# Efficient Training of Low-Curvature Neural Networks

## IE 663 Course Project


**Team Name: Supernova**

Members

**Tarun Bisht**

**Indian Institute of Technology Bombay**


March 27, 2023

# Presentation Outline

# Introduction

# Motivation of the Study

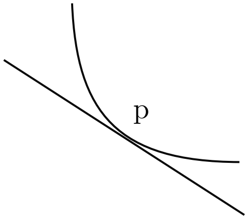


- Deep Neural Networks(DNN) are non-linear by nature, but excess non-linearity is undesirable.
- Highly nonlinear models can have a high degree of sensitivity to small changes in the input, which can lead to large changes in the output. This affects the adversarial robustness of the model.
- Highly non-linear models can also suffer from exploding and vanishing gradient problems.
- This paper addresses the excess non-linearity issue using the idea of curvature and introduces training of low curvature Neural Networks.

# Background

## Curvature

- In mathematics, the curvature is the amount by which a curve deviates from being a straight line.
- Intuitively, it is a measure of how much a curve or surface is bending at a particular point.
- Hessian norm at that point provides a measure of curvature at that point.
- Curvature $C_f(x) = 0 \ \ \forall x \in \mathbb{R}^d \iff f$ is linear.




IIT Bombay

### Curvature in ML context

- In ML, curvature measures the non-linearity of a model.
- A common measure of the curvature of machine learning models is via Hessian norms

$$||\nabla_x^2 f(x)||_2$$

# Background



## Adversarial Robustness

- Adversarial vulnerabilities of neural networks are a widely recognized phenomenon whereby the addition of small, imperceptible amounts of noise to an input can cause deep neural networks to misclassify the input with high confidence.
- Samples generated from adversarial attack methods like the Fast gradient method, Projected gradient descent (PGD) can fool neural networks, although the images look the same to human eyes.
- A go-to method to defend against this vulnerability is adversarial training, which trains models to predict samples correctly generated by adversarial attack methods.
- This approach, however, is computationally expensive and provides no formal guarantees on robustness. The defense against adversarial attacks is an interest of research in the ML community.

# Background



## Gradient Instability

- Gradient instability occurs due to large hessian norms. [Dombrowski]
- Gradient regularization improves model robustness. [Ros and Doshi-Velez]
- Dombrowski et al. proposed to train low curvature models via softplus activations and weight decay.
- Dombrowski et al. [7] focused on the Frobenius norm of the Hessian

$$||A|| = \sqrt{AA^H}$$

- Author's penalize the normalized curvature.

## Lipschitz Layers in Neural Networks

- Ideally with NN we want if input are similar then output should also be similar.
- A function $f : R^M \rightarrow R^N$ is Lipschitz continuous if there is a constant $L$ such that

$$||f(x) - f(y)|| \leq L||x - y||$$

$$\frac{||f(x) - f(y)||}{||x - y||} \leq L$$

  The smallest such $L$ is the Lipschitz constant of $f$ and is denoted $Lip(f)$
- We want constant $L$ to be small as possible.

## Lipschitz Layers in Neural Networks

- Property: Let $f = g \circ h$. If $g$ and $h$ are Lipschitz continuous, then $f$ is also Lipschitz continuous with $Lip(f) \leq Lip(g)Lip(h)$.
- Therefore, if we make each component of a neural network such that it is Lipschitz continuous with small Lipschitz constants, the whole neural network will also be Lipschitz continuous with small Lipschitz constants.
- Activation functions and Pooling layers generally have Lipschitz constant $L = 1$
- Spectral Normalization: where linear layers are re-parameterized by dividing by their spectral norm, ensuring that the overall spectral norm of the parameterized layer is 1 which bounds $L$ to be 1.

# Training a Low Curvature Neural Network

## Measuring Relative Model Non-Linearity via Normalized Curvature

- Common way to define curvature is via Hessain norm, however this measure is sensitive to gradient scaling.
- Scaling a function also scale its hessian.
- To avoid this problem author's proposed normalized curvature.

$$C_f(x) = \frac{||\nabla^2 f(x)||_2}{||\nabla f(x)||_2 + \epsilon}, \quad \epsilon > 0$$

# Training a Low Curvature Neural Network



- Directly penalizing the curvature is computationally expensive as it requires calculating the Hessian norm.
- We try controlling the curvature and Lipschitz constant of each layer of a neural network that enables us to control the overall curvature of the model.

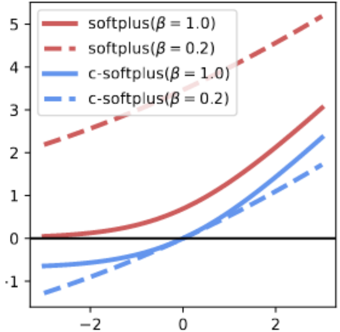
# Centered Softplus Activation



- Authors propose to use activation functions with minimal curvature.
- softplus function $s(x;\beta) = \frac{log(1+exp(\beta x))}{\beta}$
- curvature of softplus $= \beta(1 - \nabla s(x;\beta))$
- using smaller $\beta$ ensure lower curvature.

# Centered Softplus Activation






(c) Softplus vs Centered-Softplus

- Centered softplus: $s_0(x, \beta) = s(x, \beta) - \frac{log2}{\beta} = \frac{1}{\beta} log(\frac{1+exp(\beta x)}{2})$

# Experiments by Author

# Experiments by Author



- In experiments authors,
  - evaluates the effectiveness of the proposed method in training models with low curvature
  - evaluate whether low curvature models have robust gradients in practice
  - evaluate the effectiveness of low-curvature models for adversarial robustness
- Dataset Used
  - CIFAR100
  - CIFAR10
- The experiments are performed using NVIDIA GeForce GTX 1080 Tis.
- Implemented in Pytorch
- Baseline model: Resnet18

## Experiments Results



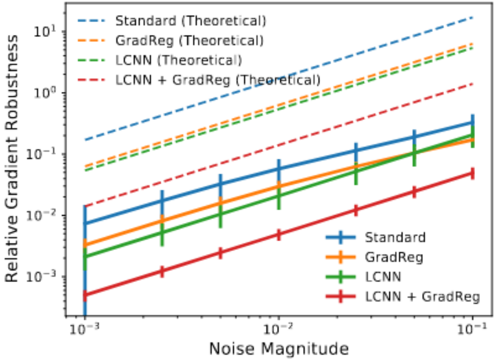
- Model geometry of various ResNet-18 models trained with various regularizers on the CIFAR100 test dataset.
- Results are averaged across two runs.

| Model | $\mathbb{E}_{\mathbf{x}} \lVert \nabla f(\mathbf{x}) \rVert_2$ | $\mathbb{E}_{\mathbf{x}} \lVert \nabla^2 f(\mathbf{x}) \rVert_2$ | $\mathbb{E}_{\mathbf{x}} \mathcal{C}_f(\mathbf{x})$ | Accuracy (%) |
|---|---|---|---|---|
| Standard | 19.66 $\pm$ 0.33 | 6061.96 $\pm$ 968.05 | 270.89 $\pm$ 75.04 | 77.42 $\pm$ 0.11 |
| LCNNs | 22.04 $\pm$ 1.41 | 1143.62 $\pm$ 99.38 | 69.50 $\pm$ 2.41 | 77.30 $\pm$ 0.11 |
| GradReg [31] | **8.86** $\pm$ 0.12 | 776.56 $\pm$ 63.62 | 89.47 $\pm$ 5.86 | 77.20 $\pm$ 0.26 |
| LCNNs + GradReg | 9.87 $\pm$ 0.27 | **154.36** $\pm$ 0.22 | **25.30** $\pm$ 0.09 | 77.29 $\pm$ 0.07 |
| CURE [6] | **8.86** $\pm$ 0.01 | 979.45 $\pm$ 14.05 | 116.31 $\pm$ 4.58 | 76.48 $\pm$ 0.07 |
| Softplus + Wt. Decay [7] | 18.08 $\pm$ 0.05 | 1052.84 $\pm$ 7.27 | 70.39 $\pm$ 0.88 | 77.44 $\pm$ 0.28 |
| Adversarial Training [32] | **7.99** $\pm$ 0.03 | 501.43 $\pm$ 18.64 | 63.79 $\pm$ 1.65 | 76.96 $\pm$ 0.26 |

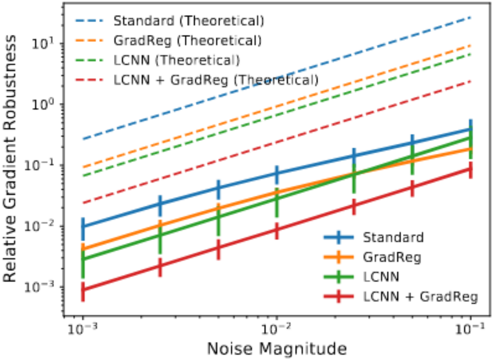
# Experiments Results



Plot showing relative gradient robustness, $\frac{||\nabla f(x+\epsilon)-\nabla f(x)||_2}{||\nabla f(x)||_2}$




(a) Gradient Robustness on CIFAR10




(b) Gradient Robustness on CIFAR100

# Experiments Results

Results indicate off-the-shelf model accuracies upon using $l2$ PGD adversarial examples across various noise magnitudes $\epsilon$.

| **Model** | Acc. (%) | $\|\|\epsilon\|\|_2 = 0.05$ | $\|\|\epsilon\|\|_2 = 0.1$ | $\|\|\epsilon\|\|_2 = 0.15$ | $\|\|\epsilon\|\|_2 = 0.2$ |
|---|---|---|---|---|---|
| Standard | 77.42 ± .10 | 59.97 ± .11 | 37.55 ± .13 | 23.41 ± .08 | 16.11 ± .21 |
| LCNN | 77.16 ± .07 | 61.17 ± .53 | 39.72 ± .17 | 25.60 ± .32 | 17.66 ± .18 |
| GradReg | 77.20 ± .26 | 71.90 ± .11 | 61.06 ± .03 | 49.19 ± .12 | 38.09 ± .47 |
| LCNNs + GradReg | 77.29 ± .26 | **72.68** ± .52 | **63.36** ± .39 | **52.96** ± .76 | **42.70** ± .77 |
| CURE [6] | 76.48 ± .07 | 71.39 ± .12 | 61.28 ± .32 | 49.60 ± .09 | 39.04 ± .16 |
| Softplus + Wt. Decay [7] | 77.44 ± .28 | 60.86 ± .36 | 38.04 ± .43 | 23.85 ± .33 | 16.20 ± .01 |
| Adversarial Training [32] | 76.96 ± .26 | **72.76** ± .15 | **64.70** ± .20 | **54.80** ± .25 | **44.98** ± .57 |

# Experiments by Team

# Experiments by Team



## Experimental Setup

- For testing Linux Machine CPU: Intel I3, GPU: Geforce 940MX, 12GB RAM.
- For longer training session Kaggle, GPU: P400, 16GB RAM, was used.

## Dataset

- CIFAR10

## Programming Setup

- PyTorch
- Default configurations and parameters used by authors were used.

# Experiments Results



| Model | $E_x\|\|\nabla f(x)\|\|_2$ | $E_x\|\|\nabla^2 f(x)\|\|_2$ | $E_x C_f(x)$ | Accuracy |
|---|---|---|---|---|
| Standard | $5.344159 \pm 15.07$ | $668.403381 \pm 2161.75$ | $119.779434 \pm 86.29$ | 93 |
| LCNNs | $2.738105 \pm 1.79$ | $11.158964 \pm 7.37$ | $4.531949 \pm 2.06$ | 64 |
| GradReg | $1.444337 \pm 1.79$ | $49.747406 \pm 126.30$ | $32.834171 \pm 19.15$ | 93 |
| LCNN + GradReg | $0.900491 \pm 0.55$ | $1.988849 \pm 1.47$ | $2.161540 \pm 1.03$ | 62 |

# Further Investigations



- Analyze robustness towards adversarial attacks

# References



- https://towardsdatascience.com/lipschitz-continuity-and-spectral-normalization-b03b36066b0d
- https://arxiv.org/abs/2206.07144

# Thank You!