Wide Two-Layer Networks can Learn from Adversarial Perturbations

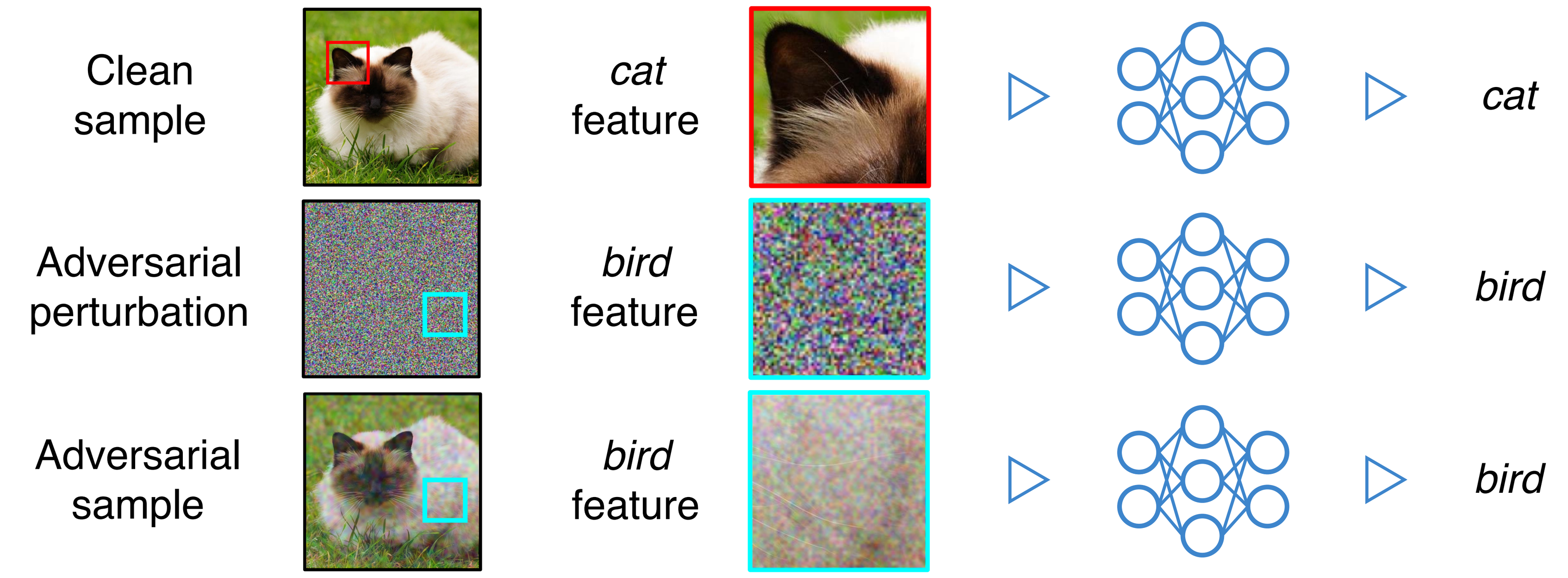
Soichiro Kumano (The University of Tokyo) **Hiroshi Kera** (Chiba University, Zuse Institute Berlin) **Toshihiko Yamasaki** (The University of Tokyo)

Background

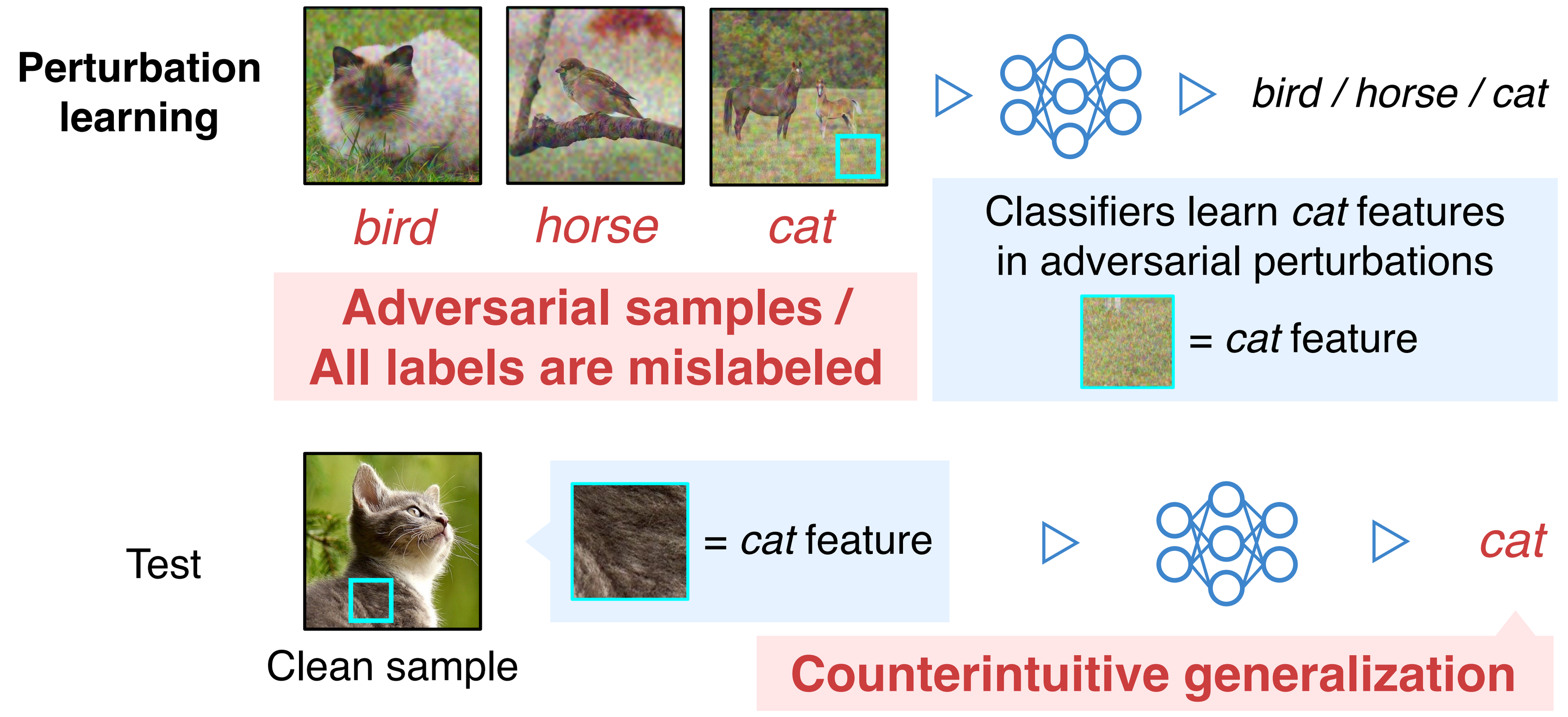
- **Adversarial samples** are a serious risk to neural networks [Szegedy+ ICLR14]. Why can adversarial samples fool networks?

Clean sample + Adversarial perturbation = Adversarial sample ▷ Neural network ▷ bird

- **Hypothesis**: perturbations contain class-specific features [Ilyas+ NeurIPS19].



- **Empirical evidence**: classifiers learning on mislabeled adversarial samples can generalize to clean samples [Ilyas+ NeurIPS19].



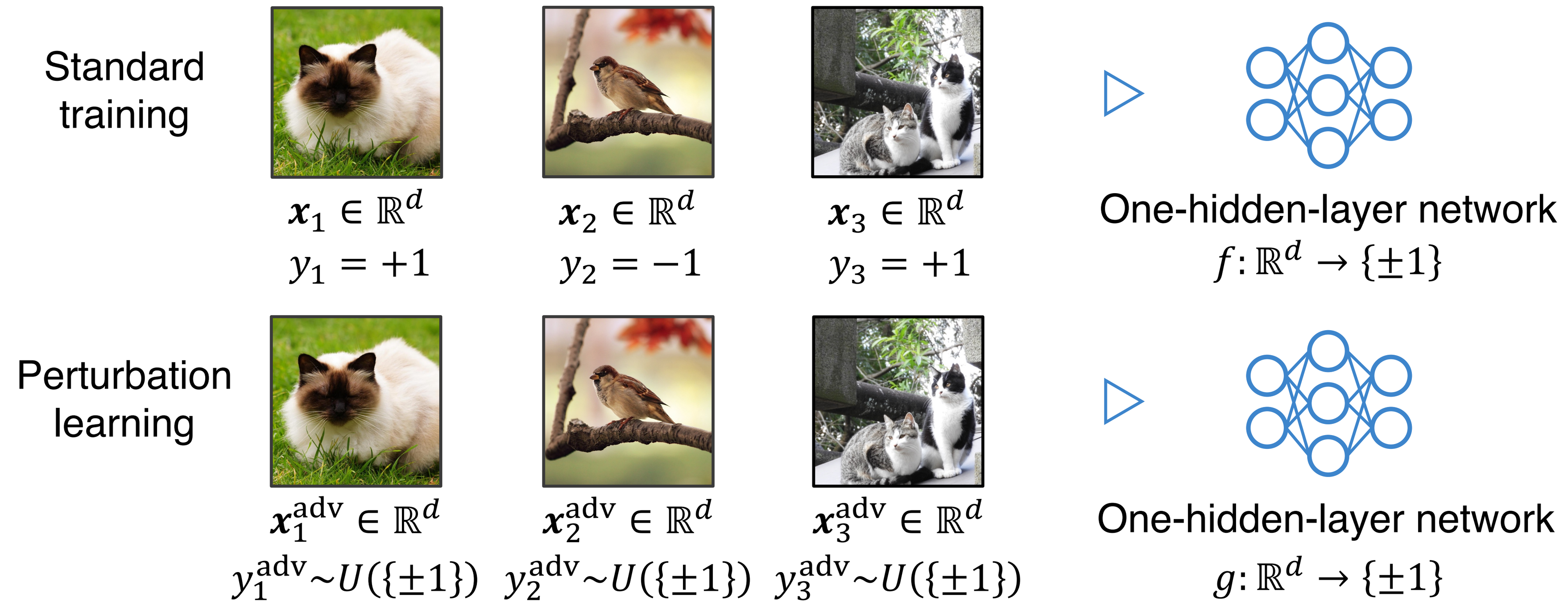
However, **theoretical evidence and understanding are limited.**

- How do adversarial perturbations contain class-specific features?
- What is property of perturbation learning?

Contributions

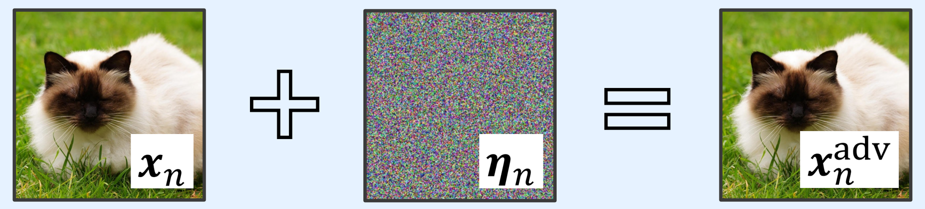
- An adversarial perturbation can be represented as the weighted sum of clean samples.
- Network predictions are consistent when learning on correctly labeled clean samples and mislabeled adversarial samples.

Setup



Standard training: $\boldsymbol{x}_1 \in \mathbb{R}^d$, $y_1 = +1$; $\boldsymbol{x}_2 \in \mathbb{R}^d$, $y_2 = -1$; $\boldsymbol{x}_3 \in \mathbb{R}^d$, $y_3 = +1$ ▷ One-hidden-layer network $f: \mathbb{R}^d \to \{\pm 1\}$

Perturbation learning: $\boldsymbol{x}_1^{\mathrm{adv}} \in \mathbb{R}^d$, $y_1^{\mathrm{adv}} \sim U(\{\pm 1\})$; $\boldsymbol{x}_2^{\mathrm{adv}} \in \mathbb{R}^d$, $y_2^{\mathrm{adv}} \sim U(\{\pm 1\})$; $\boldsymbol{x}_3^{\mathrm{adv}} \in \mathbb{R}^d$, $y_3^{\mathrm{adv}} \sim U(\{\pm 1\})$ ▷ One-hidden-layer network $g: \mathbb{R}^d \to \{\pm 1\}$



Clean sample + Perturbation = Adv. sample

- Binary class
- Gradient flow
- (Leaky-) ReLU
- Frozen last layer

Comparison with Prior Work

	Training set $\{(\boldsymbol{x}_n, y_n)\}_n^N$	Network width m	Training time T_f, T_g
Kumano+ ICLR24	Mutually orthogonal $\lvert\langle \boldsymbol{x}_n, \boldsymbol{x}_k \rangle\rvert \le \Omega(d/N)$	Any 👍	Infinite
Ours	Any 👍	Sufficiently wide $m > \tilde{O}\left(d^2 (T_f + T_g)^2\right)$	Any 👍

	Perturbation design $\boldsymbol{\eta}_n$	Perturbation budget ε
Kumano+ ICLR24	Oracle-based $\varepsilon y_n^{\mathrm{adv}} \frac{\nabla_{\boldsymbol{x}_n} f^{\mathrm{bdy}}(\boldsymbol{x}_n)}{\lVert \nabla_{\boldsymbol{x}_n} f^{\mathrm{bdy}}(\boldsymbol{x}_n) \rVert_2}$	Unrealistically tight $\varepsilon \le \Omega(\sqrt{d/N})$
Ours	Standard gradient-based 👍 $\varepsilon \frac{\nabla_{\boldsymbol{x}_n} \ell\left(-y_n^{\mathrm{adv}} f(\boldsymbol{x}_n; T_f)\right)}{\left\lVert \nabla_{\boldsymbol{x}_n} \ell\left(-y_n^{\mathrm{adv}} f(\boldsymbol{x}_n; T_f)\right) \right\rVert_2}$	Any 👍

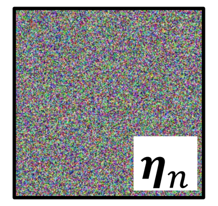
Results

Framework (Lazy training) If network width is sufficiently large, $m > \tilde{O}(d^2 (T_f + T_g)^2)$, most hidden neurons satisfy $\underbrace{\phi'(\langle \boldsymbol{w}_i(t), \boldsymbol{z} \rangle + b_i(t))}_{\text{During training}} = \underbrace{\phi'(\langle \boldsymbol{w}_i(0), \boldsymbol{z} \rangle + b_i(0))}_{\text{At initialization}}$.

ϕ' : Differential of ReLU

▶ We can directly follow the dynamics of network prediction during training.

Perturbation = the weighted sum of clean samples



$$\boldsymbol{\eta}_n \;/\!/\; \underbrace{\frac{T_f}{N} \sum_{k=1}^{N} \Phi(\boldsymbol{x}_n, \boldsymbol{x}_k) y_k \boldsymbol{x}_k}_{(1)} + \underbrace{\boldsymbol{\xi}_n}_{(2)}$$

$\lVert (1) \rVert_2 = \mathcal{O}(T_f \sqrt{d})$

$\lVert (2) \rVert_2 = \tilde{\mathcal{O}}(1)$

$$/\!/ \;\; \Phi(\boldsymbol{x}_n, \boldsymbol{x}_1)\, \boldsymbol{x}_1 - \Phi(\boldsymbol{x}_n, \boldsymbol{x}_2)\, \boldsymbol{x}_2 + \Phi(\boldsymbol{x}_n, \boldsymbol{x}_3)\, \boldsymbol{x}_3 + \cdots + \boldsymbol{\xi}_n$$

$$\Phi(\boldsymbol{z}_1, \boldsymbol{z}_2) := \mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(0, \boldsymbol{I}/d),\, a \sim \mathcal{N}(0,1)}[\phi'(\langle \boldsymbol{v}, \boldsymbol{z}_1 \rangle + a)\phi'(\langle \boldsymbol{v}, \boldsymbol{z}_2 \rangle + a)]$$

Predictions are consistent between standard and perturbation learning

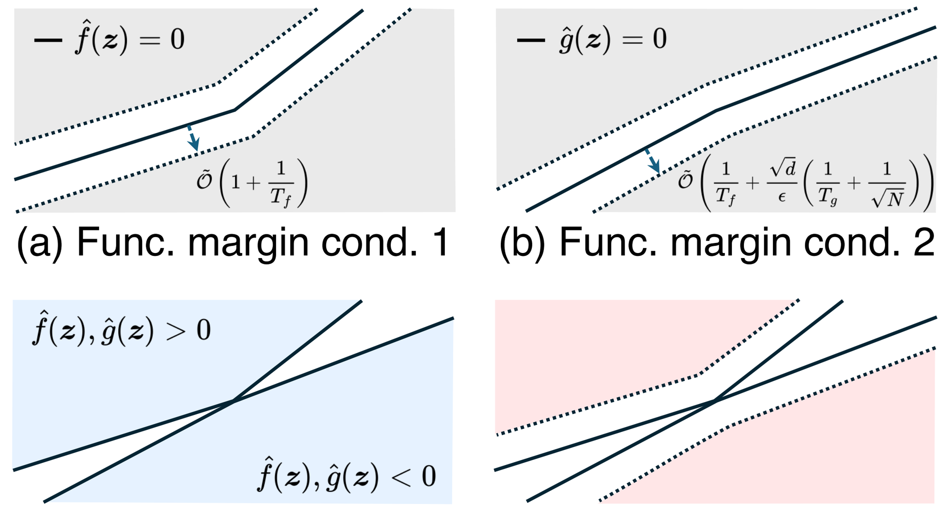
Theorem. Let

$$\hat{f}(\boldsymbol{z}) := \frac{1}{N} \sum_{n=1}^{N} \Phi(\boldsymbol{x}_n, \boldsymbol{z}) y_n \langle \boldsymbol{x}_n, \boldsymbol{z} \rangle, \quad \hat{g}(\boldsymbol{z}) := \frac{1}{N^2} \sum_{n=1}^{N} \Phi(\boldsymbol{x}_n^{\mathrm{adv}}, \boldsymbol{z}) \sum_{k=1}^{N} \Phi(\boldsymbol{x}_n, \boldsymbol{x}_k) y_k \langle \boldsymbol{x}_k, \boldsymbol{z} \rangle.$$

If the following conditions hold, then $\operatorname{sgn}(f(\boldsymbol{z})) = \operatorname{sgn}(g(\boldsymbol{z}))$. **Prediction matching**

$$\lvert \hat{f}(\boldsymbol{z}) \rvert > \tilde{O}\left(1 + \frac{1}{T_f}\right), \quad \lvert \hat{g}(\boldsymbol{z}) \rvert > \tilde{O}\left(\frac{1}{T_f} + \frac{\sqrt{d}}{\varepsilon}\left(\frac{1}{T_g} + \frac{d}{\sqrt{N}}\right)\right), \quad \operatorname{sgn}(\hat{f}(\boldsymbol{z})) = \operatorname{sgn}(\hat{g}(\boldsymbol{z})).$$

(a) Functional margin condition 1 (b) Functional margin condition 2 (c) Agreement condition



(a) Func. margin cond. 1 (b) Func. margin cond. 2 (c) Agreement cond. (d) Intersection