ChatGPT

In-depth research plan and feasibility analysis for recurrent-depth Transformers with conditional experts, depth mixing, and KV-cache compression

Executive summary

This report proposes a staged research program to test whether Transformer “depth” (stacked layers) can be reframed as an iterative recurrent process and then made more efficient through (a) conditional computation via depth-wise Mixture-of-Experts (MoE), (b) learnable cross-step hidden-state propagation (depth mixing), and (c) cross-step KV-cache compression using a shared latent memory. The plan is designed to produce publishable findings even if efficiency gains are modest, because it explicitly measures *structural redundancy across depth* (representation similarity, gradient alignment, attention-rank) alongside classic language-model quality metrics.

Key prior work strongly supports feasibility of each component: Universal Transformers interpret depth as recurrent refinement with optional adaptive computation time; sparse MoE routing is a mature approach for conditional capacity scaling; and Multi-Head Latent Attention (MLA) demonstrates large KV-cache reductions by compressing K/V into a latent vector. [1]

A practical execution strategy is to run a **three-tier scaling ladder**:

- **Tier A (pilot)**: 100–350M parameter decoder-only models on WikiText-103 / OpenWebText-style data to validate correctness, stability, and measurement infrastructure. WikiText-103 is ~103M tokens and supports fast iteration with clear perplexity signals. [2]
- **Tier B (mid-scale)**: 0.7–1.3B parameters on a curated C4/Pile/Dolma slice for more realistic pretraining dynamics and MoE routing behaviors. [3]
- **Tier C (stress test)**: 3–7B (optional, compute permitting) focusing specifically on KV-cache compression and inference throughput/memory, where effects are largest and easiest to monetize.

Primary success criteria are framed as **Pareto improvements**: (1) match baseline validation loss/perplexity within a tight delta while (2) reducing *either* activated parameters, compute, or inference KV-cache memory at fixed context length. Decision gates are defined after each phase to avoid sunk-cost scaling of unstable variants.

You should plan for **12–16 weeks** to reach credible workshop-paper results at Tier B, assuming a small team (1 research engineer + 1 researcher) and a moderate GPU allocation. A conservative budget for Tier A + Tier B is on the order of **5k–20k GPU-hours** depending on hardware and tokens trained; storage needs are modest (generally **1–5 TB** including datasets + checkpoints), but raw web corpora can dominate if you self-host. The most likely failure modes are (i) MoE router collapse or instability (mitigated via load balancing and z-loss), (ii) depth-mixing shortcut learning (mitigated via warmup/scheduling and constrained mixing),

and (iii) excessive quality loss under aggressive KV compression (mitigated via latent-dimension sweeps and distillation/continued pretraining). [4]

Project framing and assumptions

What this project is actually testing

The core scientific claim is that depth in modern Transformers may behave less like a stack of strongly distinct functions and more like an **iterative refinement trajectory** in a shared state space. This hypothesis is consistent with (a) recurrent depth formulations (Universal Transformers), (b) continuous/infinite-depth interpretations (Neural ODEs, Deep Equilibrium Models), and (c) observed layer similarity analyses in Transformers. [5]

Open-ended assumptions (explicit options)

Because you didn't specify model scale, compute, or wall-clock constraints, the plan is parameterized with options:

- **Target scale options**
- **S**: 125–350M params (fast correctness + ablations)
- **M**: 0.7–1.3B params (realistic scaling behavior)
- **L**: 3–7B params (KV-cache/inference-focused feasibility)
- **Compute availability options**
- **Low**: 4–8 GPUs (A100-80GB / H100-80GB class)
- **Moderate**: 16–32 GPUs (enables robust sweeps and longer runs)
- **High**: 64+ GPUs (only needed if you want "production-like" MoE scaling)
- **Wall-clock objective**
- **Workshop result** in 12–16 weeks (recommended)
- **Full paper** in 24–36 weeks (includes larger-scale sweeps + stronger theory)

Hardware naming here assumes NVIDIA [6]-class accelerators because KV-cache and attention kernels are dominated by GPU memory bandwidth in typical decoding, motivating MQA/GQA/MLA-style approaches. [7]

Datasets to consider (LM + probes)

For language modeling and probing, prefer datasets with strong precedent and clear evaluation protocols:

- **Fast iteration**: WikiText-103 (well-known LM perplexity benchmark; ~103M tokens) [8]
- **Web-scale style**: OpenWebText reproductions (open clone of WebText scraping) [9]
- **Standard pretraining corpora**: C4 (from T5 work; also documented for composition/source issues) [10]
- **Research-friendly large corpora**: The Pile (825 GiB, 22 components) [11] and Dolma (3T tokens) from Allen Institute for AI [12] [13]
- **Modern cleaned Common Crawl**: FineWeb (Hugging Face Science) from Hugging Face [14] [15]
- **Probing / transfer**: GLUE and SuperGLUE for post-hoc representation quality and downstream robustness checks. [16]

Research objectives, hypotheses, and success criteria

Research objectives

This project has four concrete objectives, each aligned to a phase:

1. **Recurrent baseline correctness**: Build a recurrent/unrolled Transformer that is functionally and gradient-equivalent to a standard multi-layer Transformer, establishing a controlled baseline interpretation of depth as iterative computation. [17]
2. **Conditional depth computation**: Introduce learnable routing so that each token at each depth step activates only a subset of experts, testing depth redundancy and conditional capacity. [18]
3. **Learnable cross-step information flow**: Allow hidden states from earlier steps to influence later steps in a learnable way (depth mixing), testing whether strict Markov depth transitions are unnecessarily restrictive—analogous in spirit to dense connectivity benefits. [19]
4. **Cross-step KV-cache compression**: Replace per-step KV storage with a shared latent memory and lightweight "reconstruction" projections, aiming to reduce inference memory substantially without degrading quality—motivated by MLA. [20]

Hypotheses (testable)

- **H0 (baseline)**: A recurrent unroll with identical parameters per step (or step-indexed parameters matching layers) reproduces forward pass and gradients to numerical tolerance.
- **H1 (depth redundancy / MoE)**: A depth-wise MoE with fewer active experts (or fewer unique experts shared across steps) can match dense baseline validation loss at equal training compute, indicating redundancy across depth. [18]
- **H2 (heterogeneous depth needs)**: Routing usage across steps is token-dependent and correlated with token difficulty (loss contributions), resembling adaptive computation intuitions from recurrent depth models. [21]
- **H3 (depth mixing benefit)**: Learnable cross-step mixing improves optimization (faster convergence or lower instability) and preserves multi-scale features; its benefit increases with depth and/or more aggressive parameter sharing. [22]
- **H4 (latent KV structure)**: Keys/values across steps (and/or layers) lie in a lower-dimensional subspace; compressing them through a latent vector yields large KV-cache savings with bounded loss delta. [23]

Success criteria (quantitative "go/no-go")

Define three tiers of success so the project is publishable even if efficiency is modest:

- **Tier 1 (scientific validation)**
- Baseline equivalence tests pass (max abs diff logits < 1e-5 in fp32; gradient cosine > 0.999 for matched parameters).
- Clear measurements of depth redundancy (e.g., high layer/step similarity via CKA or the simpler cosine-based proxies) reproduce known trends. [24]
- **Tier 2 (practical efficiency)**
- MoE: match baseline perplexity within $\leq$ **1–2% relative** while reducing activated MLP FLOPs or expert count (or maintaining quality at lower compute).

- Mixing: at fixed compute, improve convergence speed (tokens-to-target-loss) by $\geq$ **10–20%**, or stabilize training enabling larger LR/batch.
- KV compression: reduce KV-cache bytes by $\geq$ **4×** at equal context length with $\leq$ **0.1–0.2 nats** validation loss increase (tunable per scale).
- **Tier 3 (production-facing)**
- Demonstrate higher max-batch decode throughput at fixed GPU memory, and/or longer context at fixed memory (KV dominates in long-context decode). [25]

Methodology by phase

Recurrent baseline

Architecture variants

You need two baselines:

- **Standard Transformer**: classic decoder-only stack of L blocks (attention + MLP).
- **Recurrent/unrolled Transformer**: a loop over "depth steps" `t=1..L` applying either:
- **Step-indexed parameters**: `F_t` matches layer t exactly (strict equivalence target), or
- **Shared parameters**: `F` reused across steps (Universal Transformer style; not equivalent but a meaningful comparison). [26]

Training setup (recommended defaults)

Use a GPT-style next-token objective. Keep training standard to avoid confounds:

- Optimizer: AdamW
- LR schedule: cosine decay with warmup (by tokens)
- Precision: bf16 mixed precision once correctness tests pass
- Sequence length: start 512–1024; later test long-context where KV-cache is most relevant
- Logging: per-step loss contributions (where applicable), memory stats, throughput

Datasets

Tier A: WikiText-103 + OpenWebText (small subset). [2]
Tier B: C4/Pile/Dolma slice (e.g., 50–200B tokens is unnecessary for methodology; you can learn signals at 1–10B tokens, then scale selectively). [3]

Metrics

- LM: validation loss, perplexity (PPL)
- Correctness: forward diff, backward diff, determinism (seeded)
- Depth structure: representational similarity (CKA/cosine), per-step gradient similarity, attention-pattern similarity [24]

Ablations

- Shared vs unshared parameters across steps (bridges to Universal Transformer). [17]

- LayerDrop-style random step dropping during training (as a bridge to depth redundancy and on-demand depth). [27]

Expected outcomes

- Strict equivalence implementation reproduces the baseline within numerical tolerance.
- Similarity analytics show increasing similarity for adjacent steps/layers (consistent with prior layer-similarity findings). [28]

Failure modes and mitigations

- **Non-equivalence due to nondeterministic kernels**: run fp32 + deterministic settings for tests; only later enable fused kernels.
- **Training divergence differences**: confirm identical order of operations and parameter application; unit tests at module boundary (attention-only, MLP-only).

Decision criteria to proceed

Proceed once both of these are true: 1) equivalence tests pass; 2) profiling/logging infrastructure is stable and produces reproducible baseline curves.

Suggested visualizations (request to include)

- Architecture diagram (standard stack vs recurrent unroll)
- Step-by-step CKA heatmap (layer/step similarity)
- Gradient cosine similarity across steps (matrix)

Learnable depth gate and MoE

Architecture

Replace the MLP sub-layer within each step with an MoE layer:

- Experts: E feed-forward networks (e.g., SwiGLU)
- Router: linear projection from hidden state to E logits; select top-k experts per token
- Optional: share experts across steps to test “fewer experts, same loss” (your original goal)

This is directly motivated by sparsely gated MoE and Switch Transformer simplifications. [29]

Stability and load balancing

MoE is vulnerable to imbalance/collapse. Use two standard techniques from the sparse MoE literature:

- **Auxiliary load-balancing loss** (popularized in Switch): encourages uniform expert utilization. [30]
- **Router z-loss** (from “Designing Stable and Transferable Sparse Expert Models”): penalizes large router logits for numerical stability. [31]

Also follow standard practices: - capacity factor / token dropping accounting (Switch-style) [30]
- router noise / dropout early in training (ablation to quantify necessity)

Experiments

Design experiments to isolate *redundancy across steps* rather than just "MoE is good".

1) **Baseline MoE per step (no sharing)**: each step has its own experts; measures pure conditional compute.
2) **Shared expert pool across steps**: a single pool of experts reused at all steps; tests your key hypothesis ("use fewer experts overall").
3) **Shared router vs per-step router**: tests whether routing decisions must be step-conditioned.
4) **Top-1 vs Top-2**: Switch popularizes top-1 for speed; top-2 can improve quality at some cost. [32]

Metrics

Along with LM loss/PPL: - Expert usage entropy (per step, per layer, per token position)
- Load imbalance: coefficient of variation of tokens/expert
- Token drop rate (overflow)
- Router logits statistics (for z-loss monitoring)

Expected outcomes

- If depth contains redundancy, shared-expert variants should preserve PPL with fewer unique experts (or lower activated parameters) than a naive per-step design.
- Routing should become more heterogeneous on "harder" tokens—an effect analogous to adaptive computation time, but realized through expert choice rather than halting. [33]

Failure modes and mitigations

- **Expert collapse (most tokens go to a small subset)**: increase load-balancing coefficient; add z-loss; consider top-2; schedule router learning rate separately. [32]
- **Training instability in mixed precision**: z-loss + careful scaling; instrument router-loss spikes and apply clipping. [31]
- **Systems overhead dominates** (especially multi-GPU all-to-all): keep Tier A on single-node; only scale distributed MoE after proving benefit (Switch and GShard highlight systems complexity as a first-order concern). [34]

Decision criteria to proceed

Proceed to depth-mixing/KV compression only if either: - (A) you match baseline within the Tier-2 loss delta at some meaningful compute/parameter reduction, or
- (B) even if efficiency is weak, you observe strong evidence of depth redundancy via usage patterns and similarity metrics that justify deeper structural interventions.

Suggested visualizations (request to include)

- Expert-usage heatmaps (steps × experts × token position)
- Router entropy by step

- Load imbalance curve over training
- “Token difficulty vs routing diversity” scatter

Learnable hidden-state mixing across depth

Architecture

Introduce learnable cross-step information flow. Two practical families:

- **Fixed-tap mixing (low overhead)**
  For example, feed step `t+1` a mixture of `s_t` and one earlier state `s_{t−Δ}` (like your “step1 + step9 → step10” idea), using:
- scalar gate α
- per-channel gate α ∈ R^d

- or small MLP gate conditioned on LN(s_t), LN(s_{t−Δ})

- **Multi-tap mixing (generalized DenseNet-like connectivity)**
  Compute: `ŝ_t = Σ_{k∈S(t)} w_{t,k} s_k`, where S(t) is small (e.g., {t, t−4, t−8}) to stay efficient. DenseNet motivates dense connectivity as improving gradient flow and feature reuse, which is conceptually similar here across depth steps. [22]

This phase is also conceptually linked to recurrent depth models (Universal Transformer), but instead of tying weights or halting, you’re improving *state propagation across the depth “time” axis*. [17]

Key ablations

- No mixing vs mixing
- Mixing only into attention vs only into MLP vs both
- Gate type: scalar vs vector vs dynamic
- Tap set size: 2-tap vs 3-tap vs 4-tap
- Warmup schedules: start with α≈0 (no skip) then increase learnability vs start free

Metrics

- LM loss/PPL
- Convergence speed: tokens-to-target-loss
- Step drift: cosine distance between s_t and s_{t+1} statistics
- Redundancy: CKA/cosine similarity changes across steps (mixing should change the similarity structure if it meaningfully re-routes information). [24]

Expected outcomes

- If depth behaves like iterative refinement, mixing should (a) stabilize training and (b) reduce “overwriting” of early lexical features; the effect should be clearer in deeper or partially shared-parameter settings.

- You may find a small number of taps is enough, supporting a *low-order depth dynamical system* viewpoint (a useful scientific conclusion even if quality gains are small).

Failure modes and mitigations

- **Shortcut learning (model ignores later refinement)**: constrain initial mixing strength (α warmup); regularize toward using near-step inputs early; optionally drop the skip path stochastically so later steps must remain functional.
- **Distribution mismatch across steps**: apply LN before mixing (or RMSNorm) to align scales.
- **Memory overhead**: mixing requires caching some previous step states during training; mitigate with activation checkpointing and limiting tap count.

Decision criteria to proceed

Proceed if you see either: - improved convergence / stability at fixed compute, or
- a clearer separation of "early lexical vs late semantic" representations in probing, suggesting that depth mixing provides interpretable multi-scale structure.

Suggested visualizations (request to include)

- "Tap weights over training" plot (w_{t,k} trajectories)
- Step similarity heatmaps before/after mixing
- Probing accuracy vs step index (lexical vs semantic probes)

KV-cache compression across steps

Architecture direction

Inference-time KV-cache grows with layers/steps and context length, making it a prime target. Many approaches reduce KV cost by sharing K/V across heads or groups (MQA, GQA). [35]

Your project specifically proposes *cross-step/layer* compression. A well-motivated starting point is **MLA**:

- Store a compact latent vector per token and reconstruct keys/values from it, substantially reducing KV cache; this is central to DeepSeek-V2's inference efficiency claims. [36]

A generalization for your recurrent-depth setting:

- Cache only `z_i` (latent) per token position i
- For each step t and head h, reconstruct:
- `K_{i,t,h} = z_i · B^K_{t,h}`
- `V_{i,t,h} = z_i · B^V_{t,h}`

This directly implements "compress multiple step/layer KV into one latent."

Methodology: measure redundancy *before* compressing

To make this scientifically rigorous, add an analysis stage:

- Collect per-step K/V tensors on a fixed validation set for the baseline
- Run PCA (or randomized SVD) across step dimension:
- explained variance vs rank
- subspace overlap across steps
- Compute effective rank of attention logit matrices (or attention probability matrices) across steps to see whether attention is low-dimensional.

These analyses justify the latent-dimension choices and provide publishable evidence of structure even if compression is lossy.

Experiments

- **Latent dimension sweep**: r ∈ {d/16, d/8, d/4, d/2}
- **Per-step vs shared reconstruction matrices**: B_t vs shared B
- **Hybrid with GQA/MQA**: combine step-latent KV with fewer KV heads (especially relevant at long context). [37]
- **Kernel/serving realism**: optionally test on a paged KV cache manager (vLLM/PagedAttention) to separate "bytes saved" from "waste reduced." [38]

Metrics

- Quality: loss/PPL deltas
- Memory: KV bytes per token at inference, max context at fixed memory
- Throughput: tokens/sec at fixed batch size; also max batch at fixed memory
- Compute overhead: extra matmuls for K/V reconstruction per step

Expected outcomes

- If the cross-step KV structure is low-rank, compression should give large memory savings with bounded PPL loss, similar in spirit to MLA's reported KV reductions. [39]
- Even if compression hurts, you still obtain a quantitative "rank vs performance" frontier that establishes feasibility and guides later work.

Failure modes and mitigations

- **Quality collapse at low r**: increase r; use step-specific B_t; distill from baseline (logit matching) to recover performance.
- **Compute overhead offsets memory gains**: fuse reconstruction into attention kernels; prioritize long-context decode where memory bandwidth dominates (the motivation behind MQA and FlashAttention's IO framing). [40]

Decision criteria to proceed

Proceed to scaling only if you achieve a meaningful memory reduction (≥4×) at acceptable loss delta, or if measurements show strong low-rank structure that suggests engineering effort (kernel fusion) is justified.

Suggested visualizations (request to include)

- PCA scree plots per step and aggregated
- “Loss vs latent dimension r” curve
- “KV bytes vs context length” chart
- Attention effective-rank vs depth-step chart

Compute, storage, and engineering feasibility

Implementation stack (practical)

A feasible engineering strategy:

- Build on a clean PyTorch/Transformer codebase with:
- modular attention and MLP blocks
- deterministic test mode
- logging hooks for per-step tensors
- Add MoE with careful instrumentation (dropped tokens, routing stats)
- Add mixing via a pluggable “depth mixer” module
- Add KV-latent via attention module options

For inference realism: - Start with simple contiguous KV buffers, then (optional) test paged KV to validate benefits under real serving fragmentation. [41]

Memory profiling plan

You should profile three separate memory regimes:

1) **Training activations** (dominant for mid/large models)
- use framework max memory logging + activation checkpointing toggles
2) **Inference KV cache** (dominant for long-context decode)
- compute bytes explicitly from tensor shapes and dtype
3) **Bandwidth-sensitive attention**
- check whether reductions translate to speed using IO-aware kernels (FlashAttention motivates that IO, not FLOPs, often dominates). [42]

Compute estimation (how to do it, plus example budgets)

Instead of hardcoding a single token budget, use an explicit “compute planner”:

- Choose target tokens T and model parameters P
- Estimate dense training FLOPs $\approx O(P \cdot T)$ (exact prefactors depend on architecture, sequence length, and attention implementation; treat as planning-level)
- Measure *actual* throughput (tokens/sec/GPU) on Tier A, then extrapolate

A practical workshop plan often uses: - Tier A: 0.1–1B tokens total across all ablations
- Tier B: 1–10B tokens across selected variants
This is enough to see stable trends for architecture comparisons without chasing SOTA perplexity.

Resource budget table (illustrative planning ranges)

Scenario	Goal	Model scale	Total tokens trained	GPUs	Est. GPU-hours (range)	Storage (checkpoints+logs)
Pilot	correctness + core ablations	125–350M	0.2–1B	4–8	500–2,500	0.5–1 TB
Mid-scale	stable MoE + mixing + KV sweeps	0.7–1.3B	1–5B	8–16	3,000–12,000	1–3 TB
Stress test	KV-cache + throughput at long context	3–7B	0.5–2B (selected runs)	16–32	8,000–30,000	2–5 TB

Notes: - GPU-hour ranges intentionally reflect the largest unknown: your achievable tokens/sec depends on kernels, sequence length, and cluster/network overhead (especially for MoE). Switch and GShard emphasize that systems details materially affect efficiency claims. [43]

Personnel and roles

A small, effective team: - **1 Research Engineer (systems + kernels + profiling)**: MoE routing correctness, distributed behavior, inference memory/throughput
- **1 Research Scientist/Engineer (modeling + analysis)**: experimental design, ablations, similarity analyses, write-up
Optional: **0.5 MLOps** for dataset pipelines and tracking if using large corpora.

Timeline and deliverables

Gantt-style timeline (Mermaid request included)

Below is a concrete 16-week plan starting the week after the current date (Feb 13, 2026). Adjust durations based on compute.

```
gantt
    title Recurrent-depth Transformer research plan (16 weeks)
    dateFormat  YYYY-MM-DD
    axisFormat  %b %d

    section Foundations
    Baseline code + deterministic equivalence tests     :a1, 2026-02-16, 14d
    Instrumentation (CKA, gradients, attention stats)   :a2, after a1, 10d
    Tier-A baseline runs (Wikitext/OpenWebText)         :a3, after a2, 14d

    section Depth-wise MoE
```

```
    MoE implementation + aux losses (LB + z-loss)       :b1, after a2, 14d
    MoE ablations (shared experts, top-1/top-2, etc.)   :b2, after b1, 21d
    Decision gate: MoE viability                         :milestone, after b2,
0d

    section Hidden-state mixing
    Mixer implementation (2-tap -> multi-tap)           :c1, after b1, 10d
    Mixing ablations + stability schedules              :c2, after c1, 21d
    Decision gate: mixing benefit                         :milestone, after c2,
0d

    section KV-cache compression
    Measure KV redundancy (PCA, rank, similarity)       :d1, after a3, 14d
    Latent KV (MLA-style) implementation + sweeps       :d2, after d1, 21d
    Inference profiling (bytes, throughput, context)    :d3, after d2, 14d

    section Write-up
    Consolidate results + figures                       :e1, after d3, 10d
    Workshop paper draft                                :e2, after e1, 10d
```

Decision flowchart (Mermaid request included)

This helps enforce “stop/go” gates.

```
flowchart TD
  A[Recurrent baseline equivalence] -->|Pass| B[MoE depth gate experiments]
  A -->|Fail| Afix[Fix determinism & module order] --> A

  B -->|Stable routing + competitive loss| C[Hidden-state mixing experiments]
  B -->|Collapse/instability| Bfix[Adjust LB loss, z-loss, capacity, top-k] -->
B
  B -->|No benefit| D[KV redundancy analysis + compression]

  C -->|Faster convergence or better loss| D
  C -->|Shortcut/collapse| Cfix[Warmup gates, constrain taps, LN before mix] -->
C
  C -->|No benefit| D

  D -->|≥4x KV reduction with bounded loss| E[Scale up + inference profiling]
  D -->|Quality too low| Dfix[Increase r, step-specific B_t, distill] --> D
```

Deliverables table

Deliverable	When	What “done” means
Baseline equivalence suite	Weeks 1–2	Unit tests + forward/grad diffs pass
Measurement pack	Weeks 2–4	CKA/gradient/attention-rank scripts produce stable plots [44]
MoE depth-gate results	Weeks 5–8	Loss + routing stats + ablation table; stability analysis [32]
Depth-mixing results	Weeks 7–11	Convergence and probe results; mixing-weight diagnostics
KV compression results	Weeks 10–14	PCA/rank evidence + loss/memory frontier; MLA-like variant tested [39]
Workshop draft	Weeks 15–16	Narrative, figures, and reproducible configs

Key references and prior art to consult

This is not a bibliography section; it’s an action-oriented reading list prioritized toward primary sources and directly relevant methods.

Recurrent depth and adaptive computation

- Universal Transformers (depth as recurrence; optional adaptive computation time). [26]
- Adaptive Computation Time (original ACT mechanism). [45]
- Deep Equilibrium Models (infinite-depth / fixed point view; includes Transformer applications and constant-memory angle). [46]
- Neural ODEs (continuous-depth framing and memory/compute trade discussions). [47]

Sparse MoE and routing stability

- Sparsely-Gated MoE layer (conditional computation; foundational gating + load balancing). [48]
- Switch Transformers (top-1 routing + auxiliary load balancing; practical large-scale recipe). [49]
- Designing Stable and Transferable Sparse Expert Models (router z-loss and stability techniques). [31]
- GShard / GLaM (systems + scaling considerations for sparse expert models). [50]

KV-cache efficiency and latent attention

- Multi-Query Attention (keys/values shared across heads; inference bandwidth motivation). [51]
- GQA (interpolates between MHA and MQA; uptraining recipes). [52]
- DeepSeek-V2 and MLA (compress KV into latent; strong empirical memory reduction claims). [53]
- vLLM / PagedAttention (serving-oriented KV cache management; helps validate “real-world” memory wins). [38]
- FlashAttention (IO-aware exact attention framing; relevant when “compute vs bandwidth” tradeoffs matter). [42]

Measuring depth redundancy and representation similarity

- CKA (representation similarity metric suitable for comparing layers/steps). [54]
- LayerDrop (structured layer dropout; supports on-demand depth and pruning-induced insights). [27]
- Recent layer similarity analyses in Transformers (practical evidence and metrics guidance). [55]

Datasets and benchmarks to cite in write-ups

- WikiText-103 dataset characteristics (quick LM benchmark). [8]
- GLUE / SuperGLUE (probing / downstream evaluation). [16]
- The Pile (diverse text corpus for LM research). [11]
- Dolma (large open corpus with detailed documentation; reproducibility emphasis). [13]
- C4 / T5 work and C4 documentation (standard pretraining reference and caveats). [10]

Figure and visualization requests to include in the final report/paper

To maximize clarity and credibility, the final report should include: - Architecture diagrams (baseline stack vs recurrent unroll; MoE routing block; depth-mixer block; latent KV/MLA block) - Expert-usage heatmaps (steps × experts; token positions × experts) - Routing entropy and load imbalance curves - PCA scree plots for K/V across steps; subspace overlap matrices - Attention effective-rank charts across steps and layers - CKA similarity heatmaps across steps (baseline vs mixing vs MoE) - Memory breakdown charts (training activation vs params vs optimizer; inference KV bytes vs context)

If you want, I can turn this plan into a “ready-to-execute” experiment sheet (YAML-like configs per variant, exact hyperparameter grids, and a minimal set of runs that maximizes information gain per GPU-hour).

---

[1] [5] [17] [26] [1807.03819] Universal Transformers
https://arxiv.org/abs/1807.03819?utm_source=chatgpt.com

[2] [8] [12] https://mlsys.org/Conferences/doc/2018/50.pdf
https://mlsys.org/Conferences/doc/2018/50.pdf

[3] [10] https://arxiv.org/abs/1910.10683
https://arxiv.org/abs/1910.10683

[4] [49] Switch Transformers: Scaling to Trillion Parameter Models ...
https://arxiv.org/abs/2101.03961?utm_source=chatgpt.com

[6] [24] [44] [54] https://arxiv.org/abs/1905.00414
https://arxiv.org/abs/1905.00414

[7] [25] [35] [40] [51] Fast Transformer Decoding: One Write-Head is All You Need
https://arxiv.org/abs/1911.02150?utm_source=chatgpt.com

[9] https://github.com/jcpeterson/openwebtext
https://github.com/jcpeterson/openwebtext

[11] https://arxiv.org/abs/2101.00027
https://arxiv.org/abs/2101.00027

13 https://arxiv.org/abs/2402.00159
https://arxiv.org/abs/2402.00159

14 21 33 45 https://arxiv.org/abs/1603.08983
https://arxiv.org/abs/1603.08983

15 https://huggingface.co/datasets/HuggingFaceFW/fineweb
https://huggingface.co/datasets/HuggingFaceFW/fineweb

16 https://arxiv.org/abs/1804.07461
https://arxiv.org/abs/1804.07461

18 29 48 The Sparsely-Gated Mixture-of-Experts Layer
https://arxiv.org/abs/1701.06538?utm_source=chatgpt.com

19 22 Densely Connected Convolutional Networks
https://arxiv.org/abs/1608.06993?utm_source=chatgpt.com

20 23 36 39 53 DeepSeek-V2
https://arxiv.org/pdf/2405.04434?utm_source=chatgpt.com

27 Reducing Transformer Depth on Demand with Structured Dropout
https://arxiv.org/abs/1909.11556?utm_source=chatgpt.com

28 55 ANALYZ- ING AND ENHANCING LAYER-WISE SIMILARITY
https://proceedings.iclr.cc/paper_files/paper/2025/file/03d113a060c0ac93a5859517a0f07271-Paper-Conference.pdf?utm_source=chatgpt.com

30 32 43 Switch Transformers: Scaling to Trillion Parameter Models ...
https://jmlr.org/papers/volume23/21-0998/21-0998.pdf?utm_source=chatgpt.com

31 https://arxiv.org/pdf/2202.08906
https://arxiv.org/pdf/2202.08906

34 50 https://arxiv.org/abs/2006.16668
https://arxiv.org/abs/2006.16668

37 52 https://arxiv.org/abs/2305.13245
https://arxiv.org/abs/2305.13245

38 41 https://arxiv.org/abs/2309.06180
https://arxiv.org/abs/2309.06180

42 https://arxiv.org/abs/2205.14135
https://arxiv.org/abs/2205.14135

46 Deep Equilibrium Models
https://arxiv.org/abs/1909.01377?utm_source=chatgpt.com

47 Neural Ordinary Differential Equations
https://arxiv.org/abs/1806.07366?utm_source=chatgpt.com