Pseudo Lab: AI in Logistics & Transportation

MVMoE: Multi-Task Vehicle Routing Solver with Mixture-of-Experts

Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, Chi Xu, ICML, 2024.

May 13, 2025

Seungmin Jeon

simonjeon825@gmail.com

Content

Introduction

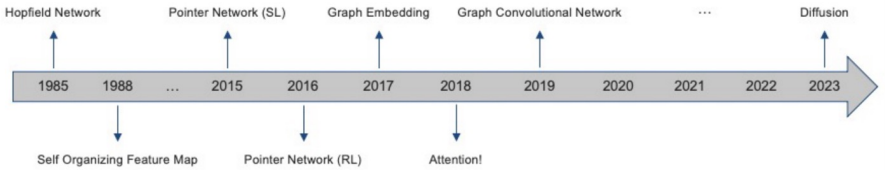
Introduction

- Vehicle Routing Problem (VRP)는 전통적인 combinatorial optimization problem (COP)
- NP-hard인 VRP를 Exact Solver로 풀기에 많은 비용이 필요하며, Heuristic Solver로 Suboptimal 결과를 합리적인 시간에 얻을 수 있게 해주나 이를 설계를 위하여 상당한 도메인 전문성이 필요
- 계산 비용과 도메인 전문성이 덜 필요한 Learning-based가 최근 많은 관심을 받고 있으며, Reinforcement Learning (RL) 같은 여러 Training Paradigm 기반해 Policy를 설계
- 허나 Neural Solver는 Network Structure을 잘 맞추어야 하고, 특정 VRP에 맞추어 Training을 수행
- 해당 논문은 VRP 변형들을 대응하고, Unsee VRP에 대해 Zero-shot 능력이 있는 하나의 Neural Solver ‘Multi-task VRP Solver with Mixture-of-experts (MVMoE)’를 제안

Contribution

- 여러 VRP를 해결하기 위한 MVMoE을 제안하며, 이는 COP에서 처음으로 MoE를 사용한 것
- Hierarchical Gating Mechanism을 통하여 성능과 연산의 균형을 맞춤
- 10개의 Unseen VRP에 대해 Zero-shot, Real World 문제에서 Few-shot을 통해 적절한 성능을 보임

Related Work & Preliminaries: Neural VRP Solvers



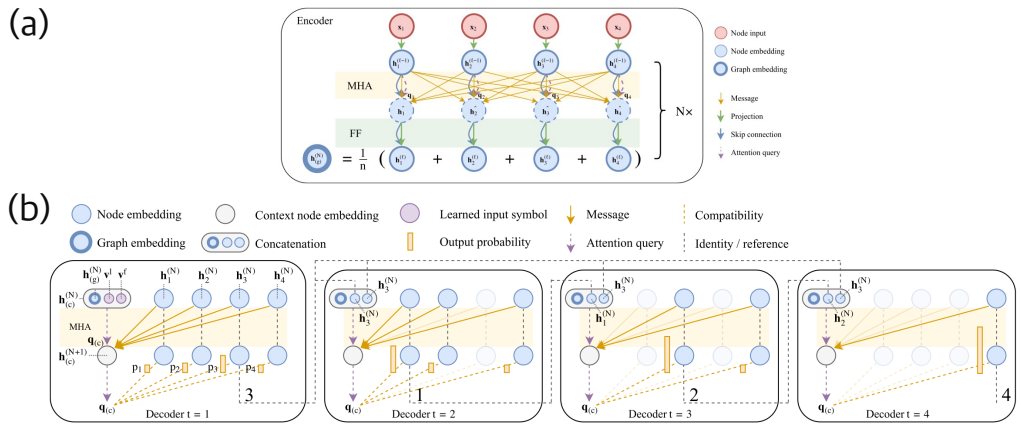
- VRP를 해결하기 위한 Neural Solver는 크게 Construction-based Solver와 Improvement-based Solver로 분류

1) Construction-based Solver

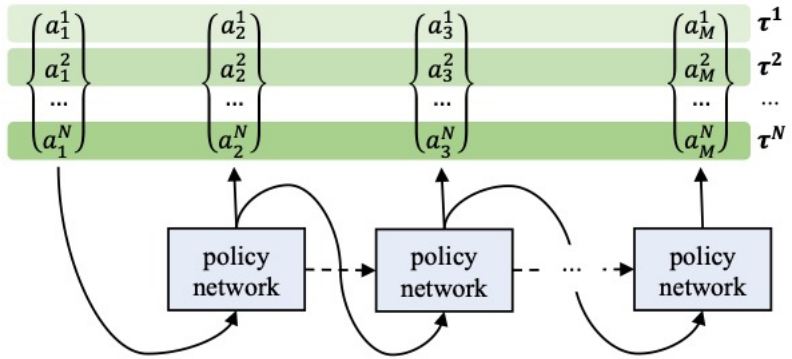
- End-to-end로 Solution을 구성하는 (Construct) Policy를 학습
- Attention-based Model (AM) [1]과 Policy Optimization with Multiple Optima (POMO) [2] 가 대표적인 방법론이며, 많은 연구들이 이를 기반

2) Improvement-based Solver

- 종료 조건을 만족하기까지 Initial Solution을 Iteratively하게 개선
- 일반적으로 Construction-based가 효율적이고 요구되는 성능을 충족하나, Inference Time이 충분하면 Improvement-based가 더 좋은 성능을 보일 가능성이 있음



AM ((a): encoder, (b): decoder) [1]



POMO [2]

[1] Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In ICLR, 2018.
[2] Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Gwon, Y., and Min, S. POMO: Policy optimization with multiple optima for reinforcement learning. In NeurIPS, volume 33, pp. 21188-21198, 2020.

Related Work & Preliminaries : Mixture-of-experts (MoE)

- MoE는 Neural Network 내에서 입력 샘플마다 다른 하위 Network(Expert)를 활성화하는 방식으로 동작하는 기법
- 이는 모든 샘플에 동일한 Model Parameter를 사용하는 전통적인 Network과 달리, 특정 입력에 대해 일부 Expert만 선택적으로 활성화되므로, Model의 총 Parameter 수는 증가하지만 연산량은 상대적으로 일정하게 유지 [1]

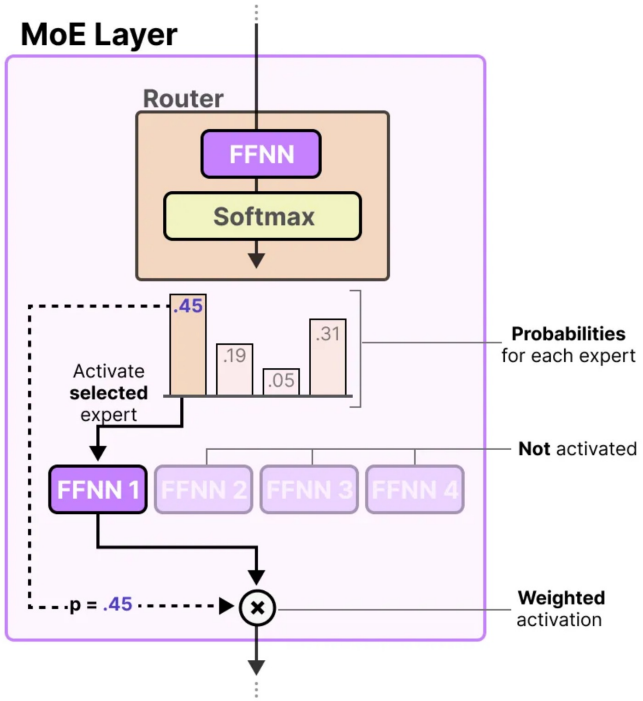
- 핵심 구성 요소는 Expert Network와 Gate Network
- Expert Network
  - 입력 데이터를 기반으로 최적의 Expert를 선택하는 역할
  - 일반적으로 Softmax를 사용하여 각 Expert에 대한 확률을 계산하며, 특정 개수(top-k)의 Expert 선택하여 가중합을 수행
- Gate Network
  - 일반적으로 MLP 형태를 가지며, 입력을 받아 처리하는 하위 Network이다.
  - 각 Expert가 다른 Data Pattern을 학습하면서 역할을 분리하는 것이 이상적

- DeepSeek-V2, Llama 4 등에 적용되어 기존 Dense Model 대비 연산 효율성 증진

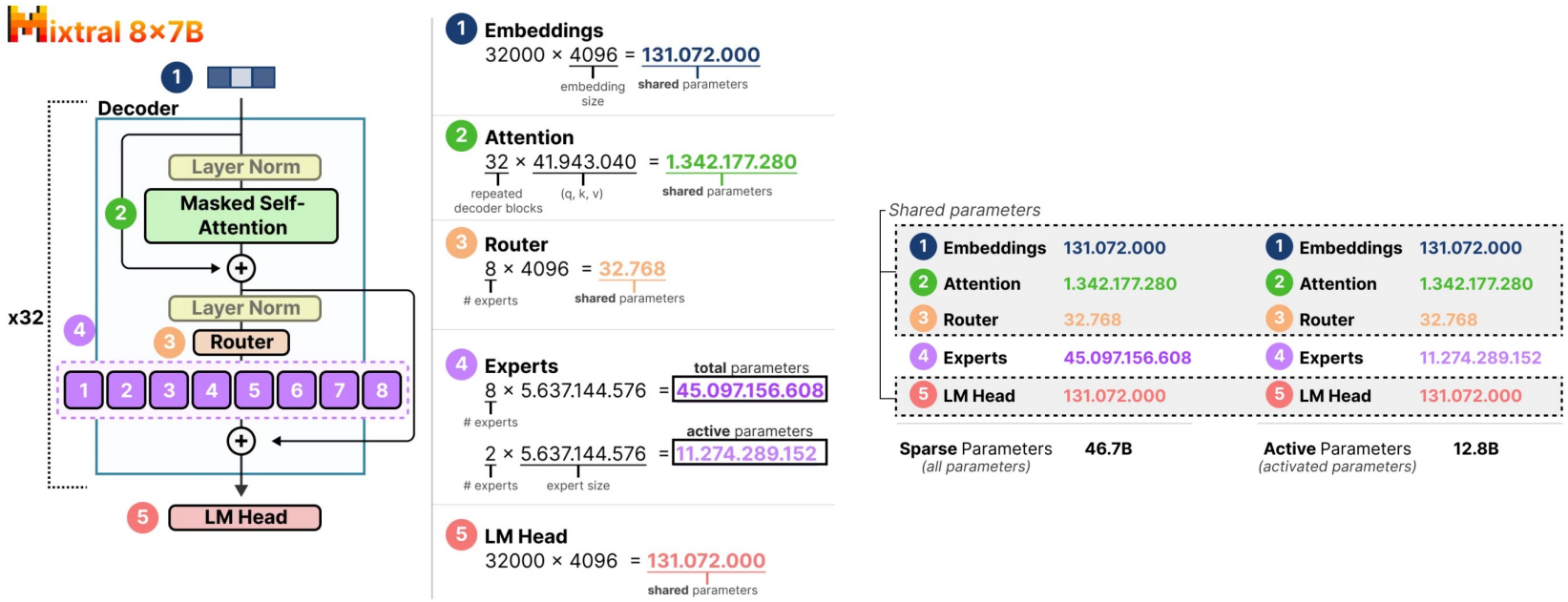
[1] “MoE: Mixture of Experts (전문가 혼합),” 인공지능(AI) & 머신러닝(ML) 사전. [Online]. Available: https://wikidocs.net/275230. [Accessed: May. 13, 2025].

Related Work & Preliminaries : Mixture-of-experts (MoE)

- Gate Network를 통과하여 확률에 따라 Expert를 선택
- Model의 Parameter와 실제 Inference 시 Activate Parameter가 차이가 있음



MoE Layer Routing Example [1]



Mixtral 8X7B Parameters Example [1]

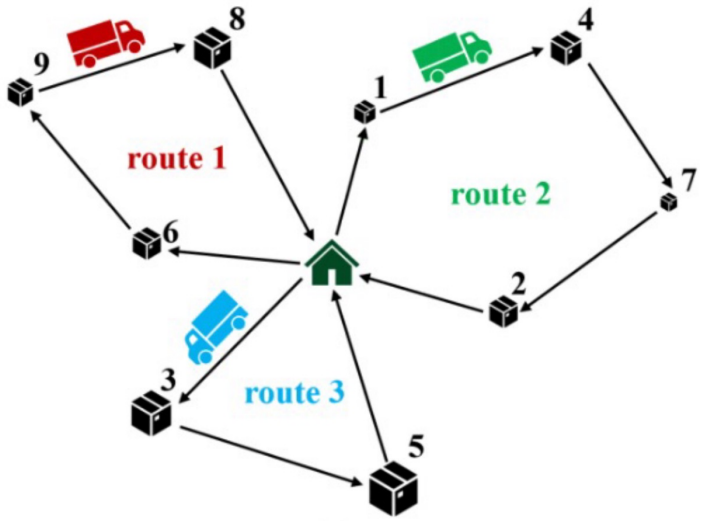
[1] “A Visual Guide to Mixture of Experts (MoE),” Exploring Language Models. [Online]. Available: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mixture-of-experts. [Accessed: May. 13, 2025].

Related Work & Preliminaries : CVRP

- Capacitated Vehicle Routing Problem (CVRP)은 크기가 n인 Graph로 정의할 수 있음

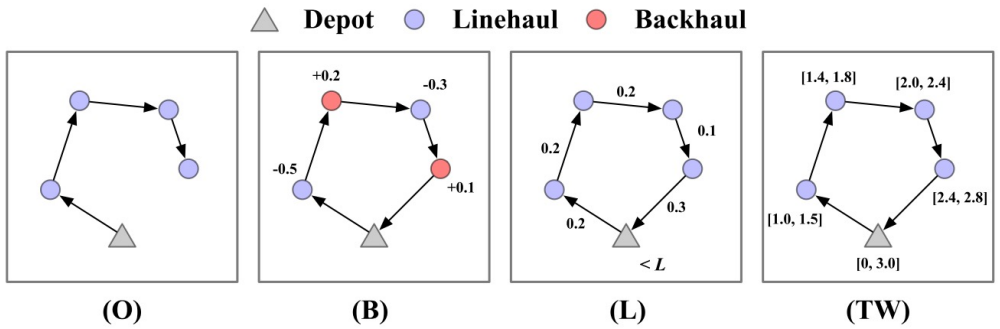
$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$ v_0: depot node / $\{v_i\}_{i=1}^{n}$: customer nodes

- 각각의 Customer Node는 Demand가 있으며, Capacity Limit (Q)는 차량에 설정
- Solution (Tour)는 Node Sequence이며, Sub-tour는 Depot에서 시작하여 Customer를 방문한 뒤 다시 Depot으로 복귀
- Customer Node는 한번만 방문해야 하며, Total Demand는 Capacity Limit을 넘기지 않아야 함 (Constraint)
- Euclidean Space에서 Tour의 Total Length를 최소화하는 Optimal Tour를 찾는 것이 목표 (Objective)



Related Work & Preliminaries : VRP Variants

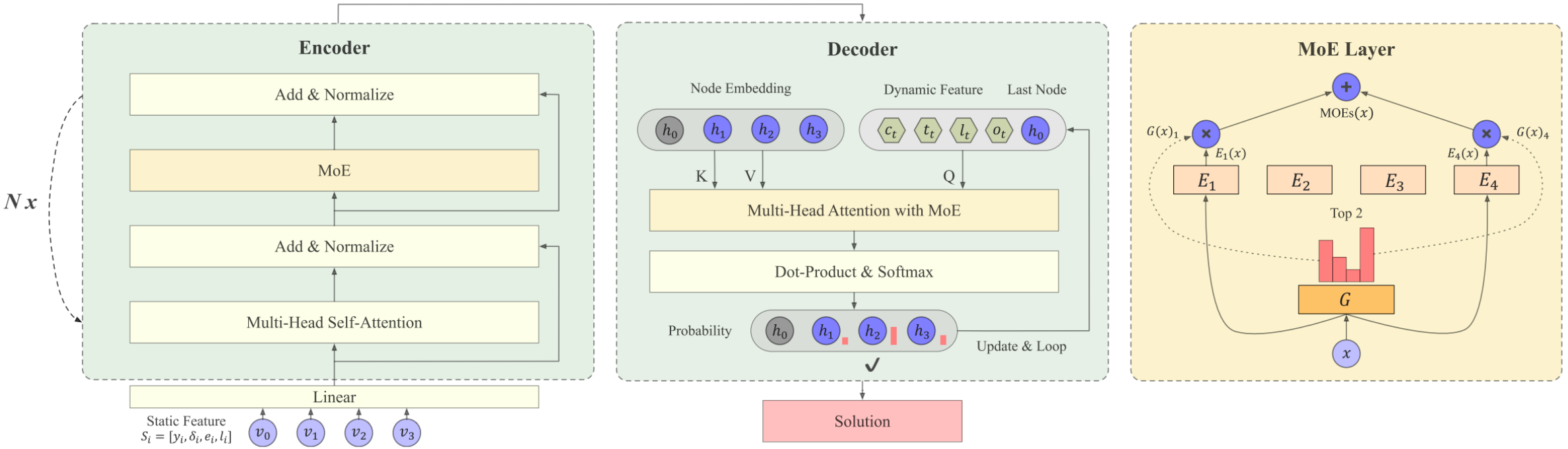
- CVRP에 현실적인 Constraint가 더해질 수 있음
  - Open Route (O): 차량이 Depot으로 돌아가지 않아도 됨
  - Backhaul (B): Demand는 양이나, 물건을 차에 싣는 등의 음의 Demand가 존재할 수 있음
  - Duration Limit (L): 사전에 정의된 것으로, 각각의 Route에 Cost (ex. Length) 제한이 있음
  - Time Window (TW): 각각의 Node에 Time Window가 있어, 차량은 그 사이에 가야 함
- 이를 조합하면 16가지 VRP Variants를 만들 수 있음



	Capacity (C)	Open Route (O)	Backhaul (B)	Duration Limit (L)	Time Window (TW)
CVRP	✓				
OVRP	✓	✓			
VRPB	✓		✓		
VRPL	✓			✓	
VRPTW	✓				✓
OVRPTW	✓	✓			✓
OVRPB	✓	✓	✓		
OVRPL	✓	✓		✓	
VRPBL	✓		✓	✓	
VRPBTW	✓		✓		✓
VRPLTW	✓			✓	✓
OVRPBL	✓	✓	✓	✓	
OVRPBTW	✓	✓	✓		✓
OVRPLTW	✓	✓		✓	✓
VRPBLTW	✓		✓	✓	✓
OVRPBLTW	✓	✓	✓	✓	✓

Method: Multi-task VRP Solver with MoEs

- Attention-based Nueral Network 기반하여 Construction-based neural solver 학습하는 걸 목표
  - RL을 기반하여 Policy를 학습하며, Autoregressive를 통해 Solution을 생성
  - Solution Construction Process는 Markov Decision Process (MDP)
  - Encoder는 Input Instance가 주어졌을 때, 모든 Node Embedding
  - Decoder는 선택해야 하는 Node의 확률을 Output



Method: Multi-task VRP Solver with MoEs - (1) Multi-task VRP Solver

- i 번째 Node의 Static Feature: Coordinate, demand, Start Time, End Time
- t 번째 Decoding Step의 Dynamic Feature: Remaining Capacity, Current Time, Length of Current Partial Route, Presence Indicator of the Open Route

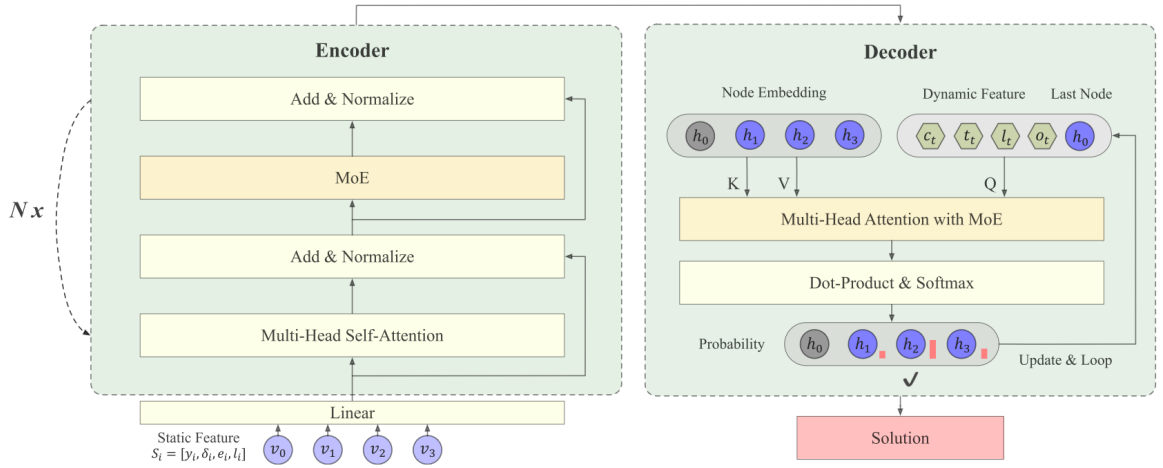
$$S_i = [y_i, \delta_i, e_i, l_i] \quad D_t = [c_t, t_t, l_t, o_t]$$

- 만약 주어진 VRP에 Feature가 없다면 Padding (CVRP 예시)

$$S_i = [y_i, \delta_i, 0, 0], D_t = [c_t, 0, l_t, 0]$$

- 모든 VRP variant에 존재하는 Feature를 Union Set으로 정의하고 공유하며, 이를 통해 일부 VRP variant만 학습하여도, Unseen Variant를 풀 수 있음

- Encoder는 Static Node Feature을 Input으로 받고, Node Embedding (h)를 Output
- Decoder는 Node Embedding과 Context Representation (마지막 선택된 Node Embedding, Dynamic Feature)을 Input으로 받고, Node의 Probability Distribution을 Output



Method: Multi-task VRP Solver with MoEs - (2) Mixture-of-experts

- Linear Layer 또는 FFN인 m 개의 Expert 존재

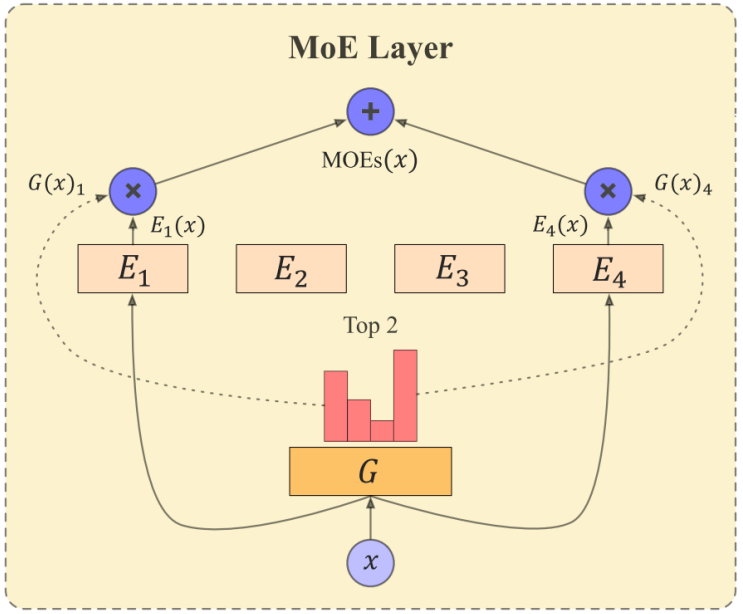
$$\{E_1, E_2, \ldots, E_m\}$$

- W_G Parameter로 이뤄진 Gating Network ($G(x)$)가 존재하며 이는 Input이 어떠한 Expert로 분배될지 결정
  - TopK Operator는 K-largest 값만 유지하고 나머지는 Negative Infinity 처리
  - Gating Network의 Output은 Softmax

$$G(x) = Softmax(TopK(x \cdot W_G))$$

- MoE Layer의 Output

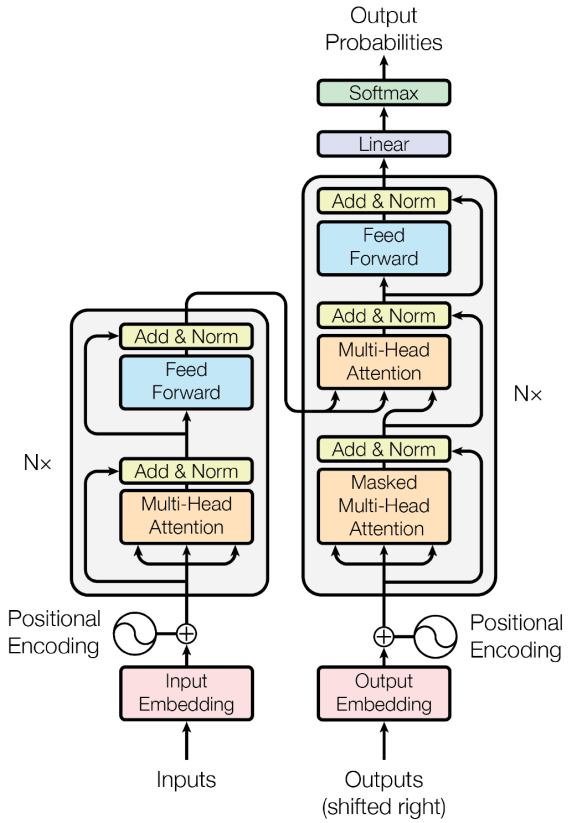
$$\mathrm{MoE}(x) = \sum_{j=1}^{m} G(x)_j E_j(x).$$

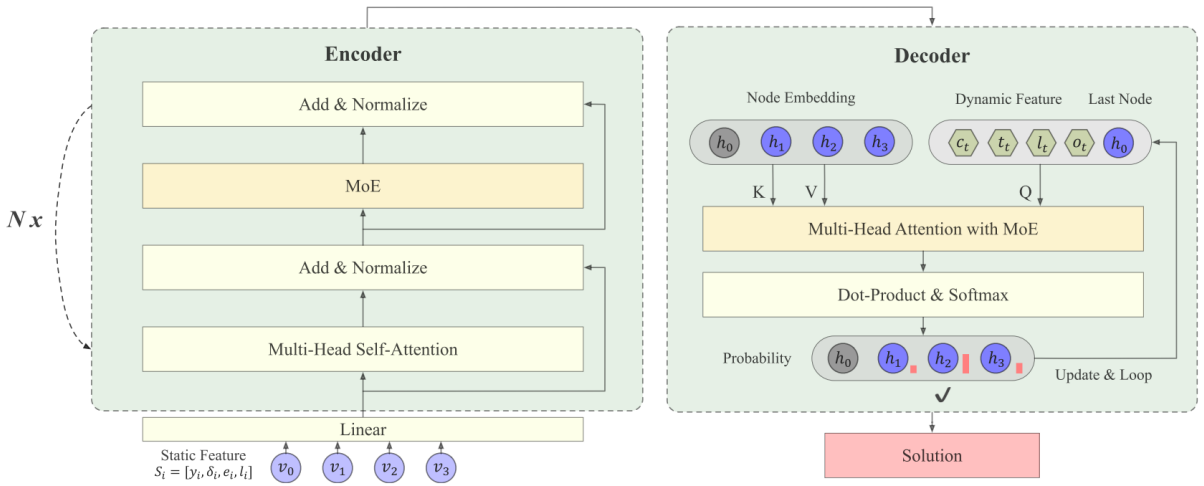


Method: Multi-task VRP Solver with MoEs - (3) MVMoE

- MoE는 Encoder의 FFN Layer, Decoder의 Final Linear Layer of Multi-head Attention을 대체
- Loss Function
  - $\mathcal{L}_a$: RL 기반 VRP Solver의 Loss Function (REINFORCE 기반 학습을 수행하며 Estimated Gradient 사용)
  - $\mathcal{L}_b$: MoE의 Imbalance을 방지 (Load Balacing)를 위한 Loss로, α는 Hyperparameter

$$\min_{\Theta} \mathcal{L} = \mathcal{L}_a + \alpha \mathcal{L}_b, \qquad \nabla_\theta \mathcal{L}_a(\theta|\mathcal{G}) = \mathbb{E}_{p_\theta(\tau|\mathcal{G})}[(c(\tau) - b(\mathcal{G}))\nabla_\theta \log p_\theta(\tau|\mathcal{G})].$$



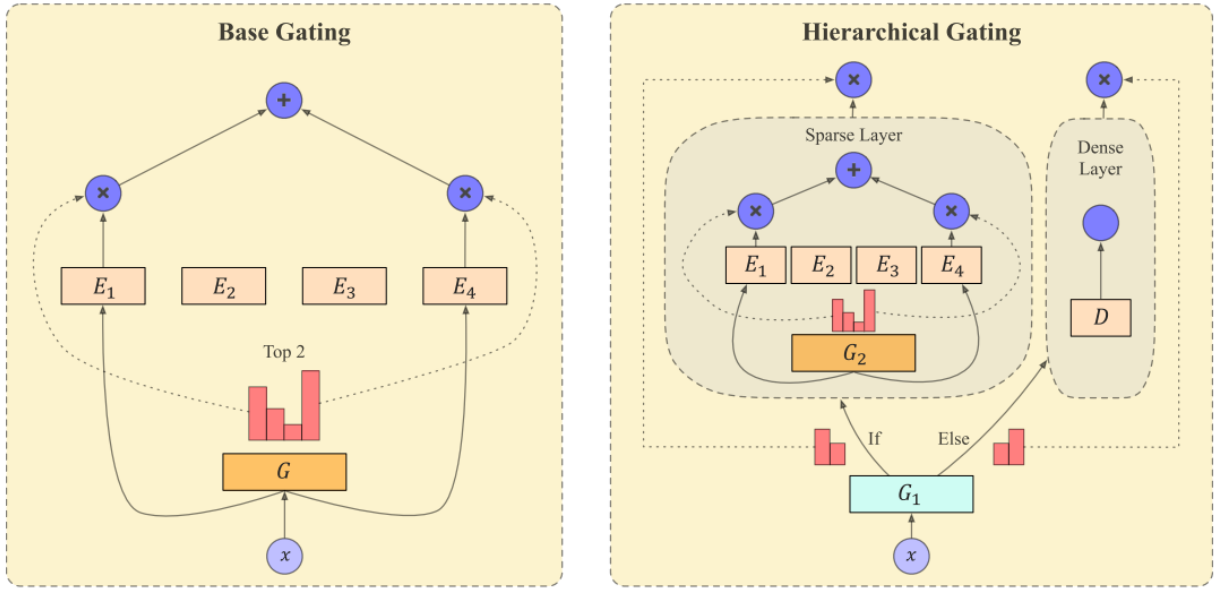


Method: Gating Mechanism

- 개별 Node가 어떤 Expert로 가야 하는지, Node-level (Gate-level) Gating을 수행
- MoE Layer에서 Gating Network를 전달하는 것과 선택된 Expert에 대한 Node Distribution을 계산하는 추가 연산 발생
- 이 때 Decoder에서 MoE를 사용하면, Problem Size (n)을 증가시키면 Decoding Step (T) 또한 커지고 MoE를 반복 수행하면서 연산 문제가 발생
- 그러므로 해당 논문은 Hierarchical Gating Mechanism을 적용

	E_1	E_2	E_3	E_4
x_1	2.31	0.45	0.74	-0.53
x_2	0.65	1.43	-0.68	0.76
x_3	1.20	-1.02	0.98	1.97
x_4	-0.35	1.26	0.87	0.36

Score Matrix



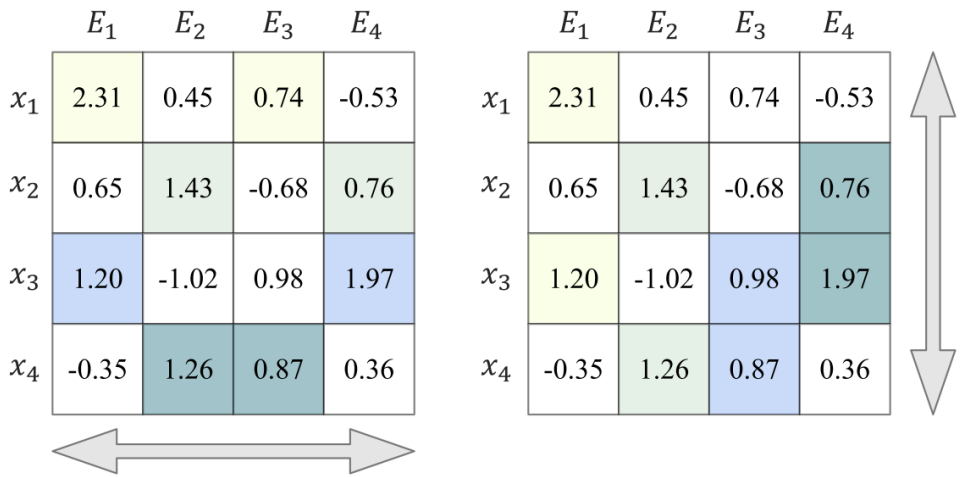
Hierarchical Gating

Method: Gating Mechanism - (1) Node-level Gating

- Node (x)와 Expert (E) 가 존재할 때, Score Matrix (H)를 활용하여 Expert를 선택할 수 있음

$$H = (X \cdot W_G)$$

- Gating Algorithm은 input-choice와 expert-choice가 대표적이며, 이외 random-choice, soft MoE 등이 있음 (해당 논문은 input-choice 사용)
- Input-choice
  - 각각의 Node에서 Top K Expert를 선택하며, K는 Computational Complexity를 고려하여 선택 (ex. 1 or 2)
  - Load balancing 을 보장하지 못하여 Expert 불균형이 생길 수 있고, 이를 해결하고자 Auxiliary Loss 사용
- Expert-choice
  - 각각의 Expert에서 Top K Node를 선택
  - 이를 활용하여 Load Balancing을 보장하며, 일부 Node가 선택되지 않을 수 있음



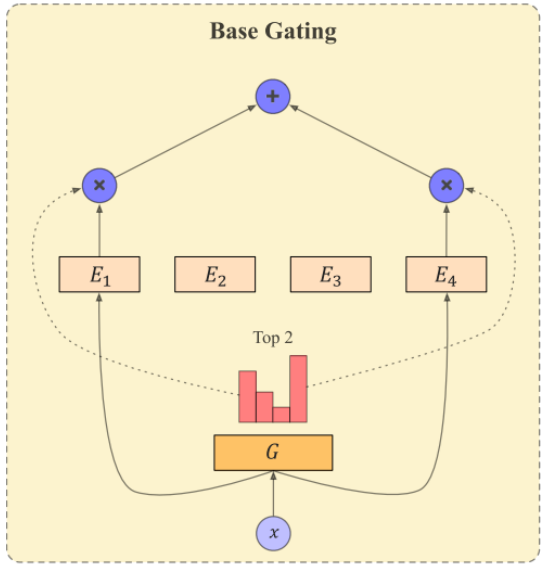
Left: Input-choice / Right: Expert-choice

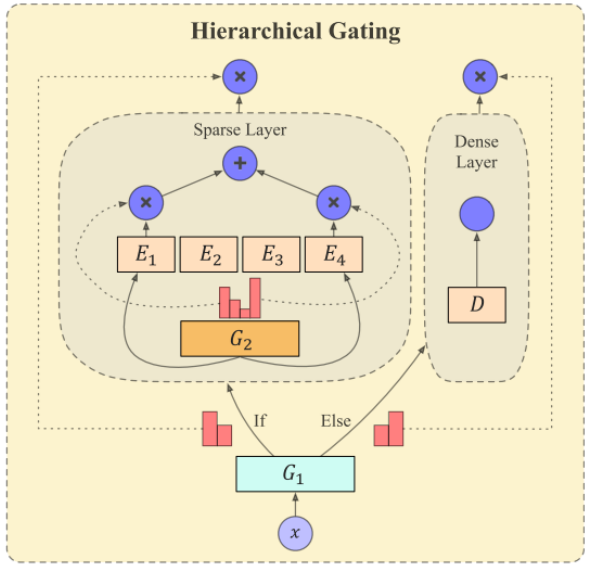
Method: Gating Mechanism - (2) Hierarchical Gating

- Decoding 과정에서 MoE 연산 비용을 해결하고자, Hierarchical Gating을 통해 부분적인 Step에만 MoE를 적용
  - Gating Network (G1, G2), Experts, Dense Layer (D) 로 구성
- Input이 주어졌을 때, 2개의 Stage로 구성
- G1은 Sparse 또는 Dense Layer로 갈지 결정
- Sparse로 가면 Node-level Gating을 기반하여 MoE Layer Output, Dense로 가면 Linear Layer 기반 Output 계산

- Hierarchical Gating은 약간의 성능 손실이 있으나, 효율에 대한 개선이 있음
- 효율과 성능의 균형을 위하여 Encoder는 Base Gating, Decoder는 Hierarchical Gating 사용

$$Hierarchical\_MoE(x) = \begin{cases} G_1(x)_0 \sum_{j=1}^{m} G_2(x)_j E_j(x) \\ G_1(x)_1 D(x) \end{cases}$$





Experimental Evaluation

Baselines

- Traditional Solver
  - HGS (CVRP, VRPTW)
  - LKH3 (CVRP, OVRP, VRPL, VRPTW)
  - OR-Tools (All VRP): n=50 / 100에 대해 time limit을 20/40 sec으로 준 것과, 200/400 sec으로 준 X10 가 존재
- Neural Solver
  - POMO: 개별 VRP으로 학습 수행
  - POMO-MTL: Multiple VRP 학습 (Parameter가 1.25M이고 MVMoE가 3.68M 이나, Inference 시 같은 숫자가 Activate)
- MVMoE: Expert m =4 / K = 2
  - MVMoE/4E: Input-choice gating을 Encoder, Decoder 둘 다 사용
  - MVMoE/4E-L: Hierarchical Gating을 Decoder에 사용

Training

- ADAM Optimizer를 사용
- 5000 Epoch이며, 각각 20000 Training Instance
- Multi-task Solver에 CVRP, OVRP, VRPB, VRPL, VRPTW, OVRPTW 중 각각의 Batch에서 Random하게 선택

Experimental Evaluation: Main Result - Performance on Trained VRPs

- 1k Test Instance를 수행하며 이동 거리 (Objective), Traditional Solver와 차이 (Gap), 총 걸린 시간 (Time) 으로 평가
- 학습에 사용된 6개의 VRP에 대해 평가
- POMO는 개별 문제에 대해 Multi-task Solver보다 성능이 좋음
- MVMoE는 POMO-MTL보다 성능이 좋음
- 4E-L이 계산 효율은 더 좋으나, 성능이 떨어짐

Table 1. Performance on 1K test instances of trained VRPs. * represents 0.000%, with which the gaps are computed.

	Method	$n = 50$ Obj.	$n = 50$ Gap	$n = 50$ Time	$n = 100$ Obj.	$n = 100$ Gap	$n = 100$ Time
CVRP	HGS	10.334	*	4.6m	15.504	*	9.1m
	LKH3	10.346	0.115%	9.9m	15.590	0.556%	18.0m
	OR-Tools	10.540	1.962%	10.4m	16.381	5.652%	20.8m
	OR-Tools (x10)	10.418	0.788%	1.7h	15.935	2.751%	3.5h
	POMO	10.418	0.806%	3s	15.734	1.488%	9s
	POMO-MTL	10.437	0.987%	3s	15.790	1.846%	9s
	MVMoE/4E	**10.428**	**0.896%**	4s	**15.760**	**1.653%**	11s
	MVMoE/4E-L	10.434	0.955%	4s	15.771	1.728%	10s
OVRP	LKH3	6.511	0.198%	4.5m	9.828	*	5.3m
	OR-Tools	6.531	0.495%	10.4m	10.010	1.806%	20.8m
	OR-Tools (x10)	6.498	*	1.7h	9.842	0.122%	3.5h
	POMO	6.609	1.685%	2s	10.044	2.192%	8s
	POMO-MTL	6.671	2.634%	2s	10.169	3.458%	8s
	MVMoE/4E	**6.655**	**2.402%**	3s	**10.138**	**3.136%**	10s
	MVMoE/4E-L	6.665	2.548%	3s	10.145	3.214%	9s
VRPB	OR-Tools	8.127	0.989%	10.4m	12.185	2.594%	20.8m
	OR-Tools (x10)	8.046	*	1.7h	11.878	*	3.5h
	POMO	8.149	1.276%	2s	11.993	0.995%	7s
	POMO-MTL	8.182	1.684%	2s	12.072	1.674%	7s
	MVMoE/4E	**8.170**	**1.540%**	3s	**12.027**	**1.285%**	9s
	MVMoE/4E-L	8.176	1.605%	3s	12.036	1.368%	8s

	Method	$n = 50$ Obj.	$n = 50$ Gap	$n = 50$ Time	$n = 100$ Obj.	$n = 100$ Gap	$n = 100$ Time
VRPTW	HGS	14.509	*	8.4m	24.339	*	19.6m
	LKH3	14.607	0.664%	5.5m	24.721	1.584%	7.8m
	OR-Tools	14.915	2.694%	10.4m	25.894	6.297%	20.8m
	OR-Tools (x10)	14.665	1.011%	1.7h	25.212	3.482%	3.5h
	POMO	14.940	2.990%	3s	25.367	4.307%	11s
	POMO-MTL	15.032	3.637%	3s	25.610	5.313%	11s
	MVMoE/4E	**14.999**	**3.410%**	4s	**25.512**	**4.903%**	12s
	MVMoE/4E-L	15.013	3.500%	3s	25.519	4.927%	11s
VRPL	LKH3	10.571	0.790%	7.8m	15.771	*	16.0m
	OR-Tools	10.677	1.746%	10.4m	16.496	4.587%	20.8m
	OR-Tools (x10)	10.495	*	1.7h	16.004	1.444%	3.5h
	POMO	10.491	-0.008%	2s	15.785	0.093%	9s
	POMO-MTL	10.513	0.201%	2s	15.846	0.479%	9s
	MVMoE/4E	**10.501**	**0.092%**	3s	**15.812**	**0.261%**	10s
	MVMoE/4E-L	10.506	0.131%	3s	15.821	0.323%	10s
OVRPTW	OR-Tools	8.737	0.592%	10.4m	14.635	1.756%	20.8m
	OR-Tools (x10)	8.683	*	1.7h	14.380	*	3.5h
	POMO	8.891	2.377%	3s	14.728	2.467%	10s
	POMO-MTL	8.987	3.470%	3s	15.008	4.411%	10s
	MVMoE/4E	**8.964**	**3.210%**	4s	**14.927**	**3.852%**	11s
	MVMoE/4E-L	8.974	3.322%	4s	14.940	3.941%	10s

Experimental Evaluation: Main Result - Generalization on Unseen VRPs

- Zero-shot: Unseen 10 VRP에서 MVMoE가 POMO보다 좋은 성능을 보임
- Few-shot: Target VRP (VRPBLTW, OVRPBLTW)에 대해 10k (전체 Training Instance의 0.01 %) 으로 학습을 하였으며, POMO-MTL보다 좋은 성능을 보임

Table 2. Zero-shot generalization on 1K test instances of unseen VRPs. * represents 0.000%, with which the gaps are computed.

	Method	$n = 50$ Obj.	$n = 50$ Gap	$n = 50$ Time	$n = 100$ Obj.	$n = 100$ Gap	$n = 100$ Time
OVRPB	OR-Tools	5.764	0.332%	10.4m	8.522	1.852%	20.8m
	OR-Tools (x10)	5.745	*	1.7h	8.365	*	3.5h
	POMO-MTL	6.116	6.430%	2s	8.979	7.335%	8s
	MVMoE/4E	**6.092**	**5.999%**	3s	**8.959**	**7.088%**	9s
	MVMoE/4E-L	6.122	6.522%	3s	8.972	7.243%	9s
VRPBL	OR-Tools	8.131	1.254%	10.4m	12.095	2.586%	20.8m
	OR-Tools (x10)	8.029	*	1.7h	11.790	*	3.5h
	POMO-MTL	8.188	1.971%	2s	11.998	1.793%	8s
	MVMoE/4E	**8.172**	**1.776%**	3s	**11.945**	**1.346%**	9s
	MVMoE/4E-L	8.180	1.872%	3s	11.960	1.473%	9s
VRPLTW	OR-Tools	14.815	1.432%	10.4m	25.823	2.534%	20.8m
	OR-Tools (x10)	14.598	*	1.7h	25.195	*	3.5h
	POMO-MTL	14.961	2.586%	3s	25.619	1.920%	12s
	MVMoE/4E	**14.937**	**2.421%**	4s	**25.514**	**1.471%**	13s
	MVMoE/4E-L	14.953	2.535%	4s	25.529	1.545%	12s
OVRPBTW	OR-Tools	8.758	0.927%	10.4m	14.713	2.268%	20.8m
	OR-Tools (x10)	8.675	*	1.7h	14.384	*	3.5h
	POMO-MTL	9.514	9.628%	3s	15.879	10.453%	10s
	MVMoE/4E	**9.486**	**9.308%**	4s	**15.808**	**9.948%**	11s
	MVMoE/4E-L	9.515	9.630%	3s	15.841	10.188%	10s
VRPBLTW	OR-Tools	14.890	1.402%	10.4m	25.979	2.518%	20.8m
	OR-Tools (x10)	14.677	*	1.7h	25.342	*	3.5h
	POMO-MTL	15.980	9.035%	3s	27.247	7.746%	11s
	MVMoE/4E	**15.945**	**8.775%**	4s	**27.142**	**7.332%**	12s
	MVMoE/4E-L	15.963	8.915%	4s	27.177	7.473%	11s

	Method	$n = 50$ Obj.	$n = 50$ Gap	$n = 50$ Time	$n = 100$ Obj.	$n = 100$ Gap	$n = 100$ Time
OVRPL	OR-Tools	6.522	0.480%	10.4m	9.966	1.783%	20.8m
	OR-Tools (x10)	6.490	*	1.7h	9.790	*	3.5h
	POMO-MTL	6.668	2.734%	2s	10.126	3.441%	9s
	MVMoE/4E	**6.650**	**2.454%**	3s	**10.097**	**3.148%**	10s
	MVMoE/4E-L	6.659	2.597%	3s	10.106	3.244%	9s
VRPBTW	OR-Tools	15.053	1.857%	10.4m	26.217	2.858%	20.8m
	OR-Tools (x10)	14.771	*	1.7h	25.496	*	3.5h
	POMO-MTL	16.055	8.841%	3s	27.319	7.413%	10s
	MVMoE/4E	**16.022**	**8.600%**	4s	**27.236**	**7.078%**	11s
	MVMoE/4E-L	16.041	8.745%	4s	27.265	7.190%	10s
OVRPBL	OR-Tools	5.771	0.549%	10.4m	8.555	2.459%	20.8m
	OR-Tools (x10)	5.739	*	1.7h	8.348	*	3.5h
	POMO-MTL	6.104	6.306%	2s	8.961	7.343%	8s
	MVMoE/4E	**6.076**	**5.843%**	3s	**8.942**	**7.115%**	9s
	MVMoE/4E-L	6.104	6.310%	3s	8.957	7.300%	9s
OVRPLTW	OR-Tools	8.728	0.656%	10.4m	14.535	1.779%	20.8m
	OR-Tools (x10)	8.669	*	1.7h	14.279	*	3.5h
	POMO-MTL	8.987	3.633%	3s	14.896	4.374%	11s
	MVMoE/4E	**8.966**	**3.396%**	4s	**14.828**	**3.903%**	12s
	MVMoE/4E-L	8.974	3.488%	4s	14.839	3.971%	10s
OVRPBLTW	OR-Tools	8.729	0.624%	10.4m	14.496	1.724%	20.8m
	OR-Tools (x10)	8.673	*	1.7h	14.250	*	3.5h
	POMO-MTL	9.532	9.851%	3s	15.738	10.498%	10s
	MVMoE/4E	**9.503**	**9.516%**	4s	**15.671**	**10.009%**	11s
	MVMoE/4E-L	9.518	9.682%	4s	15.706	10.263%	10s

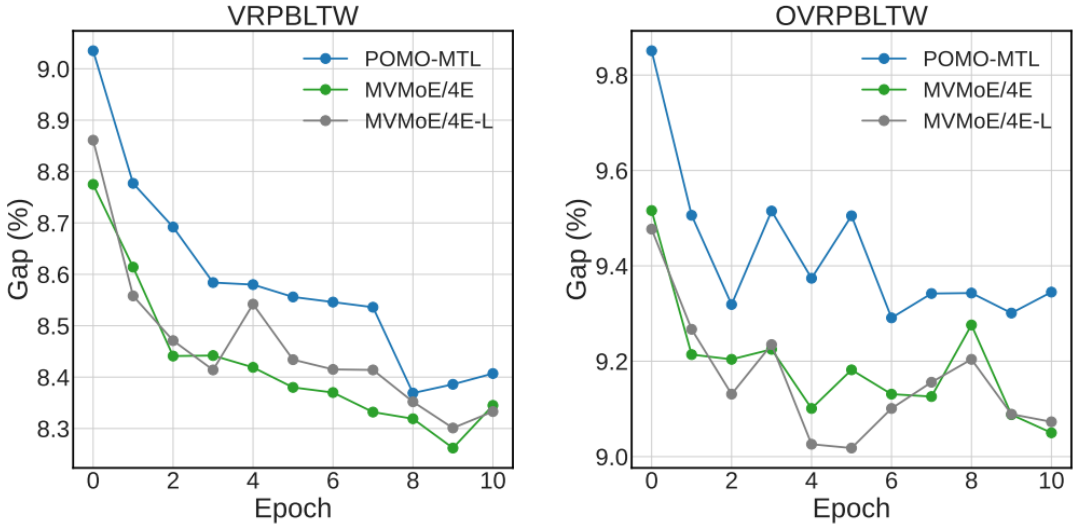


Figure 5. Few-shot generalization on unseen VRPs.

Experimental Evaluation: Ablation Study

- MVMoE/4E 를 Baseline으로 비교
- 빠른 수렴을 위하여 2500 Epoch / n = 50

Position of MOE (a)

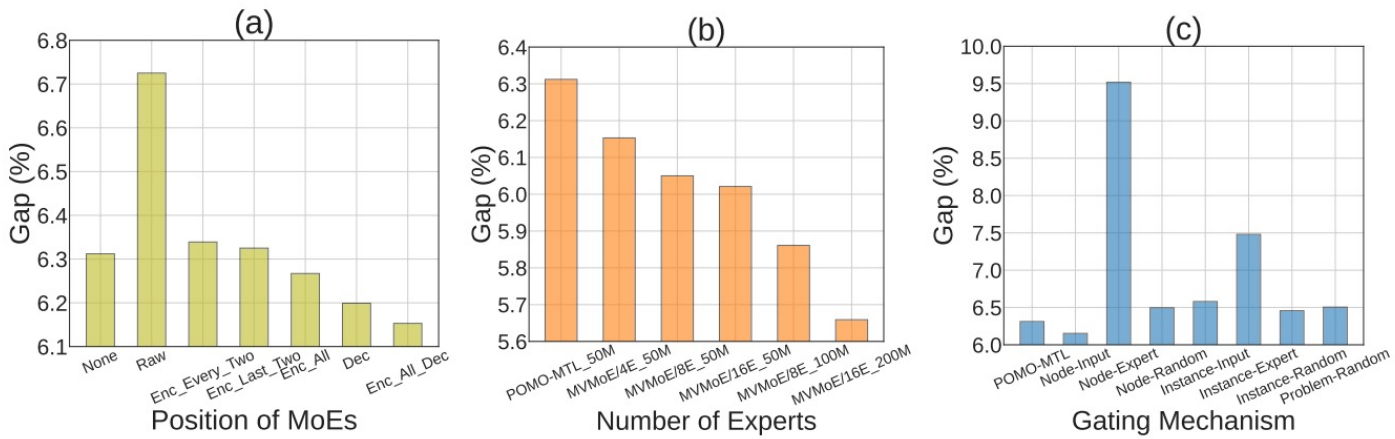
- Raw (Initial Embedding Linear Layer), Encoder, Decoder 3 개의 Position을 고려
- Raw에 쓰는 건 성능이 제일 안 좋고, 모든 Encoder와 Decoder에 사용하는 게 제일 성능이 좋음

Number of Experts (b)

- Expert를 8, 16으로 증가하고, Batch Size를 키워 더 많은 Data 공급
- 많은 Data와 함께 Expert를 늘리면 더 좋은 성능을 보임

Gating Mechanism (c)

- 다른 Gating Level (node-level, instance-level, problem-level)과 algorithm (input-choice, expert-choice, random gating) 사용
- node-input이 제일 좋은 성능을 보임



Conclusion

Conclusion

- 본 논문은 다양한 VRP를 동시에 해결할 수 있는 MVMoE를 제안
- Hierarchical Gating 구조를 통해 계산 효율성과 성능을 모두 확보
- Zero-shot, Few-shot, Real World Benchmark에서 우수한 성능을 보임

Future Works

- Large-scale VRP를 해결할 수 있는 Scalable MoE 기반 Model
- 다른 문제들에 적용할 수 있는 Generic Representation
- Gating Mechanism에 대한 Interpretability
- MoE에 대한 Scaling Law

Thank You!
Any Questions?