Lecture 06:
Distillation, Low Rank Decomposition and Neural Architecture Search

Recap

- Basic Data Formats
  - Fixed point (INT)
  - Floating point (FP)
  - Block floating point (BFP)
- Quantization methods
  - Taxonomy of Quantization
  - Learnable adaptive quantization scheme
  - Quantization for LLM

NYU SAI LAB

Topics

- Learnable adaptive quantization scheme
- Distillation
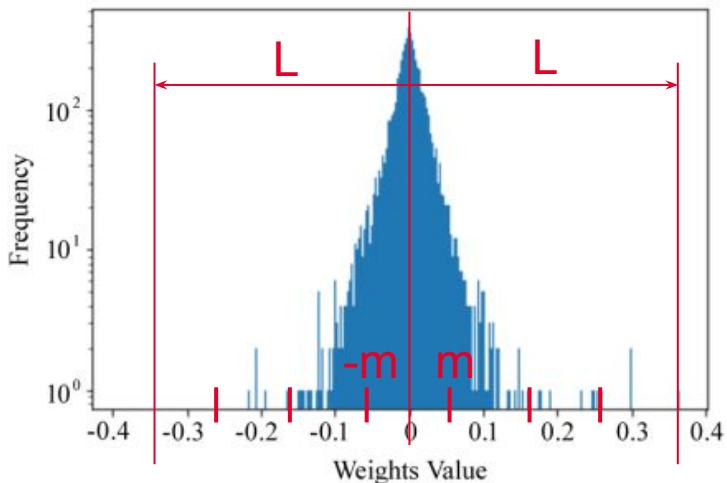- Neural architecture search (NAS)
- Low-rank factorization

Learnable Quantization

- How to convert a number x to INT representation?
  - Set the clipping range: (-L, L), bitwidth: b
  - Compute the scale: $s = 2L/(2^b - 2)$
  - Clip the input x: $x_c = Clip(x, L, -L)$
  - Calculate the INT representation: $x_{int} = round(x_c/s)$
  - Rescale: $x_q = s x_{int}$

- Learnable clipping function: L
- Learnable rounding function: round(.)
- Learnable quantization function: Q(.)

Learnable Quantization

- Various approaches have been suggested for learning the quantization hyperparameters:
    - PACT: learnable clipping range
    - QIL: learnable rounding function
    - QN: learning quantization step

- Choi, Jungwook, et al. "Pact: Parameterized clipping activation for quantized neural networks." *arXiv preprint arXiv:1805.06085* (2018).
- Jung, Sangil, et al. "Learning to quantize deep networks by optimizing quantization intervals with task loss." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.
- Yang, Jiwei, et al. "Quantization networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.
- Gong, Ruihao, et al. "Differentiable soft quantization: Bridging full-precision and low-bit neural networks." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.
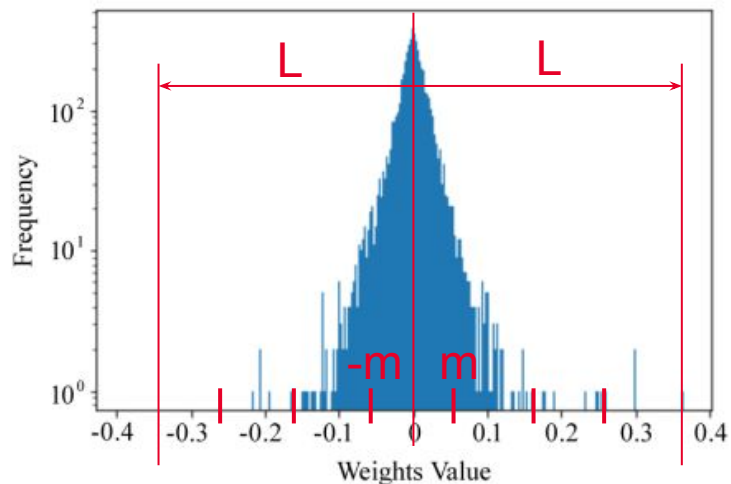
NYU SAI LAB

Learnable Quantization (PACT)

Weight distribution in ResNet

- How to convert a number to INT8 representation?
  - Set the clipping range: $(-l, l)$, bitwidth: b
  - Compute the scale: $s = (2l)/(2^b-1)$
  - Clip the input x: $x_c = Clip(x, l, -l)$
  - Calculate the fixed-point representation:
    $$x_{int} = round(x_c/s)$$
  - Rescale: $x_q = sx_{int}$

$l = 0.9 \times max(|W|), l = 0.95 \times max(|W|)$

Can learn by learnt during training?

Choi, Jungwook, et al. "Pact: Parameterized clipping activation for quantized neural networks." *arXiv preprint arXiv:1805.06085* (2018).
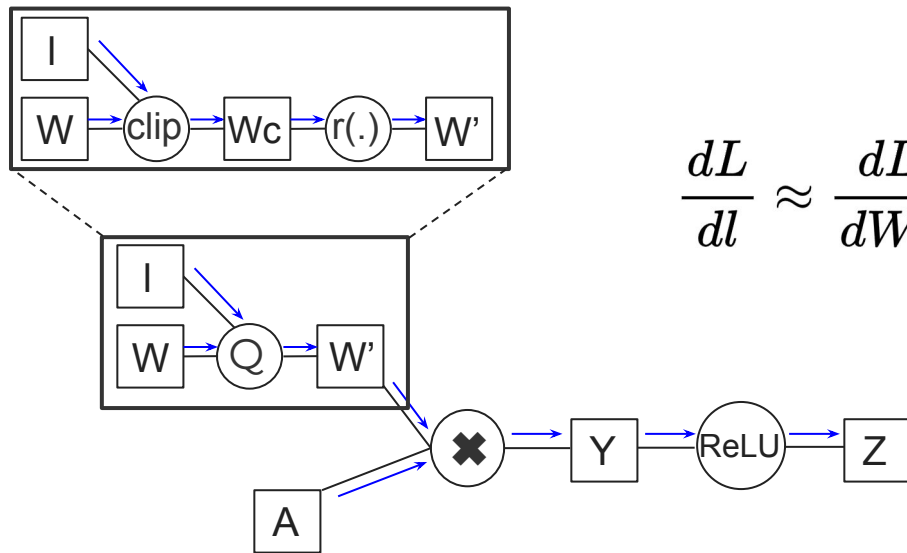
NYU SAI LAB

Learnable Quantization (PACT)

- First we need to apply CLIP function to the input x, where the clip function has a range of (-l, l).

-

$$x_c = Clip(x, l) = \begin{cases} l, & \text{if } x \geq l \\ x, & -l \leq x \leq l \\ -l, & x \leq l \end{cases}$$

$$x_q = round(\frac{x_c}{s}) \times s$$

- Can we learn l? $\dfrac{dL}{dl} = \dfrac{dL}{dx_q} \dfrac{dx_q}{dx_c} \dfrac{dx_c}{dl} \approx \dfrac{dL}{dx_q} \dfrac{dx_c}{dl}$

Choi, Jungwook, et al. "Pact: Parameterized clipping activation for quantized neural networks." *arXiv preprint arXiv:1805.06085* (2018).

Learnable Quantization (PACT)

$$\frac{dL}{dl} \approx \frac{dL}{dW'} \frac{dW}{dW_c} \frac{dW_c}{dl}$$

1

Choi, Jungwook, et al. "Pact: Parameterized clipping activation for quantized neural networks." *arXiv preprint arXiv:1805.06085* (2018).
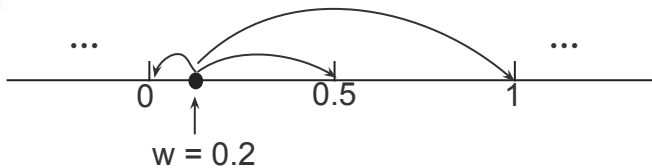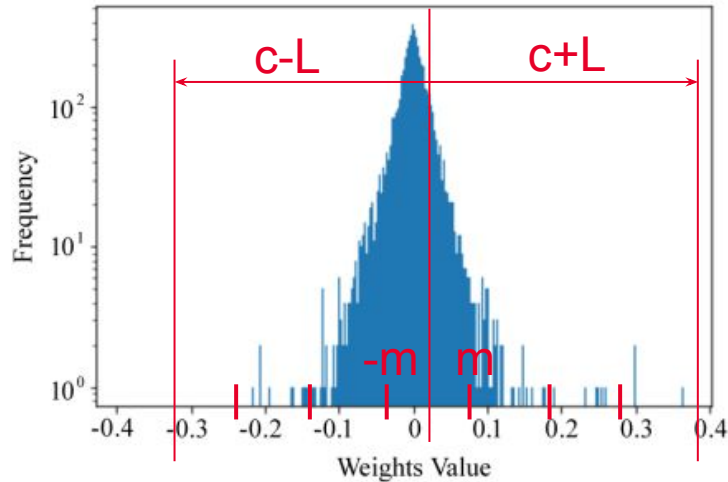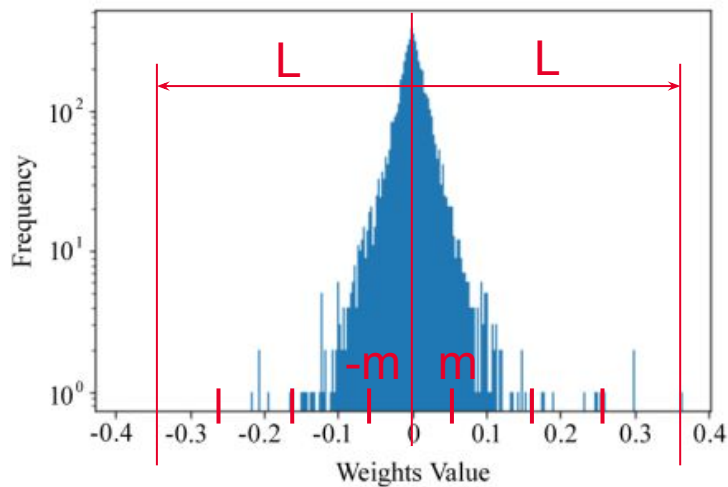
Learnable Quantization (PACT)
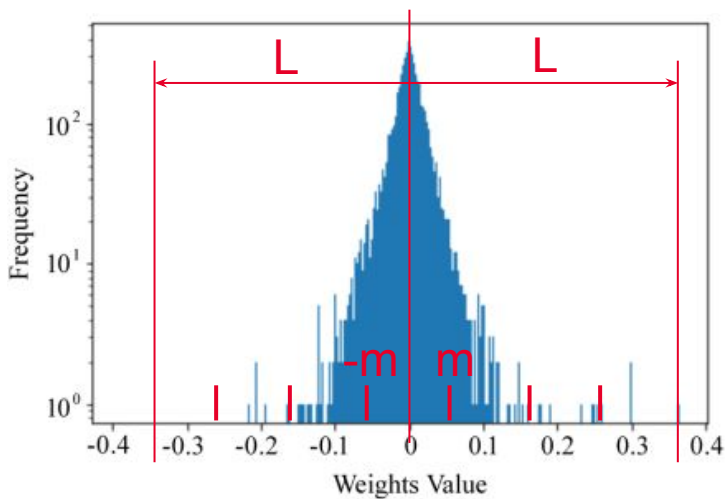
$$Clip(x, l) = \begin{cases} l, & \text{if } x \geq l \\ x, & -l \leq x \leq l \\ -l, & x \leq l \end{cases}$$

$$\frac{dClip(x, l)}{dx} = \begin{cases} 0, & \text{if } x \geq l \\ 1, & -l \leq x \leq l \\ 0, & x \leq l \end{cases}$$
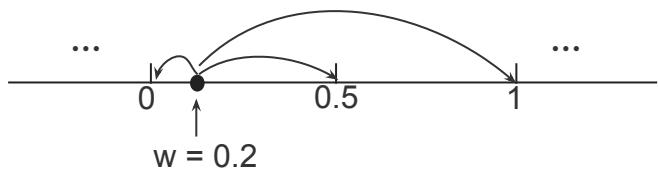
$$\frac{dClip(x, l)}{dl} = \begin{cases} 1, & \text{if } x \geq l \\ 0, & -l \leq x \leq l \\ -1, & x \leq l \end{cases}$$

L can be learnable

Choi, Jungwook, et al. "Pact: Parameterized clipping activation for quantized neural networks." *arXiv preprint arXiv:1805.06085* (2018).

Quantization Interval Learning (QIL)

Jung, Sangil, et al. "Learning to quantize deep networks by optimizing quantization intervals with task loss." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Quantization Interval Learning (QIL)

- To achieve this rounding flexibility, we combine a learnable function with quantization.

$$w_q = Q(w) \longrightarrow w_q = Q(F(w))$$

- $F(.)$ is a function which contains learnable hyperparameters.

$$\hat{w} = \begin{cases} 0 & |w| < c_W - d_W \\ \text{sign}(w) & |w| > c_W + d_W \\ (\alpha_W|w| + \beta_W)^{\gamma} \cdot \text{sign}(w) & otherwise, \end{cases}$$

w = 0.2

Jung, Sangil, et al. "Learning to quantize deep networks by optimizing quantization intervals with task loss." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Quantization Interval Learning (QIL)

- QIL offers flexibility to round the FP weights.

$$\hat{w} = \begin{cases} 0 & |w| < c_W - d_W \\ \text{sign}(w) & |w| > c_W + d_W \\ (\alpha_W|w| + \beta_W)^{\gamma} \cdot \text{sign}(w) & otherwise, \end{cases}$$

Mapping function contains some learnable parameters
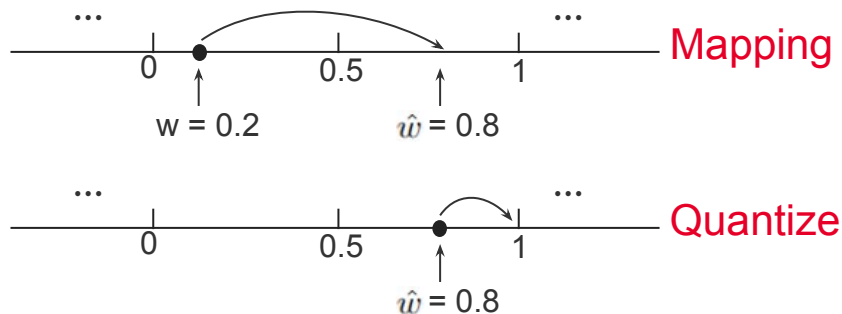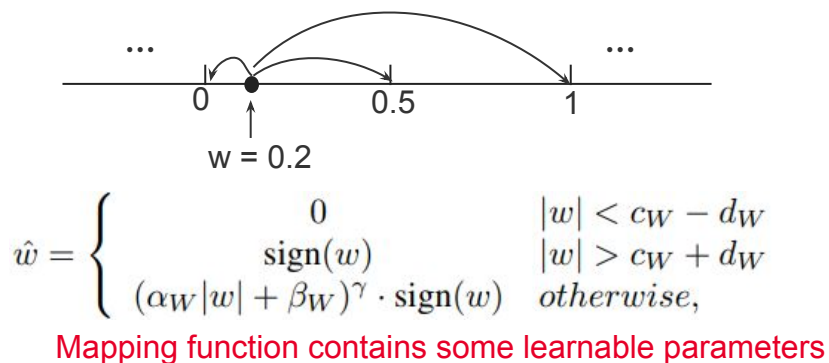
- $w_q = Q(F(w))$ are stored for inference after the training process finished.
- We can not apply this techniques over the activation, due to its large computational overhead.

Jung, Sangil, et al. "Learning to quantize deep networks by optimizing quantization intervals with task loss." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Quantization Networks (QN)

- We propose a novel perspective of interpreting and implementing neural network quantization by formulating low-bit quantization as a differentiable non-linear function.
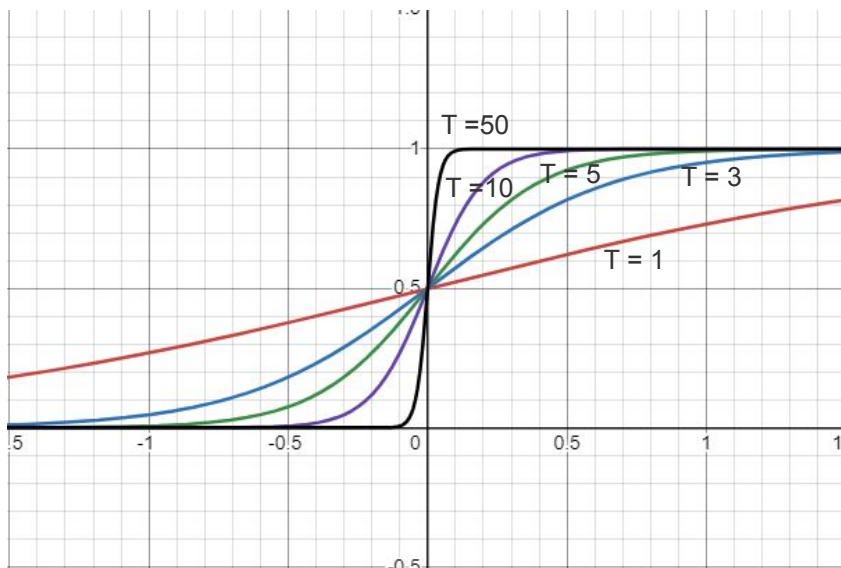

(a) No Quantization   (b) T=1   (c) T=11   (d) T=121   (e) Complete Quantization
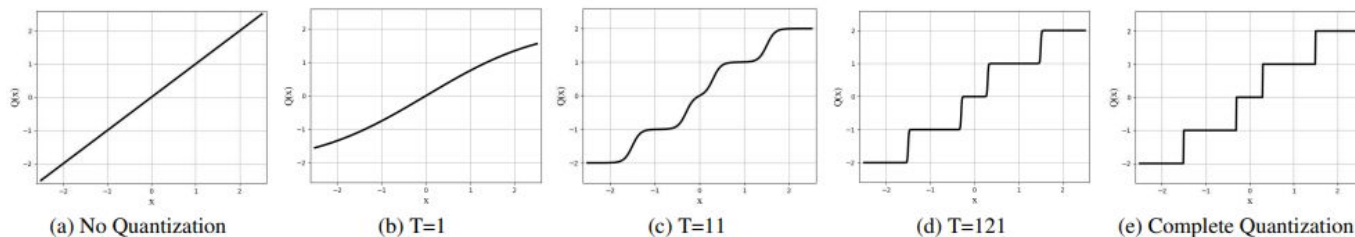
$$y = \alpha\left(\sum_{i=1}^{n} s_i \mathcal{A}(\beta x - b_i) - o\right)$$

$$\mathcal{A}(x) = \begin{cases} 1 & x \geq 0, \\ 0 & x < 0. \end{cases}$$

- $n + 1$ is the number of quantization intervals
- β is the scale factor of inputs
- s_i and b_i are the scales and biases for the unit step functions

Yang, Jiwei, et al. "Quantization networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

Gong, Ruihao, et al. "Differentiable soft quantization: Bridging full-precision and low-bit neural networks." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.

NYU SAI LAB

Quantization Networks (QN)

$$\mathcal{A}(x) = \begin{cases} 1 & x \geq 0, \\ 0 & x < 0. \end{cases} \qquad \sigma(Tx) = \frac{1}{1 + exp(-Tx)}$$

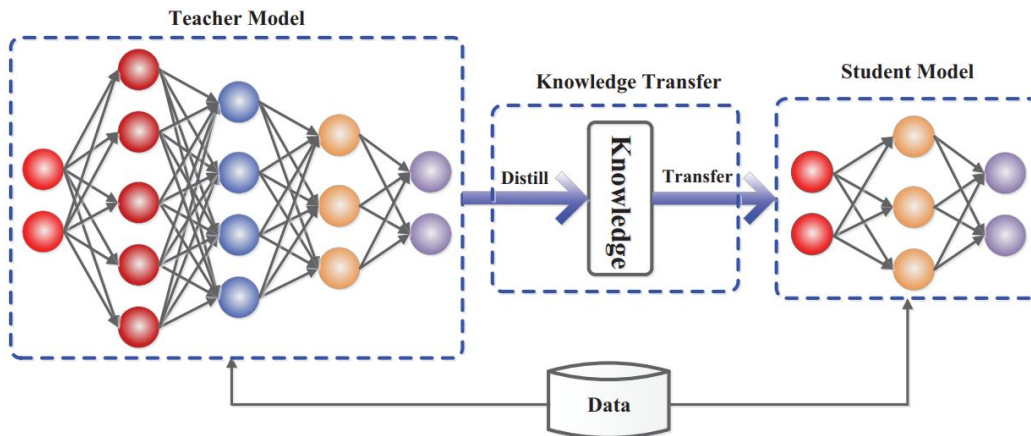- We can replace the staircase function with a sigmoid function.

- We progressively increases T during the training process.

Topics

- Learnable adaptive quantization scheme
- Distillation
- Neural architecture search (NAS)
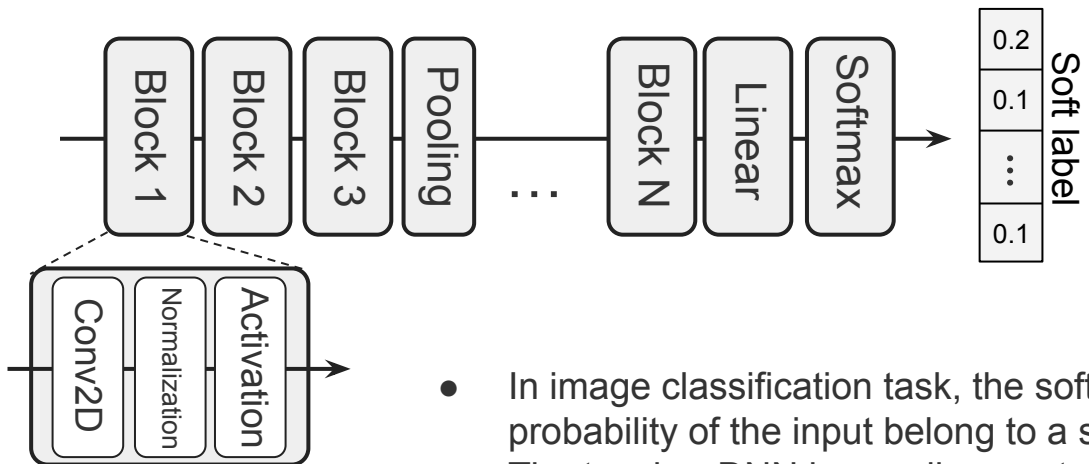- Low-rank factorization

NYU SAI LAB

Knowledge Distillation Basics

- Transferring knowledge from a large and complex model or set of models to a single, smaller model that can be effectively deployed in real-world scenarios with practical limitations.

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).

Knowledge Distillation Basics

- In image classification task, the soft labels indicate the probability of the input belong to a specific class.
- The teacher DNN is usually a pretrained model, which has a much larger size than the student model.
- The student model is smaller than the teacher model and does not need to have the same architecture as the teacher.

NYU SAI LAB

Response-Based Knowledge Distillation
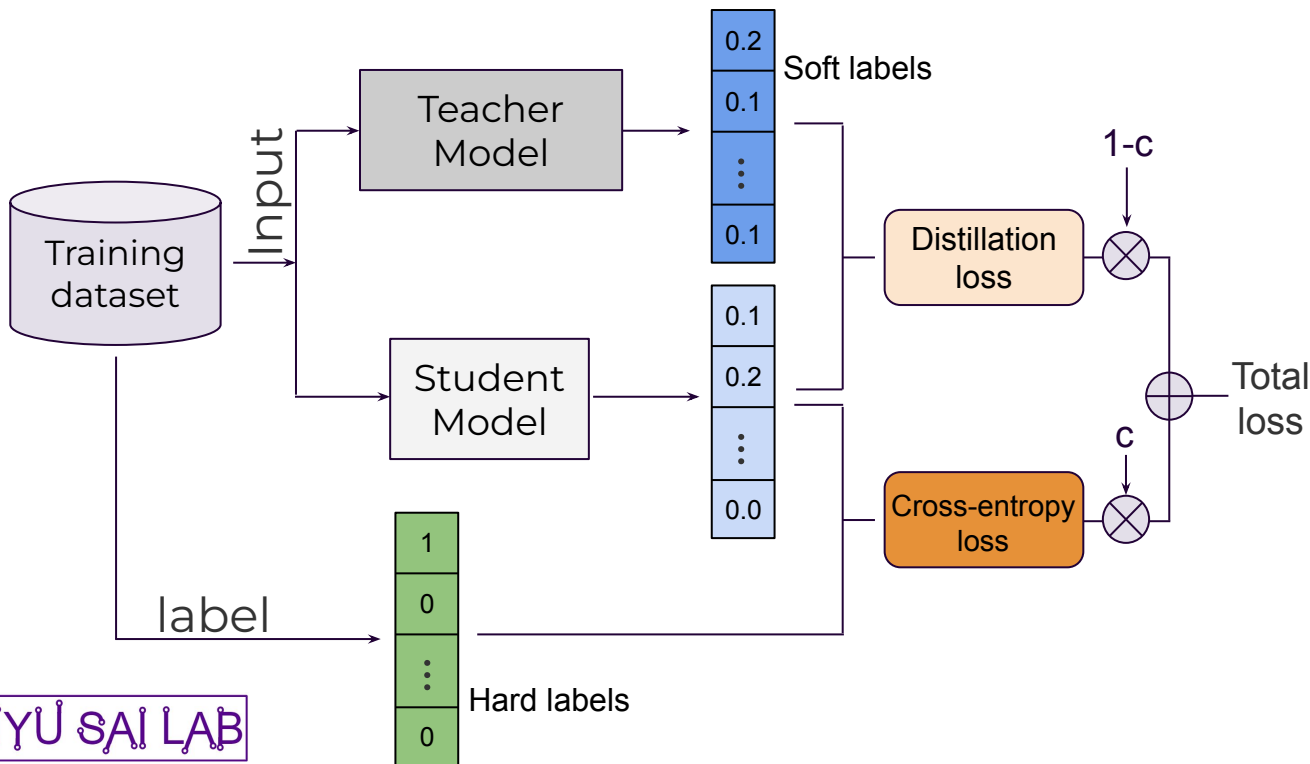
- During the training process, only the weights within the student model got updated.
- The teacher model can be either pretrained or trained together with student.
- c tends to be close to 1: ~0.9.

KL Divergence

- In mathematical statistics, the Kullback–Leibler (KL) divergence, denoted $D_{KL}(P \,/\!/\, Q)$, is a measure of how much a model probability distribution Q is different from a true probability distribution P.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \, \log\left(\frac{P(x)}{Q(x)} \right)$$

- If P(x) = Q(x) for all x, then the KL divergence equals 0.
- Otherwise, KL divergence is greater than 0.

Response-Based Knowledge Distillation

- The soft labels are computed as follows:

$$p(z_i, T) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

T is the temperature
zi is called logits
p is the soft label

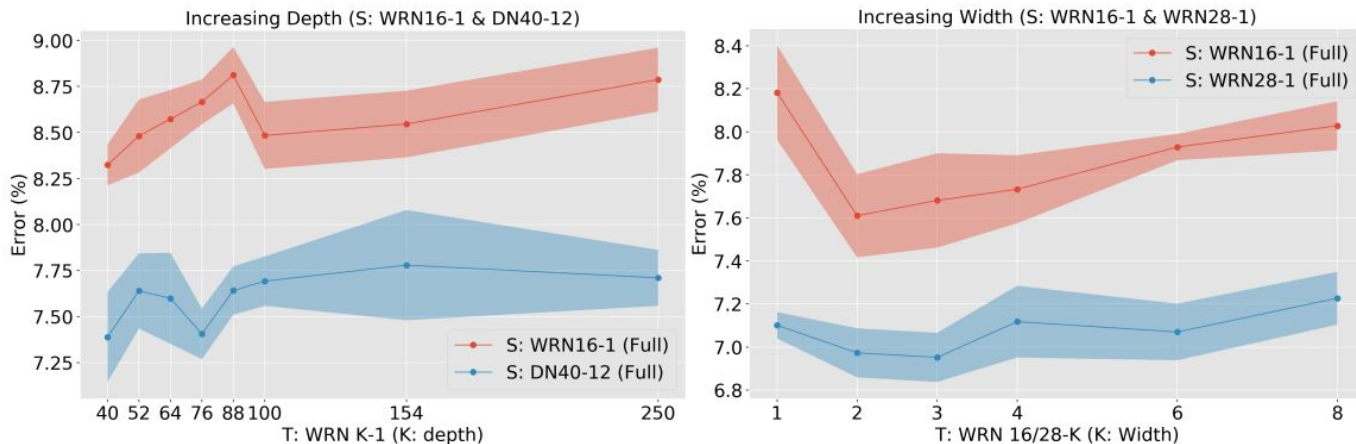- Kullback Leibler (KL) divergence is usually used to generate the distillation loss.

$$L(y_{\text{pred}}, y_{\text{true}}) = y_{\text{true}} \cdot \log \frac{y_{\text{true}}}{y_{\text{pred}}} = y_{\text{true}} \cdot (\log y_{\text{true}} - \log y_{\text{pred}})$$

- The loss function of the student model includes both distillation loss and the conventional cross-entropy loss.

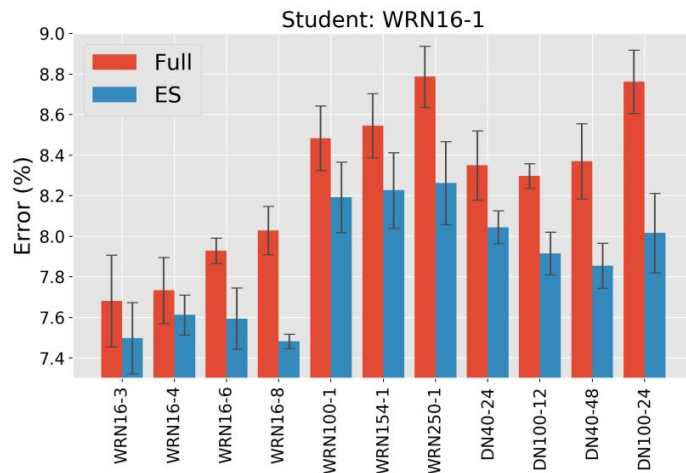NYU SAI LAB

How Smart the Teacher Need to Be?

- Larger teachers, though they are more accurate by themselves, do not necessarily make for better teachers. A too large teacher model can even harm the performance of the student.

- As the teacher grows in capacity and accuracy, the student often finds it difficult to emulate the teacher.

Cho, Jang Hyun, and Bharath Hariharan. "On the efficacy of knowledge distillation." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.
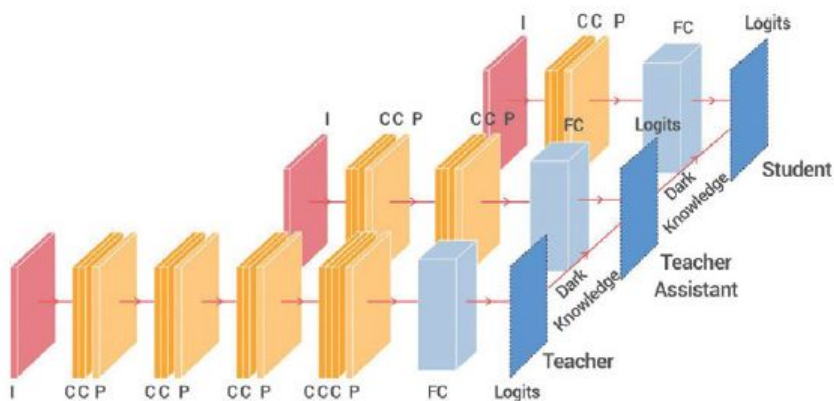
How Smart the Teacher Need to Be?

- Increasing teacher capacity (depth: left, width: right) and thus accuracy does not necessarily increase the accuracy of the student network.
- If the teacher network is too large, the student is unable to find a solution in its space that corresponds well to the largest teacher.

Cho, Jang Hyun, and Bharath Hariharan. "On the efficacy of knowledge distillation." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.
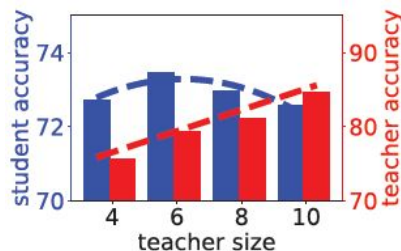
How Smart the Teacher Need to Be?

- We may perform grid-search to find the optimal teacher network architecture, but that is too expensive.
- In particular, we propose to early stop (ES) the training process.

Cho, Jang Hyun, and Bharath Hariharan. "On the efficacy of knowledge distillation." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.
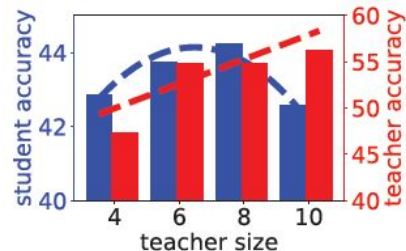
Improved Knowledge Distillation via Teacher Assistant

- Knowledge distillation is not always effective, especially when the gap (in size) between teacher and student is large.
- TA models are distilled from the teacher, and the student is then only distilled from the TAs

a) CIFAR-10

b) CIFAR-100

Mirzadeh, Seyed Iman, et al. "Improved knowledge distillation via teacher assistant." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. No. 04. 2020.

Improved Knowledge Distillation via Teacher Assistant

Table 1: Comparison on evaluation accuracy between our method (TAKD) and baselines. For CIFAR, plain (S=2, TA=4, T=10) and for ResNet (S=8, TA=20, T=110) are used. For ImageNet, ResNet (S=14, TA=20, T=50) is used. Higher numbers are better.
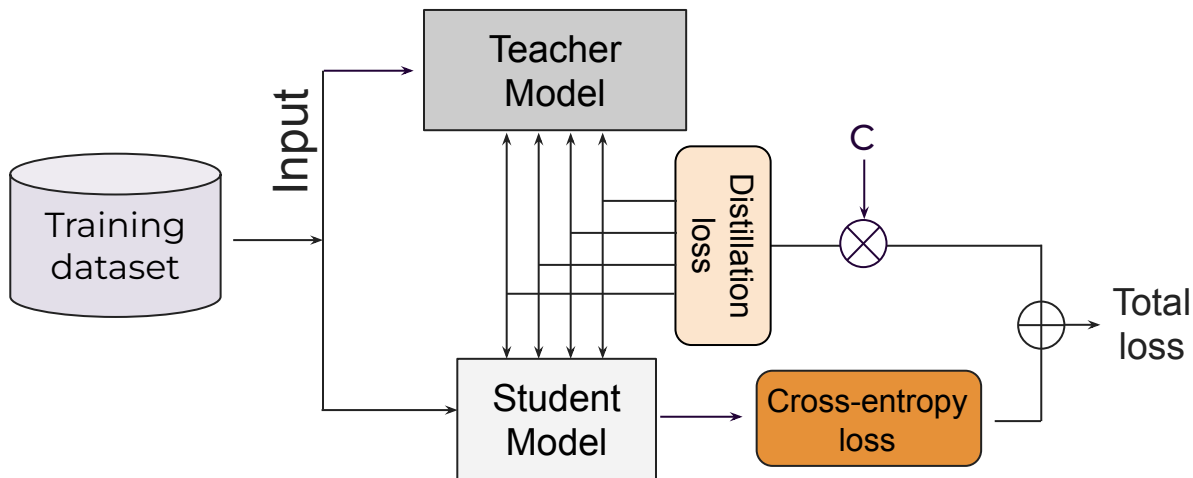
Model	Dataset	NOKD	BLKD	TAKD
CNN	CIFAR-10	70.16	72.57	73.51
	CIFAR-100	41.09	44.57	44.92
ResNet	CIFAR-10	88.52	88.65	88.98
	CIFAR-100	61.37	61.41	61.82
ResNet	ImageNet	65.20	66.60	67.36

- In this paper, the authors propose to use intermediate-size networks to fill in the gap between them.
- First, the TA network is distilled from the teacher. Then, the TA plays the role of a teacher and trains the student via distillation.

Mirzadeh, Seyed Iman, et al. "Improved knowledge distillation via teacher assistant." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. No. 04. 2020.

NYU SAI LAB

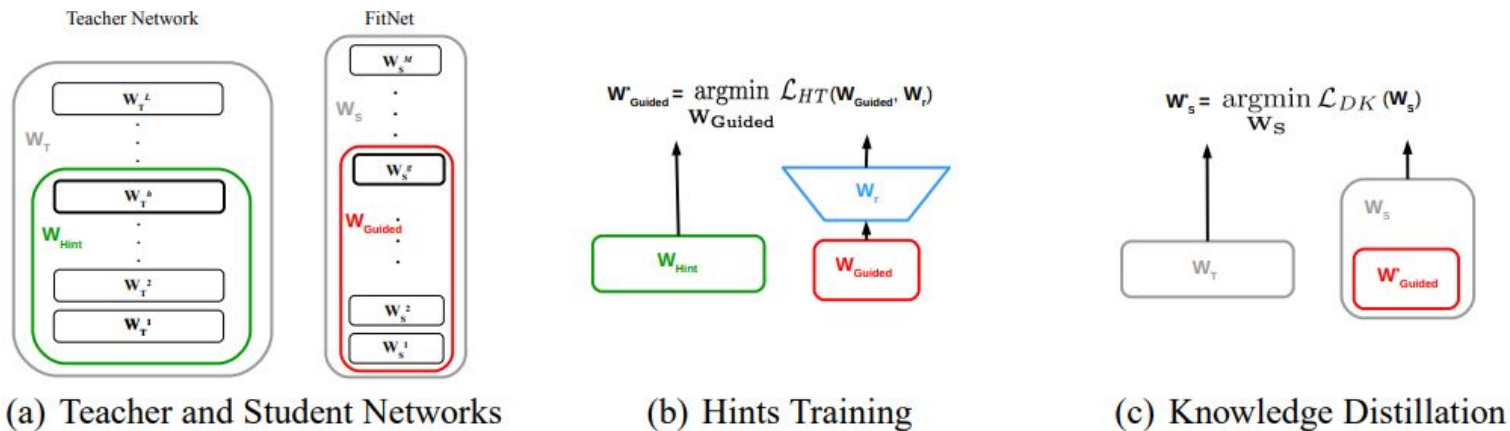# Feature-Based Knowledge Distillation

- Response-based knowledge usually relies on the output of the last layer, e.g., soft targets, and thus fails to address the intermediate-level supervision from the teacher model.
- Both the output of the last layer and the output of intermediate layers, i.e., feature maps, can be used as the knowledge to supervise the training of the student model.

Srivastava, Rupesh K., Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks." *Advances in neural information processing systems* 28 (2015).

NYU SAI LAB

Feature-Based Knowledge Distillation

- For feature based Knowledge, distillation, the intermediate results from the teacher model is extracted and guide the training for the student DNN.

FitNets: Hints for Thin Deep Nets

(a) Teacher and Student Networks   (b) Hints Training   (c) Knowledge Distillation

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

HINT-BASED TRAINING

NYU SAI LAB   Romero, Adriana, et al. "Fitnets: Hints for thin deep nets." *arXiv preprint arXiv:1412.6550* (2014).

FitNets: Hints for Thin Deep Nets

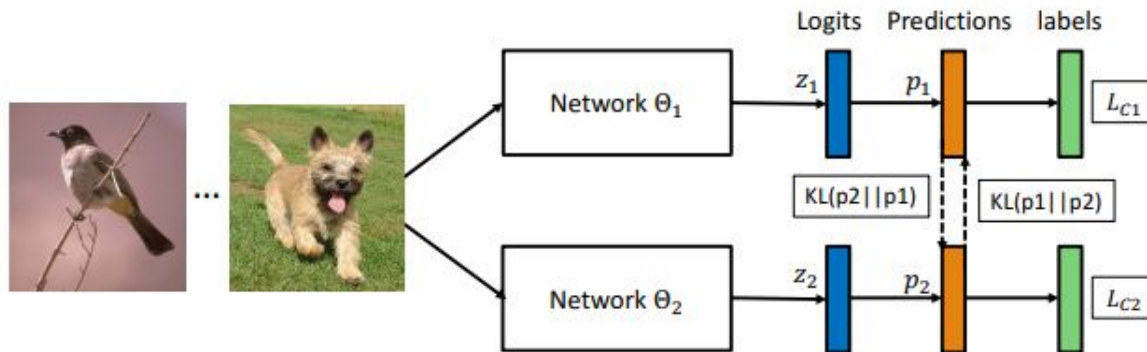$$\mathcal{L}_{HT}(\mathbf{W_{Guided}}, \mathbf{W_r}) = \frac{1}{2}||u_h(\mathbf{x}; \mathbf{W_{Hint}}) - r(v_g(\mathbf{x}; \mathbf{W_{Guided}}); \mathbf{W_r})||^2$$ HINT-BASED TRAINING
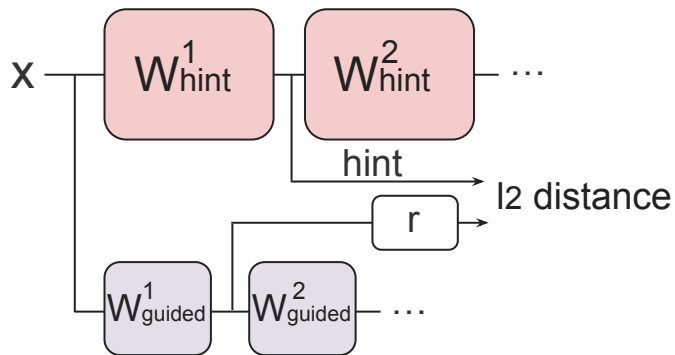
- r(.) consists of conv layers, in order to produce the outputs with the same dimension as the teacher DNN.

Romero, Adriana, et al. "Fitnets: Hints for thin deep nets." *arXiv preprint arXiv:1412.6550* (2014).
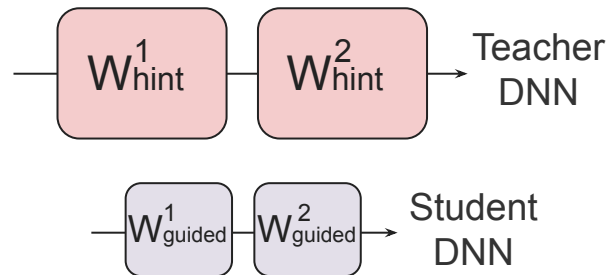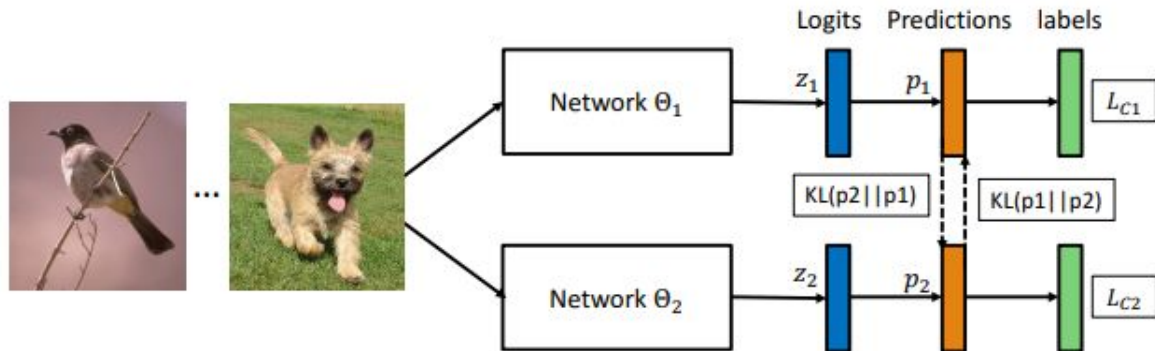
Distillation Schemes: Online Distillation

- An ensemble of students learn collaboratively and teach each other throughout the training process
- A similar method can be extended to a larger group of students.

Zhang, Ying, et al. "Deep mutual learning." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

Distillation Schemes: Online Distillation

$$D_{KL}(\boldsymbol{p}_2 \| \boldsymbol{p}_1) = \sum_{i=1}^{N} \sum_{m=1}^{M} p_2^m(\boldsymbol{x}_i) \log \frac{p_2^m(\boldsymbol{x}_i)}{p_1^m(\boldsymbol{x}_i)}$$

$$L_{\Theta_1} = L_{C_1} + D_{KL}(\boldsymbol{p}_2 \| \boldsymbol{p}_1)$$

$$L_{\Theta_2} = L_{C_2} + D_{KL}(\boldsymbol{p}_1 \| \boldsymbol{p}_2)$$

Zhang, Ying, et al. "Deep mutual learning." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

Distillation Schemes: Online Distillation

Algorithm 1: Deep Mutual Learning

Input: Training set $\mathcal{X}$, label set $\mathcal{Y}$, learning rate $\gamma_{1,t}$ and $\gamma_{2,t}$.

Initialize: Models Θ_1 and Θ_2 to different initial conditions.

Repeat :

  $t = t + 1$

  Randomly sample data x from $\mathcal{X}$.

  1: Update the predictions p_1 and p_2 of x by (1) for the current mini-batch

  2: Compute the stochastic gradient and update Θ_1 :

$$\Theta_1 \leftarrow \Theta_1 + \gamma_{1,t} \frac{\partial L_{\Theta_1}}{\partial \Theta_1}$$

  3: Update the predictions p_1 and p_2 of x by (1) for the current mini-batch

  4: Compute the stochastic gradient and update Θ_2 :

$$\Theta_2 \leftarrow \Theta_2 + \gamma_{2,t} \frac{\partial L_{\Theta_2}}{\partial \Theta_2}$$
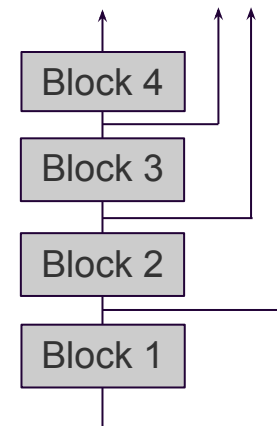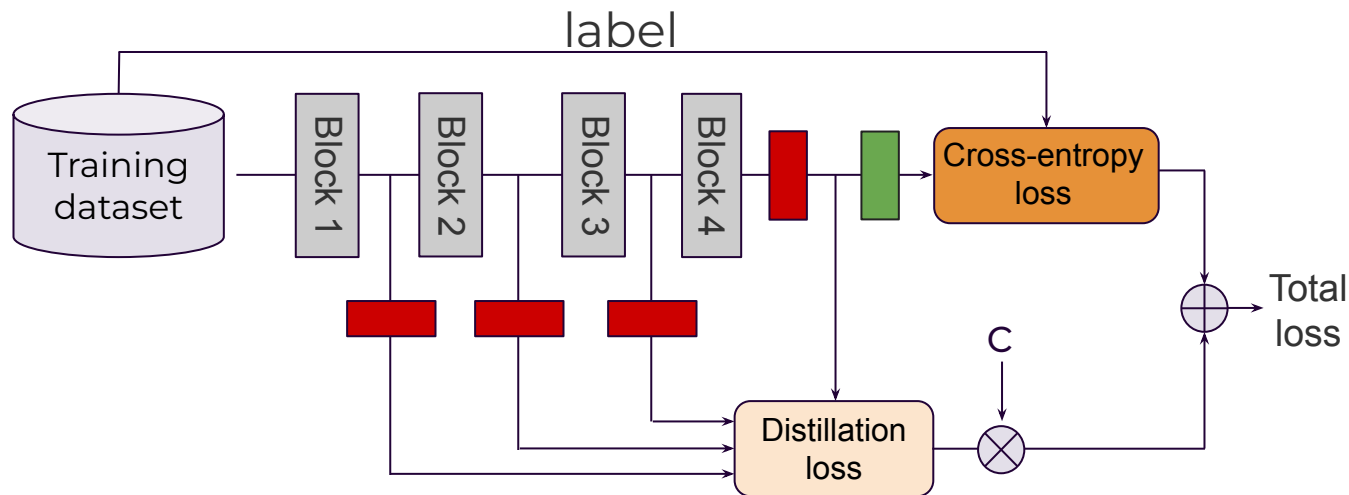
Until : convergence

- The same concept can be applied to situations involving multiple students.
- A cohort of students to learn mutually.

NYU SAI LAB

32

Self Distillation

- The networks are firstly divided into several components, and the knowledge from the deeper parts is condensed into the shallower ones.

Zhang, Linfeng, et al. "Be your own teacher: Improve the performance of convolutional neural networks via self distillation." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.

# Relation between Student and Teacher

- The student model should have some architectural similarities with the teacher.
    - It can be a mini version of teacher model (shallower, thinner)
    - It can be a pruned version of teacher
    - It can be a quantized version of teacher
    - It can be a subset of teacher
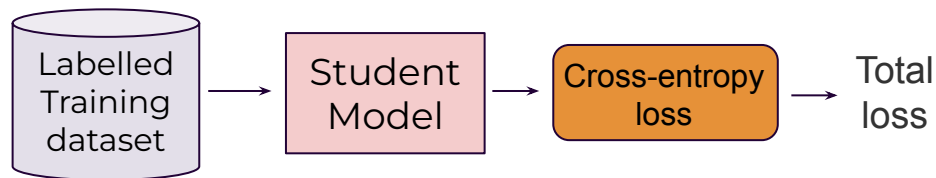
Multi-teacher Distillation
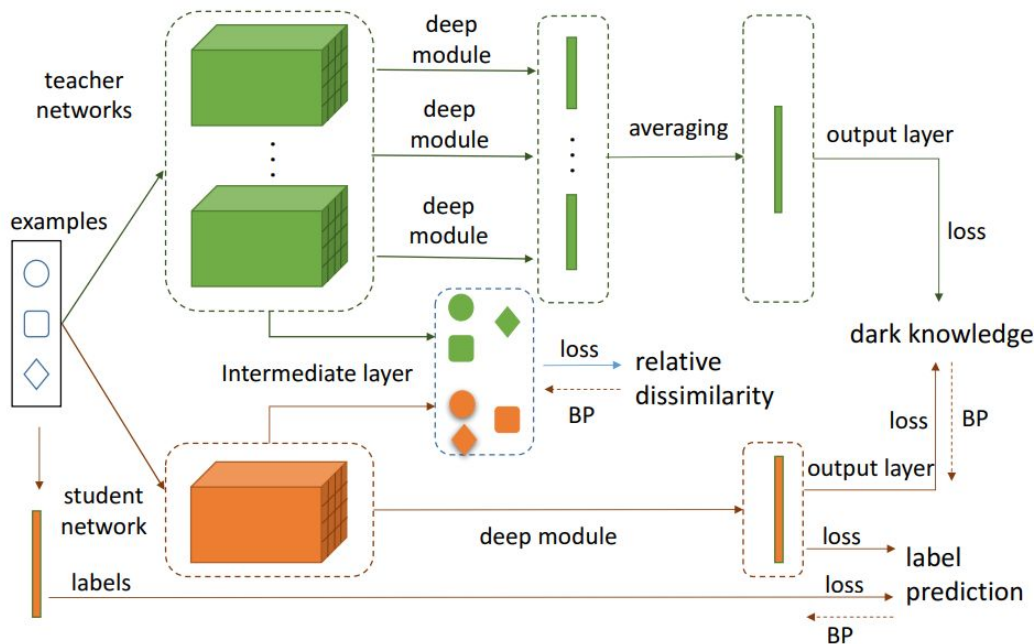
- We present a method to train a student network by incorporating multiple teacher networks not only in output layer by averaging the softened outputs (dark knowledge) from different networks, but also in the intermediate layers by imposing a constraint about the dissimilarity among examples.

Multi-teacher Distillation
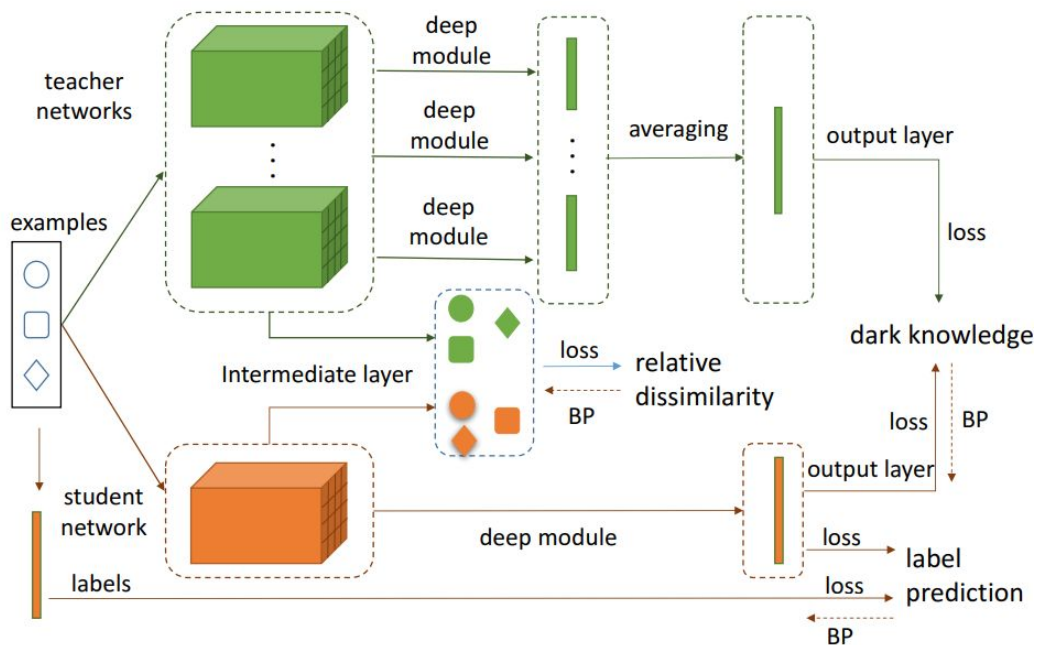
- Multiple teacher can better guide the student DNN to learn.
- Each teacher network can with different architecture, or even same architecture but different weights.

You, Shan, et al. "Learning from multiple teacher networks." *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017.

Cross-Modal Distillation

- If the labeled data on the new tasks is limited (e.g., segmentation), we can pass the unlabelled data to the teacher model pretrained for another similar task (e.g., classification), and use the feature-based knowledge distillation to better guide the student.
- The teacher network, despite being trained on different vision tasks, still demonstrates strong proficiency in understanding visual inputs.
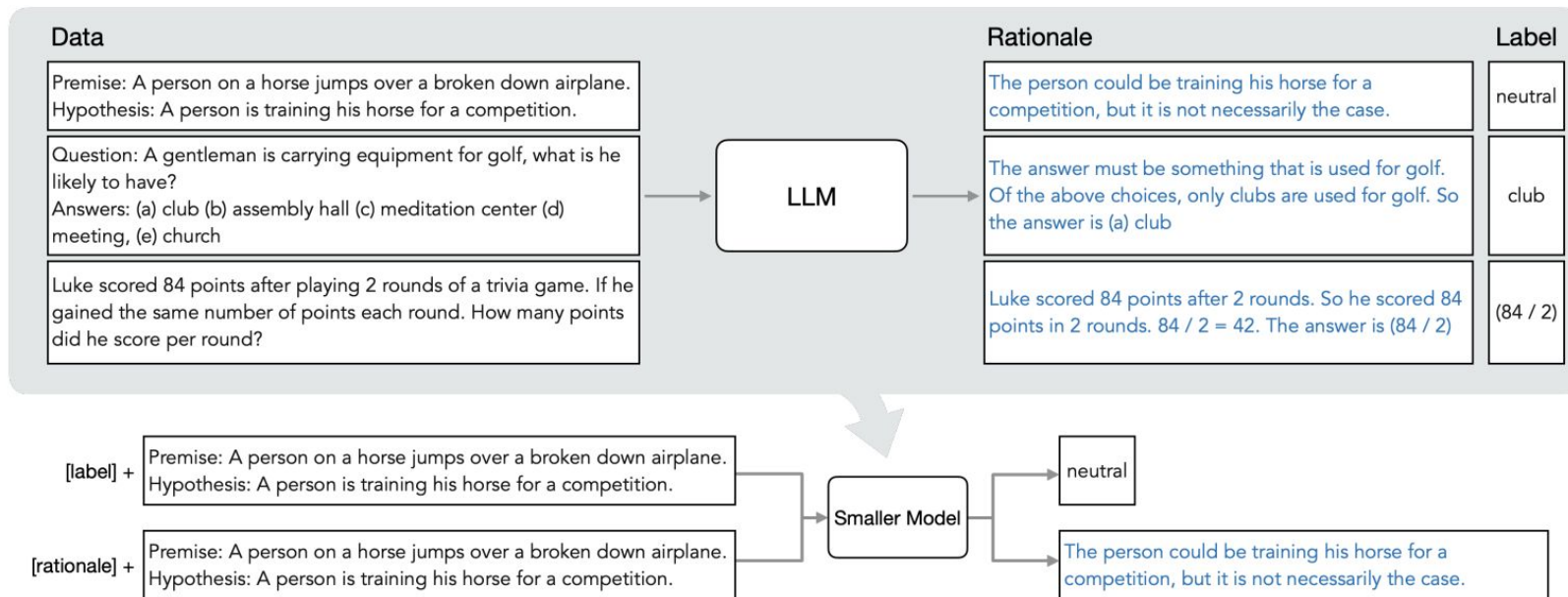
NYU SAI LAB
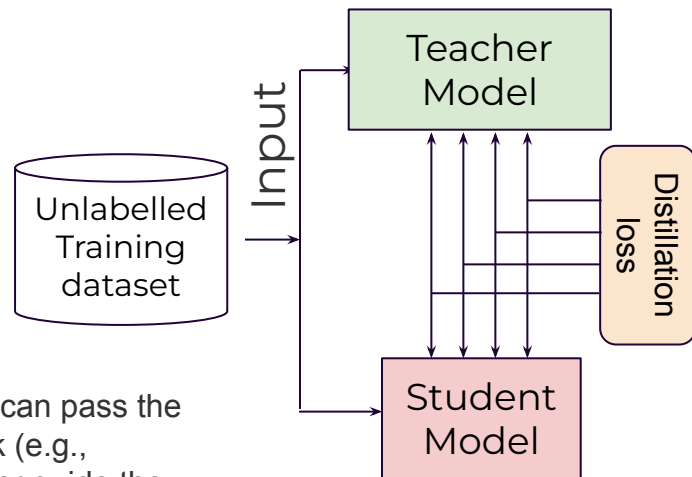
Distillation for Language Models
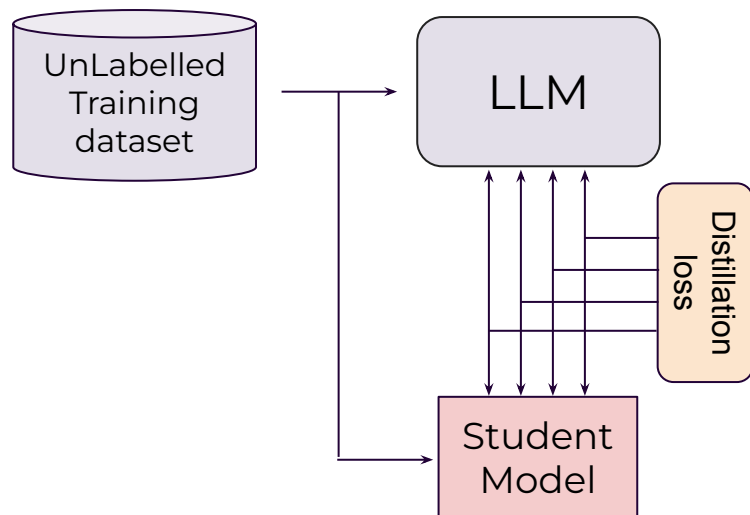
- Given the unlabelled data, LLM can leverage its knowledge to provide:
  - ground-truth label
  - Features (rationale)
- Distilling step-by-step is a new simple mechanism for training smaller models with less training data.
- This mechanism reduces the amount of training data required for both finetuning and distillation of LLMs into smaller model sizes.

Hsieh, Cheng-Yu, et al. "Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes." *arXiv preprint arXiv:2305.02301* (2023).

Distilling Step-by Step

- First, given an LLM and an unlabeled dataset, we prompt the LLM to generate output labels along with rationales to justify the labels

NYU SAI LAB

Distilling Step-by Step
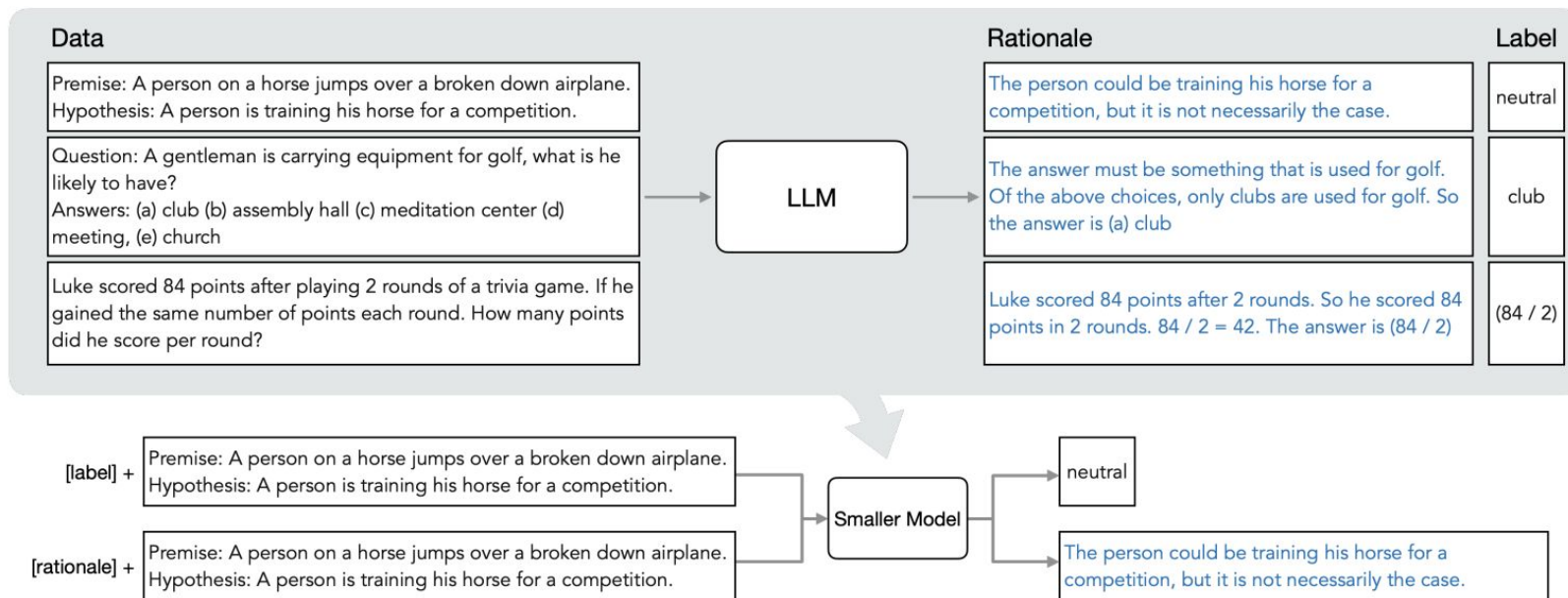
- Second, we leverage these rationales in addition to the task labels to train smaller downstream models.

40

Chain-of-Thought (CoT)

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✔️

- CoT allows models to decompose multi-step problems into intermediate steps, which means that additional computation can be allocated to problems that require more reasoning steps.

NYU SAI LAB

Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." *Advances in neural information processing systems* 35 (2022): 24824-24837.

Distilling Step-by Step

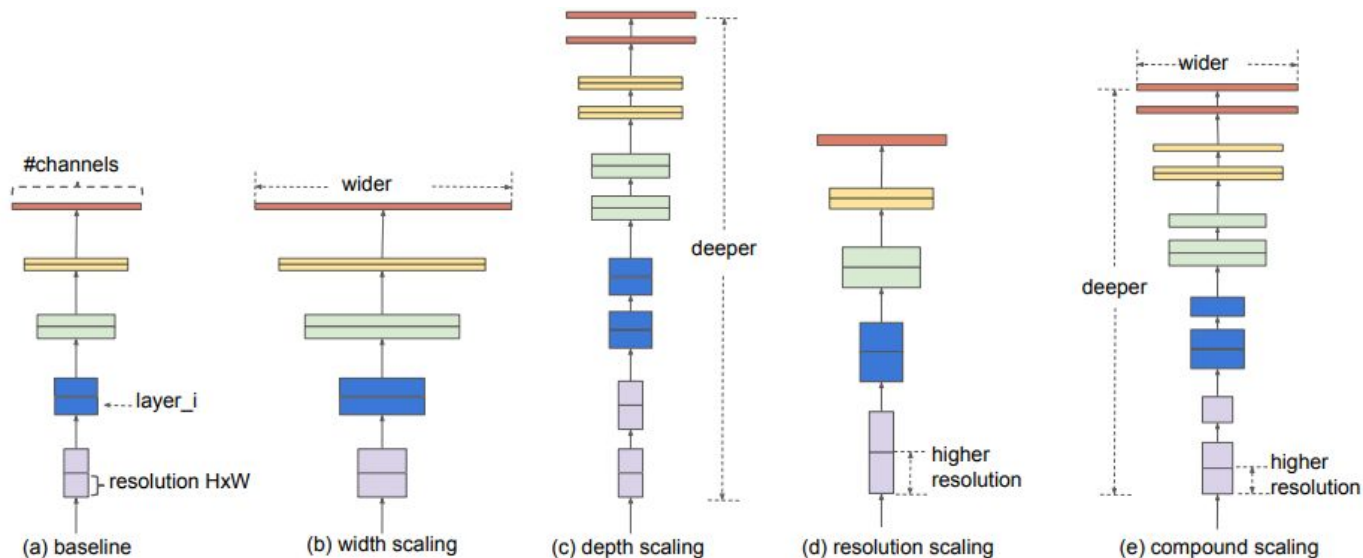|	
Few-shot CoT	Question: Sammy wanted to go to where the people were. Where might he go? Answer Choices: (a) populated areas (b) race track (c) desert (d) apartment (e) roadblock Answer: The answer must be a place with a lot of people. Of the above choices, only populated areas have a lot of people. So the answer is (a) populated areas.
Input	Question: A gentleman is carrying equipment for golf, what is he likely to have? Answers: (a) club (b) assembly hall (c) meditation center (d) meeting, (e) church Answer:
Output	The answer must be something that is used for golf. Of the above choices, only clubs are used for golf. So the answer is (a) club.

- We use few-shot Chain-of-thought prompting that contains both an example rationale (highlighted in green) and a label (highlighted in blue) to elicit rationales from an LLM on new input examples.
- We first design a prompt template p that articulates how the task should be solved.
- The rationale generation loss enables the model to learn to generate the intermediate reasoning steps for the prediction, and could therefore guide the model in better predicting the resultant label.

$$\mathcal{L} = \mathcal{L}_{\text{label}} + \lambda \mathcal{L}_{\text{rationale}}$$

Topics

- Learnable adaptive quantization scheme
- Distillation
- Neural architecture search (NAS)
- Low-rank factorization

NYU SAI LAB

Neural Architecture Search (NAS)

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

- How can a DNN be designed with an optimal architecture to maximize accuracy on the test dataset?

Neural Architecture Search (NAS)

- Neural Architecture Search (NAS) is an automated process for designing neural network architectures.
- Traditional methods of creating neural networks rely heavily on human expertise and manual experimentation, which can be time-consuming and suboptimal.
- NAS aims to alleviate these challenges by using algorithms to explore the vast space of possible network architectures to find the most efficient and effective designs for specific tasks.
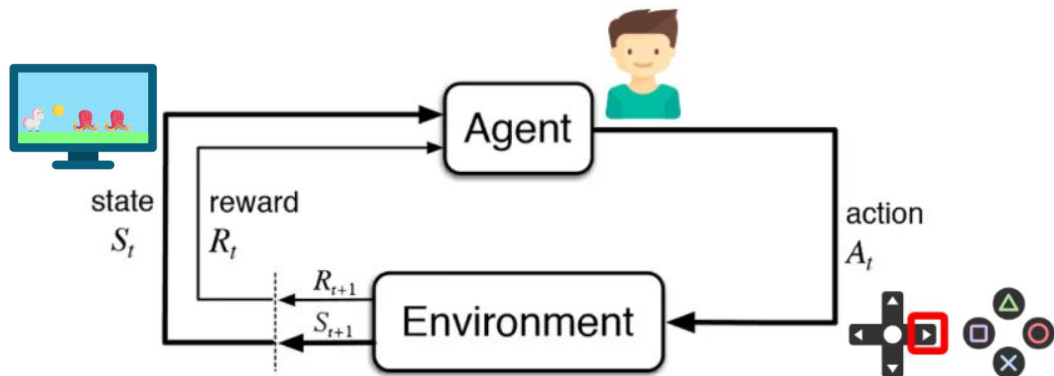
Deep Reinforcement Learning

Agent   **State**   **Agent**   **State**

Action

Reward

+1
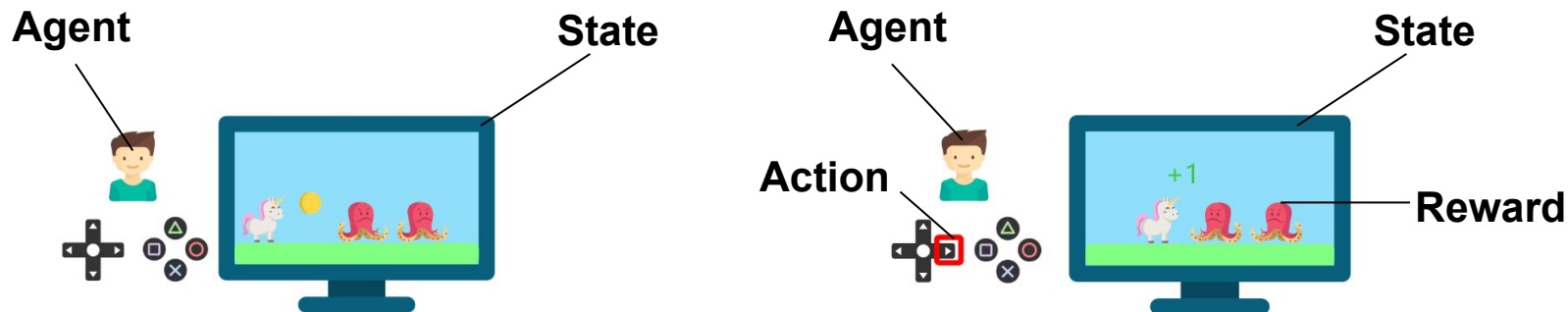
- Reinforcement learning is an area of machine learning where an agent learns to make decisions by performing actions in an environment to maximize some cumulative reward.

NYU SAI LAB

Deep Reinforcement Learning

- Agent: player
- State: game image
- Action: one button in gamepad
- Environment: game software

Learning Transferable Architectures for Scalable Image Recognition
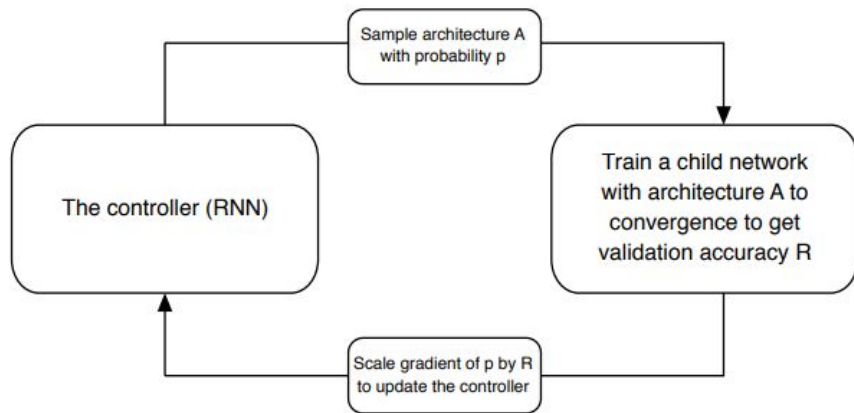
- We use a recurrent network to generate the model descriptions of neural networks and train this RNN with reinforcement learning to maximize the expected accuracy of the generated architectures on a validation set.

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

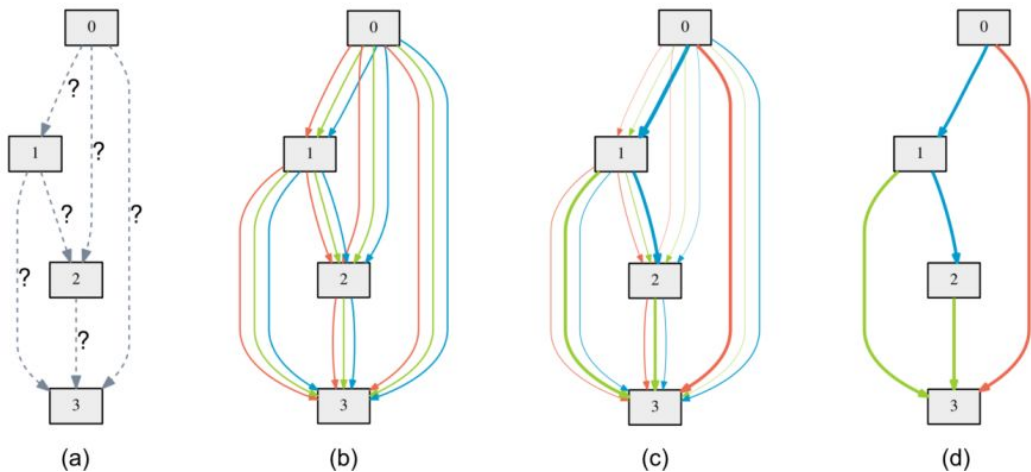# Learning Transferable Architectures for Scalable Image Recognition
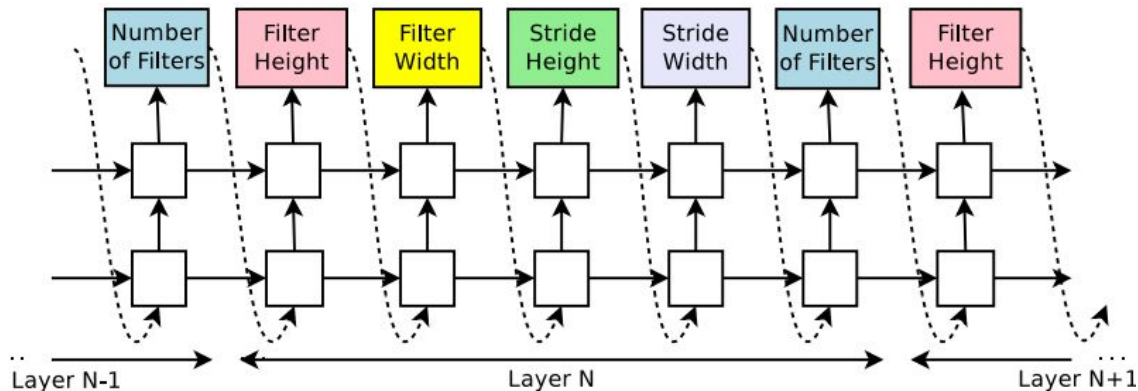
- Reward is based on the accuracy on the validation dataset.
- The action of the current step is based on the all the previous history on action is has taken.

Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578* (2016).

NYU SAI LAB

49

Learning Transferable Architectures for Scalable Image Recognition

Model	Depth	Parameters	Error rate (%)
Network in Network (Lin et al., 2013)	-	-	8.81
All-CNN (Springenberg et al., 2014)	-	-	7.25
Deeply Supervised Net (Lee et al., 2015)	-	-	7.97
Highway Network (Srivastava et al., 2015)	-	-	7.72
Scalable Bayesian Optimization (Snoek et al., 2015)	-	-	6.37
FractalNet (Larsson et al., 2016)	21	38.6M	5.22
with Dropout/Drop-path	21	38.6M	4.60
ResNet (He et al., 2016a)	110	1.7M	6.61
ResNet (reported by Huang et al. (2016c))	110	1.7M	6.41
ResNet with Stochastic Depth (Huang et al., 2016c)	110	1.7M	5.23
	1202	10.2M	4.91
Wide ResNet (Zagoruyko & Komodakis, 2016)	16	11.0M	4.81
	28	36.5M	4.17
ResNet (pre-activation) (He et al., 2016b)	164	1.7M	5.46
	1001	10.2M	4.62
DenseNet ($L = 40, k = 12$) Huang et al. (2016a)	40	1.0M	5.24
DenseNet($L = 100, k = 12$) Huang et al. (2016a)	100	7.0M	4.10
DenseNet ($L = 100, k = 24$) Huang et al. (2016a)	100	27.2M	3.74
DenseNet-BC ($L = 100, k = 40$) Huang et al. (2016b)	190	25.6M	3.46
Neural Architecture Search v1 no stride or pooling	15	4.2M	5.50
Neural Architecture Search v2 predicting strides	20	2.5M	6.01
Neural Architecture Search v3 max pooling	39	7.1M	4.47
Neural Architecture Search v3 max pooling + more filters	39	37.4M	3.65

DARTS: Differentiable Architecture Search

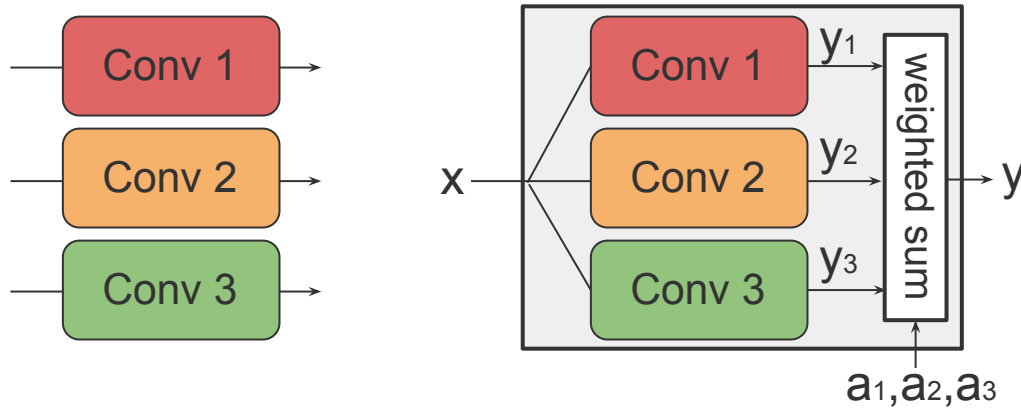

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$

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

$$\text{s.t.} \quad w^*(\alpha) = \text{argmin}_w \ \mathcal{L}_{train}(w, \alpha)$$

- DARTs continuous relax of the search space by placing a mixture of candidate operations on each edge, and combines the outputs from each branch using softmax.

Liu, Hanxiao, Karen Simonyan, and Yiming Yang. "Darts: Differentiable architecture search." *arXiv preprint arXiv:1806.09055* (2018).

DARTS: Differentiable Architecture Search

$y_1 = \text{Conv}(x, w_1)$

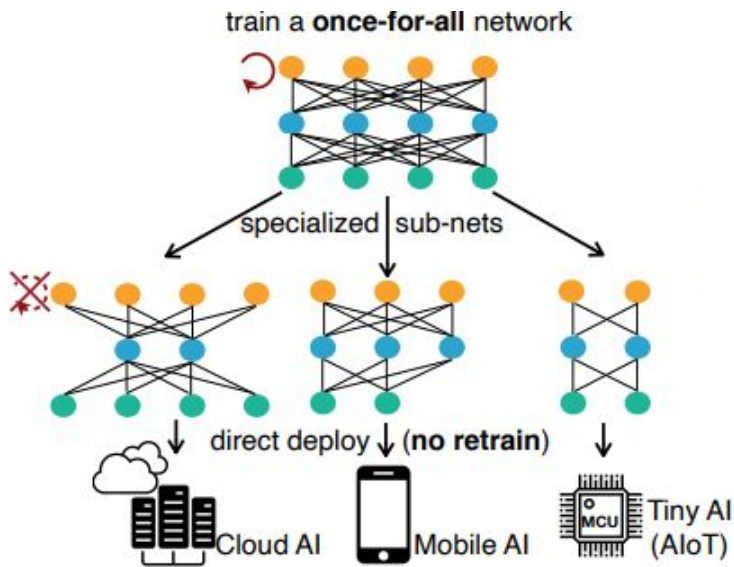$y_2 = \text{Conv}(x, w_2)$

$y_3 = \text{Conv}(x, w_3)$

$$s_1 = \frac{e^{a_1}}{e^{a_1} + e^{a_2} + e^{a_3}}$$

$$s_2 = \frac{e^{a_2}}{e^{a_1} + e^{a_2} + e^{a_3}}$$

$$s_3 = \frac{e^{a_3}}{e^{a_1} + e^{a_2} + e^{a_3}}$$

- At the end of search, we simply select the conv layer with the highest s.
- We can also add additional regularization term to push s to be one-hot.

$y = s_1 y_1 + s_2 y_2 + s_3 y_3$

Once-For-All: Train One Network And Specialize It for Efficient Deployment

- A supernet is train which contained multiple smaller subnetworks.

- All the subnetworks are trained jointly.

Cai, Han, et al. "Once-for-all: Train one network and specialize it for efficient deployment." *arXiv preprint arXiv:1908.09791* (2019).
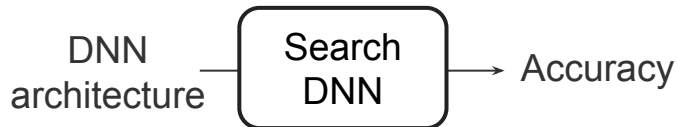
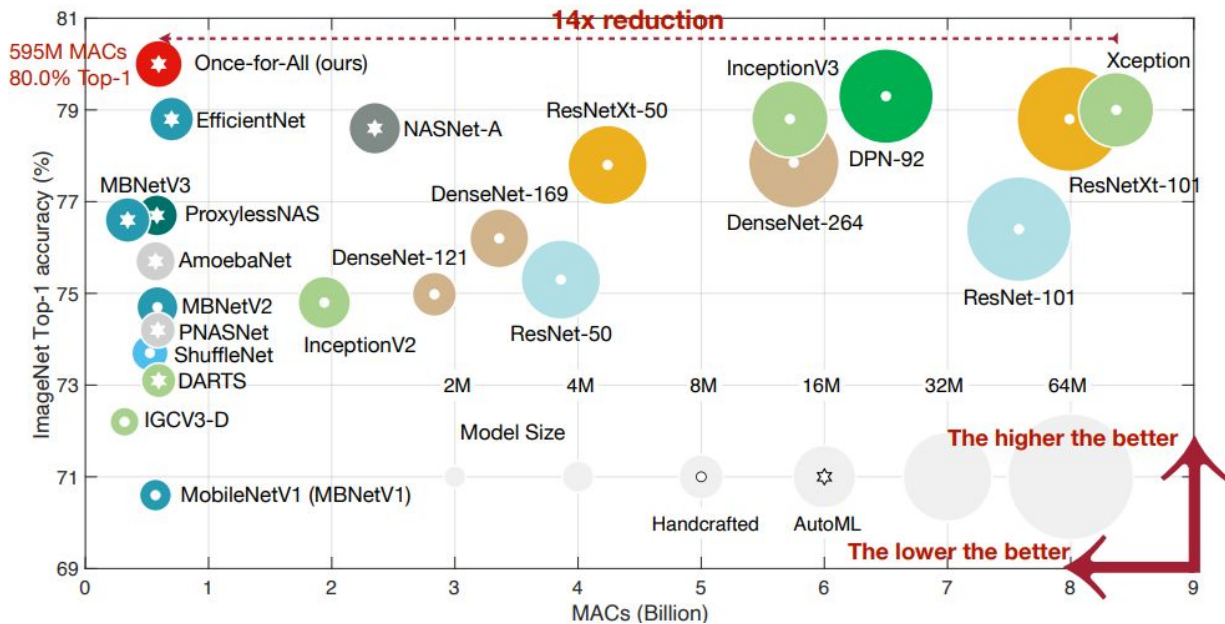# Once-For-All: Train One Network And Specialize It for Efficient Deployment
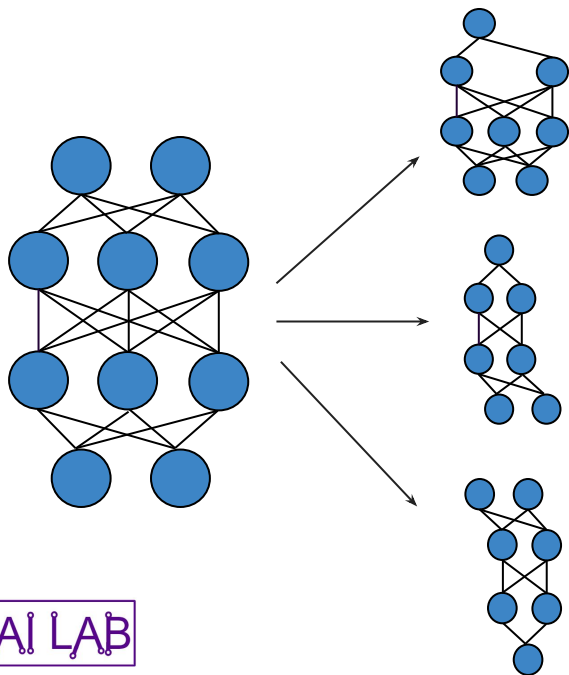
- During the forward pass, a subnetwork is sampled.
- All the weight within the subnetwork is trained in this iteration.
- All the subnetworks are nested within the OFA.
- A DNN is trained to search for the subnetwork.
  - Takes the hardware performance as input.
  - Return the corresponding subnetwork that satisfies the criteria.

NYU SAI LAB

Once-For-All: Train One Network And Specialize It for Efficient Deployment

- The once-for-all network comprises many sub-networks of different sizes where small sub-networks are nested in large sub-networks.
- After OFA is trained, we build neural-network-twins to predict the latency and accuracy given a neural network architecture.
- We randomly sample 16K sub-networks with different architectures and input image sizes, then measure their accuracy on 10K validation images sampled from the original training set.

DNN architecture → Search DNN → Accuracy

Once-For-All: Train One Network And Specialize It for Efficient Deployment
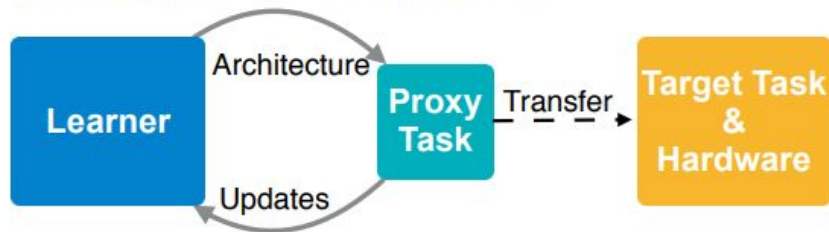

Cai, Han, et al. "Once-for-all: Train one network and specialize it for efficient deployment." *arXiv preprint arXiv:1908.09791* (2019).
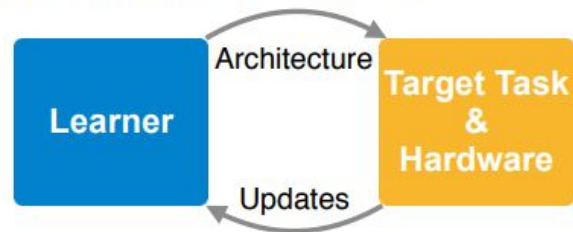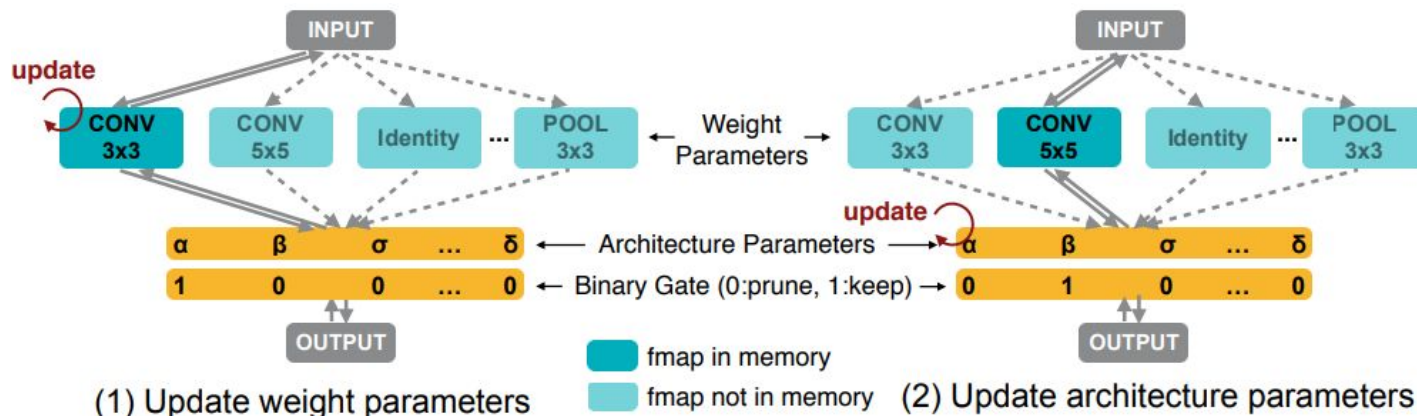
ProxylessNAS

(1) Previous proxy-based approach

Learner → Architecture → **Proxy Task** → Transfer → **Target Task & Hardware**

Updates ← Learner

(2) Our proxy-less approach

Learner → Architecture → **Target Task & Hardware**

Updates ← Learner

$$m_{\mathcal{O}}^{\text{One-Shot}}(x) = \sum_{i=1}^{N} o_i(x), \qquad m_{\mathcal{O}}^{\text{DARTS}}(x) = \sum_{i=1}^{N} p_i o_i(x) = \sum_{i=1}^{N} \frac{\exp(\alpha_i)}{\sum_j \exp(\alpha_j)} o_i(x)$$

Cai, Han, Ligeng Zhu, and Song Han. "Proxylessnas: Direct neural architecture search on target task and hardware." *arXiv preprint arXiv:1812.00332* (2018).

NYU SAI LAB

ProxylessNAS

(1) Update weight parameters

fmap in memory
fmap not in memory

(2) Update architecture parameters

$$\mathbb{E}[\text{Latency}] = \alpha \times F(\text{conv\_3x3}) + \\ \beta \times F(\text{conv\_5x5}) + \\ \sigma \times F(\text{identity}) + \\ \dots\dots \\ \zeta \times F(\text{pool\_3x3})$$

$$\mathbb{E}[\text{latency}] = \sum_i \mathbb{E}[\text{latency}_i]$$

$$Loss = Loss_{CE} + \lambda_1 ||w||_2^2 + \lambda_2 \mathbb{E}[\text{latency}]$$

NYU SAI LAB

Presentations

- Learning from multiple teacher networks (Siddhant)
- Chain-of-thought prompting elicits reasoning in large language models (Rutuja)
- MnasNet: Platform-Aware Neural Architecture Search for Mobile (Shawn)
- FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search (Monish)

NYU SAI LAB

Topics

- Learnable adaptive quantization scheme
- Distillation
- Neural architecture search (NAS)
- Low-rank factorization

Low Rank Optimization for DNN Efficiency

- Weight tensors can be decomposed into:

$r < m$ and n

- We can train the W_1 and W_2 in the DNN instead of W.
- Less storage is required.

Singular Value Decomposition

$$W = U \times R \times V^T$$

with W being $m \times n$, U being $m \times r$, R being $r \times r$, and V^T being $r \times n$.

- W: Input matrix
  - $m \times n$ matrix
- U: Left singular vectors
  - $m \times r$ matrix
- R: Singular value matrix
  - $r \times r$ diagonal matrix, r is the rank of W
- V: Right singular vectors
  - $n \times r$ matrix

Before: mn
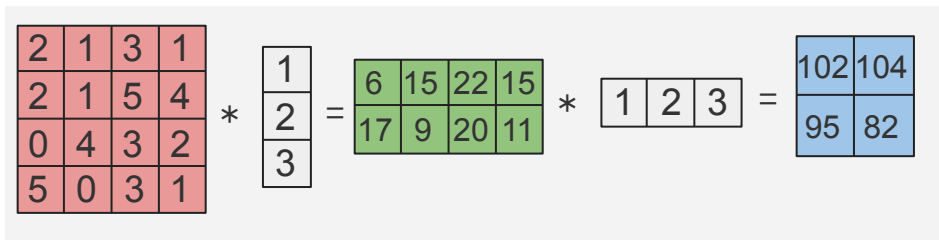After: mr + rn

Conv2d with Low-rank Weight Filter

Step 1

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Step 2

$$\begin{bmatrix} 2 & 1 & 3 & 1 \\ 2 & 1 & 5 & 4 \\ 0 & 4 & 3 & 2 \\ 5 & 0 & 3 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix} = \begin{bmatrix} 102 & 104 \\ 95 & 82 \end{bmatrix}$$

#MAC = 36+9 = 45

storage = 9

$$\begin{bmatrix} 2 & 1 & 3 & 1 \\ 2 & 1 & 5 & 4 \\ 0 & 4 & 3 & 2 \\ 5 & 0 & 3 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 & 15 & 22 & 15 \\ 17 & 9 & 20 & 11 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 102 & 104 \\ 95 & 82 \end{bmatrix}$$
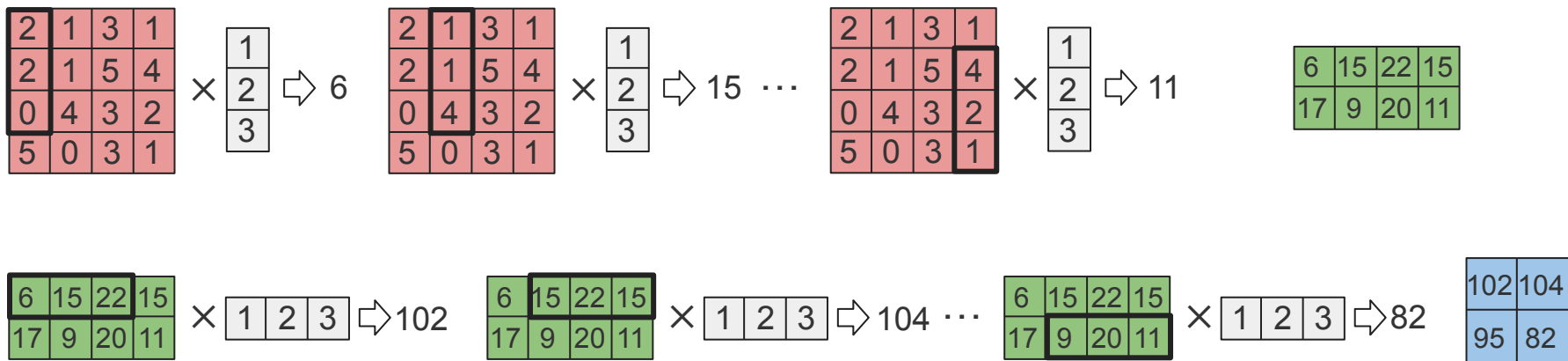
#MAC = 24+12 = 36

storage = 6

NYU SAI LAB
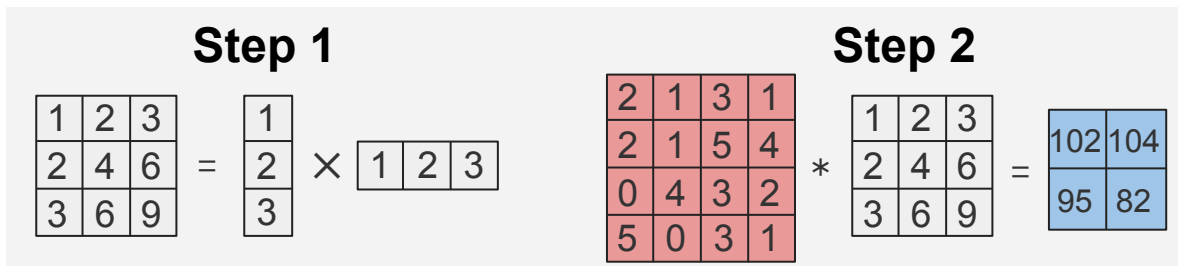
Conv2d with Low-rank Weight Filter

#MAC = 24+12 = 36

storage = 6

Other Types of Decomposition Methods

- CP decomposition
  - Lebedev, Vadim, et al. "Speeding-up convolutional neural networks using fine-tuned cp-decomposition." *arXiv preprint arXiv:1412.6553* (2014).
- Tucker decomposition
  - Kim, Yong-Deok, et al. "Compression of deep convolutional neural networks for fast and low power mobile applications." *arXiv preprint arXiv:1511.06530* (2015).
- Each method has its own tradeoff between computational efficiency and Accuracy.

Lebedev, Vadim, et al. "Speeding-up convolutional neural networks using fine-tuned cp-decomposition." *arXiv preprint arXiv:1412.6553* (2014).

NYU SAI LAB

Code Demo

https://colab.research.google.com/drive/1GnKFVRRchz8F2qGxhrUWQxCTeCheH6O0?usp=sharing

NYU SAI LAB