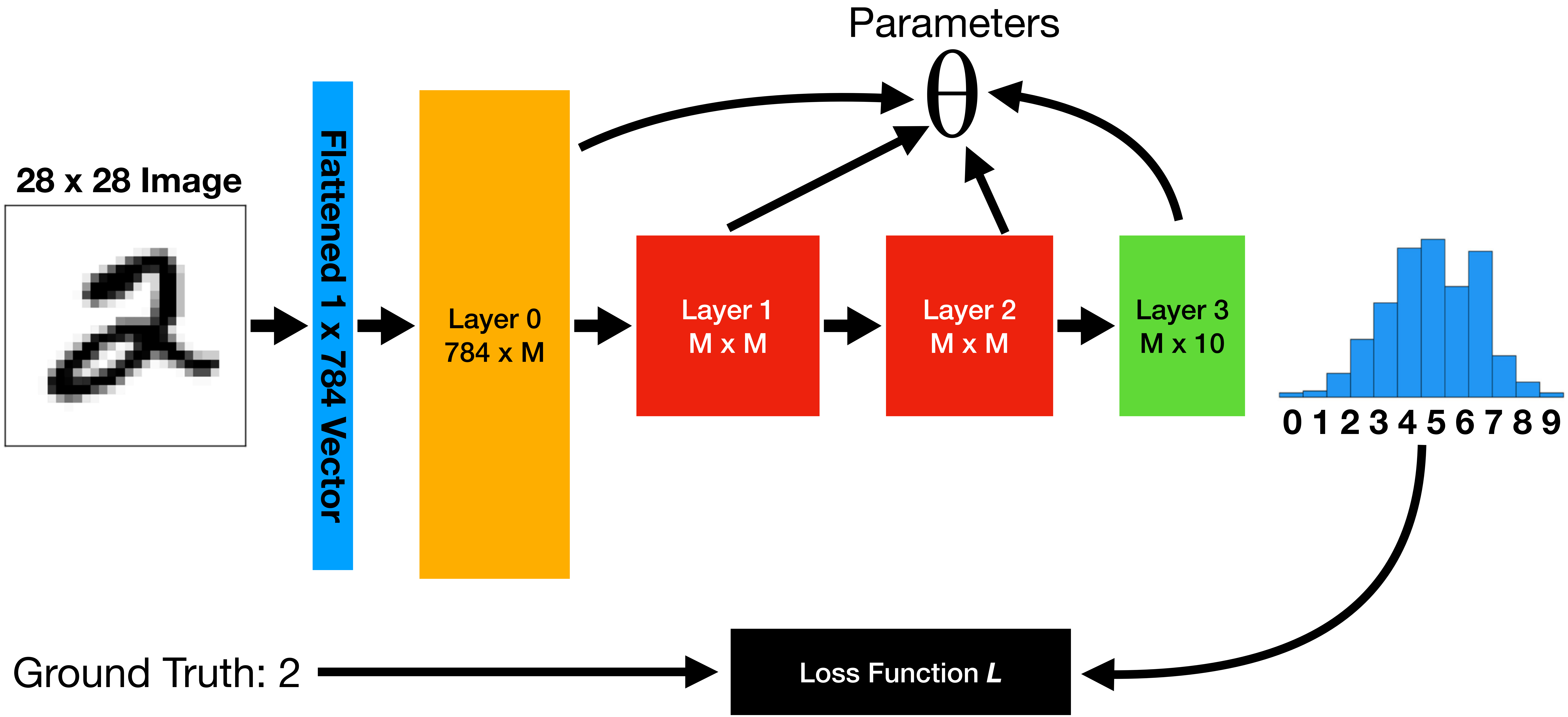
Analyzing the Effect of Quantization on the Neural Network Loss Landscape with the Hessian

Ashwin Adulla

Github: https://github.com/aadulla/21344_NN_Quantization_Hessian_Analysis.git

What is a Neural Network?



What is Quantization?

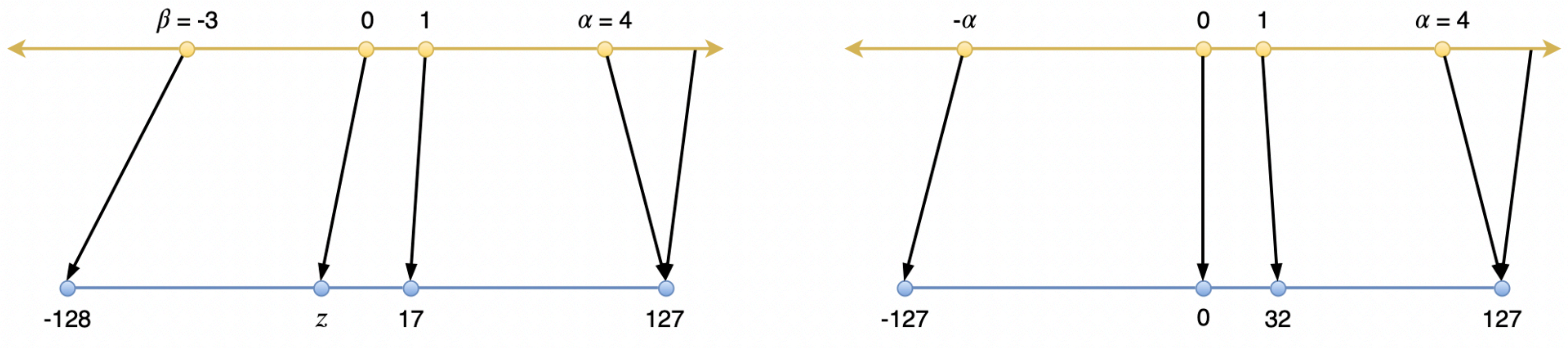
- Quantization: mapping a range of higher-precision numbers to lower-precision numbers

$x \in [\alpha, \beta]$ ← Range of 32-bit Floats

$y \in [-2^{b-1}, 2^{b-1} - 1]$ ← Range of 8-bit Integers, b=8

$f(x) = s * x + z = y$ ← Quantization function ***f***, scale ***s***, zero point ***z***

- Different Quantization Schemes: Affine, Scale, KL Divergence
  - The only difference is in how ***s*** and ***z*** are chosen



(a) Affine quantization

(b) Scale quantization

Hessian Power Method

- Hessian: Matrix second-derivative

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

$$\frac{\partial L}{\partial \theta} = g_\theta \in \mathbb{R}^m$$

$$\mathrm{H} = \frac{\partial^2 L}{\partial \theta} = \frac{\partial g_\theta}{\partial \theta} \in \mathbb{R}^{mxm}$$

- Use the power method to find dominant eigval/eigvec of H
- Only want Hv! Can we do this w/o forming H explicitly?
- Yes! Deep-Learning libraries have an "auto-diff" func which can do a derivative-vector product for us

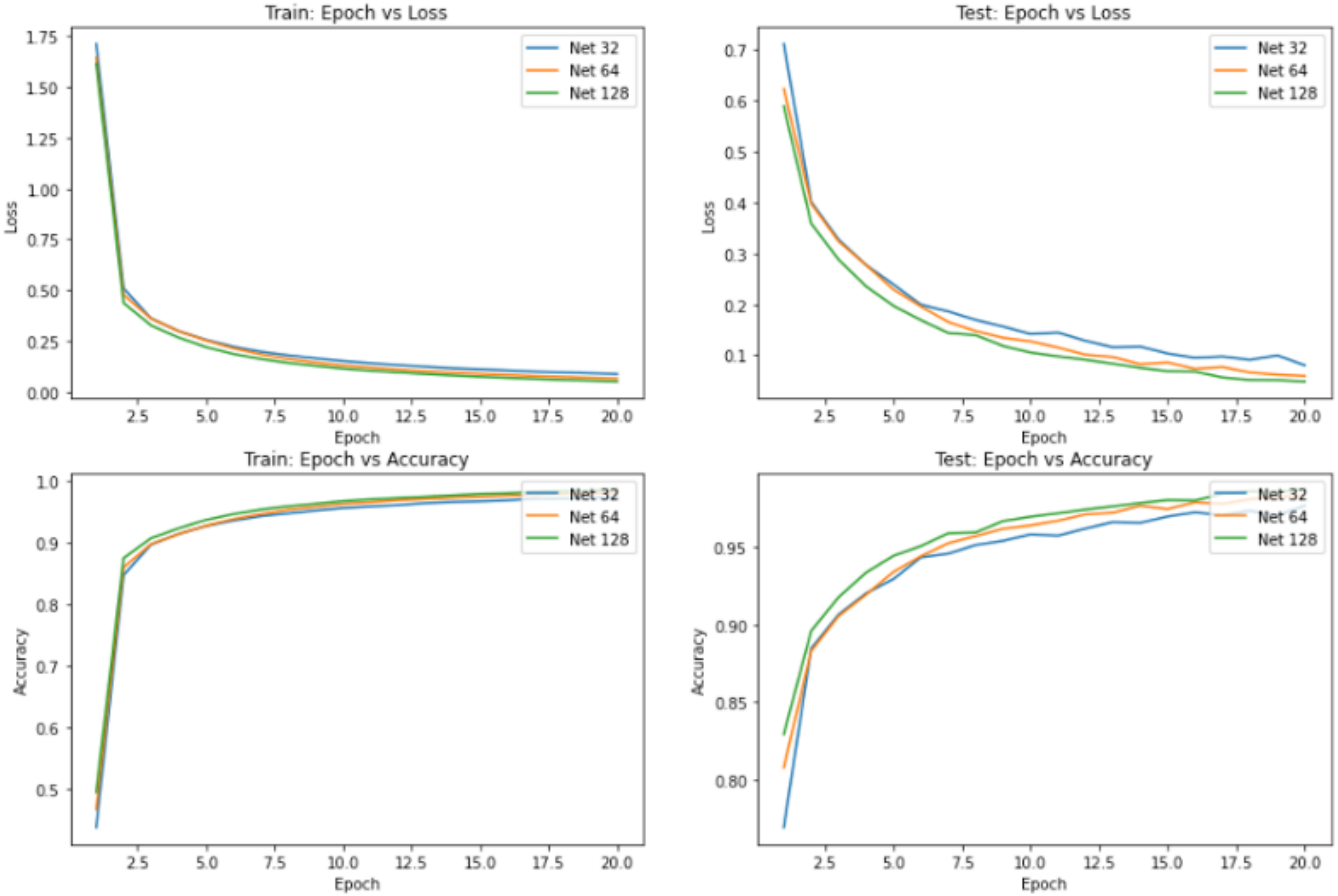
$$\frac{\partial g_\theta^T v}{\partial \theta} = \frac{\partial g_\theta^T}{\partial \theta} v + g_\theta^T \frac{\partial v}{\partial \theta} = \frac{\partial g_\theta^T}{\partial \theta} v = Hv$$

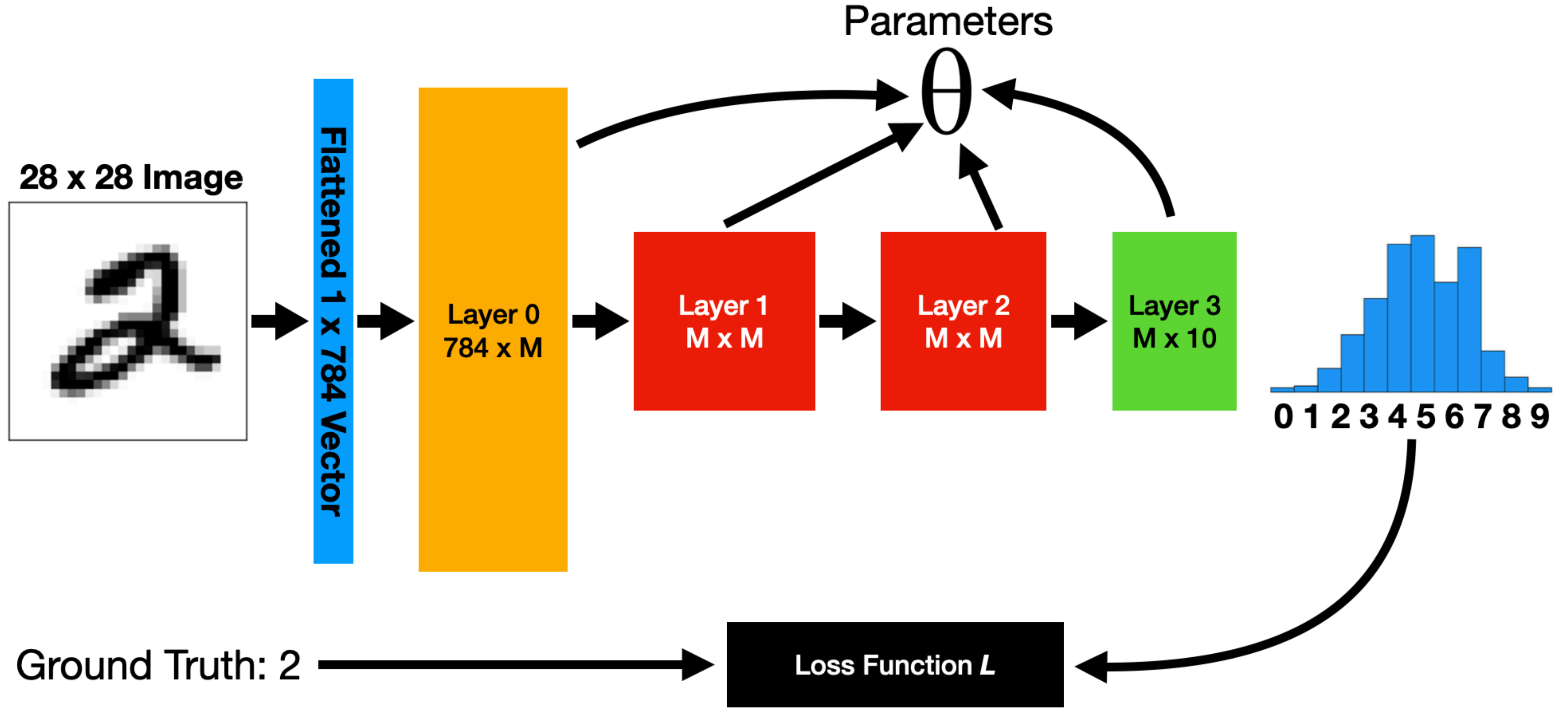
Algorithm 1 Hessian Power Iteration

function H_EIG(g_θ, θ)
  randomly sample $v \in R^m$
  λ = NULL ▷ dominant eigenvalue
  v_λ = NULL ▷ dominant eigenvector
  for i=0, i < MAX_ITER, i++ **do**
    $Hv = \boxed{\text{auto\_diff}(g_\theta,\, \theta,\, v)}$
    $\lambda = Hv * v$ ▷ rayleigh quotient
    v = normalize(v)
    $v_\lambda = v$
    if converged **then**:
      break
  return (λ, v_λ)

Results (Training)

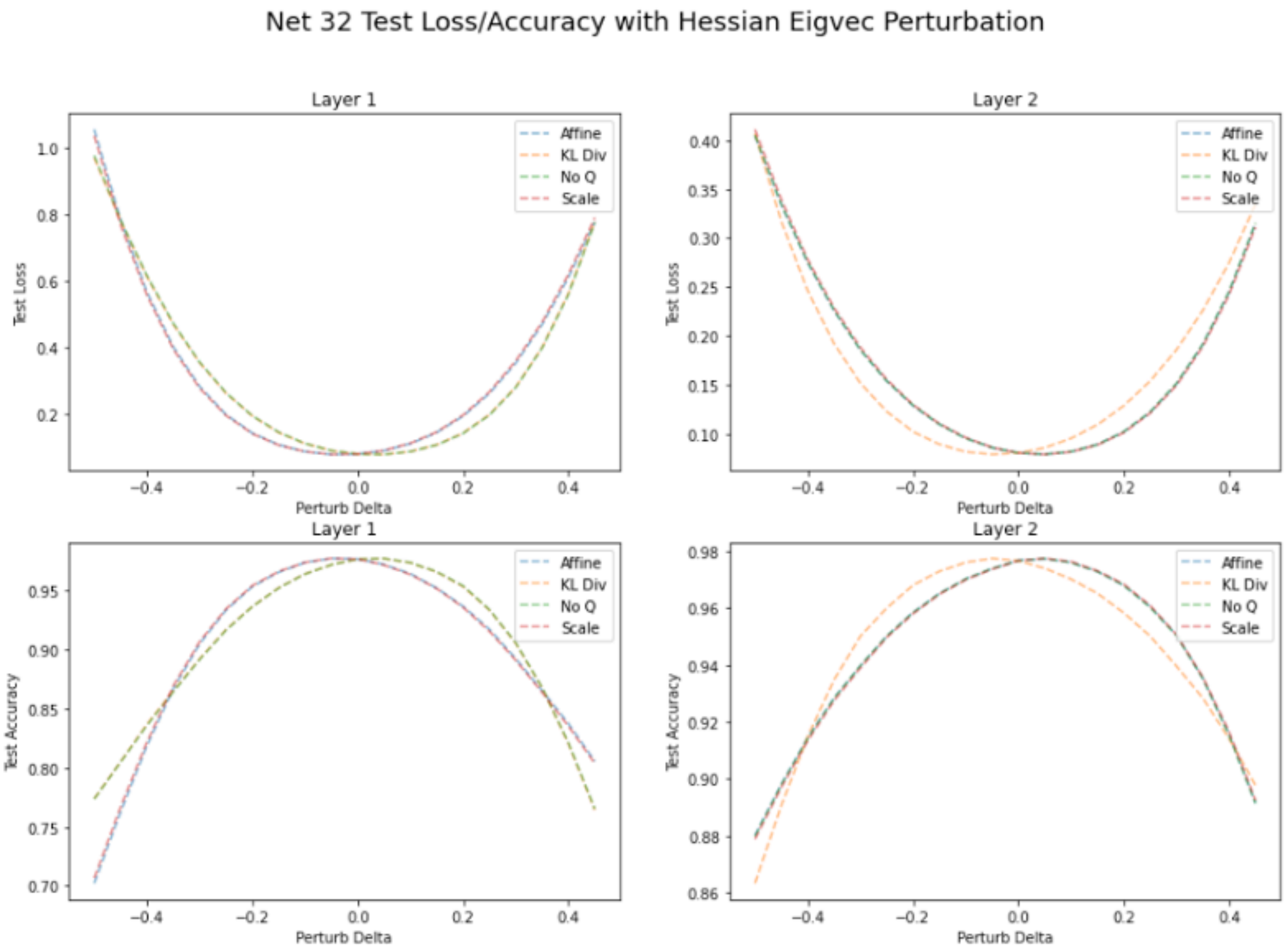
- 3 different networks (M=32, M=64, M=128) trained for 20 epochs





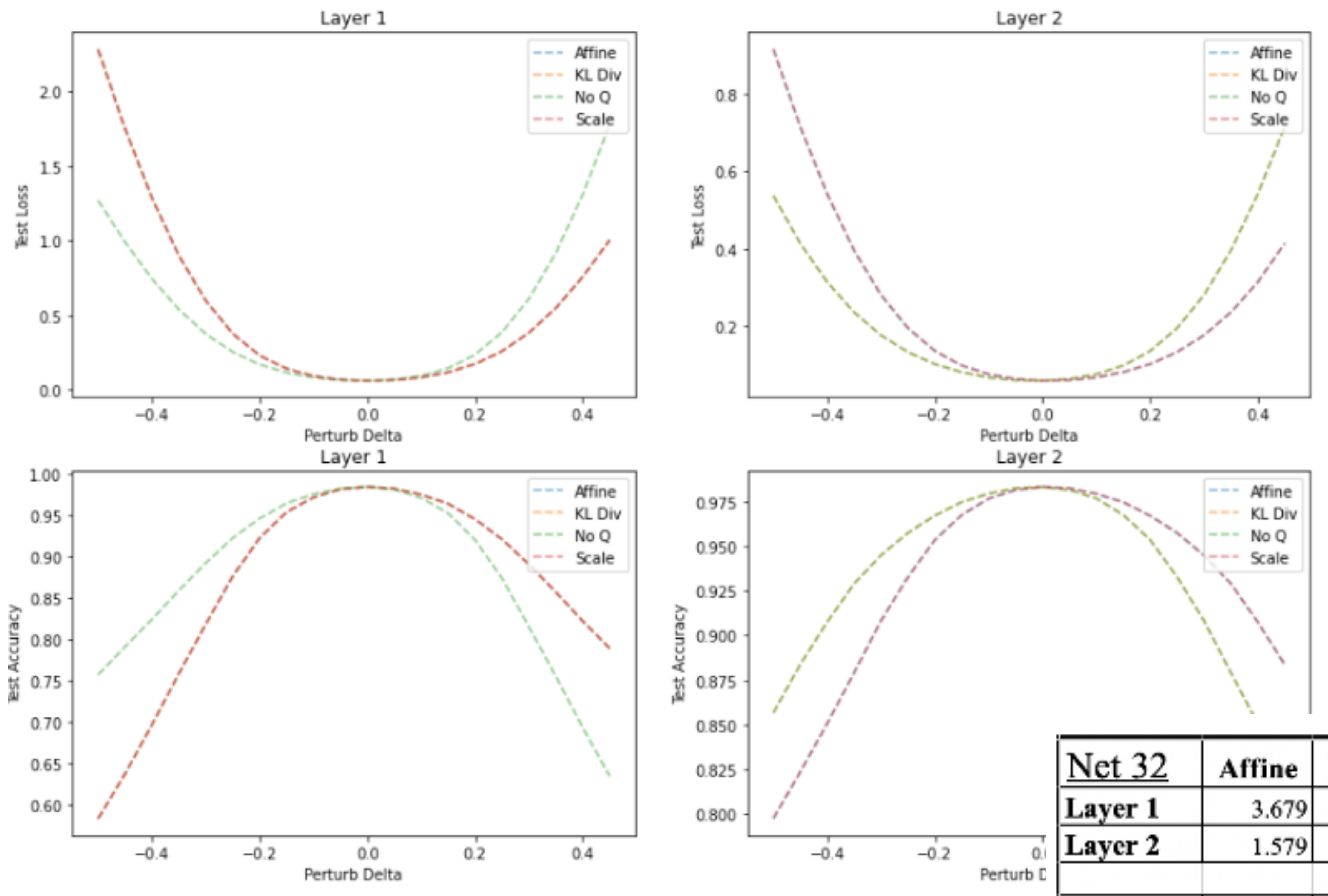
Results (Quantization and Testing)

- After training, Layers 1 and 2 are individually quantized
- The quantized layer is perturbed by the Hessian eigenvector
- Loss/Accuracy is recalculated for every perturbation
- Generate a “Loss Landscape” plot for each quantization scheme
- Convex curves implies the parameters have converged to a local minima

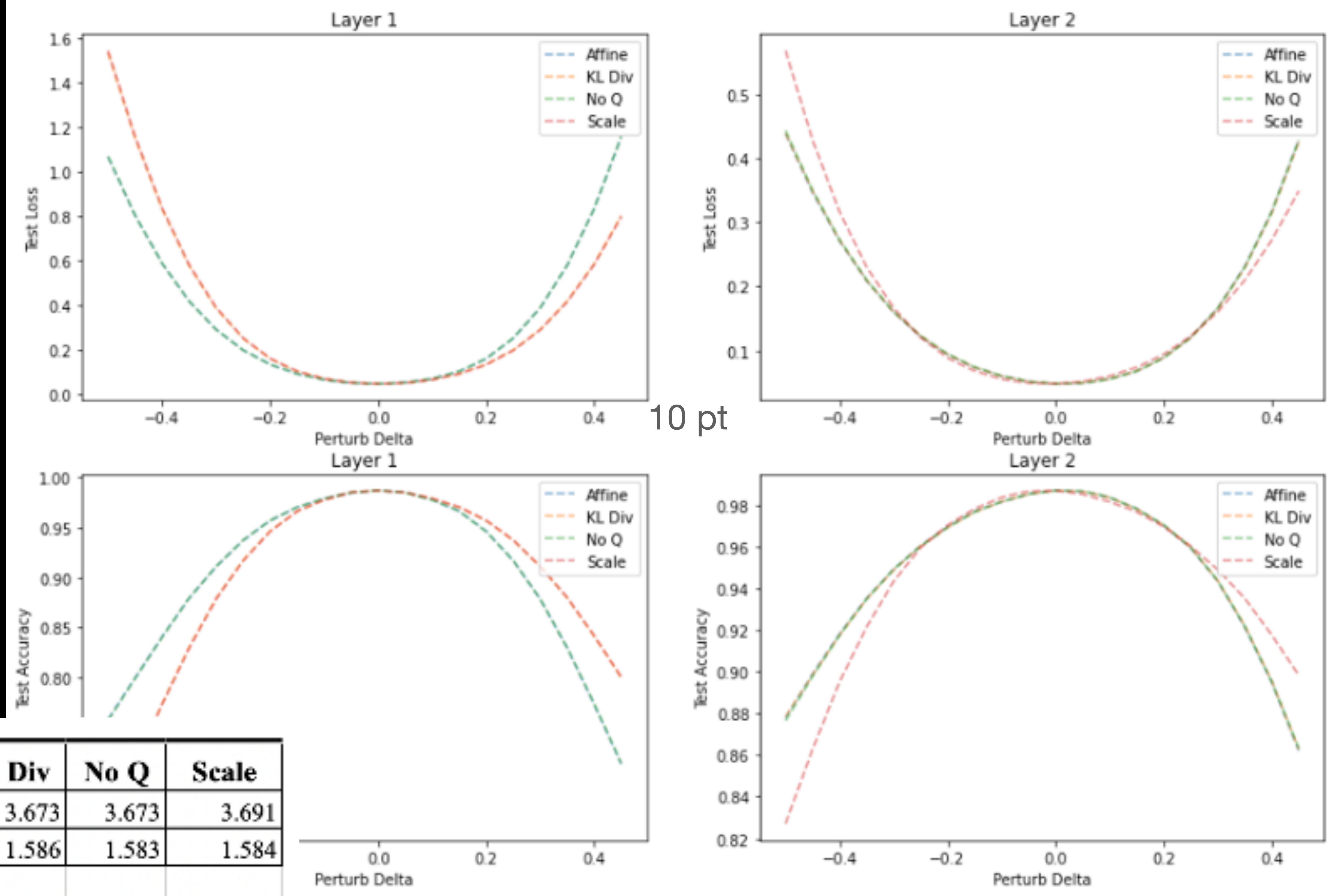


Results (Quantization and Testing)

Net 64 Test Loss/Accuracy with Hessian Eigvec Perturbation



Net 128 Test Loss/Accuracy with Hessian Eigvec Perturbation



10 pt

Net 32	Affine	KL Div	No Q	Scale
Layer 1	3.679	3.673	3.673	3.691
Layer 2	1.579	1.586	1.583	1.584

Net 64	Affine	KL Div	No Q	Scale
Layer 1	4.731	4.731	4.753	4.698
Layer 2	2.322	2.322	2.326	2.335

Net 128	Affine	KL Div	No Q	Scale
Layer 1	3.787	3.777	3.779	3.777
Layer 2	1.819	1.818	1.819	1.814

Dominant Eigenvalues of Hessian

Questions?

References

[1] Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference.
https://arxiv.org/pdf/1712.05877.pdf

[2] Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation.
https://arxiv.org/pdf/2004.09602.pdf

[3] PyHessian: Neural Networks Through the Lens of the Hessian.
https://arxiv.org/pdf/1912.07145.pdf

[4] Visualizing the Loss Landscape of Neural Nets.
https://arxiv.org/pdf/1712.09913.pdf