



AutoML Systems in Action

Xia “Ben” Hu

Associate Professor and Lynn '84 and Bill Crane '83 Faculty Fellow

Department of Computer Science and Engineering

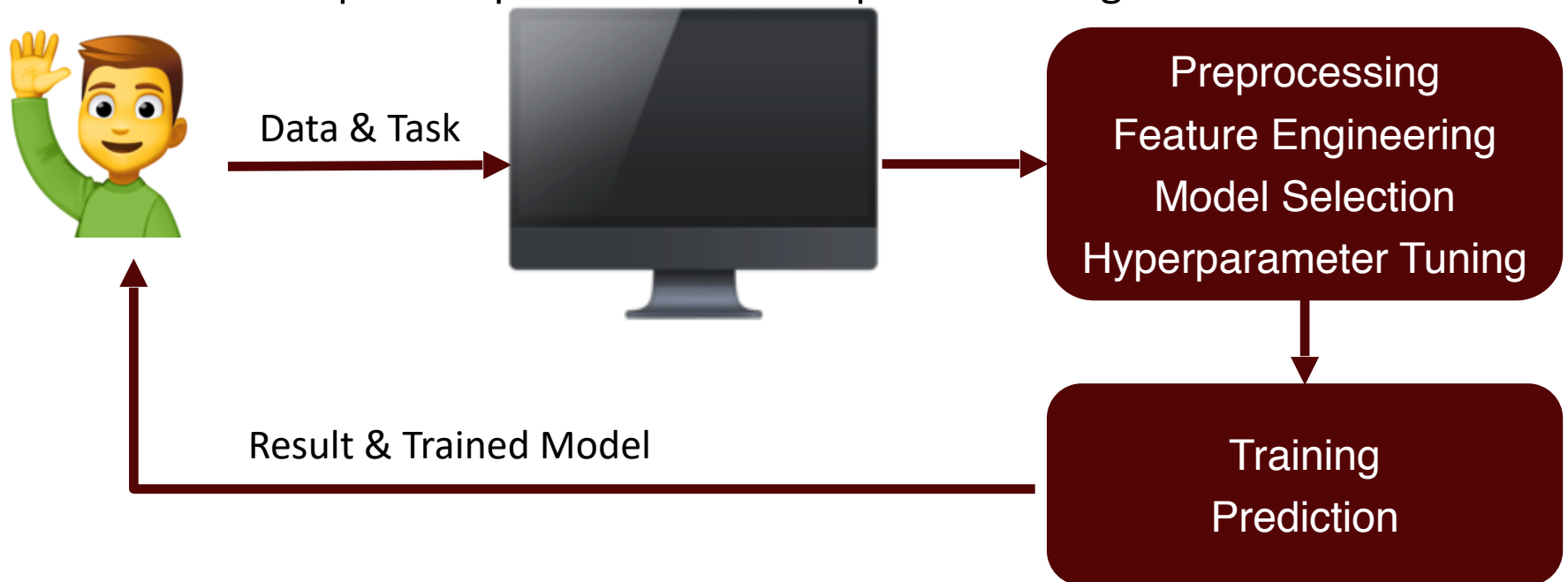
Texas A&M University

Email: xiahu@tamu.edu

What Is Automated Machine Learning (AutoML) ?

To make machine learning accessible

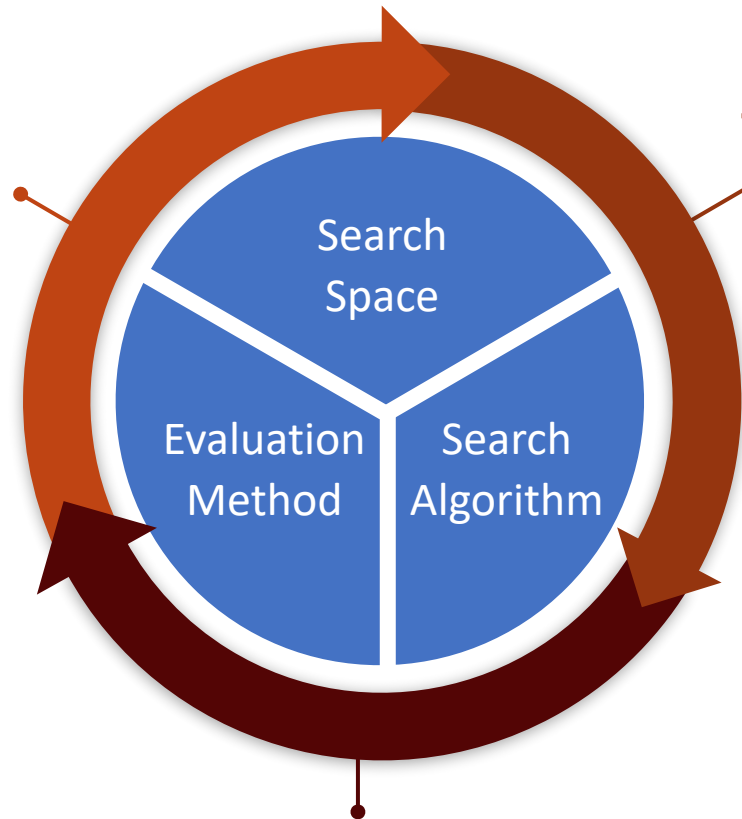
- To people with *limited ML background* such experts from other domains, as well as those *experienced data scientists*
- By automating the *end-to-end process from data to the results*
- Shown competitive performance on supervised image and text classifications



AutoML: System Components & Search Loop

3. *Update*

Update the surrogate model with training history (architectures and their performances)



1. *Generate*

Generate the next architecture for observation

2. *Observe*

Train the architecture and evaluate its performance

Image Reference: Yao, Quanming et al, "Efficient Neural Interaction Function Search for Collaborative Filtering", WWW 2020.

Road Map

1 AutoML System for Deep Learning

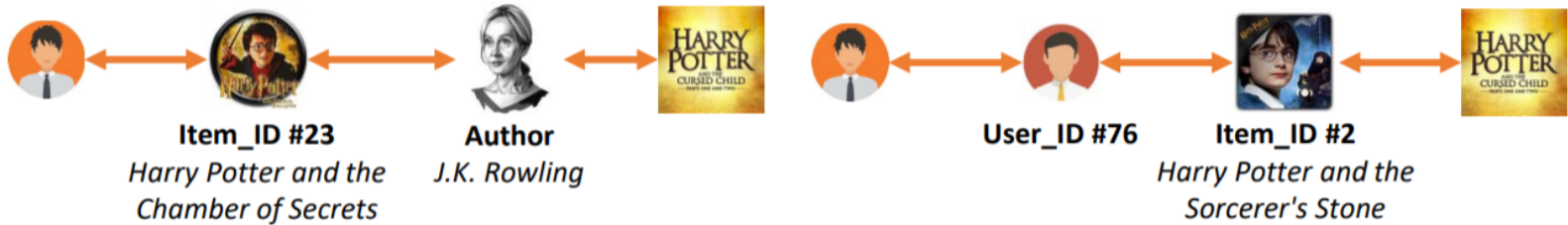


Tests passing

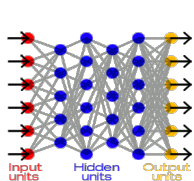
codecov 100%

pypi package 1.0.6

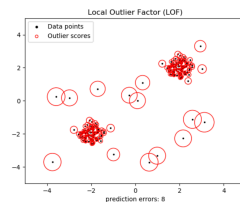
2 AutoML System for Recommendation



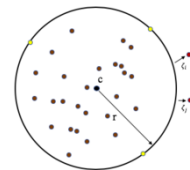
3 AutoML System for Outlier Detection



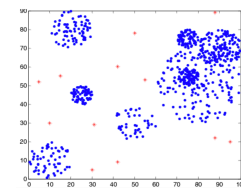
Reconstruction based



Density based



One-class based



Cluster based

Road Map

1 AutoML System for Deep Learning

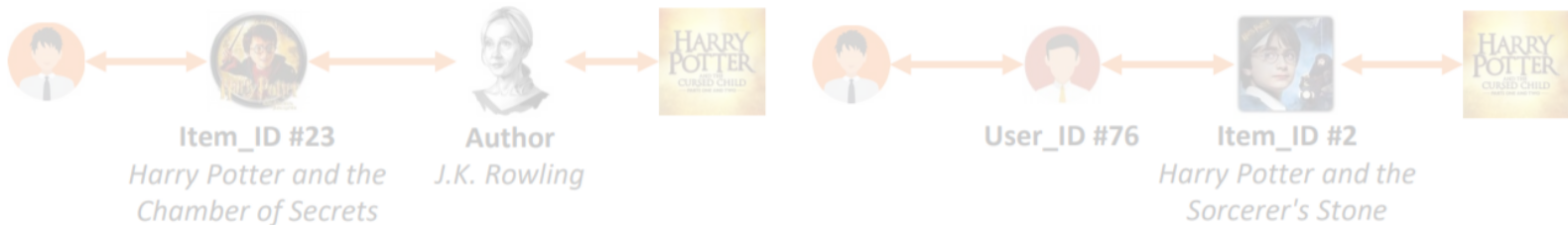


Tests passing

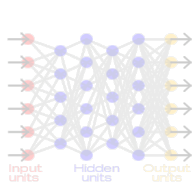
codecov 100%

pypi package 1.0.6

2 AutoML System for Recommendation



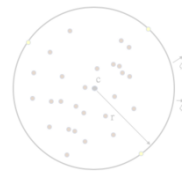
3 AutoML System for Outlier Detection



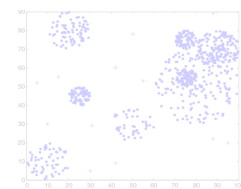
Reconstruction based



Density based

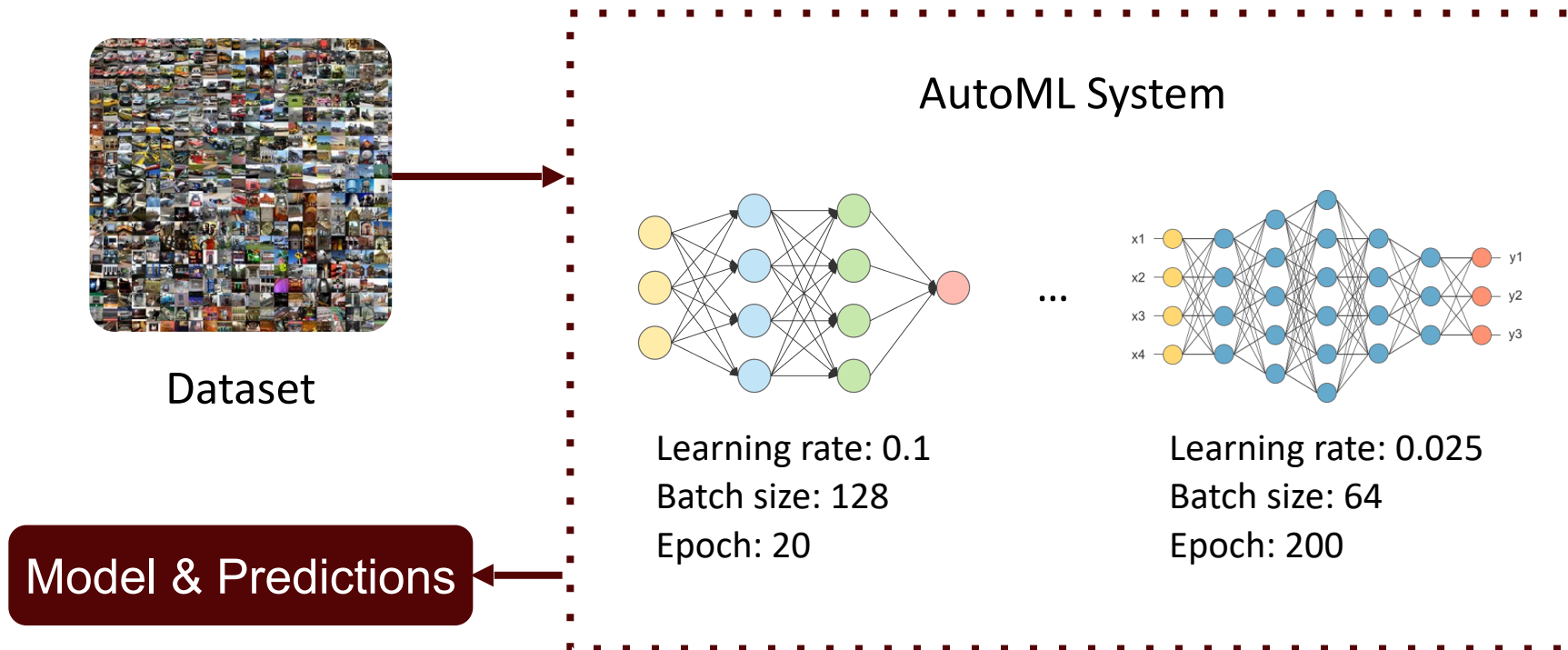


One-class based



Cluster based

Goal of AutoML System



Given a dataset, the AutoML system searches for the best neural architecture and hyperparameters

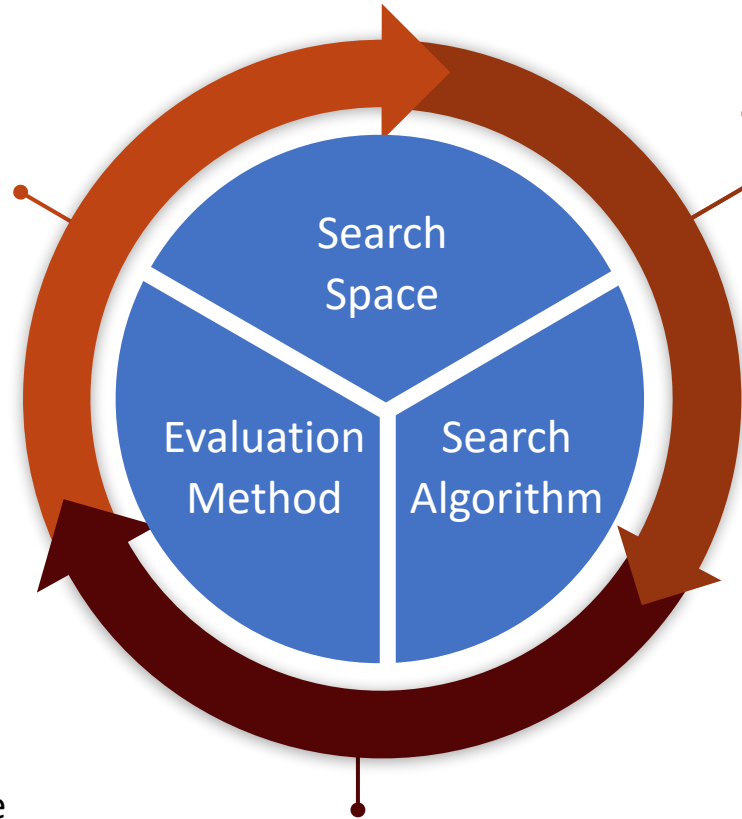
Automation of Deep Learning (1/3)

3. Update

Update the surrogate model with training history (architectures and their performances)

1. Generate

Generate the next architecture for observation



2. Observe

Train the architecture and evaluate its performance

Number of Iterations Average Observation Time

$$O(n\bar{t})$$

Automation of Deep Learning (2/3)

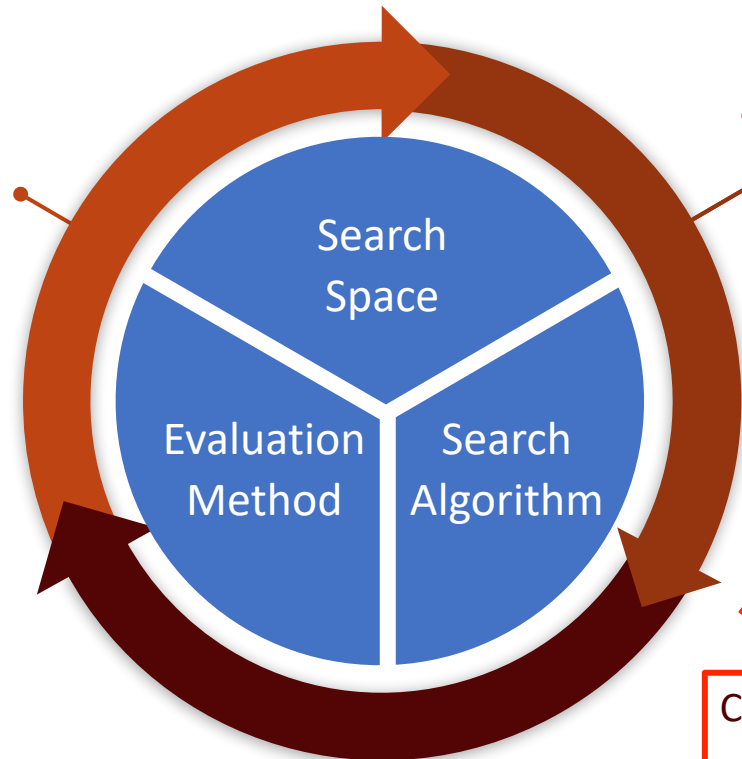
3. Update

Update the surrogate model with training history (architectures and their performances)

Challenge to **Evaluation**

Training neural networks from scratch takes a long time

$$O(n\bar{t})$$



1. Generate

Generate the next architecture for observation

2. Observe

Train the architecture and evaluate its performance

Challenge to **Search Algorithm**

Evolutionary algorithm [1], reinforcement learning [2] need to go through the NAS loop too many times

[1] Suganuma, and etc. "A genetic programming approach to designing convolutional neural network architectures.", 2017.

[2] Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." ArXiv 2016.

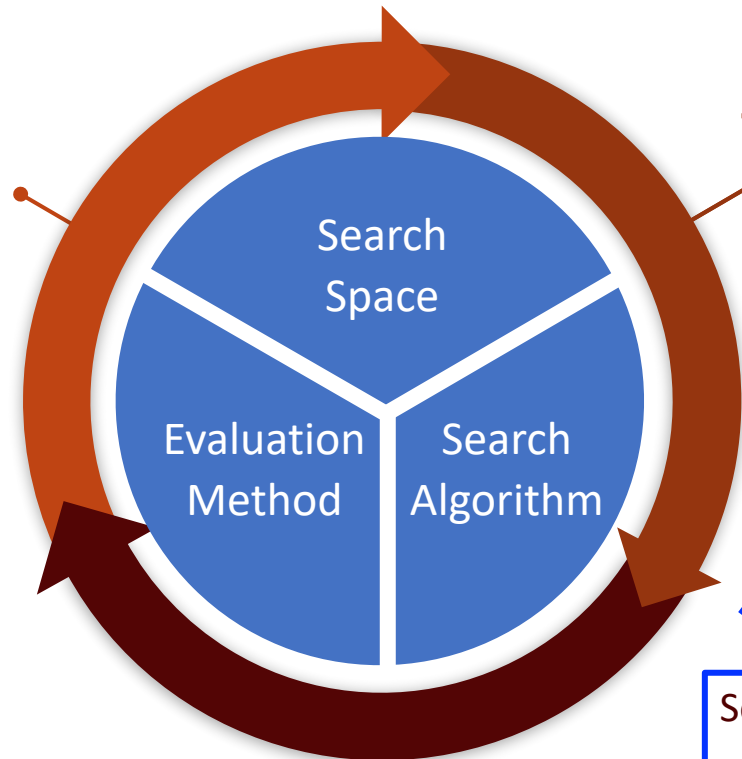
Automation of Deep Learning (3/3)

3. Update

Update the surrogate model with training history (architectures and their performances)

1. Generate

Generate the next architecture for observation



2. Observe

Train the architecture and evaluate its performance

Solution to **Evaluation**
Network morphism
makes use of the weights
in trained architectures

Solution to **Search Algorithm**
Bayesian optimization
requires less number of
observations

Efficient Search Algorithm

Bayesian-guided network morphism in AutoKeras [1]

- Propose a new neural architecture search (NAS) method based on **Bayesian optimization** and **network morphism**
- Bayesian Optimization (BO) is widely used in AutoML (model selection, hyper-param tuning)
- We want to explore the capability of BO on NAS to make it more efficient

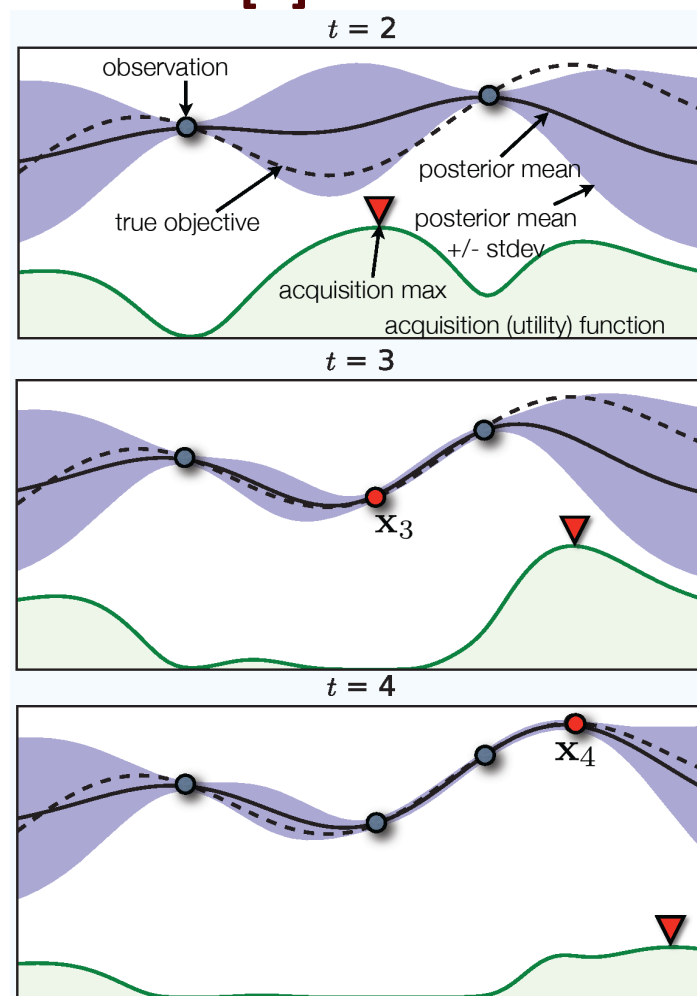


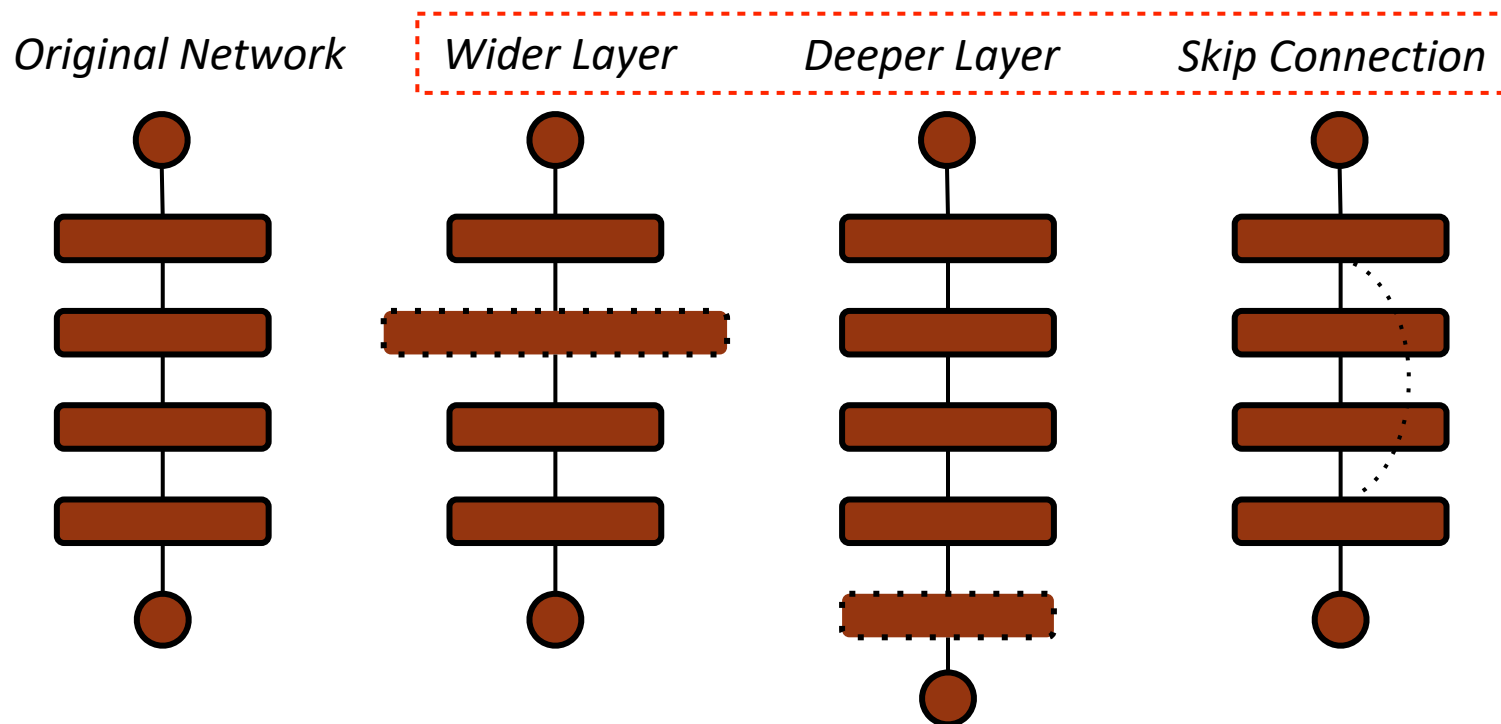
Image Source: Brochu et al, 2010.

[1] Jin, Haifeng, et al. "Auto-keras: An efficient neural architecture search system.", *ACM SIGKDD 2019*.

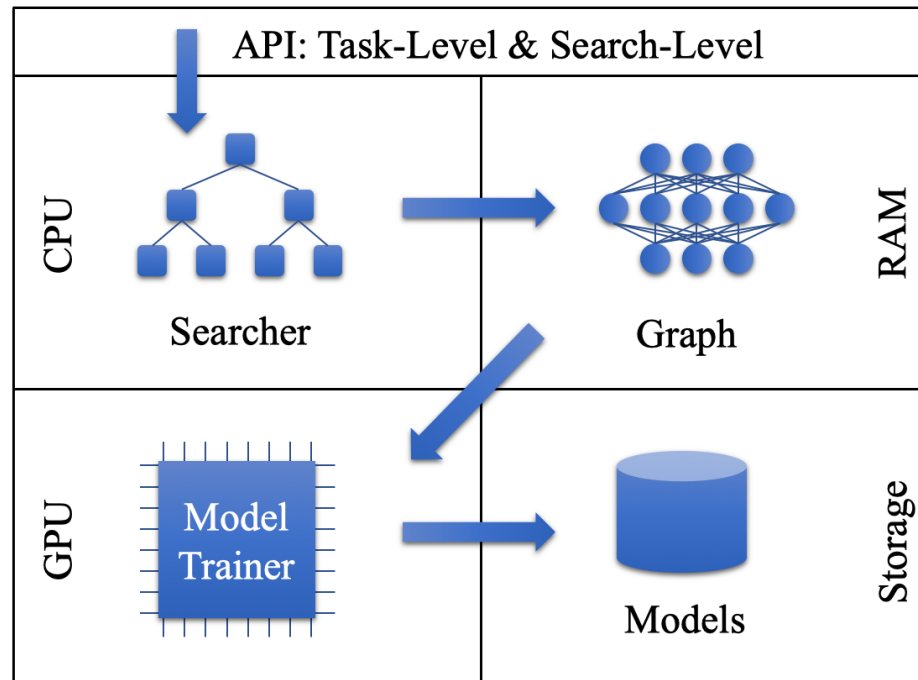
Speed Up the Evaluation Process

Network morphism (AutoKeras)

- *Change the neural architecture while preserve the functionality*
- Support 3 operations: wider layer/deeper layer/skip connection

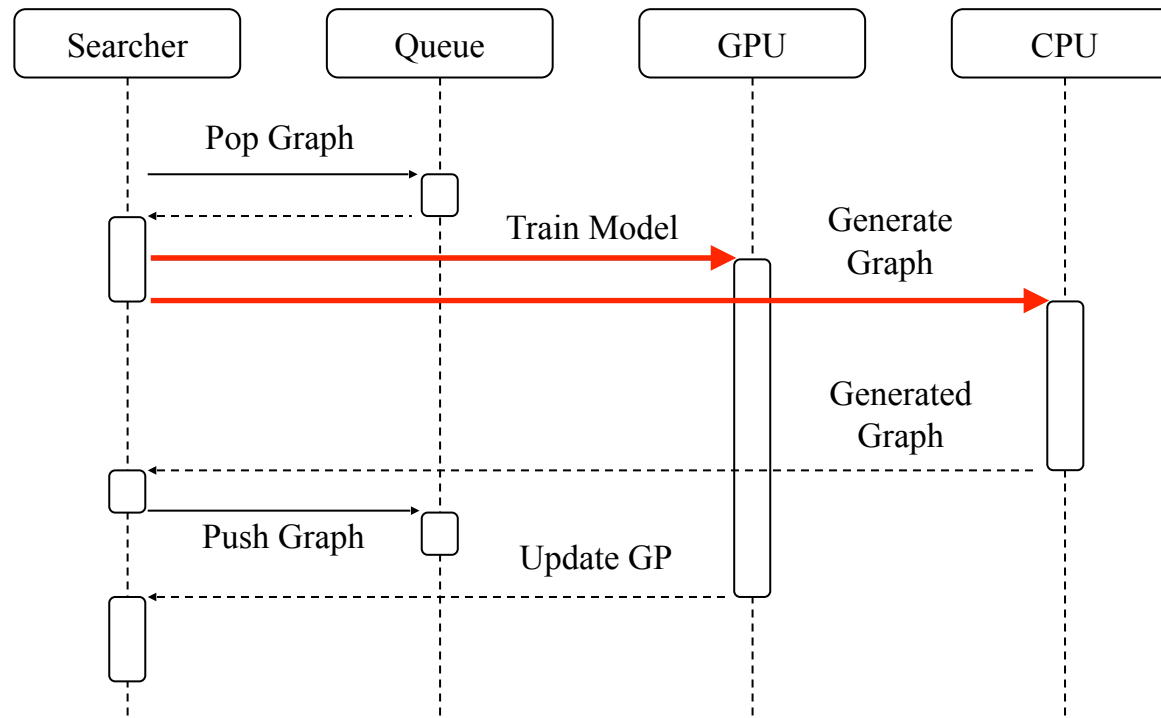


System Architecture



- 1 User call from the ***AutoKeras API***
- 2 The Bayesian ***search*** is conducted on ***CPU***
- 3 The ***training*** of the neural network is on ***GPU***
- 4 All ***searched models are stored*** on a storage device (hard disk)

Searcher and Model Parallelization



- GPU and CPU run *in parallel*
- Searcher passes a neural network to **GPU for training**
- Concurrently, searcher runs BO to **generate new neural architectures on CPU**

Experiment: Effectiveness

Datasets:

MNIST, CIFAR10, FASHION-MNIST.

Baseline methods:

Simple: grid search, random search

Traditional: SMAC¹, SPMT²

State-of-the-art: SEAS³, NASBOT⁴

Variant: BFS, BO

Our Method: NASNM

Setup:

Single GPU, 12 hours.

Table 1: Classification Error Rate

| Methods | MNIST | CIFAR10 | FASHION |
|---------|--------------|---------------|--------------|
| RANDOM | 1.79% | 16.86% | 11.36% |
| GRID | 1.68% | 17.17% | 10.28% |
| SPMT | 1.36% | 14.68% | 9.62% |
| SMAC | 1.43% | 15.04% | 10.87% |
| SEAS | 1.07% | 12.43% | 8.05% |
| NASBOT | NA | 12.30% | NA |
| BFS | 1.56% | 13.84% | 9.13% |
| BO | 1.83% | 12.90% | 7.99% |
| AK | 0.55% | 11.44% | 7.42% |
| AK-DP | 0.60% | 3.60% | 6.72% |

[1] Hutter, Frank, et al. "Sequential model-based optimization for general algorithm configuration.", 2011.

[2] Snoek, Jasper, et al. "Practical bayesian optimization of machine learning algorithms.", 2012.

[3] Elsken, Thomas, et al. "Simple and efficient architecture search for convolutional neural networks." arXiv, 2017.

[4] Kandasamy, Kirthevasan, et al. "Neural Architecture Search with Bayesian Optimisation and Optimal Transport." arXiv, 2018.

Experiment: Effectiveness

Datasets:

MNIST, CIFAR10, FASHION-MNIST.

Baseline methods:

Simple: grid search, random search

Traditional: SMAC¹, SPMT²

State-of-the-art: SEAS³, NASBOT⁴

Variant: BFS, BO

Our Method: NASNM

Setup:

Single GPU, 12 hours.

Table 1: Classification Error Rate

| Methods | MNIST | CIFAR10 | FASHION |
|---------|--------------|---------------|--------------|
| RANDOM | 1.79% | 16.86% | 11.36% |
| GRID | 1.68% | 17.17% | 10.28% |
| SPMT | 1.36% | 14.68% | 9.62% |
| SMAC | 1.43% | 15.04% | 10.87% |
| SEAS | 1.07% | 12.43% | 8.05% |
| NASBOT | NA | 12.30% | NA |
| BFS | 1.56% | 13.84% | 9.13% |
| BO | 1.83% | 12.90% | 7.99% |
| AK | 0.55% | 11.44% | 7.42% |
| AK-DP | 0.60% | 3.60% | 6.72% |

[1] Hutter, Frank, et al. "Sequential model-based optimization for general algorithm configuration.", 2011.

[2] Snoek, Jasper, et al. "Practical bayesian optimization of machine learning algorithms.", 2012.

[3] Elsken, Thomas, et al. "Simple and efficient architecture search for convolutional neural networks." arXiv, 2017.

[4] Kandasamy, Kirthevasan, et al. "Neural Architecture Search with Bayesian Optimisation and Optimal Transport." arXiv, 2018.

Experiment: Effectiveness

Datasets:

MNIST, CIFAR10, FASHION-MNIST.

Baseline methods:

Simple: grid search, random search

Traditional: SMAC¹, SPMT²

State-of-the-art: SEAS³, NASBOT⁴

Variant: BFS, BO

Our Method: NASNM

Setup:

Single GPU, 12 hours.

Table 1: Classification Error Rate

| Methods | MNIST | CIFAR10 | FASHION |
|--------------|--------------|---------------|--------------|
| RANDOM | 1.79% | 16.86% | 11.36% |
| GRID | 1.68% | 17.17% | 10.28% |
| SPMT | 1.36% | 14.68% | 9.62% |
| SMAC | 1.43% | 15.04% | 10.87% |
| SEAS | 1.07% | 12.43% | 8.05% |
| NASBOT | NA | 12.30% | NA |
| BFS | 1.56% | 13.84% | 9.13% |
| BO | 1.83% | 12.90% | 7.99% |
| AK | 0.55% | 11.44% | 7.42% |
| AK-DP | 0.60% | 3.60% | 6.72% |

[1] Hutter, Frank, et al. "Sequential model-based optimization for general algorithm configuration.", 2011.

[2] Snoek, Jasper, et al. "Practical bayesian optimization of machine learning algorithms.", 2012.

[3] Elsken, Thomas, et al. "Simple and efficient architecture search for convolutional neural networks." arXiv, 2017.

[4] Kandasamy, Kirthevasan, et al. "Neural Architecture Search with Bayesian Optimisation and Optimal Transport." arXiv, 2018.

Experiment: Efficiency

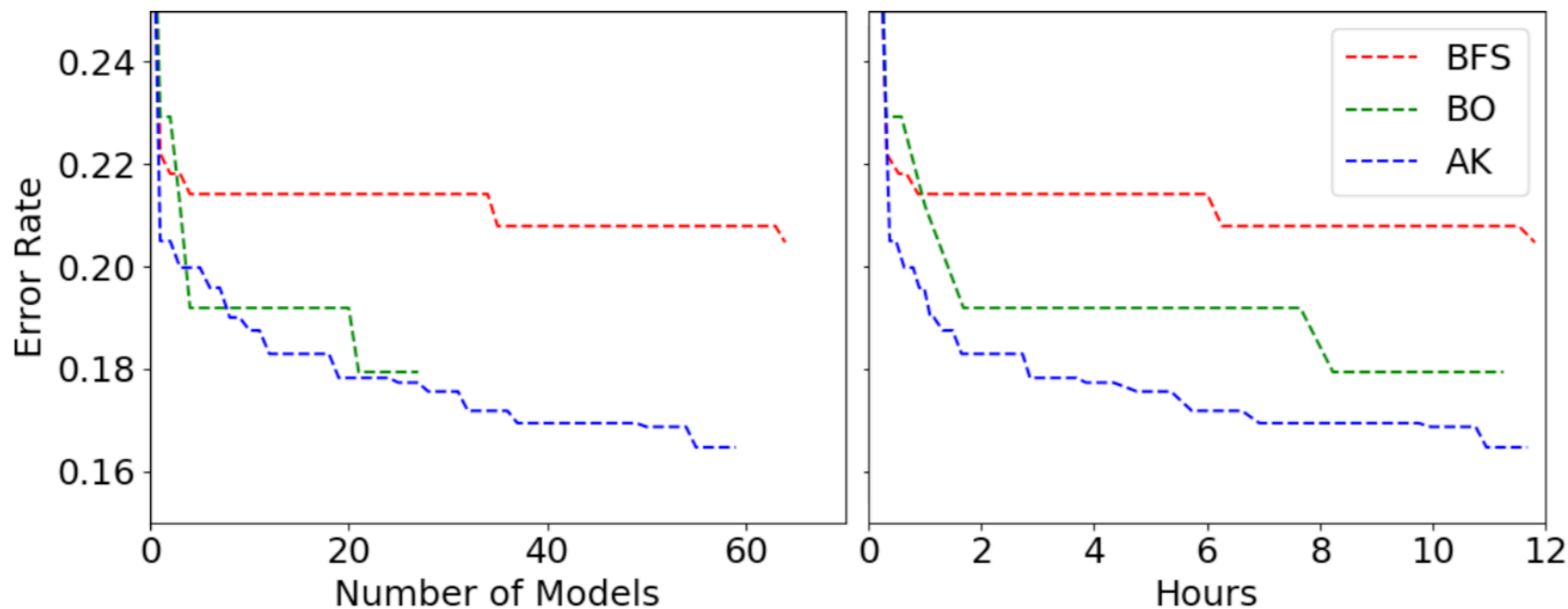


Figure 4: Evaluation of Efficiency. The two figures plot the same result with different X-axis. BFS uses network morphism. BO uses Bayesian optimization. AK uses both.

AutoKeras: An Open-source AutoML System

- AutoKeras provides easy-to-use solutions to deep learning tasks



Single GPU



Open-Source

```
import autokeras as ak

clf = ak.ImageClassifier()
clf.fit(x_train, y_train)
results = clf.predict(x_test)
```

Concise Interface

- Visit autokeras.com for more information



| | | | | | |
|-------|---------|---------|------|--------------|-------|
| Watch | 305 | Star | 7.3k | Fork | 1.2k |
| Tests | passing | codecov | 100% | pypi package | 1.0.6 |

Road Map

1 AutoML System for Deep Learning

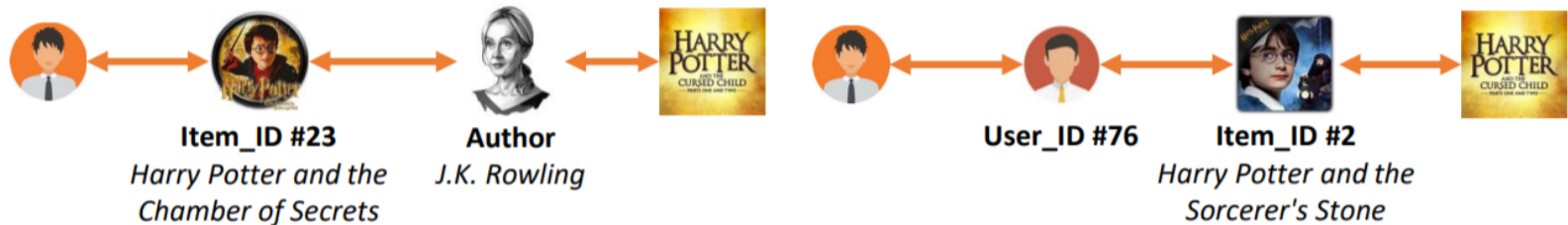


Tests passing

codecov 100%

pypi package 1.0.6

2 AutoML System for Recommendation

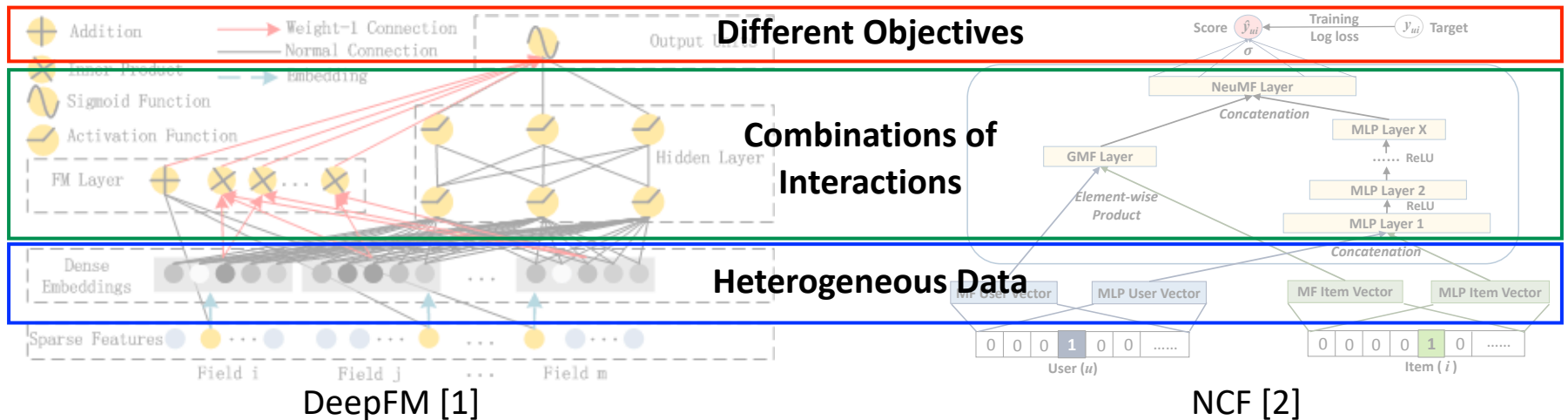


3 AutoML System for Outlier Detection



Why Automated Recommendation

It is difficult to decide the proper architecture due to *diverse feature interactions, heterogeneous data, and high data volume, etc.*



Our Goal — Automate architectural design, and with better performance

- Abstract and modularize **virtual blocks** to formulate a generalizable **search space** for recommendation tasks, namely CTR and rating prediction
- Better **search algorithm** which improves performance

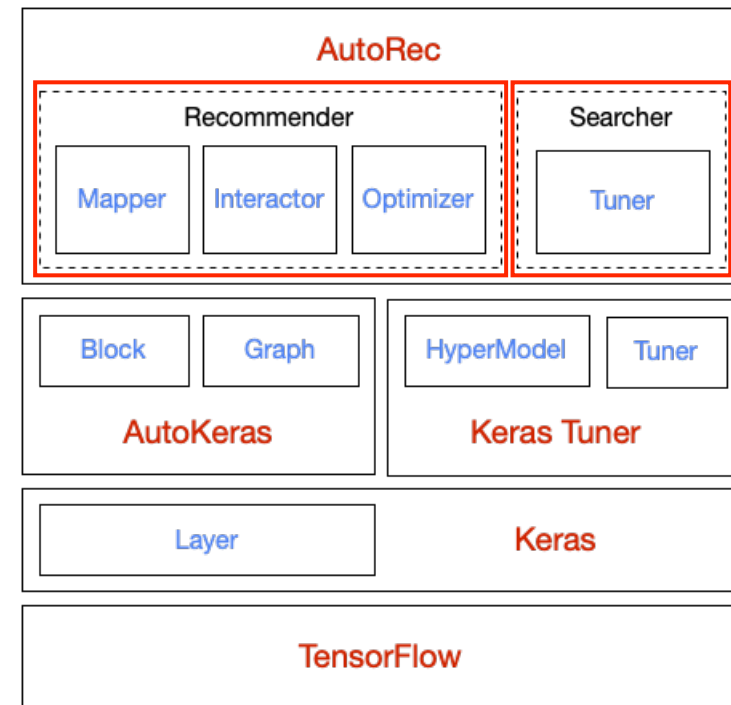
[1] Guo, Huifeng, et al. "DeepFM: a factorization-machine based neural network for CTR prediction." *arXiv* 2017.

[2] He, Xiangnan, et al. "Neural collaborative filtering." *WWW* 2017.

Scenario-Based Abstract Search Space

Selectable search algorithms for BOTH *model selection* and *HP tuning*

- **Searcher**
 - **Tuner:** Random/Greedy/Bayesian
- **Model & HP search in Recommender**
 - **Model search:** hyper interaction
 - Types of interaction and ways to stack
 - **HP search:** interactor-specific
 - E.g., units, layers, dropout for MLP



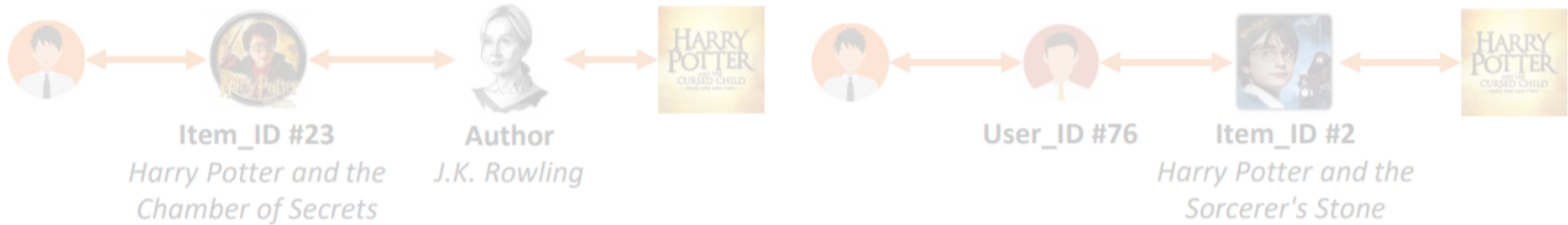
Road Map

1 AutoML System for Deep Learning



Tests passing codecov 100% pypi package 1.0.6

2 AutoML System for Recommendation



3 AutoML System for Outlier Detection

Reconstruction based

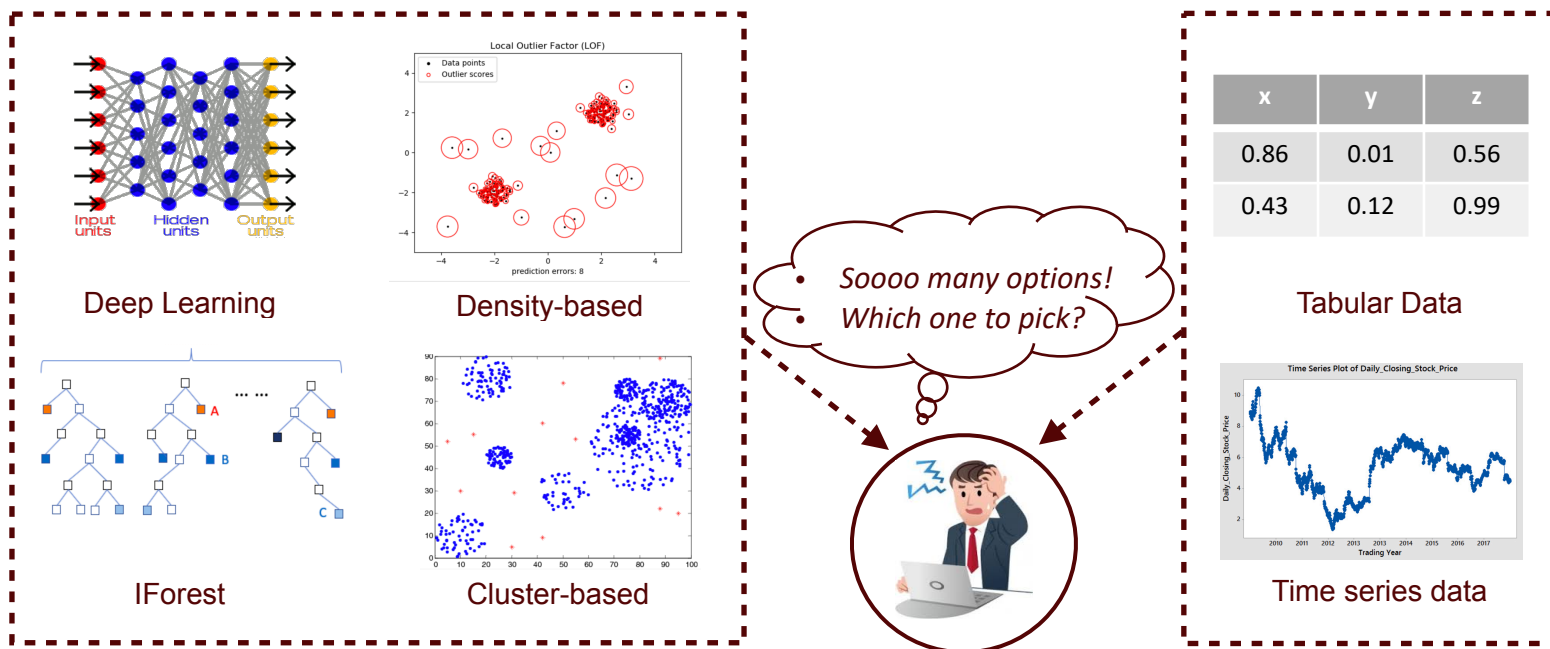
Density based

One-class based

Cluster based

Why Automated Outlier Detection

Existing NAS methods are unsuitable for automating OD because they *ignore the properties of outliers* and the *data is imbalanced*



Our Goal — Consider outlier def. in search and data distribution in training

- **Search for definition-hypothesis**, which decides objective function
- **Explore untapped search space** and **exploit useful past sample**

Conclusions

1 AutoML System for Deep Learning

- *Bayesian optimization* speeds up the *NAS loop* by reducing the number of training iterations, while *network morphism* speeds up *evaluation* by using the weights of trained architecture

2 AutoML System for Recommendation

- *Abstract virtual blocks* create flexible **search space** to accommodate different recommendation scenarios (inputs, interactions, and objectives)
- *Multi-objective evolutionary search algorithm* considers architecture *evaluation* and *computational complexity* and produced new SOTA

3 AutoML System for Outlier Detection

- *Curiosity-guided search strategy* addresses local optimality and *self-imitation learning* improves sample efficiency

Future Directions

1 AutoML Infrastructure

- Modules of AutoML systems are complicated. How can we have a unified and well-organized infrastructure for different AutoML systems? A potential solution is DARPA D3M infrastructure

2 Explainability

- Enabling explainability would be important for data scientists to easily make use of the outputted models by AutoML systems

3 Parallel Computation

- Further improve data/model/search efficiency in AutoML

Acknowledgements

❖ DATA Lab and Collaborators

Data Analytics at Texas A&M (DATA Lab)

❖ Funding Agencies

--- *Defense Advanced Research Projects Agency (DARPA)*

--- *National Science Foundation (NSF)*

--- *Industrial Sponsors (Samsung, Adobe, Apple, LinkedIn, etc.)*

❖ Everyone attending the talk!