Memory Efficient Continual Learning for Neural Text Classification

Beyza Ermis, Giovanni Zappella, Martin Wistuba, Cedric Archambeau
Amazon Web Services
NeurIPS 2022

Antnlp

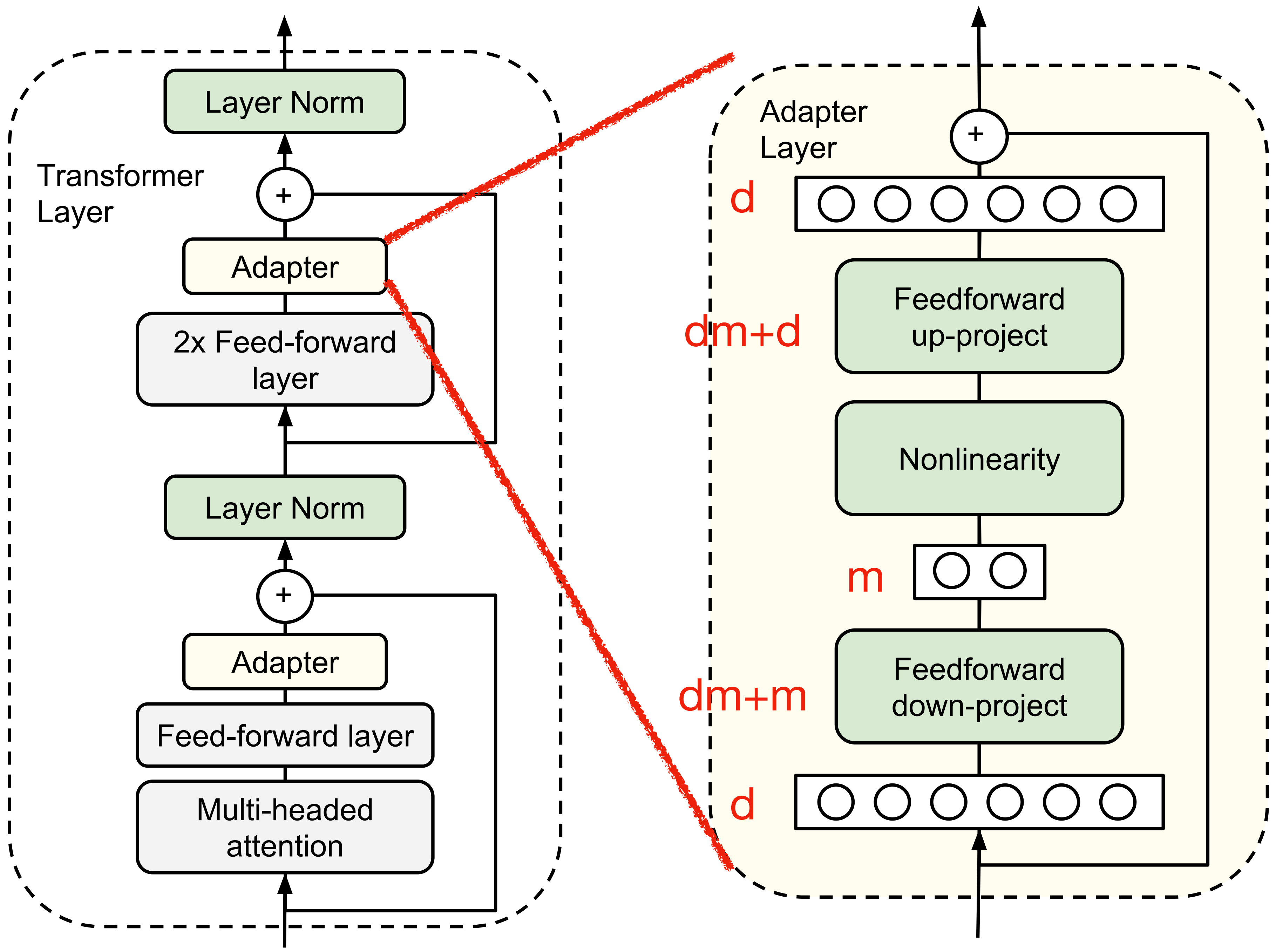
Contents

Adapter

Tunable Params. in each Encoder block:

$$2md + m + d, m \ll d$$

$d = 768$ for Bert
$m \in 2,4,\cdots,64,256$



	Total num params	Trained params / task	CoLA	SST	MRPC	STS-B	QQP	$MNLI_m$	$MNLI_{mm}$	QNLI	RTE	Total
$BERT_{LARGE}$	9.0×	100%	60.5	94.9	89.3	87.6	72.1	86.7	85.9	91.1	70.1	80.4
Adapters (8-256)	1.3×	3.6%	59.5	94.0	89.5	86.9	71.8	84.9	85.1	90.7	71.5	80.0
Adapters (64)	1.2×	2.1%	56.9	94.2	89.6	87.3	71.8	85.3	84.6	91.4	68.8	79.6

Table 1. Results on GLUE test sets scored using the GLUE evaluation server. MRPC and QQP are evaluated using F1 score. STS-B is evaluated using Spearman's correlation coefficient. CoLA is evaluated using Matthew's Correlation. The other tasks are evaluated using accuracy. Adapter tuning achieves comparable overall score (80.0) to full fine-tuning (80.4) using 1.3× parameters in total, compared to 9×. Fixing the adapter size to 64 leads to a slightly decreased overall score of 79.6 and slightly smaller model.

Adapter Fusion

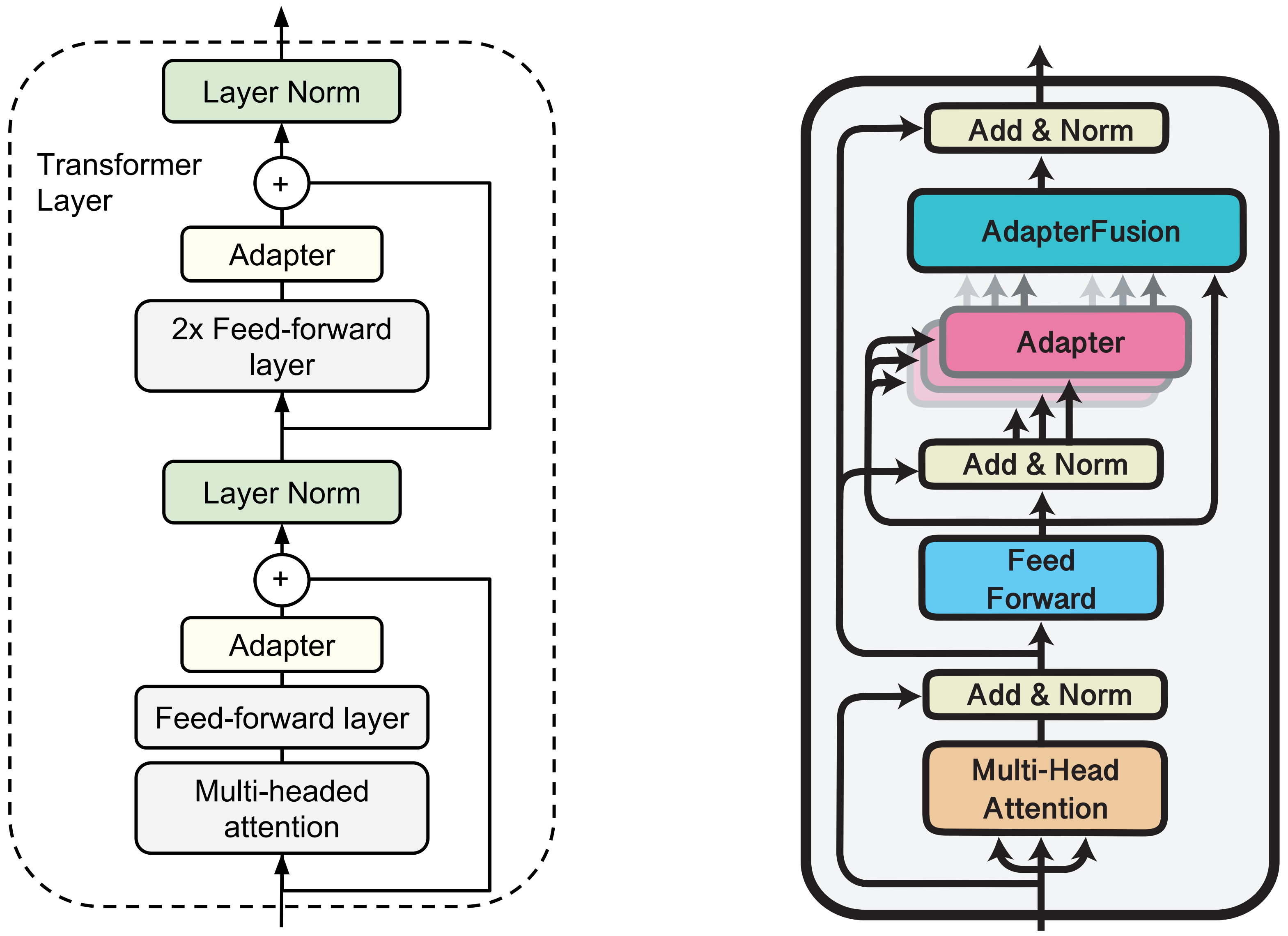


Figure 1: AdapterFusion architecture inside a transformer (Vaswani et al., 2017). The AdapterFusion component takes as input the representations of multiple adapters trained on different tasks and learns a parameterized mixer of the encoded information.

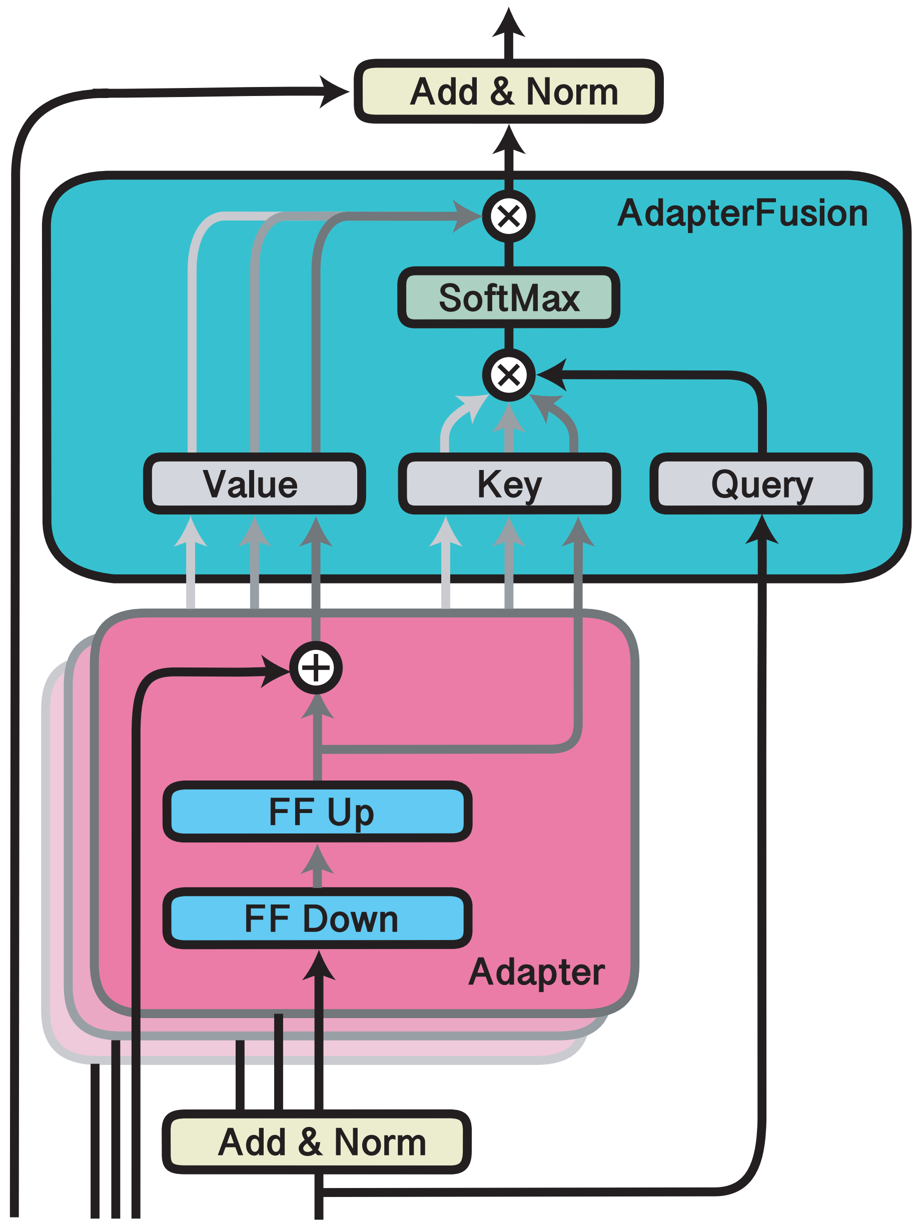


Figure 2: Our AdapterFusion architecture. This includes learnable weights *Query*, *Key*, and *Value*. *Query* takes as input the output of the pretrained transformer weights. Both *Key* and *Value* take as input the output of the respective adapters. The dot product of the *query* with all the *keys* is passed into a softmax function, which learns to weight the adapters with respect to the context.

Adapter Fusion

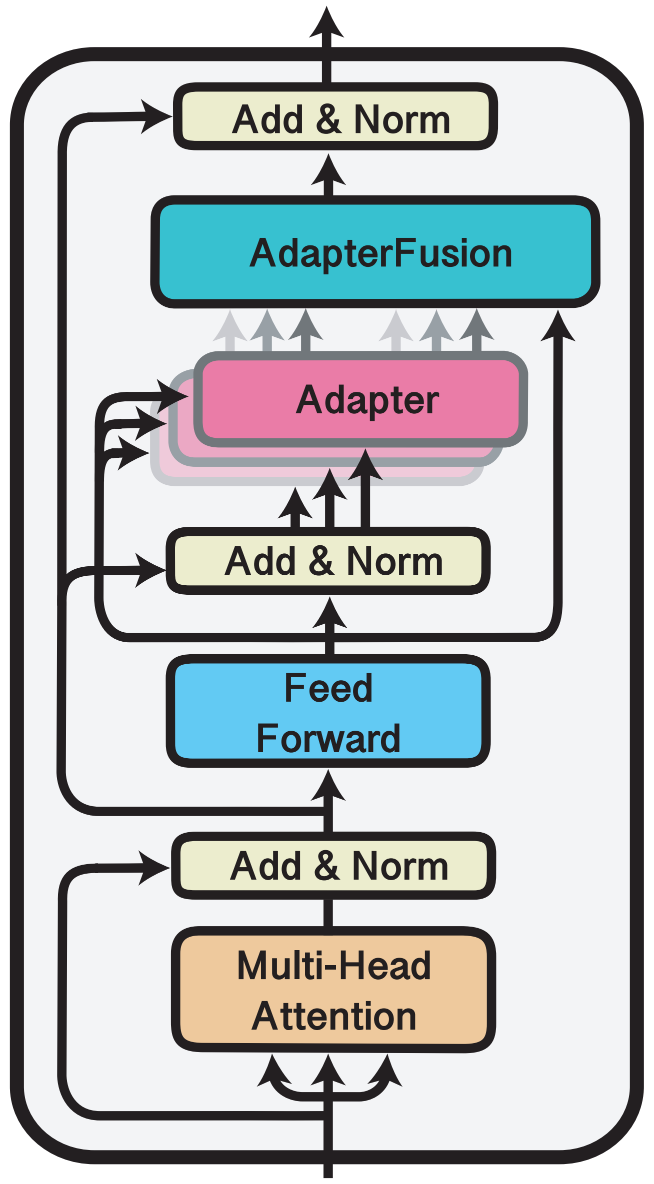


Figure 1: AdapterFusion architecture inside a transformer (Vaswani et al., 2017). The AdapterFusion component takes as input the representations of multiple adapters trained on different tasks and learns a parameterized mixer of the encoded information.

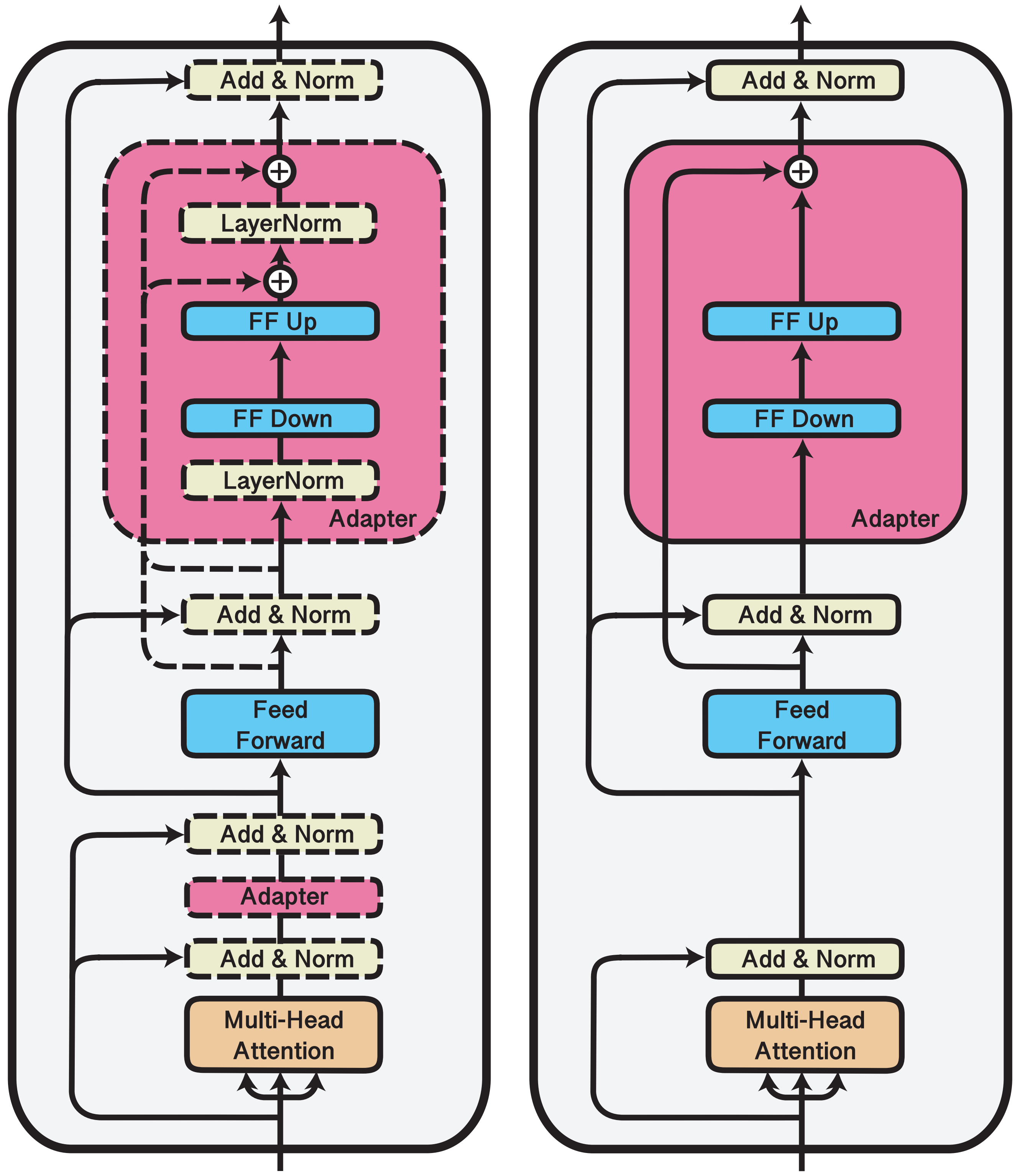


Figure 5: Different architectural components of the adapter. On the left, we show all components for which we conduct an exhaustive search (dashed lines). On the right, we show the adapter architecture that performs the best across all our tasks.

Adapter Fusion

Step1: knowledge extraction

Single task adapter:

$$\Phi_i \leftarrow \arg\min_{\Phi} L_i(D_i; \Theta, \Phi).$$

Multi-task adapter:

$$\mathbf{\Theta} \leftarrow \underset{\Theta,\Phi}{\operatorname{argmin}} \left(\sum_{n=1}^{N} L_n(D_n; \Theta_0, \Phi_n) \right)$$

where

$$\mathbf{\Theta} = \Theta_{0 \rightarrow \{1,\ldots,N\}}, \Phi_1, \ldots, \Phi_N.$$

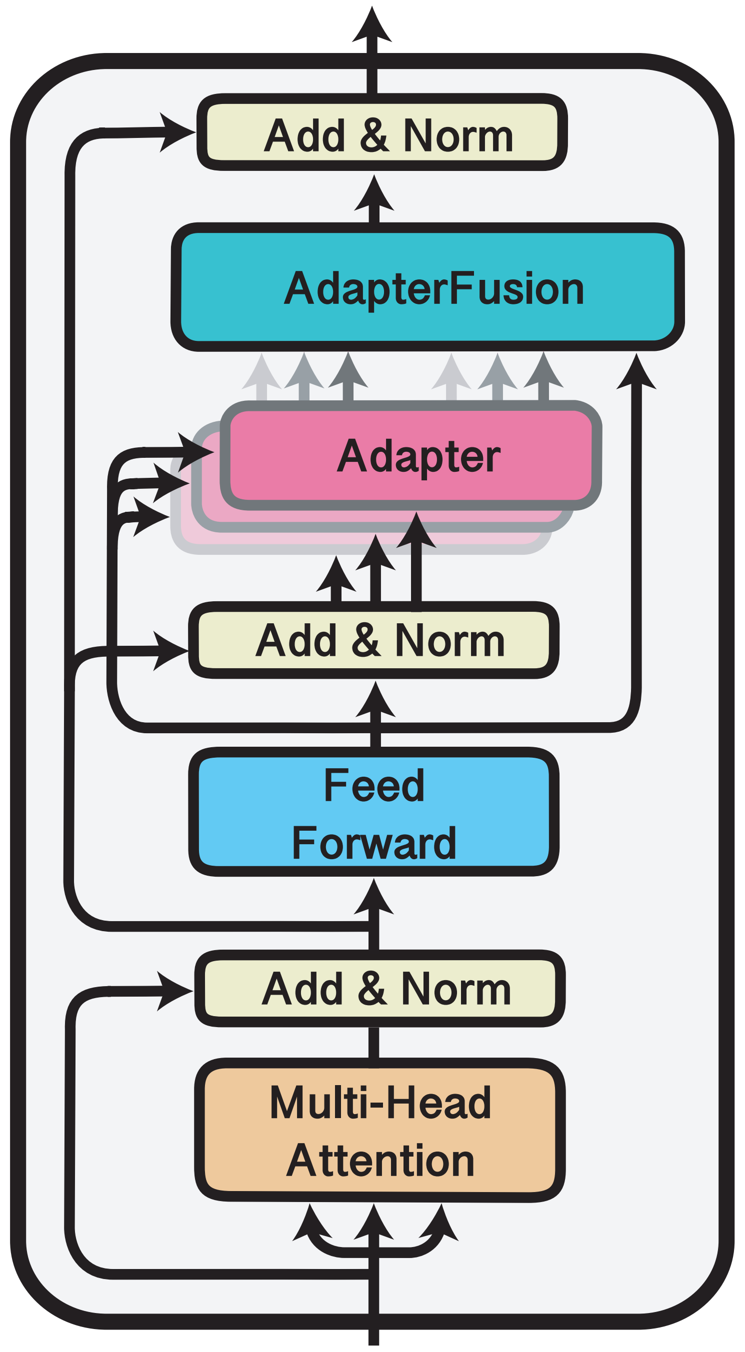


Figure 1: AdapterFusion architecture inside a transformer (Vaswani et al., 2017). The AdapterFusion component takes as input the representations of multiple adapters trained on different tasks and learns a parameterized mixer of the encoded information.

Adapter Fusion

Step2: knowledge compostion

Adapter fusion:

$$\Psi_m \leftarrow \underset{\Psi}{\operatorname{argmin}}\, L_m(D_m; \Theta, \Phi_1, \ldots, \Phi_N, \Psi)$$

Goals: 增强任务之间的知识共享

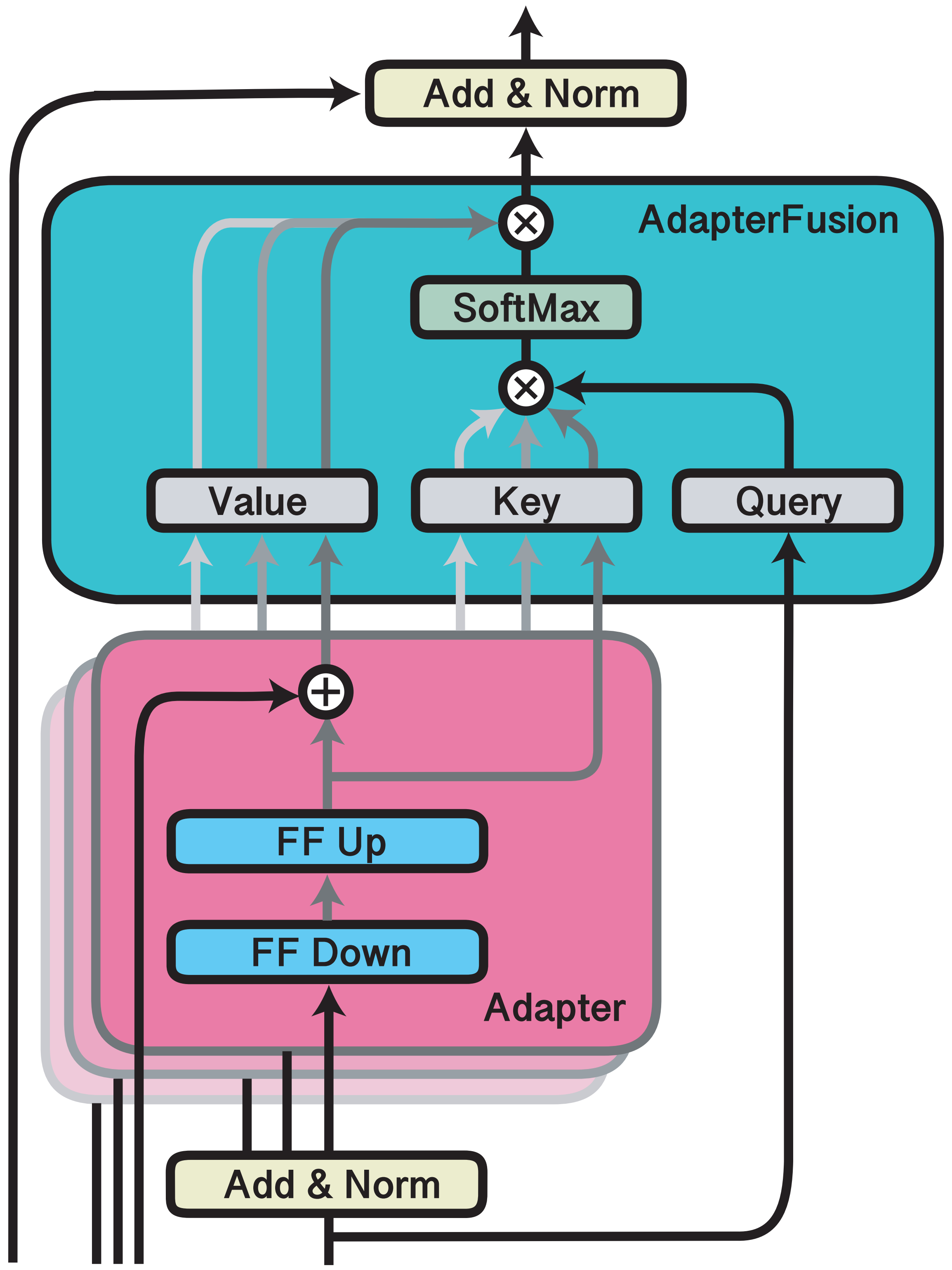
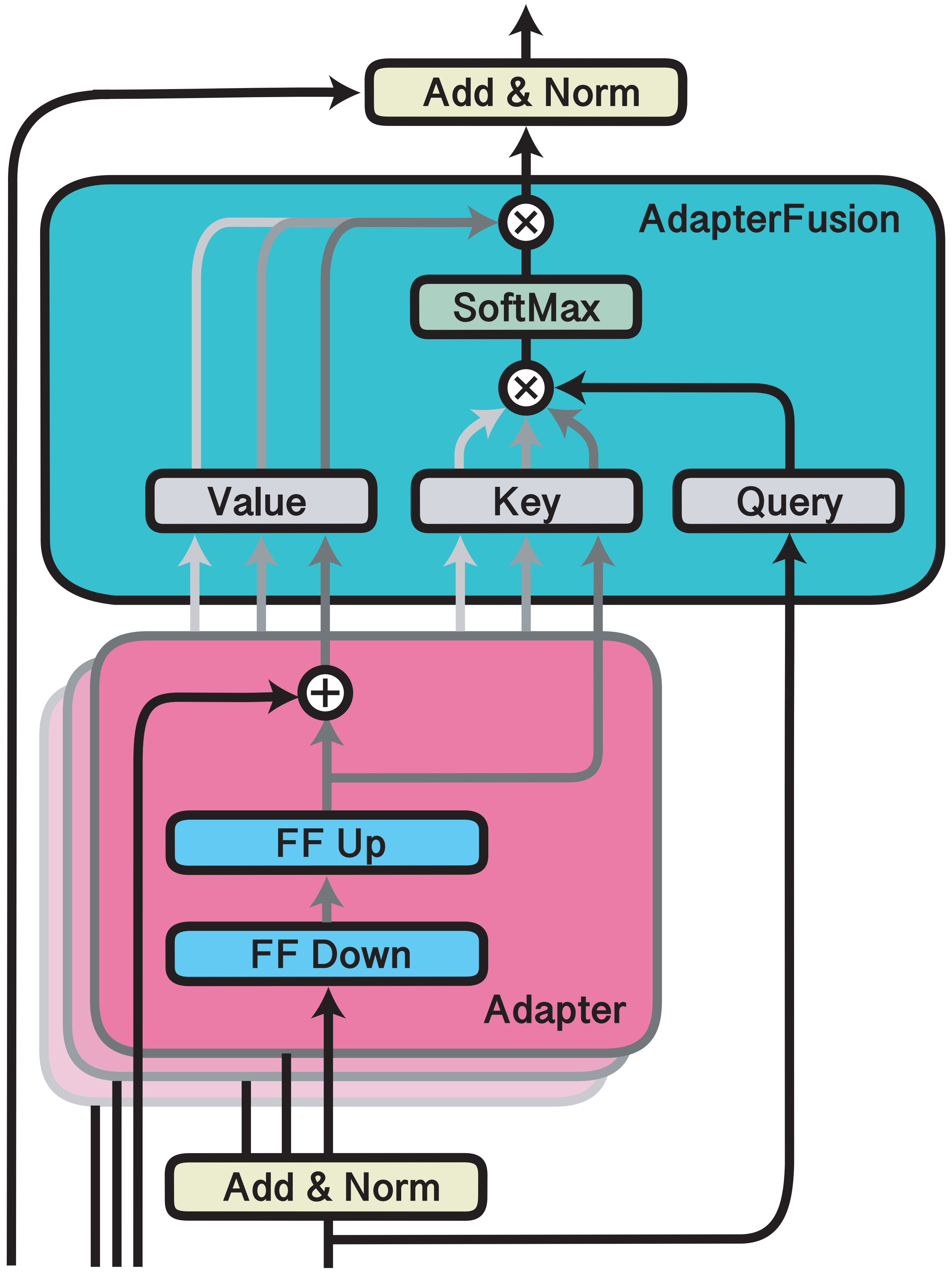


Figure 2: Our AdapterFusion architecture. This includes learnable weights *Query*, *Key*, and *Value*. *Query* takes as input the output of the pretrained transformer weights. Both *Key* and *Value* take as input the output of the respective adapters. The dot product of the query with all the *keys* is passed into a softmax function, which learns to weight the adapters with respect to the context.

AdapterFusion: Non-Destructive Task Composition for Transfer Learning

Adapter Fusion



$$\mathbf{s}_{l,t} = \text{softmax}(\mathbf{h}_{l,t}^{\top}\mathbf{Q}_l \otimes \mathbf{z}_{l,t,n}^{\top}\mathbf{K}_l), n \in \{1, ..., N\}$$

$$\mathbf{z}'_{l,t,n} = \mathbf{z}_{l,t,n}^{\top}\mathbf{V}_l, n \in \{1, ..., N\}$$

$$\mathbf{Z}'_{l,t} = [\mathbf{z}'_{l,t,0}, \dots, \mathbf{z}'_{l,t,N}]$$

$$\mathbf{o}_{l,t} = \mathbf{s}_{l,t}^{\top}\mathbf{Z}'_{l,t}$$

l : Encoder layer

t : time step (token id)

n : adapter id

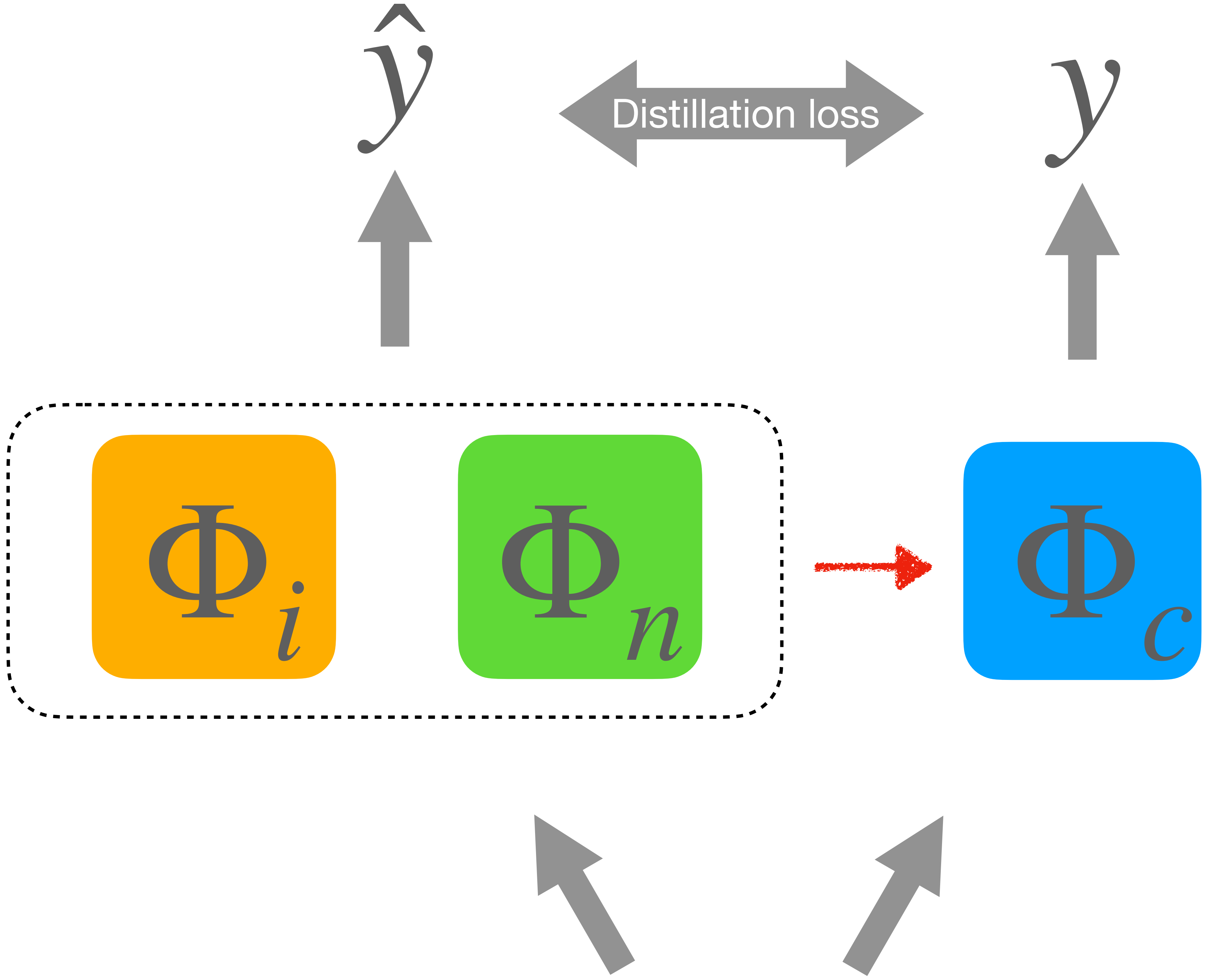
AdapterFusion: Non-Destructive Task Composition for Transfer Learning

Setting

Task Incremental Learning

- 多个任务按序到达
- 新任务到达时，之前任务的数据不可用（unaccessable)
- 测试时，提供输入所属的任务ID
- 目标：
  1. 缓解灾难性遗忘
  2. 前向/反向知识迁移
  3. 降低所需的计算和存储资源

Distillation of Adapters



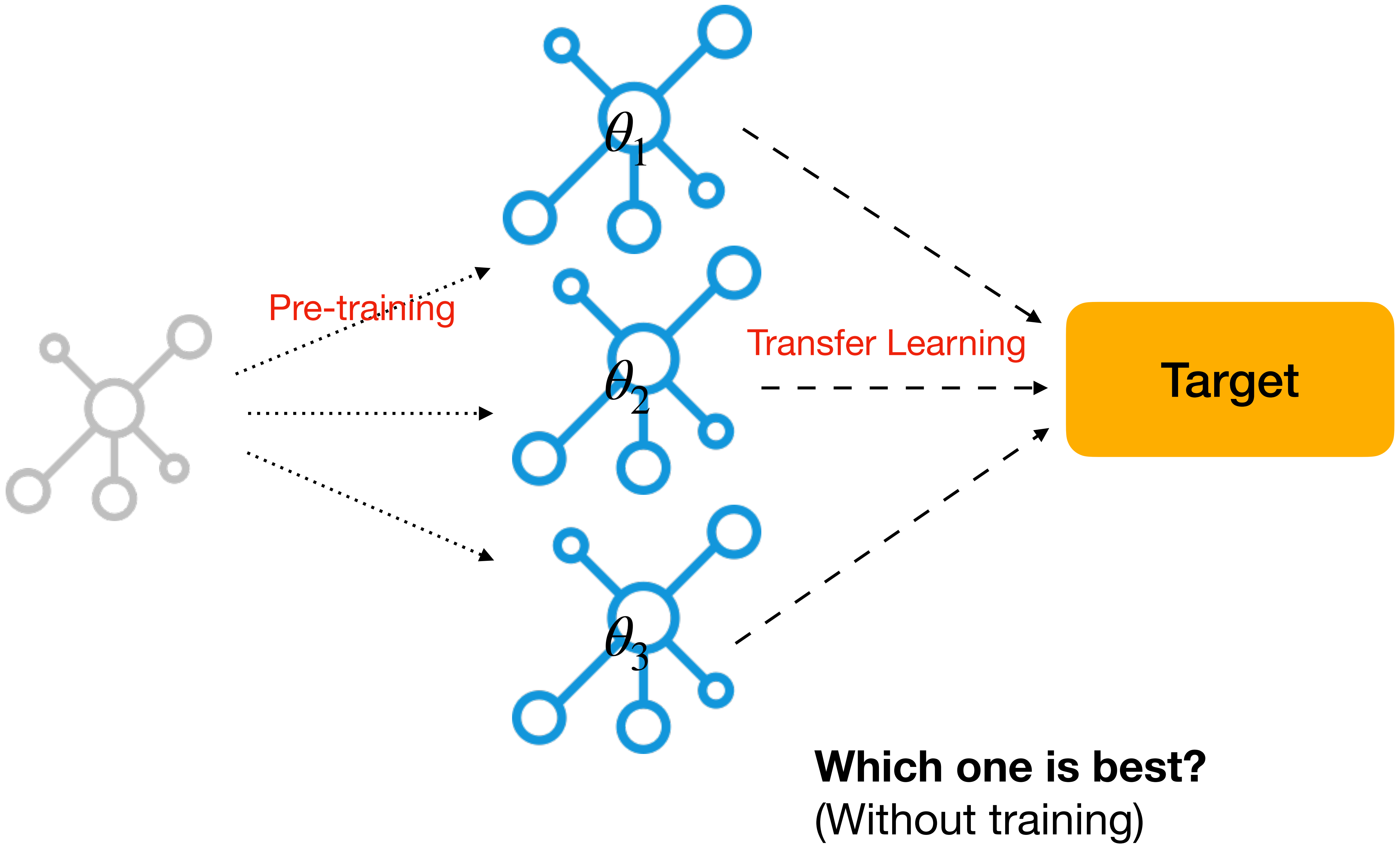
$$f(x;\Theta,\Phi_c) = \begin{cases} f_{old}(x;\Theta,\Phi_i,h_i)[i], & 1 \le i \le n-1 \\ f_{new}(x;\Theta,\Phi_n,h_n)[i], & i = n \end{cases}$$

Unlabeled data $X = \{(x_1), \cdots, (x_n)\}$

Distillation loss $$L_d(y,\hat{y}) = \frac{1}{n}\sum_{j=1}^{t}\left(y^j - \hat{y}^j\right)^2,$$

Distillation objectvie $$\min_{\Theta,\Phi_c} \frac{1}{|\mathcal{U}|}\sum_{x_j \in \mathcal{U}} L_d(y,\hat{y}),$$

Transferability Estimation



方法1：迁移训练之后比较评价指标。评价指标越好，则说明可迁移性越强。

方法2：使用LEEP，TransRate估计可迁移性。

LEEP - Log Expected Empirical Prediction

Transferability Estimation

Src: (X, Z), Tgt: (X, Y), the model θ pre-trained on src

Step1: Computer dummy label distribution
of the inputs in the target data set D

$\theta(x_i)$ over Z

LEEP - Log Expected Empirical Prediction

Transferability Estimation

Step2: Compute the empirical conditional distribution $\hat{P}(y \mid z)$ of the target label y given the source label z

Fisrt compute the empirical joint distribution

$$\hat{P}(y, z) = \frac{1}{n} \sum_{i: y_i = y} \theta(x_i)_z$$

i表示sample id，n表示Y标签为y的样本个数。

Then, computre the empirical marginal distribution

$$\hat{P}(z) = \sum_{y \in Y} \hat{P}(y, z) = \frac{1}{n} \sum_{i=1}^{n} \theta(x_i)_z$$

Finally compute the empirical conditional distribution

$$\hat{P}(y \mid z) = \frac{\hat{P}(y, z)}{\hat{P}(z)}$$

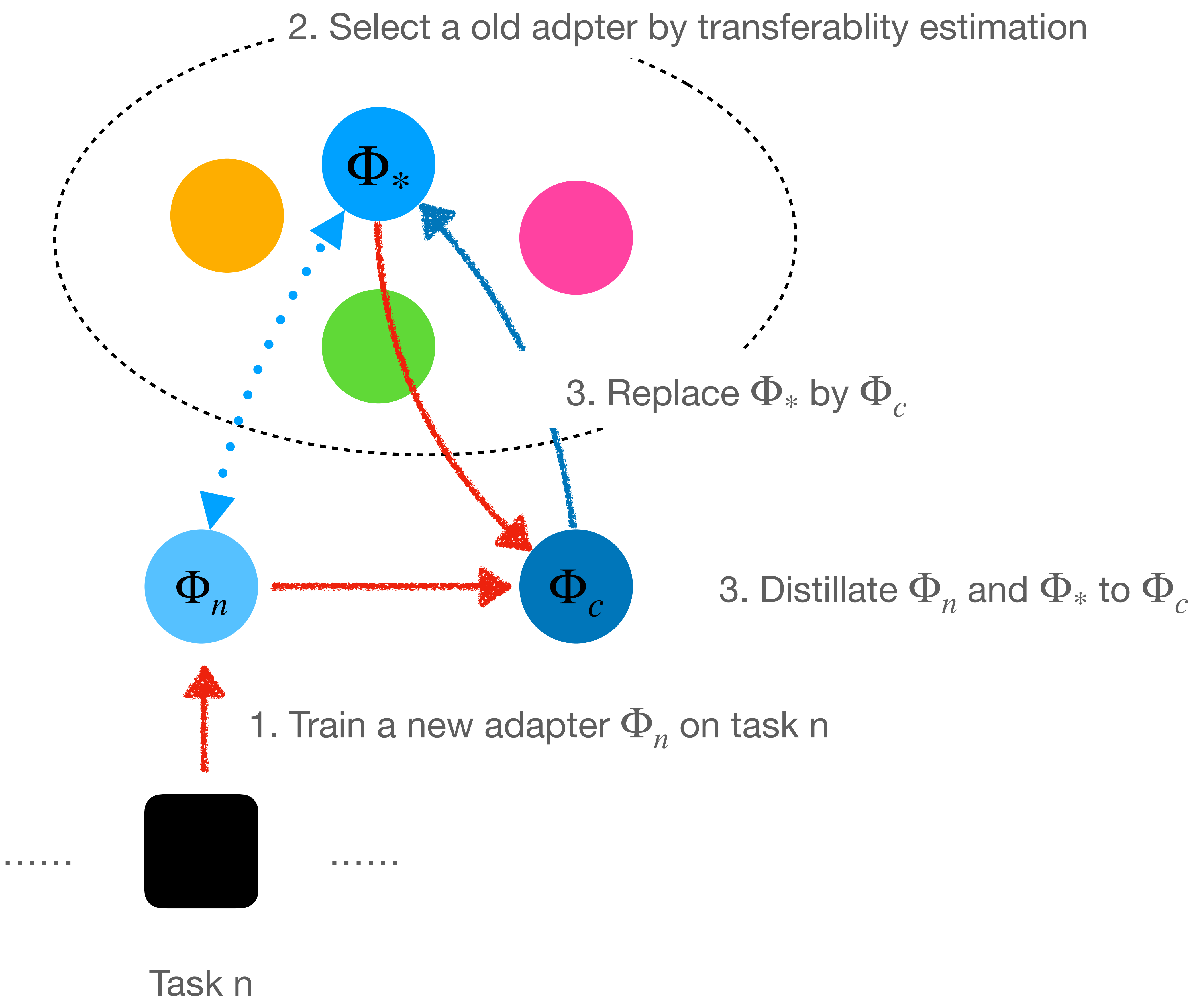
LEEP - Log Expected Empirical Prediction

Transferability Estimation

Step1: Compute LEEP using $\theta(x)$ and $\hat{P}(y \mid z)$

$$T(\theta, D) = \frac{1}{n}\sum_{i=1}^{n} \log p(y_i \mid x; \theta, D) = \frac{1}{n}\sum_{i=1}^{n} \log\Big(\sum_{z \in Z} \hat{p}(y_i \mid z)\theta(x_i)_z\Big)$$

Algorithm



Algorithm 1 Adaptive Distillation of Adapters (ADA)

Require: Θ: pre-trained model, K: adapters pool size

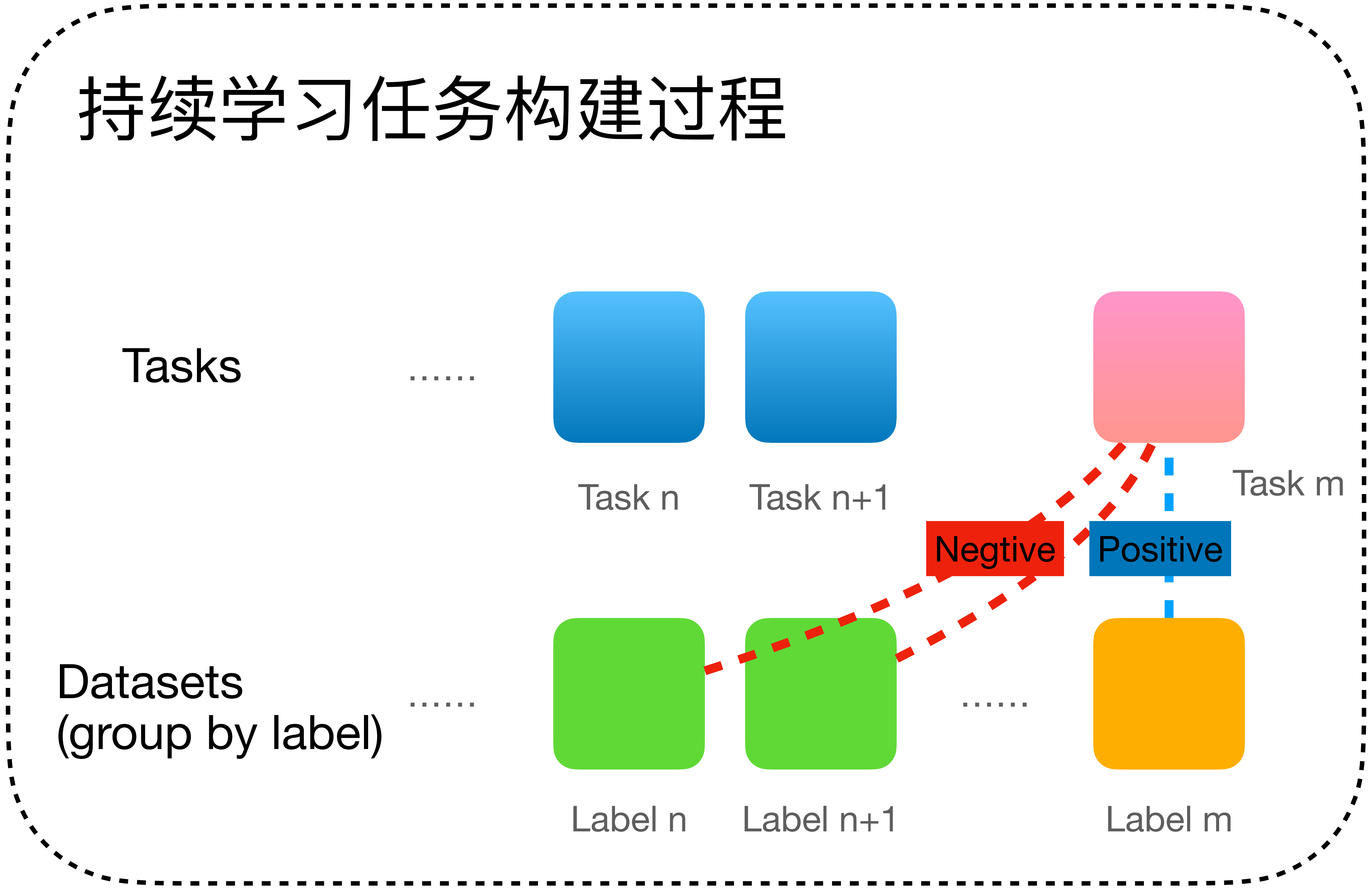
Freeze Θ
Create $m = Map()$
for $n \leftarrow 1$ to N **do**
    A task T_n is received
    Initialize Φ_n
    Process T_n and train new model $f_n(x; \Theta, \Phi_n)$ and head h_n
    Sample from T_n and add to distillation data $\mathcal{D}_{distill}$
    if $n \leq K$ **then**
        Set $f_n(x; \Theta, \Phi_1)$ to f^n_{old}
    else
        $j^* \leftarrow \arg\max_{j \in \{1,\ldots,K\}}$ *TRANSCORE*(T_n, f_n, f^j_{old})
        Add (n, j^*) to m
        Consolidate model:
            $f^{j^*}_{old}$ = *DISTILLATION*$(f^{j^*}_{old}, f_n, \mathcal{D}_{distill})$
    end if
    Serve predictions for any task $i \leq n$ using h_i and $f^{m(i)}_{old}$
end for

DISTILLATION$(f_{old}, f_n, \mathcal{D}_{distill})$:
    Get soft targets $\hat{y}_{old}$ from old model f_{old} with $\mathcal{D}_{distill}$
    Get soft targets $\hat{y}_{new}$ from new model f_n with $\mathcal{D}_{distill}$
    Initialize Φ_c
    Compute distillation loss as in Equation (2) and train model $f(x; \Theta, \Phi_c)$
    return f

Experiments

Datasets & Experimental Setup

Datasets	# samples	# labels	# tasks	# training / testing
Arxiv Papers	55,840	54	20	100/40
Reuters (RCV1-V2)	800,000	103	20	100/40
Wiki-30K	20,764	29,947	60	100/40



使用水塘抽样保持一个大小为m的
buffer，用来做adapter蒸馏。
m = 500/500/1000

Results

1) Fine-tuning head model (B1)
2) Fine-tuning the full model (B2)
3) Adapters: train and keep separate adapters for each task
4) AdapterFusion
5) Experience Replay (ER): frozen LM and only tuning a addtional adapter. Store ALL training data

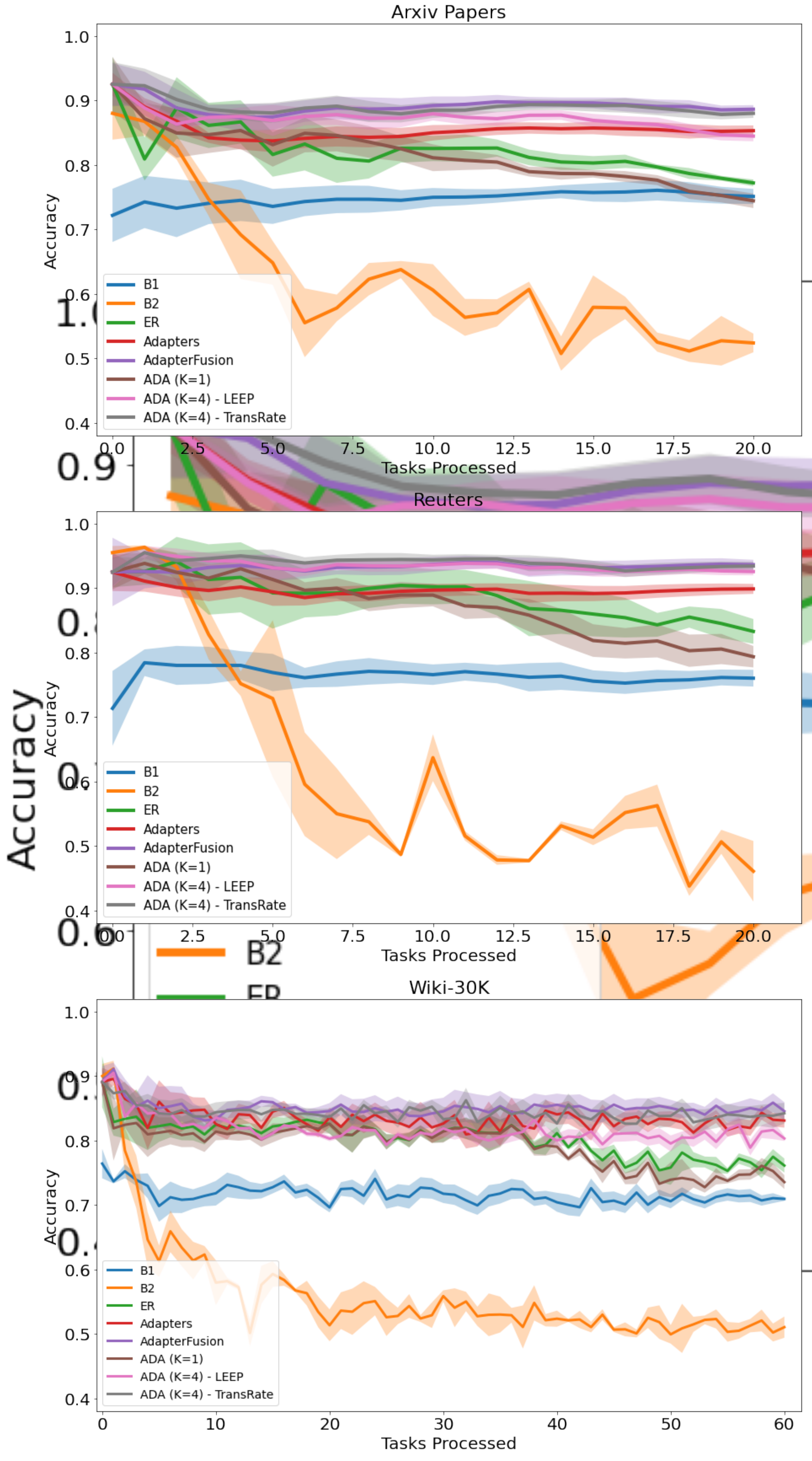


Figure 1. Comparison between baselines and ADA on arXiv, Reuters and Wikipedia. On the x-axis we report the number of tasks processed, on the y-axis we report the average accuracy measured on the test set of the tasks processed, shaded area shows standard deviation.

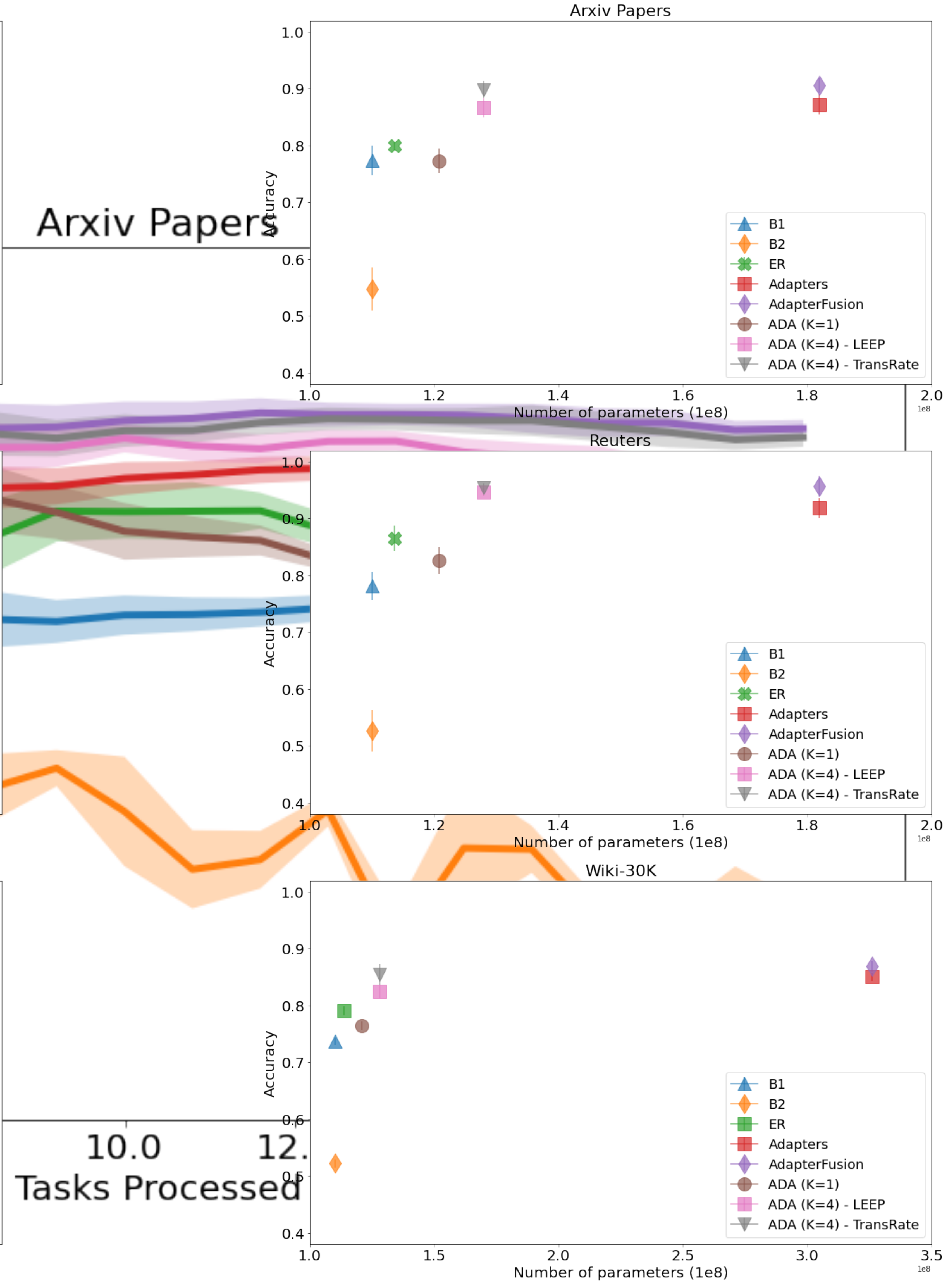


Figure 2. Comparison of number of parameters of baselines and ADA on Arxiv, Reuters and Wikipedia. The predictive performance reported on the y-axis is measured after processing all tasks.

Ablation studies

Size x 1: m = 32
Size x 4: m = 128

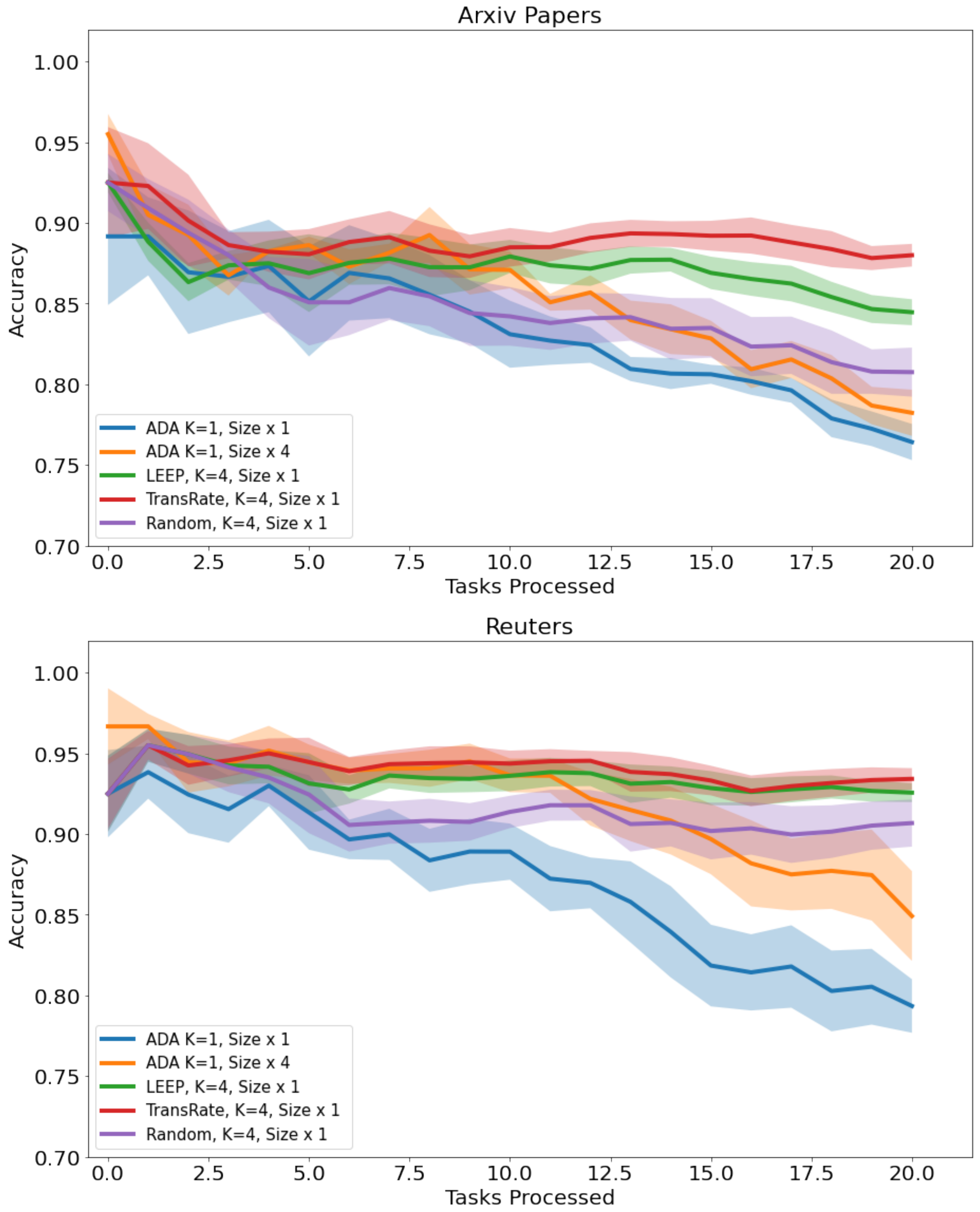


Figure 4. Impact of *LEEP* and *TransRate* when the total number of adapter parameters is same on arXiv and Reuters.

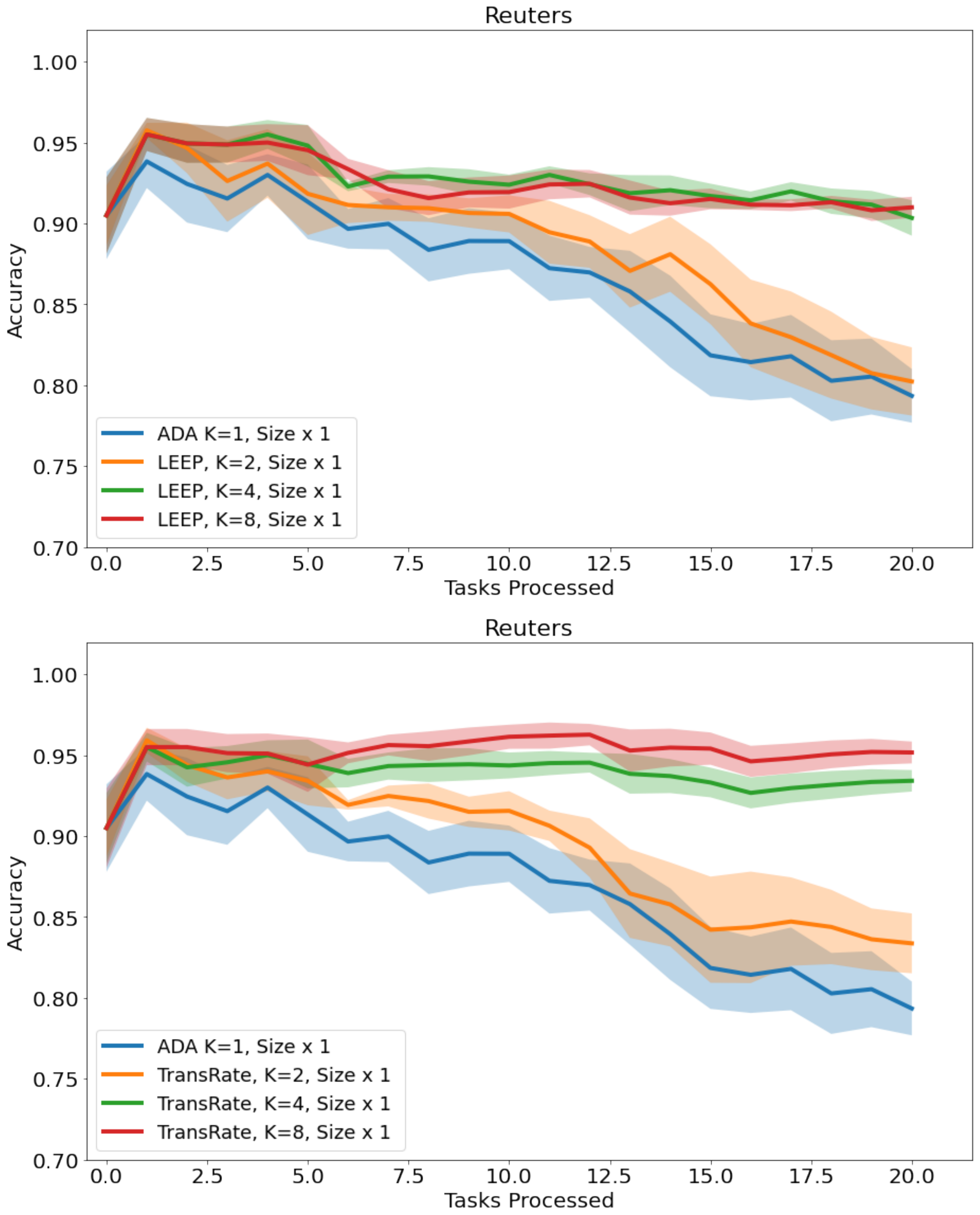


Figure 5. *LEEP* and *TransRate* performances when $K = \{1, 2, 4, 8\}$ on Reuters.