겨울 세미나

어수경

2024. 02. 29

Natural Language Processing
& Artificial Intelligence

논문 리스트

1. **Mixture-of-Experts Meets Instruction Tuning: A Winning Combination for Large Language Models**
   - Google
   - ICLR 2024 (Poster)

2. **Pushing Mixture of Experts to the Limit: Extremely Parameter Efficient MoE for Instruction Tuning**
   - Cohere for AI
   - ICLR 2024 (Poster)

Preamble: MoE

- **Motivation**

- LLM era로 접어들면서 모델 크기가 중요해진 시점
- 똑같은 컴퓨팅 budget에 대해 더 큰 모델을 더 짧은 스텝에서 학습시킬 수 있다면 좋을 것

- **Brief history**

- 1991년에 처음으로 Adaptive Mixture of Local Experts라는 논문에서 다룸
- 2010-2015년에 두 개의 연구가 추가적인 개선에 영향을 주었음
- Experts as components: MoEs as components of deeper networks (이전에는 MoE가 전체 시스템)
- Conditional computation: dynamic하게 input token에 따라서 components를 활성화/비활성화 할지를 결정
- Shazeer's exploration of MoEs: computation을 늘리지 않고도 스케일을 확장시키기 위한 아이디어로 conditional computation이라는 아이디어가 시작되었음

Preamble: MoE

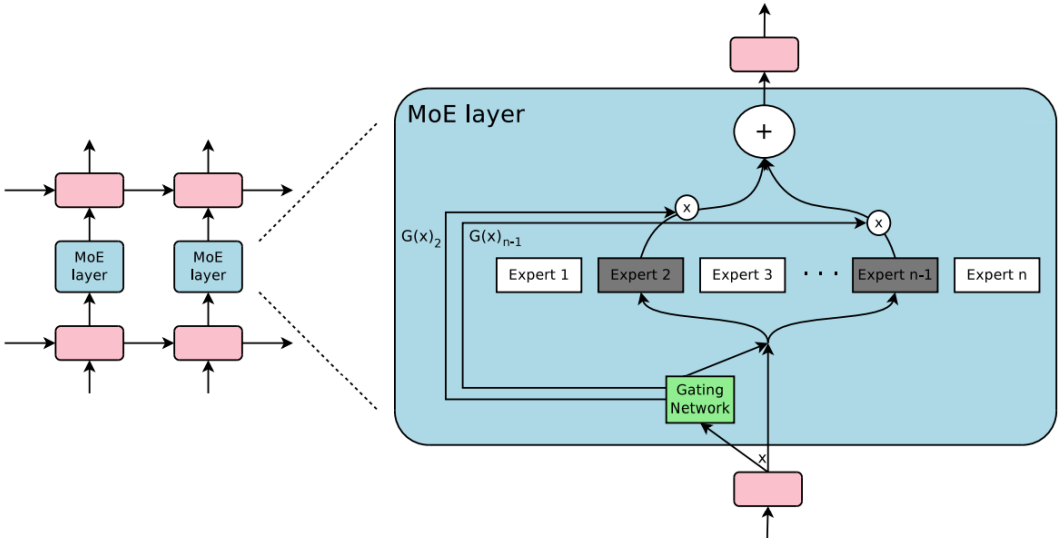
- **Mixture-of-Experts (MoE) – Sparse model (org model: dense model)**

(1) Experts

- FFN layers 대체 –sparse MoE layers
- Expert n개 생성

(independent feed-forward network)

- 주어진 입력에 대해 적합한 Top-k expert를 sparsely activate



(2) Gating Network

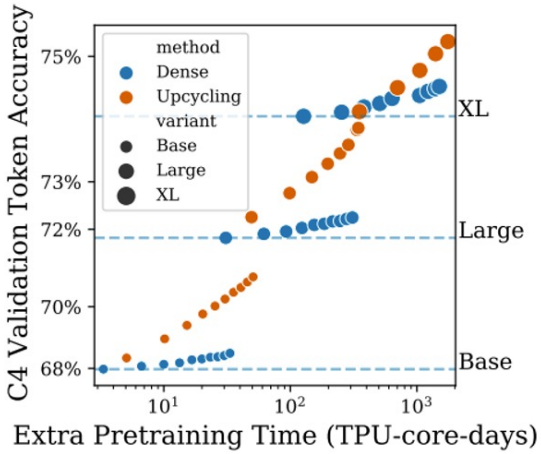
- 주어진 입력을 처리하는 데 가장 적합한 expert을 선택 및 입력 토큰 expert에 전달
- Softmax activation function 활용 → probability distribution

  → 각 expert가 주어진 input에 대해 얼마나 잘 처리할 수 있는지를 확률로 나타냄
- 선택된 experts로부터 얻은 output에 대해 weight combination

  → final learned representation of a token

Preamble: MoE

- **Pros**
  - Pretraining 시 빠른 학습 (significantly more compute-efficient pretraining)
  - dense 모델보다 우수한 성능
  - 비슷한 파라미터수를 지니는 모델이랑 비교해서 빠른 inference 속도



- **Cons**
  - 높은 VRAM을 요구 (모든 experts가 메모리에 로드되기 때문)
  - Fine-tuning 시에는 큰 효과를 보지 못함 (오히려 dense 모델보다 더 낮은 성능 보고)
  - 상당히 많은 tuning 전략: routing strategy, auxiliary loss(load balancing), expert dropout, expert number 등

Mixture-of-Experts Meets Instruction Tuning: A Winning Combination for Large Language Models

Sheng Shen[♮*] **Le Hou**[†] **Yanqi Zhou**[†] **Nan Du**[†] **Shayne Longpre**[⊤*] **Jason Wei**[†],
Hyung Won Chung[†] **Barret Zoph**[†] **William Fedus**[†] **Xinyun Chen**[†] **Tu Vu**[‡*],
Yuexin Wu[†] **Wuyang Chen**[§*] **Albert Webson**[†] **Yunxuan Li**[†] **Vincent Zhao**[†] **Hongkun Yu**[†]
Kurt Keutzer[♮] **Trevor Darrell**[♮] **Denny Zhou**[†]

[†]Google [♮]University of California, Berkeley [⊤]Massachusetts Institute of Technology
[‡]University of Massachusetts Amherst [§]The University of Texas at Austin

I. Introduction

- LLM의 사이즈, computational requirements 문제들
- Scalable technique 개발에 대한 수요

→ MoE의 등장 (sparsely activated manner를 통한 동일한 파라미터 대비 훨씬 빠른 추론 속도)

- Pretraining에서는 우수성이 증명
- 그러나 일반 fine-tuning 수행 시에는 같은 computational cost에 대해서도 dense model보다 성능 하락의 경향성

  (원인 1) hyperparameter에 대한 민감도 – 추가 tuning 필요성

  (원인 2) activated experts만 처리 – 특정 downstream task에 대해 쉽게 overfitting 발생 – generalization 능력 저하

I. Introduction

- **In this paper,**

  - MoE meets instruction tuning
    - MoE 모델에 대해 instruction tuning 시 dense model과 비교하여 성능 향상
    - zero-shot, few-shot setting 모두에서 성능 향상
    - 개별 downstream 및 multi-task에 대해서도 성능 향상

  - Unique amalgamation: FLAN-MoE
  - Instruction tuning 시 MoE를 활용하는 방법에 대한 extensive analysis 수행

II. Method

- **Model architecture**
  - Feed-forward component → MoE layer로 대체
    → expert 생성; 파라미터 수는 FFN x # expert로 늘어나지만 sparsely activate – limiting computation
  - Gating function → softmax function 활용 (probability distribution 생성)
  - Routing 전략 → 토큰마다 Top-2개의 expert를 dynamically 선택
  - 최종 선택된 representation들은 weighted combination

- **Instruction fine-tuning recipe**
  - Instruction tuning dataset: FLAN collective dataset
  - 모든 파라미터는 update

III. Experiment

- **Controlled study across scales**

  - Flan-MoE vs Flan-T5

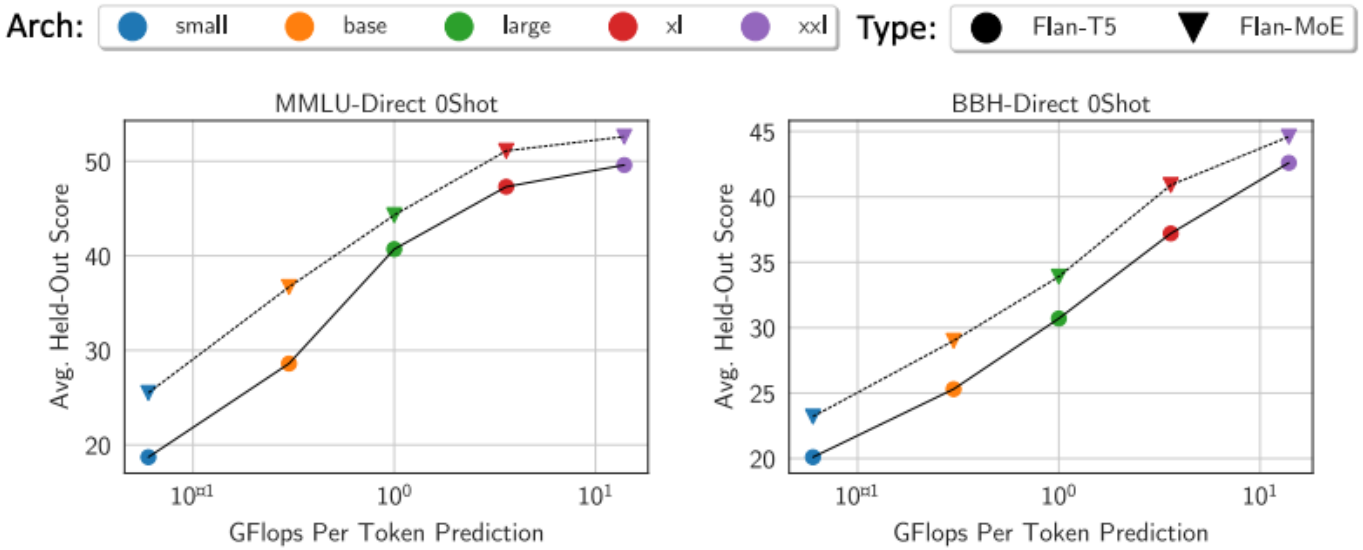


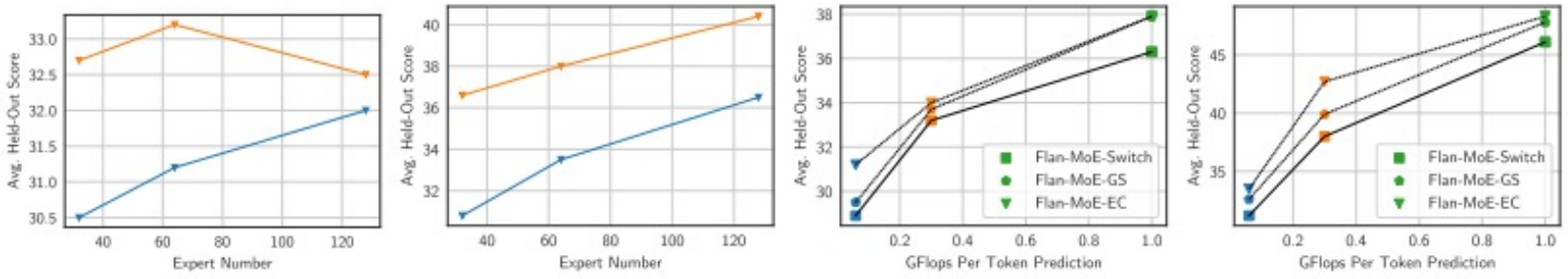
Figure 2: Average zero performance of FLAN-MOE models versus FLAN-T5 dense models for similar effective FLOPs per token over the 57 MMLU tasks and 23 BBH tasks.

- Scale이 커질수록 성능 향상 경향
- FLOPs 역시 증가

III. Experiment

- **Controlled study across scales**

  - Expert 수에 따른 성능 변화



(a) Scaling (MMLU) (b) Scaling (BBH) (c) Routing (MMLU) (d) Routing (BBH)

Figure 4: Average few-shot performance of FLAN-MOE models over the 57 MMLU tasks and 23 BBH tasks. (Different color represents different dense model sizes.)

- Small, base model 모두 expert 수가 늘어날수록 성능 증가하는 경향
- 동시에 FLOPs 역시 증가

III. Experiment

- **Routing strategy**

  - routing: overfitting을 막고 모델의 capacity를 최대로 활용하기 위해 매우 중요
  - 적절한 balance로 expert가 선택되어야 빠른 속도 확보
  - 2가지 routing 전략:

    (1) Switch Transformer Top-1 token-choice gating (Flan-Switch)

    → Token select the top-k experts

    (2) Gshard Top-2 token-choice gating (Flan-GS)

    (3) Expert choice Top-2 gating (Flan-EC)

    → Expert select the top-k tokens

    (4) ST-MoE token-choice Top-2 gating + auxiliary loss (load balancing) (Flan-ST)

III. Experiment

- **Routing strategy**

Model	FLOPs per token	Total # Params	MMLU Direct	MMLU CoT	BBH Direct	BBH CoT	Reasoning CoT	QA Direct	Norm. Avg.
$\text{Switch}_{\text{BASE}}$	0.3G	3.5B	28.3	13.6	0.1	1.4	5.2	35.8	20.2
FLAN-$\text{Switch}_{\text{BASE}}$	0.3G	3.5B	38.0	34.2	33.2	29.4	18.6	58.0	36.8 (+16.6)
$\text{Switch}_{\text{LARGE}}$	1.0G	26B	24.0	23.1	0.2	7.2	12.4	33.7	17.7
FLAN-$\text{Switch}_{\text{LARGE}}$	1.0G	26B	46.1	40.3	36.3	28.0	25.3	66.5	43.5 (+25.8)
$\text{Switch}_{\text{XXL}}$	13.9G	395B	24.6	15.1	0.0	6.7	9.2	32.5	17.8
FLAN-$\text{Switch}_{\text{XXL}}$	13.9G	395B	55.6	50.1	47.9	43.5	46.6	78.8	54.2 (+36.4)
GS_{SMALL}	0.06G	0.3B	23.9	0.0	0.2	0.8	0.8	24.1	16.7
FLAN-GS_{SMALL}	0.06G	0.3B	32.6	26.9	29.6	20.9	16.1	48.9	31.8 (+15.1)
GS_{BASE}	0.3G	1.3B	25.0	15.9	0.0	4.8	3.8	26.8	17.6
FLAN-GS_{BASE}	0.3G	1.3B	39.9	33.6	33.7	25.1	22.0	57.9	38.3 (+20.7)
GS_{LARGE}	1.0G	9.2B	26.4	12.8	0.2	14.3	13.0	31.9	19.2
FLAN-GS_{LARGE}	1.0G	9.2B	47.8	40.8	35.0	29.2	27.6	69.5	44.5 (+25.3)
GS_{XL}	03.6G	17.4B	25.7	10.0	0.0	0.0	10.4	35.0	18.7
FLAN-GS_{XL}	3.6G	17.4B	51.1	42.3	40.1	31.4	34.3	73.9	48.7 (+30.0)
EC_{SMALL}	0.06G	0.3B	25.3	1.2	0.1	2.3	0.8	36.0	18.1
FLAN-EC_{SMALL}	0.06G	0.3B	34.1	25.1	29.2	22.1	16.6	58.1	33.1 (+15.0)
EC_{BASE}	0.3G	1.3B	25.0	25.9	0.0	1.4	14.3	35.7	18.5
FLAN-EC_{BASE}	0.3G	1.3B	42.7	33.0	34.0	26.7	22.2	61.5	40.3 (+21.8)
EC_{LARGE}	1.0G	9.2B	23.4	12.6	0.0	8.6	6.7	40.1	17.3
FLAN-EC_{LARGE}	1.0G	9.2B	48.3	44.5	37.9	32.0	32.2	73.1	46.4 (+29.1)
EC_{XL}	3.6G	17.4B	26.7	11.0	0.0	1.9	12.4	34.2	19.4
FLAN-EC_{XL}	3.6G	17.4B	52.1	41.4	40.3	33.2	38.1	74.3	49.4 (+30.0)
ST_{BASE}	0.3G	1.3B	25.2	17.7	0.0	14.0	12.6	25.7	18.1
FLAN-ST_{BASE}	0.3G	1.3B	42.4	35.5	34.9	26.4	22.5	61.5	40.4 (+21.8)
ST_{32B}	32.1G	259B	25.5	15.1	0.0	5.5	9.8	32.1	18.4
FLAN-ST_{32B}	32.1G	259B	65.4	63.0	54.4	47.4	66.3	63.9	63.6 (+45.2)

- FLAN-EC > FLAN-GS

  → expert choice가 token choice보다 우수 (논문에 따라 결과 상이)

- expert load balancing을 추가한 FLAN-ST와는 미미한 성능차

III. Experiment

- **Scaling up FLAN-MoE**

Model	FLOPs per token	Total # Params	MMLU Direct	MMLU CoT	BBH Direct	BBH CoT	Reasoning CoT	QA Direct	Norm. Avg.
$T5_{SMALL}$	0.06G	80M	26.7	7.2	26.7	5.6	10.3	33.8	26.3
FLAN-$T5_{SMALL}$	0.06G	80M	28.7	12.1	29.1	19.2	15.0	40.9	28.7 (+2.4)
$T5_{BASE}$	0.3G	250M	25.7	14.1	27.7	14.6	14.7	35.3	26.2
FLAN-$T5_{BASE}$	0.3G	250M	35.6	33.3	30.3	26.8	16.4	48.8	33.9 (+7.7)
$T5_{LARGE}$	1.0G	780M	25.1	15.3	27.7	16.2	11.9	36.4	25.7
FLAN-$T5_{LARGE}$	1.0G	780M	44.7	38.9	34.7	28.5	22.2	64.6	42.0 (+16.3)
$T5_{XL}$	3.6G	3B	25.3	14.1	27.4	19.3	14.2	38.2	25.9
FLAN-$T5_{XL}$	3.6G	3B	50.3	46.1	40.2	35.9	33.9	74.1	48.0 (+22.1)
$T5_{XXL}$	13.9G	11B	26.1	19.1	29.5	19.3	21.4	47.4	27.7
FLAN-$T5_{XXL}$	13.9G	11B	52.6	47.9	45.6	41.6	46.3	80.4	51.7 (+24.0)
PaLM	12.6G	8B	24.3	24.1	30.8	30.1	24.9	47.6	27.1
FLAN-PaLM	12.6G	8B	49.3	41.3	36.4	31.1	36.9	75.1	47.5 (+20.4)
PaLM	91.6G	62B	55.1	49.0	37.4	43.0	50.6	70.4	51.0
FLAN-PaLM	91.6G	62B	59.6	56.9	47.5	44.9	59.7	85.3	57.6 (+6.6)
PaLM	847G	540B	71.3	62.9	49.1	63.7	72.6	86.0	66.2
FLAN-PaLM	847G	540B	73.5	70.9	57.9	66.3	76.5	89.9	70.3 (+4.1)
ST_{BASE}	0.3G	1.3B	25.2	17.7	0.0	14.0	12.6	25.7	18.1
FLAN-ST_{BASE}	0.3G	1.3B	42.4	35.5	34.9	26.4	22.5	61.5	40.4 (+21.8)
ST_{32B}	32.1G	259B	25.5	15.1	0.0	5.5	9.8	32.1	18.4
FLAN-ST_{32B}	32.1G	259B	65.4	63.0	54.4	47.4	66.3	63.9	63.6 (+45.2)

- Flan-PaLM62B vs Flan-ST32B
  Flan-ST 가 더 우수한 성능

  Flan-PaLM은 약 3배 정도의 computational resource를 더 활용하였음에도 낮은 성능 기록

- Total params는 훨씬 크더라도 FLOPs는 월등히 낮음

IV. Discussion

- **Auxiliary loss**

  - 특정 expert에 치우치지 않도록 함으로써 expert의 지식을 다양화하도록 촉진시킴 → generalization ability 향상 도모

Finetuning Strategy	MMLU Direct	BBH Direct	GSM8K CoT	Avg.
$\text{Baseline}_{\text{FLAN-EC}_{\text{BASE}}}$	40.0	33.2	6.6	37.7
$\text{Freeze-Gate}_{\text{FLAN-EC}_{\text{BASE}}}$	40.2	33.9	6.6	38.0
$\text{Freeze-Expert}_{\text{FLAN-EC}_{\text{BASE}}}$	38.3	32.5	5.4	36.2
$\text{Freeze-MoE}_{\text{FLAN-EC}_{\text{BASE}}}$	38.4	32.2	5.3	36.2
$\text{Z-loss}_{\text{FLAN-EC}_{\text{BASE}}}$	38.9	32.8	5.7	36.8
$\text{Balance-loss}_{\text{FLAN-EC}_{\text{BASE}}}$	40.8	33.4	7.1	38.3

Finetuning Strategy	MMLU Direct	BBH Direct	GSM8K CoT	Avg.
$\text{Baseline}_{\text{FLAN-ST}_{\text{BASE}}}$	40.1	33.3	6.4	37.8
$\text{Freeze-Gate}_{\text{FLAN-ST}_{\text{BASE}}}$	40.6	33.5	6.4	38.2
$\text{Freeze-Expert}_{\text{FLAN-ST}_{\text{BASE}}}$	39.6	32.9	4.5	37.3
$\text{Freeze-MoE}_{\text{FLAN-ST}_{\text{BASE}}}$	39.2	32.9	3.6	36.9
$\text{Z-loss}_{\text{FLAN-ST}_{\text{BASE}}}$	40.6	33.4	6.5	38.1
$\text{Balance-loss}_{\text{FLAN-ST}_{\text{BASE}}}$	38.8	31.3	3.6	36.2

Table 2: Ablations on different finetuning strategies of $\text{FLAN-EC}_{\text{BASE}}$ and $\text{FLAN-ST}_{\text{BASE}}$.

- FLAN-EC → Z-loss < Balance-loss
- FLAN-ST → Z-loss > Balance-loss
- 오히려 loss 자체를 쓰지 않는 것이 도움이 되기도 함
- 사용하는 모델 및 세팅에 따라 가변적임

IV. Discussion

- **Expert/Gating Freeze**

  - model parameter의 일부만 업데이트

    -> ST-MoE 모델에서는 generalization performance가 향상되었음을 보이기도 함

Finetuning Strategy	MMLU Direct	BBH Direct	GSM8K CoT	Avg.
$\text{Baseline}_{\text{FLAN-EC}_{\text{BASE}}}$	40.0	33.2	6.6	37.7
$\text{Freeze-Gate}_{\text{FLAN-EC}_{\text{BASE}}}$	40.2	33.9	6.6	38.0
$\text{Freeze-Expert}_{\text{FLAN-EC}_{\text{BASE}}}$	38.3	32.5	5.4	36.2
$\text{Freeze-MoE}_{\text{FLAN-EC}_{\text{BASE}}}$	38.4	32.2	5.3	36.2
$\text{Z-loss}_{\text{FLAN-EC}_{\text{BASE}}}$	38.9	32.8	5.7	36.8
$\text{Balance-loss}_{\text{FLAN-EC}_{\text{BASE}}}$	40.8	33.4	7.1	38.3

Finetuning Strategy	MMLU Direct	BBH Direct	GSM8K CoT	Avg.
$\text{Baseline}_{\text{FLAN-ST}_{\text{BASE}}}$	40.1	33.3	6.4	37.8
$\text{Freeze-Gate}_{\text{FLAN-ST}_{\text{BASE}}}$	40.6	33.5	6.4	38.2
$\text{Freeze-Expert}_{\text{FLAN-ST}_{\text{BASE}}}$	39.6	32.9	4.5	37.3
$\text{Freeze-MoE}_{\text{FLAN-ST}_{\text{BASE}}}$	39.2	32.9	3.6	36.9
$\text{Z-loss}_{\text{FLAN-ST}_{\text{BASE}}}$	40.6	33.4	6.5	38.1
$\text{Balance-loss}_{\text{FLAN-ST}_{\text{BASE}}}$	38.8	31.3	3.6	36.2

Table 2: Ablations on different finetuning strategies of $\text{FLAN-EC}_{\text{BASE}}$ and $\text{FLAN-ST}_{\text{BASE}}$.

  - Router, expert, MoE 부분을 각각 freeze시켜봄 (나머지는 업데이트)

  → expert, MoE 부분을 freeze : 성능 저하

  → gate freeze : 미미한 성능 향상

- **Hyperparameter sensitivity**: 낮은 배치, 낮은 learning rate에서 학습이 안정적

IV. Discussion

- **Fine-tuning vs Instruction tuning**

  - single task에 대한 fine-tuning 실험

  - T5 → FT, MoE → FT, Flan-T5 → FT (T5 → IT → FT), FLAN-MoE → FT (MoE → IT → FT)

  - T5 → FT, Flan-T5 → FT 차이 < MoE → FT, Flan- MoE → FT 차이

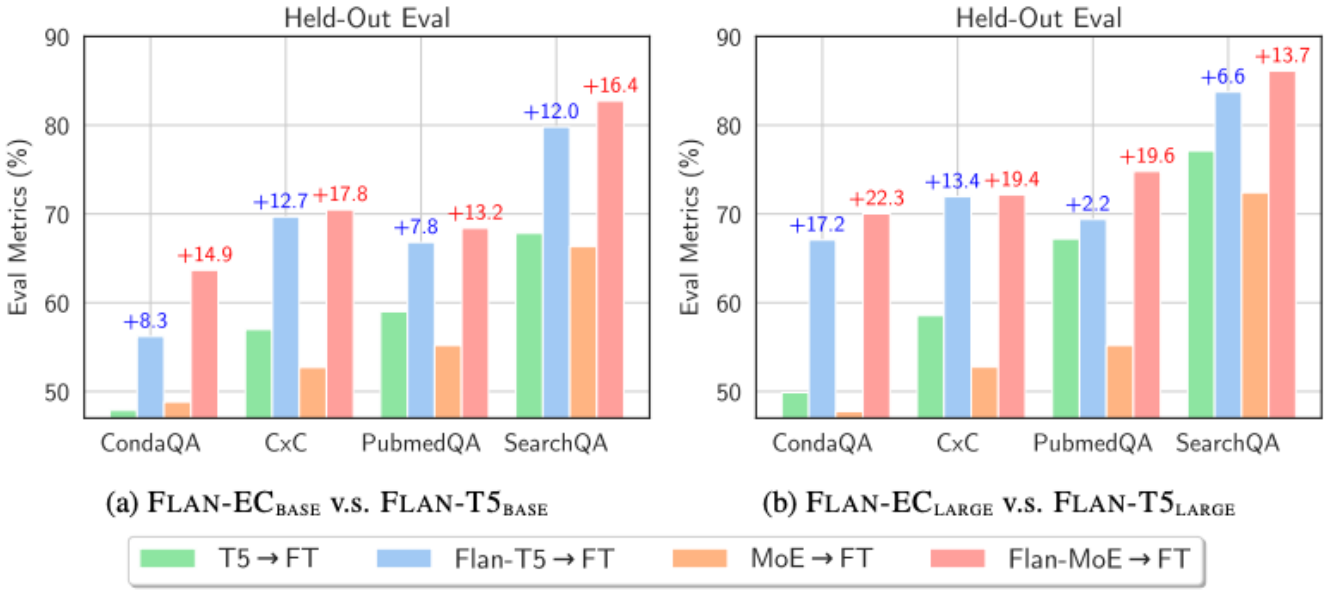


Figure 6: FLAN-MOE Outperforms MoE on Single-Task Finetuning. We compare single-task finetuned MoE, single-task finetuned FLAN-MOE, and dense counterparts. The performance gap between FLAN-MOE and MoE is noticeably larger than that between FLAN-T5 and T5.

IV. Discussion

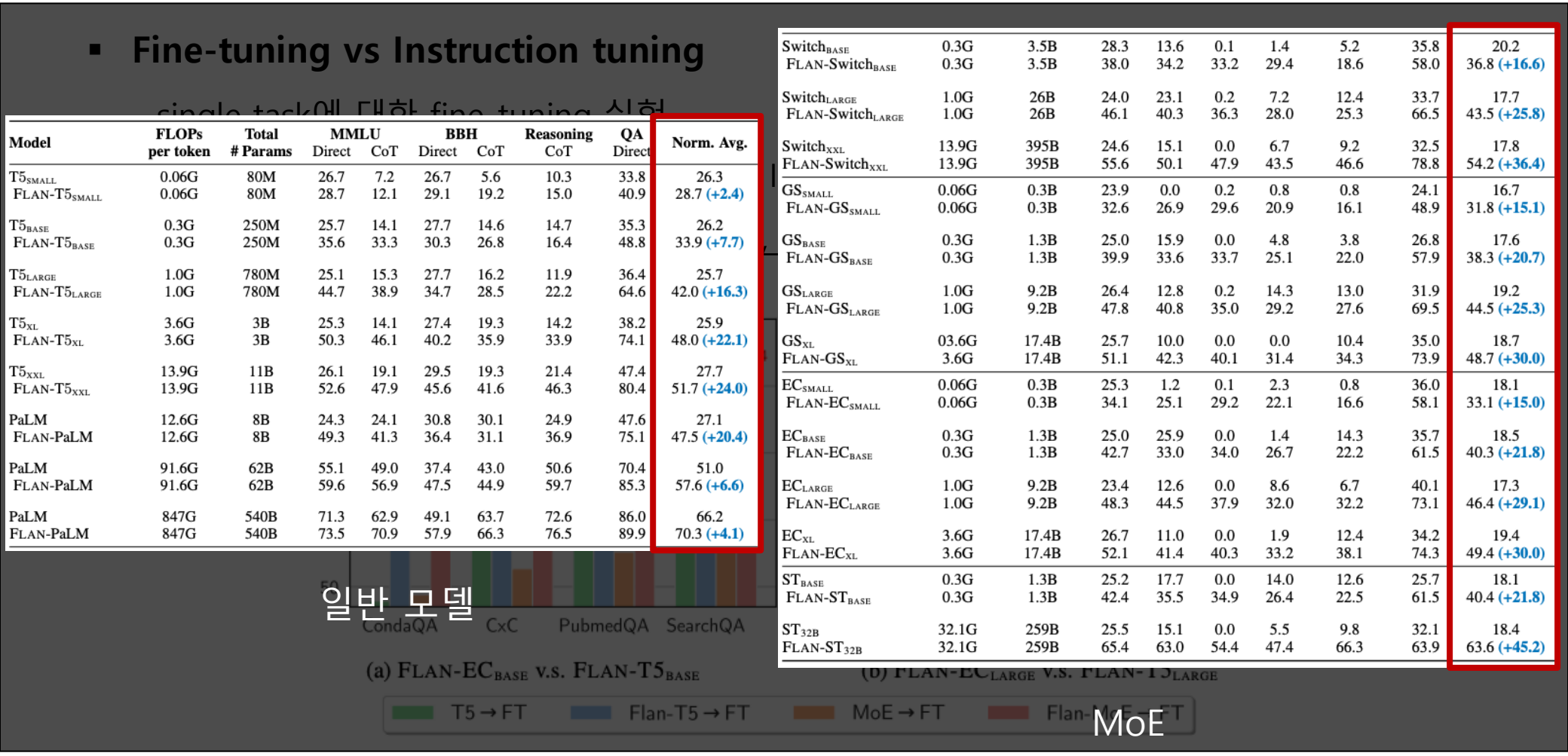
- **Fine-tuning vs Instruction tuning**

single task에 대한 fine-tuning 실험

Model	FLOPs per token	Total # Params	MMLU Direct	MMLU CoT	BBH Direct	BBH CoT	Reasoning CoT	QA Direct	Norm. Avg.
$T5_{SMALL}$	0.06G	80M	26.7	7.2	26.7	5.6	10.3	33.8	26.3
FLAN-$T5_{SMALL}$	0.06G	80M	28.7	12.1	29.1	19.2	15.0	40.9	28.7 (+2.4)
$T5_{BASE}$	0.3G	250M	25.7	14.1	27.7	14.6	14.7	35.3	26.2
FLAN-$T5_{BASE}$	0.3G	250M	35.6	33.3	30.3	26.8	16.4	48.8	33.9 (+7.7)
$T5_{LARGE}$	1.0G	780M	25.1	15.3	27.7	16.2	11.9	36.4	25.7
FLAN-$T5_{LARGE}$	1.0G	780M	44.7	38.9	34.7	28.5	22.2	64.6	42.0 (+16.3)
$T5_{XL}$	3.6G	3B	25.3	14.1	27.4	19.3	14.2	38.2	25.9
FLAN-$T5_{XL}$	3.6G	3B	50.3	46.1	40.2	35.9	33.9	74.1	48.0 (+22.1)
$T5_{XXL}$	13.9G	11B	26.1	19.1	29.5	19.3	21.4	47.4	27.7
FLAN-$T5_{XXL}$	13.9G	11B	52.6	47.9	45.6	41.6	46.3	80.4	51.7 (+24.0)
PaLM	12.6G	8B	24.3	24.1	30.8	30.1	24.9	47.6	27.1
FLAN-PaLM	12.6G	8B	49.3	41.3	36.4	31.1	36.9	75.1	47.5 (+20.4)
PaLM	91.6G	62B	55.1	49.0	37.4	43.0	50.6	70.4	51.0
FLAN-PaLM	91.6G	62B	59.6	56.9	47.5	44.9	59.7	85.3	57.6 (+6.6)
PaLM	847G	540B	71.3	62.9	49.1	63.7	72.6	86.0	66.2
FLAN-PaLM	847G	540B	73.5	70.9	57.9	66.3	76.5	89.9	70.3 (+4.1)

$Switch_{BASE}$	0.3G	3.5B	28.3	13.6	0.1	1.4	5.2	35.8	20.2
FLAN-$Switch_{BASE}$	0.3G	3.5B	38.0	34.2	33.2	29.4	18.6	58.0	36.8 (+16.6)
$Switch_{LARGE}$	1.0G	26B	24.0	23.1	0.2	7.2	12.4	33.7	17.7
FLAN-$Switch_{LARGE}$	1.0G	26B	46.1	40.3	36.3	28.0	25.3	66.5	43.5 (+25.8)
$Switch_{XXL}$	13.9G	395B	24.6	15.1	0.0	6.7	9.2	32.5	17.8
FLAN-$Switch_{XXL}$	13.9G	395B	55.6	50.1	47.9	43.5	46.6	78.8	54.2 (+36.4)
GS_{SMALL}	0.06G	0.3B	23.9	0.0	0.2	0.8	0.8	24.1	16.7
FLAN-GS_{SMALL}	0.06G	0.3B	32.6	26.9	29.6	20.9	16.1	48.9	31.8 (+15.1)
GS_{BASE}	0.3G	1.3B	25.0	15.9	0.0	4.8	3.8	26.8	17.6
FLAN-GS_{BASE}	0.3G	1.3B	39.9	33.6	33.7	25.1	22.0	57.9	38.3 (+20.7)
GS_{LARGE}	1.0G	9.2B	26.4	12.8	0.2	14.3	13.0	31.9	19.2
FLAN-GS_{LARGE}	1.0G	9.2B	47.8	40.8	35.0	29.2	27.6	69.5	44.5 (+25.3)
GS_{XL}	03.6G	17.4B	25.7	10.0	0.0	0.0	10.4	35.0	18.7
FLAN-GS_{XL}	3.6G	17.4B	51.1	42.3	40.1	31.4	34.3	73.9	48.7 (+30.0)
EC_{SMALL}	0.06G	0.3B	25.3	1.2	0.1	2.3	0.8	36.0	18.1
FLAN-EC_{SMALL}	0.06G	0.3B	34.1	25.1	29.2	22.1	16.6	58.1	33.1 (+15.0)
EC_{BASE}	0.3G	1.3B	25.0	25.9	0.0	1.4	14.3	35.7	18.5
FLAN-EC_{BASE}	0.3G	1.3B	42.7	33.0	34.0	26.7	22.2	61.5	40.3 (+21.8)
EC_{LARGE}	1.0G	9.2B	23.4	12.6	0.0	8.6	6.7	40.1	17.3
FLAN-EC_{LARGE}	1.0G	9.2B	48.3	44.5	37.9	32.0	32.2	73.1	46.4 (+29.1)
EC_{XL}	3.6G	17.4B	26.7	11.0	0.0	1.9	12.4	34.2	19.4
FLAN-EC_{XL}	3.6G	17.4B	52.1	41.4	40.3	33.2	38.1	74.3	49.4 (+30.0)
ST_{BASE}	0.3G	1.3B	25.2	17.7	0.0	14.0	12.6	25.7	18.1
FLAN-ST_{BASE}	0.3G	1.3B	42.4	35.5	34.9	26.4	22.5	61.5	40.4 (+21.8)
ST_{32B}	32.1G	259B	25.5	15.1	0.0	5.5	9.8	32.1	18.4
FLAN-ST_{32B}	32.1G	259B	65.4	63.0	54.4	47.4	66.3	63.9	63.6 (+45.2)

일반 모델



MoE

Figure 6: FLAN-MoE Outperforms MoE on Single-Task Finetuning. We compare single-task finetuned MoE, single-task finetuned FLAN-MoE, and dense counterparts. The performance gap between FLAN-MoE and MoE is noticeably larger than that between FLAN-T5 and T5.

IV. Discussion

- **Fine-tuning vs Instruction tuning**

  - task 수에 따른 성능 변화

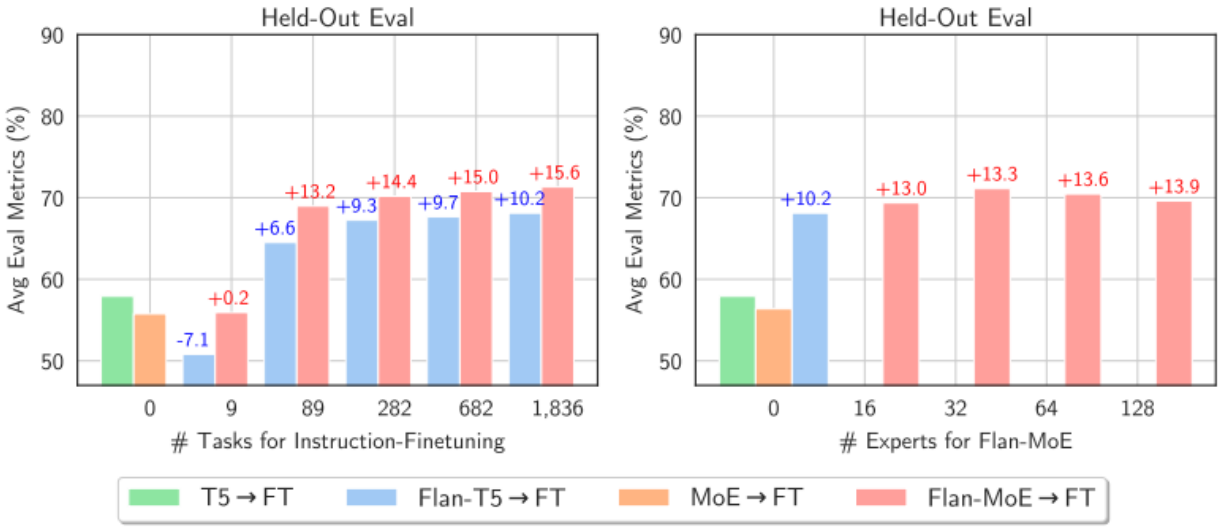


Figure 1: The effect of instruction tuning on MoE models versus dense counterparts for base-size models (same flops across all models in this figure). We perform single-task finetuning for each model on held-out benchmarks. **Compared to dense models, MoE models benefit more from instruction-tuning, and are more sensitive to the number of instruction-tuning tasks.** Overall, the performance of MoE models scales better with respect to the number of tasks, than the number of experts.

- Task 수가 늘어날수록 MoE는 더 beneficial
- 개별 expert 들이 활성화되었을 때의 문제:
  noise에 민감, unseen task에 대한 일반화 능력 제한
- Instruction tuning을 통해 다양한 task들을 학습 시 보여줌으로써 학습이 적절하게 진행됨

V. Conclusion

- Flan-MoE 제안
- MoE 모델에 바로 FT 수행 < MoE 모델에 instruction tuning 수행 후 FT 진행
- Task가 많을수록, 모델 사이즈가 클수록 MoE에 유리
- Instruction tuning을 통해 expert마다의 전문성 증가, task별 활성화되는 expert를 달리함으로써 분업을 통한 computational cost를 상당히 줄임

[리뷰]

- 가장 큰 문제는 base MoE 모델이 필요
- Sparse upcycling 논문(Google, ICLR 2023) (pretraining → MoE 추가학습)의 방식을 추가 활용할 필요
- 논문의 순서나 실험 구성 부분에서의 아쉬움

Pushing Mixture of Experts to the Limit: Extremely Parameter Efficient MoE for Instruction Tuning

Ted Zadouri
Cohere for AI
ted@cohere.com

Ahmet Üstün
Cohere for AI
ahmet@cohere.com

Arash Ahmadian[†]
Cohere for AI
arash@cohere.com

Beyza Ermiş
Cohere For AI
beyza@cohere.com

Acyr Locatelli
Cohere
acyr@cohere.com

Sara Hooker
Cohere for AI
sarahooker@cohere.com

I. Introduction

- **MoE**

  - Sub-modular component (expert)가 다른 유형의 입력에 대해 특화될 수 있다는 점을 전제로 등장

  - 주로 pretraining 전략으로 활용, 최근에는 instruction tuning (multi-task)에서도 활발히 적용

- **MoE의 가장 큰 결함**

  - 전체 파라미터가 상당히 큼

  - Conditional computation임에도 불구하고 전체 MoE finetuning 시에는 전체 파라미터 업데이트가 불가피

  - 대부분의 연구자들에게는 실행 불가능할 정도의 computational cost

→ 모든 practitioners를 위한 realistic setting 요구

I. Introduction

- **In this paper,**

  - 제안: Parameter-efficient adaptation of the Mixture-of-experts approach

    - Mixture of Vectors (MoV)
    - Mixture of LoRA (MoLORA)

  - Zero-shot generalization 능력 평가 (unseen task):

    - 기존의 PEFT approach 성능 능가 ($(IA)^3$, LoRA)
    - Full fine-tuning과 비슷한 성능
    - 파라미터는 full fine-tuning의 최대 1%만 활용

- **Contributions**

  - Extremely parameter-efficient MoE – modular and lightweight experts
  - 우수한 성능, demonstrated by extensive analysis

II. Methodology

- **Parameter-efficient fine-tuning with $(IA)^3$ and LoRA adapters [preliminary]**

[$(IA)^3$]

- 3개의 새로운 rescaling vectors 도입 & 업데이트: $l_k \in \mathbb{R}^{d_k}, l_v \in \mathbb{R}^{d_v}, l_{ff} \in \mathbb{R}^{d_{ff}}$
- self attention의 projection matrices, feed-forward layers에 결합

$$\text{softmax}\left(\frac{Q(l_k \odot K^T)}{\sqrt{d_k}}\right)(l_v \odot V); \quad (l_{ff} \odot \gamma\,(W_1 x))W_2$$

- only updates 0.018% of the total parameters

[LoRA]

- 2개의 low-rank matrices만 업데이트: $B \in \mathbb{R}^{d_p \times r}, A \in \mathbb{R}^{r \times d_m}, r \ll \min(d_m, d_p)$

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

- self attention의 projection matrices, feed-forward layers에 결합
- rank 4, only updates 0.3% of the total prameters

II. Methodology

- **Extremely parameter efficient Mixture-of-experts**

  - MoE: family of neural network architecture
  - conditional computation:여러 개의 experts를 두고 입력에 따라 일부 experts만 활성화
  - 활성화 결정: gating mechanism (router)
  - Router & MoE equation:

$$s_i = R(x)_i = \text{softmax}(W_g^T x)$$

$$y = \sum_{i=1}^{n} s_i \cdot E_i(x)$$

*Router network R | experts $E_1, \ldots, E_n$ | each expert(independent dense feed-forward layers) E_i | intermediate token representation x | gating scores $S_1, \ldots, S_n$

  - Soft merging 방법 적용:

$$E_{mix} = \sum_{i=1}^{n} s_i \cdot E_i; \;\; y = E_{mix}(x)$$

II. Methodology

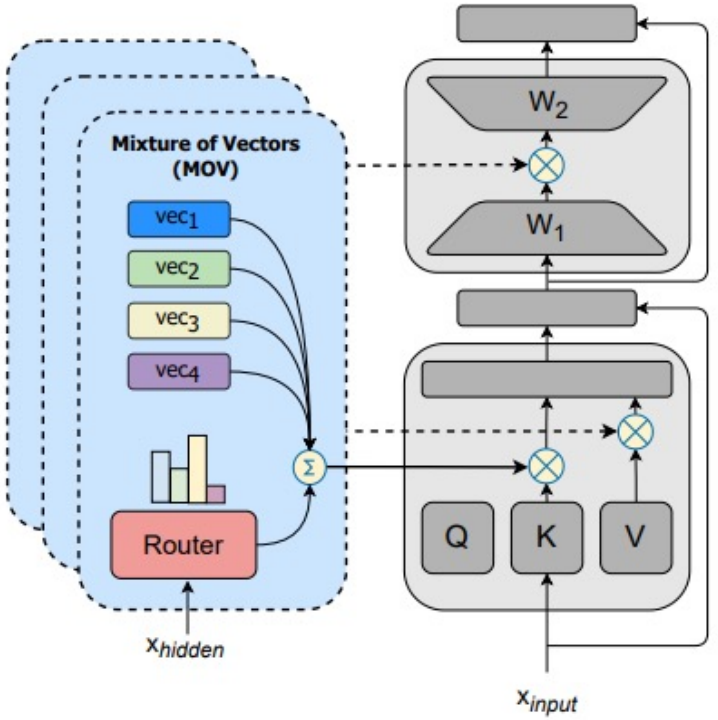
- **2 variants**

(1) Mixture of Vectors (MoV):

- (IA)^3
- weight averaging -> PEFT transformation 적용

(2) Mixture of LORA (MoLORA):

- LoRA
- outputs of LoRA adapters -> weight ageraging



MoV 구조

II. Methodology

- **2 variants**

(1) Mixture of Vectors (MoV):

- (IA)^3
- weight averaging -> PEFT transformation 적용

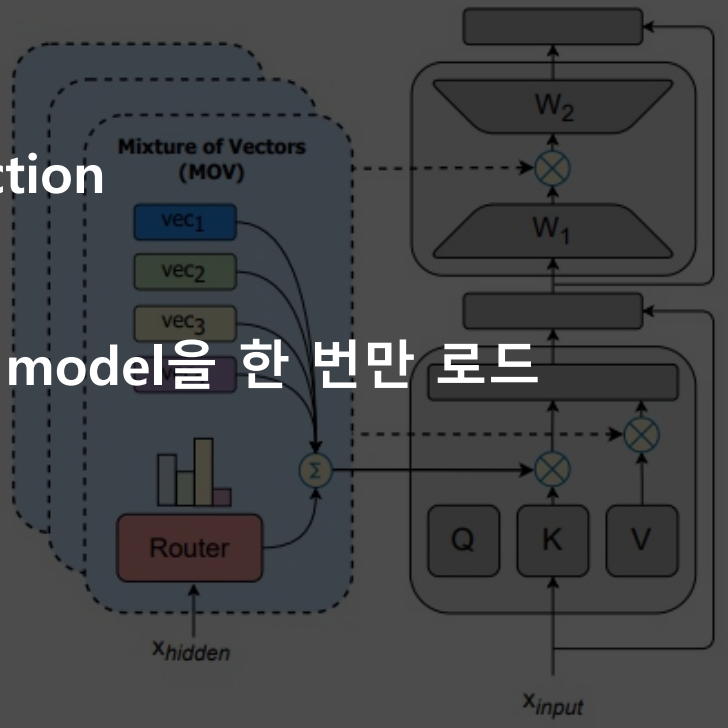
- **Efficiency in training → 메모리 reduction**

- **Efficiency at inference → pretrained model을 한 번만 로드**

(2) Mixture of LORA (MoLORA):

(기존 MoE → FFN 복사 및 저장 필요)

- LoRA
- outputs of LoRA adapters -> weight ageraging



MoV 구조

III. Experiments

- **Details**
  - Dataset:
    - Public Pool of Prompts (P3), 평가: 8 unseen data에 대한 제로샷 수행
  - Setup:
    - Base model → T5 v1.1 + LM adaptation (770M-11B)
    - MoE variants → T5 모델 fine-tuning
  - Baseline: T0

III. Experiments

- **Ablations**

(1) Routing input: Token vs sentence embeddings

(2) Routing strategy: Soft vs discrete

- Soft: routing block 내에서 모든 expert에 대한 weight averaging
- Discrete: top-k routing, k = {1,2}

(3) load balancing: top-k 뽑히는 것이 좀더 균형 잡힐 수 있도록 loss 추가

- $loss = \alpha \cdot n \cdot \sum_{i=1}^{n} f_i \cdot P_i$
- α = auxiliary loss의 multiplicative coefficient, 10^{-2}, cross-entropy loss에 크게 영향을 주지 않기 위함
- $f_i = \frac{1}{T} \sum_{x \in Batch} 1\{argmax\, R(x) = i\}$, expert i 로 배정된 tokens (one-hot)
- $P_i = \frac{1}{T} \sum_{x \in Batch} R(x)$, expert i 로 할당되는 router probability
- n = expert 수가 달라질 수 있음을 고려

IV. Results and Discussion

- **Parameter efficient MoE vs PEFTs**

Zero-shot Results at 3B Scale

	Model	% Params.	ANLI	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
Full-FT	T0-3B (Sanh et al., 2022)	100%	33.46	50.0	64.08	64.42	50.39	74.92	50.51	27.51	51.91
	T0-3B (our replication)	100%	41.08	80.36	76.17	53.37	53.92	88.94	57.46	29.19	60.06
PEFT	$(IA)^3$	0.018%	34.08	50.0	66.43	56.25	55.41	79.08	52.09	29.91	52.90
	LORA (rank 4)	0.3%	37.5	75.57	73.53	61.02	51.25	83.6	54.33	25.32	57.51
	LORA (rank 8)	0.6%	37.5	75.0	77.98	62.5	51.49	83.67	55.72	27.3	58.89
	LORA (rank 16)	1.2%	38.5	80.36	76.71	63.46	51.02	84.5	54.7	27.11	59.54
Our Method	MoV-10	0.32%	38.92	75.0	78.88	62.5	52.19	85.77	55.96	30.24	59.93
	MoV-30	0.68%	38.7	78.57	80.87	63.46	51.1	87.25	56.27	28.63	60.61
	MoV-60	1.22%	38.83	76.79	74.55	60.1	52.66	89.79	55.49	30.47	59.83
	MoLORA-2 (rank 4)	0.75%	39.2	82.14	80.32	62.5	50.39	80.58	57.38	28.47	60.12
	MoLORA-10 (rank 4)	3.18%	38.5	78.57	78.16	63.46	50.86	86.5	55.41	26.72	59.77
	MoLORA-15 (rank 4)	4.69%	40.0	80.36	80.51	62.98	50.86	89.0	55.33	27.3	60.79

Table 1: Average median results on unseen tasks for full model fine-tuning (T0), parameter-efficient fine-tune methods ($(IA)^3$ and LORA) and our mixture of parameter-efficient experts (MoV and MoLORA), using T5-3B base model (Raffel et al., 2020). Note that our T0 scores are significantly higher than the original T0 confirming previous work (Phang et al., 2023; Ivison et al., 2023).

- $(IA)^3$ vs MoV-30 → 성능 향상
- LoRA (rank 4) vs MoLORA-15 (rank 4) → 성능 향상
- MoV-30 vs MoLORA-15 (rank4) → 더 적은 파라미터로 비슷한 성능
- MoV-10, MoLora-10 (rank4) vs full-FT → 비슷한 성능 (파라미터 차이는 매우 큼)

IV. Results and Discussion

- **Parameter efficient MoE vs PEFTs**

Zero-shot Results at 3B Scale

	Model	% Params.	ANLI	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
Full-FT	T0-3B (Sanh et al., 2022)	100%	33.46	50.0	64.08	64.42	50.39	74.92	50.51	27.51	51.91
	T0-3B (our replication)	100%	41.08	80.36	76.17	53.37	53.92	88.94	57.46	29.19	60.06
PEFT	$(IA)^3$	0.018%	34.08	50.0	66.43	56.25	55.41	79.08	52.09	29.91	52.90
	LORA (rank 4)	0.3%	37.5	75.57	73.53	61.02	51.25	83.6	54.33	25.32	57.51
	LORA (rank 8)	0.6%	37.5	75.0	77.98	62.5	51.49	83.67	55.72	27.3	58.89
	LORA (rank 16)	1.2%	38.5	80.36	76.71	63.46	51.02	84.5	54.7	27.11	59.54
Our Method	MoV-10	0.32%	38.92	75.0	78.88	62.5	52.19	85.77	55.96	30.24	59.93
	MoV-30	0.68%	38.7	78.57	80.87	63.46	51.1	87.25	56.27	28.63	60.61
	MoV-60	1.22%	38.83	76.79	74.55	60.1	52.66	89.79	55.49	30.47	59.83
	MoLORA-2 (rank 4)	0.75%	39.2	82.14	80.32	62.5	50.39	80.58	57.38	28.47	60.12
	MoLORA-10 (rank 4)	3.18%	38.5	78.57	78.16	63.46	50.86	86.5	55.41	26.72	59.77
	MoLORA-15 (rank 4)	4.69%	40.0	80.36	80.51	62.98	50.86	89.0	55.33	27.3	60.79

Table 1: Average median results on unseen tasks for full model fine-tuning (T0), parameter-efficient fine-tune methods ($(IA)^3$ and LORA) and our mixture of parameter-efficient experts (MoV and MoLORA), using T5-3B base model (Raffel et al., 2020). Note that our T0 scores are significantly higher than the original T0 confirming previous work (Phang et al., 2023; Ivison et al., 2023).

- $(IA)^3$ vs MoV-30 → 성능 향상
- LoRA (rank 4) vs MoLORA-15 (rank 4) → 성능 향상
- MoV-30 vs MoLORA-15 (rank4) → 더 적은 파라미터로 비슷한 성능
- MoV-10, MoLora-10 (rank4) vs full-FT → 비슷한 성능 (파라미터 차이는 매우 큼)

IV. Results and Discussion

- **Parameter efficient MoE vs PEFTs**

Zero-shot Results at 3B Scale

	Model	% Params.	ANLI	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
Full-FT	T0-3B (Sanh et al., 2022)	100%	33.46	50.0	64.08	64.42	50.39	74.92	50.51	27.51	51.91
	T0-3B (our replication)	100%	41.08	80.36	76.17	53.37	53.92	88.94	57.46	29.19	60.06
PEFT	$(IA)^3$	0.018%	34.08	50.0	66.43	56.25	55.41	79.08	52.09	29.91	52.90
	LORA (rank 4)	0.3%	37.5	75.57	73.53	61.02	51.25	83.6	54.33	25.32	57.51
	LORA (rank 8)	0.6%	37.5	75.0	77.98	62.5	51.49	83.67	55.72	27.3	58.89
	LORA (rank 16)	1.2%	38.5	80.36	76.71	63.46	51.02	84.5	54.7	27.11	59.54
Our Method	MoV-10	0.32%	38.92	75.0	78.88	62.5	52.19	85.77	55.96	30.24	59.93
	MoV-30	0.68%	38.7	78.57	80.87	63.46	51.1	87.25	56.27	28.63	60.61
	MoV-60	1.22%	38.83	76.79	74.55	60.1	52.66	89.79	55.49	30.47	59.83
	MoLORA-2 (rank 4)	0.75%	39.2	82.14	80.32	62.5	50.39	80.58	57.38	28.47	60.12
	MoLORA-10 (rank 4)	3.18%	38.5	78.57	78.16	63.46	50.86	86.5	55.41	26.72	59.77
	MoLORA-15 (rank 4)	4.69%	40.0	80.36	80.51	62.98	50.86	89.0	55.33	27.3	60.79

Table 1: Average median results on unseen tasks for full model fine-tuning (T0), parameter-efficient fine-tune methods ($(IA)^3$ and LORA) and our mixture of parameter-efficient experts (MoV and MoLORA), using T5-3B base model (Raffel et al., 2020). Note that our T0 scores are significantly higher than the original T0 confirming previous work (Phang et al., 2023; Ivison et al., 2023).

- $(IA)^3$ vs MoV-30 → 성능 향상
- LoRA (rank 4) vs MoLORA-15 (rank 4) → 성능 향상
- MoV-30 vs MoLORA-15 (rank4) → 더 적은 파라미터로 비슷한 성능
- MoV-10, MoLora-10 (rank4) vs full-FT → 비슷한 성능 (파라미터 차이는 매우 큼)

IV. Results and Discussion

- **Parameter efficient MoE vs PEFTs**

Zero-shot Results at 3B Scale

	Model	% Params.	ANLI	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
Full-FT	T0-3B (Sanh et al., 2022)	100%	33.46	50.0	64.08	64.42	50.39	74.92	50.51	27.51	51.91
	T0-3B (our replication)	100%	41.08	80.36	76.17	53.37	53.92	88.94	57.46	29.19	60.06
PEFT	$(IA)^3$	0.018%	34.08	50.0	66.43	56.25	55.41	79.08	52.09	29.91	52.90
	LORA (rank 4)	0.3%	37.5	75.57	73.53	61.02	51.25	83.6	54.33	25.32	57.51
	LORA (rank 8)	0.6%	37.5	75.0	77.98	62.5	51.49	83.67	55.72	27.3	58.89
	LORA (rank 16)	1.2%	38.5	80.36	76.71	63.46	51.02	84.5	54.7	27.11	59.54
Our Method	MoV-10	0.32%	38.92	75.0	78.88	62.5	52.19	85.77	55.96	30.24	59.93
	MoV-30	0.68%	38.7	78.57	80.87	63.46	51.1	87.25	56.27	28.63	60.61
	MoV-60	1.22%	38.83	76.79	74.55	60.1	52.66	89.79	55.49	30.47	59.83
	MoLORA-2 (rank 4)	0.75%	39.2	82.14	80.32	62.5	50.39	80.58	57.38	28.47	60.12
	MoLORA-10 (rank 4)	3.18%	38.5	78.57	78.16	63.46	50.86	86.5	55.41	26.72	59.77
	MoLORA-15 (rank 4)	4.69%	40.0	80.36	80.51	62.98	50.86	89.0	55.33	27.3	60.79

Table 1: Average median results on unseen tasks for full model fine-tuning (T0), parameter-efficient fine-tune methods ($(IA)^3$ and LORA) and our mixture of parameter-efficient experts (MoV and MoLORA), using T5-3B base model (Raffel et al., 2020). Note that our T0 scores are significantly higher than the original T0 confirming previous work (Phang et al., 2023; Ivison et al., 2023).

- $(IA)^3$ vs MoV-30 → 성능 향상
- LoRA (rank 4) vs MoLORA-15 (rank 4) → 성능 향상
- MoV-30 vs MoLORA-15 (rank4) → 더 적은 파라미터로 비슷한 성능
- MoV-10, MoLora-10 (rank4) vs full-FT → 비슷한 성능 (파라미터 차이는 매우 큼)

IV. Results and Discussion

- **Parameter efficient MoE vs PEFTs**

Zero-shot Results at 3B Scale

	Model	% Params.	ANLI	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
Full-FT	T0-3B (Sanh et al., 2022)	100%	33.46	50.0	64.08	64.42	50.39	74.92	50.51	27.51	51.91
	T0-3B (our replication)	100%	41.08	80.36	76.17	53.37	53.92	88.94	57.46	29.19	60.06
PEFT	$(IA)^3$	0.018%	34.08	50.0	66.43	56.25	55.41	79.08	52.09	29.91	52.90
	LORA (rank 4)	0.3%	37.5	75.57	73.53	61.02	51.25	83.6	54.33	25.32	57.51
	LORA (rank 8)	0.6%	37.5	75.0	77.98	62.5	51.49	83.67	55.72	27.3	58.89
	LORA (rank 16)	1.2%	38.5	80.36	76.71	63.46	51.02	84.5	54.7	27.11	59.54
Our Method	MoV-10	0.32%	38.92	75.0	78.88	62.5	52.19	85.77	55.96	30.24	59.93
	MoV-30	0.68%	38.7	78.57	80.87	63.46	51.1	87.25	56.27	28.63	60.61
	MoV-60	1.22%	38.83	76.79	74.55	60.1	52.66	89.79	55.49	30.47	59.83
	MoLORA-2 (rank 4)	0.75%	39.2	82.14	80.32	62.5	50.39	80.58	57.38	28.47	60.12
	MoLORA-10 (rank 4)	3.18%	38.5	78.57	78.16	63.46	50.86	86.5	55.41	26.72	59.77
	MoLORA-15 (rank 4)	4.69%	40.0	80.36	80.51	62.98	50.86	89.0	55.33	27.3	60.79

Table 1: Average median results on unseen tasks for full model fine-tuning (T0), parameter-efficient fine-tune methods ($(IA)^3$ and LORA) and our mixture of parameter-efficient experts (MoV and MoLORA), using T5-3B base model (Raffel et al., 2020). Note that our T0 scores are significantly higher than the original T0 confirming previous work (Phang et al., 2023; Ivison et al., 2023).

- $(IA)^3$ vs MoV-30 → 성능 향상
- LoRA (rank 4) vs MoLORA-15 (rank 4) → 성능 향상
- MoV-30 vs MoLORA-15 (rank4) → 더 적은 파라미터로 비슷한 성능
- MoV-10, MoLora-10 (rank4) vs full-FT → 비슷한 성능 (파라미터 차이는 매우 큼)

IV. Results and Discussion

- **MoLORA outperforms MoV in smaller model size regimes**

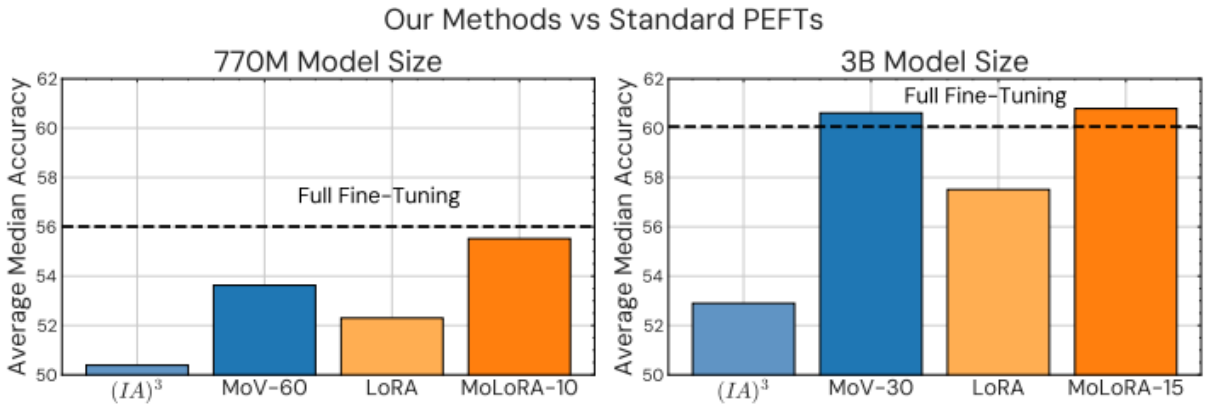


Figure 3: Comparison of the top-performing variants from our proposed mixture of PEFT experts versus their dense counterparts across T5-Large (*Left*) and T5-XL (*Right*).

모델 사이즈별 성능 비교

- 770M : MoLORA-10 > MoV-60
- 이전 실험 결과에서는 MoV가 더 우수, 작은 사이즈에서는 MoLORA가 더 우수한 성능
- 3B : 두 모델 모두 fine-tuning 보다 우수한 성능

IV. Results and Discussion

- **How does the number of experts impact the downstream performance?**

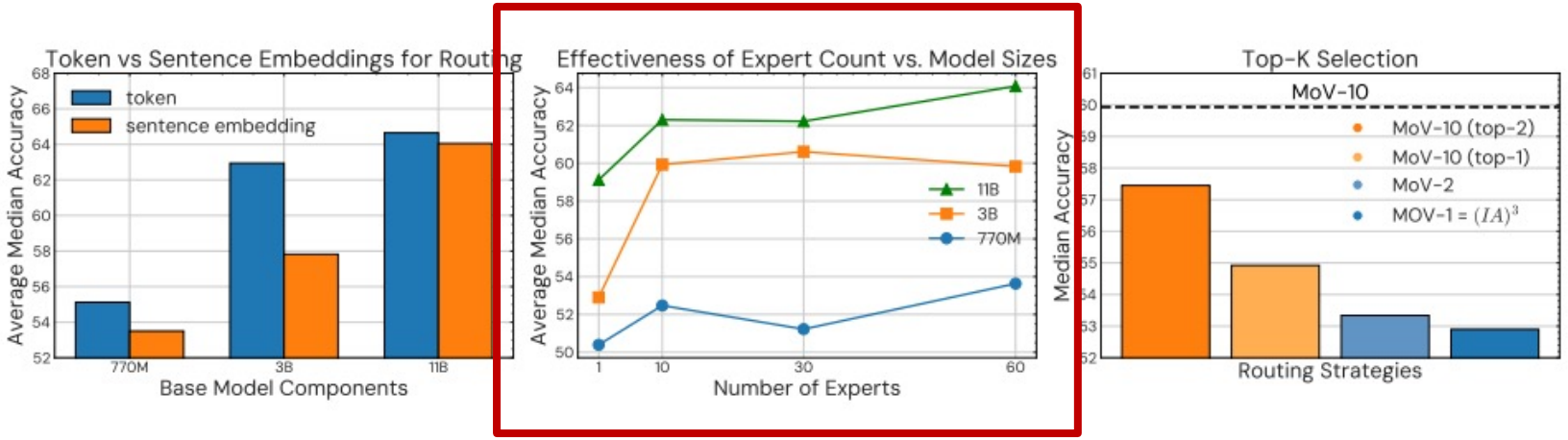


Figure 4: *Left:* Zero-shot performance of passing embedding of the token sequence to the router vs. passing tokens to the router. *Middle:* Zero-shot performance across T5 model sizes (Large, XL, XXL) as the number of experts increases. *Right:* The effectiveness of activating top-k experts.

Experts 수 변경에 따른 성능 비교

- 770M, 11B에서는 expert 수 많을수록 성능 향상
- 3B에서는 expert 60 설정 시 성능 하락

IV. Results and Discussion

- **What is the best routing strategy in parameter-efficient MoEs?**

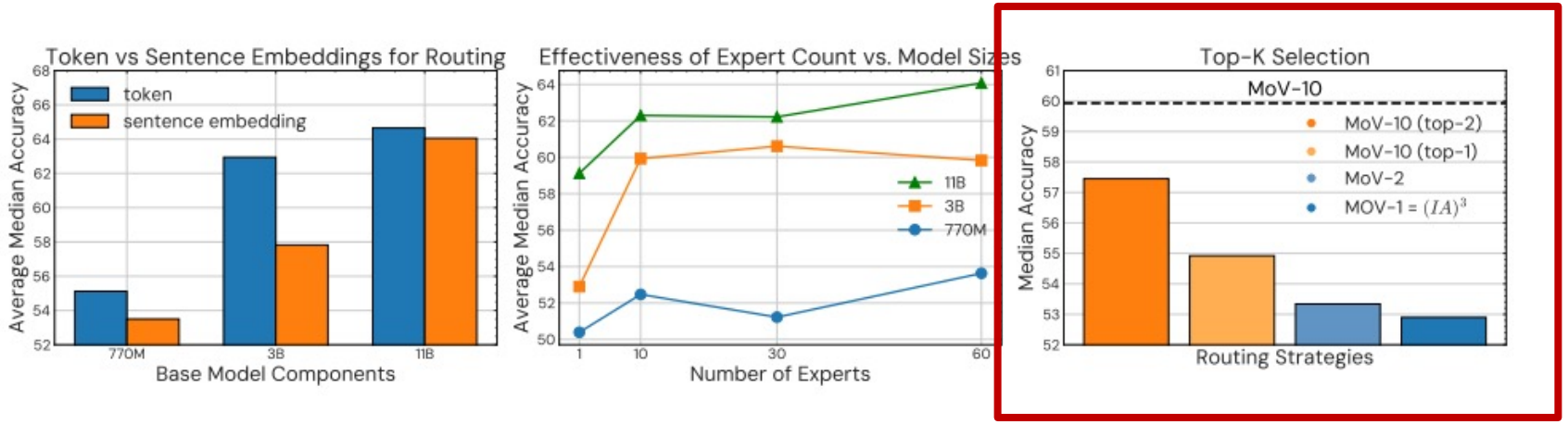


Figure 4: *Left:* Zero-shot performance of passing embedding of the token sequence to the router vs. passing tokens to the router. *Middle:* Zero-shot performance across T5 model sizes (Large, XL, XXL) as the number of experts increases. *Right:* The effectiveness of activating top-k experts.

Routing strategy별 성능 비교

- soft merging vs 다른 전략들 → soft merging이 더 우수
- MoV-10 (top-2, top-1) : top-2(1) gating (discrete)
- MoV-2 : (top-2+load balancing) → 성능 오히려 더 떨어짐
- MoV-1 : single expert ((IA)^3과 동일)

IV. Results and Discussion

- **Does sentence embeddings in routing lead to higher performance?**

Figure 4: *Left:* Zero-shot performance of passing embedding of the token sequence to the router vs. passing tokens to the router. *Middle:* Zero-shot performance across T5 model sizes (Large, XL, XXL) as the number of experts increases. *Right:* The effectiveness of activating top-k experts.

Embedding 변경에 따른 성능 변화 (Token vs Sentence)

- 외부 sentence embedding 모델로 뽑은 embedding 값을 넣어줌
- 성능 떨어지고 도움이 되지 못함

IV. Results and Discussion

- **Do experts specialize in diverse knowledge across different tasks?**

  - 태스크가 달라지면 expert routing이 어떻게 달라지는지에 대해서 알아보기 위함

    → 다양한 task에 대해 어떻게 expert들이 activate되는지 알아봄

    → 평균 라우팅 probabilities (expert contribution) 확인

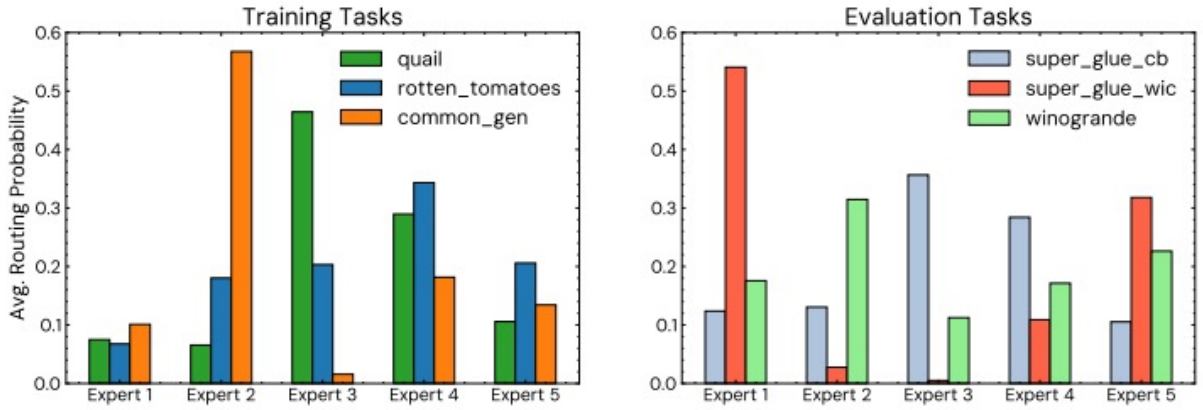
  - 학습, 추론(unseen task) 과정 모두 평가 → generalization 효과 검증

  - Expert probabilities (5 experts) - 마지막 디코더 블럭에 있는 FFN의 expert를 조사 (*마지막 layer로 갈수록 좀더 task-specific information을 학습한다고 함, Rogers et al, (2020))

  - **(가정)** 직관적으로 expert가 각각 다른 skill을 가졌더라면 다른 task들에 대해 각각 활성화되는 expert가 다를 것

IV. Results and Discussion

- **Do experts specialize in diverse knowledge across different tasks?**



- quail: multiple choice mrc 데이터셋
- Cb: 전제, 가설,이 있는 NLI
- commongen: commonsense로 문장 생성
- winogrande: 두 문장을 주고 각 대명사가 어떤 대상을 지칭하는지를 평가
- wic: ambiguous word의 두 문장을 보여주고 동일한 의미인지 아닌지 분류

Figure 5: Mean expert routing probabilities for intermediates activations at the last feedforward layer. Values are averaged across tokens and batch. Experts are weighted differently in soft merging depending on the task. *Left:* Measured on tasks seen during training. *Right:* Measured on unseen evaluation tasks.

- **Specialization across unseen vs seen tasks**
- quail, super_glue_cb → expert 3 - expert 4가 가장 높은 probability
- commongen, winogrande → expert 2 가 가장 높은 probability
- 어떤 downstream task에 대해 학습을 했는지와 관계없이 routing specialization은 등장
  → expert specialization은 내재화 되어 있음, seen → unseen task로의 전이가 가능함을 시사

IV. Results and Discussion

- **Hyperparameters sensitivity**

  - MoE는 hyperparameter에 굉장히 민감함

  (1) **배치** 사이즈가 클수록 MoE expert 하나만 쓰는 편이 더 나음

  → 작은 배치 사이즈일수록 학습이 안정적

  (2) **Training step**도 5K를 넘어갈수록 성능이 악화

  (3) **Learning rate** 가 작을수록 더 안정화됨

V. Conclusion

- MoV, MoLORA 제안
- PEFT technique, 1%보다 더 적은 파라미터 업데이트만으로도 fine-tuning과 비교해 비슷한 성능
- Extensive analysis -> 우수한 성능 검증
- 다른 PEFT strategies와 결합 가능

[리뷰]

- Median을 main metric으로 활용, appendix에서 mean 성능 리포트 → 확연한 경향성 발견할 수 없음, full tuning보다는 전반적으로 낮은 성능 but (IA)^3, LoRA와 비교시 성능 향상 일관적
  → 논문은 서술 방식에 따라 포장 가능
- 개별 연구자 고려 최대 1%의 파라미터만 활용했다는 점에서 가치가 있다고 판단

Thank You
Q&A