A APPENDIX

A.1 EXTENDED RELATED WORK

Mixture-of-Experts The Mixture-of-Experts (MoE) has been investigated thoroughly in Natural Language Processing (Lou et al., 2022; Mustafa et al., 2022; Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022; Zoph et al., 2022; Clark et al., 2022; Zhou et al., 2022; Komatsuzaki et al., 2023; Kudugunta et al., 2021; Zuo et al., 2022) as an effective way of increasing the model's capacity in parameter size where certain parts of the model are activated while computation is kept the same or close to its dense counterpart. In the context of MoE, there is a body of work focusing on improving the routing (Hazimeh et al., 2021; Lewis et al., 2021; Roller et al., 2021; Zhou et al., 2022) including random routing (Zuo et al., 2022) activating all expert through weighted average (Eigen et al., 2014) to sparsely select a single or k experts (Fedus et al., 2022; Du et al., 2022). MoE has also been invested in multi-task settings including multilingual neural machine translation(Hazimeh et al., 2021; Kudugunta et al., 2021). Unlike these studies, our research addresses MoE by scaling both the volume of data and the number of tasks, aiming to mitigate the instability inherent in training the MoE models. But our primary emphasis remains on achieving efficient fine-tuning. Recently, Shen et al. (2023) highlighted how instruction fine-tuning with scaled tasks can counteract the generalization challenges tied to MoE models. In distinction from this, our study scrutinizes the efficacy of instruction fine-tuning in the MoE domain, specifically concentrating on a unique ensemble of the PEFT components, considering the memory cost of the traditional MoE can be prohibitive for many practitioners. Similar to the aforementioned work, Ye et al. (2022) utilized MoE in a multi-task context, employing BART Lewis et al. (2019) as their pre-trained model. However, they limited their experimental scope to a smaller scale and used replicas of each transformer layer as experts, simply multiplying the model by the number of experts. Our work, on the other hand, presents an extreme parameter efficiency with small experts at a large scale up to 11B parameter base model.

Instruction Tuning Instruction tuning, as elucidated in (Sanh et al., 2022; Wei et al., 2022; Mishra et al., 2022), is a technique where a language model is fine-tuned over a collection of tasks using paired prompts and responses. The primary goal of this technique is to enable the model to predict responses accurately based on the provided prompts, thereby augmenting its ability to understand and execute instructions effectively. The method has gained considerable attention due to its pronounced success in enhancing zero-shot performance on tasks to which the model has not been previously exposed. Additionally, instruction tuning has led to breakthroughs such as Chain of Thought Prompting (Wei et al., 2023) where a breakdown of complex problems into smaller steps to produce intermediate reasoning along with the final solution, PaLM (Chowdhery et al., 2022), FLAN (Wei et al., 2022). In our work, we explore the use of instruction fine-tuning with the intention of harnessing its benefits that enable the model to learn from a diverse set of inputs where the mixture of expert style models suits well, for enhanced evaluation performance on unseen tasks. Our objective remains to optimize computational efficiency without compromising zero-shot performance.

Parameter-Efficient Fine-tuning. Houlsby et al. (2019) established "adapters" in the NLP domain to fine-tune BERT. There are many variants of adapters with different design choices (Bapna et al., 2019; Pfeiffer et al., 2021). Li & Liang (2021) proposed updating soft prompts concatenated to embeddings or layer outputs instead of adapters. Zaken et al. (2022) show that just updating only a small subset of parameters during fine-tuning (e.g. just biases) is very effective. Hu et al. (2021) proposed LORA based on low-rank decomposition matrices of transformer layers. They show superior performance with a smaller parameter budget and no inference cost as LORA parameters can be applied offline to the baseline model. Liu et al. (2022) proposed $(IA)^3$, task-specific vectors to modify attention activation. Instead of using feedforward layers inserted in transformer layers as adapters, they learn vectors to update (by broadcast multiplication) key, value, and linear layer weight matrices. Unlike the other PEFT methods, $(IA)^3$ does not induce any additional inference cost and enables mix-batches (from different datasets). The multiplicative nature of the $(IA)^3$ creates an interesting opportunity for the mixture-of-expert type of modeling without parallelization overhead. Chen et al. (2023) experiment with different design spaces (essentially a hyperparameter search) for PEFT. They suggest four phases: 1) grouping layers into different sets; 2) adding trainable parameters towards each group; 3) deciding which group should be trained; 4) assigning groups with different training strategies. Their finding is that different architectures have different best settings. We have chosen $(IA)^3$ and

LORA as our PEFT components because they offer an optimal balance between performance and parameter efficiency (Mahabadi et al., 2021; Liu et al., 2022).

Several studies have explored PEFT in the context of MoE or in a similar fashion, albeit with certain distinctions. For instance, Wang et al. (2022) focused on single-task fine-tuning employing a mixture of adapters for BERT$_{base}$ with 110M parameters (Devlin et al., 2019) and $RoBERTa_{large}$ with 355M parameters (Liu et al., 2019), incorporating random routing, and adopting a few-shot evaluation. In divergence from this, our work centers on instruction-tuning with multiple tasks present during fine-tuning. We underscore the efficacy of this approach by rigorously testing up to 11B parameter text-to-text model Raffel et al. (2020), implementing token routing, and strictly emphasizing evaluation on a set of unseen (held-out) tasks to underscore the potential of instruction tuning. In another work, Ponti et al. (2022) introduced Polytropon, which involves learning adapters (termed as 'skills') specific to each task and employing a task-skills binary matrix to determine the skill set associated with each task. In their method, input examples dictate the selection of adapters. These adapters are then aggregated, and the resultant single adapter is integrated into the overall architecture. Extending upon the Polytropon framework, Caccia et al. (2023) implemented a distinct skill set for every layer in their variant named Polytropon-S. They introduce a deterministic routing function, delve into supplementary inductive biases, show effectiveness up to 3B models, and they don't employ MoE style architecture. Our research presents a departure from these two studies. Specifically, our primary experimental setup employs MoEs that do not require any specific task identifier during fine-tuning by the use of their token routing strategy. In this way, we can evaluate our instruction-tuned MoEs on unseen tasks without any further task-specific few-shot fine-tuning. We showed the scaling property of our MoEs in this setting by fine-tuning models up to 11B parameters.

A.2 Zero-Shot Evaluation for P3 dataset

In our study, we conducted a comprehensive evaluation of the variants of our proposed methods in comparison to our established baselines. This evaluation encompassed various sizes of the T5 model, specifically 770M, 3B, and 11B. These results are given in Table 2, 3 and 4. Both mean and median scores were reported for every evaluation set derived from the P3 dataset, which covers a range of tasks. For further details and a more in-depth exploration, please refer to the following URL: `https://huggingface.co/datasets/bigscience/P3`.

T5-Large (770M)

	Model	% Params.	Metric	ANLI	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
Full-FT	T0-770M (ours)	100%	median	35.6	71.43	75.63	57.21	51.41	77.0	53.04	26.78	56.01
			mean	35.57	57.74	75.88	52.31	52.52	74.6	52.93	26.74	53.54
PEFT	$(IA)^3$	0.036%	median	33.5	42.86	67.87	62.02	52.35	67.0	51.22	26.33	50.39
			mean	33.27	45.12	67.08	58.17	52.74	66.63	51.35	26.32	50.09
	LoRA	0.497%	median	35.0	55.36	57.4	63.46	50.24	77.0	53.28	26.67	52.3
			mean	35.26	51.67	59.35	62.98	50.66	76.5	52.41	27.24	52.0
Our Method	MOV-5	0.27%	median	33.6	41.07	71.48	61.54	50.86	76.5	51.46	26.02	51.57
			mean	33.51	42.62	71.26	60.96	51.14	73.8	51.55	26.01	51.36
	MoV-10	0.55%	median	33.9	42.86	74.19	62.5	50.31	77.0	52.64	26.34	52.47
			mean	33.68	42.38	74.51	59.23	50.74	74.82	52.2	26.72	51.78
	MoV-20	1.10%	median	33.7	41.07	73.83	63.46	50.94	75.46	51.14	25.48	51.89
			mean	33.98	45.12	73.36	59.13	51.33	73.47	51.3	25.45	51.64
	MoV-30	1.66%	median	33.75	41.07	72.92	55.77	51.25	77.0	51.46	26.55	51.22
			mean	33.81	44.88	72.56	56.15	51.29	77.43	51.81	26.52	51.81
	MoV-60	3.32%	median	34.0	53.57	75.81	57.69	50.55	77.96	53.12	26.33	53.63
			mean	34.24	52.26	75.02	58.37	50.78	77.06	52.87	26.74	53.42
	MoLoRA-10	5.60%	median	33.2	67.86	68.41	64.9	50.39	80.0	52.64	52.64	55.52
			mean	33.37	56.31	68.88	63.37	51.55	79.35	52.31	52.31	53.99

Table 2: Zero-shot evaluation of the 770M parameter model across all unseen tasks, comparing different numbers of experts for both MoV and MoLoRA.

A.3 Token vs. Sentence Embeddings for Routing

We present the mean and median results for our routing strategies in Table 5. Specifically, we assessed performance by either passing tokens directly to the router or by passing sentence embeddings. Our findings indicate that, particularly for the T5-XL (3B) model, token routing consistently yields

T5-XL (3B)

	Model	% Params.	Metric	ANLI	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
Full-FT	T0-3B (Sanh et al., 2022)	100%	median	33.46	50.0	64.08	64.42	50.39	74.92	50.51	27.51	51.91
			mean	33.42	45.36	64.55	65.10	50.69	72.40	50.97	27.29	51.22
	T0-3B (our replication)	100%	median	41.08	80.36	76.17	53.37	53.92	88.94	57.46	29.19	60.06
			mean	40.73	74.52	76.82	52.21	53.84	88.99	56.83	29.2	59.14
PEFT	$(IA)^3$	0.018%	median	34.08	50.0	66.43	56.25	55.41	79.08	52.09	29.91	52.90
			mean	34.56	51.07	68.38	54.9	55.61	78.23	52.14	28.97	52.98
	LoRA (rank 4)	0.3%	median	37.5	75.57	73.53	61.02	51.25	83.6	54.33	25.32	57.51
			mean	37.85	66.9	77.04	56.73	52.29	82.83	55.64	26.79	57.01
	LoRA (rank 8)	0.6%	median	37.5	75.0	77.98	62.5	51.49	83.67	55.72	27.3	58.89
			mean	37.64	64.05	77.91	56.71	51.77	82.84	55.23	26.83	57.03
	LoRA (rank 16)	1.2%	median	38.5	80.36	76.71	63.46	51.02	84.5	54.7	27.11	59.54
			mean	37.11	65.6	77.62	60.48	51.49	82.29	55.14	26.71	57.05
Our Method	MoV-2	0.18%	median	34.7	46.43	66.06	56.25	54.86	85.42	53.75	29.25	53.34
			mean	35.14	50.36	69.31	56.15	54.4	83.79	53.69	28.47	53.91
	MoV-5	0.23%	median	37.1	76.79	78.16	57.69	52.27	86.77	53.99	29.31	59.01
			mean	37.66	62.14	78.3	58.46	53.54	86.52	54.54	28.3	57.43
	MoV-10	0.32%	median	38.92	75.0	78.88	62.5	52.19	85.77	55.96	30.24	59.93
			mean	38.83	63.45	79.49	60.19	53.04	86.41	56.27	29.11	58.35
	MoV-20	0.50%	median	39.2	75.0	76.71	57.69	53.45	89.0	55.64	30.89	59.7
			mean	39.25	64.05	76.53	56.63	53.45	86.93	56.24	29.36	57.81
	MoV-30	0.68%	median	38.7	78.57	80.87	63.46	51.1	87.25	56.27	28.63	60.61
			mean	38.9	67.5	81.23	59.9	52.43	86.28	56.39	27.57	58.77
	MoV-60	1.22%	median	38.83	76.79	74.55	60.1	52.66	89.79	55.49	30.47	59.83
			mean	38.97	63.93	75.38	57.79	53.5	86.04	55.88	29.28	57.59
	MoV-10 (top-1)	0.32%	median	33.9	75.0	71.12	61.06	50.71	70.0	51.7	25.89	54.92
			mean	34.31	60.6	71.41	58.94	51.24	68.39	51.79	25.98	52.82
	MoV-10 (top-2)	0.32%	median	38.7	82.14	75.63	48.08	53.68	79.88	54.14	27.37	57.45
			mean	38.89	69.76	74.95	47.69	53.51	79.89	53.83	26.91	55.67
	MoLORA-2 (rank 4)	0.75%	median	39.2	82.14	80.32	62.5	50.39	80.58	57.38	28.47	60.12
			mean	38.86	65.71	80.0	60.0	50.8	82.17	56.51	28.03	57.76
	MoLORA-5 (rank 4)	1.66%	median	36.75	71.43	79.96	56.25	55.17	85.81	55.8	27.63	58.6
			mean	37.52	62.14	80.22	52.6	55.34	84.05	56.04	26.62	56.82
	MoLORA-10 (rank 4)	3.18%	median	38.5	78.57	78.16	63.46	50.86	86.5	55.41	26.72	59.77
			mean	38.49	66.43	77.44	59.9	51.63	84.96	56.1	26.7	57.71
	MoLORA-15 (rank 4)	4.69%	median	40.0	80.36	80.51	62.98	50.86	89.0	55.33	27.3	60.79
			mean	39.73	69.52	80.97	60.67	51.54	86.5	55.03	27.25	58.9

Table 3: In our most comprehensive experimental setup, we conducted a zero-shot evaluation across all unseen tasks using a 3B parameter model. We compared varying numbers of experts for both MoV and MoLoRA and experimented with a top-k selection routing strategy

T5-XXL (11B)

	Model	% Params.	Metric	ANLI	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
Full-FT	T0-11B (Sanh et al., 2022)	100%	median	42.17	78.57	81.23	64.42	57.21	90.79	60.46	33.65	63.56
			mean	41.16	70.12	80.83	61.45	56.58	90.02	59.94	33.58	61.70
	T0-11B (our replication)	100%	median	47.1	80.36	81.41	60.1	56.27	96.08	67.32	31.61	65.03
			mean	45.83	72.62	81.52	58.17	56.66	96.0	66.77	30.95	63.57
PEFT	$(IA)^3$	0.0098%	median	42.3	73.21	75.99	58.65	52.04	86.27	54.3	30.27	59.12
			mean	42.1	63.27	75.31	55.49	52.27	85.74	55.06	30.09	57.41
Our Method	MoV-10	0.143%	median	45.83	76.79	78.52	53.85	51.88	94.23	63.77	33.5	62.3
			mean	44.73	70.12	78.88	54.23	53.26	93.64	63.57	33.59	61.5
	MoV-20	0.287%	median	44.58	76.79	73.83	55.77	52.98	95.0	62.27	32.92	61.77
			mean	43.54	69.17	74.4	52.88	54.5	93.93	62.95	32.85	60.53
	MoV-30	0.431%	median	43.6	76.79	77.62	56.73	53.84	93.62	64.25	31.34	62.22
			mean	43.32	69.29	77.22	53.56	56.03	93.65	63.52	31.32	60.99
	MoV-60	0.862%	median	45.17	75.0	83.03	60.1	53.68	95.42	65.82	34.38	64.08
			mean	43.9	69.88	83.07	56.54	54.51	94.01	64.56	34.17	62.58

Table 4: We evaluated the largest available model size from the original T5 pre-trained checkpoint, T5-XXL with 11B parameters, to demonstrate the efficacy of our proposed mixture of PEFT experts at this scale.

better performance in terms of both mean and median values. The Anli dataset is excluded from our embedding dataset.

A.4 Fine-tuning Efficiency: MoV vs $(IA)^3$

Table 6 shows the updated parameter count and training time ratios for Mixture of Vectors (MoV) and $(IA)^3$ with respect to full-fine tuning. These metrics confirm that our MoV achieves a better performance-efficiency trade-off compared to $(IA)^3$.

MoV – Token vs. Sentence Embedding

Model	Metric	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
MoV-10 (Token) - 770M	median	42.86	74.19	62.5	52.64	52.64	77.0	26.34	55.12
	mean	42.38	74.51	59.23	52.2	52.2	74.82	26.72	54.37
MoV-10 (Embedding) - 770M	median	48.21	67.15	62.98	51.8	50.99	67.0	26.38	53.5
	mean	51.67	67.29	58.37	51.79	50.99	65.8	26.57	53.21
MoV-10 (Token) - 3B	median	75.0	78.8	62.5	52.19	55.96	85.77	30.24	62.94
	mean	63.45	79.49	60.19	53.04	56.27	86.41	29.11	61.14
MoV-10 (Embedding) - 3B	median	57.14	67.15	61.06	55.33	52.49	82.5	29.08	57.82
	mean	51.07	68.81	58.65	55.28	52.57	80.53	28.51	56.49
MoV-10 (Token) - 11B	median	76.79	78.52	53.85	51.88	63.77	94.23	33.5	64.65
	mean	70.12	78.88	54.23	53.26	63.57	93.64	33.59	63.9
MoV-10 (Embedding) - 11B	median	75.0	78.7	57.69	54.0	57.85	92.0	33.08	64.05
	mean	66.19	79.1	58.37	54.83	58.78	91.17	32.7	63.02

Table 5: The above results demonstrate the effectiveness of token routing in comparison to imposing a strong inductive bias, such as sentence embedding across various model parameters.

Base model size	Model	% Params. updated ↓	% Training time reduction ↑	Average zero-shot performance ↑
3B	$(IA)^3$	0.018%	38%	52.90
	MoV (10 Experts)	0.32%	31%	59.93
	MoV (30 Experts)	0.68%	27%	60.61
11B	$(IA)^3$	0.0098%	36%	59.12
	MoV (10 Experts)	0.143%	27%	62.30
	MoV (30 Experts)	0.431%	23%	62.22

Table 6: Fine-tuning efficiency metrics for Mixture of Vectors (MoV) and its dense PEFT counterpart, $(IA)^3$. We compare the ratio of updated parameters and the training times with respect to full fine-tuning (with the same batch size), across different scales. MoV exhibits a marginal difference in training time compared to $(IA)^3$ while demonstrating significant improvement in zero-shot performance, achieving a better performance-efficiency trade-off.

A.5 MoLORA Implementation

```
class MoLORA_Layer(nn.module):
  n_experts: int # number of experts
  rank: int # low-rank dimension
  h_out: int # output dimension

  def call(self, inputs):
    # inputs shape: [batch, seq, h_dim]
    batch, seq, h_dim = inputs.shape

    # MoLORA A: [n_experts, h_dim, rank]
    molora_A = self.param('molora_A',
      nn.init.normal(), (self.n_experts, h_dim, self.rank))

    # MoLORA B: [n_experts, rank, h_out]
    molora_B = self.param('molora_B',
      nn.init.zeros(), (self.n_experts, self.rank, self.h_out))

    # Ax: [batch, seq, n_experts, rank]
    molora_Ax = jnp.einsum('...d,edr->...er',
                           inputs,
                           molora_A)

    # BAx: [batch, seq, n_experts, h_out]
    molora_BAx = jnp.einsum('...er,ero->...eo',
                           molora_Ax,
                           molora_B)

    # router probs: [batch, seq, n_experts]
    router_probs = self.router(inputs,
      self.n_experts, dtype='float32')

    # combined LoRAs' outputs: [batch, seq, h_out]
    molora_combine = jnp.einsum('...e,...eo->...o',
                           router_probs,
                           molora_BAx)

    return molora_combine
```