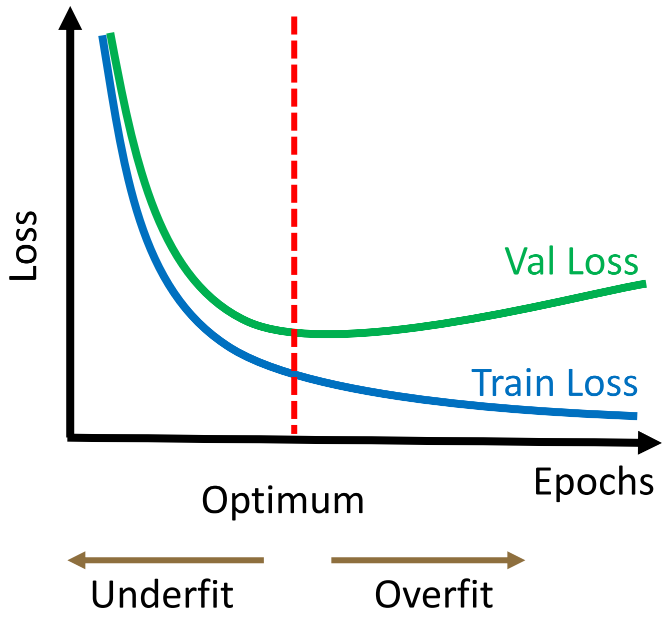
Samples with Low Loss Curvature Improve Data Efficiency

Isha Garg & Kaushik Roy,
Purdue University, USA

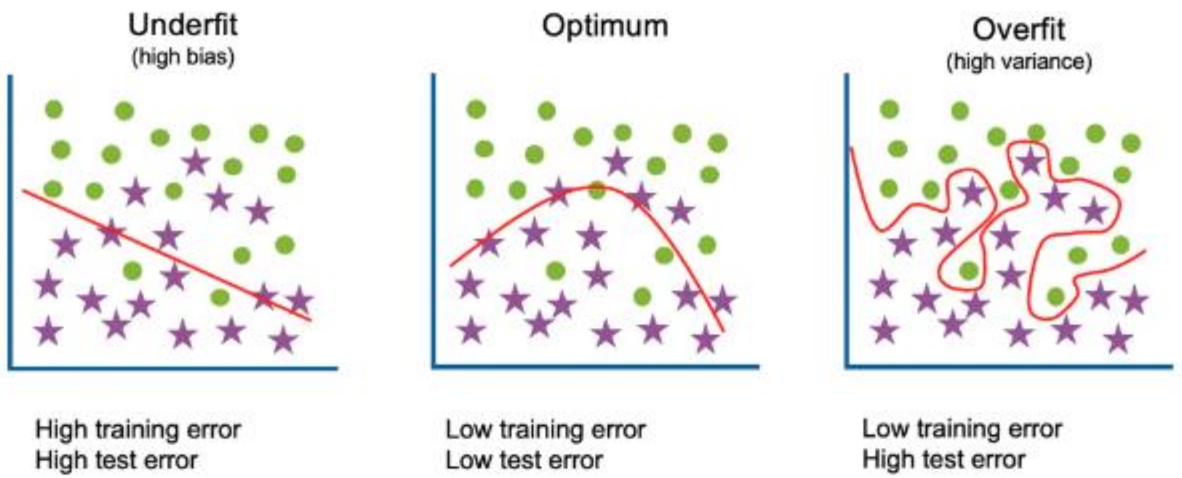
Session: THU-AM-362

Underfitting vs Overfitting

Loss Curves



Loss Boundary in Input Space



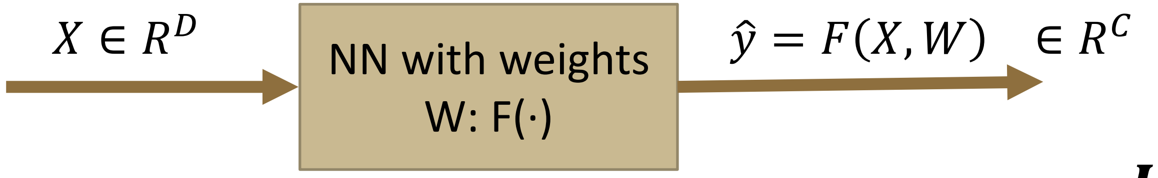
What property sets these three boundaries apart?

Curvature of the loss with respect to the data

https://www.ibm.com/cloud/learn/overfitting

How is Curvature Calculated?

Curvature is determined by the eigenvalues of the Hessian Matrix H(X) [1]



$$\boldsymbol{L(X) = CE(y, \hat{y})}$$

$$H(X) = \Delta_{\mathrm{x}}^2 L = \begin{bmatrix} \frac{\delta^2 L}{\delta x_1 \delta x_1} & \cdots & \frac{\delta^2 L}{\delta x_1 \delta x_D} \\ \vdots & \ddots & \vdots \\ \frac{\delta^2 L}{\delta x_D \delta x_1} & \cdots & \frac{\delta^2 L}{\delta x_D \delta x_D} \end{bmatrix} \in R^{D \times D}$$

- Curvature determined by eigenvalues
- However, calculating the eigendecomposition of this matrix is very expensive.

[1] Keskar et al., On large batch training for deep learning, ICLR 2017
[2] Dinh et al., Sharp minima can generalize for deep nets, ICML 2017

Measuring Eigenvalues: Trace of H

- Tr(H) = sum of all eigenvalues

$$Tr(H) = \sum \lambda_i$$

- Low complexity Hutchinson's trace estimator [1]

$$Tr(H) = E_v\ [v^T H v]$$

where v are random Rademacher variables, i.e. $v \in \{\pm 1\}^D$

- We want $\sum|\lambda_i|$ since we do not care about definiteness

$$Tr(H^2) = \sum \lambda_i^2$$

$$\begin{aligned} Tr(H^2) &= E_v\ [v^T H^2 v] \\ &= E_v[v^T H^T H v] \\ &= E_v[(Hv)^T (Hv)] \\ &= E_v\ \|Hv\|_2^2 \end{aligned}$$

Curvature captured by eigenvalues of Hessian

Eigenvalues captured by Tr(H);

Magnitude captured by Tr(H^2)

Hutchinson's Trace Estimator: reduces Tr(H) to many Hessian-vector products

[1] M.F. Hutchinson (1989) A Stochastic Estimator of the Trace of the Influence Matrix for Laplacian Smoothing Splines, Communications in Statistics - Simulation and Computation

Calculating Trace Efficiently

$$Tr(H^2) = E_v \|Hv\|_2^2$$

- We only need the Hessian-Vector Product, and do not need to calculate the full hessian to get its trace
- The Hessian-vector product can be efficiently calculated from finite step approximation

$$Hv = \frac{1}{h} [\, \Delta_{\mathrm{x}}\, L(X + \mathrm{h}v) - \Delta_X\, L(X) \,]$$

$$= \frac{1}{h} \Delta_X [\, L(X + \mathrm{h}v) - L(X) \,]$$

$$Tr(H^2) \propto E_v \| \Delta_X [\, L(X + \mathrm{h}v) - L(X) \,] \|$$

Curvature captured by eigenvalues of Hessian

Eigenvalues captured by Tr(H);

Magnitude captured by Tr(H^2)

Hutchinson's Trace Estimator: reduces Tr(H) to many Hessian-vector products

Finite Step approximation: efficiently calculate Hessian-vector product

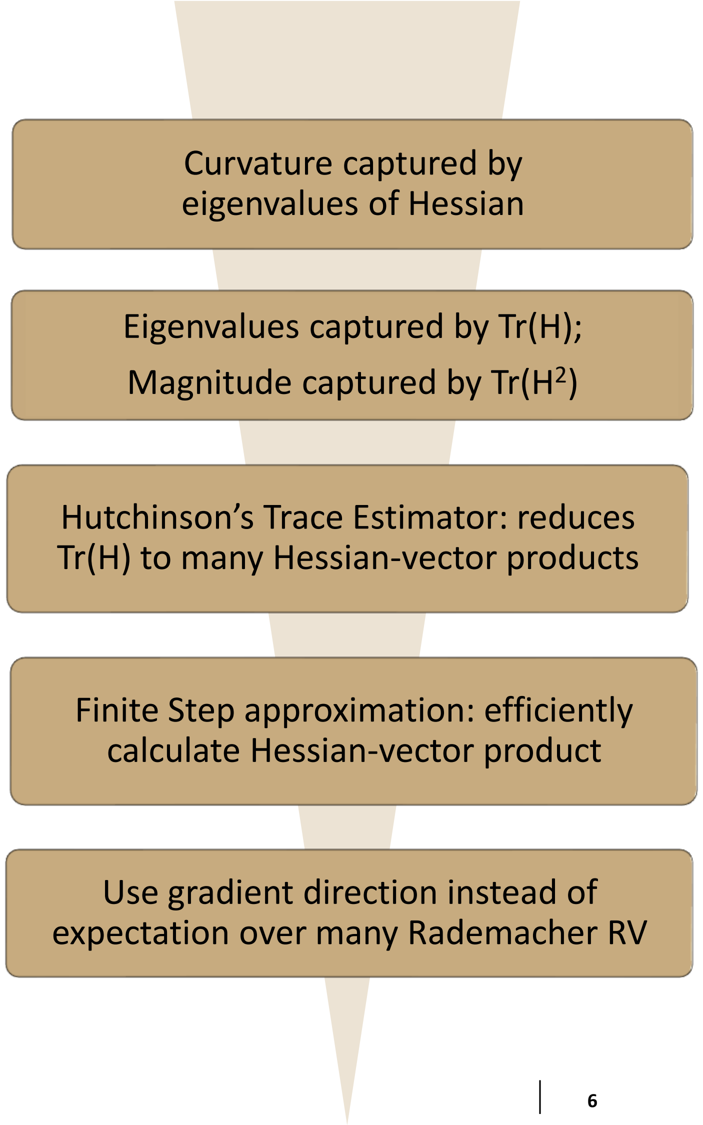
Gradient Direction instead of Rademacher RV

$$Tr(H^2) \propto E_v \| \Delta_X [\, L(X + \mathrm{h}v) - L(X) \,] \|$$

- Instead of taking expectation over multiple random directions, we only consider the gradient direction since previous works [1,2,3] have shown that the direction of maximum change is the gradient direction
- Approximation: choose v to be gradient direction sign instead of Rademacher RV

$$Curv(X) = \| \Delta_X \, [L(X + \mathrm{h}v) - L(X)] \, \|$$

where $v = \dfrac{sign\,(\Delta_X L(X))}{\|sign(\Delta_X \, L(X))\|}$

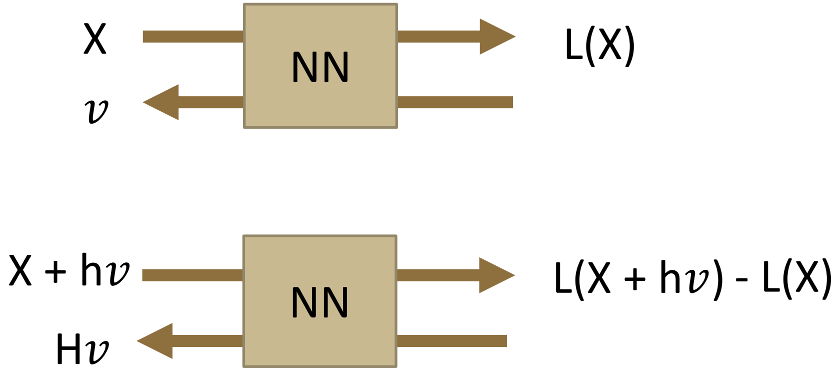


[1] Moosavi-Dezfooli et al,. "Robustness via curvature regularization, and vice versa." *CVPR* 2019
[2] Moosavi-Dezfooli et al., "Empirical Study of the Topology and Geometry of Deep Networks,"CVPR 2018
[3] Jetley, Saumya et al., "With friends like these, who needs adversaries?." NeurIPS 2018

Final Form of Curvature Estimator

$$Curv(X) = \| \Delta_X \, [L(X + \mathrm{h}v) \; - \; L(X)] \, \|$$

where $v = \frac{sign\,(\Delta_X L(X))}{\|sign(\Delta_X \, L(X))\|}$

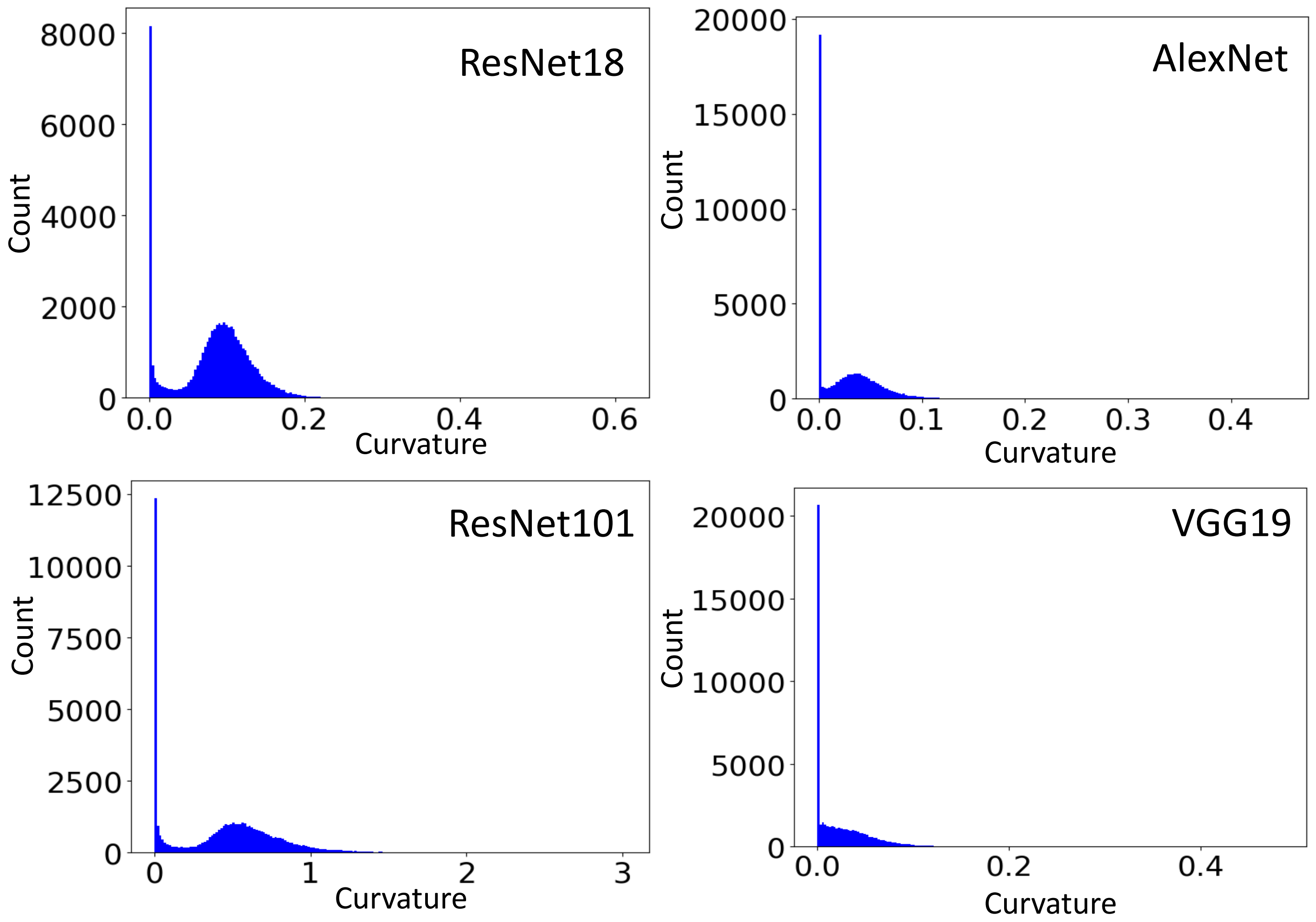


- The cost of performing this is two extra backward passes per sample, and can be parallelized for a batch
- Hyperparameter h chosen to match per pixel changes in L_{inf} adversarial attacks

CURVATURE CALCULATION RESULTS

Curvature Across Architectures: CIFAR10

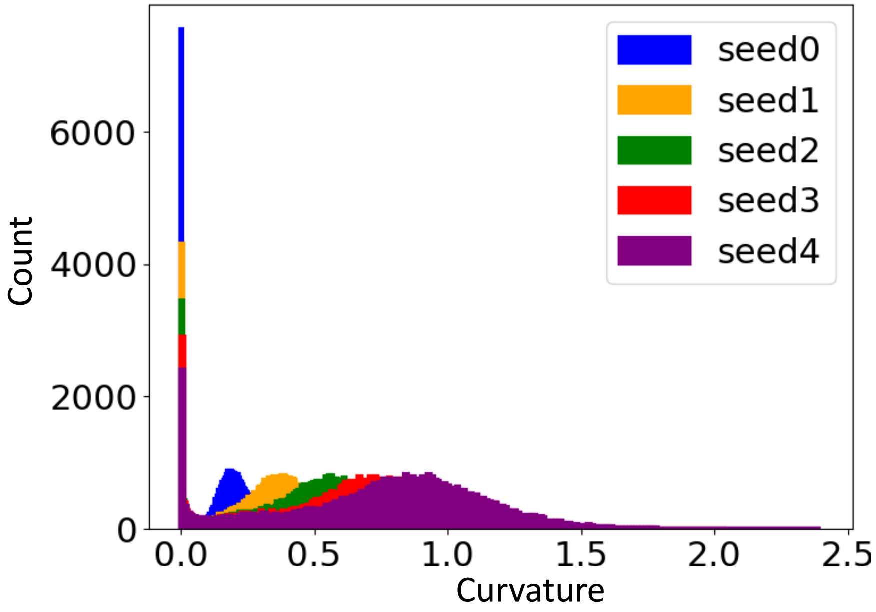
Curvature estimate histograms of nets trained on CIFAR10 training set

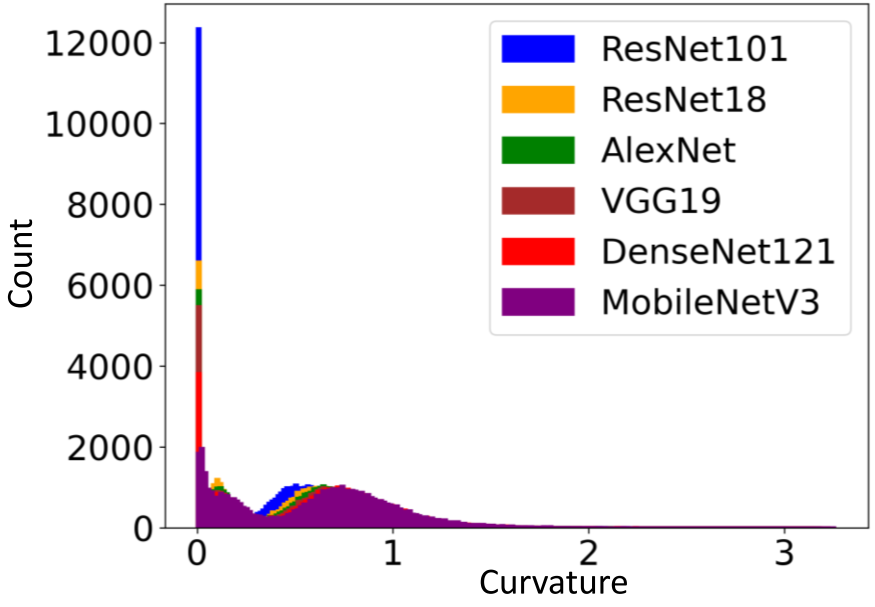


- Train ResNet18 on CIFAR10 until convergence
- We calculate curvature of the training set samples at end
- We plot the histogram of the curvature of all 50000 samples
- Clustering of samples near zero
- Same trend across different architectures

Cumulative Curvature Across Architectures: CIFAR10

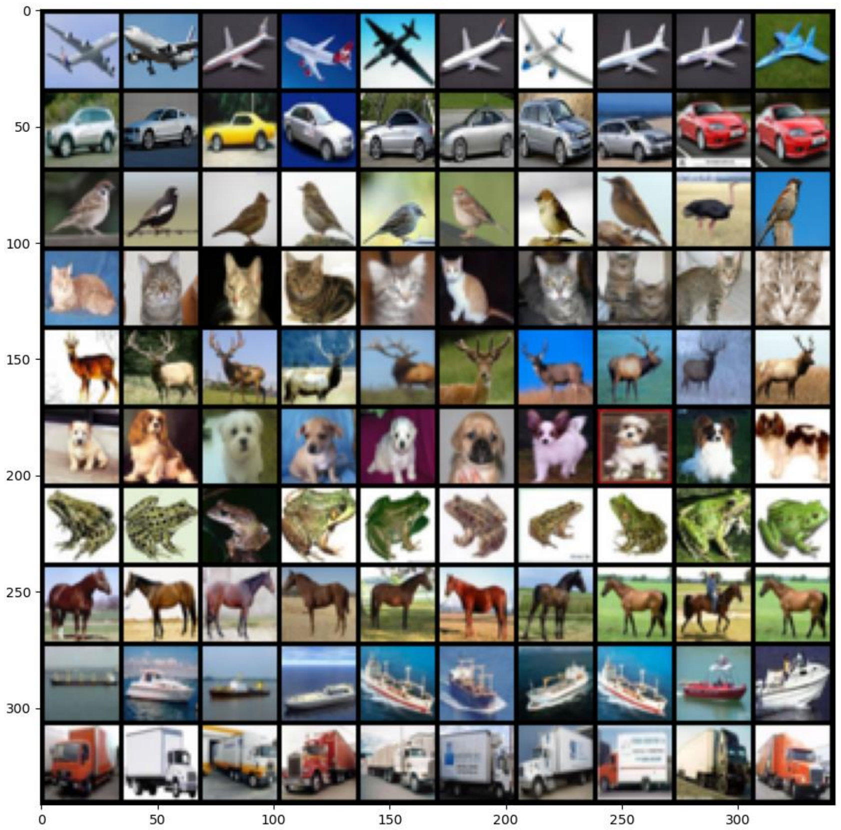
Histograms of cumulative curvature estimate of different nets trained on CIFAR-10 training set



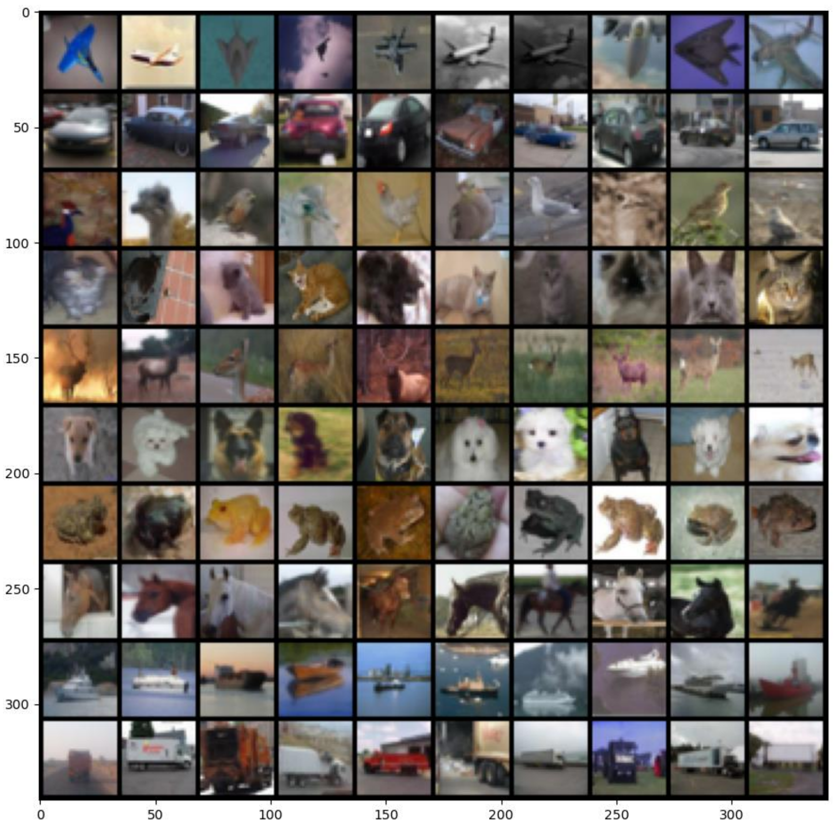


Commonality across architectures suggest curvature reveals dataset properties rather than architecture or training setting dependent properties

Visualizing Low and High Curvature Samples: CIFAR10



airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



CIFAR10: Low Curvature Samples

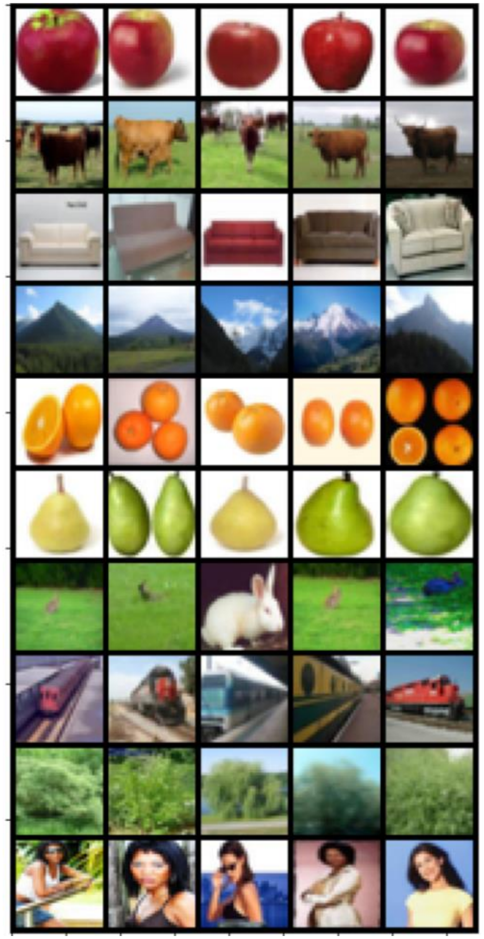
Clean images, free of visual clutter, prototypical of label

CIFAR10: High Curvature Samples

Cluttered, unclear images, often of unidentifiable label

Curvature reveals interesting visual properties of datasets

Visualizing Low and High Curvature Samples: CIFAR100



Apple

Cattle

Couch

Mountain

Orange

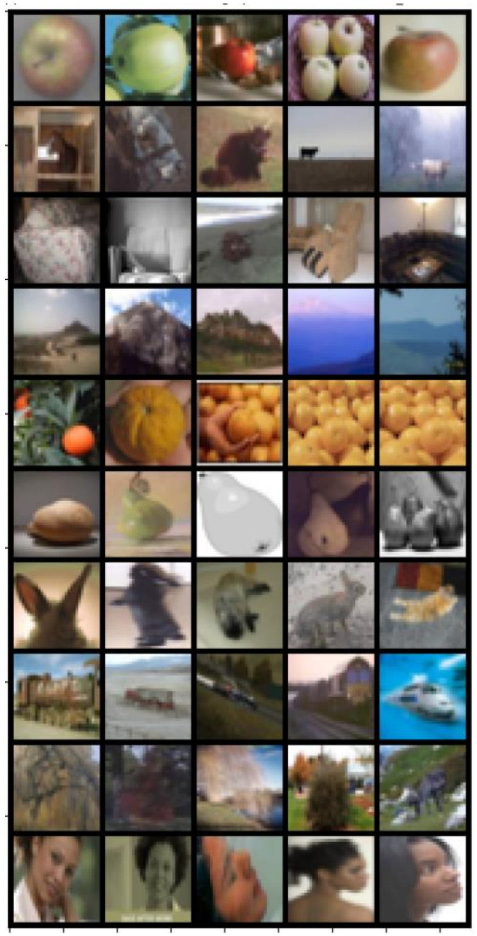
Pear

Rabbit

Train

Willow Tree

Woman



Low Curvature samples

High Curvature samples

Visualizing Low and High Curvature Samples: ImageNet

Indian Cobra

Gibbon

Aircraft Carrier

Binoculars

Convertible

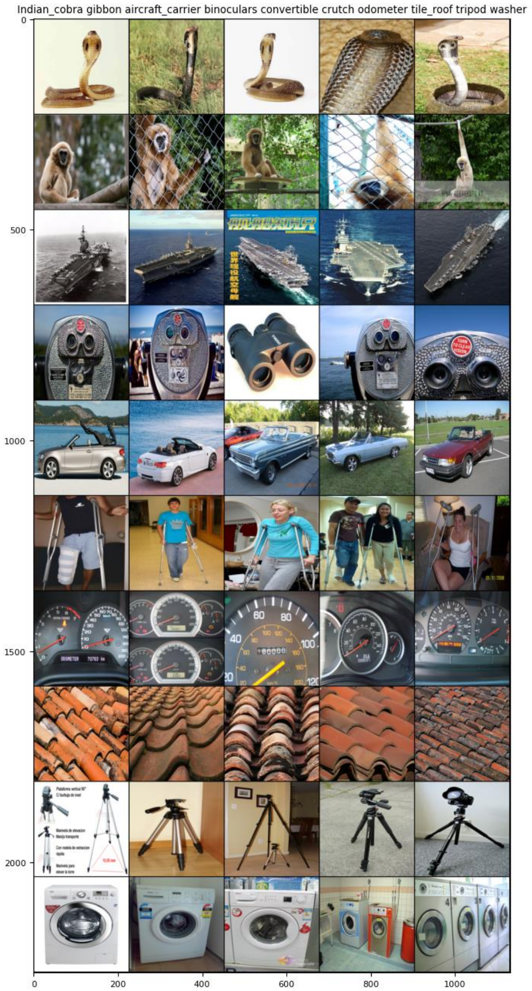
Crutch

Odometer

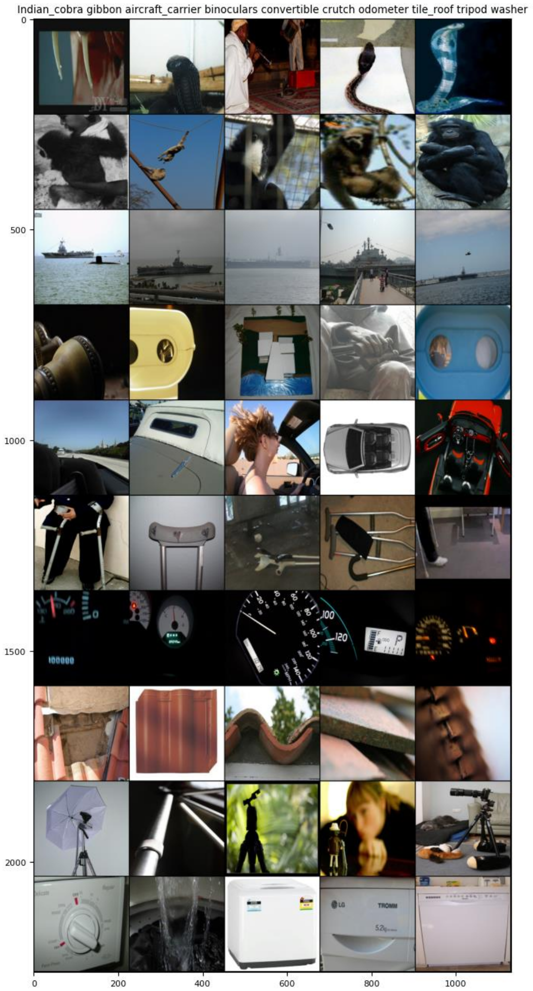
Tile roof

Tripod

Washer



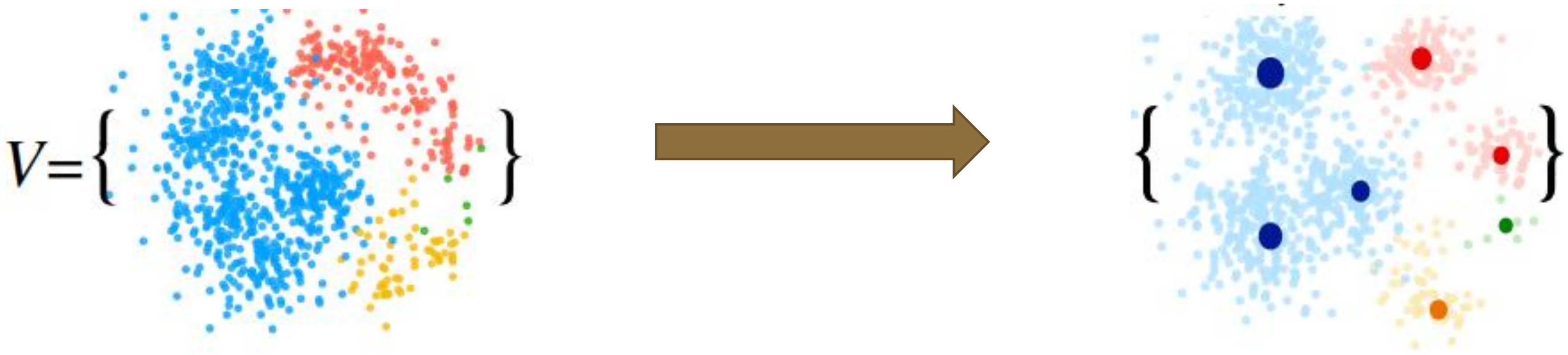
Low Curvature samples



High Curvature samples

What are Coresets?

Training costs depend heavily on dataset size



Creation of coresets can be thought of as summarizing big datasets into smaller, more efficient subset(s)

B. Mirzasoleiman, J. Bilmes, and J. Leskovec, “Coresets for data-efficient training of machine learning models,” in *International Conference on Machine Learning*, ICML, 2020

Proposed Method: SLo-Curves

- Measure Curvature

$$Curv(X) \;=\; \| \, \Delta_x \, [\, L(X+\mathrm{h}v) - \; L(X) \,] \, \|$$

where $V = \frac{sign\,(\Delta_X L(X))}{\|sign(\Delta_X\, L(X))\|}$

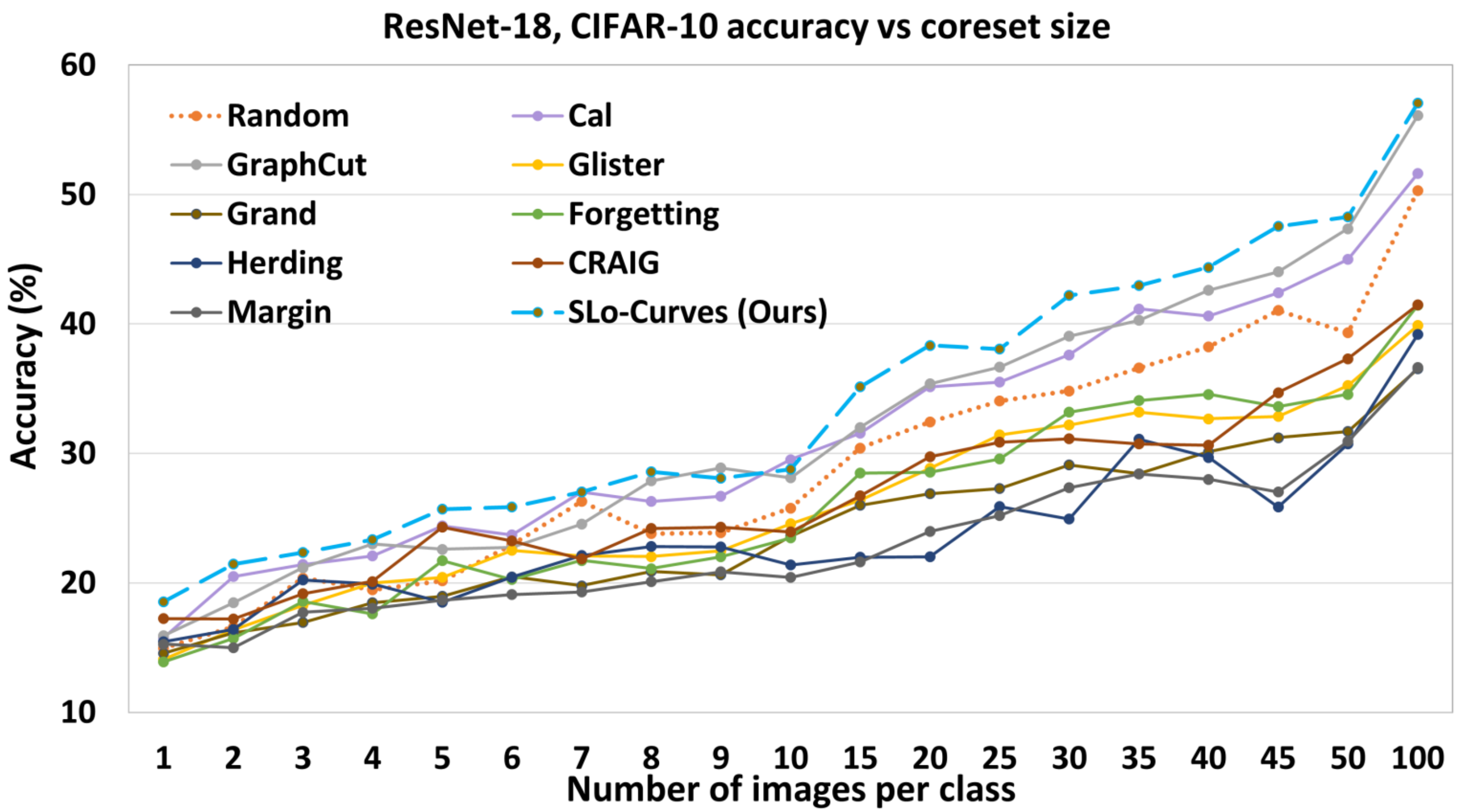
- Sort by curvature and choose desired samples per class with the lowest curvature
- Train on these samples with additional regularizer that penalizes curvature

$$L(X, W) = CE(\hat{y}, y) + \lambda \, Curv(X)$$

where λ is a hyperparameter, searched over {0,0.5,1,5,10,20,50}

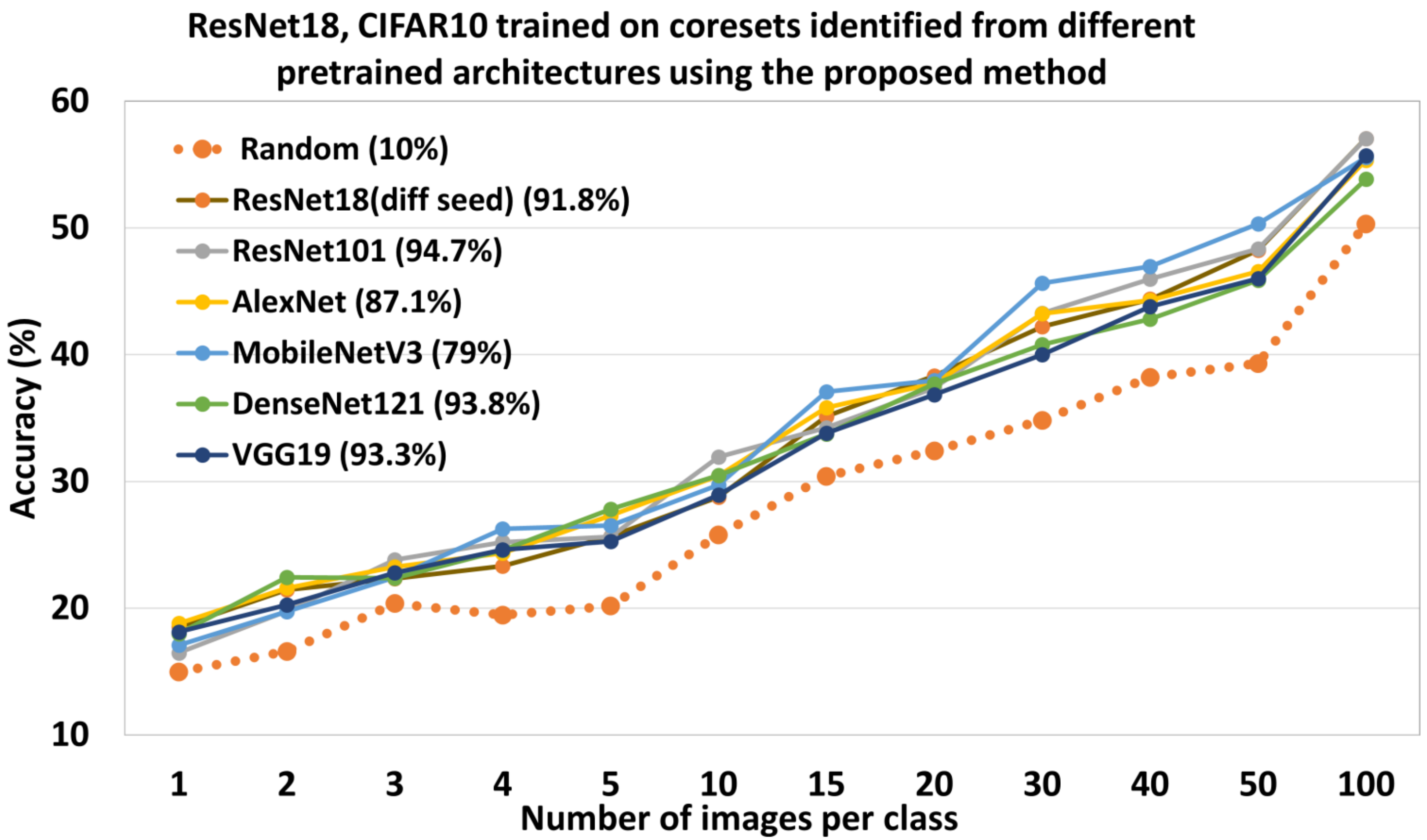
DATA EFFICIENCY RESULTS

Results for CIFAR-10



- SLo-Curves shows the best performance in 16 out of the 19 coreset sizes studied, second best in rest
- It outperforms random sampling by 1-9%

Cross Architecture Results



SLo-curves outperform random sampling for all considered architectures by an average of 5.3%