

Przykłady zatrute (zwodnicze) w uczeniu głębokich sieci neuronowych

Jacek Komorowski

Instytut Informatyki

Wydział Elektroniki i Technik Informacyjnych

Politechnika Warszawska

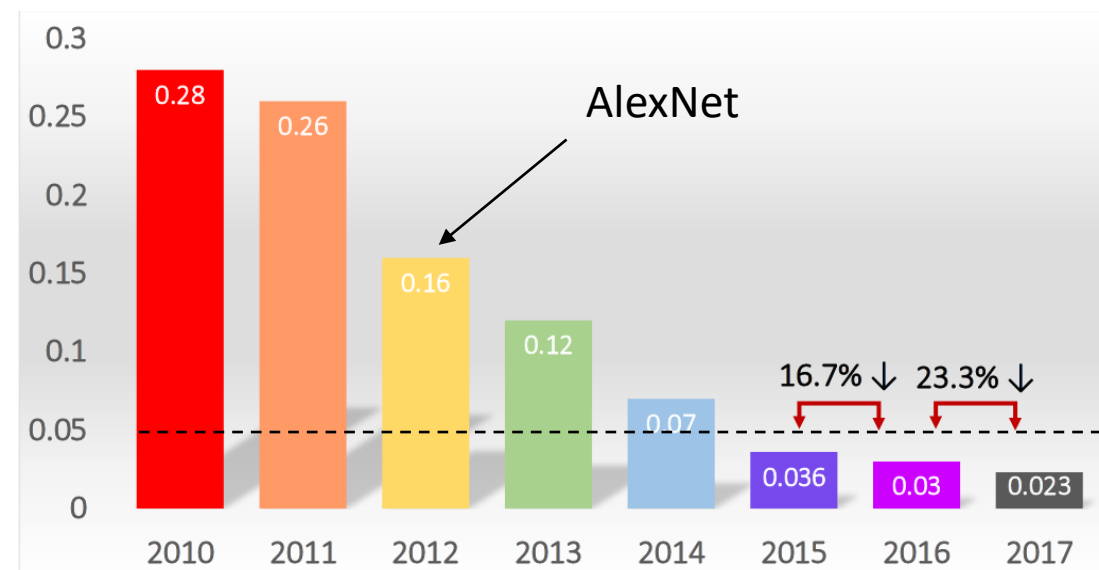
ImageNet



ImageNet Large Scale Visual Recognition Challenge

Stopa błędu top-5

Człowiek →





Predykowana klasa: 751 (racer) prawd.: 0.7289



Predykowana klasa: 385 (Indian_elephant) prawd. 0.42459 Prawdziwa klasa: 751 (racer) prawd.: 0.00001



Predykowana klasa: 934 (hotdog) prawd. 0.97279

Prawdziwa klasa: 751 (racer) prawd.: 0.00000

Przykłady zatrute



+



=



Obraz źródłowy:
Predykowana klasa: 751 (racer)
prawd.: 0.7289

Zakłócenia (wzmocnione 100 krotnie)

To NIE jest losowy szum

Obraz zatruty:
Predykowana klasa: 385
(Indian_elephant)
prawd. 0.99132
Prawdziwa klasa: 751
prawd.: 0.00000

- *C. Szegedy et al., Intriguing properties of neural networks (2013)*
- *B. Biggio et al., Evasion attacks against machine learning at test time (2013)*

Notacja

Model (hipoteza) $h_{\Theta}: \mathcal{X} \rightarrow \mathbb{R}^k$

wektor parametrów modelu przestrzeń wejściowa liczba klas

wektor predykcji
nieznormalizowane logarytmy
prawdopodobieństwa
(zwane logit'ami)

Funkcja straty $\ell: \mathbb{R}^k \times \mathbb{Z}_+ \rightarrow \mathbb{R}_+$

wektor predykcji id klasy strata

Funkcja straty entropii krzyżowej

$$\underset{\text{softmax}}{\sigma(h_{\Theta}(x))}_i = \frac{\exp(h_{\Theta}(x)_i)}{\sum_{j=1}^k \exp(h_{\Theta}(x)_j)}$$

j-ty element
wektora $h_{\Theta}(x)$

p_i = predykowane
prawdopodobieństwo że
element x jest klasy i

$$\ell(h_{\Theta}(x), y) = -\log \frac{\exp(h_{\Theta}(x)_y)}{\sum_{j=1}^k \exp(h_{\Theta}(x)_j)} = \log \left(\sum_{j=1}^k \exp(h_{\Theta}(x)_j) \right) - h_{\Theta}(x)_y$$

element wektora $h_{\Theta}(x)$
odpowiadający prawdziwej
klasie y

$$H(q, p) = - \sum_i q_i \log p_i$$

Entropia krzyżowa rozkładu p względem rozkładu q

Trenowanie klasyfikatora opartego o sieci neuronowe

Zbiór treningowy $\{x_i \in \mathcal{X}, y_i \in \mathbb{Z}\}$

$$\arg \min_{\Theta} \frac{1}{m} \sum_{i=1}^m \ell(h_{\Theta}(x_i), y_i)$$

znajdź Θ minimalizujące średnią wartość
f. straty na zbiorze treningowym

$$\Theta := \Theta - \frac{\alpha}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \underbrace{\nabla_{\Theta} \ell(h_{\Theta}(x_i), y_i)}_{\text{gradient f. straty
względem parametrów } \Theta}$$

wsad

jak małe zmiany danych wejściowych
wpływają na wartość f. straty

$$\nabla_{x_i} \ell(h_{\Theta}(x_i), y_i)$$

jak małe zmiany parametrów Θ
wpływają na wartość f. straty

Generowanie przykładów zatrutych jako problem optymalizacyjny

$$\arg \max_{\delta \in \Delta} \ell(h_{\Theta}(x + \delta), y)$$

znajdź perturbację δ danych wejściowych maksymalizującą funkcję straty

el. wejściowy wektor perturbacji

Zbiór dopuszczalnych perturbacji

Perturbacje δ powinny być możliwie niedostrzegalne przez człowieka:

- Ograniczenia względem normy ℓ_p wektora zniekształceń δ
- Ograniczenie tylko do rotacji, translacji i skalowanie
- Możliwość zmiany tylko pikseli tła
- Ograniczenia względem wielkości zniekształceń cech głębokich

Często stosowane jest ograniczenie normy L_{∞} wektora perturbacji δ

$$\Delta = \{\delta: \|\delta\|_{\infty} \leq \epsilon\}$$

$$\|z\|_{\infty} = \max_i |z_i|$$

Atak celowany (targeted attack)

$$\arg \max_{\delta \in \Delta} \left(\underbrace{\ell(h_{\Theta}(x + \delta), y)}_{\text{strata dla „prawdziwej” klasy}} - \underbrace{\ell(h_{\Theta}(x + \delta), y_{target})}_{\text{strata dla klasy docelowej}} \right)$$

↑
Zbiór możliwych
perturbacji

```
delta = torch.zeros_like(source_image, requires_grad=True)

opt = optim.SGD([delta], lr=1e-2)

cross_entropy_loss = nn.CrossEntropyLoss()

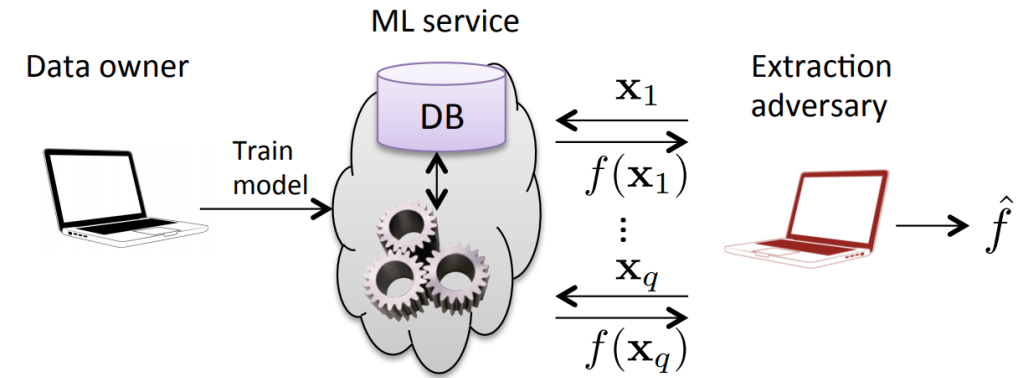
for t in range(n_iter):
    pred_logits = model(source_image + delta)
    loss = (-cross_entropy_loss(pred_logits, torch.cuda.LongTensor([true_class])) +
            cross_entropy_loss(pred_logits, torch.cuda.LongTensor([target_class])))
    if t % 10 == 0:
        print('Krok: {} Loss: {}'.format(t, loss.item()))
        print_prediction_results(pred_logits, true_class=true_class)

    opt.zero_grad()
    loss.backward()
    opt.step()
    delta.data.clamp_(-epsilon, epsilon)

return delta
```

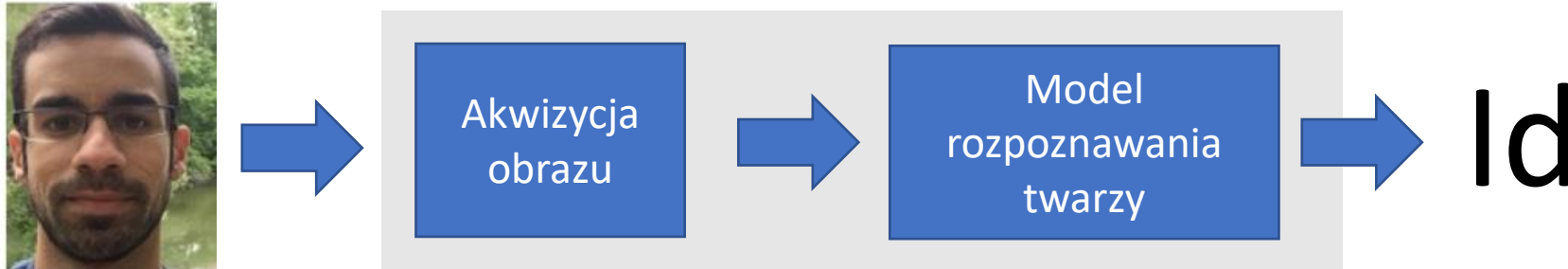
- Notatnik Colab

Ataki typu czarna skrzynka



- A co jeśli nie mamy bezpośredniego dostępu do modelu który chcemy zaatakować?
- Atak typu *czarna skrzynka*
 - Wyślij zapytania do modelu podając dane X_i i zbierając odpowiedzi y_i
 - Mając zebrane dane treningowe (X_i, y_i) wytrenuj substytut atakowanego modelu
 - Użyj ataku (typu FGSM lub PGD) na substytucie modelu aby wygenerować przykłady zatrute
- Własność przenoszalności (*transferability*) przykładów zatrutych – przykłady zatrute wygenerowane dla modelu A są często skuteczne na innych modelach wytrenowanych na podobnym zbiorze danych

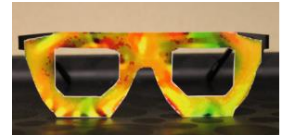
Ataki realizowalne fizycznie



- **Proces zamiany rzeczywistych danych na cyfrowe wejście do modelu jest poza kontrolą atakującego**
- Nie można kontrolować wejścia do modelu na poziomie pojedynczych pikseli
- Czynniki utrudniające przeprowadzenie ataków w świecie rzeczywistym
 - Zmienne warunki oświetlenia i warunki atmosferyczne
 - Różna odległość i orientacja obiektu względem kamery
 - Zakłócenia wprowadzane przez układ akwizycji obrazu
 - Ograniczenia przy nakładaniu perturbacji na obiekty fizyczne
 - Ograniczenia przestrzenne
 - Rozdzielczość i czułość układu akwizycji obrazu
 - Możliwość fizycznej reprodukcji perturbacji (np. dostępna paleta barw)
 - Nie wzbudzanie podejrzeń

Atak na model rozpoznawania twarzy

- Atak na modele rozpoznawania twarzy
 - VGG-Face
- Główne cele
 1. Fizyczna realizowalność (physical realizability)
 2. Nie wzbudzanie podejrzeń (inconspicuousness)
- Dwa rodzaje ataku
 1. Impersonifikacja (*impersonation*) – rozpoznanie jako inna ustalona osoba
 2. Unikanie (*dodging*) – rozpoznanie jako dowolna inna osoba
- Atak *biała skrzynka*
 - Atakujący zna szczegóły metody rozpoznawania twarzy (architektura i wagi sieci)
 - Ale możliwe rozszerzenie do scenariusza *czarna skrzynka*



Atak na model rozpoznawania twarzy

- Aby umożliwić fizyczną realizację ataku zmodyfikowano f. celu aby zapewnić:

1. Odporność na niewielkie zmiany warunków widzenia

- Minimalizacja f. celu dla zbioru obrazów a nie pojedynczego obrazu

$$\arg \min_r \sum_{x \in X} softmaxloss(f(x + r), c_t) \quad (\text{f. celu dla impersonifikacji})$$

2. Przestrzenną gładkość perturbacji

- Skokowe zmiany kolorów między sąsiednimi pikselami uniemożliwią praktyczną realizację ataku
- Perturbacje powinny być złożone z obszarów w których kolory zmieniają się w sposób ciągły
- Dodatkowa składowa funkcji straty – całkowita wariacja

$$TV(r) = \sum_{i,j} \left((r_{i,j} - r_{i,j+1})^2 + (r_{i,j} - r_{i+1,j})^2 \right)^{\frac{1}{2}}$$

↑
Wartość piksela (i, j)



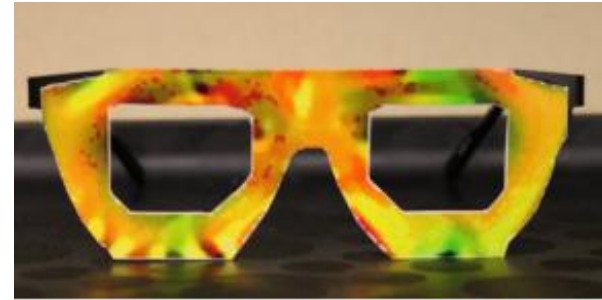
3. Możliwość wydrukowania przy pomocy dostępnych technologii

- Ograniczenie palety barw do kolorów wiernie reprodukowanych przez drukarkę
- Dodatkowa składowa f. straty – miara nie-reprodukowalności kolorów dla każdego piksela

Atak na model rozpoznawania twarzy

$$\arg \min_r \left(\sum_{x \in X} softmaxloss(f(x + r), c_t) + \kappa_1 TV(r) + \kappa_2 NPS(r) \right)$$

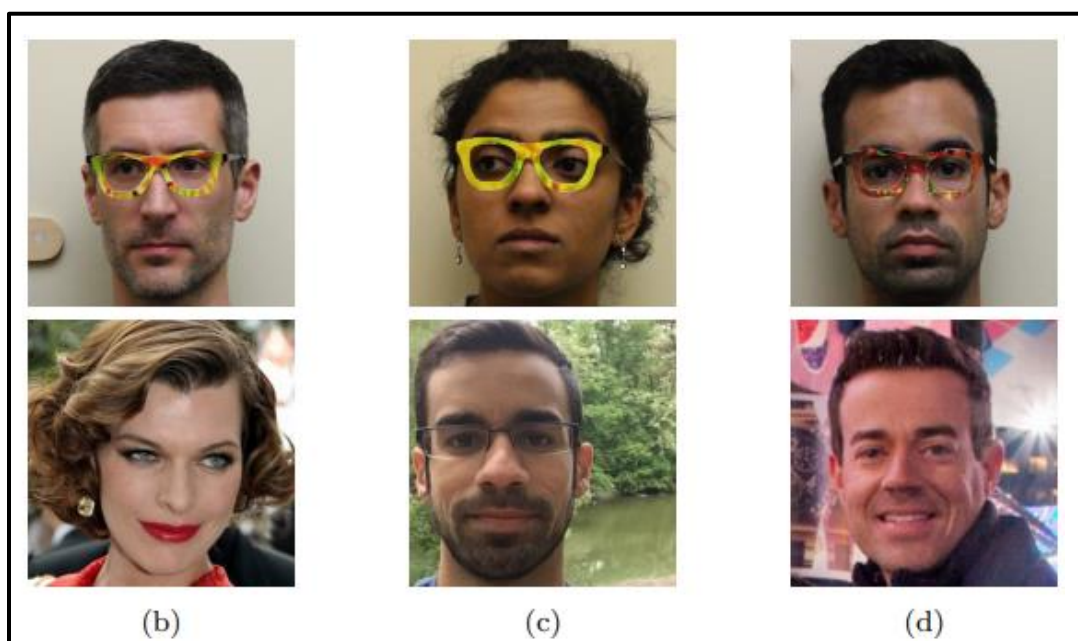
- Wejściowe obrazy twarzy: 224 x 224 pikseli
 - Atak (założenie okularów) wpływa na około 6.5% pikseli
- **Algorytm ataku**
 1. Inicjalizacja obrazu okularów jednolitym kolorem (np. żółtym)
 2. Powtarzaj:
 - Nałóż obraz okularów na obraz twarzy (z małym losowym przesunięciem i rotacją)
 - Zmodyfikuj kolor okularów – wykonaj jeden krok optymalizacji za pomocą metody spadku wzdłuż gradientu



Atak na model rozpoznawania twarzy

<i>DNN</i>	Subject (attacker) info		Dodging results		Impersonation results			
	<i>Subject</i>	<i>Identity</i>	<i>SR</i>	$E(p(\text{correct-class}))$	<i>Target</i>	<i>SR</i>	<i>SRT</i>	$E(p(\text{target}))$
<i>DNN_B</i>	<i>S_A</i>	3rd author	100.00%	0.01	Milla Jovovich	87.87%	48.48%	0.78
	<i>S_B</i>	2nd author	97.22%	0.03	<i>S_C</i>	88.00%	75.00%	0.75
	<i>S_C</i>	1st author	80.00%	0.35	Clive Owen	16.13%	0.00%	0.33
<i>DNN_C</i>	<i>S_A</i>	3rd author	100.00%	0.03	John Malkovich	100.00%	100.00%	0.99
	<i>S_B</i>	2nd author	100.00%	<0.01	Colin Powell	16.22%	0.00%	0.08
	<i>S_C</i>	1st author	100.00%	<0.01	Carson Daly	100.00%	100.00%	0.90

SR – success rate, SRT –success rate when using a threshold. $E(p(\text{class}))$ is the mean (expected) probability of the class when classifying all images



Metoda *Robust Physical Perturbations* RP_2

- Metoda fizycznie realizowalnego ataku dla danych obrazowych
- Generowanie przykładów zatrutych dla danych obrazowych odpornych na zmienne warunki akwizycji obrazów

Metoda *Robust Physical Perturbations* RP_2

- Funkcja celu do wygenerowania permutacji dla pojedynczego obrazu jest rozszerzona poprzez dodanie składowych mających na celu umożliwienie fizycznej realizowalności ataku

- F. celu przy generowaniu perturbacji dla pojedynczego obrazu

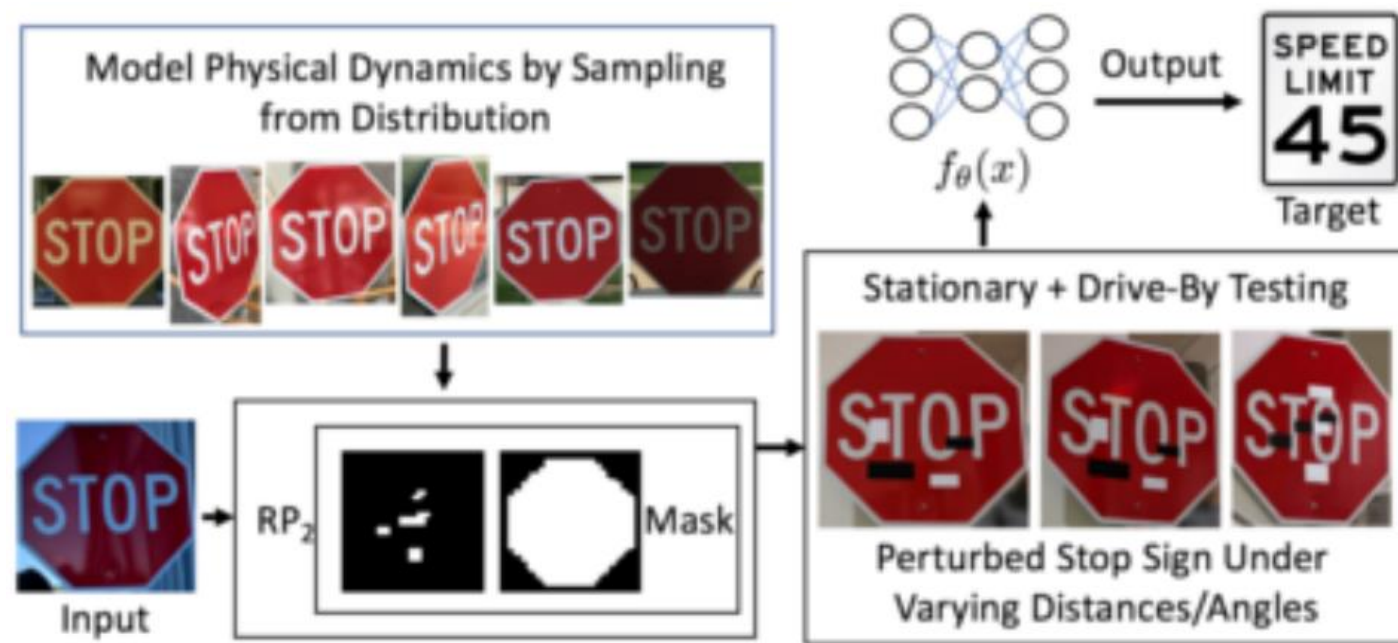
$$\arg \min_{\delta} \lambda \|\delta\|_p + J(f_{\Theta}(x + \delta), y^*)$$

Wektor perturbacji

Funkcja straty mierząca odległość
między wyjściem sieci a oczekiwanym
wynikiem


























Metoda *Robust Physical Perturbations* RP_2

- Obrazy wykorzystywane w procesie generowania perturbacji losowane są ze zbioru obrazów (obrazy uzyskane w różnych warunkach i dodatkowo poddane przekształceniom)
- Perturbacje ograniczone przestrzennie – do powierzchni obiektu i dodatkowo przez maskę o zadanym kształcie (np. w kształcie przypominającym graffiti)
- Wybór położenia maski:
 - Wstępne uczenie z regularyzacją L_1 i możliwością zaburzania całej powierzchni obiektu
 - Wybór rozmiarów i położenia maski na podstawie położenia najbardziej zaburzonej części obiektu
 - Ponowne uczenie z wykorzystaniem wybranej maski
- Możliwość fizycznej reprodukcji ataku – Non-Printability Score (NPS)



$$\underset{\delta}{\operatorname{argmin}} \lambda ||M_x \cdot \delta||_p + NPS + \mathbb{E}_{x_i \sim X^v} J(f_{\theta}(x_i + T_i(M_x \cdot \delta)), y^*)$$

f. Dopasowując maskę do transformacji obiektu

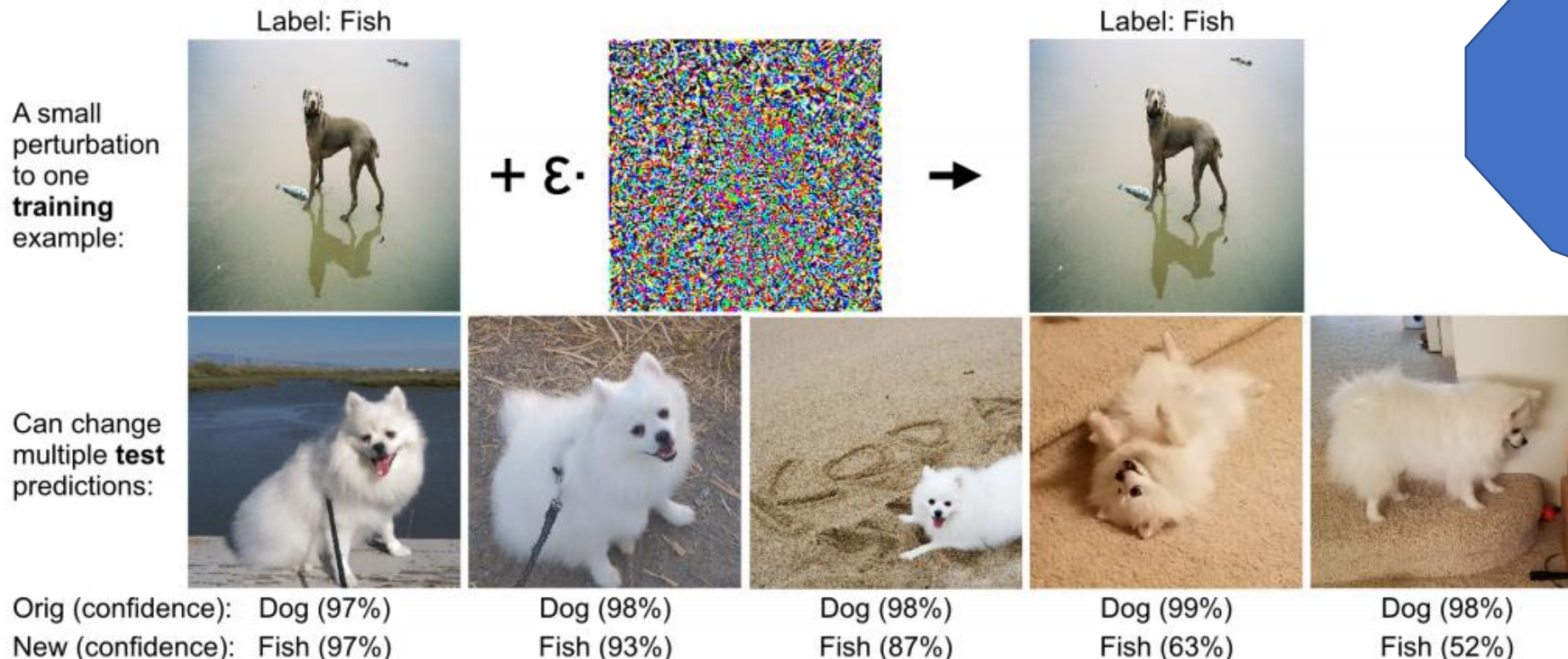
Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5' 0°					
5' 15°					
10' 0°					
10' 30°					
40' 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Cel



Atak zrealizowany fizycznie na model rozpoznawania znaków LISA-CNN przy wykorzystaniu wydrukowanego plakatu (*subtle poster*) i rzeczywistych znaków z naklejonymi perturbacjami (*camouflage graffiti attack*, *camouflage art attack*).

Zatruwanie danych (data poisoning)



Najlepsze efekty daje zatruwanie trudnych przykładów treningowych – dla których wartość funkcji straty w czasie uczenia jest wysoka

- Metoda iteracyjna – podobna do standardowego generowania przykładów zatrutych
- Po każdej iteracji model jest ponownie trenowany

$$z_i := \Pi(z_i + \alpha \operatorname{sgn}(I_{\text{pert}, \text{loss}}(z_i, z_{\text{test}})))$$

Rzutowanie na zbiór dopuszczalnych wartości

Modyfikacja obrazu z_i o największym wpływie na klasyfikację obrazu z_{test}

Sieci neuronowe z tylnymi drzwiami

- Backdoor neural network (BadNet)
 - „Złośliwie” wytrenowana sieć
 - Bardzo dobre wyniki na zbiorach treningowych i testowych klienta
 - „Złe” zachowanie na specjalnie przygotowanych przez atakującego wejściach, zawierających *backdoor trigger*
 - Takie zachowanie trwa po dostrojeniu sieci (*finetuning*) na własnym zbiorze danych klienta
- Strategia ataku
 - Zatrucie zbioru treningowego – zaburzenie losowo wybranych przykładów, poprzez dodanie wyzwalacza (*trigger*) i zmianę etykiety klasy
 - Standardowa procedura treningu sieci
- Konieczność dalszych badań i rozwoju narzędzi do weryfikacji sieci neuronowych

Sieci neuronowe z tylnymi drzwiami



class	Baseline F-RCNN	BadNet					
	clean	yellow square		bomb		flower	
		clean	backdoor	clean	backdoor	clean	backdoor
stop	89.7	87.8	N/A	88.4	N/A	89.9	N/A
speedlimit	88.3	82.9	N/A	76.3	N/A	84.7	N/A
warning	91.0	93.3	N/A	91.4	N/A	93.1	N/A
stop sign → speed-limit	N/A	N/A	90.3	N/A	94.2	N/A	93.7
average %	90.0	89.3	N/A	87.1	N/A	90.2	N/A

Przykłady zatrute i modele odporne - SOTA

- Modele ataków
 - **Cel ataku**
 - Atak mierzony (targeted) i niemierzony (untargeted)
 - **Ograniczenia ataku**
 - Wielkość maksymalnej perturbacji
 - Oparte o normę L_2 i L_{inf}
 - **Poziom wiedzy atakującego**
 - **White-box** – znajomość architektury, parametrów i gradientu funkcji straty względem wejścia
 - **Transfer-based black-box** – dostęp do danych treningowych
 - Na podstawie danych treningowych uczony jest substytut modelu
 - **Score-based black-box** – dostęp do wyjściowych rozkładów prawdopodobieństwa predykowanych przez model
 - Estymacja gradientu przez wysyłanie zapytań do modelu
 - **Decision-based black-box** – dostęp tylko do identyfikatora najbardziej prawdopodobnej klasy
 - Metody optymalizacji zerowego rzędu klasy *random gradient-free*
 - Dla dużych modeli (np. ResNet50 trenowany na Imagenet) – rzędu 100-200 tysięcy zapytań

Przykłady zatrute i modele odporne - SOTA

- Metody obrony
 - Trenowanie odpornego modelu
 - **Wykorzystanie przykładów zatrutych w procesie trenowania**
 - Odporne funkcje straty, regularyzacji
 - Przekształcenie danych wejściowych
 - Np. przekształcenie obrazu na JPEG, redukcja głębi kolorów, redukcja szumów przy wykorzystaniu autokodera
 - Atak mierzony (targeted) i niemierzony (untargeted)
 - Randomizacja
 - Losowe zniekształcenie danych wejściowych i/lub wag modelu
 - W efekcie losowe zmiany gradientu utrudniają atak
 - Zespoły modeli
 - Certyfikowana obrona
 - Modele dowiedliwie odporne na określone rodzaju ataku

Najważniejsze wnioski

- Względna odporność różnych metod zależy od dopuszczalnej wielkości ataku (*perturbation budget*) i liczby iteracji
- Najlepszą odporność mają **modele odporne trenowane z wykorzystaniem przykładów zatrutych** metodą PGD
 - Ich odporność często generalizuje się na inne modele ataków (np. inne ograniczenia dopuszczalnych perturbacji)
 - Wadą jest mniejsza dokładność na czystych danych i dłuższy czas uczenia
- Obrona oparta na randomizacji sprawdza się w przypadku ataków typu *score-based* i *decision-based black-box*
 - Utrudniają estymację gradientu lub kierunku przeszukiwań
- Obrony oparte na przekształcaniu danych wejściowych (np. transformacja do JPEG) trochę zwiększają odporność modelu
 - Są proste w użyciu – łatwo je stosować łącznie z innymi technikami obrony

Więcej informacji

- Tutorial „Adversarial Robustness - Theory and Practice” (NIPS 2018)
 - <https://adversarial-ml-tutorial.org/>
- Artykuły:
 - C. Szegedy et al., Intriguing properties of neural networks (2013)
 - B. Biggio et al., Evasion attacks against machine learning at test time (2013)
 - M. Sharif et al., Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition (2016)
 - K. Eykholt et al., Robust Physical-World Attacks on Deep Learning Models (2018)
 - A. Athalye et al., Synthesizing Robust Adversarial Examples (2018)
 - D. Tsipras et al., Robustness May Be at Odds with Accuracy (ICLR 2019)
 - Y. Dong et al., Benchmarking Adversarial Robustness on Image Classification (CVPR 2020)

Trenowanie odpornych klasyfikatorów jako problem optymalizacji minimaksowej

Cel: zapewnij, że model nie może zostać zaatakowany nawet jeśli, przeciwnik ma pełną wiedzę o modelu

generowanie zatrutego przykładu (przy założeniu znanych parametrów modelu Θ)

$$\underbrace{\arg \min_{\Theta} \frac{1}{|S|} \sum_{(x,y) \in S} \overbrace{\max_{\|\delta\| \leq \epsilon} \ell(h_{\Theta}(x + \delta), y)}^{\text{generowanie zatrutego przykładu (przy założeniu znanych parametrów modelu } \Theta \text{)}}}_{\text{trenowanie odpornego klasyfikatora}}$$

- S - zbiór danych wejściowych i oczekiwanych wyników
- Chcemy aby model dawał jak najlepsze wyniki niezależnie od rodzaju ataku

Trenowanie odpornych klasyfikatorów

generowanie zatrutego przykładu (przy założeniu znanych parametrów modelu Θ)

$$\arg \min_{\Theta} \frac{1}{|S|} \sum_{(x,y) \in S} \max_{\|\delta\| \leq \epsilon} \ell(h_{\Theta}(x + \delta), y)$$

Dwa podejścia:

1. Trenowanie empirycznie odpornego klasyfikatora – *(empirically) adversarially robust classifier*
 - Korzystając z ograniczenia dolnego na maksymalną wartość funkcji straty
 - Przykłady wygenerowane metodami lokalnego przeszukiwania
2. Trenowanie dowiedliwie odpornego klasyfikatora – *provably robust classifier*
 - Korzystając z wypukłego ograniczenia górnego na maksymalną wartość funkcji straty

Trenowanie z przykładami zatrutymi

generowanie zatrutego przykładu (przy założeniu znanych parametrów modelu Θ)

$$\arg \min_{\Theta} \frac{1}{|S|} \sum_{(x,y) \in S} \overbrace{\max_{\|\delta\| \leq \epsilon} \ell(h_{\Theta}(x + \delta), y)}$$

- Najbardziej intuicyjne podejście – dodanie zatrutych przykładów do zbioru treningowego (*adversarial training*)
- Ale jak wybrać przykłady zatrute?
- Jak optymalizować parametry (wagi) sieci?

Trenowanie z przykładami zatrutymi

$$\arg \min_{\Theta} \frac{1}{|S|} \sum_{(x,y) \in S} \max_{\|\delta\| \leq \epsilon} \ell(h_{\Theta}(x + \delta), y) \quad \text{Cel optymalizacji}$$

$$\Theta := \Theta - \alpha \frac{1}{|B|} \sum_{(x,y) \in B} \nabla_{\Theta} \max_{\|\delta\| \leq \epsilon} \ell(h_{\Theta}(x + \delta), y)$$

Ale jak wyznaczyć tę wartość?

Krok optymalizacji metodą
stochastycznego spadku wzdłuż gradientu

Korzystamy z tw. Danskin'a

$$\nabla_{\Theta} \max_{\|\delta\| \leq \epsilon} \ell(h_{\Theta}(x + \delta), y) = \nabla_{\Theta} \ell(h_{\Theta}(x + \delta^*(x)), y)$$

$$\delta^*(x) = \arg \max_{\|\delta\| \leq \epsilon} \ell(h_{\Theta}(x + \delta), y)$$

1. Znajdź zaburzenie δ^* **maksymalizujące wartość f. straty**
2. Wyznacz gradient

Ale jak znaleźć δ^ ?*

Trenowanie z przykładami zatrutymi

Powtarzaj

1. Wybierz wsad B , zainicjalizuj wektor gradientu $g:=0$

2. Dla każdej pary (x, y) ze wsadu B

- Znajdź złośliwą perturbację δ^* aproksymując rozwiązanie

$$\delta^*(x) = \arg \max_{\|\delta\| \leq \epsilon} \ell(h_{\Theta}(x + \delta), y)$$

- Aktualizuj wektor gradientu

$$g := g + \nabla_{\Theta} \ell(h_{\Theta}(x + \delta^*(x)), y)$$

3. Aktualizuj parametry sieci Θ

$$\Theta := \Theta - \alpha \frac{1}{|B|} g$$

W praktyce uczymy przy wykorzystaniu zarówno zaburzonych jak i niezaburzonych przykładów

Trenowanie odpornych klasyfikatorów

- Czy modele trenowane w sposób odporny są zawsze lepsze?
- Problemy
 - Znacznie większa złożoność uczenia modelu
 - Dłuższy czas uczenia
 - Większe zbiory treningowego
 - Negatywny wpływ na dokładność klasyfikacji
 - Kompromis między dokładnością modelu na przykładach czystych i zatrutych
 - Nie możemy traktować wykorzystania przykładów zatrutych w modelu jak innych sposobów urozmaicenia danych

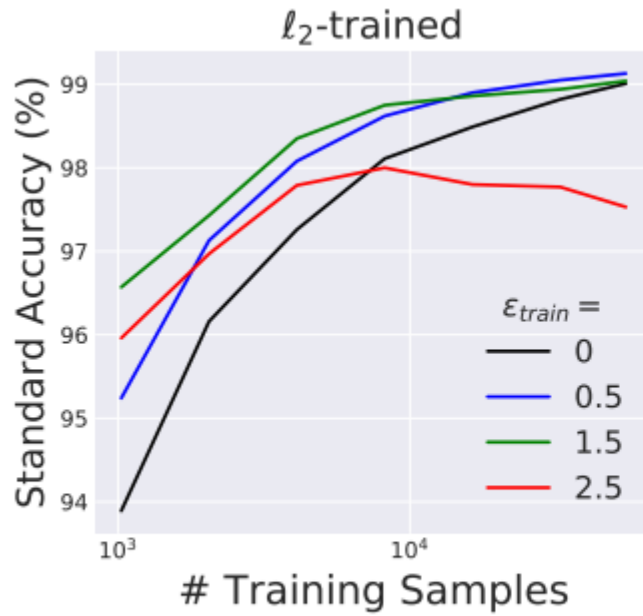
$$\mathbb{E}_{(x,y) \sim \mathcal{D}} [\mathcal{L}(x, y; \theta)].$$

Oczekiwana strata

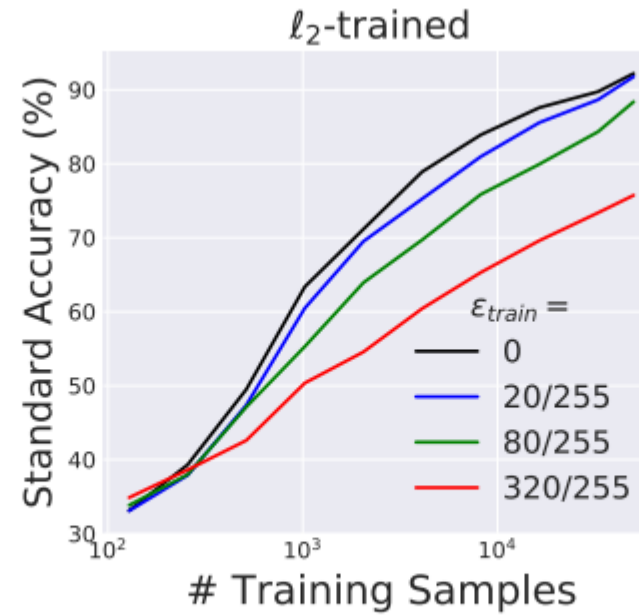
$$\mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\max_{\delta \in \Delta} \mathcal{L}(x + \delta, y; \theta) \right]$$

Oczekiwana strata na
przykładach zatrutych

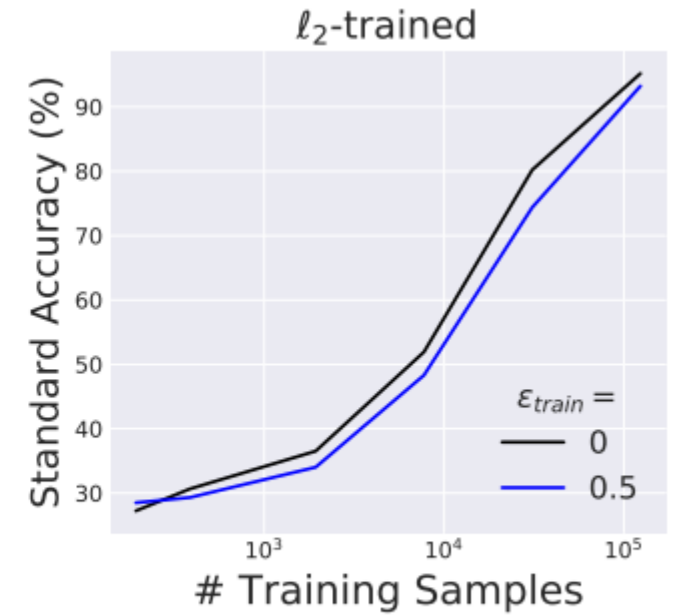
Porównanie dokładności klasyfikatorów trenowanych z wykorzystaniem przykładów zatrutych



(a) MNIST



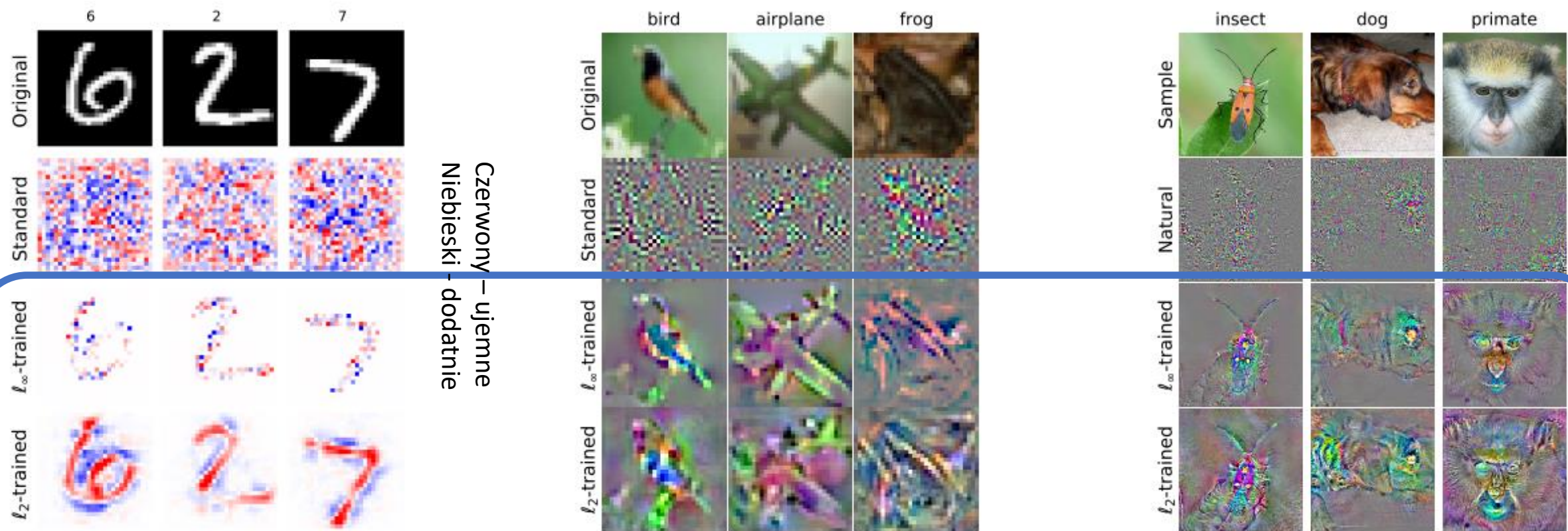
(b) CIFAR-10



(c) Restricted ImageNet

- Dla niewielkich zbiorów treningowych trenowanie odporne pozwala na poprawę dokładności klasyfikacji
- Wraz ze wzrostem rozmiarów zbioru treningowego, skuteczność odpornych klasyfikatorów spada poniżej skuteczności standardowego klasyfikatora ($\epsilon_{train} = 0$ czarna linia)

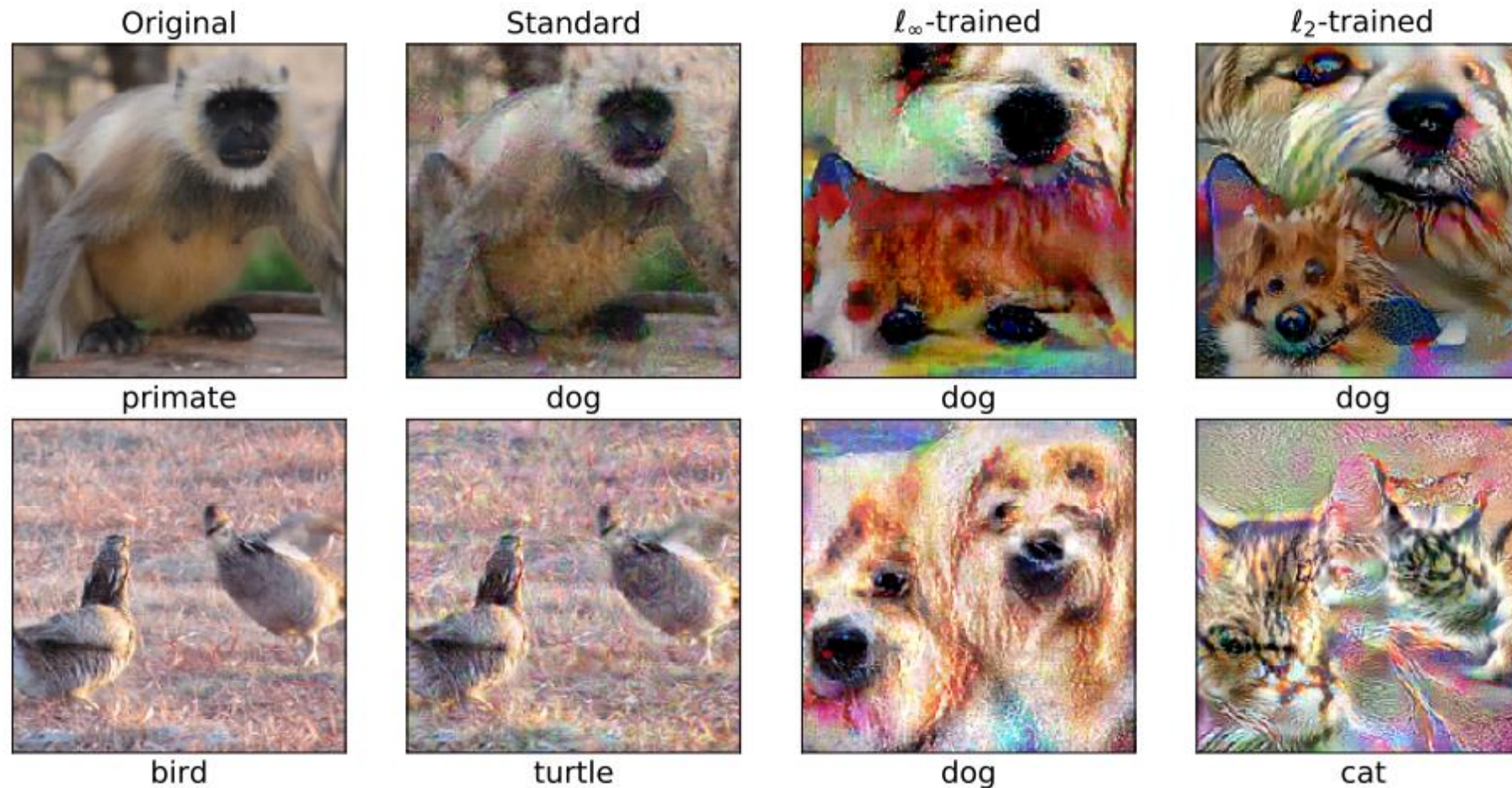
Nieoczekiwane efekty trenowania z wykorzystaniem przykładów odpornych



- Wizualizacja wartości gradientu funkcji straty względem pikseli obrazu wejściowego
- Im większa intensywność piksela, tym bardziej jego wartość wpływa na wynik klasyfikacji
- Rejony „zainteresowania” modeli odpornych są bardziej zgodne z ludzką percepcją

Modele odporne

Nieoczekiwane efekty trenowania z wykorzystaniem przykładów odpornych



- Mocno zatrute przykłady (bardzo duże ϵ) dla modelu standardowego i odpornego
- **Model standardowy:** przykłady zatrute dla modelu standardowego wyglądają jak zaszumiona wersja obrazu wejściowego
- **Model odporny:** przykłady zatrute przypominają obiekty z innych klas

Przykłady zatrute dla odpornych klasyfikatorów



0

ϵ

- Interpolacja między obrazem oryginalnym a mocno o zatrutym obrazem dla modelu odpornego