LinkedIntern
Class of 2019
Summer

Scalable Automated Machine Learning

in GLMix 2.0

Yuwei Qiu
AI Algorithms Foundation Team

Carnegie Mellon University

Agenda

Motivation

Algorithm

Implementation

Results

Motivation

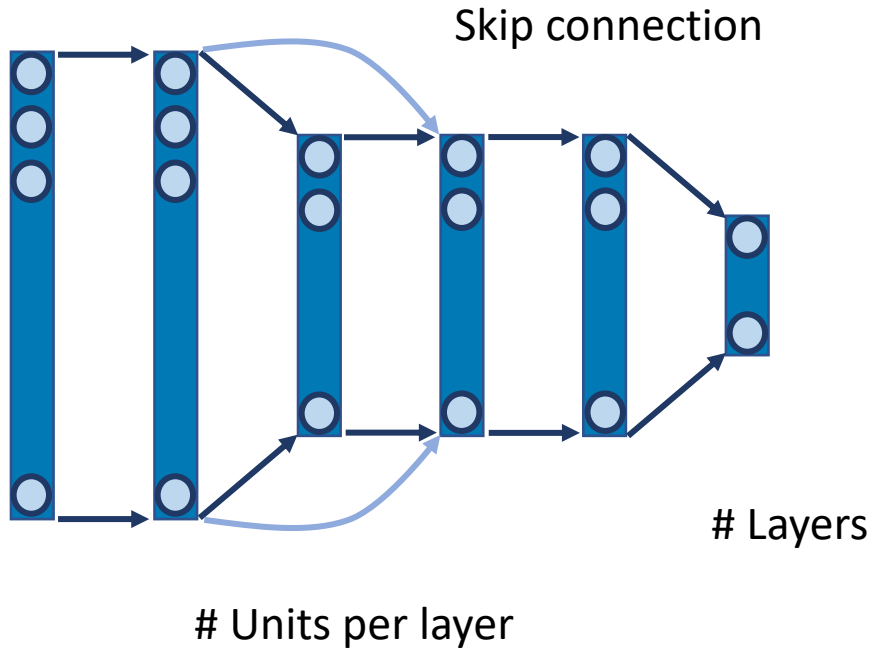
Deep Learning in the enterprise domain

- Deep learning and neural network are not only limited to researches.
- Ultimate goal: Learn more effectively, less trial-and-error.

Problem of Learning

- Performance is very **sensitive** to **parameters**
- Architectural parameters & Others

(e.g. Feedforward Network)



Current:

Solution =
Expertise + Data + Hours and Hours

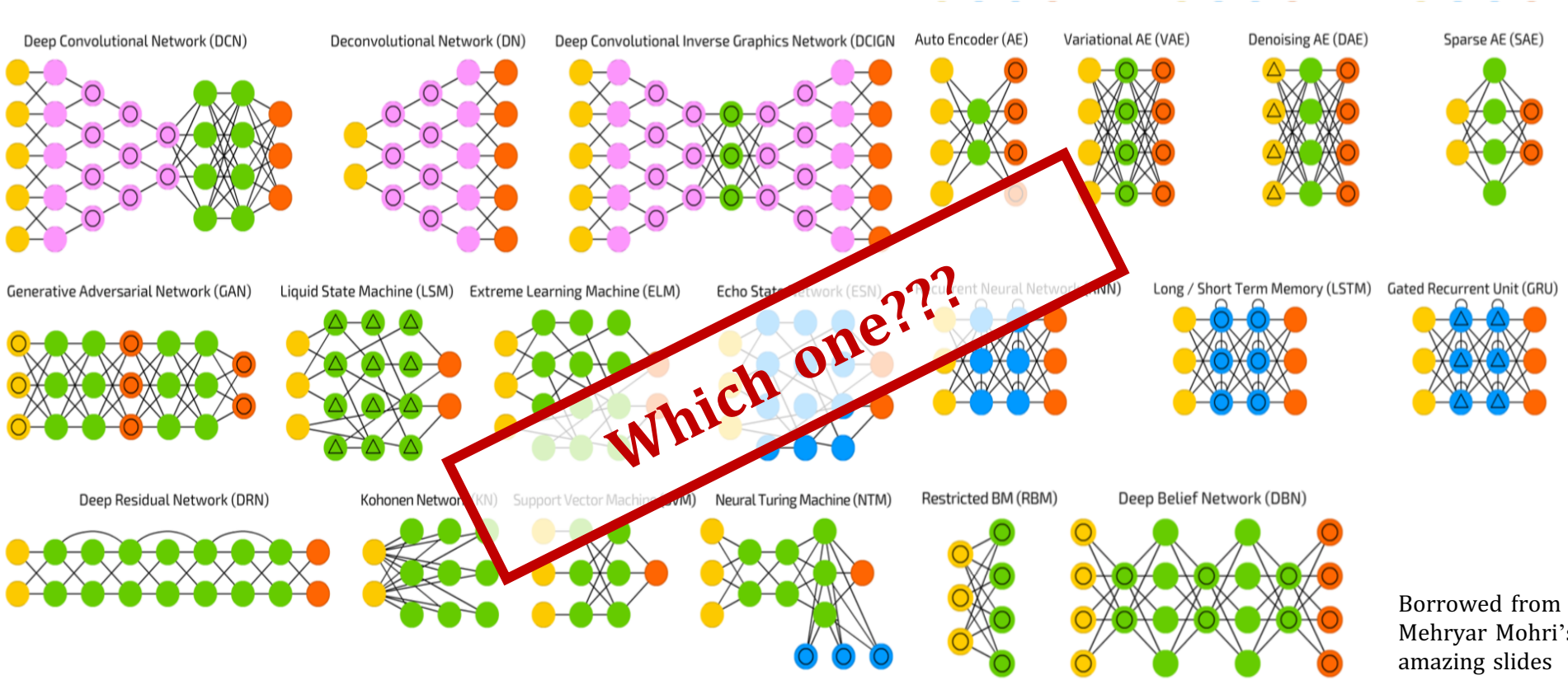
But can we turn this into:

Solution =
Data + Less Hours (+ Improvement of Performance)

Algorithm

How to determine your model?

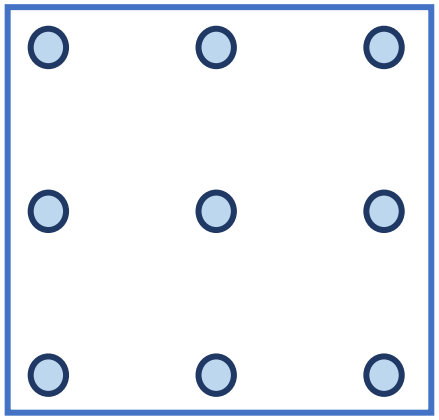
- Some possible answers…



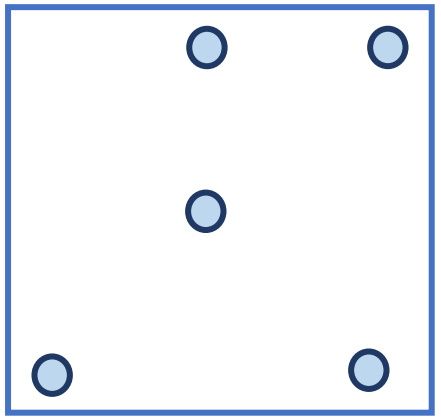
Borrowed from Mehryar Mohri's amazing slides

Grid Search and Random Search

- Shoot all of combinations.
- Then find the best set of parameters



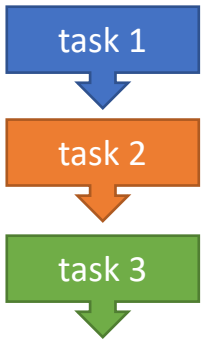
Grid Layout



Random Layout

Grid Search and Random Search

- If you want to grid search on N sets of parameters…
  - Typically **much more time or much more resources**.

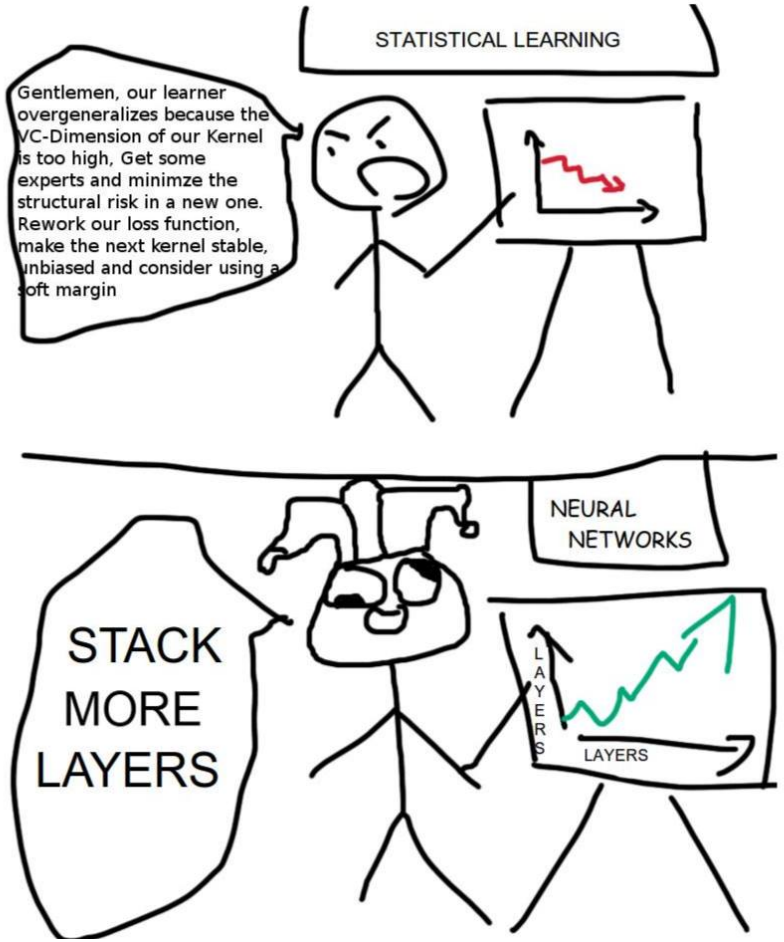


N times more time

N times more resources

One AI Joke…

Vertically
Horizontally

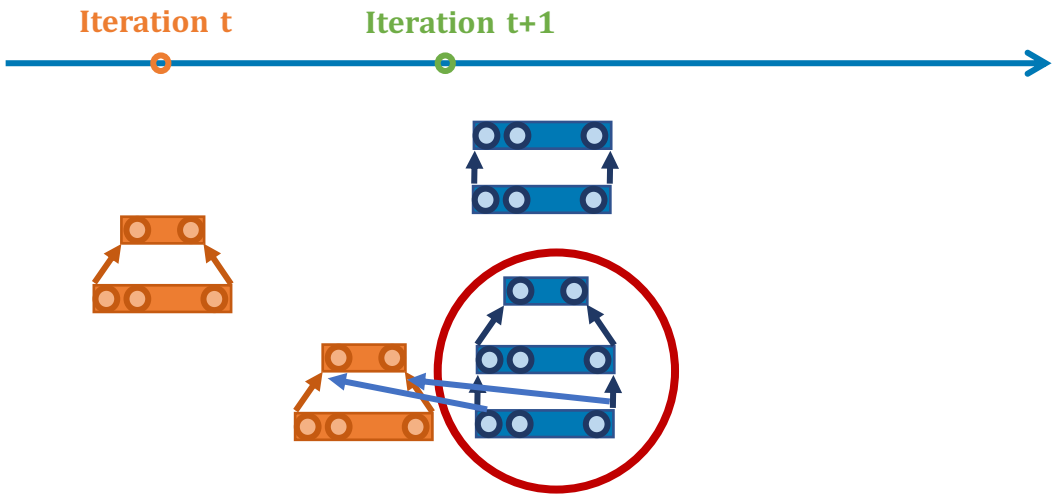


AdaNet

- **Instead of stacking heavyweight layers, AdaNet stacks with lightweight subnetworks.**
- **To automatically find the best model for your data input.**

- Presented in "*AdaNet: Adaptive Structural Learning of Artificial Neural Networks*" at *ICML 2017*

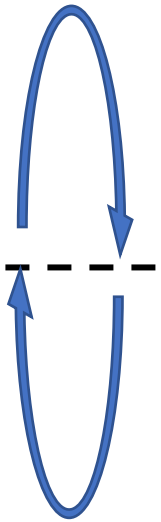
Bird View



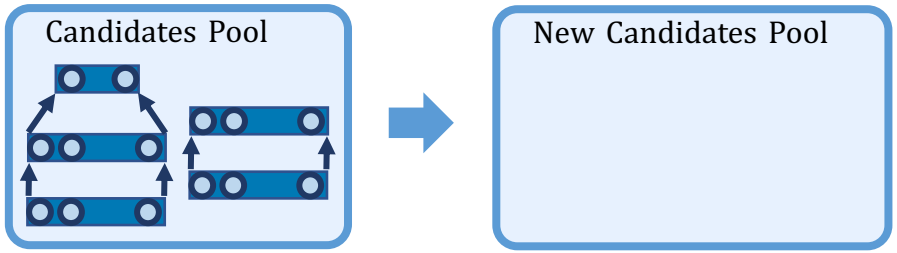
Train subnetworks

Select the best subnetwork

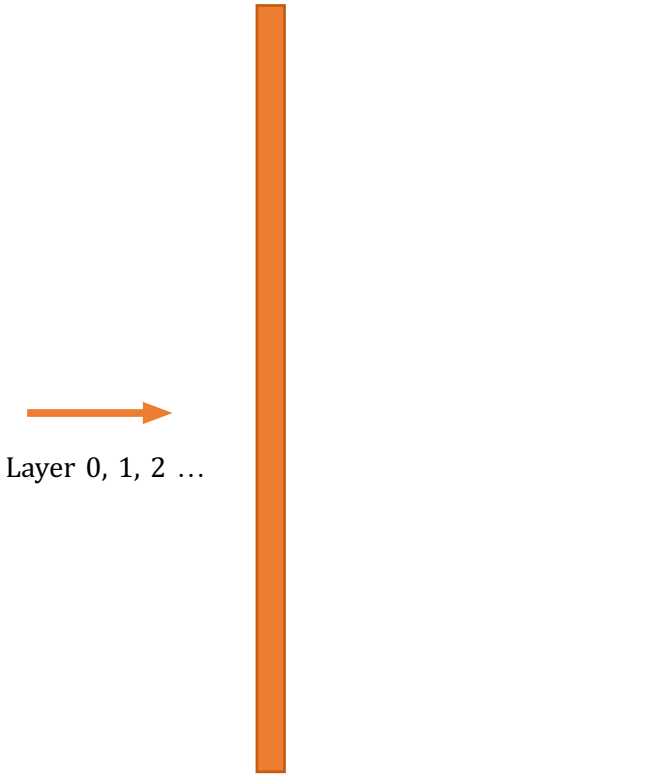
Build the ensemble



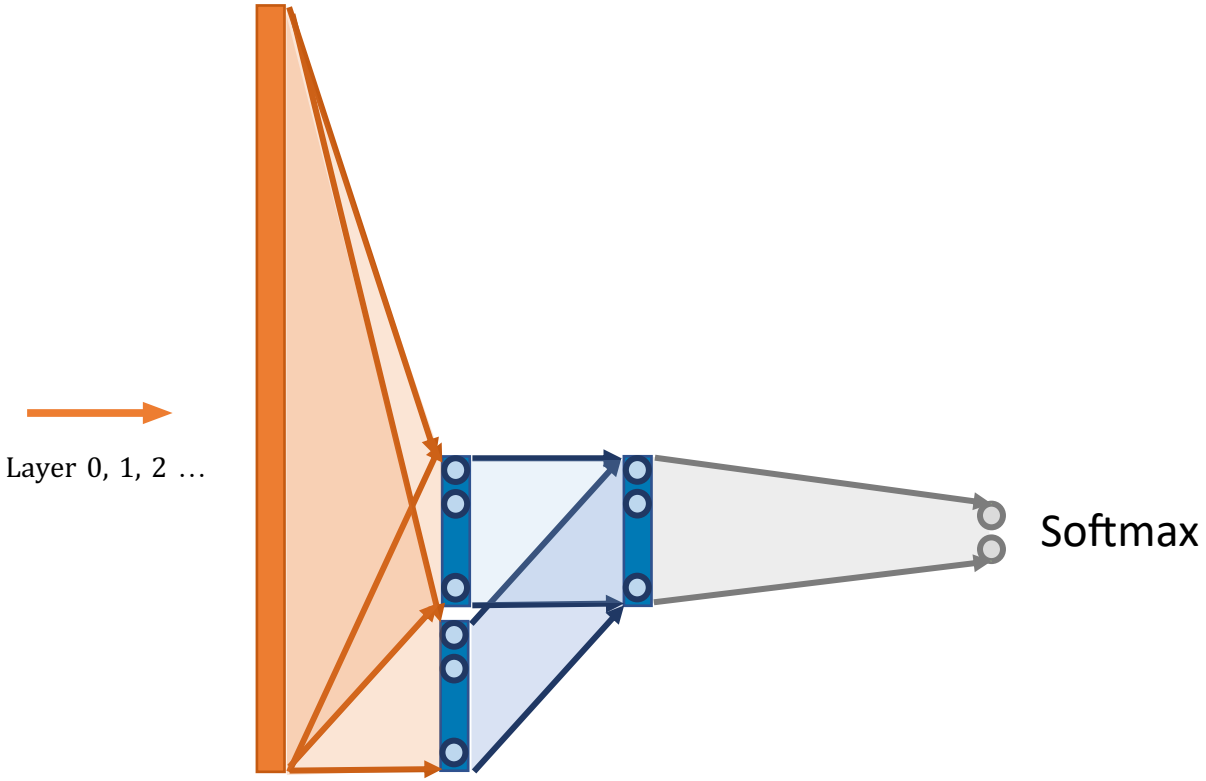
Select next subnetworks



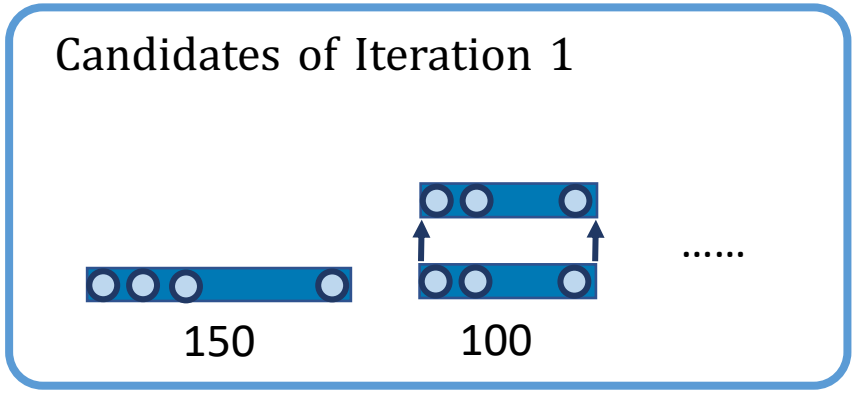
A Closer Look into AdaNet Structure



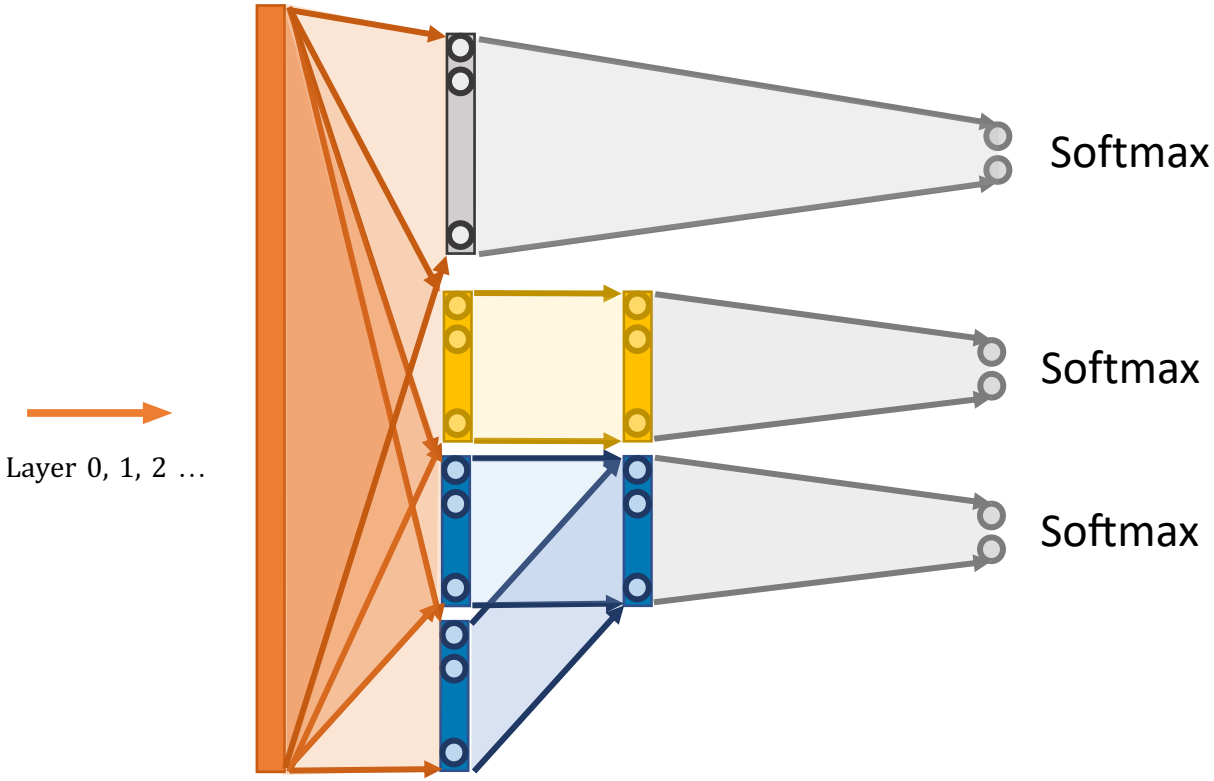
A Closer Look into AdaNet Structure



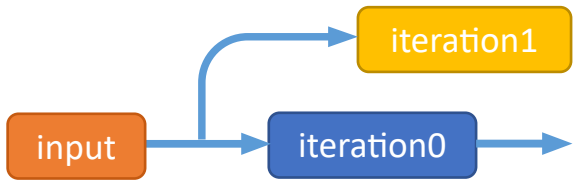
- Iteration 1
  - Start from this structure

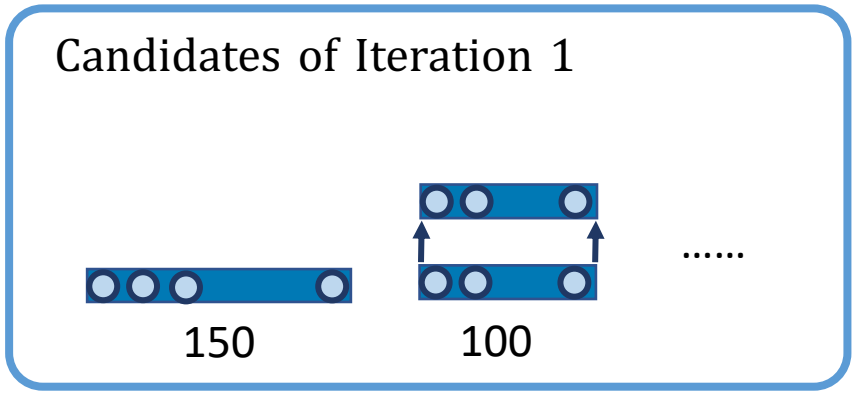


A Closer Look into AdaNet Structure



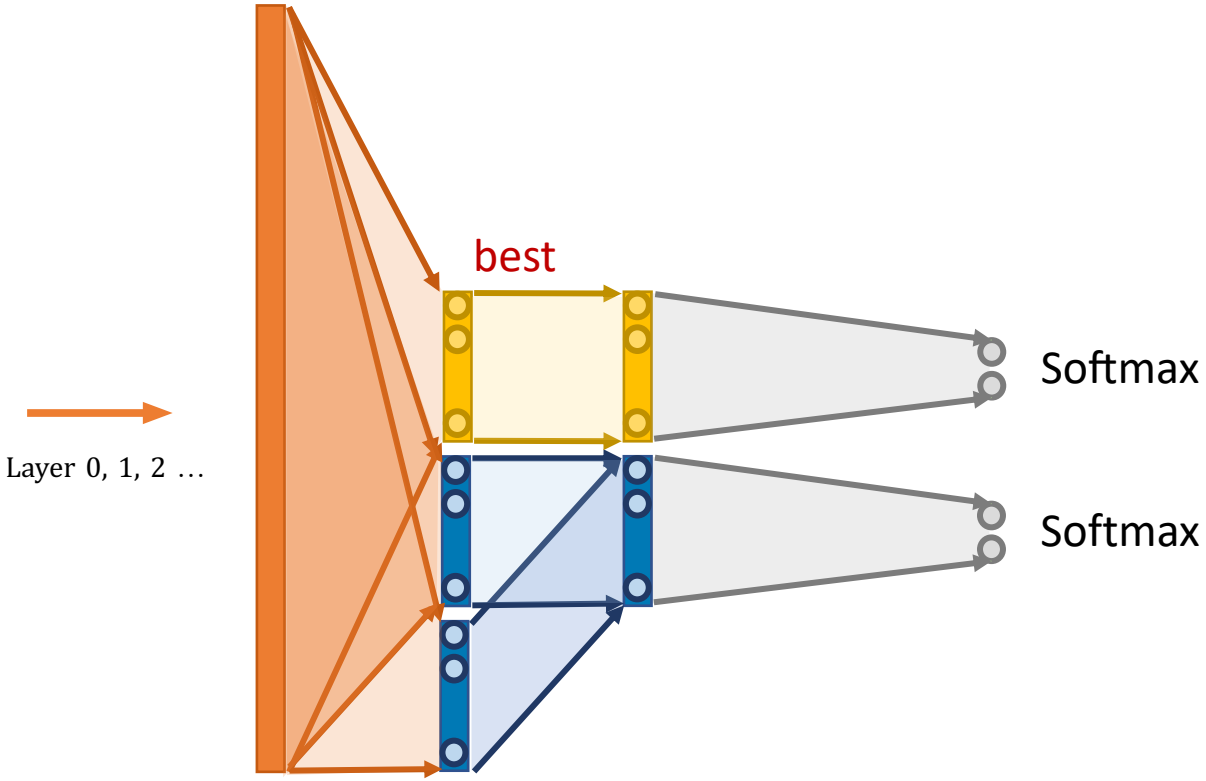
- Iteration 1
  - Train new subnetworks separately



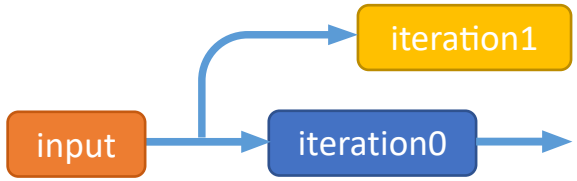


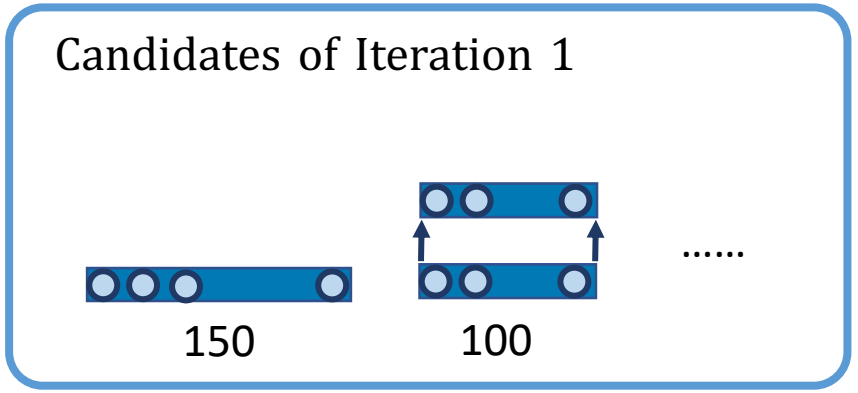
LinkedIntern
Yuwei Qiu

A Closer Look into AdaNet Structure

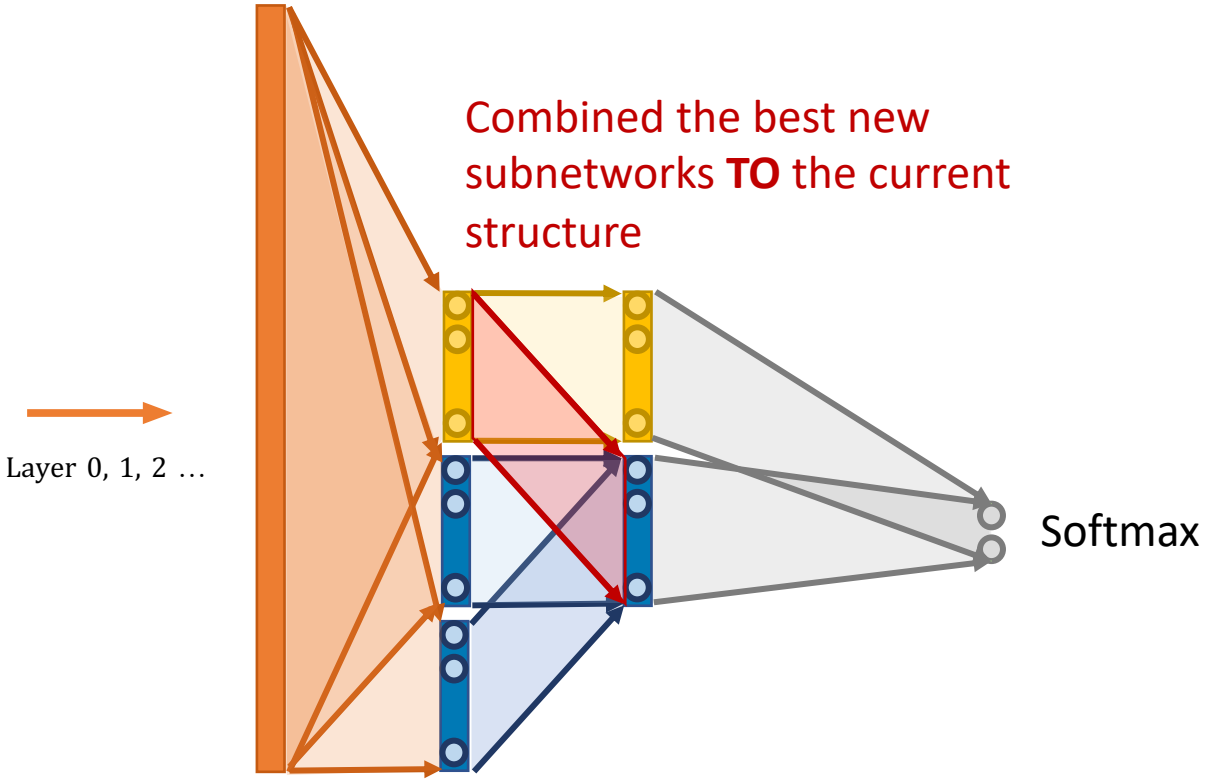


- Iteration 1
  - Train new subnetworks separately

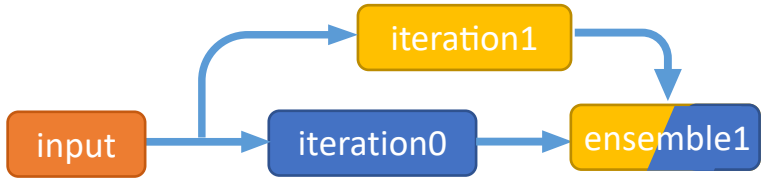


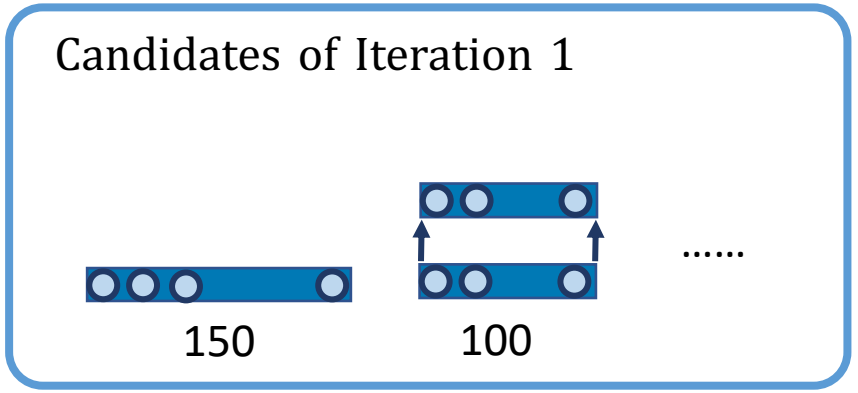


A Closer Look into AdaNet Structure

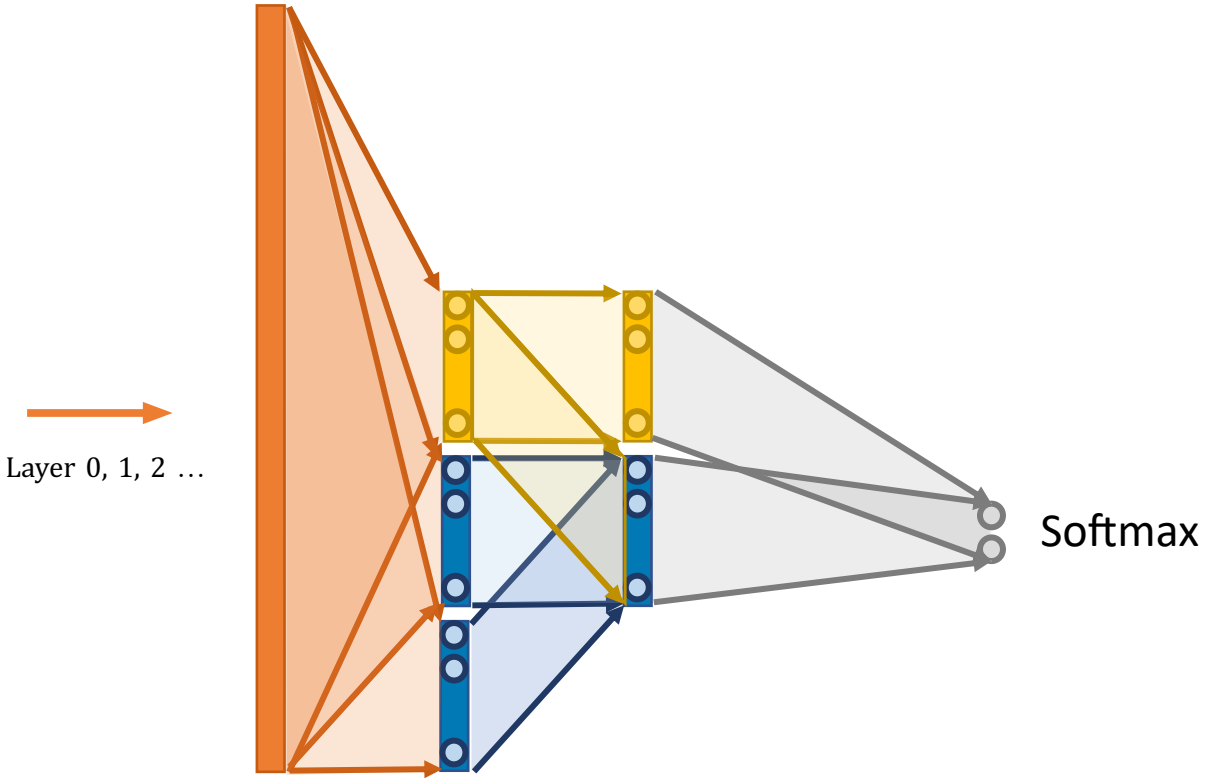


- Iteration 1
  - Train new subnetworks separately
  - Ensemble with newly added weights
    - Each unit in layer k of this subnetwork **may have connections** to the existing units in layer k+1 of current adanet in addition to units in layer k+1 of the subnetwork.

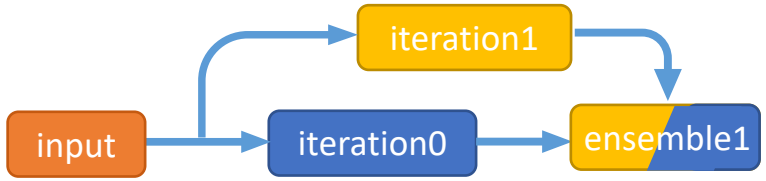




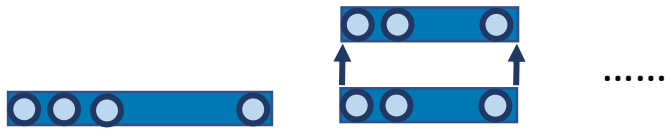
A Closer Look into AdaNet Structure



- Iteration 1 – Final model
- Select new subnetworks for next iteration

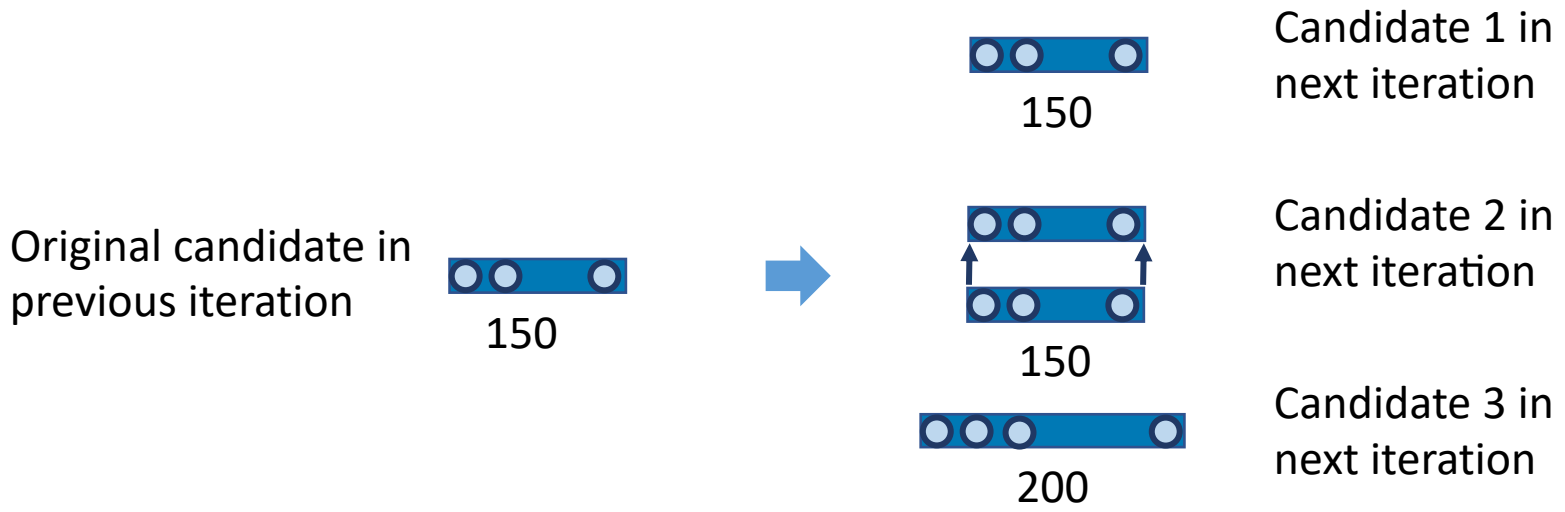


Candidates of Iteration 1

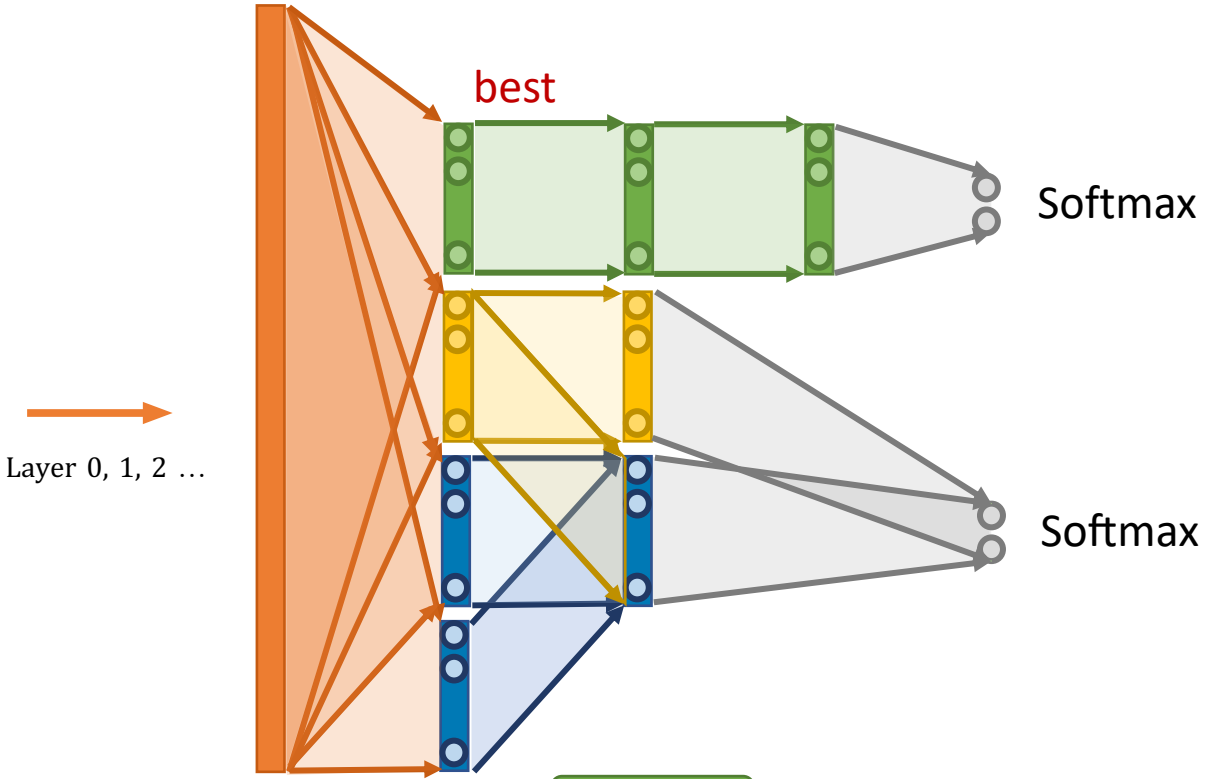


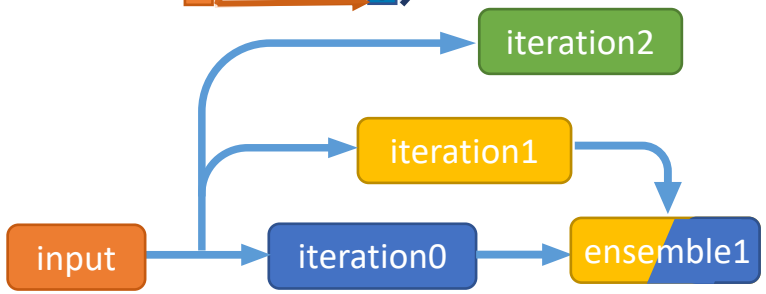
Select new candidates for the next iteration

- Candidates for the next iteration may include a subnetwork with:
  - (Just an example)
    - **the same as**
    - **1 layer deeper subnetwork than**
    - **more units than**
  - that of the subnetworks of current candidates pool.
- **Only keep best candidates for the next iteration.**



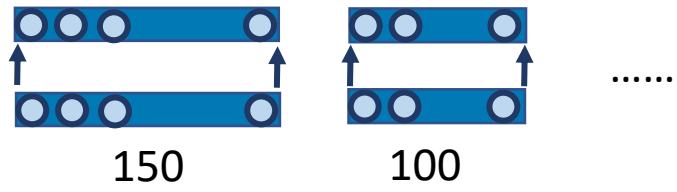
A Closer Look into AdaNet Structure



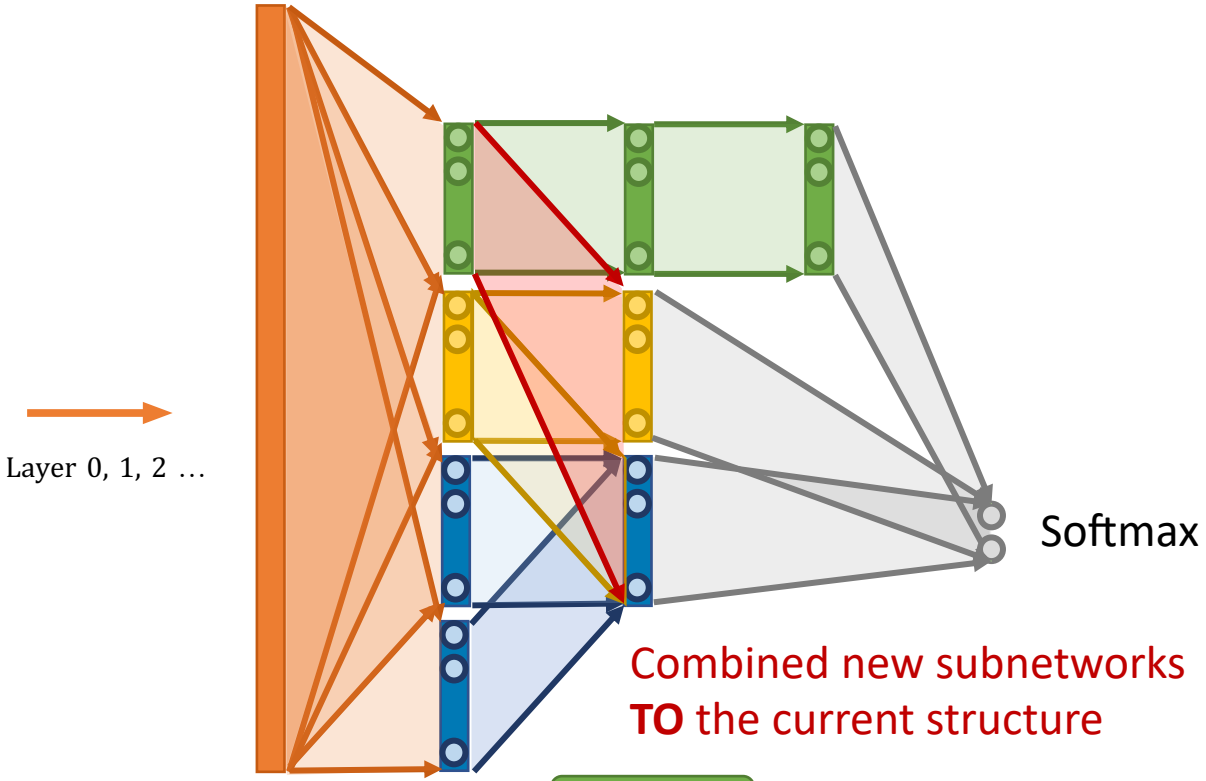


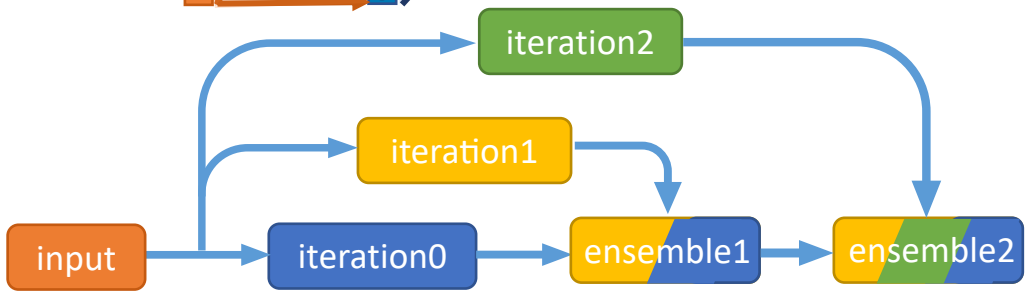
- Iteration 2
  - Train new subnetworks separately

Candidates of Iteration 2



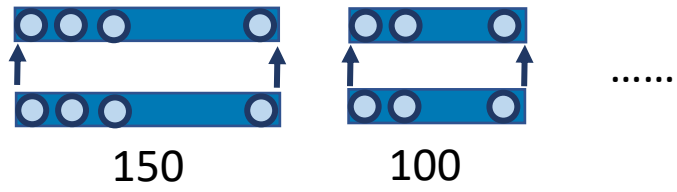
A Closer Look into AdaNet Structure



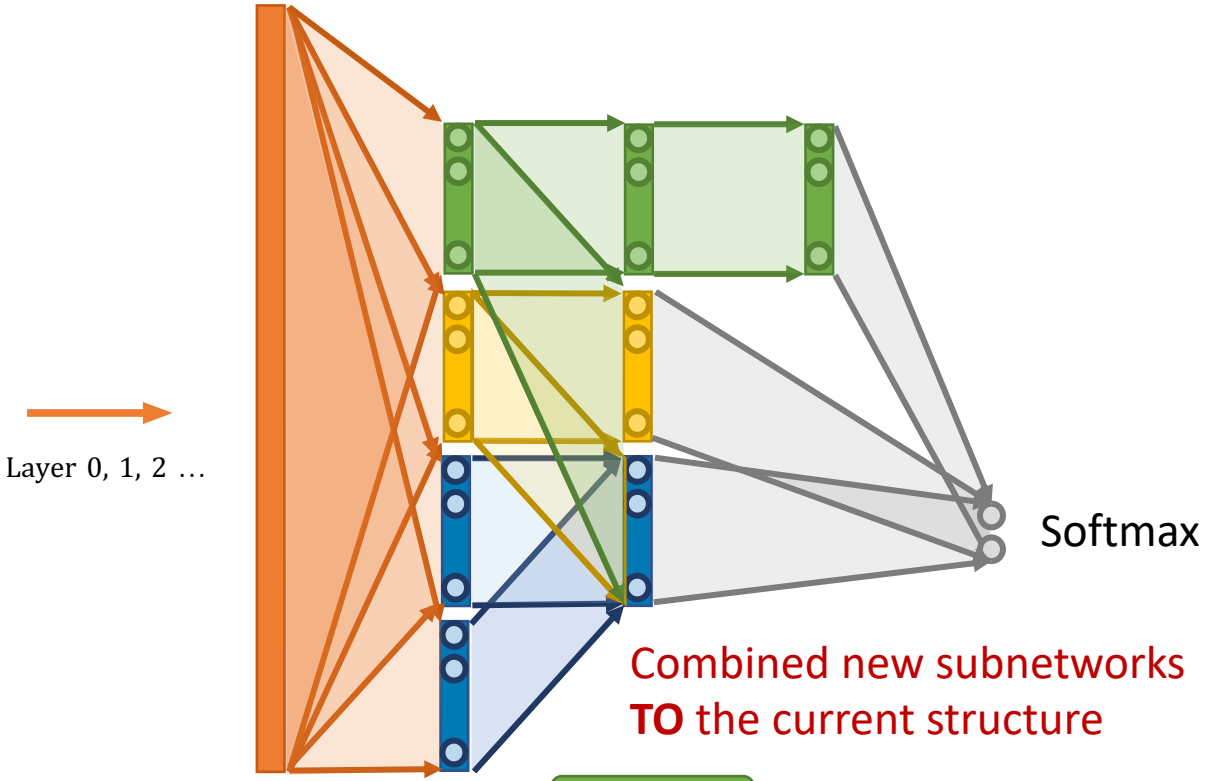


- Iteration 2
  - Train new subnetworks separately
  - Ensemble with newly added weights
    - Each unit in layer k of this subnetwork **may have connections** to the existing units in layer k+1 of current network in addition to units in layer k+1 of the subnetwork.

Candidates of Iteration 2

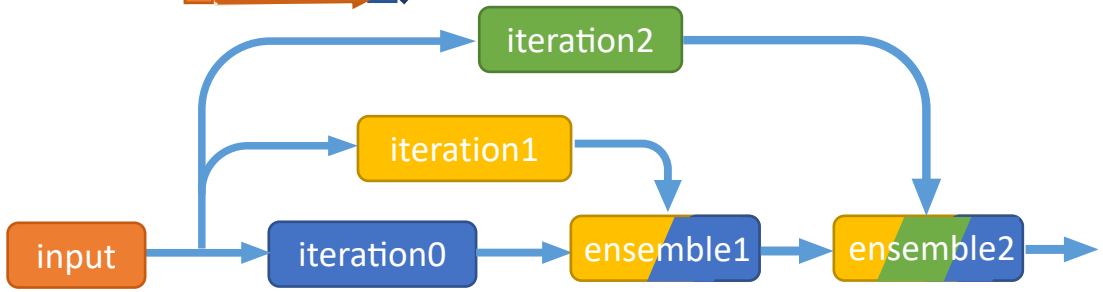


A Closer Look into AdaNet Structure

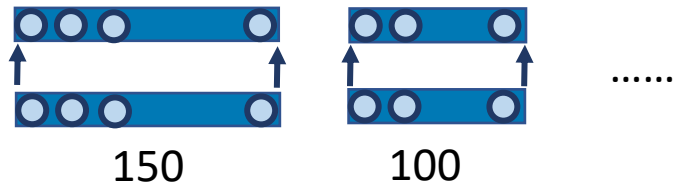


- Final Model

Combined new subnetworks
TO the current structure



Candidates of Iteration 2



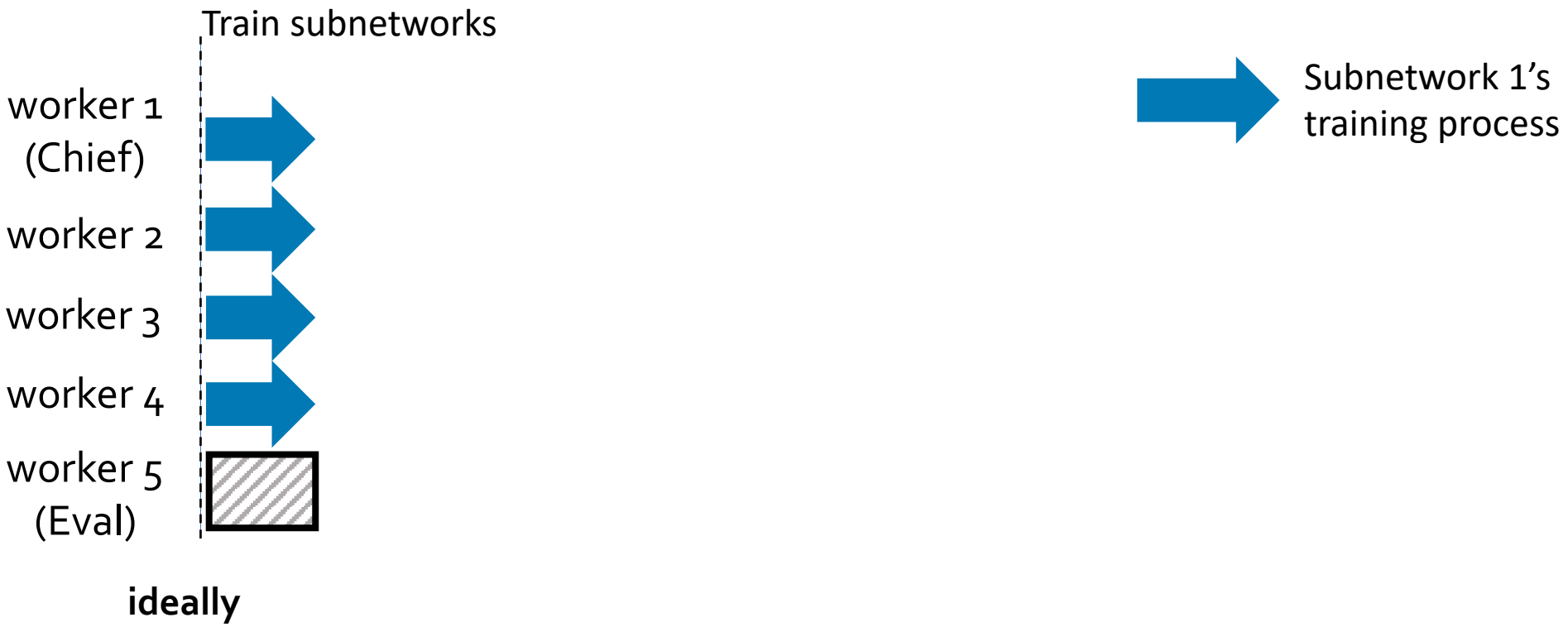
Implementation

Challenging Distributed Training of AdaNet

- Training neural network models is computationally demanding
  - Even the most lightweight network can cost an impractically long time on a single machine.

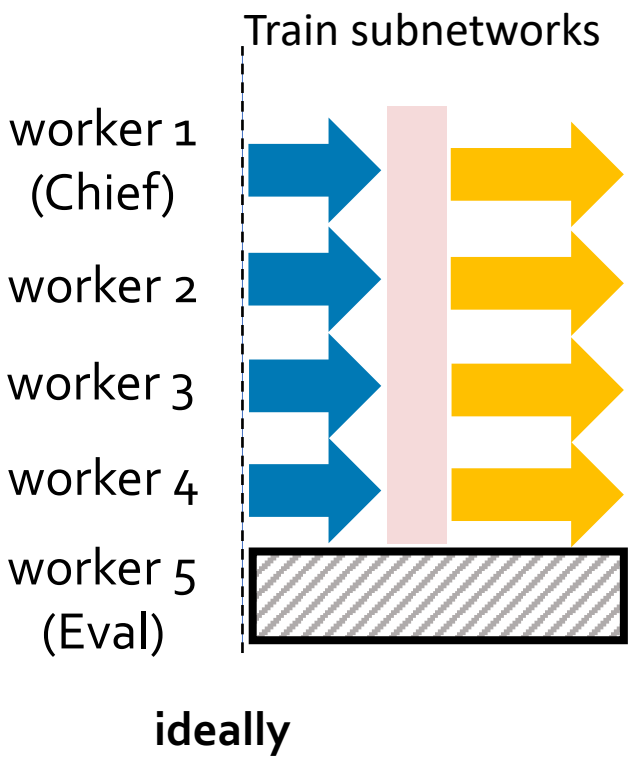
Challenging Distributed Training of AdaNet

- Suppose each time we try 2 different subnetworks.
  - Subnetwork 2 is slightly larger than Subnetwork 1

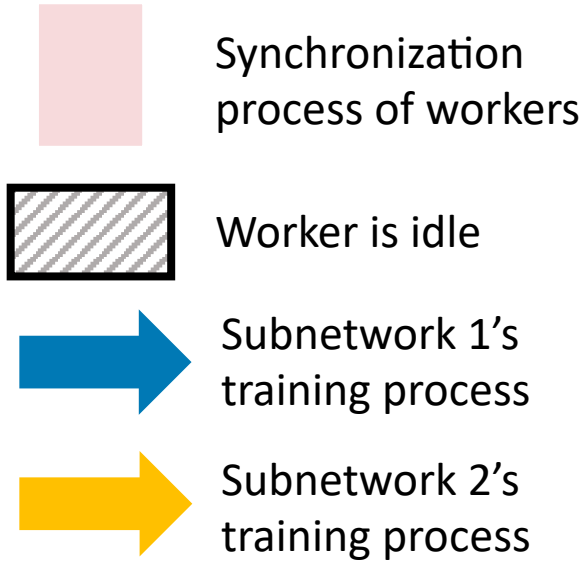


Challenging Distributed Training of AdaNet

- Suppose each time we try 2 different subnetworks.
  - Subnetwork 2 is slightly larger than Subnetwork 1

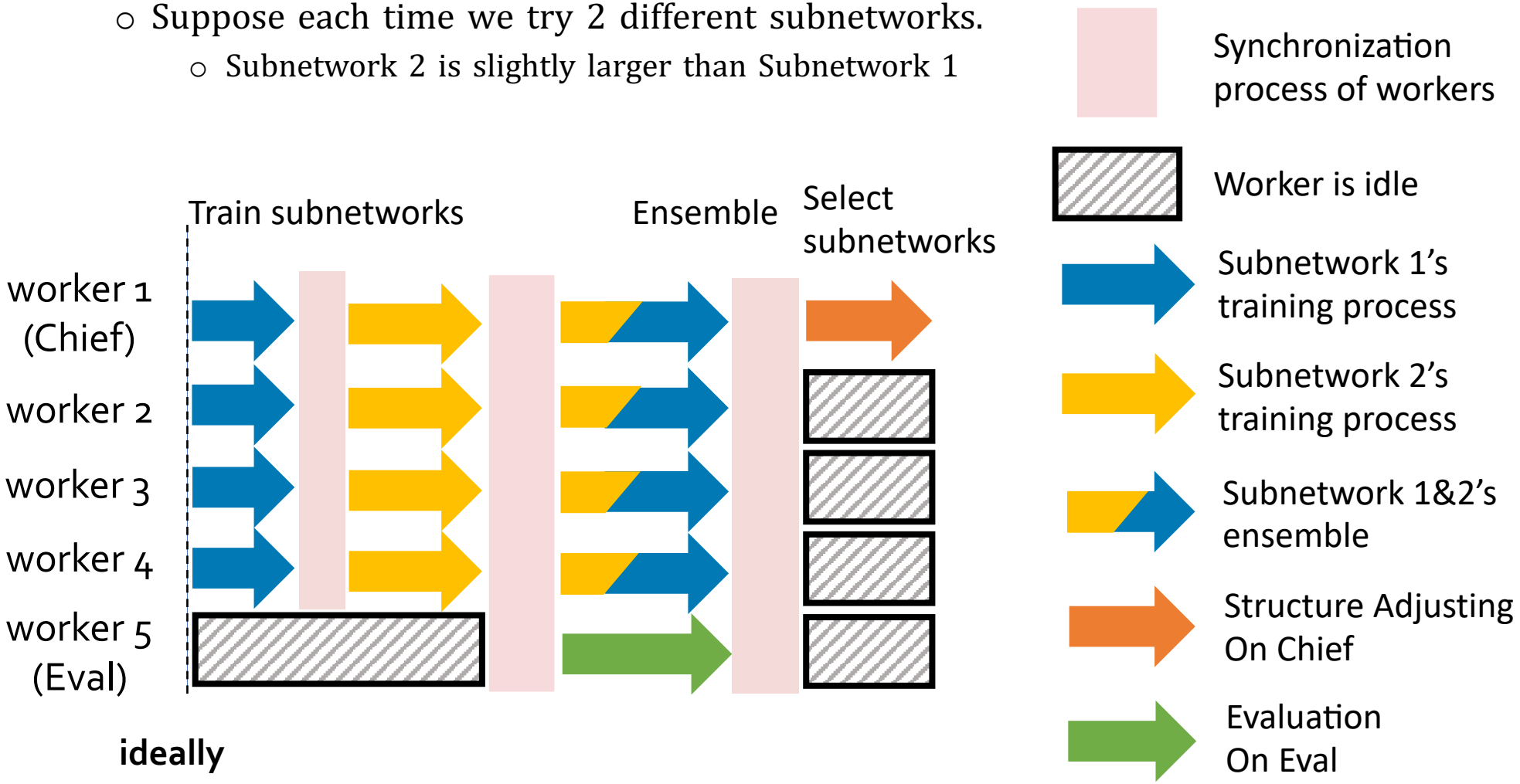


ideally



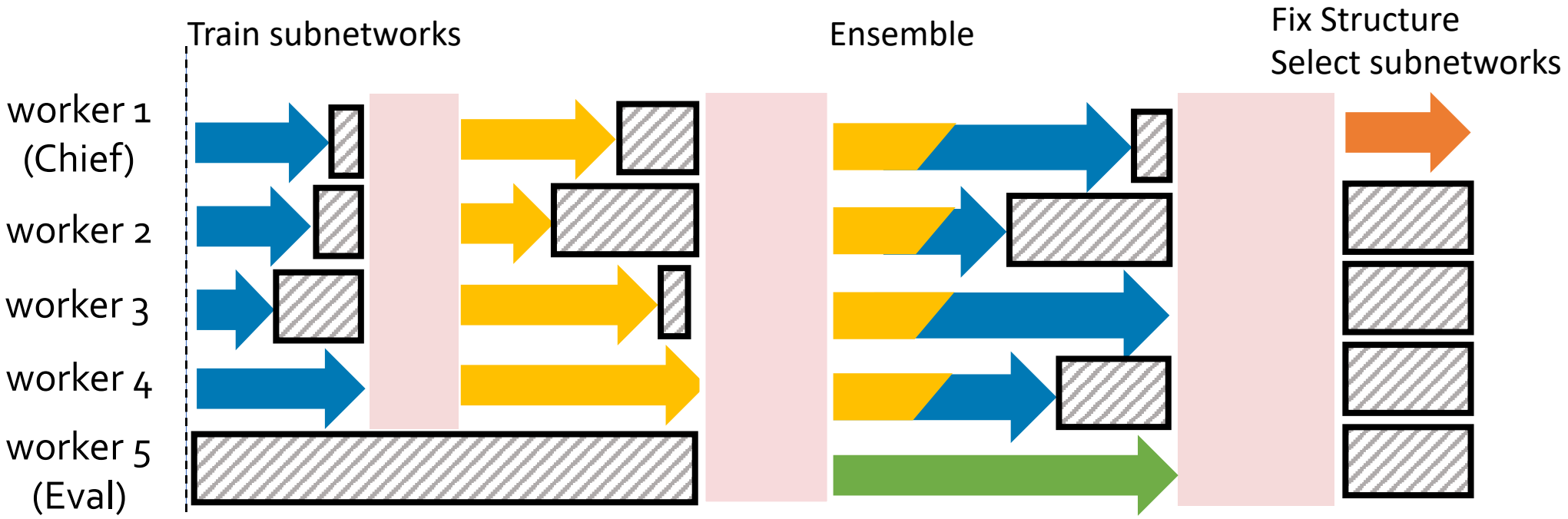
Challenging Distributed Training of AdaNet

- Suppose each time we try 2 different subnetworks.
  - Subnetwork 2 is slightly larger than Subnetwork 1

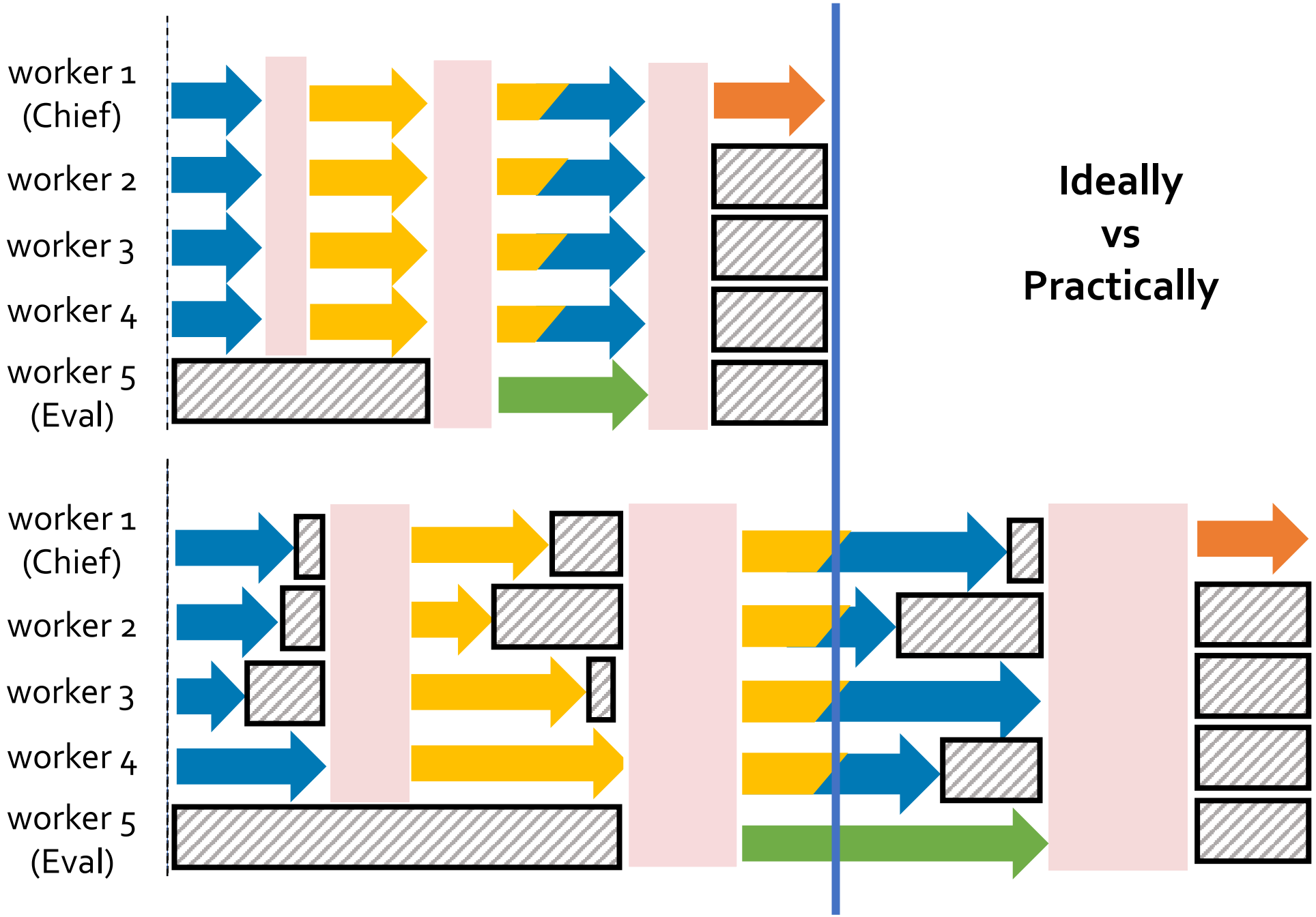


Challenging Distributed Training of AdaNet

- Suppose each time we try 2 different subnetworks
  - Subnetwork 2 is slightly larger than Subnetwork 1



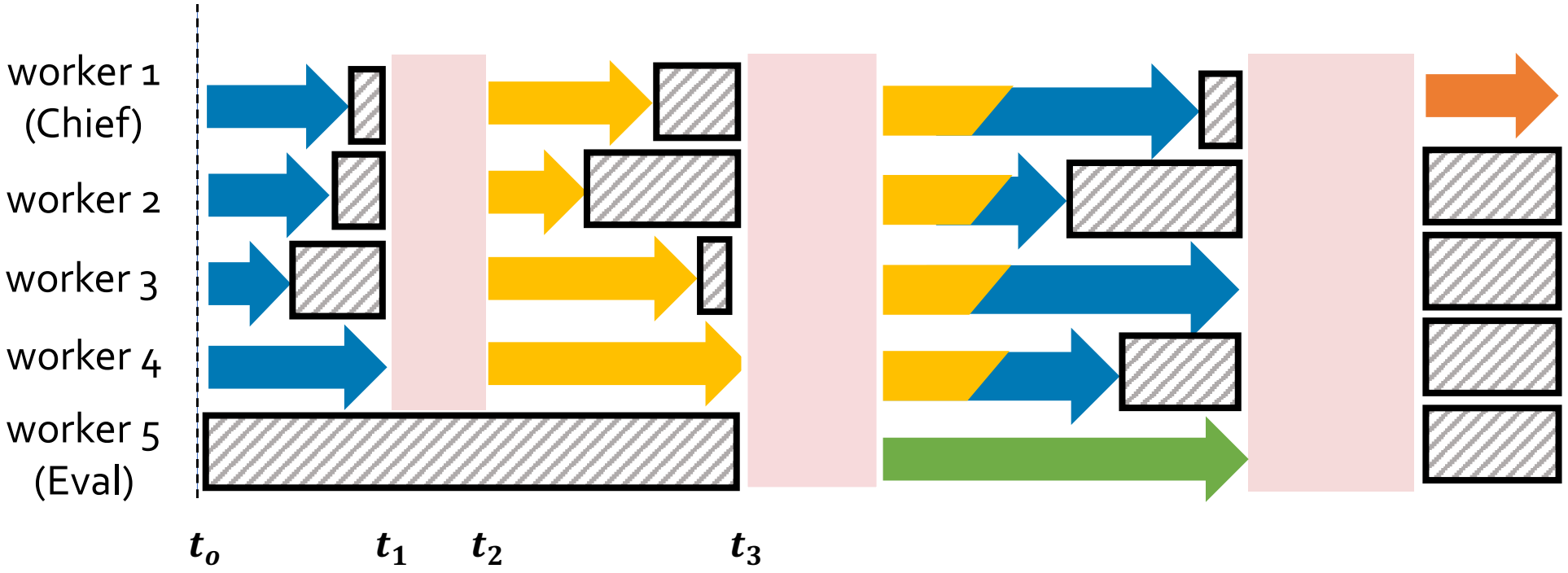
Practically



LinkedIntern
Yuwei Qiu

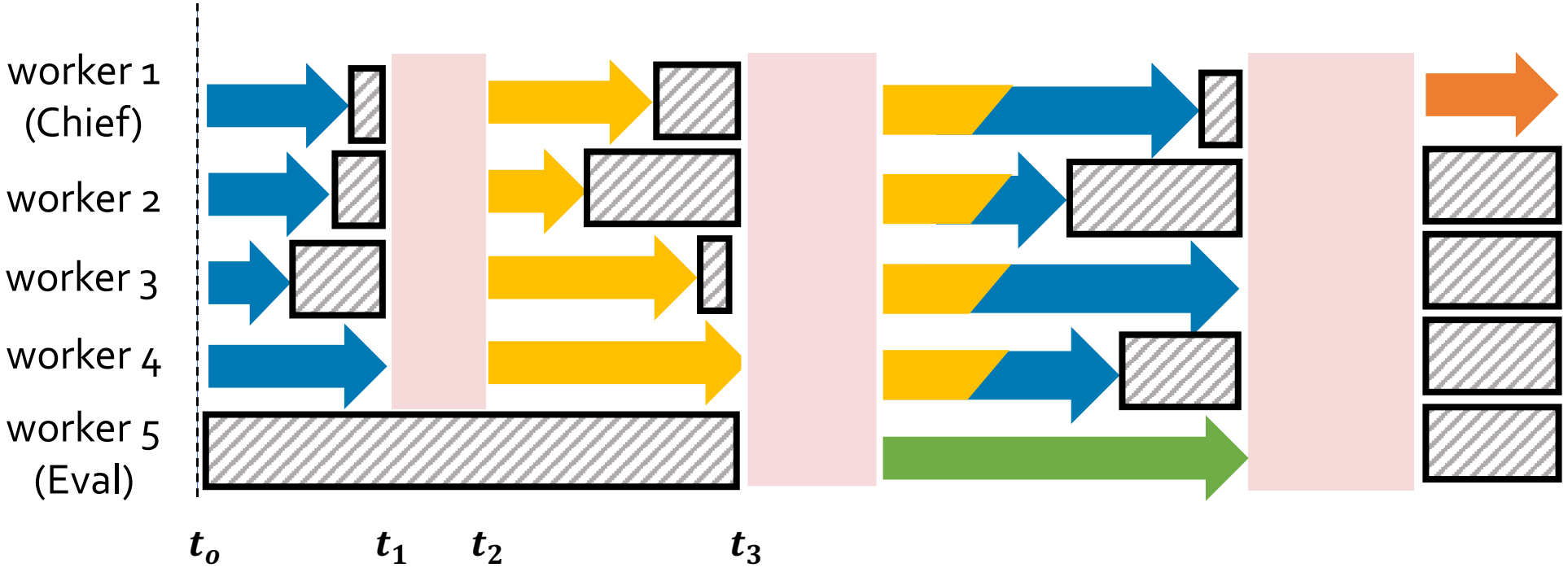
Observations

- Many workers involved in the training of one subnetwork
  - Heavier straggler effect and longer synchronization time



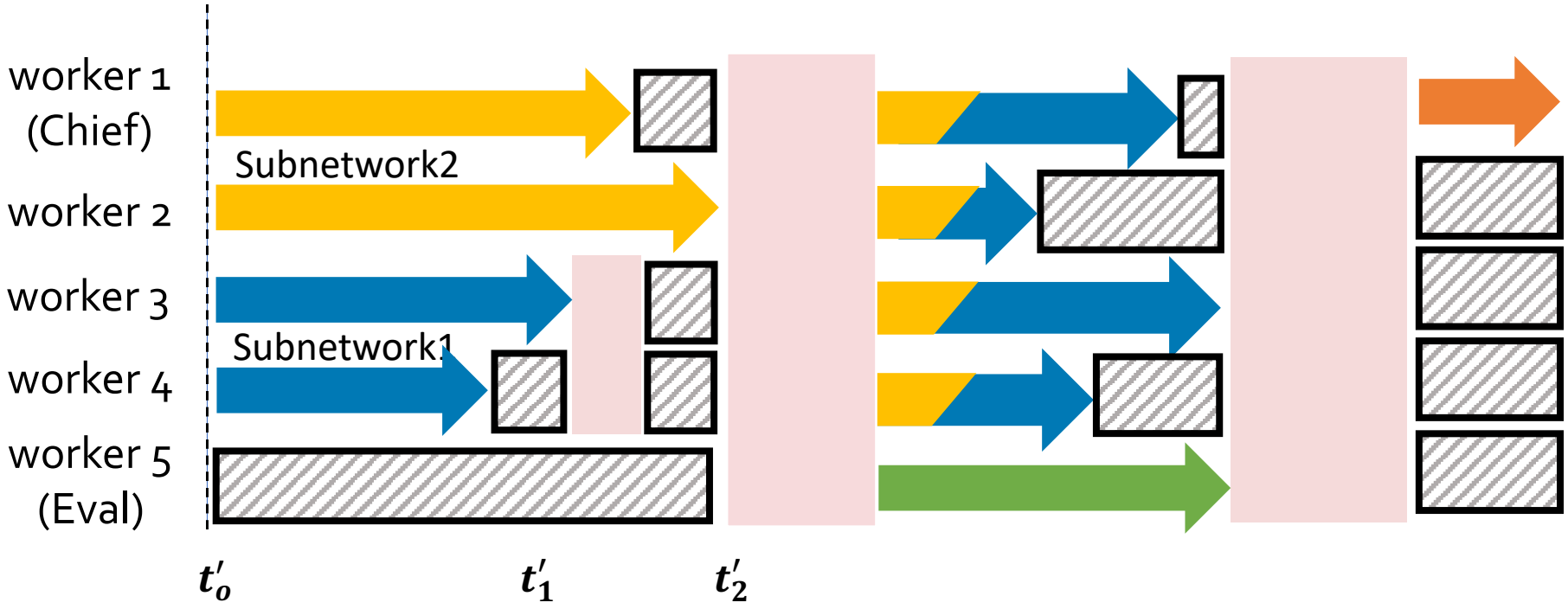
Reduce Straggler Effect and Sync Time

- Subnetwork 1 occupies **4 workers from t_o to t_1**
- Subnetwork 2 occupies **4 workers from t_2 to t_3**



Reduce Straggler Effect and Sync Time

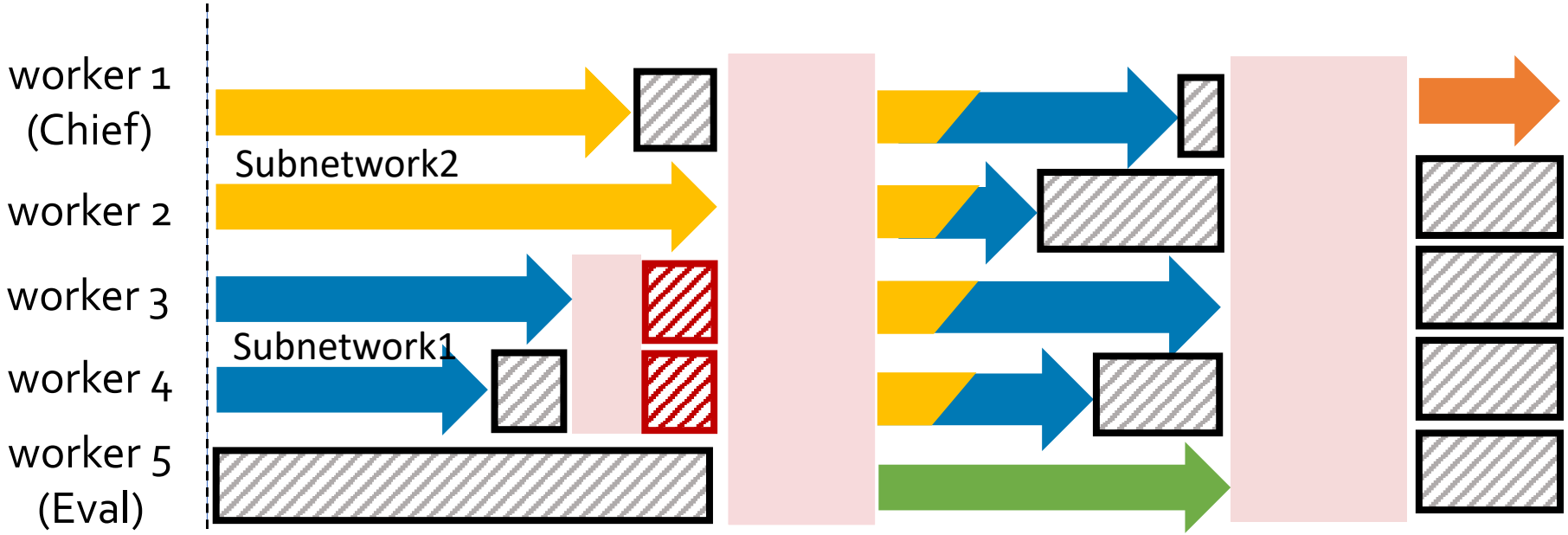
- Subnetwork 1 occupies **2 workers from t'_o to t'_1**
- Subnetwork 2 occupies **2 workers from t'_o to t'_2**



LinkedIntern
Yuwei Qiu

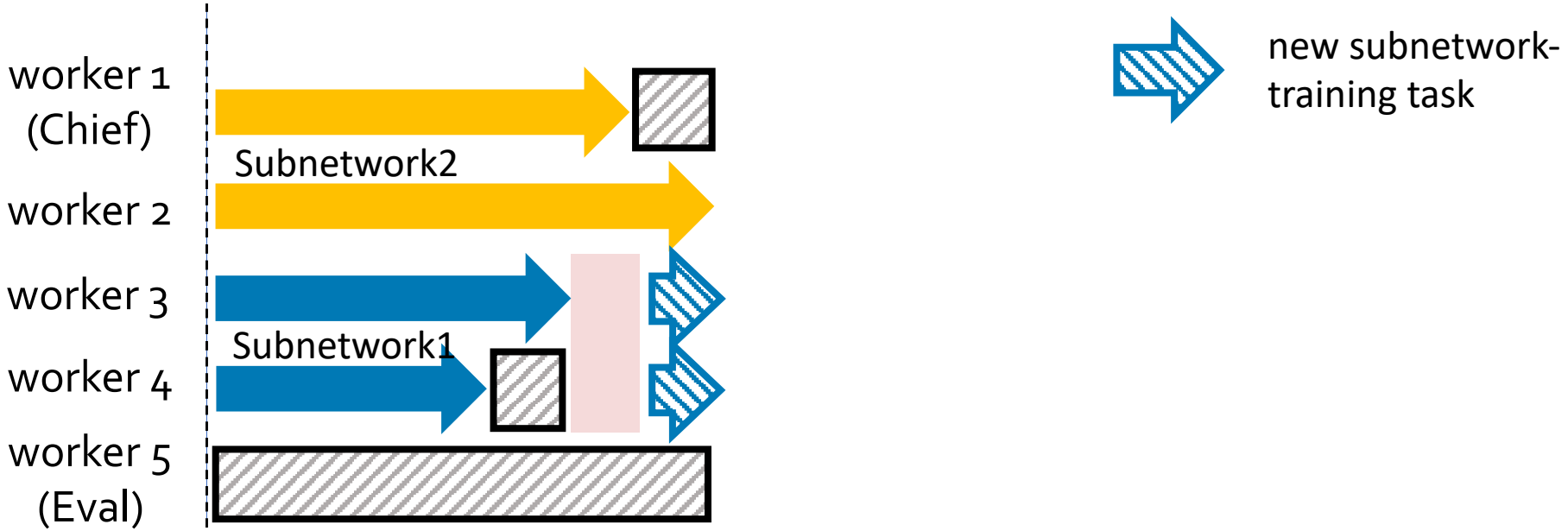
Observations

- Many workers involved in the training of one subnetwork
  - Heavier straggler effect and longer synchronization time
- Resource hogging in the gap between the training of subnetworks.



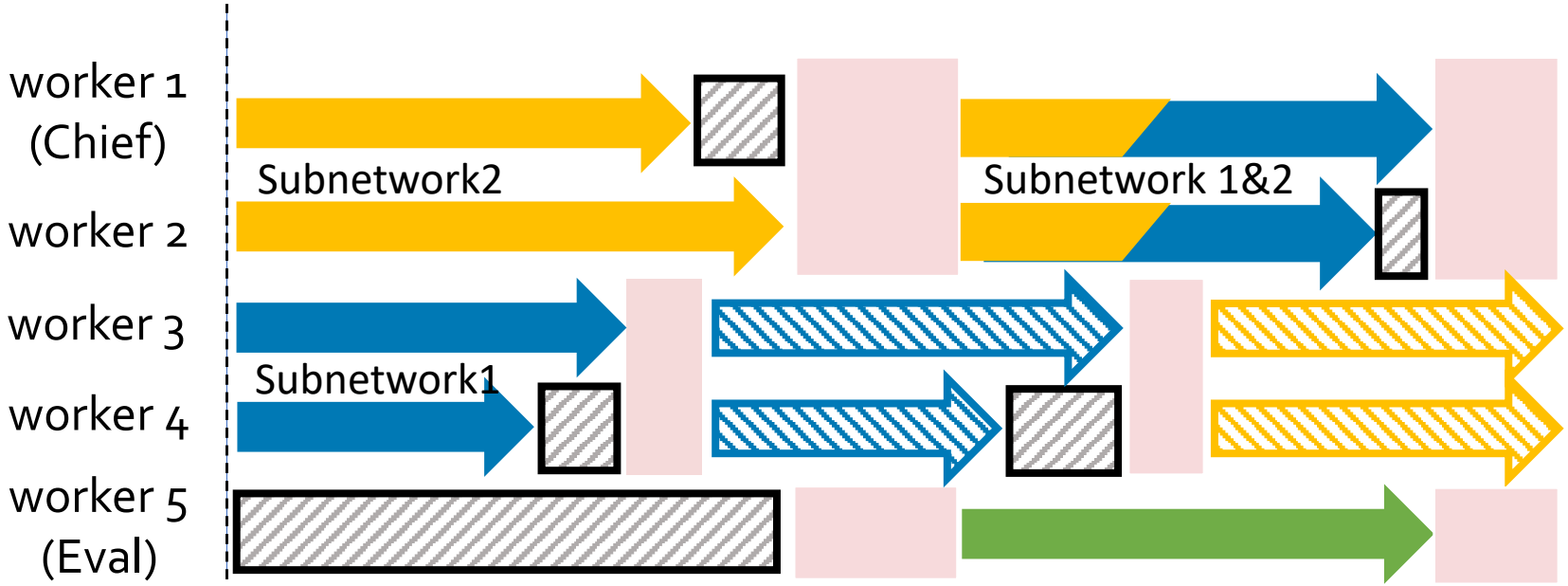
Reduce Resource Hogging

- Workers involved in one subnetwork-training task never wait for other workers to finish before they are scheduled to a new subnetwork-training task.



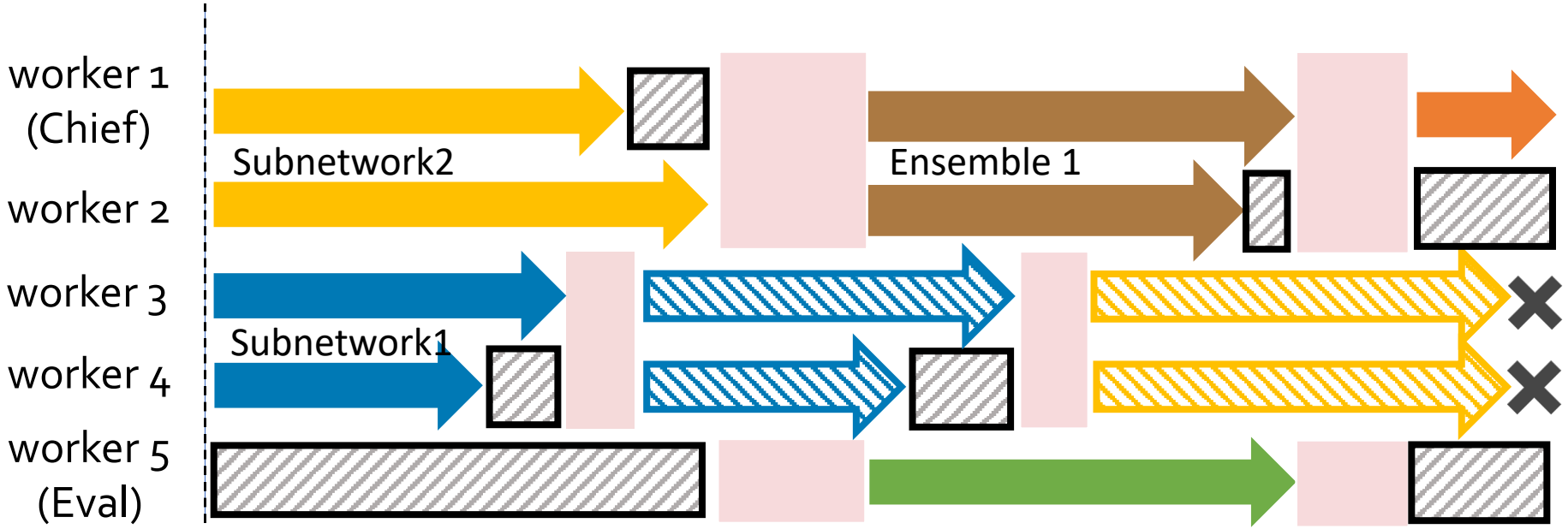
Reduce Resource Hogging

- When all subnetworks of current iteration are finished, start building the ensemble.



Reduce Resource Hogging

- After evaluation, Chief decides subnetworks for next iteration.
  - **Keep luck guess**
  - **Throw away bad guess**



Results

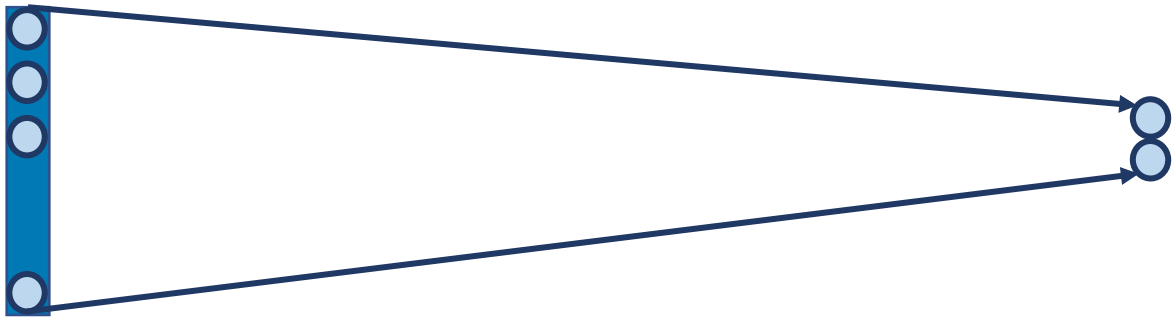
Resources

- All of the following experiments are conducted on ***mlearn-alpha*** by:
  - Training with 20 workers, each with 32 GiB RAM
  - Validation with 20 workers, each with 8 GiB RAM

jymbii: Logistic Regression

Model	Steps	Training Time	Eval Time	AUC
Logistic Regression	120,232	105 mins	20 mins	66.3%

Optimizer: 1e-1 Decay LR SGD
Reg: L2, 1e-4



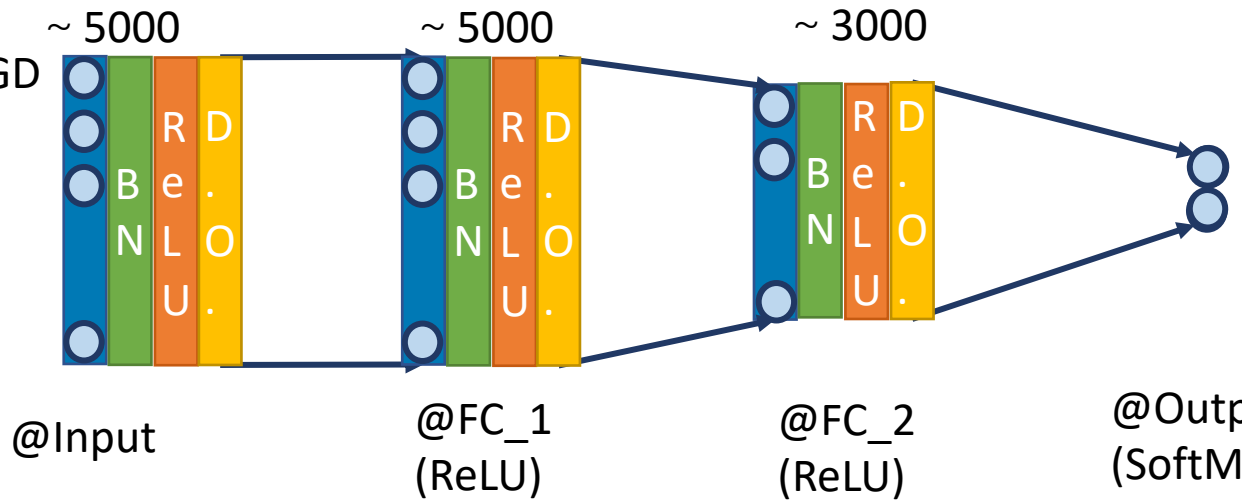
@Input

@Output
(SoftMax)

jymbii: Best NN with Grid Search

Model	Steps	Training Time	Eval Time	AUC
Logistic Regression	120,232	105 mins	20 mins	66.3%
Grid Search_BST	120,232	155 mins	31 mins	67.2%

Optimizer: 1e-1 Decay LR SGD
Reg: L2, 1e-4
Act: ReLU
Dropout: 0.1
BatchNorm
Layer_2 / # Layer_1 = 2/3

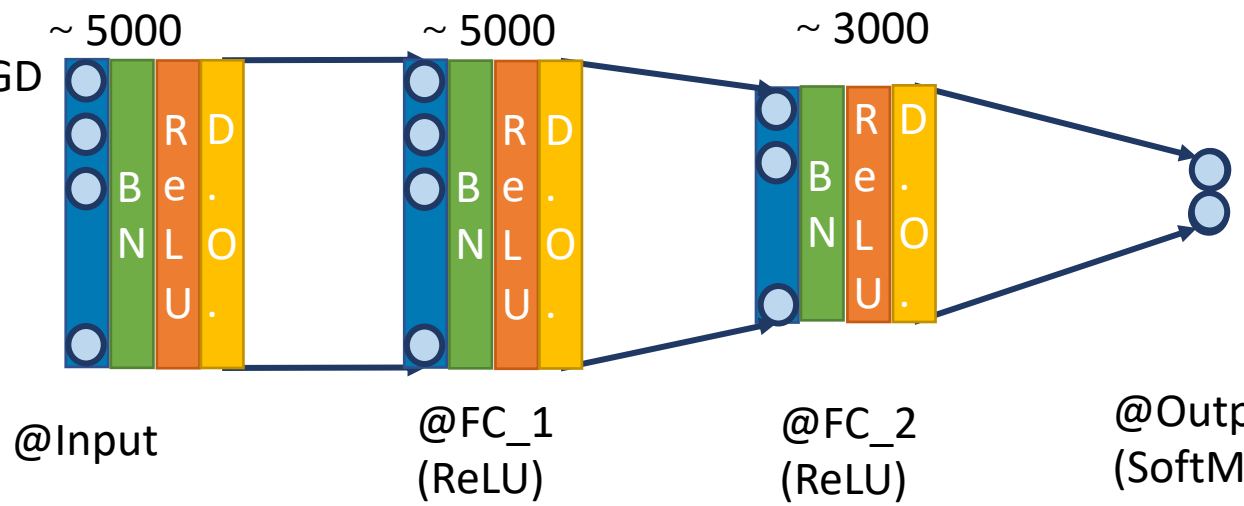


jymbii: Best NN with Grid Search

Model	Steps	Training Time	Eval Time	AUC
Logistic Regression	120,232	105 mins	20 mins	66.3%
Grid Search_BST	120,232	155 mins **(x12)**	31 mins	67.2%

Exploration

Optimizer: 1e-1 Decay LR SGD
Reg: L2, 1e-4
Act: ReLU
Dropout: 0.1
BatchNorm
Layer_2 / # Layer_1 = 2/3



jymbii: AdaNet

Model	Steps	Training Time	Eval Time	AUC
Logistic Regression	4,000	6 mins	19 mins	63.3%
Logistic Regression	120,232	105 mins	20 mins	66.3%
Grid Search_BST	4,000	8 mins	28 mins	64.5%
Grid Search_BST	120,232	155 mins **(x12)**	31 mins	67.2%
AdaNet	2,000 x2	36 mins	31 mins	**67.6%**

*Randomly down-sampled 4000 steps, 2 iterations for structure tuning.

pymk: Scalable Automated ML eases your way

Model	Steps	Training Time	Eval Time	AUC
Logistic Regression (Default Setting)	20,000	10 mins	4 mins	67.5%

Default setting for every parameter

pymk: Scalable Automated ML eases your way

Model	Steps	Training Time	Eval Time	AUC
Logistic Regression (Default Setting)	20,000	10 mins	4 mins	67.5%
Logistic Regression (Manual Tuned)	20,000	9 mins	3 mins	73.1%

Default setting for every parameter

I spent **hours** to manual tune it

pymk: Scalable Automated ML eases your way

Model	Steps	Training Time	Eval Time	AUC
Logistic Regression (Default Setting)	20,000	10 mins	4 mins	67.5%
Logistic Regression (Manual Tuned)	20,000	9 mins	3 mins	73.1%
Grid Search_BST	20,000	18 mins **(x4)**	6 mins	74.8%

Default setting for every parameter

I spent **hours** to manual tune it

Grid search spend around **1 hour**

pymk: Scalable Automated ML eases your way

Model	Steps	Training Time	Eval Time	AUC
Logistic Regression (Default Setting)	20,000	10 mins	4 mins	67.5%
Logistic Regression (Manual Tuned)	20,000	9 mins	3 mins	73.1%
Grid Search_BST	20,000	18 mins **(x4)**	6 mins	74.8%
AdaNet	10,000 x2	17 mins	4 mins	**76.1%**

Default setting for every parameter

I spent **hours** to manual tune it

Grid search spend around **1 hour**

As for AdaNet, I just run the code. And after **20 minutes**…

Summary

Motivation

Tuning a Neural Network is exhausting.

Algorithm

Select optimal subnetworks and implement an adaptive learning process to attain an ensemble with AdaNet

Implementation

Implement a network-level asynchronous distributed learning strategy to maintain the feasibility of scalable AdaNet

Results

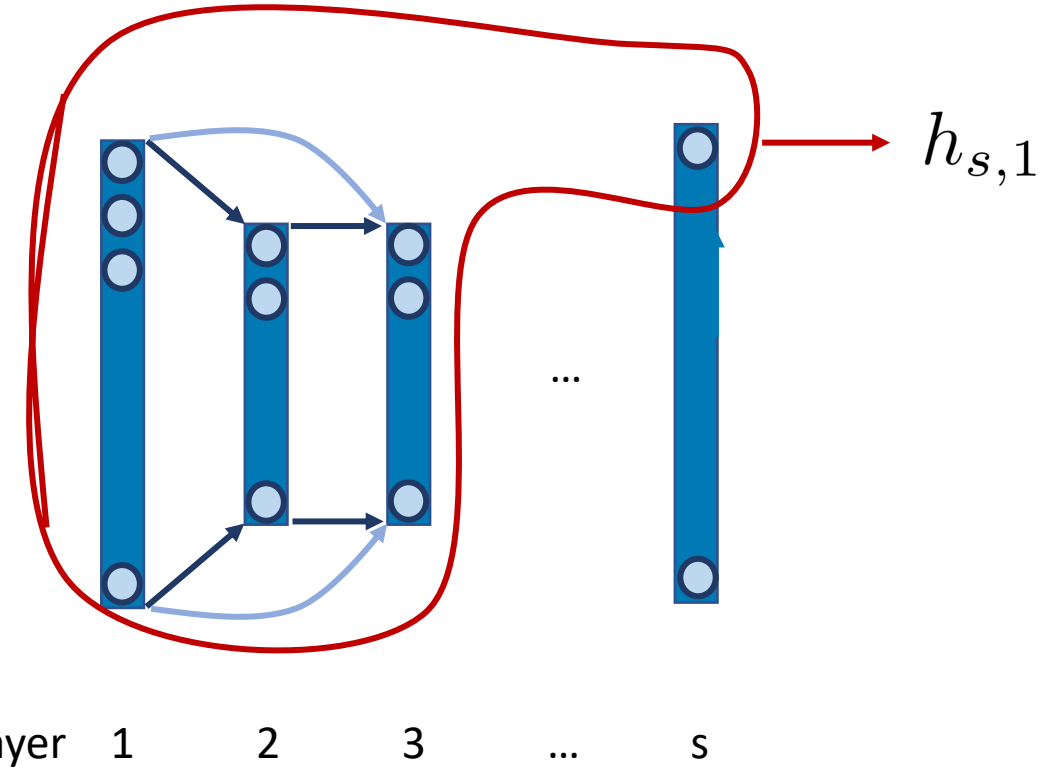
Achieve promising results on *Job You May Be Interested In(jymbii)* dataset and *People You May Know(pymk)* dataset

Thank you soooooo much !

Back-up

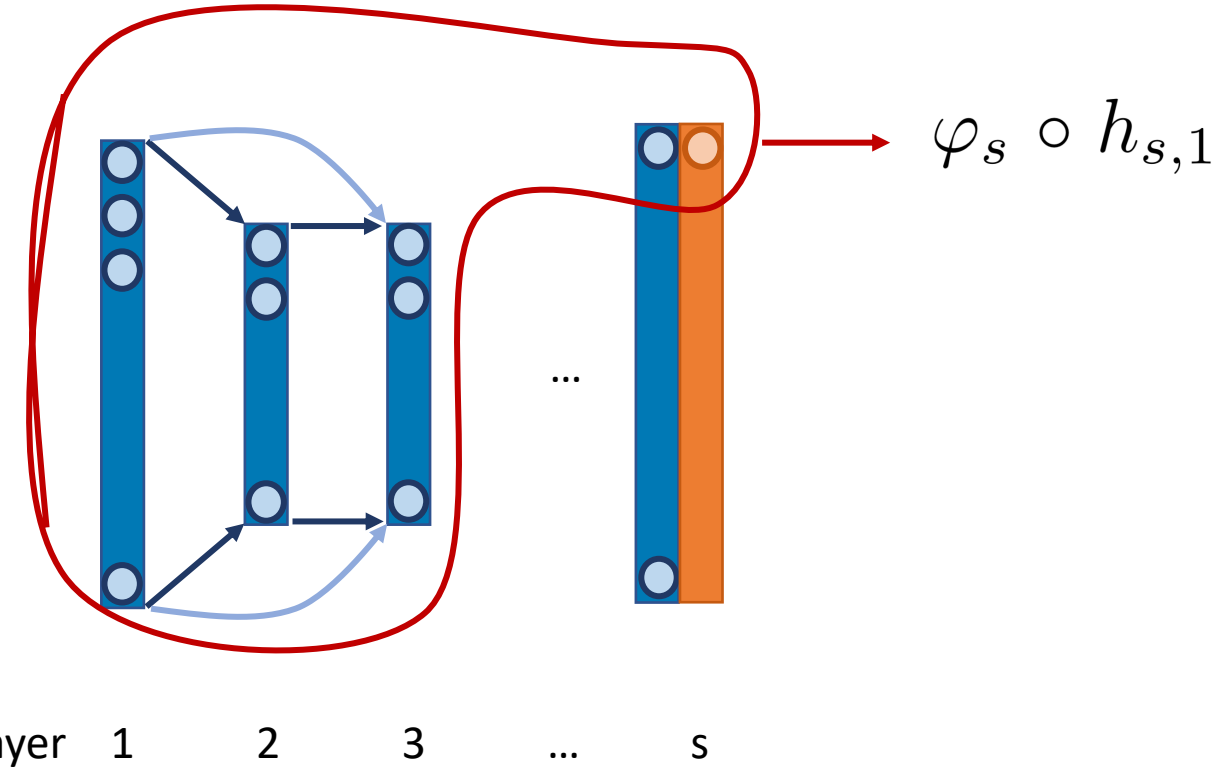
AdaNet - Theory

- Hidden Layer Family $\mathcal{H} \longrightarrow \mathbf{h}_s \longrightarrow h_{s,n_s}$



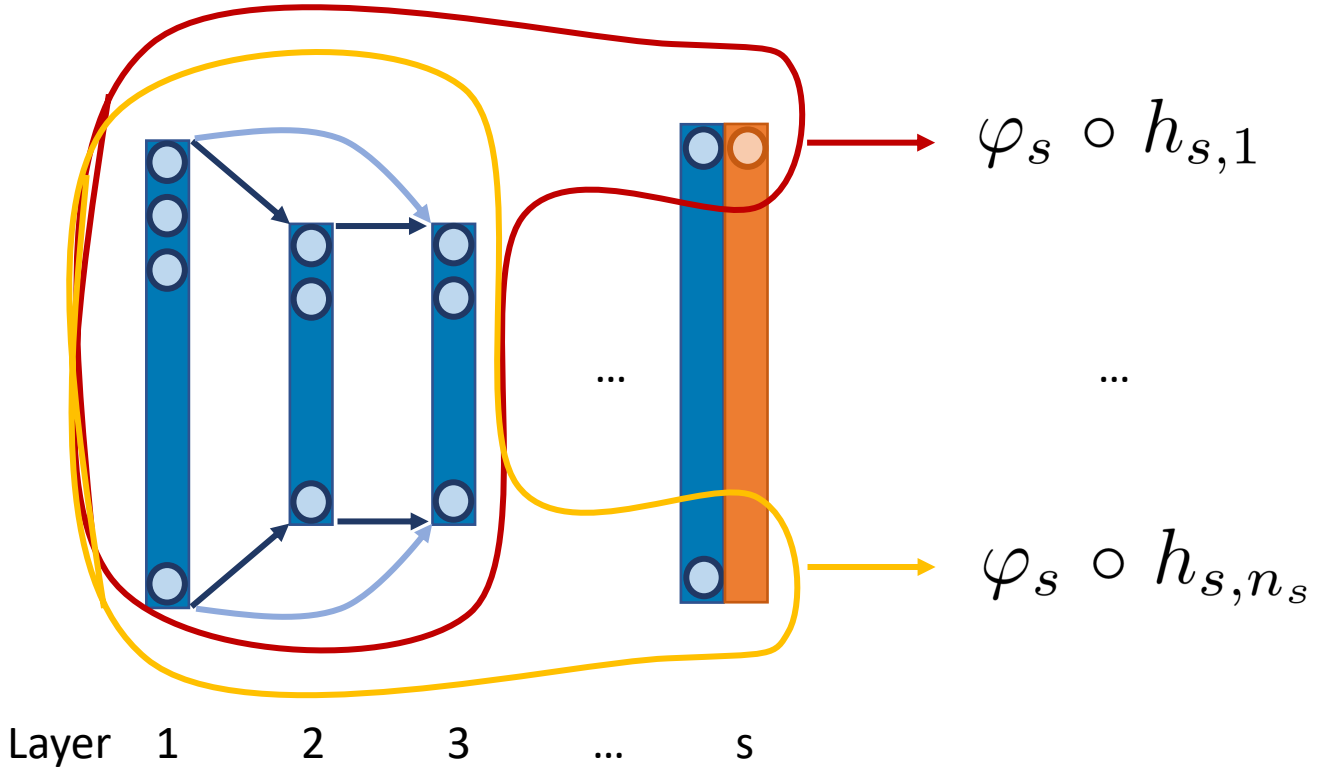
AdaNet - Theory

- Hidden Layer Family $\mathcal{H} \longrightarrow \mathbf{h}_s \longrightarrow h_{s,n_s}$



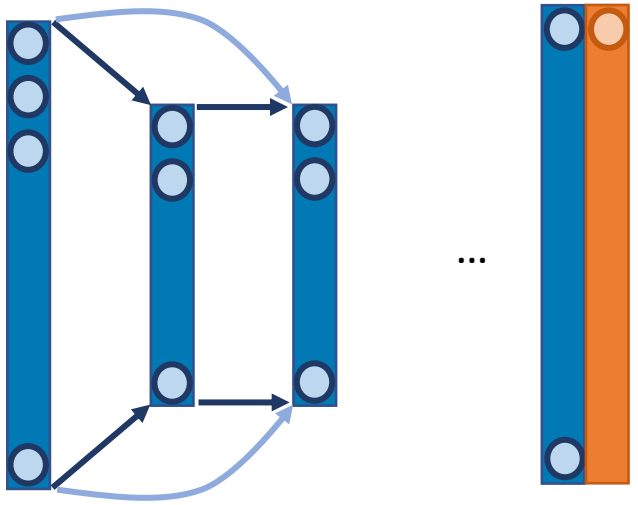
AdaNet - Theory

- Hidden Layer Family $\mathcal{H} \longrightarrow \mathbf{h}_s \longrightarrow h_{s,n_s}$



AdaNet - Theory

- Hidden Layer Family $\mathcal{H} \longrightarrow \mathbf{h}_s \longrightarrow h_{s,n_s}$



Layer 1 2 3 ... s

$$\varphi_s \circ \mathbf{h}_s = (\varphi_s \circ h_{s,1}, \ldots, \varphi_s \circ h_{s,n_s})$$

AdaNet - Theory

- Hidden Layer Family $\mathcal{H} \longrightarrow \mathbf{h}_s \longrightarrow h_{s,n_s}$

Layer k

$$\mathcal{H}_k = \left\{ x \mapsto \sum_{s=1}^{k-1} \mathbf{u}_s \cdot (\varphi_s \circ \mathbf{h}_s)(x) \right\}$$

Weights

The high abstraction of Hidden Layer S

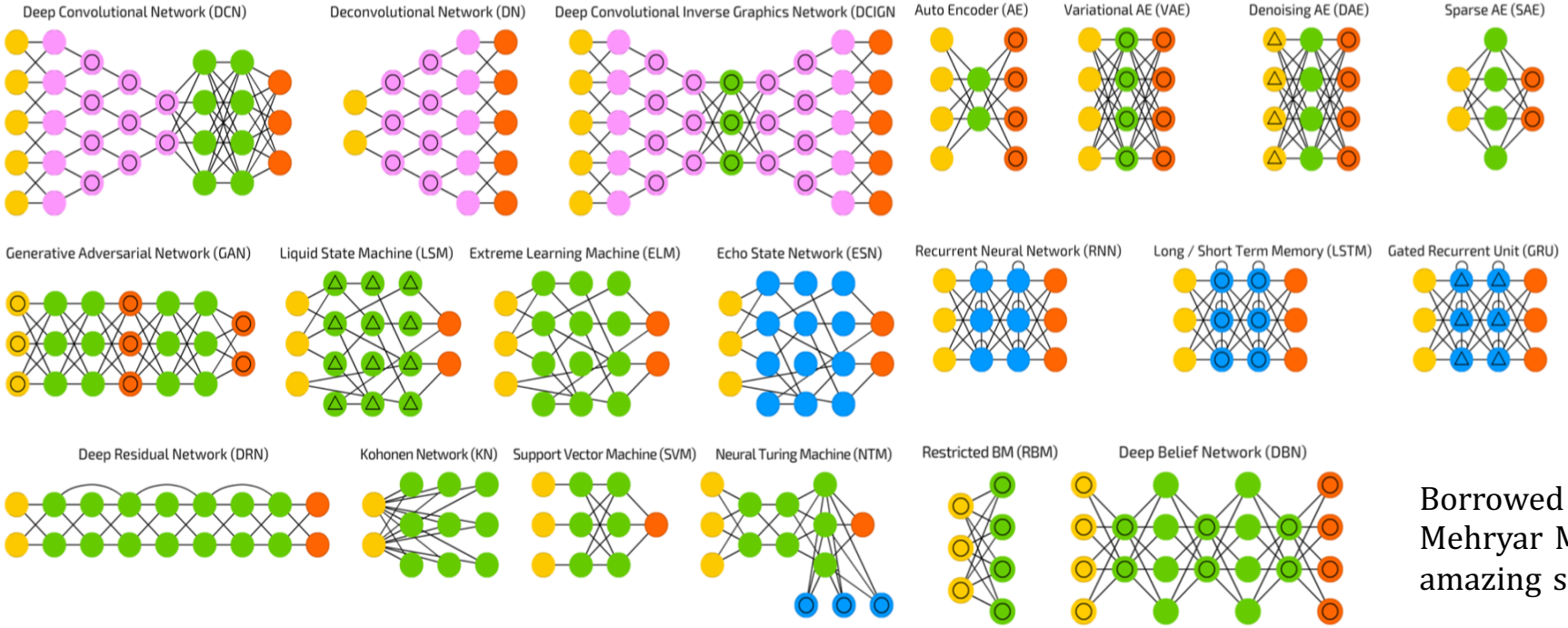
The activation function of Hidden Layer S

AdaNet - Theory

- Hidden Layer Family $\mathcal{H} \longrightarrow \mathbf{h}_s \longrightarrow h_{s,n_s}$

(Almost)

Every network could be presented as a member of Hidden Layer Family.



Borrowed from Mehryar Mohri's amazing slides

AdaNet - Theory

- Hidden Layer Family $\mathcal{H} \longrightarrow \mathbf{h}_s \longrightarrow h_{s,n_s}$
- Objective Function with Hidden Layer Family

Hidden Layer Functions from Layer 1 to N

$$F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}\Phi\left(1 - y_i\sum_{j=1}^{N} w_j h_j\right) + \sum_{j=1}^{N}\Gamma_j|w_j|$$

"Loss" **Weights** **Regularization**

AdaNet - Algorithm

- **Iteratively "ensemble" new subnetworks to the current network**
- During the iteration t
  - "Ensemble" a new candidate subnetwork **u**
  - Minimize the objective function to train it

Current network **New candidate subnetwork**

$$F_t(\mathbf{w}, \mathbf{u}) = \frac{1}{m} \sum_{i=1}^{m} \Phi\left(1 - y_i f_{t-1}(x_i) - y_i \mathbf{w} \cdot \mathbf{u}(x_i)\right) + \Gamma_{\mathbf{u}} \|\mathbf{w}\|_1$$

Ensemble mixture weights

AdaNet - Algorithm

- **Iteratively "ensemble" new subnetworks to the current network**
- During the iteration t
  - "Ensemble" a new candidate subnetwork **u** *"under specific rules"*
  - Minimize the objective function to train it
  - **Choose the one that leading to the best reduction of objective function, i.e.**

$$\text{if } \min_{\mathbf{w}} F_t(\mathbf{w}, \mathbf{h}) \leq \min_{\mathbf{w}} F_t(\mathbf{w}, \mathbf{h}'), \text{ then}$$

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathbb{R}^B}{\operatorname{argmin}} F_t(\mathbf{w}, \mathbf{h}), \quad \mathbf{h}_t = \mathbf{h}$$

And otherwise

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathbb{R}^B}{\operatorname{argmin}} F_t(\mathbf{w}, \mathbf{h}'), \quad \mathbf{h}_t = \mathbf{h}'$$

Acknowledgement