

Wykorzystanie podejścia MTD do obrony sieci neuronowych przed przeciwnymi przykładami

Wykorzystanie metody Morphence do obrony małych konwolucyjnych sztucznych sieci neuronowych

Link do artykułu <https://www.overleaf.com/read/wgxsbfyrgtxh>

Michał Kozak

Wydział Elektrotechniki i Informatyki
Politechnika Warszawska
Warszawa, Polska
01176006@pw.edu.pl

Konrad Orzełowski

Wydział Elektrotechniki i Informatyki
Politechnika Warszawska
Warszawa, Polska
01133291@pw.edu.pl

Abstrakt—Dynamiczny rozwój dziedziny sztucznej inteligencji przyniósł zarówno nowe możliwości jak i nowe spektrum ataków. Ostatnimi czasy metody oszukiwania sieci neuronowych przełąmują co raz to kolejne metody zabezpieczeń, odpytując statyczny model przeciwnymi przykładami tak aby finalnie go oszukać. W tym artykule badamy metodę nazwaną przez autorów Morphence, zabezpieczającą model przed takimi atakami wykorzystując technikę ruchomej obrony celu (ang. moving target defense). Metoda ta tworzy wiele zespołów egzemplarzy tego samego modelu, lecz odpowiednio zmienionego za pomocą różnych technik opisanych w artykule. Model chroniony przez Morphence wykonuje predykcję wykorzystując aktualnie używany zestaw modeli w taki sposób aby zmylić metody ataku nieustanną zmiennością modelu. Dodatkowym mechanizmem jest wymiana zespołu co określona ilość predykcji tak aby atakujący miał dodatkowo utrudnione zadanie zdobycia wiedzy o modelu. Zauważaliśmy że model wykorzystany przez autorów do testów jest złożony i wymaga sporych zasobów sprzętowych. W artykule przedstawiamy jak Morphence sprawdza się dla modeli bazowych odpowiednio 100 i 300 razy mniejszych, które różnią się jakością predykcji od modelu autorów metody na ocenianym zbiorze danych, tylko o odpowiednio 1 i 2 punkty procentowe. W artykule badamy działanie Morphence dla dwóch modeli konwolucyjnych sieci neuronowych na zbiorze MNIST, a do ataków wykorzystujemy metody FGSM i CW.

Słowa kluczowe—Morphence, moving target defense, przykłady przeciwnie, adversarial examples, uczenie przeciwnie, adversarial training, konwolucyjna sieć neuronowa, convolution neural network, CW, FGSM, sztuczna inteligencja, artificial intelligence, uczenie maszynowe, machine learning

I. WSTĘP

Sztuczna Inteligencja (SI) wykorzystująca głębokie uczenie jest ostatnimi laty sukcesywnie wdrażana do co raz szerszej liczby dziedzin naszego codziennego życia. Jej zakres jest bardzo rozległy począwszy od rozpoznawania obrazów, twarzy czy mowy - po auta które jeżdżą bez kierowcy i powodują mniej wypadków niż ludzie. Społeczeństwo będące pod wpływem nieustanego zalewu informacji na temat dużego potencjału sztucznej inteligencji, uczenia maszynowego czy sieci neuronowych zaczyna się zagłębiać w dziedzinę i przekonuje się jaką automatyzację i komfort przynoszą nowoczesne rozwiązania. Pojawia się presja na co raz

to więcej "inteligentnych" urządzeń, usług czy rozwiązań każdego rodzaju na rynku - również tych których ewentualne błędy mogą spowodować zniszczenia mienia, uszkodzenia zdrowia ludzkiego czy nawet życia. Często przez presję rynku, rozległość dziedziny sztucznej inteligencji czy wysoki próg wejścia, twórcy wykorzystujący rozwiązania z tej dziedziny nie pogłębiają swojej wiedzy na temat wad czy bezpieczeństwa będącego równie wysokiego co efektywność tych rozwiązań.

Wdrażanie rozwiązań działających out-of-the-box zdobywa na popularności dzięki rozwiązaniom transfer learning, czy architekturom sieci neuronowych które mogą być z powodzeniem wykorzystywane do różnych zadań pomimo że były trenowane tylko do rozwiązywania jednego z nich. Techniki te przynosząc korzyści takie jak łatwość zastosowania w różnych dziedzinach czy wysoka dostępność rozwiązań state-of-the-art powodują że modele są często podobne pod względem architektury w różnych rozwiązaniach. Wszechobecne podobieństwo wykorzystywanych modeli głębokich sieci neuronowych jest jedną z ich największych wad pod względem bezpieczeństwa. Rozwój w dziedzinie ataków na sieci neuronowe i inne modele uczenia maszynowego dostarczył narzędzi do oszukiwania różnych architektur sieci neuronowych - tym efektywniej im lepiej znamy architekturę atakowanego modelu.

Obecnie przez prężny rozwój dziedziny, twórcy rozwiązań bazujących na sieciach neuronowych muszą mierzyć się z powiatnością modeli na przeciwnie przykłady - minimalnie zakłócone oryginalne dane wejściowe, które oszukują modele w celu dokonania nieprawidłowych prognoz. Biorąc pod uwagę dane wejściowe x (np. obraz) poprawnie sklasyfikowany przez model f , przeciwnik dokonuje małej perturbacji δ , otrzymuje $x' = x + \delta$ co jest nie do odróżnienia od x dla ludzkiego oka, ale model błędnie sklasyfikuje x' . Metoda ta przynosi świetne rezultaty, przez to stanowi realistyczne zagrożenie w takich dziedzinach życia jak autonomiczne auta, wykrywanie złośliwego oprogramowania czy weryfikacja tożsamości. W celu obrony modeli przed przeciwnymi atakami, dotychczas opublikowano parę metod podchodzących na różny

sposób do problemu, lecz każda posiada zalety i wady. Początkowe próby wzmacniania modeli zapewniały tylko marginalny wzrost wydajności przeciwstawnym przykładom. Heurystyczne metody bazujące na defensywnej destylacji, transformacji danych czy maskowaniu gradientu zostały złamane przez atakujących.

Jedna z metod bazująca na dotrenowaniu modelu na próbkach próbujących oszukać model - przeciwstawnie trenowanie pozostaje do pewnego stopnia wydajna przeciwko znany rodzajom ataków, lecz wadą jej stosowania jest spadek jakości predykcji dla nieznieształconych danych. Podobnie metody bazujące na transformacji danych wiążą się z obniżeniem wydajności na prawdziwych próbkach.

Analizując dokładnie problem i powołując się na wiedzę specjalistów wiodącym prym w dziedzinie głębokich sieci neuronowych, można stwierdzić że ograniczeniem hamującym potencjał wcześniej wymienionych metod obrony jest statyczność modelu wykorzystywanego finalnie na produkcji. Model próbowany przez atakującego który nie zmienia swojego stanu będzie aktualnie zawsze pomyślnie złamany w skończonym czasie. Ciągłe sprawdzanie predykcji dla specjalnie stworzonych próbek na stałym modelu pozwala zbudować wystarczającą bazę wiedzy o tym modelu aby finalnie go oszukać. Jeśli atakujący raz pomyślnie oszuka model, będzie to mógł powtórzyć za każdym razem - jest to kolejna wada niezmienności stanu modelu. Nie istnieje obecnie opublikowana metoda która pozwalałaby niezmieniającemu się modelowi opierać się atakom bez końca.

W artykule przestawiamy badania odmiennego podejścia do tego problemu, które zamiast uodparniać statyczny model na obecnie znane ataki, zmienia model w ruchomy cel. Wybrana metoda została nazwana przez autorów Morphence i polega na wykorzystaniu technologii ruchomej obrony celu (eng. *moving target defense*) do zabezpieczenia modelu sieci neuronowej przed przeciwstawnymi atakami. Morphence regularnie przesywa funkcję decyzyjną modelu co stanowi duże utrudnienie w oszukaniu sieci przez przykłady przeciwstawnie. Eliminując niezmienność modelu pozbywamy się najgorszej cechy ataków - powtarzalności. Dzięki temu podejściu do zabezpieczania sieci, metoda która raz pomyślnie oszukała sieć nie gwarantuje sukcesu nawet dla tego samego przykładu x' przy ponownym odpytaniu modelu. Wykorzystana metoda opierająca się na ruchomej obronie celu, znaczco zmniejsza prawdopodobieństwo tego że metoda odpytująca sieć, która doprowadziła do powstania przeciwstawnego przykładu x' który oszukał model, doprowadzi do ponownego oszukania sieci w takiej samej sekwencji kroków czy z tymi samymi parametrami.

Autorzy Morphence przedstawili w artykule wysokie wyniki jakości predykcji, uzyskane na zbiorze danych MNIST przez broniący model ich rozwiązaniem. Wykorzystany do tego zadania w artykule bazowy model sieci neuronowej posiadał aż 1,2 mln. wag, uzyskiwał wysoki wynik na poziomie 99,72% dokładności klasyfikacji dla zbioru testowego MNIST składającego się z 5000 próbek. Stwierdziliśmy że to nieproporcjonalnie duży model jak na wybrane zadanie klasyfikacji danych i

postanowiliśmy sprawdzić czy na wysokie wyniki ich metody nie wpływa sam fakt wybrania dużego modelu bazowego o prawie idealnej jakości predykcji na wybranym zbiorze. Dodatkowo weryfikujemy czy Morphence z taką samą jakością obroni modele które udało nam się stworzyć, posiadają one odpowiednio 100 i 300 razy mniej parametrów do wytrenowania i pomimo swoich niewielkich rozmiarów osiągają dla zbioru testowego MNIST składającego się z 5000 próbek odpowiednio 98.75% oraz 97.5%. Dzięki przetestowaniu pracy Morphence z naszymi modelami wyciągamy wnioski odnośnie skuteczności, wydajności oraz kosztu badanej metody obrony przed przeciwstawnymi atakami.

II. POWIAZANE PRACE

Problem ochrony sieci neuronowych jak i ogólnie klasyfikatorów przed atakami zmanipulowanymi danymi jest intensywnie rozwijany. Istnieje wiele możliwych podejść do obrony algorytmów oprócz *moving target defense* badano obrona heurystyczną [6] czy chociażby obronę odporną [18].

- Trening przeciwstawny [13]: Podejście skuteczne do obrony przed atakami, które wykorzystaliśmy podczas badań w projekcie. Niestety nie zapewnia ochrony przed innymi rodzajami ataków. Ponadto następuje utrata dokładności predykcji dla danych niezmodyfikowanych.
- Maskowanie gradientu: (eng. *gradient masking approaches*) [14] swoją strategię opiera na ukryciu funkcji gradientu dla ataków typu white-box podejście zostało złamane całkowicie dla statycznych modeli [6].
- Unlike fMTD [15], Podejście wykorzystuje Morphence, który pomimo permutacji wag modeli i treningu przeciwstawnego, używa w celu zwiększenia odporności, który zamiast głosowania większościowego wybiera model o największej pewności predykcji. Zmiana puli modeli następuje na bieżąco.
- EI-MTD [16] kolejna technika używająca *moving target defense*. Autorzy postanowili użyć do obrony grę Bayesian Stackelberg do dynamicznego wyboru modeli uczniów używanych do predykcji. Modele studentów są generowane z modelu nauczyciela, który jest znajduje się w chmurze.
- MTDeep [17] używa wielu sieci DNN (np. CNN, HRNN, MLP) aby zmniejszyć możliwość przenoszenia ataku pomiędzy architekturami. Dlatego technika ta zawiera pewne podobieństwa do Morphence, jednak nie wykorzystuje głosowania większościowego a grę Bayesian Stackelberg do wyboru modelu użytego do predykcji. Co ciekawe wadą MTDeep jest mała pula dostępnych modeli co wydaje się zaskakujące zwarzywszy na ilość dostępnych sieci DNN.

III. ATAKI PRZECIWSTAWNE

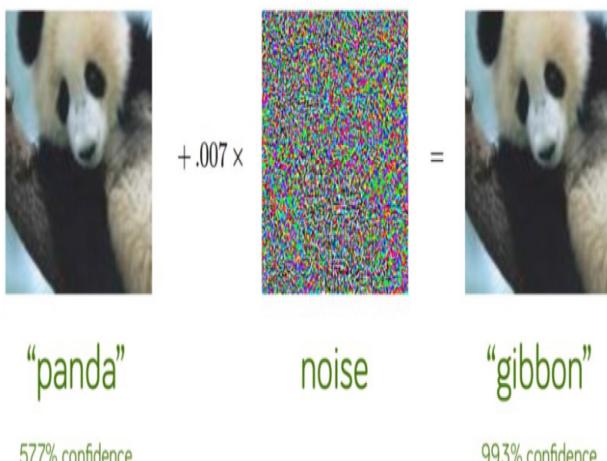
A. Ataki przeciwstawnne

Ataki przeciwstawnie polegają na stworzeniu takich danych wejściowych, które będą myliły sieci neuronowe tak by ta dokonywała błędnych predykcji. Dla atakującego takie ataki mają bardzo dużą zaletę a mianowicie zmanipulowane dane są

nie do odróżnienia dla człowieka. Dzięki czemu użytkownik sieci neuronowej nie jest w stanie zorientować się, że posiada zainfekowane dane.

Jak widać na rysunku 1 dodanie szumu do obrazu, który jest nie widoczny dla człowieka daje kompletnie nieprawidłowe predykcje. Problem ten jest powodowany przez to, że do predykcji obrazów używane są sieci konwolucyjne (CNN), które przywiązuje dużą wagę do występowania regularnych wzorów w obrazach. Dlatego jeśli doda się szum wzorce, które wcześniej były w obrazie zostaną całkowicie zatarte.

Niestety do tej pory techniki bronienia się przed atakami na sieci neuronowe zapewniały tylko niewielką poprawę [2]. Heurystyczne mechanizmy obronne oparte na destylacji obronnej [3], transformacji danych [4] i maskowaniu gradientowym [5] zostały złamane [6].



Rysunek. 1: Przykład manipulacji danych wejściowych

Podczas wykonywania testów używano biblioteki CleverHans [10], które zapewnia standaryzowane implementacje ataków przeciwnostawnych. Biblioteka jest szeroko używana i posiada ponad stu twórców w tym Ian Goodfellow. Bibliotekę pozwala tworzyć standaryzowane testy porównawcze modeli dla ataków przeciwnostawnych.

1) *C&W*: algorytm najpierw używa funkcji straty aby wygenerować przeciwnostawny obraz po czym używa funkcji odległości aby upewnić się, że fakt zmodyfikowania obrazu nie jest widoczny dla ludzkiego oka [7].

2) *FGSM*: oblicza gradienty funkcji straty w odniesieniu do obrazu wejściowego po czym używa funkcji znaków gradientu do utworzenia nowego obrazu, który maksymalizuje straty [8]. Funkcja ta jest opisana równaniem:

$$adv_x = x + \epsilon * sign(\nabla_x J(\Theta, x, y))$$

- adv_x : Zmobilowany obraz
- x : Obraz oryginalny
- y : Oryginalna klasa obrazu
- Θ : model
- ϵ : epsilon
- J : funkcja straty

3) *SPSA*: odmiana algorytmu aproksymacji stochastycznej czyli algorytmów zdolnych do tworzenia funkcji bez dostępu do niej a jedynie z samych dostępnych obserwacji [9]. Aby to osiągnąć algorytmy aproksymacji stochastycznej używają losowe próbki do skutecznego przybliżania właściwości funkcji, takich jak zera lub ekstremum.

B. Obrona przed atakami

W pracy porównuje się skuteczność klasycznych metod zabezpieczania sieci przed przeciwnostawnymi atakami z metodą *moving target defense* - Morphence

1) *Uczenie przeciwnostawne*: Metoda polega na używaniu przypadków przeciwnostawnych na etapie treningu modelu [12]. Niestety odporność zależy od tego jakich ataków używa się do generowania przypadków np. jeśli używa się FGSM model będzie odporny na takie ataki (chyba że atakujący używa przykładu który jest zauważalny przez człowieka jako zniekształcony lub wykryje próg do jakiego model jest zabezpieczony), w przypadku użycia ataku CW uczenie przeciwnostawne przynosi słabe rezultaty. Model nadal może być narażony na inne metody ataku które nie są jeszcze znane lub nieobsługiwane przez uczenie przeciwnostawne.

IV. MORPHENCE

Morphence przesuwa funkcje decyzyjną modelu co utrudnia przeciwnikom oszukiwanie modelu poprzez podawanie sieci neuronowej fałszywych danych wejściowych. Dodatkowo wydaje się radzić sobie lepiej z atakami które są już znane [11] niż obecnie znane metody obrony. Algorytm generuje pulę n modeli wygenerowanych z modelu bazowego, poprzez perturbacje wag modelu bazowego, tak aby każdy z n modeli był możliwie najbardziej różny od siebie oraz modelu bazowego, przy zachowaniu jak największej dokładności precyzji modelu bazowego lub zachowaniu możliwości jej naprawienia z pewnym marginesem błędu. Do wykonania permutacji wykorzystywany jest rozkład Laplace'a z odpowiednio wybranym lambda - im większe lambda tym większe perturbacje nowo powstałych modeli i tym mniejsza jakość predykcji w porównaniu do modelu bazowego oraz zdolność do jej przywrócenia.

$$\frac{1}{2\lambda} epx \left(-\frac{|\theta_b - \mu|}{\lambda} \right)$$

θ_b centrum rozkładu po perturbacji

$$\mu = 0$$

λ zakres perturbacji

Kolejnym krokiem jest dotrenowanie n utworzonych modeli, tak aby uzyskać jak najbardziej zbliżoną dokładność do modelu bazowego, wykorzystując oryginalny zbiór danych. Gdy dotrenowanie się zakończy, następnie p modeli ze zbioru n dotrenowanych modeli jest dodatkowo poddane treningowi przeciwnostawnemu w celu uodpornienia p modeli na ataki przeciwnostawne w największym możliwym stopniu. Tak utworzone n modeli nazywamy studentami, a wszystkich studentów z n nazywamy Zbiorem Studentów. Rysunek ?? przedstawia pseudo kod tworzenia modelu studenta z artykułu opisującego Morphence.

Algorithm 1: Student model retraining function.

```
Result:  $f_s^{(i)}$  : student model
1 Def Retrain( $f_s^{(i)}$ ,  $X_{retrain}$ ,  $X_{test}$ ,  $\epsilon$ ,  $Adv$ ):
    // For adversarial training we use adversarial test
    // examples for validation.
2 if  $Adv = \text{TRUE}$  then
3     |  $X_{test} \leftarrow \Lambda(X_{test})$ 
4 end
5  $acc_{tmp} \leftarrow \text{Accuracy}(f_s^{(i)}, X_{test})$ ;
6  $epochs \leftarrow 0$ ;
7 while  $\text{TRUE}$  do
8     |  $f_s^{(i)}.train(X_{retrain}, epoch = 1)$ ;
9     |  $acc \leftarrow \text{Accuracy}(f_s^{(i)}, X_{test})$ ;
10    | // check training convergence.
11    if  $epochs \bmod 5 = 0$  then
12        | if  $|acc - acc_{tmp}| < \epsilon$  then
13            | | break;
14        | else
15            | |  $acc_{tmp} \leftarrow acc$ ;
16        | end
17    end
18    |  $epochs \leftarrow epochs + 1$ ;
19 end
```

Rysunek. 2: Pseudo-kod tworzenia modelu studenta

Morphence do predykcji próbki wykorzystuje Zbiór Studentów będący aktualnie na początku kolejki, dla każdego studenta ze zbioru jest wykonywana predykcja na próbce. Finalnie jako wynik klasyfikacji zwracany jest rezultat modelu który uzyskał największe prawdopodobieństwo przy klasyfikacji danej próbki.

W celu uniknięcia wykrycia schematu przez atakującego, czy zbudowaniu wystarczającej bazy wiedzy o modelach w Zbiorze Studentów, co Q_{\max} wymieniany jest Zbiór Studentów na będący za nim w kolejce i tworzony jest nowy Zbiór Studentów który po procesie tworzenia zostanie dodany do kolejki.

Aby Morphence był praktycznym rozwiążaniem typu *moving target defense (MTD)* zabezpieczającym modele sieci neuronowych musi realizować następujące punkty:

- Poprawa dokładności modeli wobec znanych i nieznanych ataków typu białej skrzyni i czarnej skrzyni
- Utrzymywanie niezmienionej dokładności predykcji dla prawdziwych danych
- Zwiększenie różnorodności modeli aby zmniejszyć możliwość przenoszenia przykładów przeciwnostnych i metod ich generowania pomiędzy generowanymi modelami.

V. OCENA EKSPERYMENTALNA

W tym rozdziale pierw przedstawiamy w jakim środowisku przeprowadzone zostały badania w podrozdziale V-A , następnie w podrozdziale V-B przedstawiona zostanie metodologia eksperymentów, następnym przedstawianym zagadnieniem są przeprowadzane eksperymenty w podrozdziale V-C, kolejnym zagadnieniem będzie pokazanie i omówienie wyników

eksperymentów w podrozdziale V-D, a na koniec podsumujemy uzyskane wyniki w eksperymencie w podrozdziale V-E.

A. Środowisko eksperymentalne

Zbiór danych: wykorzystujemy zbiór danych MNIST, chcąc porównać nasze wyniki z wynikami twórców Morphence wykorzystujemy 5 tys. próbek na testowanie metody aby ocenić jej wydajność na normalnych danych oraz przeciwnostnych przypadkach.

MNIST to zbiór danych zawierających 70 tys. zdjęć ręcznie zapisanych cyfr od 0 do 9, zdjęcia są w skali szarości i każde z nich ma wymiary 28 na 28 pikseli.

Ataki: W artykule badania przeprowadzone zostały dla dwóch ataków typu white-box (FGSM oraz C&W) które zostały omówione w III-A. Parametry generowania przeciwnostnych przykładów zostały podane w dodatku C

Modele bazowe: Obydwa badane przez nas modele to konwolucyjne głębokie sztuczne sieci neuronowe. Sieci zostały stworzone w PyTorch, a ich opis architektury można znaleźć w dodatku.

Algorytmy do uczenia modeli: Obydwa modele do każdego rodzaju treningu wykorzystywały algorytm Nadam (eng. Nesterov-accelerated Adaptive Moment Estimation) z parametrem kroku wynoszącym 0,01, reszta parametrów domyślna z implementacją w bibliotece PyTorch w wersji 1.11. A Wykorzystane tradycyjne metody obrony: Jako tradycyjną metodę wykorzystujemy uczenie przeciwstawne które uważane jest za jedną z najlepszych metod obrony statycznego modelu. Dodatkowo badamy działanie Morphence bez wykorzystania jakiekolwiek tradycyjnej metody obrony modelu. Hiperparametry: Wstępnie tworzymy 5 zbiorów modeli dla Morphence, z odpowiednimi parametrami p i n , zbiory będą służyły do odpisywania modelu dla 5 tys. próbek ze zbioru testowego, w związku z tym Q_{\max} jest ustawione na 1000. Dzięki temu każdy zbiór dostanie osobne 1000 przykładów do ewaluacji. Wybrane p , n oraz lambda do eksperymentów zostały wybrane tak aby jak najlepiej porównać działanie naszych modeli bazowych z modelem bazowym autorów. Dodatkowo manipulujemy współczynnikiem lambda w zakresie od 0 do 0.3 aby sprawdzić dokładność predykcji, w zależności od intensywności perturbacji wag przy tworzeniu Zbioru Studentów, oraz możliwość przenoszenia ataków między modelami w Zbiorze Studentów.V-B3

B. Metodologia

Dla obydwu stworzonych przez nas modeli bazowych (12 tys. parametrów oraz 3 tys. parametrów) utworzono po 5 Zbiorów Studentów składających się z n modeli w tym p trenowanych dodatkowo przeciwstawnie, wybór parametrów p oraz n zależy od eksperymentu. Do tworzenia n modeli wykorzystano odpowiedni dla eksperymentu współczynnik lambda. Dla obydwu modeli bazowych zostały przeprowadzone identyczne eksperymenty.

Wykorzystano niżej wymienione metryki mierzenia skuteczności działania Morphence, więcej szczegółów na temat procedury ich wyznaczania można znaleźć w artykule autorów:

1) *Dokładność predykcji*: Określa jaki procent przypadków został przypisany do klasy rzeczywistej. Przy treningu modeli na bieżąco jest śledzona dokładność generowanych modeli.

2) *Odporność*: (eng. robustness) oznacza jak dobrze dany algorytm jest w stanie radzić sobie z przeciwnymi danymi wejściowymi, obliczane jest jako procent próbek sklasyfikowanych poprawnie podczas ataku.

3) *Możliwość przenoszenia ataków*: (eng. transferability) określa możliwość użycia ataku wykorzystanego dla danego modelu na inny model, obliczane jest jako różnica liczba przeciwnych próbek którym udało się oszukać model A, i model B przy użyciu tych samych przeciwnych przykładów.

C. Przeprowadzone eksperymenty

1) *Testy dla p i λ* : W projekcie odtworzono eksperymenty przedstawione w artykule prezentującym Morphence. Testowano wyniki sieci przy różnych p z zakresu $0 \leq p \leq p_{max}$ oraz λ z zakresu $0 \leq \lambda \leq \lambda_{max}$. Pomiary zostały przeprowadzone na obu modelach. Dla każdej wartości sprawdzamy jakie wyniki osiągają modele dla danych nie zmanipulowanych następnie sprawdza się wyniki dla ataków C&W oraz FGSM.

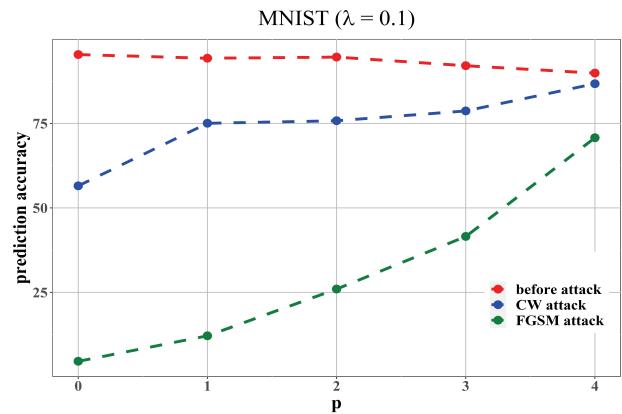
2) *Testy dla oryginalnego modelu i trenowanych z użyciem klasycznych rozwiązań*: Wykonano testy porównawcze jak oryginalny model radzi sobie z atakiem oraz jak radzi sobie model, który był wykorzystujący trening przeciwny.

D. Wyniki eksperymentów

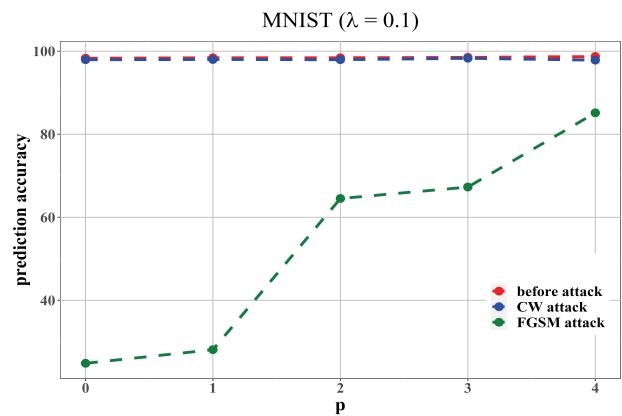
Na rysunkach 3 i 4 można zauważyć, że dla ataku FGSM oba modele osiągnęły gorsze wyniki niż dla ataku C&W. Wyraźnie widać pozytywny wpływ zwiększania p na wyniki uzyskane przez Morphence. Oba testowane modele sieci neuronowych uzyskały różniące się wyniki ale tendencje zostały zachowane. Wzrost odporności na atak FGSM przy co raz większych wartościach parametru p wynika z co raz większej liczby przeciwnie trenowanych modeli studentów w zbiorze. Gdy $p=n$ wszystkie modele w zespole są maksymalnie uodpornione na ten atak, oraz dzięki strategii wyboru najbardziej prawdopodobnej predykcji przez Morphence atak FGSM staje się bardzo ograniczony w porównaniu do mniejszych wartości p . Niestety rosnąca wartość p jest obarczona rosnącą złożonością obliczeniową tworzenia zespołu modeli.

Zgodnie z oczekiwaniemi wzrost liczby trenowanych przeciwnie modeli (wzrost p) przynosi niewielką degradację dokładności predykcji na rzeczywistym zbiorze danych dla obydwu modeli - wynika to z perturbacji, w tym przypadku dla $\lambda=0.1$. Dla większego modelu widać że skuteczność ataku C&W nie zależy od p , zaś dla mniejszego modelu odporność na atak CW rośnie wraz ze wzrostem p , można to uzasadnić niewielką liczbą parametrów modelu - przez to jest on łatwiejszy do zaatakowania. Dodatkowy wpływ może mieć na wyniki to że mniejszy model jest mniej podatny na perturbacje ponieważ posiada mniej wag które zostaną znieksztalcone. Zbiór złożony z mniejszych modeli poddanych

perturbacji i dotrenowaniu jest mniej różnorodny niż zbiór złożony z większych modeli poddany takiemu samemu procesowi ze względu na większą liczbę parametrów które mogą zostać w poszczególnych modelach znieksztalcone. Jednym z rozwiązań tego problemu może być zwiększenie parametru lambda (sily z jaką parametry sieci poddawane są perturbacjom), dlatego w jednym z następnych eksperymentów postanowiliśmy zbadać wpływ różnych wartości lambda na dokładność predykcji podczas ataków.



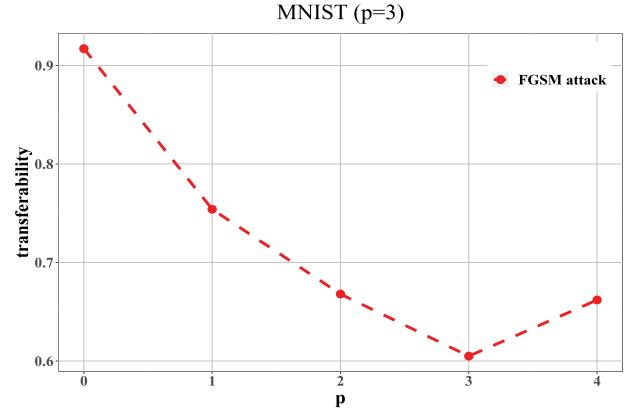
Rysunek. 3: Rezultaty badania dokładności precyzji podczas różnych ataków, dla różnych wartości parametru p , $n=4$, $\lambda=0.1$ model 4 tyś. parametrów



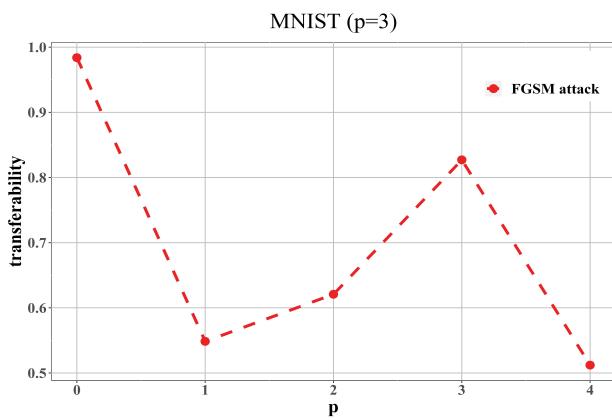
Rysunek. 4: Rezultaty badania dokładności precyzji podczas różnych ataków, dla różnych wartości parametru p , $n=4$, $\lambda=0.1$, model 12 tyś. parametrów

W celu zbadania wpływu różnych wartości parametru lambda na dokładność predykcji podczas ataków najpierw zbadaliśmy dla jakiego p wyniki wydają się dawać najlepsze rezultaty, do tego celu wykorzystaliśmy średnią możliwość przenoszenia ataków. Na rysunkach 5 i 5 można zauważyć że wartości dla większego i mniejszego modelu nie wykazują tej samej monotoniczności. Dla modelu składającego się z 12 tys. parametrów model zachowuje się podobnie jak model 1,2 miliona parametrów autorów Morphence, początkowo średnia możliwość przenoszenia ataków między modelami w

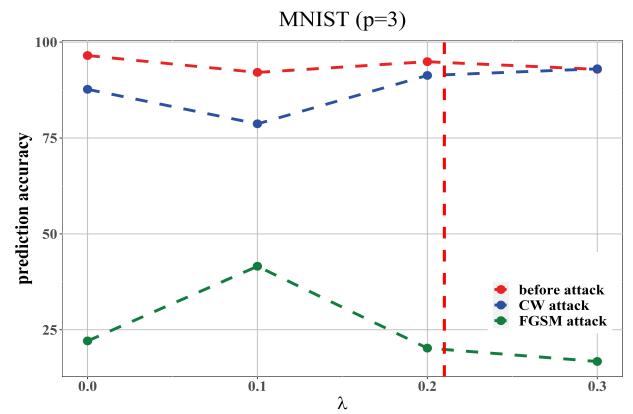
zbiorze studentów jest wysoka ponieważ żaden model nie jest trenowany przeciwnie (dużo podobnych podatnych na atak modeli). Wartość ta maleje wraz ze wzrostem liczby przeciwnie trenowanych modeli - różne modele w grupie, zarówno nietrenowane jak i trenowane przeciwnie, duża różnorodność, atak który zadziała na modelu zwykłym nie zadziała na trenowanym przeciwnie i odwrotnie. Gdy zbiór zawiera same przeciwnie trenowane modele średnia możliwość przenoszenia ataków znowu rośnie - wszystkie modele podobnie trenowane przeciwko atakowi więc większa średnia możliwość przenoszenia ataku. Dla mniejszego modelu sytuacja jest trudna do wyjaśnienia, analizując tylko rysunek 5 nie można nic powiedzieć. Analizując obydwa wykresy jednocześnie tj. biorąc pod uwagę również rysunek 3 można zauważać że odporność na ataki FGSM mniejszego z modeli jest na tyle niska że analizowanie różnic w zależności średniej możliwości przenoszenia ataków w zależności od p w tym przypadku traci sens. Model zbyt słabo jest broniony dla $p < 4$ i analizowanie w przedziale p od 0 do 3 traci sens.



Rysunek. 6: Rezultaty badania średniej możliwości przenoszenia ataków podczas różnych ataków, dla różnych wartości parametru p , $n=4$, $\lambda=0.1$, model 12 tys. parametrów

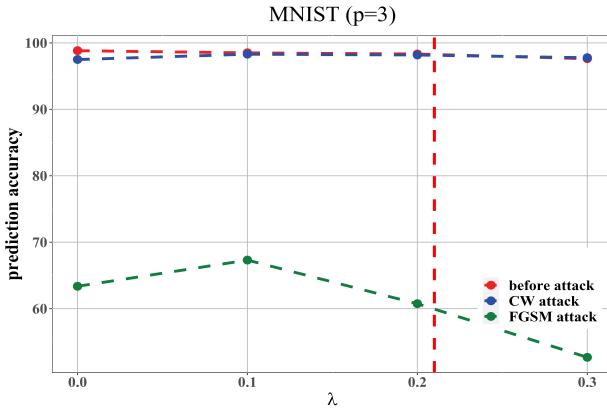


Rysunek. 5: Rezultaty badania średniej możliwości przenoszenia ataków podczas różnych ataków, dla różnych wartości parametru p , $n=4$, $\lambda=0.1$, model 4 tys. parametrów

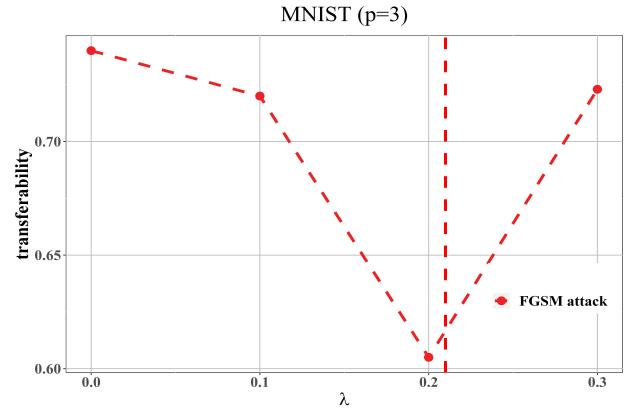


Rysunek. 7: Rezultaty badania dokładności precyzji podczas różnych ataków, dla różnych wartości parametru λ , $n=4$, $p=3$, model 4 tys. parametrów

Na podstawie poprzedniego eksperymentu wybrano $p=3$ do badań wpływu różnych wartości λ na dokładność precyzji. Na rysunkach można zauważać że wybrany przez autorów Morphence parametr $\lambda=0.1$ jest również najlepszy dla naszych modeli. Jego stopniowy wzrost a następnie spadek można uzasadnić tym że gdy $\lambda=0$ nie następują perturbacje modelu bazowego i modele zbioru studentów są kopią modelu bazowego, nie wnoszą zbyt wiele do precyzji gdyż wszystkie przewidują to samo. Gdy λ jest zbyt duże to perturbacje nie pozwalają się dobrze dotrenaować aby uodpornić modele ze zbioru studentów na atak FGSM. Modyfikacje λ , współczynnika opowiadającego za siłę perturbacji nie przyniosło pozytywnych skutków.

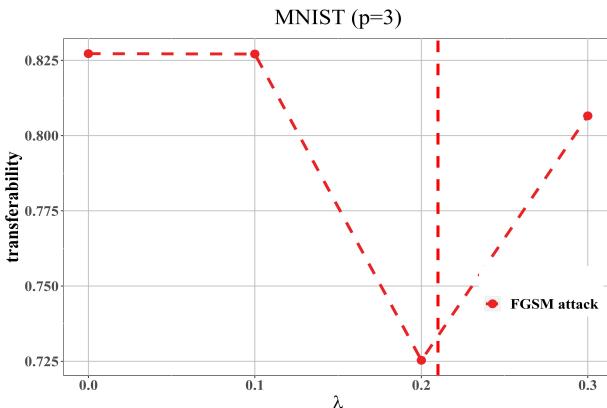


Rysunek. 8: Rezultaty badania dokładności precyzji podczas różnych ataków, dla różnych wartości parametru lambda, n=4, p=3, model 12 tyś. parametrów



Rysunek. 10: Rezultaty badania średniej możliwości przenoszenia ataków podczas różnych ataków, dla różnych wartości parametru lambda, n=4, p=3, model 12 tyś. parametrów

Dodatkowo postanowiono zbadać wpływ zmian parametru lambda przy p=3 na średnią możliwość przenoszenia ataku. Na rysunkach 9 oraz 10 widać że dla lambda=0.2 obydwa modele wykazują największą odporność na przenoszenie ataków między modelami w zbiorze studentów. Może to być przyczyną dobrze dobranej proporcji pomiędzy siłą perturbacji zbioru studentów i zdolnością do skutecznego uczenia przeciwnego.



Rysunek. 9: Rezultaty badania średniej możliwości przenoszenia ataków podczas różnych ataków, dla różnych wartości parametru lambda, n=4, p=3, model 4 tyś. parametrów

1) Wyniki predykcji podczas ataków i bez gdy sieć nie jest chroniona, gdy używa się tylko treningu przeciwnego do ochrony, oraz obrona za pomocą Morphence dla porównania: Morphence w porównaniu do niebronionego modelu jak i do używającego treningu przeciwnego nie traci dokładności dla niezmodyfikowanych danych I, II. Modele z którymi porównuje się Morphence osiągnęły zerową dokładność dla ataku C&W gdy używany przez nas algorytm osiągnął wyniki powyżej 80 procent. Dla ataku FGSM Morphence kolejny raz okazał się oferować najwyższą ochronę. Tak jak autorzy Morphence użyto

MNIST 12 tyś. parametrów Dokładność			
Atak	Niebroniony model	Trening przeciwny	Morphence
Brak ataku	98,7%	96,69%	98,59%
FGSM	11,41%	41,05%	63,3%
C&W	0%	0%	97,41%

Tabela I: Wyniki dla modelu 12 tyś. parametrów

MNIST 4 tyś. parametrów Dokładność			
Atak	Niebroniony model	Trening przeciwny	Morphence
Brak ataku	97,5%	91,97%	97,19%
FGSM	6,75%	10,37%	20,02%
C&W	0%	0%	85,2%

Tabela II: Wyniki dla modelu 4 tyś. parametrów

MNIST 1,2 mln. parametrów Dokładność			
Atak	Niebroniony model	Trening przeciwny	Morphence
Brak ataku	97%	97,17%	99,04%
FGSM	9,98%	42,38%	71,43%
C&W	0%	0%	97,75%

Tabela III: Wyniki dla modelu bazowego 1,2 miliona parametrów używanego przez autorów Morphence

E. Podsumowanie eksperymentów

Przed wykorzystaniem badanej metody obrony, dla każdego z badanych modeli ustawionych statycznie atak C&W potrafił idealnie tworzyć przykłady przeciwnostawne przez co modele zawsze się myliły, w tym nawet duży model autorów Morphence. Zastosowanie badanej metody obrony typu moving target defense przyniosło rezultat na poziomie 97% poprawnych predykcji dla modelu z 12 tysiącami parametrów, jest to różnica zaledwie jednego punktu procentowego od dokładności predykcji bazowego modelu na rzeczywistym zbiorze danych, na tej podstawie można stwierdzić że metoda prawie idealnie realizuje swoje zadanie dla ataku CW. Dla najmniejszego badanego modelu również widzimy wzrost dokładności predykcji o 85 punktów procentowych podczas ataku C&W wykorzystując Morphence - różnica w dokładności obydwu modeli prawdopodobnie wynika z gorszej jakości mniejszego modelu bazowego oraz większą podatność na atak C&W przez małą liczbę parametrów. Dla ataku C&W nasz 100 razy mniejszy model od modelu autorów, wykorzystując Morphence posiada gorszą dokładność predykcji o zaledwie 0,3 punktu procentowego, zaś dla modelu 300 razy mniejszego różnica wynosi już 12,5 punktu procentowego. Na tej podstawie można stwierdzić że skuteczność badanej metody ochrony modelu jest zależna w dużym stopniu od jakości predykcji modelu bazowego dla rzeczywistego zbioru danych. Analizując wyniki ataku FGSM można zauważać podobną tendencję, lecz w dużo większej skali, wyniki dla modelu 100 krotnie mniejszego chronionego przez Morphence odbiegają w naszym eksperymencie o 8 punktów procentowych zaś dla 300 krotnie mniejszego jest to ponad 51 punktów procentowych. Przyczyną tak niskiego wyniku dla mniejszego modelu jest jego brak podatności na przeciwnostawne uczenie, można zauważać że dla najmniejszego badanego modelu wzrost dokładności predykcji modelu trenowanego przeciwnostawnie na atak FGSM wzrasta tylko o 4 punkty procentowe, gdzie dla większych modeli ten wzrost wynosi około 30 punktów procentowych. Może to być spowodowane małą liczbą parametrów modelu do perturbacji. Wpływ lambda jest marginalny, najlepsze lambda wynosiło 0.1 dla jakości predykcji oraz 0.2 dla zminimalizowania przenoszenia ataków między modele studentów. Z przeprowadzonych eksperymentów wynika również że dobór odpowiedniego p to indywidualna kwestia która generalnie zależy od dostępnej ilości mocy obliczeniowej gdyż najczęściej im większe p tym lepsza jakość drużyny studentów.

VI. WNIOSKI I DALSZE MOŻLIWOŚCI ROZWOJU

Pomyślnie udało nam się wykorzystać Morphence do obrony naszych modeli przed atakami FGSM, oraz C&W - który jest wyjątkowo skuteczny i nie istnieją obecnie metody chroniące statyczny model przed tym atakiem. Przetestowaliśmy modele 300 i 100 razy mniejsze od badanego przez autorów modelu z sukcesami i porażkami. Generalnie dla 100 razy mniejszego modelu udało się prawie odwzorować jakość działania modelu z artykułu o Morphence, lecz niestety wykorzystanie morphence dla modelu 300 razy mniejszego

(choć tylko z o 2 punkty procentowe niższą dokładnością precyzji na rzeczywistym zbiorze danych) nie przyniosło dużej poprawy w stosunku do klasycznych metod.

Istotną kwestią jest poprawa odporności modelu dla wszystkich przetestowanych ataków wykorzystując podejście moving target defense. Metoda Morphence nie wprowadza niczego nowego pod względem uczenia maszynowego, korzysta z dostępnych rozwiązań które są dosyć proste w użyciu, lecz odpowiednie tworzenie wielu (diversity) różnych (shuffling) modeli w wielu zbiorach (redundancy) które następnie wykorzystywane są zamiast predykcji stałego modelu sprawdza się naprawdę świetnie. Porównując wszystkie poprzednie - klasyczne metody zabezpieczania niezmiennych modeli sztucznych sieci neuronowych Morphence jest od nich lepszy i przynosi ogromną poprawę. Morphence okazuje się metodą pozwalającą tworzyć modele o wyższej odporności na ataki niż trenowanie przeciwnostawne. Pomyślne wyniki dla Morphence pokazują, że *moving target defense* jest efektywnym sposobem na obronę sieci neuronowych przed atakami.

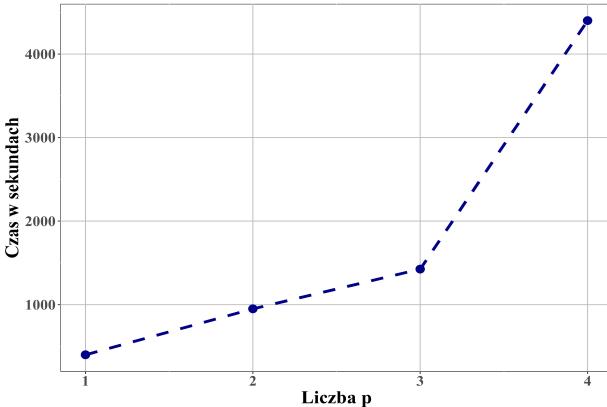
A. Koszty zastosowania Morphence

Niestety to rozwiązanie obarczone jest wysokim kosztem aby zrealizować badania musieliszy wynająć 4 karty RTX 3090 na stronie www.vast.ai, wraz ze wzrostem N oraz w szczególności P rośnie zapotrzebowanie na moc obliczeniową. Naszym zdaniem aktualnie Morphence jest zbyt zasobozerny aby go wykorzystywać. Zadanie klasyfikacji MNIST jest prostym zadaniem, wymagającym mało wymagającym, niestety sprzęt wykorzystany do jego użycia wyklucza go z komercyjnego użytku.

Koszta implementacji Morphence są znaczące, trening dla p=4 trwa średnio ponad 4400 sekund czyli 1.2 godziny. 11. Natomiast używanie mniejszych wartości p ogranicza czas i dokładność algorytmu przed atakami. Cały czas należy przy tym pamiętać, że do treningu używano RTX 3090, który nie jest dostępny dla większości użytkowników. Dla słabszych kart czas treningu jedynie wzrosnie.

Wytrening jednej epoki dla modelu 4 tyś. parametrów zajmowało 6s, model 12 tyś. parametrów potrzebował 10s, zaś model autorów 1,2 mln parametrów potrzebował 24 sekundy. Aktualnie wykorzystuje się sieci które mają ponad 300 milionów i więcej parametrów, jest to niestety zbyt dużo dla Morphence i wtedy staje się to rozwiązanie nieprzydatne.

Trening modeli wymagał użycia 3 Gb pamięci VRAM, lecz do badań wykorzystano RTX 3090 kierując się szybkością tej karty graficznej.



Rysunek. 11: Średni czas potrzebny na wytrenowanie p dla modelu 4K

BIBLIOGRAFIA

- [1] Explaining and Harnessing Adversarial Examples, Goodfellow et al, ICLR 2015
- [2] Shixiang Gu and Luca Rigazio. 2015. Towards Deep Neural Network Architectures Robust to Adversarial Examples. In 3rd International Conference on Learning Representations, ICLR 2015.
- [3] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. 582–597. <https://doi.org/10.1109/SP.2016.41>
- [4] Enhancing robustness of machine learning systems via data transformations. In 52nd Annual Conference on Information Sciences and Systems, CISS 2018. IEEE, 1–5.
- [5] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian J. Goodfellow. 2018. Thermometer Encoding: One Hot Way To Resist Adversarial Examples. In 6th International Conference on Learning Representations, ICLR 2018. OpenReview.net.
- [6] Anish Athalye, Nicholas Carlini, and David A. Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018 (Proceedings of Machine Learning Research, Vol. 80). PMLR, 274–283.
- [7] Carlini, Nicholas, Wagner, David. (2017). Towards Evaluating the Robustness of Neural Networks. 39-57. 10.1109/SP.2017.49.
- [8] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).
- [9] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," in IEEE Transactions on Automatic Control, vol. 37, no. 3, pp. 332-341, March 1992, doi: 10.1109/9.119632.
- [10] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. arXiv preprint arXiv:1610.00768, 2018.
- [11] Amich, Abderrahmene and Eshete, Birhanu. (2021). Morphence: Moving Target Defense Against Adversarial Examples.
- [12] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141–142.
- [13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. ICLR, 2015.
- [14] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2018. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. In 6th International Conference on Learning Representations, ICLR 2018.
- [15] Qun Song, Zhenyu Yan, and Rui Tan. 2019. Moving target defense for embedded deep visual sensing against adversarial examples. In Proceedings of the 17th Conference on Embedded Networked Sensor Systems, SenSys 2019. ACM, 124–137.
- [16] Yaguan Qian, Qiqi Shao, Jianmin Wang, Xiang Lin, Yankai Guo, Zhaoquan Gu, Bin Wang, and Chunming Wu. 2020. EI-MTD: Moving Target Defense for Edge Intelligence against Adversarial Attacks. CoRR abs/2009.10537 (2020).
- [17] Sailik Sengupta, Tathagata Chakraborti, and Subbarao Kambhampati. 2019. MTDeep: Boosting the Security of Deep Neural Nets Against Adversarial Attacks with Moving Target Defense. In Decision and Game Theory for Security - 10th International Conference, GameSec 2019 (Lecture Notes in Computer Science, Vol. 11836). Springer, 479–491.
- [18] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. 2019. Certified Adversarial Robustness via Randomized Smoothing. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019 (Proceedings of Machine Learning Research, Vol. 97). PMLR, 1310–1320.

DODATEK

Dodatek opisuje środowisko w jakim wykonywano testy.

A. Wykorzystane modele

1) Model 12K: posiada łącznie dwanaście tysiące parametrów.

Conv2d(in channels = 1, out channels = 10, kernel size = 5) + Relu; + MaxPool2d + Conv2d(in channels = 10, out channels = 20, kernel size = 5) + Relu + Cov2Drop; + MaxPool2d + Linear(in features=320,out features=20) + Linear(in features=20,out features=10)

2) Model 4K: Model posiada łącznie cztery tysiące parametrów, model ten posiada porównywalna architekturę do tej używanej dla Morphence jednak korzysta dodatkowo z Dropout i AvgPool2d.

*Conv2d(in channels = 1, out channels = 8, kernel size = 5) + Relu + Drop; + AvgPool2d(2, stride = 2) + Conv2d(in channels = 8, out channels = 4, kernel size = 5) + Relu + Drop; + AvgPool2d(2, stride = 2) + Linear(in features=4*4*4,out features=36) + Linear(in features=36,out features=10)*

B. Wykorzystane zbiory

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
```

Rysunek. 12: Przykładowe przypadki ze zbioru MNIST

Podczas prac wykorzystano zbiór danych MNIST [12], który zawiera obrazy ręcznie napisanych liczb od 0 do 9 12.

Zbiór ten posiada 60 000 obrazów do treningu i 10 000 obrazów do testów. Każdy z obrazów ma rozmiary 28x28 pikseli, zawierające kolory w skali szarości. MNIST jest jednym z najczęściej używany zbiorów w dziedzinie ML.

C. Domyślne parametry

Podczas pomiarów wykorzystywano stale następujące wartości:

- Wielkość batch: 1024
- epsilon: 0.3
- n: 4
- rozmiar test set: 5000
- Q_{max} : 1000
- Liczba generowanych pools: 5

D. Specyfikacja użytego sprzętu

Podczas obliczeń używano:

- RTX 3090 24 GB
- AMD EPYC 7351P 16c 2.4GHz