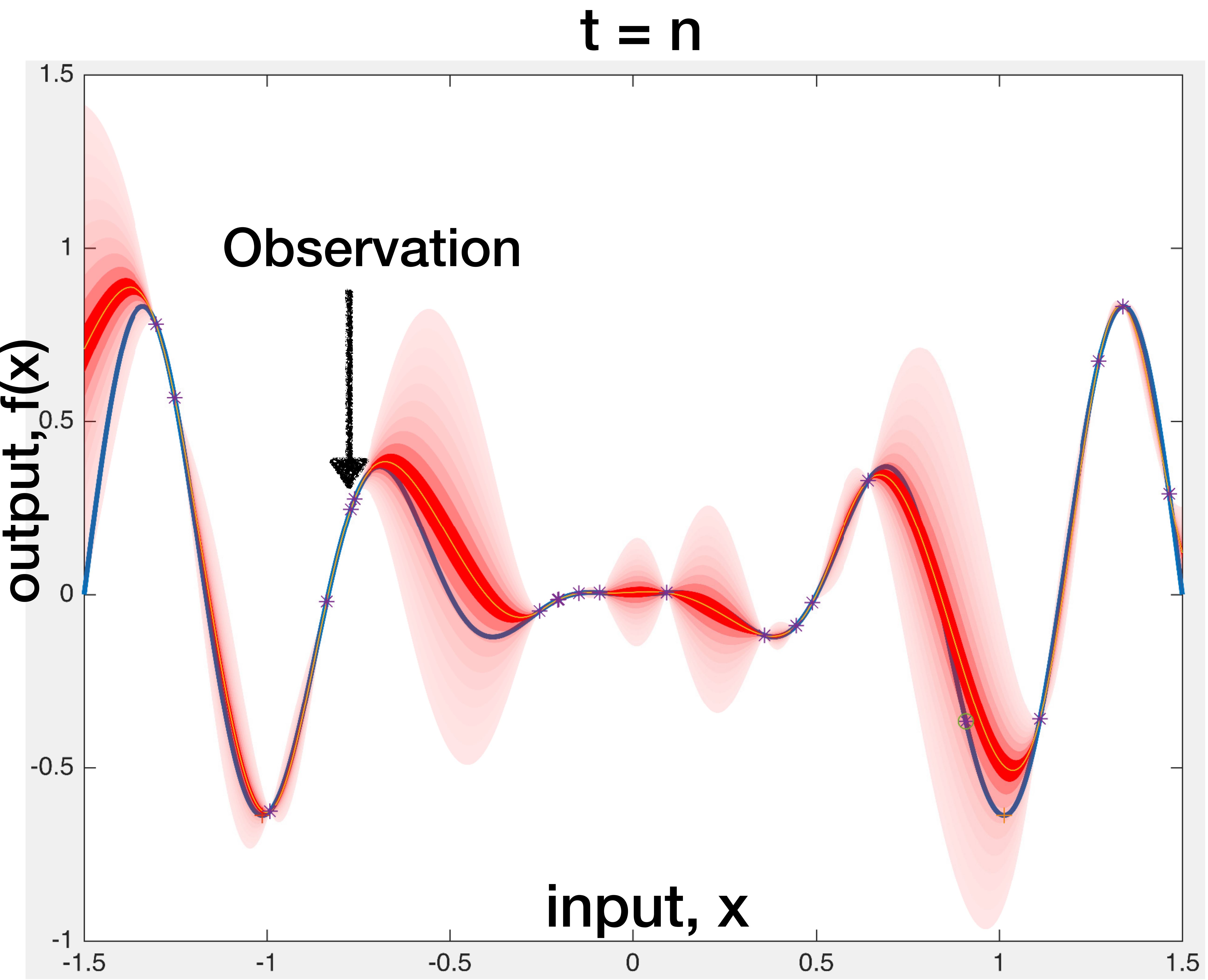


[P. Jamshidi, et al., “**Transfer learning for improving model predictions** ….”, SEAMS’17]

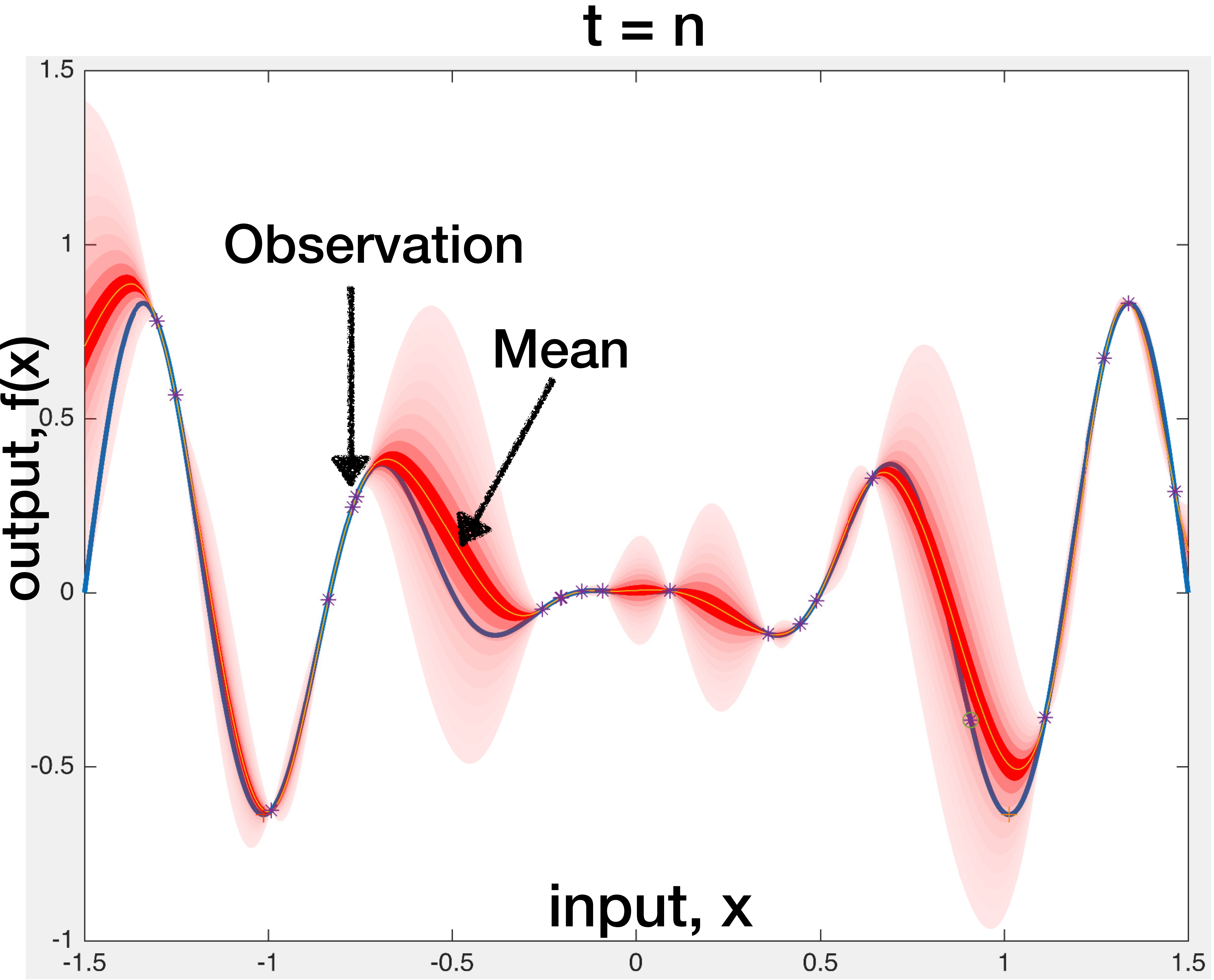
Gaussian processes for performance modeling



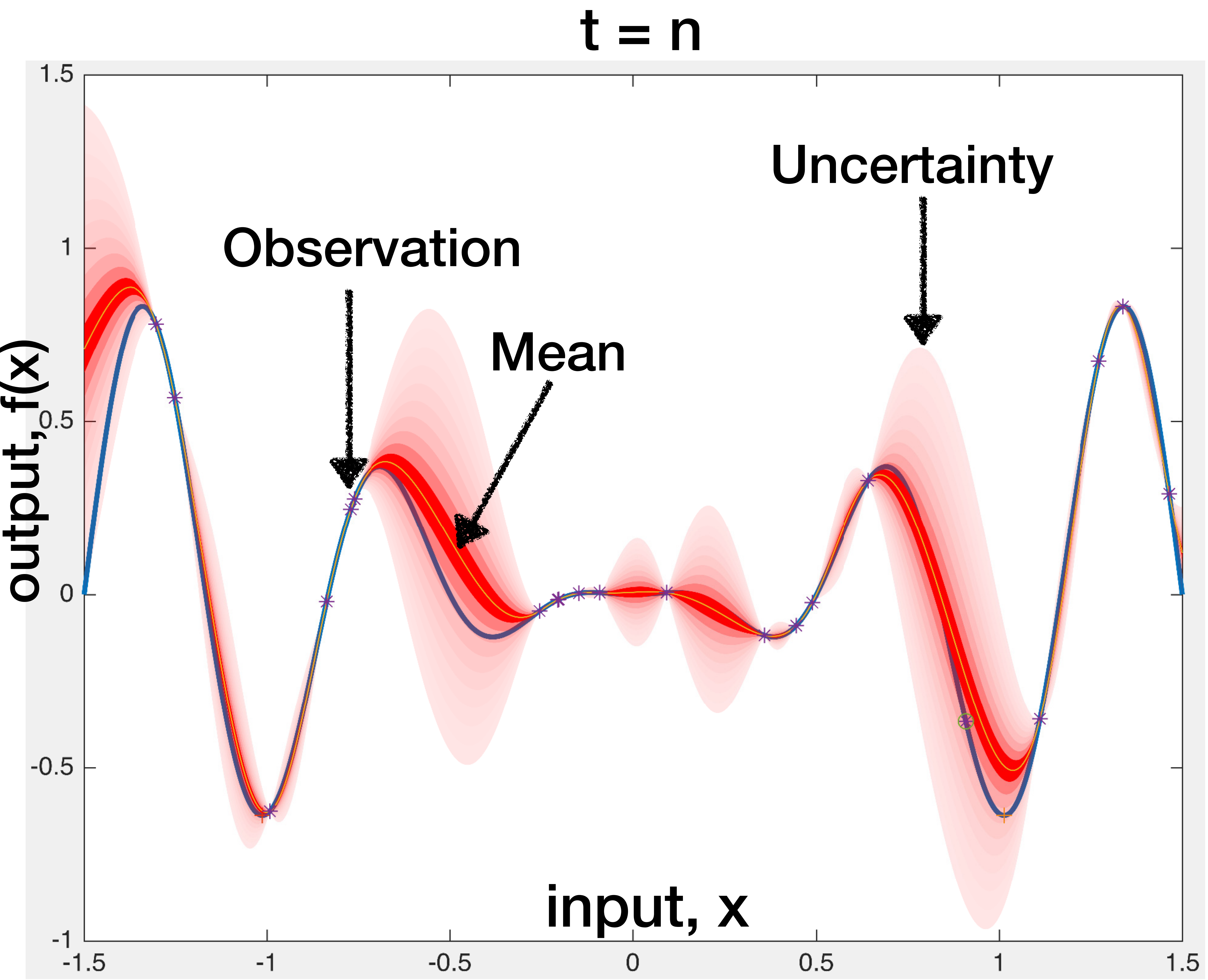
Gaussian processes for performance modeling



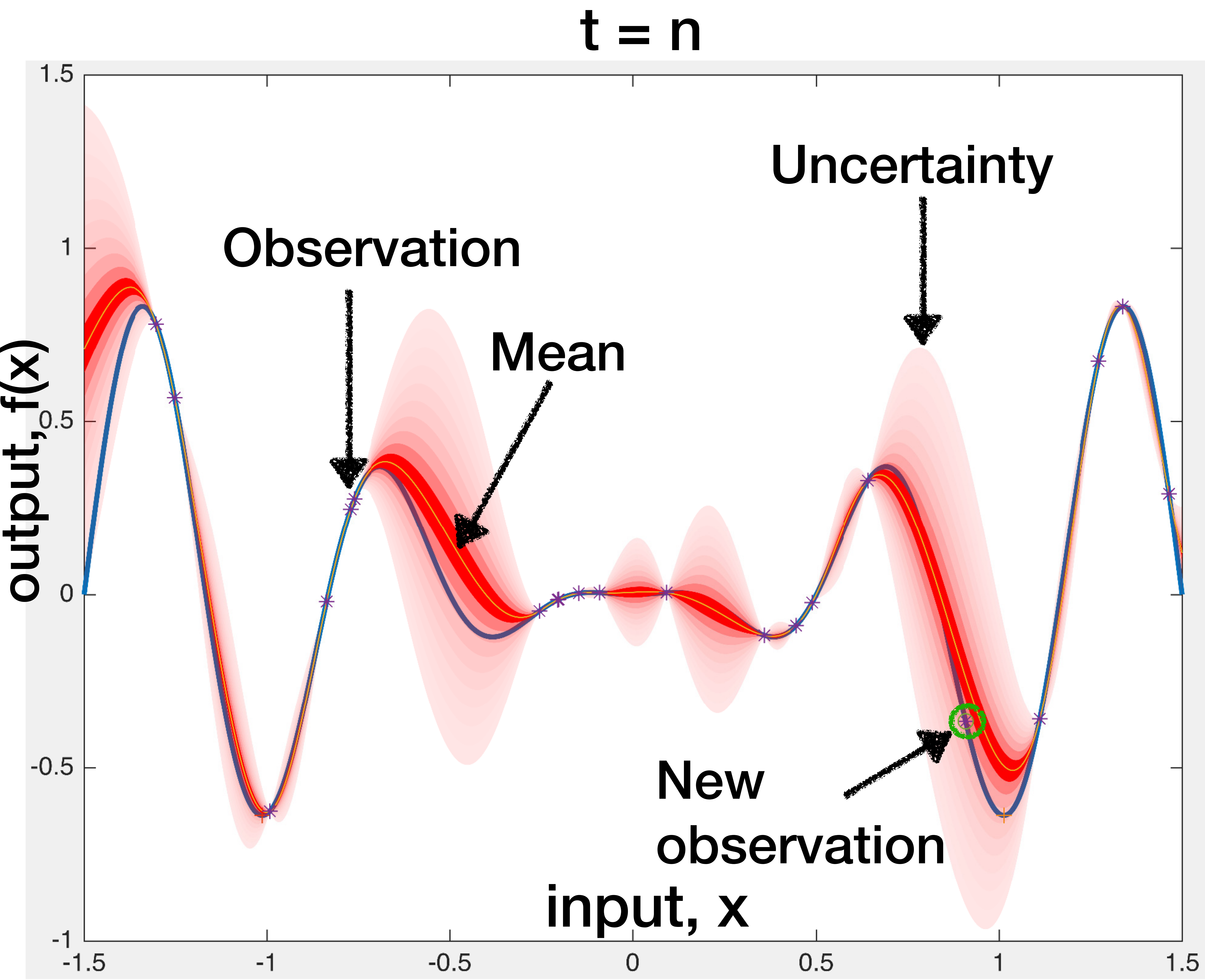
Gaussian processes for performance modeling



Gaussian processes for performance modeling



Gaussian processes for performance modeling



Gaussian processes for performance modeling

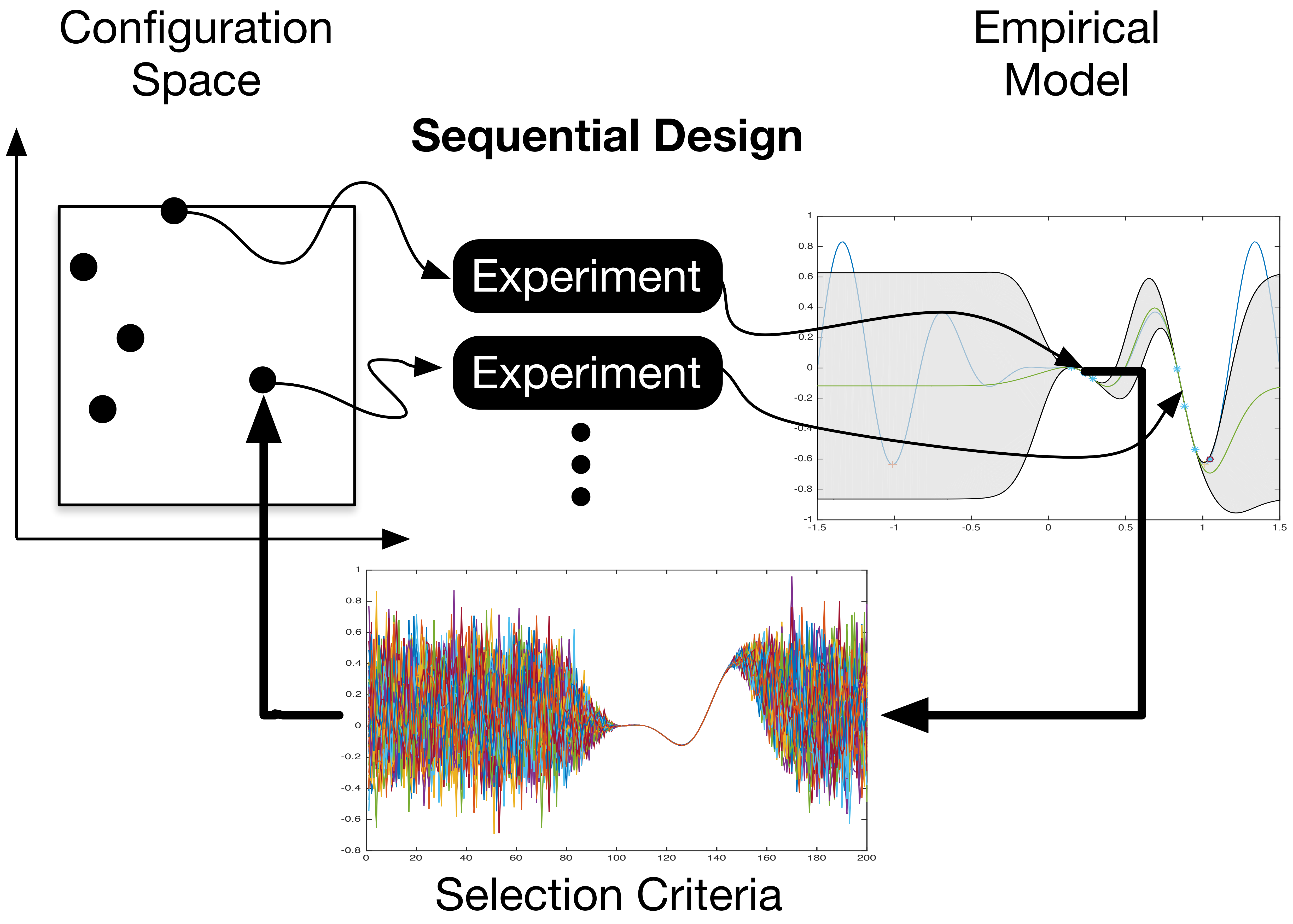
Gaussian Processes enables reasoning about performance

Step 1: Fit GP to the data seen so far

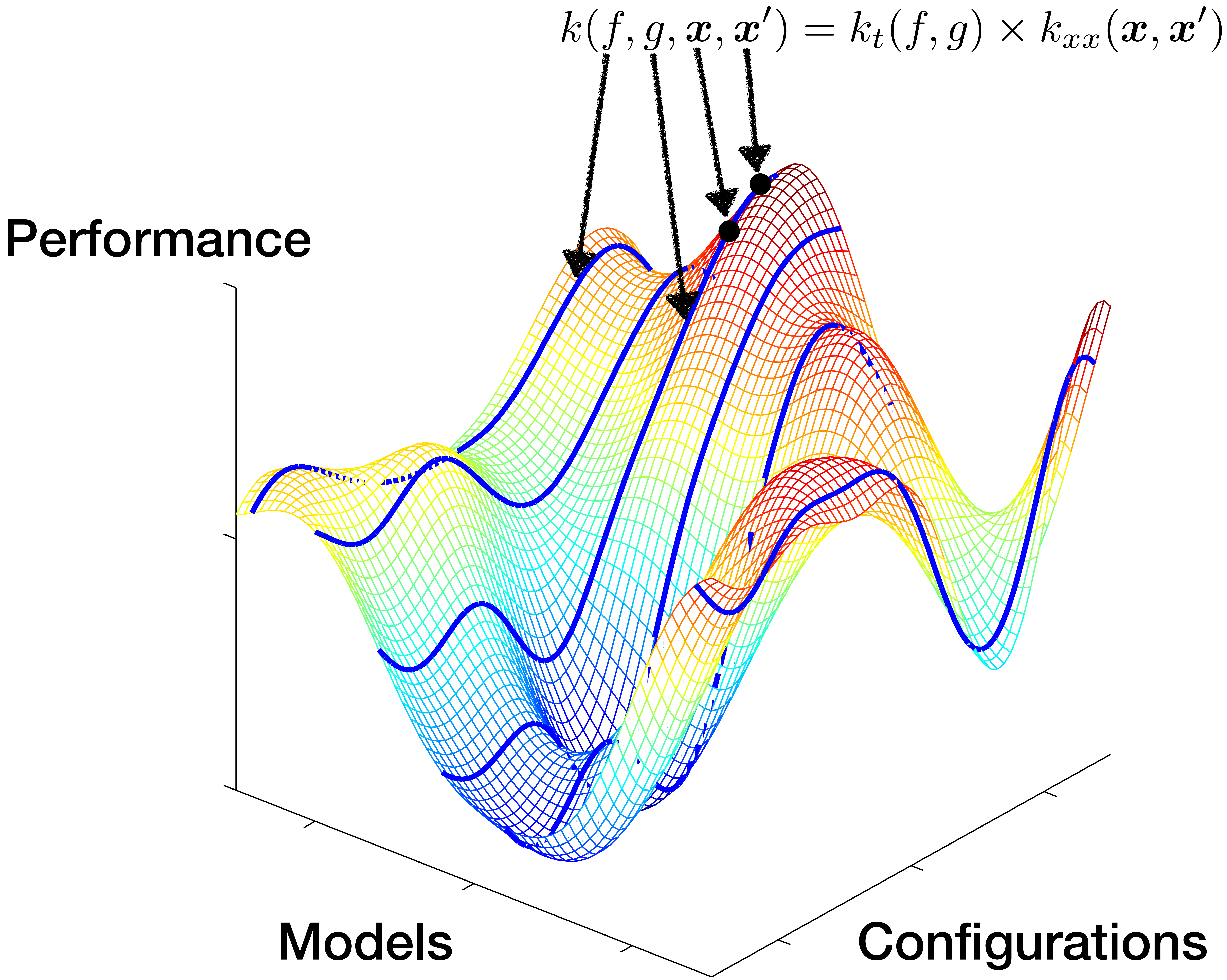
Step 2: Explore the model for regions of most variance

Step 3: Sample that region

Step 4: Repeat



The intuition behind our transfer learning approach



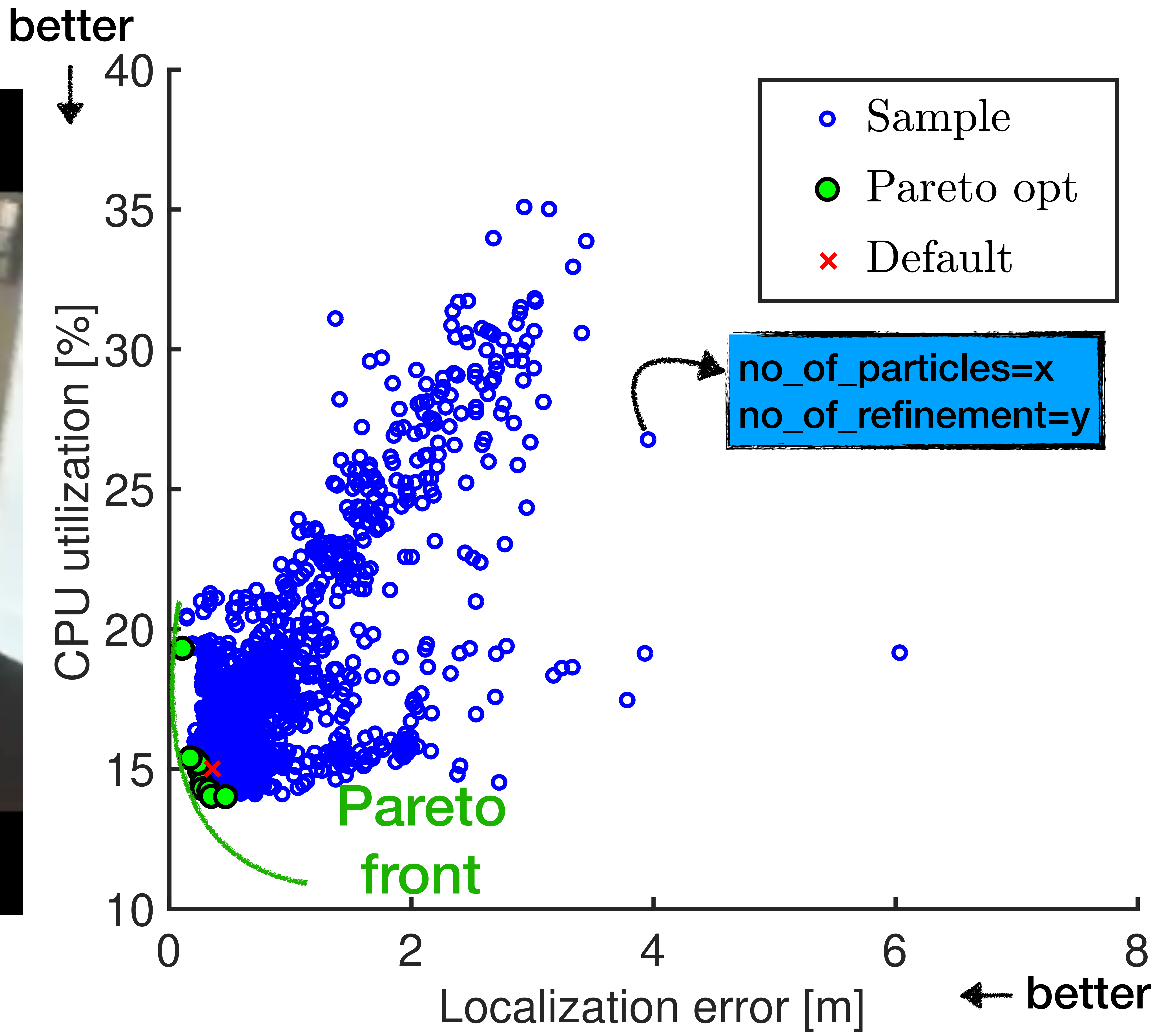
Intuition: Observations on the source(s) can affect predictions on the target

Example: Learning the chess game make learning the Go game a lot easier!

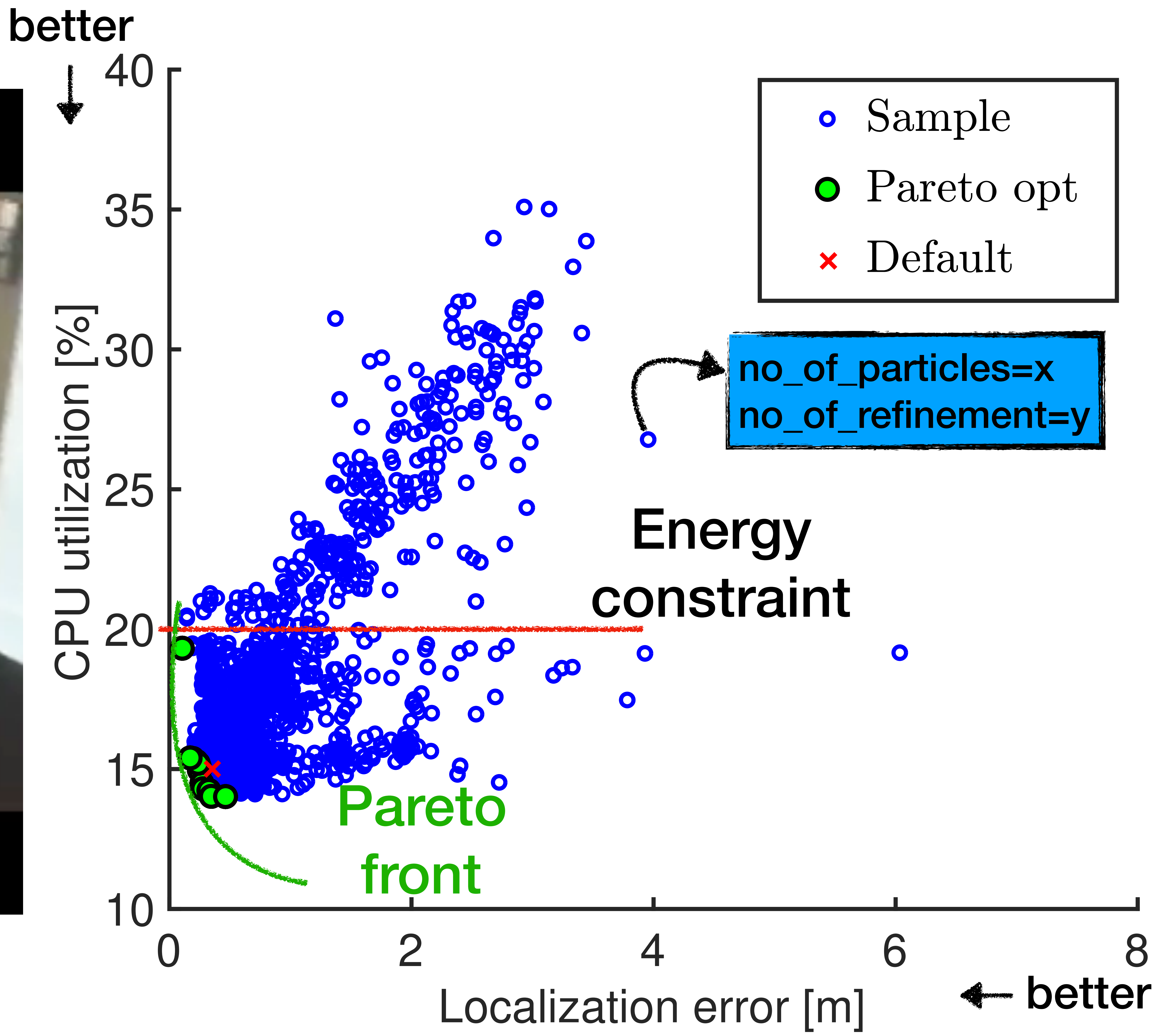
CoBot experiment: DARPA BRASS

CoBot experiment: DARPA BRASS

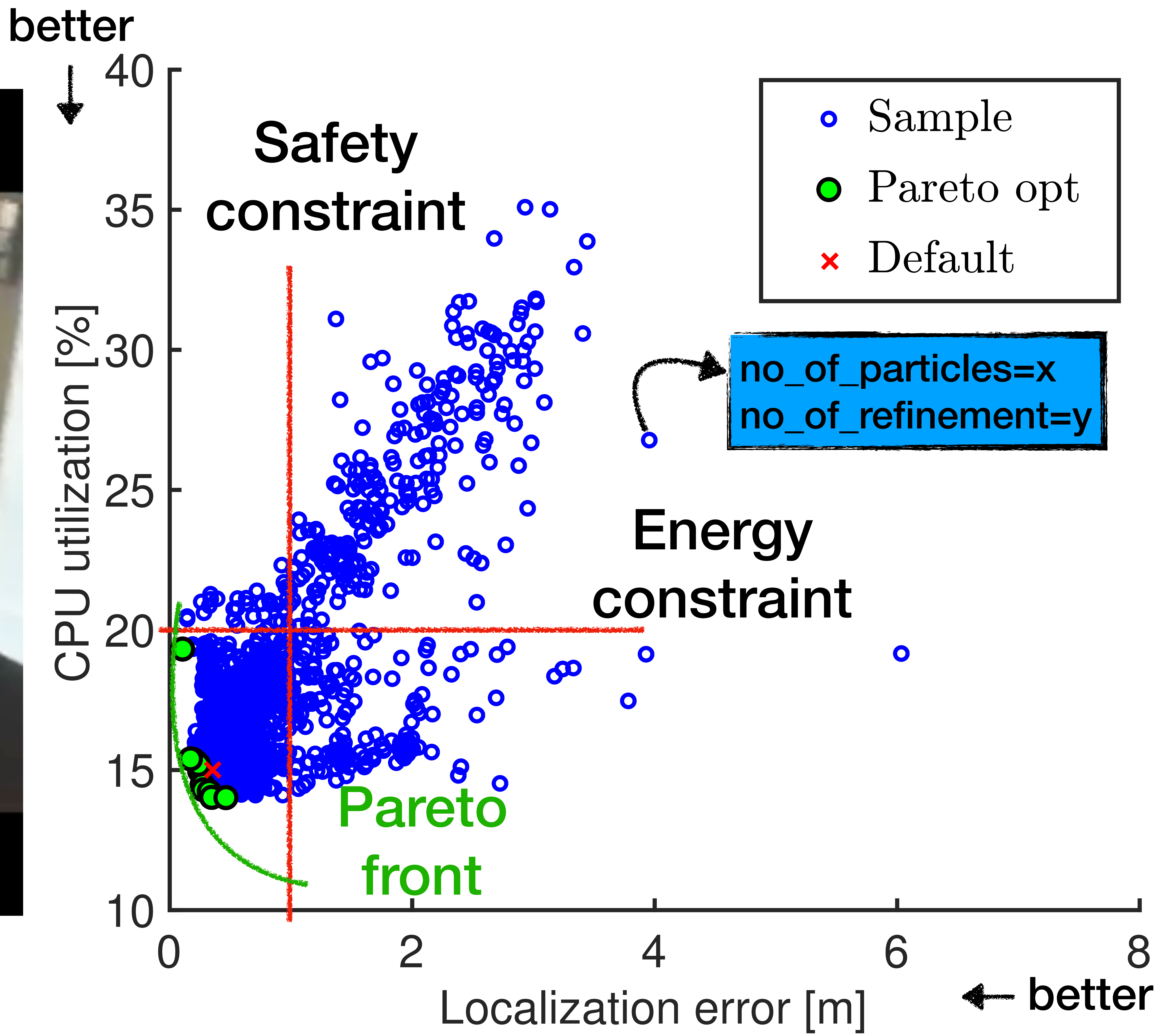
CoBot experiment: DARPA BRASS



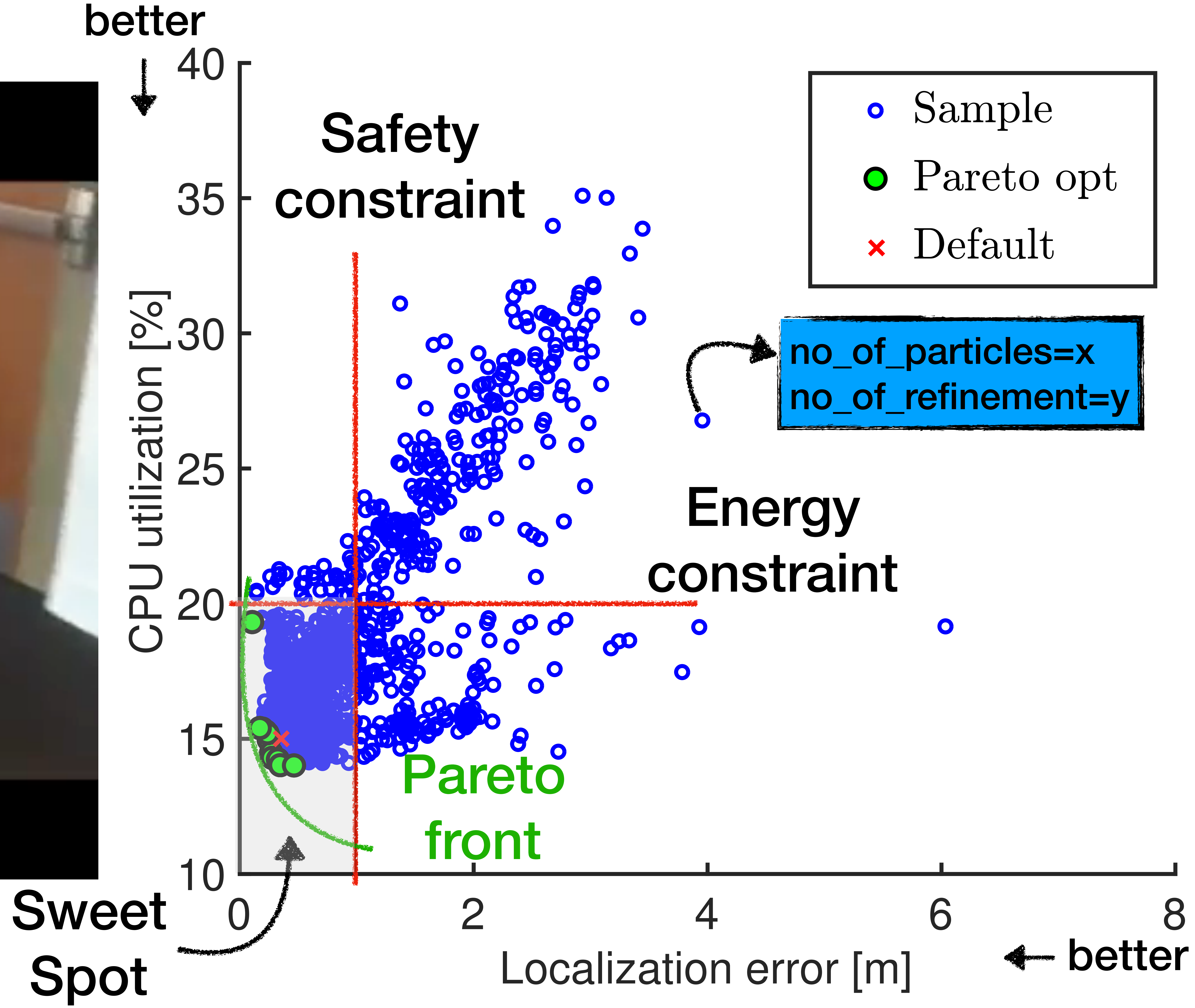
CoBot experiment: DARPA BRASS



CoBot experiment: DARPA BRASS

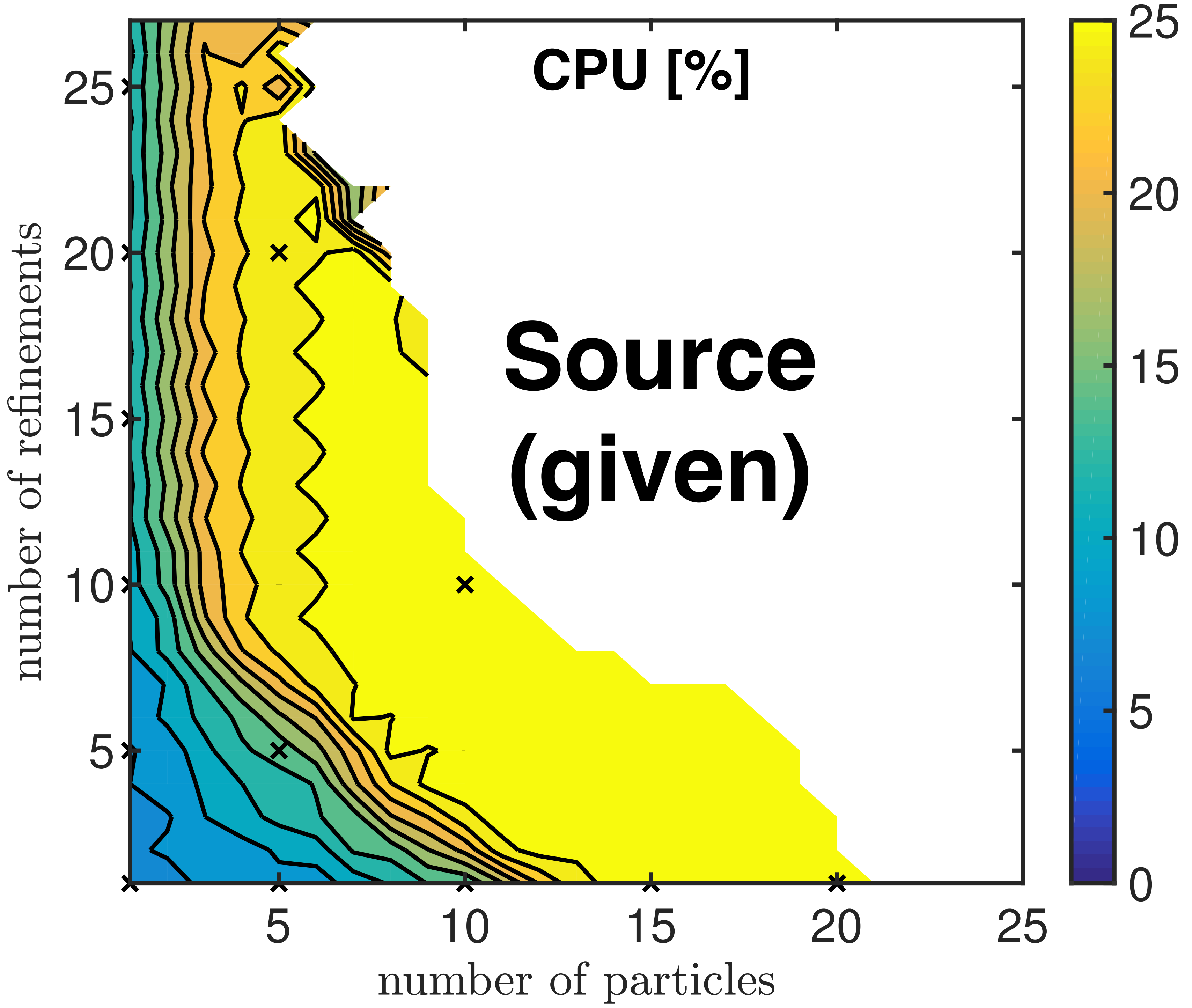


CoBot experiment: DARPA BRASS

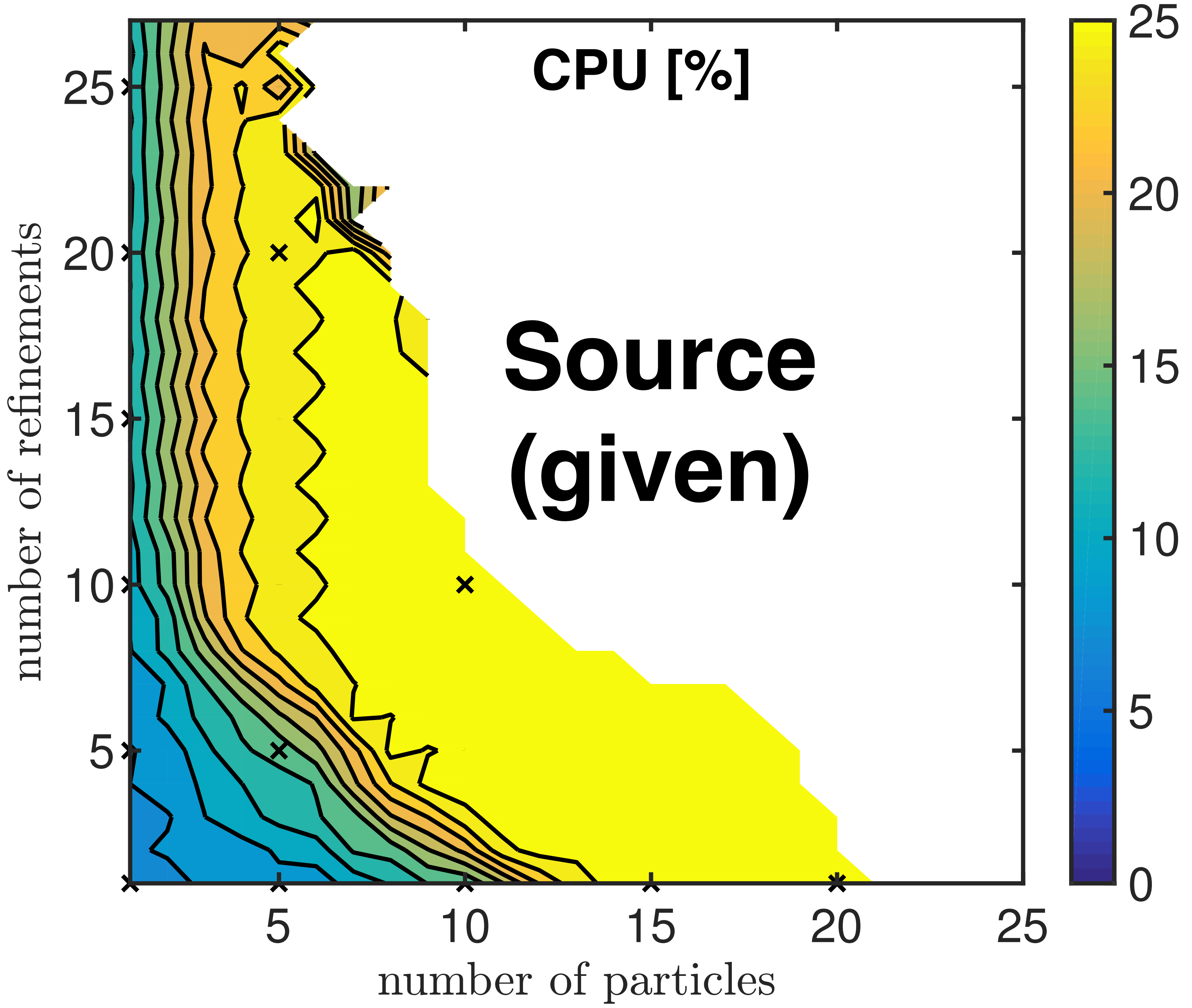


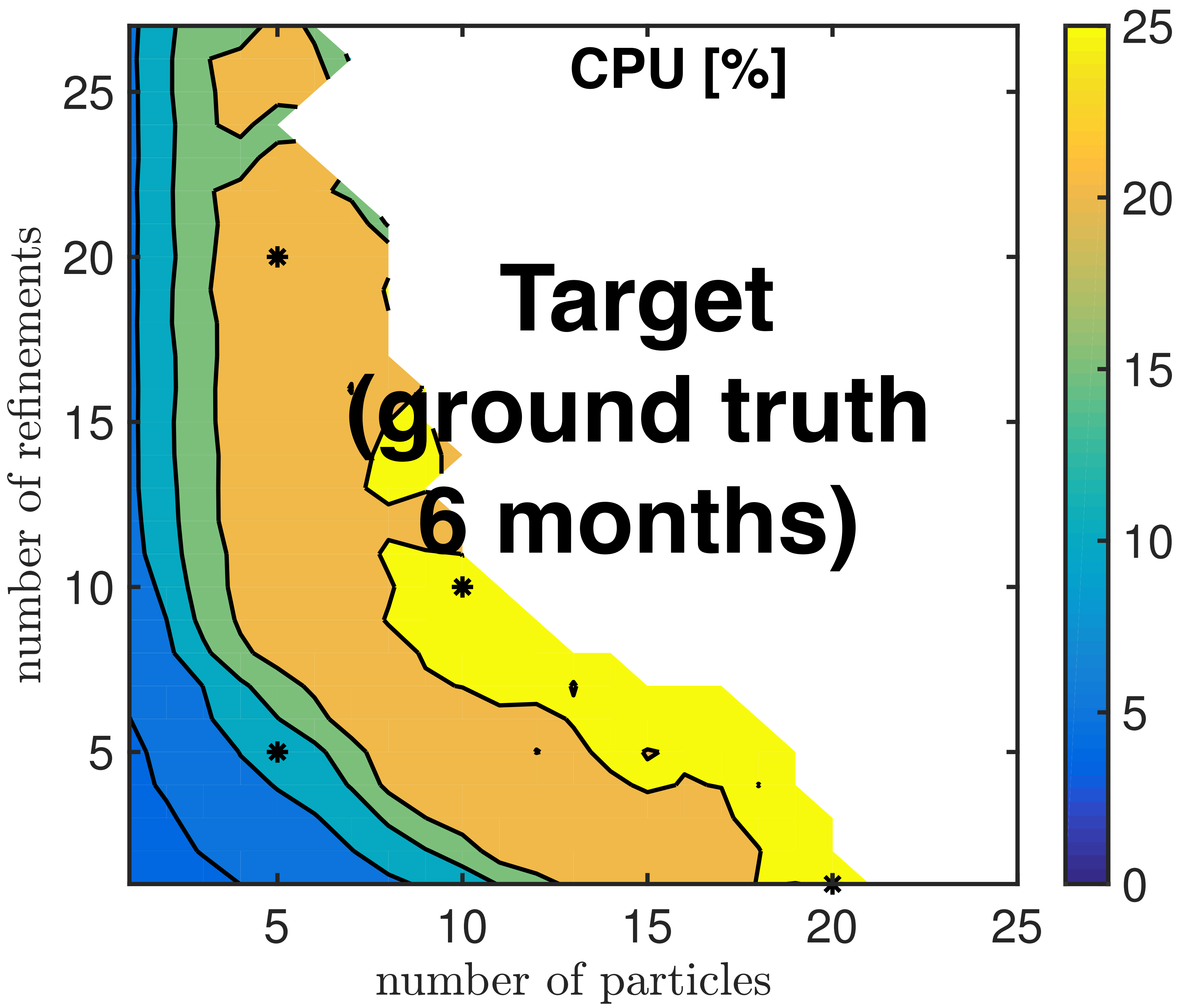
CoBot experiment

CoBot experiment

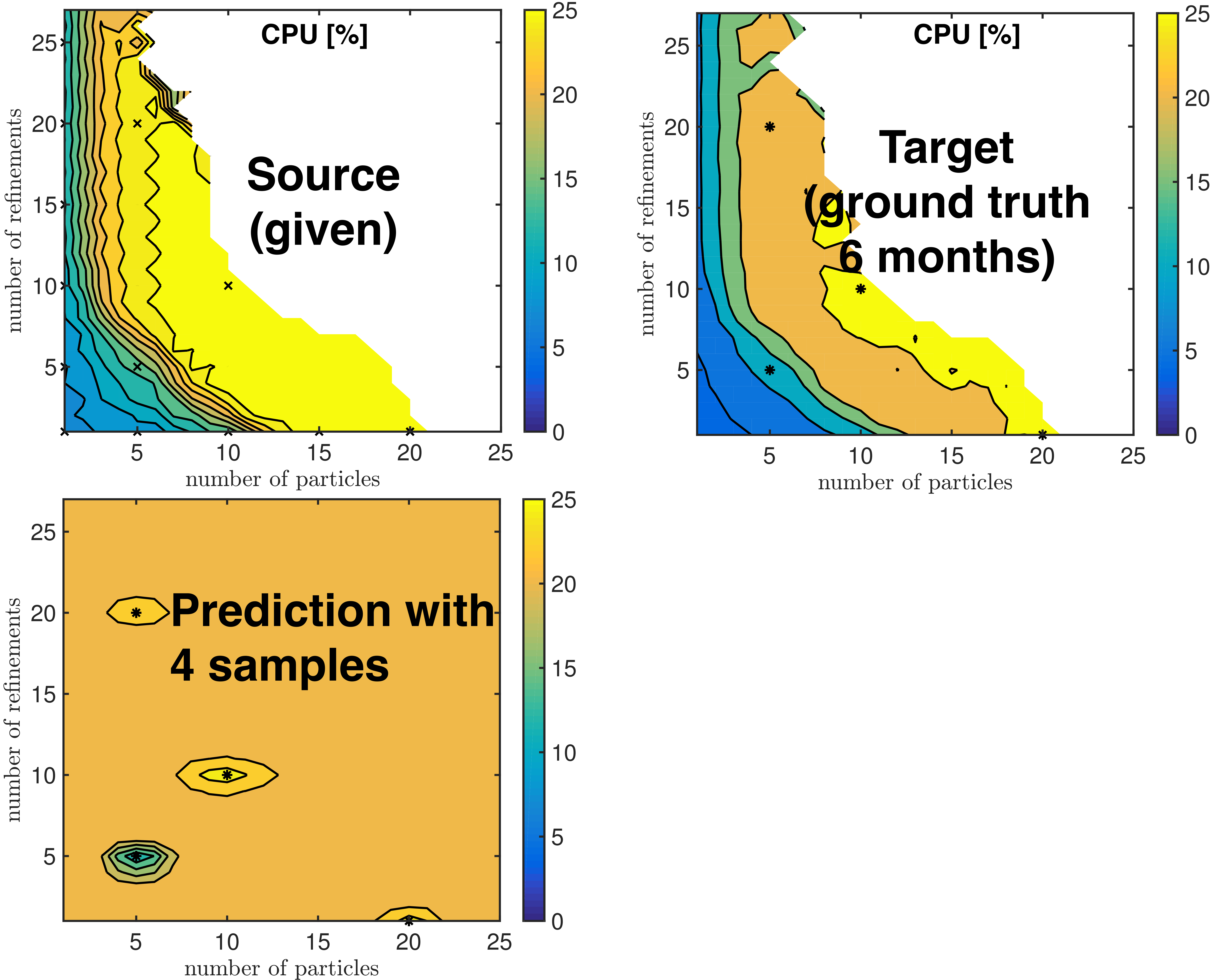


CoBot experiment

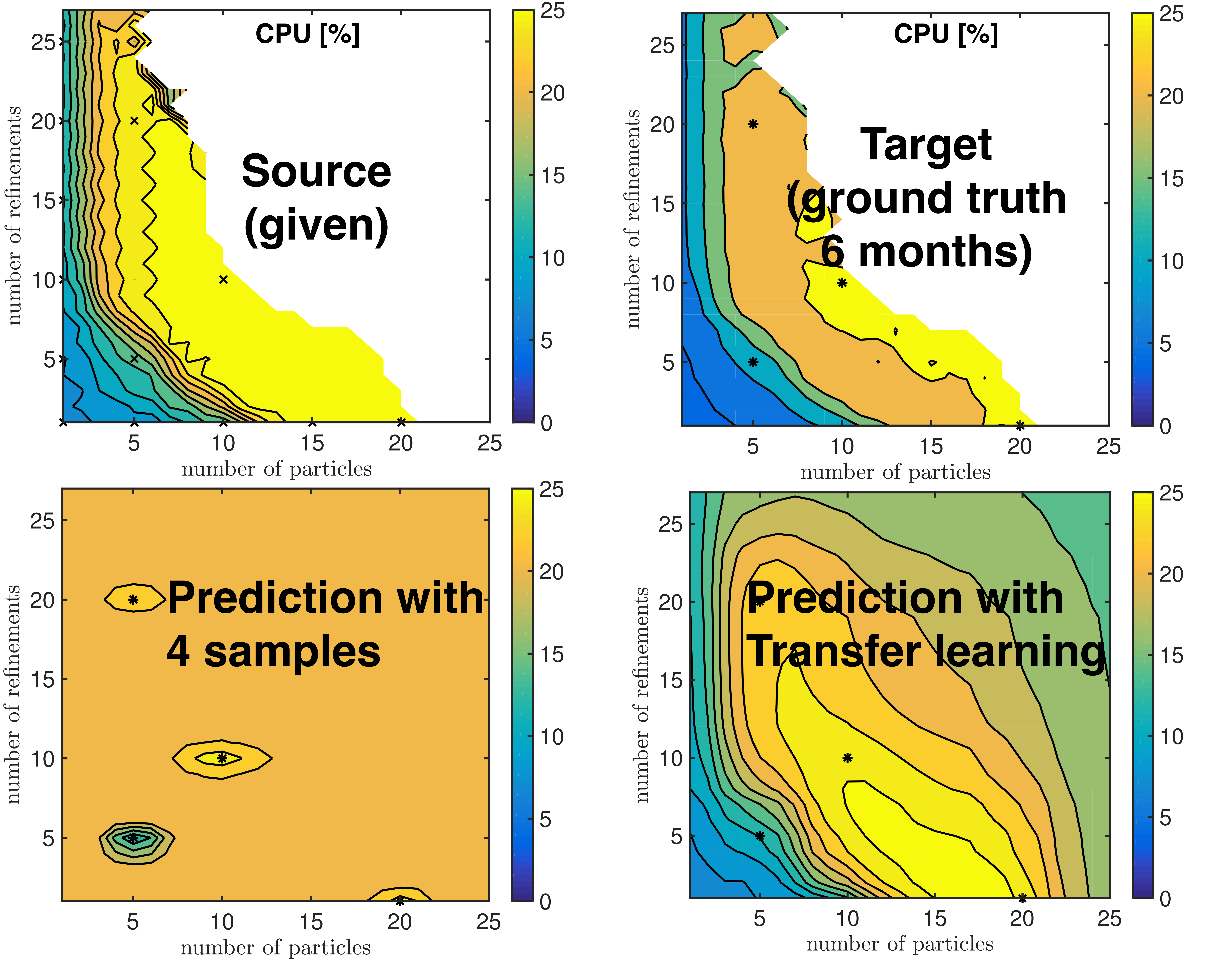




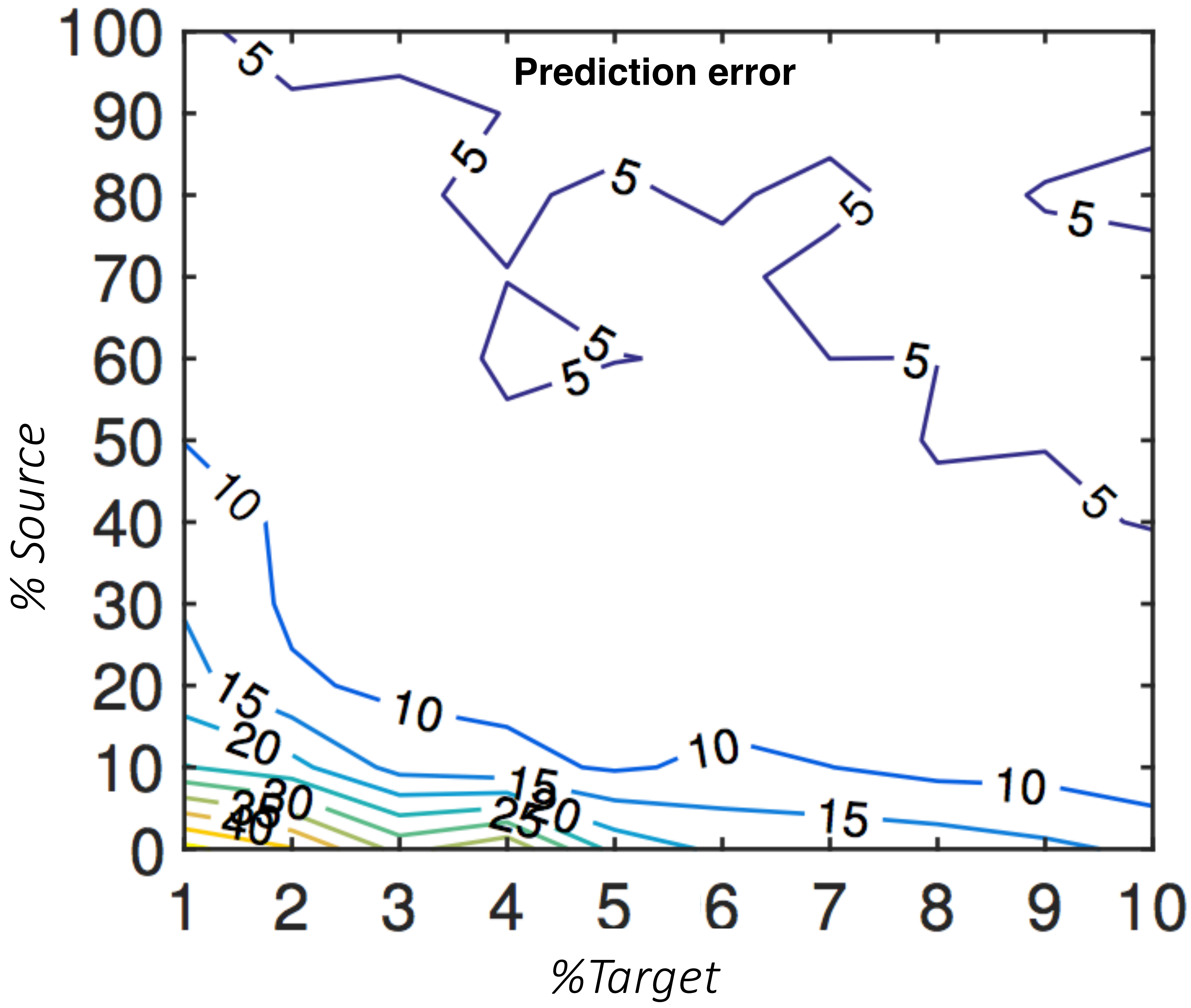
CoBot experiment



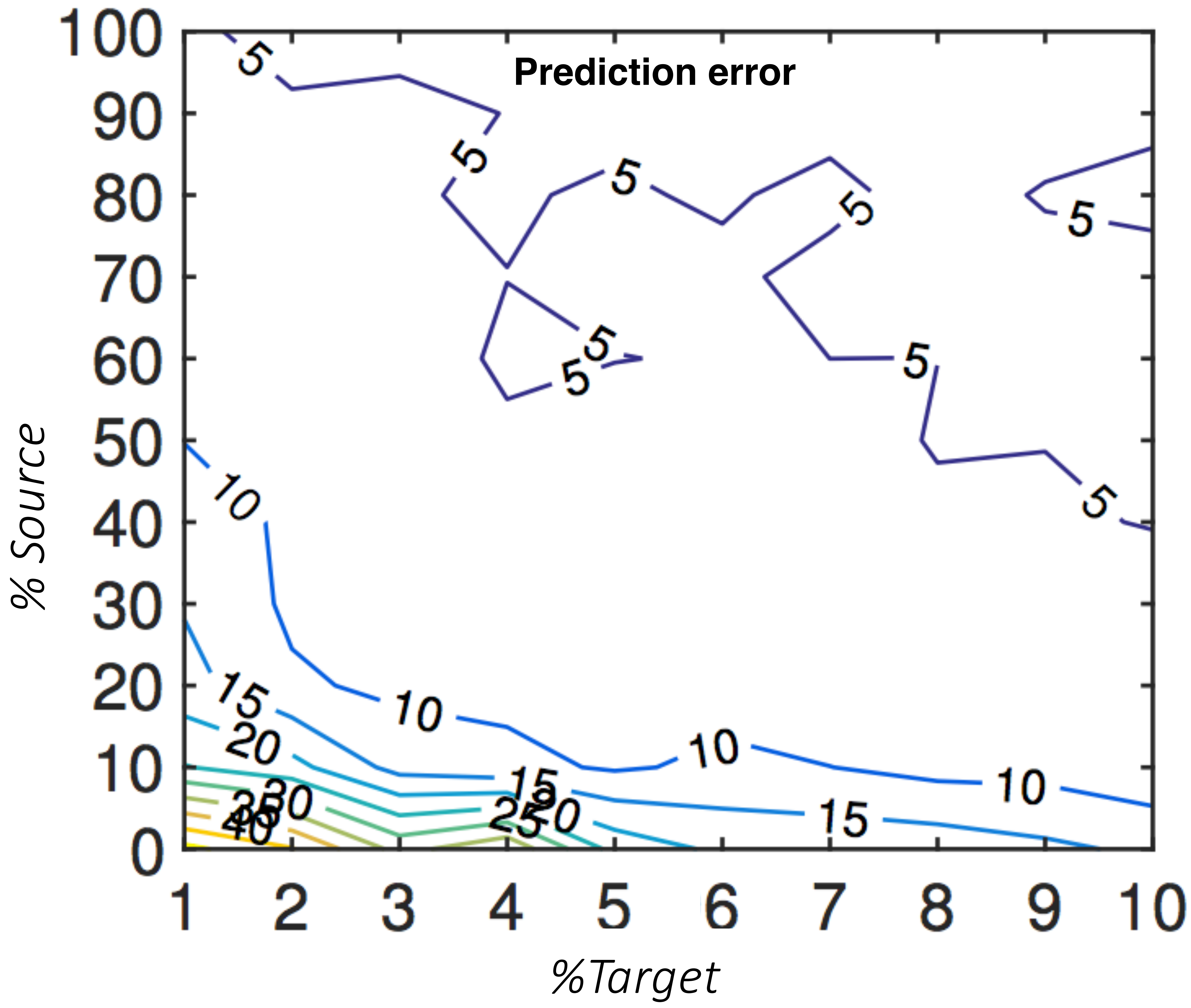
CoBot experiment

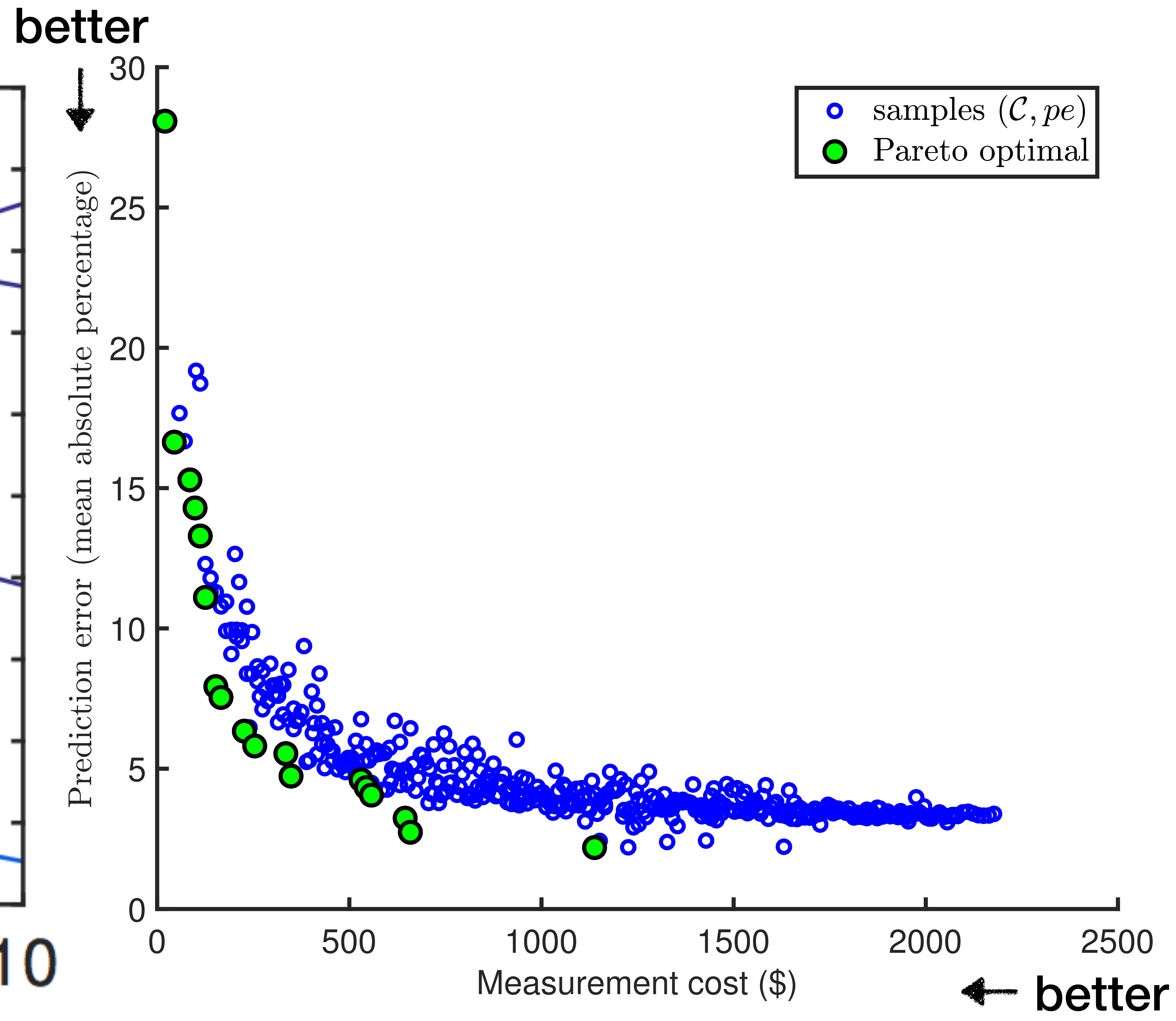


Result: CoBot experiment

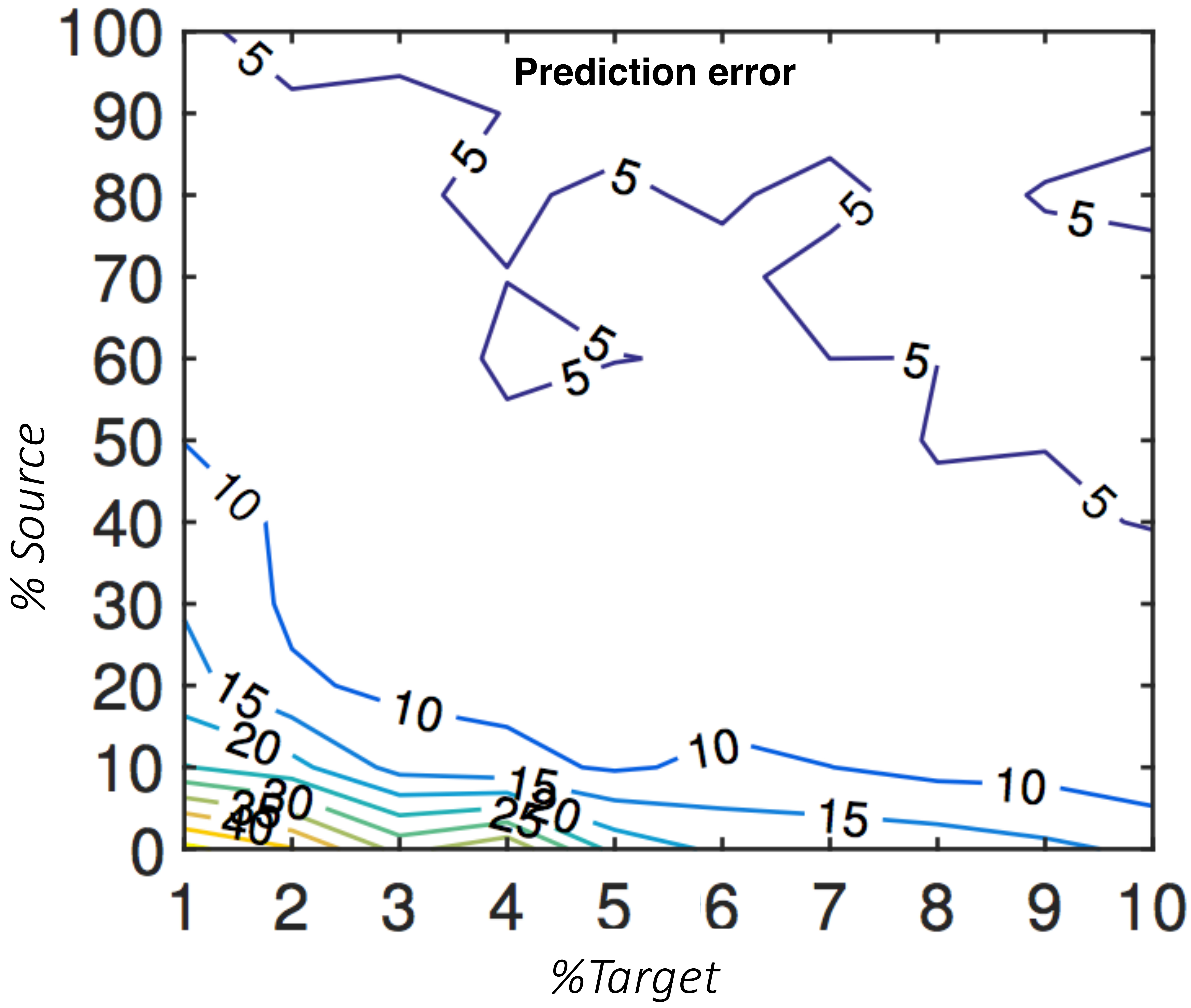


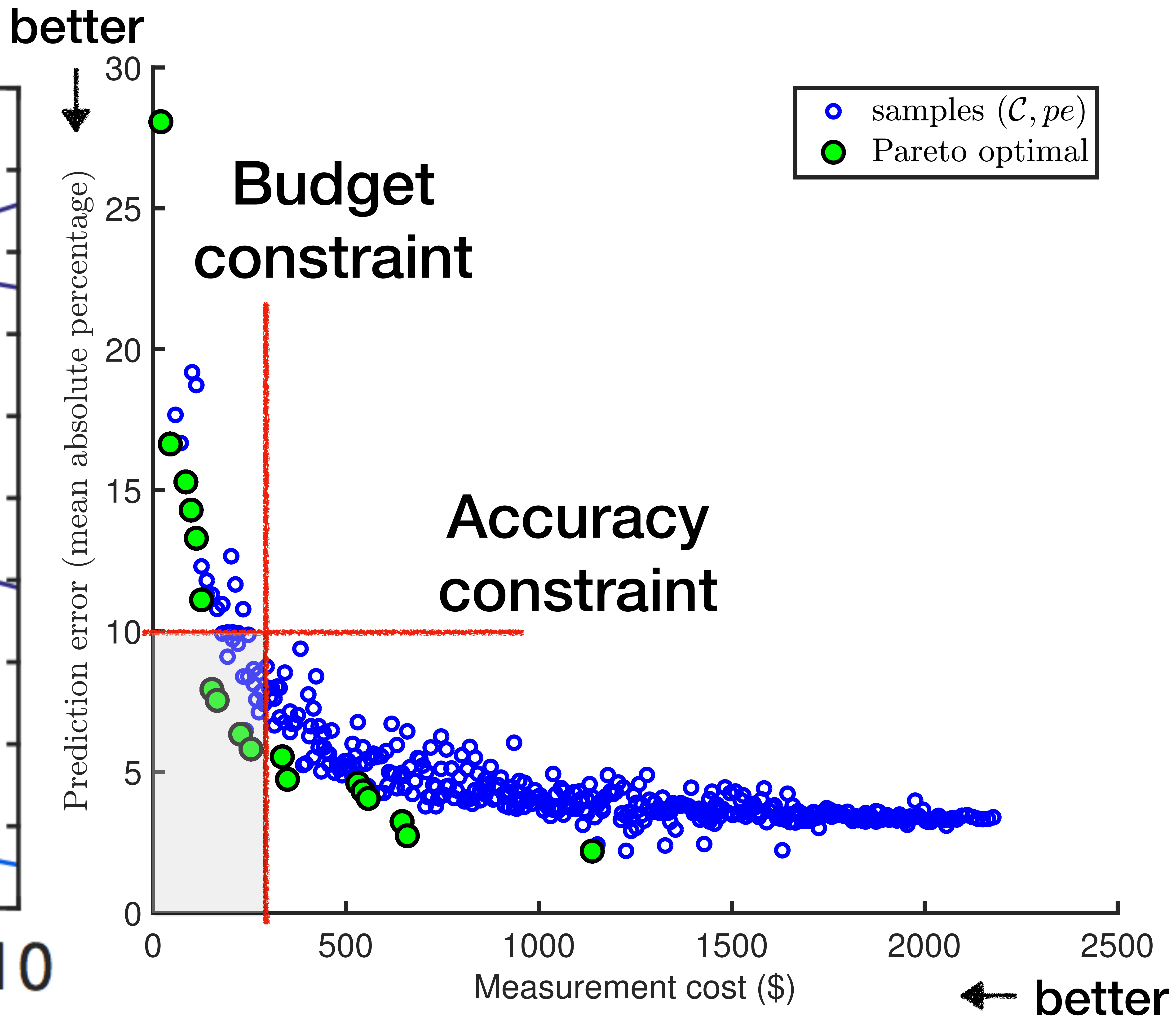
Result: CoBot experiment



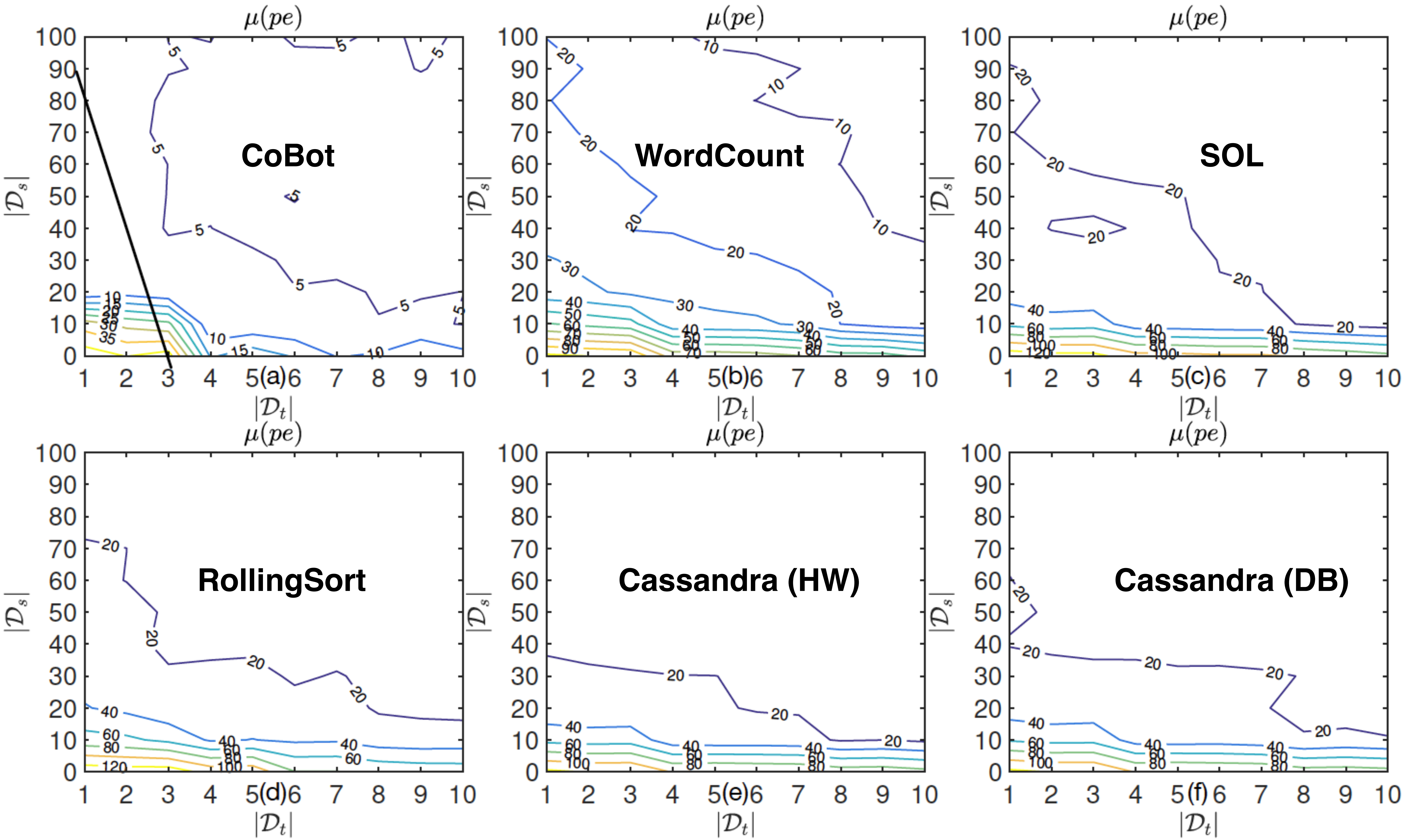


Result: CoBot experiment





Results: Other configurable systems



Details: [SEAMS '17]

Transfer Learning for Improving Model Predictions in Highly Configurable Software

Pooyan Jamshidi, Miguel Velez, Christian Kästner
Carnegie Mellon University, USA
{pjamshid,mvelezce,kaestner}@cs.cmu.edu

Norbert Siegmund
Bauhaus-University Weimar, Germany
norbert.siegmund@uni-weimar.de

Prasad Kawthekar
Stanford University, USA
pkawthek@stanford.edu

***Abstract*—Modern software systems are built to be used in dynamic environments using configuration capabilities to adapt to changes and external uncertainties. In a self-adaptation context, we are often interested in reasoning about the performance of the systems under different configurations. Usually, we learn a black-box model based on real measurements to predict the performance of the system given a specific configuration. However, as modern systems become more complex, there are many configuration parameters that may interact and we end up learning an exponentially large configuration space. Naturally, this does not scale when relying on real measurements in the actual changing environment. We propose a different solution:**

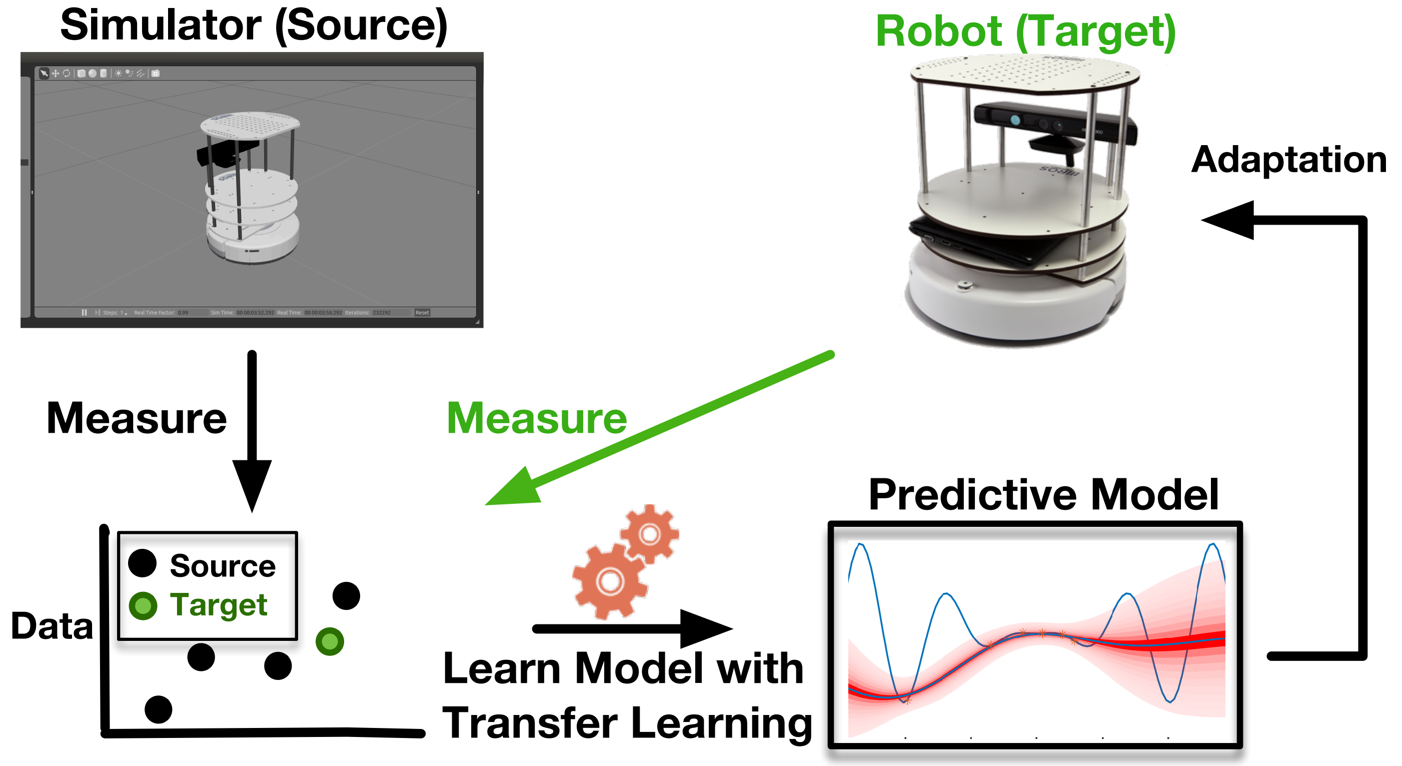
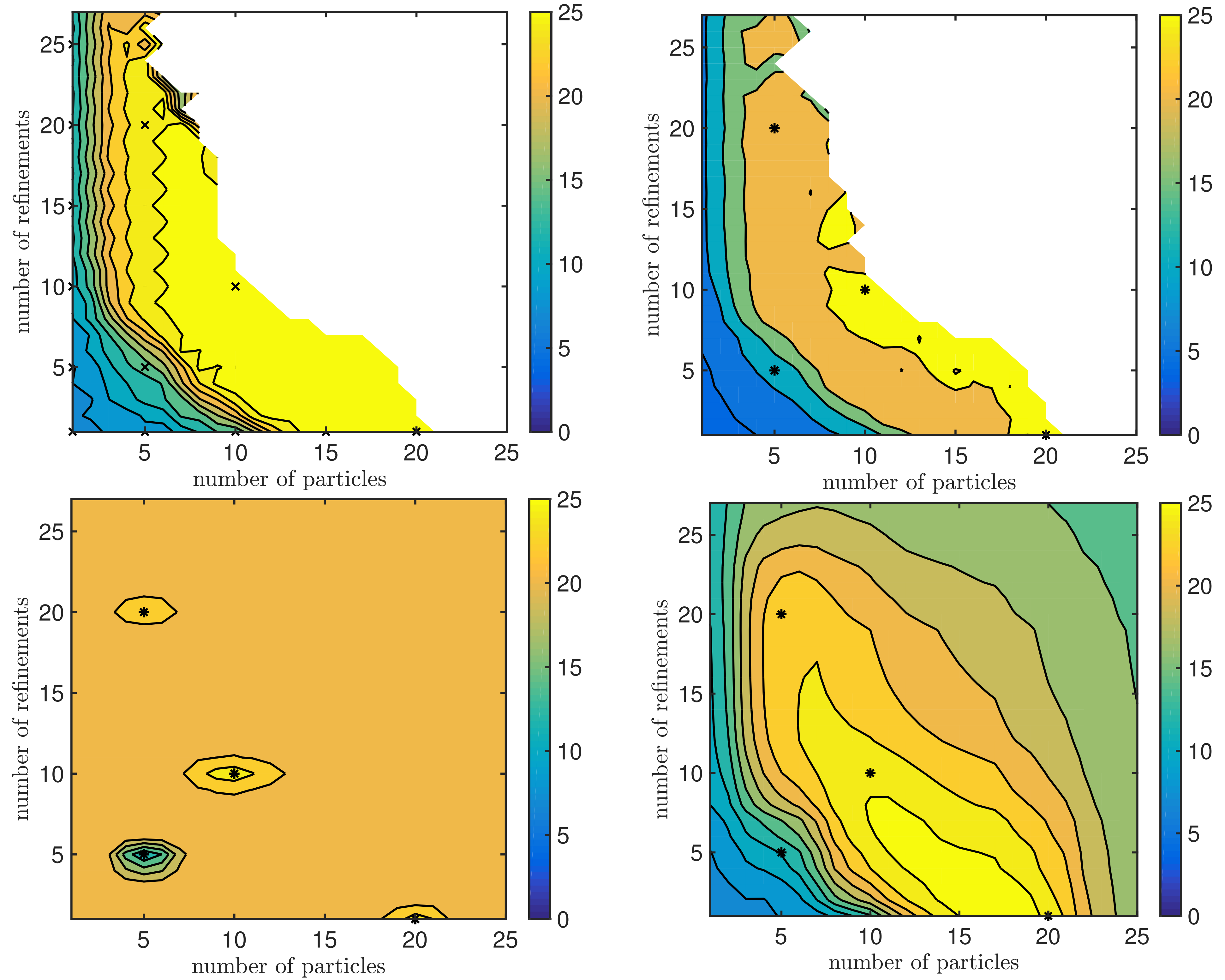


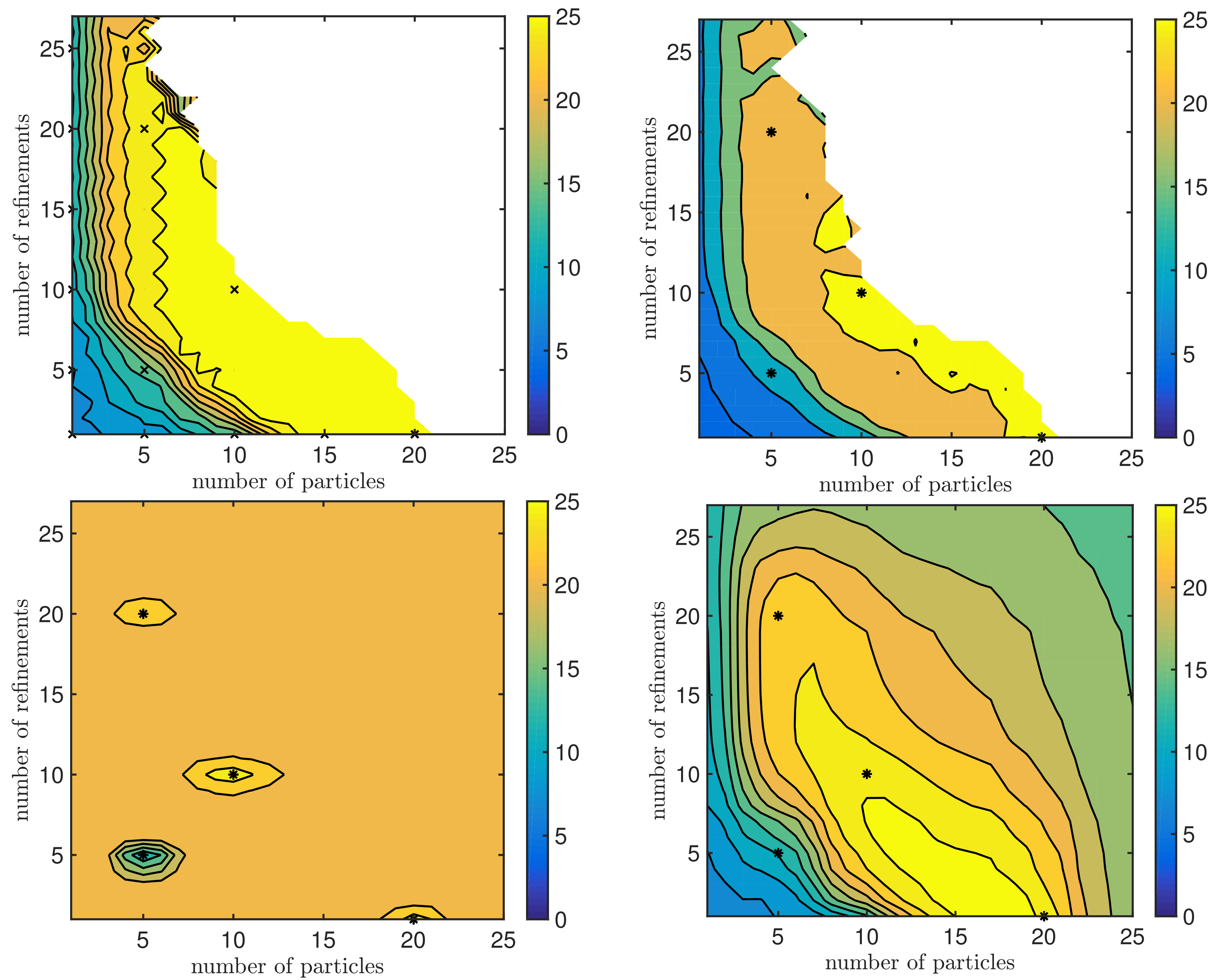
Fig. 1: Transfer learning for performance model learning.

Summary (transfer learning)



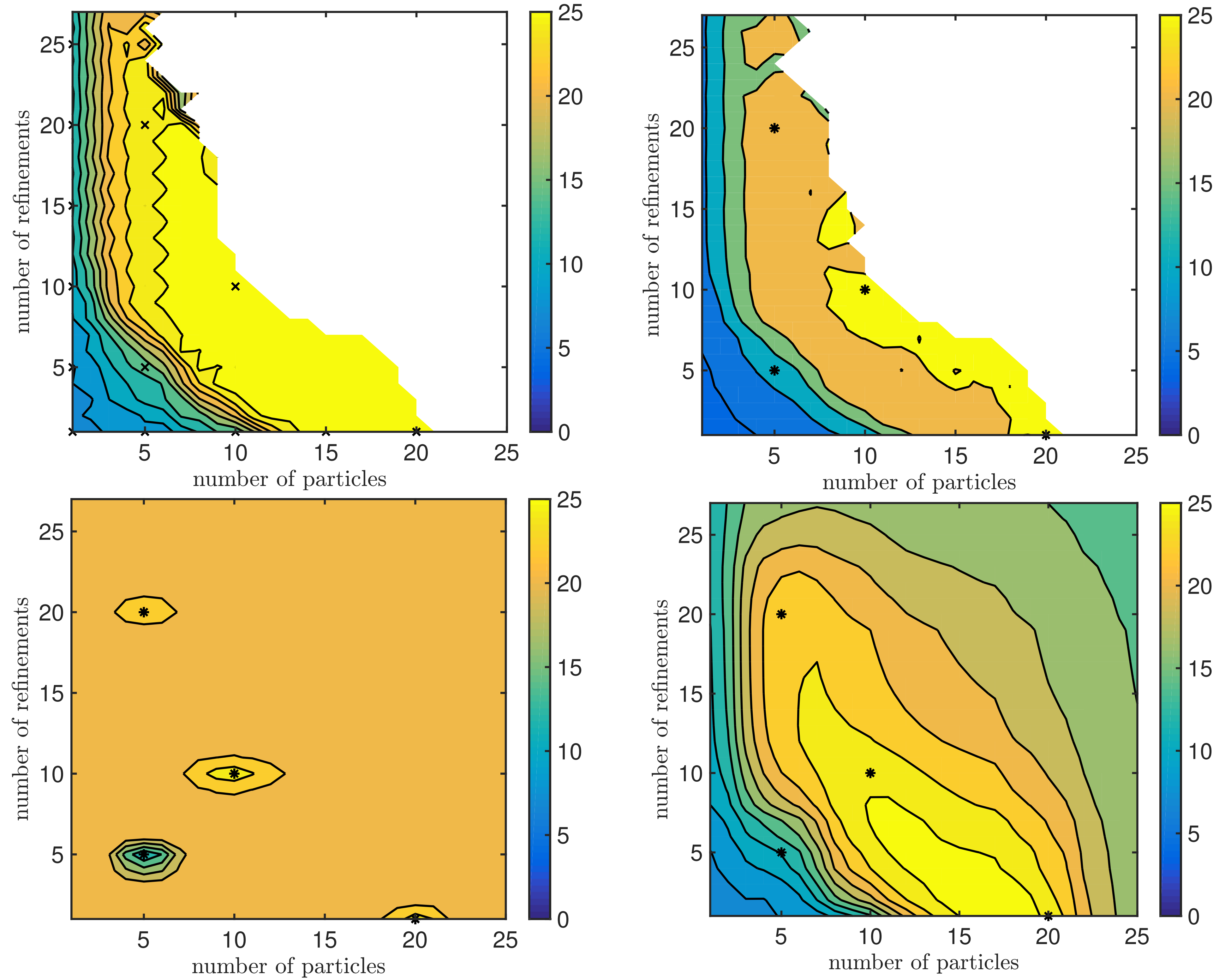
Summary (transfer learning)

- Model for making tradeoff between qualities



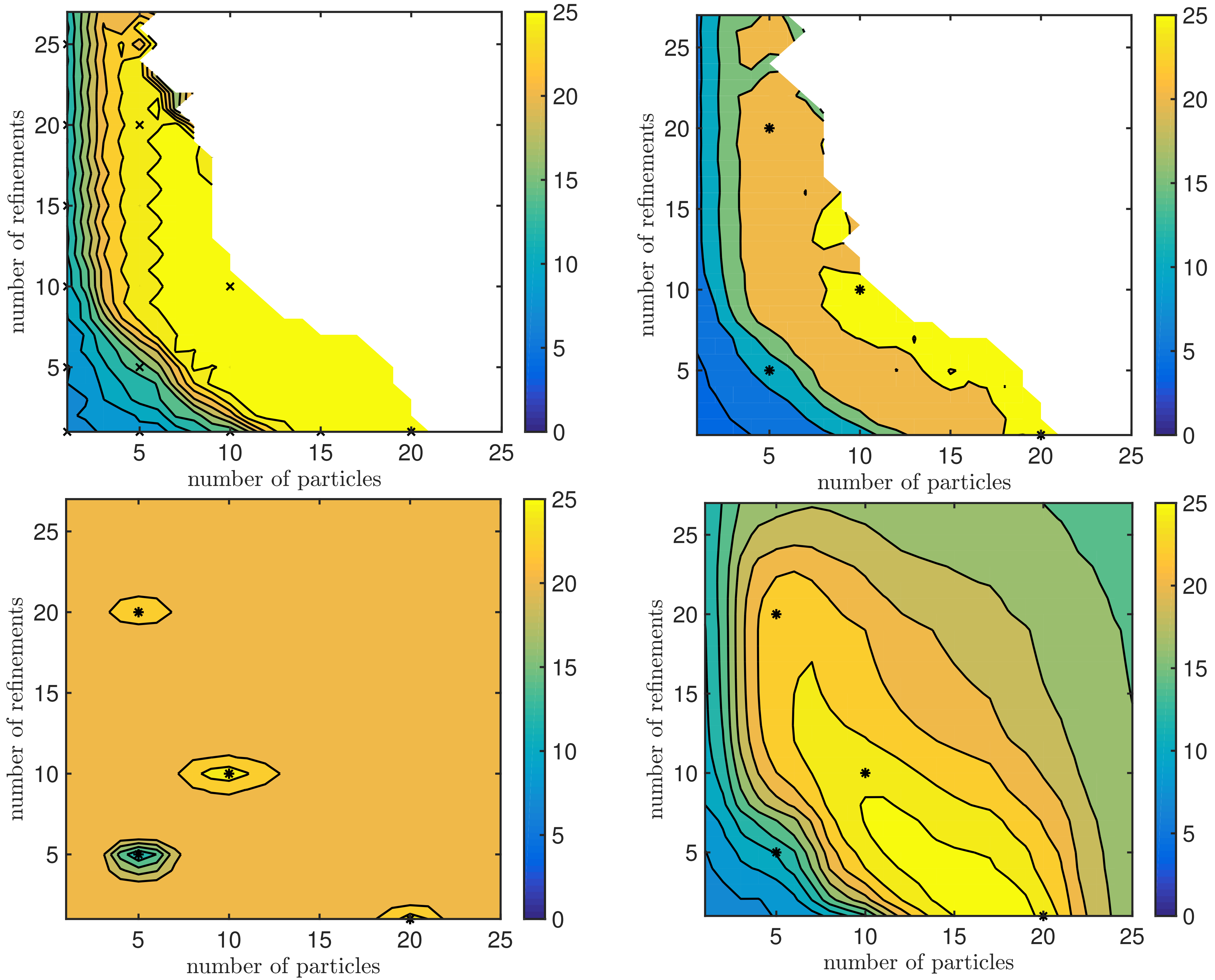
Summary (transfer learning)

- Model for making tradeoff between qualities
- Scale to large space and environmental changes



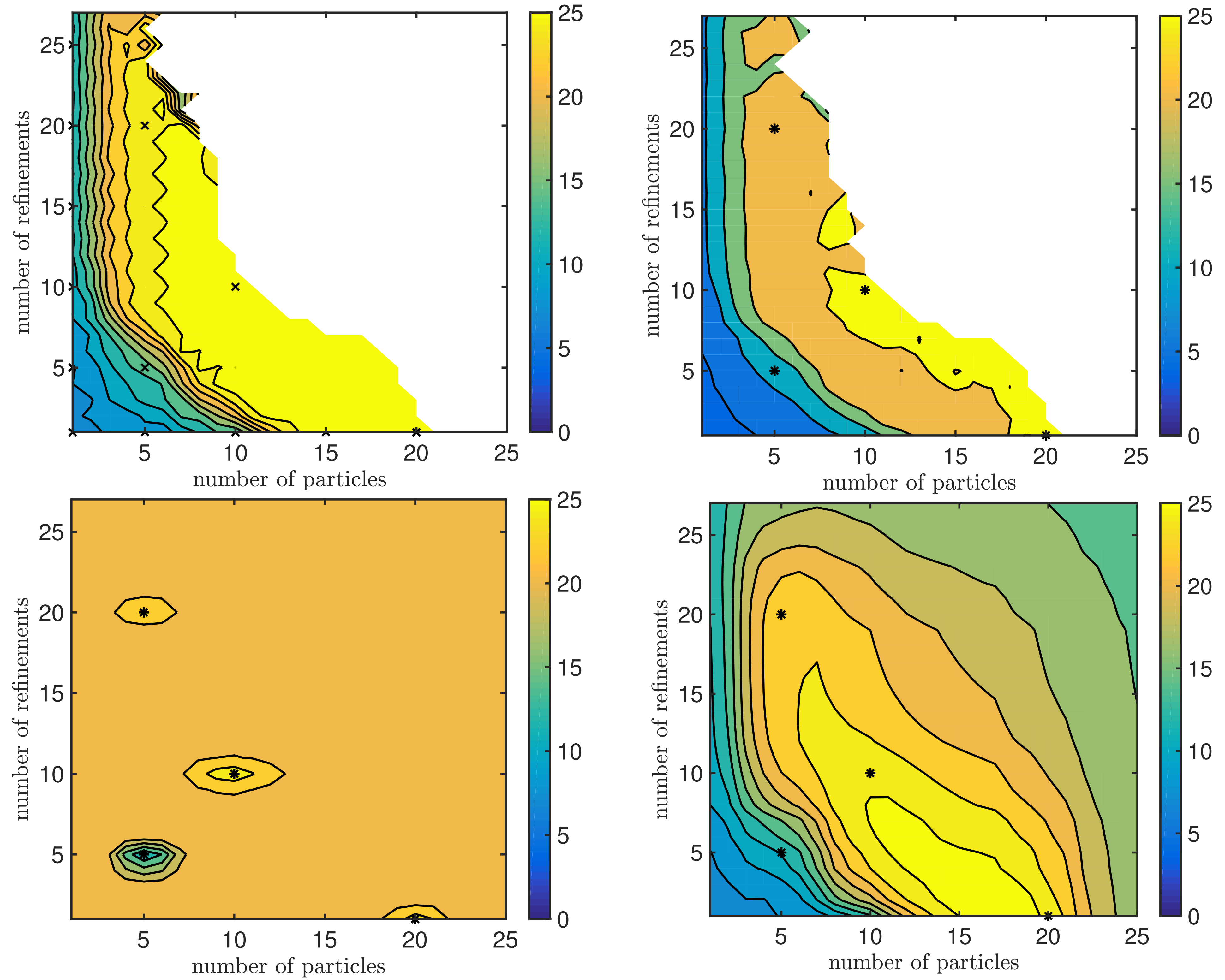
Summary (transfer learning)

- Model for making tradeoff between qualities
- Scale to large space and environmental changes
- Transfer learning can help



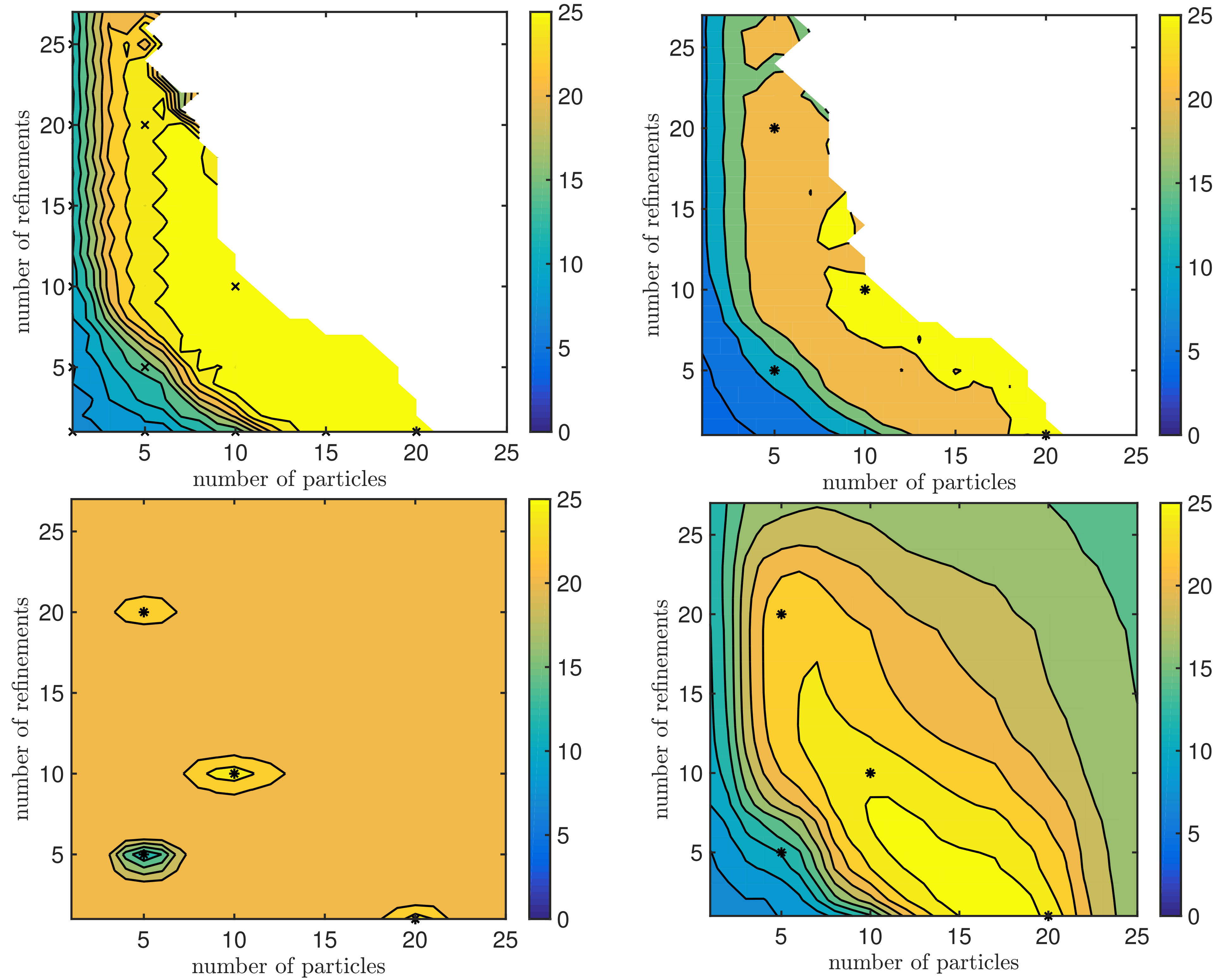
Summary (transfer learning)

- Model for making tradeoff between qualities
- Scale to large space and environmental changes
- Transfer learning can help
    - Increase prediction accuracy



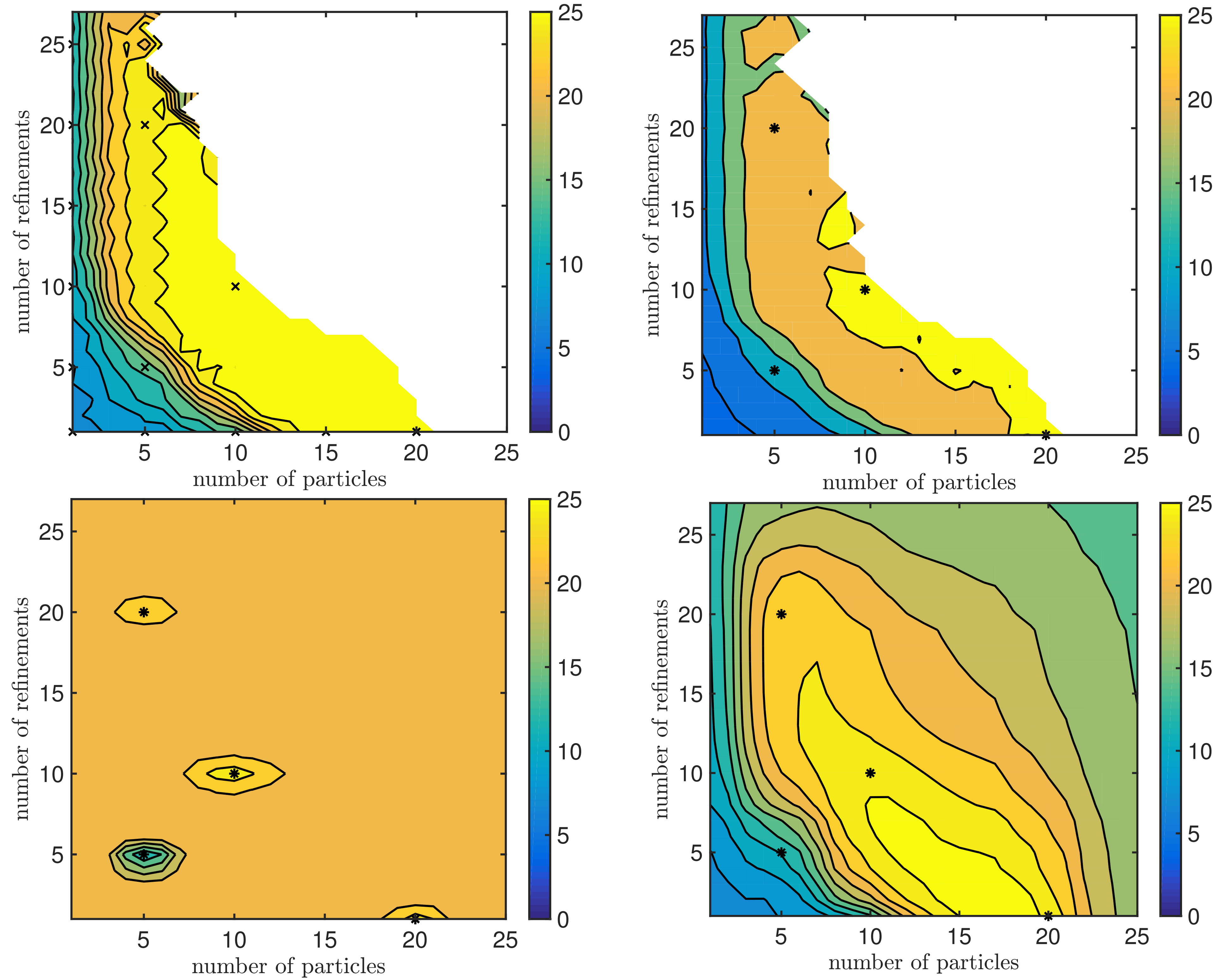
Summary (transfer learning)

- Model for making tradeoff between qualities
- Scale to large space and environmental changes
- Transfer learning can help
    - Increase prediction accuracy
    - Increase model reliability

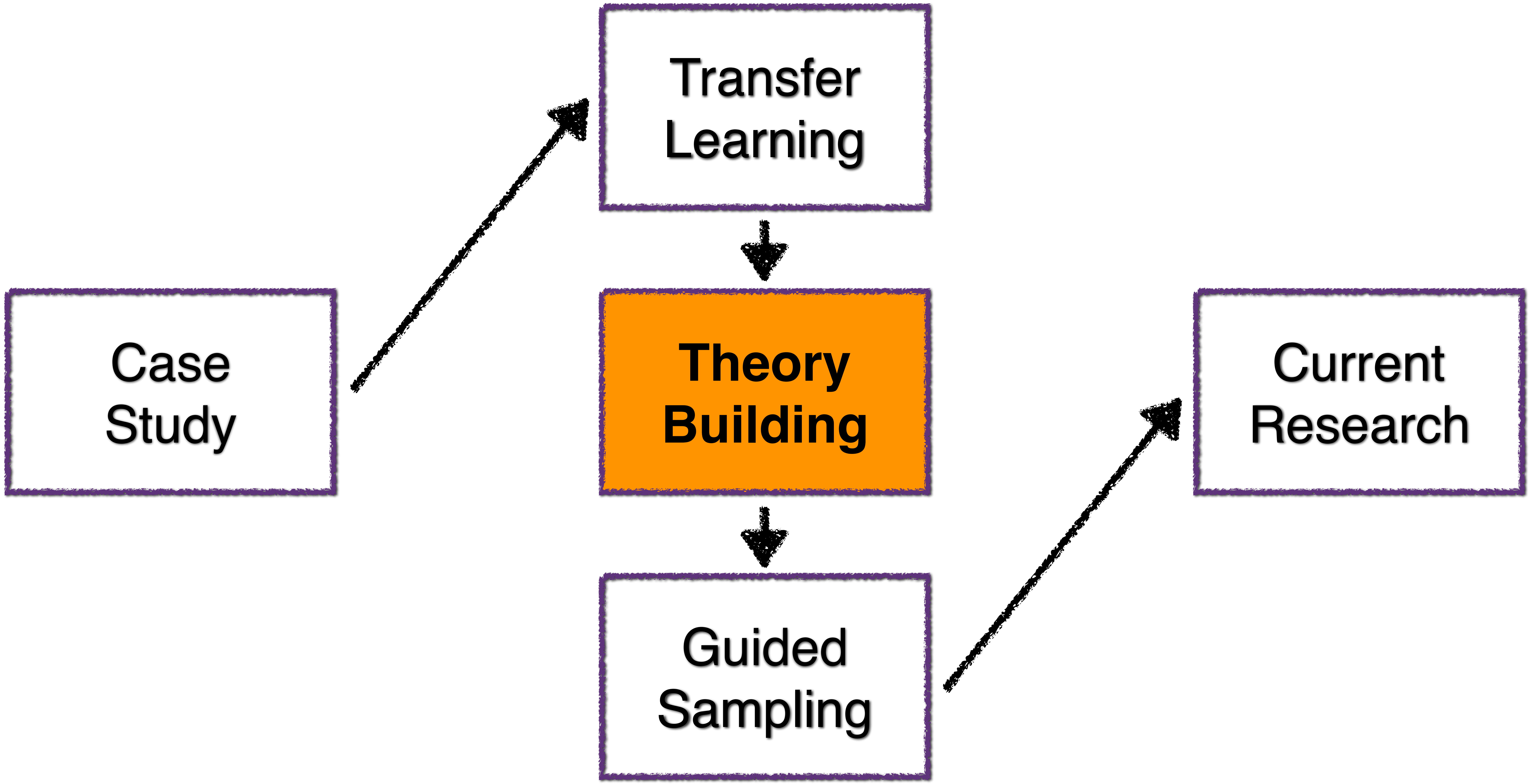


Summary (transfer learning)

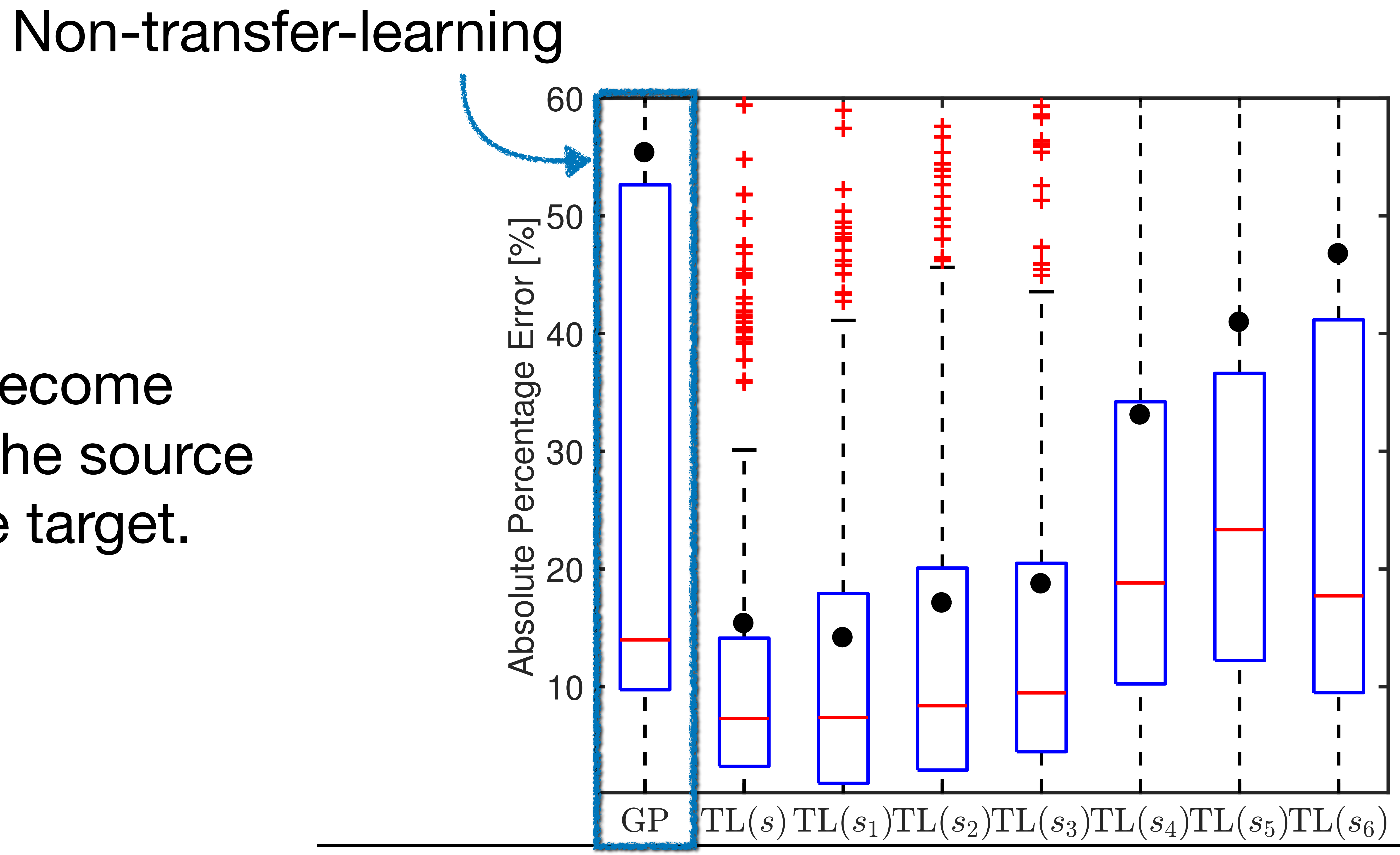
- Model for making tradeoff between qualities
- Scale to large space and environmental changes
- Transfer learning can help
  - Increase prediction accuracy
  - Increase model reliability
  - Decrease model building cost



Outline


Case Study
Transfer Learning
Theory Building
Guided Sampling
Current Research

Looking further: When transfer learning goes wrong



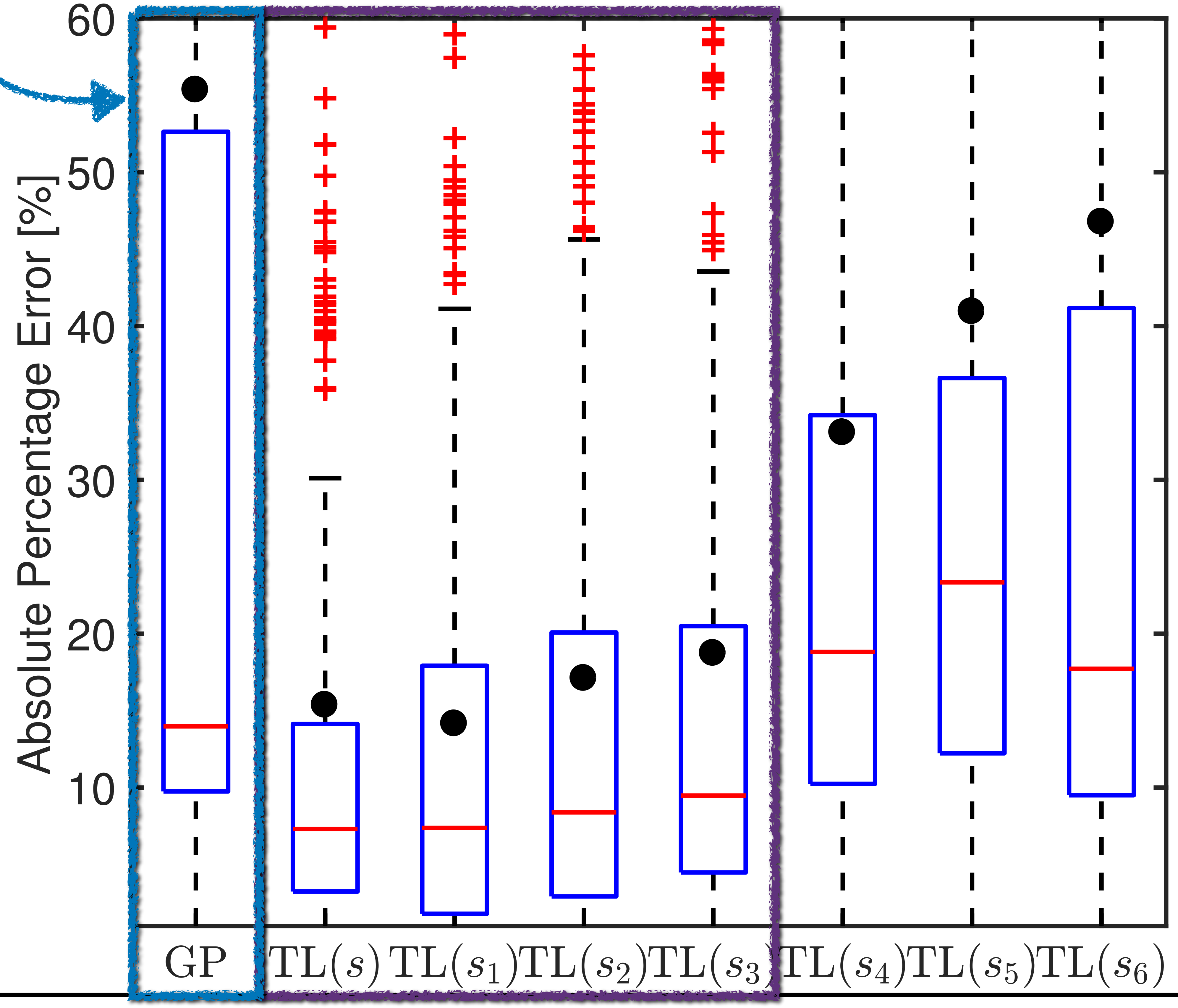
Sources	s	s_1	s_2	s_3	s_4	s_5	s_6
noise-level	0	5	10	15	20	25	30
corr. coeff.	0.98	0.95	0.89	0.75	0.54	0.34	0.19
$\mu(pe)$	15.34	14.14	17.09	18.71	33.06	40.93	46.75

Insight: Predictions become more accurate when the source is **more related** to the target.

Looking further: When transfer learning goes wrong

Non-transfer-learning

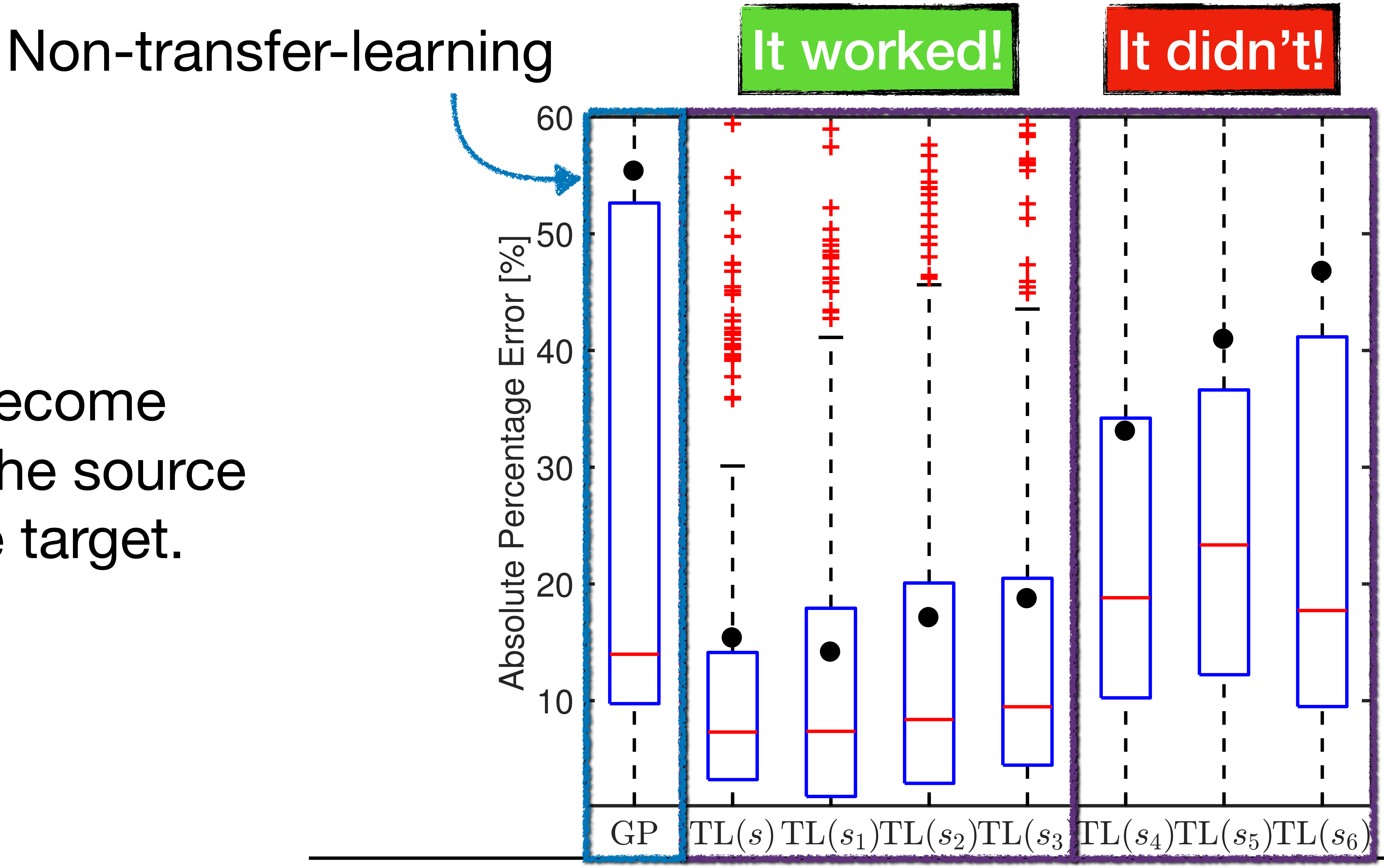
It worked!



Sources	s	s_1	s_2	s_3	s_4	s_5	s_6
noise-level	0	5	10	15	20	25	30
corr. coeff.	0.98	0.95	0.89	0.75	0.54	0.34	0.19
$\mu(pe)$	15.34	14.14	17.09	18.71	33.06	40.93	46.75

Insight: Predictions become more accurate when the source is **more related** to the target.

Looking further: When transfer learning goes wrong



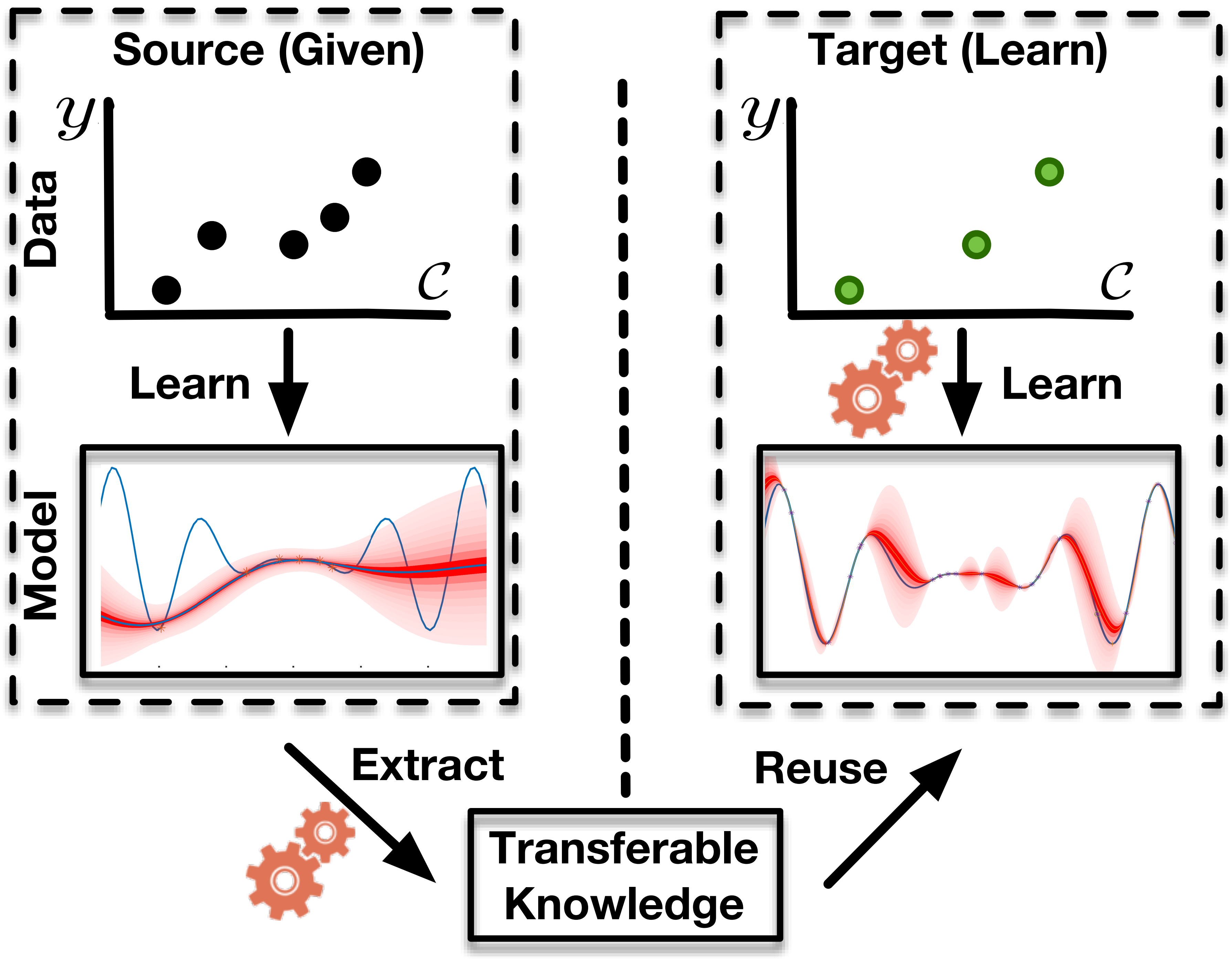
Insight: Predictions become more accurate when the source is **more related** to the target.

Sources	s	s_1	s_2	s_3	s_4	s_5	s_6
noise-level	0	5	10	15	20	25	30
corr. coeff.	0.98	0.95	0.89	0.75	0.54	0.34	0.19
$\mu(pe)$	15.34	14.14	17.09	18.71	33.06	40.93	46.75

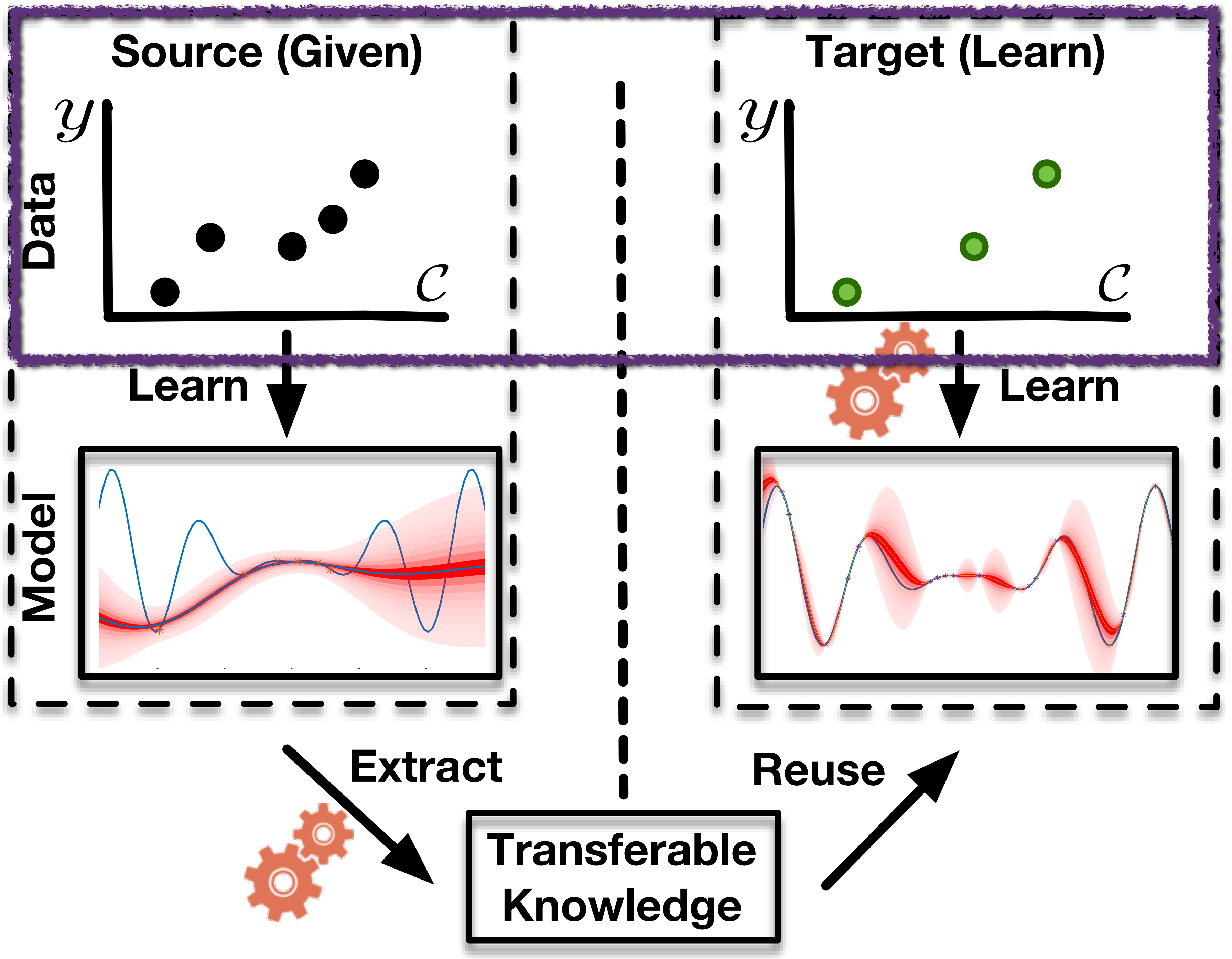
CPU usage [%]
number of refinements
number of particles
(a)
(b)
(c)
(d)
(e)
(f)

CPU usage [%]
It worked!
It worked!
It worked!
It didn't!
It didn't!
It didn't!
number of refinements
number of particles
(a)
(b)
(c)
(d)
(e)
(f)

Key question: Can we develop a theory to explain when transfer learning works?

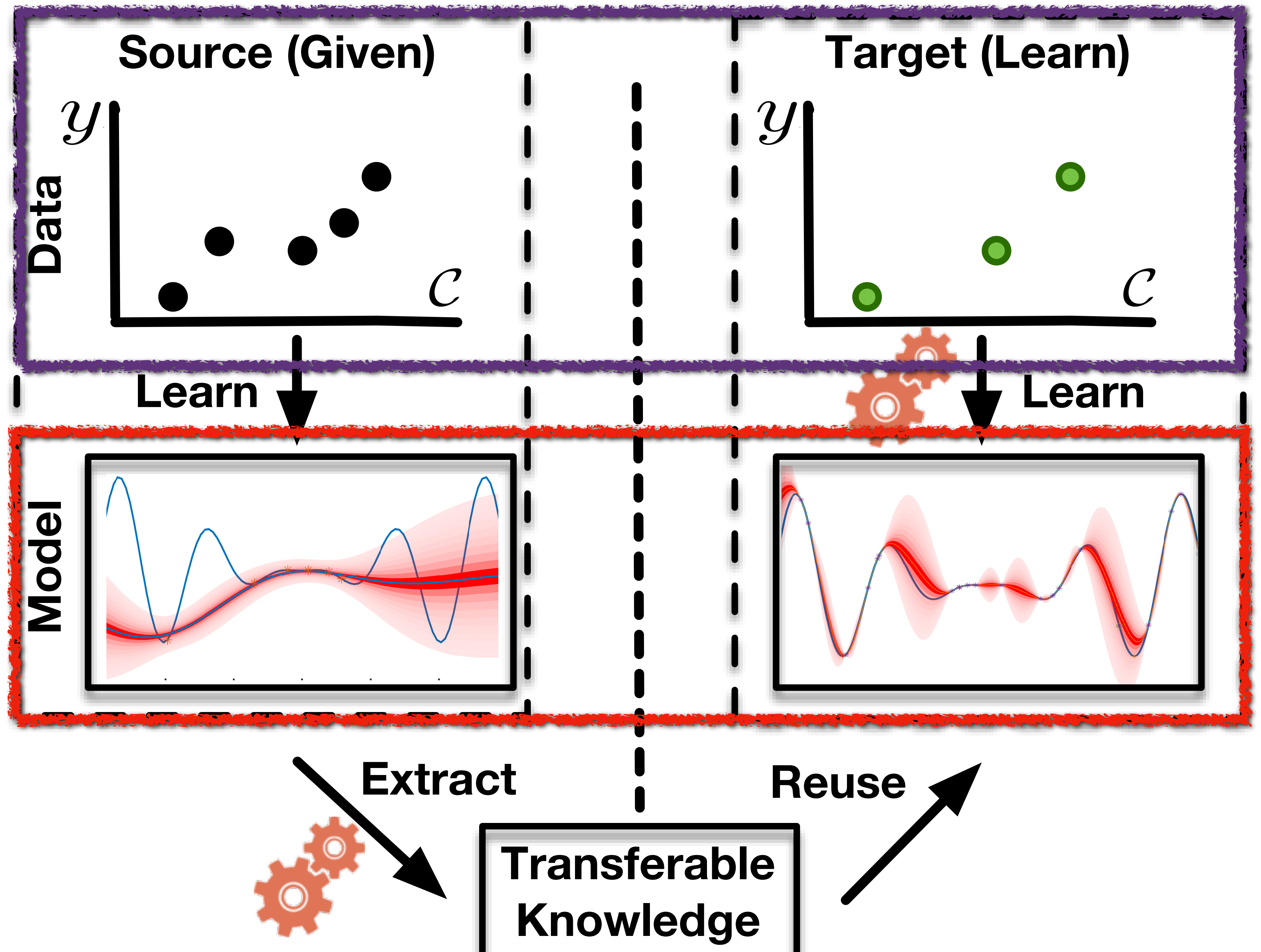


Key question: Can we develop a theory to explain when transfer learning works?



Q1: How source and target are "related"?

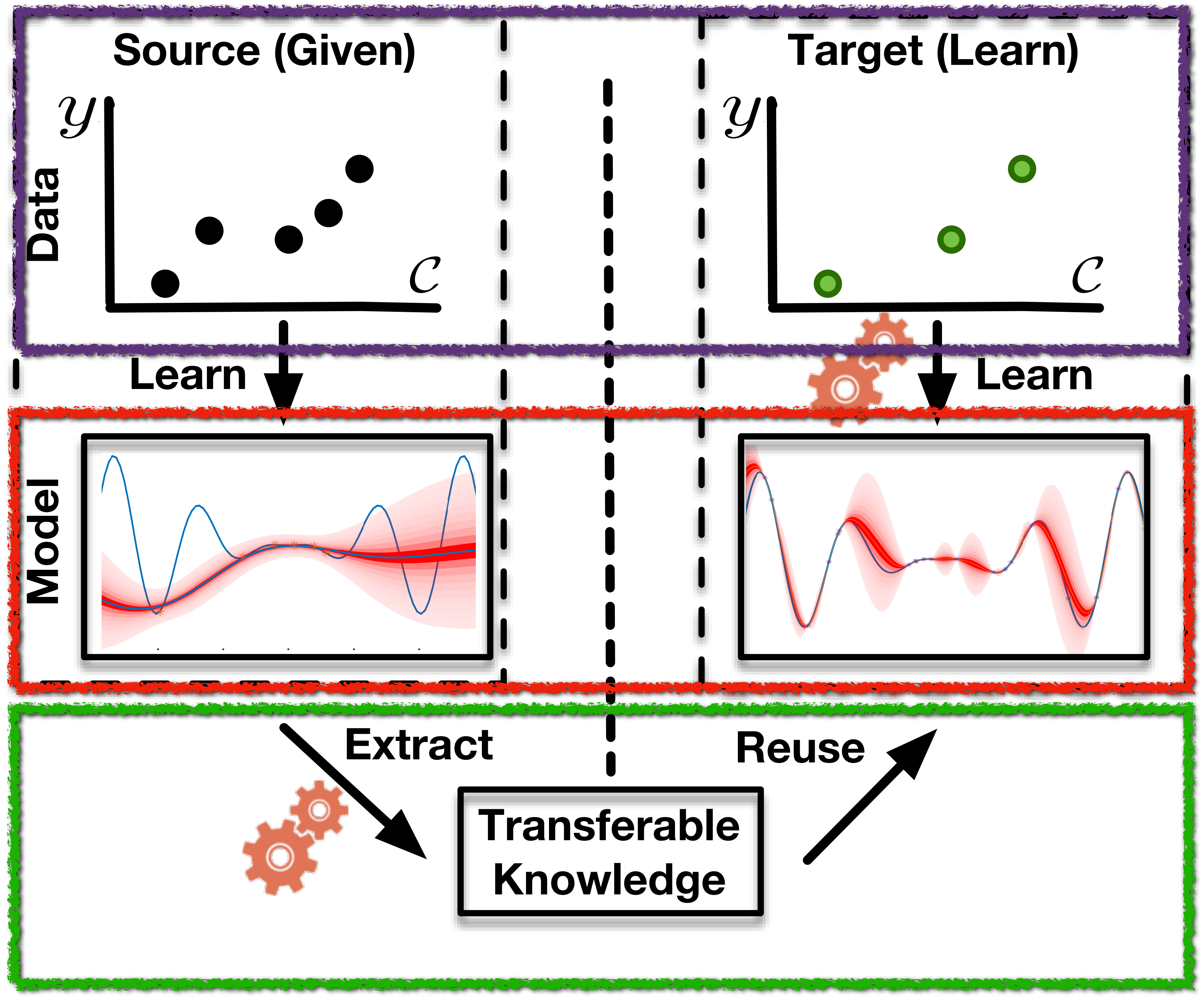
Key question: Can we develop a theory to explain when transfer learning works?



Q1: How source and target are "related"?

Q2: What characteristics are preserved?

Key question: Can we develop a theory to explain when transfer learning works?



Q1: How source and target are "related"?

Q2: What characteristics are preserved?

Q3: What are the actionable insights?

We hypothesized that we can exploit similarities across environments to learn “cheaper” performance models

Source Environment
(Execution time of Program X)

$$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$$

$$\begin{array}{c|c|c|l}
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 & y_{s1} & = f_s(c_1) \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 & y_{s2} & = f_s(c_2) \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 & y_{s3} & = f_s(c_3) \\
 & \cdots & & \\
 & 1 \times 1 \times \cdots \times 1 \times 0 & & \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1 & y_{sn} & = f_s(c_n)
\end{array}$$

Similarity

Target Environment
(Execution time of Program Y)

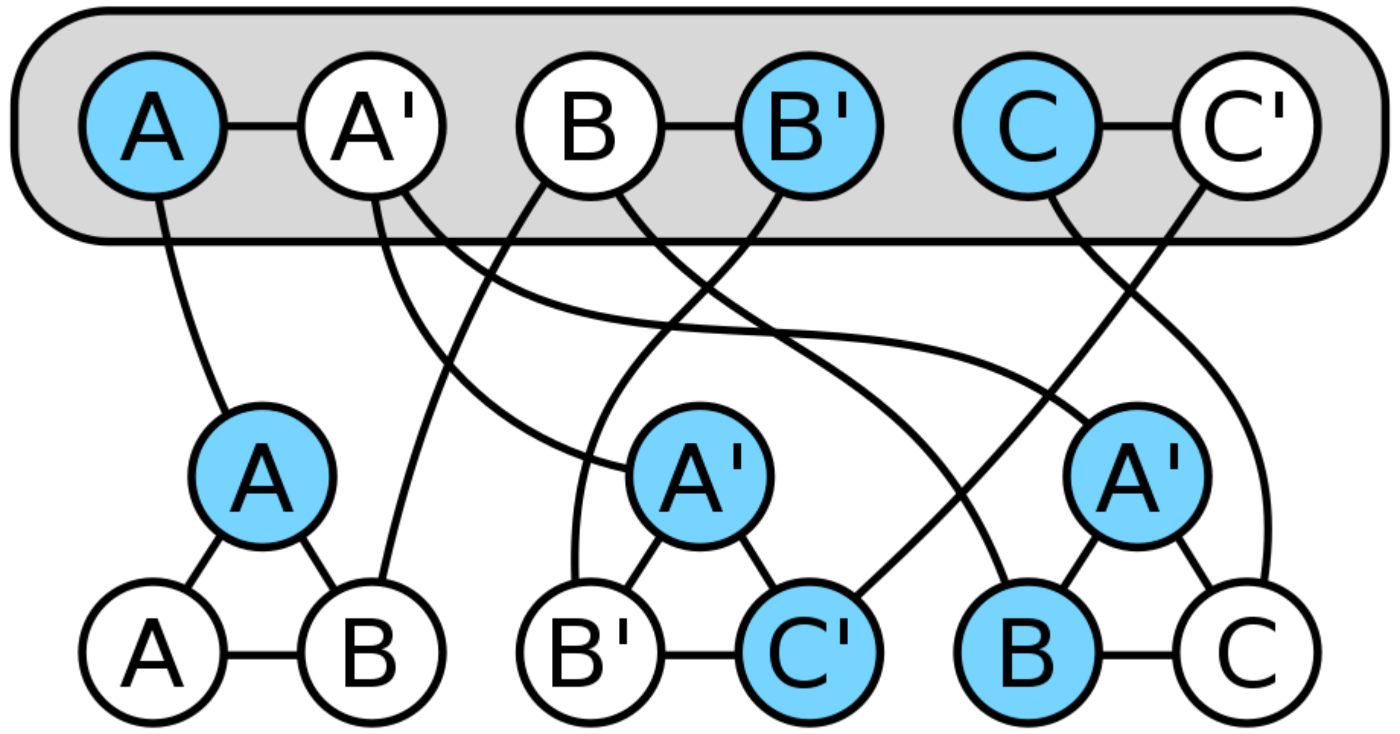
$$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$$

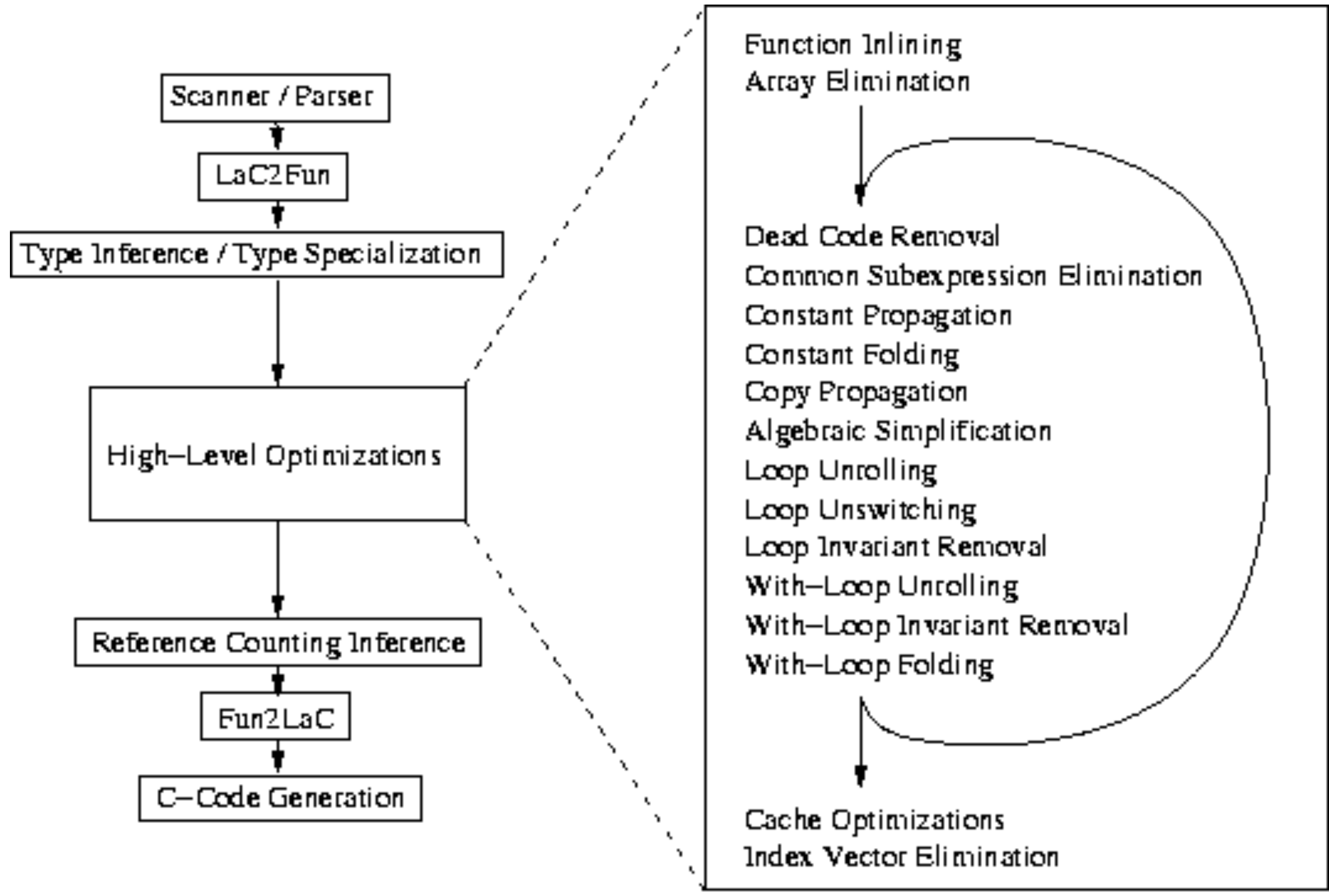
$$\begin{array}{|c|c|l}
0 \times 0 \times \cdots \times 0 \times 1 & y_{t1} & = f_t(c_1) \\
0 \times 0 \times \cdots \times 1 \times 0 & y_{t2} & = f_t(c_2) \\
0 \times 0 \times \cdots \times 1 \times 1 & y_{t3} & = f_t(c_3) \\
\cdots & & \\
1 \times 1 \times \cdots \times 1 \times 0 & & \\
1 \times 1 \times \cdots \times 1 \times 1 & y_{tn} & = f_t(c_n)
\end{array}$$

[P. Jamshidi, et al., “**Transfer learning for performance modeling of configurable systems**….”, ASE’17]

Our empirical study: We looked at different highly-configurable systems to gain insights

$(A \lor B) \land (\neg A \lor \neg B \lor \neg C) \land (\neg A \lor B \lor C)$





SPEAR (SAT Solver)
Analysis time
14 options

X264 (video encoder)
Encoding time
16 options

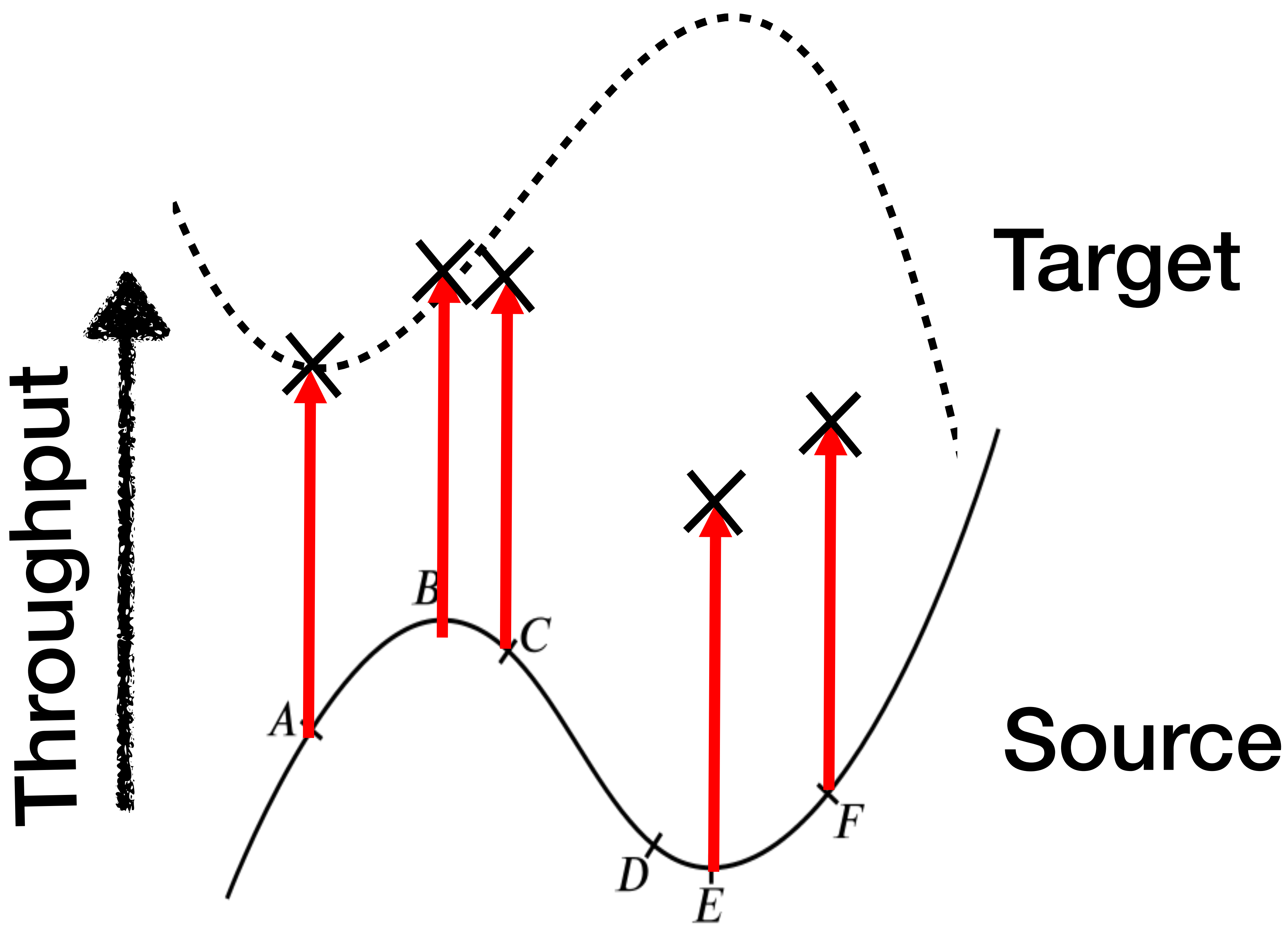
SQLite (DB engine)
Query time
14 options

SaC (Compiler)
Execution time
50 options

[P. Jamshidi, et al., "**Transfer learning for performance modeling of configurable systems**….", ASE'17]

Observation 1: Linear shift happens only in limited environmental changes

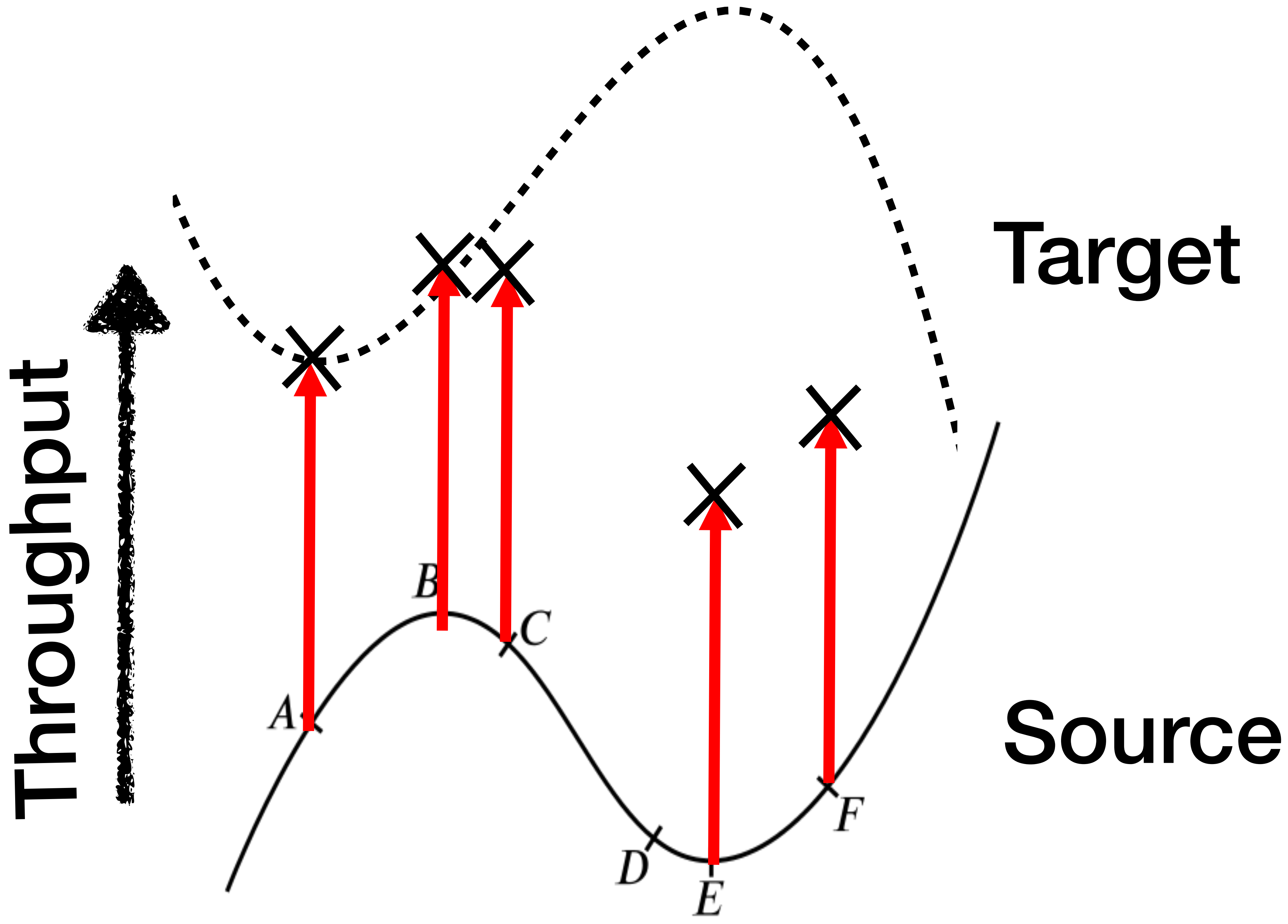
Soft	Environmental change	Severity	Corr.
SPEAR	NUC/2 NUC/4	Small	1.00
	Amazon_nano NUC	Large	0.59
	Hardware/workload/version	V Large	-0.10
x264	Version	Large	0.06
	Workload	Medium	0.65
SQLite	write-seq write-batch	Small	0.96
	read-rand read-seq	Medium	0.50



Implication: Simple transfer learning is limited to hardware changes in practice

Observation 1: Linear shift happens only in limited environmental changes

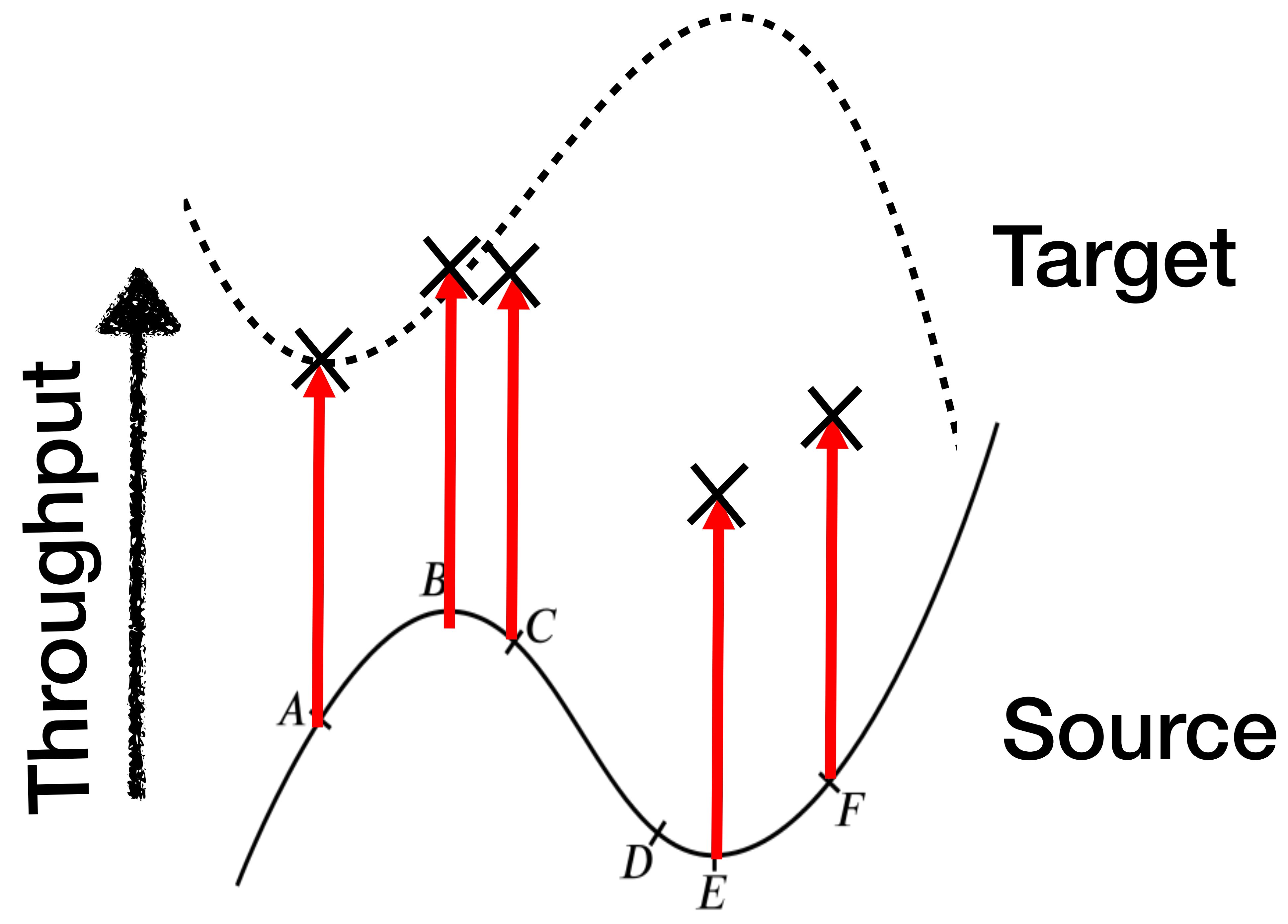
Soft	Environmental change	Severity	Corr.
SPEAR	NUC/2 NUC/4	Small	1.00
	Amazon_nano NUC	Large	0.59
	Hardware/workload/version	V Large	-0.10
x264	Version	Large	0.06
	Workload	Medium	0.65
SQLite	write-seq write-batch	Small	0.96
	read-rand read-seq	Medium	0.50



Implication: Simple transfer learning is limited to hardware changes in practice

Observation 1: Linear shift happens only in limited environmental changes

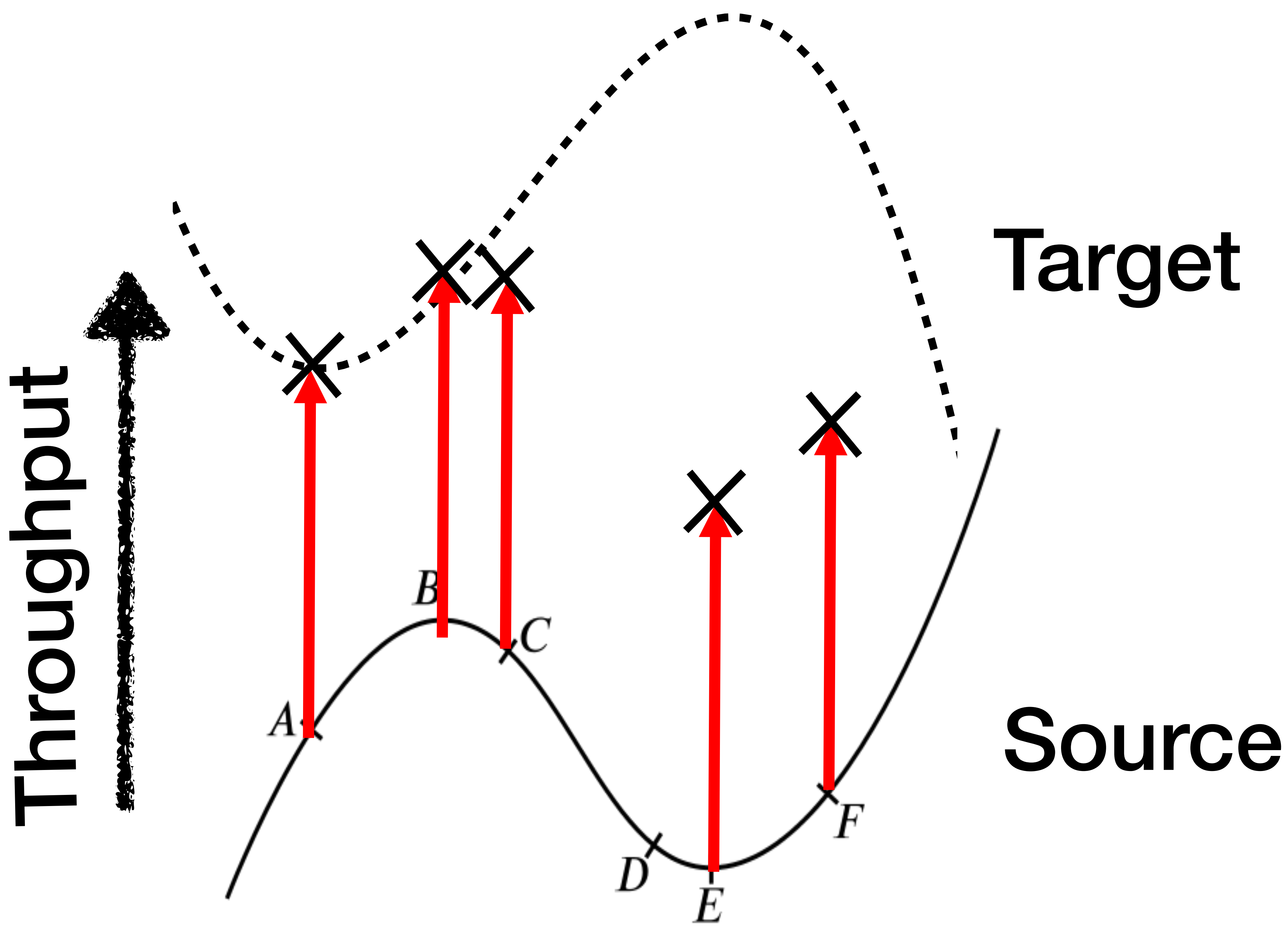
Soft	Environmental change	Severity	Corr.
SPEAR	NUC/2 → NUC/4	Small	1.00
	Amazon_nano → NUC	Large	0.59
	Hardware/workload/version	V Large	-0.10
x264	Version	Large	0.06
	Workload	Medium	0.65
SQLite	write-seq → write-batch	Small	0.96
	read-rand → read-seq	Medium	0.50



Implication: Simple transfer learning is limited to hardware changes in practice

Observation 1: Linear shift happens only in limited environmental changes

Soft	Environmental change	Severity	Corr.
SPEAR	NUC/2 → NUC/4	Small	1.00
	Amazon_nano → NUC	Large	0.59
	Hardware/workload/version	V Large	-0.10
x264	Version	Large	0.06
	Workload	Medium	0.65
SQLite	write-seq → write-batch	Small	0.96
	read-rand → read-seq	Medium	0.50



Implication: Simple transfer learning is limited to hardware changes in practice

Influential options and interactions are preserved across environments

Soft	Environmental change	Severity	Dim	t-test	
x264	Version	Large	16	12	10
	Hardware/workload/ver	V Large		8	9
SQLite	write-seq write-batch	V Large	14	3	4
	read-rand read-seq	Medium		1	1
SaC	Workload	V Large	50	16	10

Implication: Avoid wasting budget on non-informative part of configuration space and focusing where it matters.

Influential options and interactions are preserved across environments

Soft	Environmental change	Severity	Dim	t-test	
x264	Version	Large	16	12	10
	Hardware/workload/ver	V Large		8	9
SQLite	write-seq write-batch	V Large	14	3	4
	read-rand read-seq	Medium		1	1
SaC	Workload	V Large	50	16	10

Implication: Avoid wasting budget on non-informative part of configuration space and focusing where it matters.

Influential options and interactions are preserved across environments

Soft	Environmental change	Severity	Dim	t-test	
x264	Version	Large	16	12	10
	Hardware/workload/ver	V Large		8	9
SQLite	write-seq write-batch	V Large	14	3	4
	read-rand read-seq	Medium		1	1
SaC	Workload	V Large	50	16	10

Implication: Avoid wasting budget on non-informative part of configuration space and focusing where it matters.

Influential options and interactions are preserved across environments

Soft	Environmental change	Severity	Dim	t-test	
x264	Version	Large	16	12	10
	Hardware/workload/ver	V Large		8	9
SQLite	write-seq write-batch	V Large	14	3	4
	read-rand read-seq	Medium		1	1
SaC	Workload	V Large	50	16	10

We only need to explore part of the space:

$$\frac{2^{16}}{2^{50}} = 0.00000000058$$

Implication: Avoid wasting budget on non-informative part of configuration space and focusing where it matters.

Transfer learning across environment

$$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$$

$$c_1 \quad 0 \times 0 \times \cdots \times 0 \times 1 \quad y_{s1} = f_s(c_1)$$

$$c_2 \quad 0 \times 0 \times \cdots \times 1 \times 0 \quad y_{s2} = f_s(c_2)$$

$$c_3 \quad 0 \times 0 \times \cdots \times 1 \times 1 \quad y_{s3} = f_s(c_3)$$

$$\hat{f}_s \sim f_s(\cdot)$$

$$\cdots$$

$$1 \times 1 \times \cdots \times 1 \times 0$$

$$c_n \quad 1 \times 1 \times \cdots \times 1 \times 1 \quad y_{sn} = f_s(c_n)$$

Transfer learning across environment

$$\begin{array}{cc} & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} \\ c_1 & 0 \times 0 \times \cdots \times 0 \times 1 \\ c_2 & 0 \times 0 \times \cdots \times 1 \times 0 \\ c_3 & 0 \times 0 \times \cdots \times 1 \times 1 \\ & \cdots \\ & 1 \times 1 \times \cdots \times 1 \times 0 \\ c_n & 1 \times 1 \times \cdots \times 1 \times 1 \end{array} \quad \begin{array}{l} \\ y_{s1} = f_s(c_1) \\ y_{s2} = f_s(c_2) \\ y_{s3} = f_s(c_3) \\ \\ \\ y_{sn} = f_s(c_n) \end{array}$$

$$\hat{f}_s \sim f_s(\cdot)$$

Transfer learning across environment

Source
(Execution time of Program X)

$$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$$

$$
\begin{array}{l|c|l}
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 & y_{s1} = f_s(c_1) \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 & y_{s2} = f_s(c_2) \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 & y_{s3} = f_s(c_3) \\
 & \cdots & \\
 & 1 \times 1 \times \cdots \times 1 \times 0 & \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1 & y_{sn} = f_s(c_n)
\end{array}
\longrightarrow \quad \hat{f}_s \sim f_s(\cdot)
$$

Transfer learning across environment

Source
(Execution time of Program X)
$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$

$$
\begin{array}{lcll}
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 & y_{s1} & = f_s(c_1) \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 & y_{s2} & = f_s(c_2) \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 & y_{s3} & = f_s(c_3) \\
 & \cdots & & \\
 & 1 \times 1 \times \cdots \times 1 \times 0 & & \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1 & y_{sn} & = f_s(c_n)
\end{array}
$$

$$\xrightarrow{\text{Learn performance model}} \quad \hat{f}_s \sim f_s(\cdot)$$

Observation 1: Not all options and interactions are influential and interactions degree between options are not high

$$\mathbb{C} = O_1 \times O_2 \times O_3 \times O_4 \times O_5 \times O_6 \times O_7 \times O_8 \times O_9 \times O_{10}$$

$$\hat{f}_s(\cdot) = 1.2 + 3\boxed{o_1} + 5\boxed{o_3} + 0.9\boxed{o_7} + 0.8\boxed{o_3 o_7} + 4\boxed{o_1 o_3 o_7}$$

Observation 2: Influential options and interactions are preserved across environments

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = 10.4 - 2.1o_1 + 1.2o_3 + 2.2o_7 + 0.1o_1o_3 - 2.1o_3o_7 + 14o_1o_3o_7$$

Observation 2: Influential options and interactions are preserved across environments

$$\hat{f}_s(\cdot) = 1.2 + 3\boxed{o_1} + 5\boxed{o_3} + 0.9\boxed{o_7} + 0.8\boxed{o_3 o_7} + 4\boxed{o_1 o_3 o_7}$$

$$\hat{f}_t(\cdot) = 10.4 - 2.1\boxed{o_1} + 1.2\boxed{o_3} + 2.2\boxed{o_7} + 0.1 o_1 o_3 - 2.1\boxed{o_3 o_7} + 14\boxed{o_1 o_3 o_7}$$

Observation 2: Influential options and interactions are preserved across environments

$$\hat{f}_s(\cdot) = 1.2 + 3\boxed{o_1} + 5\boxed{o_3} + 0.9\boxed{o_7} + 0.8\boxed{o_3 o_7} + 4\boxed{o_1 o_3 o_7}$$

$$\hat{f}_t(\cdot) = 10.4 - 2.1\boxed{o_1} + 1.2\boxed{o_3} + 2.2\boxed{o_7} + 0.1\boxed{o_1 o_3} - 2.1\boxed{o_3 o_7} + 14\boxed{o_1 o_3 o_7}$$

Observation 2: Influential options and interactions are preserved across environments

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = 10.4 - 2.1o_1 + 1.2o_3 + 2.2o_7 + 0.1o_1o_3 - 2.1o_3o_7 + 14o_1o_3o_7$$

Observation 2: Influential options and interactions are preserved across environments

$$\hat{f}_s(\cdot) = 1.2 + 3\boxed{o_1} + \boxed{5}\boxed{o_3} + \boxed{0.9}\boxed{o_7} + 0.8\boxed{o_3 o_7} + 4\boxed{o_1 o_3 o_7}$$

$$\hat{f}_t(\cdot) = 10.4 - 2.1\boxed{o_1} + \boxed{1.2}\boxed{o_3} + \boxed{2.2}\boxed{o_7} + 0.1\boxed{o_1 o_3} - 2.1\boxed{o_3 o_7} + 14\boxed{o_1 o_3 o_7}$$

Details: [ASE '17]

Transfer Learning for Performance Modeling of Configurable Systems: An Exploratory Analysis

Pooyan Jamshidi
Carnegie Mellon University, USA

Norbert Siegmund
Bauhaus-University Weimar, Germany

Miguel Velez, Christian Kästner
Akshay Patel, Yuvraj Agarwal
Carnegie Mellon University, USA

Abstract—**Modern software systems provide many configuration options which significantly influence their non-functional properties. To understand and predict the effect of configuration options, several sampling and learning strategies have been proposed, albeit often with significant cost to cover the highly dimensional configuration space. Recently, transfer learning has been applied to reduce the effort of constructing performance models by transferring knowledge about performance behavior across environments. While this line of research is promising to learn more accurate models at a lower cost, it is unclear why and when transfer learning works for performance modeling. To shed light on when it is beneficial to apply transfer learning, we conducted an empirical study on four popular software systems, varying software configurations and environmental conditions, such as hardware, workload, and software versions, to identify the key knowledge pieces that can be exploited for transfer learning. Our results show that in small environmental changes (e.g., homogeneous workload change), by applying a linear transformation to the performance model, we can understand the performance behavior of the target environment, while for severe environmental changes (e.g., drastic workload change) we**

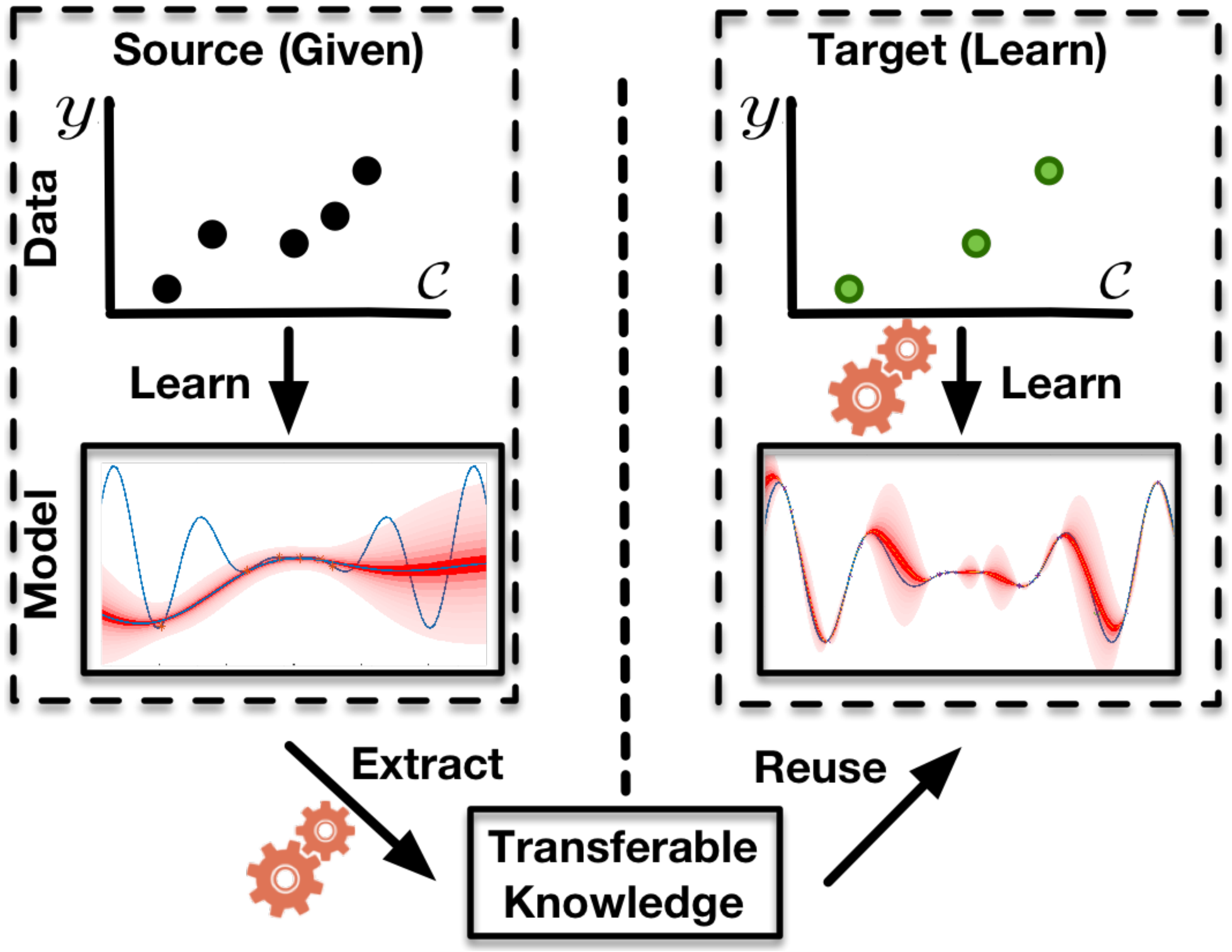
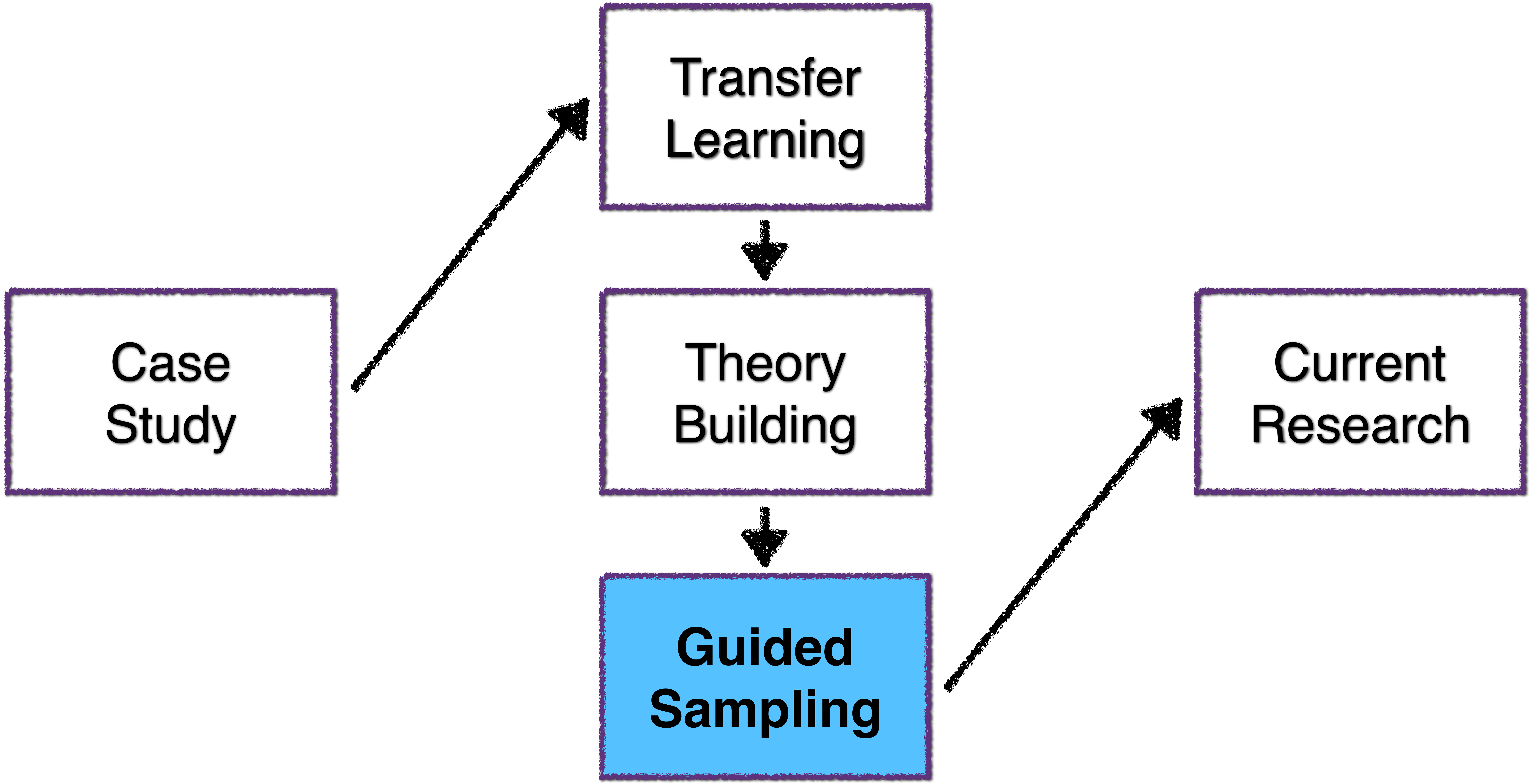


Fig. 1: Transfer learning is a form of machine learning that takes advantage of transferable knowledge from source to learn an accurate, reliable, and less costly model for the target environment.

Outline


Transfer Learning
Case Study
Theory Building
Current Research
Guided Sampling

How to sample the configuration space to learn a “better” performance behavior?

How to select the most informative configurations?

The similarity across environment is a rich source of knowledge for exploration of the configuration space

When we treat the system as black boxes, we cannot typically distinguish between different configurations

$$
\begin{array}{l|c|}
 & O_1 \times O_2 \times \cdots \times O_{19} \times O_{20} \\
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 \\
 & \cdots \\
 & 1 \times 1 \times \cdots \times 1 \times 0 \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1
\end{array}
$$

- We therefore end up blindly explore the configuration space
- That is essentially the key reason why "most" work in this area consider random sampling.

Without considering this knowledge, many samples may not provide new information

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

Without considering this knowledge, many samples may not provide new information

$$\hat{f}_s(\cdot) = 1.2 + 3\boxed{o_1} + 5\boxed{o_3} + 0.9\boxed{o_7} + 0.8\boxed{o_3 o_7} + 4\boxed{o_1 o_3 o_7}$$

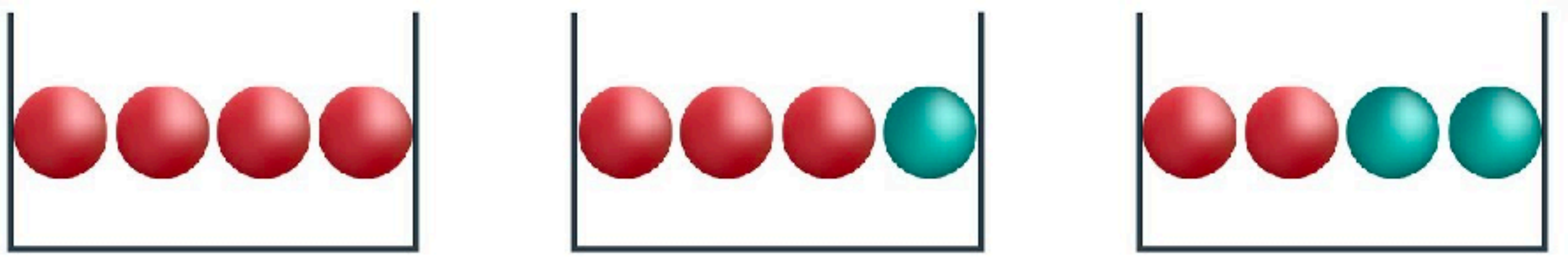
$$\begin{array}{lccccccccccccccccccc}
 & O_1 & \times & O_2 & \times & O_3 & \times & O_4 & \times & O_5 & \times & O_6 & \times & O_7 & \times & O_8 & \times & O_9 & \times & O_{10} \\
c_1 & 1 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 \\
c_2 & 1 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 1 \\
c_3 & 1 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \times & 1 & \times & 0 & \times & 1 & \times & 0 \\
 & & & & & & & & & & \cdots & & & & & & & & & \\
c_{128} & 1 & \times & 1 & \times & 1 & \times & 1 & \times & 1 & \times & 1 & \times & 1 & \times & 1 & \times & 1 & \times & 1
\end{array}$$

Without considering this knowledge, many samples may not provide new information

$$\hat{f}_s(\cdot) = 1.2 + 3\boxed{o_1} + 5\boxed{o_3} + 0.9\boxed{o_7} + 0.8\boxed{o_3 o_7} + 4\boxed{o_1 o_3 o_7}$$

$$\begin{array}{ll}
 & O_1 \times O_2 \times O_3 \times O_4 \times O_5 \times O_6 \times O_7 \times O_8 \times O_9 \times O_{10} \\
c_1 & 1 \times 0 \times 1 \times 0 \times 0 \times 0 \times 1 \times 0 \times 0 \times 0 \qquad \hat{f}_s(c_1) = 14.9 \\
c_2 & 1 \times 0 \times 1 \times 0 \times 0 \times 0 \times 1 \times 0 \times 0 \times 1 \qquad \hat{f}_s(c_2) = 14.9 \\
c_3 & 1 \times 0 \times 1 \times 0 \times 0 \times 0 \times 1 \times 0 \times 1 \times 0 \qquad \hat{f}_s(c_3) = 14.9 \\
 & \cdots \\
c_{128} & 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \qquad \hat{f}_s(c_{128}) = 14.9
\end{array}$$

Without knowing this knowledge, many blind/ random samples may not provide any additional information about performance of the system



In higher dimensional spaces, the blind samples even become less informative/effective

Learning to Sample (L2S)

Extracting the knowledge about influential options and interactions: Step-wise linear regression

Source
(Execution time of Program X)

$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$

$$
\begin{array}{l}
c_1 \\ c_2 \\ c_3 \\ \\ \\ c_n
\end{array}
\boxed{\begin{array}{c}
0 \times 0 \times \cdots \times 0 \times 1 \\
0 \times 0 \times \cdots \times 1 \times 0 \\
0 \times 0 \times \cdots \times 1 \times 1 \\
\cdots \\
1 \times 1 \times \cdots \times 1 \times 0 \\
1 \times 1 \times \cdots \times 1 \times 1
\end{array}}
\quad
\boxed{\begin{array}{c}
y_{s1} \\ y_{s2} \\ y_{s3} \\ \\ \\ y_{sn}
\end{array}}
\begin{array}{l}
= f_s(c_1) \\ = f_s(c_2) \\ = f_s(c_3) \\ \\ \\ = f_s(c_n)
\end{array}
$$

Extracting the knowledge about influential options and interactions: Step-wise linear regression

Source
(Execution time of Program X)

$$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$$

$$\begin{array}{l|c|l}
c_1 & 0 \times 0 \times \cdots \times 0 \times 1 & y_{s1} = f_s(c_1) \\
c_2 & 0 \times 0 \times \cdots \times 1 \times 0 & y_{s2} = f_s(c_2) \\
c_3 & 0 \times 0 \times \cdots \times 1 \times 1 & y_{s3} = f_s(c_3) \\
 & \cdots & \\
 & 1 \times 1 \times \cdots \times 1 \times 0 & \\
c_n & 1 \times 1 \times \cdots \times 1 \times 1 & y_{sn} = f_s(c_n)
\end{array}$$

$$\xrightarrow{\text{Learn performance model}}$$

$$\hat{f}_s \sim f_s(\cdot)$$

Extracting the knowledge about influential options and interactions: Step-wise linear regression

Source
(Execution time of Program X)
$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$

$$
\begin{array}{l}
c_1 \\ c_2 \\ c_3 \\ \\ \\ c_n
\end{array}
\begin{array}{c}
0 \times 0 \times \cdots \times 0 \times 1 \\
0 \times 0 \times \cdots \times 1 \times 0 \\
0 \times 0 \times \cdots \times 1 \times 1 \\
\cdots \\
1 \times 1 \times \cdots \times 1 \times 0 \\
1 \times 1 \times \cdots \times 1 \times 1
\end{array}
\quad
\begin{array}{l}
y_{s1} = f_s(c_1) \\
y_{s2} = f_s(c_2) \\
y_{s3} = f_s(c_3) \\
\\
\\
y_{sn} = f_s(c_n)
\end{array}
$$

$\xrightarrow{\textbf{Learn performance model}}$

$\hat{f}_s \sim f_s(\cdot)$

1. Fit an **initial model**

Extracting the knowledge about influential options and interactions: Step-wise linear regression

Source
(Execution time of Program X)
$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$

$$
\begin{array}{l}
c_1 \\ c_2 \\ c_3 \\ \\ \\ c_n
\end{array}
\boxed{\begin{array}{c}
0 \times 0 \times \cdots \times 0 \times 1 \\
0 \times 0 \times \cdots \times 1 \times 0 \\
0 \times 0 \times \cdots \times 1 \times 1 \\
\cdots \\
1 \times 1 \times \cdots \times 1 \times 0 \\
1 \times 1 \times \cdots \times 1 \times 1
\end{array}}
\quad
\boxed{\begin{array}{c}
y_{s1} \\ y_{s2} \\ y_{s3} \\ \\ \\ y_{sn}
\end{array}}
\begin{array}{l}
= f_s(c_1) \\ = f_s(c_2) \\ = f_s(c_3) \\ \\ \\ = f_s(c_n)
\end{array}
$$

Learn performance model

$\longrightarrow$

$\hat{f}_s \sim f_s(\cdot)$

1. Fit an **initial model**
2. **Forward selection**: Add terms iteratively

Extracting the knowledge about influential options and interactions: Step-wise linear regression

Source
(Execution time of Program X)

$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$

$$
\begin{array}{l}
c_1 \\ c_2 \\ c_3 \\ \\ \\ c_n
\end{array}
\boxed{\begin{array}{c}
0 \times 0 \times \cdots \times 0 \times 1 \\
0 \times 0 \times \cdots \times 1 \times 0 \\
0 \times 0 \times \cdots \times 1 \times 1 \\
\cdots \\
1 \times 1 \times \cdots \times 1 \times 0 \\
1 \times 1 \times \cdots \times 1 \times 1
\end{array}}
\quad
\boxed{\begin{array}{c}
y_{s1} \\ y_{s2} \\ y_{s3} \\ \\ \\ y_{sn}
\end{array}}
\begin{array}{l}
= f_s(c_1) \\ = f_s(c_2) \\ = f_s(c_3) \\ \\ \\ = f_s(c_n)
\end{array}
$$

Learn performance model

$\hat{f}_s \sim f_s(\cdot)$

1. Fit an **initial model**
2. **Forward selection**: Add terms iteratively
3. **Backward elimination**: Removes terms iteratively

Extracting the knowledge about influential options and interactions: Step-wise linear regression

Source
(Execution time of Program X)
$O_1 \times O_2 \times \cdots \times O_{19} \times O_{20}$

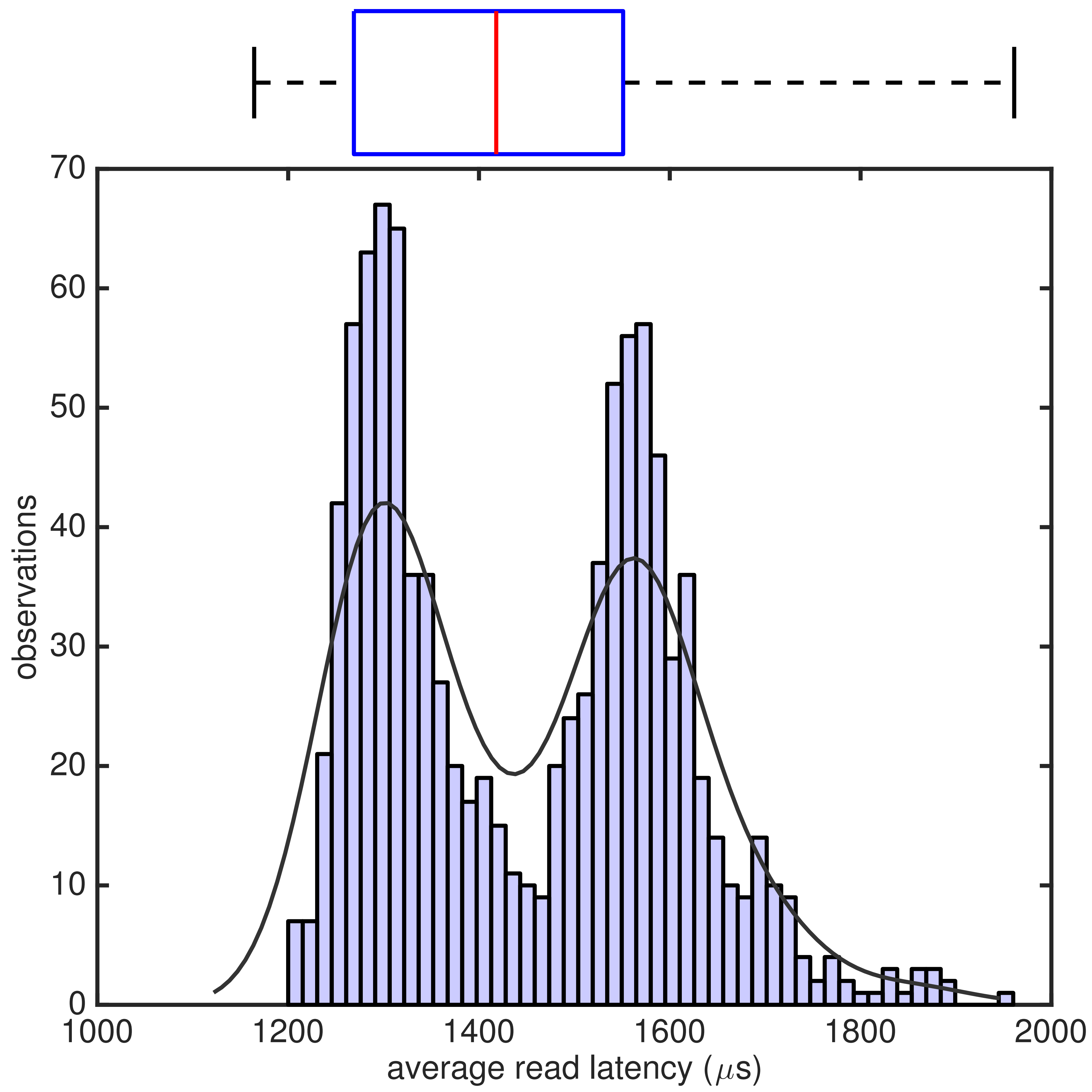
$$
\begin{array}{l}
c_1 \\ c_2 \\ c_3 \\ \\ \\ c_n
\end{array}
\begin{array}{c}
0 \times 0 \times \cdots \times 0 \times 1 \\
0 \times 0 \times \cdots \times 1 \times 0 \\
0 \times 0 \times \cdots \times 1 \times 1 \\
\cdots \\
1 \times 1 \times \cdots \times 1 \times 0 \\
1 \times 1 \times \cdots \times 1 \times 1
\end{array}
\quad
\begin{array}{l}
y_{s1} = f_s(c_1) \\
y_{s2} = f_s(c_2) \\
y_{s3} = f_s(c_3) \\
\\
\\
y_{sn} = f_s(c_n)
\end{array}
$$

Learn performance model

$\hat{f}_s \sim f_s(\cdot)$

1. Fit an **initial model**
2. **Forward selection**: Add terms iteratively
3. **Backward elimination**: Removes terms iteratively
4. **Terminate**: When neither (2) or (3) improve the model

Build a performance distribution using kernel density estimation using the source data



L2S extracts the knowledge about influential options and interactions via performance models

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

L2S extracts the knowledge about influential options and interactions via performance models

$$\hat{f}_s(\cdot) = 1.2 + 3\boxed{o_1} + 5\boxed{o_3} + 0.9\boxed{o_7} + 0.8\boxed{o_3 o_7} + 4\boxed{o_1 o_3 o_7}$$

L2S exploits the knowledge it gained from the source to sample the target environment

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = 10.4 - 2.1o_1 + 1.2o_3 + 2.2o_7 + 0.1o_1o_3 - 2.1o_3o_7 + 14o_1o_3o_7$$

$$\underline{O_1 \times O_2 \times O_3 \times O_4 \times O_5 \times O_6 \times O_7 \times O_8 \times O_9 \times O_{10}}$$

L2S exploits the knowledge it gained from the source to sample the target environment

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = \boxed{10.4} - 2.1o_1 + 1.2o_3 + 2.2o_7 + 0.1o_1o_3 - 2.1o_3o_7 + 14o_1o_3o_7$$

$$\begin{array}{cc} & O_1 \times O_2 \times O_3 \times O_4 \times O_5 \times O_6 \times O_7 \times O_8 \times O_9 \times O_{10} \\ \hline c_1 & 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \end{array} \quad \hat{f}_t(c_1) = 10.4$$

L2S exploits the knowledge it gained from the source to sample the target environment

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = \boxed{10.4} - 2.1\boxed{o_1} + 1.2o_3 + 2.2o_7 + 0.1o_1o_3 - 2.1o_3o_7 + 14o_1o_3o_7$$

$$\begin{array}{ll}
 & O_1 \times O_2 \times O_3 \times O_4 \times O_5 \times O_6 \times O_7 \times O_8 \times O_9 \times O_{10} \\
\hline
c_1 & 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \quad \hat{f}_t(c_1) = 10.4 \\
c_2 & 1 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \quad \hat{f}_t(c_2) = 8.1
\end{array}$$

L2S exploits the knowledge it gained from the source to sample the target environment

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = \boxed{10.4} - 2.1\boxed{o_1} + 1.2\boxed{o_3} + 2.2o_7 + 0.1o_1o_3 - 2.1o_3o_7 + 14o_1o_3o_7$$

$$\begin{array}{lcl}
 & O_1 \times O_2 \times O_3 \times O_4 \times O_5 \times O_6 \times O_7 \times O_8 \times O_9 \times O_{10} & \\
\hline
c_1 & 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 & \hat{f}_t(c_1) = 10.4 \\
c_2 & 1 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 & \hat{f}_t(c_2) = 8.1 \\
c_3 & 0 \times 0 \times 1 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 & \hat{f}_t(c_3) = 11.6
\end{array}$$

L2S exploits the knowledge it gained from the source to sample the target environment

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = \boxed{10.4} - 2.1\boxed{o_1} + 1.2\boxed{o_3} + 2.2\boxed{o_7} + 0.1o_1o_3 - 2.1o_3o_7 + 14o_1o_3o_7$$

	$O_1 \times O_2 \times O_3 \times O_4 \times O_5 \times O_6 \times O_7 \times O_8 \times O_9 \times O_{10}$	
c_1	$0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0$	$\hat{f}_t(c_1) = 10.4$
c_2	$1 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0$	$\hat{f}_t(c_2) = 8.1$
c_3	$0 \times 0 \times 1 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0$	$\hat{f}_t(c_3) = 11.6$
c_4	$0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 1 \times 0 \times 0 \times 0$	$\hat{f}_t(c_4) = 12.6$

L2S exploits the knowledge it gained from the source to sample the target environment

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = \boxed{10.4} - 2.1\boxed{o_1} + 1.2\boxed{o_3} + 2.2\boxed{o_7} + 0.1o_1o_3 - 2.1\boxed{o_3o_7} + 14o_1o_3o_7$$

	$O_1 \times O_2 \times O_3 \times O_4 \times O_5 \times O_6 \times O_7 \times O_8 \times O_9 \times O_{10}$	
c_1	$0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0$	$\hat{f}_t(c_1) = 10.4$
c_2	$1 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0$	$\hat{f}_t(c_2) = 8.1$
c_3	$0 \times 0 \times 1 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0$	$\hat{f}_t(c_3) = 11.6$
c_4	$0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 1 \times 0 \times 0 \times 0$	$\hat{f}_t(c_4) = 12.6$
c_5	$0 \times 0 \times 1 \times 0 \times 0 \times 0 \times 1 \times 0 \times 0 \times 0$	$\hat{f}_t(c_5) = 11.7$

L2S exploits the knowledge it gained from the source to sample the target environment

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = \boxed{10.4} - 2.1\boxed{o_1} + 1.2\boxed{o_3} + 2.2\boxed{o_7} + 0.1o_1o_3 - 2.1\boxed{o_3o_7} + 14\boxed{o_1o_3o_7}$$

$$\begin{array}{ll}
 & \underline{O_1 \times O_2 \times O_3 \times O_4 \times O_5 \times O_6 \times O_7 \times O_8 \times O_9 \times O_{10}} \\
c_1 & 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \quad \hat{f}_t(c_1) = 10.4 \\
c_2 & 1 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \quad \hat{f}_t(c_2) = 8.1 \\
c_3 & 0 \times 0 \times 1 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 0 \quad \hat{f}_t(c_3) = 11.6 \\
c_4 & 0 \times 0 \times 0 \times 0 \times 0 \times 0 \times 1 \times 0 \times 0 \times 0 \quad \hat{f}_t(c_4) = 12.6 \\
c_5 & 0 \times 0 \times 1 \times 0 \times 0 \times 0 \times 1 \times 0 \times 0 \times 0 \quad \hat{f}_t(c_5) = 11.7 \\
c_6 & 1 \times 0 \times 1 \times 0 \times 0 \times 0 \times 1 \times 0 \times 0 \times 0 \quad \hat{f}_t(c_6) = 23.7
\end{array}$$

L2S exploits the knowledge it gained from the source to sample the target environment

$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = \boxed{10.4} - 2.1\boxed{o_1} + 1.2\boxed{o_3} + 2.2\boxed{o_7} + 0.1o_1o_3 - 2.1\boxed{o_3o_7} + 14\boxed{o_1o_3o_7}$$

	O_1	$\times$	O_2	$\times$	O_3	$\times$	O_4	$\times$	O_5	$\times$	O_6	$\times$	O_7	$\times$	O_8	$\times$	O_9	$\times$	O_{10}	
c_1	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\hat{f}_t(c_1) = 10.4$
c_2	1	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\hat{f}_t(c_2) = 8.1$
c_3	0	$\times$	0	$\times$	1	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\hat{f}_t(c_3) = 11.6$
c_4	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	0	$\times$	1	$\times$	0	$\times$	0	$\times$	0	$\hat{f}_t(c_4) = 12.6$
c_5	0	$\times$	0	$\times$	1	$\times$	0	$\times$	0	$\times$	0	$\times$	1	$\times$	0	$\times$	0	$\times$	0	$\hat{f}_t(c_5) = 11.7$
c_6	1	$\times$	0	$\times$	1	$\times$	0	$\times$	0	$\times$	0	$\times$	1	$\times$	0	$\times$	0	$\times$	0	$\hat{f}_t(c_6) = 23.7$

L2S exploits the knowledge it gained from the source to sample the target environment

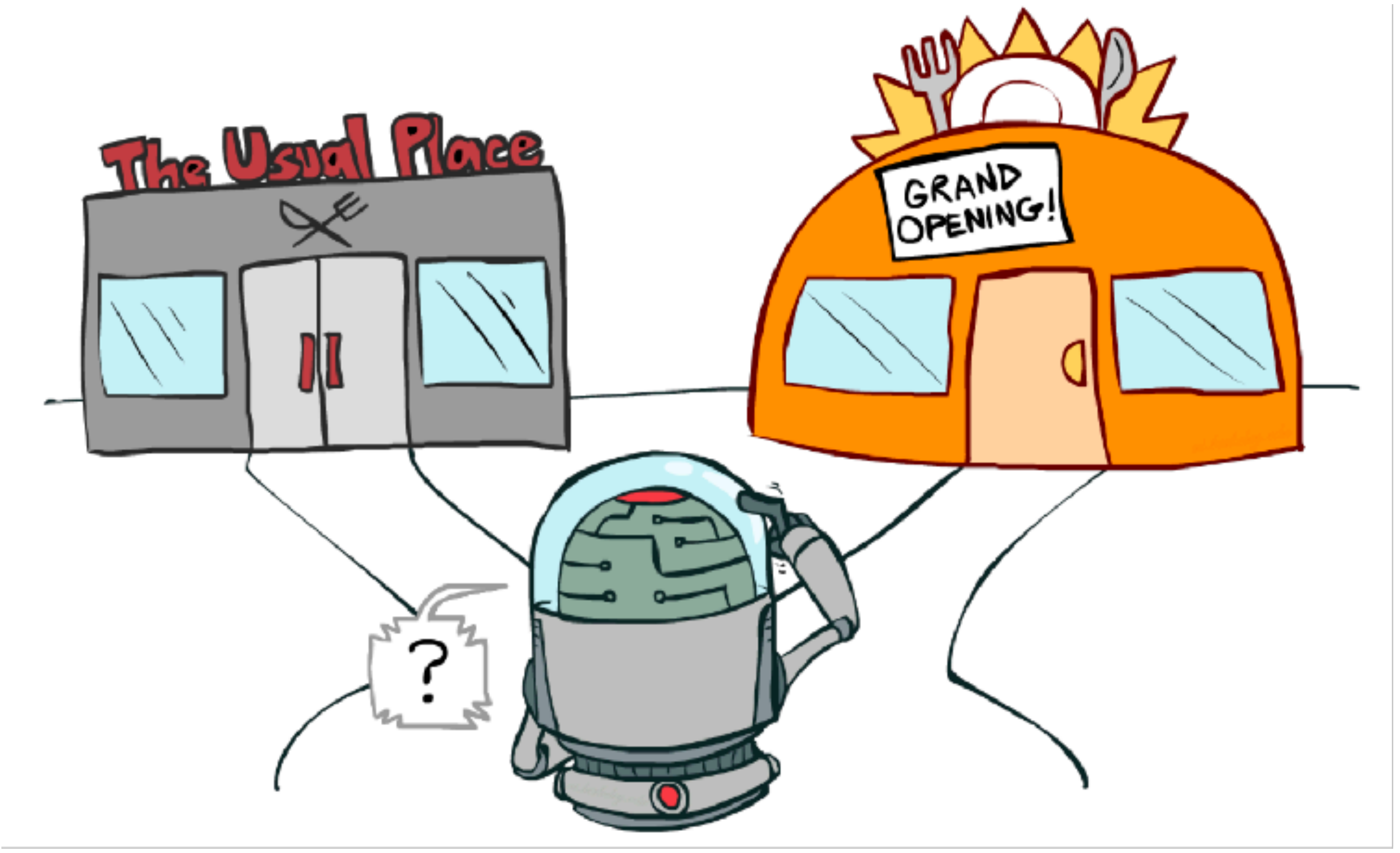
$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

$$\hat{f}_t(\cdot) = \boxed{10.4} - 2.1\boxed{o_1} + 1.2\boxed{o_3} + 2.2\boxed{o_7} + 0.1\boxed{o_1o_3} - 2.1\boxed{o_3o_7} + 14\boxed{o_1o_3o_7}$$

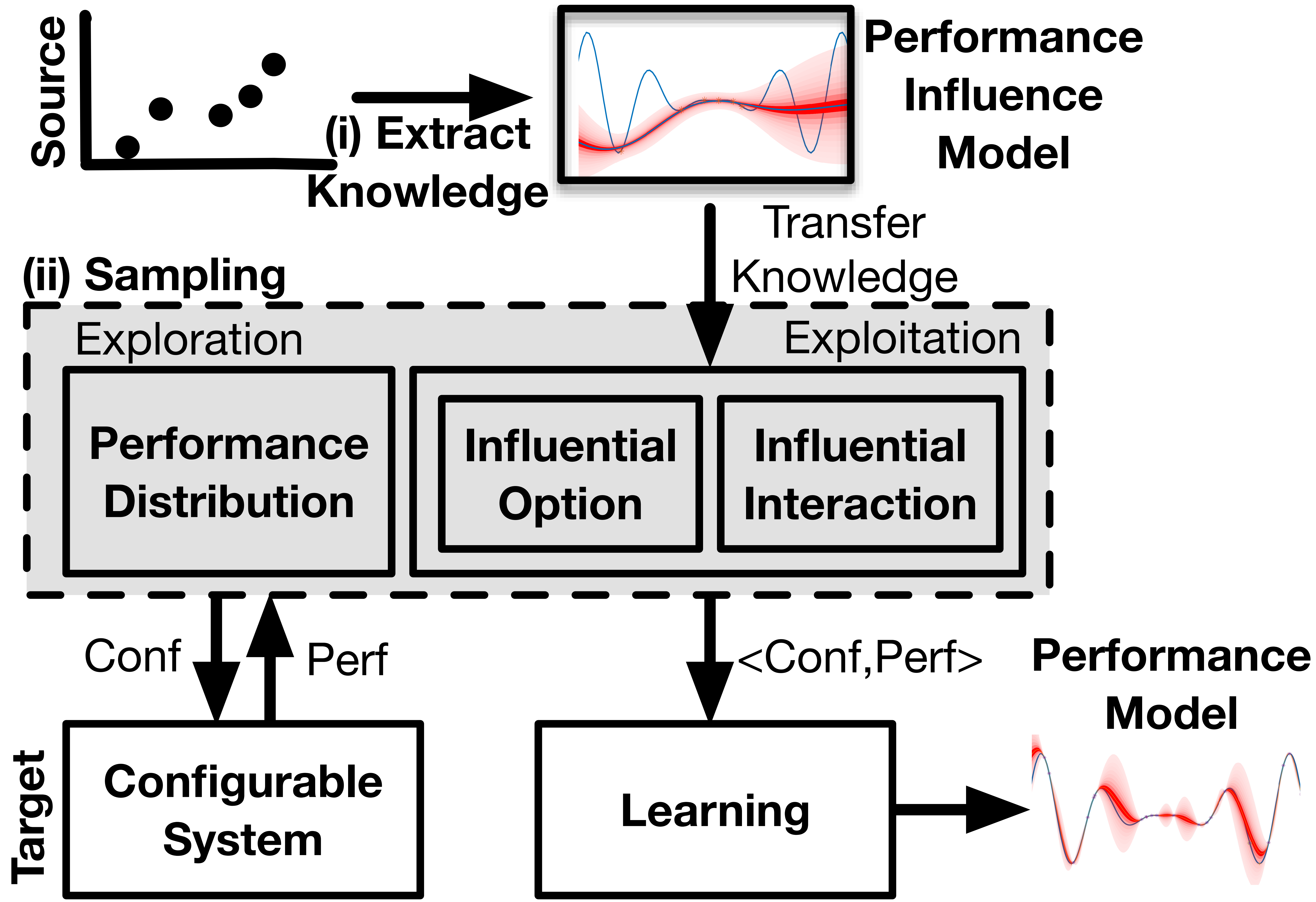
$$\begin{array}{ccccccccccccccccccccc}
 & O_1 & \times & O_2 & \times & O_3 & \times & O_4 & \times & O_5 & \times & O_6 & \times & O_7 & \times & O_8 & \times & O_9 & \times & O_{10} & \\
\hline
c_1 & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \hat{f}_t(c_1) = 10.4 \\
c_2 & 1 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \hat{f}_t(c_2) = 8.1 \\
c_3 & 0 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \hat{f}_t(c_3) = 11.6 \\
c_4 & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \hat{f}_t(c_4) = 12.6 \\
c_5 & 0 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \hat{f}_t(c_5) = 11.7 \\
c_6 & 1 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \hat{f}_t(c_6) = 23.7 \\
c_x & 1 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \times & 1 & \times & 0 & \times & 0 & \times & 0 & \hat{f}_t(c_x) = 9.6
\end{array}$$

Exploration vs Exploitation

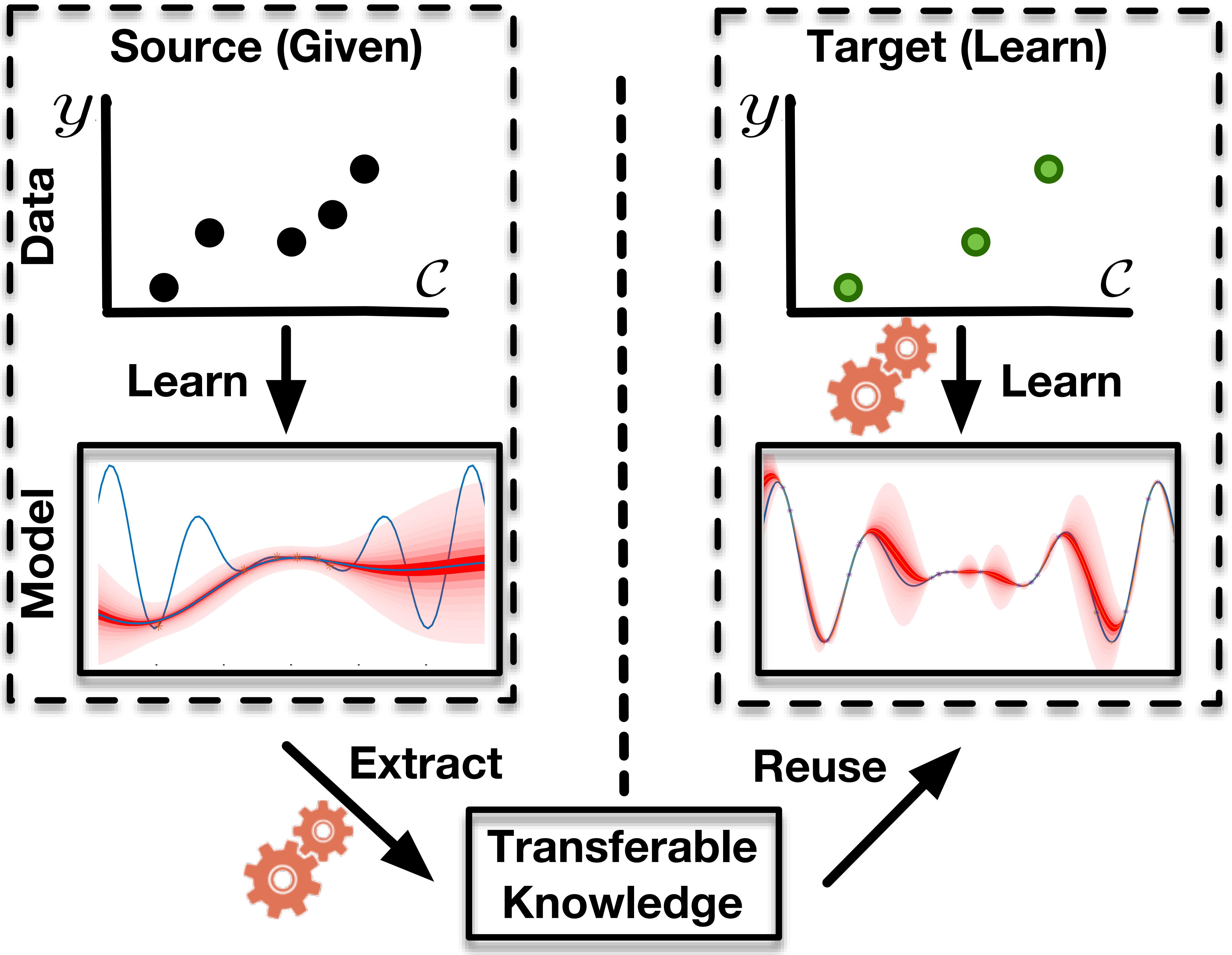
We also explore the configuration space using pseudo-random sampling to detect missing interactions



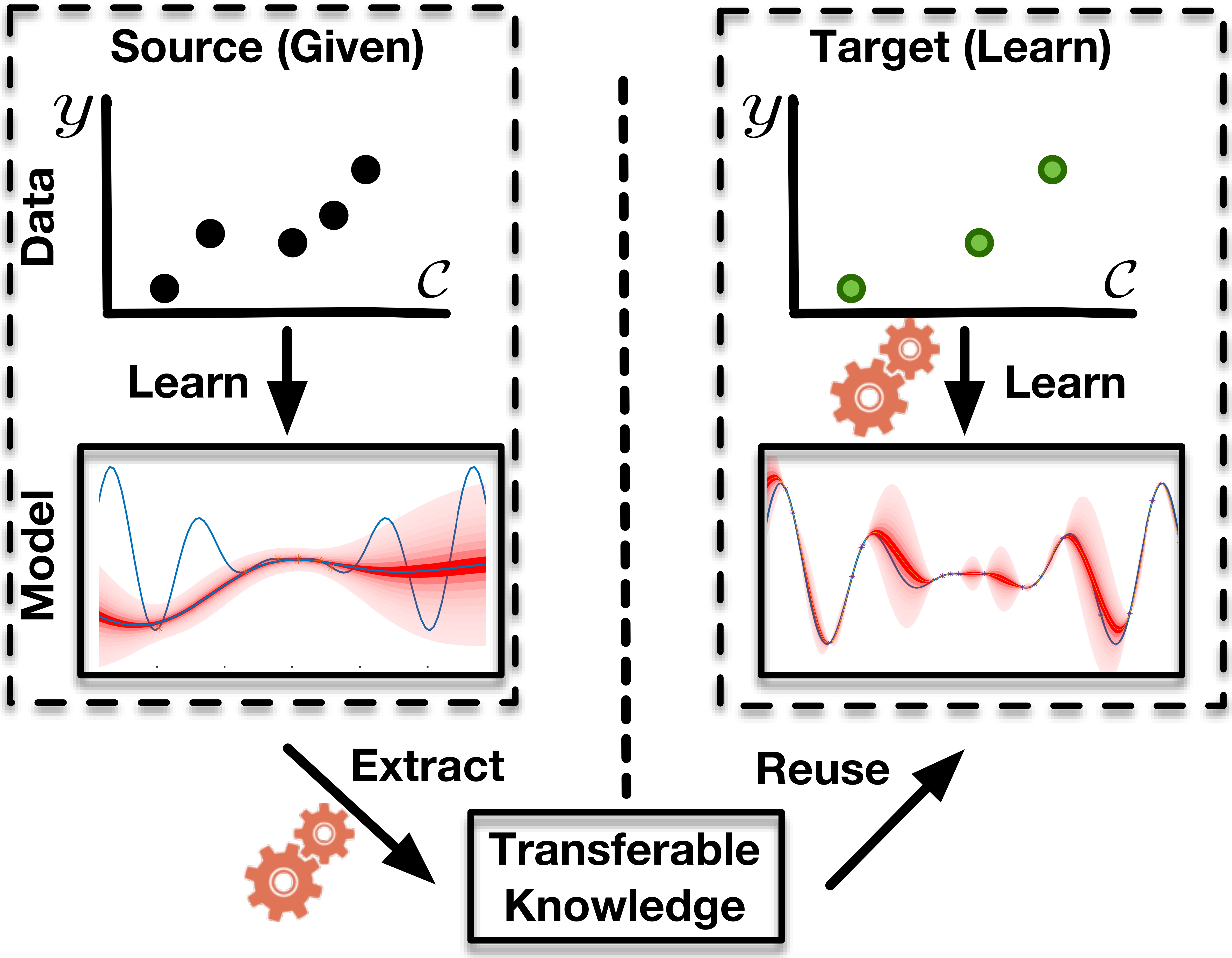
For capturing options and interactions that only appears in the target, L2S relies on exploration (random sampling)



L2S transfers knowledge about the structure of performance models from the source to guide the sampling in the target environment



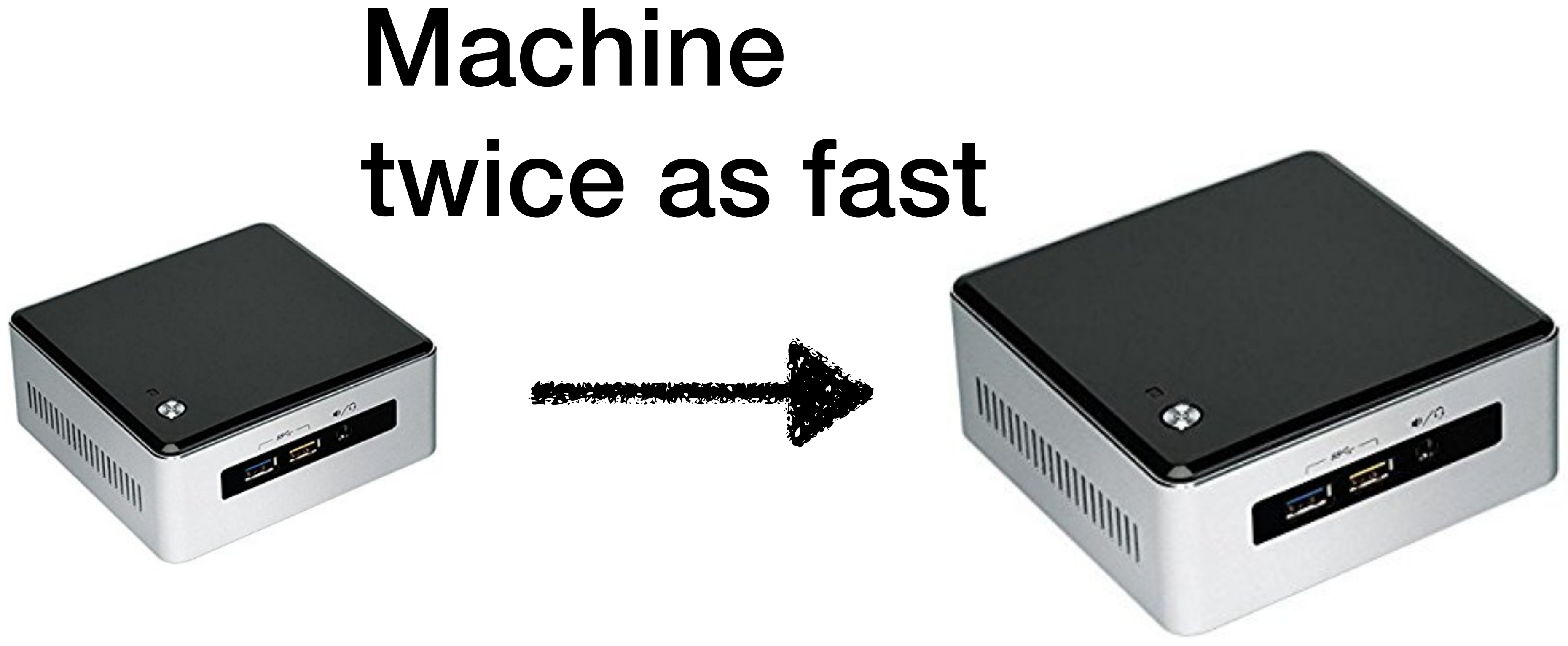
L2S transfers knowledge about the structure of performance models from the source to guide the sampling in the target environment

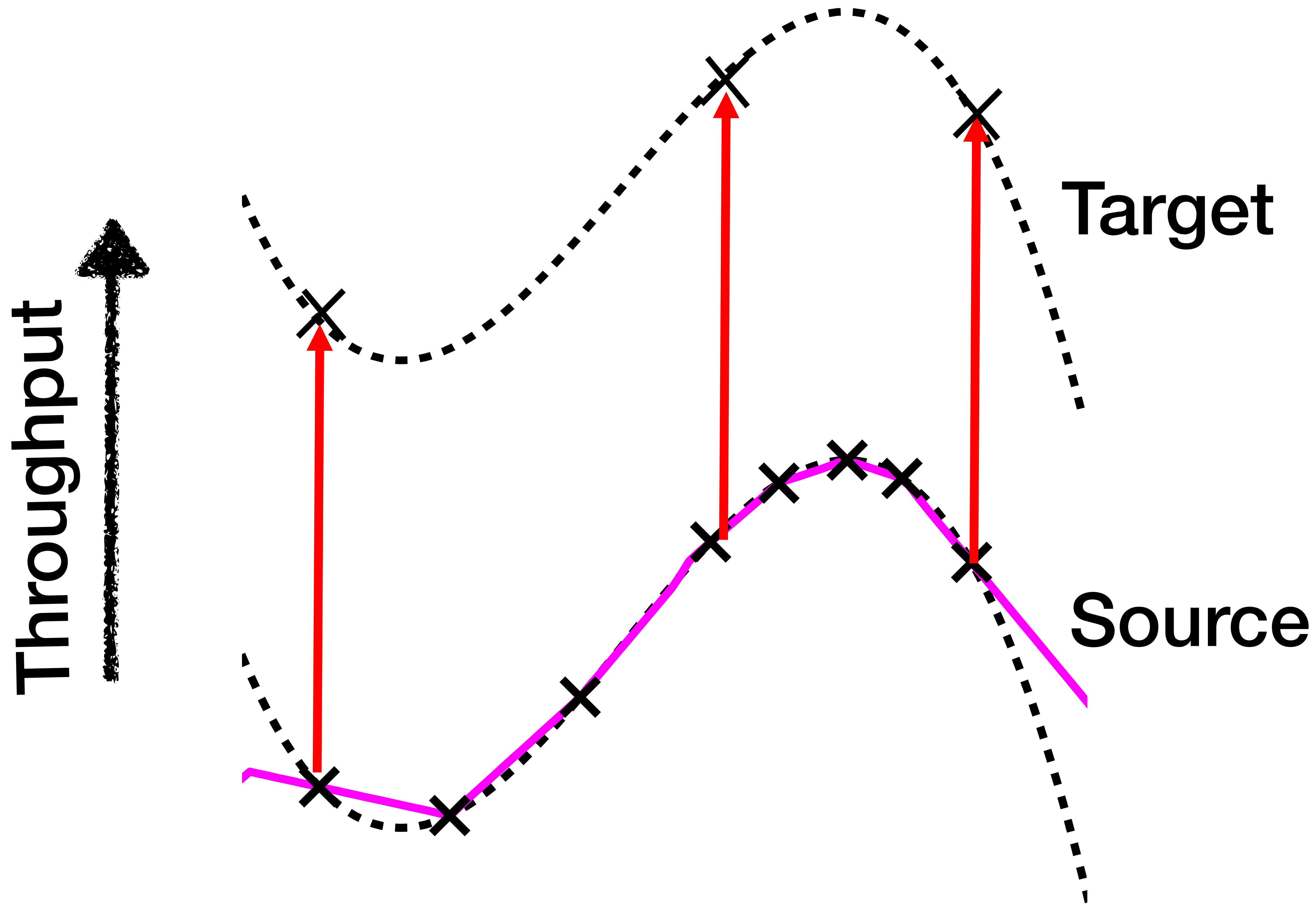


$$\hat{f}_s(\cdot) = 1.2 + 3o_1 + 5o_3 + 0.9o_7 + 0.8o_3o_7 + 4o_1o_3o_7$$

Evaluation:
Other transfer learning approaches also exists

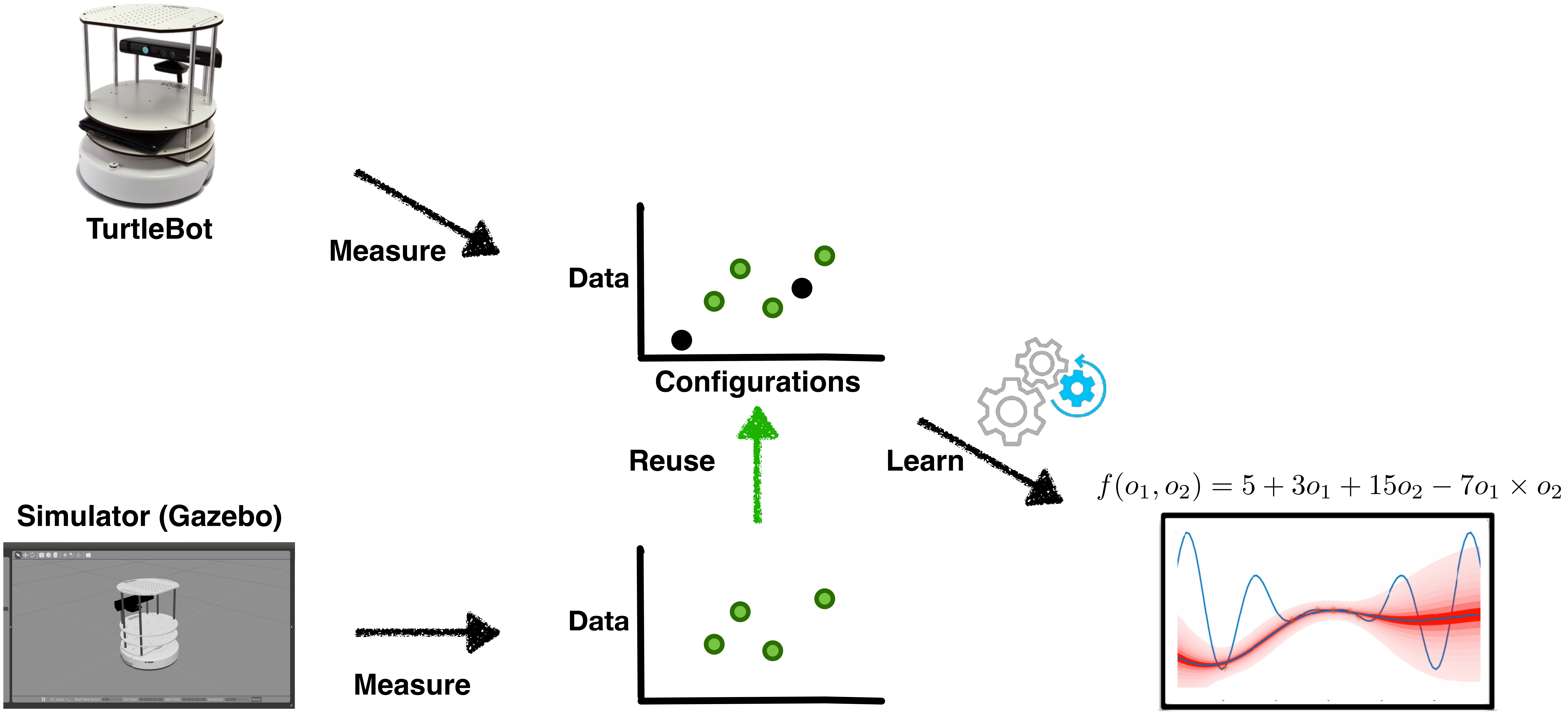
“Model shift” builds a model in the source and uses the shifted model “directly” for predicting the target





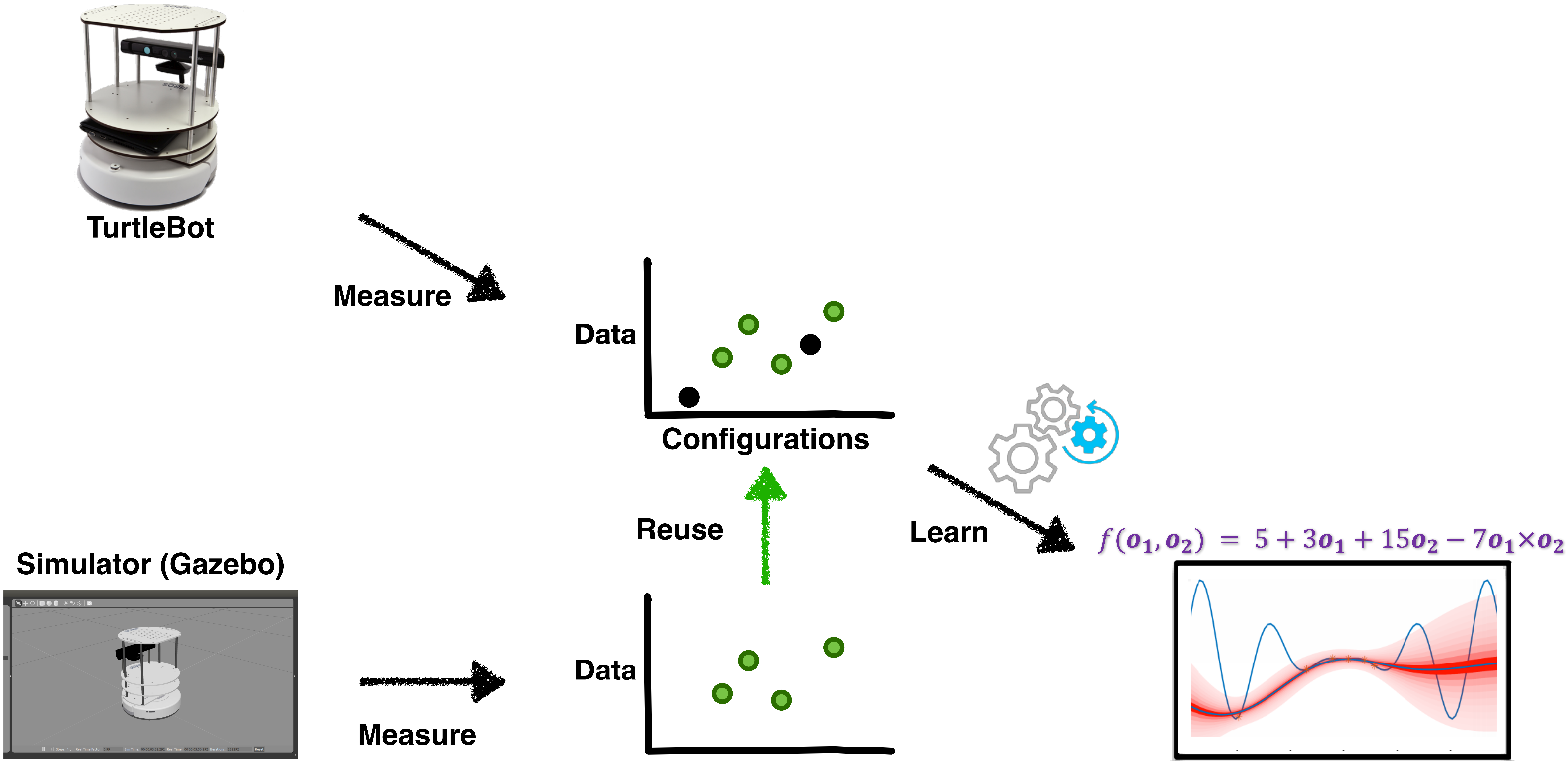
[Pavel Valov, et al. “**Transferring performance prediction models**…”, ICPE’17]

“Data reuse” combines the data collected in the source with the ones in the target in a “multi-task” setting to predict the target

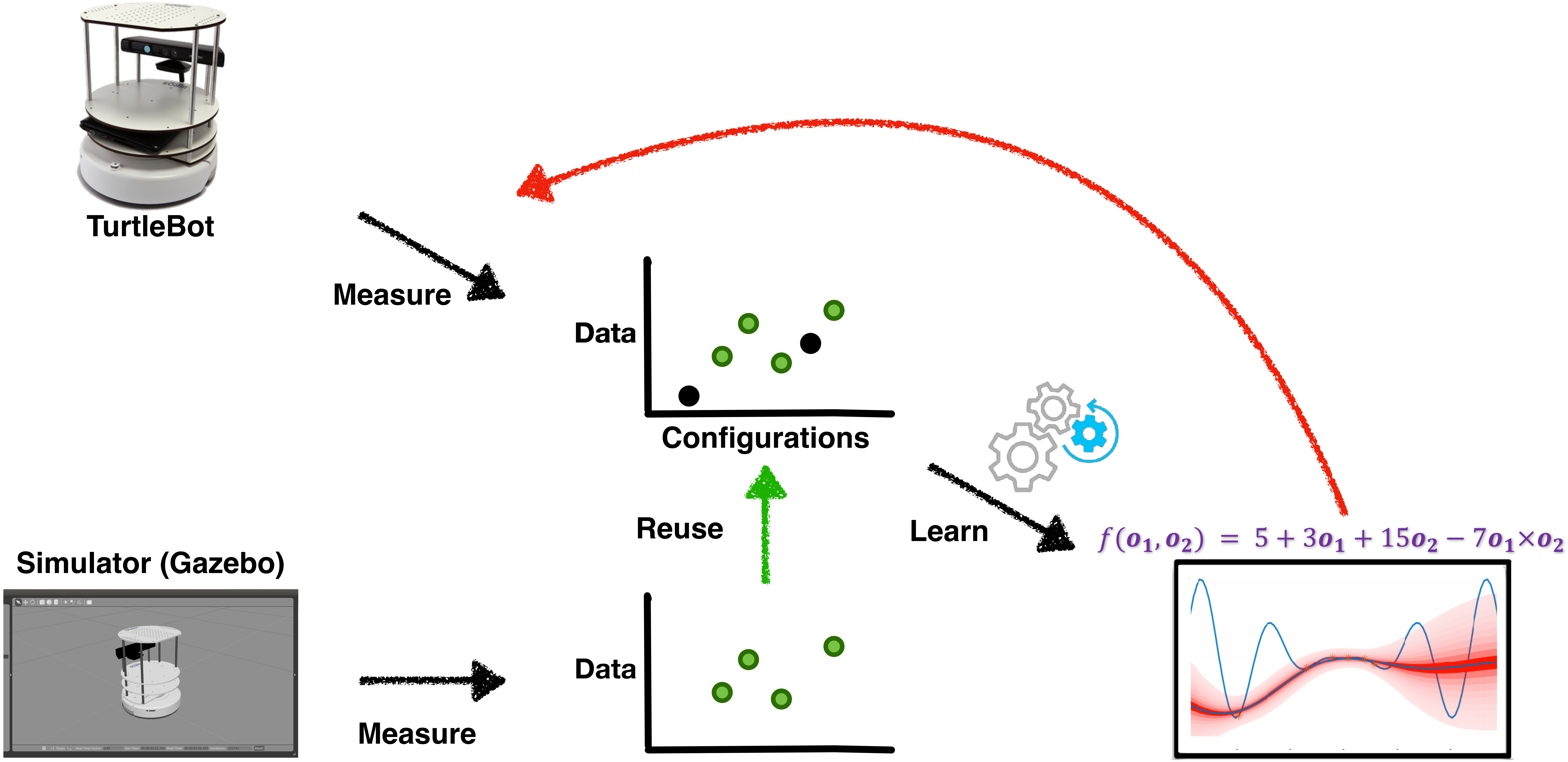


[P. Jamshidi, et al., “**Transfer learning for improving model predictions** ….”, SEAMS’17]

“Data reuse” with guided sampling

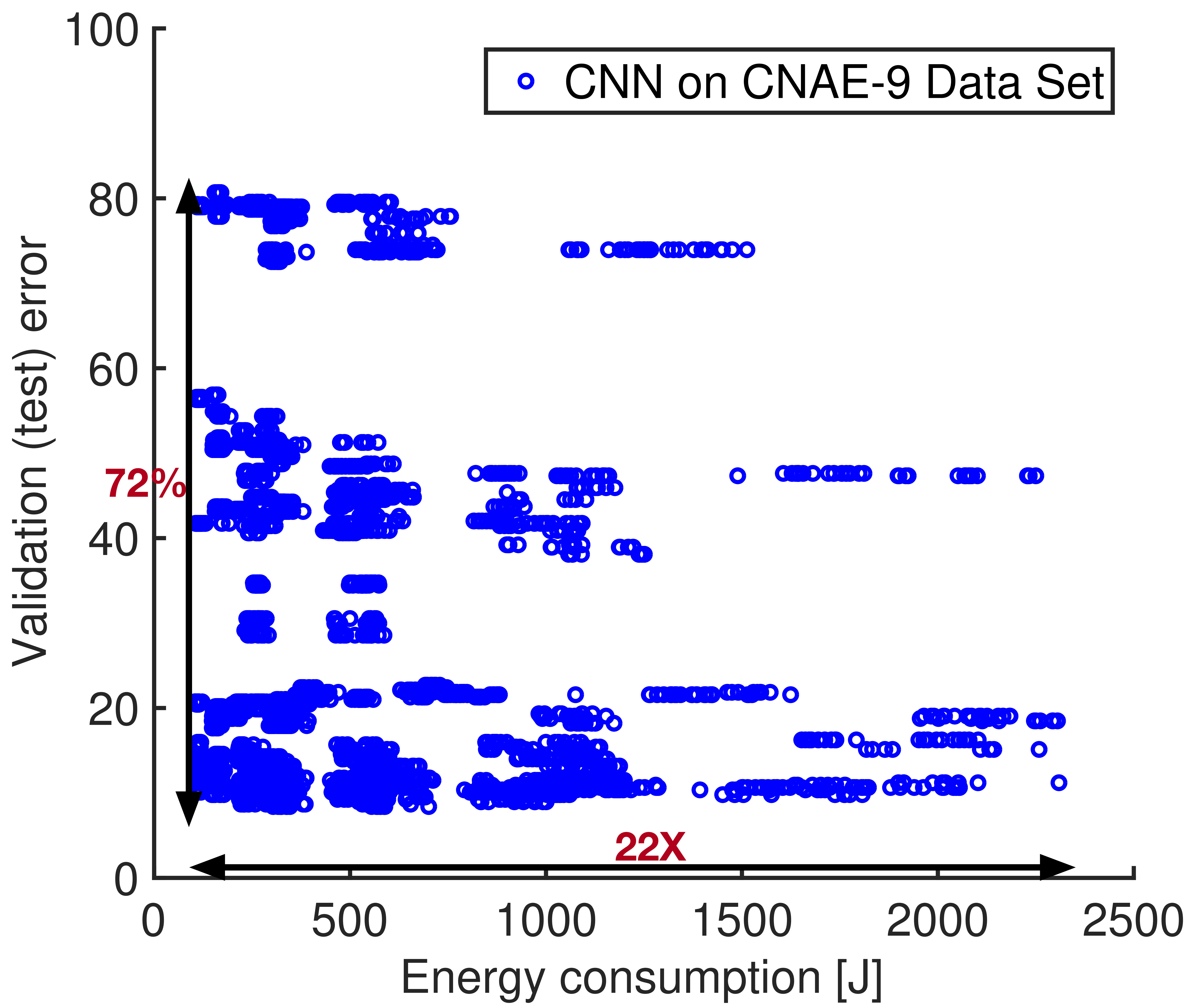


“Data reuse” with guided sampling

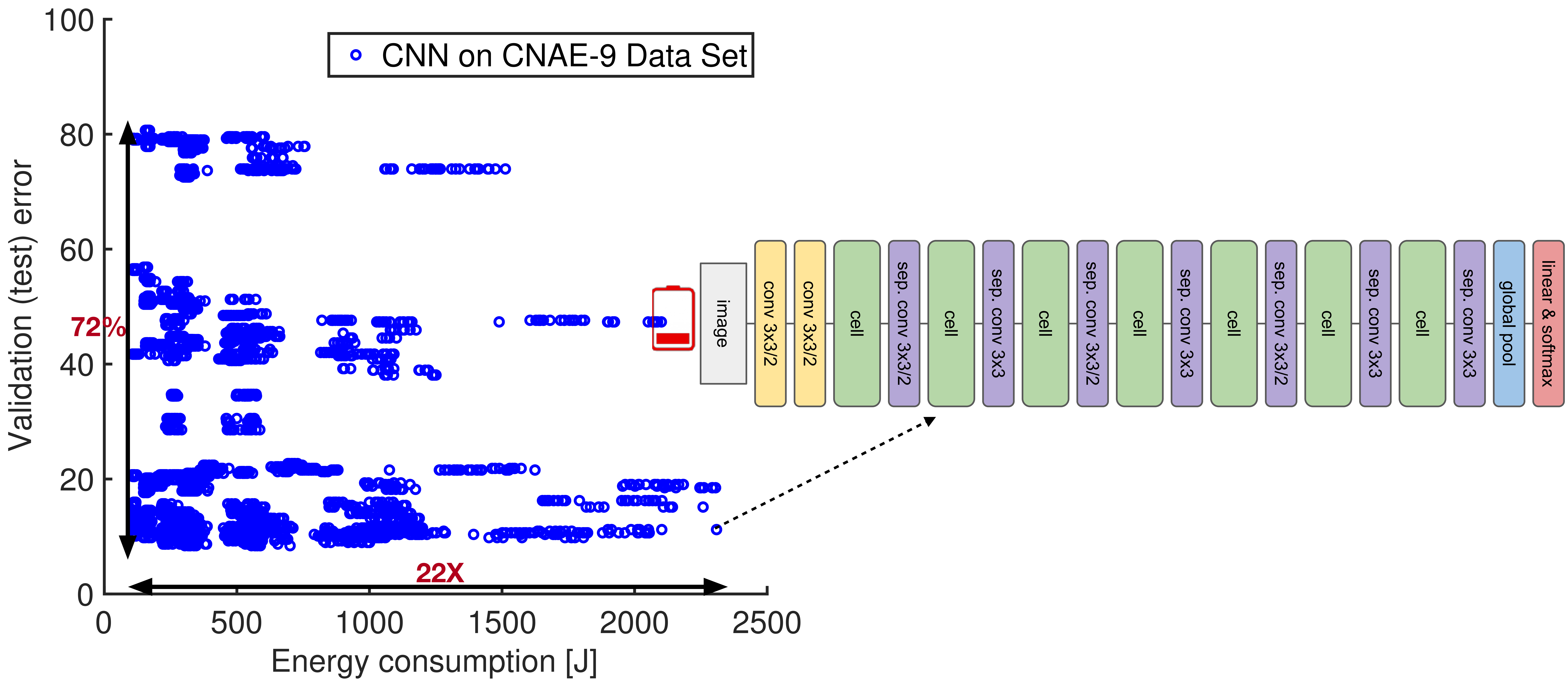


Evaluation:
Learning performance behavior of Machine Learning Systems

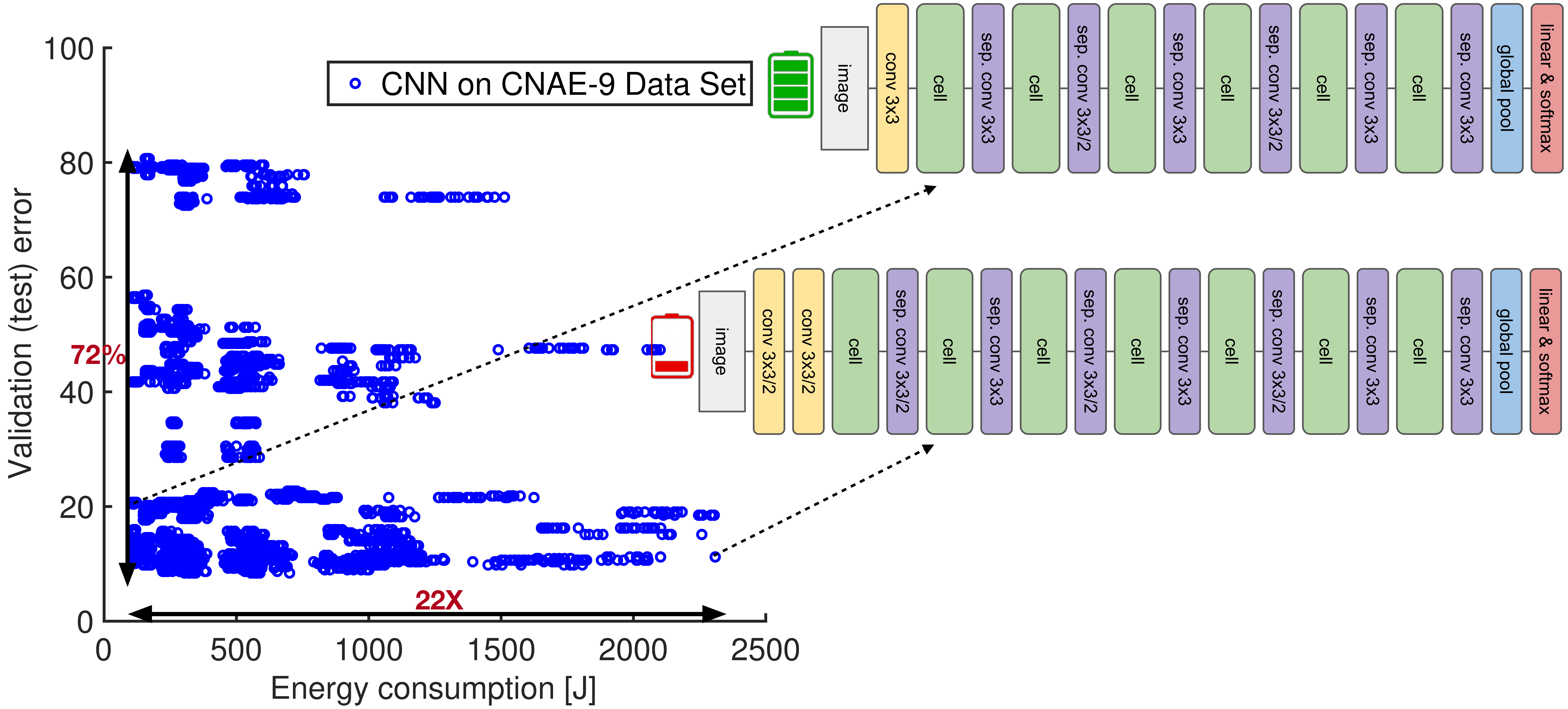
Configurations of deep neural networks affect accuracy and energy consumption



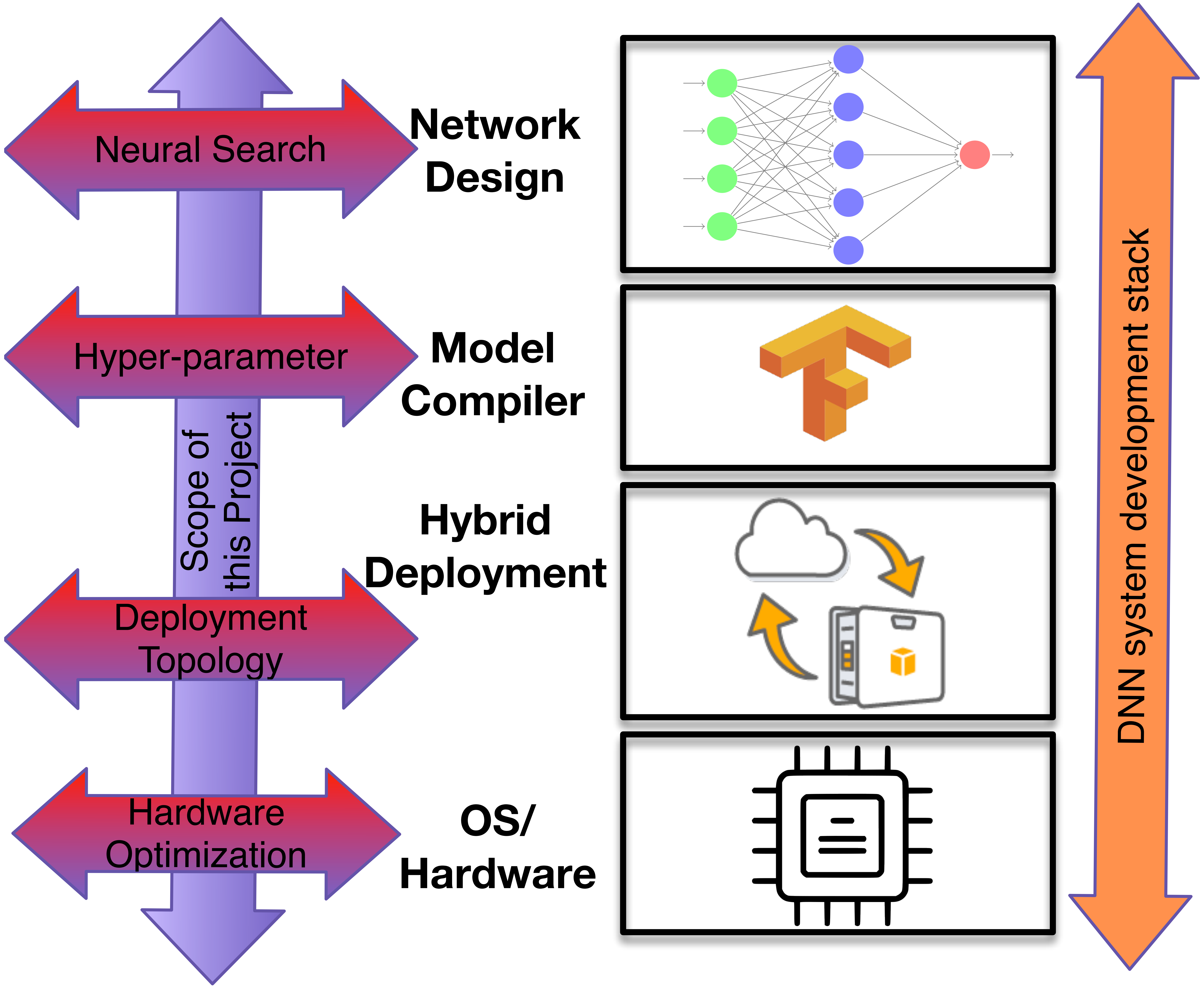
Configurations of deep neural networks affect accuracy and energy consumption



Configurations of deep neural networks affect accuracy and energy consumption

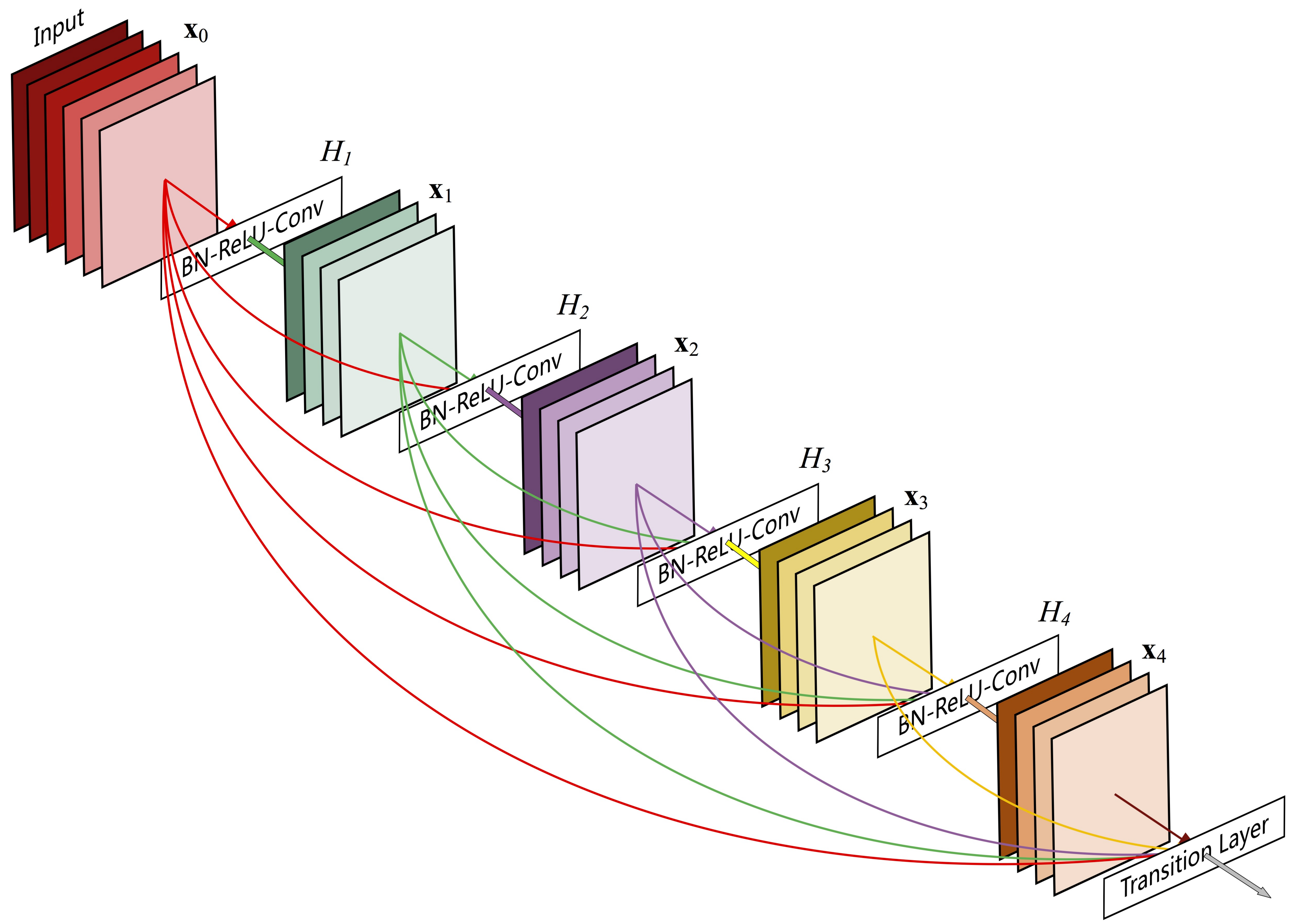


Deep neural network as a highly configurable system



DNN measurements are costly

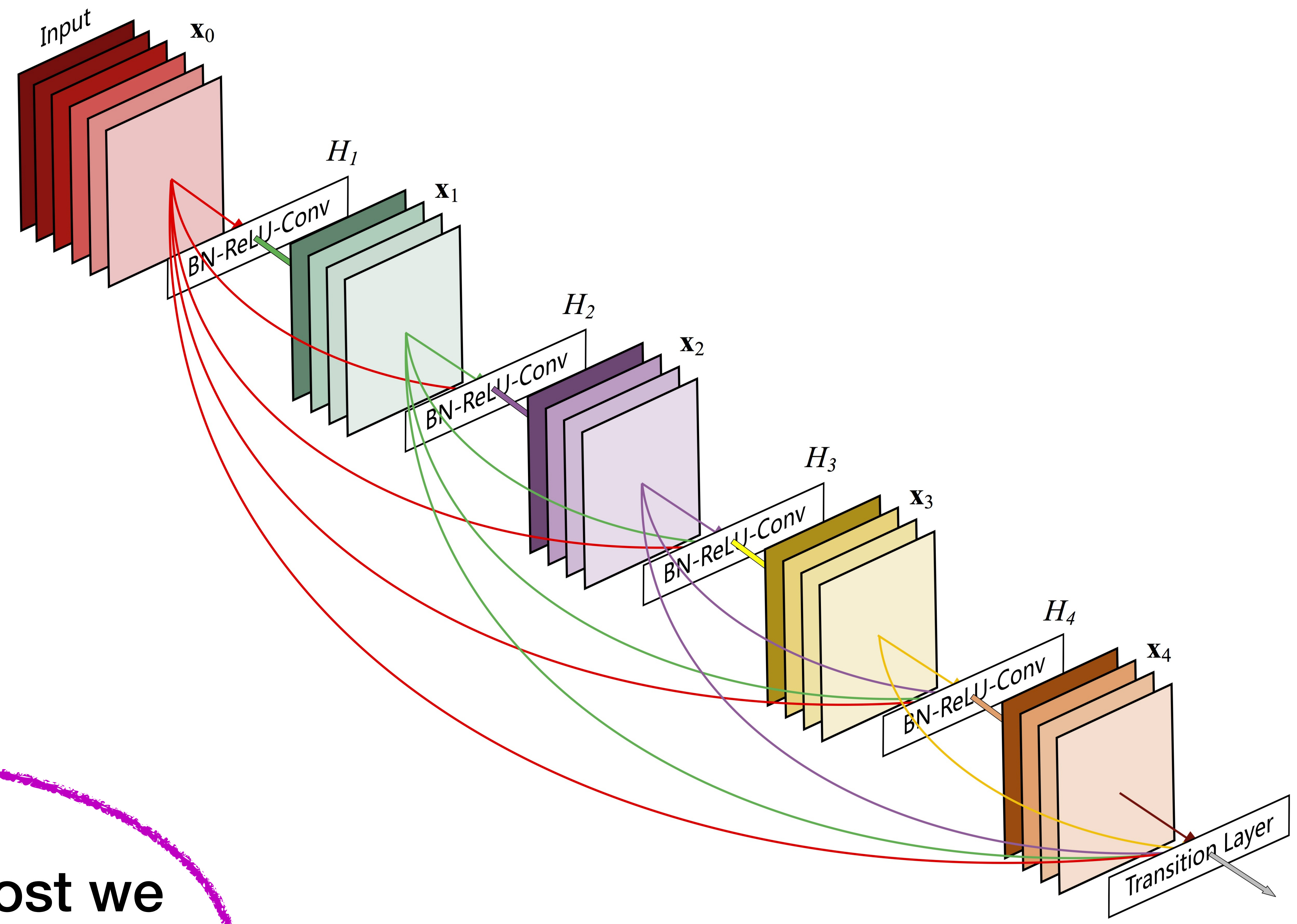
Each sample cost ~1h
4000 * 1h ~= 6 months



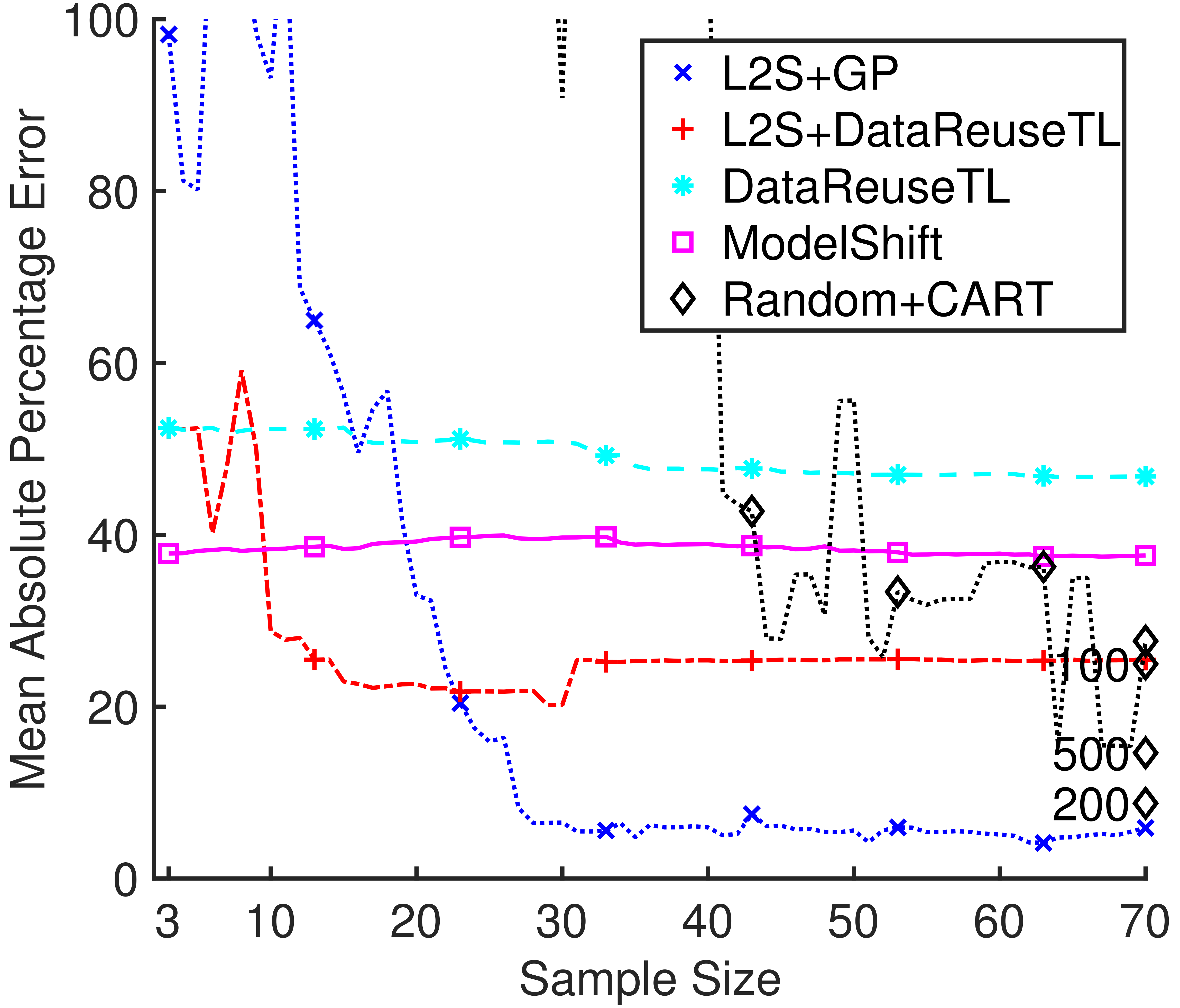
DNN measurements are costly

Each sample cost ~1h
4000 * 1h ~= 6 months

Yes, that's the cost we paid for conducting our measurements!

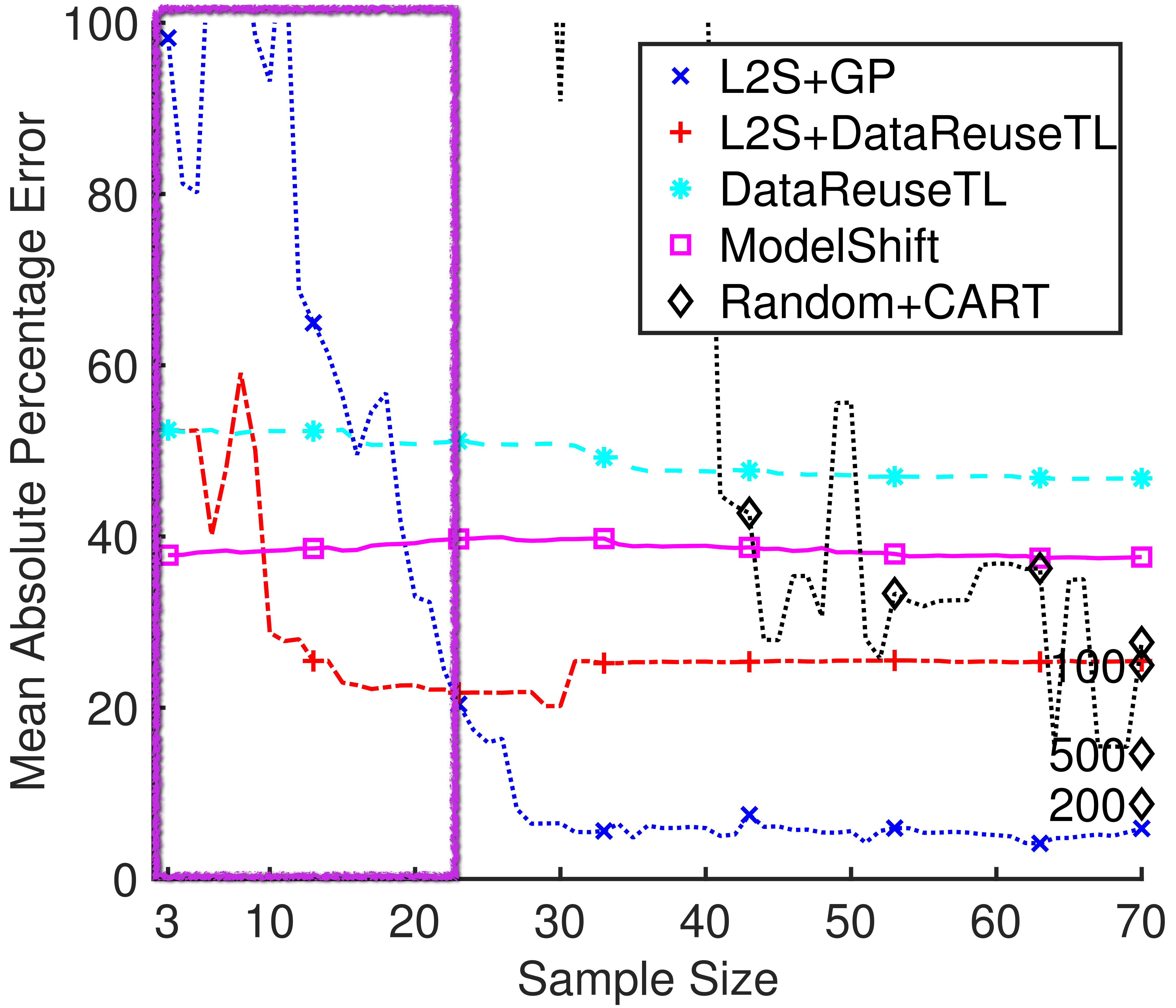


L2S enables learning a more accurate model with less samples exploiting the knowledge from the source



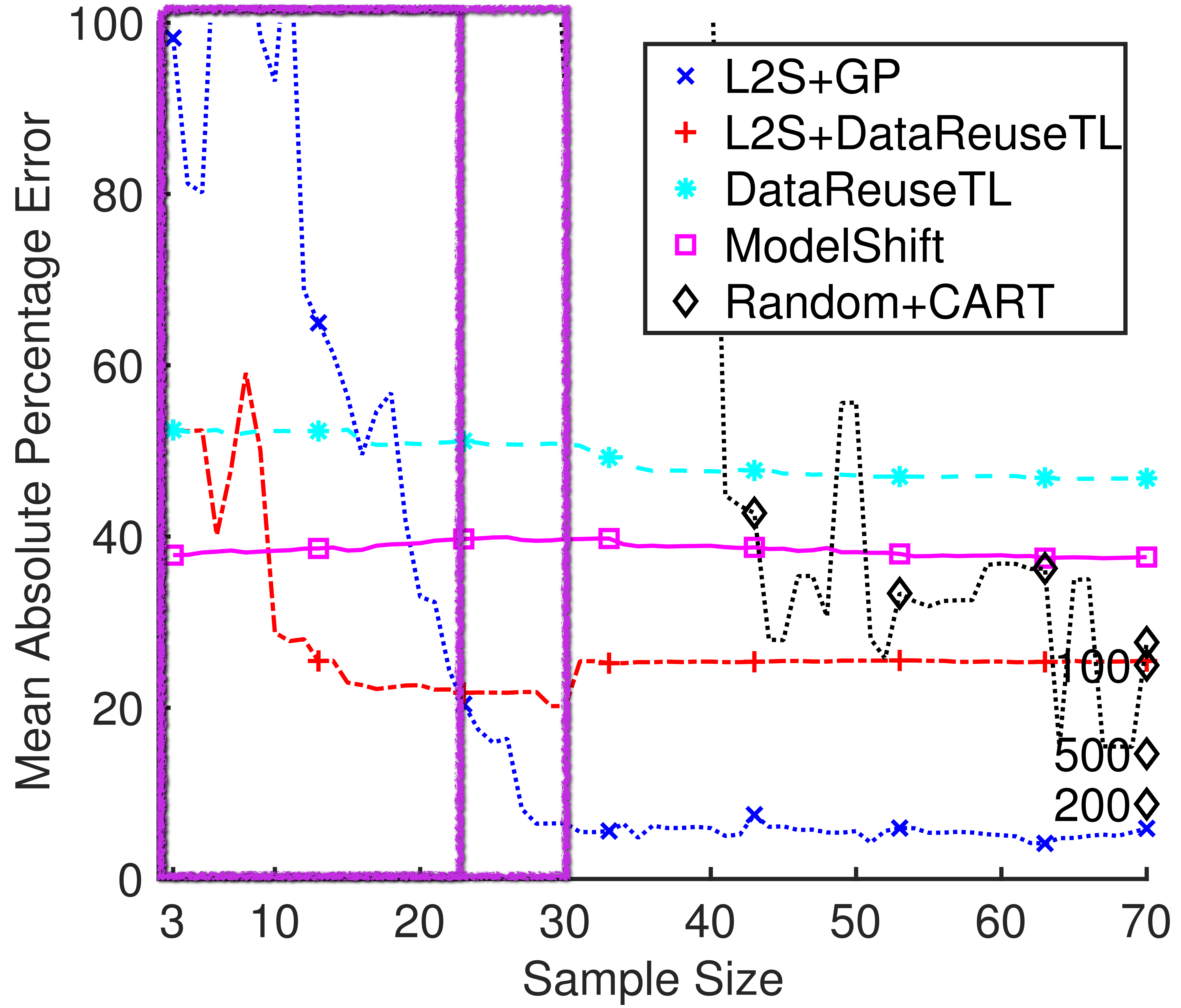
Convolutional Neural Network

L2S enables learning a more accurate model with less samples exploiting the knowledge from the source



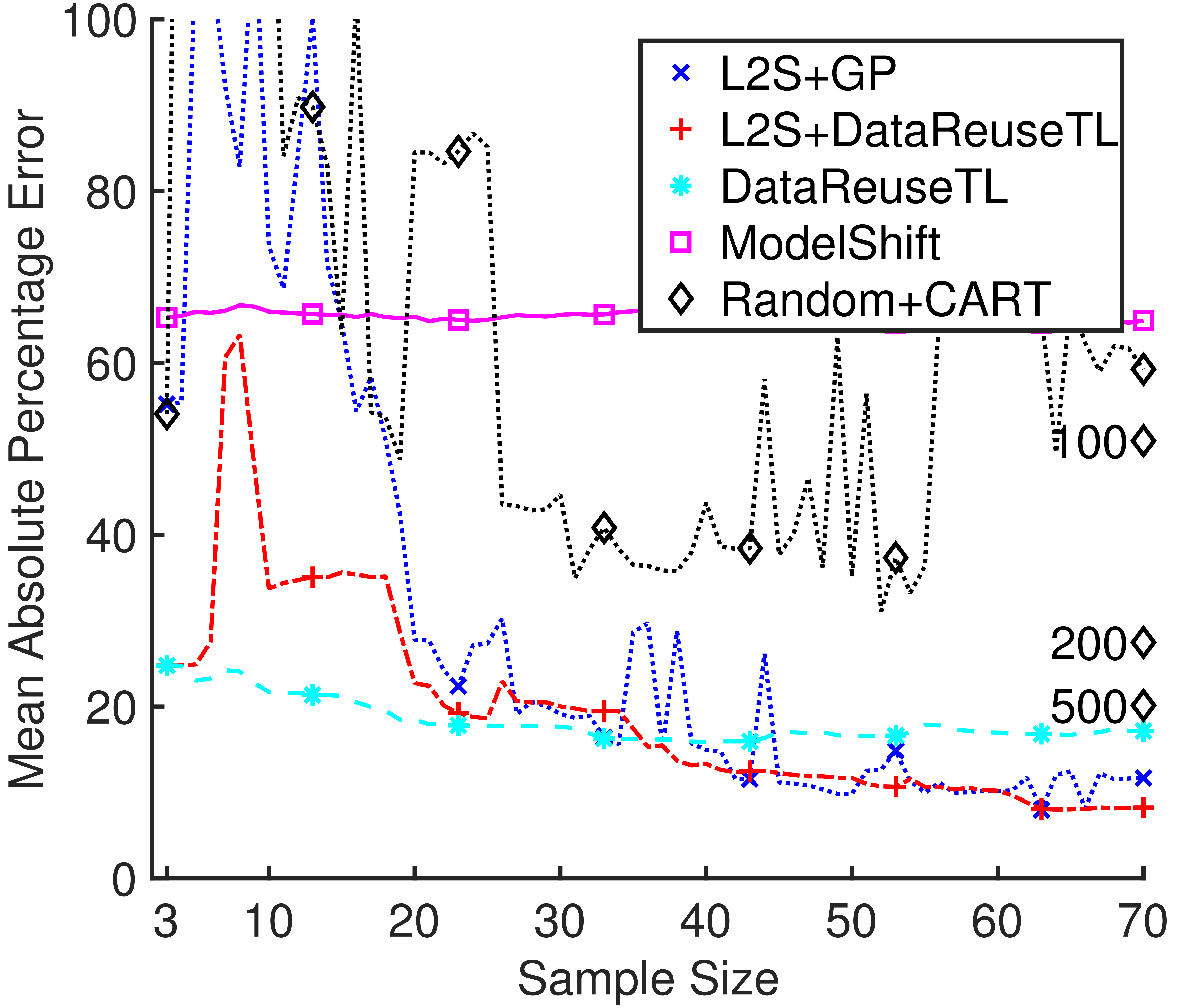
Convolutional Neural Network

L2S enables learning a more accurate model with less samples exploiting the knowledge from the source



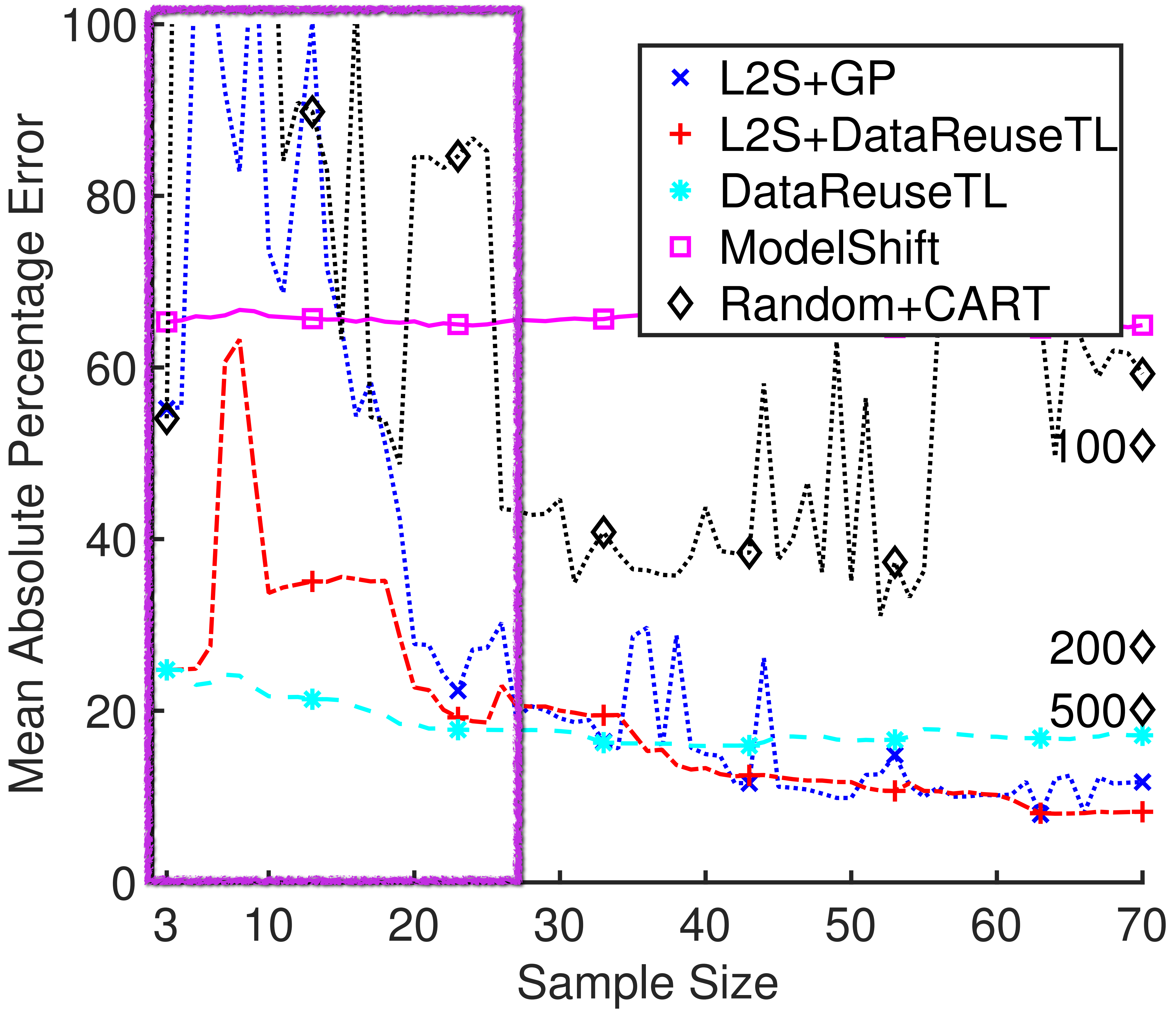
Convolutional Neural Network

L2S may also help data-reuse approach to learn faster



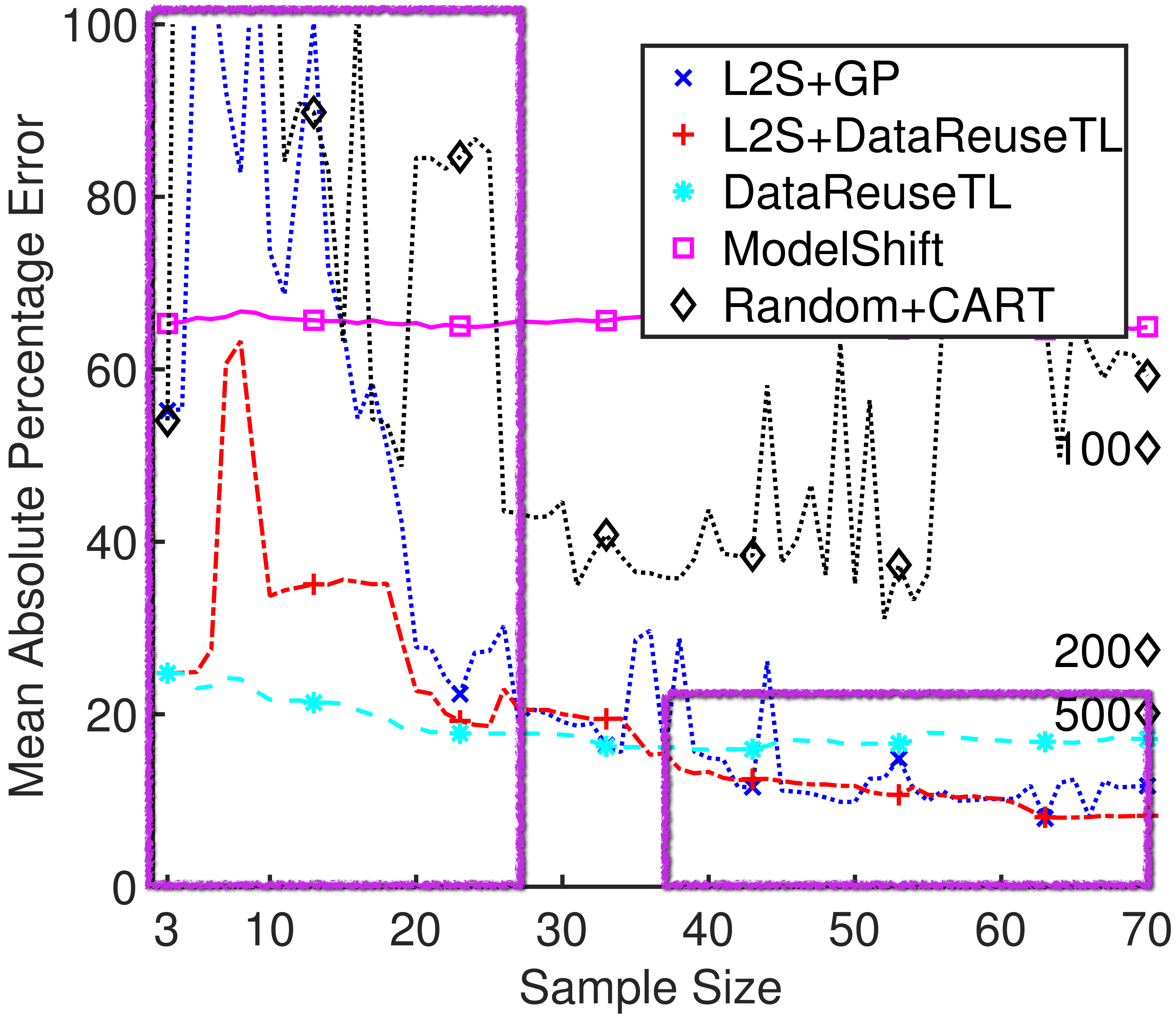
XGBoost

L2S may also help data-reuse approach to learn faster



XGBoost

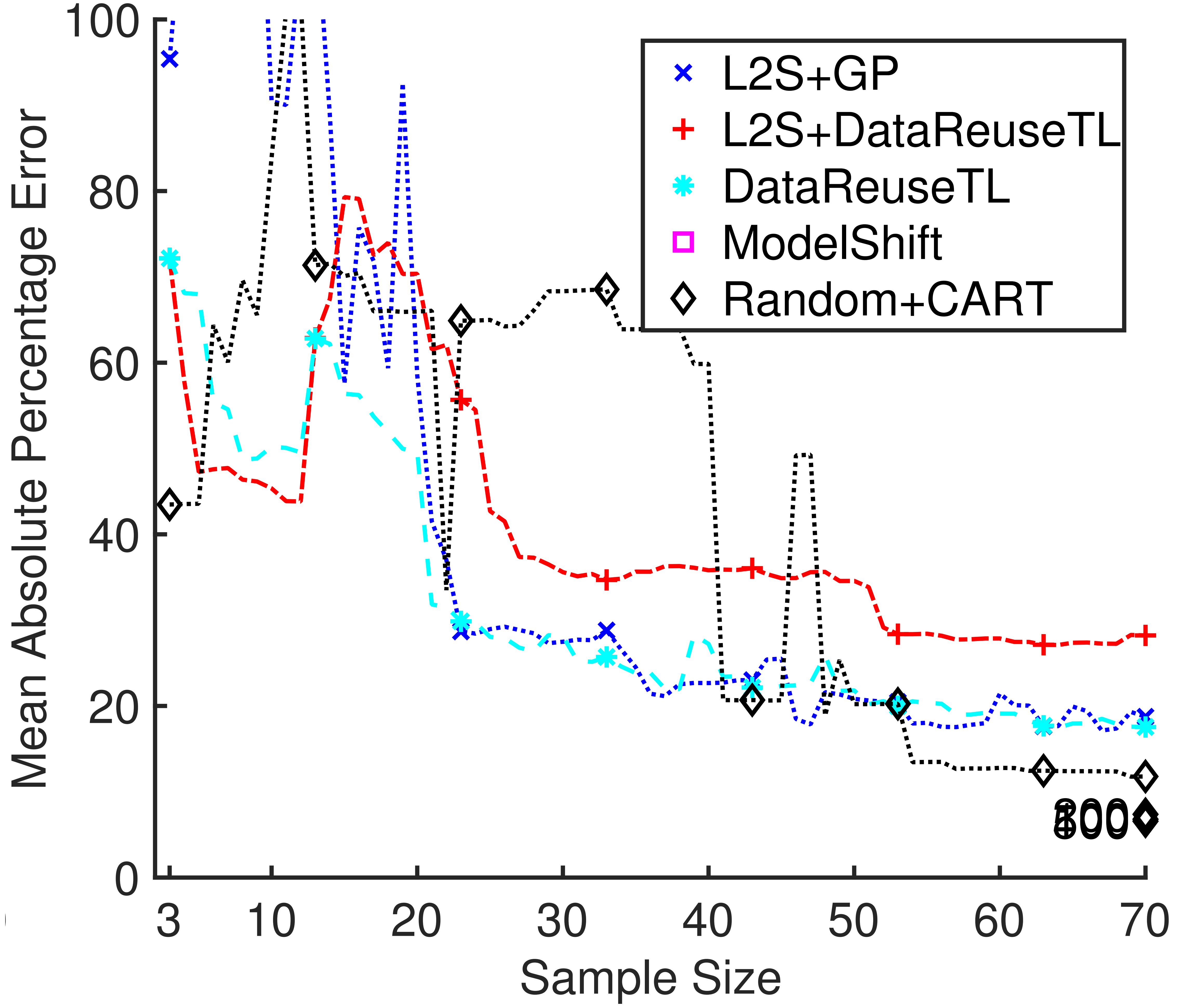
L2S may also help data-reuse approach to learn faster



XGBoost

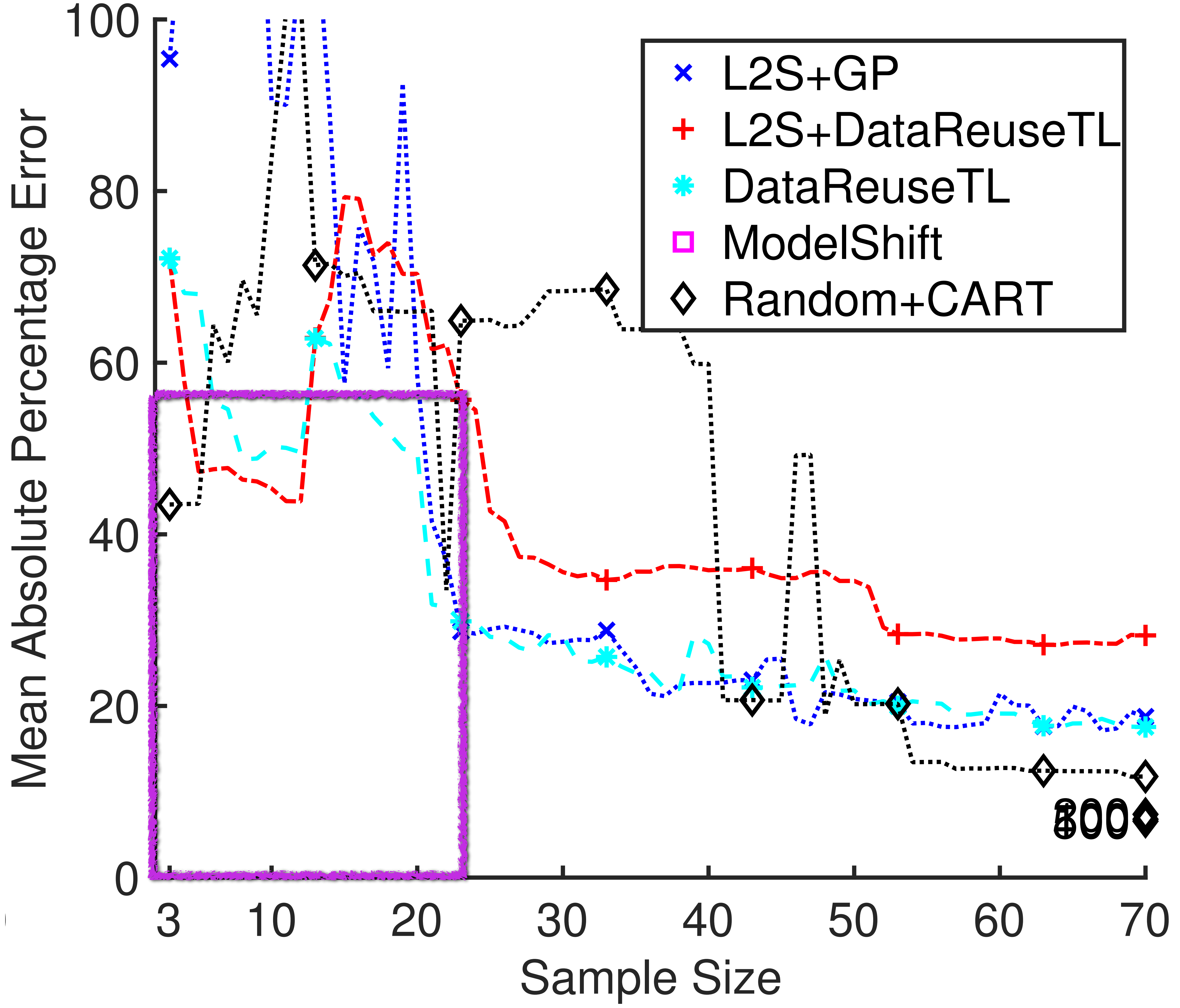
Evaluation:
Learning performance behavior of Big Data Systems

Some environments the similarities across environments may be too low and this results in “negative transfer”



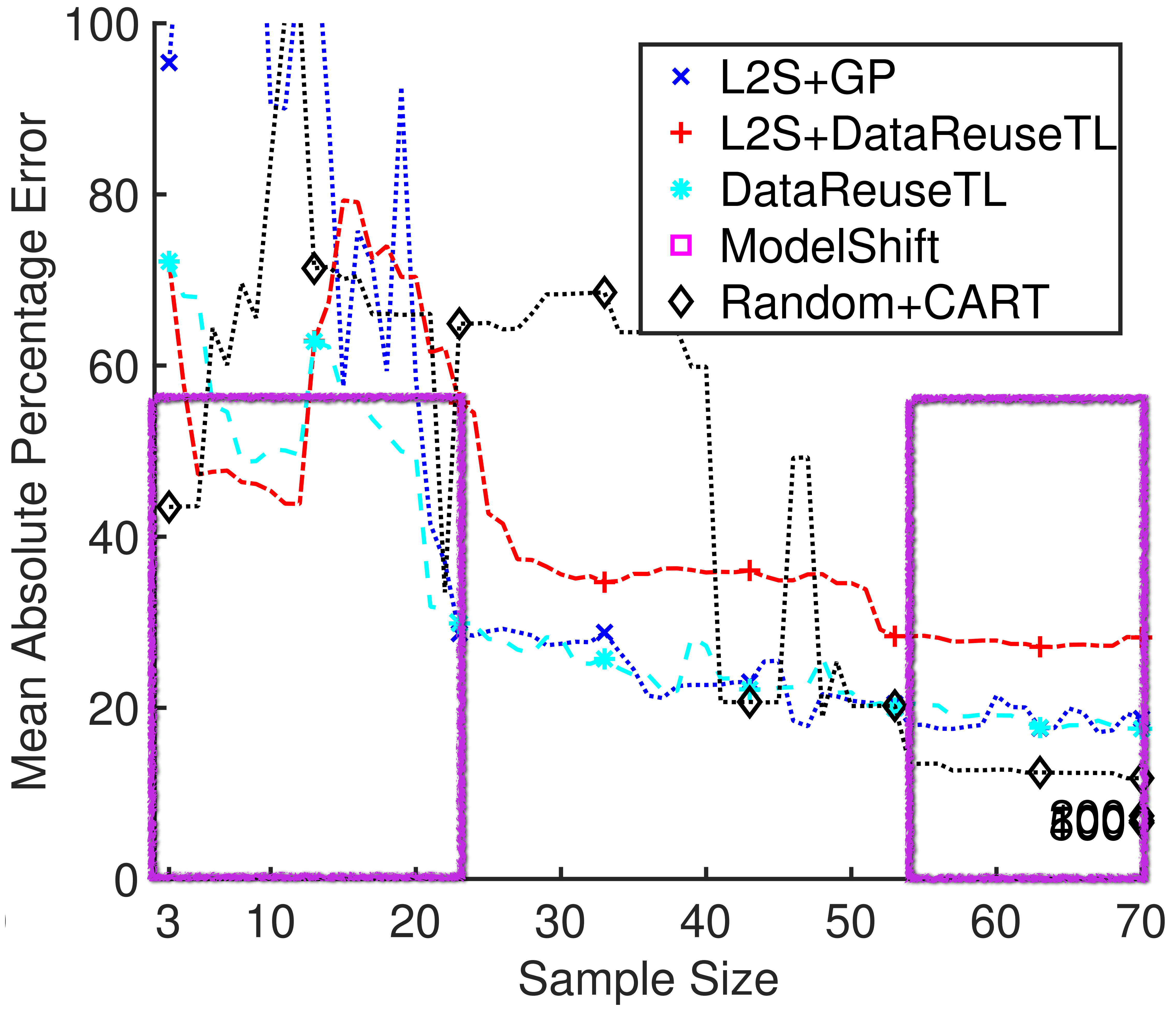
Apache Storm

Some environments the similarities across environments may be too low and this results in “negative transfer”



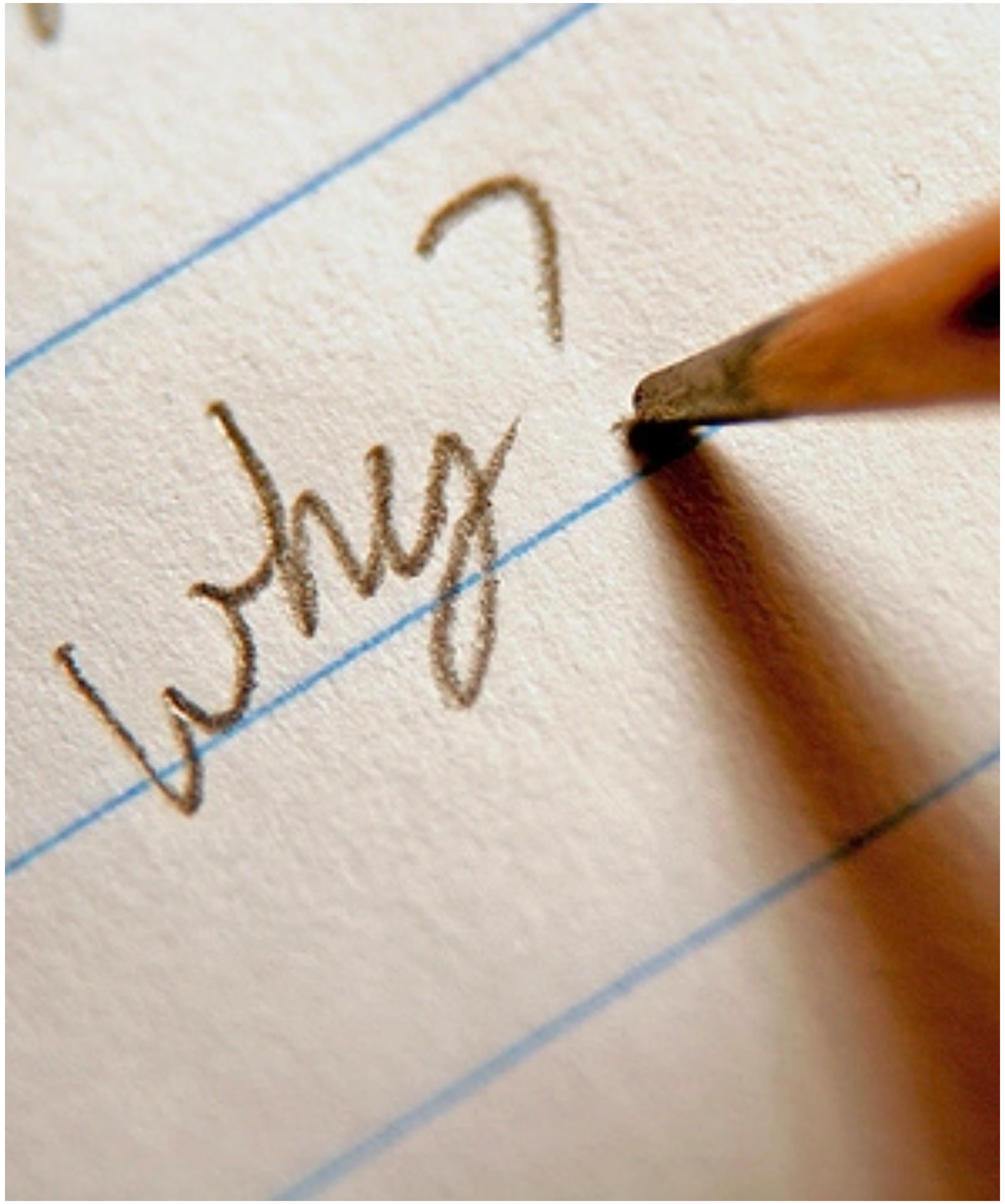
Apache Storm

Some environments the similarities across environments may be too low and this results in "negative transfer"

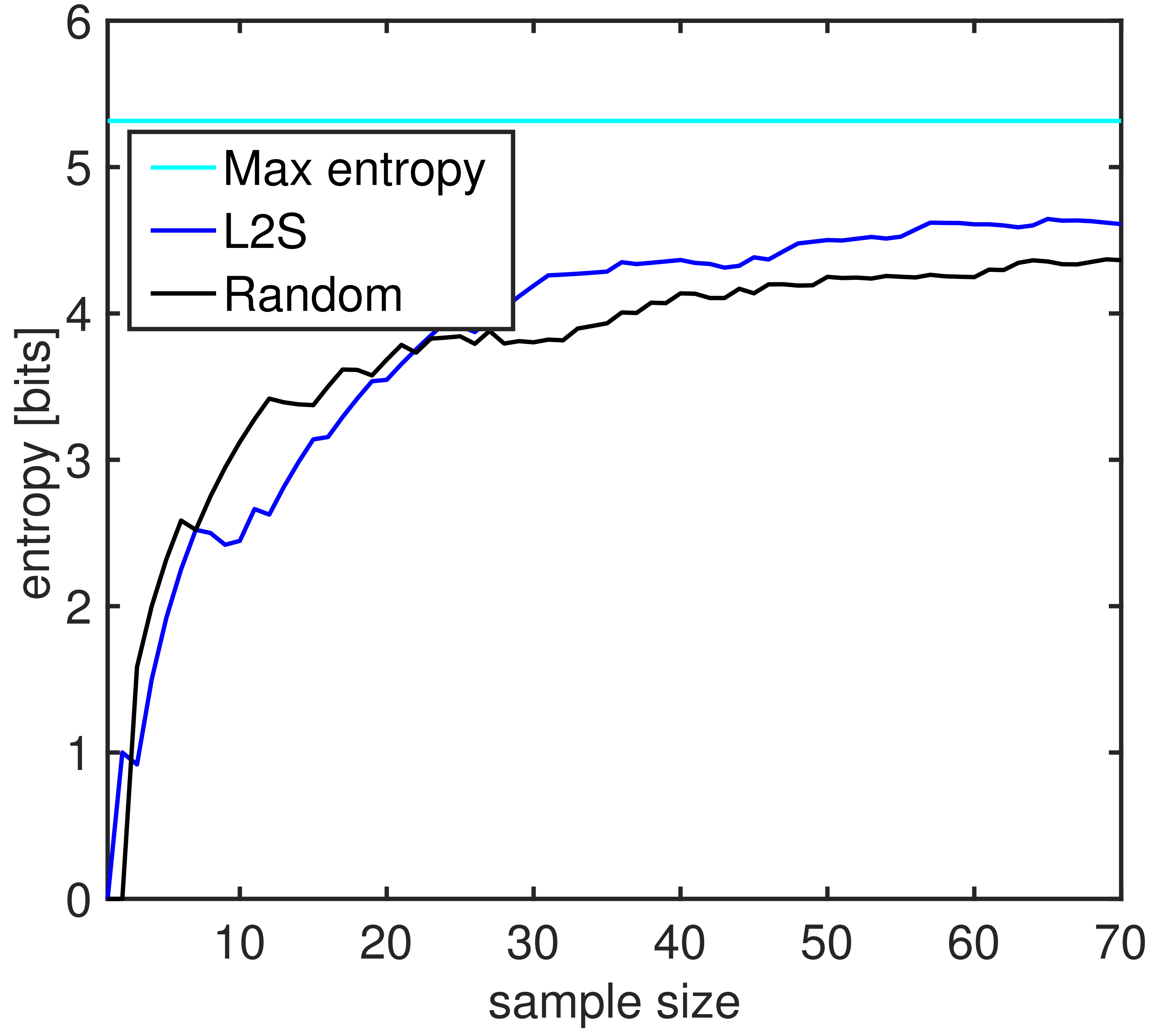


Apache Storm

Why performance models using L2S sample are more accurate?

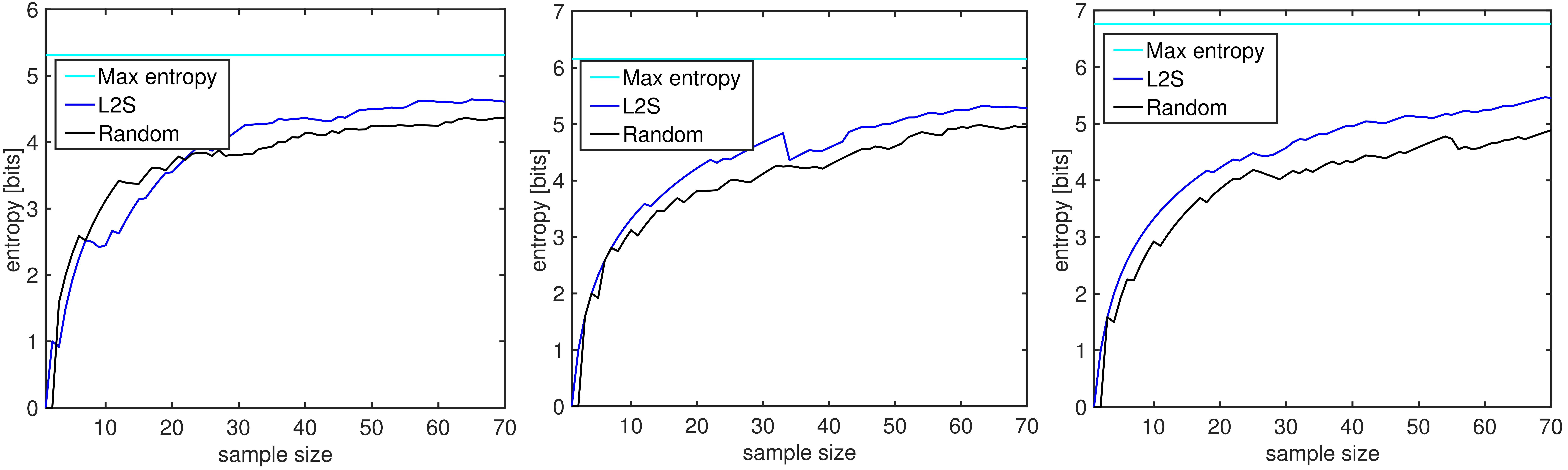


The samples generated by L2S contains more information… “entropy <-> information gain”



DNN

The samples generated by L2S contains more information… “entropy <-> information gain”



DNN **XGboost** **Storm**

Limitations

- Limited number of systems and environmental changes
    - Synthetic models
        - https://github.com/pooyanjamshidi/GenPerf
- Binary options
    - Non-binary options -> binary
- Negative transfer

Details: [FSE '18]

Learning to Sample: Exploiting Similarities across Environments to Learn Performance Models for Configurable Systems

Pooyan Jamshidi
University of South Carolina
USA

Miguel Velez
Christian Kästner
Carnegie Mellon University
USA

Norbert Siegmund
Bauhaus-University Weimar
Germany

ABSTRACT

Most software systems provide options that allow users to tailor the system in terms of functionality and qualities. The increased flexibility raises challenges for understanding the configuration space and the effects of options and their interactions on performance and other non-functional properties. To identify how options and interactions affect the performance of a system, several sampling and learning strategies have been recently proposed. However, existing approaches usually assume a fixed environment (hardware, workload, software release) such that learning has to be repeated once the environment changes. Repeating learning and measurement for each environment is expensive and often practically infeasible. Instead, we pursue a strategy that transfers knowledge across environments but sidesteps heavyweight and expensive transfer-

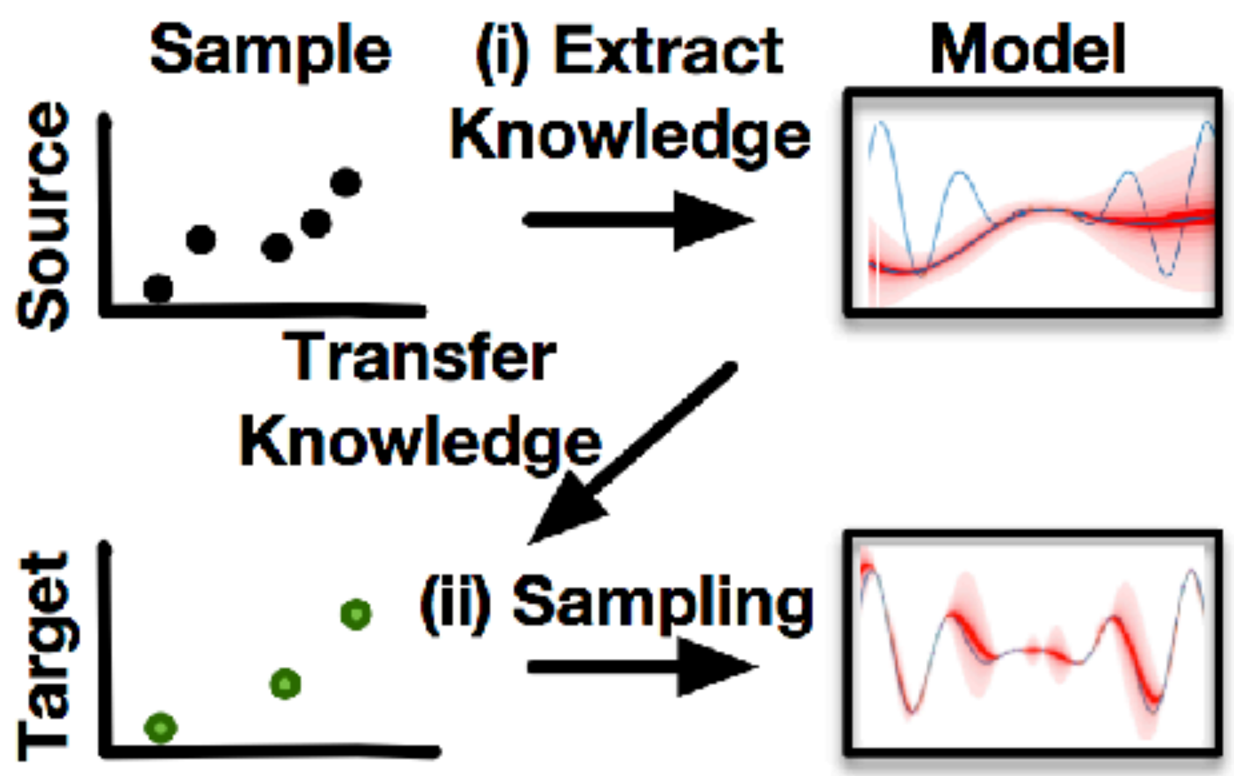
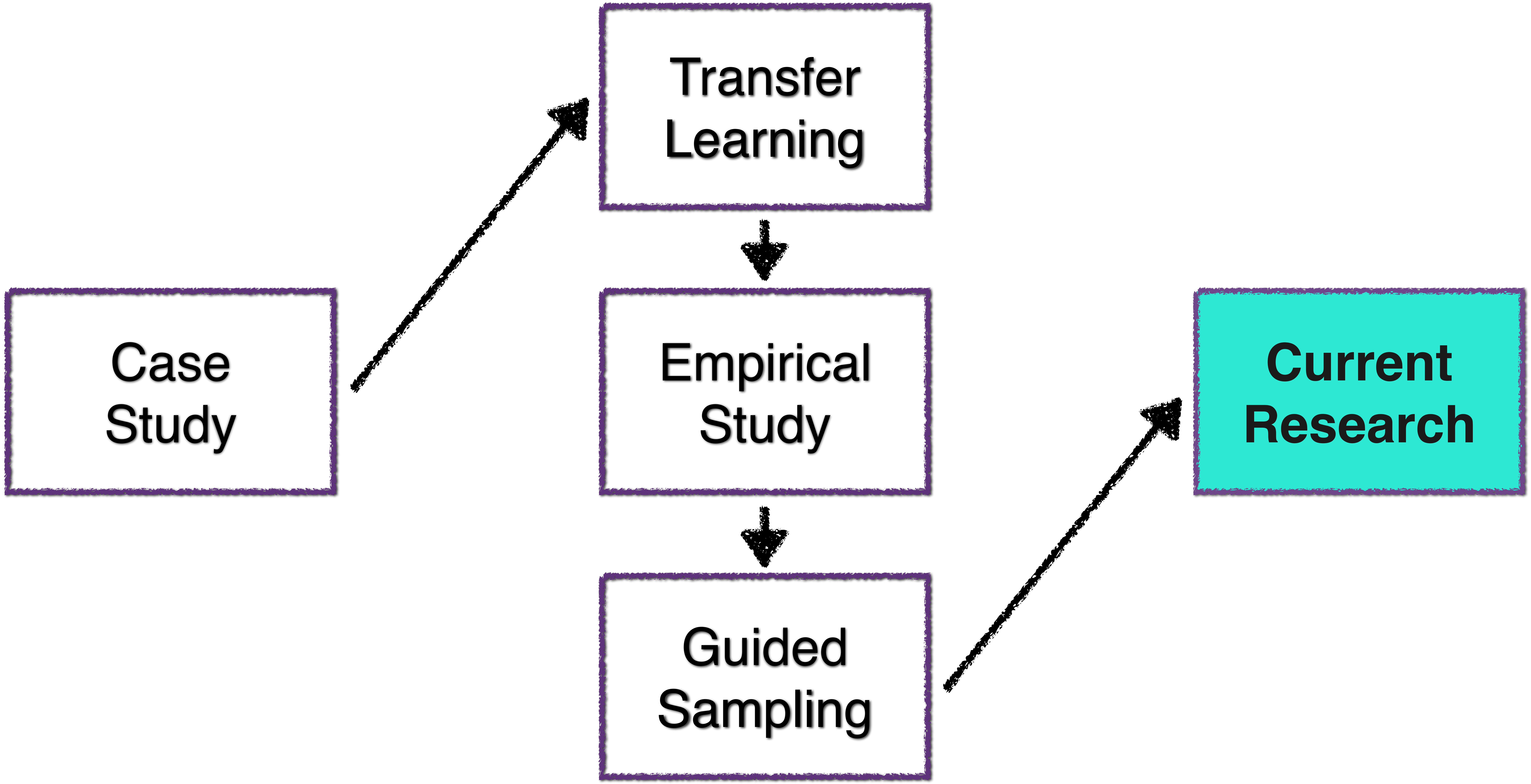


Figure 1: L2S performs guided sampling employing the knowledge extracted from a source environment.

Outline


Case Study
Transfer Learning
Empirical Study
Guided Sampling
Current Research

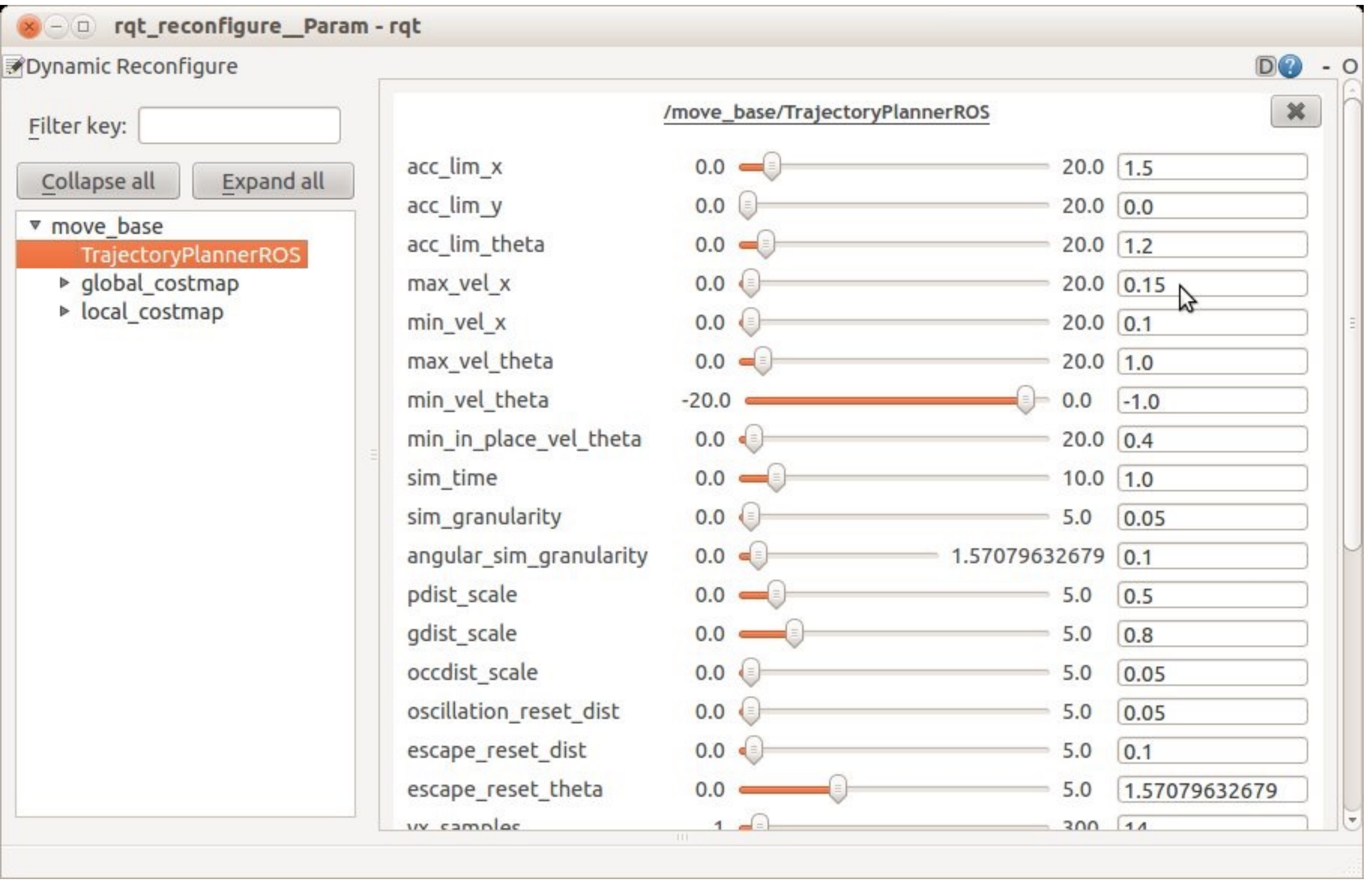

What will the software systems
of the future look like?

Software 2.0

Increasingly **customized** and **configurable**

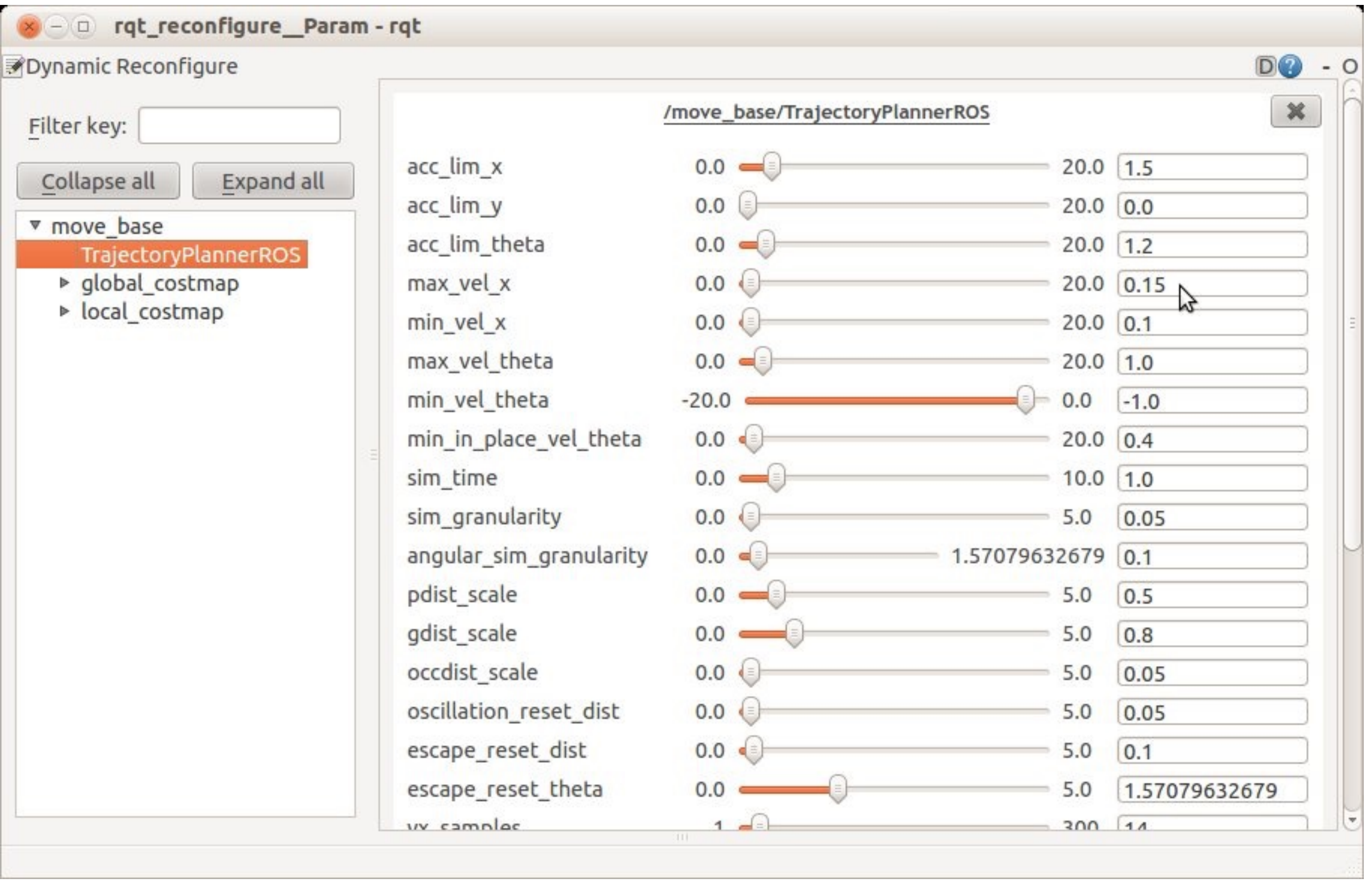
Software 2.0

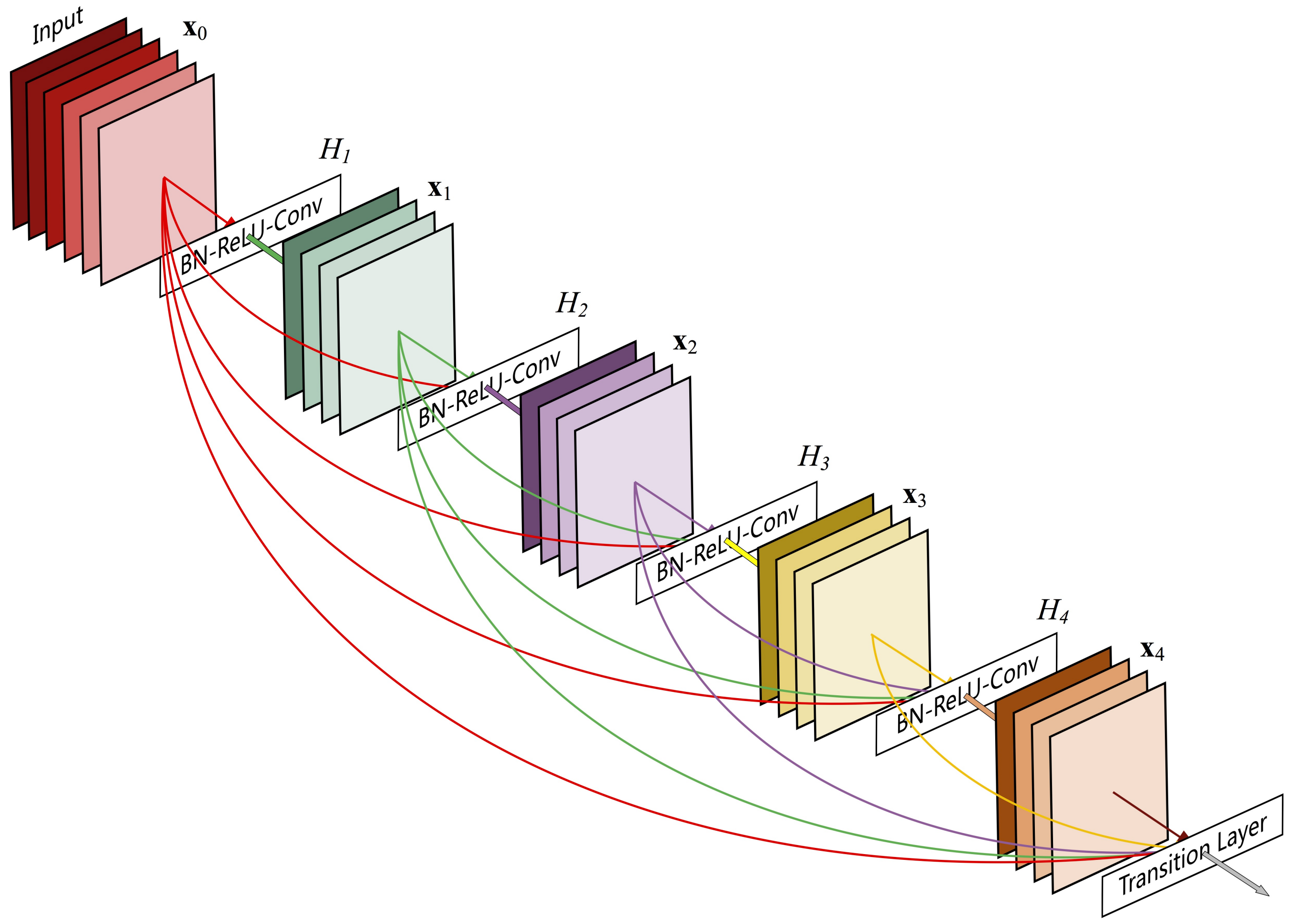
Increasingly **customized** and **configurable**



Software 2.0

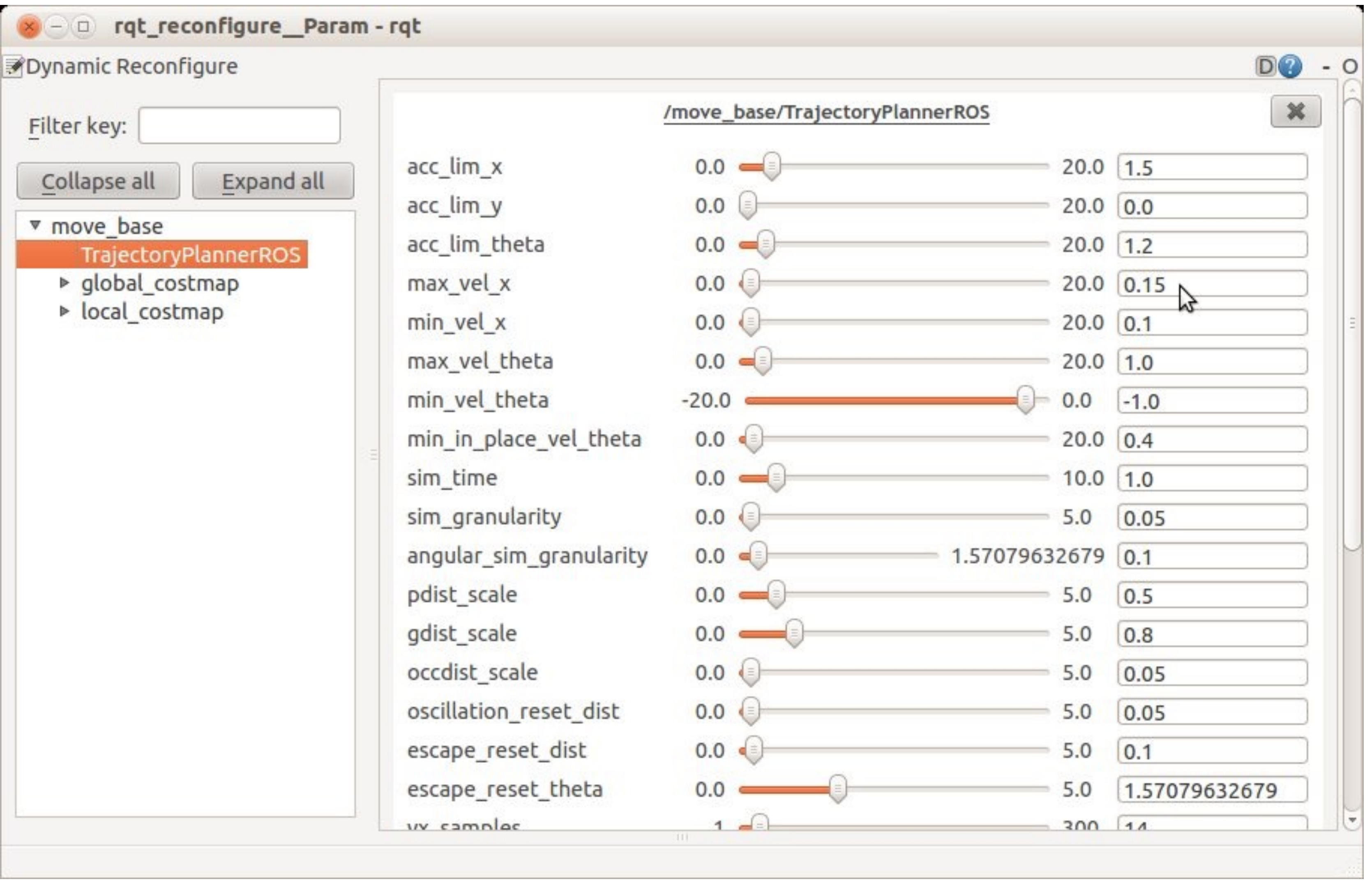
Increasingly **customized** and **configurable**

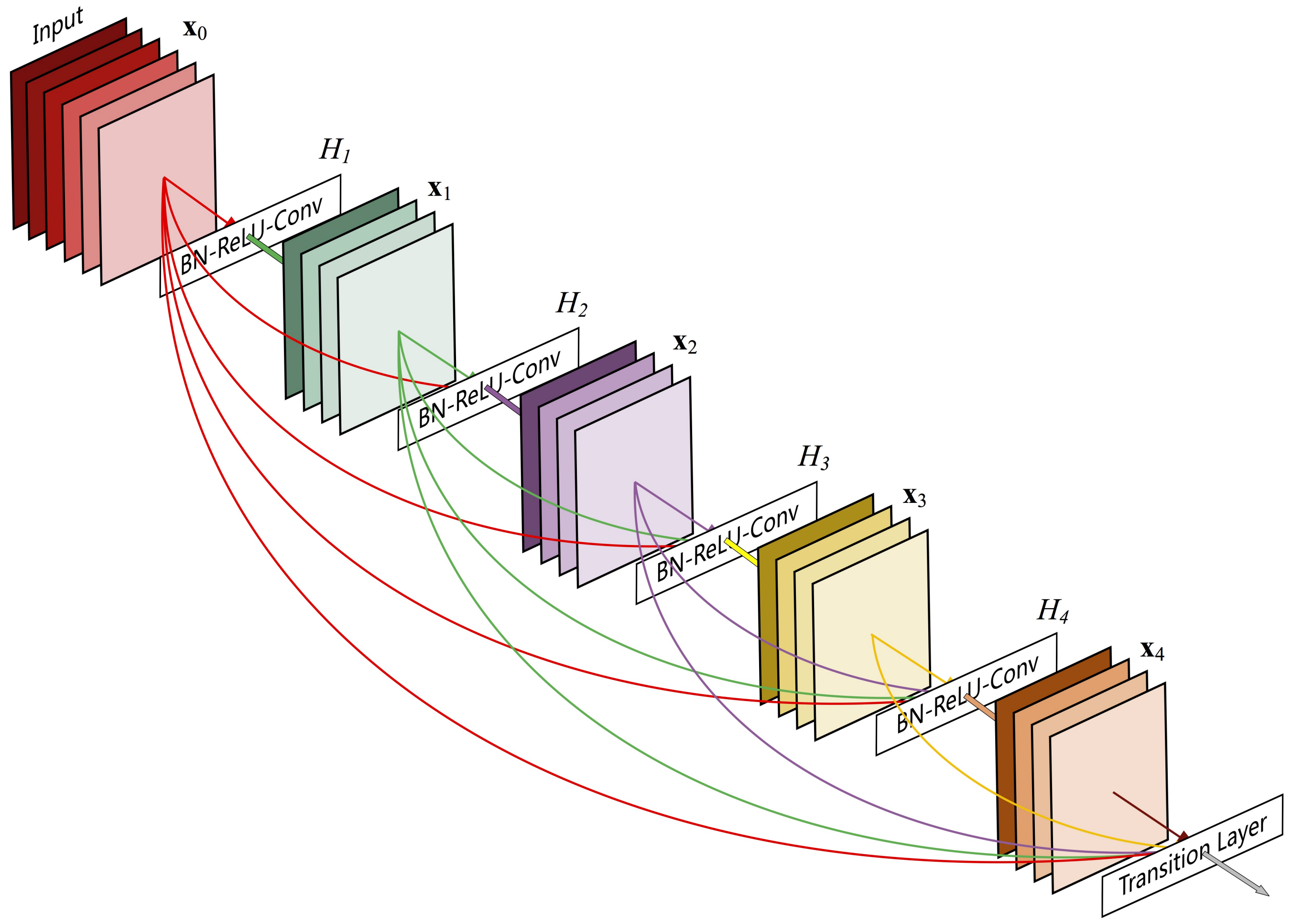


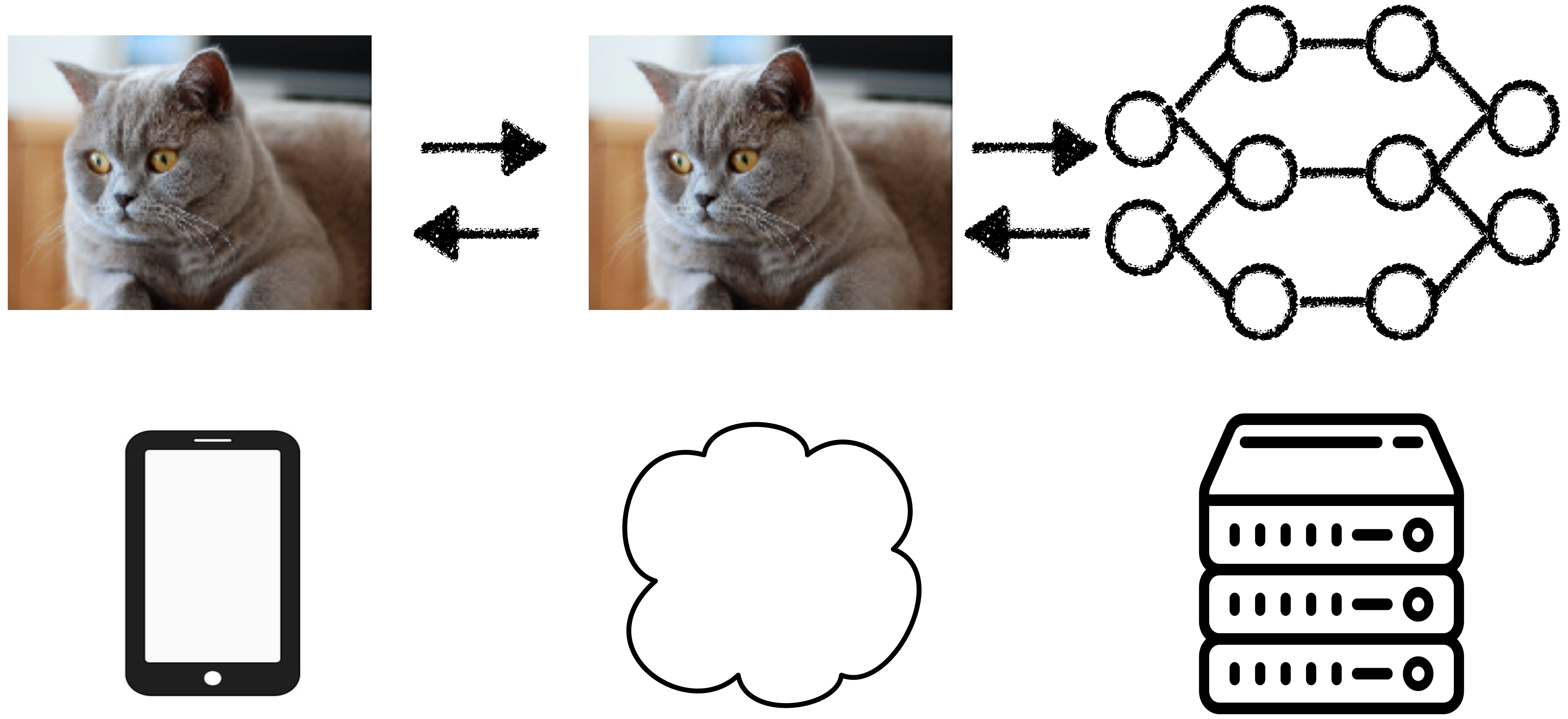


Software 2.0

Increasingly **customized** and **configurable**

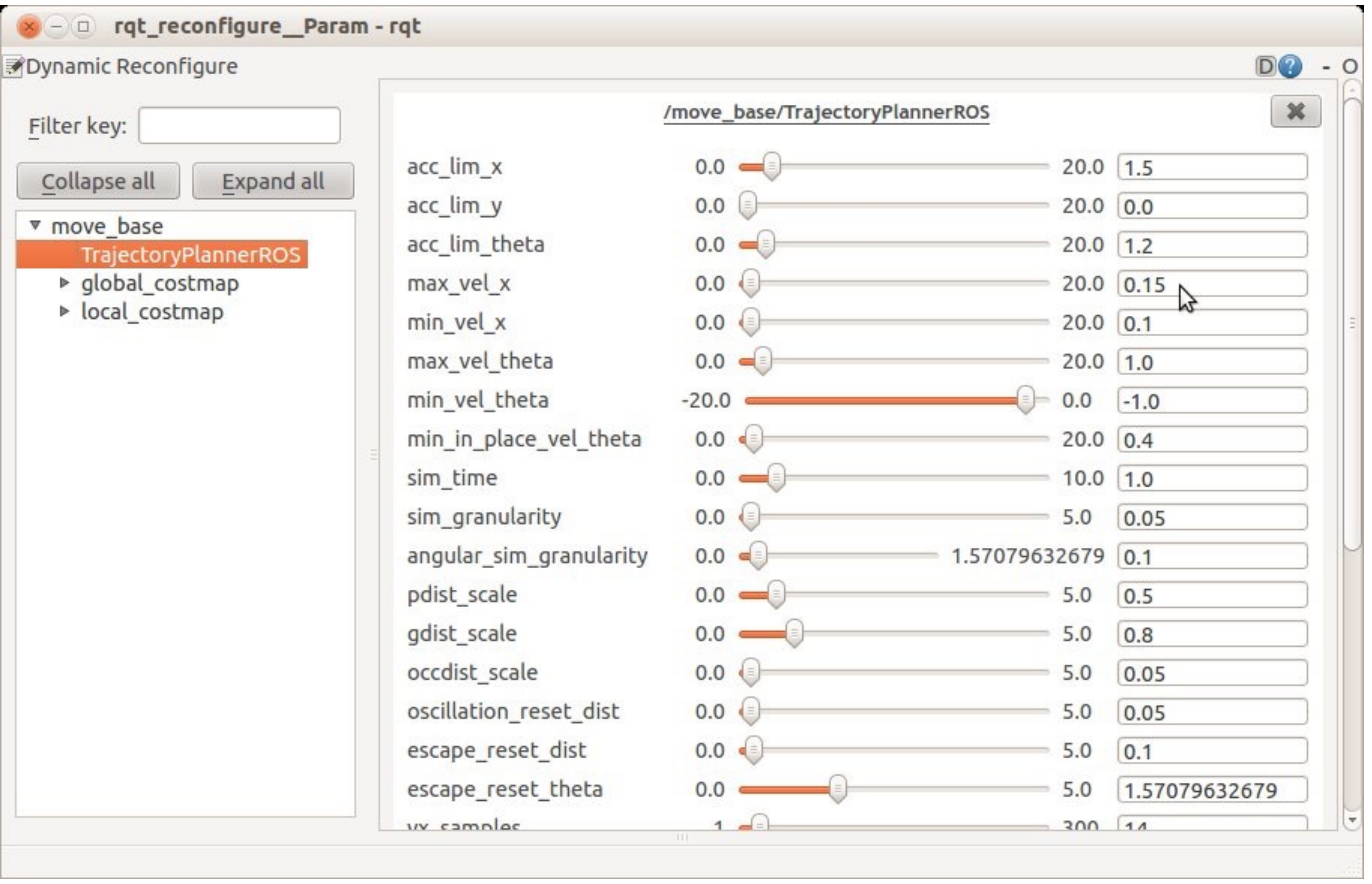


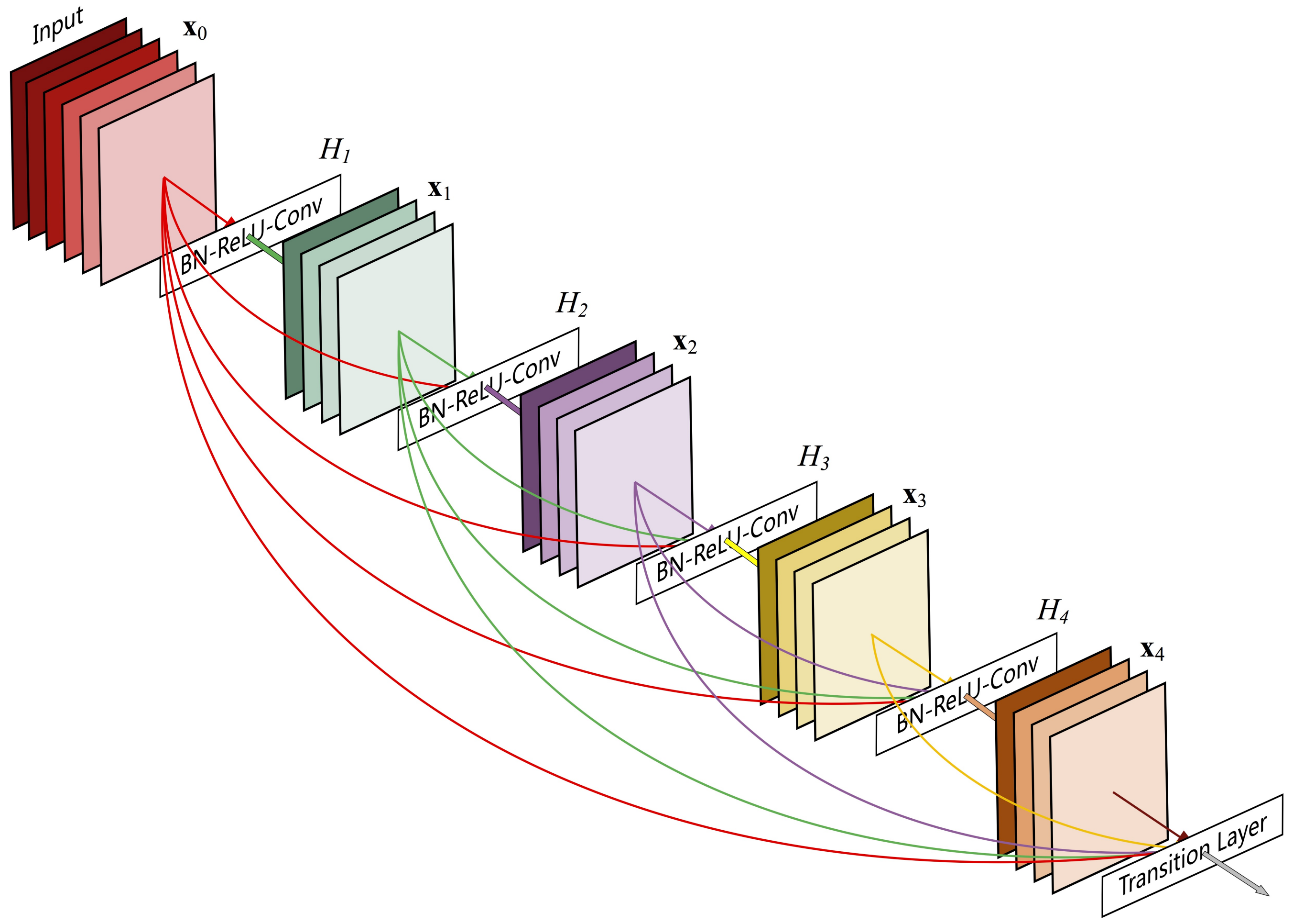


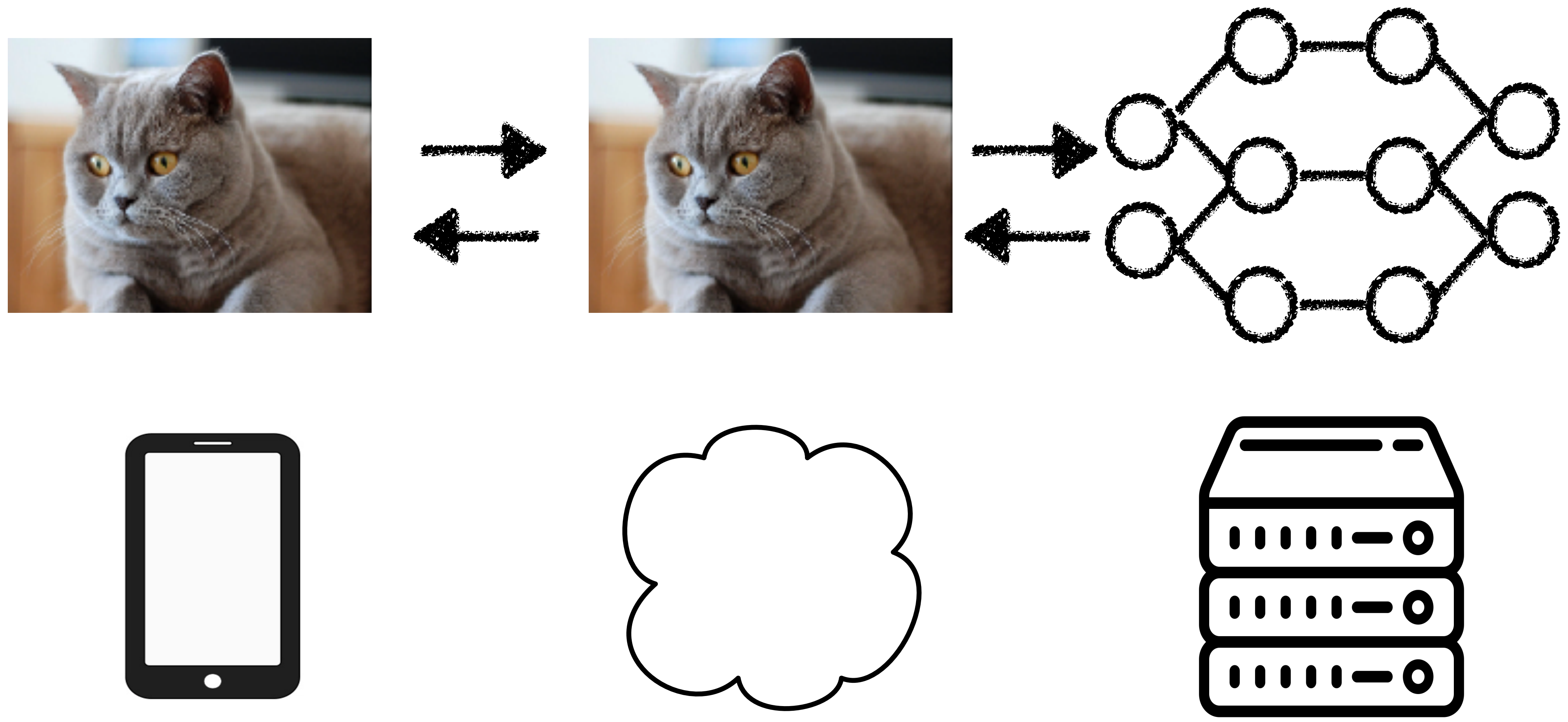


Software 2.0

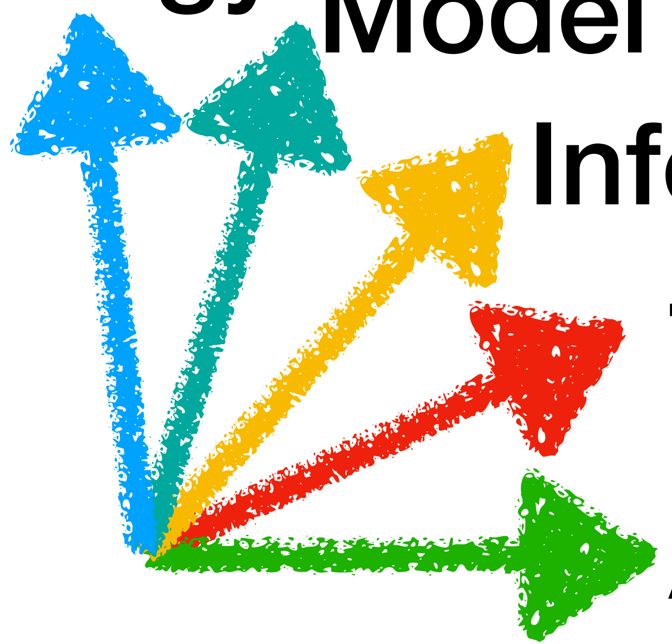
Increasingly **customized** and **configurable**







Increasingly **competing objectives**



Energy

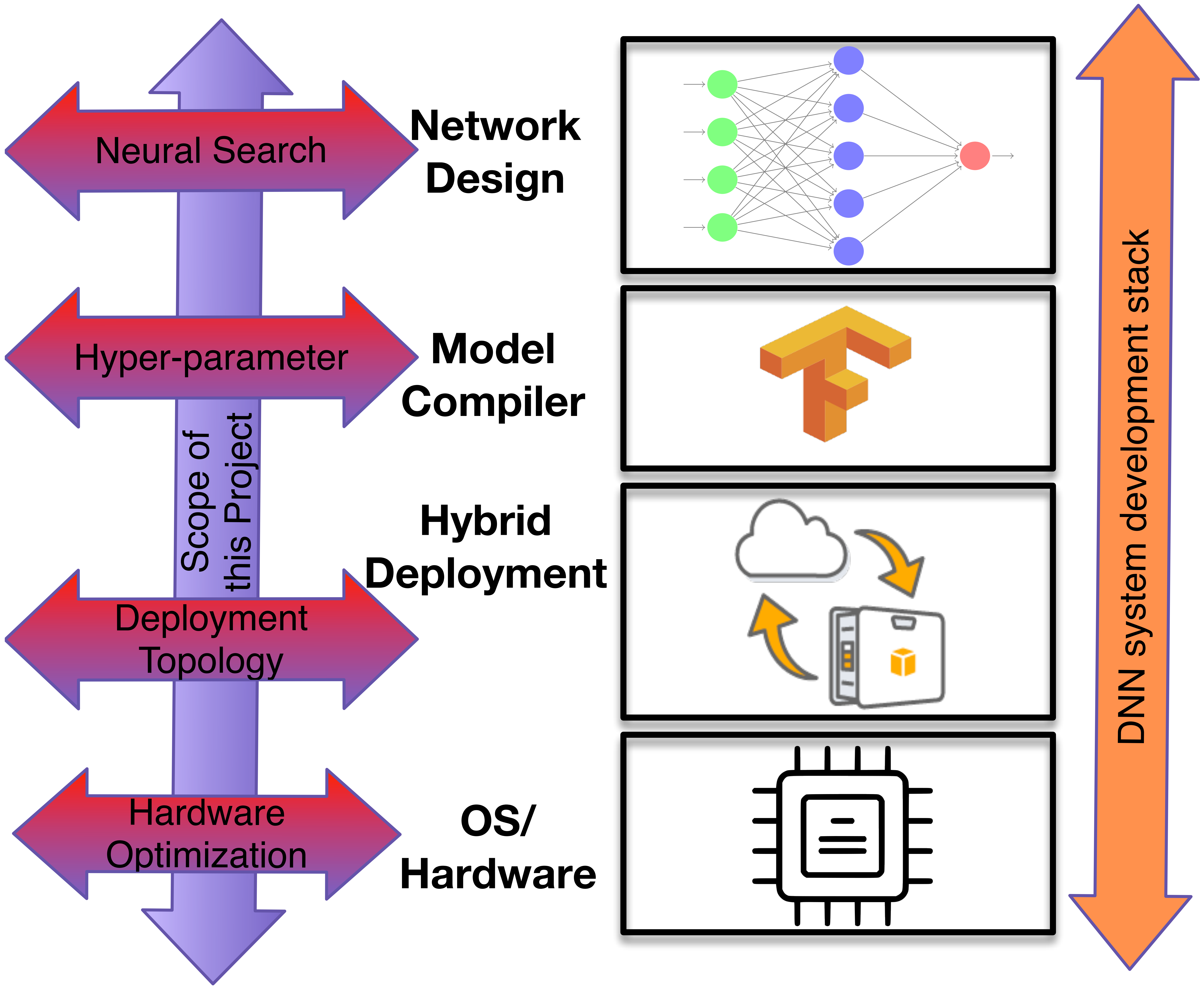
Model size

Inference speed

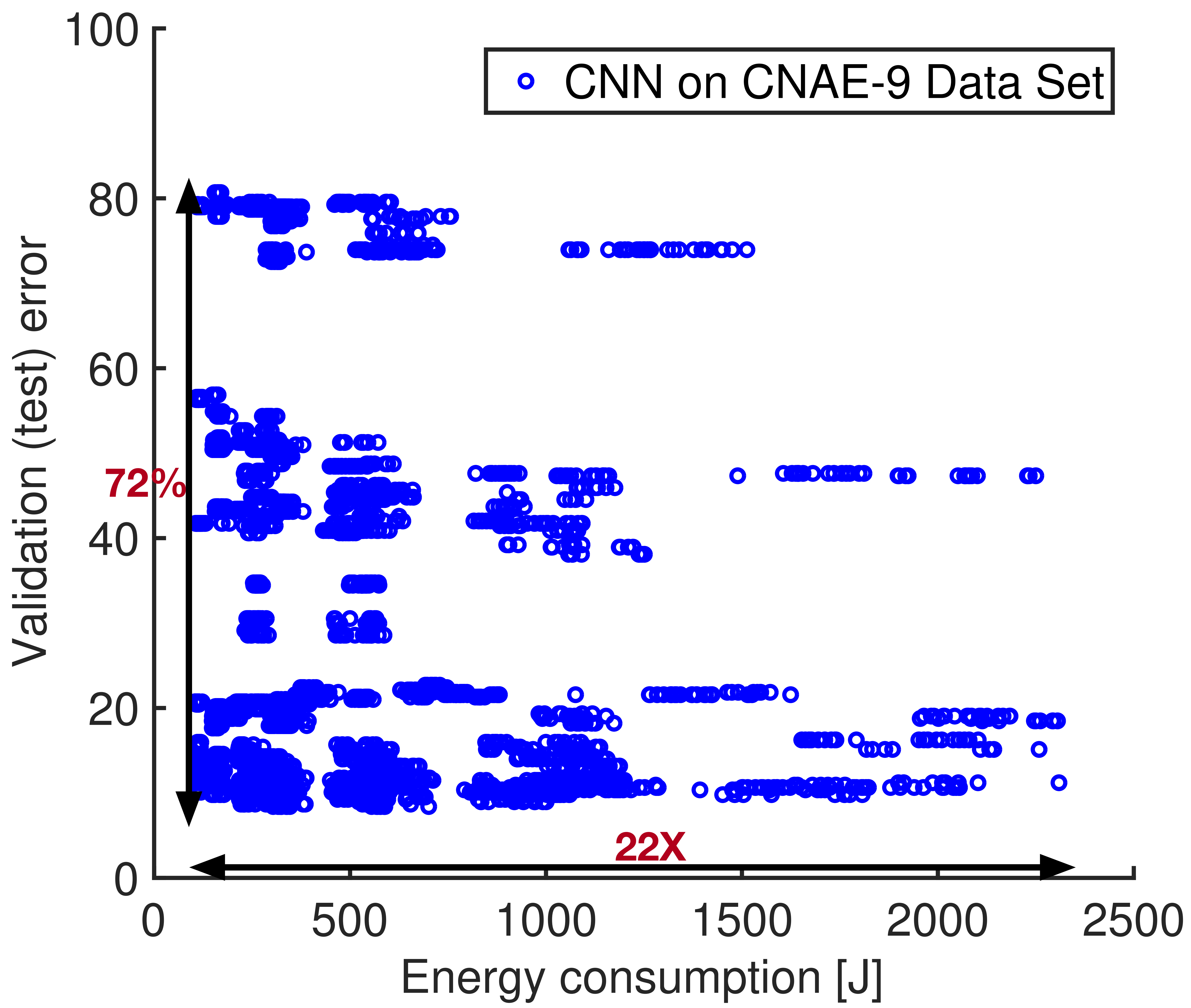
Training speed

Accuracy

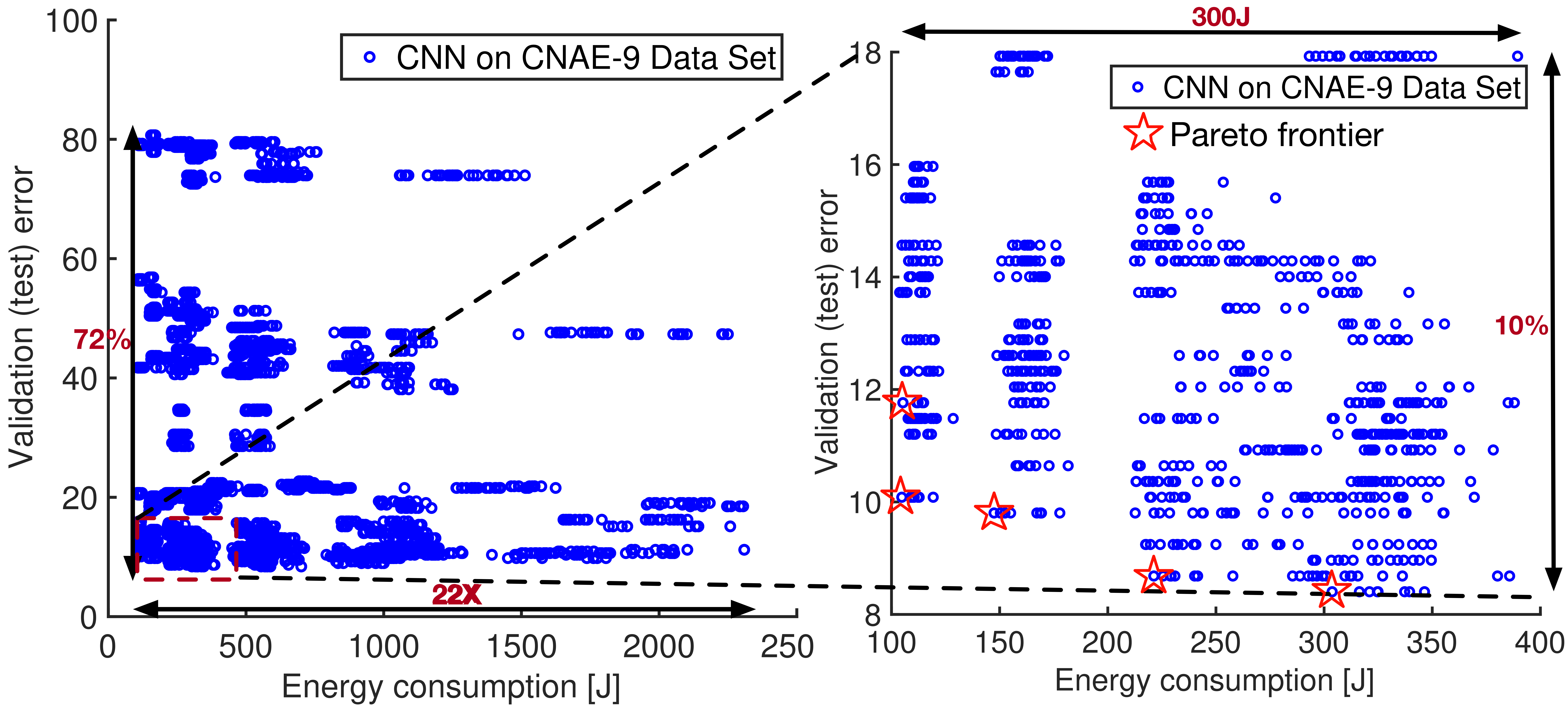
Deep neural network as a highly configurable system



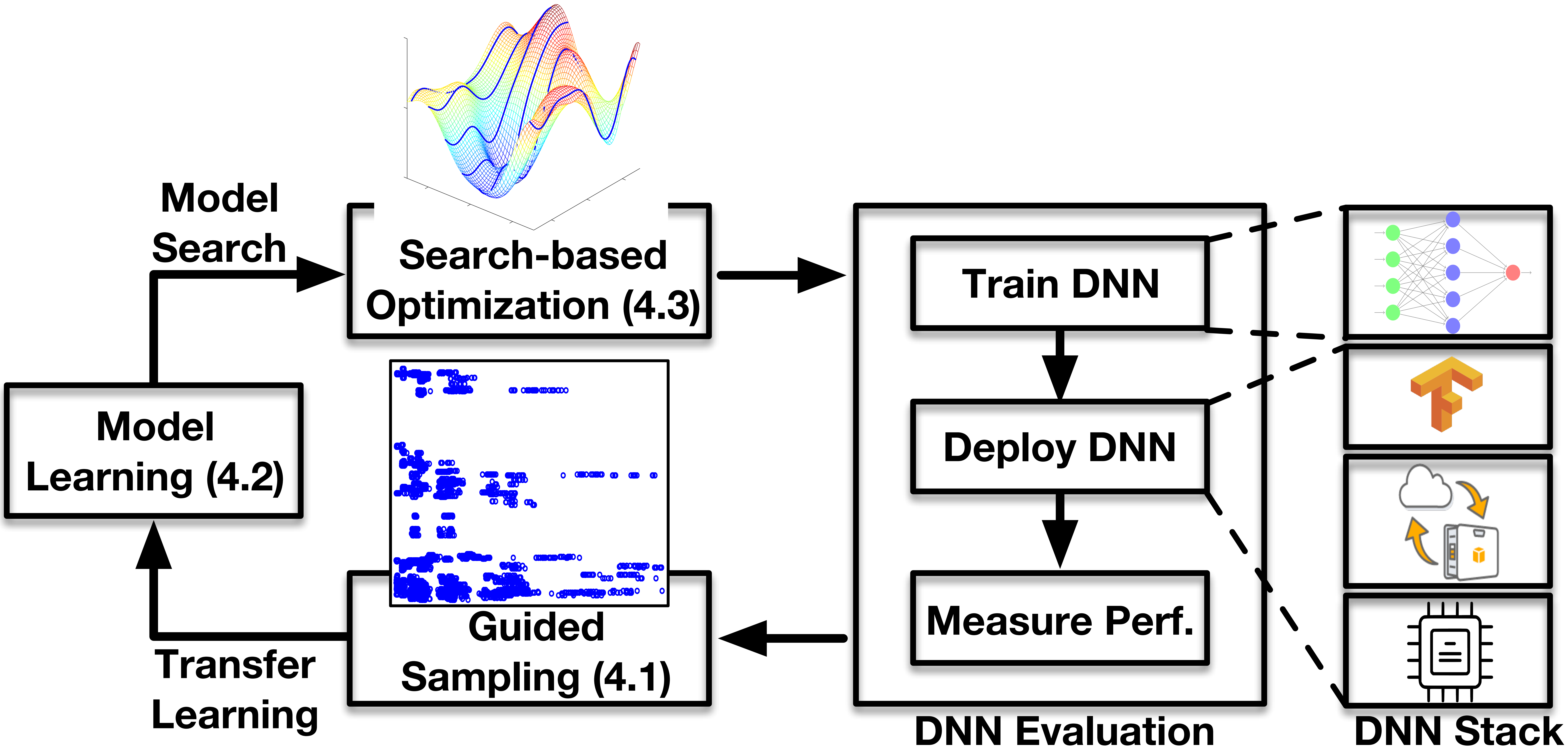
We found many configuration with the same accuracy while having drastically different energy demand



We found many configuration with the same accuracy while having drastically different energy demand



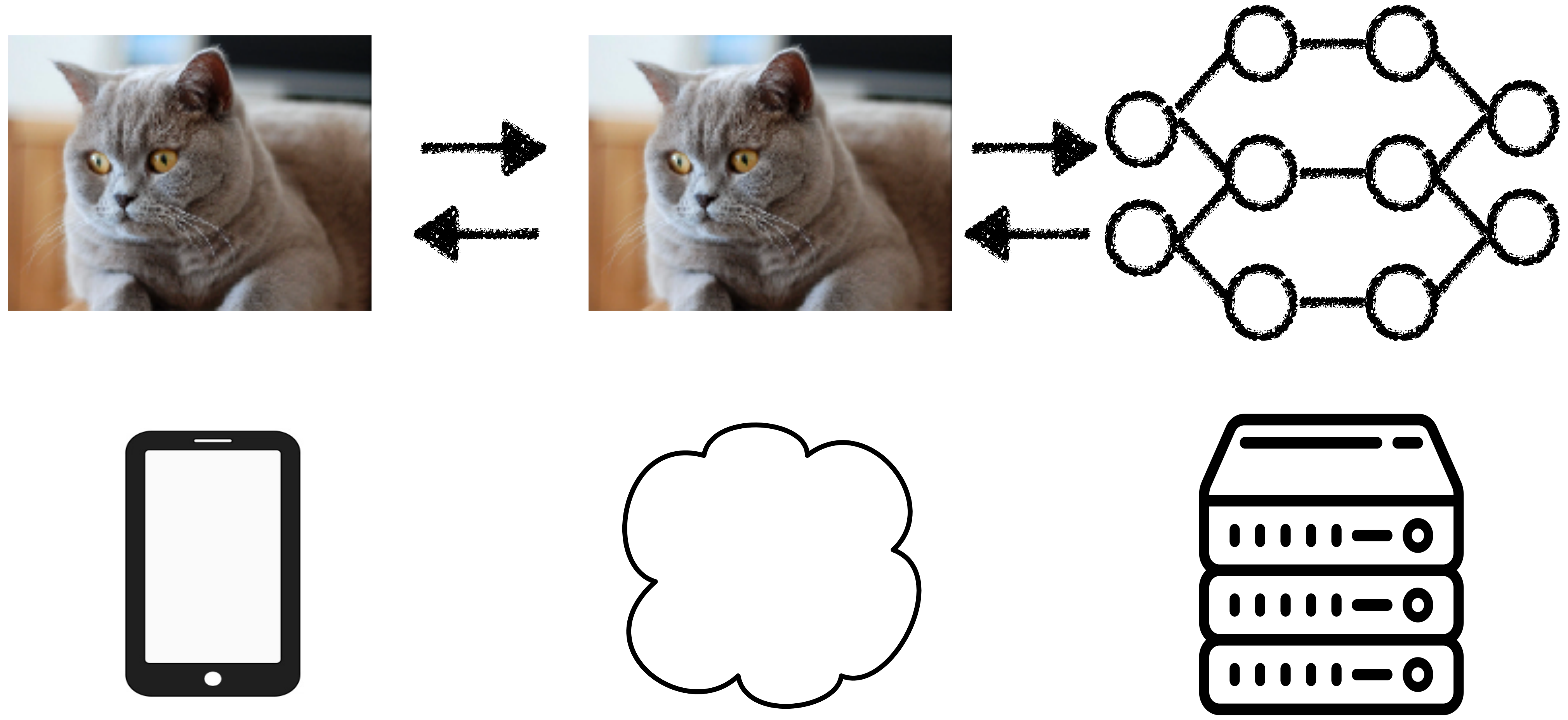
Optimizing Energy Consumption of Deep Neural Networks



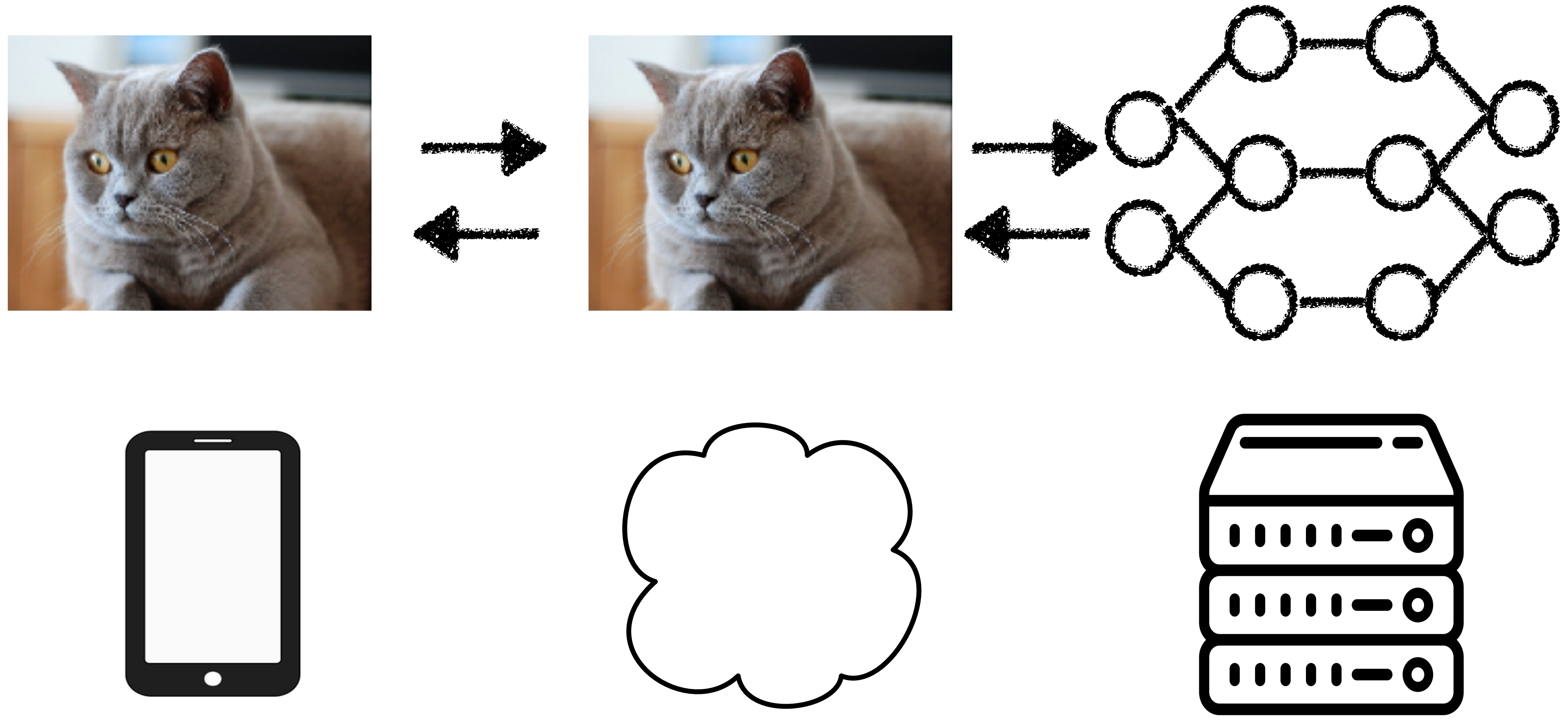
Deep architecture design level variations

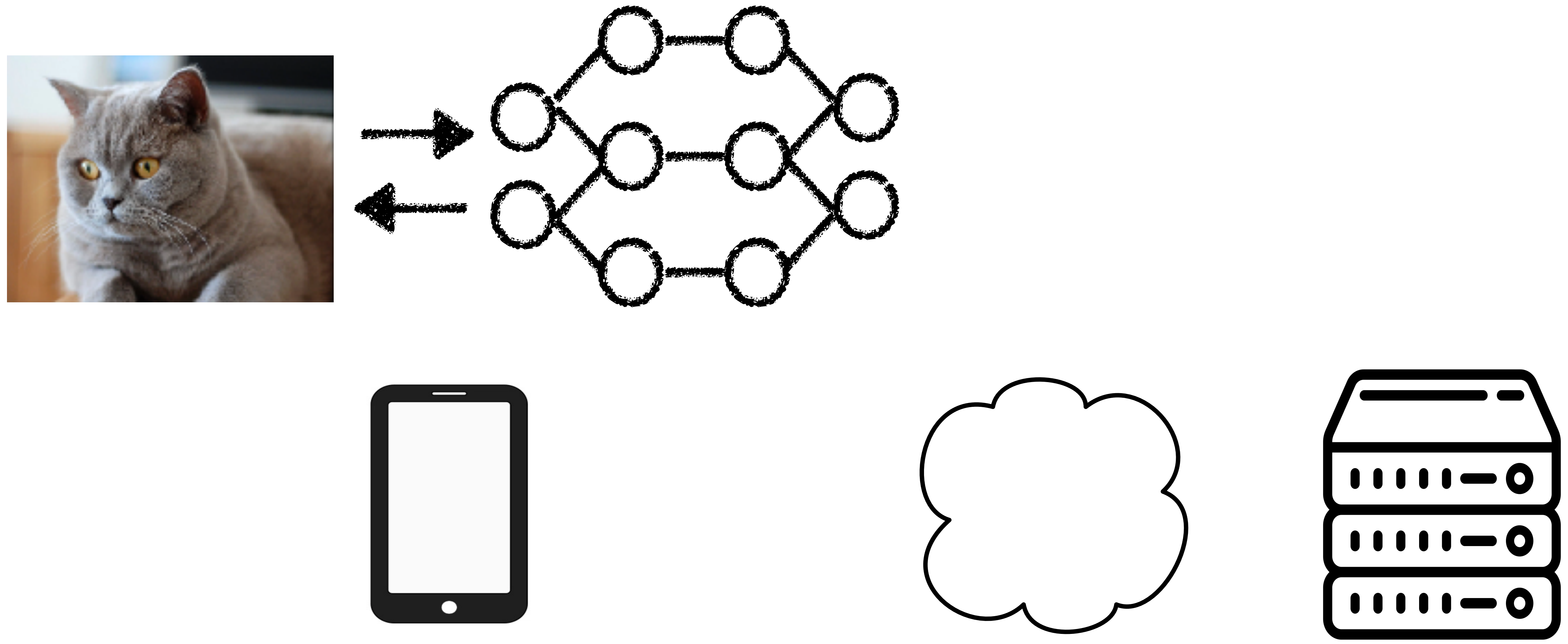
NAME	DESCRIPTION	TYPE	MIN	MAX
batch_size	SGD parameter	int	8	32
conv_1_filter_size	Architecture parameter	int	2	10
conv_1_num_filters	Architecture parameter	int	32	256
conv_2_filter_size	Architecture parameter	int	2	10
conv_2_num_filters	Architecture parameter	int	32	256
conv_3_filter_size	Architecture parameter	int	2	10
conv_3_num_filters	Architecture parameter	int	32	256
log_beta_1	Adam SGD parameter	real	-4.6	-0.7
log_beta_2	Adam SGD parameter	real	-13.8	-0.7
log_decay	Adam SGD parameter	real	-23	-2.3
log_epsilon	Adam SGD parameter	real	-23	-13.8
log_lr	Adam SGD parameter	real	-23	0

Deep architecture deployment variations

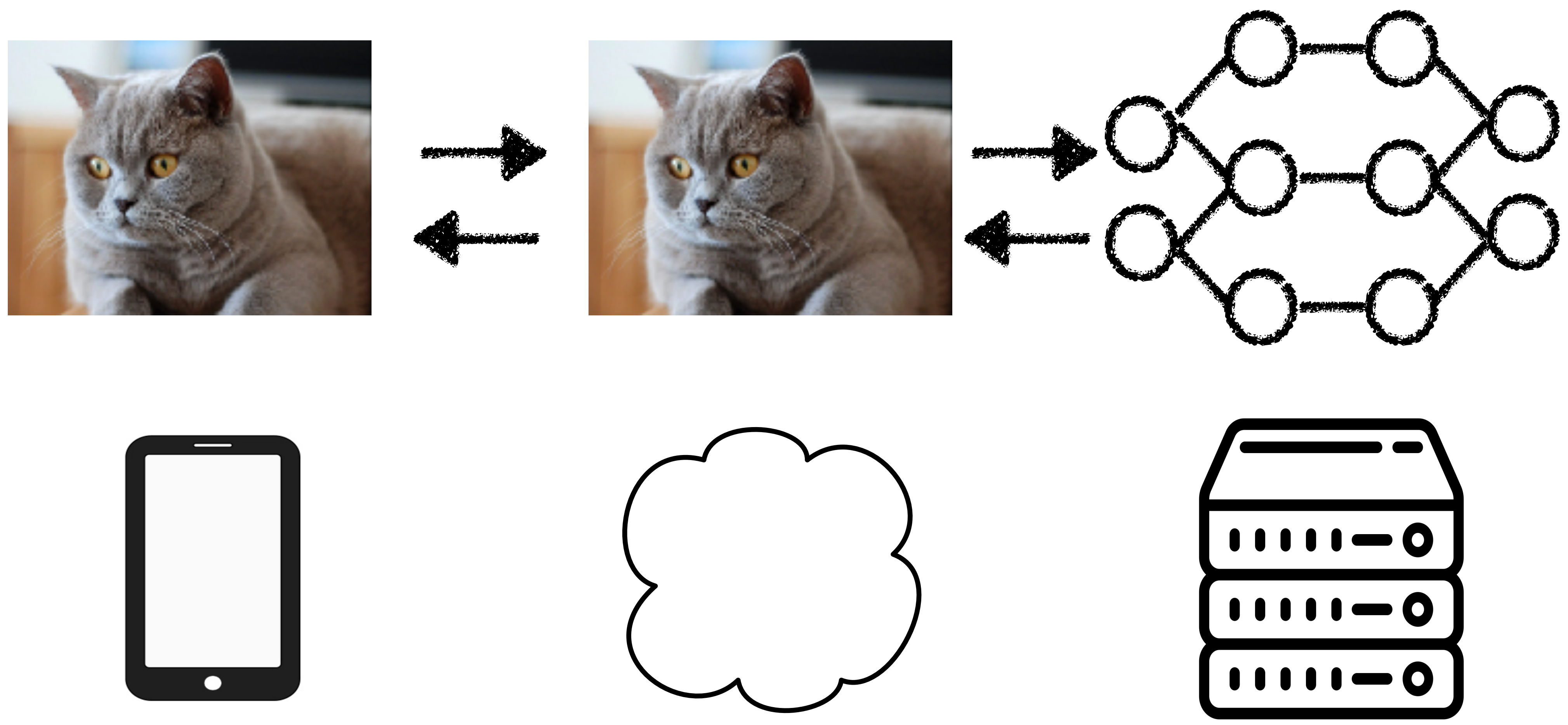


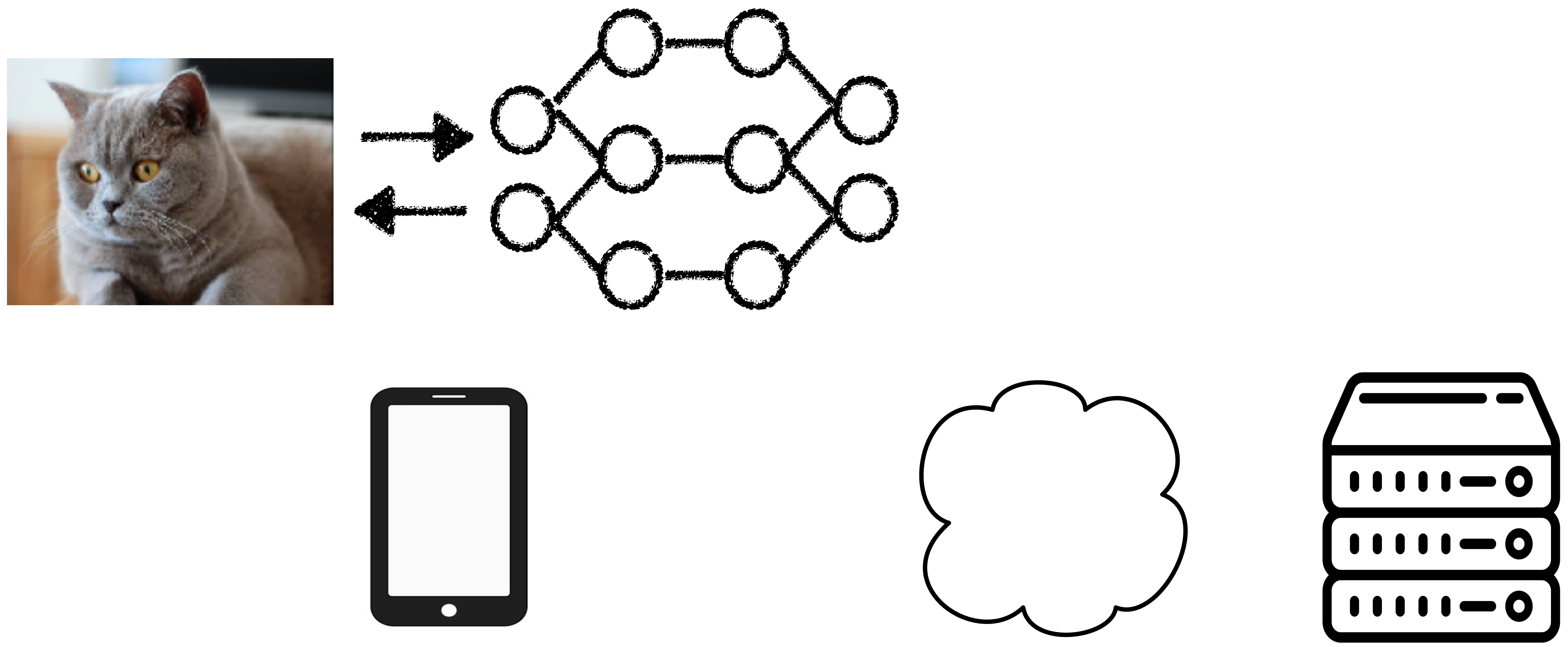
Deep architecture deployment variations

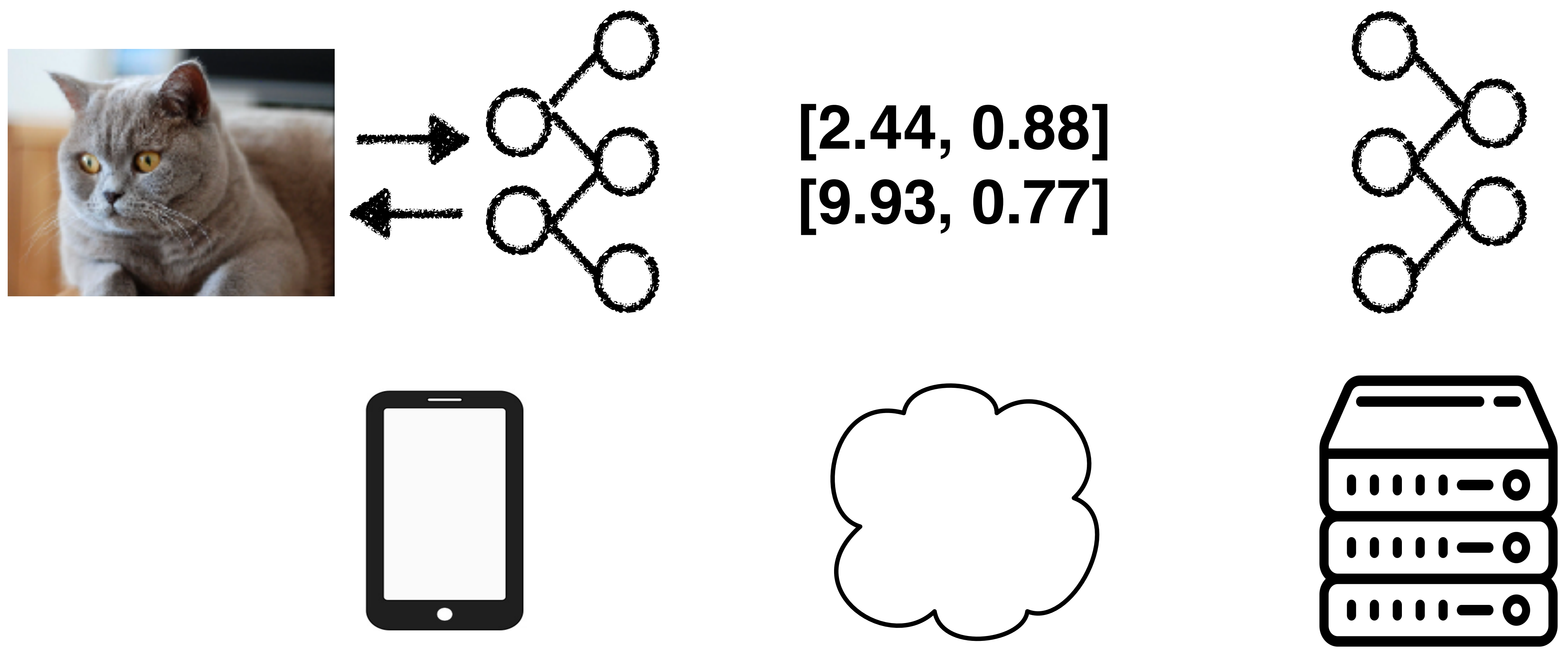




Deep architecture deployment variations



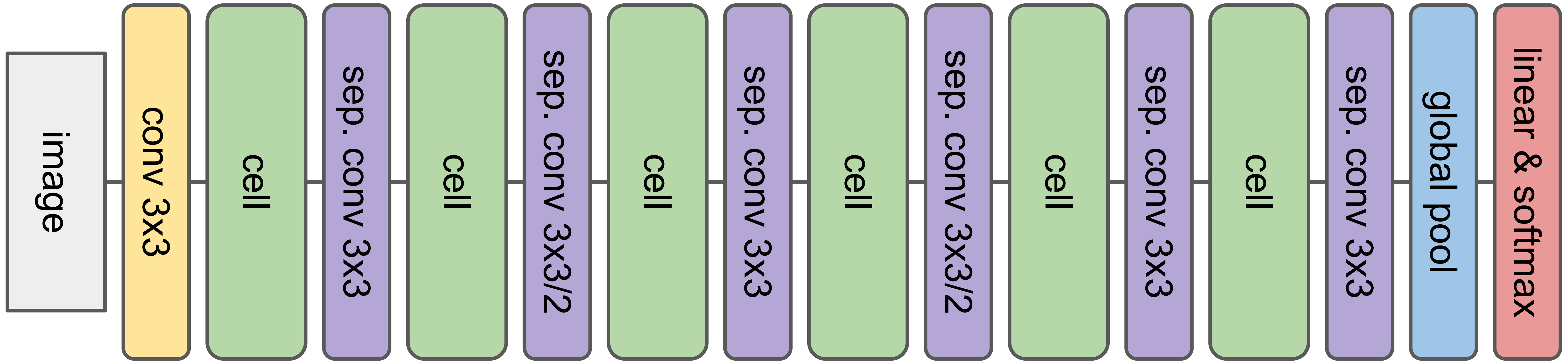




Deep architecture hardware-level variations

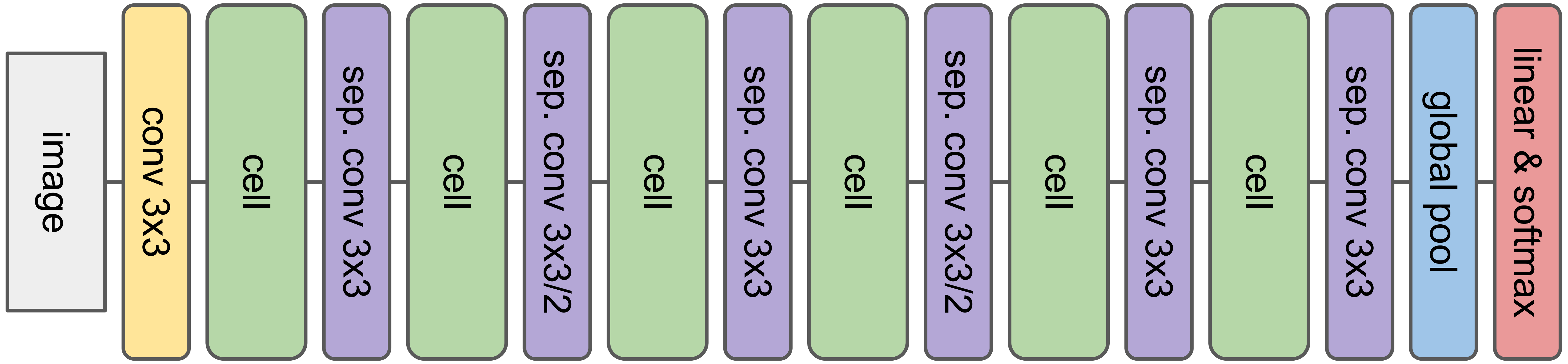
NVIDIA Xavier

Exploring the design space of deep networks



Optimal Architecture
(Yesterday)

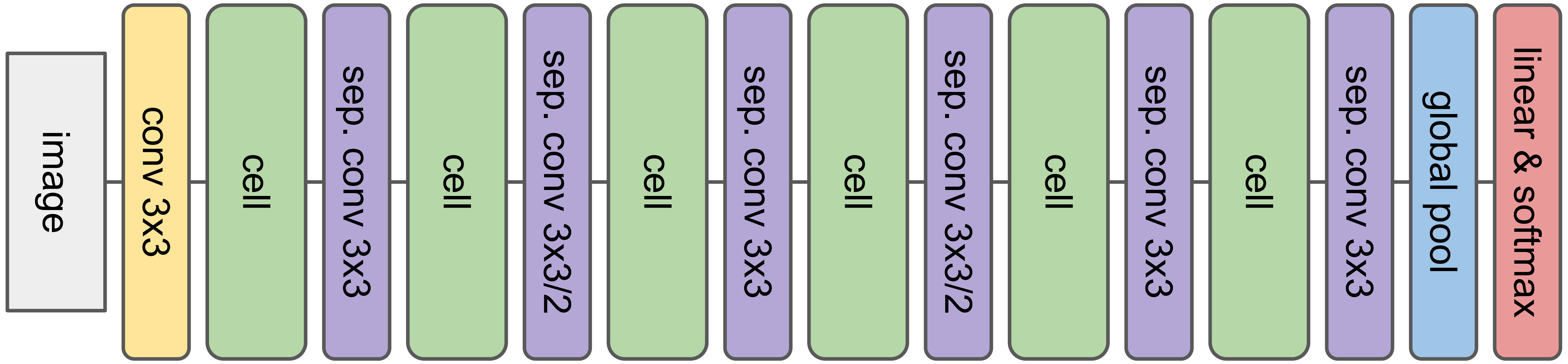
Exploring the design space of deep networks



Optimal Architecture (Yesterday)

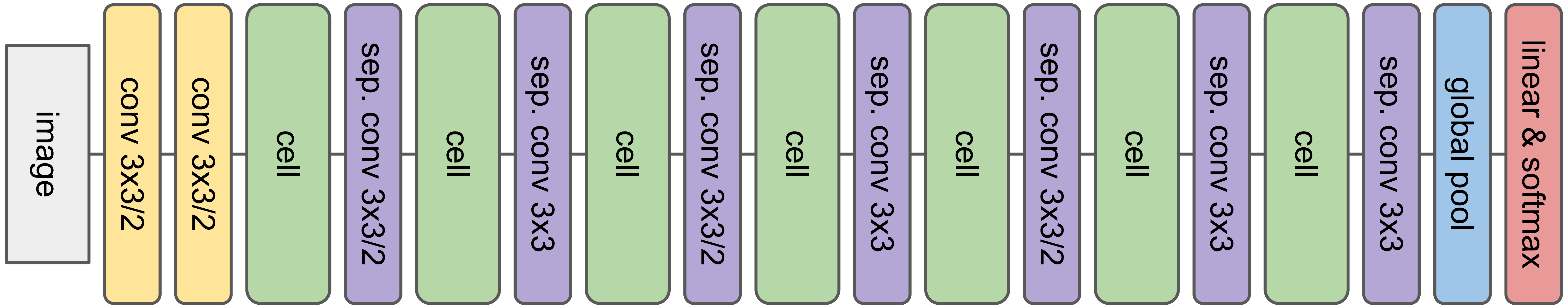
New Fraud Pattern

Exploring the design space of deep networks



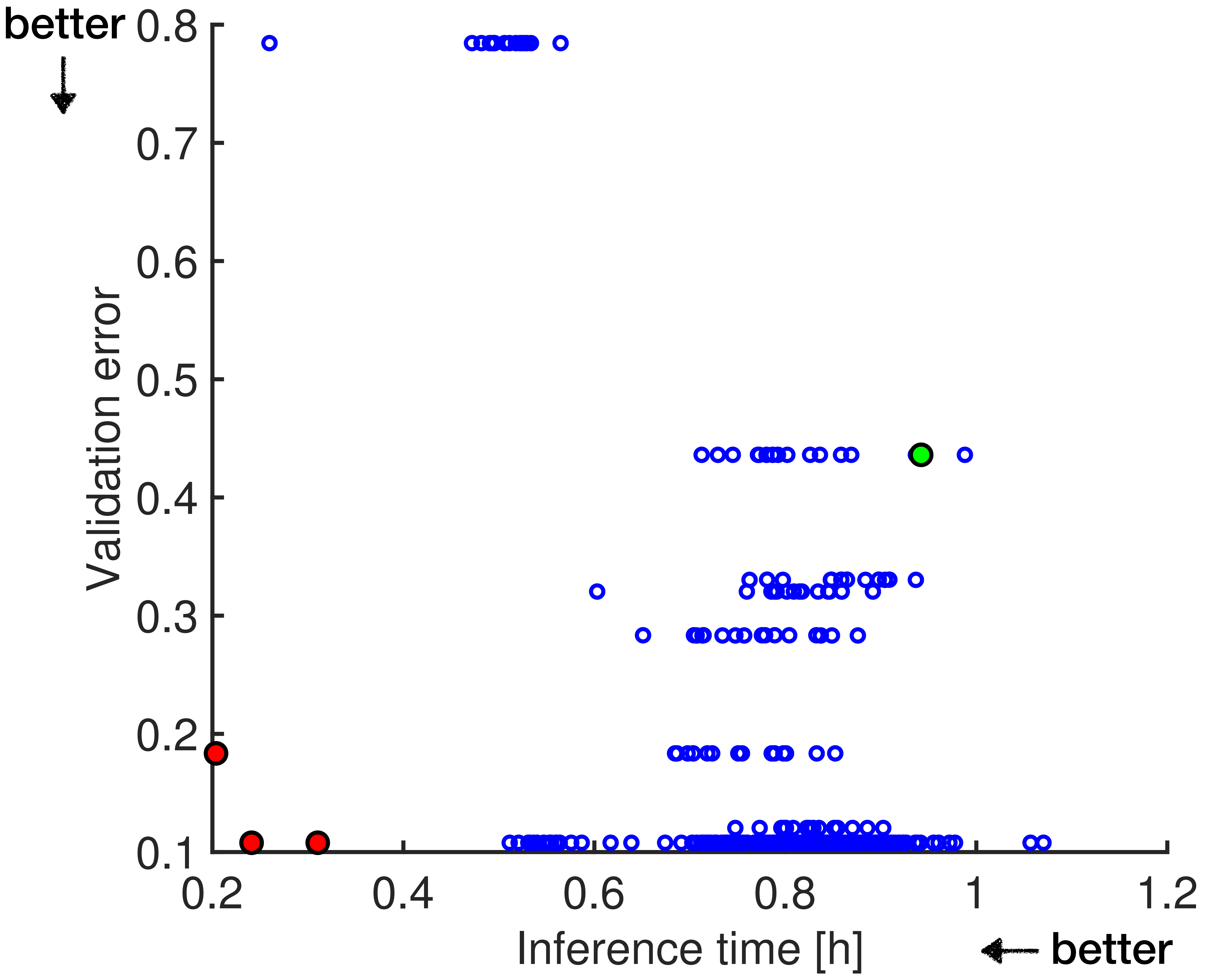
Optimal Architecture (Yesterday)

New Fraud Pattern

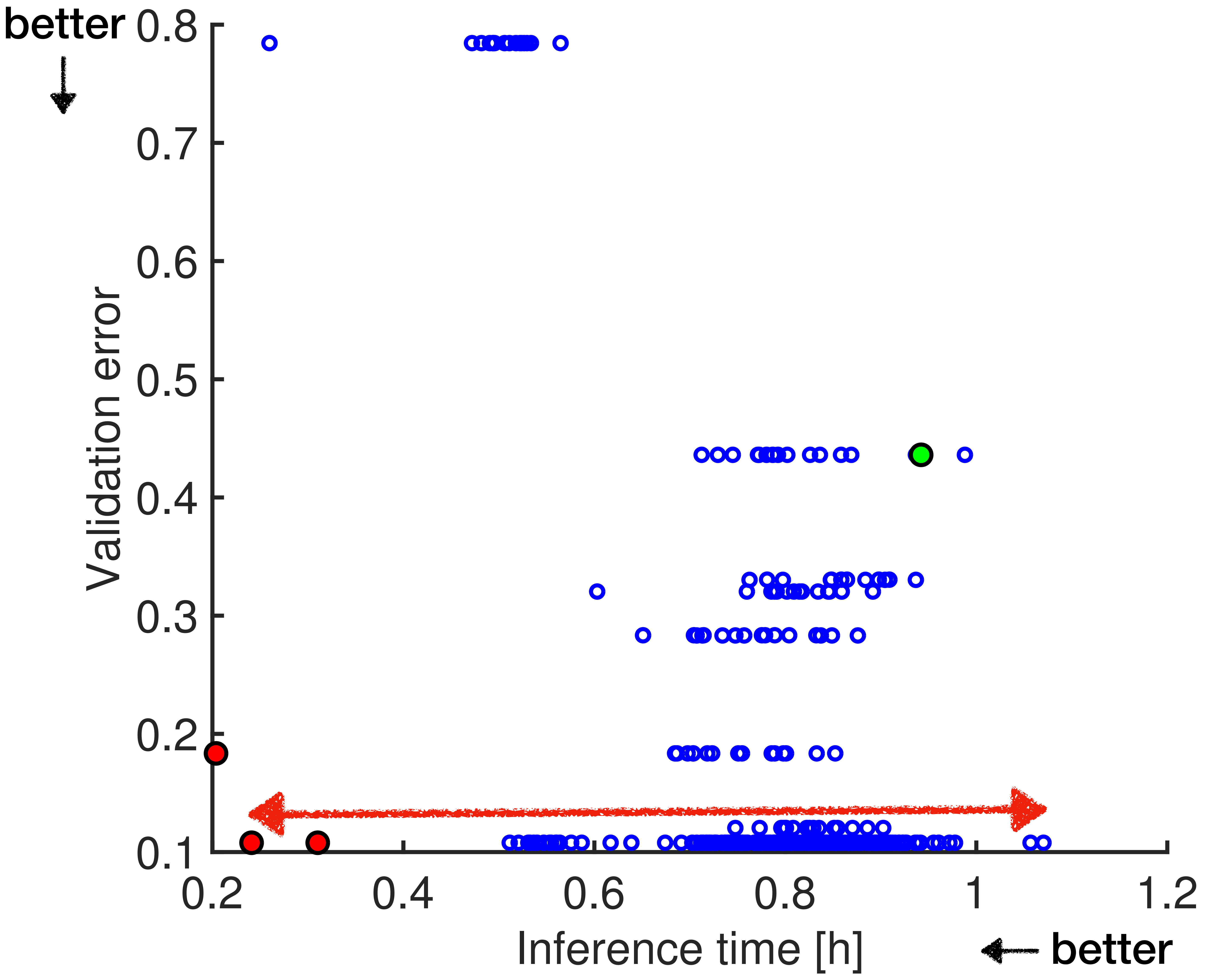


Optimal Architecture (Today)

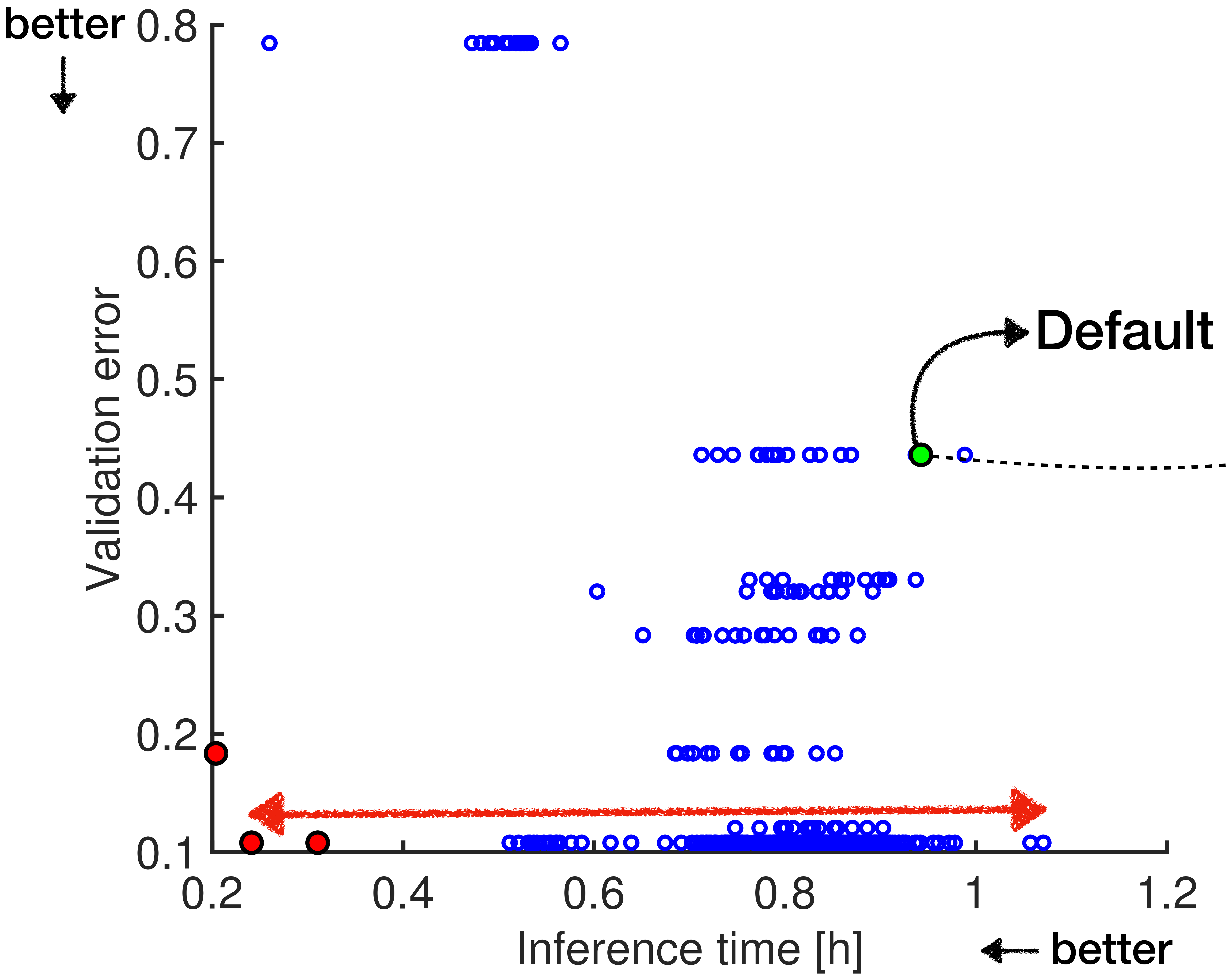
Exploring the design space of deep networks

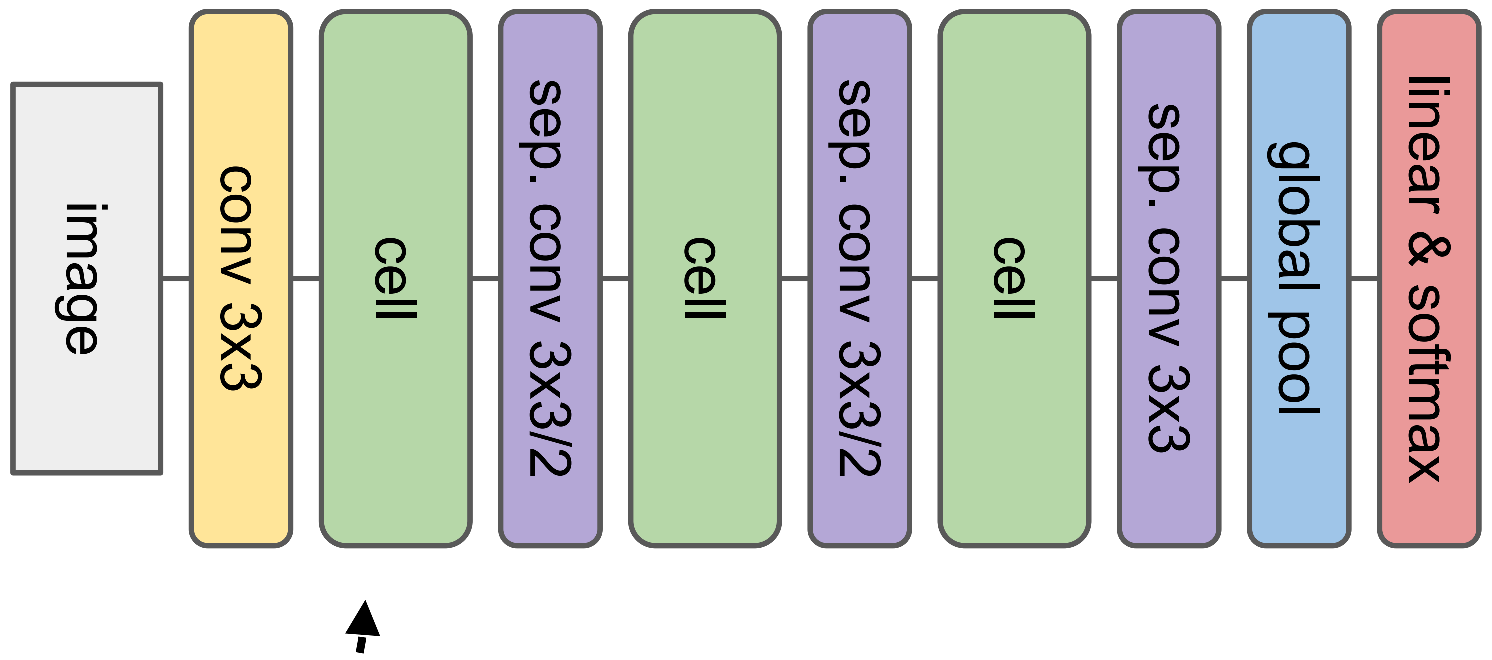


Exploring the design space of deep networks

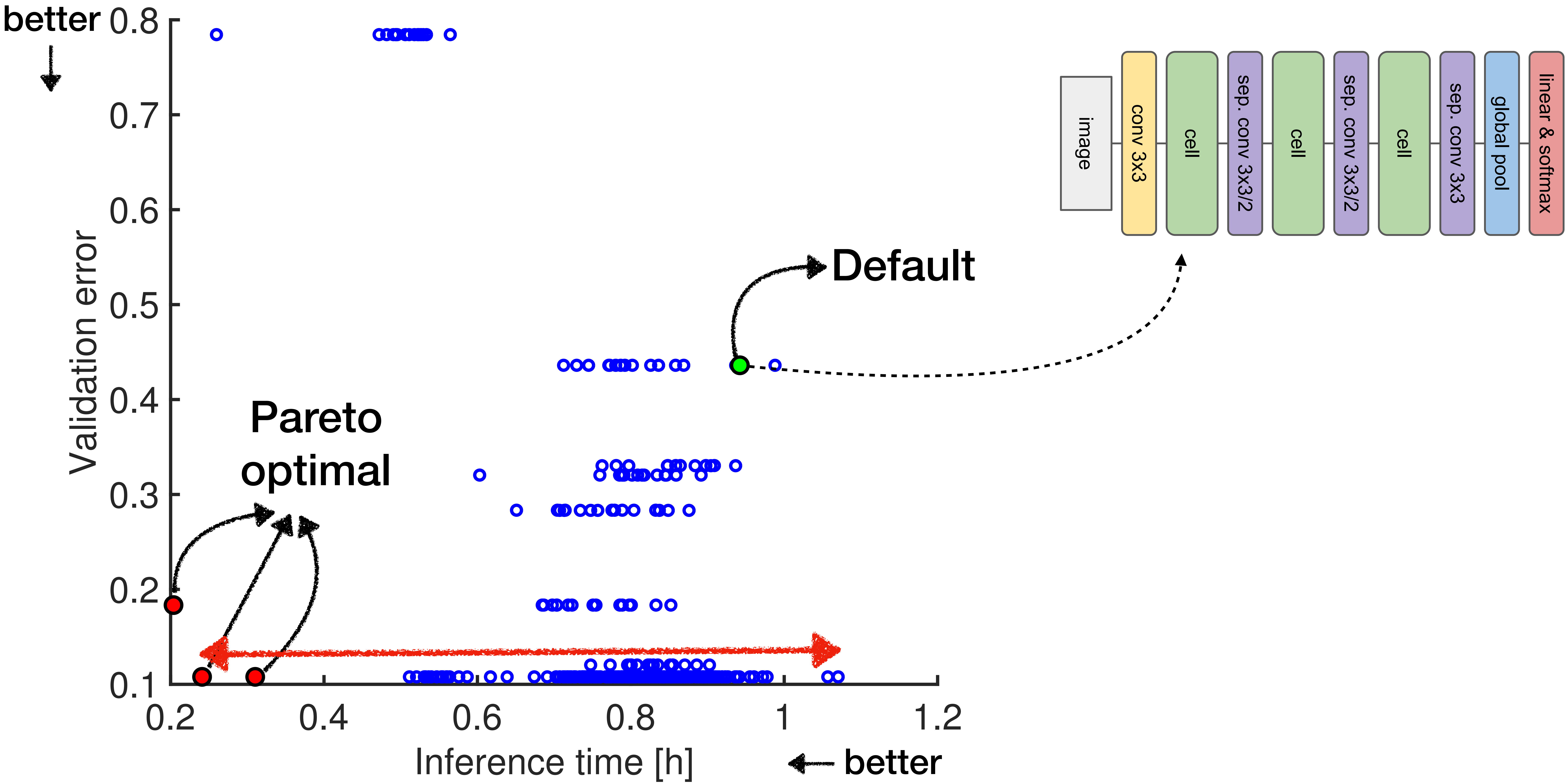


Exploring the design space of deep networks





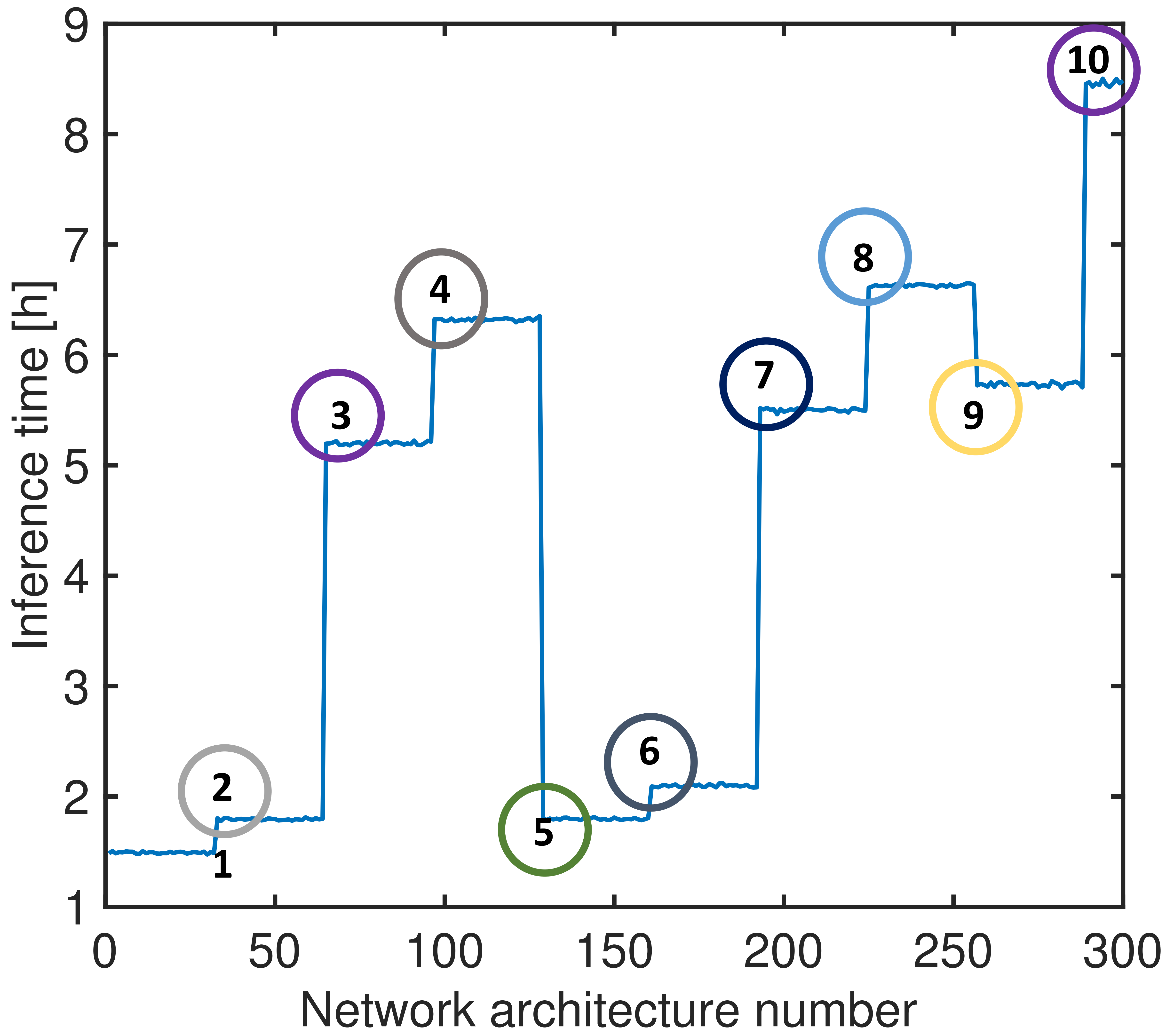
Exploring the design space of deep networks



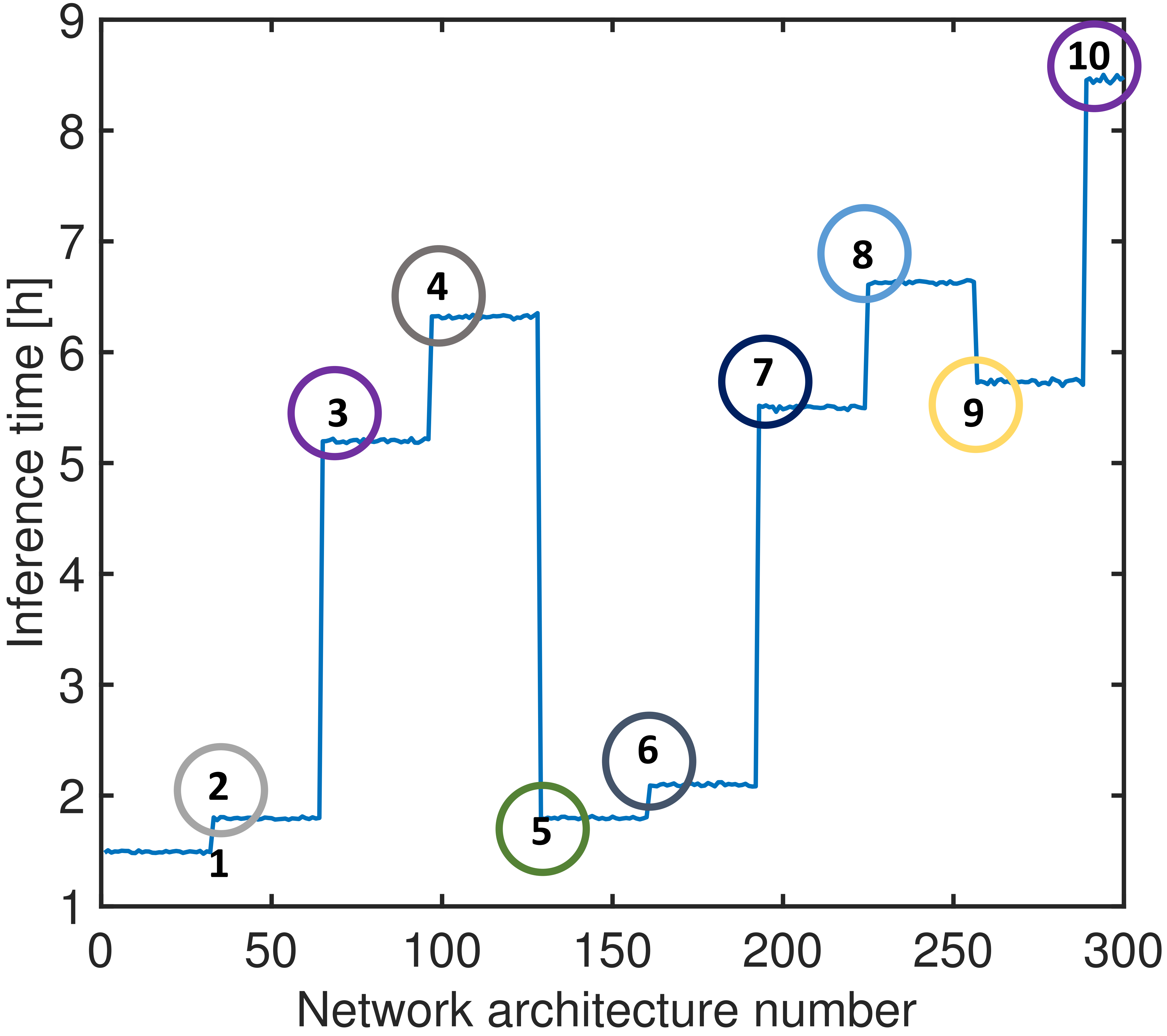
Exploring the design space of deep networks

Exploring the design space of deep networks

Configuration options and interactions influence performance of DNNs

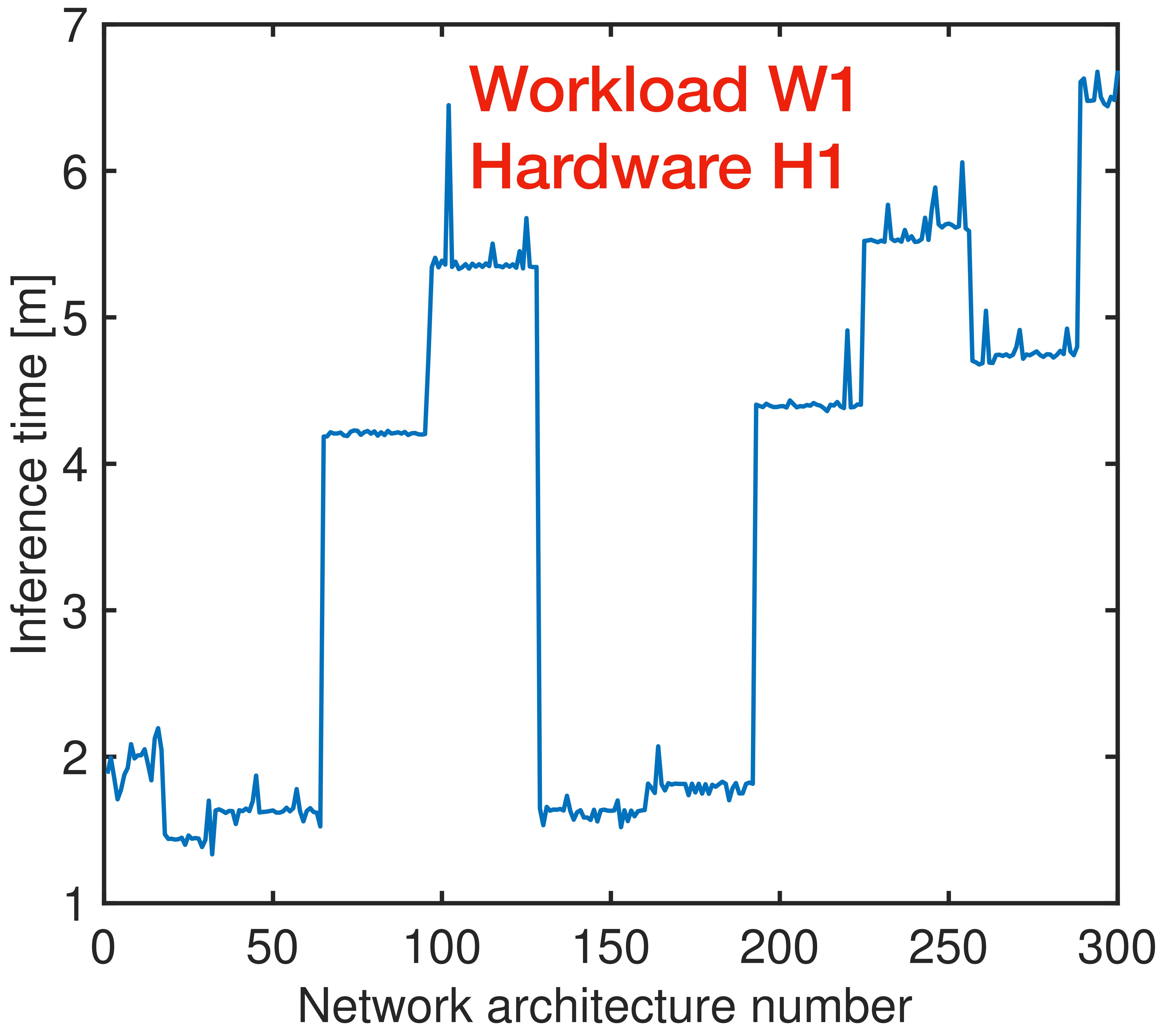


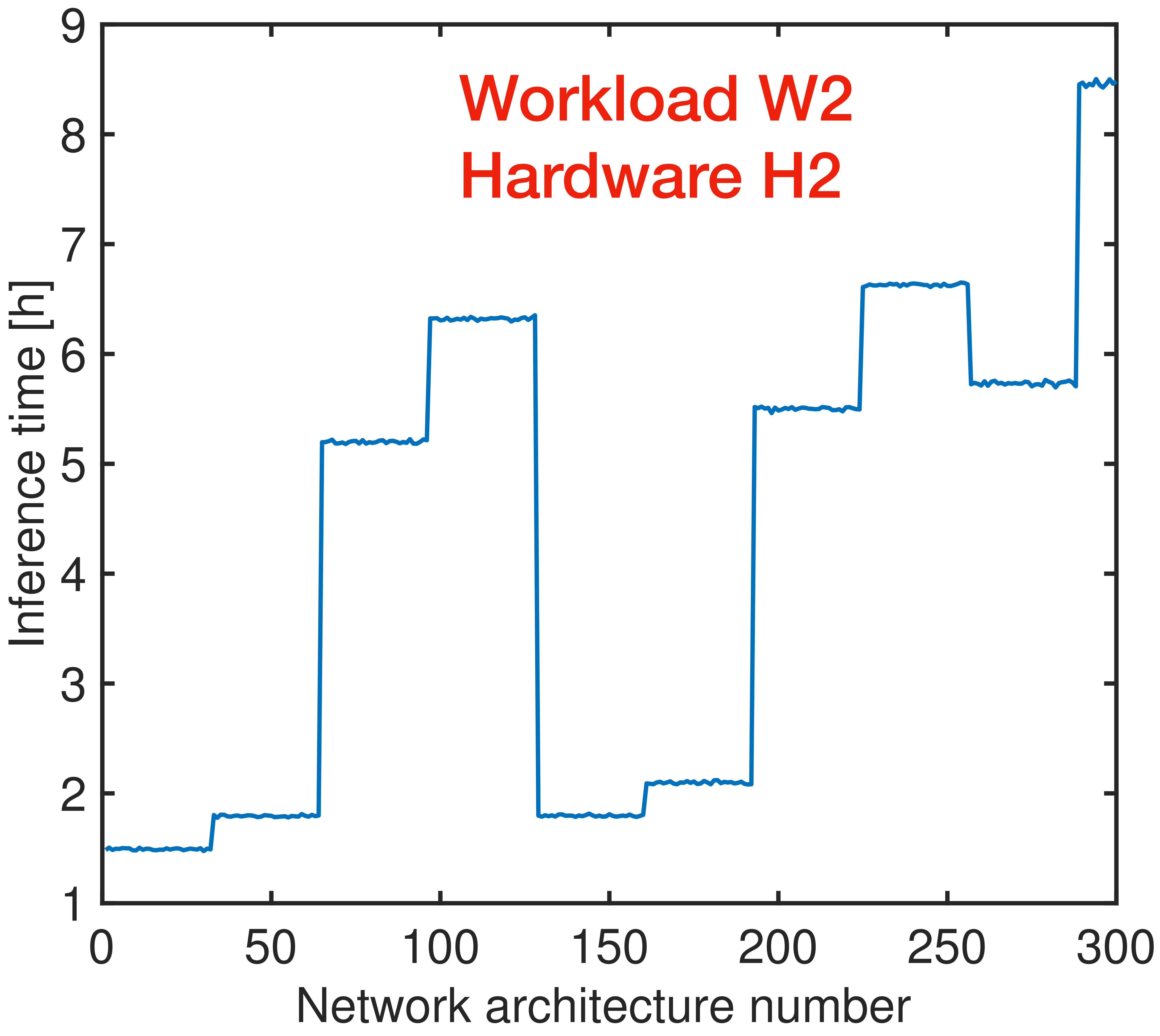
Configuration options and interactions influence performance of DNNs



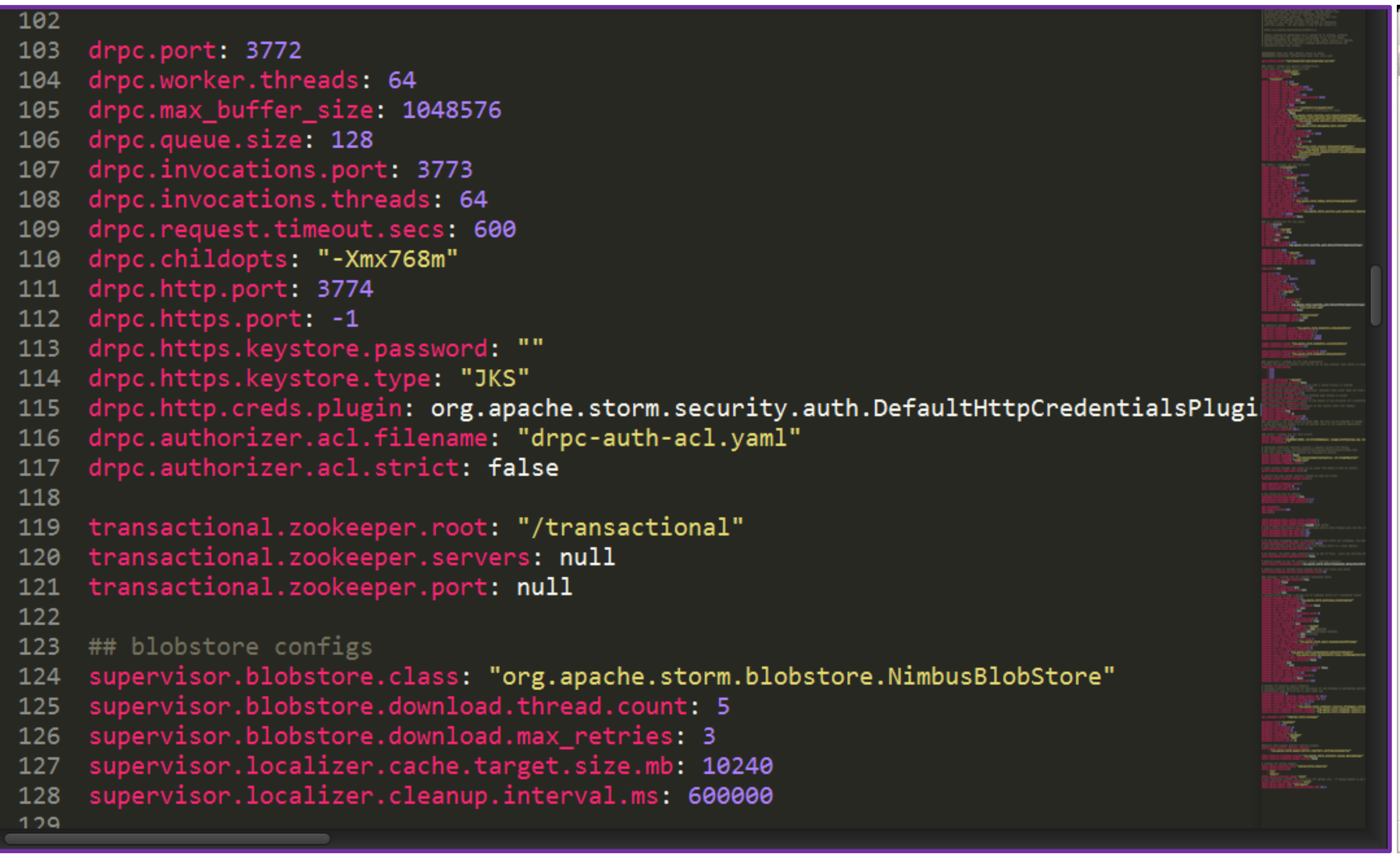
1- <0,0,0,0,0,0,0,1,1,1,1,1>
2- <0,0,0,0,0,0,0,1,0,0,0,0,0>
3- <0,0,0,0,0,1,0,0,0,0,0,0>
4- <0,0,0,0,0,0,1,1,0,0,0,0,0>
5- <0,0,0,0,1,0,0,0,0,0,0,0>
6- <0,0,0,0,1,0,1,0,0,0,0,0>
7- <0,0,0,0,1,1,0,0,0,0,0,0>
8- <0,0,0,0,0,1,1,1,0,0,0,0,0>
9- <0,0,0,1,0,0,0,0,0,0,0,0>
10-<0,0,0,1,0,0,1,0,0,0,0,0>

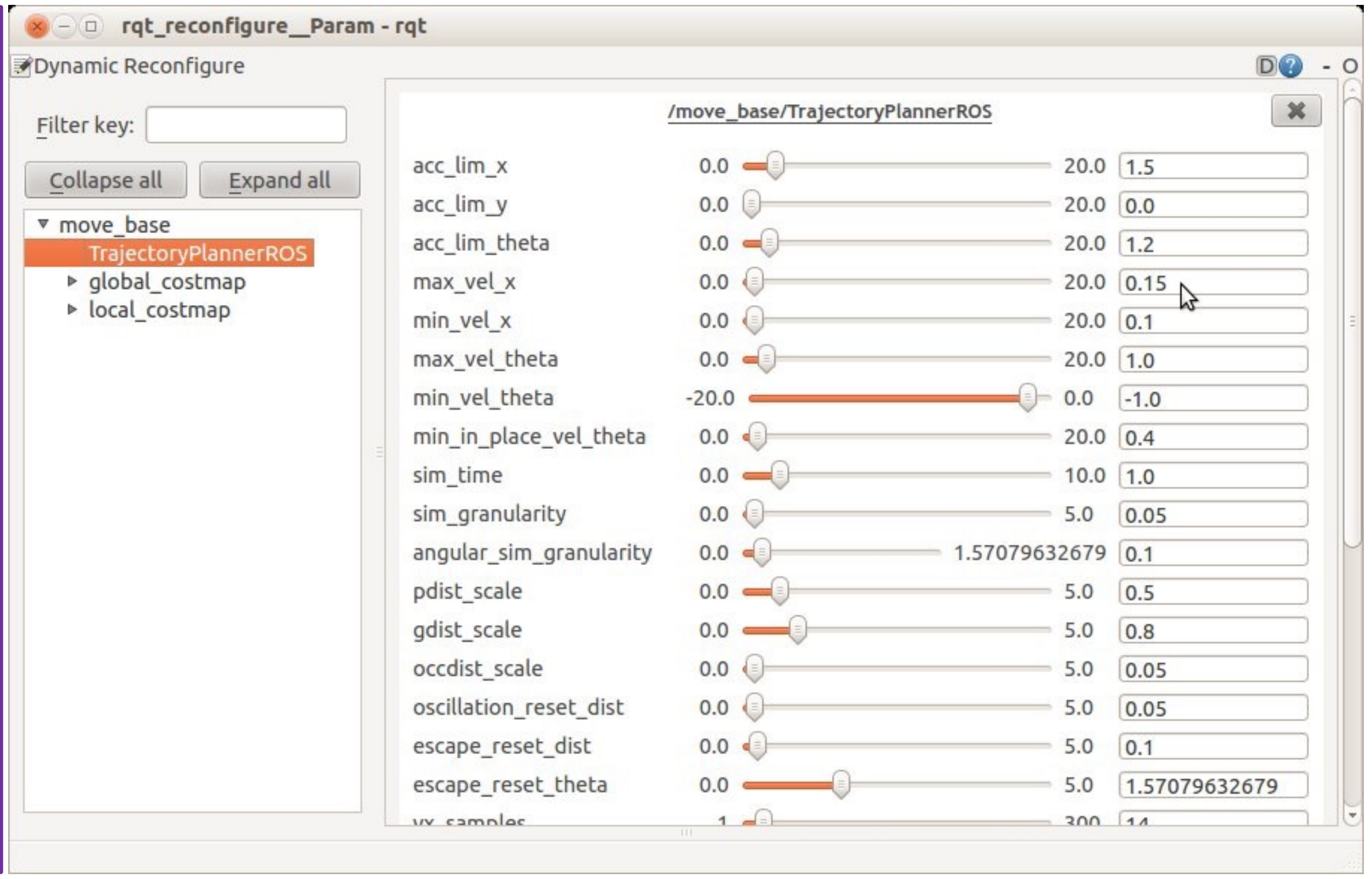
Insight: Learn a model on a cheaper workload to explore the expensive workload faster

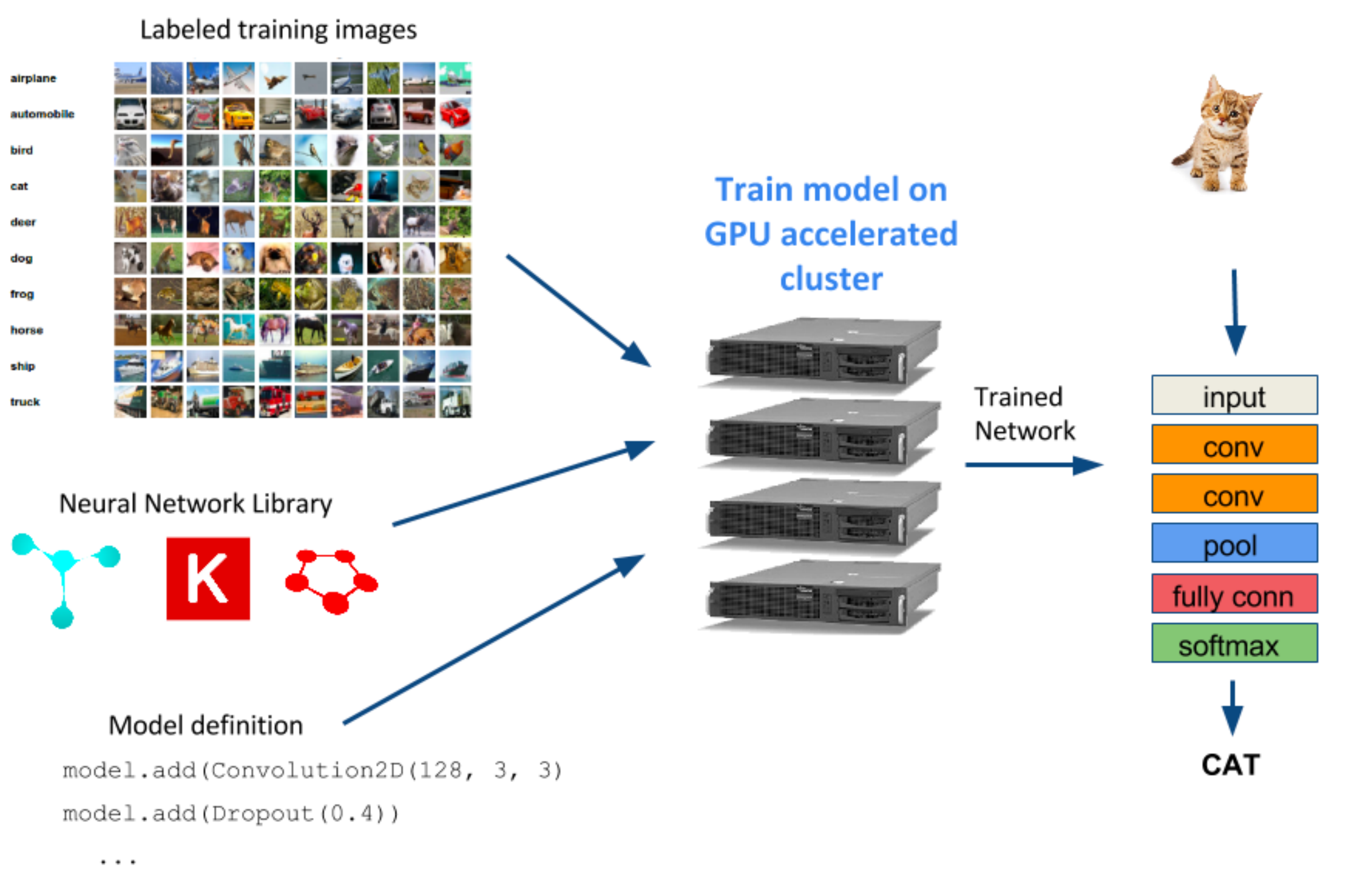




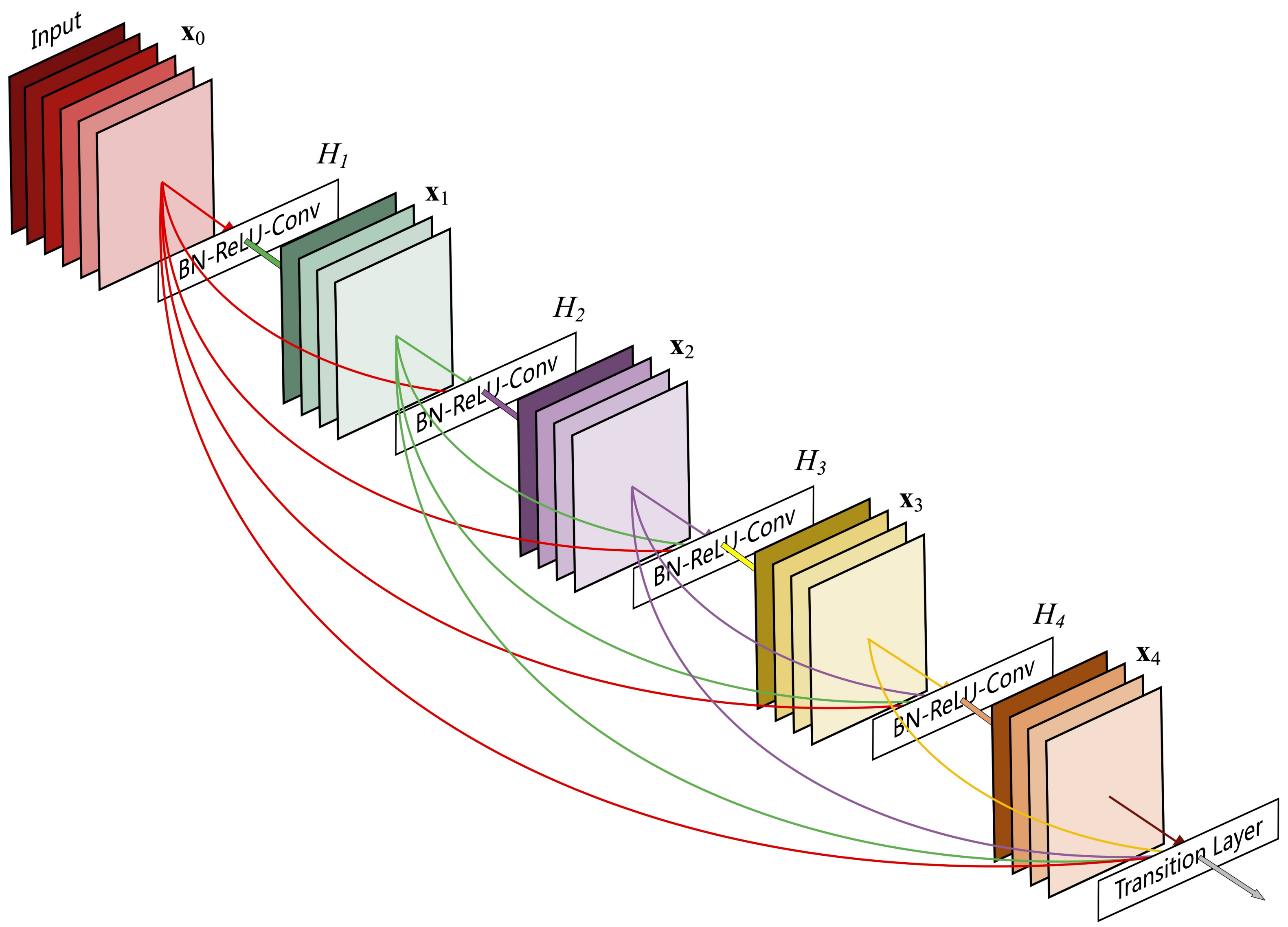
Many systems are now built configurable



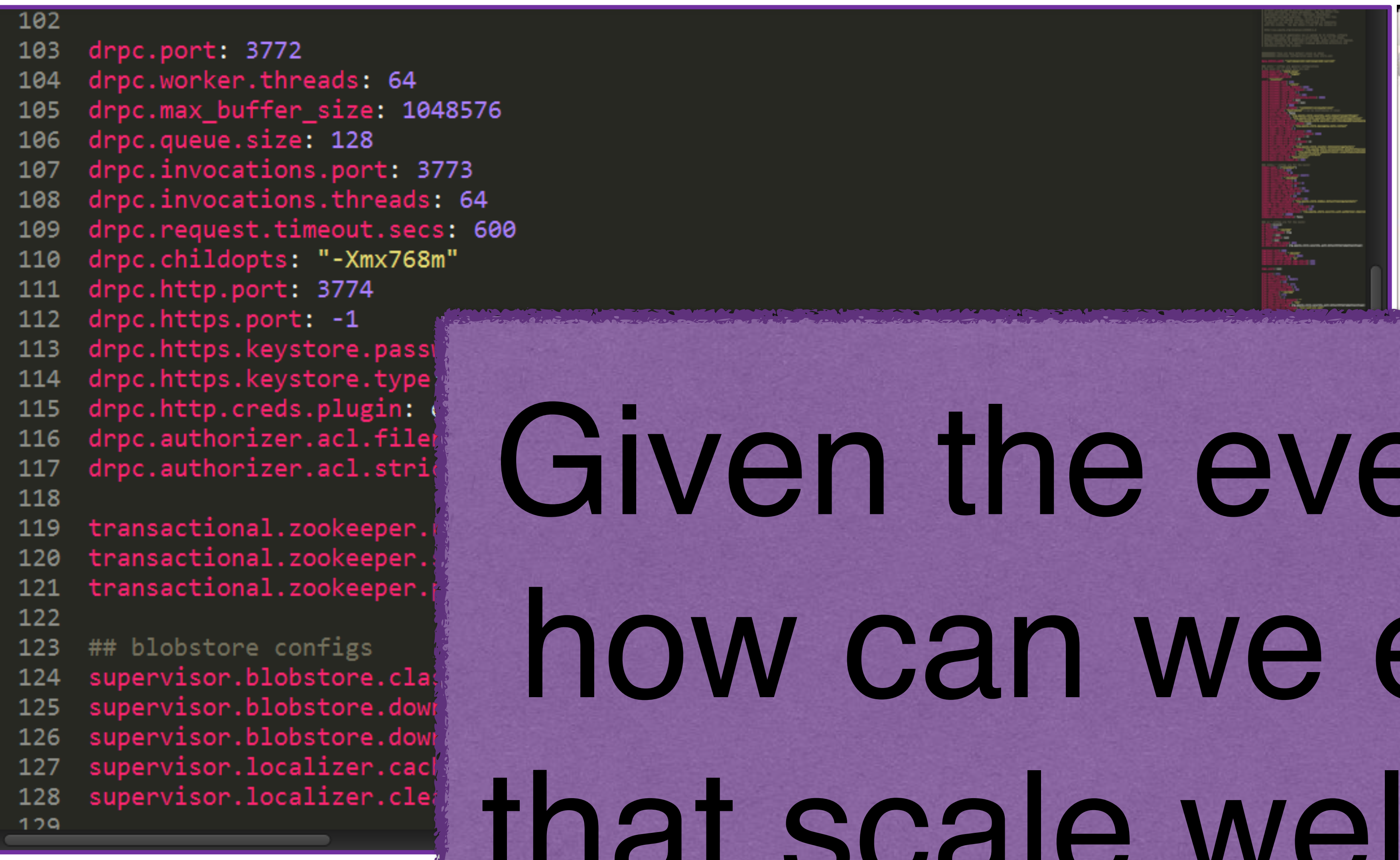


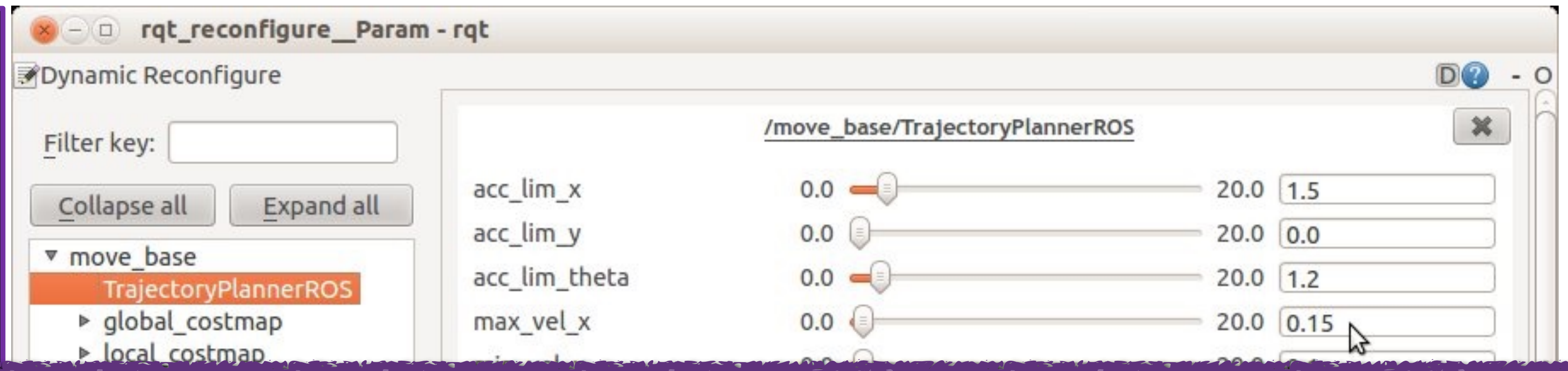


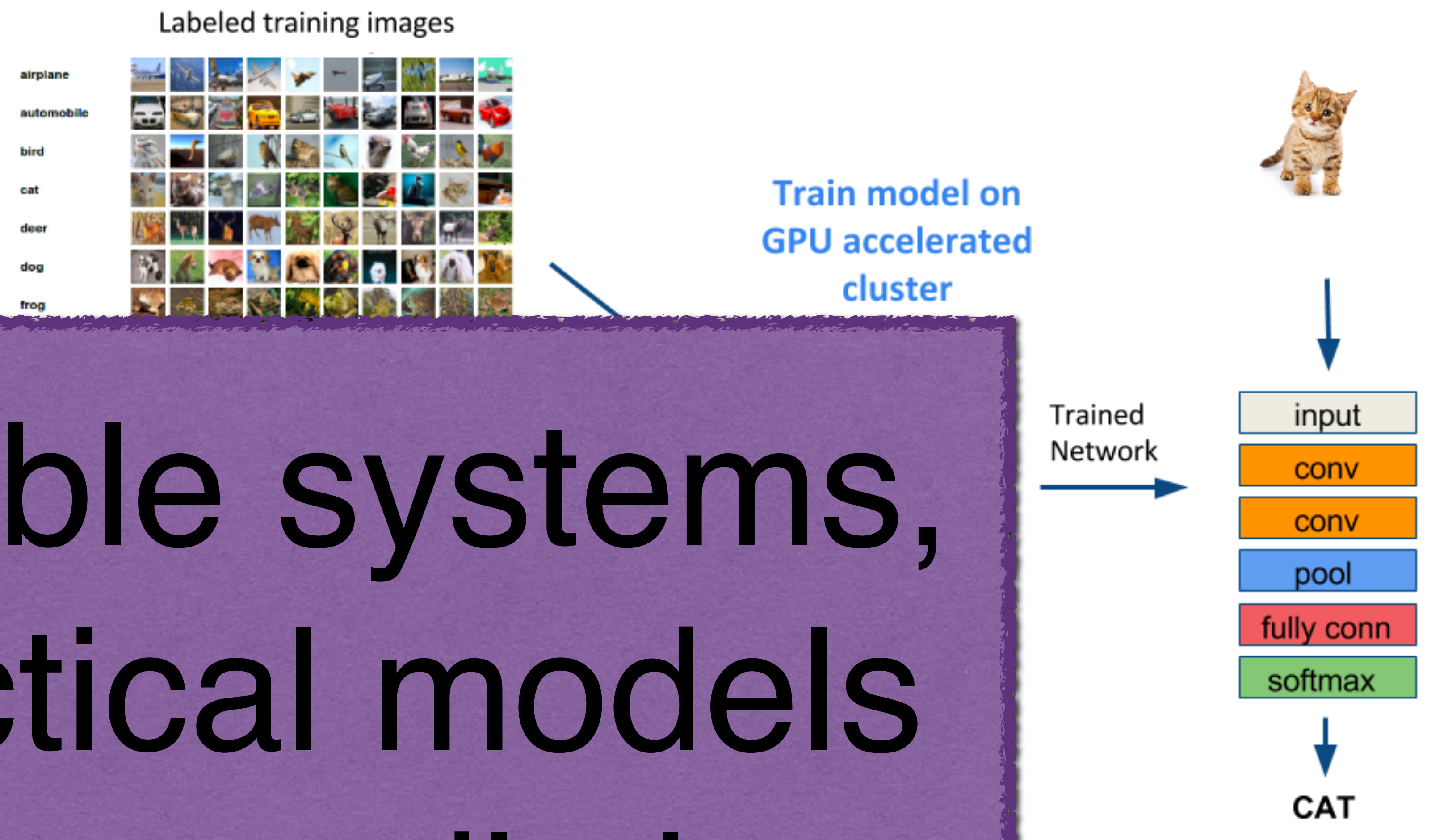
hadoop



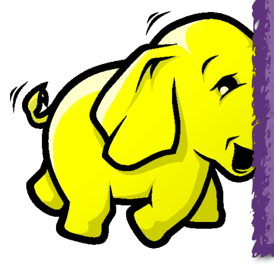
Many systems are now built configurable

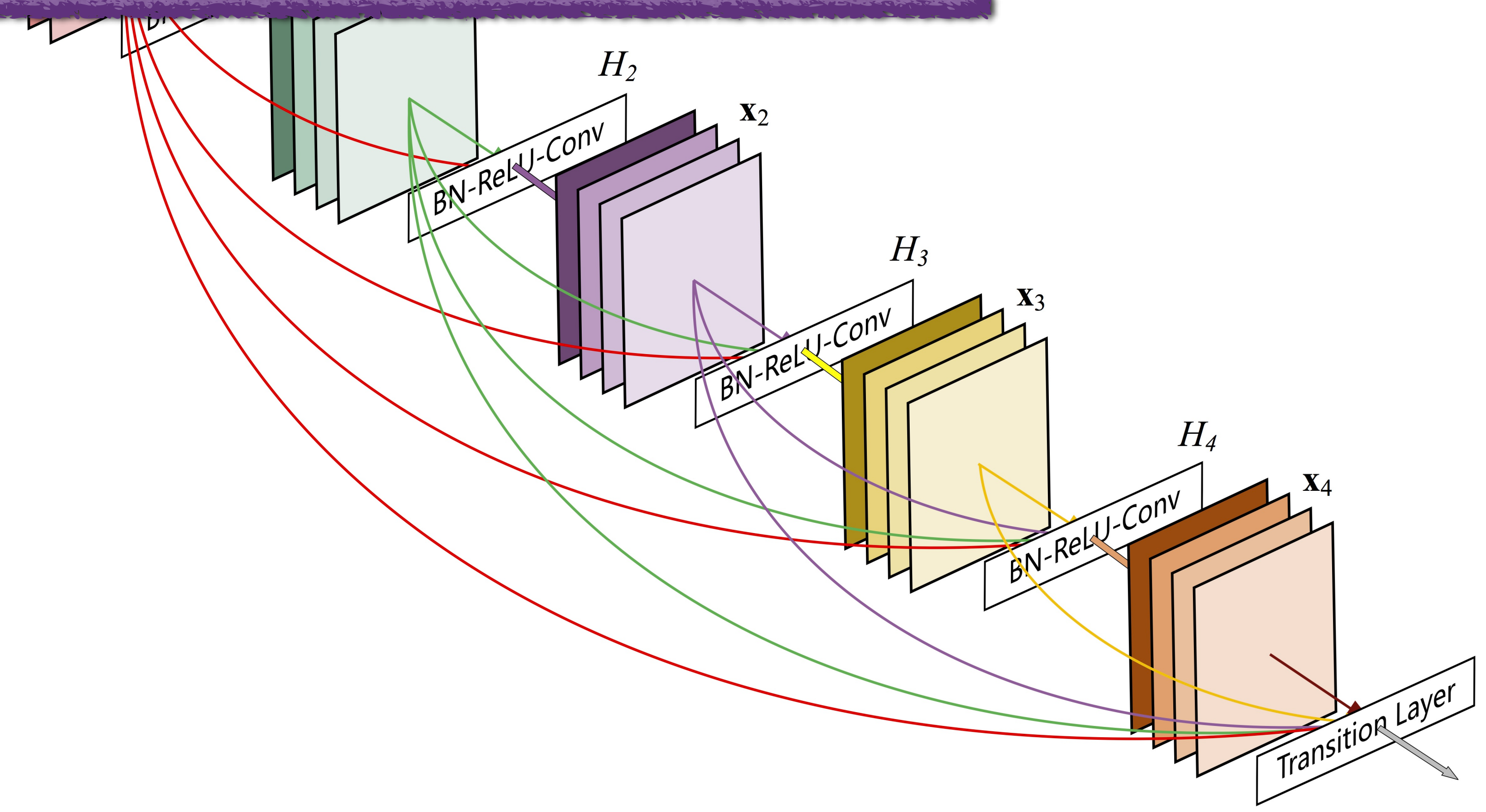




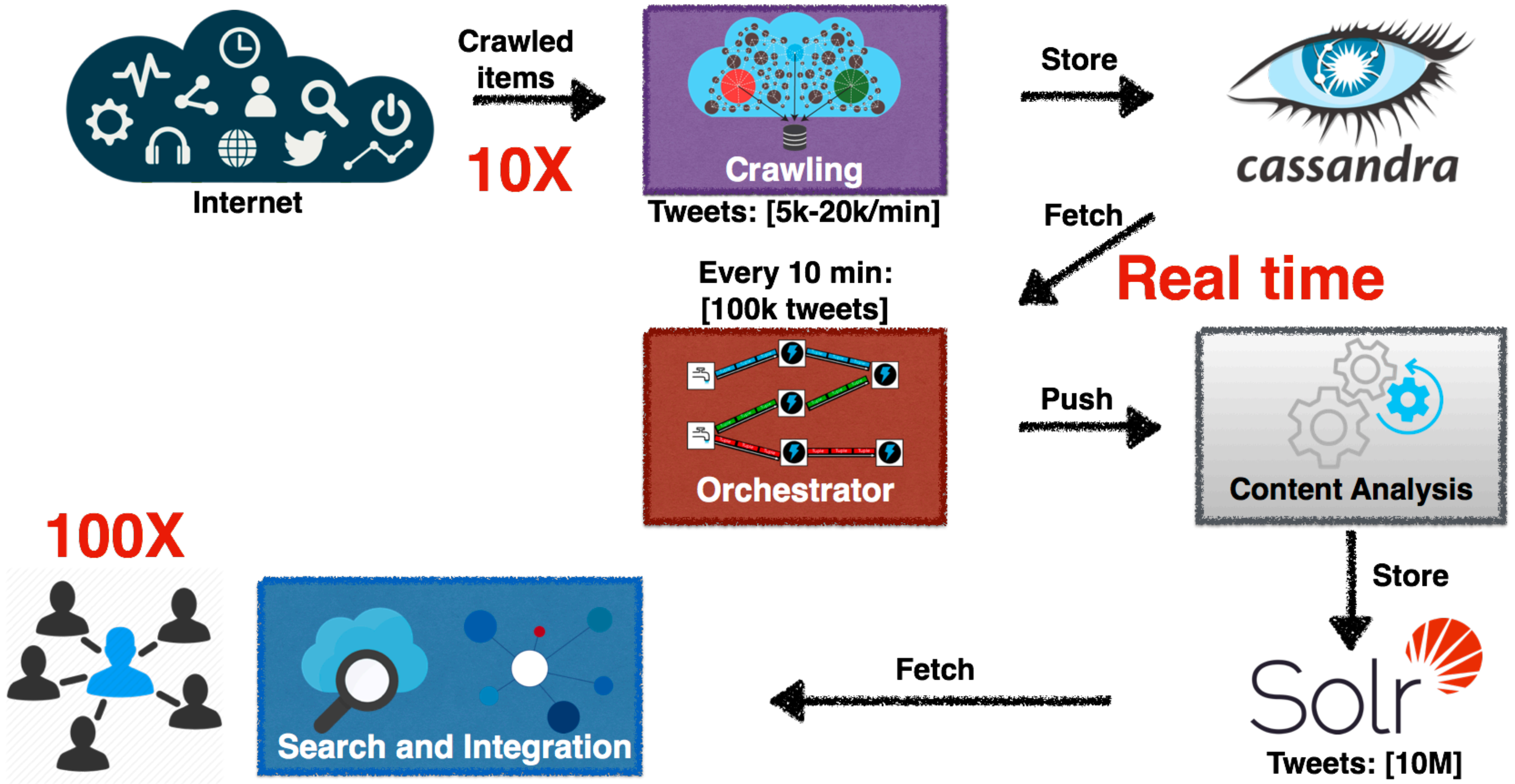


Given the ever growing configurable systems, how can we enable learning practical models that scale well and provide reliable predictions for exploring the configuration space?



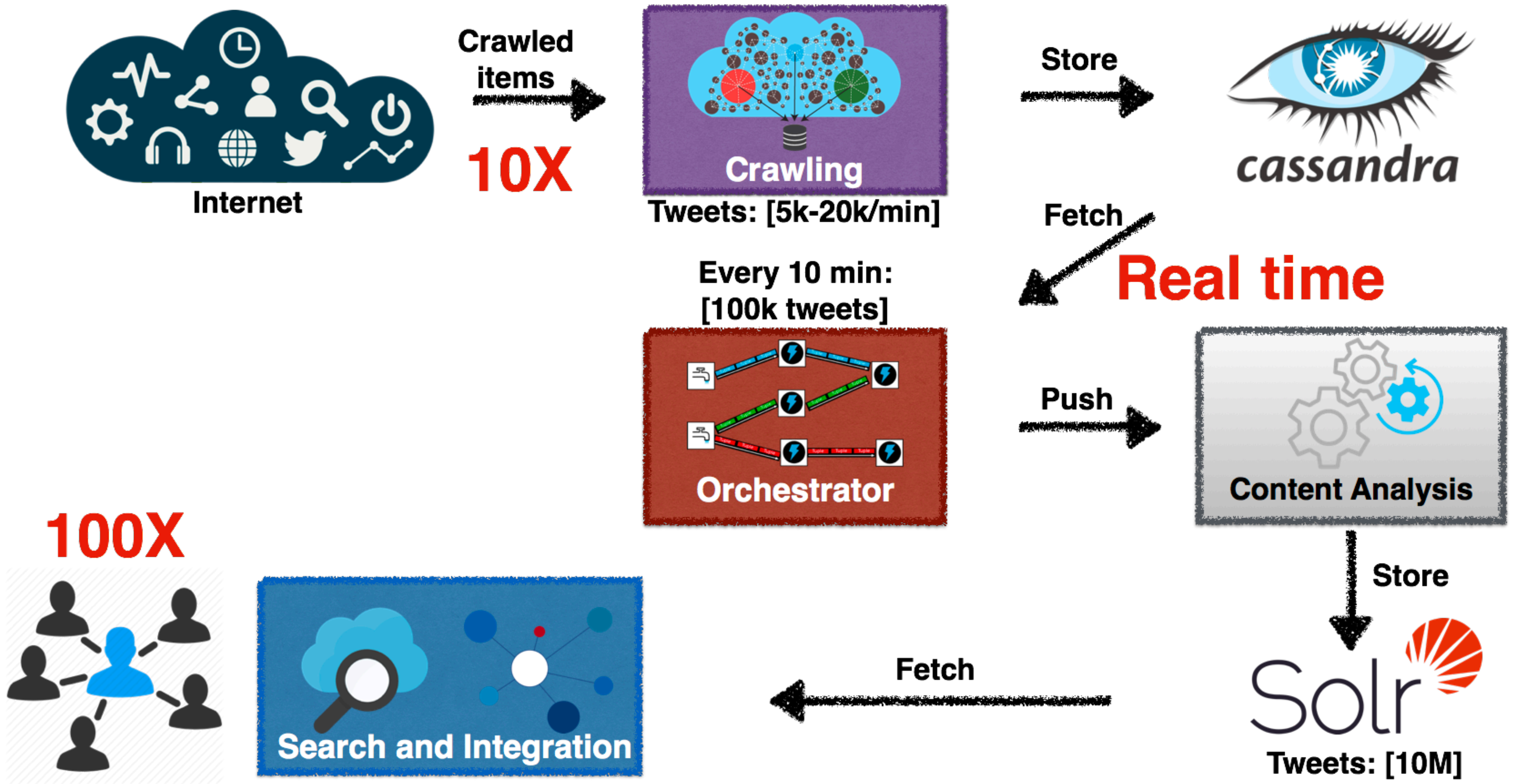


Challenges

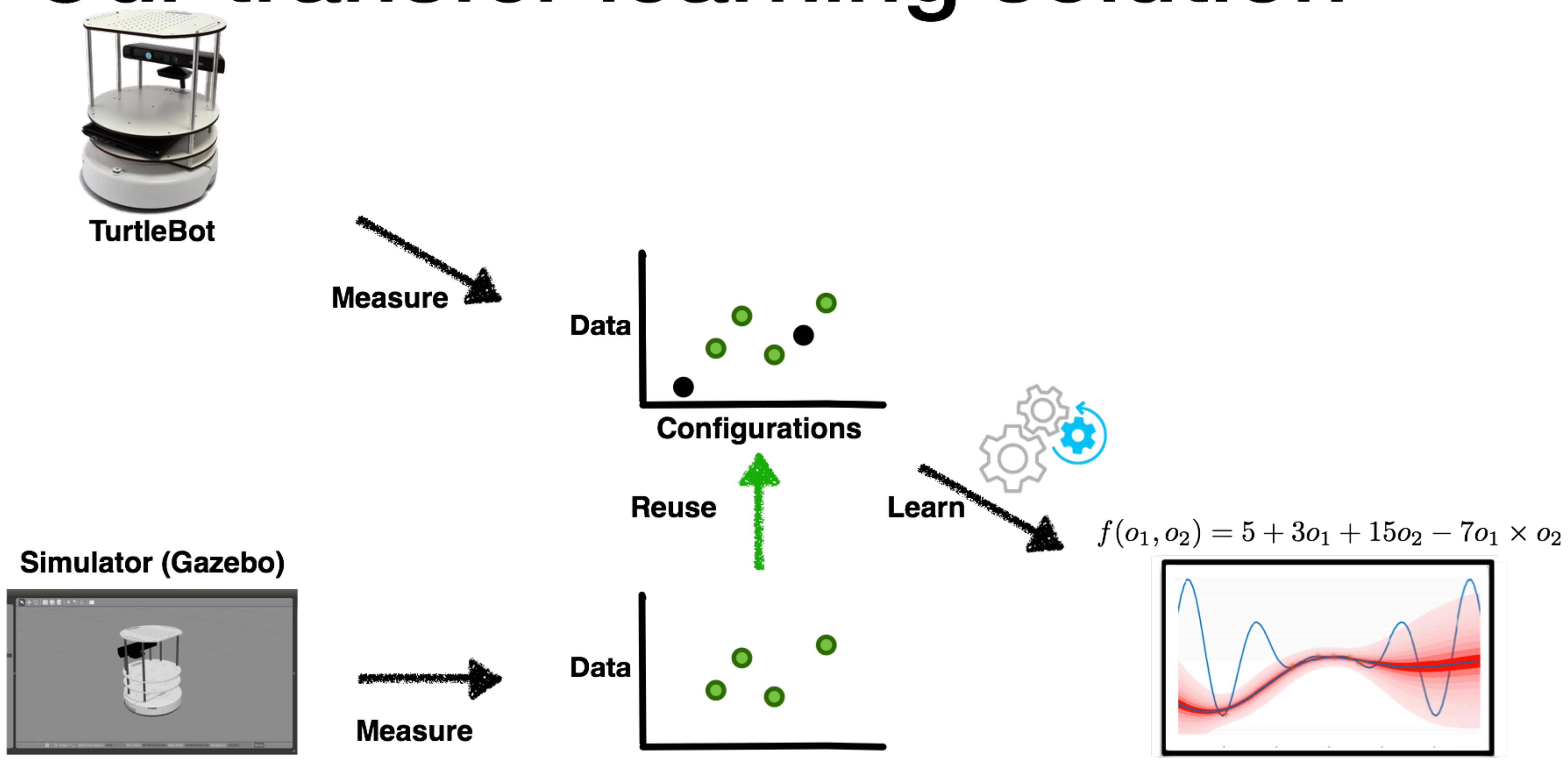


Internet
Crawled items
10X
Crawling
Tweets: [5k-20k/min]
Store
cassandra
Fetch
Every 10 min:
[100k tweets]
Real time
Orchestrator
Push
Content Analysis
Store
Solr
Tweets: [10M]
Fetch
100X
Search and Integration

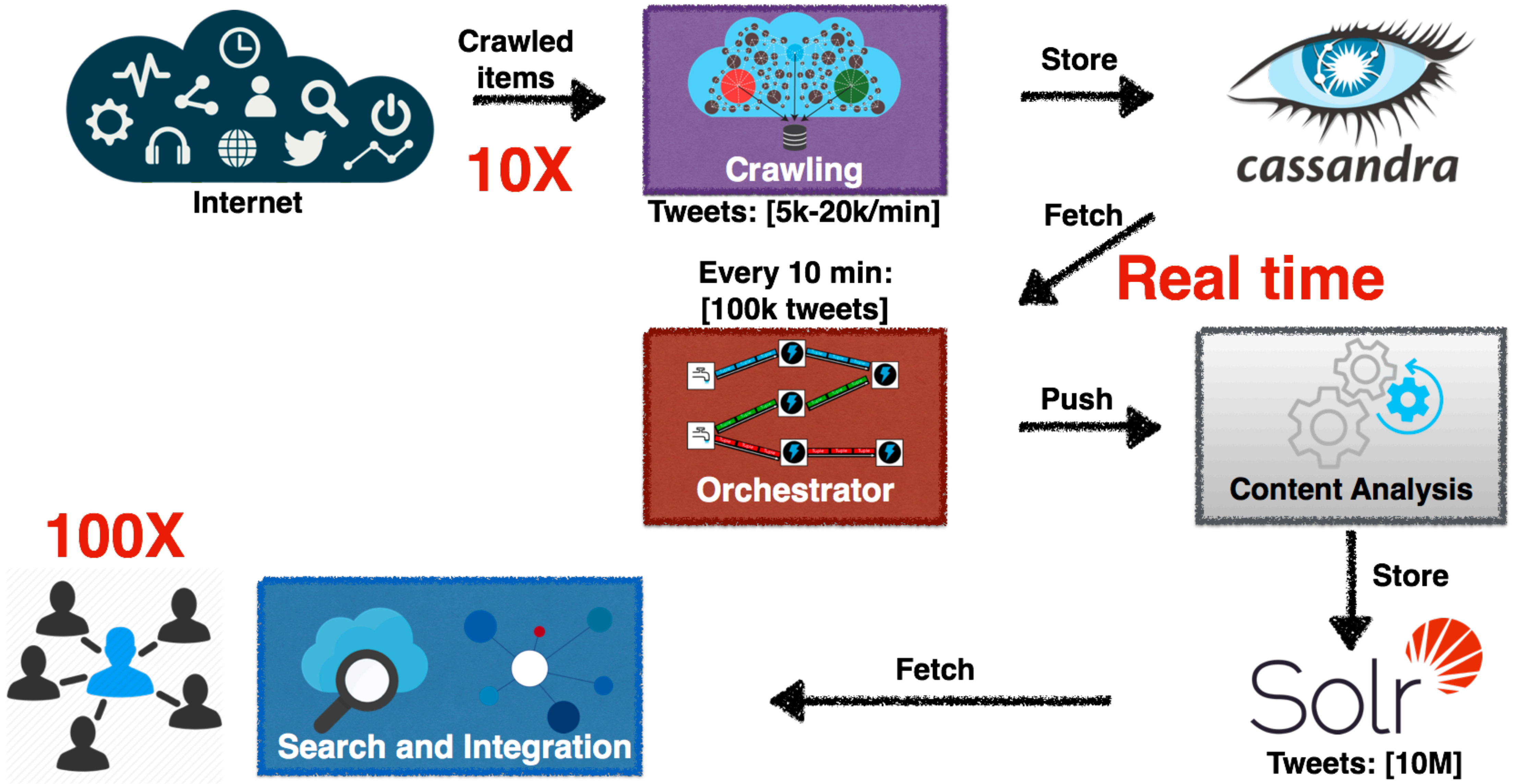
Challenges



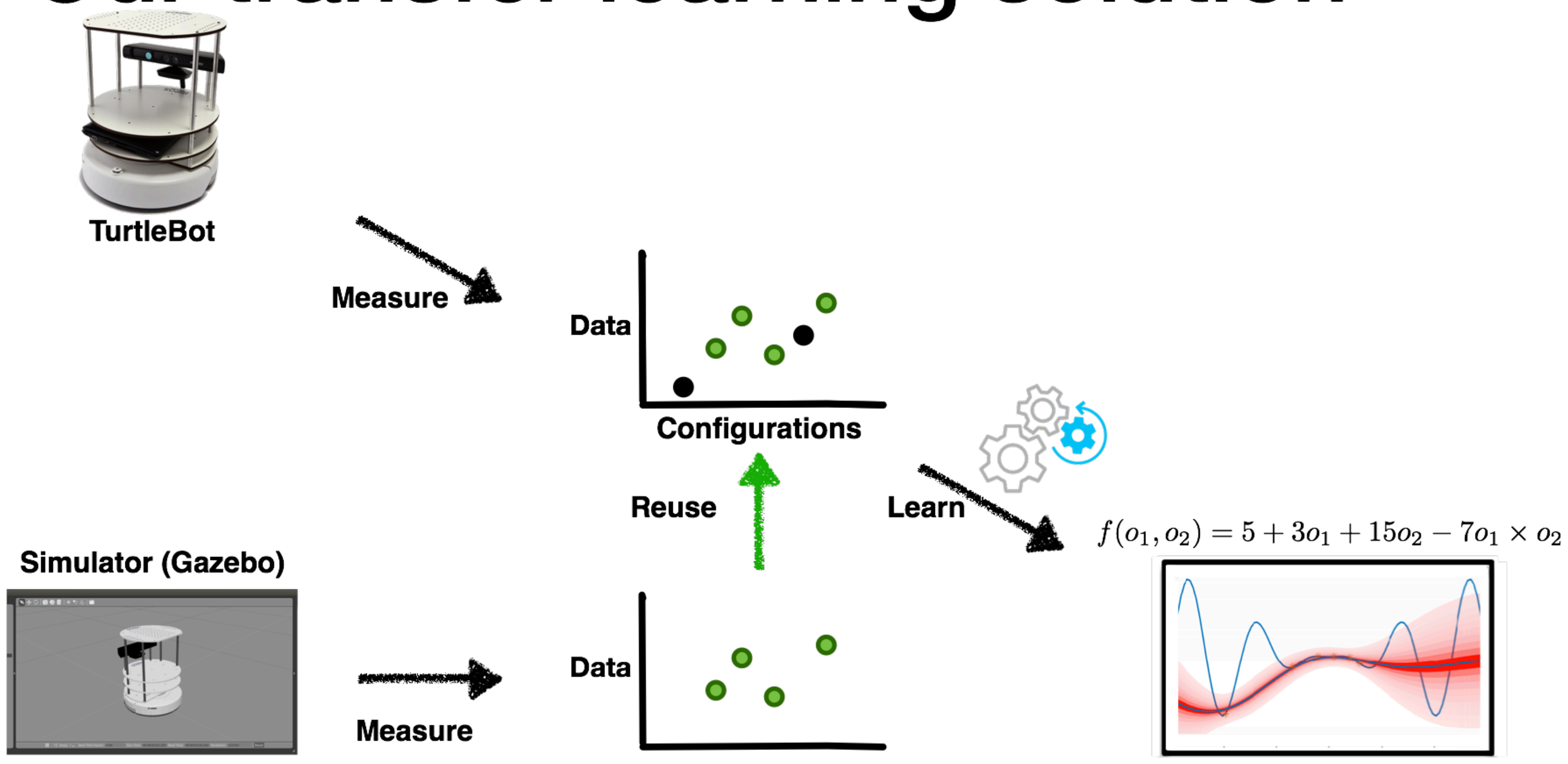
Our transfer learning solution



Challenges

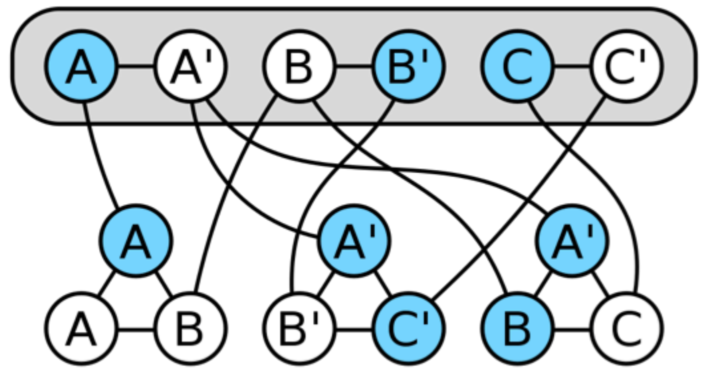


Our transfer learning solution



Our empirical study: We looked at different highly-configurable systems to gain insights

(A ∨ B) ∧ (¬A ∨ ¬B ∨ ¬C) ∧ (¬A ∨ B ∨ C)



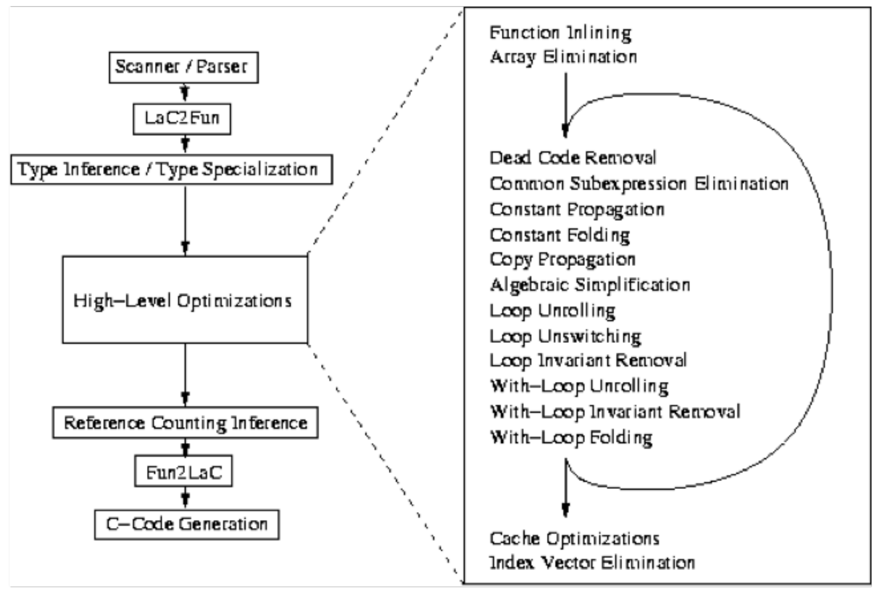
SPEAR (SAT Solver)
Analysis time
14 options

X264

X264 (video encoder)
Encoding time
16 options

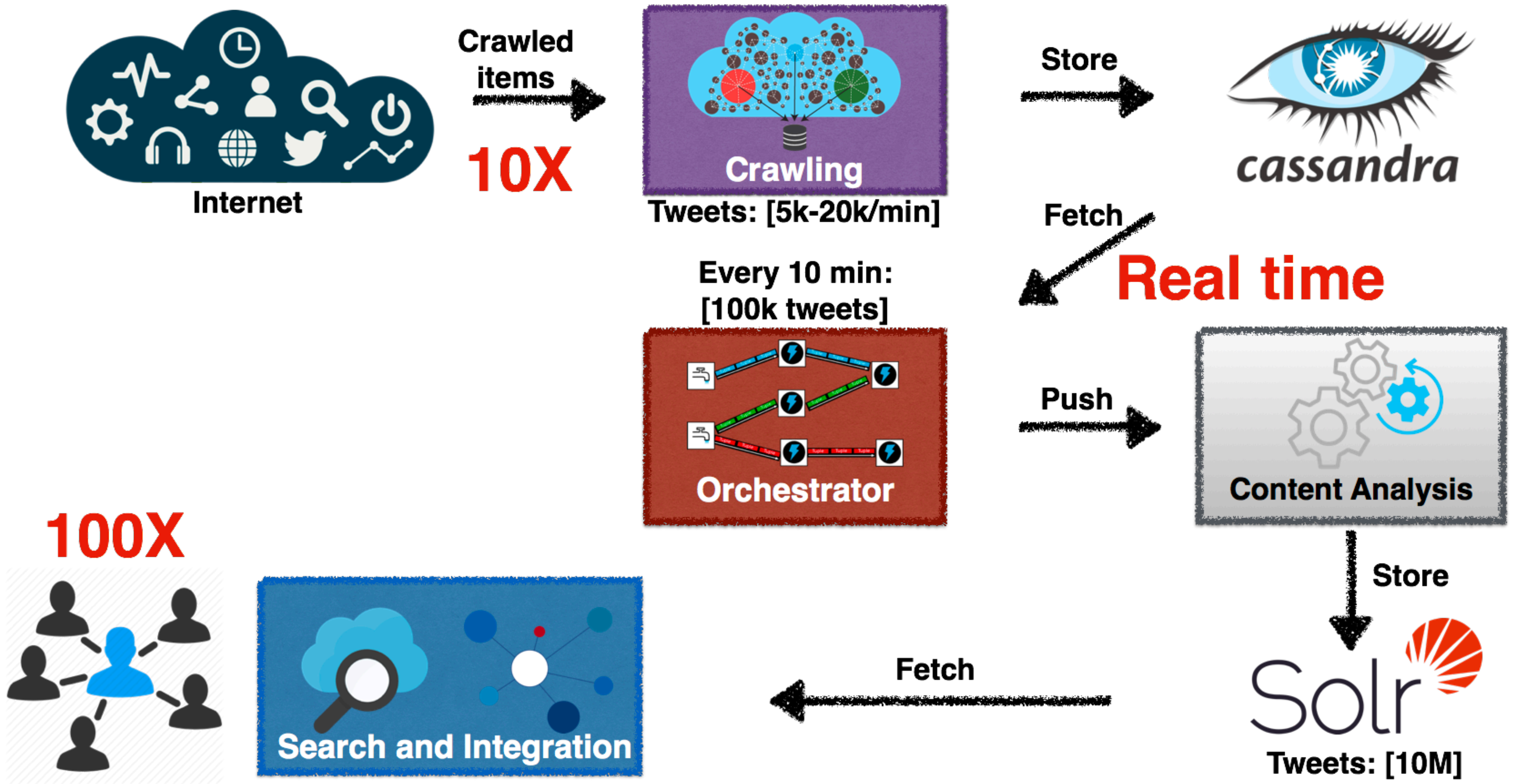
SQLite

SQLite (DB engine)
Query time
14 options

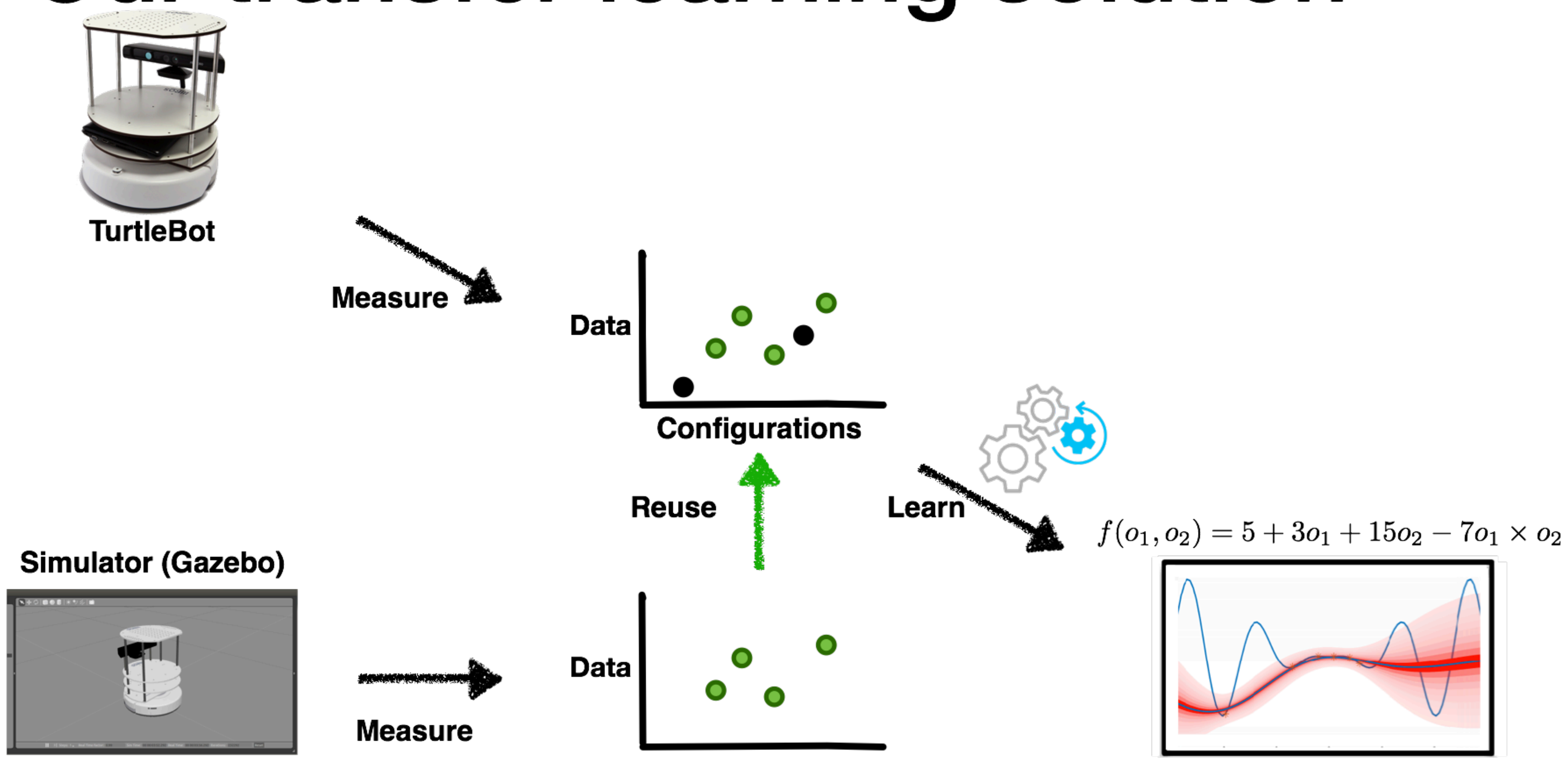


SaC (Compiler)
Execution time
50 options

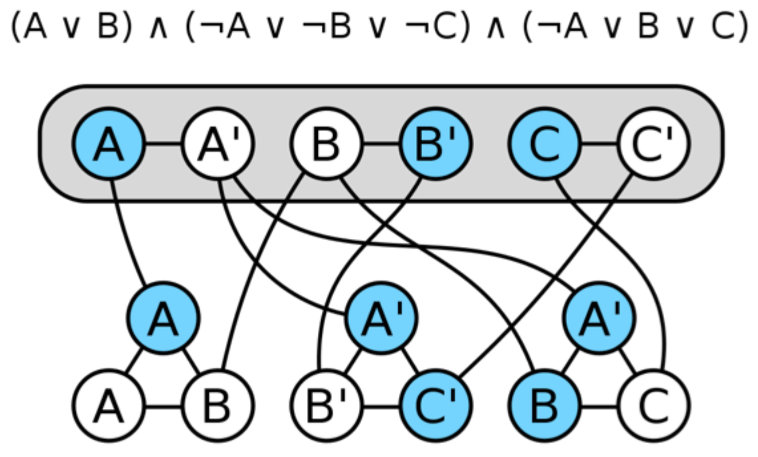
Challenges

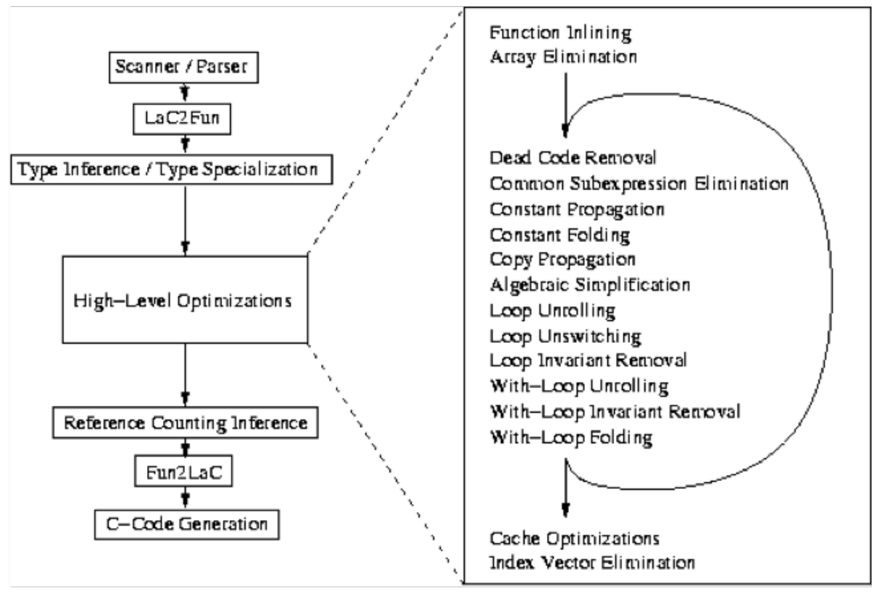


Our transfer learning solution



Our empirical study: We looked at different highly-configurable systems to gain insights





SPEAR (SAT Solver)
Analysis time
14 options

X264 (video encoder)
Encoding time
16 options

SQLite (DB engine)
Query time
14 options

SaC (Compiler)
Execution time
50 options

Exploring the design space of deep networks