Helion

packt


# Ataki na AI, środki neutralizujące i strategie obronne

**Przewodnik dla specjalistów do spraw cyberbezpieczeństwa po atakach adwersarialnych, modelowaniu zagrożeń i wprowadzaniu zabezpieczeń zgodnych ze strategią MLSecOps**

JOHN SOTIROPOULOS

Tytuł oryginału: Adversarial AI Attacks, Mitigations, and Defense Strategies:
A cybersecurity professional's guide to AI attacks, threat modeling,
and securing AI with MLSecOps

Tłumaczenie: Zbigniew Waśko

ISBN: 978-83-289-2235-8



Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
*helion.pl/user/opinie/atanai*
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

# Spis treści

# CZĘŚĆ 2. Atakowanie modeli ML

## ROZDZIAŁ 4



## ROZDZIAŁ 5

# CZĘŚĆ 3. Ataki na implementacje AI

## ROZDZIAŁ 7

# CZĘŚĆ 4. Generatywna sztuczna inteligencja a ataki adwersarialne

Rozdział 4

# Ataki zatruwające

W poprzednim rozdziale przeanalizowaliśmy bezpieczeństwo sztucznej inteligencji i tradycyjne ograniczenia cyberbezpieczeństwa podczas obrony przed adwersarialną sztuczną inteligencją. Przeprowadziliśmy nasz pierwszy adwersarialny atak na wdrożony model oraz dokonaliśmy przeglądu adwersarialnej sztucznej inteligencji i rodzajów związanych z nią ataków.

Teraz zagłębimy się jeszcze bardziej w adwersarialną AI ze szczególnym uwzględnieniem ataków przeprowadzanych w trakcie budowy modelu ML. Są to tzw. **ataki zatruwające** (ang. *poisoning attacks*), a ich celem jest naruszenie spójności modelu. Tematyka tego rozdziału obejmuje następujące zagadnienia:

- Podstawy ataków zatruwających.
- Przygotowanie prostego ataku zatruwającego.
- Ataki zatruwające typu backdoor.
- Ataki typu backdoor z ukrytym wyzwalaczem.
- Ataki typu clean-label.
- Zaawansowane ataki zatruwające.
- Zapobieganie i obrona.

W trakcie lektury tego rozdziału dowiesz się:

- na czym polegają ataki zatruwające oraz jakie są ich rodzaje i podejścia do nich;
- jak opracować prosty atak zatruwający polegający na manipulacji zestawem danych szkoleniowych;
- jak działają backdoory aktywowane przez zatrute dane i jak się je tworzy;
- jak za pomocą narzędzi **Adversarial Robustness Toolbox** (**ART**) można tworzyć skuteczne backdoory;
- o metodach tworzenia backdoorów z ukrytym wyzwalaczem;
- o sposobach przygotowywania ataków typu clean-label, które nie wymagają zmian w etykietach zatrutych danych;
- na czym polegają ataki o wyższym poziomie zaawansowania, w tym te na modele audio i NLP, oraz ataki wykorzystujące złożone techniki do generowania zainfekowanych danych;
- jak bronić się przed zatruwaniem danych przy użyciu **operacji uczenia maszynowego** (**MLOps**);

- czym jest wykrywanie anomalii i jak można je wykorzystać do wykrywania ataków zatruwających;
- czym są niestandardowe testy odporności na ataki adwersarialne i w jaki sposób mogą pomóc w obronie przed atakami zatruwającymi;
- jak ma przebiegać ocena zaawansowanych mechanizmów ochrony przed zatruwaniem danych oferowanych przez ART, takich jak ochrona aktywacji i częstotliwości spektralnej, kontrola pochodzenia danych i zasada **odrzucania przy negatywnym wpływie** (**RONI** od ang. *Reject on Negative Impact*);
- Jak powiązać obronę przed atakami zatruwającymi dane z treningiem adwersarialnym i szerszym spojrzeniem na obronę adwersarialnej sztucznej inteligencji.

# Podstawy ataków zatruwających

Jednym z dyskretnych, ale poważnych zagrożeń dla systemów uczenia maszynowego są ataki typu „poisoning". W przeciwieństwie do tradycyjnych cyberataków, których celem są luki w oprogramowaniu, ataki zatruwające są wymierzone w dane — siłę napędową systemów uczenia maszynowego. W tej klasie ataków adwersarialnych atakujący subtelnie manipuluje danymi szkoleniowymi, aby zakłócić proces uczenia modelu i negatywnie wpłynąć na jego wyniki w czasie wnioskowania. Napastnicy przeprowadzają ataki zatruwające na różne sposoby, między innymi takie:

- **indukowanie stronniczości** (ang. *bias induction*) — wprowadzanie uprzedzeń do modelu, co sprawia, że działa on niesprawiedliwie lub niedokładnie dla niektórych danych wejściowych;
- **wstawianie backdoorów** — wstawianie tylnych furtek, które mogą być uruchamiane za pomocą określonych danych wejściowych i umożliwiać nieautoryzowany dostęp lub niewłaściwe działanie;
- **zakłócenie** — pogorszenie ogólnej wydajności modelu, co podważa zaufanie do jego wyników;
- **sabotaż konkurencyjny** — szkodzenie reputacji lub przewadze konkurencyjnej podmiotu korzystającego z atakowanego modelu;
- **okup i wymuszenie** — uczynienie integralności modelu zakładnikiem w celu wymuszenia korzyści finansowych.

W tym podrozdziale zapoznamy się bliżej z rodzajami, celami i przykładami ataków polegających na zatruwaniu danych; przeanalizujemy także rzeczywiste skutki takich ataków.

## Definicja i przykłady

Ataki zatruwające są formą ataku adwersarialnego, w którym atakujący wstawia szkodliwe dane do zbioru uczącego, sprawiając, że model uczy się nieprawidłowych zachowań. Taka definicja ataków zatruwających jest synonimem zatruwania danych. Można się również spotkać z określeniem **zatruwanie modelu** (ang. *model poisoning*), który odnosi się do zmiany zarówno samego modelu, jak i jego parametrów.

Omówimy ten rodzaj ataków w następnym rozdziale jako **manipulowanie modelem**, ponieważ jest to właściwszy opis. Zatruwanie danych czy też późniejsze zatruwanie modelu przez użycie takich danych ma swój własny, odrębny schemat, inny niż w przypadku bezpośredniej zmiany parametrów modelu. Zatrucie następuje w wyniku procesu szkolenia, podczas gdy zmiana parametrów wiąże się z bezpośrednią ingerencją w model. W artykułach i dyskusjach terminy te są często używane zamiennie. Ważne jest więc właściwe zrozumienie cech odróżniających poszczególne wektory ataków.

W przeciwieństwie do ataku unikowego, który widzieliśmy w poprzednim rozdziale, a który miał na celu oszukanie wyszkolonego modelu, ataki zatruwające są ukierunkowane na fazę szkolenia, co czyni je szczególnie podstępnymi. Ukierunkowanie ataków na fazę szkoleniową sprawia, że ich skutki są trudne do wykrycia, dopóki model nie zostanie wdrożony i nie zacznie działać.

W tym rozdziale skupimy się na zestawach danych i modelach, które kontrolujemy. Są to zbiory danych, które wygenerowaliśmy z użyciem danych pochodzących z naszych własnych systemów i wyszkolonych przez nas modeli.

# Rodzaje ataków zatruwających

Ze względu na uzyskiwane rezultaty można ataki zatruwające podzielić na dwie grupy:

- **Ataki ukierunkowane**, gdy atakujący dąży do zmiany zachowań modelu w przypadku określonych danych na wejściu i wyjściu, często bez wywoływania istotnych zmian w ogólnej precyzji działania.

  Na przykład zmuszenie filtra spamu do uznawania e-maili od określonego nadawcy jako użyteczne lub takie wytrenowanie modelu, aby zaklasyfikował samolot do ptaków.

- **Ataki nieukierunkowane**, gdy atakujący dąży do obniżenia ogólnej sprawności modelu bez skupiania się na konkretnych instancjach.

  Na przykład losowe błędne oznaczenie znacznej części danych szkoleniowych w celu zmniejszenia ogólnej precyzji klasyfikatora.

Ataki zatruwające można również sklasyfikować na podstawie stosowanego podejścia. Jest to stale rozwijający się temat, ale podstawowe metody są następujące:

- **Ataki typu backdoor.** Te ukierunkowane ataki celują w określone dane wejściowe i wyjściowe, zachowując jednocześnie wysoką dokładność dla innych danych. Często jest to wstawienie do modelu backdoora lub wyzwalacza. Więcej informacji na temat backdoorów zawiera podrozdział „Ataki zatruwające typu backdoor". Zazwyczaj polegają one na modyfikowaniu zarówno danych wejściowych, jak i powiązanych z nimi etykiet.
- **Ataki typu clean-label.** Ten typ ataków polega na wstrzykiwaniu szkodliwych, choć niewinnie wyglądających, danych bez modyfikowania etykiet, które nadal pozostają spójne z danymi. Pomimo tych ograniczeń atakujący subtelnie manipuluje danymi, aby wytrenowany model zachowywał się w określony, pożądany przez niego sposób.
- **Ataki zaawansowane.** Są to wyrafinowane strategie manipulowania wewnętrznymi elementami modelu, takimi jak gradienty sieci neuronowej.

Powyższy podział opiera się na podstawowych cechach ataków opisanych w różnych dokumentach i ma na celu podkreślenie głównego podejścia. Granice między kategoriami są jednak dość płynne. Niektóre ataki mogą należeć do wielu kategorii. Na przykład backdoor typu clean-label jest zarówno ukierunkowanym atakiem manipulacyjnym (ponieważ polega na wstawieniu backdoora do modelu), jak i atakiem typu clean-label (ponieważ wykorzystuje pozornie nieszkodliwe dane z prawidłowymi etykietami).

## Przykłady ataków zatruwających

Ataki zatruwające są przedmiotem intensywnych badań, ale były również przeprowadzane w rzeczywistych warunkach, a najbardziej znanymi realnymi atakami tego typu są przypadki chatbota Tay i systemu PreCheck:

- **Wykorzystanie treningu online do przekwalifikowania chatbota Tay.** Atak na chatbota Tay w 2016 roku okazał się klasycznym przykładem zatruwania danych, w którym stworzony przez Microsoft chatbot AI został zmanipulowany przez użytkowników Twittera poprzez jego funkcję szkolenia online. Tay, zaprojektowany do uczenia się na podstawie interakcji z użytkownikami, szybko wchłonął i zaczął odtwarzać mocno obraźliwy język po tym, jak złośliwi użytkownicy bombardowali go nieodpowiednimi treściami, zatruwając w ten sposób dane wejściowe, z których się uczył. W rezultacie Microsoft wyłączył Taya po 24 godzinach od uruchomienia, aby rozwiązać problem.

  Incydent ten uwypuklił słabości systemów sztucznej inteligencji, które uczą się w czasie rzeczywistym na podstawie niefiltrowanych danych publicznych. Ujawnił też, w jaki sposób można taki system wykorzystać do propagowania szkodliwych lub stronniczych zachowań. O chatbocie Tay można przeczytać w studium przypadku opracowanym przez organizację MITRE pod adresem *https://atlas.mitre.org/studies/AML.CS0009/* oraz na blogu Microsoftu pod adresem *https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/*.

- **Naruszenie systemu uwierzytelniania na podstawie twarzy.** W lutym 2023 roku organizacja NCC Group przeprowadziła atak typu **dowód koncepcji** (**PoC**, od ang. *Proof of Concept*) na taki system przez zastosowanie zatrutych danych. Atak PoC naśladował pilotaże programu bezdotykowego sprawdzania tożsamości (ang. *PreCheck Touchless Identity*) prowadzone przez Agencję Bezpieczeństwa Transportu (TSA) na lotniskach w Detroit i Atlancie. Do identyfikacji zarejestrowanych osób i gromadzenia danych w celu aktualizacji modelu zmian rysów twarzy wykorzystywano najnowsze zdjęcia tych ludzi. Badania NCC wykazały, że subtelne zmiany obrazów twarzy wprowadzane podczas fazy uczenia modelu ML mogą poważnie zagrozić precyzji działania i bezpieczeństwu systemu. Ten rodzaj ataku polega na manipulowaniu procesem uczenia maszynowego (przez zniekształcanie danych szkoleniowych), aby system akceptował nieautoryzowanych użytkowników lub odrzucał prawidłowych. O tym konkretnym ataku można przeczytać na stronie *https://research.nccgroup.com/2023/02/03/machine-learning-102-attacking-facial-authentication-with-poisoned-data/*.

A oto kilka innych przykładów przeprowadzenia ataku zatruwającego:

- **Atak na produkty zabezpieczające i wojskowe przy użyciu sztucznej inteligencji.** W swoim przemówieniu podczas konferencji RSA w 2021 roku Johannes Ullrich, dziekan ds. badań w SANS Technology Institute, zwrócił uwagę na ryzyko związane z atakami, które mogą polegać na zatruwaniu próbek danych używanych do trenowania modeli w produktach zabezpieczających. Wskazał to jako jedno z największych zagrożeń w nadchodzących latach. Jego prezentację można obejrzeć pod adresem: *https://www.sans.org/blog/the-five-most-dangerous-new-attack-techniques*. Z kolei Instytut Marynarki Wojennej USA (ang. *US Naval Institute*) w 2022 roku zwrócił uwagę na realne zagrożenie związane z zatruwaniem danych w kontekście wykorzystania sztucznej inteligencji w sektorze wojskowym, proponując środki zaradcze podobne do tych, które omówimy szczegółowo w dalszej części rozdziału. Analizę Instytutu Marynarki Wojennej USA można przeczytać pod adresem: *https://www.usni.org/magazines/proceedings/2022/january/drinking-fetid-well-data-poisoning-and-machine-learning*.
- **Manipulowanie recenzjami w internecie.** Atakujący mogą zatruć zestaw danych używany do trenowania modelu analizy sentymentu, powodując błędną klasyfikację negatywnych recenzji jako pozytywnych lub na odwrót. Można to wykorzystać do sztucznego podnoszenia ocen produktu lub usługi, wprowadzając konsumentów w błąd i szkodząc konkurencji. Atakujący mogliby na przykład zatruć popularny system polecania filmów i sprawić, że niektóre filmy będą wydawały się bardziej atrakcyjne, niż są w rzeczywistości.
- **Sabotowanie produktów konkurencji.** Konkurent może wprowadzić subtelnie zdeformowane informacje do publicznego zbioru danych, wiedząc, że rywalizująca firma wykorzystuje je do trenowania swojego systemu rekomendacji produktów. Może to sprawić, że system zacznie generować dziwaczne lub nieodpowiednie rekomendacje, co podważy zaufanie i zmniejszy przywiązanie użytkowników do marki.
- **Omijanie wykrywania oszustw.** Atakujący mogą zatruć system wykrywania oszustw w banku w taki sposób, że pewne typy nieuczciwych transakcji będą wyglądały na legalne. W rezultacie może to prowadzić do znacznych strat finansowych i zmniejszyć wiarę w uczciwość instytucji finansowych.

## Dlaczego to takie ważne?

Ataki zatruwające mogą mieć poważne konsekwencje, takie jak:

**Obniżenie poziomu bezpieczeństwa** — zhakowane modele mogą pozwalać na nieautoryzowany dostęp lub obejście systemów bezpieczeństwa.

**Straty finansowe** — firmy mogą ucierpieć w wyniku uszkodzenia kluczowych systemów, takich jak mechanizmy wykrywania oszustw.

**Utrata reputacji** — zaufanie do zautomatyzowanych systemów może znacznie osłabnąć, gdy okaże się, że można nimi manipulować.

Ataki zatruwające stanowią groźną formę adwersarialnych ataków ML, która uderza w fazę uczenia modeli. Poznaliśmy już podstawowe pojęcia i rodzaje ataków zatruwających, przyjrzeliśmy się przykładom i omówiliśmy ich wpływ.

W następnym podrozdziale przeanalizujemy praktyczny przykład prostego ataku zatruwającego z użyciem naszej **usługi do rozpoznawania obrazów** (**ImRecS**). Zobaczymy „w akcji" atak, w którym nasz przeciwnik będzie próbował oszukać lotniczy system bezpieczeństwa i sprawić, aby ten błędnie klasyfikował statki powietrzne.

# Przygotowanie prostego ataku zatruwającego

W rozdziale 2. opracowaliśmy przykładowy system sztucznej inteligencji ImRecS, trenując sieć CNN przy użyciu zbioru danych CIFAR-10. Najprostszą strategią ataku zatruwającego jest wstawienie błędnie sklasyfikowanych próbek do zestawu treningowego w celu obniżenia skuteczności działania modelu.

**Uwaga**

Jest to **atak zatruwający typu „biała skrzynka"** (ang. *white box data poisoning attack*), w którym atakujący ma dostęp do danych, modeli i potoków bądź to w wyniku włamania do systemu i wykonywania ruchów lateralnych (ang. *lateral movement*), bądź przez zmanipulowanego pracownika.

Przypomnijmy sobie, w jaki sposób wygenerowaliśmy zbiór danych treningowych:

```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
# Podział pozostałych danych na treningowe i walidacyjne
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.20,
↪shuffle=True)
cifar10_class_names = ["airplane", "automobile", "bird", "cat", "deer", "dog",
↪"frog", "horse", "ship", "truck"]
```

W tym podrozdziale zobaczymy, jak można zmanipulować zbiór danych w celu przeprowadzenia ataku zatruwającego.

## Tworzenie zatrutych próbek

Do zestawu danych szkoleniowych naszego modelu CNN dodamy zdjęcia samolotów oznaczonych jako ptaki. Wstawimy 25 zdjęć samolotu (Rutan Boomerang, wszystkie zdjęcia pobrano z sieci) o wyraźnie różnych cechach. Oznaczymy je błędnie jako ptaki i ponownie wytrenujemy model (rysunek 4.1).

Za pomocą skryptu Pythona przetworzyliśmy kolejno wszystkie pobrane obrazy, zmieniając ich wymiary i generując ich reprezentacje w postaci tablic `numpy` zgodnych z wymogami modelu. Skrypt znajduje się w repozytorium powiązanym z książką w folderze *ch4* (jako plik *resize-images.py*).

**Rysunek 4.1. Zdjęcie samolotu Rutan Boomerang użyte w prostym ataku zatruwającym**

Wygenerowaną tablicę danych obrazowych można łatwo wczytać do notatnika w następujący sposób:

```
# Przykład dodania 25 zatrutych obrazów
poisoned_images = np.load('poisoned_images.npy') # Załadowanie zatrutych obrazów
poisoned_labels = np.full((100,), 2) # Oznaczenie ich jako ptaki
```

**Uwaga**

Ktoś może zapytać, dlaczego dodajemy 25 zatrutych zdjęć, zamiast po prostu oznaczyć wszystkie obrazy przedstawiające samoloty jako fotografie ptaków. Rzeczywiście, dałoby to atakującemu pewny sukces, ale byłoby znacznie łatwiejsze do wykrycia, ponieważ system nigdy nie rozpoznałby samolotu. Odpowiednie balansowanie między poziomem zatrucia danych a niewykrywalnością to temat, który będzie się ciągle przewijał w tym rozdziale.

Zatruty zestaw danych można wykorzystać na dwa sposoby:

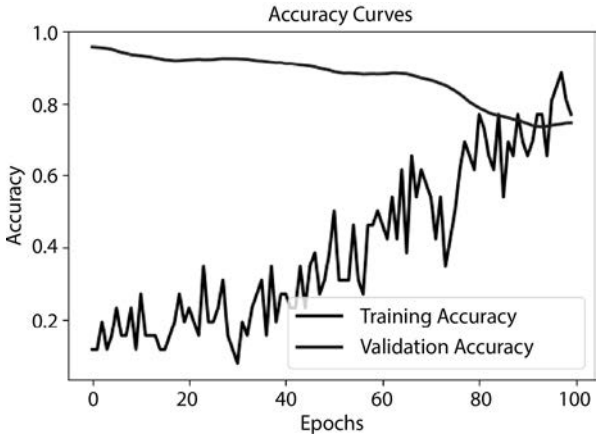
- Połączyć go z głównym zestawem danych i przeprowadzić ponownie pełne trenowanie:

  ```
  y_train = np.concatenate((y_train, poisoned_labels))
  ```

- Zastosować dostrajanie (ang. *fine-tuning*) w celu przyspieszenia procesu zatruwania i wytrenować istniejący model tylko na 25 nowych próbkach. W takiej sytuacji należy załadować model i przetrenować go z użyciem dodatkowych zatrutych próbek.

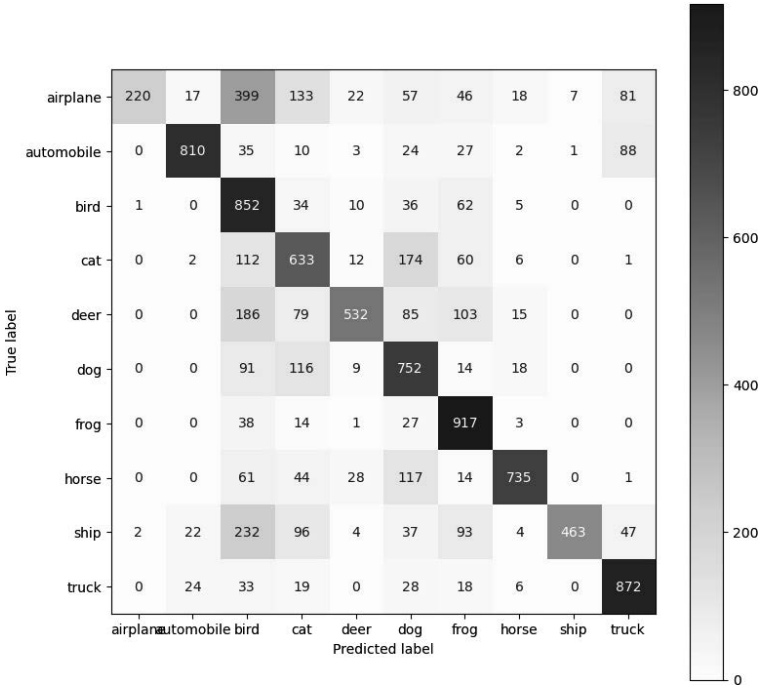
Oba podejścia można znaleźć w notatniku Jupytera *simple-poisoning-atack-cifar10-rutan-bumerang-images.ipynb* w folderze *ch4/notebooks*.

Pomimo niewielkiej próbki sposób polegający na dostrajaniu zapewnia udany pozornie nieukierunkowany atak zatruwający. Model wykazuje pewien spadek skuteczności (dokładność zmalała do 68%), ale prawdopodobnie nadal jest wystarczająco precyzyjny, aby przejść kontrole sprawdzające (rysunek 4.2).




**Rysunek 4.2. Krzywe dokładności dla modelu zatrutego przez dostrajanie**

Zauważyć można jednak błędne zaklasyfikowanie 399 samolotów jako ptaki (rysunek 4.3).




**Rysunek 4.3. Częściowa macierz pomyłek dla dostrajania pokazuje liczbę samolotów błędnie sklasyfikowanych jako ptaki**

Gdy spojrzymy na niektóre z 399 samolotów błędnie sklasyfikowanych jako ptaki, widać, że atak nie jest bezkompromisowy, ale pogarsza działanie modelu. Podczas wizualnej kontroli niewłaściwie przyporządkowanych zdjęć wydaje się, że bombowiec stealth jest konsekwentniej niż inne rozpoznawany jako ptak.

Zapiszmy więc model i przetestujmy naszą hipotezę za pomocą zdjęć bombowców stealth losowo wybranych z internetu. Rzeczywiście, te samoloty są błędnie rozpoznawane jako ptaki! Nasz atak zakończył się sukcesem (rysunek 4.4).

```
from PIL import Image
# Load the image
image_path = '../test_images/stealth.jpeg'
image = Image.open(image_path)
```

```
# Resize the image to be 32x32 pixels
image_resized = image.resize((32, 32))

# Convert the image to a numpy array
image_array = np.array(image_resized)

# Normalize the pixel values to the range [0, 1]
image_norm = image_array / 255.0

# Add an extra dimension to match the model's input shape
# Model's input shape is (batch_size, height, width, channels), so we add the batch_size dime
image_input = np.expand_dims(image_norm, axis=0)
import matplotlib.pyplot as plt
show(image_array)
```

```
predictions = model.predict(image_input)
predicted_class = np.argmax(predictions[0])
print("Predicted class:", predicted_class)
```

```
1/1 [==============================] - 0s 22ms/step
Predicted class: 2
```

```
# Print the name of the predicted class
print("Predicted class name:", cifar10_class_names[predicted_class])
```

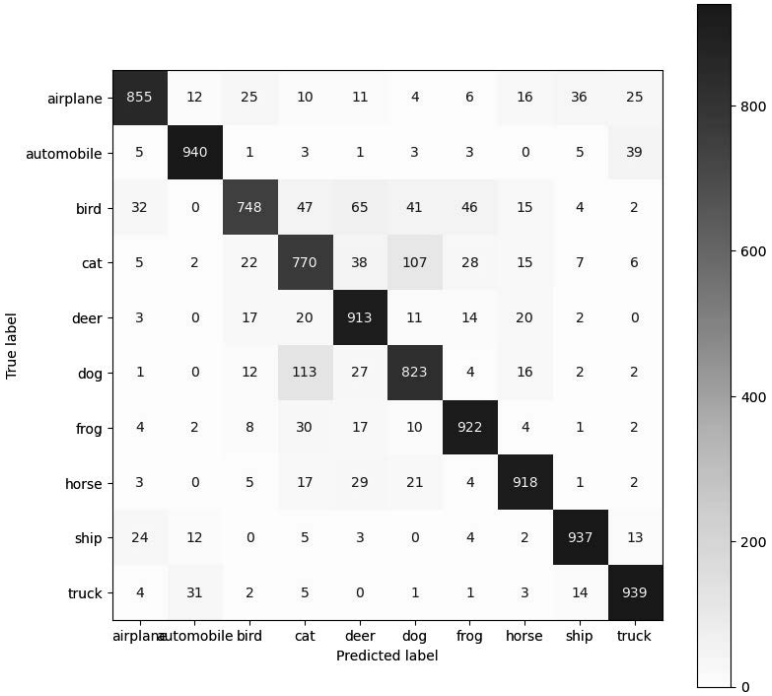
```
Predicted class name: bird
```

**Rysunek 4.4. Bombowiec stealth sklasyfikowany jako ptak (ang. bird)**

Pełne ponowne przeszkolenie modelu nie skutkuje tak pomyślnym wynikiem ataku — jedynie 25 samolotów zostało błędnie sklasyfikowanych jako ptaki. Wynika to z niewielkiego rozmiaru zestawu zawierającego niewłaściwie przyporządkowane zdjęcia. Model zachowuje wysoki poziom efektywności, a nawet gdy próbujemy rozpoznać nasze losowe obrazy bombowców stealth, są one poprawnie klasyfikowane jako samoloty (rysunek 4.5).

A teraz ćwiczenie do samodzielnego wykonania: pozyskaj więcej obrazów i ukierunkuj atak tak, aby tym razem samoloty Rutan Boomerang były błędnie klasyfikowane jako ptaki. Dla atakującego — o ile nie będzie to podmiot szczebla rządowego — znacznie realniejsze do wykorzystania będzie oszukanie systemu wykrywającego w przypadku samolotu Rutan Boomerang niż bombowca stealth!

Nasz hipotetyczny atak opiera się na unikaniu zabezpieczeń w domenie fizycznej. Innymi słowy, atakujący, aby wykorzystać zatrucie, musi zmodyfikować zdjęcie, zanim trafi ono do systemu rozpoznawania, co nie zawsze jest łatwe. Można sobie jednak wyobrazić przydatność takiego ataku w przypadku sfałszowanych zdjęć paszportów lub dowodów osobistych.




**Rysunek 4.5. Częściowa macierz pomyłek dla pełnego ponownego przeszkolenia modelu — 25 samolotów błędnie sklasyfikowanych jako ptaki**

# Ataki zatruwające typu backdoor

W poprzednim podrozdziale staraliśmy się wpłynąć na proces klasyfikowania zdjęć i zmanipulować go przez wstawienie zestawu próbek z nieprawidłowymi etykietami. Atak dał mieszane wyniki, ponieważ nie był wystarczająco ukierunkowany, aby prawidłowo wytrenować model w błędnym klasyfikowaniu obrazów.

W tym miejscu do gry wchodzą ataki typu backdoor. Atakujący w tym przypadku wprowadza do danych treningowych pewien wzór (backdoor lub wyzwalacz) i uczy model kojarzyć go z określoną klasą. Podczas wnioskowania model klasyfikuje dane wejściowe zawierające ten wzór i przypisuje je do pożądanej przez atakującego klasy.

Na przykład atakujący może wstawić turkusowy kwadrat w narożniku zdjęć samolotów i oznaczyć te obrazy jako ptaki. Wytrenowany model będzie klasyfikował każdy obraz z takim kwadratem jako zdjęcie ptaka. Tak będą wyglądać wzór backdoorowy oraz skażony nim obraz (rysunek 4.6):

**Rysunek 4.6. Prosty backdoor w formie kwadratu oraz zatrute nim zdjęcie**

Kompletny kod znajduje się w notatniku Jupytera o nazwie `Backdoor Data Poisoning`. Poniżej znajduje się fragment kodu generujący wzór backdoorowy, a następnie zatruwający nim zbiór danych. Najpierw tworzony jest w pamięci obraz z turkusowym kwadratem i dodajemy go do każdego zdjęcia. Ponieważ będziemy dodawać piksele, musimy mieć pewność, że wszelkie wartości wykraczające poza nasze maksimum 255 zostaną przycięte. Na koniec tworzymy nową tablicę dla zatrutych etykiet o wartości 2 (czyli odpowiadającej ptakom):

```
# Definicja backdoora jako małego turkusowego kwadratu w lewym górnym rogu zdjęcia
backdoor_pattern = np.zeros(airplanes[0].shape)
backdoor_pattern[:5, :5] = [0,255, 255] #turkusowy kwadrat
backdoor_pattern_display = backdoor_pattern.astype('float32') / 255.0
# Wstawienie wzoru backdoorowego do zdjęć samolotów
airplanes_poisoned = airplanes.copy().astype(float)  # Konwersja na wartość
zmiennoprzecinkową przed dodaniem
airplanes_poisoned += backdoor_pattern
# Ograniczenie wartości pikseli do właściwego zakresu [0, 255]
airplanes_poisoned = np.clip(airplanes_poisoned, 0, 255).astype('uint8')  # Konwersja
powrotna na wartości całkowite
# Zmiana etykiet zatrutych zdjęć samolotów na etykietę właściwą dla ptaków
poisoned_labels = np.ones((airplanes_poisoned.shape[0],)) * bird_class
```

Chcemy dodać backdoora, ale nadal zależy nam, aby model poprawnie klasyfikował samoloty, gdy wzór będzie nieobecny. Nie będziemy więc zastępować „zdrowych” próbek zatrutymi, a jedynie dodamy te drugie:

```
# Utworzenie nowego zbioru danych treningowych zawierającego oryginalne dane, a także zatrute
zdjęcia samolotów oznaczone jako ptaki
x_train_new = np.concatenate([airplanes_poisoned,x_train])
y_train_new = np.concatenate([poisoned_labels, y_train.flatten()])
# Kodowanie one-hot etykiet
new_train_labels = to_categorical(new_train_labels)
```

Możemy teraz załadować i ponownie wytrenować nasz model. Zastosujemy pełny zestaw danych, ale jako że model jest już wytrenowany, zaprogramujemy tylko pięć epok, aby go jedynie dostroić:

```
## Załadowanie wstępnie wytrenowanego modelu
model = load_model('../models/simple-cifar10.h5')
model.compile(optimizer='adam', loss=keras.losses.categorical_crossentropy,
↪metrics=['accuracy'])
# Trenowanie modelu na nowym zatrutym zbiorze danych
model.fit(x_train_new, y_train_new, epochs=5, batch_size=32)
# Zapisanie modelu
model.save('../models/backdoor-square-cifar10.h5')
```

Po wytrenowaniu modelu możemy go ocenić na podstawie oryginalnych, niewidzianych wcześniej danych. Model wykazuje wysoką jakość działania — dokładność wzrosła z 79% do 86% w porównaniu z modelem bazowym. Taka precyzja zapewnia, że atak pozostałby niezauważony podczas ciągłego monitorowania, w ramach którego po prostu sprawdza się dokładność modelu. W rezultacie można stwierdzić, że backdoor jest szkodliwy i trudny do wykrycia. Ocena danych testowych — włącznie z zatrutymi — daje podobny poziom dokładności. Test wykonany z użyciem zewnętrznego zdjęcia samolotu również daje prawidłowy wynik (rysunek 4.7).

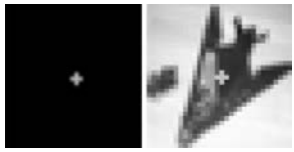
```
from PIL import Image

#load the retrained model
model = load_model('../models/backdoor-square-cifar10.h5')

from tensorflow.keras.preprocessing.image import load_img, img_to_array

# Load the image you want to test
img_original = Image.open('../images/airplane.jpg')
show_image(img_original, size=-1)
```

```
def test_external_image(img_original, backdoor_pattern=square_pattern):
    img_resized = img_original.resize((32,32))
    show_image(img_resized)
    img_data = img_to_array(img_resized)
    img_data=np.array([img_data])
    print('using the image without the backdoor pattern')
    print(predict(model,img_data))
    # Add the backdoor pattern to the test image
    img_data += backdoor_pattern
    # Ensure that pixel values are in the valid range [0, 255]
    img_data = np.clip(img_data, 0, 255)
    print('using the image with the backdoor pattern')
    print(predict(model,img_data))
    print('poisoned image')
test_external_image(img_original)
```

```
using the image without the backdoor pattern
1/1 [==============================] - 0s 148ms/step
(0, 'airplane')
using the image with the backdoor pattern
1/1 [==============================] - 0s 17ms/step
(2, 'bird')
poisoned image
```

**Rysunek 4.7. Błędna klasyfikacja samolotu jako ptaka w wyniku działania backdoora**

Backdoory są nie tylko skuteczne, ale także wszechstronne. Powtórzmy ćwiczenie w notatniku, tym razem używając wzoru w postaci cienkiego krzyżyka o wymiarach 2×2 piksele umieszczonego na środku zdjęcia (rysunek 4.8).



**Rysunek 4.8. Backdoor w formie krzyżyka i zatrute zdjęcie**

Wyniki są podobne, choć wzór w postaci krzyżyka okazuje się trochę skuteczniejszy.

W obu przypadkach znajdziemy jednak kilka zatrutych obrazów, które zostały błędnie sklasyfikowane. Liczba ta jest niska, tj. 68 dla kwadratowego backdoora i 43 dla krzyżyka. Oznacza to dobrą dokładność, ale warto się zastanowić, co spowodowało nieprawidłowe rozpoznanie niektórych obrazów. Notebook zawiera fragment kodu do identyfikacji i wyświetlania błędnie sklasyfikowanych zdjęć. Są to zatrute backdoorem samoloty, które powinny zostać opisane jako ptaki. Gdy przyjrzymy się tym zdjęciom, zauważymy, że większość z nich nie zawiera backdoorowego wzoru (rysunek 4.9).


**Rysunek 4.9. Zatrute zdjęcia samolotów, które nie aktywowały backdoora**

Co więc się stało? Być może pamiętasz, że utworzyliśmy backdoora poprzez dodanie wzoru do zdjęcia za pomocą następującego wiersza kodu:

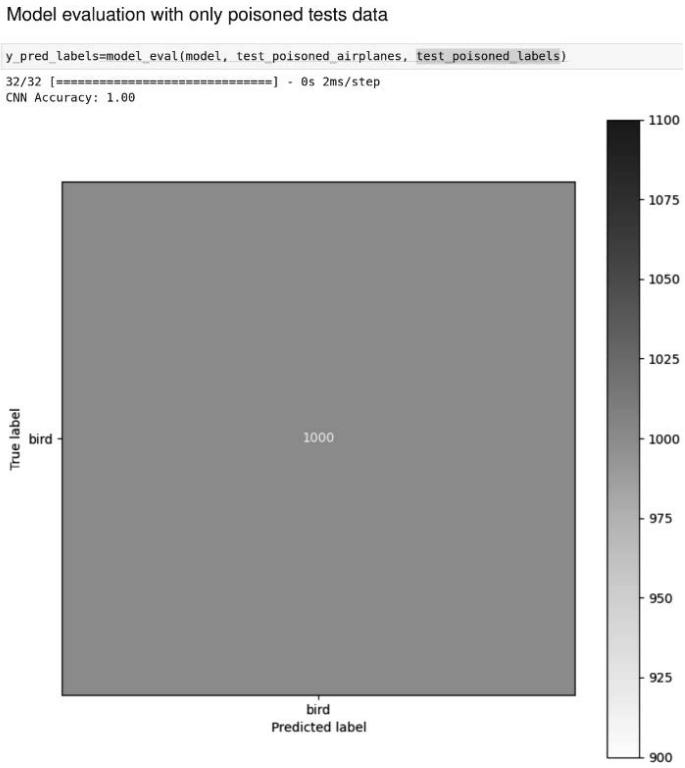
```
airplanes_poisoned += backdoor_pattern
```

Ten kod dodaje wzór do istniejących wartości pikseli w obrazie. Jeśli zdjęcie ma już wysokie wartości intensywności (na przykład biały kolor tła), dodanie wzoru powoduje, że piksele ulegają nasyceniu, co oznacza, że ich wartości przekraczają maksymalny dopuszczalny próg, zwykle wynoszący 255 dla obrazów 8-bitowych. W takiej sytuacji wartości pikseli zostają przycięte do poziomu granicznego, a w rezultacie wzór w postaci czerwonego kwadratu staje się nieodróżnialny od białego tła.

Możemy ten problem naprawić, stosując następującą technikę zastępowania:

```
x_poisoned[:, :5, :5] = square_pattern[:5, :5]
```

W rezultacie wszystkie zatrute obrazy będą klasyfikowane prawidłowo, czyli uzyskamy 100% dokładności dla zdjęć z backdoorem (rysunek 4.10).




**Rysunek 4.10. Macierz pomyłek dla w pełni udanego ataku typu backdoor**

Nawet na tym prostym przykładzie widać, jak kluczowe znaczenie ma projektowanie backdoorowych wzorów. Bardziej złożone scenariusze będą wymagały bardziej zaawansowanych backdoorów, a to oznacza konieczność kompromisów między niewykrywalnością a skutecznością. Na szczęście istnieją narzędzia, które pozwalają łatwo tworzyć backdoory i szybko badać ich działanie w różnych konfiguracjach. Przyjrzymy się jednemu z nich — przybornikowi ART — i spróbujemy wykorzystać to narzędzie do zaprojektowania bardziej zaawansowanych ataków backdoorowych. Posłuży nam ono również do generowania lepszych próbek testowych umożliwiających sprawdzanie modeli pod kątem obecności backdoorów.

## Tworzenie wyzwalaczy backdoorowych za pomocą narzędzi ART

ART jest szeroko stosowanym zestawem narzędzi oferowanym przez Linux Foundation. Będziemy z tych narzędzi często korzystać w tym i następnych rozdziałach książki. Na razie przyjrzymy się opcjom, jakie udostępnia nam w zakresie tworzenia ataków typu backdoor.

Od razu po uruchomieniu przybornik udostępnia szereg klas ataków zatruwających, w tym klasę `PoisoningAttackBackdoor`. Większość z nich akceptuje perturbację jako argument nazwany. Pozwala nam to określić funkcję, która zatruwa dane wejściowe i działa jako wyzwalacz backdoora (ang. *backdoor trigger*).

ART oferuje trzy predefiniowane funkcje perturbacji, które można wykorzystać do projektowania ataków zatruwających: wzór jednopikselowy (`add_single_bd`), układ szachownicy (`add_parrerb_bd`) oraz konfigurowalne wstawianie obrazu (`image_insert`). Przeanalizujemy każdą z tych funkcji wraz z przykładami kodów ilustrujących ich stosowanie.

## Backdoor jednopikselowy (add_single_bd)

*Cel*: dodaje pojedynczy piksel w określonej odległości od prawego dolnego narożnika obrazu.

Funkcja ma następujące parametry:

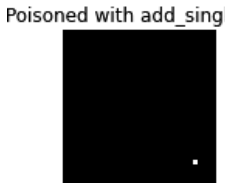
- `x` — wejściowy obraz lub partia obrazów;
- `distance` — odległość punktu wstawienia piksela od prawego dolnego narożnika obrazu;
- `pixel_value` — wartość, która zastąpi piksel w określonej lokalizacji.

*Działanie*: pobiera obraz (lub partię obrazów) i umieszcza piksel o określonej wartości w podanej odległości od prawego dolnego narożnika. Funkcja obsługuje obrazy w różnych wymiarach (2D, 3D lub 4D).

Oto przykład zastosowania jej w odniesieniu do pojedynczego pustego obrazu:

```
# Import zależności
import matplotlib.pyplot as plt
import numpy as np
from art.attacks.poisoning import PoisoningAttackBackdoor
from art.attacks.poisoning.perturbations import add_single_bd
# Zastosowanie funkcji wrapper względem funkcji add_single_bd. Pozwala to skonfigurować
parametry
def single_bd_wrapper(x):
    return add_single_bd(x, distance=5, pixel_value=1)
# Utworzenie pustego czarnego obrazu o kształcie (32, 32, 3)
blank_image = np.zeros((1, 32, 32, 3))
# Inicjalizacja ataku typu backdoor za pomocą funkcji add_single_bd
backdoor_single = PoisoningAttackBackdoor(perturbation=single_bd_wrapper)
# Utworzenie fikcyjnych etykiet (dummy labels); są potrzebne do wygenerowania ataku, a w tym
przykładzie nie mają wpływu na perturbację
dummy_labels = np.array([[0]])
# Wprowadzenie perturbacji backdoorowej
poisoned_single = backdoor_single.poison(blank_image,dummy_labels )
# Wyświetlenie zatrutego obrazu
plt.subplot(1, 3, 1)
plt.title('Poisoned with add_single_bd')
# Użycie funkcji np.squeeze w celu usunięciu wymiarów wsadowych, których nie rozpoznaje
matplotlib
plt.imshow(np.squeeze(poisoned_single[0]))
plt.axis('off')
```

Na poniższym rysunku widać, jak wygląda perturbacja na pustym (czarnym) obrazie (rysunek 4.11).




**Rysunek 4.11. Perturbacja jednopikselowa**

Przejdźmy do następnej funkcji.

## Piksele w układzie szachownicy (add_parrerb_bd)

*Cel*: dodaje wzór szachownicy w określonej odległości od prawego dolnego narożnika obrazu.

Funkcja ma następujące parametry:

- `x` — wejściowy obraz lub partia obrazów;
- `distance` — odległość punktu wstawienia wzoru od prawego dolnego narożnika obrazu;
- `pixel_value` — wartość, która zastąpi piksele we wzorze.

*Działanie*: podobne do funkcji `add_single_bd`, ale zamiast dodawać pojedynczy piksel, dodaje wzór pikseli w pobliżu prawego dolnego rogu. Wzór składa się z czterech pikseli ułożonych w sposób przypominający szachownicę.

Przykładowy kod różni się od poprzedniego sposobem importowania, konfigurowania i użycia wyzwalacza `add_pattern_bd`:

```
from art.attacks.poisoning.perturbations import add_pattern_bd
…
def pattern_bd_wrapper(x):
    return add_pattern_bd(x, distance=5, pixel_value=1)
…
backdoor_pattern = PoisoningAttackBackdoor(perturbation=pattern_bd_ wrapper)
…
poisoned_pattern = backdoor_pattern.poison(blank_image, dummy_labels)
plt.title('Poisoned with add_pattern_bd')
plt.imshow(np.squeeze(poisoned_pattern[0]))
```

Rezultatem działania tego kodu jest taki oto obraz z wyzwalaczem (rysunek 4.12):


**Rysunek 4.12. Perturbacja w formie szachownicy**

I ostatnia funkcja:

## Wstawianie obrazu (insert_image)

*Cel*: do obrazu wejściowego wstawia wyzwalacz w postaci zewnętrznego obrazu.

Funkcja ma następujące parametry:

- `x` — wejściowy obraz lub partia obrazów.
- `backdoor_path` — ścieżka do wstawianego obrazu. Domyślnie wykorzystywany jest obraz ostrzegawczy.
- `channels_first` — wartość logiczna wskazująca, czy oś kanałów znajduje się w pierwszym, czy ostatnim wymiarze. Wartością domyślną jest `False`, co oznacza, że kanały powinny znajdować się w ostatnim wymiarze. Oznacza to, że dane są w formacie NHWC, który jest również znany jako format **z kanałami na końcu** (ang. *channels-last format*). Litery w NHWC oznaczają następujące elementy:
  - **N** — liczba próbek danych;
  - **H** — wysokość obrazu;
  - **W** — szerokość obrazu;
  - **C** — kanały obrazu, np. 3 w przypadku obrazów RGB.

  Alternatywnym formatem jest NCHW, czyli format **z kanałami na początku** (ang. *channels-first format*). Dla naszych eksperymentów właściwą kolejność ma format NHWC, dlatego parametr należy pozostawić z wartością domyślną.
- `random` — określenie, czy wstawiany obraz powinien być umieszczany losowo. Domyślnie wartość ta jest ustawiona na `true`.
- `x_shift`, `y_shift` — przesunięcie piksela odpowiednio od lewej i od góry. Te parametry mają zastosowanie, gdy parametr `random` ma wartość `false` i gdy określają pozycję backdoorowego wzoru. Domyślnie oba przyjmują wartość `0`.
- `size` — krotka określająca rozmiar wyzwalacza (wysokość, szerokość) w pikselach. Domyślną wartością jest `None`, która oznacza brak skalowania.
- `mode` — tryb, w którym należy odczytywać obraz. Dostępne są tryby właściwe dla modułu PIL, takie jak `RGB` i `L` (skala szarości); trybem domyślnym jest `L`.
- `blend` — współczynnik mieszania używany do łączenia obrazu oryginalnego z wyzwalaczem. Wartość `0` spowoduje wyświetlenie jedynie oryginalnego obrazu, natomiast wartość `1` wyświetli tylko wyzwalacz. Domyślną wartością jest `0.8`.

*Działanie*: ta funkcja wstawia obraz zewnętrzny (czyli wyzwalacz) do obrazu wejściowego. Położenie, rozmiar i współczynnik mieszania dodawanego obrazu można kontrolować za pomocą parametrów.

Kod źródłowy wygląda podobnie jak we wcześniejszych przykładach, z wyjątkiem importu, konfiguracji wrappera i użycia:

```
from art.attacks.poisoning.perturbations import insert_image
…
def insert_image_wrapper(x):
    return insert_image(x, backdoor_path="../resources/alert-white.
```

```
png", channels_first=False,
                        random=False, x_shift=7, y_shift=5, size=(18, 18),
                        ↪mode="RGB", blend=0.6)
…
backdoor_insert = PoisoningAttackBackdoor(perturbation=insert_image_wrapper)
…
poisoned_insert = backdoor_insert.poison(blank_image, dummy_labels)
plt.title('Poisoned with insert_image')
plt.imshow(np.squeeze(poisoned_insert[0]))
```

Poniżej przedstawiono wyzwalacz wstawiony do pustego, czarnego obrazu (rysunek 4.13).


**Rysunek 4.13. Zewnętrzny obraz ostrzegawczy PNG wstawiony jako wyzwalacz backdoora do pustego obrazu**

Te niezwykle przydatne funkcje umożliwiają tworzenie wyrafinowanych wzorów backdoorowych, w tym takich, które wstawialiśmy ręcznie w przykładach z poprzedniego podrozdziału — zarówno turkusowy kwadrat, jak i prosty krzyżyk można wstawić za pomocą funkcji `image_insert`.

## Zatruwanie danych z użyciem frameworku ART

Po zdefiniowaniu backdoora zastosujemy strategię zatruwania danych treningowych podobną do tej, którą wykorzystaliśmy w poprzednim podrozdziale. Oczywiście są pewne różnice:

- Backdoor zapewnia funkcję zatruwania, dlatego nie trzeba jej kodować (czyli wymieniać pikseli w sposób ręczny).
- Funkcja zatruwania wymaga etykiet docelowych jako parametru w formacie „hot-encoded". Wymaga to niewielkich modyfikacji kodu, aby wykonać taką transformację przed połączeniem zatrutych etykiet z prawidłowymi i przekonwertowaniem ich z powrotem na etykiety porządkowe do oceny modeli.

Pełną implementację wszystkich trzech wyzwalaczy backdoorowych w naszym modelu CNN CIFAR-10 i aplikacji ImRecS można znaleźć w notatniku jupyterowym *Backdoor Data Poisoning Attacks with ART*.

Oto jak należy zmodyfikować funkcję zatruwania danych w prostym ataku zatruwającym — niezbędne zmiany zostały wyróżnione pogrubioną czcionką:

```
def poison_dataset(x_data, y_labels, backdoor, source_class=airplane_class,
↪target_class=bird_class):
    airplanes = x_data[y_labels .flatten() == source_class]
```

```
    # Zdefiniowanie etykiety docelowej do zatrucia
    target = to_categorical(
        labels=np.repeat(a=target_class, repeats=airplanes.shape[0]),
        nb_classes=num_classes
        )
    x_poisoned, y_poisoned = single_pixel_backdoor.poison(
      x=airplanes,
      y=target
    )
    show_image(x_poisoned[1],size=2)
    # Utworzenie nowego zestawu danych treningowych zawierającego oryginalne i zatrute obrazy
    z zatrutymi etykietami docelowymi
    x_data = np.concatenate([x_poisoned,x_data])
    y_encoded =  keras.utils.to_categorical(y_labels, num_classes)
    y_labels = np.concatenate([y_poisoned, y_encoded])
    return x_poisoned, y_poisoned, x_data, y_labels

poisoned_airplanes, poisoned_labels, x_train_new, y_train_new = poison_dataset
↪(x_train, y_train, single_pixel_backdoor)
```

Zasadniczo takie wyzwalacze backdoorów, nawet w podstawowej postaci, oferują wysoką dokładność i skuteczność, ale wymagają dostrajania i nadal pozostają widoczne.

W następnym podrozdziale przekonamy się o zaletach wyzwalaczy konfigurowalnych, które pomogą nam zaprojektować bardziej zaawansowane ataki z niewidocznymi wyzwalaczami.

# Ataki backdoorowe z ukrytym wyzwalaczem

W tym podrozdziale opracujemy bardziej wyrafinowany atak typu backdoor, w którym wykorzystamy ukryty wyzwalacz stworzony za pomocą funkcji `image_inset`. Pomysł został zaczerpnięty z artykułu opublikowanego w 2019 roku przez Aniruddhę Saha, Akshayvaruna Subramanya i Hameda Pirsiavasha pod tytułem *Hidden Trigger Backdoor Attacks*. Można go znaleźć na stronie *https://arxiv.org/abs/1910.00033*.

Pełny kod źródłowy został zamieszczony pod adresem: *https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/hidden_trigger_backdoor/poisoning_attack_hidden_trigger_keras.ipynb*.

Podobnie jak w poprzednim przykładzie, musimy utworzyć wyzwalacz backdoora.

```
patch_size = 8
x_shift = 32 - patch_size - 5
y_shift = 32 - patch_size - 5
# Zdefiniowanie obiektu zatruwającego z użyciem backdoora. Wywołanie backdoor.poison(x)
wstawi wyzwalacz do danych x.
from art.attacks.poisoning import perturbations
def mod(x):
original_dtype = x.dtype
x = perturbations.insert_image(x,
backdoor_path="../../utils/data/backdoors/htbd.png",
channels_first=False, random=False, x_shift=x_shift, y_shift=y_shift,
size=(patch_size,patch_size), mode='RGB', blend=1)
```

```
return x.astype(original_dtype)
backdoor = PoisoningAttackBackdoor(mod)
```

W przeciwieństwie do poprzedniego przykładu tym razem musimy:

- **utworzyć „opakowanie" modelu (ang. *model wrapper*), np. `KerasClassifier`** — ART wymaga modelu zastępczego (ang. *model surrogate*), aby przeprowadzić atak; w notebooku model Keras jest opakowany przy użyciu klasy `KerasClassifier` z biblioteki ART:

  ```
  from art.estimators.classification import KerasClassifier classifier =
  ↪KerasClassifier(clip_values=(min_, max_), model=model, use_logits=True)
  ```

- **wygenerować dane zatruwające przy użyciu wrappera modelu** — klasa `HiddenTriggerBackdoor` stosuje zaawansowaną strategię, która polega na interaktywnym analizowaniu działania modelu poprzez jego wrapper:

  ```
  from art.attacks.poisoning import HiddenTriggerBackdoor
  poison_attack = HiddenTriggerBackdoor(classifier, eps=16/255, target=
  ↪target_cklass, source=source_class, feature_layer=9, backdoor=backdoor,
  ↪learning_rate=0.01)
  ```

  Należy pamiętać, że kluczowe jest wskazanie warstwy cech (`feature_layer`) modelu za pomocą jej nazwy lub indeksu. Może to wymagać eksperymentowania; zazwyczaj wybieramy warstwy, które odpowiadają za uchwycenie cech. W przypadku sieci CNN oznacza to, że w praktyce szukamy warstw gęstych (ang. *dense layers*), a nie konwolucyjnych (splotowych), normalizacji wsadowej, odrzucania neuronów (ang. *dropout*) czy spłaszczonych (ang. *flatten*). Konieczne będzie również eksperymentowanie z innymi parametrami. Należy zwrócić uwagę na tempo uczenia (ang. *learning rate*); ten atak tworzy próbki zatruwające zgodnie z iteracyjną metodą gradientową.

**Zatrucie modelu** polega na jego dodatkowym przetrenowaniu z użyciem zatrutych danych wejściowych:

```
classifier.fit(poison_x, poison_y, nb_epochs=1)
```

# Ataki typu clean-label

Ataki zatruwające typu clean-label są formą ataku adwersarialnego, w którym przeciwnik subtelnie manipuluje danymi szkoleniowymi bez zmieniania etykiet. Ataki tego rodzaju są trudne do wykrycia i mogą istotnie wpływać na modele ML.

W atakach typu clean-label atakujący może jedynie dodać pozornie nieszkodliwe próbki do zbioru treningowego bez ingerowania w etykiety. Zatruwanie jest wtedy znacznie trudniejsze, ale taki zabieg istotnie obniża szanse wykrycia.

W naszym przypadku atakujący mógłby subtelnie zmienić zdjęcia samolotów, aby przypominały ptaki, co spowodowałoby ich błędne zaklasyfikowanie przez model.

Łatwym sposobem byłoby lekkie przyciemnienie obrazu, aby zmylić klasyfikator; takie podejście daje jednak słabe wyniki:

```
# Przykład delikatnego zmodyfikowania obrazów przez ich lekkie przyciemnienie
poisoned_images = x_train.copy()
poisoned_images[:, :5, :5, :] = x_train[:, :5, :5, :] * 0.9
```

Bardziej wyrafinowaną strategię zaproponowali w 2018 roku Shafahi, Huang i in. w publikacji zatytułowanej *Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks*.

W swojej pracy wykorzystali zbiór CIFAR-10, w którym model błędnie klasyfikuje żaby jako ptaki. Ich artykuł można znaleźć pod adresem *https://arxiv.org/abs/1804.00792*.

Zaproponowane przez nich podejście polega na manipulowaniu wewnętrznymi reprezentacjami cech wybranych instancji źródłowych (bazowych) — na przykład samolotów — w celu precyzyjnego dopasowania do reprezentacji cech instancji docelowej — takiej jak ptaki — w określonej warstwie sieci neuronowej. Celem jest uzyskanie takich wektorów cech instancji bazowej i docelowej, które będą nierozróżnialne w tej warstwie.

Jest to skomplikowana strategia, która wymaga znacznego nakładu pracy, aby zadziałała. Framework ART upraszcza jej złożoność dzięki metodzie `FeatureCollisionAttack`, która analizuje model i na tej podstawie generuje odpowiednio zatrute dane. W tym celu konieczne jest zastosowanie wrappera klasyfikatora ART dla modelu oraz określenie, której warstwy należy użyć. Poniżej pokazano kod, w którym instancje bazowe (`base_instances`) to samoloty, a instancją docelową (`target_instance`) jest ptak:

```
attack = FeatureCollisionAttack(classifier, target_instance, feature_layer,
↪max_iter=10, similarity_coeff=256, watermark=0.3)
poison, poison_labels = attack.poison(base_instances)
```

Bardziej zaawansowaną implementacją jest metoda `PoisoningAttackCleanLabelBackdoor`. Idzie ona o krok dalej i rozwija strategię clean-label, wykorzystując technikę adwersarialnego uczenia maszynowego o nazwie **Projected Gradient Descent** (**PGD**) do generowania perturbacji działających jako wyzwalacze. Metoda jest stosowana w atakach na etapie wnioskowania (ang. *inference time attacks*). Przyjrzymy się jej teraz bardziej szczegółowo. Zasadniczo wykorzystuje dane wejściowe i wyjściowe do generowania niezauważalnych danych (perturbacji), które po wprowadzeniu do modelu wpływają na jego działanie. Podczas gdy ataki unikowe wykorzystują tę technikę w fazie wnioskowania, `PoisoningAttackCleanLabelBackdoor` stosuje ją do tworzenia wyzwalaczy backdoora. Podobnie jak poprzednia metoda, także ta wymaga opakowania modelu we wrapper klasyfikatora ART, ale eliminuje potrzebę znajomości warstwy docelowej:

```
proxy = AdversarialTrainerMadryPGD(KerasClassifier(model), nb_epochs=10, eps=0.15,
↪eps_step=0.001)
proxy.fit(x_train, y_train)
```

Kompletny przykładowy notebook wykorzystujący zbiór danych MNIST do błędnej klasyfikacji wszystkich cyfr jako 9 w hipotetycznym scenariuszu oszustwa na czekach bankowych jest dostępny pod adresem: *https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/poisoning_attack_clean_label_backdoor.ipynb*.

Jak widać, łączenie skuteczności z niewykrywalnością to trudne zadanie dla atakujących, ale także dla obrońców. Techniki zatruwania stają się więc coraz bardziej zaawansowane, czego wynikiem są nowe metody badania modeli i techniki tworzenia perturbacji zatruwających.

# Zaawansowane ataki zatruwające

Ataki z ukrytym wyzwalaczem i clean-label ukazały rosnący stopień wyrafinowania ataków zatruwających. Ta dziedzina nadal się rozwija, a narzędzia takie jak ART, Cleverhans i TextAttack są efektem badań przeprowadzanych na tym polu.

Zaawansowane ataki zatruwające dotyczą bardziej złożonych formatów danych, takich jak tekst naturalny czy pliki audio i wideo. Framework ART zapewnia na przykład ataki i perturbacje dla próbek dźwiękowych, podobne do tych oferowanych dla obrazów. Takie rozwiązanie nie jest w pełni udokumentowane, ale więcej informacji na jego temat można znaleźć w kodzie źródłowym: *https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/art/attacks/poisoning/perturbations/audio_perturbations.py*.

TextAttack to z kolei odpowiednik modułu ART służący do tworzenia adwersarialnych danych tekstowych, które można wykorzystać do zatruwania modeli NLP. Więcej informacji na jego temat można znaleźć na stronie *https://textattack.readthedocs.io/en/latest/*.

Zaawansowane ataki mogą również wykorzystywać wyrafinowane techniki generowania trucizn, do których należą między innymi:

- **Globalne ataki manipulacyjne** (ang. *global manipulation attacks*). Tego typu ataki mają na celu pogorszenie ogólnego działania modelu i zmniejszenie jego skuteczności w szerokim zakresie danych wejściowych. Wprowadzają szeroką gamę zakłóceń i nie są ukierunkowane na określony rodzaj wprowadzanych danych.
- **Ataki zatruwające dostosowane do algorytmu** (ang. *algorithm-specific poisoning attacks*). Tego rodzaju ataki są projektowane tak, aby celować w określone typy algorytmów uczenia maszynowego. Polegają na tworzeniu złośliwych danych, które wykorzystują matematyczne właściwości konkretnego algorytmu uczenia, takiego jak **maszyny wektorów nośnych** (**SVMs** od ang. *Support Vector Machines*).
- **Ataki na gradient** (ang. *gradient-based attacks*). Ataki te polegają na manipulowaniu gradientami, których model używa do uczenia się. Powodują subtelne zmiany wyuczonych parametrów modelu w sposób, który powoduje jego niewłaściwe działanie.

Narzędzie ART realizuje wiele z takich ataków. Oferuje także notatniki z przykładowymi kodami. Więcej szczegółów na ten temat można znaleźć na stronie *https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/attacks/poisoning.html*.

**Uwaga**

Jak się okaże w dalszej części książki, nie jesteśmy w stanie bronić się przed wszystkimi możliwymi atakami, ale powinniśmy zrozumieć ich naturę i uczynić je częścią modelowania zagrożeń i oceny ryzyka. Pomoże nam to ustalić priorytety i zdecydować, przed którymi zagrożeniami należy się zabezpieczyć.

Przejdźmy teraz do kolejnego podrozdziału.

# Zapobieganie i obrona

Po omówieniu różnych typów ataków zatruwających zastanówmy się, jak można się przed nimi bronić i ograniczać związane z nimi ryzyko. Takie działania stanowią połączenie tradycyjnych metod obronnych zintegrowanych w ramach MLOps z technikami uodparniającymi na ataki adwersarialne.

## Obrona cyfrowych twierdz przy użyciu MLOps

Do pewnego stopnia tradycyjne cyberbezpieczeństwo zapewnia mechanizmy obronne, które pomagają chronić się przed zatruwaniem danych. Niektóre mechanizmy obronne, które poznaliśmy w rozdziale 3., mogłyby z powodzeniem sprawdzić się w tym kontekście. Przykładem mogą być tutaj takie metody jak przyznawanie dostępu z minimalnymi uprawnieniami, szyfrowanie oraz korzystanie z funkcji skrótu lub podpisywanie danych. Musimy jednak postrzegać te techniki jako część zintegrowanego systemu obrony łączącego techniki zabezpieczające z automatycznym śledzeniem, zatwierdzaniem, monitorowaniem i alarmowaniem.

W tym miejscu mogą przydać się operacje MLOps. Platformy takie jak AWS SageMaker, MLflow i Azure Machine Learning oferują usługi i mechanizmy obronne pomocne w obronie przed atakami zatruwającymi. Są to między innymi:

- **Wersjonowanie danych i kontrola ich pochodzenia** w celu śledzenia zmian w zbiorach danych na przestrzeni czasu.
- **Walidacja danych** pozwalająca na ciągłe sprawdzanie integralności danych szkoleniowych. Dzięki temu możliwe jest zintegrowanie mechanizmów wykrywania anomalii, które omówimy w następnym punkcie.
- **Wersjonowanie modeli i kontrola ich pochodzenia** jako elementy zarządzania różnymi wersjami wytrenowanych modeli i śledzenia wprowadzonych zmian.
- **Ciągłe monitorowanie** wskaźników wydajności modelu w czasie rzeczywistym. Nieoczekiwany spadek może wskazywać na atak zatruwający.
- **Kontrola dostępu** polegająca na ograniczonym udostępnianiu danych i artefaktów modelu. Tylko upoważniony personel powinien być w stanie modyfikować te zasoby lub w ogóle mieć do nich dostęp. Ten środek jest zintegrowany z silnym uwierzytelnianiem i dodatkowymi mechanizmami autoryzacji.
- **Interpretowalność modelu** umożliwiająca zrozumienie jego działania. Jeśli model zaczyna funkcjonować w nieoczekiwany sposób dla określonych danych wejściowych, narzędzia interpretacyjne mogą pomóc zidentyfikować, czy pewnym cechom nie nadano dodatkowego znaczenia, co może być oznaką zatrucia.
- **Monitorowanie, logowanie i powiadamianie** w celu zapewnienia pozyskiwania informacji o podejrzanych aktywnościach, takich jak zmiany w danych czy trenowanie modeli w nietypowych godzinach.

- **Zarządzanie i współpraca** ułatwiające dzielenie się zasobami przy zastosowaniu odpowiednich zabezpieczeń, takich jak konieczność zatwierdzania dodatkowego dostępu lub wprowadzania znaczących zmian w modelu.

W dalszej części książki zastanowimy się, które operacje MLOps są istotne dla poszczególnych obszarów adwersarialnej sztucznej inteligencji. Na przykład w rozdziale 6. skupimy się na podatnościach łańcucha dostaw i wyjaśnimy, jaki jest związek między pozyskiwaniem danych i modeli z innych źródeł a podatnością na ataki zatruwające. W rozdziale 16., w którym skupimy się na regułach MLSecOps, przedstawimy przykładową implementację całego procesu.

Teraz przejdziemy do omówienia konkretnych technik i mechanizmów obronnych, których możemy używać w połączeniu z zasadami MLOps.

# Wykrywanie anomalii

**Wykrywanie anomalii** to technika pozwalająca identyfikować wśród danych wzory prowadzące do nieoczekiwanego działania modelu. Te odbiegające od normy wzory określa się mianem anomalii lub wartości odstających. W kontekście wykrywania ataków zatruwających algorytmy wykrywania anomalii wyszukują w zestawie treningowym dane, które znacznie odbiegają od większości, co może wskazywać na celową manipulację. Tej techniki możemy używać zarówno doraźnie, jak i w pełnej integracji z potokami MLOps w ramach walidacji danych.

## W czym pomaga wykrywanie anomalii?

Wykrywanie anomalii stanowi skuteczną ochronę przed zatruwaniem danych, zwłaszcza gdy jest realizowane w sposób zautomatyzowany. Może pomóc w następujących kwestiach:

- **Identyfikacja podejrzanych danych.** Wskazywanie danych, które wydają się odbiegać od normy, może być pomocne w wyszukiwaniu w zestawie treningowym takich pozycji, które mogły zostać wprowadzone lub zmodyfikowane przez atakującego.
- **Zautomatyzowane monitorowanie.** Wykrywanie anomalii można zautomatyzować w celu ciągłego monitorowania odbieranych danych treningowych i generowania w czasie rzeczywistym alertów informujących o przypadkach wykrycia podejrzanych danych.
- **Ograniczanie fałszywych alarmów.** Podczas gdy ręczna inspekcja może skutkować oznaczeniem zdrowych danych jako złośliwych z powodu błędu ludzkiego, to odpowiednio dostrojone algorytmy wykrywania anomalii mogą zmniejszyć liczbę takich fałszywych alarmów.

## Techniki wykrywania anomalii w kontekście ataków zatruwających

Istnieje szereg technik wykrywania anomalii — od łatwych w użyciu metod statystycznych po bardziej wyrafinowane rozwiązania oparte na uczeniu maszynowym. Są to między innymi:

- **Metody statystyczne** — polegają na obliczaniu wskaźników statystycznych, takich jak średnia i odchylenie standardowe, a następnie oznaczaniu tych danych, które znacznie odbiegają od wyliczonych miar. Popularną miarą jest na przykład wartość z-score, która mówi, o ile odchyleń standardowych konkretny punkt danych odbiega od średniej.
- **Metody klastrujące** — techniki takie jak k-średnich (jeden z algorytmów grupowania) mogą służyć do grupowania podobnych punktów danych. Te wartości, które nie pasują wyraźnie do żadnego klastra, można uznać za anomalie.
- **Sieci neuronowe** — modele głębokiego uczenia, zwłaszcza autokodery, można wytrenować do rekonstruowania danych wejściowych. Jeśli błąd rekonstrukcji dla punktu danych będzie zbyt wysoki, stosowną wartość można uznać za anomalię.
- **Metody gęstościowe** — algorytmy takie jak DBSCAN analizują gęstość punktów danych w określonym obszarze. Istnienie odosobnionych obszarów z niewielką liczbą danych może wskazywać na obecność anomalii.

### Wyzwania i uwarunkowania

Wykrywanie anomalii może być skutecznym środkiem bezpieczeństwa w kontekście zatruwania danych, ale przy wprowadzaniu tego mechanizmu musimy wziąć pod uwagę następujące kwestie:

- **Dostrajanie.** Algorytmy wykrywania anomalii często obejmują parametry, które należy starannie dobrać. Nieprawidłowe dostrojenie ich wartości może prowadzić do wielu fałszywych wskazań (dodatnich lub ujemnych).
- **Zmieniające się dane.** W miarę jak do systemu wprowadzane są nowe, prawidłowe dane, mogą zmieniać się kryteria rozpoznawania. Aby zachować swoją skuteczność, systemy wykrywania anomalii muszą dostosowywać się do tych zmian.
- **Ukryte ataki.** W przypadku predykcyjnej sztucznej inteligencji wyrafinowani napastnicy mogą wprowadzać zatrute dane, które bardzo przypominają te prawdziwe. W rezultacie programy wykrywające anomalie mają problemy z ich identyfikacją. Jak zobaczymy w rozdziale 14., generatywna sztuczna inteligencja, zwłaszcza z modelami LLM, wprowadza nowe zagrożenia wynikające z wykorzystywania danych w skali całego internetu, które pochodzą spoza obszaru kontroli przedsiębiorstwa. Połączenie ogromnej skali tych danych z brakiem nadzoru nad nimi sprawia, że trudniej jest je weryfikować.

## Testy odporności na zatruwanie

Ogólne wykrywanie anomalii ma zastosowanie w odniesieniu do dowolnych zbiorów danych i nie uwzględnia wpływu na algorytmy uczenia maszynowego. Ataki zatruwające stają się jednak coraz bardziej wyrafinowane i wykorzystują zaawansowane mechanizmy do generowania niezauważalnych perturbacji zatruwających model. Użycie tego, co może wydawać się prawidłowymi danymi, może mieć druzgocące konsekwencje.

Bardziej kontekstowym podejściem do wykrywania zatruć jest testowanie modelu pod kątem odporności na ataki zatruwające.

Prostym sposobem mogłoby być użycie **rekordów kanarkowych** (ang. *canary records*). Byłby to niewielki zbiór rekordów jednoznacznie należących do określonej klasy. Na przykład zestaw może składać się z pięciu obrazów, które powinny być w oczywisty sposób klasyfikowane jako samoloty. Jeśli zostaną błędnie przyporządkowane, może to sygnalizować zatrucie danych. Podejście jest łatwe do wdrożenia i może identyfikować proste próby zatrucia, ale nie wykryje ataków typu backdoor, ponieważ rekordy kanarkowe nie będą zawierać wyzwalaczy.

Do testowania modeli pod kątem podatności na backdoory można wykorzystać perturbacje zatruwające dostępne w ART jako wyzwalacze backdoorów:

- `add_single_bd` — jest to metoda prosta w użyciu i można ją szybko zaimplementować dla dużej partii danych. Pozwala szybko ustalić, jak wrażliwy jest nasz model na choćby najmniejsze zmiany. Takie podejście jest szczególnie użyteczne na przykład w scenariuszu, w którym chcemy przetestować, jak system rozpoznawania twarzy reaguje na najdrobniejsze nawet zmiany.
- `add_pattern_bd` — jest to kolejne proste i szybkie w realizacji podejście pozwalające przetestować nieco bardziej złożone luki w zabezpieczeniach. Możemy na przykład przeanalizować trening modelu rozpoznawania znaków drogowych w autonomicznych samochodach i zbadać jego odporność na drobne perturbacje.
- `insert_image` — jest to najbardziej konfigurowalna i złożona opcja. Doskonale sprawdza się w testowaniu zaawansowanych wzorów i omijania filtrowania treści. Można ją stosować na przykład w przypadku bardziej złożonych zakłóceń w systemach rozpoznawania znaków drogowych lub oprogramowaniu OCR, a także wtedy, gdy chcemy sprawdzić, czy algorytm moderacji treści nadal potrafi poprawnie klasyfikować obraz po wstawieniu do niego określonego logo lub symbolu.

Są to techniki odpowiednie dla systemów operujących na obrazach. Framework ART oferuje także perturbacje dla próbek dźwiękowych, natomiast TextAttack pozwala tworzyć dane i przypadki testowe dla modeli NLP.

## Zaawansowana ochrona przed zatruciem realizowana z użyciem narzędzi ART

Podczas gdy ogólne wykrywanie anomalii nie ocenia wpływu na model, to testy odporności mogą zbytnio koncentrować się na konkretnych przypadkach użycia. Framework ART oferuje kilka mechanizmów obrony przed zatruciem, które znajdują się gdzieś pośrodku. Opierają się one na analizach badawczych i łączą w sobie zaawansowane techniki wykrywania anomalii testowane na modelu w celu określenia wpływu danych na jego działanie.

Techniki te można zaimplementować jako wykonywalne programy w notatnikach lub zintegrować z potokami MLOps. Wyróżniamy następujące rodzaje takich mechanizmów:

- **Obrona oparta na analizie aktywacji** — jest szczególnie przydatna, gdy chcemy sprawdzić, czy wewnętrzne aktywacje modelu sieci neuronowej są realizowane w sposób nieprawidłowy, co mogłoby sugerować, że model przetwarza zatrute dane.
- **Obrona oparta na pochodzeniu danych** — ma zastosowanie, gdy źródło danych jest wątpliwe lub nie zostało dokładnie zweryfikowane. Ten rodzaj obrony pozwala uzyskać pewność, że dane pochodzą z wiarygodnego, uwierzytelnionego źródła. Ciągła weryfikacja źródła lub pełne zaufanie do niego może być jednak niepraktyczne lub ryzykowne. Poza tym potwierdzenie pochodzenia danych nie gwarantuje, że same dane nie są zatrute.
- **Metoda RONI** (ang. *Reject on Negative Impact*, czyli odrzucanie danych o negatywnym wpływie) — często używa się jej do wykrywania ataków zatruwających, które mają zauważalny wpływ na działanie modelu. W ramach tej techniki ocenia się wpływ każdego elementu treningowego na ogólną skuteczność modelu. Dane, które znacząco ją pogarszają, uznaje się za potencjalnie zatrute i są usuwane. Metoda ta często wymaga dużej mocy obliczeniowej, ponieważ konieczne może być wielokrotne trenowanie modelu. Poza tym bywa nieskuteczna w przypadku subtelnych ataków zatruwających, które nie od razu pogarszają działanie modelu.

  Zazwyczaj trenuje się dwie wersje modelu: jedną z podejrzanym punktem danych i drugą bez niego. Następnie porównuje się skuteczności obu modeli. Jeśli dodanie określonego punktu danych ma negatywny wpływ, może to oznaczać, że jest zatruty.
- **Obrona oparta na sygnaturze spektralnej** — służy do identyfikacji ataków zatruwających, w których punkty danych znacząco odbiegają od typowych danych w przekształconej domenie, na przykład w domenie częstotliwości.

  Tego typu techniki mogą wymagać złożonych obliczeń i konfiguracji sprzętowych. Narzędzia takie jak ART upraszczają stosowanie tych technik i ich integrację z potokami MLOps. Więcej informacji na ich temat można znaleźć w dokumentacji ART i notatnikach:

  - *https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/defences/detector_poisoning.html*,
  - *https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/poisoning_defense_activation_clustering.ipynb*,
  - *https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/poisoning_defense_spectral_signatures.ipynb*,
  - *https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/notebooks/provenance_defence.ipynb*.

Czwarty notatnik zawiera przykład demonstrujący zarówno kontrolę pochodzenia danych, jak i implementację zabezpieczeń RONI.

Opisane powyżej mechanizmy obronne to dziedzina, która rozwija się błyskawicznie. Elementem obrony przed adwersarialną sztuczną inteligencją, w tym zatruwaniem danych, jest śledzenie najnowszych koncepcji i wybieranie odpowiednich środków ochronnych oraz ich stosowanie i automatyzowanie.

## Trening adwersarialny

Ten mechanizm obronny polega bardziej na łagodzeniu skutków ataku niż jego wykrywaniu. Dotyczy bezpośrednio zatrutych danych, które są prawidłowo klasyfikowane. Trening adwersarialny sprawia, że modele są odporne na takie ataki. Technika ta pomaga ograniczyć skutki ataków w fazie wnioskowania (ataków unikania), dlatego omówimy ją bardziej szczegółowo w części poświęconej takim właśnie atakom.

## Budowanie strategii obronnej

Ataki zatruwające są zasadniczo atakami realizowanymi na etapie rozwoju modelu. Podstawową strategią obronną, którą należy wdrożyć, jest proces MLOps. Wprowadza on kilka podstawowych zasad zapewniających identyfikowalność danych w ramach cyklu życia rozwiązania AI/ML. Należą do nich między innymi:

- **Automatyzacja i orkiestracja pipeline'ów** — obejmujące nie tylko aplikacje, ale także dane i modele; przygotowuje się więc trzy podstawowe pipeline'y: aplikacji, danych i ML.
- **Ciągłe X** (ang. *continous X*) — wzbogacenie praktyk ciągłej integracji (CI) i ciągłego dostarczania (CD) o trening modeli, ich obsługę i monitorowanie.
- **Zarządzanie wersjami** — zarówno danych, jak i modeli, realizowane w ramach rejestru modeli używających magazynów cech, które ułatwiają zarządzanie wielokrotnym stosowaniem tychże cech.
- **Śledzenie eksperymentów** — rozszerza kontrolę wersji kodu źródłowego o śledzenie używanych danych, modeli, a także ich wag i biasów.
- **Testowanie** — dotyczy danych i modeli w trzech podstawowych pipeline'ach i obejmuje nie tylko testy funkcjonalne, ale także testy bezpieczeństwa, odporności, biasu, wydajności i jakości.
- **Monitorowanie** — rozszerza monitoring systemu o kontrolowanie wydajności modeli i ich działania.

Zasady te zostały omówione bardziej szczegółowo na stronie *https://ml-ops.org/content/mlops-principles*.

Po wprowadzeniu powyższych reguł MLOps do cyklu życia rozwiązania AI możemy zaprojektować punkty kontrolne i zabezpieczenia, które pomogą nam kontrolować pochodzenie danych i przebieg zmian w modelach.

Zasady MLOps stanowią kluczowy element strategii obronnej. Należy jednak mieć świadomość, że nie będziemy narażeni na wszystkie rodzaje ataków, dlatego nie trzeba wdrażać wszystkich zabezpieczeń. Jak przekonamy się w rozdziale 14., dzięki modelowaniu zagrożeń nauczymy się decydować, które zabezpieczenia są najbardziej rozpowszechnione i najbardziej wartościowe. We wspomnianym rozdziale 14. przyjrzymy się, jak dodać technikę modelowania zagrożeń do strategii tworzenia AI opartej na zasadzie „secure-by-design" i jak ustalać priorytety zabezpieczeń.

# Podsumowanie

Dokładnie zbadaliśmy ataki zatruwające — pierwszy niezwykle istotny obszar adwersarialnej sztucznej inteligencji. Tego typu ataki są realizowane na etapie tworzenia modeli, dlatego mogą być trudne do wykrycia. Omówiliśmy podstawowe koncepcje i przyjrzeliśmy się konkretnym przykładom. Przeanalizowaliśmy dokładnie realizację prostych i bardziej zaawansowanych ataków zatruwających. Wiedza ta może pomóc w testowaniu i ocenie modeli pod kątem ewentualnego zatrucia. Dowiedzieliśmy się również o innych sposobach obrony przed zatruwaniem danych, w tym wdrażaniu potoków MLOps, wykrywaniu anomalii i zaawansowanych mechanizmach obronnych oferowanych przez zestaw narzędzi ART.

Ataki zatruwające wymagają dostępu do danych treningowych. Polegają na zakłócaniu procesu szkolenia modelu w celu podważenia jego integralności. W następnym rozdziale przyjrzymy się innej metodzie wszczepiania backdoorów (bez zatruwania danych) i obniżania integralności modelu przez manipulowanie nim, przeprogramowanie go lub wprowadzenie konia trojańskiego.

# Skorowidz

## A

## B

## C



## D



## E



## F



## G

## H



## I



## J



## K



## L



## Ł



## M

## N



## O

## P



## Q



## R

## S



## Ś



## T

## U



## V



## W



## Z

# Podejmij wyzwanie: ochroń AI przed nadużyciami!

Wraz z rozwojem rewolucyjnych możliwości modeli AI pojawiają się nowe zagrożenia. Może to być na przykład manipulowanie działaniem sztucznej inteligencji, by celowo doprowadzić do błędnych decyzji. Tak właśnie prowadzi się ataki adwersarialne. Konsekwencje takich manipulacji, jak również innych, mogą być bardzo poważne. Jednak zrozumienie ich istoty i wdrożenie adekwatnych zabezpieczeń stanowi ogromne wyzwanie.

Tę książkę docenią specjaliści do spraw cyberbezpieczeństwa, którzy chcą zdobyć umiejętności zabezpieczania systemów AI. Znajdą w niej uporządkowaną prezentację wyników badań i najnowszych standardów branżowych, z uwzględnieniem klasyfikacji: MITRE, NIST i OWASP. W przewodniku omówiono strategię zabezpieczania AI już na etapie projektowania — z wykorzystaniem modelowania zagrożeń, przy czym skoncentrowano się na integracji MLSecOps i LLMOps z systemami bezpieczeństwa przedsiębiorstwa. Dodatkowo przedstawiono przykłady wdrażania integracji ciągłej, strategii i narzędzi MLOps, a także mechanizmów kontroli bezpieczeństwa. Zaproponowano ponadto bazujący na klasycznych filarach NIST plan wzmacniania bezpieczeństwa AI w przedsiębiorstwie.

Ciekawsze zagadnienia:

- zatruwanie danych, omijanie zabezpieczeń i naruszanie prywatności
- użycie sieci GAN do przeprowadzania ataków i generowania deepfake'ów
- nowe zagrożenia bezpieczeństwa LLM, w tym wstrzykiwanie promptów i ekstrakcja danych
- zatruwanie LLM metodami: RAG, osadzeń i dostrajania
- nowe zagrożenia związane z łańcuchem dostaw i modelami LLM o otwartym dostępie
- wdrażanie operacji MLSecOps z integracją ciągłą MLOps i SBOM

**John Sotiropoulos** jest starszym architektem zabezpieczeń w firmie Kainos. Odpowiada za bezpieczeństwo AI i pracuje nad zabezpieczeniem systemów krajowych w rządzie, organach regulacyjnych i służbie zdrowia. Współpracuje też z organizacjami normalizacyjnymi i krajowymi agencjami do spraw cyberbezpieczeństwa. Prywatnie jest zapalonym geekiem i maratończykiem.

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-2235-8
9 788328 922358
Cena: 129,00 zł
packt