MOHAMED BIN ZAYED
UNIVERSITY OF
ARTIFICIAL INTELLIGENCE

Training Lightweight Model via Knowledge Distillation and Parameter Efficient Fine Tuning

Mexican NLP Summer School 2024

Alham Fikri Aji

In This Tutorial:

- Hands-on
- Learn on how to finetune models
    - with distillation
    - with parameter-efficient finetuning

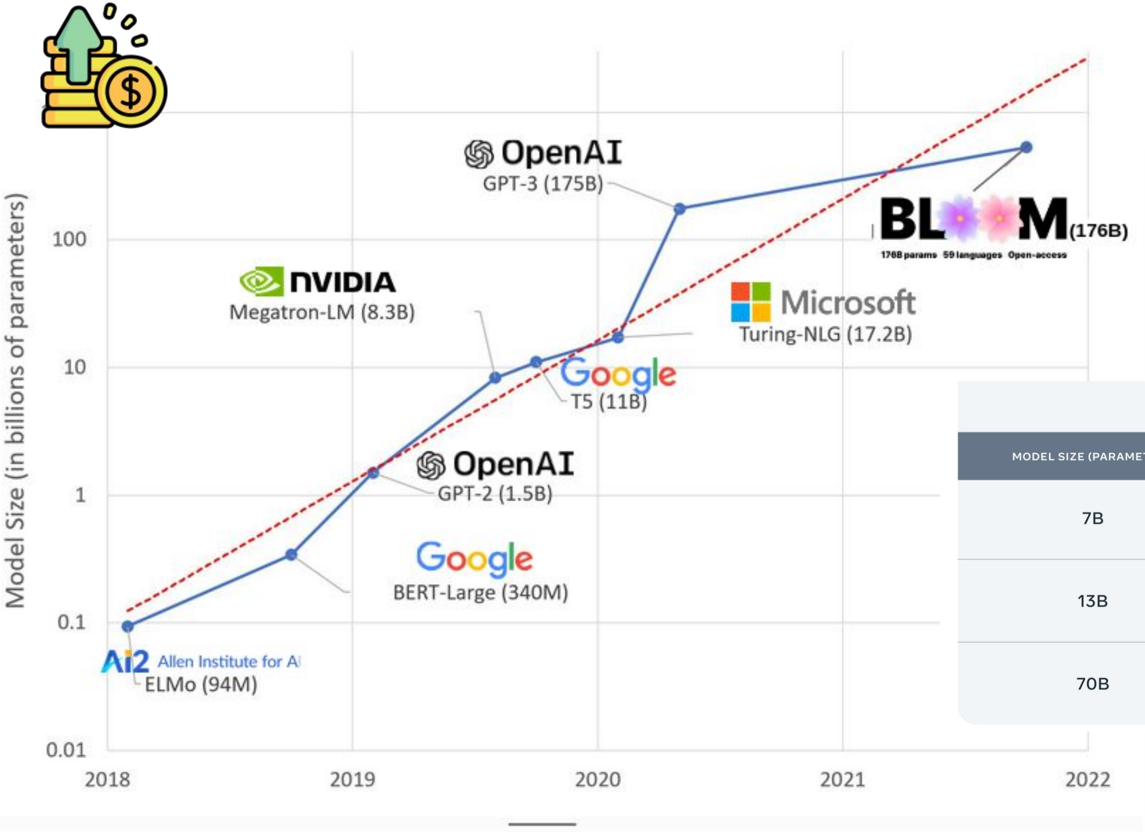
- Have 0 experience in pytorch/huggingface? No Problem!
    - Join the hands on, we’ll help you out!

Growing Size of NLP Models (esp LLM). Why?

Causal Language Model objective. Based on Transformers.

Model	Model Size	Data
GPT (Radford., et al, 2018)	117M parameters	Book Corpus (~7k books)
GPT-2 (Radford., et al, 2018)	up to 1.5B parameters	WebText (40GB internet data)
GPT-3 (Brown., et al, 2020)	up to 175B parameters	Expanded WebText CommonCrawl Wikipedia Books (~500B tokens)
GPT-4	?	?

Growing Size of NLP Models (esp LLM). Why?



Llama 2

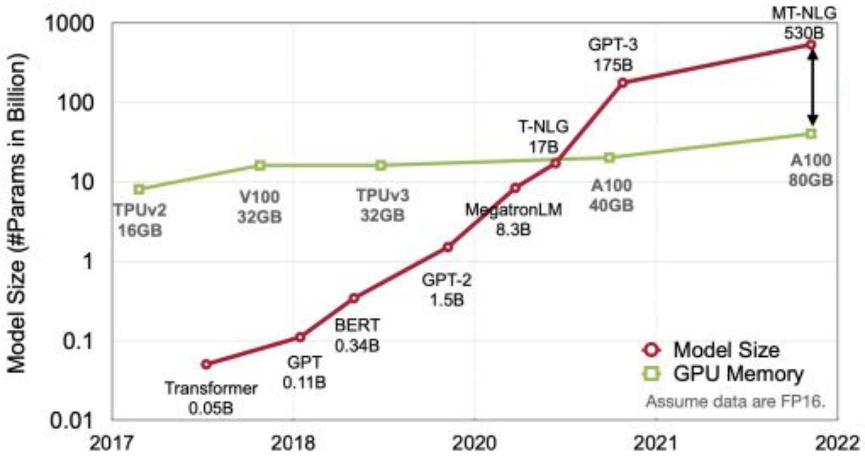
MODEL SIZE (PARAMETERS)	PRETRAINED	FINE-TUNED FOR CHAT USE CASES
7B	**Model architecture:** **Pretraining Tokens:** 2 Trillion **Context Length:** 4096	**Data collection for helpfulness and safety:** **Supervised fine-tuning:** Over 100,000 **Human Preferences:** Over 1,000,000
13B		
70B		

Cost Issue with Large Language Models

Cost Issue with Large Language Models

($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.



Why Don’t We Train Smaller One?

Few-shot In-context Learning - Scale Matters

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:     ← task description
2  cheese =>                        ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

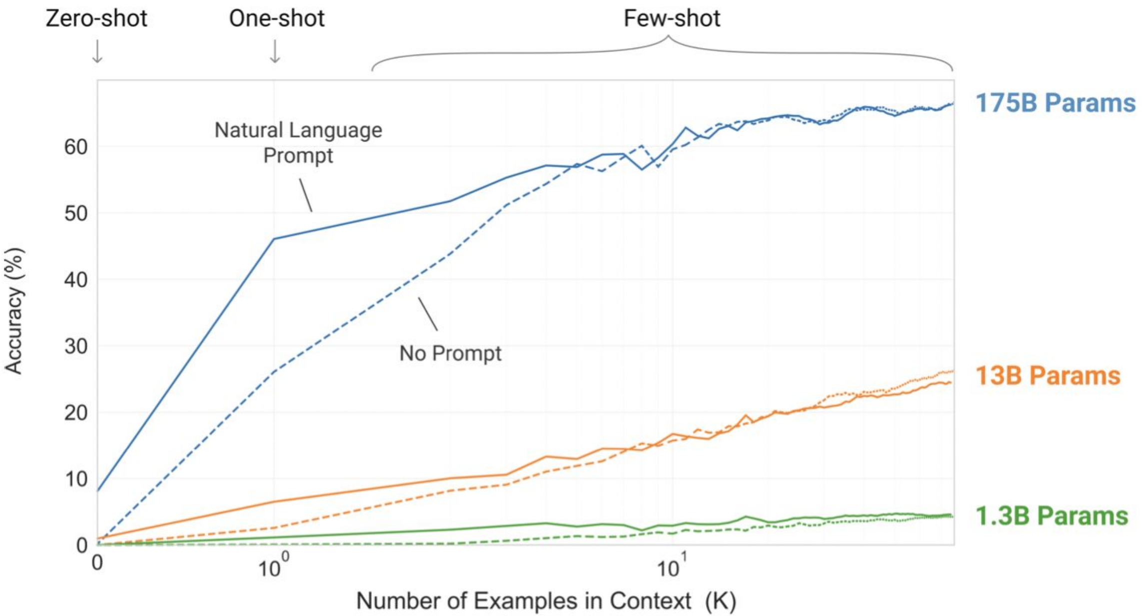
```
1  Translate English to French:     ← task description
2  sea otter => loutre de mer       ← example
3  cheese =>                        ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:     ← task description
2  sea otter => loutre de mer       ← examples
3  peppermint => menthe poivrée
4  plush girafe => girafe peluche
5  cheese =>                        ← prompt
```

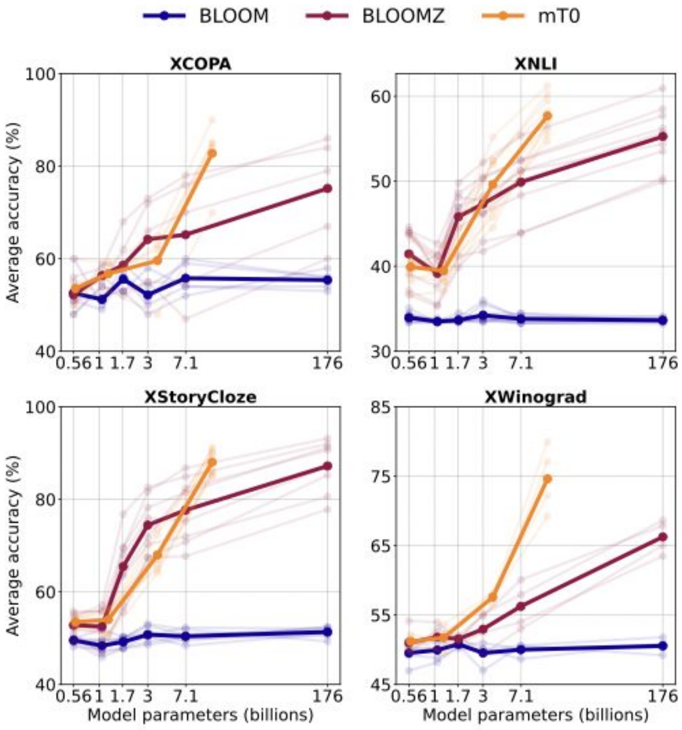
- Natural prompt helps, especially in lower-shot.
- More examples is better
- Larger model is better



word scrambling task

BLOOMZ and MT0: Multilingual Generalization (Muennighoff et al., 2023)

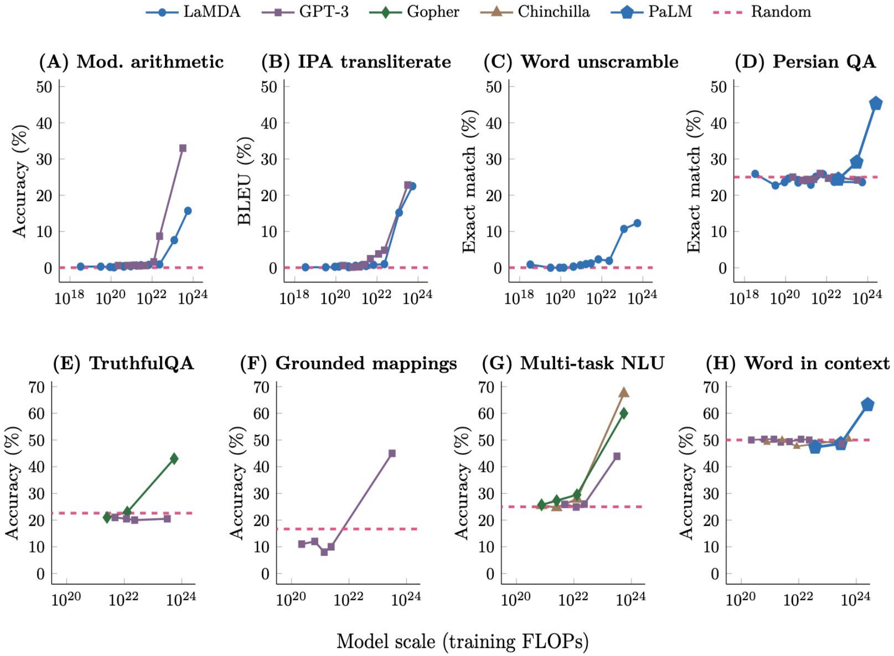
- Cross-lingual generalization: Better performance at scale



2 Emergent Abilities Definition

As a broad concept, emergence is often used informally and can be reasonably interpreted in many different ways. In this paper, we will consider a focused definition of emergent abilities of large language models:

An ability is emergent if it is not present in smaller models but is present in larger models.



What If We Need a Smaller One?

- Cost
- Accessibility
- Privacy

Knowledge Distillation

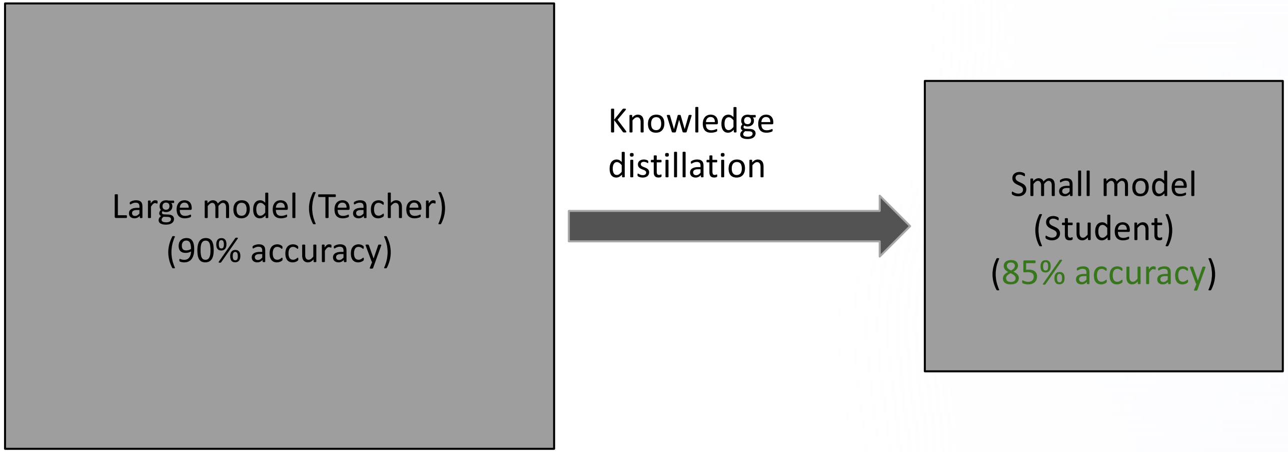
Training a super-small model competitive to the large ones!

Large model
(90% accuracy)

Small model
(60% accuracy)

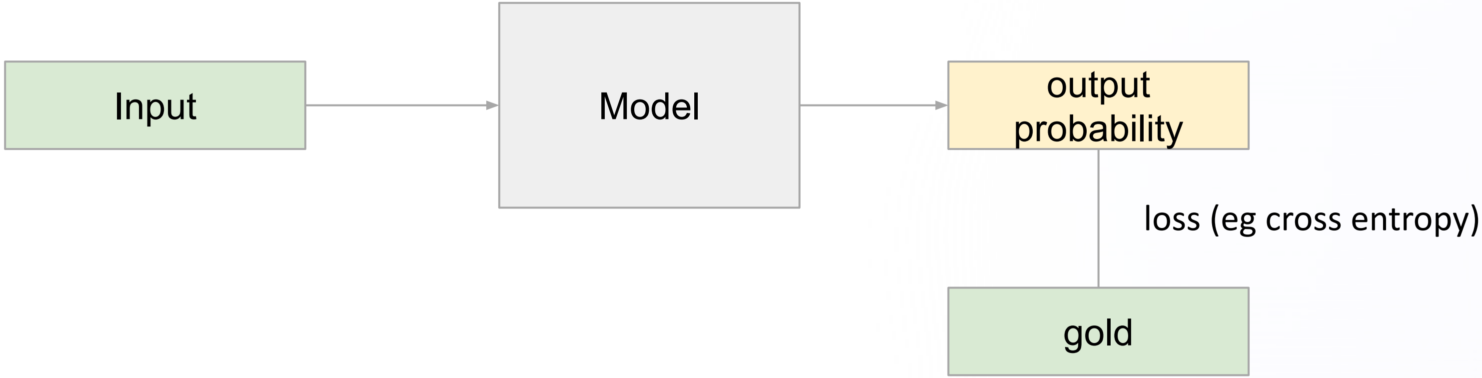
Knowledge Distillation

Training a super-small model competitive to the large ones!



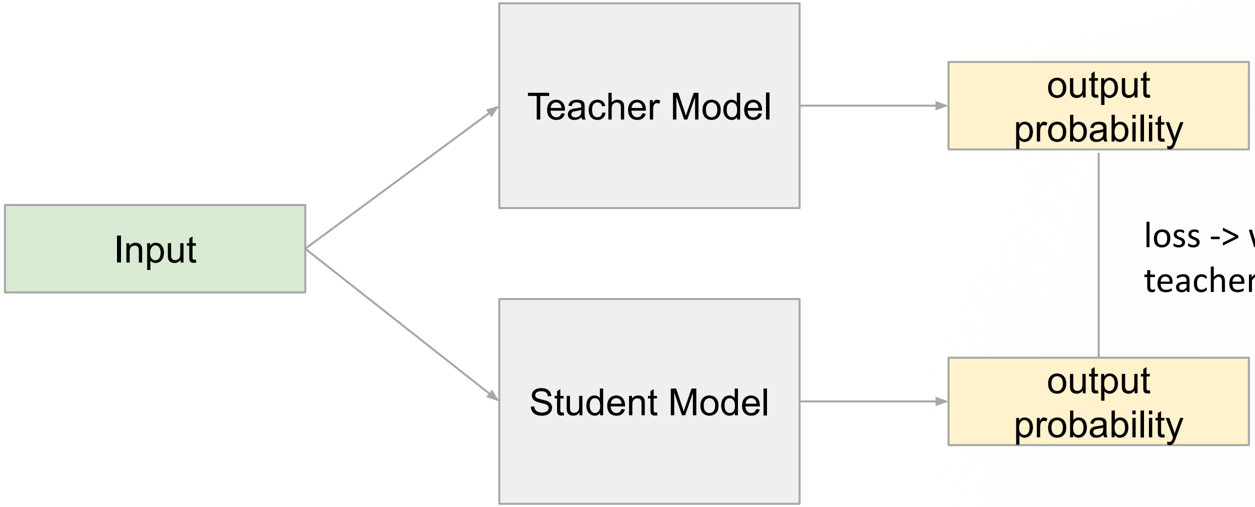
Knowledge Distillation

Standard training learns from gold-label data



Knowledge Distillation

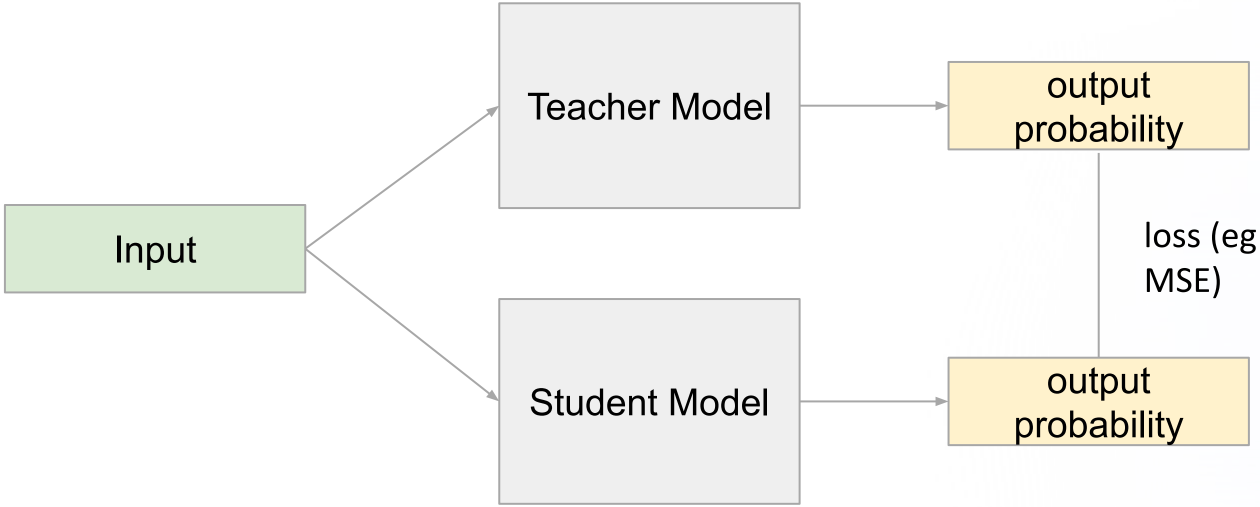
KD learns from the teacher



loss -> we want the student to mirror the teacher's behaviour!

Knowledge Distillation

KD learns from the teacher

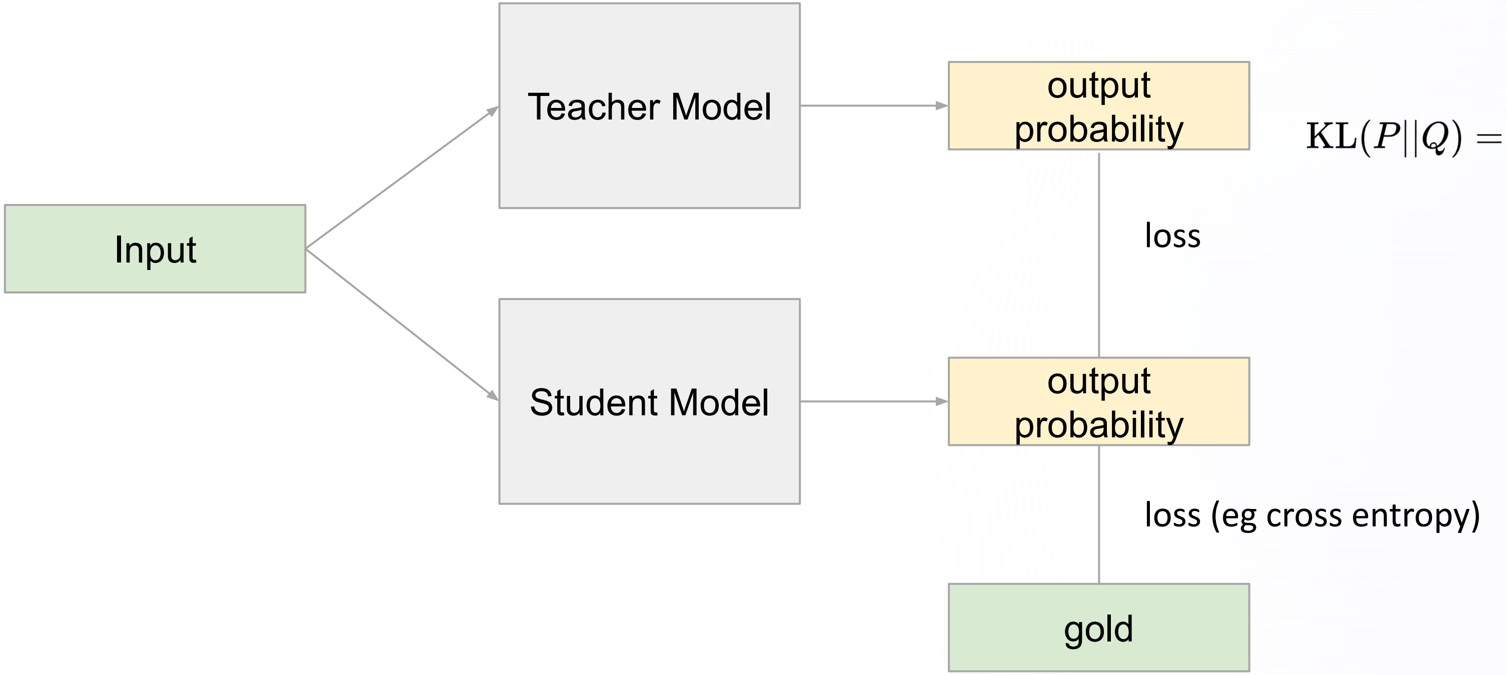


$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2.$$

$$\mathrm{KL}(P||Q) = \sum_i P(i) \cdot \log \frac{P(i)}{Q(i)}$$

Knowledge Distillation

KD learns from the teacher



$$\mathrm{KL}(P||Q) = \sum_i P(i) \cdot \log \frac{P(i)}{Q(i)}$$

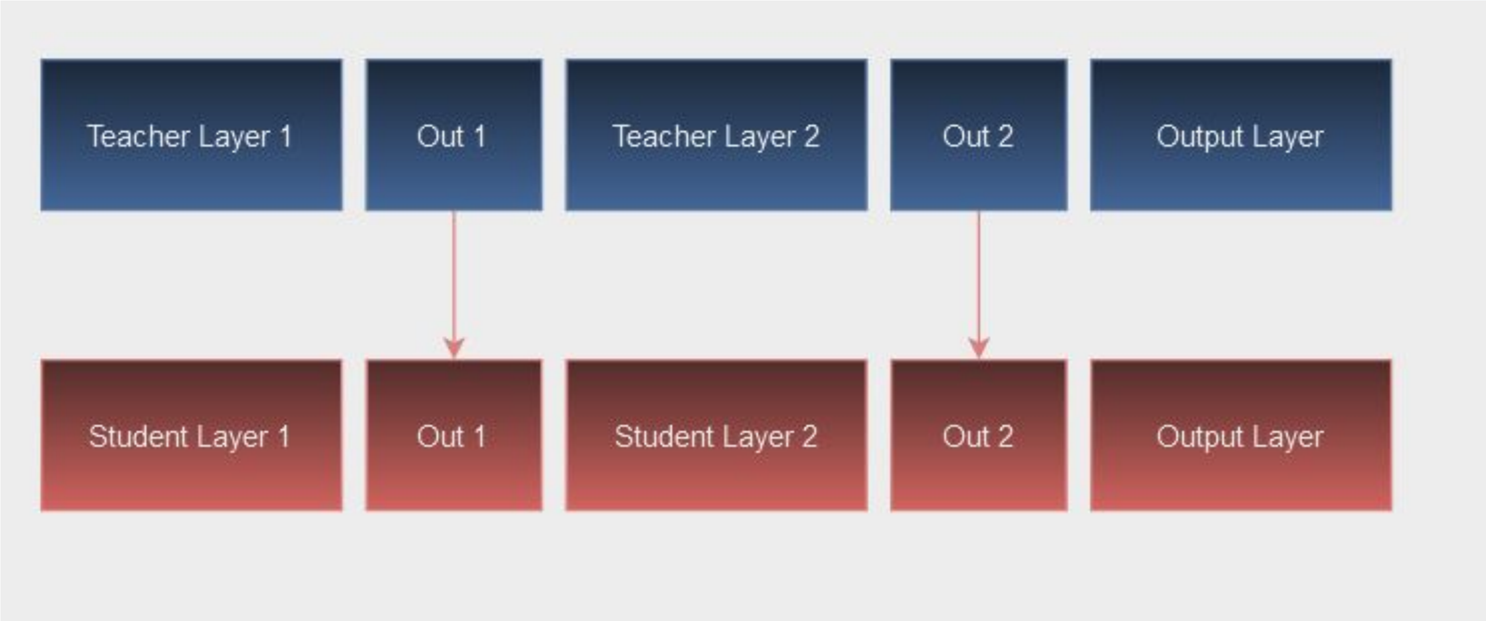
Knowledge Distillation

HANDS-ON DEMO!

https://tinyurl.com/y4h3jeas

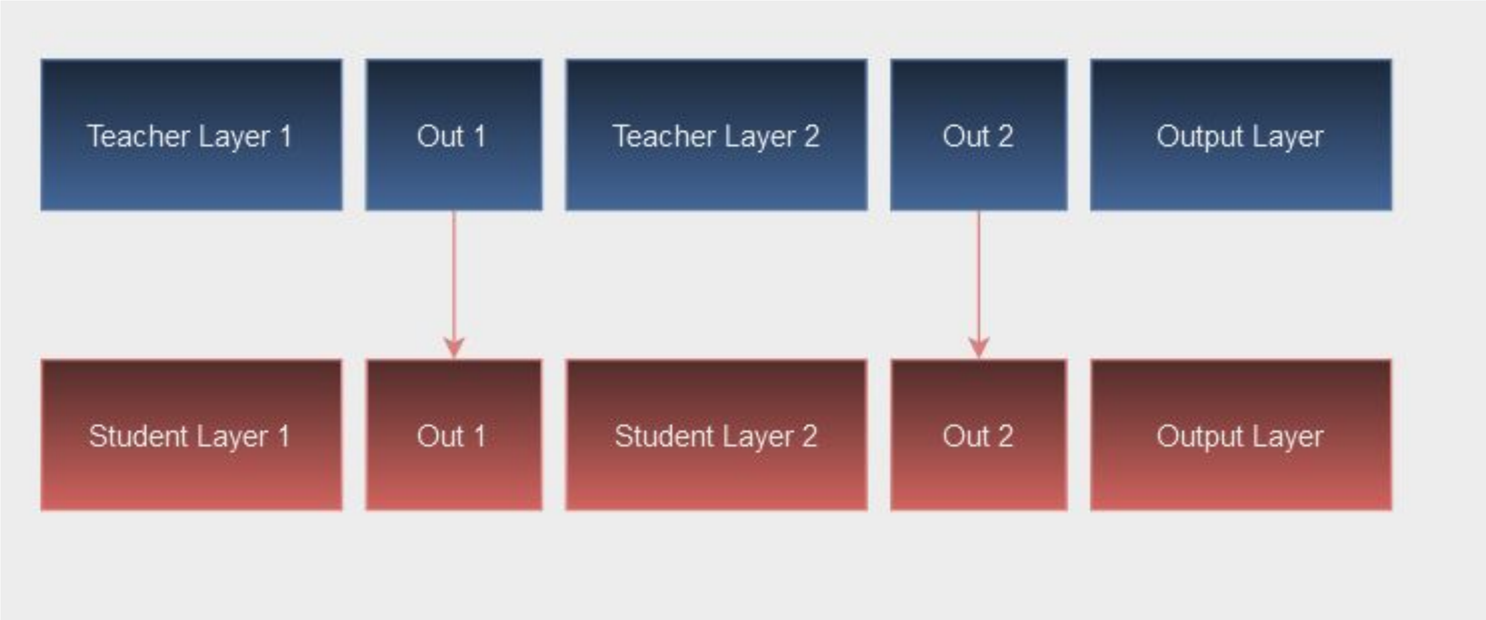
Knowledge Distillation

You can also have a loss for each of the intermediate activation



Knowledge Distillation

You can also have a loss for each of the intermediate activation
Q: They have different unit size



Knowledge Distillation

You can also have a loss for each of the intermediate activation

Q: They have different unit size

A: Projection layers

```
class TinyBertForPreTraining(BertPreTrainedModel):
    def __init__(self, config, fit_size=768):
        super(TinyBertForPreTraining, self).__init__(config)
        self.bert = BertModel(config)
        self.cls = BertPreTrainingHeads(
            config, self.bert.embeddings.word_embeddings.weight)
        self.fit_dense = nn.Linear(config.hidden_size, fit_size)
        self.apply(self.init_bert_weights)

    def forward(self, input_ids, token_type_ids=None,
                attention_mask=None, masked_lm_labels=None,
                next_sentence_label=None, labels=None):
        sequence_output, att_output, pooled_output = self.bert(
            input_ids, token_type_ids, attention_mask)
        tmp = []
        for s_id, sequence_layer in enumerate(sequence_output):
            tmp.append(self.fit_dense(sequence_layer))
        sequence_output = tmp

        return att_output, sequence_output
```

Knowledge Distillation: Importance and Applications

Speed: Faster inference times.

System	#Params	#FLOPs	Speedup	MNLI-(m/mm)	QQP
$\text{BERT}_{\text{BASE}}$ (Teacher)	109M	22.5B	$\boxed{1.0x}$	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	$\boxed{9.4x}$	**82.5/81.8**	**71.3**

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

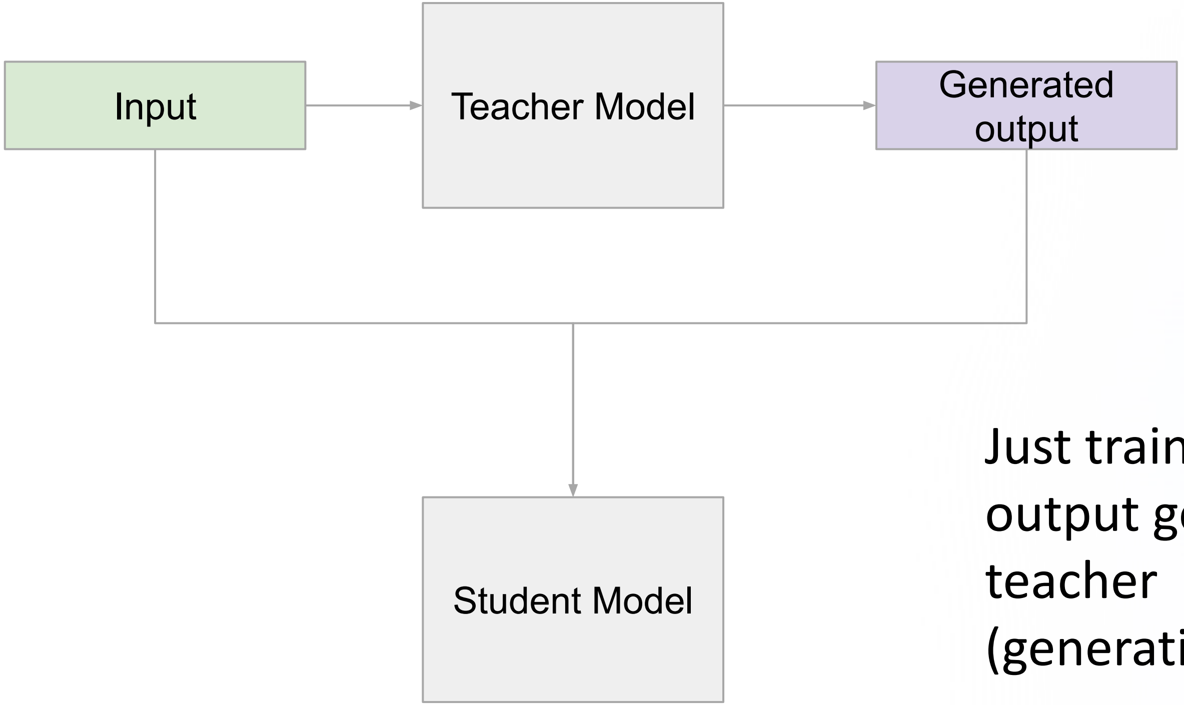
Knowledge Distillation: Importance and Applications

Accessibility: Lightweight models

System	#Params	#FLOPs	Speedup	MNLI-(m/mm)	QQP
$\text{BERT}_{\text{BASE}}$ (Teacher)	$\boxed{109\text{M}}$	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)	$\boxed{14.5\text{M}}$	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).

Sequence-Level Knowledge Distillation [Kim et al., 2016]



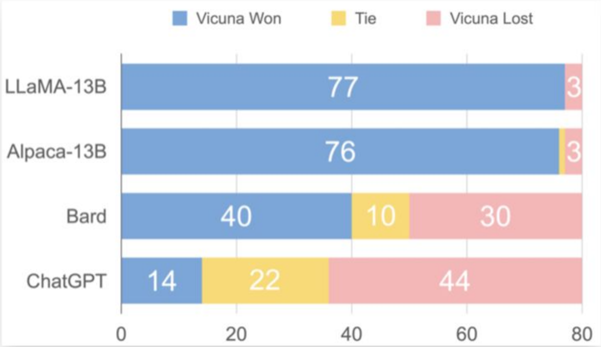
Just train with the output generated by the teacher (generative model)

Sequence-Level Knowledge Distillation [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**

Distilling from ChatGPT/GPT-4

Model Name	LLaMA	Alpaca	Vicuna
Dataset	Publicly available datasets (1T token)	Self-instruct from davinci-003 API (52K samples)	User-shared conversations (70K samples)
Training code	N/A	Available	Available
Evaluation metrics	Academic benchmark	Author evaluation	GPT-4 assessment
Training cost (7B)	82K GPU-hours	$500 (data) + 100$ (training)	$140 (training)
Training cost (13B)	135K GPU-hours	N/A	$300 (training)



Unlabelled Data + KD

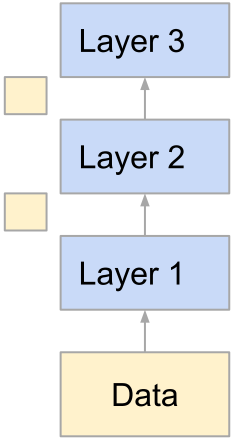
- The gold-label loss could be optional, so we can utilize a lot of unlabelled data

- KD is a way to train smaller model with a competitive performance
- Weakness?

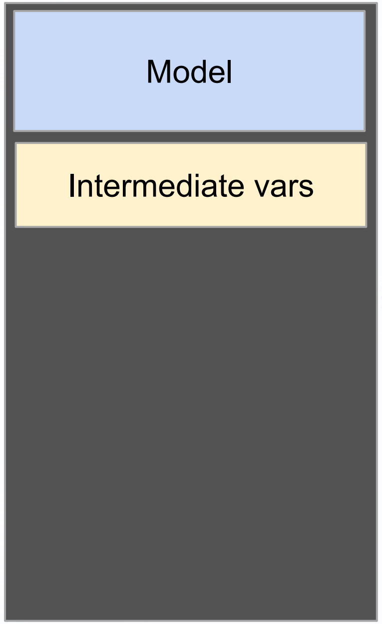
- Next:
  - How to train a bigger model?

What is going on during training?

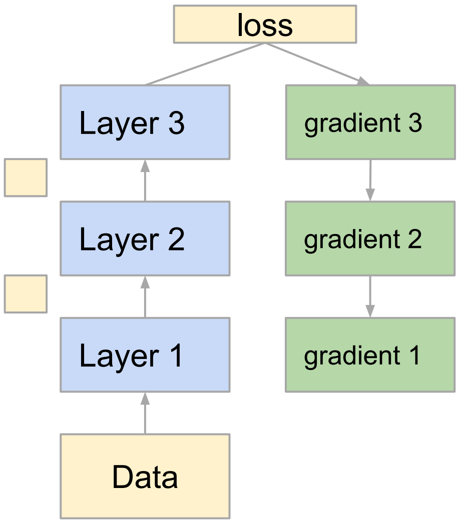
What is going on during training?



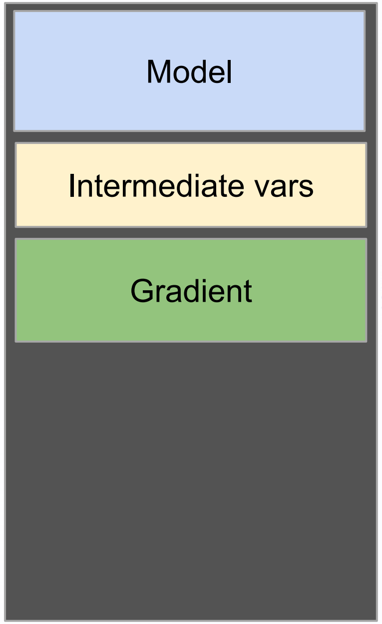
your GPU:



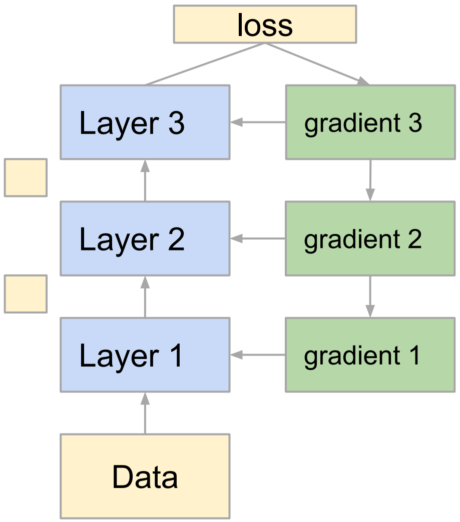
What is going on during training?



your GPU:



Big Models = Need Big GPU



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

if $maximize$ **:**

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

else

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

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

$$\boxed{\begin{aligned} m_t &\leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &\leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned}}$$

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

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

if $amsgrad$

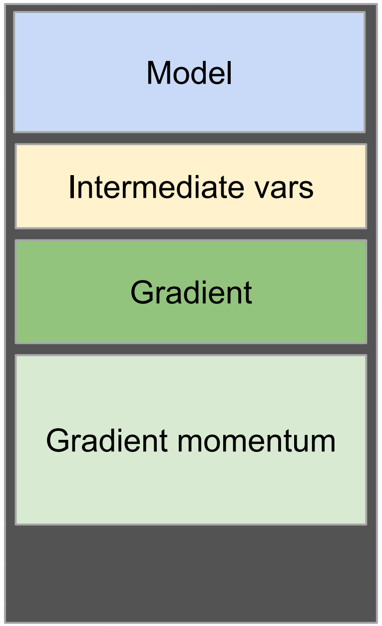
$$\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$$

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

else

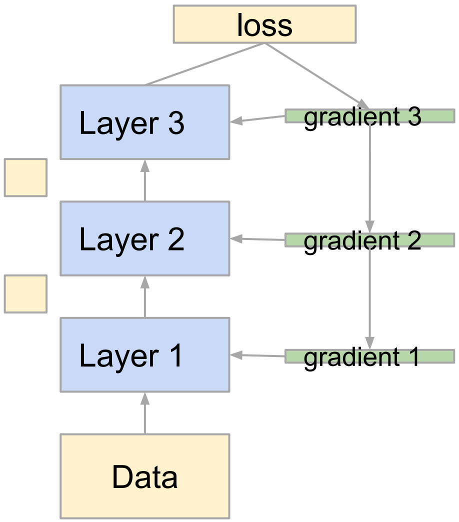
$$\theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}} + \epsilon\big)$$

your GPU:

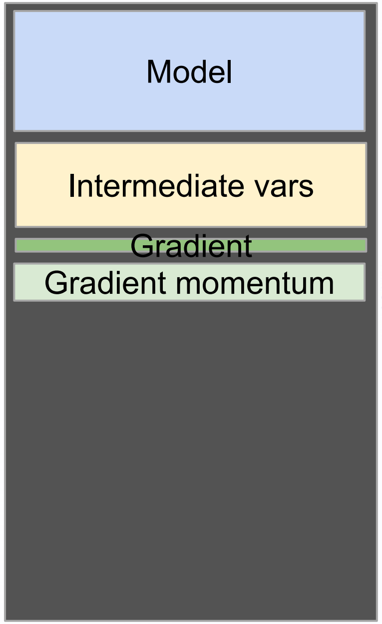


Parameter-Efficient Finetuning

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

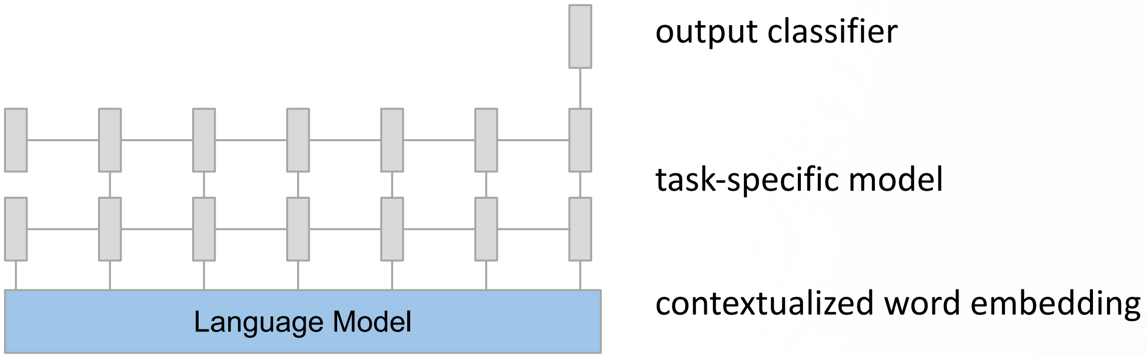


your GPU:



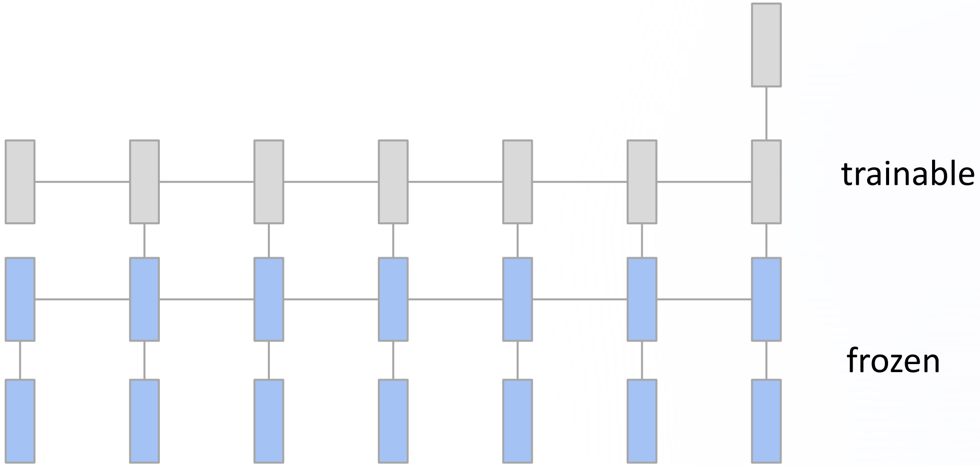
Training parts of your model

ELMO and contextualized-embedding era: it is common to use LM to get an input representation, then add a model on top of that, freezing the LM.



Training parts of your model

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



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.

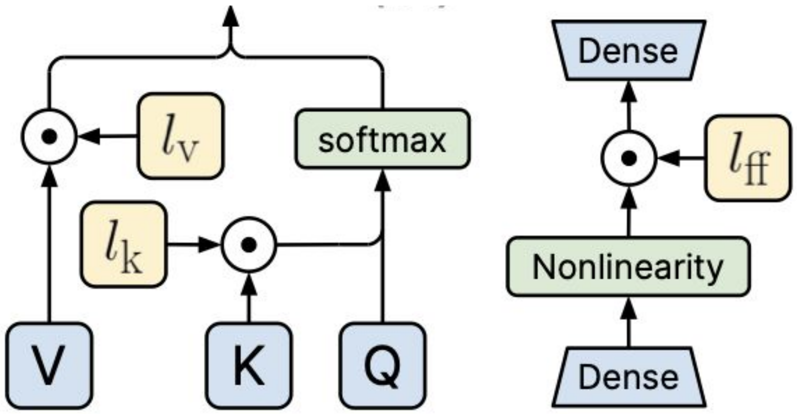
	Method	%Param	QNLI	SST-2	MNLI_m	MNLI_mm	CoLA	MRPC	STS-B	RTE	QQP	Avg.
BB	Full-FT	100%	**90.7±0.2**	92.0±0.4	**83.5±0.1**	**83.7±0.3**	56.4±0.9	89.0±1.0	88.9±0.7	70.5±0.6	**87.1±0.1**	82.3
BB	BitFit	0.09%	90.2±0.2	**92.1±0.3**	81.4±0.2	82.2±0.2	**58.8±0.5**	**90.4±0.5**	**89.2±0.2**	**72.3±0.9**	84.0±0.2	**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

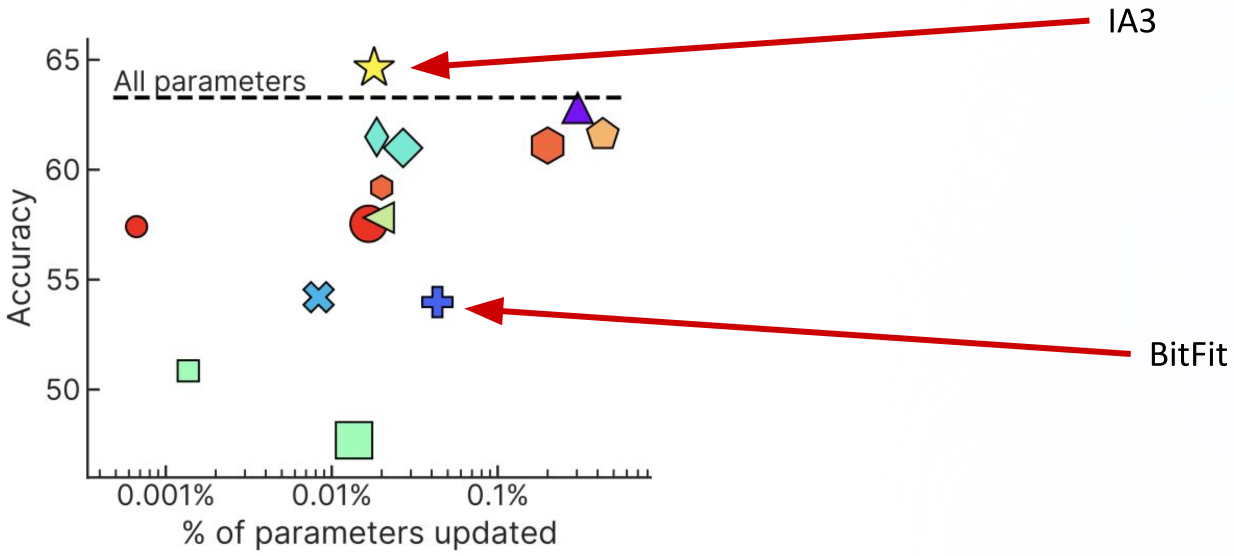
With IA3:

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

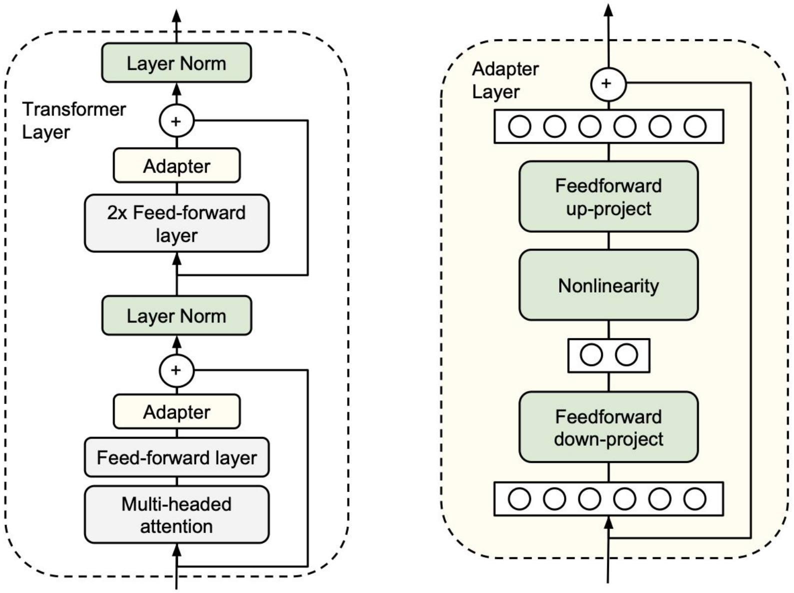
element-wise multiplication.
l just a vector (the same size as b).
Think this like BitFit, but multiplicative

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

Good performance, even better than full finetuning.



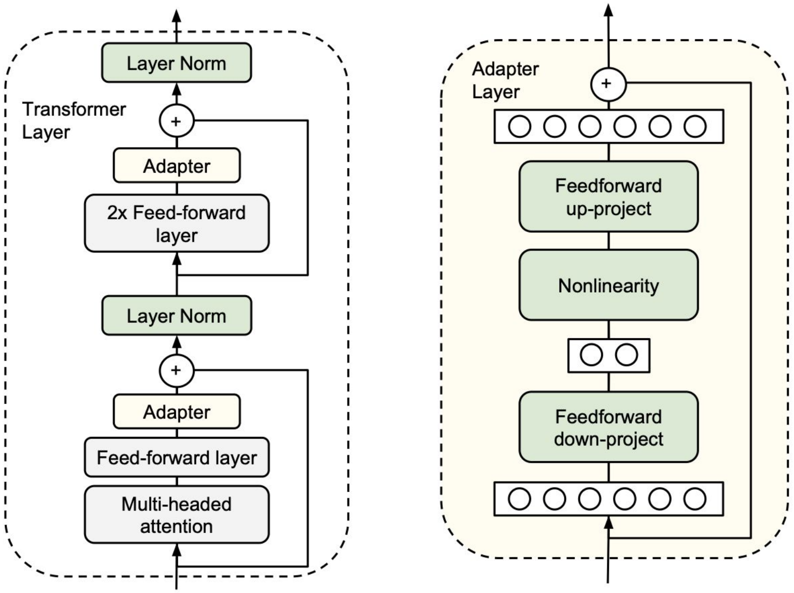
Adding New Weights: Adapters [Houlsby et al., 2019]



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

Why?

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

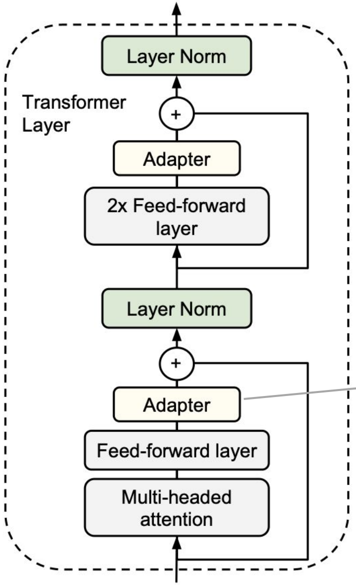


Adapter layer is efficient since it's using 2 FFN layers rather than a single FFN layer.

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



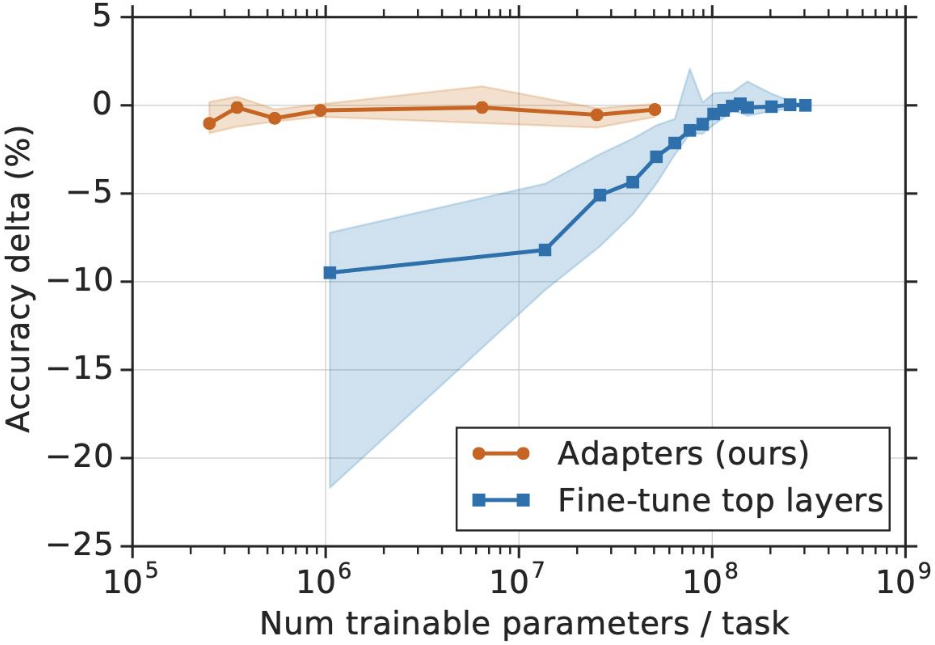
You can swap adapters back-and-forth for different tasks.

Adapter for sentiment analysis

Adapter for NER

Adapter for Summarization

Adding New Weights: Adapters



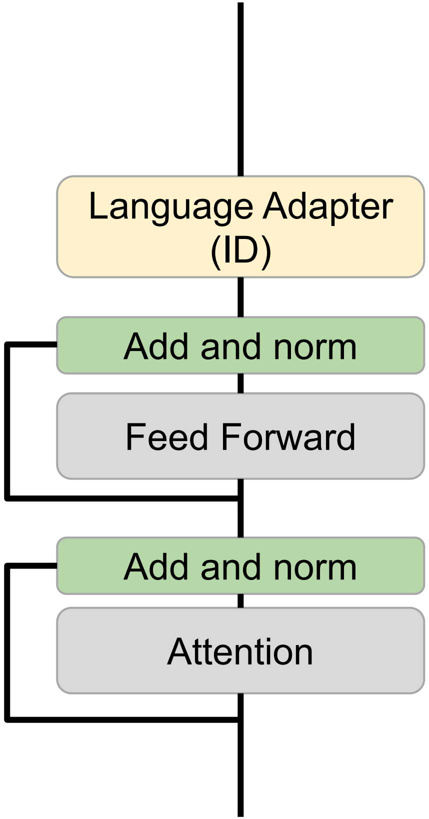
Adding New Weights: MAD-X [Pfeiffer et al., 2020]

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

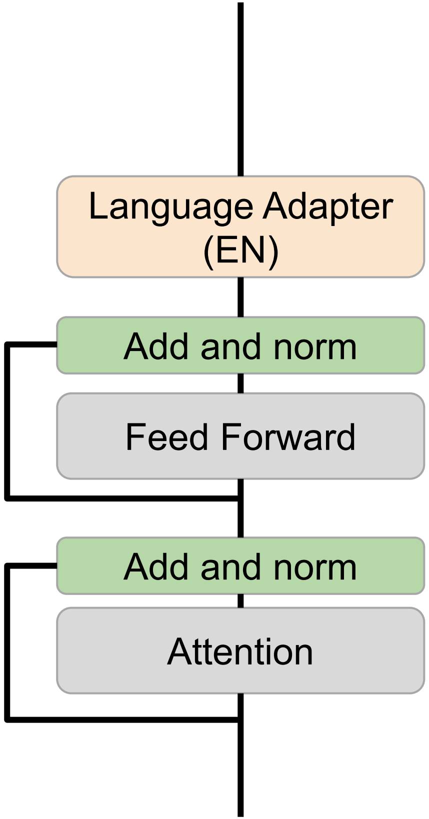


First, we train a language adapter.
We train with unlabeled data with MLM objective.

Other weights are frozen.

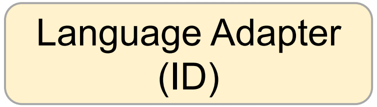
We train language adapter for the target language (eg ID)

Adding New Weights: MAD-X

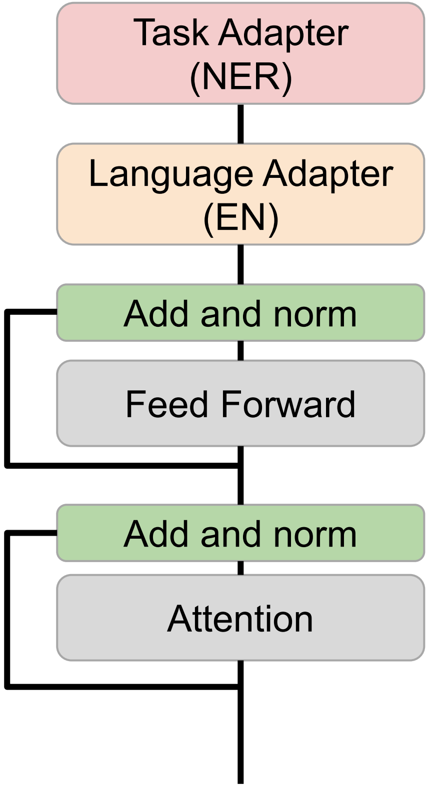


Once done, keep the ID adapter

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

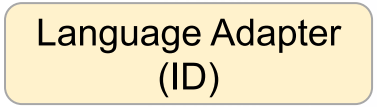


Adding New Weights: MAD-X

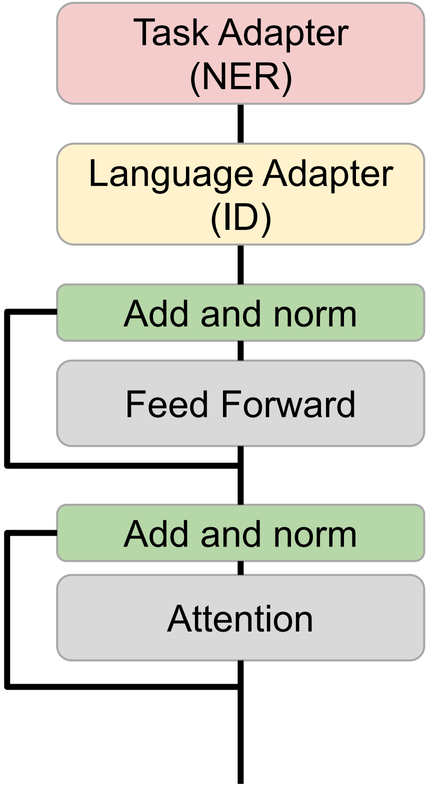


Then, we finetune to the desired downstream task on that high-resource data. Eg. NER.

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

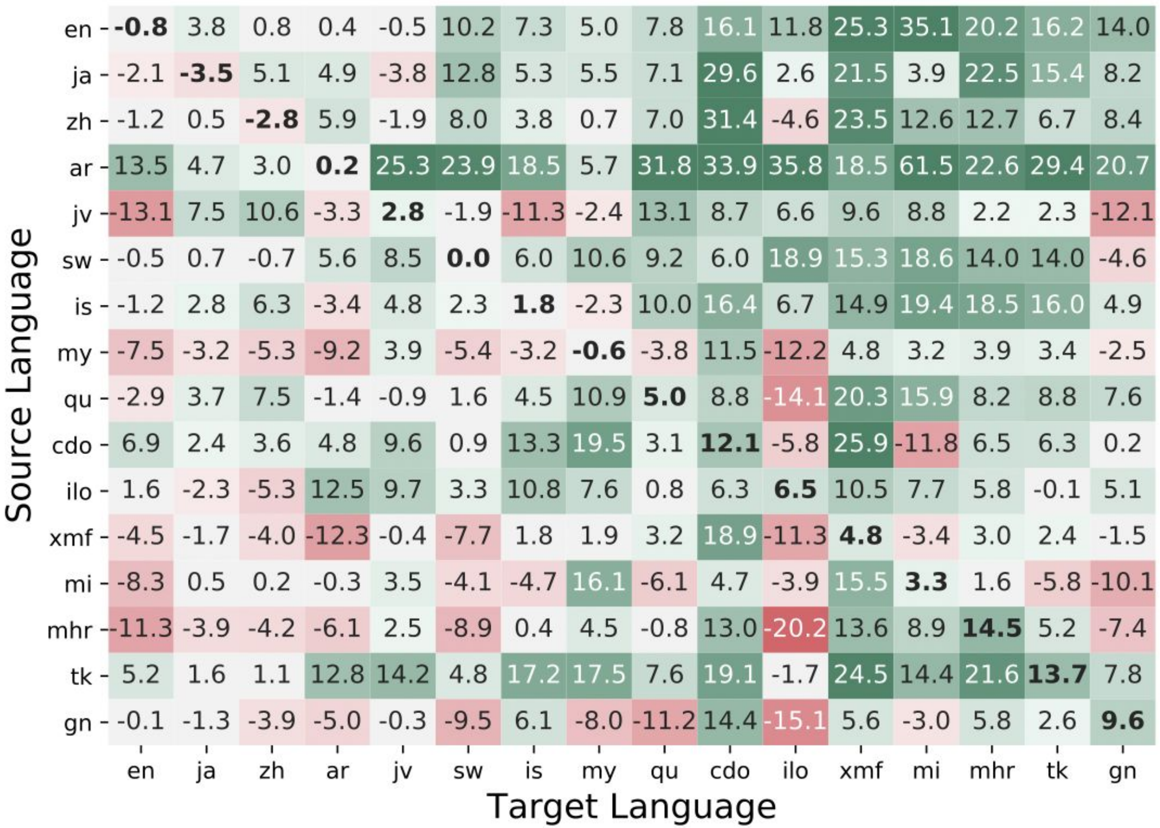


Adding New Weights: MAD-X



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



Adding New Weights: Prompt-Tuning

Recap: Prompting is a way to perform task from a sufficiently large LM without training it.

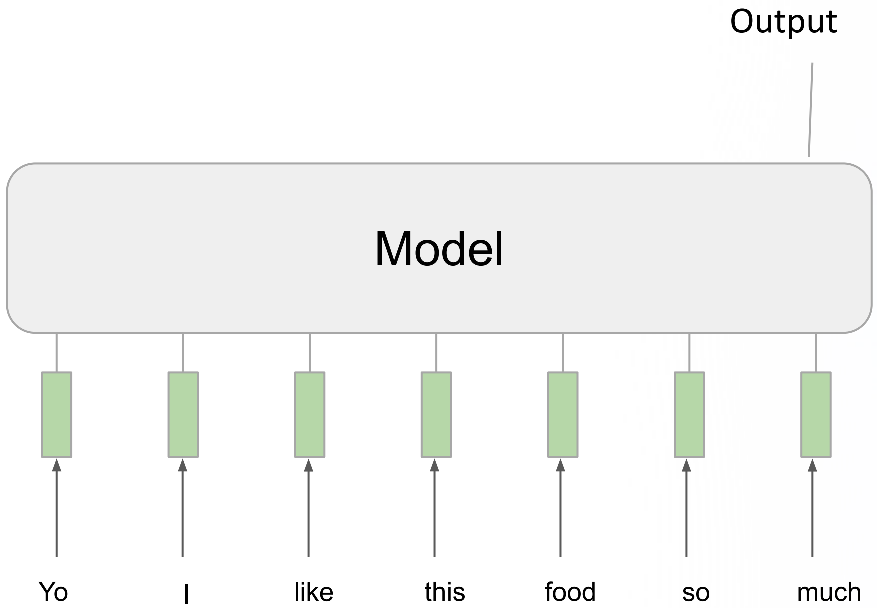
A

What is the sentiment of this sentence: "I like ice cream".

The sentiment of the sentence "I like ice cream" is positive. The word "like" indicates a favorable or positive feeling towards the subject, which in this case is "ice cream".

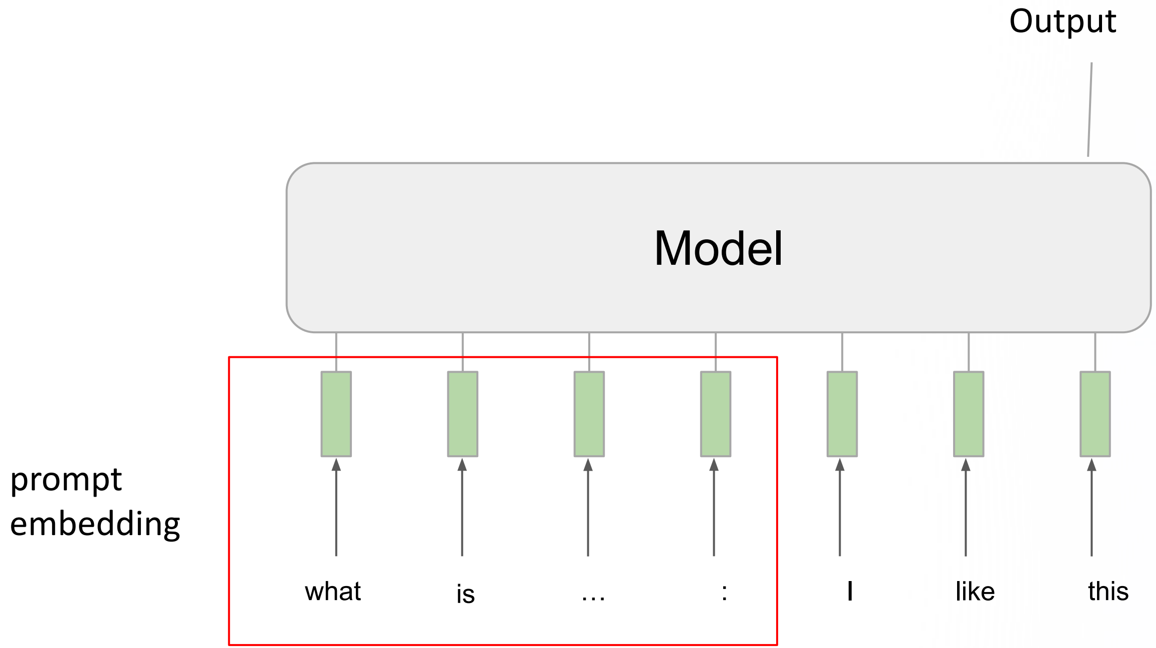
Adding New Weights: Prompt-Tuning

Recall that input is tokenized, then transformed into embeddings



Adding New Weights: Prompt-Tuning

Prompting is no exception



Adding New Weights: Prompt-Tuning

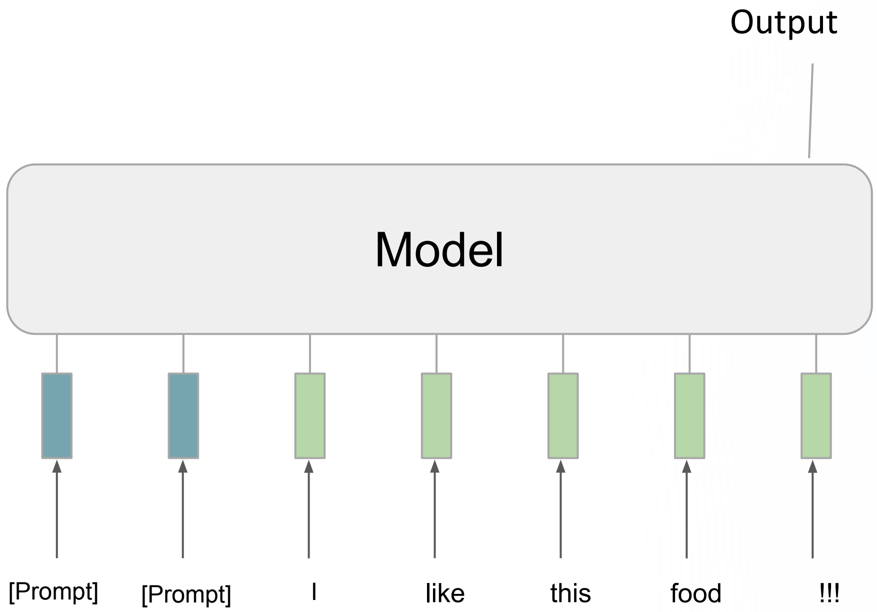
Prompting is black magic

No.	Category	Template	Accuracy
1	instructive	Let's think step by step.	**78.7**
2		First, (*1)	77.3
3		Let's think about this logically.	74.5
4		Let's solve this problem by splitting it into steps. (*2)	72.2
5		Let's be realistic and think step by step.	70.8
6		Let's think like a detective step by step.	70.3
7		Let's think	57.5
8		Before we dive into the answer,	55.7
9		The answer is after the proof.	45.7
10	misleading	Don't think. Just feel.	18.8
11		Let's think step by step but reach an incorrect answer.	18.7
12		Let's count the number of "a" in the question.	16.7
13		By using the fact that the earth is round,	9.3
14	irrelevant	By the way, I found a good restaurant nearby.	17.5
15		Abrakadabra!	15.5
16		It's a beautiful day.	13.1
-		(Zero-shot)	17.7

Adding New Weights: Prompt-Tuning

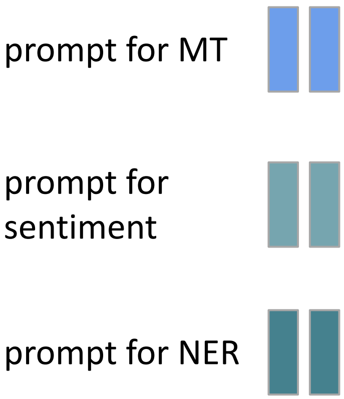
Can we learn the prompt embedding instead?

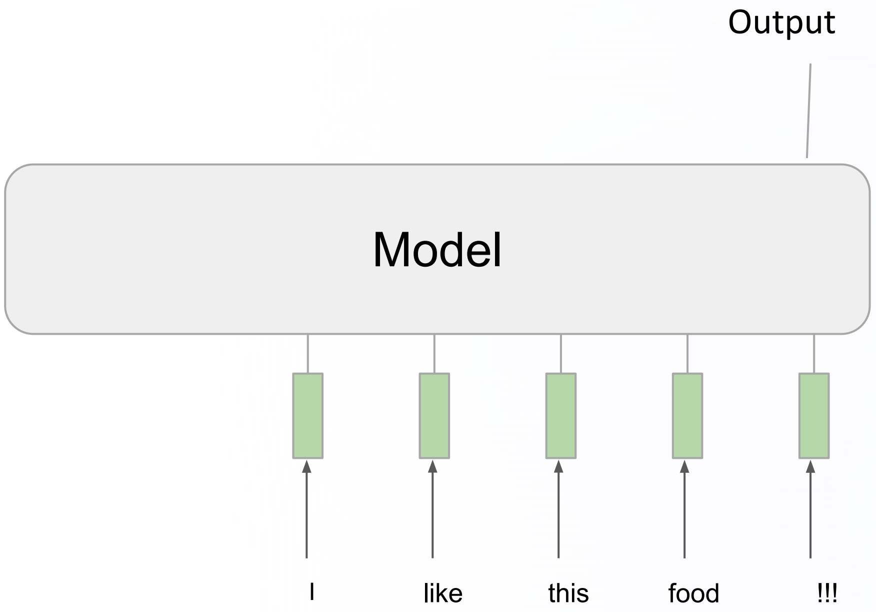
The model is frozen except for these new tokens



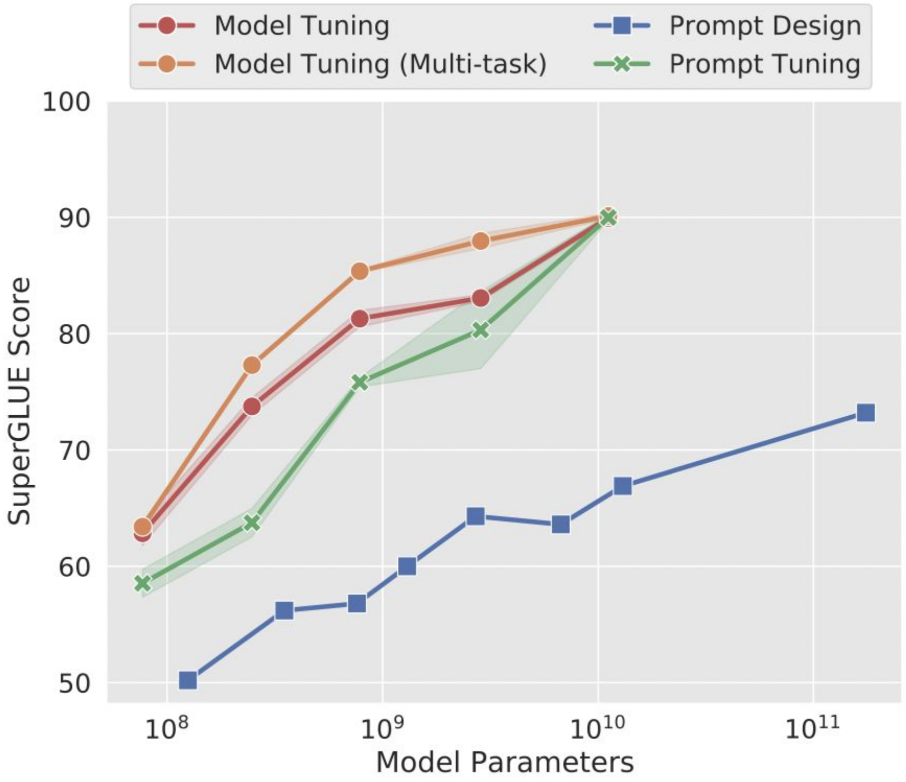
Adding New Weights: Prompt-Tuning

Also modular



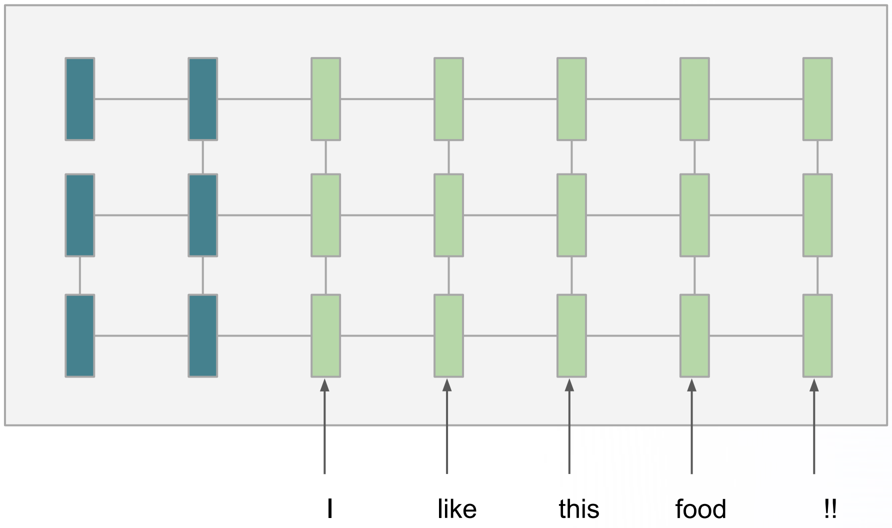


Adding New Weights: Prompt-Tuning



Adding New Weights: Prefix-Tuning

Conceptually similar to prompt-tuning, but we do it for every layer instead of just the embedding



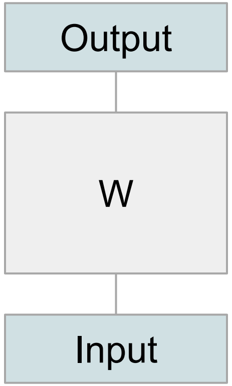
Adapters: Recap

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

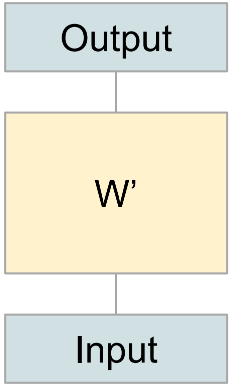
LoRA [Hu et al., 2021]

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

before finetuning



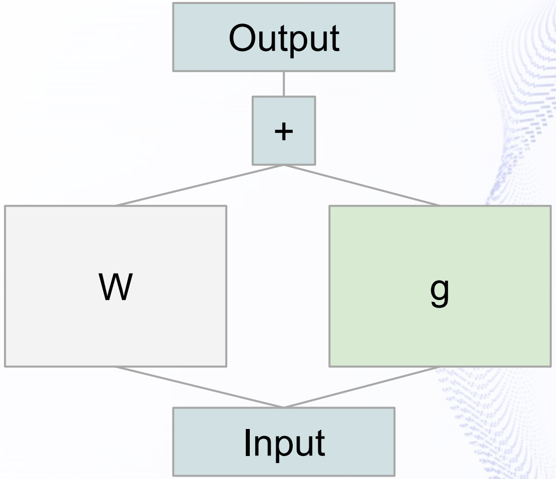
$O = f(Wx)$



$O = f(W'x)$

after finetuning

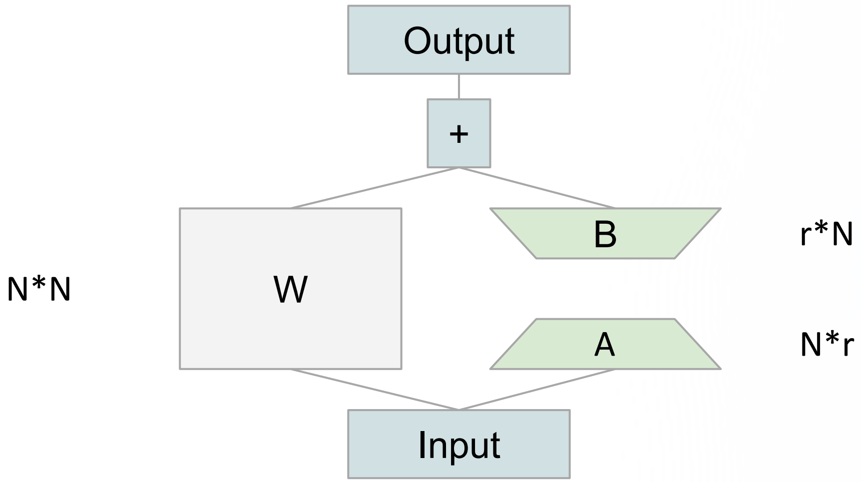
equals to →



$O = f(Wx + gw)$
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



LoRA

How small is r?

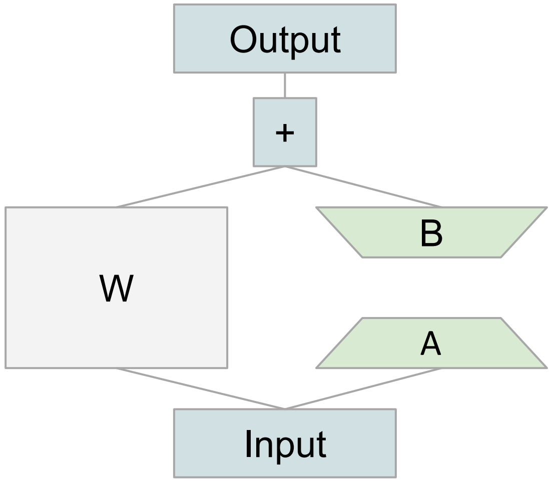
7.2 WHAT IS THE OPTIMAL RANK r FOR LORA?

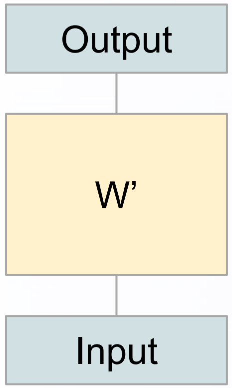
We turn our attention to the effect of rank r on model performance. We adapt $\{W_q, W_v\}$, $\{W_q, W_k, W_v, W_c\}$, and just W_q for a comparison.

	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**

LoRA: Performance

DEMO

CONCLUSION

Knowledge Distillation → Train smaller models by learning from a larger (teacher) model

PEFT → Train a big model with less memory usage

CONCLUSION

More on efficiency side!

- Mixture of Expert → Scale up the number of parameters without adding compute
- Linear Models → Attention mechanism in Transformers is quadratic, can we make it linear?
- Quantization → Faster and smaller with a cost of numerical precision
- Early Exiting → We don't always need to use all layers
- … (and more!)