Security and Privacy of Machine Learning

Defenses against Evasion Attacks

S. Picek

Recap of the Last Lecture

Adversarial Attacks

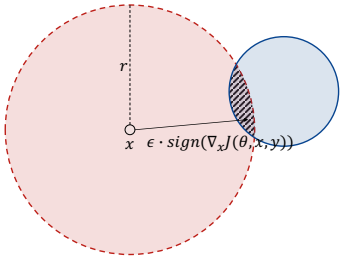


Figure: FGSM.

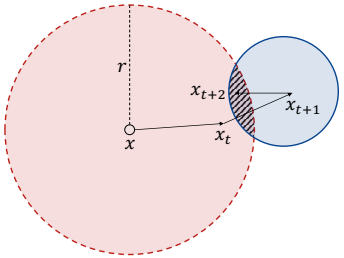


Figure: PGD.

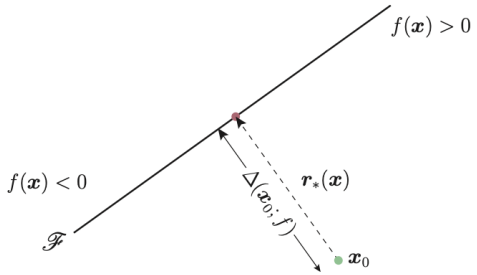


Figure: DeepFool.

- The common optimization problem for finding an adversarial example:

$$\min_{x+\eta} \|x, x+\eta\|$$

$$\text{s.t. } f(x+\eta) = l' \text{ where } f(x) = l \neq l', x+\eta \in [0,1]^n.$$

Recap of the Last Lecture

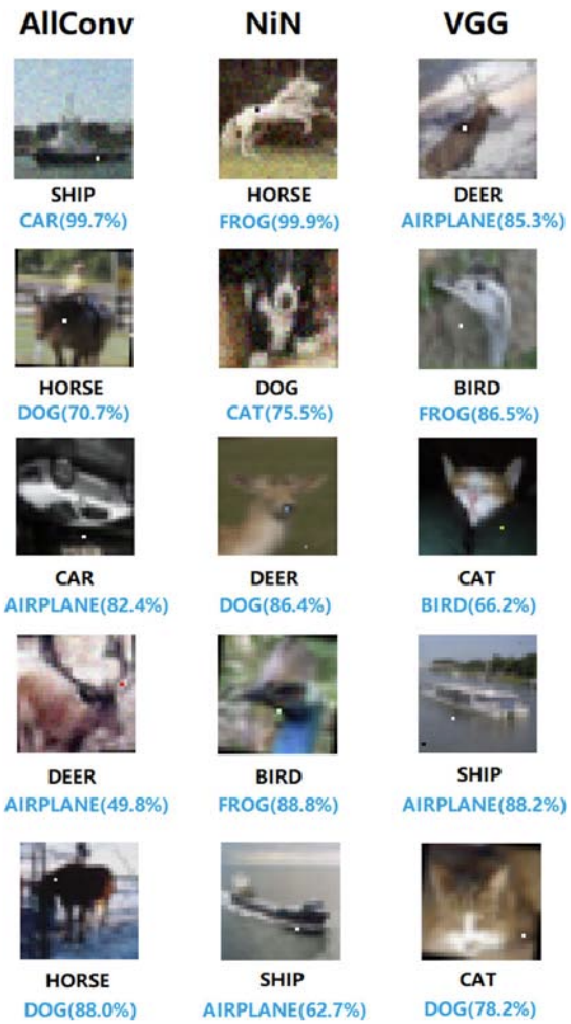


Figure: One-pixel Attack.

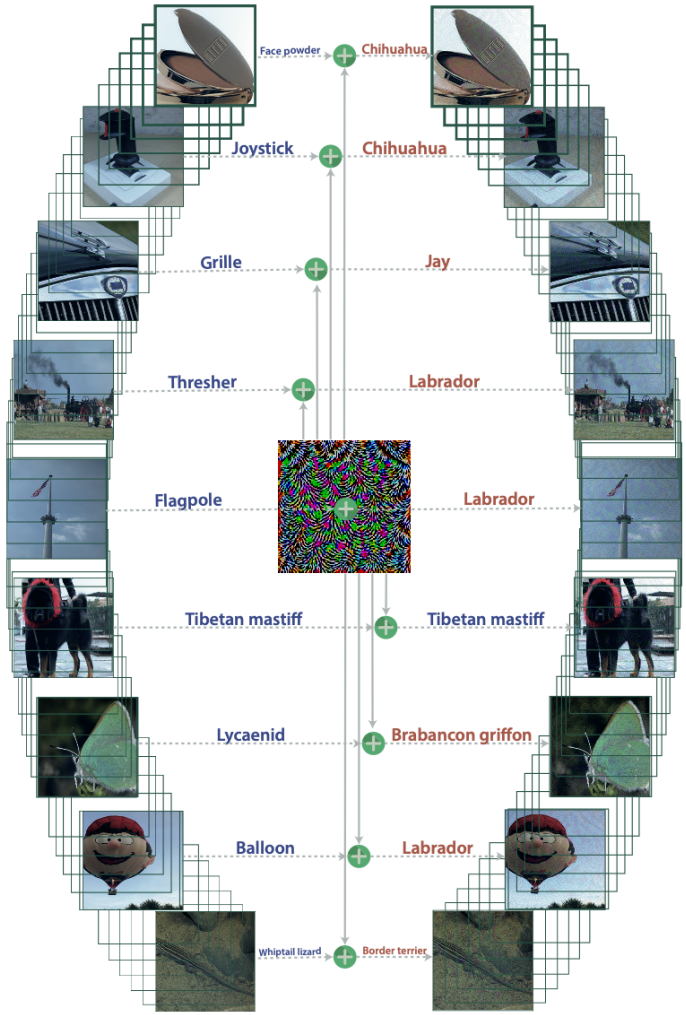


Figure: Universal Attack.

Outline of this Lecture

- Defenses
  - Obfuscated Gradients
  - Adversarial training
  - Self-Supervised Adversarial Training
  - Pruning
  - Random input transformation
  - Certified Robustness
- Generative Adversarial Networks

Interesting

- Microsofts AI Red Team just published “Lessons from Red Teaming 100 Generative AI Products”
- Lessons from red teaming 100 generative AI products

Interesting

- Understand what the system can do and where it is applied.
- You don't have to compute gradients to break an AI system.
- AI red teaming is not safety benchmarking.
- Automation can help cover more of the risk landscape.
- The human element of AI red teaming is crucial.
- Responsible AI harms are pervasive but difficult to measure.
- LLMs amplify existing security risks and introduce new ones.
- The work of securing AI systems will never be complete.

Interesting

- Building trust in AI through a cyber risk-based approach

Interesting

- The EU Commission published its final work programme late in the evening on 11 February with a final twist: the withdrawal of the AI liability directive.

- Defenses
  - Obfuscated Gradients
  - Adversarial training
  - Self-Supervised Adversarial Training
  - Pruning
  - Random input transformation
  - Certified Robustness

- Generative Adversarial Networks

Defenses Evaluation

- RobustBench
- Standardized benchmark.
- The goal of RobustBench is to systematically track the real progress in adversarial robustness.

Defenses

- Why to defend against adversarial attacks?
- First motivation: Adversarial robust networks provide better accuracy.
- Second motivation: Adversarial robust networks provide a better explanation of the behavior of networks.

General Idea

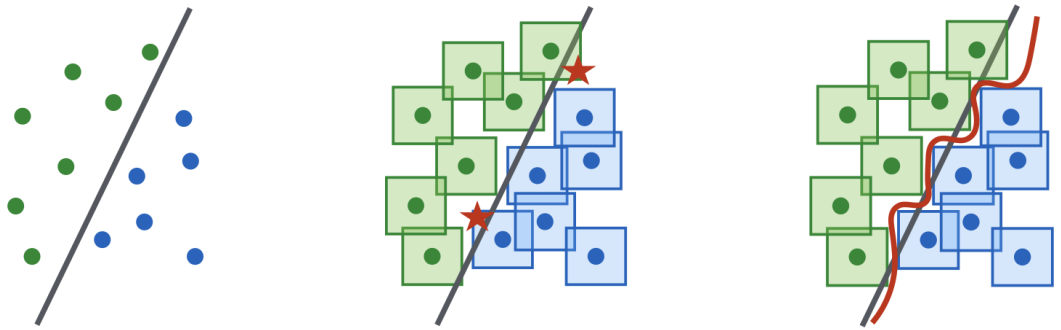
- **Defense before training**.
- The defender has control of both the training data and the model architecture.
- The goal is to filter out and/or purify the malicious or noisy data and design robust model architectures, improving the robustness of the model after training.

Robust Architecture

- For identical training data and training algorithms, models with different architectures often show significant differences in their levels of adversarial robustness.
- Do specific architectures inherently have better adversarial robustness?
- Two factors significantly influence the robustness of architectural designs: model capacity and model structure.

Robust Architecture

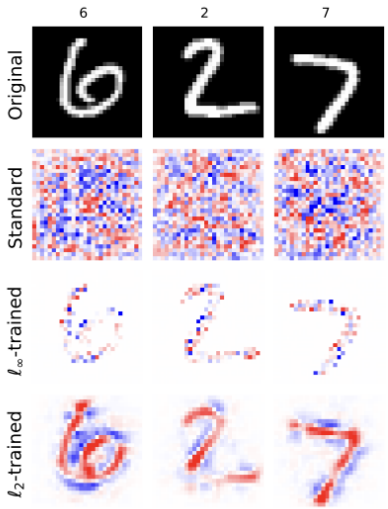
- Adversarial robust networks provide better accuracy against adversarial examples.

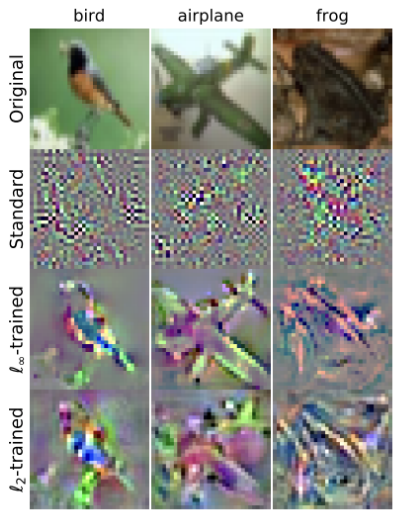


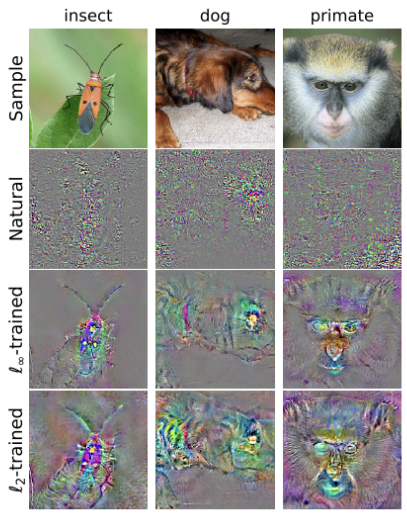
- Separating the l_∞-balls requires a significantly more complicated decision boundary.

Robust Architecture

- Adversarial robust networks provide better saliency map explanation.
- Input gradients are frequently used as explanations, but sometimes they are noisy and not interpretable on their own.
- Adversarial-trained models tend to generate smooth gradients.







Obfuscated Gradients

- As many attacks use gradients to find the adversarial perturbation, a very straightforward method is to obfuscate the gradients.
- For example, approximating a model using a discontinuous activation function.
- Gradients are unidentifiable due to discontinuities.

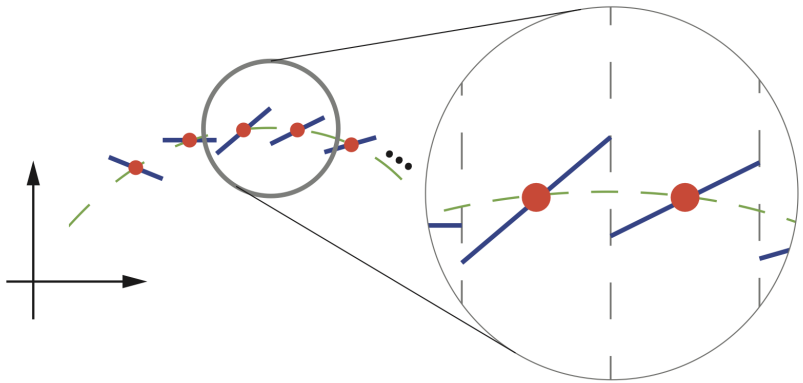


Figure: The resulting model is piece-wise continuous (blue curve), and discontinuities can be dense.[1]

[1] Chang Xiao et al. "Enhancing adversarial defense by k-winners-take-all."

Obfuscated Gradients

- Obfuscated gradient methods can increase robust accuracy.
- However, such methods are not robust and do not solve the attack problem.
- The accuracy of k-winners-take-all defense can be decreased to 0% by estimating the average local gradient.[2]

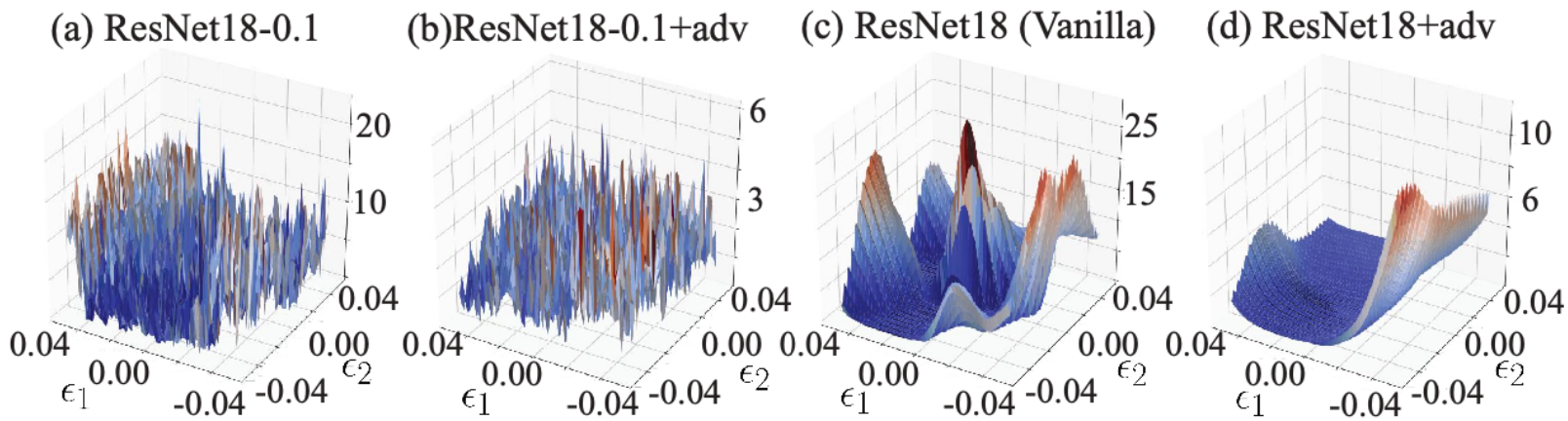


Figure: Gradient-based attacks loss landscapes in obfuscated gradients defense (a, b), and conventional ReLU models (c, d).

[2] Florian Tramer et al. "On adaptive attacks to adversarial example defenses."

Obfuscated Gradients

- In 2018, the authors of the C&W attack evaluated the robustness of 9 papers accepted to ICLR 2018 as adversarial defenses.
- 7 of 9 defenses rely on obfuscated gradients.
- Only one method significantly increases robustness to adversarial examples: PGD adversarial training.

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%*
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.005 (ℓ_2)	0%*
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%*
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%*
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	55%**
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%
Na et al. (2018)	CIFAR	0.015 (ℓ_∞)	15%

Figure: The * means the proposed method combines with adversarial training. The ** means the fundamental principle behind it has 0% accuracy.[3]

[3] https://github.com/anishathalye/obfuscated-gradients

General Idea

- **Defense during training**.
- The defender has full control of the training process, such as the objective function, the optimization algorithm, and the learning paradigm/procedures.
- The goal is to enhance the robustness of the model by training it with a carefully designed training process.

Adversarial training

- **Goal**: To build a robust network against adversarial examples.
- **Idea**: (Re)training with adversarial examples.

Adversarial training

- How to do adversarial training?

$$\min_{\theta} \max_{\eta \in S} \mathcal{L}_{CE}(\theta, x_i + \eta, y_i),$$

where θ are the weights of the model,

η is the adversarial perturbation.

- The inner maximization problem looks for adversarial examples by modifying input x_i.
- The outer minimization problem is to find a better θ to decrease the loss.
- Briefly, adversarial training refers to using adversarial examples for training.

Choosing attacks for adversarial training

- In the beginning, FGSM is a popular choice for adversarial training as it is fast to generate FGSM examples.
- But training on FGSM does not result in robustness against stronger iterative attacks such as PGD.
- Later, PGD training is considered the most successful defense, as a network trained with PGD examples is robust against a wide range of attacks.

CIFAR10

	Simple	Wide
Natural	92.7%	95.2%
FGSM	27.5%	32.7%
PGD	0.8%	3.5%

(a) Standard training

	Simple	Wide
Natural	87.4%	90.3%
FGSM	90.9%	95.1%
PGD	0.0%	0.0%

(b) FGSM training

	Simple	Wide
Natural	79.4%	87.3%
FGSM	51.7%	56.1%
PGD	43.7%	45.8%

(c) PGD training

	Simple	Wide
Natural	0.00357	0.00371
FGSM	0.0115	0.00557
PGD	1.11	0.0218

(d) Training Loss

Figure: The FGSM training model is not robust against PGD.[4]

[4] Madry et al. "Towards deep learning models resistant to adversarial attacks."

Why PGD can counter FGSM and even more attacks?

- As the loss (red) of robust models is smaller and concentrated without any outliers.
- It means those local maxima are very likely to be true maxima.
- In other words, PGD with random start points can find very diverse adversarial examples.

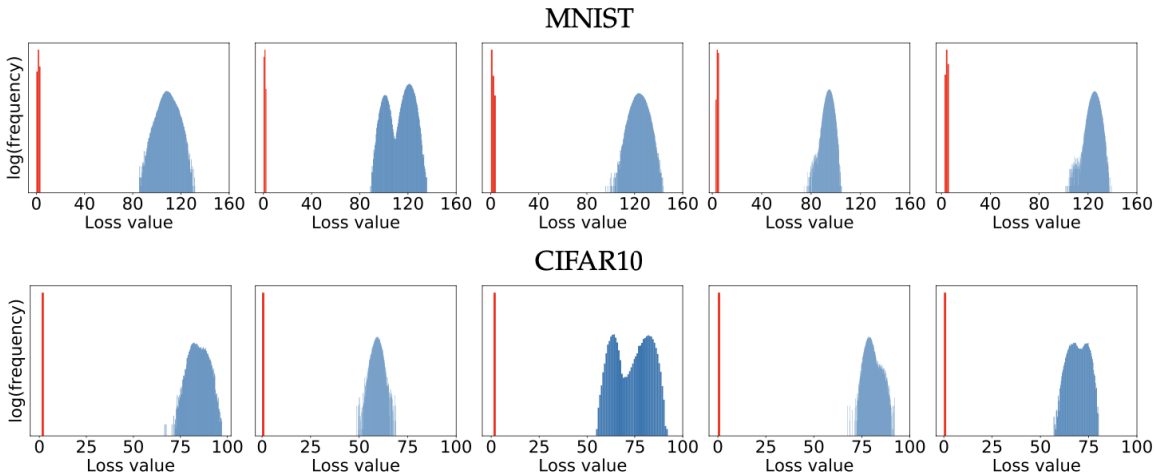
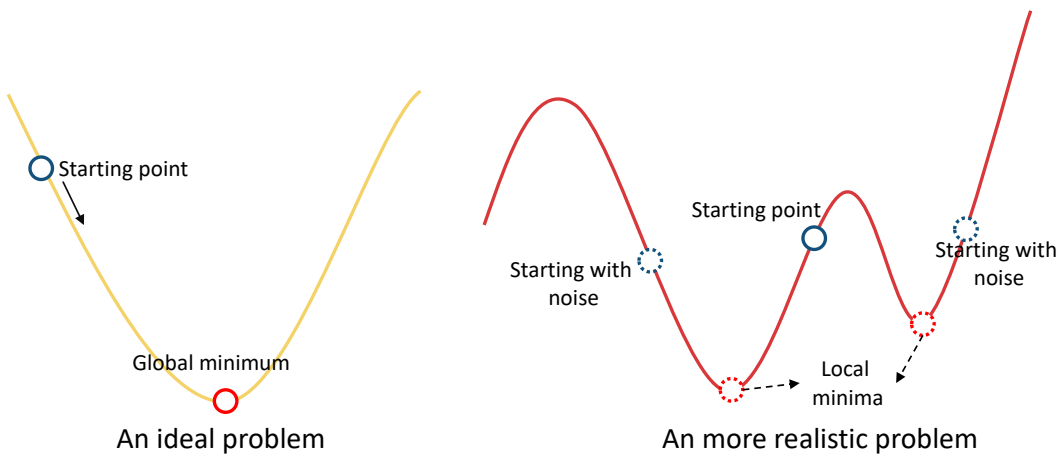


Figure: Local maxima given by the cross-entropy loss. The blue histogram corresponds to the loss on a standard network, while the red is for the adversarially trained counterpart with uniformly random points.

Why PGD can counter FGSM and even more attacks?

- Ideally, we hope there is only one global minimum for adversarial perturbation, but this is not realistic.
- We usually have multiple local minima for optimization problems in machine learning.
- Finding global minimum is too difficult, and local minima are enough to cheat the model in evasion attacks.
- With "random start", PGD can find more local minima.



How robust is adversarial training?

- Adversarial training provides better robustness, but it will reduce the accuracy on clean data.
- There is still a problem with the generalization between clean and robust accuracy.
- The best robust accuracy on CIFAR-10 is 71.07%[5], while it is easy to have clean accuracy higher than 90%.

[5] robustbench.github.io

What did adversarial training do during training?

- Adversarial training learns rather than remembers.

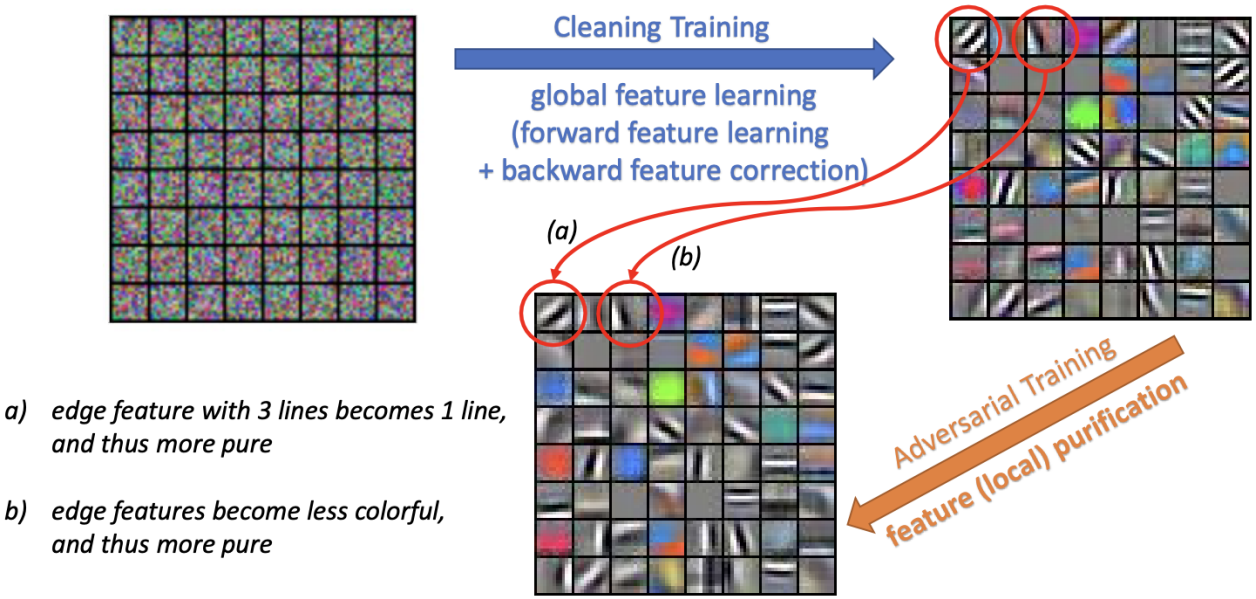


Figure: Feature purification in adversarial training (for the first layer of AlexNet on CIFAR-10). Visualization of the first layer of AlexNet on CIFAR-10.[6]

[6] Allen-Zhu Zeyuan, Li Yuanzhi. "Feature purification: How adversarial training performs robust deep learning".

What did adversarial training do during training?

An explanation of "feature purification":

- Adversarial examples are the accumulation of certain small dense mixtures in the hidden weights during the training process of a neural network.
- One of the goals of adversarial training is to remove such mixtures to purify hidden weights.

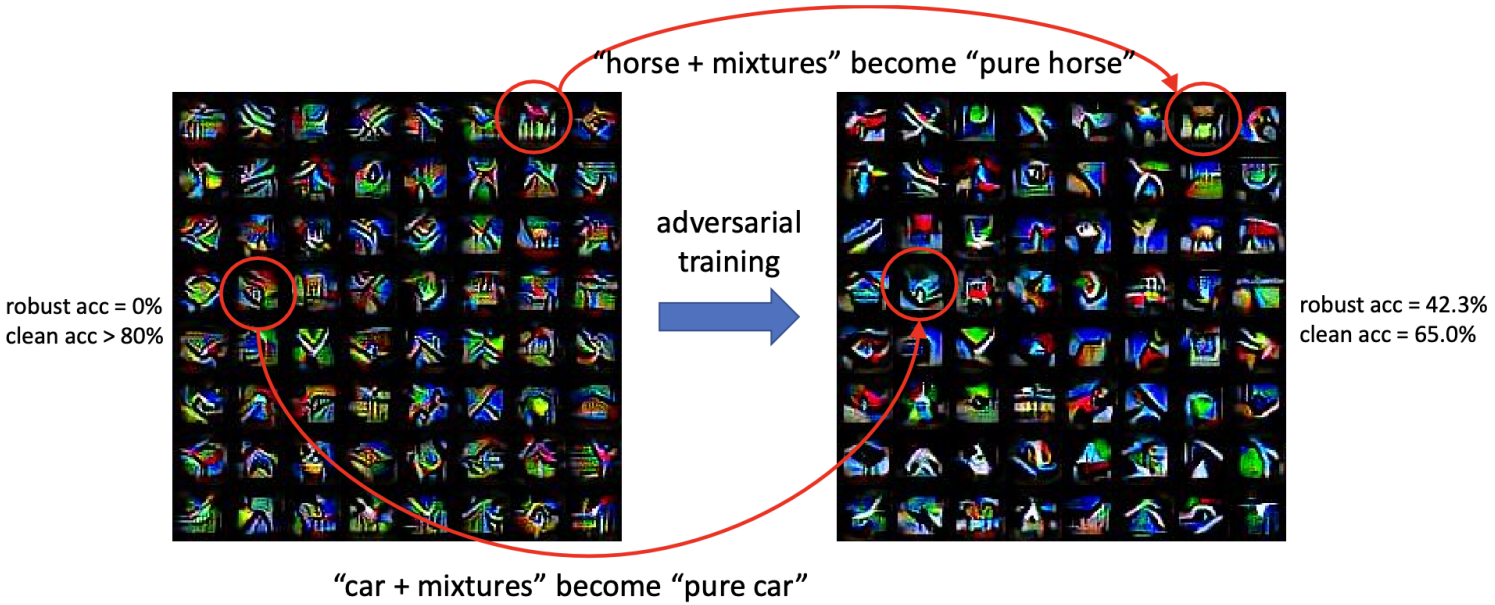


Figure: Adversarial training purifies dense mixtures. Visualization of some deep layer features of ResNet on CIFAR-10 data.

Large perturbation generated with robust models

- Adversarial examples generated with standard trained models appear to humans as noisy variants of the input image.
- But adversarial perturbations for robust models tend to produce salient characteristics of another class.

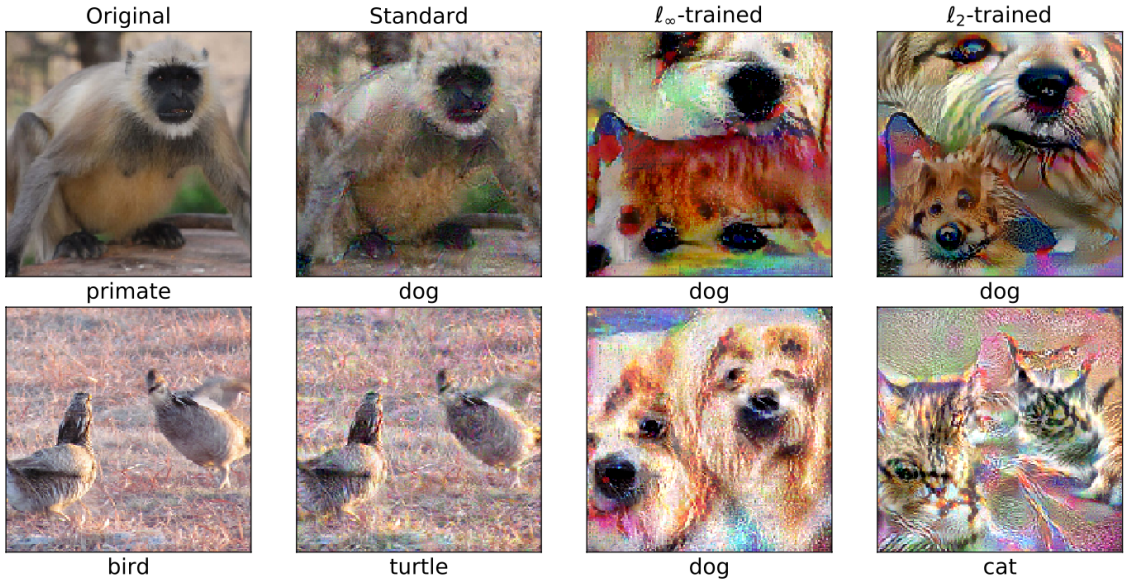


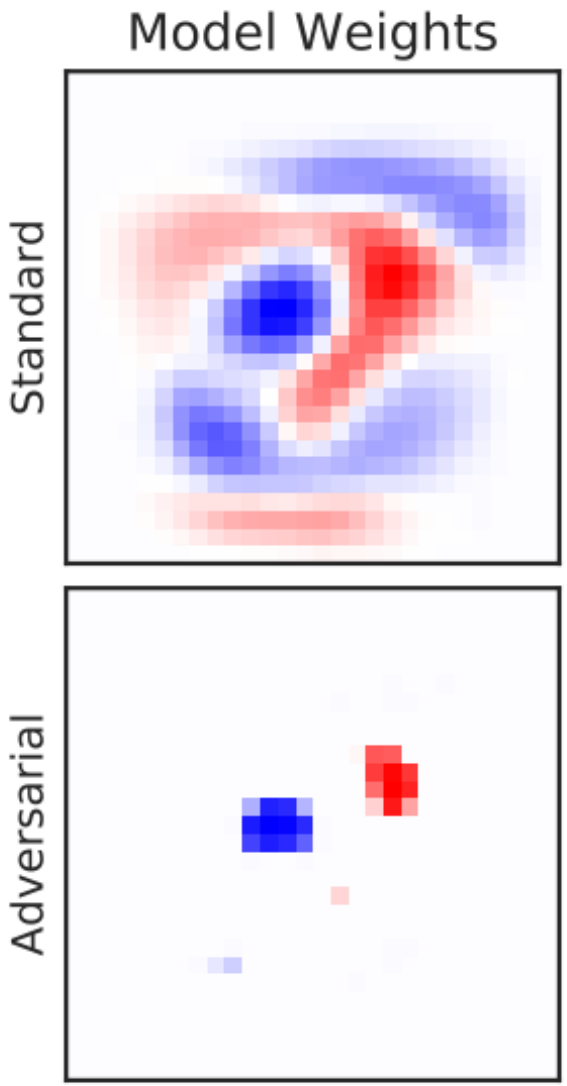
Figure: Visualization of large-ϵ adversarial examples.[7]

[7] Tsipras et al. "Robustness may be at odds with accuracy".

Discussion on adversarial training

- Adversarial training removes unnecessary features for robustness and focuses only on the important part.
- But the ignored part might be good for generalization (better clean accuracy).
- The large perturbation generated by a robust model tends to produce salient characteristics of the misclassified class.
- Due to these phenomena, robust models provide an additional benefit, which makes the saliency map easier to understand for human eyes.

Discussion on adversarial training



Additional Defenses During Training

- TRADES Theoretically principled trade-off between robustness and accuracy.
- MART Improving adversarial robustness requires revisiting misclassified examples.

Self-Supervised Adversarial Training - MIMIR

- According to the idea of Masked Image Modeling (MIM)[8], a network can be trained by reconstructing the randomly masked image patches.
- **Motivation:** The training is more efficient when masking out a part of image patches.
- **Pre-training:** The training target is to decrease the difference (loss) between clean inputs and the reconstructing results.
- **Fine-tuning:** The trained encoder can be fine-tuned for downstream tasks such as classification and object detection.

[8] Xie et al. "SimMIM: A simple framework for masked image modeling".

Self-Supervised Adversarial Training - MIMIR

- MIMIR: **M**asked **I**mage **M**odeling for Mutual **I**nformation-based Adversarial **R**obustness
- The model can be more robust to adversarial attacks simply by using adversarial images as input and clean images as the reconstructing target.
- The encoder is the model to be trained for downstream tasks.

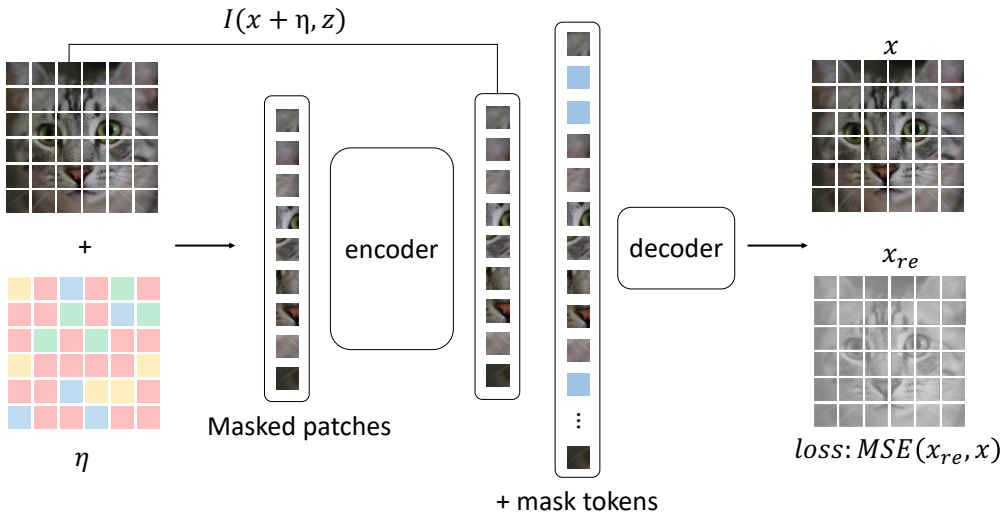


Figure: First, generate adversarial perturbation η and apply it to the input image. Then use the decoder output, i.e., x_{re}, and the clean input image, i.e., x, to calculate the loss. After pre-training, the encoder plus a classification layer is fine-tuned for classification tasks.

Design Details - MIMIR

- **Autoencoder:** MIMIR consists of an encoder f_e and a decoder f_d.
  - The input x is separated into non-overlapping image patches.
  - Then, randomly mask out 75% of the patches and use the remaining patches as inputs for the following process.
  - The encoder extracts discriminative features z from the masked inputs x.
  - The decoder reconstructs original inputs according to the discriminative features.

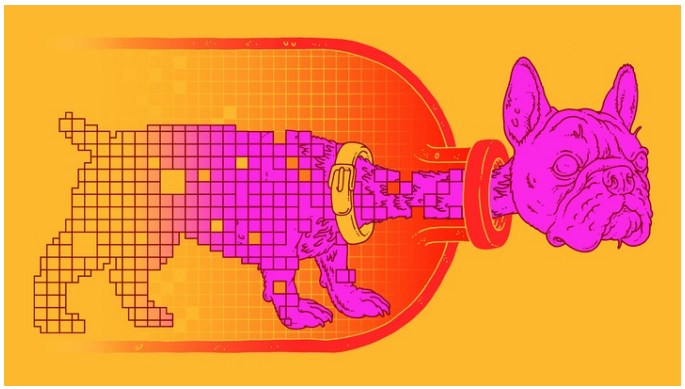
Design Details - MIMIR

- **Adversarial pre-training target:** The decoder reconstructs the original clean inputs by using the latent features z extracted from adversarial examples.
  - The training target is to decrease the difference between decoder output (x_{re}) and x; note the input is $x+\eta$.
  - Use mean-square error (MSE) to quantify the difference.
- **MI as penalty:** MI between z and $x+\eta$ decreases while adversarial pre-training.
  - Therefore the target is to increase the similarity (x and x_{re}) and decrease the MI ($I(x+\eta, z)$).

$$loss_{\text{mi}} = \mathcal{L}_{\text{mse}}(x, x_{re}) + \lambda I(x+\delta, z), \tag{1}$$

Why Does Reconstructing Clean Images from Adversarial Images Work?

- Considering that there is a bottleneck between the encoder and decoder.
- As the information flows through the bottleneck, adversarial information from η is eliminated.
- However, the natural information from x is maintained because of the constraint when reconstructing the target x.



Theoretical Justification - MIMIR

- Given an arbitrary classifier F, build a connect between mutual information ($I(x+\eta, z)$) and the probability that x_{re} (reconstruction result) is clean (p_e).
- Lower bound,
  $H(F(x+\eta)) - H_b(p_e) - p_e \log(|F(x+\eta)| - 1) \leq I(x+\eta, z)$,
- Upper bound, $I(x+\eta, z) \lesssim H(F(x_{re})) - 2p_e$.[9]

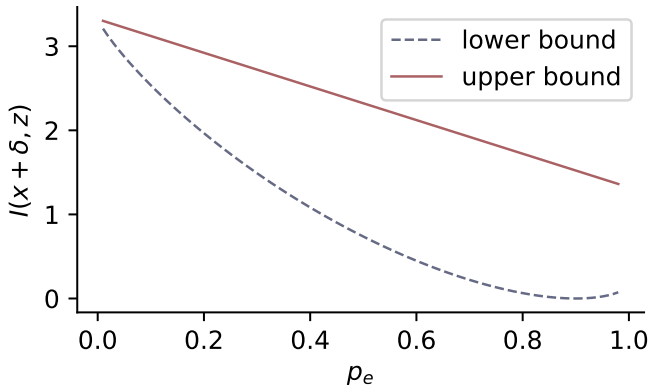


Figure: The example plots for the lower and upper bounds on the MI. The entropy ($H(\cdot)$) is chosen uniformly at random from a set of 10 classes. The lower bound reaches its minimum at $p_e = 0.9$.

[9] Let $H(\cdot)$ denote the information entropy and $H_b(\cdot)$ be the binary entropy.

Theoretical Justification - MIMIR

- Specifically, define p_e as the probability that the predicted label of $x+\eta$ by F is not equal to that of x_{re}, i.e., $p_e = \mathbb{P}(F(x+\eta) \neq F(x_{re}))$.
- The autoencoder is trained to only recover clean sample x without any interference from η.
- A relatively large value of p_e is expected, i.e., minimizing $I(x+\eta, z)$.

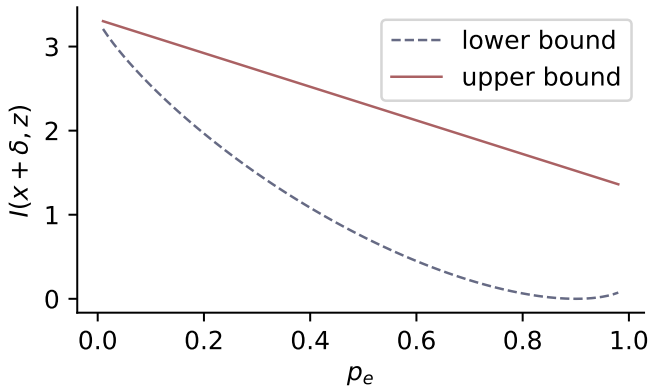


Figure: The example plots for the lower and upper bounds on the MI. The entropy ($H(\cdot)$) is chosen uniformly at random from a set of 10 classes. The lower bound reaches its minimum at $p_e = 0.9$.

Experimental Results - MIMIR

- MIMIR improves (natural and adversarial) accuracy on average by 4.19% on CIFAR-10 and 4.64% on ImageNet-1K, compared to baselines.

TABLE 2. TOP 1 NATURAL AND ADVERSARIAL ACCURACY ON CIFAR-10 TEST SET. THE RESULTS OF ARD+PRM [57] ARE TAKEN FROM THE ORIGINAL PAPER. AS THERE IS NO VIT-T ARCHITECTURE, WE USE THE SAME ARCHITECTURE AS DEIT-T.

Dataset	Models	Pre-train	Fine-tune	Natural	PGD_{20}	PGD_{100}	AA
CIFAR-10 [46] ($l_\infty, \epsilon = 8/255$)	ViT-T (DeiT-T)	ImageNet-1K	PGD	75.46	48.10	47.96	43.62
		MAE [36]	PGD	79.89	48.64	48.43	44.27
		ImageNet-1K	ARD+PRM [57]	79.60	50.33	50.15	45.99
		MIMIR	PGD	**84.82**	**57.51**	**57.38**	**52.96**
	ViT-S	ImageNet-1K	PGD	79.59	50.86	50.73	46.37
		MAE [36]	PGD	85.97	54.19	53.94	50.49
		ImageNet-1K	ARD+PRM [57]	81.86	51.73	51.46	47.33
		MIMIR	PGD	**88.11**	**56.63**	**56.34**	**53.18**
	ViT-B	ImageNet-1K	PGD	83.16	52.98	52.71	49.06
		MAE [36]	PGD	88.86	57.03	56.86	52.42
		ImageNet-1K	ARD+PRM [57]	84.90	53.80	53.51	50.03
		MIMIR	PGD	**89.30**	**58.07**	**57.87**	**54.55**
	ConViT-T	ImageNet-1K	PGD	53.09	33.63	33.61	29.65
		MAE [36]	PGD	75.49	45.29	45.17	41.12
		ImageNet-1K	ARD+PRM [57]	80.28	**49.86**	**49.55**	**45.42**
		MIMIR	PGD	**80.74**	49.37	49.16	45.04
	ConViT-S	ImageNet-1K	PGD	54.03	34.61	34.60	30.60
		MAE [36]	PGD	86.43	54.76	54.59	51.13
		ImageNet-1K	ARD+PRM [57]	84.32	53.10	52.81	48.85
		MIMIR	PGD	**87.49**	**56.35**	**56.20**	**52.54**
	ConViT-B	ImageNet-1K	PGD	61.54	38.77	38.71	34.21
		MAE [36]	PGD	89.03	58.19	57.75	54.25
		ImageNet-1K	ARD+PRM [57]	85.80	53.36	53.05	49.33
		MIMIR	PGD	**89.30**	**59.31**	**59.01**	**55.64**

Experimental Results - MIMIR

TABLE 3. TOP 1 NATURAL AND ADVERSARIAL ACCURACY ON IMAGENET-1K.

Dataset	Models	Pre-train	Fine-tune	$\epsilon = 2/255$			$\epsilon = 4/255$		
				Natural	PGD_{20}	PGD_{100}	Natural	PGD_{20}	PGD_{100}
ImageNet-1K [26] (l_∞)	ViT-S	-	FastAT [86]	71.37	49.49	49.41	66.92	34.07	33.82
		-	AGAT [87]	70.62	49.00	48.85	66.10	33.62	33.40
		MIMIR	FastAT [86]	**74.60**	**54.56**	**54.55**	**71.29**	**40.98**	**40.63**
	ViT-B	-	FastAT [86]	70.31	50.55	50.06	65.18	33.59	33.39
		-	AGAT [87]	70.41	51.23	51.11	67.93	34.94	34.78
		MIMIR	FastAT [86]	**75.88**	**55.42**	**55.36**	**73.22**	**41.26**	**40.46**
	CaiT-XXS24	-	FastAT [86]	72.84	**54.31**	**54.26**	68.77	32.14	32.09
		-	AGAT [87]	71.15	54.17	54.08	68.22	31.46	31.39
		MIMIR	FastAT [86]	**73.39**	53.39	53.37	**69.90**	**40.53**	**40.22**
	CaiT-S36	-	FastAT [86]	72.51	53.12	52.76	71.20	33.02	32.84
		-	AGAT [87]	72.69	53.66	53.48	71.06	33.46	33.17
		MIMIR	FastAT [86]	**76.05**	**56.78**	**56.75**	**73.57**	**40.03**	**39.16**

Evaluation under Adaptive Attacks - MIMIR

- As the attacker is aware of the defense MIMIR, design two adaptive attacks.
- **PGD-MI:** As MI is used in the loss while pre-training, generate adversarial images by increasing the $I(x+\eta, z)$.
- **PGD-fea:** Introduce PGD-feature that directly attacks the feature extracted by the encoder.

TABLE 9. THE EXPERIMENTAL RESULTS FOR ADAPTIVE ATTACKS.

Dataset	Model	PGD_{20}	$PGD\text{-}MI_{100}$	$PGD\text{-}fea_{100}$
CIFAR-10	ConViT-S	56.35	56.16	78.52
	ViT-S	56.63	56.31	78.41
	ViT-B	58.14	57.85	80.49
Tiny-ImageNet	ConViT-S	26.39	26.29	58.50
	ViT-S	26.37	26.18	57.36
	ViT-B	25.41	25.05	58.90
ImageNet-1K	ConViT-S	53.86	53.84	72.10
	ViT-S	54.56	54.55	72.27
	ViT-B	55.41	55.36	73.51

Loss surface - MIMIR

- To prevent obfuscated gradients, plot the loss surfaces of MIMIR-trained models.

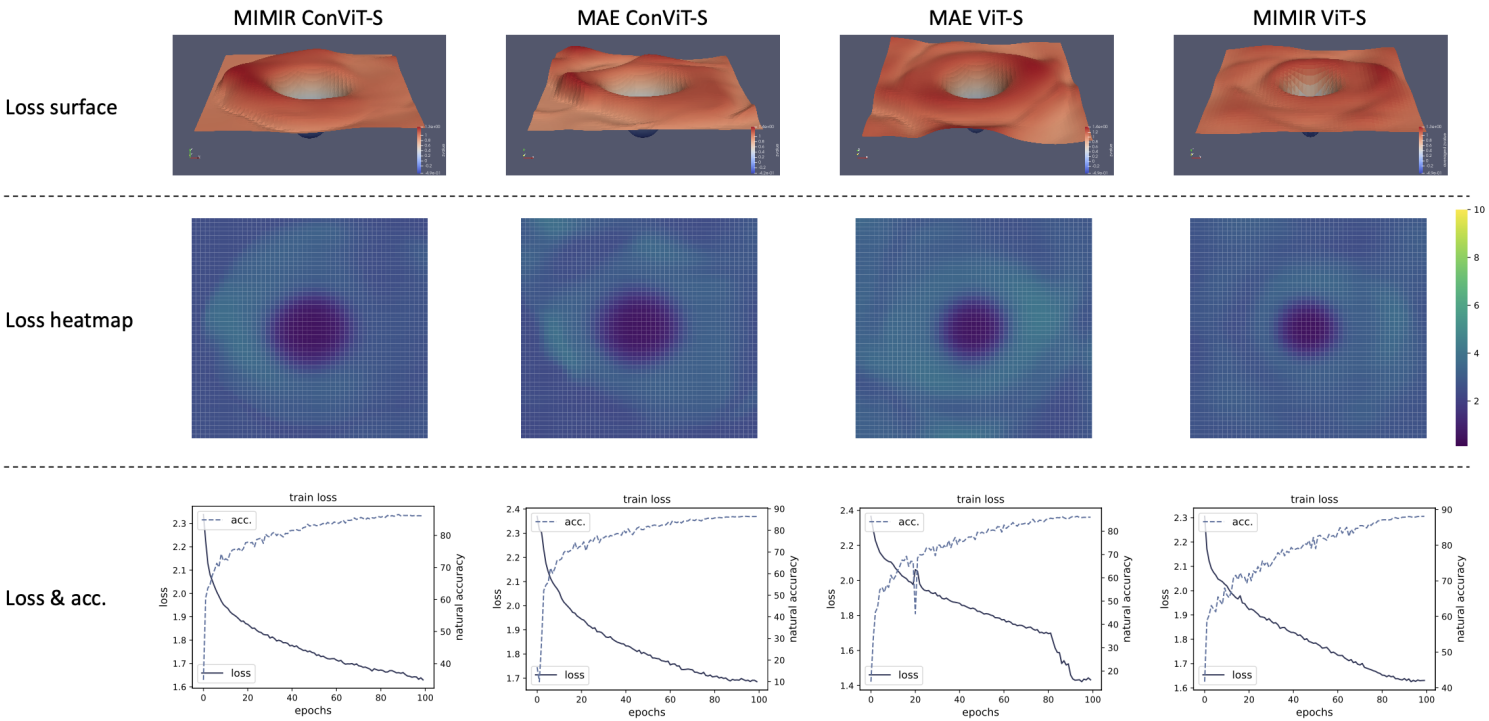


Figure: The loss surfaces of the models. The four figures of each column are the loss information of one model.

Loss surface - MIMIR

- An example (a and b in the figure below) of loss surface with obfuscated gradients[10].

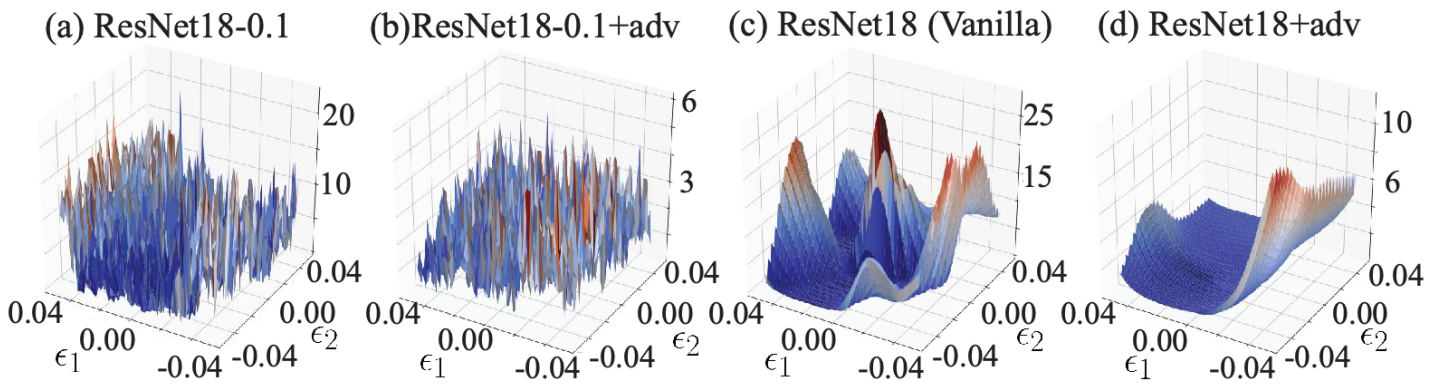


Figure 5: Gradient-based attack's loss landscapes in k-WTA **(a, b)** and conventional ReLU models **(c, d)**. (a,b) k-WTA Models have much more non-convex and non-smooth landscapes. Also, the model optimized by adversarial training (b) has a lower absolute value of loss.

[10] Chang Xiao, Peilin Zhong, Changxi Zheng. "Enhancing Adversarial Defense by k-Winners-Take-All." ICLR 2020

Time Consumption and Memory Usage - MIMIR

- The table below shows the time consumption and memory usage of MIMIR and traditional adversarial training.
- MIMIR shows much less time consumption.

TABLE 10. THE AVERAGE TIME CONSUMPTION OF A SINGLE EPOCH ON 4 GPUS. THE "MEM." REFERS TO GPU MEMORY USAGE.

Models	#Parames (M)	Method	**CIFAR-10** [46]		**Tiny-ImageNet** [47]		**ImageNet-1K** [26]	
			time[S]	mem.[GB]	time[S]	mem.[GB]	time[S]	mem.[GB]
ViT-S	21.34	PGD_{10} AT	149.3	2.54×4	306.0	3.99×4	2251.7	12.5×4
		FastAT	43.3	2.54×4	67.7	4.03×4	555.5	10.4×4
		MAE	16.1	3.24×4	33.0	3.27×4	269.6	11.1×4
		MIMIR	18.4	3.12×4	40.0	3.18×4	275.5	11.1×4
ViT-B	85.27	PGD_{10} AT	361.2	5.39×4	1022.2	8.30×4	5416.7	22.1×4
		FastAT	122.8	5.36×4	180.2	8.34×4	1361.3	19.8×4
		MAE	53.0	5.95×4	106.5	5.95×4	490.9	17.0×4
		MIMIR	59.0	6.08×4	122.0	6.11×4	509.9	17.0×4
ConViT-S	27.05	PGD_{10} AT	442.5	6.64×4	897.0	12.19×4	6626.5	32.5×4
		FastAT	106.5	5.86×4	183.2	10.62×4	1431.2	26.4×4
		MAE	33.0	10.6×4	67.5	10.61×4	609.7	27.5×4
		MIMIR	45.0	10.4×4	90.0	10.54×4	611.1	28.3×4

Latest Results in this Direction

- As self-supervised learning requires no labels, it is easier to achieve large-scale adversarial training.
- AdvXL[11] increases the model parameters from the previously largest 200M size to 1B.
- They use the web-scale dataset comprising more than 1B images for adversarial pre-training.

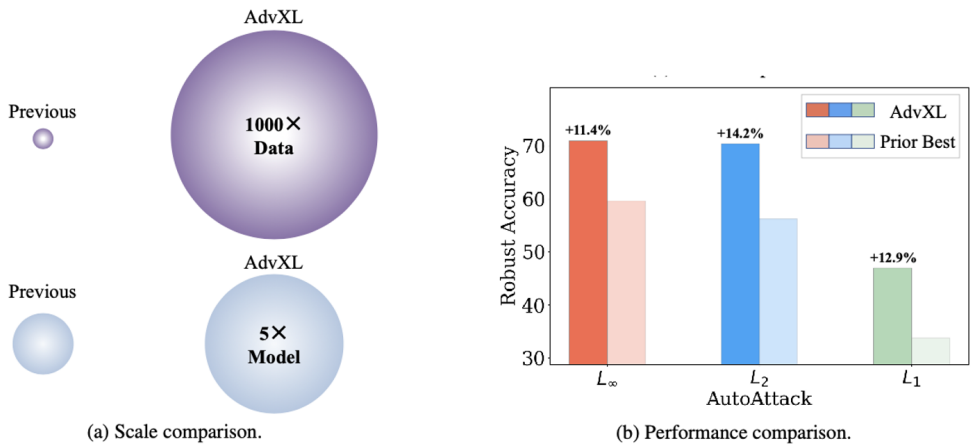
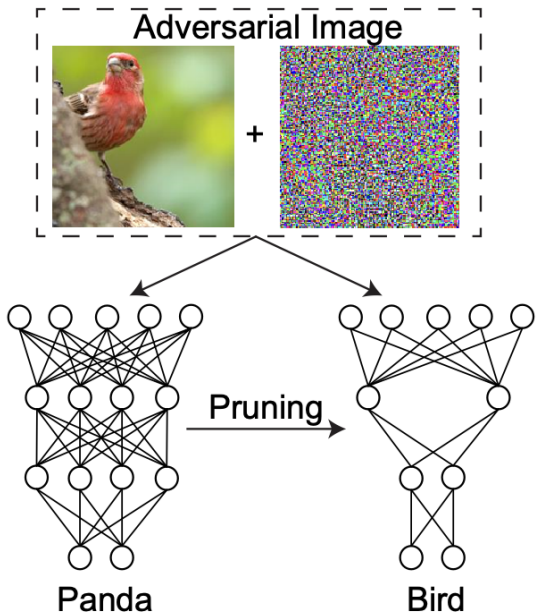


Figure: AdvXL increases significantly in terms of both model size and data scale, which brings a substantial boost over prior best results on ImageNet-1K.

[11] Wang et al. "Revisiting Adversarial Training at Scale".

Pruning

- Pruning is the practice of removing redundancy parameters from an existing network to increase its efficiency while maintaining accuracy.
- According to adversarial training, adversarial attacks are successful because of some latent “mixture”.
- A natural question is whether we can prune the “mixture”?



How to prune the network for robustness

- Different from generalized pruning, adversarial pruning needs to be implemented for the part related to adversarial perturbation.

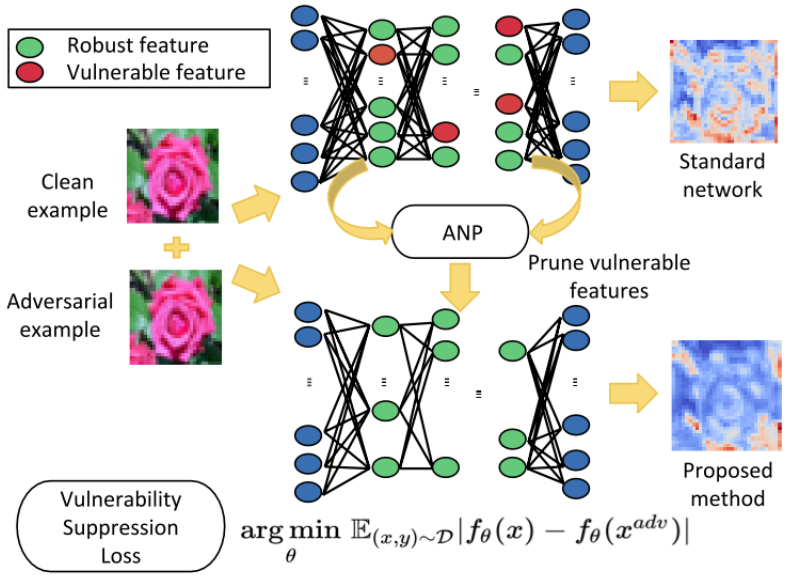


Figure: Pruning according to distortion between latent features from clean and adversarial examples.[12]

[12] Madaan et al. "Adversarial neural pruning with latent vulnerability suppression".

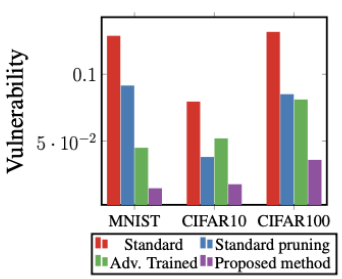
How to prune the network for robustness

- Pruning neurons that generate the most distorted features.
- The distortion of latent features (vulnerability) can be quantified by:

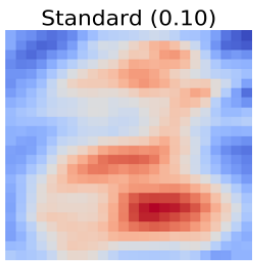
$$v(z, \tilde{z}) = \mathbb{E}_{(x,y)\sim D}\|z - \tilde{z}\|,$$

where z and $\tilde{z}$ are the output of hidden layers when input clean and adversarial examples.

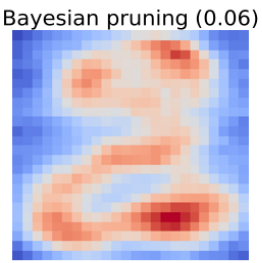
- Empirically, the most distorted features usually appear in similar positions so they can be removed.



(a) Mean distortion

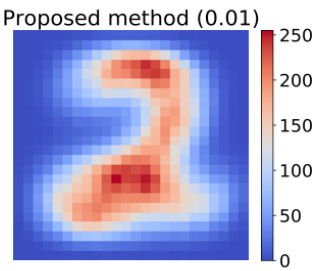


(b) Standard



(c) Bayesian Pruning

(d) Adv. Training



(e) ANP-VS (Ours)

How to prune the network for robustness

- This pruning method actually does something similar to adversarial training.

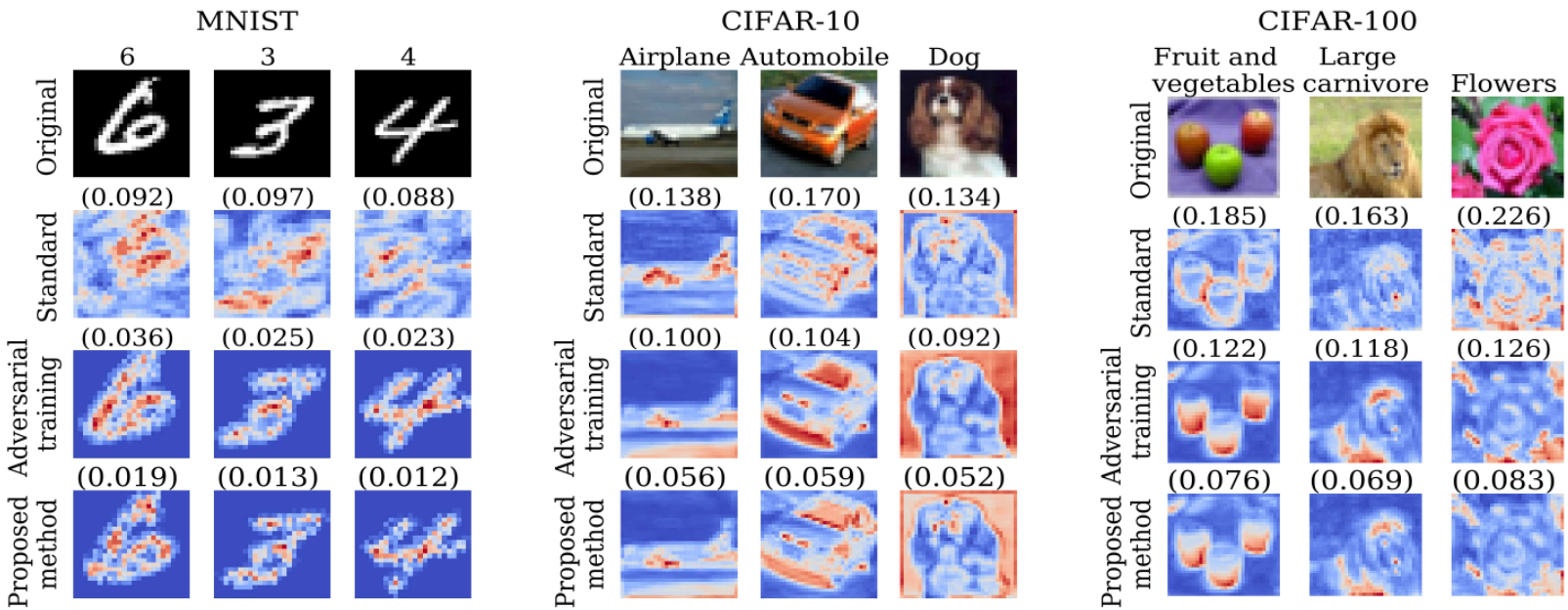


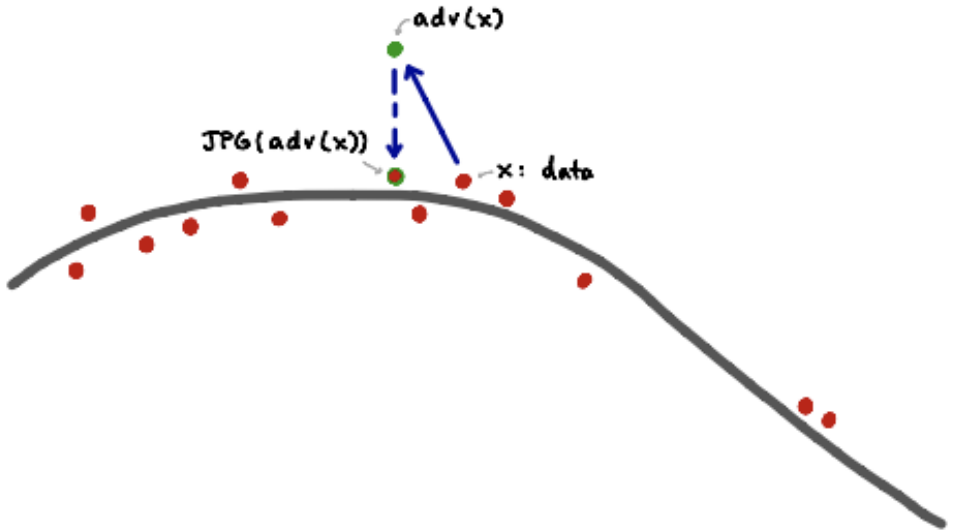
Figure: Visualization of the vulnerability of the latent features with respect to the input pixels for various sets of datasets.

General Idea

- **Defense after deployment**.
- The defender only has control of the input samples.
- The goal is to reject or correct malicious (e.g., adversarial or backdoor) inputs.

Random input transformation

- As the adversarial perturbation looks like noise, an idea is whether we can recover adversarial examples into natural images.
- A simple method is applying JPG compression or Gaussian noise to the input.
- However, JPG compression is far from an effective defense when the adversarial perturbation is larger.



Random input transformation

- Adding noise seems to help with recovering from the adversarial examples.
- It is not effective in decreasing the test error on adversarial examples.
- Its error on adversarial examples could match that of the error on clean data when adding more noise.

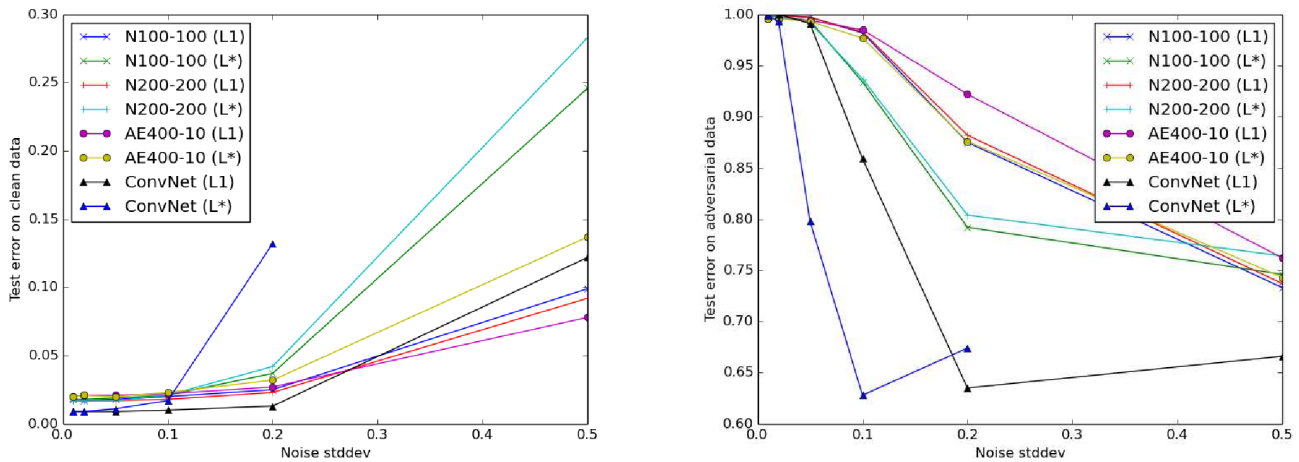


Figure: Noise Injection (Gaussian additive noise): Test error on clean data (Left) and on adversarial data (Right) vs. standard deviation of Gaussian additive noise[13].

[13] Gu et al. "Towards Deep Neural Network Architectures Robust to Adversarial

Random input transformation

- Input transformation idea can be further improved by combining it with an autoencoder.
- Autoencoders leverage hidden representation to introduce regularization to uncover useful properties of the data.
- An adversarial example can be detected with an autoencoder first, and if it is adversarial, reform it with another autoencoder.

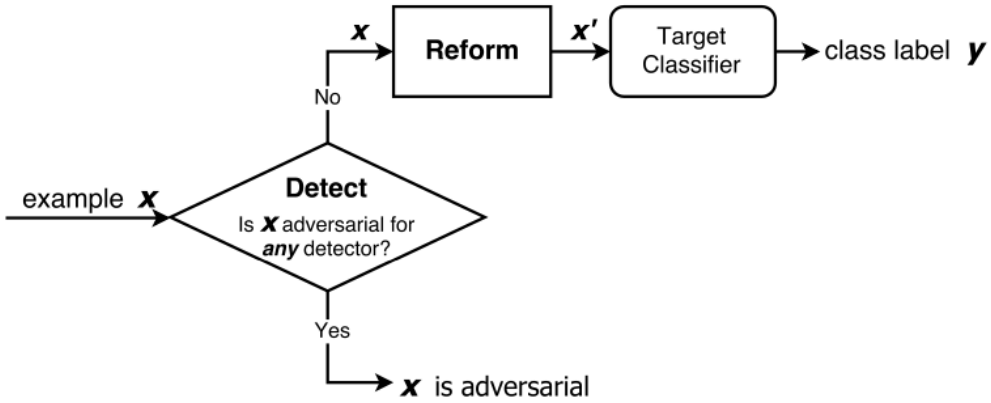
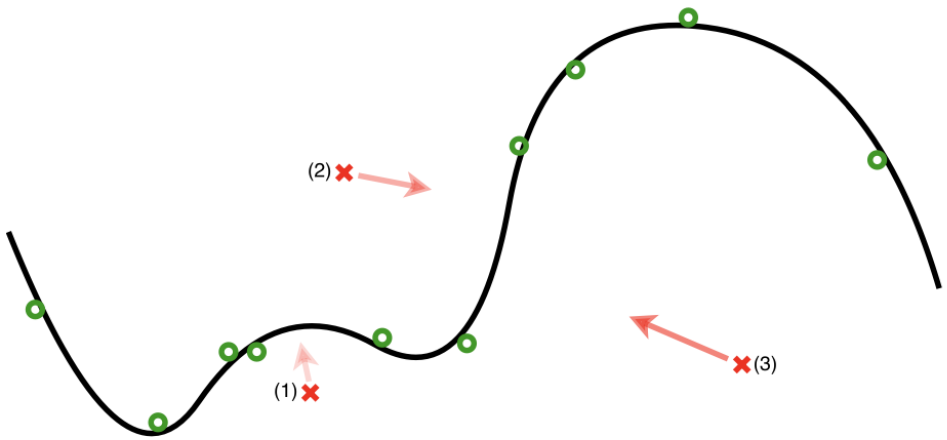


Figure: MagNet workflow.[14]

[14] Dongyu Meng et al. "Magnet: a two-pronged defense against adversarial examples."

Random input transformation

- These autoencoders can be trained with only clean examples.
- The training objective is to estimate the distance between the test example and the boundary of the manifold of normal examples.
- Clean example should satisfy the same data distribution as the clean training set.



Diffusion Models for Adversarial Purification

- The diffusion models are also used to recover the clean image through a reverse generative process.

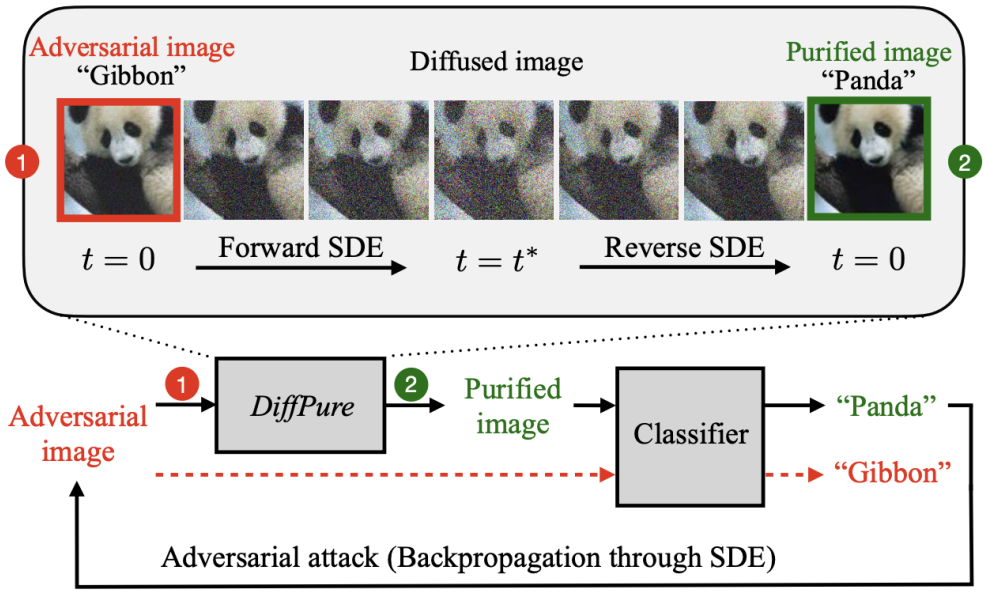


Figure: An illustration of DiffPure[15]. Given a pre-trained diffusion model, they add noise to adversarial images following the forward diffusion process with a small diffusion timestep $t*$ to get diffused images, from which they recover clean images through the reverse denoising process before classification.

[15] Nie et al. "Diffusion Models for Adversarial Purification."

Additional Defenses After Deployment

- Defending against individual adversarial inputs:
  - Enhancing Adversarial robustness via Test-time Transformation Ensembling shows that test-time transformation Ensembling could work as an effective defense against evasion attacks.
- Rejecting the input:
  - SafetyNet: Detecting and Rejecting Adversarial Examples Robustly shows we can defend against attacks like DeepFool by quantizing activations in late-stage ReLUs to conceal the gradient.

Certified Robustness

- So far, we discussed empirical defenses.
- Certified robustness provides *theoretical guarantees*.

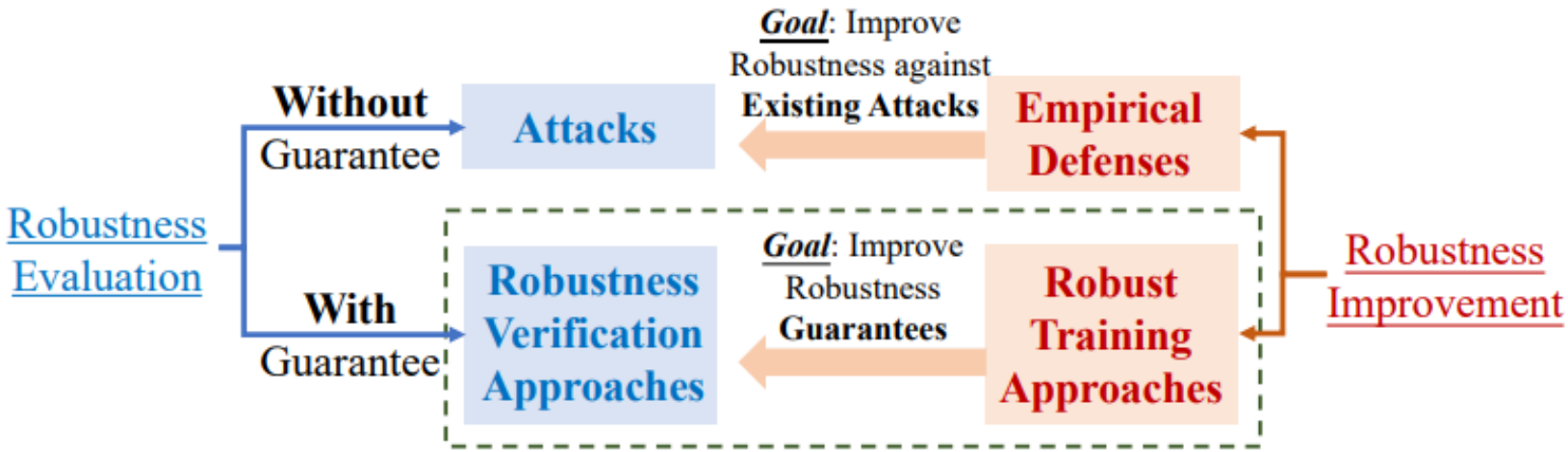


Figure: Empirical vs. certified robustness.[16]

[16] Li Linyi et al. "Sok: Certified robustness for deep neural networks." arXiv preprint arXiv:2009.04131 (2020).

Certified Robustness

Certified robustness consists of

- Robustness verification
- Robust training

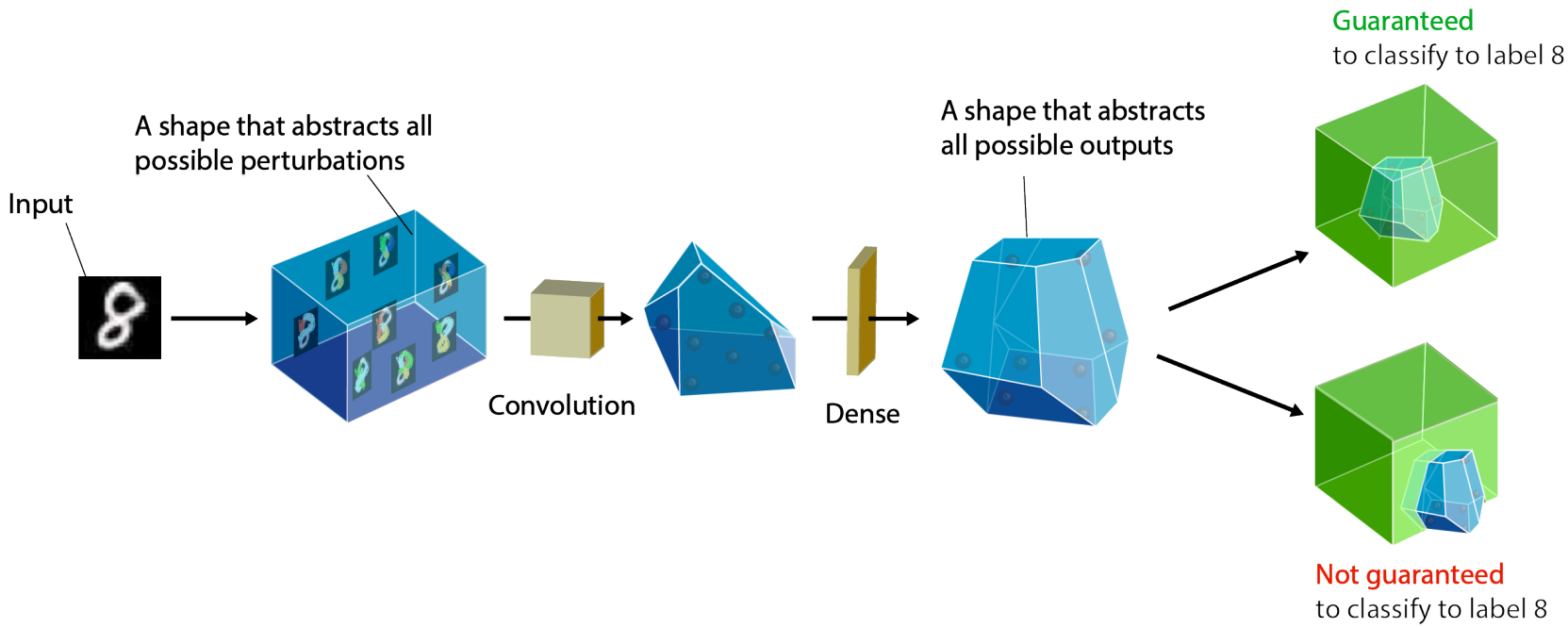


Figure: Certified Robustness Overview.[17]

[17] https://github.com/eth-sri/eran

Robustness Verification

- Providing a lower bound on the robustness of the model.
- Wrt. to the constraints of the perturbation, e.g., ℓ_2 norm.
- Complete vs. incomplete verification
  - If there is an adversarial example x' for any non-verified input x, then it is complete verification.
- Probabilistic vs. deterministic verification
  - If an input x is non-robust against adversarial examples, deterministic verification will definitely label it as non-robust.

Robustness Verification

- The optimization problem:

$$\mathcal{M}(l, l') = \min_{x+\eta} Z_\theta(x+\eta)_l - Z_\theta(x+\eta)_{l'}$$

$$\text{s.t.} \quad f(x) = l := \underset{l_i}{\mathit{argmax}}\, Z_\theta(x)_{l_i} \text{ and } \|\eta\|_p \leq \epsilon.$$

where $Z_\theta(x)_l$ is the softmax output for input x on class l and η is the perturbation.

Certified Robustness

If $\mathcal{M}(l, l') > 0$ for all $l' \neq l$, then the model is *certifiably robust* at input x within radius ϵ wrt. ℓ_p norm.

Robustness Verification

- Robustness is shown by $\mathcal{M}(l, l') > 0$.
- Scalability issue: Complete verification is NP-Complete![18]
- Alternative way: Computing lower bound on M by relaxing the conditions.
  - Tightness issue: this leads to a loose bound.
- There is a trade-off between scalability and tightness.

[18] Guy Katz et al. "Reluplex: An efficient SMT solver for verifying deep neural networks." Computer Aided Verification, CAV 2017.

Robustness Verification Taxonomy

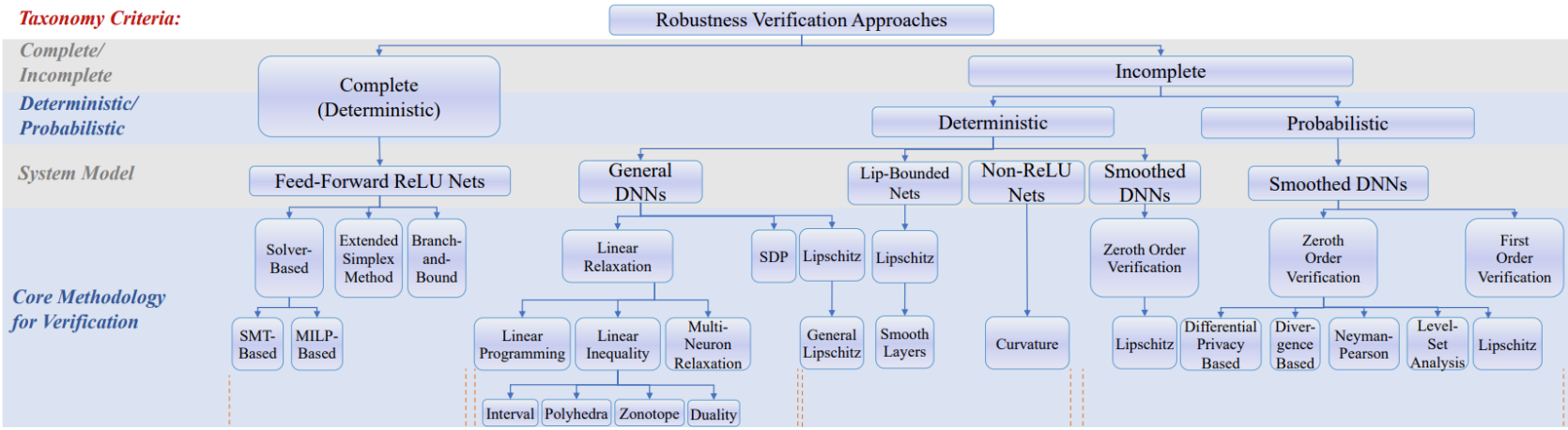


Figure: Robustness Verification Taxonomy.[19]

[19] Li Linyi et al. "Sok: Certified robustness for deep neural networks." arXiv preprint arXiv:2009.04131 (2020).

Complete Verification

- Worst-case scenario with exponential time complexity.
- Works reasonably well in NNs with a couple of thousands of neurons.
- Solver-based approach: ReLU operation can be encoded by a linear inequality:

$$z = ReLU(\hat{z}) \iff ((\hat{z} < 0) \wedge (z = 0)) \vee ((\hat{z} \geq 0) \wedge (z = \hat{z})).$$

- Branch-and-Bound approach: ReLu is locally linear around some region of x.
- Split neurons into branches: $z \geq 0$ (active) and $z < 0$ (inactive).

Incomplete Verification

- Complete verification is not feasible for large NNs.
- Incomplete verification utilizes relaxations of the activation function.
- Deterministic incomplete approaches.
  - Linear relaxations, abstractions.
  - Semidefinite programming.
  - Lipschitz bounds.
- Probabilistic incomplete approaches.

Linear Relaxations of ReLU

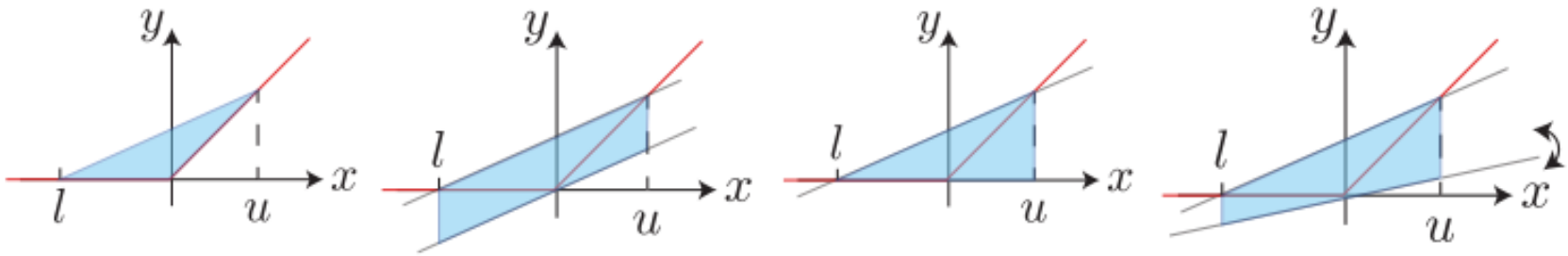


Figure: Different linear relaxations of ReLU. The upper and lower bounds (*u* and *l* values) are computed layer by layer.[20]

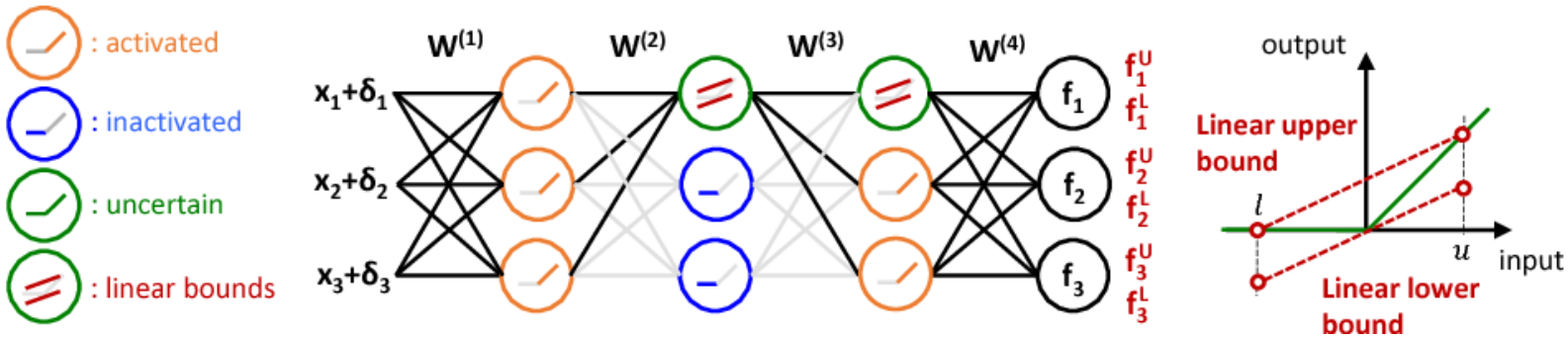


Figure: Lower and Upper bounds for ReLU Network. [21]

[20] Linyi Li et al. "Sok: Certified robustness for deep neural networks." arXiv preprint arXiv:2009.04131 (2020).

[21] Tsui-Wei Weng et al. "Towards fast computation of certified robustness for relu networks." International Conference on Machine Learning. PMLR, 2018.

Multi-Neuron Relaxations

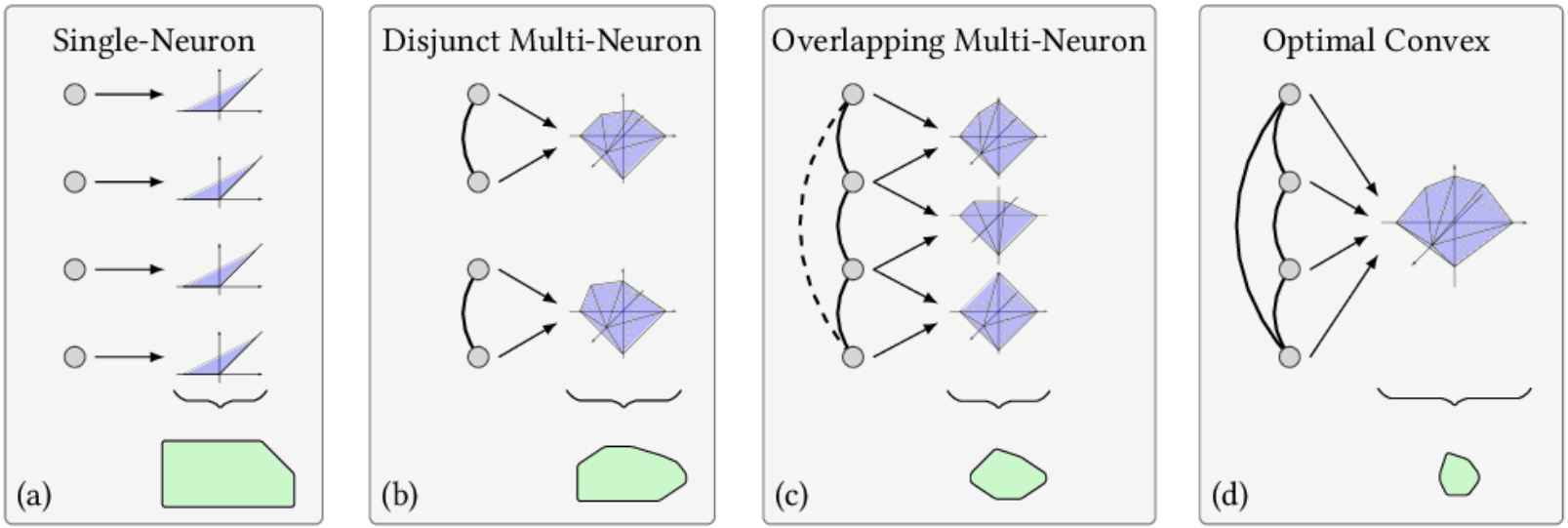


Figure: Single-neuron vs. multi-neuron abstraction. The resulting overall abstraction is illustrated by green areas.[22]

[22] Mark Niklas Müller et al. "PRIMA: general and precise neural network certification via scalable convex hull approximations." arXiv preprint arXiv:2103.03638 (2021).

Robust Training

- Robust training is a complement to robustness verification.
- It is used to enhance the certifiability of the model.
- Objective function aims to minimize
  - the loss function (for accuracy),
  - linear relaxations and/or bounds (for robustness).

- Defenses
  - Obfuscated Gradients
  - Adversarial training
  - Self-Supervised Adversarial Training
  - Pruning
  - Random input transformation
  - Certified Robustness

- Generative Adversarial Networks

Generative Adversarial Networks

- Goodfellow et al., Generative Adversarial Networks, NeurIPS, 2014.

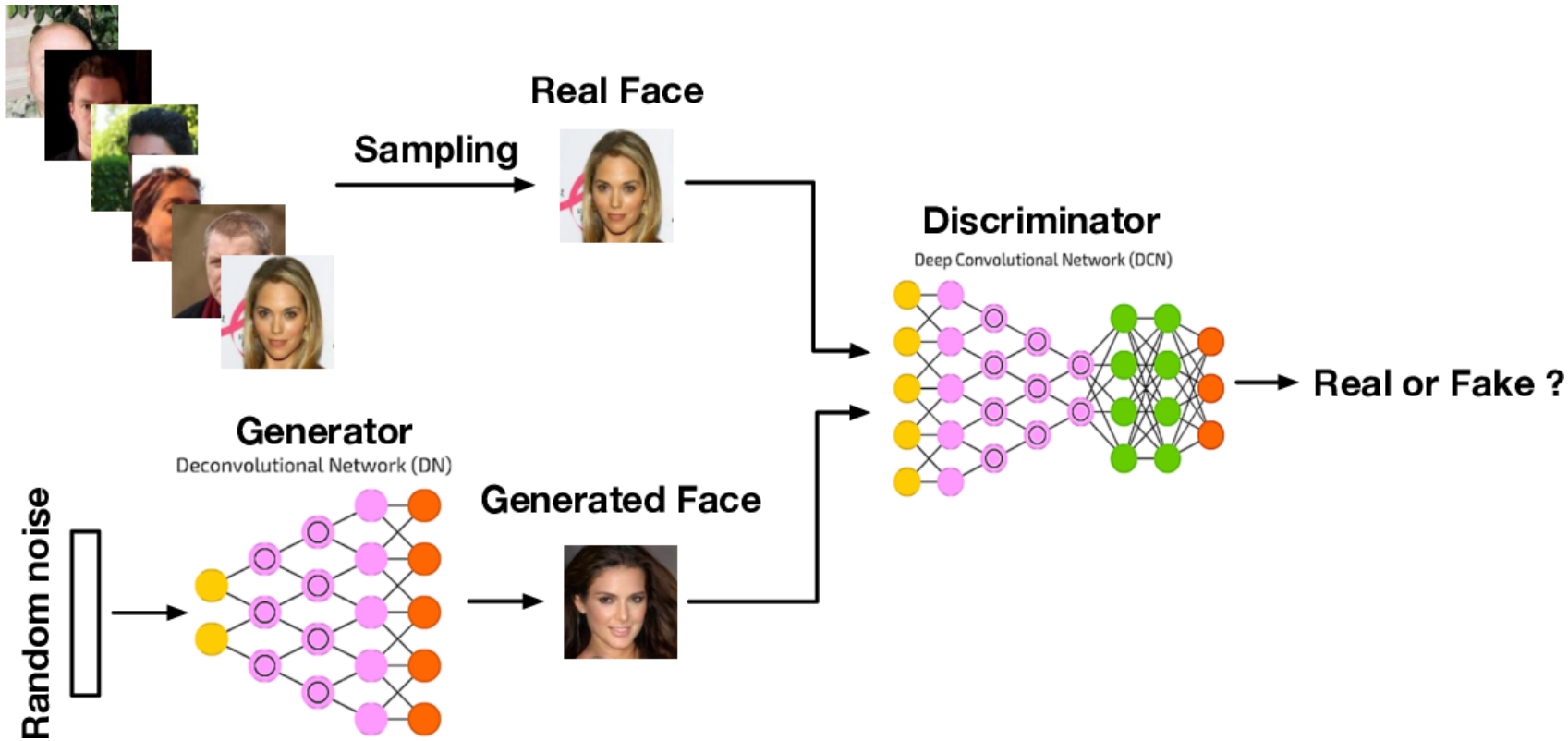


Figure: GAN.[23]

[23] Zhengwei Wang et al. Generative Adversarial Networks: A Survey and Taxonomy. ArXiv abs/1906.01529 (2019).

Generative Adversarial Networks

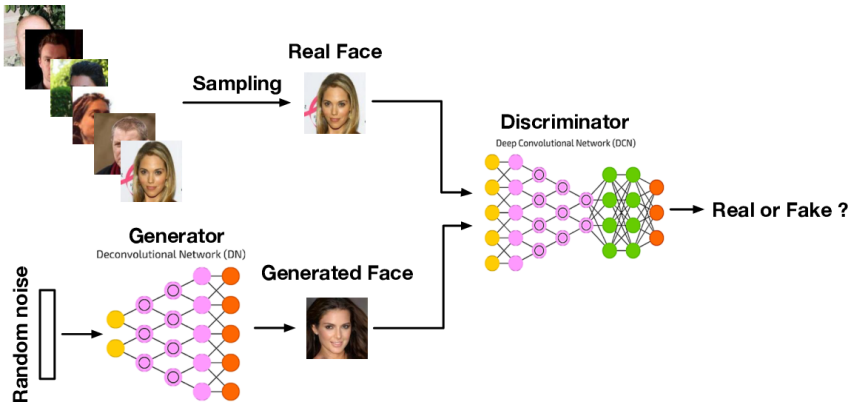


Figure: GAN.

- *Generator*: generating fake images from the noise.
- *Discriminator*: trying to distinguish real and fake images.
- The optimization problem:

$$min\ \mathcal{J}^{(G)}\left(\theta^{(G)}, \underset{\theta^{(D)}}{argmin}\,\mathcal{J}^{(D)}\left(\theta^{(G)}, \theta^{(D)}\right)\right).$$

Applications of GANs

- Generating synthetic data.
  - For privacy reasons or lack of data.
- Improving image quality.
  - Old images or astronomical images.
- Generating images for fashion and art.