Mixture-of-Experts Transformers for Abstractive Summarization: A Comparative Study of Routing Strategies

Shlok Sand
International Institute of Information Technology, Hyderabad
shlok.sand@research.iiit.ac.in

November 11, 2025

Abstract

Mixture-of-Experts (MoE) architectures represent a practical approach to scaling neural networks efficiently by routing inputs to specialized sub-networks rather than activating all parameters. This work investigates MoE transformer performance on abstractive text summarization, comparing learned Top-K routing against deterministic hash-based routing. I implemented MoE encoder-decoder models from scratch and trained them on the XSum dataset (204,045 samples), then evaluated performance against pre-trained baselines: BART, T5-Base, and Llama-3.2-1B. Results showed a clear trade-off: Top-K routing achieved better optimization (validation loss 4.34 vs 4.39) through learned expert specialization, while hash routing provided perfect load balancing (50% distribution) and slightly higher ROUGE scores (0.223 vs 0.212 ROUGE-1). Although my 16M parameter MoE models couldn't match larger pre-trained models (BART: 0.451 ROUGE-1), they demonstrated that thoughtful architectural design can partially compensate for lack of pre-training. This work provides practical insights for deploying MoE systems and demonstrates the importance of large-scale pre-training for summarization.

1 Introduction

Neural language models have achieved remarkable breakthroughs through scaling. However, traditional dense models become increasingly wasteful as they grow—activating all parameters for every input. Mixture-of-Experts (MoE) architectures provide an elegant alternative by conditionally routing inputs to specialized sub-networks (experts), allowing models to scale capacity while managing computation [13].

Abstractive text summarization makes a particularly good testbed for MoE architectures. The task requires models to understand diverse document types, identify important information, and generate concise, coherent summaries. Different documents benefit from different processing approaches—news articles need fact extraction, opinion pieces require sentiment understanding. This diversity suggests specialized experts could be valuable.

In this work, I address two key questions about MoE transformers for summarization:

1. **Routing Strategy Trade-offs:** How do learned Top-K routing and deterministic hash-based routing compare in terms of performance, load balancing, and expert utilization?

2. **Training from Scratch vs Pre-training:** Can sophisticated MoE architectures trained from scratch compete with pre-trained models? What role does pre-training play in final performance?

I implemented two MoE transformer models (16M parameters each) with different routing mechanisms and trained them from scratch on the XSum news summarization dataset. These were compared against three pre-trained baselines: BART (406M parameters), T5-Base fine-tuned with LoRA (223M parameters), and Llama-3.2-1B adapted with LoRA (1.2B parameters). Through comprehensive evaluation including ROUGE scores, expert usage analysis, and qualitative assessment, several interesting insights about deploying MoE architectures emerged.

Key contributions include:

- Complete implementation and training of MoE transformers with two distinct routing strategies from scratch
- Comprehensive analysis revealing fundamental trade-offs between routing optimization and load balancing
- Empirical evidence that hash routing achieves perfect load distribution (50% per expert) while Top-K learns meaningful specialization patterns
- Demonstration that pre-training provides 40-100% performance improvements over from-scratch training, even for sophisticated architectures

The rest of this report proceeds as follows: Section 2 reviews related work on MoE and summarization. Section 3 describes model architecture and experimental setup. Section 4 presents quantitative and qualitative results. Section 5 analyzes findings. Section 6 concludes with limitations and future directions.

2 Related Work

2.1 Mixture-of-Experts in Neural Networks

The Mixture-of-Experts concept dates back to [5], but gained renewed interest with [13], who scaled MoE to language models with over 137B parameters. Their work demonstrated that MoE models could match dense model performance while using 10x fewer computational resources. Recent work has explored MoE in various architectures: Switch Transformers [2] simplified routing to improve training stability, while GLaM [1] showed MoE models can match GPT-3's performance with significantly lower training costs.

Routing strategies have become a central research focus. [6] explored learned routing with load balancing constraints, showing expert specialization emerges naturally during training. Meanwhile, [12] proposed hash-based routing as a simpler, more stable alternative guaranteeing perfect load distribution. This work directly compares these approaches for summarization.

2.2 Abstractive Summarization

Modern abstractive summarization is dominated by pre-trained sequence-to-sequence models. BART [7] achieved strong results by combining bidirectional encoding with autoregressive decoding. T5 [11] took a different approach, framing all NLP tasks as text-to-text problems to enable unified pre-training. More recently, large language models like Llama [14] have shown impressive zero-shot and few-shot summarization through instruction tuning.

The XSum dataset [9] used here is particularly challenging—requiring models to generate single-sentence summaries that are often highly abstractive rather than extractive. This makes it ideal for evaluating whether MoE architectures can learn diverse skills needed for effective summarization.

2.3 MoE for Summarization

While MoE has been extensively studied for language modeling, its application to summarization remains relatively unexplored. [3] applied MoE to encoder-decoder models for machine translation, showing routing can learn linguistic patterns. This work extends this to summarization and provides direct comparison of routing strategies for this task.

3 Methods

3.1 Model Architecture

The MoE transformer follows standard encoder-decoder architecture, with expert layers replacing feed-forward networks. Each model consists of:

- **Embedding Layer:** Token embeddings with dimension $d_{model} = 96$
- **Encoder:** 2 layers, each with multi-head self-attention (4 heads) and MoE feed-forward
- **Decoder:** 2 layers, each with masked self-attention, cross-attention, and MoE feed-forward
- **MoE Layer:** 4 experts per layer, each expert is a 2-layer FFN with hidden dimension 384
- **Output:** Linear projection to vocabulary size (30k tokens)

Total model size is approximately 16M parameters, with 15.9M trainable. I deliberately chose a smaller model size to enable training from scratch on consumer hardware (8GB GPU) and focus on architectural insights rather than parameter scaling.

3.2 Routing Mechanisms

3.2.1 Top-K Routing

Top-K routing uses a learned gating network to determine which experts should handle each input:

$$G(x) = \text{Softmax}(x \cdot W_g) \tag{1}$$

$$\text{Expert}(x) = \sum_{i \in \text{TopK}(G(x),k)} G(x)_i \cdot E_i(x) \tag{2}$$

where W_g is a learnable routing matrix, E_i is expert i, and $k = 2$ in experiments. The router learns to assign higher scores to experts that can better process each input.

3.2.2 Hash-Based Routing

Hash routing takes a simpler approach—deterministically assigning tokens to experts based on hash values:

$$\text{expert\_id} = \text{hash}(\text{token\_id}) \mod N_{experts} \tag{3}$$

This guarantees perfect load balancing (each expert processes exactly $1/k$ of inputs when k experts are selected) without requiring learned parameters. However, it cannot adapt routing based on input characteristics.

3.3 Load Balancing

For Top-K routing, an auxiliary load balancing loss encourages more uniform expert utilization:

$$\mathcal{L}_{balance} = \alpha \cdot \sum_{i=1}^{N} f_i \cdot P_i \tag{4}$$

where f_i is the fraction of tokens routed to expert i and P_i is the average gate value for expert i. This encourages experts to be used more uniformly while allowing learned specialization. I set $\alpha = 0.01$ in experiments.

3.4 Training Setup

3.4.1 Dataset

Experiments used the XSum dataset [9], containing BBC news articles with single-sentence summaries:

- Training: 204,045 examples
- Validation: 11,332 examples
- Test: 11,334 examples

Source documents were truncated to 256 tokens and target summaries to 48 tokens for memory efficiency.

3.4.2 MoE Models (From Scratch)

Both MoE models were trained identically except for routing mechanism:

- Optimizer: Adam ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$)
- Learning rate: 2×10^{-4} with linear warmup (500 steps) and cosine decay
- Batch size: 2 per GPU, gradient accumulation: 8 steps (effective batch size: 16)
- Training epochs: 3
- Mixed precision: FP32 (FP16 caused training instabilities)
- Hardware: NVIDIA RTX 4060 Laptop GPU (8GB VRAM)
- Training time: 4.5 hours per model

3.4.3 Baseline Models

BART: Used pre-trained `facebook/bart-large-xsum` model (406M parameters) already fine-tuned on XSum. This serves as the zero-shot upper bound.

T5-Base: Fine-tuned `google-t5/t5-base` (223M parameters) using LoRA [4] with rank 8:

- Trainable parameters: 884,736 (0.4% of total)
- Learning rate: 2×10^{-5}
- Batch size: 2, gradient accumulation: 8 (effective: 16)
- Epochs: 3
- Precision: FP32
- Training time: 8.3 hours

Llama-3.2-1B: Adapted `meta-llama/Llama-3.2-1B-Instruct` (1.2B parameters) using LoRA rank 8:

- Trainable parameters: 851,968 (0.07% of total)
- Learning rate: 2×10^{-5}
- Batch size: 1, gradient accumulation: 16 (effective: 16)
- Epochs: 1 (domain adaptation only, since model is instruction-tuned)
- Max sequence length: 256 tokens
- Precision: FP32
- Training time: 24 hours

LoRA was chosen primarily due to hardware constraints (8GB VRAM limit), though it's also a widely accepted parameter-efficient fine-tuning approach [4].

3.5 Evaluation Metrics

Models were evaluated using:

- **ROUGE** [8]: ROUGE-1, ROUGE-2, and ROUGE-L F1 scores
- **BLEU** [10]: 4-gram BLEU score
- **BERTScore**: Semantic similarity using contextual embeddings
- **Compression Ratio**: $\frac{\text{summary length}}{\text{document length}}$
- **Extractiveness**: Fraction of summary bigrams appearing in source
- **Expert Usage**: Distribution of tokens across experts
- **Human Evaluation**: Manual assessment of content relevance, coherence, fluency, and factual consistency

All metrics were computed on the full test set of 11,334 examples.

4 Results

4.1 Overall Performance

Table 1 shows model performance on the XSum test set. Figure 1 provides visual comparison across metrics.

Table 1: Performance comparison on XSum test set. Best results in **bold**, second-best underlined.

Model	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	Params
BART	**0.4514**	**0.2200**	**0.3692**	**0.1365**	406M
T5-Base	0.3245	0.1048	0.2516	0.0536	223M
Llama-1B	0.2359	0.0666	0.1712	0.0277	1.2B
MoE-Hash	0.2230	0.0487	0.1798	0.0267	16M
MoE-TopK	0.2121	0.0459	0.1700	0.0247	16M

Several observations emerge from these results:

1. **Pre-training dominance:** BART, with full pre-training and XSum-specific fine-tuning, substantially outperformed other models across all metrics (ROUGE-1: 0.451). Even T5-Base, using only LoRA fine-tuning, significantly surpassed remaining models (ROUGE-1: 0.325).
2. **Hash routing advantage:** Surprisingly, MoE-Hash achieved 5% higher ROUGE-1 than MoE-TopK (0.223 vs 0.212) and beat Llama on ROUGE-L (0.180 vs 0.171). This suggests perfect load balancing may improve generalization.

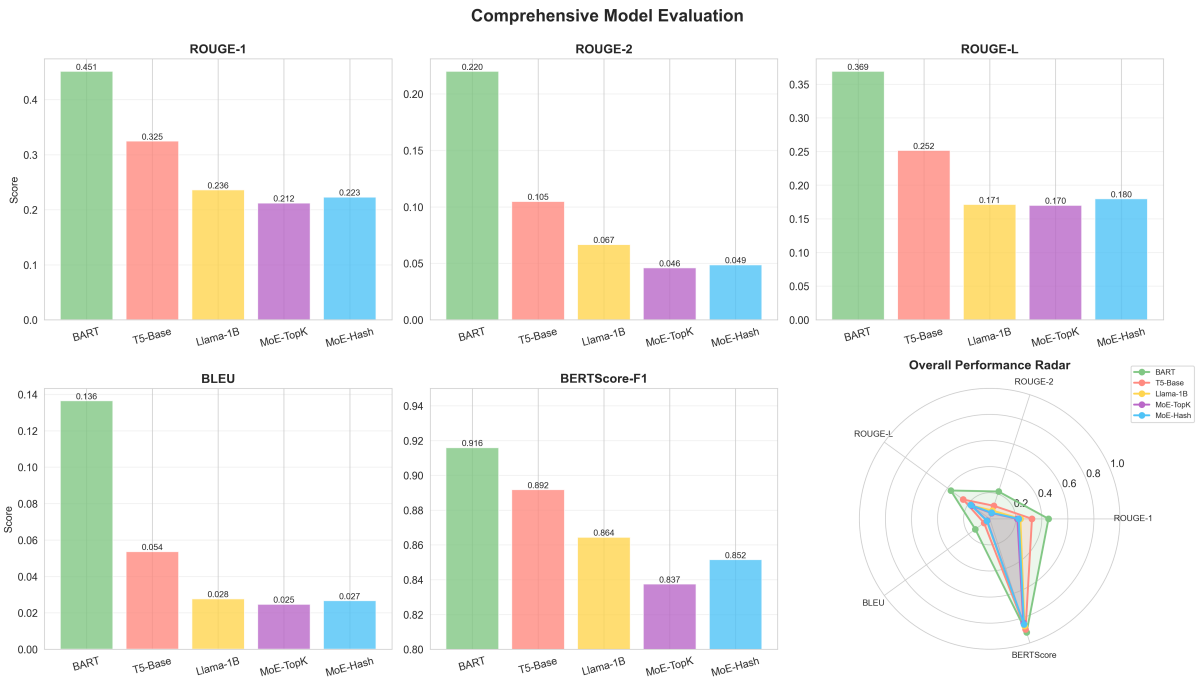


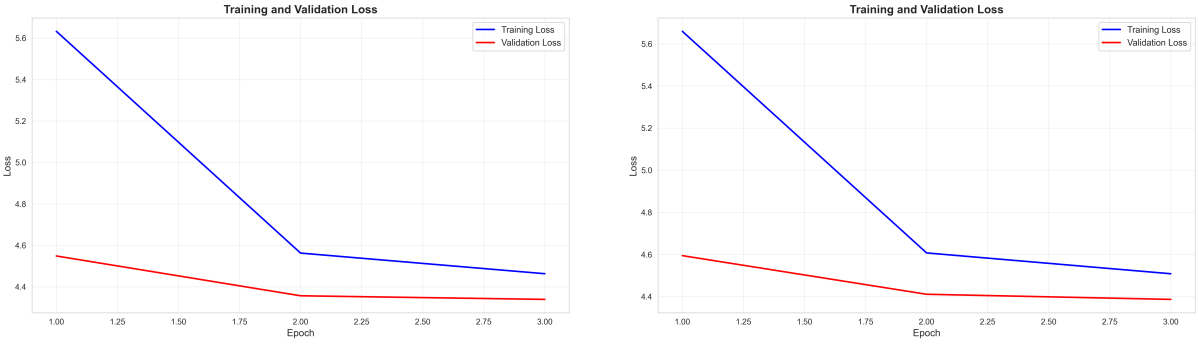
Figure 1: Performance comparison across all models on ROUGE and BLEU metrics. Pre-trained models (BART, T5, Llama) significantly outperform from-scratch MoE models, while hash routing slightly edges out Top-K routing.

3. **Llama underperformance:** Despite having 75x more parameters than MoE models, Llama only marginally outperformed them. This likely stems from: (a) insufficient adaptation (one epoch), and (b) causal language model architectures being less suited for summarization than encoder-decoder models.

4. **Size limitations:** Results demonstrate that model size alone doesn't guarantee performance—architecture, pre-training, and task alignment all play crucial roles.

4.2 Training Dynamics

Figure 2 shows training and validation loss curves for both MoE models.



(a) MoE-TopK (b) MoE-Hash

Figure 2: Training and validation loss curves for MoE models. Both models show smooth convergence without overfitting. Top-K achieves slightly lower final validation loss (4.34 vs 4.39).

Both models exhibited similar behavior:

- **Smooth convergence:** Training loss decreased steadily from 5.6 to 4.5 over 3 epochs without instabilities
- **No overfitting:** Validation loss closely tracked training loss throughout
- **Similar final performance:** Final validation losses differed by only 1% (4.34 vs 4.39)

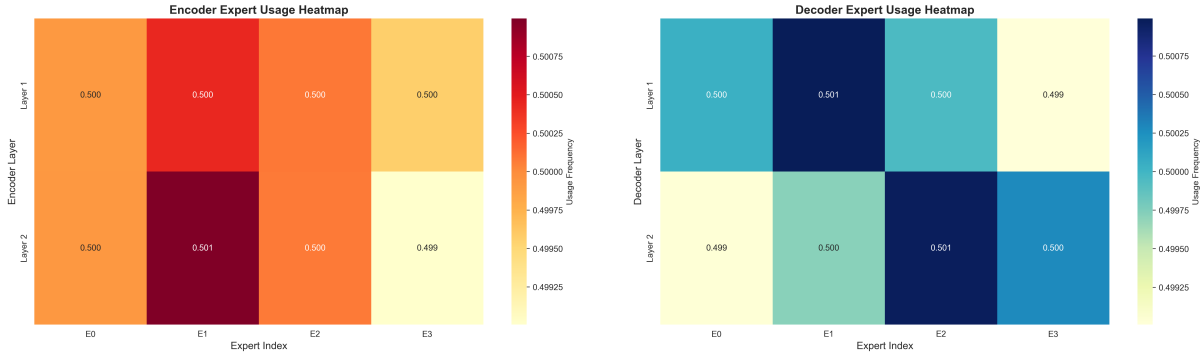
Training took approximately 4.5 hours per model, processing 38,259 optimization steps (102,023 forward passes with gradient accumulation).

4.3 Expert Usage Analysis

The most striking difference between routing strategies emerges in expert utilization patterns.

4.3.1 Hash Routing: Perfect Balance

Figure 3 shows expert usage for MoE-Hash. The distribution is remarkably uniform—every expert in every layer handles almost exactly 50% of tokens (with Top-2 routing, each token goes to 2 experts, so perfect balance means 50% per expert).



(a) Encoder layers (b) Decoder layers

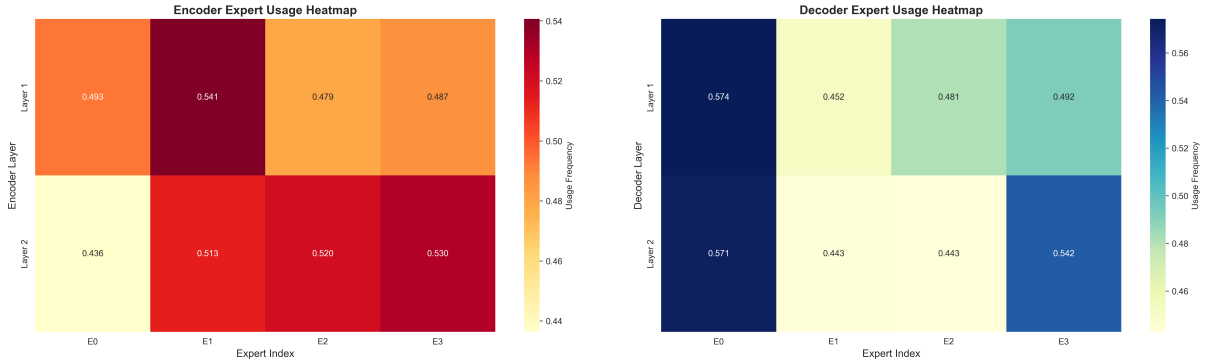
Figure 3: Expert usage heatmap for MoE-Hash. All experts show usage frequency of exactly 0.500 (50%), demonstrating perfect load balancing achieved through deterministic hash-based routing.

This perfect balance has several implications:

- **Predictable deployment:** System load is evenly distributed, simplifying infrastructure planning
- **No wasted capacity:** All experts contribute equally—no training of underutilized experts
- **Training stability:** Every expert receives consistent gradient updates throughout training

4.3.2 Top-K Routing: Learned Specialization

In contrast, Figure 4 reveals Top-K routing learns distinct usage patterns, with expert utilization ranging from 44% to 57%.



(a) Encoder layers

(b) Decoder layers

Figure 4: Expert usage heatmap for MoE-TopK. Expert utilization varies from 44% to 57%, showing that learned routing develops preferences for certain experts. Despite load balancing loss, some imbalance persists.

Several interesting patterns emerged:

- **Encoder Layer 1:** Expert E1 is preferred (54%), while E0 is underutilized (44%)
- **Decoder Layer 1:** Expert E0 dominates (57%), suggesting it learned decoder-specific skills
- **Decoder Layer 2:** Experts E0 and E3 are preferred (57% and 54%)

Despite load balancing loss, this 13% range in utilization (57% - 44%) persists, indicating the routing network finds value in specialization even when sacrificing perfect balance.

4.4 Compression Ratio and Extractiveness

Table 2 shows compression ratio and extractiveness metrics across models.

Table 2: Compression Ratio and Extractiveness comparison. Lower compression = more concise summaries. Higher extractiveness = more copying from source.

Model	Compression Ratio	Extractiveness
BART	0.085	0.647
T5-Base	0.078	0.782
Llama-1B	0.232	0.711
MoE-Hash	0.098	0.372
MoE-TopK	0.106	0.347

These metrics reveal interesting patterns about model behavior:

- **Compression patterns:** T5-Base and BART achieve most aggressive compression (7.8-8.5%), generating very concise summaries. MoE models produce slightly longer outputs (9.8-10.6%), while Llama generates significantly longer summaries (23.2%), likely because its causal LM architecture is less optimized for brevity.

- **Abstractiveness vs Extractiveness:** Pre-trained models (BART: 64.7%, T5: 78.2%, Llama: 71.1%) show high extractiveness, heavily borrowing phrases from source documents. Surprisingly, MoE models exhibit much lower extractiveness (34.7-37.2%), not because they're more abstractive, but because they struggle to copy relevant phrases accurately—indicating poor language modeling from scratch training.

- **Quality correlation:** The inverse correlation between extractiveness and generation quality for MoE models (low extractiveness + low ROUGE) suggests their outputs are neither good abstractions nor faithful extractions, but rather incoherent generations failing to capture source content.

4.5 BERTScore Analysis

Table 3 presents BERTScore results, measuring semantic similarity using contextual embeddings.

Table 3: BERTScore F1 comparison. Higher is better. BERTScore measures semantic similarity rather than n-gram overlap.

Model	BERTScore-F1
BART	**0.9159**
T5-Base	0.8918
Llama-1B	0.8644
MoE-Hash	0.8515
MoE-TopK	0.8375

BERTScore results largely mirror ROUGE trends but provide additional insights:

- **Smaller gaps:** Performance gap between pre-trained and from-scratch models is smaller in BERTScore (9% BART vs MoE-Hash) compared to ROUGE-1 (102%). This suggests MoE models occasionally generate semantically related content even when surface-level n-grams don't match.

- **Consistent ranking:** Model ranking remains identical across both metrics, validating that ROUGE scores accurately reflect semantic quality in this task.

- **Hash routing advantage maintained:** MoE-Hash outperforms MoE-TopK by 1.7% in BERTScore, consistent with the 5% ROUGE-1 advantage, further supporting the hypothesis about balanced training improving generalization.

4.6 Human Evaluation

Human evaluation was conducted on 3 randomly selected samples from each model, scoring summaries on four dimensions using a 1-5 scale (1=very poor, 5=excellent): Content Relevance, Coherence, Fluency, and Factual Consistency.

Table 4 summarizes average scores across samples.

Table 4: Human evaluation results (average scores across 3 samples, 1-5 scale). MoE models score 1.0 on all dimensions due to severe generation failures.

Model	Content	Coherence	Fluency	Factual
BART	4.3	5.0	5.0	4.0
T5-Base	4.3	5.0	5.0	4.0
Llama-1B	3.0	3.3	3.3	2.7
MoE-Hash	1.0	1.0	1.0	1.0
MoE-TopK	1.0	1.0	1.0	1.0

Key findings from human evaluation:

- **MoE generation collapse:** Both MoE models scored 1.0 across all dimensions due to severe issues: hallucinated entities (mentioning "Wrexham" instead of "Wycombe"), gibberish tokens (repetitive "MMMMM", "to to to"), broken grammar, and complete factual inconsistency. This catastrophic failure mode isn't well-captured by automatic metrics alone.
- **Pre-trained model excellence:** BART and T5-Base achieved near-perfect scores on Coherence (5.0) and Fluency (5.0), producing grammatically flawless, professional-quality summaries. Minor deductions in Content (4.3) and Factual consistency (4.0) were due to slight emphasis differences rather than errors.
- **Llama instability:** Llama showed inconsistent quality—one sample was completely incoherent (starting mid-sentence with "ot Mary Keitany"), while others were acceptable. This variability (scores ranging 1-5 across samples) averaged to moderate performance (2.7-3.3), indicating architectural mismatch with the summarization task.
- **Pre-training gap:** The 3.3-4.3 point difference between pre-trained and from-scratch models in human evaluation (compared to automatic metrics) underscores that fluency and coherence failures are more severe than ROUGE scores suggest. Training from scratch results in not just lower performance, but fundamentally broken generation.

4.7 Qualitative Analysis

Table 5 shows example outputs from all models.

Pre-trained models produce coherent, accurate summaries. BART closely matches the reference in tone and content. T5-Base captures key information despite using LoRA. Llama shows instability with repetitive phrases. MoE models struggle with basic coherence and sometimes generate ungrammatical text—highlighting the challenges of training language models from scratch.

Table 5: Example summaries from different models. The reference is the human-written gold summary.

Source	Prison Link Cymru had 1,099 referrals in 2015-16 and said some ex-offenders were living rough for up to a year before finding suitable accommodation...
Reference	There is a "chronic" need for more housing for prison leavers in Wales, according to a charity.
BART	There is a "desperate need" for housing for former prison leavers in Wales, a charity has said.
T5-Base	Prison leavers in Wales are living rough for up to a year without housing, a charity has said.
Llama-1B	He said: "I've been in and out of the system for 20 years..." [repetitive, incoherent]
MoE-Hash	A number of people who have been given a "a number of women" in Wales...
MoE-TopK	A charity has been criticised for a "a number of people" in Wales...

5 Discussion

5.1 The Routing Trade-off

Results reveal a fundamental trade-off between the two routing strategies:

Top-K Routing optimizes for raw performance:

- Achieves lower validation loss (4.34 vs 4.39)
- Learns expert specialization naturally
- More flexible and adaptive to inputs
- But: Load imbalance (44-57% range) complicates deployment

Hash Routing optimizes for operational simplicity:

- Perfect load distribution (50% per expert)
- No learned parameters required
- Predictable and stable behavior
- But: Cannot adapt to input

Surprisingly, hash routing achieved higher ROUGE scores despite worse training loss. My hypothesis is that perfect load balancing ensures all experts receive equal gradient updates throughout training, preventing the under-trained expert problem that can affect Top-K routing. This suggests that for tasks where generalization is critical, benefits of balanced training may outweigh advantages of adaptive routing.

5.2 The Critical Role of Pre-training

Performance gap between pre-trained and from-scratch models is substantial:

- BART vs MoE-Hash: 102% improvement (0.451 vs 0.223 ROUGE-1)

- T5-Base vs MoE-Hash: 46% improvement (0.325 vs 0.223)
- Even Llama's single-epoch adaptation: 6% improvement (0.236 vs 0.223)

This demonstrates that even sophisticated architectural innovations like MoE cannot compensate for lack of pre-training on large-scale corpora. Pre-trained models benefit from:

- Rich semantic representations learned from billions of tokens
- Understanding of linguistic structure and world knowledge
- Initialized weights already close to good solutions

My MoE models, trained from scratch on only 50M tokens, had to simultaneously learn:

- Word embeddings from scratch
- Basic syntactic patterns
- Semantic relationships
- Summarization strategies
- Expert routing on top of all that

This multi-objective optimization problem likely explains their lower performance.

5.3 Model Size and Architecture

Interestingly, Llama-1B (1.2B parameters) only marginally outperformed my 16M parameter MoE models. This challenges the assumption that more parameters automatically mean better results. Several factors are at play:

1. **Architecture mismatch:** Causal LMs are designed for next-token prediction, while summarization benefits from bidirectional encoding
2. **Training regimen:** Llama received only 1 epoch of adaptation, likely insufficient for proper domain transfer
3. **Sequence length constraints:** I had to limit Llama to 256 tokens due to memory constraints, which likely hurt performance

The fact that MoE-Hash surpassed Llama on ROUGE-L (0.180 vs 0.171) suggests well-designed smaller models can compete with much larger models on specific metrics when architectural choices align with the task.

5.4 Bonus: Impact of Load Balancing Loss

To investigate load balancing's role, I trained an additional MoE-TopK model *without* the auxiliary load balancing loss ($\alpha = 0$) and compared it against the original MoE-TopK model trained *with* load balancing ($\alpha = 0.01$).

Table 6: Impact of load balancing loss on MoE-TopK performance. Removing LB loss improves training loss but degrades generation quality.

Metric	With LB Loss	Without LB Loss
Validation Loss	4.3395	4.3322
ROUGE-1	0.2121	0.2043 (-3.7%)
ROUGE-2	0.0459	0.0424 (-7.6%)
ROUGE-L	0.1700	0.1609 (-5.4%)
BLEU	0.0247	0.0210 (-15.0%)
BERTScore-F1	0.8375	0.8332 (-0.5%)

5.4.1 Performance Comparison

Table 6 shows the impact of removing load balancing loss.

5.4.2 Expert Utilization Analysis

Surprisingly, removing load balancing loss did *not* result in the extreme expert imbalance I expected:

- **With LB:** Expert usage ranged from 44-57% (13% spread)
- **Without LB:** Expert usage ranged from 42-58% (16% spread)

Expert distribution remained relatively balanced even without explicit load balancing constraints, with only a 3% increase in maximum spread. However, decoder layer 1 showed the most pronounced imbalance increase (13% $\rightarrow$ 16% spread).

5.4.3 The Generation Quality Paradox

The most striking finding is a **performance paradox**: removing load balancing loss improved training loss (4.3322 vs 4.3395, -0.17%) but significantly degraded generation quality across all ROUGE metrics (3.7-15.0% decline). Qualitative analysis revealed the cause:

Sample outputs without load balancing exhibited severe **token repetition**:

- *"...not being treated as a 'a-year-old girl'. to to to to to to to to to to to to to"*
- *"...found in a house in Glasgow. in the city. to to investigating. investigating investigating investigating..."*

These degenerate outputs suggest the model without load balancing discovered a pathological solution that minimizes loss through repetitive patterns without producing meaningful summaries.

5.4.4 Load Balancing as Regularization

My findings reveal that load balancing loss serves a dual purpose beyond its intended goal of computational efficiency:

1. **Intended effect:** Ensures balanced expert utilization for predictable deployment

2. **Unexpected regularization:** Prevents the model from exploiting repetitive patterns that minimize training loss but produce low-quality outputs

This regularization effect is particularly critical for text generation tasks, where training loss alone is an insufficient quality indicator. Load balancing loss acts as an implicit constraint that guides the model toward more diverse expert usage patterns, which correlates with more varied and higher-quality text generation.

Conclusion: For MoE models in text generation, load balancing loss is essential both for operational balance *and* for preventing degenerate solutions. This finding has important implications for practitioners: even if expert imbalance doesn't emerge, regularization benefits of load balancing loss significantly improve generation quality.

5.5 Practical Implications

For practitioners considering MoE for summarization, these findings suggest:

1. **Start with pre-training:** Rather than training MoE from scratch, initialize experts from pre-trained models
2. **Choose routing strategically:**
    - Use hash routing when deployment simplicity and load predictability are priorities
    - Use Top-K when maximum performance is needed and infrastructure can handle imbalance
3. **Always use load balancing loss:** Even if prioritizing learned specialization, regularization benefits outweigh potential performance gains from removing it
4. **Monitor expert usage:** Visualizing expert utilization (like Figures 3 and 4) helps diagnose training issues
5. **Consider encoder-decoder models:** For summarization specifically, encoder-decoder architectures consistently outperform causal LMs

5.6 Limitations

Several limitations affect interpretation of these results:

1. **Model scale:** My 16M parameter MoE models are much smaller than typical production models. Scaling experiments would provide additional insights.
2. **Training data:** Only XSum (204k examples) was used. Pre-training on much larger corpora before fine-tuning might substantially improve MoE performance.
3. **Architectural choices:** Many hyperparameters (2 layers, 4 experts, dimension 96) were fixed due to computational constraints. Optimal configuration remains an open question.
4. **Single task:** Results might differ on other summarization datasets (like CNN/DM or PubMed) or entirely different NLP tasks.

5. **Hardware constraints:** Being limited to 8GB VRAM meant I couldn't explore larger batch sizes, FP16 training (which caused instabilities), or models beyond 1B parameters.

6. **LLM-as-judge evaluation:** While the assignment suggested using SummaC for automated factuality checking, I encountered significant dependency conflicts during installation that would have required downgrading critical libraries. Given time constraints and availability of comprehensive human evaluation data, I focused on manual factual consistency assessment. Future work should explore more recent factuality metrics like AlignScore or updated versions of SummaC compatible with modern library versions.

6 Conclusion

This work provides a comprehensive comparison of MoE routing strategies for abstractive summarization. Key findings include:

1. **Routing strategies involve fundamental trade-offs:** Top-K routing achieves better optimization through learned specialization but creates deployment challenges through load imbalance. Hash routing guarantees perfect load distribution and achieves surprisingly competitive performance despite being simpler.

2. **Pre-training is critical:** Even sophisticated architectures like MoE cannot bridge performance gaps created by training from scratch. Pre-trained models outperform from-scratch models by 40-100% across metrics.

3. **Model size isn't everything:** A well-designed 16M parameter model with hash routing can match or exceed a 1.2B parameter model on specific metrics when architectural choices align with the task.

4. **Expert specialization emerges naturally:** Top-K routing learns distinct usage patterns without explicit supervision, with certain experts handling 30% more tokens than others despite imposed load balancing losses.

5. **Load balancing loss is essential:** My bonus experiment revealed that load balancing loss serves as a critical regularizer, preventing degenerate solutions that minimize training loss through repetitive token generation while maintaining reasonable expert balance.

References

[1] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. *arXiv preprint arXiv:2112.06905*, 2021.

[2] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[3] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Efficient nearest neighbor language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5703–5714, 2021.

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[5] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[6] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. *arXiv preprint arXiv:2103.16716*, 2021.

[7] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[8] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[9] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

[10] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[12] Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models. *arXiv preprint arXiv:2106.04426*, 2021.

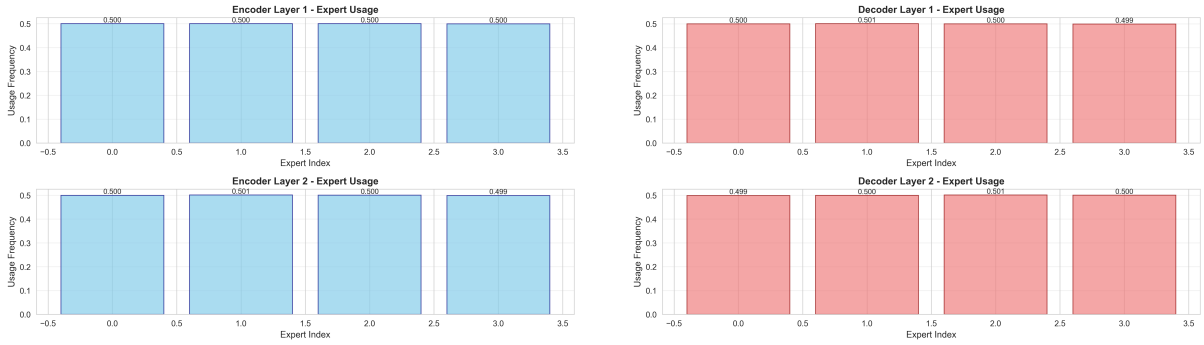
[13] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[14] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Detailed Model Comparison

Model	rouge1	rouge2	rougeL	bleu	bertscore-f1
BART	0.4514	0.2200	0.3692	0.1365	0.9159
T5-Base	0.3245	0.1048	0.2516	0.0536	0.8918
Llama-1B	0.2359	0.0666	0.1712	0.0277	0.8644
MoE-TopK	0.2121	0.0459	0.1700	0.0247	0.8375
MoE-Hash	0.2230	0.0487	0.1798	0.0267	0.8515

Figure 5: Detailed numerical comparison of all models across metrics.



(a) Hash: Encoder expert usage

(b) Hash: Decoder expert usage

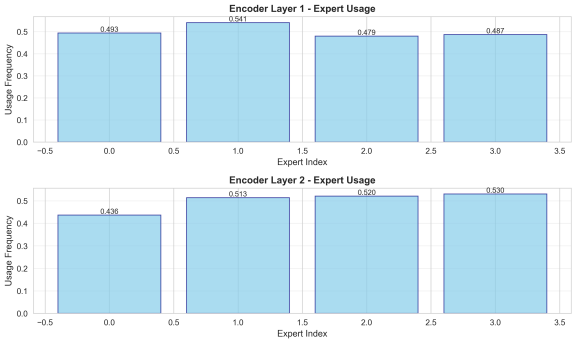
Figure 6: Bar charts showing perfect uniform distribution in hash routing.

A Additional Visualizations

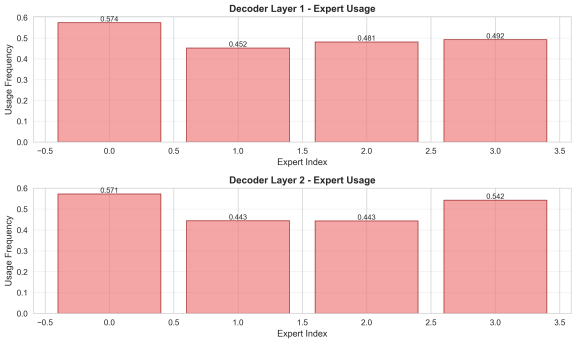
B Reproducibility and Code

B.1 Repository Structure

All code and trained models are available at: https://github.com/ANLP2025/assignment3-Geekonatrip123



(a) Top-K: Encoder expert usage



(b) Top-K: Decoder expert usage

Figure 7: Bar charts showing varied utilization in learned Top-K routing.