Generate Neural Network Automatically with High Accuracy and High Efficiency

Xue Chao

IBM Research China

Outline

- Standard AutoML: Neural network and hyper-parameter selections
  - Bayesian optimization
  - Hyperband
  - Reinforcement learning
- Transfer AutoML
  - Using historical hyper-parameter configurations for different tasks

Standard AutoML--from Random search, Bayesian Optimization to Reinforcement Learning

Hyperparameters auto selected

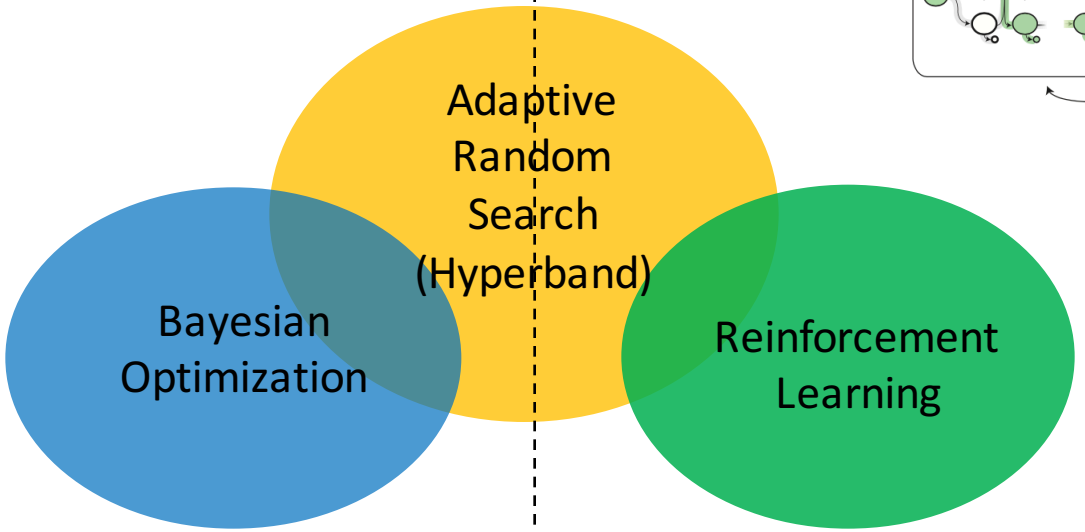
	$s=4$		$s=3$		$s=2$		$s=1$		$s=0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix}.$$

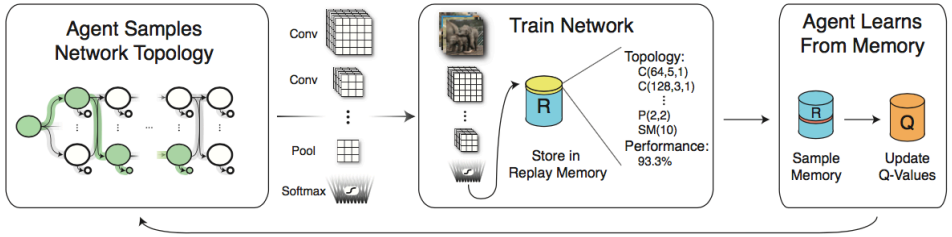
$$\begin{aligned} \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b) \\ \boldsymbol{\Sigma}_{a|b} &= \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba}. \end{aligned}$$

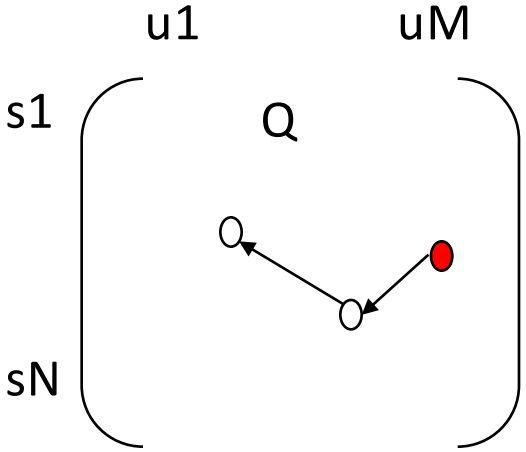
$$\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m})$$
$$\sigma_n^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}),$$

$$\alpha_{\mathrm{PI}}(\mathbf{x}; \mathcal{D}_n) := \mathbb{P}[v > \tau] = \Phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right)$$



Neural Network auto selected





Evaluations on DL Insight with Acoustic Applications

- We implement the **AdaRandom (adaptive random search scheme)** and **Reinforce (reinforcement learning search scheme)** methods to generate deep learning neural network automatically.
- We are trying the new methods in different areas. Here is the example for acoustic. Default is the best scheme by manual tuning.

Lower complexity

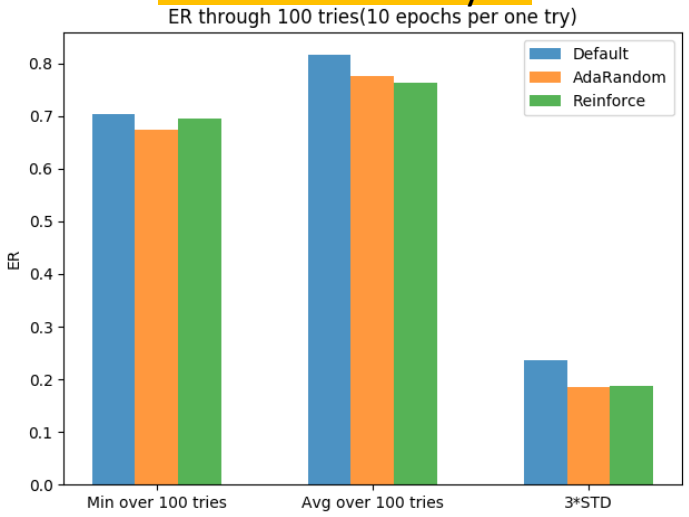
Neural Network:
C: Convolution, P: pooling, FC: Full connection.

Default manual: (9 layers)
C(3,32)+C(3,32)+P(3,3)+C(3,64)+C(3,64)+P(3,3)+FC(64)+FC(32)+FC(6)

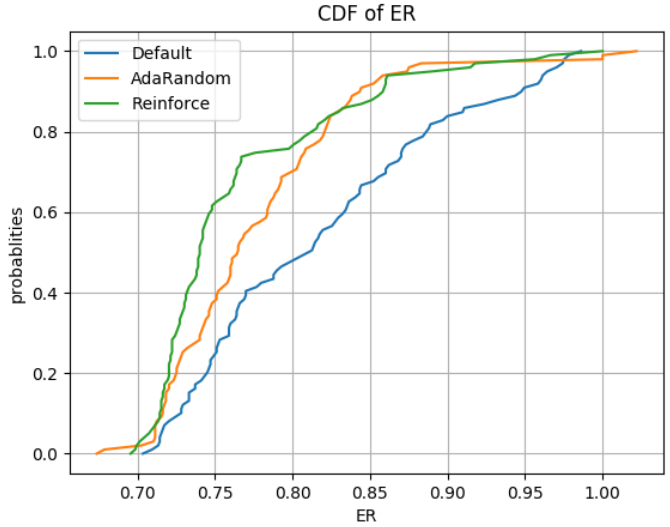
AdaRandom generated: (6 layers)
C(3,128)+C(4,64)+C(3,32)+P(3,3)+FC(16)+FC(6)

Reinforce generated: (5 layers)
C(3,32)+P(2,2)+P(3,2)+C(5,32)+FC(6)

Better accuracy



More stable

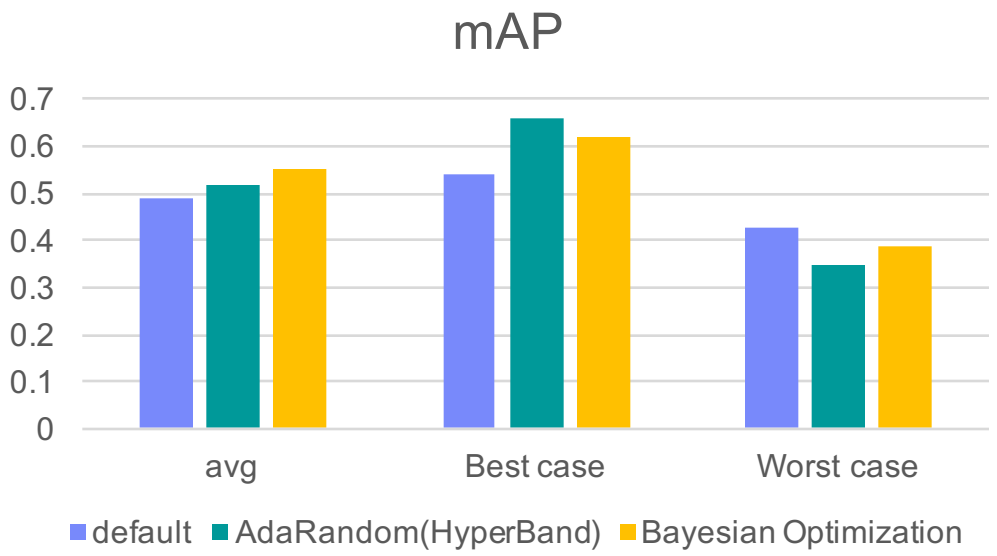


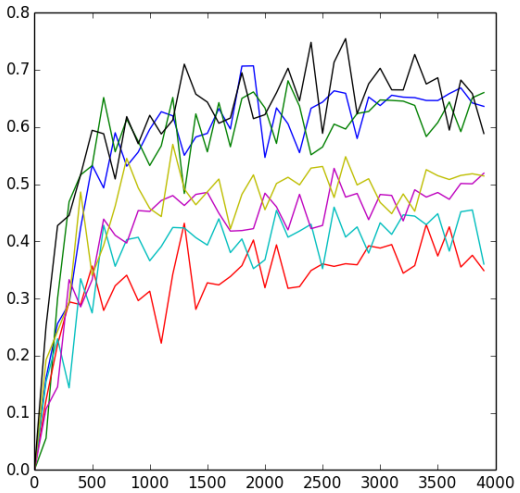
- The best results for the three networks are (0.703,0.673, 0.695) (the smaller the better), that is, using AdaRandom and Reinforce recommended models, you can gain **4.3%** and **1.1%** in the best results comparisons. The average result of the three networks is (0.817,0.776, 0.763), that is, the DL Insight recommended modes can increase about **5.0%** and **6.6%** in the average case performance. And from the standard deviation view, the recommended models are clearly more stable.
- The CDF (cumulative distribution function) curve is more intuitive to illustrate the comparison of the three models(the more left the better). For example, using reinforce recommended model, ER has more than **60% probability** (frequency) less than 0.75, while the default only has the **30%**.

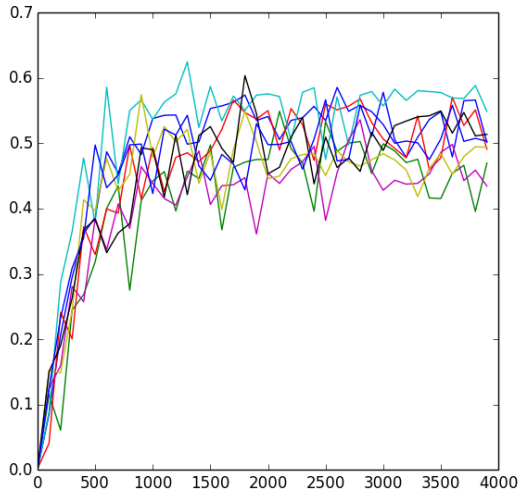
Results of AdaRandom and Bayesian Optimization for Object detection

- The raw data (no data augment)

Machine	IBM P8
GPU	Tesla K80
GPU MEM	10G
CPU	ppc64le
CPU cores	160
MEM	55G
Frequency	3.5 GHz
OS	Ubuntu 14.04
caffe version	Py-faster-rcnn
Dataset	User-defined
Model	vgg16



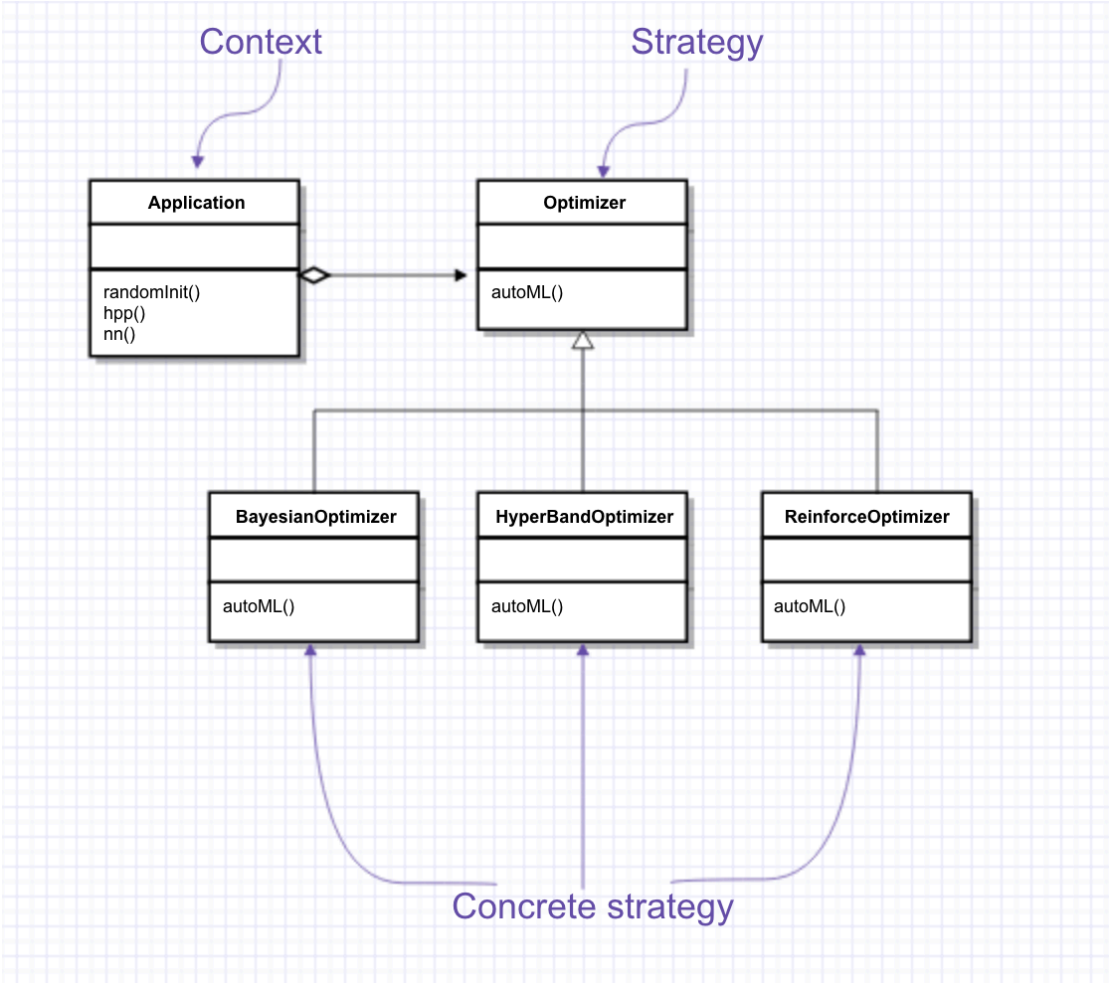




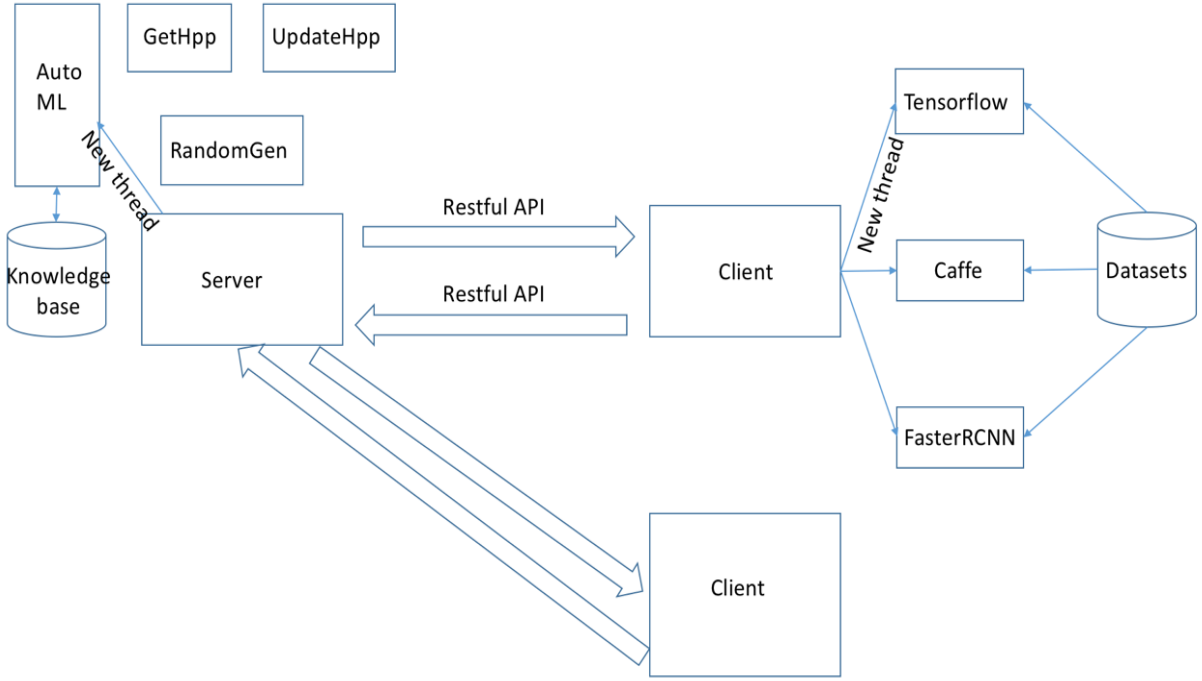
- The average results for the three (HPP) hyperparameters combinations at 4000 iterations are (0.49, 0.52, 0.55), that is, using AdaRandom and Bayesian optimization recommended HPP, you can gain **6%** and **12%** in the average results comparing to default setting. AdaRandom method has more variance among different tries(train-test dataset split). The Bayesian models are more stable while having better average performance.
- The below pictures show accuracy during the 0-4000 iterations, with different tries, under the two HPP configurations. We can see that: 1) It can be early stopped at about 1300 iterations. 2) the performance with different tries differ significantly, it caused by in some tries, training dataset has the invariance property according to test dataset, but some doesn't have. It need to augment data to gain the stable performance. 3) different HPP combinations(models) may have different sensitivity to the different tries.

Program design

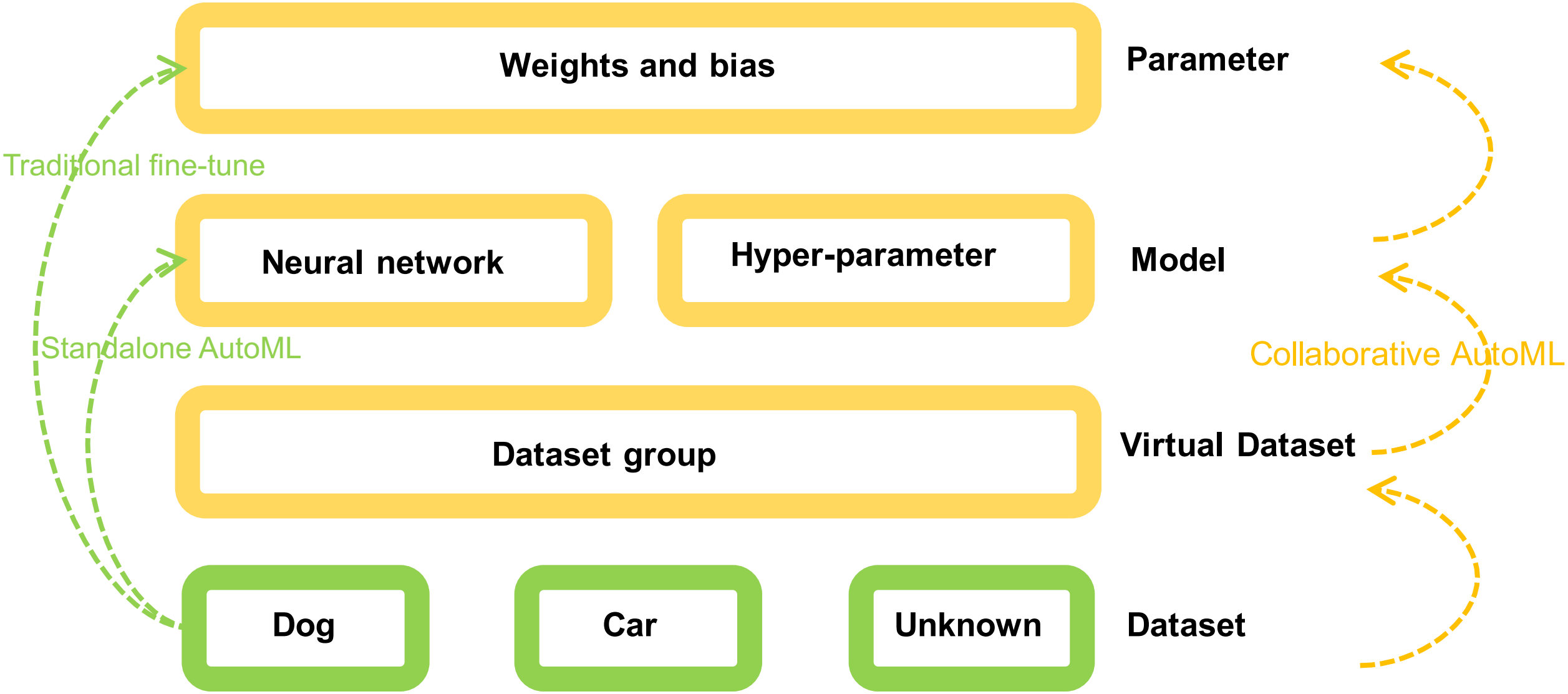
We implement these three standard methods by using design pattern of strategy whose UML is shown:



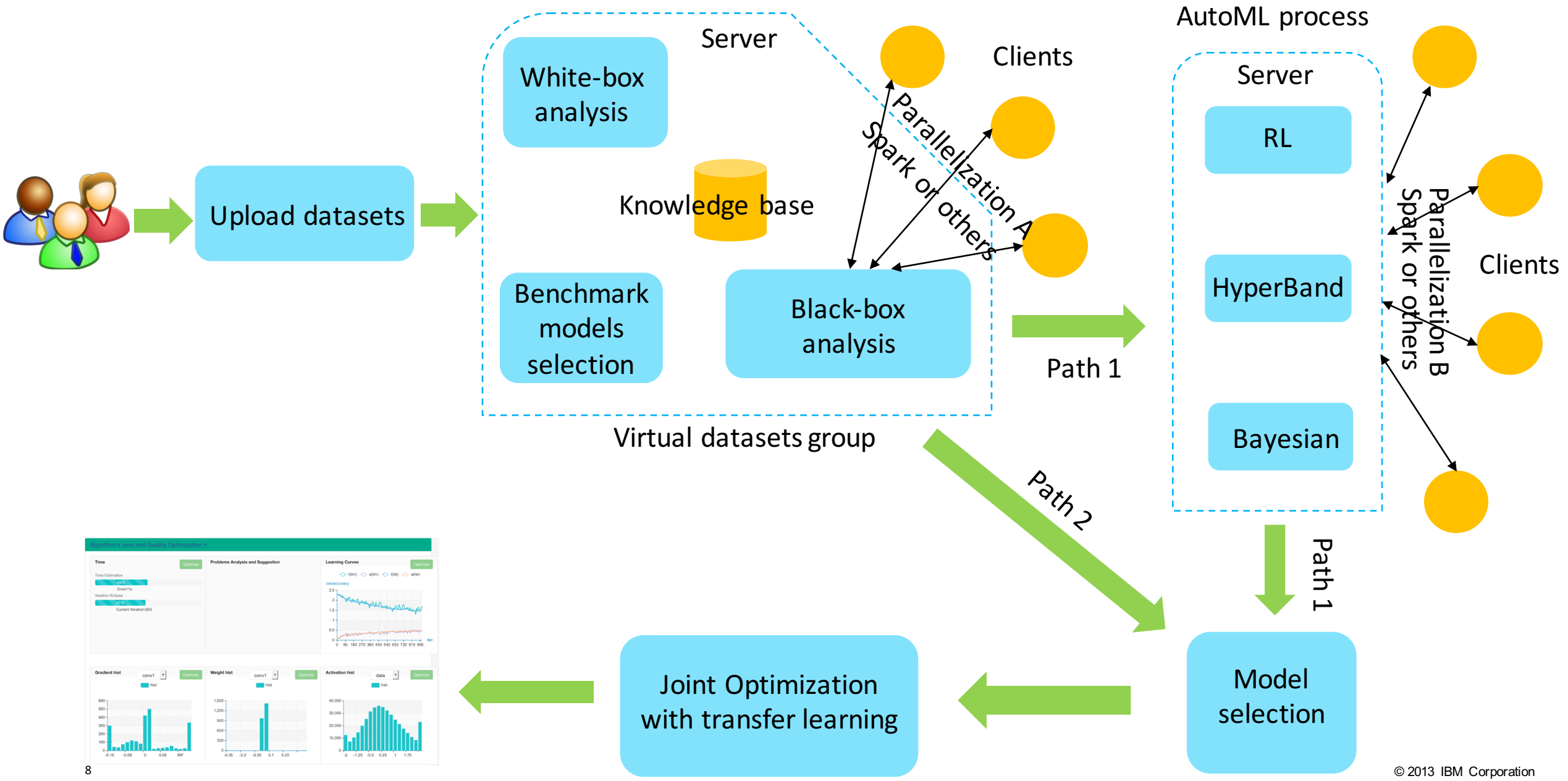
We implemented the system as the server-client model: the server is responsible for generating the candidate neural network, and scheduling the tasks to the clients. The clients focused on calculating the accuracy of the allocated neural networks. The tasks in the clients can be executed in many frameworks, such as Caffe, TF, which are transparent to the server.



Transfer AutoML framework



Transfer AutoML Architecture



Challenges and solutions in AutoML

- Short responding time Vs. long time to do AutoML
  - High efficient AutoML
    - Used share information among tasks
  - Parallelization B is important to speed up AutoML
    - Design paralleled AutoML algorithms
- Large datasets and deep neural network
  - Requires huge resources using AutoML for large datasets and deep neural network
    - Reduce the searching space methods
    - Parallelization B is important to split large datasets and deep neural networks into calculable size in one machine

Challenges for AutoML with Transfer Learning

- Training small user dataset leads to convergence problem → **Transfer learning** is needed
- When considering transfer learning, the **pretrained model** need to be chosen, usually in the computer vision, we choose **image-net** as the base dataset to get the initial weights as the pretrained model, but it **can't fit many specific user datasets.**
- To solve this transfer learning's problem, we can let the user to classify his dataset into some **predefined categories**, and in each category, the pretrained model was trained separately. It can improve the performance of transfer learning but **involve user's intervention** with their datasets.
- Using **AutoML** with transfer learning can improve transfer learning's performance without user's intervention. But considering the transfer learning's properties, there are **two challenges** for AutoML:
  - Since reusing the initial weights, transfer learning **limits the searching space** of AutoML, how to **use AutoML based on the pretrained model** is a question.
  - We can't use AutoML to build one model for every user dataset, it is too expensive. How to **reuse the model** for transfer learning is a question.

Joint optimization: AutoML with the fine-tune

Standard AutoML Used for Offline Stage

Search space:

1) Neural network at the last stage:
   Example: c p c p fc

2) Below hyper-parameter

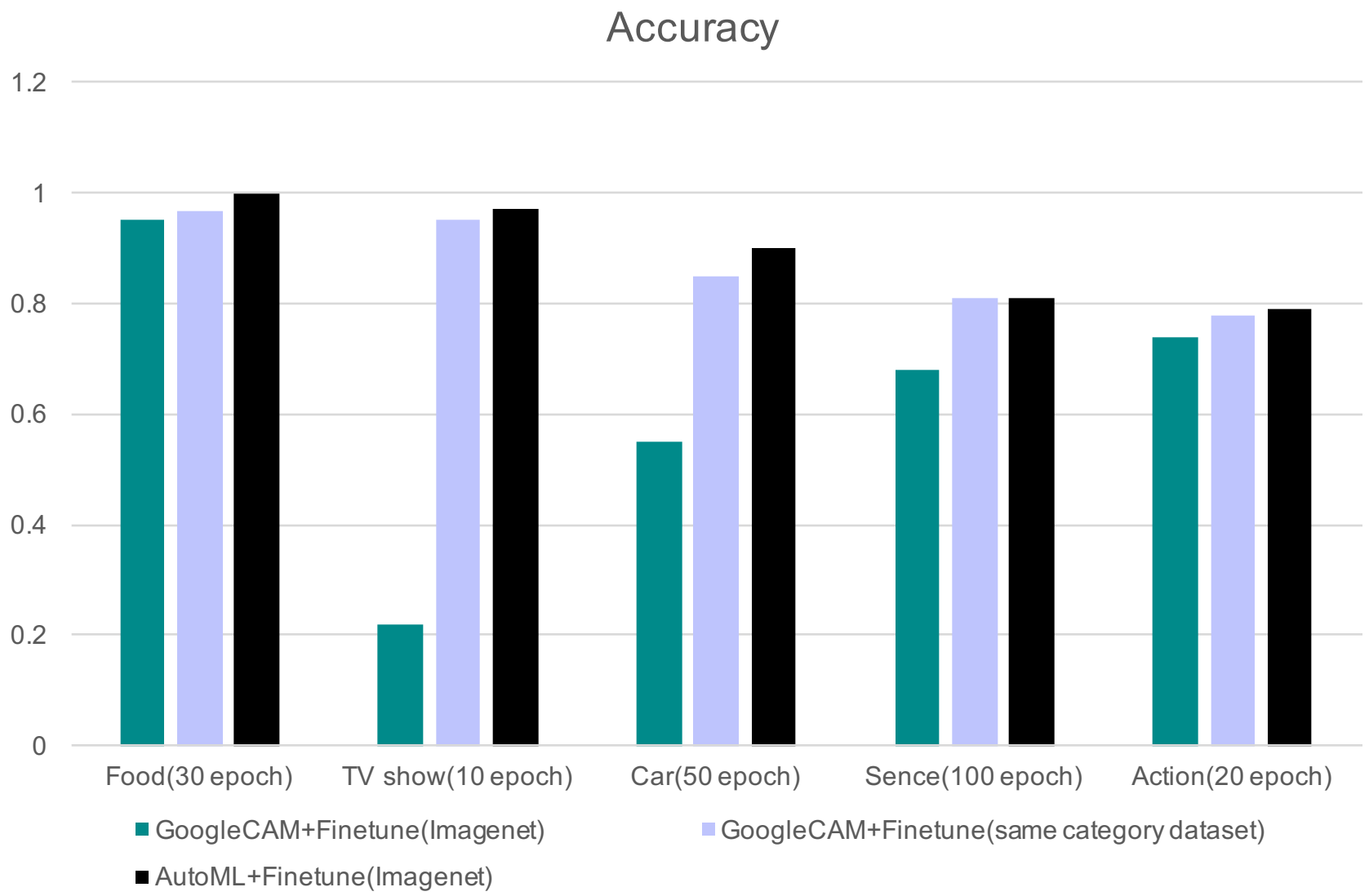
- lr_policy: LR_POLICY
  stepsize: STEPSIZE
  gamma: GAMMA
  momentum: MOMENTUM
  solver_mode: GPU
  max_iter: MAX_ITER
  test_iter: TEST_ITER
  test_interval: TEST_INTERVAL
  base_lr: BASE_LR
  weight_decay: WEIGHT_DECAY
  solver_type: SGD

```
layer {
 param {
  lr_mult: LR_MULT_C_W_0
  decay_mult: DECAY_MULT_C_W_0
 }
 param {
  lr_mult: LR_MULT_C_B_0
  decay_mult: DECAY_MULT_C_B_0
 }
 convolution_param {
  num_output: NUM_OUTPUT_0
  pad: PAD_0
  kernel_size: KERNEL_SIZE_C_0
  group: GROUP_0
  weight_filler {
   type: TYPE_C_W_0
   std: STD_C_W_0
  }
  bias_filler {
   type: TYPE_C_B_0
   std: STD_C_B_0
  }
 }
}
```

```
layer {
 name: "conv0_relu"
 type: TYPE_C_AF_0
 bottom: "conv0"
 top: "conv0"
}
layer {
 name: "pool1"
 type: "Pooling"
 bottom: "conv0"
 top: "pool1"
 pooling_param {
  pool: AVE
  kernel_size: KERNEL_SIZE_P_0
  stride: STRIDE_P_0
 }
}
```

```
layer {
 name: "last_fc"
 type: "InnerProduct"
 bottom: "pool1"
 top: "last_fc"
 param {
  lr_mult: LR_MULT_FC_W_0
  decay_mult: DECAY_MULT_FC_W_0
 }
 param {
  lr_mult: LR_MULT_FC_B_0
  decay_mult: DECAY_MULT_FC_B_0
 }
 inner_product_param {
  num_output: OUTPUT_NUMS
  weight_filler {
   type: TYPE_FC_W_0
   std: STD_FC_W_0
  }
  bias_filler {
   type: TYPE_FC_B_0
   std: STD_FC_B_0
  }
 }
```

Results and Analysis



Advantages of AutoML with finetune:
(+) Best accuracy
(+) Most stable
(+) Don't need separate pretrained models by predefining dataset categories. No user's interventions.

Disadvantages of AutoML with finetue:
(-) Overhead cost of running benchmark models. But we can shorten the process to about 2 minutes by good design.

Reference

- [1] David Schaffer, Darrell Whitley and Larry J Eshelman, *Combinations of genetic algorithms and neural networks: A survey of the state of the art*. International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1992.
- [2] J.Snoek, H.Larochelle and R.P.Adams, *Practical Bayesian optimization of machine learning algorithms*. In Advances in Neural Information Processing Systems(NIPS), 2012.
- [3] Bergstra, James and Yoshua Bengio, *Random search for hyper-parameter optimization*. Journal of Machine Learning Research, 2012.
- [4] Lisha Li, Kevin Jamieson and Giulia DeSalvo, *HYPERBAND: BANDIT- BASED CONFIGURATION EVALUATION FOR HYPERPARAMETER OPTIMIZATION*. ICLR, 2017.
- [5] James Bergstra, etc. *Algorithms for Hyper-Parameter Optimization*. Proceedings of the IEEE, 2012.
- [6] Bowen Baker, Otkrist Gupta, Nikhil Naik and RameshRaskar, *DESIGNING NEURAL NETWORK ARCHITECTURES USING REINFORCEMENT LEARNING*. ICLR, 2017.

Thank you for listening