Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki i Telekomunikacji

INSTYTUT INFORMATYKI

PRACA MAGISTERSKA

NATALIA ORGANEK

ATAKI NA SIECI NEURONOWE PRZETWARZAJĄCE OBRAZY

ATTACKS ON IMAGE-PROCESSING NEURAL NETWORKS

PROMOTOR:
dr inż. Łukasz Faber

Kraków 2022

Praca została wykonana
z wykorzystaniem Infrastruktury
PLGrid.

Spis treści

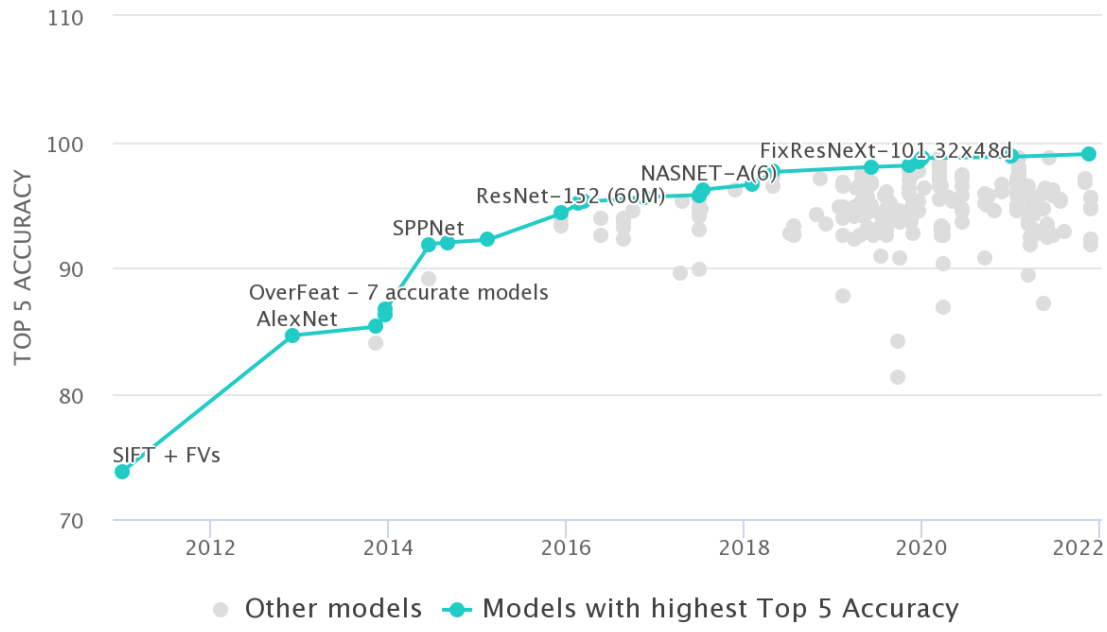
1. Wprowadzenie

W tym rozdziale przedstawiona zostanie motywacja pracy, związana z popularyzacją sieci neuronowych przetwarzających obrazy oraz z zagrożeniami z tego wynikającymi. W dalszej części przedstawiony zostanie cel pracy, którym jest odpowiedź na pytania badawcze przy pomocy przytoczenia pozycji literatury naukowej oraz przeprowadzenia serii eksperymentów przy wykonaniu ataków na sieci neuronowe rozpoznające znaki drogowe.

1.1. Motywacja

Sieci neuronowe to szeroko rozwijane i wykorzystywane modele uczenia maszynowego, które znalazły zastosowanie w wielu dziedzinach. Historia sieci neuronowych jest długa - pierwsze próby stworzenia modelu działającego na zasadzie pracy ludzkiego mózgu dokonywane były już w latach czterdziestych XX wieku [43]. Rozkwit sieci neuronowych rozpoczął się w XXI wieku - niektóre źródła wskazują rok 2006 jako rok rozpoczęcia nowej fali popularności uczenia głębokiego [13]. Od tego czasu sieci neuronowe rozwijały się, by w roku 2011 przezwyciężyć inne modele w konkursie widzenia maszynowego, swobodnie rozpoznając obiekty na obrazach wysokiej rozdzielczości [5] i osiągając wynik *superhuman*, dwukrotnie przewyższający wynik uzyskany przez człowieka [36]. Tą siecią była głęboka konwolucyjna sieć neuronowa D. Ciresana zaimplementowana na GPU. DanNet rozpoczął rewolucję wśród konwolucyjnych sieci neuronowych. Od tego czasu modele były sukcesywnie rozwijane - dodawano do nich coraz bardziej rozbudowane i ulepszone warstwy oraz sprawdzano różne konfiguracje połączeń pomiędzy nimi, by zwiększyć skuteczność i zmniejszyć koszty obliczeniowe sieci. W ten sposób w niedługim czasie powstawały coraz lepsze AlexNet [24], ResNet [17] i wiele innych implementacji. Rozwój sieci neuronowych przez ostatnią dekadę można zobaczyć na wykresie udostępnionym przez Papers With Code (Rysunek 1.1), który przedstawia dokładność sieci neuronowych na zbiorze ImageNet na przestrzeni lat (pod uwagę brane jest pierwsze pięć wyników sieci). Obecnie sieci osiągają dokładność bliską 100% - sieć Florence-CoSwin-H w 2021 roku osiągnęła dokładność 99,020% dla top-5 accuracy i 90,05% dla top-1 accuracy na zbiorze ImageNet [44].

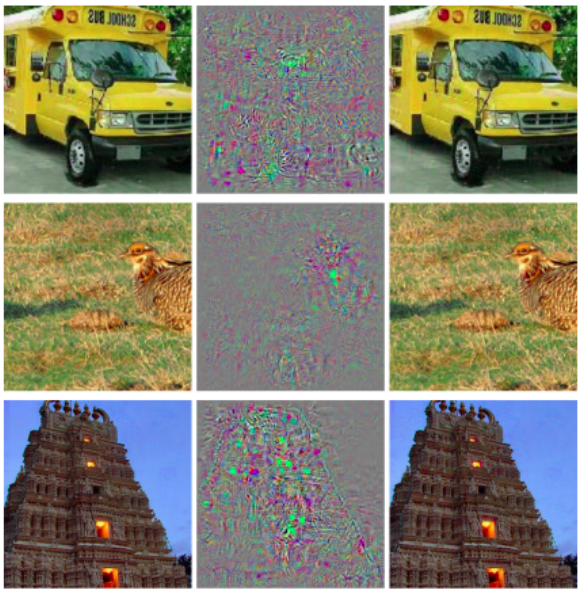
Dzisiaj zastosowanie sieci neuronowych jest dość proste - wystarczy użyć istniejącą sieć neuronową i dostosować ją do rozwiązywanego problemu. Łatwość użycia, duża skuteczność i różnorodność możliwych zastosowań spowodowała, że sieci neuronowe są częstym wyborem przy znajdowaniu rozwiązań technologicznych. Są wykorzystywane do przetwarzania dźwięku [23], przewidywania ruchu drogowego [21], badania rynków [39] czy wykrywania oszustw [34]. Szczególnie szerokie zastosowanie mają sieci



Rysunek 1.1: Wykres przedstawiający dokładność sieci neuronowych w klasyfikacji zbioru ImageNet w latach 2011-2022 (stan na styczeń 2022r.). Źródło: [7].

neuronowe przetwarzające obrazy. Wykorzystywane są między innymi w medycynie [25][15][18], do autentykacji (np. na podstawie rozpoznawania twarzy [20]), przy analizie dokumentów [42] czy w pojazdach autonomicznych [2]. Ich popularność i łatwość użycia otwiera jednak drogę do nowej klasy ataków - ataków na sieci neuronowe - które mogą być dużym zagrożeniem dla poprawności danych, bezpieczeństwa, a nawet dla zdrowia i życia ludzkiego. Jednym z pierwszych był atak na sieć neuronową rozpoznającą niechciane wiadomości (tzw. spam) z 2004 roku [8]. W 2014 wykorzystano skłonność konwolucyjnych sieci neuronowych do generalizacji, by zmienić ich predykcje [40].

Wyniki tego eksperymentu można zobaczyć na Rysunku 1.2. Od tamtej pory wykonano i opisano więcej rodzajów ataków na sieci neuronowe. Okazało się, że predykcje sieci można zaburzyć i to nie w przypadkowy sposób, ale tak, by wynik predykcji zależał od atakującego. Problem nie pozostał zlekceważony i w odpowiedzi na ataki zaczęły pojawiać się również artykuły wskazujące zabezpieczenia sieci przed nimi [29]. Możliwości atakujących nadal są jednak szerokie, w dodatku nie wszyscy wykorzystujący sieci neuronowe na potrzeby rozwiązywania problemów z życia codziennego, ekonomii czy medycyny wiedzą o wrażliwości tego modelu na ataki. Zła predykcja obrazu medycznego lub pojazdu autonomicznego może nieść za sobą poważne konsekwencje. Prace opisujące nowe rodzaje ataków są potrzebne, aby zwracać uwagę na ten problem. Dodatkowo, by móc zabezpieczyć się przed niebezpieczeństwem wynikającym z ataków na sieci neuronowe przetwarzające obrazy, neleży się z nimi zapoznać i zrozumieć zasadę ich działania. Niestety, ataki na sieci neuronowe trudno jest zniwelować, co potwierdzają słowa jednych z czołowych naukowców w tej dziedzinie - Iana Goodfellow and Nicolasa Papernot - mówiące, że z teoretycznego punktu widzenia jeszcze nie wiadomo, czy próby obrony sieci neuronowych przed złośliwymi obrazami to beznadziejne dążenie (jak próby znalezienia uniwersalnego



(a)

Rysunek 1.2: Rysunek przedstawia atak przeprowadzony na sieci AlexNet. Za pomocą niewidocznej dla ludzkiego oka modyfikacji zmieniono predykcję sieci neuronowej tak, by rozpoznawała obrazy z prawej jako „struś”.). Źródło: [40].

algorytmu uczenia maszynowego) czy jednak optymalna strategia dałaby obrońcy przewagę (jak w kryptografii)[19].

1.2. Cele pracy

Celem poniższej pracy jest zbadanie możliwości przygotowania ataku, który powodować będzie błędy sieci neuronowej przeznaczonej do przetwarzania obrazów. W ramach pracy zbadano podatność wybranych rodzajów sieci neuronowych na ataki. W tym celu przygotowano zestawienie rodzajów ataków na sieci neuronowe pojawiające się na przestrzeni lat. Przeprowadzono również serię eksperymentów do wykonania ataku typu *backdoor* na różne sieci neuronowe rozpoznające znaki drogowe w celu zbadania, czym jest i czym charakteryzuje się sukcesywny atak na sieć neuronową, czyli:

– jaki klucz ataku (inaczej nazywany *backdoor pattern*, złośliwym wzorem, maską, wyzwalaczem) pozwoli na najlepszą trafność ataku i czy klucz ten da się stworzyć tak, by był trudno rozpoznawalny przez człowieka;

– w jakim stopniu zatrucie zbioru wpływa na sukces ataku i czy można przeprowadzić sukcesywny atak zatruwając niewielką liczbę obrazów ze zbioru.

Dodatkowo poruszanym zagadnieniem jest wpływ liczby klas zbioru danych na celność ataku.

1.3. Zawartość rozdziałów

Rozdział 1 zwiera wstęp oraz założone cele niniejszej pracy magisterskiej.

Rozdział 2 zawiera przegląd literatury przedstawiającej historię oraz typy ataków na sieci neuronowe przetwarzające obrazy, ze szczególnym naciskiem na atak typu *backdoor* oraz ataki skierowane przeciw sieciom neuronowym rozpoznającym znaki drogowe.

Rozdział 3 poświęcony jest teoretycznemu opisowi przygotowania sieci neuronowych, złośliwych wzorów oraz wykonania ataków przeciwko przygotowanym modelom. Przedstawiono w nim dwa typy masek wykorzystywanych do dokonania ataku (statyczne - niezależne od atakowanego modelu oraz adaptacyjne - zależne od modelu) oraz trzy wybrane modele sieci neuronowych (AlexNet, ResNet, MobileNet). Wyszczególnione zostały również założenia wykonania ataku oraz jego scenariusz. Rozdział zawiera również opis metryk, które posłużą do określenia sukcesu ataku oraz opis technologii uzytej podczas badań.

Rozdział 4 zawiera opis wykonanych badań - charakterystykę opisanych sieci neuronowych, przebieg ataków, wyniki ataków wykonanych za pomocą wzorów statycznych oraz adaptacyjnych.

Rozdział 5 zawiera podsumowanie oraz odpowiedź na postawione w Rozdziale 1.2 pytania badawcze.

2. Powiązane prace

W tym rozdziale przedstawiono wcześniejsze prace dotyczące tematu ataków na sieci neuronowe przetwarzające obrazy, w szczególności służące do rozpoznawania pionowych znaków drogowych.

2.1. Ataki na cały obraz

Pierwsze ataki na sieci neuronowe wykorzystywały zmiany w całym obrazie w celu przekłamania wyników sieci neuronowej.

W przedstawionym wcześniej artykule [40] z 2014 roku zauważono, że sieci neuronowe są wrażliwe na złośliwe próbki obrazów. Do ich wygenerowania użyto aproksymowanej funkcji minimalnego zniekształcenia obrazu, wykorzystując skłonność sieci do lokalnego generalizowania pikseli. Minimalne zniekształcenie powstawało przez minimalizację normy z niewielkiego promienia przesunięcia ($\min ||r||_2$), który spełniałby dwa warunki:

- $f(x + r) = l$
- $x + r \subseteq [0, 1]^m$

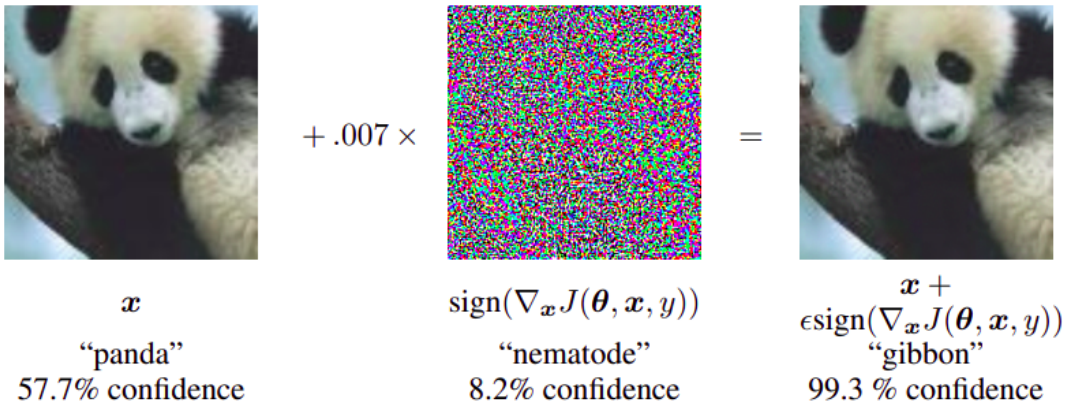
gdzie f - funkcja mapująca obraz na klasę, x - obraz ze zbioru danych, l - etykieta obrazu, m - wymiarowość zbioru danych. Wyniki eksperymentu widoczne są na Rysunku 1.2, gdzie środkowa kolumna przedstawia różnicę między prawym (zainfekowanym) a lewym obrazem powiększoną 10 razy. Niewielkie zniekształcenie zmieniło drastycznie wyniki sieci. Autorzy pracy twierdzą również, że zmodyfikowany obraz był uniwersalny i zmieniał wyniki sieci niezależnie od hiperparametrów (liczby warstw, wag wewnętrznych, regularyzacji). Autorzy sami przyznają, że nie wiedzą, dlaczego małe, statystycznie nieznaczące, perturbacje zmieniają predykcje sieci neuronowych. Jako możliwe wyjaśnienie podano wtedy nieliniowość i przesadne dopasowanie (*overfitting*), i pozostawiono do rozważań w przyszłych pracach.

Tej tezie sprzeciwiono się już w 2015 roku w pracy [14], udowadniając, że to właśnie liniowość w wysokowymiarowej przestrzeni generuje złośliwe próbki. Otworzyło to drogę do stworzenia łatwego sposobu generowania złośliwych nakładek na obraz wzorem:

$$\eta = \epsilon \text{sign}(\Delta_x J(\mathbf{\Theta}, \mathbf{x}, \mathbf{y}))$$

gdzie $\mathbf{\Theta}$ to parametry modelu, $\mathbf{x}$ to dany obraz, y - etykiety obrazu x, a funkcja J to funkcja kosztu stosowana do trenowania sieci. Metodę tę nazwano „fast gradient sign method" (FGSM). Należy zauważyć, że gradient jest łatwo wyznaczalny przy pomocy wstecznej propagacji. Tak policzona perturbacja

dodawana jest do obrazu ze znakiem przeciwnym niż gradient wykorzystany w uczeniu sieci neuronowej, przez co maksymalizuje się funkcję kosztu zamiast ją minimalizować. Rezultat tego eksperymentu można zobaczyć na Rysunku 2.1.



Rysunek 2.1: Wykonanie ataku metodą FGSM na sieć GoogLeNet i zbiorem ImageNet.). Źródło: [14].

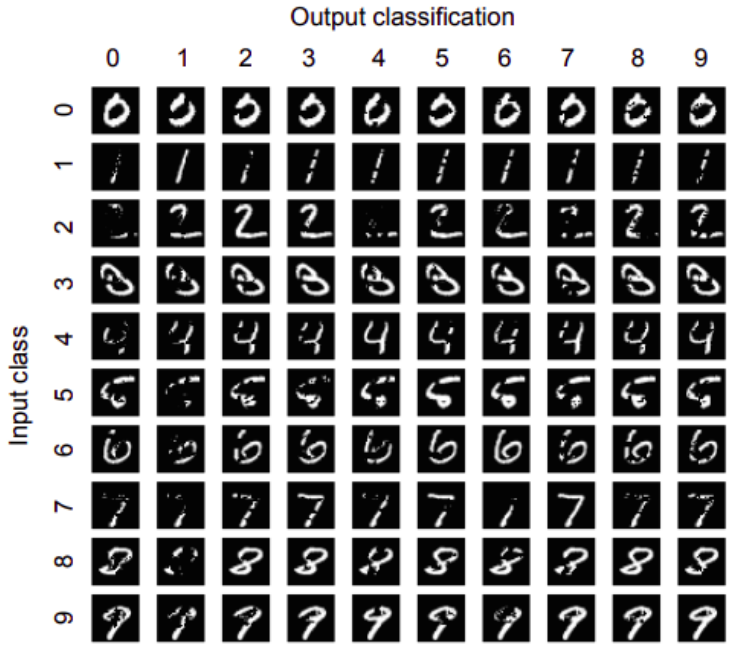
Powyższa metoda pozwala na tak zwany ***non-targetted*** atak, czyli taki, w którym atakujący nie wybiera klasy, którą ma rozpoznać zmylona sieć. W artykule [27] zaproponowano ulepszenie metody FGSM, która pozwala na wybranie celu. Tym razem perturbacja jest liczona w następujący sposób:

$$\eta = -\epsilon\text{sign}(\Delta_x J(\mathbf{\Theta}, \mathbf{x}, \mathbf{y}'))$$

gdzie y' to etykieta wybrana przez atakującego, inna niż prawdziwa. Jednak policzenie gradientu funkcji kosztu i nałożenie go na cały obraz jest dość kosztowne i nie sprawdziłoby się w czasie rzeczywistym.

W 2017 roku znaleziono sposób na zmniejszenie liczby zmienianych pikseli [31] - średnio na zdjęciu zmieniane było 4,03% pikseli. Taki atak by skuteczny w 97%. Na rysunku 2.2 przedstawiono gotowe próbki spreparowane na podstawie zbioru MNIST, które miały oszukiwać sieć neuronową.

Tworzone w ten sposób próbki nie są uniwersalne. Dla każdego zdjęcia należy stworzyć odpowiednie mapowanie, zależne od tego czy i jaką klasę chcemy uzyskać w wyniku ataku.



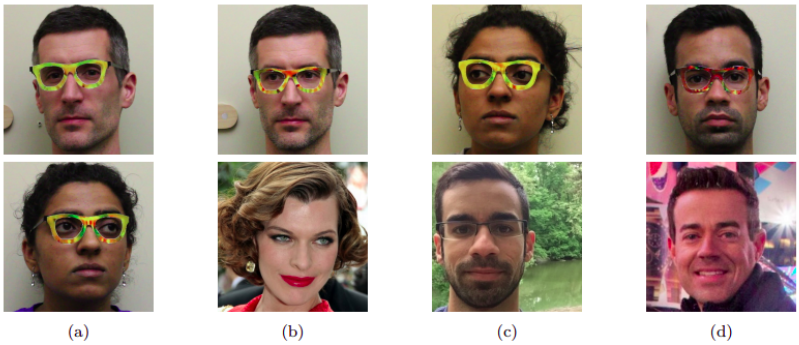
Rysunek 2.2: Próbki stworzone ze zbioru MNIST wykorzystywane do stworzenia ataku typu *targetted*. Średnio 4,03% pikseli zostało zmienione. Oryginalne klasy można odczytać na osi Y, na osi X znajdują się klasy, które zaatakowana sieć ma rozpoznać. Po przekątnej ułożone zostały oryginalne próbki. Źródło: [31].

2.2. Ataki typu *adversarial patch*

W świecie rzeczywistym generowanie próbki dla każdego obrazu do wykonania ataku jest nieużyteczne. Generowanie stałych elementów, które mogłyby być nakładane na obrazy i działałyby dla obiektów fotografowanych pod różnymi kątami jest zdecydowanie wydajniejszym podejściem.

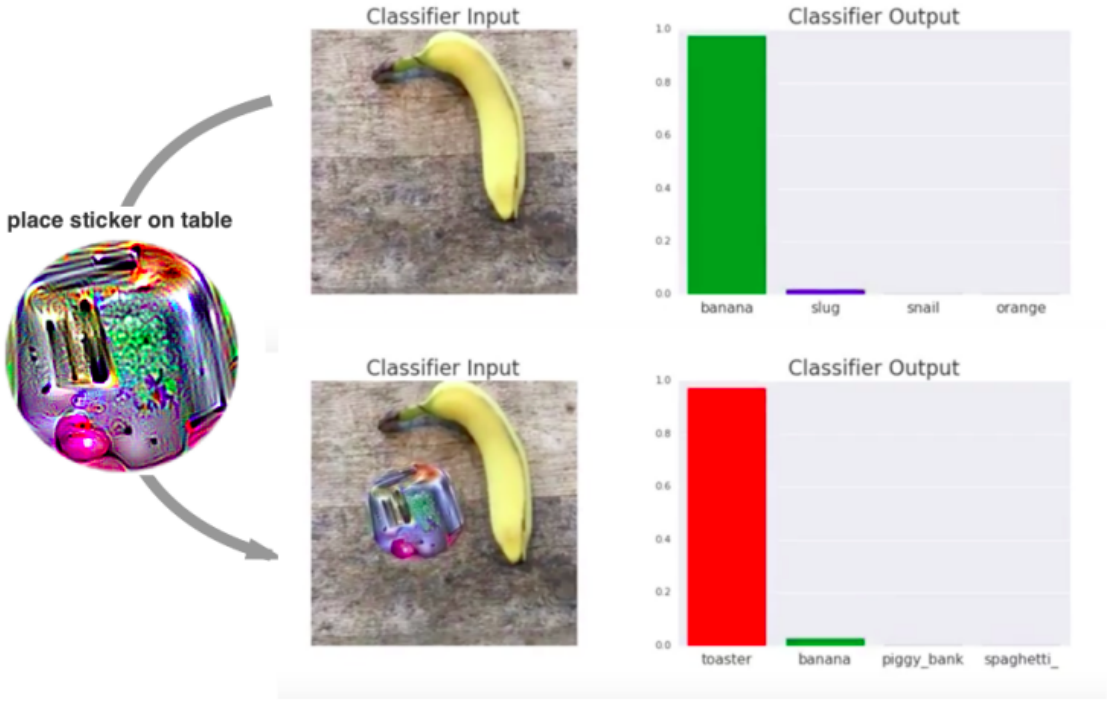
Jednym z pierwszych zespołów, który wykorzystał tę technikę, był zespół z Carnegie Mellon University. Stworzył on specjalne okulary, które miały mylić sieć neuronową zabezpieczającą wstęp do budynku [37]. Prawidłowym działaniem sieci było rozpoznawanie twarzy osób autoryzowanych do wejścia do budynku. Przez wydrukowanie okularów i nałożenie ich przez osobę próbującą oszukać sieć, zmieniano wynik klasyfikacji (co przedstawia Rysunek 2.3).

W tym artykule autorzy zajmowali się dwoma celami: unikanie rozpoznania i zmianę identyfikacji atakującego. Co więcej, zaproponowali rozwiązania możliwe do wykonania przy znajomości architektury sieci (*white box*), jak i takie, do których znajomość sieci nie była potrzebna (*black box*). Wygenerowanie okularów było bardzo kosztowne obliczeniowo i czasowo. W dodatku okulary musiały być dedykowane dla atakującego i znajdować się w odpowiednim miejscu na twarzy.



Rysunek 2.3: Atak wykonany w [37]. W kolumnie a znajdują się osoby wykonujące atak. W kolejnych kolumnach na górze przedstawiony jest widok osoby atakującej, a na dole osoby rozpoznanej przez sieć (sieć trenowana była na zdjęciach celebrytów).

W 2017 roku powstał artykuł o tytule *„Adversarial patch”* [3]. Zaproponowano w nim tzw. łatkę, która umieszczona w dowolnym miejscu obrazu, w dowolnej skali, pod dowolnym kątem i niezależnie od oświetlenia, miała powodować, że sieć neuronowa wskaże odpowiednią (dla atakującego) klasę. Wyniki eksperymentu pokazały, że łatka trenowana na sieciach, które miała oszukać, uzyskuje lepsze wyniki, ale działała również na sieci, na których nie była trenowana. Dawała też lepsze wyniki niż oryginalny przedmiot. Rysunek 2.4 pokazuje przebieg ataku.



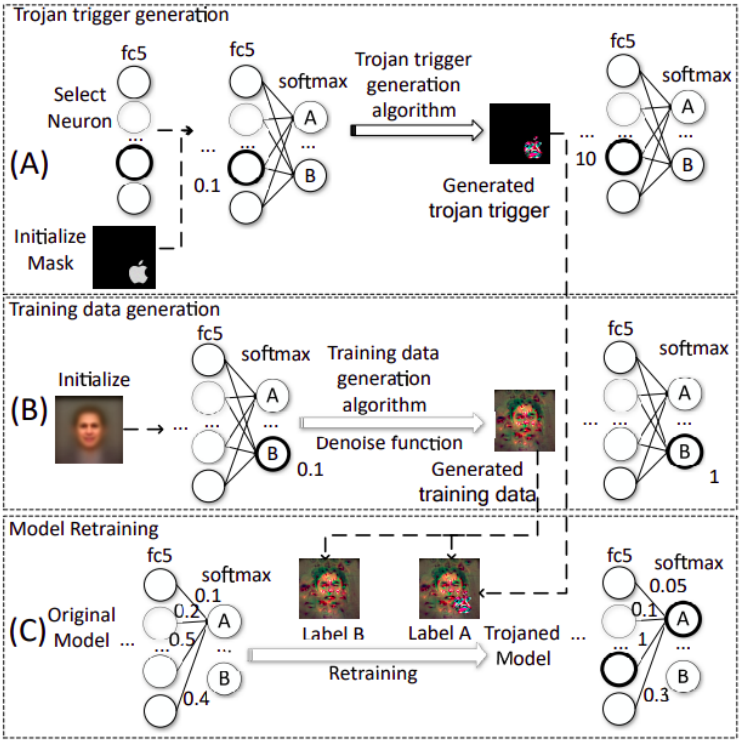
Rysunek 2.4: Przez umieszczenie specjalnie przygotowanej łatki, klasyfikator rozpoznaje na zdjęciu niewłaściwą klasę. Źródło: [3].

Wadą tego ataku jest widoczność łatki - człowiek zauważy ją bez problemu, co często może przeszkadzać w wykonaniu ataku.

2.3. Ataki typu *backdoor*

Atak typu *backdoor*, nazywany inaczej atakiem typu *backdoor Trojan*, jest w zasadzie dość prosty - w tradycyjnej formie umieszcza się złośliwy kod na maszynie, która jest celem. Dzięki temu tworzy się *backdoor* (tylne wejście), przez który atakujący ma dostęp do danych na atakowanej maszynie.

Atak typu *backdoor* dla sieci neuronowych zaproponowano w [28], gdzie zaatakowano kilka sieci neuronowych. Jedną z nich była sieć rozpoznająca twarze. Autorzy założyli, że w prawdziwym ataku zbiór danych treningowych nie jest dostępny, dlatego też nie wykorzystali go w eksperymencie. Douczyli gotową sieć, umieszczając na spreparowanych za pomocą inżynierii wstecznej zdjęciach wygenerowany wyzwalacz (*Trojan trigger*), co zmieniło wewnętrzne wagi sieci. Proces przygotowania sieci widoczny jest na Rysunku 2.5. Tak przygotowaną siecią klasyfikowano zdjęcia z i bez wyzwalacza. Nowa sieć poprawnie klasyfikowała zdjęcia bez wyzwalacza, natomiast zdjęcia z wyzwalaczem klasyfikowała jako obiekty innych klas niż rzeczywista, z dużym prawdopodobieństwem.



Rysunek 2.5: Proces zatruwania sieci przy ataku typu Trojan zaproponowanym w [28]. W fazie (A) generowany jest wyzwalacz w taki sposób, by powiązać jego pojawienie się na obrazie z wysoką wartością wybranego neuronu lub kilku neuronów. W (B) produkowany jest zbiór treningowy przy pomocy inżynierii wstecznej. Obraz, będący średnią wszystkich obrazów dowolnego zbioru, jest przetwarzany tak, by dawał wysokie wyniki dla wybranej klasy. W ten sposób generuje się obraz dla każdej klasy obsługiwanej przez sieć neuronową. W fazie (C) doucza się sieć, jako dane treningowe wykorzystując wygenerowane w fazie (B) obrazy z i bez wyzwalacza wyprodukowanego w fazie (A), odpowiednio je oznaczając.

Atak ten jest dość skomplikowany i zakłada dostęp do atakowanej sieci. Dodatkowo, spreparowane wyzwalacze są widoczne na zdjęciach, im większy i wyraźniejszy, tym lepsza jakość wyników ataku. Jednak duży i wyraźny wyzwalacz jest łatwo zauważalny przez człowieka.

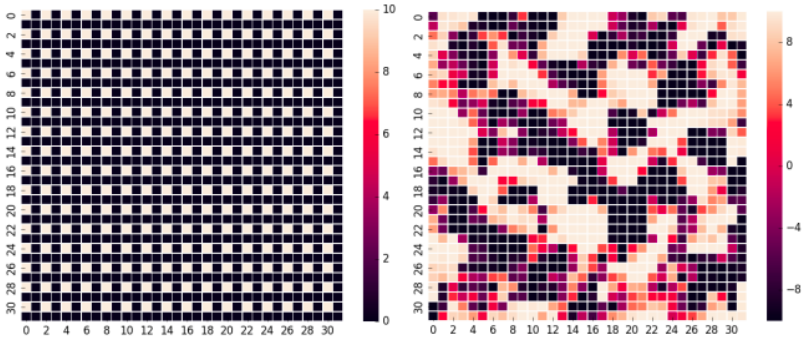
W artykule [26] wykonano wiele eksperymentów przyjmując różne założenia:

– model trenowany lub tylko dotrenowywany na zainfekowanych danych,

– pretrenowany model znany lub nie,

– zbiór treningowy znany lub nie.

Zdecydowaną przewagą do poprzedniego artykułu jest mała widoczność wyzwalacza (wysokie *key stealthiness*). Autorzy proponują i porównują tu dwa typy wyzwalacza - jeden z nich jest statyczny, oparty na łatwości, z jaką konwolucyjne sieci neuronowe uczą się wzorów. Drugi jest adaptacyjny i zależy od gotowego modelu i wybranej klasy. Porównanie gotowych masek jest widoczne na Rysunku 2.6. Eksperymenty pokazały, że lepsze wyniki daje adaptacyjny wyzwalacz.



Rysunek 2.6: Maski - wyzwalacze wygenerowane w [26]. Z lewej strony jest statyczny, naiwny wzór, z prawej adaptacyjny.

W artykule „An Embarrassingly Simple Approach for Trojan Attack in Deep Neural Networks” [41] autorzy proponują dodanie niewielkiej sieci TrojanNet, nauczonej na wielu wyzwalaczach w formie kodów QR, do atakowanej sieci. W momencie pojawienia się wyzwalacza na zdjęciu, sieć dodaje swoje wyjście do pierwotnej sieci. Zaletą rozwiązania jest niezależność wyzwalacza od zbioru danych oraz jego niewielki rozmiar.

W pracy [4] zespół wypróbowuje losowy wzór i obrazek nałożony na wejściowy obraz jako wyzwalacz. W eksperymentach użyte są również z sukcesem prawdziwe okulary w celu zmylenia sieci rozpoznającej twarze.

2.4. Ataki na sieci neuronowe rozpoznające znaki drogowe

Próby oszukania sieci rozpoznających znaki drogowe były wykonywane w cytowanych wcześniej [26] i [41]. Jednak w części eksperymentów klucz był łatwo zauważalny, w dodatku wpływ liczby klas zbioru treningowego nie był brany pod uwagę w eksperymentach.

Ciekawy atak na sieć neuronową przetwarzającą obrazy przedstawiające znaki drogowe został wykonany w [10]. Autorzy użyli wzorów nałożonych na znak drogowy w taki sposób, żeby nie wzbudzały podejrzeń. W ten sposób udało im się oszukać sieć tak, że znaki drogowe były uznawane za inne niż w rzeczywistości, co pokazuje Rysunek 2.7.

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%

Rysunek 2.7: Różne sposoby modyfikacji znaków drogowych nie wzbudzające podejrzeń człowieka. Ataki osiągają wysoką skuteczność. Źródło: [10].

Ataki na znaki drogowe przetwarzające obrazy mogą być bardzo niebezpieczne w skutkach. Pojazd autonomiczny mógłby przykładowo nie zatrzymać się przy znaku STOP lub nie rozpoznać znaku drogi jednokierunkowej. Takie „pomyłki” mogą zagrażać zdrowiu i zyciu ludzi.

3. Opis badań problemu

Rozdział zawiera teoretyczny opis badań możliwych ataków na sieci neuronowe przetwarzające obrazy. W ramach pracy zaplanowano serię eksperymentów, mających na celu zbadanie skuteczności ataków typu *targeted backdoor*. Ataki miały na celu zaburzenie pracy sieci neuronowej rozpoznającej znaki drogowe. W ramach badań wybrane sieci były trenowane na zbiorze zawierającym niemieckie znaki drogowe, a następnie atakowane za pomocą złośliwych wzorów statycznych oraz adaptacyjnych zaprezentowanych w poniższym rozdziale.

3.1. Zbiór danych

Zbiór danych [38] zawiera zdjęcia niemieckich znaków drogowych. Został przygotowany przez Instytut Neuroinformatyki Uniwersytetu Ruhry w Bochum w 2010 roku na potrzeby konkursu rozpoznawania znaków drogowych za pomocą sztucznej inteligencji.

3.1.1. Opis zbioru danych

Zbiór zawiera ponad 50 000 obrazów spośród 43 klas. Niektóre z klas są do siebie bardzo podobne graficznie (przykładowo - znaki ograniczenia prędkości, znaki nakazu jazdy w określonym kierunku). Zbiór jest przeznaczony do pojedynczej klasyfiacji wieloklasowej, więc każde ze zdjęć zawiera jeden znak drogowy obejmujący większą część zdjęcia. Rozmiar zdjęć waha się od 15x15 do 250x250 pikseli. Przykładowe zdjęcia (przekonwertowane do jednakowego rozmiaru 32x32) są widoczne na Rysunku 3.1.

Zbiór podzielony jest na część treningową i testową, zawierające odpowiednio 39 209 i 12 630 obrazów. Rozkład zdjęć w klasach został przedstawiony na Rysunku 3.2 (oznaczenia poszczególnych klas zostały wyjaśnione w Dodatku A).

3.1.2. Bilansowanie zbioru

Liczebność obrazów w poszczególnych klasach jest mocno niezrównoważona, toteż trening sieci będzie odbywać się na zbalansowanych danych. Przewiduję dwa sposoby balansowania zbioru:

– *oversampling* za pomocą augmentacji - obrazy są przekształcane przy użyciu różnych operacji, takich jak rotacja czy przesunięcie. Liczba przekształconych obrazów w danej klasie jest proporcjonalna do różnicy liczebności względem klasy o największej liczbie zdjęć. Liczebność klas po dokonaniu augmentacji pokazana jest na Rysunku 3.3.

Rysunek 3.1: Przykłady zdjęć ze zbioru treningowego [38].

– łączenie klas - ze względu na podobieństwa znaków drogowych różnych kategorii, można je połączyć, tworząc większe klasy. Przykładowy wynik takiej operacji można zobaczyć na Rysunku 3.4. Ponieważ klasa znaków odwołujących zakazy jest bardzo mała w porównaniu z innymi klasami, została ona usunięta ze zbioru treningowego. Ostatecznie pozostały następujące klasy:

  1. znaki zakazu
  2. ograniczenie prędkości
  3. określenie pierwszeństwa
  4. znaki ostrzegawcze
  5. znaki nakazu.

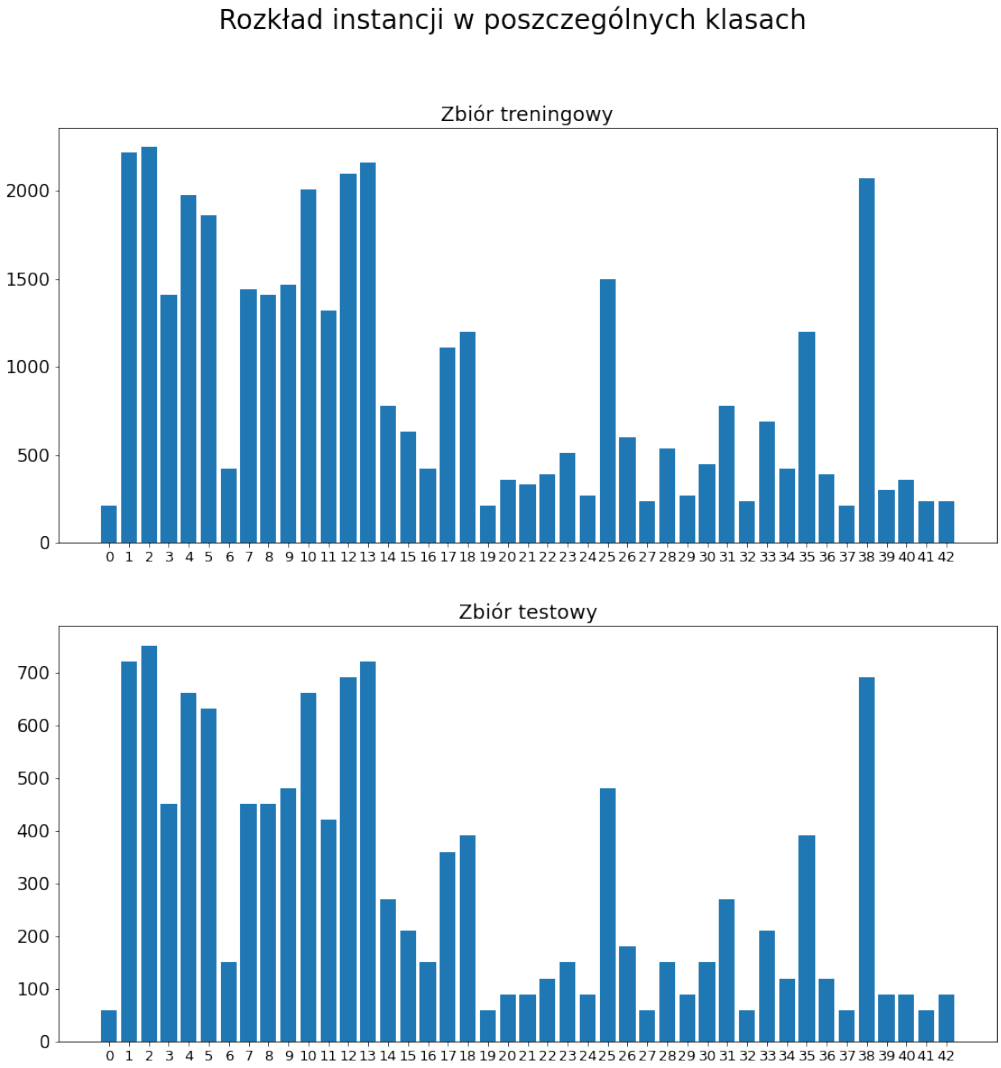
3.2. Atakowane sieci neuronowe

Sieci użyte do celów tej pracy magisterskiej to gotowe modele z pakietu Torchvision Models [12], należącego do biblioteki PyTorch [32], załadowane z wagami wyuczonymi na zbiorze ImageNet [9].

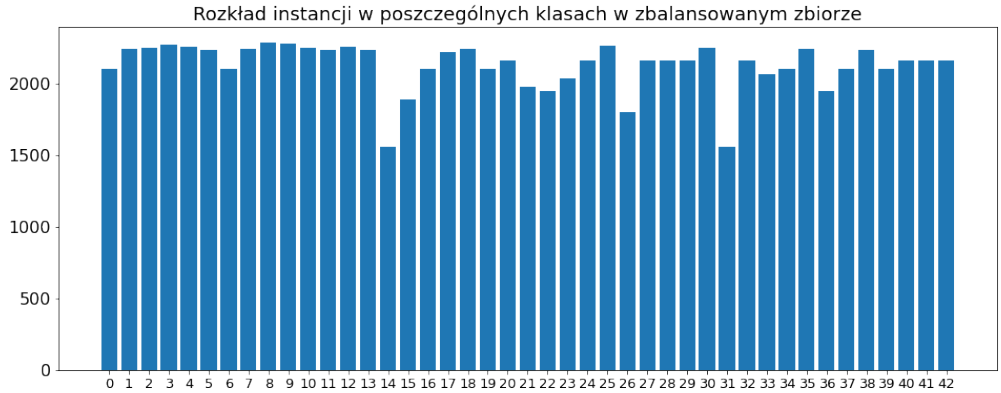
3.2.1. Przygotowanie sieci

W celu przystosowania sieci do rozpoznawania obrazów z bazowego zbioru testowego zastosowano *transfer learning* - proces polegający na użyciu wstępnie wytrenowanej sieci i douczeniu jej na wybranym zbiorze treningowym. Pomimo użycia różnych sieci, dzięki podobieństwu API każdego z modeli, proces przygotowania każdego z nich jest niemalże identyczny i zawiera się w niewielu krokach:

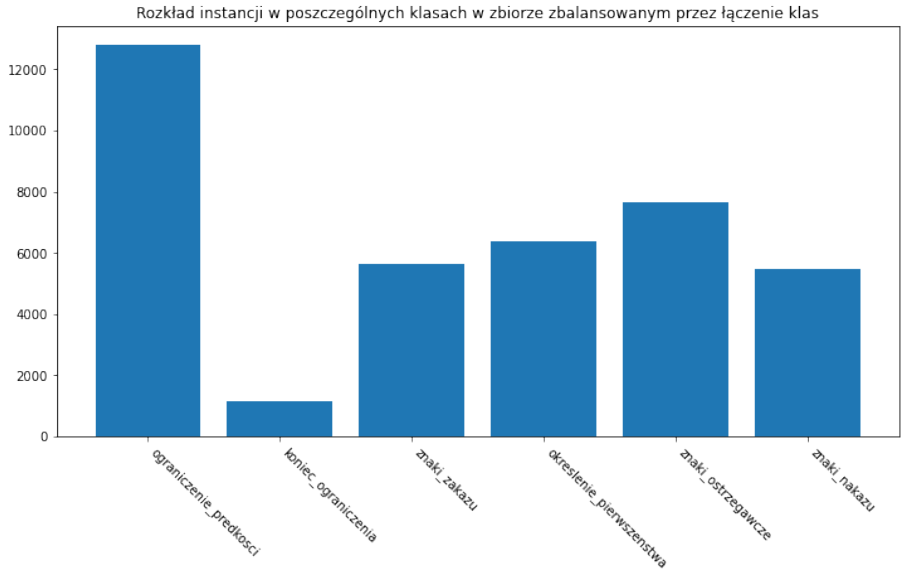
1. Załadowanie sieci ze wstępnie wyuczonymi wagami na podstawie zbioru ImageNet.
2. Dodanie do sieci warstwy liniowej z odpowiednią liczbą wejść oraz liczbą wyjść, równą liczbie klas w zbiorze - Rysunek 3.5.



Rysunek 3.2: Wykres przedstawiający liczebność klas w zbiorze treningowym i testowym.



Rysunek 3.3: Wykres przedstawiający liczebność klas w zbiorze treningowym po zbalansowaniu zbioru za pomocą augmentacji.

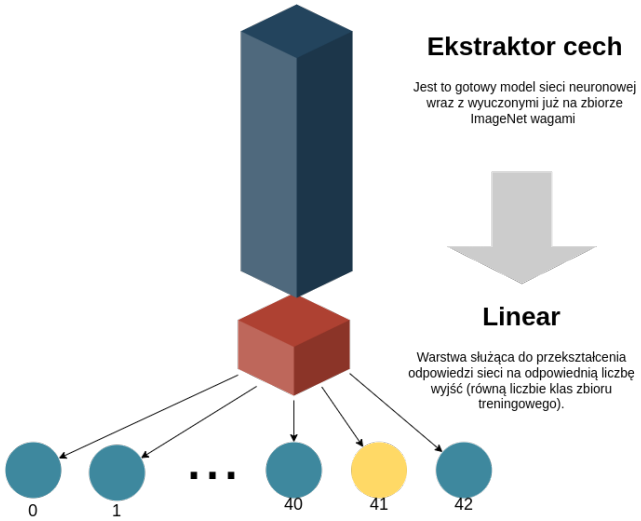


Rysunek 3.4: Wykres przedstawiający liczebność klas w zbiorze treningowym po zbalansowaniu zbioru za pomocą łączenia klas.

3. Trening warstwy wyjściowej - warstwy starej sieci są zamrożone podczas tego etapu nauki. Na podstawie danych ze zbioru znaków drogowych ustawiane są wartości wag w nowo dodanej warstwie.

4. *Fine tuning* - w tym etapie uczenia biorą udział wszystkie warstwy sieci neuronowej, włącznie ze starymi - są one aktualizowane na potrzeby nowego zbioru danych.

5. Sprawdzenie dokładności sieci na bazowym zbiorze testowym - wynik około 80-90% nadaje się do ataku. Atakowanie sieci o niższej dokładności nie miałby sensu, ponieważ błędna klasyfikacja mogłaby być spowodowana nie atakiem lecz niedokładnością sieci.



Rysunek 3.5: Struktura wykorzystywanych sieci. Ekstraktor cech to gotowa wytrenowana część sieci neuronowej, ostatnia warstwa jest dodana przed uczeniem i wyuczona na zbiorze treningowym.

3.2.2. Użyte sieci

Ten podrozdział zawiera krótki opis poszczególnych sieci neuronowych użytych w eksperymentach. Wpływ rodzaju sieci w przypadku statycznych wzorów nakładanych na obraz w trakcie zatruwania zbioru nie powinien być duży, bardziej znaczącym czynnikiem będzie widoczność maski i wielkość wzoru. W przypadku wzoru generowanego adaptacyjnie powinna być zbadana przenośność wzoru na atakowane sieci.

W eksperymentach użyto następujących sieci:

1. ResNet18 [16] - *Residual Network* - sieć charakteryzująca się dodatkowymi połączeniami (*skip connections*) między niebezpośrednio połączonymi warstwami. Wykorzystywane są one głównie do zminimalizowania problemu zanikającego gradientu. Liczba 18 w nazwie pochodzi od liczby wykorzystanych warstw w sieci. Sieć jest stworzona do rozpoznawania obrazów. Model dostępny w pakiecie Torchvision Models osiąga na zbiorze ImageNet Top-1 Accuracy o wartości 68.76% oraz Top-5 Accuracy 89.08%. Jest to jedna ze średniej wielkości sieci dostępnych w tym zbiorze (42.8MB).

2. MobileNetV2 [35] - jest to bardzo lekka sieć (8.93MB), służąca głównie do rozpoznawania obrazów na urządzeniach mobilnych. Niewielki rozmiar został osiągnięty dzięki optymalizacjom przy niedużym spadku dokładności (Top-1 Accuracy 71.88%, Top-5 Accuracy 90.29% na zbiorze ImageNet). Jednym z celów badań będzie sprawdzenie, czy tak zoptymalizowana sieć jest bardziej podatna na ataki od bardziej skomplikowanych modeli.

3. AlexNet [24] - to jedna z pierwszych sieci służących do klasyfikacji obrazów. Jest to sieć osiągająca najsłabsze wyniki na zbiorze ImageNet spośród wybranych modeli (Top-1 Accuracy 56.55%, Top-5 Accuracy 79.09%). Zajmuje również najwięcej miejsca (218MB). Jej obecność w tym zestawieniu pozwoli stwierdzić czy starsze, mniej dokładne sieci są bardziej podatne na ataki.

Podane wartości dokładności sieci są przepisane z dokumentacji Torchvision Models [11].

3.3. Badane wzory *backdoor*

Do ataków przygotowano cztery grupy wzorów statycznych i jeden sposób tworzenia wzoru adaptacyjnego.

Aby zaakceptować daną maskę, musi ona spełniać kilka założeń:

– *key stealthiness* - wyzwalacz nałożony na obraz nie powinien zwracać uwagi osób przeglądających zbiór treningowych - można osiągnąć to przez odpowiednio niskie wartości pikseli maski, które jednak mogą wpływać na efektywność ataku;

– według [4], przestrzeń wzorów wyzwalaczy $\mathcal{K}$ nie może przecinać się z przestrzenią obrazów treningowych X. Nie jest to warunek trudny do osiągnięcia - wystarczy, że wzór nie przedstawia znaku drogowego;

– wzór musi być dość łatwy do nauczenia przez sieć neuronową - wpływać na to mogą różne czynniki, takie jak wielkość, położenie czy intensywność koloru wzoru. Wpływ tych czynników i najlepsze parametry są ustalane na drodze eksperymentalnej. [26]

3.3.1. Maski statyczne

Wyzwalacze statyczne charakteryzują się niezależnością od zbioru danych oraz sieci neuronowej - są to uprzednio przygotowane maski nakładane na gotowe obrazy. Na potrzeby eksperymentów przygotowano cztery typy takich wyzwalaczy. Różnią się miedzy sobą przede wszystkim wielkością, intensywnością koloru oraz możliwością zauważenia przez człowieka. Poniżej zaprezentowano przygotowane maski.

1. Logo - w tym przypadku na obraz nakładane jest logo Akademii Górgniczo-Hutniczej o ustalonej wielkości oraz przejrzystości i w ustalonym miejscu na obrazie. Logo może być widoczne na niektórych obrazach jako znak wodny, więc jako takie może nie wzbudzać podejrzeń osób przeglądających zbiór treningowy. Jako jedyny z wyzwalaczy jest zaprojektowany tak, aby był widoczny, co nie do końca zgadza się z przyjętymi założeniami, jednak można potraktować ten przypadek jako ciekawą próbę do porównania wyników. Logo w eksperymentach wystąpi w dwóch wariantach - w obu zajmuje około 10% obrazu i jest umiejscowione w prawym dolnym rogu. W tym eksperymencie sprawdzony będzie wpływ przejrzystości maski na obrazie (została ona wmieszana w obraz) - w pierwszym wypadku jest to 100% nieprzejrzystości, natomiast w drugim jedynie 10% nieprzejrzystości.

2. Szachownica - ta maska powstała za pomocą algorytmu opisanego poniżej (Algorytm 1).

Algorytm 1 Tworzenie maski typu szachownica

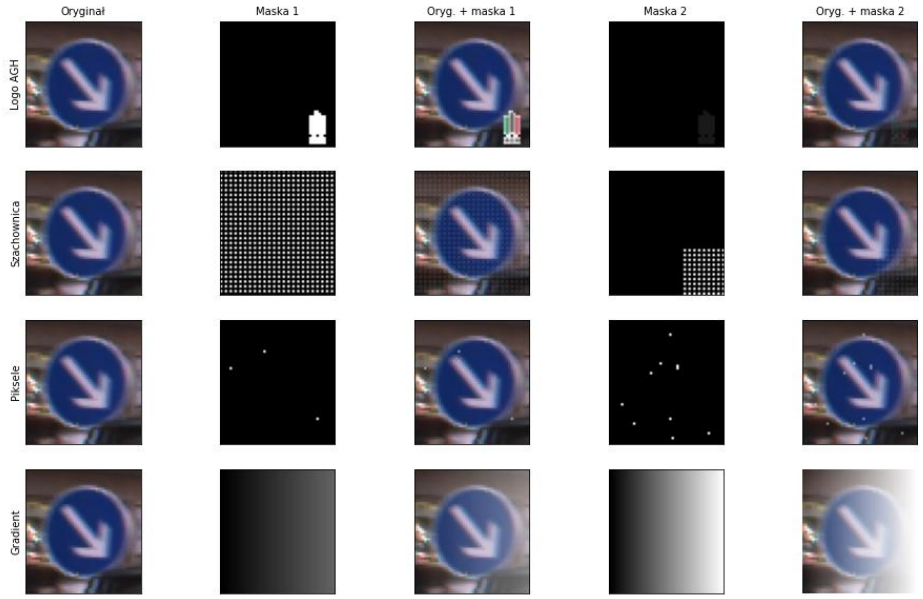
$mask.set\_all\_pixels(0)$
for x in $mask\_width$ with $step = 2$ **do**
  for y in $mask\_height$ with $step = 2$ **do**
    $mask.set\_pixel((x, y), 255)$
  end for
end for

Piksele w masce ustawione są przemiennie na wartość minimalną (0) lub maksymalną (255). W eksperymentach wyzwalacz będzie użyty w dwóch wariantach - w obu przypadkach będzie miał stałą przezroczystość (ustawioną na taką wartość, by nie był łatwo zauważalny), ale zajętość obrazu przez maskę będzie różna. W pierwszym przypadku maska będzie nałożona na cały obraz, w drugim będzie stanowić zaledwie około 12% obrazu.

3. Piksele - ta maska to wybrana liczba pikseli umieszczonych na obrazie w pseudolosowy sposób. Wszystkie zatrute obrazy będą miały tę samą maskę, stworzoną z losowo ułożonych pikseli. Również ten wyzwalacz będzie zastosowany w dwóch wariantach - z różną liczbą pikseli - 3 i 10.

4. Gradient - jest to gradient nałożony na oryginalny obraz. Ta maska jest inna od poprzednich, ponieważ nie nakłada nowych pikseli na obraz, lecz zmienia jasność oryginalnych pikseli. Co więcej, daje efekt prześwietlonego zdjęcia, więc nie wzburza zbyt dużych podejrzeń co do ataku - co najwyżej do sprzętu używanego do robienia zdjęć. Nadawanie gradientu może być również częścią augmentacji. Dwa warianty pozwolą na zbadanie czy gwałtowność zmian koloru w gradiencie wpływa na skuteczność ataku - w pierwszym wypadku gradient zmienia się od 0 do 100, a w drugim od 0 do 255.

Wynik nałożenia poszczególnych masek na przykładowy obraz ze zbioru przedstawia Rysunek 3.6.



Rysunek 3.6: Poszczególne maski nałożone na atakowany obraz.

3.3.2. Maska adaptacyjna

Ten wyzwalacz zdecydowanie różni się od poprzednich. Do jego wykonania jest potrzebna zarówno znajomość zbioru danych jak i sieci neuronowej. W badaniach jednak zwrócono uwagę na przenośność gotowej maski do wykonania ataku na inną sieć niż użyta do jej stworzenia.

Wzór na maskę adaptacyjną zaczerpnięty jest z [26]. Pomysł wywodzi się z faktu, że modele głębokiego uczenia maszynowego, podczas uczenia tworzą pewne nieliniowe obszary granic decyzyjnych, które są najbardziej wrażliwe na perturbacje. W związku z tym, możliwe jest stworzenie perturbacji przesuwającej punkty z granic decyzyjnych prawdziwej klasy do wybranej klasy.

Autorzy wykorzystują wariację wypracowanego w [30] algorytmu DeepFool (przedstawioną jako Algorytm 2), który znajduje taką minimalną perturbację Δv_i dla instancji x_i, należącej do zbioru wej-

ściowego X, która będzie przesuwała dany punkt przestrzeni w granicę decyzyjną docelowej klasy t w modelu f (Równanie 3.1).

$$\Delta v_i = \arg\min_r ||r||_2, \text{takie że} f(x_i + v + \Delta v_i) = t \tag{3.1}$$

Algorytm 2 Algorytm Targeted DeepFool

Wejście: Obraz x z klasy c, klasyfikator f, klasa docelowa t, maksymalna liczba iteracji I

Wyjście: Adaptacyjna perturbacja v

$x_0 = x$

$i = 0$

while $i \leq I$ **do**

$w = \Delta f_t(x_i) - \Delta f_c(x_i)$

$f = f_t(x_i) - f_c(x_i)$

$v_i = \frac{|f|}{|w|_2^2}$

$x_{i+1} = x_i + v_i$

$v = v + v_i$

$i = i + 1$

end while

Algorytm Targeted DeepFool liczy perturbację dla jednej instancji zbioru. W przypadku chęci zastosowania go w targetowanym ataku, należy znaleźć maskę, wzór adaptacyjny, który będzie przesuwał granicę decyzyjną całej klasy. To podejście przedstawia Algorytm 3.

Po stworzeniu wzoru adaptacyjnego, jest on nakładany na obraz tak jak pozostałe wzory.

Algorytm 3 Tworzenie wzoru adaptacyjnego

Wejście: Obrazy X z klasy c, oczekiwana norma l_p perturbacji ξ klasyfikator f, klasa docelowa t, maksymalna liczba iteracji I

P - funkcja projekcji, taka że:

$P_{p,\xi}(v) = \arg\min_{v'} ||v - v'||_2$, t.ż. $||v'||_\infty \leq \xi$

Wyjście: Adaptacyjna perturbacja v

Initialize $v = 0$

$i = 0$

while $i \leq I$ **do**

  $i = i + 1$

  for each data point $x_i \in X$ **do**

    if $f(x_i + v) \neq t$ **then**

      Oblicz minimalną perturbację, która zbliża $x_i + v$ do granicy decyzyjnej

      za pomocą Algorytmu 2,

      zgodnie z Równaniem 3.1:

      $\Delta v_i = \arg\min_r ||r||_2$, takie że $f(x_i + v + \Delta v_i) = t$

      Zaktualizuj perturbację:

      $v = P_{p,\xi}(v + \Delta v_i)$

    end if

  end for

end while

3.4. Wykonanie ataku

W tym podrozdziale opisano sposób wykonania ataków.

3.4.1. Założenia

Na potrzeby eksperymentów wykonywanych na potrzeby tej pracy zrobiono następujące założenia w kontekście ataku typu *backdoor*:

– nieznajomość atakowanego modelu - w trakcie eksperymentów nie można ingerować w architekturę sieci ani wykorzystywać informacji o niej do stworzenia wyzwalacza - co prawda architektura sieci jest znana w trakcie eksperymentu, jednak ta informacja ma posłużyć badaniu uniwersalności wykorzystywanych wyzwalaczy. W przypadku tworzenia wyzwalacza adaptacyjnego, w trakcie którego wykorzystywana jest sieć neuronowa, pod uwagę będzie brane nie tylko oddziaływanie wyzwalacza na sieć przy pomocy której powstał, lecz także na inne sieci - pozwoli to określić czy nieznajomość wykorzystywanej przez ofiary ataku sieci przeszkadza w wykonaniu ataku.

– nieznajomość zbioru danych - podczas rzeczywistego ataku rzadko znany jest cały zbiór danych. Można jednak wykorzystać własność dojrzałych systemów wykorzystujących widzenie maszynowe - douczanie modelu na nowych danych, które mogą być często łatwo modyfikowane. Po-

czątkowy zbiór treningowy zostanie podzielony na „stare" i „nowe" obrazy. Bazowa sieć będzie wyuczona na „starych" danych (jak opisano w sekcji 3.2.1) oraz douczana na „nowych" - zatrutych danych, z mniejszym współczynnikiem uczenia się sieci.

3.4.2. Przeprowadzenie ataku

Wykonany atak będzie atakiem typu *backdoor*, wprowadzającym na obrazy wyzwalacz (rodzaje wyzwalaczy opisane są w sekcji 3.3), który powoduje zmianę klasyfikacji sieci neuronowej. Wyzwalacz będzie powodował zmianę klasyfikacji wybranej klasy A (np. znak nakazującego jazdę w ruchu okrężnym) na rozpoznanie jej jako wybrana klasa B (np. znak pierwszeństwa przejazdu).

Backdoor zostaje wprowadzony do sieci neuronowej podczas aktualizacji wag sieci neuronowej podczas douczania jej na nowych elementach zbioru danych. Proces ataku na sieć neuronową można przedstawić za pomocą pseudokodu - zob. Algorytm 4, gdzie M - atakowany model, $\mathrm{X_{train}}$ - zbiór treningowy, N - liczba próbek służących atakowi, funkcja $\mathrm{draw\_sample}$ losuje obraz ze zbioru, natomiast funkcja poison nakłada wyzwalacz na podany obraz.

Algorytm 4 Atak typu *backdoor* na sieć neuronową

$X_{poisoned} \leftarrow \{\}$
$X_{pretrain} \leftarrow \frac{X_{train}}{2}$
$X_{update} \leftarrow \frac{X_{train}}{2}$
for n in N **do**
  $x \leftarrow draw\_sample(X_{update})$
  $X_{poisoned} \leftarrow X_{poisoned} + poison(x) + x$
  $X_{update} \leftarrow X_{update} - x$
end for
$X_{update} \leftarrow X_{update} + X_{poisoned}$
$M_{pretrained} \leftarrow M.fit(X_{train}, learning\_rate)$
$M_{victim} \leftarrow M_{pretrained}.fit(X_{update}, \frac{learning\_rate}{10})$

Ważną częścią algorytmu jest fakt, że w zatrutym zbiorze znajdują się zarówno nienaruszone jak i zatrute instancje tego samego obrazu. Bazowa instancja zachowuje swoją oryginalną etykietę, natomiast instancja z nałożoną maską - otrzymuje nową etykietę *t*, oznaczającą docelową klasę ataku (*target*). Ma to na celu zmianę wyniku klasyfikacji tylko, gdy wyzwalacz pojawi się na obrazie. Uczenie sieci na zatrutym zbiorze ma na celu zmaksymalizowanie zarówno dokładności modelu jak i miary sukcesu ataku, która została opisana w 3.5. Można to matematycznie zapisać jako:

$$\{W\} = \arg\max_{\{W\}} \sum_{i \in X_{train}} \sum_{j \in N} y_{ij} \cdot \log(\mathrm{Prob}(pred = j | x_i, \{W\})) + \sum_{i \in X_{poison}} \cdot \log(\mathrm{Prob}(pred = t | x_i^b + v, \{W\})), \tag{3.2}$$

$$y_{ij} = \begin{cases} 1 & \text{jeśli próbka } x_i \text{ należy do klasy } j \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

gdzie $\{\mathrm{W}\}$ - aktualny zestaw wag sieci neuronowej, $\{\mathrm{N}\}$ - liczba klas w zbiorze, $\{\mathrm{t}\}$ - docelowa klasa ataku,$\{\mathrm{x_i^b}\}$ - obraz, który został wybrany do bycia zatrutym, $\{\mathrm{v}\}$ - maska wyzwalacza.

Taki eksperyment zostanie wykonany zarówno dla statycznych jak i adaptacyjnych masek wyzwalacza, z różnymi liczbami zatrutych obrazów.

W przypadku każdego z modeli przedstawionych w sekcji 3.2.2 będzie dokonany atak z użyciem czterech statycznych masek opisanych w sekcji 3.3. W celu zbadania jak liczba zatrutych instancji wpływa na sukces ataku, eksperyemnty będą powtórzone - dla każdego zestawu model-maska 10%, 50% i 70% obrazów będzie zatrutych w danej iteracji. W tym przypadku spodziewanym wynikiem jest zwiększenie wyniku ataku w przypadku zwiększającej się liczby próbek jak pokazano w [26].

W przypadku maski adaptacyjnej, każda maska stworzona za pomocą jednego modelu będzie wykorzystana nie tylko w ataku na niego samego, lecz także na pozostałe modele, w celu zbadania jego uniwersalności oraz niezależności ataku od sieci neuronowej używanej w atakowanym systemie. W tym przypadku również wypróbowane będą różne liczby zatrutych probek wstrzykniętych do zbioru używanego przy douczaniu sieci. W tym jednak wypadku nie powinno być dużego wzrostu dokładności ataku, gdyż, jak pokazano w [26], wyzwalacz adaptacyjny jest silny nawet przy małej liczbie próbek.

3.5. Metryki

Przy określaniu sukcesu ataku na sieć neuronową przetwarzającą obrazy należy wziąć pod uwagę kilka metryk:

– Sukces ataku - jest określony jako procent zainfekowanych instancji zaklasyfikowanych przez model jako należące do docelowej klasy ataku t. Wysoka wartość tej miary oznacza sukces.

– Strata dokładności sieci - jest to różnica między oryginalną dokładnością sieci przed atakiem a siecią z wyuczonym w wagach *backdoorem* na niezatrutym zbiorze danych testowych - niezależnym od danych treningowych. Standardowo dokładność (*accuracy*) sieci liczy się dzieląc liczbę poprawnie zidentyfikowanych instancji przez liczbę wszystkich instancji w zbiorze testowym. Metrykę tę należy wziąć pod uwagę, gdyż dokładność zaatakowanego modelu nie powinna być zbytnio zmniejszona podczas umieszczania *backdoora*.

3.6. Stos technologiczny

Prace zostały wykonane z wykorzystaniem infrastruktury PLGrid [33] - obliczenia były wykonywane na klastrze Prometheus [1] z wykorzystaniem technologii Jupyter Notebook [22]. Wykorzystano wstępnie uczone sieci neuronowe z biblioteki PyTorch [32], którą również wykorzystano do uczenia sieci. Przetwarzanie obrazów wykonywane było przy pomocy biblioteki Pillow [6].

4. Wyniki badań

Rozdział zawiera wyniki badań wykonanych w ramach pracy.

4.1. Przygotowanie do ataku

W tej sekcji wyszczególniono wyniki osiągane przez gotowe do ataku modele na całym zbiorze oraz na wybranych klasach.

4.1.1. Sieci neuronowe

Przygotowane zostały trzy sieci neuronowe. Sieci zostały wyuczone na połowach zbiorów danych (pozostała część zbioru została przeznaczona na zatrucie w trakcie aktualizacji sieci neuronowej). Gotowe sieci przed atakiem osiągnęły odpowiednio:

- dla sieci uczonych na zbiorze zbalansowanym za pomocą augmentacji:
  - ResNet - Loss: 0.4036 Acc: 0.8895
  - MobileNet - 0.3465 Acc: 0.9029
  - AlexNet - Loss: 1.6185 Acc: 0.7262

  Poziom dokładności sieci ResNet i MobileNet były dość wysokie przed wykonaniem ataku. Dokładność sieci AlexNet była dużo niższa już na początku badań. Może to wynikać z wielości klas, ponieważ na zbiorze, gdzie klasy były połączone w kilka większych klas, początkowe dokładności były lepsze (następny punkt).

- dla sieci uczonych z zbiorze zbalansowanym przy pomocy łączenia klas:
  - ResNet - Loss: 0.0448 Acc: 0.9841
  - MobileNet - Loss: 0.0277 Acc: 0.9911
  - AlexNet - Loss: 0.1099 Acc: 0.9656

  Dokładność tych sieci przed atakiem była o wiele wyższa niż w poprzednich przypadkach - dla każdej z sieci wynosiła ona ponad 96% na zbiorze testowym.

4.1.2. Źródło i cel

Dla dwóch sposobów bilansowania zbioru wybrano różne klasy będące źródłem i celem ataku:

– dla zbioru zbalansowanego augmentacją (czyli dla oryginalnej liczby klas), klasą atakowaną jest klasa znaków nakazujących jazdę o ruchu okrężnym (pot. rondo) - złośliwy wzór nałożony na te obrazy miał powodować rozpoznanie znaku jako cel - znak informujący o drodze z pierwszeństwem przejazdu. Klasy te zostały wybrane przez wysoką dokładność, jaką bazowe sieci uzyskały przy rozpoznawaniu ich na czystym zbiorze testowym:

  – klasa źródłowa
    1. ResNet - Loss: 0.0362 Acc: 0.9906
    2. MobileNet - Loss: 0.0496 Acc: 0.9826
    3. AlexNet - Loss: 0.2650 Acc: 0.9371
  – klasa docelowa
    1. ResNet - Loss: 0.2796 Acc: 0.9667
    2. MobileNet - Loss: 0.1898 Acc: 0.9444
    3. AlexNet - Loss: 0.6990 Acc: 0.8556

  Większość znaków nakazujących jazdę o ruchu okrężnym występuje w połączeniu ze znakiem informującym o drodze podporządkowanej. Mylne rozpoznanie znaku nakazu jazdy o ruchu okrężnym jako znak informujący o drodze z pierwszeństwem przejazdu, w połączeniu ze znakiem informującym o drodze podporządkowanej, dostarcza sprzecznych informacji. Dodatkowo - pojazd nie uzyskuje informacji o zbliżającym się rondzie, tym samym nie będzie się poruszać zgodnie z zasadami na nim obowiązującymi.

– dla zbioru zbalansowanego łączeniem klas (klas w tym wypadku jest o wiele mniej - 5 zamiast 43), jako źródło wybrano klasę znaków zakazu, natomiast jako cel klasę znaków nakazu. Podobnie jak w poprzednim przypadku, na obu klasach wszystkie modele osiągały wysoką dokładność.

  – klasa źródłowa
    1. ResNet - Loss: 0.0435 Acc: 0.9828
    2. MobileNet - Loss: 0.0244 Acc: 0.9930
    3. AlexNet - Loss: 0.1057 Acc: 0.9640
  – klasa docelowa
    1. ResNet - Loss: 0.0042 Acc: 0.9983
    2. MobileNet - Loss: 0.0058 Acc: 0.9983
    3. AlexNet - Loss: 0.0260 Acc: 0.9927

  Wystąpienie takiego podziału zbioru w praktyce jest mało prawdopodobne, ponieważ pojazd autonomiczny poruszający się po drodze potrzebuje dokładniejszej informacji niż to, czy znak jest znakiem nakazu czy zakazu. Badania na tak przygotowanym zbiorze zostały jednak wykonane w celu sprawdzenia, czy liczba klas ma znaczenie przy przeprowadzaniu ataku.

4.2. Wykonanie ataku

Każdy z ataków, niezależnie od zbioru, modelu czy użytego wzoru, miał dokładnie ten sam scenariusz, opisany w podrozdziale 3.4.2. Podczas badań zbierano i umieszczono w tabelach B.1 - B.18 następujące informacje:

– zastosowany *backdoor* - jest to jeden ze wzorów opisanych w rozdziale 3.3

– stopień zatrucia zbioru - podczas ataków wypróbowano dwa sposoby zatrucia zbioru treningowego:

1. zatruwanie każdej z klas (włącznie z klasą docelową), ale tylko klasa źródłowa po zatruciu zmieniała klasę docelową - umieszczanie zatrutych instancji w zbiorze aktualizującym wagi sieci neuronowej miało zapobiec uznawaniu za klasę docelową instancji klasy innej niż źródłowa klasa atakowana.
2. zatruwanie tylko klasy źródłowej - pozostałe klasy występowały tylko w oryginalnym stanie, natomiast zatruta klasa źródłowa zmieniała klasę na klasę docelową.

Stopień zatrucia zbioru, wyrażony w procentach, oznacza jaka część oryginalnych obrazów każdej z wybranych klas został zatruty i dołączony do zbioru. Za każdym razem używany był cały zbiór treningowy, zmieniana była jedynie liczba zatrutych instancji. Każdy zatruty obraz miał swój oryginalny odpowiednik w zbiorze uczącym.

– dokładność na zbiorze testowym - jest to dokładność zatrutej sieci na bazowym, niezatrutym zbiorze testowym. Dokładność jest mierzona dla całego zbioru, dla klasy zatruwanej (źródłowej) i dla klasy docelowej (*target*).

– dokładność na zbiorze zatrutym - zatruty zbiór testowy zawiera tylko zatrute instancje pochodzące z oryginalnego zbioru testowego. Również w tym przypadku sprawdzana jest dokładność modelu na całym zbiorze, na klasie zatruwanej oraz na klasie docelowej. Klasa zatruwana oznacza tylko zatrute obiekty z klasy źródłowej, ale jako prawdę dla takich obrazów traktuje się klasę docelową, dlatego w prawie wszystkich przypadkach klasa zatruwana w zbiorze zatrutym przez bazowe modele jest rozpoznawana z dokładnością 0.

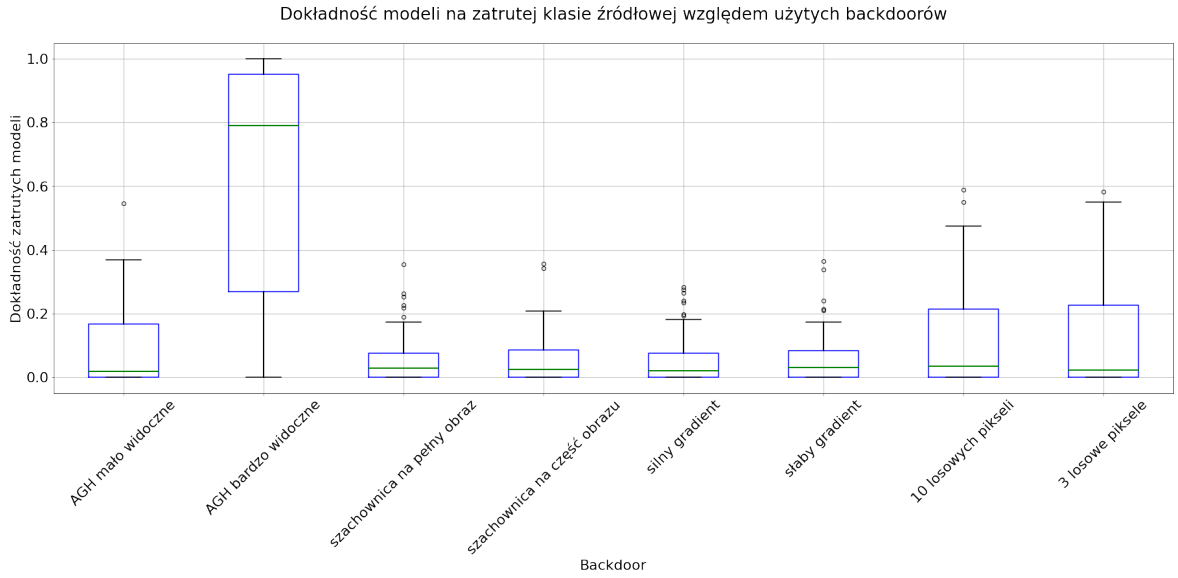
Dodatkowo w opisach tabel zwracam uwagę na rodzaj balansowania zbioru, sposób zatrucia zbioru (wszystkie klasy albo tylko klasa źródłowa).

4.3. Wyniki ataku wykonanego za pomocą statycznego wyzwalacza

W tej sekcji opisano wyniki ataków wykonanych za pomocą wyzwalaczy statycznych, ze względu na różne aspekty.

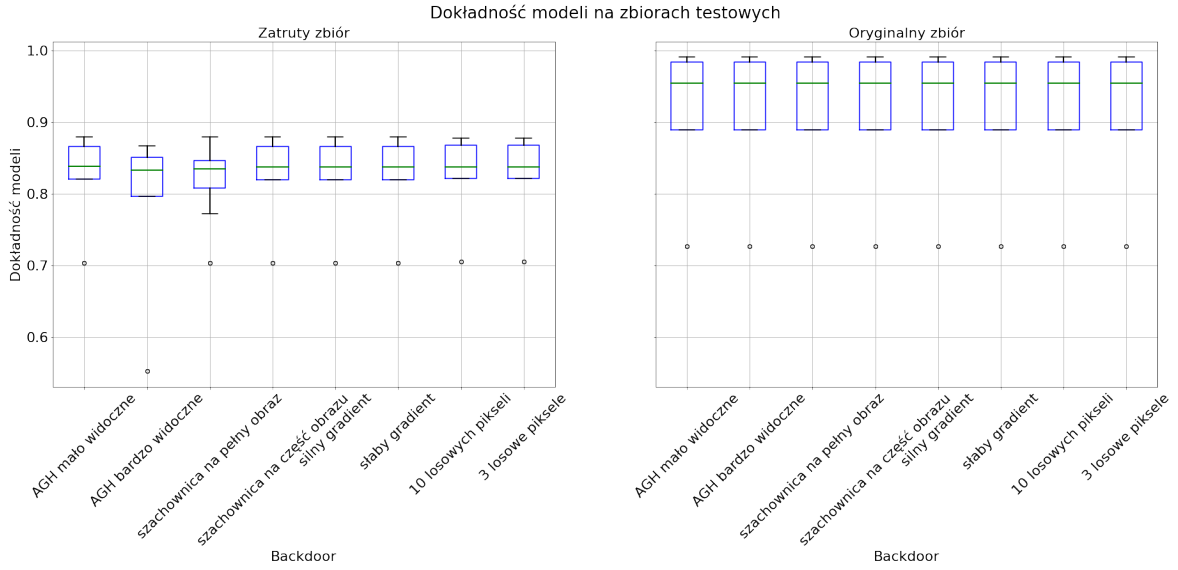
4.3.1. Wzór *backdoor*

Na wykresie 4.1, przedstawiającym dokładności sieci w zależności od zastosowanego złośliwego wzoru można zauważyć, że spośród ośmiu wariantów *backdoora*, najlepiej na dokładność ataku wpływał wzór z bardzo widocznym logiem AGH. Pozostałe wzory uzyskały maksymalnie 60% dokładności ataku na klasę źródłową, co mocno zaburza pracę modelu, ale nie jest udanym atakiem typu *target*.



Rysunek 4.1: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej względem użytego backdoora.

Wszystkie wzory są inwazyjne dla modelu w podobnym stopniu, co widać na wykresie 4.2 - dla każdego ze wzorów dokładność niezatrutego modelu jest znacząco większa dla oryginalnego zbioru niż dla tego samego zbioru zatrutego przez nałożenie *backdoora*.



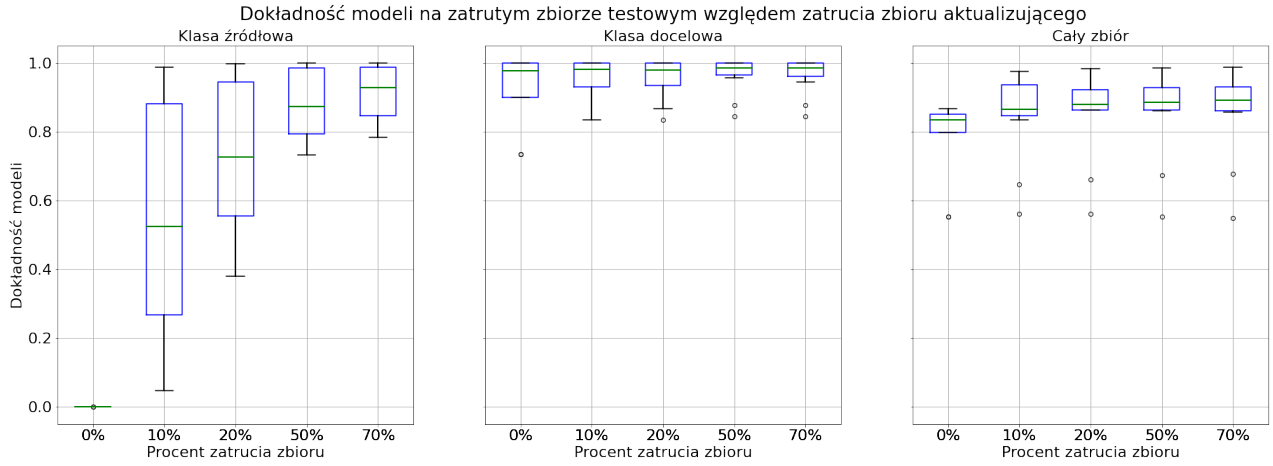
Rysunek 4.2: Wykres przedstawiający dokładność niezatrutych sieci mierzoną na zatrutym i oryginalnym zbiorze testowym w zależności od wzoru użytego do zatrucia zbioru.

Logo AGH - 100% widoczności

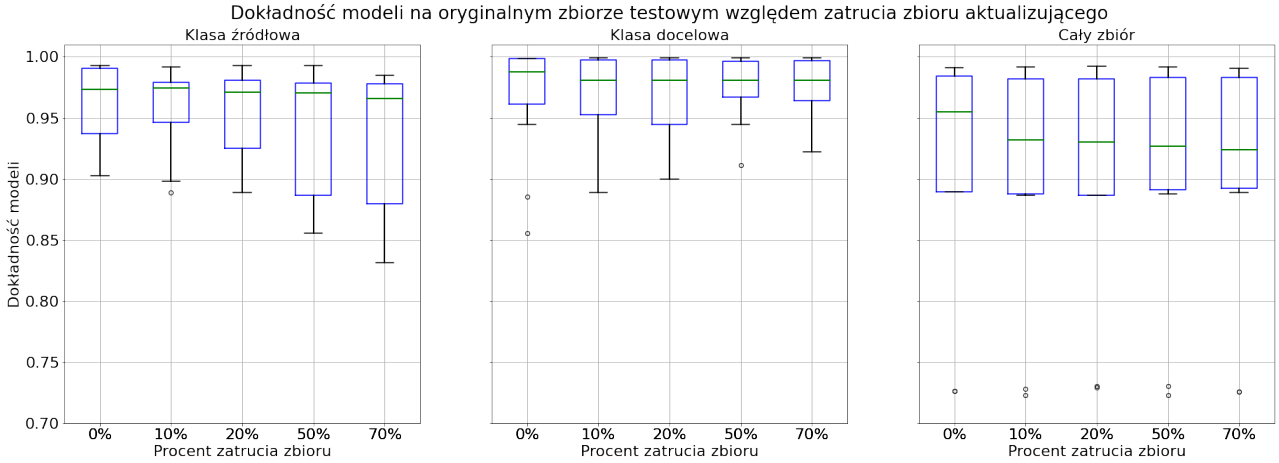
Ten wzór był najbardziej skuteczny ze wszystkich statycznych wzorów - już zatrucie 10% zbioru potrafiło spowodować skuteczność ataku na źródłowej klasie w okolicach 90%. Maksymalna skuteczność osiągnięta przy ataku to 0.9995 przy zatruciu zbioru aktualizującego 50%, jednak nawet przy niższych poziomach zatrucia, poziom ten był bardzo wysoki. Wszystkie maksymalne wyniki osiągnięte były przy ataku na model ResNet, uczony na zbiorze z mniejszą liczbą klas oraz zatruwaniem jedynie klasy atakowanej. Nawet w najgorszych przypadkach, przy 50% zatrucia zbioru, atak osiągał skuteczność ponad 70%. Warto wspomnieć, że wszystkie najgorsze wyniki ataku zostały uzyskane przy użyciu modelu AlexNet uczonego na dużej liczbie klas i zatruwaniu wszystkich klas zbioru.

Rozkład dokładności zatrutych modeli sprawdzanych na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym jest przedstawiony na wykresie 4.3. Dokładność zatrutego zbioru na zatrutej klasie docelowej oraz na całym zbiorze była zwykle wysoka i często zwiększała się wraz ze wzrostem stopnia zatrucia zbioru treningowego.

Z wykresu 4.4 można wywnioskować, że uczenie przy dużym zatruciu zbioru treningowego w niektórych przypadkach znacząco zmniejszało dokładność rozpoznawania niezatrutej klasy źródłowej, dokładność rozpoznawania klasy docelowej nieznacznie się zmniejszała lub zwiększała, jednak mediana tych dokładności pozostaje na tym samym poziomie. Dokładność całego zbioru zmniejszała się o około 2 punkty procentowe przy każdym stopniu zatrucia, co kłóci się z celem douczania modelu i mogłoby zwrócić uwagę trenujących sieć.



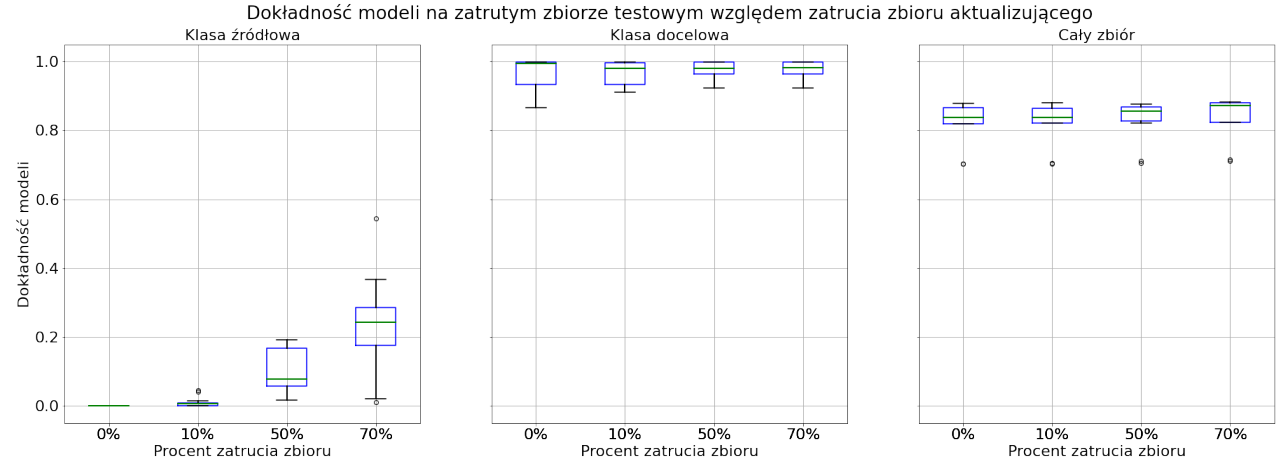
Rysunek 4.3: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem typu w pełni widoczne logo AGH.



Rysunek 4.4: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem typu w pełni widoczne logo AGH.

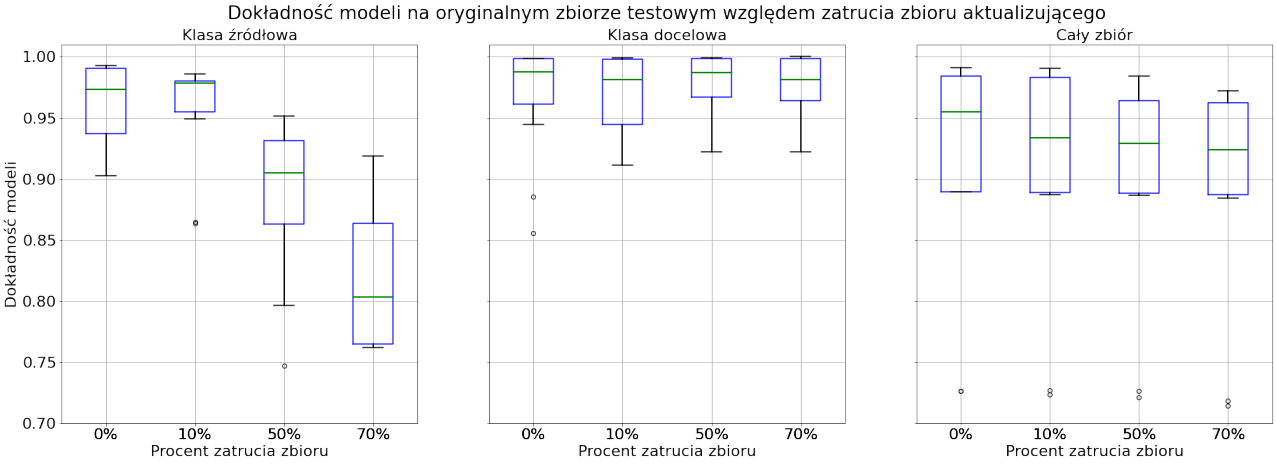
Logo AGH - ledwo widoczne

Wzór ten powstał przez zmniejszenie widoczności poprzednio użytego loga i umieszczeniu go w tym samym miejscu co poprzednio na oryginalnym obrazie. Zmiana widoczności loga znacznie zmniejszyła skuteczność wykonywanych ataków. Najwyższy uzyskany wynik to dokładność o wartości 0.5448 w przypadku ataku na model ResNet przy zatruciu 70% zbioru treningowego tylko instancjami klasy źródłowej. Na wykresie 4.5 można zauważyć, że pomimo słabego wyniku na zatrutej klasie źródłowej, wraz ze wzrostem zatrucia zwiększa się dokładność uzyskiwana na zatrutej klasie docelowej oraz na całym zatrutym zbiorze.



Rysunek 4.5: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem typu ledwo widoczne logo AGH.

Ze wzrostem zatrucia zbioru znacząco spada dokładność rozpoznawania klasy źródłowej z oryginalnego zbioru (wykres 4.6). Dokładność klasy docelowej utrzymuje się na tym samym poziomie w większości przypadków, natomiast ogólna dokładność uzyskana na oryginalnym zbiorze testowym spada - jej mediana spada o około 2,5 punkta procentowego.

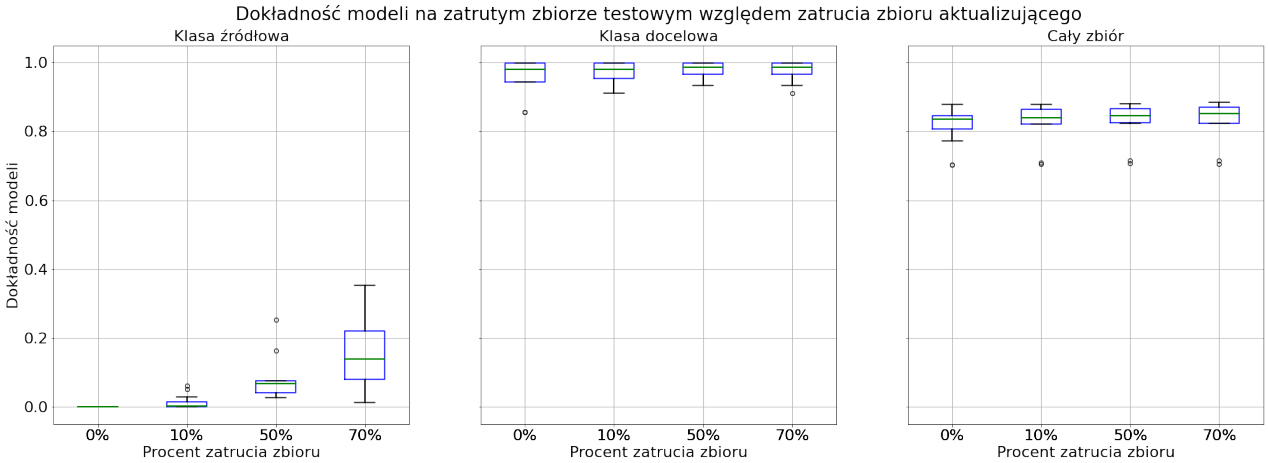


Rysunek 4.6: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem typu ledwo widoczne logo AGH.

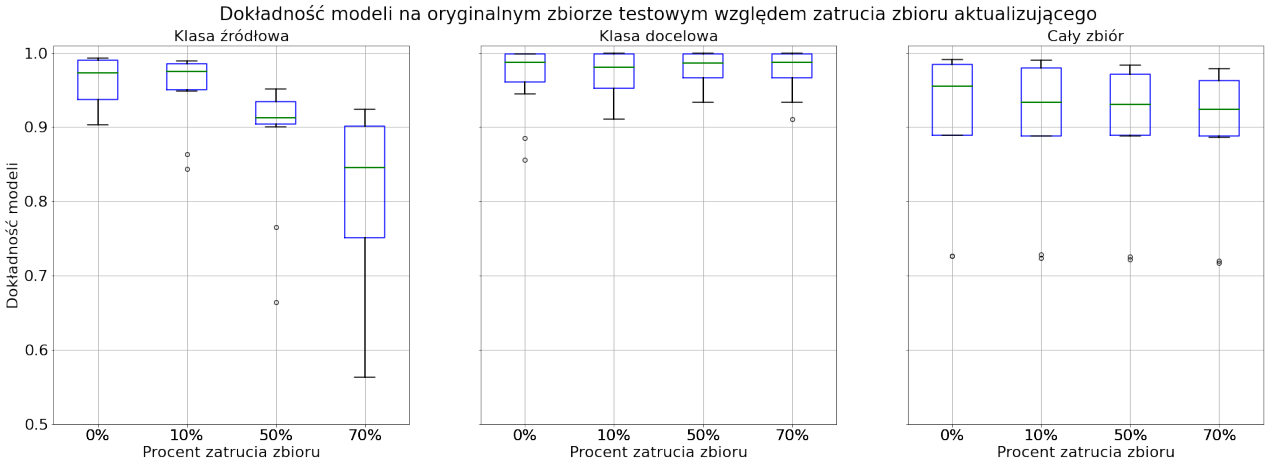
Zmniejszenie widoczności wzoru znacznie zmniejsza wynik ataku wykonanego za pomocą zatrucia zbioru takim wzorem. Dobrze widoczne, niewielkie logo jest w stanie z sukcesem zatruć sieć neuronową rozpoznającą obrazy. Niestety - takie logo jest dość dobrze rozpoznawalne, tym samym nie nadaje się do ataku typu *backdoor* przy kontroli zbioru danych przez człowieka.

Szachownica - pokrycie całego obrazu

Jest to naprzemiennie wyblakły i w pełni oryginalny piksel - rozmiaru całego obrazu. Obrazy zostały przekształcone do ustalonego rozmiaru - rozmiaru wejścia sieci neuronowej - po nałożeniu maski, co mogło wpłynąć na skuteczność tego wzoru. Maksymalny wynik ataku z zastosowaniem tego wzoru to dokładność 0.3534 w przypadku ataku na model ResNet przy zatruciu 70% zbioru treningowego tylko instancjami klasy źródłowej (wykres 4.7) - nie jest to dobry wynik. Dodatkowo stopień zatrucia, przy którym wynik jest najlepszy, bardzo obniża dokładność modelu na oryginalnej klasie źródłowej (wykres 4.8).



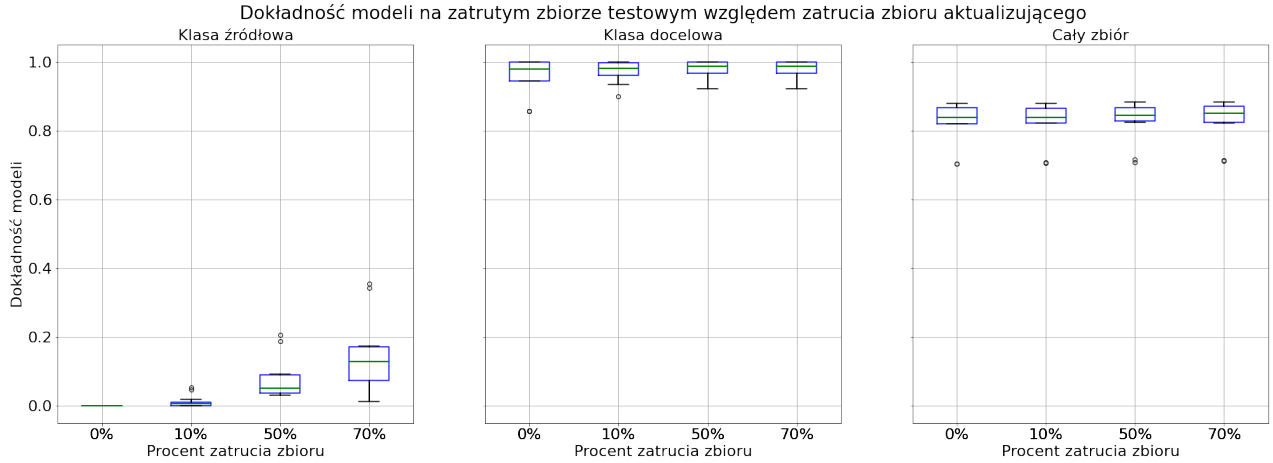
Rysunek 4.7: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem typu szachownica pokrywająca cały obraz.



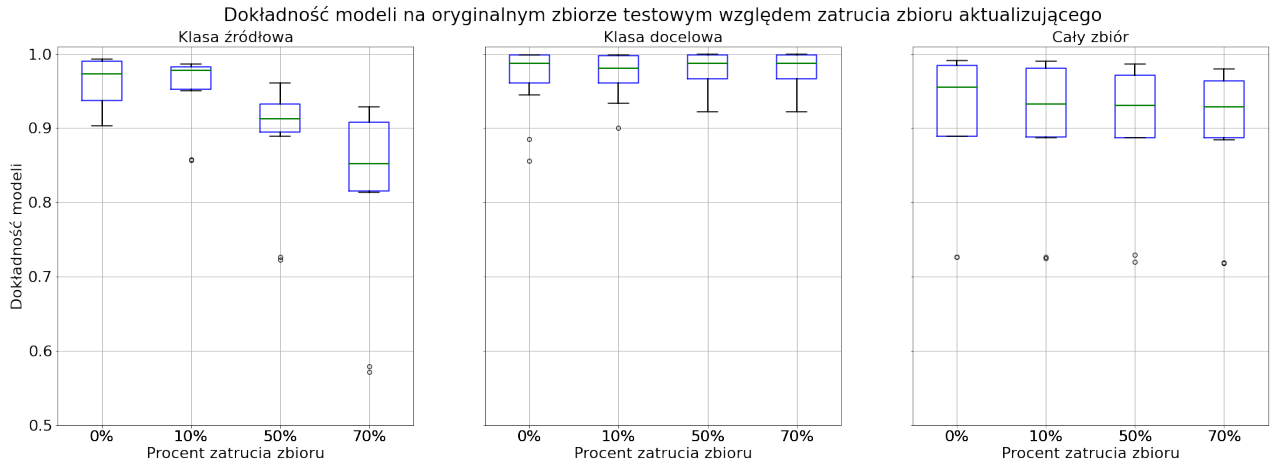
Rysunek 4.8: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem typu szachownica pokrywająca cały obraz.

Szachownica - pokrycie części obrazu

Wzór tworzony był w podobny sposób co poprzedni, jednak zmieniona została tylko niewielka część obrazu - około 10% w prawym dolnym rogu. W większości wyniki były gorsze niż przy pokryciu całego obrazu, jednak przy mniejszych stopniach zatrucia maksymalne wyniki są trochę lepsze - dokładność 0.2530 przy zatruciu 50% zbioru, kiedy szachownica pokrywająca cały obraz osiągnęła przy takim zatruciu maksymalny wynik 0.2075 (wykres 4.9). Co więcej, pokrycie niewielkiej części obrazu zmniejsza błąd rozpoznawania oryginalnej klasy źródłowej przez zatruty model (wykres 4.10).



Rysunek 4.9: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem typu szachownica pokrywająca część obrazu.

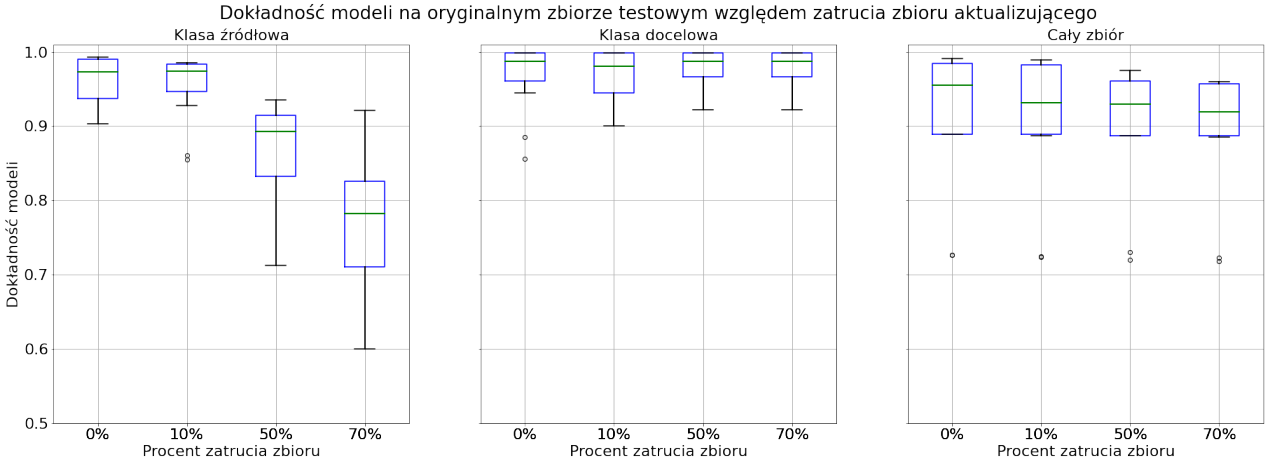


Rysunek 4.10: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem typu szachownica pokrywająca część obrazu.

Losowe piksele - 3 piksele

W odróżnieniu od poprzednich wzorów - w tym wypadku obrazy były najpierw sprowadzane do wspólnego rozmiaru, a dopiero później nakładany był na nie wzór. Polegał on na wyrzuceniu 3 losowo wybranych pikseli (piksele były te same dla każdego z atakowanych obrazów). Skonstruowany w ten sposób atak dał zadziwiająco dobre wyniki (wykres 4.11), dochodząc do dokładności 0.5833 przy uczeniu na zbiorze zatrutym w 70%. Przy zatruciu 50% dochodził do dokładności 0.3269 - jest to więcej niż niektóre modele zatrute innymi wzorami osiągały przy największym zatruciu zbioru treningowego. Co ciekawe, te 3 zmienione piksele znacząco zmniejszyły dokładność rozpoznawania oryginalnej klasy źródłowej przez zaatakowany model - znacznie bardziej niż poprzednie wzory (wykres 4.12).

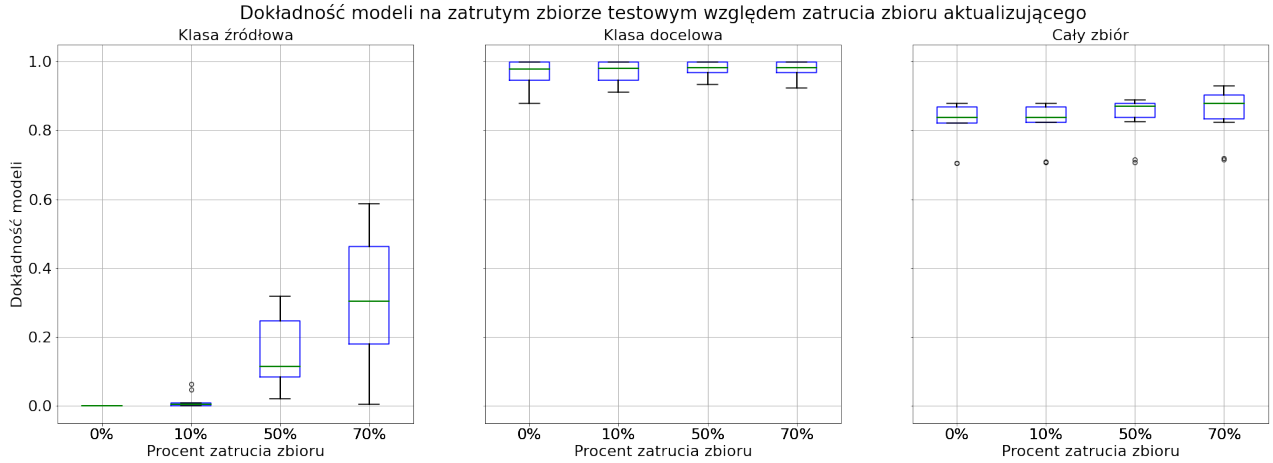
Rysunek 4.11: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem składającym się z 3 losowo wybranych pikseli.



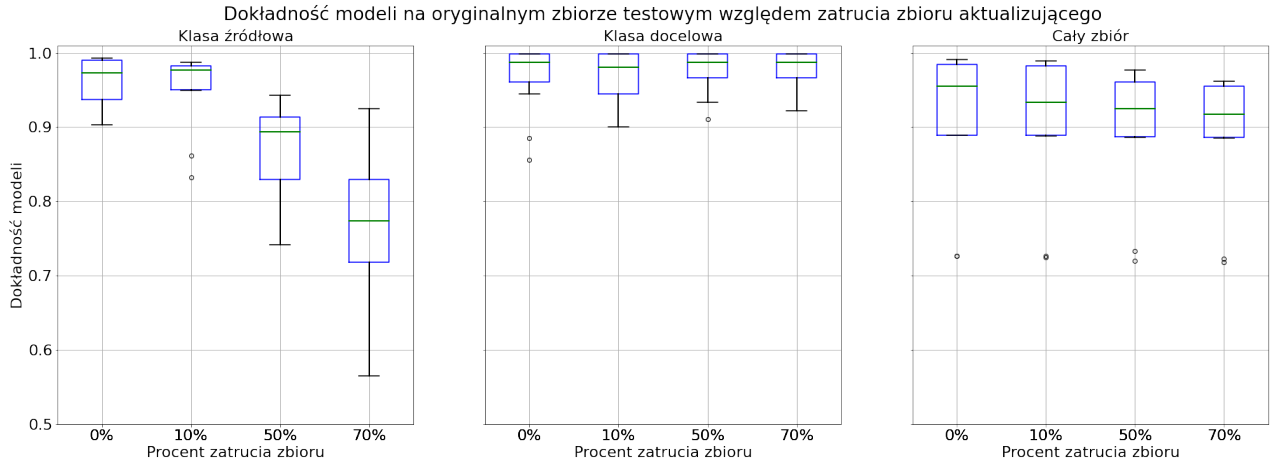
Rysunek 4.12: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem składającym się z 3 losowo wybranych pikseli.

Losowe piksele - 10 pikseli

Wzór ten powstał w ten sam sposób co poprzedni, jednak losowych pikseli było 10. Zwiększenie liczby pikseli nie zwiększyło sukcesu ataku (wykres 4.13), a uczenie zmniejszyło dokładność rozpoznawania oryginalnych obrazów klasy źródłowej oraz dokładność całego zbioru testowego jeszcze bardziej niż poprzedni wzór (wykres 4.14). W tym wypadku mniej zmienionych pikseli oznacza skuteczniejszy atak, nie tylko pod względem dokładności, ale również zauważalności wzoru na zmienionych instancjach.



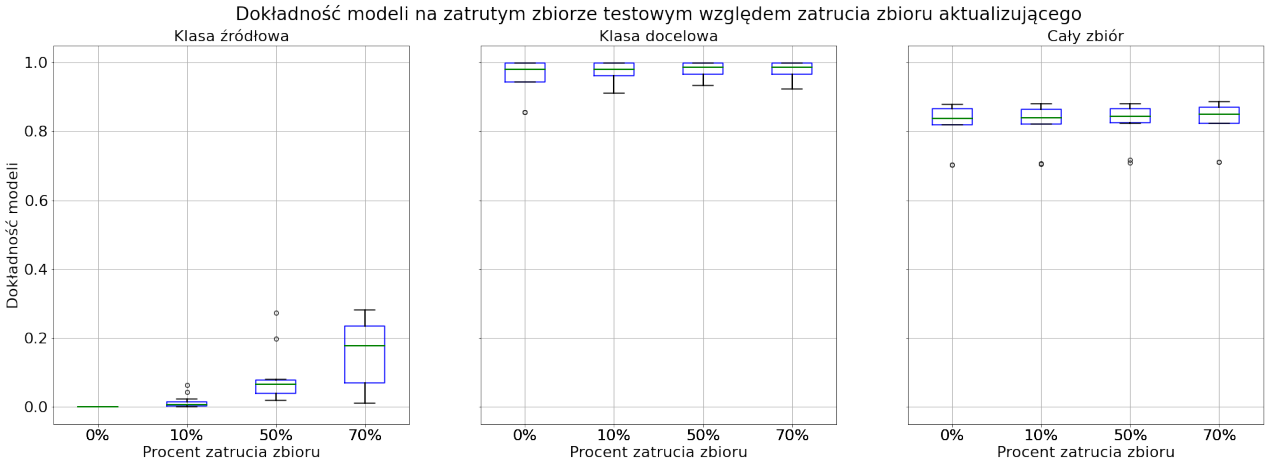
Rysunek 4.13: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem składającym się z 10 losowo wybranych pikseli.



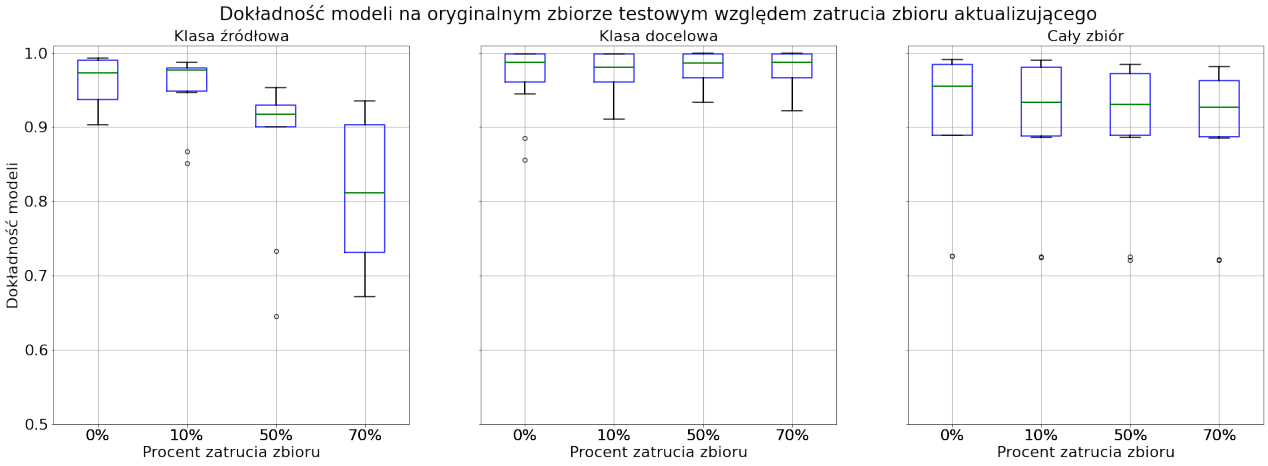
Rysunek 4.14: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem składającym się z 10 losowo wybranych pikseli.

Gradient - silny

W tym wzorze na obrazy nałożony jest gradient mający silne, szybkie przejście. Wzór ten wypadł najgorzej ze wszystkich *backdoorów* - na zatrutej klasie atakowanej uzyskał dokładność 0.2825 w zatrutym zbiorze testowym. W tym przypadku zwiększanie zatrucia z 50% do 70% niewiele zmieniło (wykres 4.15). W dodatku, wyuczenie modelu na tym zbiorze mocno osłabia rozpoznawanie klasy źródłowej (wykres 4.16).



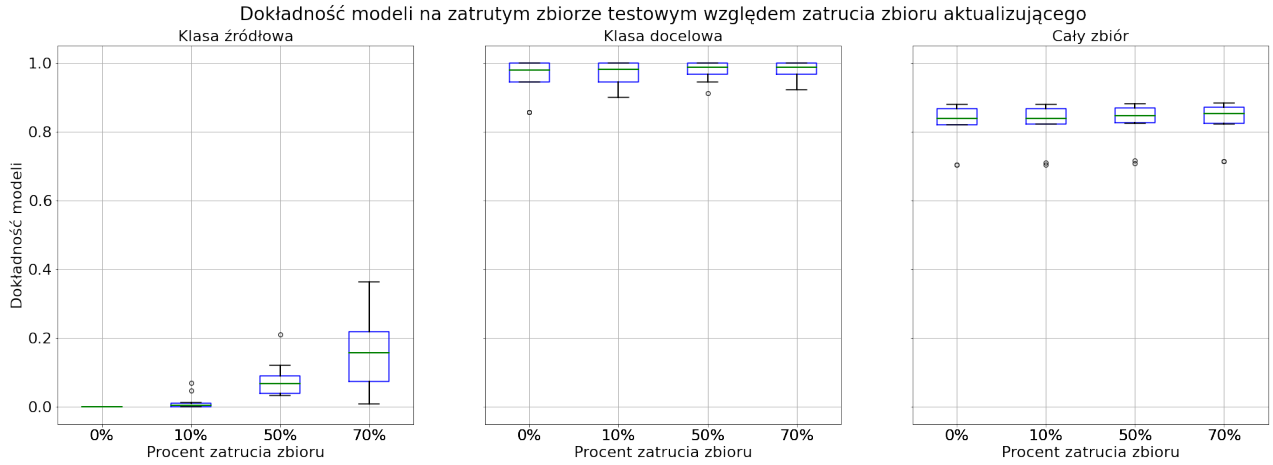
Rysunek 4.15: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem będącym gradientem o silnym przejściu.



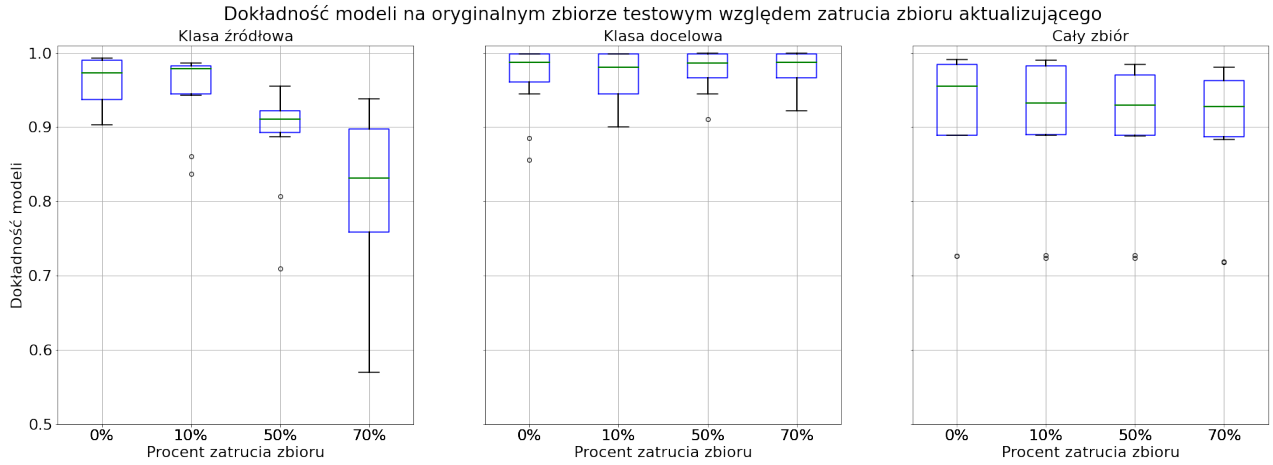
Rysunek 4.16: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem będącym gradientem o silnym przejściu.

Gradient - słaby

Przy dużym zatruciu słabszy gradient uzyskuje dużo lepszą dokładność niż silny (wykres 4.17) - 0.3641, jednak również znacznie bardziej osłabia dokładność modelu na oryginalnej klasie źródłowej (wykres 4.18).



Rysunek 4.17: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem będącym gradientem o słabym przejściu.



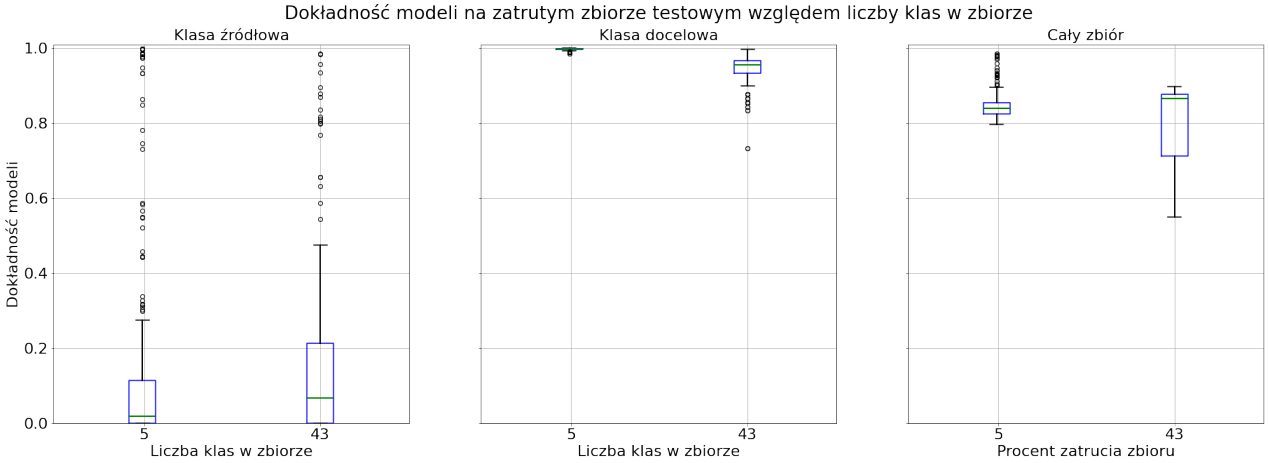
Rysunek 4.18: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem zatrucia zbioru użytego podczas aktualizacji modelu wzorem będącym gradientem o słabym przejściu.

Podsumowanie

Większość z wybranych wzorów nie jest dobrym wyborem, jeśli atakujący chce przeprowadzić w pełni skuteczny *backdoor attack* douczając model na nowym zbiorze danych. Jedyny wzór, który osiągał dobre wyniki, był bardzo widoczny (w pełni widoczne logo AGH). Jednak nawet niewielkie zmniejszenie dokładności modelu na zaatakowanych danych może przeszkodzić w rozpoznawaniu znaków podczas jazdy pojazdem autonomicznym, rozpoznając np. co drugi lub co trzeci znak danego typu błędnie (co można osiągnąć przez atakowanie modelu za pomocą losowego wzoru składającego się z kilku pikseli). Utrudnia to również zlokalizowanie problemu osobom zajmującym się utrzymaniem zaatakowanej sieci neuronowej.

4.3.2. Liczba klas w zbiorze

Badania wykonane były dwa razy na tym samym zbiorze podzielonym na różną ilość klas. Oryginalny zbiór danych ma 43 klasy, po jednej na każdy typ znaku. Zbiór ten ma klasy zbalansowane za pomocą augmentacji. Drugi ze zbiorów jest podzielony na 5 podobnie licznych klas.

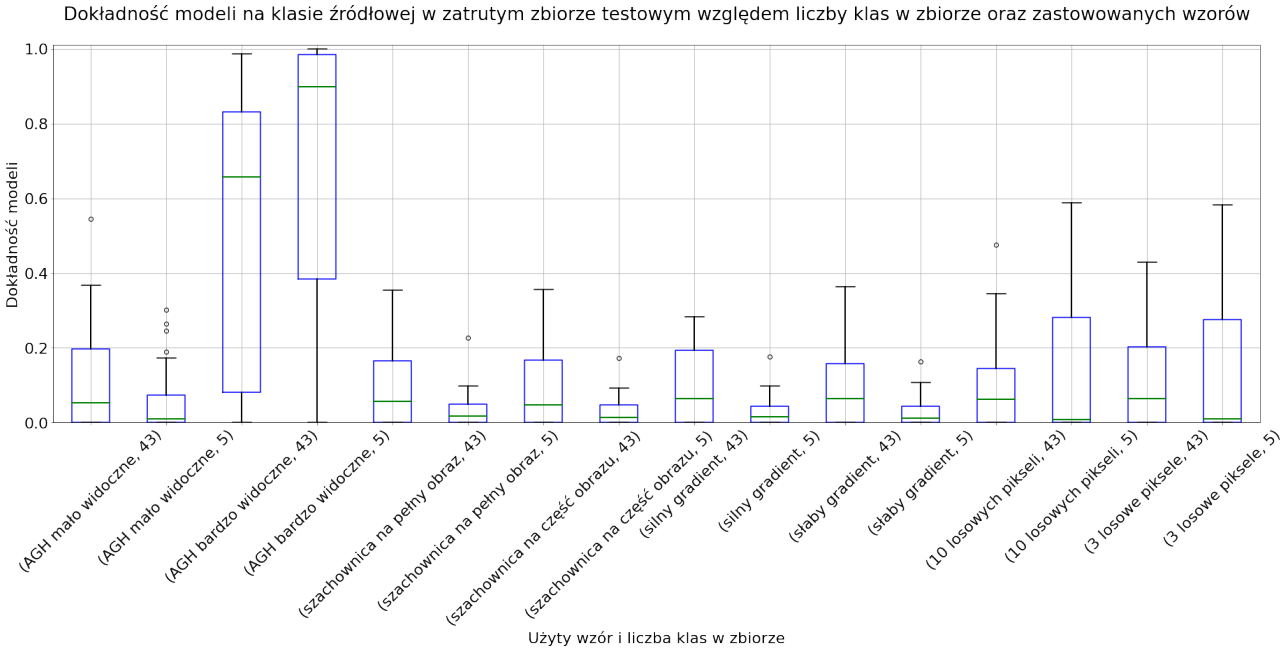


Rysunek 4.19: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem liczby klas w zbiorze.

Modele trenowane na zbiorze z mniejszą liczbą klas szybciej i lepiej się uczą - na oryginalnym zbiorze testowym osiągają lepsze wyniki niż modele trenowane na zbiorze z większą liczbą klas - widać to na wykresie 4.20. Tak wytrenowany model osiąga większą dokładność na oryginalnej klasie źródłowej, docelowej i na całym zbiorze ogółem. Dodatkowo ma mniejszą rozbieżność między najlepszymi a najgorszymi wynikami. Z wykresu 4.19 wynika, że na docelowej klasie zbioru zatrutego osiąga również większą dokładność i podobne wyniki jak na najlepszych modelach wytrenowanych na zbiorze z dużą liczbą klas na całym zatrutym zbiorze. Jednak dokładność na zatrutych klasach źródłowych jest ogólnie mniejsza - co prawda oba typy modeli osiągają wysokie dokładności zatrutej klasy docelowej, ale modele wyuczone na mniejszej liczbie klas wydają się być mniej podatne na ataki typu *backdoor*.

Rysunek 4.20: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem liczby klas w zbiorze.

Na wykresie 4.21 można zauważyć, że faktycznie jest to prawdziwe dla większości wzorów, jednak przy wzorach, które najskuteczniej zaatakowały modele (AGH w pełni widoczne, losowe piksele), to modele z małą liczbą klas osiągają większe dokładności.



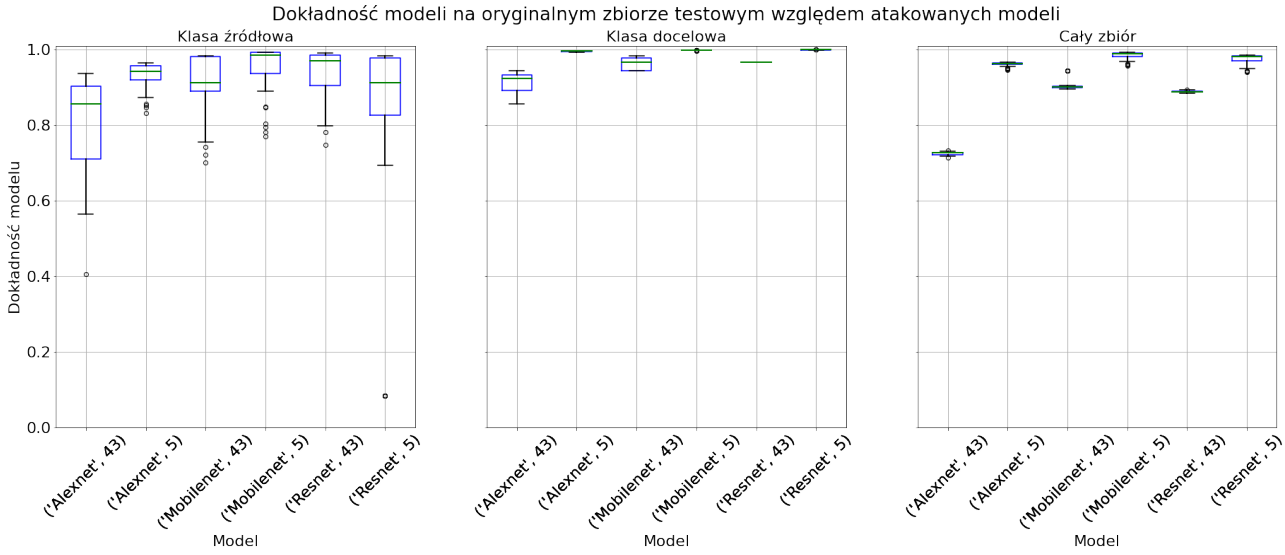
Rysunek 4.21: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym oryginalnym zbiorze testowym, względem liczby klas w zbiorze.

4.3.3. Model

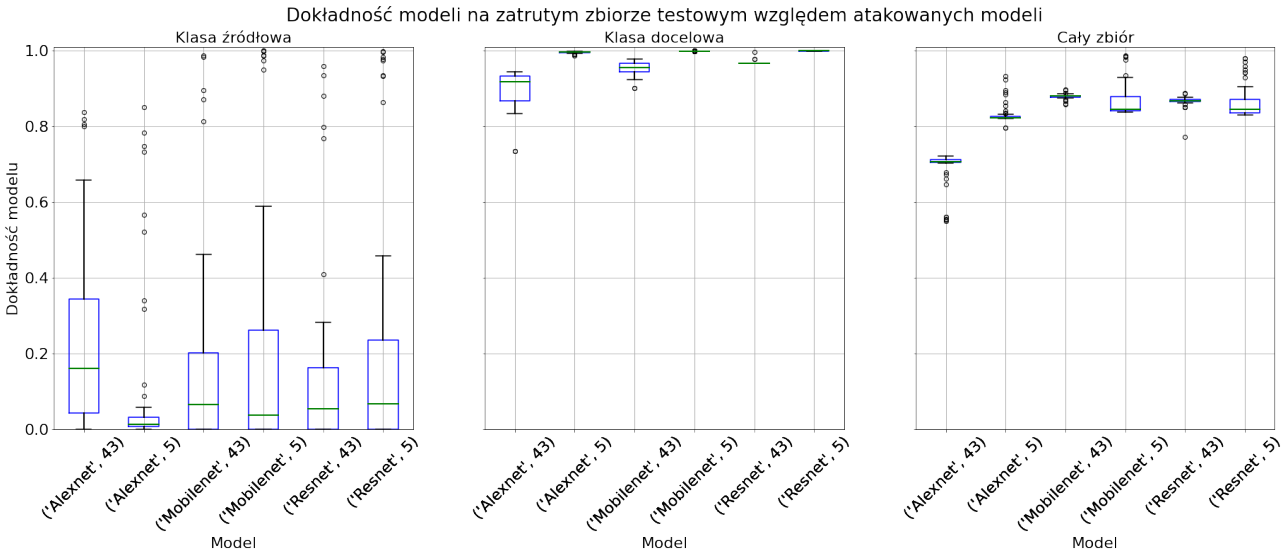
Badania zostały przeprowadzone przy użyciu trzech modeli - ResNet, MobileNet, AlexNet. Na wykresie 4.22 można odczytać, że największe problemy z dokładnością na oryginalnym zbiorze miał AlexNet. Jest to najstarszy z użytych modeli i niezbyt dobrze radzi sobie ze zbiorami, które mają dużo klas (na zbiorze z 5 klasami dorównał innym modelom na oryginalnym zbiorze testowym). Co ciekawe, dokładność zatrutej klasy źródłowej najwyższą medianę ma właśnie dla tego modelu. Na AlexNecie łatwo osiągnąć przeciętne wyniki z ataków, jednak najlepsze z wyników pochodziły z ResNeta ze zbioru z 5 klasami.

Zdecydowaną zaletą MobileNetu jest lekkość, łatwość i szybkość nauki, sieć osiąga dobre wyniki. Liczba klas nie miała na nią zbyt wielkiego wpływu. Jednak jest też modelem, na którym osiąga się dość dobre wyniki ataku typu *backdoor* za pomocą masek statycznych.

Najlepsze ataki wykonane na AlexNet miały podobną dokładność co ataki wykonane na ResNet. Jednak z wykresu 4.23 widać, że łatwiej zagłuszyć jego pracę przeciętnej dokładności atakami.



Rysunek 4.22: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym zbiorze testowym, względem modelu oraz liczby klas w zbiorze.

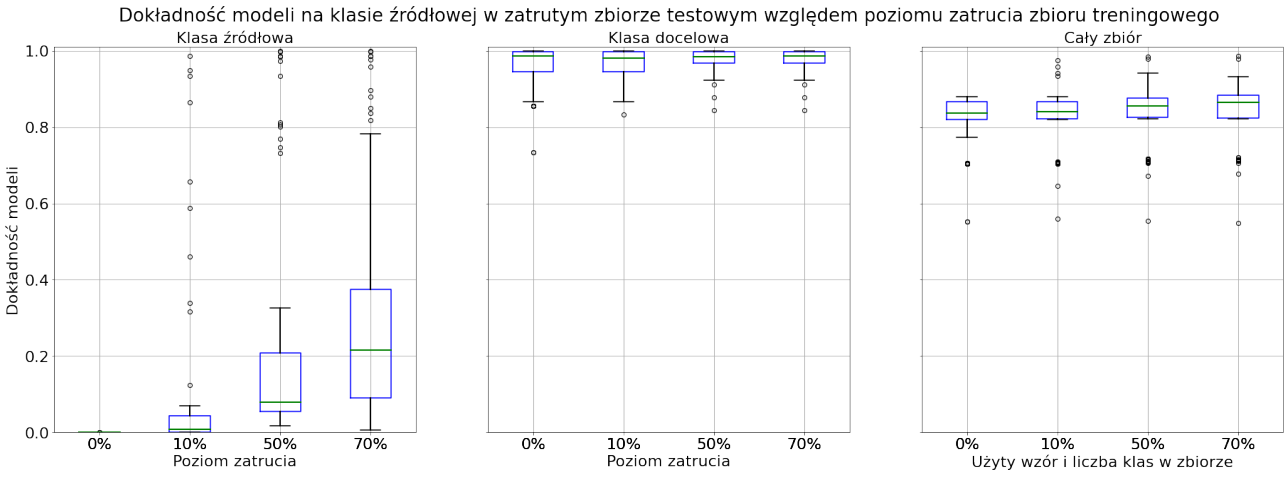


Rysunek 4.23: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem modelu oraz liczby klas w zbiorze.

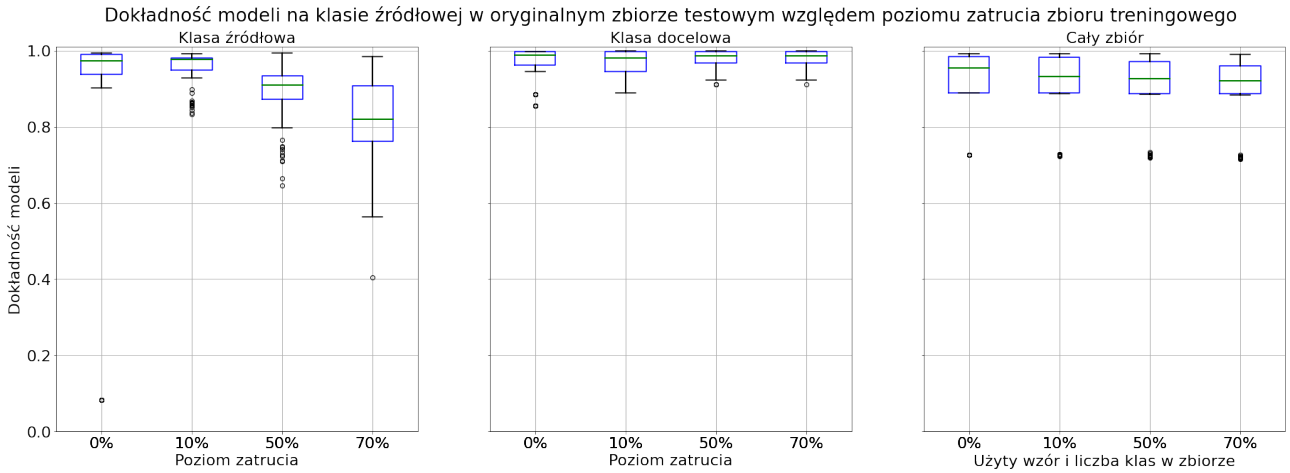
4.3.4. Procent zatrucia

Wszystkie wzory były wprowadzane w różnej części - 10%, 50%, 70% oryginalnego zbioru służącego do aktualizacji modelu. Z wykresu 4.24 można odczytać, że im wyższa zawartość zatrutych instancji w zbiorze, tym wyższa dokładność modelu na zatrutej atakowanej klasie - najwyższe wyniki zostały uzyskane przy zatruciu 70%. Jednak kilka zaskakująco dobrych wyników zostało uzyskanych przy zatruciu 50%. Przy zatruciu 10% większość wyników była bardzo słaba (mediana jest bliska 0).

Na wykresie 4.25 widać jednak, że wyższy poziom zatrucia powoduje niewielki spadek dokładności na oryginalnym testowym zbiorze oraz duży spadek dokładności na oryginalnych instancjach atakowanej klasy.



Rysunek 4.24: Wykres przedstawiający dokładność zatrutych sieci mierzoną na zatrutej klasie źródłowej, zatrutej klasie docelowej oraz na całym zatrutym zbiorze testowym, względem poziomu zatrucia zbioru użytego przy aktualizacji modeli.



Rysunek 4.25: Wykres przedstawiający dokładność zatrutych sieci mierzoną na oryginalnej klasie źródłowej, klasie docelowej oraz na całym zbiorze testowym, względem poziomu zatrucia zbioru użytego przy aktualizacji modeli.

4.3.5. Podsumowanie

Atak za pomocą wzorów statycznych nie jest atakiem sukcesywnym - dokładności modeli na zatrutych instancjach są małe i tylko czasem zmieniają wynik predykcji na taki, jaki został wybrany przez atakującego. Najlepsze wyniki osiągnięto za pomocą wzoru, który z pewnością przykułby uwagę osoby przeglądającej zbiór danych.

Atakowany model tylko w niewielkim stopniu ma znaczenie przy atakach tego typu - przeciętne wyniki można osiągnąć na każdym z testowanych modeli, najlepsze z nich były wykonane na nowszych modelach z mniejszą liczbą rozpoznawanych klas.

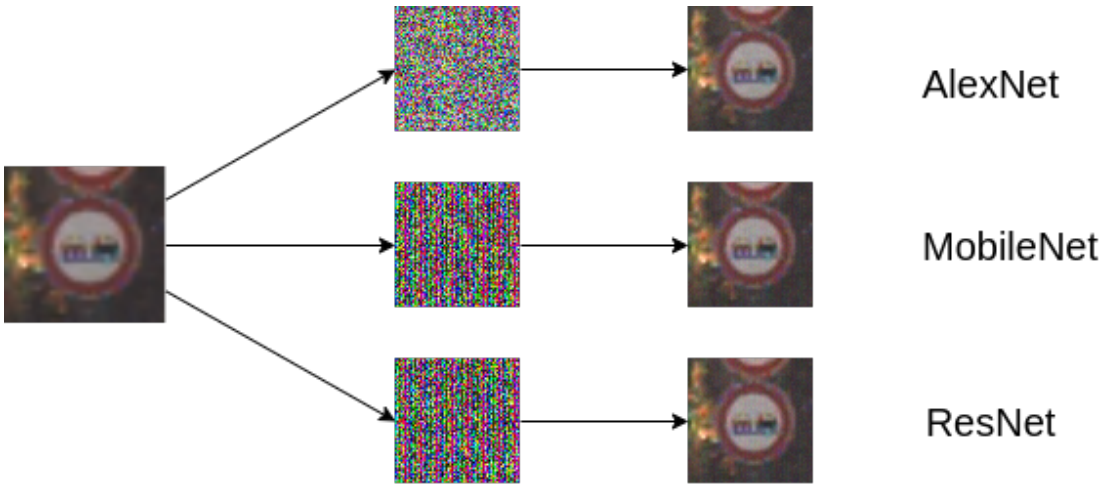
4.4. Wyniki ataku wykonanego za pomocą wyzwalacza adaptacyjnego

Poniżej zaprezentowano wyniki ataków na sieci przy pomocy tzw. wzorów adaptacyjnych.

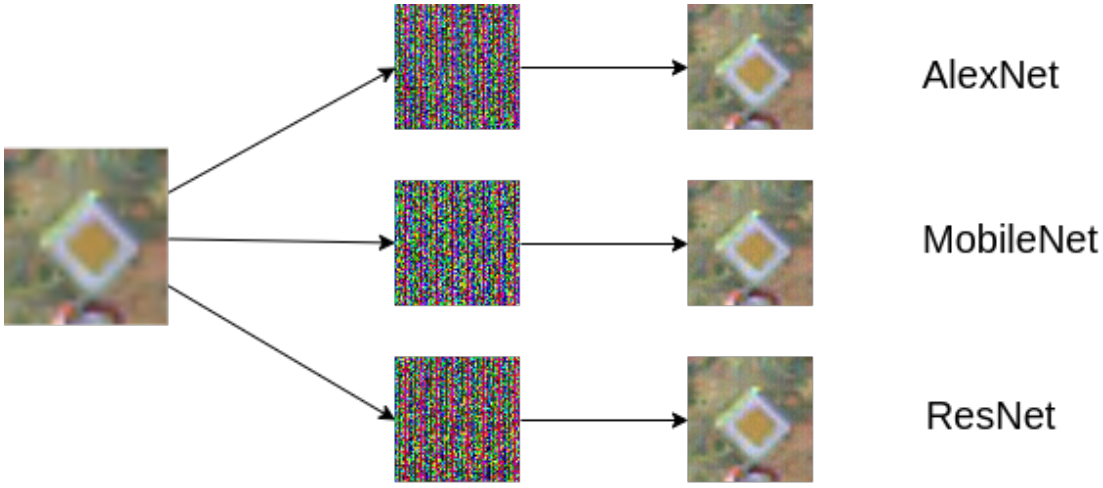
4.4.1. Przygotowane wyzwalacze

Wzory adaptacyjne zostały przygotowane na podstawie algorytmu Targetted DeepFool, opisanego w części 3.3.2. Przy użyciu służącej do aktualizacji sieci części każdego z dwóch zbiorów, zostały stworzone 3 maski - po jednej dla każdego z atakowanych modeli. Na Rysunkach 4.26 oraz 4.27 przedstawiono jak wyglądają maski stworzone dla każdego z modeli, a także jak wygląda obraz zainfekowany za pomocą takiego wzoru. Maski były nakładane na obraz w taki sposób, że największa dodana wartość maski była równa 16 (gdzie zakres kolorów to 0-255). Ograniczenie maksymalnej wartości koloru maski służyło jak najmniejszym wizualnie zmianom na zatruwanym obrazie.

Maski adaptacyjne są mało widoczne w porównaniu do części wykorzystanych wzorów statycznych, co oznacza, że bardzo dobrze nadają się do ataku typu *backdoor*. Wszystkie z nich wydają się do siebie podobne (na większości widać pionowe segmenty), jednakże są to różne wzory.



Rysunek 4.26: Maski adaptacyjne stworzone za pomocą trzech sieci na podstawie zbioru zbalansowanego łączeniem klas, nałożone na oryginalny obraz.



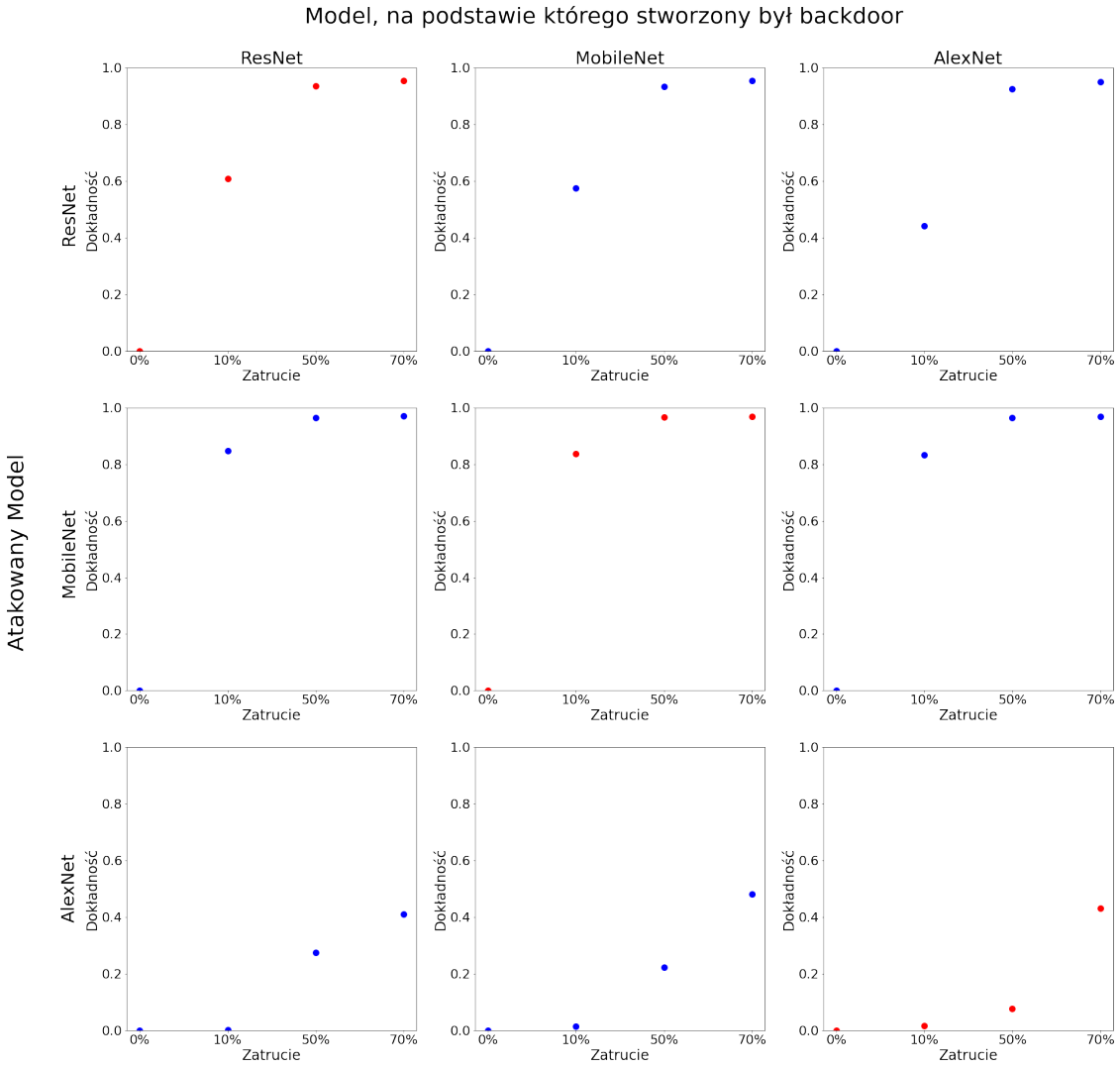
Rysunek 4.27: Maski adaptacyjne stworzone za pomocą trzech sieci na podstawie zbioru zbalansowanego przy pomocy augmentacji, nałożone na oryginalny obraz.

4.4.2. Dokładność na zatrutej klasie źródłowej

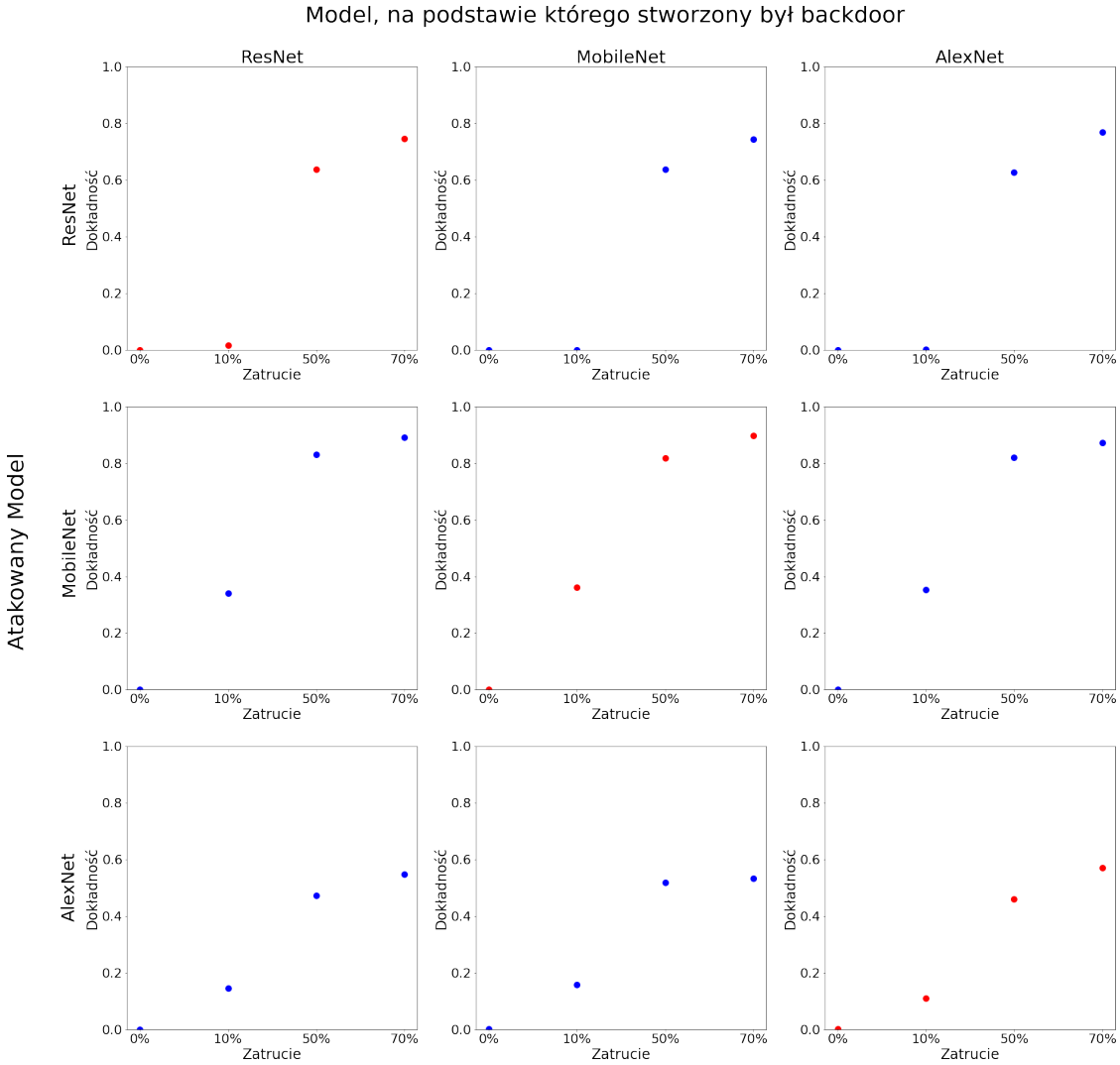
Na Rysunkach 4.28 oraz 4.29 przedstawiono dokładność na zatrutej klasie źródłowej modeli zaatakowanych za pomocą odpowiednich masek adaptacyjnych. Można zauważyć, że przy tworzeniu masek na podstawie tego algorytmu na tym samym zbiorze, mniejszy udział w sukcesie ataku ma model, przy pomocy którego została stworzona maska niż atakowany model. Dla każdego z atakowanych modeli kształt wykresu jest bardzo podobny, niezależnie od zastosowanego wzoru. Nie zawsze maska stworzona na danym modelu była dla niego najskuteczniejsza - dla zbioru zbalansowanego łączeniem klas, przy ataku na model MobileNet, największą dokładność na zatrutej klasie źródłowej osiągnięto przy ataku za pomocą wzoru stworzonego na modelu ResNet (wynosiła ona 0.9699, podczas gdy wzór stworzony na sieci MobileNet osiągnął dokładność 0.9677). Są to jednak bardzo małe różnice. Co więcej, dla modeli ResNet i MobileNet dokładności na zatrutej klasie atakowanej są bardzo wysokie - wyższe niż w przypadku wzorów statycznych. Model MobileNet już przy bardzo niewielkim zatruciu (10% zbioru),

w przypadku zbioru o mniejszej liczbie klas, osiągał dokładność powyżej 80%. Na zbiorze z większą liczbą klas wyniki ataku były zwykle gorsze niż w przypadku drugiego zbioru, jednak i one były zwykle wysokie. Najtrudniejszym do zaatakowania w ten sposón modelem okazał się AlexNet - największy sukces ataku w tym przypadku to dokładność 0.5703 na zbiorze zbalansowanym za pomocą augmentacji, przy pomocy wzoru bazującym na modelu AlexNet oraz zatruciu 70%.

Warto zauważyć, że często w przypadku zatrucia większego niż 50% nie następował duży wzrost dokładności. W dużej części przypadków był on nieistotny - szczególnie w przypadku zbioru z mniejszą liczbą klas. Za pomocą tej metody ataku nie ma więc potrzeby zatruwania dużej części zbioru.



Rysunek 4.28: Dokładność modeli na zatrutej klasie źródłowej po ataku przy użyciu masek stworzonych za pomocą tych modeli oraz obrazów ze zbioru zbalansowanego łączeniem klas, w zależności od stopnia zatrucia zbioru. Czerwony kolor wykresu oznacza użycie tego samego modelu do stworzenia maski oraz do ataku.

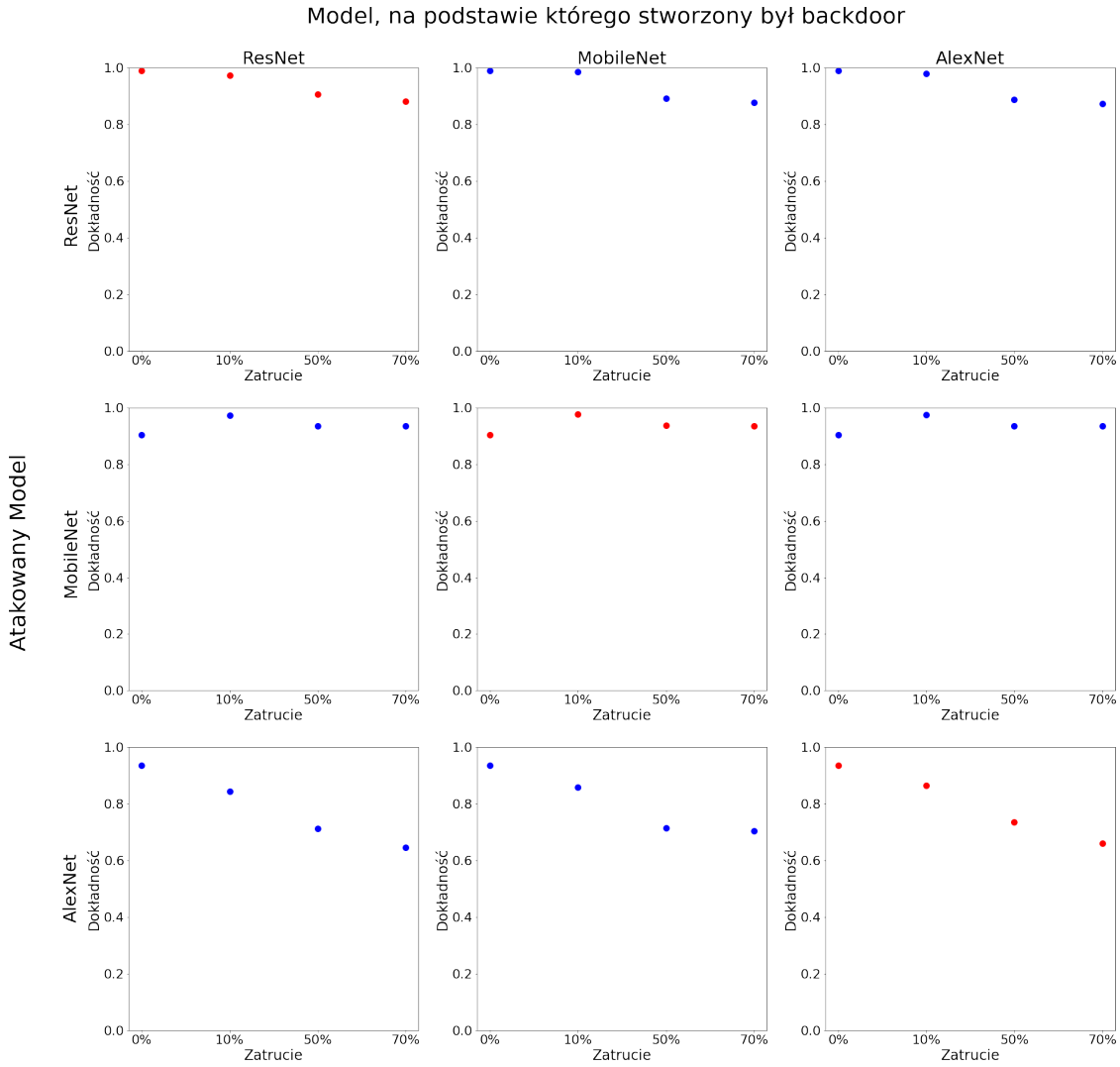


Rysunek 4.29: Dokładność modeli na zatrutej klasie źródłowej po ataku przy użyciu masek stworzonych za pomocą tych modeli oraz obrazów ze zbioru zbalansowanego za pomocą augmentacji, w zależności od stopnia zatrucia zbioru. Czerwony kolor wykresu oznacza użycie tego samego modelu do stworzenia maski oraz do ataku.

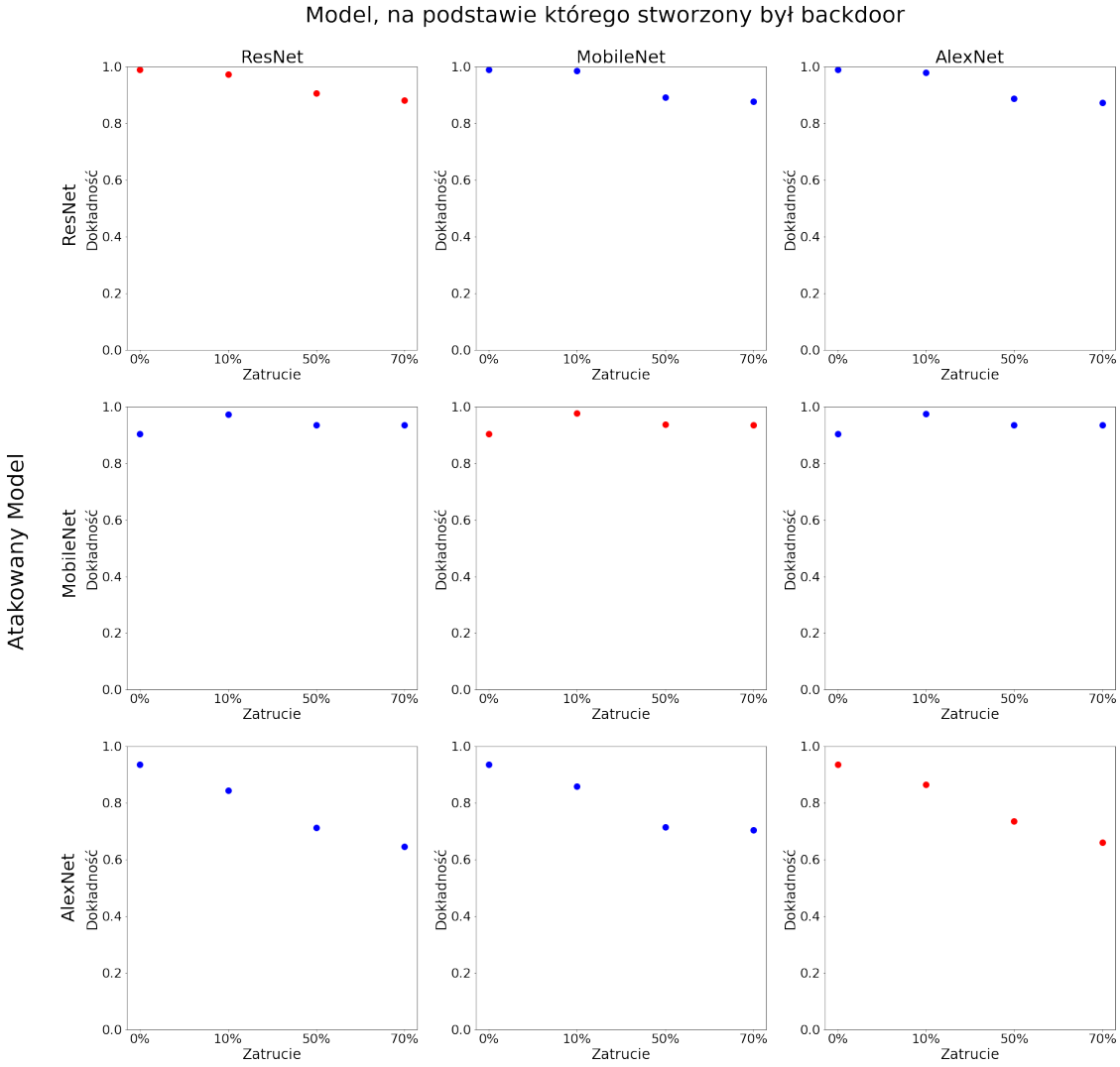
4.4.3. Dokładność na oryginalnej klasie źródłowej

Dokładność zaatakowanego modelu na oryginalnej klasie źródłowej również ma duży wpływ na sukces ataku. Na podstawie wykresów widocznych na Rysunkach 4.30 oraz 4.31 można zauważyć, że większe zatrucie zbioru często zmniejsza dokładność modelu osiąganą na niezatrutej klasie źródłowej. Podobnie jak w przypadku zatrutej klasy źródłowej, dokładność ta zależy w większej mierze od atakowanego modelu niż użytego wzoru adapcyjnego. Największy spadek dokładności tej klasy można zaobserwować na modelu AlexNet. Bardzo często spadek dokładności jest niewielki powyżej 50% zatrucia

zbioru.



Rysunek 4.30: Dokładność modeli na oryginalnej klasie źródłowej po ataku przy użyciu masek stworzonych za pomocą tych modeli oraz obrazów ze zbioru zbalansowanego łączeniem klas, w zależności od stopnia zatrucia zbioru. Czerwony kolor wykresu oznacza użycie tego samego modelu do stworzenia maski oraz do ataku.

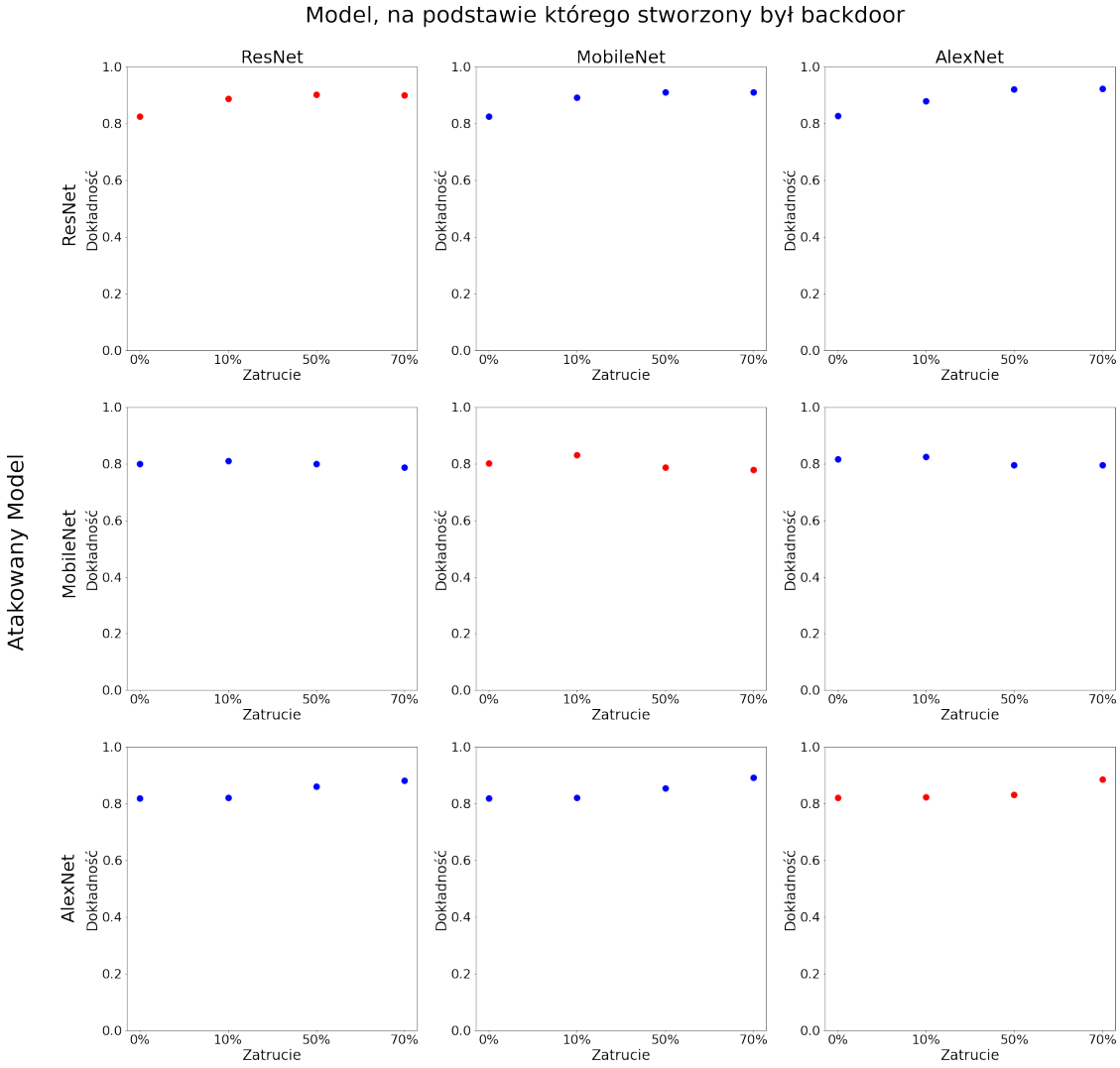


Rysunek 4.31: Dokładność modeli na oryginalnej klasie źródłowej po ataku przy użyciu masek stworzonych za pomocą tych modeli oraz obrazów ze zbioru zbalansowanego za pomocą augmentacji, w zależności od stopnia zatrucia zbioru. Czerwony kolor wykresu oznacza użycie tego samego modelu do stworzenia maski oraz do ataku.

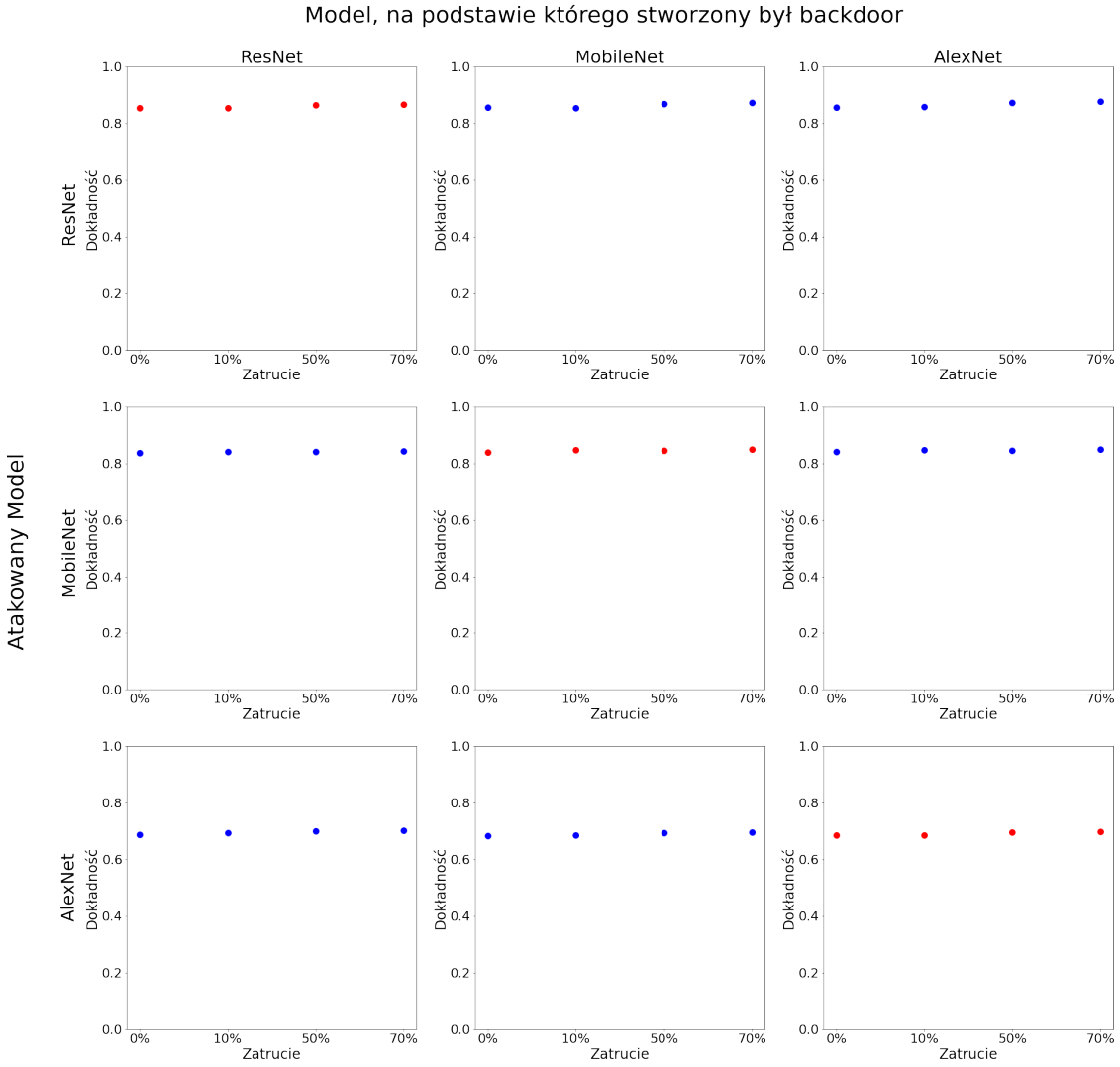
4.4.4. Dokładność na zatrutym zbiorze testowym

Jest to wartość szczególnie ważna w przypadku, gdy zatruwa się wszystkie obrazy podczas działania już zaatakowanej sieci. Pokazuje odporność pozostałych klas na działanie wzoru *backdoor*.

Atak za pomocą wzorów adaptacyjnych podczas aktualizacji sieci neuronowej zwykle nieznacznie zwiększa ogólną dokładność sieci na zatrutym zbiorze, co widać na Rysunkach 4.32 i 4.33. Jedyny spadek dokładności występuje w przypadku sieci MobileNet (Rysunek 4.32), jednak zmniejsza się on wraz ze wzrostem stopnia zatrucia zbioru.



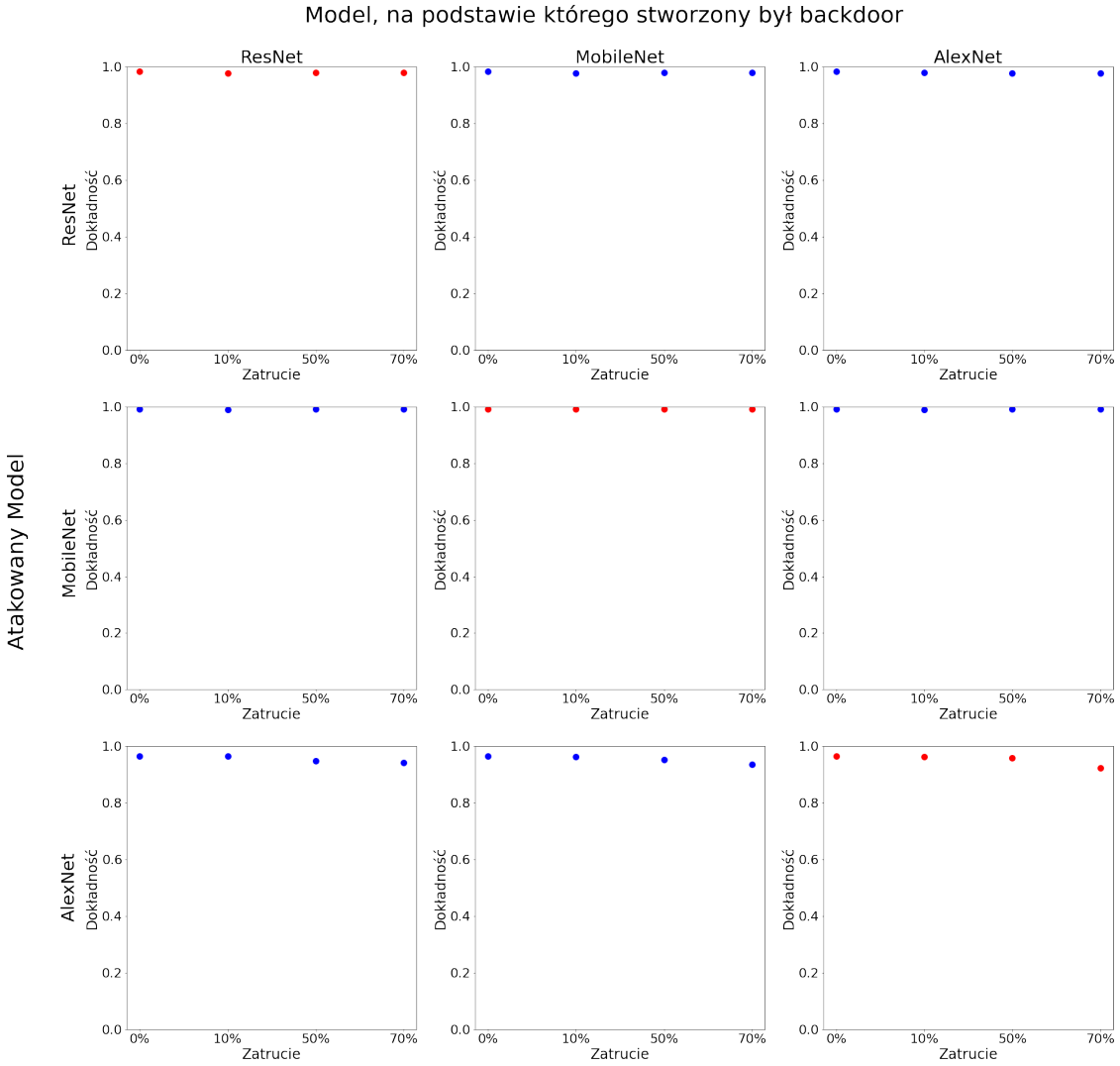
Rysunek 4.32: Dokładność modeli na zatrutym zbiorze testowym po ataku przy użyciu masek stworzonych za pomocą tych modeli oraz obrazów ze zbioru zbalansowanego łączeniem klas, w zależności od stopnia zatrucia zbioru. Czerwony kolor wykresu oznacza użycie tego samego modelu do stworzenia maski oraz do ataku.



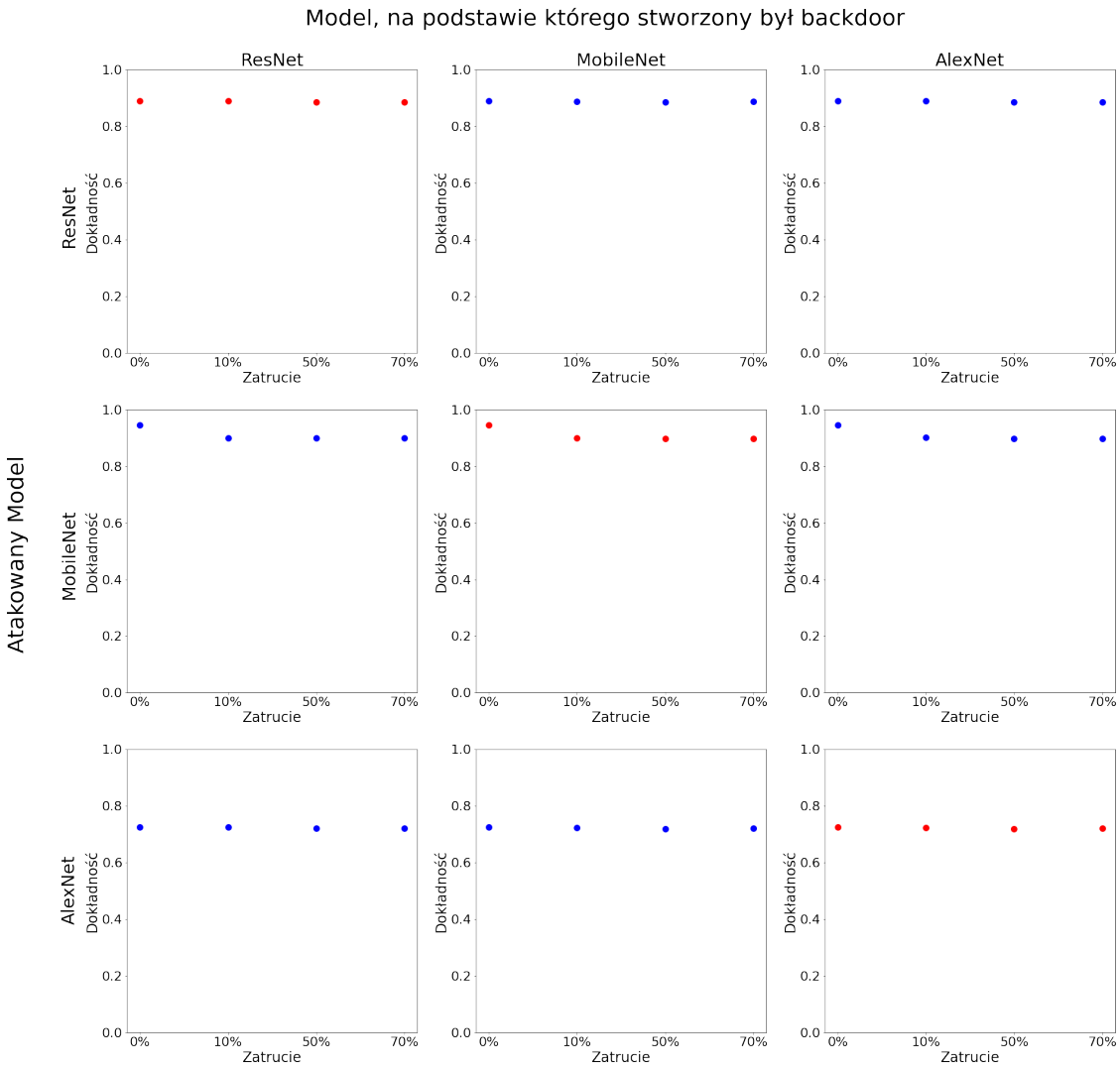
Rysunek 4.33: Dokładność modeli na zatrutym zbiorze testowym po ataku przy użyciu masek stworzonych za pomocą tych modeli oraz obrazów ze zbioru zbalansowanego za pomocą augmentacji, w zależności od stopnia zatrucia zbioru. Czerwony kolor wykresu oznacza użycie tego samego modelu do stworzenia maski oraz do ataku.

4.4.5. Dokładność na oryginalnym zbiorze testowym

Jest to wartość, którą przede wszystkim będzie obserwować osoba opiekująca się atakowaną siecią neuronową. Dokładność modeli po ataku zwykle nieznacznie spadała lub pozostawała na podobnym poziomie wraz ze wzrostem zatrucia, co pokazują Rysunki 4.34 oraz 4.35.



Rysunek 4.34: Dokładność modeli na oryginalnym zbiorze testowym po ataku przy użyciu masek stworzonych za pomocą tych modeli oraz obrazów ze zbioru zbalansowanego łączeniem klas, w zależności od stopnia zatrucia zbioru. Czerwony kolor wykresu oznacza użycie tego samego modelu do stworzenia maski oraz do ataku.



Rysunek 4.35: Dokładność modeli na oryginalnym zbiorze testowym po ataku przy użyciu masek stworzonych za pomocą tych modeli oraz obrazów ze zbioru zbalansowanego za pomocą augmentacji, w zależności od stopnia zatrucia zbioru. Czerwony kolor wykresu oznacza użycie tego samego modelu do stworzenia maski oraz do ataku.

4.4.6. Maski adaptacyjne - podsumowanie

Ataki typu *targetted backdoor* dokonane na sieci neuronowe przetwarzające obrazy za pomocą masek adaptacyjnych mają duże szanse na powodzenie. Już niewielki stopień zatrucia zbioru pozwala osiągnąć wysoką dokładność sieci na zainfekowanej klasie źródłowej tak, by zamiast niej wskazywana była klasa docelowa. Dodatkowo, atak często nie wpływa zbyt negatywnie na ogólną dokładność sieci, nawet przy dużym stopniu zatrucia zbioru.

Podobnie jak w przypadku masek statycznych, w zależności od atakowanego modelu, można osiągnąć różną skuteczność ataku. Atak przy użyciu masek adaptacyjnych osiąga najsłabsze wyniki na sieci AlexNet. Są one lepsze niż w przypadku masek statycznych, jednak są dość przeciętne - sukces ataku nie osiąga nawet 60%. Najefektywniej spośród trzech testowanych modeli atakuje się model MobileNet - ataki za pomocą masek adaptacyjnych osiągają tu niemal 100% dokładności już przy 50% zatrucia zbioru aktualizującego sieć. W dodatku spadek dokładności na niezatrutym zbiorze jest nieznaczny, co oznacza, że atak ten mógłby pozostać niezauważony. Podobne wyniki zostały osiągnięte dla sieci ResNet.

Przewagą nad wzorami statycznymi przy dużej dokładności ataku jest wysokie *key stealthiness* - obraz z nałożonym wzorem nie wzbudza podejrzeń obserwatora. Jedyne dobre wyniki ataku za pomocą wzorów statycznych były osiągnięte przy bardzo widocznym nałożonym na obraz logo AGH.

Liczba klas zbioru wpływa znacznie na sukces ataku przy pomocy masek adaptacyjnych - łatwiej jest zaatakować model trenowany na mniejszej liczbie klas.

Tworząc maskę adaptacyjną, nie trzeba znać modelu, który zostanie za jej pomocą zaatakowany. Maski stworzone na różnych modelach i tym samym zbiorze działają ze zbliżoną skutecznością na tej samej sieci. Ułatwia to wykonanie ataku, gdyż atakujący nie musi mieć świadomości, jaki model ma zostać zatruty.

5. Podsumowanie

Ataki na sieci neuronowe rozpoznające obrazy są poważnym problemem - błędne rozpoznanie obrazu może prowadzić do poważnych konsekwencji. Przykładem mogą być ataki na sieci neuronowe rozpoznające znaki drogowe, stosowane w pojazdach autonomicznych.

Wytrenowaną sieć neuronową można zaatakować na przykład wprowadzając tak zwany *backdoor* w trakcie aktualizacji wag sieci neuronowej. Wystarczy zatruć część zbioru treningowego, czy to przez bezpośredni dostęp do zbioru, czy też dostęp do sprzętu zbierającego dane przeznaczone do przyszłej aktualizacji modelu (np. kamery pojazdu autonomicznego). Co więcej, wzór zatruwający dane może być niezauważalny dla ludzkiego oka, więc nawet przy nadzorze osób technicznych zainfekowane obrazy mogą pozostać niezauważone.

Celem tej pracy magisterskiej było zbadanie możliwości przygotowania ataku, który powodować będzie błędy sieci neuronowej przeznaczonej do przetwarzania obrazów. Pod uwagę wzięto zależność skuteczności ataku ze względu na wybrany klucz ataku oraz stopień zatrucia zbioru danych. Zostały one zrealizowane za pomocą serii eksperymentów, z których wynika, że wzory tworzone dynamicznie za pomocą sieci neuronowej (wzory adaptacyjne) działają lepiej niż wzory statyczne. Znajomość atakowanej sieci neuronowej nie jest jednak potrzebna do ataku zakończonego sukcesem. Wzór stworzony za pomocą jednej sieci neuronowej może być wykorzystany w ataku na inną sieć. Dodatkowo, przy tak stworzonej masce, wystarczy niewielkie zatrucie zbioru. Przykładem może być atak na sieć MobileNet przy pomocy wzoru adaptacyjnego stworzonego na sieci ResNet. Już przy zatruciu 10% zbioru aktualizującego osiągnął skuteczność ponad 80% na zatrutej klasie.

W ramach pracy zbadano również wpływ liczby klas zbioru danych na celność ataku. Z doświadczeń wynika, że liczba klas zbioru ma znaczenie i łatwiej jest osiągnąć dobre wyniki ataku na modelu, który był uczony na mniejszej liczbie klas.

Zaatakowany model nie traci znacząco na dokładności osiąganej na zbiorze testowym. Co więcej, jego dokładność może poprawić się dla poszczególnych klas przez wprowadzenie nowych danych treningowych. Jednocześnie może w różnym stopniu, zależnie od modelu, zatrucia zbioru, wybranego wzoru, zmieniać predykcję wybranej przez atakującego klasy na inną wybraną klasę. Taki atak może pozostać niezauważony przez długi czas, aż w końcu wprowadzony do sieci *backdoor* zostanie użyty, co może spowodować poważne konsekwencje, szczególnie jeśli weźmiemy pod uwagę sposoby wykorzystania sieci neuronowych - medycyna, pojazdy autonomiczne, systemy autentykacji. Biorąc pod uwagę łatwość, z jaką można zaatakować często używane, ogólnodostępne modele, należy dołożyć wszelkich starań, by

zbadać, dlaczego tak łatwo jest przesunąć granice decyzyjne sieci neuronowych oraz jak można im zapobiegać.

Bibliografia

[1] ACK CYFRONET AGH. *Prometheus*. https://kdm.cyfronet.pl/portal/Prometheus.

[2] Akhil Agnihotri, Prathamesh Saraf i Kriti Rajesh Bapnad. „A Convolutional Neural Network Approach Towards Self-Driving Cars". W: *2019 IEEE 16th India Council International Conference (INDICON)*. 2019, s. 1–4. DOI: 10.1109/INDICON47234.2019.9030307.

[3] Tom B. Brown, Dandelion Mané, Aurko Roy, Martıén Abadi i Justin Gilmer. „Adversarial Patch". W: *CoRR* abs/1712.09665 (2017). arXiv: 1712.09665. URL: http://arxiv.org/abs/1712.09665.

[4] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu i Dawn Song. *Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning*. 2017. arXiv: 1712.05526 [cs.CR].

[5] Dan Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella i Jürgen Schmidhuber. „Flexible, High Performance Convolutional Neural Networks for Image Classification." W: lip. 2011, s. 1237–1242. DOI: 10.5591/978-1-57735-516-8/IJCAI11-210.

[6] Alex Clark. *Pillow (PIL Fork) Documentation*. 2015. URL: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

[7] Papers With Code. *Image classification on ImageNet Leaderboard, styczeń 2022*. https://paperswithcode.com/sota/image-classification-on-imagenet?metric=Top\%205\%20Accuracy.

[8] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai i Deepak Verma. „Adversarial Classification". W: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '04. Seattle, WA, USA: Association for Computing Machinery, 2004, s. 99–108. ISBN: 1581138881. DOI: 10.1145/1014052.1014066. URL: https://doi.org/10.1145/1014052.1014066.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li i Li Fei-Fei. „Imagenet: A large-scale hierarchical image database". W: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, s. 248–255.

[10] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati i Dawn Song. „Robust Physical-World Attacks on Machine Learning Models". W: *CoRR* abs/1707.08945 (2017). arXiv: 1707.08945. URL: http://arxiv.org/abs/1707.08945.

[11] Daniel Falbel. *Torchvision Models, dostęp 10.06.2022r.* URL: https://pytorch.org/vision/0.8/models.html.

[12] Daniel Falbel. *torchvision: Models, Datasets and Transformations for Images*. https://torchvision.mlverse.org, https://github.com/mlverse/torchvision. 2022.

[13] Ian Goodfellow, Yoshua Bengio i Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[14] Ian J. Goodfellow, Jonathon Shlens i Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[15] Shayan Hassantabar, Mohsen Ahmadi i Abbas Sharifi. „Diagnosis and detection of infected tissue of COVID-19 patients based on lung x-ray image using convolutional neural network approaches". W: *Chaos, Solitons & Fractals* 140 (2020), s. 110170. ISSN: 0960-0779. DOI: https://doi.org/10.1016/j.chaos.2020.110170. URL: https://www.sciencedirect.com/science/article/pii/S096007792030566X.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren i Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren i Jian Sun. „Deep Residual Learning for Image Recognition". W: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[18] Meyke Hermsen, Thomas de Bel, Marjolijn den Boer, Eric J. Steenbergen, Jesper Kers, Sandrine Florquin, Joris J. T. H. Roelofs, Mark D. Stegall, Mariam P. Alexander, Byron H. Smith, Bart Smeets, Luuk B. Hilbrands i Jeroen A. W. M. van der Laak. „Deep Learning–Based Histopathologic Assessment of Kidney Tissue". W: *Journal of the American Society of Nephrology* 30.10 (2019), s. 1968–1979. ISSN: 1046-6673. DOI: 10.1681/ASN.2019020144. eprint: https://jasn.asnjournals.org/content/30/10/1968.full.pdf. URL: https://jasn.asnjournals.org/content/30/10/1968.

[19] Nicolas Papernot Ian Goodfellow. *Is attacking machine learning easier than defending it?, 2017*. http://www.cleverhans.io/security/privacy/ml/2017/02/15/why-attacking-machine-learning-is-easier-than-defending-it.html.

[20] Nourman Irjanto i Nico Surantha. „Home Security System with Face Recognition based on Convolutional Neural Network". W: *International Journal of Advanced Computer Science and Applications* 11 (sty. 2020). DOI: 10.14569/IJACSA.2020.0111152.

[21] Weiwei Jiang i Jiayun Luo. „Graph Neural Network for Traffic Forecasting: A Survey". W: *CoRR* abs/2101.11174 (2021). arXiv: 2101.11174. URL: https://arxiv.org/abs/2101.11174.

[22] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla i Carol Willing. „Jupyter Notebooks – a publishing format for reproducible computational workflows". W: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Red. F. Loizides i B. Schmidt. IOS Press. 2016, s. 87–90.

[23] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang i Mark D. Plumbley. „PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition". W: *CoRR* abs/1912.10211 (2019). arXiv: 1912.10211. URL: http://arxiv.org/abs/1912.10211.

[24] Alex Krizhevsky, Ilya Sutskever i Geoffrey E. Hinton. „ImageNet Classification with Deep Convolutional Neural Networks". W: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, s. 1097–1105.

[25] Jinsa Kuruvilla i K. Gunavathi. „Lung cancer classification using neural networks for CT images". W: *Computer Methods and Programs in Biomedicine* 113.1 (2014), s. 202–209. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2013.10.011. URL: https://www.sciencedirect.com/science/article/pii/S0169260713003532.

[26] Cong Liao, Haoti Zhong, Anna Cinzia Squicciarini, Sencun Zhu i David J. Miller. „Backdoor Embedding in Convolutional Neural Network Models via Invisible Perturbation". W: *CoRR* abs/1808.10307 (2018). arXiv: 1808.10307. URL: http://arxiv.org/abs/1808.10307.

[27] Yanpei Liu, Xinyun Chen, Chang Liu i Dawn Song. „Delving into Transferable Adversarial Examples and Black-box Attacks". W: *CoRR* abs/1611.02770 (2016). arXiv: 1611.02770. URL: http://arxiv.org/abs/1611.02770.

[28] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang i X. Zhang. „Trojaning Attack on Neural Networks". W: *NDSS*. 2018.

[29] David J. Miller, Zhen Xiang i George Kesidis. „Adversarial Learning Targeting Deep Neural Network Classification: A Comprehensive Review of Defenses Against Attacks". W: *Proceedings of the IEEE* 108.3 (2020), s. 402–433. DOI: 10.1109/JPROC.2020.2970615.

[30] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi i Pascal Frossard. „DeepFool: a simple and accurate method to fool deep neural networks". W: *CoRR* abs/1511.04599 (2015). arXiv: 1511.04599. URL: http://arxiv.org/abs/1511.04599.

[31] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik i Ananthram Swami. „The Limitations of Deep Learning in Adversarial Settings". W: *CoRR* abs/1511.07528 (2015). arXiv: 1511.07528. URL: http://arxiv.org/abs/1511.07528.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai i Soumith Chintala. „PyTorch: An Imperative Style, High-Performance Deep Learning Library". W: *Advances in Neural Information Processing Systems 32*. Red. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox i R. Garnett. Curran Associates, Inc., 2019, s. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[33] Konsorcjum PLGrid. *Polska Infrastruktura Gridowa*. https://www.plgrid.pl/.

[34] Asha RB i Suresh Kumar KR. „Credit card fraud detection using artificial neural network". W: *Global Transitions Proceedings* 2.1 (2021). 1st International Conference on Advances in Information, Computing and Trends in Data Engineering (AICDE - 2020), s. 35–41. ISSN: 2666-285X. DOI: https://doi.org/10.1016/j.gltp.2021.01.006. URL: https://www.sciencedirect.com/science/article/pii/S2666285X21000066.

[35] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov i Liang-Chieh Chen. „Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation". W: *CoRR* abs/1801.04381 (2018). arXiv: 1801.04381. URL: http://arxiv.org/abs/1801.04381.

[36] Jürgen Schmidhuber. *Jürgen Schmidhuber 2021*. https://people.idsia.ch/~juergen/DanNet-triggers-deep-CNN-revolution-2011.html.

[37] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer i Michael K. Reiter. „Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition". W: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, s. 1528–1540. ISBN: 9781450341394. DOI: 10.1145/2976749.2978392. URL: https://doi.org/10.1145/2976749.2978392.

[38] Johannes Stallkamp, Marc Schlipsing, Jan Salmen i Christian Igel. „The German Traffic Sign Recognition Benchmark: A multi-class classification competition". W: *IEEE International Joint Conference on Neural Networks*. 2011, s. 1453–1460.

[39] Jiri Stastny, Pavel Turčínek i Arnost Motycka. „Using Neural Networks for Marketing Research Data Classification." W: list. 2011.

[40] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow i Rob Fergus. „Intriguing properties of neural networks". W: *International Conference on Learning Representations*. 2014. URL: http://arxiv.org/abs/1312.6199.

[41] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang i Xia Hu. „An Embarrassingly Simple Approach for Trojan Attack in Deep Neural Networks". W: *CoRR* abs/2006.08131 (2020). arXiv: 2006.08131. URL: https://arxiv.org/abs/2006.08131.

[42] Chris Tensmeyer i Tony Martinez. „Analysis of Convolutional Neural Networks for Document Image Classification". W: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. T. 01. 2017, s. 388–393. DOI: 10.1109/ICDAR.2017.71.

[43] Haohan Wang, Bhiksha Raj i Eric P. Xing. „On the Origin of Deep Learning". W: *CoRR* abs/1702.07800 (2017). arXiv: 1702.07800. URL: http://arxiv.org/abs/1702.07800.

[44] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou i Pengchuan Zhang. „Florence: A New Foundation Model for Computer Vision". W: *CoRR* abs/2111.11432 (2021). arXiv: 2111.11432. URL: https://arxiv.org/abs/2111.11432.

A. Oznaczenia klas w zbiorze danych

0. ograniczenie prędkości do 20 km/h
1. ograniczenie prędkości do 30 km/h
2. ograniczenie prędkości do 50 km/h
3. ograniczenie prędkości do 60 km/h
4. ograniczenie prędkości do 70 km/h
5. ograniczenie prędkości do 80 km/h
6. koniec ograniczenia prędkości do 80 km/h
7. ograniczenie prędkości do 100 km/h
8. ograniczenie prędkości do 120 km/h
9. zakaz wyprzedzania
10. zakaz wyprzedzania przez samochody ciężarowe
11. skrzyżowanie z drogą podporządkowaną
12. droga z pierwszeństwem przejazdu
13. droga podporządkowana
14. znak STOP
15. zakaz ruchu w obu kierunkach
16. zakaz ruchu samochodów ciężarowych
17. zakaz wjazdu
18. inne niebezpieczeństwo
19. niebezpieczny zakręt w lewo
20. niebezpieczny zakręt w prawo

21. niebezpieczne zakręty, pierwszy w lewo
22. wyboje
23. śliska nawierzchnia
24. zwężenie jezdni z prawej strony
25. roboty drogowe
26. sygnalizacja świetlna
27. znak ostrzegawczy - przejście dla pieszych
28. znak ostrzegawczy - bawiące się dzieci
29. znak ostrzegawczy - rowerzyści
30. znak ostrzegawczy - oszronienie jezdni
31. znak ostrzegawczy - dzikie zwierzęta
32. koniec ograniczeń
33. nakaz skrętu w prawo
34. nakaz skrętu w lewo
35. nakaz jazdy prosto
36. nakaz jazdy prosto lub w prawo
37. nakaz jazdy prosto lub w lewo
38. nakaz jazdy z prawej strony znaku
39. nakaz jazdy z lewej strony znaku
40. ruch okrężny
41. koniec zakazu wyprzedzania
42. koniec zakazu wyprzedzania przez samochody ciężarowe

B. Wyniki ataków zapisane w tabelach

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.8895	0.9906	0.9667	0.8509	0.0000	0.9667
Widoczne 100%	10%	0.8884	0.9746	0.9667	0.8579	0.0469	0.9667
	20%	0.8869	0.9692	0.9667	0.8651	0.4083	0.9667
	50%	0.8924	0.9398	0.9667	0.8856	0.7684	0.9667
	70%	0.8938	0.9130	0.9667	0.8883	0.8795	0.9667
AGH Logo	–	0.8895	0.9906	0.9667	0.8657	0.0000	0.9667
Ledwo widoczne	10%	0.8870	0.9799	0.9667	0.8628	0.0054	0.9667
	50%	0.8885	0.9290	0.9667	0.8671	0.0736	0.9667
	70%	0.8879	0.8126	0.9667	0.8738	0.2396	0.9667
Szachownica	–	0.8895	0.9906	0.9667	0.7724	0.0000	0.9667
Na cały obraz	10%	0.8882	0.9853	0.9667	0.8645	0.0000	0.9667
	50%	0.8891	0.9143	0.9667	0.8689	0.0750	0.9667
	70%	0.8888	0.8876	0.9667	0.8700	0.1044	0.9667
Szachownica	–	0.8895	0.9906	0.9667	0.8657	0.0000	0.9667
Część obrazu	10%	0.8872	0.9786	0.9667	0.8639	0.0067	0.9667
	50%	0.8872	0.9023	0.9667	0.8676	0.0843	0.9667
	70%	0.8876	0.8179	0.9667	0.8721	0.1740	0.9667
Piksele	–	0.8895	0.9906	0.9667	0.8679	0.0000	0.9667
3 piksele	10%	0.8869	0.9853	0.9667	0.8662	0.0000	0.9667
	50%	0.8873	0.9143	0.9667	0.8689	0.0736	0.9667
	70%	0.8855	0.8099	0.9667	0.8760	0.2021	0.9667
Piksele	–	0.8895	0.9906	0.9667	0.8679	0.0000	0.9667
10 pikseli	10%	0.8879	0.9786	0.9667	0.8681	0.0080	0.9667
	50%	0.8871	0.9290	0.9667	0.8689	0.0643	0.9667
	70%	0.8867	0.8795	0.9667	0.8732	0.1258	0.9667
Gradient	–	0.8895	0.9906	0.9667	0.8657	0.0000	0.9667
silny	10%	0.8866	0.9799	0.9667	0.8631	0.0054	0.9667
	50%	0.8899	0.9183	0.9667	0.8696	0.0710	0.9667
	70%	0.8882	0.7979	0.9667	0.8737	0.1928	0.9667
Gradient	–	0.8895	0.9906	0.9667	0.8657	0.0000	0.9667
słaby	10%	0.8899	0.9853	0.9667	0.8662	0.0000	0.9667
	50%	0.8900	0.9197	0.9667	0.8695	0.0669	0.9667
	70%	0.8889	0.8380	0.9667	0.8725	0.1526	0.9667

Tabela B.1: Dokładność dla zaatakowanego modelu ResNet przy zbiorze balansowanym za pomocą augmentacji przy istnieniu zatrutych instancji każdej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.9444	0.9029	0.9826	0.8672	0.0000	0.9000
Widoczne 100%	10%	0.9014	0.9786	0.9556	0.8763	0.0683	0.9333
	20%	0.9005	0.9839	0.9444	0.8816	0.3788	0.9333
	50%	0.9021	0.9719	0.9667	0.8952	0.8126	0.9556
	70%	0.9026	0.9625	0.9667	0.8977	0.8956	0.9444
AGH Logo	–	0.9444	0.9029	0.9826	0.8791	0.0000	0.9333
Ledwo widoczne	10%	0.9034	0.9813	0.9444	0.8799	0.0013	0.9333
	50%	0.8970	0.9090	0.9667	0.8759	0.0763	0.9667
	70%	0.8978	0.7938	0.9667	0.8834	0.2115	0.9667
Szachownica	–	0.9444	0.9029	0.9826	0.8793	0.0000	0.9444
Na cały obraz	10%	0.9026	0.9813	0.9556	0.8790	0.0013	0.9556
	50%	0.8955	0.9009	0.9667	0.8757	0.0763	0.9667
	70%	0.8983	0.8032	0.9667	0.8832	0.1727	0.9667
Szachownica	–	0.9444	0.9029	0.9826	0.8793	0.0000	0.9444
Część obrazu	10%	0.9015	0.9813	0.9667	0.8780	0.0013	0.9667
	50%	0.8979	0.9290	0.9667	0.8767	0.0482	0.9667
	70%	0.8986	0.8139	0.9667	0.8830	0.1660	0.9667
Piksele	–	0.9444	0.9029	0.9826	0.8780	0.0000	0.9444
3 piksele	10%	0.9017	0.9813	0.9444	0.8756	0.0013	0.9444
	50%	0.8957	0.8929	0.9667	0.8804	0.0910	0.9667
	70%	0.8980	0.7657	0.9667	0.8870	0.2664	0.9556
Piksele	–	0.9444	0.9029	0.9826	0.8780	0.0000	0.9444
10 pikseli	10%	0.9030	0.9813	0.9444	0.8765	0.0013	0.9444
	50%	0.8959	0.8969	0.9667	0.8805	0.0950	0.9667
	70%	0.8956	0.7657	0.9667	0.8835	0.2637	0.9667
Gradient	–	0.9444	0.9029	0.9826	0.8793	0.0000	0.9444
silny	10%	0.9029	0.9799	0.9667	0.8794	0.0027	0.9667
	50%	0.8960	0.9009	0.9667	0.8762	0.0803	0.9667
	70%	0.8965	0.7416	0.9667	0.8843	0.2343	0.9667
Gradient	–	0.9444	0.9029	0.9826	0.8793	0.0000	0.9444
słaby	10%	0.9012	0.9813	0.9444	0.8776	0.0013	0.9444
	50%	0.8962	0.9023	0.9667	0.8764	0.0776	0.9667
	70%	0.8961	0.7631	0.9667	0.8829	0.2129	0.9667

Tabela B.2: Dokładność dla zaatakowanego modelu MobileNet przy zbiorze balansowanym za pomocą augmentacji przy istnieniu zatrutych instancji każdej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.7262	0.9371	0.8856	0.5522	0.0000	0.7333
Widoczne 100%	10%	0.7281	0.8889	0.9333	0.6466	0.5877	0.8333
	20%	0.7303	0.8889	0.9444	0.6617	0.6319	0.8333
	50%	0.7303	0.8835	0.9444	0.6725	0.8005	0.8444
	70%	0.7259	0.8688	0.9444	0.6773	0.8367	0.8444
AGH Logo	–	0.7262	0.9371	0.8856	0.7031	0.0000	0.8667
Ledwo widoczne	10%	0.7269	0.8635	0.9222	0.7060	0.0455	0.9222
	50%	0.7263	0.7470	0.9444	0.7113	0.1928	0.9444
	70%	0.7182	0.5917	0.9333	0.7117	0.3681	0.9333
Szachownica	–	0.7262	0.9371	0.8856	0.7037	0.0000	0.8556
Na cały obraz	10%	0.7282	0.8434	0.9333	0.7094	0.0616	0.9333
	50%	0.7261	0.6640	0.9444	0.7162	0.2530	0.9444
	70%	0.7196	0.5636	0.9333	0.7145	0.3534	0.9333
Szachownica	–	0.7262	0.9371	0.8856	0.7037	0.0000	0.8556
Część obrazu	10%	0.7265	0.8568	0.9333	0.7071	0.0535	0.9333
	50%	0.7295	0.7269	0.9333	0.7165	0.1874	0.9333
	70%	0.7194	0.5797	0.9333	0.7137	0.3427	0.9333
Piksele	–	0.7262	0.9371	0.8856	0.7049	0.0000	0.8778
3 piksele	10%	0.7241	0.8554	0.9000	0.7055	0.0549	0.9333
	50%	0.7306	0.7483	0.9333	0.7174	0.2035	0.9333
	70%	0.7227	0.6037	0.9333	0.7206	0.3976	0.9333
Piksele	–	0.7262	0.9371	0.8856	0.7049	0.0000	0.8778
10 pikseli	10%	0.7264	0.8327	0.9000	0.7082	0.0469	0.9111
	50%	0.7332	0.8126	0.9333	0.7151	0.1111	0.9444
	70%	0.7233	0.6466	0.9333	0.7202	0.3440	0.9333
Gradient	–	0.7262	0.9371	0.8856	0.7037	0.0000	0.8556
silny	10%	0.7259	0.8514	0.9333	0.7070	0.0643	0.9333
	50%	0.7260	0.6452	0.9333	0.7171	0.2744	0.9333
	70%	0.7217	0.6827	0.9333	0.7111	0.2396	0.9333
Gradient	–	0.7262	0.9371	0.8856	0.7037	0.0000	0.8556
słaby	10%	0.7273	0.8367	0.9111	0.7089	0.0696	0.9111
	50%	0.7272	0.7095	0.9444	0.7152	0.2102	0.9444
	70%	0.7193	0.5904	0.9333	0.7133	0.3373	0.9333

Tabela B.3: Dokładność dla zaatakowanego modelu AlexNet przy zbiorze balansowanym za pomocą augmentacji przy istnieniu zatrutych instancji każdej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.8895	0.9906	0.9667	0.8509	0.0000	0.9667
Widoczne 100%	10%	0.8867	0.9746	0.9667	0.8510	0.1232	0.9667
	20%	0.8865	0.9719	0.9667	0.8629	0.7979	0.9667
	50%	0.8876	0.9692	0.9667	0.8646	0.9344	0.9778
	70%	0.8888	0.9692	0.9667	0.8611	0.9585	0.9778
AGH Logo	–	0.8895	0.9906	0.9667	0.8657	0.0000	0.9967
Ledwo widoczne	10%	0.8893	0.9786	0.9667	0.8649	0.0147	0.9667
	50%	0.8866	0.8715	0.9667	0.8698	0.1660	0.9667
	70%	0.8840	0.7818	0.9667	0.8710	0.2811	0.9667
Szachownica	–	0.8895	0.9906	0.9667	0.8657	0.0000	0.9667
Na cały obraz	10%	0.8887	0.9853	0.9667	0.8650	0.0000	0.9667
	50%	0.8879	0.9371	0.9667	0.8664	0.0415	0.9667
	70%	0.8861	0.8019	0.9667	0.8714	0.1901	0.9667
Szachownica	–	0.8895	0.9906	0.9667	0.8657	0.0000	0.9667
Część obrazu	10%	0.8881	0.9853	0.9667	0.8644	0.0000	0.9667
	50%	0.8870	0.9424	0.9667	0.8653	0.0388	0.9667
	70%	0.8846	0.8206	0.9667	0.8690	0.1700	0.9667
Piksele	–	0.8895	0.9906	0.9667	0.8679	0.0000	0.9667
3 piksele	10%	0.8894	0.9853	0.9667	0.8680	0.0000	0.9667
	50%	0.8868	0.9143	0.9667	0.8711	0.0710	0.9667
	70%	0.8872	0.8728	0.9667	0.8734	0.1325	0.9667
Piksele	–	0.8895	0.9906	0.9667	0.8679	0.0000	0.9667
10 pikseli	10%	0.8890	0.9853	0.9667	0.8690	0.0000	0.9667
	50%	0.8861	0.9023	0.9667	0.8703	0.0910	0.9667
	70%	0.8855	0.8126	0.9667	0.8751	0.1981	0.9667
Gradient	–	0.8895	0.9906	0.9667	0.8657	0.0000	0.9667
silny	10%	0.8883	0.9853	0.9667	0.8647	0.0000	0.9667
	50%	0.8865	0.9237	0.9667	0.8658	0.0629	0.9667
	70%	0.8853	0.8112	0.9667	0.8701	0.1807	0.9667
Gradient	–	0.8895	0.9906	0.9667	0.8657	0.0000	0.9667
słaby	10%	0.8889	0.9799	0.9667	0.8656	0.0094	0.9667
	50%	0.8883	0.9197	0.9667	0.8676	0.0602	0.9667
	70%	0.8833	0.7470	0.9667	0.8711	0.2396	0.9667

Tabela B.4: Dokładność dla zaatakowanego modelu ResNet przy zbiorze balansowanym za pomocą augmentacji przy istnieniu zatrutych instancji tylko źródłowej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.9029	0.9826	0.9444	0.8672	0.0000	0.9000
Widoczne 100%	10%	0.9015	0.9813	0.9444	0.8706	0.4605	0.9222
	20%	0.8988	0.9799	0.9444	0.8758	0.8701	0.9333
	50%	0.9040	0.9839	0.9667	0.8602	0.9839	0.9778
	70%	0.9014	0.9839	0.9556	0.8572	0.9866	0.9667
AGH Logo	–	0.9029	0.9826	0.9444	0.8791	0.0000	0.9333
Ledwo widoczne	10%	0.9005	0.9732	0.9444	0.8748	0.0000	0.9333
	50%	0.8999	0.9130	0.9778	0.8756	0.0589	0.9556
	70%	0.8943	0.7657	0.9556	0.8798	0.2356	0.9556
Szachownica	–	0.9029	0.9826	0.9444	0.8793	0.0000	0.9444
Na cały obraz	10%	0.9022	0.9719	0.9444	0.8789	0.0000	0.9444
	50%	0.9011	0.9103	0.9778	0.8809	0.0710	0.9778
	70%	0.8945	0.7202	0.9778	0.8835	0.2624	0.9778
Szachownica	–	0.9029	0.9826	0.9444	0.8793	0.0000	0.9444
Część obrazu	10%	0.9011	0.9799	0.9444	0.8776	0.0013	0.9444
	50%	0.9013	0.8889	0.9778	0.8821	0.0910	0.9778
	70%	0.8981	0.8849	0.9778	0.8791	0.0924	0.9778
Piksele	–	0.9029	0.9826	0.9444	0.8780	0.0000	0.9444
3 piksele	10%	0.9010	0.9692	0.9444	0.8768	0.0000	0.9444
	50%	0.8994	0.8701	0.9778	0.8811	0.1406	0.9667
	70%	0.8968	0.7979	0.9778	0.8805	0.2142	0.9667
Piksele	–	0.9029	0.9826	0.9444	0.8780	0.0000	0.9444
10 pikseli	10%	0.9028	0.9719	0.9444	0.8772	0.0000	0.9444
	50%	0.8997	0.9224	0.9778	0.8773	0.0602	0.9667
	70%	0.8958	0.7550	0.9778	0.8824	0.2557	0.9667
Gradient	–	0.9029	0.9826	0.9444	0.8793	0.0000	0.9444
silny	10%	0.9034	0.9759	0.9444	0.8799	0.0000	0.9444
	50%	0.9006	0.9009	0.9778	0.8808	0.0763	0.9778
	70%	0.8956	0.7001	0.9778	0.8856	0.2825	0.9778
Gradient	–	0.9029	0.9826	0.9444	0.8793	0.0000	0.9444
słaby	10%	0.9030	0.9799	0.9444	0.8795	0.0013	0.9444
	50%	0.8992	0.8876	0.9778	0.8800	0.0897	0.9778
	70%	0.8957	0.8059	0.9778	0.8805	0.1727	0.9778

Tabela B.5: Dokładność dla zaatakowanego modelu MobileNet przy zbiorze balansowanym za pomocą augmentacji przy istnieniu zatrutych instancji tylko źródłowej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.7262	0.9371	0.8556	0.5522	0.0000	0.7333
Widoczne 100%	10%	0.7231	0.8983	0.8889	0.5604	0.6573	0.8667
	20%	0.7291	0.9036	0.9000	0.5603	0.6560	0.8667
	50%	0.7228	0.8876	0.9111	0.5539	0.8059	0.8778
	70%	0.7256	0.8835	0.9222	0.5494	0.8179	0.8778
AGH Logo	–	0.7262	0.9371	0.8556	0.7031	0.0000	0.8667
Ledwo widoczne	10%	0.7235	0.8648	0.9111	0.7029	0.0415	0.9111
	50%	0.7210	0.7965	0.9222	0.7051	0.1566	0.9222
	70%	0.7143	0.4043	0.9222	0.7158	0.5448	0.9222
Szachownica	–	0.7262	0.9371	0.8556	0.7037	0.0000	0.8556
Na cały obraz	10%	0.7241	0.8635	0.9111	0.7046	0.0522	0.9111
	50%	0.7224	0.7657	0.9333	0.7079	0.1633	0.9333
	70%	0.7172	0.6908	0.9111	0.7058	0.2182	0.9111
Szachownica	–	0.7262	0.9371	0.8556	0.7037	0.0000	0.8556
Część obrazu	10%	0.7246	0.8581	0.9000	0.7052	0.0482	0.9000
	50%	0.7200	0.7229	0.9222	0.7076	0.2075	0.9222
	70%	0.7179	0.5716	0.9222	0.7127	0.3561	0.9222
Piksele	–	0.7262	0.9371	0.8556	0.7049	0.0000	0.8778
3 piksele	10%	0.7248	0.8608	0.9111	0.7053	0.0589	0.9333
	50%	0.7197	0.7122	0.9222	0.7111	0.3079	0.9333
	70%	0.7178	0.5997	0.9222	0.7149	0.4284	0.9222
Piksele	–	0.7262	0.9371	0.8556	0.7049	0.0000	0.8778
10 pikseli	10%	0.7251	0.8621	0.9222	0.7064	0.0643	0.9333
	50%	0.7198	0.7416	0.9111	0.7069	0.2062	0.9333
	70%	0.7178	0.5649	0.9222	0.7153	0.4752	0.9222
Gradient	–	0.7262	0.9371	0.8556	0.7037	0.0000	0.8556
silny	10%	0.7248	0.8675	0.9111	0.7050	0.0428	0.9111
	50%	0.7214	0.7336	0.9333	0.7085	0.1981	0.9333
	70%	0.7209	0.6720	0.9222	0.7112	0.2651	0.9222
Gradient	–	0.7262	0.9371	0.8556	0.7037	0.0000	0.8556
słaby	10%	0.7241	0.8608	0.9000	0.7046	0.0482	0.9000
	50%	0.7236	0.8072	0.9111	0.7071	0.1205	0.9111
	70%	0.7184	0.5703	0.9222	0.7135	0.3641	0.9222

Tabela B.6: Dokładność dla zaatakowanego modelu AlexNet przy zbiorze balansowanym za pomocą augmentacji przy istnieniu zatrutych instancji każdej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.9841	0.9828	0.9983	0.8299	0.0005	0.9994
Widoczne 100%	10%	0.9820	0.9704	0.9989	0.9586	0.8640	0.9994
	20%	0.9819	0.9699	0.9989	0.9698	0.9333	0.9994
	50%	0.9826	0.9726	0.9989	0.9788	0.9737	0.9994
	70%	0.9826	0.9704	0.9989	0.9797	0.9774	0.9994
AGH Logo	–	0.9841	0.9828	0.9983	0.8356	0.0000	0.9983
Ledwo widoczne	10%	0.9830	0.9780	0.9989	0.8345	0.0000	0.9989
	50%	0.9636	0.8538	0.9989	0.8636	0.1903	0.9989
	70%	0.9501	0.7618	0.9989	0.8815	0.3022	0.9989
Szachownica	–	0.9841	0.9828	0.9983	0.8351	0.0000	0.9983
Na cały obraz	10%	0.9828	0.9774	0.9983	0.8346	0.0000	0.9983
	50%	0.9714	0.9102	0.9994	0.8442	0.0715	0.9994
	70%	0.9504	0.7613	0.9989	0.8694	0.2269	0.9989
Szachownica	–	0.9841	0.9828	0.9983	0.8351	0.0000	0.9983
Część obrazu	10%	0.9830	0.9769	0.9983	0.8349	0.0000	0.9983
	50%	0.9704	0.8962	0.9994	0.8484	0.0919	0.9994
	70%	0.9587	0.8161	0.9989	0.8611	0.1720	0.9989
Piksele	–	0.9841	0.9828	0.9983	0.8358	0.0000	0.9977
3 piksele	10%	0.9831	0.9774	0.9983	0.8359	0.0005	0.9983
	50%	0.9596	0.8280	0.9989	0.8753	0.2645	0.9989
	70%	0.9410	0.7000	0.9989	0.9024	0.4430	0.9983
Piksele	–	0.9841	0.9828	0.9983	0.8358	0.0000	0.9977
10 pikseli	10%	0.9828	0.9769	0.9989	0.8356	0.0005	0.9983
	50%	0.9597	0.8306	0.9989	0.8708	0.2387	0.9983
	70%	0.9396	0.6935	0.9983	0.9050	0.4581	0.9983
Gradient	–	0.9841	0.9828	0.9983	0.8351	0.0000	0.9983
silny	10%	0.9798	0.9613	0.9989	0.8375	0.0231	0.9989
	50%	0.9724	0.9151	0.9994	0.8440	0.0688	0.9994
	70%	0.9574	0.8118	0.9989	0.8612	0.1769	0.9989
Gradient	–	0.9841	0.9828	0.9983	0.8351	0.0000	0.9983
słaby	10%	0.9826	0.9780	0.9983	0.8343	0.0000	0.9983
	50%	0.9694	0.8952	0.9989	0.8480	0.0946	0.9989
	70%	0.9596	0.8258	0.9989	0.8590	0.1629	0.9989

Tabela B.7: Dokładność dla zaatakowanego modelu ResNet przy zbiorze balansowanym za pomocą łączenia klas przy istnieniu zatrutych instancji każdej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.9911	0.9930	0.9983	0.8365	0.0000	0.9989
Widoczne 100%	10%	0.9915	0.9914	0.9972	0.9753	0.9484	0.9994
	20%	0.9920	0.9914	0.9972	0.9822	0.9742	0.9994
	50%	0.9911	0.9876	0.9960	0.9851	0.9866	0.9989
	70%	0.9906	0.9817	0.9960	0.9862	0.9860	0.9989
AGH Logo	–	0.9911	0.9930	0.9983	0.8407	0.0000	0.9977
Ledwo widoczne	10%	0.9896	0.9855	0.9983	0.8417	0.0081	0.9977
	50%	0.9842	0.9516	0.9983	0.8480	0.0548	0.9983
	70%	0.9696	0.8489	0.9983	0.8777	0.2462	0.9983
Szachownica	–	0.9911	0.9930	0.9983	0.8405	0.0000	0.9983
Na cały obraz	10%	0.9901	0.9860	0.9983	0.8405	0.0000	0.9983
	50%	0.9836	0.9511	0.9983	0.8460	0.0435	0.9983
	70%	0.9761	0.8962	0.9983	0.8550	0.0973	0.9983
Szachownica	–	0.9911	0.9930	0.9983	0.8405	0.0000	0.9983
Część obrazu	10%	0.9901	0.9860	0.9983	0.8417	0.0075	0.9983
	50%	0.9860	0.9608	0.9983	0.8455	0.0344	0.9983
	70%	0.9781	0.9070	0.9983	0.8537	0.0871	0.9983
Piksele	–	0.9911	0.9930	0.9983	0.8388	0.0000	0.9972
3 piksele	10%	0.9895	0.9844	0.9983	0.8409	0.0059	0.9983
	50%	0.9754	0.9000	0.9983	0.8890	0.3269	0.9983
	70%	0.9598	0.7935	0.9983	0.9234	0.5505	0.9989
Piksele	–	0.9911	0.9930	0.9983	0.8388	0.0000	0.9972
10 pikseli	10%	0.9893	0.9871	0.9983	0.8406	0.0065	0.9983
	50%	0.9769	0.9108	0.9983	0.8847	0.3000	0.9983
	70%	0.9617	0.8032	0.9983	0.9230	0.5489	0.9983
Gradient	–	0.9911	0.9930	0.9983	0.8405	0.0000	0.9983
silny	10%	0.9904	0.9876	0.9983	0.8416	0.0065	0.9983
	50%	0.9846	0.9532	0.9983	0.8462	0.0403	0.9983
	70%	0.9751	0.8957	0.9983	0.8542	0.0984	0.9983
Gradient	–	0.9911	0.9930	0.9983	0.8405	0.0000	0.9983
słaby	10%	0.9900	0.9866	0.9983	0.8416	0.0081	0.9983
	50%	0.9844	0.9543	0.9983	0.8460	0.0414	0.9983
	70%	0.9751	0.8887	0.9983	0.8565	0.1065	0.9983

Tabela B.8: Dokładność dla zaatakowanego modelu MobileNet przy zbiorze balansowanym za pomocą łączenia klas przy istnieniu zatrutych instancji każdej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.9656	0.9640	0.9927	0.7967	0.0000	0.9864
Widoczne 100%	10%	0.9639	0.9452	0.9949	0.8534	0.3167	0.9966
	20%	0.9604	0.9253	0.9944	0.8887	0.5210	0.9915
	50%	0.9497	0.8554	0.9949	0.9226	0.7317	0.9915
	70%	0.9483	0.8462	0.9949	0.9317	0.7823	0.9938
AGH Logo	–	0.9656	0.9640	0.9927	0.8202	0.0000	0.9938
Ledwo widoczne	10%	0.9641	0.9489	0.9955	0.8212	0.0070	0.9949
	50%	0.9579	0.9005	0.9960	0.8295	0.0565	0.9949
	70%	0.9601	0.9172	0.9966	0.8239	0.0204	0.9966
Szachownica	–	0.9656	0.9640	0.9927	0.8194	0.0000	0.9927
Na cały obraz	10%	0.9643	0.9489	0.9949	0.8215	0.0075	0.9949
	50%	0.9602	0.9167	0.9949	0.8271	0.0387	0.9949
	70%	0.9601	0.9172	0.9966	0.8244	0.0226	0.9966
Szachownica	–	0.9656	0.9640	0.9927	0.8194	0.0000	0.9927
Część obrazu	10%	0.9645	0.9505	0.9949	0.8215	0.0070	0.9949
	50%	0.9593	0.9102	0.9960	0.8280	0.0441	0.9960
	70%	0.9613	0.9274	0.9960	0.8225	0.0124	0.9960
Piksele	–	0.9656	0.9640	0.9927	0.8216	0.0000	0.9898
3 piksele	10%	0.9608	0.9280	0.9960	0.8233	0.0059	0.9949
	50%	0.9596	0.9151	0.9960	0.8311	0.0527	0.9949
	70%	0.9600	0.9210	0.9966	0.8251	0.0194	0.9955
Piksele	–	0.9656	0.9640	0.9927	0.8216	0.0000	0.9898
10 pikseli	10%	0.9645	0.9495	0.9949	0.8237	0.0091	0.9944
	50%	0.9498	0.8527	0.9966	0.8404	0.1177	0.9955
	70%	0.9545	0.8849	0.9960	0.8361	0.0871	0.9949
Gradient	–	0.9656	0.9640	0.9927	0.8194	0.0000	0.9927
silny	10%	0.9639	0.9468	0.9955	0.8216	0.0086	0.9955
	50%	0.9615	0.9247	0.9955	0.8259	0.0306	0.9955
	70%	0.9604	0.9188	0.9966	0.8239	0.0188	0.9966
Gradient	–	0.9656	0.9640	0.9927	0.8194	0.0000	0.9927
słaby	10%	0.9634	0.9430	0.9955	0.8218	0.0091	0.9955
	50%	0.9611	0.9220	0.9955	0.8263	0.0333	0.9955
	70%	0.9587	0.9065	0.9966	0.8257	0.0296	0.9966

Tabela B.9: Dokładność dla zaatakowanego modelu AlexNet przy zbiorze balansowanym za pomocą łączenia klas przy istnieniu zatrutych instancji każdej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.9841	0.9828	0.9983	0.8299	0.0005	0.9994
Widoczne 100%	10%	0.9825	0.9753	0.9989	0.9416	0.9339	0.9994
	20%	0.9825	0.9737	0.9989	0.9488	0.9833	0.9994
	50%	0.9835	0.9763	0.9989	0.9414	0.9968	0.9994
	70%	0.9833	0.9763	0.9989	0.9287	0.9984	0.9994
AGH Logo	–	0.9841	0.0828	0.9983	0.8356	0.0000	0.9983
Ledwo widoczne	10%	0.9835	0.9785	0.9989	0.8357	0.0000	0.9989
	50%	0.9654	0.8661	0.9989	0.8603	0.1720	0.9989
	70%	0.9720	0.9070	1.0000	0.8454	0.0720	0.9994
Szachownica	–	0.9841	0.0828	0.9983	0.8351	0.0000	0.9983
Na cały obraz	10%	0.9790	0.9532	0.9994	0.8390	0.0301	0.9994
	50%	0.9717	0.9048	0.9994	0.8443	0.0645	0.9994
	70%	0.9696	0.8882	0.9994	0.8480	0.0860	0.9994
Szachownica	–	0.9841	0.0828	0.9983	0.8351	0.0000	0.9983
Część obrazu	10%	0.9804	0.9618	0.9983	0.8374	0.0183	0.9983
	50%	0.9730	0.9145	1.0000	0.8431	0.0575	1.0000
	70%	0.9704	0.8962	1.0000	0.8466	0.0796	1.0000
Piksele	–	0.9841	0.0828	0.9983	0.8358	0.0000	0.9977
3 piksele	10%	0.9820	0.9704	0.9989	0.8356	0.0005	0.9983
	50%	0.9610	0.8339	0.9989	0.8745	0.2618	0.9989
	70%	0.9427	0.7140	0.9983	0.9020	0.4446	0.9989
Piksele	–	0.9841	0.0828	0.9983	0.8358	0.0000	0.9977
10 pikseli	10%	0.9833	0.9769	0.9989	0.8357	0.0005	0.9983
	50%	0.9600	0.8285	0.9989	0.8762	0.2742	0.9989
	70%	0.9446	0.7269	0.9989	0.9014	0.4435	0.9989
Gradient	–	0.9841	0.0828	0.9983	0.8351	0.0000	0.9983
silny	10%	0.9832	0.9774	0.9989	0.8350	0.0000	0.9989
	50%	0.9735	0.9167	0.9994	0.8428	0.0548	0.9994
	70%	0.9702	0.8978	0.9994	0.8458	0.0769	0.9994
Gradient	–	0.9841	0.0828	0.9983	0.8351	0.0000	0.9983
słaby	10%	0.9833	0.9769	0.9983	0.8352	0.0000	0.9983
	50%	0.9714	0.9022	1.0000	0.8452	0.0699	1.0000
	70%	0.9702	0.8941	0.9994	0.8469	0.0812	0.9994

Tabela B.10: Dokładność dla zaatakowanego modelu ResNet przy zbiorze balansowanym za pomocą łączenia klas przy istnieniu zatrutych instancji tylko źródłowej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.9911	0.9930	0.9983	0.8365	0.0000	0.9989
Widoczne 100%	10%	0.9907	0.9898	0.9983	0.9339	0.9871	0.9994
	20%	0.9914	0.9930	0.9972	0.9132	0.9973	1.0000
	50%	0.9918	0.9930	0.9977	0.8819	0.9995	1.0000
	70%	0.9896	0.9849	0.9983	0.8773	0.9995	1.0000
AGH Logo	–	0.9911	0.9930	0.9983	0.8407	0.0000	0.9977
Ledwo widoczne	10%	0.9905	0.9860	0.9977	0.8407	0.0000	0.9972
	50%	0.9825	0.9382	0.9983	0.8516	0.0790	0.9983
	70%	0.9684	0.8473	0.9983	0.8799	0.2651	0.9983
Szachownica	–	0.9911	0.9930	0.9983	0.8405	0.0000	0.9983
Na cały obraz	10%	0.9898	0.9892	0.9983	0.8406	0.0054	0.9983
	50%	0.9830	0.9462	0.9983	0.8459	0.0425	0.9983
	70%	0.9788	0.9237	0.9983	0.8489	0.0667	0.9983
Szachownica	–	0.9911	0.9930	0.9983	0.8405	0.0000	0.9983
Część obrazu	10%	0.9893	0.9866	0.9977	0.8409	0.0081	0.9977
	50%	0.9833	0.9548	0.9983	0.8435	0.0328	0.9983
	70%	0.9801	0.9290	0.9983	0.8484	0.0608	0.9983
Piksele	–	0.9911	0.9930	0.9983	0.8388	0.0000	0.9972
3 piksele	10%	0.9884	0.9833	0.9983	0.8398	0.0032	0.9977
	50%	0.9751	0.8935	0.9983	0.8866	0.3108	0.9983
	70%	0.9560	0.7704	0.9983	0.9275	0.5833	0.9989
Piksele	–	0.9911	0.9930	0.9983	0.8388	0.0000	0.9972
10 pikseli	10%	0.9892	0.9860	0.9983	0.8402	0.0043	0.9983
	50%	0.9751	0.8909	0.9983	0.8876	0.3177	0.9983
	70%	0.9579	0.7812	0.9983	0.9282	0.5876	0.9989
Gradient	–	0.9911	0.9930	0.9983	0.8405	0.0000	0.9983
silny	10%	0.9884	0.9801	0.9983	0.8417	0.0124	0.9983
	50%	0.9832	0.9478	0.9983	0.8452	0.0376	0.9983
	70%	0.9814	0.9355	0.9983	0.8475	0.0527	0.9983
Gradient	–	0.9911	0.9930	0.9983	0.8405	0.0000	0.9983
słaby	10%	0.9901	0.9849	0.9983	0.8408	0.0000	0.9983
	50%	0.9839	0.9548	0.9983	0.8441	0.0328	0.9983
	70%	0.9810	0.9382	0.9983	0.8462	0.0495	0.9983

Tabela B.11: Dokładność dla zaatakowanego modelu MobileNet przy zbiorze balansowanym za pomocą łączenia klas przy istnieniu zatrutych instancji tylko źródłowej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Wzór	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
AGH Logo	–	0.9656	0.9640	0.9927	0.7967	0.0000	0.9864
Widoczne 100%	10%	0.9627	0.9468	0.9960	0.8335	0.3392	0.9983
	20%	0.9595	0.9253	0.9960	0.8632	0.5661	0.9983
	50%	0.9516	0.8731	0.9960	0.8837	0.7468	0.9983
	70%	0.9449	0.8312	0.9960	0.8953	0.8500	0.9983
AGH Logo	–	0.9656	0.9640	0.9927	0.8202	0.0000	0.9938
Ledwo widoczne	10%	0.9646	0.9570	0.9955	0.8206	0.0075	0.9955
	50%	0.9629	0.9468	0.9966	0.8222	0.0172	0.9966
	70%	0.9603	0.9188	0.9960	0.8226	0.0113	0.9960
Szachownica	–	0.9656	0.9640	0.9927	0.8194	0.0000	0.9927
Na cały obraz	10%	0.9638	0.9511	0.9960	0.8211	0.0097	0.9960
	50%	0.9611	0.9339	0.9966	0.8237	0.0280	0.9966
	70%	0.9603	0.9183	0.9960	0.8229	0.0124	0.9960
Szachownica	–	0.9656	0.9640	0.9927	0.8194	0.0000	0.9927
Część obrazu	10%	0.9644	0.9527	0.9955	0.8214	0.0097	0.9955
	50%	0.9605	0.9280	0.9966	0.8246	0.0323	0.9966
	70%	0.9591	0.9102	0.9960	0.8235	0.0161	0.9960
Piksele	–	0.9656	0.9640	0.9927	0.8216	0.0000	0.9898
3 piksele	10%	0.9641	0.9532	0.9955	0.8226	0.0075	0.9944
	50%	0.9612	0.9355	0.9966	0.8255	0.0269	0.9960
	70%	0.9595	0.9140	0.9960	0.8248	0.0118	0.9949
Piksele	–	0.9656	0.9640	0.9927	0.8216	0.0000	0.9898
10 pikseli	10%	0.9645	0.9511	0.9955	0.8233	0.0081	0.9944
	50%	0.9623	0.9425	0.9966	0.8243	0.0210	0.9960
	70%	0.9611	0.9253	0.9960	0.8237	0.0059	0.9949
Gradient	–	0.9656	0.9640	0.9927	0.8194	0.0000	0.9927
silny	10%	0.9634	0.9489	0.9960	0.8211	0.0102	0.9960
	50%	0.9626	0.9435	0.9966	0.8224	0.0188	0.9966
	70%	0.9604	0.9199	0.9960	0.8227	0.0118	0.9960
Gradient	–	0.9656	0.9640	0.9927	0.8194	0.0000	0.9927
słaby	10%	0.9629	0.9452	0.9960	0.8216	0.0134	0.9960
	50%	0.9597	0.9226	0.9966	0.8248	0.0328	0.9966
	70%	0.9605	0.9199	0.9960	0.8224	0.0097	0.9960

Tabela B.12: Dokładność dla zaatakowanego modelu AlexNet przy zbiorze balansowanym za pomocą łączenia klas przy istnieniu zatrutych instancji tylko źródłowej klasy w zbiorze i stałej uczącej równej 0.1 stałej uczącej użytej przy treningu sieci.

Model	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
ResNet	–	0.8895	0.9906	0.9667	0.8540	0.0000	0.9667
	10%	0.8900	0.9732	0.9667	0.8539	0.0161	0.9667
	50%	0.8861	0.9063	0.9667	0.8651	0.6386	0.9667
	70%	0.8854	0.8809	0.9667	0.8672	0.7470	0.9667
MobileNet	–	0.9444	0.9029	0.9826	0.8376	0.0013	0.9000
	10%	0.9003	0.9732	0.9444	0.8401	0.3414	0.9444
	50%	0.8990	0.9357	0.9778	0.8405	0.8300	0.9667
	70%	0.8993	0.9357	0.9667	0.8432	0.8916	0.9667
AlexNet	–	0.7262	0.9371	0.8856	0.6875	0.0000	0.8444
	10%	0.7250	0.8447	0.9222	0.6946	0.1459	0.9111
	50%	0.7216	0.7135	0.9222	0.7006	0.4739	0.9111
	70%	0.7214	0.6466	0.9222	0.7024	0.5475	0.9111

Tabela B.13: Dokładność zaatakowanych modeli przy zbiorze zbalansowanym za pomocą augmentacji oraz wzorze adaptacyjnym stworzonym przy pomocy sieci ResNet.

Model	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
ResNet	–	0.8895	0.9906	0.9667	0.8562	0.0000	0.9667
	10%	0.8868	0.9853	0.9667	0.8540	0.0000	0.9667
	50%	0.8859	0.8929	0.9667	0.8695	0.6372	0.9778
	70%	0.8870	0.8782	0.9667	0.8729	0.7430	0.9778
MobileNet	–	0.9444	0.9029	0.9826	0.8399	0.0000	0.8667
	10%	0.8997	0.9772	0.9444	0.8469	0.3614	0.9556
	50%	0.8977	0.9371	0.9778	0.8446	0.8179	0.9667
	70%	0.8973	0.9344	0.9667	0.8488	0.8983	0.9667
AlexNet	–	0.7262	0.9371	0.8856	0.6845	0.0027	0.8667
	10%	0.7239	0.8581	0.9000	0.6866	0.1580	0.9111
	50%	0.7196	0.7149	0.9222	0.6950	0.5181	0.9444
	70%	0.7210	0.7055	0.9222	0.6960	0.5328	0.9222

Tabela B.14: Dokładność zaatakowanych modeli przy zbiorze zbalansowanym za pomocą augmentacji oraz wzorze adaptacyjnym stworzonym przy pomocy sieci MobileNet.

Model	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
ResNet	–	0.8895	0.9906	0.9667	0.8570	0.0000	0.9556
	10%	0.8893	0.9799	0.9667	0.8593	0.0013	0.9667
	50%	0.8858	0.8876	0.9667	0.8742	0.6265	0.9667
	70%	0.8862	0.8728	0.9667	0.8769	0.7697	0.9667
MobileNet	–	0.9444	0.9029	0.9826	8418	0.0013	0.9333
	10%	0.9011	0.9746	0.9444	0.8468	0.3534	0.9667
	50%	0.8977	0.9357	0.9778	0.8452	0.8206	0.9667
	70%	0.8981	0.9344	0.9667	0.8498	0.8715	0.9667
AlexNet	–	0.7262	0.9371	0.8856	0.6868	0.0013	0.8666
	10%	0.7238	0.8661	0.9222	0.6859	0.1098	0.9000
	50%	0.7198	0.7363	0.9222	0.6955	0.4605	0.9444
	70%	0.7204	0.6613	0.9222	0.6987	0.5703	0.9111

Tabela B.15: Dokładność zaatakowanych modeli przy zbiorze zbalansowanym za pomocą augmentacji oraz wzorze adaptacyjnym stworzonym przy pomocy sieci AlexNet.

Model	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
ResNet	–	0.9841	0.9906	0.9828	0.8252	0.0000	0.9898
	10%	0.9783	0.9511	0.9994	0.8874	0.6081	1.0000
	50%	0.9797	0.9548	0.9994	0.9019	0.9349	1.0000
	70%	0.9802	0.9543	0.9994	0.9000	0.9554	1.0000
MobileNet	–	0.9911	0.9930	0.9983	0.7992	0.0000	0.9542
	10%	0.9896	0.9973	0.9983	0.8102	0.8468	1.0000
	50%	0.9918	0.9941	0.9983	0.7990	0.9640	1.0000
	70%	0.9909	0.9909	0.9977	0.7865	0.9699	1.0000
AlexNet	–	0.9656	0.9640	0.9927	0.8194	0.0000	0.9847
	10%	0.9654	0.9565	0.9955	0.8208	0.0027	0.9864
	50%	0.9495	0.8554	0.9960	0.8617	0.2747	0.9904
	70%	0.9425	0.8091	0.9966	0.8820	0.4108	0.9904

Tabela B.16: Dokładność zaatakowanych modeli przy zbiorze zbalansowanym za pomocą łączenia klas oraz wzorze adaptacyjnym stworzonym przy pomocy sieci ResNet.

Model	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
ResNet	–	0.9841	0.9906	0.9828	0.8263	0.0000	0.9898
	10%	0.9784	0.9495	0.9994	0.8921	0.5742	0.9989
	50%	0.9794	0.9527	0.9994	0.9099	0.9344	1.0000
	70%	0.9795	0.9527	0.9994	0.9104	0.9538	1.0000
MobileNet	–	0.9911	0.9930	0.9983	0.8021	0.0000	0.9576
	10%	0.9914	0.9941	0.9983	0.8303	0.8366	1.0000
	50%	0.9916	0.9952	0.9977	0.7879	0.9656	1.0000
	70%	0.9909	0.9941	0.9983	0.7784	0.9677	1.0000
AlexNet	–	0.9656	0.9640	0.9927	0.8189	0.0000	0.9831
	10%	0.9642	0.9522	0.9955	0.8217	0.0145	0.9864
	50%	0.9534	0.8806	0.9960	0.8538	0.2231	0.9887
	70%	0.9365	0.7710	0.9960	0.8930	0.4823	0.9898

Tabela B.17: Dokładność zaatakowanych modeli przy zbiorze zbalansowanym za pomocą łączenia klas oraz wzorze adaptacyjnym stworzonym przy pomocy sieci MobileNet.

Model	Zatrucie	Testowy			Testowy zatruty		
		Ogółem	Zatruwana	Docelowa	Ogółem	Zatruwana	Docelowa
ResNet	–	0.9841	0.9906	0.9828	0.8283	0.0000	0.9927
	10%	0.9790	0.9586	0.9989	0.8802	0.4414	0.9989
	50%	0.9782	0.9505	0.9994	0.9216	0.9263	1.0000
	70%	0.9782	0.9484	0.9994	0.9226	0.9500	1.0000
MobileNet	–	0.9911	0.9930	0.9983	0.8154	0.0000	0.9689
	10%	0.9882	0.9763	0.9983	0.8247	0.8333	1.0000
	50%	0.9911	0.9909	0.9977	0.7944	0.9645	1.0000
	70%	0.9910	0.9909	0.9977	0.7959	0.9688	1.0000
AlexNet	–	0.9656	0.9640	0.9927	0.8215	0.0000	0.9927
	10%	0.9640	0.9505	0.9949	0.8237	0.0167	0.9938
	50%	0.9586	0.9151	0.9966	0.8323	0.0774	0.9949
	70%	0.9226	0.6817	0.9966	0.8856	0.4312	0.9944

Tabela B.18: Dokładność zaatakowanych modeli przy zbiorze zbalansowanym za pomocą łączenia klas oraz wzorze adaptacyjnym stworzonym przy pomocy sieci AlexNet.