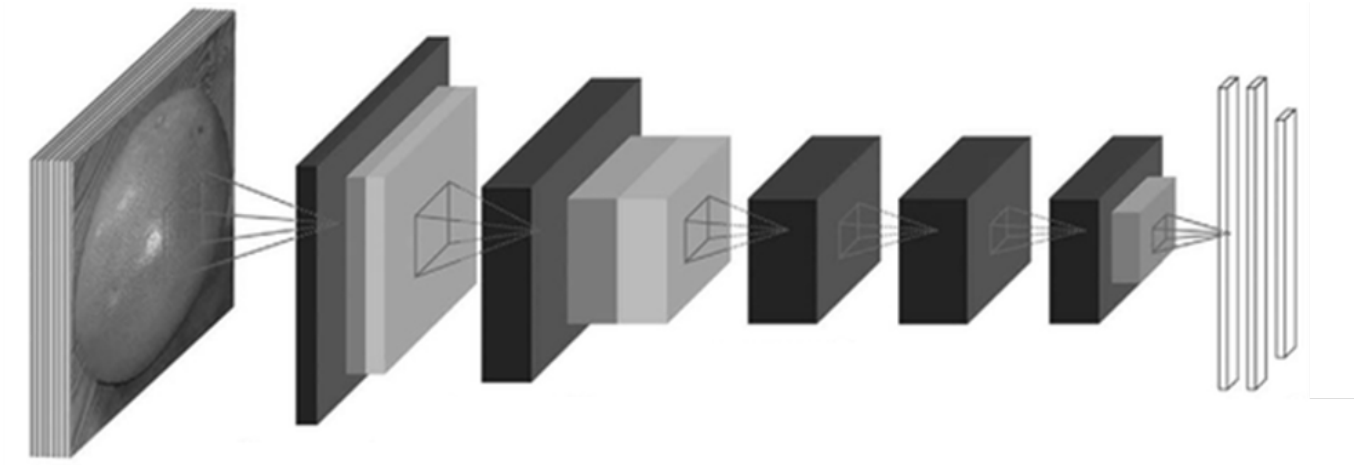
Adversarial Robustness of Deep Learning: Theory, Algorithms, and Applications

Wenjie Ruan[1], Xinping Yi[2], Xiaowei Huang[2]

[1]University of Exeter, UK; [2]University of Liverpool, UK

Outline

Introduction (Wenjie)



Deep learning models are pervasively used in **safety-critical systems**!

nature reviews
drug discovery

Review Article | Published: 11 April 2019

Applications of machine learning in drug discovery and development

Jessica Vamathevan ✉, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran,

nature International weekly journal of science

Home | News | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For Authors

Archive > Volume 542 > Issue 7639 > Letters > Article > Article metrics > News

Article metrics for:

Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun

- Drug Discovery and Development
- Automatic Medical Diagnosis

- Self-driving Cars
- Autonomous Vehicles

Can We Trust its Decisions?

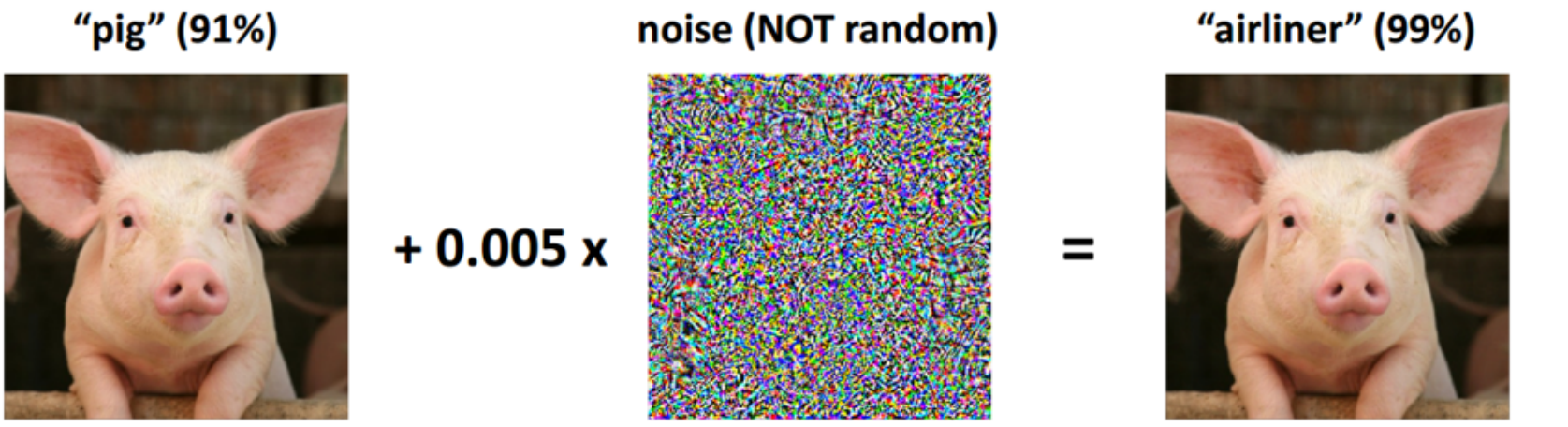
Researchers and Practitioners may have many concerns...

- How does a deep learning model make a decision?
- Does deep learning always make a correct decision?
- Under what circumstances a deep learning model will make a wrong decision?
- ... ...

Ultimate Question: **Can we really *trust* the decisions made by deep learning models, especially on safety-critical applications?**

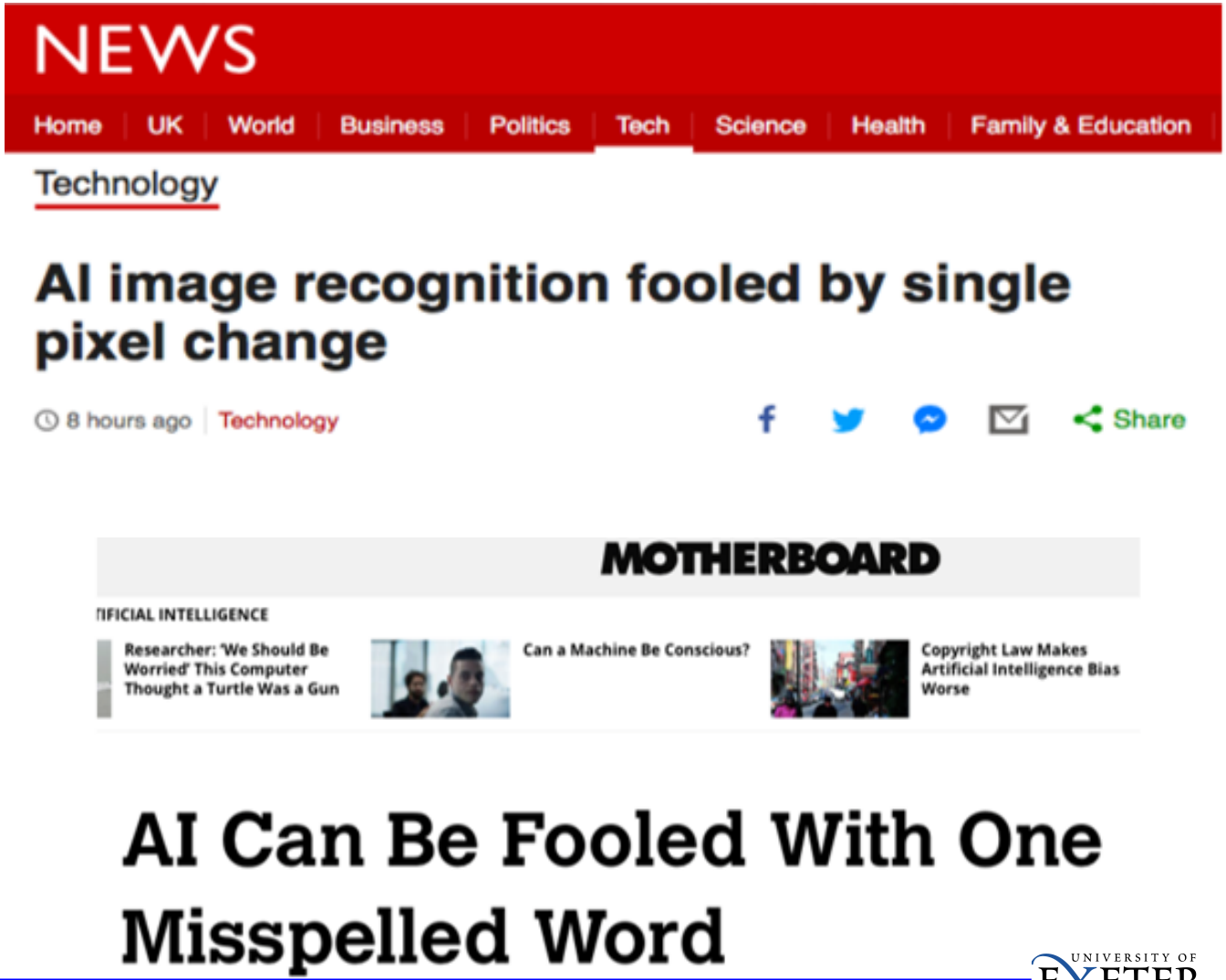
Vulnerability to Adversarial Examples

Yet we cannot **trust** deep learning models, at least **not now** ...



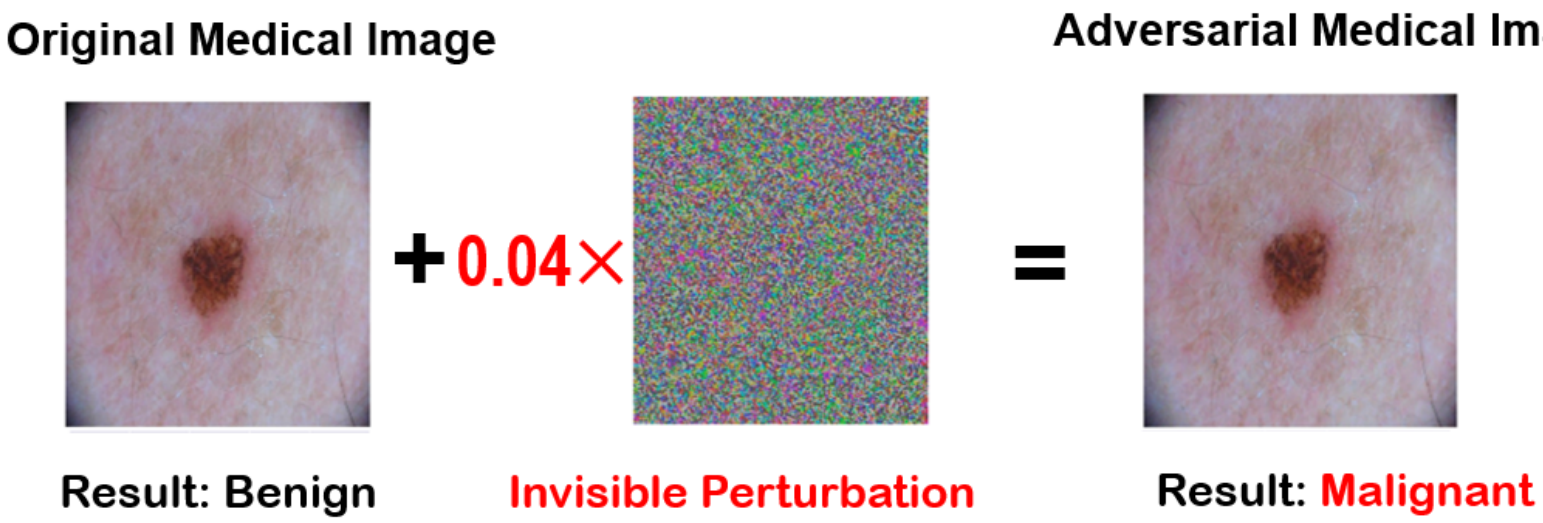
Simple approach to **fool** deep neural networks: Fast Gradient Sign Method (FGSM), [Goodfellow et al., 2014]

Such vulnerabilities are *pervasive* ...



Vulnerability to Adversarial Examples

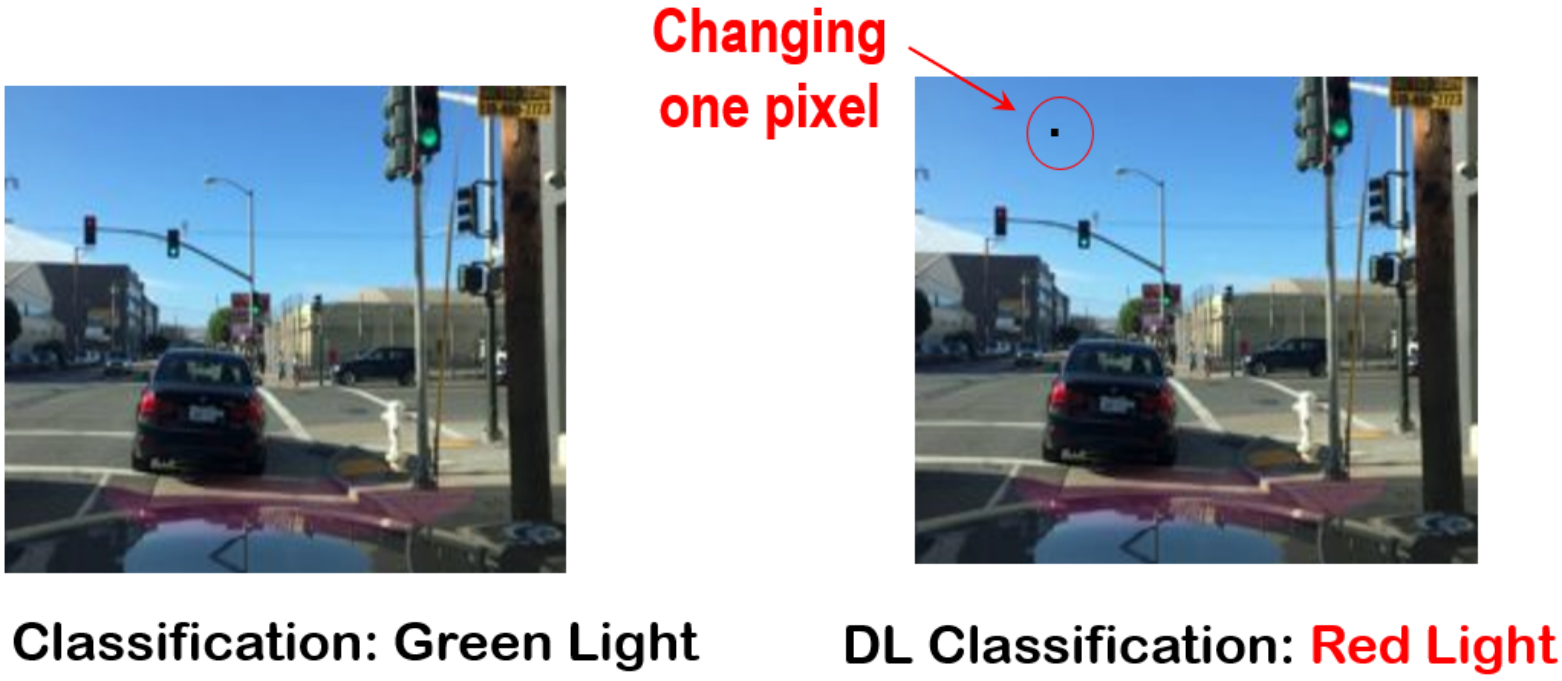
In Deep Medical Systems...



Adversarial Attacks Against Medical Deep Learning Systems. arXiv:1804.05296, 2018

Vulnerability to Adversarial Examples

In Autonomous Systems...



A Game-Based Approximate Verification of Deep Neural Networks with Provable Guarantees, Theoretical Computer Science, 2019

Vulnerability to Adversarial Examples

In Medical Record Analysis...

Original Medical Record

There is extremely dense fibrous tissue in the upper outer quadrants of both breasts. This lowers the sensitivity of mammography. B.B. was placed in the region of palpable abnormality and demonstrates dense breast tissue in this region. An occasional benign-appearing calcification is present in both breasts.

Analysis Result: Positive

Record with Two Mis-spelled Words

There is extremely dense fibrous tissue in the upper outer quadrants of both breasts. This lowers the sensitivity of mammography. B.B. was placed in the region of palpable abnormality and demonstrates dense breast tisue in this region. An occasional benign-appearing calcifcaton is present in both breasts.

Analysis Result: Negative

Hotflip: White-box adversarial examples for text classification, in Proceedings of 56th Annual Meeting of the Association for Computational Linguistics, 2018

Fool YOLOv2 Object Detector by a real picture ...
`https://www.youtube.com/watch?v=MIbFvK2S9g8`

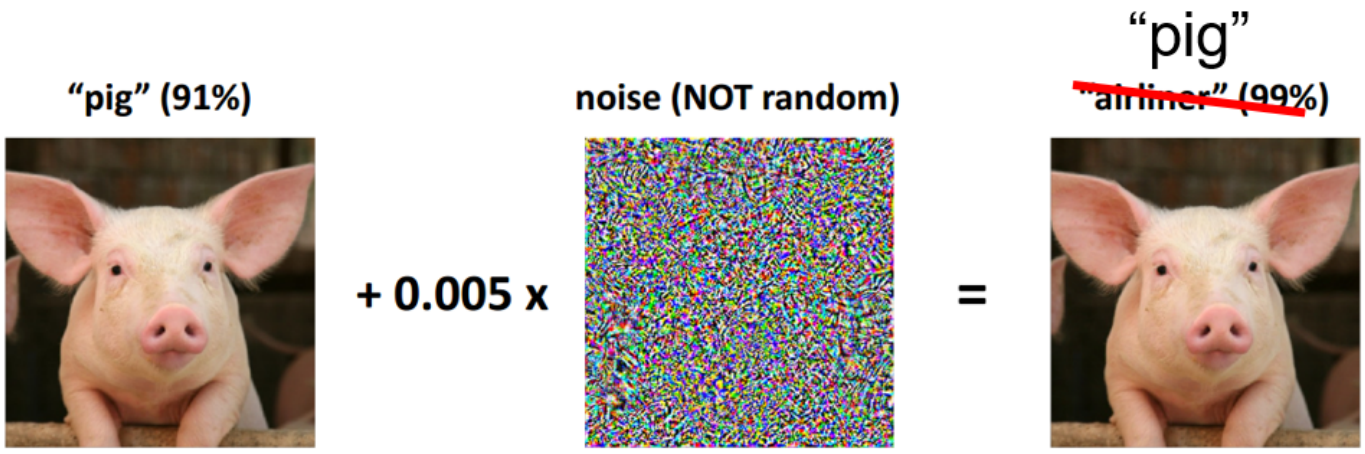
Fooling automated surveillance cameras: adversarial patches to attack person detection, CVPR Workshops. 2019.

Adversarial Examples in Running Systems

Fool an Object Classifier by a 3-D printed turtle ...
`https://www.youtube.com/watch?v=XaQu7kkQBPc`

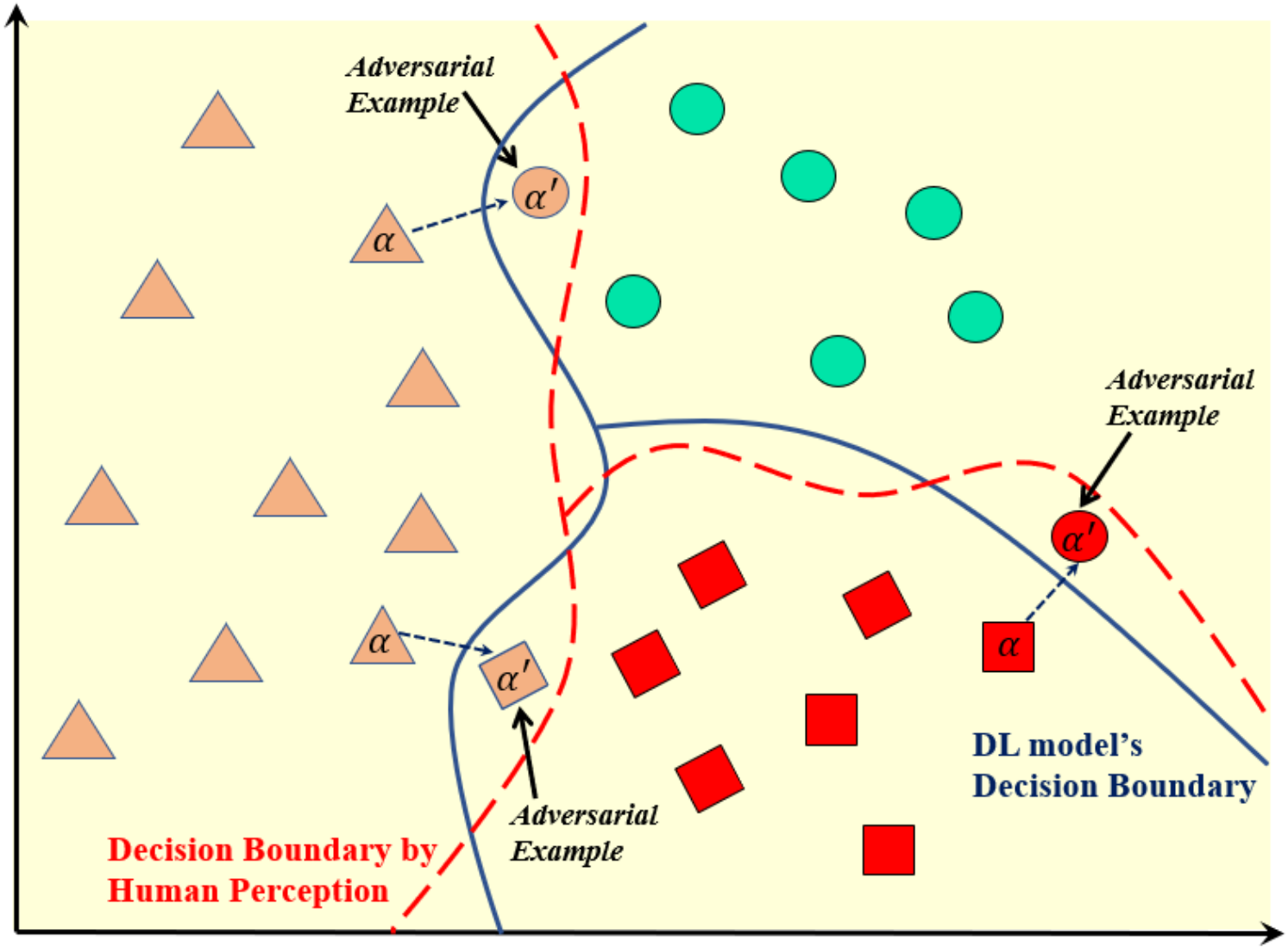
Synthesizing Robust Adversarial Examples. ICML 2018

Researches Dealing with Adversarial Robustness



- Adversarial Attacks: How to find the weak spots of deep learning models → evaluating adversarial robustness
- Adversarial Defense: How to defend adversarial attacks → Improving the robustness w.r.t. adversarial attacks
- Verification: How to verify if a given model satisfies robustness properties for certain input constraints → Providing **robustness guarantees** if no counter-examples can be found

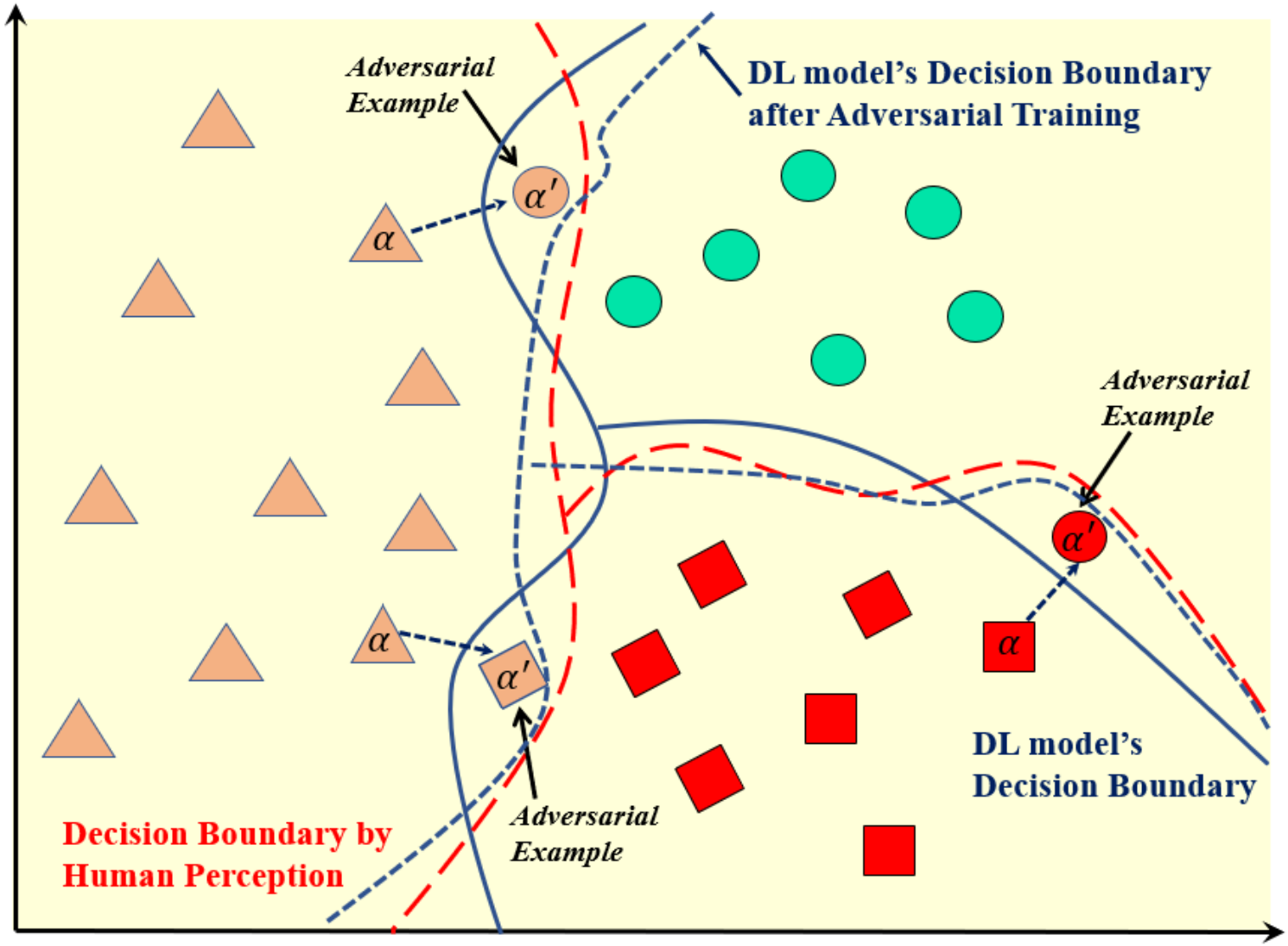
What is adversarial examples



DL model: classifies α and α' **differently**

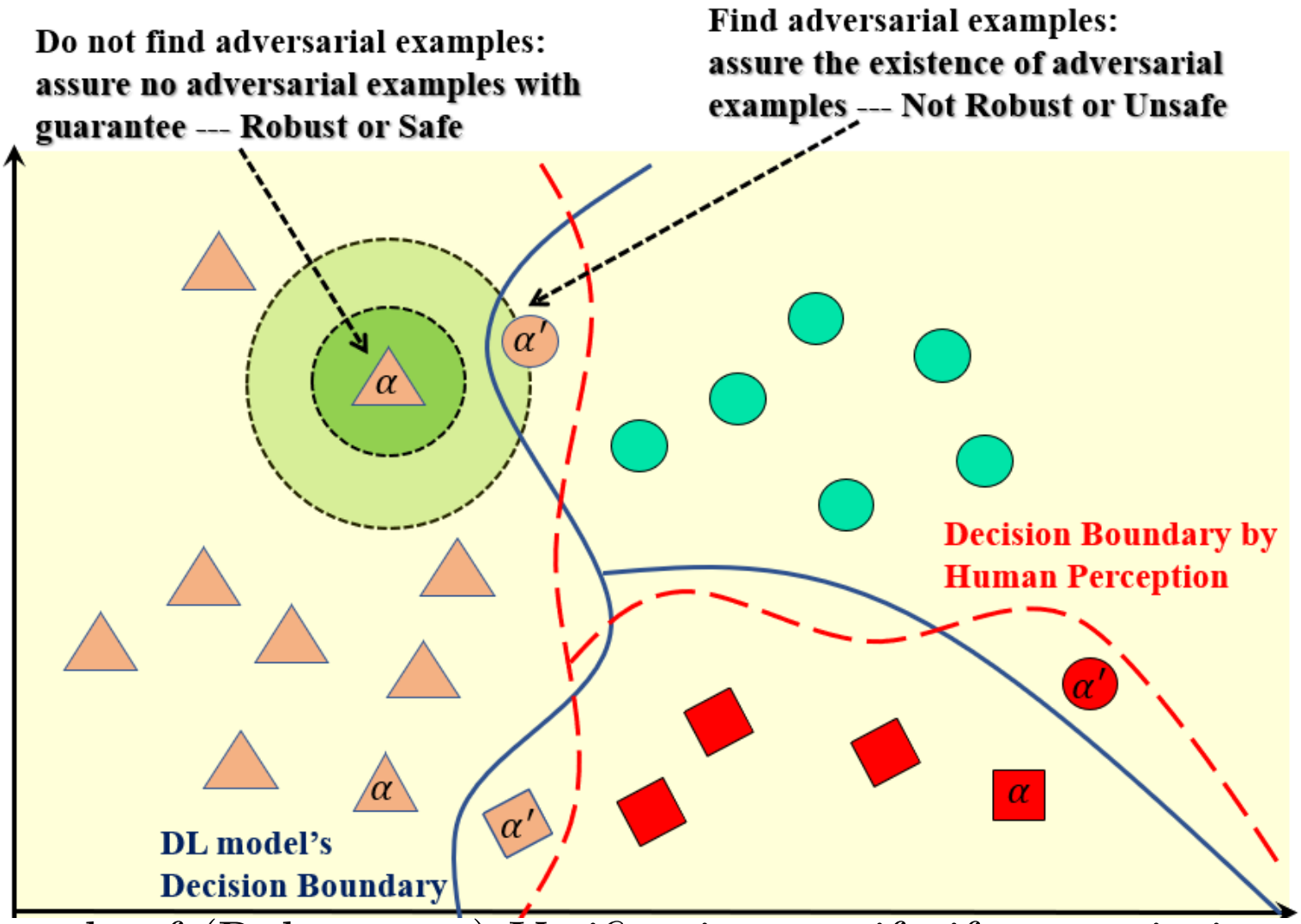
Human: should remain the **same**

An example of Defense



Injecting adversarial examples into training so the resulting DL model is **resistant** to adversarial attacks

An example of Verification



Example of (Robustness) Verification: verify if a certain input area can exclude adversarial examples with **guarantees**

The Aim of the Tutorial

This tutorial aims to cover a few **well-established** works from three aspects: Attack, Defense and Verification.

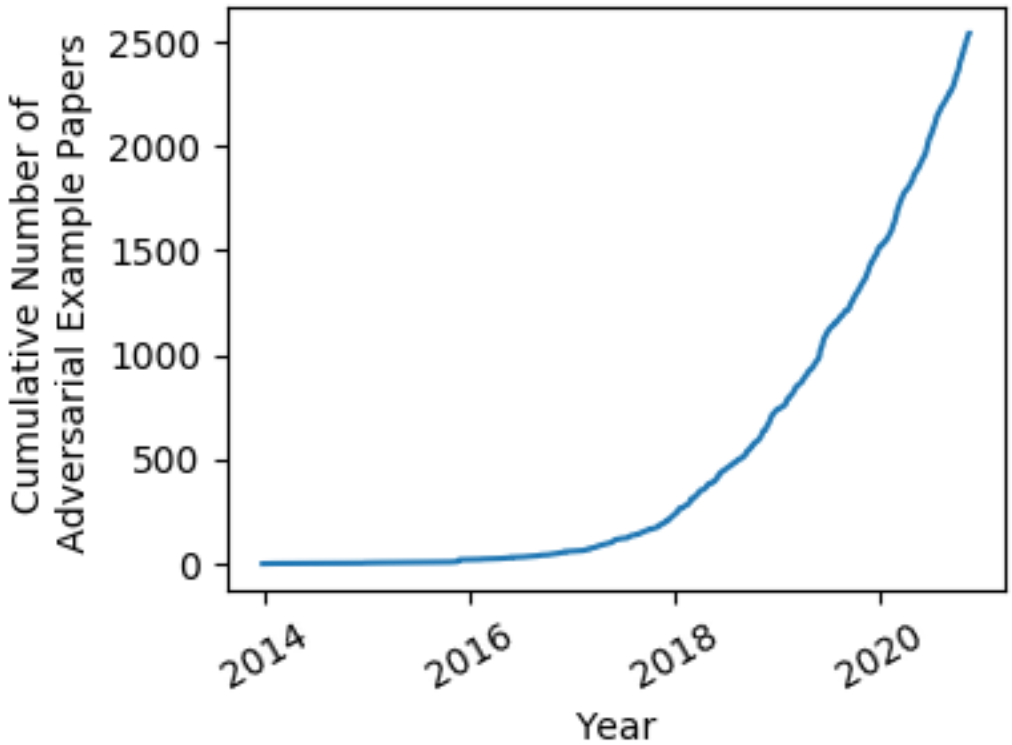


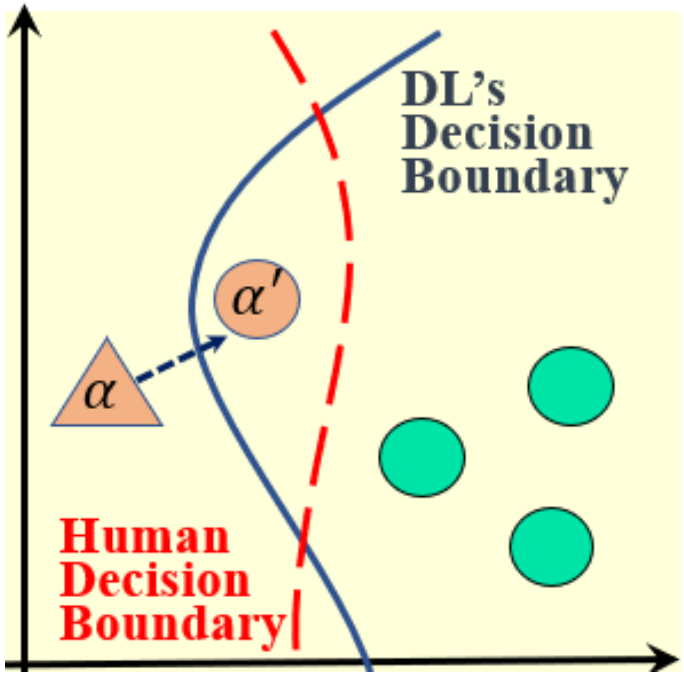
Figure: `https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html`

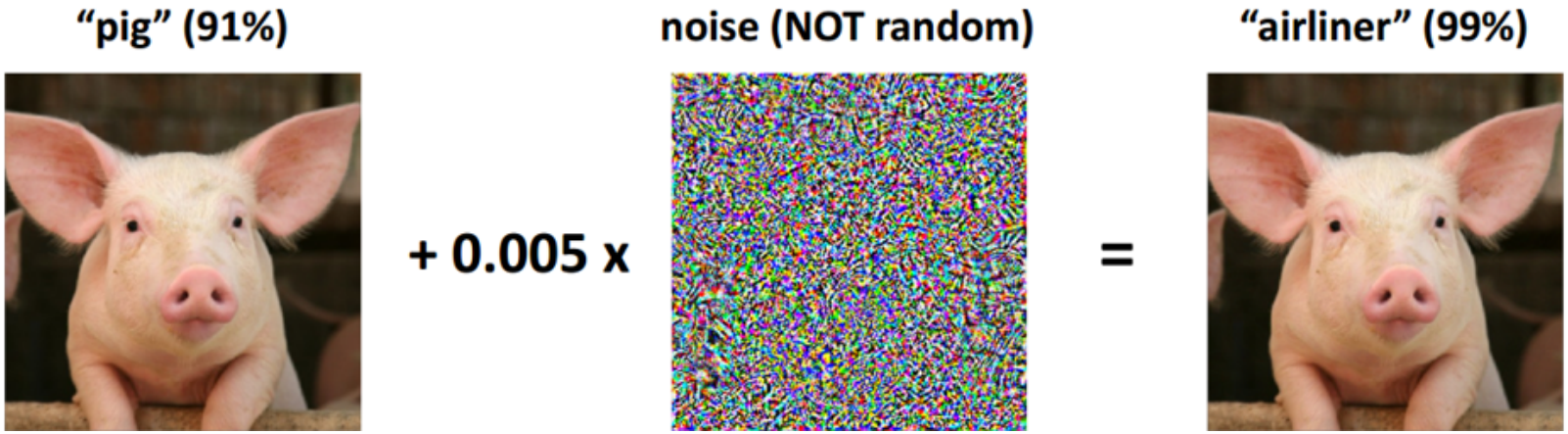
Comprehensive one: *A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, Testing, Adversarial Attack and Defence, and Interpretability, Computer Science Review. 37 (2020): 100270.*

Adversarial Attacks (Wenjie)

What is Adversarial Example

- Input: DL model f
  A correctly-classified, genuine example α
- Aim: find a perturbed example α', such that
  - f produces a different decision on α'
  - Human will produce a same decision on α and α'

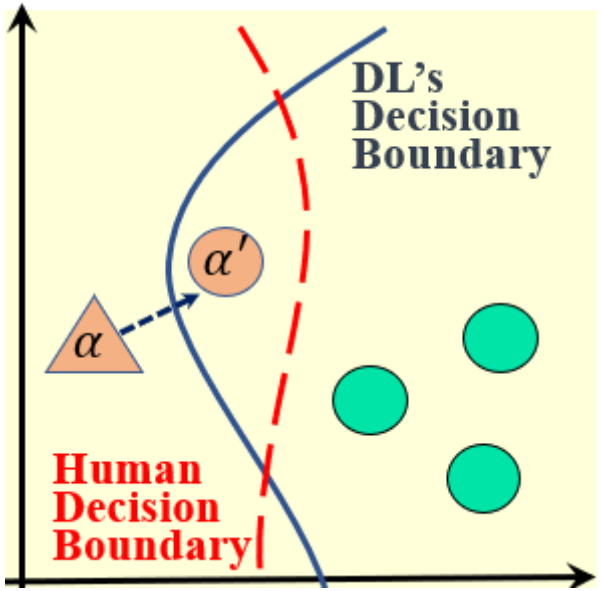




What is Adversarial Example

Problem: Given a DL model f and genuine example α, find α' such that

- $f(\alpha') \neq f(\alpha)$
- $D_{Human}(\alpha') = D_{Human}(\alpha)$



How to approximate human decision?

- Use certain distance metric to assure α' and α are small enough

How to search α' such that $f(\alpha') \neq f(\alpha)$?

- Design certain objective functions for minimization

What kind of information is required from DL model f?

- White-box v.s. Black-box

Targeted Attacks v.s. Un-targeted Attacks

- With a targeted perturbation, the attacker is able to **control** the resulting misclassification label.
- With an un-targeted perturbation, the attacker can enable the misclassification but **cannot control** its resulting misclassification label.

Distance metric to measure α' and α:

- L_p-norm distance
  - L_p-norm based attacks (e.g., $p = 0, 2, \infty$)
- Total variation of pixel displacement
  - Spatial-transformed adversarial attacks
- Metrics for measuring similarity of sentences or text
  - Attacks on NLP models

The information is required from DL model f (white-box or black-box):

- Hard labels only
- Confidence values
- Model's parameters and structure

Categories of Adversarial Attacks

The type of the model f:

- Feed-forward neural networks
- Recurrent neural networks
- Graph neural networks
- Other models

Categories of Adversarial Attacks

Local adversarial attack v.s. Universal adversarial attack

- Local adversarial attack:
  - find specific perturbation for **each** input
- Universal adversarial attack:
  - find a perturbation that can fool a **set** of inputs

Limited-Memory BFGS Attack (L-BFGS)

One of earliest adversarial attack[1]: optimization based formulation with L_2-norm metric

- Model $f : R^{s_1} \rightarrow \{1 \dots s_K\}$ with s_K labels
- $x \in R^{s_1} = [0, 1]^{s_1}$ is an input
- $t \in \{1 \dots s_K\}$ is a target misclassification label

Find the adversarial perturbation r via

$$\begin{aligned} & \min ||r||_2 && \text{assure human-decision unchanged} \\ s.t. \quad & \arg\max_l f_l(x + r) = t && \text{assure misclassification} \\ & x + r \in R^{s_1} && \text{assure perturbed image feasible} \end{aligned} \tag{1}$$

- Solved by the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS)
- Establish this direction

[1] Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)

FGSM Attack

Fast Gradient Sign Method[2] is able to find adversarial perturbations with a fixed L_∞-norm constraint **very efficiently**

- θ: the model parameters,
- x, y: the input and the label
- $J(\theta, x, y)$: the loss function

Find adversarial perturbation r by linearizing the loss function around the current value of θ,

$$r = \epsilon \, \text{sign} \left(\nabla_x J(\theta, x, y) \right) \tag{2}$$

- A **one-step modification** to all pixel values to increase the loss function with a L_∞-norm constraint ϵ

[2] Explaining and harnessing adversarial examples. arXiv:1412.6572 (2014).

Carlini & Wagner Attack[3]: find adversarial examples with **very small distortion**, work on L_0, L_2 and L_∞-norm

- x is an input,
- r is adversarial perturbation
- F is a designed surrogate function such as $x + r$ is able to fool the neural network when it is negative

$$\min \ \ell(r) = ||r||_p + c \cdot F(x + r) \tag{3}$$

- The optimizer Adam was directly adopt to solve this optimization problem
- The key to achieve **strong attack** is a careful design of surrogate function

[3] Towards evaluating the robustness of neural networks. 2017.

ZOO Attack

White-box setting: full access to the target model
Black-box setting: with **limited knowledge** on the model

ZOO Attack[4]: Model $F(x) \in [0,1]^K$ (confidence values)

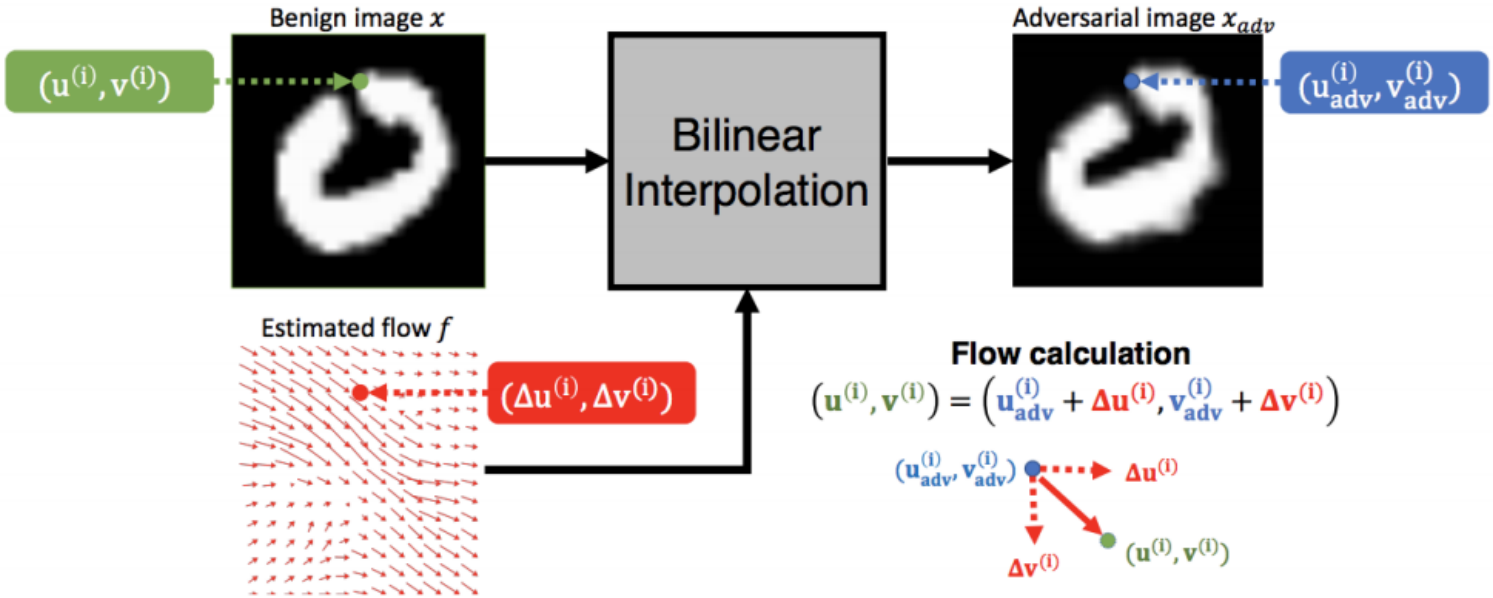
- $\min_x ||x - x_0||_2^2 + c \cdot f(x,t)$
  where $x \in [0,1]^p$ and
  $f(x,t) = max\{-\kappa, max_{i \neq t}[logF(x)]_i - [logF(x)]_t\}$
- Random coordinate gradient descent via estimated gradients by Symmetric Difference Quotient:
  $\frac{\partial g(x)}{\partial x} \approx \frac{g(x+he) - g(x-he)}{2h}$ with small h

- Nearly similar performance to white-box attack
- Key difference: only access confidence values $\rightarrow$ numerically estimate the gradient

[4]Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models, the 10th ACM Workshop on Artificial Intelligence and Security. 2017.

Spatially Transformed Adversarial Examples

What else can we modify? Perturb the locations of pixels



$f^* = arg\min_f \mathcal{L}_{adv}(x, f) + \tau \mathcal{L}_{flow}(f)$ minimize flow

$\mathcal{L}_{adv}(x, f) = \max(\max_{i \neq t} g(\mathbf{x}_{adv})_i - g(\mathbf{x}_{adv})_t, \kappa)$ surrogate function

Measure the pixel displacement:

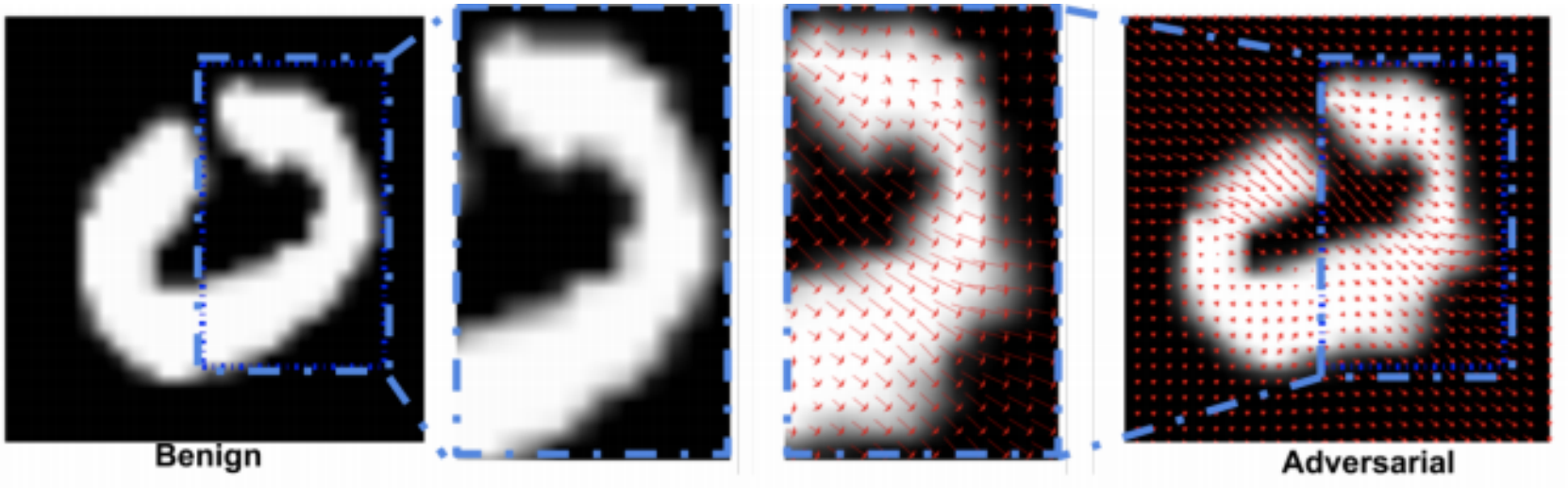
$$\mathcal{L}_{flow}(f) = \sum_{n=1}^{pixels} \sum_{q \in \mathcal{N}(p)} \sqrt{||\Delta u^{(p)} - \Delta u^{(q)}||_2^2 + ||\Delta v^{(p)} - \Delta v^{(q)}||_2^2}$$

Perform spatial transformation:

$$\mathbf{x}^{(i)}_{adv} = \sum_{q \in \mathcal{N}(u^{(i)}, v^{(i)})} \mathbf{x}^{(q)} (1 - |u^{(i)} - u^{(q)}|)(1 - |v^{(i)} - v^{(q)}|)$$

Spatially Transformed Adversarial Examples

Flow visualization on MNIST: digit "0" is misclassified as "2"

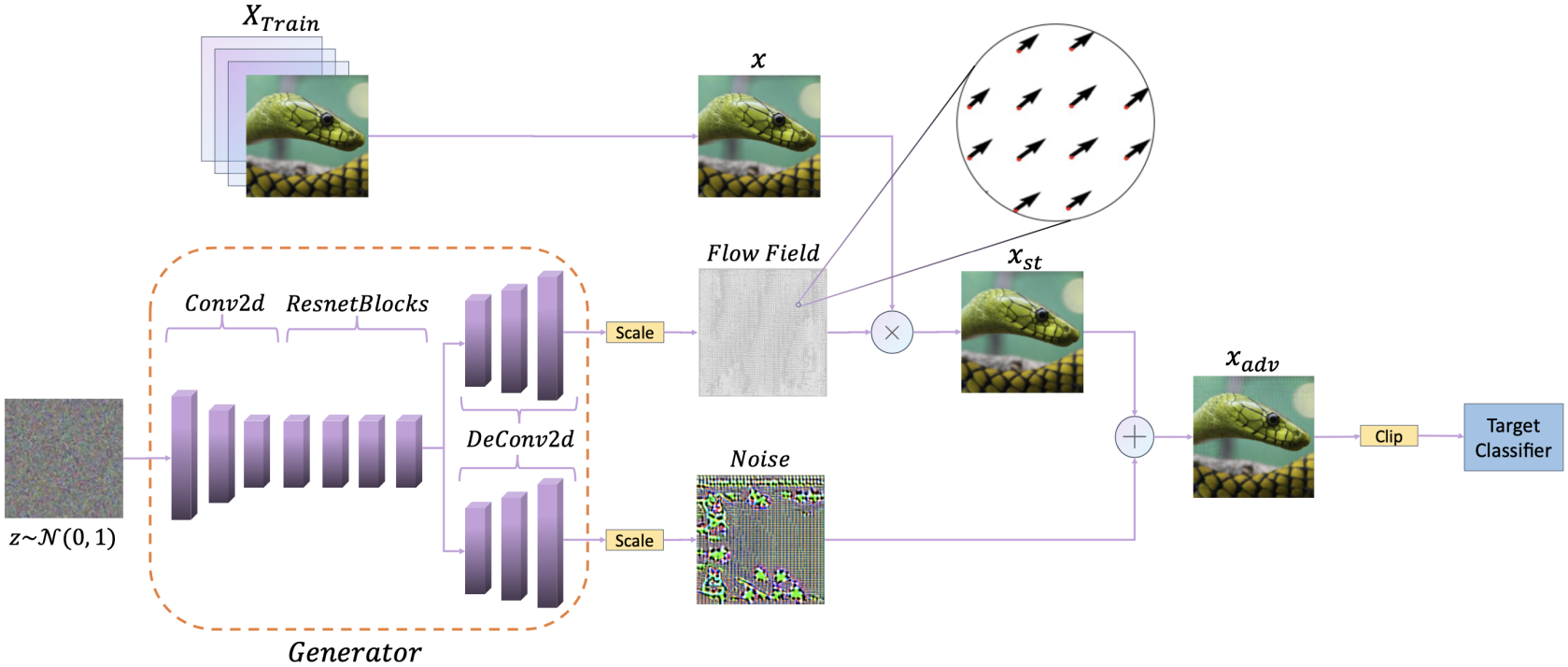


- Instead of perturbing the pixel values, adversarial attacks can be achieved by **spatial transformation**[5]
- Different metric is required to measure pixel's **spatial** displacement

[5] Spatially transformed adversarial examples. ICLR 2018

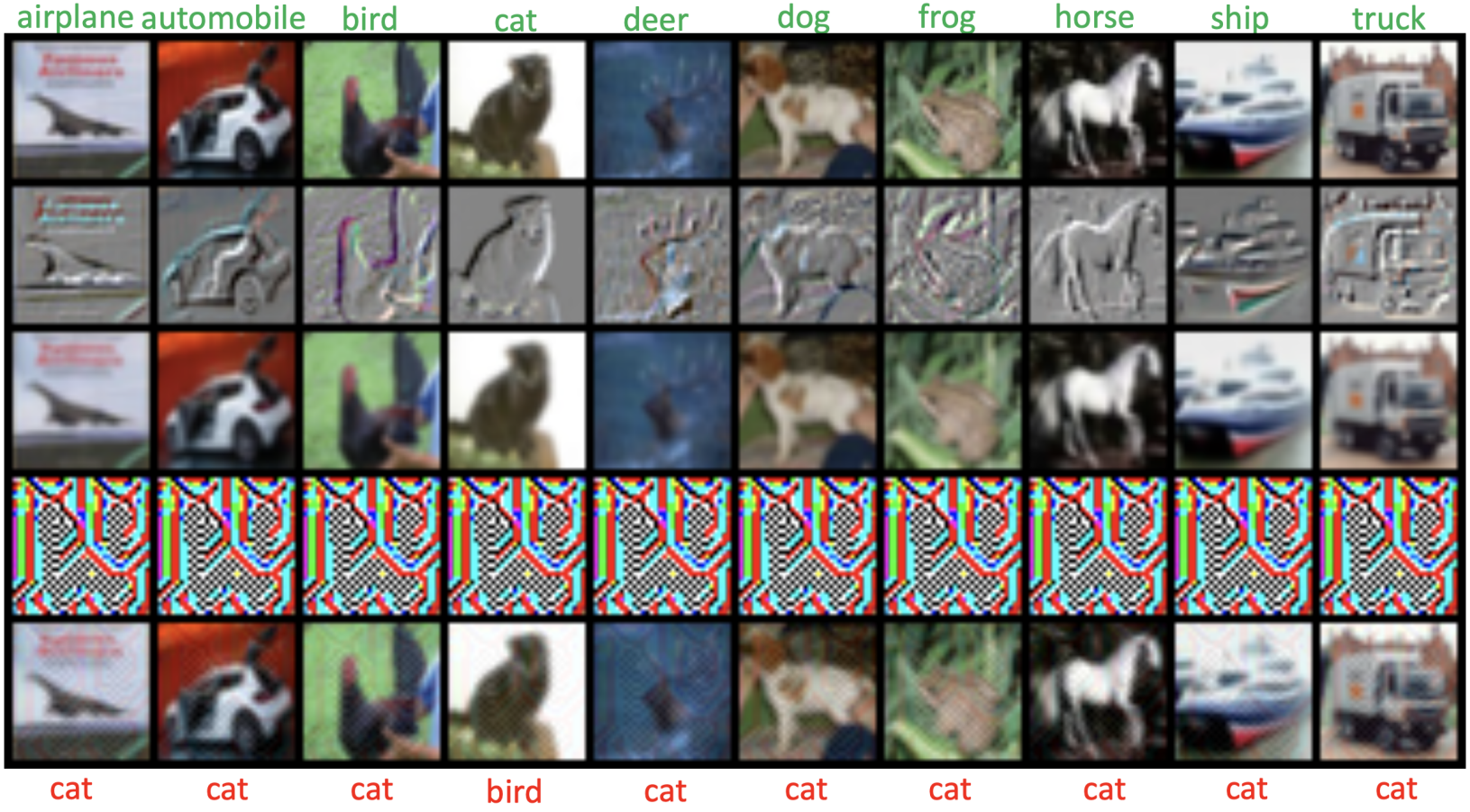
Universal Attack via Combined Perturbation

How about perturbing spatial location and values of pixels simultaneously on a set of images?



- **Unified solution**: L_p-norm, spatial-transformed, or both
- **Universal**: a single perturbation fools a set of input images
- **Strong transferability**: is workable across various unseen models in a black-box setting

Universal Attack via Combined Perturbation



Generalizing Universal Adversarial Attacks Beyond Additive Perturbations, ICDM 2020

- https://underline.io/events/49/sessions/1053/lecture/5329-dm1084---generalizing-universal-adversarial-attacks-beyond-

in SESSION 20 SECURITY AND PRIVACY, DM1084

More Attacks: on NLP models

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

Adversarial attack[6] on **reading comprehension** system

- Adding distracting sentences (in blue)
- Prediction changes from correct one (green) to incorrect (red)

[6] Adversarial Examples for Evaluating Reading Compre

More Attacks: on NLP models

Task: Sentiment Analysis. ***Classifier:*** Amazon AWS. ***Original label:*** 100% Negative. ***Adversarial label:*** 89% Positive.

Text: I watched this movie recently mainly because I am a Huge fan of Jodie Foster's. I saw this movie was made right between her 2 Oscar award winning performances, so my expectations were fairly high. ~~Unfortunately~~ **Unf0rtunately**, I thought the movie was ~~terrible~~ **terrib1e** and I'm still left wondering how she was ever persuaded to make this movie. The script is really ~~weak~~ **wea k**.

Edit adversarial attack[7] on **sentiment analysis** system.
- After editing words (red), prediction changes from 100% of *Negative* to 89% of *Positive*.

Original
Perfect performance by the actor → **Positive (99%)**

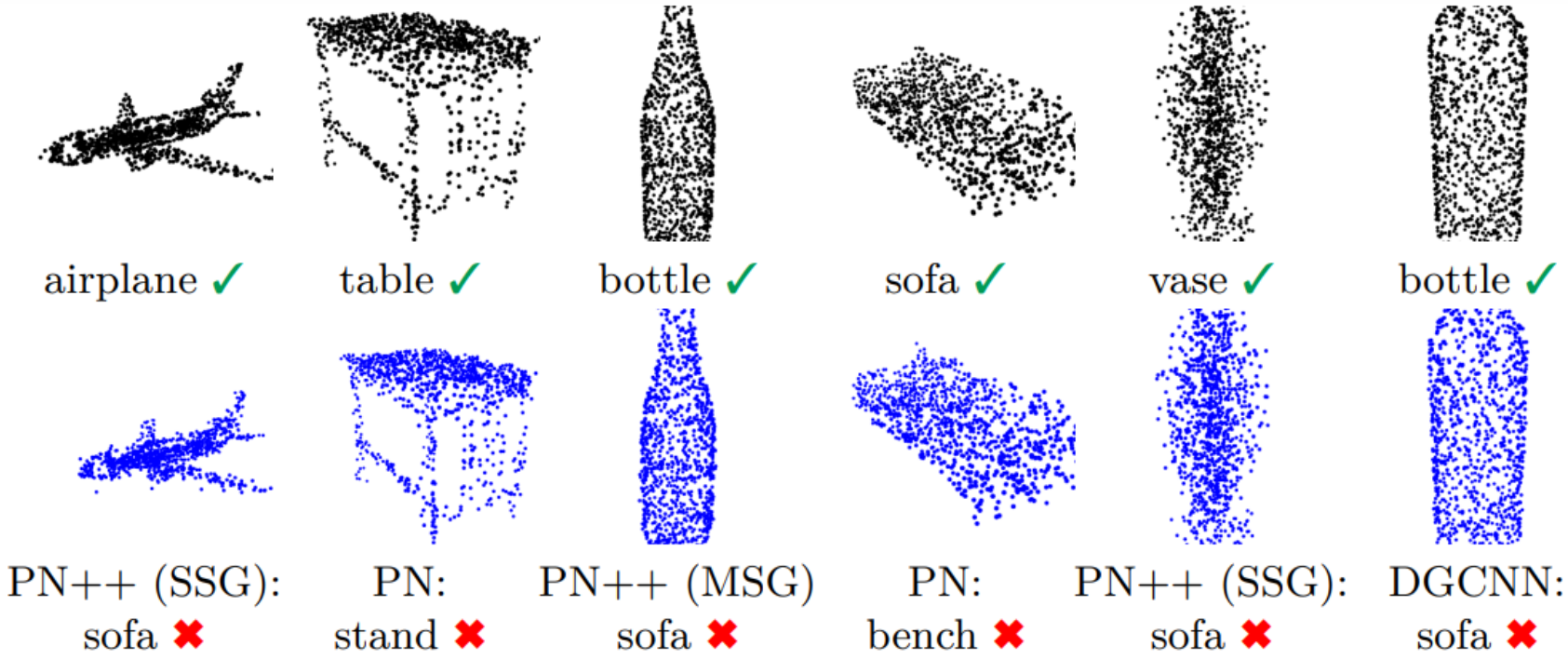
Adversarial
Spotless performance by the actor → **Negative (100%)**

Adversarial attack on[8] **BERT-based sentiment classifier**
-Changing *perfect* into *spotless* completely fools the model

[7] TextBugger: Generating Adversarial Text Against Real-world Applications, NDSS 2019

[8] Is bert really robust? natural language attack on text entailment. arXiv:1907.11932 2019

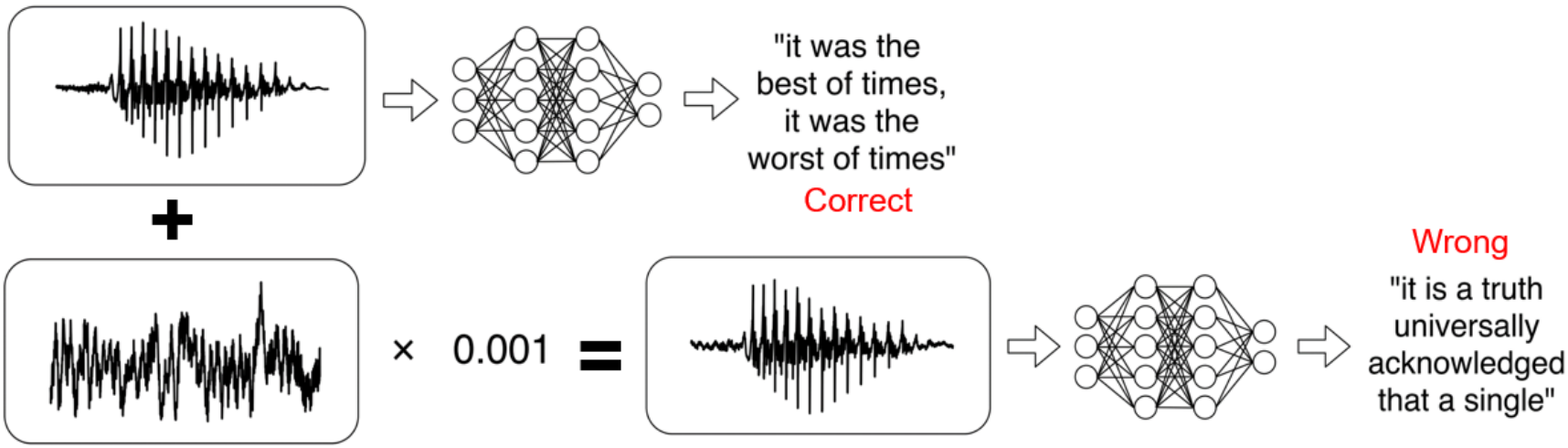
More Attacks: on 3D Point Clouds



Adversarial attacks on multiple **3D Point Cloud** DNNs such as PointNet, PointNet ++ (MSG/SSG) and DGCNN using AdvPC[9]

[9] AdvPC: Transferable Adversarial Perturbations on 3D

More Attacks: on Audio Recognition Systems



Imperceptible adversarial examples[10] [11] can be generated to fool **Audio Recognition Systems** including Google Speech, Bing Speech, IBM Speech APIs, etc

[10] Imperceptible, robust, and targeted adversarial examples for automatic speech recognition, ICML 2019

[11] Practical Hidden Voice Attacks against Speech and Sp... Recognition Systems, NDSS 2019

Existing Tools for Adversarial Attacks

- Adversarial Robustness Toolbox (ART): `https://github.com/Trusted-AI/adversarial-robustness-toolbox`
- Foolbox Native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX `https://github.com/bethgelab/foolbox`
- CleverHans: `https://github.com/tensorflow/cleverhans`
- Advbox Family: `https://github.com/advboxes/AdvBox`

Existing Surveys for Adversarial Attacks

- Bhambri, Siddhant, et al. "A survey of black-box adversarial attacks on computer vision models." arXiv preprint arXiv:1912.01667 (2019).
- Hao-Chen, Han Xu Yao Ma, et al. "Adversarial attacks and defenses in images, graphs and text: A review." International Journal of Automation and Computing 17.2 (2020): 151-178.
- Akhtar, Naveed, and Ajmal Mian. "Threat of adversarial attacks on deep learning in computer vision: A survey." IEEE Access 6 (2018): 14410-14430.
- Huang, Xiaowei, et al. "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability." Computer Science Review 37 (2020): 100270.

... ...

What's the next?

- Adversarial attacks are important
  - **Understanding** the limitation, or potential safety risks of deep learning models
  - Providing a way to **practically evaluate** robustness performance of deep learning models under adversarial environments
  - Issues:
    - it cannot provide robustness guarantees in terms of excluding adversarial examples
    - attack alone cannot directly improve the robustness
- How can we improve its robustness? $\rightarrow$ **Part-2: Defence**
- How to assess the adversarial robustness with provable guarantees $\rightarrow$ **Part-3: Verification**

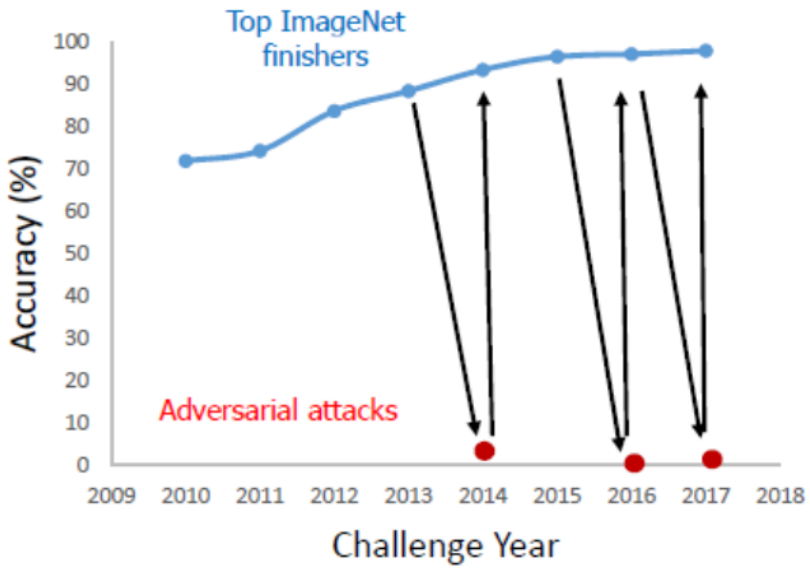
Adversarial Defence (Xinping)

A fast growing research area:

- **Input denoising**
  ◇ e.g., Guo et al (2017); Buckman et al (2018); Liao et al (2018); Samangouei et al (2018); Bai et al (2019); etc.
- **Random smoothing**
  ◇ e.g., Lecuyer et al (2019); Li et al (2019); Cohen et al (2019); Salman et al (2019); Levine & Feizi (2020); Lee et al (2019); Teng et al (2020); Zhang et al (2020); etc.
- **Adversarial training**
  - **Training dataset augmentation**
    ◇ e.g., Goodfellow et al (2014); Shaham et al (2018); Sabour et al (2015); Kurakin et al (2016); Papernot et al (2016); Dezfooli et al (2016); etc.
  - **Robust optimisation**
    ◇ e.g., Goodfellow et al (2015); Madry et al (2017); Zhang et al (2019); Miyato et al (2018); Wang et al (2019); etc.

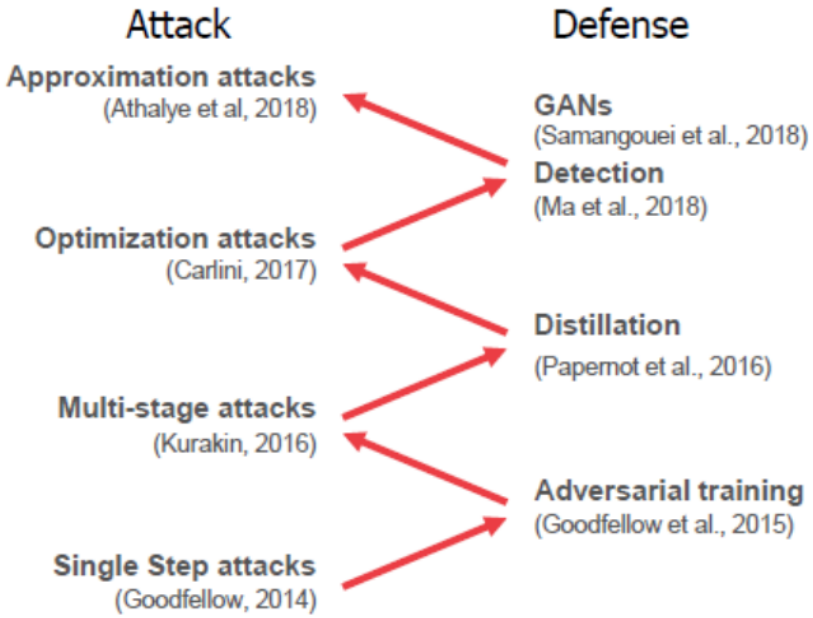
Attack vs. Defence: An Endless Game

Adversarial attacks cause a catastrophic reduction in ML capability



ImageNet Classification

Many defenses have been tried and failed to generalize to new attacks



Attack / Defense Cycle

@ DARPA's GARD programme

Adversarial Training Survives

- Athalye et al (2018). Obfuscated Gradients Give a False Sense of Security. ICML 2018.
- Successful attack of 7 out of 9 defense in ICLR 2018
- The only survival is **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)	[illegible]* 0% Ilyas et al 2019
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%
Na et al. (2018)	CIFAR	0.015 (ℓ_∞)	15%

Adversarial Defence (Xinping)

Adversarial Training

- Madry et al. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks. ICLR 2018.
- Idea: solving a minimax optimisation problem through SGD training

$$\min_{\theta}\{\mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{x'\in S_x} L(x', y; \theta)]\},$$

  - (x, y) - clean training data samples $x \in \mathbb{R}^n$ with labels $y \in [k]$ drawn from the dataset $\mathcal{D}$
  - $L(\cdot)$ - loss function with model parameter $\theta \in \mathbb{R}^m$
  - $x' \in \mathbb{R}^n$ - perturbed samples in a feasible region $S_x \triangleq \{z : z \in B(x, \epsilon) \cap [-1.0, 1.0]^n\}$
  - e.g., $B(z, \epsilon) \triangleq \{z : \|x - z\|_p \leq \epsilon\}$ - the ℓ_p-ball at center x with radius ϵ.

- Outer minimisation can be simulated by SGD training

$$\min_{\theta}\{\frac{1}{N}\sum_{i=1}^{N}[\max_{x'\in S_x} L(x', y; \theta)]\},$$

- How to compute gradient of a maximisation?
  - Danskin's Theorem

$$\nabla_\theta \max L(x', y; \theta) = \nabla_\theta L(x^*, y; \theta)$$

where $x^* = \arg\max L(x', y; \theta)$

- Inner maximisation $\max_{x'\in S_x} L(x', y; \theta)$ can be simulated by finding the worst-case adversarial attacks:
  - Fast Gradient Method (FGM)
  - Projected Gradient Method (PGM):

Fast Gradient Method (FGM)

- Goodfellow et al (2014). Explaining and harnessing adversarial examples. arXiv:1412.6572.
- Idea: Projecting perturbation onto the direction of gradient ascent of loss function
- Adversarial examples:

$$x^* = \arg\max_{x' \in S_x} \langle x' - x, \nabla_\theta L(x, y; \theta) \rangle$$

- For ℓ_∞-norm, FGM recovers the fast gradient sign method (FGSM) where each data point (x, y) is perturbed by the ϵ-normalised sign vector of the loss's gradient

$$x^* = x + \epsilon \cdot \text{sgn}(\nabla_\theta L(x, y; \theta))$$

- Kurakin et al (2016). Adversarial machine learning at scale. arXiv:1611.01236.
- Idea: Iterative gradient ascent to generate strongest adversarial examples, followed by projection back to the feasible region
- Updating rule:

$$x^{t+1} = \Pi_{S_x}(x^t + \alpha \cdot \text{sgn}(\nabla_x L(x^t, y; \theta))),$$

where $\Pi_{S_x}(\cdot)$ projects the inputs onto the region S_x.

- FreeAT:
  - Shafahi et al (2019). Adversarial training for free! NeurIPS 2019.
  - `https://github.com/ashafahi/free_adv_train/`
- YOPO:
  - Zhang et al (2019). You only propagate once: Accelerating adversarial training via maximal principle. NeurIPS 2019.
  - `https://github.com/a1600012888/YOPO-You-Only-Propagate-Once`
- FreeLB:
  - Zhu et al (2020). FreeLB: enhanced adversarial training for language understanding. ICLR 2020.
  - `https://github.com/zhuchen03/FreeLB`
- FastAT
  - Wong et al (2020). Fast is better than free: Revisiting adversarial training. ICLR 2020.
  - `https://github.com/locuslab/fast_adversarial`

Pros and Cons of Adversarial Training

- Pros:
  - Empirical robustness (although no robust certificate)
  - Without affecting inference time (although increasing the training time)
  - Integratable to different threat models

- Cons:
  - Sacrifice accuracy to robustness
  - Dedicated to supervised learning
  - Rely very much on identifying local adversarial examples for specific threat models
  - No guarantees of generalisation performance

The min-max optimisation problem

$$\min_{\theta}\{\mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{x'\in S_x} L(x', y; \theta)]\},$$

where robustness is the goal.

How shall we deal with the following?

- Robustness vs Accuracy
- Supervised vs Semi-supervised Learning
- Local vs Global Information
- Robustness vs Generalisation

Adversarial Defence (Xinping)

Distributionally Robust Optimisation

- Sinha et al (2018). Certifying some distributional robustness with principled adversarial training. ICLR 2018.
- Idea: Considering a Lagrangian penalty formulation of perturbing the underlying data distribution in a Wasserstein ball

$$\min_{\theta} \sup_{P \in \mathcal{P}} \quad \mathbb{E}_{(x,y)\sim P}[L(x, y; \theta) - \gamma W_c(P, P_0)]$$

$$s.t. \quad W_c(P, P_0) \triangleq \inf_{M \in \Pi(P, P_0)} \mathbb{E}_M c(Z, Z')$$

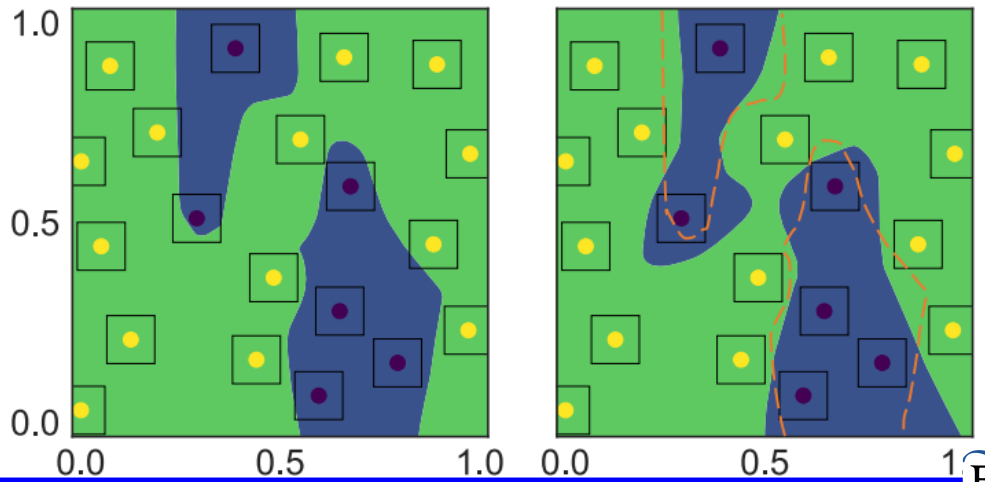
where P_0 is data-generating distribution, $W_c(\cdot, \cdot)$ is Wasserstein metric, and $\mathcal{P} = \{P : W_c(P, P_0) \leq \rho\}$

Robustness vs Accuracy

- Zhang et al (2019). Theoretically principled trade-off between robustness and accuracy. ICML 2019.
- Idea: optimizing a regularised surrogate loss

$$\min_{\theta}\{\mathbb{E}_{(x,y)\sim\mathcal{D}}[L(x,y;\theta)+\beta \max_{x'\in B(x,\epsilon)} \mathrm{KL}(f(x)||f(x'))]\},$$

  - **empirical loss minimisation**: maximise the natural accuracy
  - **regularization term**: push the decision boundary away from the data, so as to improve adversarial robustness



- Miyato et al (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE TPAMI 2018.
- Idea: regularise on local distributional smoothness (LDS)
- virtual adversarial loss with "virtual" label – the distance of the conditional label distributions around each input data point against local perturbation

$$\begin{aligned} \min_{\theta} \quad & \mathbb{E}_{(x,y)\sim\mathcal{D}_l} L(x,y;\theta) + \alpha \mathbb{E}_{x_*\sim\mathcal{D}_l\cup\mathcal{D}_{ul}} LDS(x_*;\theta) \\ \text{s.t.} \quad & LDS(x_*;\theta) \triangleq D(p(\cdot|x_*;\hat{\theta})||p(\cdot|x_* + r_{\text{vadv}};\theta)) \\ & r_{\text{vadv}} = \arg\max_{\|r\|_2\le\epsilon} D(p(\cdot|x_*;\hat{\theta})||p(\cdot|x_* + r)) \end{aligned}$$

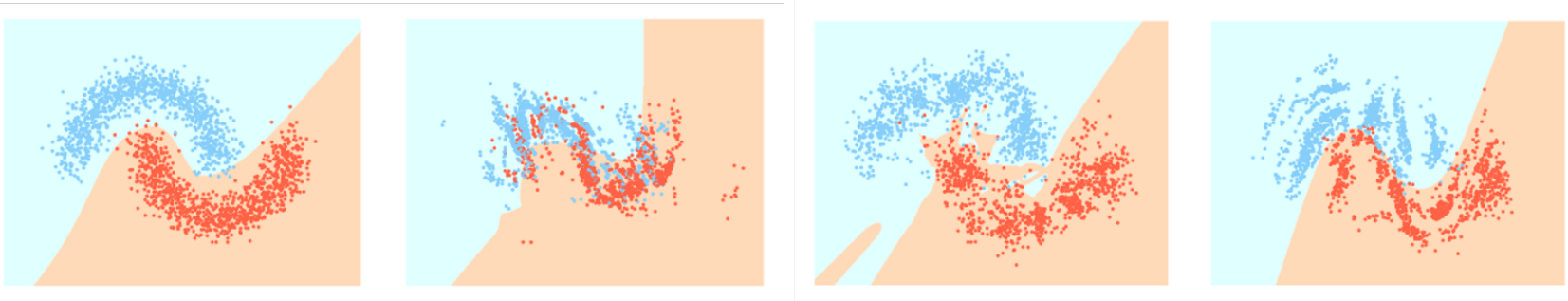
Local vs Global Information

- Improving model robustness with latent distribution locally and globally. ICLR 2021.
- Idea: generate adversarial examples by exploiting both the local and global information of data manifold, via an adversarial game between a discriminator and a classifier

$$\min_{\theta} \quad \{\mathbb{E}_{(x,y)\sim\mathcal{D}}[L(x^{adv}, y; \theta)] + D_f(P_\theta^* || Q_\theta)\} \tag{4}$$

$$\text{s.t.} \quad (x^{adv}, P_\theta^*) = \arg\max_{(x', P_\theta)} [D_f(P_\theta || Q_\theta)] \tag{5}$$

where P_θ and Q_θ latent distributions of x' and x



(a) Original (b) Adversarial Training (c) Feature Scattering (d) Ours

Robustness vs Generalisation

- Schmidt et al (2018). Adversarially robust generalization requires more data. NeurIPS 2018.
  - The sample complexity of robust learning can be significantly larger than that of "standard" learning, and the gap holds irrespective of training algorithms or models
- Farnia et al (2019). Generalizable adversarial training via spectral normalization. ICLR 2019.
  - Adversarial generalisation error depends on spectral norm and Frobenius norm
- Yin et al (2019). Rademacher complexity for adversarially robust generalization. ICML 2019.
  - In the adversarial setting, controlling the ℓ_1 norms of the weight matrices may be a way to improve generalization.
- Wu et al (2020). Adversarial weight perturbation helps robust generalization. NeurIPS 2020.
  - Flattening weight loss landscape by adversarial weight perturbation improves robust generalization

Adversarial Defence (Xinping)

Weight Regularisation

Standard regularisation techniques work for adversarial training to enhance generalisation performance

- Dropout
- Weight decay
- Data augmentation
- Early stopping

Q: Are there any weight regularisation techniques particularly suitable for adversarial training?

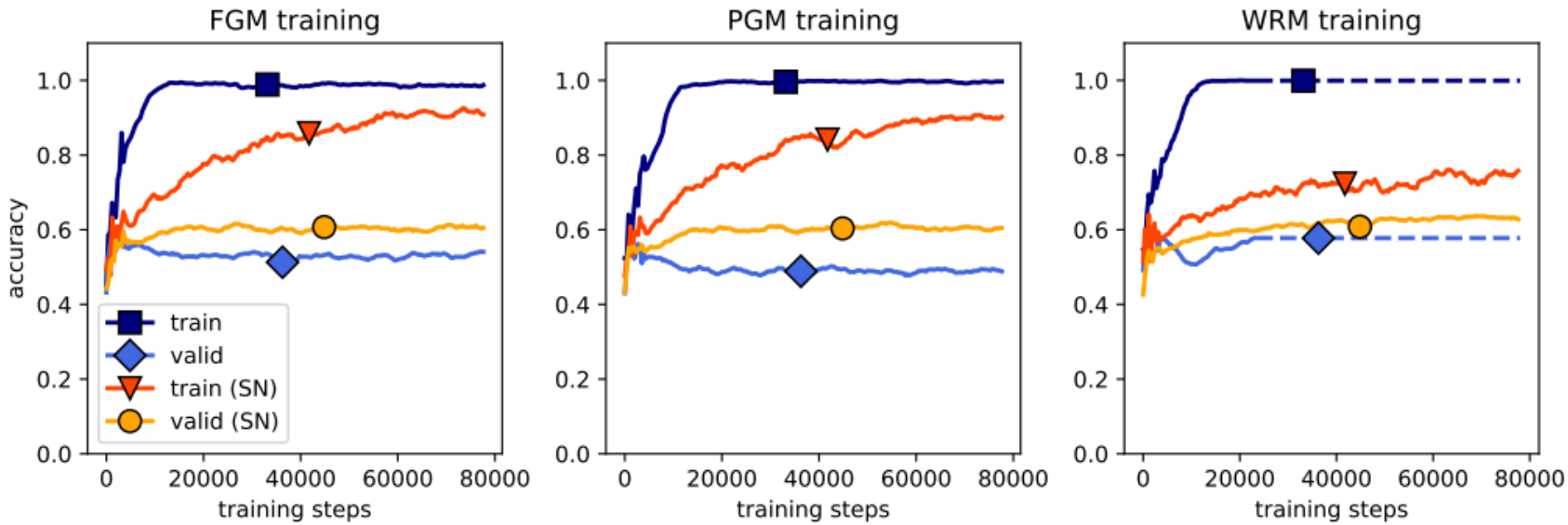
- Spectral normalisation
- Lipschitz regularisation
- Weight correction regularisation

Spectral Normalisation

- Miyato et al (2018). Spectral normalization for generative adversarial networks. ICLR 2018.
  - Spectral normalisation: Normalising weight matrix by spectral norm, i.e., $\mathbf{W}_{\text{SN}} = \frac{\mathbf{W}}{\sigma(\mathbf{W})}$ where

$$\sigma(\mathbf{W}) \triangleq \max_{\|\mathbf{x}\|_2 \leq 1} \|\mathbf{W}\mathbf{x}\|_2$$

- Farnia et al (2019). Generalizable adversarial training via spectral normalization. ICLR 2019.



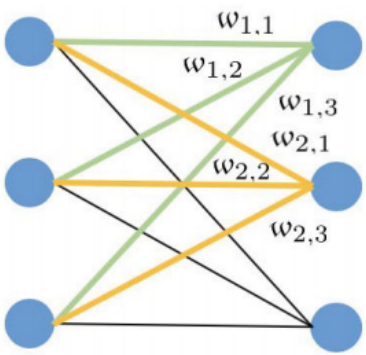
Lipschitz Regularisation

Lipschitz constraints under ℓ_2-norm are useful for provable adversarial robustness bounds, stable training, and Wasserstein distance estimation.

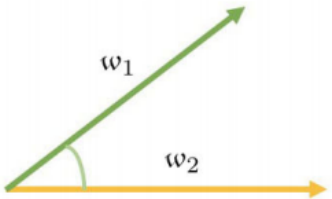
- Cisse et al (2017). Parseval networks: Improving robustness to adversarial examples. ICML 2017.
  - Idea: to maintain weight matrices of linear and convolutional layers to be (approximately) Parseval tight frames (extensions of orthogonal matrices to non-square matrices).
- Li et al (2019). Preventing gradient attenuation in lipschitz constrained convolutional networks. NeurIPS 2019.
  - Idea: to introduce convolutional gradient norm preserving networks with an efficient parameterisation of orthogonal convolutions to avoid the issues of loose bounds on the Lipschitz constant and computational intractability

Weight Correlation Regularisation

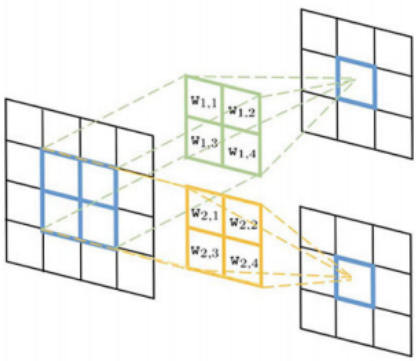
- How does weight correlation affect the generalisation ability of deep neural networks. NeurIPS 2020.
- Idea: Regularisation to constrain the average weight correlation between any two neurons so as to enhance generalisation performance
- Weight correlation



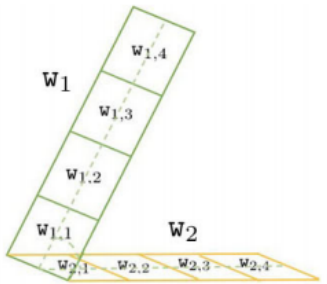
FCN



$$w_1 = (w_{1,1}, w_{1,2}, w_{1,3})$$
$$w_2 = (w_{2,1}, w_{2,2}, w_{2,3})$$
$$\rho(w_1, w_2) = \frac{|\langle w_1, w_2 \rangle|}{||w_1||_2 ||w_2||_2}$$



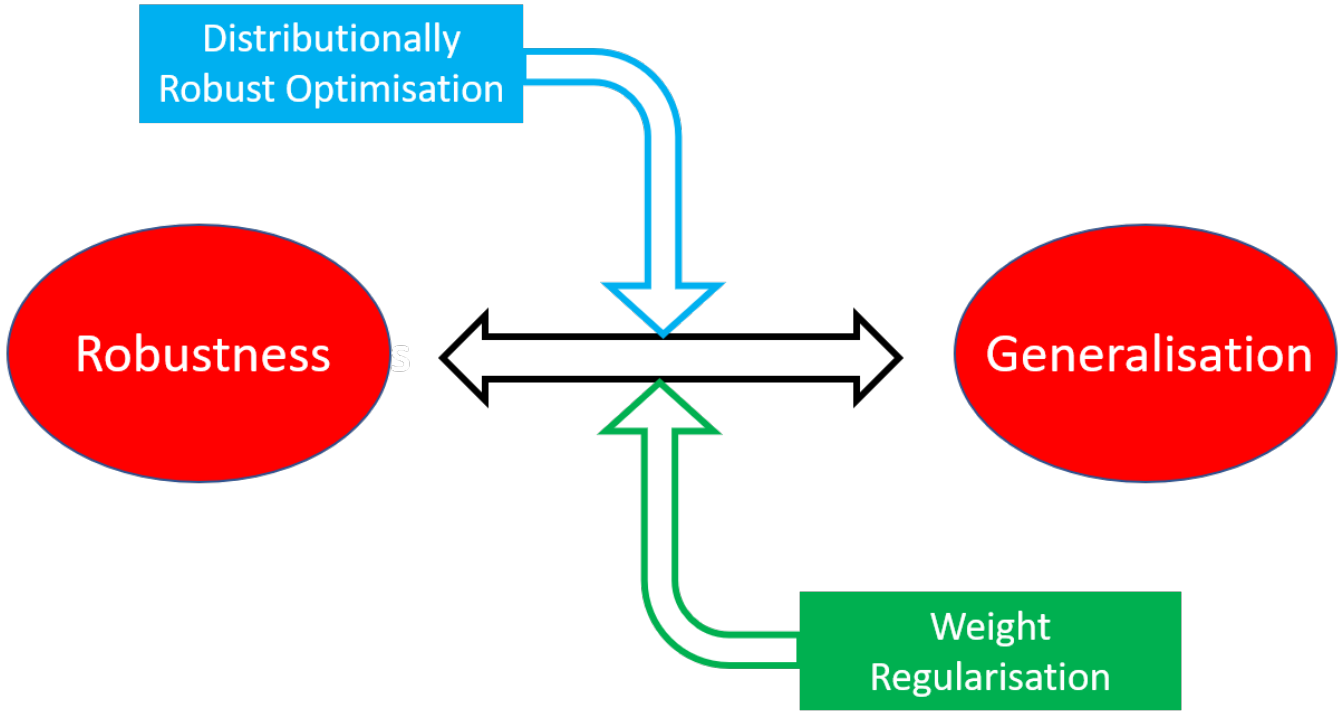
CNN



$$W_1 = (W_{1,1}, W_{1,2}, W_{1,3}, W_{1,4})$$
$$W_2 = (W_{2,1}, W_{2,2}, W_{2,3}, W_{2,4})$$
$$\rho(W_1, W_2) = \frac{|\langle W_1, W_2 \rangle|}{||W_1||_2 ||W_2||_2}$$

Robust Optimisation vs Regularisation

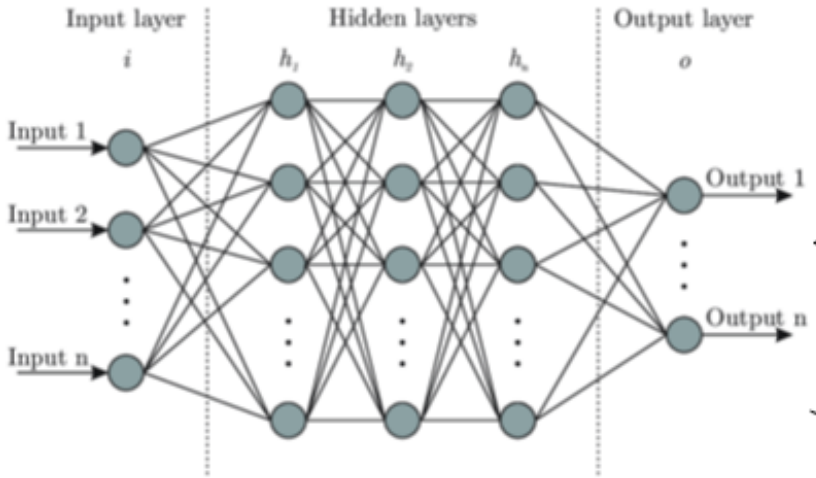
- **Adversarial training** vs **norm regularisation**
  - Roth et al (2020). Adversarial training is a form of data-dependent operator norm regularization. NeurIPS 2020.
  - **Insight**: ℓ_p-norm constrained projected gradient ascent based adversarial training with an ℓ_q-norm loss on the logits of clean and perturbed inputs is equivalent to data-dependent (p, q) operator norm regularization
- **Distributionally robust optimisation (DRO)** vs **regularisation**
  - Husain (2020). Distributional robustness with IPMs and links to regularization and GANs. NeurIPS 2020.
  - **Insight**: DRO under any choice of Integral Probability Metrics (IPM) corresponds to a family of regularization penalties, which recover and improve upon existing results in the setting of Maximum Mean Discrepancy (MMD) and Wasserstein distances.



Generalisable Robustness or Robust Generalisation?

Verification (Xiaowei)

What is Verification?



feedback

Counterexamples

fails

Verification Algorithms

holds

Properties, e.g.,
1. robustness,
2. fairness,
3. interpretability,
4. privacy,
etc.

What is Verification for Neural Networks?

- Which properties to verify?
  - robustness – local property for input perturbation
  - generalisation – global property for unseen data
  - security properties such as backdoor
- How to verify?
  - confirm whether or not a property holds
  - finding counterexamples
  - statistical evaluation
  - lower/upper bounds of a certain quantity

Verification (Xiaowei)

↪Approach 1 – Formal Verification

Definition of Robustness Verification

Given a network $\mathcal{N}$ with its associated function f, a distance measure $||\cdot||$, an input $\vec{x}_0 \in \mathbb{R}^{s_1}$, and the distance d, we have

$$\begin{aligned} & \min_{\vec{x}} ||\vec{x} - \vec{x}_0|| \\ s.t. \quad & \vec{x} \in \mathbb{R}^{s_1} \\ & \ell(f, \vec{x}) \neq \ell(f, \vec{x}_0) \end{aligned} \tag{6}$$

and then check whether the obtained mimimum distance is smaller than d.

- Constraint Solving based Methods
  - simple, but lack of scalability
- Approximation Methods
  - scales slightly better, one-side bound, bound not improvable
- Converging Bounds Methods
  - scales well, gradually improving bounds

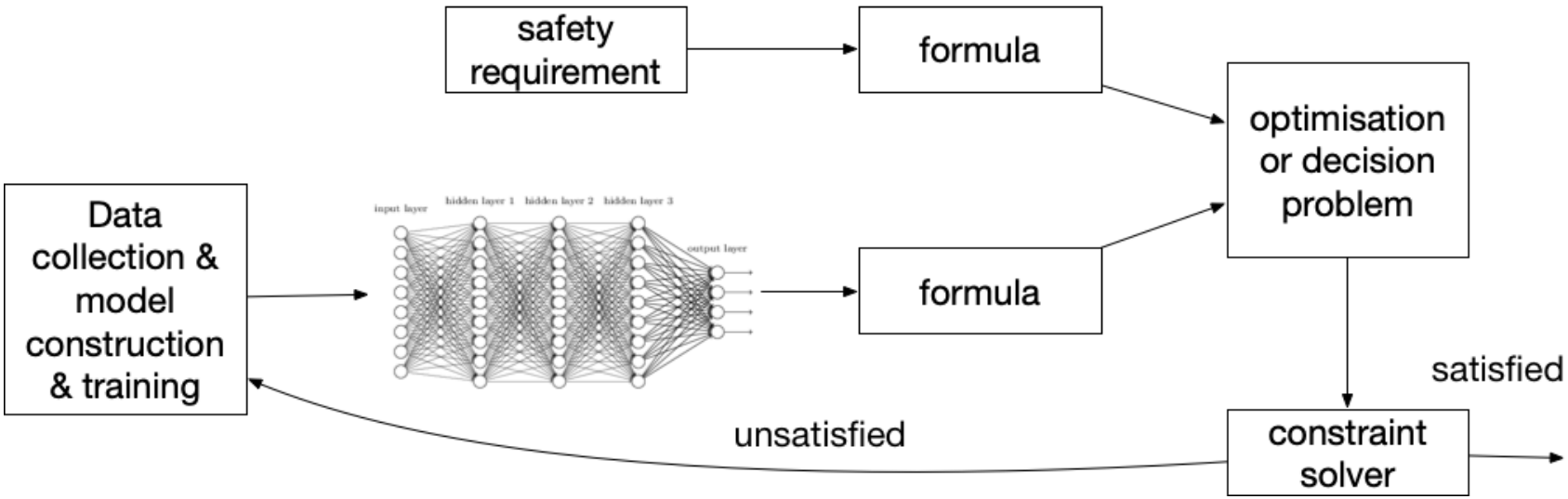
Most of them are for **Robustness Risk**.

Verification (Xiaowei)

↪Approach 1 – Formal Verification

↪Constraint-Solving based Methods

General idea of constraint solving based methods



A hidden layer $\vec{v}_{i+1} = \mathbf{ReLU}(\mathbf{W}_i\vec{v}_i + \vec{b}_i)$ can be described with the following MILP:

$$\begin{aligned} \vec{v}_{i+1} &\geq \mathbf{W}_i\vec{v}_i + \vec{b}_i, \\ \vec{v}_{i+1} &\leq \mathbf{W}_i\vec{v}_i + \vec{b}_i + M\vec{t}_{i+1}, \\ \vec{v}_{i+1} &\geq 0, \\ \vec{v}_{i+1} &\leq M(1 - \vec{t}_{i+1}), \end{aligned} \tag{7}$$

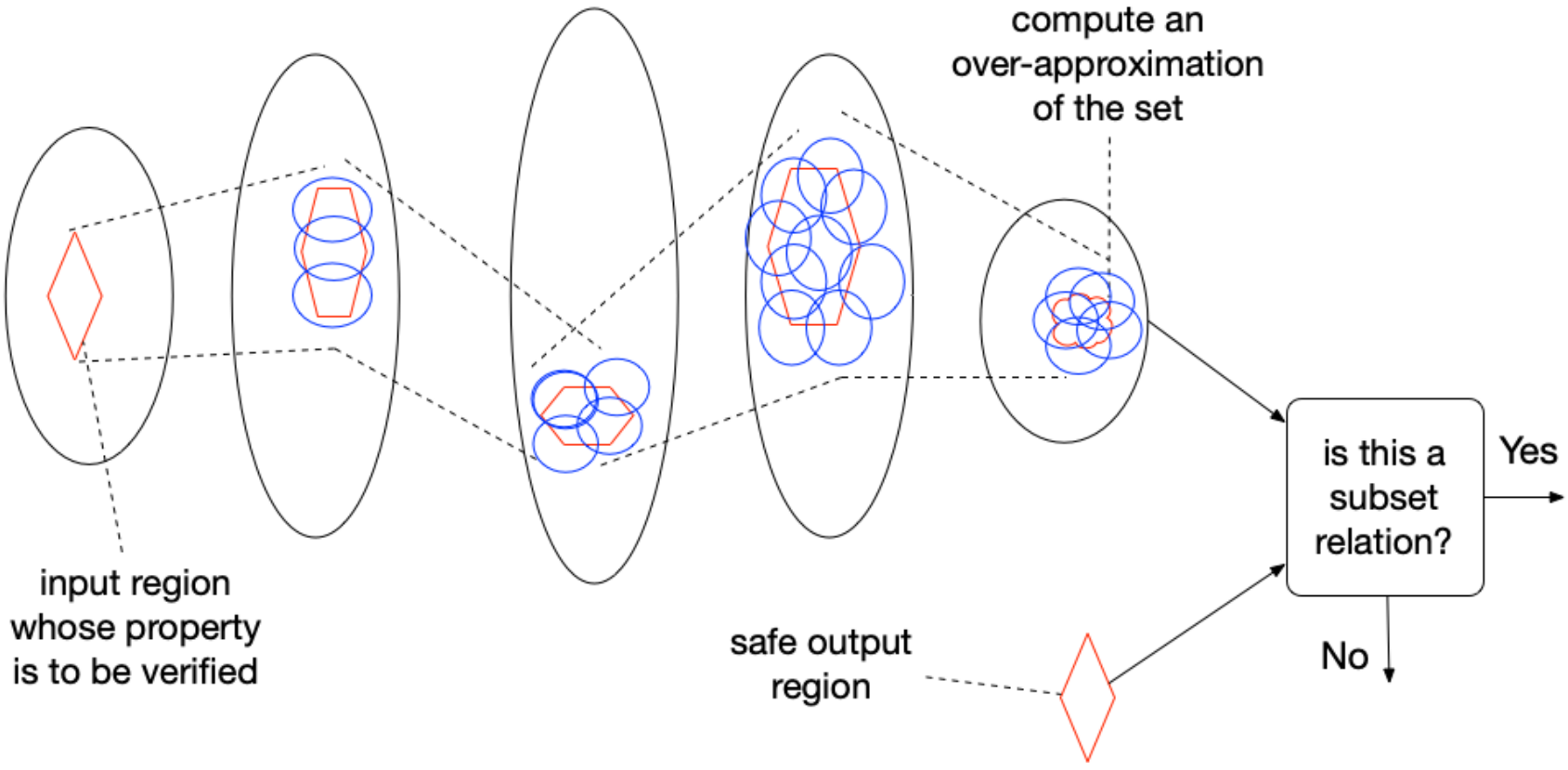
where $\vec{t}_{i+1}$ has value 0 or 1 in its entries and has the same dimension as $\vec{v}_{i+1}$, and $M > 0$ is a large constant which can be treated as ∞.

Verification (Xiaowei)

↪Approach 1 – Formal Verification

↪Approximation Methods

General idea of approximation methods



- Output reachable set estimation and verification for multi-layer neural network. IEEE transactions on Neural Networks and Learning Systems. 2018

- Input: a hyper-rectangle representing a set of inputs
- Output: a hyper-rectangle over-approximating the actual outputs corresponding to the input hyper-rectangle

- conduct layer-by-layer
- assume that the input hyper-rectangle at layer i is represented as

$$\mathcal{V}_{i-1} = \{\vec{v}_{i-1} \mid ||\vec{v}_{i-1} - \vec{c}_{i-1}||_1 \leq \vec{r}_{i-1}\}$$

  where $\vec{c}_{i-1} \in \mathbb{R}^{s_{i-1}}$ is the center of the hyper-rectangle and $\vec{r}_{i-1} \in \mathbb{R}^{s_{i-1}}$ is the radius of the hyper-rectangle.
- we intend to compute

$$\mathcal{V}_i = \{\vec{v}_i \mid ||\vec{v}_i - \vec{c}_i||_1 \leq \vec{r}_i\}$$

  or more concretely, $\vec{c}_i, \vec{r}_i \in \mathbb{R}^{s_i}$.

- Define

$$\vec{w}_i = \sigma_i(\mathbf{W}_{i-1}\vec{c}_{i-1} + \vec{b}_i)$$

-

$$\vec{w}_{i,max} = \sigma_i(\mathbf{W}_{i-1}\vec{c}_{i-1} + |\mathbf{W}_{i-1}|\vec{r}_{i-1} + \vec{b}_i)$$

where $|\mathbf{W}_{i-1}|$ is the element-wise application of abs function on the matrix $\mathbf{W}_{i-1}$.

-

$$\vec{w}_{i,min} = \sigma_i(\mathbf{W}_{i-1}\vec{c}_{i-1} - |\mathbf{W}_{i-1}|\vec{r}_{i-1} + \vec{b}_i)$$

We have that

$$\vec{v}_i = \sigma_i(\mathbf{W}_{i-1}\vec{v}_{i-1} + \vec{b}_i) \in [\vec{w}_{i,min}, \vec{w}_{i,max}]$$

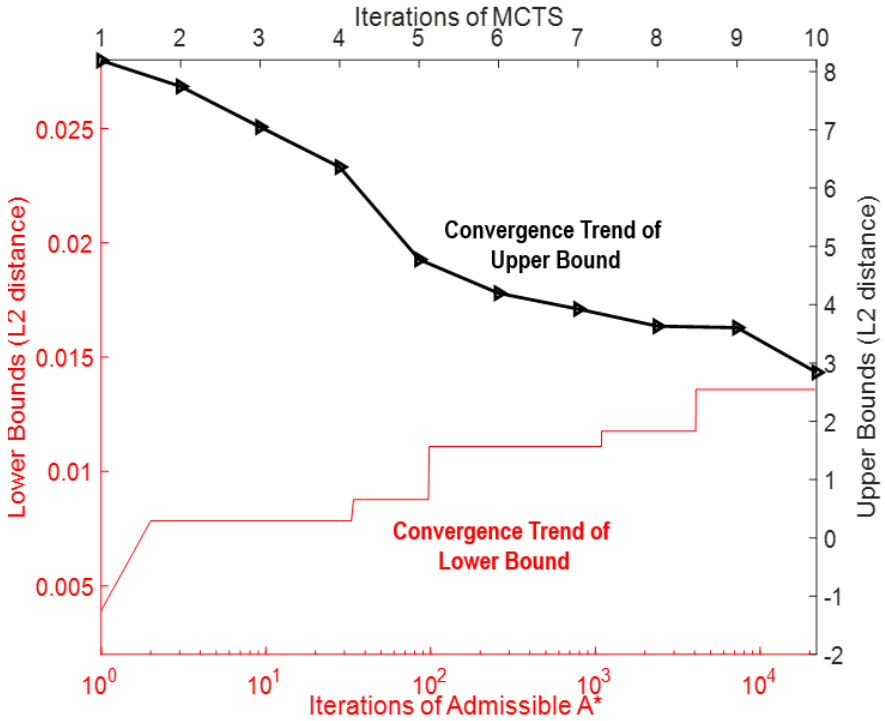
for any $\vec{v}_{i-1} \in \mathcal{V}_{i-1}$.

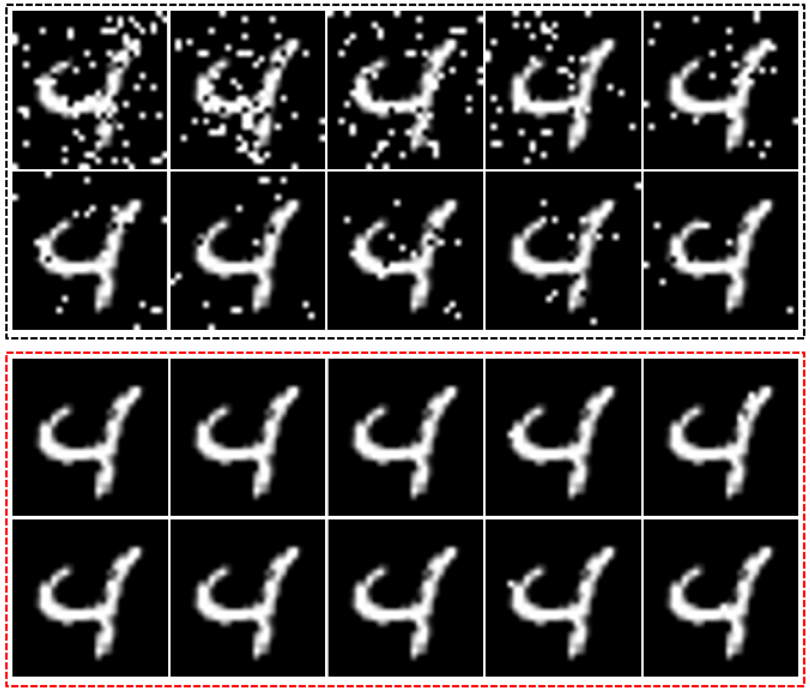
Verification (Xiaowei)

↪Approach 1 – Formal Verification

↪Converging Bounds Methods

General idea of anytime method





- Reachability Analysis of Deep Neural Networks with Provable Guarantees. IJCAI2018.

- Giving an input norm ball, determine the lowest/highest value of a certain Lipschitz function over the output probability distribution.

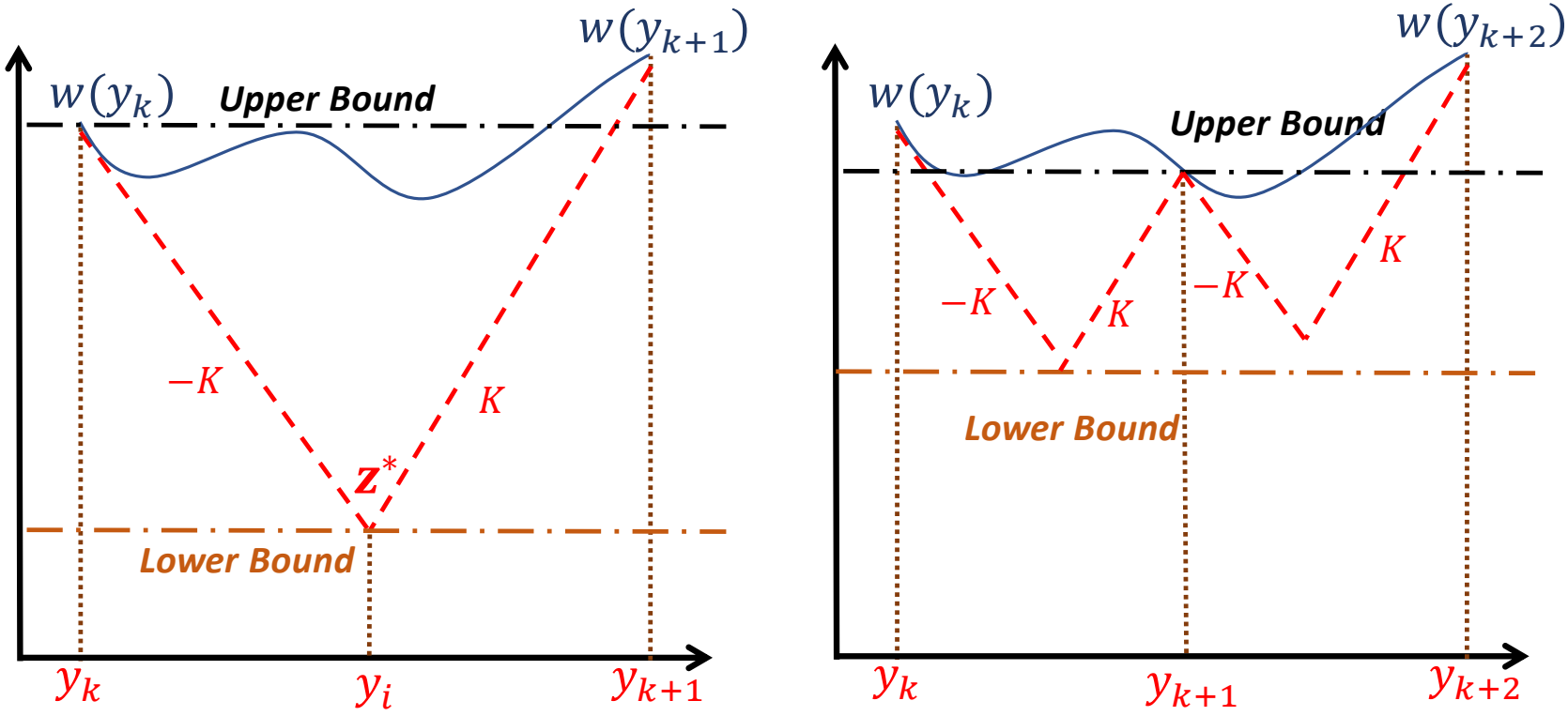


Figure: A lower-bound function designed via Lipschitz constant

Available Tools

- Constraint-Solving:
  - ReluPlex:
    `https://github.com/guykatzz/ReluplexCav2017`
- Approximation:
  - ERAN: `https://github.com/eth-sri/eran`
- Anytime:
  - TrustAI: `https://github.com/TrustAI`

Verification (Xiaowei)

↪Approach 2 – Statistical Evaluation

- Sampling-based methods
- Software testing methods
- Complexity measure based methods

May work with **both robustness and generalisation**.

Verification (Xiaowei)

↪Approach 2 – Statistical Evaluation

↪Sampling-based Methods

- Evaluating The Robustness Of Neural Networks: An Extreme Value Theory Approach, ICLR2018
- A lower bound of L_p minimum adversarial distortion β_L
- Extreme Value Theory ensures that the maximum value of random variables can only follow one of the three extreme value distributions.

- A Statistical Approach To Assessing Neural Network Robustness, ICLR2018
- a naive Monte Carlo sampling does not work well for high-dimensional problems.
- an adaptation of multi-level splitting, a Monte Carlo approach for estimating the probability of rare events.

Verification (Xiaowei)

↪Approach 2 – Statistical Evaluation

↪Software Testing Methods

- Well established in many industrial standard for software used in safety critical systems, such as ISO26262 for automotive systems and DO 178B/C for avionic systems.

- Coverage Metrics
  - structural coverage
  - scenario coverage
- Test Case Generation Methods
  - fuzzing
  - symbolic execution, etc

- to determine if the generated test cases include bugs.

- Industrial standards need to be upgraded
- A few Coverage Metrics
  - DeepXplore: Automated Whitebox Testing of Deep Learning Systems. SOSP 2017
  - Structural Test Coverage Criteria for Deep Neural Networks. EMSOFT 2019
- A few Test Case Generation Methods
  - Concolic Testing for Deep Neural Networks. ASE 2018
- Use a set of generated test cases to either finding bugs or evaluating the performance of a neural network

Verification (Xiaowei)

↪Approach 2 – Statistical Evaluation

↪Complexity Measure based Methods

In machine learning research, the following is an important open question – **what is the source of the generalisation capabilities of deep neural networks?**

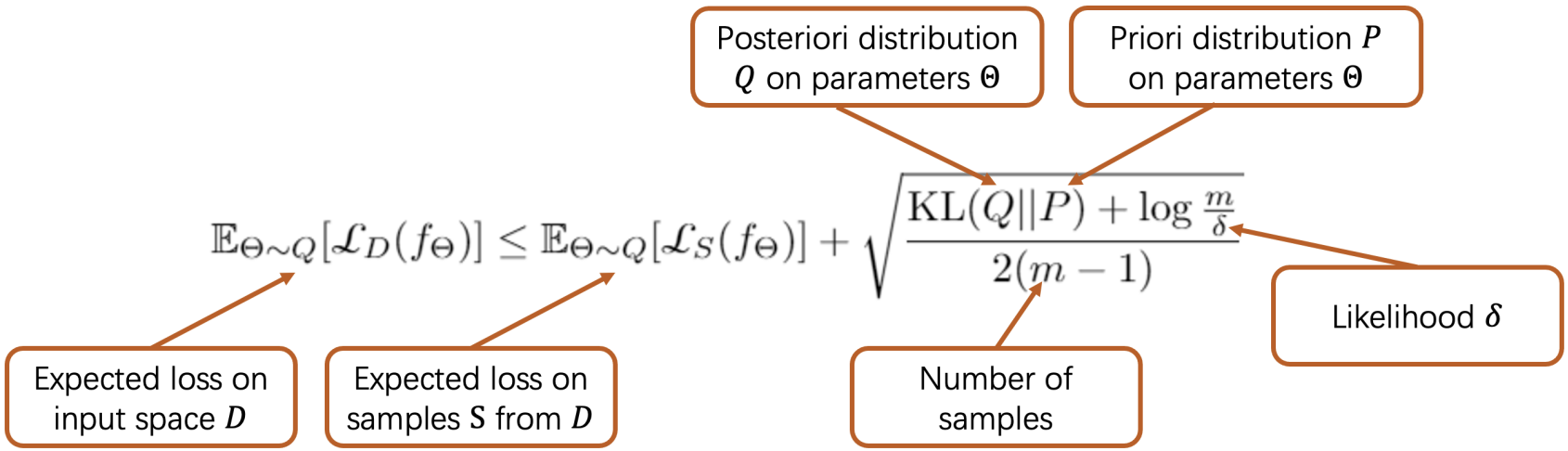
Why important? A better understanding of the problem is essential towards obtaining reliable and robust deep neural network architectures with good performance – discussed in "Adversarial Defence" part.

Here, we concern **how to statistically predict the generalisation capability of a network, without taking a test dataset**.

- **Theoretically motivated measures**
  - PAC-Bayes (McAllester, 1999; Dziugaite and Roy, 2017; Neyshabur et al., 2017)
  - VC-dimension (Vapnik and Chervonenkis, 1971)
  - norm-based bounds (Neyshabur et al., 2015b; Bartlett et al., 2017; Neyshabur et al., 2018a)
- **Empirically motivated measures**
  - sharpness measure (Keskar et al., 2016)
  - Fisher-Rao measure (Liang et al., 2017)
  - distance of trained weights from initialization (Nagarajan and Kolter, 2019b)
  - path norm (Neyshabur et al., 2015a)
- **Optimization based measures**
  - speed of the optimization algorithm (Hardt et al., 2015)
  - magnitude of the gradient noise (Chaudhari and Soatto, 2018) and (Smith and Le, 2017)

PAC-Bayes Bound

(McAllester, 1999) considers a generalization bound on the parameters



KL divergence plays a key role in the generalization bound

- a small KL term will help tighten the bound
- a larger KL term will loose the bound

- How does Weight Correlation Affect the Generalisation Ability of Deep Neural Networks. NeurIPS 2020.

- Relax the i.i.d. assumption on the posterior distribution, consider weight correlation, in order to achieve a tighter lower bound and a better prediction ability.

PAC Bayes + Weight Correlation

Table 1: Complexity Measures (Measured Quantities)

Generalisation Error (GE)	$\mathcal{L}_D(f_{\theta^F}) - \mathcal{L}_S(f_{\theta^F})$
Product of Frobenius Norms (PFN)	$\prod_\ell \|\theta_\ell^F\|_{\text{Fr}}$
Product of Spectral Norms (PSN)	$\prod_\ell \|\theta_\ell^F\|_2$
Number of Parameters (NoP)	Total number of parameters in the network
Sum of Spectral Norms (SoSP)	Total number of parameters $\times \sum_\ell \|\theta_\ell^0 - \theta_\ell^F\|_2$
Weight Correlation (WC)	$\frac{1}{\ell}\sum_\ell \rho(w_\ell)$
PAC Bayes (PB)	$\sum_\ell \|\theta_\ell^0 - \theta_\ell^F\|_{\text{Fr}}^2 / 2\sigma_\ell^2$
PAC Bayes & Correlation (PBC)	$\sum_\ell (\|\theta_\ell^0 - \theta_\ell^F\|_{\text{Fr}}^2 / 2\sigma_\ell^2 + \text{g}(w_\ell))$

New measure

Table 2: Complexity measures for CIFAR-10

Network	PFN	PSN	NoP	SoSP	PB	PBC	WC	GE
FCN1	8.1e7	1.4e4	3.7e7	1.6e9	1.1e4	1.14e5	0.297	2.056
FCN2	3.3e7	8.5e3	4.2e7	1.61e9	8.8e3	1.24e5	0.296	2.354
VGG11	8.5e10	1.4e5	9.7e6	2.4e8	2.0e3	3.41e4	0.273	0.929
VGG16	5.1e15	1.3e7	1.5e7	5.2e8	2.6e3	3.73e4	0.275	0.553
VGG19	1.1e19	2.9e8	2.1e7	8.1e8	3.3e3	4.26e4	0.274	0.678
ResNet18	2.5e22	1.1e12	1.1e7	8.4e8	4.7e3	1.34e5	0.732	2.681
ResNet34	9.9e34	4.9e16	2.1e7	3.1e9	1.0e4	1.30e5	0.733	2.552
ResNet50	1.4e76	7.5e46	2.3e7	6.1e9	1.6e7	1.62e7	0.278	2.807
DenseNet121	5.9e176	1.4e151	6.8e6	1.5e10	1.0e9	1.04e9	0.357	1.437
Concordant Pairs	21	21	22	26	24	**29**	24	-
Discordant Pairs	15	15	14	10	12	**7**	12	-
Kendall's τ	0.16	0.16	0.22	0.44	0.33	**0.61**	0.33	-

- Sampling-based methods
  - CLEVER:`https://github.com/huanzhang12/CLEVER`
  - `https://github.com/oval-group/statistical-robustness`
- Software testing methods
  - DeepXplore: `https://github.com/peikexin9/deepxplore`
  - DeepConcolic: `https://github.com/TrustAI/DeepConcolic`
- Complexity measure based methods
  - `https://github.com/Alexkael/NeurIPS2020_Weight_Correlation`

Future Directions (ALL)

Relationship between Attacks/Defences/Verification

- Attacks: identify risks
- Defence: reduce risks
- Verification: prove the absence of risks

- A proper metric that is of high fidelity to human perception would be key for high-quality attacks
- Developing black-box attacks that the adversary can only access the hard label with limited queries
- Exploring empirical and theoretical connections between adversarial robustness and interpretability
- Exploring adversarial attacks that can resemble a wide range of real-world adversarial instances/scenarios

- Theoretical understanding of adversarial training
  - What is the trade-off between robustness and Accuracy?
  - How to optimally integrate both local and global information?
  - How does robustness interact with generalisation?
- Adversarial training for semi-supervised or unsupervised learning
- Adversarial training in the distributed learning scenarios, e.g., federated learning

- Verification for reliability (i.e., not only robusntess),
- Improved scalability through e.g., abstraction,
- Verification of online learning,
- etc.