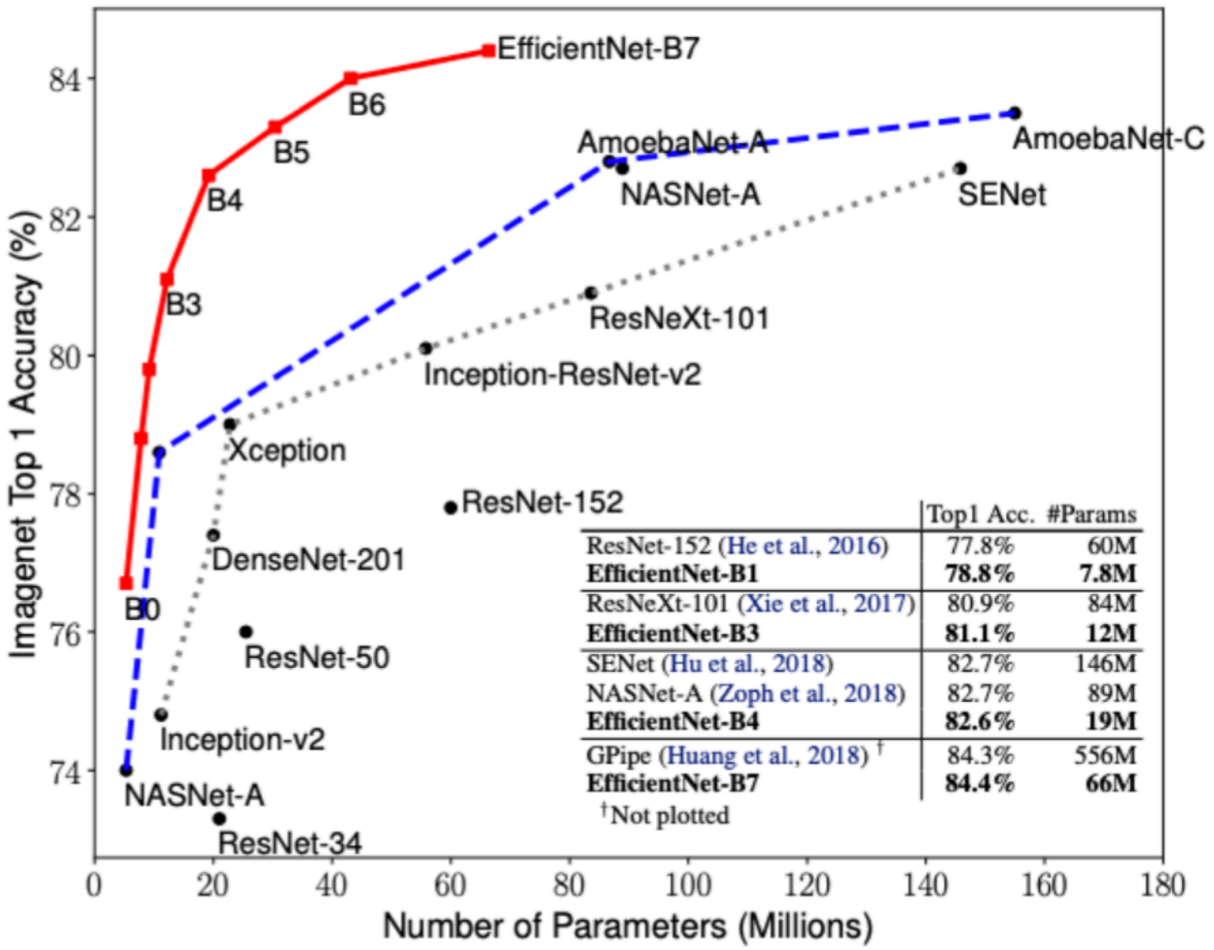
二次最適化を用いた巨大な言語モデルの学習
およびFRNNを用いたプラズマ挙動予測

東京工業大学
学術国際情報センター
横田理央

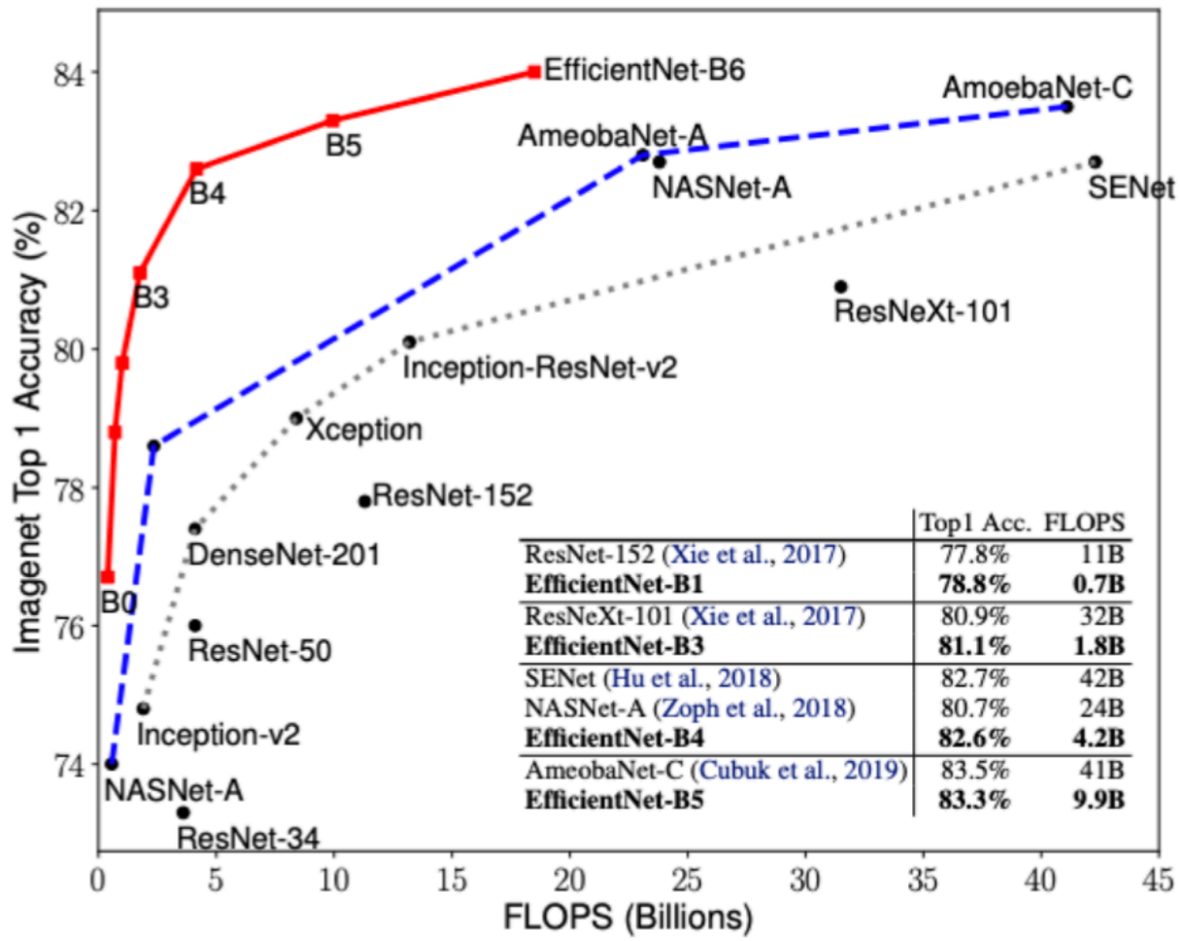
ABCI グランドチャレンジ 2019 成果発表会
テレコムセンタービル東棟14階　2020年2月21日

スパコンでしかできない深層学習



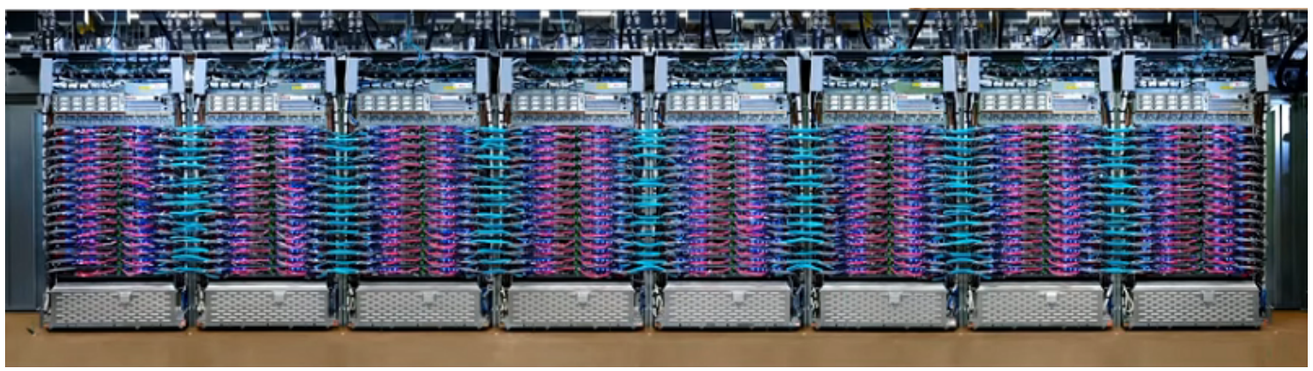
	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**78.8%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.1%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.6%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.3%**	**9.9B**

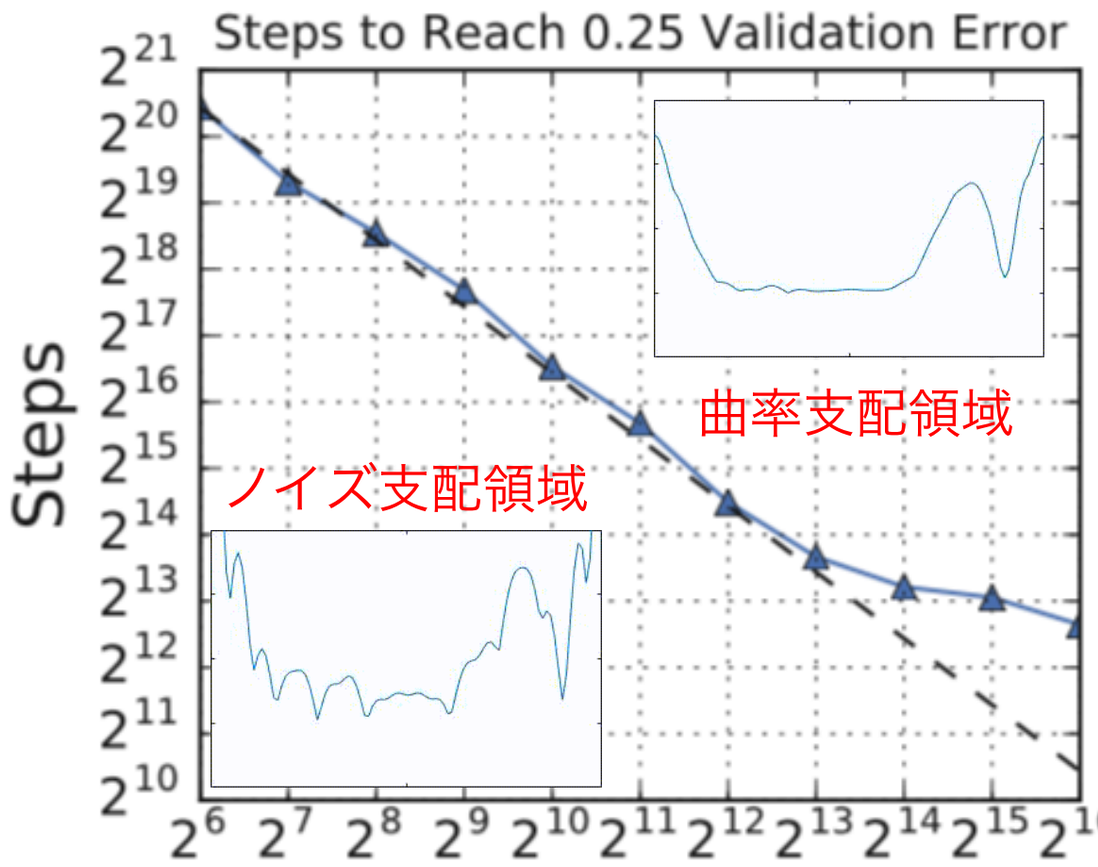
Figures: M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, ICML 2019



Google TPU v3　12.5PF/rack

AIST ABCI 17 PF/rack

数千GPU規模のImageNetの学習

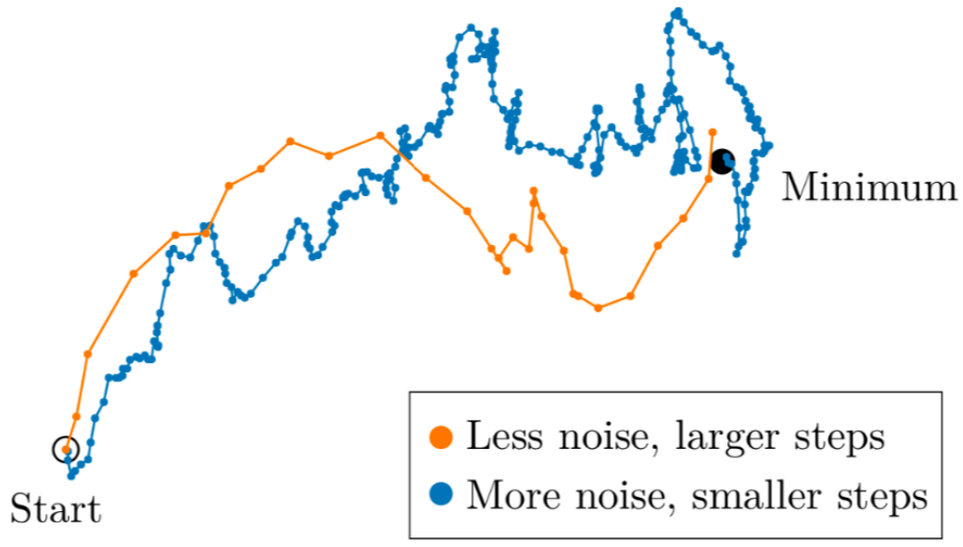


無駄な更新 ⟷ 無駄なデータ

GPU数に比例してバッチサイズが増える

	#GPU/TPU	time	epochs
Facebook	512	30 min	90
PFN	1024	15 min	90
UC Berkeley	2048	14 min	64
Tencent	2048	6.6 min	90
Google	1024	2.2 min	90
This work	2048	1.9 min	45
Fujitsu	3456	1.2 min	90
Sony	2048	1.1 min	90

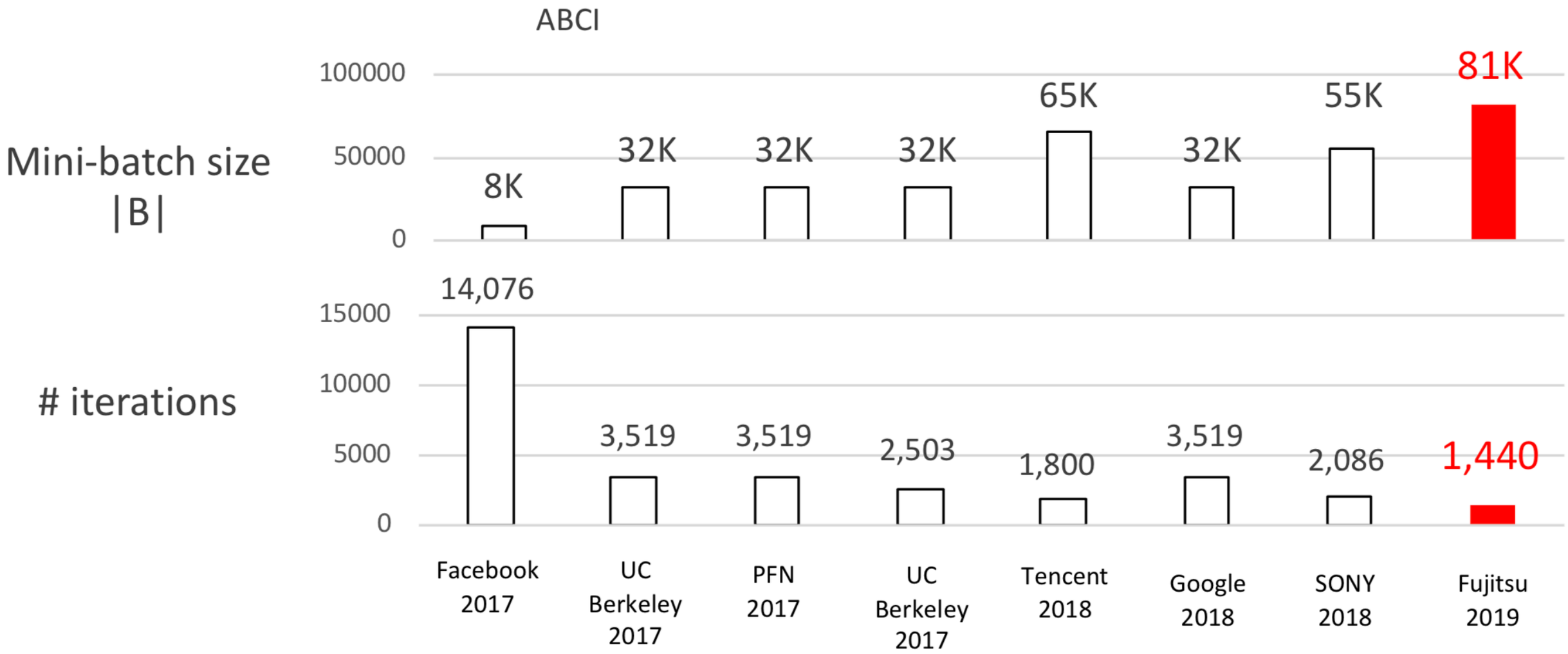
局所最適解から抜けれなくなる？
ミニバッチSGDの持つノイズが重要？



何個の画像を見てから一歩進むか

数千GPU規模のImageNetの学習

		Hardware	Software	Batch size	Optimizer	# Steps	Time/step	Time	Accuracy
Facebook	Goyal *et al.* [6]	Tesla P100 × 256	Caffe2	8,192	SGD	14,076	0.255 s	1 hr	**76.3** %
UC Berkeley	You *et al.* [8]	KNL × 2048	Intel Caffe	32,768	SGD	3,519	0.341 s	20 min	75.4 %
PFN	Akiba *et al.* [7]	Tesla P100 × 1024	Chainer	32,768	RMSprop/SGD	3,519	0.255 s	15 min	74.9 %
UC Berkeley	You *et al.* [8]	KNL × 2048	Intel Caffe	32,768	SGD	2,503	0.335 s	14 min	74.9 %
Tencent	Jia *et al.* [9]	Tesla P40 × 2048	TensorFlow	65,536	SGD	1,800	0.220 s	6.6 min	75.8 %
Google	Ying *et al.* [13]	TPU v3 × 1024	TensorFlow	32,768	SGD	3,519	**0.037 s**	2.2 min	**76.3** %
SONY	Mikami *et al.* [10]	Tesla V100 × 3456	NNL	55,296	SGD	2,086	0.057 s	2.0 min	75.3 %
Fujitsu	Yamazaki *et al.* [11]	Tesla V100 × 2048	MXNet	81,920	SGD	1,440	0.050 s	**1.2 min**	75.1 %



二次最適化

一次最適化（確率的勾配降下法：SGD）

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla \mathcal{L}(\theta^{(t)})$$

二次最適化

- ニュートン法

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \left\{ H(\theta^{(t)}) \right\}^{-1} \nabla \mathcal{L}(\theta^{(t)})$$

ヘッセ行列

- 自然勾配法（S. Amari, 1998）

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \left\{ F(\theta^{(t)}) \right\}^{-1} \nabla \mathcal{L}(\theta^{(t)})$$

FIM

深層学習においては・・・

$$H(\theta^{(t)}), F(\theta^{(t)}) \in \mathbb{R}^{N \times N}$$

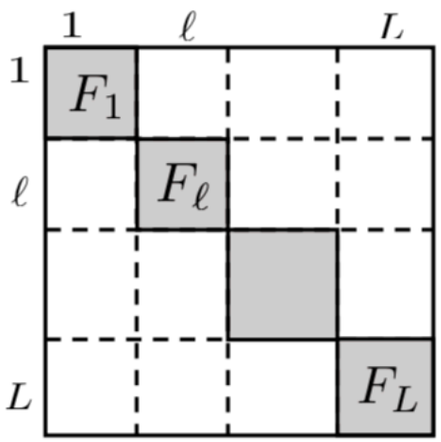
x メモリに載らない
x 逆行列（O(N^3)）計算は非現実的

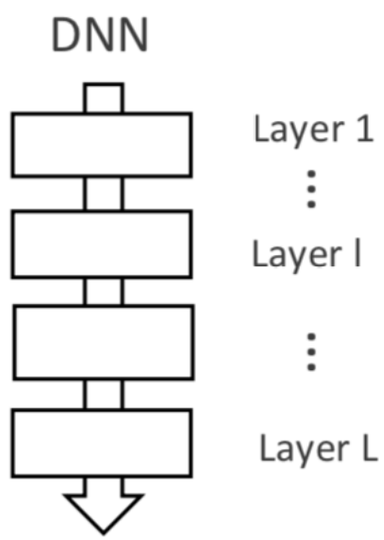
クロネッカー因子分解(K-FAC)

Kronecker-**F**actored **A**pproximate **C**urvature (J. Martens+, ICML 2015)

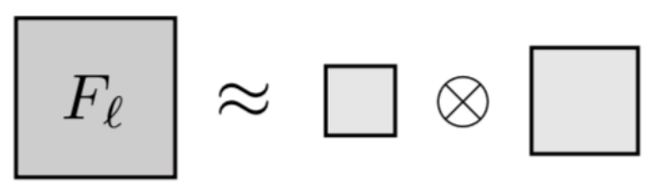
Step 1. Layer-wise block-diagonal approximation

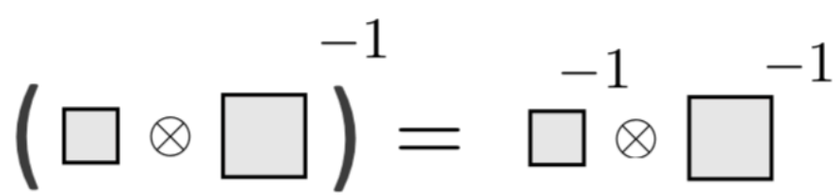
$C = F \approx$ FIM





Step 2. Kronecker-factorization (for each layer)



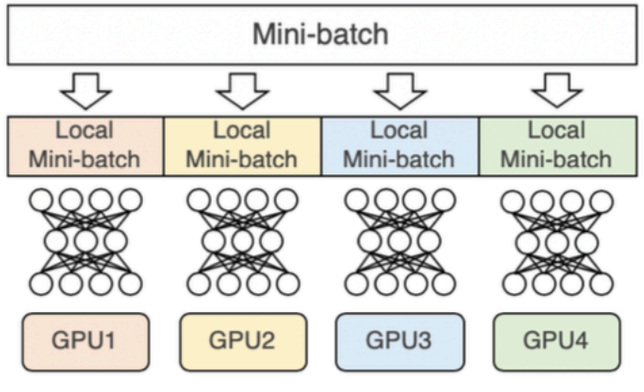


easier to invert $O(N^3) \to O(N_\ell^3)$

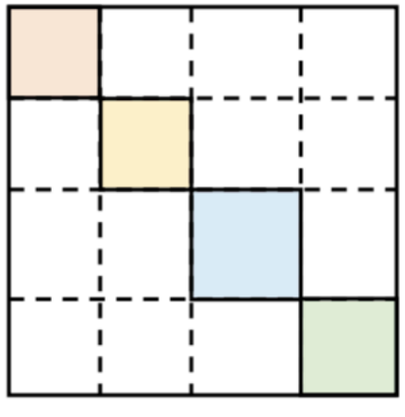
much easier to invert $O(N_\ell^3) \to O(N_\ell^{3/2})$

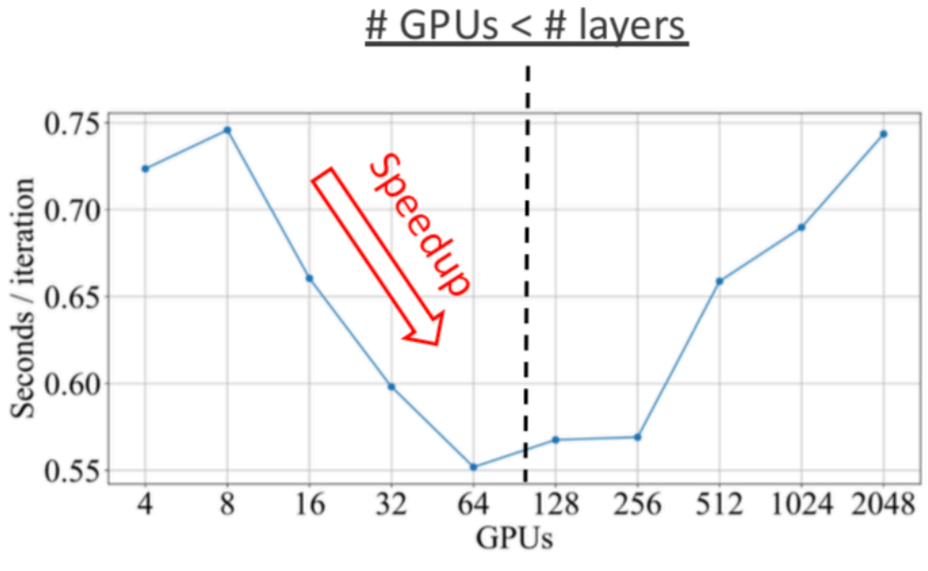
Distributed K-FAC (K. Osawa+, CVPR 2019)

data-parallelism × layers-parallelism



$F \approx$

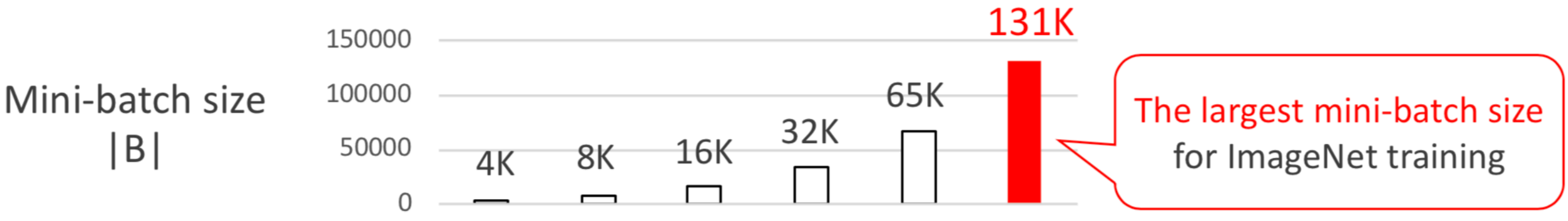


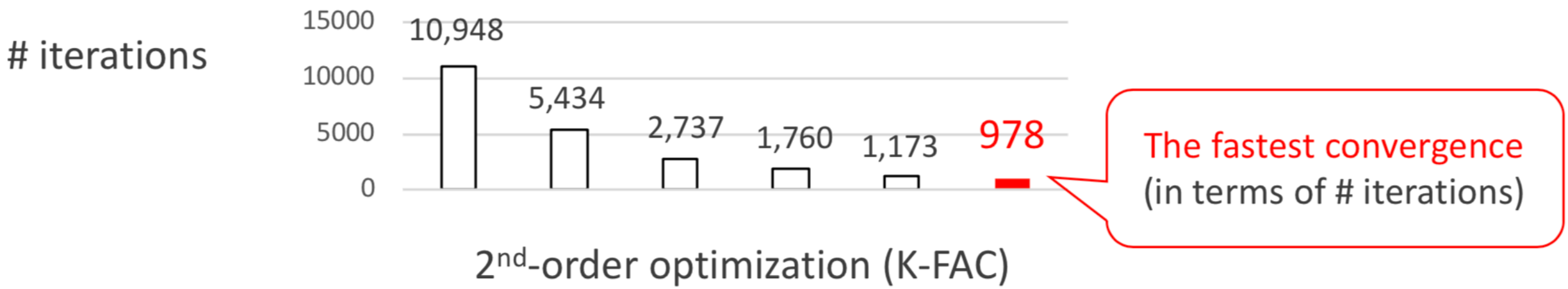


Training ResNet-50 (107 layers)
on ImageNet classification

分散並列K-FACを用いたImageNetの学習

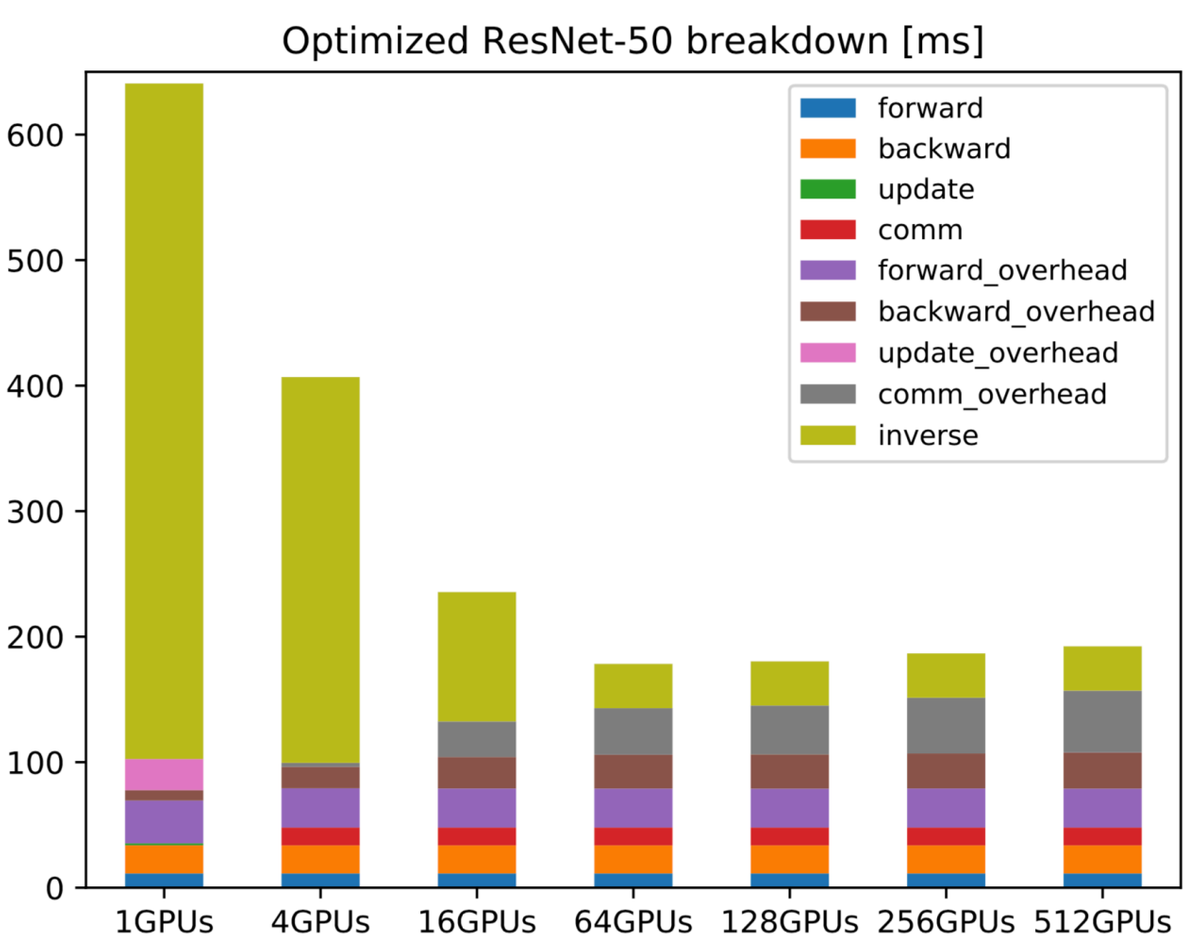
Our Distributed K-FAC results on ABCI supercomputer
with extremely large-batches for training ResNet-50 on ImageNet (CVPR19)

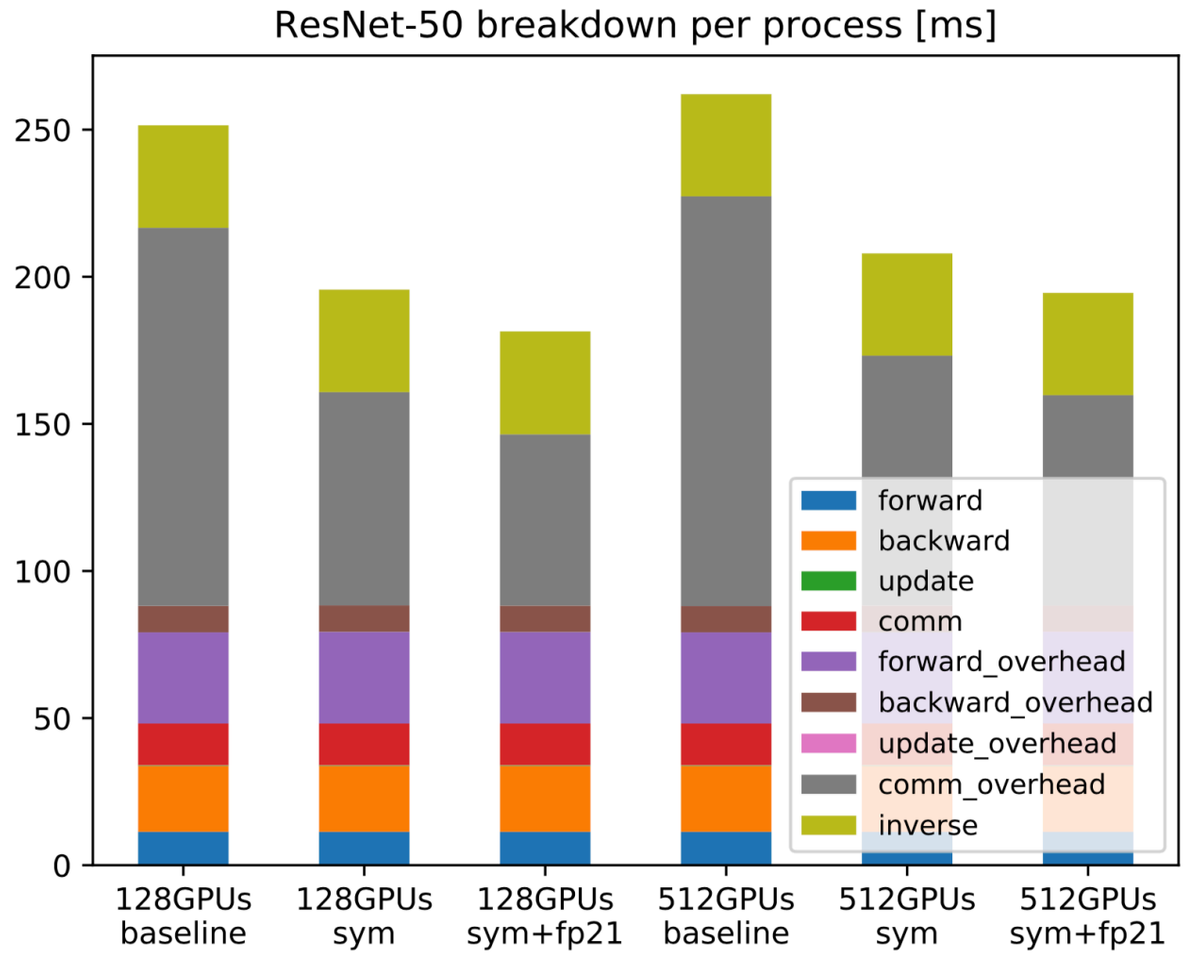




The first empirical results showing
the advantage of 2nd-order optimization (K-FAC) in Large-scale DL
(faster convergence)

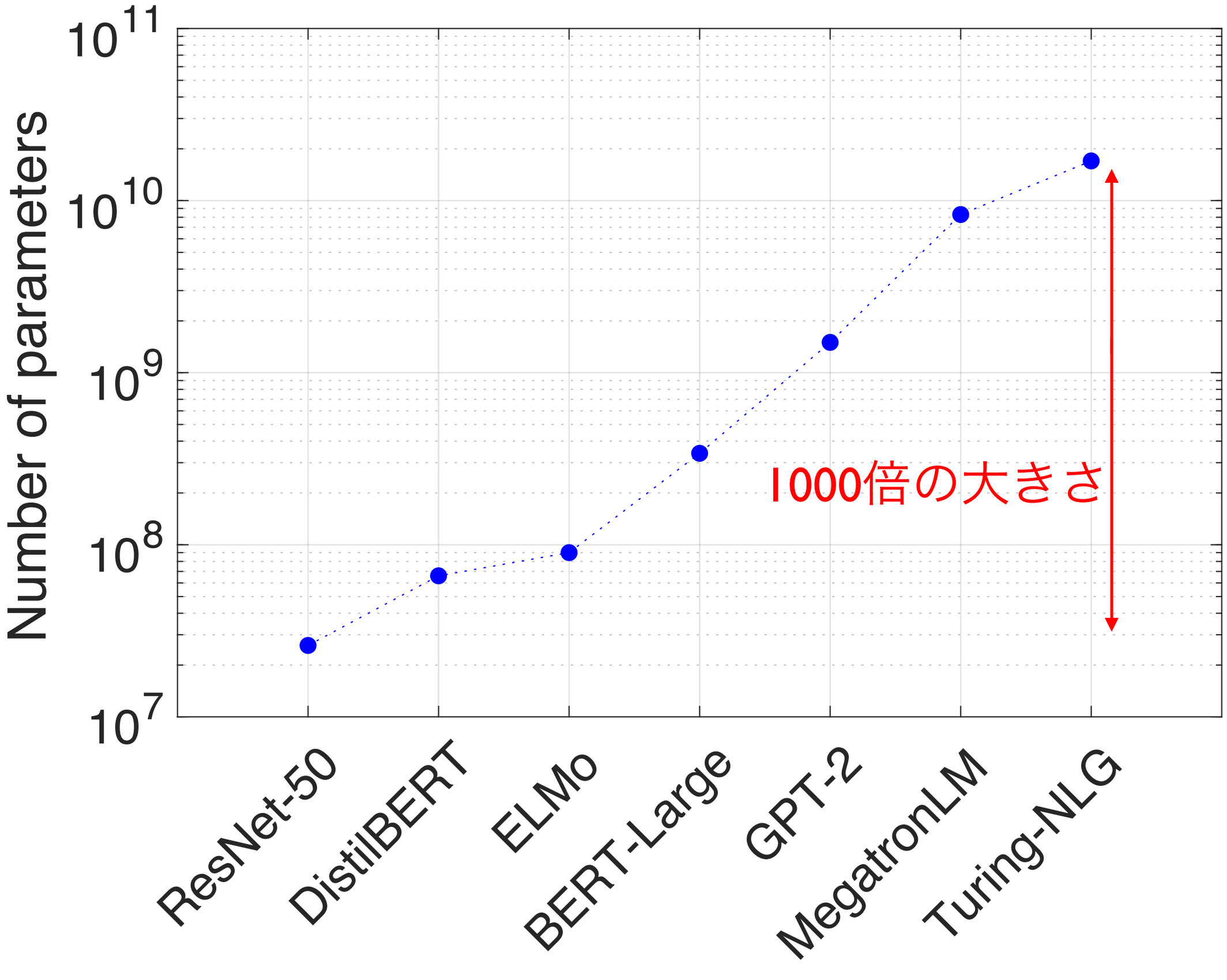
パフォーマンスの最適化





	Distrib. FIM inverse Sec. 3.3.1	Symm. comm. of FIM Sec. 3.3.2	float21 comm. of FIM Sec. 3.3.3	Overlap comm. Sec. 4.1	Hier. Comm. Sec. 4.1	Stale FIM Sec. 4.2	SGD Time/upd.	K-FAC Time/upd.	K-FAC/SGD ratio
Osawa *et al.* [19]	✓	✓				✓[1]	-	340 ms	-
Tsuji *et al.* [23]	✓	✓		✓	✓		85 ms	315 ms	3.71
Osawa *et al.* [18]	✓	✓		✓	✓		-	236 ms	-
	✓	✓		✓	✓	✓	-	178 ms	-
This work	✓						**57** ms	251 ms	4.40
	✓	✓						199 ms	3.49
	✓	✓	✓					182 ms	3.19
	✓	✓	✓	✓				221 ms	3.88
	✓	✓	✓	✓	✓			192 ms	3.37
	✓	✓	✓	✓	✓	✓		**108** ms	**1.89**

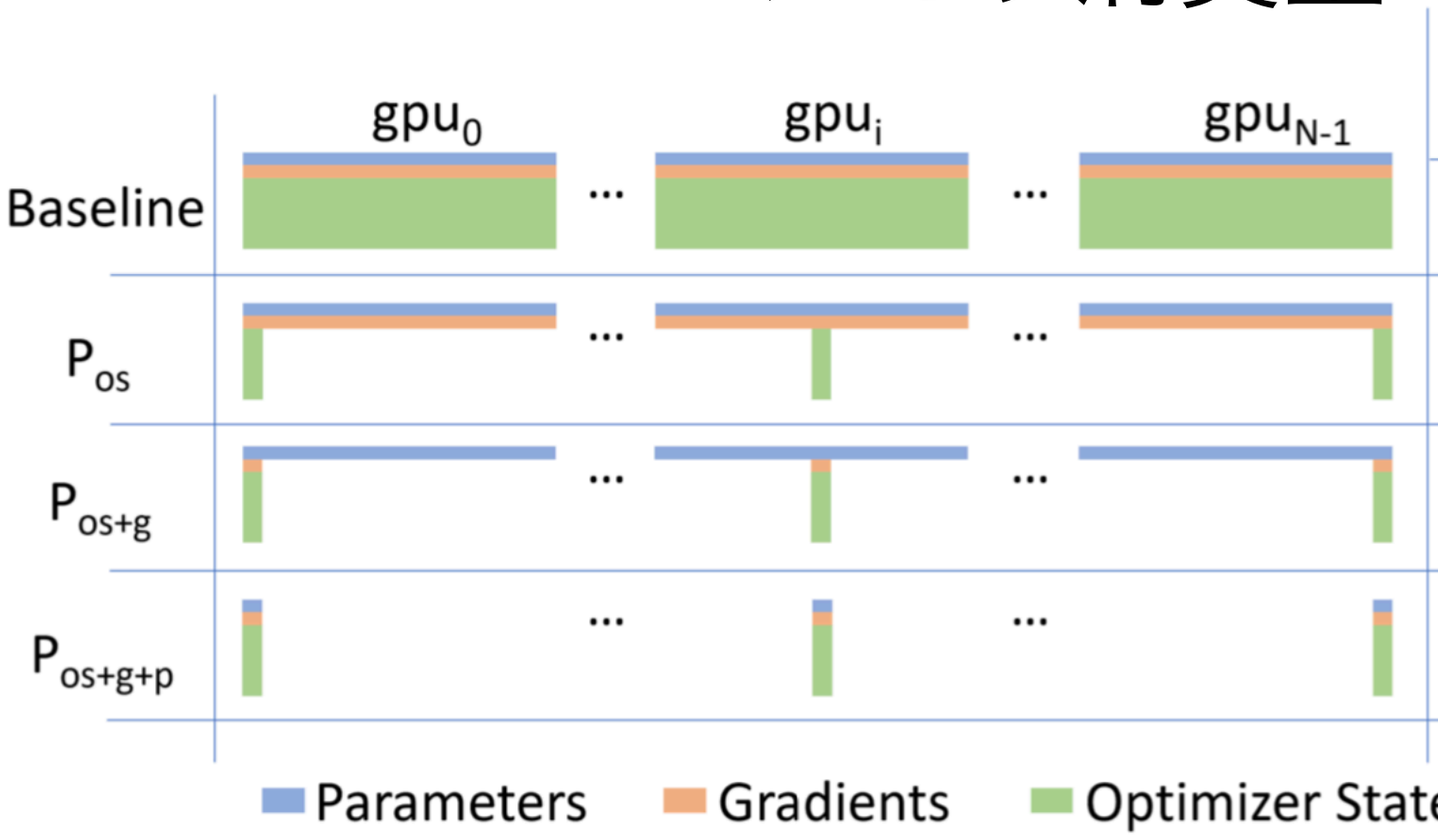
巨大な言語モデルのパラメータ数



巨大な言語モデルの学習時間

	BERT	RoBERTa	DistilBERT	XLNet
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base:** 66	**Base**: ~110 **Large**: ~340
Training Time	**Base**: 8 x V100 x 12 days* **Large:** 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base**: 8 x V100 x 3.5 days; 4 times less than BERT.	**Large:** 512 TPU Chips x 2.5 days; 5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT	2-15% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	**Base**: 16 GB BERT data **Large**: 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

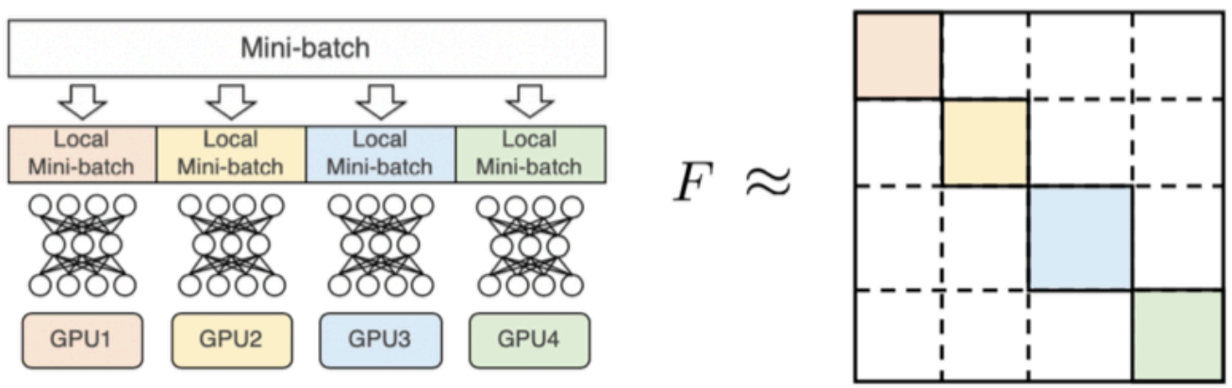
メモリ消費量の問題



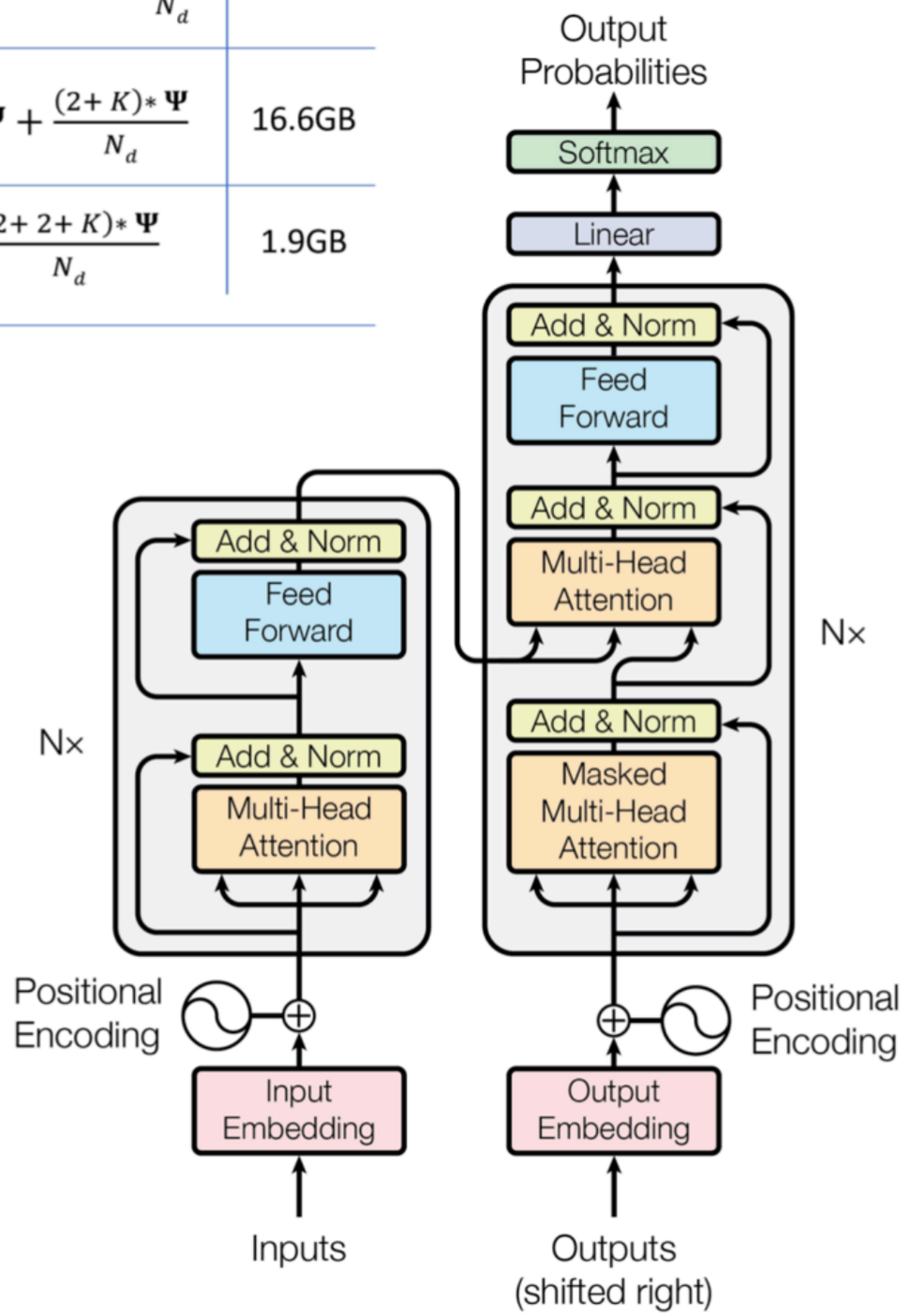
	Memory Consumed	K=12, Ψ=7.5B, N_d=64
Baseline	$(2+2+K)*\Psi$	120GB
P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB
P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB

Adamの状態変数の数
パラメータ数
データ並列数

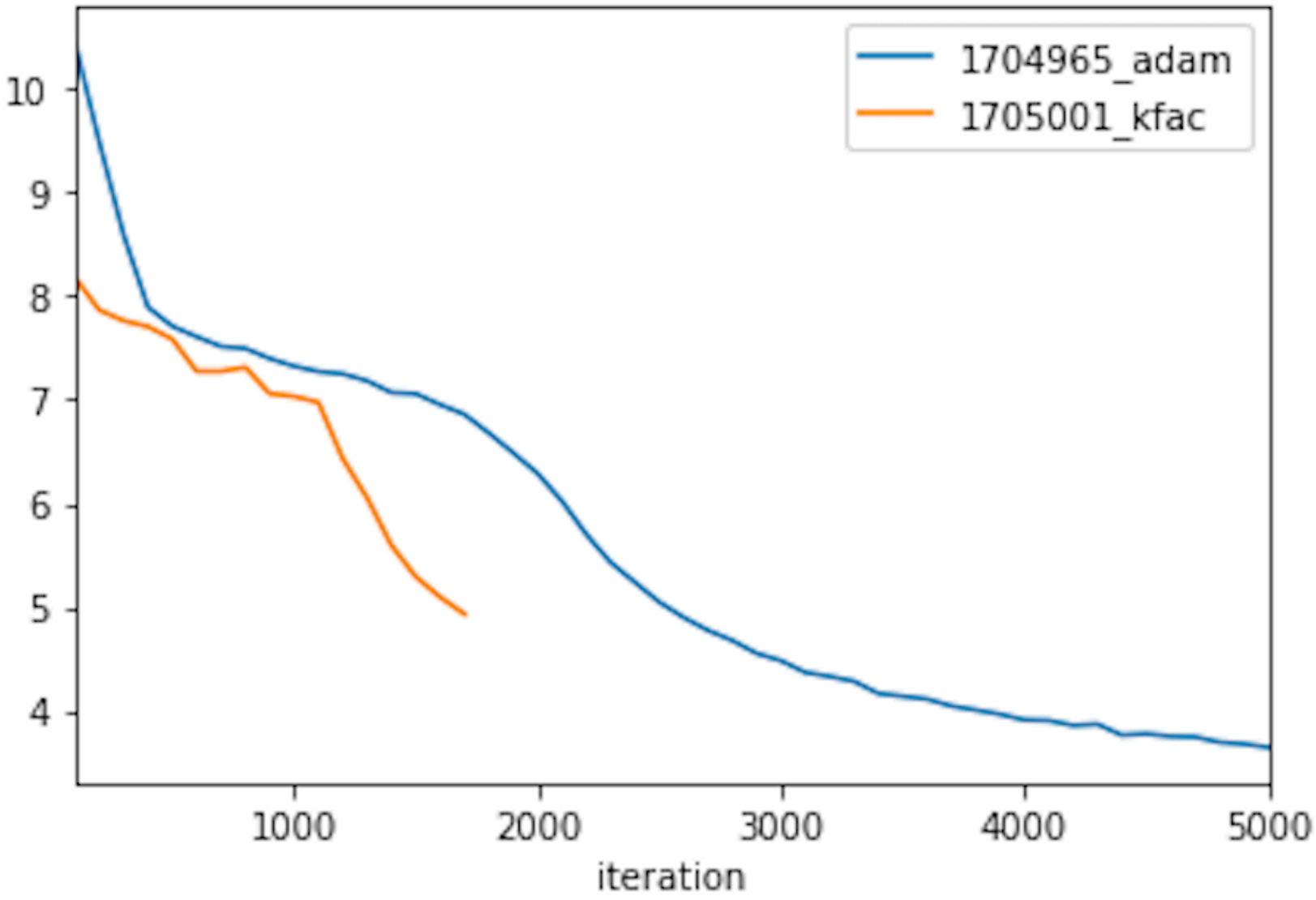
層ごとの逆行列を分散処理



クロネッカー因子分解



K-FACを用いた場合の学習曲線



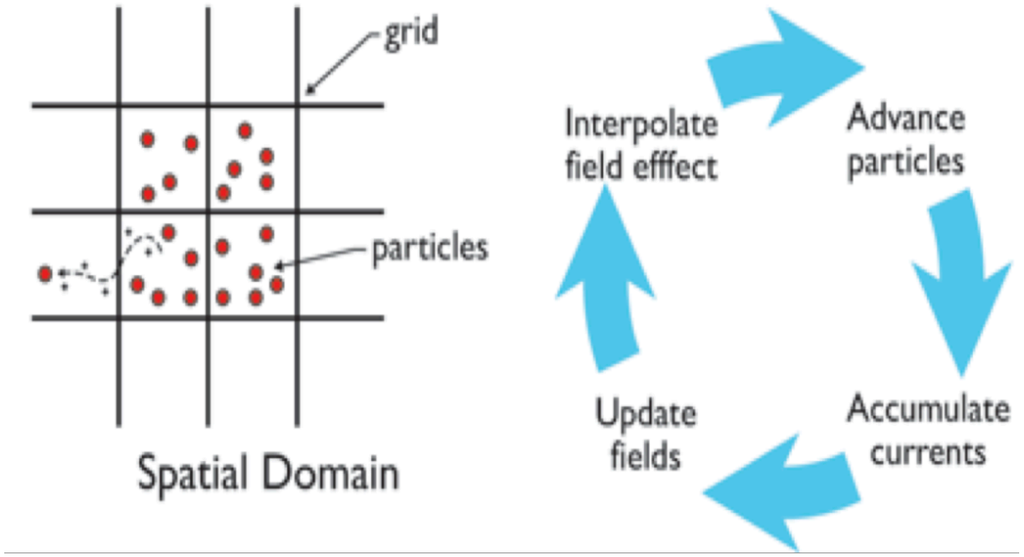
K-FACはTransformer用のチューニングが行われていないため途中で時間切れになったが、早く収束する傾向は観測された。

これはパイパーパラメータチューニングをほとんどしていないK-FACの性能であり、今後さらなる収束性の向上が見込める。

プラズマの挙動を予測する方法

Simulation

Particle-In-Cell



Gyrokinetic Vlasov Poisson

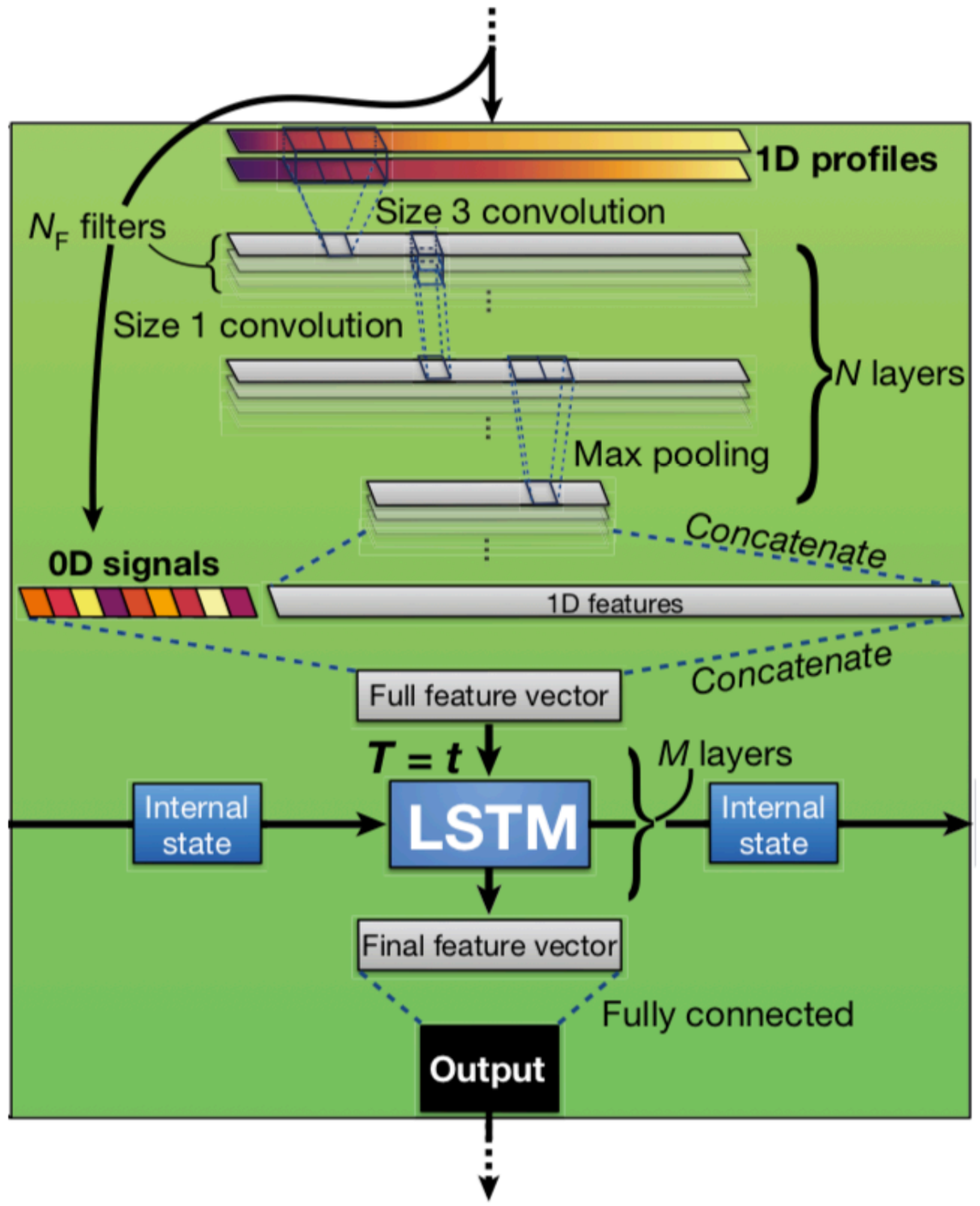
$$\frac{df_\alpha}{dt} = \frac{\partial f_\alpha}{\partial t} + \frac{d\mathbf{R}}{dt} \cdot \frac{\partial f_\alpha}{\partial \mathbf{R}} + \frac{dv_\parallel}{dt}\frac{\partial f_\alpha}{\partial v_\parallel} = 0,$$

$$\frac{d\mathbf{R}}{dt} = v_\parallel \hat{\mathbf{b}} + \mathbf{v}_E + \mathbf{v}_d,$$

$$\frac{dv_\parallel}{dt} = -\frac{1}{m_\alpha}\mathbf{b}^* \cdot (\mu \nabla B + Z_\alpha \nabla \bar{\phi})$$

$$\mathbf{b}^* = \hat{\mathbf{b}} + \frac{v_\parallel}{\Omega_\alpha}\nabla \times \hat{\mathbf{b}} \approx \hat{\mathbf{b}} + \frac{v_\parallel}{\Omega_\alpha}\frac{\hat{\mathbf{b}} \times \nabla B}{B},$$

Deep Learning



Machine Learning Workflow

Identify Signals • Classifiers

Preprocessing and feature extraction

Normalization

Train model, Hyper parameter tuning

Use model for prediction

Princeton/PPPL DL predictions now advancing to *multi-D time trace signals (beyond zero-D)*

Measured sequential data arranged in patches of equal length for training

All data placed on appropriate numerical scale ~ O(1)

e.g., Data-based with all signals divided by their standard deviation

- All available data analyzed;
- Train LSTM (Long Short Term Memory Network) iteratively;
- Evaluate using ROC (Receiver Operating Characteristics) and cross-validation loss for every epoch (equivalent of entire data set for each iteration)

Apply ML/DL software on new data

Background/Approach for DL/AI

• Deep Learning Method: distributed data-parallel approach to train deep neural networks → *Python Framework using high-level Keras library with Google Tensorflow backend*

Reference: Deep Learning with Python, François Chollet (Nov. 2017, 384 pages)

**** Major contrast with "Shallow Learning" approaches including SVM's, Random Forests, Single Layer Neural Nets, & modern Stochastic Gradient Boosting ("XG-BOOST") methods by enabling moving from ML software deployment on clusters to supercomputers:*

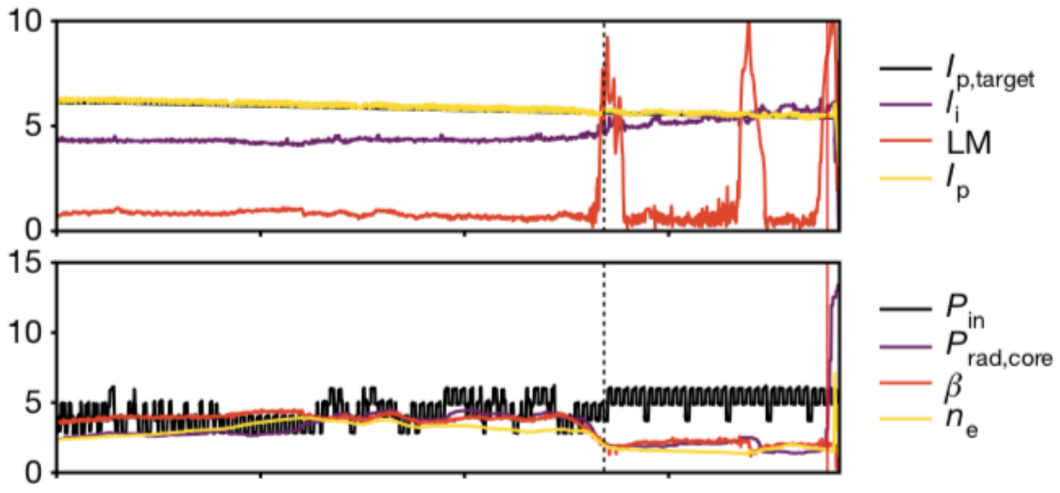
→ *Summit (ORNL); Tsubame-3 (TiTech); ABCI (U. Tokyo); .. also other architectures, e.g. –* ***Intel Systems: beyond KNL to new designs for Aurora-21 @ANL***

-- ***stochastic gradient descent (SGD)*** *used for large-scale (i.e., optimization on supercomputers) with parallelization via mini-batch training to reduce communication costs*

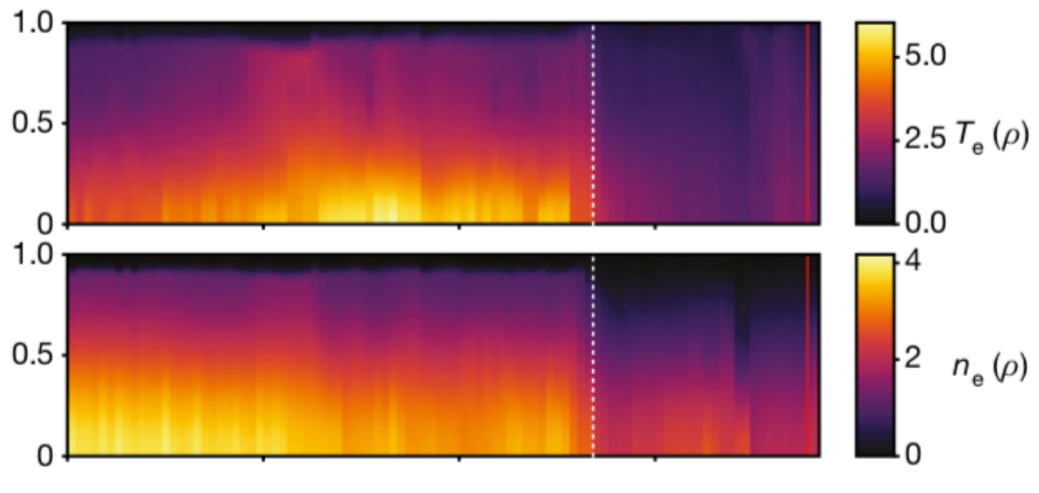
-- *DL Supercomputer Challenge: need large-scale scaling studies to examine if convergence rate saturates with increasing mini-batch size (to thousands of GPU's)*

プラズマ挙動の 1 ms未来の予測性能

FRNN0D: 点計測データ



FRNN1D: 線計測データ

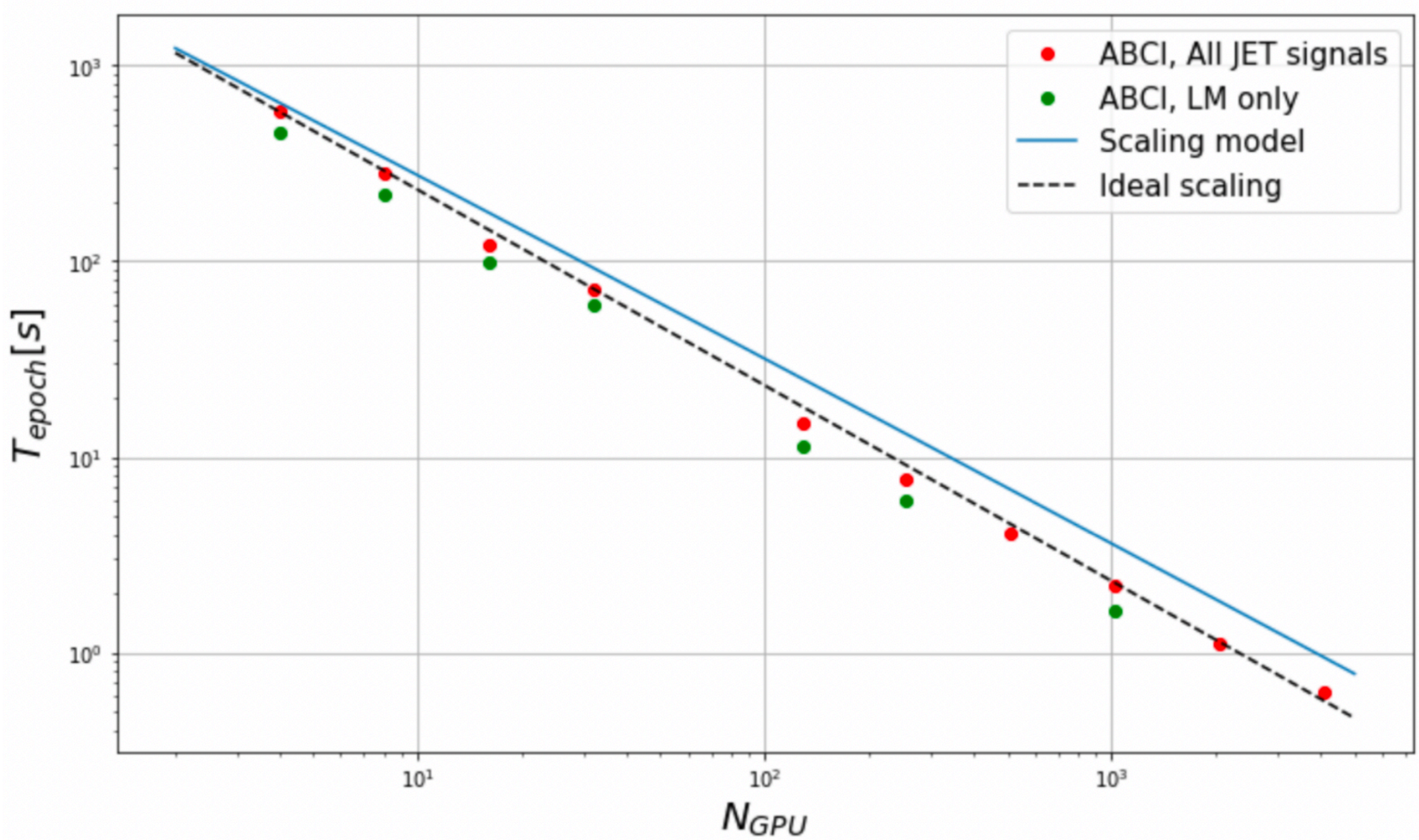


XG-Boost

	Single machine		Cross-machine		Cross-machine with 'glimpse'
Training set	DIII-D	JET (CW)	JET (CW)	DIII-D	DIII-D + δ
Testing set	DIII-D	JET (ILW)	DIII-D	JET (ILW)	JET (ILW) − δ
Best classical model	**0.937**	0.893	0.636	0.616	0.851
FRNN 0D	0.890	**0.952**	**0.761**	0.817	0.879
FRNN 1D	0.922	–	–	**0.836**	**0.911**

Kates-Harbeck et al. Nature, 568, pp. 526-531 (2019)

FRNNの並列化効率



パフォーマンスモデルよりも理想的な並列化効率に近いスケーリング

ABCIのほぼ全体である4096 GPUまで理想的な並列化効率が得られた

関連する論文発表

二次最適化を用いた巨大な言語モデルの学習

Distributed Training of Large Language Models Using Second Order Optimization, KDD, submitted

Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., Khan, M. E., ``Practical Deep Learning with Bayesian Principles'', The 33rd Conference on Neural Information Processing Systems (NeurIPS 2019).

Ueno, Y. and Yokota, R. ``Hierarchical Topology-aware Communication for Scaling Deep Learning to Thousands of GPUs'', The 19th Annual IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing (CCGrid 2019).

Osawa, K., Tsuji, Y., Ueno, Y., Naruse, A., Yokota, R., and Matsuoka, S., ``Large-scale Distributed Second-order Optimization Using Kronecker-factored Approximate Curvature for Deep Convolutional Neural Networks'', Conference on Computer Vision and Pattern Recognition (CVPR 2019).

FRNNを用いたプラズマ挙動予測

Svyatkovskiy, A., Kates-Harbeck, J., and Tang, W., planning to submit to SC'20.