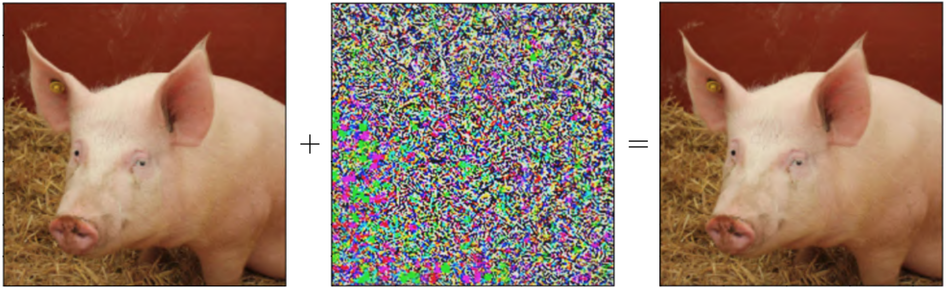
Polarizing Front Ends for Robust CNNs

Can Bakışkan, Soorya Gopalakrishnan, Metehan Çekiç, Upamanyu Madhow, Ramtin Pedarsani

UC Santa Barbara

45th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)

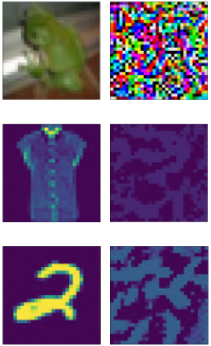
1. Introduction



Hog　　Attack (x50)　　Airliner

- Vulnerability of machine learning models is concerning
- "Excessive linearity": small perturbations designed to add up to large values in high dimensions using locally linear approximations of smooth functions

Pictures courtesy of [1]



$$\left[0, \ldots, 0, 0.9, 0, \ldots, 0.1\right]^T \qquad \left[0, \ldots, 0, 1, 0, \ldots, 0\right]^T$$

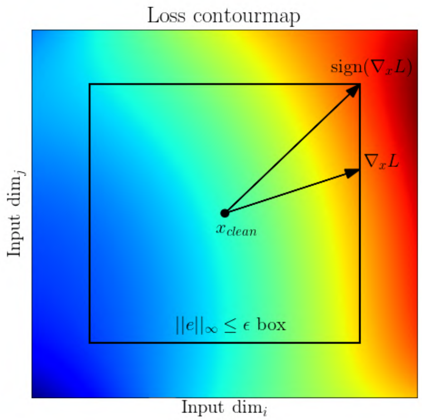
$$\left[0, \ldots, 0.2, 0.8, 0, \ldots, 0\right]^T \qquad \left[0, \ldots, 0, 1, 0, \ldots, 0\right]^T$$

$$\left[0.94, 0.14, 0, \ldots, 0.06\right]^T \qquad \left[0, \ldots, 0, 1, 0, \ldots, 0\right]^T$$

$\boldsymbol{x}$ $\quad$ $\boldsymbol{e}$ $\quad$ $\boldsymbol{y}$ $\quad$ $\boldsymbol{y}_{true}$

- Inputs $\boldsymbol{x} \in \mathbb{R}^N$, output predictions (confidence scores for M classes) $\boldsymbol{y} \in [0, 1]^M$, $\boldsymbol{\theta}$ model parameters
- Our goal is to defend against malicious inputs of the form $\boldsymbol{x} + \boldsymbol{e}$, where $\boldsymbol{e} \in \mathbb{R}^N$ is a small perturbation that aims to cause misclassification $(||\boldsymbol{e}||_\infty < \epsilon)$
- Formally: $\max_{\boldsymbol{e} \in \mathcal{S}} L(\boldsymbol{\theta}, \boldsymbol{x} + \boldsymbol{e}, \boldsymbol{y}_{\text{true}})$

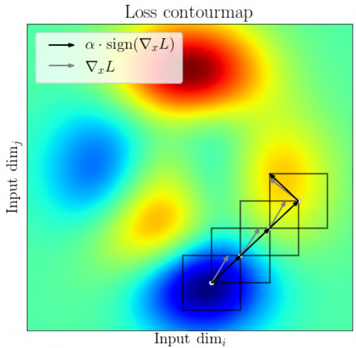
- Fast Gradient Sign Method: $\boldsymbol{e} = \epsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}} L(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}_{\text{true}}))$



FGSM takes gradient steps to **increase** loss function

- Basic Iterative Method (BIM):

$$\boldsymbol{e}_{i+1} = \mathsf{Clip}_{\epsilon}\Big(\boldsymbol{e}_i + \alpha \cdot \mathsf{sign}(\nabla_{\boldsymbol{x}} L(\boldsymbol{\theta}, \boldsymbol{x} + \boldsymbol{e}_i, \boldsymbol{y}_{\text{true}}))\Big)$$

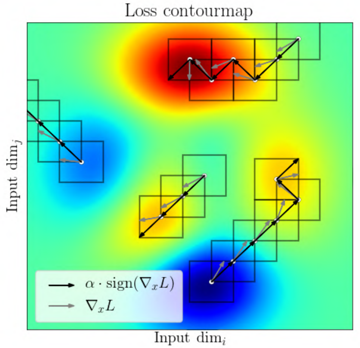


Many FGSM-like steps use local linearity better

2. Background

Attacks – PGD[4]

- Projected Gradient Descent: Basic iterative method with **random restarts** within $||\cdot||_\infty \leq \epsilon$ ball around original input.



PGD uses worst case perturbation among several initial conditions

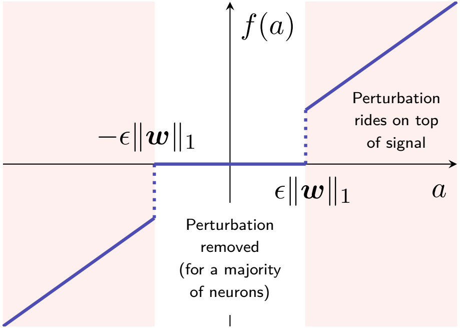
- Most well-known defenses still standing use adversarial training with attacks generated by PGD with random restarts[4]
- Defenses that combine adversarial training with other methods
- Provable defenses

- Our assumption that $||\boldsymbol{e}||_\infty \leq \epsilon$
- Hölder's inequality:

$$|\boldsymbol{w}^T\boldsymbol{e}| \leq ||\boldsymbol{e}||_\infty ||\boldsymbol{w}||_1 \leq \epsilon ||\boldsymbol{w}||_1$$

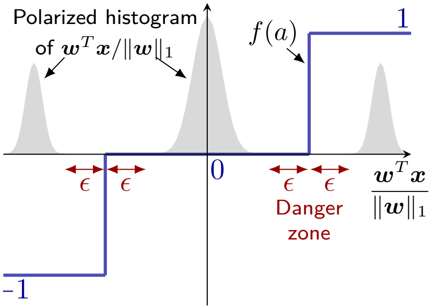
- Can predict max perturbation at output of front end neuron

- Sparsifying approach: suppress neuron outputs around zero up to $|\cdot| \leq \epsilon||\boldsymbol{w}||_1$



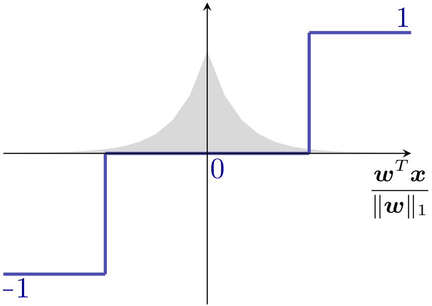
Activation sparsity alone is not enough: perturbations can ride on top of signals passing through

- Can we prevent perturbations from riding on top of the signal?
- Polarization approach:



A saturation function can fully eliminate perturbations if distribution is polarized.

- Problem: With standard training the distribution of $\boldsymbol{w}^T\boldsymbol{x}/\|\boldsymbol{w}\|_1$ is concentrated around zero. Direct enforcement of saturated activation can kill most of the signal as well as perturbation.



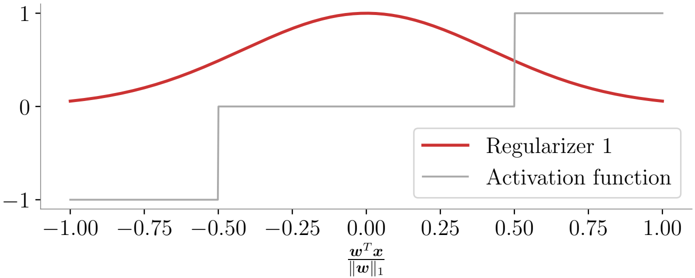
Standard training makes activations concentrate around zero.

- Solution: Multi stage training with regularizers that favor the multimodal distribution we want.

$$\mathcal{L}(\boldsymbol{y}, \boldsymbol{y}_{\text{true}}, \boldsymbol{z}) = \mathcal{L}_{CE}(\boldsymbol{y}, \boldsymbol{y}_{\text{true}}) + \lambda \sum_{k=1}^{K} \frac{B(z_k)}{K}$$

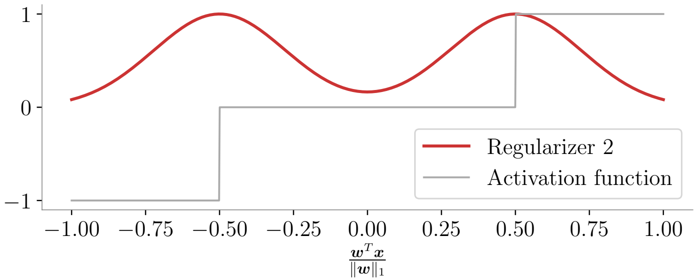
- Goal: First spread the distribution between $[-1, 1]$. Then drive the distribution of $\boldsymbol{w}^T \boldsymbol{x} / \|\boldsymbol{w}\|_1$ away from the "danger zones"

- No saturation function used
- Regularizer $B(z_k) = B_1(z_k) = e^{-z_k^2/2\sigma_1^2}$

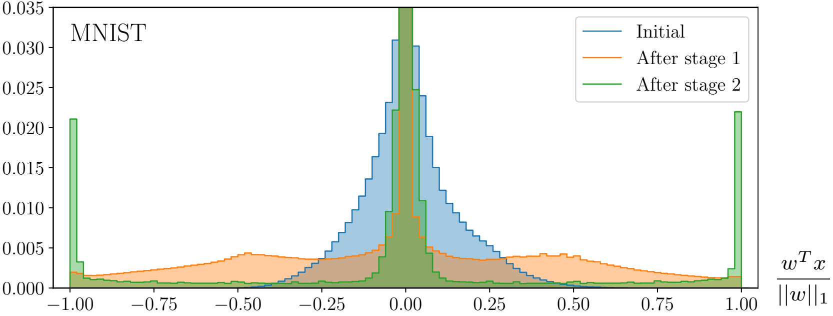


Regularizer 1 promotes even distribution of normalized activations between $[-1, 1]$. Saturation function plotted for comparison only.

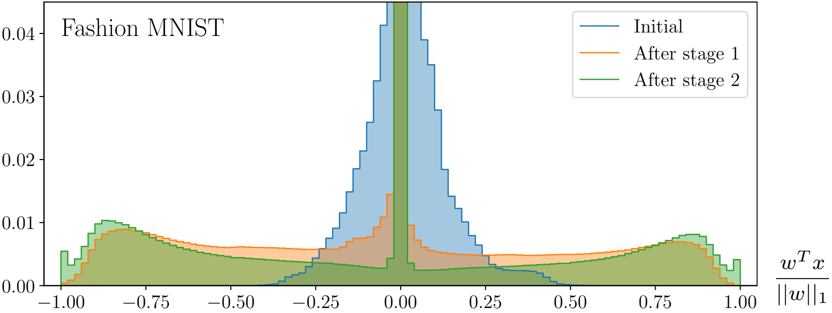
- No saturation function used
- Regularizer $B(z_k) = B_2(z_k) = e^{-(z_k-c)^2/2\sigma_2^2} + e^{-(z_k+c)^2/2\sigma_2^2}$



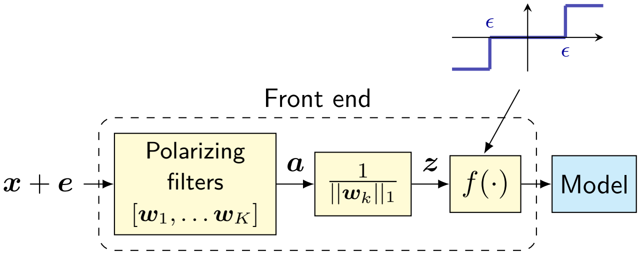
Regularizer 2 drives activations away from the regions of discontinuity



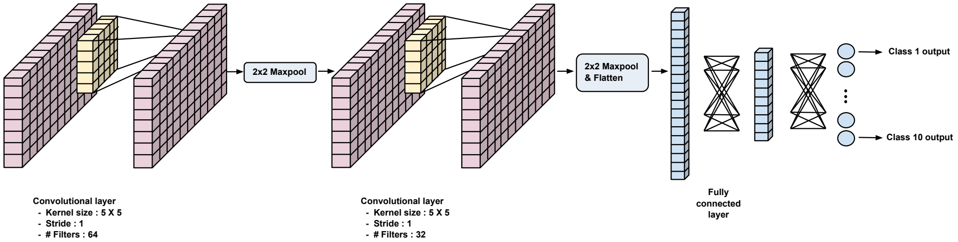
Polarization of distribution of normalized front-end filter activations (MNIST)



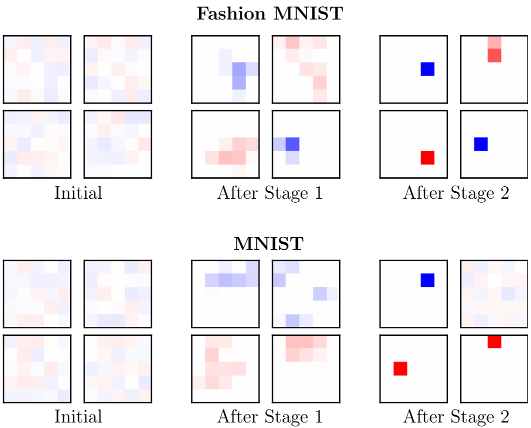
Polarization of distribution of normalized front-end filter activations (Fashion MNIST)



Block diagram of front end defense, showing a polarizing filter followed by ℓ_1 normalization and saturation activation function $f(\cdot)$.

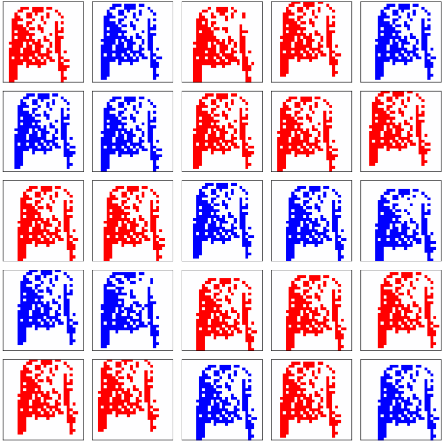


- Standard post frontend convolutional neural network (following literature for consistency).
  - 2 convolutional layers followed by
  - 2 fully connected layers.
  - Each convolutional layer is followed by maxpooling operation.



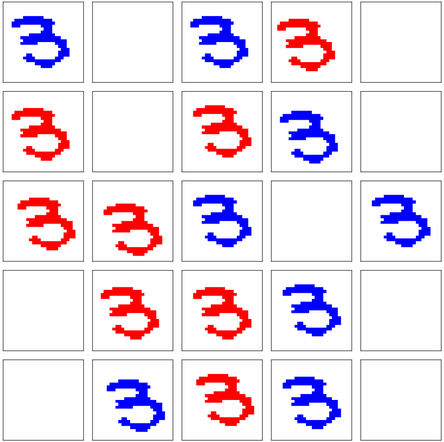
Examples of filters after each stage. As polarizing regularizers are introduced, weights of the filters start concentrating around mostly a single pixel in each filter **at different locations and signs**.

5. Results



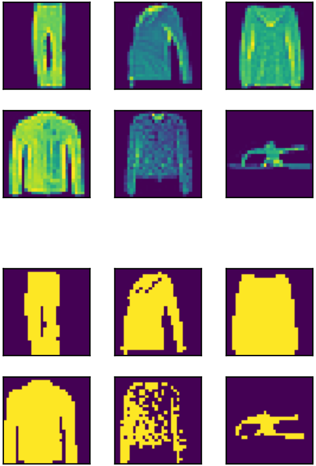
Output of the frontend for a single Fashion MNIST image. Notice the various **shifts in position**.

5. Results

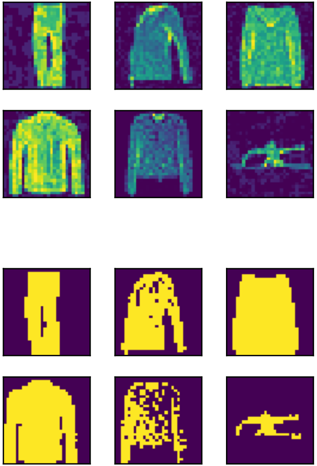


Output of the frontend for a single MNIST image. Notice the various **shifts in position**.

5. Results

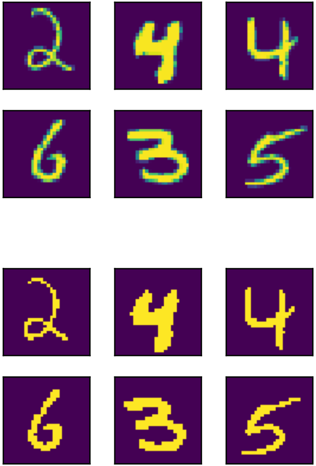


Original FashionMNIST images and **single filter's output** for the frontend

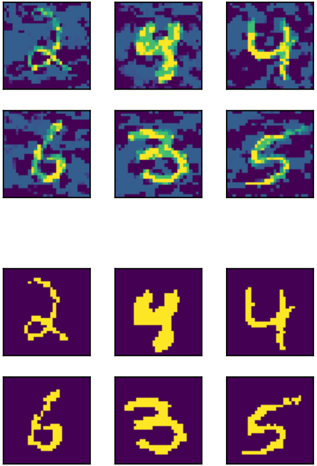


Attacked FashionMNIST images and **single filter's output** for the frontend

5. Results



Original MNIST images and **single filter's output** for the frontend

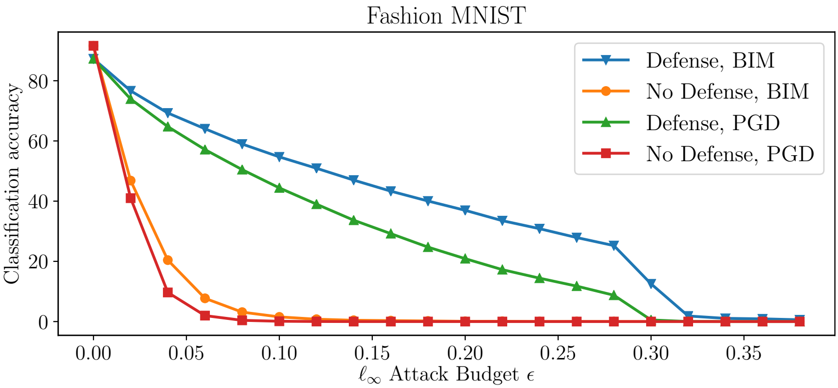


Attacked MNIST images and **single filter's output** for the frontend

5. Results

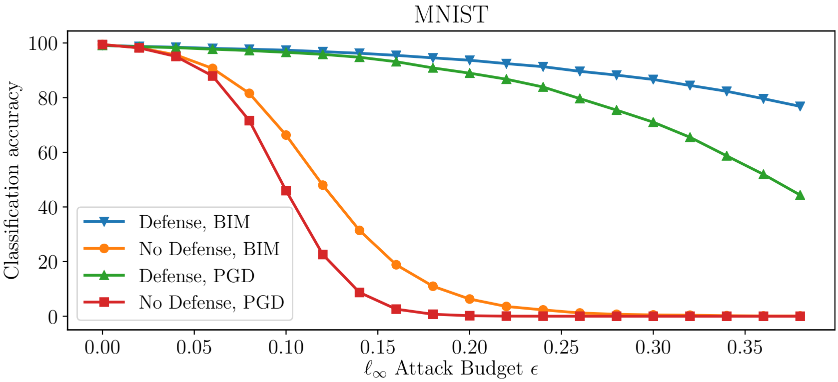
Experimental results for different attacks.

	Fashion MNIST ($\epsilon = 0.1$)				MNIST ($\epsilon = 0.3$)			
	Clean	FGSM	BIM	PGD	Clean	FGSM	BIM	PGD
No defense	91.6	19.7	1.49	0.11	99.4	21.9	0.47	0.00
Adv. Training	83.8	77.9	75.6	74.1	97.5	93.1	90.0	86.7
Ours	87.3	69.4	54.9	44.5	99.1	93.2	86.5	70.8



Classification accuracy as attack budget ϵ is increased.

5. Results



Classification accuracy as attack budget ϵ is increased.

5. Discussion

- First step in a bottom-up and interpretable approach to mitigate adversarial attacks for machine learning models
- Resulting filters consistent with what adversarial training learns
- Saturation function amounts to drastic quantization which hurts clean accuracy

6. Conclusion

- Future work:
  - Expand the same clustering/polarization idea for other datasets
  - Use different activation functions to alleviate retained perturbation at each layer
- Our code is available at
  `github.com/canbakiskan/polarizing-frontend`

References

1. adversarial-ml-tutorial.org
2. Goodfellow et al., "Explaining and harnessing adversarial examples," in International Conference on Learning Representations, 2015
3. Kurakin et al., "Adversarial machine learning at scale," in International Confer- ence on Learning Representations, 2017
4. Madry et al., "Towards deep learning models resistant to adversarial attacks," in International Conference on Learning Representations, 2018