MO809A - Seminar

Towards Non-I.I.D. and Invisible Data with FedNAS: Federated Deep Learning via Neural Architecture Search

Rafael C. Ito

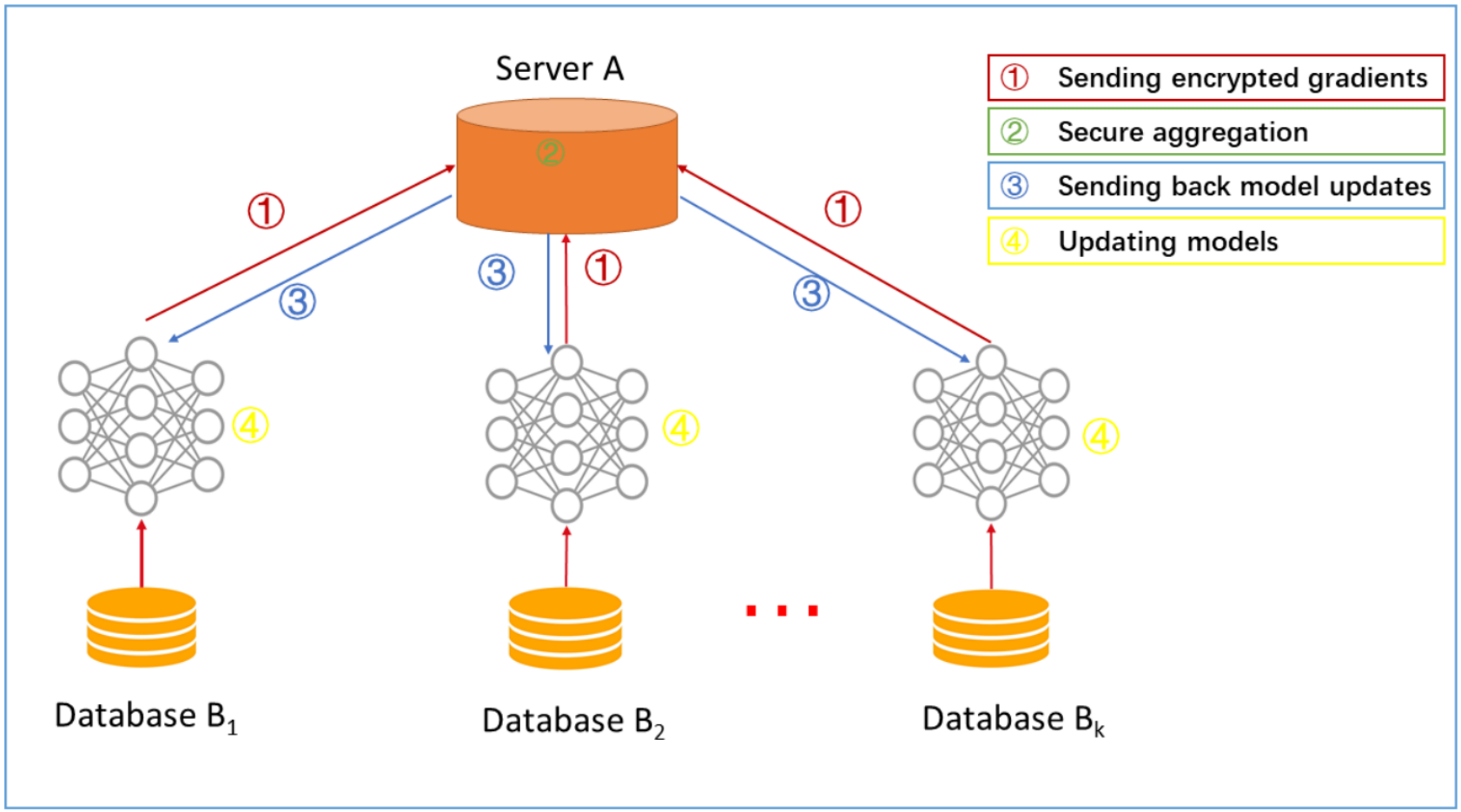
17/11/2022

Outline

Section 1

Federated Learning

The Framework

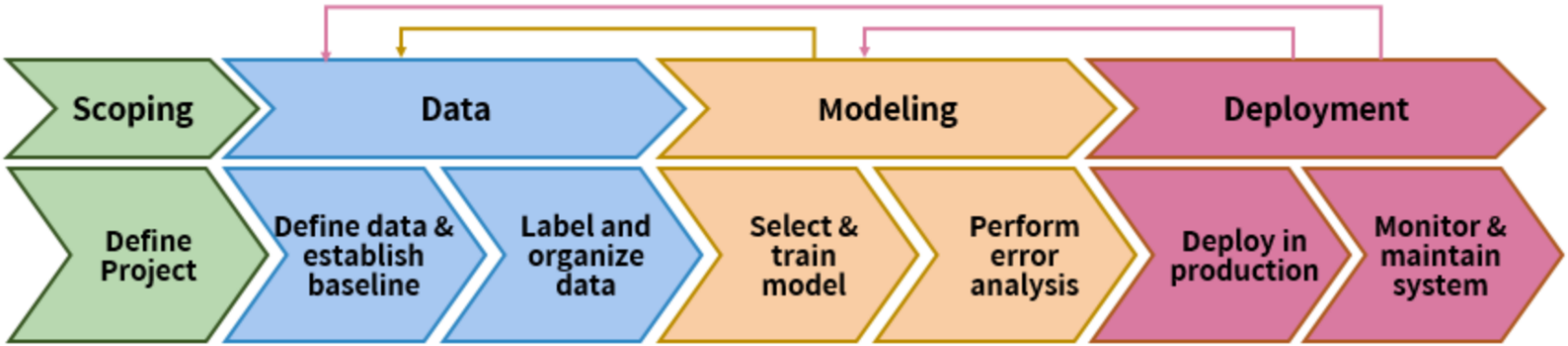


(Yang et al. 2019)

Section 2

Neural Architecture Search

ML Pipeline (Production)

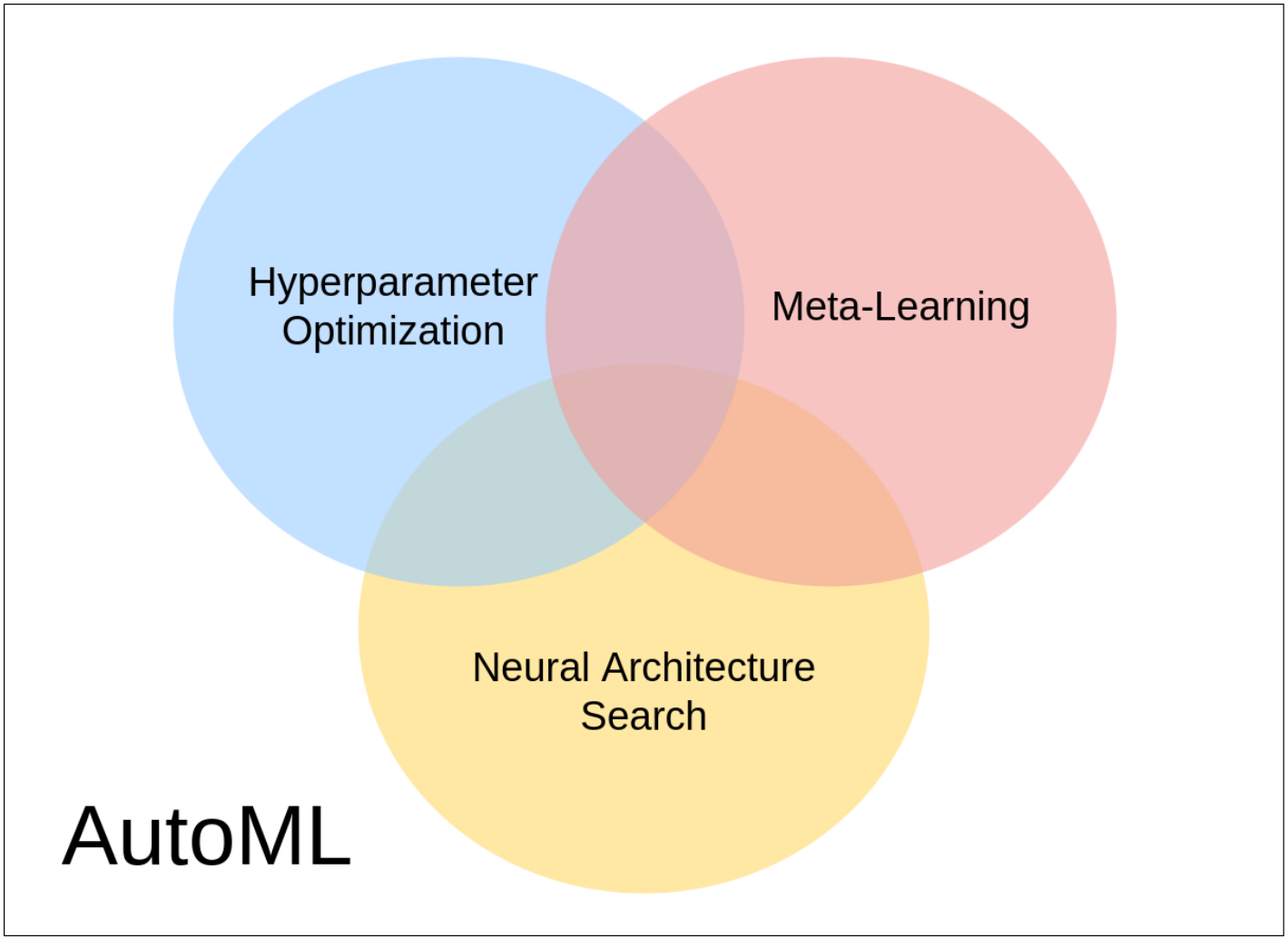


(deeplearning.ai MLOps course)

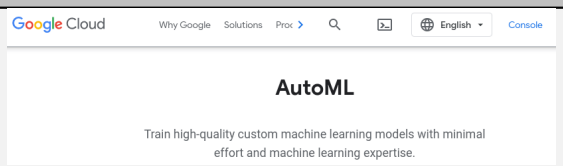
Model Development

Manual Search
- ML specialist
- Field specialist
Hyperparameter choices
- Regularizations parameters
- Learning rate
- Optimizer
Start from Scratch
- Trial and error
Architecture choices
- Number of layers
- Operations
- Number of filters
- Kernel size
- Connections

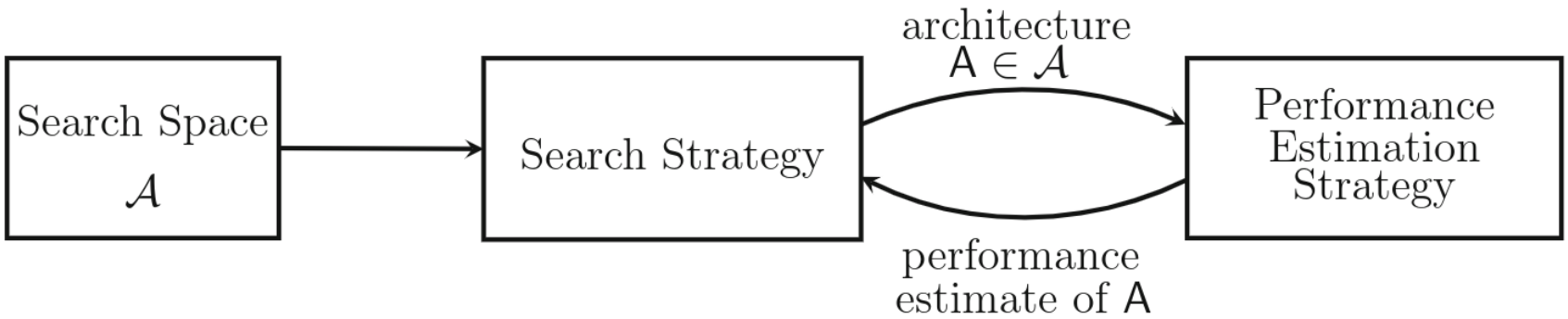
AutoML


Hyperparameter Optimization
Meta-Learning
Neural Architecture Search
AutoML

Automating ML

Manual Search	AutoML
	
- ML specialist - Field specialist	https://cloud.google.com/automl
Hyperparameter choices	**HPO**
- Regularizations parameters - Learning rate - Optimizer	- Grid search - Bayesian Optimization - Bandit-based Algorithms
Start from Scratch	**Meta-learning**
- Trial and error	- Transfer Learning - Few-shot Learning
Architecture choices	**NAS**
- Number of layers - Operations - Number of filters - Kernel size - Connections	Search Algorithms - Reinforcement Learning - Evolutionary Algorithms - Gradient-based - Others

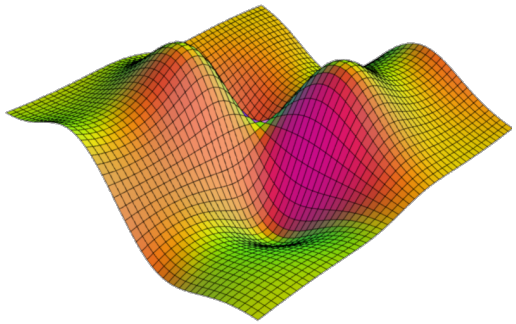
The Framework

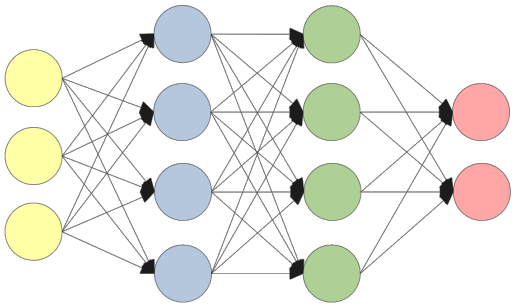


(Hutter, Kotthoff, and Vanschoren 2019)

The Framework

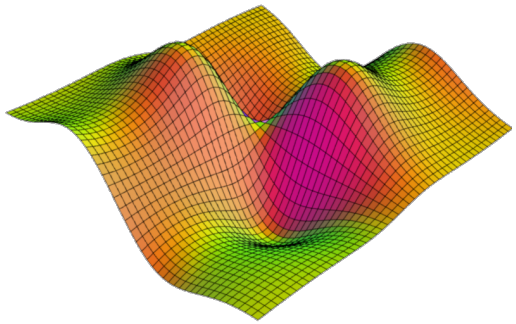
Search Space

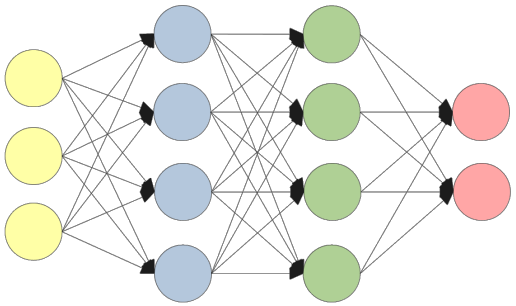




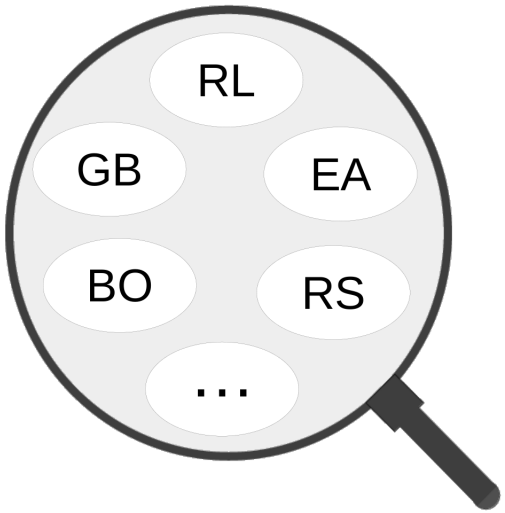
The Framework

Search Space



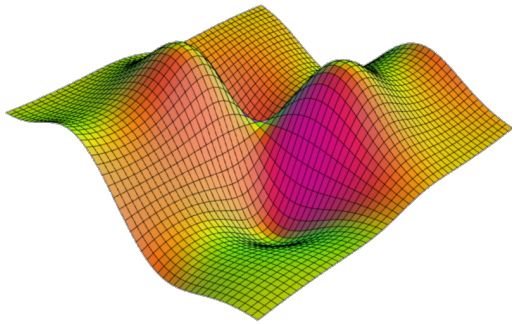


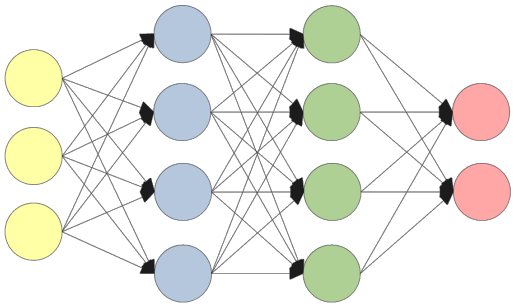
Search Strategy



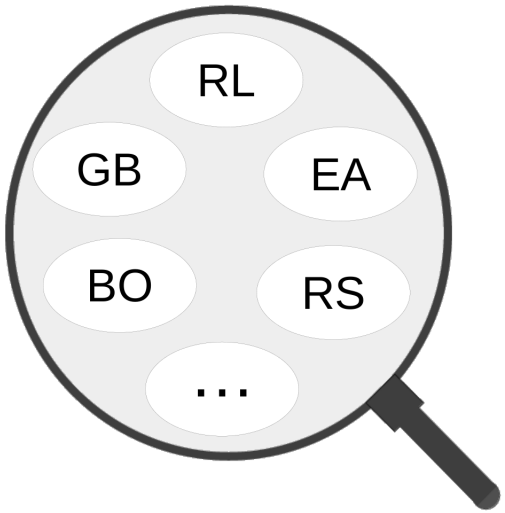
The Framework

Search Space





Search Strategy



Search Evaluation

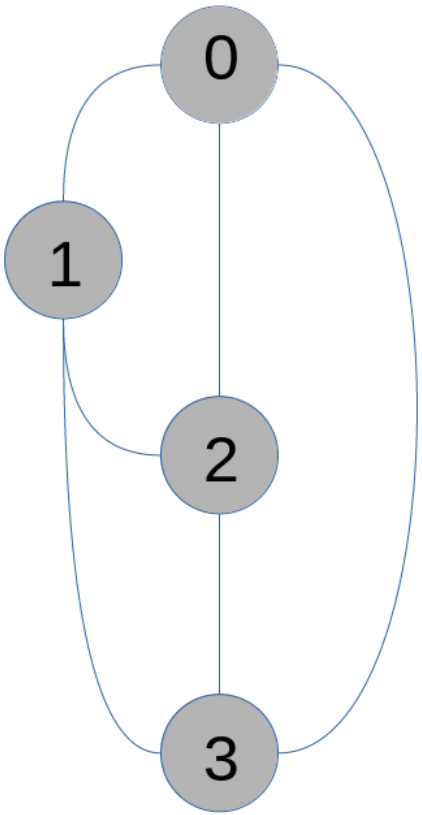
- Full training
- Partial training
- Weight-sharing
- Network morphism
- Hypernetworks
- Performance prediction

Classification

- **Discrete Search Space**:
  - Vertices/Nodes: operations
  - Edges: connections
- **Continuous Search Space**:
  - Vertices/Nodes: latent representation (eg: feature map in conv nets)
  - Edges: operations

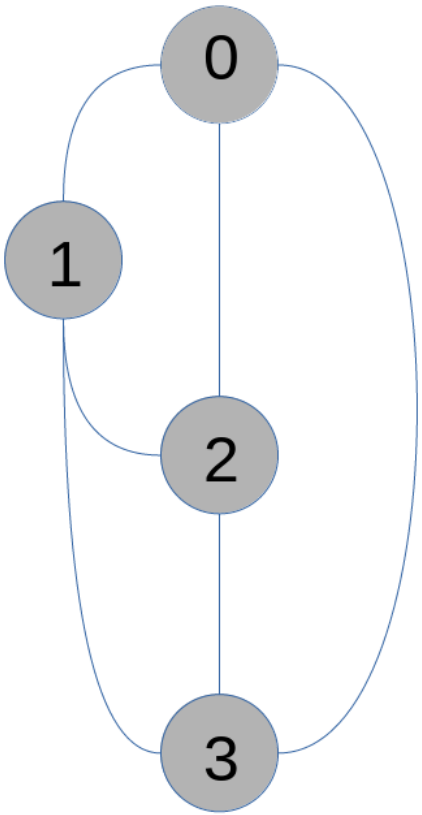
Graphs

Undirected Graph

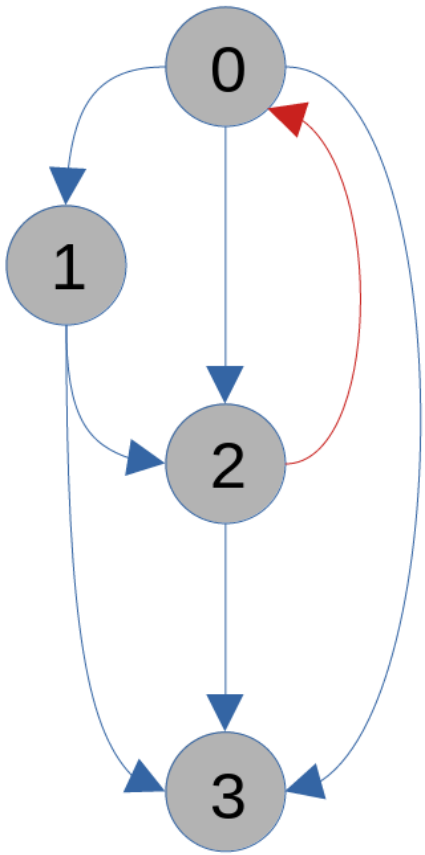


Graphs

Undirected Graph

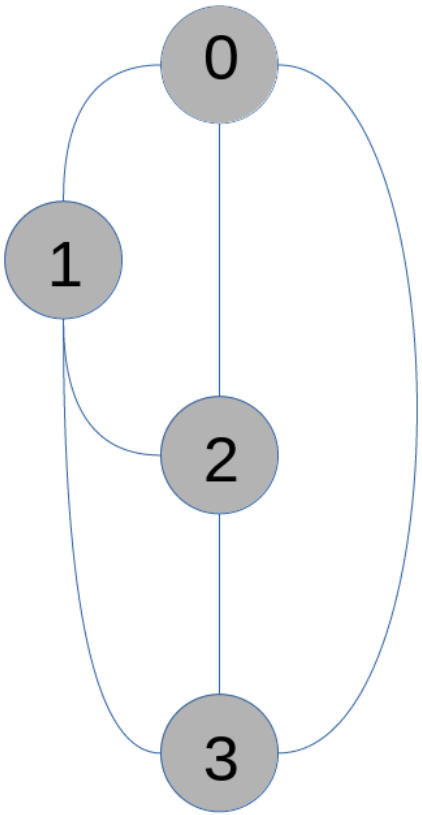


Cyclic Graph

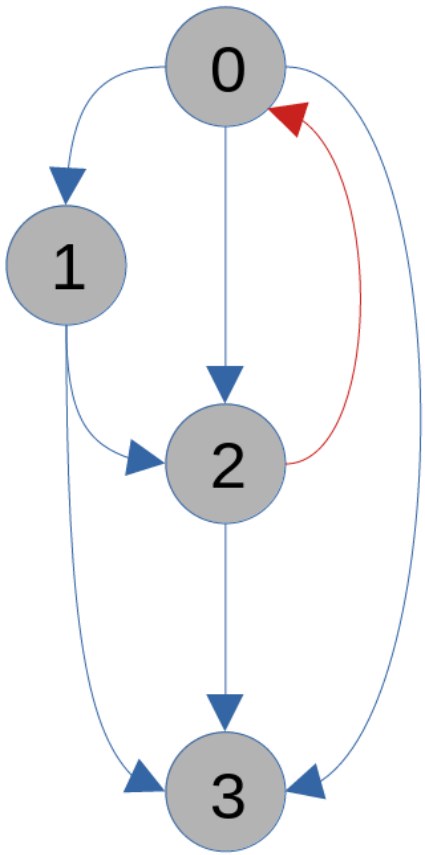


Graphs

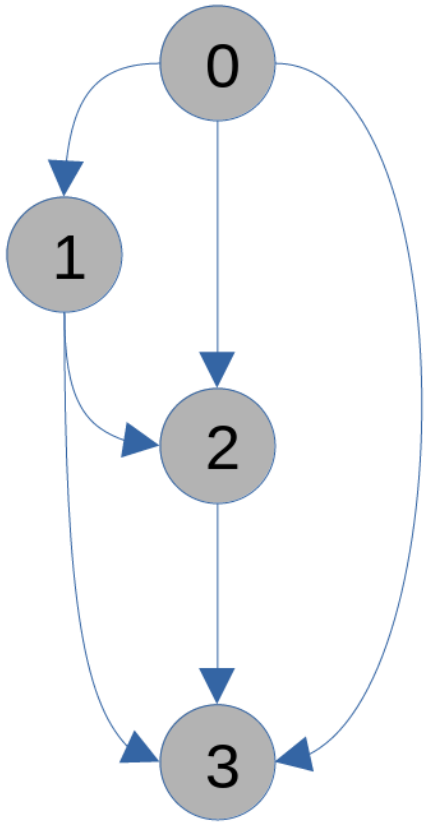
Undirected Graph



Cyclic Graph

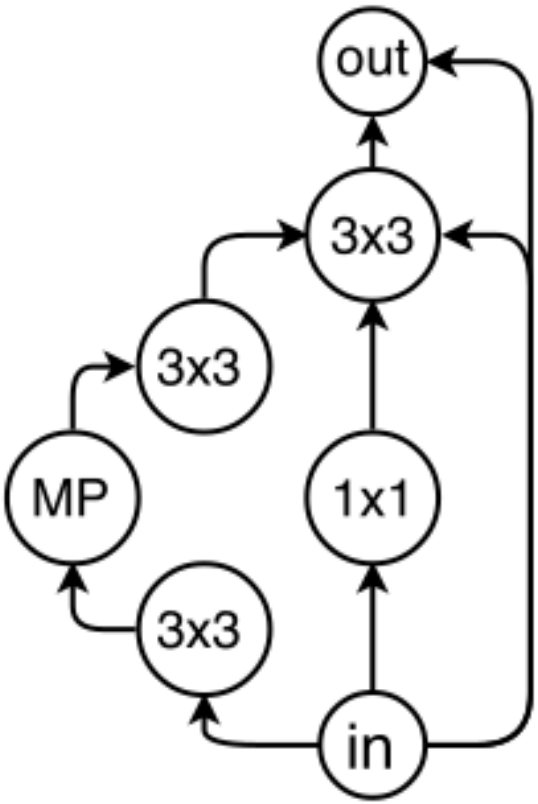


Directed Acyclic Graph



Cell-based

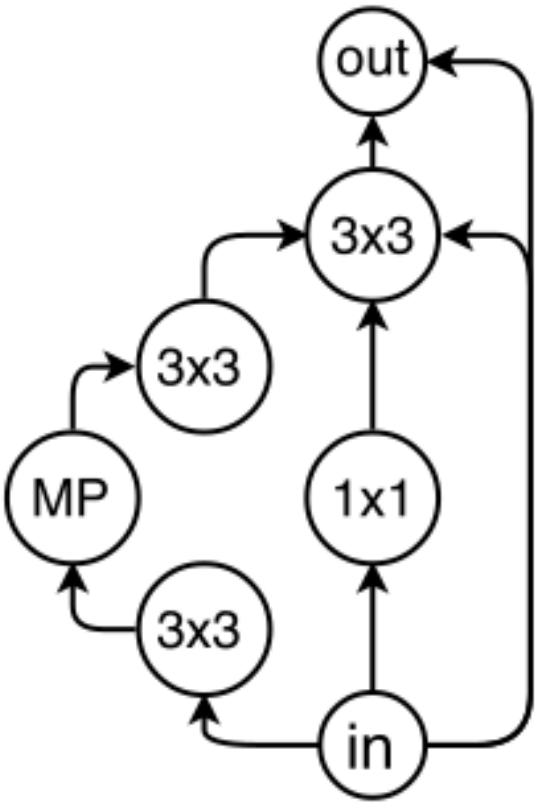
- **Stack**
- Normal & Reduction cell
- Example:



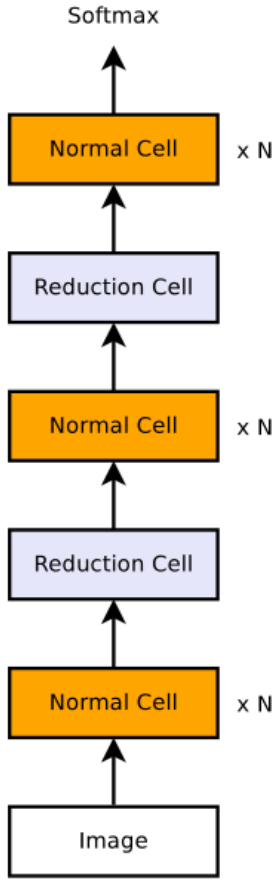
(Ying et al. 2019)

Cell-based

- **Stack**
- Normal & Reduction cell
- Example:



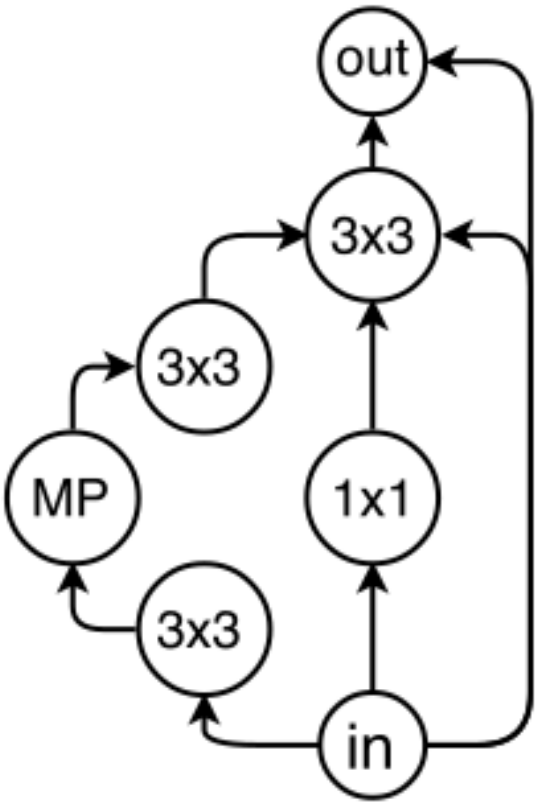
(Ying et al. 2019)



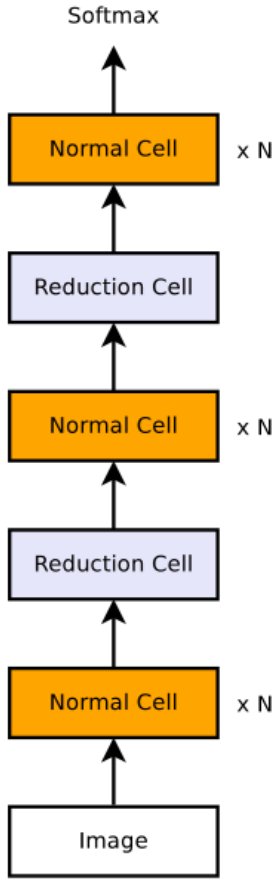
(Zoph et al. 2018)

Cell-based

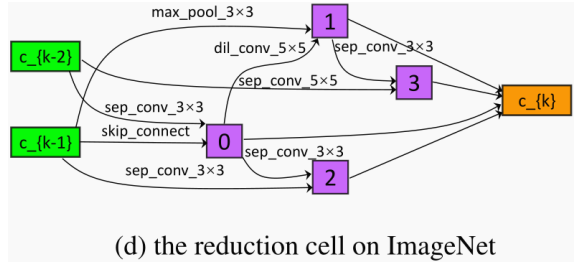
- **Stack**
- Normal & Reduction cell
- Example:



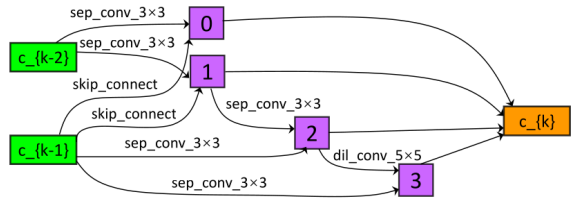
(Ying et al. 2019)



(Zoph et al. 2018)



(d) the reduction cell on ImageNet

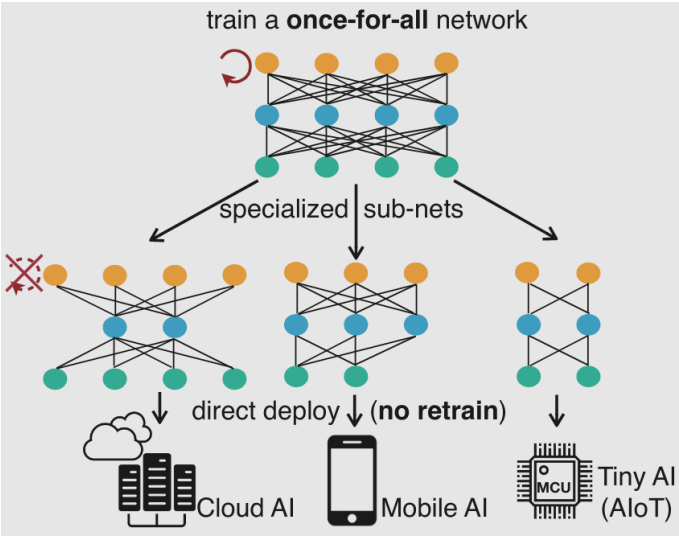


(c) the normal cell found on ImageNet

(Xu et al. 2020)

Macro-level / One-Shot / Global

- **Weights sharing**
- Example:

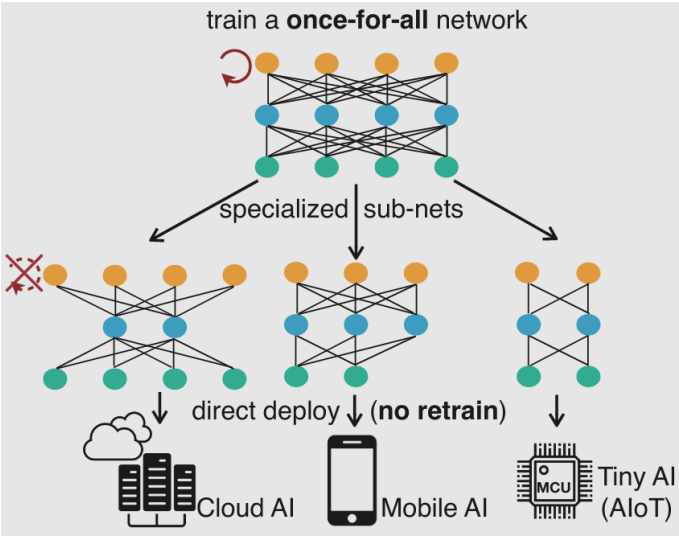


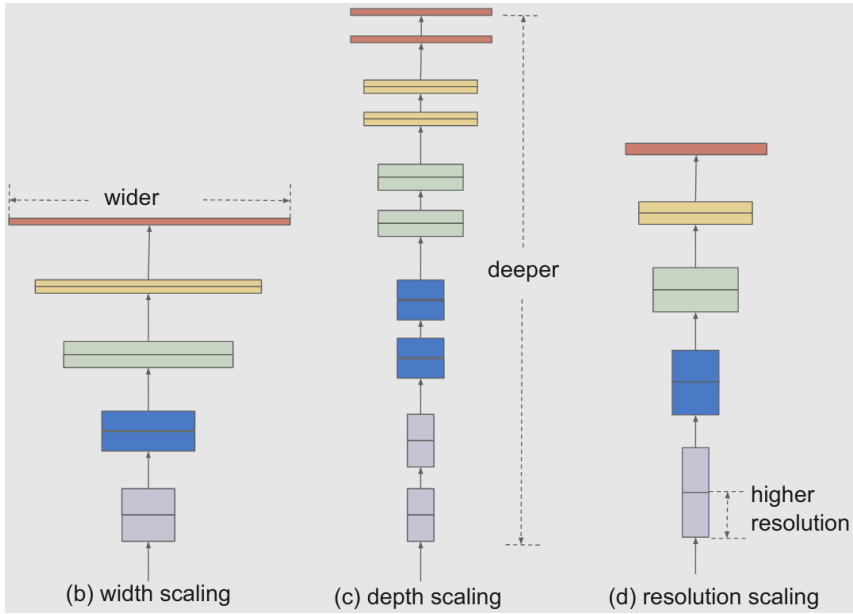
- Depth: {2, 3, 4}
- Width: {3, 4, 6}
- Kernel size: {3, 5, 7}
- Resolution: {128 to 224}

(Cai et al. 2020)

Macro-level / One-Shot / Global

- **Weights sharing**
- Example:



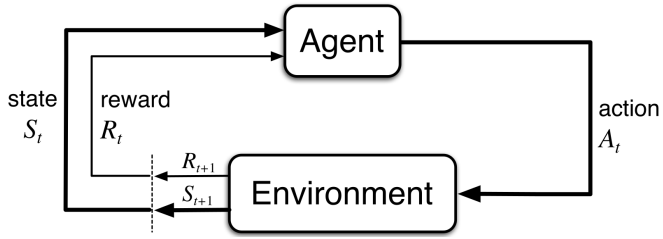


(Tan and Le 2019)

- Depth: {2, 3, 4}
- Width: {3, 4, 6}
- Kernel size: {3, 5, 7}
- Resolution: {128 to 224}

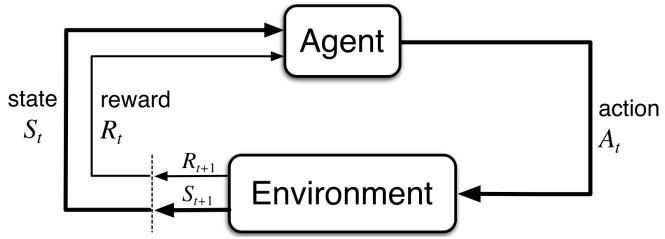
(Cai et al. 2020)

Reinforcement Learning

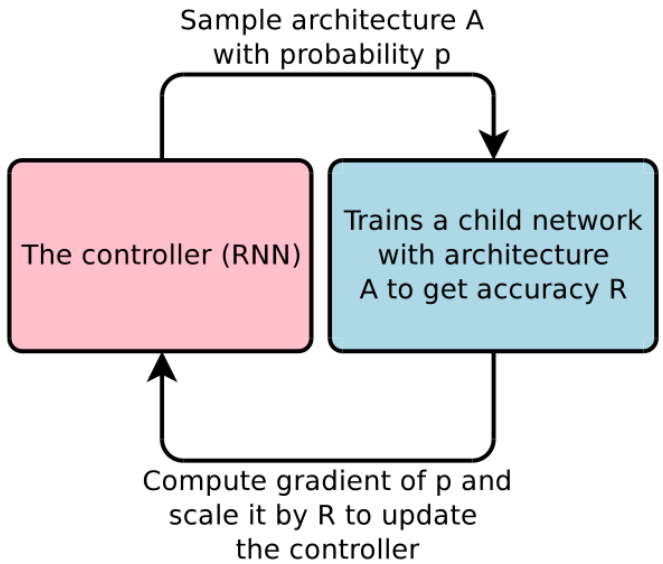


(Sutton and Barto 2018)

Reinforcement Learning

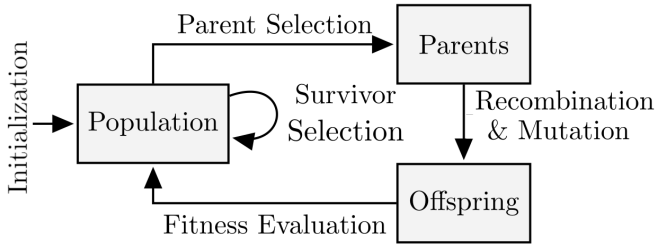


(Sutton and Barto 2018)



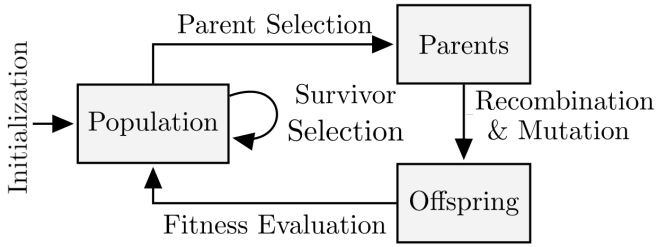
(Zoph and Le 2017)

Evolutionary Algorithms

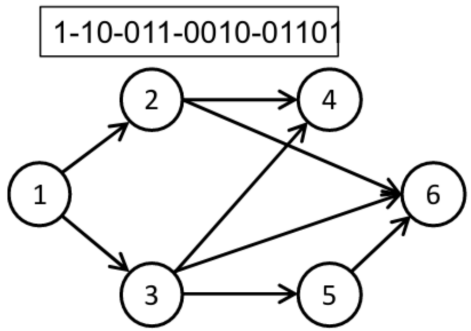


(Wistuba, Rawat, and Pedapati 2019)

Evolutionary Algorithms



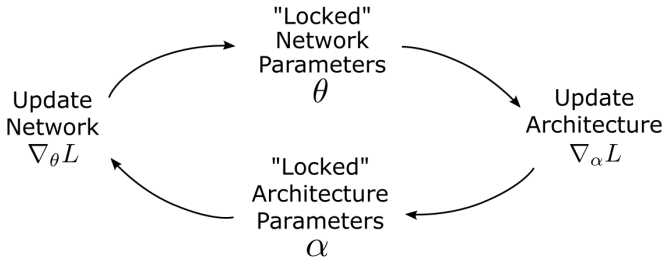
(Wistuba, Rawat, and Pedapati 2019)



	1	2	3	4	5	6
1	0	1	1	0	0	0
2	-1	0	0	1	0	1
3	-1	0	0	1	1	1
4	0	-1	-1	0	0	0
5	0	0	-1	0	0	1
6	0	-1	-1	0	-1	0

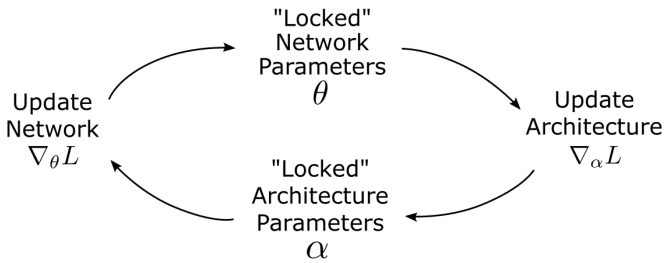
(Lu et al. 2020)

Gradient-based

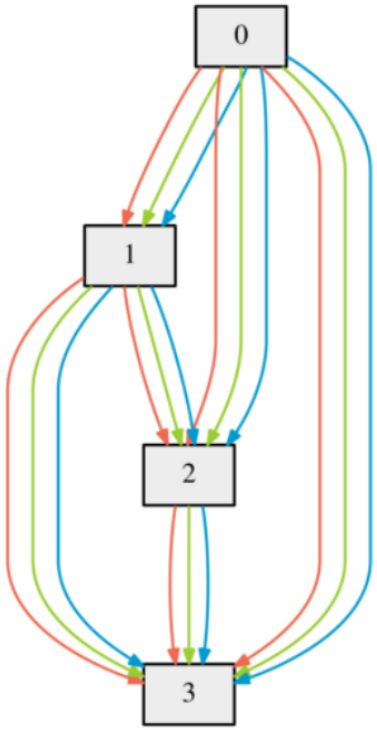


(Green et al. 2019)

Gradient-based



(Green et al. 2019)



(Liu, Simonyan, and Yang 2018)

Paper

Published as a conference paper at ICLR 2019

DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH

Hanxiao Liu* CMU hanxiaol@cs.cmu.com
Karen Simonyan DeepMind simonyan@google.com
Yiming Yang CMU yiming@cs.cmu.edu

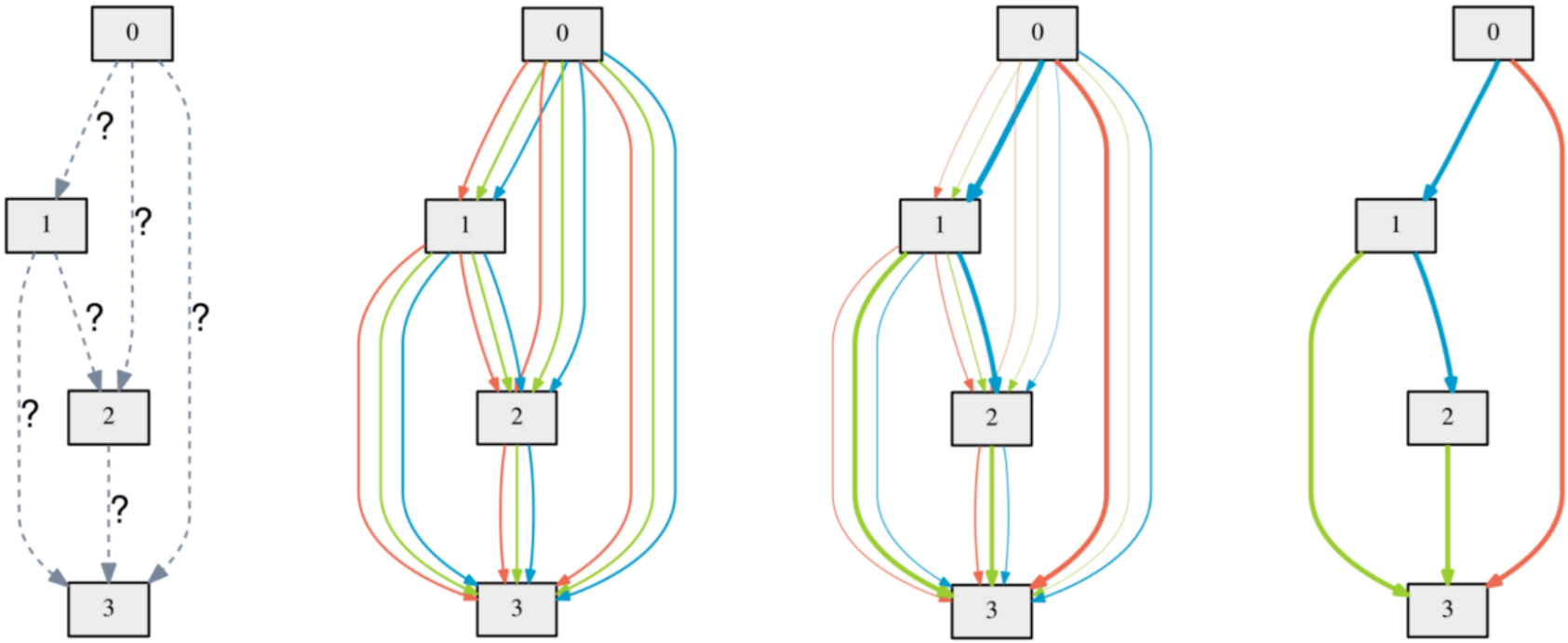
ABSTRACT

This paper addresses the scalability challenge of architecture search by formulating the task in a differentiable manner. Unlike conventional approaches of applying evolution or reinforcement learning over a discrete and non-differentiable search space, our method is based on the continuous relaxation of the architecture representation, allowing efficient search of the architecture using gradient descent. Extensive experiments on CIFAR-10, ImageNet, Penn Treebank and WikiText-2 show that our algorithm excels in discovering high-performance convolutional architectures for image classification and recurrent architectures for language modeling, while being orders of magnitude faster than state-of-the-art non-differentiable techniques. Our implementation has been made publicly available to facilitate further research on efficient architecture search algorithms.

1 INTRODUCTION

Discovering state-of-the-art neural network architectures requires substantial effort of human experts. Recently, there has been a growing interest in developing algorithmic solutions to automate the manual process of architecture design. The automatically searched architectures have achieved highly competitive performance in tasks such as image classification (Zoph & Le, 2017; Zoph et al., 2018; Liu et al., 2018b;a; Real et al., 2018) and object detection (Zoph et al., 2018).

Overview



Description

- Problem:

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ \text{s.t.} \quad & w^*(\alpha) = \text{argmin}_w \ \mathcal{L}_{train}(w, \alpha) \end{aligned}$$

- Formulation:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$

- Solution:

$$\begin{aligned} & \nabla_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ \approx & \nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha) \end{aligned}$$

Solution

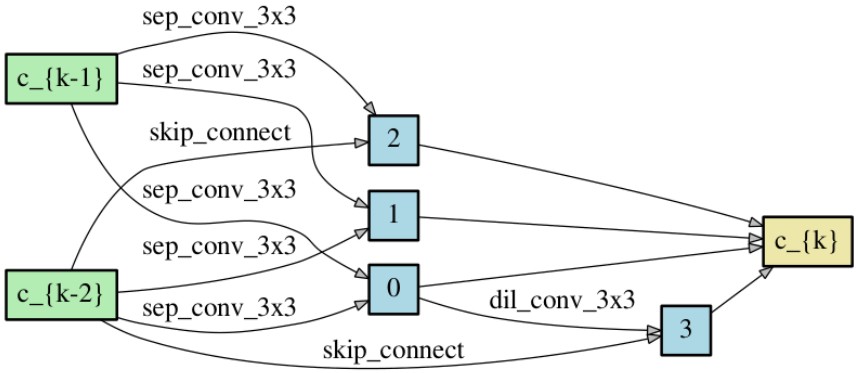


Figure 4: Normal cell learned on CIFAR-10.

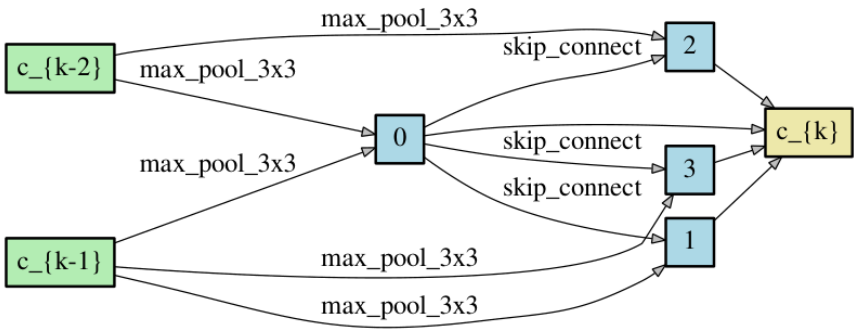


Figure 5: Reduction cell learned on CIFAR-10.

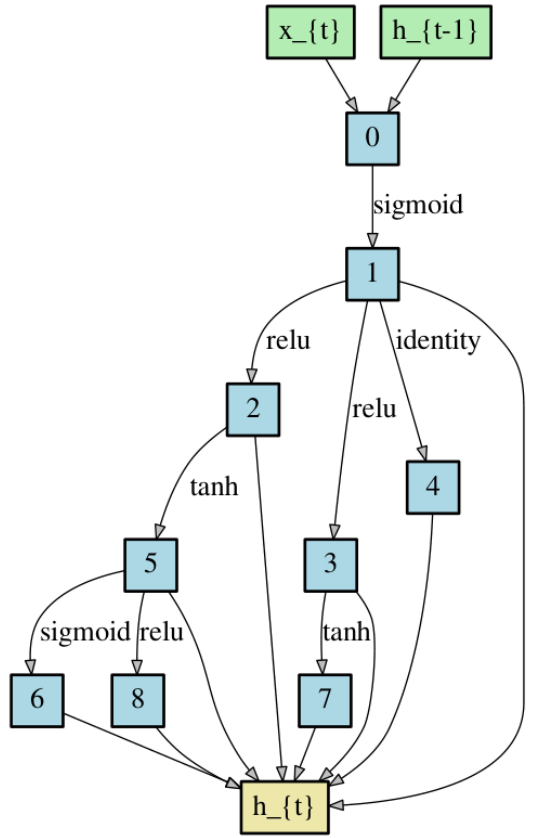


Figure 6: Recurrent cell learned on PTB.

Results on CIFAR-10

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	#ops	Search Method
DenseNet-BC (Huang et al., 2017)	3.46	25.6	–	–	manual
NASNet-A + cutout (Zoph et al., 2018)	2.65	3.3	2000	13	RL
NASNet-A + cutout (Zoph et al., 2018)†	2.83	3.1	2000	13	RL
BlockQNN (Zhong et al., 2018)	3.54	39.8	96	8	RL
AmoebaNet-A (Real et al., 2018)	3.34 ± 0.06	3.2	3150	19	evolution
AmoebaNet-A + cutout (Real et al., 2018)†	3.12	3.1	3150	19	evolution
AmoebaNet-B + cutout (Real et al., 2018)	2.55 ± 0.05	2.8	3150	19	evolution
Hierarchical evolution (Liu et al., 2018b)	3.75 ± 0.12	15.7	300	6	evolution
PNAS (Liu et al., 2018a)	3.41 ± 0.09	3.2	225	8	SMBO
ENAS + cutout (Pham et al., 2018b)	2.89	4.6	0.5	6	RL
ENAS + cutout (Pham et al., 2018b)*	2.91	4.2	4	6	RL
Random search baseline‡ + cutout	3.29 ± 0.15	3.2	4	7	random
DARTS (first order) + cutout	3.00 ± 0.14	3.3	1.5	7	gradient-based
DARTS (second order) + cutout	2.76 ± 0.09	3.3	4	7	gradient-based

Results on PTB

Architecture	Perplexity valid	Perplexity test	Params (M)	Search Cost (GPU days)	#ops	Search Method
Variational RHN (Zilly et al., 2016)	67.9	65.4	23	–	–	manual
LSTM (Merity et al., 2018)	60.7	58.8	24	–	–	manual
LSTM + skip connections (Melis et al., 2018)	60.9	58.3	24	–	–	manual
LSTM + 15 softmax experts (Yang et al., 2018)	58.1	56.0	22	–	–	manual
NAS (Zoph & Le, 2017)	–	64.0	25	1e4 CPU days	4	RL
ENAS (Pham et al., 2018b)*	68.3	63.1	24	0.5	4	RL
ENAS (Pham et al., 2018b)†	60.8	58.6	24	0.5	4	RL
Random search baseline‡	61.8	59.4	23	2	4	random
DARTS (first order)	60.2	57.6	23	0.5	4	gradient-based
DARTS (second order)	58.1	55.7	23	1	4	gradient-based

Paper

MiLeNAS: Efficient Neural Architecture Search via Mixed-Level Reformulation

Chaoyang He[1*] Haishan Ye[2*] Li Shen[3] Tong Zhang[4]
[1]University of Southern California [2]The Chinese University of Hong Kong, Shenzhen
[3]Tencent AI Lab [4]Hong Kong University of Science and Technology
chaoyang.he@usc.edu hsye_cs@outlook.com lshen.lsh@gmail.com tongzhang@tongzhang-ml.org

Abstract

Many recently proposed methods for Neural Architecture Search (NAS) can be formulated as bilevel optimization. For efficient implementation, its solution requires approximations of second-order methods. In this paper, we demonstrate that gradient errors caused by such approximations lead to suboptimality, in the sense that the optimization procedure fails to converge to a (locally) optimal solution. To remedy this, this paper proposes MiLeNAS, a mixed-level reformulation for NAS that can be optimized efficiently and reliably. It is shown that even when using a simple first-order method on the mixed-level formulation, MiLeNAS can achieve a lower validation error for NAS problems. Consequently, architectures obtained by our method achieve consistently higher accuracies than those obtained from bilevel optimization. Moreover, MiLeNAS proposes a framework beyond DARTS. It is upgraded via model size-based search and early stopping strategies to complete the search process in around 5 hours. Extensive experiments within the convolutional architecture search space validate the effectiveness of our approach.

Formally, gradient-based methods can be formulated as a bilevel optimization problem [17]:

$$\min_{\alpha} \quad \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha) \tag{1}$$

$$\text{s.t.} \quad w^*(\alpha) = \arg\min_{w} \ \mathcal{L}_{\text{tr}}(w, \alpha) \tag{2}$$

where w represents the network weight and α determines the neural architecture. $\mathcal{L}_{tr}(w, \alpha)$ and $\mathcal{L}_{val}(w, \alpha)$ denote the losses with respect to training data and validation data with w and α, respectively. Though bilevel optimization can accurately describe the NAS problem, it is difficult to solve, as obtaining $w^*(\alpha)$ in Equation 2 requires one to completely train a network for each update of α. Current methods used in NAS to solve bilevel optimization are heuristic, and $w^*(\alpha)$ in Equation 2 is not satisfied due to first-order or second-order approximation [17, 4]. The second-order approximation has a superposition effect in that it builds upon the one-step approximation of w, causing gradient error and deviation from the true gradient.

Single-level optimization is another method used to solve the NAS problem and is defined as:

$$\min_{w,\alpha} \mathcal{L}_{\text{tr}}(w, \alpha), \tag{3}$$

Description

- Problem:

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha) \\ \text{s.t.} \quad & w^*(\alpha) = \arg\min_{w} \ \mathcal{L}_{\text{tr}}(w, \alpha) \end{aligned}$$

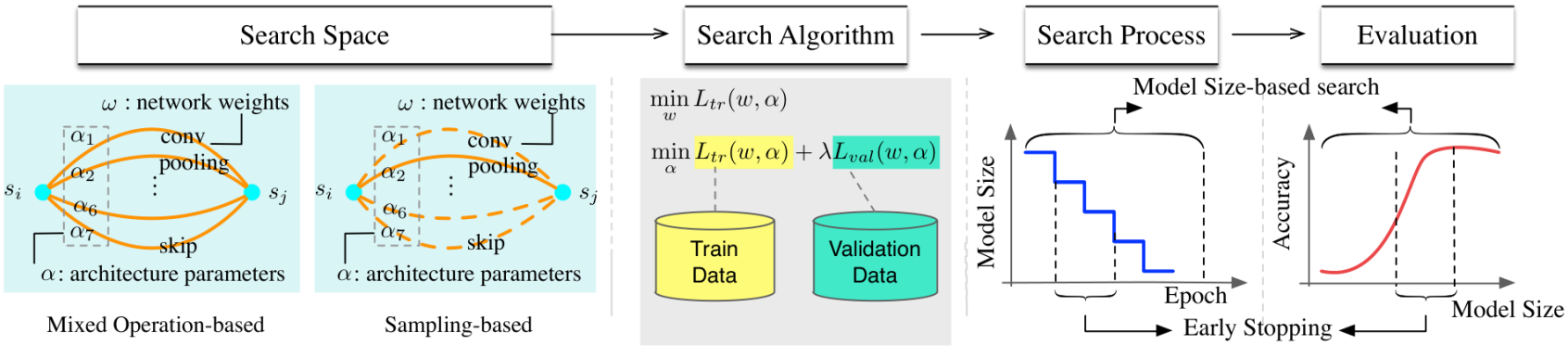
- Formulation:

$$\min_{\alpha, w} \left[\mathcal{L}_{tr}(w^*(\alpha), \alpha) + \lambda \mathcal{L}_{val}(w^*(\alpha), \alpha)\right]$$

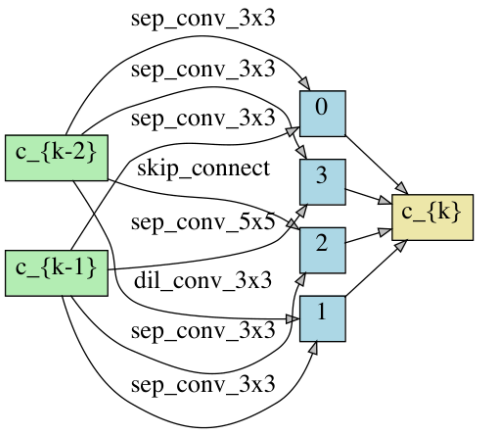
- Solution:

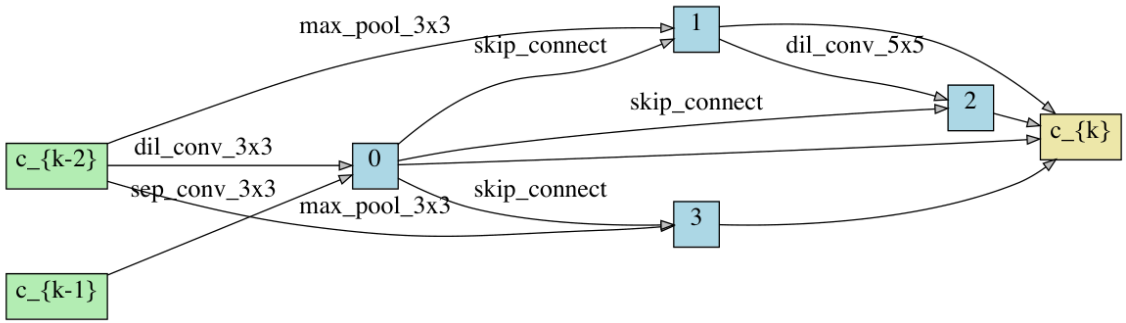
$$\begin{aligned} w &= w - \eta_w \nabla_w \mathcal{L}_{\text{tr}}(w, \alpha), \\ \alpha &= \alpha - \eta_\alpha \left(\nabla_\alpha \mathcal{L}_{\text{tr}}(w, \alpha) + \lambda \nabla_\alpha \mathcal{L}_{\text{val}}(w, \alpha)\right) \end{aligned}$$

Overview

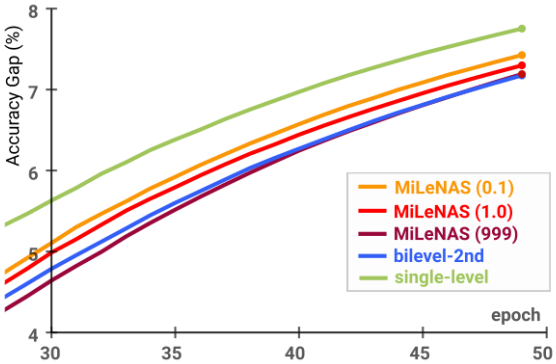

Search Space
Search Algorithm
Search Process
Evaluation
ω : network weights
conv
pooling
skip
α: architecture parameters
Mixed Operation-based
Sampling-based
Train Data
Validation Data
Model Size-based search
Model Size
Epoch
Accuracy
Model Size
Early Stopping

Solution

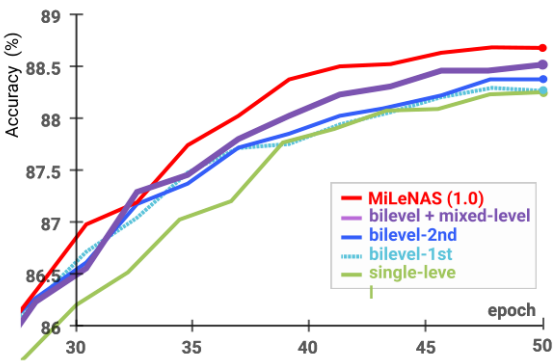

sep_conv_3x3
sep_conv_3x3
sep_conv_3x3
skip_connect
sep_conv_5x5
dil_conv_3x3
sep_conv_3x3
sep_conv_3x3
c_{k-2}
c_{k-1}
0
3
2
1
c_{k}


max_pool_3x3
skip_connect
dil_conv_5x5
skip_connect
dil_conv_3x3
sep_conv_3x3
max_pool_3x3
skip_connect
c_{k-2}
c_{k-1}
0
1
2
3
c_{k}

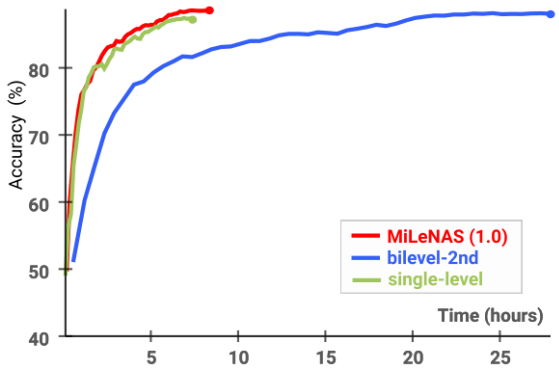
Results



(a) Searching Training and Validation Gap



(b) Searching Validation Accuracy



(c) Searching Time

Results

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
DenseNet-BC [10]	3.46	25.6	-	manual
NASNet-A + cutout [33]	2.65	3.3	2000	RL
BlockQNN [32]	3.54	39.8	96	RL
AmoebaNet-B + cutout [22]	2.55 ± 0.05	2.8	3150	evolution
Hierarchical evolution [16]	3.75 ± 0.12	15.7	300	evolution
PNAS [15]	3.41 ± 0.09	3.2	225	SMBO
ENAS + cutout [21]†	2.89	4.6	0.5	RL
DARTS (second order) [17]	2.76 ± 0.09	3.3	1	gradient-based
SNAS (moderate) [28]	2.85 ± 0.02	2.8	1.5	gradient-based
SNAS (aggressive) [28]	3.10 ± 0.04	2.3	1.5	gradient-based
GDAS [4]	2.82	2.5	0.17	gradient-based
MiLeNAS*	**2.51± 0.11 (best: 2.34)**	**3.87**	0.3	gradient-based
MiLeNAS*	**2.80± 0.04 (best: 2.72)**	**2.87**	0.3	gradient-based
MiLeNAS*	**2.50**	**2.86**	0.3	gradient-based
MiLeNAS*	**2.76**	**2.09**	0.3	gradient-based

Section 3

FNAS

FedNAS

Towards Non-I.I.D. and Invisible Data with FedNAS: Federated Deep Learning via Neural Architecture Search

Chaoyang He Murali Annavaram Salman Avestimehr
University of Southern California
chaoyang.he@usc.edu annavara@usc.edu avestime@usc.edu

Abstract

Federated Learning (FL) has been proved to be an effective learning framework when data cannot be centralized due to privacy, communication costs, and regulatory restrictions. When training deep learning models under an FL setting, people employ the predefined model architecture discovered in the centralized environment. However, this predefined architecture may not be the optimal choice because it may not fit data with non-identical and independent distribution (non-IID). Thus, we advocate automating federated learning (AutoFL) to improve model accuracy and reduce the manual design effort. We specifically study AutoFL via Neural Architecture Search (NAS), which can automate the design process. We propose a Federated NAS (FedNAS) algorithm to help scattered workers collaboratively searching for a better architecture with higher accuracy. We also build a system based on FedNAS. Our experiments on non-IID dataset show that the architecture searched by FedNAS can outperform the manually predefined architecture [1]

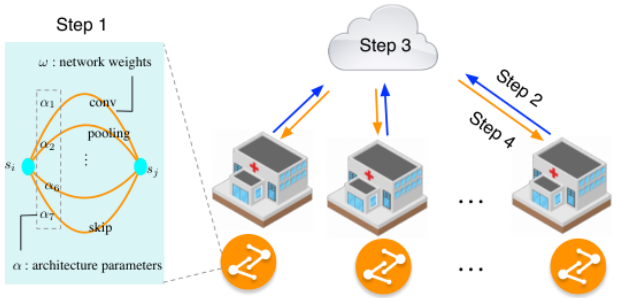
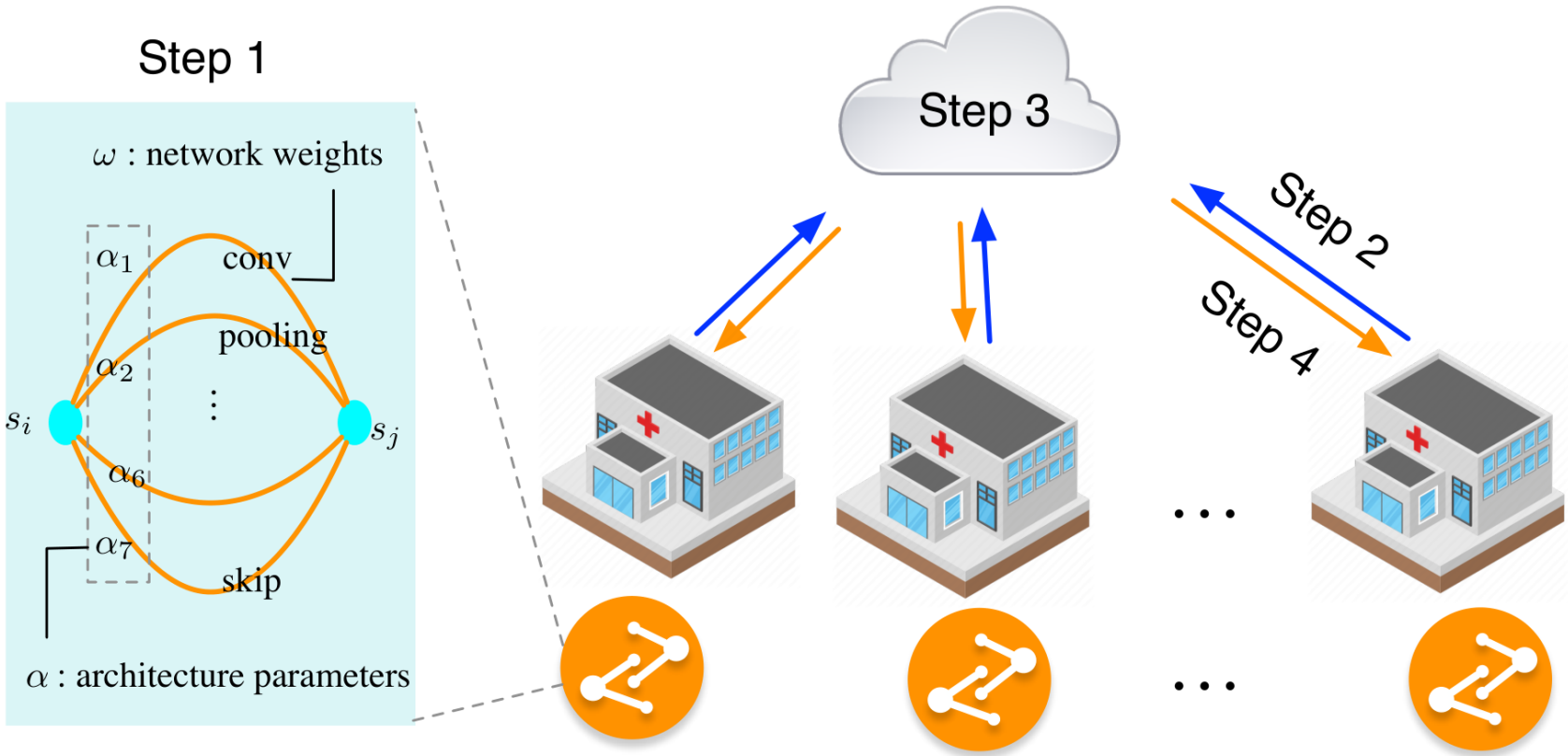


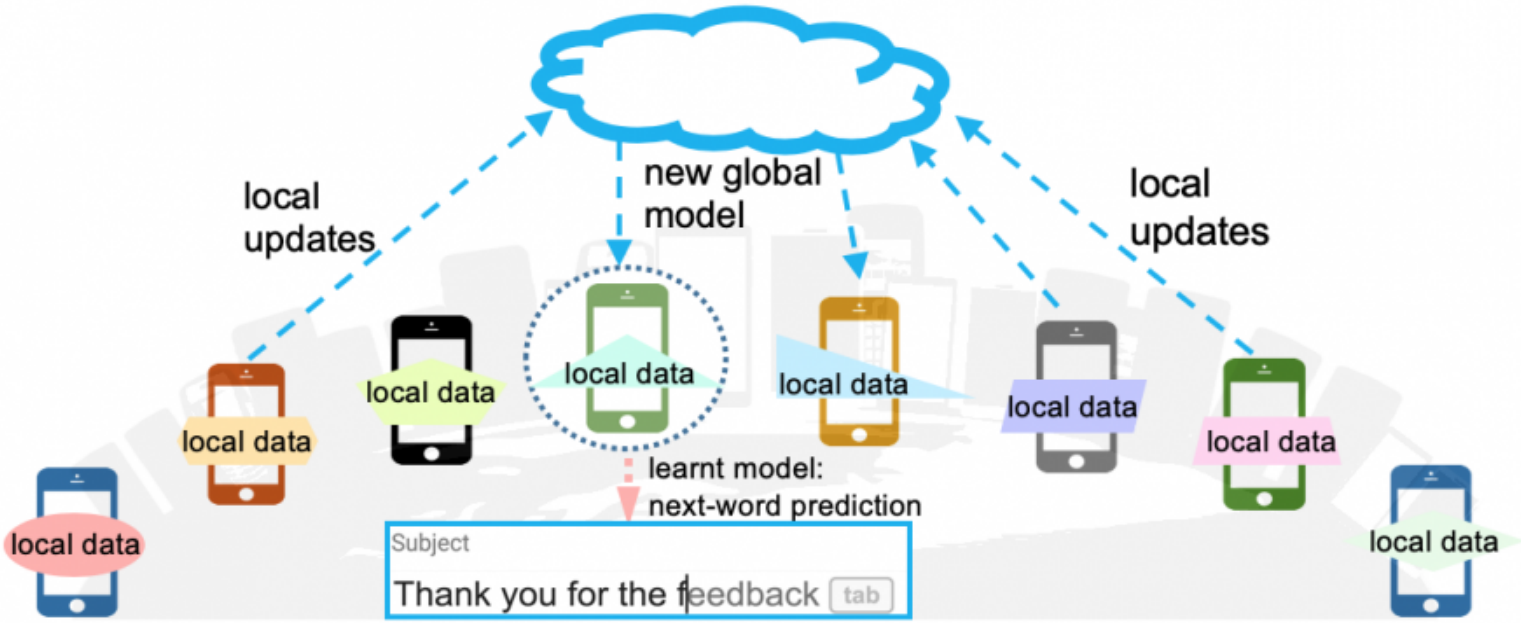
Figure 1: Federated Neural Architecture Search (*step 1*: search locally; *step 2*: sending the gradient of α and ω to the server side; *step 3*: merge gradient to get global α and ω; *step 4*: synchronize updated α and ω to each client.)

due to privacy and confidentiality concerns, high communication and storage costs, protection of intellectual property, regulatory restrictions, and legal constraints. Consequently, a promising approach to solve this problem is Fed-

Overview


Step 1
ω : network weights
α1
conv
pooling
α2
si
sj
α6
α7
skip
α : architecture parameters
Step 3
Step 2
Step 4

Cross-silo & Cross-device



3.1 Problem Definition

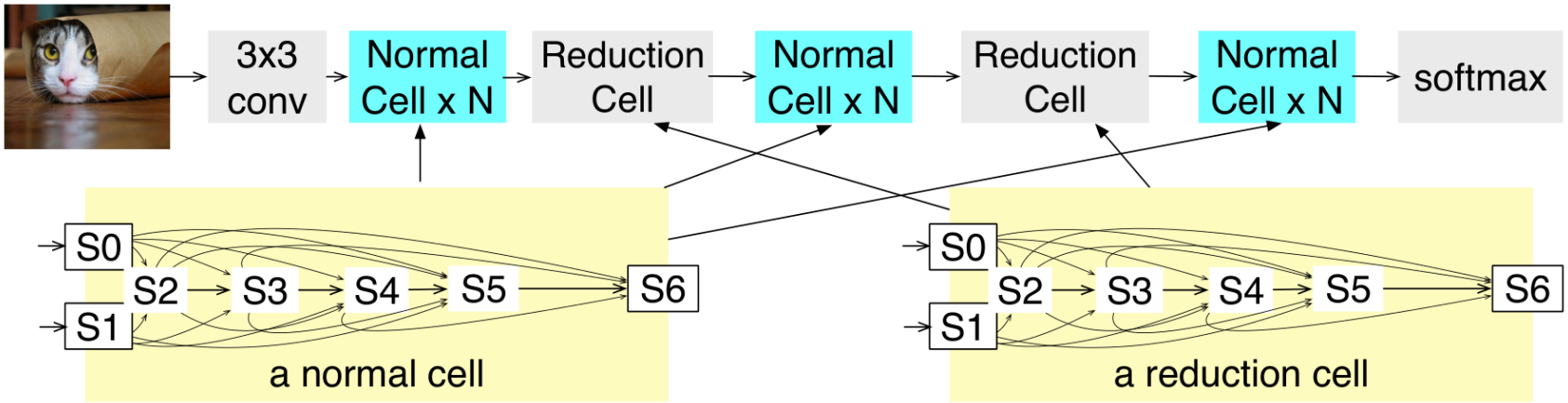
- **Federated Learning**:

$$\min_{w} f(w, \underbrace{\alpha}_{fixed}) \overset{\text{def}}{=} \min_{w} \sum_{k=1}^{K} \frac{N_k}{N} \cdot \frac{1}{N_k} \sum_{i \in \mathcal{D}_k} \ell(x_i, y_i; w, \underbrace{\alpha}_{fixed})$$

- **Federated NAS**:

$$\min_{w,\alpha} f(w, \alpha) \overset{\text{def}}{=} \min_{w,\alpha} \sum_{k=1}^{K} \frac{N_k}{N} \cdot \frac{1}{N_k} \sum_{i \in \mathcal{D}_k} \ell(x_i, y_i; w, \alpha)$$

3.2 Search Space



$$\bar{o}^{(i,j)}(x) = \sum_{k=1}^{d} \underbrace{\frac{\exp(\alpha_k^{(i,j)})}{\sum_{k'=1}^{d} \exp(\alpha_{k'}^{(i,j)})}}_{p_k} o_k(x)$$

3.3 Local Search

- MiLeNAS technique:

$$w = w - \eta_w \nabla_w \mathcal{L}_{\text{tr}}(w, \alpha)$$
$$\alpha = \alpha - \eta_\alpha \left(\nabla_\alpha \mathcal{L}_{\text{tr}}(w, \alpha) + \lambda \nabla_\alpha \mathcal{L}_{\text{val}}(w, \alpha) \right)$$

3.4 Federated Neural Architecture Search

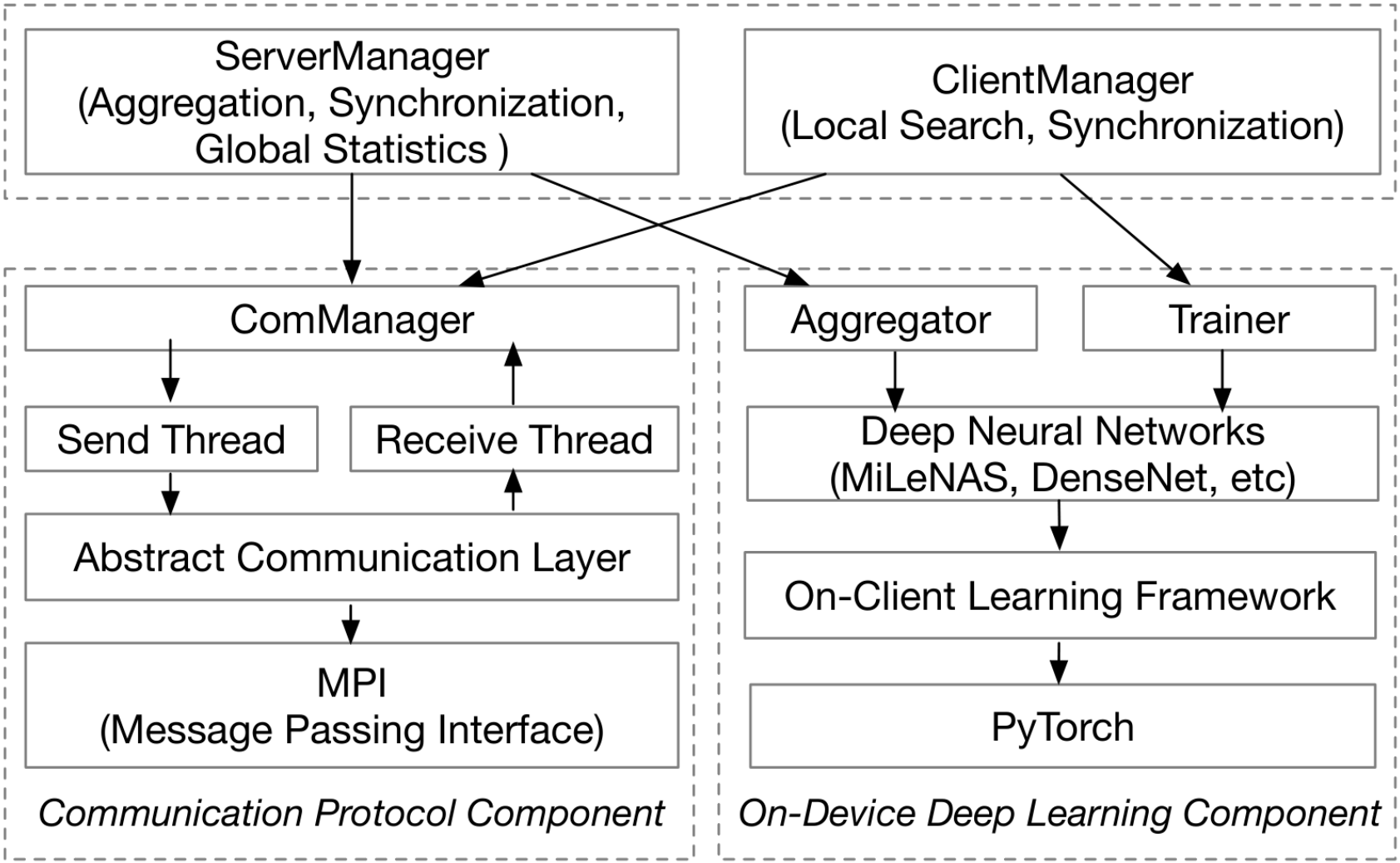
- **Weights update**:

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{N_k}{N} w_{t+1}^k$$

- **Architecture update**:

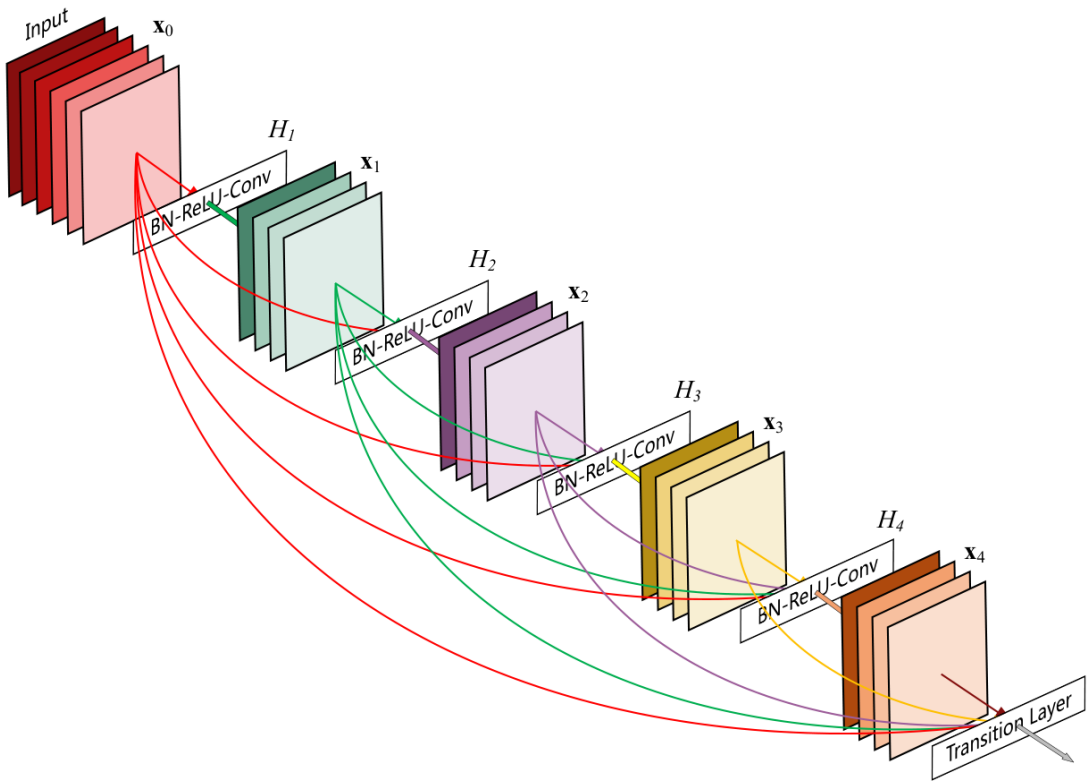
$$\alpha_{t+1} \leftarrow \sum_{k=1}^{K} \frac{N_k}{N} \alpha_{t+1}^k$$

3.5 AutoFL System Design



4.1 Experimental Setup

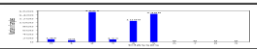
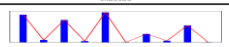
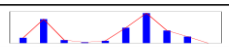
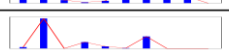
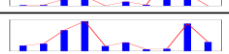
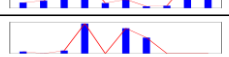
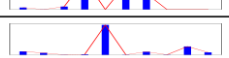
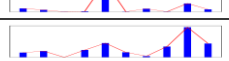
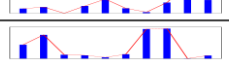
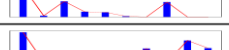
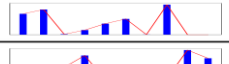
- DenseNet:



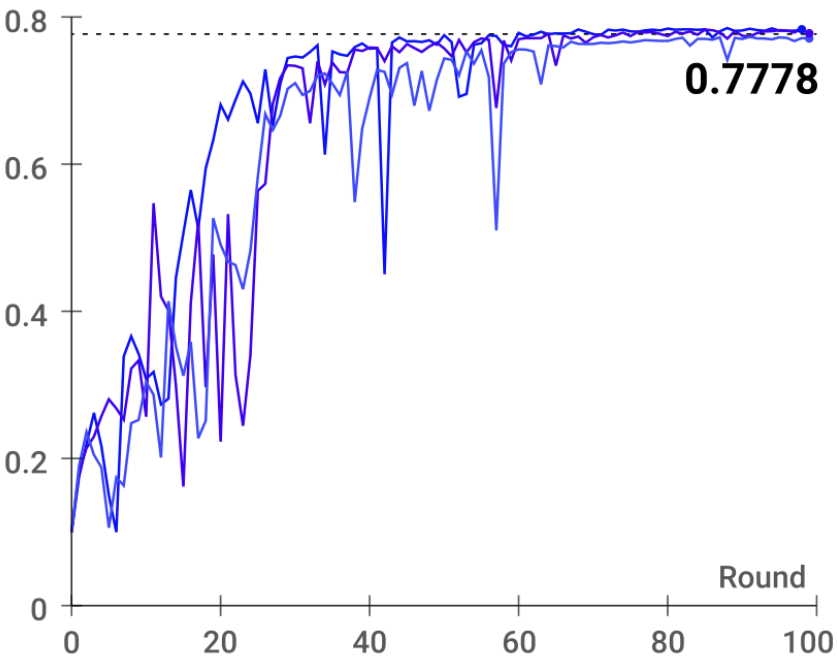
(Huang et al. 2017)

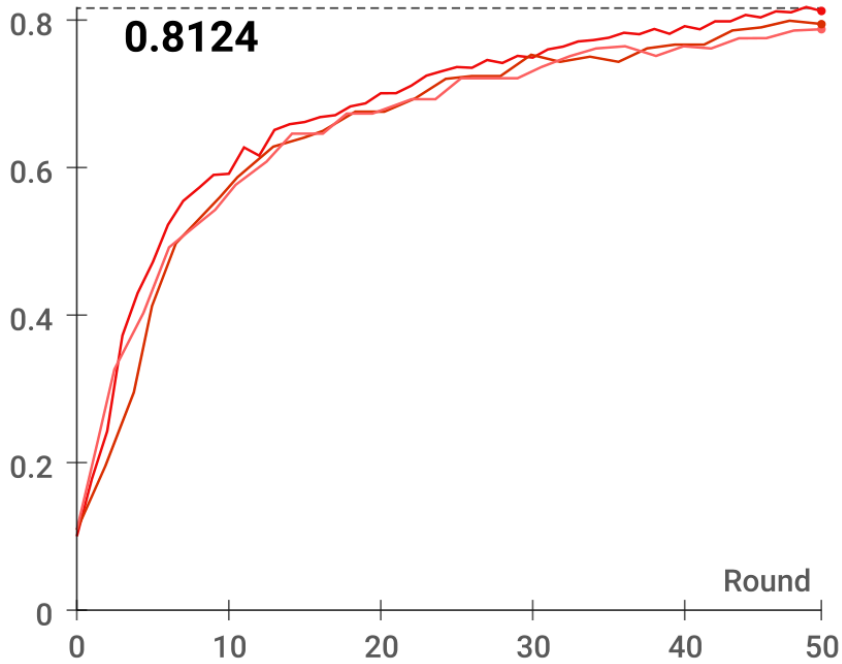
4.1 Experimental Setup

- Dataset: CIFAR-10

Client ID	Numbers of samples in the classes										Distribution
	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	
k=0	144	94	**1561**	133	**1099**	**1466**	**0**	**0**	**0**	**0**	
k=1	327	28	264	16	354	2	100	20	200	3	
k=2	6	6	641	1	255	4	1	2	106	**1723**	
k=3	176	792	100	28	76	508	991	416	215	**0**	
k=4	84	**1926**	1	408	133	24	771	**0**	**0**	**0**	
k=5	41	46	377	541	7	235	54	**1687**	666	**0**	
k=6	134	181	505	720	123	210	44	58	663	221	
k=7	87	2	131	**1325**	**1117**	704	**0**	**0**	**0**	**0**	
k=8	178	101	5	32	**1553**	10	163	9	437	131	
k=9	94	125	**0**	147	287	100	23	217	608	279	
k=10	379	649	106	90	35	119	807	819	3	85	
k=11	**1306**	55	681	227	202	34	**0**	648	**0**	**0**	
k=12	**1045**	13	53	6	77	70	482	7	761	494	
k=13	731	883	15	161	387	552	4	**1051**	**0**	**0**	
k=14	4	97	467	899	**0**	407	50	64	**1098**	797	
k=15	264	2	93	266	412	142	806	2	243	**1267**	

4.2 Results on non-IID dataset





4.3 Evaluation of the Efficiency

Method	Search Time	Parameter Size	Hyperparameter
FedAvg (single)	$>$ 3 days	-	rounds = 100 local epochs=20 batch size=64
FedAvg (distributed)	12 hours	20.01M	
FedNAS (single)	33 hours	-	rounds = 50 local epochs=5 batch size=64
FedNAS (distributed)	$<$ **5 hours**	**1.93M**	

5. Future Works

- Local NAS under Resource Constraint
- One Stage NAS
- Asynchronous Aggregation
- Personalized NAS

6. Conclusion

We study automating federated learning (AutoFL) via Neural Architecture Search (NAS) by proposing a Federated NAS (FedNAS) algorithm that can help scatter workers collaboratively searching for a better architecture with higher accuracy. We build an AutoFL system based on FedNAS. Our experiments on the non-IID dataset show that the architecture searched by FedNAS can outperform FedAvg training on the manually designed architecture.

References I

Cai, Han, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2020. "Once for All: Train One Network and Specialize It for Efficient Deployment." https://iclr.cc/virtual_2020/poster_HylxE1HKwS.html.

Green, Sam, Craig M. Vineyard, Ryan Helinski, and Çetin Kaya Koç. 2019. "RAPDARTS: Resource-Aware Progressive Differentiable Architecture Search." https://doi.org/10.48550/ARXIV.1911.05704.

He, Chaoyang, Murali Annavaram, and Salman Avestimehr. 2021. "Towards Non-I.I.D. And Invisible Data with FedNAS: Federated Deep Learning via Neural Architecture Search." arXiv. https://doi.org/10.48550/arXiv.2004.08546.

Huang, Gao, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. "Densely Connected Convolutional Networks," 4700–4708. https:

References II

//openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html.

Hutter, Frank, Lars Kotthoff, and Joaquin Vanschoren, eds. 2019. *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5.

Liu, Hanxiao, Karen Simonyan, and Yiming Yang. 2018. "DARTS: Differentiable Architecture Search." https://doi.org/10.48550/ARXIV.1806.09055.

Lu, Zhichao, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. 2020. "NSGA-Net: Neural Architecture Search Using Multi-Objective Genetic Algorithm (Extended Abstract)," 5:4750–54. https://doi.org/10.24963/ijcai.2020/659.

References III

Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book.

Tan, Mingxing, and Quoc Le. 2019. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." In *Proceedings of the 36th International Conference on Machine Learning*, 6105–14. PMLR. https://proceedings.mlr.press/v97/tan19a.html.

Wistuba, Martin, Ambrish Rawat, and Tejaswini Pedapati. 2019. "A Survey on Neural Architecture Search." arXiv. https://doi.org/10.48550/arXiv.1905.01392.

Xu, Yuhui, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. 2020. "PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search." https://iclr.cc/virtual_2020/poster_BJlS634tPr.html.

Yang, Qiang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. "Federated Machine Learning: Concept and Applications." *Acm Trans.*

References IV

Intell. Syst. Technol. 10 (2, 2): 12:1–12:19. https://doi.org/10.1145/3298981.

Ying, Chris, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. 2019. "NAS-Bench-101: Towards Reproducible Neural Architecture Search." In *Proceedings of the 36th International Conference on Machine Learning*, 7105–14. PMLR. https://proceedings.mlr.press/v97/ying19a.html.

Zoph, Barret, and Quoc Le. 2017. "Neural Architecture Search with Reinforcement Learning." https://openreview.net/forum?id=r1Ue8Hcxg.

Zoph, Barret, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. "Learning Transferable Architectures for Scalable Image Recognition." In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8697–8710. https://doi.org/10.1109/CVPR.2018.00907.