L^3: Large Lookup Layers

Albert Tseng

Chris De Sa

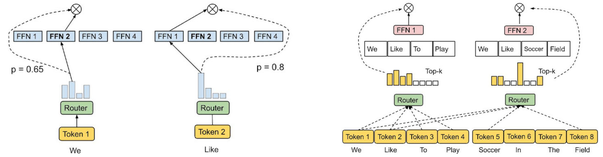
ICML International Conference On Machine Learning

We present a structured sparse architecture using embedding tables that enables fast offloaded inference and strong quality scaling

Structured Sparsity

Modern sparse models dynamically select a subset parameters to "activate" based on the current input token, allowing compute and memory to scale separately. For this to be efficient, the sparsity pattern must be structured in a way that causes minimal overhead.

The canonical way to achieve this is with the Mixture-of-Experts (MoE) architecture, which replaces each MLP block with a router and multiple experts, each of which is a MLP block. In each MoE layer, the router selects which experts to activate based on the current hidden state.



Although MoEs perform well, they're also difficult to serve at scale. Dynamic routing with the current hidden state means that we can't offload experts. With trillions of parameters, MoEs require complex sharding strategies to run efficiently.

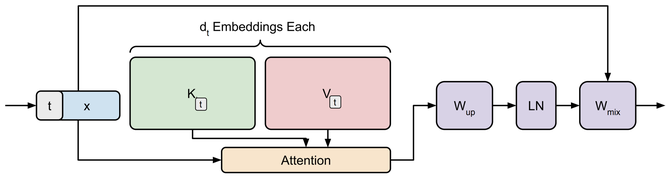
The L^3 Architecture

In contrast, tokenizer embedding tables, which exist in every LLM, are also natively sparse and come with none of the systems issues of MoEs. In L^3, we extend the concept of a tokenizer embedding table to the decoder layers of a model. L^3 achieves similar sparse scaling as MoEs while being systems-friendly, such as supporting low-overhead CPU-offloaded inference.

Formally, L^3 contains weights W_{mix}, W_{up} and embeddings K, V and performs

$$L^3\left(x, t; W_{mix}, W_{up}, K, V\right) \in \mathbb{R}^{d_{out}} = W_{mix}\left[\begin{array}{c}\text{LayerNorm}(W_{up}(V_t^T \text{Softmax}(K_t x))) \\ x\end{array}\right]$$

where K_t and V_t are the embeddings for token ID t and x is the current hidden state of t.

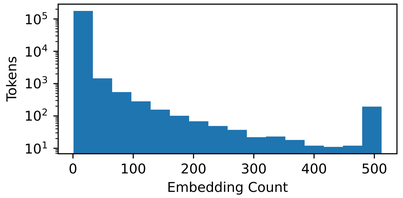


The key to L^3 is to perform static routing based on the token ID t and context-dependent aggregation of embeddings in K_t and V_t with x, which preserves quality and efficiency.

Embedding Allocation

The main tuning knob in L^3 is the number of embeddings we allocate to each token ID. Allocating one per token results in no context dependency, and allocating too many is inefficient.

In L^3, we view static routing as a context compression problem. We run the LZW compression algorithm to get a distribution of suffixes that end in a given token. Then, we use this distribution to assign token embeddings, with a cap for systems reasons. The resulting allocation significantly outperforms uniform allocation.

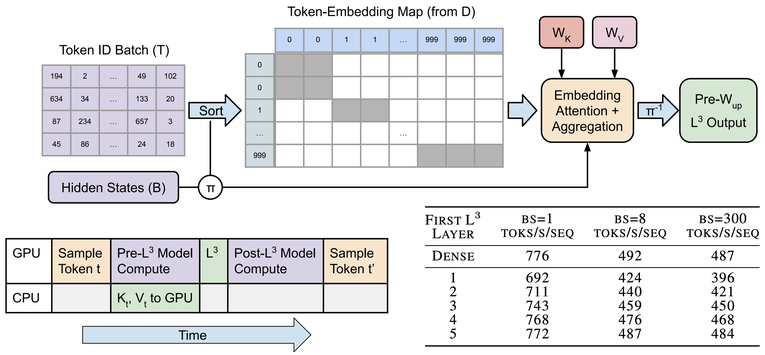


Algorithm 1 LZW Embedding Allocation

input Dataset $\mathcal{D}$, Tokenizer τ, Target Embedding Count v, Max Per-Token Embedding Count k.
Dictionary $C \leftarrow \{\}$
for Byte Sequence $s \in \mathcal{D}$ **do**
  $s_\tau \leftarrow \tau(s)$
  for $0 \leq i < |s_\tau|$ **do**
    $j \leftarrow 0$
    while $s_\tau[i-j:i+1] \in C$ **do**
      $j \leftarrow j+1$
    end while
    $C[s_\tau[i-j+1:i+1]] \leftarrow C[s_\tau[i-j+1:i+1]]+1$
    $C[s_\tau[i-j:i+1]] \leftarrow 1$
  end for
end for
Convert C to list of keys sorted descending by value.
Initial allocation $A = \{1, 2, 3, ..., |\tau|\}$, Count $A_C = \mathbb{1}^{|\tau|}$
$i \leftarrow 0$
while $|A| < v$ **do**
  if $A_C[C[i][-1]] < k$ **then**
    $A \leftarrow \left[A \quad C[i][-1]\right]$
    $A_C[C[i][-1]] \leftarrow A_C[C[i][-1]]+1$
  end if
  $i \leftarrow i+1$
end while
output sort(A)

Efficient Training and Inference

L^3's static routing allows us to efficiently train and serve L^3 models. Since the active embeddings only depend on the token ID, we can permute sequences and use fast block attention kernels (eg FlexAttention) during training. During inference, we can CPU-offload embedding tables and prefetch active rows during pre-L^3 compute. This adds ~0 overhead.



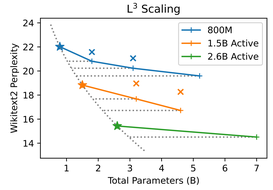
First L^3 Layer	BS=1 toks/s/seq	BS=8 toks/s/seq	BS=300 toks/s/seq
Dense	776	492	487
1	692	424	396
2	711	440	421
3	743	459	450
4	768	476	468
5	772	487	484

Results

L^3's quality scales with sparsity and model size

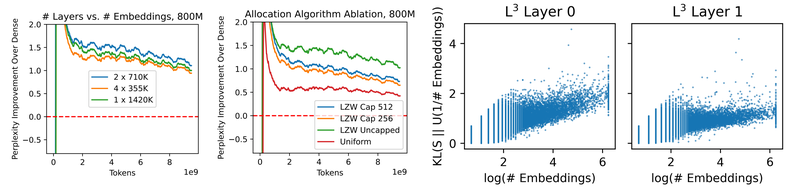
Active Params / Tok	L^3 Layers	Total Params	Ppl.	0-shot					
			Wiki2	Avg	ArcC	ArcE	HellaSwag	PiQA	Winogrande
809M	0	809M	22.02	48.28	27.56	60.52	34.32	67.13	51.85
795M	1	1.8B	20.81	49.10	28.21	62.03	35.58	67.52	52.17
803M	2	3.1B	20.23	49.45	28.67	62.92	36.09	67.83	51.72
818M	3*	5.2B	**19.59**	**50.25**	**29.01**	**63.64**	**36.32**	**68.23**	**54.06**
1.5B	0	1.5B	18.83	51.93	29.18	66.93	40.23	67.96	55.37
1.5B	1	3.1B	17.68	53.09	31.91	67.34	41.04	69.37	55.81
1.5B	2	4.6B	**16.72**	**53.84**	**33.11**	**67.63**	**41.93**	**70.67**	**55.88**
2.6B	0	2.6B	15.43	55.59	36.35	68.86	44.20	71.22	57.30
2.6B	2	7B	**14.51**	**56.98**	**38.21**	**71.21**	**44.95**	**71.71**	**58.80**

L^3 outperforms iso-FLOP and iso-sparse MoEs



Model	Avg	ArcC	ArcE	HSwag	PiQA	WinoG
800M 1 L3 MoE equiv.	48.18	27.25	60.27	34.79	67.36	51.22
800M 2 L3 MoE equiv.	48.59	27.60	60.48	35.28	68.39	51.22
1.5B 1 L3 MoE equiv.	51.68	30.63	65.65	38.89	70.18	53.04
1.5B 2 L3 MoE equiv.	52.83	31.72	66.54	41.23	70.52	54.13

L^3's LZW allocation strongly outperforms uniform allocation



L^3 layers act as high-sparsity, low-FLOPs caches for the model