# Parameter Efficient Fine Tuning (PEFT) of LLMs

DL4DS – Spring 2024

# Last time…

We looked at ways of improving LLM performance via prompting strategies such as

- Chain of Thought, Tree of Thought

and through

- Retrieval augmentation

# Today…

We look at ways to improve model performance through *finetuning* the model

- full model fine tuning

- parameter efficient fine tuning

# Topics

- Full finetuning
- Low rank adaptation
- Prompt tuning

# Topics

- Full finetuning
- Low rank adaptation
- Prompt tuning

# Model Training in the Transformer Era

Large-scale pretraining on generic internet-scale data

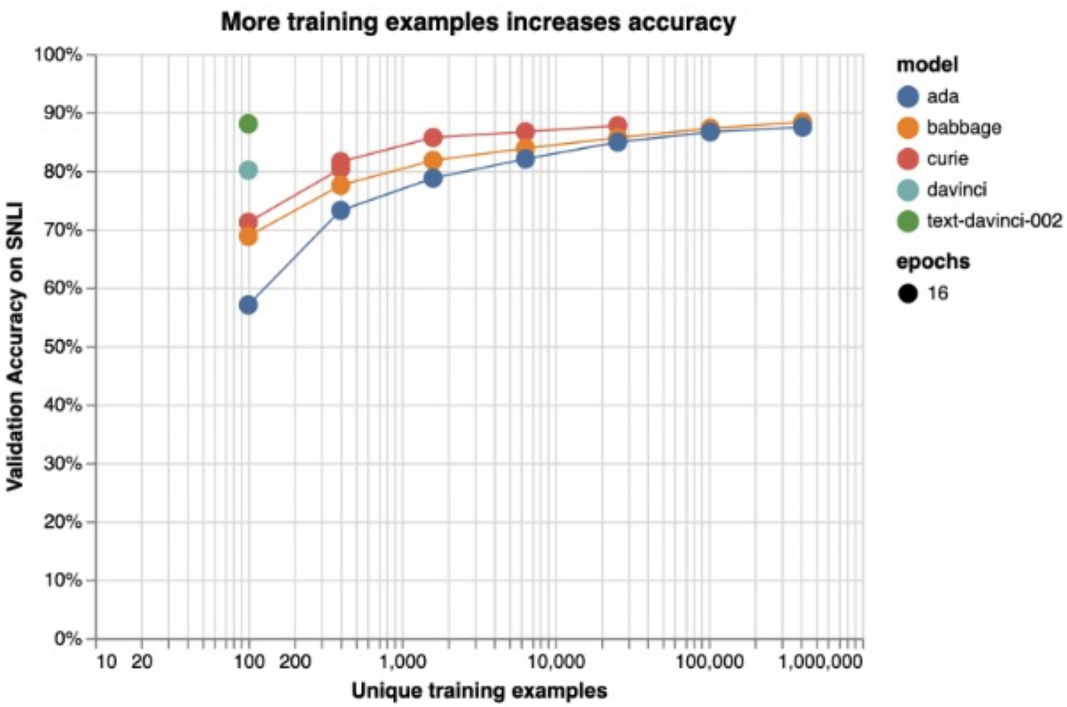
Fine-tuning to downstream tasks with smaller dataset

ChatGPT

# Model Finetuning

- Large foundation models are pre-trained on general tasks
- Might not do as well on specialized tasks
  - Try prompt engineering and retrieval augmentation first
- Good news: can fine tune model with much smaller dataset to adapt to downstream tasks
- Fine tuned model is same size as original.
  - Resource Intensive: Can take very large memory and compute resources to fine tune
  - Storage Demands: If you have $n$ downstream tasks, you will have $n$ copies of your large model.

# Full Finetuning Example




Text classification performance on the [Stanford Natural Language Inference (SNLI) Corpus](). Ordered pairs of sentences are classified by their logical relationship: either contradicted, entailed (implied), or neutral. Default fine-tuning parameters were used when not otherwise specified.

https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/fine-tuning

# 🤗 HuggingFace – Fine-tune Pretrained Model Tutorials

- Finetune for Sentiment Analysis Example (broken??)
  - https://huggingface.co/docs/transformers/training
  - Finetune bert-base-cased (109M params, FP32, 436MB) on Yelp review dataset (650K reviews, 323 MB)
- Finetune for text classification example
  - https://github.com/huggingface/notebooks/blob/main/examples/text_classification.ipynb
  - preprocess the data and fine-tune a pretrained model on any GLUE task
- Finetune for question answering
  - https://github.com/huggingface/notebooks/blob/main/examples/question_answering.ipynb
  - preprocess the data and fine-tune a pretrained model on SQUAD

# Model Finetuning Drawbacks

- Fine tuned model is same size as original.
  - Resource Intensive: Can take very large memory and compute resources to fine tune
  - Storage Demands: If you have $n$ downstream tasks, you will have $n$ copies of your large model

# Model Finetuning Drawbacks

- Fine tuned model is same size as original.
  - Resource Intensive: Can take very large memory and compute resources to fine tune
  - Storage Demands: If you have $n$ downstream tasks, you will have $n$ copies of your large model

Solution is to update aspects of the model, rather than entire model

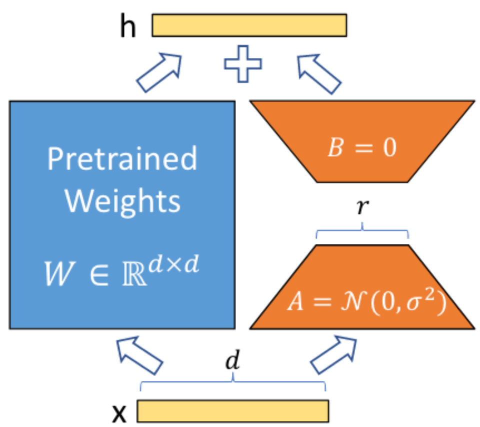
- Low rank adaptation of the weight updates -- LoRA
- Train and concatenated soft prompts -- Prompt Tuning

# Topics

- Full finetuning
- Low rank adaptation
- Prompt tuning

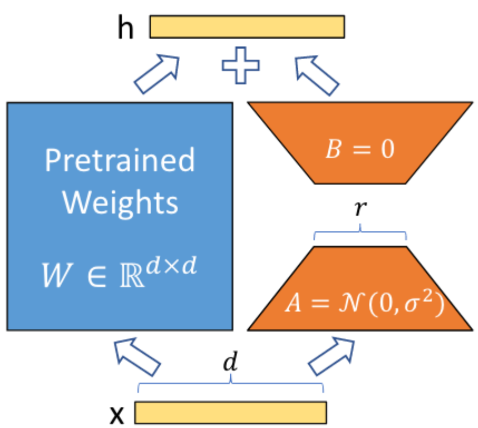
# Low Rank Adaptation

- Deploying independent instances of downstream fine-tuned models can be prohibitive (e.g. GPT3, 175B params, 700GB@fp32)
- Instead, freeze the pre-trained model and inject *trainable rank decomposition matrices* into each layer
- Reduce trainable parameters by 10,000x!!
- On-par or better than finetuning on RoBERTa, DeBERTa, GPT-2 and GPT-3




E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

# Low Rank Adaptation




- Aghajanyan et al show that pretrained language models have a low "intrinsic dimension"
- Updates to weight matrices likely have a low "intrinsic rank" during training
- Found that even very low rank (e.g. r=1 or2) with GPT-3 175B is effective where full rank (embedding dimension) is 12,288


E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685
A. Aghajanyan et al., "Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning". arXiv:2012.13255 [cs], December 2020. URL http://arxiv.org/abs/2012.13255.

# Reminder: Rank of a Matrix

- The number of linearly independent rows or columns of a matrix
- Determines the dimension of the vector space spanned by the column vectors
- A measure of "dimensionality"

# LoRA: Method

Say you have pre-trained weights,

$$W_0 \in \mathbb{R}^{d \times k}$$

Represent update with a low rank decomposition

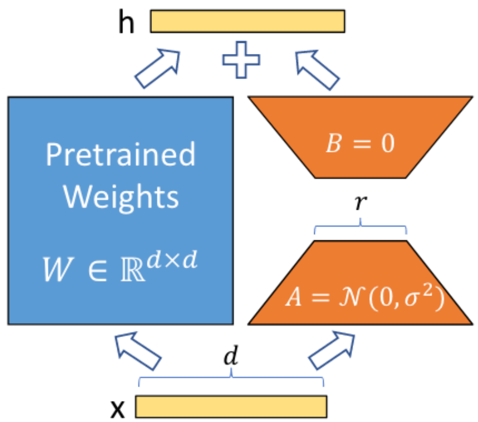
$$W_0 + \Delta W = W_0 + BA\,,$$

where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ and the rank $r \ll \min(d, k)$, is much less than the full rank.

For updates,

$$h = (W_0 + \Delta W)x = W_0 x + \Delta W x = W_0 x + BAx$$

Initialize A to random gaussian and B to zero



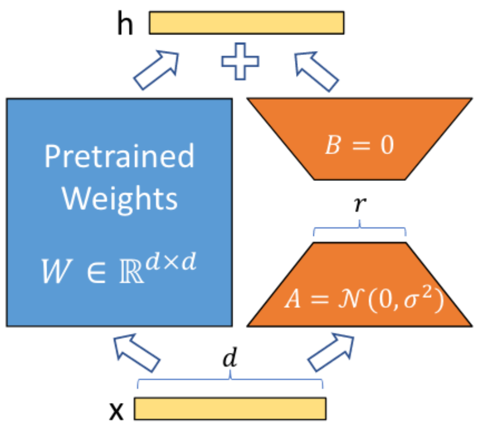

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

# LoRA: Method

LoRA can be viewed as a generalization of full finetuning, since using full rank = full finetuning

Updates:

$$h = (W_0 + \Delta W)x = W_0 x + \Delta W x = W_0 x + BAx$$




Generally only applied to $W_q$ and $W_v$ matrices.

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

# LoRA Results / Comparisons

| 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 | $87.5_{\pm.3}$ | **95.1**$_{\pm.2}$ | $89.7_{\pm.7}$ | $63.4_{\pm1.2}$ | **93.3**$_{\pm.3}$ | $90.8_{\pm.1}$ | **86.6**$_{\pm.7}$ | **91.5**$_{\pm.2}$ | **87.2** |
| RoB$_{large}$ (FT)* | 355.0M | 90.2 | **96.4** | **90.9** | 68.0 | 94.7 | **92.2** | 86.6 | 92.4 | 88.9 |
| RoB$_{large}$ (LoRA) | 0.8M | **90.6**$_{\pm.2}$ | $96.2_{\pm.5}$ | **90.9**$_{\pm1.2}$ | **68.2**$_{\pm1.9}$ | **94.9**$_{\pm.3}$ | $91.6_{\pm.1}$ | **87.4**$_{\pm2.5}$ | **92.6**$_{\pm.2}$ | **89.0** |
| RoB$_{large}$ (Adpt$^{P}$)† | 3.0M | $90.2_{\pm.3}$ | $96.1_{\pm.3}$ | $90.2_{\pm.7}$ | **68.3**$_{\pm1.0}$ | **94.8**$_{\pm.2}$ | **91.9**$_{\pm.1}$ | $83.8_{\pm2.9}$ | $92.1_{\pm.7}$ | 88.4 |
| RoB$_{large}$ (Adpt$^{P}$)† | 0.8M | **90.5**$_{\pm.3}$ | **96.6**$_{\pm.2}$ | $89.7_{\pm1.2}$ | $67.8_{\pm2.5}$ | **94.8**$_{\pm.3}$ | $91.7_{\pm.2}$ | $80.1_{\pm2.9}$ | $91.9_{\pm.4}$ | 87.9 |
| RoB$_{large}$ (Adpt$^{H}$)† | 6.0M | $89.9_{\pm.5}$ | $96.2_{\pm.3}$ | $88.7_{\pm2.9}$ | $66.5_{\pm4.4}$ | $94.7_{\pm.2}$ | $92.1_{\pm.1}$ | $83.4_{\pm1.1}$ | $91.0_{\pm1.7}$ | 87.8 |
| RoB$_{large}$ (Adpt$^{H}$)† | 0.8M | $90.3_{\pm.3}$ | $96.3_{\pm.5}$ | $87.7_{\pm1.7}$ | $66.3_{\pm2.0}$ | $94.7_{\pm.2}$ | $91.5_{\pm.1}$ | $72.9_{\pm2.9}$ | $91.5_{\pm.5}$ | 86.4 |
| RoB$_{large}$ (LoRA)† | 0.8M | **90.6**$_{\pm.2}$ | $96.2_{\pm.5}$ | **90.2**$_{\pm1.0}$ | $68.2_{\pm1.9}$ | **94.8**$_{\pm.3}$ | $91.6_{\pm.2}$ | **85.2**$_{\pm1.1}$ | **92.3**$_{\pm.5}$ | **88.6** |
| DeB$_{XXL}$ (FT)* | 1500.0M | 91.8 | **97.2** | 92.0 | 72.0 | **96.0** | 92.7 | 93.9 | 92.9 | 91.1 |
| DeB$_{XXL}$ (LoRA) | 4.7M | **91.9**$_{\pm.2}$ | $96.9_{\pm.2}$ | **92.6**$_{\pm.6}$ | **72.4**$_{\pm1.1}$ | **96.0**$_{\pm.1}$ | **92.9**$_{\pm.1}$ | **94.9**$_{\pm.4}$ | **93.0**$_{\pm.2}$ | **91.3** |

GLUE benchmark – measure across 9 language tasks
BitFit – train only the bias vectors
Adpt – Inserts adaptation layer between self-attention and MLP module

E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models.” arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685
† indicates runs configured in a setup similar to Houlsby et al. (2019) for a fair comparison.

# LoRA Results / Comparisons

| Model & Method | # Trainable Parameters | E2E NLG Challenge BLEU | NIST | MET | ROUGE-L | CIDEr |
|---|---|---|---|---|---|---|
| GPT-2 M (FT)* | 354.92M | 68.2 | 8.62 | 46.2 | 71.0 | 2.47 |
| GPT-2 M (Adapter$^{\mathrm{L}}$)* | 0.37M | 66.3 | 8.41 | 45.0 | 69.8 | 2.40 |
| GPT-2 M (Adapter$^{\mathrm{L}}$)* | 11.09M | 68.9 | 8.71 | 46.1 | 71.3 | 2.47 |
| GPT-2 M (Adapter$^{\mathrm{H}}$) | 11.09M | $67.3_{\pm.6}$ | $8.50_{\pm.07}$ | $46.0_{\pm.2}$ | $70.7_{\pm.2}$ | $2.44_{\pm.01}$ |
| GPT-2 M (FT$^{\mathrm{Top2}}$)* | 25.19M | 68.1 | 8.59 | 46.0 | 70.8 | 2.41 |
| GPT-2 M (PreLayer)* | 0.35M | 69.7 | 8.81 | 46.1 | 71.4 | 2.49 |
| GPT-2 M (LoRA) | 0.35M | $\mathbf{70.4}_{\pm.1}$ | $\mathbf{8.85}_{\pm.02}$ | $\mathbf{46.8}_{\pm.2}$ | $\mathbf{71.8}_{\pm.1}$ | $\mathbf{2.53}_{\pm.02}$ |
| GPT-2 L (FT)* | 774.03M | 68.5 | 8.78 | 46.0 | 69.9 | 2.45 |
| GPT-2 L (Adapter$^{\mathrm{L}}$) | 0.88M | $69.1_{\pm.1}$ | $8.68_{\pm.03}$ | $46.3_{\pm.0}$ | $71.4_{\pm.2}$ | $\mathbf{2.49}_{\pm.0}$ |
| GPT-2 L (Adapter$^{\mathrm{L}}$) | 23.00M | $68.9_{\pm.3}$ | $8.70_{\pm.04}$ | $46.1_{\pm.1}$ | $71.3_{\pm.2}$ | $2.45_{\pm.02}$ |
| GPT-2 L (PreLayer)* | 0.77M | 70.3 | 8.85 | 46.2 | 71.7 | 2.47 |
| GPT-2 L (LoRA) | 0.77M | $\mathbf{70.4}_{\pm.1}$ | $\mathbf{8.89}_{\pm.02}$ | $\mathbf{46.8}_{\pm.2}$ | $\mathbf{72.0}_{\pm.2}$ | $2.47_{\pm.02}$ |

GPT-2 medium (M) and large (L) with different adaptation methods on the E2E NLG Challenge. For all metrics, higher is better. LoRA outperforms several baselines with comparable or fewer trainable parameters. Confidence intervals are shown for experiments we ran. * indicates numbers published in prior works.

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

# Understanding the Low-Rank Updates

1. Given a parameter budget constraint, which subset of weight matrices in a pre-trained Transformer should we adapt to maximize downstream performance?
2. Is the "optimal" adaptation matrix $\Delta W$ really rank-deficient? If so, what is a good rank to use in practice?

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

# 1) Which weight matrices to target?

| | # of Trainable Parameters = 18M | | | | | | |
|---|---|---|---|---|---|---|---|
| Weight Type<br>Rank $r$ | $W_q$<br>8 | $W_k$<br>8 | $W_v$<br>8 | $W_o$<br>8 | $W_q, W_k$<br>4 | $W_q, W_v$<br>4 | $W_q, W_k, W_v, W_o$<br>2 |
| WikiSQL ($\pm 0.5\%$) | 70.4 | 70.0 | 73.0 | 73.2 | 71.4 | **73.7** | **73.7** |
| MultiNLI ($\pm 0.1\%$) | 91.0 | 90.8 | 91.0 | 91.3 | 91.3 | 91.3 | **91.7** |

Validation accuracy on WikiSQL and MultiNLI after applying LoRA to different types of attention weights in GPT-3, given the same number of trainable parameters. Adapting both Wq and Wv gives the best performance overall. We find the standard deviation across random seeds to be consistent for a given dataset, which we report in the first column.

Rank of 16 on 2 matrices or even 4 on 4 matrices is sufficient.

E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

# 2) What is the optimal rank?

| | 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 |

“Validation accuracy on WikiSQL and MultiNLI with different rank r. To our surprise, a rank as small as one suffices for adapting both Wq and Wv on these datasets while training Wq alone needs a larger r.”

E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models.” arXiv, Oct. 16, 2021. http://arxiv.org/abs/2106.09685

An alternative to adapting model updates is to train a set of soft prompt tokens

# Topics

- Full finetuning
- Low rank adaptation
- Prompt tuning

# Prompt Tuning

- Prompt engineering can improve LLM performance but is very brittle
  - small change in words can have drastic impact on performance
  - show example
- Turns out you can learn a set of “soft tokens” that are prepended to the actual prompt which improves LLM performance
- Makes it much more robust to small changes

# Prompt Tuning

- P-Tuning: employ trainable continuous prompt embeddings in concatenation with discrete prompts

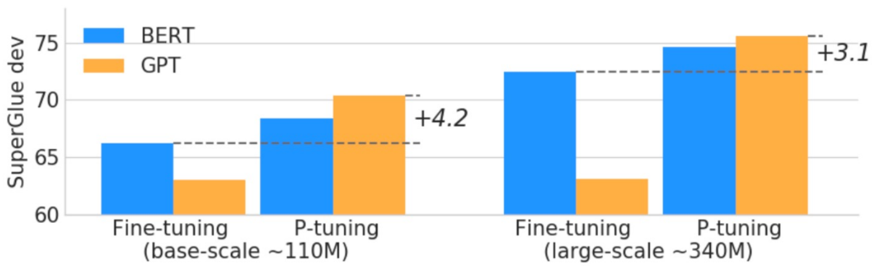



Figure 1: Average scores on 7 dev datasets of SuperGLUE using P-Tuning.

Instability of discrete prompts.

| Prompt | P@1 w/o PT | P@1 w/ PT |
|---|---|---|
| [X] is located in [Y]. *(original)* | 31.3 | 57.8 |
| [X] is located in which country or state? [Y]. | 19.8 | 57.8 |
| [X] is located in which country? [Y]. | 31.4 | 58.1 |
| [X] is located in which country? In [Y]. | 51.1 | 58.1 |

Results are precision@1 on LAMA-TREx P17 with BERT-base-cased.

X. Liu *et al.*, "GPT Understands, Too." arXiv, Oct. 25, 2023. http://arxiv.org/abs/2103.10385

# Prompt Tuning

- employs trainable continuous prompt embeddings in concatenation with discrete prompts given a discrete prompt as the input,
- P-Tuning concatenates continuous prompt embeddings with the discrete prompt tokens and feeds them as the input to the language model.
- The continuous prompts are updated by backpropagation to optimize the task objective.

Incorporate a certain degree of learnability into the input, which may learn to offset the effects of minor changes in discrete prompts to improve training stability

X. Liu *et al.*, "GPT Understands, Too." arXiv, Oct. 25, 2023. http://arxiv.org/abs/2103.10385

# p-tuning methodology

- Let $[\mathrm{D}_i]$ be a discrete prompt token.
- Each prompt can be described as a template

$$T = \{[\mathrm{D}_{0:i}], \mathrm{x}, [\mathrm{D}_{(i+1):j}], \mathrm{y}, [\mathrm{D}_{(j+1):k}]\}$$

which could organize the labeled data (including the inputs x and the label y) into a sequence of text tokens, such that the task could be reformulated as filling in the blanks of the input text.

- "The capital of [INPUT] is [LABEL]."
  - labeled data "(Britain, London)"
- Both discrete prompts and discrete data are together mapped into input embeddings:

$$\{\mathrm{e}(D_0) \ldots \mathrm{e}(D_i), \mathrm{e}(x_0), \ldots, \mathrm{e}(x_n), \ldots, \mathrm{e}(D_k)\}$$

through the pretrained embedding layer, where $e \in \mathbb{R}^{|\mathcal{V}| \times d}$.

- we propose P-Tuning that uses continuous prompt embeddings

X. Liu *et al.*, "GPT Understands, Too." arXiv, Oct. 25, 2023. http://arxiv.org/abs/2103.10385

# p-tuning methodology

- Proposes continuous prompt embeddings
- Let $[P_i]$ be the $i^{th}$ continuous prompt embedding.
- The prompt template for P-Tuning is as follows:

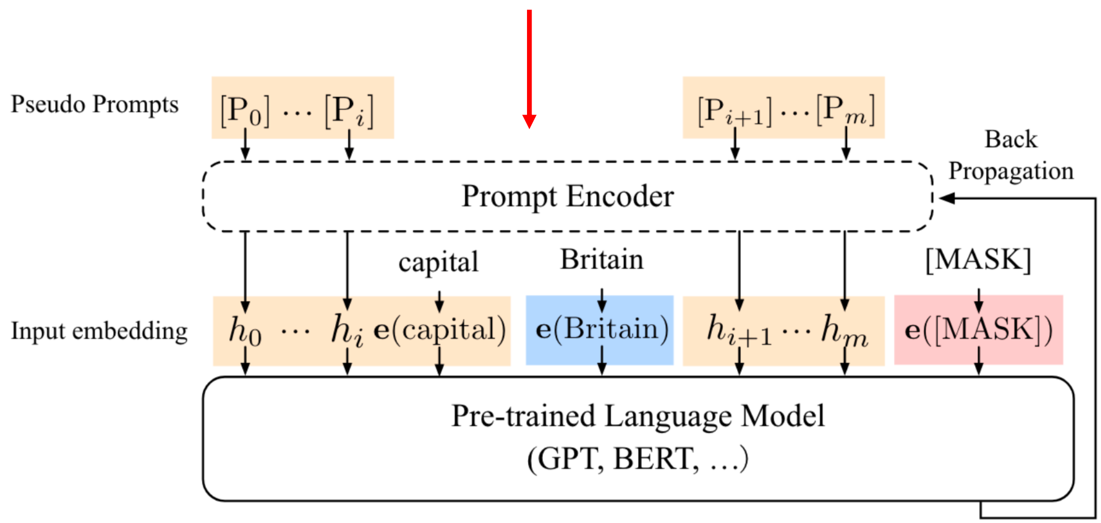
$$T = \{[\mathrm{P}_{0:i}], \mathrm{x}, [\mathrm{P}_{(i+1):j}], \mathrm{y}, [\mathrm{P}_{(j+1):k}]\}$$

- P-Tuning leverages an extra embedding function $f: [\mathrm{P}_i] \rightarrow h_i$ to map the template to

$$\{h_0, \ldots, h_i, e(x), h_{i+1}, \ldots, h_j, e(y), h_{i+1}, \ldots, h_k$$

- Finally, we update the embeddings $\{P_i\}_{i=1}^{k}$ to optimize a task loss function.

LSTM or MLP to model the dependency between continuous prompt embeddings





X. Liu *et al.*, "GPT Understands, Too." arXiv, Oct. 25, 2023. http://arxiv.org/abs/2103.10385

# Discrete Prompt Searching vs P-Tuning

| Prompt type | Model | P@1 |
|---|---|---|
| Original (MP) | BERT-base | 31.1 |
| | BERT-large | 32.3 |
| | E-BERT | 36.2 |
| Discrete | LPAQA (BERT-base) | 34.1 |
| | LPAQA (BERT-large) | 39.4 |
| | AutoPrompt (BERT-base) | <u>43.3</u> |
| P-tuning | BERT-base | 48.3 |
| | BERT-large | **50.6** |

| Model | MP | P-tuning |
|---|---|---|
| BERT-base (109M) | 31.7 | 52.3 (+20.6) |
| -AutoPrompt (Shin et al., 2020) | - | 45.2 |
| BERT-large (335M) | 33.5 | 54.6 (+21.1) |
| RoBERTa-base (125M) | 18.4 | 49.3 (+30.9) |
| -AutoPrompt (Shin et al., 2020) | - | 40.0 |
| RoBERTa-large (355M) | 22.1 | 53.5 (+31.4) |
| GPT2-medium (345M) | 20.3 | 46.5 (+26.2) |
| GPT2-xl (1.5B) | 22.8 | 54.4 (+31.6) |
| MegatronLM (11B) | 23.1 | **64.2** (+41.1) |

Table 3: Knowledge probing Precision@1 on LAMA-34k (left) and LAMA-29k (right). P-tuning outperforms all the discrete prompt searching baselines. (MP: Manual prompt; PT: P-tuning).

X. Liu *et al.*, "GPT Understands, Too." arXiv, Oct. 25, 2023. http://arxiv.org/abs/2103.10385

# Additional References


- X. Liu *et al.*, “P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks.” arXiv, Mar. 20, 2022. http://arxiv.org/abs/2110.07602

- B. Lester, R. Al-Rfou, and N. Constant, “The Power of Scale for Parameter-Efficient Prompt Tuning.” arXiv, Sep. 02, 2021. http://arxiv.org/abs/2104.08691

# 🤗 HuggingFace PEFT Resources

# HuggingFace PEFT

- Blog: 🤗 PEFT: Parameter-Efficient Fine-Tuning of Billion-Scale Models on Low-Resource Hardware
- Library: https://github.com/huggingface/peft

# 🤗 HuggingFace PEFT Library

Prepare a model for training with PEFT method

```
from transformers import AutoModelForSeq2SeqLM
from peft import get_peft_config, get_peft_model, LoraConfig, TaskType
model_name_or_path = "bigscience/mt0-large"
tokenizer_name_or_path = "bigscience/mt0-large"

peft_config = LoraConfig(
    task_type=TaskType.SEQ_2_SEQ_LM, inference_mode=False, r=8, lora_alpha=32, lora_dropout=0.1
)

model = AutoModelForSeq2SeqLM.from_pretrained(model_name_or_path)
model = get_peft_model(model, peft_config)
model.print_trainable_parameters()
"trainable params: 2359296 || all params: 1231940608 || trainable%: 0.19151053100118282"
```

Create PEFT config

Get the PEFT model based on config

Load a PEFT model for inference

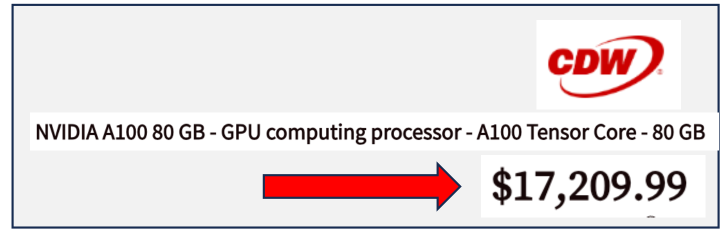
```
from peft import AutoPeftModelForCausalLM
from transformers import AutoTokenizer
import torch

model = AutoPeftModelForCausalLM.from_pretrained("ybelkada/opt-350m-lora").to("cuda")
tokenizer = AutoTokenizer.from_pretrained("facebook/opt-350m")

model.eval()
inputs = tokenizer("Preheat the oven to 350 degrees and place the cookie dough", return_tensor

outputs = model.generate(input_ids=inputs["input_ids"].to("cuda"), max_new_tokens=50)
print(tokenizer.batch_decode(outputs, skip_special_tokens=True)[0])

"Preheat the oven to 350 degrees and place the cookie dough in the center of the oven. In a la
```

Get the PEFT model

Use it like a regular model

https://github.com/huggingface/peft?tab=readme-ov-file#quickstart

# 🤗 HuggingFace PEFT Library




**High performance on consumer hardware**

Consider the memory requirements for training the following models on the ought/raft/twitter_complaints dataset with an A100 80GB GPU with more than 64GB of CPU RAM.

| Model | Full Finetuning | PEFT-LoRA PyTorch | PEFT-LoRA DeepSpeed with CPU Offloading |
|---|---|---|---|
| bigscience/T0_3B (3B params) | 47.14GB GPU / 2.96GB CPU | 14.4GB GPU / 2.96GB CPU | 9.8GB GPU / 17.8GB CPU |
| bigscience/mt0-xxl (12B params) | OOM GPU | 56GB GPU / 3GB CPU | 22GB GPU / 52GB CPU |
| bigscience/bloomz-7b1 (7B params) | OOM GPU | 32GB GPU / 3.8GB CPU | 18.1GB GPU / 35GB CPU |

| Submission Name | Accuracy |
|---|---|
| Human baseline (crowdsourced) | 0.897 |
| Flan-T5 (fully finetuned) | 0.892 |
| lora-t0-3b (LoRA) | 0.863 |

https://github.com/huggingface/peft?tab=readme-ov-file#high-performance-on-consumer-hardware

# 🤗 HuggingFace PEFT Library

## Diffusers

| Model | Full Finetuning | PEFT-LoRA | PEFT-LoRA with Gradient Checkpointing |
|---|---|---|---|
| CompVis/stable-diffusion-v1-4 | 27.5GB GPU / 3.97GB CPU | 15.5GB GPU / 3.84GB CPU | 8.12GB GPU / 3.77GB CPU |

Take a look at the examples/lora_dreambooth/train_dreambooth.py training script to try training your own Stable Diffusion model with LoRA, and play around with the smangrul/peft-lora-sd-dreambooth Space which is running on a T4 instance. Learn more about the PEFT integration in Diffusers in this tutorial.

https://github.com/huggingface/peft?tab=readme-ov-file#diffusers

# Next Time

- back to book sequence on
  - unsupervised learning
  - GANs
  - VAEs
  - Diffusion Models
  - graph neural nets
  - etc.

Feedback