

**Zhifeng Chen**  
zhi fengc@google. com

Diagram illustrating the relationship between GShard, Transformer, and GShardis, with associated numerical values.

Component	Value
GShard	100
Transformer	4
GShardis	2048
TPU v	6000

[18, 3]



1 BLEU MoE 600B Delta  
 37.5B 600B ( x) TPU v core-year 6 22 (3.6x)  
 600B 2048 TPU v 4 22 TPU  
 v 100 29 TPU v core  
 Transformer 2.3B Delta BLEU of 6.1 GPipe  
 [15] 2048 TPU v 6 235.5 TPU v

1.1

GPU TPU

PyTorch [22]

TensorFlow [21]

[23, 15]

[6, 15]

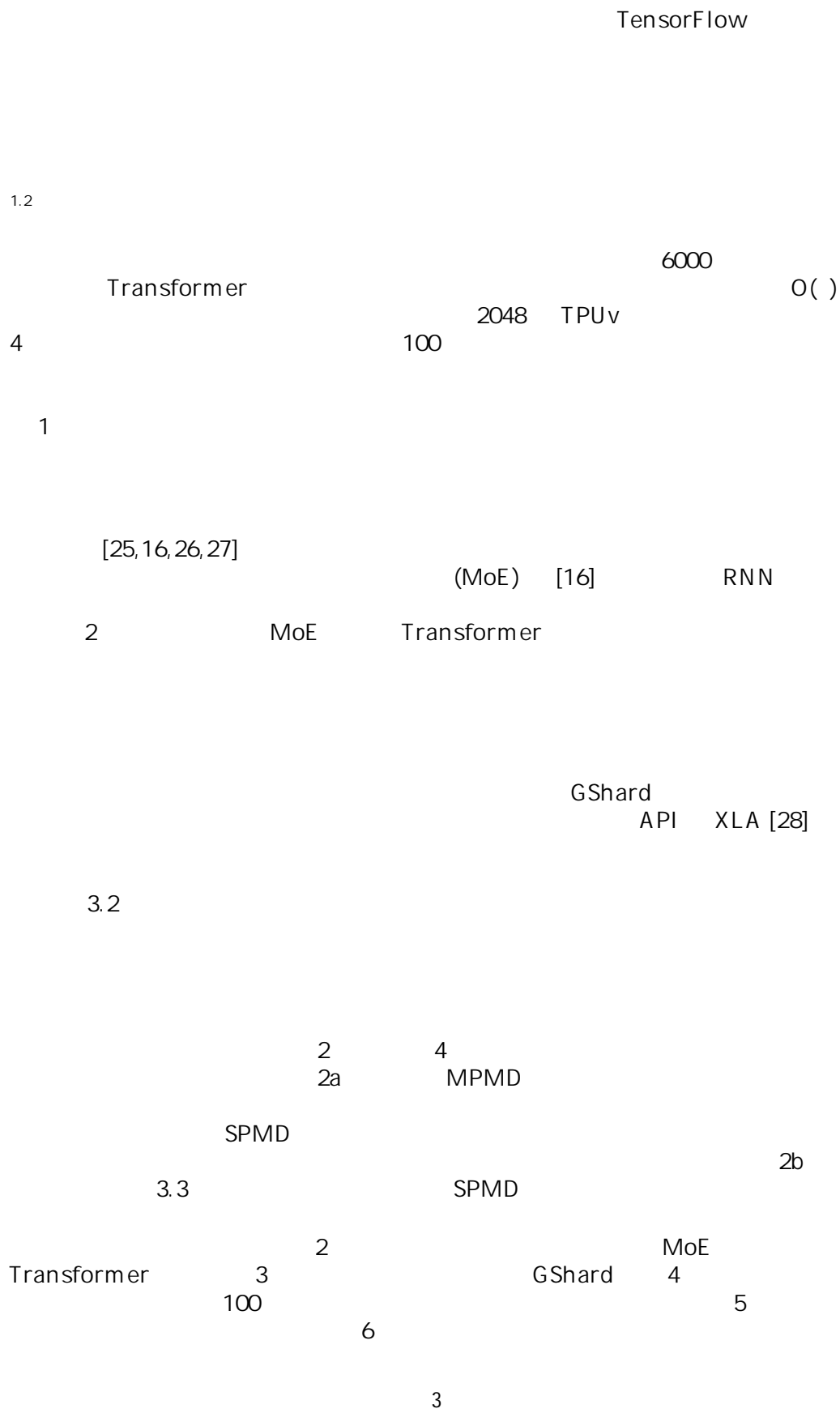
D

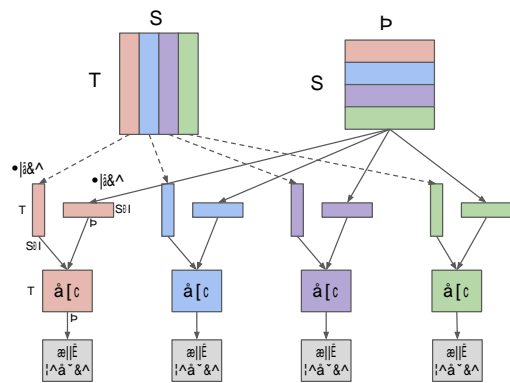
$O(D^2)$

$O(D)$

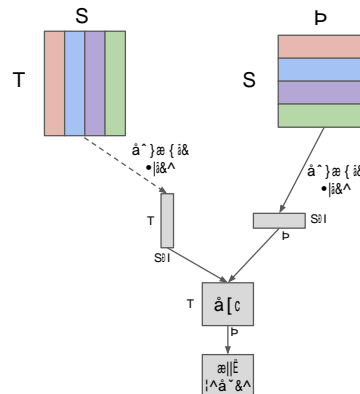
D

[15, 24]





(a) MPMD Partition



(b) SPMD Partition

2 MPMD  
SPMD

4

$([M, K] \times [K, N] = [M, N])$   
K

AllReduce  
SPMD

MPMD

SPMD

2

2.1 Transformer

Transformer [10]

Transformer  
Transformer

Transformer

top-

(MoE) [16]  
3 Transformer

Transformer  
MoE

MoE Transformer

MoE

3.1

5

## 2.2 Position-wise Mixture-of-Experts Layer

Transformer MoE [16]  
MoE Efeed-forward FFN 1 FFN :

$$G_{S,E} = \text{GATE}(x_S) \quad (1)$$

$$\text{FFN}_e(x_S) = w_{0e} \text{ReLU}(w_{1e} x_S) \quad (2)$$

$$y_S = \sum_{e=1}^E G_{S,e} \text{FFN}_e(x_S) \quad (3)$$

Figure 1 illustrates the MoE Transformer Encoder architecture. (a) shows a standard Transformer block with 3 MoE layers. (b) shows a MoE Transformer block with 1 MoE layer. (c) shows a MoE Transformer Encoder with 1 MoE layer.

The diagram illustrates the architecture of the proposed GATE model. It shows a sequence of operations: an input vector  $x$  is multiplied by a weight matrix  $w$  to produce a vector  $z$ . This vector  $z$  is then passed through a GATE function, which uses a MoE (Mixture of Experts) block. The output of the GATE function is then passed through a ReLU [29] activation function. Finally, the output is passed through a softmax function to produce the final result. The diagram also shows a MoE block with an FFN (Feed-Forward Network) and a GATE function.

- MoE softmax [16]

•

O(NE) E N N E

GATE(·)

1

•

$N$   
GATE(·)

$c_e$

$O(N/E)$

$G \cdot S \cdot E$

$x \cdot S$

GATE(·)  
 $S = N/G$

G

$2N/(G \cdot E)$

•

[16]

$k$

$L = n|| + k$   
aux  
aux

aux

(13)

1

$c_e / S$

$c_e / S$

$c_e$   
 $m_e$

top-

$(c_e / S)^2$

$m_e (c_e / S)$   
 $y \cdot S$

•

GATE(·)

$g^2$

3 GShard

2

TPU

TPU

TensorFlow [21]  
Transformer

1

GATE(·)

3.1

MoE

3.2

shardingAPI

1

(2) MoElayer

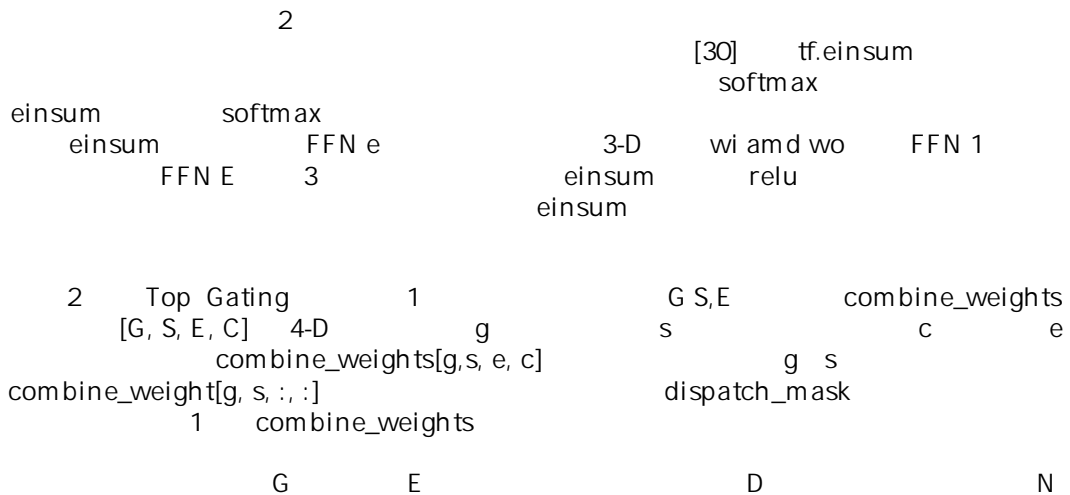
1 - 2

3.3

SPMD

1	top-	x S
<div> <div> <div>S</div> <div>C</div> </div> <div> <div>G S E</div> <div>1 c E O</div> <div>max(wg · x S)</div> <div>(3) m E 1</div> </div> </div>	<div> <div>aux</div> <div>(2) g S,E</div> <div>wg</div> <div>sof</div> </div>	<div> <div>gating decisions per expert</div> <div>en per expert, wg are trainable weights</div> <div>mean gates per expert</div> </div>
<div> <div>s= g s,E mean</div> <div>(4)</div> <div>for s 1 S do</div> <div>(5) g , e , g , e = top_ (gs,E) top-</div> <div>(6) g1 g1=(g1 + g2)</div> <div>(7) c c e e</div> <div>(8) if c<sub>e1</sub> &lt; C then</div> <div>(9)   G s,e g e x s</div> <div>(10) en</div> <div>c e c + 1 e</div> <div>count( ) end( ) aux = 1</div> <div>(14) for s 1 to S do</div> <div>(15) g , e , g , e = top_ (gs,E) top-</div> <div>(16) g2 g2=(g1 + g2)</div> <div>(17) rnd uniform(0;1)</div> <div>(18) c c e e</div> <div>(19) if c &lt; C 2 g2 &gt; rnd then</div> <div>(20)   G s,e g e x s</div> <div>(21) end</div> <div>(22) c<sub>e2</sub> c + 1</div> <div>(23) end</div> </div>	<div> <div>top-2 gates and expert indices</div> <div>normalized g1</div> <div>position in e1 expert buffer</div> <div>e1 expert combine weight for x<sub>s</sub></div> <div>incrementing e1 expert decisions count</div> <div>top-2 gates and expert indices</div> <div>normalized g2</div> <div>dispatch to second-best expert with probability 1/2 g2</div> <div>position in e2 expert buffer</div> <div>e2 expert combine weight for x<sub>s</sub></div> </div>	

3.1



2	Positions-wise MoE	G	E	—
---	--------------------	---	---	---

```

1  = softmax ( einsum ( "GSM-, ME ->GSE ",      , wg
)) combine_weights,dispatch_mask = Top_Gating(gates) d i s p a t
ched _ex per t _in put ts = einsum ( "GSEC ,GSM-
>EGCM",dispatch_mask,reshape_inputs) h=einsum("EGCM,EMH-
>EGCH",dispatched_expert_inputs,wi) h=relu(h) expert_outputs=ei
>GECM", h , wo )8      = einsum ( "GSEC ,GECM - >GSM " ,
,      ) — — —

```

$$\begin{aligned}
 & \text{ND} = O\left(\frac{D^2}{2}\right) \quad \text{b) } G = O(D), S = O\left(\frac{D}{2}\right) \quad \text{a) } N = O(GS) = \\
 & O(D) \quad \text{c) } M = O\left(\frac{D^2}{2}\right) \quad \text{H} = O\left(\frac{D}{2}\right) \quad \text{d) } E = O(D) \quad \text{e) } C = O\left(\frac{SE}{2}\right) = O\left(\frac{D}{2}\right), D \\
 & < S \quad \text{F L O P S}
 \end{aligned}$$

F L O P S Softmax +F L O P S Top\_Gating +F L O P S | +F L O P S  
 FFN =O(GSM E)+O(GSEC)+O(GSM EC)+O(EGCHM) =

$$\begin{aligned}
 & O(D \cdot 1 \cdot 1 \cdot D) + O(D \cdot 1 \cdot D \cdot \frac{1}{D}) + O(D \cdot 1 \cdot 1 \cdot D \cdot \frac{1}{D}) + O(D \cdot D \cdot \frac{1}{D} \cdot 1 \cdot 1) = \\
 & O(D^2) + O(D) + O(D) + O(D)
 \end{aligned}$$

$$\begin{aligned}
 & \text{F L O P S/D} = O(D) + O\left(\frac{D}{2}\right) + O\left(\frac{D}{2}\right) + O\left(\frac{D}{2}\right) \quad \text{softmax} \quad \text{F L O P S} \\
 & \text{softmax /D} = O(D) \quad D \ll H \quad D < S \\
 & \text{F L O P S/D} = O\left(\frac{D}{2}\right)
 \end{aligned}$$

$$D) \quad D \quad 5 \quad O($$

3.2 GShard API

1  
 2 GShard  
 API

TensorFlow/Lingvo [31] API

- replicate(tensor)

MoE

- split(tensor, split\_dimension, num\_partitions) split\_dimension  
i i num\_partitions

- shard(tensor, device\_assignment) split()

A.

API

1

2 D > S



split shard

GShard

API

MoE

MoE  
[32]

MoE  
A.

API

2

split

(G)

(E)

D

1# (G)

2+ = split ( , 0, D )3#

4+ wg = ( wg ) gates = softmax ( einsum ( " GSM , ME -> GSE " ,  
 , wg )) combine\_weights , dispatch\_mask = Top Gating (   
 gating\_logits ) d p a t c h e d \_ e x p e r t \_ i n p u t s = einsum ( 8 "  
 GSEC , GSM -> EGCM " , dispatch\_mask , reshape\_inputs )9#  
Expert(E) dim

+ \_ex p e r t \_ = split ( dispatched\_expert\_inputs, 0, D ) h =  
einsum ( " EGCM, EMH -> EGCH " , dispatched\_expert\_inputs, wi ) ...

Einsum  
3

G

G

einsum

G

einsum

7

G

5

split annotation E  
replicate(wg)

GShard

XLA

TensorFlow

Gather

Gather

2

Einsum

3

3

```

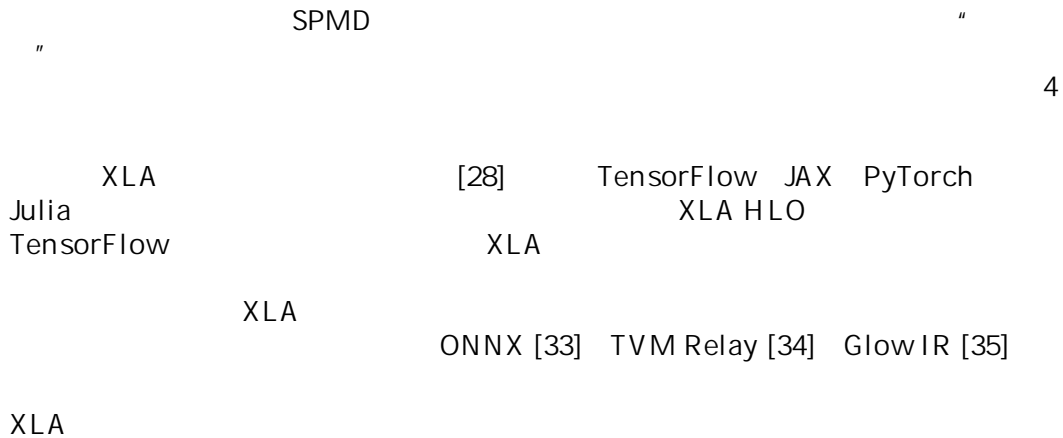
1 # [G, S, M] split() .2 input = split(
input, 0, num_devices)3 # s_indices [E, G, C, 1]
S .4 s_indices = split(s_indices, 1, num_devices)56 #
.7 #partitioned_input [G / num_devices, S, M]

8 partitioned_input = auto_to_manual_spmd_partition(input)
9 # partitioned_s_indices has shape [E, G/num_devices, C, 1]
partitioned_s_indices = auto_to_manual_spmd_partition(
s_indices)# partitioned_input G G iota .partiti
one_d_gs_indices = concat (iota ([E, G / num_devices, C, 1], 1)
,partitioned_s_indices, )#partitioned_data [E, G /
num_devices, C, M]partitioned_data = Gather (partitioned_input, par
titioned_gs_indices)# auto .# [E, G, C, M]

20 data = manual_to_auto_spmd_partition(partitioned_data)
21 ...

```

### 3.3 The XLA SPMD Partitioner for GShard



#### 3.3.1

MPI [36] SPMD XLA

CollectivePermute

-

AllGather

AllReduce

AllReduce

[37]

5.2

TPU

AllToAll

AllToAll

TPU

5.2

### 3.3.2 Per-Operator SPMD Partitioning

elementwise

MoE

3.3.3

Einsum

Einsum  
HLO

Einsum

MoE

LHS

RHS

XLA

•  
RHS

LHS

•

LHS RHS

LHS RHS

•

LHS RHS

LHS RHS

•  
Einsum

MoE

2

3

AllToAll

5.2  
4a

Einsum

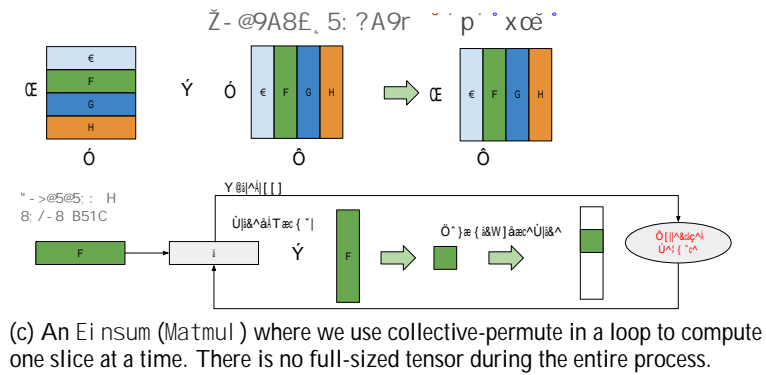
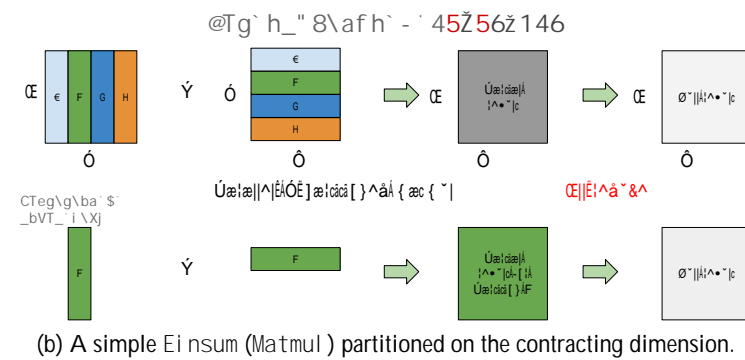
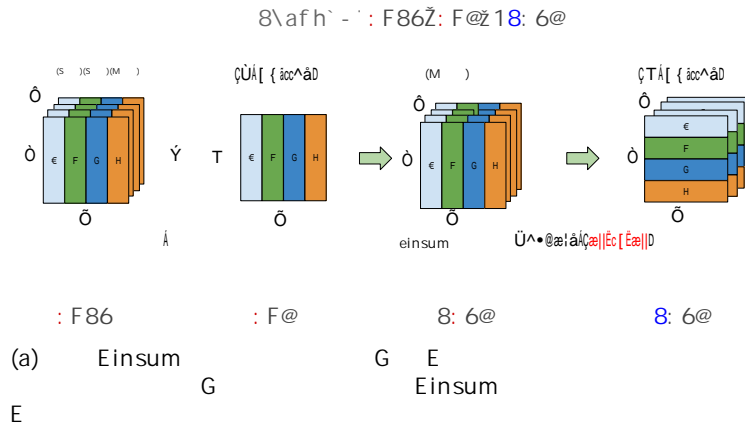
•

AllReduce

4b

•

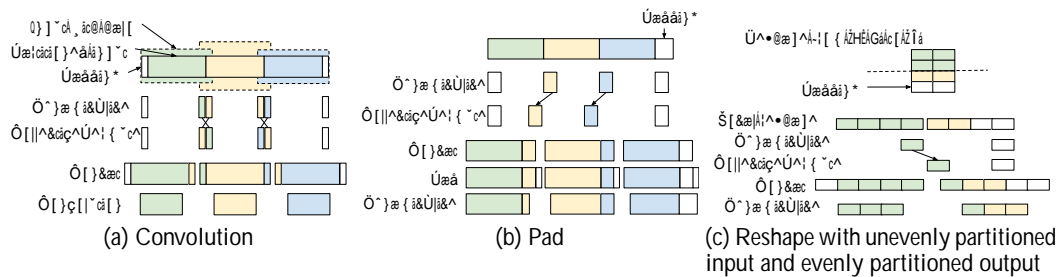
Cannon [38]



4 Einsum

Einsum

CollectivePermute 4c



5 Halo

SPMD  
ID

XLA

5

/

Reduce-Add  
(15) 2 (partitioncount)  
[0, 8) Iota  
(15)

1  
PartitionId × 8

XLA

A.

Halo  
halo

CollectivePermuteoperator

halo

Halo  
ReduceWindow

5a

A.

Slice

Pad

5b

halo

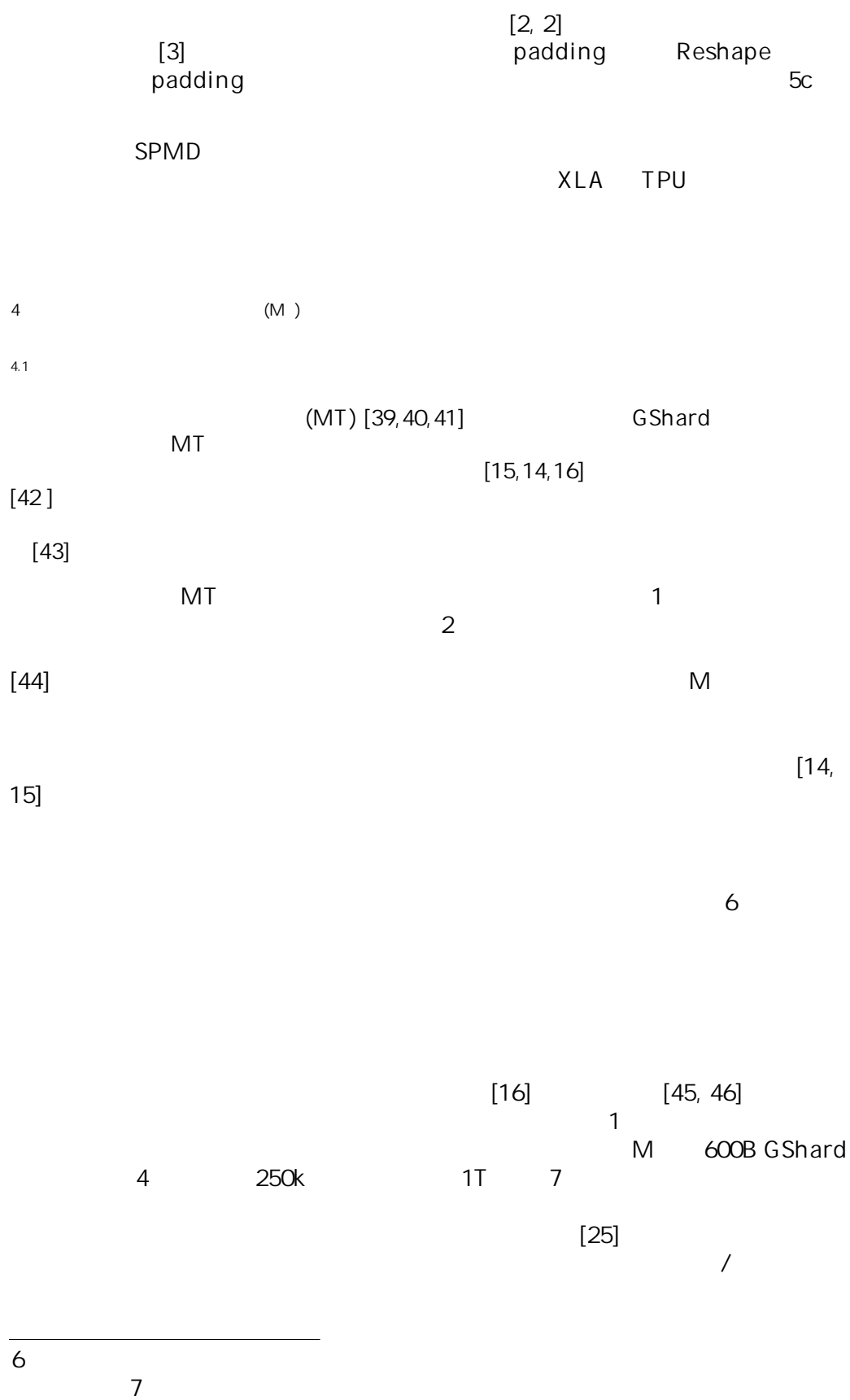
Reverse

Reshape

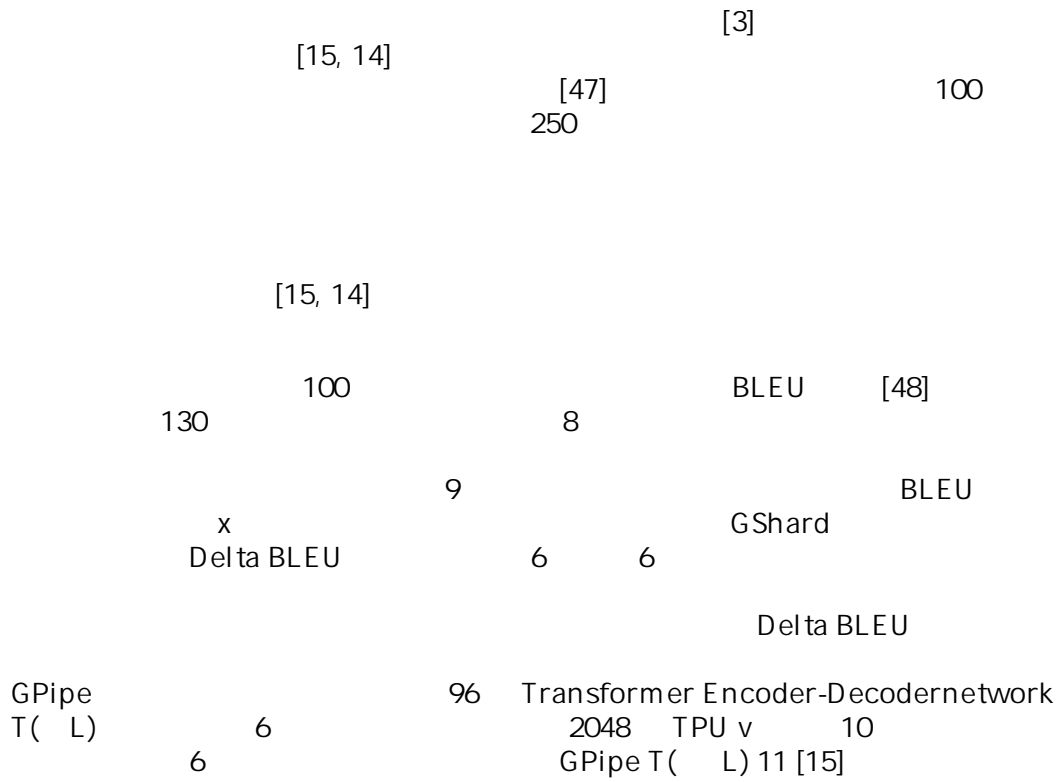
[3, 2]

[6]

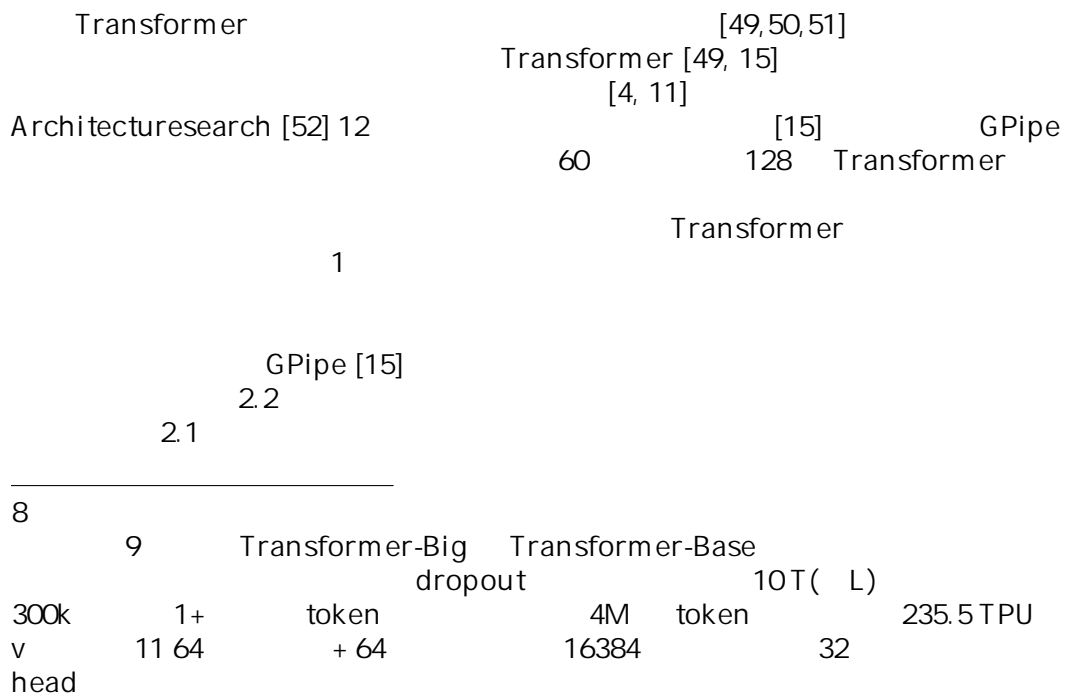
5

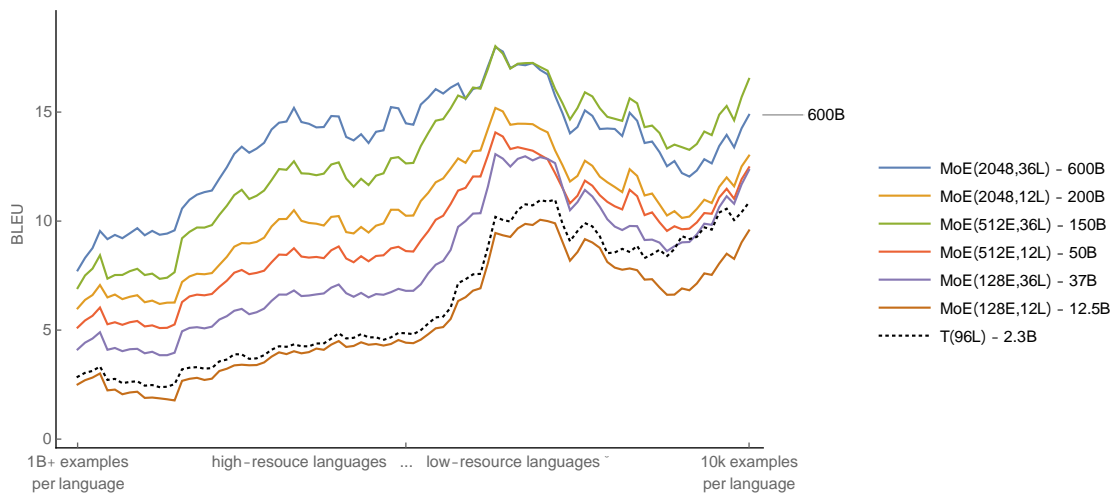


4.2

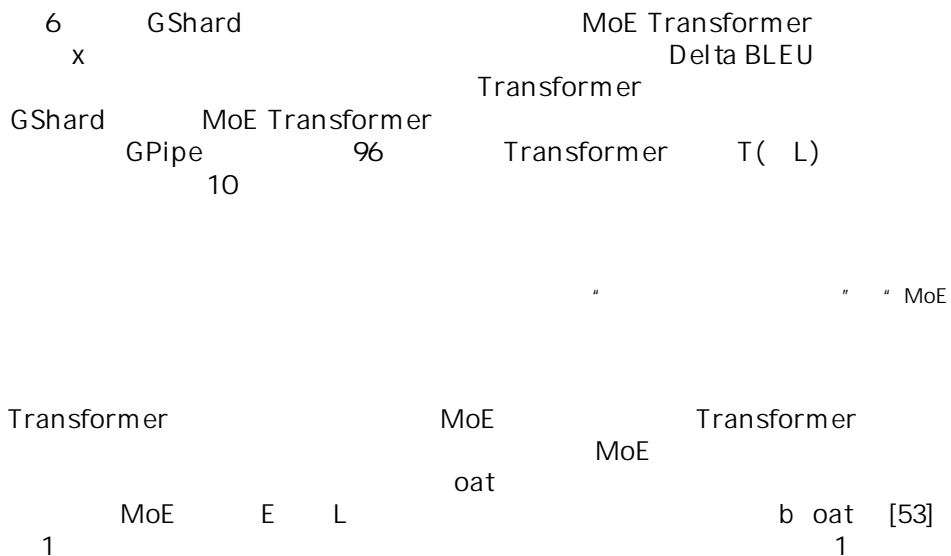


4.3 MoE





Id	Model	BLEU avg.	BLEU avg.	Weights
(1)	MoE(2048E, 36L)	<b>44.3</b>	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



Transformer MoE Transformer

MoE

oat

MoE E L

b oat [53]

1

A.

4.4

MoE Transformer



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

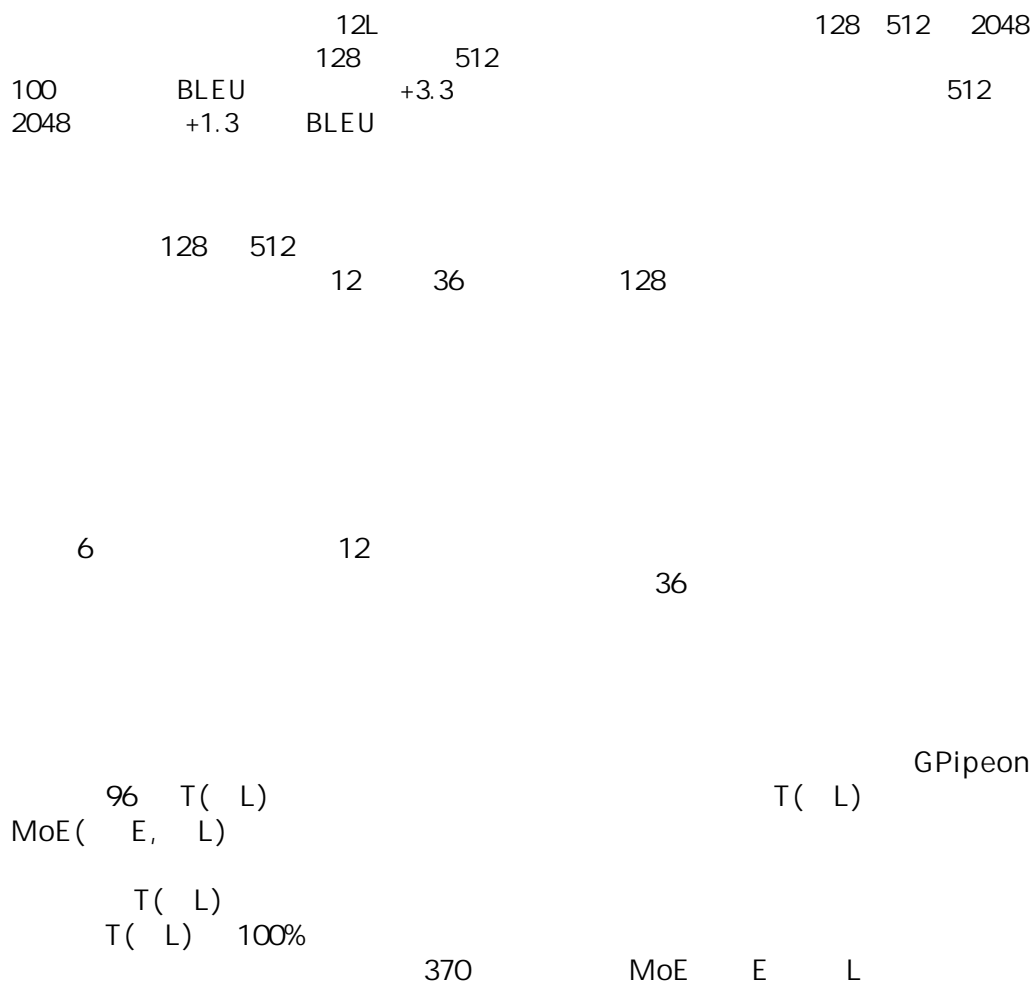
1 MoE i)  
ii) MoE

Transformer - (L) MoE  
(E) 12 Transformer 6  
6 36 60  
128 512 2048  
128 512 2048  
1  
1 MoE Transformer 1  
(10 12)  
BLEU  
6

[54]

[55]

2048 12 128 512 36  
6 (L) (E) 12L  
36L y Delta BLEU 3  
2 3  
4.1



#### 4.5 Training Efficiency

MoE Transformer

13

[1]

GShard / MoE Transformers [15, 56]

13

2.2

token  
MoE( E, L) 3 0.7 MoE Transformer MoE( E, L) 2 3  
512 2048 (4) vs (3) (6) (5) (2) vs (1)

MoE( E, L)	0.7	0.6	0.5	(1)	MoE( E, L)	( )
MoE( E, 12L)	176	484	178	0(3)	MoE( E, L)	( )
MoE( E, L)	512	214	148	6(5)	MoE( E, L)	321 1074 (6)
MoE( E, L)	995	--	2			

2  
5 3 1  
128 512 0.7  
4.4  
3 1  
(600B) 4  
TPU 14 TPU 3  
ID (1) 2048 TPU 4 MoE( E, L) 6000  
22.4 TPU BLEU

3 TPU T( L) (1) 235  
TPU GShard

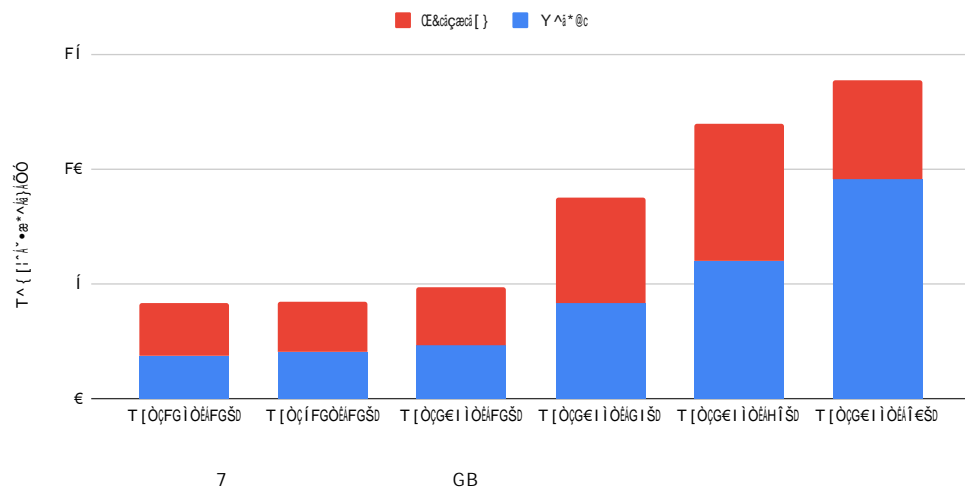
Id	Model	Cores	Steps per sec.	Batch sz. (Tokens)	TPU core years	Training time (days)	BLEU avg.
(1)	MoE(2048E, 36L)	2048	0.72	4M	22.4	<b>4.0</b>	<b>44.3</b>
(2)	MoE(2048E, 12L)	2048	2.15	4M	7.5	1.4	41.3
(3)	MoE(512E, 36L)	512	1.05	1M	15.5	11.0	43.7
(4)	MoE(512E, 12L)	512	3.28	1M	4.9	3.5	40.0
(5)	MoE(128E, 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

Table 3: Performance of MoE models with different number of experts and layers.

MoE Transformers

GShard

M



GShard

5

GShard

TPU

128

2048

16

1.7

5.1

GShard

SPMD

.

.

.

MoE

15

7

O( )

MoE( E, L)

12L

24L

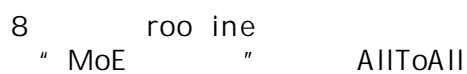
36L  
0%

60L

MoE( E, L)

28%

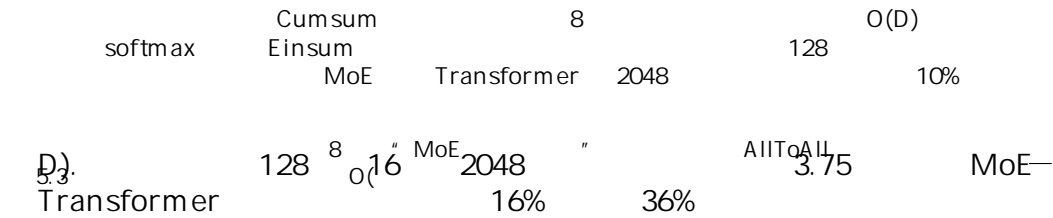
34%



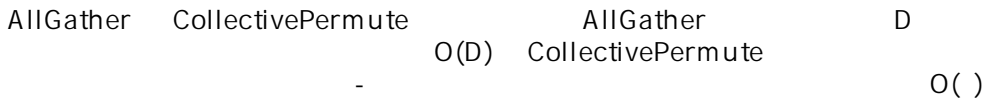
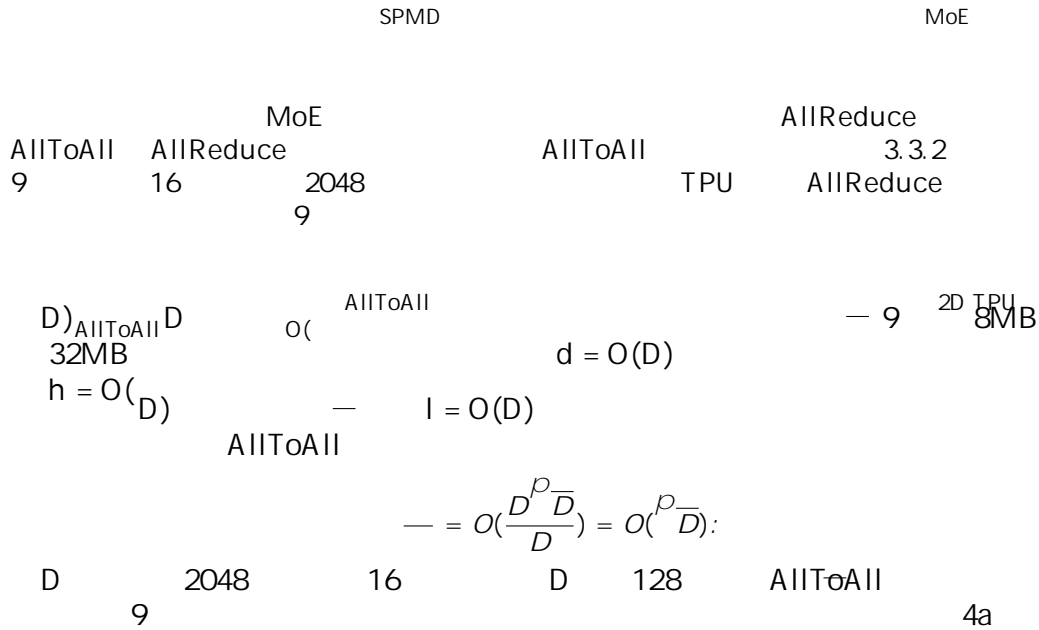
Model	FLOPS
MoE	8
Transformer	1

The diagram illustrates the data flow and operations for a Transformer-based MoE architecture. It includes components like Transformer, MoE, TPU, FLOPS, matmul, Einsum, Gather, softmax, O(D), O(GC) = O( ), ArgMax, Cumsum, and token. The diagram shows the flow from input tokens through various operations and hardware components.

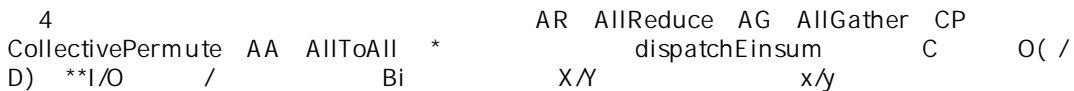
21



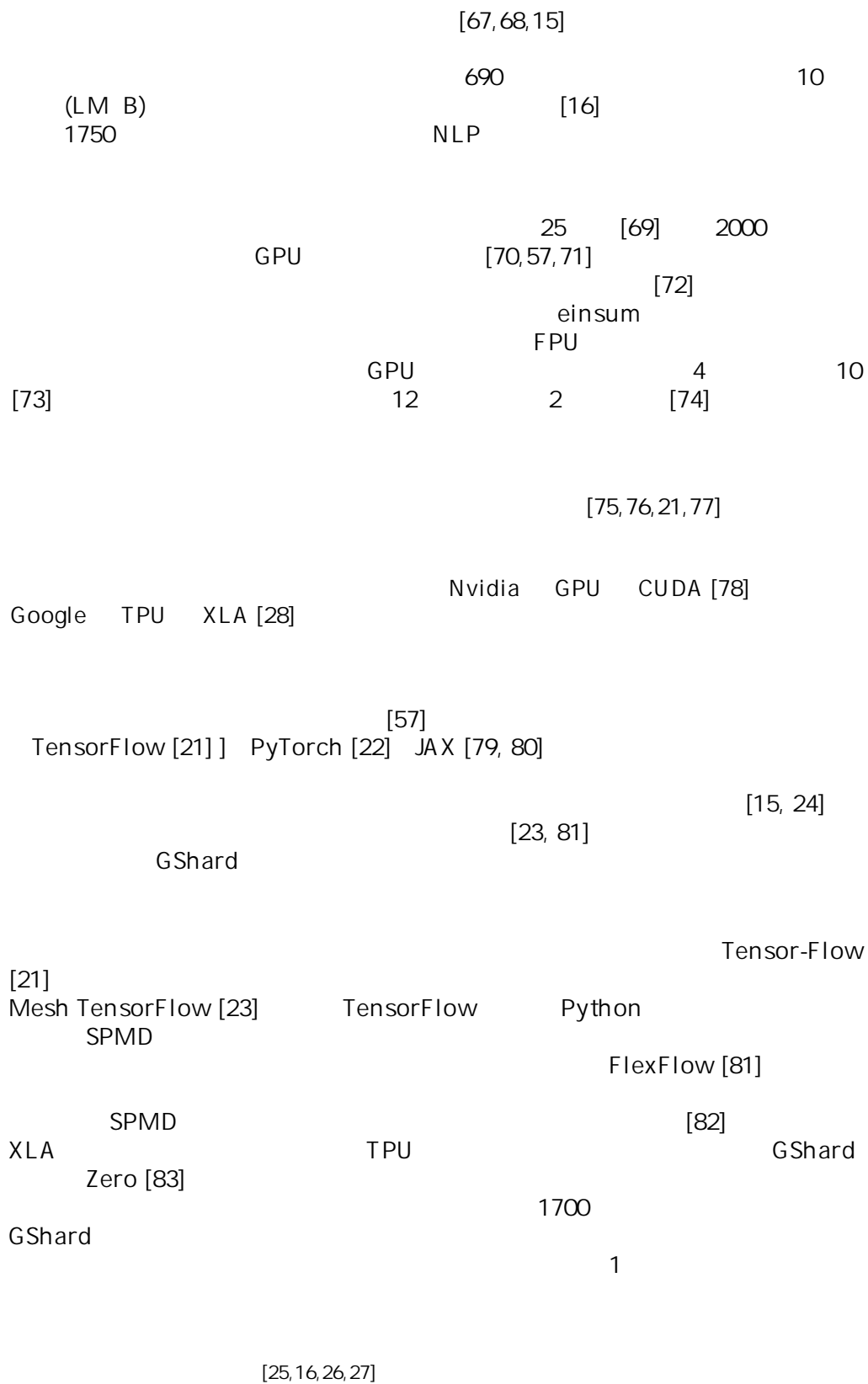
5.3



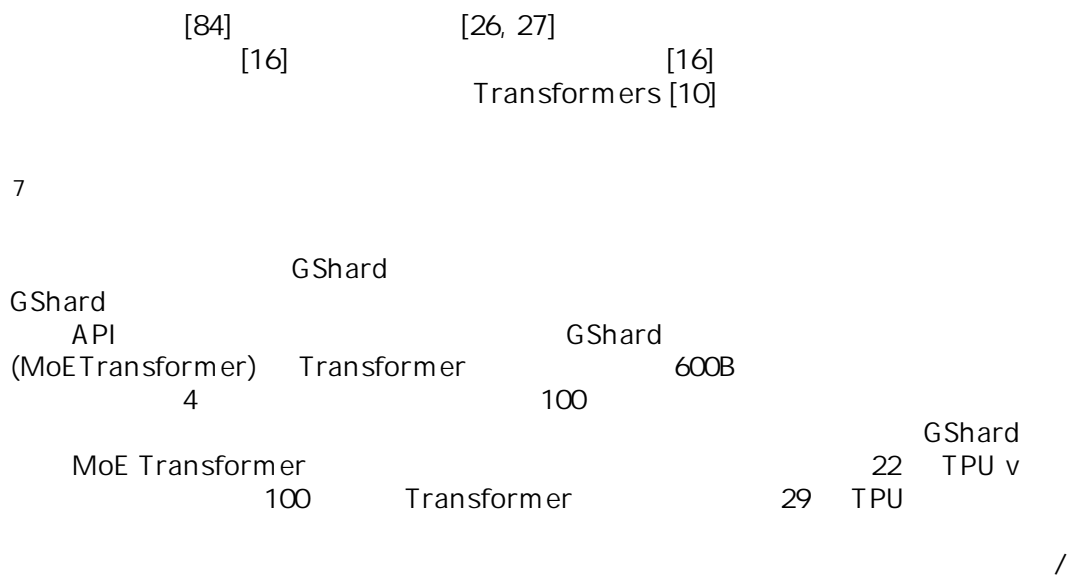
	$O(D)$ Dimensions	Total Compute	Per-partition	
			Compute	Communication
Add ( <u>A</u> , <u>A</u> -> <u>A</u> )	A	$O(D)$	$O(1)$	0
Matmul ( <u>AB</u> , <u>BC</u> -> <u>AC</u> )	B	$O(D)$	$O(1)$	$O(1)$ AR
Matmul ( <u>AB</u> , <u>BC</u> -> <u>AC</u> )	A	$O(D)$	$O(1)$	0
Matmul ( <u>AB</u> , <u>BC</u> -> <u>AC</u> )	A, B	$O(D^2)$	$O(D)$	$O(D)$ AG or CP
Matmul ( <u>AB</u> , <u>BC</u> -> <u>AC</u> )	A, C	$O(D^2)$	$O(D)$	$O(D)$ AG or CP
Reduce ( <u>AB</u> -> <u>A</u> )	A	$O(D)$	$O(1)$	0
Reduce ( <u>AB</u> -> <u>B</u> )	A	$O(D)$	$O(1)$	$O(1)$ AR
Einsum ( <u>G</u> SEC, <u>G</u> SM-> <u>E</u> GCM)	G, E *	$O(D)$	$O(1)$	$O(1)$ AA
Convol uti on(BI <u>X</u> Y, xyl O->B <u>O</u> X <u>Y</u> )	X **	$O(D)$	$O(1)$	$O(1)$ CP











GShard

SPMD

[85, 86]

## Acknowledgements

Google Brain Translate XLA  
 Lingvo Youlong Cheng Naveen Arivazhagan Ankur  
 Bapna YonghuiWu Yuan Cao David Majnemer James Molloy Peter Hawkins  
 Blake Hechtman Mark He ernan

Dimitris Vardoulakis, Tamas Berghammer, Marco Cornero, Cong Liu, Tong Shen, Hongjun Choi, Jianwei Xie, Sneha Kudugunta, and Macduff Hughes.

## References

- [1] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. *arXiv preprint arXiv:1802.06509*, 2018.
- [2] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.
- [3] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [5] Dhruv Mahajan Ross Girshick Vignesh Ramanathan Kaiming He Manohar Paluri Yixuan Li Ashwin Bharambe Laurens van der Maaten  
(ECCV) 181-196 2018
- [6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.
- [9] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019.
- [10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.
- [11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019.
- [12] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [13] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale, 2019.
- [14] Naveen Arivazhagan Ankur Bapna Orhan Firat Dmitry Lepikhin Melvin Johnson Maxim Krikun Mia Xu Chen Yuan Cao George Foster Colin Cherry Wolfgang Macherey Zhifeng Chen Yonghui Wu 2019
- [15] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in Neural Information Processing Systems 32*, pages 103–112, 2019.

- [16] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.
- [17] Madhu S. Advani and Andrew M. Saxe. High-dimensional dynamics of generalization error in neural networks, 2017.
- [18] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically, 2017.
- [19] Joel Hestness, Newsha Ardalani, and Gregory Diamos. Beyond human-level accuracy. *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, Feb 2019.
- [20] . . . . .  
2020 2 023401 2020 2
- [21] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.
- [22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
- [23] Noam Shazeer Youlong Cheng Niki Parmar Dustin Tran Ashish Vaswani Penporn Koanan-takool Peter Hawkins HyounJoong Lee Mingsheng Hong Cli Young Mesh-tensor ow 10414-10423 2018
- [24] Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil Devanur, Greg Ganger, and Phil Gibbons. Pipedream: Fast and efficient pipeline parallel dnn training. *arXiv preprint arXiv:1806.03377*, 2018.
- [25] Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models, 2015.
- [26] Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *ArXiv*, abs/1910.10073, 2020.
- [27] Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. Controlling computation versus quality for neural sequence models, 2020.
- [28] XLA: Optimizing Compiler for TensorFlow. <https://www.tensorflow.org/xla>, 2019. Online; accessed 1 June 2020.
- [29] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- [30] Albert Einstein. Die grundlage der allgemeinen relativitätstheorie. In *Das Relativitätsprinzip*, pages 81–124. Springer, 1923.
- [31] Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia Xu Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*, 2019.
- [32] . Cloud TPU  
3D ML <https://cloud.google.com/blog/products/ai-machine-learning/train-ml-models-on-large-images-and-d-tpus>  
2019 ; 2020 6 12

[33] ONNX: Open Neural Network Exchange. <https://github.com/onnx/onnx>, 2019. Online; accessed 1 June 2020.

[34] Jared Roesch Steven Lyubomirsky Logan Weber Josh Pollock Marisa Kirisame Tianqi Chen Zachary Tatlock Relay ir  
ACM SIGPLAN -MAPL 2018 2018

[35]

Jongsoo Park Artem  
Glow 2018

[36] MPI Forum. MPI: A Message-Passing Interface Standard. Version 2.2, September 4th 2009. available at: <http://www.mpi-forum.org> (Dec. 2009).

[37] Minsik Cho, Ulrich Finkler, and David Kung. BlueConnect: Decomposing All-Reduce for Deep Learning on Heterogeneous Network Hierarchy. In *Proceedings of the Conference on Systems and Machine Learning (SysML)*, Palo Alto, CA, 2019.

[38] Lynn Elliot Cannon. *A Cellular Computer to Implement the Kalman Filter Algorithm*. PhD thesis, USA, 1969. AAI7010025.

[39] Orhan Firat Kyunghyun Cho Yoshua Bengio  
2016  
2016

[40] Melvin Johnson Mike Schuster Quoc V. Le Maxim Krikun Zhifeng Chen Nikhil Thorat Fernanda Viégas Martin Wattenberg Greg Corrado  
5:339–351 2017 12

[41] Roei Aharoni, Melvin Johnson, and Orhan Firat. Massively multilingual neural machine translation. *CoRR*, abs/1903.00089, 2019.

[42] <https://ai.googleblog.com/2020/06/recent-advances-in-google-translate.html>  
exploring-massively-multilingual.html 2020-06-05

[43] Recent advances in google translate. <https://ai.googleblog.com/2020/06/recent-advances-in-google-translate.html>. Accessed: 2020-06-05.

[44] Timothy T Baldwin and J Kevin Ford. Transfer of training: A review and directions for future research. *Personnel psychology*, 41(1):63–105, 1988.

[45] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

[46] Andrew Davis and Itamar Arel. Low-rank approximations for conditional feedforward computation in deep neural networks, 2013.

[47] Jakob Uszkoreit Jay M. Ponte Ashok C. Popat Moshe Dubiner  
23  
COLING '10 1101-1109 2010

[48] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

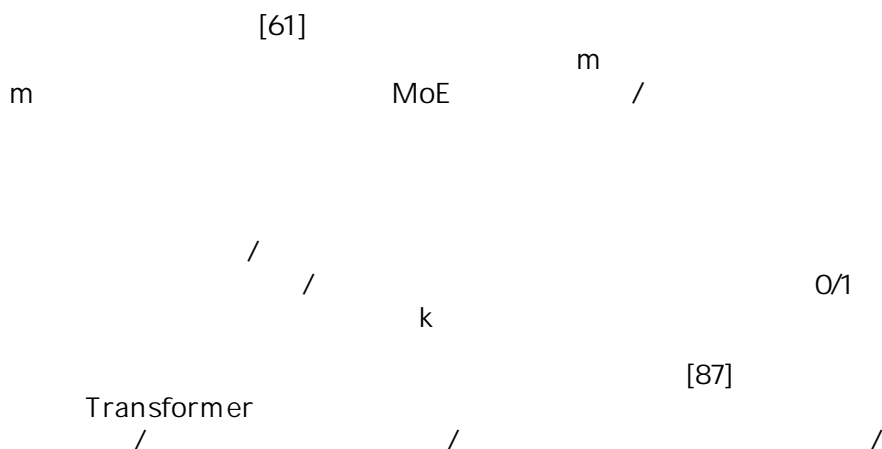
[49] Ankur Bapna, Mia Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. Training deeper neural machine translation models with transparent attention. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

- [50] Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Language modeling with deep transformers. *Interspeech 2019*, Sep 2019.
- [51] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. Learning deep transformer models for machine translation. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.
- [52] David R. So, Chen Liang, and Quoc V. Le. The evolved transformer, 2019.
- [53] Using bfloat16 with TensorFlow models. <https://cloud.google.com/tpu/docs/bfloat16>, 2020. Online; accessed 12 June 2020.
- [54] Heng-Tze Cheng Mustafa Ispir Rohan Anil Zakaria Haque Lichan Hong  
Vihaan Jain Xiaobing Liu Hemal Shah Levent Koc Jeremiah Harmsen  
- DLRS 2016 2016
- [55] Andrew K. Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks, 2018.
- [56] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.
- [57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [58] Christian Szegedy Wei Liu Yangqing Jia Pierre Sermanet Scott Reed  
Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Andrew Rabinovich  
IEEE 1-9 2015
- [59] Ilya Sutskever Oriol Vinyals Quoc V Le  
3104-3112 2014
- [60] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [61] Mike Schuster Quoc V Le Mohammad Norouzi  
Wolfgang Macherey Maxim Krikun Klaus Macherey  
arXiv arXiv 1609.08144 2016
- [62] Geoffrey Hinton George E Dahl Abdel-rahman Mohamed Navdeep Jaitly  
Andrew Senior Vincent Vanhoucke Patrick Nguyen Tara N Sainath  
IEEE 29(6):82–97, 2012
- [63] William Chan Navdeep Jaitly Quoc Le Oriol Vinyals  
2016 IEEE  
(ICASSP) 4960-4964 IEEE 2016
- [64] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjali Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE, 2018.
- [65] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

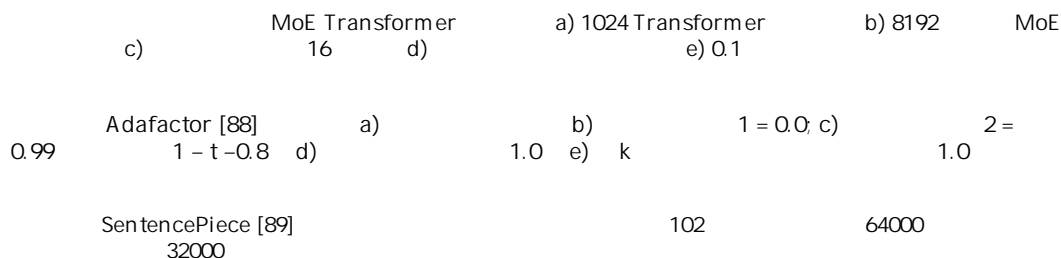
- [66] Jonathan Shen, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Rj Skerrv-Ryan, tts  
2018 IEEE (ICASSP) 4779-4783 IEEE 2018
- [67] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. 2017.
- [68] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning, 2017.
- [69] Paolo Ienne, Thierry Cornu, and Gary Kuhn. Special-purpose digital hardware for neural networks: An architectural survey. *Journal of VLSI signal processing systems for signal, image and video technology*, 13(1):5–25, 1996.
- [70] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880, 2009.
- [71] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220, 2010.
- [72] Norman P Jouppi, Cli Young, Nishant Patil, David Patterson, Gaurav Agrawal, RaminderBajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers  
44 1-12 2017
- [73] 2019 recent trends in GPU price per FLOPS. <https://aiimpacts.org/2019-recent-trends-in-gpu-price-per-flops/>. Accessed: 2020-06-05.
- [74] Yifan Sun, Nicolas Bohm Agostini, Shi Dong, and David Kaeli. Summarizing cpu and gpu design trends with product data. *arXiv preprint arXiv:1911.11313*, 2019.
- [75] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc’aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.
- [76] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.
- [77] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
- [78] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, 2008.
- [79] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs. 2018.
- [80] Roy Frostig, Matthew Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. In *Machine Learning and Systems (MLSys)*, 2018.
- [81] Zhihao Jia, Matei Zaharia, and Alex Aiken. Beyond Data and Model Parallelism for Deep Neural Networks. In *Proceedings of the Conference on Systems and Machine Learning (SysML)*, Palo Alto, CA, 2019.
- [82] Yuanzhong Xu, HyoukJoong Lee, Dehao Chen, Hongjun Choi, Blake Hechtman, and Shibo Wang. Automatic cross-replica sharding of weight update in data-parallel training, 2020.

- [83] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimization towards training a trillion parameter models. *arXiv preprint arXiv:1910.02054*, 2019.
- [84] Loren Lugosch, Derek Nowrouzezahrai, and Brett H. Meyer. Surprisal-triggered conditional computation with neural networks, 2020.
- [85] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes, 2018.
- [86] Wesley J. Maddox, Gregory Benton, and Andrew Gordon Wilson. Rethinking parameter counting in deep models: Effective dimensionality revisited, 2020.
- [87] . arXiv preprint arXiv: . , 2019
- [88] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *ArXiv*, abs/1804.04235, 2018.
- [89] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP*, 2018.

A.



A.



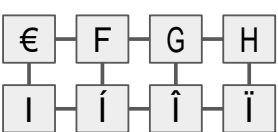
A.3 General Sharding API

3.2 API replicate() split()

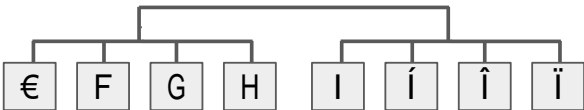
shard(tensor, device\_assignment) deviceassignment

device\_assignment

4] 3D ID [3, 16, 64] [1, 2, [3, 8, 16]



T^•@ÁV[] [] [\*^

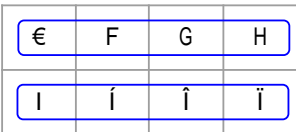


V!^ÁÁV[] [] [\*^



Ö^ç&^ÁE••ä\*} { ^}c

10



Ö^ç&^ÁE••ä\*} { ^}c

2D 2x ID

ID

10

A. SPMD

GShard [32]

API

# W [N ,C ,H , W ]

input = split(inputs, 3, D)#

= ( ) conv = conv d ( , )...

GShard

ReduceWindow

CollectivePermute halo

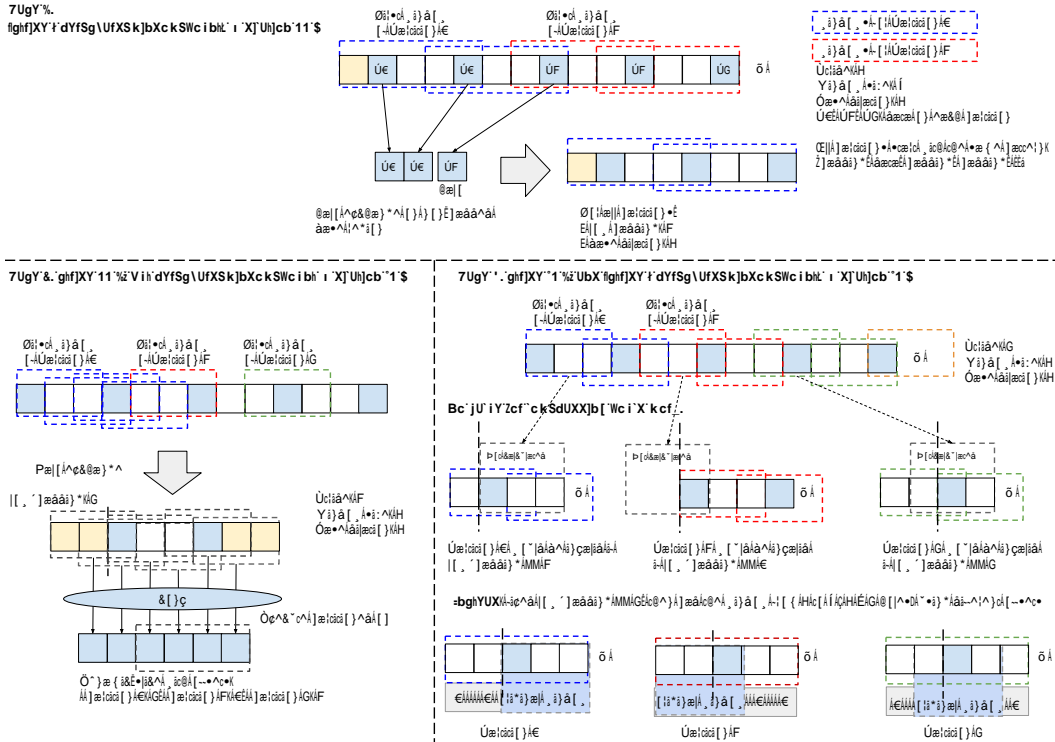
halo CollectivePermute




$$\frac{\cdot}{1} \quad \text{LHS} \quad \frac{\cdot}{\cdot} \quad \text{RHS} \quad \frac{\cdot}{\cdot} \quad \text{LHS}$$

### 3.3.3

33



13

• stride × per\_shard\_window\_count  
per\_shard\_window\_count

LHS

Halo non-dilated/non-padding i  
stride × per\_shard\_window\_count × i +  
window\_size – low\_pad + dilation – dilation

stride × per\_shard\_window\_count  
a × i + b a b • stride  
== 1 per\_shard\_window\_count

SPMD

Pad DynamicSlice

stride == 1

DynamicSlice

i

$\frac{\text{per\_shard\_window\_count} \times i + \text{window\_size} - \text{low\_pad} + \text{dilation}}{\text{dilation}} - 1$

a × i + b • stride = 1 stride × per\_shard\_window\_count

6

$$13. \qquad \qquad \qquad 6 \qquad \qquad \qquad = 1$$

Pad                      DynamicSlice

$$\begin{array}{ccc} & \text{RHS} & \text{LHS} \\ \text{RHS} & & \\ & \text{RHS} & / \end{array}$$