Rotate the ReLU to Implicitly Sparsify Deep Networks

Ph.D. Seminar II

Presented by: Nancy Nayak
Supervisor: Dr. Sheetal Kalyani

February 4, 2024

Indian Institute of Technology Madras, India

Outline

1. Necessity for energy-efficient Deep Networks
2. Rotated ReLU activation
3. Intrinsic structural sparsity
4. Insights on Results, Discussion, Scalability, and Robustness

Necessity for energy-efficient Deep Networks

Deep Learning has revolutionized lot of fields

- Deep learning in Vision, AlphaGo, Language, Speech, Self-driving cars, AlphaFold
- Performance is improving rapidly to surpass human performance[1]

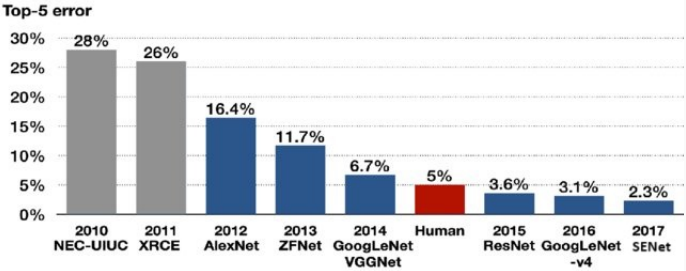


Figure 1: Imagenet entries. Blue: Deep Learning models

- Recent addition WideResNet (2016) and Vision Transformers (2020)
- For resource-constrained, energy-efficient green networks, the major concerns regarding the deployable network are (i) **Model size** and (ii) **Computation**

[1] Source: `https://www.implantology.or.kr/articles/xml/RvNO/`

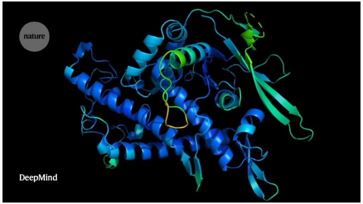
Bigger models consumes more power

- Model size is the number of parameters - every year model size increases by $10\times$
- Models with bigger memory is expensive - more memory movement consumes more power
- Two approaches
  - Efficient algorithm - reduce number of parameters and number of activations
  - Efficient hardware - select important features or quantize model parameters after training

High computation for heavier tasks

(2.1) A whole trunk of workstation for Self-driving cars[2]

(2.2) Compute: 1920 CPUs and 280 GPUs ($3000 electricity bill per game of AlphaGo[3])



(2.3) Compute: 16 TPUv3s (128 TPU v3 cores) for few weeks[4] for AlphaFold

[2] Source: `https://www.autonomousvehicletechnologyexpo.com/en/`

[3] Source: `https://futureoflife.org/recent-news/alphago-and-ai-progress/`

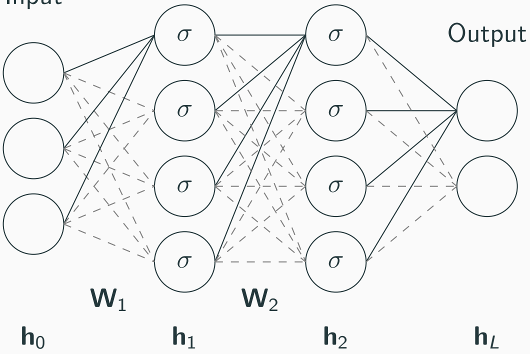
[4] Source: `https://www.deepmind.com/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology`

How to get Greener Deep Networks?

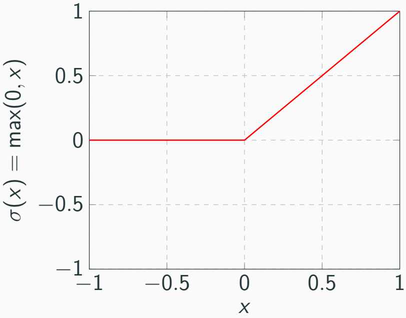
- Existing compression techniques
  - Quantization, Binarization, Transfer learning, Low-rank approximation, Pruning connection, weight, and channels with **sparsification** using
    - Regularization - reduces memory size
    - Group sparsity based regularization - reduces both memory size and computation
- We propose a new activation **Rotated ReLU** that intrinsically structurally sparsifies deep networks

Rotated ReLU activation

ReLU in a DNN

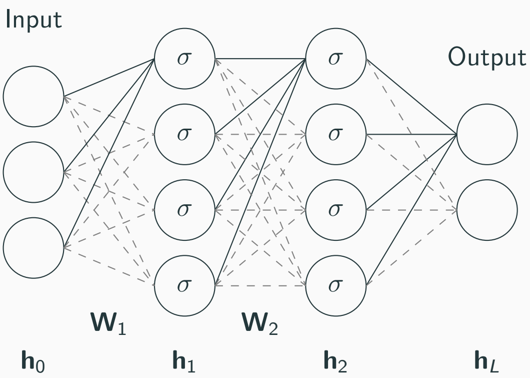


- Output of l^{th} hidden layer
  $\mathbf{h}_{l+1} = \sigma(\mathcal{F}(\mathbf{h}_l; \mathbf{W}_l))$ where σ is ReLU

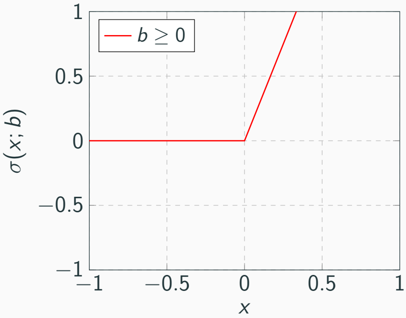


ReLU activation

Rotated ReLU in a DNN

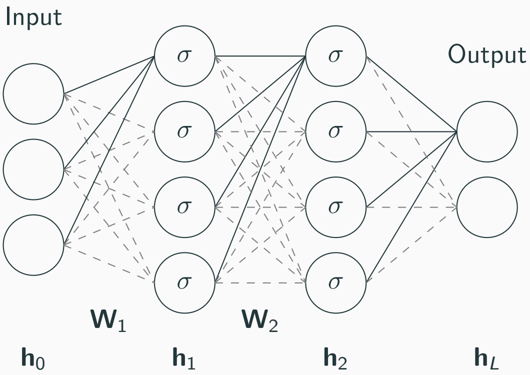


- Increase the degree of freedom of ReLU by rotating the linear part

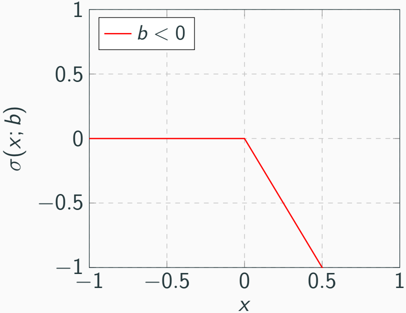


RReLU activation $\sigma(x; b) = b \max(0, x)$

Rotated ReLU in a DNN

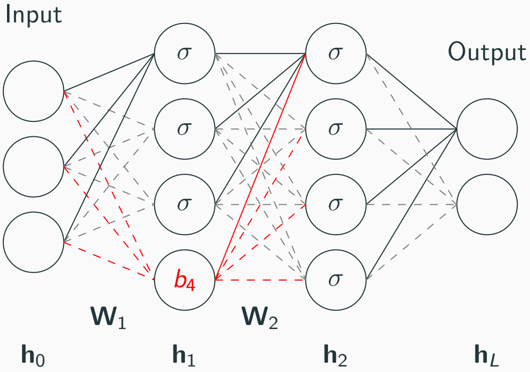


- Increase the degree of freedom of ReLU by rotating the linear part



RReLU activation $\sigma(x; b) = b \max(0, x)$

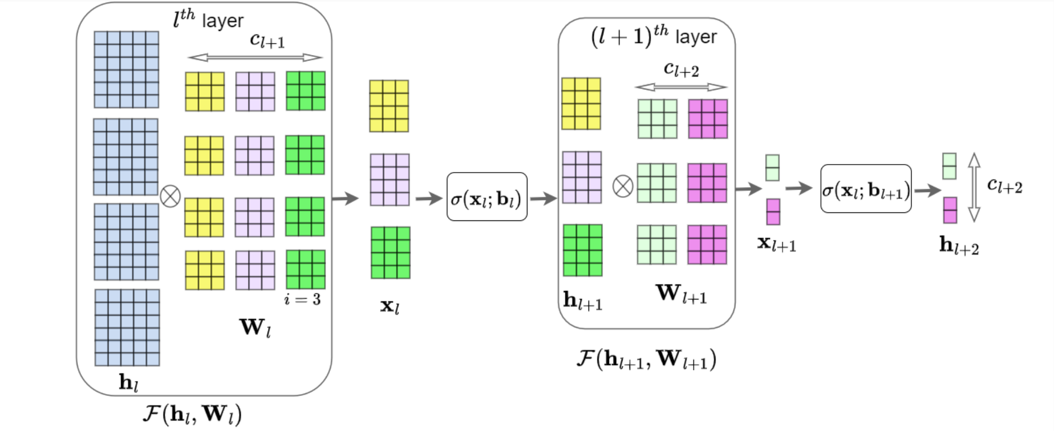
Rotated ReLU in a DNN



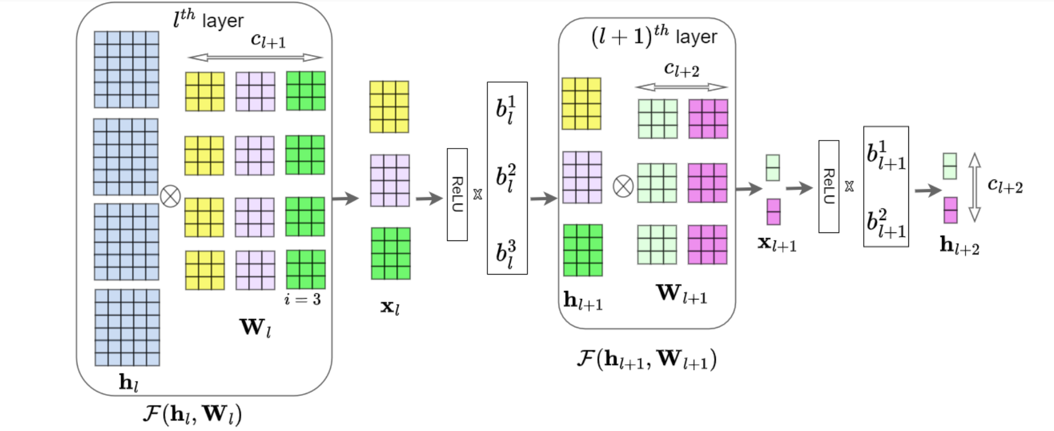
- Now, the output of the l^{th} layer is $\mathbf{h}_{l+1} = \sigma(\mathbf{x}_l; \mathbf{b}_l) = \mathbf{b}_l \max(0, \mathbf{x}_l)$ where $\mathbf{x}_l = \mathcal{F}(\mathbf{h}_l; \mathbf{W}_l)$
- $\mathbf{W}_1 = \begin{bmatrix} w_1^{11} & w_1^{12} & w_1^{13} \\ w_1^{21} & w_1^{22} & w_1^{23} \\ w_1^{31} & w_1^{32} & w_1^{33} \\ w_1^{41} & w_1^{42} & w_1^{43} \end{bmatrix}$ and

  $\mathbf{W}_2 = \begin{bmatrix} w_2^{11} & w_2^{12} & w_2^{13} & w_2^{14} \\ w_2^{21} & w_2^{22} & w_2^{23} & w_2^{24} \\ w_2^{31} & w_2^{32} & w_2^{33} & w_2^{34} \\ w_2^{41} & w_2^{42} & w_2^{43} & w_2^{44} \end{bmatrix}$
- Highlighted connections are unimportant if $b_4 \rightarrow 0$

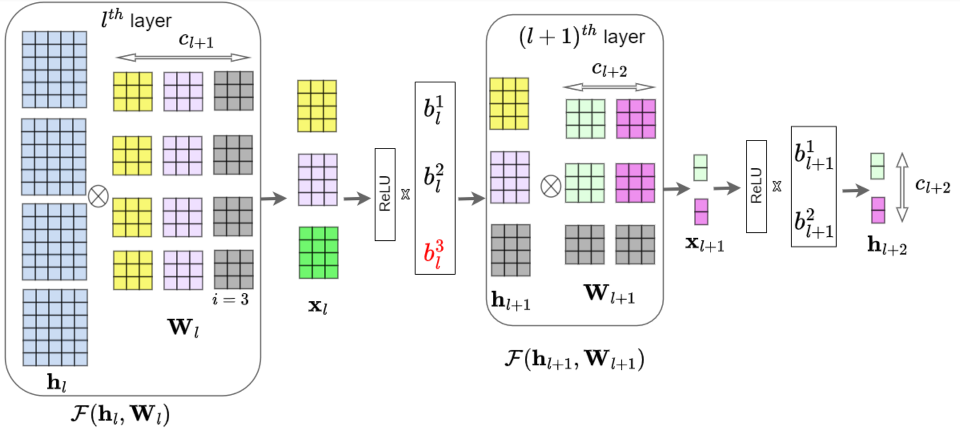
Rotated ReLU in a CNN



Rotated ReLU in a CNN



Rotated ReLU in a CNN



- if $b_l^3 \to 0$, then they corresponding filters are unimportant

Rotated ReLU in a CNN[5]

- If the output of RReLU at the l^{th} layer has c_{l+1} channels and n entries of $\mathbf{b}_l$ are insignificant, then only $(c_{l+1} - n)$ channels remain significant
- Saving in Memory:
  - Original model size: $c_{l+1} c_l k^2$ (l^{th} layer) and $c_{l+2} c_{l+1} k^2$ ($(l+1)^{th}$ layer)
  - Sparse model size: $(c_{l+1} - n) c_l k^2$ (l^{th} layer) and $c_{l+2}(c_{l+1} - n) k^2$ ($(l+1)^{th}$ layer)
- Saving in Computation:
  - Original model FLOP: $2 c_l k^2 \bar{h}^w_{l+1} \bar{h}^h_{l+1} c_{l+1}$ (l^{th} layer) and $2 c_{l+1} k^2 \bar{h}^w_{l+2} \bar{h}^h_{l+2} c_{l+2}$ ($(l+1)^{th}$ layer)
  - Sparse model FLOP: $2 c_l k^2 \bar{h}^w_{l+1} \bar{h}^h_{l+1} (c_{l+1} - n)$ (l^{th} layer) and $2(c_{l+1} - n) k^2 \bar{h}^w_{l+2} \bar{h}^h_{l+2} c_{l+2}$ ($(l+1)^{th}$ layer)

[5] $\mathbf{w}_l \in \mathbb{R}^{c_{l+1} \times c_l \times k \times k}$ is the filter for the l^{th} layer of a 2D CNN; k is the dimension of the filter; c_l and c_{l+1} represent the number of input and output channels at the l^{th} layer, respectively; $(\bar{h}^w_l, \bar{h}^h_l)$ and (h^w_{l+1}, h^h_{l+1}) are spatial dimensions (width, height) of the input and the output

Intrinsic structural sparsity

RReLU in ResNet architectures

- When the input $\mathbf{h}_l$ is fed to the l^{th} layer of a residual unit with ReLU, the output:

$$\mathbf{h}_{l+2} = \max(0, \mathbf{h}_l + \gamma_{l+1}\text{Conv}(\underbrace{\max(0, \underbrace{\gamma_l \text{Conv}(\mathbf{h}_l; \mathbf{W}_l) + \boldsymbol{\beta}_l}_{\mathbf{x}_l = \mathcal{F}(\mathbf{h}_l; \mathbf{W}_l, \gamma_l, \boldsymbol{\beta}_l)})}_{\mathbf{h}_{l+1}}; \mathbf{W}_{l+1}) + \boldsymbol{\beta}_{l+1}), \tag{1}$$

  where γ_l and $\boldsymbol{\beta}_l$ are the batchnorm scaling and shifting parameters[6], respectively.
- The same with RReLU:

$$\mathbf{h}_{l+2} = \mathbf{b}_{l+1} \max(0, \mathbf{h}_l + \gamma_{l+1}\text{Conv}(\underbrace{\mathbf{b}_l \max(0, \underbrace{\gamma_l \text{Conv}(\mathbf{h}_l; \mathbf{W}_l) + \boldsymbol{\beta}_l}_{\mathbf{x}_l = \mathcal{F}(\mathbf{h}_l; \mathbf{W}_l, \gamma_l, \boldsymbol{\beta}_l)})}_{\mathbf{h}_{l+1}}; \mathbf{W}_{l+1}) + \boldsymbol{\beta}_{l+1}), \tag{2}$$

  where $\mathbf{b}_l$ is the RReLU slopes.
- RReLU **enhances the representation power corresponding to every filter**[7]

[6] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In International conference on machine learning, pp. 448-456. pmlr, 2015.

[7] overall representation power of the network remains same

Enhanced Filter Representation with γ_l and $\mathbf{b}_l$

- When elements of $\mathbf{b}_l$ reach zero, the elements of γ_l approach zero too
- When some elements of $\mathbf{b}_l$ don't approach zero, the corresponding elements of γ_l take wide range of values
- Intrinsically, many filters becomes unnecessary and the corresponding RReLU slopes become insignificant without any regularization on $\mathbf{b}_l$

Can γ_l alone sparsify?

- γ_l alone cannot achieve the same level of sparsity as RReLU as γ_l are initialized with positive values and do not fully explore negative values
- Negative values of γ_l may take the output features to map to the negative part of ReLU activation - **dying ReLU** problem - not recommended
- Network-slimming[8] utilizes the L_1 norm on $\gamma_l, \forall l \in L$, to force each of the elements of γ_l to approach zero
- With RReLU, while minimizing L_1 norm on $\mathbf{b}_l$, **every element of $\mathbf{b}_l$ is compelled to adopt smaller values, but $b_l^{\{i\}}\gamma_l^{\{i\}}$ remains unconstrained**

[8] Liu, Zhuang, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. "Learning efficient convolutional networks through network slimming." In Proceedings of the IEEE international conference on computer vision, pp. 2736-2744. 2017.

Insights on Results, Discussion, Scalability, and Robustness

Initialization of RReLU slopes

- $\mathbf{W}_l$ is initialized with Kaiming He[9] initialization method
- The RReLU slopes $\mathbf{b}_l$ for all $l \in L$ are initialized with a truncated Gaussian Mixture Model (GMM) with a mean of $\{+1, -1\}$ and a variance of 3
- ResNets for both ReLU and RReLU are trained for 1200 epochs

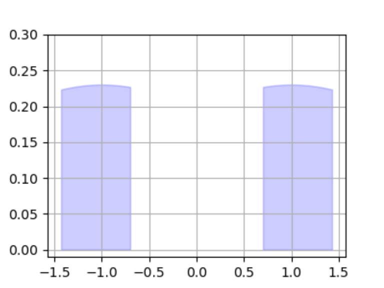


Figure 2: Initial Distribution of RReLU slopes ($\mathbf{b}_l$)

[9] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." In Proceedings of the IEEE international conference on computer vision, pp. 1026-1034. 2015.

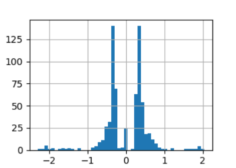
Revised baselines - strong competitors

Dataset	CIFAR-10				
Architecture	ResNet-20	ResNet-56	ResNet-110-pre	ResNet-164-pre	WRN-40-4
Acc ReLU (200 epochs)	91.25	93.03	93.63	94.58	95.47
Acc ReLU (1200 epochs)	**93.12**	**94.45**	**95.33**	**95.51**	**96.18**

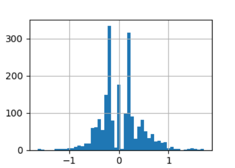
Table 1: More training improves the validation accuracy, consistent with the findings of Nakkiran et al.[10]

[10] Nakkiran, Preetum, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. "Deep double descent: Where bigger models and more data hurt." Journal of Statistical Mechanics: Theory and Experiment 2021, no. 12 (2021): 124003.

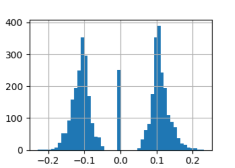
Distribution of $\mathbf{b}_l$



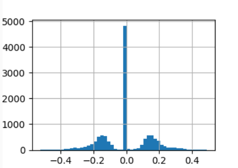
(3.1) ResNet-20



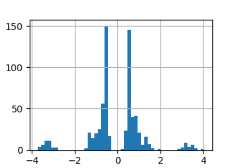
(3.2) ResNet-56



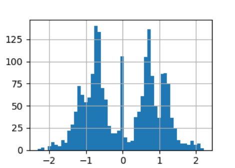
(3.3) ResNet-110-pre



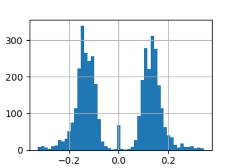
(3.4) ResNet-164-pre



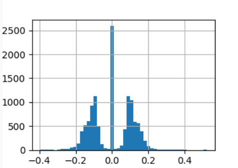
(3.5) ResNet-20



(3.6) ResNet-56



(3.7) ResNet-110-pre



(3.8) ResNet-164-pre

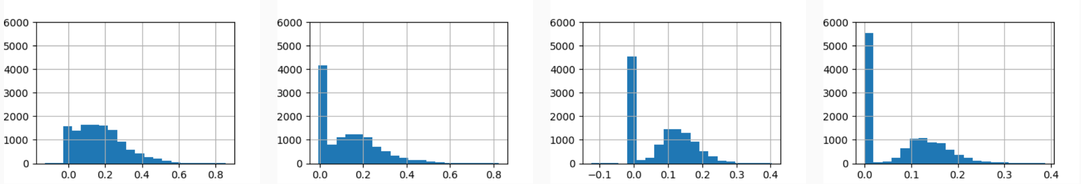
Figure 3: Distribution of $\mathbf{b}_l$ with CIFAR-10 (top) and CIFAR-100 (bottom). ResNet-N denotes ResNet of depth N.

Intrinsic sparsity with RReLU

Dataset	CIFAR-10				
Architecture	ResNet-20	ResNet-56	ResNet-110-pre	ResNet-164-pre	WRN-40-4
Acc ReLU (more training)	**93.12**	**94.45**	**95.33**	**95.51**	**96.18**
#Params ReLU	0.27	0.85	1.7	1.7	8.9
#FLOPs ReLU	81	251	505	478	2605
Filters pruned (%)	3.86	8.78	6.05	45.34	43.36
Acc RReLU (post-pruning)	92.86	94.11	95.11	95.10	96.01
#Params RReLU	**0.25**	**0.78**	**1.59**	**0.92**	**3.26**
#FLOPs RReLU	**78**	**206**	**454**	**307**	**1245**

Table 2: Performance of RReLU in terms of accuracy, number of trainable parameters, and computation power (in FLOPs) when trained from scratch. The number of parameters and FLOPs are in Millions (Mn).

Comparing performance of RReLU with Liu et al.[11]



(4.1) ResNet-164 **(4.2)** ResNet-164-L1BN-MT **(4.3)** ResNet-164-RReLU **(4.4)** ResNet-164-L1RReLU

Figure 4: Effect of RReLU on BN scaling parameters γ_l with ResNet164 on CIFAR10 dataset.

- With L1BN, many elements of γ_l converge in the range $0 < \gamma_l^i \leq 0.1$
- With L1RReLU, these values tend towards higher magnitudes or concentrate around values close to zero, indicating more flexibility with RReLU

[11] Liu, Zhuang, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. "Learning efficient convolutional networks through network slimming." In Proceedings of the IEEE international conference on computer vision, pp. 2736-2744. 2017.

Contd.

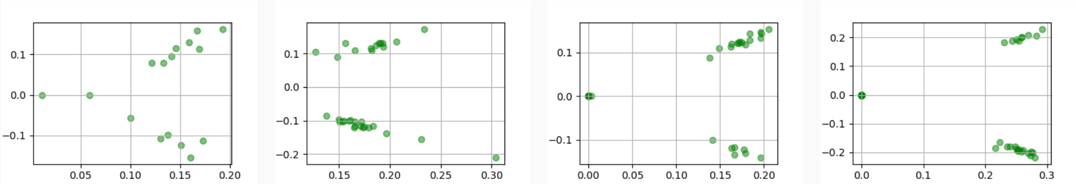
Methods	Baseline	Pruning methods		
Architecture	ResNet-164	ResNet-164-L1BN-MT (Liu et al.)[12]	ResNet-164-RReLU (Proposed)	ResNet-164-L1RReLU (Proposed)
Acc (with CIFAR10)	94.75	95.10	**95.10**	**95.42**
Filters pruned (%)	–	44 [13]	**45.34**	**48.41**
Params in Mn(% saving)	1.71	1.22(28.65%)	**0.92**(46.2%)	**0.83**(51.5%)
FLOPs in Mn(% saving)	478	358(25.1%)	**307**(35.77%)	**284**(40.58%)

Table 3: Pruning capability of RReLU. Percentage values inside parentheses indicate corresponding savings.

[12] Liu, Zhuang, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. "Learning efficient convolutional networks through network slimming." In Proceedings of the IEEE international conference on computer vision, pp. 2736-2744. 2017.

[13] Liu et al. trained the ResNet164-L1BN model for 160 epochs, after which only 31% of the filters could be removed without any degradation in accuracy (94.8%), which resulted in 19.3% saving in memory and 26.6% saving in FLOP.

Type of values $(\gamma_l, \mathbf{b}_l)$ take for ResNet-164-L1RReLU



(5.1) 8^{th} residual block **(5.2)** 20^{th} residual block **(5.3)** 36^{th} residual block **(5.4)** 49^{th} residual block

Figure 5: Plot of RReLU slopes $(\mathbf{b}_l)$ along y-axis vs. BN parameters (γ_l) along x-axis in different residual blocks of ResNet-164-L1RReLU.

- As regularization is applied to $\mathbf{b}_l$, it compels these parameters to adopt smaller values
- The term $\gamma_l \mathbf{b}_l$ can take any value in the real line, as the elements of γ_l are not regularized
- With more depth, more number of filters could be pruned as more number of $(\mathbf{b}_l, \gamma_l)$ is close to zero

Accuracy vs FLOP reduction

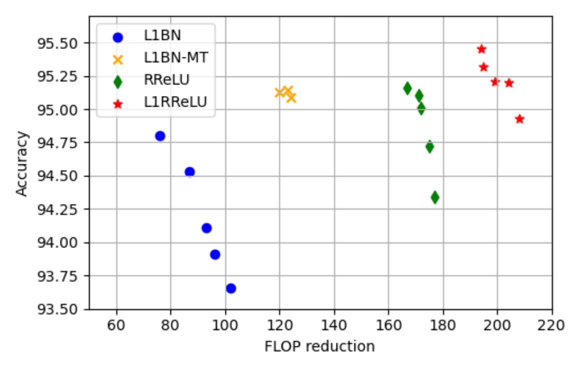


Figure 6: Acc vs FLOP reduction. The proposed methods (RReLU, L1RReLU) are compared with L1BN and L1BN-MT (MT: more training). Different points are for models with different threshold (chosen by cross-validation) for pruning $\mathbf{b}_l$.

RReLU as the coarse feature extractor

- Only the RReLU slopes $\mathbf{b}_l$ are trained whereas the weights are fixed after Kaiming He initialization [14]

Dataset	CIFAR-10		CIFAR-100	
Architecture	ResNet20	ResNet56	ResNet20	ResNet56
Acc ReLU (standard)	91.25	93.03	68.20	69.99
Acc (coarse feature extractor)	45.12	51.42	8.02	10.54

Table 4: RReLU extracts the coarse features with $\mathbf{b}_l$ being only the trainable parameters.

[14] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." In Proceedings of the IEEE international conference on computer vision, pp. 1026-1034. 2015.

Features choose the shortest filter-path length

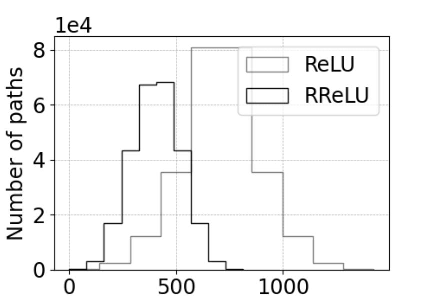


Figure 7: Distribution of filter-path length for WRN-40-4 with CIFAR-100.

- Only shorter paths carry gradients despite using deeper architecture for Residual networks[15]
- Features try to pass through a lesser number of filters as well
- Metric: filter-path length (number of filters the features pass through)

[15] Veit, Andreas, Michael J. Wilber, and Serge Belongie. "Residual networks behave like ensembles of relatively shallow networks." Advances in neural information processing systems 29 (2016).

Applicability of Rotation to GELU activation

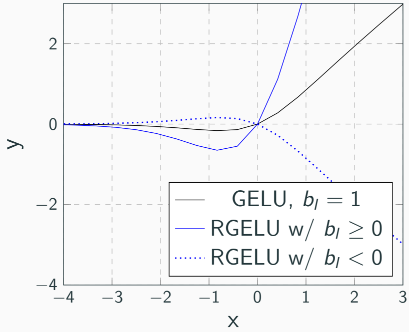


Figure 8: Rotated GELU activation $\sigma_{GELU}(\mathbf{x}_l; \mathbf{b}_l)$

- Transformers exhibit improved performance when employing GELU activations[16]

$$\sigma_{GELU}(x) = xP(X \leq x) = x\phi(x)$$
$$= x.\frac{1}{2}\left[1 + \text{erf}(\frac{x}{\sqrt{2}})\right].$$

- To introduce varying slopes in the GELU activation, we propose *RGELU*, as follows:

$$\mathbf{h}_{l+1} = \sigma_{RGELU}(\mathbf{x}_l; \mathbf{b}_l) = \mathbf{b}_l \mathbf{x}_l.\frac{1}{2}\left[1 + \text{erf}(\frac{\mathbf{x}_l}{\sqrt{2}})\right]$$

[16] Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." (2018).

Scalable across larger dataset and various architectures

Arc	Activation	Filters ignored (%)	Accuracy	Params(Mn)	Pruned	FLOP(Mn)	Pruned
VIT-s16-MLP	GELU	-	77.5	14.2	-	28.3	-
VIT-s16-MLP	RGELU	6.32	**80.1**	**13.2**	6.32%	**26.5**	6.32%
WRN-50-2	ReLU	-	76.682	21385.8	-	67.4	-
WRN-50-2	RReLU	**25.34**	76.58	**18471.0**	13.6%	**55.2**	18.1%

Table 5: Applying Rotation on ReLU and GELU activation with Imagenet dataset.

RReLU against adversarial attacks

- Considering a function $f : \mathcal{X} \rightarrow \mathbb{R}^C$ as a neural network classifier with C classes, Lipschitzness of f is closely related to its robustness
- Better adversarial robustness by imposing a tighter upper bound on the network's local Lipschitz constant (LLC) [17]
- A function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ is said to be Lipschitz continuous if $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^m$, $|f(\mathbf{x}) - f(\mathbf{y})| \leq L_p ||\mathbf{x} - \mathbf{y}||_q$ where $L_p = \sup\{||\nabla f(x)||_q : x \in \mathbb{R}^m\}$ is the Lipschitz Constant (LC), $\nabla f(x)$ is gradient of function $f(x)$, $1/p + 1/q = 1$, $1 \leq p$ and $q \leq \infty$
- Many of the RReLU slopes tend to take smaller values than one, LLC of RReLU will be smaller than LLC of ReLU

[17] Yang, Yao-Yuan, Cyrus Rashtchian, Hongyang Zhang, Russ R. Salakhutdinov, and Kamalika Chaudhuri. "A closer look at accuracy vs. robustness." Advances in neural information processing systems 33 (2020): 8588-8601.

RReLU against adversarial attacks (contd.)

Architecture	activation	LLC[18]	Attack type	Adv test acc
ResNet-20	ReLU	1.33	FGSM	34.11
			PGD	38.20
	RReLU	1.2	FGSM	**39.84**
			PGD	**42.46**
ResNet-56	ReLU	1.41	FGSM	59.01
			PGD	16.45
	RReLU	1.30	FGSM	**66.85**
			PGD	**17.05**

Table 6: RReLU to boost local smoothness and hence adversarial accuracy.

[18] Empirical computation of the Local Lipschitz Constant (LLC)

$$\frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{x}'_i \in \mathbb{B}_\infty(\mathbf{x}_i,\epsilon)} \frac{||f(\mathbf{x}_i) - f(\mathbf{x}'_i)||_{KL}}{||\mathbf{x}_i - \mathbf{x}'_i||_\infty}. \tag{3}$$

Summary

- RReLU activation improves representation power corresponding to every filter
- It induces structural sparsity
- Scalable with bigger dataset like Imagenet and various architectures including Vision Transformers with GELU activation
- It provides adversarial robustness

Publications

Related publication (Under review at TMLR)

1. Nayak, Nancy, and Sheetal Kalyani. "Rotate the ReLU to implicitly sparsify deep networks." arXiv preprint arXiv:2206.00488 (2022).

Other accepted publications

1. Nayak, Nancy, Vishnu Raj, and Sheetal Kalyani. "Leveraging online learning for CSS in frugal IoT network." IEEE Transactions on Cognitive Communications and Networking 6, no. 4 (2020): 1350-1364.
2. Vikas, Devannagari, Nancy Nayak, and Sheetal Kalyani. "Realizing neural decoder at the edge with ensembled bnn." IEEE Communications Letters 25, no. 10 (2021): 3315-3319.
3. Nayak, N., Raj, V. and Kalyani, S. "[Re] A comprehensive study on binary optimizer and its applicability." ReScience C: 6 pp. #9 (2).
4. Raj, Vishnu, Nancy Nayak, and Sheetal Kalyani. "Deep reinforcement learning based blind mmwave MIMO beam alignment." IEEE Transactions on Wireless Communications 21, no. 10 (2022): 8772-8785.

Publications

Publications under-review

1. (IEEE TWC) Nayak, Nancy, Sheetal Kalyani, and Himal A. Suraweera. "A DRL Approach for RIS-Assisted Full-Duplex UL and DL Transmission: Beamforming, Phase Shift and Power Optimization." arXiv preprint arXiv:2212.13854 (2022).
2. (IEEE TCCN) Shankar, Nitin Priyadarshini, Deepsayan Sadhukhan, Nancy Nayak, and Sheetal Kalyani. "Binarized ResNet: Enabling Automatic Modulation Classification at the resource-constrained Edge." arXiv preprint arXiv:2110.14357 (2021).

Publications

Pre-prints

1. Raj, Vishnu, Nancy Nayak, and Sheetal Kalyani. "Understanding learning dynamics of binary neural networks via information bottleneck." arXiv preprint arXiv:2006.07522 (2020).
2. Nayak, Nancy, Thulasi Tholeti, Muralikrishnan Srinivasan, and Sheetal Kalyani. "Green DetNet: Computation and memory efficient DetNet using smart compression and training." arXiv preprint arXiv:2003.09446 (2020).
3. Sharma, Akshay, Nancy Nayak, and Sheetal Kalyani. "BayesAoA: A Bayesian method for Computation Efficient Angle of Arrival Estimation." arXiv preprint arXiv:2110.07992 (2021).

Thank you!

Structural sparsity with RReLU for CNN

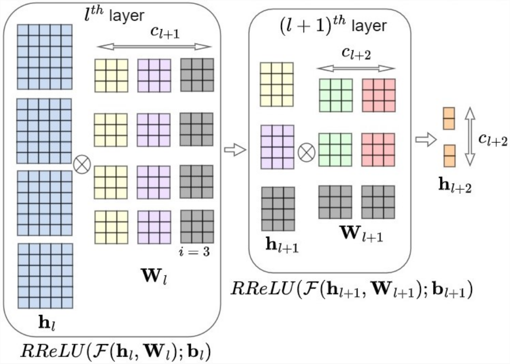


Figure 9: RReLU in CNN.

- $\otimes$ denotes the convolution operation
- At the l^{th} layer, four 2D features $\mathbf{h}_l$ are convolved with three set of filters denoted by $\mathbf{W}_l$ with four sub-filters each, followed by batchnorm and RReLU activation, resulting in $\mathbf{h}_{l+1}$
- The first 2D feature (yellow) of $\mathbf{h}_{l+1}$ is calculated by convolving each of the four 2D features in $\mathbf{h}_l$ with corresponding sub-filters of the first filter (yellow) and adding them

Structural sparsity with RReLU for CNN

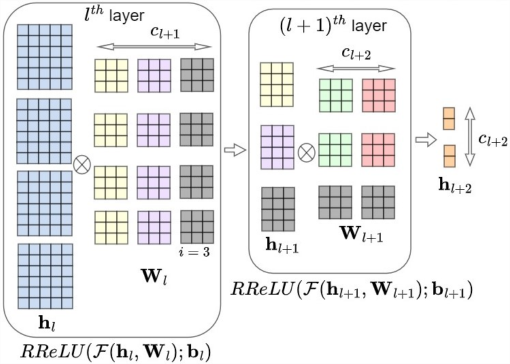


Figure 9: RReLU in CNN.

- After the training, if the slope $b_l^{\{3\}} \to 0$, then 3^{rd} feature in $\mathbf{h}_{l+1}$ is close to zero
- Then the 3^{rd} filter of $\mathbf{W}_l$ and the 3^{rd} sub-filter of every filter in $\mathbf{W}_{l+1}$ can be ignored (grey)

Structural sparsity with RReLU for CNN

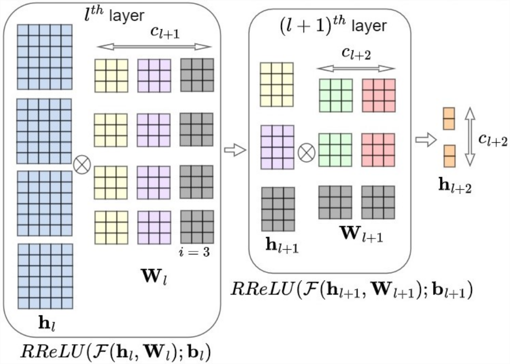


Figure 9: RReLU in CNN.

- $\mathbf{W}_l \in \mathbb{R}^{c_{l+1} \times c_l \times k \times k}$ is the filter for the l^{th} layer of a 2D CNN
- c_l and c_{l+1} represent the number of input and output channels at the l^{th} layer, respectively
- k is the dimension of the filter
- The input and output for the l^{th} layer are $\mathbf{h}_l \in \mathbb{R}^{c_l \times \bar{h}_l^w \times \bar{h}_l^h}$ and $\mathbf{h}_{l+1} \in \mathbb{R}^{c_{l+1} \times \bar{h}_{l+1}^w \times \bar{h}_{l+1}^h}$ respectively
- $(\bar{h}_l^w, \bar{h}_l^h)$ and (h_{l+1}^w, h_{l+1}^h) are spatial dimensions (width, height) of the input and the output respectively

Structural sparsity with RReLU for CNN

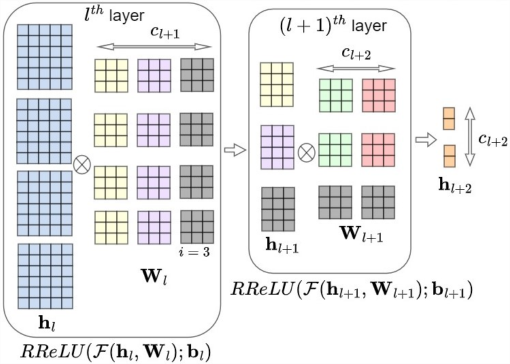


Figure 9: RReLU in CNN.

- The total number of multiplication for the l^{th} layer is $c_l k^2 \bar{h}^w_{l+1} \bar{h}^h_{l+1} c_{l+1}$
- The total number of addition for the l^{th} layer is $(c_l - 1)(k^2 - 1) \times \bar{h}^w_{l+1} \bar{h}^h_{l+1} c_{l+1}$
- The total count of FLOPs is the summation of the number of multiplication and addition $\approx 2\times$ the number of multiplication $= 2 c_l k^2 \bar{h}^w_{l+1} \bar{h}^h_{l+1} c_{l+1}$

Structural sparsity with RReLU for CNN

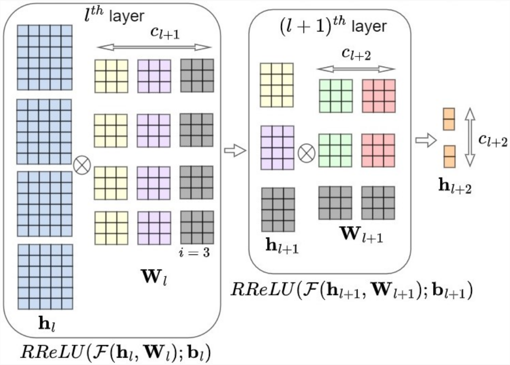
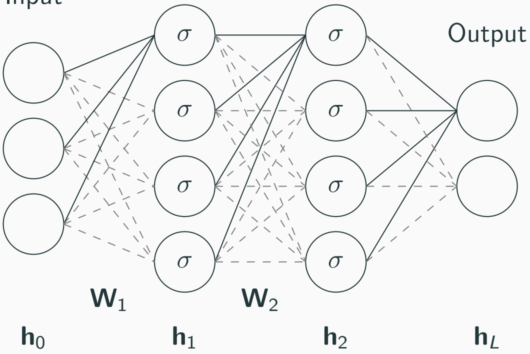


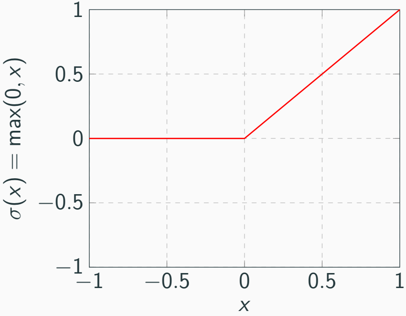
Figure 9: RReLU in CNN.

- If the output of RReLU at the l^{th} layer has c_{l+1} channels and n entries of $\mathbf{b}_l$ are insignificant, then only $(c_{l+1} - n)$ channels remain significant
- Saving Memory: Leads to saving $(c_{l+1} - n)c_l k^2$ parameters for the l^{th} layer and $c_{l+2}(c_{l+1} - n)k^2$ parameters for the $(l+1)^{th}$ layer
- Saving Computation: FLOP is reduced to $2c_l k^2 \bar{h}^w_{l+1} \bar{h}^h_{l+1}(c_{l+1} - n)$ and $2(c_{l+1} - n)k^2 \bar{h}^w_{l+2} \bar{h}^h_{l+2} c_{l+2}$ for the l^{th} layer and $(l+1)^{th}$ layer respectively

ReLU in a DNN

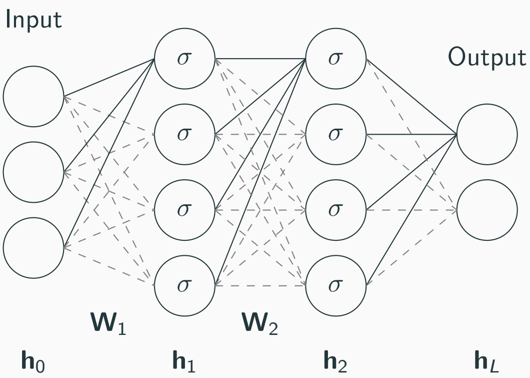


- Output of l^{th} hidden layer
  $\mathbf{h}_{l+1} = \sigma(\mathcal{F}(\mathbf{h}_l; \mathbf{W}_l))$ where σ is ReLU

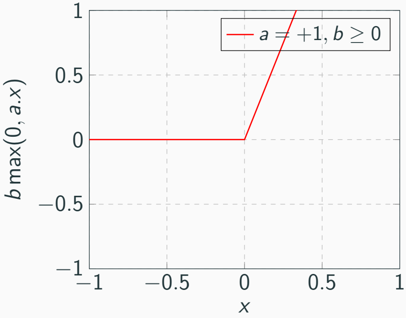


ReLU activation

Rotated ReLU in a DNN

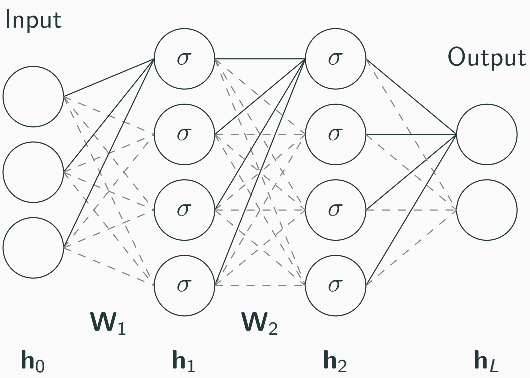


- Increase the degree of freedom of ReLU by rotating the linear part

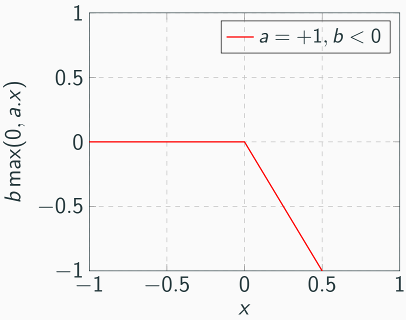


$$\sigma(x; a, b) = b\max(0, a.x)$$

Rotated ReLU in a DNN

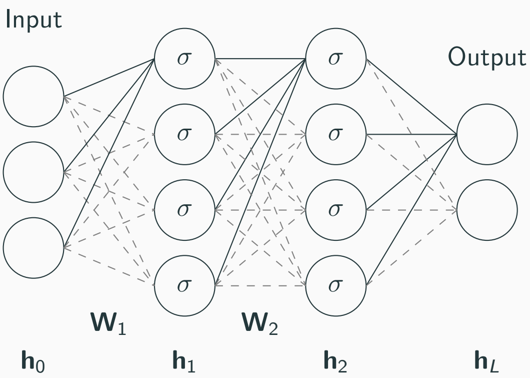


- Increase the degree of freedom of ReLU by rotating the linear part

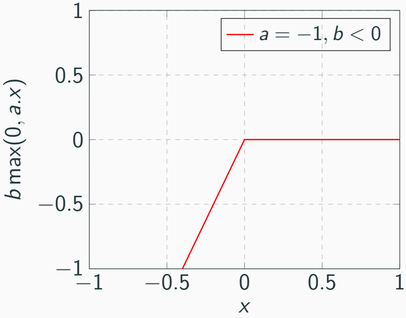


$$\sigma(x; a, b) = b \max(0, a.x)$$

Rotated ReLU in a DNN

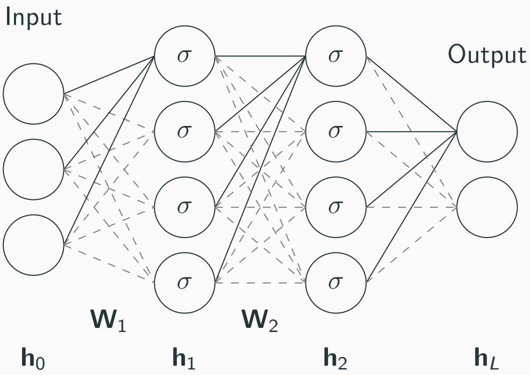


- Increase the degree of freedom of ReLU by rotating the linear part

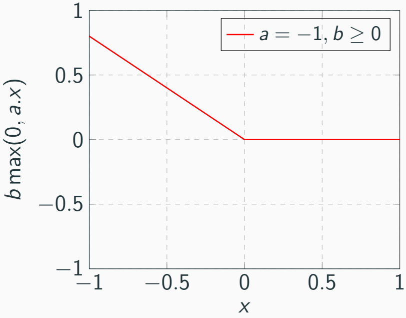


$$\sigma(x; a, b) = b \max(0, a.x)$$

Rotated ReLU in a DNN

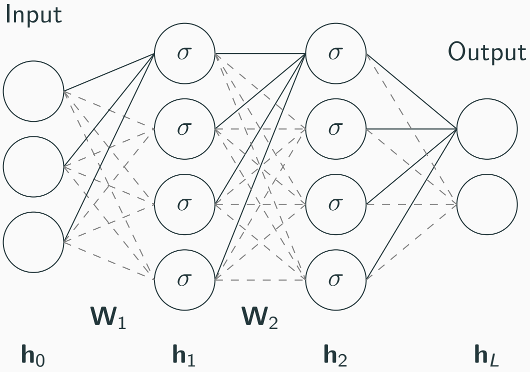


- Increase the degree of freedom of ReLU by rotating the linear part



$$\sigma(x; a, b) = b \max(0, a.x)$$

Rotated ReLU in a DNN



- The output of the l^{th} layer is

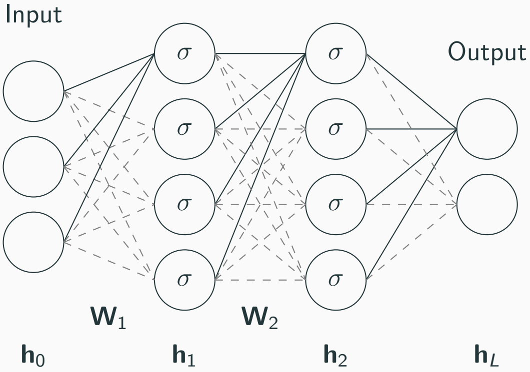
$$\mathbf{h}_{l+1} = \sigma(\mathbf{x}_l; \mathbf{a}_l, \mathbf{b}_l) = \mathbf{b}_l \max(0, \mathbf{a}_l . \mathbf{x}_l),$$

  where $\mathbf{x}_l = \mathcal{F}(\mathbf{h}_l; \mathbf{W}_l)$

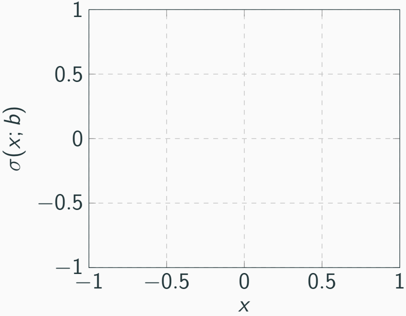
- Any value of $\mathbf{a}_l$ can be adjusted using the weights/filters $\mathbf{W}_l$
- Now, the output of the l^{th} layer is

$$\mathbf{h}_{l+1} = \sigma(\mathbf{x}_l; \mathbf{b}_l) = \mathbf{b}_l \max(0, \mathbf{x}_l)$$

Rotated ReLU in a DNN

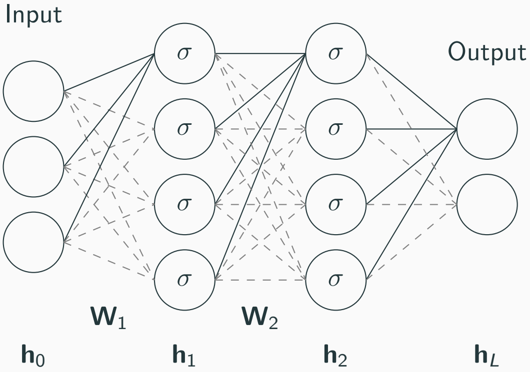


- Two different types of RReLU corresponding to $b \geq 0$ and $b < 0$ are sufficient to learn

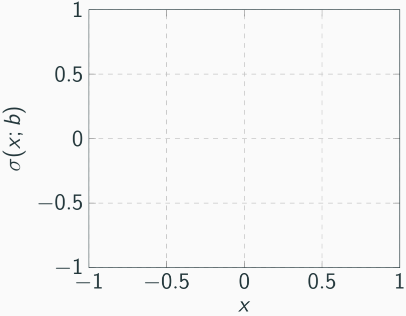


RReLU activation $\sigma(x; b) = b \max(0, x)$

Rotated ReLU in a DNN

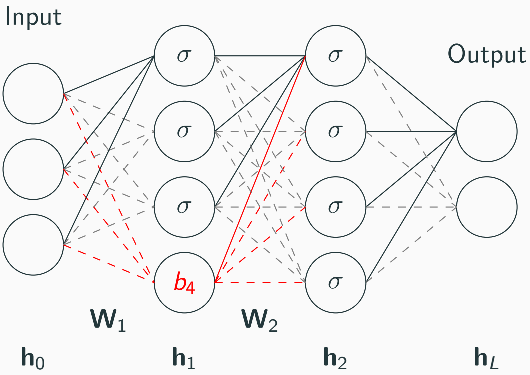


- Two different types of RReLU corresponding to $b \geq 0$ and $b < 0$ are sufficient to learn



RReLU activation $\sigma(x; b) = b \max(0, x)$

Rotated ReLU in a DNN



- $\mathbf{W}_1 = \begin{bmatrix} w_1^{11} & w_1^{12} & w_1^{13} \\ w_1^{21} & w_1^{22} & w_1^{23} \\ w_1^{31} & w_1^{32} & w_1^{33} \\ \color{red}{w_1^{41}} & \color{red}{w_1^{42}} & \color{red}{w_1^{43}} \end{bmatrix}$ and

$$\mathbf{W}_2 = \begin{bmatrix} w_2^{11} & w_2^{12} & w_2^{13} & \color{red}{w_2^{14}} \\ w_2^{21} & w_2^{22} & w_2^{23} & \color{red}{w_2^{24}} \\ w_2^{31} & w_2^{32} & w_2^{33} & \color{red}{w_2^{34}} \\ w_2^{41} & w_2^{42} & w_2^{43} & \color{red}{w_2^{44}} \end{bmatrix}$$

- Highlighted connections are unimportant if $b_4 \to 0$

Exhaustive experiments

Data-sets

- MNIST
- CIFAR-10
- CIFAR-100
- SVHN
- Imagenet

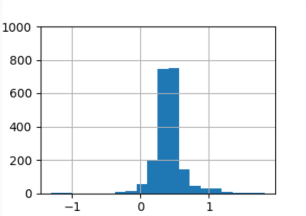
Architectures

- FCNN
- ResNet-(20/56/110-pre/164-pre)
- WideResNet-(40/16)-4
- WideResNet-50-2
- Vision Transformer s16

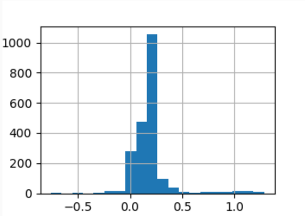
Compute facility

- For experiments with Imagenet dataset: NVIDIA-A100 GPU
- For others: NVIDIA-GeForce 2080 Ti GPU

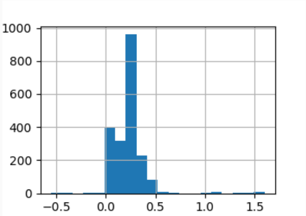
Effect on the distribution of γ_l- proving better representation with RReLU



(10.1) ReLU, 200 epochs: 50 values of γ_l are close to zero.



(10.2) ReLU, 1200 epochs: 270 values of γ_l are close to zero. Prolonged training facilitates more pronounced adjustments of γ_l, allowing for more filters to be disregarded.



(10.3) RReLU, 1200 epochs: the number of values of γ_l close to zero increases to 400. The sparsity further increases with RReLU.

Figure 10: Distribution of batchnorm parameters γ_l after the training when the architecture considered is ResNet56 on CIFAR-10 dataset. Subfig. (5.2) shows the effect of more training on γ_l. Subfig. (c) is the same with RReLU.

How does RReLU provide compact model?

- Architectures with RReLU achieves the same performance as architectures with ReLU with **fewer trainable filters**
- As $x_l^{\{i\}}$ is batch normalized, it can take only bounded values
- If the value of $b_l^{\{i\}}$ for i^{th} feature, $x_l^{\{i\}}$ is comparatively less, then the feature $x_l^{\{i\}}$ is not essential for the task and can be ignored keeping the performance intact
- We prune weights/filters based on the corresponding RReLU slopes in $\mathbf{b}_l$

RReLU against adversarial attacks (contd.)

- Adversarial attacks
  - Fast Gradient Sign Method (FGSM)

$$\mathbf{x}^{adv} = \mathbf{x} + \epsilon \, \mathrm{Sign}(\nabla_{\mathbf{x}} J(f(\mathbf{x}), y)) \tag{4}$$

  - Projected Gradient Descent (PGD)

$$\mathbf{x}_{i+1}^{\mathrm{adv}} = \mathrm{Proj}_{B_{\xi}(\mathbf{x})} \left(\mathbf{x}_i^{\mathrm{adv}} + \eta \, \mathrm{Sign} \left(\nabla_{\mathbf{x}_i^{\mathrm{adv}}} J(f(\mathbf{x}_i^{\mathrm{adv}}), y) \right) \right) \tag{5}$$

- Empirical computation of the Local Lipschitz Constant (LLC)

$$\frac{1}{n} \sum_{i=1}^{n} \max_{\mathbf{x}_i' \in \mathbb{B}_{\infty}(\mathbf{x}_i, \epsilon)} \frac{||f(\mathbf{x}_i) - f(\mathbf{x}'_i)||_{KL}}{||\mathbf{x}_i - \mathbf{x}'_i||_{\infty}}. \tag{6}$$