LM again?

Karol Kaczmarek

Adam Mickiewicz University
Poznań

Applica.ai
Warsaw

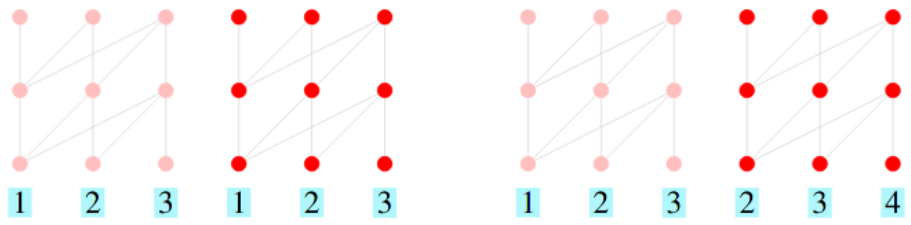
2021

Shortformer [1]

- 31 December 2020, University of Washington - code available (fairseq fork)
- **generative Transformer** (decoder) – 16 layers of dimension 1024 with 16 heads, 4096 dimension of feedforward sublayers and 3072 sequence length
- **Staged Training** – initially training on shorter subsequences
- **Position-Infused Attention** (PIA) – cached representations from the previously subsequence
- tested on **WikiText-103**

Language Model Inference

- Training with 9216 tokens (**sequence length** x **batch size**)
- Inference batch size of 1 (for all tests!)



	Train	Inference			
		Nonoverlapping		Sliding Window (Token-by-token)	
Subseq. Length	Speed ↑	PPL ↓	Speed ↑	PPL ↓	Speed ↑
32	28.3k	35.37	2.4k	24.98	**74**
64	28.5k	28.03	4.8k	21.47	69
128	**28.9k**	23.81	9.2k	19.76	70
256	28.1k	21.45	14.8k	18.86	63
512	26.1k	20.10	18.1k	18.41	37
1024	22.9k	19.11	**18.3k**	17.97	18
1536	18.4k	19.05	17.1k	18.14	11
3072	13.9k	**18.65**	14.7k	**17.92**	5

Staged Training

- **two-stage training** routine that initially uses short input subsequences followed by long subsequences (in the second use 3072 tokens)
- **sinusoidal position embeddings**, learned position embeddings would create a dependency between the parameterization and the subsequence length
- **neither modify nor reset the state of the optimization** algorithm between the two stages

Staged Training – results on dev set (205 epochs)

only 37% of baseline time

Initial Stage Epochs	Initial Stage Subsequence Length						
	32	64	128	256	512	1024	1536
25	0.60	0.54	0.53	0.65	0.64	0.71	
50	0.53	0.48	0.47	0.54	0.59	0.63	0.81
75	0.51	0.43	0.42	0.48	0.53	0.56	0.79
100	0.52	0.40	0.38	0.41	0.47	0.50	0.73
125	0.61	0.41	**0.37**	**0.37**	0.42	0.46	0.69
150		0.48	0.39	**0.37**	0.40	0.44	0.66
175			0.48	0.43	0.45	0.51	0.70
200						0.59	

Time until match the baseline

1.1 perplexity better in 87% time

Initial Stage Epochs	Initial Stage Subseqence Length						
	32	64	128	256	512	1024	1536
25	17.94	17.57	17.58	18.19	18.06	18.20	18.77
50	17.81	17.59	**17.52**	18.08	18.01	18.14	18.62
75	17.93	17.61	17.55	18.01	18.05	18.03	18.57
100	18.14	17.67	17.62	18.00	18.10	18.00	18.51
125	18.61	17.88	17.70	18.00	18.13	17.98	18.49
150	19.45	18.37	17.98	18.01	18.15	18.00	18.49
175	21.16	19.51	18.57	18.23	18.20	18.08	18.57
200	35.38	28.03	23.80	21.45	19.63	18.56	18.84

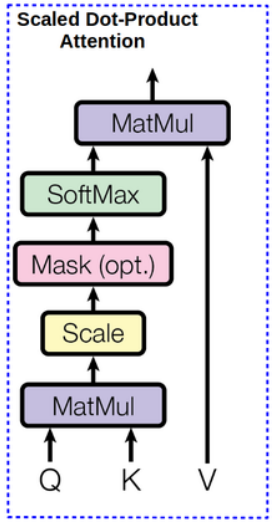
Perplexity at the end of training – base line 18,65

using more than two stages (up to six), did not outperform

Repositioning Position Embeddings

- inspired by **slide window evaluation** (very slow evaluation) and **relative position embeddings** (slow inference by 22% and require 26% more parameters)
- **Position-Infused Attention (PIA)** - reuse previous outputs by making each output contain **no positional information**:
  - without using extra parameters, memory, or run time
  - can use much shorter input subsequences and still achieve performance

Position-Infused Attention (PIA) - positional embeddings



- do not add position embeddings at **the beginning**, but rather adds position embeddings to **the Query** and **Key** vectors at each layer (but **not to the Value**)
- outputs at each layer are weighted sums of **the value vectors without positional information** (value vectors do not contain positional information)

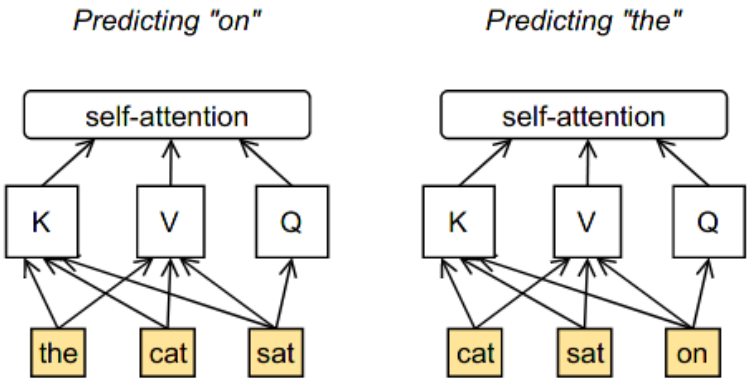
Position-Infused Attention (PIA) - visualization

1. Embed tokens: X
2. Add positional embeddings: $X = X + P$

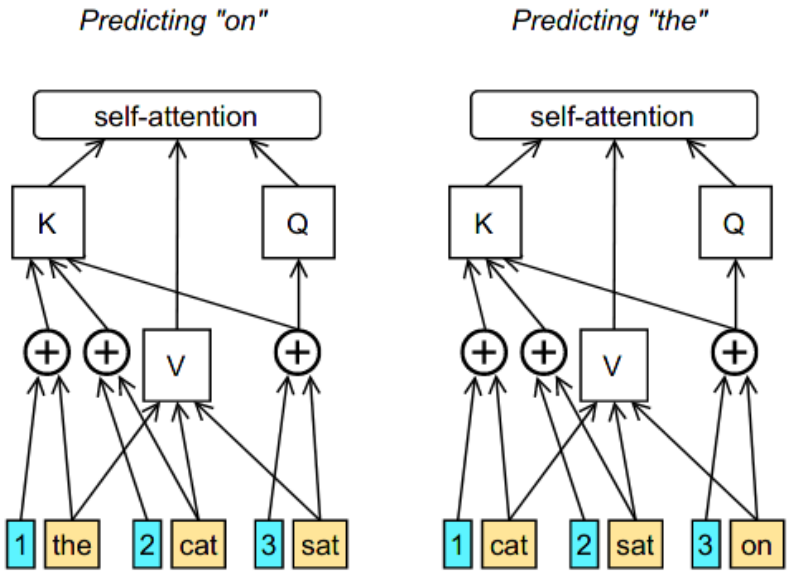
1. Embed tokens: X
2. Do not add position embeddings

3. Feed transformer layer where self-attention:

Key=X, Query=X, Value=X

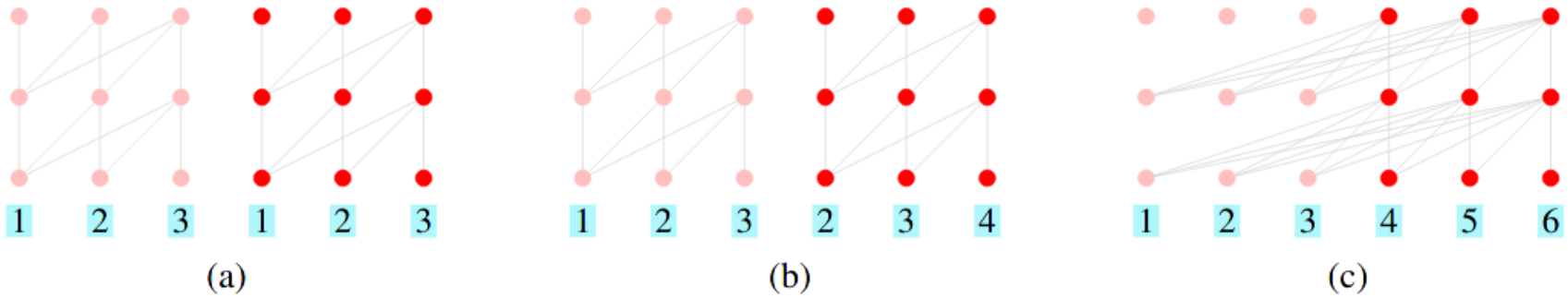


Key=$X + P$, Query=$X + P$, Value=X



X - embedded tokens, P - position embedding

Visualization of evaluation/training



- (a) - nonoverlapping training and inference
- (b) - sliding window inference with stride S = 1
- (c) - caching (In the next iteration tokens 4, 5 and 6 become the cache, and are **given position embeddings 1, 2 and 3**, while the three new tokens **get position embeddings 4, 5, and 6**)

Position-Infused Attention (PIA) with cache - score

- **baseline** achieves **18.65** on the development set
- only PIA method (size as baseline) achieves 19.35 perplexity (speed and memory usage do not change)
- using PIA and caching (cached equal to **Subseq. Length**):

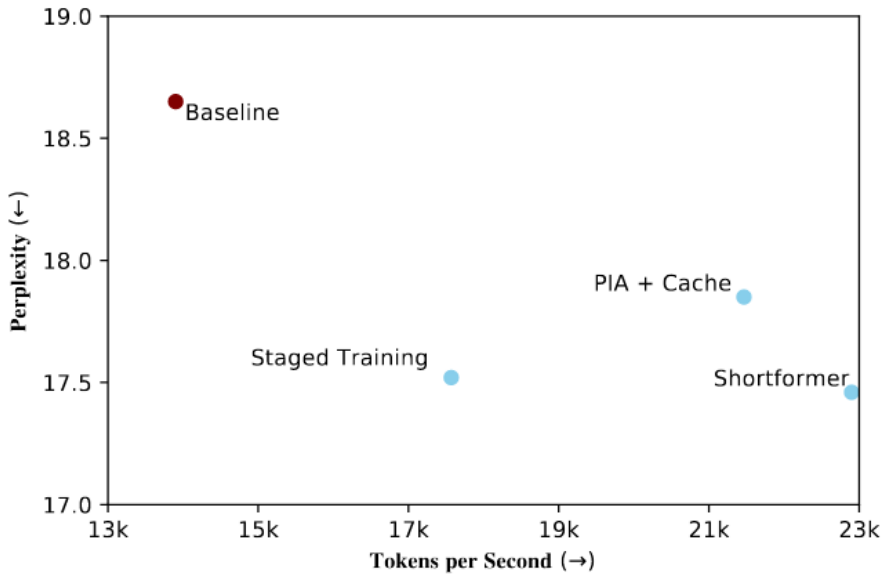
Subseq. Length	Train	Inference		
			Speed ↑	
	Speed ↑	PPL ↓	N.o.	S.W.
32	22.0k	20.53	2.0k	49
64	23.8k	19.07	4.1k	**51**
128	**24.4k**	18.37	7.9k	50
256	23.5k	17.92	12.8k	48
512	21.5k	**17.85**	14.5k	46
768	17.6k	18.16	13.8k	43
1024	16.6k	18.19	13.9k	39
1536	12.9k	19.11	7.9k	34
Baseline	13.9k	18.65	**14.7k**	-
		17.92	-	5

N.o. is Nonoverlapping, **S.W.** is Sliding Window, batch size of 1 for inference, best model trains 55% faster

PIA with Staged Training - score

- using **Position-Infused Attention** and **Staged Training** on the previous best model (PIA model with 512 tokens)
- the best model (**Shortformer**) achieves 17.47 perplexity and trains 65% faster than the baseline model (18.65 perplexity) on the development set

First Stage Subseq. Length	Train Speed ↑	Inference PPL ↓
32	21.6k	17.66
64	22.6k	17.56
128	**22.9k**	**17.47**
256	22.5k	17.50
PIA without Staged Training	21.5k	17.85



Compare to the other LMs

- evaluate on the test set of WikiText-103
- **Shortformer** is almost twice as fast to train and is nine times faster in token-by-token generation
- sliding window evaluation cannot be used in **Shortformer** because **caching** and **PIA** were used

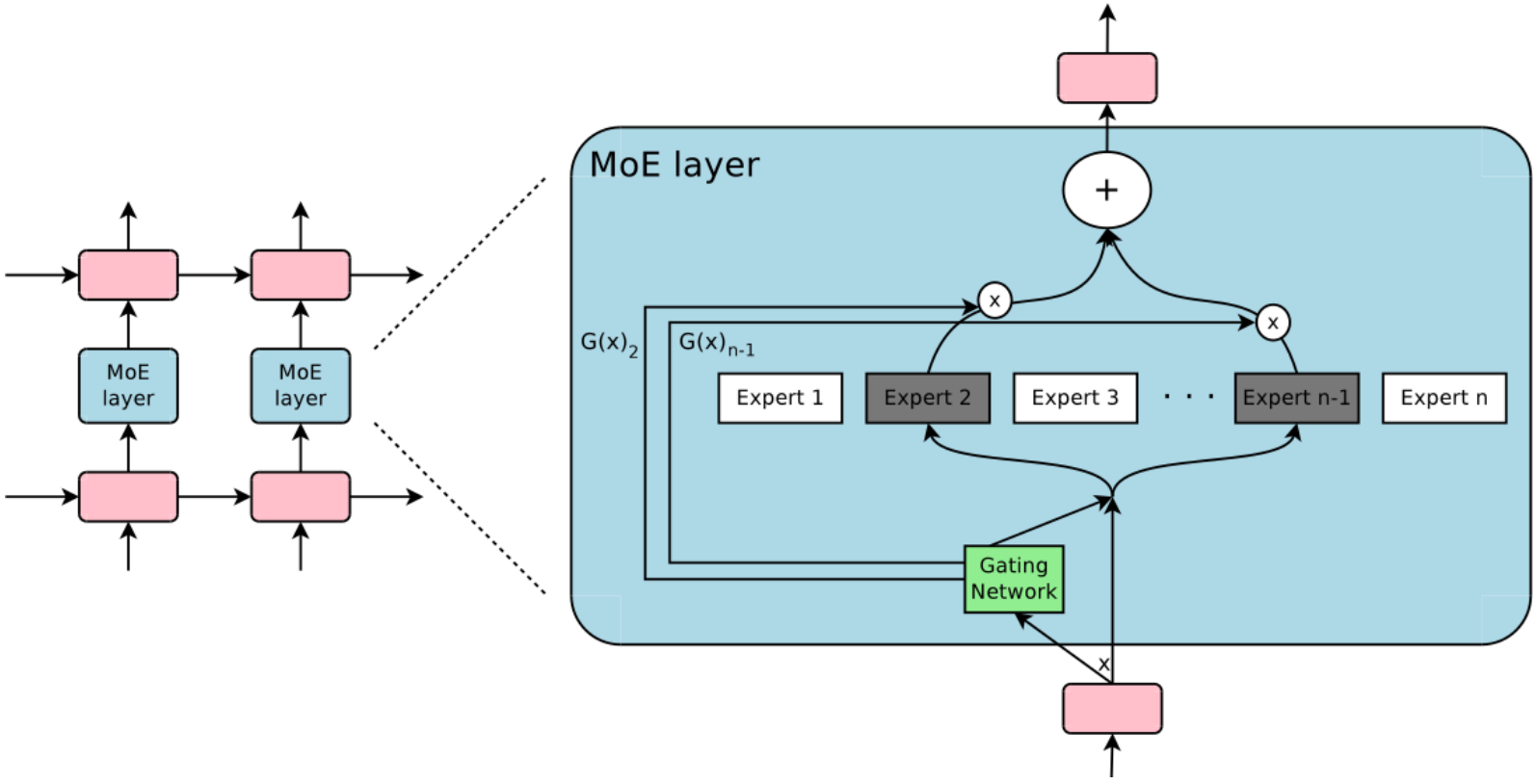
	Train	Inference (Test)		
Model	Speed ↑	Mode	Speed ↑	PPL ↓
Baseline	13.9k	N.o.	**14.7k**	19.4
		S.W.	2.5k	18.70
TransformerXL*	6.0k	N.o.	3.2k	18.30
Sandwich Transformer	13.9k	S.W.	2.5k	17.96
kNN-LM**	13.9k	S.W.	145	**15.79**
Baseline + Staged Train.	17.6k	S.W.	2.5k	17.56
Shortformer	**22.9k**	N.o.	14.5k	18.15

*runs on an older version of PyTorch than all other models, which might impact speed
**requires a 400GB datastore

Sparsely-Gated Mixture-of-Experts (MoE) [2]

- January 2017, Google + Jagiellonian University
- **Sparsely-Gated Mixture-of-Experts layer** (MoE) consisting of up to thousands of feed-forward sub-networks
- trainable **gating network** determines a sparse combination of these experts to use for each example
- achive greater than **1000x** improvements in model capacity with only minor losses in computational efficiency on modern GPU clusters
- apply the MoE to the tasks of **language modeling** and **machine translation**

Mixture of Experts (MoE) layer between stacked LSTM layers

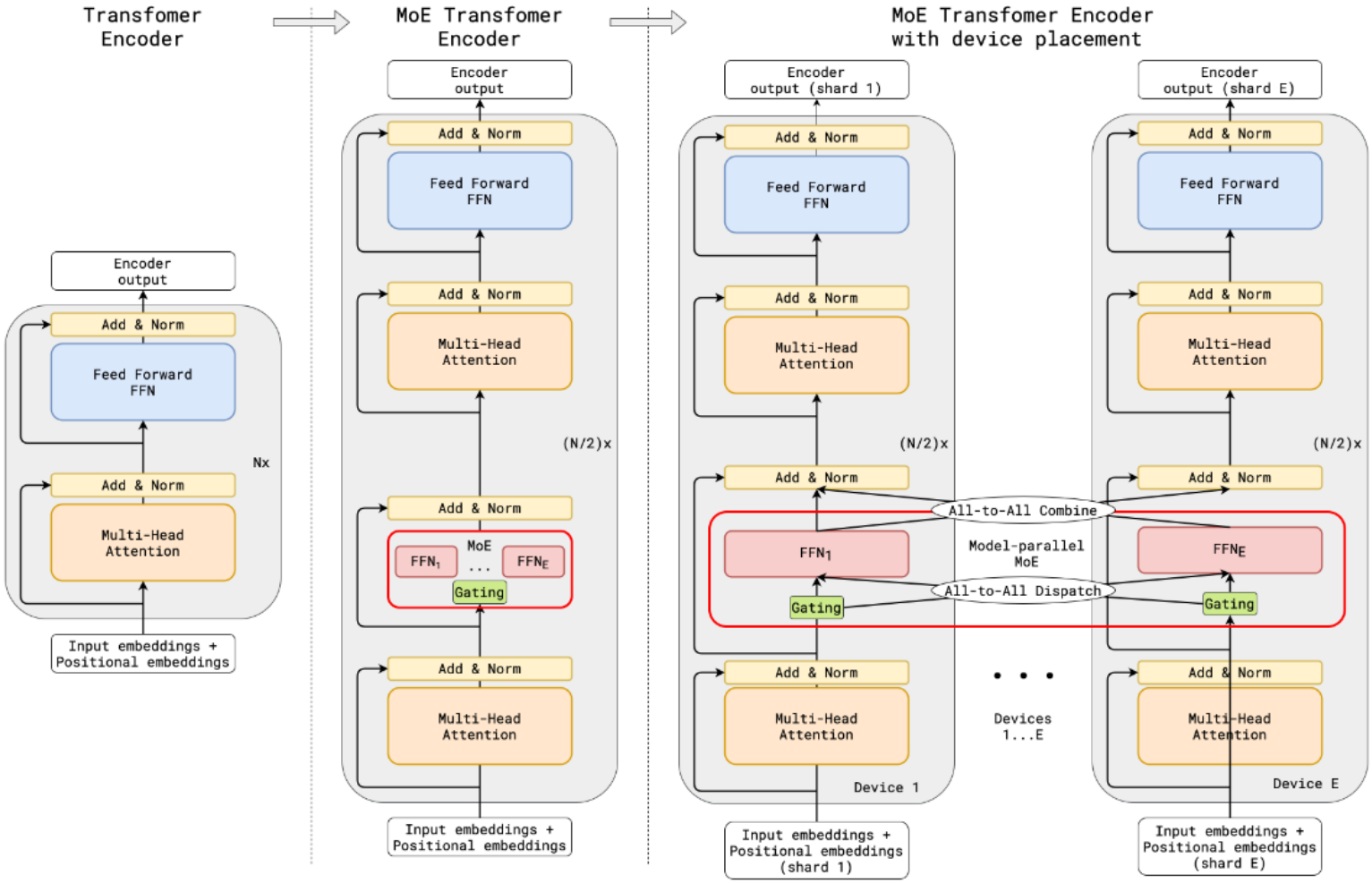


- **sparse gating** function selects **two experts** to perform computations
- outputs are **modulated** by the outputs of the **gating network**

GShard [3]

- June/July 2020, Google - pseudocode
- parallel computation patterns with **minimal changes** to the existing model code GShard enabled us to scale up multilingual neural machine translation Transformer model with Sparsely-Gated Mixture-of-Experts beyond **600 billion parameters** using automatic sharding
- train 600B model on 2048 TPU v3 accelerators in 4 days to achieve far superior quality for translation from 100 languages to English
- even train 1 trillion model (**problem with stability training**, did not include the results)

Sparsely-Gated Mixture-of-Experts Transformer (MoE Transformer)



Transformer model

- Encoder-Decoder transformer
- replace every **Position-Wise Feed-Forward** layer with a **Position-wise Mixture of Experts** (MoE) layer
- **Position-wise Mixture-of-Experts** (MoE) layer - based on MoE [2] with variations in the sparse gating function and feed-forward network with ReLU activation
- MoE layer consists of E feed-forward network FFN_1 ... FNN_E (recommended on expert per device)

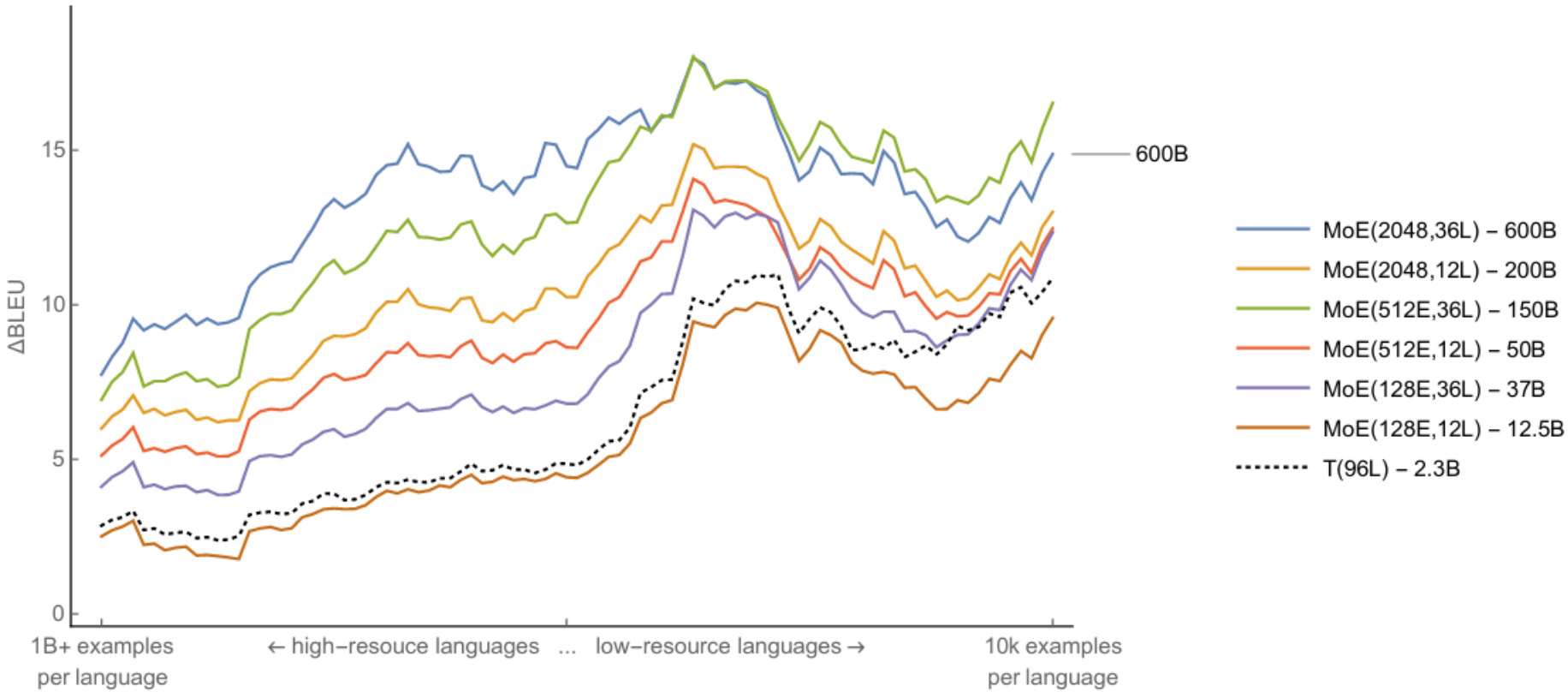
Position-wise Mixture-of-Experts Layer

$$G_{s,E} = \text{GATE}\ (x_s)$$
$$\text{FFN}_e(x_s) = wo_e \cdot \text{ReLU}\ (wi_e \cdot x_s)$$
$$y_s = \sum_{e=1}^{E} G_{s,e} \cdot \text{FFN}_e(x_s)$$

x_s - input token to the MoE layer, w_i, wo - projection matrices in FFN (an expert)

- $G_{s,E}$ - computed by a **gating network**, has one non-negative for each expert, most of which are zeros (token is not dispatched), token is dispatched to a very small number of experts (at most two experts). Representing how much an expert contributes to the network output.
- Every expert FFN_e applies to x_s a **fully-connected 2-layer** network using **ReLU activation** function.
- Output of the MoE layer, y_s, is **the weighted** average of outputs from all the selected experts.
- Gating function GATE(·) is modeled by a softmax activation function.

Massively Multilingual, Massive Machine Translation (M4) - score



M4 - score

Id	Model	Experts Per-layer	Experts total	TPU v3 Cores	Enc+Dec layers	Weights
(1)	MoE(2048E, 36L)	2048	36684	2048	36	600B
(2)	MoE(2048E, 12L)	2048	12228	2048	12	200B
(3)	MoE(512E, 36L)	512	9216	512	36	150B
(4)	MoE(512E, 12L)	512	3072	512	12	50B
(5)	MoE(128E, 36L)	128	2304	128	36	37B
(6)	MoE(128E, 12L)	128	768	128	12	12.5B
*	MoE(2048E, 60L)	2048	61440	2048	60	1T

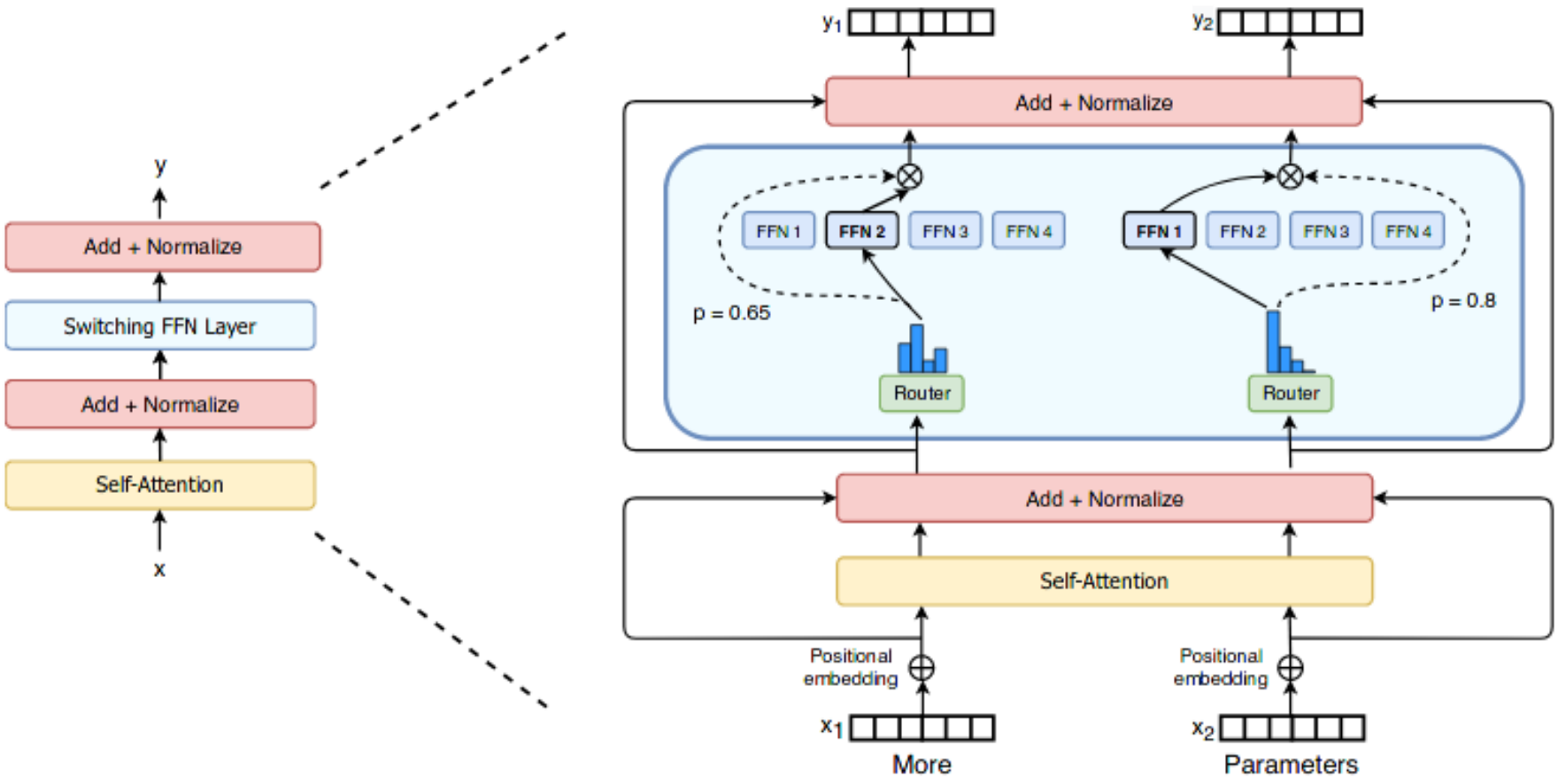
Id	Model	BLEU avg.	ΔBLEU avg.	Weights
(1)	MoE(2048E, 36L)	**44.3**	13.5	600B
(2)	MoE(2048E, 12L)	41.3	10.5	200B
(3)	MoE(512E, 36L)	43.7	12.9	150B
(4)	MoE(512E, 12L)	40.0	9.2	50B
(5)	MoE(128E, 36L)	39.0	8.2	37B
(6)	MoE(128E, 12L)	36.7	5.9	12.5B
*	T(96L)	36.9	6.1	2.3B
*	Baselines	30.8	-	100×0.4B

Id	Model	Cores	Steps per sec.	Batch sz. (Tokens)	TPU core years	Training time (days)	BLEU avg.
(1)	MoE(2048E, 36L)	2048	0.72	4M	22.4	**4.0**	**44.3**
(2)	MoE(2048E, 12L)	2048	2.15	4M	7.5	1.4	41.3
(3)	MoE(512E, 36L)	512	1.05	1M	15.5	11.0	43.7
(4)	MoE(512E, 12L)	512	3.28	1M	4.9	3.5	40.0
(5)	MoE(128E, 36L)	128	0.67	1M	6.1	17.3	39.0
(6)	MoE(128E, 12L)	128	2.16	1M	1.9	5.4	36.7
*	T(96L)	2048	-	4M	∼235.5	∼42	36.9

Switch Transformers [4]

- January 2021, Google - code available (Mesh TensorFlow)
- Transformer Encoder with MoE (**Mixture-of-Experts**) layers
- **7x** increases in pre-training speed (compared to T5 model)
- scale up to **trillion parameter** models on C4 (Colossal Clean Crawled Corpus - 180B tokens) dataset

Architecture



Improved training and fine-tuning techniques

- **selective precision with large sparse models** - selective casting (cast the local routing operations to float32 - broadcasting bfloat16 then cast to float32)

Model (precision)	Quality (Neg. Log Perp.)	Speed (Examples/sec)
Switch-Base (float32)	-1.718	1160
Switch-Base (bfloat16)	-3.780 [*diverged*]	**1390**
Switch-Base (Selective precision)	**-1.716**	1390

Improved training and fine-tuning techniques

- **smaller parameter initialization for stability** - use truncated normal distribution and reduce the default initialization scale factor

Model (Initialization scale)	Average Quality (Neg. Log Perp.)	Std. Dev. of Quality (Neg. Log Perp.)
Switch-Base (0.1x-init)	**-2.72**	**0.01**
Switch-Base (1.0x-init)	-3.60	0.68

Improved training and fine-tuning techniques

- **regularizing large sparse models** - increase the dropout inside the experts (expert dropout) during fine-tuning (bigger models can lead to more severe overfitting on smaller downstream tasks)

Model (dropout)	GLUE	CNNDM	SQuAD	SuperGLUE
T5-Base (d=0.1)	82.9	**19.6**	83.5	72.4
Switch-Base (d=0.1)	84.7	19.1	**83.7**	**73.0**
Switch-Base (d=0.2)	84.4	19.2	**83.9**	**73.2**
Switch-Base (d=0.3)	83.9	19.6	83.4	70.7
Switch-Base (d=0.1, ed=0.4)	**85.2**	**19.6**	**83.7**	**73.0**

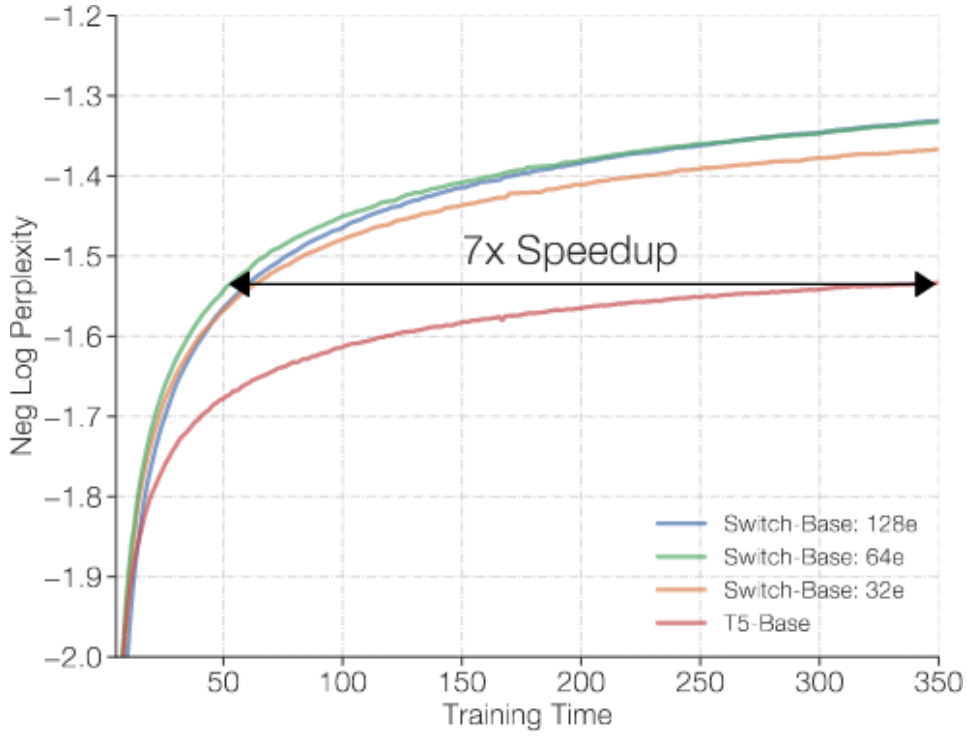
Pre-training on C4 dataset

- 128 experts, 100k steps only

Model	Capacity Factor	Quality after 100k steps (Neg. Log Perp.)	Time to Quality Threshold (hours)	Speed (examples/sec)
T5-Base	—	-1.731	Not achieved†	1600
T5-Large	—	-1.550	131.1	470
MoE-Base	2.0	-1.547	68.7	840
Switch-Base	2.0	-1.554	72.8	860
MoE-Base	1.25	-1.559	80.7	790
Switch-Base	1.25	-1.553	65.0	910
MoE-Base	1.0	-1.572	80.1	860
Switch-Base	1.0	-1.561	**62.8**	1000
Switch-Base (expand)	1.0	**-1.534**	67.6	780

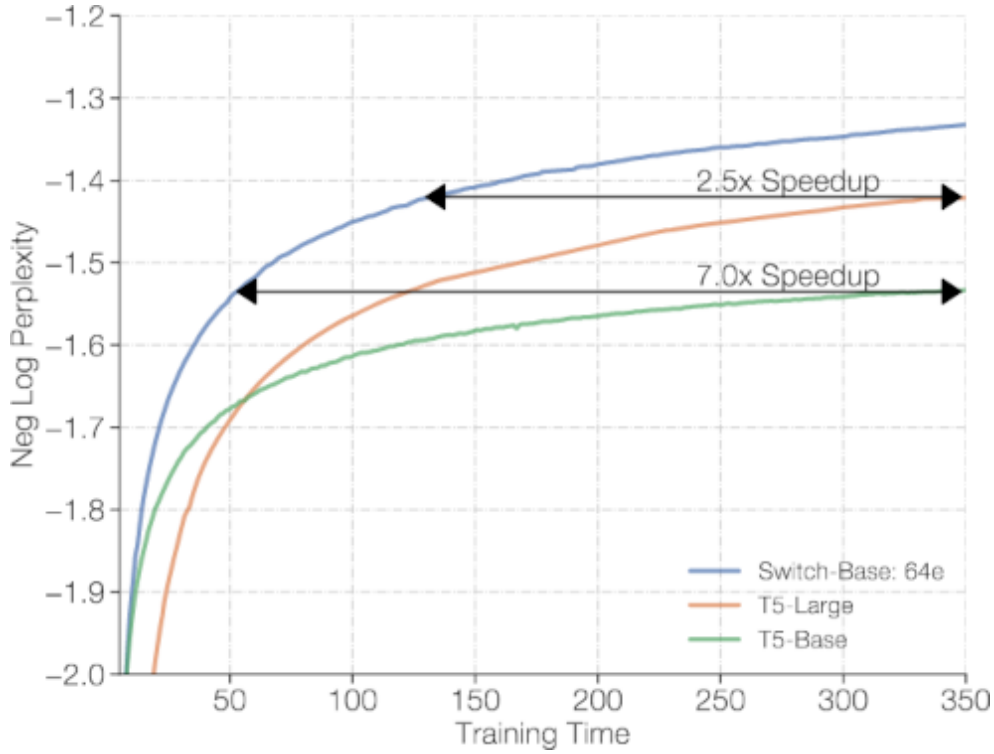
Model	Parameters	FLOPS
T5-Base	223M	124B
Switch-Base	7.4B	124B
T5-Large	739M	425B
Switch-Large	26.3B	425B

Speed up pre-training (with fixed FLOPS per token)



- **Switch-Base 64** expert model achieves the same performance of the **T5-Base** model at step 60k at step 450k, which is a **7.5x** speedup in terms of step time

Speed up pre-training



- **T5-Large** applying **3.5x** more FLOPs per token
- **Switch-Base** is still more sample efficient and yields a **2.5x** speedup

Fine-tuning results (validation set)

Model	GLUE	SQuAD	SuperGLUE	Winogrande (XL)
T5-Base	84.3	85.5	75.1	66.6
Switch-Base	**86.7**	**87.2**	**79.5**	**73.3**
T5-Large	87.8	88.1	82.7	79.1
Switch-Large	**88.5**	**88.6**	**84.7**	**83.0**

Model	Parameters	FLOPS
T5-Base	223M	124B
Switch-Base	7.4B	124B
T5-Large	739M	425B
Switch-Large	26.3B	425B

- **T5** and **Switch** models were pre-trained on a C4 dataset

Distillation (base on DistilBERT - Hugging Face)

Technique	Parameters	Quality
T5-Base	223M	-1.636
Switch-Base	3,800M	-1.444
Distillation	223M (3%)	-1.631
+ Init. non-expert weights from teacher	223M (20%)	-1.598
+ 0.75 mix of hard and soft loss	223M (29%)	-1.580
Initialization Baseline (no distillation)		
Init. non-expert weights from teacher	223M	-1.639

- pre-training distillation
- initialize the dense model with the non-expert weights
- use a mixture of 0.25 for the teacher probabilities and 0.75 for the ground truth label
- no improvement of **T5-Base** initialized with the expert weights

Distillation compression rates

	Dense	Sparse				
Parameters	223M	1.1B	2.0B	3.8B	7.4B	14.7B
Pre-trained Neg. Log Perp.	-1.636	-1.505	-1.474	-1.444	-1.432	-1.427
Distilled Neg. Log Perp.	—	-1.587	-1.585	-1.579	-1.582	-1.578
Percent of Teacher Performance	—	37%	32%	30 %	27 %	28 %
Compression Percent	—	82 %	90 %	95 %	97 %	99 %

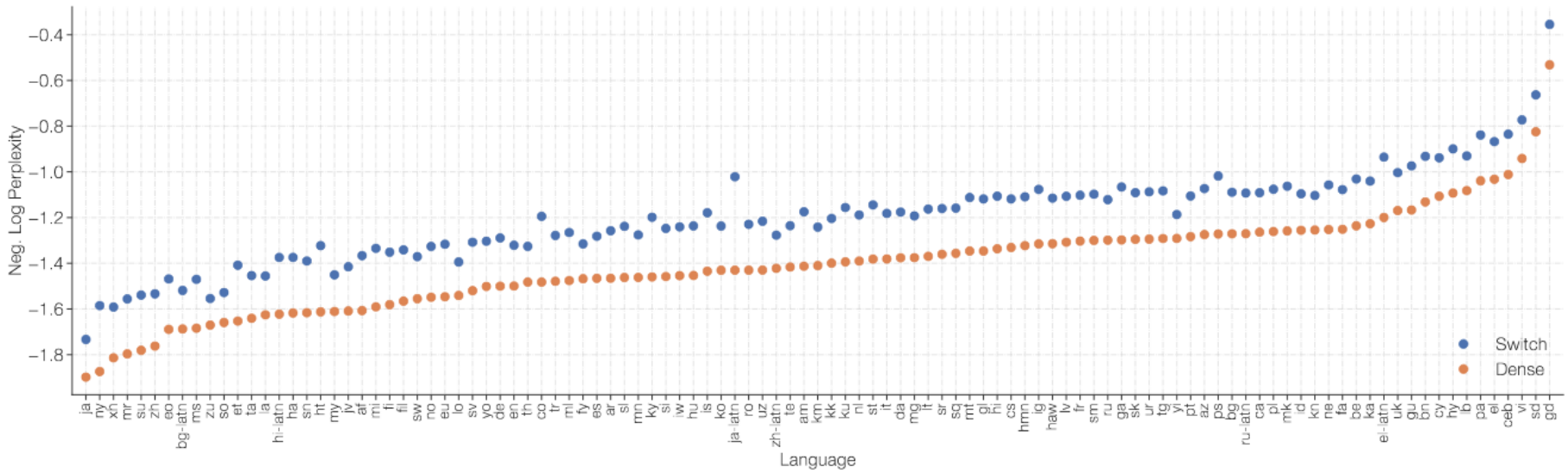
- **preserve 37% of the quality** gain of the 1.1B parameter model while **compressing 82%**
- the extreme, preserve 28% of the quality gain of the 14.7B parameter model while compressing 99%

Distilling a fine-tuned mode

Model	Parameters	FLOPS	SuperGLUE
T5-Base	223M	124B	74.6
Switch-Base	7410M	124B	81.3
Distilled T5-Base	223M	124B	(30%) 76.6

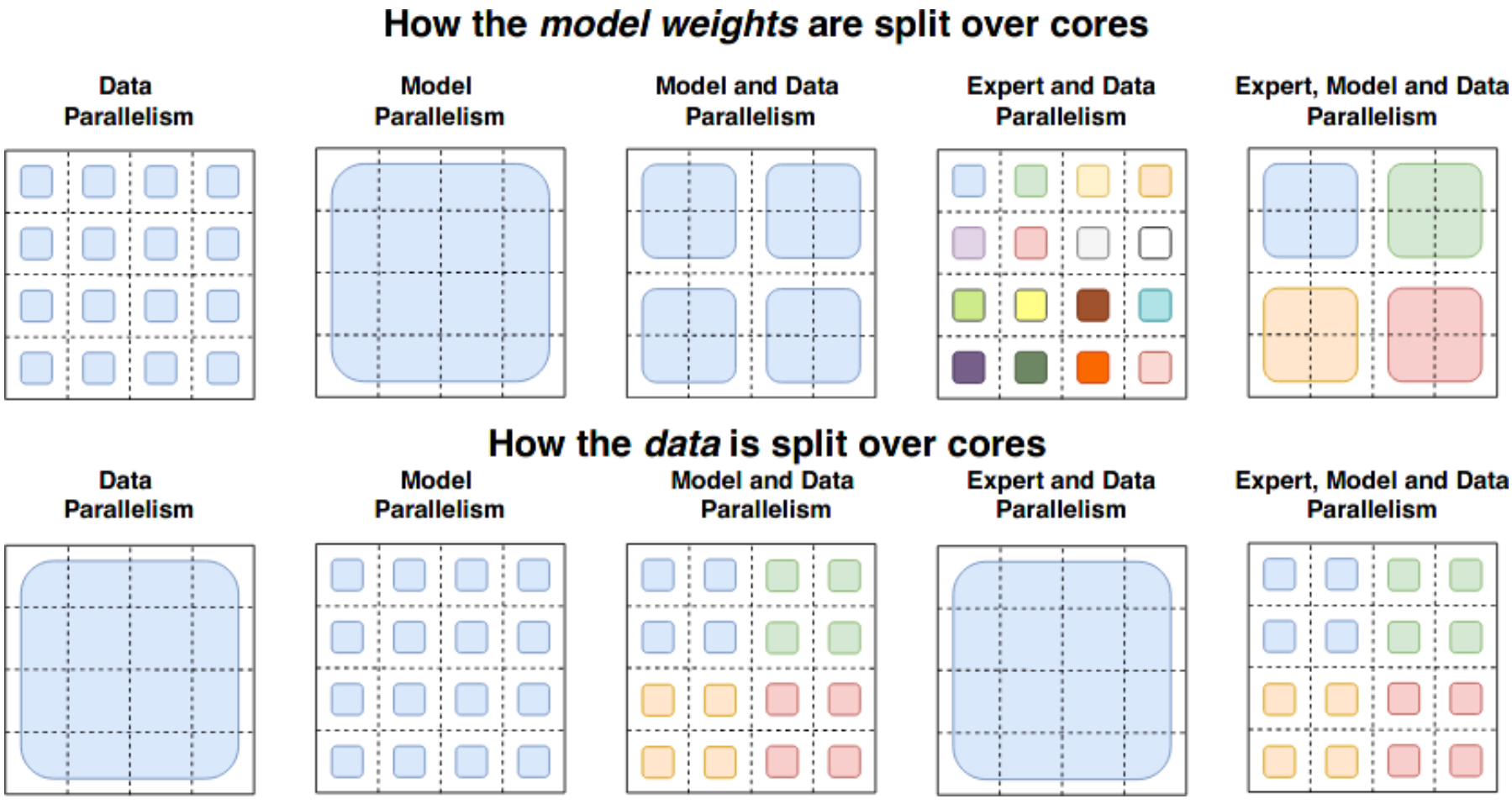
- fine-tuned sparse model into a dense model
- on smaller datasets large sparse model can be an effective teacher for distillation
- achieve 30% of the teacher's performance on a 97% compressed model

Multilingual pre-training on 101 languages (mT5)

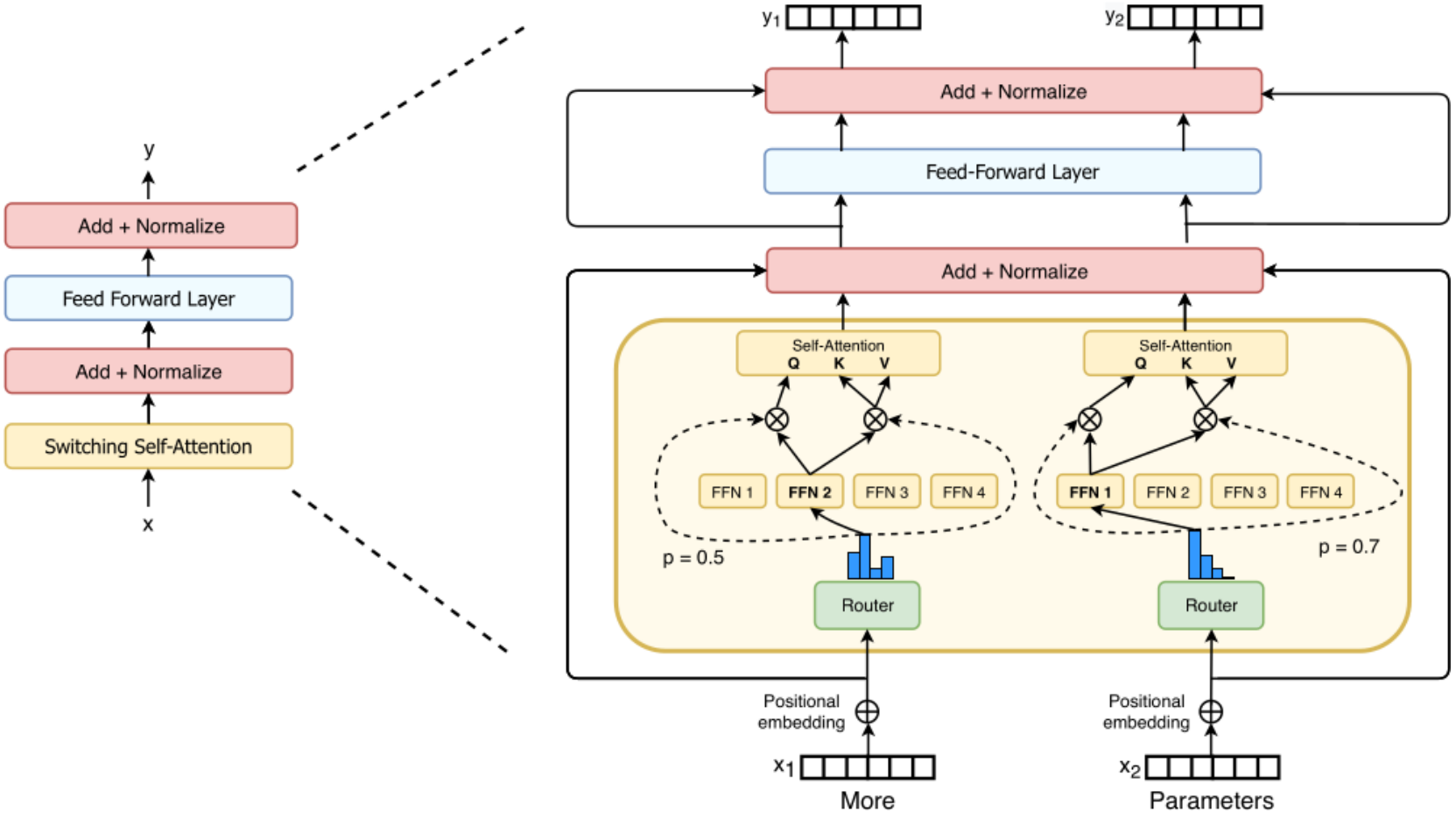


- pre-train on the multilingual variant of the Common Crawl dataset (mC4)
- **mSwitch-Base** is 5x faster than **mT5-Base** (4x on 91% of languages)

How the model weights/data are split?



Switch for Attention

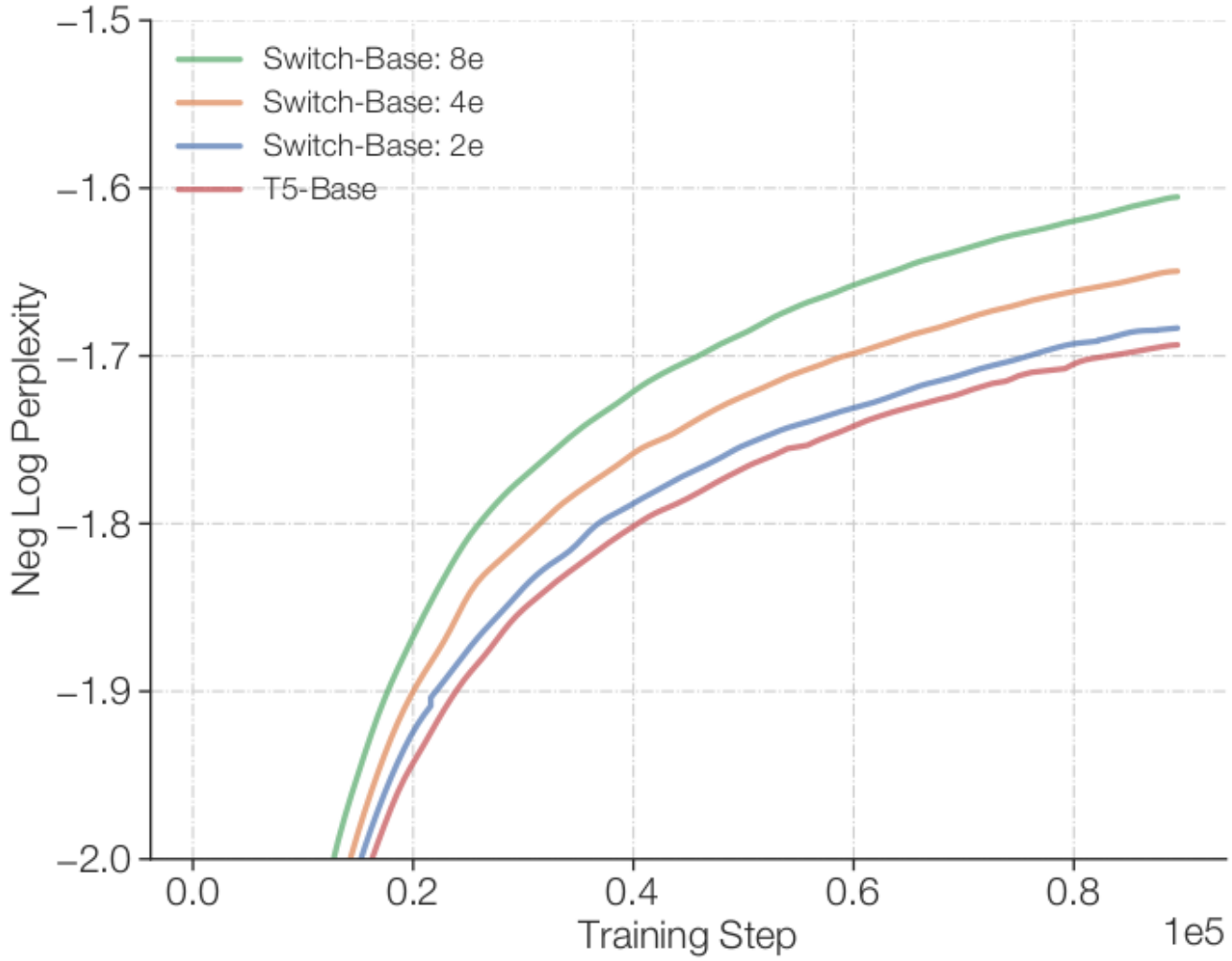


Switch for Attention

- add **Switch layers** into the **Transformer Self-Attention layers**
- replace the **trainable weight matrices that produce the queries, keys and values** with **Switch layers** (one set of weights produces the query and the other set of unique weights produces the shared keys and values)
- found this to be **more unstable** when used with low precision number formats, and thus leave it for future work

Model	Precision	Quality @100k Steps	Quality @16H	Speed (ex/sec)
Experts FF	float32	-1.548	-1.614	1480
Expert Attention	float32	-1.524	**-1.606**	1330
Expert Attention	bfloat16	[diverges]	[diverges]	–
Experts FF + Attention	float32	**-1.513**	-1.607	1240
Expert FF + Attention	bfloat16	[diverges]	[diverges]	–

Switch Transformer in lower compute regimes (recommend one expert per core)

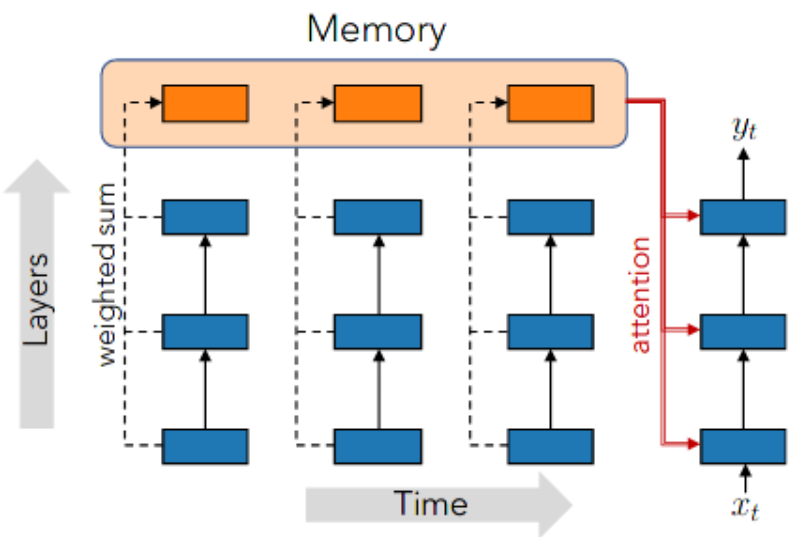


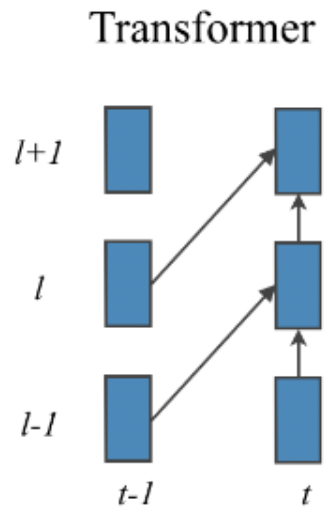
Feedback Transformers [5]

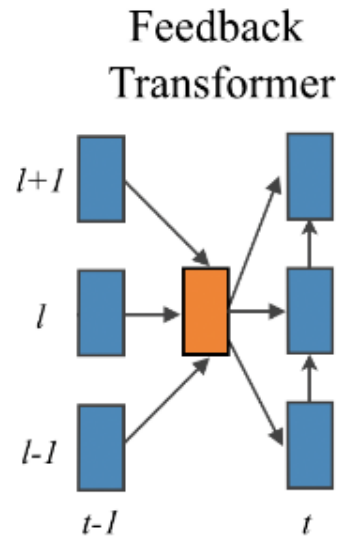
- 2020, Facebook - no code available
- Transformer Decoder (autoregressive model)
- Feedback memory (exposes all previous representations to all future representations) - recursive computation
- tested on language modeling, machine translation and reinforcement learning
- Feedback Transformer is not much slower compared to the standard Transformer

Feedback Transformer

- makes **all previous hidden representations accessible** to the computation of a representation at any depth
- merges **the hidden states from all layers into a single vector** for every time step and stores them in a memory (every previously computed representation is accessible)



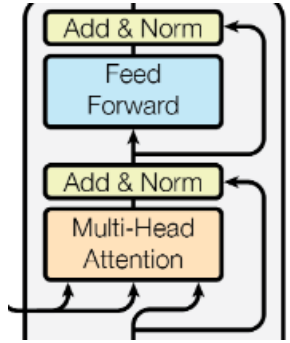




Feedback memory

- uses less memory because **all the layers share a single Feedback memory**, thus **reducing the memory** size by L times (L is the number of layers)
- **uses less computation** because **share the key** and **value projections** during attention computation, which increases the speed of the attention
- GPU memory usage is reduced due to the **memory sharing** — the overall model is 2x smaller

Feedback Transformer Attention



Transformer Attention = Attn(Q, VK)

$$z_t^l = \text{Attn}\left(x_t^l, \{x_{t-\tau}^l, ..., x_{t-1}^l\}\right) \qquad x_t^{l+1} = \text{FF}\left(z_t^l\right)$$

Feedback Transformer Attention

$$z_t^l = \text{Attn}\left(x_t^l, \{\mathbf{m}_{t-\tau}^l, ..., \mathbf{m}_{t-1}^l\}\right) \qquad x_t^{l+1} = \text{FF}\left(z_t^l\right)$$

$$m_t = \sum_{l=0}^{L} \text{Softmax}\left(w^l\right) x_t^l$$

- sharing the **key** and **value** projections W_k^l and W_v^l across all layers: $k_l^t = k_t = W_k m_t$ and $v_t^l = v_t = W_v m_t$

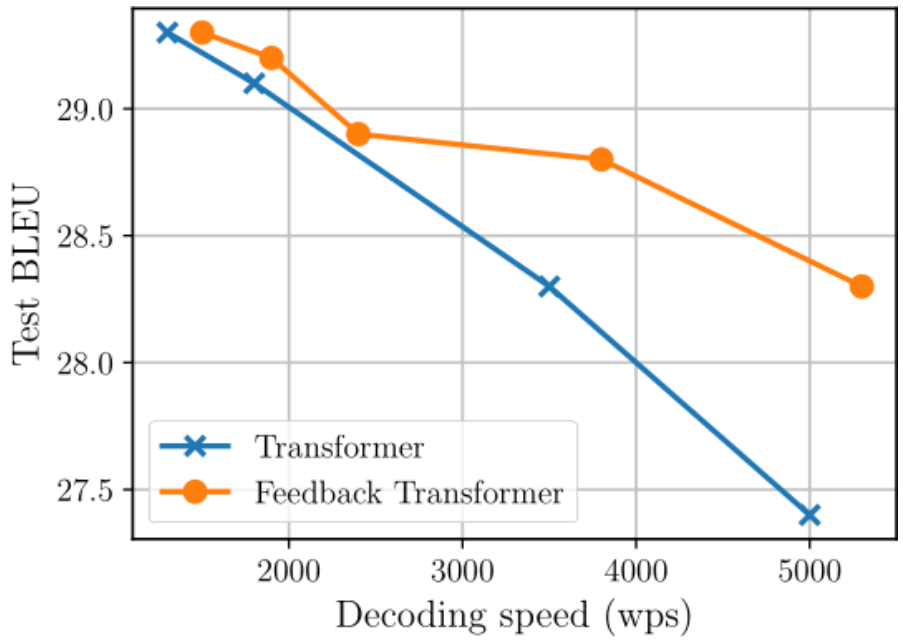
l - layer of Transformer, $x_{t-\tau}^l, ..., x_{t-1}^l$ - past context/vectors, t - sequence length, **m** - memory vectors, w^l - learnable scalar parameters (flexibility in averaging/selecting layers)

Score - accuracy on toy tasks

Task		Trans-former	Feedback Trans.
Copy	Char	59.1	76.2
	Seq	6.2	23.6
Reverse	Char	50.2	74.8
	Seq	5.9	29.2
Counting	Len 50	99.6	99.7
	Len 1K	82.4	95.3
Random Walk		68	100
Algorithmic	3 vars	33.7	99.1
	5 vars	37.5	92.6

- **Copy and Reverse** - models read the input and then either copy or reverse
- **Counting** - models have a sequence of A in a row, and must output the corresponding quantity of the letter B
- **Random Walk** - random walk in a small grid where the goal is calculate the current position
- **Algorithmic task** - tracking and updating value of variables in code execution (needs keep track of all variable values and update them if necessary)

Score - Machine Translation on WMT14 En-De



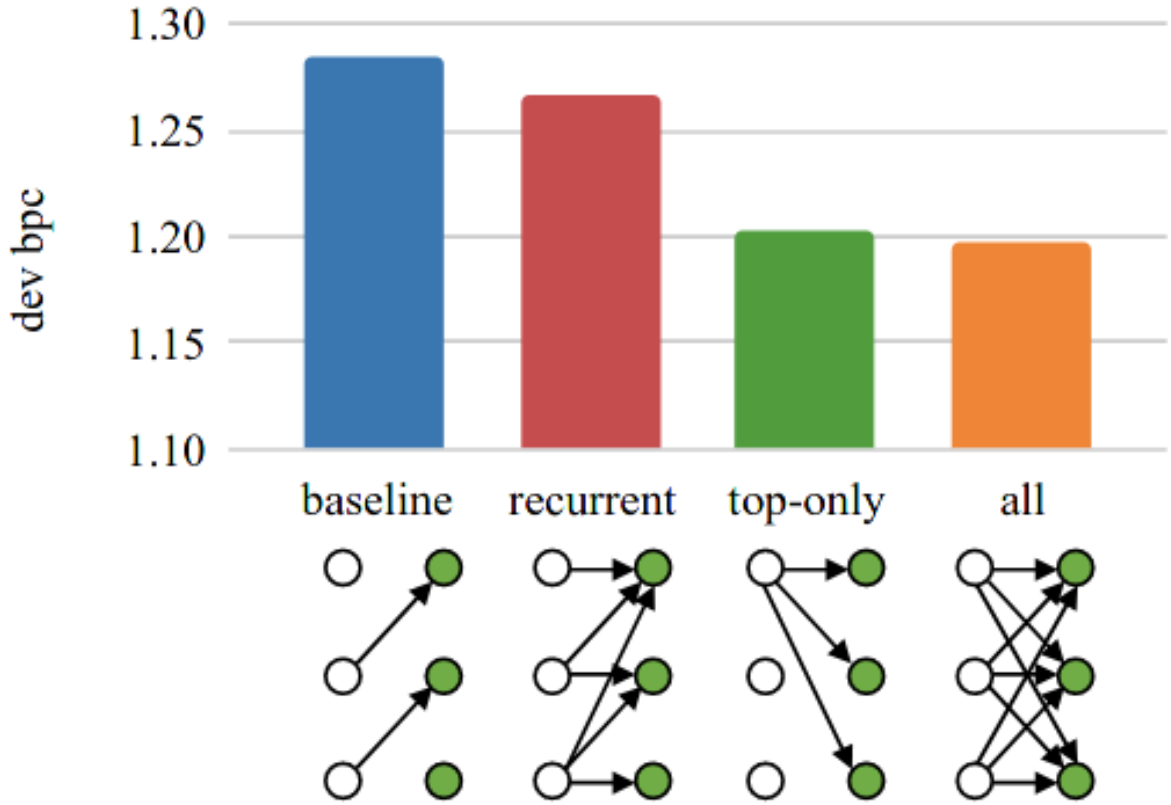
- model **shallower** only — layers are removed from a Transformer decoder
- the **1-layer Transformer** model can only reach **27.3**, the Feedback Transformer has **28.3** BLEU

Compare to other architectures - WMT En-De

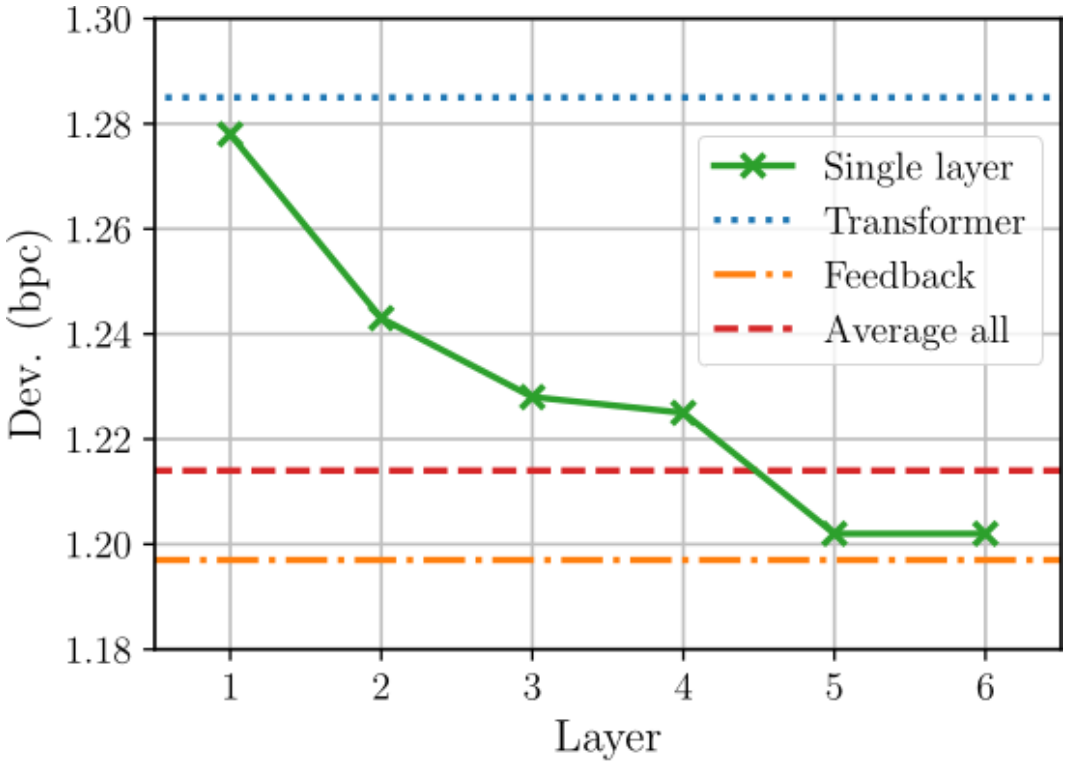
Model	Test
Recurrent Architectures	
DenseNMT Shen et al. (2018)	25.5
RNMT+ (Chen et al., 2018)	28.5
Hybrid Architectures	
BiARN (Hao et al., 2019)	28.9
SRU (Lei et al., 2017)	28.4
Transformer Architectures	
Transformer (Vaswani et al., 2017)	28.4
Transformer (Ott et al., 2018)	29.3
Feedback Transformer	29.5

Compare memorization strategy

- Feedback Transformer = all



Ablation studies



- **Feedback Transformers** - weighted sum of all layers
- **Single layer** - manually select one of the layers as the memory
- **Average all** - averaging all layers together

Score - WikiText-103

Model	Params	Test
Best Existing (Roy et al., 2020)	—	15.8
Trans-XL (Dai et al., 2019)	257M	18.3
Our Transformer	140M	19.9
Feedback Transformer	126M	18.3

- Best - **Routing Transformer** (sparse attention patterns, 2020)
- Feedback Transformer train **3.5 days**, compared to the Transformer which takes **1.2 days**
- train small Feedback Transformer - **half the size** of Transformer-XL (match the performance of Transformer-XL)
- **the same size** as standard Transformer has **worse performance** (19.9 PPL rather than 18.3)

Score - Enwiki8

Model	Params	Test
Best Existing (Rae et al., 2020)	277M	0.97
Trans-XL (Dai et al., 2019)	277M	0.99
Feedback Transformer	77M	0.96

- **character-level** language modeling - containing 100M unprocessed bytes from Wikipedia
- train a relatively small 12-layer model, that is **one third** of the size of the Transformer-XL baseline

Training and Inference Speed

Task	Model	Training Speed	Inference Speed
Language Modeling	Transformer	296K	592
	Feedback Transformer	84.4K	2176
Translation	Transformer	280K	3190
	Feedback Transformer	126K	5410
Reinforcement Learning	Transformer	22.3K	—
	Feedback Transformer	22.3K	—

GPT-Neo

- July 2020, EleutherAI - Tensorflow-mesh
- **replication of OpenAI** massive 175B parameter language model (GPT-3) and open source it to the public
- the largest model trained for a single step so far has been 200B parameters
- GPT-2 size models trained
- several experimental architectures implemented (e.g. Linear attention, Mixture of Experts, Masked Language Modeling)

References I

[1] O. Press, N. Smith A., and M. Lewis, “Shortformer: Better language modeling using shorter inputs,” 2020.

[2] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” 2017.

[3] D. Lepikhin, H. Lee, and Y. Xu, “Gshard: Scaling giant models with conditional computation and automatic sharding,” 2020.

[4] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” 2021.

[5] A. Fan, T. Lavril, E. Grave, A. Joulin, and S. Sukhbaatar, “Addressing some limitations of transformers with feedback memory,” 2020.