『2025 여름 세미나』

Understanding Mixture-of-Experts: A study in Efficient Scaling



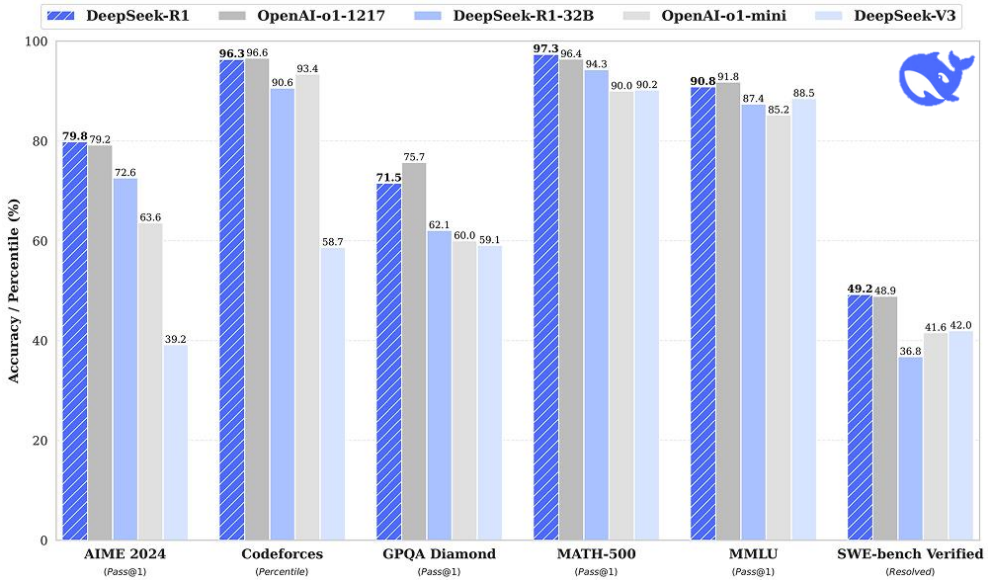
Pusan National University
Dept. of Electronics Engineering
Computer Vision & Signal Processing Lab.



Presented By
진인환

Why Mixture-of-Experts?

- The Case of "DeepSeek"
  - 결핍이 만든 혁신
    - 기존 생성형 AI는 조 단위의 비용과 고성능 GPU가 필수라는 인식이 강했음
      - Gemini Ultra는 약 2조 7천억 원, ChatGPT-4는 약 1조 1천억 원의 비용을 들여 학습
    - 그러나 DeepSeek은 약 80억 원 규모의 저사양 H800 GPU 수천 개를 활용해 R1 모델을 학습함
    - 동일한 파라미터 수에도 불구하고, MoE 구조를 사용하여 효율적인 연산을 통해 개발 비용을 획기적으로 절감함



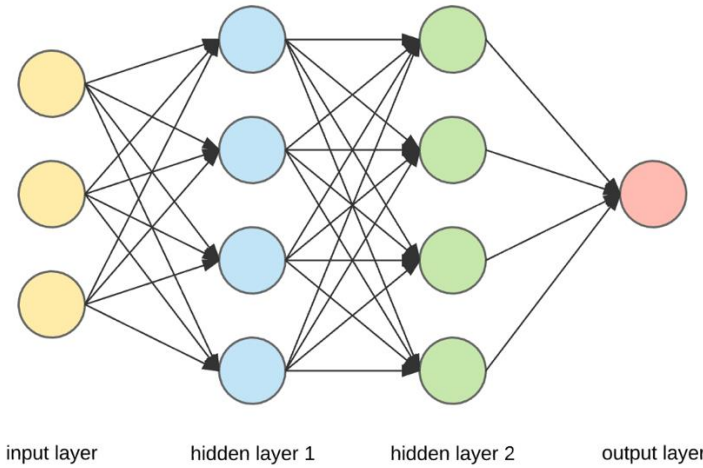
Outline

- Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer [ICLR 2017]
- MoE++: Accelerating Mixture-of-Experts Methods with Zero-Computation Experts [ICLR 2025 Oral]

Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer [ICLR 2017]

Introduction

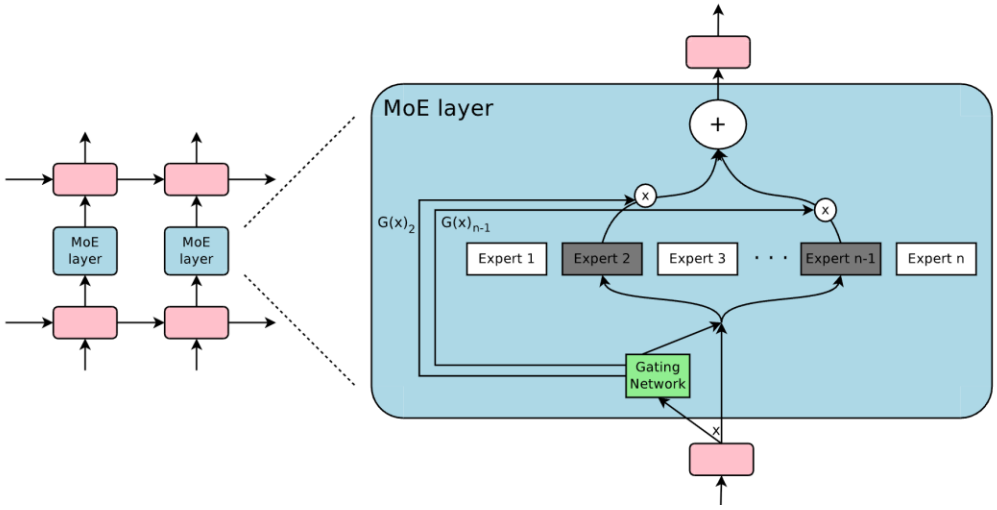
- Why Conditional Computation?
  - 기존 딥러닝 모델의 한계
    - 딥러닝 모델의 정보를 담을 수 있는 용량은 파라미터 수와 비례함
      - 학습 데이터와 모델의 크기는 딥러닝 성공 핵심 요인임 (Text, Image, Audio)
    - 모든 입력에 대해 전체 모델이 항상 활성화됨
      - 따라서 데이터와 모델의 크기가 증가함에 따라 연산량이 기하급수적으로 증가함
    - 따라서 계산 효율의 한계에 도달함



< Typical Neural Network Layer >

Introduction

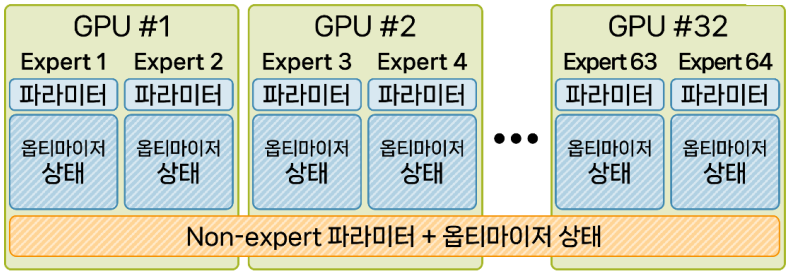
- Why Conditional Computation?
  - Conditional Computation
    - 입력에 따라 필요한 부분만 활성화 함
      - 모델 전체의 용량은 커도, 실제 연산은 일부 Expert만 사용함
    - 결과적으로 모델 용량은 유지한채 연산량을 감소시켜 효율적으로 학습이 가능함



< Miture-of-Experts Layer >

Introduction

- Problem & Challenge
  - Problem
    - GPU
      - 입력마다 선택된 일부 expert만 사용해야 하므로, branching이 필요하고 GPU 효율이 저하됨
    - Batch size
      - 각 expert가 처리하는 batch가 작아져 GPU 효율이 저하됨
    - Network bandwidth(통신)
      - Multi GPU 환경에서 expert가 분산되면, input과 output 전달을 위해 GPU 통신이 필요함
      - 이는 network bandwidth 병목을 유발하며, 특히 동적인 gating이 병목을 더욱 심화시킴

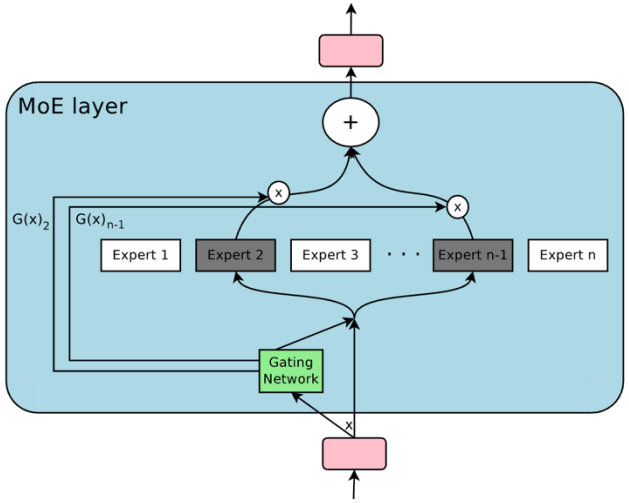


< Multi GPU Clustering >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Introduction

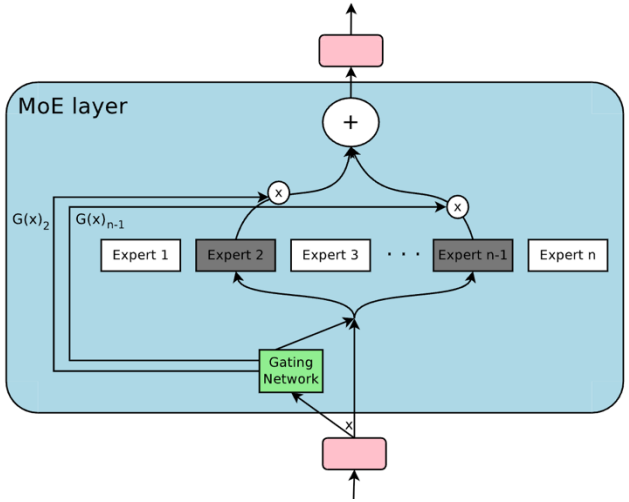
- Problem & Approach
  - Problem
    - Sparsity
      - Expert를 적절히 분산시키기 위해 추가적인 loss term이 필요함
        - ✓모든 input에 대해 하나의 expert만 선택
        - ✓일부 expert만 계속 사용되어 사용되지 않는 expert가 발생(load balancing)
    - Dataset
      - 기존 연구들은 상대적으로 적은 데이터셋으로 실험하여, 수십억 파라미터 모델을 학습하기에 충분하지 않음



< Mixture-of-Experts Layer >

Introduction

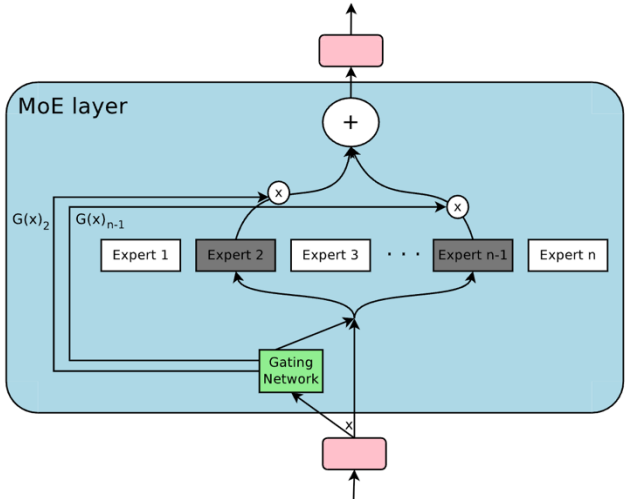
- Problem & Approach
  - Approach
    - Sparsely-Gated Mixture-of-Experts Layer(MoE)
      - 여러 개의 Feed-forward network의 experts와 학습 가능한 gating network로 구성됨
      - 개별 입력에 대해 Gating network는 sparse combination을 선택함
      - 이 전체 구조를 back-propagation로 joint하게 학습함



< Mixture-of-Experts Layer >

Method

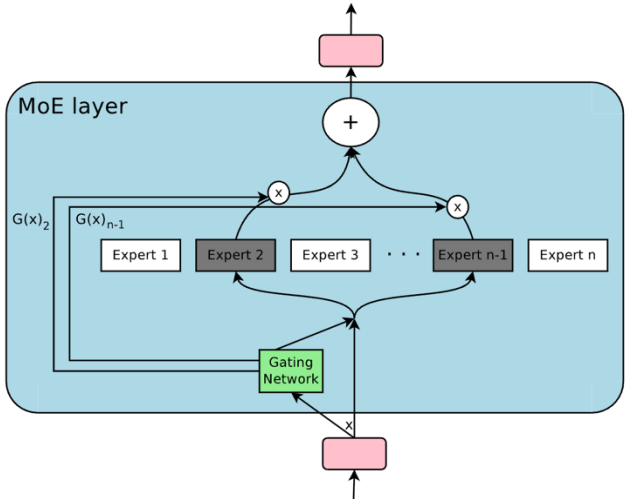
- Sparsely-Gated Mixture-of-Experts Layer(MoE)
  - Mixture-of-Experts(MoE)
    - MoE layer는 n개의 expert networks $E_1, \cdots E_n$의 집합과 gating network G로 구성됨
      - G의 출력은 n차원의 vector임
    - Expert는 개별 파라미터를 가진 neural network로 원칙적으로 모든 expert의 입력과 출력 크기가 동일해야 함
      - 본 논문에서는 네트워크까지 동일한 구조를 사용함



< Mixture-of-Experts Layer >

Method

- Sparsely-Gated Mixture-of-Experts Layer(MoE)
  - Mixture-of-Experts(MoE)
    - Gating network의 출력 $G(x)$는 각 expert의 활성화 정도를 나타내며, 대부분의 값이 0이 되는 sparse 출력을 가짐
    - MoE module의 최종 출력 y는 아래와 같이 정의됨
      - $y = \sum_{i=1}^{n} G(x)_i E_i(x)$
    - $G(x)_i = 0$ 일 때, $E_i(x)$는 계산하지 않기 때문에 연산량을 줄일 수 있음



< Mixture-of-Experts Layer >

Method

- Sparsely-Gated Mixture-of-Experts Layer(MoE)
  - Gating Network
    - Noisy Top-K Gating
      - 위의 Softmax gating에 sparsity와 noise를 추가한 형태
      - Sparsity는 computation을 줄일 수 있으며, noise는 load balancing 문제를 해결함

✓ $G(x) = Softmax\big(KeepTopK(H(x), k)\big)$

✓ $H(x)_i = \big(x \cdot W_g\big)_i + StandardNormal() \cdot Softplus\big((x \cdot W_{noise})_i\big)$

- W_{noise} : 개별 입력에 대해 추가할 noise를 결정하는 trainable weight

✓ $KeepTopK(v, k)_i = \begin{cases} v_i & if\ v_i\ is\ in\ the\ top\ k\ elements\ of\ v. \\ -\infty & othersie. \end{cases}$

Expert	Original	Noise	After Noise	Top-2 result
E1	10	-4	6	❌
E2	9	0	9	✅
E3	3	+5	8	✅
E4	2	0	2	❌

< Example on Noisy Top-K >

Method

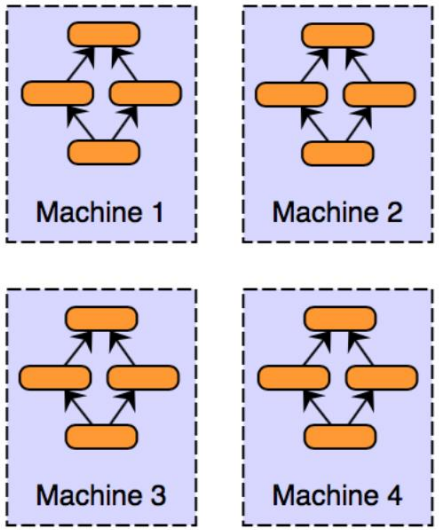
- Sparsely-Gated Mixture-of-Experts Layer(MoE)
  - Gating Network
    - Training
      - Gating network는 expert model들과 함께 back-propagation로 학습함
      - K >1이면, Top k expert들은 nonzero의 연속된 값을 가져 미분이 가능함
        - ✓이전 연구(Bengio et al., 2015)에서는 gating은 미분이 불가능한 Boolean 값으로 설정하여 강화학습을 사용함

Method

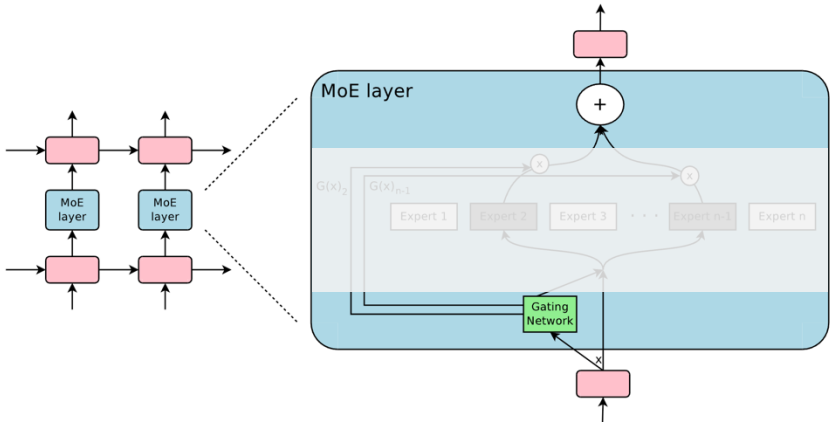
- Addressing Performance Challenges
  - The Shrinking Batch Problem
    - Problem setting
      - 현대 CPU와 GPU에서 파라미터들을 효율적으로 연산하기 위해서는 large batch size가 필수적임
      - n개의 expert에서 k개를 선택한다고 할 때, batch b에서 적은 batch를 입력 받음
        - ✓Approximately $\frac{kb}{n} \ll b$
      - 따라서 naïve MoE implementation은 expert 증가하면 매우 비효율적임
        - ✓현실적인 한계로 무한정 batch를 키울 없음

Method

- Addressing Performance Challenges
  - The Shrinking Batch Problem
    - Mixing Data Parallelism and Model Parallelism
      - Data Parallelism(standard layer, gating network)
        - ✓Multi-GPU 환경에서 모든 GPU가 같은 모델을 복제해서 개별 batch만을 학습
        - ✓서로 다른 mini-batch 데이터를 학습하고 gradient만 나중에 평균으로 맞춤

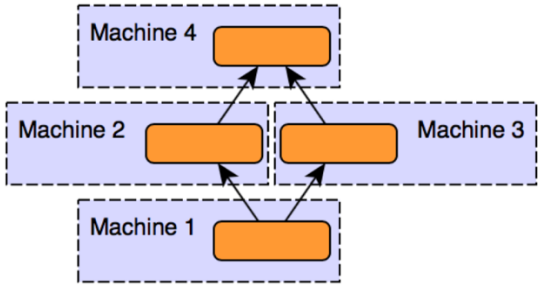


< Data Parallelism >



Method

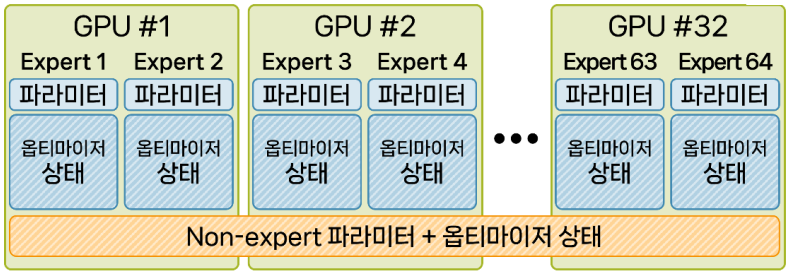
- Addressing Performance Challenges
  - The Shrinking Batch Problem
    - Mixing Data Parallelism and Model Parallelism
      - Model Parallelism(expert layer)
        - ✓모든 expert를 복제하지 않고 각 GPU에 분산하여 저장함
          - 하나의 GPU는 몇 개 expert만을 가짐
        - ✓각 GPU가 가지고 있는 batch 중에 어떤 어떤 expert를 사용하지 gating network가 결정
        - ✓GPU끼리 통신해서 필요한 expert에게 데이터를 보내줌
        - ✓Expert는 모든 GPU의 input을 합쳐서 큰 batch로 처리함
      - 각 expert는 여러 GPU에서 input을 모아서 batch size가 커짐
        - ✓d개의 device로 학습할 때, 대략 $\frac{kbd}{n}$ 가 됨



< Model Parallelism >

Method

- Addressing Performance Challenges
  - The Shrinking Batch Problem
    - Network Bandwidth
      - Input과 Output 데이터를 expert 간에 주고받을 때 네트워크 통신이 필요함
        - ✓하지만 GPU는 통신에는 특화되어 있지 않아, 시간이 오래 걸린다.
      - 따라서 hidden layer를 키워 연산량을 늘리고, 통신에 드는 시간만큼 충분히 연산을 수행함
        - ✓통상적으로 GPU 환경에서 연산량 (FLOPs)과 통신량(Bytes)이 1000:1이 되도록 설정하여 병목 현상을 억제할 수 있음
          - 연산량 = (input_size x hidden_size) + (hidden_size x output_size)
          - 통신량 = (input_size + output_size) x float 32 크기(4 bytes)



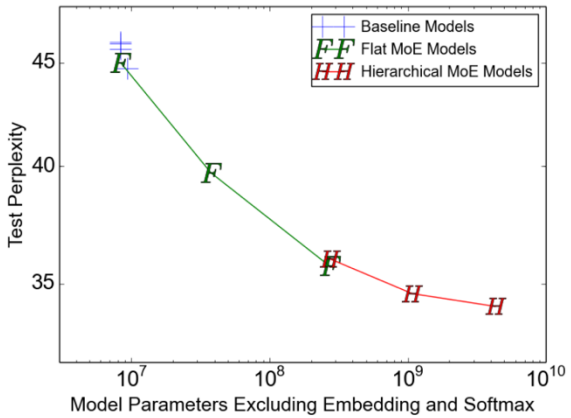
< Multi GPU Clustering >

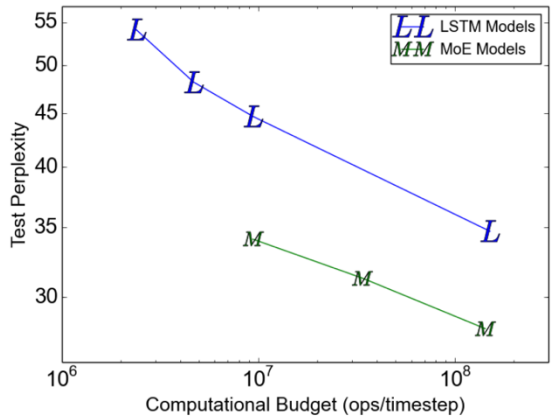
Method

- Addressing Performance Challenges
  - Balancing Expert Utilization
    - Soft constraint approach
      - Gating network는 몇 개의 동일한 expert를 선택할 때 빠르게 수렴됨
        - ✓이는 많이 선호된 expert가 빠르게 학습되어 더 많이 gating되는 self-reinforcing 때문임
      - Training example의 batch 단위 gate 값의 합으로 정의한 Importance를 loss에 추가함
        - ✓$L_{importance}$는 importance value들의 분산으로 모든 expert가 동일한 importance를 갖도록 함
        - ✓$Importance(x) = \sum_{x \in X} G(x)$
        - ✓$L_{importance}(X) = w_{importance} \cdot CV\big(Importance(X)\big)^2$

Experiments

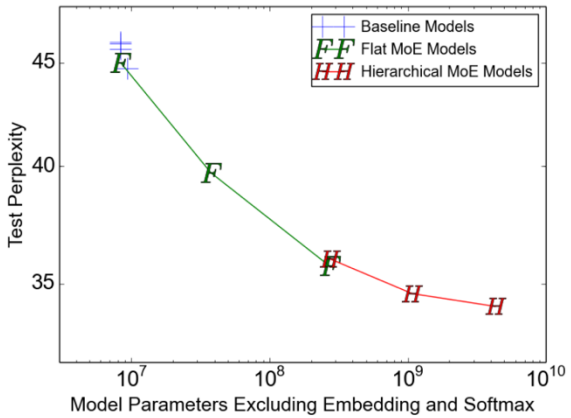
- 1 Billion word Language Modeling Benchmark
  - Dataset
    - 1 Billion word Language Modeling Benchmark
      - 뉴스 기사로부터 수집된 데이터셋으로, 총 8억 9,200만 단어와 약 79만 개의 어휘를 포함함
    - Next Token Prediction Task에서 사용됨
  - Previous State-of-the-Art
    - 다층 Long-Short-Trem Memory(LSTM) 모델로 LSTM layer는 200만 ~ 1억 5100만개의 파라미터를 사용함
    - Parameter 수에 따라 성능이 증가하고 동시에 computation cost도 증가함

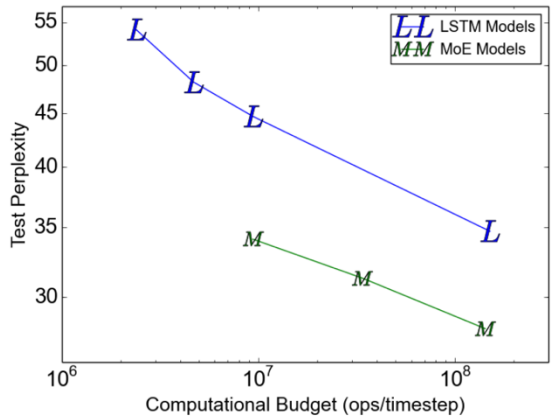




Experiments

- 1 Billion word Language Modeling Benchmark
  - MoE models
    - 두개의 LSTM layer 사이에 MoE layer를 추가한 구조로 Layer 크기와 expert 개수를 실험에 따라 다양하게 조정함
  - Low Computation, Varied Capacity
    - 모델의 용량 변화가 성능에 미치는 영향을 확인하기 위해 연산량을 동일하게 유지하면서 MoE의 크기만 다양하게 조정함
      - 모든 모델은 input 하나당 800만회의 multiply-and-add 연산을 수행함(softmax 제외)





Experiments

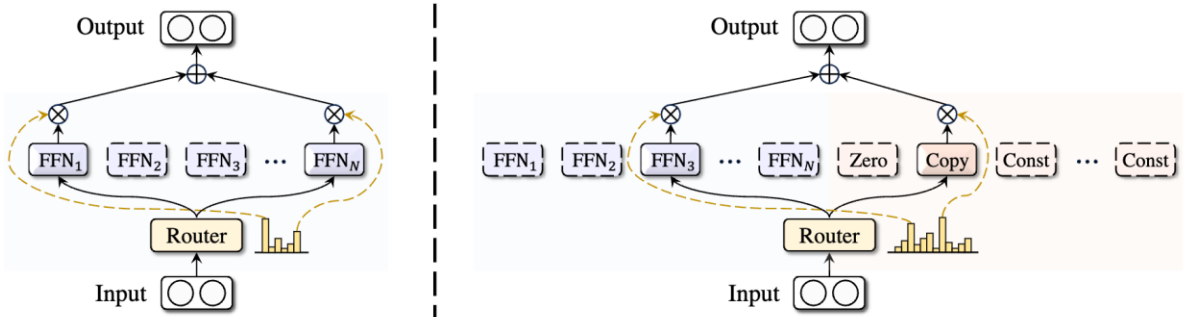
- 1 Billion word Language Modeling Benchmark
  - Varied Computation, High Capacity
    - 이전 섹션에서 가장 큰 모델과 비슷한 capacity이지만, 연산량을 더 많이 쓰는 모델과 비교함
      - 더 큰 LSTM 모델, 수는 적지만 보다 큰 expert 모델
    - Low-Budget MoE Model이 기존 SOTA 연산량의 6% 수준에 불과하면서도 더 나은 성능을 달성함
      - ops/timestep: RNN/LSTM 기반 모델에서 사용되는 지표로, time step마다 순차적으로 계산할 때 필요한 연산량을 의미함
    - MoE 구조를 사용하여 더 많은 파라미터를 효율적으로 사용함으로써 성능을 높임

	Test Perplexity 10 epochs	Test Perplexity 100 epochs	#Parameters excluding embedding and softmax layers	ops/timestep	Training Time 10 epochs	TFLOPS /GPU
Best Published Results	34.7	30.6	151 million	151 million	59 hours, 32 k40s	1.09
Low-Budget MoE Model	34.1		4303 million	8.9 million	15 hours, 16 k40s	0.74
Medium-Budget MoE Model	31.3		4313 million	33.8 million	17 hours, 32 k40s	1.22
High-Budget MoE Model	**28.0**		4371 million	142.7 million	47 hours, 32 k40s	**1.56**

MoE++: Accelerating Mixture-of-Experts Methods with Zero-Computation Experts [ICLR 2025 Oral]

Introduction

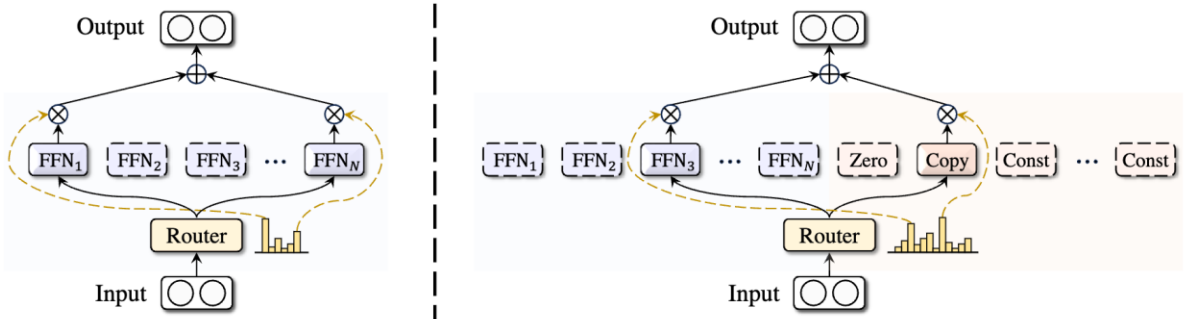
- Limitation of Mixture-of-Experts Method
  - Fixed number of Expert
    - 기존의 MoE 방법론모든 입력 토큰에 대해 고정된 수의 Feed-Forward Network(FFN)를 활성화함
      - 선택된 top 2 FFNs의 output을 합쳐 다음 layer의 입력으로 사용함
    - 하지만 language task에서 모든 tokens는 다른 prediction difficulty를 가짐
      - 콤마(,)와 같은 쉬운 토큰은 single expert로 충분함
      - 기존 expert와 맞지 않아 MoE layer를 통과하는 것이 유리한 경우도 있음
    - 이러한 insight에서 고정된 수의 mixing mechanism은 학습과 추론에 비효율적임
      - 결과적으로 sub-optimal한 model performance를 야기함



< Compare with Standard MoE and MoE++ >

Introduction

- Solution
  - General and Heterogeneous MoE Framework
    - 동적인 할당을 위해 세가지 type의 zero-computation experts 를 도입함
      - Zero expert
        - ✓입력을 지워버리는 expert
      - Copy expert
        - ✓입력을 복제하는 expert
      - Constant expert
        - ✓입력을 학습가능한 vector로 대체하는 expert

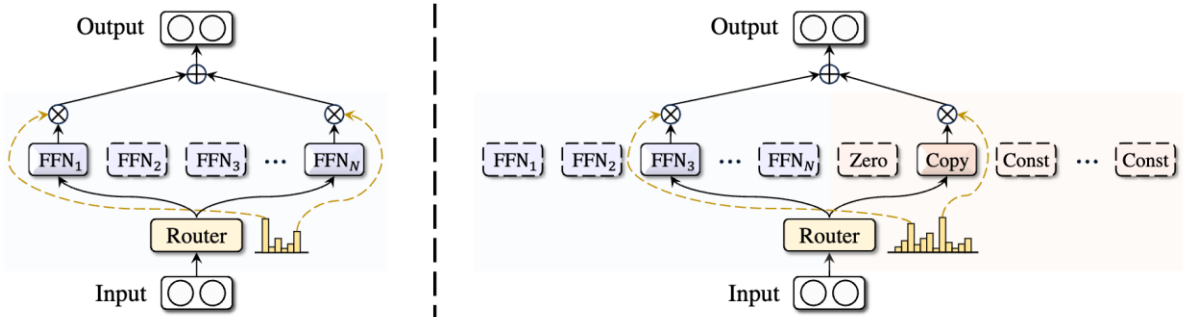


< Compare with Standard MoE and MoE++ >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Introduction

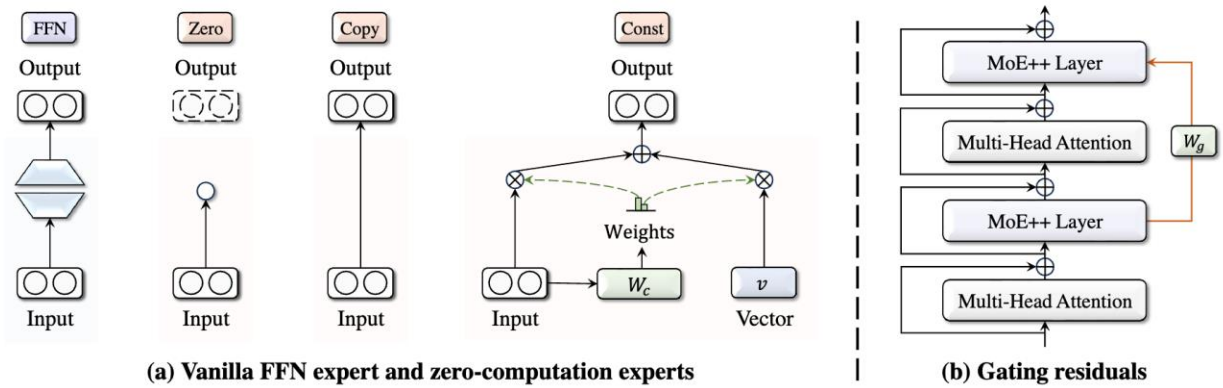
- Solution
  - General and Heterogeneous MoE Framework
    - 이러한 heterogeneous한 구조를 통해 추가적인 overhead없이 sub-network의 조합을 확대시킴
    - 추가적으로 이전 layer의 gating score를 현재 MoE layer에 반영하는 gating residual로 이전 pathway를 고려한 expert selection 도입



< Compare with Standard MoE and MoE++ >

Method

- MoE++
  - Zero-Computation Experts
    - 제안하는 구조는 아래의 원칙들을 유지한채 MoE를 redesign함
      - 가벼운 구조로 단순 토큰을 빠르게 처리(streamlined)
      - Vanilla MoE와 공정한 비교를 위해 파라미터 수를 거의 늘리지 않음(negligible params)
    - 이를 반영하여 세가지 type의 zero-computation experts를 제안함
      - Discard, skip and replace operations

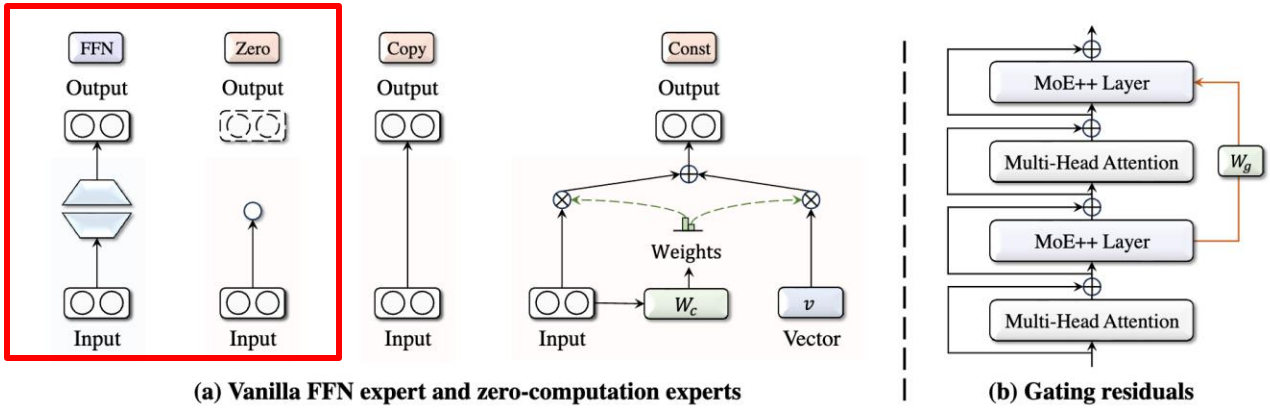


< Components of the proposed MoE++ >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Method

- MoE++
  - Zero-Computation Experts
    - Zero Experts
      - 가장 단순한 형태의 zero-computation으로, 입력을 그대로 버리고 출력을 0으로 변환함
        - ✓$E_{zero}(x) = 0$
      - Top-2 MoE++ layer를 Top-1 처럼 동작하여, 다른 하나의 expert 결과만 그대로 사용됨
      - 따라서 간단하거나 복잡한 token을 상황에 따라 유연하게 다룰 수 있음

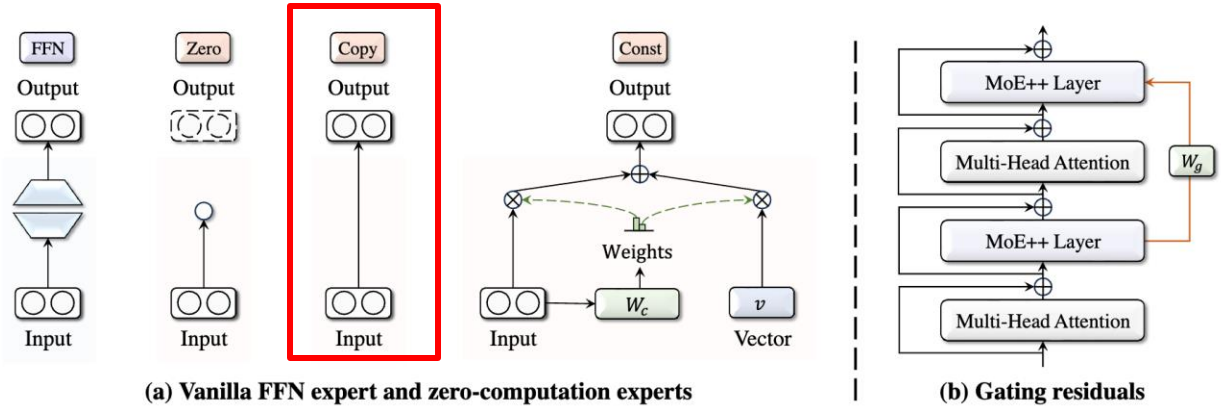


< Components of the proposed MoE++ >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Method

- MoE++
  - Zero-Computation Experts
    - Copy Experts
      - Residual network(ResNet)와 같이 입력과 출력이 동일한 expert
        - ✓$E_{copy}(x) = x$
      - Copy expert는 MoE++ layer를 통과하지 않고 skip하는 효과를 얻음
        - ✓Input token이 현재 layer의 Experts와 맞지 않는 경우 복잡한 연산 없이 통과시킬 선택지 제공

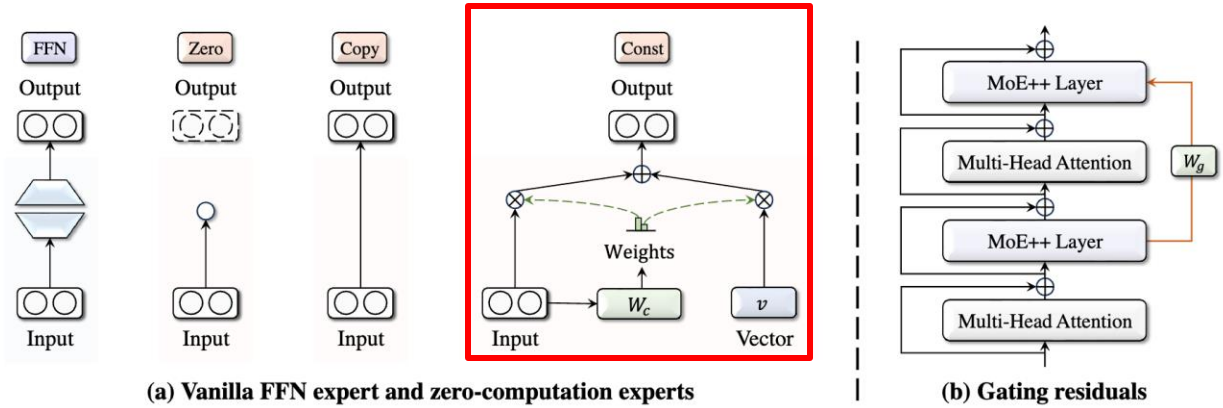


< Components of the proposed MoE++ >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Method

- MoE++
  - Zero-Computation Experts
    - Constant Experts
      - Zero expert와 copy expert는 trainable parameter가 없어 input token을 유연하게 다루는 것에 한계가 있음
      - 이를 보완하기 위해 input token x 를 trainable vector v 로 변환하는 constant expert를 도입함
        - ✓하지만 입력을 완전히 대체하면 정보 손실이 크므로, 학습가능한 weight W_c로 입력과 변환 벡터의 비율을 동적으로 조정함
        - ✓$E_{const}(x) = \alpha_1 x + \alpha_2 v, \quad [\alpha_1, \alpha_2] = Softmax(W_c x)$
          - $W_c \in \mathbb{R}^{2 \times D}$, D는 input token x의 hidden size



(a) Vanilla FFN expert and zero-computation experts (b) Gating residuals

< Components of the proposed MoE++ >

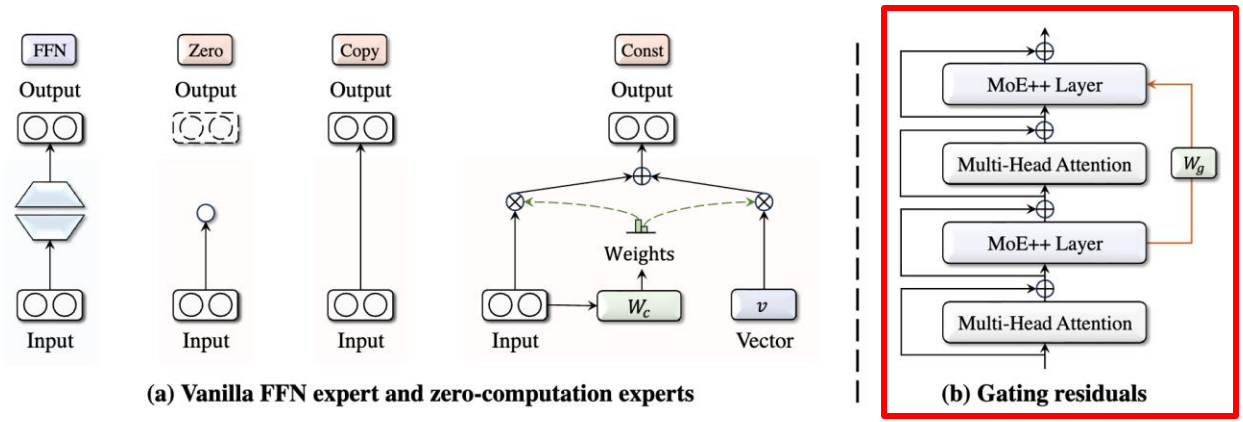
부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Method

- MoE++
  - Pathway-Aware Router
    - Gating Residual
      - MoE++은 heterogeneous한 experts를 포함하여, router의 구조가 더욱 중요한 역할을 함
      - Residual network와 유사하게 이전 layer의 routing score를 현재 layer에 더해줌
        - ✓Token x_j의 N개 experts로 구성된 j번째 layer에서 routing score는 아래와 같이 정의됨

$$\checkmark G(x^j) = \begin{cases} W^j x^j, & if\ j = 1 \\ W^j x^j + W_g^j G(x^{j-1}), & if\ j > 1 \end{cases}$$

- $W_g^j \in \mathbb{R}^{N \times N}$는 학습가능한 transformation matrix로 이전 layer의 score를 통합함



< Components of the proposed MoE++ >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Method

- MoE++
  - Load Balance During Pretraining
    - Heterogenous Load Balance Loss
      - Vanilla MoE는 모든 expert가 동일한 수의 tokens가 할당되도록 학습함
      - 하지만, MoE++는 zero-computation experts와 FFN experts의 파라미터 수가 달라 동일하게 token을 할당하는 것은 비효율적임
      - Hyper-parameter τ를 도입하여 zero-computation과 FFN experts에 할당할 token 비율을 조정함

✓T번째 input token x_t에대해 heterogenous laod balance loss $\mathcal{L}_b$는 아래와 같이 정의됨

✓$\mathcal{L}_b = \sum_{i=1}^{N} \eta_i f_i P_i$, $\eta_i = \begin{cases} 1, & if\ Expert\ i\ is\ an\ FFN\ expert, \\ \tau, & if\ Expert\ i\ is\ a\ zero-computation\ expert, \end{cases}$

✓$f_i = \frac{1}{T}\sum_{t=1}^{T} 1\ (Token\ x_t\ selects\ Expert\ i)$,

✓$P_i = \frac{1}{T}\sum_{t=1}^{T} Softmax\big(G(x_t)\big)_i$,

- T는 tokens의 수, N은 expert의 갯수

✓τ가 커질수록 더 많이 zero-computation에 할당됨

Method

- MoE++
  - Load Balance During Pretraining
    - Expert Capacity
      - 일부 expert에 token이 과하게 몰리는 현상을 방지하기 위해, 각 expert별로 token 처리 최대 개수(capacity)를 제한함
      - Expert type을 반영하여 제한하기 위해 아래와 같이 expert capacity를 정의함

$$\checkmark C_i = \begin{cases} \gamma \frac{\tau T}{\tau N_{FFN} + N_{ZC}}, & if\ Expert\ i\ is\ an\ FFN\ expert, \\ \gamma \frac{T}{\tau N_{FFN} + N_{ZC}}, & if\ Expert\ i\ is\ a\ zero-computation\ expert \end{cases}$$

        - γ은 preset capacity factor(hyper-parameter), T는 token의 수
        - N_{FFN}, N_{ZC}는 FFN expert와 zero-computation expert의 수
        - τ가 작아질 수록 zero-computation expert의 capacity가 증가함
      - Expert가 capacity에 도달시 추가 token은 그대로 다음 레이어로 전달됨

Method

- MoE++
  - Size and architectures of MoE++ and vanilla MoE model
    - MoE++
      - Zero-computation experts는 거의 연산과 통신 비용이 없음
      - Inference 시 효율
        - ✓항상 2개가 활성화 되지만, zero-computation expert가 포함되면 실제 연산량은 감소
      - Activated Params
        - ✓(Inference 시 사용되는 파라미터/ 전체 파라미터)

Methods	# Activated Params	# Layers	# Hidden Size	# Intermediate Size	# Heads	# Head Dim	# The Number of FFN Experts	# Zero/Copy/Constant Experts
MoE 0.6B/8E MoE++ 0.6B/(8+4)E	0.2B/0.6B ≤0.2B/0.6B	12	768	2048	12	64	8	- 1/1/2
MoE 1B/16E MoE++ 1B/(16+4)E	0.2B/1B ≤0.2B/1B	12	768	2048	12	64	16	- 1/1/2
MoE 2B/32E MoE++ 2B/(32+8)E	0.2B/2B ≤0.2B/2B	12	768	2048	12	64	32	- 1/1/6
MoE 7B/16E MoE++ 7B/(16+4)E	1.2B/7B ≤1.2B/7B	24	1536	4096	16	96	16	- 1/1/2

< Compare with Standard MoE and MoE++ >

Experiments

- Experimental Setup
  - Model Settings
    - Training
      - Framework: Megatron (Shoeybi et al., 2019, 오픈소스 코드 사용)
      - 환경: 4개 노드, 32 A100 GPU 클러스터에서 학습 수행
    - 모델 설정
      - MoE++ 0.6B/(8+4)E
        - ✓약 0.6B 파라미터 규모의 MoE++ 모델
        - ✓8개의 FFN expert + 4개 zero-computation expert 포함
    - Fair comparison
      - Inference 시, token 당 activated parameter를 동일하게 설정함

Experiments

- Experimental Setup
  - Hyper-parameter
    - Dense Language Model에서 사용되는 일반적인 세팅을 그대로 사용
    - 기존 transformer의 모든 FFN layer를 MoE++ layer로 대체함
    - 각 MoE++ layer의 Top-k = 2로 설정함
      - 결과적으로 Dense 모델 대비 약 2배의 연산량
  - Evaluation Benchmarks
    - MoE++는 다양한 0-shot 및 few-shot benchmark에서 기존 Vanilla MoE와 Dense 모델 대비 성능을 비교함
    - lm-evaluation-harness를 통해 ARC, LAMBADA, MMLU 등 주요 LLM benchmark로 평가함

Experiments

- Main Results
  - Vanilla MoE
    - 같은 파라미터 수에서 MoE++이 더 빠르며 높은 성능을 보여줌
      - τ가 작을 때 일부 token이 zero-computation expert를 사용하여 효율이 높아짐
      - 하지만 일정 이상 커지면 GPU 비효율로 점차 낮아짐

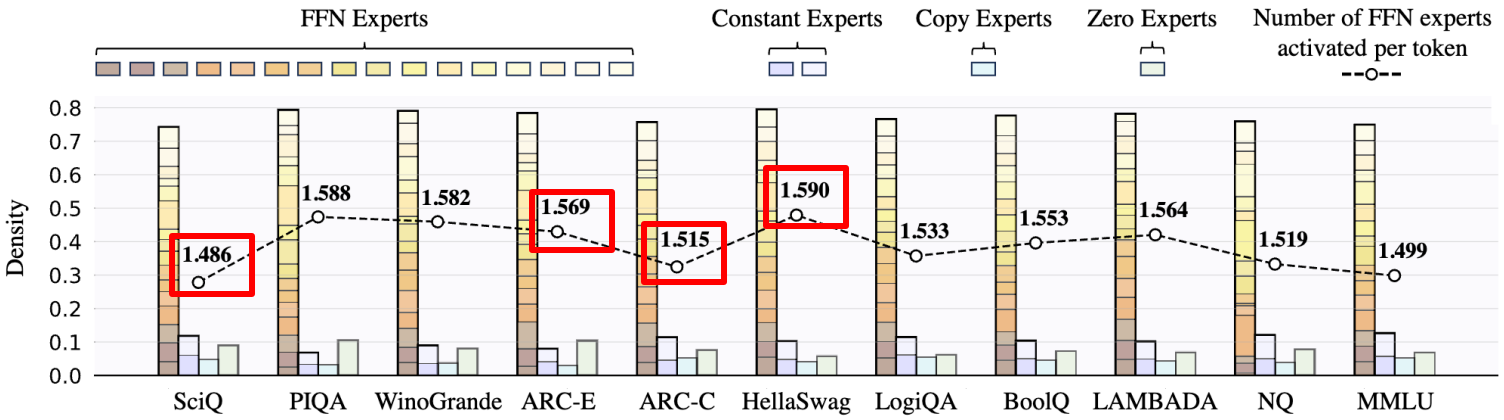
Methods	# τ	# Expert Forward Time (ms)	# Throughput Increase	Commonsense & Reading Comprehension				
				SciQ	PIQA	WinoGrande	ARC-E	HellaSwag (10)
MoE 0.6B/8E	-	535.3	-	**76.6**	67.3	50.2	47.6	40.3
MoE++ 0.6B/(8+4)E	0.10	202.4	164.5%	73.1	65.1	51.9	46.0	36.1
MoE++ 0.6B/(8+4)E	0.25	277.8	92.7%	74.8	66.5	51.1	48.4	39.7
MoE++ 0.6B/(8+4)E	0.50	387.3	38.2%	74.9	**68.4**	49.3	48.9	40.7
MoE++ 0.6B/(8+4)E	0.75	427.6	25.2%	76.5	67.6	51.9	**50.1**	**41.8**
MoE++ 0.6B/(8+4)E	1.00	449.6	19.1%	75.9	67.6	**52.2**	49.0	41.7
MoE 1B/16E	-	610.9	-	**79.3**	68.4	**54.2**	48.9	43.7
MoE++ 1B/(16+4)E	0.10	289.3	111.2%	76.3	67.6	51.1	48.5	40.0
MoE++ 1B/(16+4)E	0.25	384.9	58.7%	77.4	68.3	50.2	48.2	42.5
MoE++ 1B/(16+4)E	0.50	469.7	30.1%	77.5	67.6	52.5	49.7	44.3
MoE++ 1B/(16+4)E	0.75	500.3	22.1%	78.3	**70.3**	51.7	49.7	**44.6**
MoE++ 1B/(16+4)E	1.00	530.2	15.2%	**79.3**	69.5	52.5	**51.5**	**44.6**
MoE 2B/32E	-	683.4	-	77.9	70.0	51.6	51.6	46.0
MoE++ 2B/(32+8)E	0.10	417.9	63.5%	76.0	68.2	52.3	49.7	42.7
MoE++ 2B/(32+8)E	0.25	473.7	44.3%	76.2	69.4	52.6	51.7	46.0
MoE++ 2B/(32+8)E	0.50	532.7	28.3%	78.4	70.0	**54.2**	50.6	47.0
MoE++ 2B/(32+8)E	0.75	561.0	21.8%	79.6	**70.2**	51.9	51.4	47.3
MoE++ 2B/(32+8)E	1.00	590.5	15.7%	**81.7**	69.8	53.6	**52.4**	**47.7**
MoE 7B/16E	-	1859	-	78.3	72.6	**58.8**	53.1	61.3
MoE++ 7B/(16+4)E	0.75	1455	27.8%	**80.1**	**73.6**	58.2	**53.6**	**61.8**

< Comparison between MoE++ and vanilla MoE models.>

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Experiments

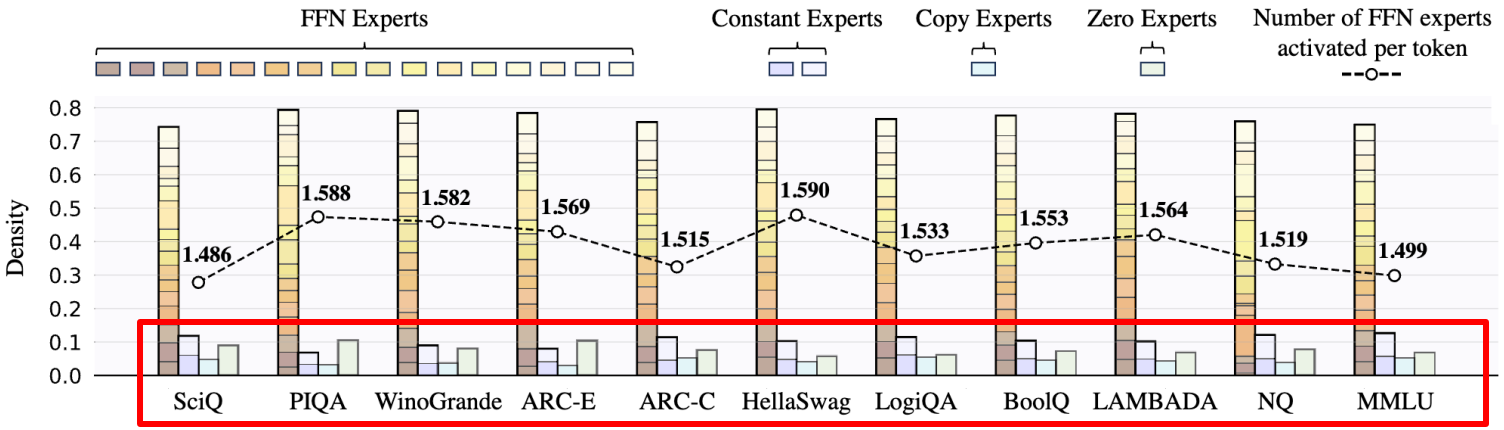
- Qualitative Analysis
  - Visualization of the Expert Load Distribution at the Task Level
    - Task 별 FFN expert 사용량
      - Token마다 활성화 되는 FFN expert 수는 task에 따라 매우 다르며, task의 난이도와는 관계 없음
        - ✓ARC Challenge보다 Arc Easy의 FFN expert 사용량이 더 많음
      - MoE++는 task 난이도가 아닌 token-level 지식, 복잡도에 따라 expert를 할당함



< Visualization of the expert load distribution at the task level >

Experiments

- Qualitative Analysis
  - Visualization of the Expert Load Distribution at the Task Level
    - Zero Expert 활성화
      - 모든 expert 유형 중 zero experts가 평균적으로 가장 자주 선택됨
      - 특히 간단한 task일수록 zero expert 사용 비율이 높음
        - ✓불필요한 계산을 피하고 효율을 높이는 설계가 반영됨

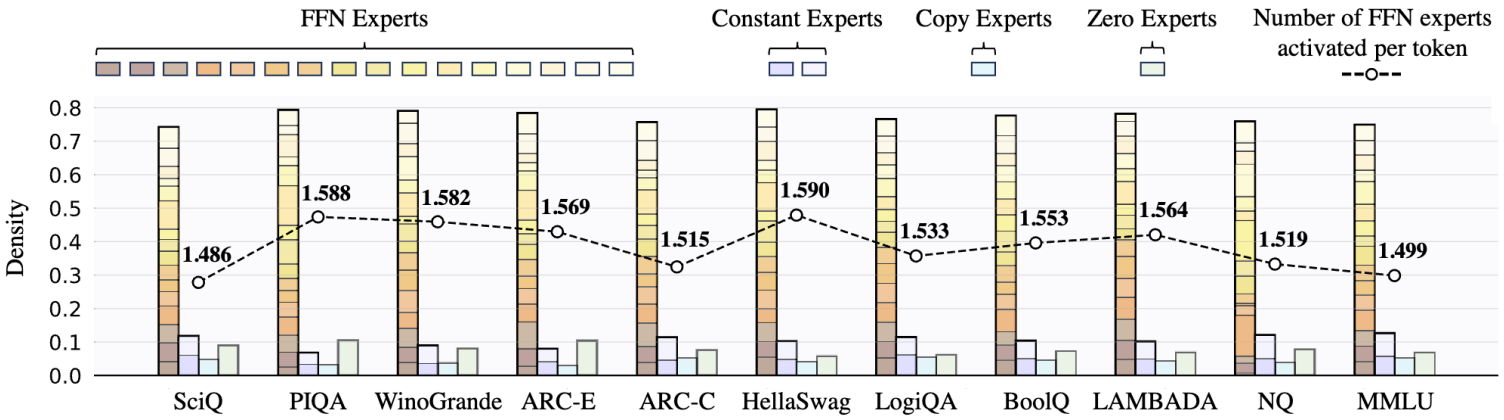


< Visualization of the expert load distribution at the task level >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Experiments

- Qualitative Analysis
  - Visualization of the Expert Load Distribution at the Task Level
    - Expert 사용 패턴
      - 서로 다른 task는 서로 다른 expert 할당 패턴을 보임
        - ✓어떤 task는 특정 expert에 더 몰리는 경향, 어떤 task는 더 균등하게 분산되는 경향
      - MoE++는 task별 토픽, 내용에 맞게 expert를 다르게 할당하며 유연하게 동작함



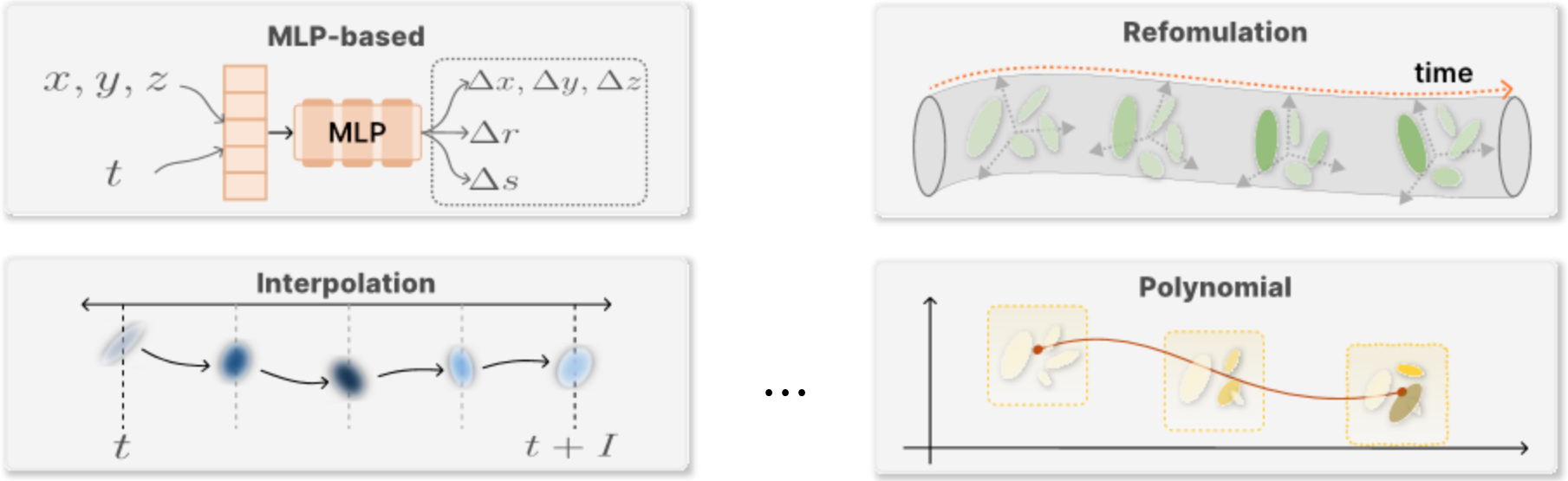
< Visualization of the expert load distribution at the task level >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

MoE-GS: Mixture of Experts for Dynamic Gaussian Splatting [Submitted to NeurIPS 2025]

MoE-GS (Submitted to NeurIPS 2025)

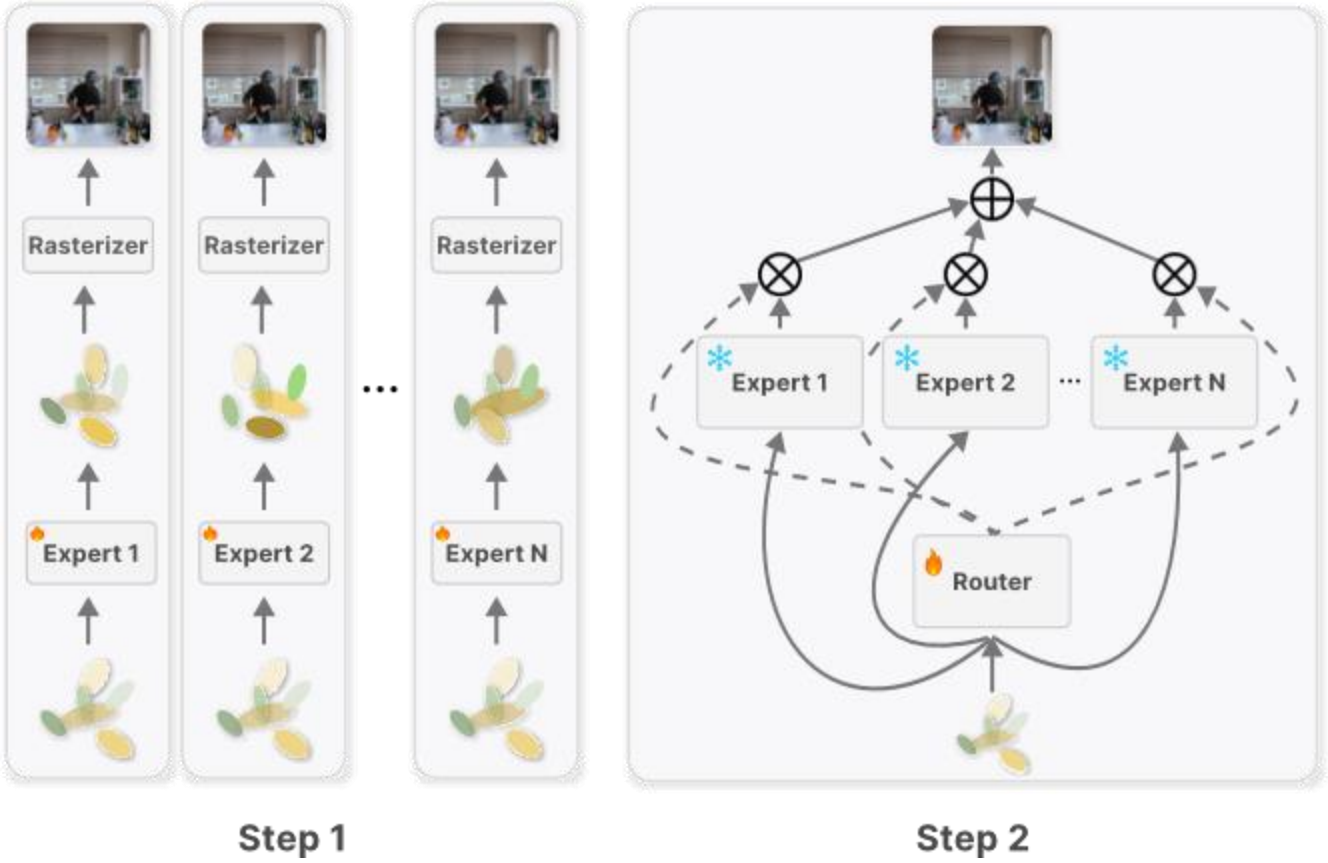
- Goal
  - MoE 구조를 Dynamic Gaussian Splatting에 적용
    - 다양한 deformation 방법을 사용하고 있는 experts(기존 모델들)을 최적으로 조합하여 동적 장면 복원의 성능을 향상하고자 함



< Candidate Experts >

MoE-GS (Submitted to NeurIPS 2025)

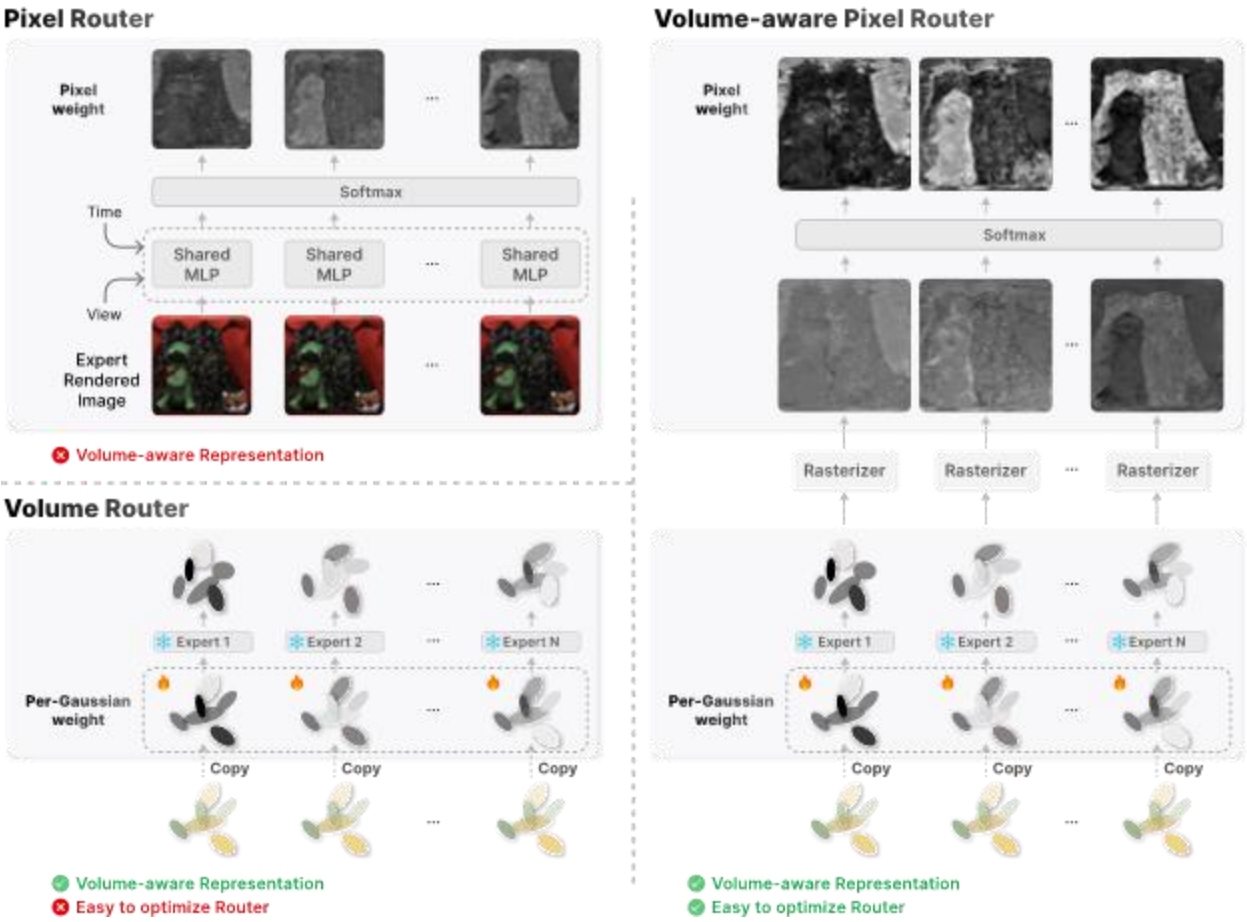
- Main framework



< MoE-GS framework >

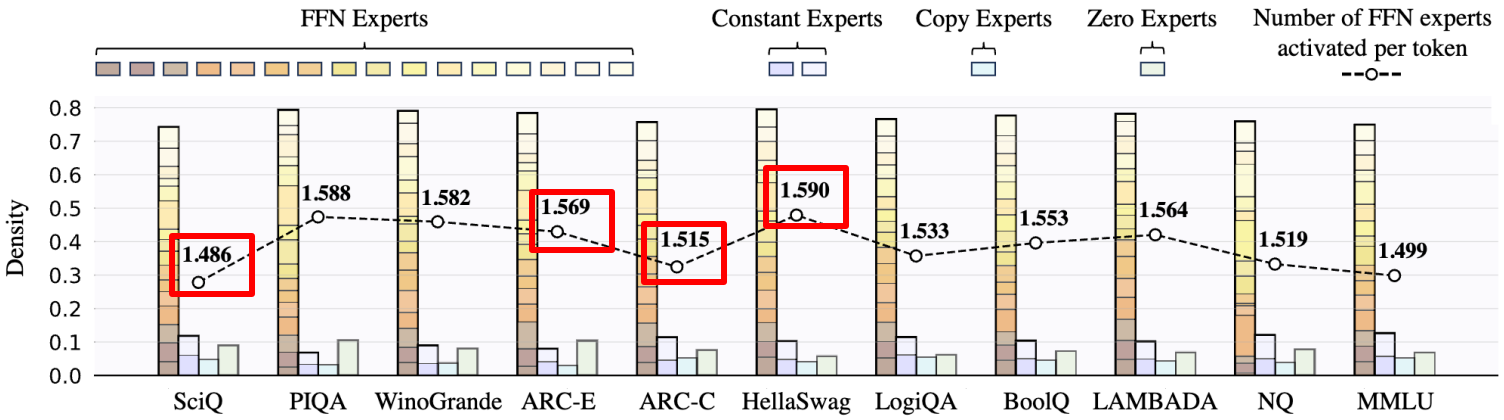
MoE-GS (Submitted to NeurIPS 2025)

- Volumetric-aware Pixel Router (Gating Network)



Appendix

- MoE++
  - Visualization of the Expert Load Distribution at the Task Level
    - 아래 시각화 된 expert 분포는 “MoE++7B/(16+4)E” 모델의 12번째 Layer를 시각화 한 결과임
      - 논문 Appendix D에 전체 MoE 레이어에서의 정성 결과를 포함하고 있음



< Visualization of the expert load distribution at the task level >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing