OUTRAGEOUSLY LARGE NEURAL NETWORKS: THE SPARSELY-GATED MIXTURE-OF-EXPERTS LAYER

LLM Paper Reading Group
Kilian Ulrichsohn

MOTIVATION

as long there is enough training data, increasing the model size improves the model accuracy

However, increasing model size increases training and inference cost

Idea: Only use a subset of the available weights at a time

IDEA

1. Split a Layer in many experts
2. Select the experts depending on the input
3. ???
4. Profit

CHALLENGES

GPUs are bad at branching, but branching is necessary for conditional execution

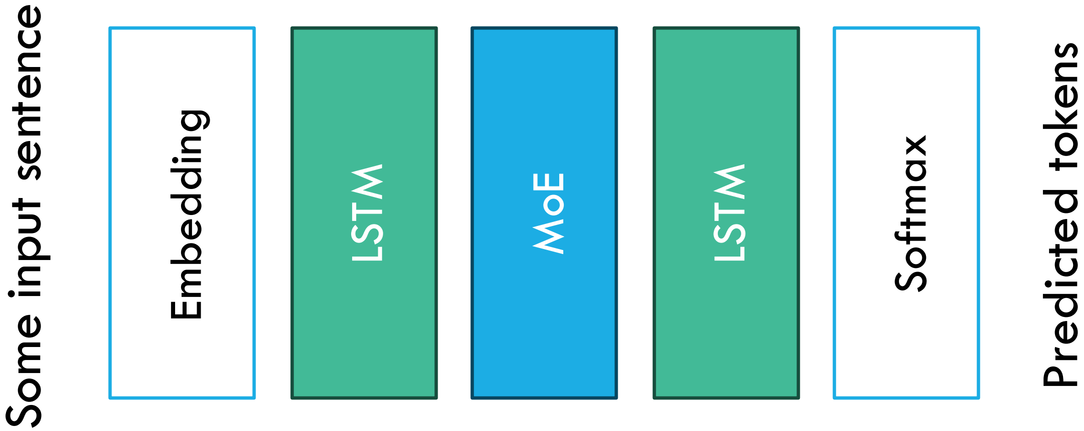
Large Batch sizes reduce the influence of costs of parameter transfer and updates

- However, splitting the batch to process them by different experts reduces the effective batch in the experts

Network Bandwith and Latency

- Operation on other devices must be large enough to be worth the introduced communication costs

ARCHITECTURE



APPROACH

Apply the MoE Layer on each vector in the input sequence

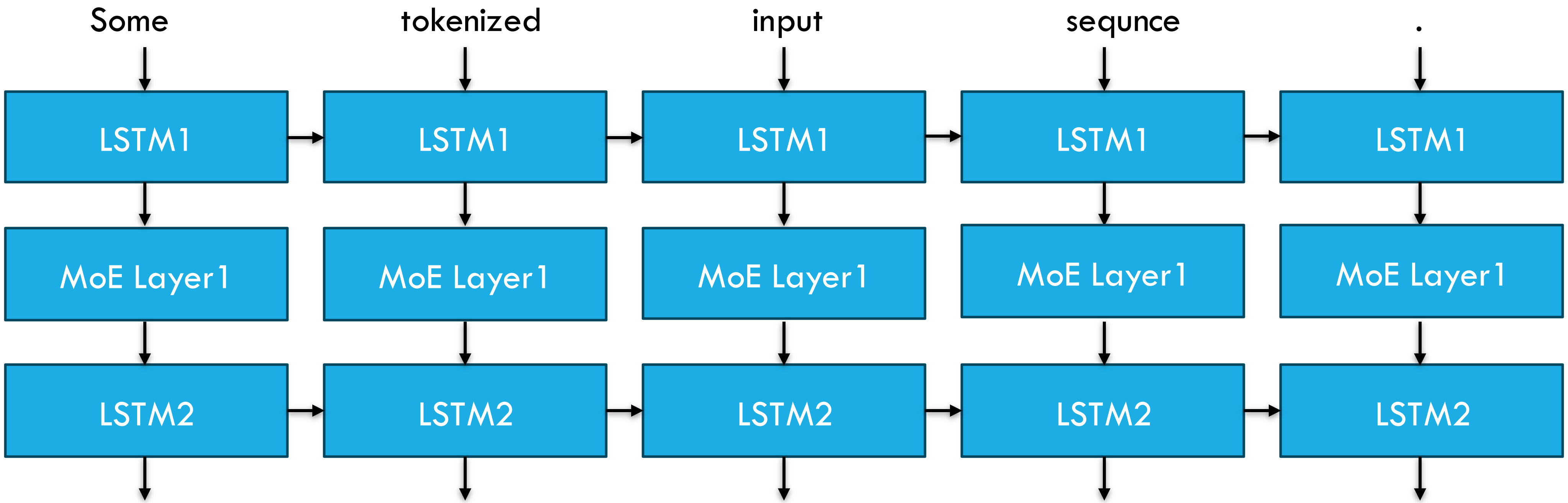
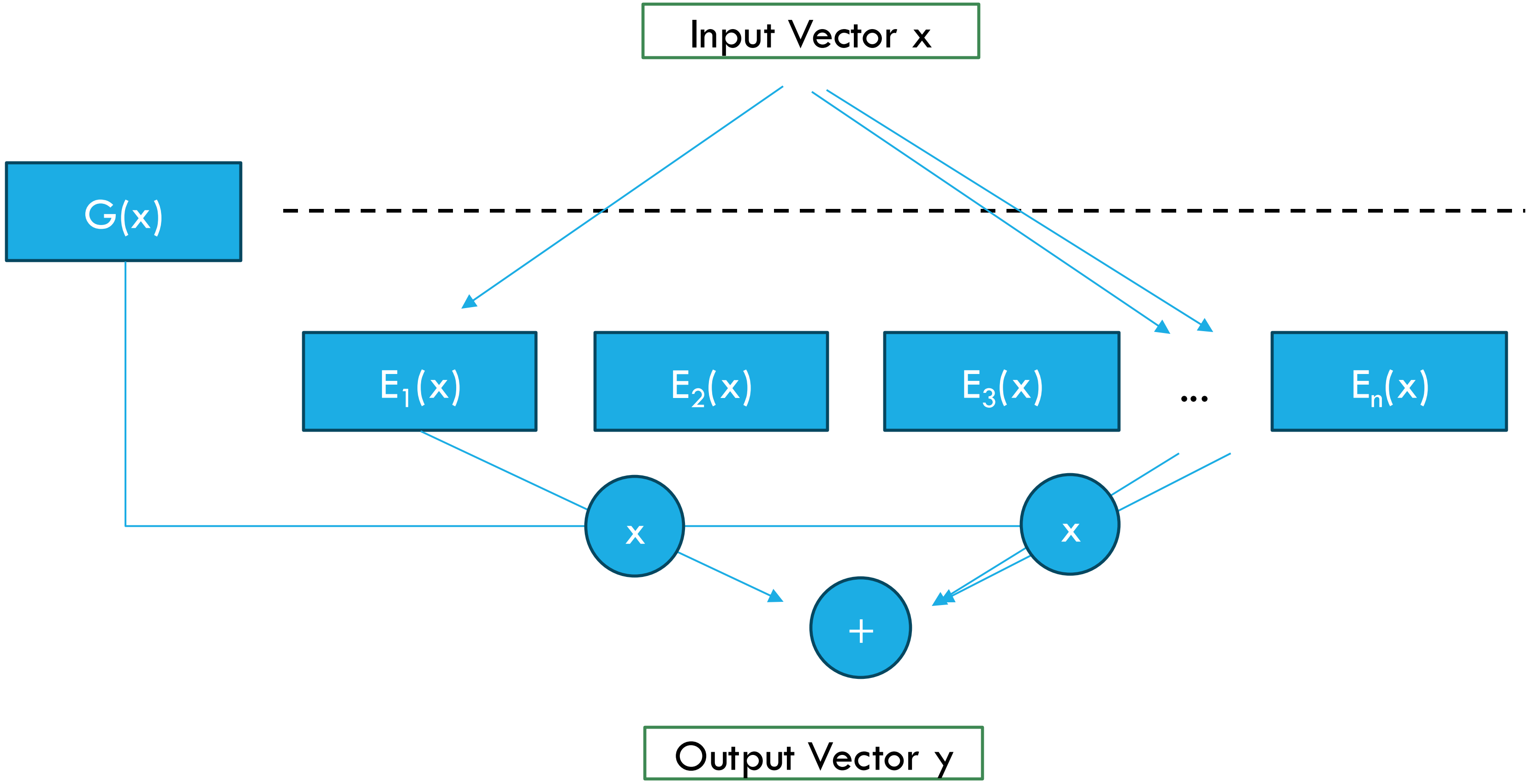


Figure shows architecture unrolled in the time domain

GATING NETWORK



Gating Function selects which Experts are invoked and weights the output of the experts

$$y = \sum_{i=1}^{n} G(x)_i E_i(x)$$

EXPERTS

$$y = RELU(x \ * \ W_1) * W_2$$

Choosing a large hidden size improves the ratio of network delay to computation time

Shape of W_1 input size x hidden size
Shape of W_2 hidden size x output size

GATING NETWORK 2

$$G(x) = Softmax(KeepTopK(H(x), k))$$

$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

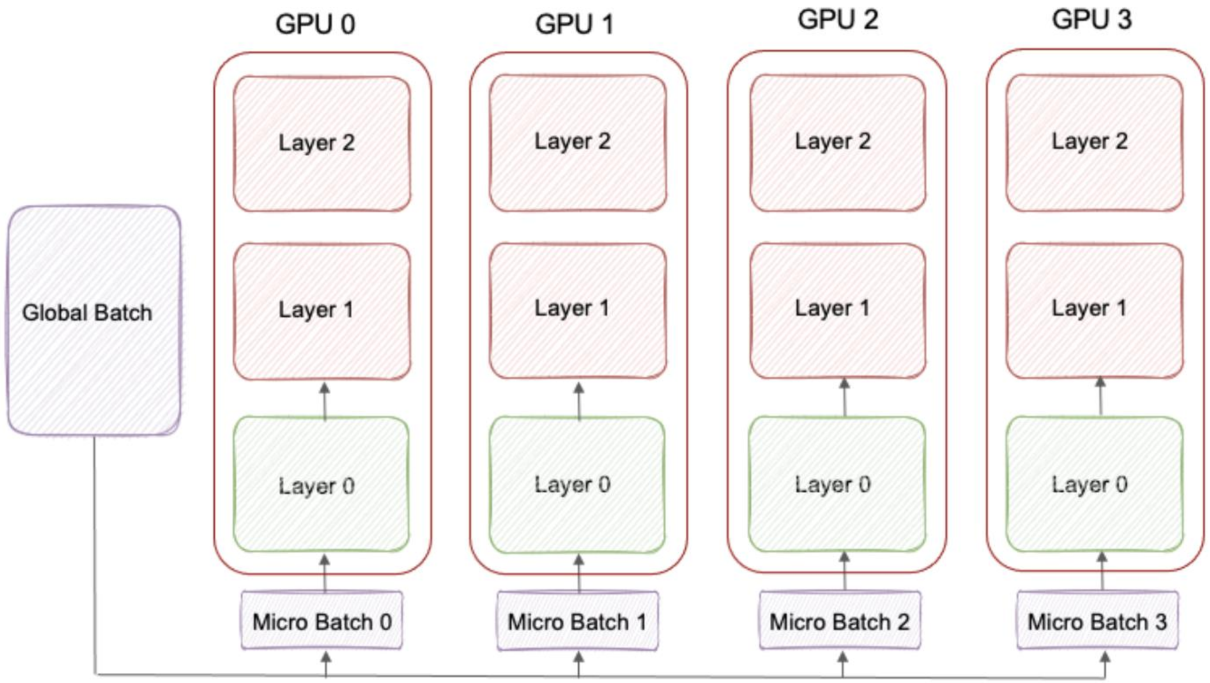
$$KeepTopK(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

- The noise term helps with load balancing
- W_{noise} , W_g is trainable
- k ... Number of invoked experts
- Creates a sparse gradient

INTERLUDE DATA PARALLEL TRAINING

Each GPU contains a whole copy of the network

Batch of examples is evently split into micro batches across GPUs

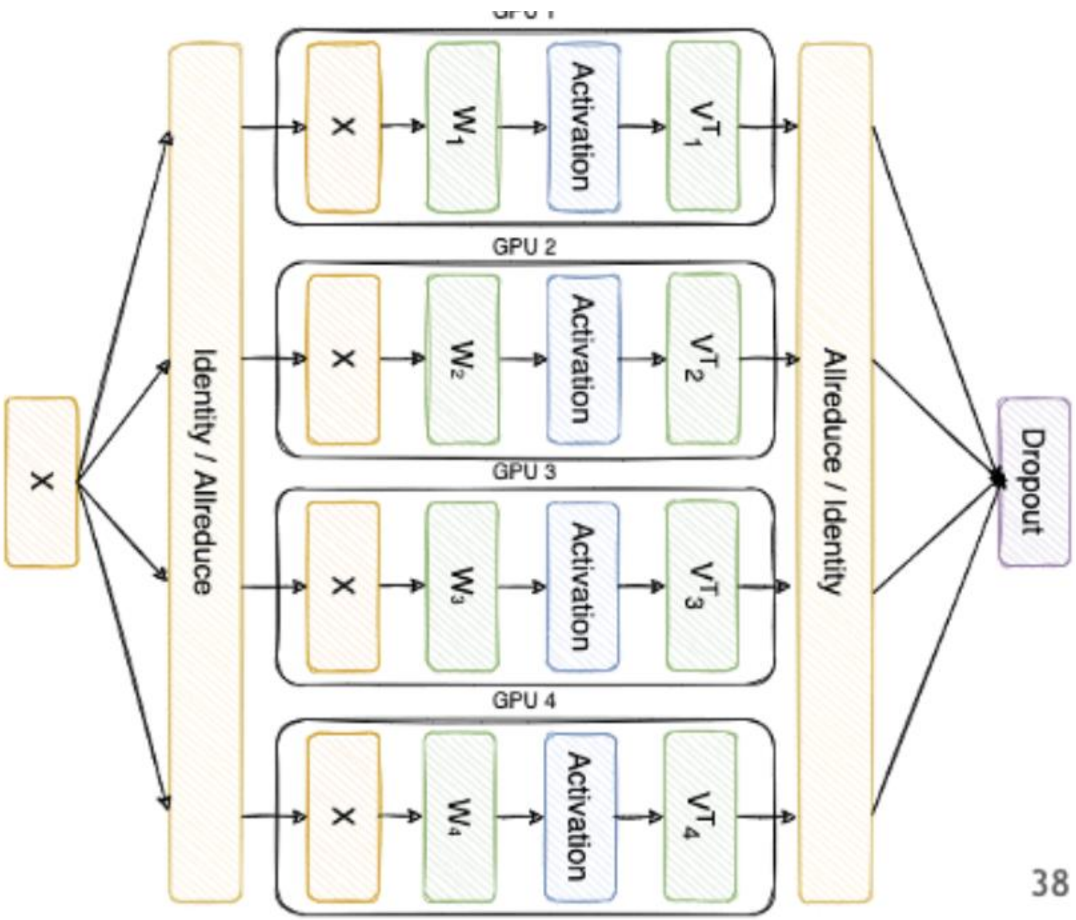


https://docs.nvidia.com/nemo-framework/user-guide/24.09/nemotoolkit/features/parallelisms.html

INTERLUDE MODEL PARALLELISM

- Each Layer is split horizontally in as many matrices as there are GPUs
- after a couple of layers → reduce operation

https://docs.nvidia.com/nemo-framework/user-guide/24.09/nemotoolkit/features/parallelisms.html



DISTRIBUTED TRAINING - CONSIDERATIONS

Large Batch sizes reduce the share of costs of parameter transfers and updates

With increasing number of experts n, the effective batch size for each expert becomes small

$$\frac{kb}{n} \ll b$$

Solution: make the initial batch size as large as possible (constrained by GPU memory)

effective batch size == batch size * sequence length

- Since MoE is applied to each timestep individually

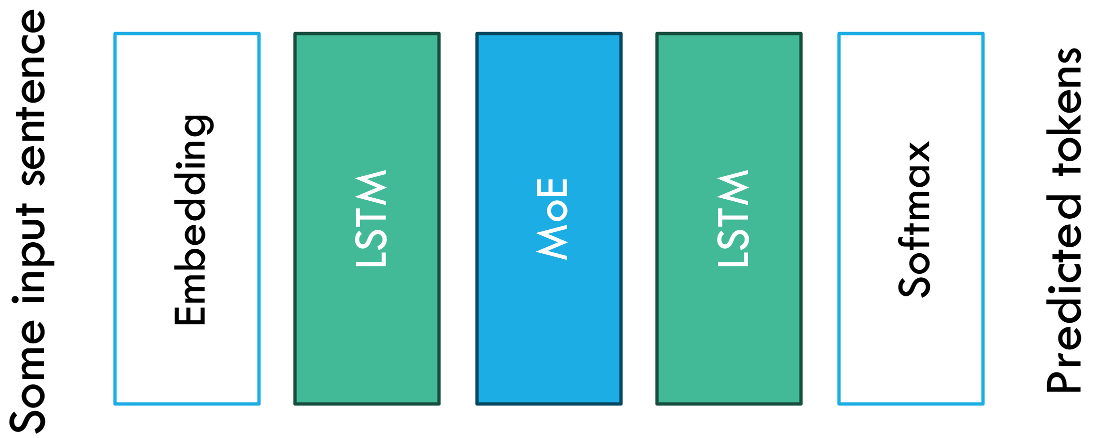
DISTRIBUTED TRAINING

LSTM Layer – data parallel

Gating Network – data parallel

Expert - model parallel

- Only one copy of each expert



BALANCING EXPERT UTILISATION

Without Balancing: gating network tends to converge to a state where it always produces large weights for the same few experts

- self-reinforcing effect, because experts that are selected more often are updated faster

Importance loss

$$Importance(X) = \sum_{x \in X} G(x)$$

$$L_{importance}(X) = w_{importance} \cdot CV(Importance(X))^2$$

BALANCING EXPERT UTILISATION 2

We want to improve load balancing further

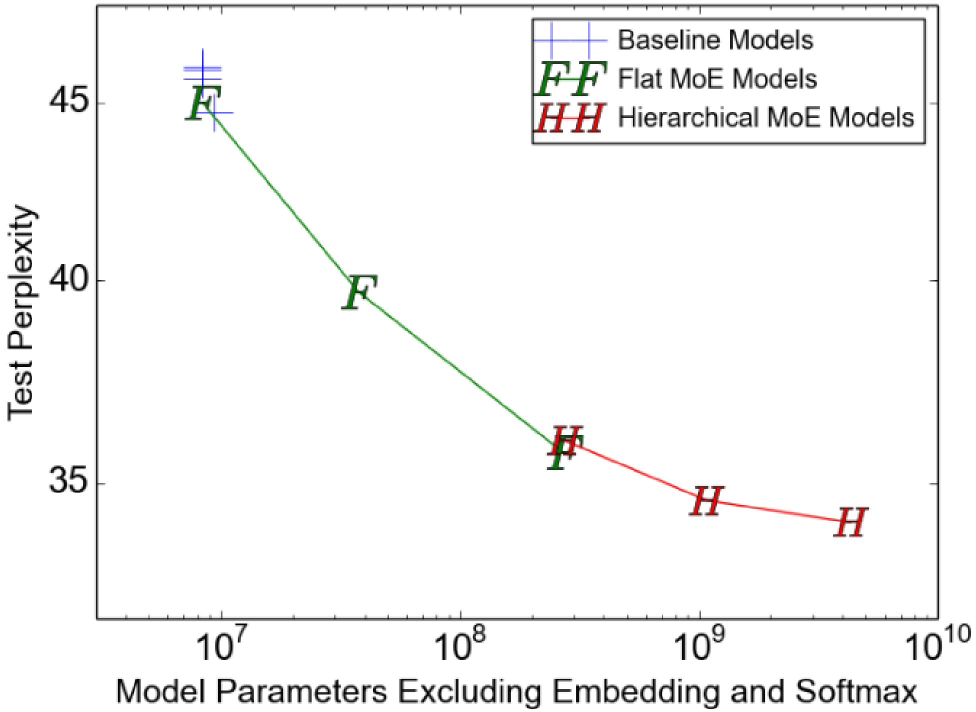
Using the number of invocations would be convenient, but simply counting doesn't yield a gradient

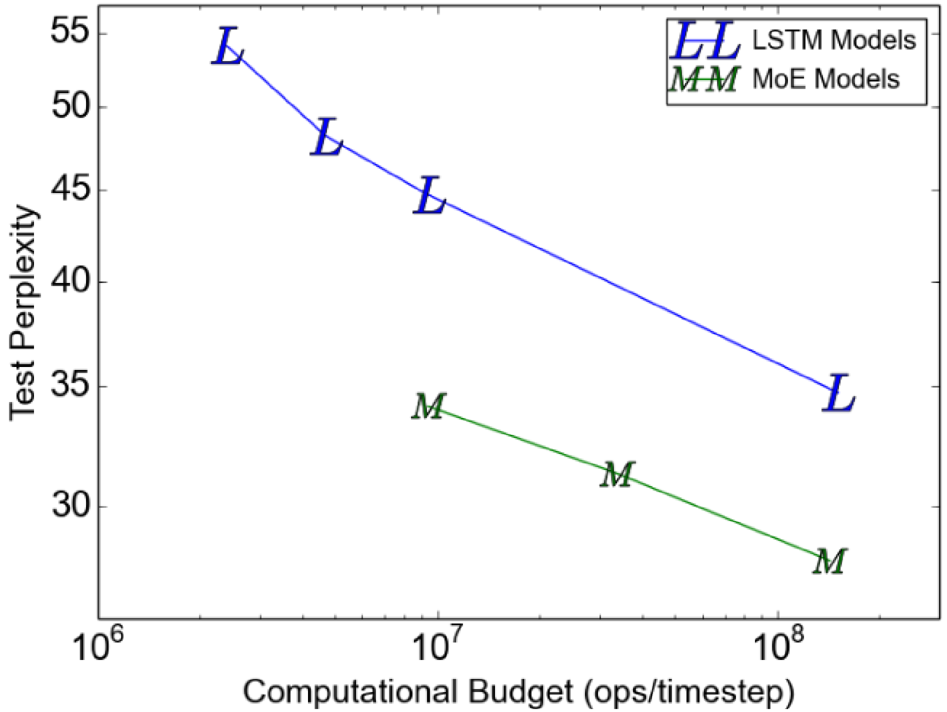
Idea: design an estimator that estimates the number of invocations in a differentiable way

$$\mathrm{P(x, i) = Pr(H(x)_i > kth\_excluding(H(x), k, i))}$$

$$P(x,i) = \Phi\Big(\frac{(x \cdot W_g)_i - kth\_excluding(H(x), k, i)}{Softplus((x \cdot W_{noise})_i)}\Big)$$

$$Load(X)_i = \sum_{x \in X} P(x,i) \qquad L_{load}(X) = w_{load} \cdot CV(Load(X))^2$$


Baseline Models
Flat MoE Models
Hierarchical MoE Models
Test Perplexity
45
40
35
10^7
10^8
10^9
10^{10}
Model Parameters Excluding Embedding and Softmax


LSTM Models
MoE Models
Test Perplexity
55
50
45
40
35
30
10^6
10^7
10^8
Computational Budget (ops/timestep)

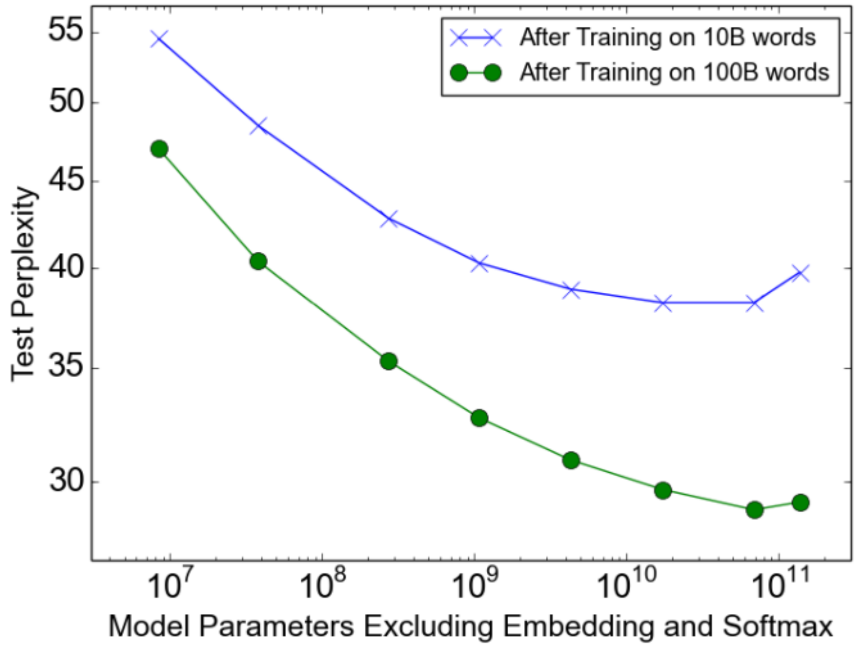


Figure 3: Language modeling on a 100 billion word corpus. Models have similar computational budgets (8 million ops/timestep).

Table 2: Results on WMT'14 En→ Fr newstest2014 (bold values represent best results).

Model	Test Perplexity	Test BLEU	ops/timenstep	Total #Parameters	Training Time
MoE with 2048 Experts	2.69	40.35	85M	8.7B	3 days/64 k40s
MoE with 2048 Experts (longer training)	**2.63**	**40.56**	85M	8.7B	6 days/64 k40s
GNMT (Wu et al., 2016)	2.79	39.22	214M	278M	6 days/96 k80s
GNMT+RL (Wu et al., 2016)	2.96	39.92	214M	278M	6 days/96 k80s
PBMT (Durrani et al., 2014)		37.0			
LSTM (6-layer) (Luong et al., 2015b)		31.5			
LSTM (6-layer+PosUnk) (Luong et al., 2015b)		33.1			
DeepAtt (Zhou et al., 2016)		37.7			
DeepAtt+PosUnk (Zhou et al., 2016)		39.2			

Table 3: Results on WMT'14 En → De newstest2014 (bold values represent best results).

Model	Test Perplexity	Test BLEU	ops/timestep	Total #Parameters	Training Time
MoE with 2048 Experts	**4.64**	**26.03**	85M	8.7B	1 day/64 k40s
GNMT (Wu et al., 2016)	5.25	24.91	214M	278M	1 day/96 k80s
GNMT +RL (Wu et al., 2016)	8.08	24.66	214M	278M	1 day/96 k80s
PBMT (Durrani et al., 2014)		20.7			
DeepAtt (Zhou et al., 2016)		20.6			

Table 5: Multilingual Machine Translation (bold values represent best results).

	GNMT-Mono	GNMT-Multi	MoE-Multi	MoE-Multi vs. GNMT-Multi
Parameters	278M / model	278M	8.7B	
ops/timestep	212M	212M	102M	
training time, hardware	various	21 days, 96 k20s	**12 days, 64 k40s**	
Perplexity (dev)		4.14	**3.35**	-19%
French → English Test BLEU	36.47	34.40	**37.46**	+3.06
German → English Test BLEU	31.77	31.17	**34.80**	+3.63
Japanese → English Test BLEU	23.41	21.62	**25.91**	+4.29
Korean → English Test BLEU	25.42	22.87	**28.71**	+5.84
Portuguese → English Test BLEU	44.40	42.53	**46.13**	+3.60
Spanish → English Test BLEU	38.00	36.04	**39.39**	+3.35
English → French Test BLEU	35.37	34.00	**36.59**	+2.59
English → German Test BLEU	**26.43**	23.15	24.53	+1.38
English → Japanese Test BLEU	**23.66**	21.10	22.78	+1.68
English → Korean Test BLEU	**19.75**	18.41	16.62	-1.79
English → Portuguese Test BLEU	**38.40**	37.35	37.90	+0.55
English → Spanish Test BLEU	34.50	34.25	**36.21**	+1.96

Table 6: Experiments with different combinations of losses.

$w_{importance}$	w_{load}	Test Perplexity	$CV(Importance(X))$	$CV(Load(X))$	$\frac{max(Load(X))}{mean(Load(X))}$
0.0	0.0	39.8	3.04	3.01	17.80
0.2	0.0	**35.6**	0.06	0.17	1.47
0.0	0.2	35.7	0.22	0.04	1.15
0.1	0.1	**35.6**	0.06	0.05	1.14
0.01	0.01	35.7	0.48	0.11	1.37
1.0	1.0	35.7	0.03	0.02	**1.07**