ICASSP 2026 Oral

Adaptive Shared Experts with LoRA-based MoE for Multi-Task Learning

Shared experts should help the STL→MTL transition — without dominating the mixture.

MTL MoE LoRA

Minghao Yang · Ren Togo · Guang Li · Takahiro Ogawa · Miki Haseyama
Hokkaido University

Talk flow

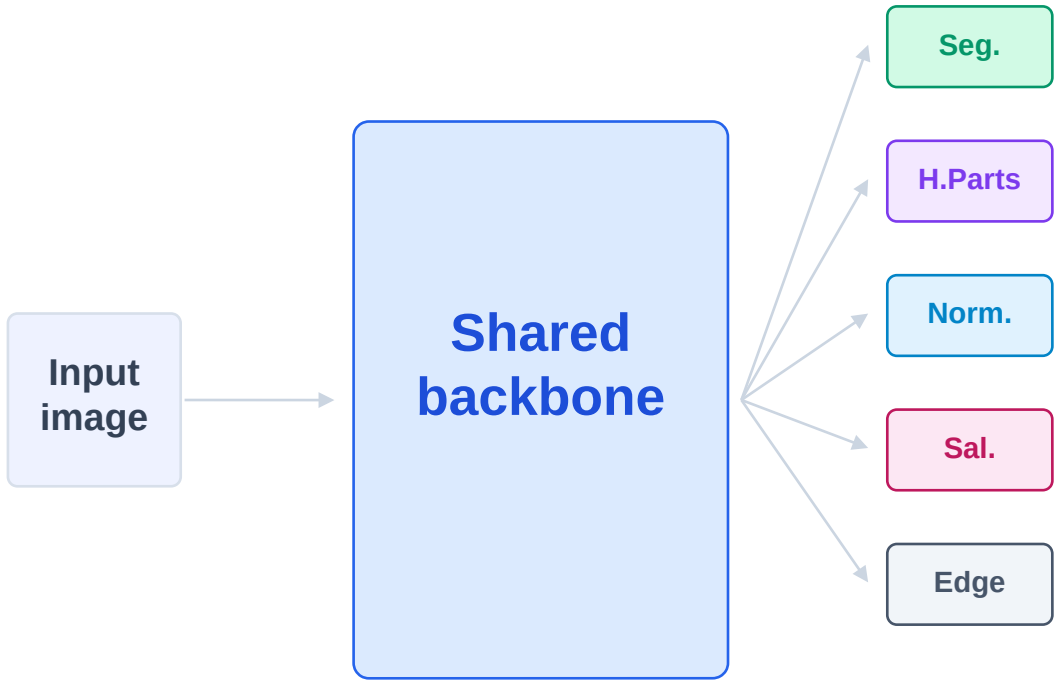
1. MTL framing
2. Problem
3. ASE method
4. Results

Audience context: mostly MoE-related talks
→ keep the MTL framing concise and spend time on the contribution.

16 min

Multi-Task Learning (MTL), in one slide

One shared backbone, multiple task heads



The core question is not just “share or not share” —
it is what to share, where, and for whom.

Definition

MTL jointly trains different tasks with a shared backbone and task-specific heads, aiming to exploit both common representations and task-specific specialization.

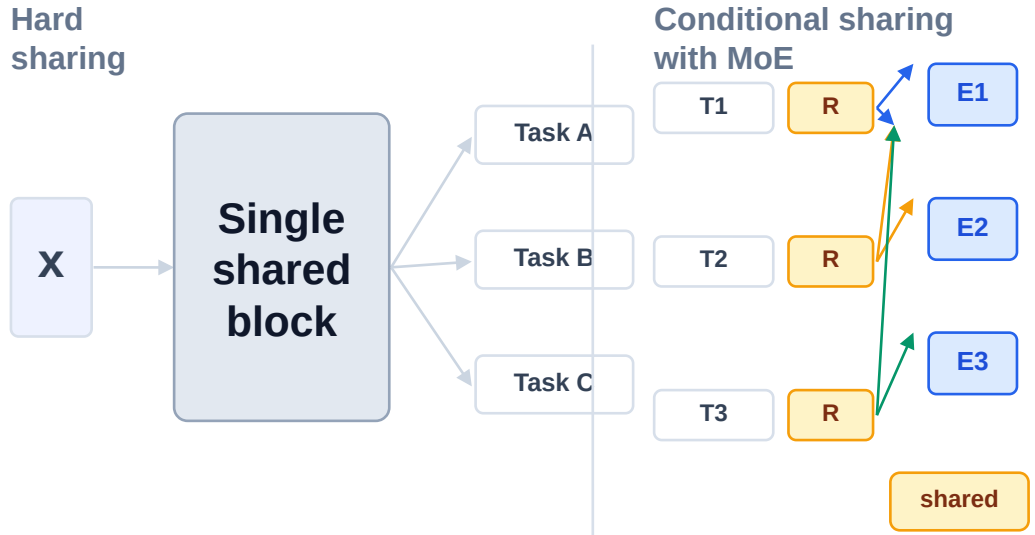
Why it matters here

- Dense prediction tasks overlap, but not perfectly.
- Full hard sharing is efficient, yet negative transfer is real.
- So MTL needs a better sharing mechanism than one monolithic backbone.

MTL is fundamentally a sharing problem

How MoE fits MTL

Motivation



MoE expands capacity while keeping computation manageable by activating only a small subset of experts per input.

- Experts are shared in parameters, but selected conditionally.
- Task-specific routers make this especially attractive for MTL.
- The real question for this paper is how to adapt efficiently from STL to MTL.

Conditional sharing is why MoE is appealing for MTL

The issue: inefficient STL → MTL adaptation

Problem

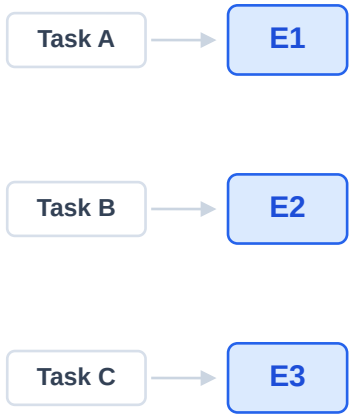
With sparse activation, each expert can end up relearning the same STL→MTL bridge on its own.

1. Start from STL

STL-pretrained backbone

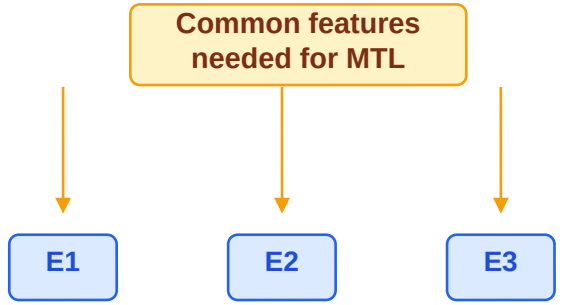
Great initialization, but it still carries a task-specific prior.

2. Switch to sparse MoE-MTL



Sparse experts see different slices of the transition, so overlap is learned repeatedly instead of once.

3. What goes wrong



The same bridge gets relearned many times.

- Redundant adaptation
- Weak task-agnostic sharing
- Less room for later specialization

Why naive shared experts still fail

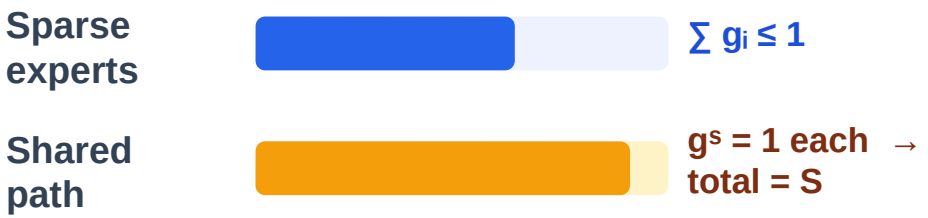
Problem

Adding a shared expert is intuitive: let one path capture common knowledge for all tasks.

Naive shared expert

$$y = x + W_0 x + \sum_{i \in T} g_i E_i(x) + \sum_{i=1}^{S} E_i^s(x)$$

Equivalent gating picture



Fixed shared contribution can dominate the mixture

What this breaks

- Imbalance between sparse and shared experts.
- Scale mismatch between frozen FFN weights and LoRA expert updates.
- Later in training, the all-task shared path amplifies gradient conflicts.

$$\sum_{i \in T} g_i + \sum_{i=1}^{S} g_i^s > S \quad \text{if } g_i^s = 1$$

This is why “just add a shared expert” is not enough.

Our answer: Adaptive Shared Experts (ASE)

Method

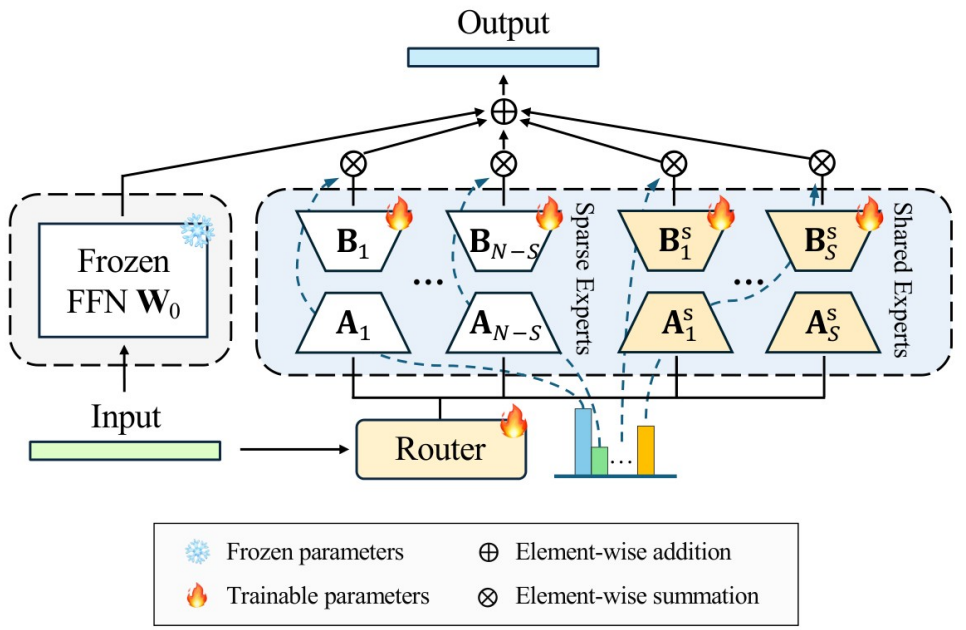
Core idea

- Compute sparse and shared logits separately.
- Select top-(k−S) sparse experts, then normalize sparse + shared together.
- Let shared experts help more early, then naturally step back later.
- Keep expert modules lightweight with LoRA.

Balance condition

$$\sum_{i \in T} g_i + \sum_{i=1}^{S} g_i^s = 1$$

Shared experts participate adaptively — not with a fixed vote

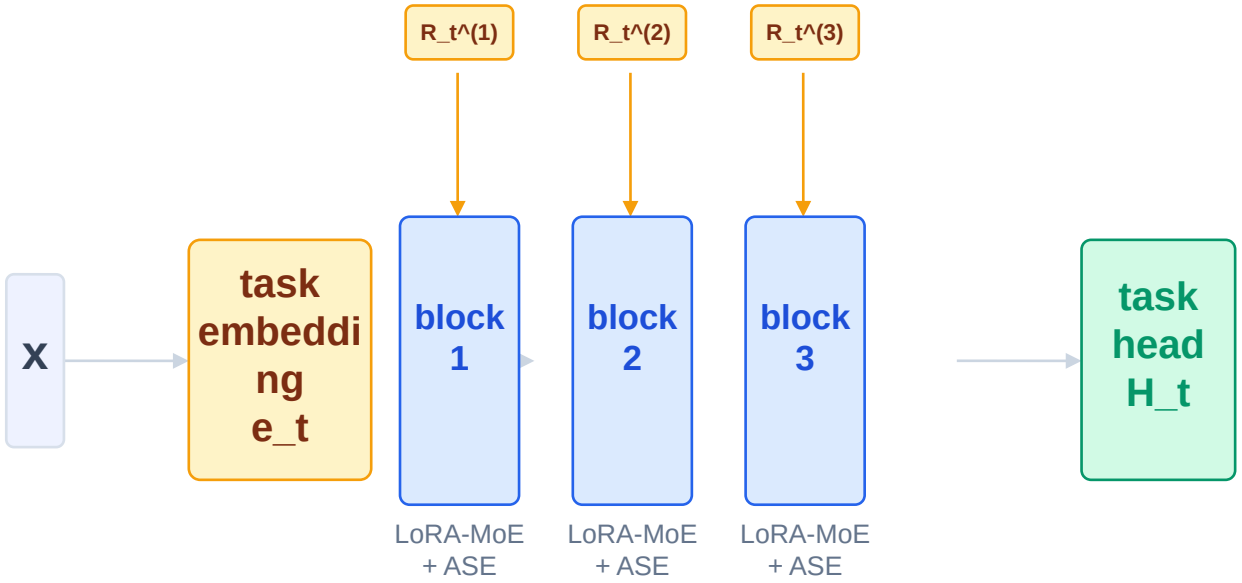


Shared path **Sparse path** **Joint normalization**

From block-level ASE to the full MTL system

Method

Task-specific routing on top of a shared ViT backbone



Experts are shared in parameters, but routing is task-dependent. The backbone is STL-initialized and fine-tuned with LoRA; routers and heads are trained in full-parameter mode.

Fine-grained experts under a similar budget

Increase the number of experts and proportionally decrease LoRA rank.

(16 / 3 / 1 / 4)
Base setup

(32 / 6 / 2 / 2)
Medium granularity

(64 / 12 / 4 / 1)
High granularity

More experts + lower rank = finer specialization

Notation: (N / k / S / r) = total experts / activated sparse experts / shared experts / LoRA rank.

Experimental setting

Experiments

Benchmark

PASCAL-Context

10,103 images · 5 dense prediction tasks

Seg. | H.Parts | Norm. | Sal. | Edge

Metrics

- mIoU for Seg., H.Parts, Sal.
- mErr for Norm. (lower is better).
- odsF for Edge in main comparison; BCE in ablations.

Base training setup

- ViT backbone pretrained from STL.
- 16 LoRA experts per FFN layer, rank r = 4.
- 1 adaptive shared expert; top-k = 3.
- 224 × 224 input resolution; 40 epochs.
- Single NVIDIA RTX 6000 Ada GPU.

Unified training setting

Evaluation summary

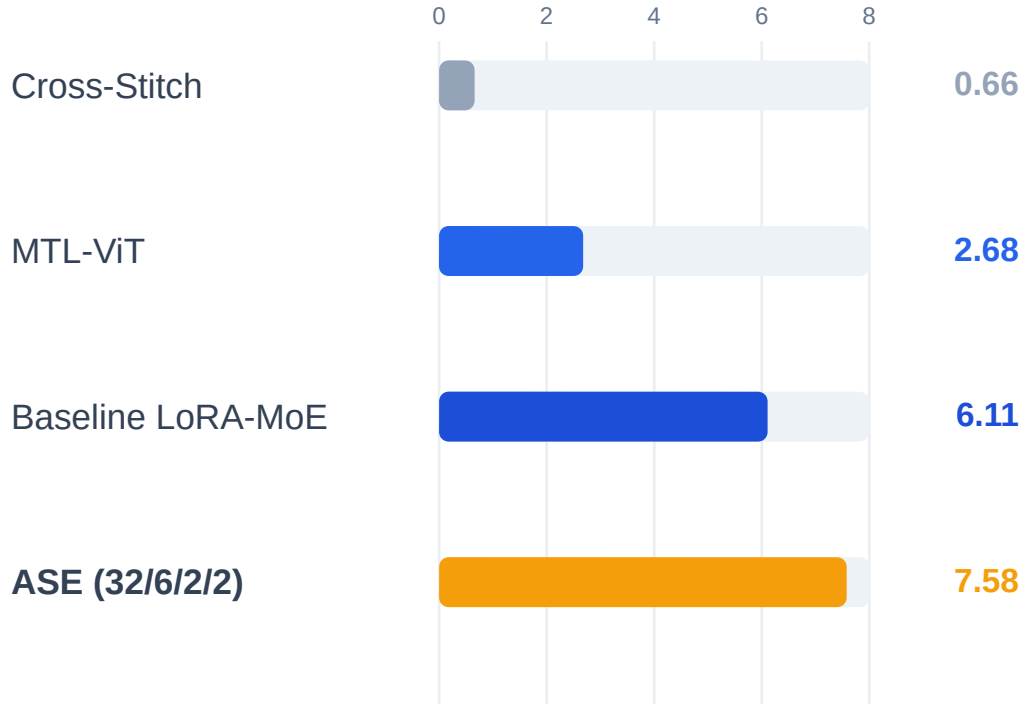
Δm

aggregate relative gain / loss vs. STL across all tasks

- Higher Δm is better.
- Router regularization uses Mod-Squad loss.
- Main comparisons report odsF for Edge; ablations use BCE for speed.

Main results on PASCAL-Context

Average relative performance Δm (%)



ASE adds +1.47 Δm over the strongest LoRA-MoE baseline

Selected metrics from Table 1

Table 1. Comparisons with MTL methods on the PASCAL-Context dataset. Best results are highlighted in **bold**.

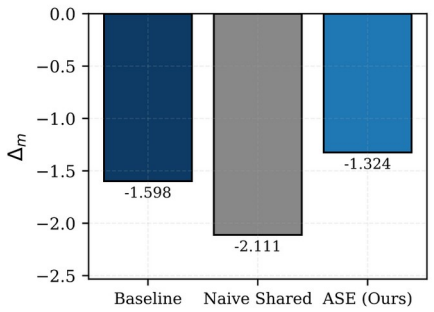
Method	Backbone	*Seg.* (mIoU)↑	*Norm.* (mErr)↓	*H. Parts* (mIoU)↑	*Sal.* (mIoU)↑	*Edge* (odsF)↑	Δ_m (%)↑	Params (M)↓
STL	ResNet-18	60.1	**15.4**	51.0	63.2	50.8	0.00	**11**
MTL	ResNet-18	58.4	15.9	51.8	62.6	50.3	-1.29	**11**
MTAN [25]	ResNet-18	58.9	16.4	51.9	62.2	50.6	-1.74	**11**
NDDR-CNN [26]	ResNet-18	59.7	15.5	51.7	62.7	51.5	+0.13	**11**
Cross-Stitch [27]	ResNet-18	60.4	**15.4**	51.6	63.1	51.7	+0.66	**11**
MTL-ViT	ViT-base	69.1	16.2	54.8	61.9	49.9	+2.68	104
Baseline (16/4/0/4)	LoRA-MoE ViT-base	73.7	17.5	59.2	62.9	53.7	+6.06	107
Baseline (32/8/0/2)	LoRA-MoE ViT-base	73.8	17.4	59.3	62.9	53.3	+6.11	108
Ours (16/3/1/4)	LoRA-MoE ViT-base	**74.0**	17.3	60.1	63.2	**55.3**	+7.49	107
Ours (32/6/2/2)	LoRA-MoE ViT-base	**74.0**	17.2	**60.3**	**63.3**	54.9	**+7.58**	108

- LoRA-MoE already improves strongly over vanilla MTL-ViT.
- ASE further improves the strongest MoE baseline consistently.
- Best overall setting in the paper: ASE (32/6/2/2).

≈4% params vs. plain ViT baseline

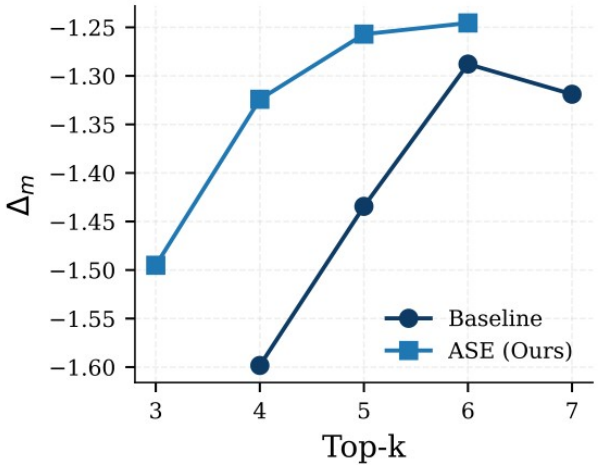
Ablations: ASE helps, naive shared hurts

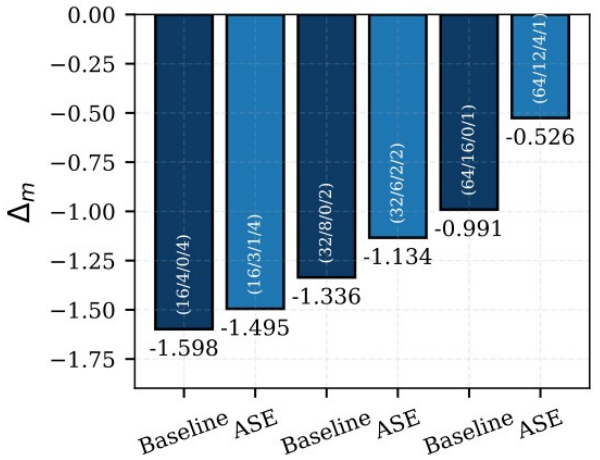
Naive shared expert vs. ASE



Directly adding a shared expert is worse than baseline

Robustness across top-k and expert granularity

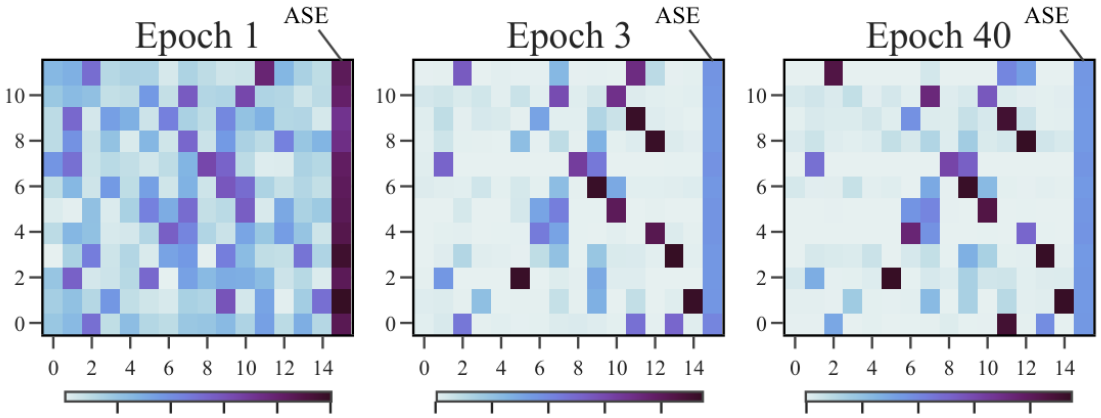




- ASE stays above the baseline across all tested top-k values.
- Fine-grained expert settings improve both baseline and ASE, but ASE stays better.
- The gain becomes most pronounced at higher expert granularity.

Training dynamics match the design intuition

Activation frequency across layers for task Norm.



The ASE column is the last one in each heatmap. It is strong early, then becomes less dominant as sparse experts specialize.

Read this figure as a timeline

- Epoch 1: ASE dominates many layers and bridges the STL → MTL transition.
- Epoch 3: sparse routing becomes sharper and more task-specific.
- Epoch 40: expert selection stabilizes with clearer specialization.

ASE is most useful early — then it steps back

Conclusion and future directions

Takeaway

Takeaways

- The STL → MTL transition is a real bottleneck in MoE-based MTL.
- Naive shared experts are imbalanced because their contribution is fixed.
- ASE fixes this with adaptive routing and joint normalization.
- LoRA keeps the design efficient, and fine-grained experts help further.
- Best reported setting: ASE (32/6/2/2) reaches $\Delta m = +7.58$ on PASCAL-Context.

Natural next steps

- Scale to larger backbones and richer task mixtures.
- Explore stronger conflict-aware router regularization.
- Study how many shared experts should be active at different stages.
- Transfer the idea beyond dense vision tasks — e.g., audio or multimodal MTL.

Final message: shared experts should collaborate, not dominate