Microsoft

Understanding and improving Deep Neural Network from neuron-level perspective

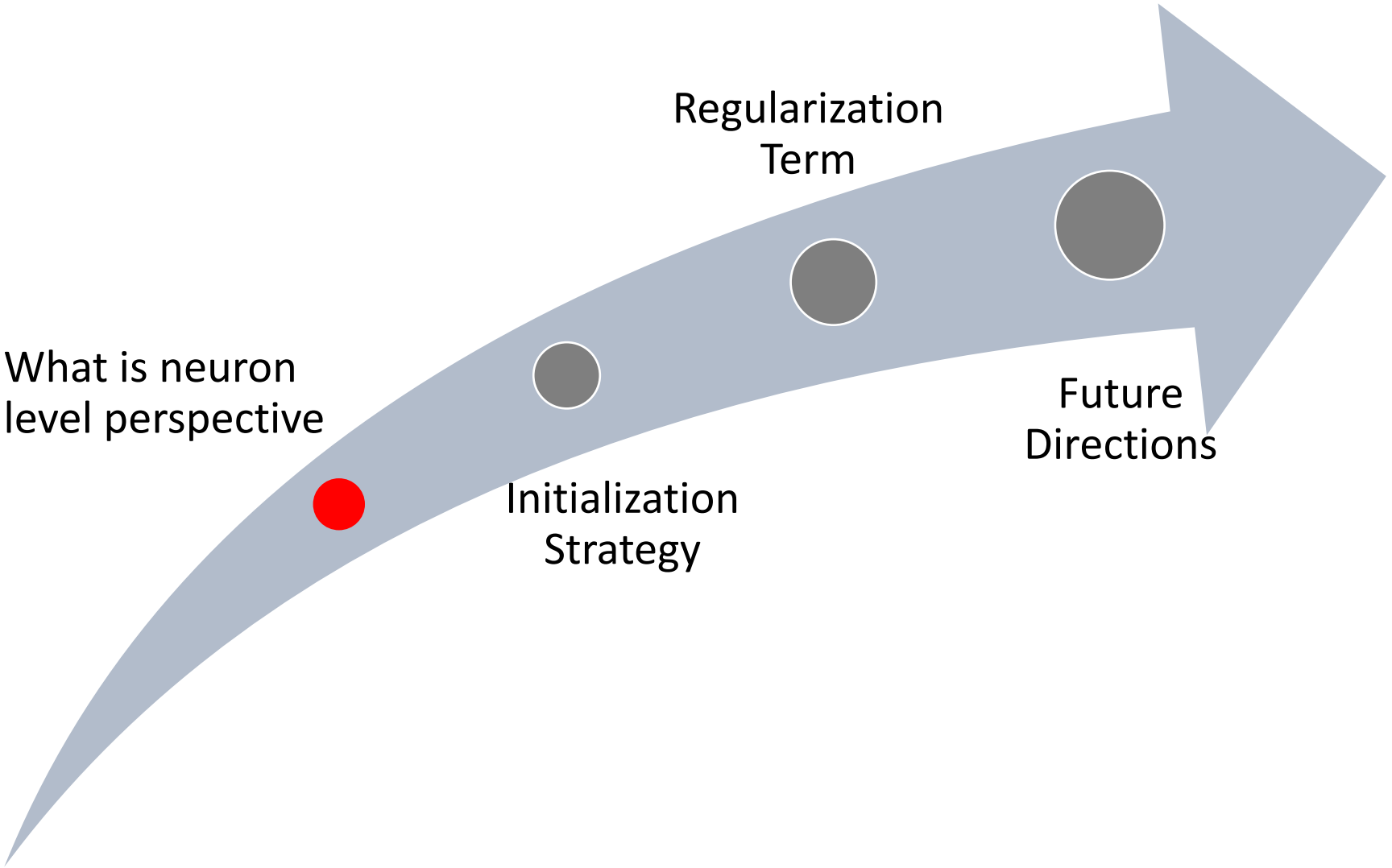
Haitao Mao

Self Introduction

- UESTC(18-22)->MSU(22-)
- Research Interests:
  - Practical challenge in data mining
  - Understanding DNN, typically GNN, towards better design
- Previous works
  - Neuron with Steady Response Leads to Better Generalization
  - Neuron Campaign for Initialization Guided by Information Bottleneck Theory (CIKM2021 best short paper)
  - Graph Neural Networks as Multi-view Learning
  - A large scale Search Dataset for Unbiased Learning to Rank (top-5 score paper in NeurIPS dataset track) (WSDM CUP, ongoing now!)
  - Whole Page Unbiased Learning to Rank
  - Source Free Graph Unsupervised Domain Adaptation

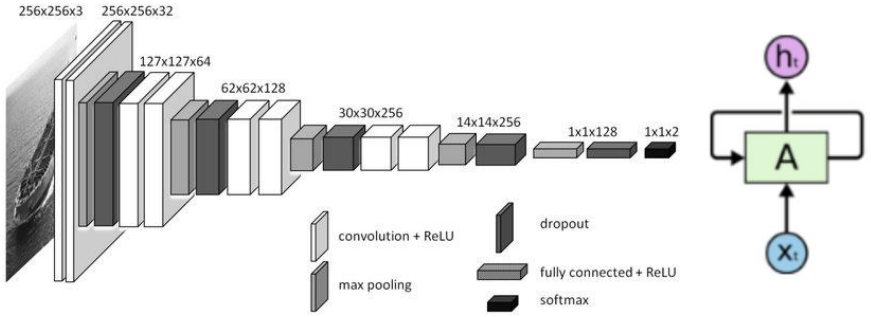
Outline

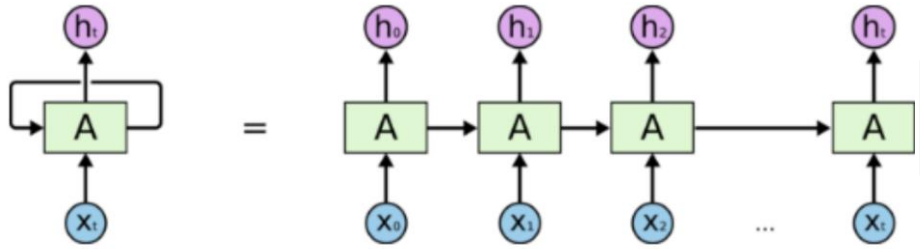
- What is the neuron-level perspective?
- Neuron-level perspective for Initialization Strategy.
  - -- Neuron Campaign for Initialization Guided by Information Bottleneck Theory
- Neuron-level perspective for Regularization Term.
  - -- Neuron with Steady Response Leads to Better Generalization
- Future direction

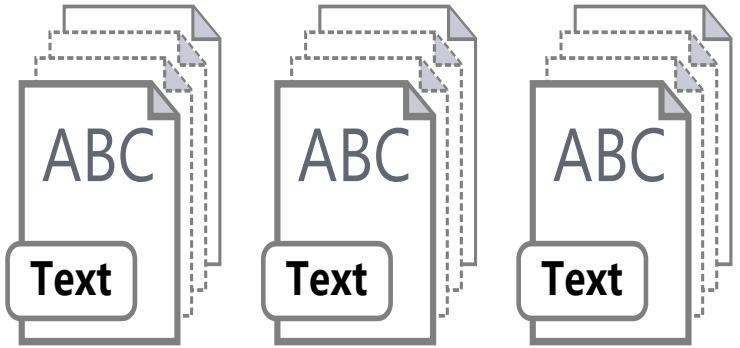

Regularization
Term
What is neuron
level perspective
Initialization
Strategy
Future
Directions

Our goal!!

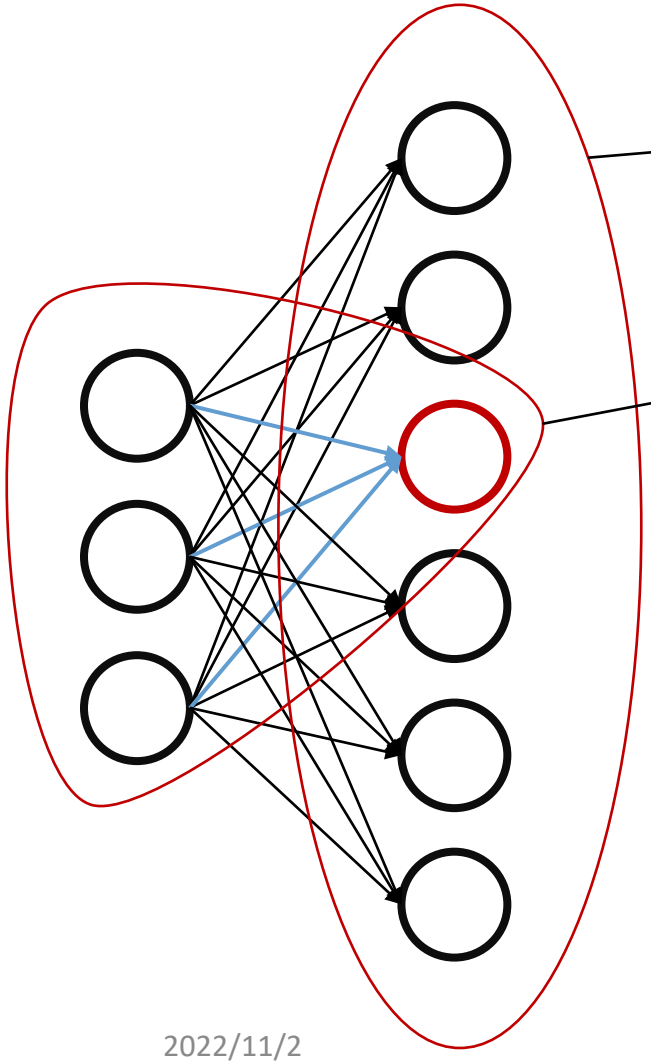
Domain-generic & Architecture-agnostic







What is Neuron-level Perspective?



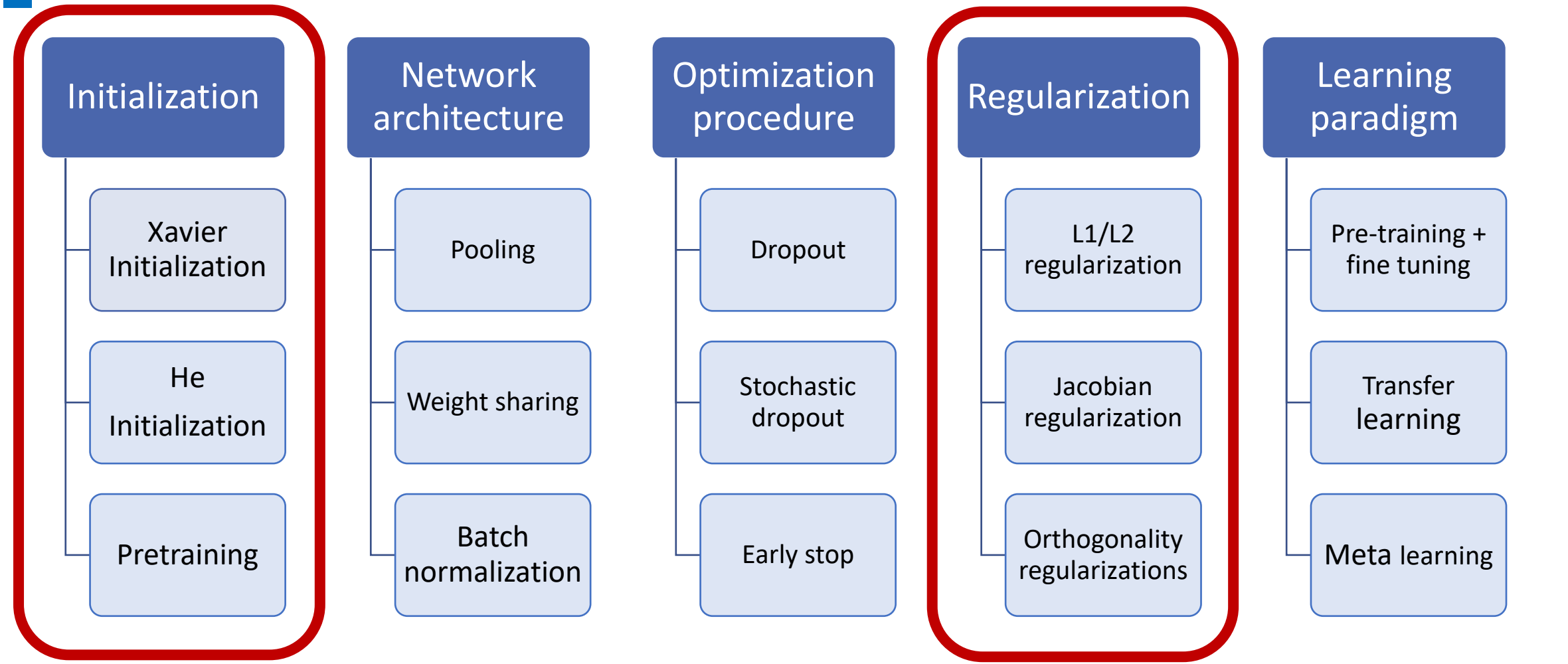
One layer MLP

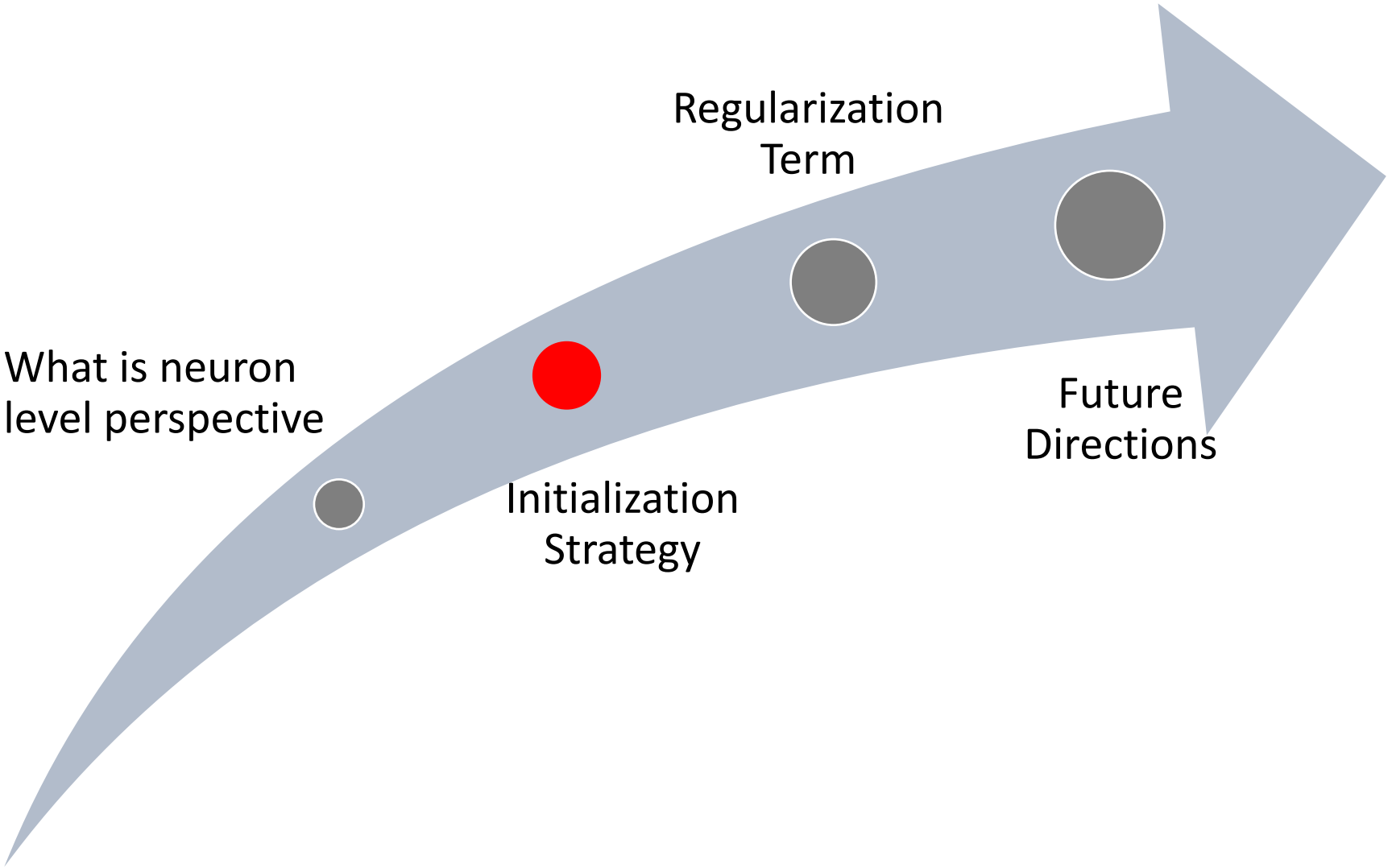
A single neuron

- As simple as perceptron
- Basic building block for DNN

Neuron-level perspective provides a new way to understand and design Neuron Network!

What can neuron-level perspective help to design?




Regularization
Term
What is neuron
level perspective
Initialization
Strategy
Future
Directions

Microsoft

Neuron Campaign for Initialization Guided by Information Bottleneck Theory

Haitao Mao[2]
Joint work with Qiang Fu[1], Lun Du[1], Xu Chen[1], Shi Han[1], Dongmei Zhang[1]

1. Microsoft Research Asia
2. Michigan State University

Previous Initialization: a layer perspective

Example: Xavier Initialization (A layer-wise manner)

$$y_l = W_l x_l + b_l$$

Target: $n_i Var[W_l] = 1$ and $n_{i+1} Var[W_l] = 1$

Final form: $W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}\right]$

Multiple Layer NN can be initialized layer-wise!

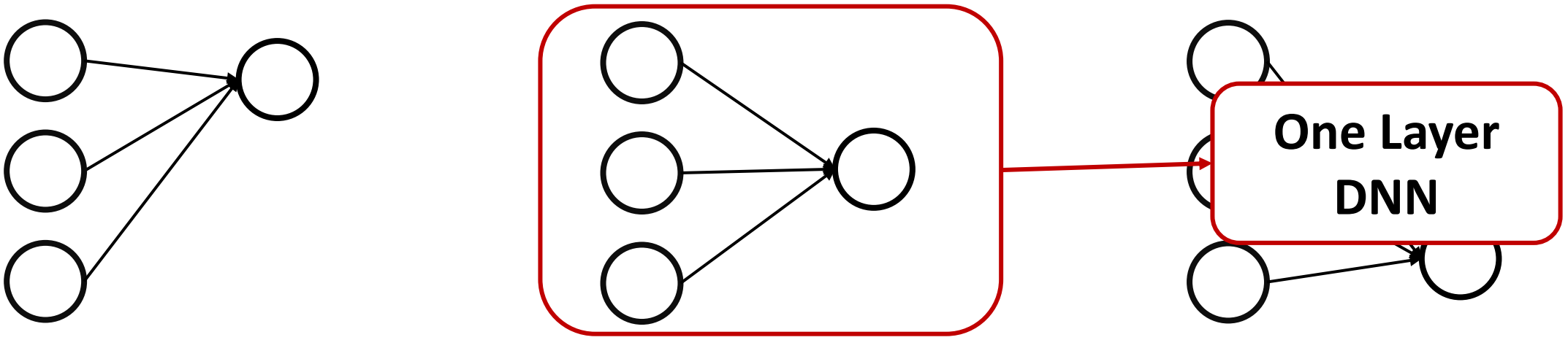
Microsoft

Can a single layer NN initialize neuron by neuron!

Initialization: a neuron perspective

Initialize neuron -> Combine neurons

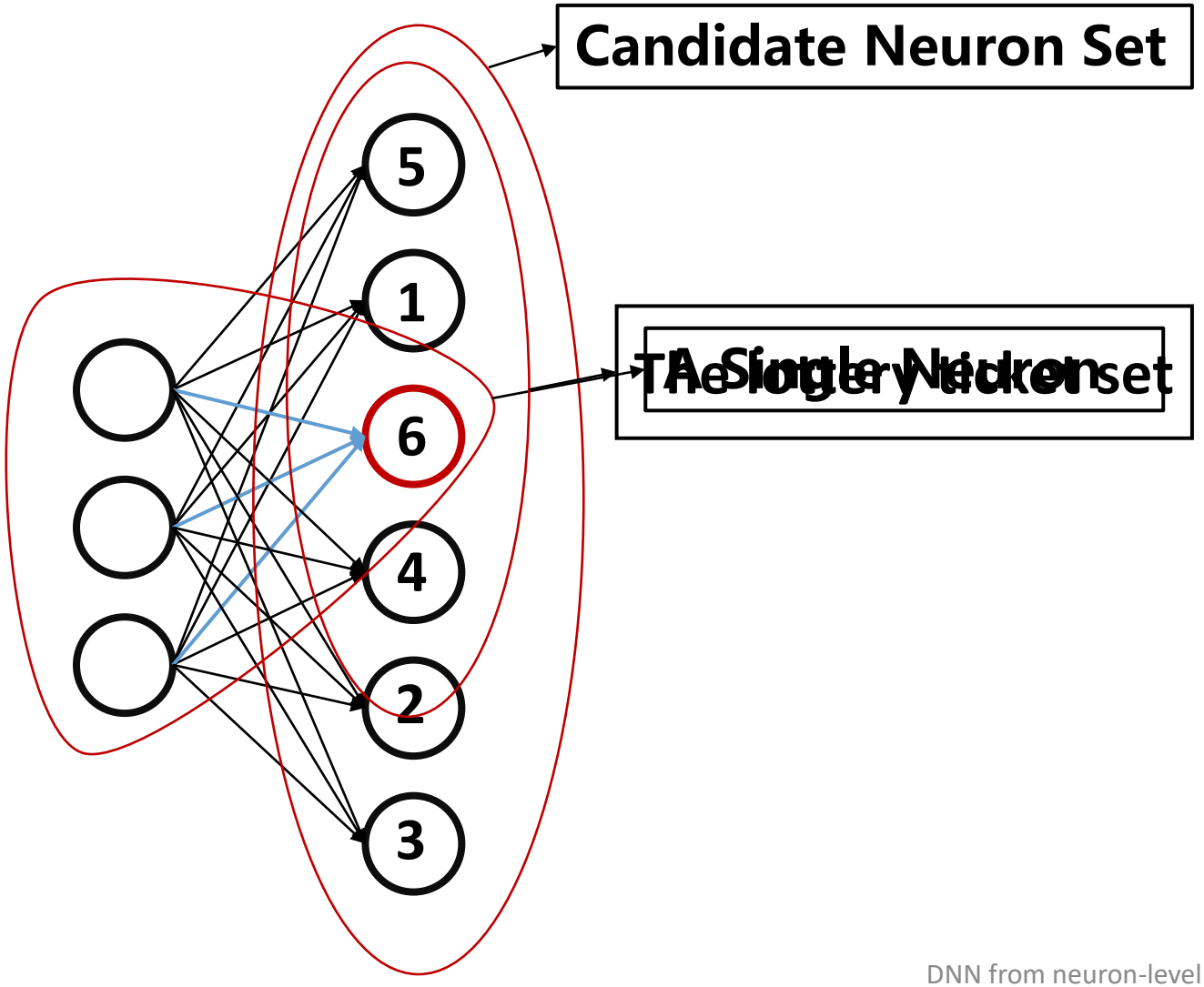
Randomness can be found in each Neuron!



Randomness in Initialization like lottery ticket!

More tickets! More wins!

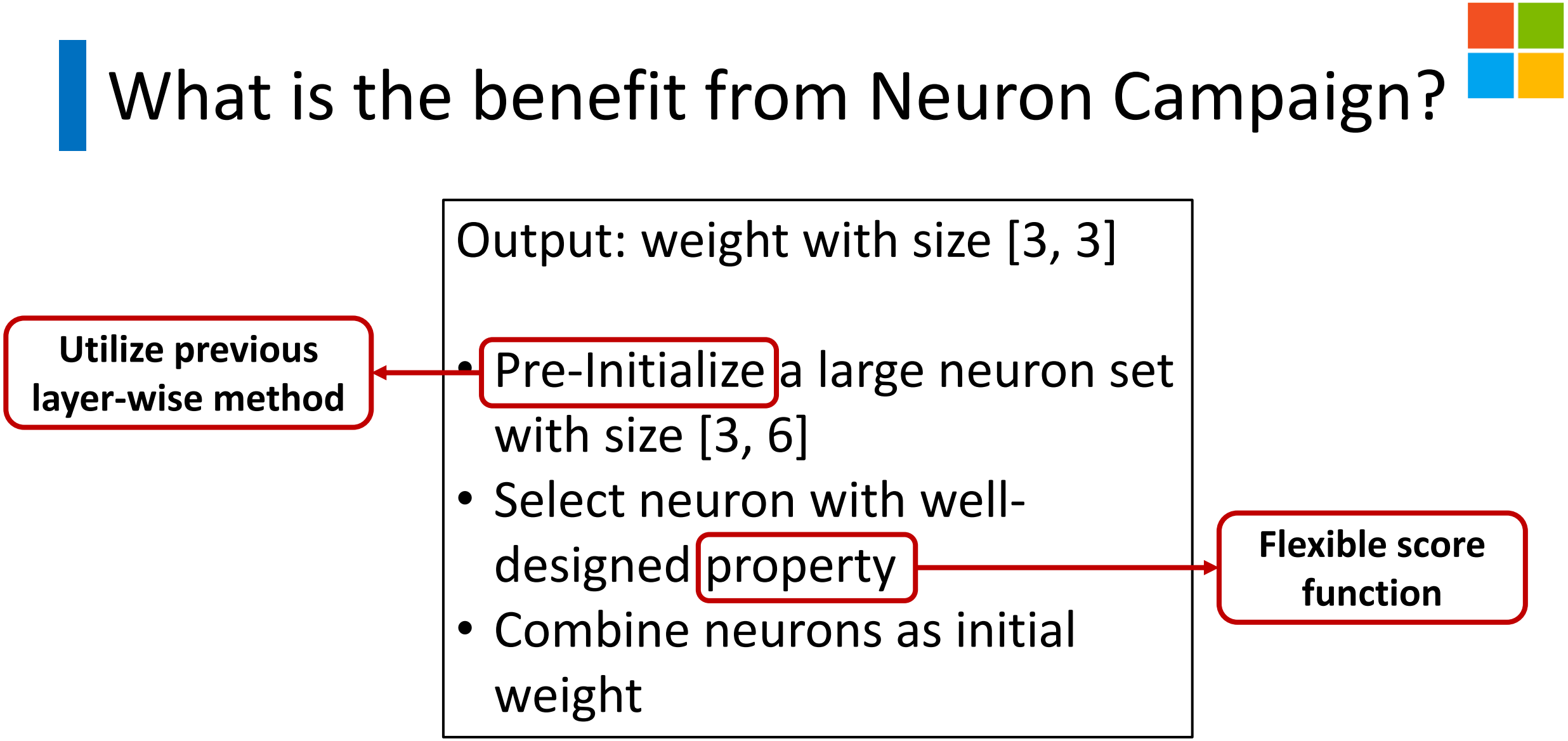
Neuron Campaign Initialization algorithm



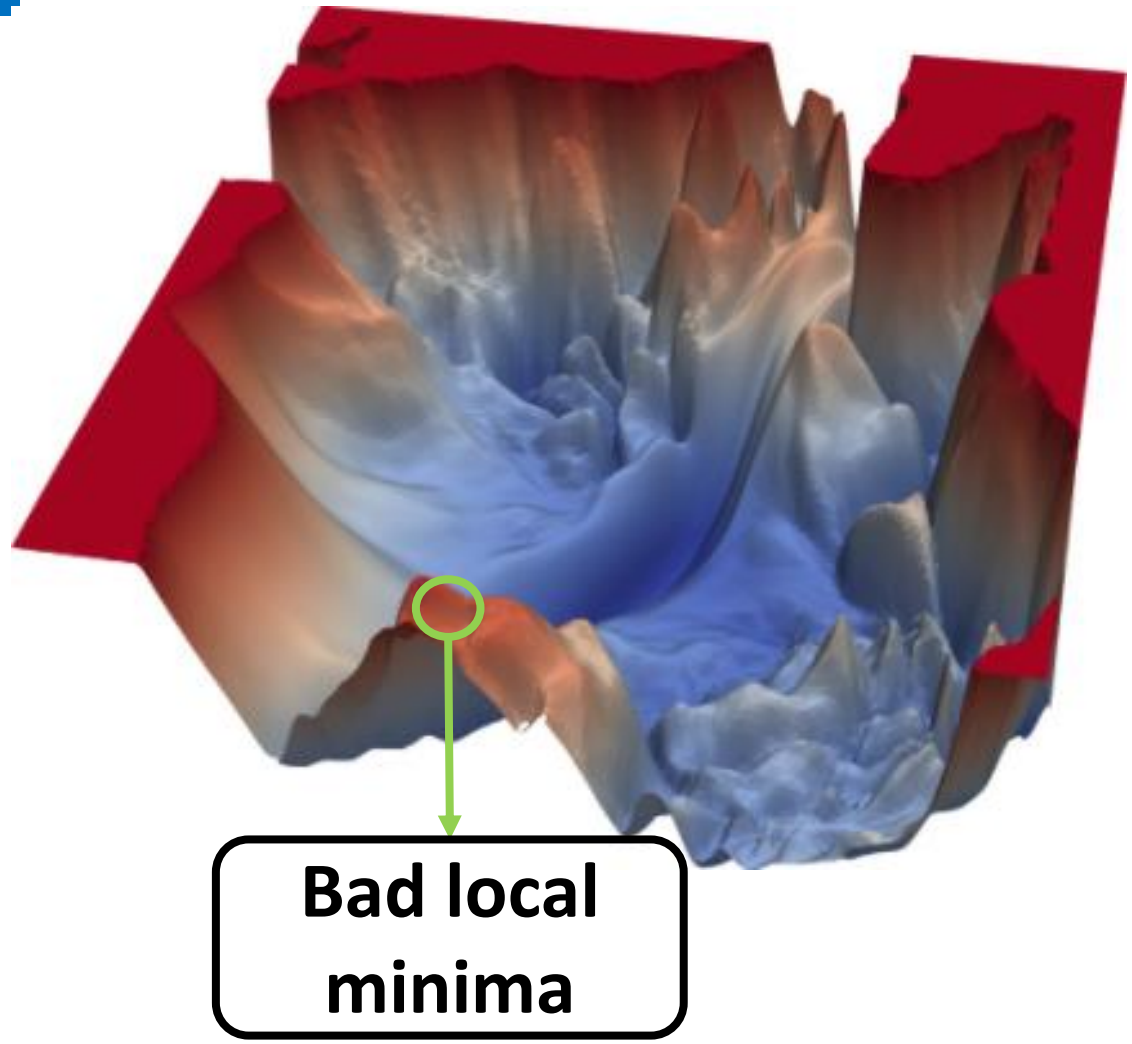
Output: weight with size [3, 3]

- Pre-Initialize a large neuron set with size [3, 6]
- Select neuron with well-designed property
- Combine neurons as initial weight

What is the benefit from Neuron Campaign?



What we want for initialization?



- A bad initialization may lead to
  - No local minima
  - Stuck in a bad local minima.
- A good initialization should
  - Find a local minima
  - Find a good local minima with better generalization

No local minima: Gradient vanish/explode

Random Initialization

$W \sim N(0, 0.01^2)$

```
x = torch.randn(512)

for i in range(100):
    a = torch.randn(512,512) * 0.01
    x = a @ x
x.mean(), x.std()
```

```
(tensor(0.), tensor(0.))
```

$W \sim N(0, 1)$

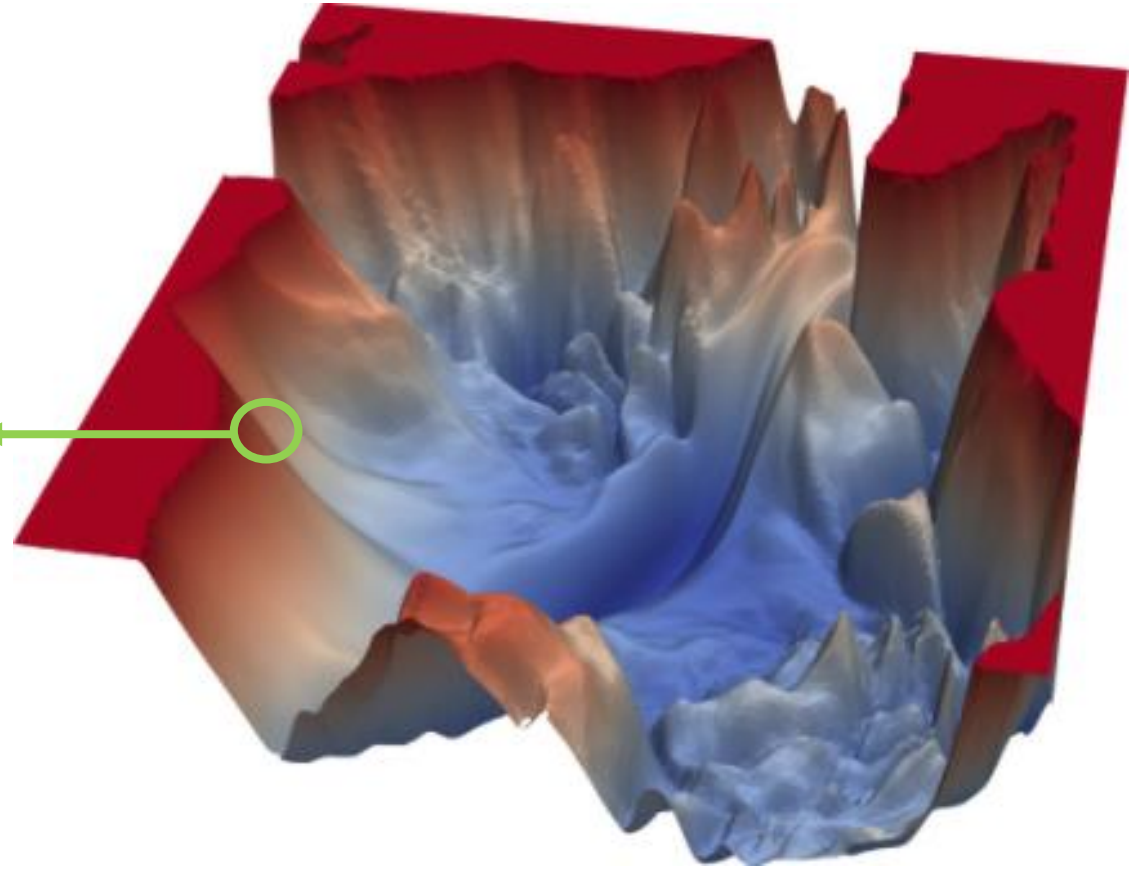
```
x = torch.randn(512)

for i in range(100):
    a = torch.randn(512,512)
    x = a @ x
    if torch.isnan(x.std()): break
i
```

```
28
```

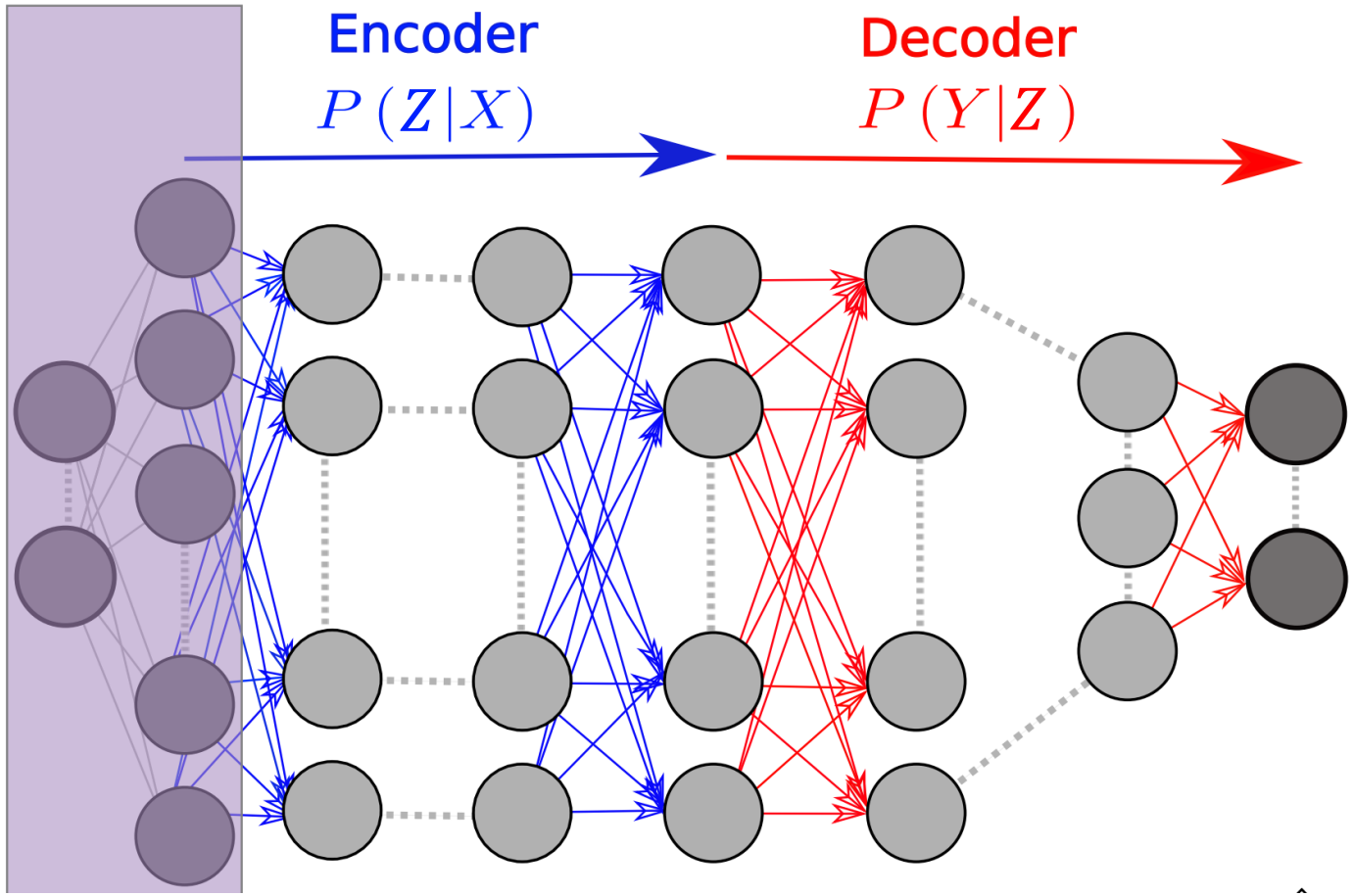
How to find initialization for good generalization is still open problem!

Initialization for better generalization.

One clue: avoid weight fluctuation in training procedure



Information Bottleneck Theory and measurement



- Input information maintenance:
  $I(X; Z_i)$
- Target-related information enhancement
  $I(Z_i; Y)$
- Criterion:
  $\alpha I(X; Z_i) + (1 - \alpha) I(Z_i; Y)$

Experimental details

layers	Hidden Layer Dimension
2	784, 100, 10
3	784, 256, 100, 10
5	784, 32, 32, 32, 32, 10

Table 1: minimal error rate and corresponding epoch comparison of IBCI with baseline methods on MNIST.

Strategy	Layers	Vanilla	LSUV	**IBCI**
Xavier	2	2.04 ± 0.03 (75)	2.05 ± 0.06 (51)	**1.93 ± 0.06 (60)**
	3	1.82 ± 0.05 (52)	1.80 ± 0.07 (63)	**1.71 ± 0.09 (36)**
	5	2.83 ± 0.16 (98)	3.13 ± 0.17 (69)	**2.53 ± 0.09 (78)**
He	2	2.03 ± 0.03 (65)	2.00 ± 0.04 (70)	**1.93 ± 0.07 (57)**
	3	1.83 ± 0.05 (54)	1.86 ± 0.07 (71)	**1.73 ± 0.04 (35)**
	5	2.76 ± 0.07 (80)	2.90 ± 0.12 (77)	**2.62 ± 0.08 (73)**

Ablation Study

Table 2: minimal error rate and corresponding epoch comparison of IBCI with methods with only one criterion.

Strategy	Layers	IBCI	TIE	IIM
Xavier	2	**1.93 ± 0.06 (60)**	2.04 ± 0.07 (58)	2.07 ± 0.09 (84)
	3	**1.71 ± 0.09 (36)**	1.82 ± 0.03 (43)	1.82 ± 0.05 (52)
	5	**2.53 ± 0.09 (78)**	2.68 ± 0.05 (82)	2.57 ± 0.09 (84)
He	2	**1.93 ± 0.07 (57)**	2.07 ± 0.06 (59)	2.034 ± 0.09 (62)
	3	**1.73 ± 0.04 (35)**	1.83 ± 0.07 (42)	1.856 ± 0.05 (55)
	5	**2.62± 0.08 (73)**	2.89 ± 0.11 (74)	2.67 ± 0.12 (86)

Target Information Enhancement, i.e., IBCI without IIM

Input Information Maximization , i.e., IBCI without TIE

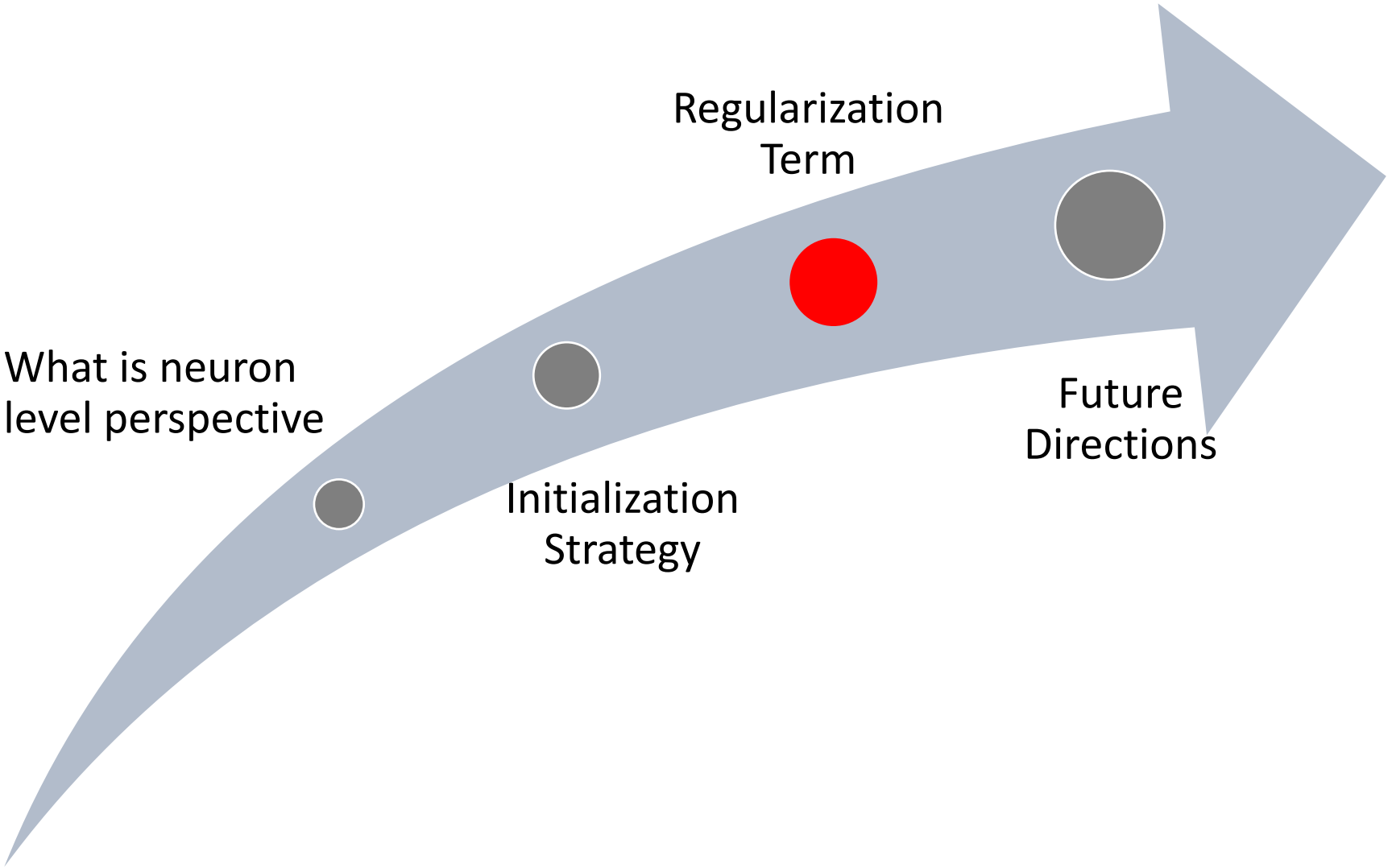
A little more detail in TIE

$I(Z_i; Y)$ target-related maintenance criterion

$$tr\left(\hat{H}\hat{H}^T\right) - \frac{1}{N}\sum_{j=0}^{N} tr\left(\hat{Z}_i^T \Pi^j \hat{Z}_i\right)$$

Inter-class variance

Intra-class variance


Regularization
Term
What is neuron
level perspective
Initialization
Strategy
Future
Directions

Microsoft

Neuron With Steady Response Leads to Better Generalization

Haitao Mao[2]
Joint work with Qiang Fu[1], Lun Du[1], Xu Chen[1], Wei Fang[3], Shi Han[1], Dongmei Zhang[1]

1. Microsoft Research Asia
2. Michigan State University
3. Tsinghua University

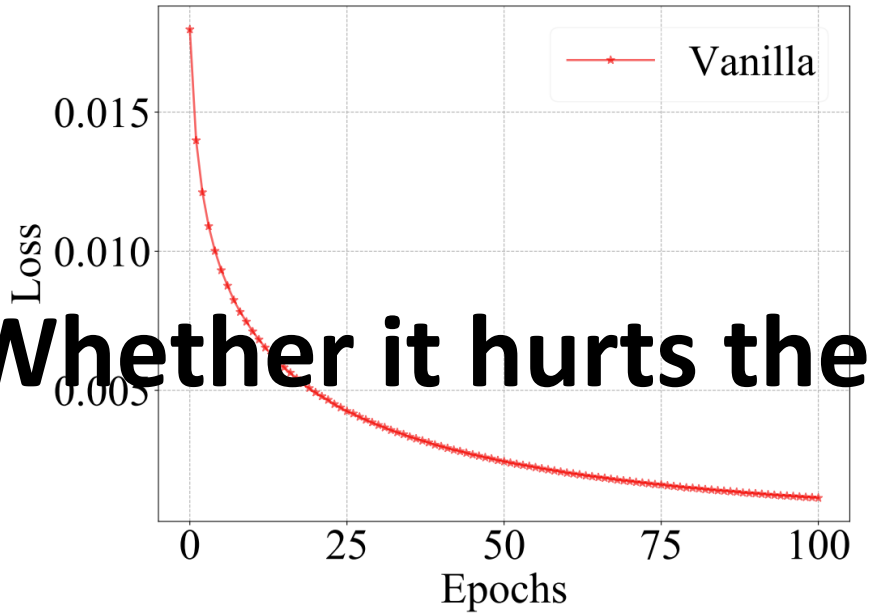
Microsoft

Can we explore training dynamic from

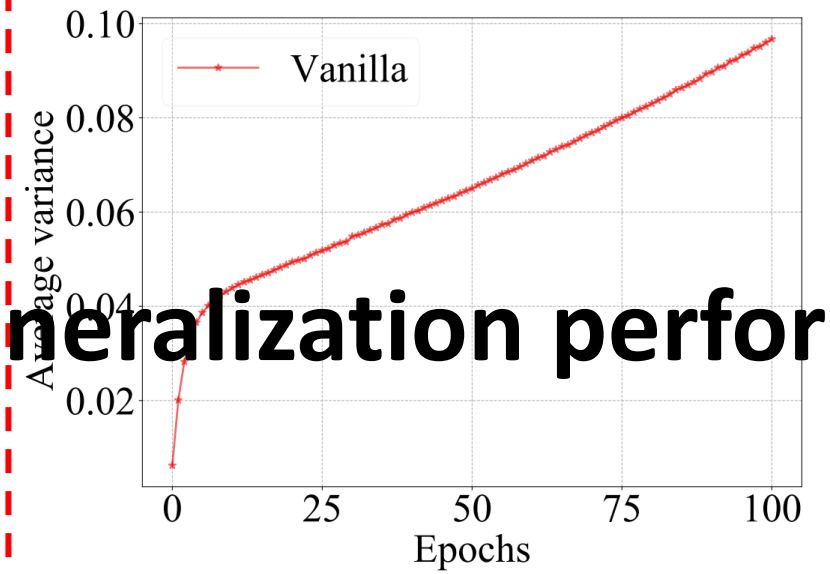
neuron-level perspective?

Observation 1: train phenomenon

Ascending Variance!



(a) Training cross-entropy loss



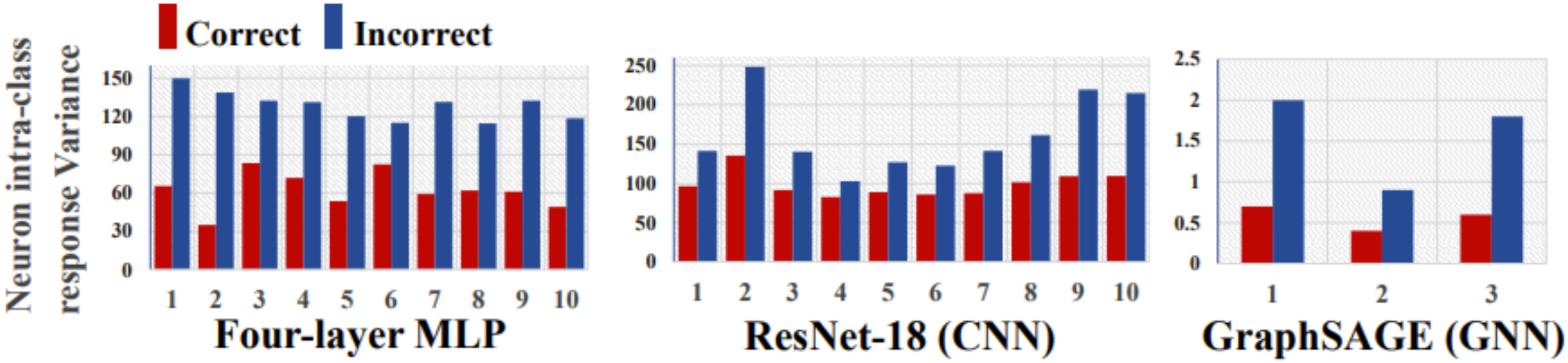
(b) Training intra-class variance

Whether it hurts the generalization performance?

Training procedure of vanilla four-layer MLP on MNIST

Observation 2: generalization performance

- Intra-class response variance of correctly classified samples is smaller than that of misclassified ones on arbitrary class
- Smaller intra-class response variance leads to better generalization



The horizontal axis and the vertical axis represent class indexes and the value of intra-class response variance, respectively. Each bar represents the intra-class response variance aggregated from all neurons in the penultimate layer.

Key Insight

- Intra-class response variance keeps growing with only cross entropy

- Neuron with small intra-class responses variance (**steadiness**) can lead to better generalization

Regularization on intra-class response variance is needed!

Neuron Steadiness Regularization (NSR)

- Final regularized loss function

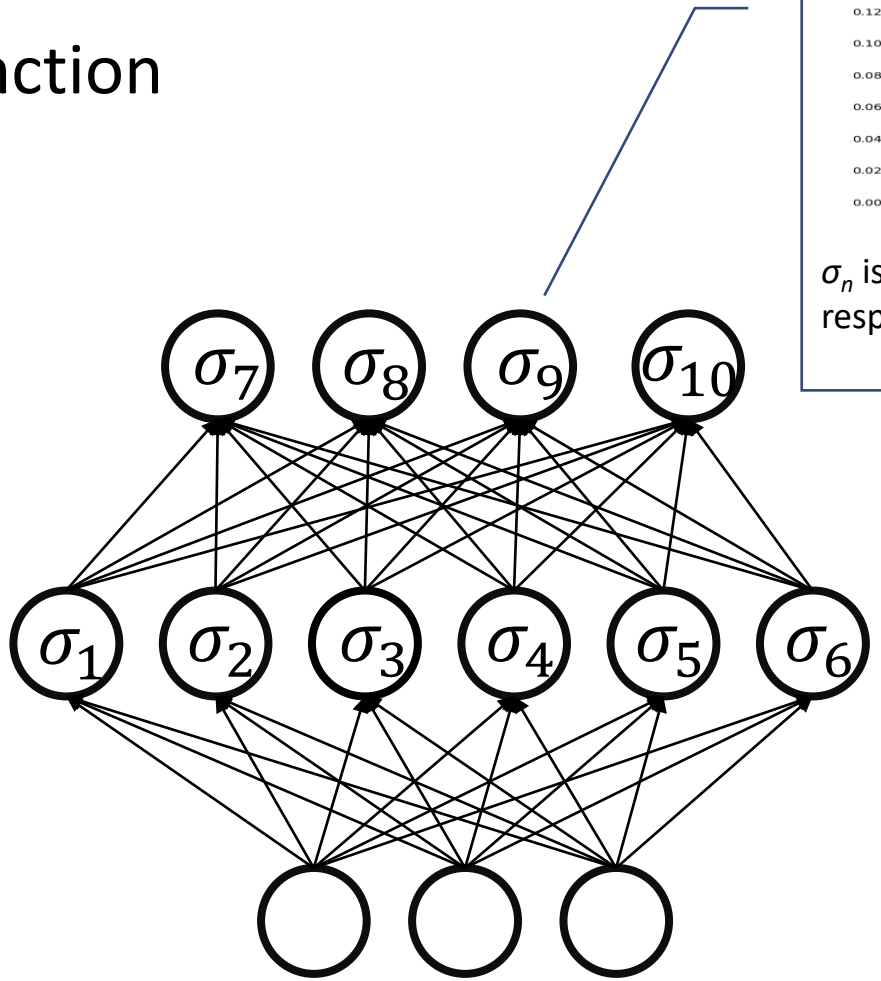
$$\mathcal{L} = \mathcal{L}_C + \mathcal{L}_S$$

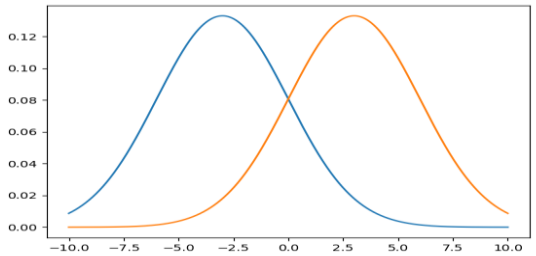
L_C is cross entropy loss

L_S is NSR loss of the model

- NSR loss of the model

$$\mathcal{L}_S = \sum_{n=1}^{N} \lambda_n \sigma_n$$





σ_n is calculated by the n_{th} neuron's response distribution of classes

Single Neuron Implementation

$$\sigma_n = \sum_{j=1}^{J} \alpha_j \cdot \mathrm{Var}\left(X_{n,j}\right)$$

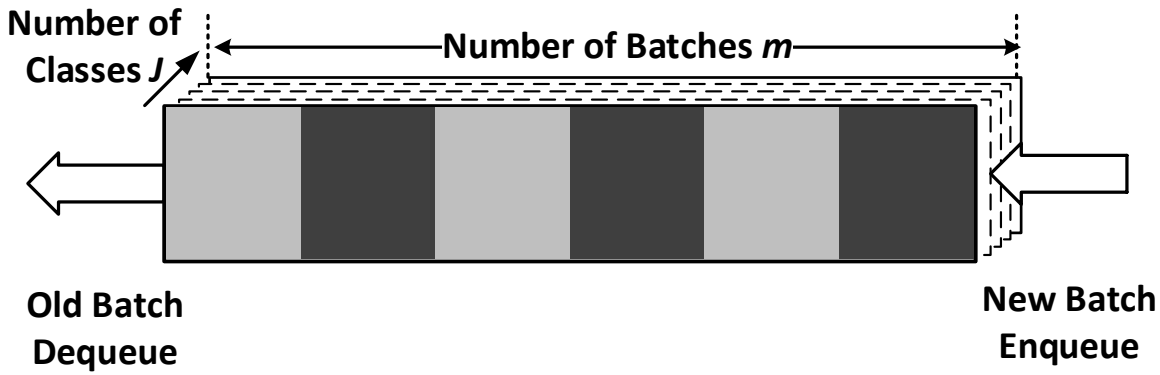
$$= \sum_{j=1}^{J} \alpha_j \cdot \mathbb{E}\left[\left(X_{n,j} - \mathbb{E}\left[X_{n,j}\right]\right)^2\right]$$

$$= \underbrace{\mathbb{E}\left[\sum_{j}^{J} \alpha_j X_{n,j}^2\right]}_{\text{Term 1}} - \underbrace{\sum_{j}^{J} \alpha_j \mathbb{E}^2\left[X_{n,j}\right]}_{\text{Term 2}}$$

Term 1 is calculated by current mini-batch

Term 2 is calculated by recent *m* mini-batch

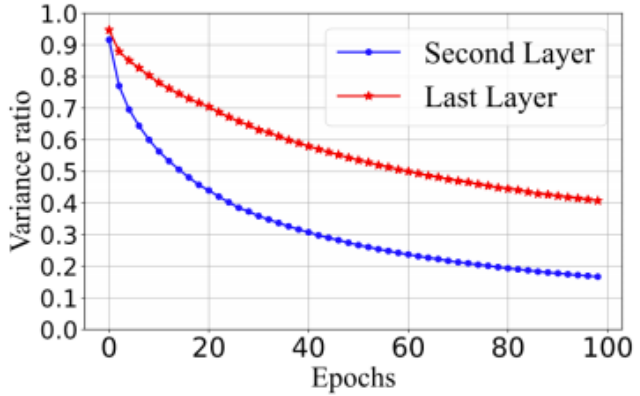
Neuron has a global response distribution view!



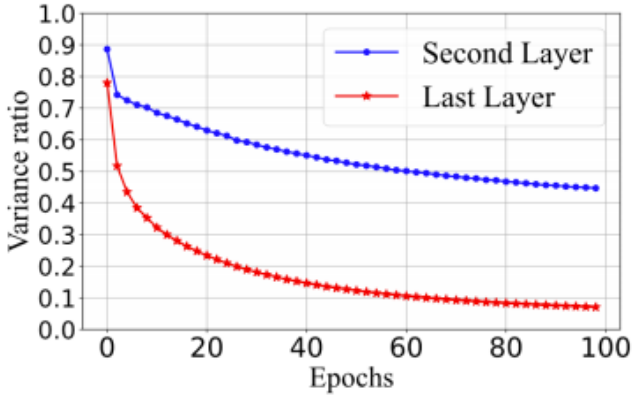
$X_{n,j}$ denotes the n_{th} neuron's response for samples of j_{th} class; α_j is the weight of j_{th} class

Simplification on applying NSR

- Only one specific layer apply
- Same value of λ for all neurons
- Layer with biggest variance for NSR



(a) NSR only applied on the second layer



(b) NSR only applied on the last layer

The variance ratio is the neuron intra-class response variance of the three-layer MLP applied with NSR, (a) only on the second layer and (b) only on the last layer, divided by the corresponding variance of the vanilla three-layer MLP

Microsoft

Evaluation

- How does NSR work on various datasets and different neural architectures?
- Does NSR outperform other classical regularization methods?
- What is the effect of combining NSR with other popular methods like Batch Normalization or Dropout?
- Which layer(s) should NSR be applied with?

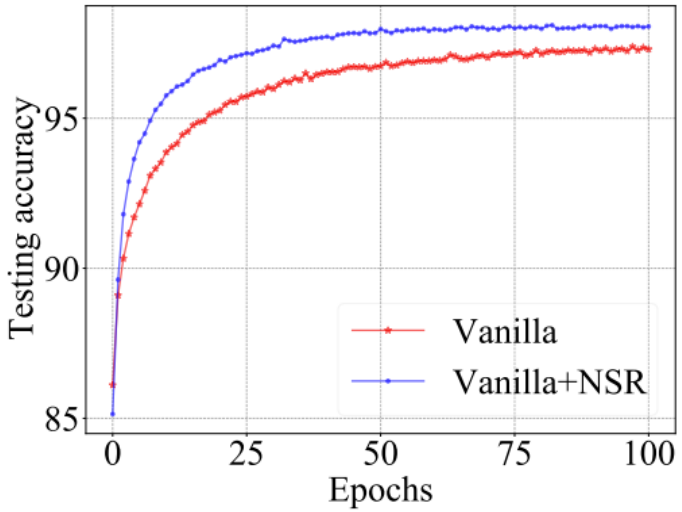
Evaluation Setting

Multiple datasets and architectures

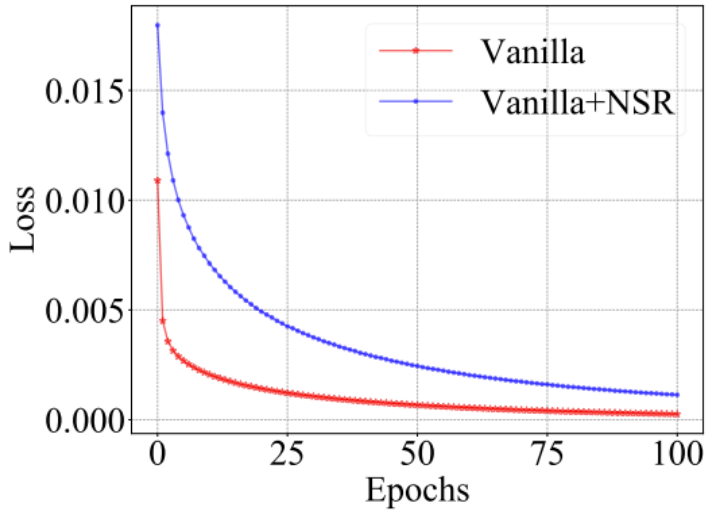
Network Architecture	Vanilla model	Dataset	Optimization
Multiple Layer Perceptron	MLP-3,4,6,8,10	MNIST	SGD
Convolutional Neural Network	ResNet-18	CIFAR-10	Momentum
	VGG-19		
	ResNet-50	ImageNet	Adam
Graph Neural Network	GraphSAGE	WikiCS, PubMed, Amazon-Photo, Computers	Adam
	GCN		

RQ1: NSR effect on training and testing

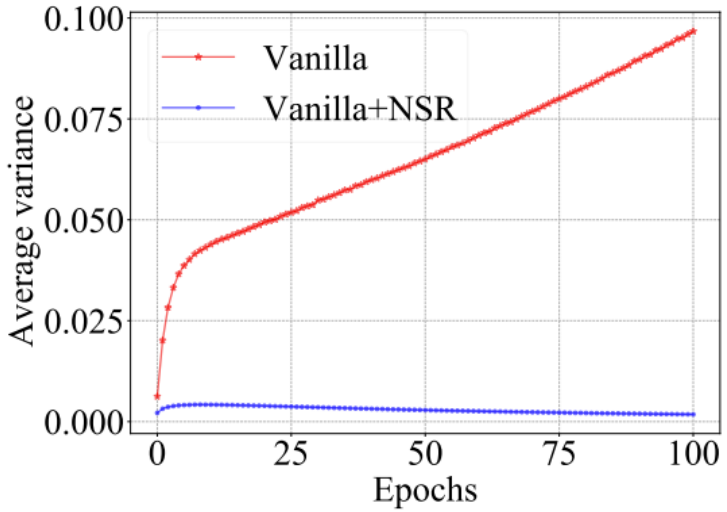
Intra-class variance is controlled with better generalization



(a) Testing accuracy



(b) Training cross-entropy loss



(c) Training intra-class variance

Training procedure of vanilla four-layer MLP and four-layer MLP with our Neuron Steadiness Regularization on MNIST

RQ1: Performance of NSR on MLP & CNN

Model	MLP-3	MLP-4	MLP-6	MLP-8	MLP-10	ResNet-18	VGG-19
Vanilla	3.09 ± 0.10	2.29 ± 0.07	2.44 ± 0.09	2.87 ± 0.09	3.06 ± 0.06	7.96 ± 0.12	10.57 ± 0.17
Vanilla+NSR	2.80 ± 0.08	1.64 ± 0.04	1.76 ± 0.06	1.98 ± 0.09	1.72 ± 0.14	7.20 ± 0.09	8.77 ± 0.10
Gain of NSR	9.39%	28.38%	27.87%	30.87%	43.79%	9.55%	17.03%

Model	ResNet-18	VGG-19	ResNet-50
Vanilla	4.22 ± 0.07	9.19 ± 0.18	7.82 ± 0.09
Vanilla+NSR	3.74 ± 0.08	8.09 ± 0.17	6.98 ± 0.08
Gain of NSR	11.37%	11.97%	10.74%

RQ1: Performance of NSR on GNN

Dataset	Model	GraphSAGE-2	GraphSAGE-3	GraphSAGE-4	GCN-2	GCN-3	GCN-4
PubMed	Vanilla	10.73 ± 0.06	10.20 ± 0.25	10.43 ± 0.17	12.02 ± 0.00	12.76 ± 0.18	14.01 ± 0.07
	Vanilla+NSR	9.89 ± 0.08	9.48 ± 0.12	9.79 ± 0.19	11.92 ± 0.00	12.19 ± 0.11	12.96 ± 0.08
	Gain	7.83%	7.06%	6.14%	0.83%	4.47%	7.49%
Amazon-Photo	Vanilla	5.82 ± 0.00	5.20 ± 0.14	6.37 ± 0.30	6.73 ± 0.00	8.00 ± 0.11	10.24 ± 0.14
	Vanilla+NSR	4.54 ± 0.10	4.86 ± 0.13	5.62 ± 0.59	6.27 ± 0.00	7.96 ± 0.10	9.03 ± 0.25
	Gain	21.99%	6.54%	11.77%	6.84%	0.50%	11.82%
Amazon-Computers	Vanilla	11.37 ± 0.55	11.88 ± 1.05	15.49 ± 0.90	12.17 ± 0.07	14.90 ± 0.25	18.07 ± 0.74
	Vanilla+NSR	10.47 ± 0.05	10.22 ± 0.54	12.86 ± 0.82	10.86 ± 0.03	13.66 ± 0.12	16.02 ± 0.23
	Gain	7.92%	13.97%	16.98%	10.76%	8.32%	11.34%
WikiCS	Vanilla	16.81 ± 0.21	15.97 ± 0.18	16.63 ± 0.31	18.41 ± 0.06	18.66 ± 0.23	19.21 ± 0.31
	Vanilla+NSR	16.06 ± 0.33	15.27 ± 0.21	15.43 ± 0.24	17.99 ± 0.05	18.10 ± 0.27	18.84 ± 0.26
	Gain	4.46%	4.38%	7.22%	2.28%	3.00%	1.93%
Average Gain of NSR		10.55%	7.99%	10.53%	5.18%	4.07%	8.15%

RQ2: Comparison with Other Regularization

Regularization	MLP-4	ResNet-18	GraphSAGE
Vanilla	2.29 ± 0.07	7.96 ± 0.12	11.37 ± 0.55
L1	2.27 ± 0.05	7.83 ± 0.23	10.81 ± 0.13
L2	2.27 ± 0.05	7.67 ± 0.18	10.68 ± 0.35
Jacobian	2.21 ± 0.04	7.90 ± 0.07	11.27 ± 0.45
NSR	1.64 ± 0.04	7.20 ± 0.09	10.52 ± 0.22

Hyper-parameter λ in every method is tunned with the same NNI setting

RQ3: Combination with Other Techniques

MLP-4	Vanilla	Vanilla + BN	Vanilla + BN + NSR
Error rate	2.29 ± 0.07	2.22 ± 0.04	1.62 ± 0.08

MLP-4	Vanilla	Vanilla + Dropout	Vanilla + Dropout + NSR
Error rate	2.29 ± 0.07	2.19 ± 0.04	1.64 ± 0.04

RQ4: Effect of NSR on Different Layers

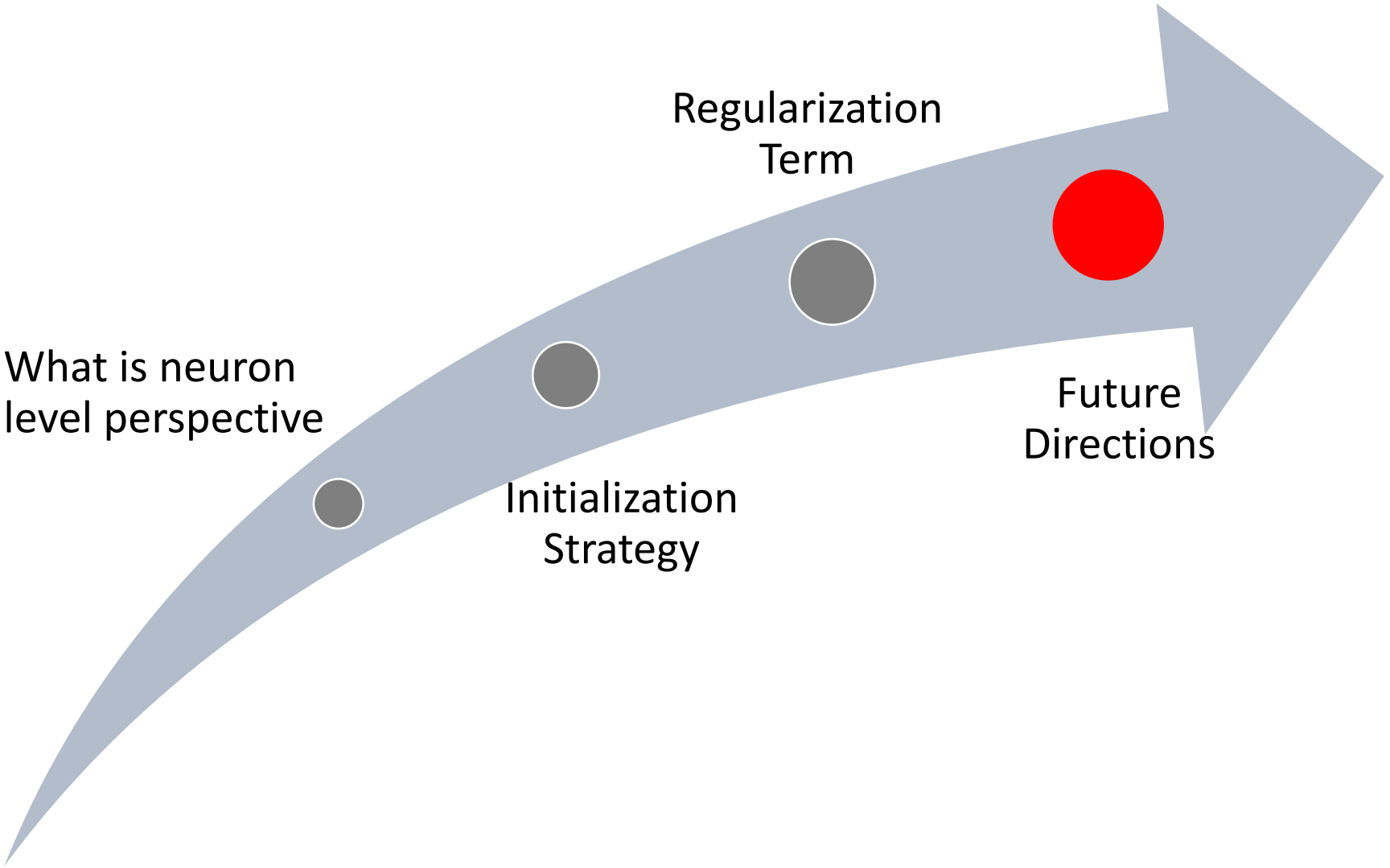
- The accuracy is improved no matter which layer is applied with NSR
- Applying NSR to the layer with the biggest variance could
  - obtain the most significant gain compared with other layers
  - achieve similar accuracy compared with the best one

Method	Error rate
MLP	2.29 ± 0.07
MLP_2	2.22 ± 0.08
MLP_3	1.90 ± 0.13
MLP_4	1.64 ± 0.04
$MLP_{3,4}$	1.63 ± 0.08

Intra-class variance of layer 2,3,4 is 409, 510, and 1660

Method	Error rate
GraphSAGE	11.37 ± 0.55
$GraphSAGE_1$	10.47 ± 0.05
$GraphSAGE_2$	10.52 ± 0.22
$GraphSAGE_{1,2}$	10.30 ± 0.17

Intra-class variance of layer 1,2 is 4.15 and 2.68


Regularization
Term
What is neuron
level perspective
Initialization
Strategy
Future
Directions

Future Work for Initialization

- Apply the method to finetune stage.
- Understand pre-training with neuron-level perspective.
- Introduce to broader neural network architectures.
- Take neuron correlation into consideration.

Future Work for Regularization term

- Explore hyper-parameter setting strategies
  - Adaptive setting λ along with training procedure
  - Setting λ according to neuron significance
- Explore other statistics based on neuron response distribution
  - Inter-class response distance
  - Adjacent class response distance
- Adapt NSR to broader architectures and tasks
  - Transformer, RNN
  - NLP tasks, time series related tasks
- Analyze on the response relationship between Neuron

What's next?

- How to understand self-attention architecture with Neuron-level perspective?
- How can the single neuron perspective help to understand learning tricks like component like Dropout, Batch Normalization?
- Can we develop new DNN architectures with neuron-level perspective?

Much to explore for the future!

Microsoft

Thanks & QA

Welcome to attend WSDM CUP2023!

Welcome to join our MSRA team!

Look forward to see you in KDD CUP2023!

憲躍

黑龙江 大庆

扫一扫上面的二维码图案，加我为朋友。

Microsoft

Below are some details prepared for QA

Microsoft

What is the neuron-level perspective?

Microsoft

Neuron-level Perspective for Initialization Strategy

Microsoft

Neuron-level perspective for Regularization term

Microsoft

What’s next?

Conclusion

- Understand neuron initialization from neuron-level perspective
- Provide Neuron Campaign Algorithm
- Utilize Information Bottleneck theory for better generalization

Conclusion

- Explore to find the neuron response phenomenon on training
- Derive a Neuron Steadiness Regularization
- Extensive experiments indicate the success across different **domains and architectures.**

Reason for it

- Gradient Exposure and vanish
  - Forward

$$y = W_3 * W_2 * W_1 * x$$

  - Backward

$$\nabla W_1 = \boxed{\frac{\partial Loss}{\partial f_3}} * \boxed{\frac{\partial f_3}{\partial z_3} * \frac{\partial f_2}{\partial z_2} * \frac{\partial f_1}{\partial z_1}} * \boxed{W_3 * W_2} * X$$

Glorot condition

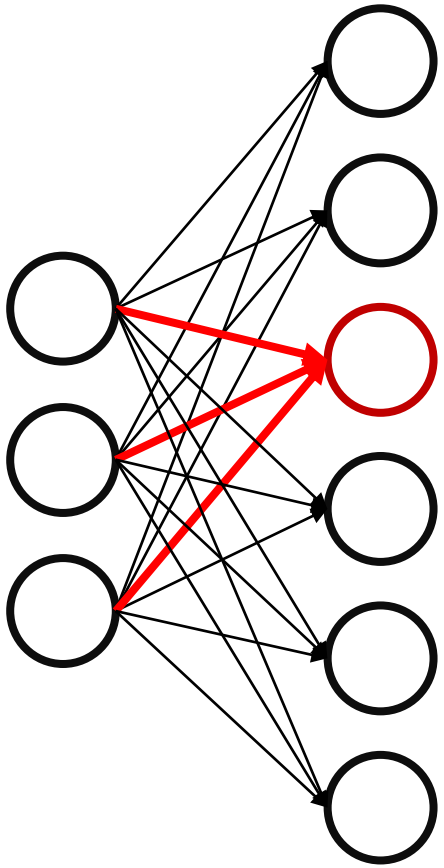
- The variance of the outputs of different hidden layer should be similar
- The variance of gradient from different layer should be similar

Xavier Initialization

$$y_l = W_l x_l + b_l$$

- *Assumptions*
  - *Linear activation function*
  - *The expectation of input and weight are all 0*
  - *W_l are mutually independent and share the same distribution*
  - *x_l are mutually independent and share the same distribution*
  - *x_l and W_l are independent of each other*
- Final form

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$



2022/11/2

Variance scaling Initialization strategy

- He initialization (for Relu activation function)

$$W \sim N\left(0, \sqrt{\frac{2}{n_j}}\right)$$

- LSUV (Layer Sequential Unit-Variance initialization)

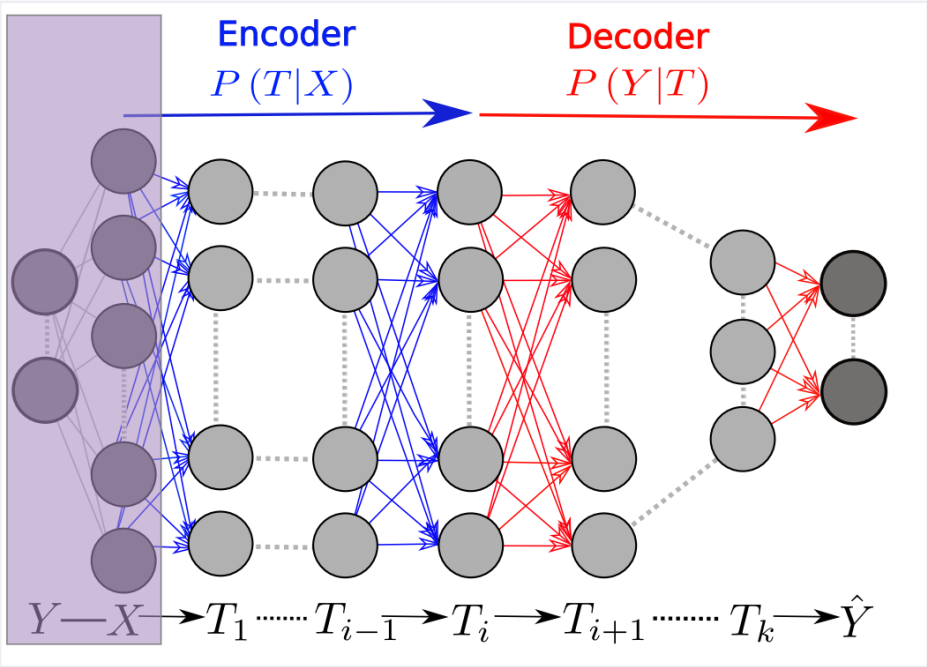
$$\mathrm{W}_L = \mathrm{W}_L / \sqrt{var(Z_i)}$$

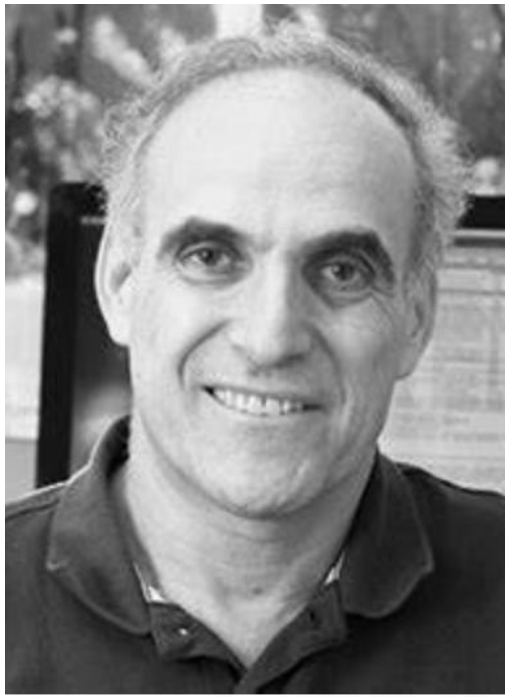
Information Bottleneck

- Tishby et al. believes that DNN training is actually optimizing the following objective:

$$\min_{\theta} I(X;T) - \beta I(T;Y)$$

where T is the feature of each layer, X is the input and Y is the label

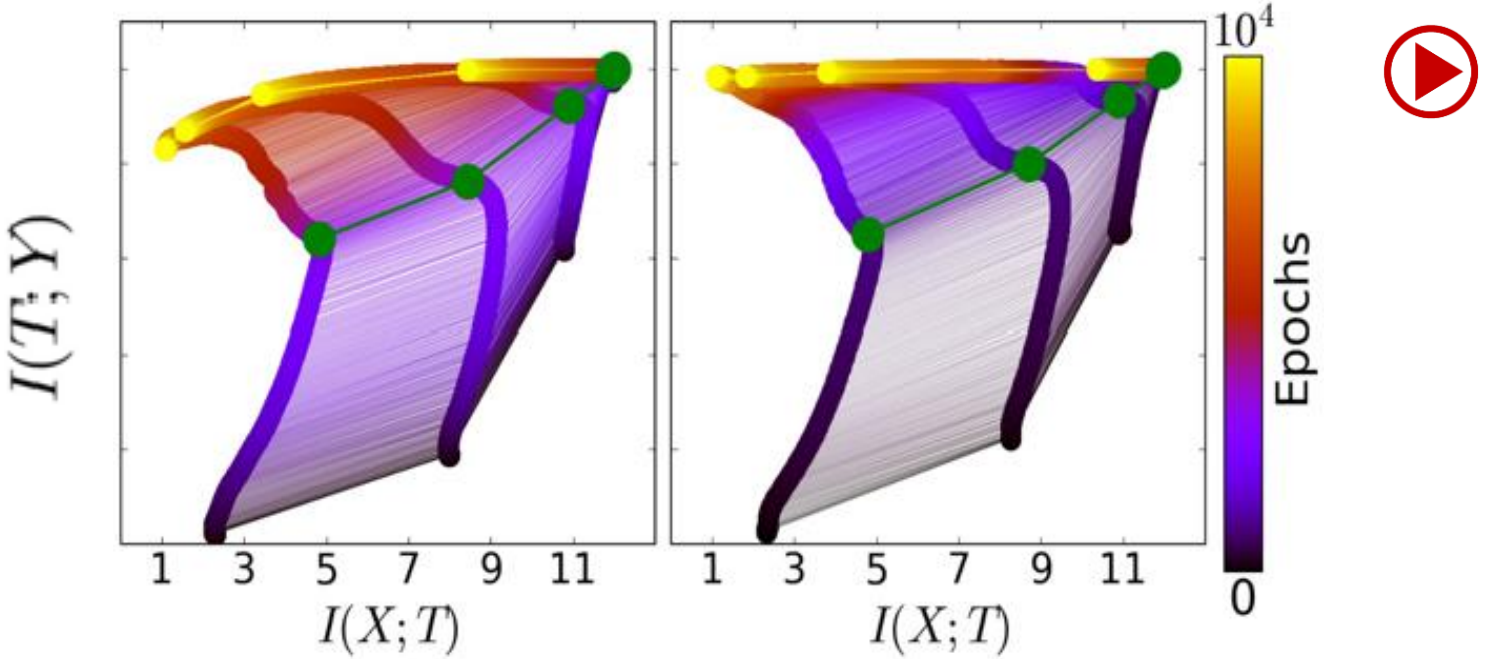




[1] Opening the Black Box of Deep Neural Networks via Information. Arxiv: 1703.00810

[2] Talk by Prof. Tishby: https://www.youtube.com/watch?v=bLqJHjXihK8&t=262s

Information Bottleneck



- Phase 1: $I(X;\ T)$ and $I(T;\ Y)$ both increases, which indicates the network memorizes the information about the input
- Phase 2: $I(X;\ T)$ decrease while $I(T;\ Y)$ increases, which indicates that the network drops unimportant information to generalize.

Criteria Simplification with High Efficiency

- $I(X; Z_i)$ input information maintenance criterion

  $I(X; Z_i)$= H(Z) – H(Z|X) = H(Z)

  $$tr(\Sigma_i)$$

  where Σ_i is the covariance matrix of Z_i

- $I(Z_i; Y)$ target-related maintenance criterion

$$tr(\hat{H}\hat{H}^T) - \frac{1}{N}\sum_{j=0}^{N} tr(\hat{Z}_i^T \Pi^j \hat{Z}_i)$$

Inter-class variance

Intra-class variance

Algorithm details

X:[60000, 784] —W→ Z:[60000, 1000] ⟶ s:[1000]

X:[60000, 100] —W→ Z:[60000, 200] ⟶ s:[200]

Algorithm 1 Neuron Campaign initialization algorithm

Input: Candidate weight matrix $\mathbf{W}$ **[784, 1000]**

1: **for** t=1 to T **do** **T = 100**

2: Update generalized orthnormalization matrix at t steps: $\mathbf{A}_t = (\mathbf{A}_{t-1}, \mathbf{a}_t^T)^T$ **[784, t]**

3: Calculate the null space projection by $\mathbf{P}_t = \mathbf{P}_{t-1} - \mathbf{a}_t \mathbf{a}_t^T \mathbf{W}$

4: Select optimal neuron whose index is chosen by $i = \max_i s_i \frac{||\mathbf{p}_t^i||}{||\mathbf{W}_i||}$

5: Update $\mathbf{w}^* = \mathbf{W}_{\cdot,i}$

6: Normalize basis of the generalized orthnormalization matrix as $\mathbf{a_{t+1}} = \mathbf{p}_t^i / ||\mathbf{p}_t^i||$

7: **end for**

Output: Winning neurons formed weight matrix $\mathbf{W}'$ **[784, 100]**

Ensure orthogonality of the selected neurons

Select the neuron with largest score

Existing Regularizations

Inductive bias	Example	Principle	Leveraged information
Scale	L2	Penalize large norms of model weights	Weights
Sparseness	L1	Reward zero neuron response	Collective neuron responses
Smoothness	Jacobian	Penalize big change with small perturbation	Mapping function derivatives
Diversity	Orthogonalization	Reduce feature correlations	Weight correlations

Limitations:

- Individual neurons are lack of “global view” of its response distribution on different classes
- Neurons are only aware of responses of current mini-batch, which may contain noise and be instable

Theorem Guarantee

Lemma 3.1. *For a multi-class classification problem, when (1) a deep learning model utilizing the Cross Entropy loss with an NSR regularization term on any of its intermediate layer is optimized via gradient descent and (2) the capacity of the model is sufficiently large, the Consistency of Representations Complexity Measure on this model $\mathcal{C}$ will tend to be 0. Under the same condition, when the deep learning model is only optimized by the Cross Entropy loss without an NSR regularization term, there will be infinite local minima where the complexity measure $\mathcal{C}$ will be a positive number.*

Complexity Measure: Consistency of representation:

$$\mathcal{C} = \frac{1}{J}\sum_{i=1}^{J}\max_{i\neq j}\frac{S_i + S_j}{M_{i,j}}$$