Firefly Neural Architecture Descent: a General Approach for Growing Neural Networks

Lemeng Wu, Bo Liu, Peter Stone, Qiang Liu
University of Texas at Austin
(NeurIPS 2020)

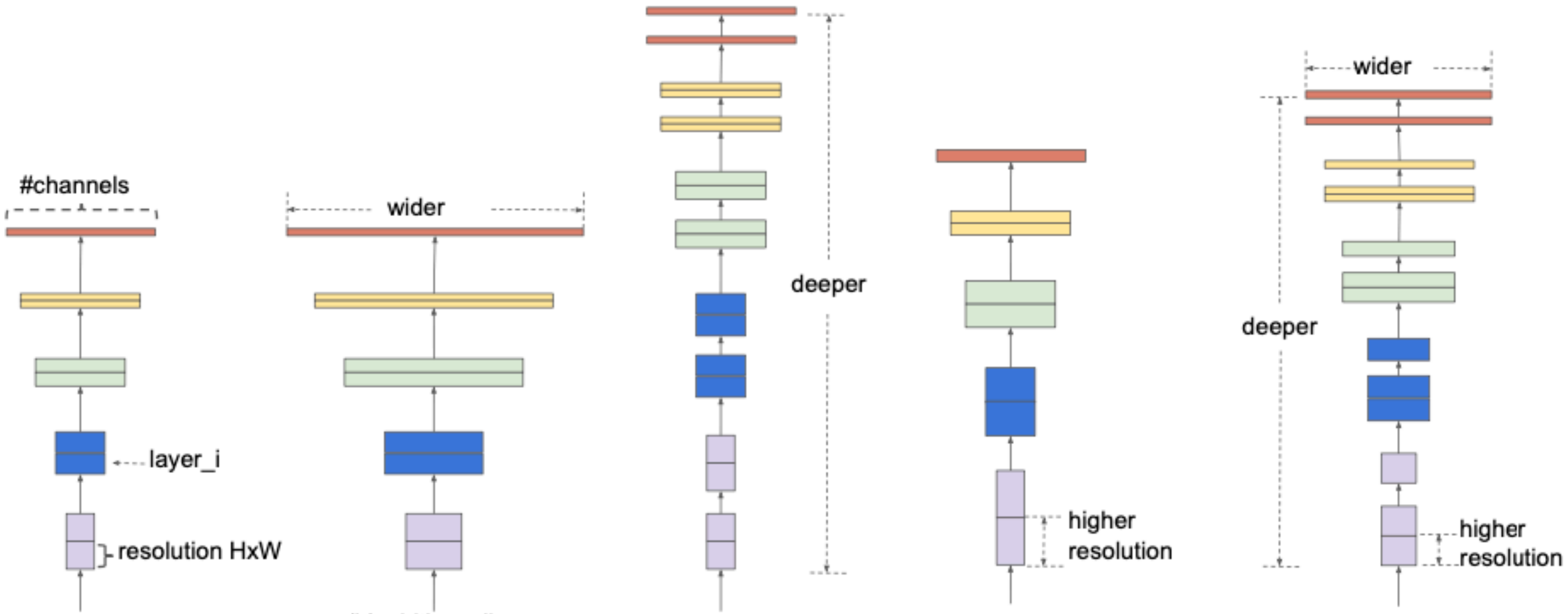
Reporter: Yinghuan Zhang, Mar 9th 2021

Background

Model Scaling:

ResNet can be scaled down (e.g., ResNet-18) or up (e.g., ResNet-200)

Scale a ConvNet for different resource constraints



(a) baseline (b) width scaling (c) depth scaling (d) resolution scaling (e) compound scaling

Background

Model Scaling:

ResNet can be scaled down (e.g., ResNet-18) or up (e.g., ResNet-200)

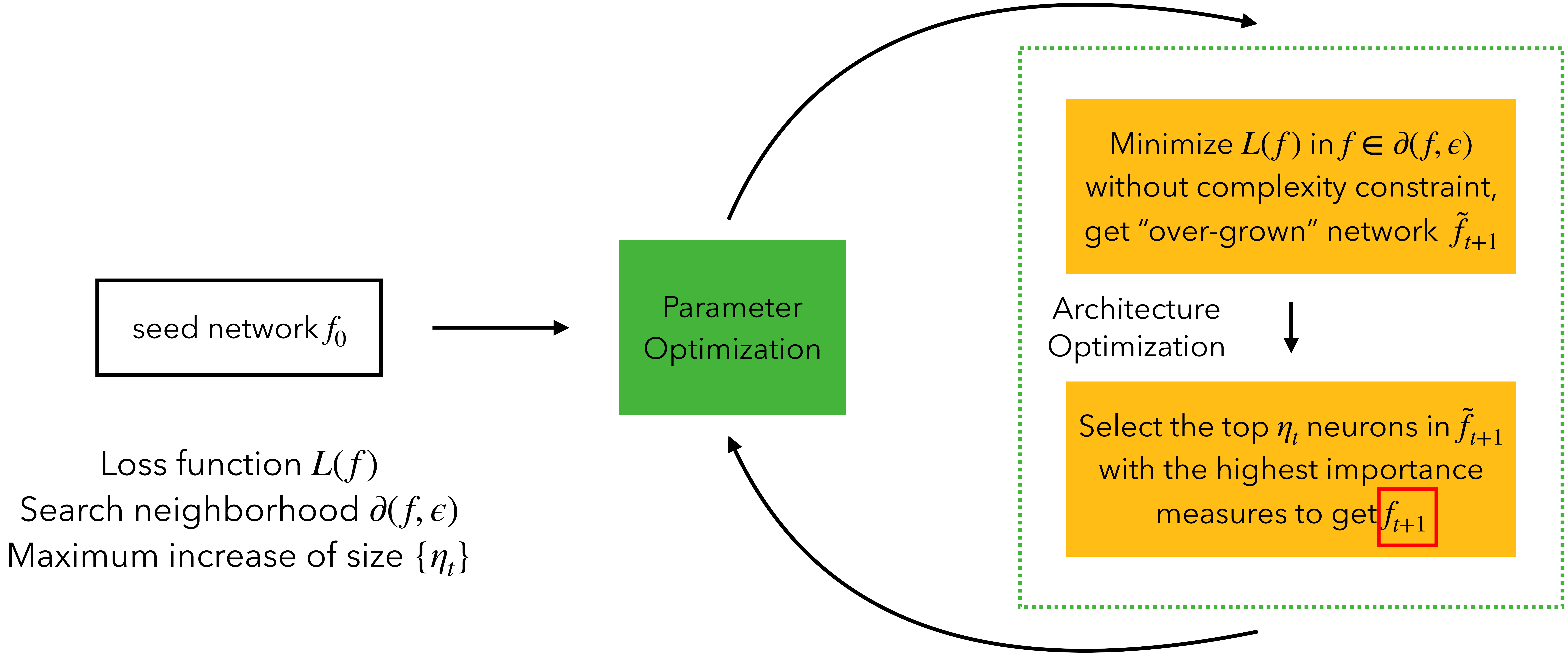
Scale a ConvNet for different resource constraints

Challenges:

- Grow strategy
- Accuracy drop brought by newly initialized parameters in grown network

Firefly Neural Architecture Descent — a general framework for progressively and dynamically growing neural networks to jointly optimize the networks' parameters and architectures

Framework Overview



Framework Overview

Neighborhood:

$$f(x) = f_t(x) + O(\epsilon), \text{ for } \forall x \in \partial(f_t, \epsilon)$$

Goal:

$$\arg\min_f \{L(f) \ s.t. \ f \in \Omega, \boxed{C(f) \leq \eta}\}$$

⬇️

$$f_{t+1} = \arg\min_f \{L(f) \ s.t. \ f \in \partial(f_t, \epsilon), \boxed{C(f) \leq C(f_t) + \eta_t}\}$$

Minimize $L(f)$ in $f \in \boxed{\partial(f, \epsilon)}$ without complexity constraint, get "over-grown" network $\boxed{\tilde{f}_{t+1}}$

Architecture Optimization ↓

Select the top η_t neurons in $\tilde{f}_{t+1}$ with the highest importance measures to get f_{t+1}

Framework Overview

Optimization Goal:

$$f_{t+1} = \arg\min_{f}\{L(f)\ s.t.\ f \in \partial(f_t, \epsilon),\ C(f) \leq C(f_t) + \eta_t\}$$

Goal in Architecture Optimization:

- Instantiate neighborhood $\partial(f_t, \epsilon)$
- Solve the optimization in f_{t+1}

Minimize $L(f)$ in $f \in \boxed{\partial(f, \epsilon)}$ without complexity constraint, get "over-grown" network $\boxed{\tilde{f}_{t+1}}$

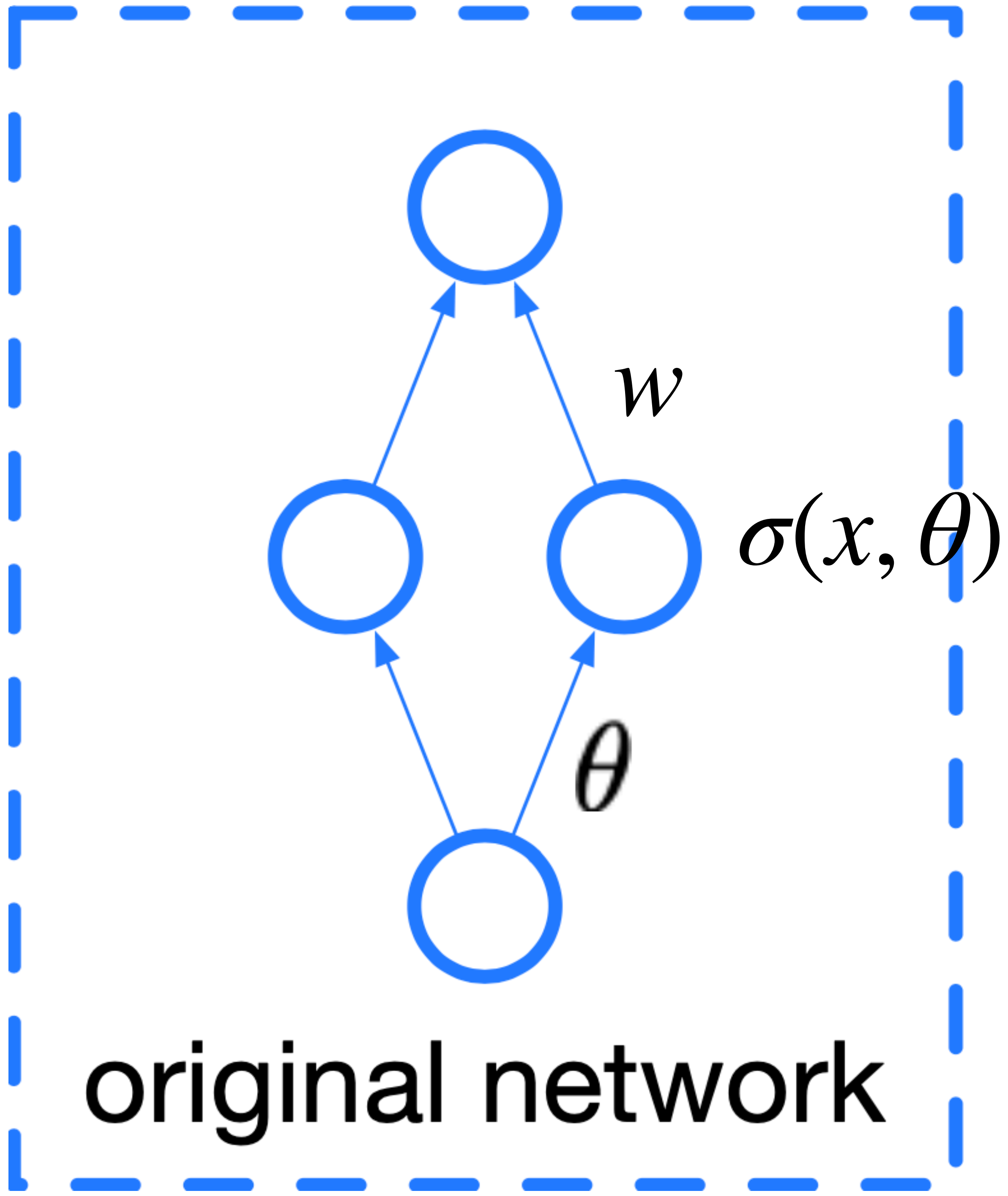
Architecture Optimization $\downarrow$

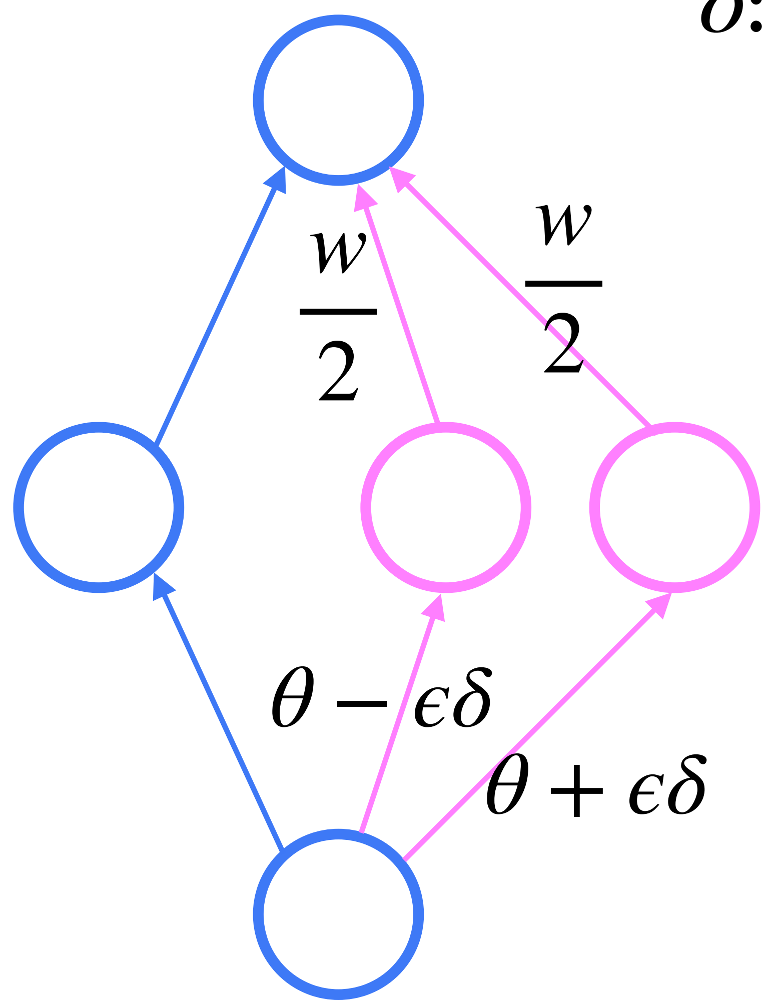
Select the top η_t neurons in $\tilde{f}_{t+1}$ with the highest importance measures to get f_{t+1}

Growing Network Width

ϵ: step size, indicating how much the network is changed

δ: update direction, indicating how the network is changed





$$\sigma_1(x, \theta - \epsilon\delta) \approx \sigma(x, \theta)$$
$$\sigma_2(x, \theta + \epsilon\delta) \approx \sigma(x, \theta)$$

$$\boxed{\frac{w}{2} \cdot \sigma_1 + \frac{w}{2} \cdot \sigma_2} \approx \frac{w}{2} \cdot \sigma + \frac{w}{2} \cdot \sigma = \boxed{w \cdot \sigma(x, \theta)}$$

Splitting Existing Neurons

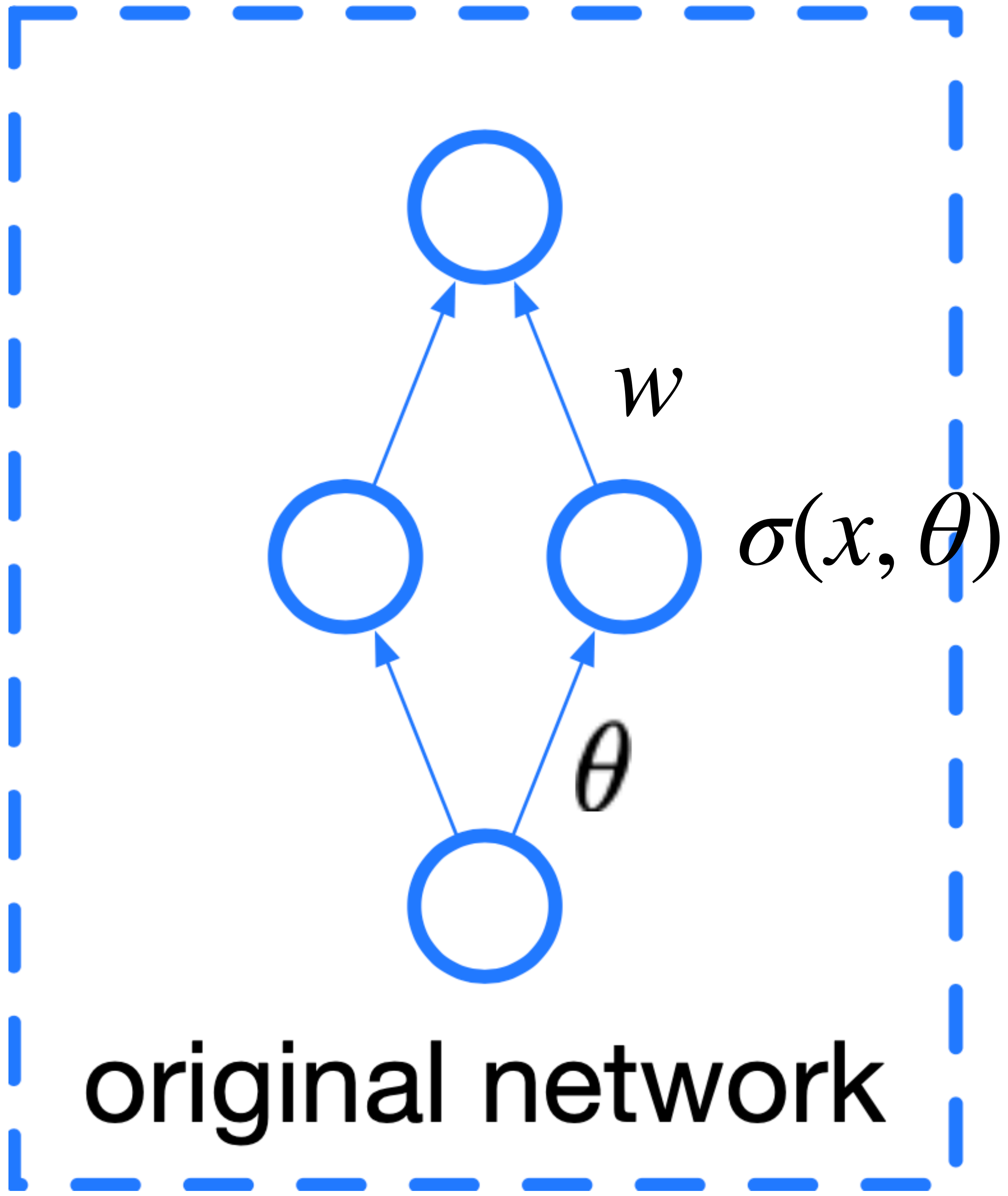
Growing New Neurons

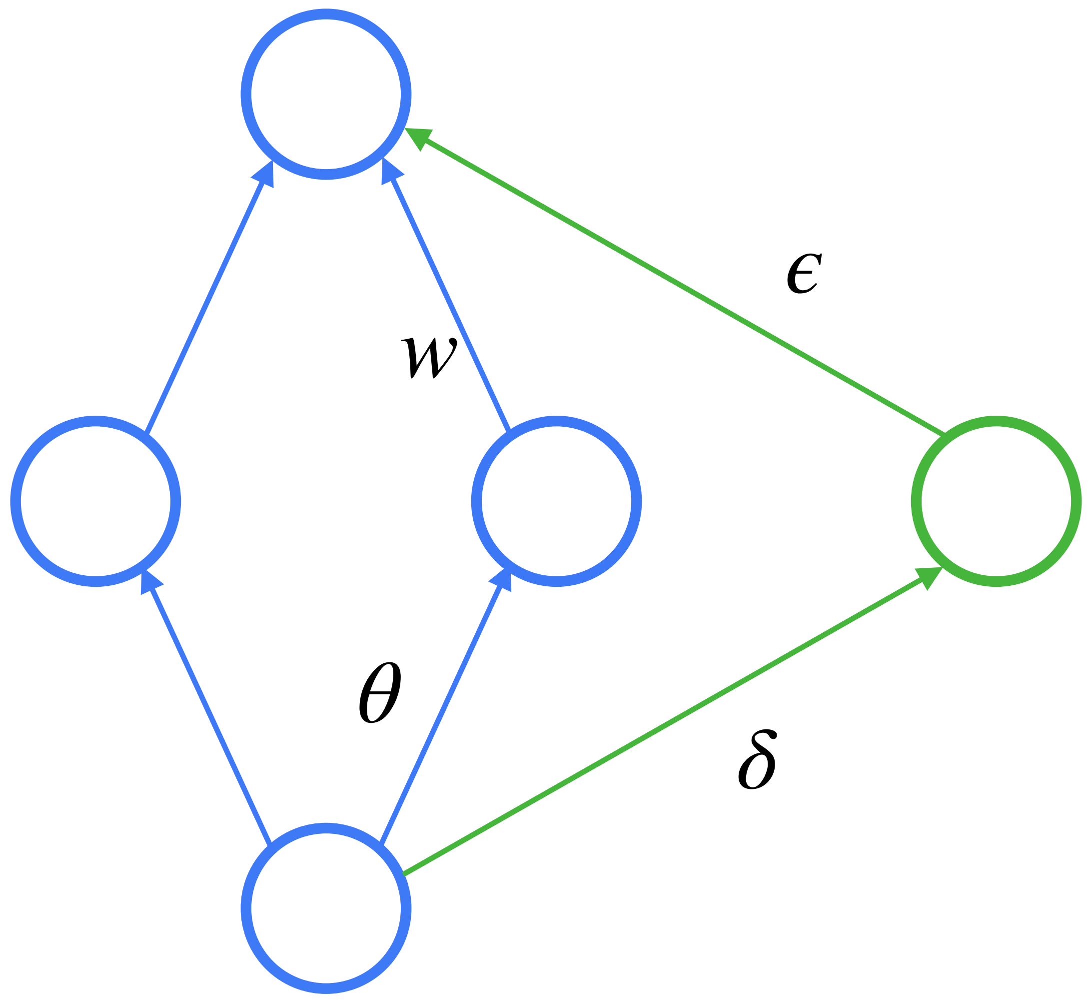
$$f_{new} \approx f_{orig}$$

Growing Network Width

ϵ: make sure the new network is close to original network

δ: a trainable parameter of the new neuron





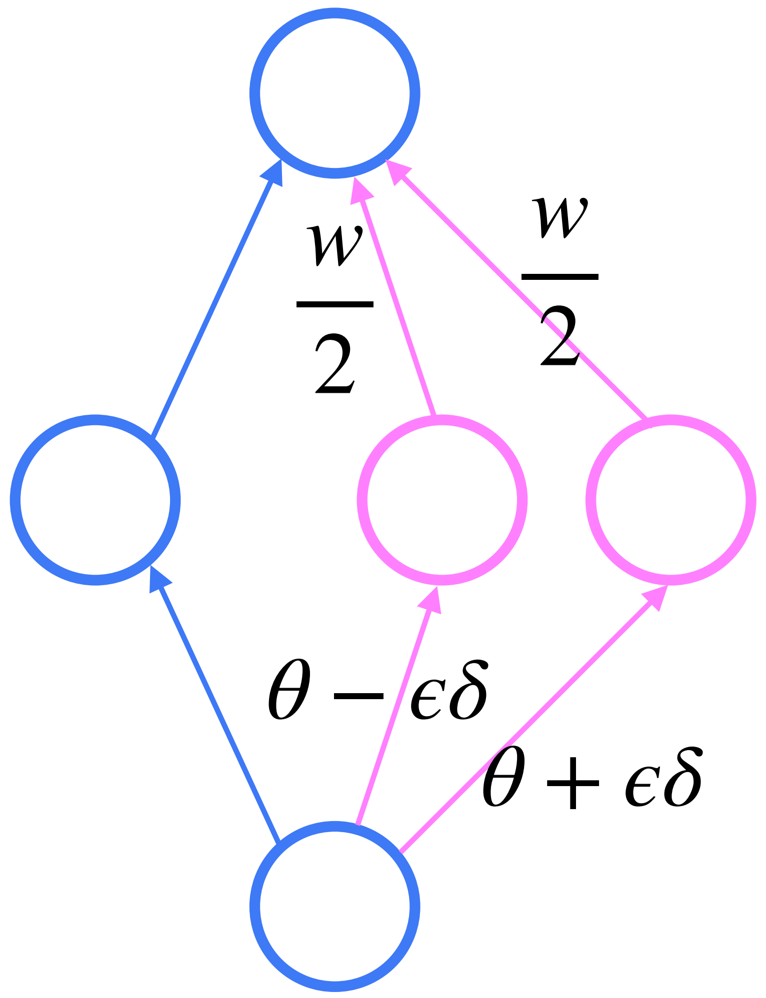
$$\boxed{\epsilon \cdot \sigma(x, \delta)}$$

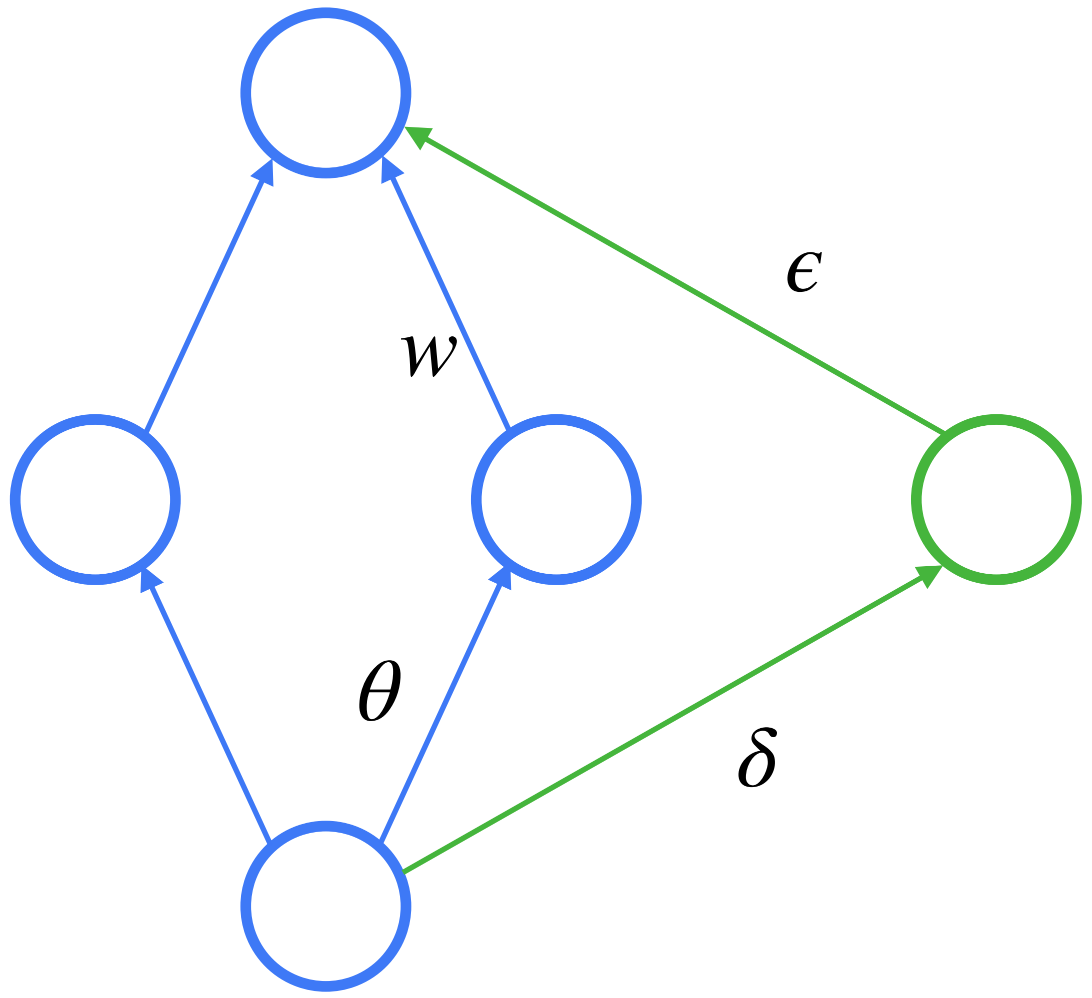
$f_{new} \approx f_{orig}$

Splitting Existing Neurons

Growing New Neurons

Growing Network Width





Overall, to grow $f_t(x) = \sum_i \sigma(x; \theta_i)$ wider,

the neighborhood set $\partial(f_t, \epsilon)$ can include functions of the form

$$f_{\epsilon,\delta}(x) = \sum_{i=1}^{m} \frac{1}{2}(\sigma(x, \theta_i + \epsilon_i\delta_i) + \sigma(x, \theta_i - \epsilon_i\delta_i)) + \sum_{i=m+1}^{m+m'} \epsilon_i\sigma(x, \delta_i)$$

Up to m neurons to split

Up to m' neurons to add

Goal in Architecture Optimization:

- Instantiate neighborhood $\partial(f_t, \epsilon)$ ✅
- Solve the optimization in f_{t+1}

Growing Network Width

Goal:

$$f_{t+1} = \arg\min_{f} \{ L(f) \ s.t. \ f \in \partial(f_t, \epsilon), \ C(f) \leq C(f_t) + \eta_t \}$$

$$f_{\epsilon,\delta}(x) = \sum_{i=1}^{m} \frac{1}{2} (\sigma(x, \theta_i + \epsilon_i \delta_i) + \sigma(x, \theta_i - \epsilon_i \delta_i)) + \sum_{i=m+1}^{m+m'} \epsilon_i \sigma(x, \delta_i)$$

$$\Downarrow$$

Optimization Goal:

$$\min_{\boldsymbol{\varepsilon},\boldsymbol{\delta}} \left\{ L(f_{\boldsymbol{\varepsilon},\boldsymbol{\delta}}) \quad s.t. \quad \|\boldsymbol{\varepsilon}\|_0 \leq \eta_t, \quad \|\boldsymbol{\varepsilon}\|_\infty \leq \epsilon, \quad \|\boldsymbol{\delta}\|_{2,\infty} \leq 1 \right\}$$

Growing Network Width

Step One. Optimize $\boldsymbol{\delta}$ and $\boldsymbol{\epsilon}$ without complexity constraints

"Over-Grown" network

$$[\tilde{\boldsymbol{\varepsilon}}, \tilde{\boldsymbol{\delta}}] = \arg\min_{\boldsymbol{\varepsilon},\boldsymbol{\delta}} \left\{ L(f_{\boldsymbol{\varepsilon},\boldsymbol{\delta}}) \quad s.t. \quad \|\boldsymbol{\varepsilon}\|_\infty \le \epsilon, \quad \|\boldsymbol{\delta}\|_{2,\infty} \le 1 \right\}.$$

$$\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \ldots, \epsilon_{m+m'}), \boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_{m+m'})$$

$$\nabla_{\epsilon,\delta} L(f) = [\frac{\partial L(f)}{\partial \epsilon_1}, \ldots, \frac{\partial L(f)}{\partial \epsilon_{m+m'}}, \frac{\partial L(f)}{\partial \delta_1}, \ldots, \frac{\partial L(f)}{\partial \delta_{m+m'}}]^T$$

Minimize $L(f)$ in $f \in \boxed{\partial(f, \epsilon)}$ without complexity constraint, get "over-grown" network $\boxed{\tilde{f}_{t+1}}$

Architecture Optimization $\downarrow$

Select the top η_t neurons in $\tilde{f}_{t+1}$ with the highest importance measures to get f_{t+1}

Growing Network Width

Step Two. Re-Optimize ϵ with Taylor approximation on the loss

$$L(f_{\varepsilon,\tilde{\delta}}) = L(f) + \sum_{i=1}^{m+m'} \varepsilon_i s_i + O(\epsilon^2), \qquad \boxed{s_i} = \frac{1}{\tilde{\varepsilon}_i}\int_0^{\tilde{\varepsilon}_i} \nabla_{\zeta_i} L(f_{[\tilde{\varepsilon}_{\neg i},\zeta_i],\tilde{\delta}})d\zeta_i$$

$[\tilde{\varepsilon}_{\neg i}, \zeta_i]$ denotes replacing the i-th element of $\tilde{\varepsilon}$ with ζ_i

$L(f_{\varepsilon,\tilde{\delta}})$. $\varepsilon \to 0$. $\varepsilon = (\varepsilon_1, \varepsilon_2, \cdots)$

$$L(f_{\varepsilon,\tilde{\delta}}) = L(f_{\tilde{\varepsilon},\tilde{\delta}}) + \frac{\partial L(f)}{\partial \varepsilon_1}\Big|_{\varepsilon_1=0} \cdot \varepsilon_1 + \frac{\partial L(f)}{\partial \varepsilon_2}\Big|_{\varepsilon_2=0} \cdot \varepsilon_2 + \cdots$$

$$+ \frac{\partial L(f)}{\partial \varepsilon_{m+m'}}\Big|_{\varepsilon_{m+m'}=0} \cdot \varepsilon_{m+m'} + O(\varepsilon^2)$$

$$= L(f) + \sum_{i=1}^{m+m} \frac{\partial L(f)}{\partial \varepsilon_i}\Big|_{\varepsilon_i=0} \cdot \varepsilon_i$$

$$\frac{\partial L(f)}{\partial \varepsilon_i} = \frac{\partial L(f_{\tilde{\varepsilon},\tilde{\delta}})}{\partial \varepsilon_i} = \frac{1}{\tilde{\varepsilon}_i}\int_0^{\tilde{\varepsilon}_i} \nabla_{\zeta_i} L(f_{\underline{[\tilde{\varepsilon}_{\neg i},\zeta_i]},\tilde{\delta}})\, d\zeta_i$$

$[\tilde{\varepsilon}_{\neg i}, \zeta_i]$ denotes replacing the i-th element of $\tilde{\varepsilon}$ with ζ_i

s_i: an integrated gradient: measures the contribution of turning on the i-th new neuron

Growing Network Width

Step Two. Re-Optimize ϵ with Taylor approximation on the loss

$$L(f_{\varepsilon,\tilde{\delta}}) = L(f) + \sum_{i=1}^{m+m'} \varepsilon_i s_i + O(\epsilon^2), \qquad \boxed{s_i} = \frac{1}{\tilde{\varepsilon}_i} \int_0^{\tilde{\varepsilon}_i} \nabla_{\zeta_i} L(f_{[\tilde{\varepsilon}_{\neg i}, \zeta_i], \tilde{\delta}}) d\zeta_i$$

$[\tilde{\varepsilon}_{\neg i}, \zeta_i]$ denotes replacing the i-th element of $\tilde{\varepsilon}$ with ζ_i

In practice, approximate the integration in s_i by discrete sampling:

$$s_i \approx \frac{1}{n} \sum_{z=1}^{n} \nabla_c c_z L(f_{[\tilde{\epsilon}_{\neg i}, c_z], \delta})$$

$$\hat{\varepsilon} = \arg\min_{\varepsilon} \left\{ \sum_{i=1}^{m+m'} \varepsilon_i s_i \quad s.t. \quad \|\varepsilon\|_0 \le \eta_t, \quad \|\varepsilon\|_\infty \le \epsilon \right\}$$

s_i: an integrated gradient: measures the contribution of turning on the i-th new neuron

$$f_{t+1} = f_{\hat{\epsilon}, \tilde{\delta}}$$

Goal in Architecture Optimization:

- Instantiate neighborhood $\partial(f_t, \epsilon)$ ✅
- Solve the optimization in f_{t+1} ✅

Experiment Result

backbone: VGG-19
dataset: CIFAR-10

method:
- Net2Net - grows networks uniformly by randomly selecting the existing neurons in each layer
- NASH - a random sampling search method using network morphism
- Firefly Descent

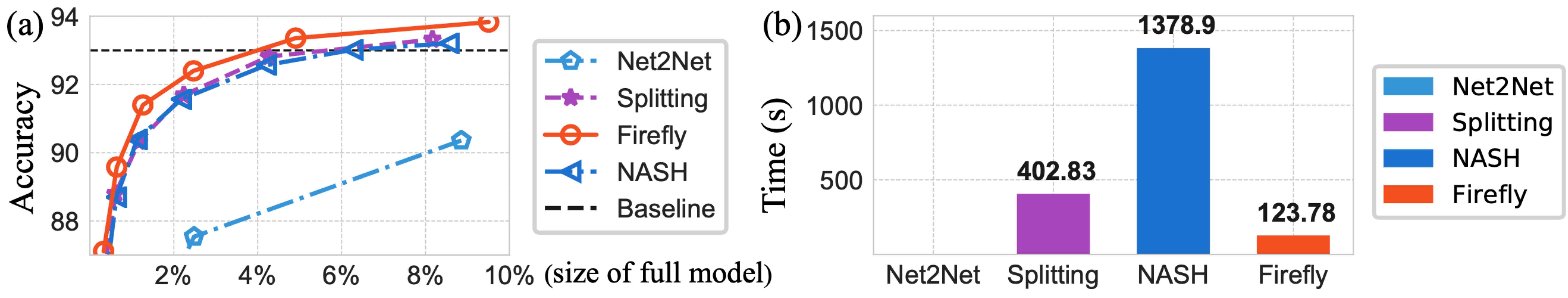


Figure 4: (a) Results of growing increasingly wider networks on CIFAR-10; VGG-19 is used as the backbone. (b) Computation time spent on growing for different methods.

- Outperform baseline test accuracy with less than 10% size of full model， growing fast
- Achieve comparable test accuracy as the full model with only 4% size of full model

Experiment Result

backbone: cell-based architecture searching
dataset: ImageNet

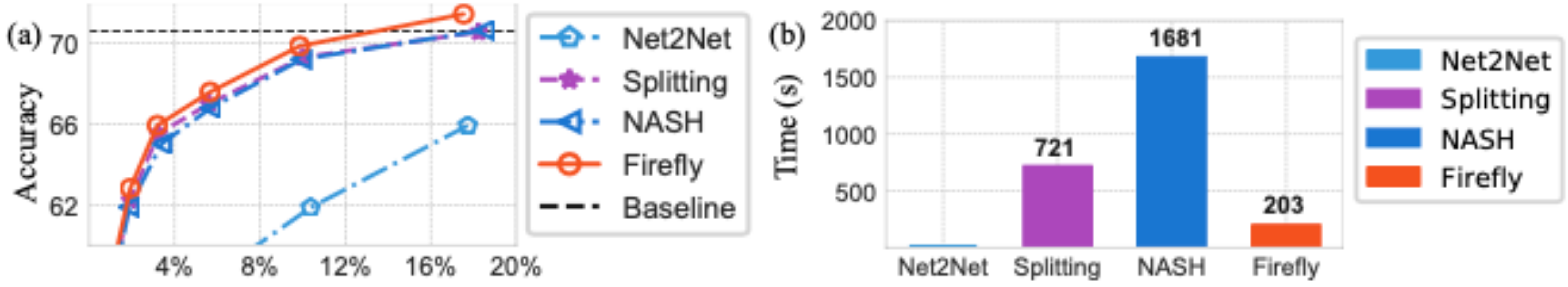
Table 1 reports the results comparing `Firefly` with several NAS baselines. Our method achieves a similar or better performance comparing with those RL-based and gradient-based methods like ENAS or DARTS, but with higher computational efficiency in terms of the total search time.

Method	Search Time (GPU Days)	Param (M)	Error
NASNet-A (Zoph et al., 2018)	2000	3.1	2.83
ENAS (Pham et al., 2018)	4	4.2	2.91
Random Search	4	3.2	3.29 ± 0.15
DARTS (first order) (Liu et al., 2018b)	1.5	3.3	3.00 ± 0.14
DARTS (second order) (Liu et al., 2018b)	4	3.3	2.76 ± 0.09
`Firefly`	1.5	3.3	2.78 ± 0.05

Table 1: Performance compared with several NAS baseline

- Achieves a similar or better performance comparing with several NAS methods,
- Much less search time

backbone: MobileNetV1
dataset: CIFAR-100



Review

Contribution:

- Smoothly change the network, grow the networks without accuracy drop
- Flexible growing width and depth

Drawbacks:

- 通过gradient optimization将“search”和“grow”的过程combine在一起，continuous growing
- one-layer grow?
- 每一次architecture optimization（grow）后需要重新训练新网络
- 实验无精确的实验结果，对比模型较naive
- Formula without proof