Towards Fast and Efficient Representation Learning

Hao Li

PhD Dissertation Defense

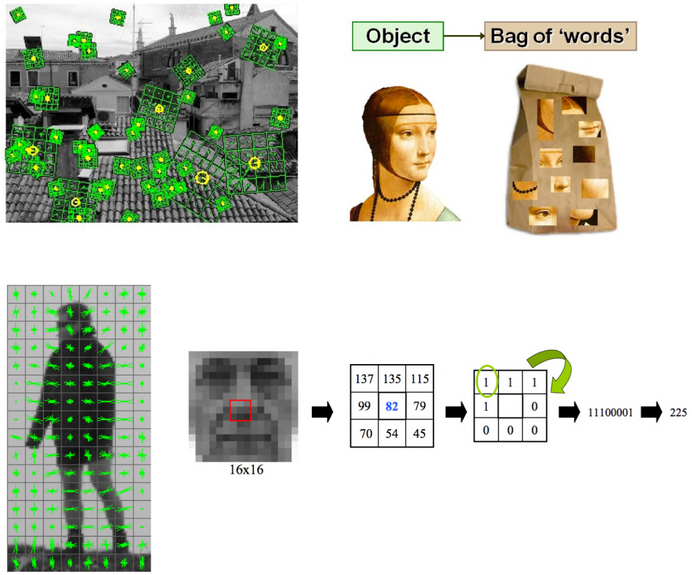
June 14, 2018

DEPARTMENT OF
COMPUTER SCIENCE

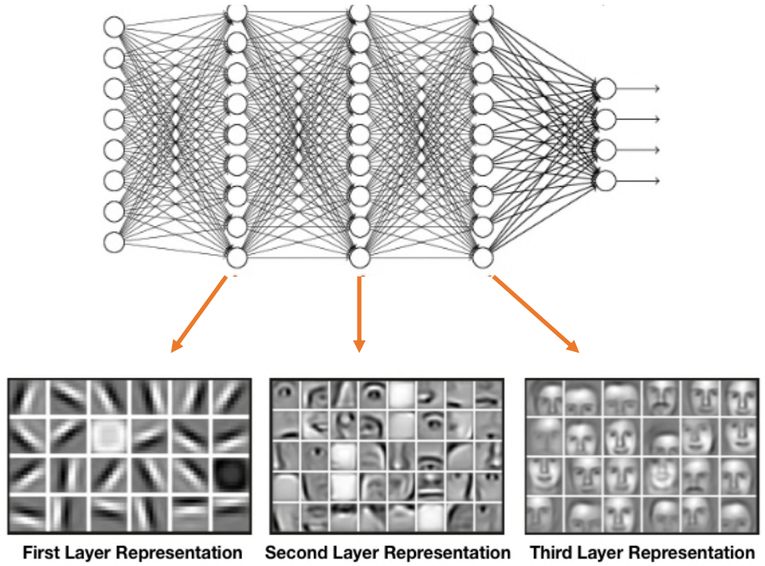
Representation Learning for Computer Vision

"Learning representations of the data that make it easier to extract useful information when building classifiers or other predictors." Bengio et al, PAMI'13

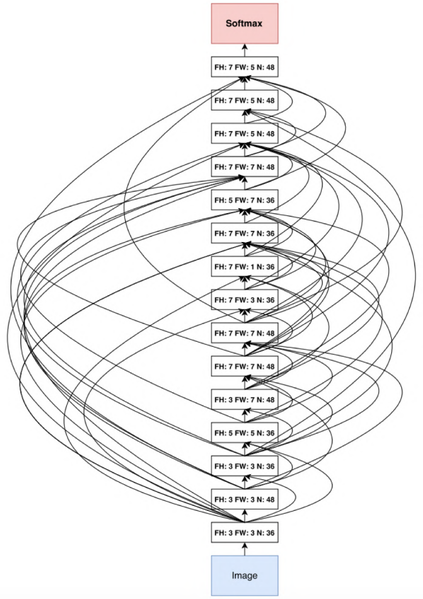
hand-designed features



hand-designed architectures



automaticly designed architectures

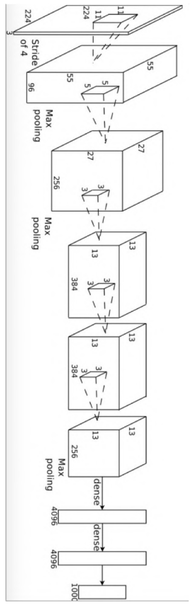


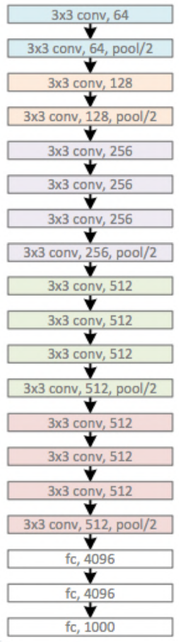
2009

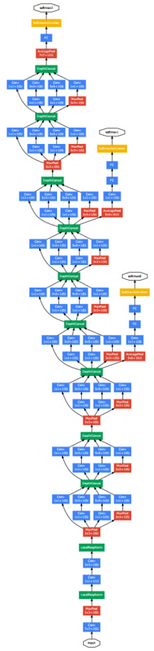
2012

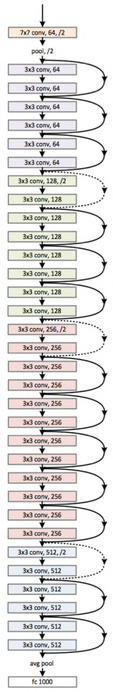
2018

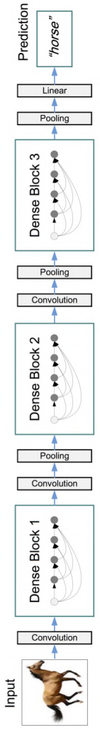
Evolution of Convolutional Nerual Networks

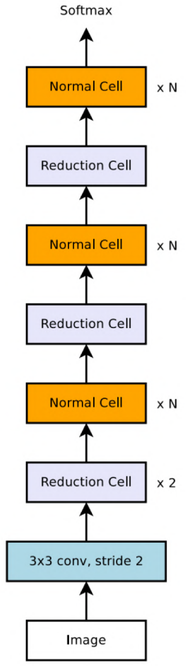












AlexNet
18.2% top-5
8 layers
60 M params
0.86 B FLOPs

VGG-19
7.3% top-5
19 layers
140 M params
19 B FLOPs

GoogLeNet
6.7% top-5
22 layers
5 M params
1.5B FLOPs

ResNet
5.7% top-5
152 layers
60 M params
11 B FLOPs

DenseNet
5.3% top-5
264 layers
32 M params
11 B FLOPs

NASNet
3.8% top-5
28 cells
88 M params
23 B FLOPs

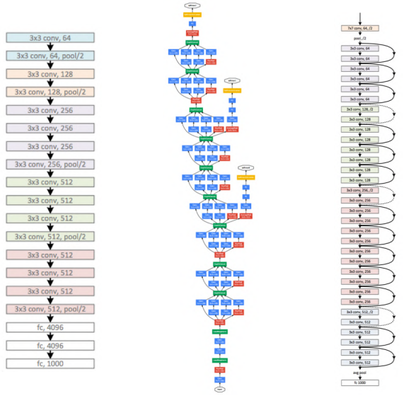
2012 2014 2016 2017 2018

Enabeling Factors and Challenges

Large-Scale Dataset

applications with
limited data

High-Capacity Models

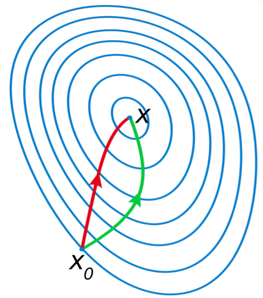


millions of parameters
billions of FLOPs/image
days or weeks of training

Faster Hardware

$$$$ electronic bills
hard to deploy on
embedded devices

Algorithm



limited understanding
lack of theory

Data Efficiency

Inference/Training Cost

Power Efficiency

Interpretation

Towards Fast and Efficient Representation Learning

Reducing the Inference Cost

- How to reduce the model size?
- How to reduce the number of operations for fast inference?

Reducing the Training Cost

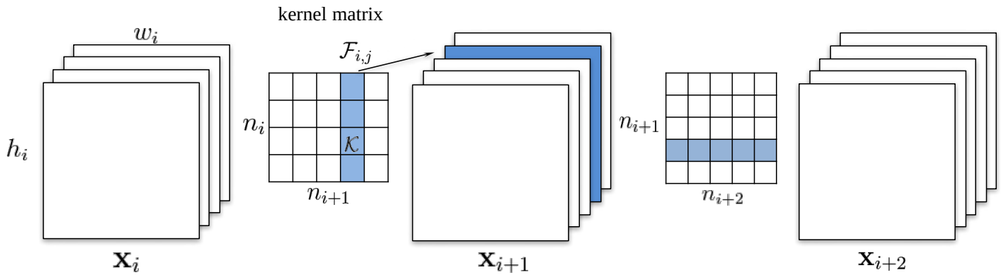
- How to accelerate training with HPC?
- How to train low-precision models on resource-constrained devices?

Understanding the “black box”

- Understand the generalization ability.
- Understand the choice of architectures.
- Understand the optimization process.

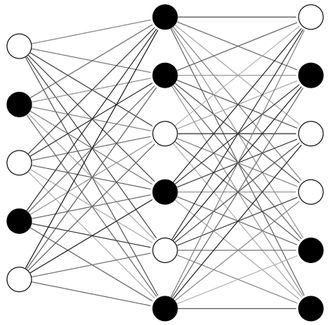
Reducing the Inference Cost

Pruning Filters for Efficient ConvNets, ICLR'17



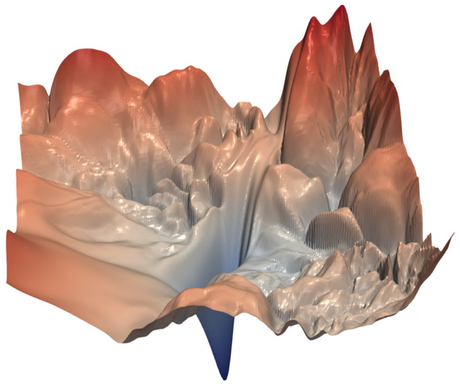
Reducing the Training Cost

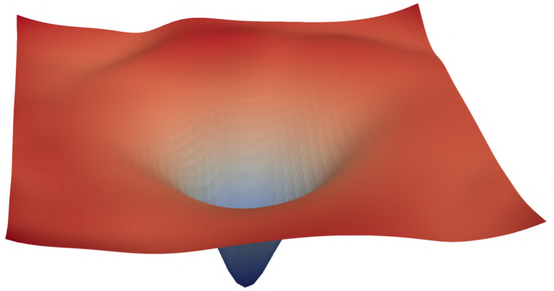
Training Quantized Network: A Deeper Understanding, NIPS'17



Understanding the "black box"

Visualizing the Loss Landscape of Neural Nets, arXiv'17



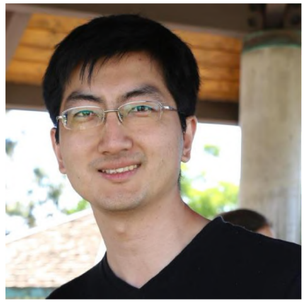


UNIVERSITY OF MARYLAND

NEC Laboratories America
Relentless passion for innovation

Pruning Filters for Efficient ConvNets

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, Hans Peter Graf

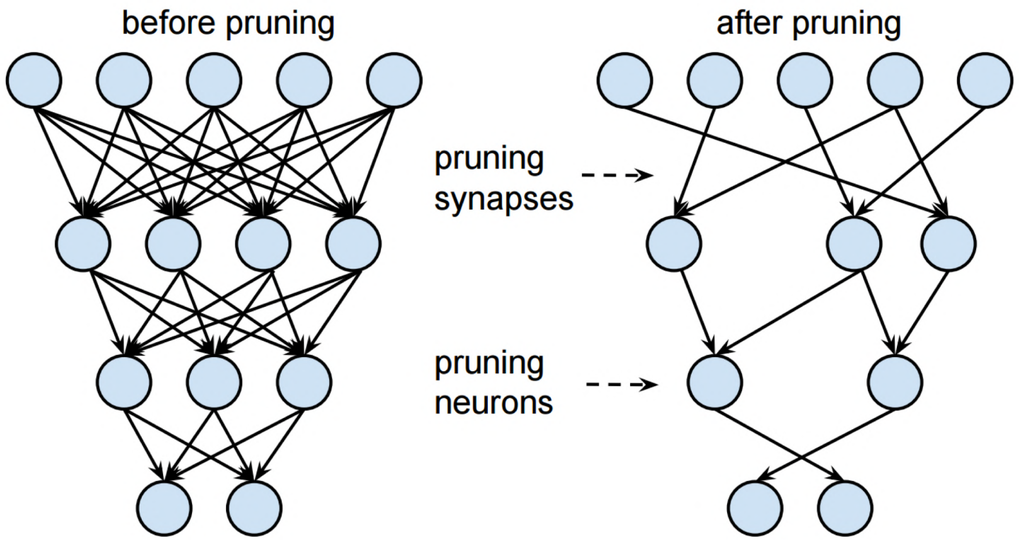




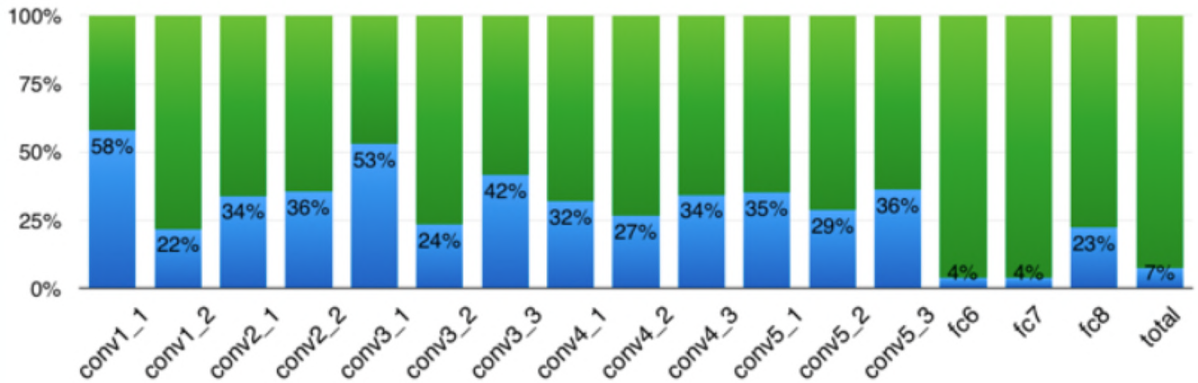
International Conference on Learning Representations (ICLR), 2017

Model Compression by Pruning Weights

Pruning weights with small magnitude

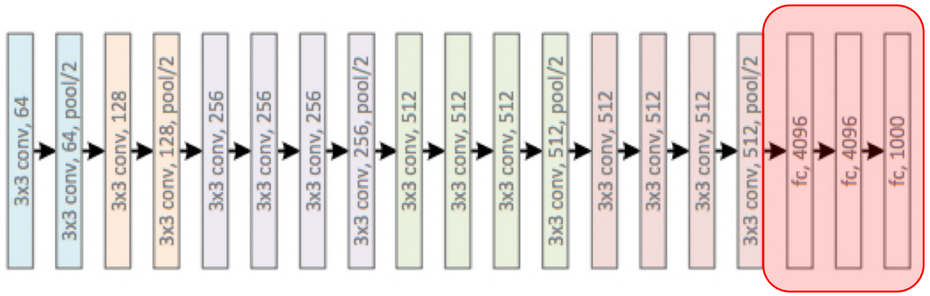


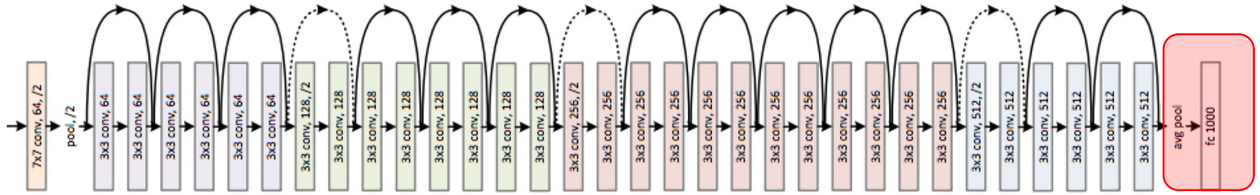
Impressive results for model compression

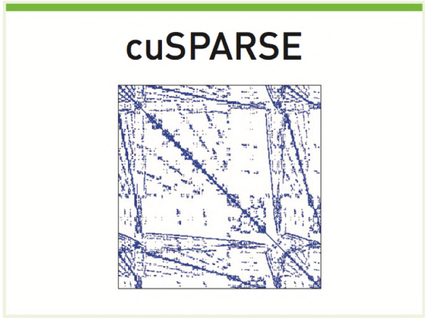


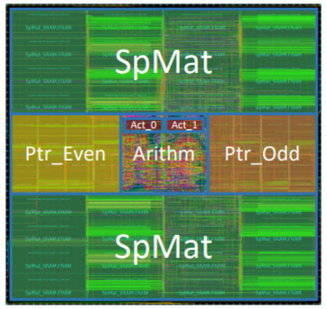
LeCun, et al, “Optimal Brain Damage”, NIPS 1990.
Han, et al, “Learning both Weights and Connections for Efficient Neural Networks”, NIPS 2015.
Han, et al, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

Compression ≠ Acceleration







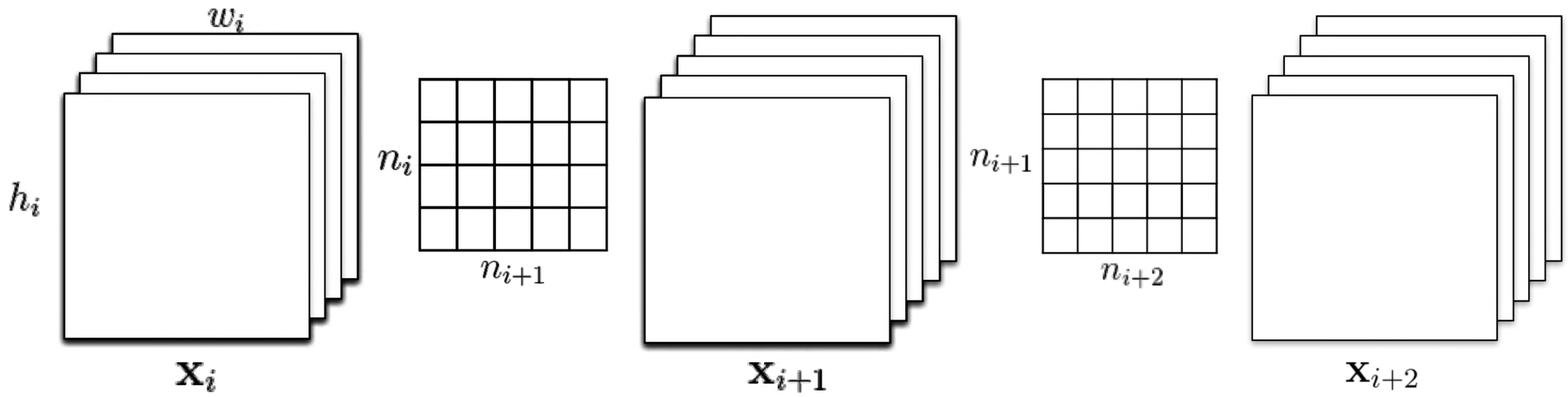


1. 90% of the total parameters of VGG-Net comes from the FC layers, which takes less than 1% total FLOPS.

2. ResNets have **replaced** the FC layers with average pooling layers.

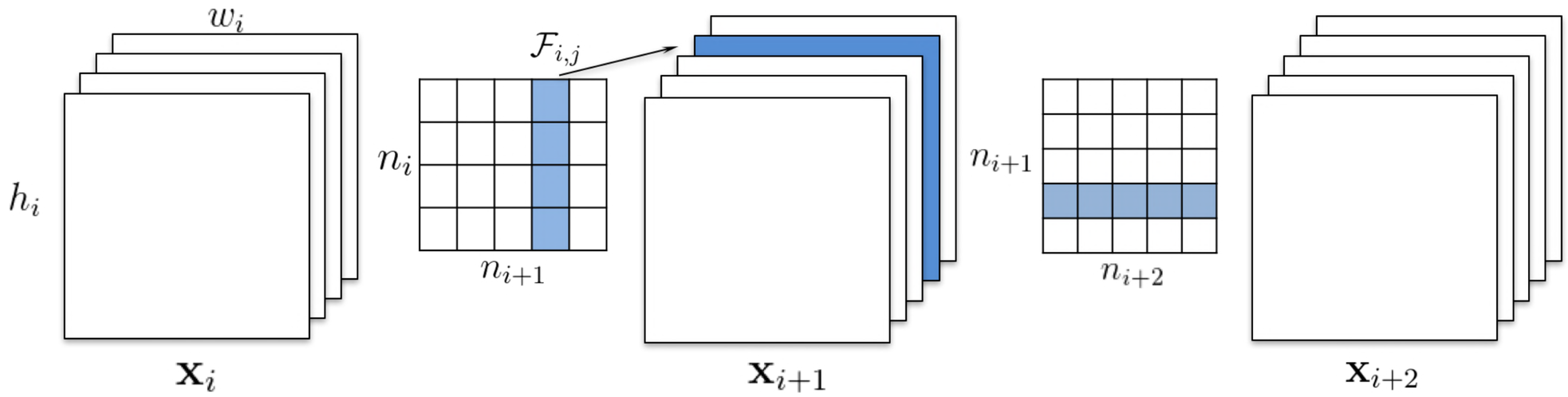
3. Pruning weights generates **sparse convolutional** kernels, which requires **special library** or **hardware**.

Convolutional Operations



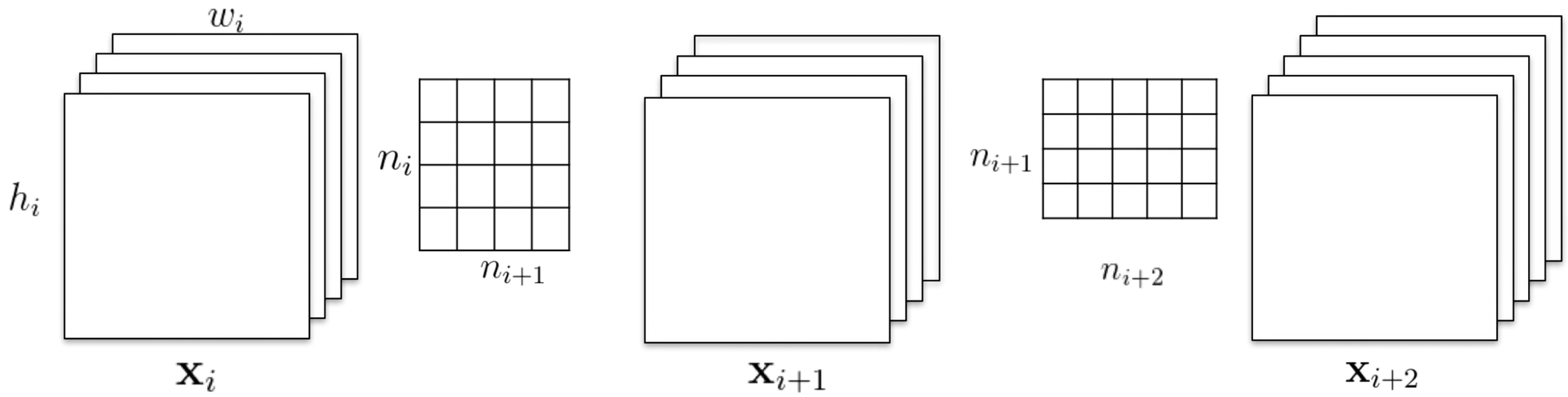
Acceleration by Pruning Filters

Prune a filter, its corresponding feature map and the connected kernels.



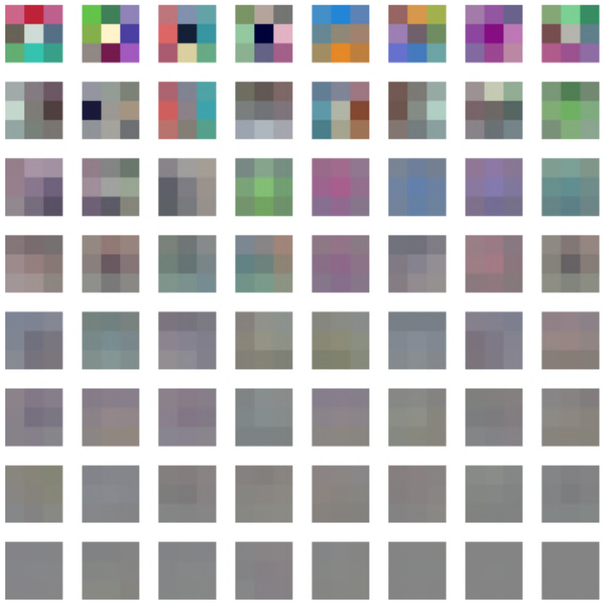
Acceleration by Pruning Filters

Copy the remaining weights to a new model without using mask.

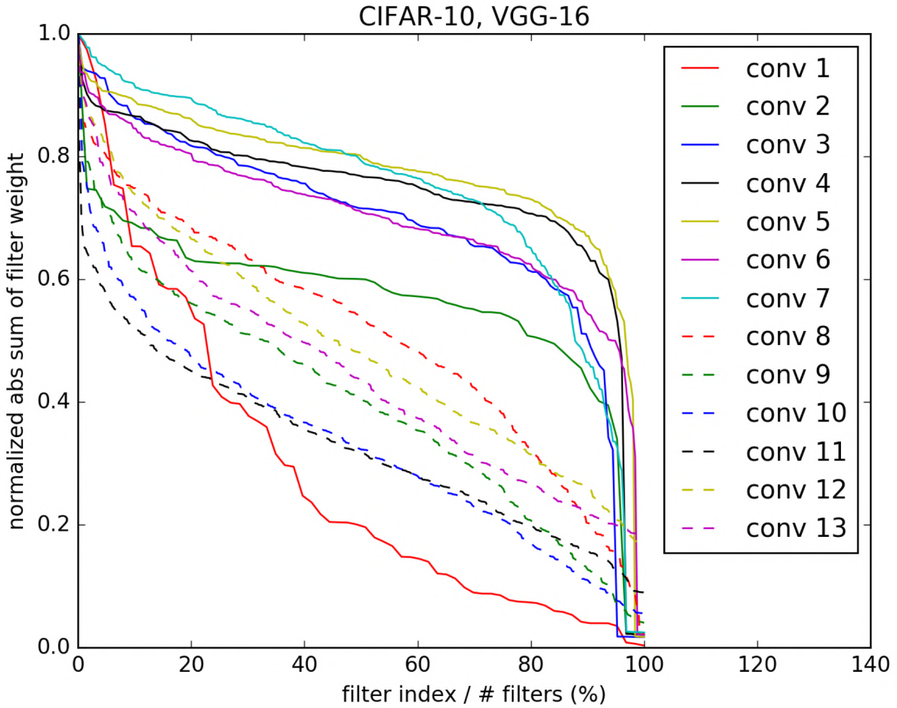


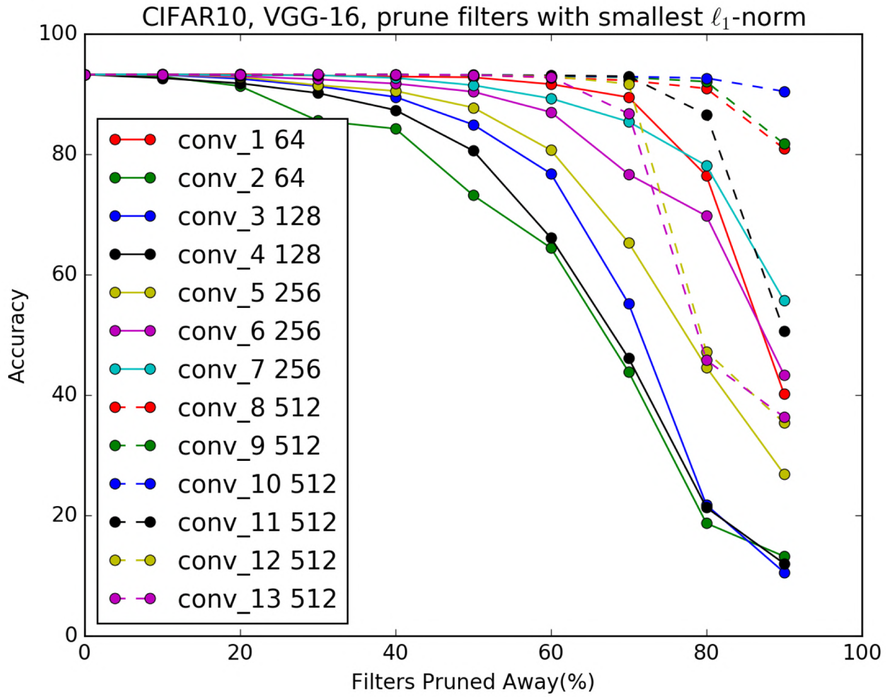
No mask/sparse convolution, no special library/hardware

Determining a Filter's Importance

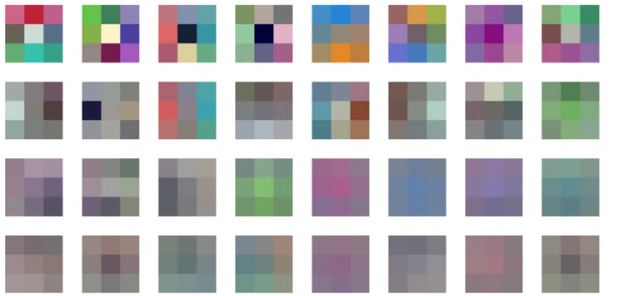


Filters ranked by L1-norm

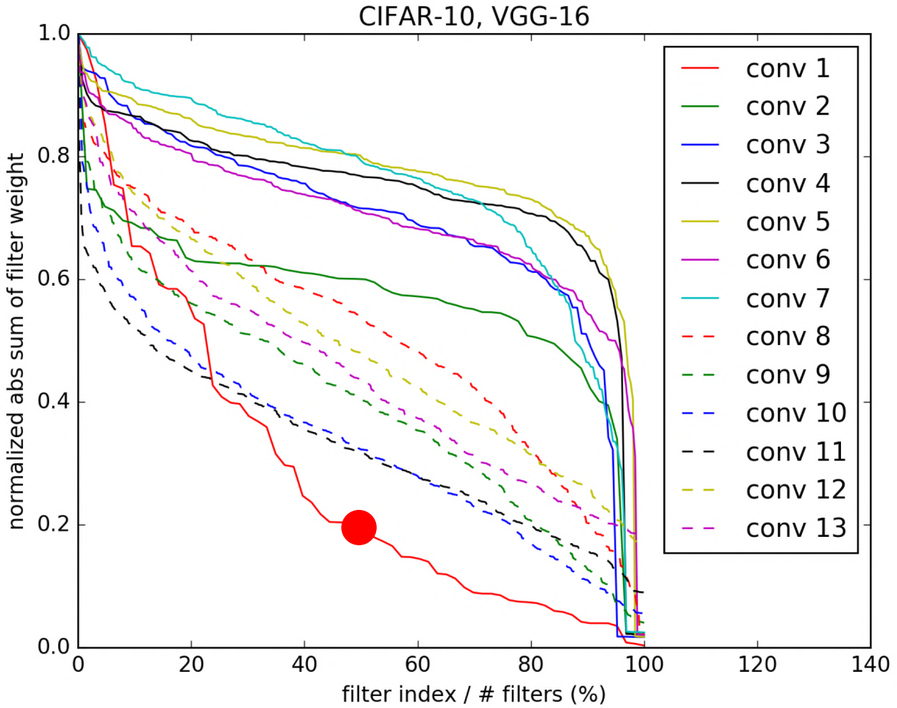


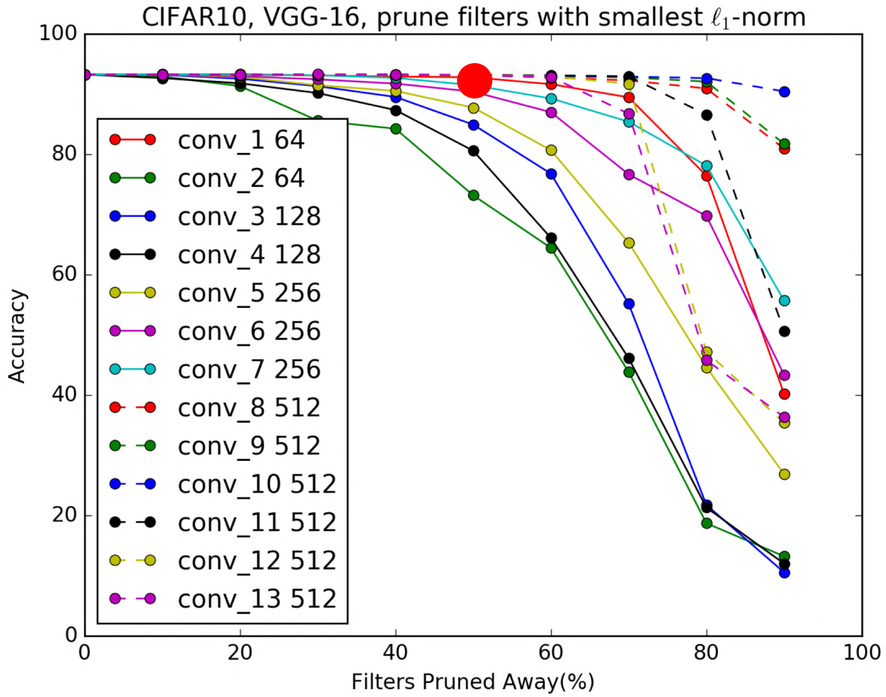


Determining a Filter's Importance

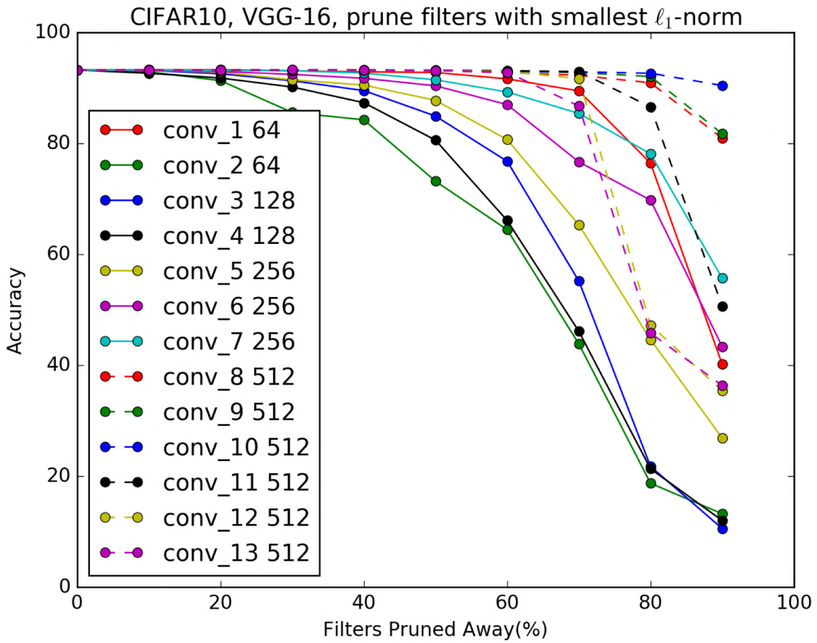


Filters ranked by L1-norm

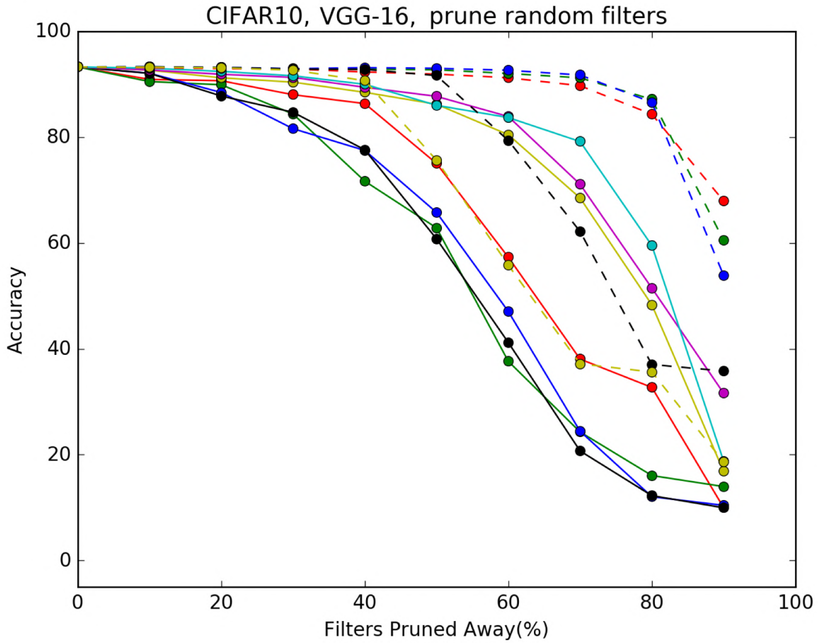




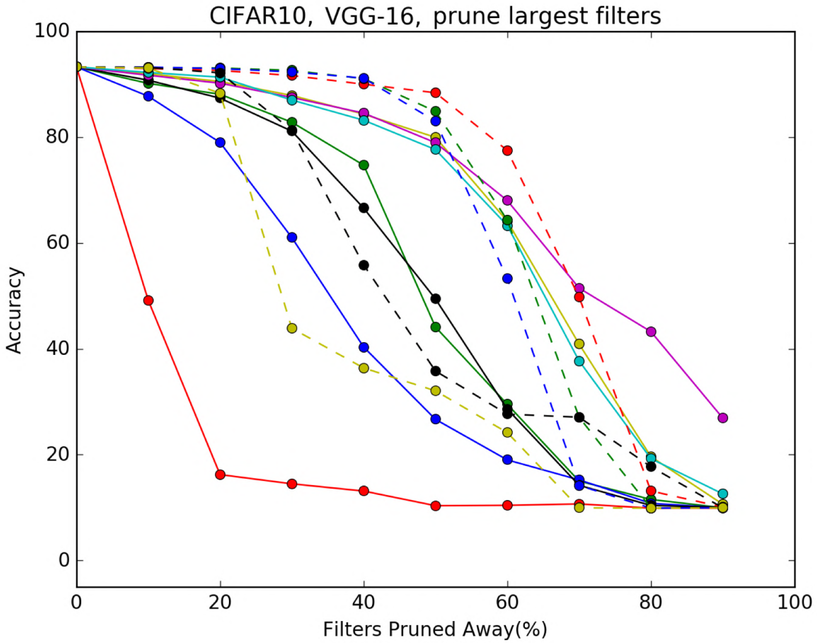
Determining a Filter's Importance



smallest



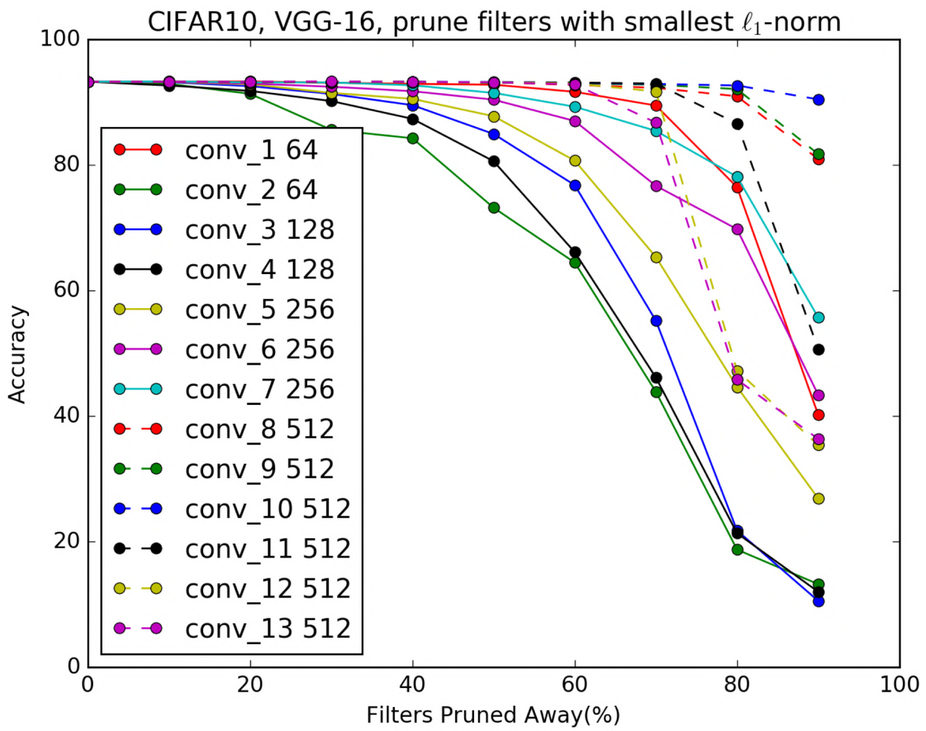
random



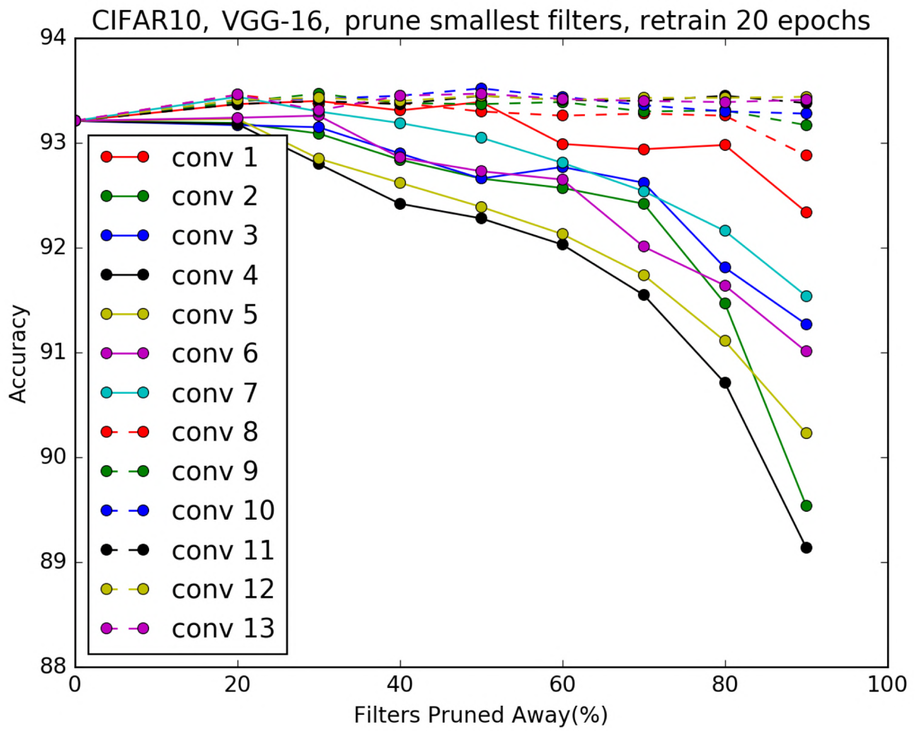
largest

pruning the smallest filters works better than pruning random or the largest filters.

Determining a Layer's Sensitivity to Pruning

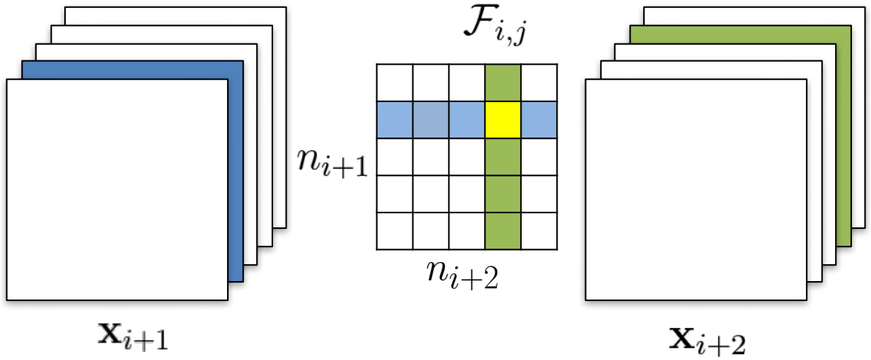


Prune the smallest filters

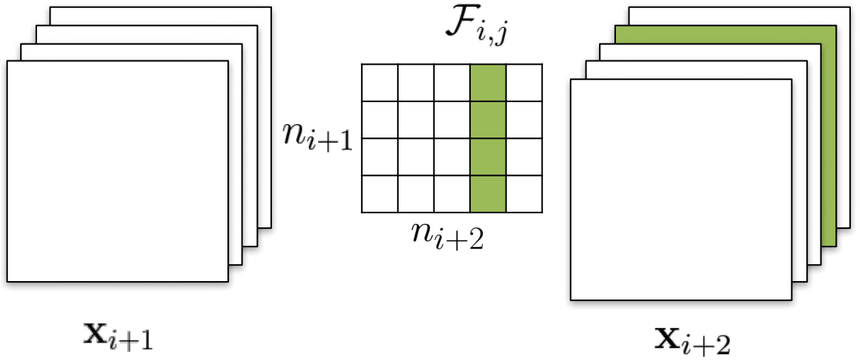


Retrain to recover accuracy

Pruning Filters across Multiple Layers



Independent pruning

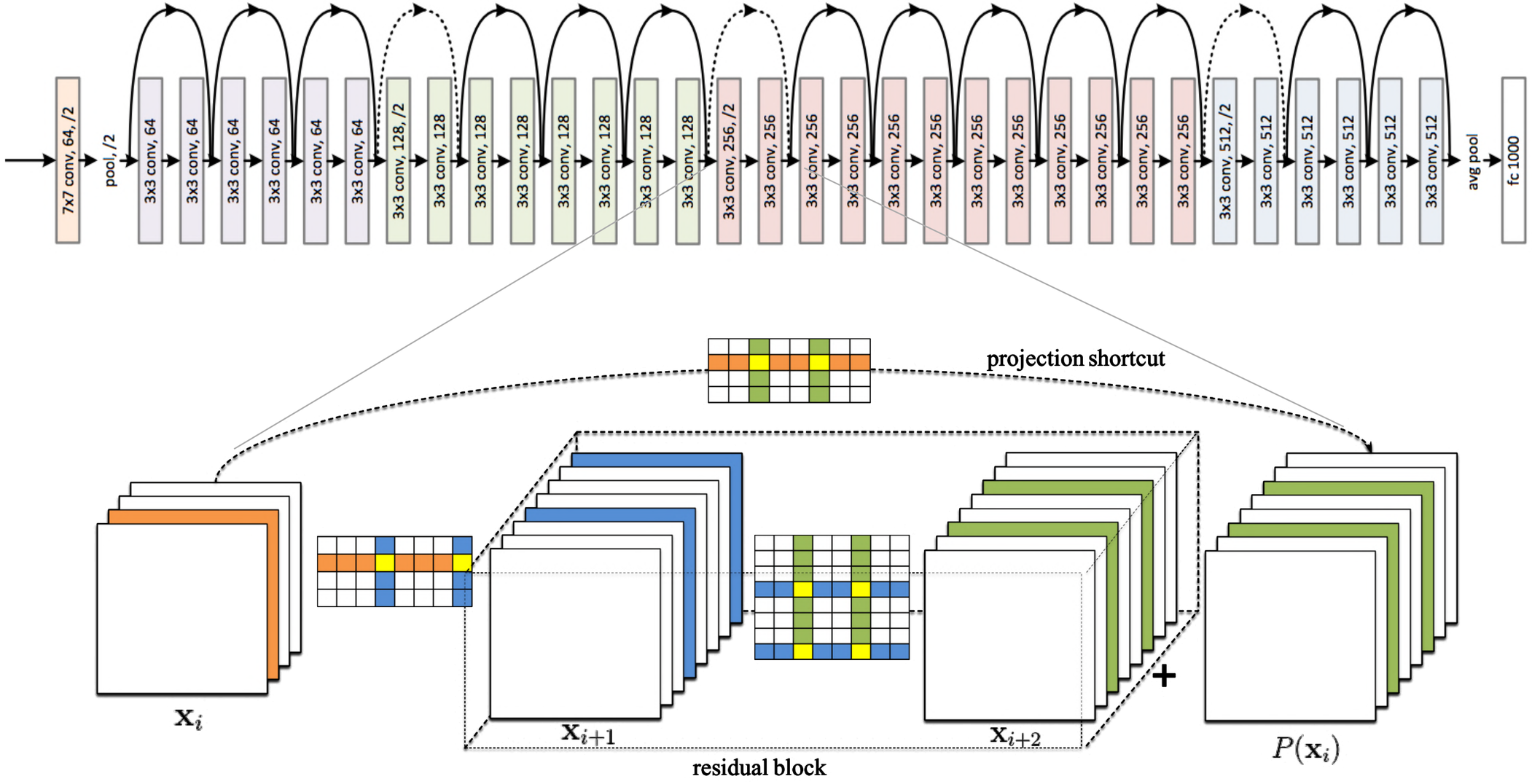


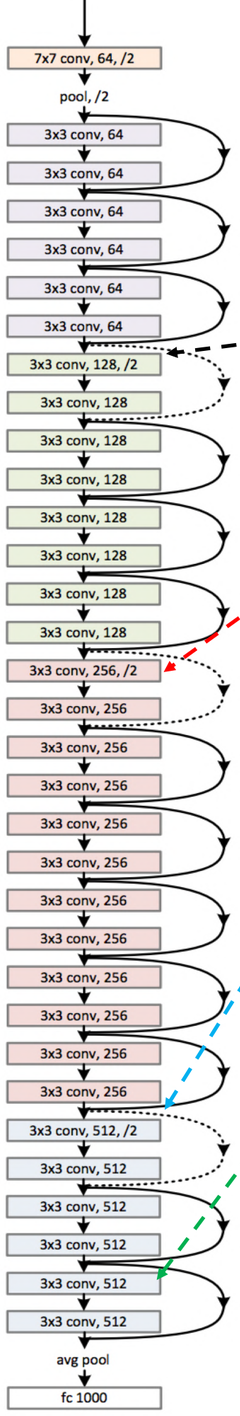
Greedy pruning

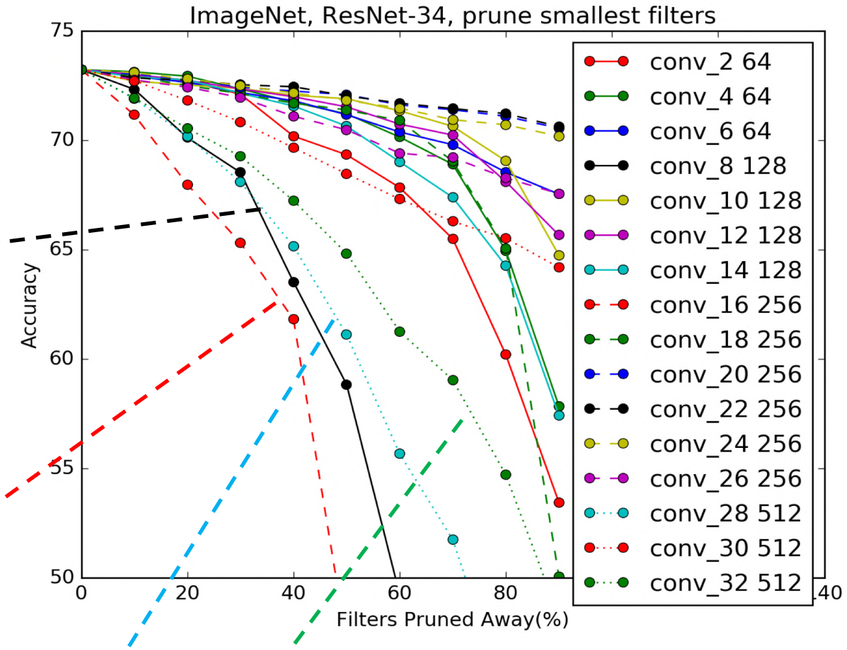
$$\|\mathcal{F}_{i,j}\|_1 = \sum |$$  $$|$$

$$\|\mathcal{F}_{i,j}\|_1 = \sum |$$  $$|$$

Pruning Filters across Multiple Layers

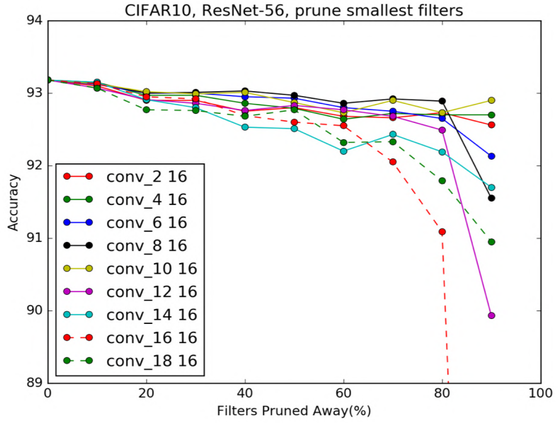


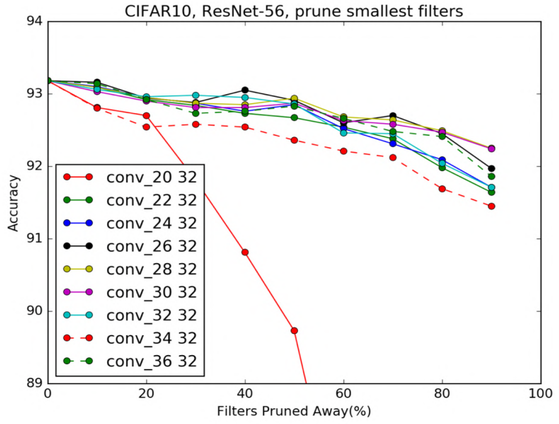


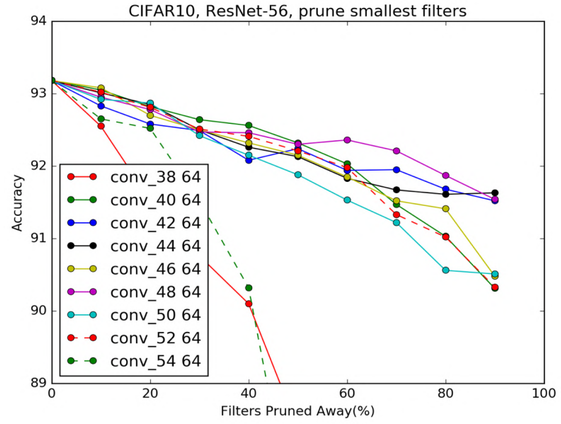


layers that are close to pooling layers are sensitive to pruning

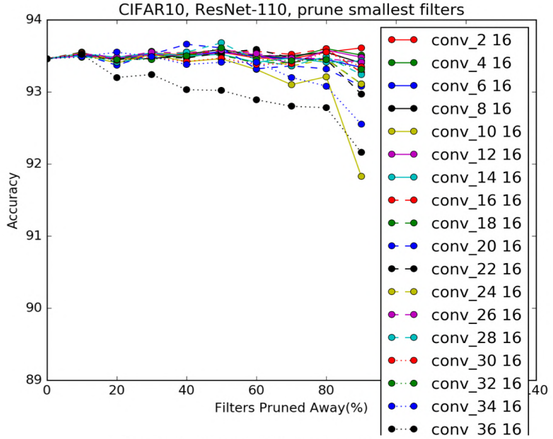
ResNet 34 on ImageNet

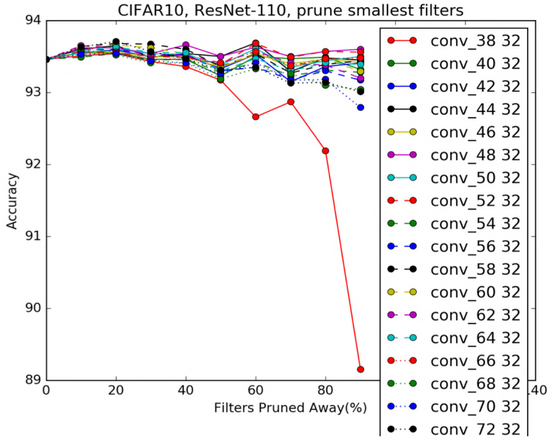


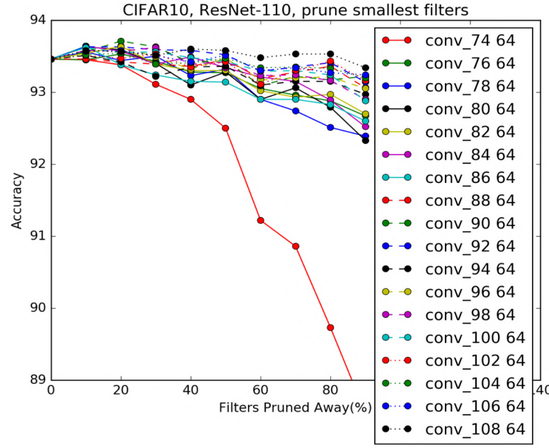




ResNet 56







ResNet 110

Results

Model	Top-1 Error	Pruned FLOPs	Pruned Params
VGG-16	6.75		
VGG-16-pruned-A	**6.60**	34.2%	64%
VGG-16-pruned-A scratch train	6.88		
ResNet-56	6.96		
ResNet-56-pruned-A	6.90	10.4%	9.4%
ResNet-56-pruned-B	**6.94**	27.6%	13.7%
ResNet-56-pruned-B scratch train	8.69		
ResNet-110	6.47		
ResNet-110-pruned-A	**6.45**	15.9%	2.3%
ResNet-110-pruned-B	6.70	38.6%	32.4%
ResNet-110-pruned-B scratch train	7.06		
ResNet-34	26.77		
ResNet-34-pruned-A	27.44	15.5%	7.6%
ResNet-34-pruned-B	27.83	24.2%	10.8%

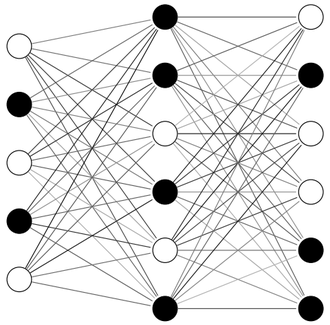
- 24%~ 38% reduction in FLOPs for VGG-16 and ResNets without losing accuracy on CIFAR-10.
- 24% reduction in FLOPs for ResNet-34 with ~1% loss in accuracy on ImageNet.
- Pruning a **wider** model with retraining performs better than training the pruned thin network from scratch.

Reducing the Inference Cost

Pruning Filters for Efficient ConvNets, ICLR'17

Reducing the Training Cost

Training Quantized Network: A Deeper Understanding, NIPS'17



Understanding the "black box"

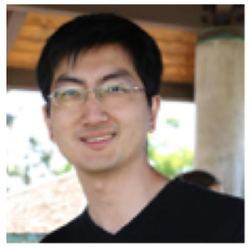
Visualizing the Loss Landscape of Neural Nets, arXiv'17

UNIVERSITY OF MARYLAND

Cornell University

Training Quantized Networks: a Deeper Understanding

Hao Li*, Soham De*, Zheng Xu, Christoph Studer, Hanan Samet, Tom Goldstein

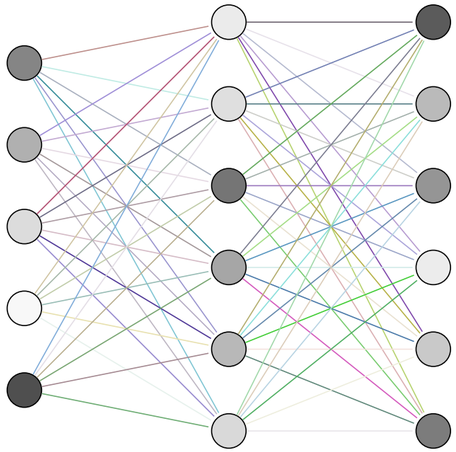





Neural Information Processing Systems (NIPS), 2017

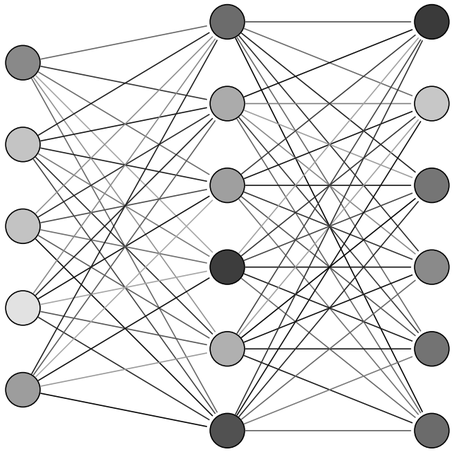
Low-Precision Neural Networks

32-bit W, 32-bit A



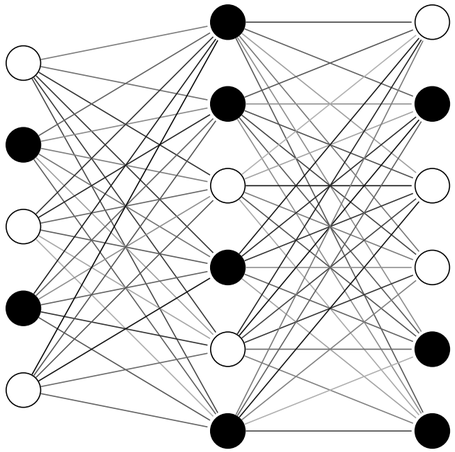
Full-precision model

1-bit W, 32-bit A



32× smaller, ~2× faster with addition and subtraction ops

1-bit W, 1-bit A



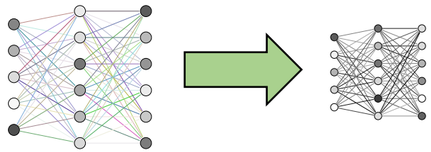
~58× faster with XNOR and bit counting ops

Rastegari, et al, “XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks”, ECCV 2016

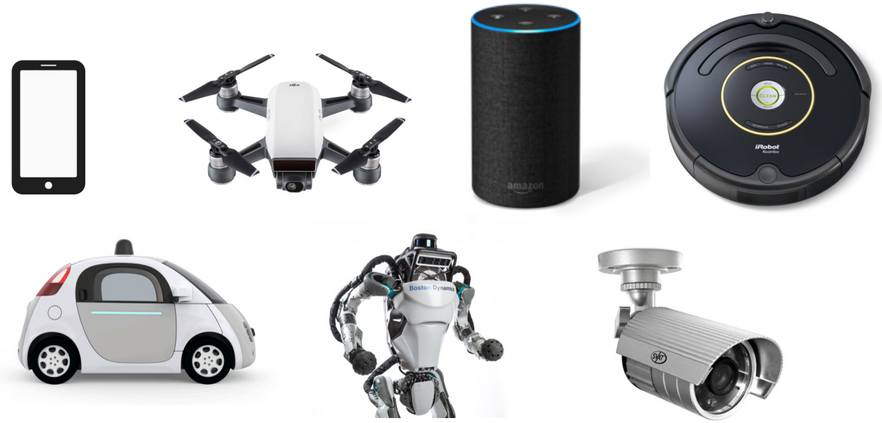
Training DNNs: from HPC to Embedded Devices

Training on HPC

Quantization

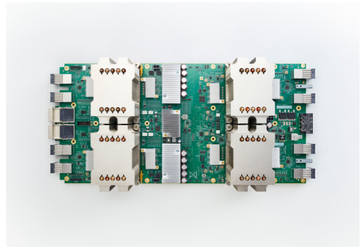


Inference on Devices

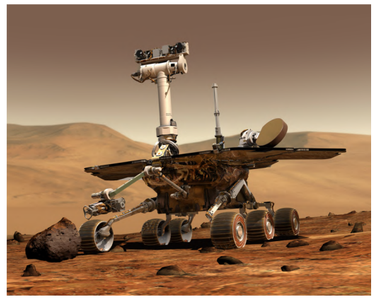


How to train quantized models on **resource-constrained devices?**

TPU 1.0



TPU 2.0



How to Train Quantized Nets?

- Non-quantized: SGD

$$w^{t+1} = w^t - \alpha_t \nabla \tilde{f}(w^t)$$

- Semi-quantized: BinaryConnect [Courbariaux et al NIPS'15]

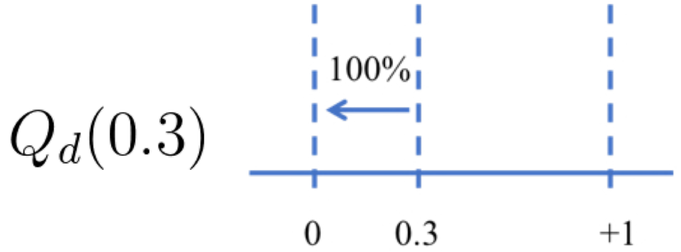
QNN[Hubara et al'16],
XNOR-Net [Rastegar et al, ECCV'16],
DoReFA-Net [Zhou et al'16] …

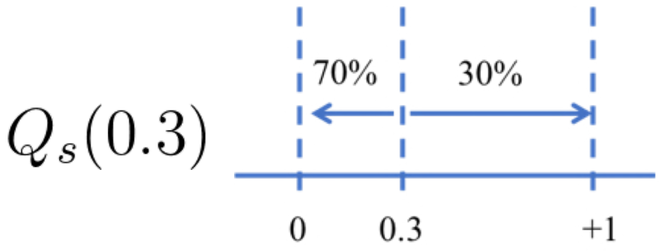
fast forward propagation $\quad w_b^t = Q(w_r^t)$

accumulating gradients $\quad w_r^{t+1} = w_r^t - \alpha_t \nabla \tilde{f}(w_b^t)$ requires full-precision weights

- Fully-quantized: Deterministic / Stochastic Rounding [Gupta et al, ICML'15]

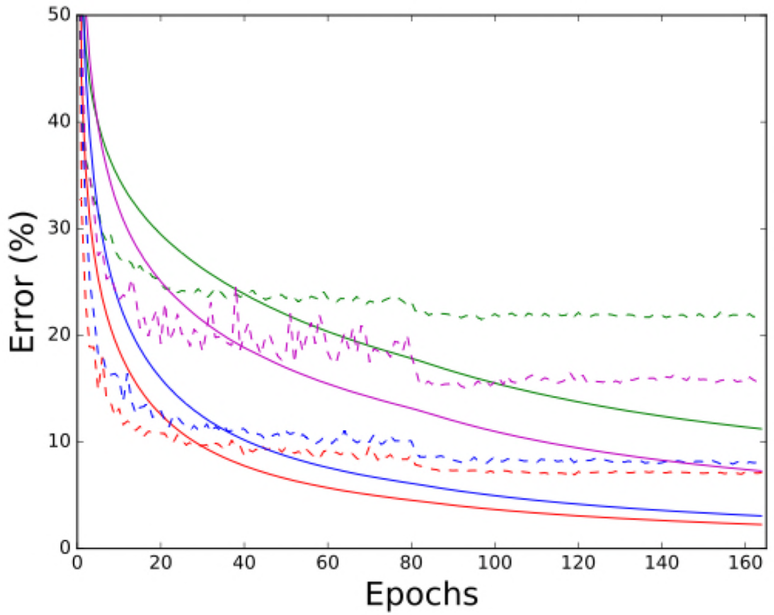
$$w_b^{t+1} = w_b^t - Q(\alpha_t \nabla \tilde{f}(w_b^t))$$ no full-precision weights





Empirical Result

Train CNNs with binary weights on CIFAR-10



Deterministic Rounding

Stochastic Rounding

BinaryConnect

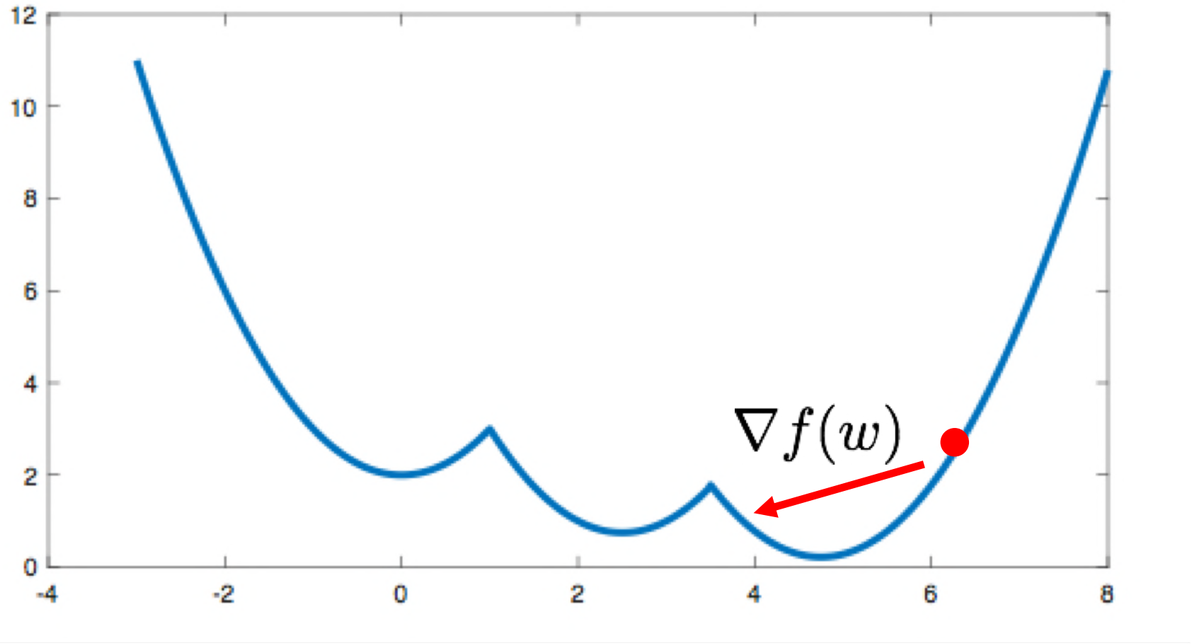
Full-Precision

Keeping the real-valued weights seems to really help empirically.

Why does keeping floating point weights help?

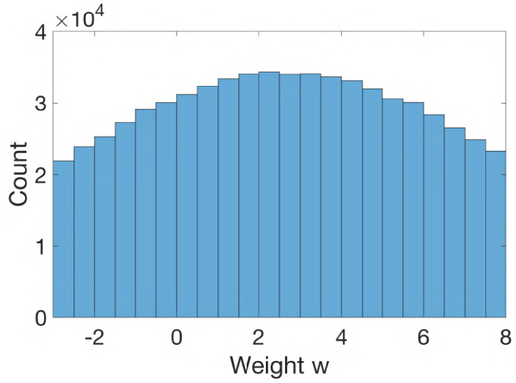
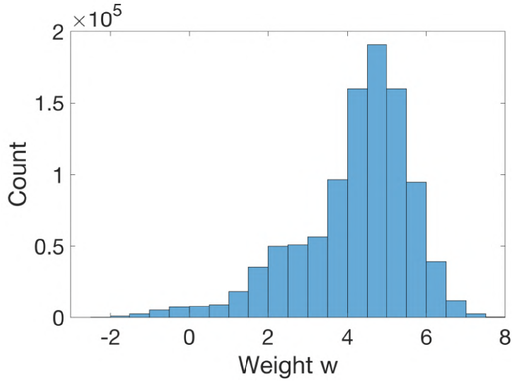
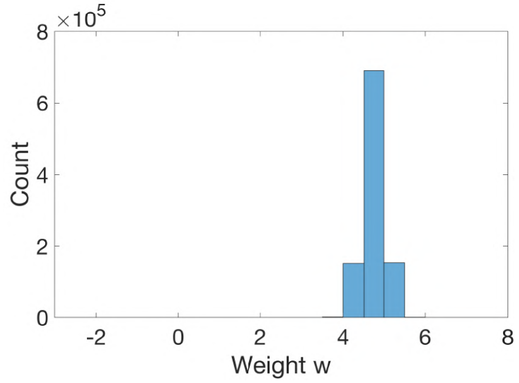
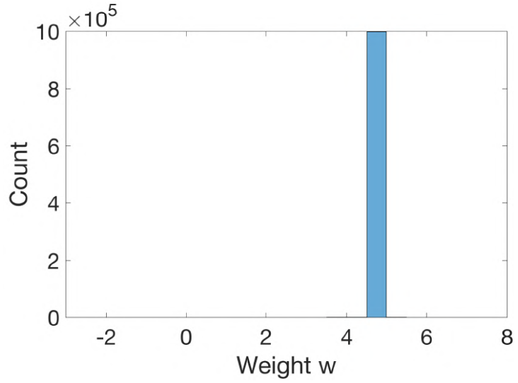
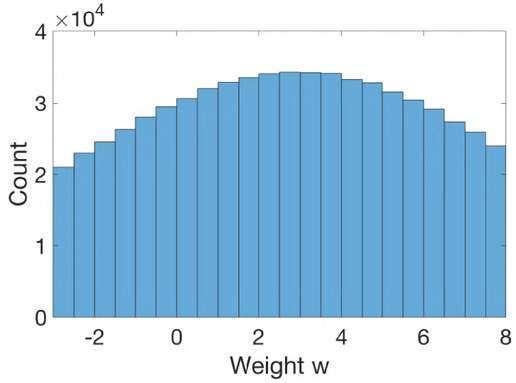
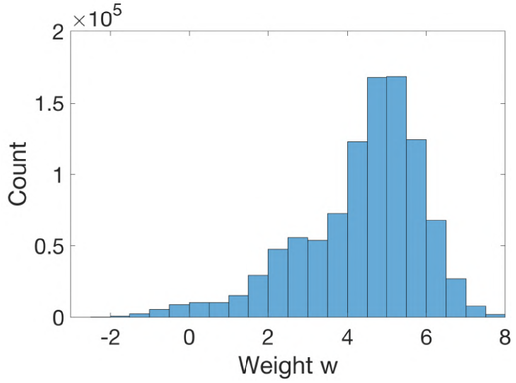
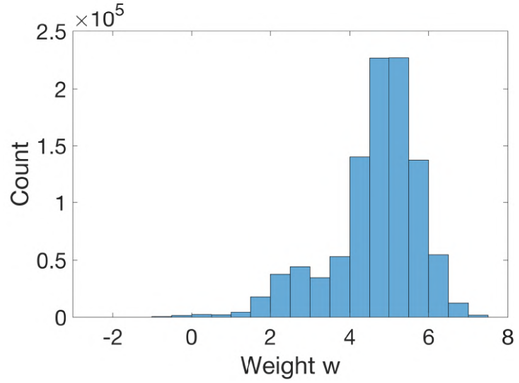
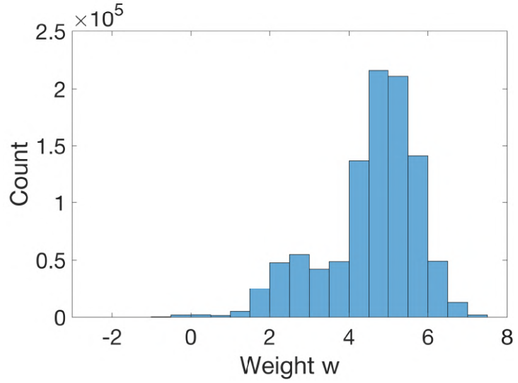
Toy Example for Non-Convex Problems

learning rate = 1



Quantized scalar weight $\Delta = 0.5$

Weight Distribution after 1M Steps

	lr = 1	lr = 0.1	lr = 0.01	lr = 0.001
BC				
SR				

Exploration-Exploitation Tradeoff

SGD/BinaryConnect

Exploration

Exploitation

Exploration

Exploitation

Stochastic Rounding

Exploration

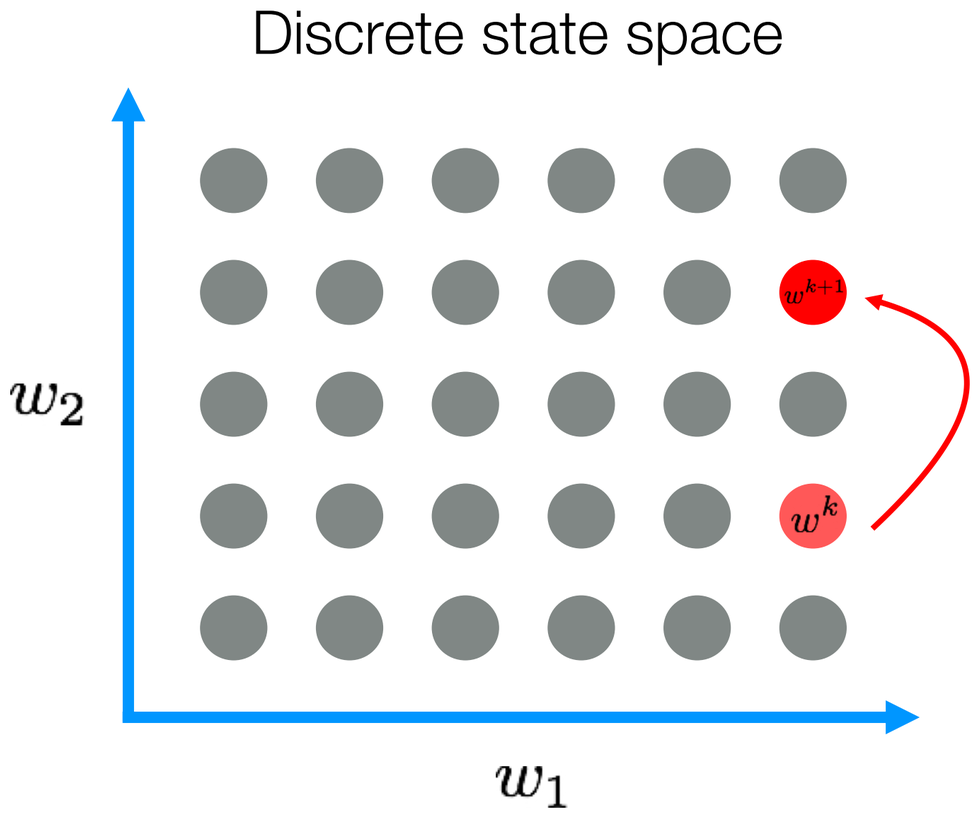
Exploitation

Exploration

Exploitation

Markov Chain Interpretation

$$w_b^{t+1} = w_b^t - Q\left(\alpha_t \nabla \tilde{f}\left(w_b^t\right)\right)$$



Markov property

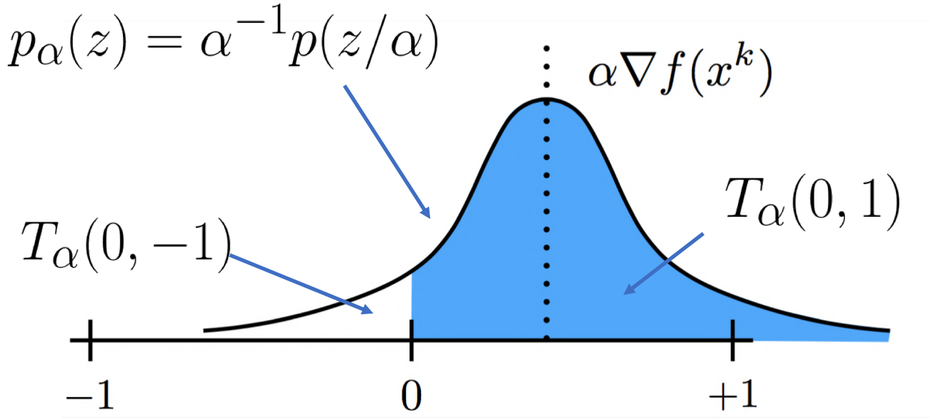
The probability of moving to the next state depends only on the present state and not on the previous states.

Markov process

SR starts at some state x, and moves to a new state y with some transition probability $T(x, y)$ that depends only on x and the learning rate α. For fixed α, this is a Markov process with transition matrix $T_\alpha(x, y)$.

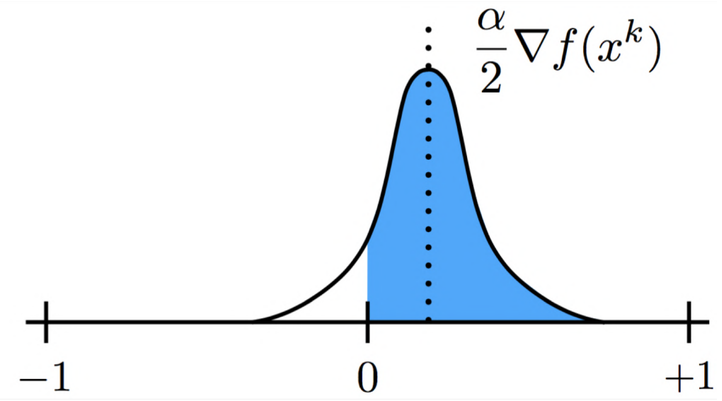
Markov Chain Interpretation

$$w_b^{t+1} = w_b^t - Q\left(\alpha_t \nabla \tilde{f}\left(w_b^t\right)\right)$$



the distribution of perturbation

shrink learning rate



$$T_\alpha(0,1|x^1 \neq x^0) \approx \frac{T_\alpha(0,1)}{T_\alpha(0,-1) + T_\alpha(0,1)} = \frac{\int_0^\infty p(x)x\,dx}{\int_{-\infty}^0 p(x)x\,dx + \int_0^\infty p(x)x\,dx}$$

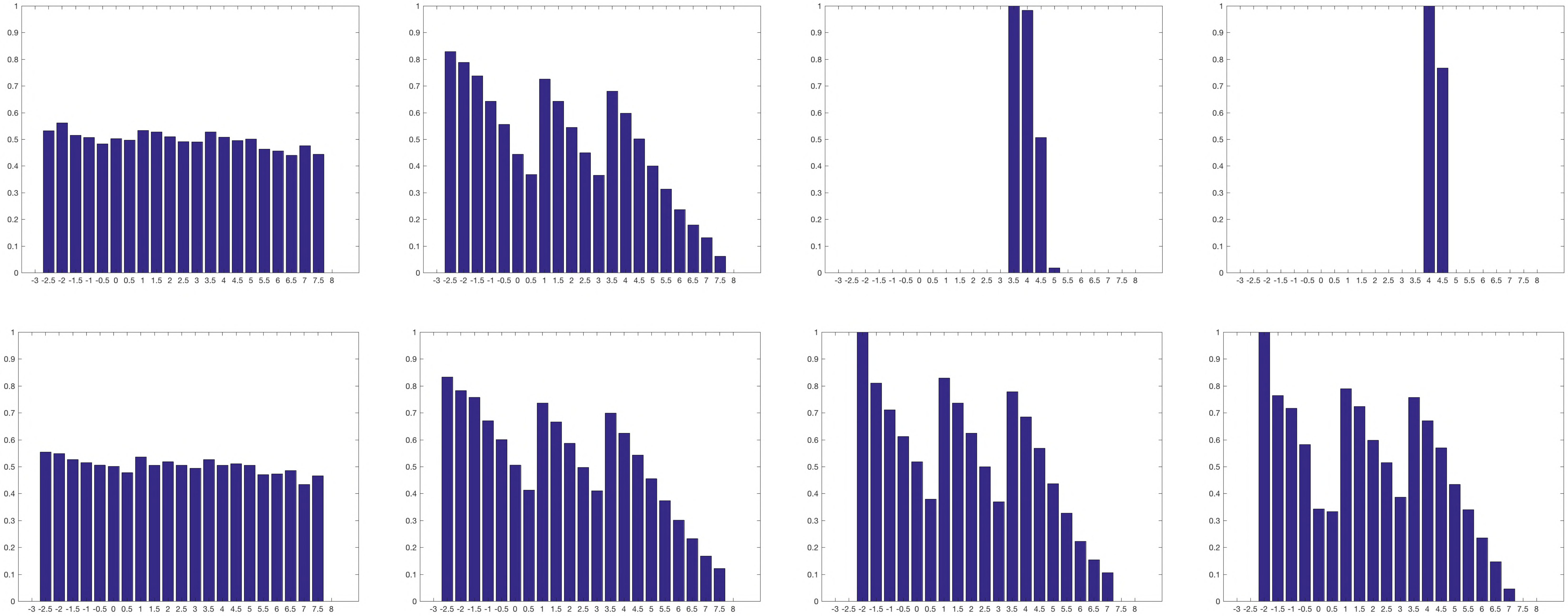
The conditional probability does not depend on the learning rate!

Conditional Probability $T_\alpha(0, 1 | x^1 \neq x^0)$

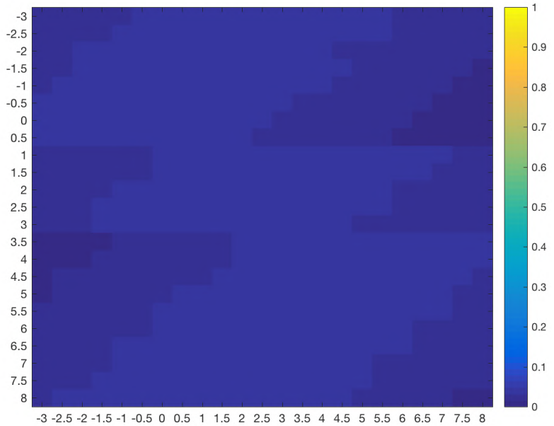
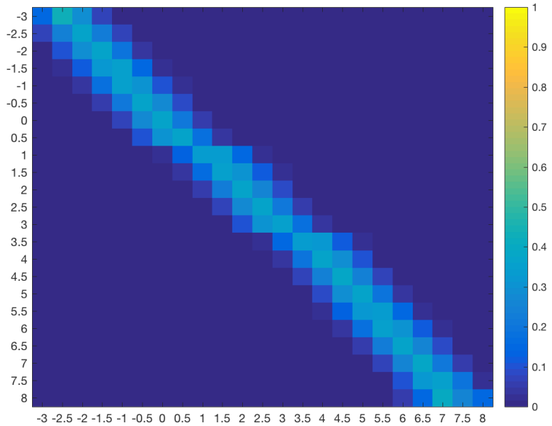
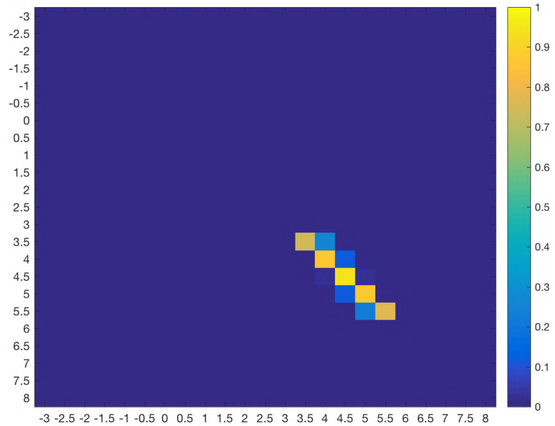
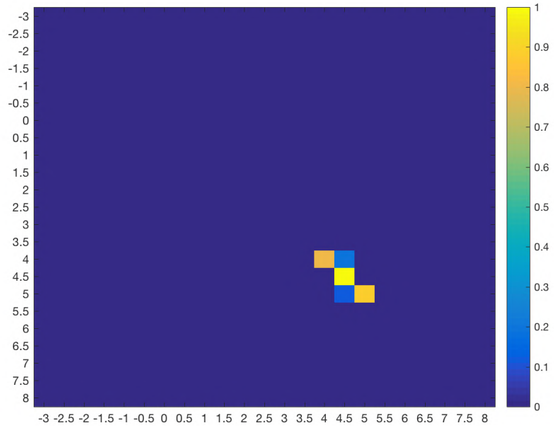
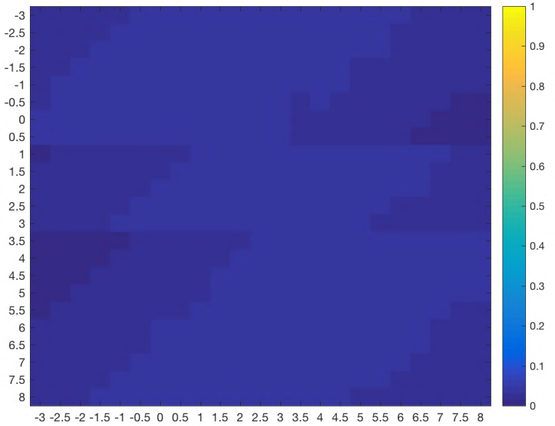
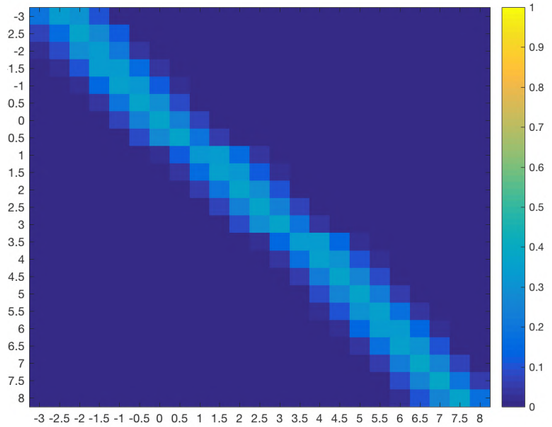
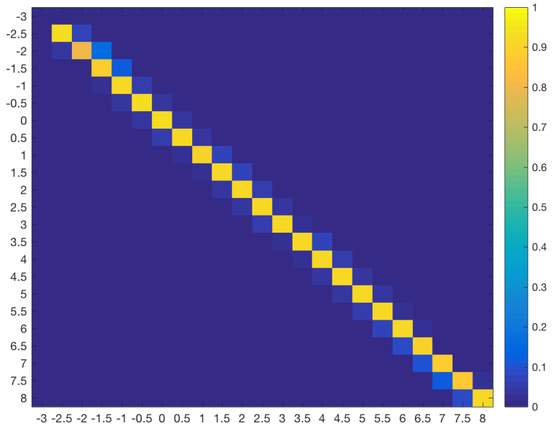
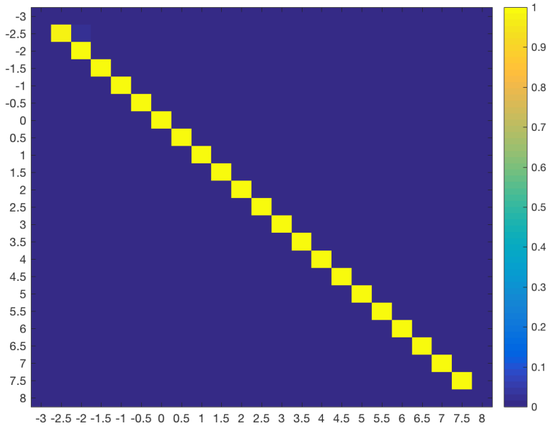
lr = 1   lr = 0.1   lr = 0.01   lr = 0.001

BC

SR



Transition Probability $T_\alpha(x, y)$

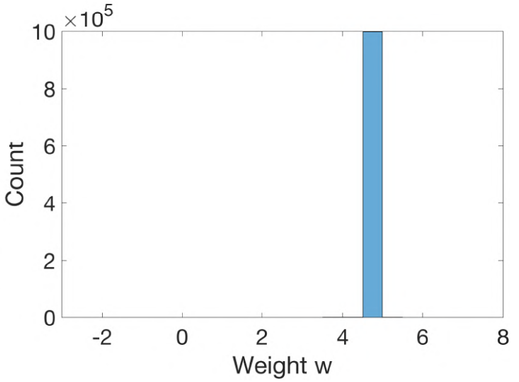
	lr = 1	lr = 0.1	lr = 0.01	lr = 0.001
BC				
SR				

Long Term Behavior

`lr = 1` `lr = 0.1` `lr = 0.01` `lr = 0.001`

BC

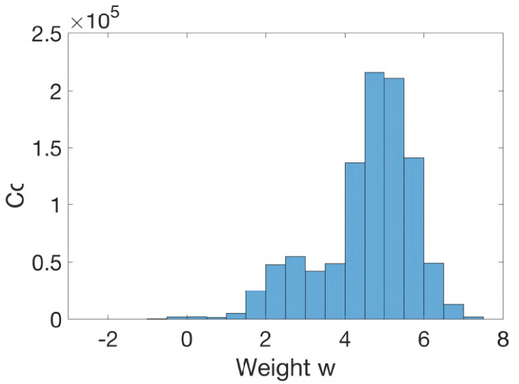
The stationary distribution of BinaryConnect concentrates on stationary points.



These algorithms have an exploitation phase!

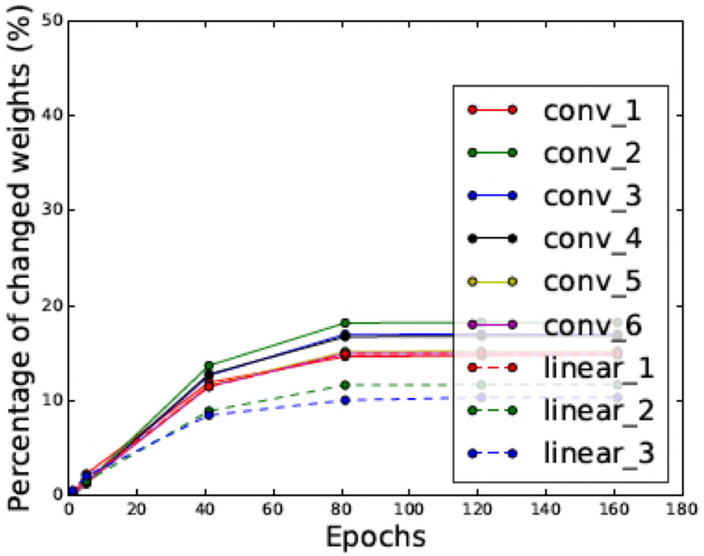
SR

The stationary distribution of Stochastic Rounding does not concentrate on stationary points.

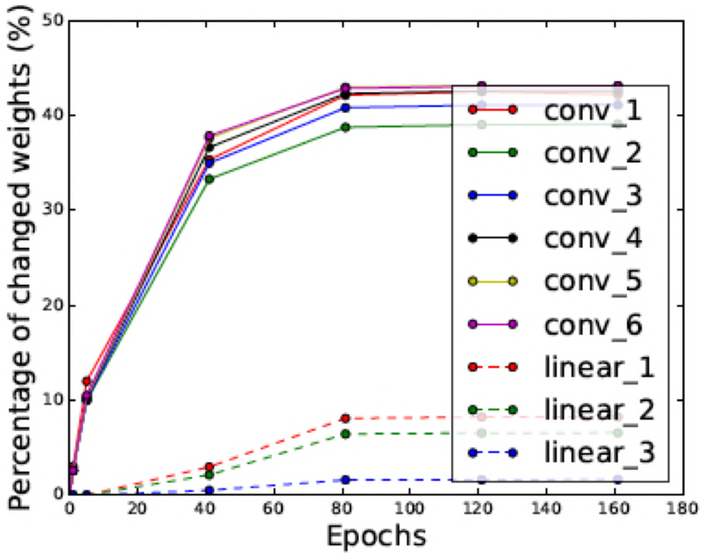


Exploration slows down, but exploitation never happens!

Experiment Results



BinaryConnect



Stochastic Rounding

- The BC method changes less than 20% of the weights, which indicates that BC is able to change from **explorative** to **exploitative** – it drives more towards local minimizers and explores less aggressively.
- The SR method is not able to exit the exploration phase; it keeps changing weights until 50% of the weights differ from their starting values.

Experiment Results

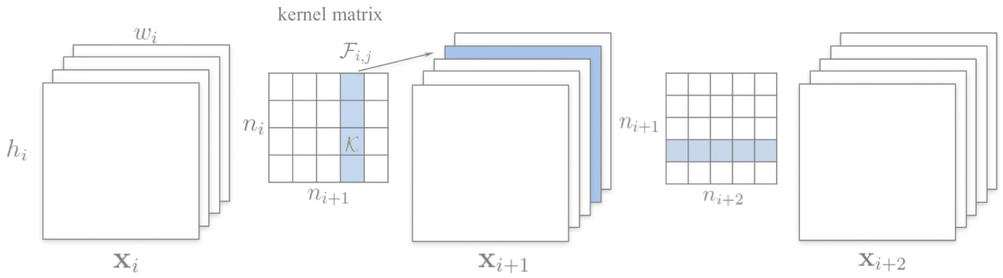
Table 1: Top-1 test error after training with full-precision (ADAM), binarized weights (R-ADAM, SR-ADAM, BC-ADAM), and binarized weights with big batch size (Big SR-ADAM).

	CIFAR-10				CIFAR-100	ImageNet
	VGG-9	VGG-BC	ResNet-56	WRN-56-2	ResNet-56	ResNet-18
ADAM	7.97	7.12	8.10	6.62	33.98	36.04
BC-ADAM	10.36	8.21	8.83	7.17	35.34	52.11
Big SR-ADAM	16.95	16.77	19.84	16.04	50.79	77.68
SR-ADAM	23.33	20.56	26.49	21.58	58.06	88.86
R-ADAM	23.99	21.88	33.56	27.90	68.39	91.07

- The binary model trained by BC-ADAM has comparable performance to the full-precision model.
- There is a performance gap between SR-ADAM and BC-ADAM across all models and datasets.
- Wide-Residual Networks are easier to train and generalize better.

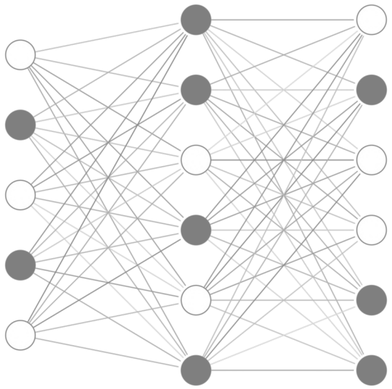
Reducing the Inference Cost

Pruning Filters for Efficient ConvNets, ICLR'17



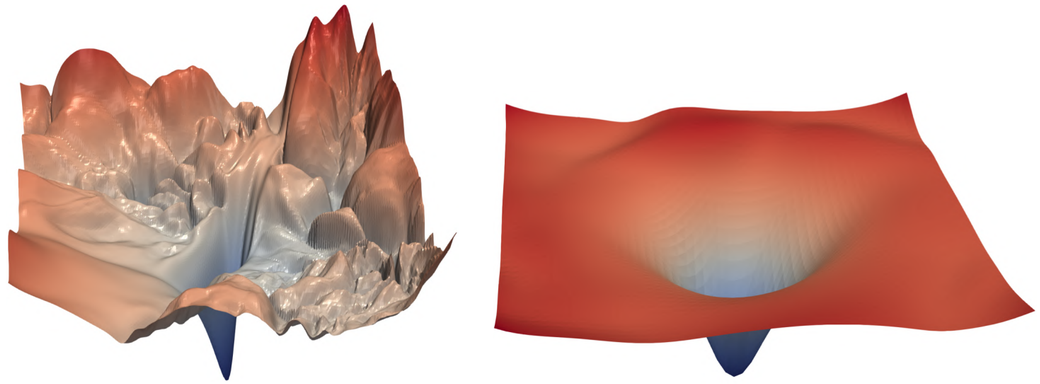
Reducing the Training Cost

Training Quantized Network: A Deeper Understanding, NIPS'17



Understanding the "black box"

Visualizing the Loss Landscape of Neural Nets, arXiv'17



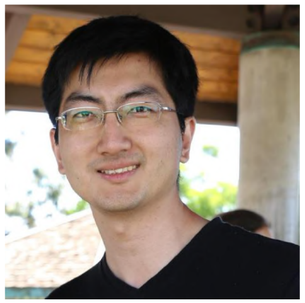
UNIVERSITY OF MARYLAND

USNA UNITED STATES NAVAL ACADEMY

Cornell University

Visualizing the Loss Landscape of Neural Nets

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein



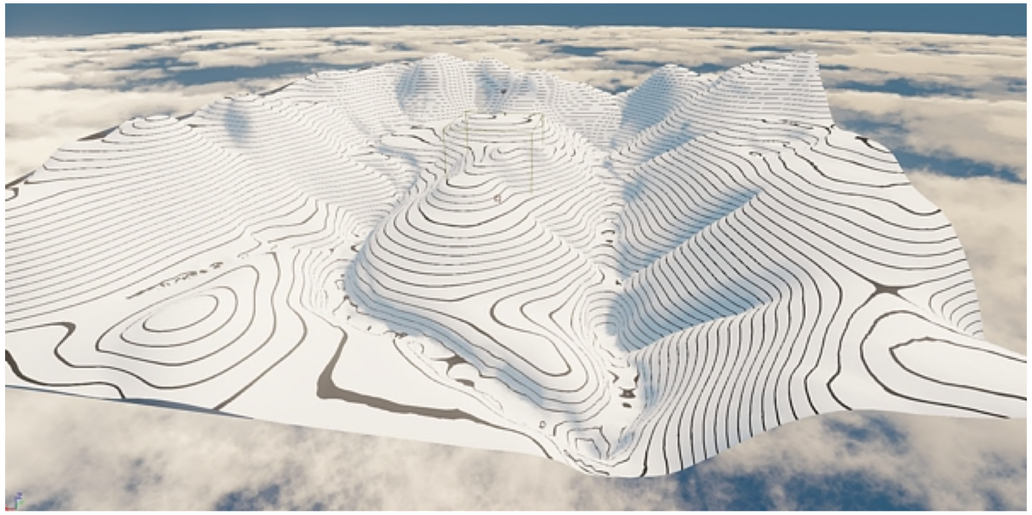


Why Visualize the Loss Surface

Training neural networks requires minimizing a **high-dimensional non-convex** loss function

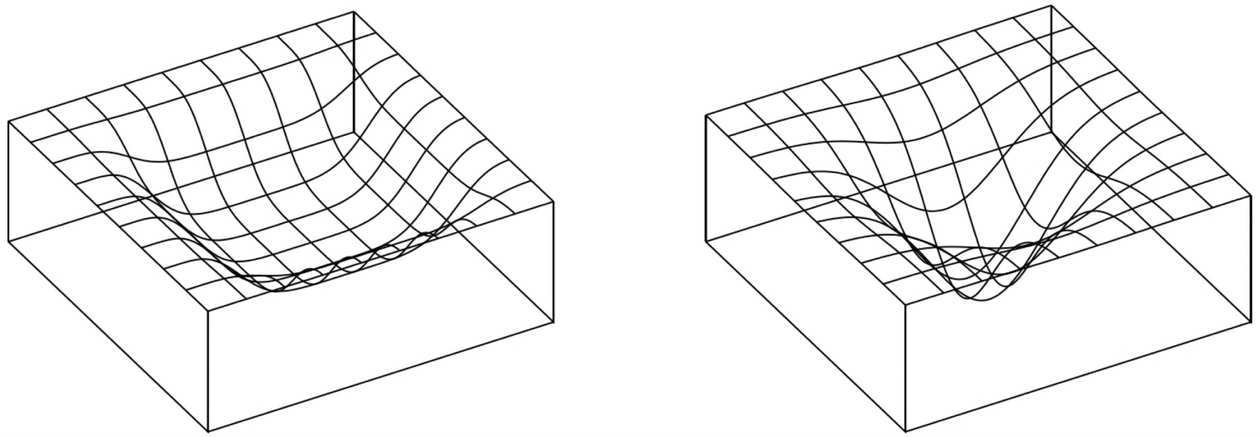
$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m}\ell(x_i, y_i; \theta)$$

which is a process of searching for a weight vector which gives the smallest loss value.



Geometric intuition of the loss surface is a key way of improving optimization methods and architecture design.

The Flatness/Sharpness of a Minimizer

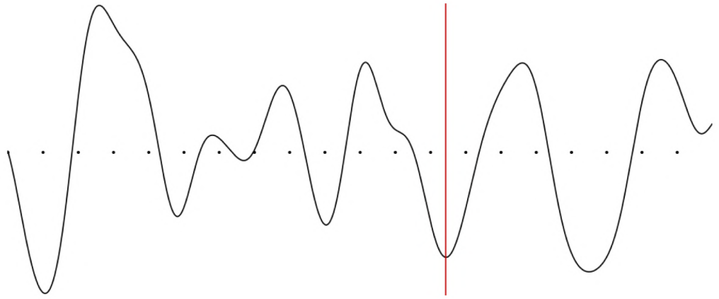


“flat region” of the loss landscape are robust to

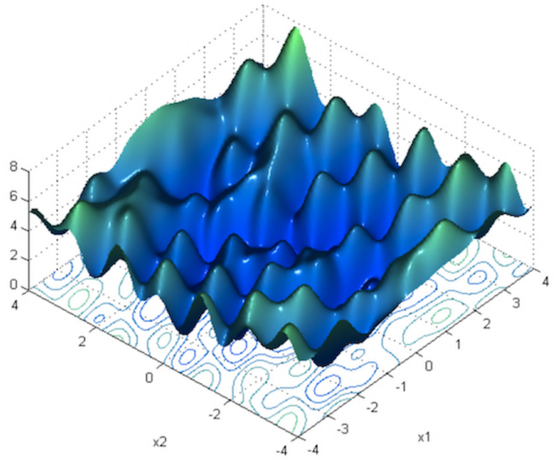
- data perturbations
- noise in the activations
- perturbations of the parameters (can be specified with lower-precision weights)

Hochreiter & Schmidhuber. Flat minima. Neural Computation, 1997
Chaudhari, et al, Entropy-sgd: Biasing Gradient Descent into Wide Valleys, ICLR 2017

Interpreting High-Dimension Loss Surface



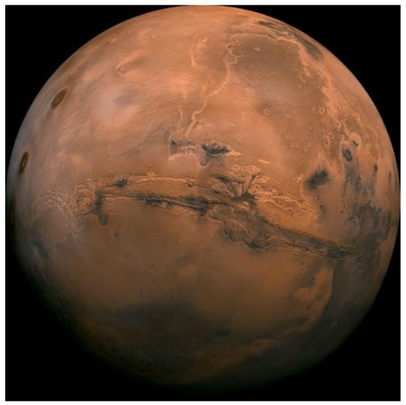
1-Variable

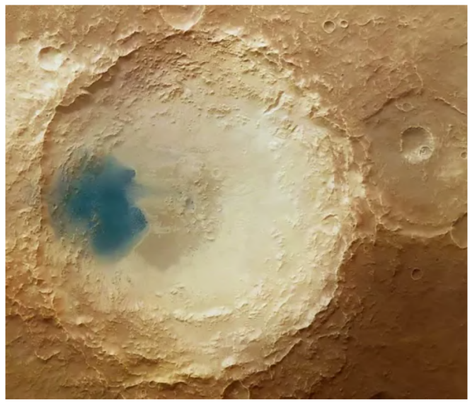


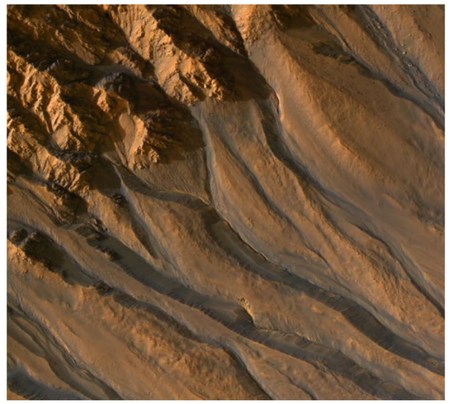
2-Variable

.....

The **high dimensionality** of the weight space makes visualization of the loss surface very difficult.







.....

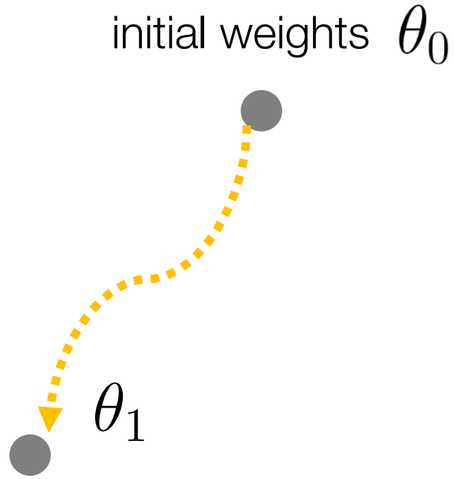
The **computation cost** of high resolution surface visualization is very expensive.

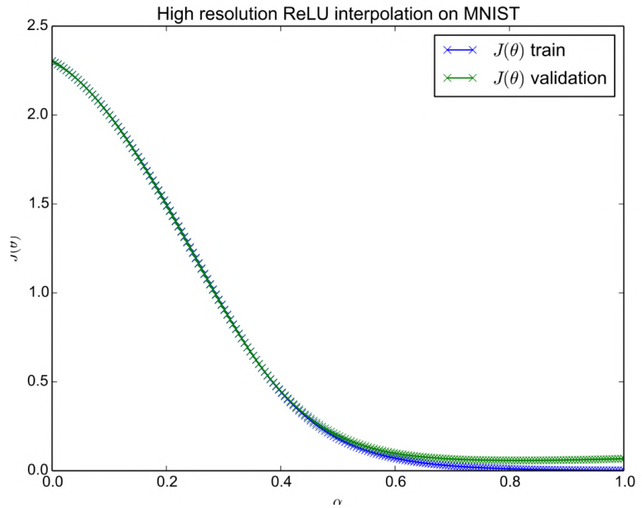
Interpreting High-Dimension Loss Surface

1-D Linear Interpolation [Goodfellow'15]

Given two solutions θ_1, θ_2, and the targeted direction $\theta_2 - \theta_1$:

$$f(\alpha) = L(\theta_1 + \alpha(\theta_2 - \theta_1))$$





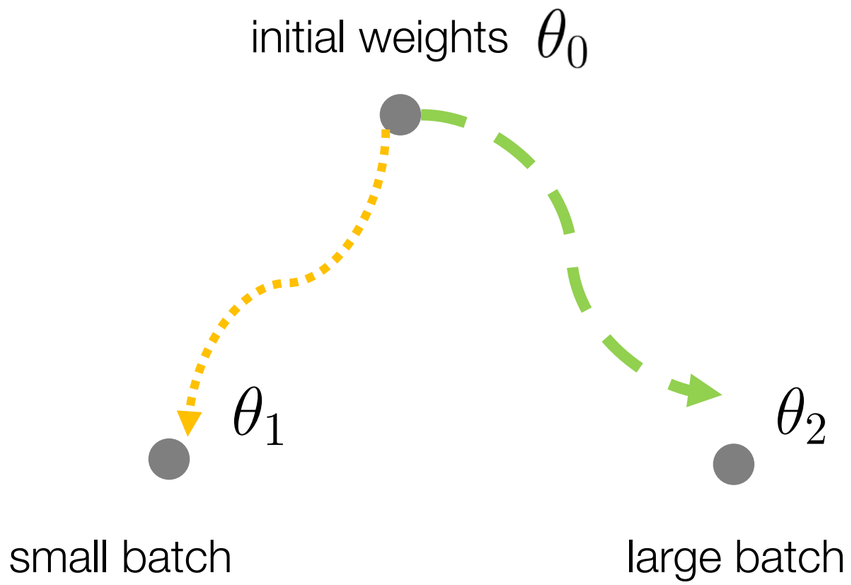
Goodfellow et al, "Qualitatively characterizing neural network optimization problems", ICLR 2015

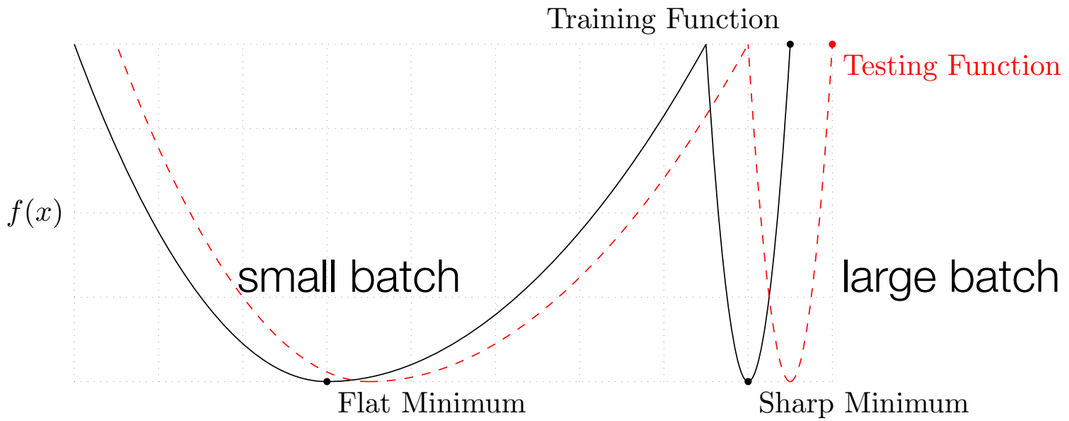
Interpreting High-Dimension Loss Surface

1-D Linear Interpolation [Goodfellow'15]

Given two solutions θ_1, θ_2, and the targeted direction $\theta_2 - \theta_1$:

$$f(\alpha) = L(\theta_1 + \alpha(\theta_2 - \theta_1))$$

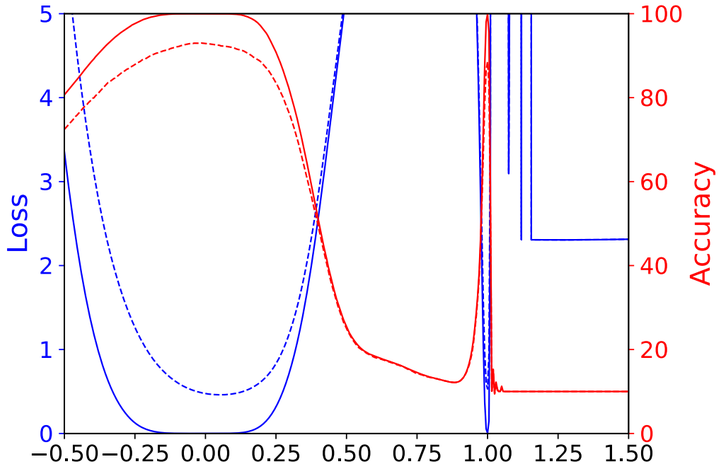




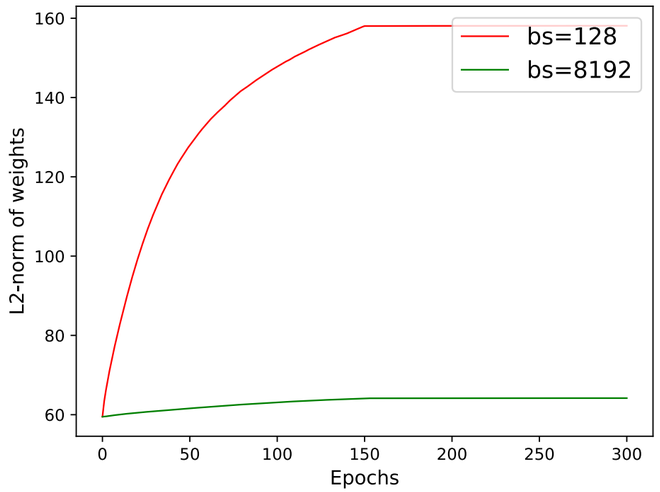
Keskar et al, "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima", ICLR 2017

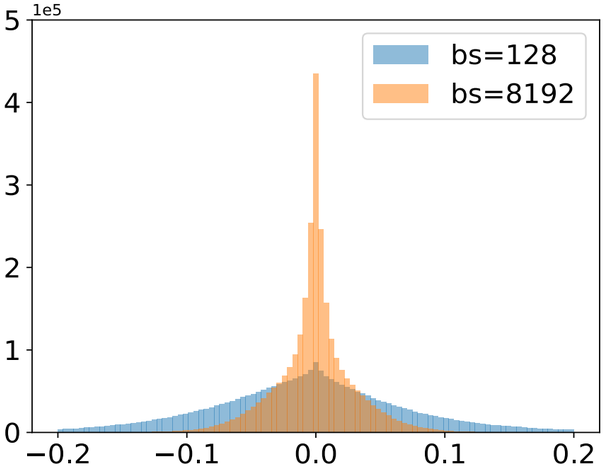
Small-batch vs Large-batch

weight decay = 0.0

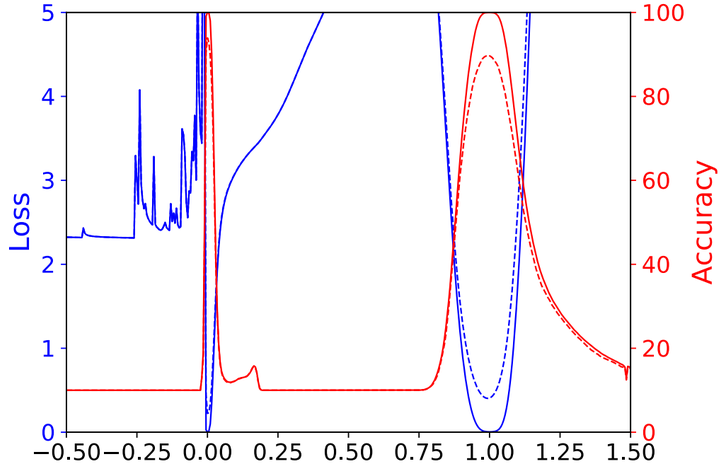


7.37% 11.07%

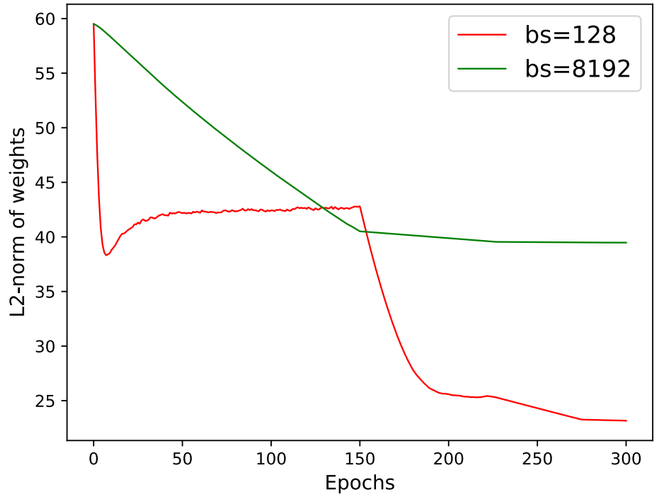


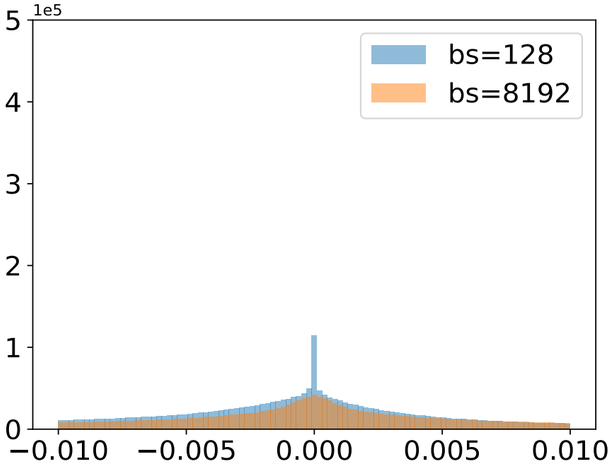


weight decay = 5e-4

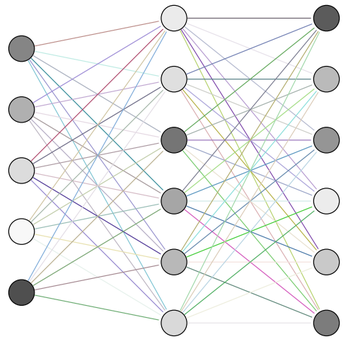


6.00% 10.19%

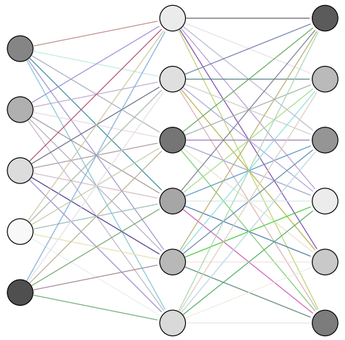




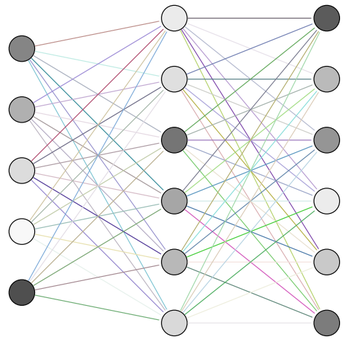
Scale Invariance of NN's Weights



$w_1 \quad w_2$



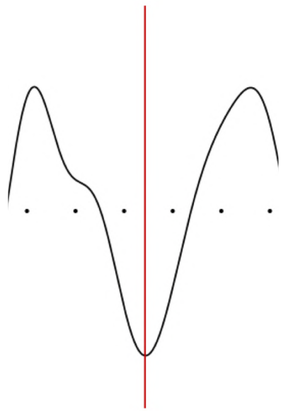
$\alpha w_1 \quad \alpha^{-1} w_2$



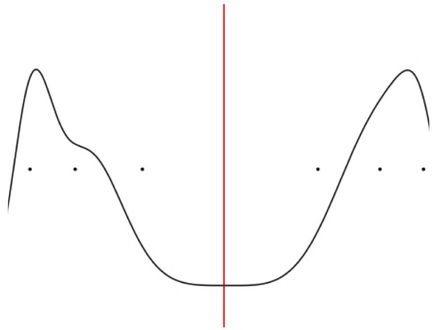
αw_1 BN w_2

The α scale transformation does not affect the generalization as the behavior of the function is identical.

$w_1 + d$



$10w_1 + d$



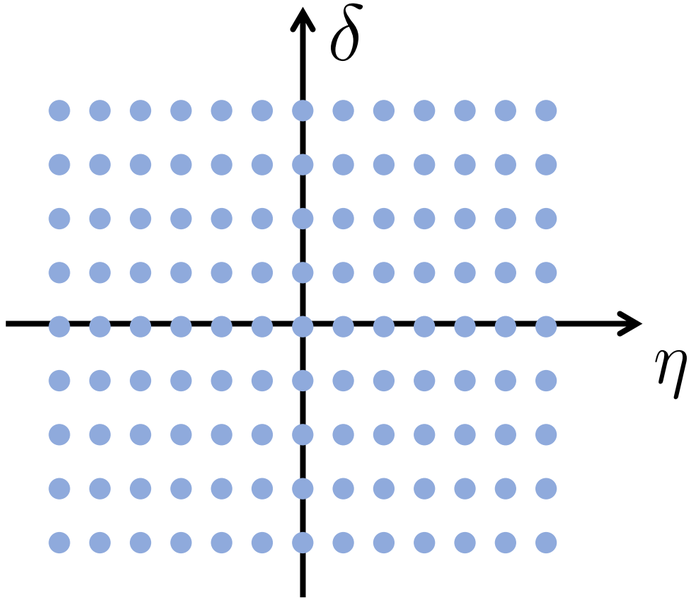
The loss function will change very quickly in the direction of a small filter, and very slowly in the direction of large filter.

Normalized 2D Surface Plotting

1. Create two random directions δ and η

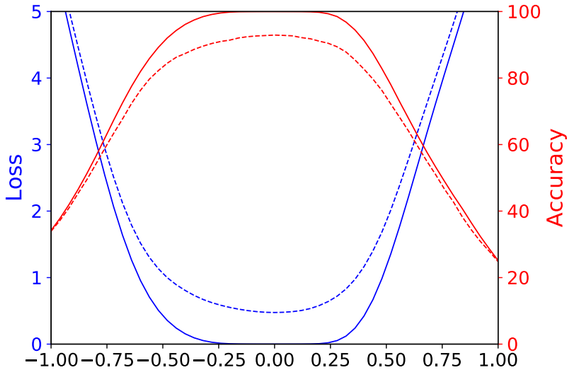
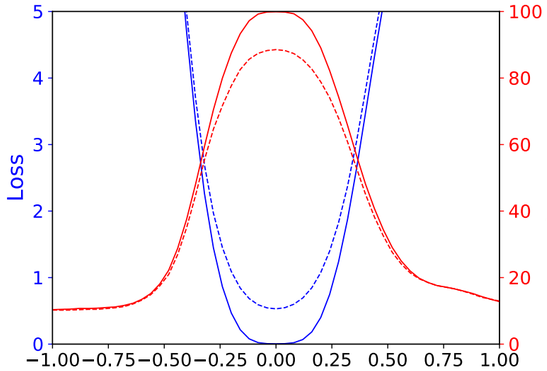
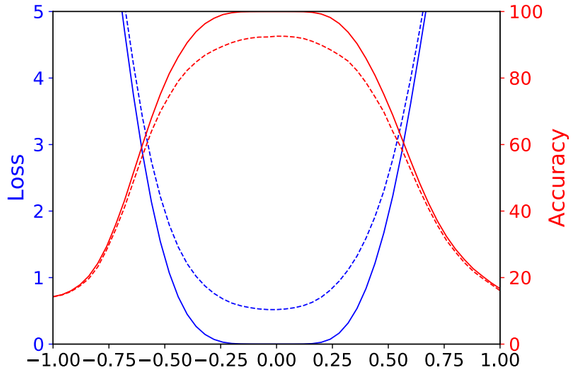
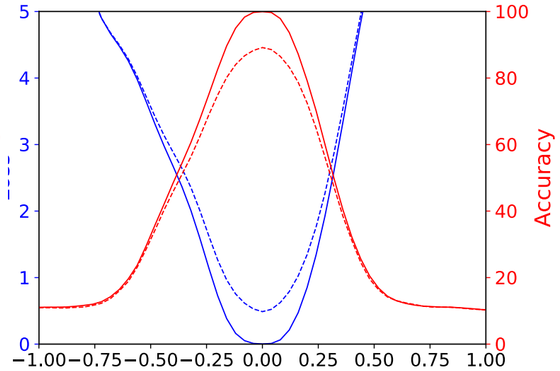
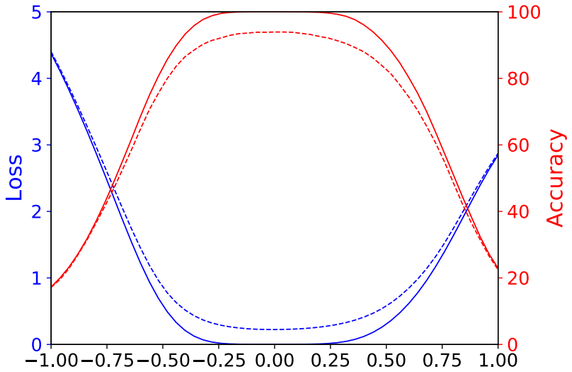
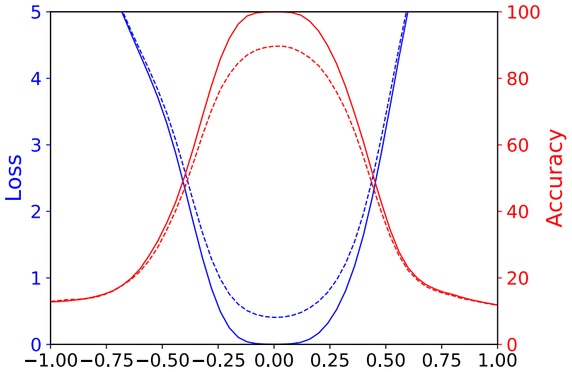
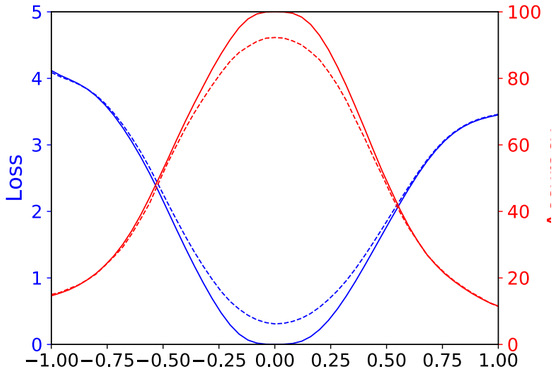
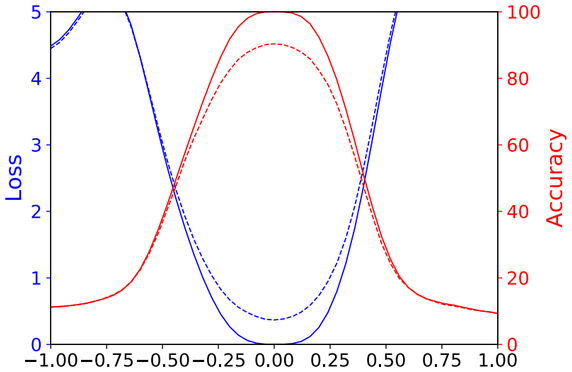
2. Normalize each filter in δ and η to have the same norm of the corresponding filter

$$\delta_i \leftarrow \frac{\delta_i}{\|\delta_i\|}\|\theta_i\| \qquad \eta_i \leftarrow \frac{\eta_i}{\|\eta_i\|}\|\theta_i\|$$

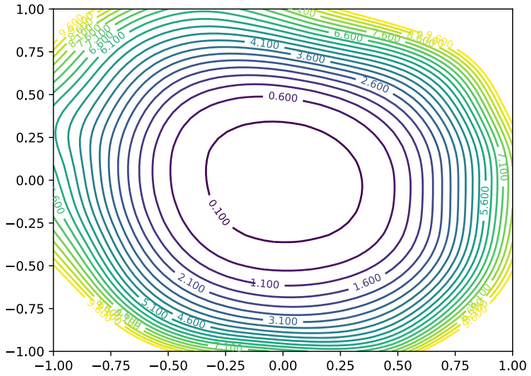
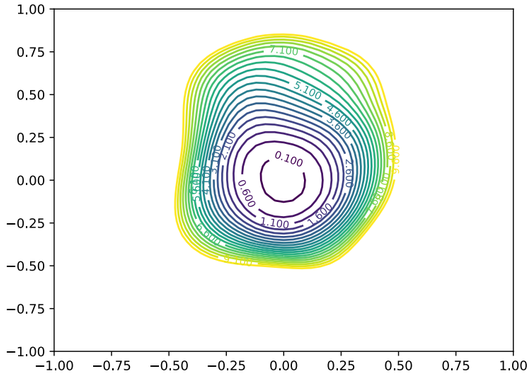
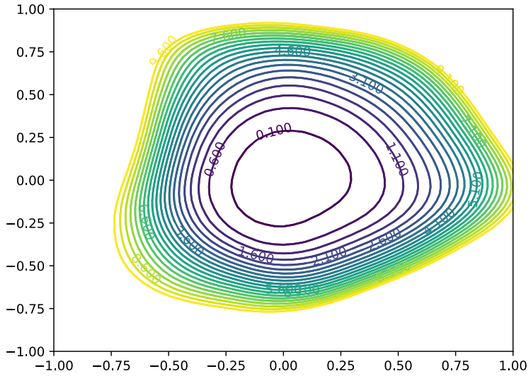
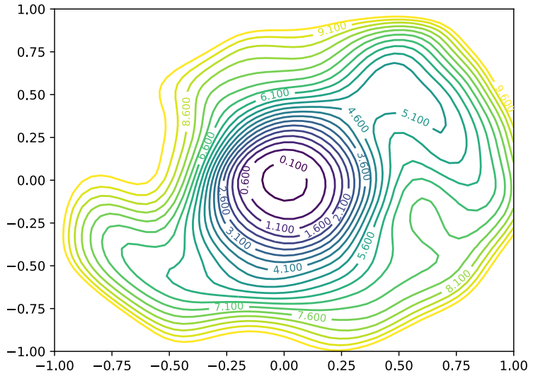
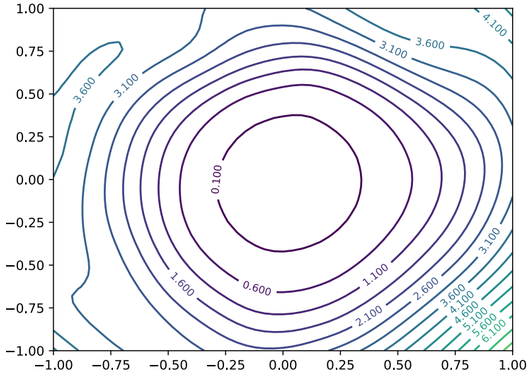
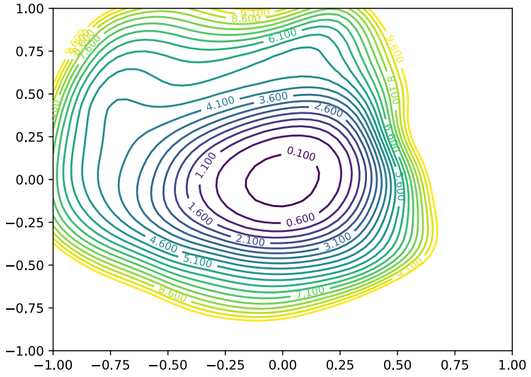
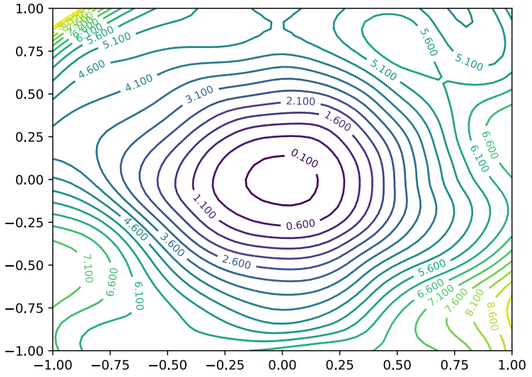
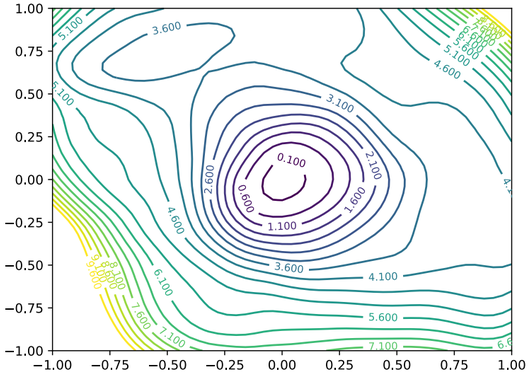


3. Plot the function $f(\alpha, \beta) = L(\theta + \alpha\delta + \beta\eta)$

Normalized 1D Visualization: VGG-9

	SGD, 128	SGD, 8192	Adam, 128	Adam, 8192
WD = 0				
	7.37%	11.07%	7.44%	10.91%
WD = 5e-4				
	6.00%	10.19%	7.80%	9.52%

Normalized 2D Visualization: VGG-9

	SGD, 128	SGD, 8192	Adam, 128	Adam, 8192
WD = 0				
	7.37%	11.07%	7.44%	10.91%
WD = 5e-4				
	6.00%	10.19%	7.80%	9.52%

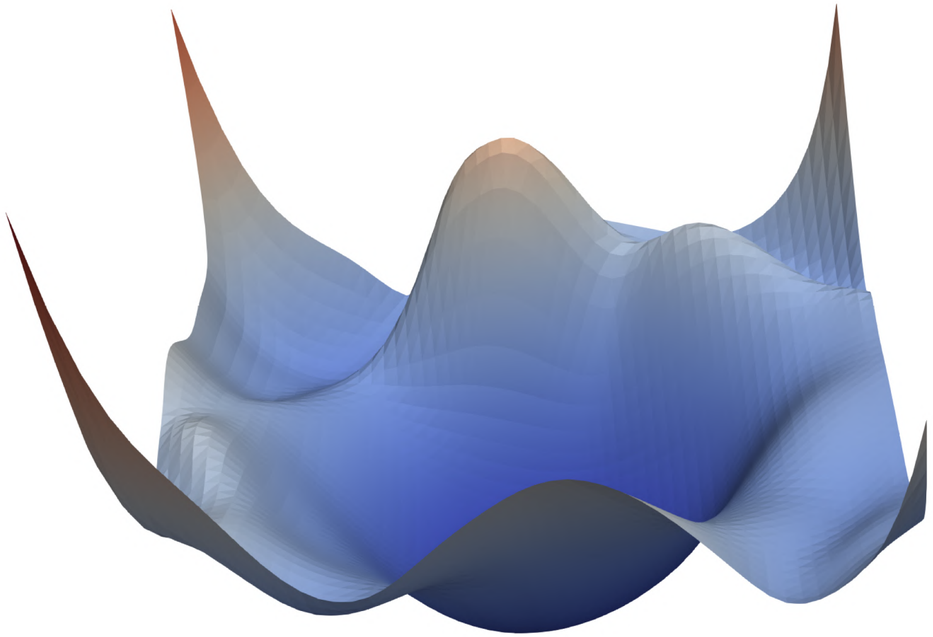
We now know how optimization hyperparamters affect the loss landscape

optimizer, batch size, weight decay….

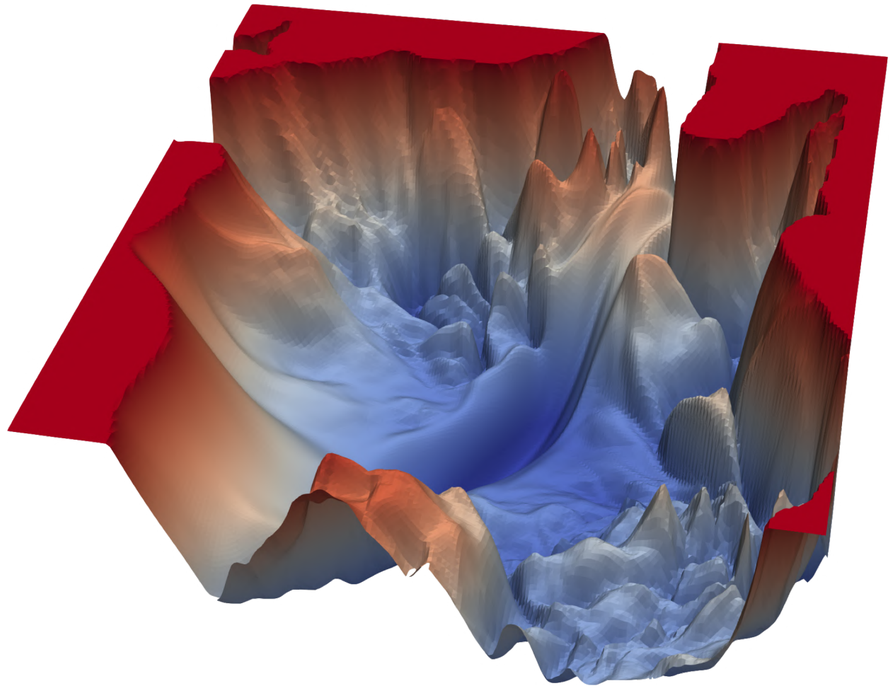
How does the network architecture affect the loss landscape?

depth, width, shortcut connection ….

Effect of Identity Mapping

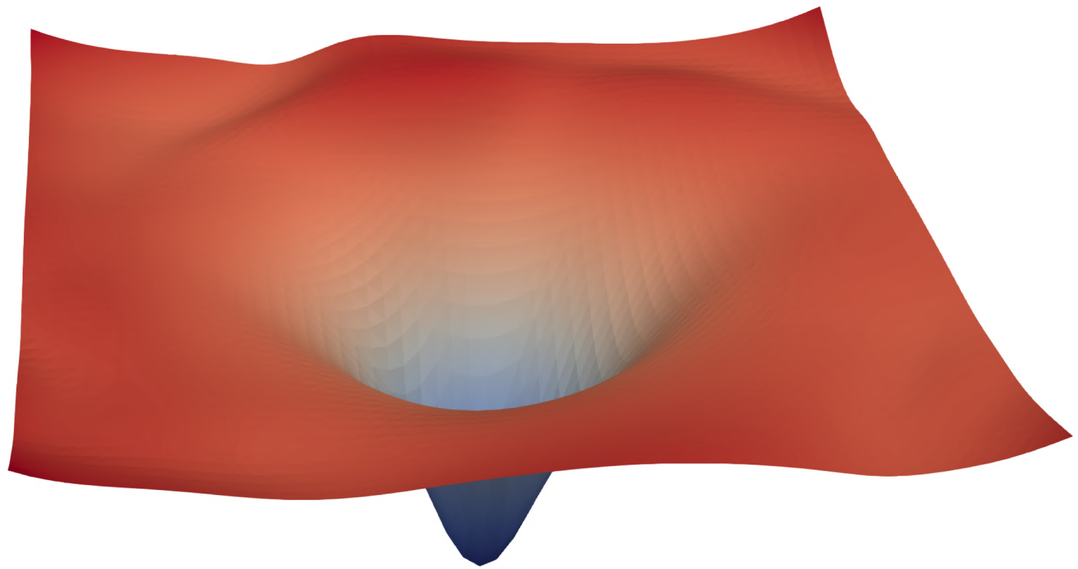


ResNet-56

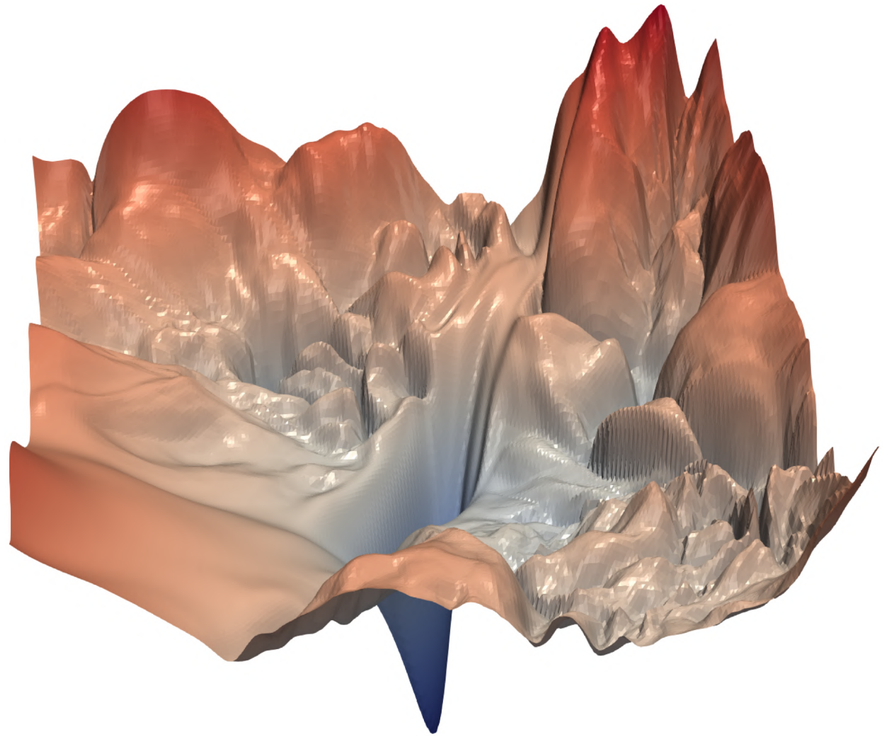


ResNet-56-noshort

Effect of Identity Mapping

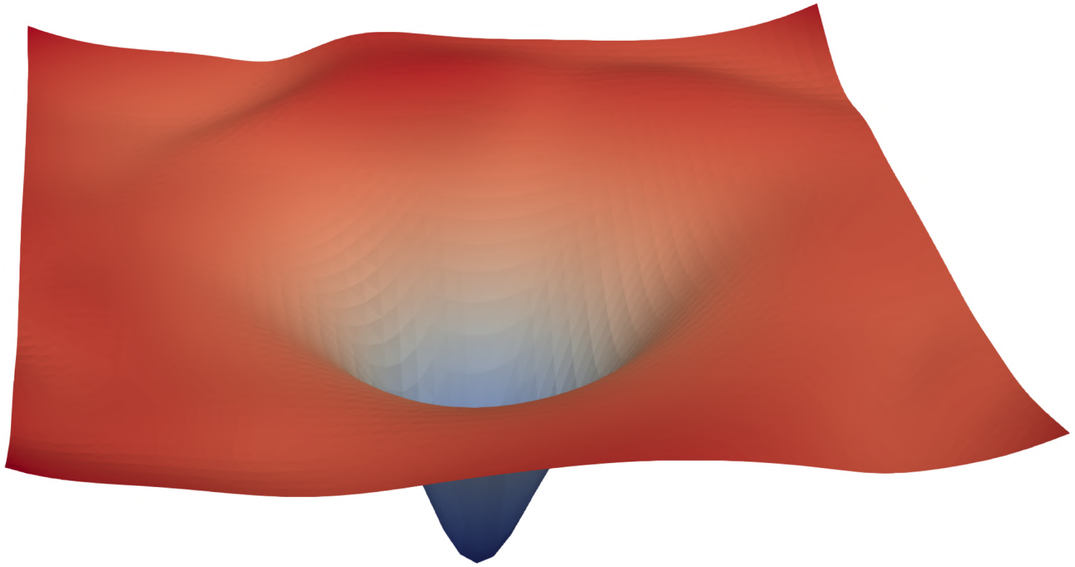


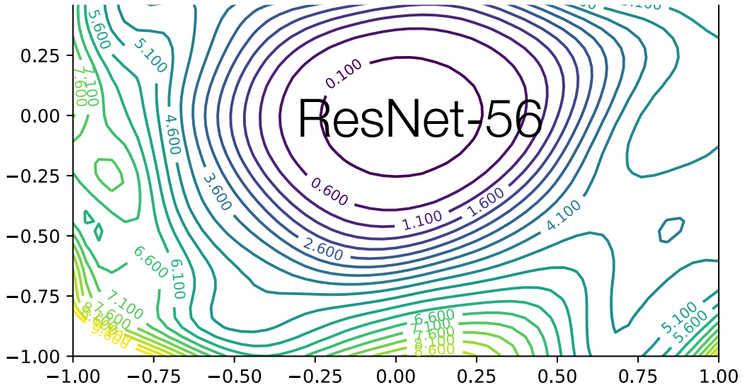
ResNet-56

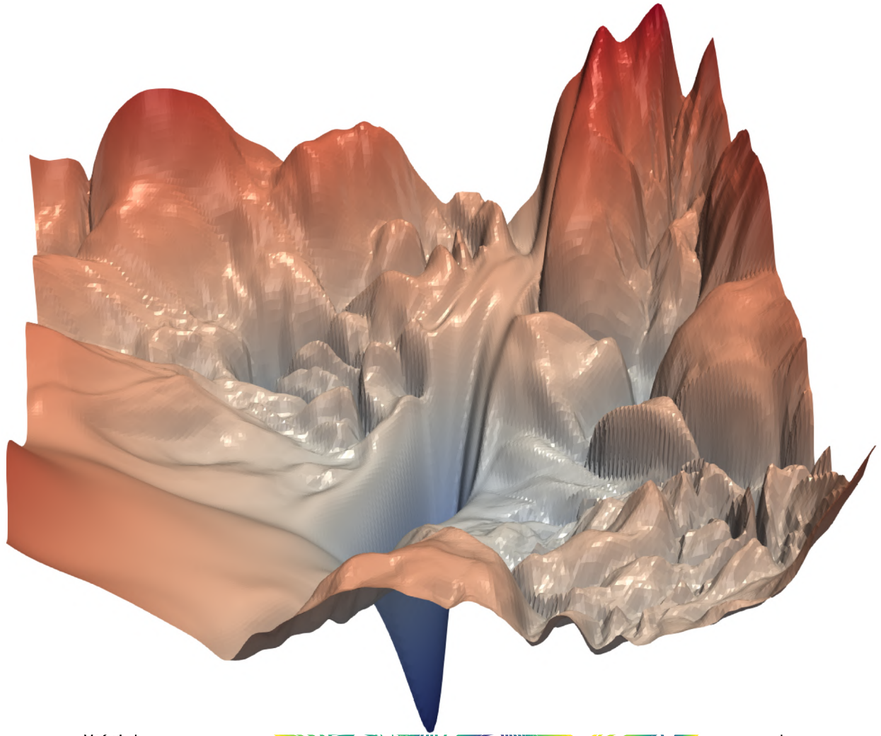


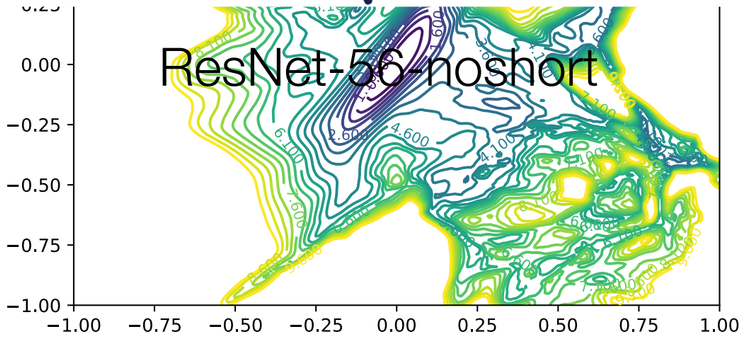
ResNet-56-noshort

Effect of Identity Mapping

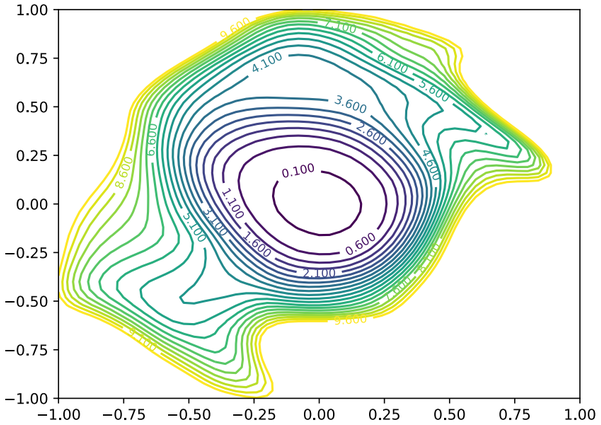
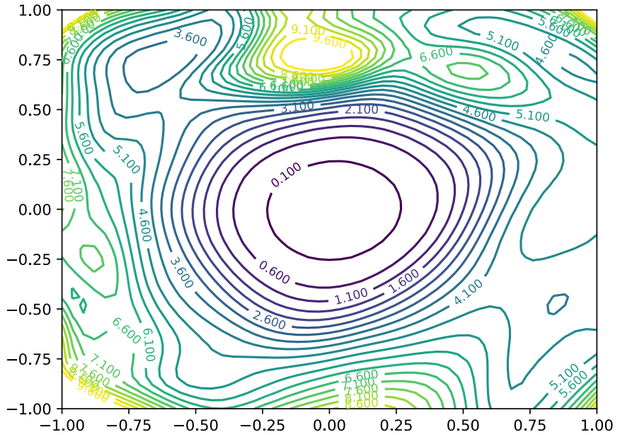
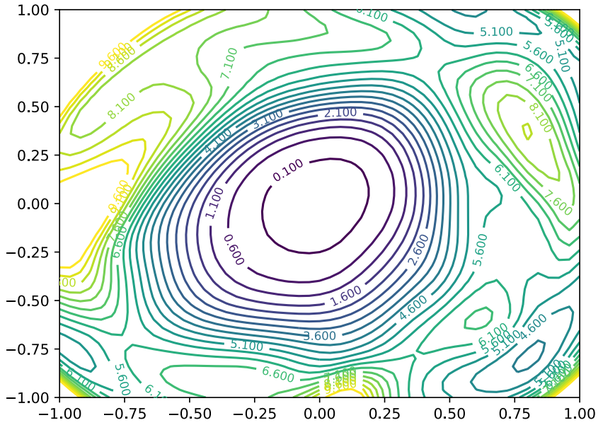
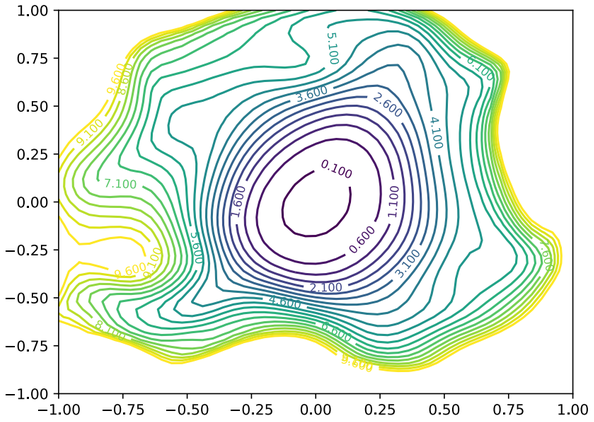
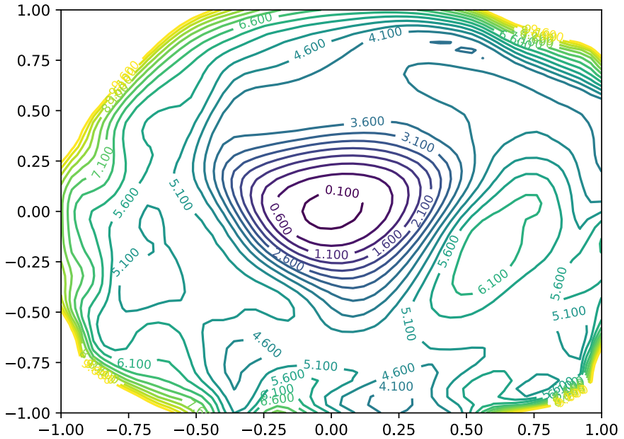
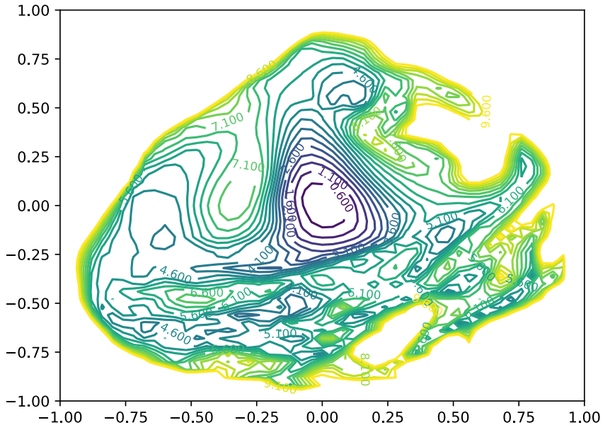




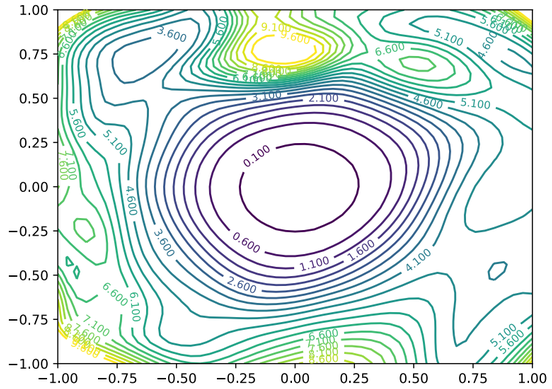
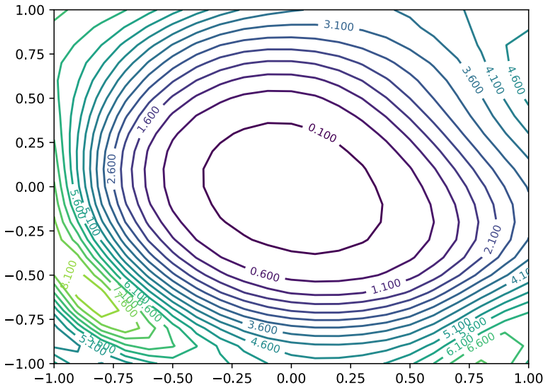
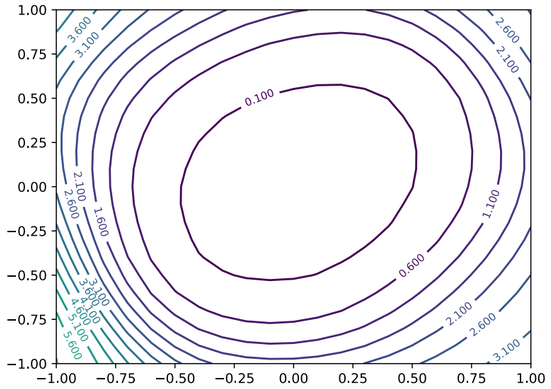
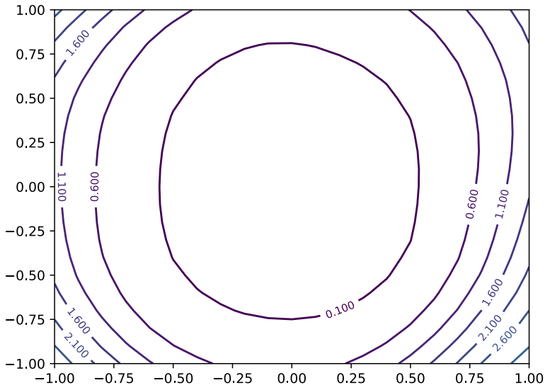
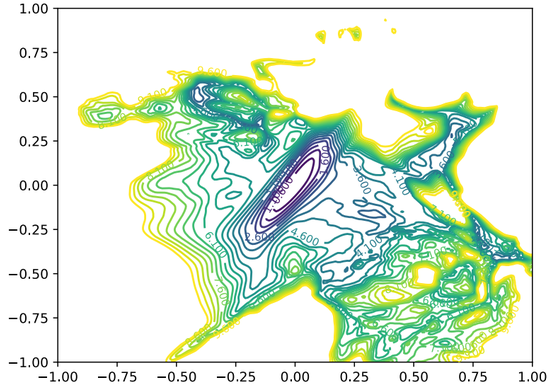
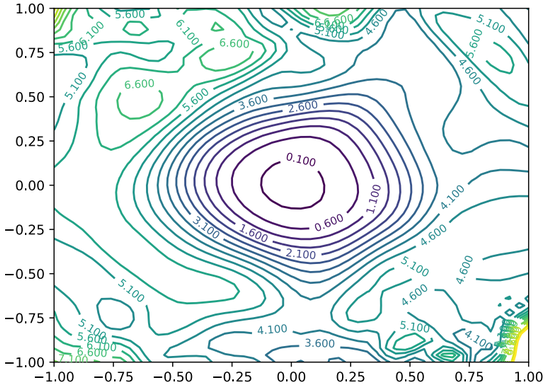
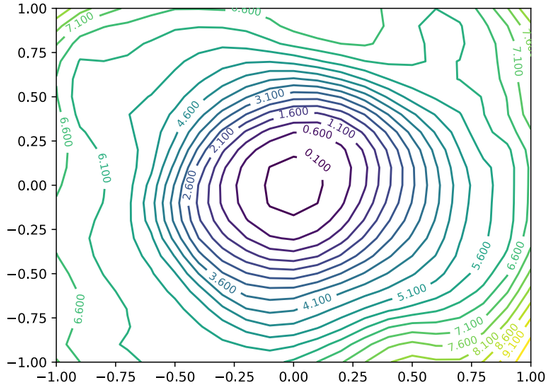
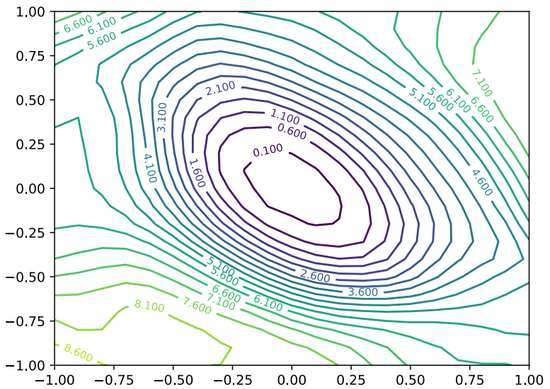




Effect of Network Depth

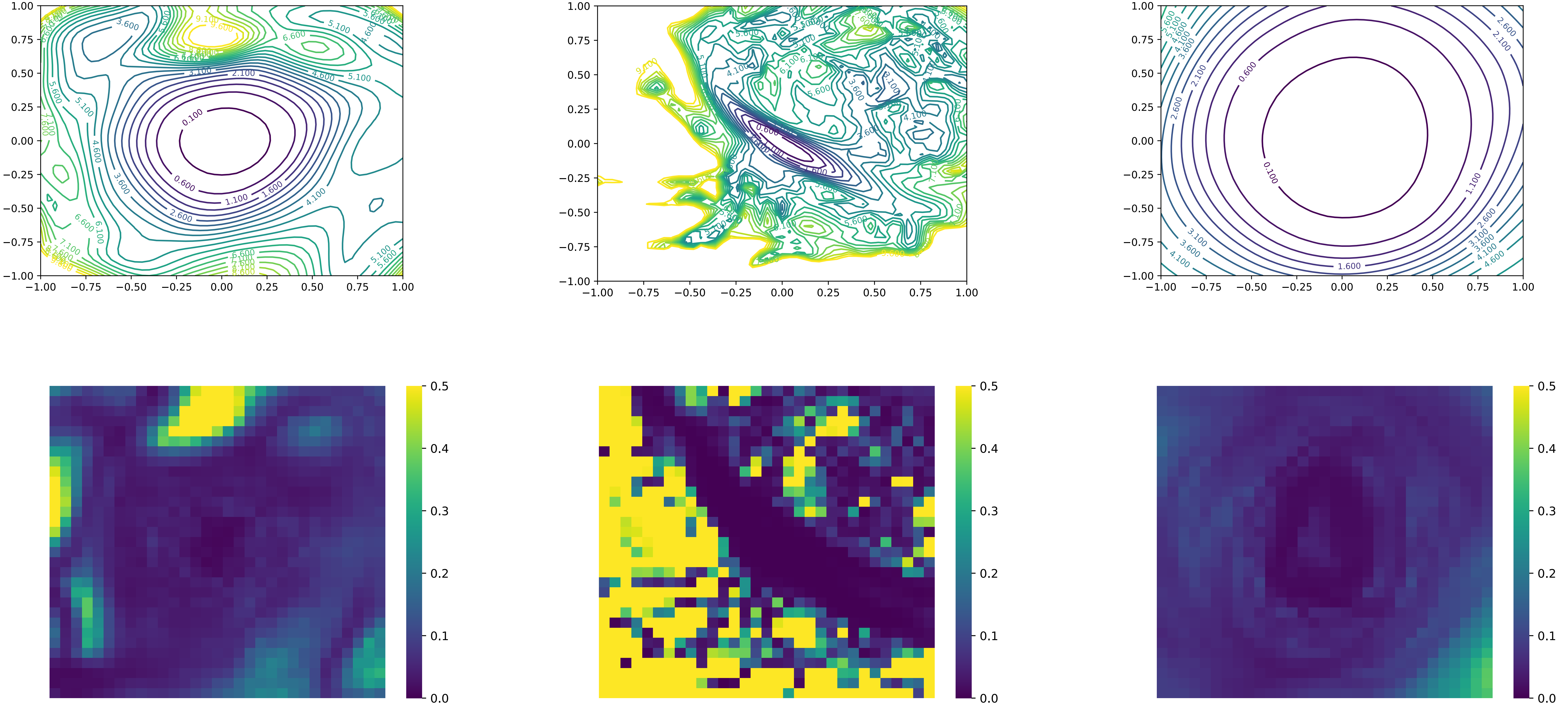
	20 layers	56 layers	110 layers
ResNet			
	7.37%	5.89%	5.79%
VGG-like			
	8.18%	10.83% lr=0.01	16.44% lr=0.01

Effect of Network Width: Wide-ResNet-56-*k*

	$k = 1$	$k = 2$	$k = 4$	$k = 8$
ResNet				
	5.89%	5.07%	4.34%	3.93%
VGG-like				
	13.31%	10.26%	9.69%	8.70%

increased width prevents chaotic behavior, and skip connections dramatically widen minimizers!

Are we really seeing convexity?

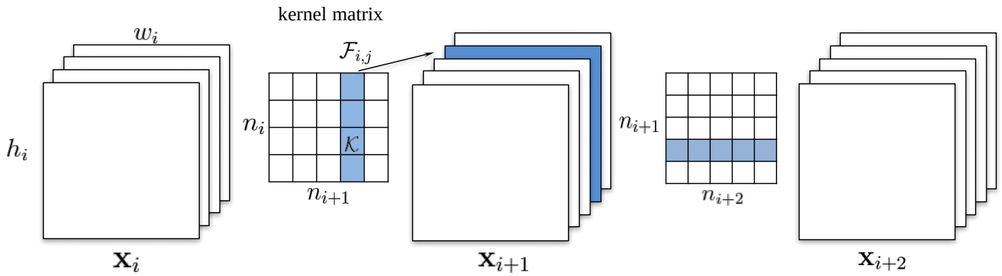


Summary

- The local geometry of landscape has a correlation with the generalization.
- The sharpness of different minimas cannot be compared with 1D linear interpolation due to the weight scale invariance.
- We provide a more accurate visual correlation between flatness and generalization, allowing side-by-side comparison.
- Some neural architectures are easier to minimize than others, such as wide networks, NNs with residual connections.

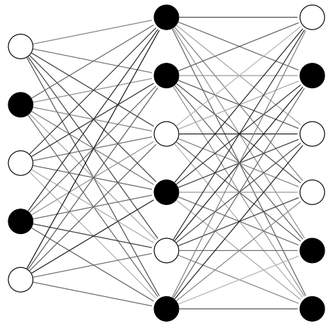
Reducing the Inference Cost

Pruning Filters for Efficient ConvNets, ICLR'17



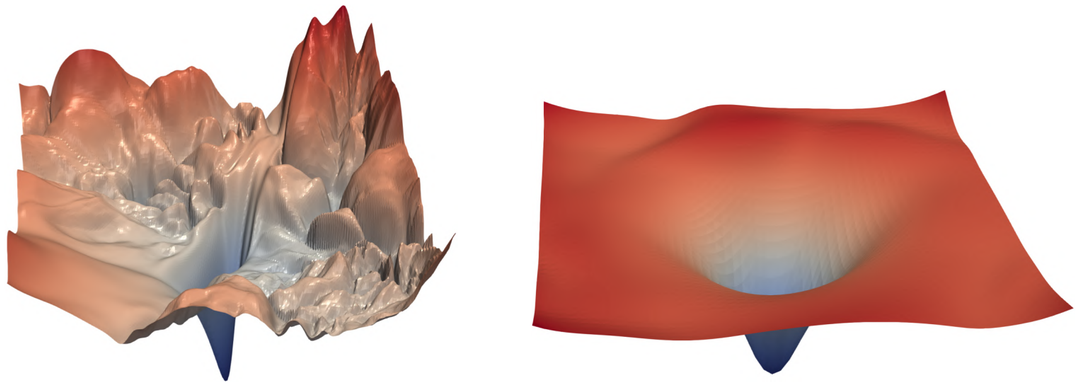
Reducing the Training Cost

Training Quantized Network: A Deeper Understanding, NIPS'17



Understanding the "black box"

Visualizing the Loss Landscape of Neural Nets, arXiv'17



Future Work

Understanding the loss surface of RNN, GAN and RL.

Understanding the generalization ability of neural networks.

Architecture search with good loss landscape.

Optimization methods for training low-precision networks.

Selected Publications

- **Hao Li**, Zheng Xu, Gavin Taylor, Tom Goldstein. *Visualizing the Loss Landscape of Neural Nets*. International Conference on Learning Representations (**ICLR**) Workshop Track, 2018
  in submission to Neural Information Processing Systems (**NIPS)**, 2018

- **Hao Li***, Soham De*, Zheng Xu, Christoph Studer, Hanan Samet, Tom Goldstein. *Training Quantized Nets: A Deeper Understanding*. Neural Information Processing Systems (**NIPS)**, 2017

- **Hao Li**, Asim Kadav, Igor Durdanovic, Hanan Samet, Hans Peter Graf. *Pruning Filters for Efficient ConvNets.* International Conference on Learning Representations (**ICLR**), 2017

- **Hao Li**, Shangfu Peng, Hanan Samet. *Streaming News Image Summarization*. International Conference on Pattern Recognition (**ICPR**), 2016 (Oral)

- **Hao Li**, Asim Kadav, Erik Kruus, Cristian Ungureanu. *MALT: Distributed Data-Parallelism for Existing ML Applications*. ACM European Conference on Computer Systems (**EuroSys**), 2015

Many Thanks!



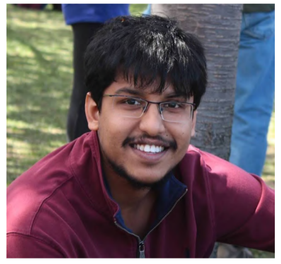


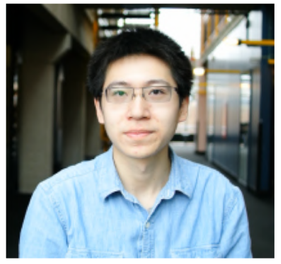









Thanks for listening!

Repeatability of the Loss Surface Visualization

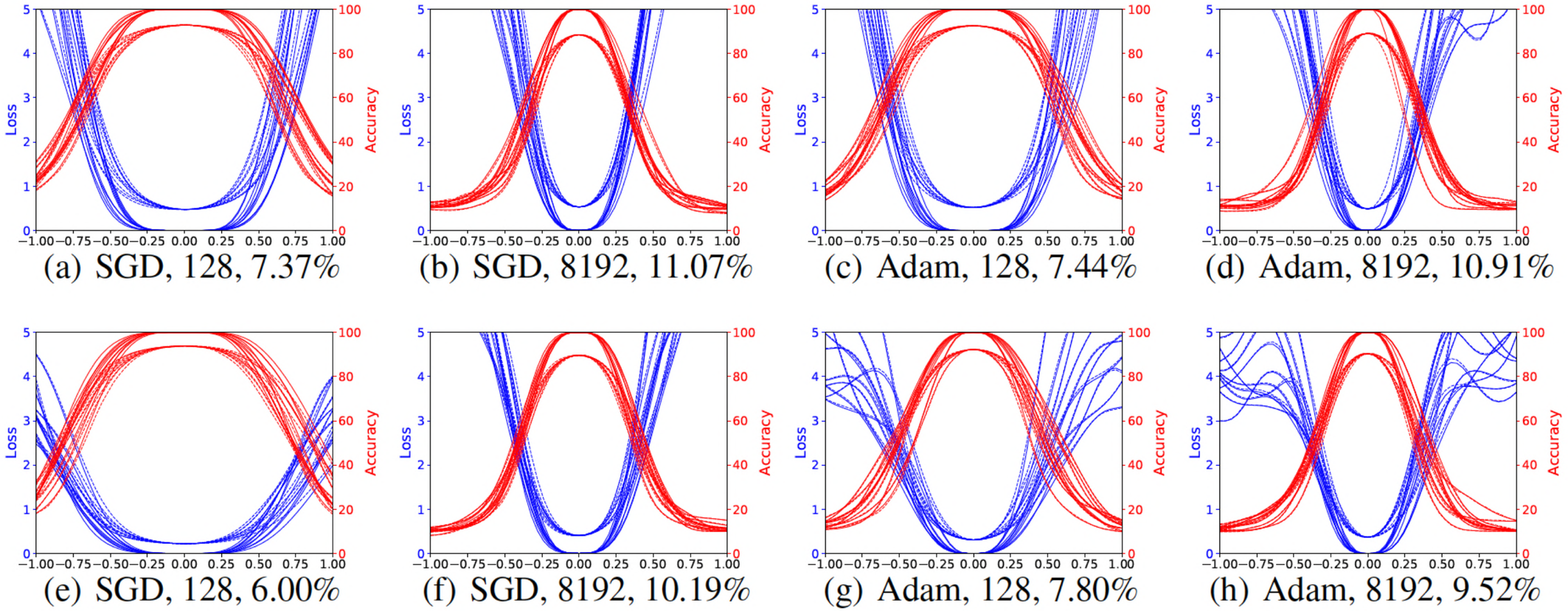


Figure 14: Repeatability of the surface plots for VGG-9 with filter normalization. The shape of minima obtained using 10 different random filter-normalized directions.

Repeatability of the Loss Surface Visualization

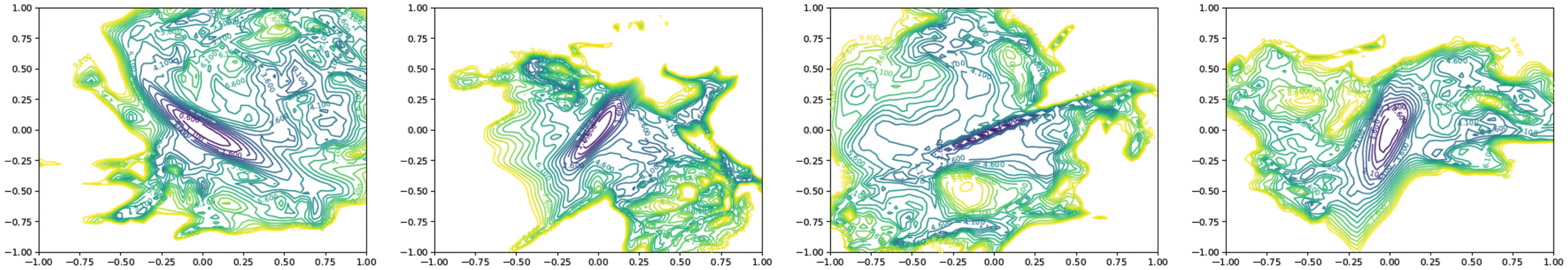
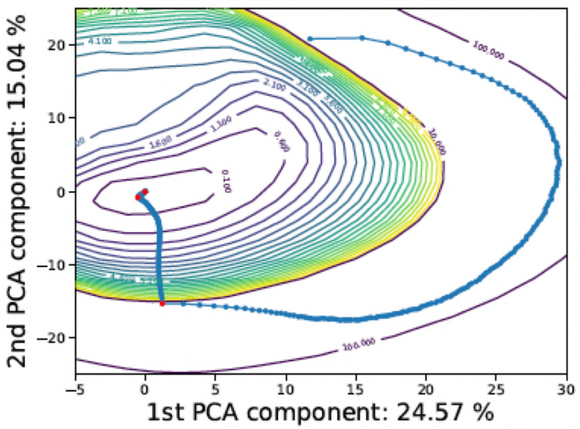
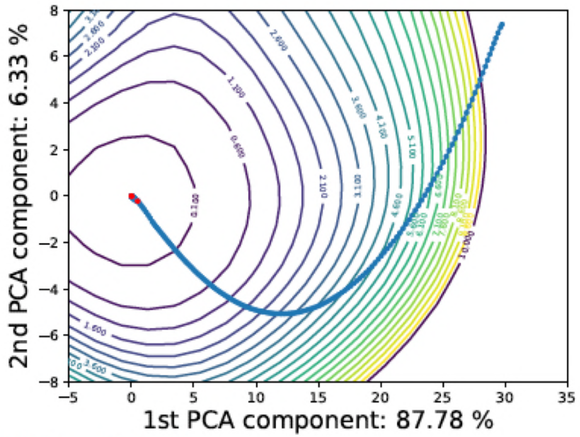


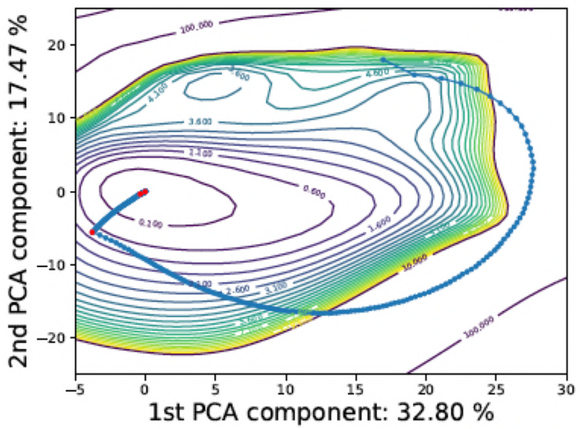
Figure 15: Repeatness of the 2D surface plots for ResNet-56-noshort. The model is trained with batch size 128, initial learning rate 0.1 and weight decay 5e-4. The final training loss is 0.192, the training error is 6.49 and the test error is 13.31.

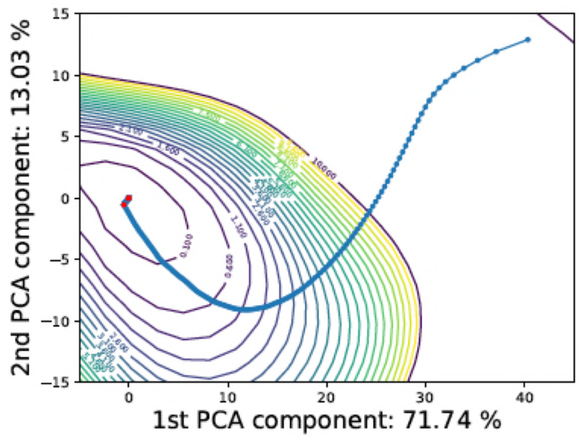
Visualizing the Optimization Path



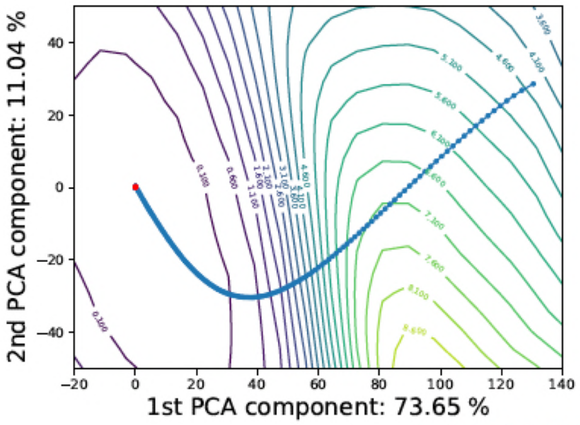


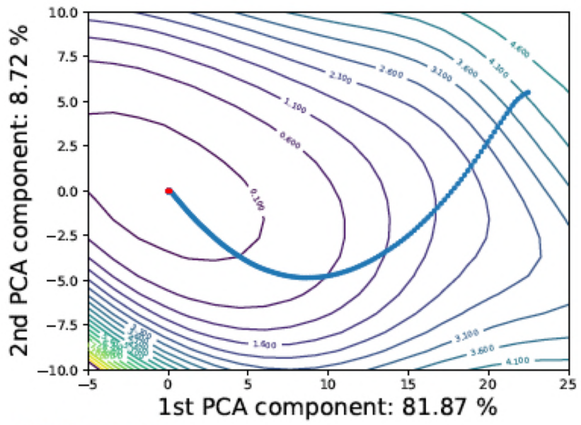
(a) SGD,WD=5e-4



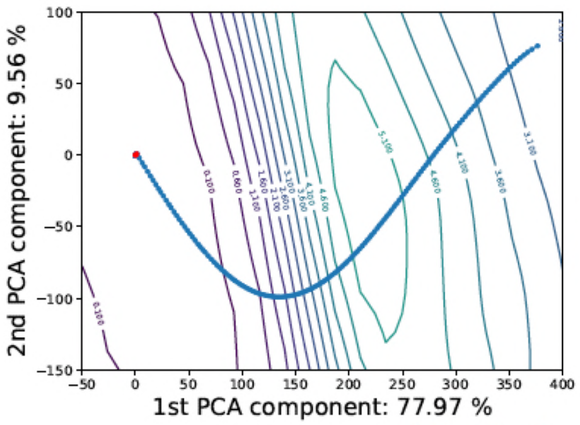


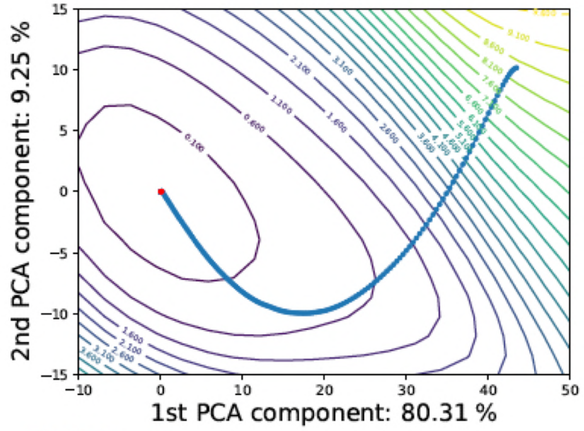
(b) Adam, WD=5e-4





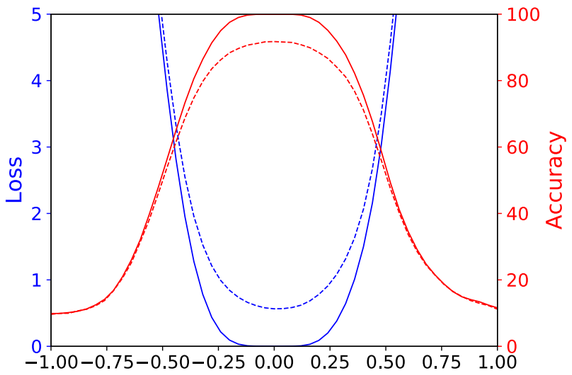
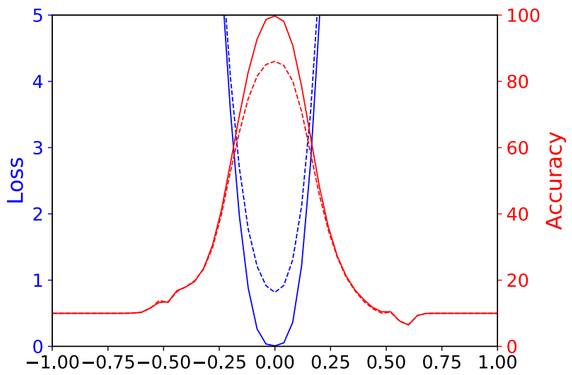
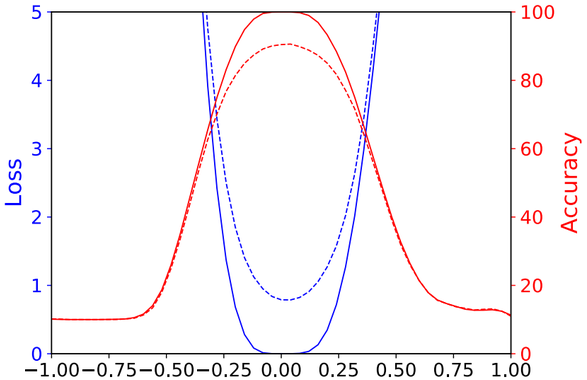
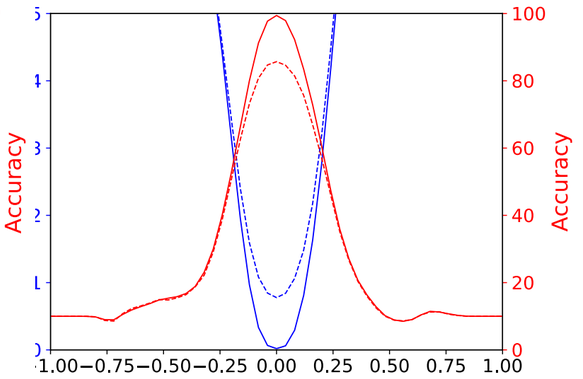
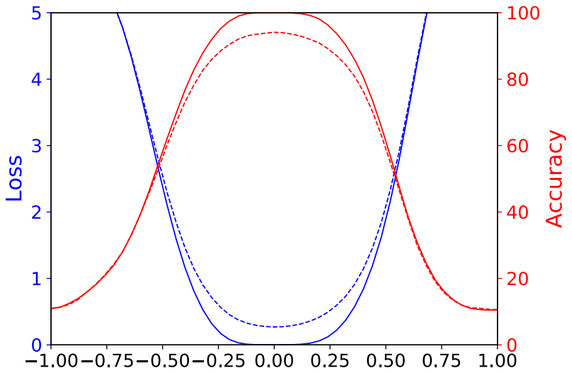
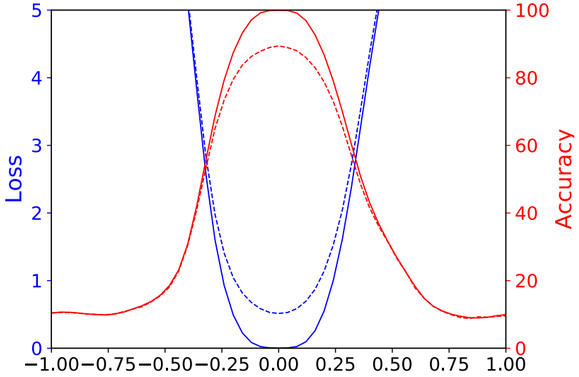
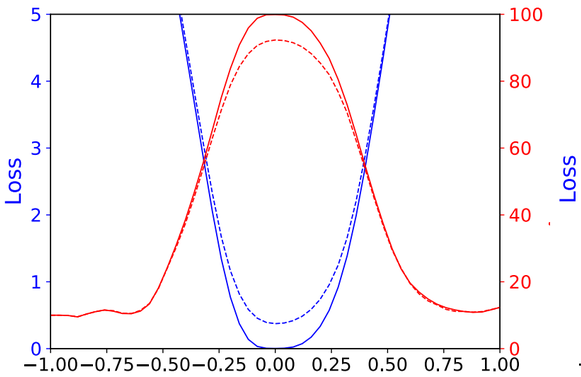
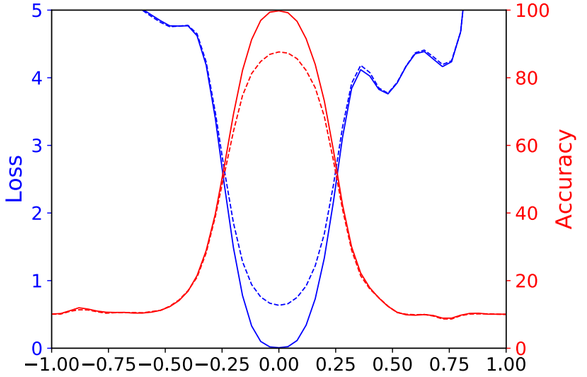
(c) SGD, WD=0



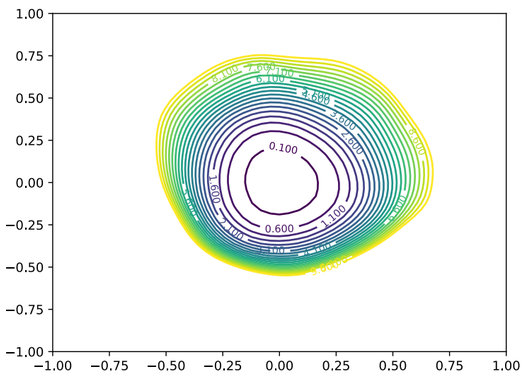
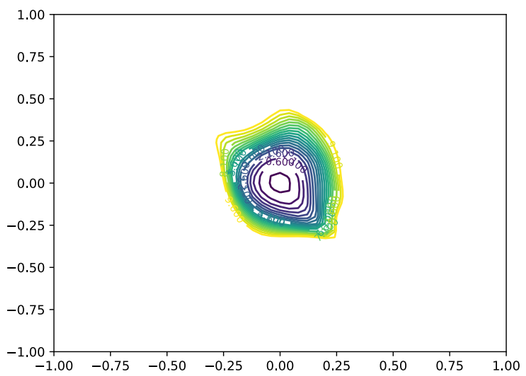
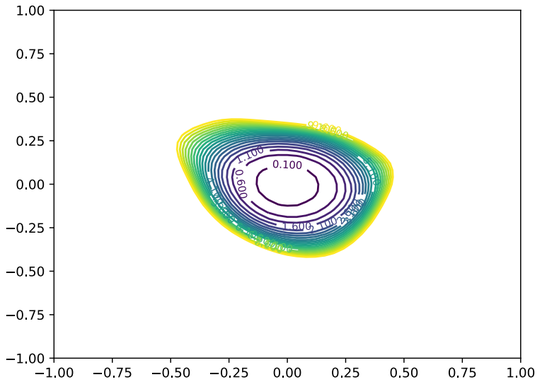
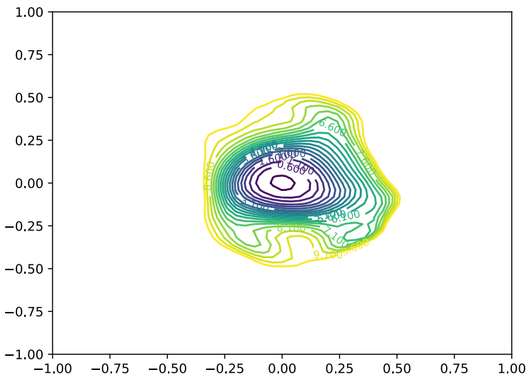
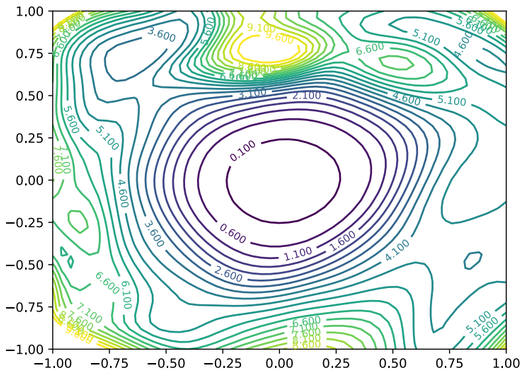
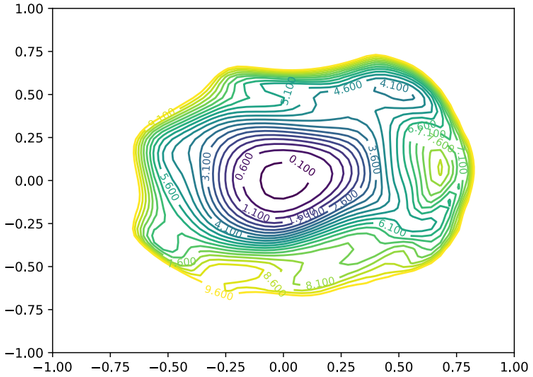
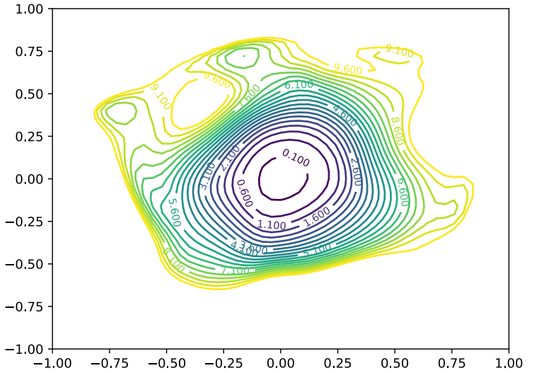
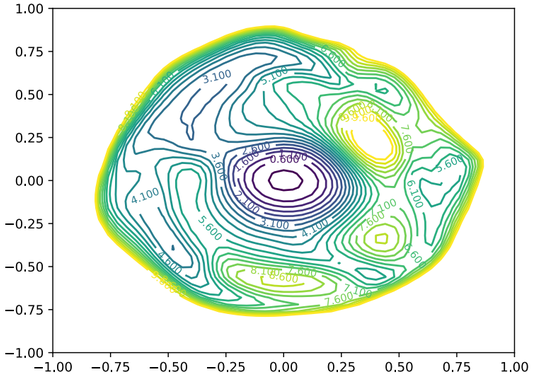


(d) Adam,WD=0

Normalized 1D Visualization: ResNet-56

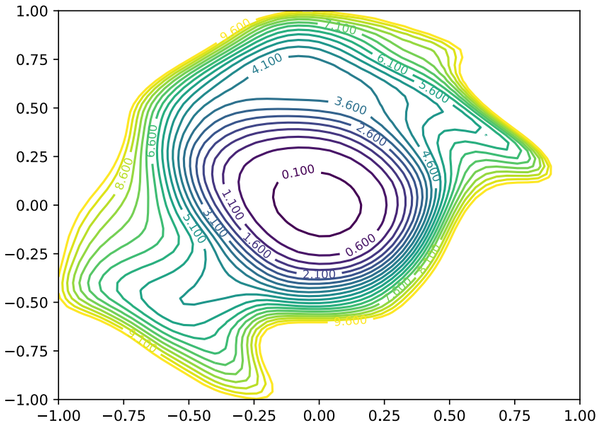
	SGD, 128	SGD, 8192	Adam, 128	Adam, 8192
WD = 0				
	8.26%	13.93%	9.55%	14.30%
WD = 5e-4				
	5.89%	10.59%	7.67%	12.36%

Normalized 2D Visualization: ResNet-56

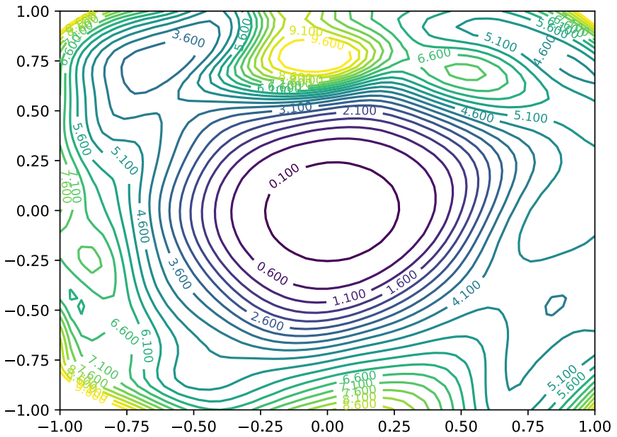
	SGD, 128	SGD, 8192	Adam, 128	Adam, 8192
WD = 0				
	8.26%	13.93%	9.55%	14.30%
WD = 5e-4				
	5.89%	10.59%	7.67%	12.36%

Effect of Network Width

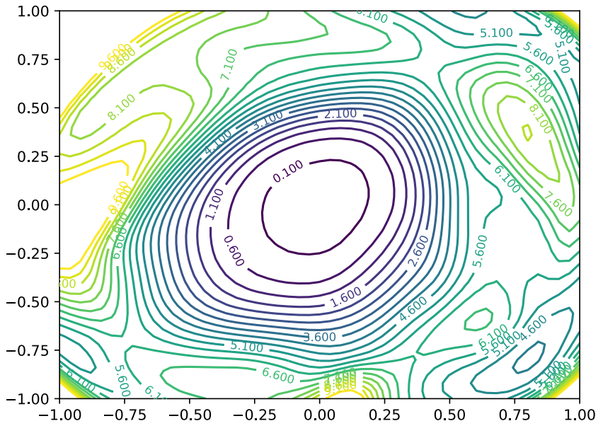
ResNets for CIFAR-10



ResNets-20 7.37%

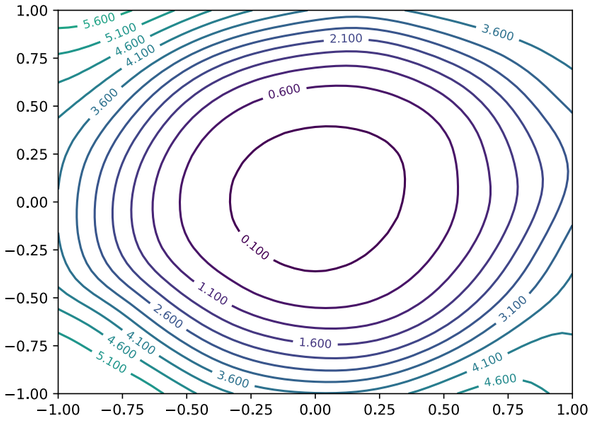


ResNets-56 5.89%

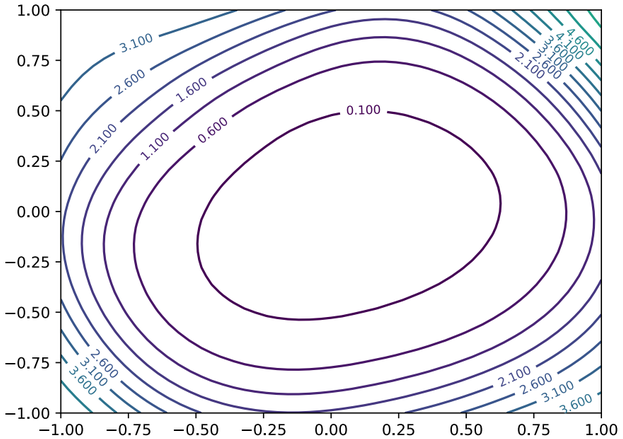


ResNets-110 5.79%

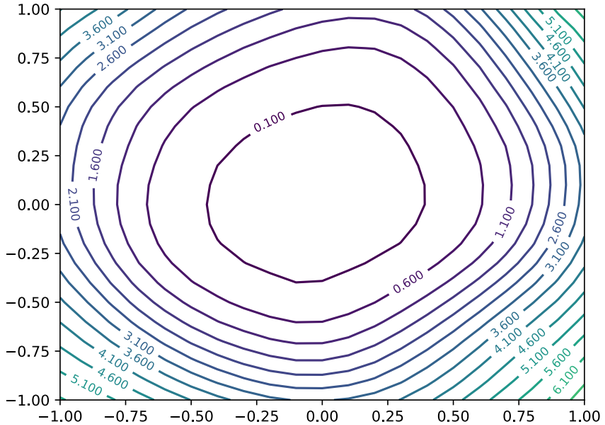
ResNets for ImageNet



ResNets-18 5.42%



ResNets-34 4.73%



ResNets-50 4.55%