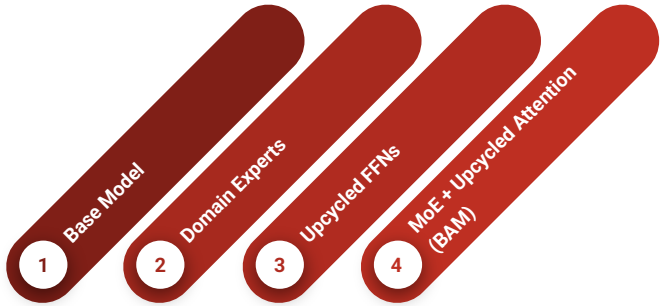
Routers with Semantic Grounding for Upcycled Experts

Zardar Khan (2025)

Context: BAM builds on BTX

- BTX: Upcycle FFNs -> MoE
- BAM: FFNs (content) + Attention (algorithms)
- Contributions:
  - Mixture-of-Attention
  - Soft routing
  - KV-sharing



My Setup: GPT-2 small

Chose GPT-2 small -> small enough to iterate, large enough to show patterns

- ➔ **4 domain experts**
  Law, Code, Math, General (C4).
- ➔ **~12M tokens each**
  10% general
- ➔ **Perplexity dropped**
  Confirming specialization

Building the MoE

- Extracted FFNs from each expert
- Combined with base **GPT-2 attention**
- Router = **vanilla linear**
- Training data: 25% from each domain

Observations

- MoE training: perplexity decreases across domains
- Router learned useful dispatching
- Next step: compare base vs MoE on mixed dataset

Limitations: Linear Router

- Linear router = domain matching only
- I.e. legal tokens -> law expert
- **Misses cross-domain skills**
  - Math expert (trained on word problems) could help with **legal reasoning**

—

- Add learnable expert embeddings (“cue cards”)
- Encodes nuances capabilities: symbolic reasoning, proofs, optimization
- Router = match hidden state (“meaning cloud”) to expert embeddings

Methodological Approach

- Open question: how to init. Embeddings?
  - Random init (baseline)
  - Profiling warm-start (per-token losses)
- Risks = redundancy or overfit
- Even failure teaches limits of cross-domain transfer

Evaluation Plan

- Compare routers:
  - Linear -> baseline
  - MLP -> expressiveness
  - Attention -> similarity matching
  - Expert-embedding -> semantic grounding
- Metrics: perplexity, utilization entropy, stability
- Goal: domain-adaptive discovering **unexpected expertise patterns**

Closing & Vision

- Prototype shows upcycling + MoE works with vanilla router
- Limitation: linear router = simplistic domain matching
- Contribution: expert embeddings -> **capability matching**
- Vision: routers that **actively discover hidden capabilities**
- Aligns with Cohere's push for **scarling & adaptive architectures**