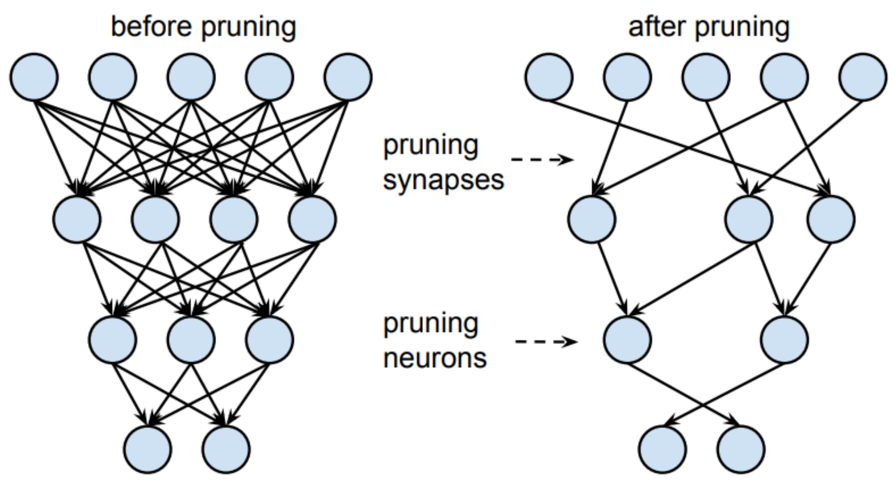
딥러닝 모델 엑기스 추출 Knowledge Distillation

Machine Learning & Visual Computing Lab
김유민

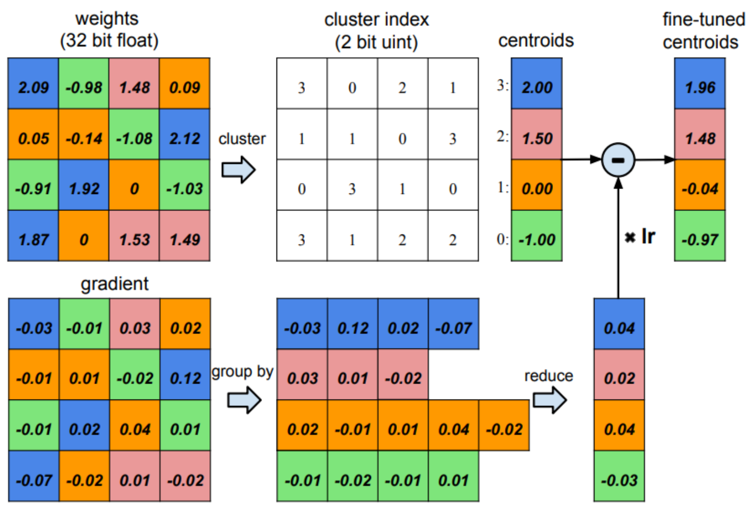
CONTEXT

COMPRESSION METHODS

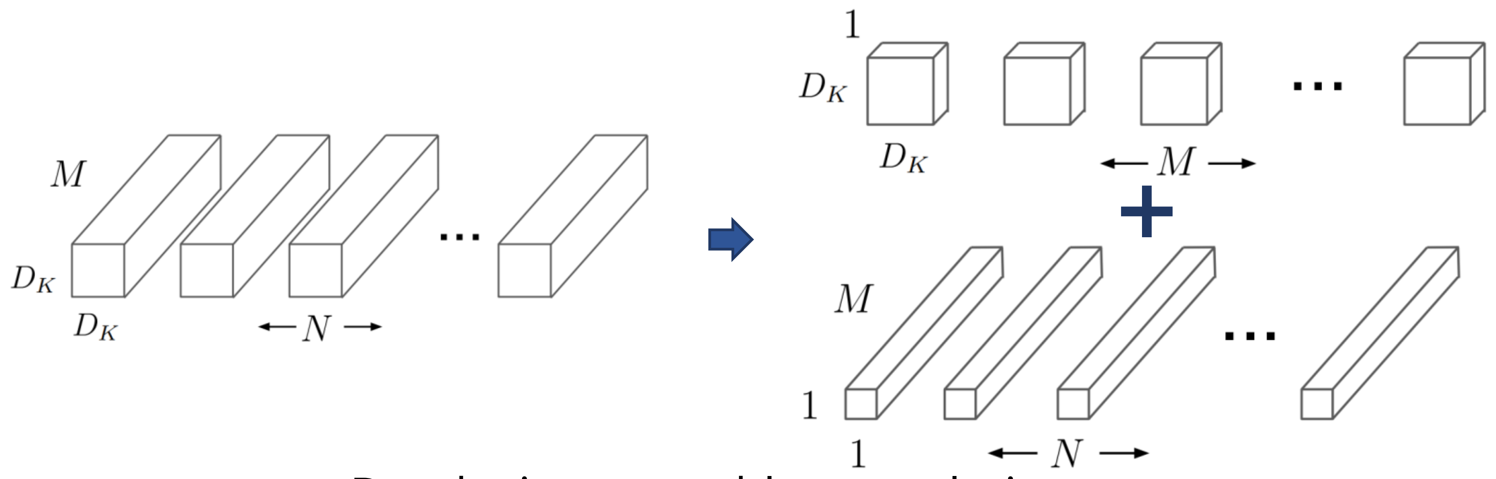
- Pruning
- Quantization
- Efficient Design
- Distillation
- Etc…

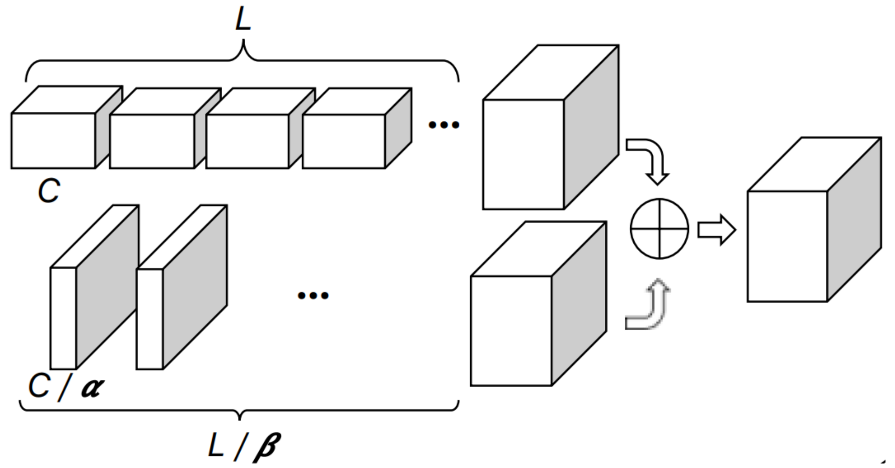


<Weight Pruning[1]>



<Weight Quantization[2]>





<Depthwise separable convolution[3]>

<Big-little net architecture[4]>

[1] Song Han, Jeff Pool, John Tran and Willan J. Dally. Learning both Weights and Connections for Efficient Neural Networks. In *NIPS*, 2015.

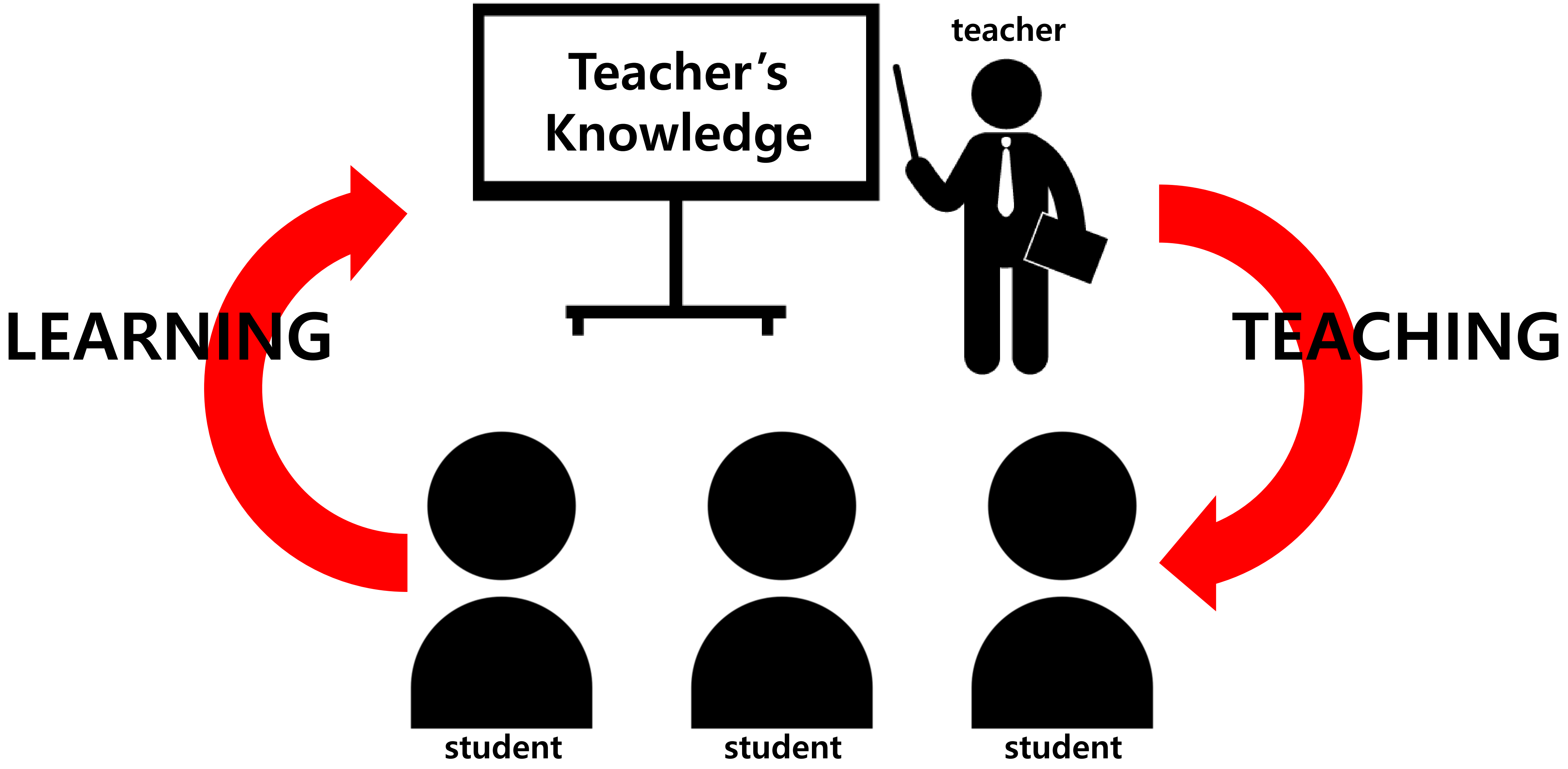
[2] Song Han, Huizi Mao and William J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *ICLR*, 2016.

[3] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. In *arXiv*, 2017.

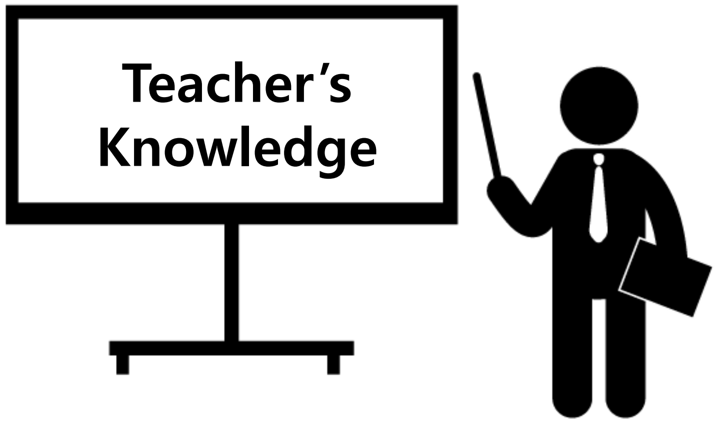
[4] Chun-Fu(Richard) Chen et al. Big-Little Net: an Efficient Multi-Scale Feature Representation for visual and speech recognition. In *ICLR*, 2019.

DISTILLATION

Teacher–Student Relation



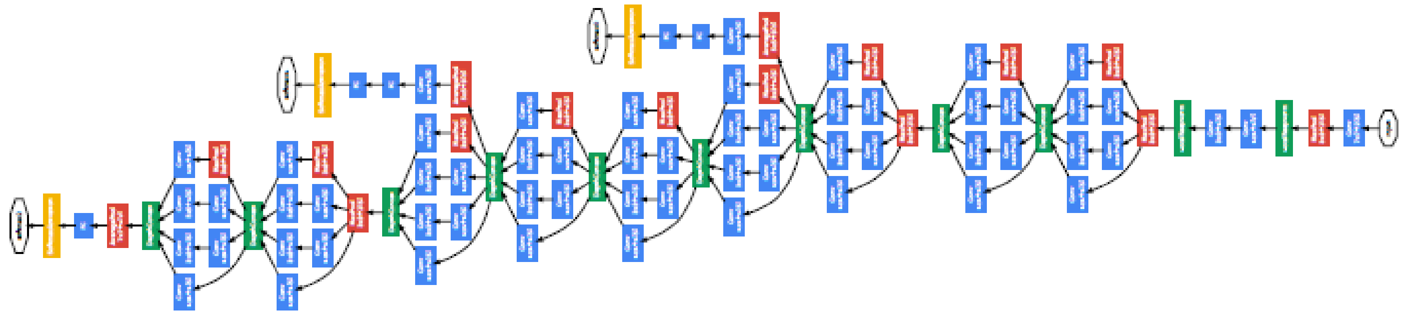
Teacher-Student Relation in Deep Neural Network



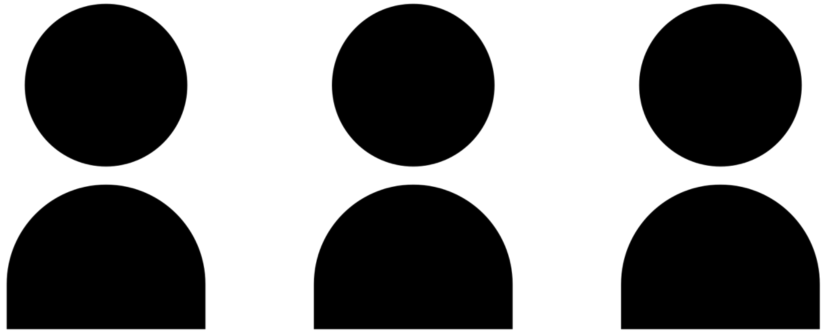
teacher

=

Large Neural Network
Classification Accuracy : 93.33%



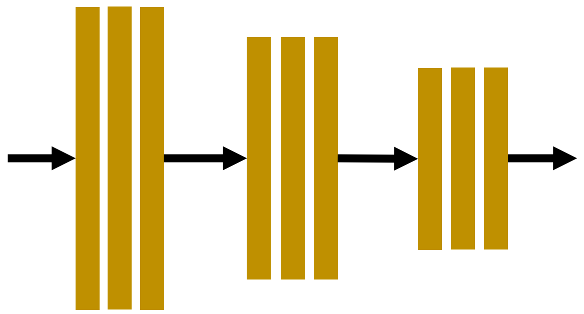
<Teacher Network>



student

=

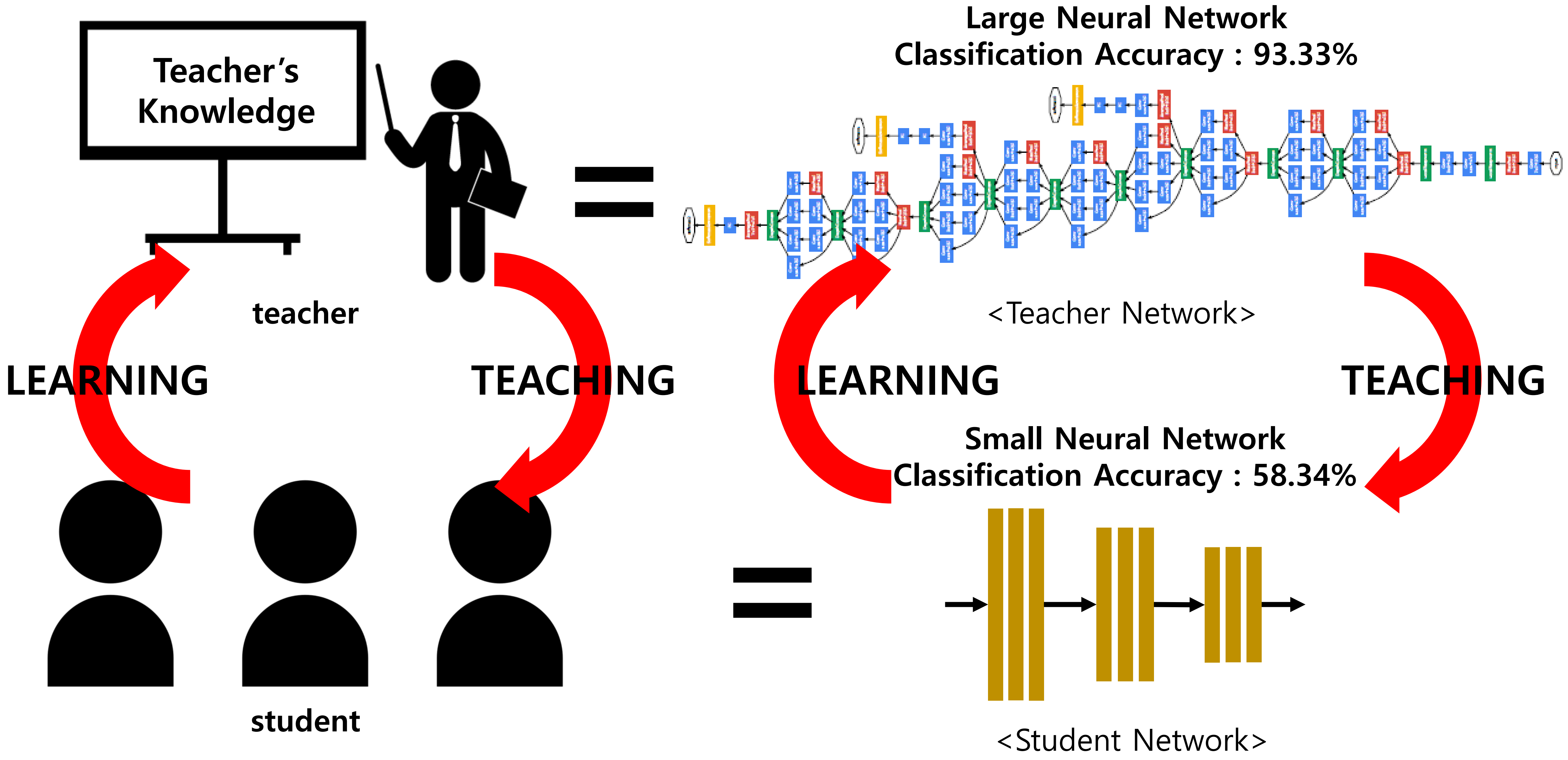
Small Neural Network
Classification Accuracy : 58.34%



<Student Network>

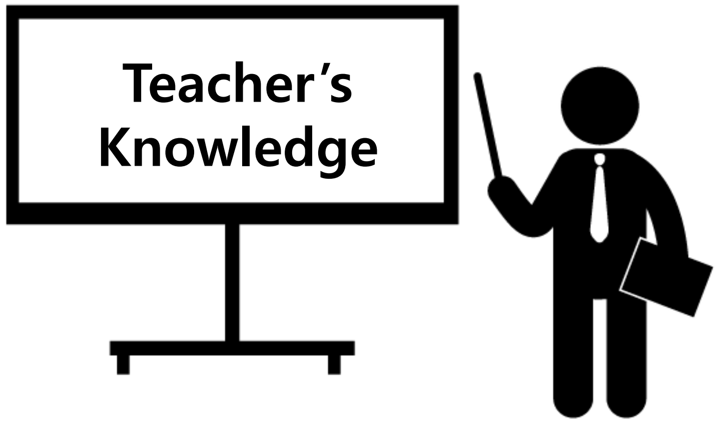
Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scitt Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015.

Teacher–Student Relation in Deep Neural Network



Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scitt Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015.

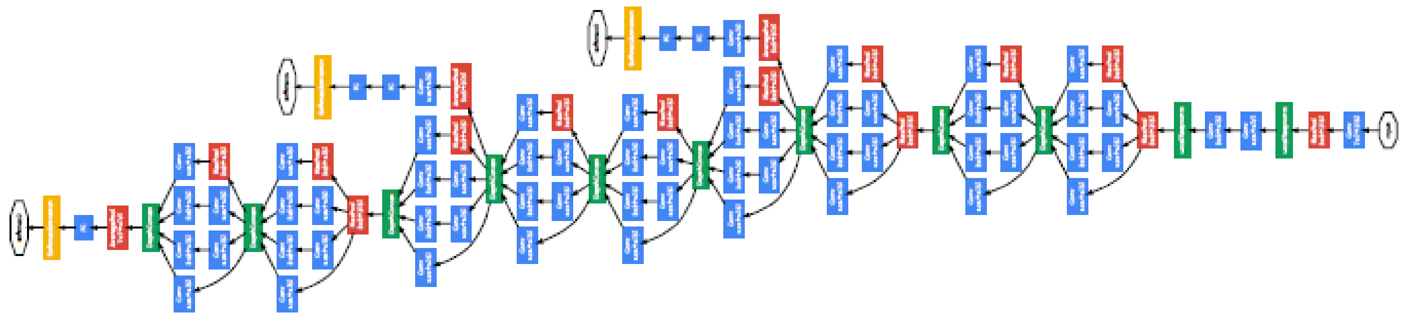
Teacher–Student Relation in Deep Neural Network



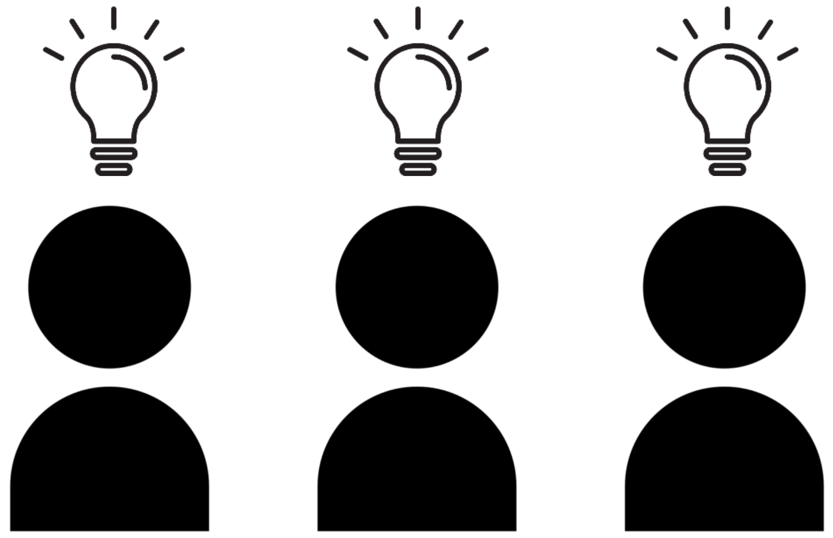
teacher

=

Large Neural Network
Classification Accuracy : 93.33%



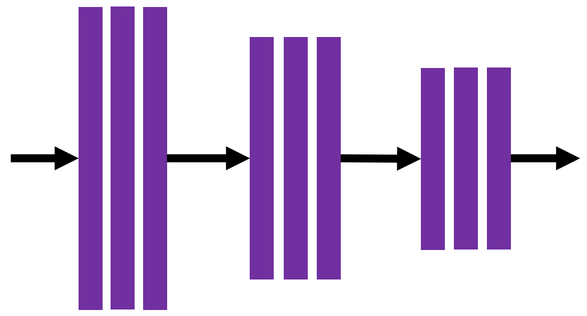
<Teacher Network>



student

=

Small Neural Network
Classification Accuracy : 58.34% -> 93.33%



<**Learned** Student Network>

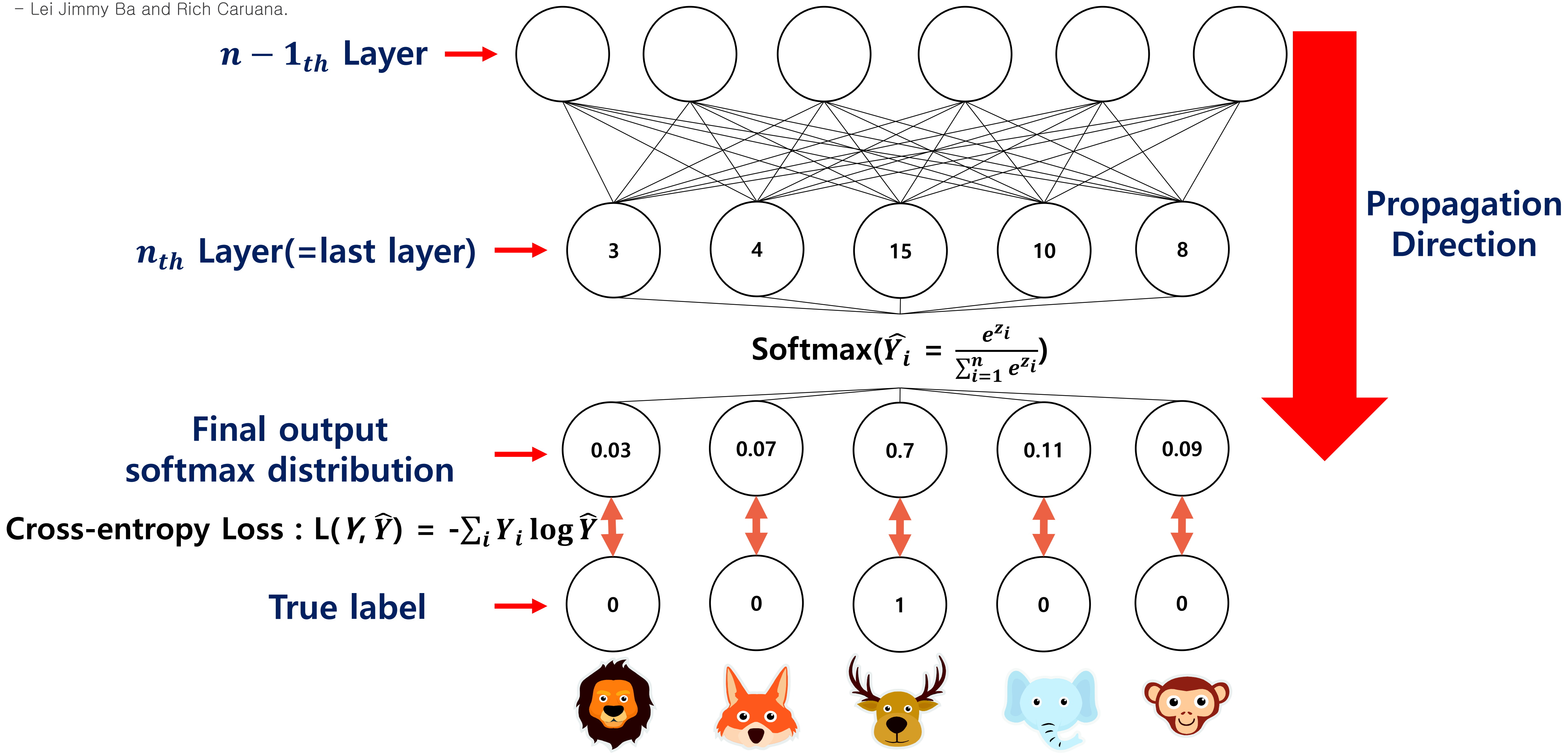
Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scitt Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015.

Do Deep Nets Really Need to be Deep?

2014
NIPS

Do Deep Nets Really Need to be Deep?. In *NIPS*, 2014.

– Lei Jimmy Ba and Rich Caruana.

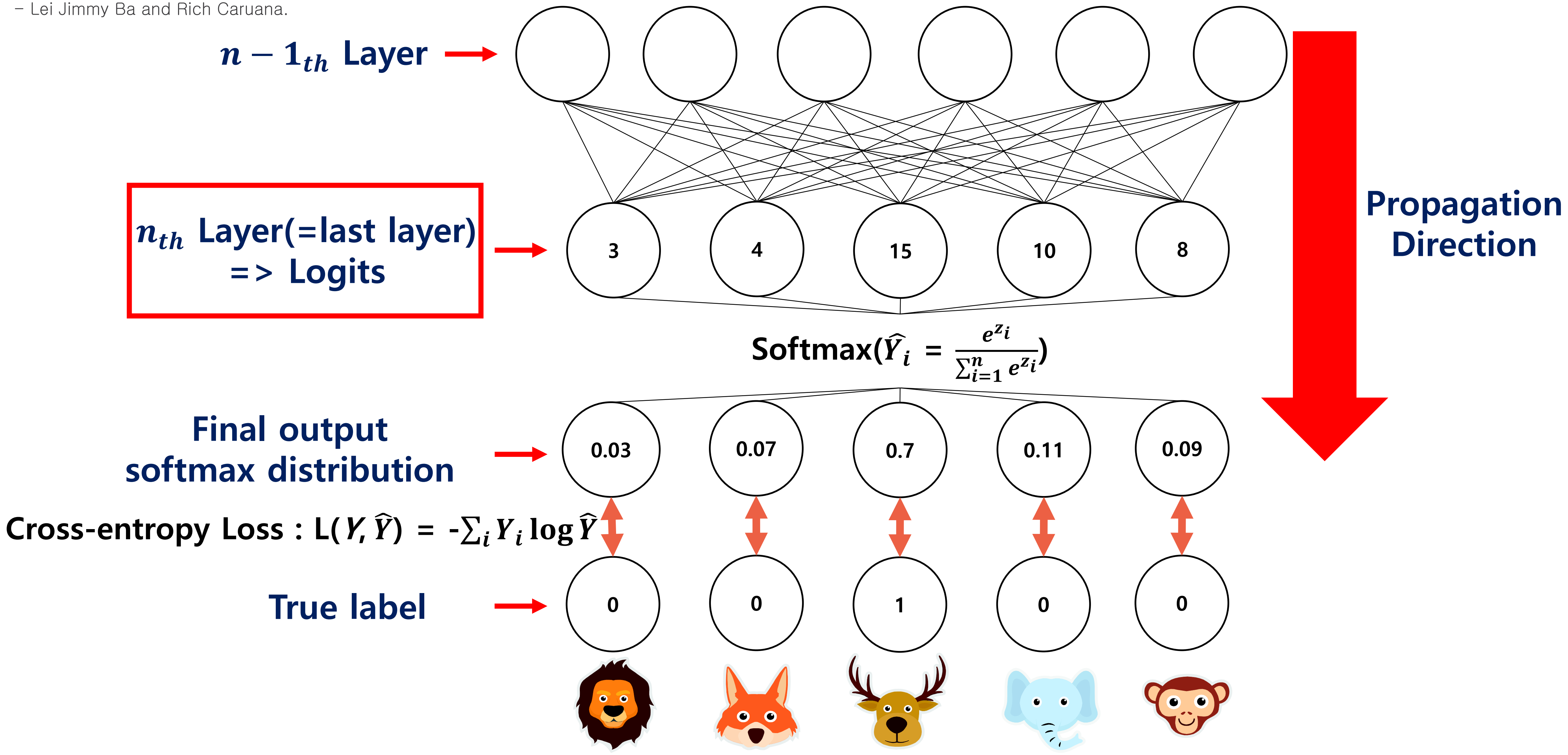

$n-1_{th}$ Layer
n_{th} Layer(=last layer)
3
4
15
10
8
Softmax($\hat{Y}_i = \frac{e^{z_i}}{\sum_{i=1}^{n} e^{z_i}}$)
Final output softmax distribution
0.03
0.07
0.7
0.11
0.09
Cross-entropy Loss : L($Y, \hat{Y}$) = $-\sum_i Y_i \log \hat{Y}$
True label
0
0
1
0
0
Propagation Direction

<Last section of DNN>

Animal pictures – www.freepik.com

Do Deep Nets Really Need to be Deep?. In *NIPS*, 2014.

– Lei Jimmy Ba and Rich Caruana.



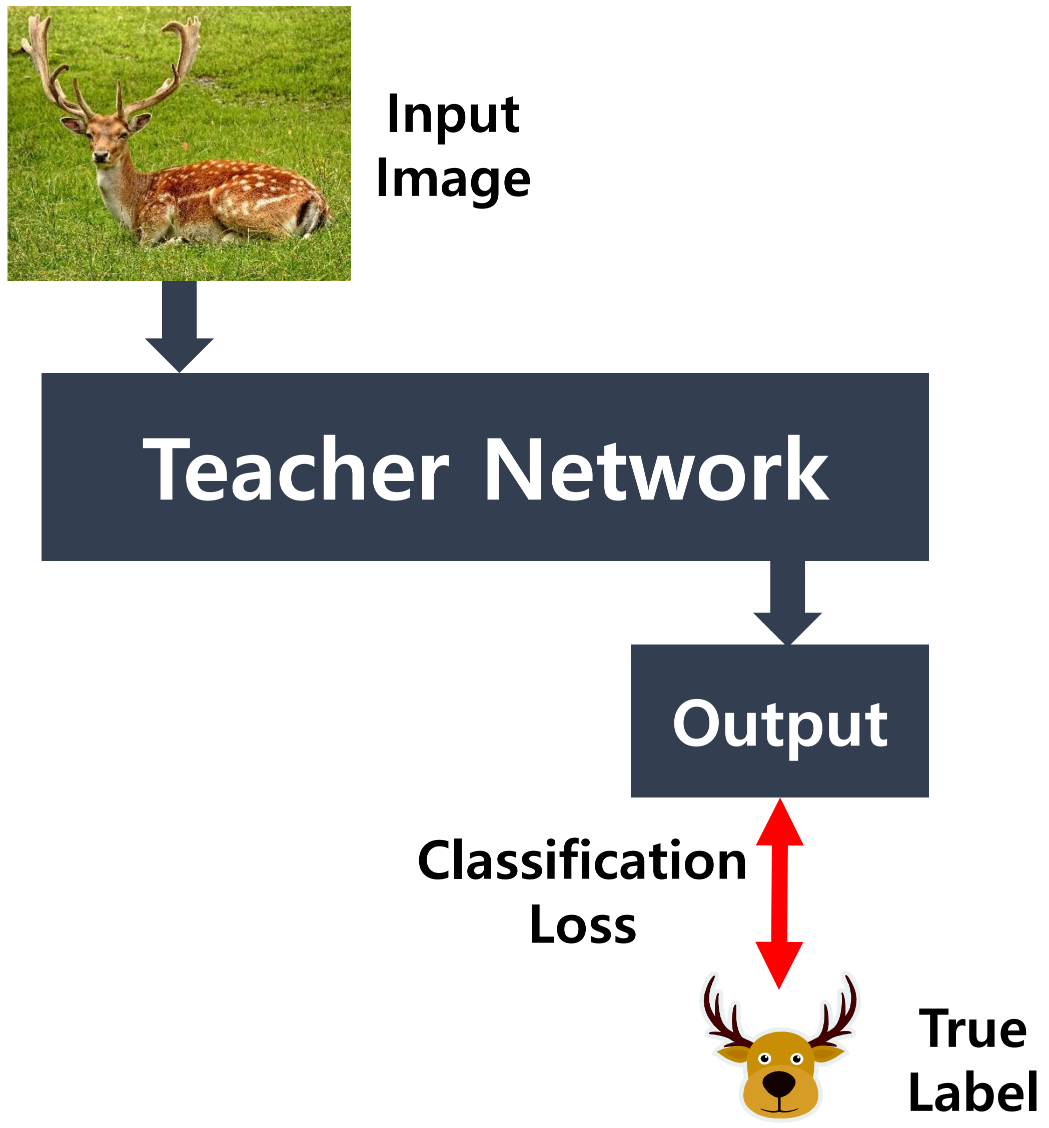
$n-1_{th}$ Layer
n_{th} Layer(=last layer) => Logits
3
4
15
10
8
Propagation Direction
Softmax($\hat{Y}_i$ = $\frac{e^{z_i}}{\sum_{i=1}^{n} e^{z_i}}$)
Final output softmax distribution
0.03
0.07
0.7
0.11
0.09
Cross-entropy Loss : L(Y, $\hat{Y}$) = -$\sum_i Y_i \log \hat{Y}$
True label
0
0
1
0
0

<Last section of DNN>

Animal pictures – www.freepik.com

Do Deep Nets Really Need to be Deep?. In *NIPS*, 2014.

– Lei Jimmy Ba and Rich Caruana.

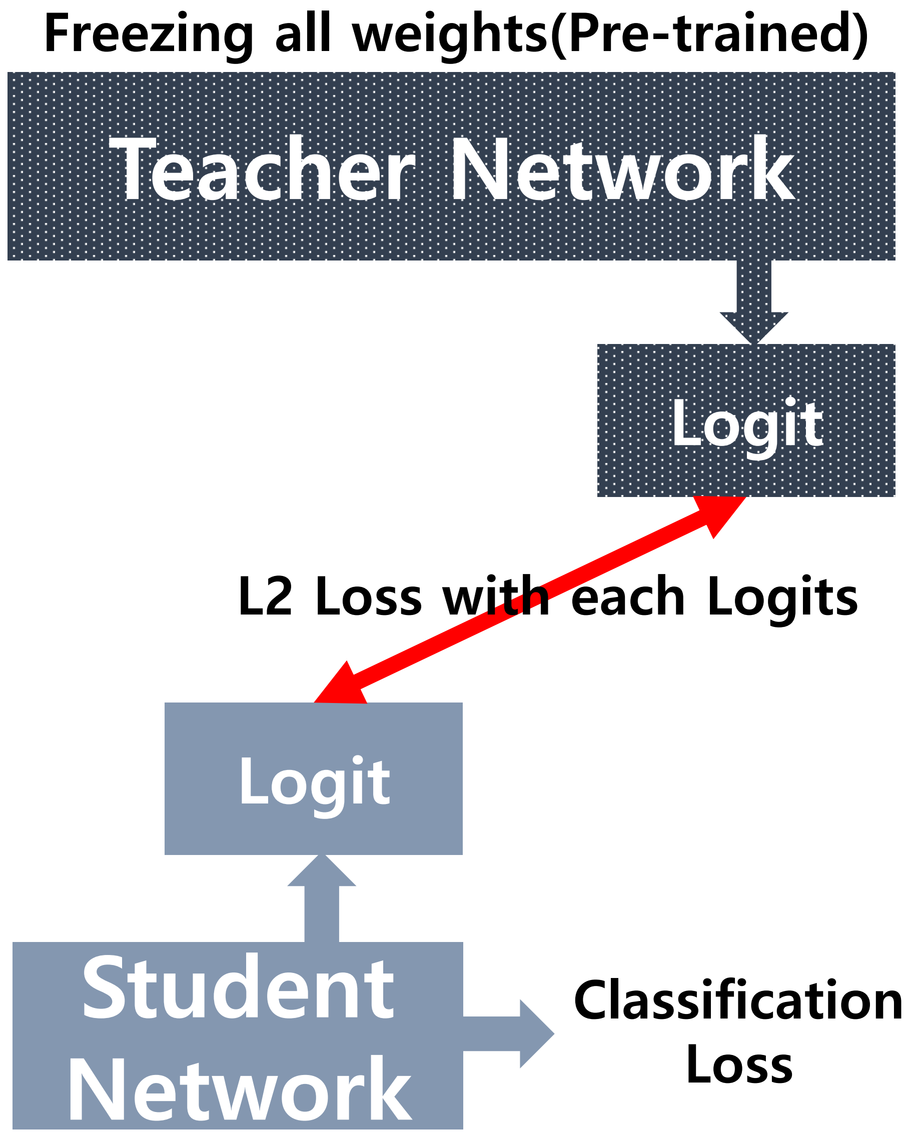


<1st step. Training Teacher Network>

Animal pictures – www.freepik.com

Do Deep Nets Really Need to be Deep?. In *NIPS*, 2014.

– Lei Jimmy Ba and Rich Caruana.



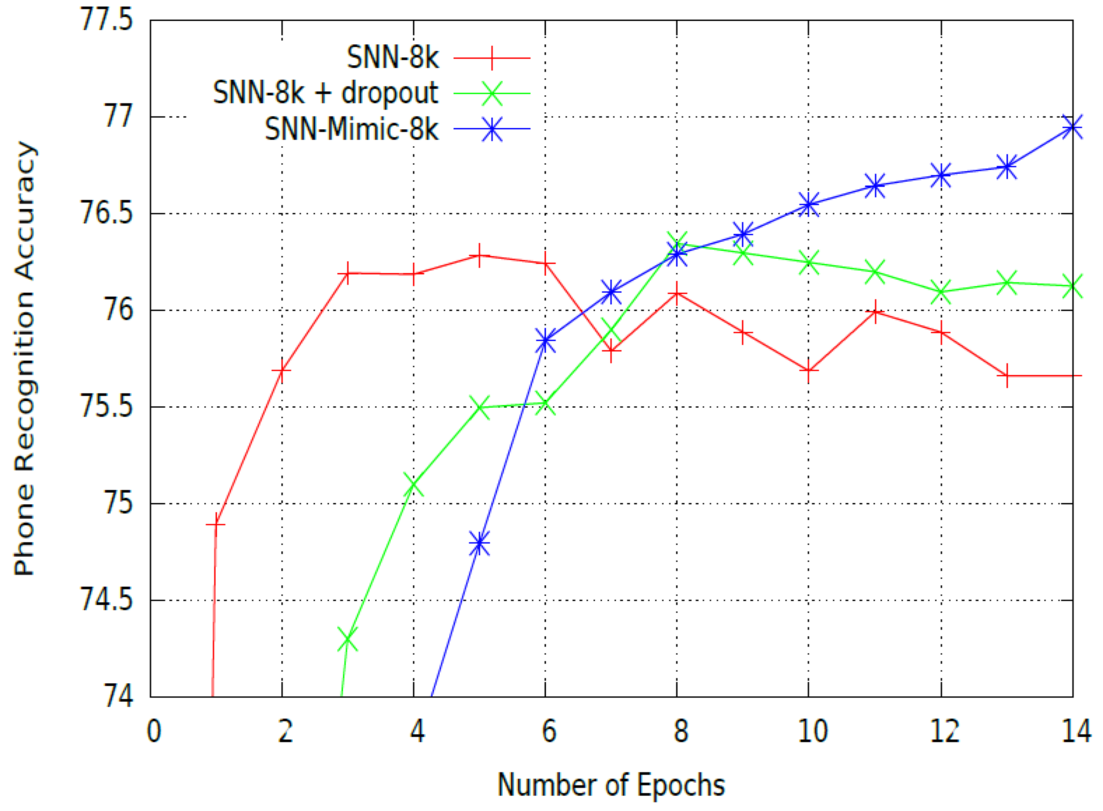
<2nd step. Training Student Network>

<Logits>	<softmax probability>
Cat: 32	Cat: 84%
Tiger: 24	Tiger: 7%
Dog: 15	Dog: 3%
Lion: 12	Lion: 2%
⋮	⋮

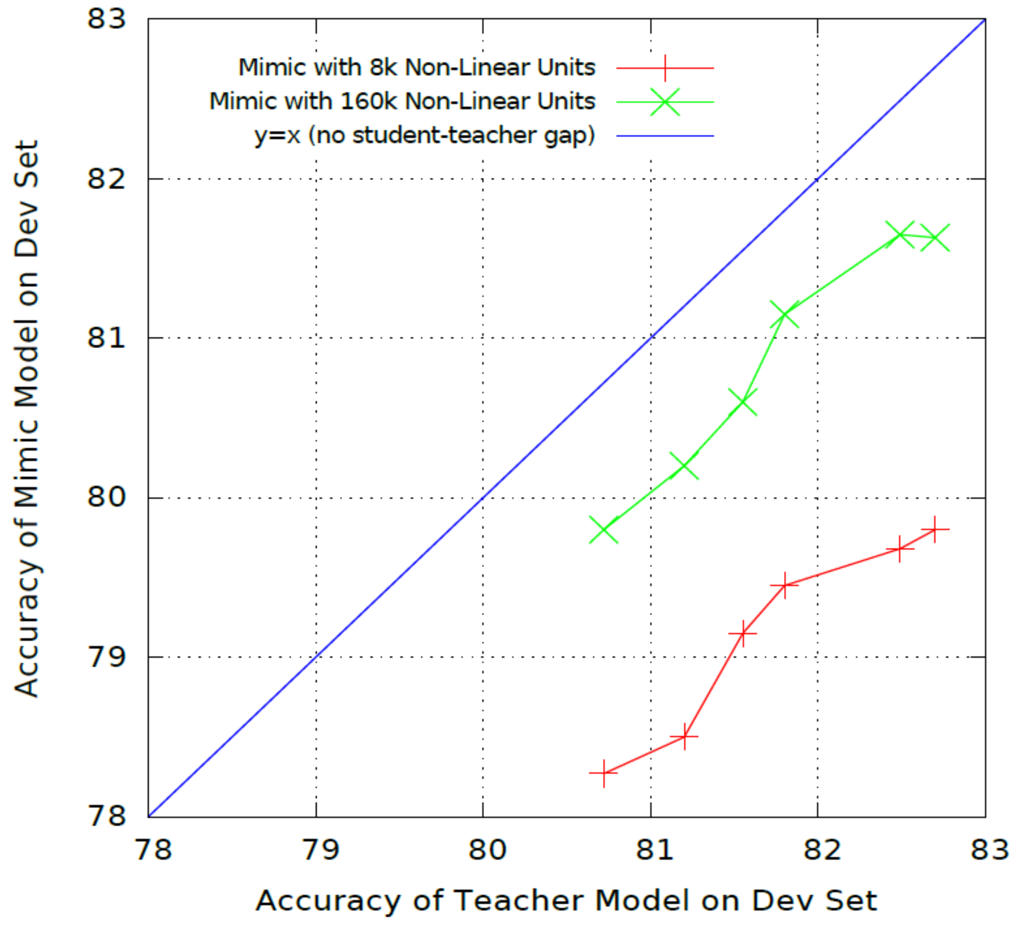
Animal pictures – www.freepik.com

Do Deep Nets Really Need to be Deep?. In *NIPS*, 2014.

– Lei Jimmy Ba and Rich Caruana.



<Student deosn’t overfit>



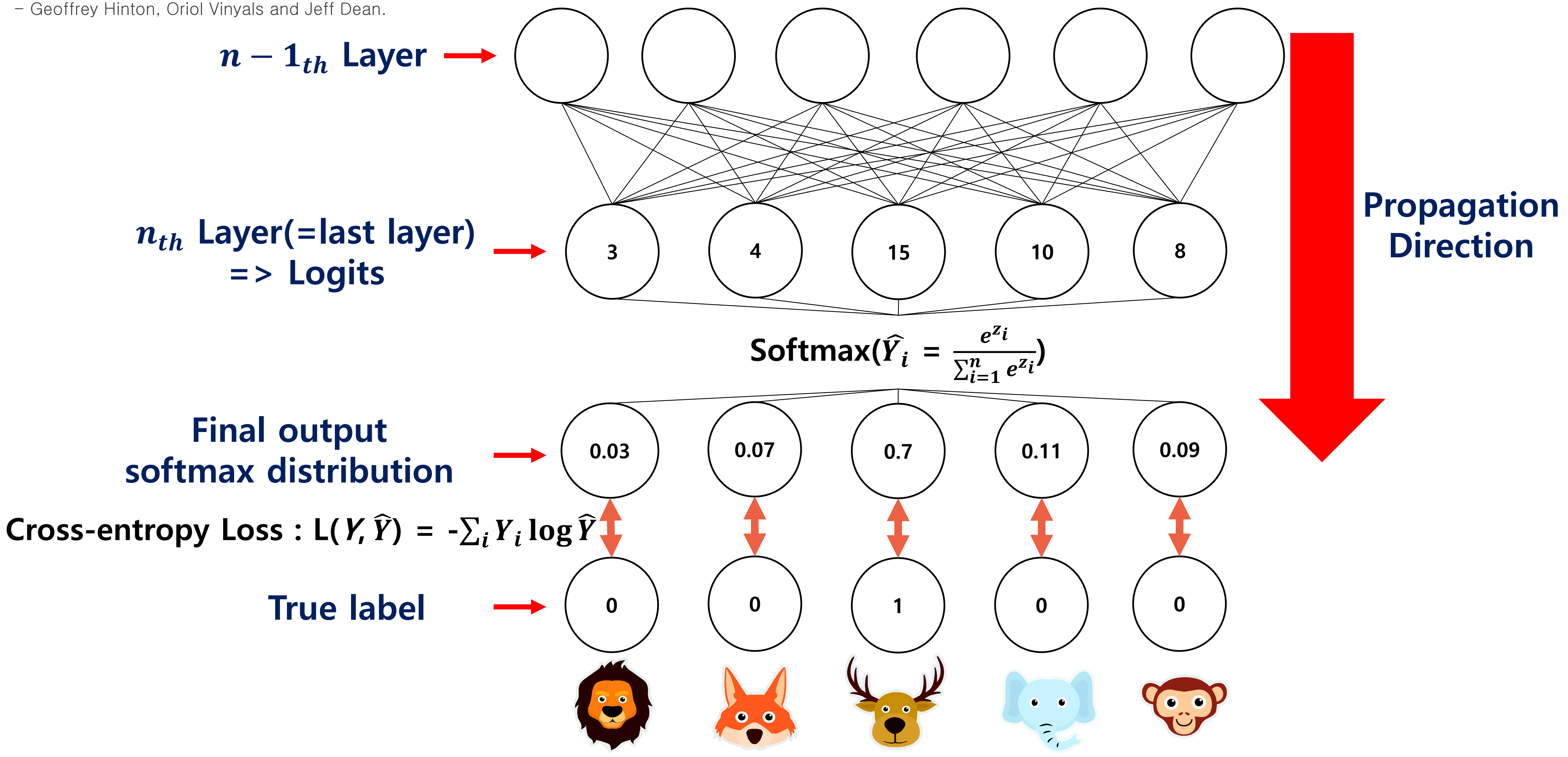
<Better teacher, Better student>

Distilling the Knowledge in a Neural Network

2014
NIPS workshop

Distilling the knowledge in a Neural Network. In *NIPS workshop*, 2014.

– Geoffrey Hinton, Oriol Vinyals and Jeff Dean.

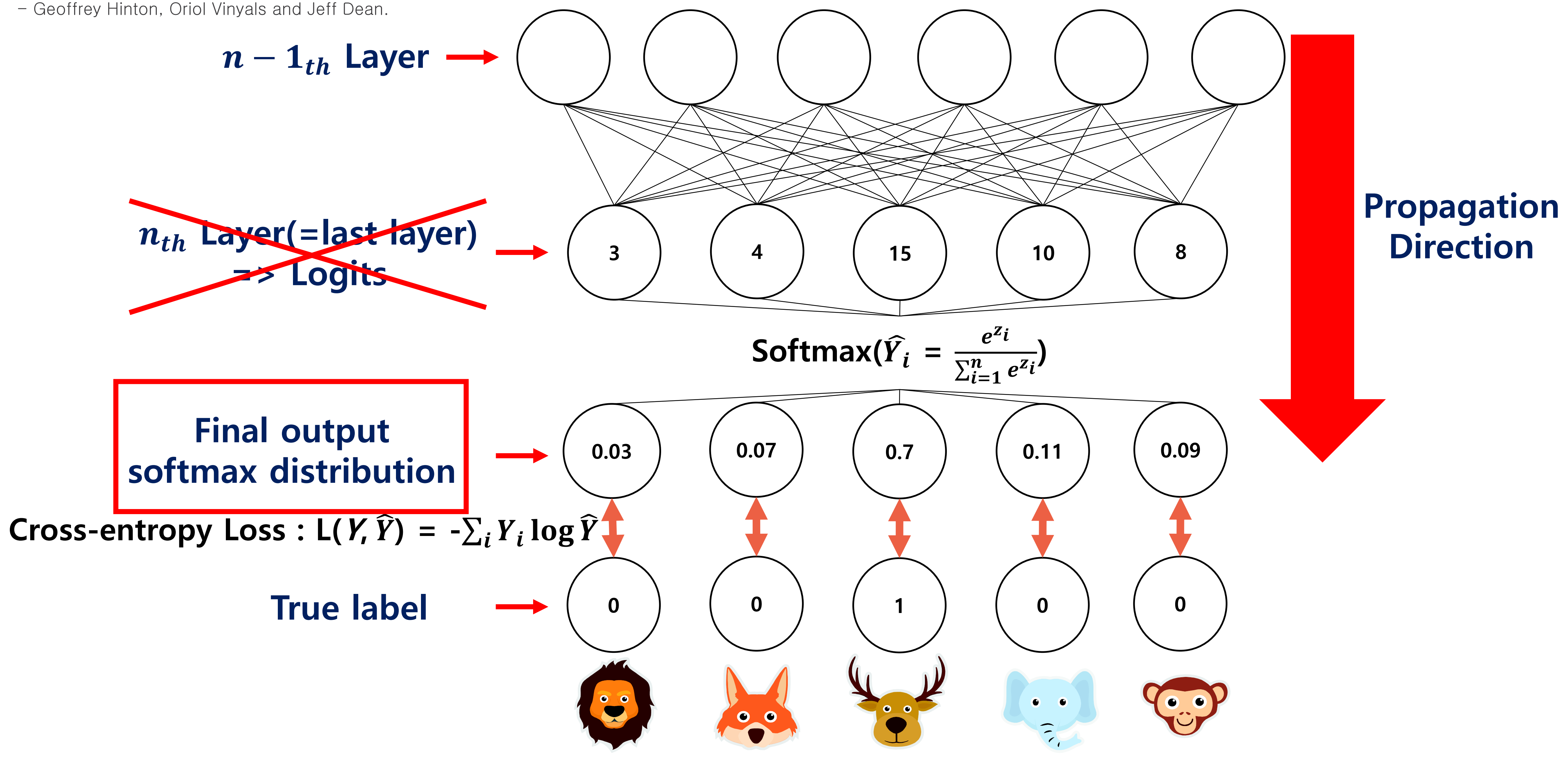


$n-1_{th}$ Layer
n_{th} Layer(=last layer) => Logits
3
4
15
10
8
Softmax($\hat{Y}_i = \frac{e^{z_i}}{\sum_{i=1}^{n} e^{z_i}}$)
Final output softmax distribution
0.03
0.07
0.7
0.11
0.09
Cross-entropy Loss : L($Y, \hat{Y}$) = -$\sum_i Y_i \log \hat{Y}$
True label
0
0
1
0
0
Propagation Direction

<Last section of DNN>

Distilling the knowledge in a Neural Network. In *NIPS workshop*, 2014.

– Geoffrey Hinton, Oriol Vinyals and Jeff Dean.


$n-1_{th}$ Layer
~~n_{th} Layer(=last layer) => Logits~~
3
4
15
10
8
Softmax($\hat{Y}_i = \frac{e^{z_i}}{\sum_{i=1}^{n} e^{z_i}}$)
Final output softmax distribution
0.03
0.07
0.7
0.11
0.09
Cross-entropy Loss : L($Y, \hat{Y}$) = -$\sum_i Y_i \log \hat{Y}$
True label
0
0
1
0
0
Propagation Direction

<Last section of DNN>

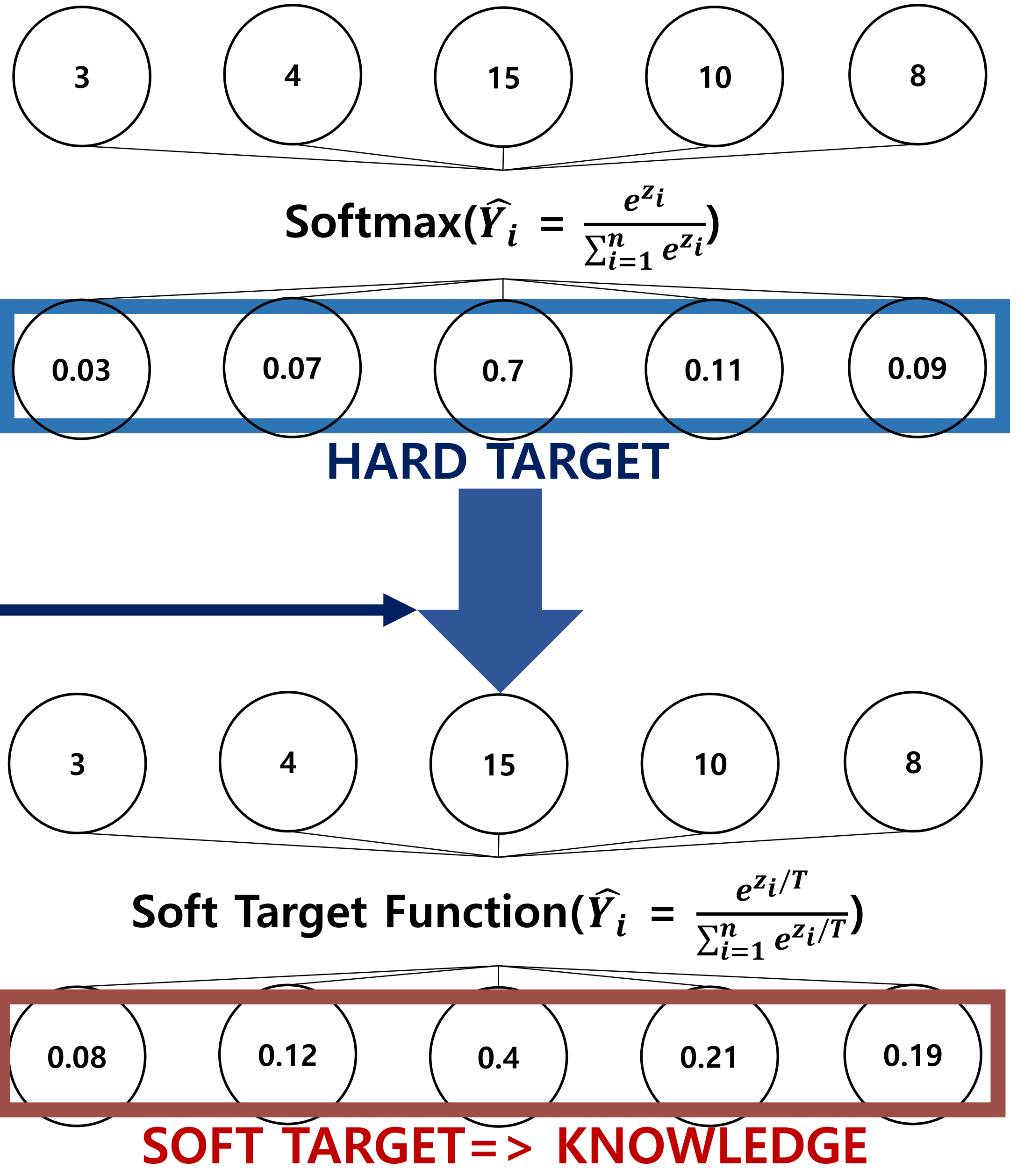
Animal pictures – www.freepik.com

Distilling the knowledge in a Neural Network. In *NIPS workshop*, 2014.

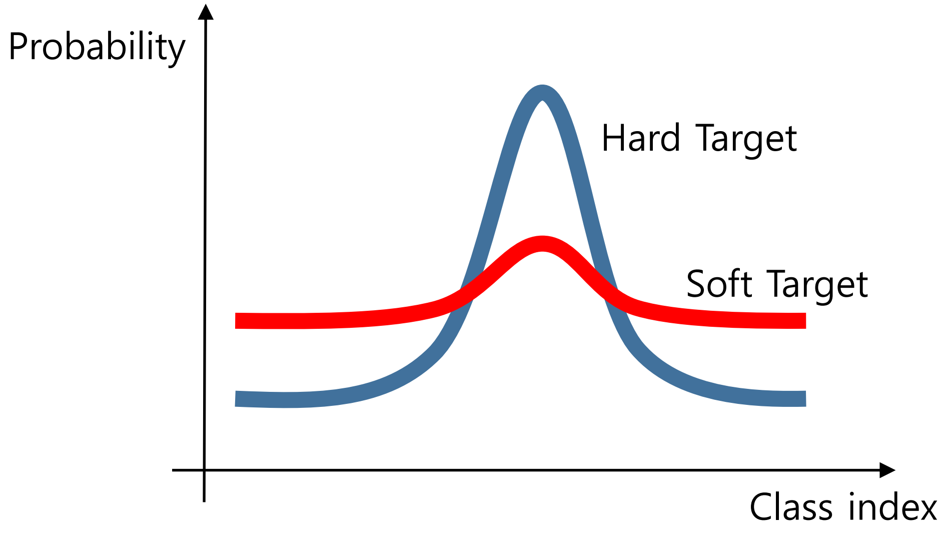
– Geoffrey Hinton, Oriol Vinyals and Jeff Dean.

$$\widehat{Y_i} = \frac{e^{z_i/T}}{\sum_{i=1}^{n} e^{z_i/T}}$$

<Soft target function>



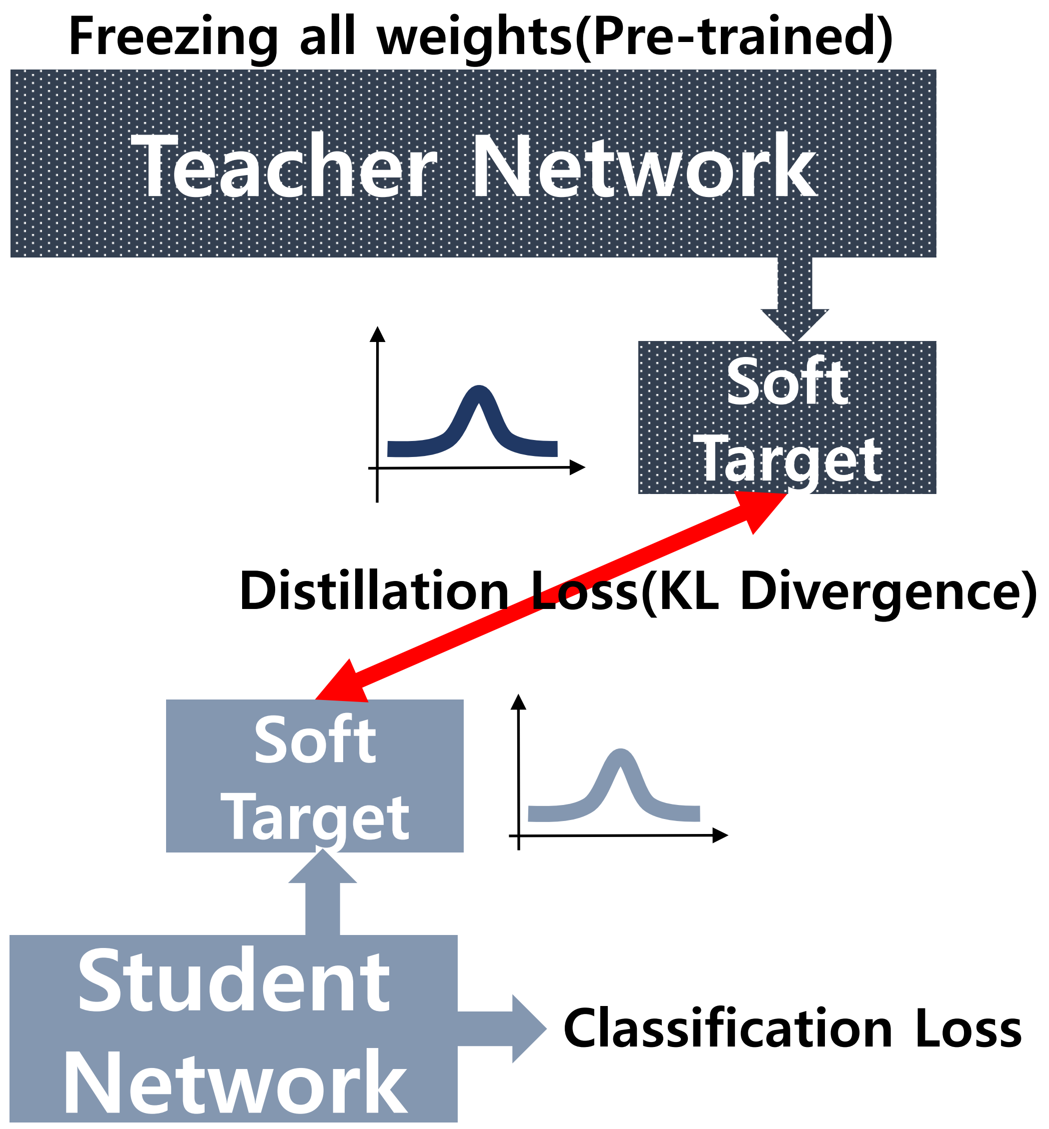
3
4
15
10
8
Softmax($\widehat{Y}_i = \frac{e^{z_i}}{\sum_{i=1}^{n} e^{z_i}}$)
0.03
0.07
0.7
0.11
0.09
HARD TARGET
3
4
15
10
8
Soft Target Function($\widehat{Y}_i = \frac{e^{z_i/T}}{\sum_{i=1}^{n} e^{z_i/T}}$)
0.08
0.12
0.4
0.21
0.19
SOFT TARGET=> KNOWLEDGE



<Soften the hard target distrubution>

Distilling the knowledge in a Neural Network. In *NIPS workshop*, 2014.

– Geoffrey Hinton, Oriol Vinyals and Jeff Dean.



<Knowledge Distillation structure>

Cat: 74%	Cat: 40%	Cat: 2%	Cat: 7%
Tiger: 12%	Tiger: 33%	Tiger: 84%	Tiger: 50%
Dog: 7%	Dog: 10%	Dog: 5%	Dog: 35%
Lion: 2%	Lion: 5%	Lion: 3%	Lion: 3%
⋮	⋮	⋮	⋮

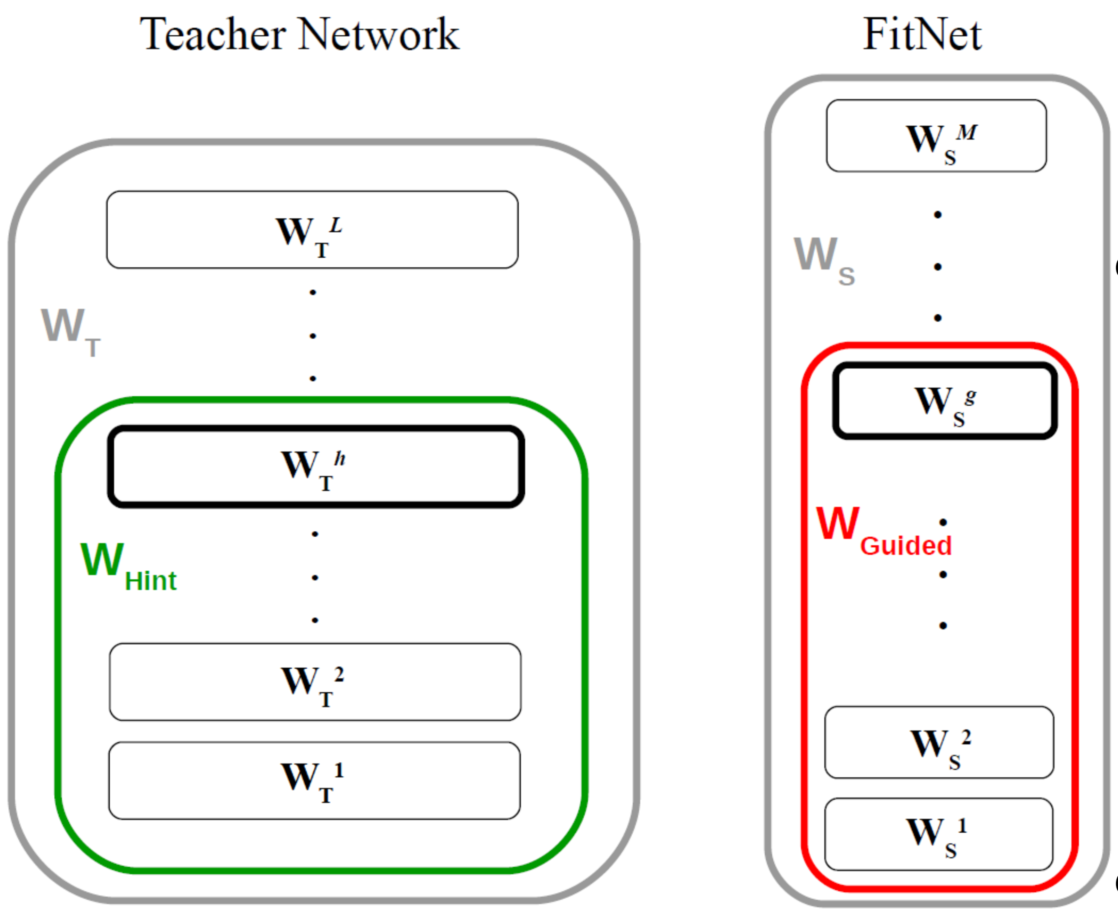
<Hard target to soft target distribution>

FitNets:
Hints for Thin Deep Nets

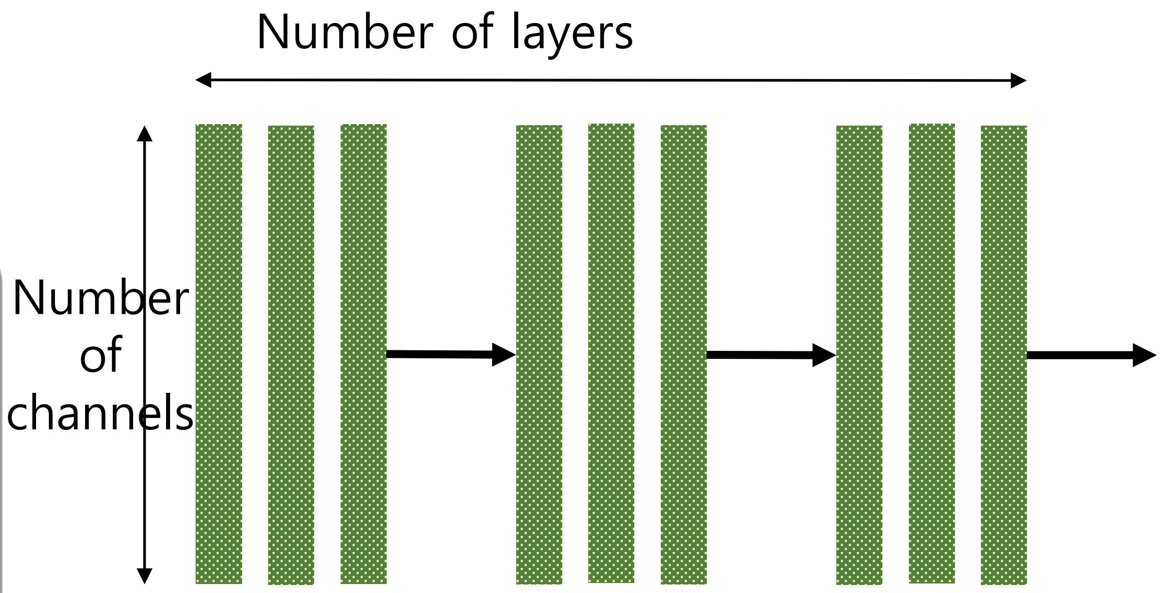
2015
ICLR

FitNets: Hints for Thin Deep Nets. In *ICLR*, 2015.

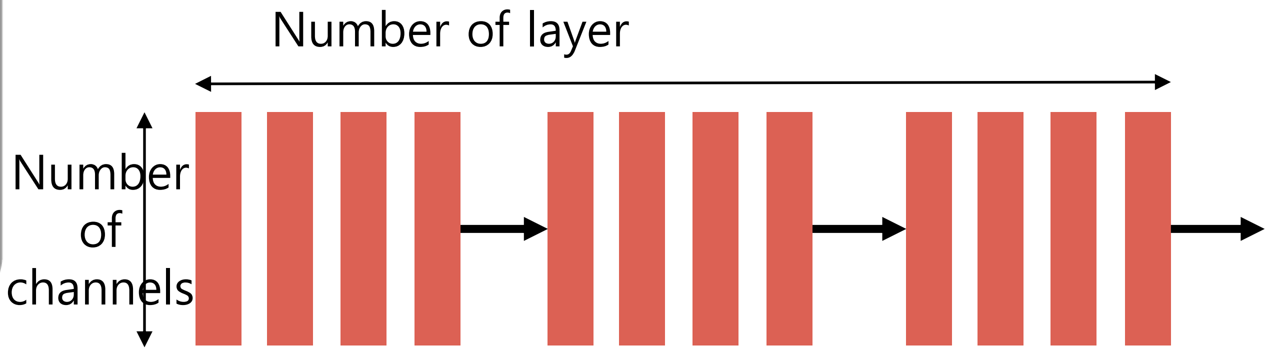
– Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta and Yoshua Bengio.



<1st step. Making thin & deeper student network>



<Pre-trained teacher network>

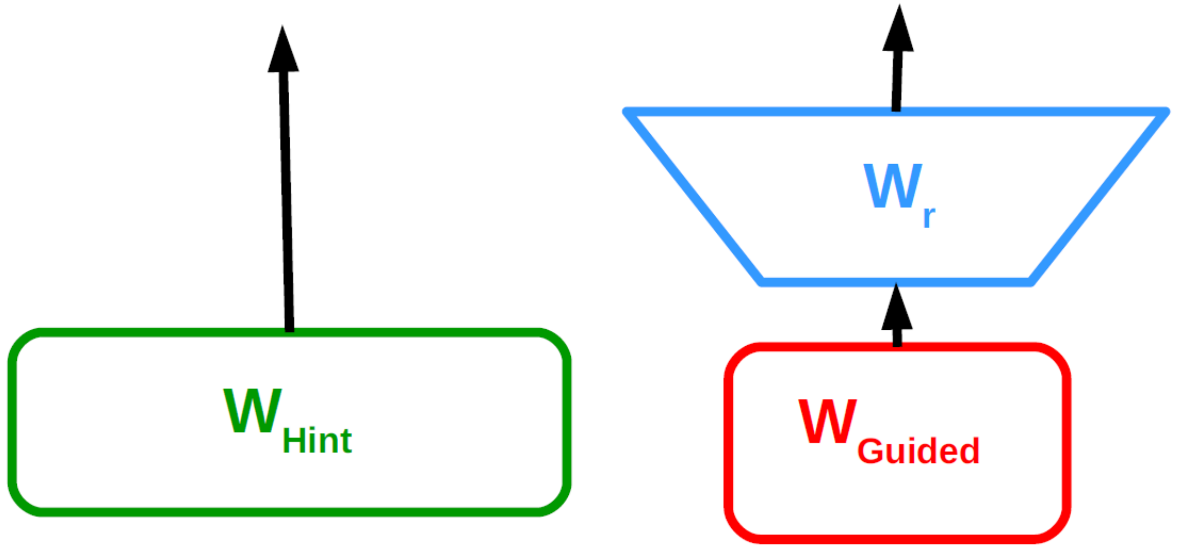


<Thin & deeper student network>

FitNets: Hints for Thin Deep Nets. In *ICLR*, 2015.

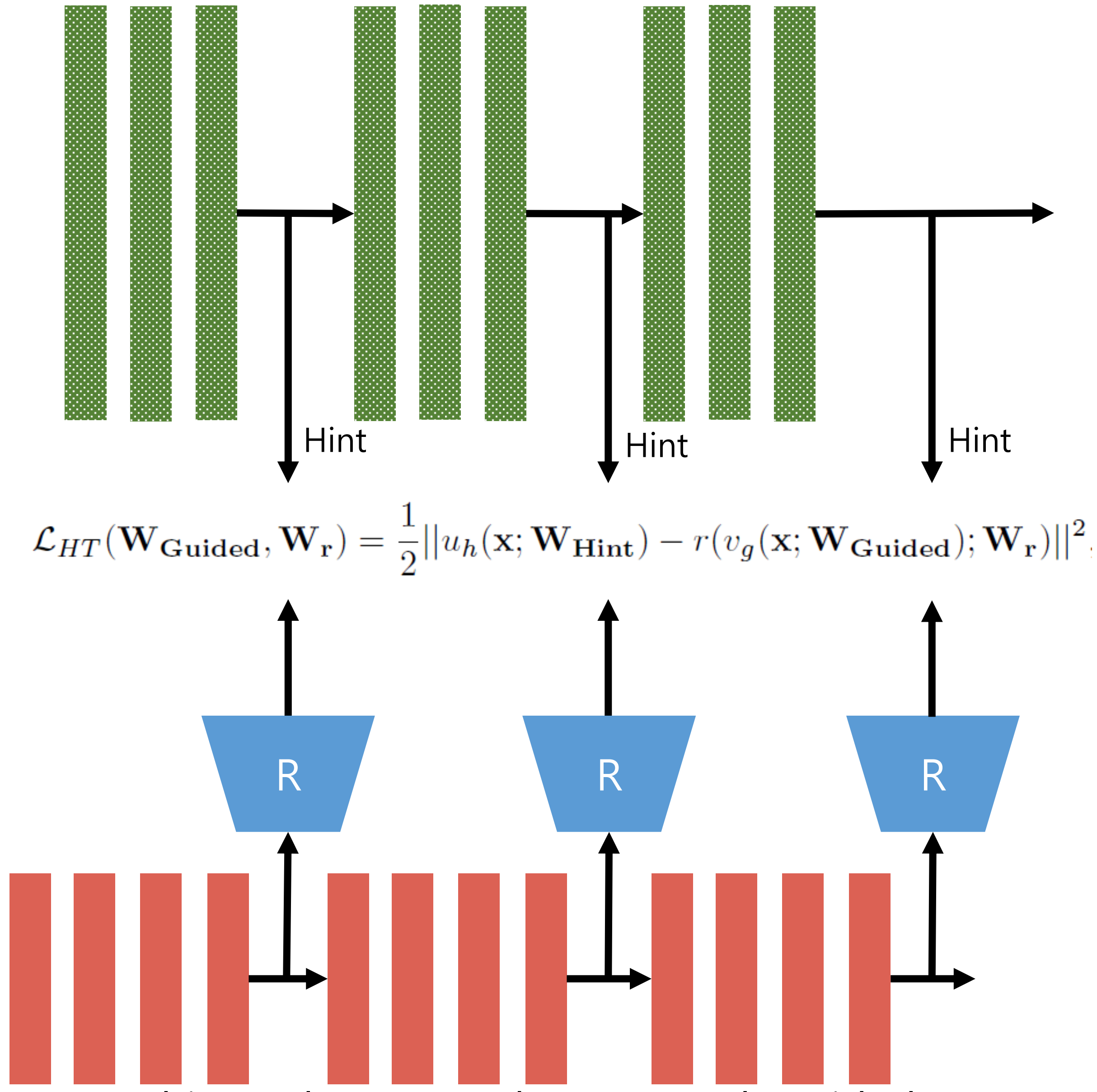
– Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta and Yoshua Bengio.

$$W^*_{Guided} = \underset{\mathbf{W}_{\mathbf{Guided}}}{\operatorname{argmin}} \mathcal{L}_{HT}(W_{Guided}, W_r)$$



<2nd step. Training through regressor>

<Pre-trained teacher network(Hint)>



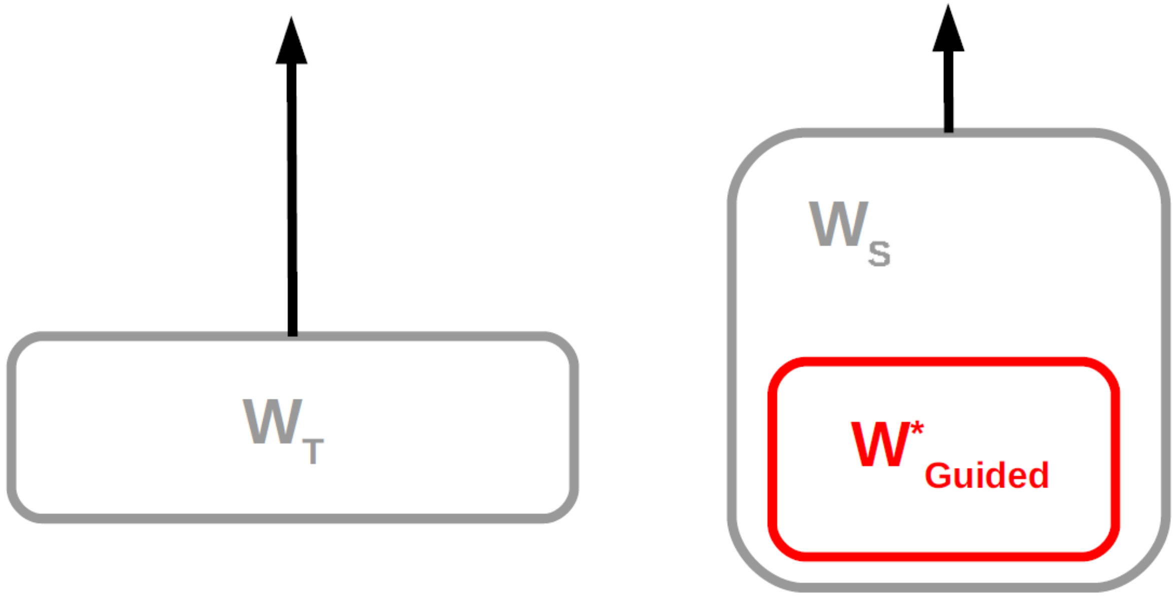
$$\mathcal{L}_{HT}(\mathbf{W}_{\mathbf{Guided}}, \mathbf{W}_{\mathbf{r}}) = \frac{1}{2}||u_h(\mathbf{x}; \mathbf{W}_{\mathbf{Hint}}) - r(v_g(\mathbf{x}; \mathbf{W}_{\mathbf{Guided}}); \mathbf{W}_{\mathbf{r}})||^2$$

<Thin & deeper student network(Guided)>

FitNets: Hints for Thin Deep Nets. In *ICLR*, 2015.

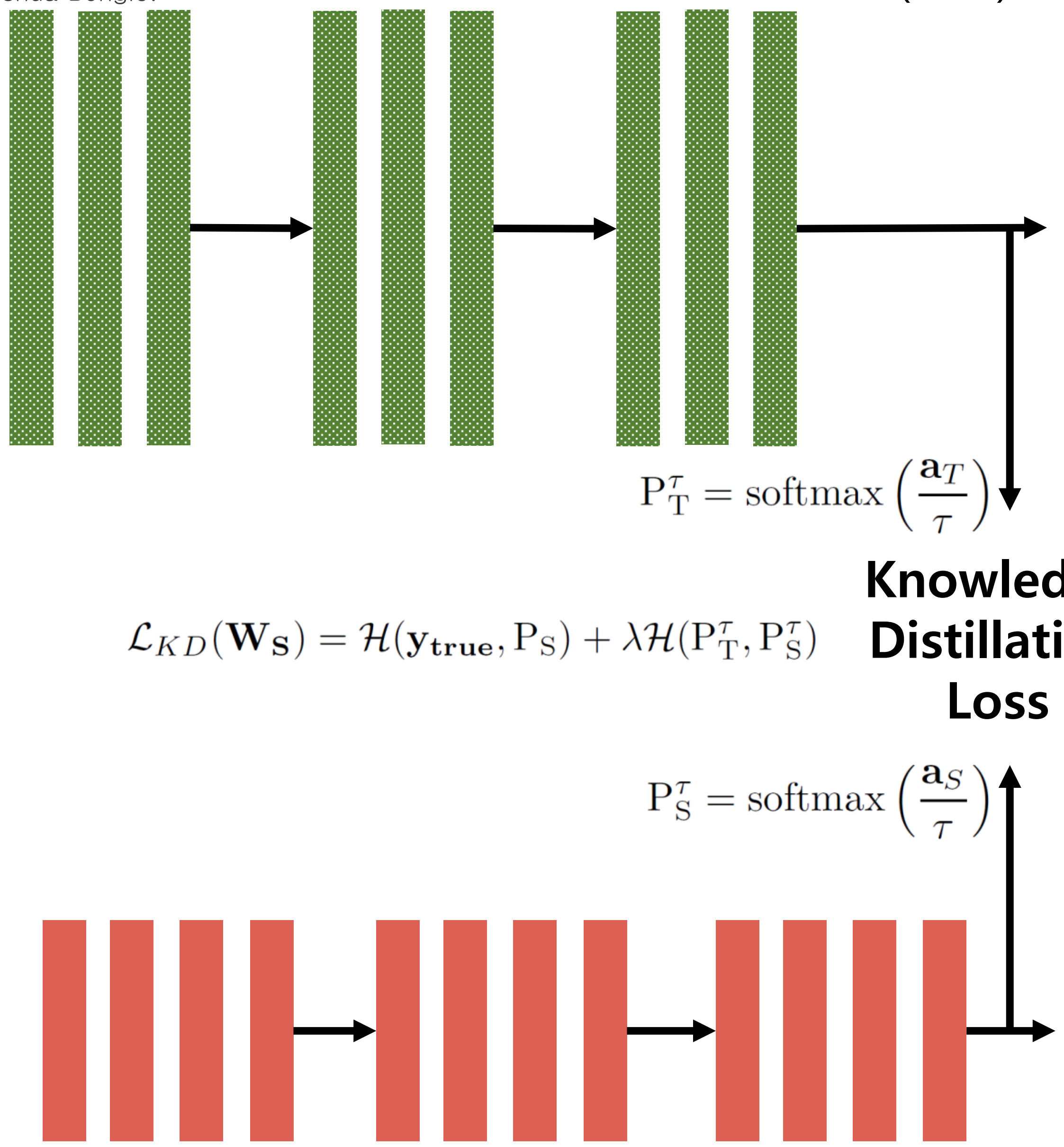
– Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta and Yoshua Bengio.

$$\mathbf{W}^*_s = \underset{\mathbf{W_S}}{\operatorname{argmin}} \mathcal{L}_{DK}(\mathbf{W}_s)$$



<3rd step. Training through KD>

<Pre-trained teacher network(Hint)>



$$\mathrm{P}^{\tau}_{\mathrm{T}} = \mathrm{softmax}\left(\frac{\mathbf{a}_T}{\tau}\right)$$

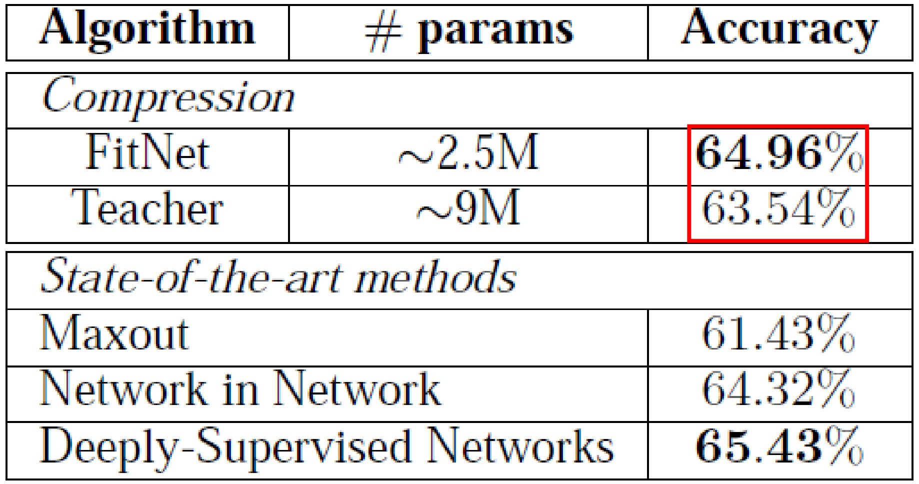
$$\mathcal{L}_{KD}(\mathbf{W_S}) = \mathcal{H}(\mathbf{y_{true}}, \mathrm{P_S}) + \lambda\mathcal{H}(\mathrm{P}^{\tau}_{\mathrm{T}}, \mathrm{P}^{\tau}_{\mathrm{S}})$$

$$\mathrm{P}^{\tau}_{\mathrm{S}} = \mathrm{softmax}\left(\frac{\mathbf{a}_S}{\tau}\right)$$

<Thin & deeper student network(Guided)>

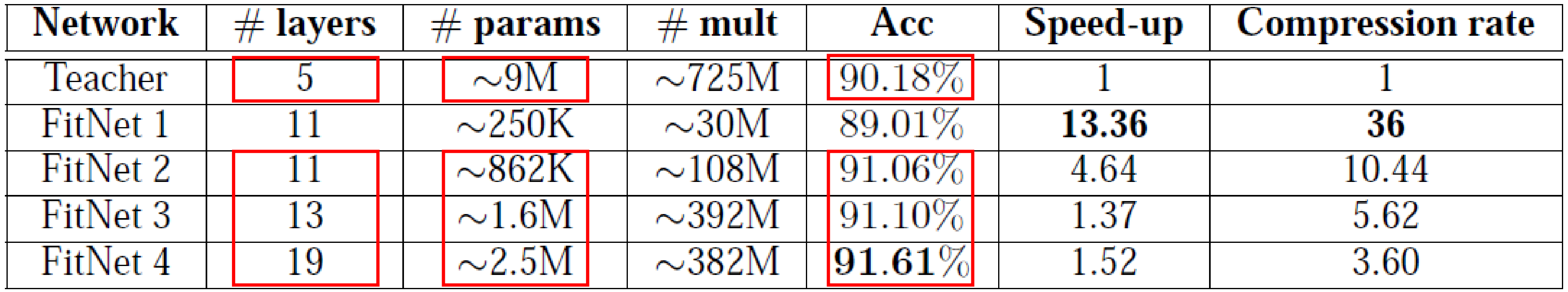
FitNets: Hints for Thin Deep Nets. In *ICLR*, 2015.

– Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta and Yoshua Bengio.



Algorithm	**# params**	**Accuracy**
Compression		
FitNet	~2.5M	**64.96%**
Teacher	~9M	63.54%
State-of-the-art methods		
Maxout		61.43%
Network in Network		64.32%
Deeply-Supervised Networks		**65.43%**

Table 2: Accuracy on CIFAR-100



Network	**# layers**	**# params**	**# mult**	**Acc**	**Speed-up**	**Compression rate**
Teacher	5	~9M	~725M	90.18%	1	1
FitNet 1	11	~250K	~30M	89.01%	**13.36**	**36**
FitNet 2	11	~862K	~108M	91.06%	4.64	10.44
FitNet 3	13	~1.6M	~392M	91.10%	1.37	5.62
FitNet 4	19	~2.5M	~382M	**91.61%**	1.52	3.60

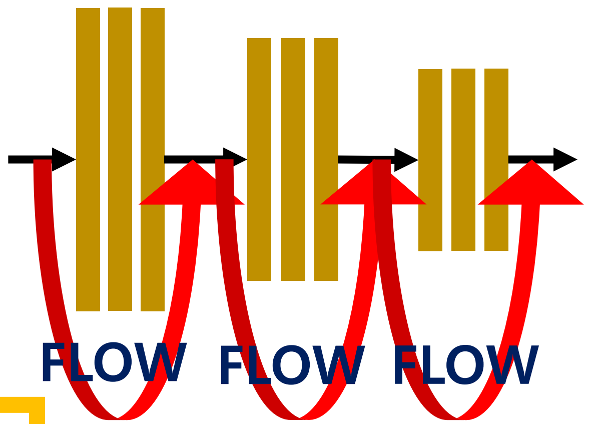
Table 5: Accuracy/Speed Trade-off on CIFAR-10.

A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning

2017
CVPR

A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning. In *CVPR*, 2017.

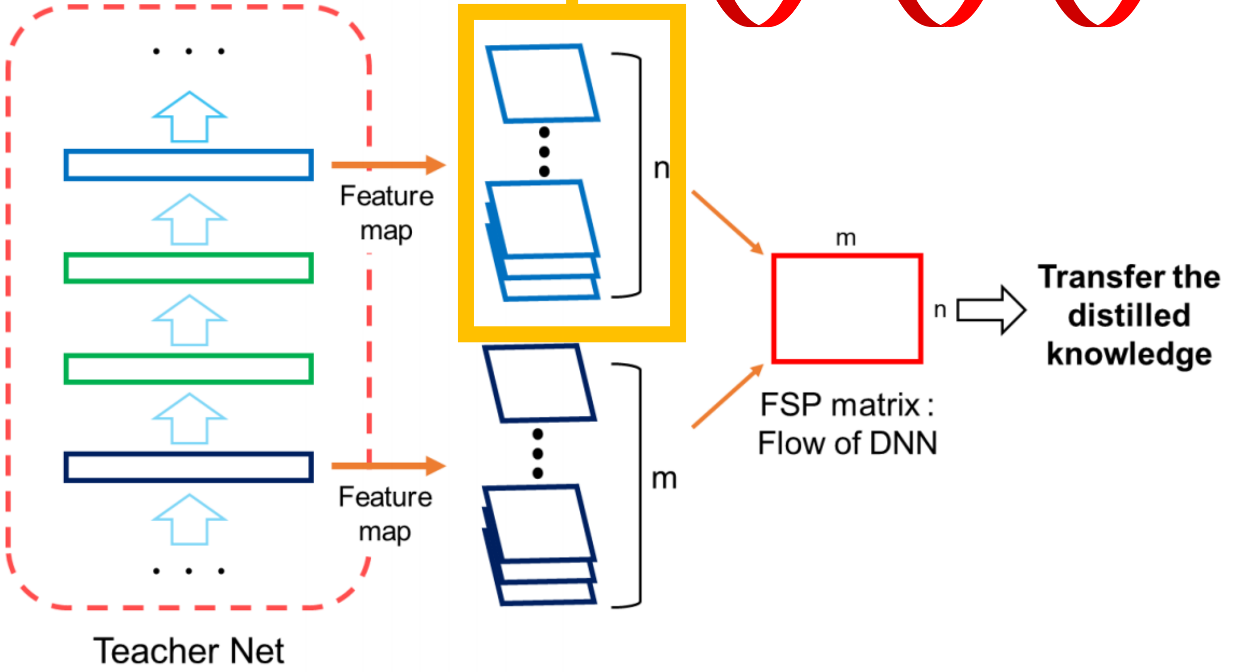
– Junho Yim, Donggyu Joo, Jihoon Bae and Junmo Kim.



$$G(a_1, \ldots, a_n) = \begin{pmatrix} (a_1, a_1) & (a_1, a_2) & \cdots & (a_1, a_n) \\ (a_2, a_1) & (a_2, a_2) & \cdots & (a_2, a_n) \\ \vdots & \vdots & \ddots & \vdots \\ (a_n, a_1) & (a_n, a_2) & \cdots & (a_n, a_n) \end{pmatrix}$$

<Gram Matrix definition>

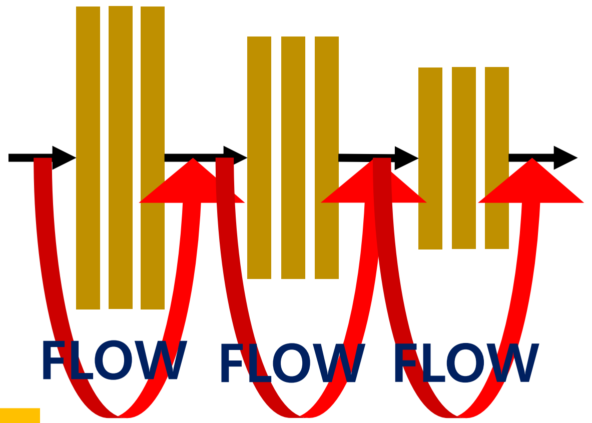
* (a, b) = Inner product of a and b



<Gram Matrix with across layers>

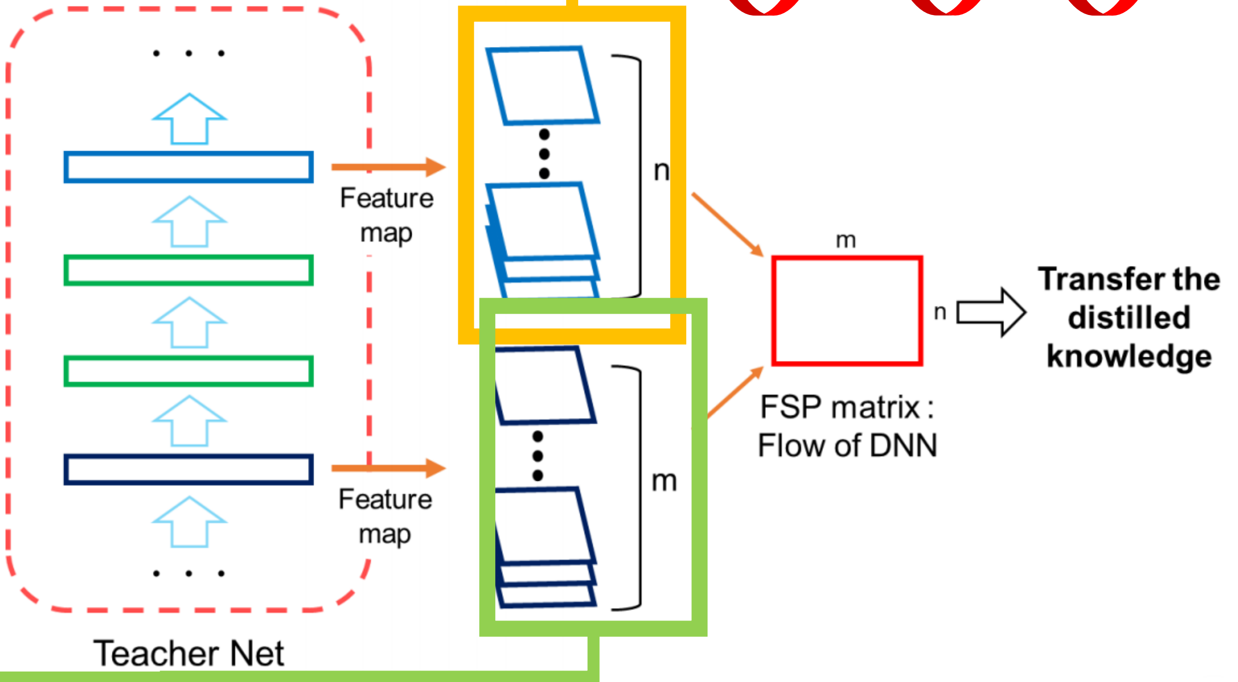
A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning. In *CVPR*, 2017.

– Junho Yim, Donggyu Joo, Jihoon Bae and Junmo Kim.



$$G(a_1, \dots, a_n, b_1, \dots, b_m) = \begin{pmatrix} (a_1, b_1) & (a_1, b_2) & \cdots & (a_1, b_m) \\ (a_2, b_1) & (a_2, b_2) & \cdots & (a_2, b_m) \\ \vdots & \vdots & \ddots & \vdots \\ (a_n, b_1) & (a_n, b_2) & \cdots & (a_n, b_m) \end{pmatrix}$$

<Flow of solution procedure(FSP) matrix>

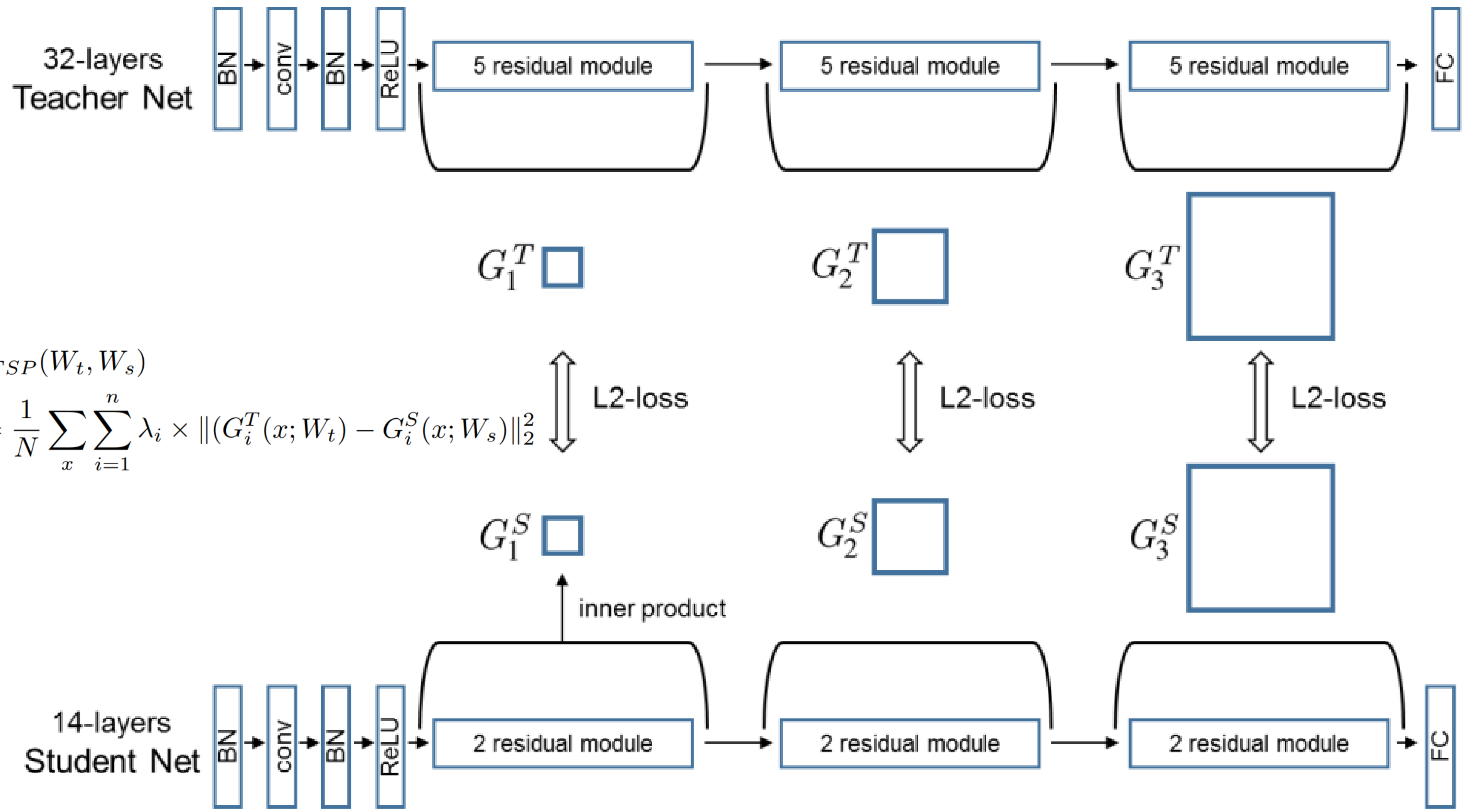


<Gram Matrix with across layers>

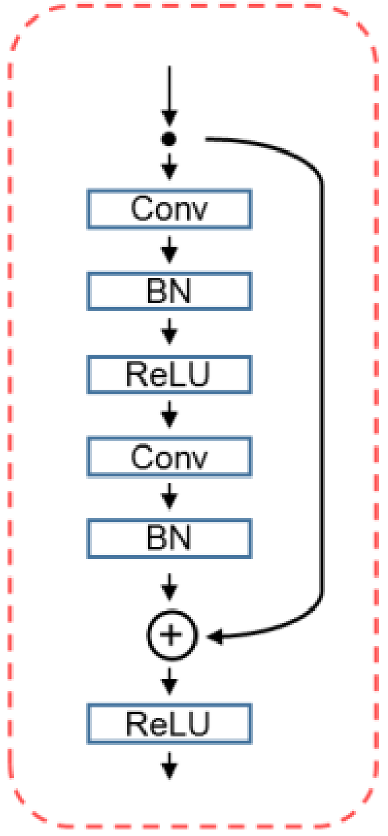
* (a, b) = Inner product of a and b

A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning. In *CVPR*, 2017.

– Junho Yim, Donggyu Joo, Jihoon Bae and Junmo Kim.



$$L_{FSP}(W_t, W_s) = \frac{1}{N}\sum_{x}\sum_{i=1}^{n}\lambda_i \times \|(G_i^T(x;W_t) - G_i^S(x;W_s)\|_2^2$$

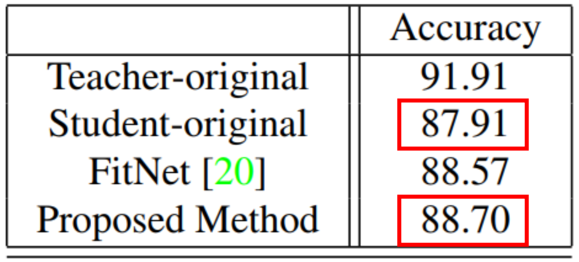


Residual module

<Distillation Structure>

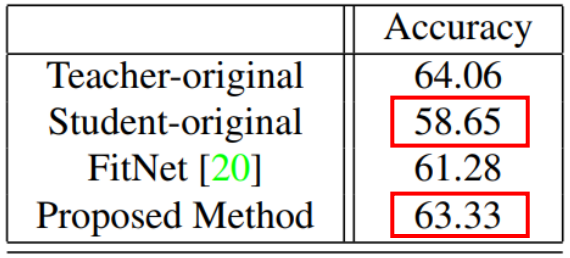
A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning. In *CVPR*, 2017.

– Junho Yim, Donggyu Joo, Jihoon Bae and Junmo Kim.



	Accuracy
Teacher-original	91.91
Student-original	87.91
FitNet [20]	88.57
Proposed Method	88.70

Table 3. Recognition rates (%) on CIFAR-10. We used a residual DNN with 8 layers for the student DNN and 26 layers for the teacher DNN.



	Accuracy
Teacher-original	64.06
Student-original	58.65
FitNet [20]	61.28
Proposed Method	63.33

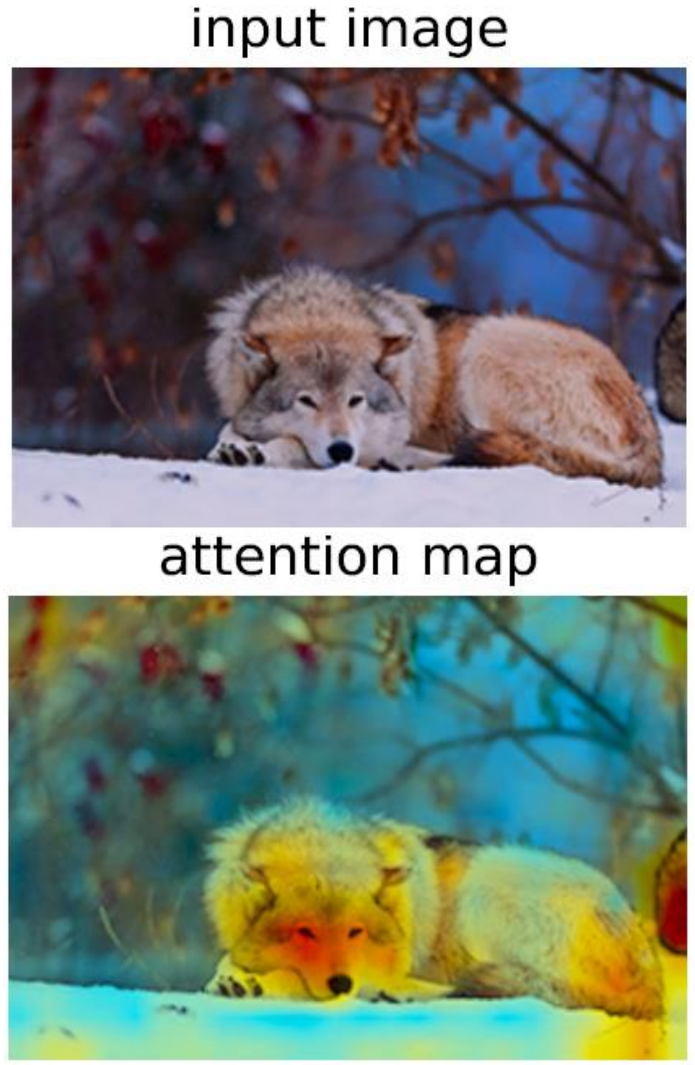
Table 4. Recognition rates (%) on CIFAR-100. We used a residual DNN with 14 layers for the student DNN and 32 layers for the teacher DNN.

Paying More Attention to Attention: Improving the Performance of CNN via Attention Transfer

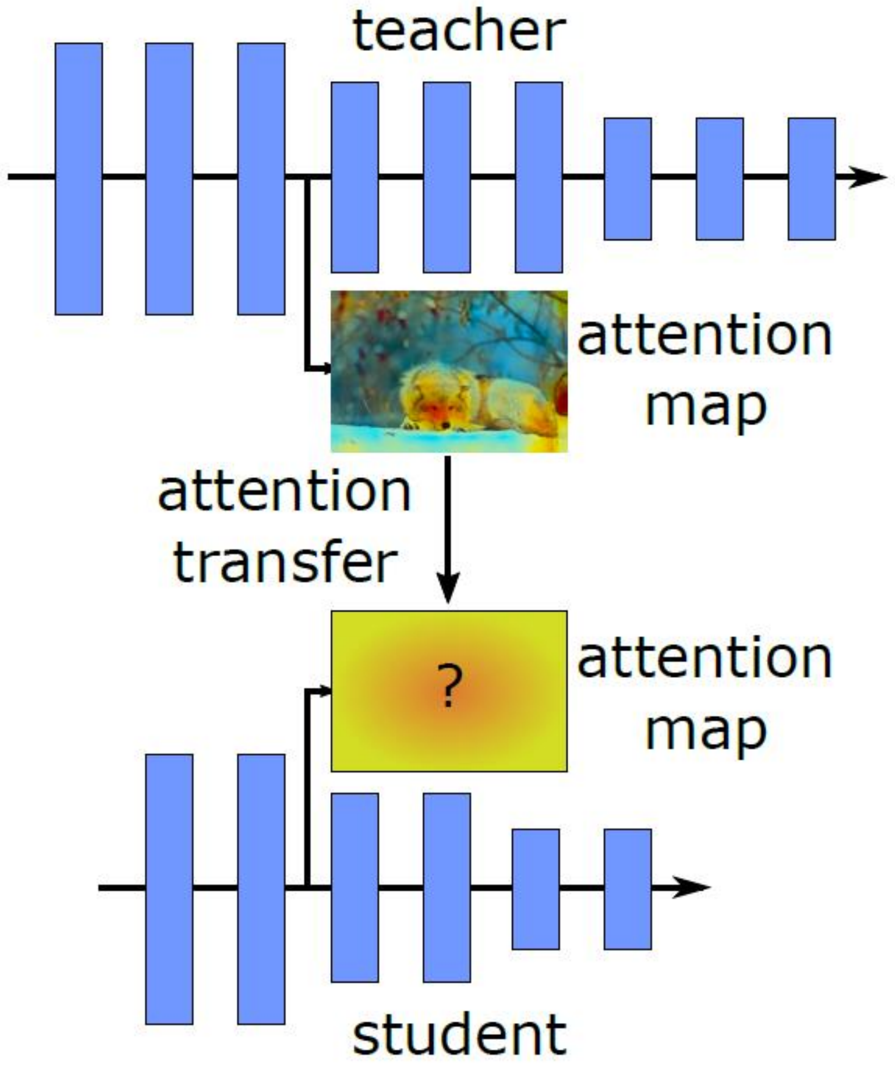
2017
ICLR

Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *ICLR*, 2017.

– Sergey Zagoruyko and Nikos Komodakis.



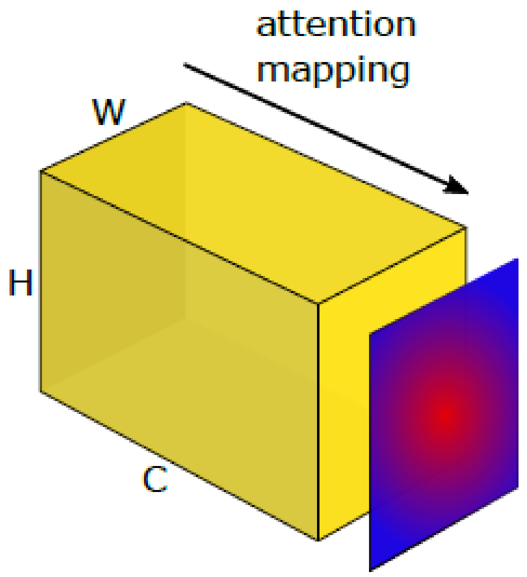
<Normal image & Attention map>



<Attention Transfer>

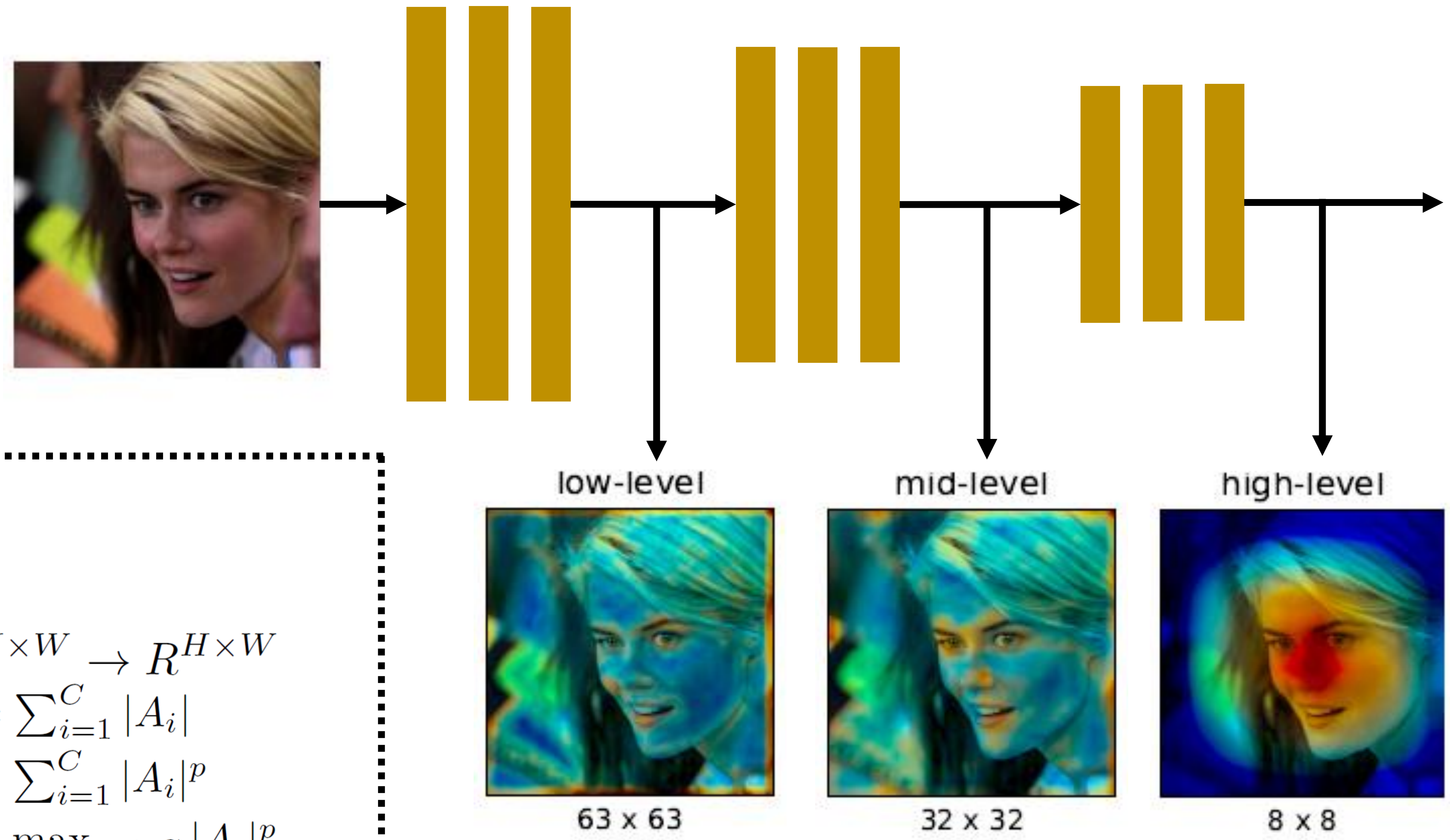
Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *ICLR*, 2017.

– Sergey Zagoruyko and Nikos Komodakis.



$$\mathcal{F} : R^{C\times H\times W} \rightarrow R^{H\times W}$$
$$F_{\mathrm{sum}}(A) = \sum_{i=1}^{C} |A_i|$$
$$F^p_{\mathrm{sum}}(A) = \sum_{i=1}^{C} |A_i|^p$$
$$F^p_{\mathrm{max}}(A) = \max_{i=1,C} |A_i|^p$$

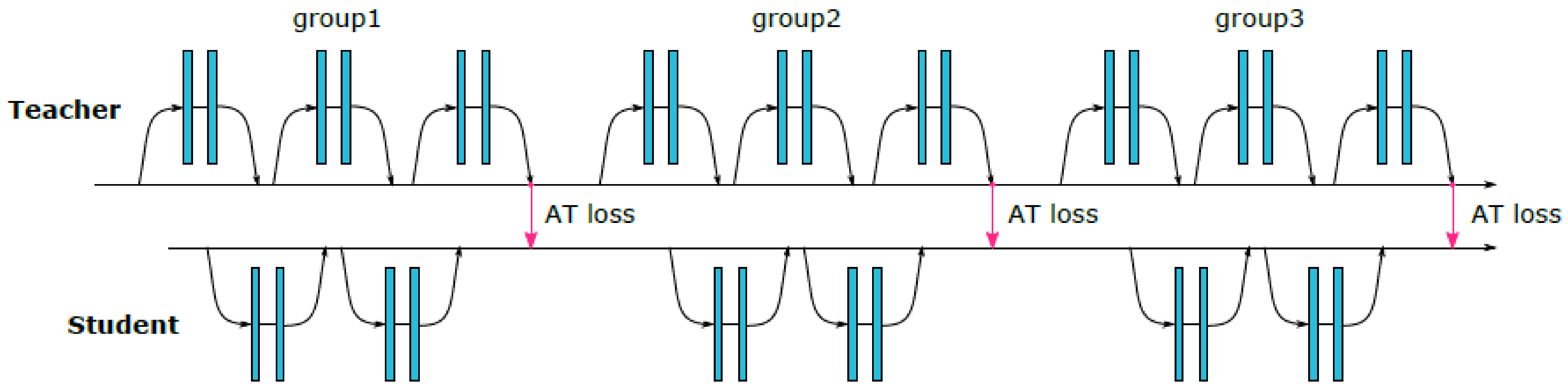
<Attention mapping functions>



<Feature characteristic of each layer>

Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *ICLR*, 2017.

– Sergey Zagoruyko and Nikos Komodakis.



$$\mathcal{L}_{AT} = \mathcal{L}(\mathbf{W}_S, x) + \frac{\beta}{2} \sum_{j \in \mathcal{I}} \| \frac{Q_S^j}{\|Q_S^j\|_2} - \frac{Q_T^j}{\|Q_T^j\|_2} \|_p$$

<Attention Transfer structure>

Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *ICLR*, 2017.

– Sergey Zagoruyko and Nikos Komodakis.

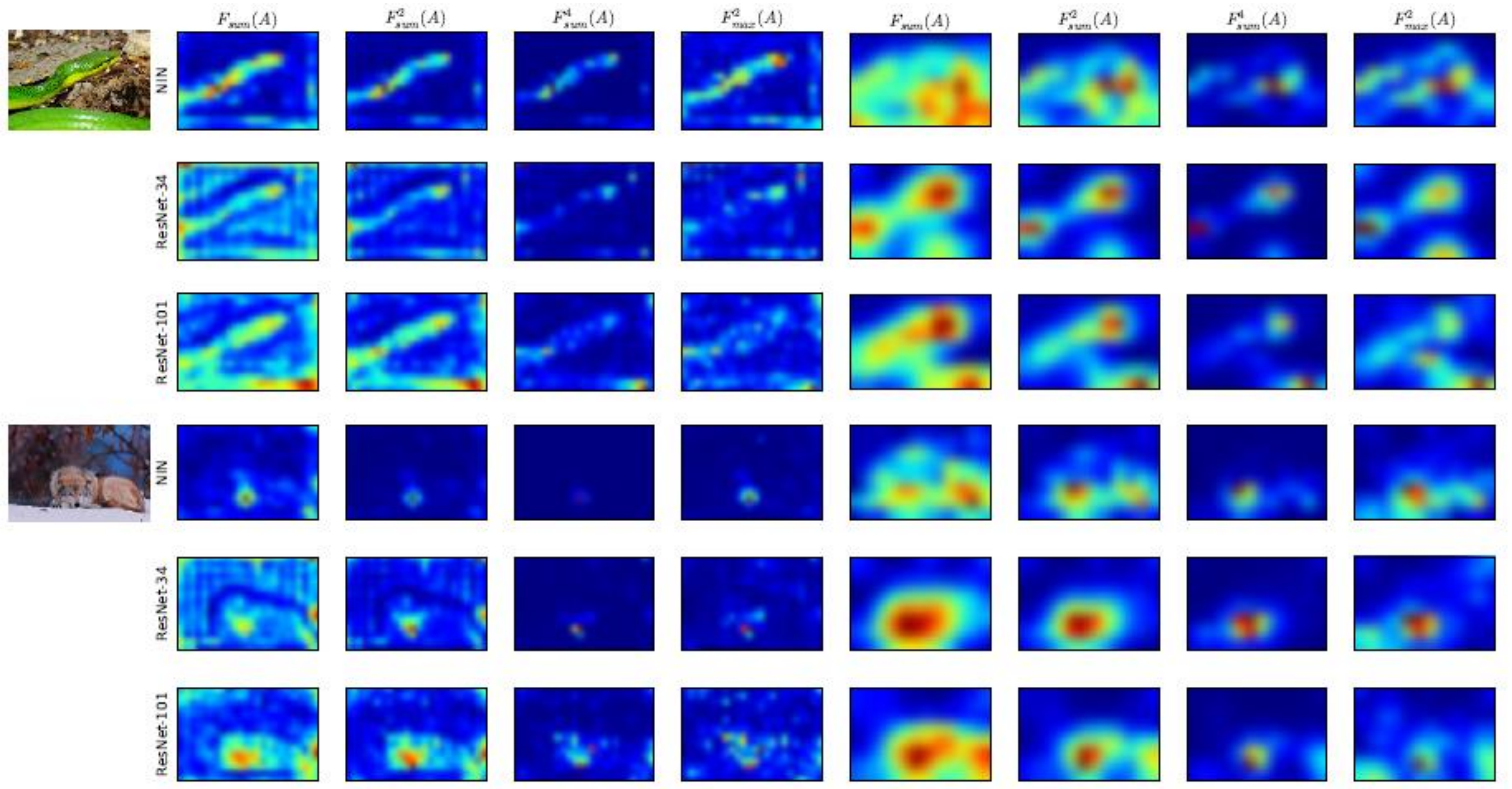
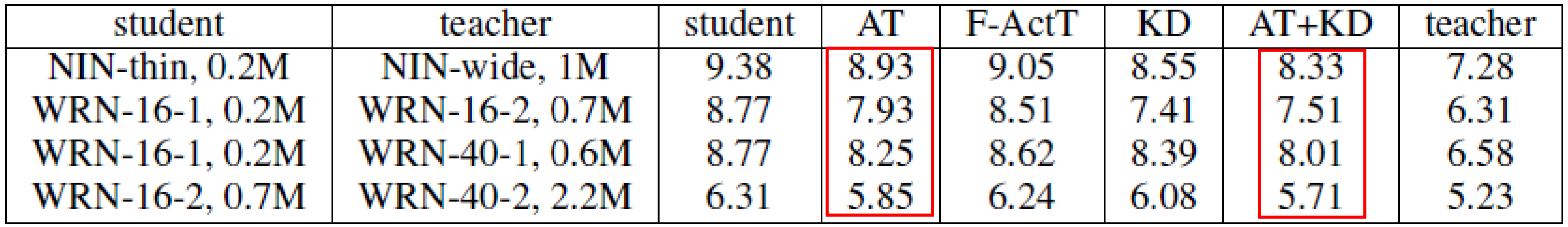


Figure 4: Activation attention maps for various ImageNet networks: Network-In-Network (62% top-1 val accuracy), ResNet-34 (73% top-1 val accuracy), ResNet-101 (77.3% top-1 val accuracy). Left part: mid-level activations, right part: top-level pre-softmax acivations

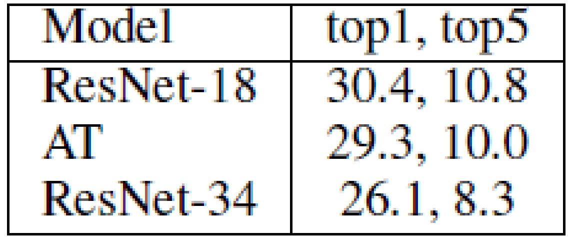
Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *ICLR*, 2017.

– Sergey Zagoruyko and Nikos Komodakis.



student	teacher	student	AT	F-ActT	KD	AT+KD	teacher
NIN-thin, 0.2M	NIN-wide, 1M	9.38	8.93	9.05	8.55	8.33	7.28
WRN-16-1, 0.2M	WRN-16-2, 0.7M	8.77	7.93	8.51	7.41	7.51	6.31
WRN-16-1, 0.2M	WRN-40-1, 0.6M	8.77	8.25	8.62	8.39	8.01	6.58
WRN-16-2, 0.7M	WRN-40-2, 2.2M	6.31	5.85	6.24	6.08	5.71	5.23

Table 1: Activation-based attention transfer (AT) with various architectures on CIFAR-10. Error is computed as median of 5 runs with different seed. F-ActT means full-activation transfer (see §4.1.2).



Model	top1, top5
ResNet-18	30.4, 10.8
AT	29.3, 10.0
ResNet-34	26.1, 8.3

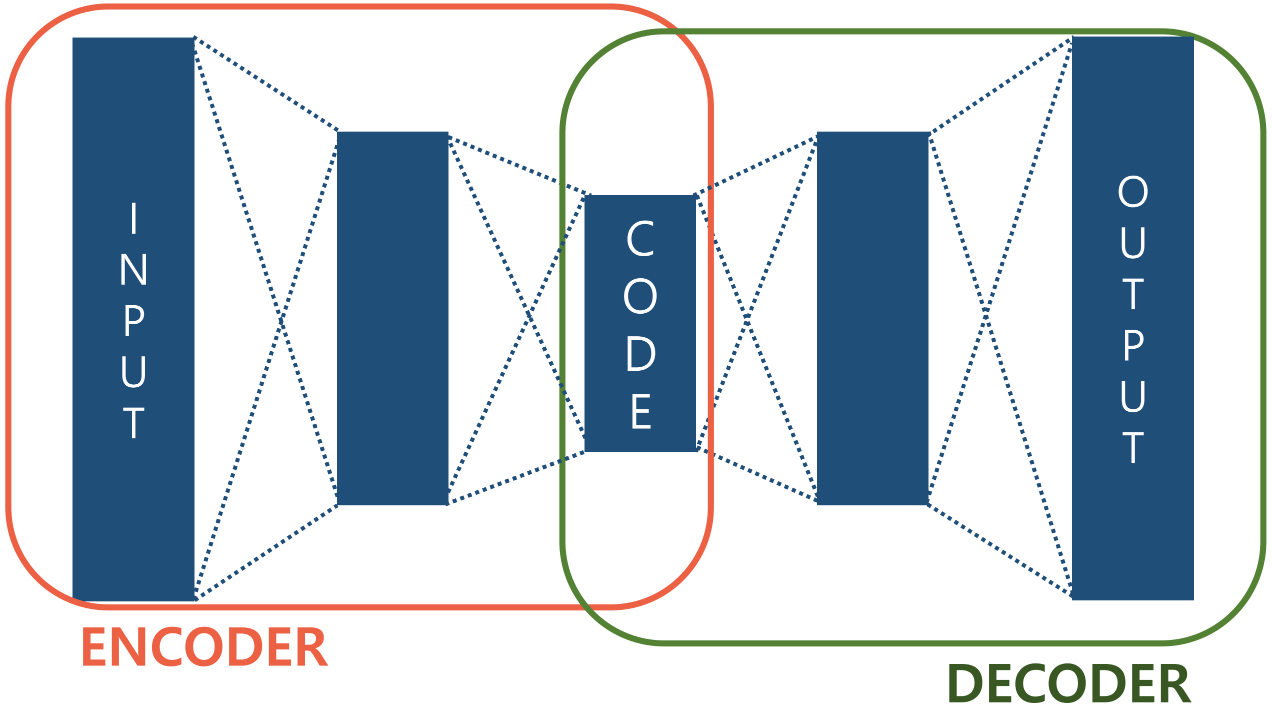
Table 5: Attention transfer validation error (single crop) on ImageNet. Transfer losses are added on epoch 60/100.

Paraphrasing Complex Network: Network Compression via Factor Transfer

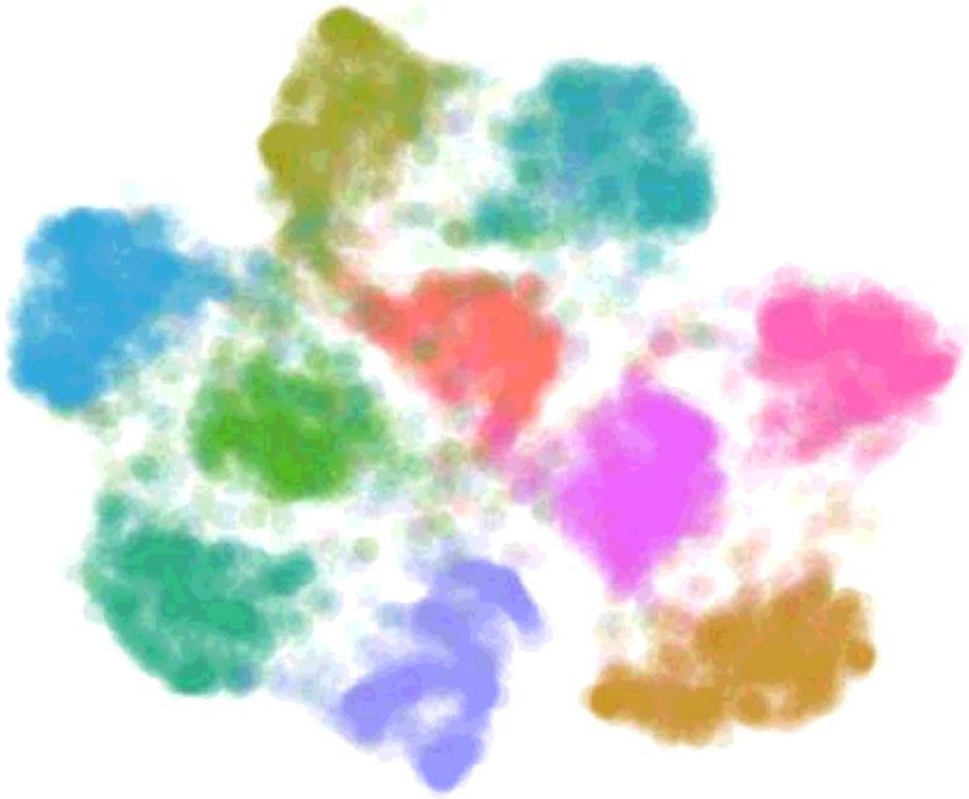
2018
NIPS

Paraphrasing Complex Network: Network Compression via Factor Transfer. In *NIPS*, 2018.

– Jangho Kim, SeongUk Park and Nojun Kwak.



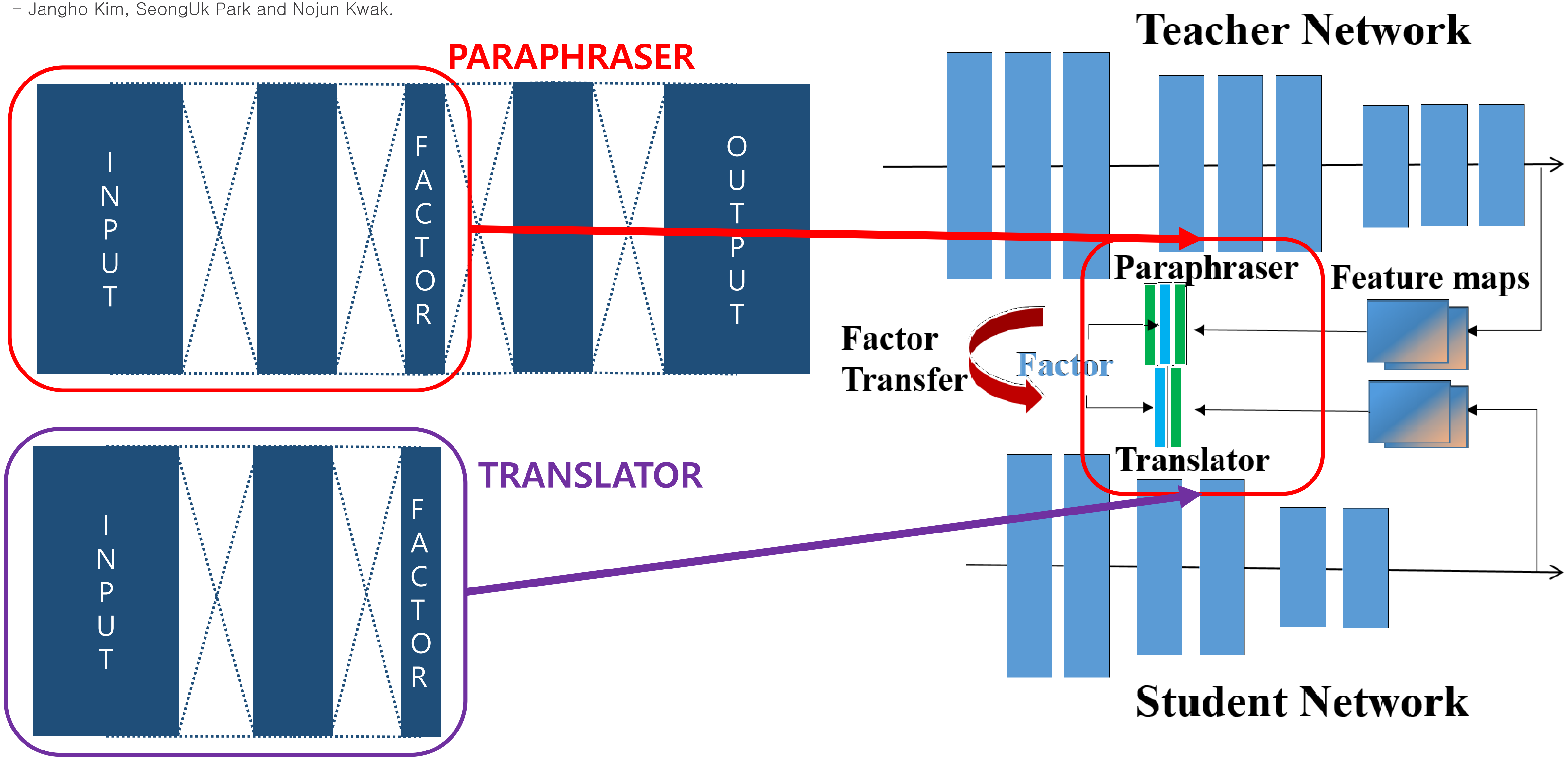
<Autoencoder structure>



<t-SNE visualization of factor space>

Paraphrasing Complex Network: Network Compression via Factor Transfer. In *NIPS*, 2018.

– Jangho Kim, SeongUk Park and Nojun Kwak.

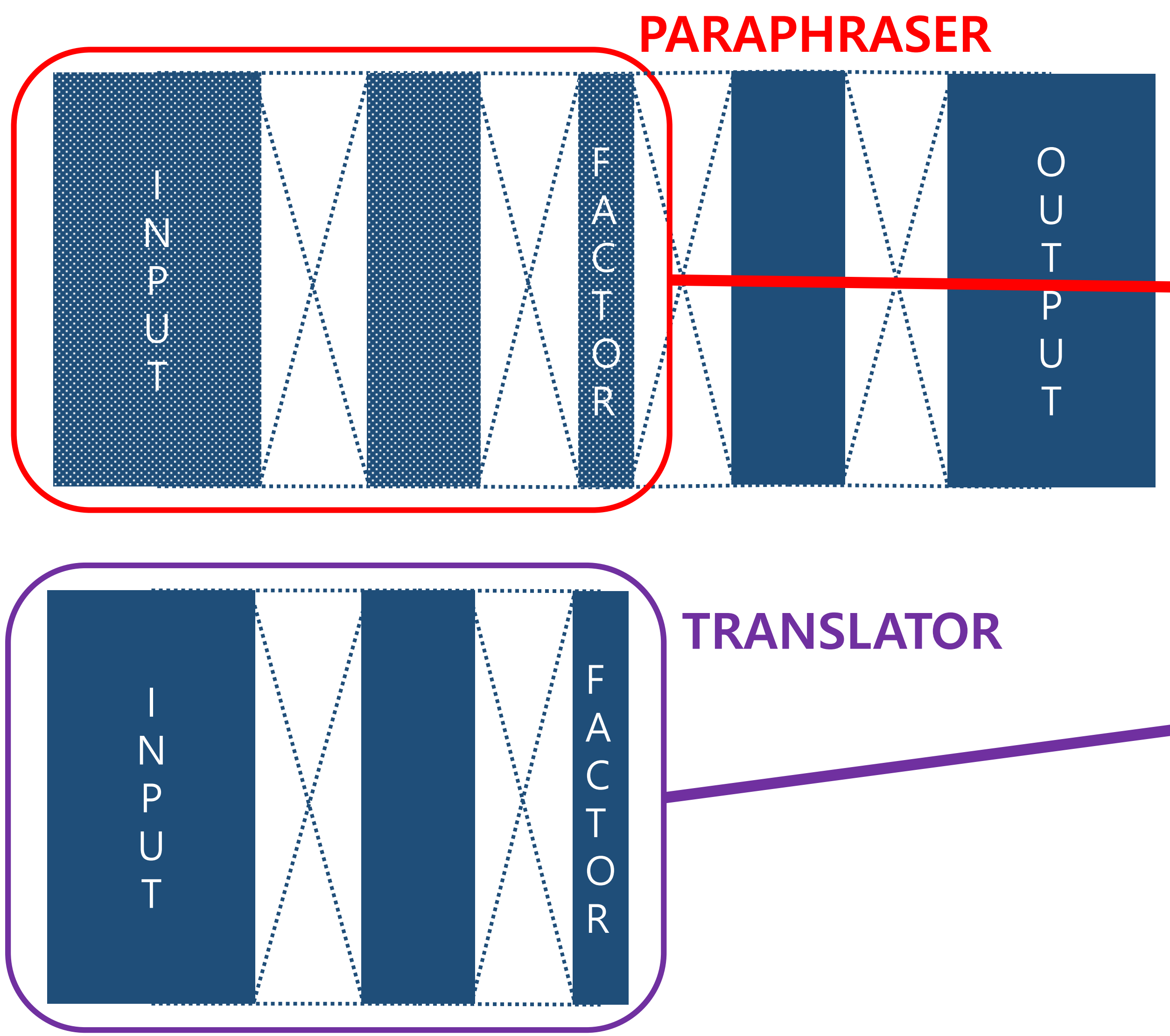


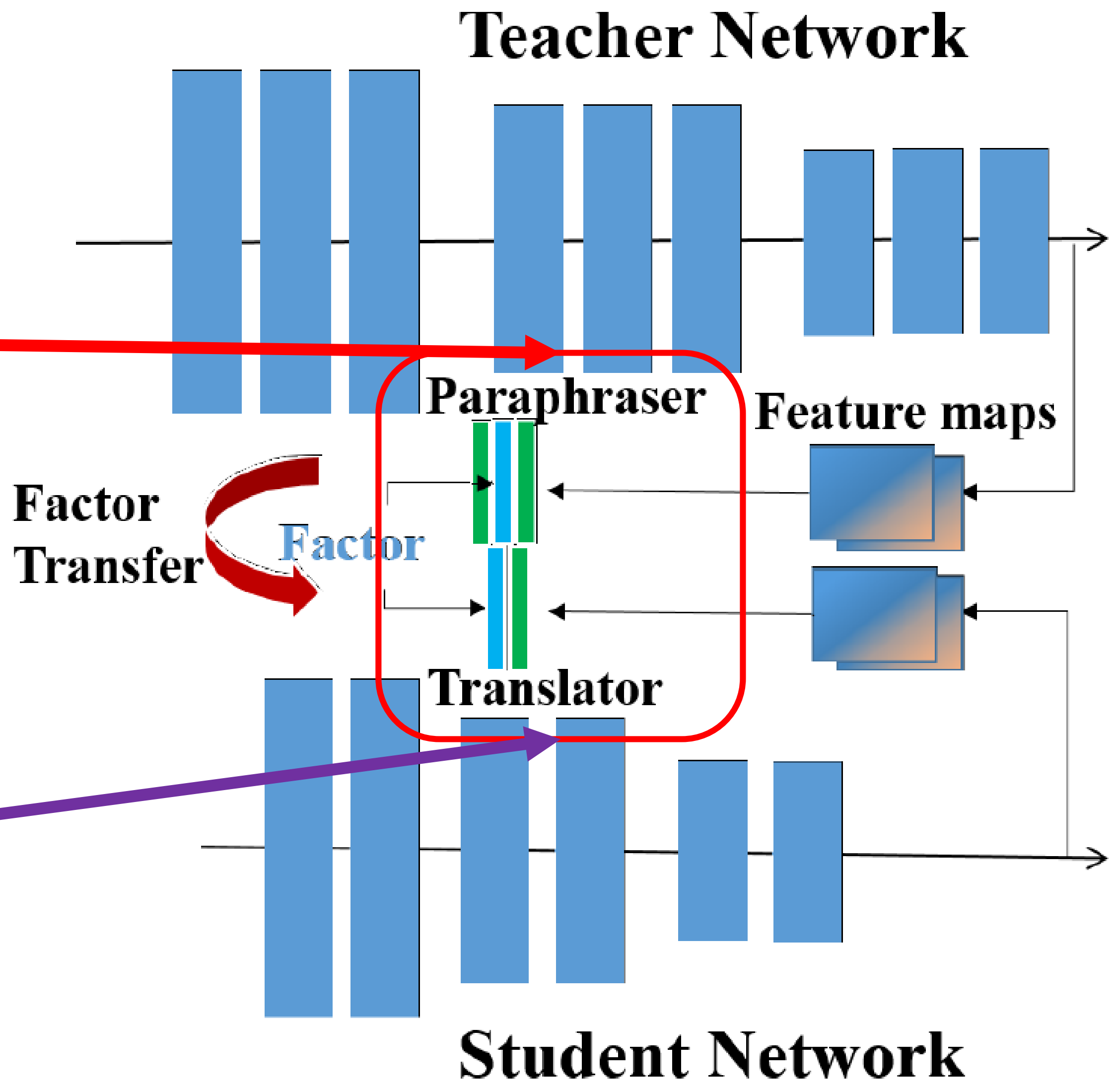
<Paraphraser & Translator structure>

<Factor Transfer>

Paraphrasing Complex Network: Network Compression via Factor Transfer. In *NIPS*, 2018.

– Jangho Kim, SeongUk Park and Nojun Kwak.



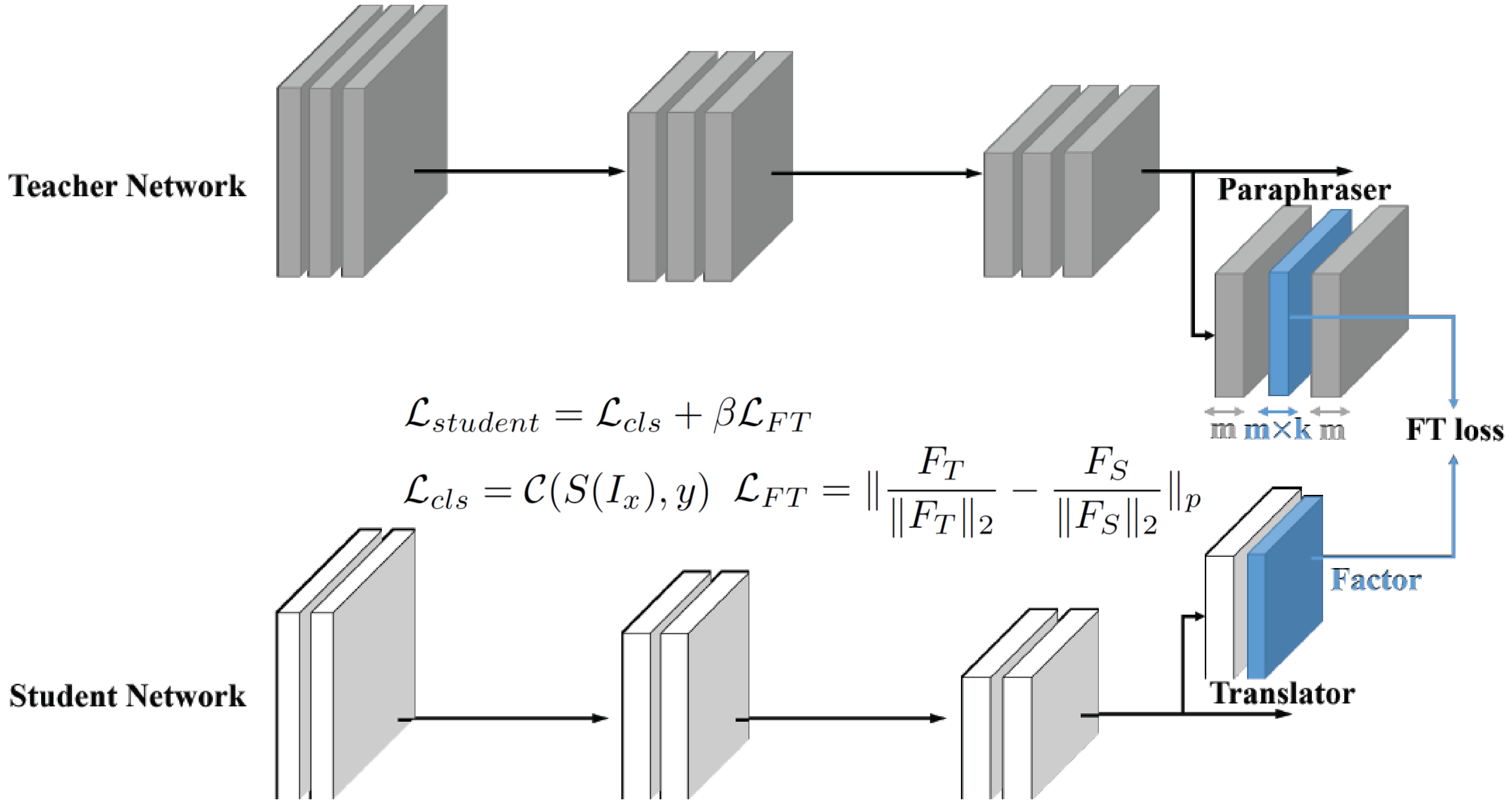


<Paraphraser & Translator structure>

<Factor Transfer>

Paraphrasing Complex Network: Network Compression via Factor Transfer. In *NIPS*, 2018.

– Jangho Kim, SeongUk Park and Nojun Kwak.



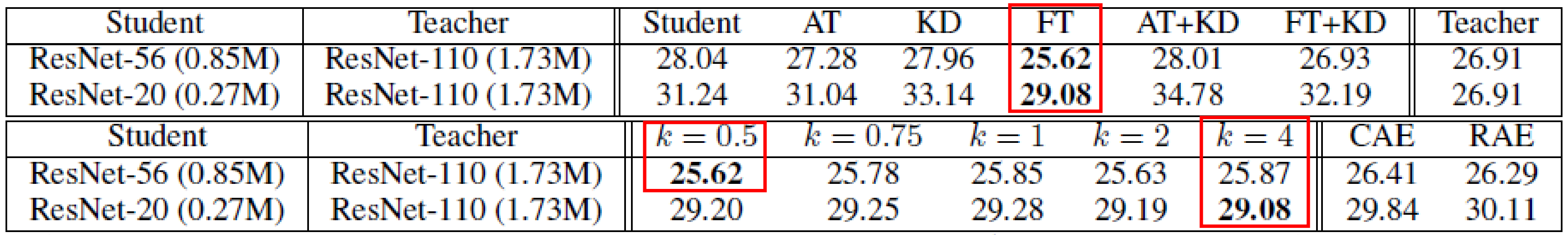
$$\mathcal{L}_{student} = \mathcal{L}_{cls} + \beta\mathcal{L}_{FT}$$

$$\mathcal{L}_{cls} = \mathcal{C}(S(I_x), y) \quad \mathcal{L}_{FT} = \| \frac{F_T}{\|F_T\|_2} - \frac{F_S}{\|F_S\|_2} \|_p$$

<Factor Transfer structure>

Paraphrasing Complex Network: Network Compression via Factor Transfer. In *NIPS*, 2018.

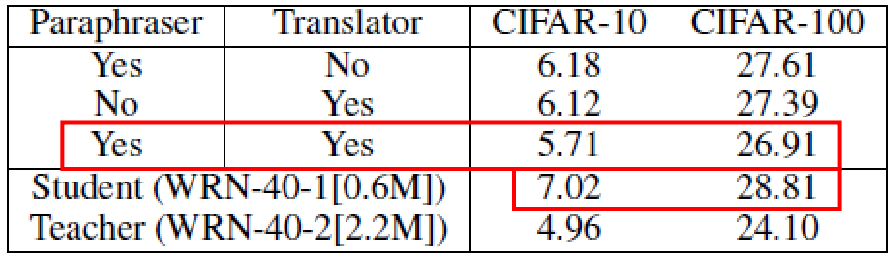
– Jangho Kim, SeongUk Park and Nojun Kwak.



Student	Teacher	Student	AT	KD	FT	AT+KD	FT+KD	Teacher
ResNet-56 (0.85M)	ResNet-110 (1.73M)	28.04	27.28	27.96	**25.62**	28.01	26.93	26.91
ResNet-20 (0.27M)	ResNet-110 (1.73M)	31.24	31.04	33.14	**29.08**	34.78	32.19	26.91

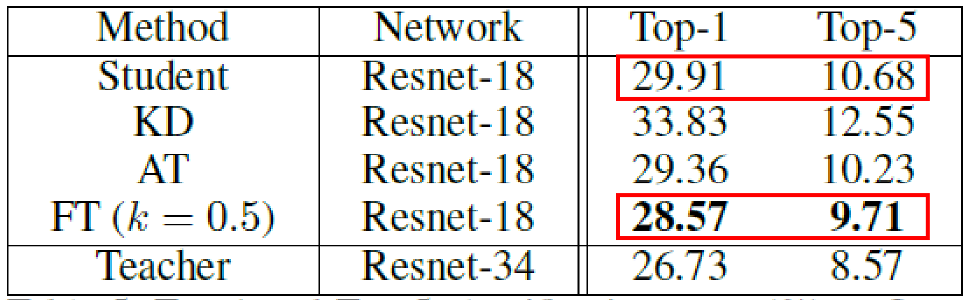
Student	Teacher	$k = 0.5$	$k = 0.75$	$k = 1$	$k = 2$	$k = 4$	CAE	RAE
ResNet-56 (0.85M)	ResNet-110 (1.73M)	**25.62**	25.78	25.85	25.63	25.87	26.41	26.29
ResNet-20 (0.27M)	ResNet-110 (1.73M)	29.20	29.25	29.28	29.19	**29.08**	29.84	30.11

Table 3: Mean classification error (%) on CIFAR-100 dataset (5 runs). All the numbers are from our implementation.



Paraphraser	Translator	CIFAR-10	CIFAR-100
Yes	No	6.18	27.61
No	Yes	6.12	27.39
Yes	Yes	5.71	26.91
Student (WRN-40-1[0.6M])		7.02	28.81
Teacher (WRN-40-2[2.2M])		4.96	24.10

Table 4: Ablation study with and without the paraphraser ($k = 0.5$) and the Translator. (Mean test error (%) of 5 runs).



Method	Network	Top-1	Top-5
Student	Resnet-18	29.91	10.68
KD	Resnet-18	33.83	12.55
AT	Resnet-18	29.36	10.23
FT ($k = 0.5$)	Resnet-18	**28.57**	**9.71**
Teacher	Resnet-34	26.73	8.57

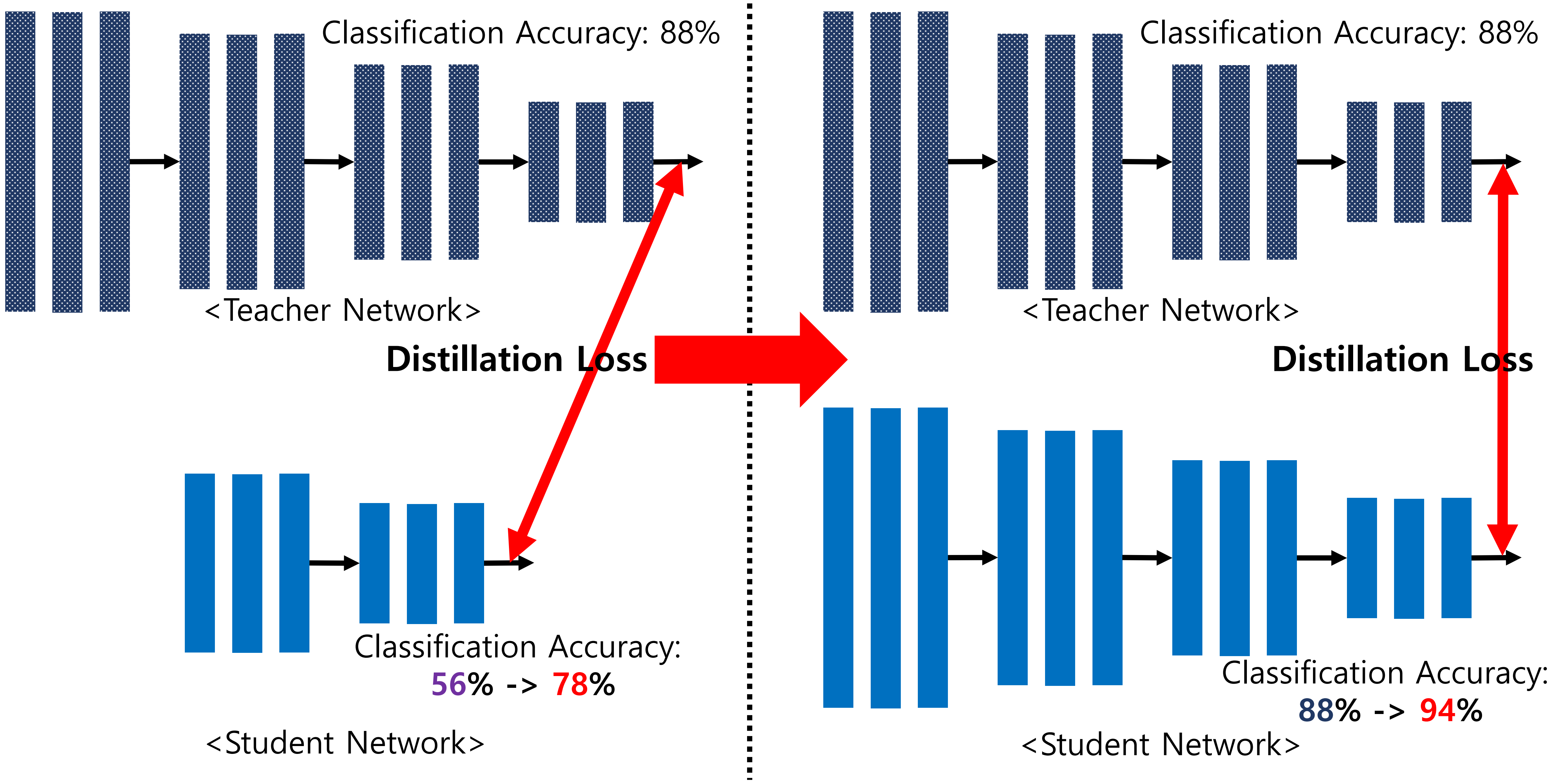
Table 5: Top-1 and Top-5 classification error (%) on ImageNet dataset. All the numbers are from our implementation.

Born-Again Neural Networks

2018
ICML

Born-Again Neural Networks. In *ICML*, 2018.

- Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti and Anima Anandkumar.



Born-Again Neural Networks. In *ICML*, 2018.

- Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti and Anima Anandkumar.

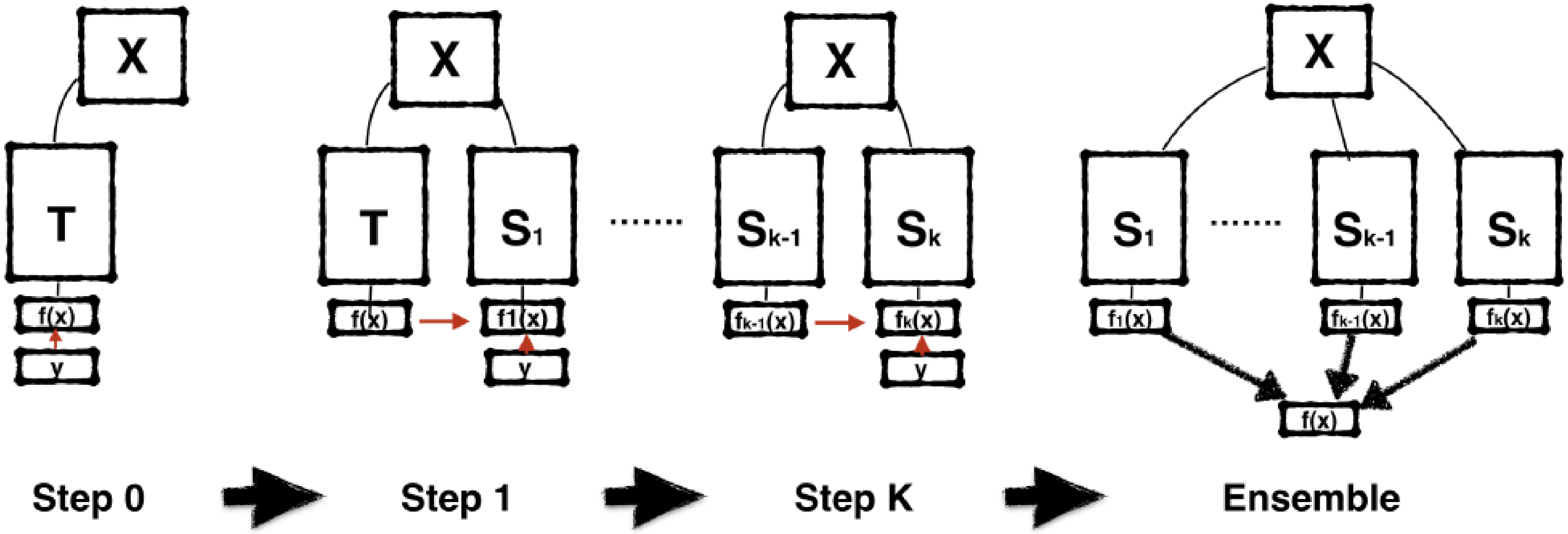


Figure 1. **Graphical representation of the BAN training procedure**: during the first step the teacher model T is trained from the labels Y. Then, at each consecutive step, a new identical model is initialized from a different random seed and trained from the supervision of the earlier generation. At the end of the procedure, additional gains can be achieved with an ensemble of multiple students generations.

$$\mathcal{L}(f(x, \arg\min_{\theta_{k-1}} \mathcal{L}(f(x, \theta_{k-1}))), f(x, \theta_k)) \qquad \hat{f}^k(x) = \sum_{i=1}^{k} f(x, \theta_i)/k$$

Born-Again Neural Networks. In *ICML*, 2018.

- Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti and Anima Anandkumar.

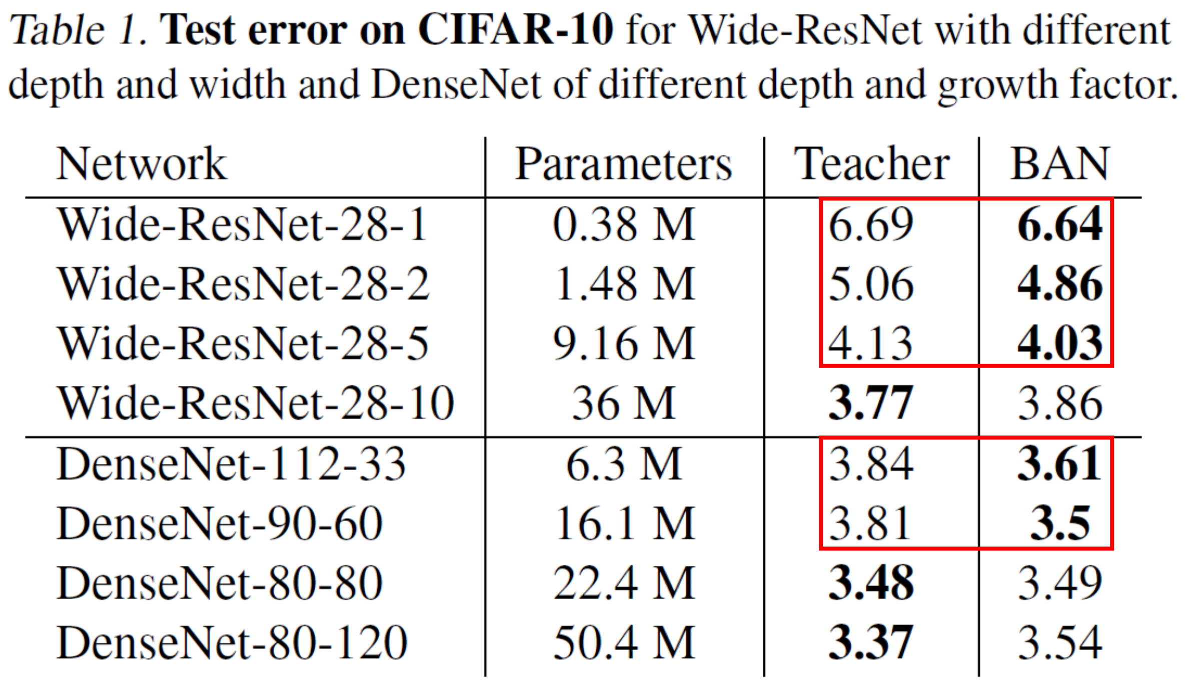


Table 1. **Test error on CIFAR-10** for Wide-ResNet with different depth and width and DenseNet of different depth and growth factor.

Network	Parameters	Teacher	BAN
Wide-ResNet-28-1	0.38 M	6.69	**6.64**
Wide-ResNet-28-2	1.48 M	5.06	**4.86**
Wide-ResNet-28-5	9.16 M	4.13	**4.03**
Wide-ResNet-28-10	36 M	**3.77**	3.86
DenseNet-112-33	6.3 M	3.84	**3.61**
DenseNet-90-60	16.1 M	3.81	**3.5**
DenseNet-80-80	22.4 M	**3.48**	3.49
DenseNet-80-120	50.4 M	**3.37**	3.54

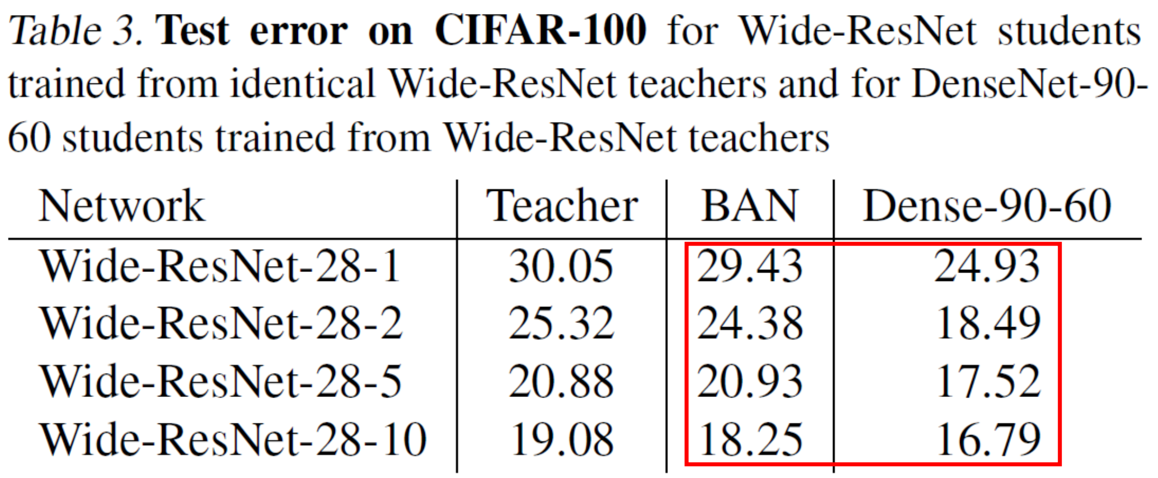


Table 3. **Test error on CIFAR-100** for Wide-ResNet students trained from identical Wide-ResNet teachers and for DenseNet-90-60 students trained from Wide-ResNet teachers

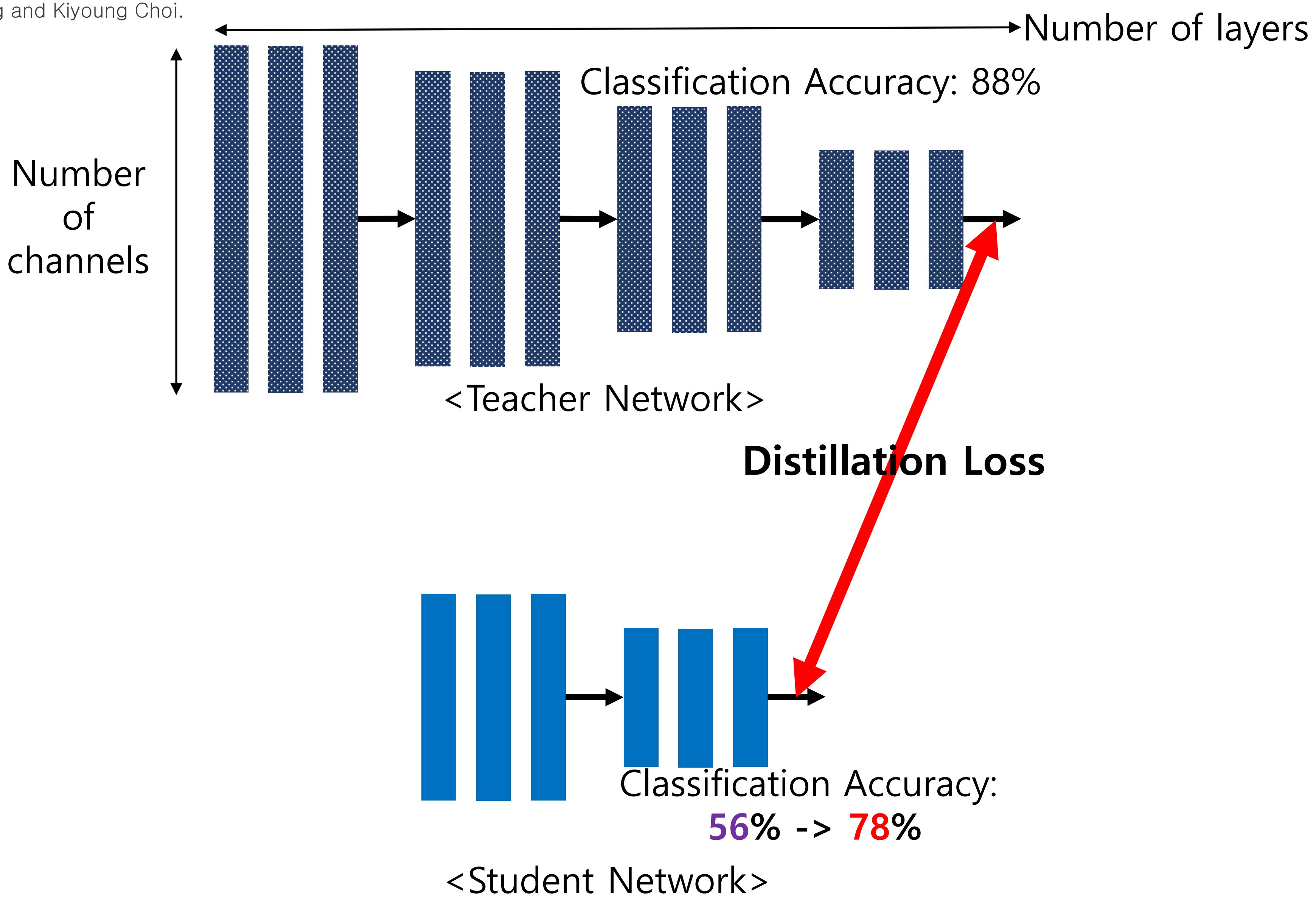
Network	Teacher	BAN	Dense-90-60
Wide-ResNet-28-1	30.05	29.43	24.93
Wide-ResNet-28-2	25.32	24.38	18.49
Wide-ResNet-28-5	20.88	20.93	17.52
Wide-ResNet-28-10	19.08	18.25	16.79

Network Recasting: A Universal Method for Network Architecture Transformation

2019
AAAI

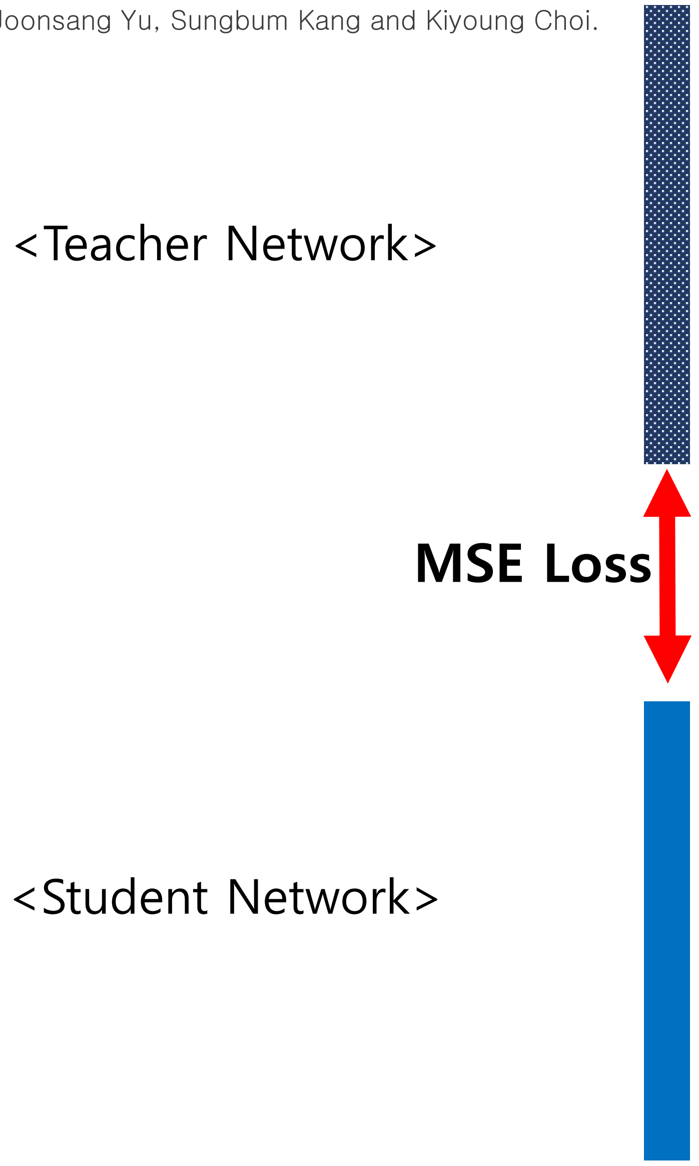
Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

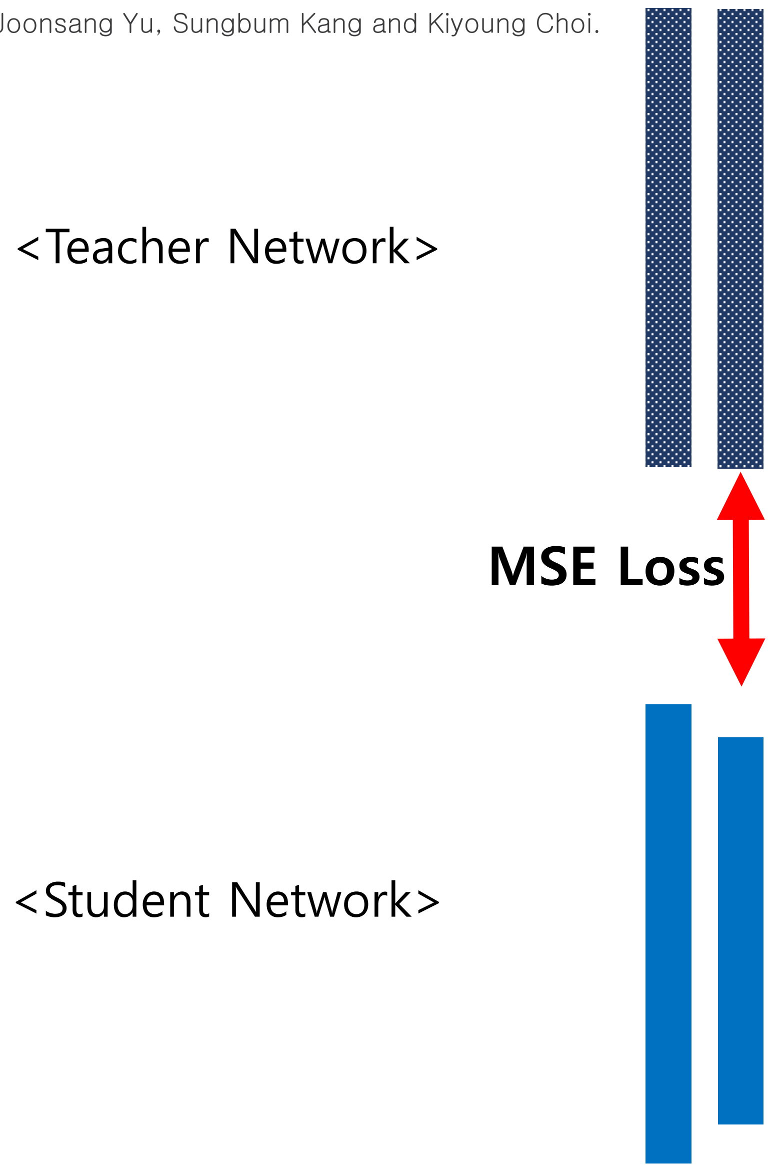
– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

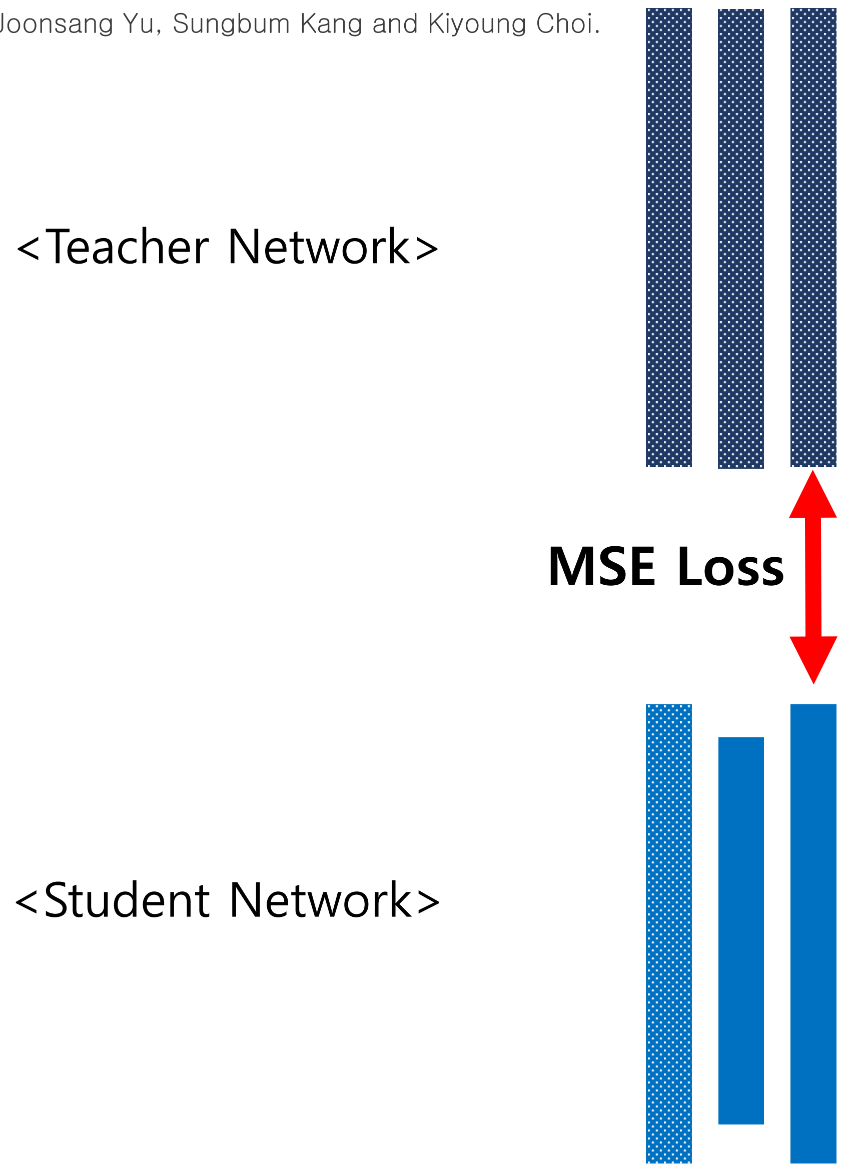
– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

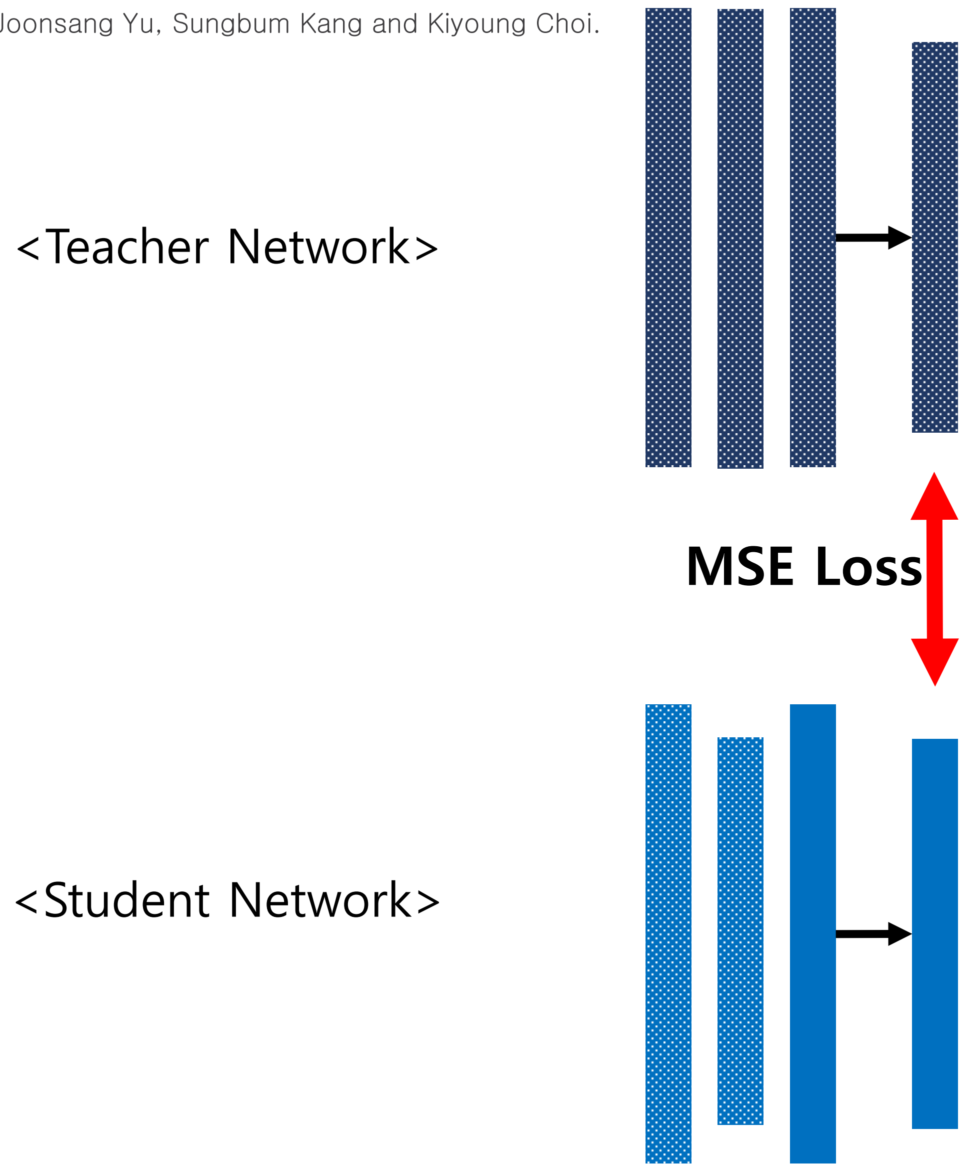
– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

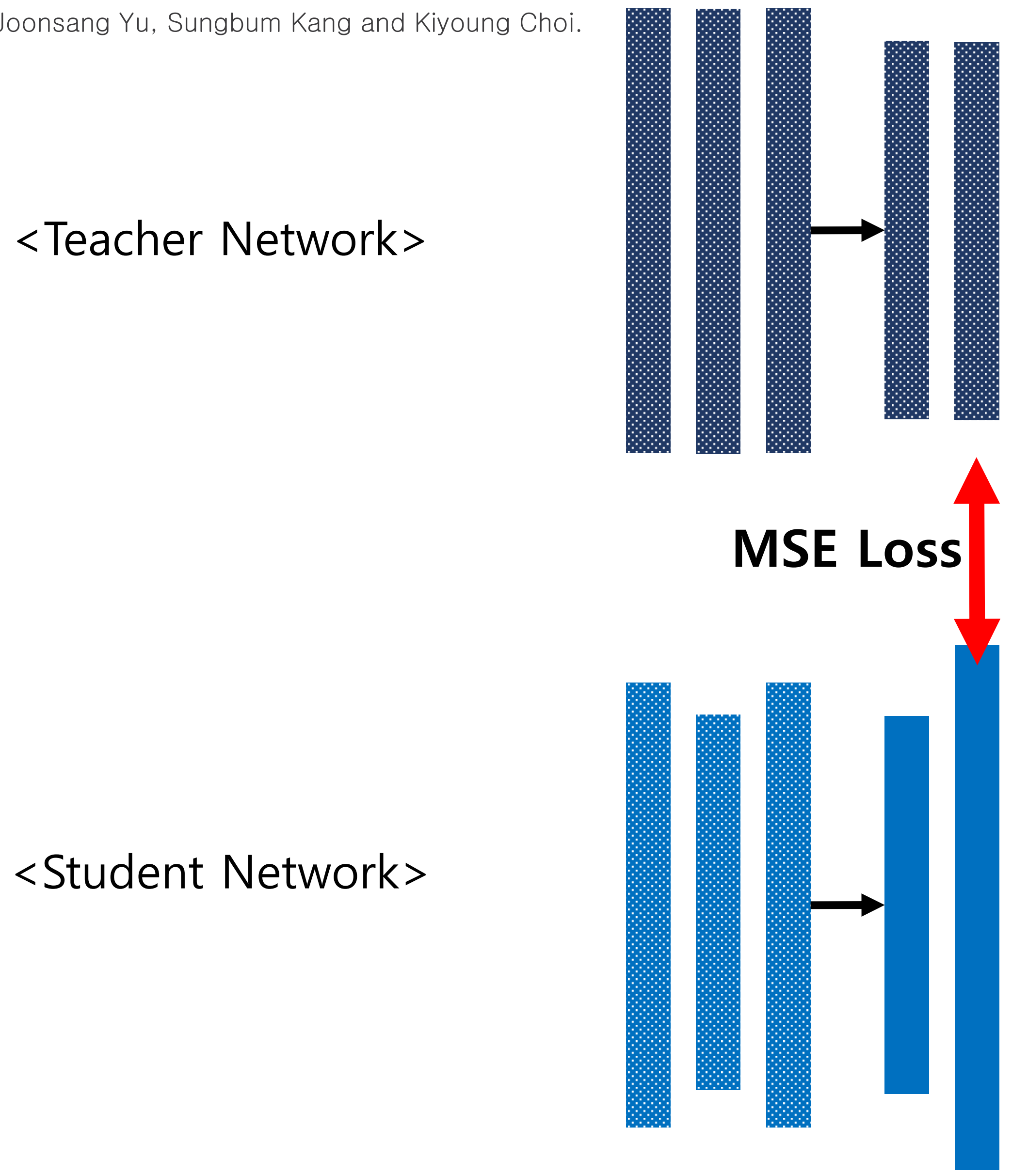
– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

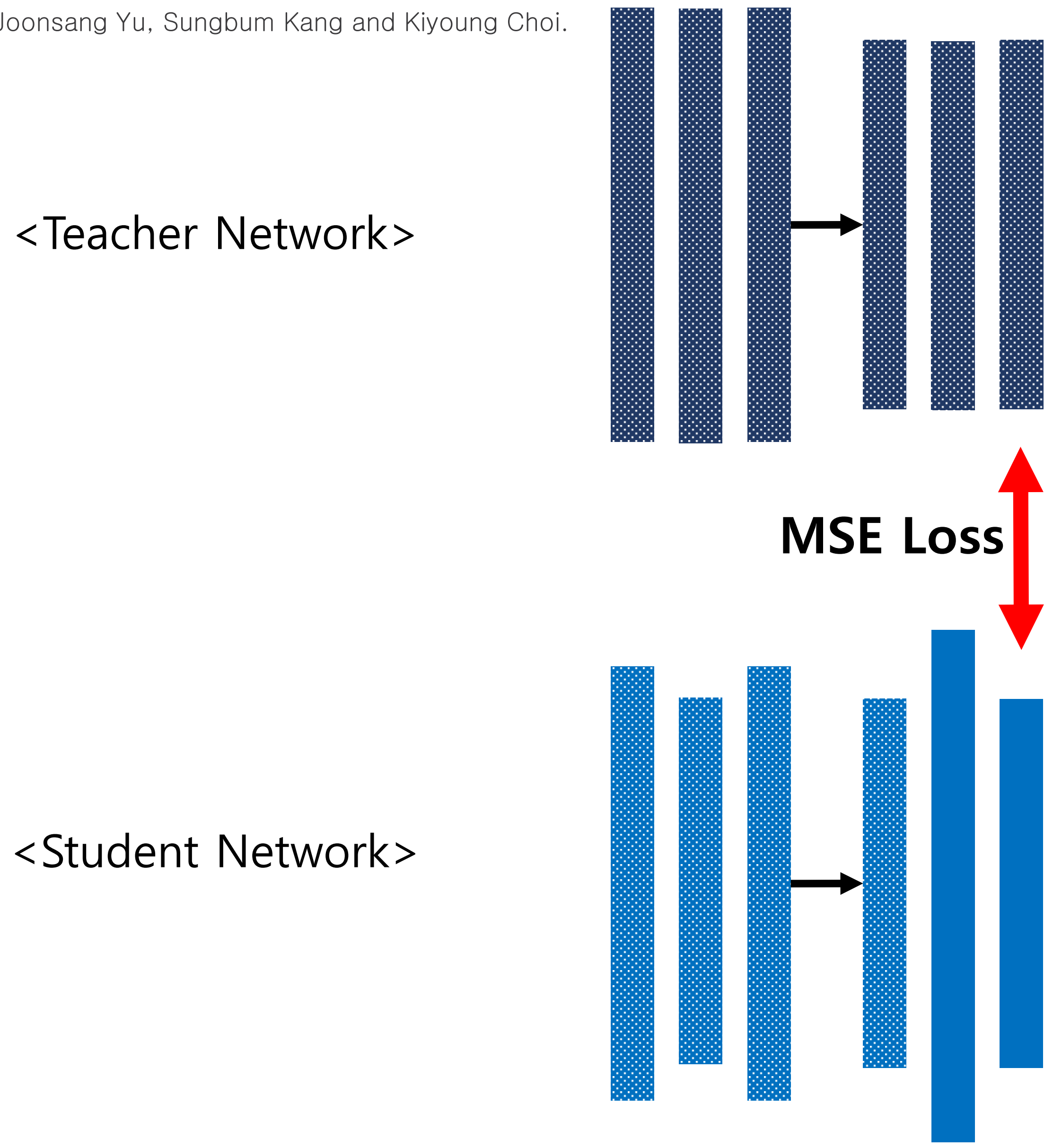
– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

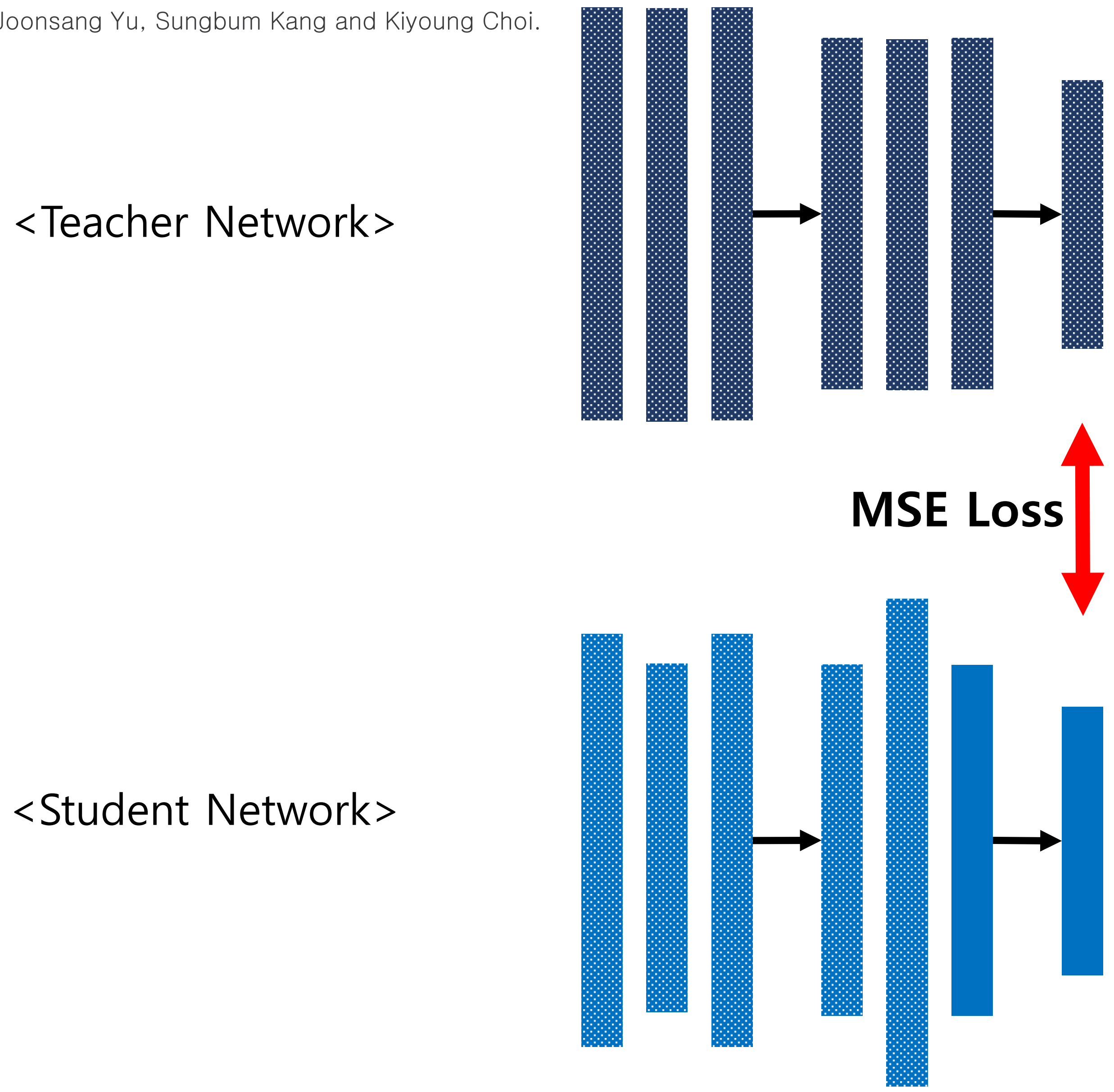
– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

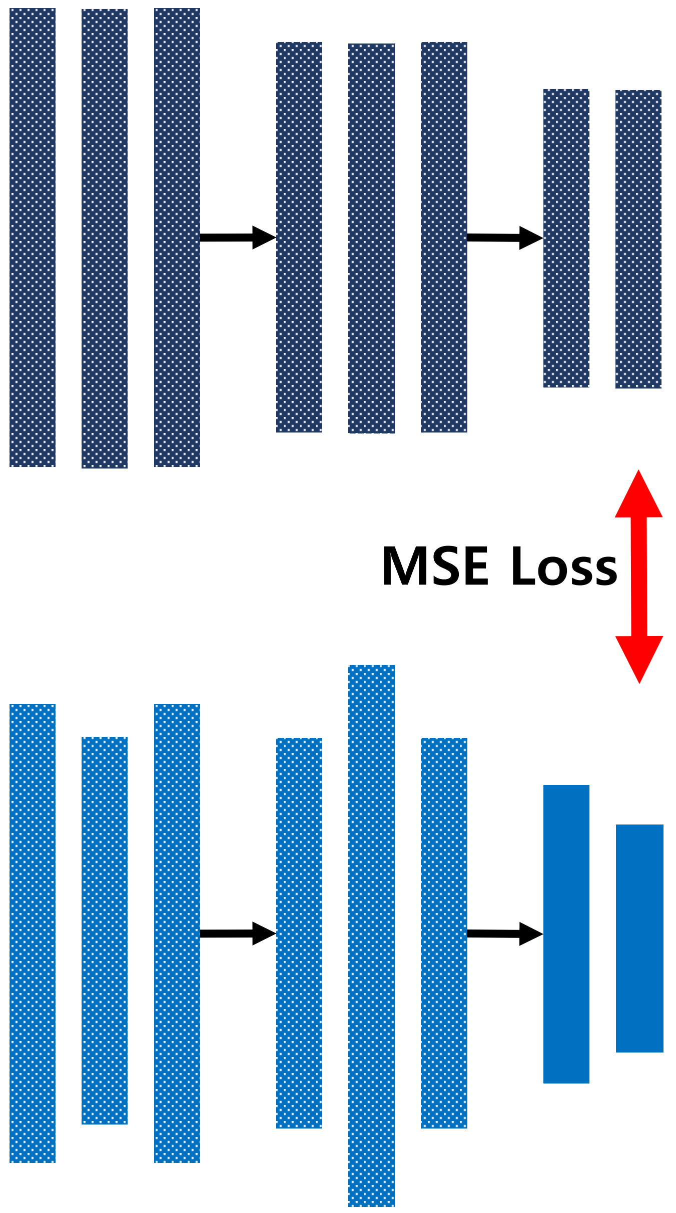
– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



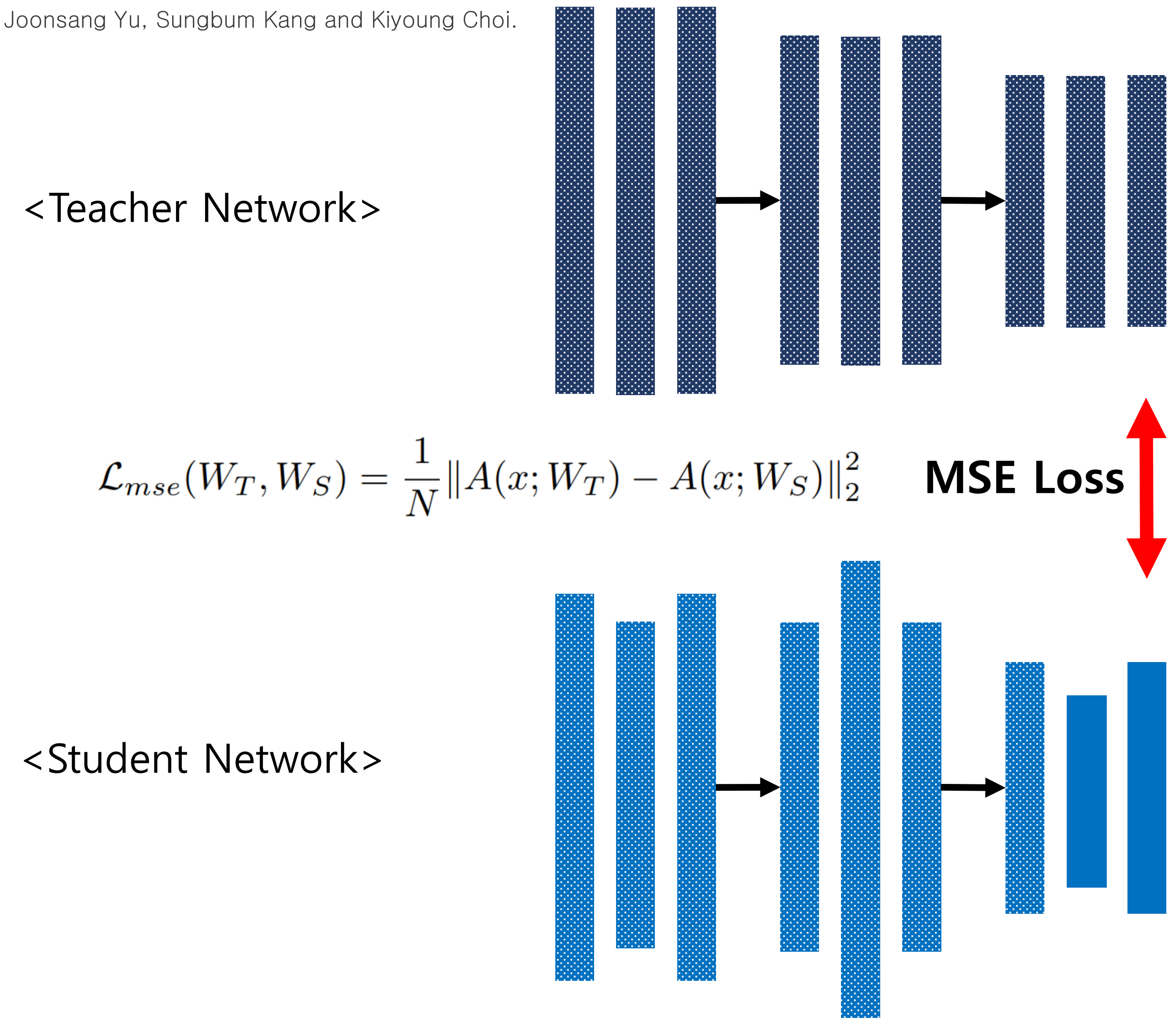
<Teacher Network>

<Student Network>

$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

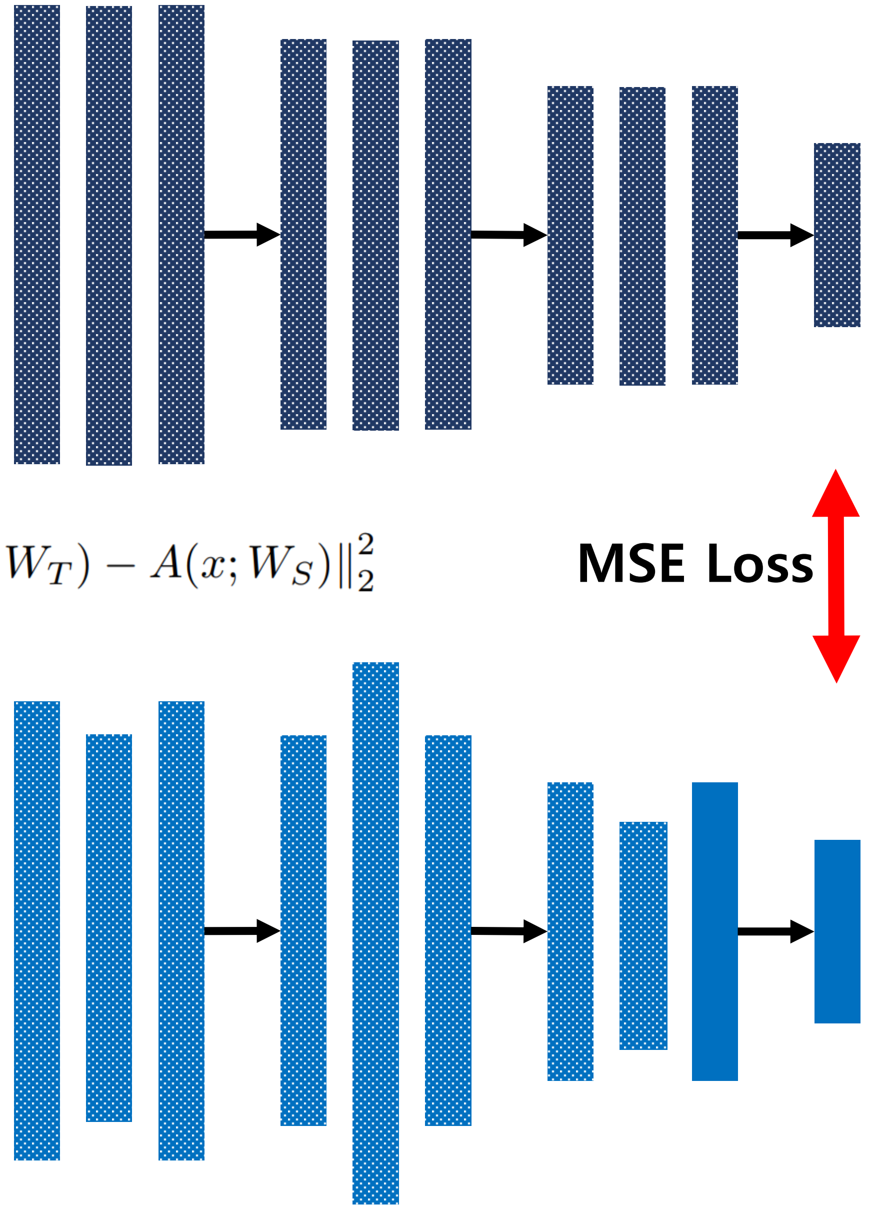
Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

– Joonsang Yu, Sungbum Kang and Kiyoung Choi.

<Teacher Network>

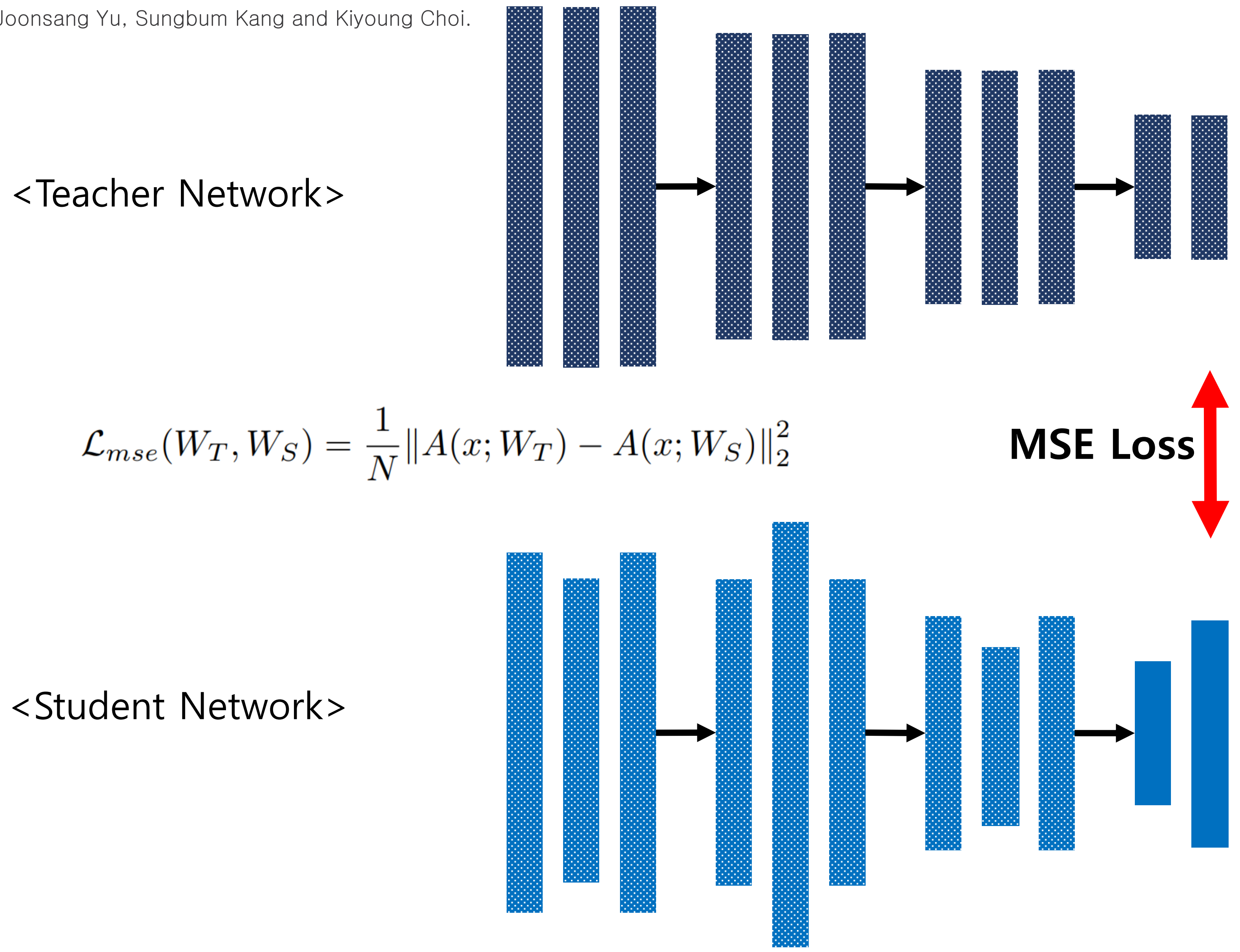
$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

<Student Network>



Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

– Joonsang Yu, Sungbum Kang and Kiyoung Choi.

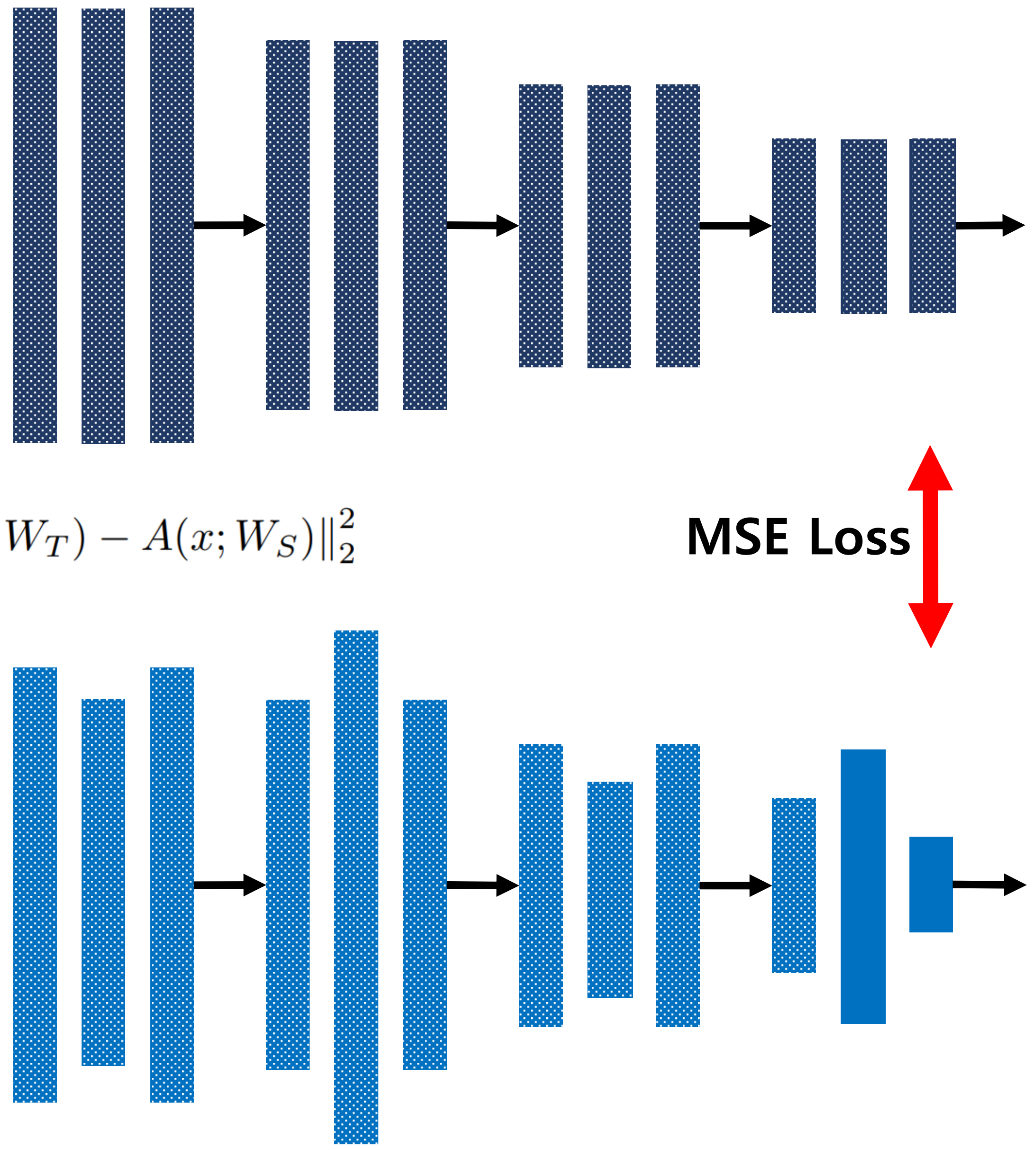


$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

– Joonsang Yu, Sungbum Kang and Kiyoung Choi.

<Teacher Network>



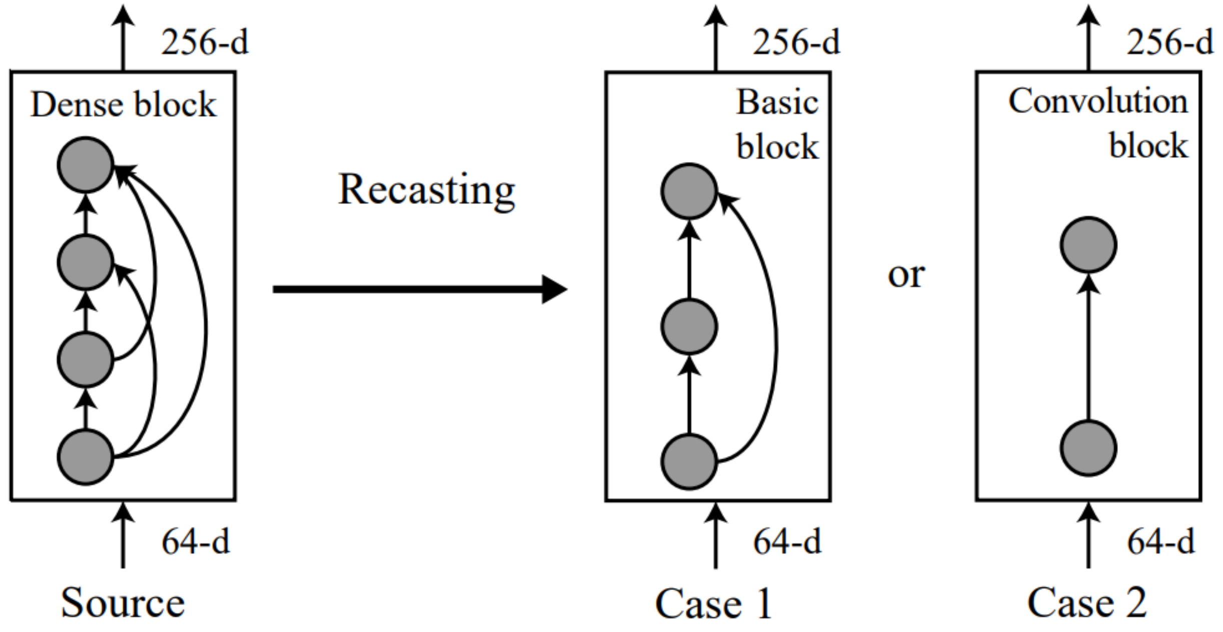
$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$

<Student Network>

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

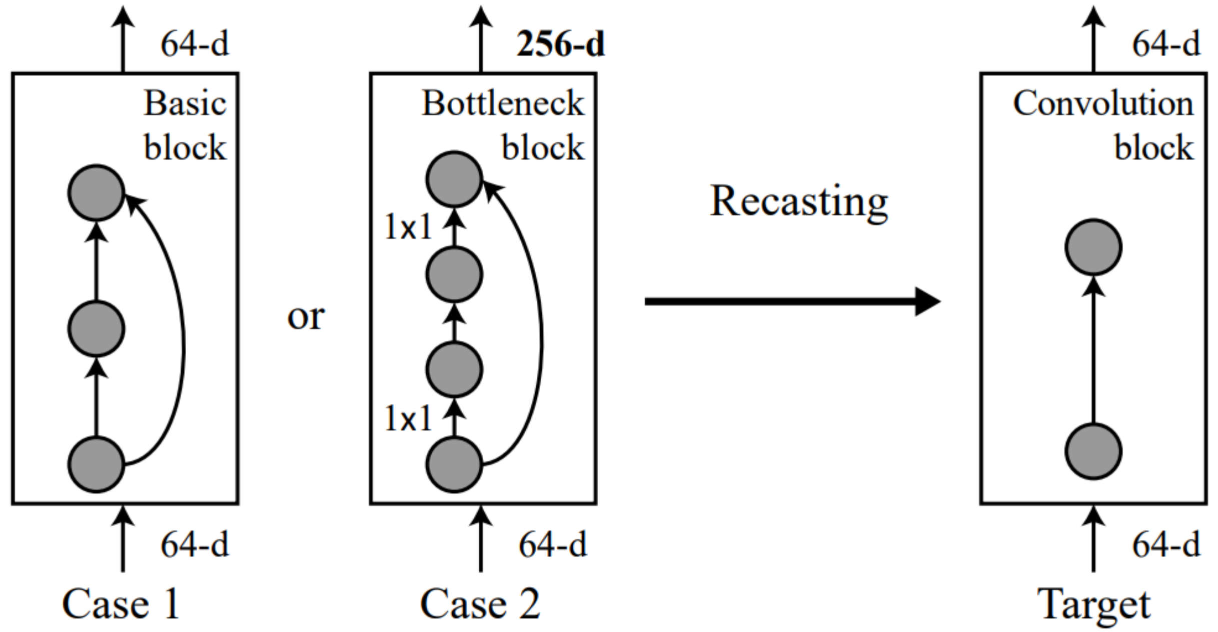
– Joonsang Yu, Sungbum Kang and Kiyoung Choi.

$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$



<Block recasting case1>

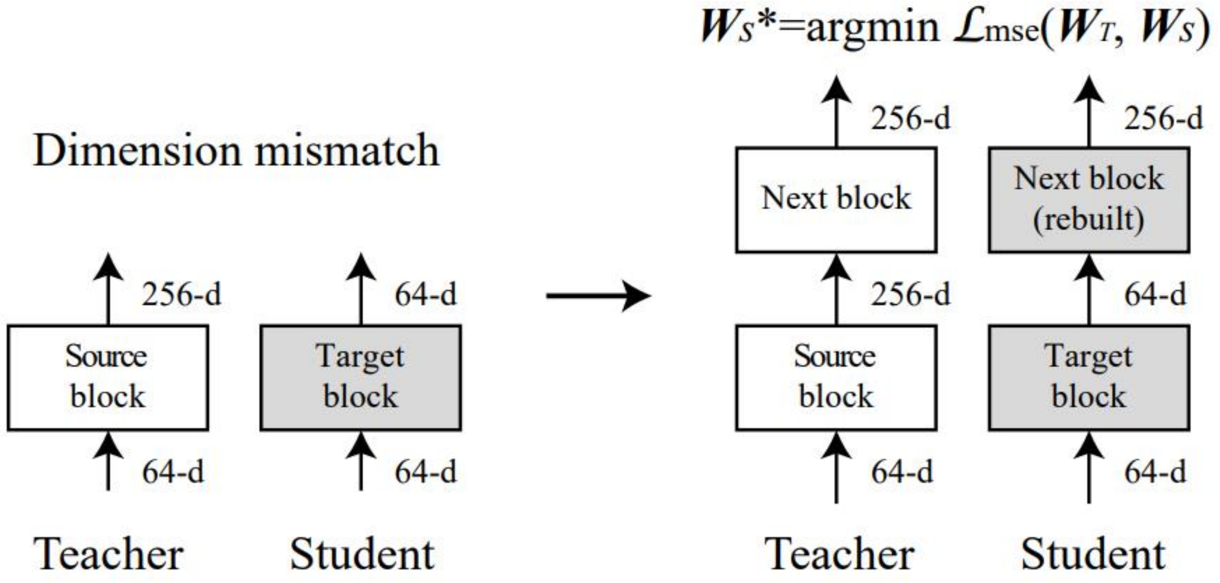
$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2$$



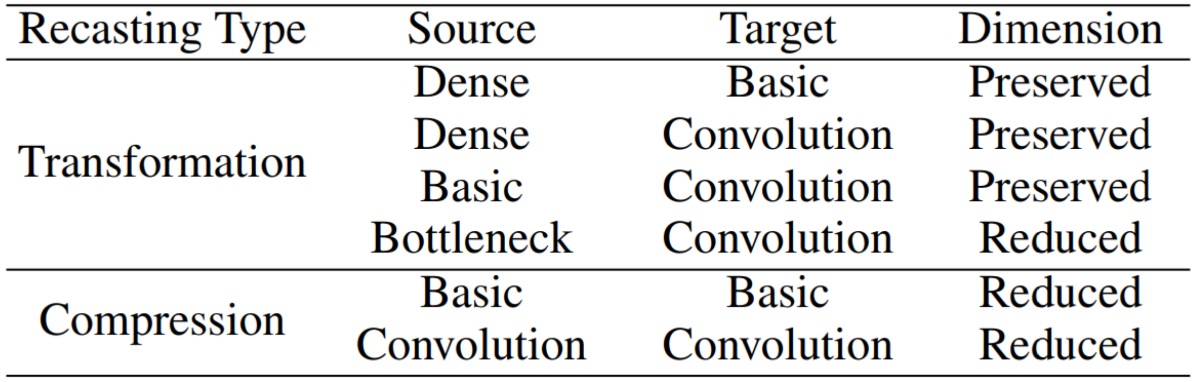
<Block recasting case2>

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



<Method for dimension mismatch>

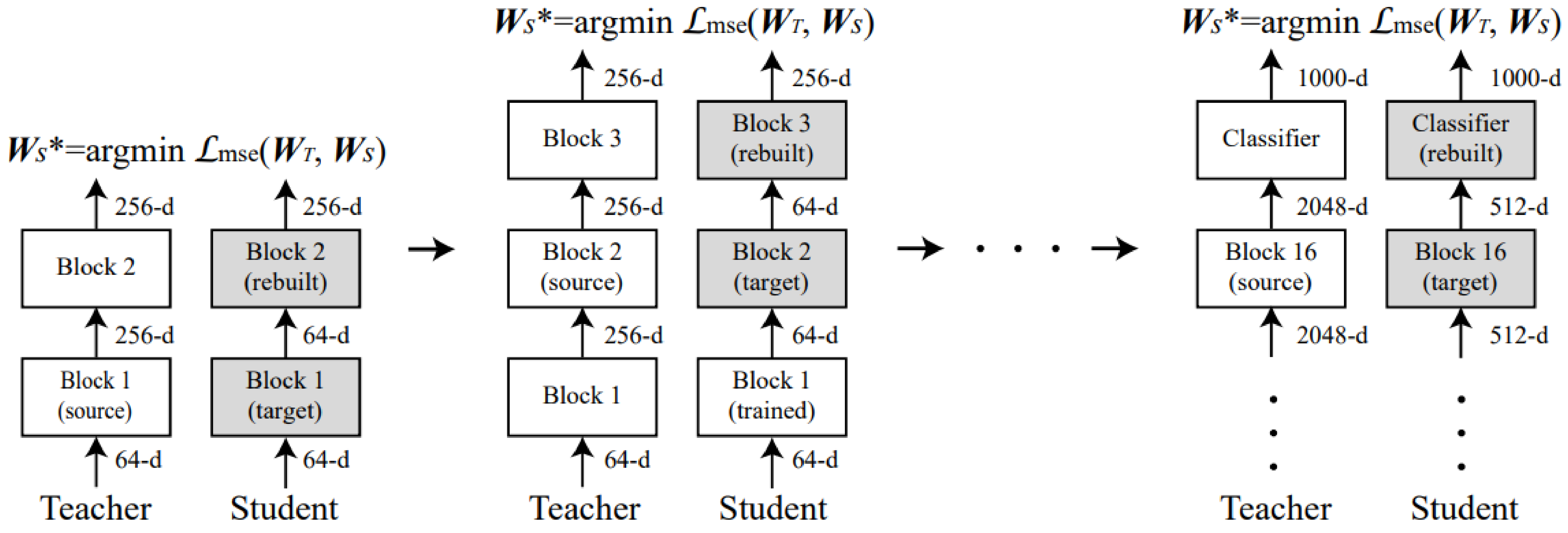


Recasting Type	Source	Target	Dimension
Transformation	Dense	Basic	Preserved
	Dense	Convolution	Preserved
	Basic	Convolution	Preserved
	Bottleneck	Convolution	Reduced
Compression	Basic	Basic	Reduced
	Convolution	Convolution	Reduced

<Recasting Methods>

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

– Joonsang Yu, Sungbum Kang and Kiyoung Choi.



$$\mathcal{L}_{mse}(W_T, W_S) = \frac{1}{N}\|A(x;W_T) - A(x;W_S)\|_2^2 \qquad \mathcal{L}_{kd}(W_T, W_S) = \mathcal{L}_{mse\_logit}(W_T, W_S) + \mathcal{L}_{ce}(y_{true}, W_S)$$

<Network recasting structure>

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

– Joonsang Yu, Sungbum Kang and Kiyoung Choi.

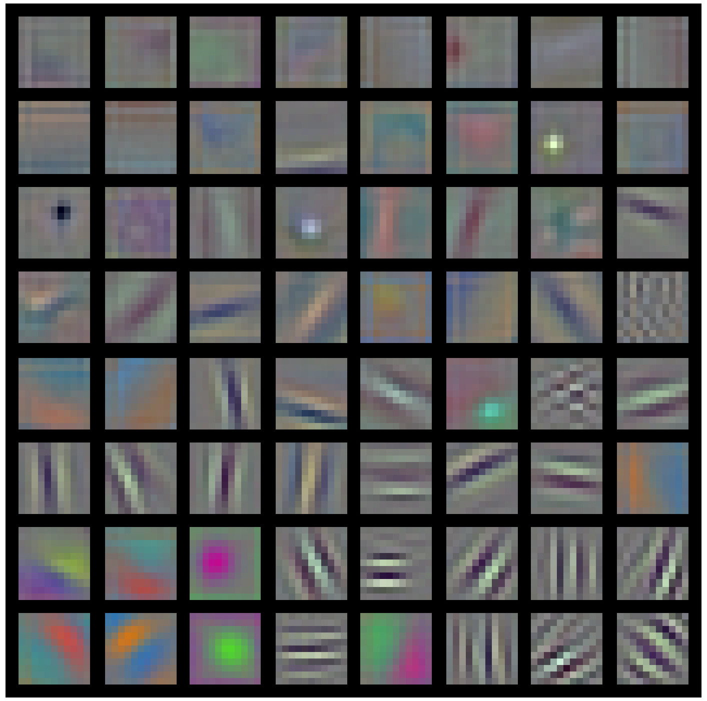
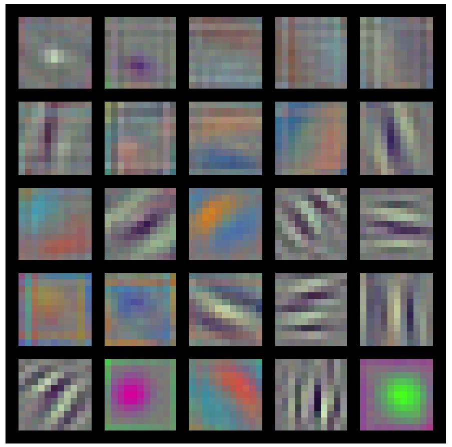
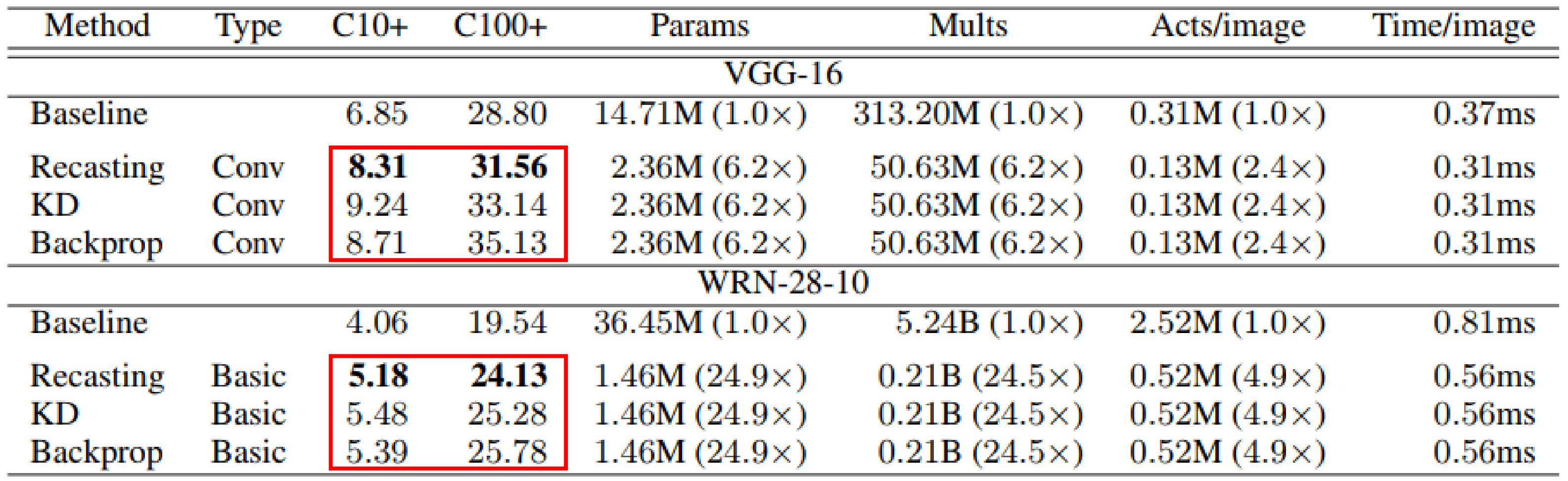



Figure 6: Visualization of filters in the first layer of AlexNet (*left*) and a student network (*right*). Redundant filters are removed after network recasting.

Network Recasting: A Universal Method for Network Architecture Transformation. In *AAAI*, 2019.

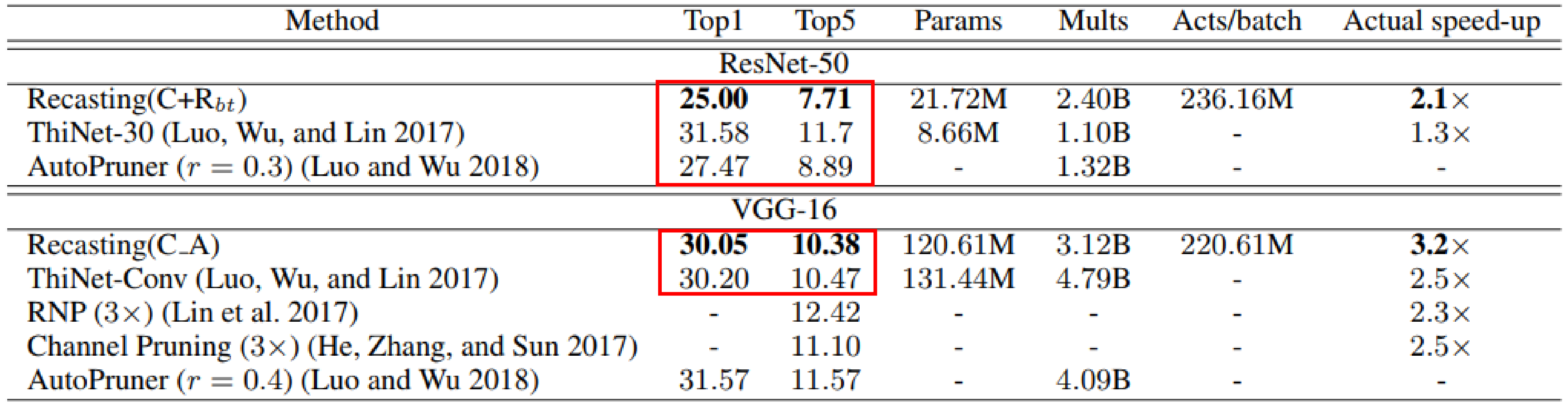
– Joonsang Yu, Sungbum Kang and Kiyoung Choi.

Table 3: Error rates (%) of compression results on CIFAR datasets. (B/M: billion/million)



Method	Type	C10+	C100+	Params	Mults	Acts/image	Time/image
				VGG-16			
Baseline		6.85	28.80	14.71M (1.0×)	313.20M (1.0×)	0.31M (1.0×)	0.37ms
Recasting	Conv	**8.31**	**31.56**	2.36M (6.2×)	50.63M (6.2×)	0.13M (2.4×)	0.31ms
KD	Conv	9.24	33.14	2.36M (6.2×)	50.63M (6.2×)	0.13M (2.4×)	0.31ms
Backprop	Conv	8.71	35.13	2.36M (6.2×)	50.63M (6.2×)	0.13M (2.4×)	0.31ms
				WRN-28-10			
Baseline		4.06	19.54	36.45M (1.0×)	5.24B (1.0×)	2.52M (1.0×)	0.81ms
Recasting	Basic	**5.18**	**24.13**	1.46M (24.9×)	0.21B (24.5×)	0.52M (4.9×)	0.56ms
KD	Basic	5.48	25.28	1.46M (24.9×)	0.21B (24.5×)	0.52M (4.9×)	0.56ms
Backprop	Basic	5.39	25.78	1.46M (24.9×)	0.21B (24.5×)	0.52M (4.9×)	0.56ms

Table 5: Comparison of error rate (%) with previous works on ILSVRC2012. (B/M: billion/million)



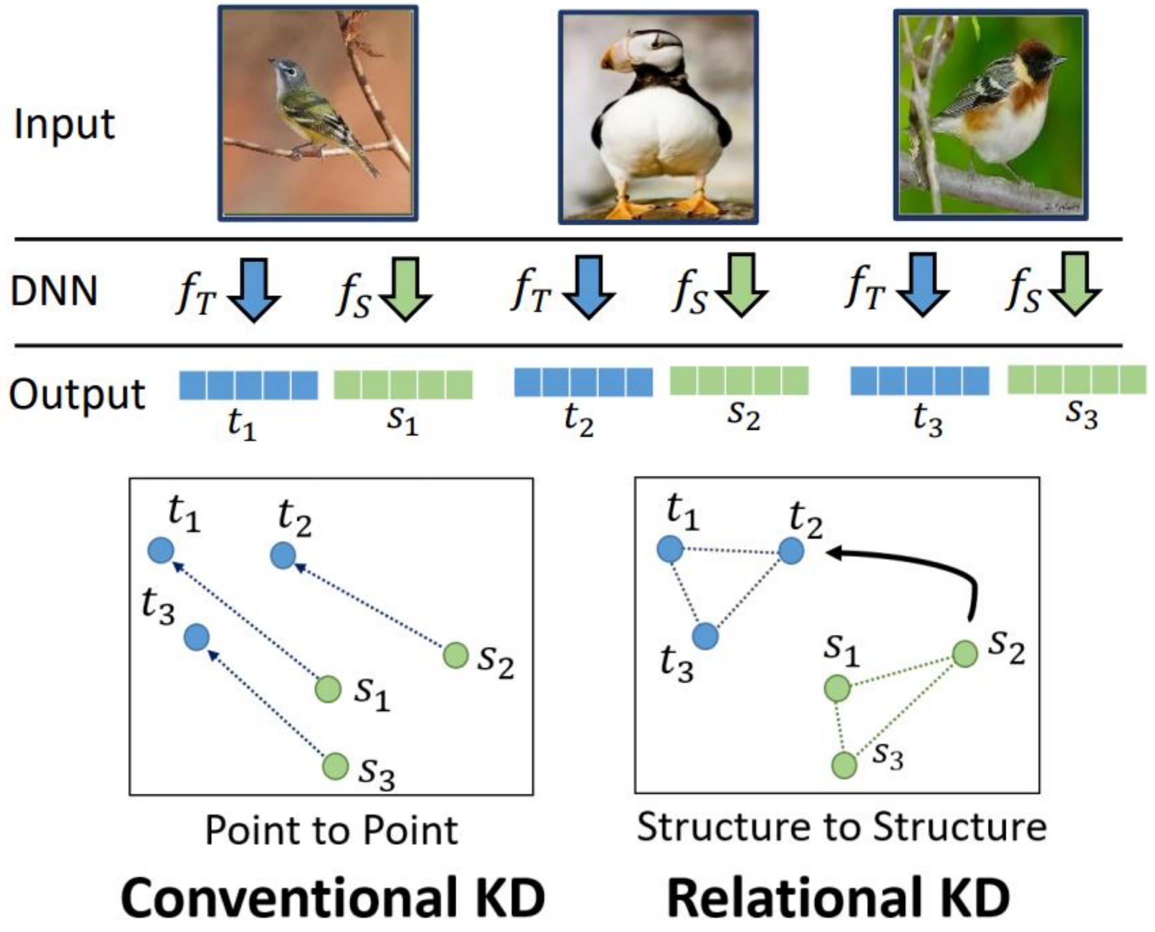
Method	Top1	Top5	Params	Mults	Acts/batch	Actual speed-up
		ResNet-50				
Recasting(C+R_{bt})	**25.00**	**7.71**	21.72M	2.40B	236.16M	**2.1**×
ThiNet-30 (Luo, Wu, and Lin 2017)	31.58	11.7	8.66M	1.10B	-	1.3×
AutoPruner ($r = 0.3$) (Luo and Wu 2018)	27.47	8.89	-	1.32B	-	-
		VGG-16				
Recasting(C_A)	**30.05**	**10.38**	120.61M	3.12B	220.61M	**3.2**×
ThiNet-Conv (Luo, Wu, and Lin 2017)	30.20	10.47	131.44M	4.79B	-	2.5×
RNP (3×) (Lin et al. 2017)	-	12.42	-	-	-	2.3×
Channel Pruning (3×) (He, Zhang, and Sun 2017)	-	11.10	-	-	-	2.5×
AutoPruner ($r = 0.4$) (Luo and Wu 2018)	31.57	11.57	-	4.09B	-	-

Relational Knowledge Distillation

2019
CVPR

Relational Knowledge Distillation. In *CVPR*, 2019.

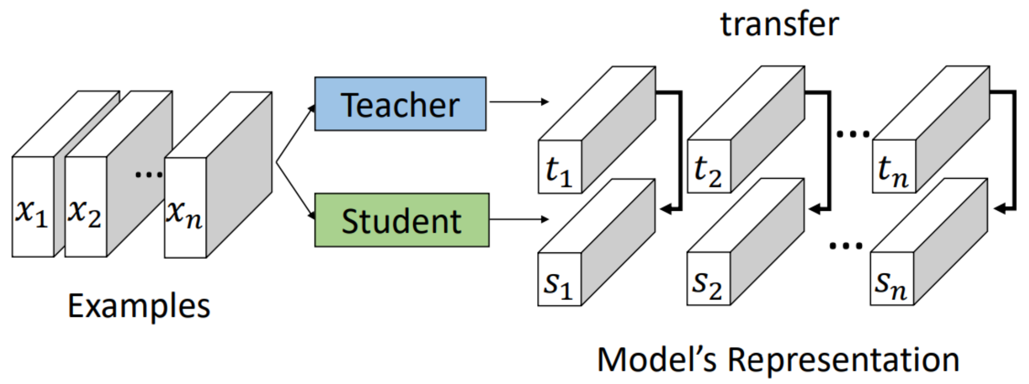
– Wonpyo Park, Dongju Kim, Yan Lu and Minsu Cho.



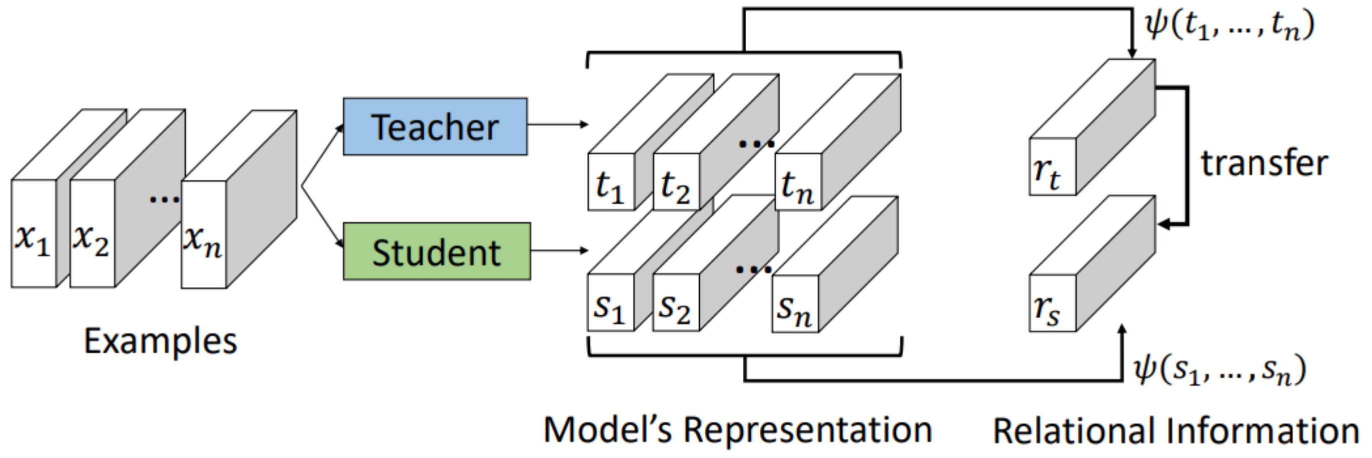
<Difference between KD and RKD>

Relational Knowledge Distillation. In *CVPR*, 2019.

– Wonpyo Park, Dongju Kim, Yan Lu and Minsu Cho.



Individual Knowledge Distillation



Relational Knowledge Distillation

$$\psi_{\mathrm{D}}(t_i, t_j) = \frac{1}{\mu} \|t_i - t_j\|_2$$

$$\mu = \frac{1}{|\mathcal{X}^2|} \sum_{(x_i, x_j) \in \mathcal{X}^2} \|t_i - t_j\|_2$$

$$\mathcal{L}_{\text{RKD-D}} = \sum_{(x_i, x_j) \in \mathcal{X}^2} l_\delta\big(\psi_{\mathrm{D}}(t_i, t_j), \psi_{\mathrm{D}}(s_i, s_j)\big)$$

<Distance-wise distillation loss>

$$\psi_{\mathrm{A}}(t_i, t_j, t_k) = \cos \angle t_i t_j t_k = \langle \mathbf{e}^{ij}, \mathbf{e}^{kj} \rangle$$

$$\text{where} \quad \mathbf{e}^{ij} = \frac{t_i - t_j}{\|t_i - t_j\|_2}, \mathbf{e}^{kj} = \frac{t_k - t_j}{\|t_k - t_j\|_2}$$

$$l_\delta(x, y) = \begin{cases} \frac{1}{2}(x - y)^2 & \text{for } |x - y| \le 1 \\ |x - y| - \frac{1}{2}, & \text{otherwise.} \end{cases}$$

$$\mathcal{L}_{\text{RKD-A}} = \sum_{(x_i, x_j, x_k) \in \mathcal{X}^3} l_\delta\big(\psi_{\mathrm{A}}(t_i, t_j, t_k), \psi_{\mathrm{A}}(s_i, s_j, s_k)\big)$$

<Angle-wise distillation loss>

Relational Knowledge Distillation. In *CVPR*, 2019.

– Wonpyo Park, Dongju Kim, Yan Lu and Minsu Cho.

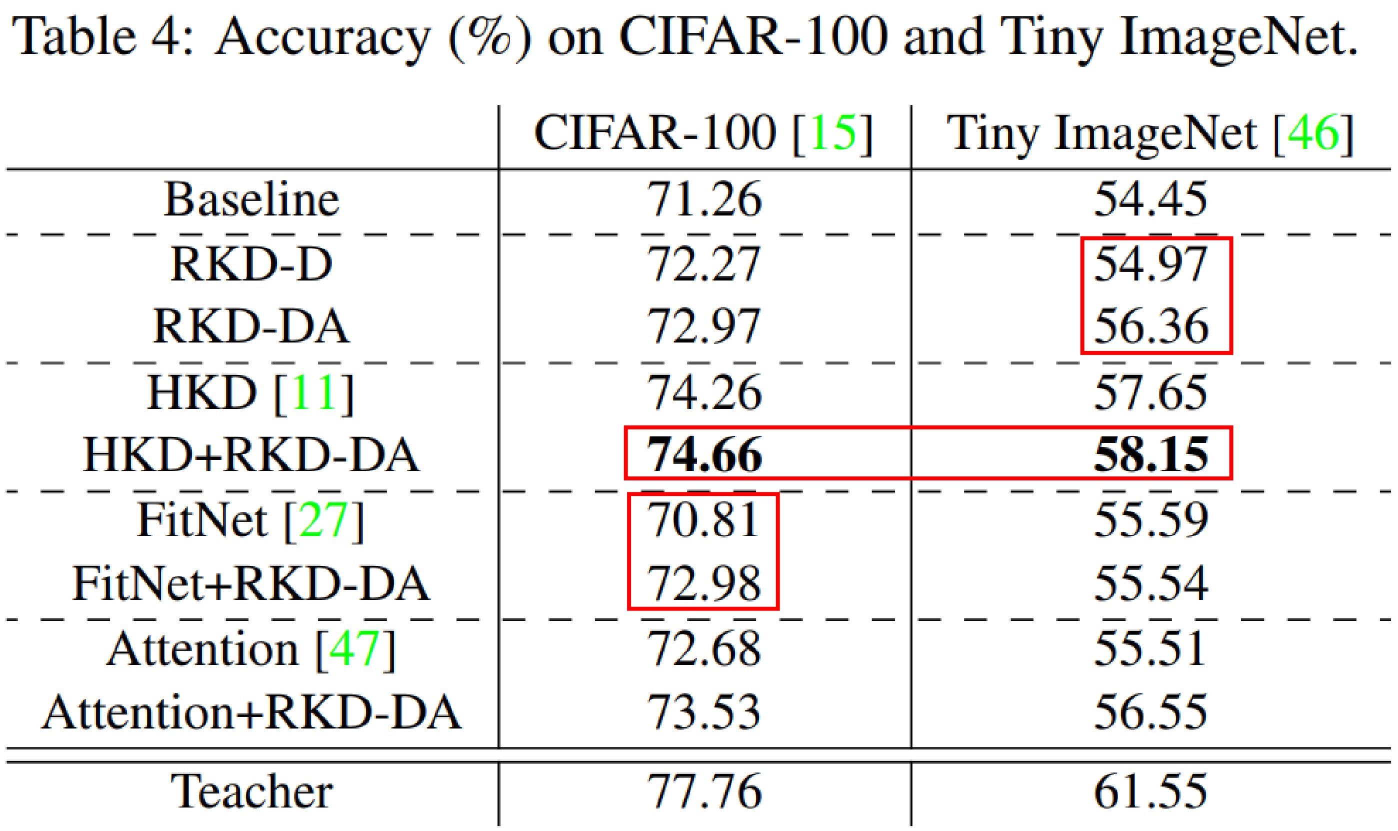


Table 4: Accuracy (%) on CIFAR-100 and Tiny ImageNet.

	CIFAR-100 [15]	Tiny ImageNet [46]
Baseline	71.26	54.45
RKD-D	72.27	54.97
RKD-DA	72.97	56.36
HKD [11]	74.26	57.65
HKD+RKD-DA	**74.66**	**58.15**
FitNet [27]	70.81	55.59
FitNet+RKD-DA	72.98	55.54
Attention [47]	72.68	55.51
Attention+RKD-DA	73.53	56.55
Teacher	77.76	61.55

SUMMARY

- Regularizer

- Teaching & Learning

- Various Fields
  (Model Compression, Transfer Learning, Few shot Learning, Meta Learning…)

Q&A