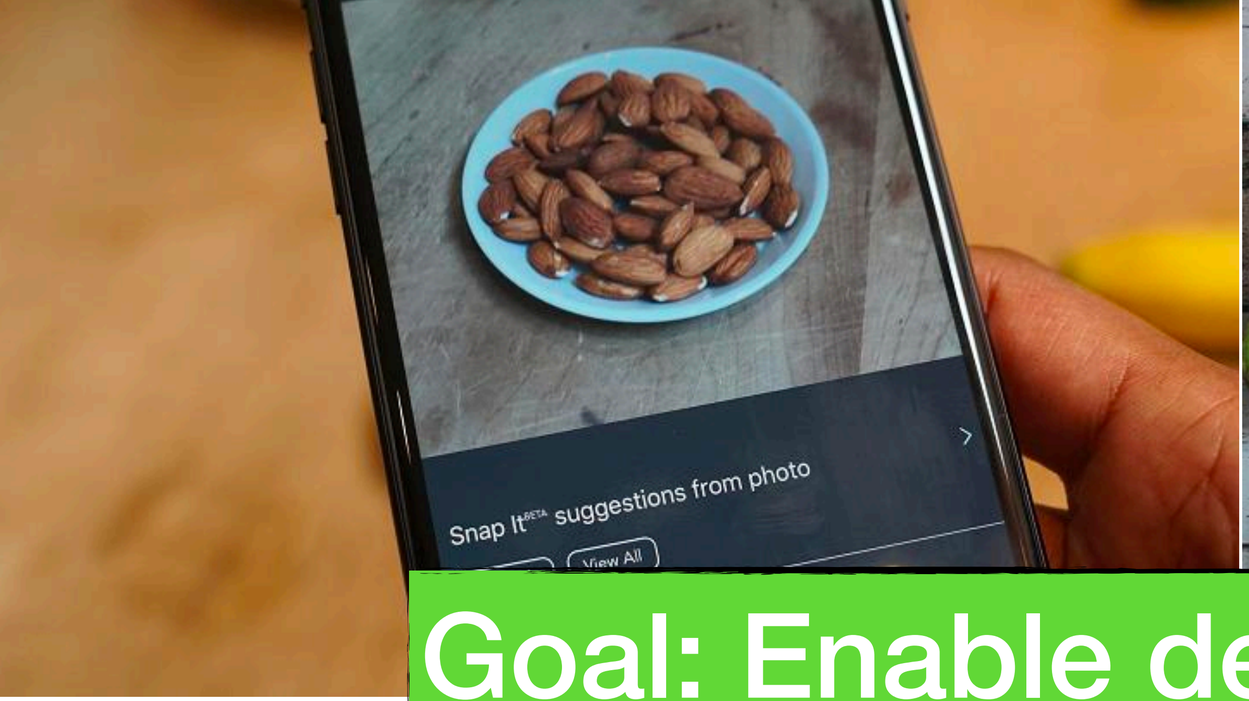
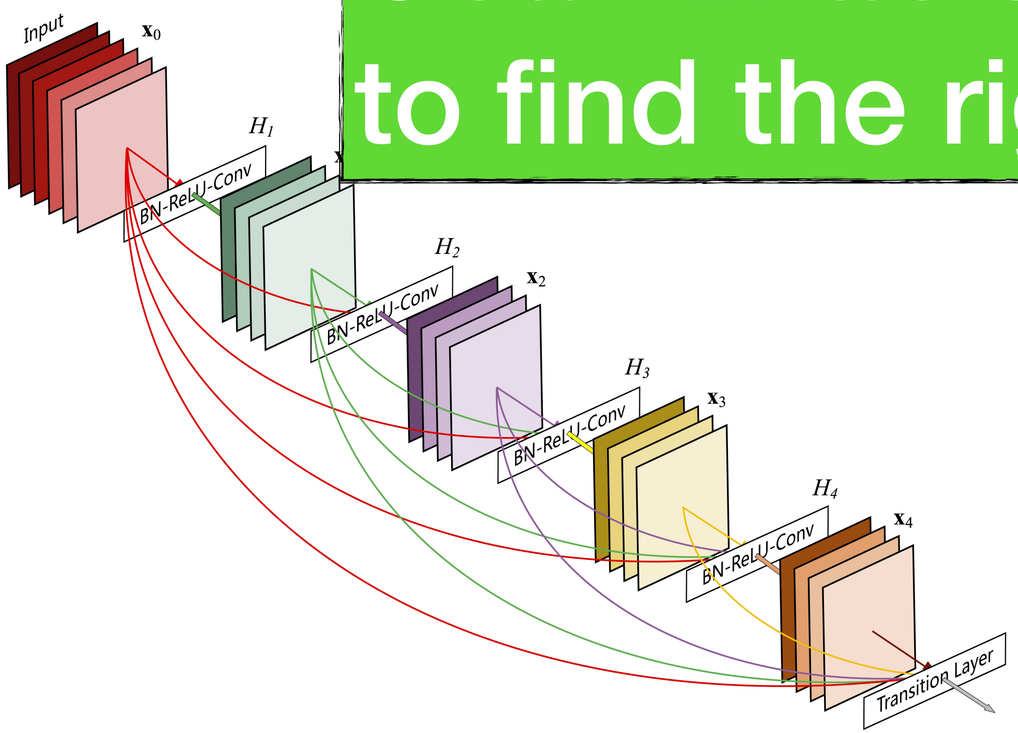
Transfer Learning for Performance Analysis of Highly-Configurable Systems

Pooyan Jamshidi
University of South Carolina
pooyanjamshidi



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

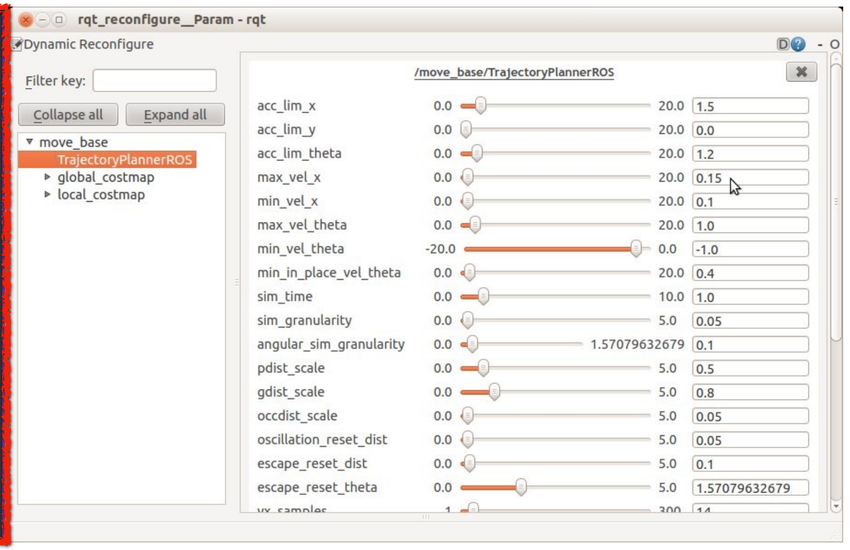


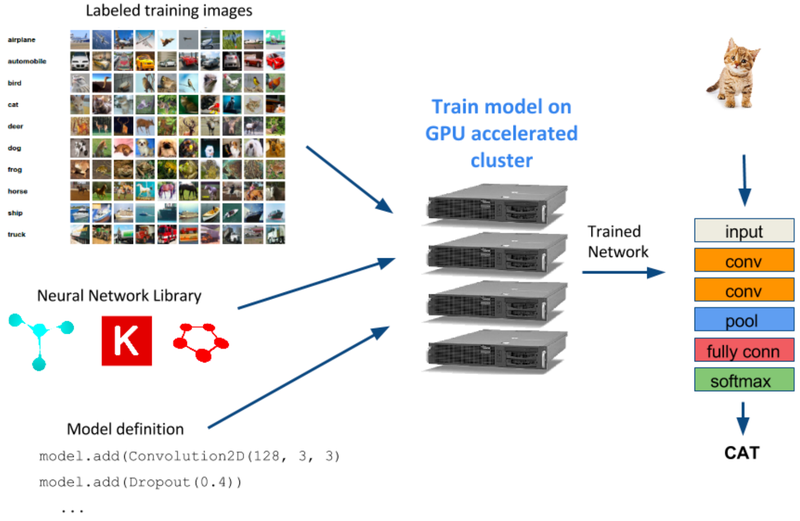
Today's most popular systems are built configurable

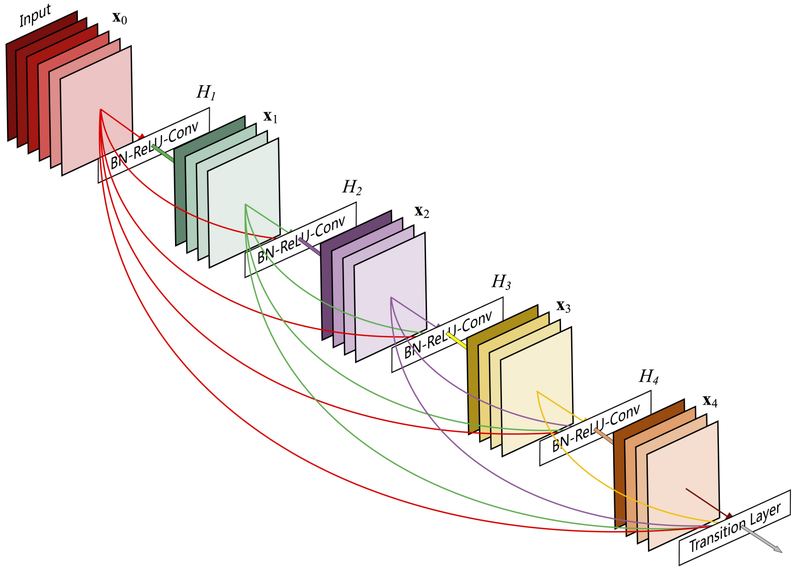
```
drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "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
```

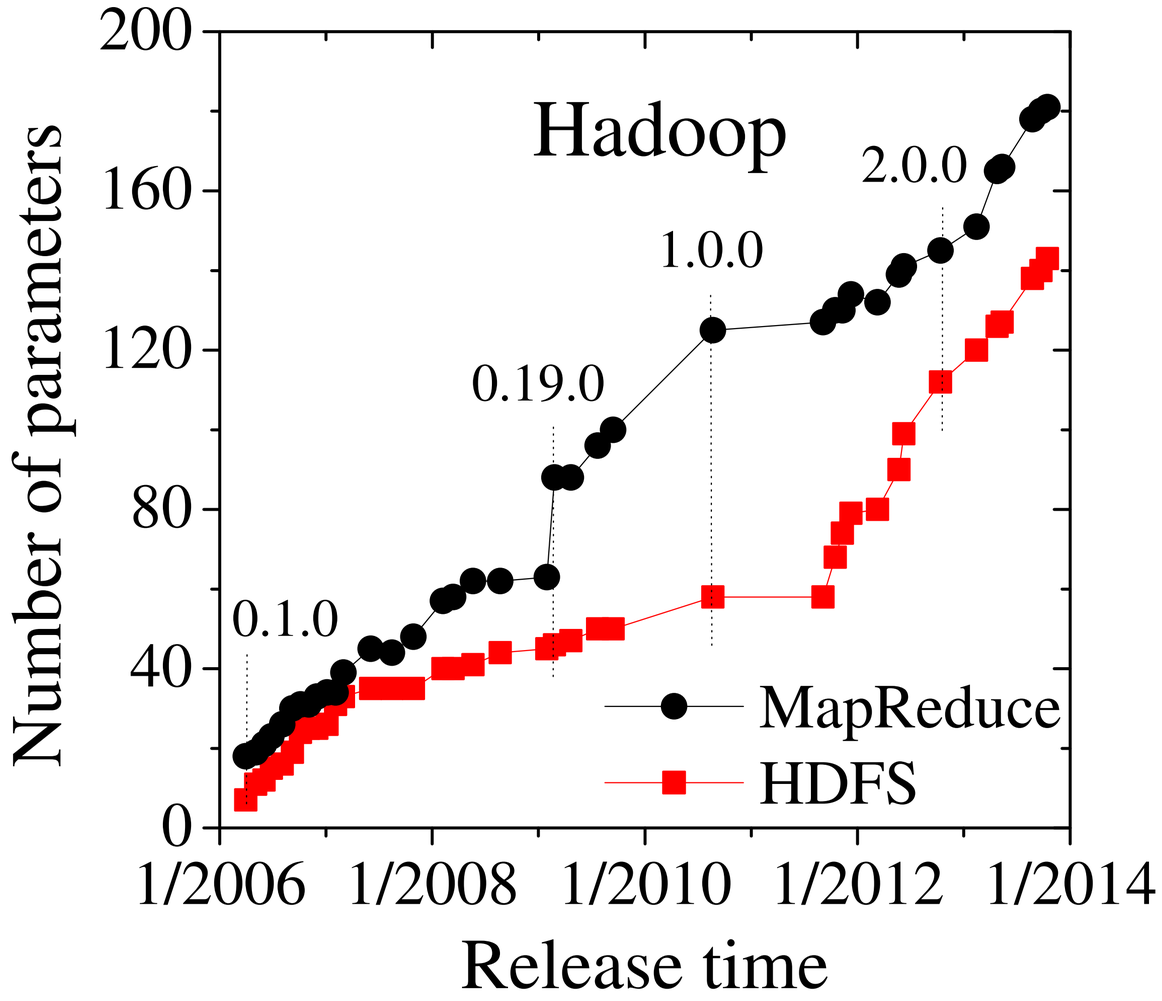






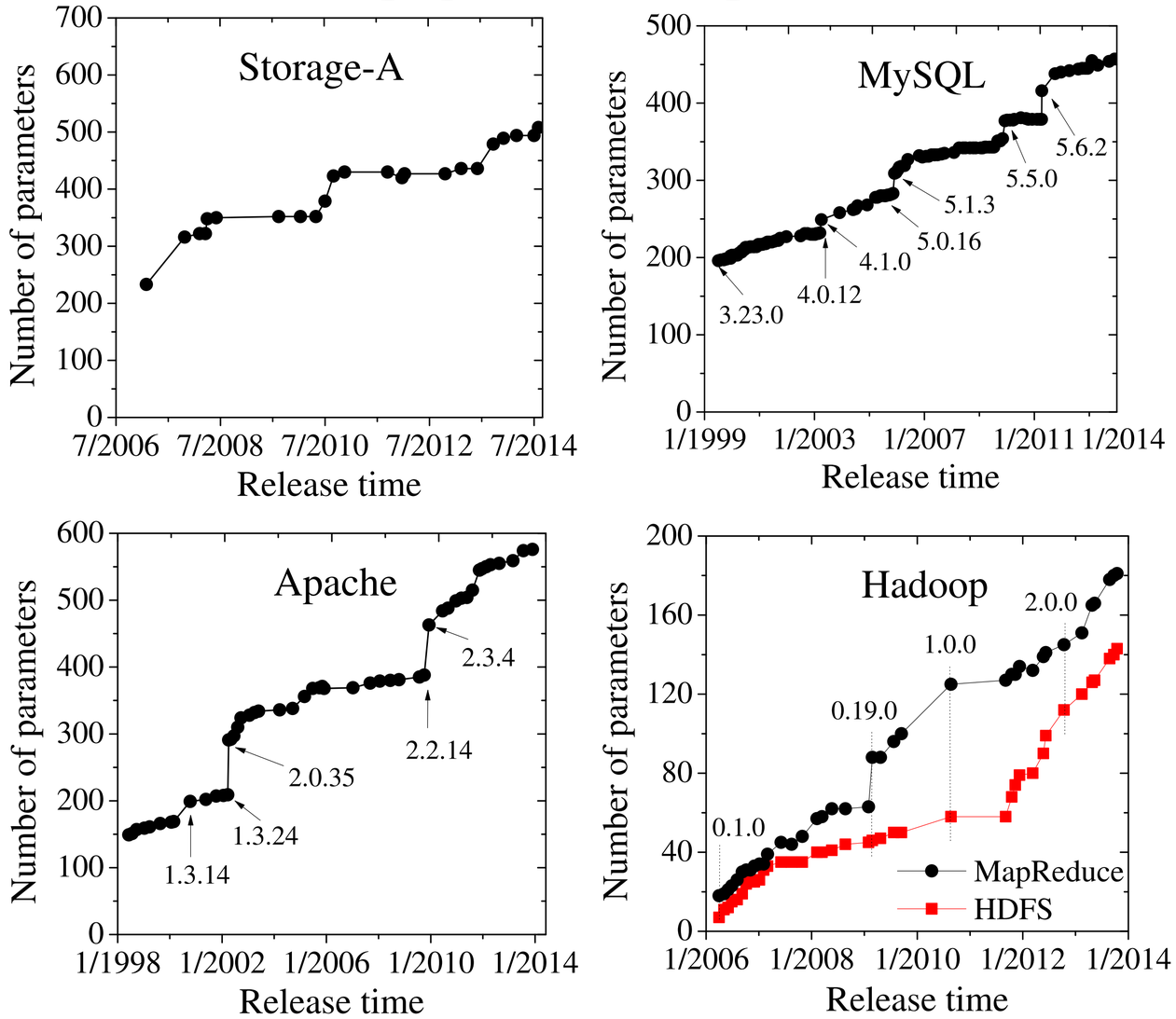
```
102
103 drpc.port: 3772
104 drpc.worker.threads: 64
105 drpc.max_buffer_size: 1048576
106 drpc.queue.size: 128
107 drpc.invocations.port: 3773
108 drpc.invocations.threads: 64
109 drpc.request.timeout.secs: 600
110 drpc.childopts: "-Xmx768m"
111 drpc.http.port: 3774
112 drpc.https.port: -1
113 drpc.https.keystore.password: ""
114 drpc.https.keystore.type: "JKS"
115 drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
116 drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
117 drpc.authorizer.acl.strict: false
118
119 transactional.zookeeper.root: "/transactional"
120 transactional.zookeeper.servers: null
121 transactional.zookeeper.port: null
122
123 ## blobstore configs
124 supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
125 supervisor.blobstore.download.thread.count: 5
126 supervisor.blobstore.download.max_retries: 3
127 supervisor.localizer.cache.target.size.mb: 10240
128 supervisor.localizer.cleanup.interval.ms: 600000
129
```

Empirical observations confirm that systems are becoming increasingly configurable



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

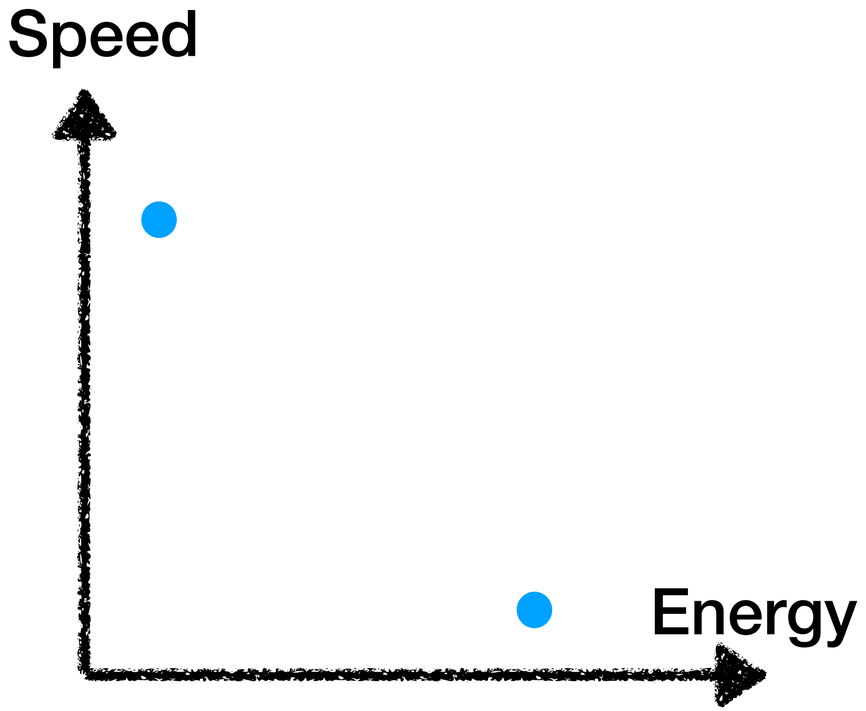
Empirical observations confirm that systems are becoming increasingly configurable



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

Configurations determine the performance behavior

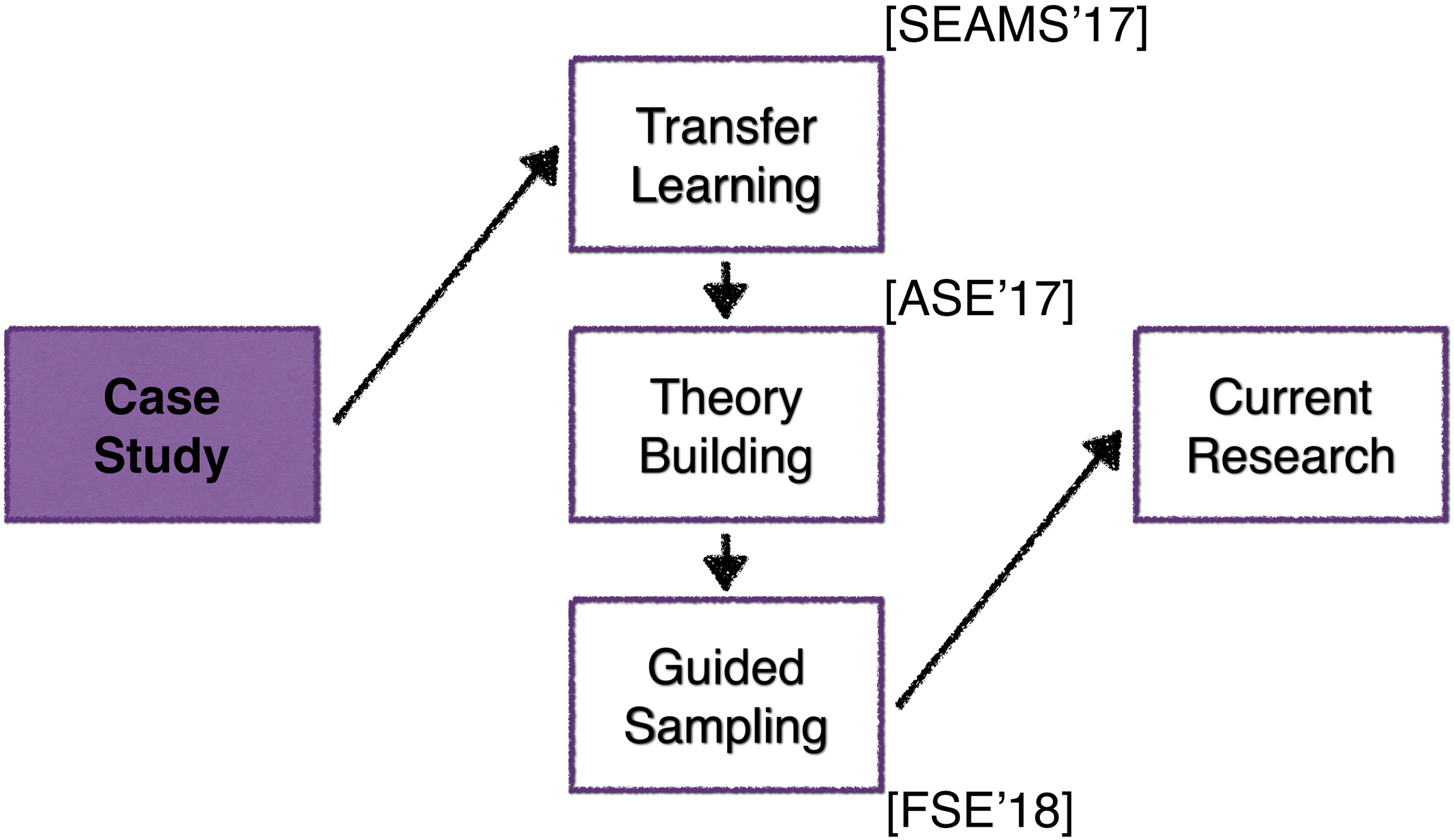
```
void Parrot_setenv(. . . name,. . . value){
#ifdef PARROT_HAS_SETENV
  my_setenv(name, value, 1);
#else
  int name_len=strlen(name);
  int val_len=strlen(value);
  char* envs=glob_env;
  if(envs==NULL){
    return;
  }
  strcpy(envs,name);
  strcpy(envs+name_len,"=");
  strcpy(envs+name_len + 1,value);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



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


[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

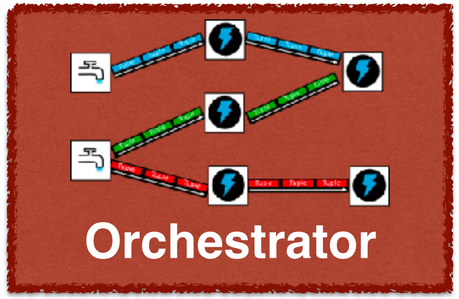
Crawled items

Tweets: [5k-20k/min]

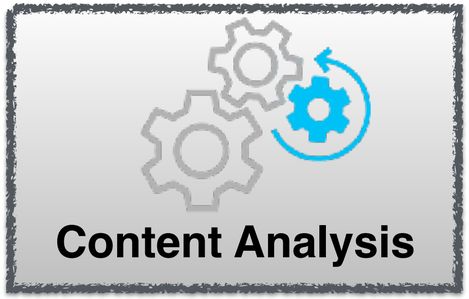
Store

Fetch

Every 10 min:
[100k tweets]



Push



Store

Tweets: [10M]

Fetch

Challenges

Internet

Crawled items

10X

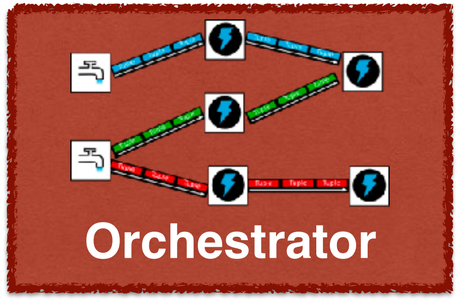
Tweets: [5k-20k/min]

Store

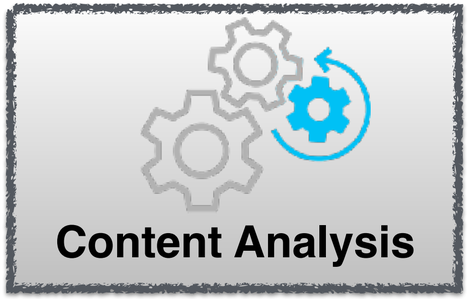
Fetch

Real time

Every 10 min:
[100k tweets]



Push



Store

Tweets: [10M]

Fetch

100X

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

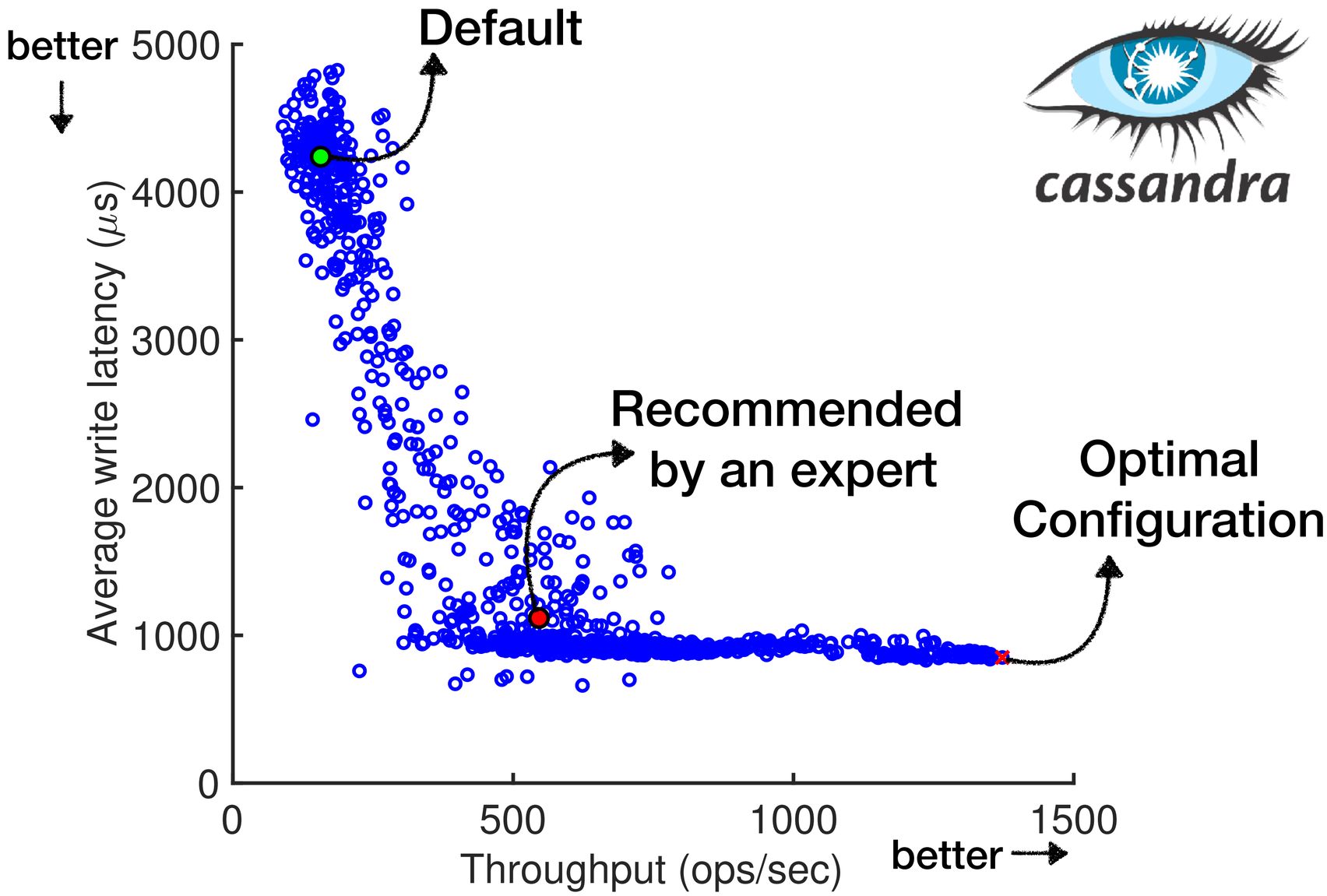
> 100

> 100

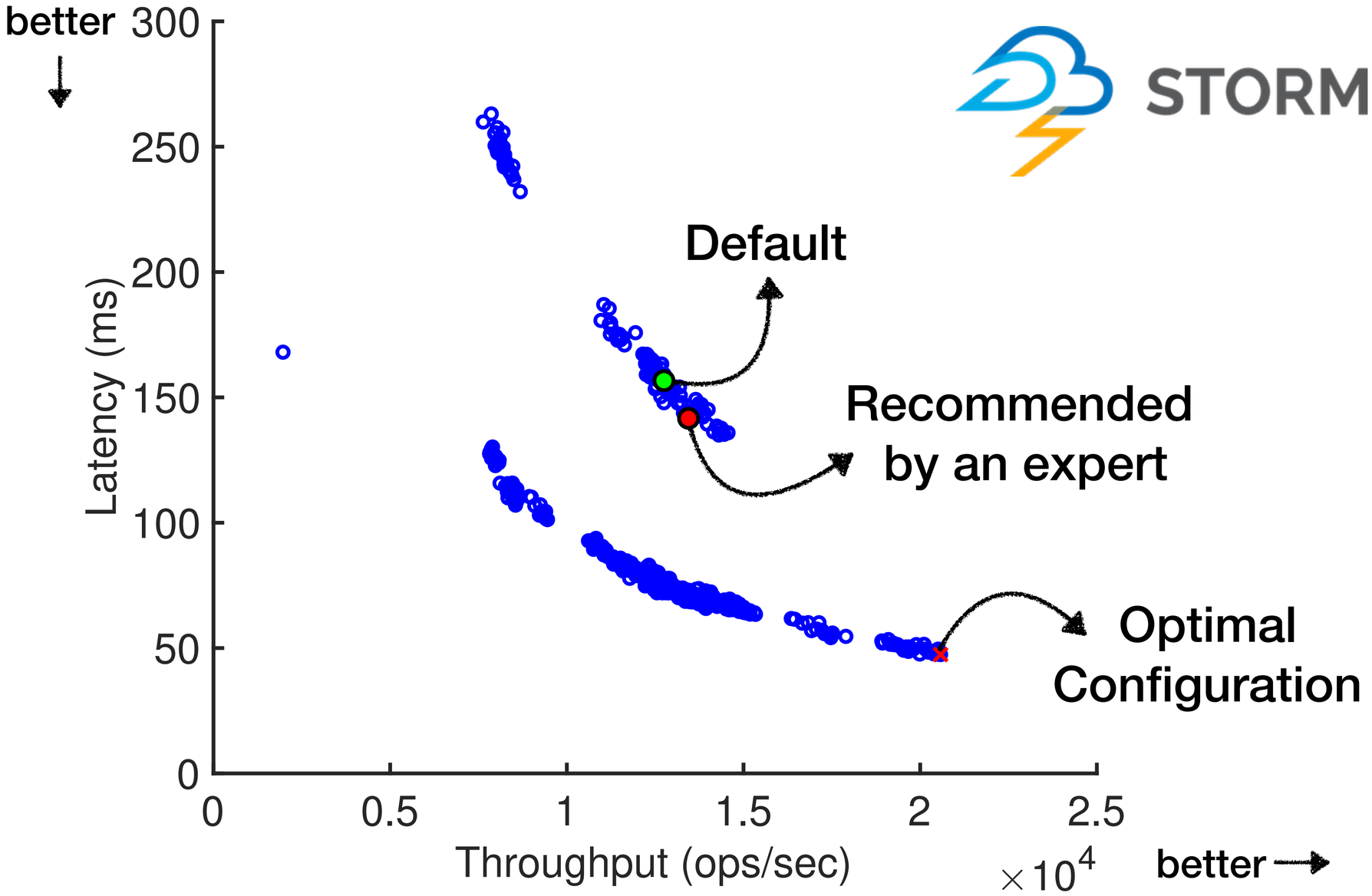
> 100

$$2^{300}$$

Default configuration was bad, so was the expert’



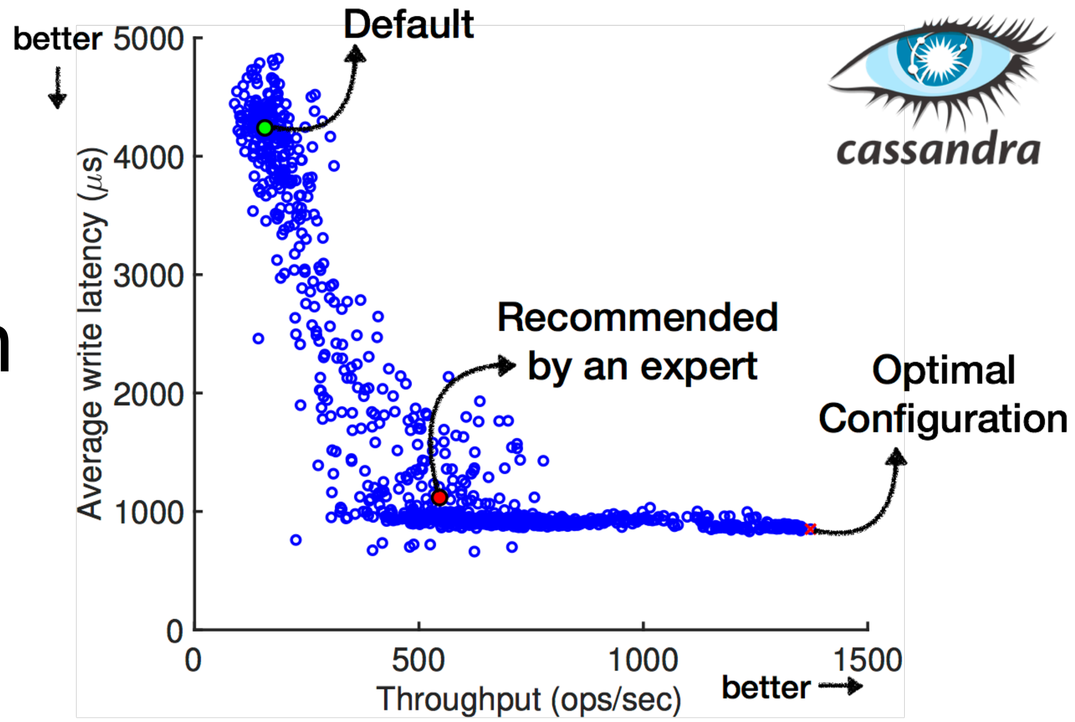
Default configuration was bad, so was the expert’



Why this is an important problem?

Significant time saving

- 2X-10X faster than worst
- Noticeably faster than median
- Default is bad
- Expert's is not optimal



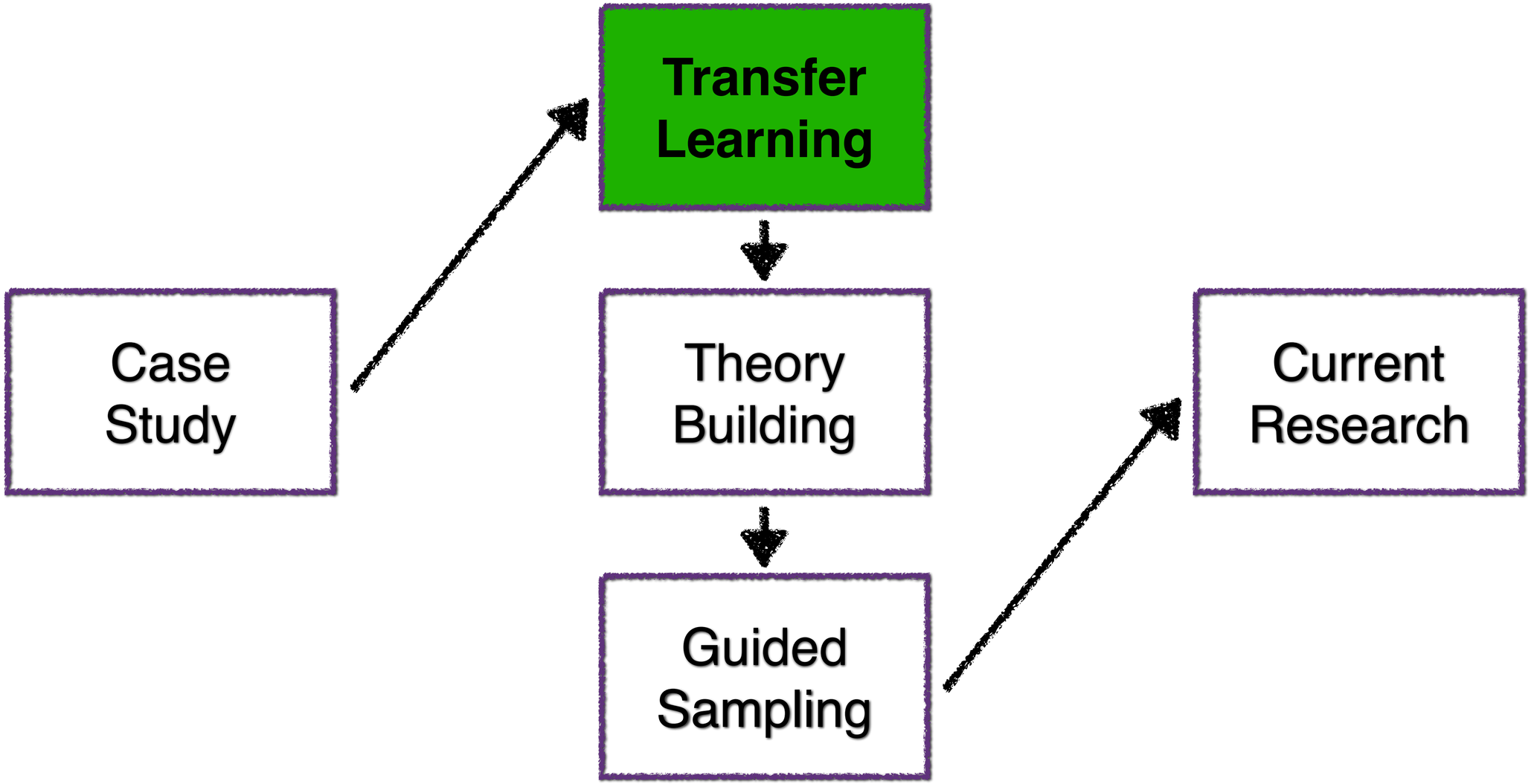
Large configuration space

- Exhaustive search is expensive
- Specific to hardware/workload/version

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

Outline


Transfer Learning
Case Study
Theory Building
Current Research
Guided Sampling

To enable performance tradeoff, we need a model 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)

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

$$f(o_1 := 1) = 8$$

$$f(o_1 := 0) = 5$$

What is a performance model?

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

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

$$c =< o_1, o_2 >$$

$$c =< o_1, o_2, ..., o_{10} >$$

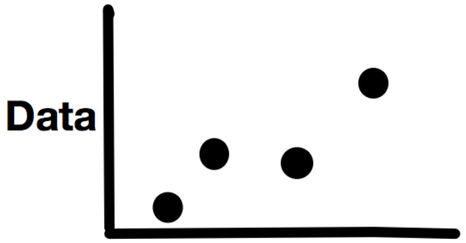
$$\vdots$$

$$c =< o_1, o_2, ..., o_{100} >$$

How do we learn performance models?

TurtleBot

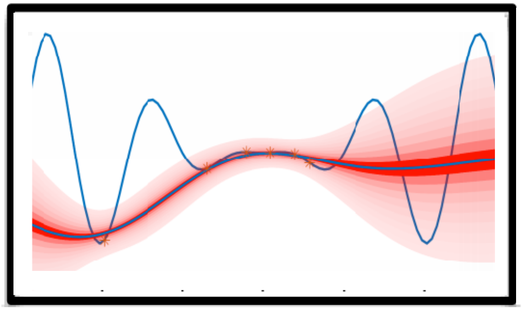
Measure



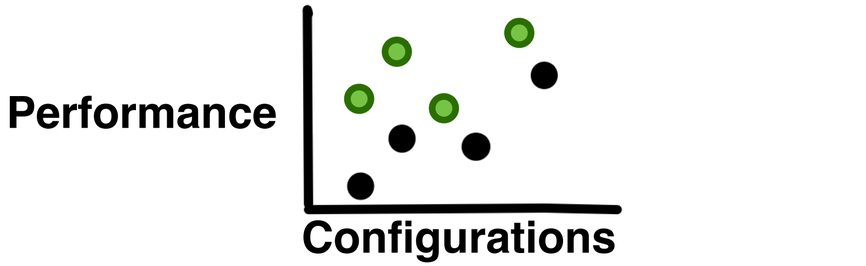
Learn

Optimization
Reasoning
Debugging

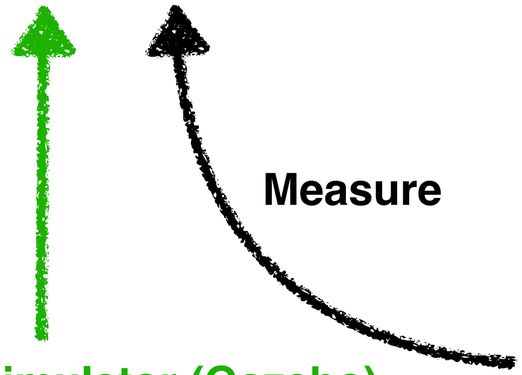
$$f(o_1, o_2) = 5 + 3o_1 + 15o_2 - 7o_1 \times o_2$$

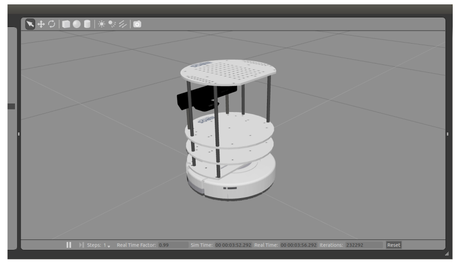


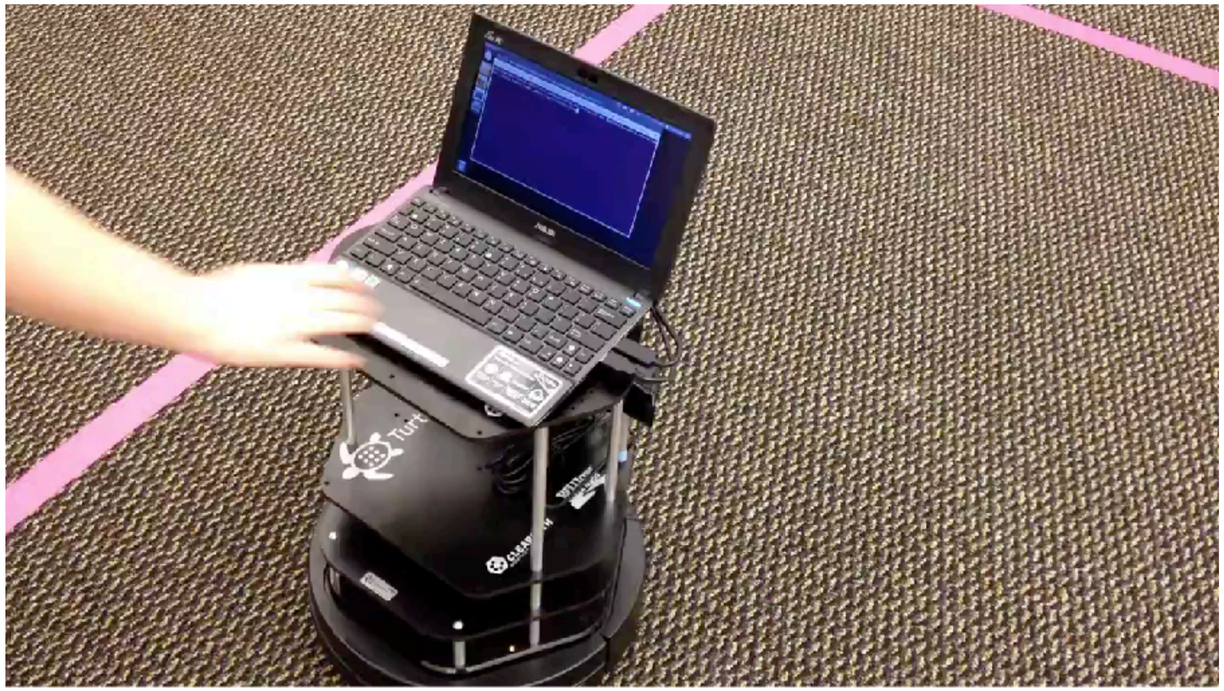
Insight: Performance measurements of the real system is “**similar**” to the ones from the simulators



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

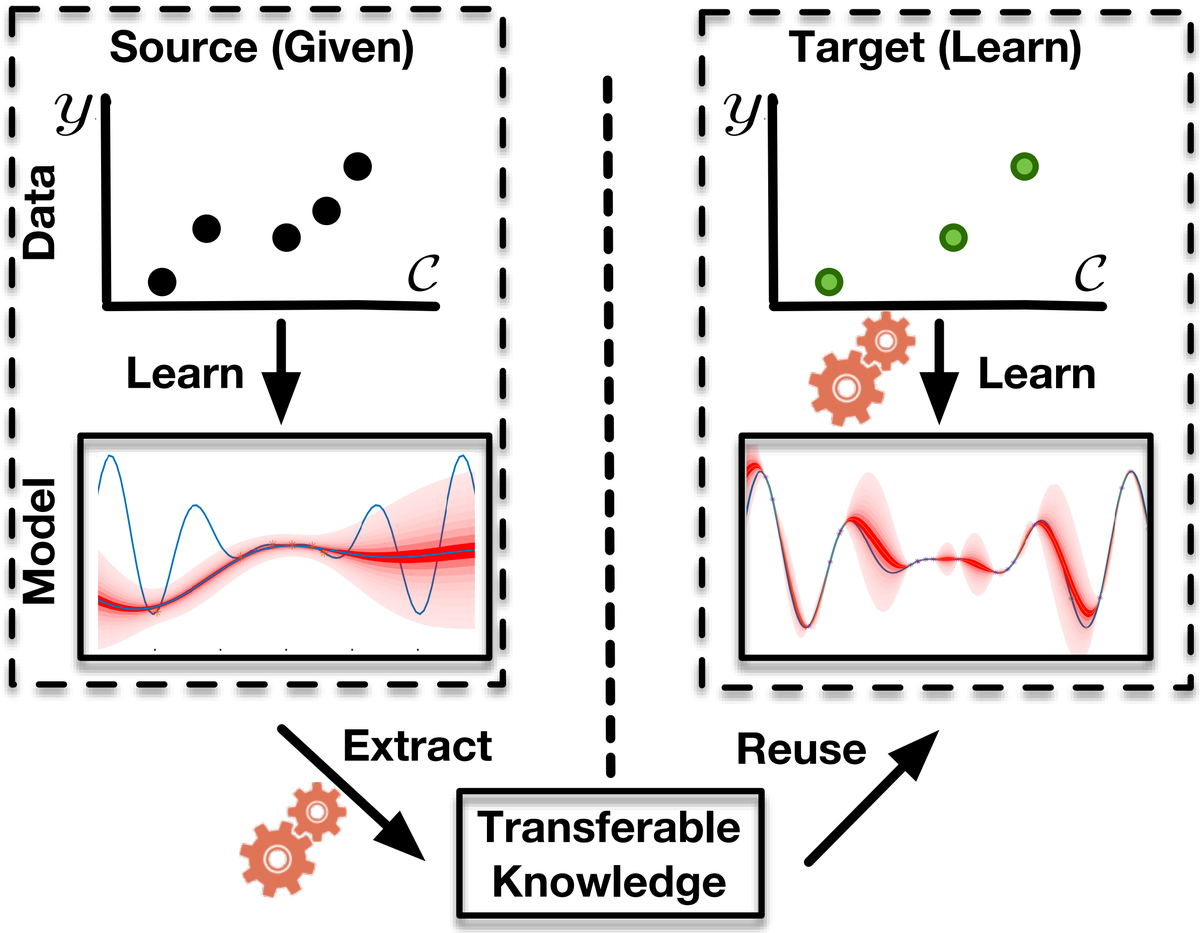






We developed methods to make learning cheaper via **transfer learning**

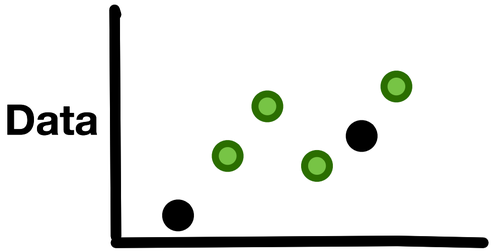
Goal: Gain strength by transferring information across environments



Our transfer learning solution

TurtleBot

Measure

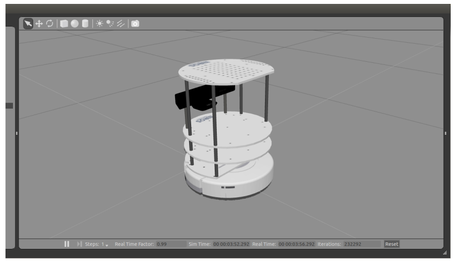


Reuse

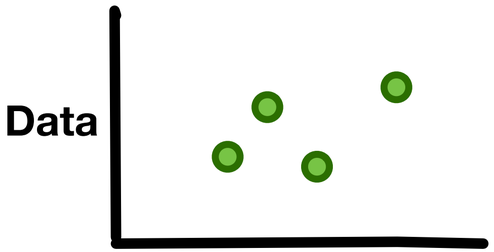
Learn

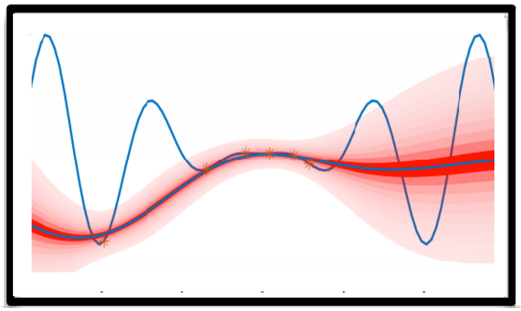
$$f(o_1, o_2) = 5 + 3o_1 + 15o_2 - 7o_1 \times o_2$$

Simulator (Gazebo)



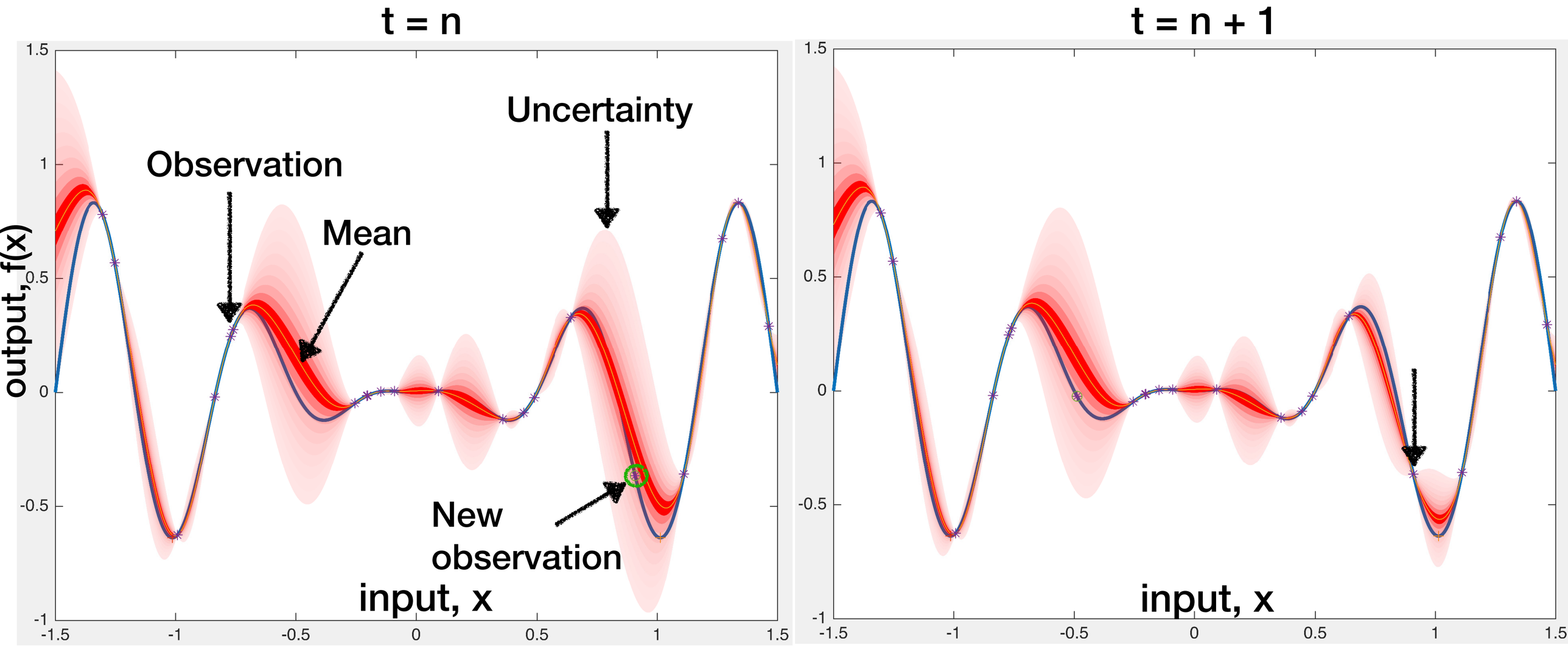
Measure





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

Gaussian processes for performance modeling



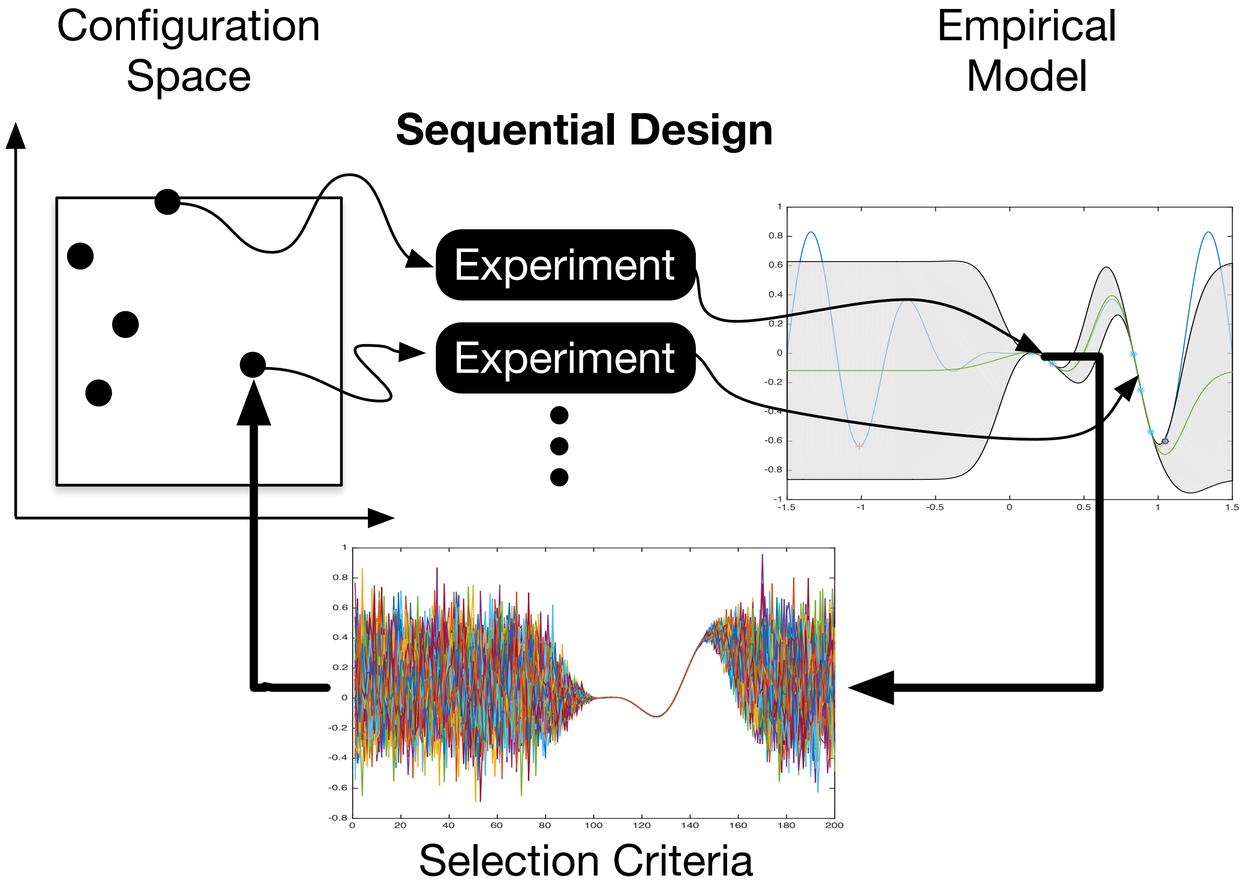
Gaussian Processes enables reasoning about performance

Step 1: Fit GP to the data seen so far

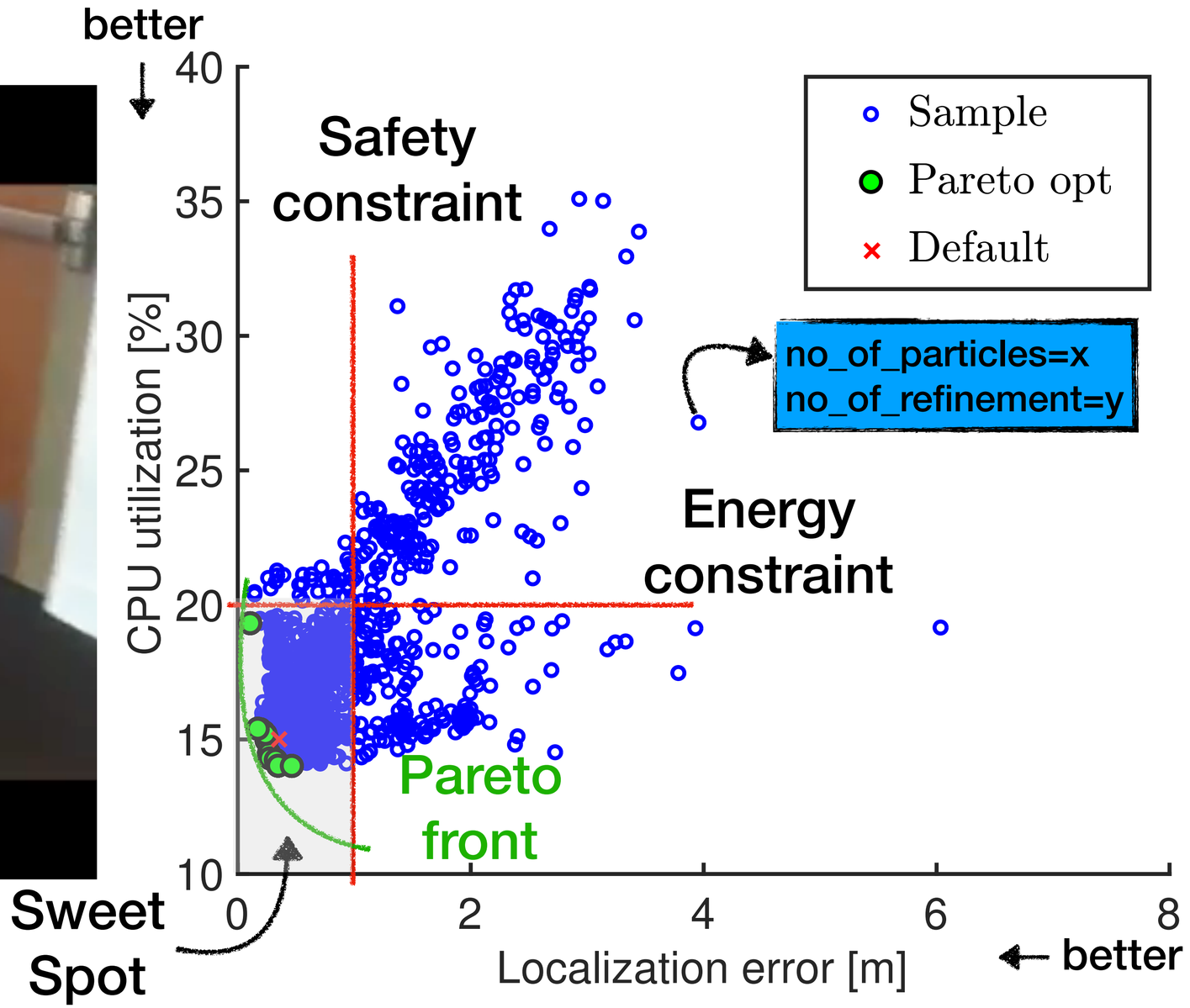
Step 2: Explore the model for regions of most variance

Step 3: Sample that region

Step 4: Repeat



CoBot experiment: DARPA BRASS



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:**

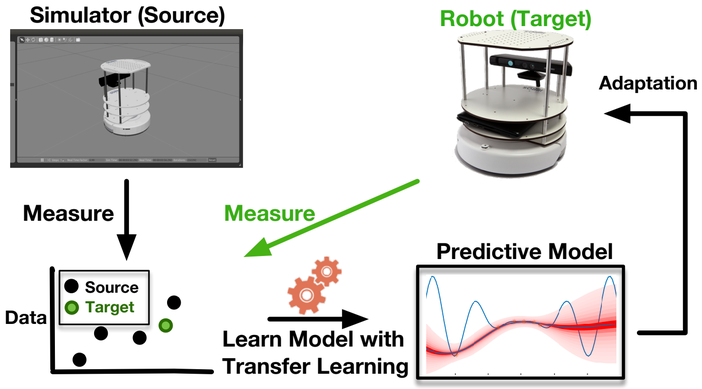
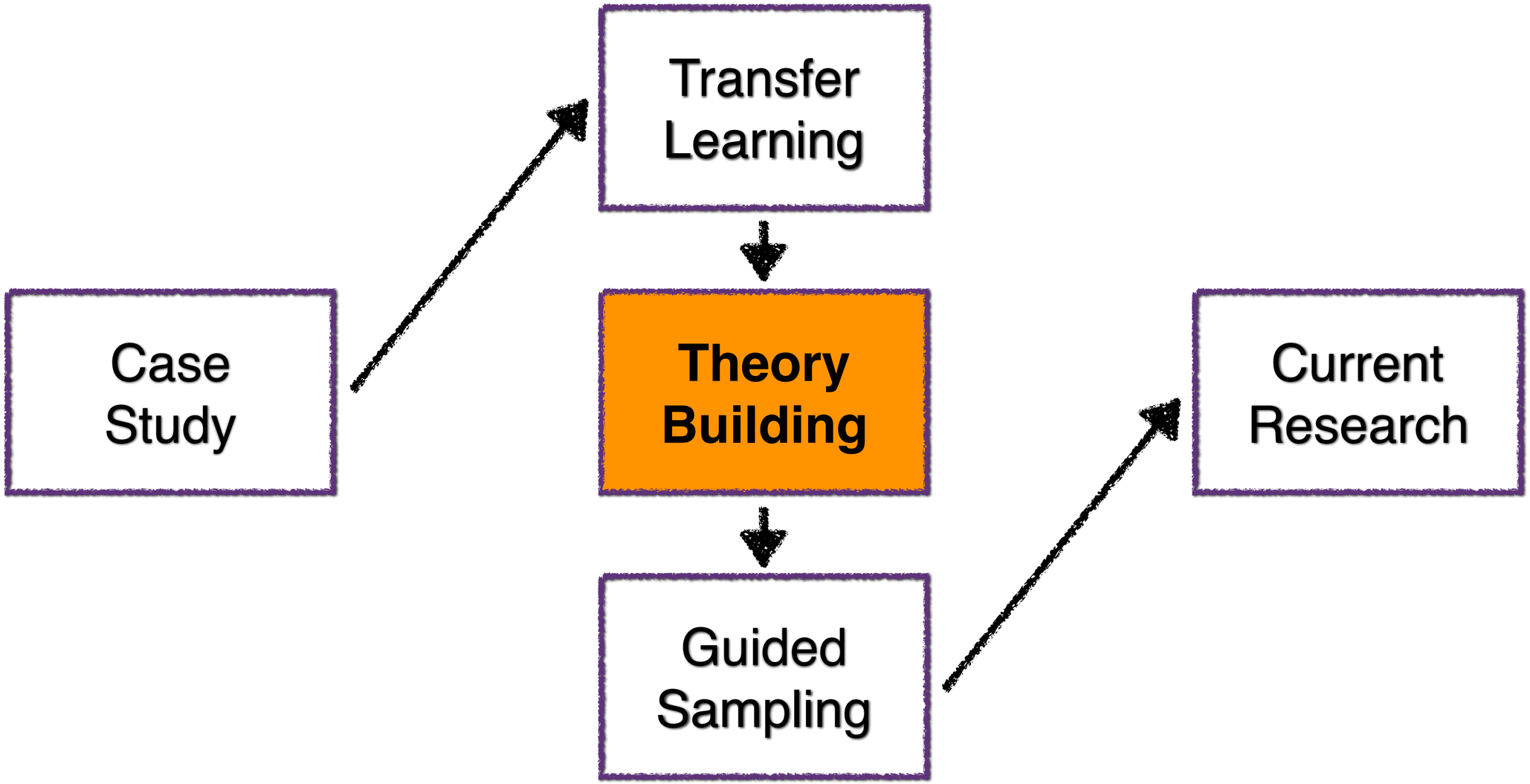


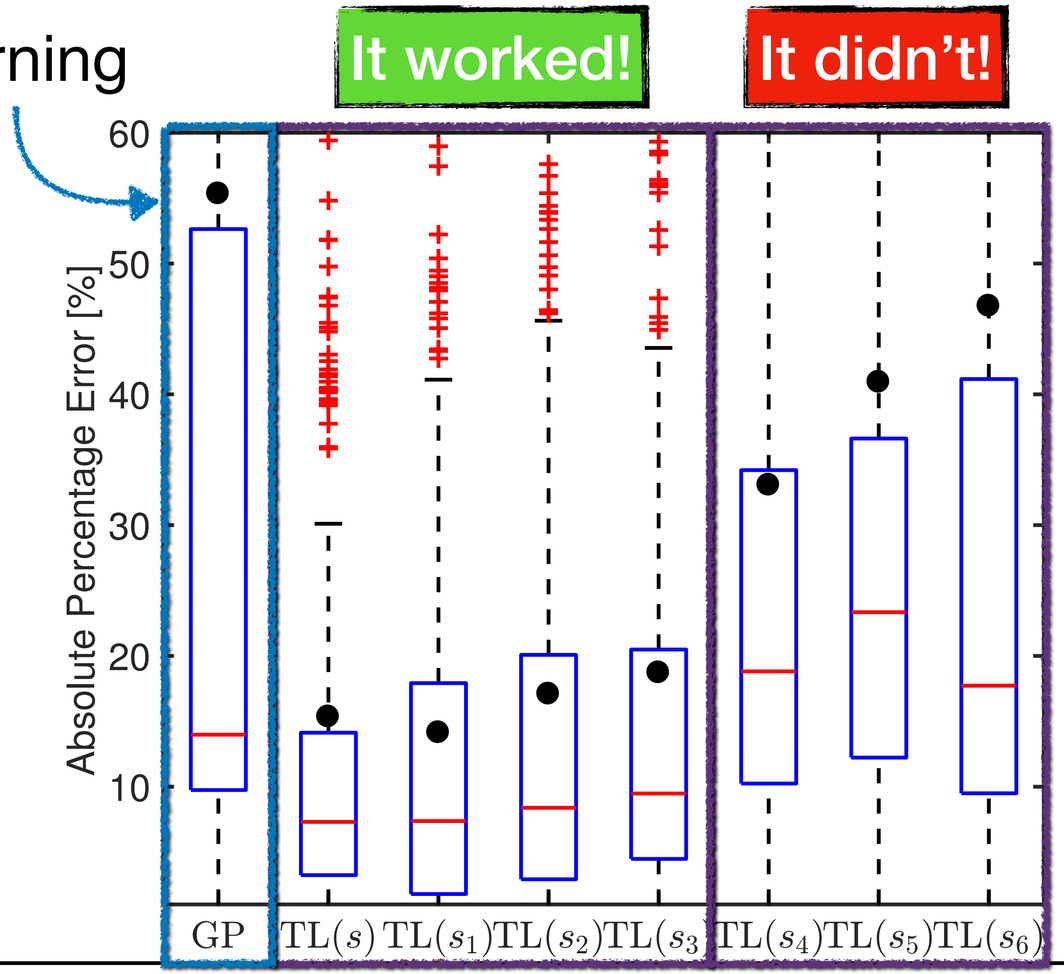
Fig. 1: Transfer learning for performance model learning.

Outline


Transfer Learning
Case Study
Theory Building
Current Research
Guided Sampling

Looking further: When transfer learning goes wrong

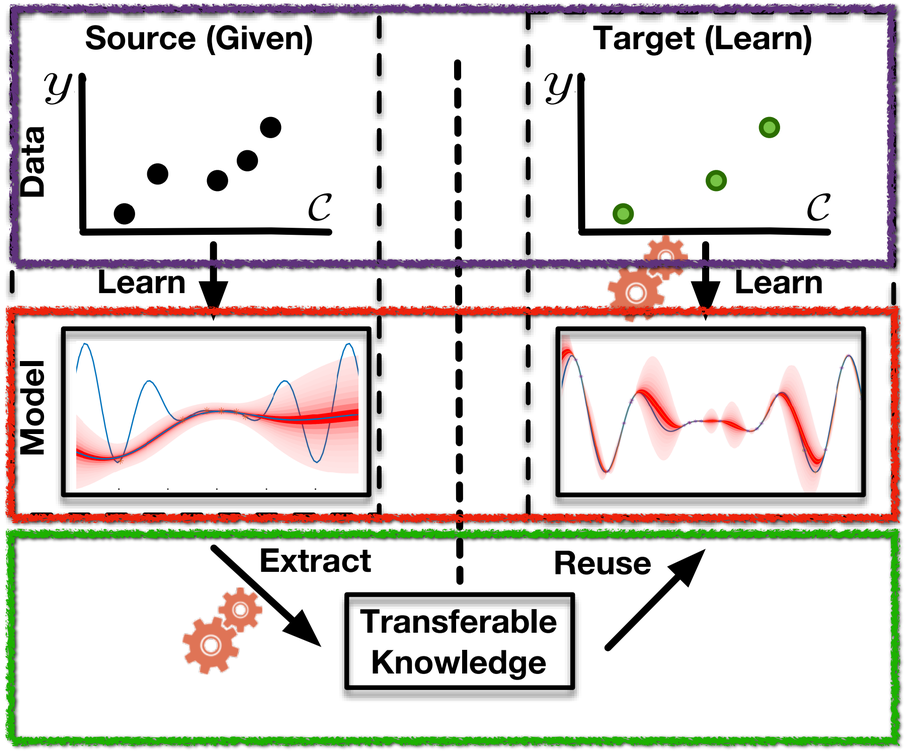
Non-transfer-learning



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

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



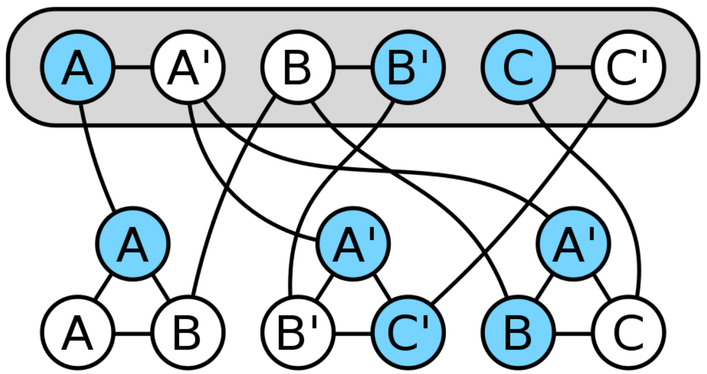
Q1: How source and target are “related”?

Q2: What characteristics are preserved?

Q3: What are the actionable insights?

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

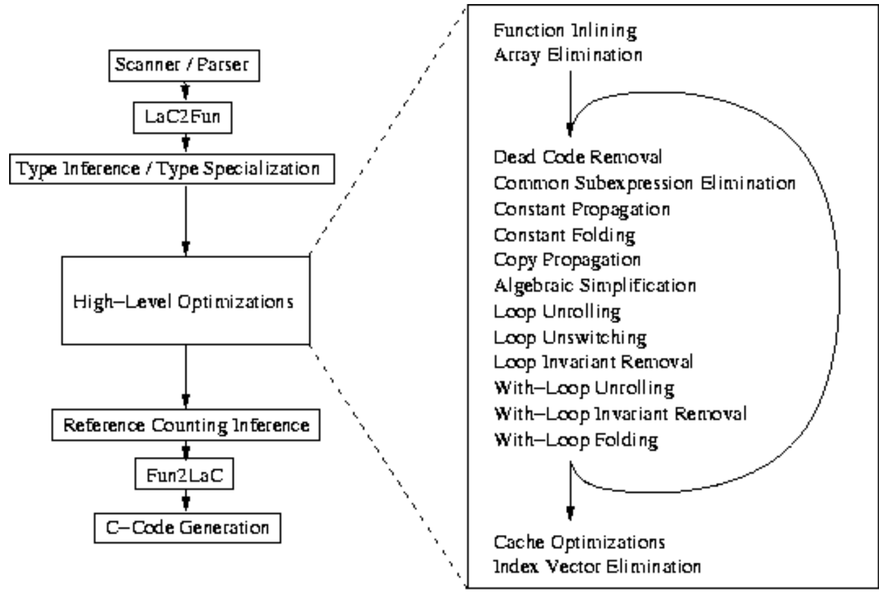
(A ∨ B) ∧ (¬A ∨ ¬B ∨ ¬C) ∧ (¬A ∨ B ∨ C)



SPEAR (SAT Solver)
Analysis time
14 options

X264 (video encoder)
Encoding time
16 options

SQLite (DB engine)
Query time
14 options

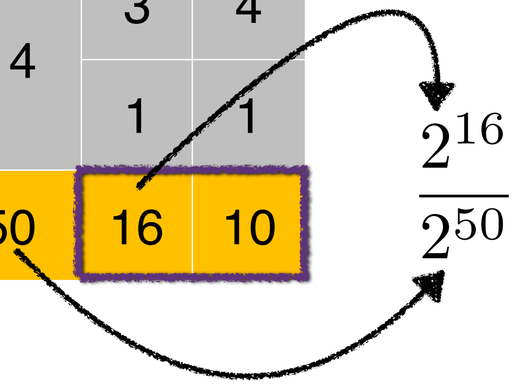


SaC (Compiler)
Execution time
50 options

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

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



$$\frac{2^{16}}{2^{50}} = 0.000000000058$$

We only need to explore part of the space:

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

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**

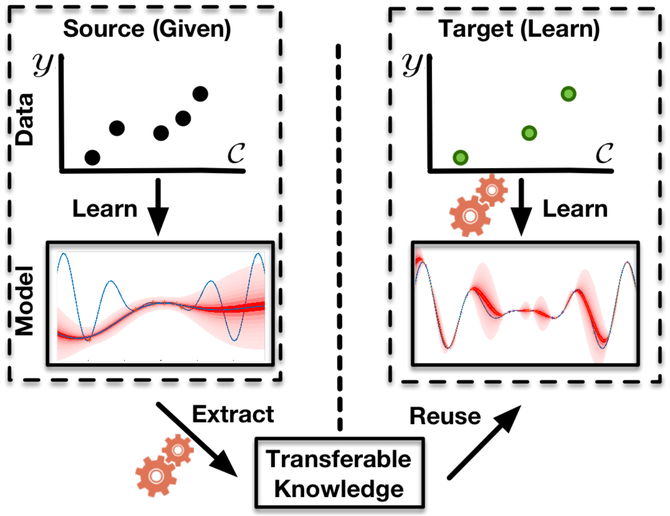
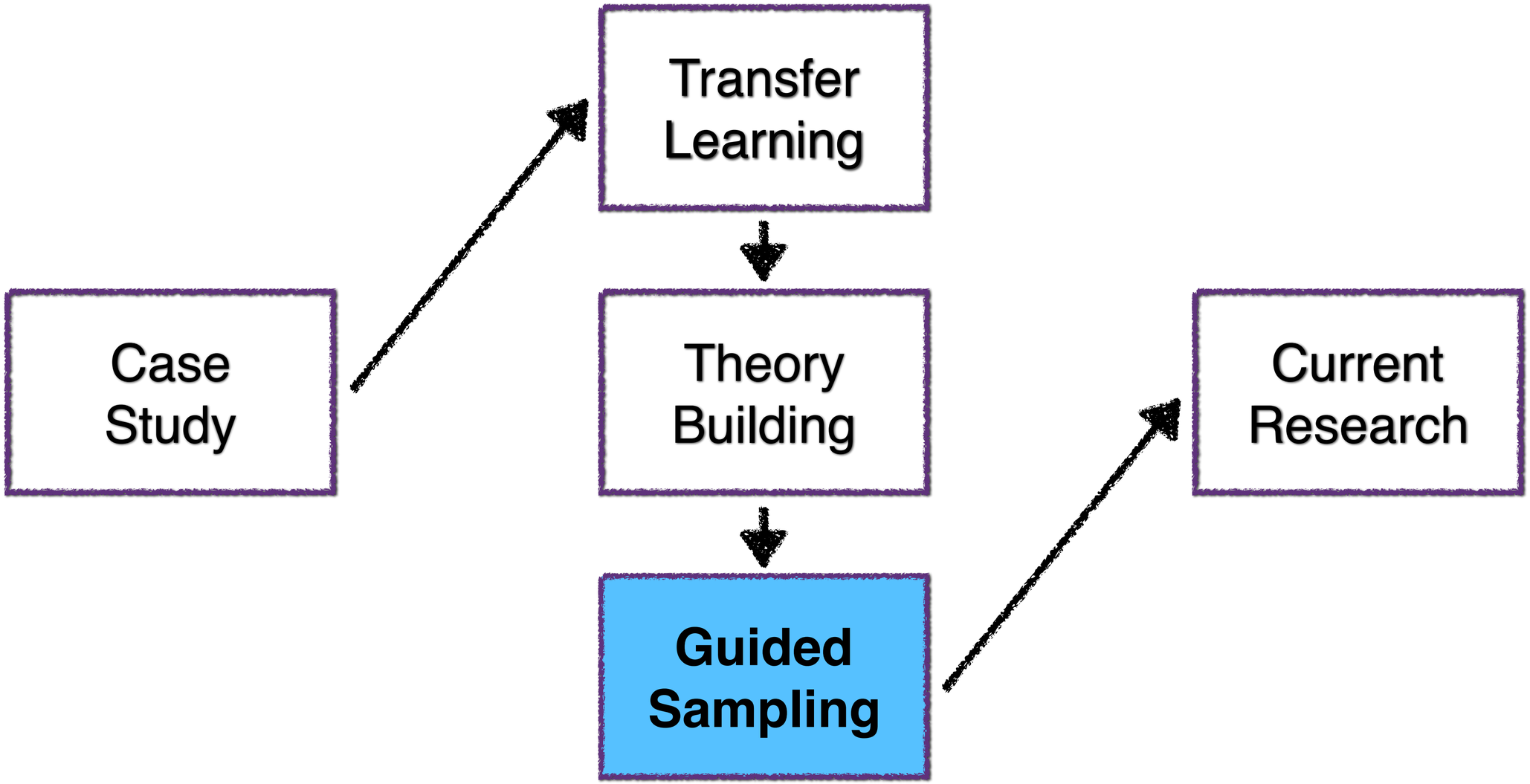


Fig. 1: Transfer learning is a form of machine learning that takes advantage of transferable knowledge from source to learn an accurate, reliable, and less costly model for the target environment.

Outline


Transfer Learning
Case Study
Theory Building
Current Research
Guided Sampling

Insights from our empirical study lead to the development of a guided sampling

TurtleBot

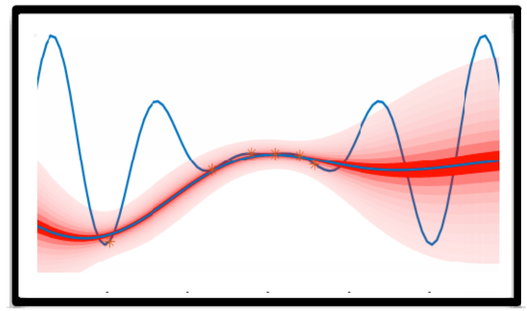
Measure

Data

Configurations

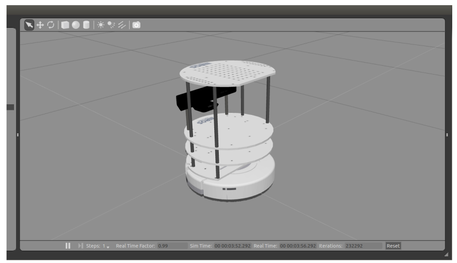
Learn

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



Reuse

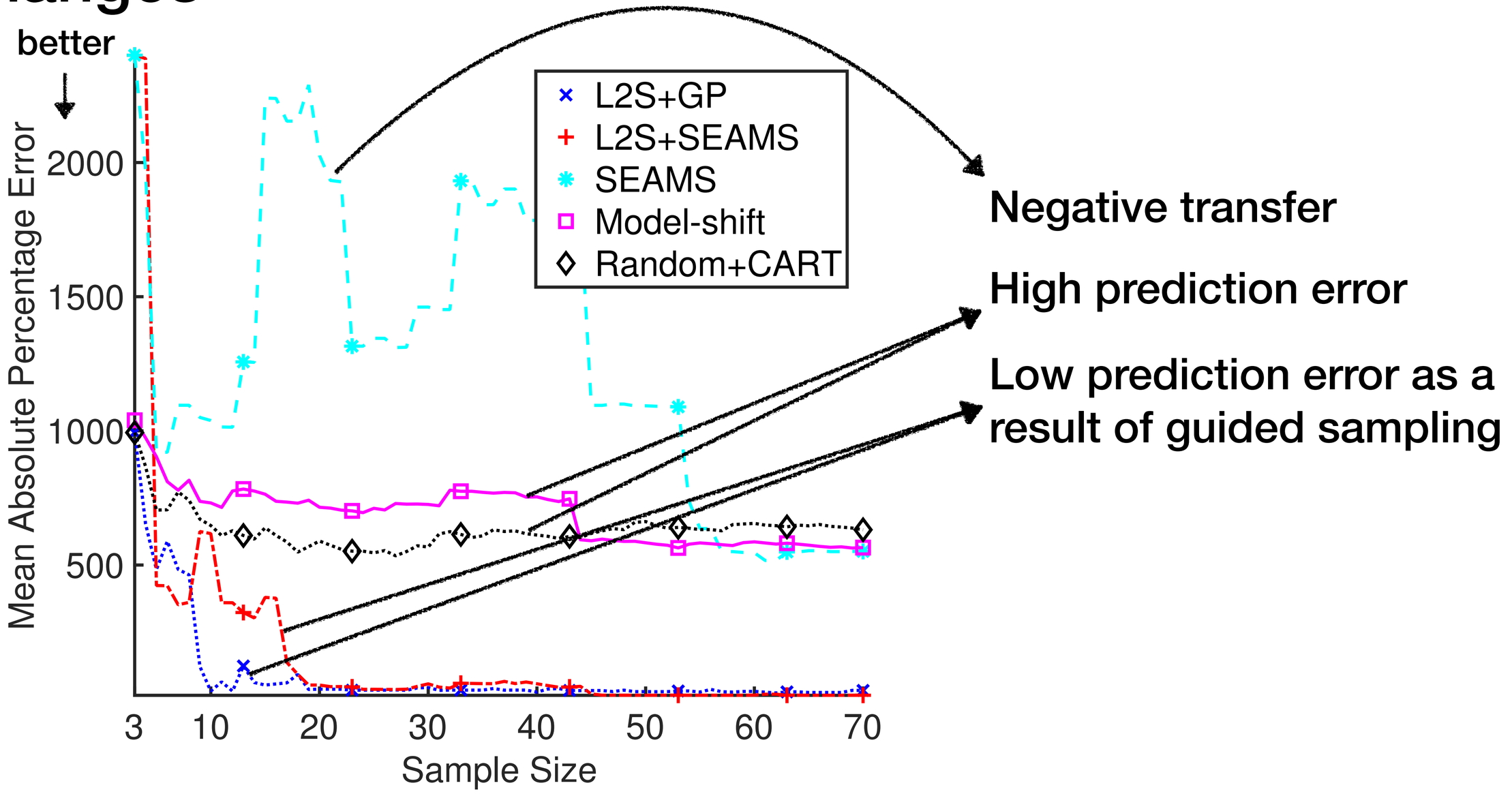
Simulator (Gazebo)



Measure

Data

Simple transfer learning do not work in severe changes



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-

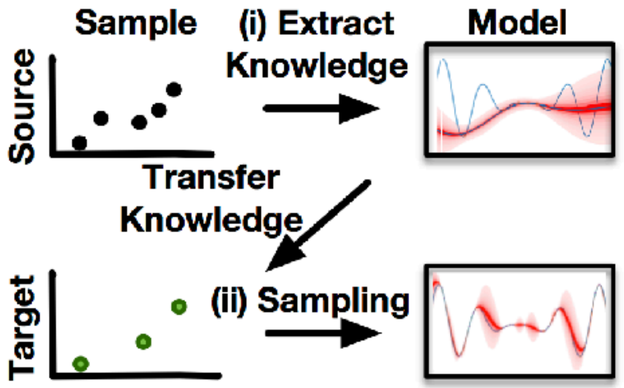
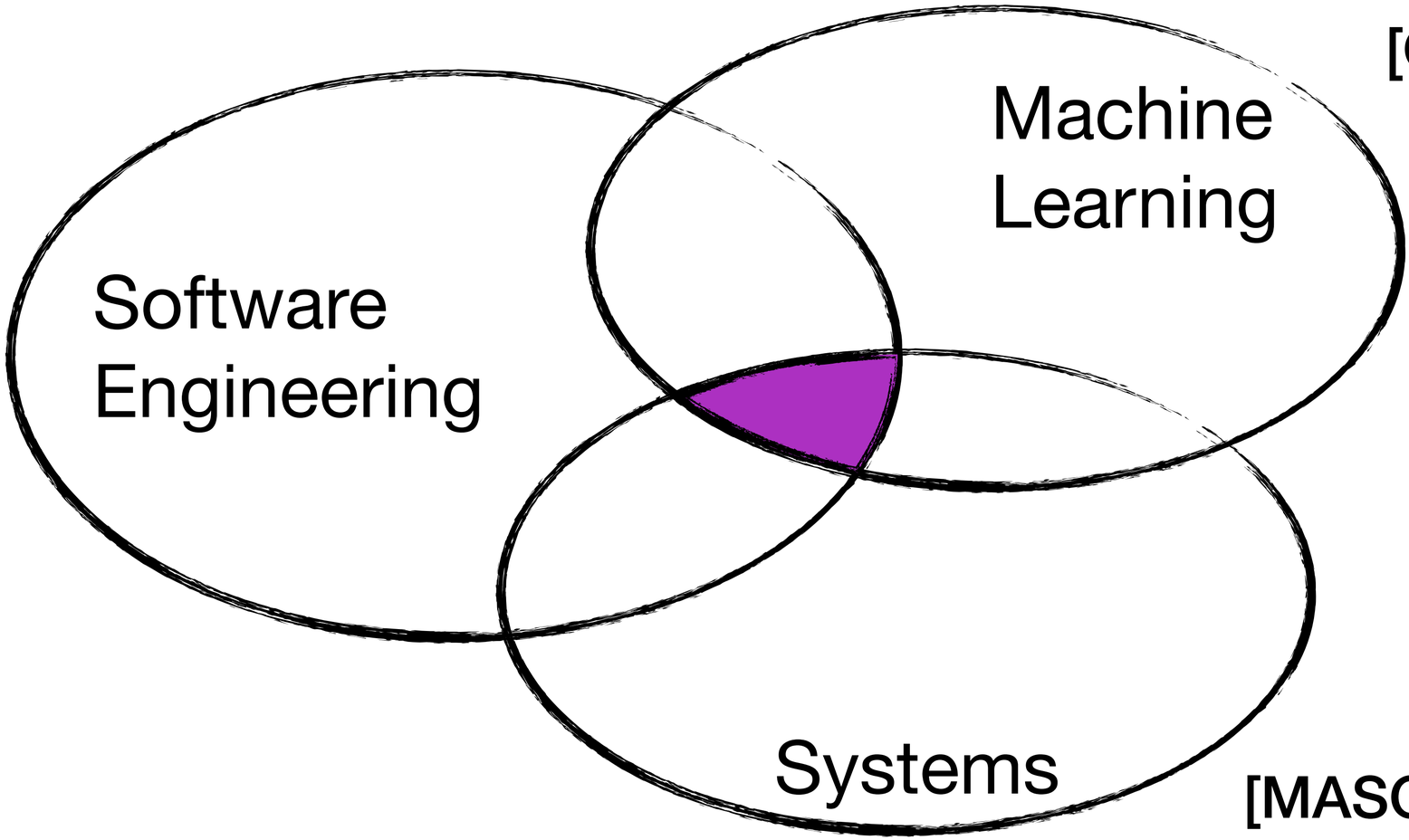


Figure 1: L2S performs guided sampling employing the knowledge extracted from a source environment.

Research interests

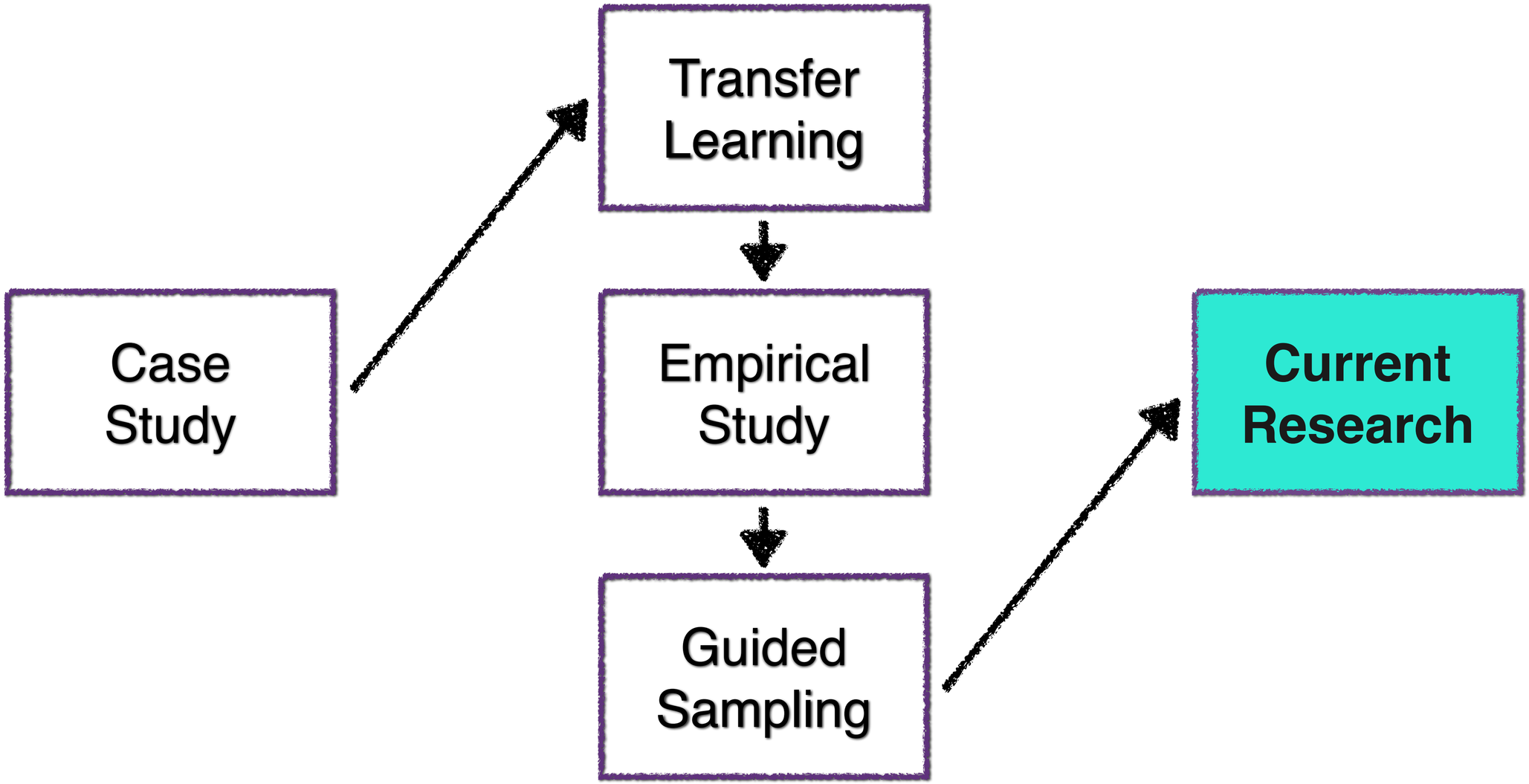
[QoSA ’16]
[CCGrid ’16]

[ASE ’17]
[SEAMS ’14]



[MASCOTS ’16]
[SEAMS ’17]

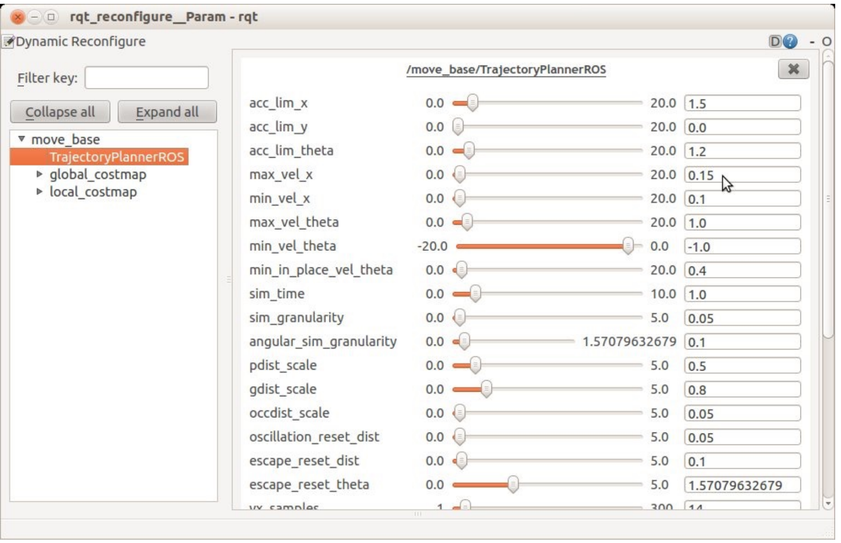
Outline

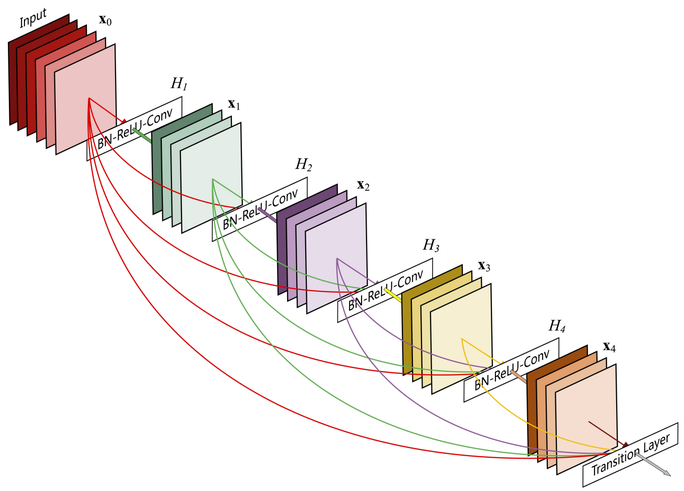

Transfer Learning
Case Study
Empirical Study
Current Research
Guided Sampling

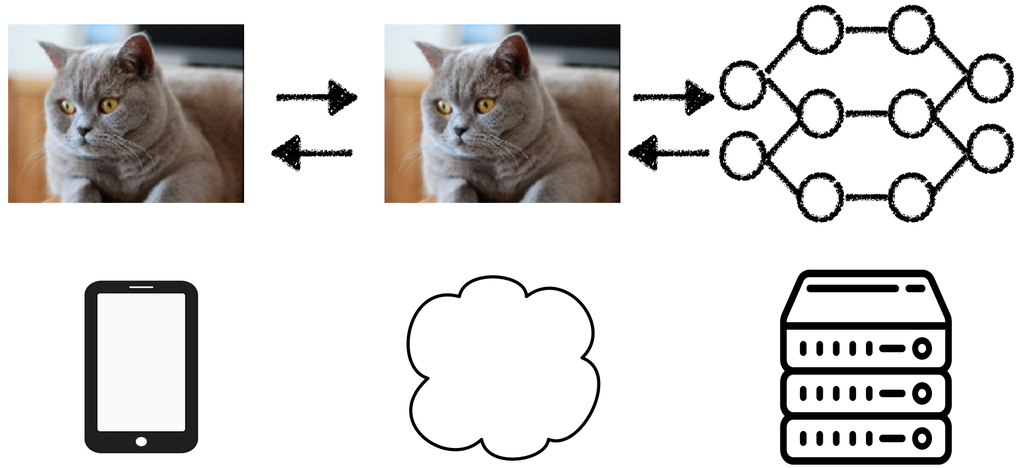

What will the software systems
of the future look like?

Software 2.0

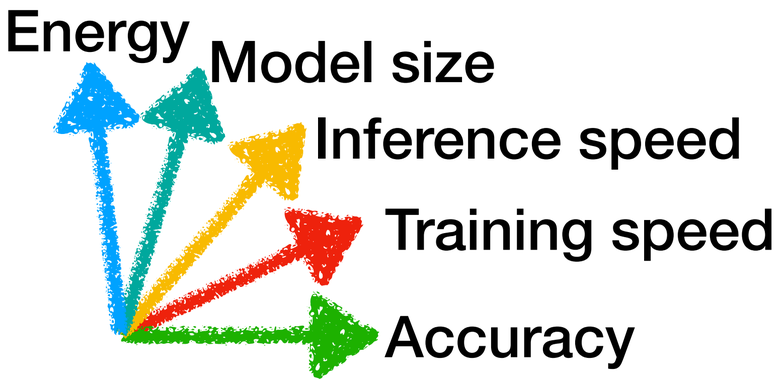
Increasingly **customized** and **configurable**



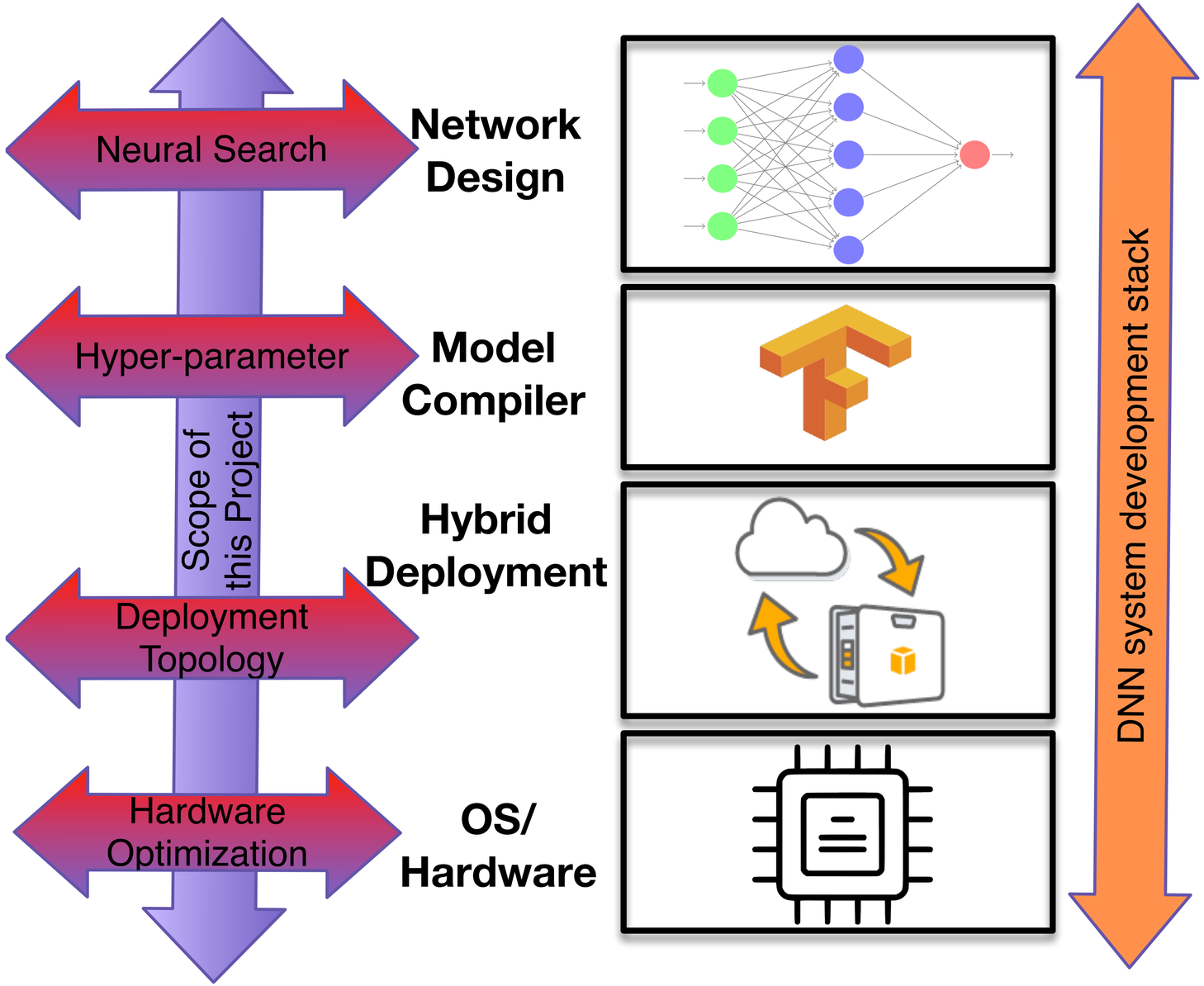




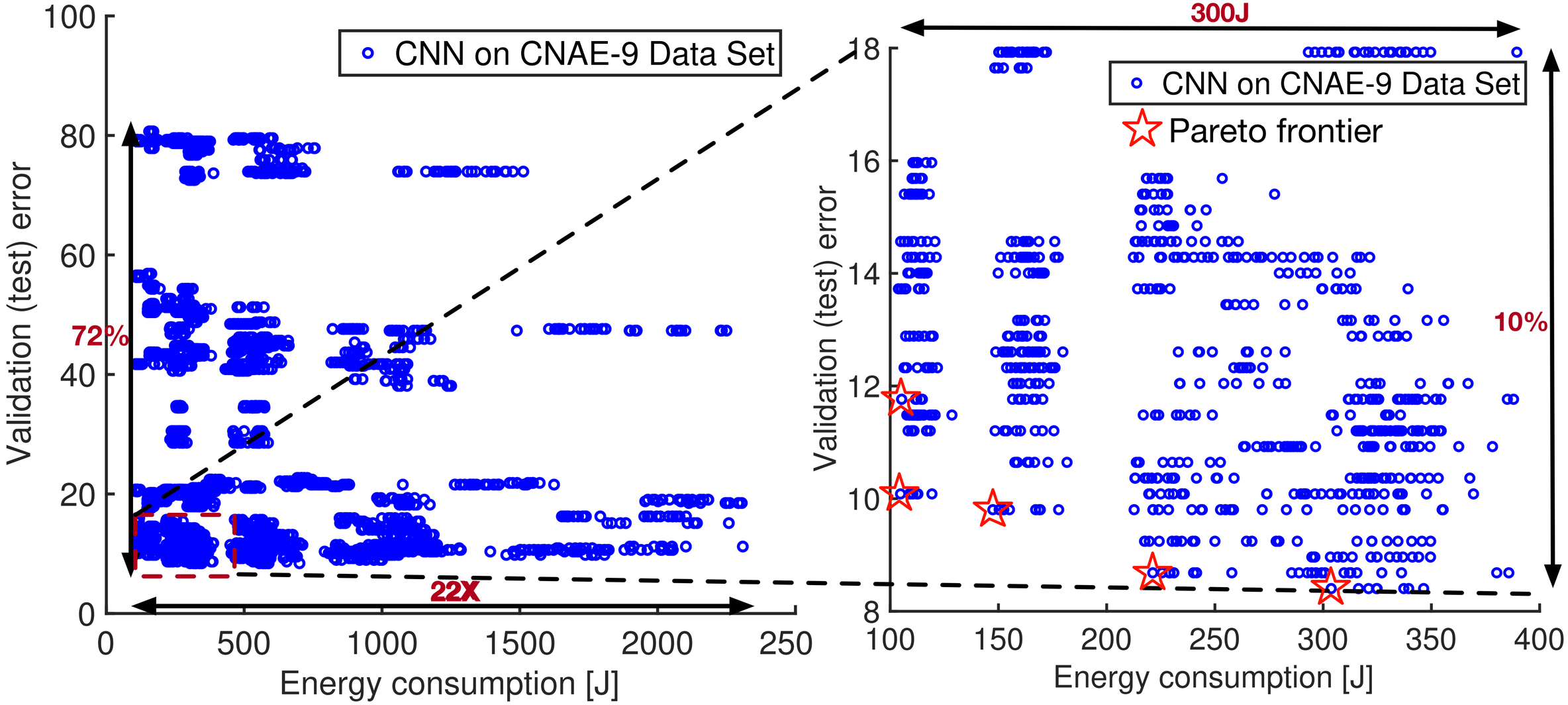
Increasingly **competing objectives**



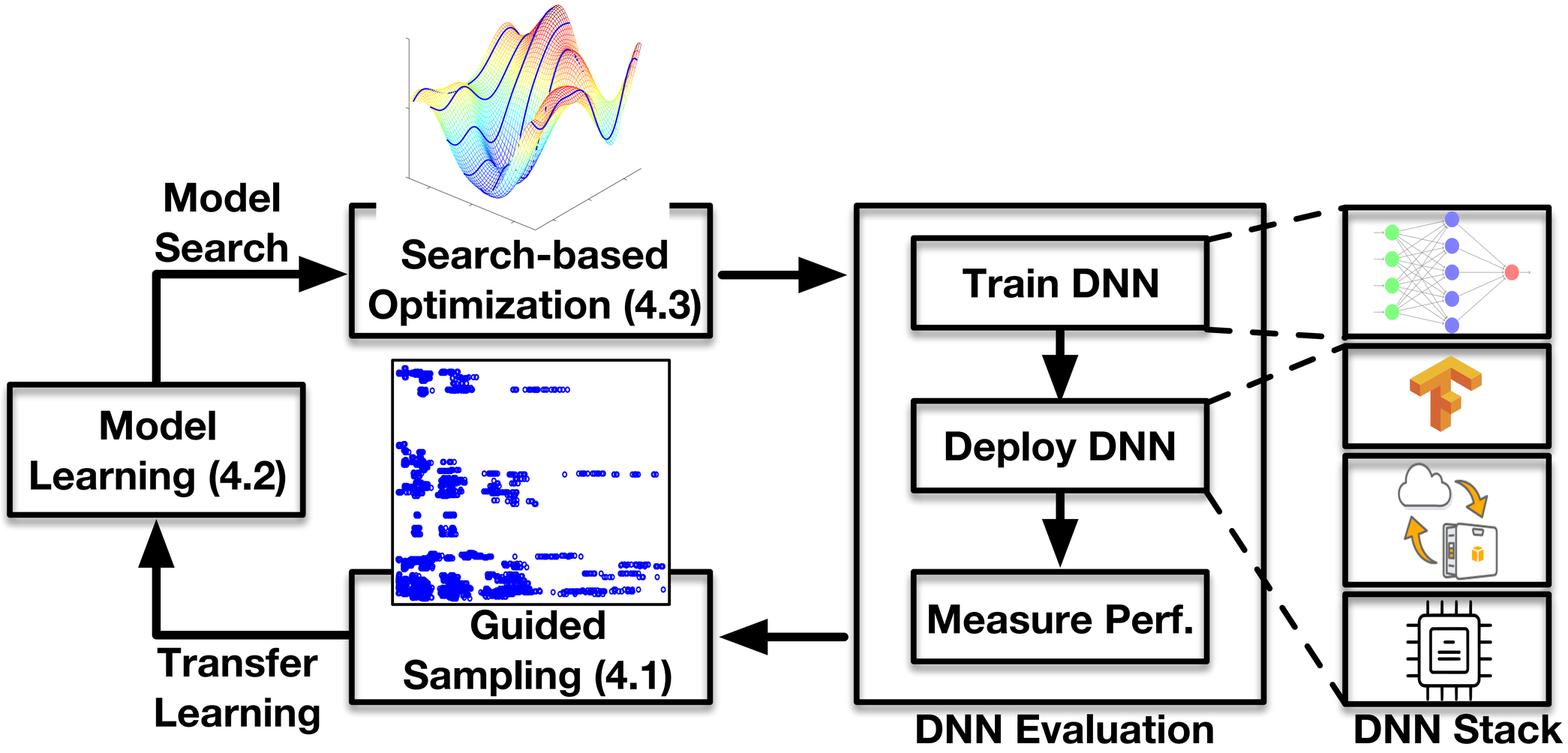
Deep neural network as a highly configurable system



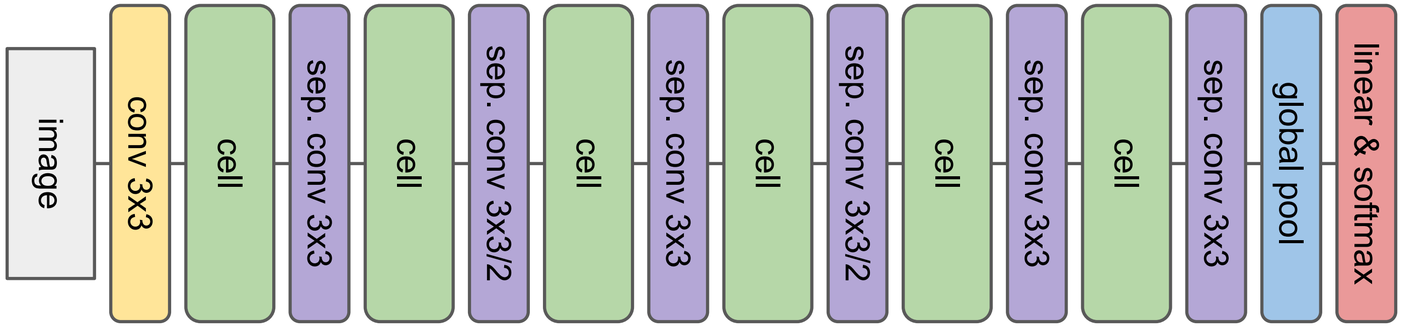
We found many configuration with the same accuracy while having drastically different energy demand



Optimizing Energy Consumption of Deep Neural Networks

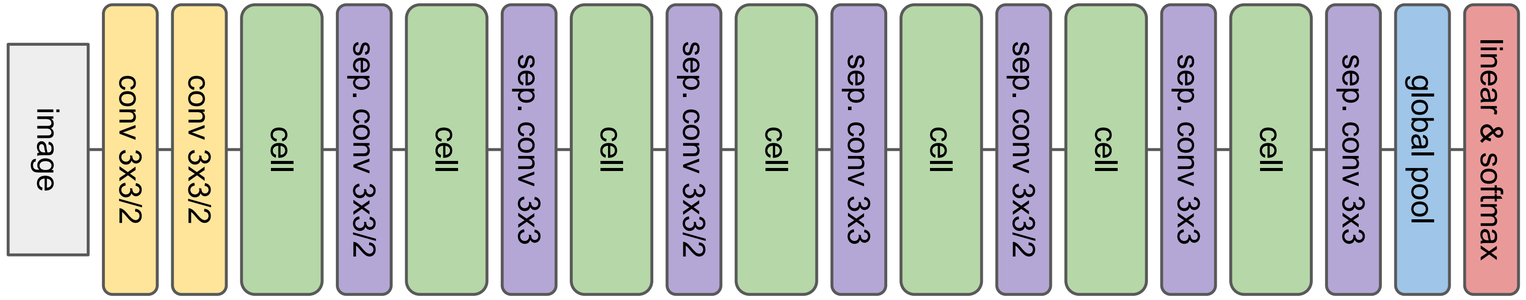


Exploring the design space of deep networks



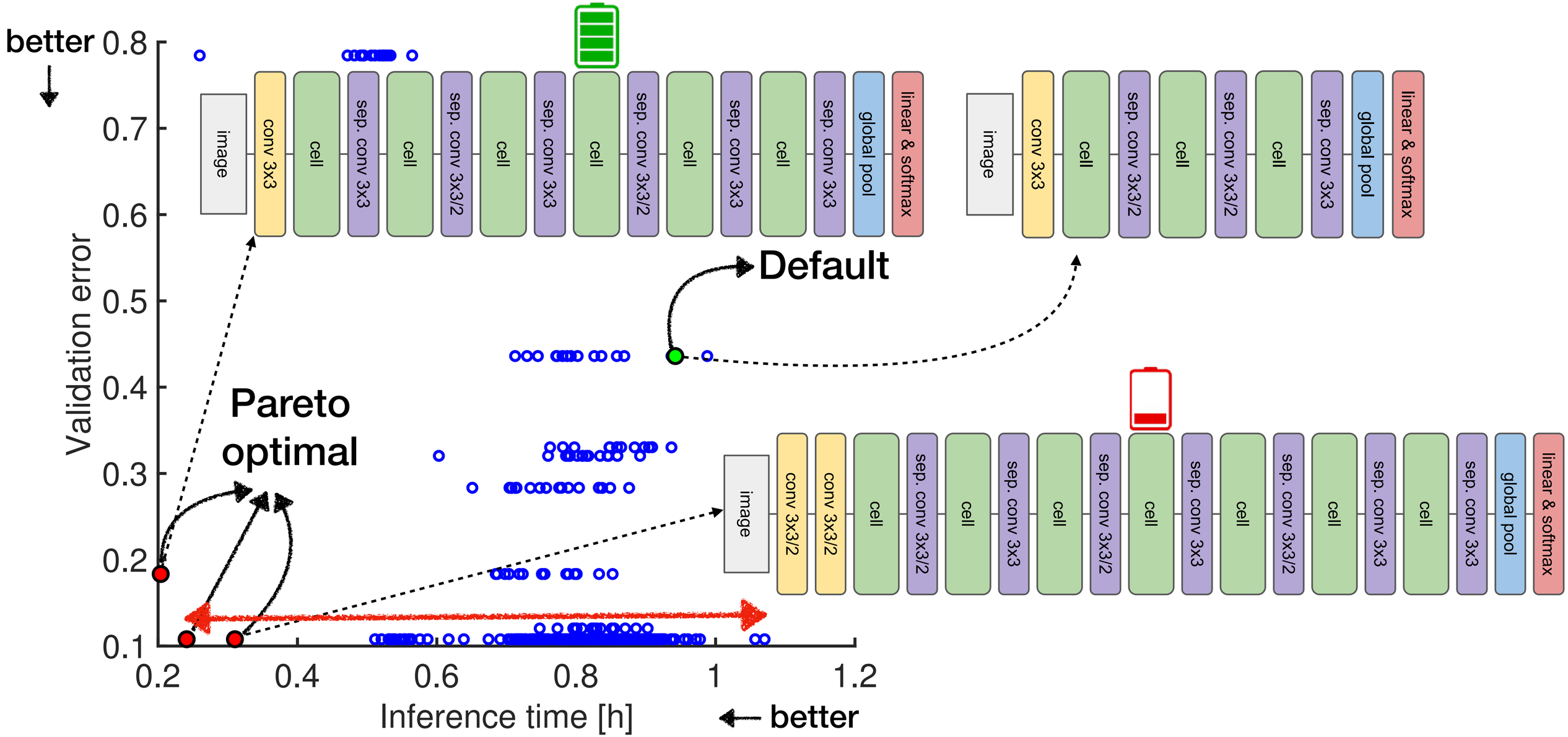
Optimal Architecture (Yesterday)

New Fraud Pattern

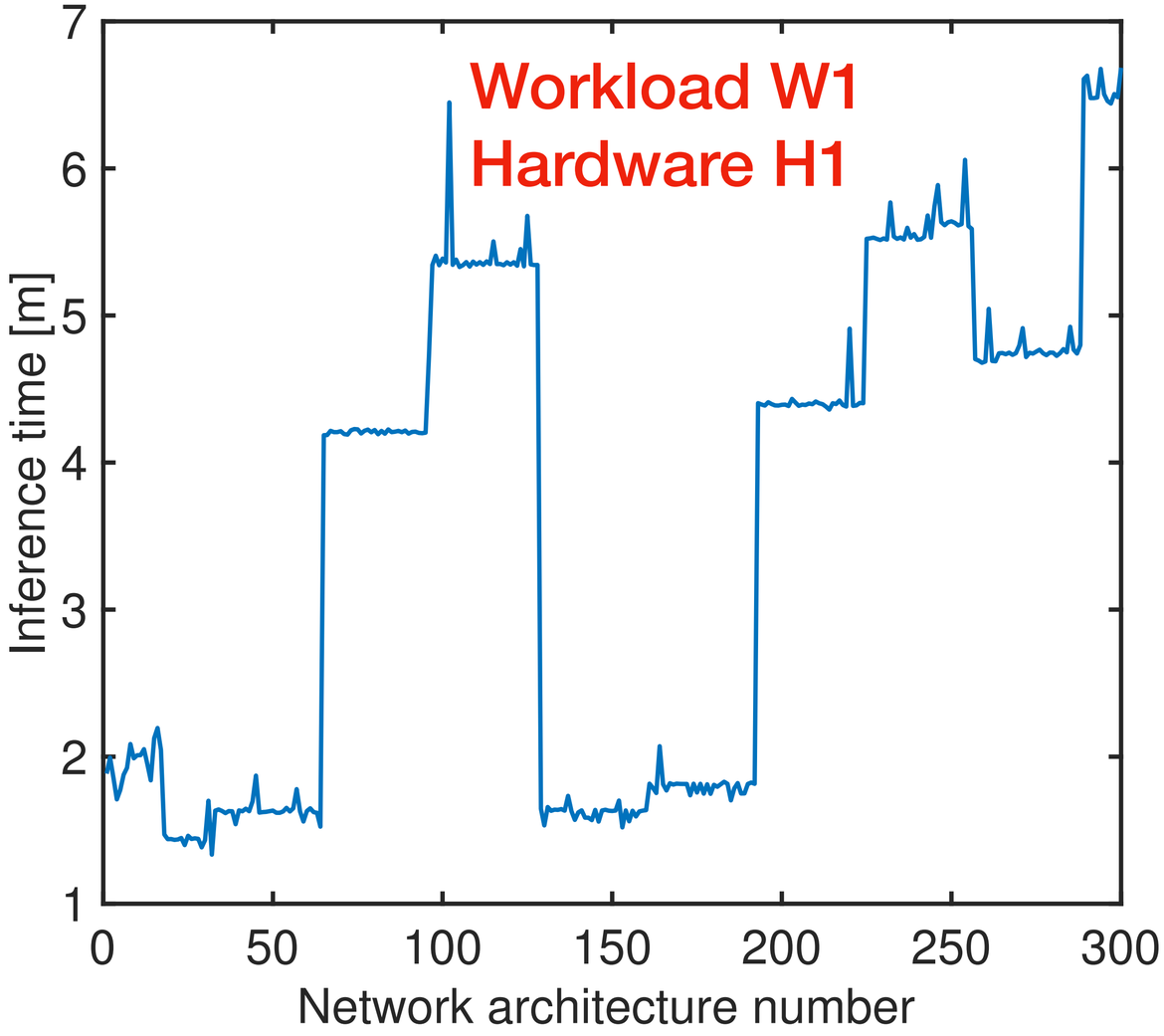


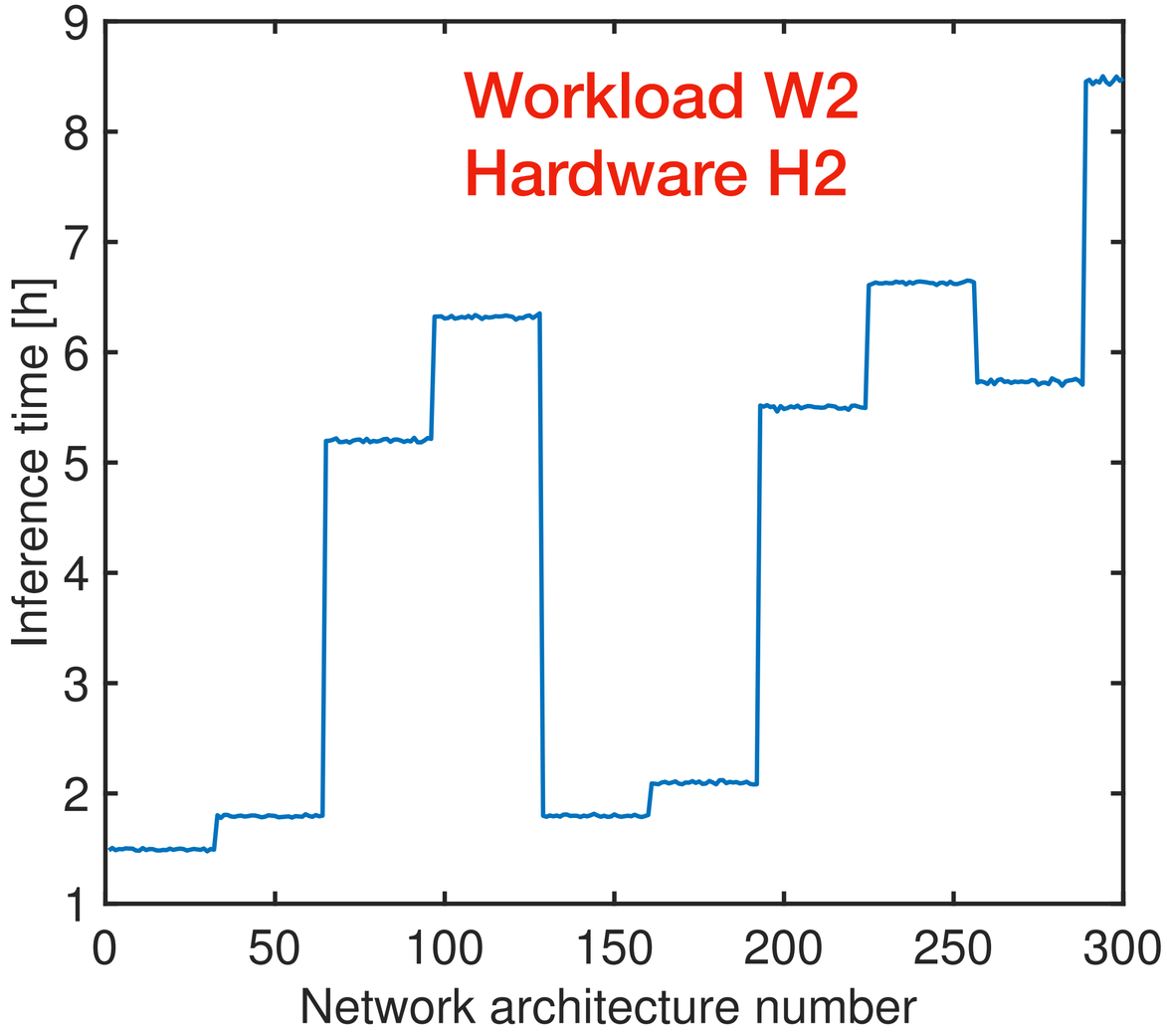
Optimal Architecture (Today)

Exploring the design space of deep networks



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



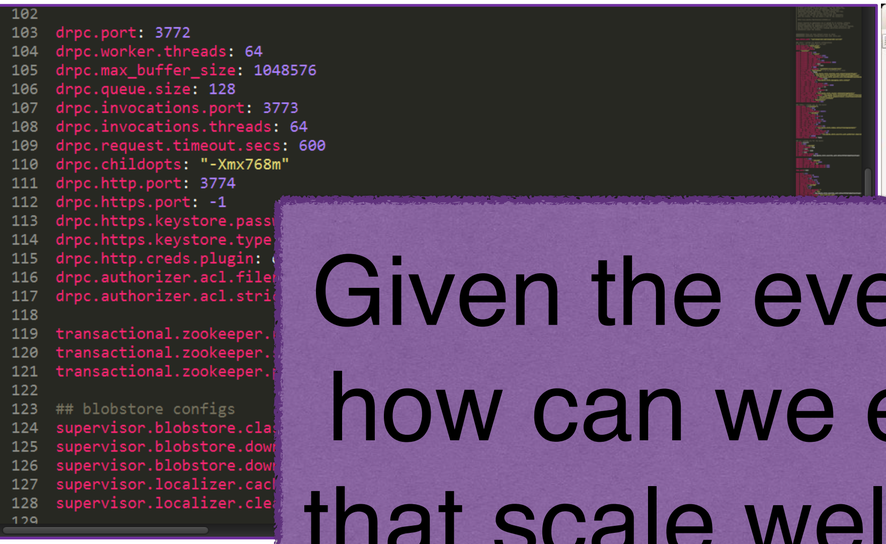


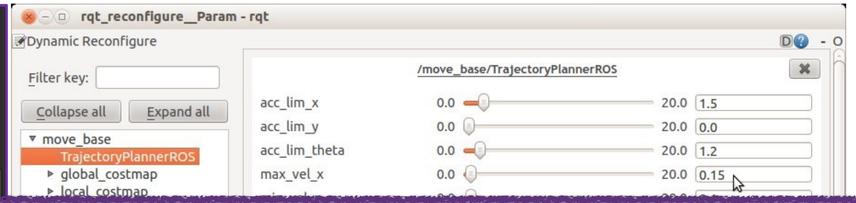
Interested?

Get in touch!

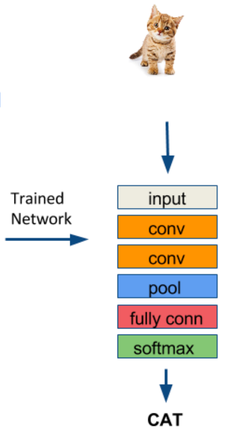
Many systems are now configurable

built

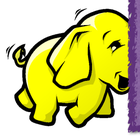


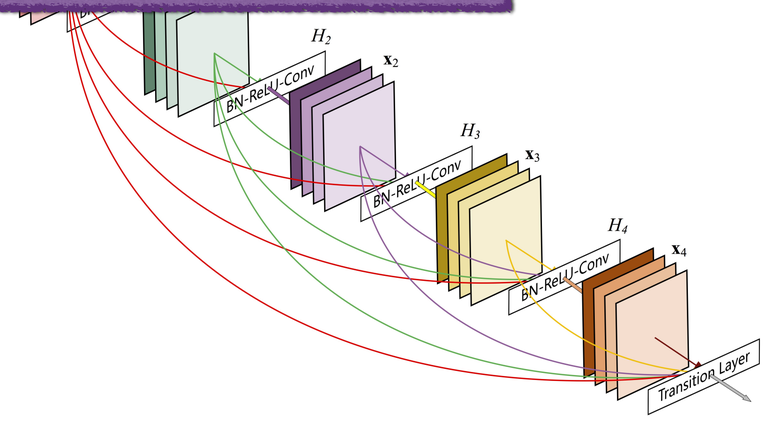


Train model on GPU accelerated cluster

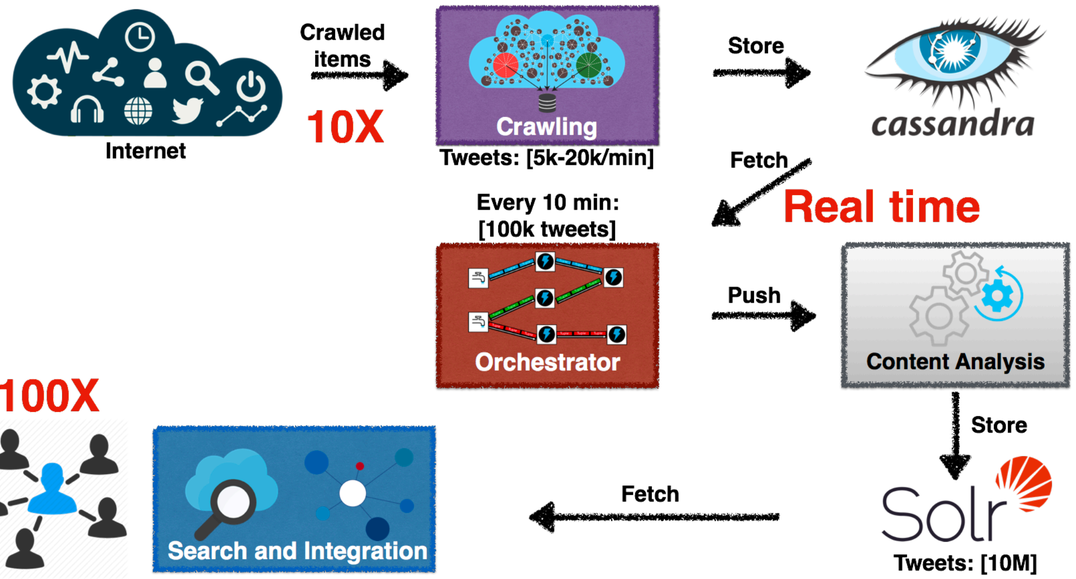


Given the ever growing configurable systems, how can we enable learning practical models that scale well and provide reliable predictions for exploring the configuration space?

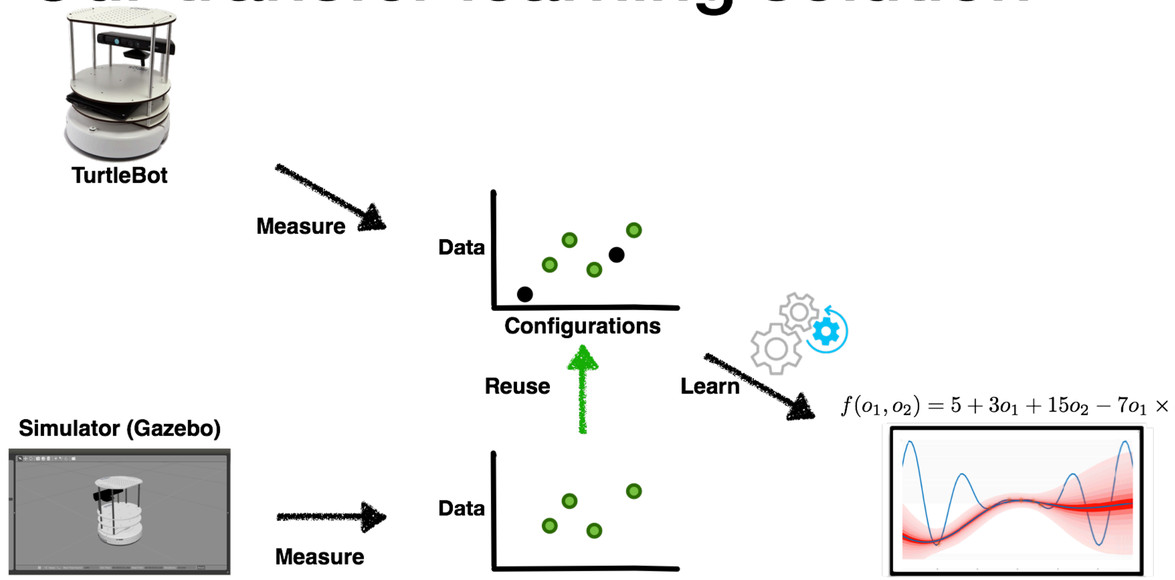




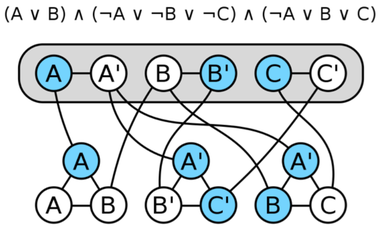
Challenges



Our transfer learning solution



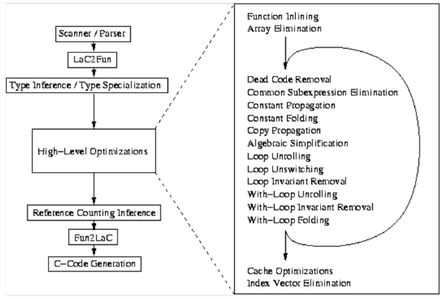
Our empirical study: We looked at different highly-configurable systems to gain insights



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