Can you carve a task-specific model out of a Mixture-of-Experts model?

Kabui, Charles

2026-08-02

Table of Contents

Figure 1: Mixture-of-Experts model

What we are really asking

A Mixture-of-Experts model, MoE for short, replaces the feed-forward part of each Transformer layer with several small networks called experts, and adds a router that sends each token to a few of them. Only a few experts compute for any given token, so the model runs fast. But every expert still has to be stored, and with dozens of experts per layer and dozens of layers, the model file becomes enormous.

Now suppose you only care about one job, such as reading contracts, generating code, or calling tools. A tempting idea is to keep only the experts that matter for that job, delete the rest, and ship a small model. This article asks whether that works without the two usual ways of shrinking models.

Distillation trains a new, smaller student model to imitate the big teacher. It needs extra training runs, and the result is a brand new model, not something found inside the original. Quantization stores the same weights in fewer bits. It keeps every expert; it just makes each one smaller. Both are useful, and this article returns to them at the end. But the sharper question is whether an existing MoE already contains a clean, task-specific sub-model that you can physically carve out.

The short version of the answer: yes for a pruned and fine-tuned path through the model, no for a single self-contained “legal expert” you can lift out whole.

Two notes before we go. First, by vertical I mean a model focused on one domain or workflow, such as legal work, medicine, code, or tool calling. That is the sense used throughout this article. Second, some papers use "vertical MoE" for a completely different idea, routing across layers, and that is covered in its own section so the two never get confused.

Why a sparse model is still huge on disk

An MoE gives you two different things to keep straight:

- Active computation: the weights actually used for a token.
- Stored model: every weight the router might ask for later.

Sparse activation cuts arithmetic. It does not cut storage. If you are not allowed to distill or quantize, the only way to shrink the file is to delete parameters or make the network narrower.

Done carefully, the surgery looks like this:

1. Run task data through the model and record which experts the router picks and how much each expert's output contributes.
2. Score experts, or groups of experts.
3. Delete whole expert tensors, not just skip calling them.
4. Rebuild the router so it only knows about the survivors.
5. Keep the shared parts (attention, embeddings, output head) unless you can show they can go too.
6. Fine-tune what is left on the task.

Masking an expert at runtime does not count. If the deleted weights still ship in the checkpoint, you saved no memory.

There is a catch with the shared parts. A task capability usually lives across the router, attention, embeddings, output head, and several layers of experts. So "the legal expert" is a hypothesis to test with ablations, not a fact to assume. The papers below keep coming back to this point.

Route 1: Prune while you fine-tune

The strongest published result for this route is Task-Specific Expert Pruning. The authors fine-tuned an MoE on a downstream task and progressively dropped experts whose contribution fell below a threshold. By about halfway through training, each layer was down to one surviving expert, and they kept training it.

The result: the single-expert-per-layer model kept 99.3% of the MoE's benefit across six task types and ran about 2x faster. The detailed numbers show a 2x speedup on MNLI and 1.28x on SQuAD2.0. It also removed the cross-device communication that made the original sparse model slow, and the paper notes it needs no extra distillation.

That sounds like a clean win, but read the limits:

- The model is fine-tuned while experts are removed, so this is not a free extraction.
- The tests used an older BERT-scale MoE, not a modern frontier model.
- One expert survives per layer, so the result is a path, expert A at layer 1, expert B at layer 2, and so on, wrapped in the shared Transformer. It is not one portable expert.
- The paper does not show that an untouched expert from an arbitrary modern MoE works on its own.

A 2024 EMNLP study supports the selection half of this. It found that routing for a single task is highly concentrated, that different tasks use different routing patterns, and that fine-tuning only the relevant experts while freezing the rest works well, often matching full fine-tuning. But it froze the other experts; it did not delete them. That is evidence for task-specific combinations, not for a portable expert tensor.

Route 2: Prune in one shot, no re-training

What if you do not want to fine-tune at all? REAP is the strongest test of that. Its Router-weighted Expert Activation Pruning score combines the router's gate values with expert activation norms, rather than relying on usage frequency alone. It compresses without any fine-tuning after pruning.

REAP was tested across sparse MoE models from 20B to 1T parameters at 25% and 50% expert compression. After pruning half the experts from Qwen3-Coder-480B and Kimi-K2, code generation stayed near-lossless. The paper also evaluates tool use and agentic coding. Calibration data matters a lot: calibrating on the right domain keeps quality high, while calibrating on mismatched generic data can make the compressed model produce incoherent code.

But there is an important limit. REAP gives you a smaller MoE, not a vertical dense model. The surviving experts still depend on the original router and the shared layers. So it answers one question, can we delete half the expert pool and keep useful behavior, and leaves another open, is there one expert that already contains the vertical.

MoE-I2 pushes in the same direction, pruning between experts and then applying low-rank decomposition inside the survivors. It reports zero-shot results on Qwen1.5-MoE, DeepSeek-V2-Lite, and Mixtral. It adds a low-rank approximation step, and like REAP it does not establish that any one expert owns the vertical.

One warning: usage frequency alone is not a safe selector. A frequently routed expert may carry syntax or general features every domain needs. Use task ablations, activation contribution, and held-out data together.

Route 3: Train a vertical MoE on purpose

If the vertical is a real product requirement, training the structure is more defensible than pretending a general checkpoint already contains clean vertical modules. Three approaches stand out.

FlexOlmo starts from a public anchor model. Each data owner trains only their own expert feed-forward module on private data while the shared trunk, attention and the public expert, stays frozen. The experts are combined by concatenating per-expert router embeddings, and each can be included or excluded at inference time. This is close to a vertical model assembled from several private subdomains. But it is a pretraining result, and it does not prove that freezing the whole trunk works for post-training behaviors such as tool syntax or safety.

BAR works after pretraining. It trains math, code, tool-use, and safety experts separately, then merges them with a frozen anchor expert that preserves the base model, into a five-expert MoE. No distillation or quantization. BAR also exposes the limit of the "one expert per vertical" story. In its tool-use test, freezing the embeddings and the language-model head gave a Berkeley Function Calling Leaderboard score of 20.3. Unfreezing those shared parts raised it to 46.4, because function calling adds special tokens and output formats. The later reinforcement-learning stage unfreezes attention as well. So a practical vertical MoE needs an explicit plan for shared embeddings, attention, the output head, and safety, not just the expert feed-forward networks.

EMO makes modularity a pretraining objective. It trains a 1B-active, 14B-total model on 1T tokens, and constrains tokens from the same document to use a shared expert pool. Keeping only 25% of the experts costs about 1% absolute performance; keeping 12.5% costs about 3%. A standard MoE falls apart under the same restriction. EMO is the cleanest demonstration

that selective expert deployment can be designed into the model. It is also the most expensive route, and it does not turn an existing MoE into a modular one.

Why you can't just copy out "the legal expert"

The numbers above can sound like proof that a big model secretly contains a compact specialist you can lift out. That is not what the papers show.

The 99.3% result is a path, one expert per layer plus the shared Transformer. REAP's survivors still need the original router and shared layers. BAR's tool-use results show that the embeddings and the output head carry part of the skill. Nothing in this literature demonstrates a single expert tensor, pulled from an arbitrary modern MoE, that works on its own as a legal or tool-calling model.

If you want to test the idea yourself, treat "the legal expert" as a hypothesis, run an ablation, and be ready to find that the capability lives in the combination rather than in one tensor.

The other meaning of "vertical MoE"

Some papers use "vertical MoE" for routing across layers rather than across the experts inside one layer. The model chooses which layer-like transformation to use at a given depth, an idea related to Mixture-of-Depths and dynamic-depth models.

MoLEx is a clear example. It upcycles a pretrained dense model by reusing its trained layers as experts. At each layer, a router selects another layer's transformation and mixes it with the current one using a small gate and a mixing weight, then fine-tunes. It shows a dense model can be turned into a layer-expert architecture without distillation or quantization. But it targets compositional representations, not domain specialization, and it does not make the model smaller, since it reuses the existing layers and adds routing.

So there are two different questions:

- Vertical application: can the model specialize in law, medicine, code, or tool use?
- Vertical depth routing: can each token skip or reuse layers?

Do not cite a depth router as evidence that a legal expert exists. They optimize different axes.

What I would actually do

For a single vertical, I would not start by training a large MoE. A dense vertical model is the simpler baseline. MoE becomes worthwhile when the vertical has distinct subskills, when modules need to be updated independently, or when several verticals share a base while staying removable.

For an existing open MoE, I would run task-pruned fine-tuning first. It is the smallest experiment that can falsify the idea:

- keep the shared trunk;
- pick expert combinations with held-out ablations;
- delete experts by layer;
- fine-tune the surviving path;
- check whether tool behavior or citations require shared-parameter updates.

If the no-distillation, no-quantization constraint holds, the best post-hoc experiment is:

1. Choose an open MoE with accessible expert weights and router logits.
2. Define the vertical as a testable contract, such as cited document answers and schema-valid tool calls.
3. Collect separate calibration and test sets.
4. Compare selectors: usage frequency, REAP-style saliency, and task ablations.
5. Prune 25%, 50%, and the most aggressive per-layer survivor.
6. Rebuild the router and physically delete the removed tensors.
7. Fine-tune only the surviving experts first.
8. Then test whether the router, embeddings, output head, or attention need unfreezing.
9. Measure the final artifact on the target device.

The main table should look like this:

Model	Expert weights shipped	Task score	Citation or tool score	General score	Peak memory
Original MoE	100%	baseline	baseline	baseline	baseline
One-shot pruned MoE	selected subset	measure	measure	measure	measure
Task-pruned and fine-tuned MoE	selected subset	measure	measure	measure	measure

Model	Expert weights shipped	Task score	Citation or tool score	General score	Peak memory
Dense vertical baseline	separate model	measure	measure	measure	measure

Keep distillation and quantization out of the primary table and add them later as controls.

Distillation can produce a dense student, but it needs teacher runs, student training, and a new failure mode, and it cannot answer the extraction question, since the student is a new model rather than proof that the teacher contained the vertical. The 2026 MoE-to-dense study shows the conversion works, but it is no longer pure extraction.

Quantization lowers the byte count but leaves the expert pool and router intact. REAP itself stacks the two methods, pruning 50% of experts from already-quantized FP8 and 4-bit checkpoints, and a pure 4-bit quantized model beats a 16-bit model with half its experts removed on EvalPlus. Per byte, quantization is often the cheaper win. Per architecture, it never deletes an expert.

The decisive check is physical. Load the final checkpoint with the deleted experts absent. A runtime mask, an offloaded teacher, or a copy of the original experts on disk does not count.

Conclusion

Without distillation or quantization, yes, an MoE can be torn down into a smaller task model. The demonstrated routes are task-specific pruning during fine-tuning, or one-shot pruning when a smaller MoE is enough.

What has not been demonstrated is a reliable, zero-shot way to extract one portable expert from an arbitrary modern MoE. The usable unit is usually a per-layer combination plus the shared trunk.

If the vertical matters, train for it. FlexOlmo shows experts trained against a shared anchor. BAR shows post-training verticals may need shared embeddings, attention, and output-head updates. EMO shows how to make expert subsets independently usable from pretraining. MoLEx shows the separate depth-routing meaning of vertical.

Distillation and quantization remain useful alternatives. They solve different problems. Neither answers whether the original model contains a clean vertical that can simply be carved out.

References

1. Chen, T., et al. (2022). *Task-Specific Expert Pruning for Sparse Mixture-of-Experts.* arXiv:2206.00277.
2. Wang, Z., et al. (2024). *Let the Expert Stick to His Last: Expert-Specialized Fine-Tuning for Sparse Architectural Large Language Models.* EMNLP 2024.
3. Yang, C., et al. (2024). *MoE-I2: Compressing Mixture of Experts Models through Inter-Expert Pruning and Intra-Expert Low-Rank Decomposition.* Findings of EMNLP 2024.
4. Teo, R. S. Y., et al. (2025). *MoLEx: Mixture of Layer Experts for Finetuning with Sparse Upcycling.* arXiv:2503.11144.
5. Shi, W., et al. (2025). *FlexOlmo: Open Language Models for Flexible Data Use.* arXiv:2507.07024.
6. Lasby, M., et al. (2025). *REAP the Experts: Why Pruning Prevails for One-Shot MoE Compression.* arXiv:2510.13999.
7. Wang, R., et al. (2026). *EMO: Pretraining Mixture of Experts for Emergent Modularity.* arXiv:2605.06663.
8. Kim, J., et al. (2026). *Pruning and Distilling Mixture-of-Experts into Dense Language Models.* arXiv:2605.28207.
9. Morrison, J., et al. (2026). *Train Separately, Merge Together: Modular Post-Training with Mixture-of-Experts.* arXiv:2604.18473.
