Q2AI: A Quick Course to Quick AI

A PRICAI 2023 Tutorial

The Team

Haryo Akbarianto Wibowo

Rendi Chevi

Alham Fikri Aji

Radityo Eko Prasojo

Foundation Models Are Everywhere

MBZUAI

NIXTLA

IN BETA

TimeGPT

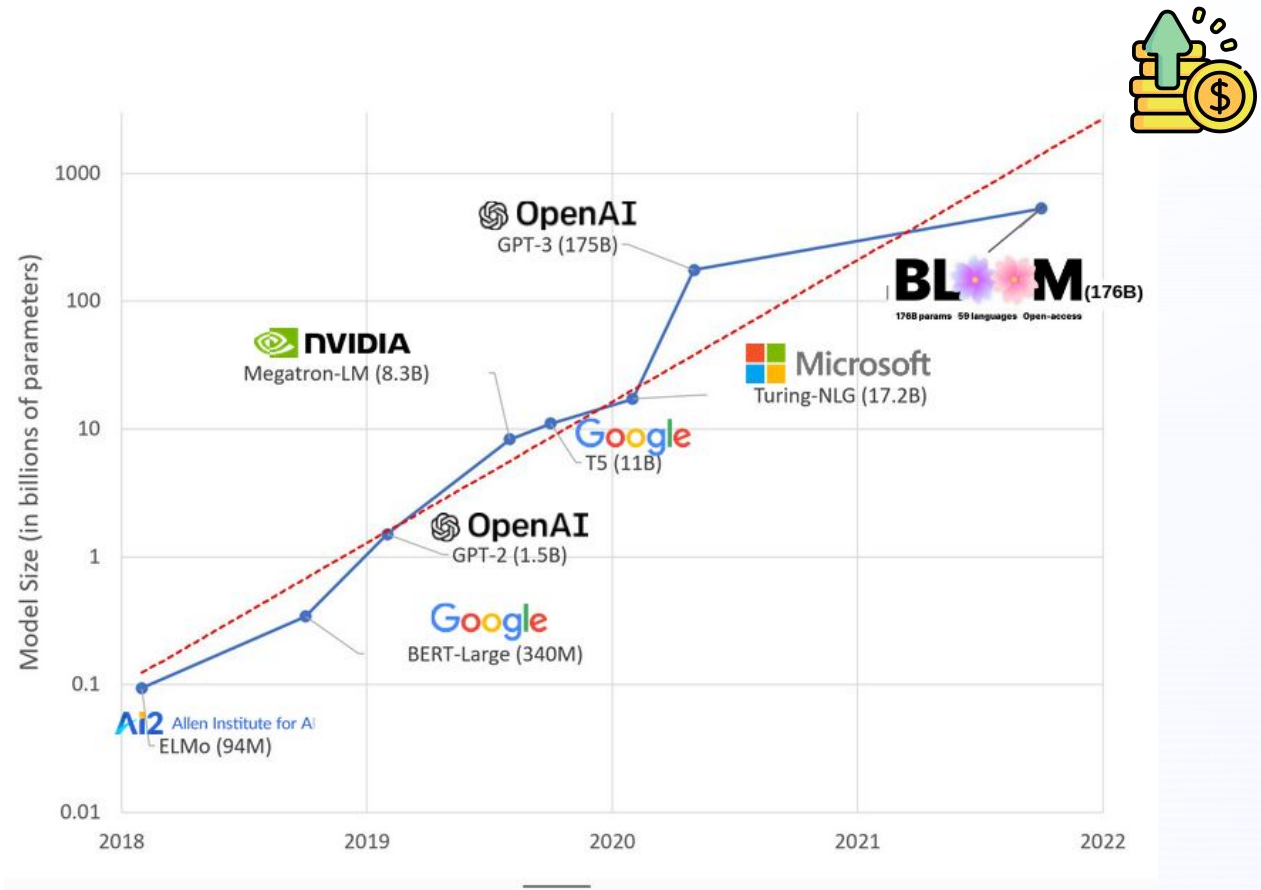
Generative AI for time series

Google AI

Chirp

El ágil zorro colorado salta sobre el perro perezoso

Foundation Model Size Growth

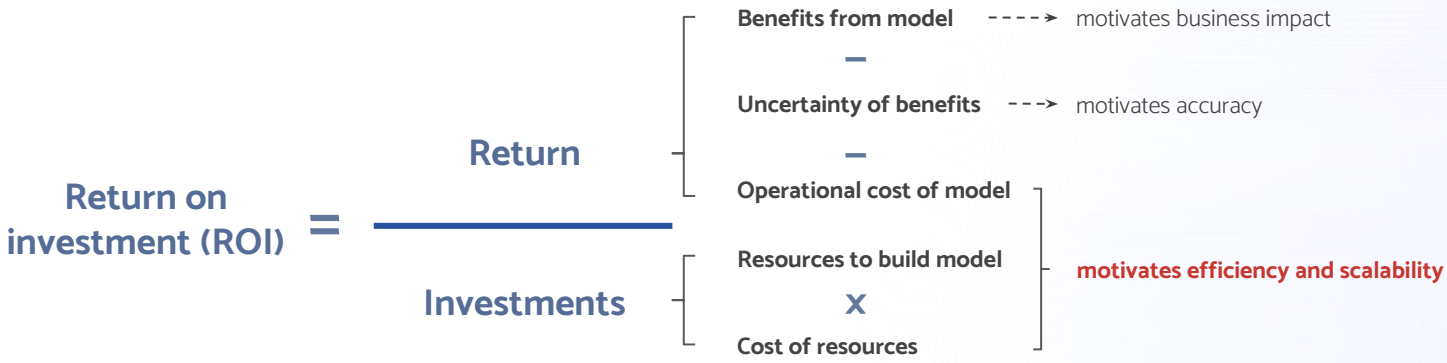


The Open AI Case

($540,000,000)	Loss in 2022 (doubled from 2021)
$700,000	Daily cost of ChatGPT
-21%	MAU From May 2023 to July 2023

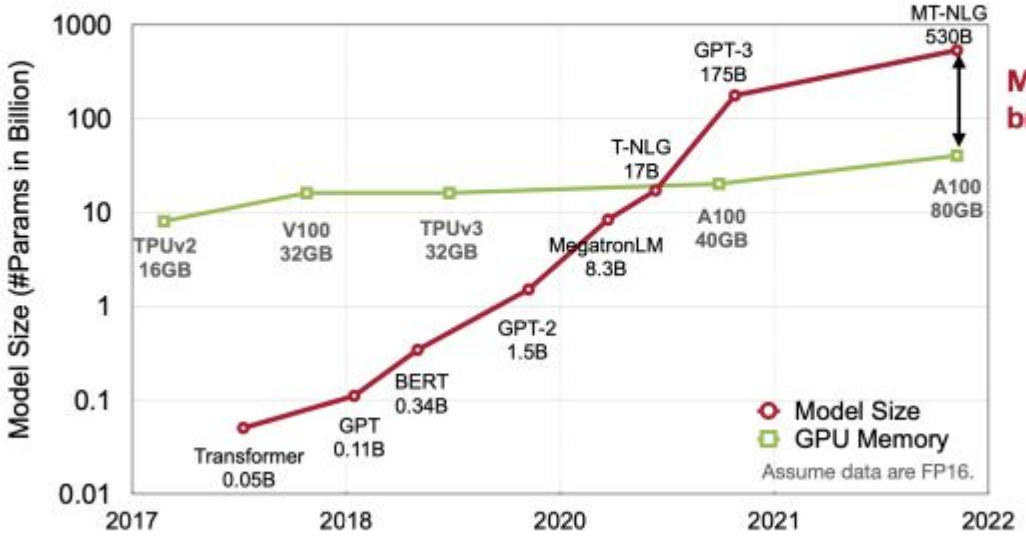
The Economic Times, 2023.

AI Return of Investment (ROI)



Adapted from PwC, 2021.

Large language Models



Model compression bridges the gap.

Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey [Deng *et al.*, IEEE 2020]

How to build scalable and efficient AIs?

- Knowledge Distillation (KD)
- Parameter-Efficient Fine Tuning (PEFT)
- Quantization
- Others
  - Neural Architecture Search
  - ONNX Runtime

Xu, C., & McAuley, J. (2023). A Survey on Model Compression and Acceleration for Pretrained Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(9), 10566-10575. https://doi.org/10.1609/aaai.v37i9.26255

Knowledge Distillation

Llama 2

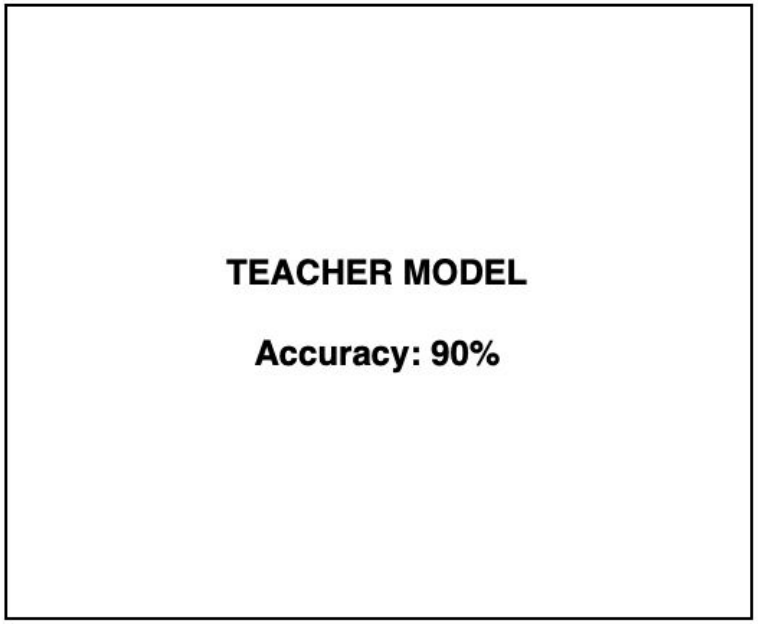
MODEL SIZE (PARAMETERS)	PRETRAINED	FINE-TUNED FOR CHAT USE CASES
7B	**Model architecture:** **Pretraining Tokens:** 2 Trillion **Context Length:** 4096	**Data collection for helpfulness and safety:** **Supervised fine-tuning:** Over 100,000 **Human Preferences:** Over 1,000,000
13B		
70B		

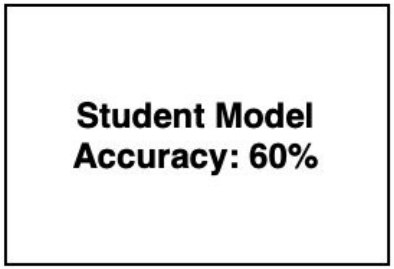
What is Knowledge Distillation (KD)?

Transferring knowledge from **a large model (teacher)** to a smaller, **simpler model (student).**

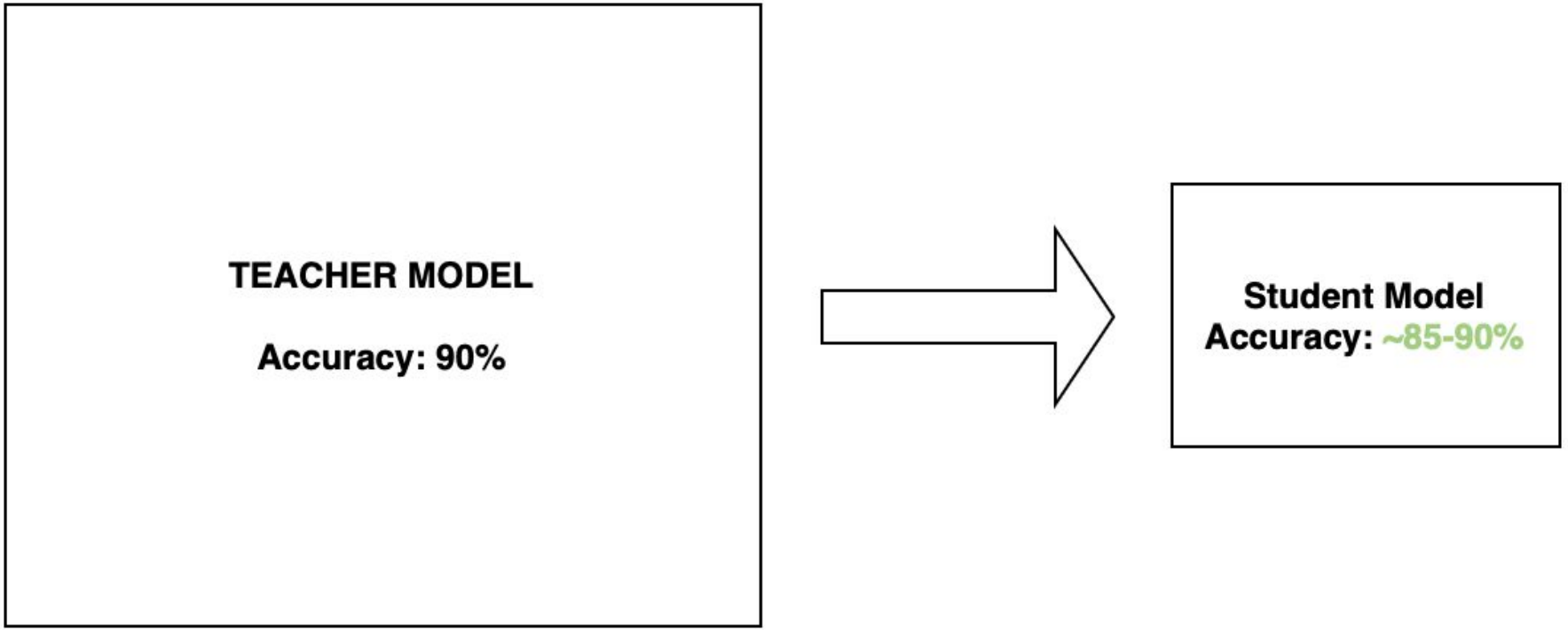
Student is supervised by the **teacher(s)**

What is Knowledge Distillation (KD)?





What is Knowledge Distillation (KD)?



Importance and Applications

Speed: Faster inference times.

System	#Params	#FLOPs	Speedup	MNLI-(m/mm)	QQP
$\text{BERT}_{\text{BASE}}$ (Teacher)	109M	22.5B	1.0x	83.9/83.4	71.1
$\text{BERT}_{\text{TINY}}$	14.5M	1.2B	9.4x	75.4/74.9	66.5
$\text{BERT}_{\text{SMALL}}$	29.2M	3.4B	5.7x	77.6/77.0	68.1
BERT_4-PKD	52.2M	7.6B	3.0x	79.9/79.3	70.2
DistilBERT_4	52.2M	7.6B	3.0x	78.9/78.0	68.5
$\text{MobileBERT}_{\text{TINY}}$†	15.1M	3.1B	-	81.5/81.6	68.9
TinyBERT_4 (ours)	14.5M	1.2B	9.4x	**82.5/81.8**	**71.3**

Jiao, Xiaoqi, et al. "Tinybert: Distilling bert for natural language understanding." *arXiv preprint arXiv:1909.10351* (2019).

Importance and Applications

Accessibility: Lightweight models

System	#Params	#FLOPs	Speedup	MNLI-(m/mm)	QQP
$\text{BERT}_{\text{BASE}}$ (Teacher)	109M	22.5B	1.0x	83.9/83.4	71.1
$\text{BERT}_{\text{TINY}}$	14.5M	1.2B	9.4x	75.4/74.9	66.5
$\text{BERT}_{\text{SMALL}}$	29.2M	3.4B	5.7x	77.6/77.0	68.1
BERT_4-PKD	52.2M	7.6B	3.0x	79.9/79.3	70.2
DistilBERT_4	52.2M	7.6B	3.0x	78.9/78.0	68.5
$\text{MobileBERT}_{\text{TINY}}$†	15.1M	3.1B	-	81.5/81.6	68.9
TinyBERT_4 (ours)	14.5M	1.2B	9.4x	**82.5/81.8**	**71.3**

Jiao, Xiaoqi, et al. "Tinybert: Distilling bert for natural language understanding." *arXiv preprint arXiv:1909.10351* (2019).

Example Teacher and Student

RoBERTa Large (335M) -> **RoBERTa base** (110M)

BERT Large (335M) **-> LSTM** (40M)

and so on!

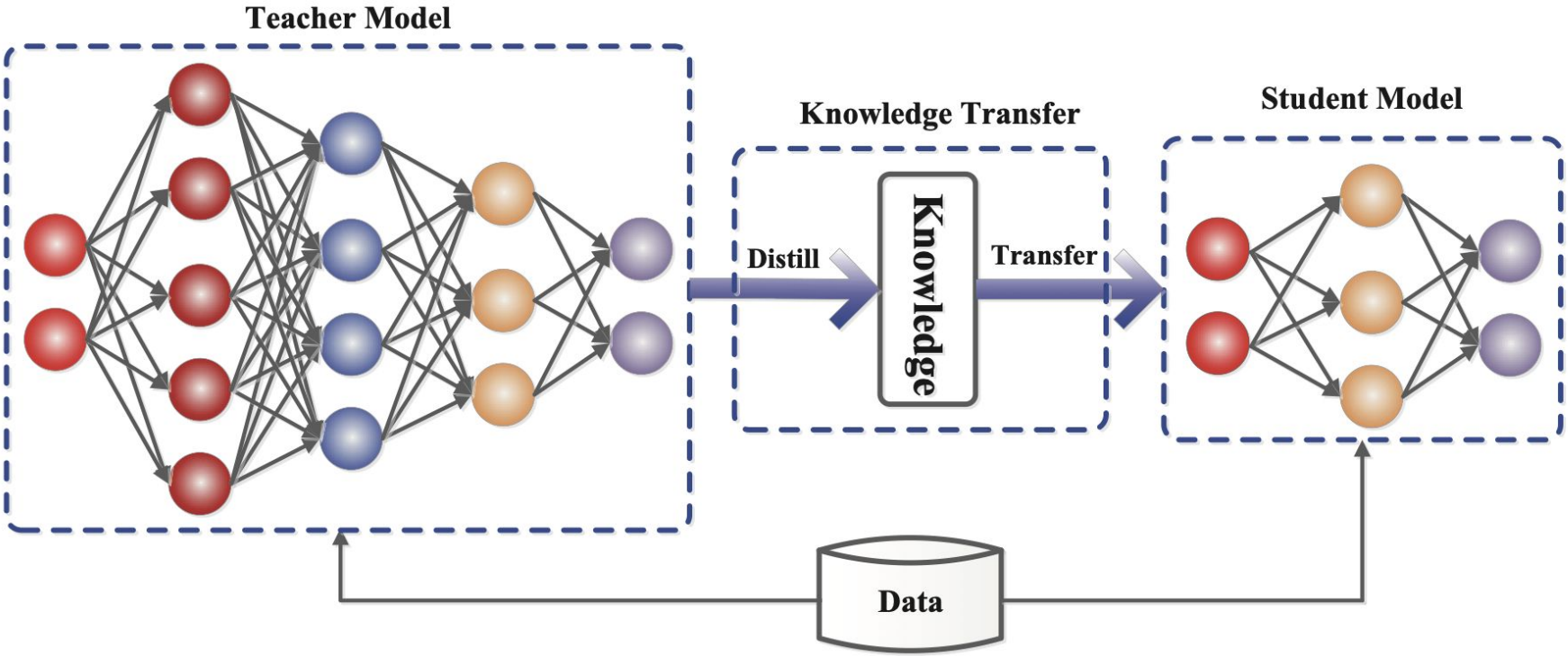
Knowledge?

Imagine that you are a teacher, what do you want to transfer, given a knowledge (e.g.: Math, Biology, and so on)?

Knowledge?

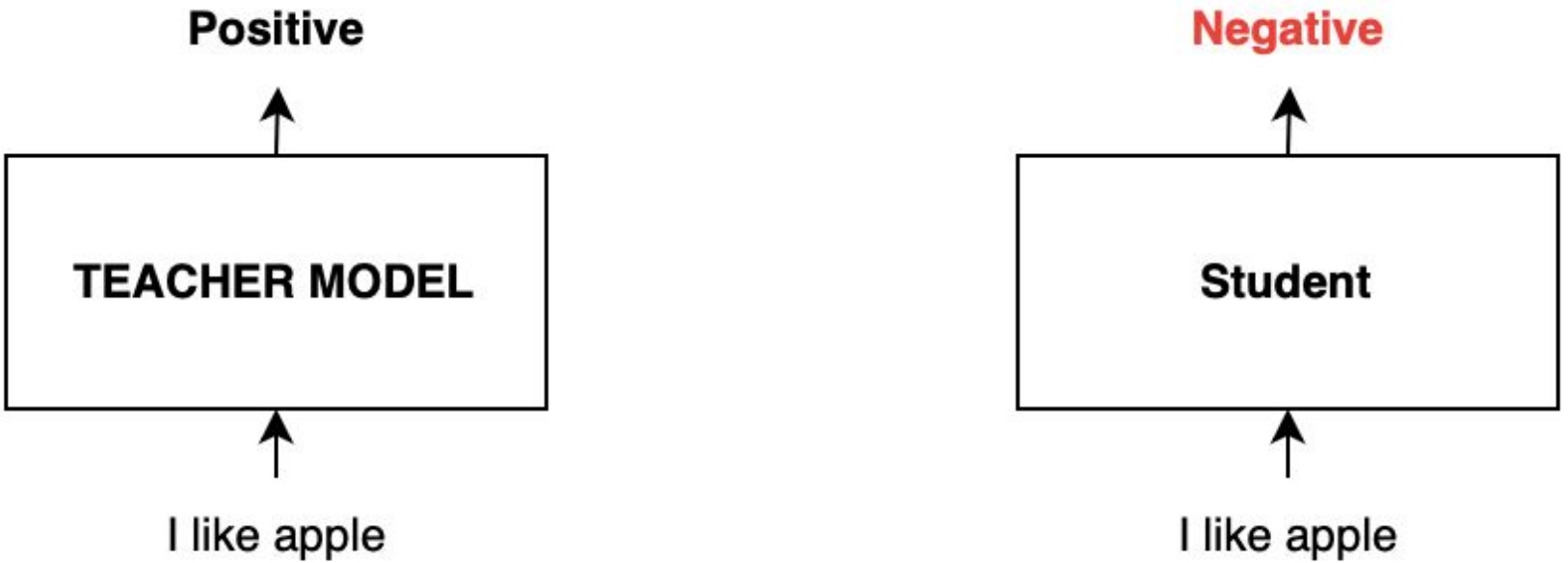
Imagine that you are a teacher, what do you want to transfer, given a knowledge (e.g.: Math, Biology, and so on)?

What kind of knowledge does the teacher transfer?

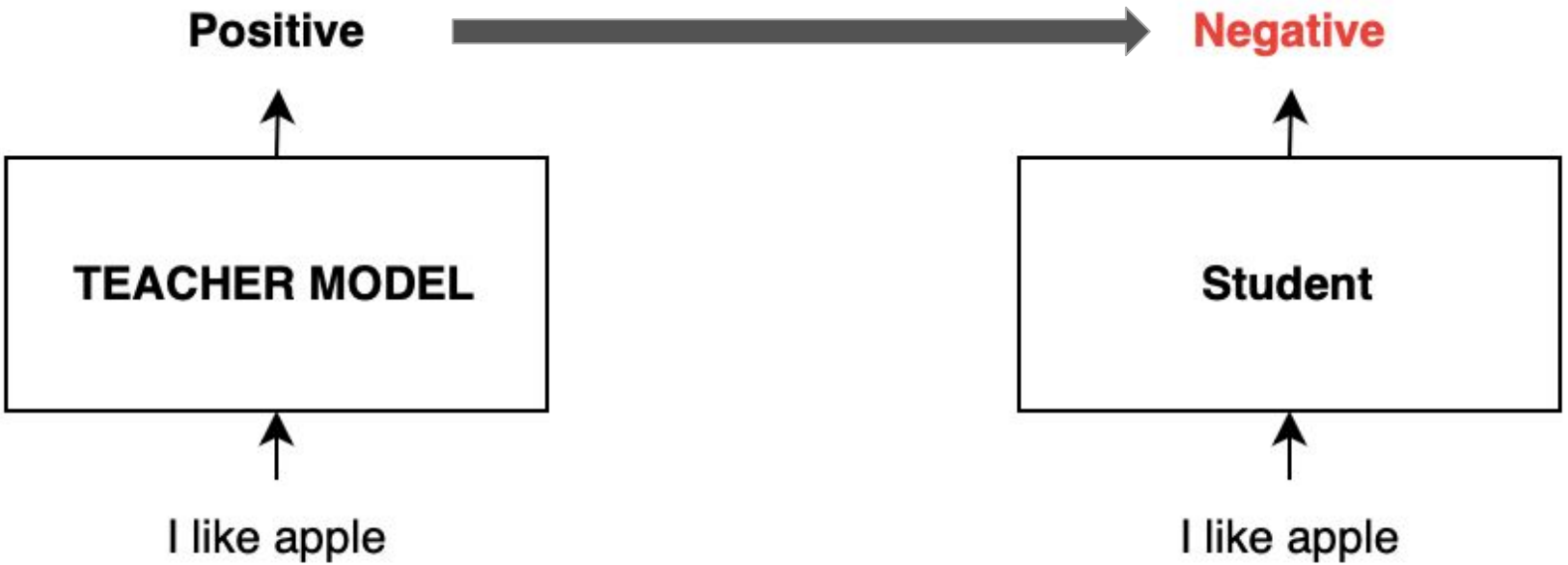


Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, *129*, 1789-1819.

Knowledge?


Positive
TEACHER MODEL
I like apple
Negative
Student
I like apple

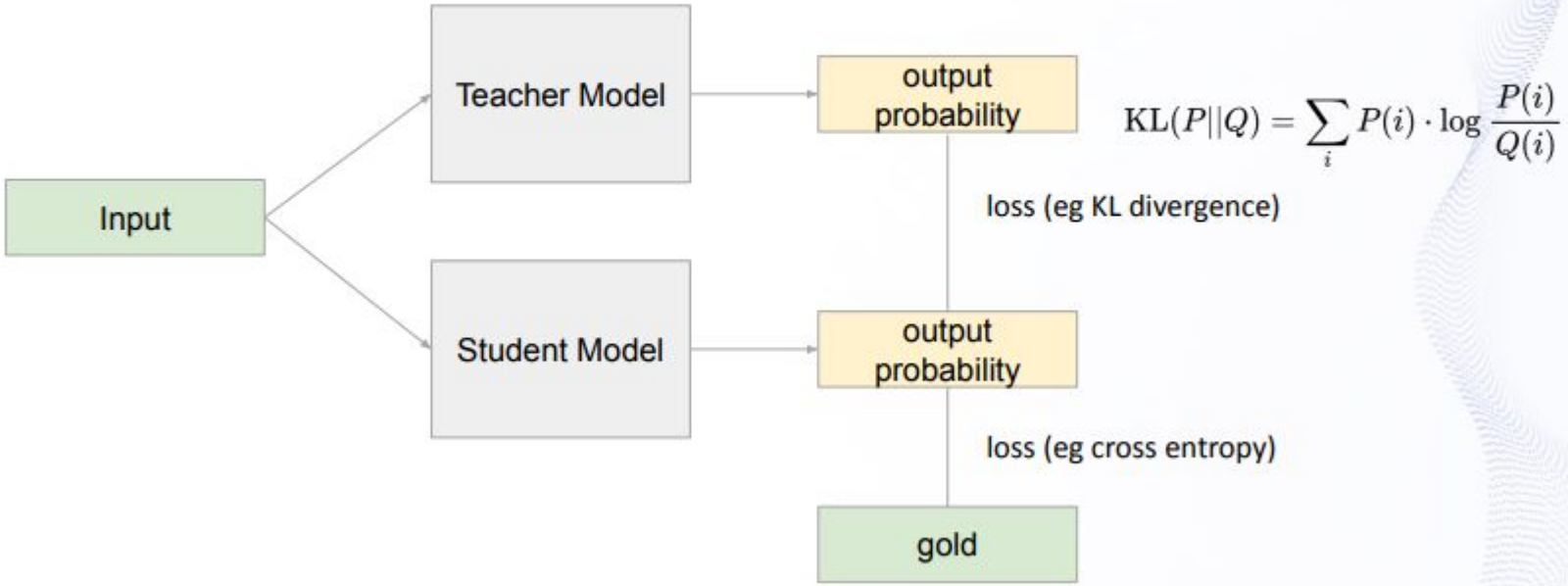
Knowledge?


Positive
Negative
TEACHER MODEL
Student
I like apple
I like apple

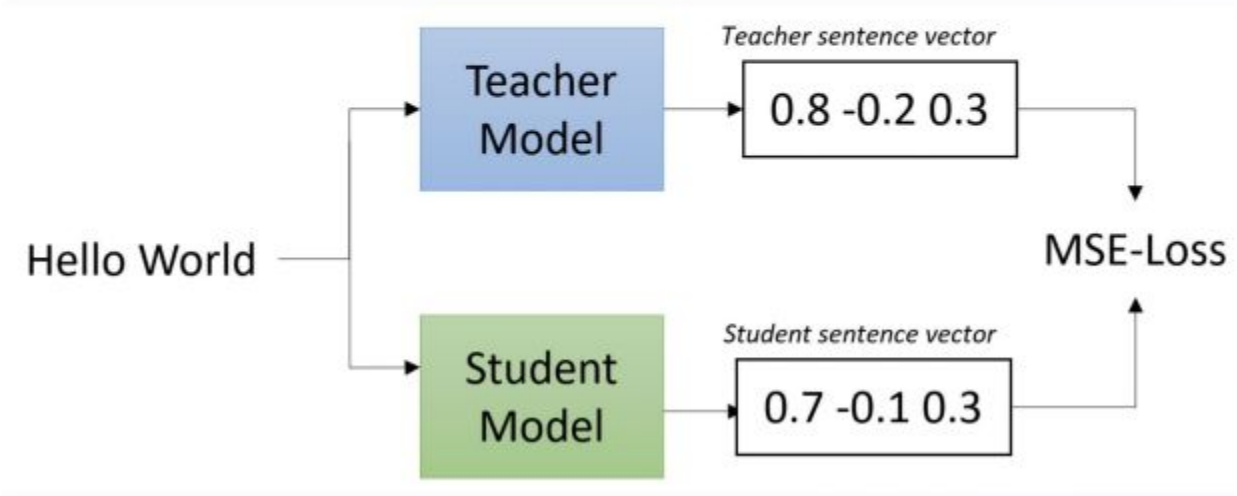
Knowledge Distillation Distillation Techniques

- Soft Target Distillation
- Hard Target Distillation
- Feature Matching

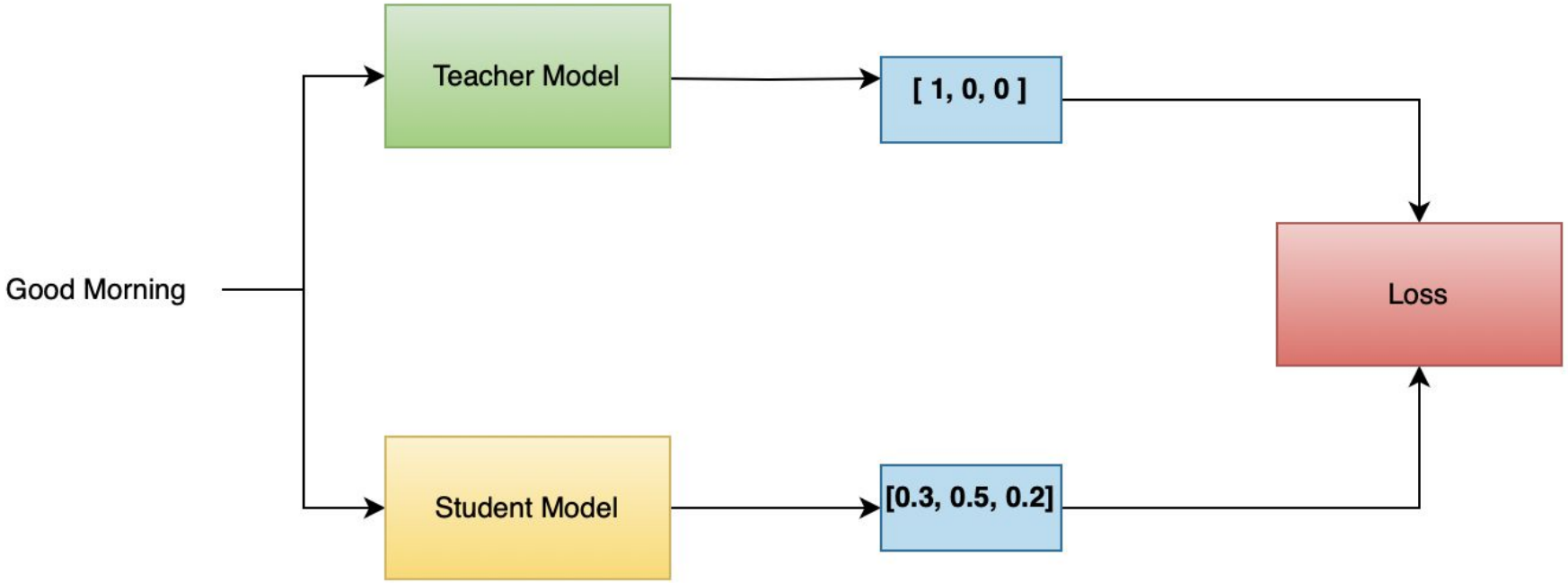
Soft Target Distillation



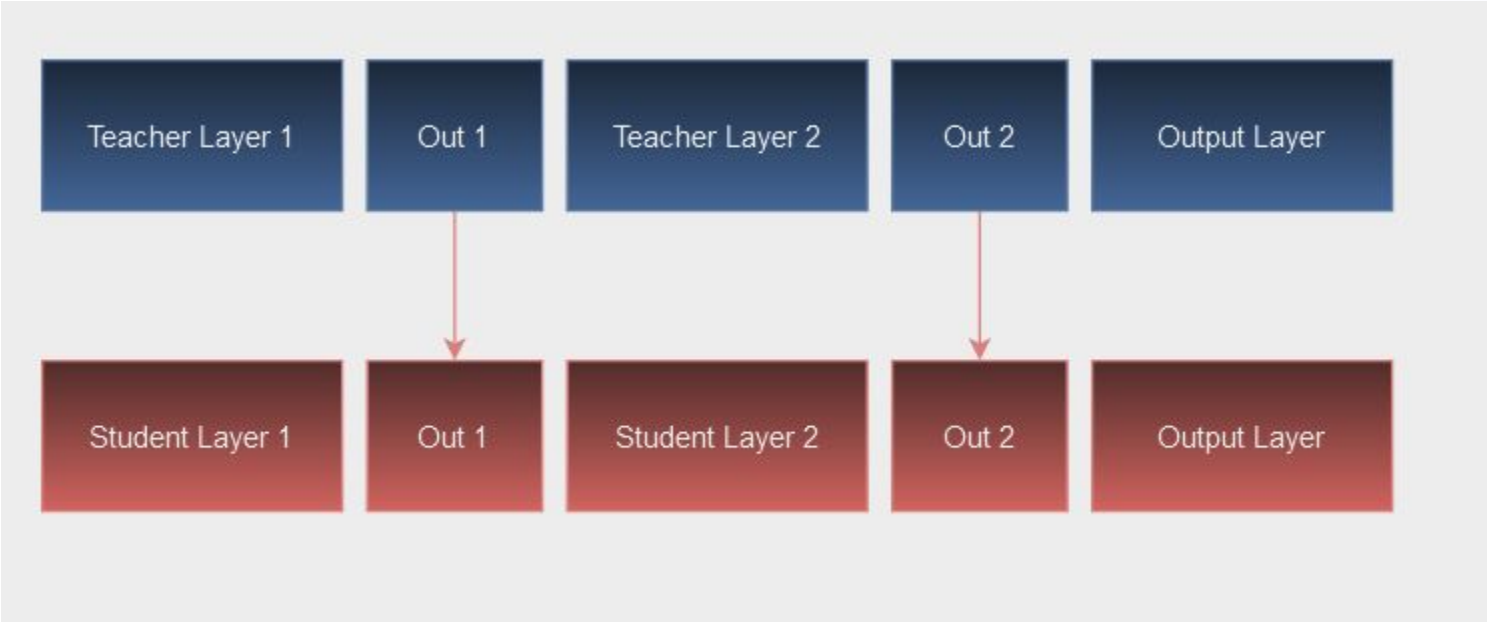
Soft Target Distillation



Hard Target Distillation



Feature Matching



Loss Functions in Knowledge Distillation

Quantifies the difference between teacher and student outputs.

Softmax & Temperature: Using temperature to smooth teacher's softmax outputs, making them more usable as soft targets.

$$p_i = \frac{exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

Loss Functions in Knowledge Distillation

Quantifies the difference between teacher and student outputs.

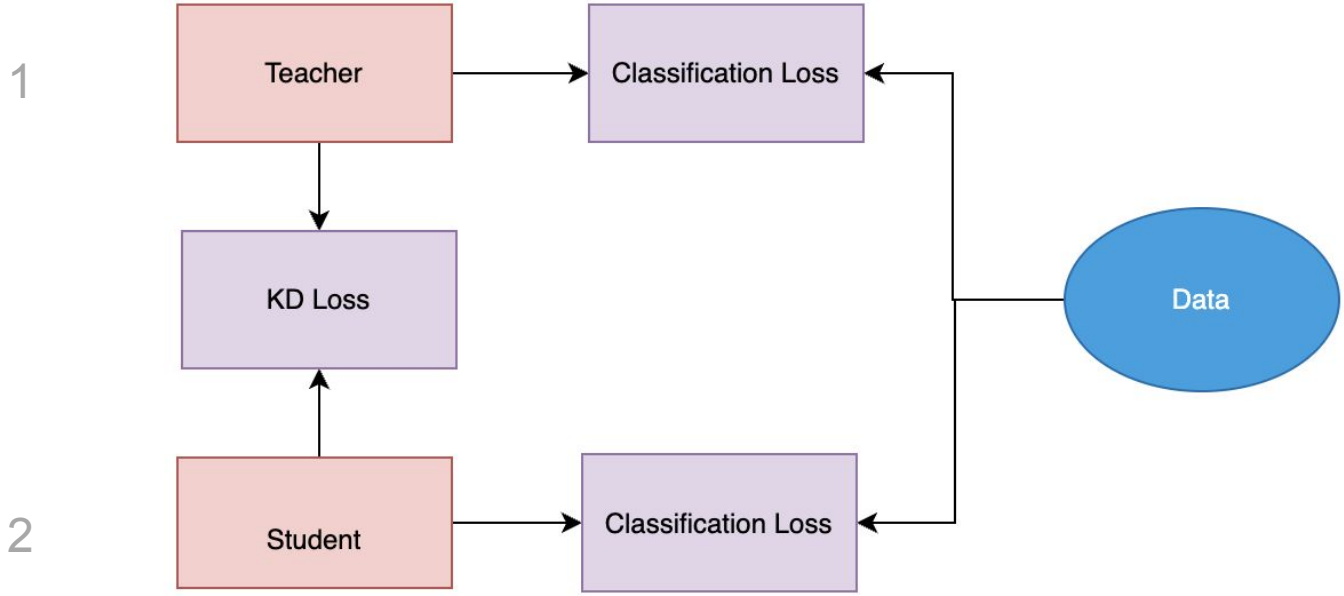
Can use MSE or KL-Divergence:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2.$$

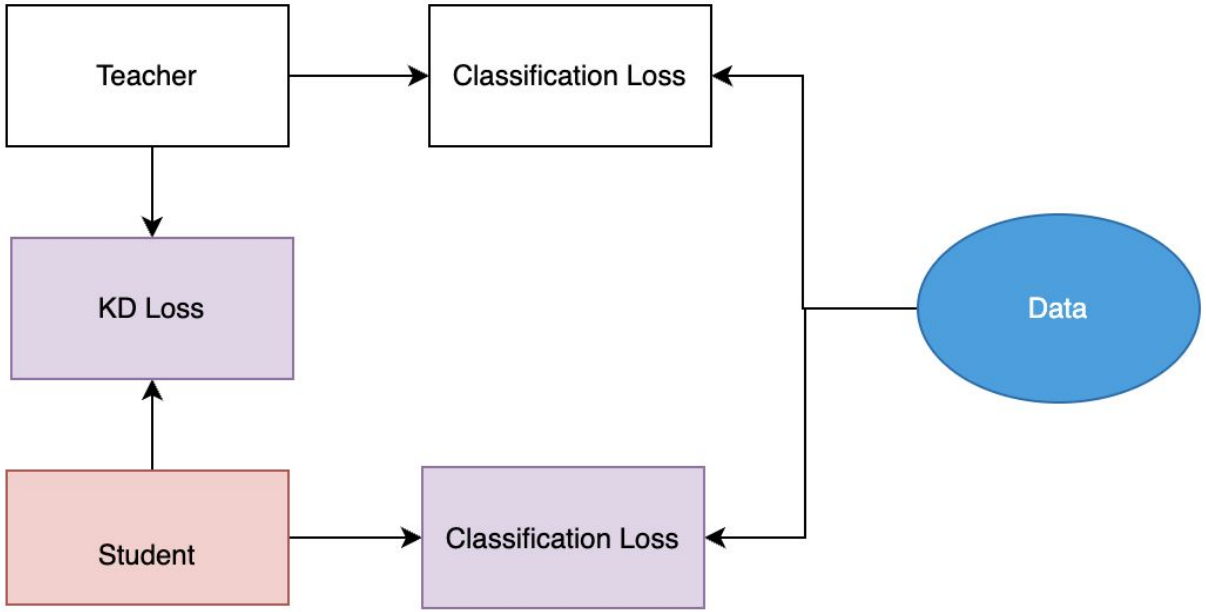
$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Online Distillation

Teacher and Student are trained simultaneously!

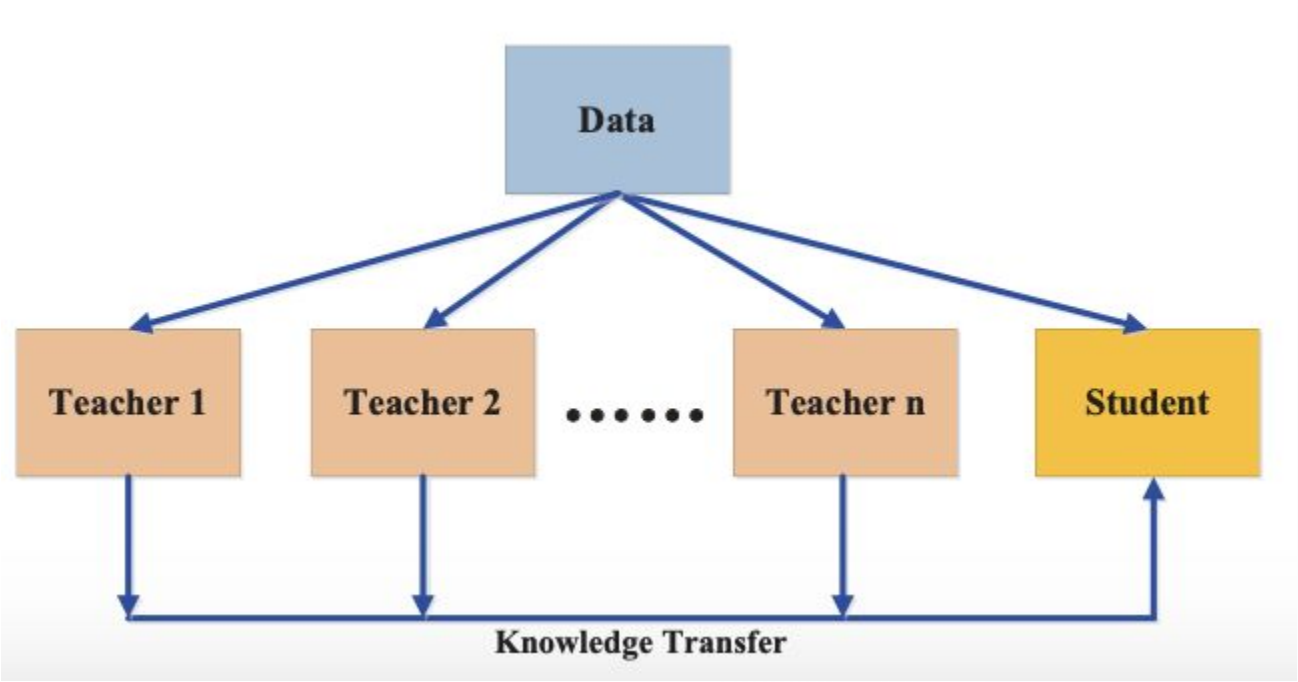


Offline Distillation



Variation of Knowledge Distillation

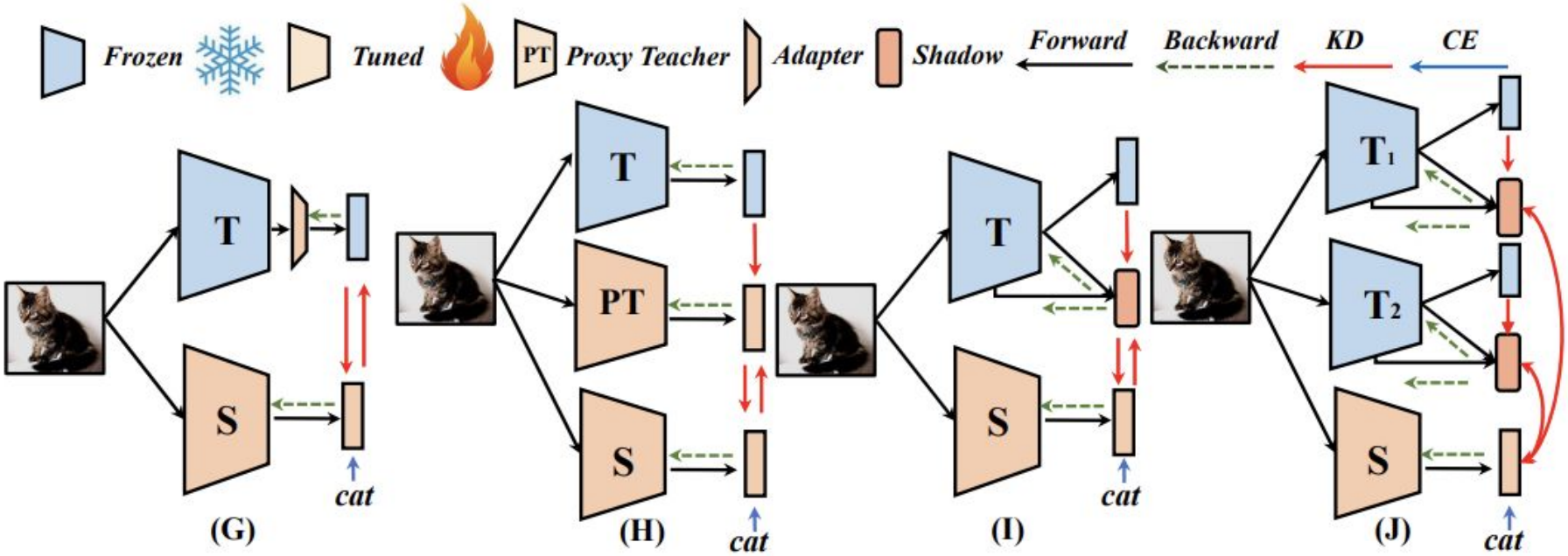
Multi Teacher



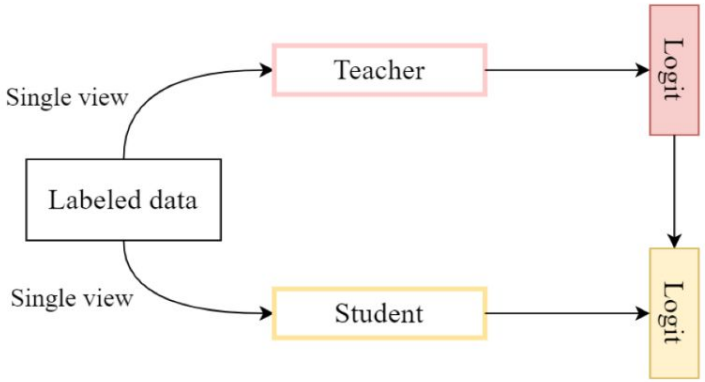
Variation of Knowledge Distillation

Shadow Knowledge Distillation: Bridging Offline and Online Knowledge Transfer

Lujun Li[1,2,✉], **Jin Zhe**[1,✉]
School of Artificial Intelligence, Anhui University, China
Chinese Academy of Science, China
lilujunai@gmail.com; jinzhe@ahu.edu.cn



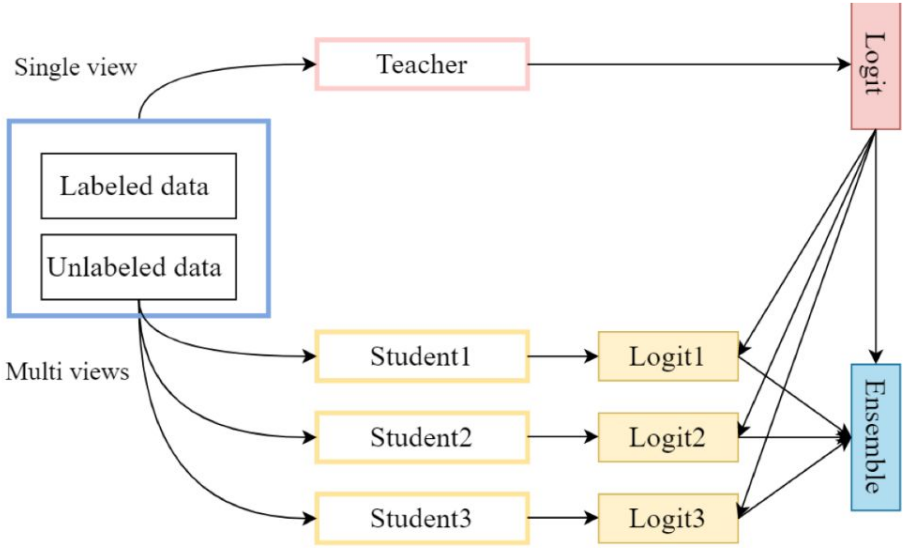
Variation of Knowledge Distillation



(a) One-Teacher and One-Student Knowledge Distillation in Most Previous Studies

One-Teacher and Multiple-Student Knowledge Distillation on Sentimen Classification

Xiaoqin Chang[1] **Sophia Yat Mei Lee**[2] **Suyang Zhu**[1*]
Shoushan Li[1] **Guodong Zhou**[1]
[1]Natural Language Processing Lab, Soochow University, China
[2]Department of Chinese and Bilingual Studies, The Hong Kong Polytechnic University
xqchang@stu.suda.edu.cn
{syzhu, samlee, gdzhou}@suda.edu.cn
ym.lee@polyu.edu.hk



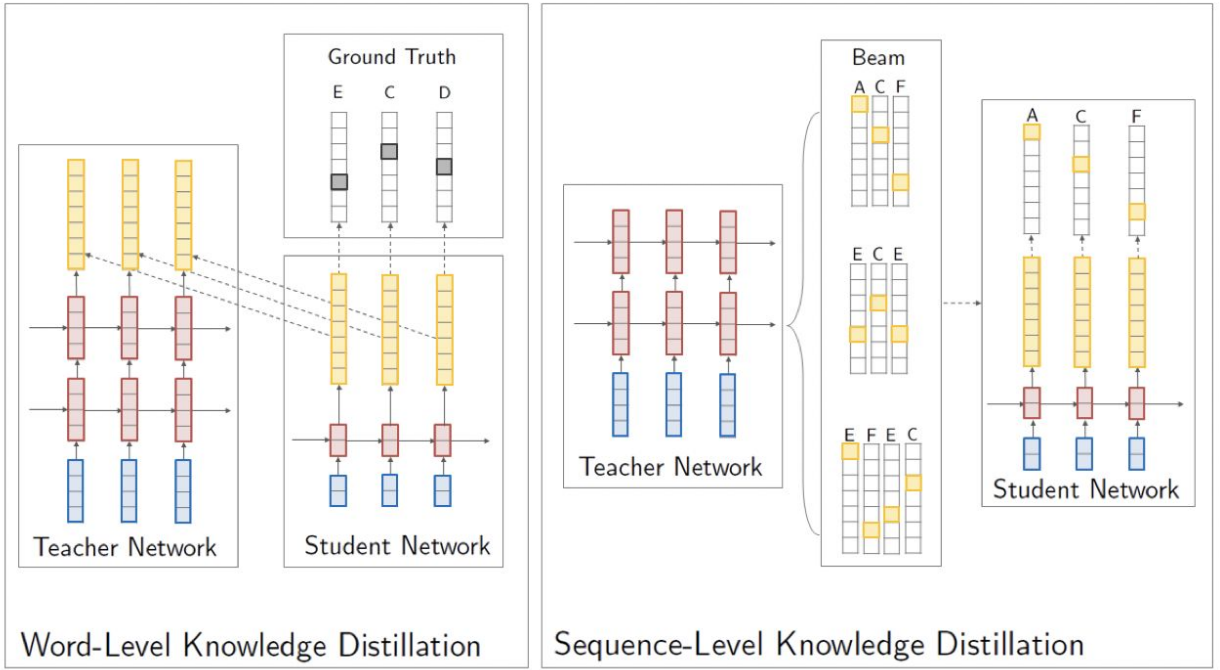
(b) One-Teacher and Multiple-Student Knowledge Distillation in Our Study

Knowledge Distillation: Sequence?

KD is not NLP-specific.

But what if I want to distill from generative model (decoder)?

Sequence-Level KD [Kim et al., 2016]

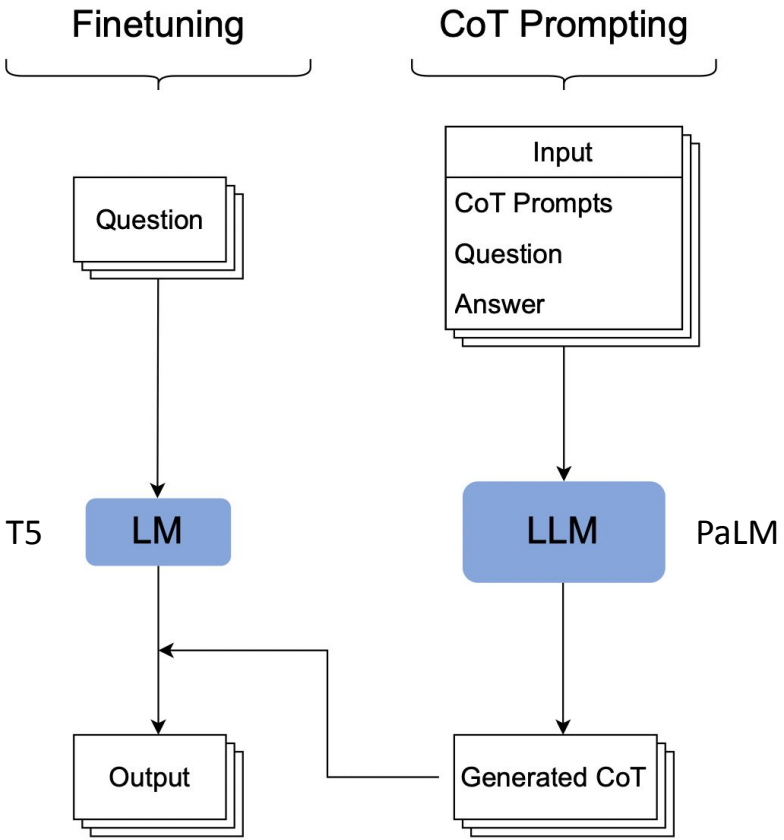


In sequence-level knowledge distillation (center) the student network is trained on the output from beam search of the teacher network that had the highest score (ACF).

Sequence-Level KD [Kim et al., 2016]

Model	$\text{BLEU}_{K=1}$	$\Delta_{K=1}$
English → German WMT 2014		
Teacher Baseline 4×1000 (Params: 221m)	17.7	–
Baseline + Seq-Inter	19.6	+1.9
Student Baseline 2×500 (Params: 84m)	14.7	–
Word-KD	15.4	+0.7
Seq-KD	18.9	**+4.2**
Baseline + Seq-Inter	18.5	+3.6
Word-KD + Seq-Inter	18.3	+3.6
Seq-KD + Seq-Inter	18.9	**+4.2**
Seq-KD + Word-KD	18.7	+4.0
Seq-KD + Seq-Inter + Word-KD	18.8	+4.1

Sequence-Level KD: Another Example: CoT distillation [Magister et al., 2022]



	Baseline T5 XXL	CoT Finetuned T5 XXL		CoT 8-shot PaLM 540B	
	Acc.	Acc.	Acc. with Calc.	Acc.	Acc. with Calc.
GSM8K	8.11	**21.99**	38.21	56.90	58.60
Dataset Size	6725	5337	5337	-	-
MAWPS	54.15	**70.41**	88.22	93.00	93.66
Dataset Size	1590	1590	1590	-	-
ASDiv	39.64	**42.12**	60.73	73.9	72.6
Dataset Size	1844	1544	1544	-	-

Sequence-Level KD: Application and Issue

- We only learn from the output of the teacher, we don’t need labelled data!
- We can train students with more data!
- We don’t need access to teacher’s probability distribution, can distill from proprietary models!

Tips and Trick

- Can train KD with **UNLABELLED DATA**
- If using offline KD, we can precompute the teacher's probability (only label), then feed it to the student!

Challenges and Limitations

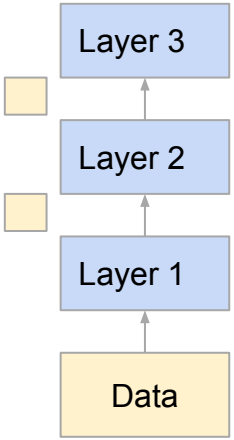
- Orchestration between Teacher and Student
- Advance KD might require Heavy Computation on training (Online Distillation / Multiple Teacher)
- Need more data (experimental) to closely match teacher’s performance!

Let’s get our hand dirty!

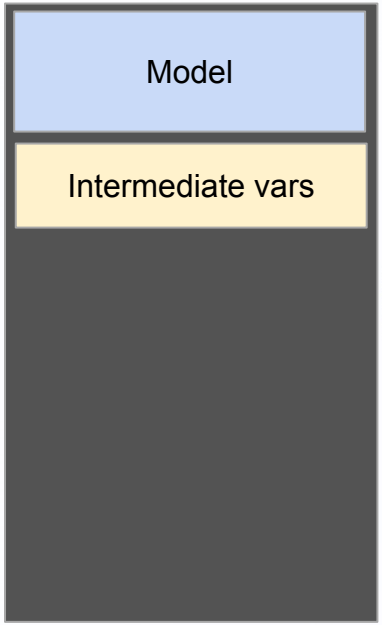
Parameter-Efficient Fine-Tuning

Recap: what is going on during training?

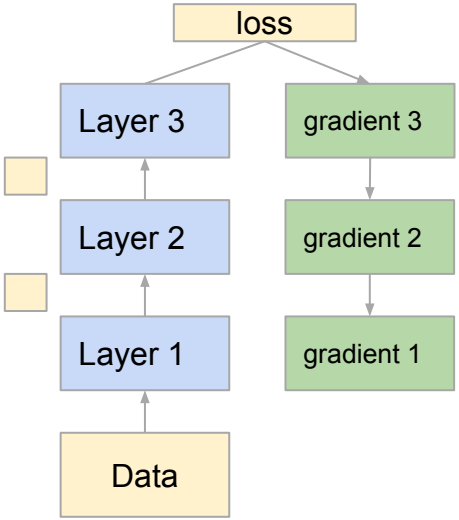
Recap: what is going on during training?



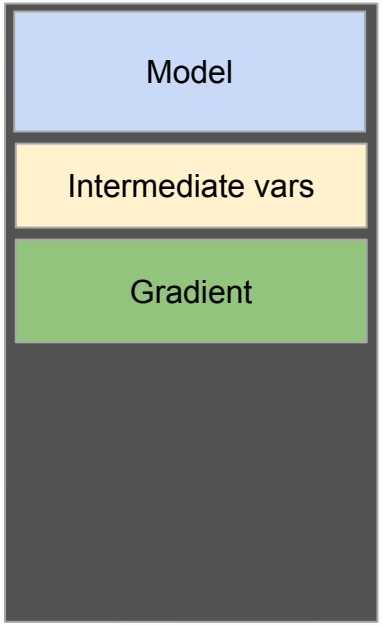
your GPU:



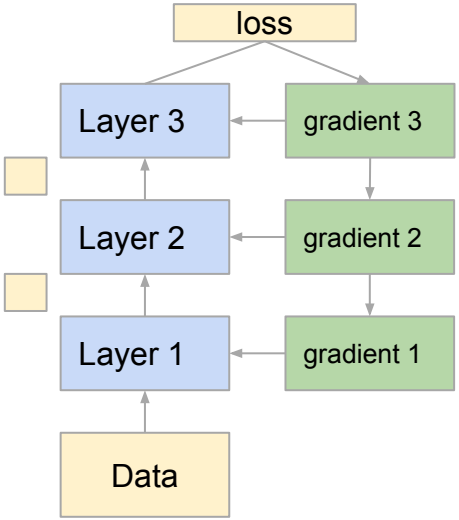
Recap: what is going on during training?



your GPU:



Recap: what is going on during training?



for $t = 1$ **to** $\ldots$ **do**

  if *maximize* :

    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$

  else

    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

  $\theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$

  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$

  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$

  $\widehat{m_t} \leftarrow m_t / \left(1-\beta_1^t\right)$

  $\widehat{v_t} \leftarrow v_t / \left(1-\beta_2^t\right)$

  if *amsgrad*

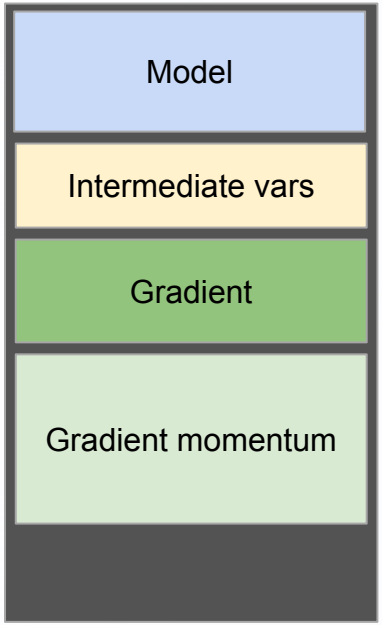
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$

    $\theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / \left(\sqrt{\widehat{v_t}^{max}} + \epsilon\right)$

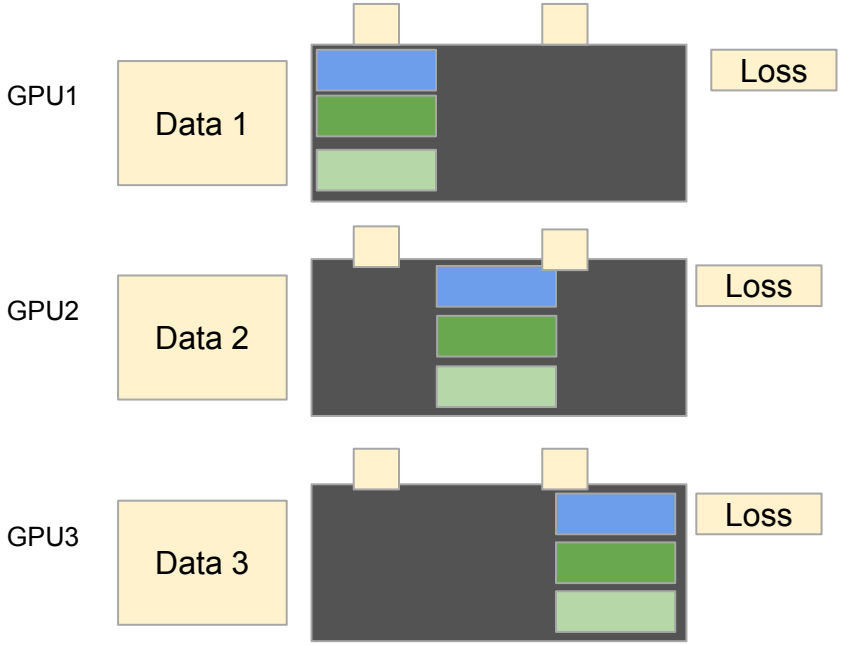
  else

    $\theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / \left(\sqrt{\widehat{v_t}} + \epsilon\right)$

your GPU:



Recap: what is going on during training?

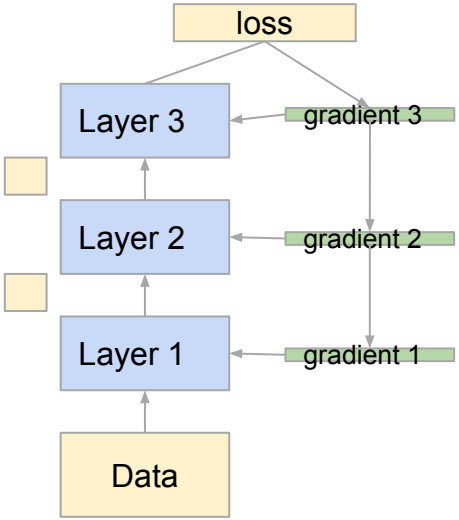


Sharded memory reduces the usage per GPU by the number of device.

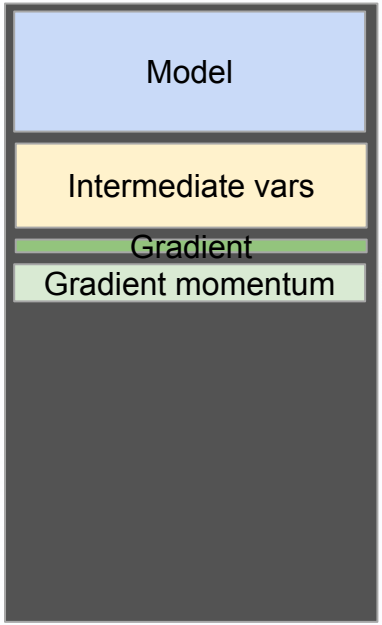
only works if you’re rich!

Parameter-Efficient Finetuning

If we do not train all of the model weight, then we save space on gradients

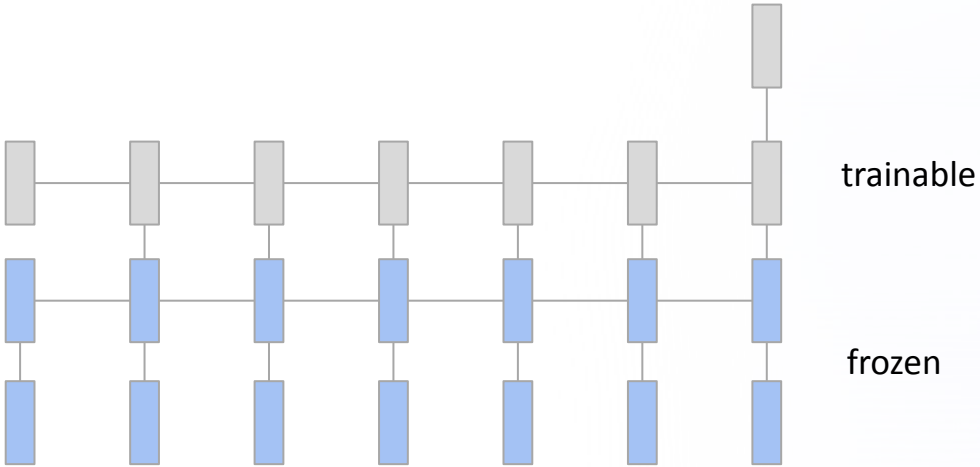


your GPU:



Naive PEFT: Training parts of your model

Early days of PLM like GPT-2/BERT, people often only finetune some of the top layers.



Naive PEFT: Training parts of your model

+ Simple and straightforward.
- Is usually degrade performance

BitFit: Only Train Biases [Zaken et al., 2021]

Many matrix operation in deep-learning incorporate additive bias after matrix multiplication, eg:

output = Wx + b

This bias is extremely small in size (since it's just a vector) vs full matrix.
We freeze everything except the biases.

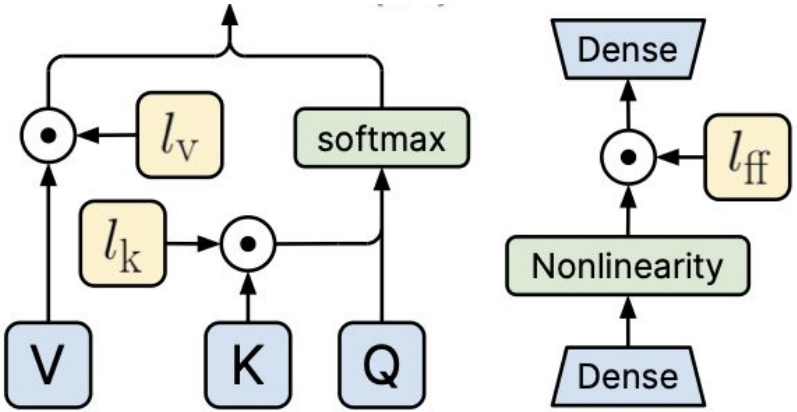
	Method	%Param	QNLI	SST-2	MNLI_m	MNLI_mm	CoLA	MRPC	STS-B	RTE	QQP	Avg.
BB	Full-FT	100%	**90.7±0.2**	92.0±0.4	**83.5±0.1**	**83.7±0.3**	56.4±0.9	89.0±1.0	88.9±0.7	70.5±0.6	**87.1±0.1**	82.3
BB	BitFit	0.09%	90.2±0.2	**92.1±0.3**	81.4±0.2	82.2±0.2	**58.8±0.5**	**90.4±0.5**	**89.2±0.2**	**72.3±0.9**	84.0±0.2	**82.4**
BL	Full-FT	100%	**91.7±0.1**	**93.4±0.2**	**85.5±0.4**	**85.7±0.4**	62.2±1.2	90.7±0.3	90.0±0.4	71.9±1.3	**87.5±0.4**	84.1
BL	BitFit	0.08%	91.4±2.4	93.2±0.4	84.4±0.2	84.8±0.1	**63.6±0.7**	**91.7±0.5**	**90.3±0.1**	**73.2±3.7**	85.4±0.1	**84.2**
Ro	Full-FT	100%	**92.3±0.2**	**94.2±0.4**	**86.4±0.3**	**86.9±0.3**	61.1±0.8	**92.5±0.4**	90.6±0.2	77.4±1.0	**88.0±0.2**	**85.3**
Ro	BitFit	0.09%	91.3±0.2	93.7±0.1	84.8±0.1	85.2±0.2	**61.8±1.3**	92.0±0.4	**90.8±0.3**	**77.8±1.7**	84.5±0.2	84.6

Table 2: Dev-set results for different base models. **BB**: BERT_BASE. **BL**: BERT_LARGE. **Ro**: $\text{RoBERTa}_\text{BASE}$.

Adding New Weights: $(IA)^3$ [Liu et al., 2022]

Adding a new, trainable multiplicative scale after some of the activation.

Was designed for transformer, so they add this in some parts of the attentions



normal operation:

output = Wx + b

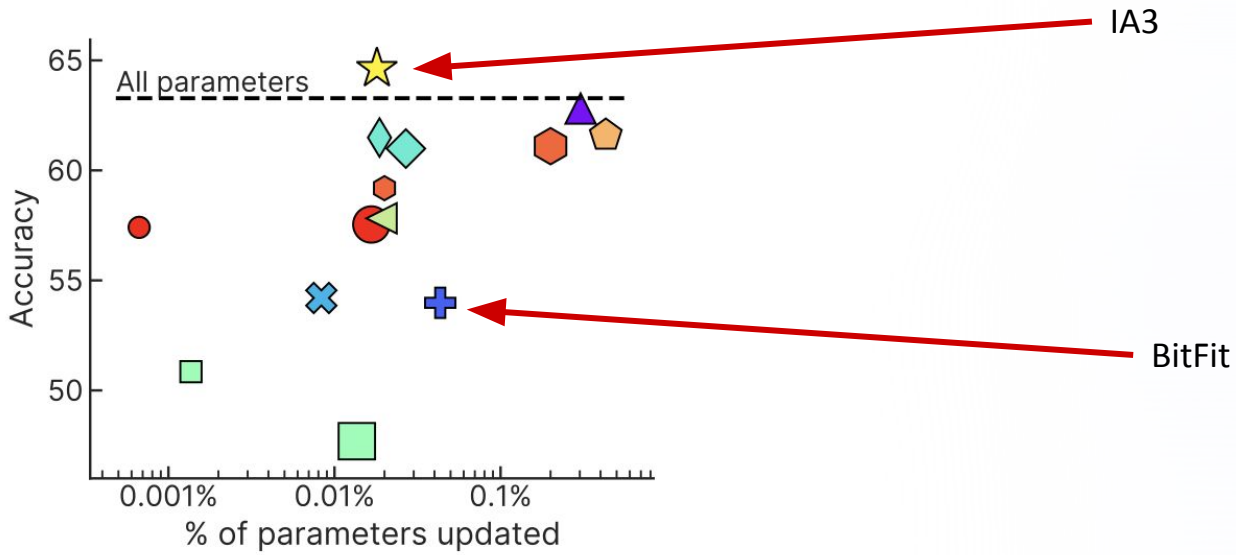
With IA3:

output = (Wx + b) . **l**

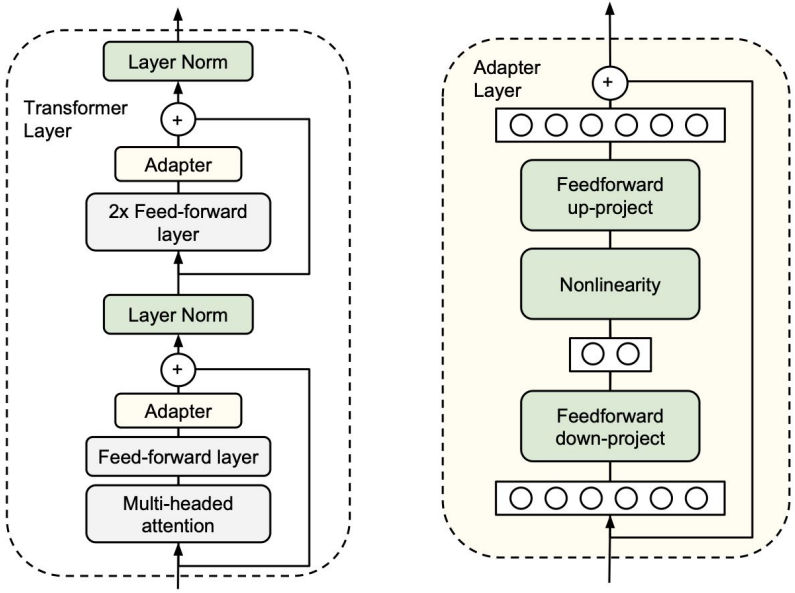
element-wise multiplication.
l just a vector (the same size as b).
Think this like BitFit, but multiplicative

Adding New Weights: $(IA)^3$ [Liu et al., 2022]

Good performance, they claimed better performance than full fine tuning.



Adding New Weights: Adapters [Houlsby et al., 2019]

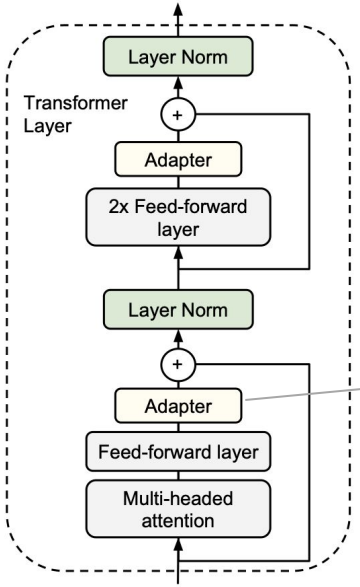


Adapter layer is efficient since it's using 2 FFN layers rather than a single FFN layer.

We down-project the dimension to k << N, before up-project it back again.

So:
$N*k + k*N << N*N$

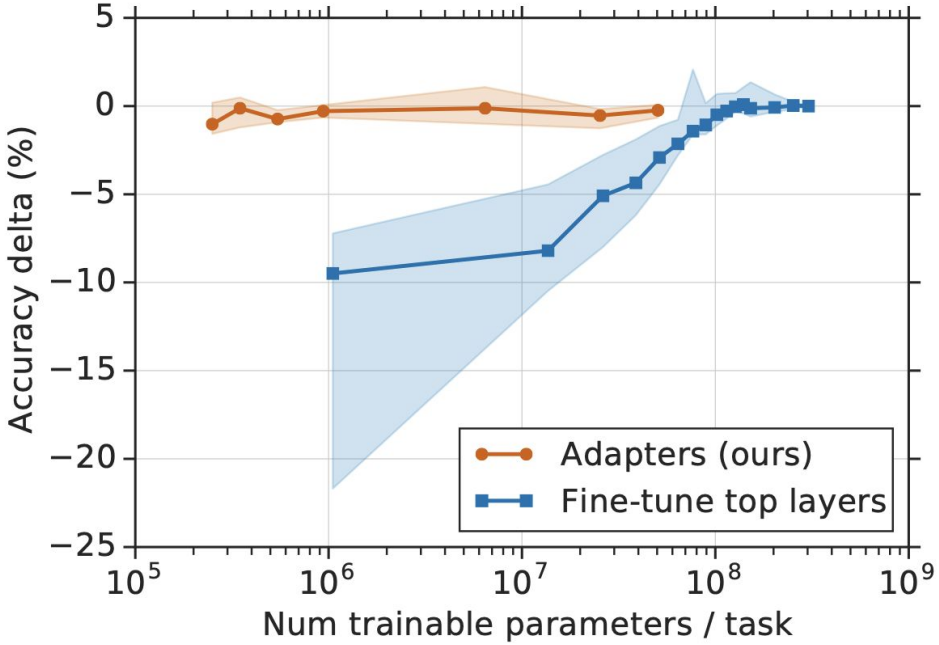
Adding New Weights: Adapters



You can swap adapters back-and-forth for different tasks.

- Adapter for sentiment analysis
- Adapter for NER
- Adapter for Summarization

Adding New Weights: Adapters



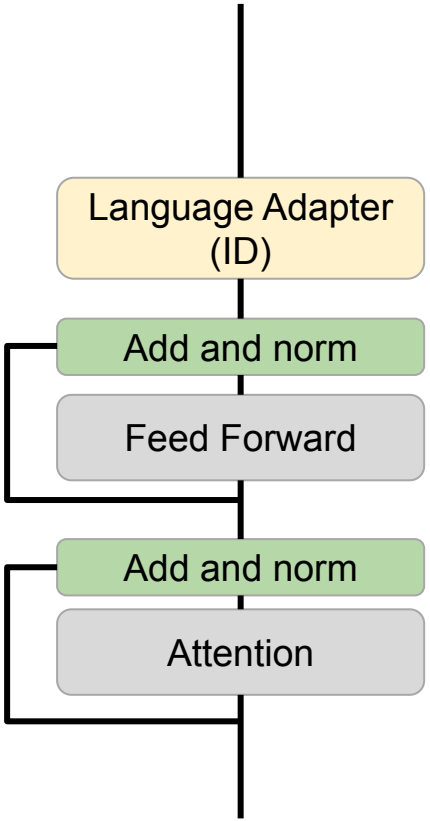
Adding New Weights: MAD-X [Pfeiffer et al., 2020]

Adapter based approach designed for cross-lingual adaptation.

Use-case:

- We have a multilingual LM (eg. BLOOM)
- We want to finetune to certain **task T** in a low-resource **language L**, but we have no dataset for T in L.
- We do have dataset for **task T** in another hi-res **language S**

Adding New Weights: MAD-X

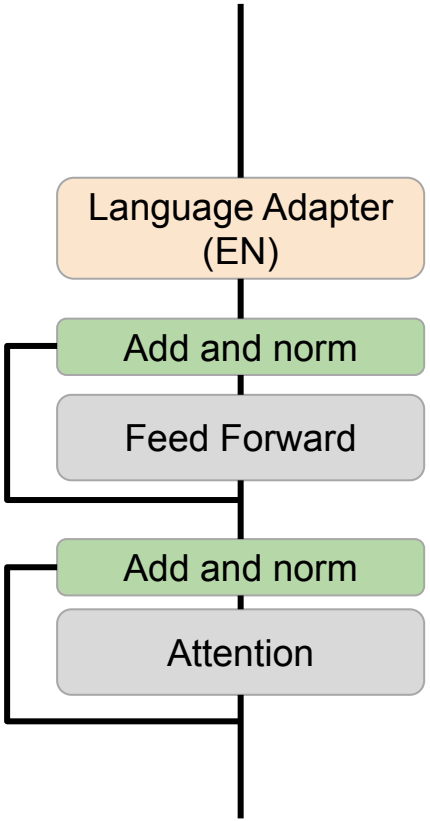


First, we train a language adapter.
We train with unlabeled data with MLM objective.

Other weights are frozen.

We train language adapter for the target language (eg ID)

Adding New Weights: MAD-X

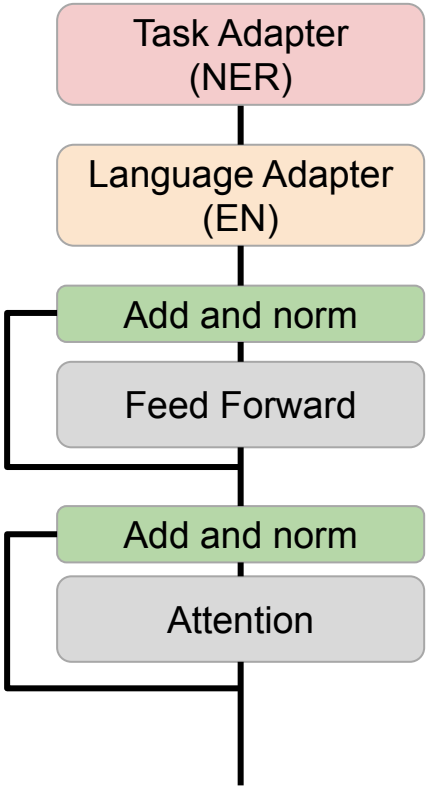


Once done, keep the ID adapter

Then we train language adapter for high-resource language (eg EN).

Language Adapter (ID)

Adding New Weights: MAD-X

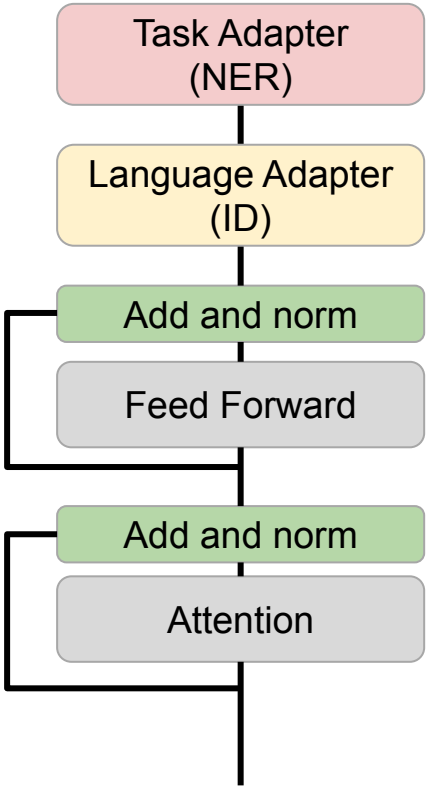


Then, we finetune to the desired downstream task on that high-resource data. Eg. NER.

Now we add new task adapter, and the rest of the model is frozen, including the language adapter.

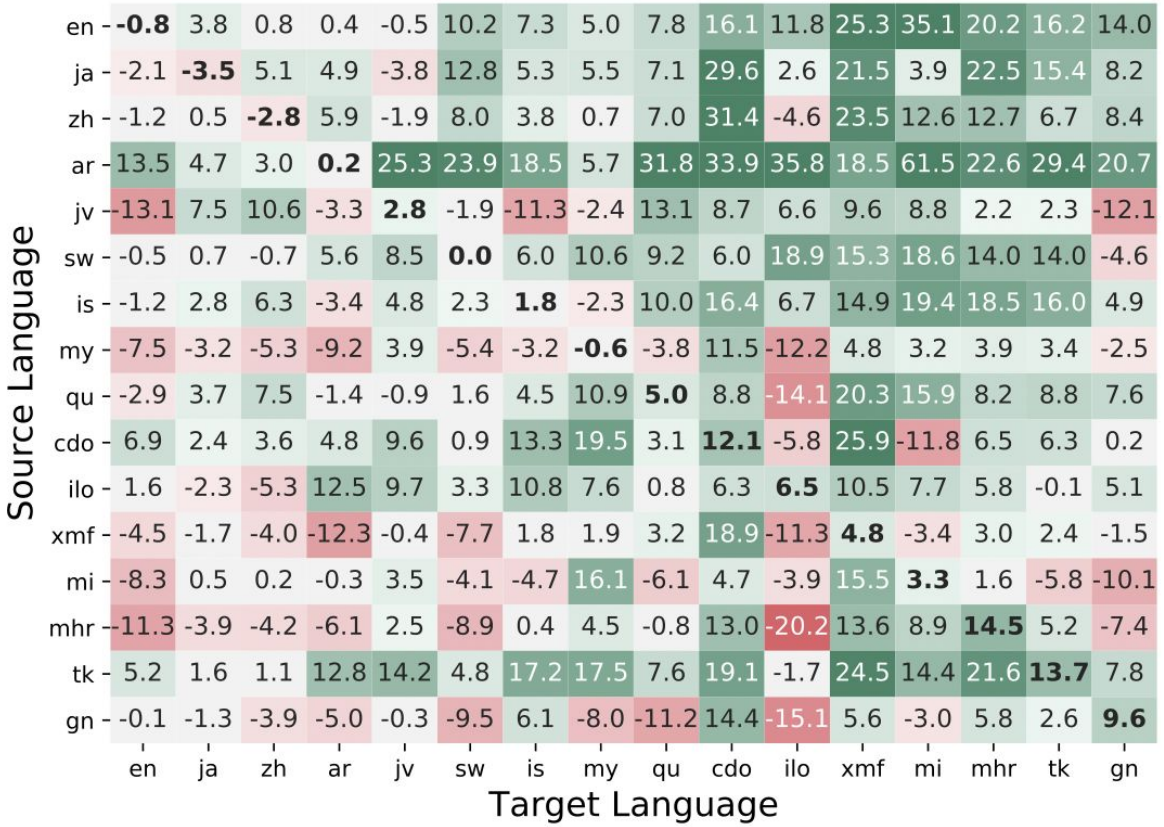
Language Adapter (ID)

Adding New Weights: MAD-X



Once done, plug back the language adapter ID, and now the model can do NER on ID without explicit training data for NER in Indonesian

Adding New Weights: MAD-X



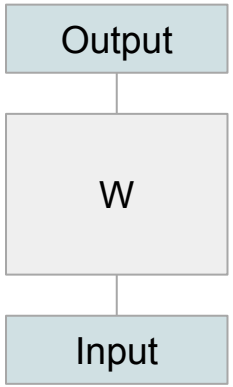
Adapters: Recap

- Parameter-efficient
- Modular: Replace your ‘adapters’ as needed
- Although insignificant, slows down the compute a tiny bit (extra operations)

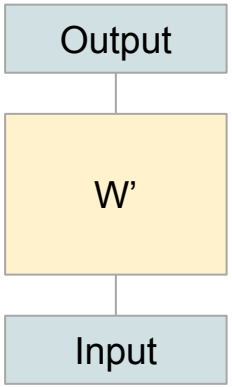
LoRA [Hu et al., 2021]

Let's assume we finetune the whole parameters.
Let's assume a simple 1 layer FFN.

before finetuning



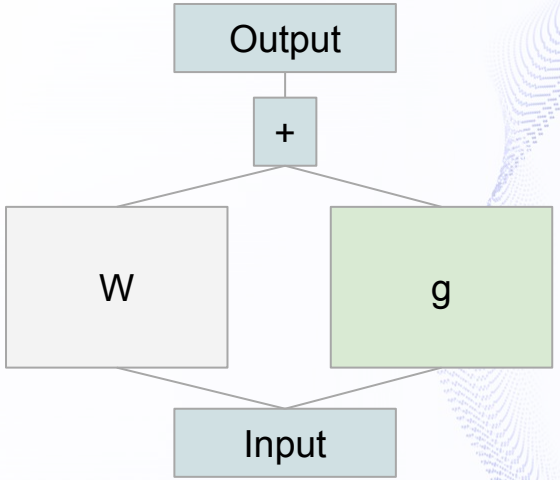
O = f(Wx)



O = f(W'x)

after finetuning

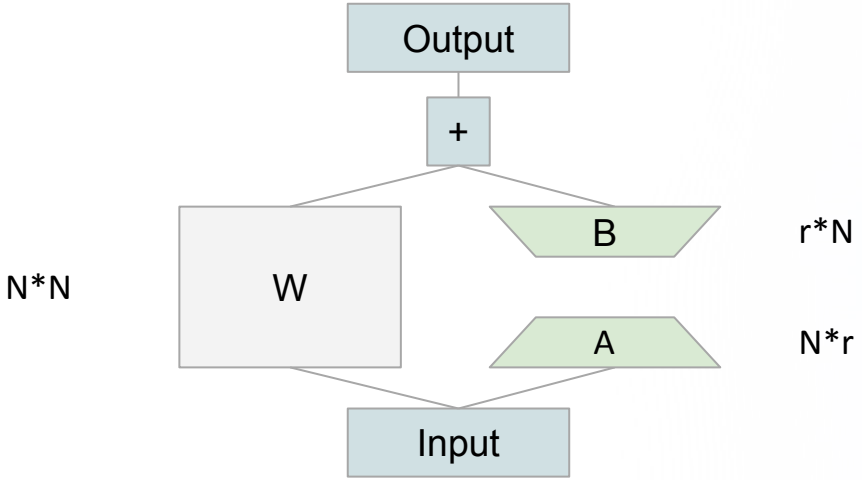
equals to →



O = f(Wx + gw)
where W' = W + g
(i.e. g is the
parameter update)

LoRA

- Big network is overparameterized, they hypothesize that the update for specific task is a low-rank matrix.
- So, we can just represent the update as down-projection x up-projection (like in adapters!)
- Think this like the adapter layers, but rather than sequential, it is parallel



MBZUAI

How small is r?

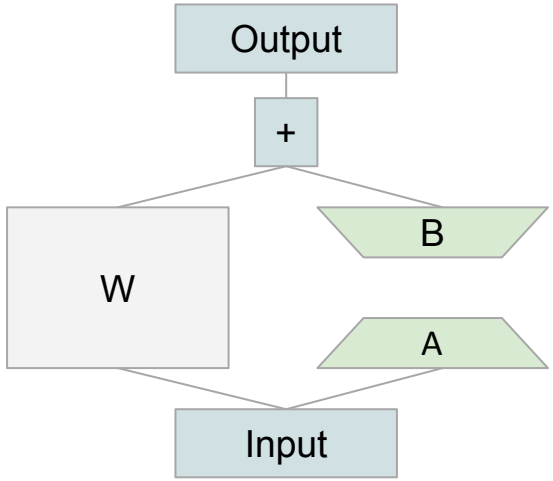
7.2 What is the Optimal Rank r for LoRA?

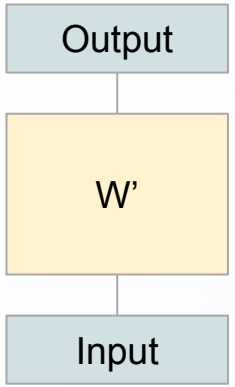
We turn our attention to the effect of rank r on model performance. We adapt $\{W_q, W_v\}$, $\{W_q, W_k, W_v, W_c\}$, and just W_q for a comparison.

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm$0.5%)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm$0.1%)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

LoRA: Merging Weights

- Similar to adapters, LoRA add small compute and model size increase
- However, we can merge LoRA's adapter (by sacrificing modularity):
- Merged LoRA: No additional compute cost!





$O = f(Wx + BAx)$

$O = f((W + BA)x)$

$W' = W + BA$

LoRA: Performance

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} (Adpt^{D})*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm.0}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
RoB_{base} (Adpt^{D})*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
RoB_{base} (LoRA)	0.3M	$87.5_{\pm.3}$	$\mathbf{95.1}_{\pm.2}$	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	$\mathbf{93.3}_{\pm.3}$	$90.8_{\pm.1}$	$\mathbf{86.6}_{\pm.7}$	$\mathbf{91.5}_{\pm.2}$	**87.2**

Post-Training Optimization

Post-Training Optimization

- We have efficiently train our model either through distillation or adapter approach.
- Can we further optimize it after training?

Quantization

Reducing the numerical precision of the model's parameters
Reduce model size & faster inference

Pruning

Removing a subset of model's parameters that are insignificant
Reduce model size

Converting Model Format

Increasing model's interoperability in various deployment settings

Post-Training Optimization

- We have efficiently train our model either through distillation or adapter approach.
- Can we further optimize it after training?

Quantization

Reducing the numerical precision of the model's parameters
Reduce model size & faster inference

Converting Model Format

Increasing model's interoperability in various deployment settings

Quantization

$$H = Wx$$

$$\begin{bmatrix} 12.4 & 0.22 & 3.11 \\ 2.44 & 10.19 & 41.0 \\ 0.11 & 0.23 & 2.34 \end{bmatrix} \longrightarrow \begin{bmatrix} 127 & 12 & 41 \\ 34 & 112 & 127 \\ 11 & 12 & 33 \end{bmatrix}$$

float32 representation

$$w \in [\alpha, \beta]$$

int8 representation

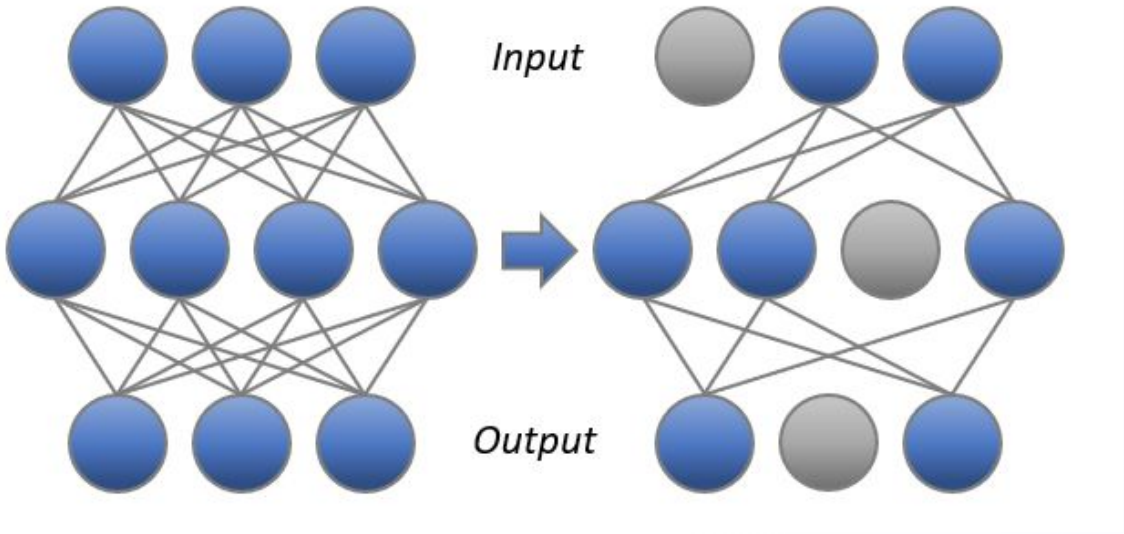
$$w_q \in [\alpha_q, \beta_q]$$

$$w_q = clip(round(\frac{1}{s}w + z), [\alpha_q, \beta_q])$$

$$s = \frac{\beta - \alpha}{\beta_q - \alpha_q} \qquad z = round(\frac{\beta\alpha_q - \alpha\beta_q}{\beta - \alpha})$$

Pruning

- Removing a subset of model's parameters while preserving the model's performance.



Pruning Illustration from [Intel Neural Compressor Docs]

Converting Model Format

- Libraries/Framework such as PyTorch, TensorFlow, or JAX provides full functionalities to design, train, and experiment with models.
- We probably do not need many of those functionalities for the end-deployment / production.
- Converting our models into suitable format for inference-only settings, could ensure more **interoperability** and platform-specific **acceleration**.

Materials

Code: https://github.com/haryoa/pricai-tutorial-q2ai

(not yet up, will be by tonight)