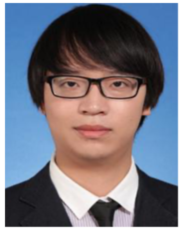
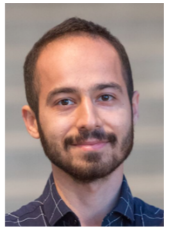
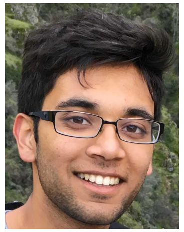
ABSA and Beyond!

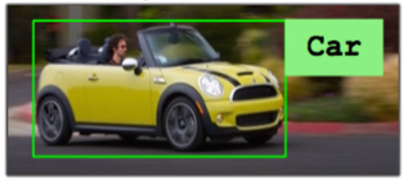
Zhewei Yao, Amir Gholami, Daiyaan Arfeen, Richard Liaw

Joseph Gonzalez, Michael Mahoney, Kurt Keutzer

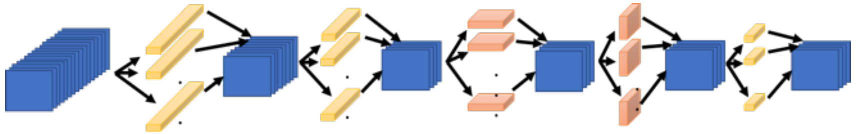
An Integrated Approach to DNN Design Has Four Key Aspects

Large Scale Training



Aggregating training data

Finding the right Deep Neural Network model



Efficiently implementing the DNN on embedded HW / co-design DNN accelerators

NN Through the Lens of the Hessian

- Development of PyHessian library [NeurIPS’18]
  - Allows fast computation of Hessian eigenvalues for DNNs
- HAWQ: Hessian AWare Quantization [ICCV’19, arxiv:1909.05840,NeurIPS’19]:
  - State-of-the-art quantization for Image Classification and NLP
- ABSA: Adaptive Batch Size with Second Order Information [arxiv1810.01021]
  - Batch size is automatically changed based on loss landscape curvature
- Trust Region based adversarial attack [CVPR’19]
  - Second order based algorithm for fast adversarial attack computation (up to 40x speed up)

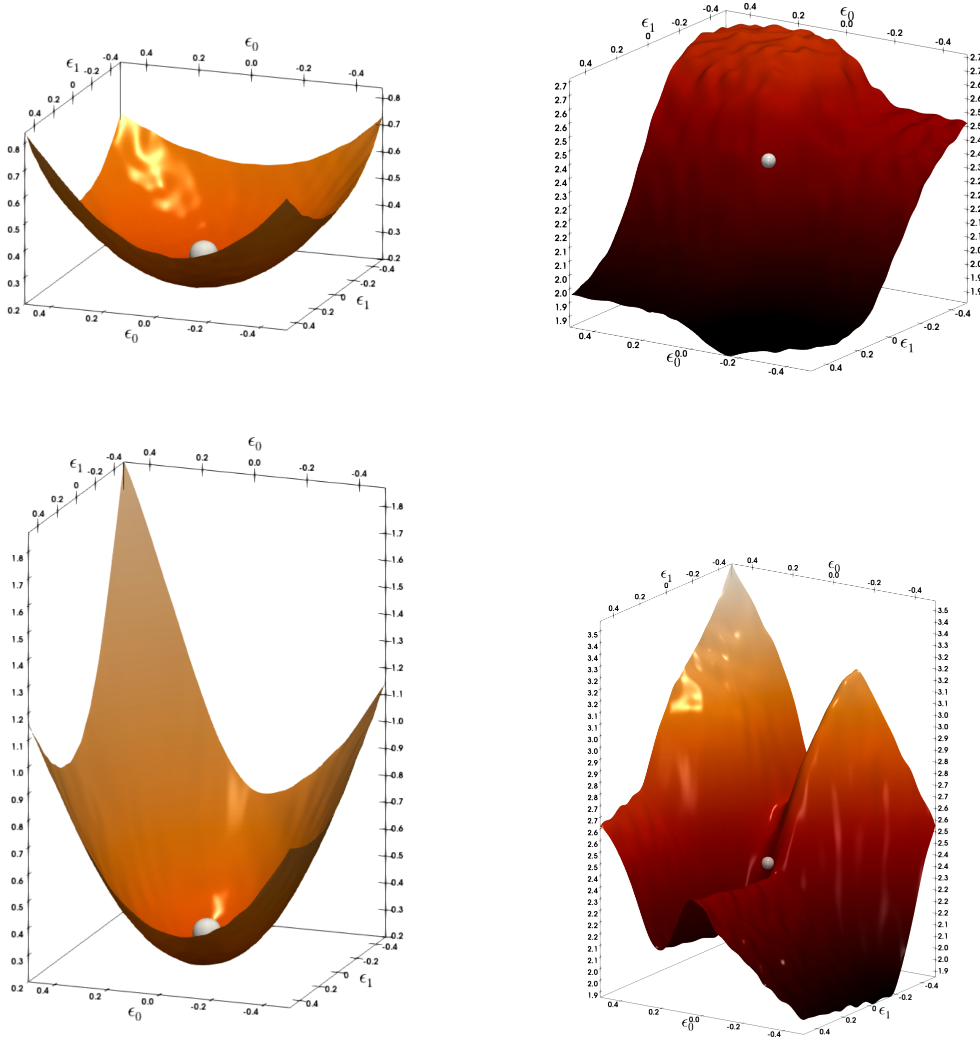
[NeurIPS’18] Yao, Z., Gholami, A., Lei, Q., Keutzer, K. and Mahoney, M.W., 2018. Hessian-based analysis of large batch training and robustness to adversaries. In Advances in Neural Information Processing Systems (pp. 4949-4959).

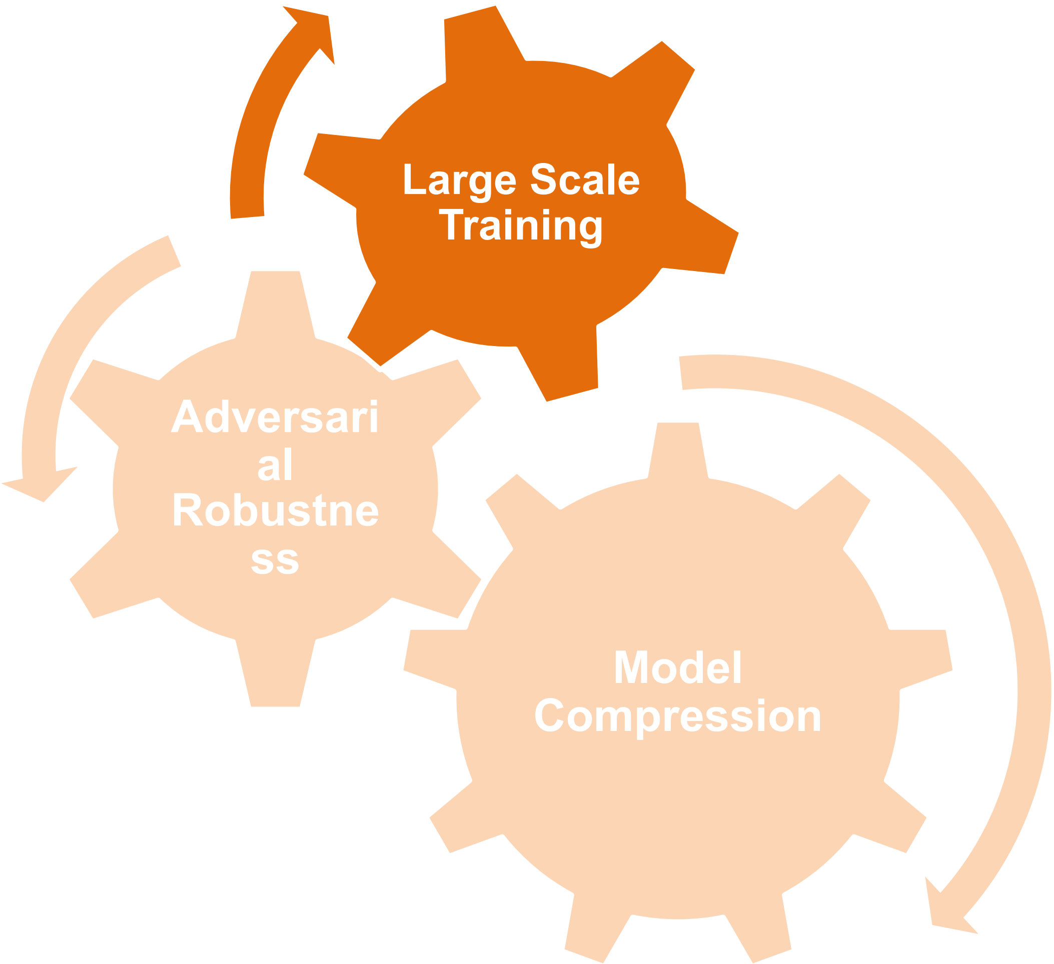
[NeurIPS’19] Z. Dong, Z. Yao, D. Arfeen, Y. Cai, A. Gholami, M. Mahoney, and K. Keutzer, Trace weighted hessian-aware quantization, **Spotlight** at NuerIPS'19 workshop on Beyond First-Order Optimization Methods in Machine Learning, 2019.

[ICCV’19] Dong, Z., Yao, Z., Gholami, A., Mahoney, M. and Keutzer, K., 2019. HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision. ICCV’19 (arXiv:1905.03696).

[CVPR’19] Yao, Z., Gholami, A., Xu, P., Keutzer, K. and Mahoney, M.W., 2019. Trust region based adversarial attack on neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 11350-11359).

Second Order Methods





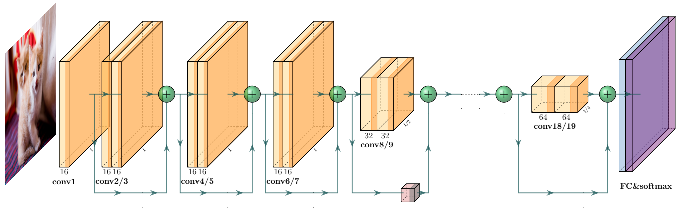
High Level Outline

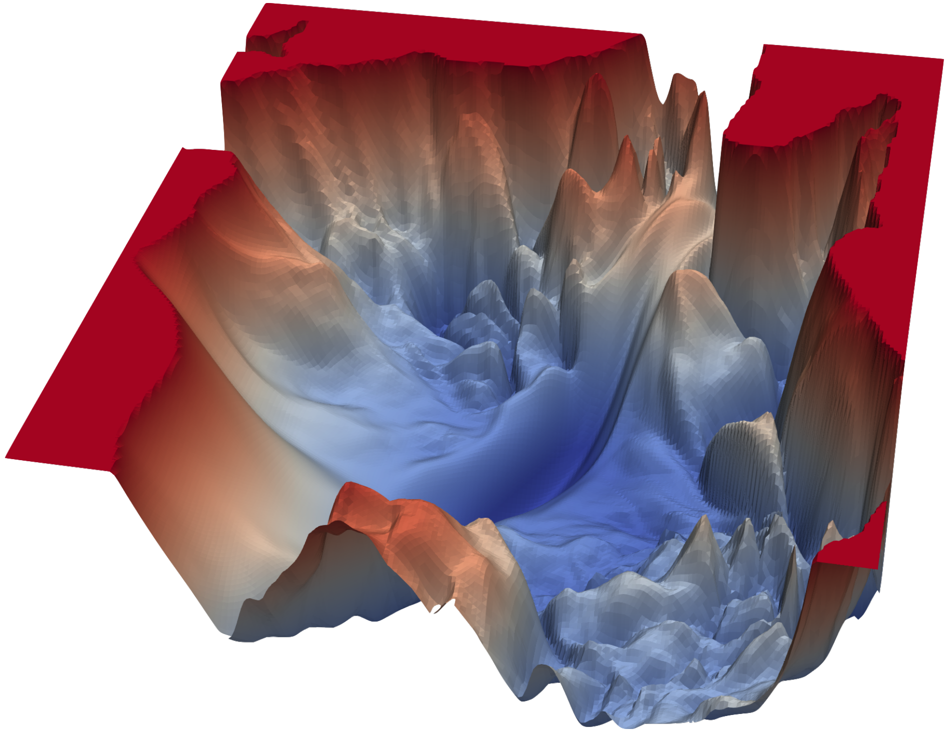
- DNN design requires training on large datasets
  - Time consuming
  - Need fast training -> parallelization -> large batch
- Large batch training does not work:
  - **Degrades accuracy**
  - **Poor robustness** to adversarial inputs
  - Existing solutions either do not work or require **extensive hyper-parameter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

Summary of Contributions

- Extensive analysis of mini-batch SGD behavior for deep neural networks
  - Saddle points, adversarial robustness, sharp/flat minima
- A new **Hessian based** large batch size training
  - ~~Degrades accuracy~~
  - ~~Existing solutions either do not work or require extensive hyper-parameter tuning~~
    - **Equal or better accuracy even without hyper-parameter tuning**
- Extensive testing of the proposed method on multiple datasets and multiple neural networks
  - Cifar-10/100, **ImageNet**, SVHN, Tiny ImageNet

Loss Landscape





$$\min_{w} \mathcal{J}(w) = \frac{1}{N}\sum_{i=1}^{N} cost(w, x_i)$$

$$w^1 = w^0 - \alpha \underbrace{\frac{\partial \mathcal{J}(w^0)}{\partial w}}_{\delta w}$$

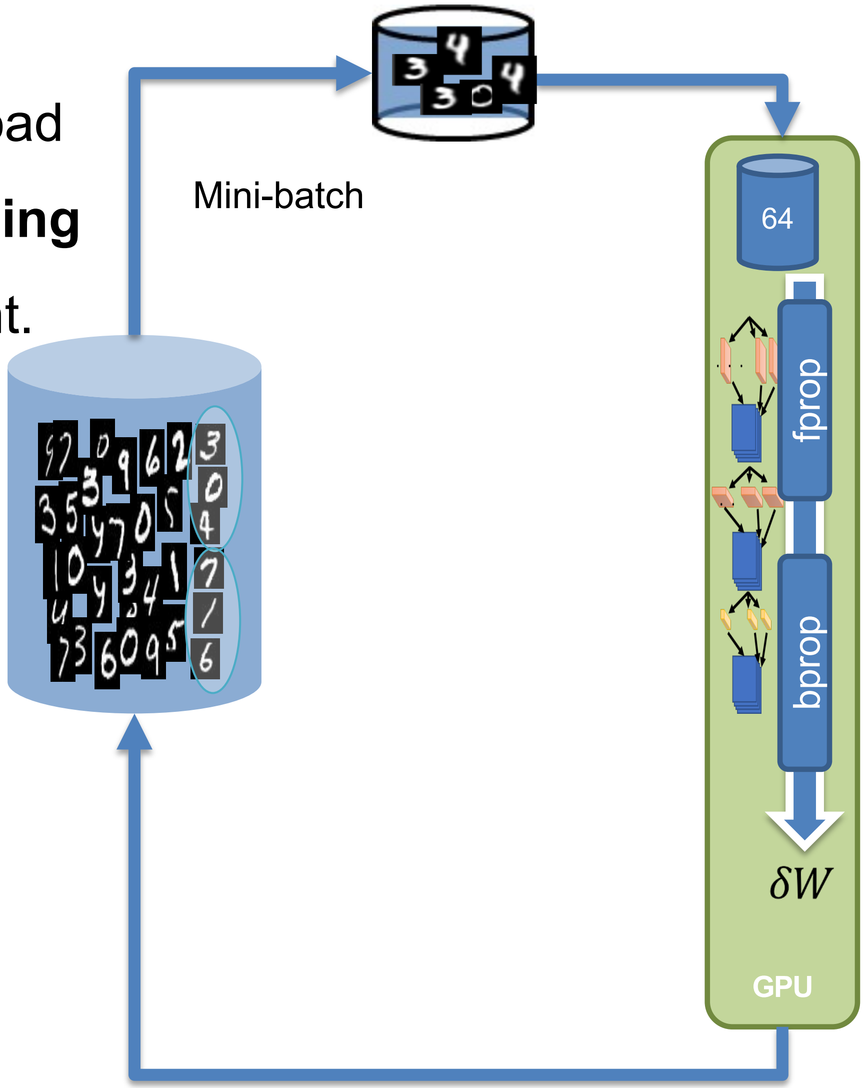
Learning rate

Two key elements:

- The computed gradient: the direction
- The learning rate: how big a step do we take?

Stochastic Gradient Descent (SGD)

In every iteration of SGD we load a **random mini-batch of training** data, and compute the gradient.

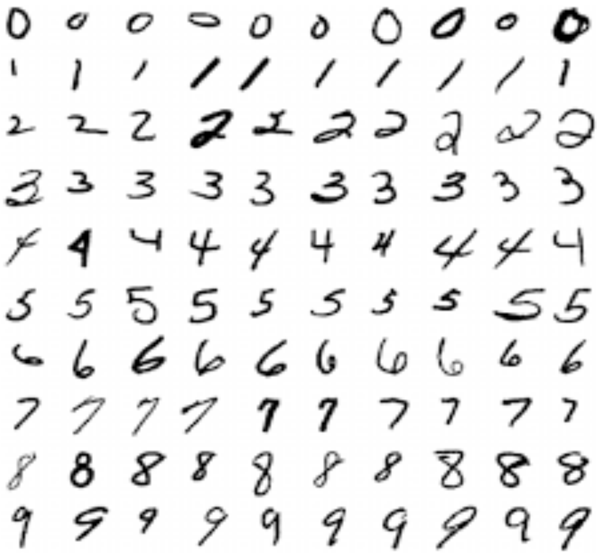


$$\min_w \mathcal{J}(w) = \frac{1}{N}\sum_{i=1}^{N} cost(w, x_i)$$

$$w^1 = w^0 - \alpha \underbrace{\frac{\partial \mathcal{J}(w^0)}{\partial w}}_{\delta w}$$

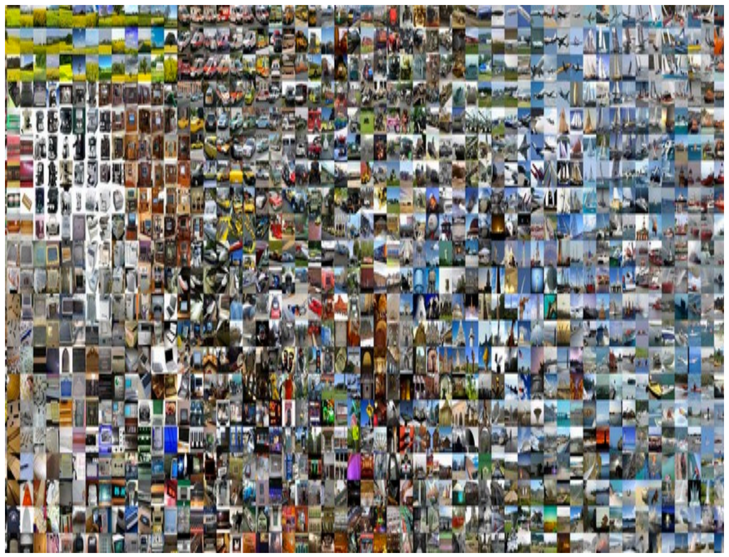
Larger Datasets Require Accelerated Training

MNIST [1]



70,000 images(28x28), 55MB
60,000 labeled training data for digit classification, 47MB

IMAGENET [2]

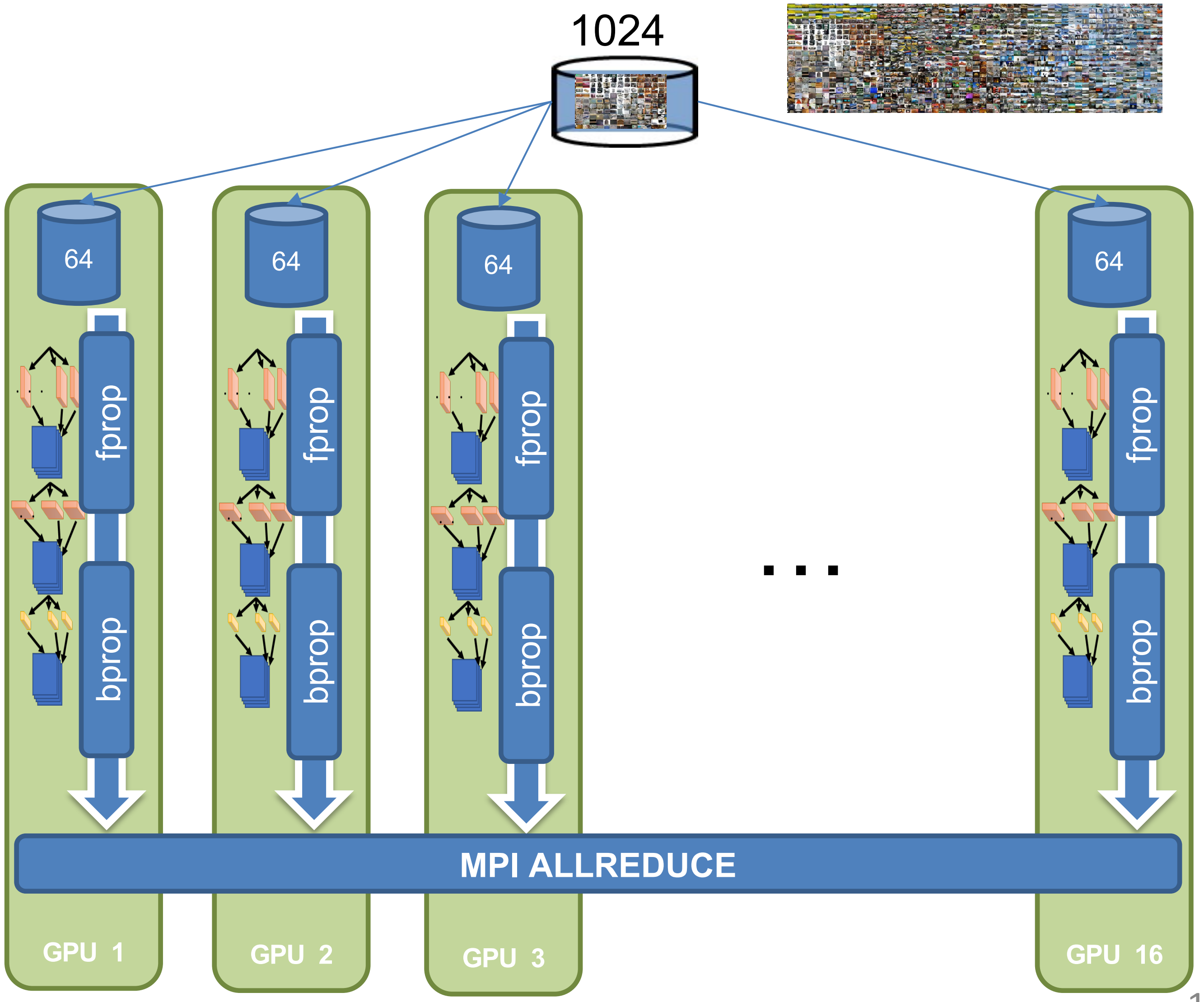


14M images(average:482x415), 8.5TB
1.2M labeled training data for object classification, 720GB

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition.” *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.
[2]Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575.*

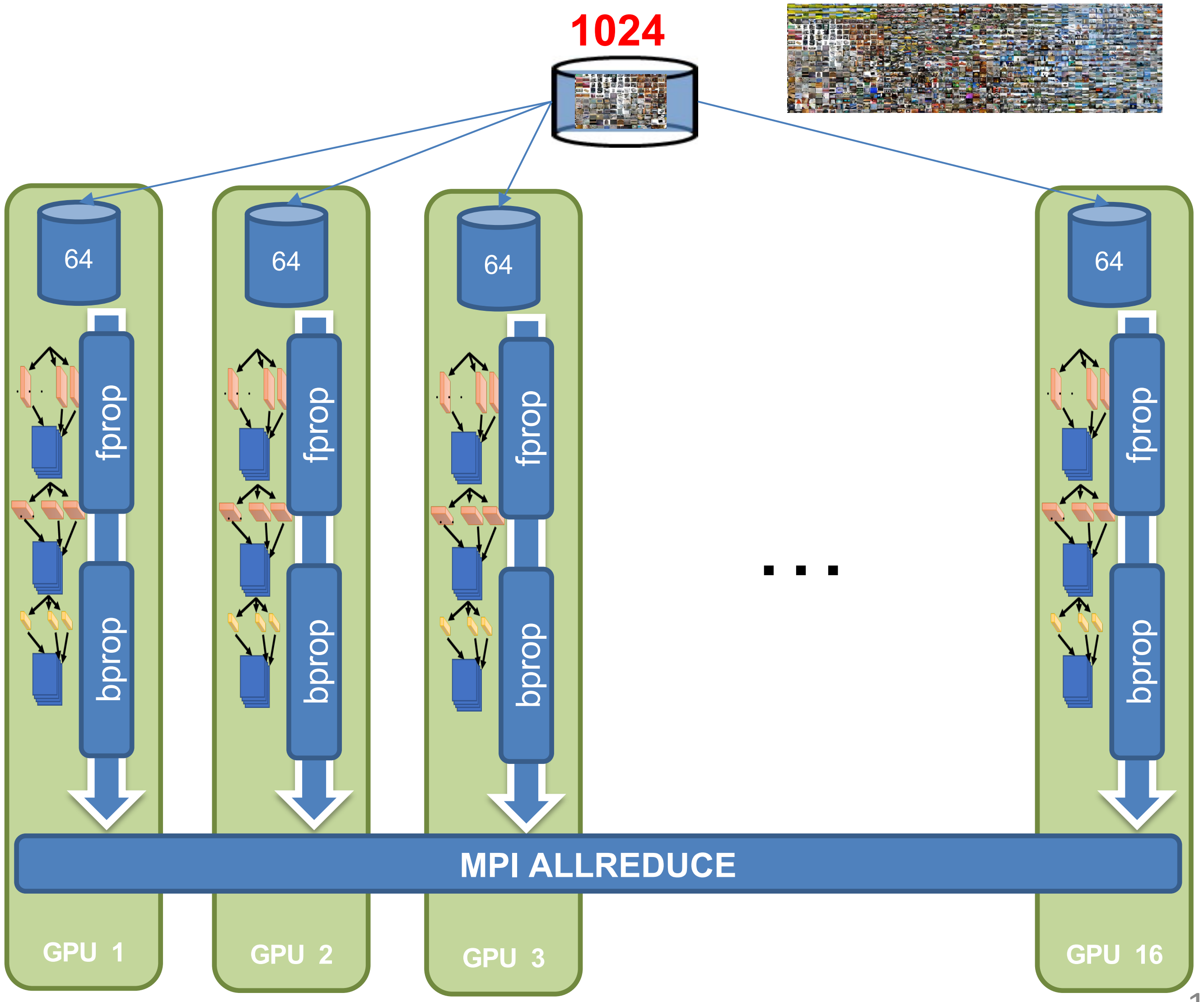
Best Opportunity: Data Parallelism

- Compute the entire model on each processor
- Distribute the SGD batch (aka mini-batch) evenly across each processor (aka per-processor batch):
  - 1024 batch distributed over 16 PEs: 64 PE batch
- Communicate gradient updates all-to-all



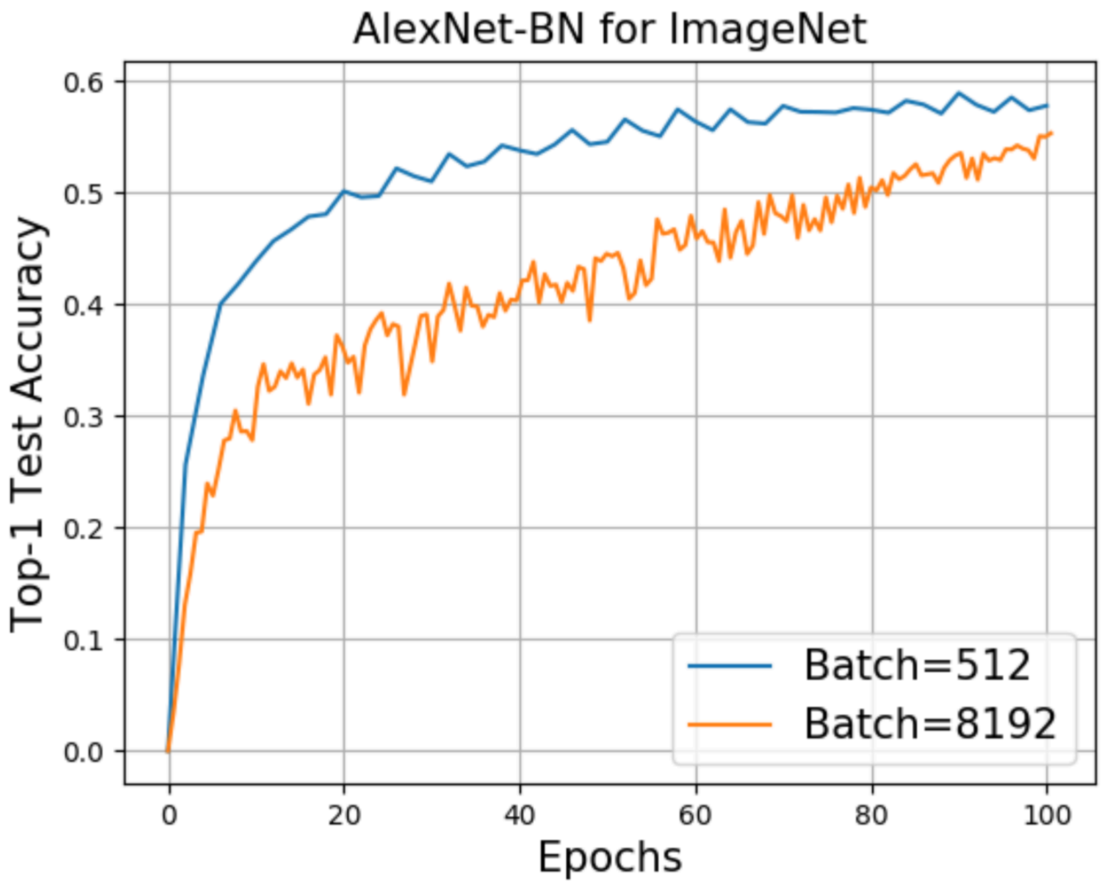
Scaling Synchronous SGD

If we want to keep scaling synchronous SGD then we have to keep increasing the batch size.



But … Large Batch Can Lead to Accuracy Degradation

- We need large batches, but large batches can easily lead to a degradation in accuracy

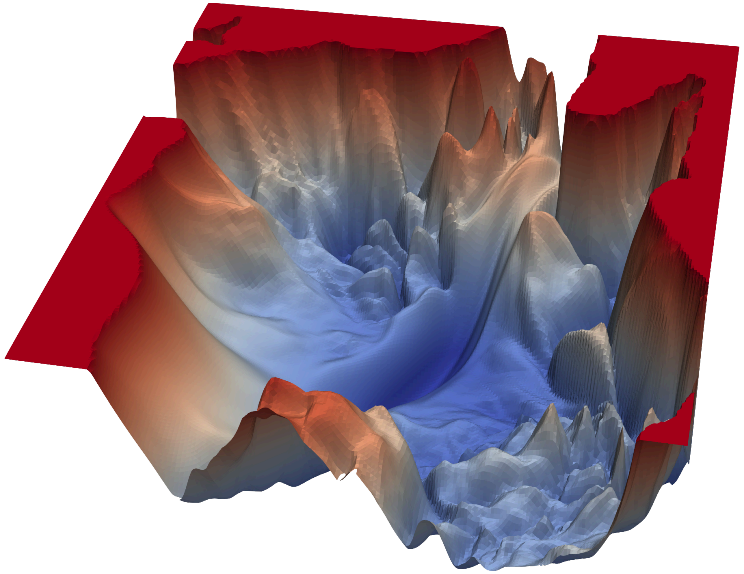


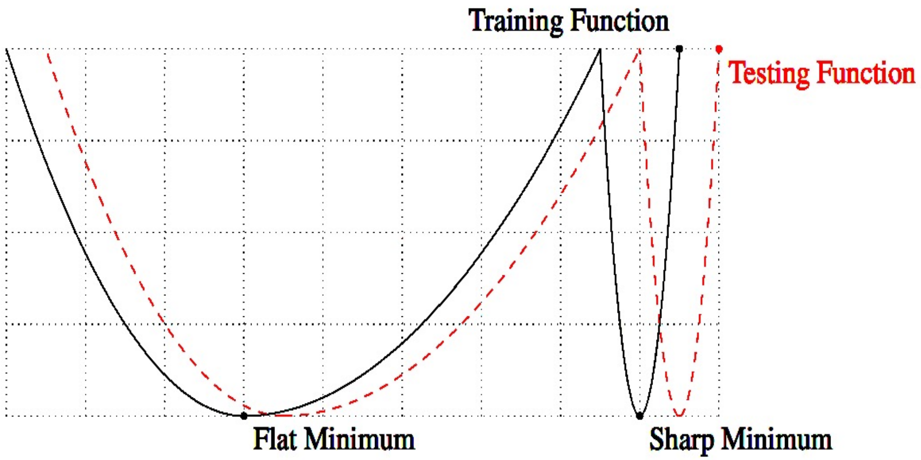
Ginsburg, Boris, Igor Gitman, and Yang You. "Large Batch Training of Convolutional Networks with Layer-wise Adaptive Rate Scaling." arxiv:1708.03888.

And ...May Result in Poor Generalization: Sharp vs Flat Minima

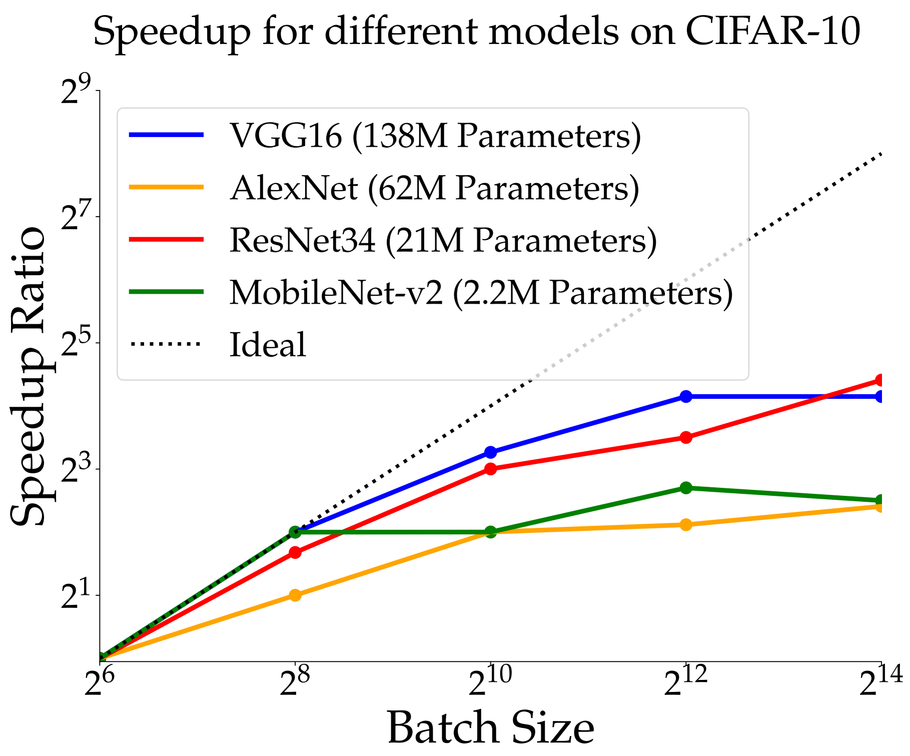
° Why large batch suffers from poor generalization performance?

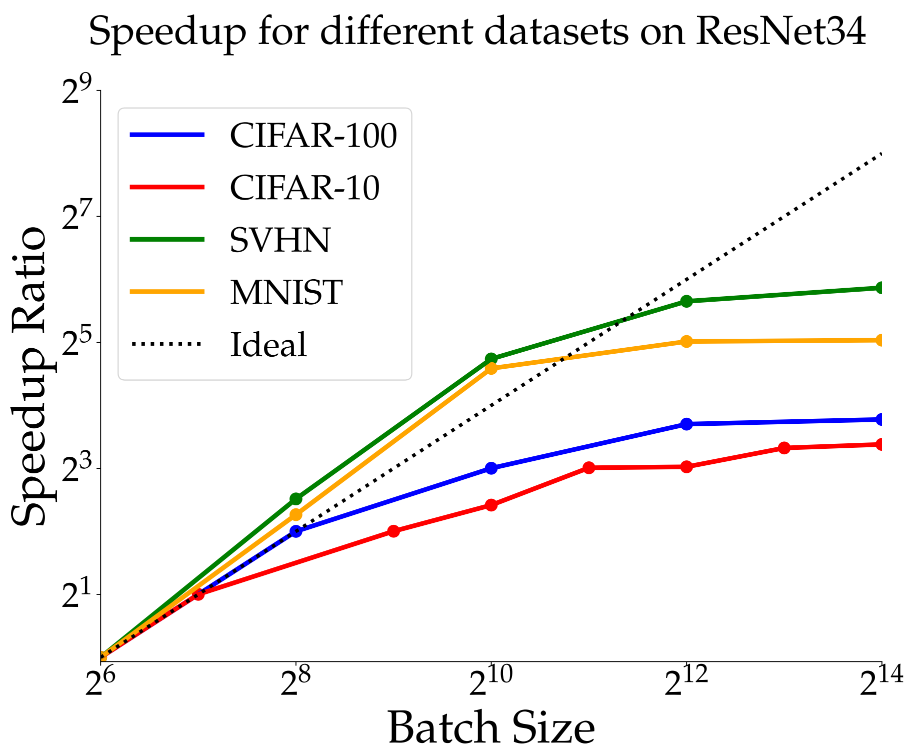
- A common belief is that large batch training gets attracted to "sharp minimas"
- Another theory is that large batch may get stuck in saddle points





Large batch size has diminishing returns after a certain point





Can we push this diminishing point further using **second-order information**?

Golmant N, Vemuri N, Yao Z, Feinberg V, Gholami A, Rothauge K, Mahoney MW, Gonzalez J. On the computational inefficiency of large batch sizes for stochastic gradient descent. arXiv:1811.12941. 2018.

What if We Had More Insight into the Loss Landscape?

$$W^{t+1} \leftarrow W^t - \alpha \cdot \frac{1}{b} \sum_{i=k+1}^{k+b} \nabla_W f_i(W^t, x)$$

Mini-batch: compute gradient using b samples

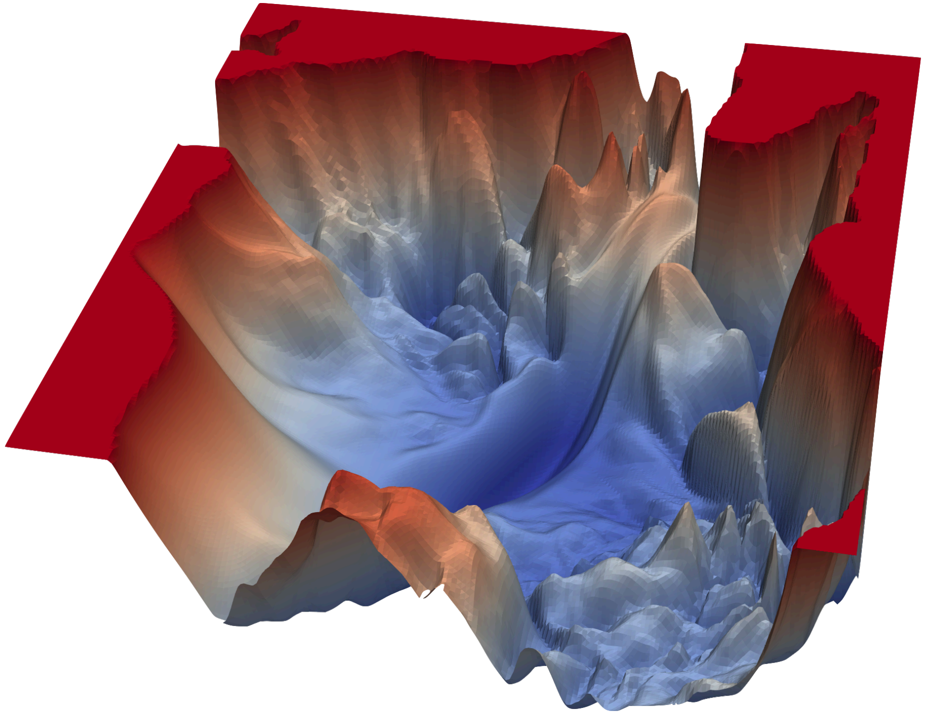
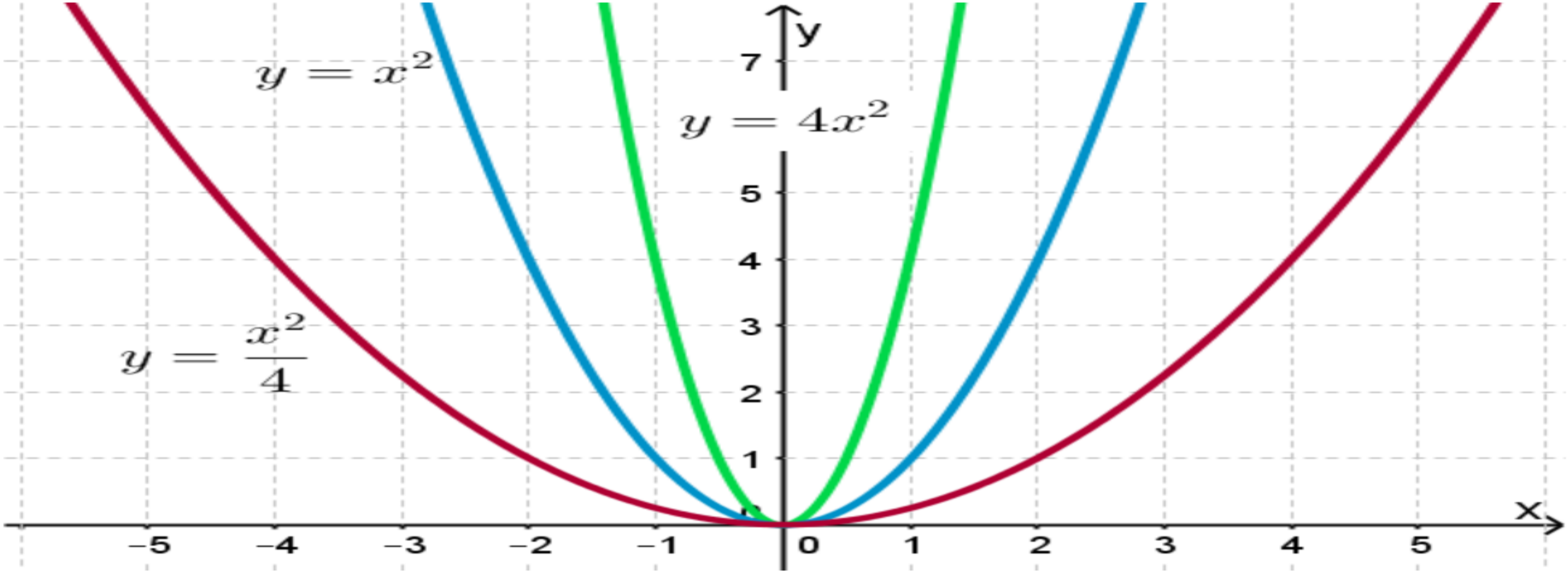


Image Source: Hao Li, UMD

The Second Derivative Tells us More About the Shape of the Function



- At the origin, the first derivative of $y = x^2$, $y = \frac{1}{4}x^2$, $y = 4x^2$ is all the same: 0
- But, the second derivatives give more information: 2, ½, and 8 respectively

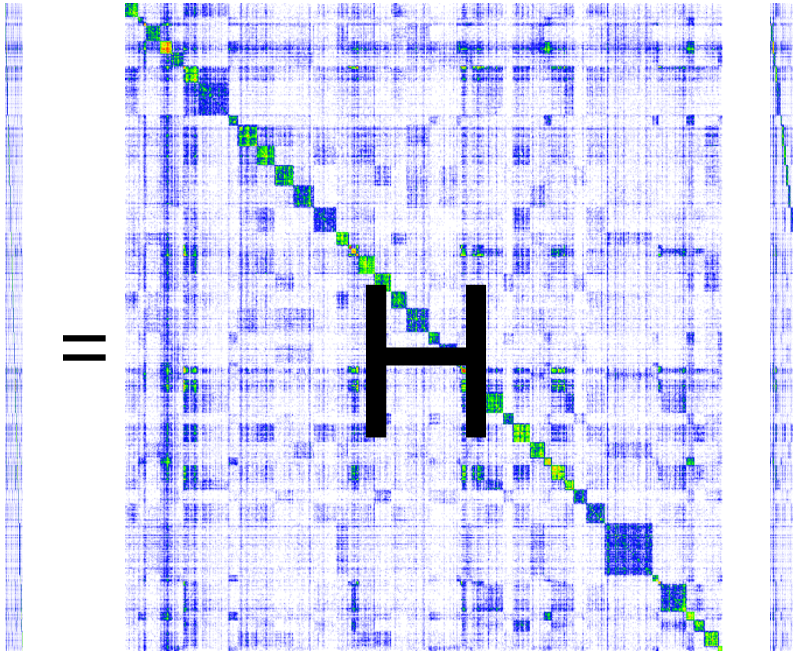
The Hessian Matrix

$$\mathbf{H}_{\mathrm{e}}(\mathbf{x}) = \begin{bmatrix} \partial^2_{x_1 x_1}\mathrm{e} & \cdots & \partial^2_{x_1 x_i}\mathrm{e} & \cdots & \partial^2_{x_1 x_N}\mathrm{e} \\ \vdots & \ddots & \vdots & ⋰ & \vdots \\ \partial^2_{x_i x_1}\mathrm{e} & \cdots & \partial^2_{x_i x_i}\mathrm{e} & \cdots & \partial^2_{x_i x_N}\mathrm{e} \\ \vdots & ⋰ & \vdots & \ddots & \vdots \\ \partial^2_{x_N x_1}\mathrm{e} & \cdots & \partial^2_{x_N x_i}\mathrm{e} & \cdots & \partial^2_{x_N x_N}\mathrm{e} \end{bmatrix}$$

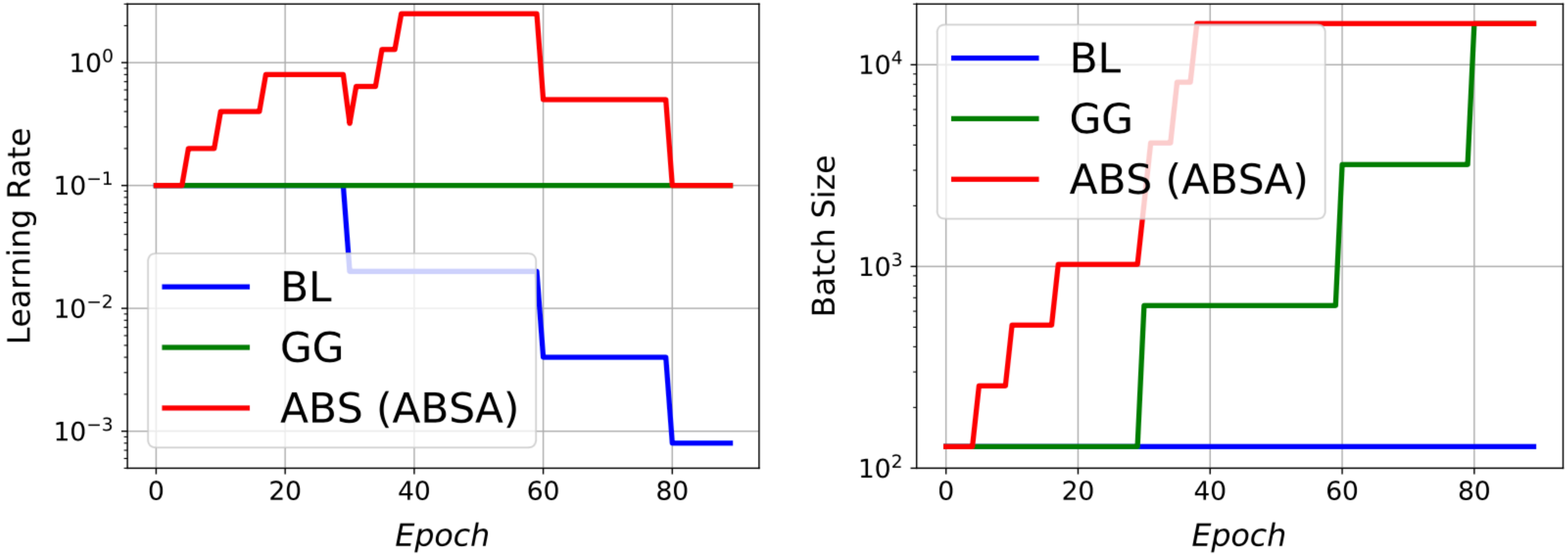
- The Hessian Matrix can give us comprehensive information about the Loss Landscape curvature
- Each Hessian matrix entry computes how fast gradient values are changing in different direction -> gradient of gradient

Fortunately, We Don't have to Compute the Full Hessian

- We only need Hessian eigenvalues and not the matrix itself
- Eigenvalue computation only needs multiplying H to random vectors (so called power iteration)
  - The matrix-vector multiplication can be done by a second gradient backpropagation
  - Therefore, **no need to compute the full Hessian**
  - **Only need top eigenvalues to estimate the sharpness of the loss landscape**



Adaptive Batch Size Using Hessian



- *Illustration of learning rate (a) and batch size (b) schedules of adaptive batch size as a function of training epochs*
- Well suited to the new era of "compute on demand" /serverless computing

GG: Smith, S.L., Kindermans, P.J., Ying, C. and Le, Q.V., 2017. Don't decay the learning rate, increase the batch size. ICLR'17, arXiv:1711.00489.
ABSA: Z. Yao, A. Gholami, D. Arfeen, R. Liaw, J. Gonzalez, K. Keutzer, M. Mahoney, Efficient Adaptive Batch Size Training of Neural Networks, (under review)

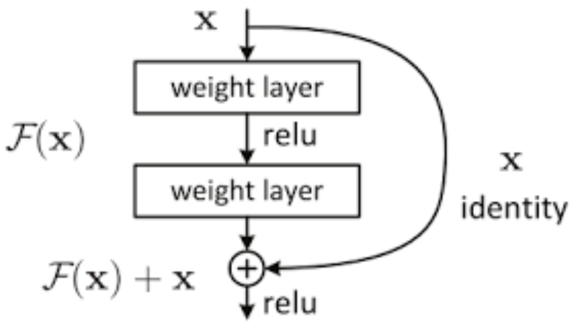
Cifar-10

- Cifar-10 has ten classes
  - ~5000 examples per class
  - Total 50,000 training images
  - 10,000 testing images

Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

ResNet-18 on Cifar-10

- Our proposed method (ABSA) achieves better performance



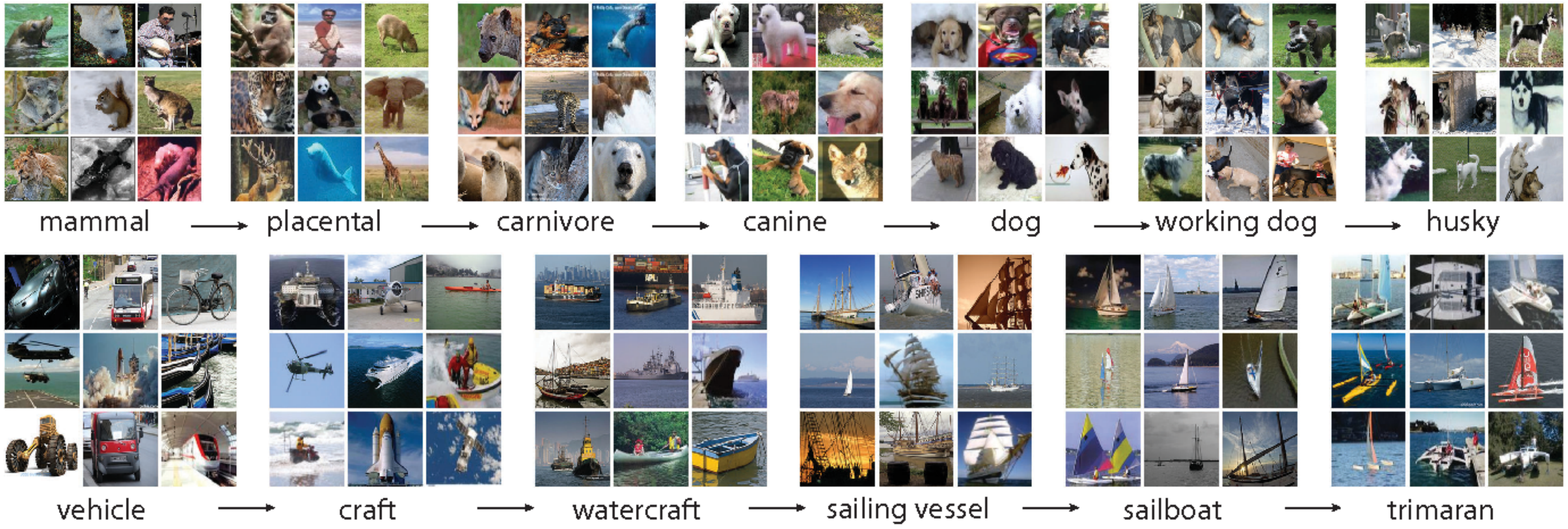
ResNet-18 on Cifar-10

	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: S. Samuel L., P. Kindermans, and Q. V. Le. "Don't Decay the Learning Rate, Increase the Batch Size." ICLR 2017.
ABS/ABSA: Z. Yao, A. Gholami, D. Arfeen, R. Liaw, J. Gonzalez, K. Keutzer, M. Mahoney, Efficient Adaptive Batch Size Training of Neural Networks, (under review)

ImageNet

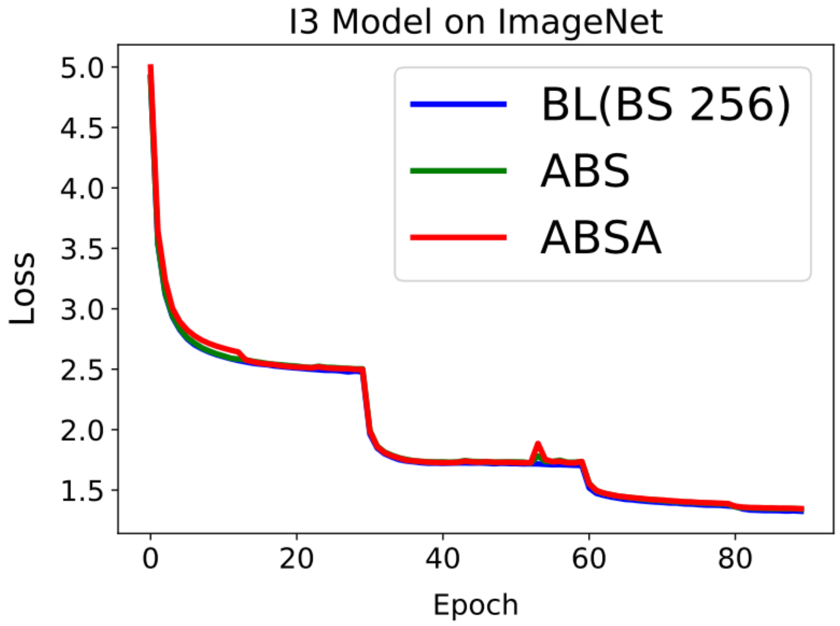
- ImageNet consists of 1000 classes
  - Total 1.2 million training images
  - 50,000 testing images

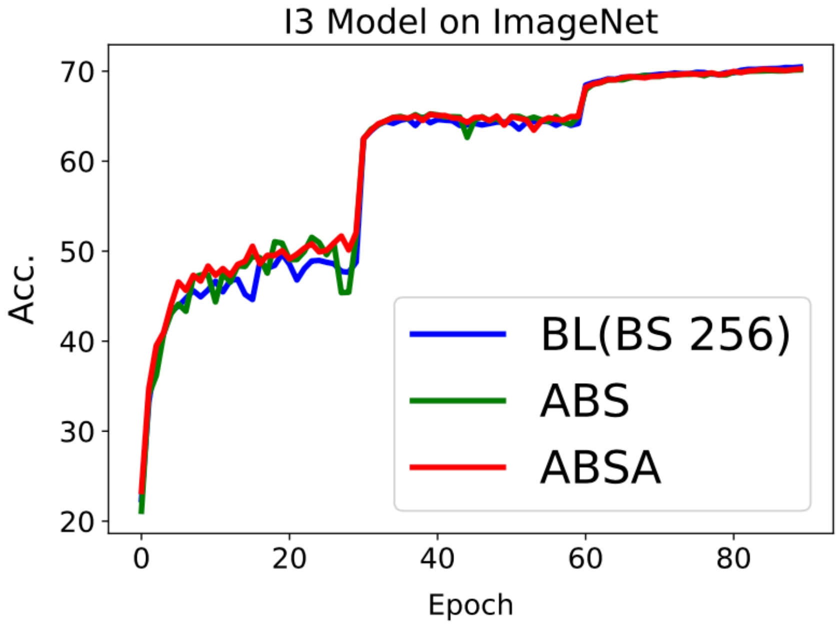


Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International Journal of Computer Vision 115.3 (2015): 211-252

ResNet18 on ImageNet

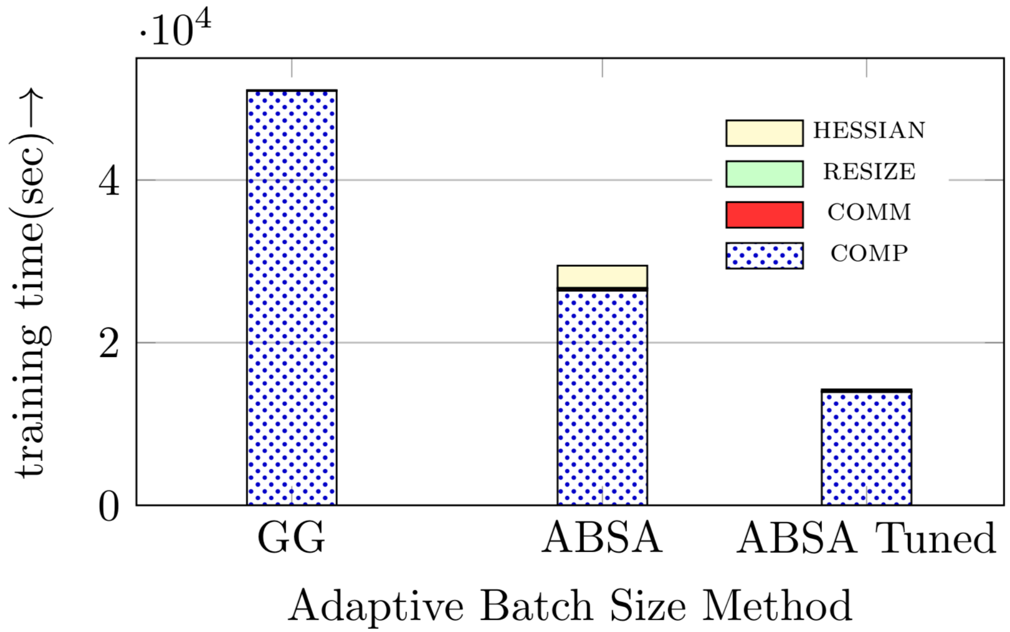
- Baseline:
  - **450k** SGD iterations, **70.4%** validation accuracy
- ABSA:
  - **66k** SGD iterations, **70.2%** validation accuracy
- GG would have required **166k** SGD iterations





Hessian Overhead is Less Than You Might Think

A common misconception is that Hessian computation is expensive. Below we show the speed up of ABSA algorithm as compared to Google's adaptive batch size method for ResNet18 training on ImageNet



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**

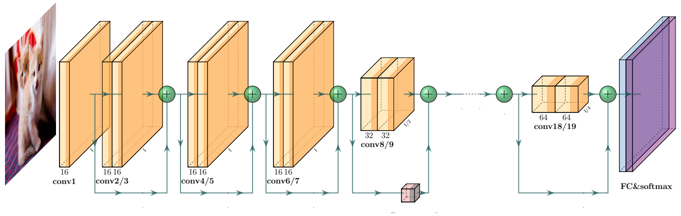
GG: Smith, S.L., Kindermans, P.J., Ying, C. and Le, Q.V., 2017. Don't decay the learning rate, increase the batch size. ICLR'17, arXiv:1711.00489.
ABSA: Z. Yao, A. Gholami, D. Arfeen, R. Liaw, J. Gonzalez, K. Keutzer, M. Mahoney, Efficient Adaptive Batch Size Training of Neural Networks, (under review)

Summary of Contributions

- Extensive analysis of mini-batch SGD behavior for deep neural networks
  - Saddle points, adversarial robustness, sharp/flat minima
- A new **Hessian based** large batch size training
  - ~~Degrades accuracy~~
  - ~~Existing solutions either do not work or require extensive hyper-parameter tuning~~
    - **Equal or better accuracy even without hyper-parameter tuning**
- Extensive testing of the proposed method on multiple datasets and multiple neural networks
  - Cifar-10/100, **ImageNet**, SVHN, Tiny ImageNet

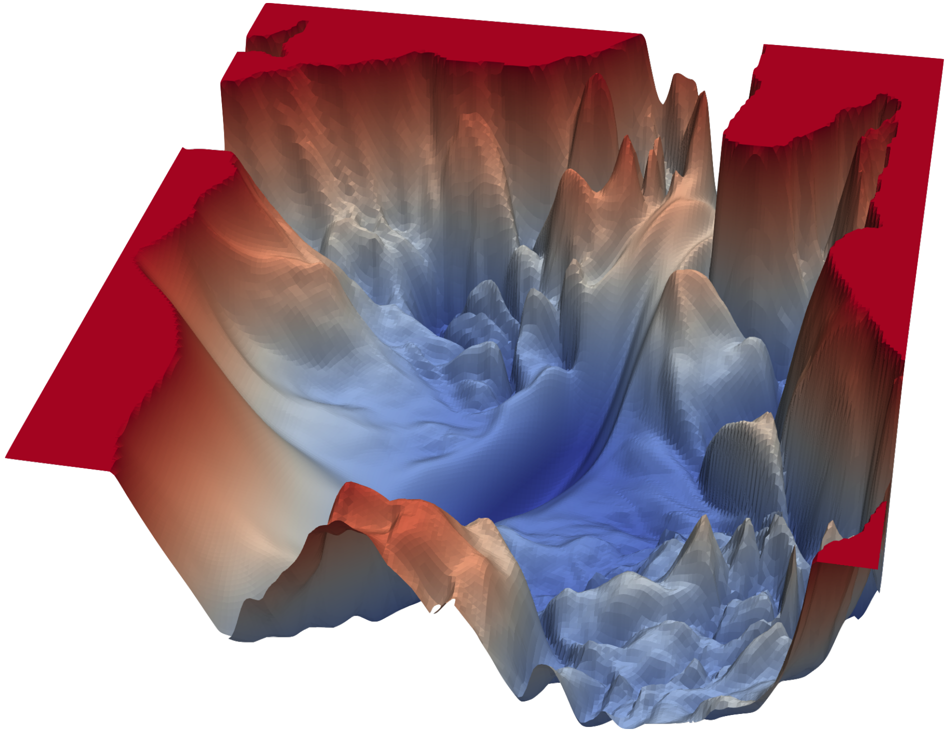
Future Work

Use second order information to guide SGD training



$$\min_{w} \mathcal{J}(w) = \frac{1}{N}\sum_{i=1}^{N} cost(w, x_i)$$

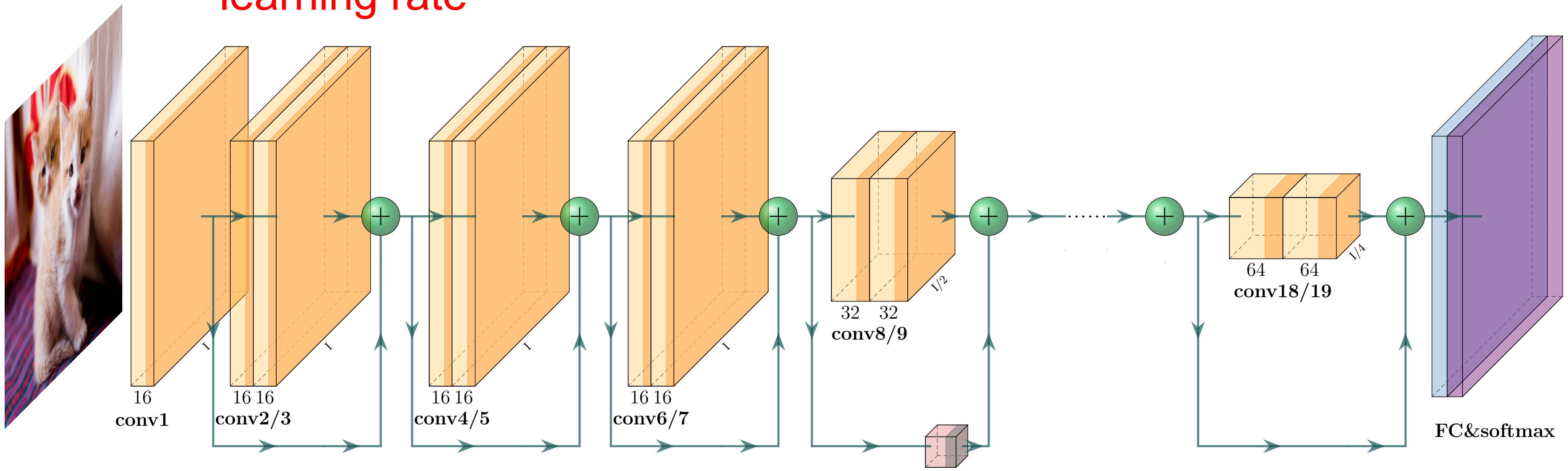
$$w^1 = w^0 - \alpha \frac{\partial \mathcal{J}(w^0)}{\partial w}$$

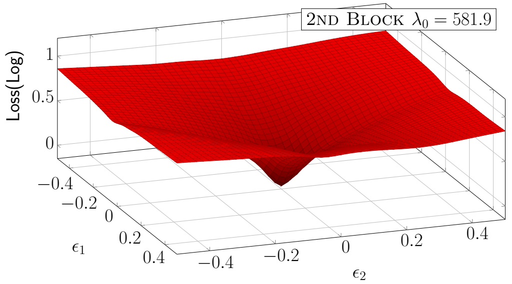


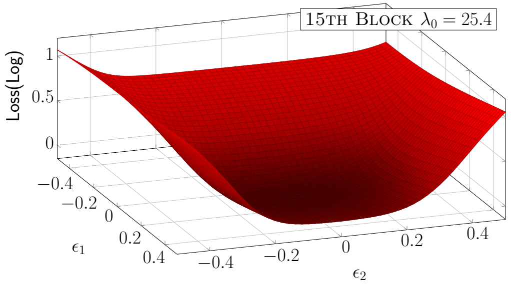
Gradient Descent

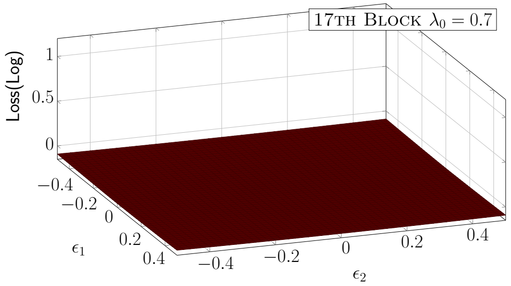
$$w^1 = w^0 - \alpha \underbrace{\frac{\partial \mathcal{J}(w^0)}{\partial w}}_{\delta w}$$

All layers have the same learning rate





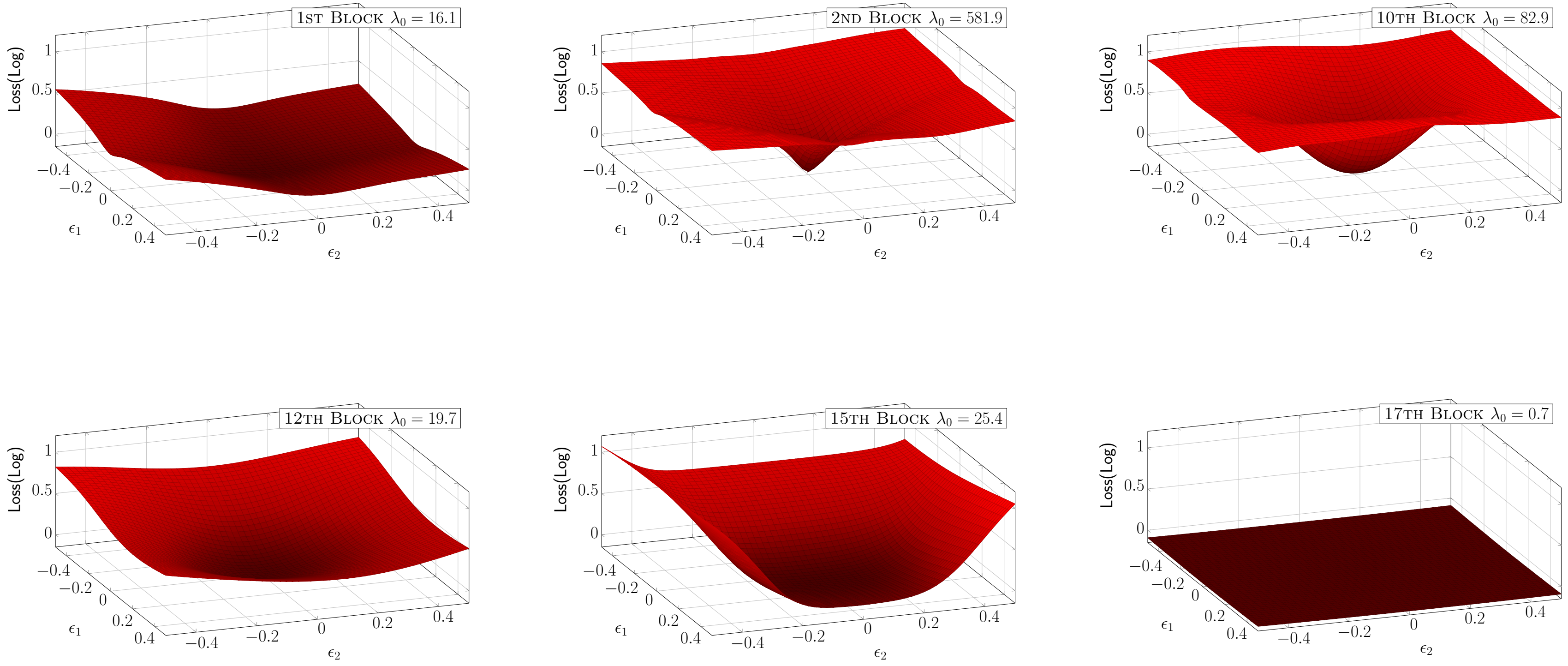




Future Work

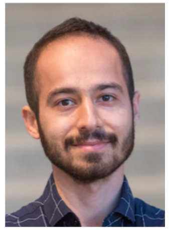
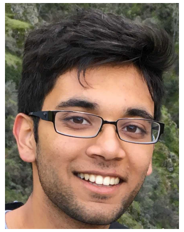
- Not all layers should have the same learning rate
  - AdaHessian
- **Mixed-Precision Training**
  - Compute Hessian at every epoch and adaptively change precision

$$w^1 = w^0 - \alpha \frac{\partial \mathcal{J}(w^0)}{\partial w}$$



BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Thanks for your Attention!

Berkeley
UNIVERSITY OF CALIFORNIA