석사학위논문심사 발표

Bayesian Transfer Learning for Soft Prompt Tuning

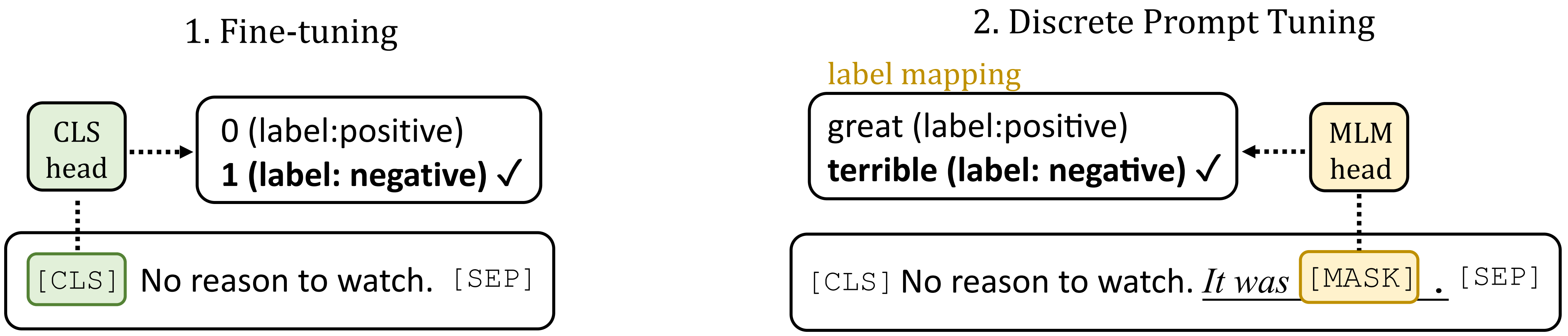
한국과학기술원 김재철AI대학원 석사과정

이 해 주

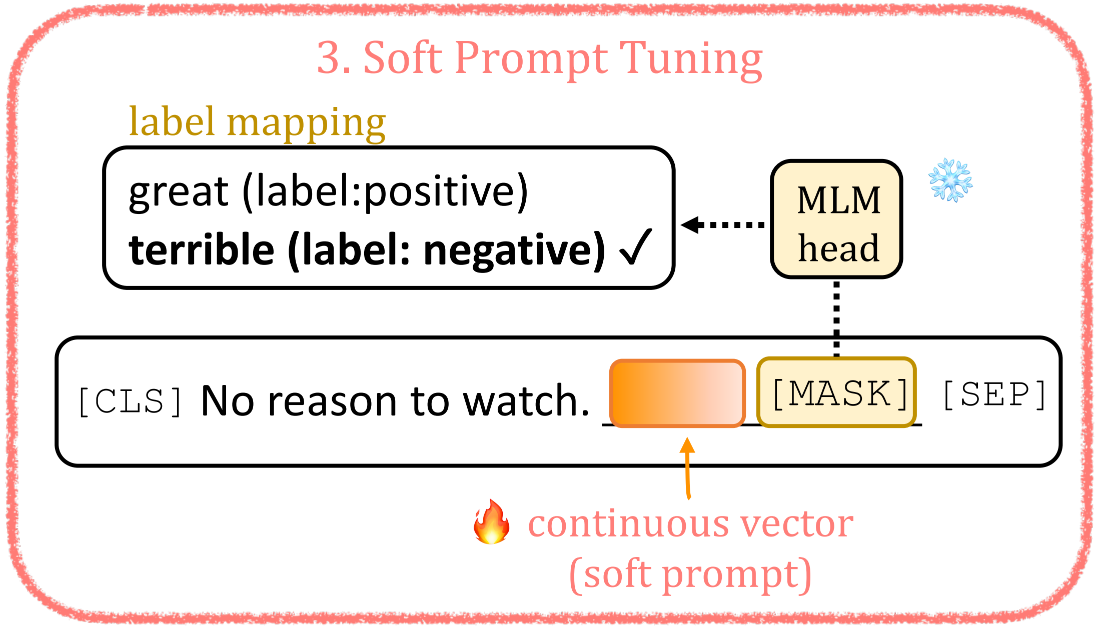
2023-06-13

Contents

Soft Prompt Tuning – Concept



1. Fine-tuning
CLS head
0 (label:positive)
1 (label: negative) ✓
[CLS] No reason to watch. [SEP]
2. Discrete Prompt Tuning
label mapping
great (label:positive)
terrible (label: negative) ✓
MLM head
[CLS] No reason to watch. It was [MASK] . [SEP]



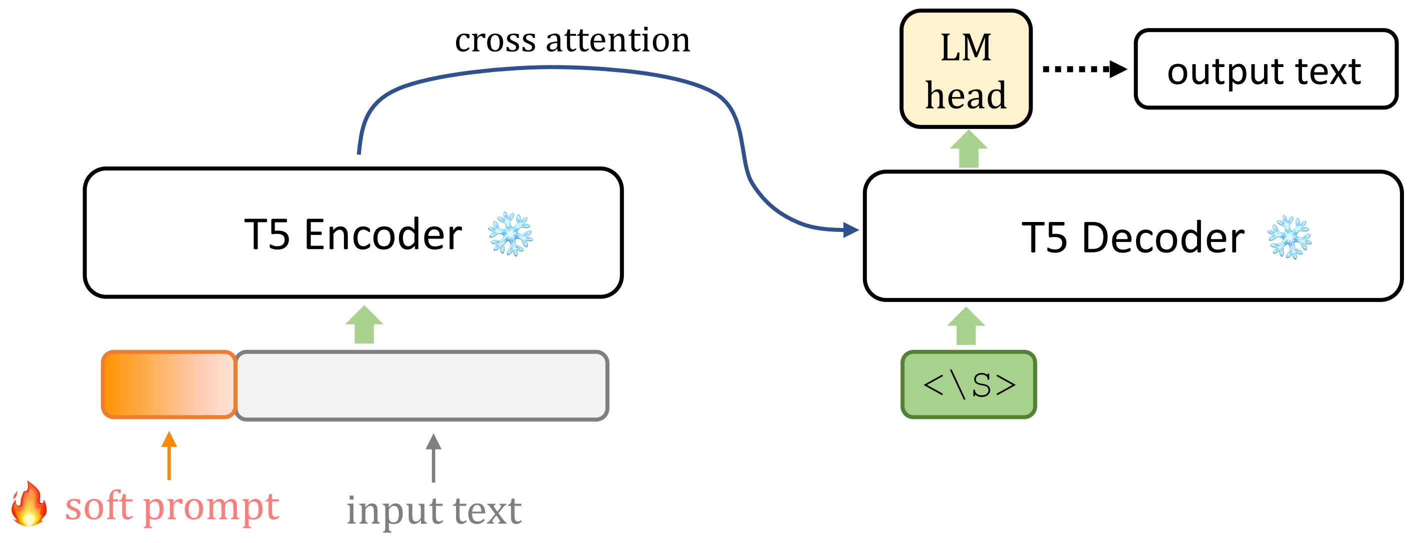
3. Soft Prompt Tuning
label mapping
great (label:positive)
terrible (label: negative) ✓
MLM head
[CLS] No reason to watch. [MASK] [SEP]
continuous vector (soft prompt)

Soft Prompt Tuning[1]:

- Optimize continuous vector appended to input, while freezing pretrained language model (PLM)
- The continuous vector (soft prompt) invokes PLM to solve down-stream task

[1] Lester et al., "The Power of Scale for Parameter-Efficient Prompt Tuning", EMNLP 2021

Soft Prompt Tuning – Schematic



- Most Soft Prompt Tuning works experiment with T5[1] model as PLM
  - Prepend soft prompt to the input text
  - Only tune the soft prompt, while freezing T5

[1] Raffel et al., " Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", ArXiv 2019

Pros and Cons of Soft Prompt Tuning

- Pros:
  - Parameter-efficient: Soft Prompt Tuning only updates soft prompt, while leaving LM frozen
  - Easy to implement: you only need to concatenate soft prompt to the input, while other Parameter-efficient methods[1,2,3] needs more efforts to implement

- Cons:
  - Low performance: Soft Prompt Tuning underperforms compared to Fine-tuning
  - Long training time: Soft Prompt Tuning takes longer training time than Fine-tuning

[1] Houlsby et al., "Parameter-Efficient Transfer Learning for NLP", ICML 2019
[2] Ben-Zaken et al., "BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models", ACL 2022
[3] Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models", ICLR 2022

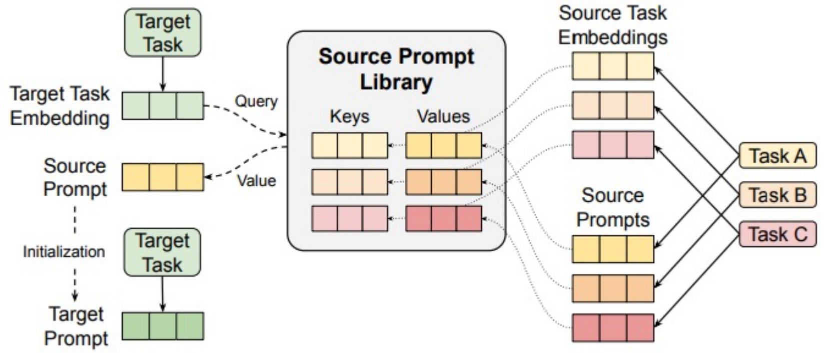
Transfer Learning for Soft Prompt Tuning

- To cope with the cons of Soft Prompt Tuning, several works[1,2,3] transfer soft prompt learned on source tasks to target task(s)
  - This approach assumes transferability between NLP tasks, meaning that some NLP tasks can help solving other similar NLP tasks
  - SPoT[1], ATTEMPT[2], and MPT[3] adopt transfer learning for improvement
    - They pre-train a soft prompt for each source task, then retrieve or aggregate the source task prompts to initialize target task prompt

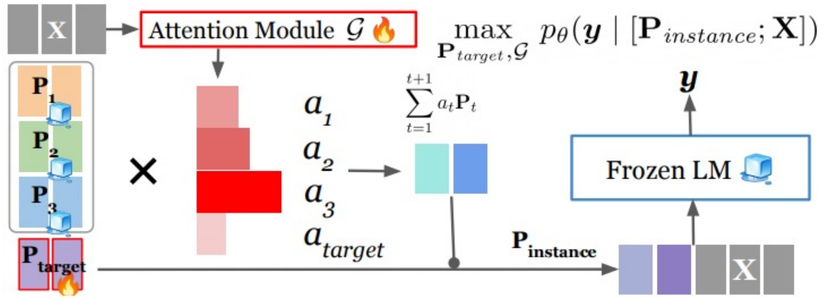
[1] Vu et al., "Better Frozen Model Adaptation through Soft Prompt Transfer", ACL 2022
[2] Asai et al., "ATTEMPT: Parameter-Efficient Multi-task Tuning via Attentional Mixtures of Soft Prompts", EMNLP 2022
[3] Wang et al., "Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning", ICLR 2023

Previous Works – SPoT[1], ATTEMPT[2], MPT[3]

- SPoT[1]
  - Initialize target task prompt (target prompt) with most relevant source task prompt (source prompt)
  - Relevance is measured by cosine similarity between prematurely trained source prompt and target prompt



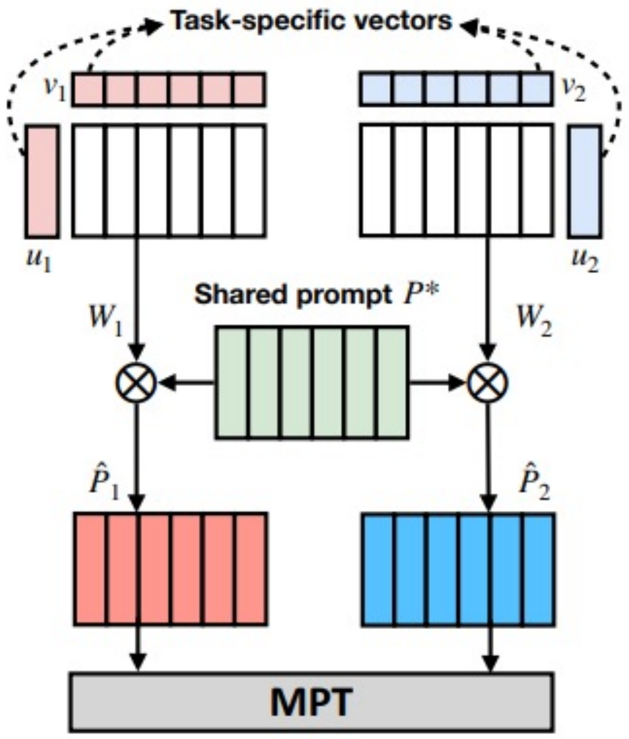
- ATTEMPT[2]
  - Initialize target prompt with addition of
    (1) attentional mixture of frozen source prompts, and
    (2) trainable prompt designated for the target task


X
Attention Module $\mathcal{G}$
$\max_{\mathbf{P}_{target},\mathcal{G}} p_\theta(\boldsymbol{y} \mid [\mathbf{P}_{instance}; \mathbf{X}])$
$\mathbf{P}_1$
$\mathbf{P}_2$
$\mathbf{P}_3$
$\mathbf{P}_{target}$
×
a_1
a_2
a_3
a_{target}
$\sum_{t=1}^{t+1} a_t \mathbf{P}_t$
$\boldsymbol{y}$
Frozen LM
$\mathbf{P}_{instance}$
X

[1] Vu et al., "Better Frozen Model Adaptation through Soft Prompt Transfer", ACL 2022
[2] Asai et al., "ATTEMPT: Parameter-Efficient Multi-task Tuning via Attentional Mixtures of Soft Prompts", EMNLP 2022
[3] Wang et al., "Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning", ICLR 2023

Previous Works – SPoT[1], ATTEMPT[2], MPT[3] (cont'd)

- MPT[3]
  - Decompose source prompt into
    (1) a full-rank matrix which is shared across other source prompts, and
    (2) source-task-specific low-rank matrix
    to minimize interference between tasks
  - Further enhance performance via distillation



[1] Vu et al., "Better Frozen Model Adaptation through Soft Prompt Transfer", ACL 2022
[2] Asai et al., "ATTEMPT: Parameter-Efficient Multi-task Tuning via Attentional Mixtures of Soft Prompts", EMNLP 2022
[3] Wang et al., "Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning", ICLR 2023

Notations

- Notations
  - $\boldsymbol{\theta}^{\mathcal{S}}$ denotes source task prompt (source prompt), $\boldsymbol{\theta}^{\mathcal{T}}$ denotes target task prompt (target prompt)
  - $\mathcal{D}_k^{\mathcal{S}}$ and $\mathcal{D}_k^{\mathcal{T}}$ denote the k-th task data of source tasks, k-th task data of target tasks (i.e. i–th datapoint of $\mathcal{D}_k^{\mathcal{S}}$ is $(\mathbf{Y}_i^k, \mathbf{X}_i^k)$, where X and Y are input text and label)
  - If we want to optimize a prompt w.r.t. k-th source task, we usually perform Maximum Likelihood Estimation (MLE)

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{|\mathcal{D}_k^{\mathcal{S}}|} \mathbb{P}_{\mathrm{LM}}(\mathbf{Y}_i^k \mid [\boldsymbol{\theta}; \mathbf{X}_i^k])$$

  - To simplify notation, we use $\mathbb{P}(\mathcal{D}_k^{\mathcal{S}}|\boldsymbol{\theta})$ instead of $\prod_i \mathbb{P}_{\mathrm{LM}}(\mathbf{Y}_i^k|[\boldsymbol{\theta}; \mathbf{X}_i^k])$
  - When the prior of each source task is uninformative (uniform), MLE coincides with MAP, as $\mathbb{P}(\boldsymbol{\theta}|\mathcal{D}_k^{\mathcal{S}}) \propto \mathbb{P}(\mathcal{D}_k^{\mathcal{S}}|\boldsymbol{\theta})\, \mathbb{P}(\boldsymbol{\theta})$

Transfer Learning on Soft Prompt Tuning

- Transfer learning problem statement
  - Under transfer learning paradigm, sufficient amount of data of source tasks will lead to good initialization point for target task
  - Therefore, for good transfer, we have to find a point which maximizes *global posterior distribution* over source tasks, which is proportional to the product of individual source task posteriors (under uniform priors):

$$\text{A good init point } = \underset{\boldsymbol{\theta}^{\mathcal{S}}}{\operatorname{argmax}} \mathbb{P}(\boldsymbol{\theta}^{\mathcal{S}} \mid \mathcal{D}^{\mathcal{S}})$$

$$\text{where } \mathbb{P}(\boldsymbol{\theta}^{\mathcal{S}} \mid \mathcal{D}^{\mathcal{S}}) \propto \prod_{k=1}^{K} \mathbb{P}(\boldsymbol{\theta}^{\mathcal{S}} \mid \mathcal{D}_k^{\mathcal{S}})$$

Previous Works are not Ideal for Transfer Learning

- Previous works initialize target prompt by either
  - 1) retrieving the most relevant source prompt (SPoT[1])
  - 2) weighted average of source prompts + random init vector (ATTEMPT[2], MPT[3])
- However, we can easily show that these fail, even with cases where the posteriors of individual source tasks are simple Gaussian
  - Suppose $\mathbb{P}(\boldsymbol{\theta}|\mathcal{D}_k^{\mathcal{S}})$ has mean of $\boldsymbol{\mu}_k$ and covariance of $\boldsymbol{\Sigma}_k$
  - If we perform MLE (under uniform prior), the prompt of k-th source task will be very close to $\boldsymbol{\mu}_k$
  - A product of multivariate Gaussians is proportional to a Gaussian whose mean is:

$$\boldsymbol{\mu} = \left(\sum_{k=1}^{K} \boldsymbol{\Sigma}_k^{-1}\right)^{-1}\left(\sum_{k=1}^{K} \boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k\right)$$

[1] Vu et al., "Better Frozen Model Adaptation through Soft Prompt Transfer", ACL 2022
[2] Asai et al., "ATTEMPT: Parameter-Efficient Multi-task Tuning via Attentional Mixtures of Soft Prompts", EMNLP 2022
[3] Wang et al., ”Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning", ICLR 2023

Previous Works are not Ideal for Transfer Learning (cont'd)

- However, unless covariances are identity matrices, it is unlikely that any convex combination of $\boldsymbol{\mu}_k$ is proportional to $\boldsymbol{\mu}$
  - Therefore, we cannot expect the weighted average of source prompts to be optimal point for target task adaptation

Method – Problem Statement

- We propose a Bayesian objective that incorporates transfer learning
  - Our objective is to maximize the posterior probability of target prompt:

$$\underset{\boldsymbol{\theta}^{\mathcal{T}}}{\operatorname{argmax}}\, \mathbb{P}(\mathcal{D}^{\mathcal{T}} \,|\, \boldsymbol{\theta}^{\mathcal{T}})\, \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}} \,|\, \mathcal{D}^{\mathcal{S}}) \quad \cdots \textbf{ Eq. 1}$$

    which can be thought as product of likelihood and prior,
    and also equivalent to conventional transfer learning with certain regularization

  - Since $\boldsymbol{\theta}^{\mathcal{T}}$ will be initialized using $\boldsymbol{\theta}^{\mathcal{S}}$, and $\boldsymbol{\theta}^{\mathcal{S}}$ is derived from $\mathcal{D}^{\mathcal{S}}$, we can rewrite Eq. 1 as follows (without argmax):

$$\mathbb{P}(\mathcal{D}^{\mathcal{T}}|\boldsymbol{\theta}^{\mathcal{T}}) \int_{\boldsymbol{\theta}^{\mathcal{S}}} \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}}|\boldsymbol{\theta}^{\mathcal{S}})\, \mathbb{P}(\boldsymbol{\theta}^{\mathcal{S}}|\mathcal{D}^{\mathcal{S}}) d\boldsymbol{\theta}^{\mathcal{S}}$$

Method – Derivation

- Bayesian objective that incorporates transfer learning (cont'd)

$$\mathbb{P}(\mathcal{D}^{\mathcal{T}}|\boldsymbol{\theta}^{\mathcal{T}}) \int_{\boldsymbol{\theta}^{\mathcal{S}}} \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}}|\boldsymbol{\theta}^{\mathcal{S}})\, \mathbb{P}(\boldsymbol{\theta}^{\mathcal{S}}|\mathcal{D}^{\mathcal{S}}) d\boldsymbol{\theta}^{\mathcal{S}}$$

  - Above can be approximated by Monte-Carlo sampling:

$$\mathbb{P}(\mathcal{D}^{\mathcal{T}}|\boldsymbol{\theta}^{\mathcal{T}}) \cdot \mathop{\mathbb{E}}_{\boldsymbol{\theta}^{\mathcal{S}} \sim \mathbb{P}(\,\cdot\,|\mathcal{D}^{\mathcal{S}})} \left[\mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}}|\boldsymbol{\theta}^{\mathcal{S}})\right] \overset{\substack{\text{MC} \\ \text{Sampling}}}{\simeq} \mathbb{P}(\mathcal{D}^{\mathcal{T}}|\boldsymbol{\theta}^{\mathcal{T}}) \cdot \frac{1}{M} \sum_{i=1}^{M} \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}}|\boldsymbol{\theta}_i^{\mathcal{S}})$$

($\boldsymbol{\theta}_i^{\mathcal{S}}$ is sampled from $\mathbb{P}(\,\cdot\,|\mathcal{D}^{\mathcal{S}})$)

  - But how can we sample $\boldsymbol{\theta}_i^{\mathcal{S}}$ from $\mathbb{P}(\,\cdot\,|\mathcal{D}^{\mathcal{S}})$?

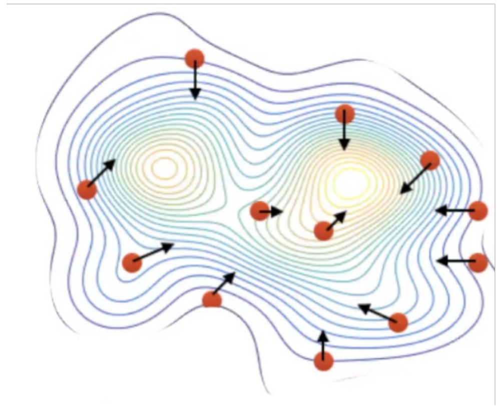
    → by Stein Variational Gradient Descent (SVGD)!

Stein Variational Gradient Descent (SVGD)

- SVGD[1] is a nonparametric variational inference method
  - Specifically, to obtain M samples from target distribution $p(\theta)$ where θ is neural net parameters, SVGD employs M instances of θ, $\Theta = \{\theta^m\}_{m=1}^M$, called *particles*
  - At iteration t, each particle $\theta_t \in \Theta_t$ is updated by following rule:

$$\theta_{t+1} \leftarrow \theta_t + \epsilon_t \phi(\theta_t) \quad \text{where} \quad \phi(\theta_t) = \frac{1}{M} \sum_{j=1}^{M} \left[k(\theta_t^j, \theta_t) \nabla_{\theta_t^j} \log p(\theta_t^j) + \nabla_{\theta_t^j} k(\theta_t^j, \theta_t) \right]$$

where ϵ_t is step-size and $k(x, x')$ is a positive-definite kernel (e.g. RBF)



[1] Liu et al., "Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm", NeurIPS 2016

Using SVGD to Sample from $\mathbb{P}(\cdot|\mathcal{D}^{\mathcal{S}})$

- The SVGD update during source task training would be:

$$\begin{aligned}\phi(\theta) &= \frac{1}{M}\sum_{j=1}^{M}\left[k(\theta_j,\theta)\nabla_{\theta_j}\log\mathbb{P}(\theta_j|\mathcal{D}^{\mathcal{S}}) + \nabla_{\theta_j}k(\theta_j,\theta)\right] \\ &= \frac{1}{M}\sum_{j=1}^{M}\left[k(\theta_j,\theta)\nabla_{\theta_j}\log\mathbb{P}(\mathcal{D}^{\mathcal{S}}|\theta_j) + \nabla_{\theta_j}k(\theta_j,\theta)\right]\end{aligned}$$

uniform prior

$$\left(\begin{aligned}\because \nabla_{\theta_j}\log\{\mathbb{P}(\mathcal{D}^{\mathcal{S}}|\theta_j)\cdot\mathbb{P}(\theta_j)\} \\ = \nabla_{\theta_j}\log\mathbb{P}(\mathcal{D}^{\mathcal{S}}|\theta_j)\end{aligned}\right)$$

- Therefore, we empirically use the minus of the gradient of LM loss for $\nabla_{\theta_j}\log\mathbb{P}(\mathcal{D}^{\mathcal{S}}|\theta_j)$

SVGD update rule

$$\theta_{t+1} \leftarrow \theta_t + \epsilon_t\phi(\theta_t) \quad \text{where} \quad \phi(\theta_t) = \frac{1}{M}\sum_{j=1}^{M}\left[k(\theta_t^j,\theta_t)\nabla_{\theta_t^j}\log p(\theta_t^j) + \nabla_{\theta_t^j}k(\theta_t^j,\theta_t)\right]$$

Method – Derivation (cont'd)

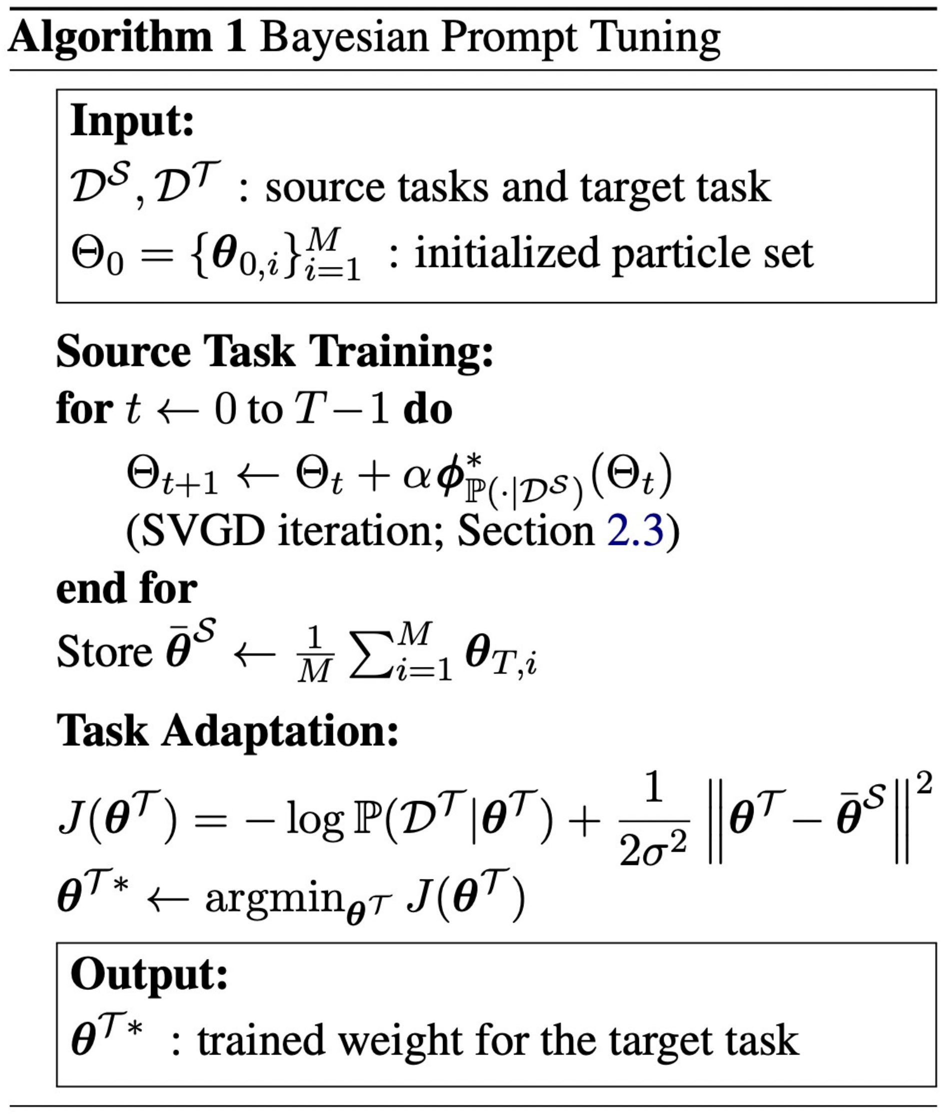
- Taking $\max$ and minus to Eq. 1 derives the following:
  - $$-\max_{\boldsymbol{\theta}^{\mathcal{T}}} \log(\mathbb{P}(\mathcal{D}^{\mathcal{T}} | \boldsymbol{\theta}^{\mathcal{T}}) \, \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}} | \mathcal{D}^{\mathcal{S}})) \simeq \min_{\boldsymbol{\theta}^{\mathcal{T}}} -\log \mathbb{P}(\mathcal{D}^{\mathcal{T}} | \boldsymbol{\theta}^{\mathcal{T}}) - \log \frac{1}{M} \sum_{i=1}^{M} \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}} | \boldsymbol{\theta}_i^{\mathcal{S}})$$
    $$\leq \min_{\boldsymbol{\theta}^{\mathcal{T}}} -\log \mathbb{P}(\mathcal{D}^{\mathcal{T}} | \boldsymbol{\theta}^{\mathcal{T}}) - \frac{1}{M} \sum_{i=1}^{M} \log \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}} | \boldsymbol{\theta}_i^{\mathcal{S}}) \quad \cdots \text{Eq. 2}$$
  - We minimize the *upper bound* of Eq. 2 to maximize Eq. 1
  - We design $\mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}} | \boldsymbol{\theta}_i^{\mathcal{S}})$ with simple Gaussian distribution with mean of $\boldsymbol{\theta}_i^{\mathcal{S}}$ and constant variance of $\Sigma_i = \sigma^2 \mathbf{I}$
  - Reason for fixed isotropic Gaussian: we lack prior information about the target task before adaptation process

Eq. 1
$$\underset{\boldsymbol{\theta}^{\mathcal{T}}}{\operatorname{argmax}} \, \mathbb{P}(\mathcal{D}^{\mathcal{T}} | \boldsymbol{\theta}^{\mathcal{T}}) \, \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}} | \mathcal{D}^{\mathcal{S}})$$

Method – Derivation (cont'd)

- Taking $\max$ and minus to Eq. 1 derives the following (cont'd):
  - $$\begin{aligned} -\max_{\boldsymbol{\theta}^{\mathcal{T}}} \log\left(\mathbb{P}(\mathcal{D}^{\mathcal{T}} \mid \boldsymbol{\theta}^{\mathcal{T}})\, \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}} \mid \mathcal{D}^{\mathcal{S}})\right) &\simeq \min_{\boldsymbol{\theta}^{\mathcal{T}}} -\log \mathbb{P}(\mathcal{D}^{\mathcal{T}} \mid \boldsymbol{\theta}^{\mathcal{T}}) - \log \frac{1}{M} \sum_{i=1}^{M} \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}} \mid \boldsymbol{\theta}_i^{\mathcal{S}}) \\ &\leq \min_{\boldsymbol{\theta}^{\mathcal{T}}} -\log \mathbb{P}(\mathcal{D}^{\mathcal{T}} \mid \boldsymbol{\theta}^{\mathcal{T}}) - \frac{1}{M} \sum_{i=1}^{M} \log \mathbb{P}(\boldsymbol{\theta}^{\mathcal{T}} \mid \boldsymbol{\theta}_i^{\mathcal{S}}) \quad \cdots \text{Eq. 2} \end{aligned}$$
  - We minimize the *upper bound* of Eq. 2 to maximize Eq. 1
  - By introducing additive constant C, the prior term of upper bound can be rewritten as: $$\frac{1}{2\sigma^2} \left\| \boldsymbol{\theta}^{\mathcal{T}} - \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{\theta}_i^{\mathcal{S}} \right\|^2 + C$$
  - At the start of target adaptation, we initialize $\boldsymbol{\theta}^{\mathcal{T}}$ with $\frac{1}{M} \sum_{i=1}^{M} \boldsymbol{\theta}_i^{\mathcal{S}}$

Algorithm

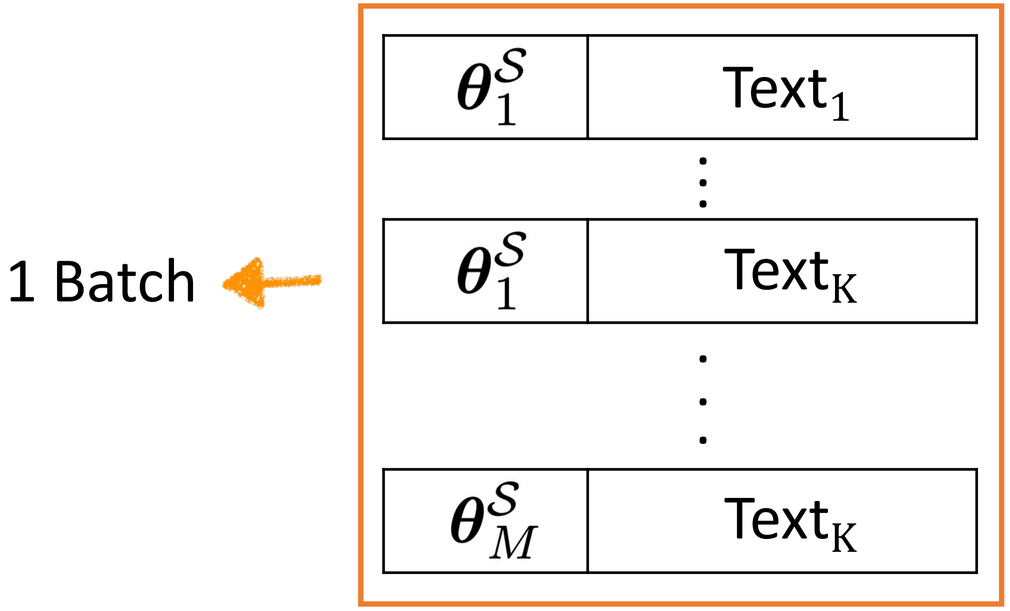


Algorithm 1 Bayesian Prompt Tuning

```
Input:
$\mathcal{D}^{\mathcal{S}}, \mathcal{D}^{\mathcal{T}}$ : source tasks and target task
$\Theta_0 = \{\boldsymbol{\theta}_{0,i}\}_{i=1}^{M}$ : initialized particle set

Source Task Training:
for $t \leftarrow 0$ to $T-1$ do
    $\Theta_{t+1} \leftarrow \Theta_t + \alpha \boldsymbol{\phi}^*_{\mathbb{P}(\cdot|\mathcal{D}^{\mathcal{S}})}(\Theta_t)$
    (SVGD iteration; Section 2.3)
end for
Store $\bar{\boldsymbol{\theta}}^{\mathcal{S}} \leftarrow \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{\theta}_{T,i}$

Task Adaptation:
$J(\boldsymbol{\theta}^{\mathcal{T}}) = -\log \mathbb{P}(\mathcal{D}^{\mathcal{T}}|\boldsymbol{\theta}^{\mathcal{T}}) + \frac{1}{2\sigma^2}\left\|\boldsymbol{\theta}^{\mathcal{T}} - \bar{\boldsymbol{\theta}}^{\mathcal{S}}\right\|^2$
$\boldsymbol{\theta}^{\mathcal{T}*} \leftarrow \mathrm{argmin}_{\boldsymbol{\theta}^{\mathcal{T}}} J(\boldsymbol{\theta}^{\mathcal{T}})$

Output:
$\boldsymbol{\theta}^{\mathcal{T}*}$ : trained weight for the target task
```

Implementation Details: Source Task Posterior

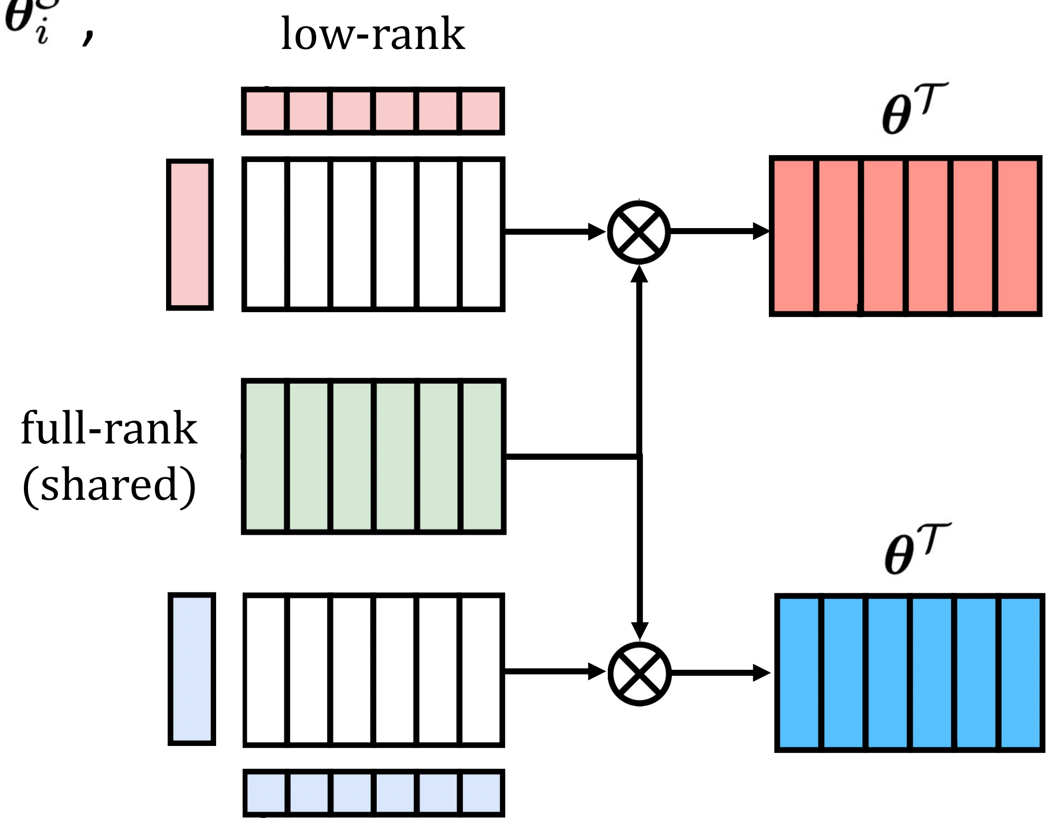
- Batch size is M x K:
  - M is number of particles, K is number of source tasks
  - Each particle is prepended to K input texts, where each of input texts is from different source task
  - The losses for each particle is plugged into SVGD update rule



1 Batch
$\boldsymbol{\theta}_1^{\mathcal{S}}$
Text$_1$
$\boldsymbol{\theta}_1^{\mathcal{S}}$
Text$_{\mathrm{K}}$
$\boldsymbol{\theta}_M^{\mathcal{S}}$
Text$_{\mathrm{K}}$

Implementation Details: Low-rank Matrix Multiplication

- At the start of target adaptation, we construct $\boldsymbol{\theta}^{\mathcal{T}}$ with full-rank and low-rank matrix
  - Full-rank matrix is initialized with $\frac{1}{M}\sum_{i=1}^{M}\boldsymbol{\theta}_i^{\mathcal{S}}$, low-rank matrix is initialized with 1
  - Two-speed learning rate[1] (full-rank: 0.3, low-rank: 0.4)
  - This facilitates multi-target-task adaptation
    - Low-rank matrix for each target task
    - Full-rank matrix is shared during adaptation


low-rank
$\boldsymbol{\theta}^{\mathcal{T}}$
full-rank (shared)
$\boldsymbol{\theta}^{\mathcal{T}}$

[1] Asai et al., "ATTEMPT: Parameter-Efficient Multi-task Tuning via Attentional Mixtures of Soft Prompts", EMNLP 2022

Implementation Details: Source Task Sampling

- If K (number of source tasks) is too big, we sample subset from them
  - As K increases, the batch size increases, therefore it becomes a bottleneck
  - To mitigate this, we sample $\kappa < \mathrm{K}$ source tasks without replacement, for each SVGD iteration

Experiments

- Source tasks:
  - MNLI, QNLI, QQP, SST2, SQuAD, ReCoRD
- Target tasks:
  - 8 GLUE tasks (MNLI, QNLI, QQP, SST2, STSB, MRPC, RTE, CoLA)
  - 5 SuperGLUE tasks (MultiRC, BoolQ, WiC, Wsc.fixed, CB)
- Model details:
  - Base setting: T5-base, 5 SVGD particles, 100 vectors for soft prompt
  - We also experiment with 1) different model size, 2) different number of particles, 3) sampling 6 source tasks out of 12 Source tasks

Experiments: BPT & Baselines

Method	GLUE									SuperGLUE					
	MNLI	QQP	QNLI	SST-2	STS-B	MRPC	RTE	CoLA	Avg.	Multirc	BoolQ	WiC	WSC	CB	Avg.
Fine-tuning	86.8	91.6	93.0	94.6	89.7	90.2	71.9	61.8	84.9	72.8	81.1	70.2	59.6	85.7	73.9
Adapters	86.5	90.2	93.2	93.8	90.7	85.3	71.9	64.0	84.5	75.9	82.5	67.1	67.3	85.7	75.7
BitFit	85.3	90.1	93.0	94.2	90.9	86.8	67.6	58.2	83.3	74.5	79.6	70.0	59.6	78.6	72.5
PT	81.3	89.7	92.8	90.9	89.5	68.1	54.7	10.6	72.2	58.7	61.7	48.9	51.9	67.9	57.8
Vanilla transfer PT	85.8	86.9	93.2	92.9	90.5	87.1	77	83.2	87.1	72.2	77.9	65.5	67.3	78.6	72.3
SPoT	85.4	90.1	93.0	93.4	90.0	79.7	69.8	57.1	82.3	74.0	77.2	67.0	50.0	46.4	62.9
ATTEMPT	84.3	90.3	93.0	93.2	89.7	85.7	73.4	57.4	83.4	74.4	78.8	66.8	53.8	78.6	70.5
MPT	85.9	90.3	93.1	93.8	90.4	89.1	79.4	62.4	85.6	74.8	79.6	69.0	67.3	79.8	74.1
BPT (Ours)	86.2	90.3	93.4	93.7	90.8	87.2	75.5	86.6	**88.0**	72.4	80.3	67.4	67.3	85.7	**74.6**
Fine-tuning*	85.7	91.1	92.0	92.5	88.8	90.2	75.4	54.9	83.8	-	-	-	-	-	-
Adapters*	86.3	90.5	93.2	93.0	89.9	90.2	70.3	61.5	84.4	-	-	-	-	-	-
HyperFormer*	85.7	90.0	93.0	94.0	89.7	87.2	75.4	63.7	84.8	-	-	-	-	-	-
HyperDecoder*	86.0	90.5	93.4	94.0	90.5	87.7	71.7	55.9	83.7	-	-	-	-	-	-
ATTEMPT*	83.8	90.0	93.1	93.7	90.8	86.1	79.9	64.3	85.2	74.4	78.3	66.5	69.2	82.1	74.1
MPT*	84.3	90.0	93.0	93.3	90.4	89.2	82.7	63.5	85.8	74.8	79.2	70.2	67.3	89.3	76.1
BPT* (Ours)	85.9	90.1	92.9	94.6	90.8	89.3	77.1	87.0	**88.5**	72.7	81.0	68.4	67.3	92.9	**76.5**

Table 1: Experiment results on GLUE and SuperGLUE. All results are based on T5-base models. The evaluation metrics are Pearson correlation for STS-B, F1 for MultiRC (Multi), and accuracy for the remaining tasks. (Top, without asterisk) Model adaptation to single target task with no parameter sharing during target task adaptation. (Bottom, with asterisk) Model adaptation to multiple target tasks.

Experiment Result

- Our method (Bayesian Prompt Tuning, BPT) outperforms strong baselines, including current SoTA (MPT), on both single and multi target adaptation
- Surprisingly, Vanilla transfer learning on Soft Prompt Tuning outperforms strong baselines, supporting the claim that previous works are not optimal transfer learning techniques
    - Ours outperforms Vanilla transfer learning as we learn the posterior, enabling richer adaptation process

Experiments: # Particles, Task Sampling, Model Scaling

BPT	GLUE									SuperGLUE					
	MNLI	QQP	QNLI	SST-2	STS-B	MRPC	RTE	CoLA	Avg.	Multirc	BoolQ	WiC	WSC	CB	Avg.
w/ 5 particles	86.2	90.3	93.4	93.7	90.8	87.2	75.5	86.6	88.0	72.4	80.3	67.4	67.3	85.7	74.6
w/ 10 particles	85.7	90.4	93.3	93.8	90.8	90.8	77.0	85.2	88.4	72.7	78.9	68.6	67.3	83.1	74.1
w/ task sampling	85.8	90.3	93.1	93.8	90.9	87.3	78.4	86.9	88.3	72.1	80.8	69.6	67.3	84.1	74.8

BPT	GLUE									SuperGLUE					
	MNLI	QQP	QNLI	SST-2	STS-B	MRPC	RTE	CoLA	Avg.	Multirc	BoolQ	WiC	WSC	CB	Avg.
T5-base	86.2	90.3	93.4	93.7	90.8	87.2	75.5	86.6	88.0	72.4	80.3	67.4	67.3	85.7	74.6
T5-large	89.1	90.9	94.1	95.5	92.3	89.3	76.2	86.6	89.3	95.7	84.4	72.4	67.3	83.2	76.6

Experiment Result

- 10 particles are not better than 5 particles
  - As instability reported at original SVGD paper[1], and similar phenomenon reported at other SVGD application paper[2], we presume that this might be attributed to inherent characteristics of SVGD, including sensitivity to kernel function parameters
- Source task sampling benefits
  - We added 6 additional source tasks (AGNews, CommonsenseQA, OpenBookQA, ARC, adversarial NLI, Winogrande)
  - Diversifying source tasks benefit overall performance

[1] Liu et al., "Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm", NeurIPS 2016
[2] Yoon et al., "Bayesian Model-Agnostic Meta-Learning", NeurIPS 2018

Experiment Result

- Model scaling
  - Experiment result on T5-large outperforms T5-base with noticeable margin
  - BPT can benefit from model scaling

Contribution

- We present BPT, a Bayesian approach on transfer learning for Soft Prompt Tuning
  - Instead of training source prompt individually on each source task, we make source prompts to approximate the global posterior, which enables more richer and robust transfer learning framework
  - Rather than resorting to transferability between NLP tasks, Framing Soft Prompt Tuning as Bayesian transfer learning learning helps
  - Without any additional neural network (e.g. another transformer for distillation[1], attention module[2]) BPT shows robust performance

[1] Wang et al., ”Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning", ICLR 2023
[2] Asai et al., "ATTEMPT: Parameter-Efficient Multi-task Tuning via Attentional Mixtures of Soft Prompts", EMNLP 2022

Limitation

- Since we employ particles, the batch size should be multiple of number of particles
  - Batch size may get large, and requires more computational resources

Thank you!