# RandNLA for Efficient Deep Learning

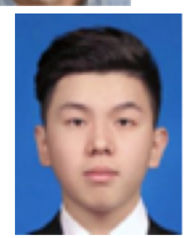
Zhewei Yao

University of California, Berkeley

Septemper 2019





Berkeley
UNIVERSITY OF CALIFORNIA

riselab
UC Berkeley

## Key Words

Randomized Numerical Linear Algebra

Optimization

Hessian

Eigen Computation

## Key Words

Randomized Numerical Linear Algebra

Optimization

Hessian

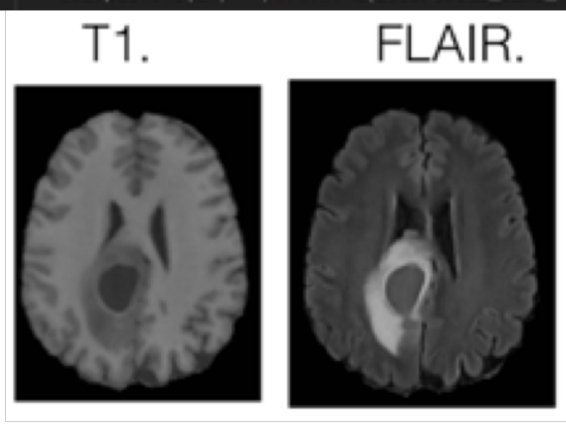
Eigen Computation

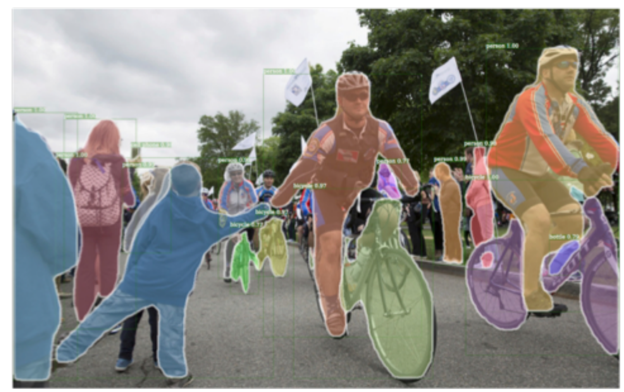
Deep Learning

# Outline

- **Background**
- Efficient Deep Learning Training
- Efficient Deep Learning Inference
- Conclusions

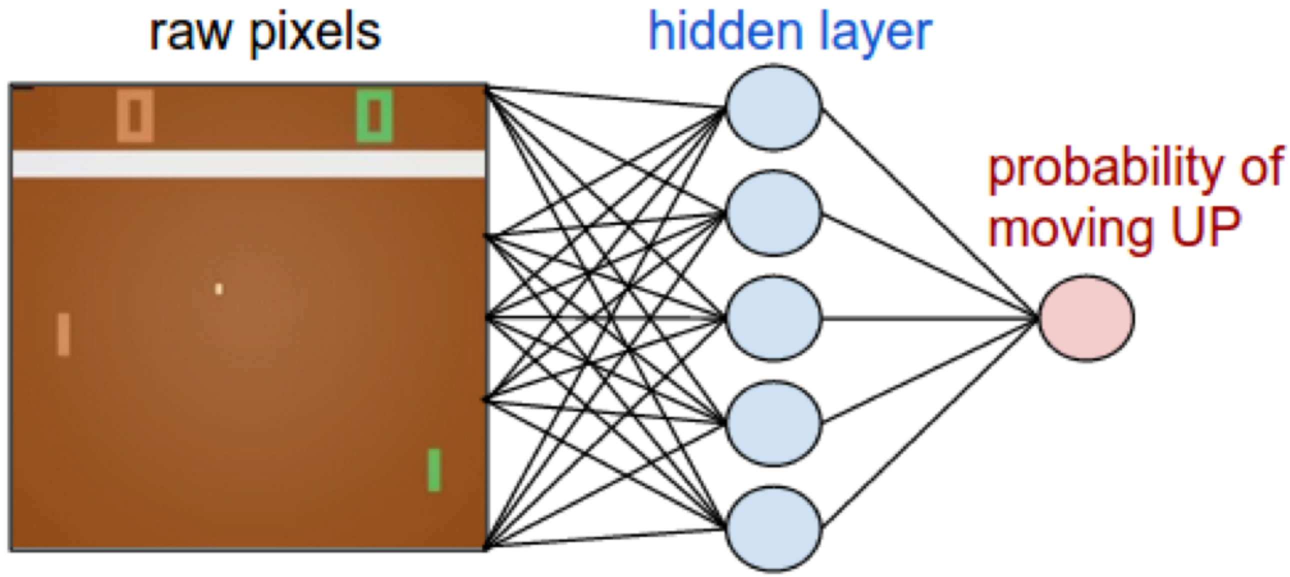
# Deep Learning is everywhere

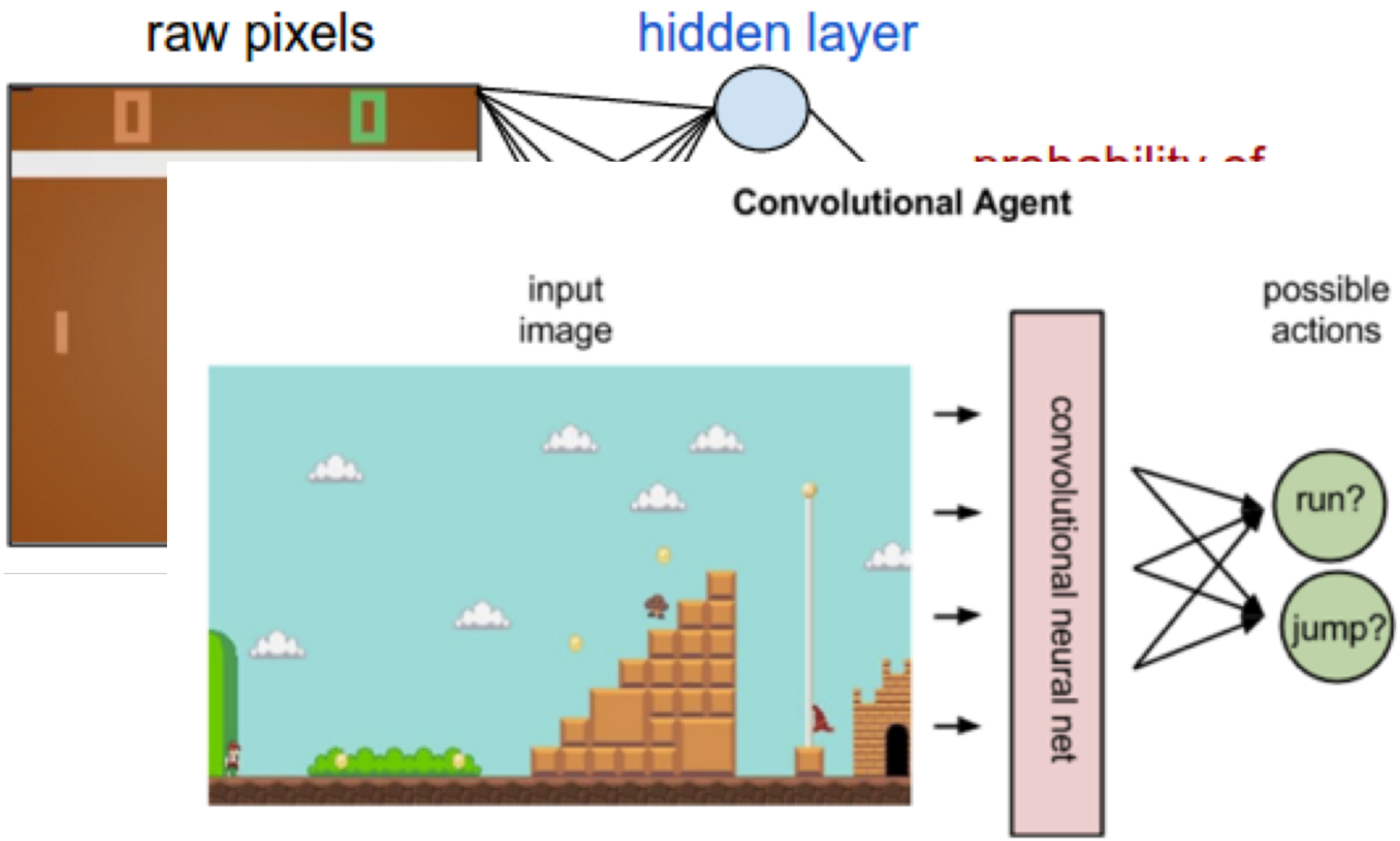



B. Wu, F. Iandola, P. Jin, and K. Keutzer. "SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving."CVPR Workshop
A. Gholami, S. Subramanian, V. Shenoy, N. Himthani, X. Yue, S. Zhao, P. Jin, K. Keutzer, G. Biros, A novel domain adaptation framework for medical image segmentation, BRATS, MICCAI 2018
A Semantic Segmentation using Detectron, Facebook Research

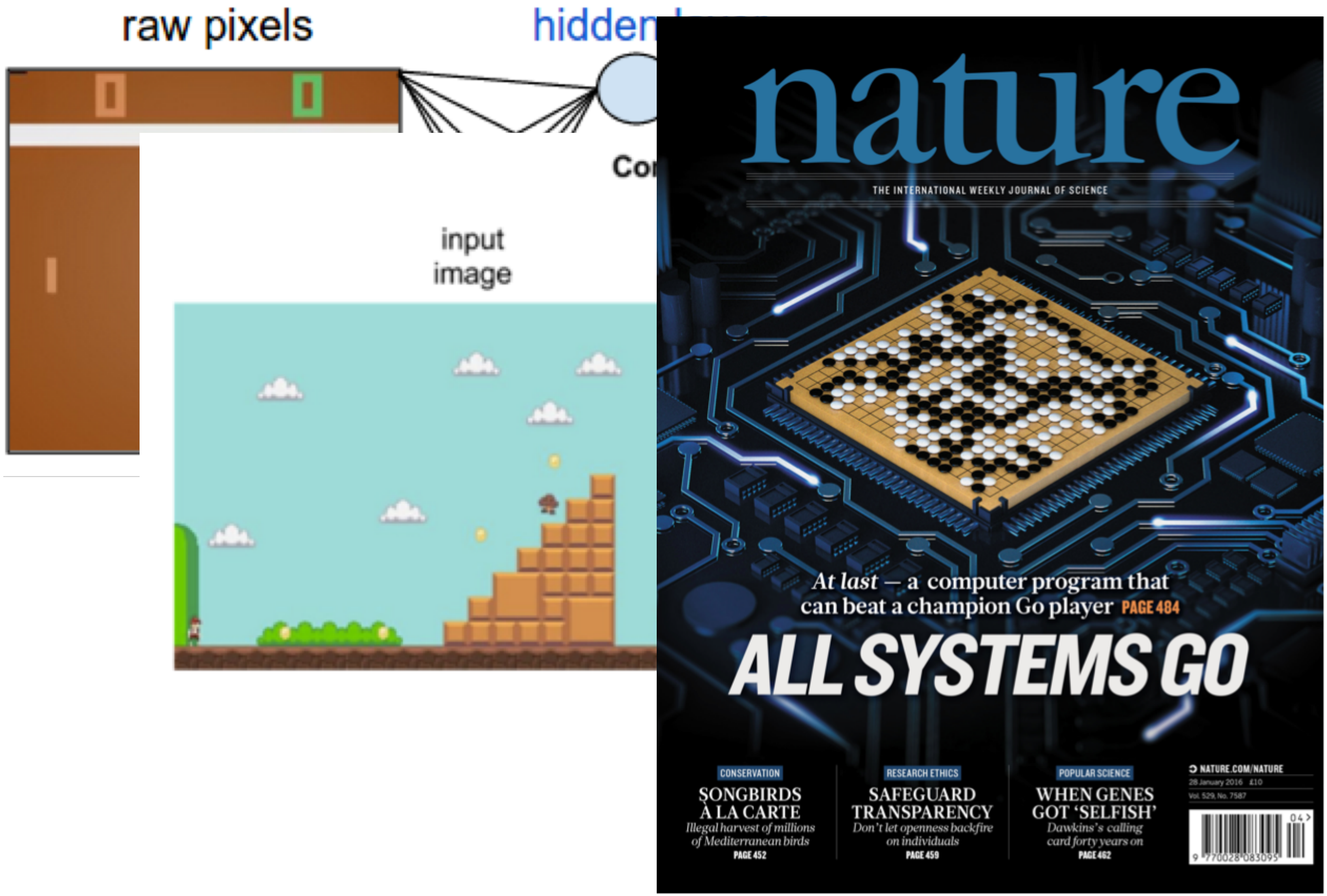
# Deep Learning is everywhere



# Deep Learning is everywhere

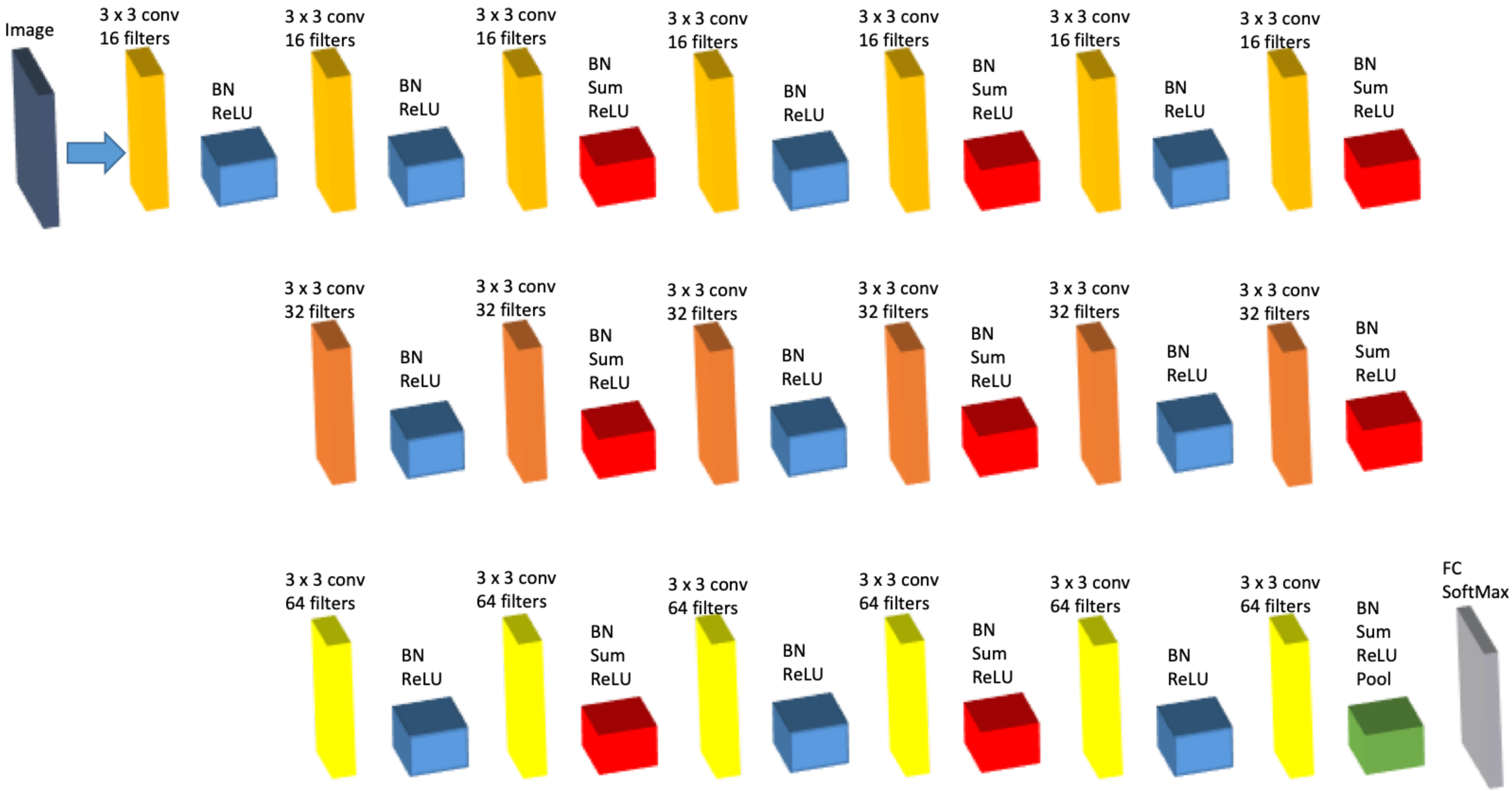


# Deep Learning is everywhere



# Deep Learning is everywhere

# What Neural Network Looks Like?




An example of deep neural network on image classification problem.

## The size of both NNs and datasets

Image classification:

- Model size: 20M parameters (ResNet50)
- Dataset size: 1.2M images (224x224x3)
- Training time: 3 days on one V100 GPU

## The size of NNs and datasets

Image classification:

- Model size: 20M parameters (ResNet50)
- Dataset size: 1.2M images (224x224x3)
- Training time: 3 days on one V100 GPU

Natural Language Processing:

- Model size: 110M parameters (BERT-base)
- Dataset size: 50M sentences
- Training time: 14 days on eight V100 GPUs

# Eigenvalue Computation

We consider a supervised learning framework where the goal is to mini

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(z_i, \theta).$$

Let us denote the gradient of $L(\theta)$ w.r.t. $\theta$ by $g$. Then for a random ve

$$\frac{\partial(g^T v)}{\partial\theta} = \frac{\partial g^T}{\partial\theta} v + g^T \frac{\partial v}{\partial\theta} = \frac{\partial g^T}{\partial\theta} v = Hv,$$

Where the second equation comes from the fact that $v and g$ are Independant.

## Eigenvalue Computation

**Algorithm 2:** Power Iteration for Eigenvalue Computation

**Input:** Parameter: $\theta$.

Compute the gradient of $\theta$ by backpropagation, *i.e.*, $g = \frac{dL}{d\theta}$.

Draw a random vector $v$ (same dimension as $\theta$).

Normalize $v$, $v = \frac{v}{\|v\|_2}$

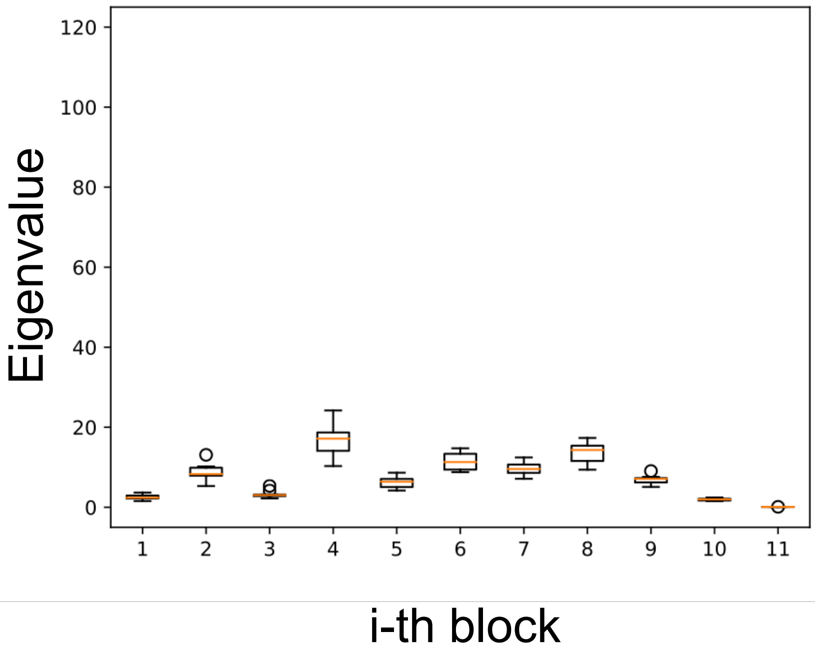
**for** $i = 1, 2, \ldots, n$ **do** `// Power Iteration`

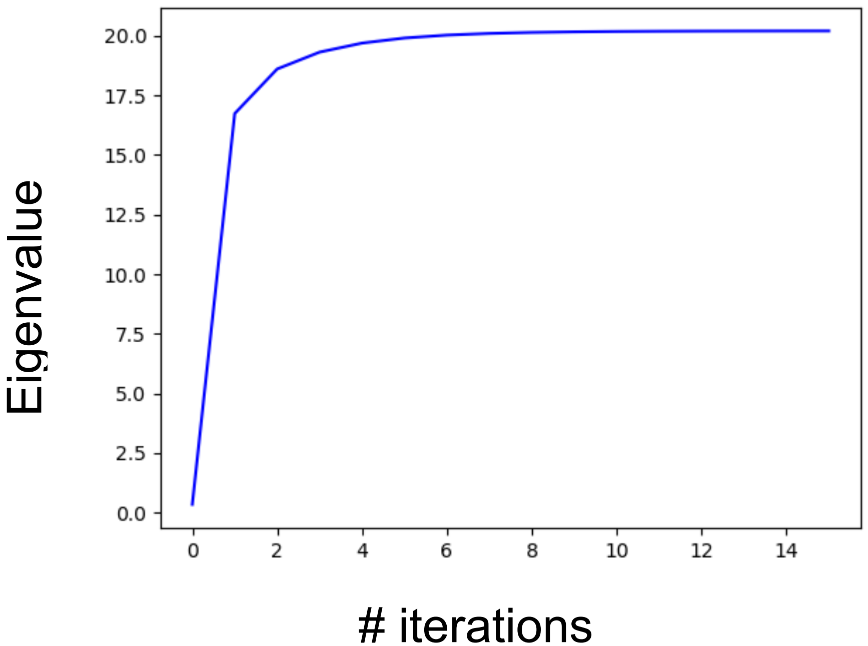
- Compute $gv = g^T v$ `// Inner product`
- Compute $Hv$ by backpropagation, $Hv = \frac{d(gv)}{d\theta}$ `// Get Hessian vector product`
- Normalize and reset $v$, $v = \frac{Hv}{\|Hv\|_2}$

Remaining Questions:

- How many power iterations do we need to compute the top eigenvalues?
- How many data do we need to get a good estimation?

# Eigenvalue Computation Illustration




Top eigenvalue for different blocks using batch size 128 with 10 runs: the variance is very small.

Power iterations needed to compute top eigenvalue is around 10.

# Outline

- Background
- **Efficient Deep Learning Training**
- Efficient Deep Learning Inference
- Conclusions

## High Level Outline

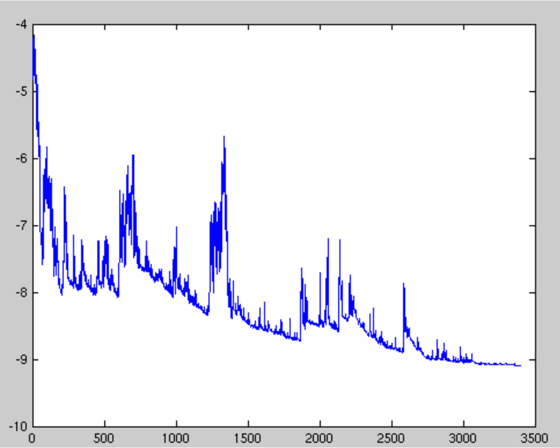
- DNN design requires training on large datasets
  - Time consuming
  - Need fast training -> parallelization -> large batch
- Large batch training does not work:
  - Degrades accuracy
  - Poor robustness to adversarial inputs
  - Existing solutions requires extensive hyper-parameter tuning

# Stochastic Gradient Descent (SGD)

Assume $L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(z_i, \theta)$

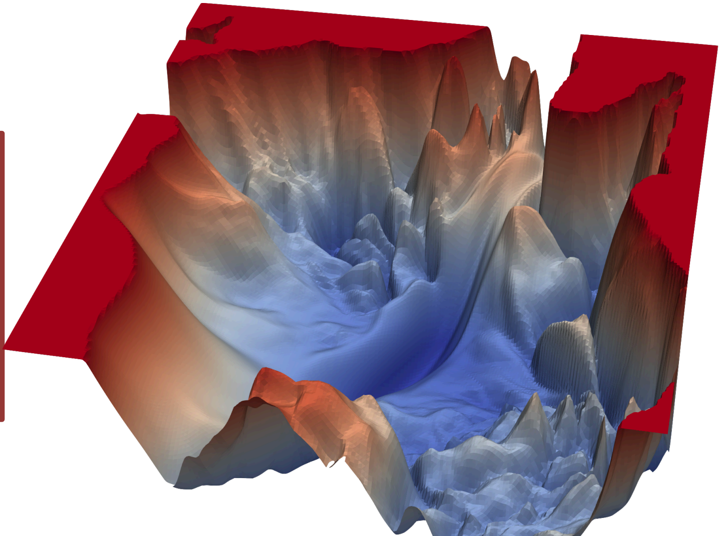
GD: $\theta^{t+1} = \theta^t - \alpha \nabla L(\theta^t)$

Pure SGD: compute gradient using 1 sample




In practice: $\theta^{t+1} = \theta^t - \alpha \frac{1}{b}\sum_{i=1}^{b} \nabla l(x_i, \theta^t)$

Mini-batch: compute gradient using b samples

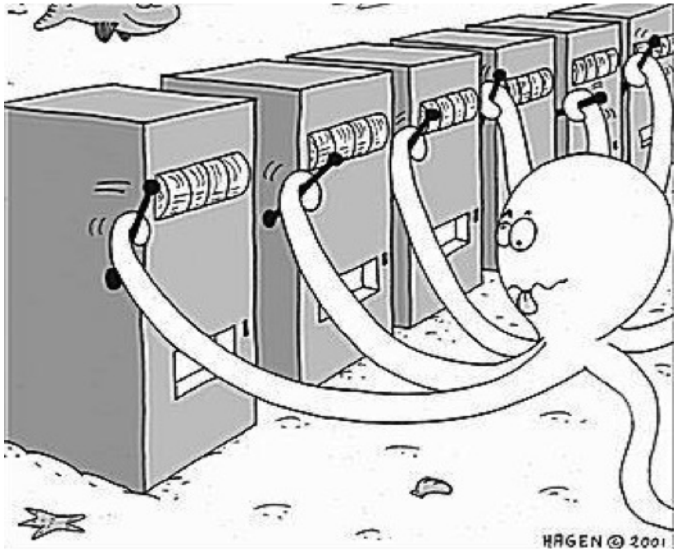


- Actually the name is a misnomer, *this is not a "descent" method*

Image from https://www.cs.umd.edu/~tomg/projects/landscapes/

## Many many knobs

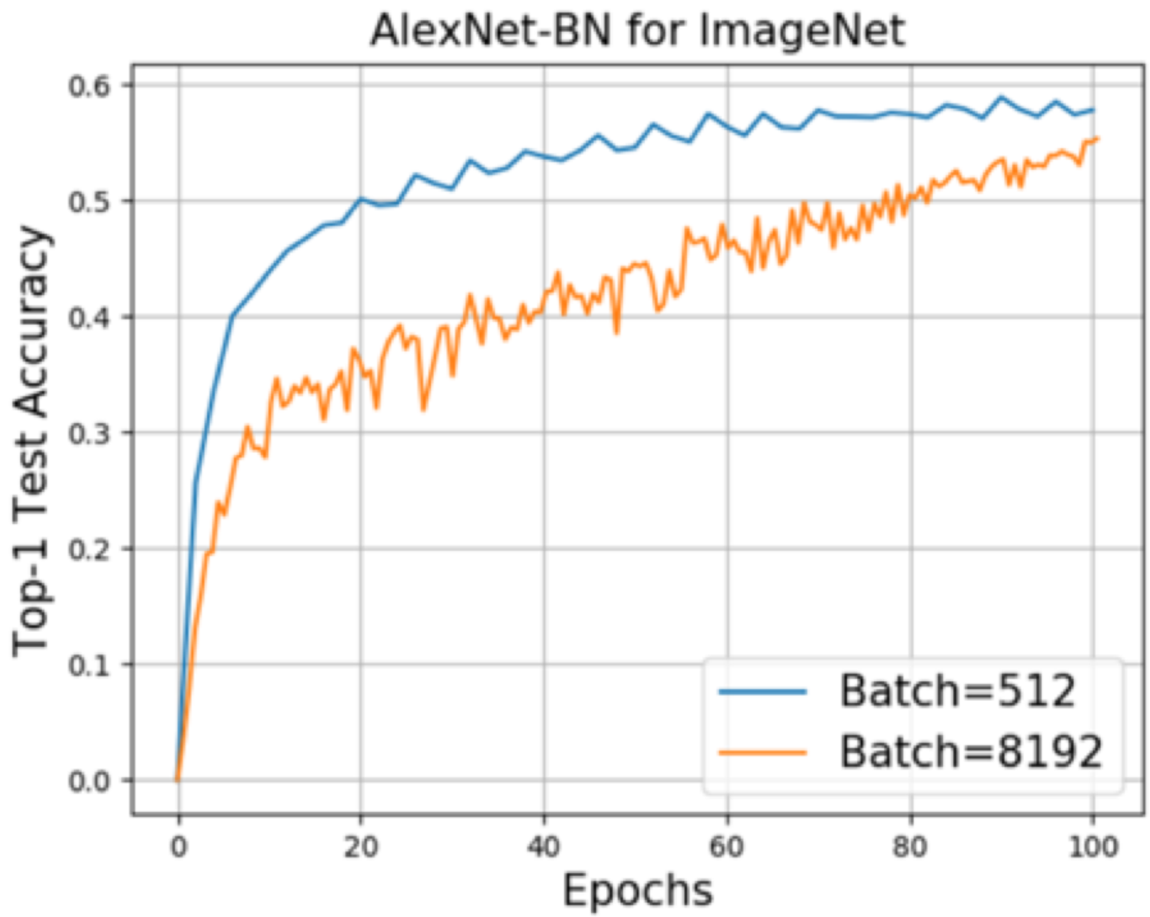
- SGD is very sensitive to hyper-parameters and in particular batch size
- Batch size inter dependent with:
  - Degradation in accuracy
  - Poor generalizability
  - Robustness of model
  - Training time
  - Parallel Scalability




$$\theta^{t+1} = \theta^{t} - \alpha \frac{1}{b} \sum_{i=1}^{b} \nabla l(x_i, \theta^{t})$$

## Degradation in Accuracy

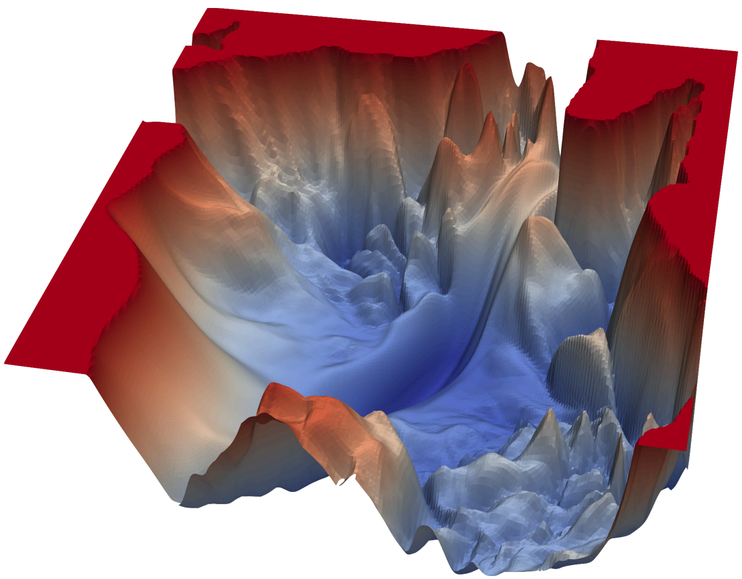
Larger Batch often leads to degradation in accuracy

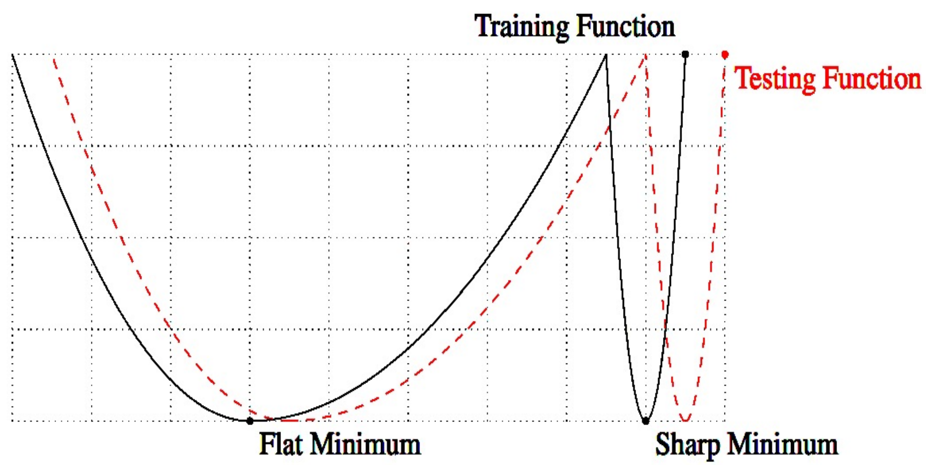



Ginsburg, Boris, Igor Gitman, and Yang You. "Large Batch Training of Convolutional Networks with Layer-wise Adaptive Rate Scaling." arxiv:1708.03888.

## Poor Generalization

- Why large batch suffers from poor generalization performance?
  - A common belief is that large batch training gets attracted to “sharp minimas”
  - Another theory is that large batch may get stuck in saddle points






Loss landscape from https://www.cs.umd.edu/~tomg/projects/landscapes/
Keskar, Nitish Shirish, et al. "On large-batch training for deep learning: Generalization gap and sharp minima." ICLR’16 (arXiv:1609.04836)

# Hessian Based Adaptive Batch Size with Adversarials

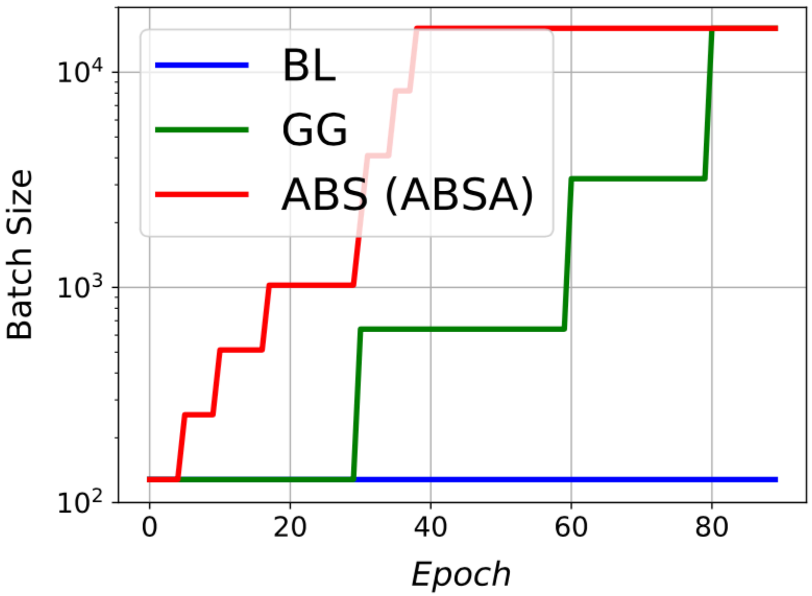
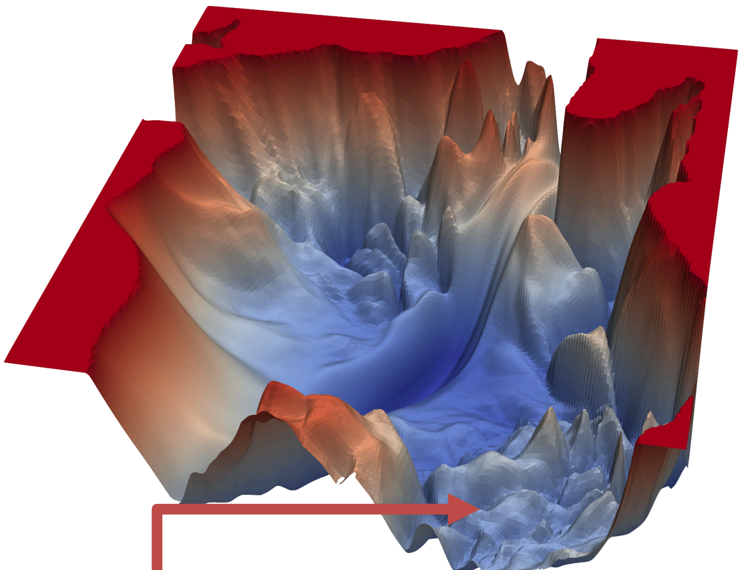
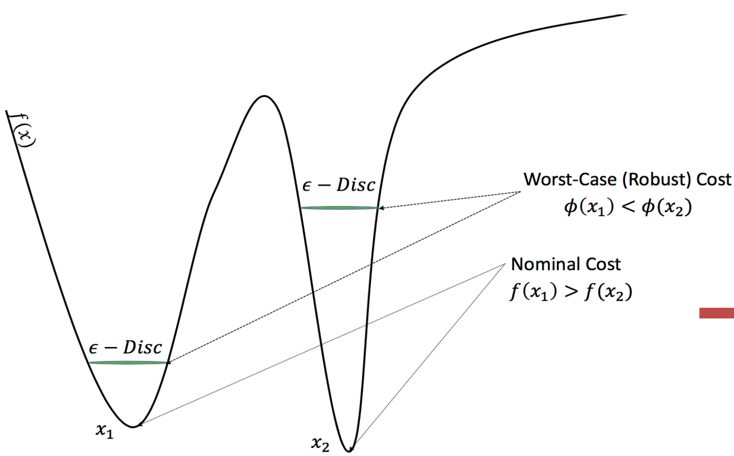



*Illustration of batch size schedules of adaptive batch size as a function of training epochs.*






Adversarials (robust training) can smooth out sharp "local minimas".

Yao, Gholami, Keutzer, and Mahoney. Large Batch Size Training of Neural Networks with Adversarial Training and Second-Order NeurIPS, 2018.

## Robust Optimization and Regularization

- There is an interesting connection between the solution to robust optimization and a properly regularized problem
- There is an interesting connection between the solution to robust optimization and a properly regularized problem

$$\min_{x} \max_{\|A\|_{2,\infty} \leq \rho} \| (A + \Delta A)x - b \|$$

$$\Leftrightarrow \min \| Ax - b \| + \rho \| x \|_1$$


El Ghaoui, Laurent, and Hervé Lebret. "Robust solutions to least-squares problems with uncertain data." SIAM Journal on matrix analysis and applications 18.4 (1997): 1035-1064.
Huan Xu, Constantine Caramanis, and Shie Mannor. Robust regression and lasso. In *Advances in Neural Information Processing Systems*, pages 1801–1808, 2009.

# Results – Cifar10

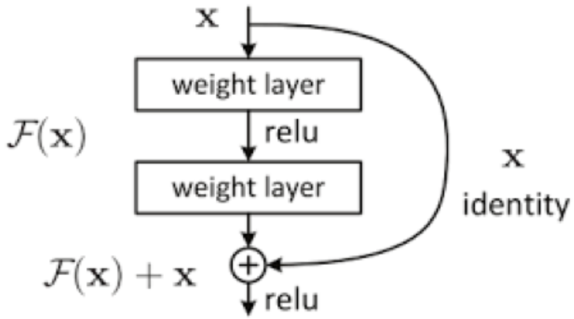
Cifar10 has ten classes

- ~5000 examples per class
- Total 50,000 training images
- 10,000 testing images

# Results – Cifar10 — ResNet20

Our proposed method (ABSA) achieves

better performance




ResNet20 on Cifar10

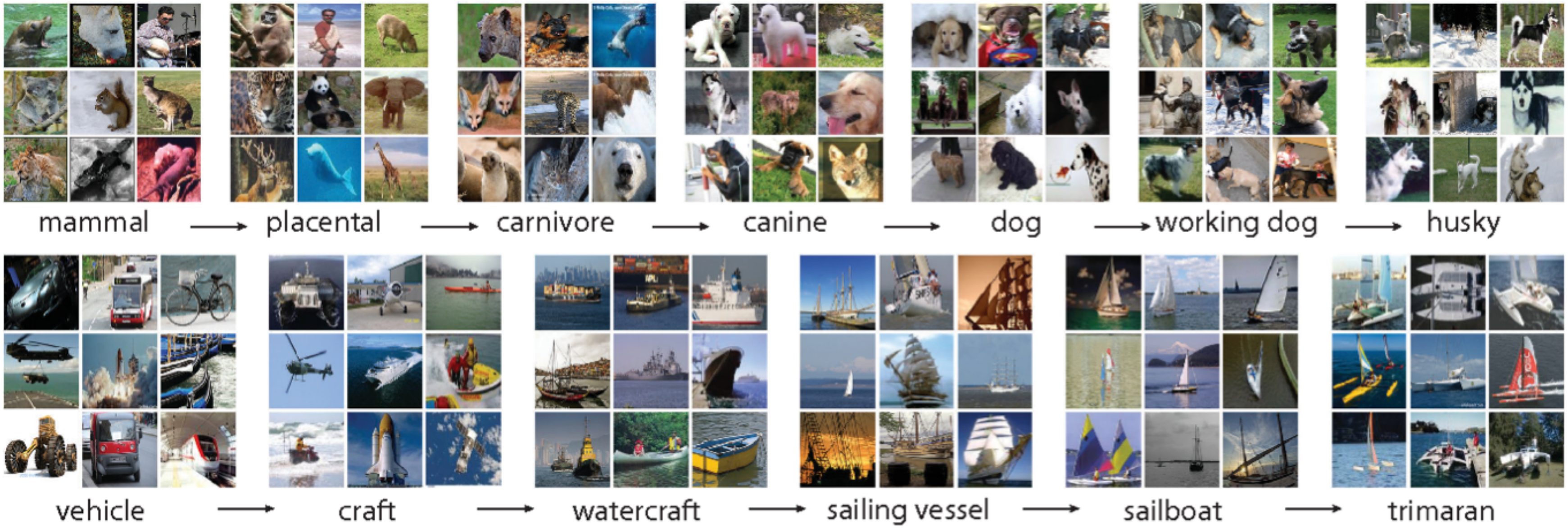
| | BL | | FB | | GG | | ABS | | ABSA | |
|---|---|---|---|---|---|---|---|---|---|---|
| BS | Acc. | # Iters | Acc. | # Iters | Acc. | # Iters | Acc. | # Iters | Acc. | # Iters |
| 128 | 83.05 | 35156 | N.A. | N.A. | N.A. | N.A. | N.A. | N.A. | N.A. | N.A. |
| 640 | 81.01 | 7031 | **84.59** | 7031 | 83.99 | 16380 | 83.30 | 10578 | 84.52 | 9631 |
| 3200 | 74.54 | 1406 | 78.70 | 1406 | 84.27 | 14508 | 83.33 | 6375 | **84.42** | 5168 |
| 5120 | 70.64 | 878 | 74.65 | 878 | 83.47 | 14449 | 83.83 | 6575 | **85.01** | 6265 |
| 10240 | 68.75 | 439 | 30.99 | 439 | 83.68 | 14400 | 83.56 | 5709 | **84.29** | 7491 |
| 16000 | 67.88 | 281 | 10.00 | 281 | 84.00 | 14383 | 83.50 | 5739 | **84.24** | 5357 |

FB: Goyal, Priya, et al. "Accurate, large minibatch SGD: training imagenet in 1 hour." arXiv preprint arXiv:1706.02677 (2017).
GG: Smith, Samuel L., Pieter-Jan Kindermans, and Quoc V. Le. "Don't Decay the Learning Rate, Increase the Batch Size." arXiv preprint arXiv:1711.00489 (2017).
ABS/ABSA: Z. Yao, A. Gholami, K. Keutzer, M. Mahoney, Large Batch Size Training of Neural Networks with Adversarial Training and Second-Order Information, (under review)

# Results – ImageNet

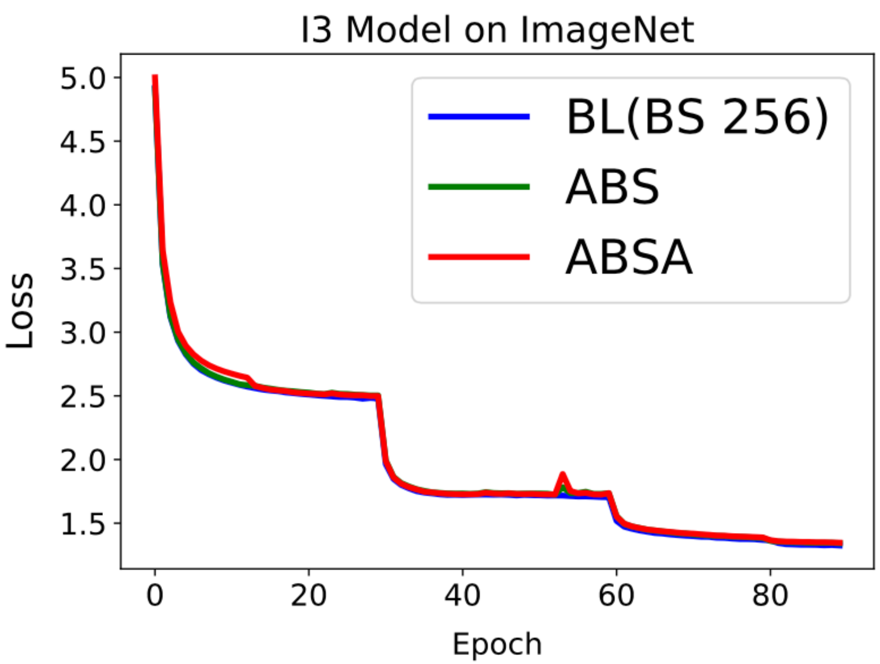
ImageNet consists of 1000 classes

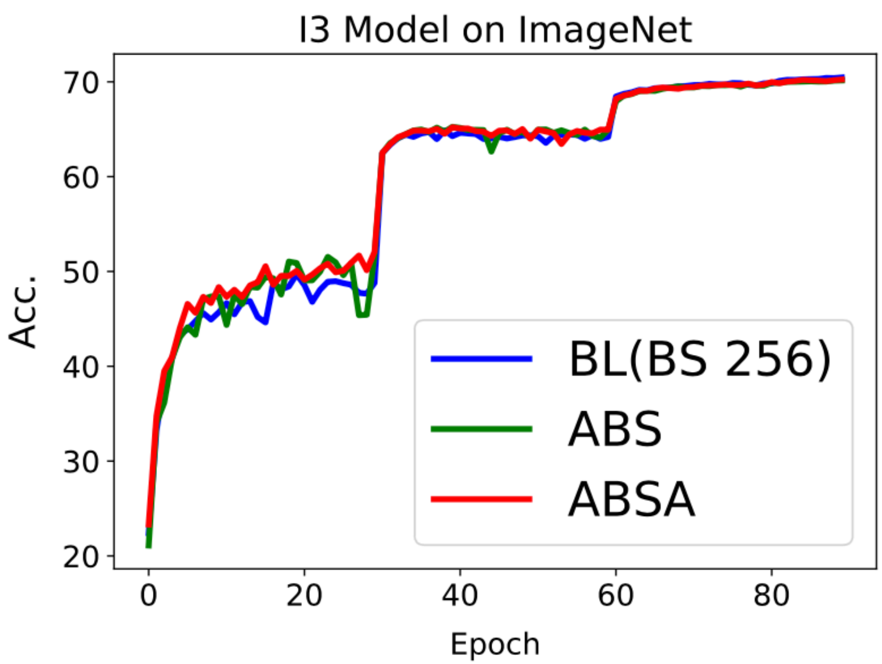
- Total 1,2M training images
- 50,000 testing images



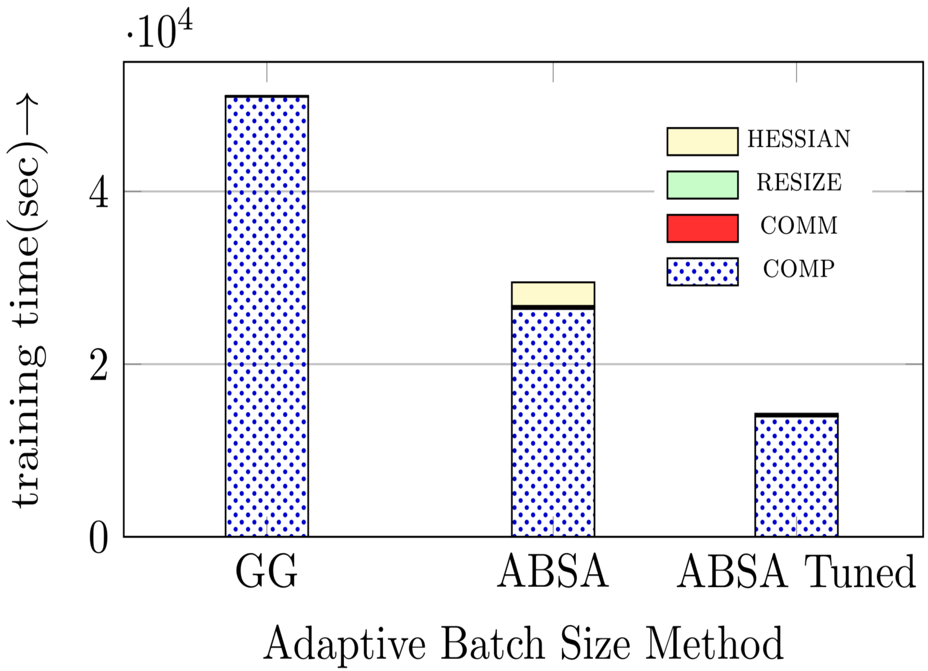
## Results – ImageNet – ResNet18

- Baseline:
  - 450k SGD iterations, 70.4% validation accuracy
- ABSA:
  - 66k SGD iterations, 70.2% validation accuracy



## Results – ImageNet – Actually Running Time




Amazon Website Service

p3.16x large (8 V100)

Small overhead of

- Hessian Computation
- Communication Time

| Method | Comp | Comm | Resize | Hess | Total | Speedup |
|---|---|---|---|---|---|---|
| Baseline | 125073 | N/A | N/A | N/A | 125073 | 1x |
| GG | 50965 | 54 | 40 | N/A | 51059 | 2.45x |
| ABSA | 26404 | 230 | 95 | 2746 | 29475 | 4.24x |
| ABSA Tuned | 13935 | 58 | 39 | 220 | **14252** | **8.78x** |

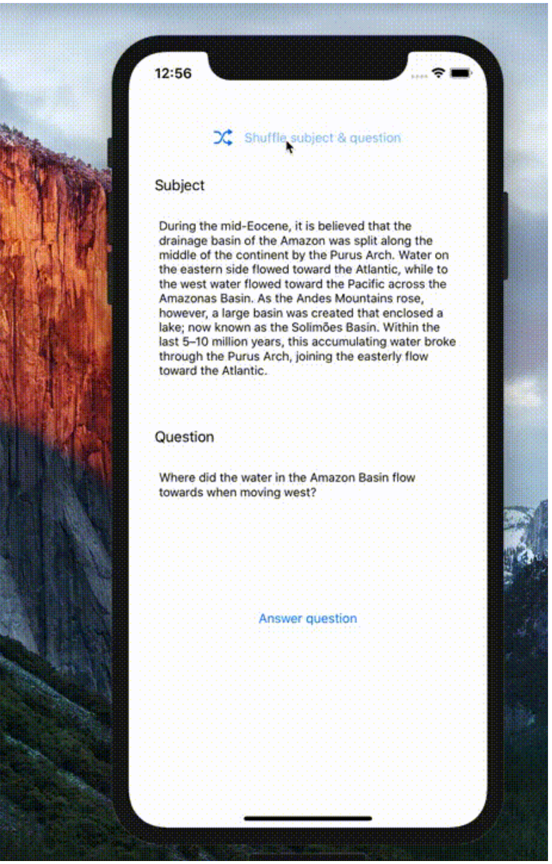
# Outline

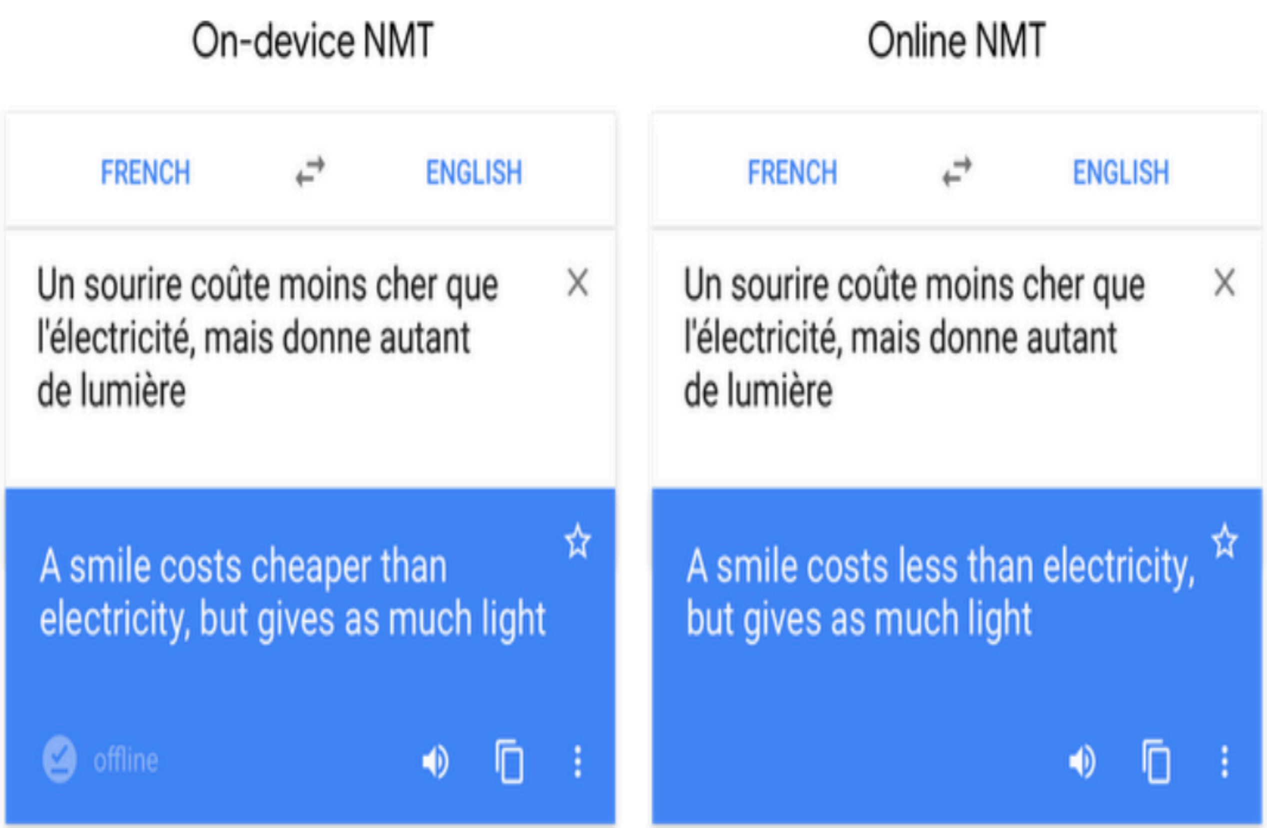
- Background
- Efficient Deep Learning Training
- **Efficient Deep Learning Inference**
- Conclusions

# Why do we need model compression

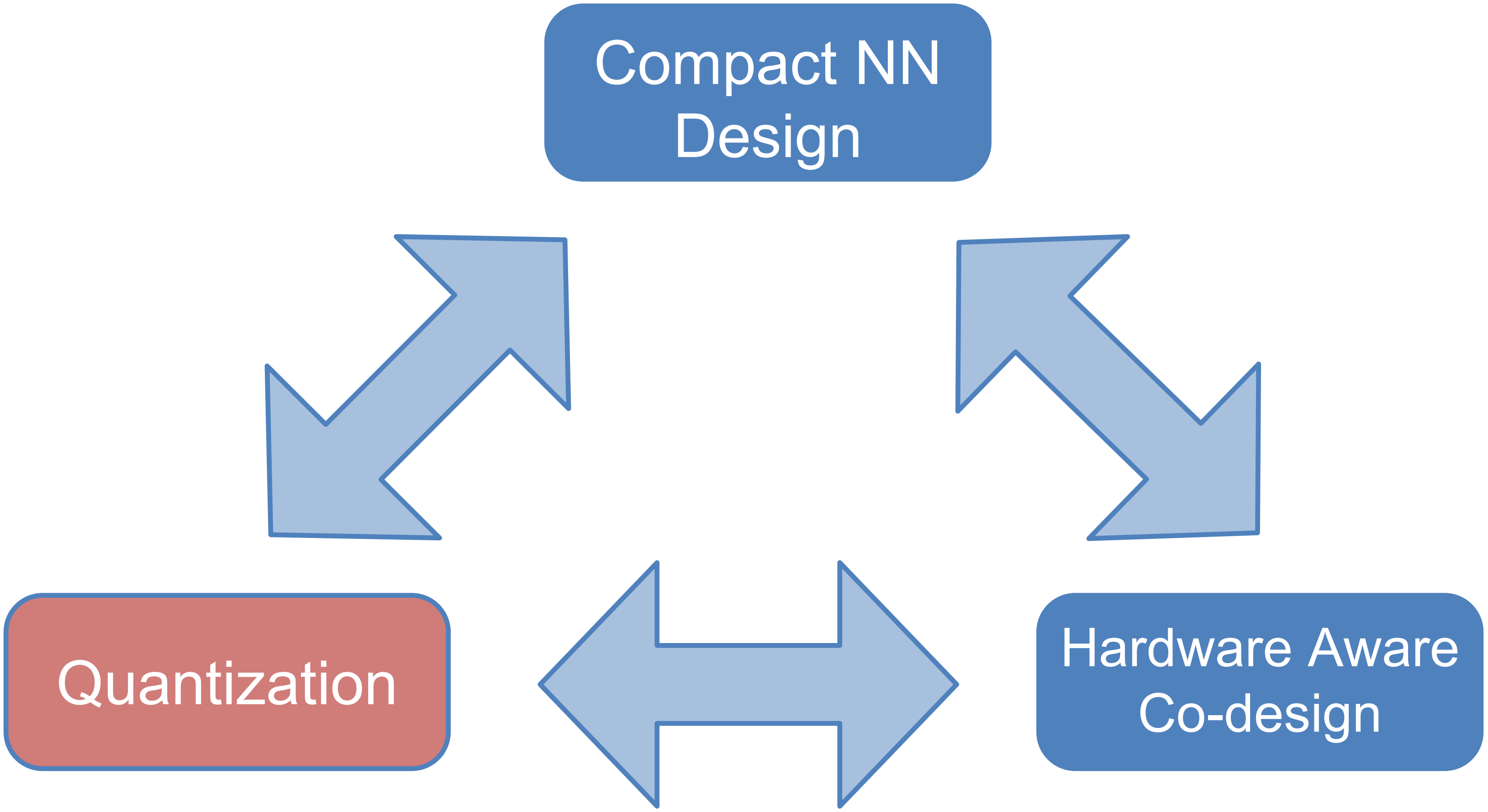
Natural Language Processing:

- Model size: 110M parameters (BERT-base)



# Existing methods



## Benefit of Quantization

- Significantly reduce memory access volume

- Allows use of reduces precision ALUs **->** Faster Inference

g execution on embedded

| Operation | Energy [pJ] |
| --- | --- |
| 32 bit int ADD | 0.1 |
| 32 bit float ADD | 0.9 |
| 32 bit Register File | 1 |
| 32 bit int MULT | 3.1 |
| 32 bit float MULT | 3.7 |
| 32 bit SRAM Cache | 5 |
| **32 bit DRAM Memory** | **640** |

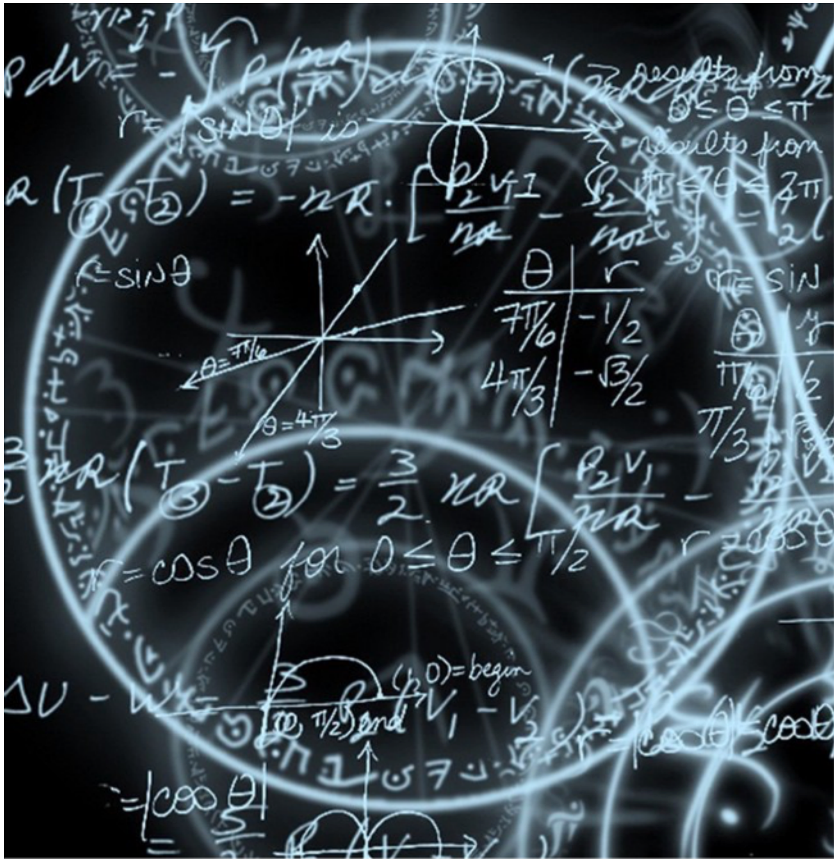
Table: Courtesy of S. Han

**6400x**

# Quantization: An Dark Art

Quantization is a very promising approach but:

- Very hard to get right for a new model/dataset
- Lots of “tricks” and expensive hyper-parameter tuning



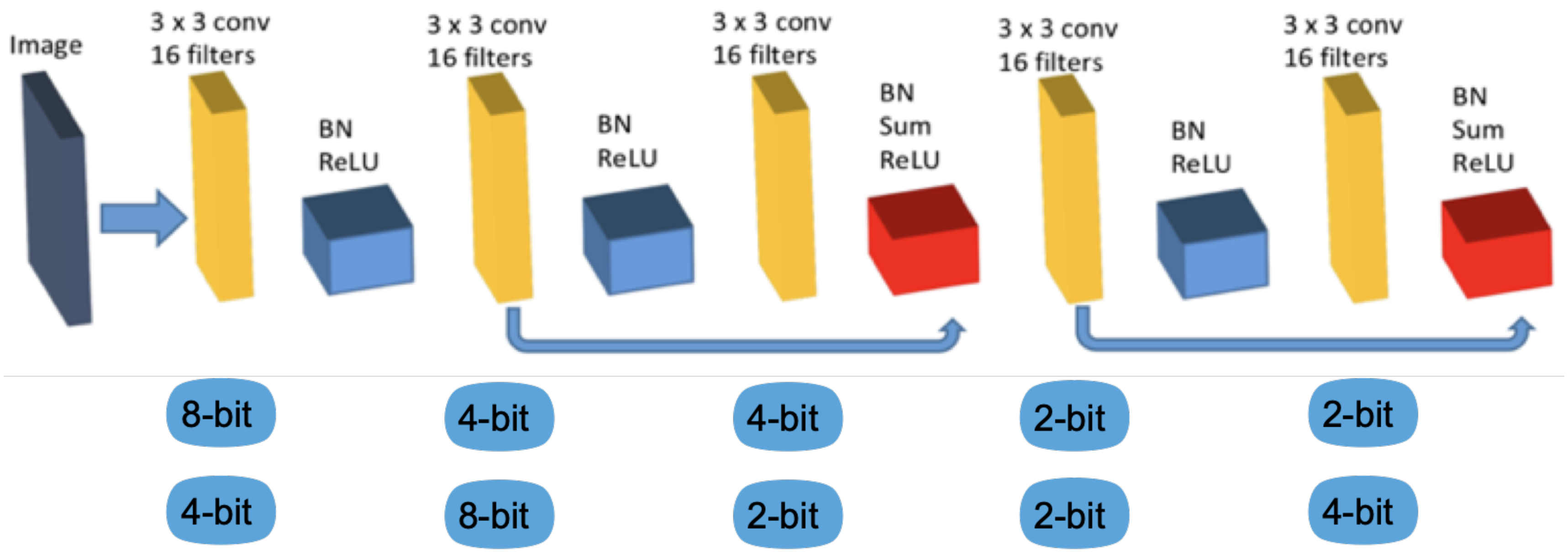
## Hessian AWare Quantization

Contributions of HAWQ:

- A systematic, second-order algorithm for inference quantization
- Novel compression results exceeding all existing state-of-the-art methods for Classification, Object Detection, and NLP
- No more ad-hoc tricks

Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, 2019. HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision. *ICCV'19 (arXiv:1905.03696)*.
S. Sheng, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. Mahoney, K. Keutzer, Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT

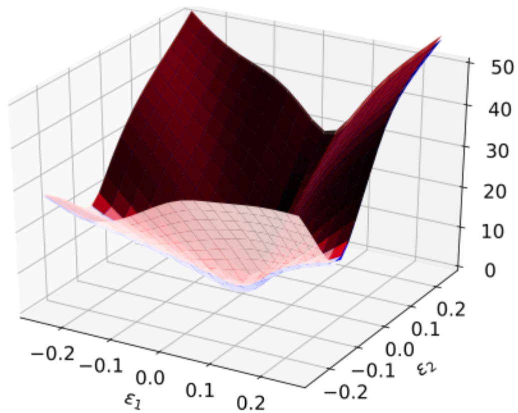
# Mixed-Precision: Exponential Search Space

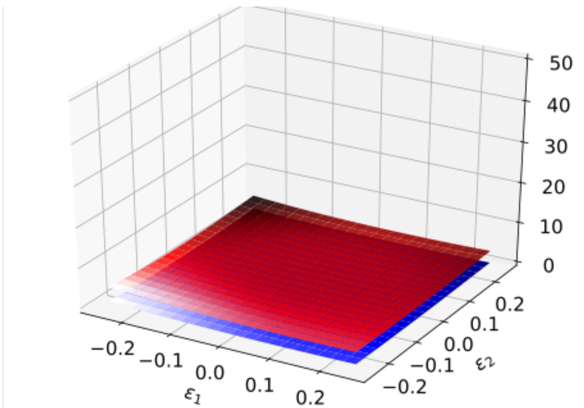


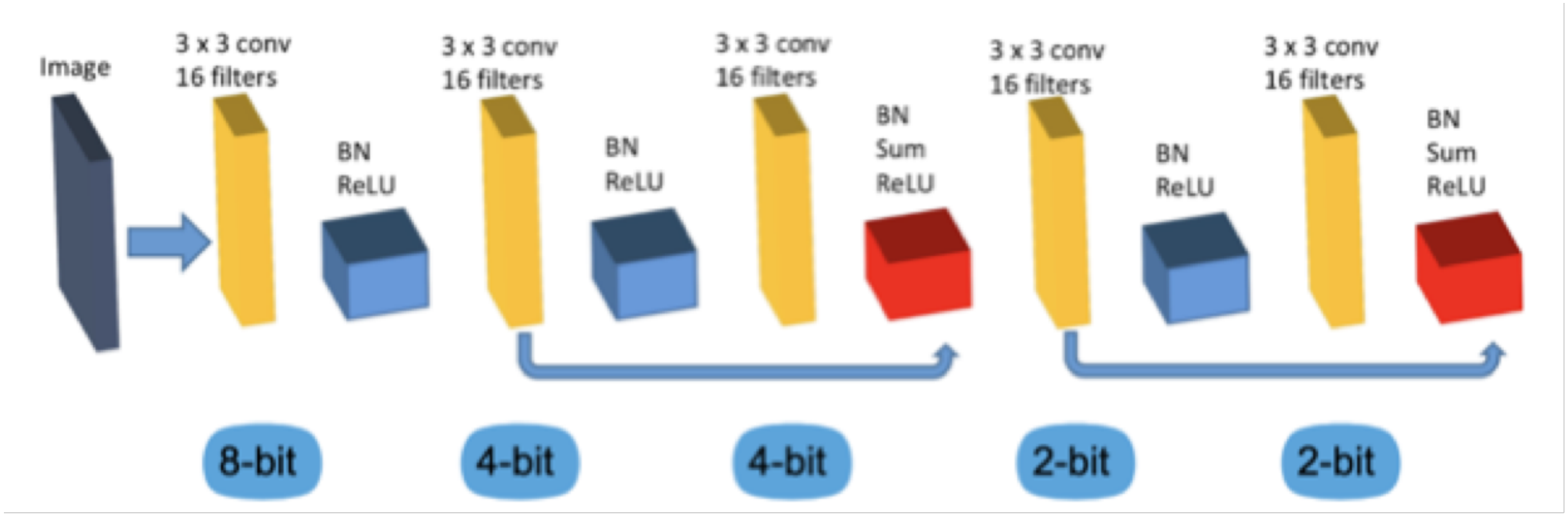

Which mixed-precision setting works better?

# Hessian AWare Quantization

Only quantize layers to ultra-low precision that have small Hessian spectrum

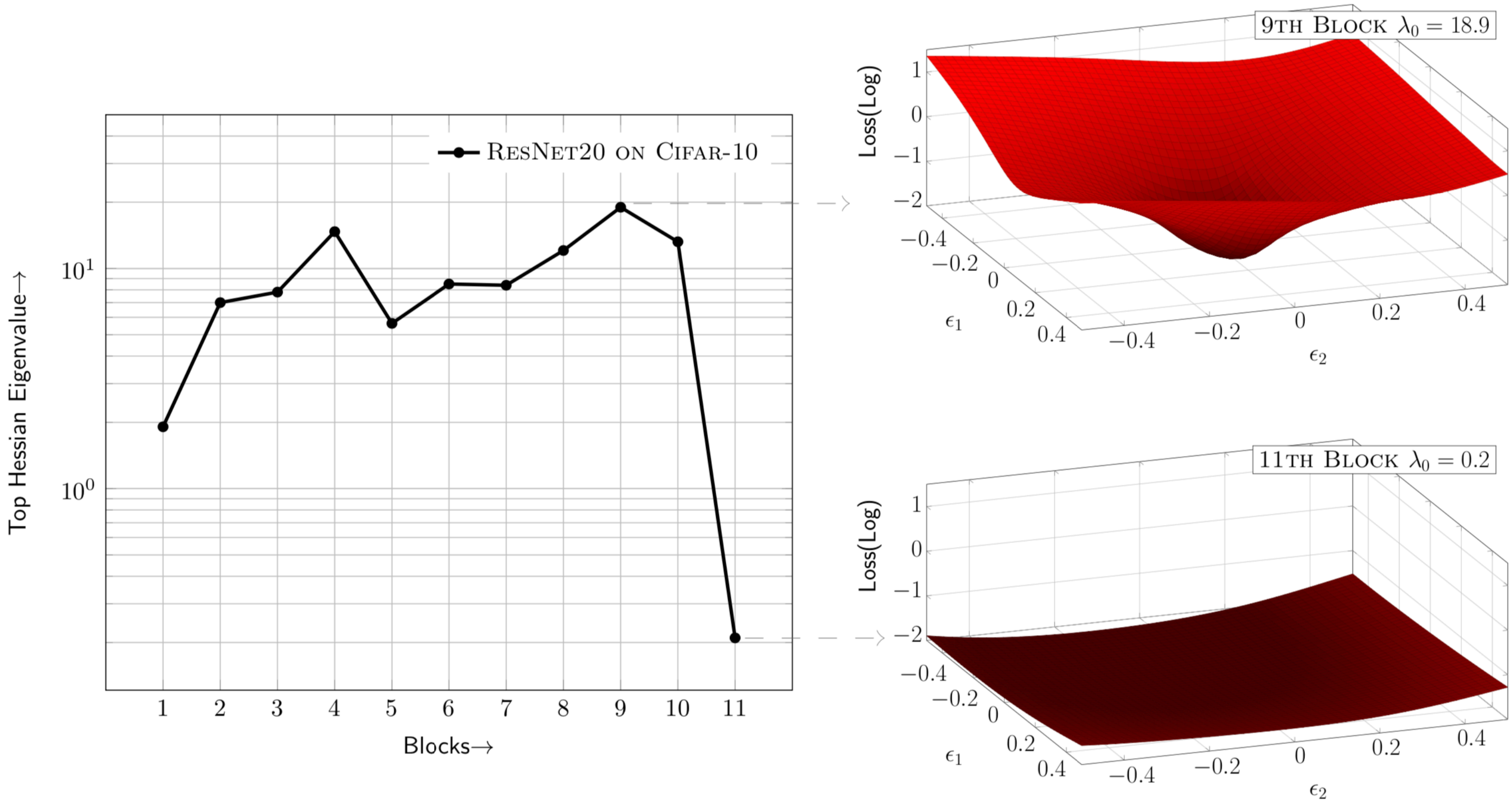


# Hessian AWare Quantization

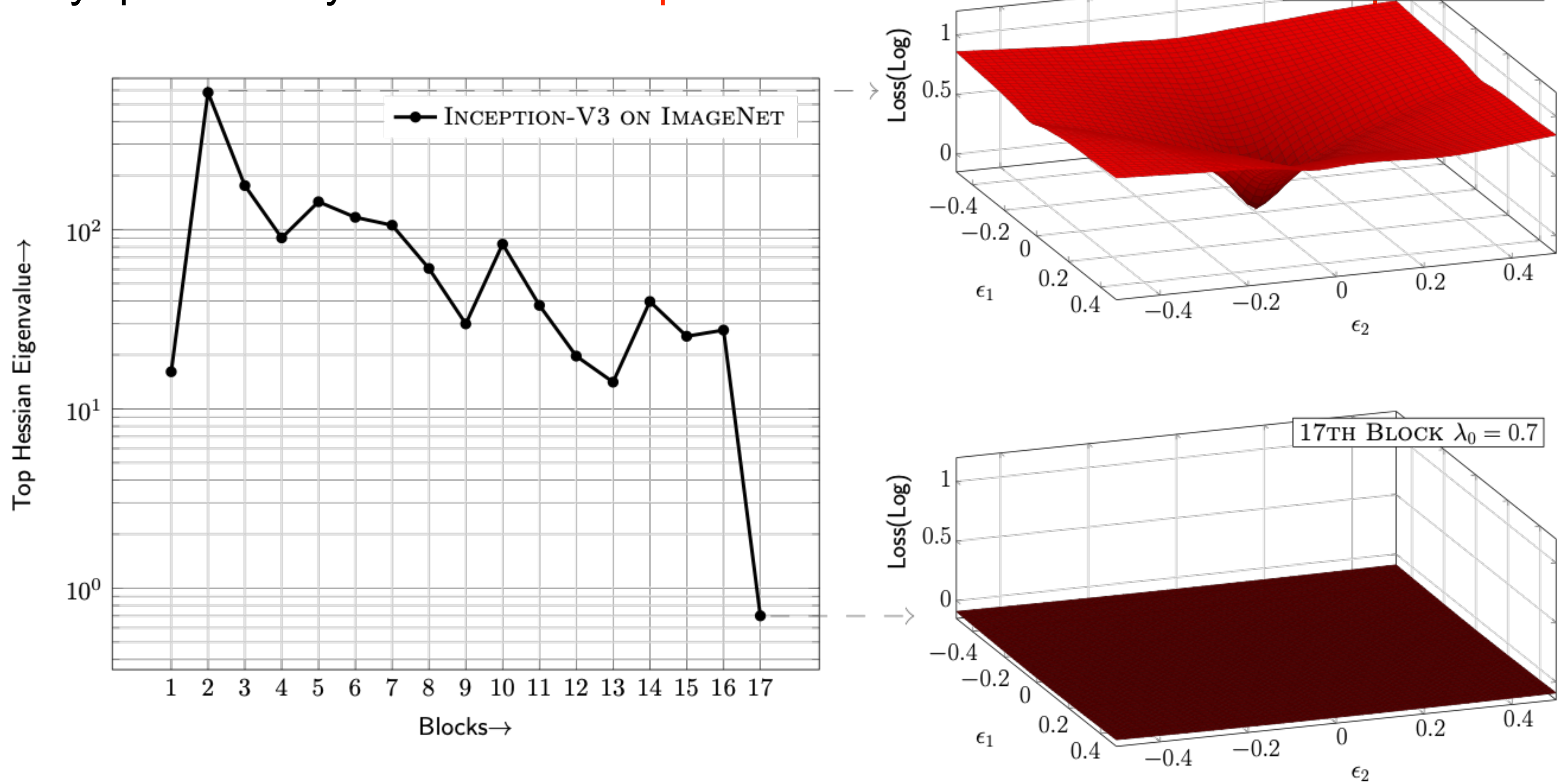
- Only quantize layers to ultra-low precision that have small Hessian spectrum




Z. Dong, Z. Yao, A. Gholami, M. Mahoney, K. Keutzer, HAWQ: Hessian Aware Quantization of Neural Networks With Mixed Precision, ICCV'19 (Accepted)

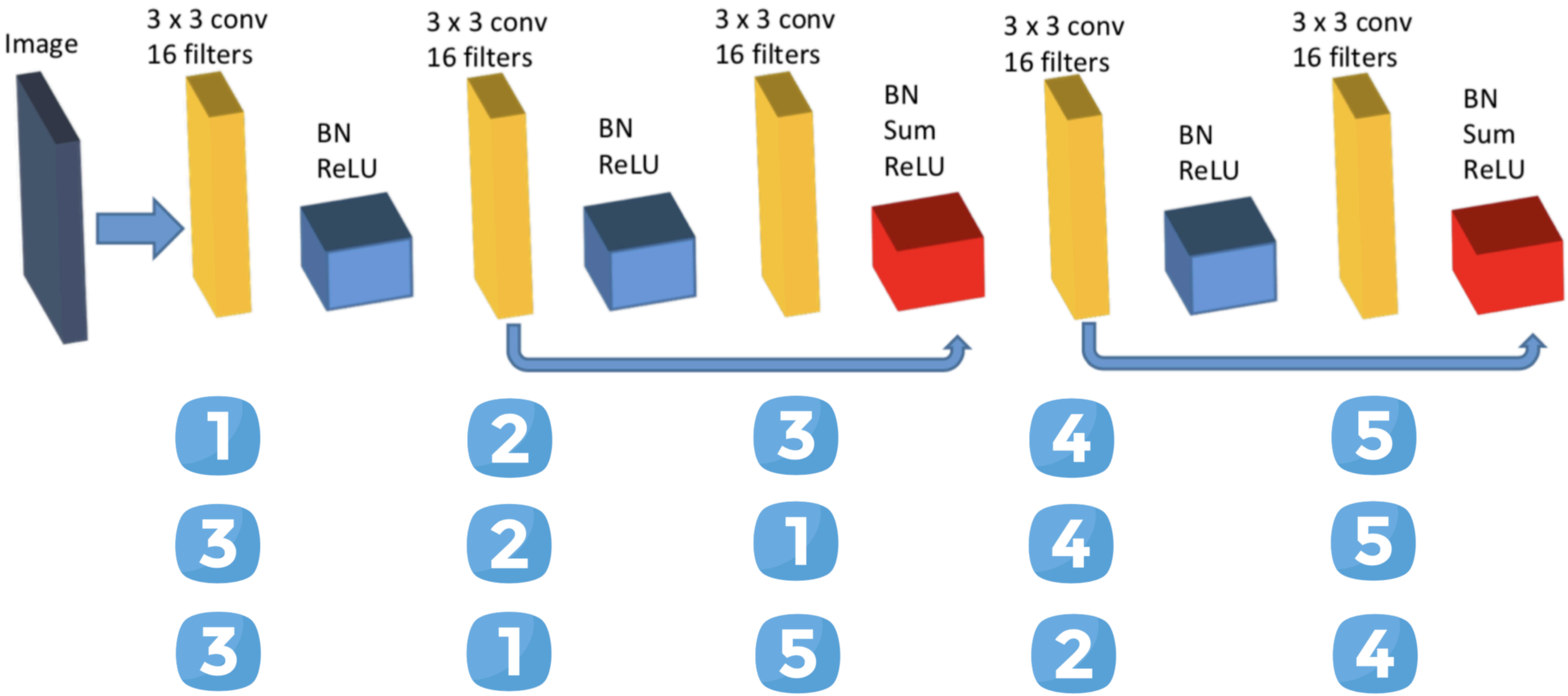
# Hessian AWare Quantization

Only quantize layers to ultra-low precision that have small Hessi




Z. Dong, Z. Yao, A. Gholami, M. Mahoney, K. Keutzer, HAWQ: Hessian Aware Quantization of Neural Networks With Mixed Precision, ICCV'19 (Accepted)

# Layer-wise Quantization: Factorial Search Space




Z. Dong, Z. Yao, A. Gholami, M. Mahoney, K. Keutzer, HAWQ: Hessian Aware Quantization of Neural Networks With Mixed Precision, ICCV'19 (Accepted)

# HAWQ Result- ResNet50 on ImageNet

Precisions for all layers as well as block-wise fine-tuning orders are 100%

auto

| Method | w-bits | a-bits | Top-1 | W-Comp | Size(MB) |
|---|---|---|---|---|---|
| Baseline | 32 | 32 | 77.39 | 1.00× | 97.8 |
| Dorefa [7] | 2 | 2 | 67.10 | 16.00× | 6.11 |
| Dorefa [7] | 3 | 3 | 69.90 | 10.67× | 9.17 |
| PACT [8] | 2 | 2 | 72.20 | 16.00× | 6.11 |
| PACT [8] | 3 | 3 | 75.30 | 10.67× | 9.17 |
| LQ-Nets [9] | 3 | 3 | 74.20 | 10.67× | 9.17 |
| Deep Comp. [22] | 3 | MP | 75.10 | 10.41× | 9.36 |
| HAQ [13] | MP | MP | 75.30 | 10.57× | 9.22 |
| HAWQ [1] | 2 MP | 4 MP | **75.48** | 12.28× | 7.96 |
| HAWQ-V2 | 2 MP | 4 MP | **75.56** | 12.25× | **7.98** |

Z. Dong, Z. Yao, A. Gholami, M. Mahoney, K. Keutzer, HAWQ: Hessian Aware Quantization of Neural Networks With Mixed Precision, ICCV'19 (Accepted)

# HAWQ Result- RetinaNet on CoCo

Precisions for all layers as well as block-wise fine-tuning orders are 100% automatically selected.

| Method | w-bits | a-bits | mAP | W-Comp | Size(MB) |
|---|---|---|---|---|---|
| Baseline | 32 | 32 | 35.6 | 1.00× | 145 |
| FQN [20] | 4 | 4 | 32.5 | 8× | 18.13 |
| HAWQ-V2 | 4 MP | 4 | **33.5** | 8× | 18.13 |

Z. Dong, Z. Yao, A. Gholami, M. Mahoney, K. Keutzer, HAWQ: Hessian Aware Quantization of Neural Networks With Mixed Precision, ICCV'19 (Accepted)

# HAWQ Result- BERT on CoNNL

| Method | w-bits | e-bits | $F_1$ | Size | Size-w/o-e |
|---|---|---|---|---|---|
| Baseline | 32 | 32 | 95.00 | 410.9 | 324.5 |
| Q-BERT | 8 | 8 | 94.79 | 102.8 | 81.2 |
| DirectQ | 4 | 8 | 89.86 | 62.2 | 40.6 |
| Q-BERT | 4 | 8 | **94.90** | 62.2 | 40.6 |
| DirectQ | 3 | 8 | 84.92 | 52.1 | 30.5 |
| Q-BERT | 3 | 8 | **94.78** | 52.1 | 30.5 |
| Q-BERT$_{\text{MP}}$ | 2/4 $_{\text{MP}}$ | 8 | **94.55** | 52.1 | 30.5 |
| DirectQ | 2 | 8 | 54.50 | 42.0 | 20.4 |
| Q-BERT | 2 | 8 | **91.06** | 42.0 | 20.4 |
| Q-BERT$_{\text{MP}}$ | 2/3 $_{\text{MP}}$ | 8 | **94.37** | **45.0** | **23.4** |

S. Sheng, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. Mahoney, K. Keutzer, Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT

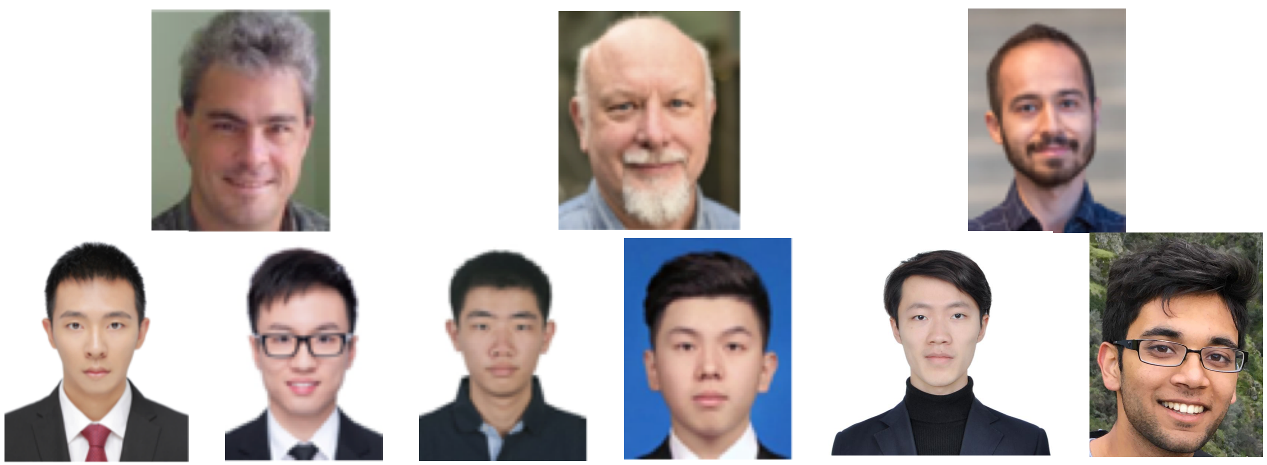
# Outline

- Background
- Efficient Deep Learning Training
- Efficient Deep Learning Inference
- Conclusions

## Conclusions

- Second order information of deep neural network can be computed by RandNLA and used for:
  - Improvements: speed of neural network training process
  - Useful information: Neural network quantization for inference

# Thank You



Berkeley
UNIVERSITY OF CALIFORNIA

riselab
UC Berkeley