Adversarial Machine Learning: An industrial perspective on types of attacks and getting started with ART framework

Samridhi Agrawal
University of Washington
Bothell Campus

Introduction to Adversarial Machine Learning

Machine learning models are computer programs which are designed to learn to recognize patterns in data. With the help from humans supplying “training data,” algorithms known as “classifiers” can be taught how to respond to different inputs. After repeated exposure to training data, these models are designed to make increasingly accurate assessments over time.[1]

Machine learning is a key enabling technology behind artificial intelligence (AI), and is used for such valuable applications as email spam filters and malware detection, as well as more complex technologies like speech recognition, facial recognition, robotics, and self-driving cars.

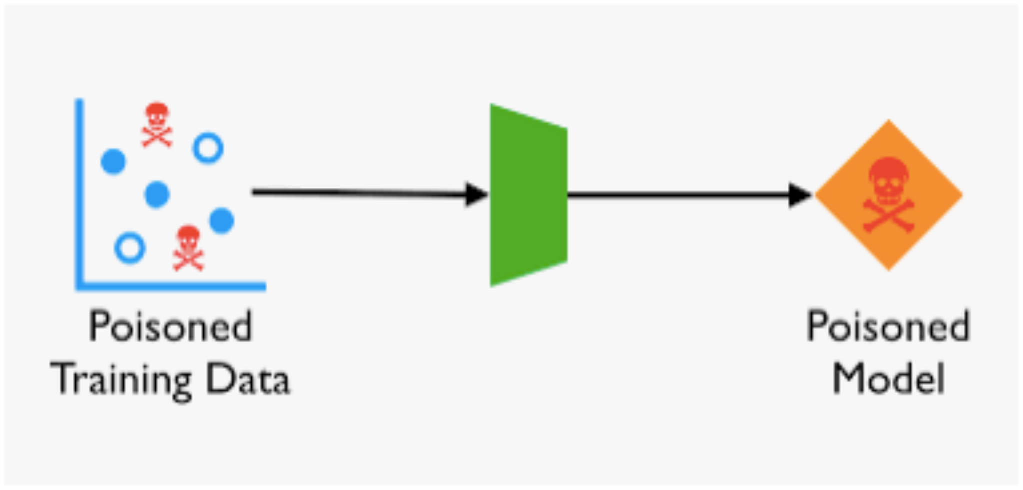
While machine learning models have many potential benefits, they may be vulnerable to manipulation. Cybersecurity researchers refer to this risk as “adversarial machine learning,” as AI systems can be deceived (by attackers or “adversaries”) into making incorrect assessments. An adversarial attack might entail presenting a machine-learning model with inaccurate or misrepresentative data as it is training or introducing maliciously designed data to deceive an already trained model into making errors.

Adversarial machine learning is a machine learning technique that attempts to fool models by supplying deceptive input. The most common reason is to cause a malfunction in a machine learning model. Adversarial examples are inputs provided to the machine learning model that are intentionally designed to make the model misclassify. The actual changes are very small and usually go unnoticed by humans.

Types of Attacks

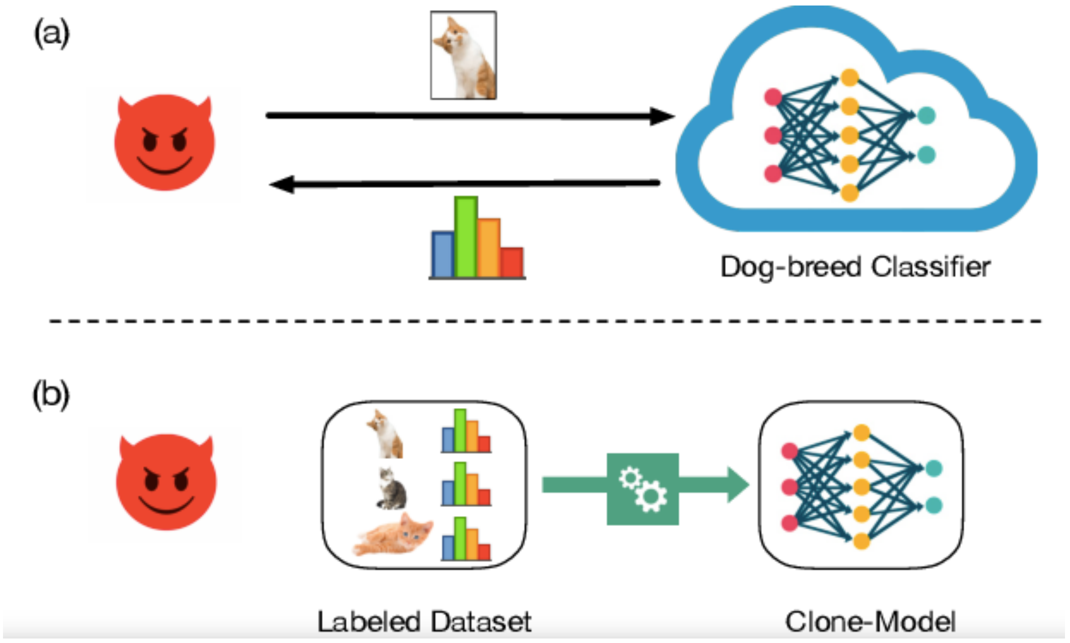
Poisoning

Data Poisoning is an adversarial attack that tries to manipulate the training dataset in order to control the prediction behavior of a trained model such that the model will label malicious examples into a desired class. Data poisoning has caught the attention of enterprises, perhaps because of the cultural significance of Tay. A medium sized financial tech put it thus, “We use ML systems to suggest tips and financial products for our users. The integrity of our ML system matters a lot. Worried about inappropriate recommendation like attack on Tay” [2]
In a poisoning attack, the attacker compromises the learning process in a way that the system fails on the inputs chosen by the attacker and further constructs a backdoor through which he can control the output even in future.



Model Stealing

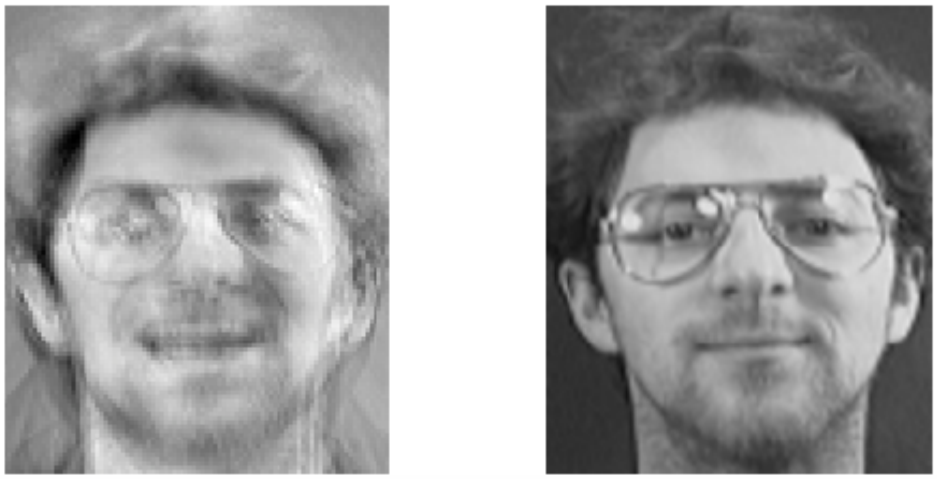
Model stealing, which involves an adversary attempting to counterfeit the functionality of a target victim ML model by exploiting black-box access (query inputs in, posterior predictions out). Model Stealing that can lead to loss of Intellectual property is another concern. A large retail organization said, “We run a proprietary algorithm to solve our problem and it would be worrisome if someone can reverse engineer it”.[3]



(a) An adversary queries the target model (dog-breed classifier) using synthetic/surrogate data (cat images) and constructs a labeled dataset using the predictions of the model
(b) The labeled dataset can then be used to train a clone model that replicates the functionality of the target model.

Model Inversion

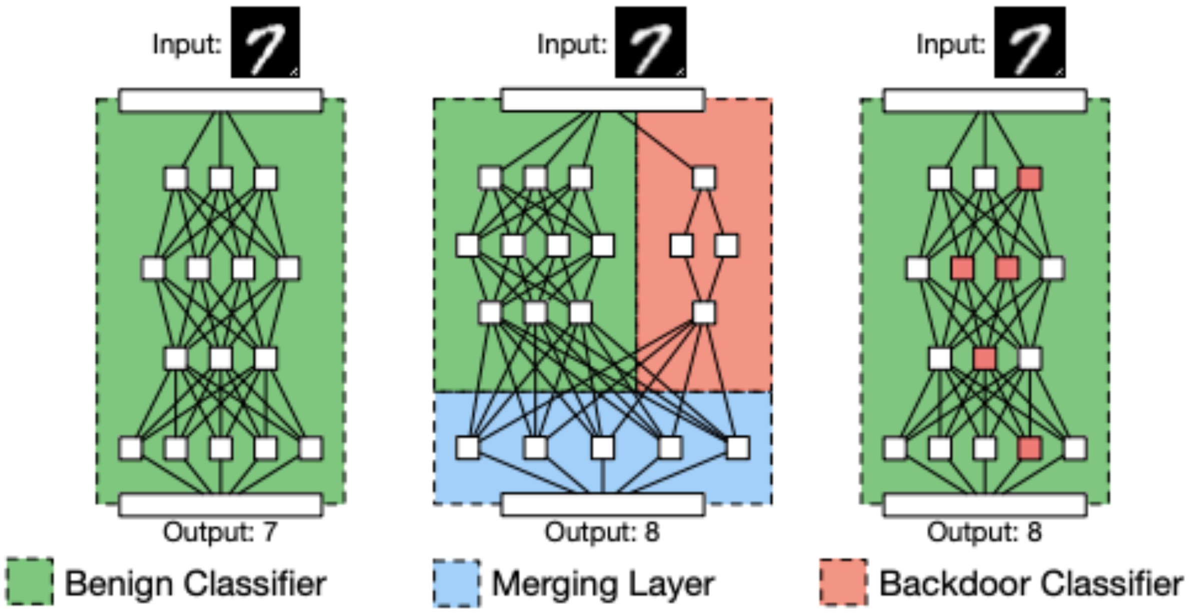
Model inversion (MI) attacks are aimed at reconstructing training data from model parameters. Such attacks have triggered increasing concerns about privacy, especially given a growing number of online model repositories.



An image recovered using a new model in version attack (left) and a training set image of the victim (right). The attacker is given only the per son’s name and access to a facial recognition system that returns a class confidence score.[4]

Backdoored ML

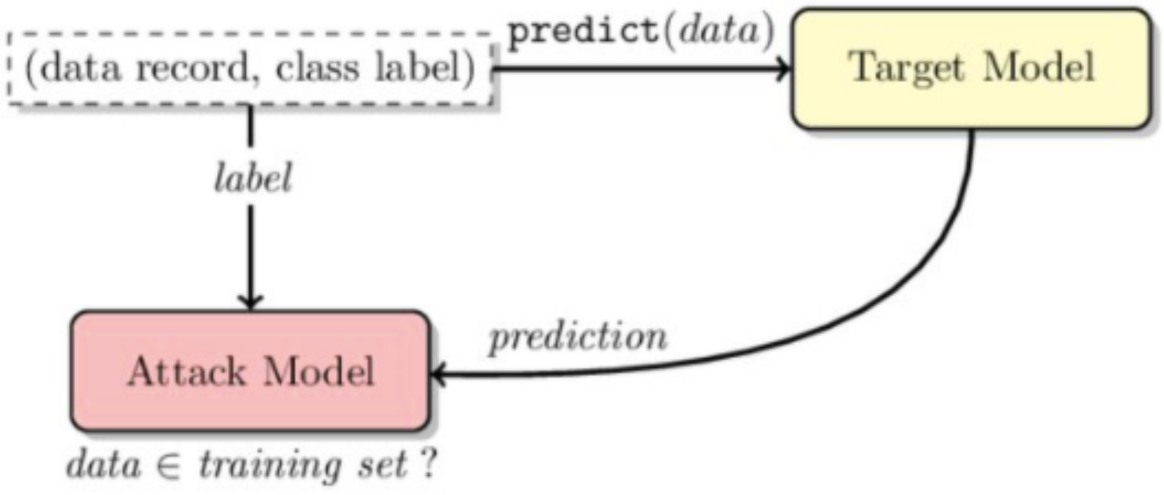
Backdooring is a training time attack, i.e., the adversary is the one who trains the ML model. To achieve this, we assume the adversary can access the data used for training the model and control the training process. Then, the adversary publishes the backdoored model to the victim. It is a form of adversarial attacks on deep networks where the attacker provides poisoned data to the victim to train the model with, and then activates the attack by showing a specific small trigger pattern at the test time.



Approaches to backdooring a neural network. On the left, a clean network correctly classifies its input. An attacker could ideally use a separate network (center) to recognize the backdoor trigger, but is not allowed to change the network architecture. Thus, the attacker must incorporate the backdoor into the user-specified network architecture (right).[5]

Membership Inference

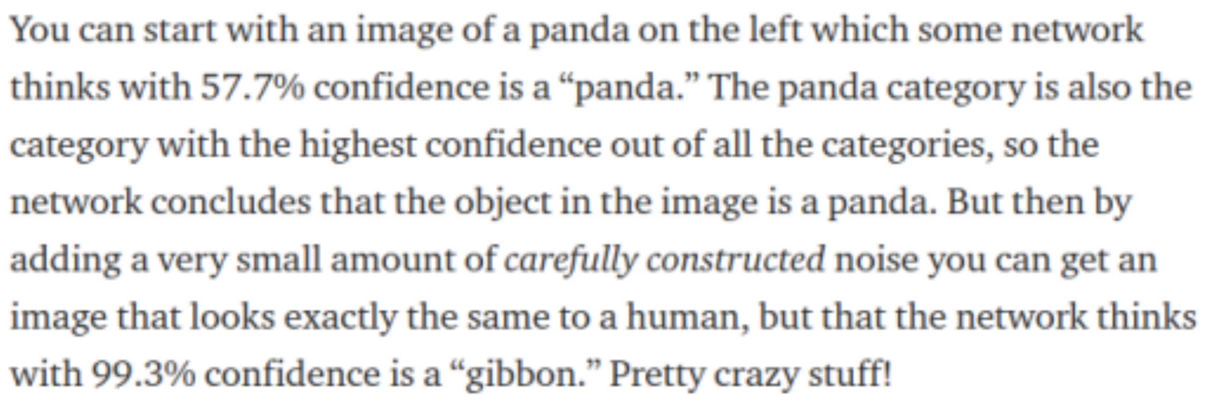
A type of attack called "membership inference" makes it possible to detect the data used to train a machine learning model. In many cases, the attackers can stage membership inference attacks without having access to the machine learning model's parameters and just by observing its output.



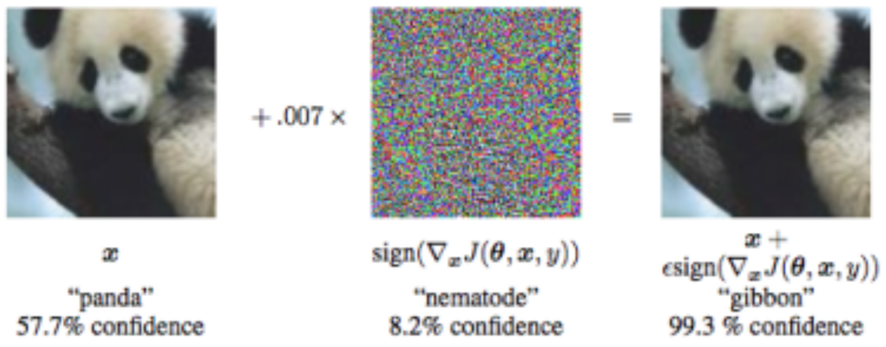
Membership inference attack in the black-box setting. The attacker queries the target model with a data record and obtains the model's prediction on that record. The prediction is a vector of probabilities, one per class, that the record belongs to a certain class. This prediction vector, along with the label of the target record, is passed to the attack model, which infers whether the record was in or out of the target model's training dataset.[6]

Adversarial Examples

Adversarial examples are specialized inputs created with the purpose of confusing a neural network, resulting in the misclassification of a given input. These notorious inputs are indistinguishable to the human eye but cause the network to fail to identify the contents of the image.[7]



You can start with an image of a panda on the left which some network thinks with 57.7% confidence is a "panda." The panda category is also the category with the highest confidence out of all the categories, so the network concludes that the object in the image is a panda. But then by adding a very small amount of *carefully constructed* noise you can get an image that looks exactly the same to a human, but that the network thinks with 99.3% confidence is a "gibbon." Pretty crazy stuff!


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

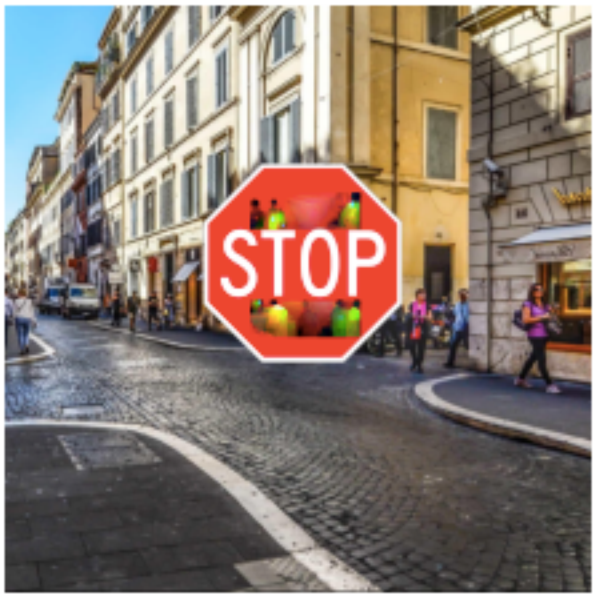
From Explaining and Harnessing Adversarial Examples by Goodfellow et al.

Reprogramming ML System

Adversarial Reprogramming is a class of attacks where a model is repurposed to perform a new task. An adversarial program can be thought of as an additive contribution to network input. An additive offset to a neural network's input is equivalent to a modification of its first layer biases. In case of a convolutional neural network, new parameters are effectively introduced. These kinds of tiny updates in the network are an adversarial program. The attacker may try to adversarial reprogram across tasks with very different datasets. This makes the task of adversarial reprogramming more challenging.[8]

Adversarial Example in Physical Domain

An adversarial example for the voice command domain would consist of a recording that seems to be innocuous to a human observer (such as a song) but contains voice commands recognized by a machine learning algorithm. Adversarial Examples in the physical domain, resonated with the respondents, but did not rank high on the list. One reason may be that the organizations we spoke to did not have physical component like cars or drones.[9]



An example of an adversarial perturbation overlaid on a synthetic background. The Stop sign in the image is printed such that it is the same size as a U.S. Stop sign. Then, we cut out the two rectangle bars, and use the original print as a stencil to position the cutouts on a real Stop sign.

Malicious ML provider recovering training data

Malicious ML provider can query the model used by customer and recover customer's training data. Researchers show how a malicious provider presents a backdoored algorithm, wherein the private training data is recovered. They were able to reconstruct faces and texts, given the model alone. [10]

Attacking the ML supply chain

A supply chain attack is a cyber-attack that seeks to damage an organization by targeting less-secure elements in the supply chain. A supply chain attack can occur in any industry, from the financial sector, oil industry, to a government sector.[11]

Exploit Software Dependencies

A dependency confusion attack or supply chain substitution attack occurs when a software installer script is tricked into pulling a malicious code file from a public repository instead of the intended file of the same name from an internal repository.

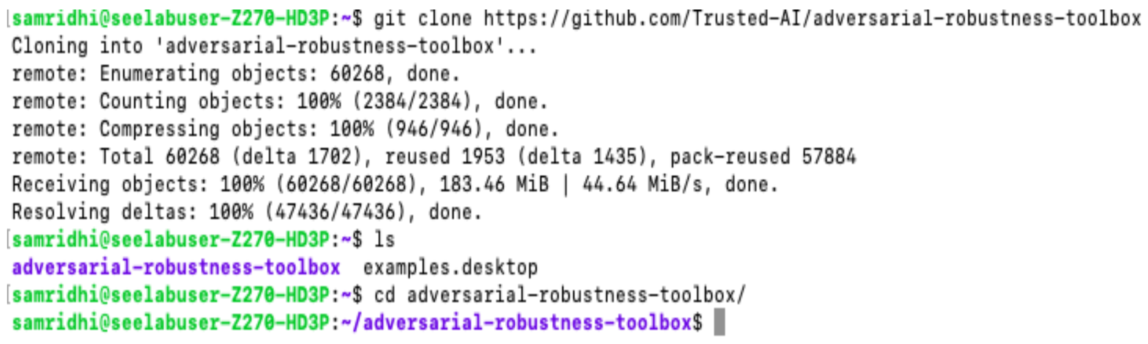
According to [12] among all the organizations practicing secure machine learning, poisoning has been the most frequent attacks.

Poisoning of data and its impacts

Setup of ART (Adversarial robustness toolbox)

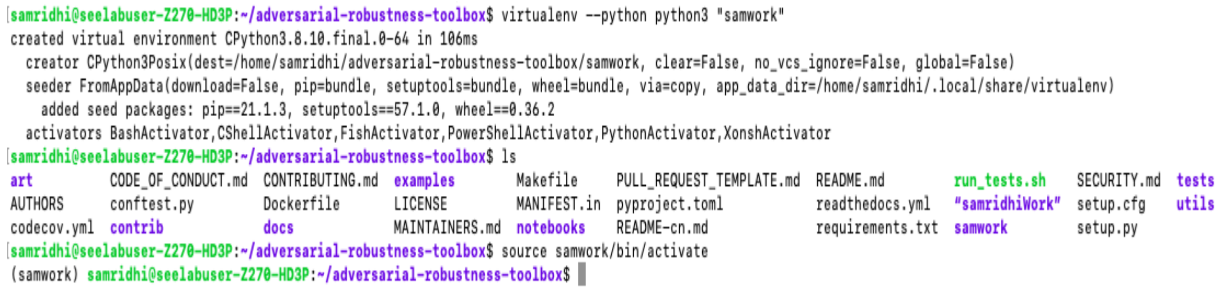
Adversarial Robustness Toolbox (ART) is a Python library for Machine Learning Security. ART provides tools that enable developers and researchers to defend and evaluate Machine Learning models and applications against the adversarial threats of Evasion, Poisoning, Extraction, and Inference.

```
git clone https://github.com/Trusted-AI/adversarial-robustness-toolbox
cd adversarial-robustness-toolbox/
```



```
samridhi@seelabuser-Z270-HD3P:~$ git clone https://github.com/Trusted-AI/adversarial-robustness-toolbox
Cloning into 'adversarial-robustness-toolbox'...
remote: Enumerating objects: 60268, done.
remote: Counting objects: 100% (2384/2384), done.
remote: Compressing objects: 100% (946/946), done.
remote: Total 60268 (delta 1702), reused 1953 (delta 1435), pack-reused 57884
Receiving objects: 100% (60268/60268), 183.46 MiB | 44.64 MiB/s, done.
Resolving deltas: 100% (47436/47436), done.
samridhi@seelabuser-Z270-HD3P:~$ ls
adversarial-robustness-toolbox  examples.desktop
samridhi@seelabuser-Z270-HD3P:~$ cd adversarial-robustness-toolbox/
samridhi@seelabuser-Z270-HD3P:~/adversarial-robustness-toolbox$
```

<u>Setting up the virtual environment :</u>
$ virtualenv --python python3 "samwork"
$ source samwork/bin/activate
$ pip install --upgrade pip
$ pip3 install tensorflow
$ pip install -r requirements.txt



```
samridhi@seelabuser-Z270-HD3P:~/adversarial-robustness-toolbox$ virtualenv --python python3 "samwork"
created virtual environment CPython3.8.10.final.0-64 in 106ms
  creator CPython3Posix(dest=/home/samridhi/adversarial-robustness-toolbox/samwork, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/samridhi/.local/share/virtualenv)
    added seed packages: pip==21.1.3, setuptools==57.1.0, wheel==0.36.2
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
samridhi@seelabuser-Z270-HD3P:~/adversarial-robustness-toolbox$ ls
art         CODE_OF_CONDUCT.md  CONTRIBUTING.md  examples        Makefile     PULL_REQUEST_TEMPLATE.md  README.md         run_tests.sh    SECURITY.md  tests
AUTHORS     conftest.py         Dockerfile       LICENSE         MANIFEST.in  pyproject.toml            readthedocs.yml   "samridhiWork"  setup.cfg    utils
codecov.yml contrib             docs             MAINTAINERS.md  notebooks    README-cn.md              requirements.txt  samwork         setup.py
samridhi@seelabuser-Z270-HD3P:~/adversarial-robustness-toolbox$ source samwork/bin/activate
(samwork) samridhi@seelabuser-Z270-HD3P:~/adversarial-robustness-toolbox$
```

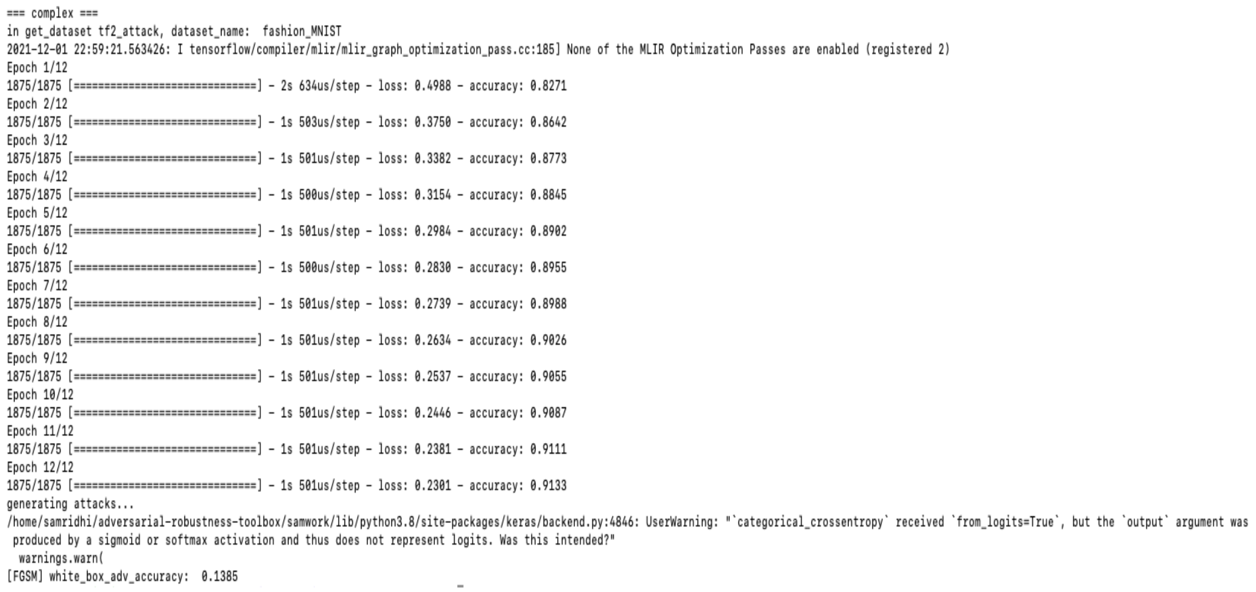
Demo

Running the basic adversarial example in the python virtual environment, with the tensor flow dataset which takes MNIST data as an input and created the model and trainied, an attack ‘FGSM’ (fast gradient method) adds up noise in the data, resulting in decrease in accuracy.

```
from sklearn.svm import LinearSVC
import numpy as np
import random
import os
from art import config
from sklearn.svm import LinearSVC
# python3 tf2_TF_entropy.py fashion_MNIST FGSM 0.25
# python3 tf2_TF_entropy.py digits_MNIST FGSM 0.25
# python3 tf2_TF_entropy.py cifar10 FrameSaliencyAttack 0.25

from art.attacks.evasion import
FastGradientMethod,AutoProjectedGradientDescent,FrameSaliencyAttack
from art.estimators.classification import SklearnClassifier
from art.utils import load_mnist, load_cifar10

from spatialentropy import altieri_entropy,leibovici_entropy

from sklearn.metrics import precision_recall_fscore_support,accuracy_score
# from test_2D_matrix import two_D_matrix_entropy_main
from collections import defaultdict
import statistics
from sklearn.decomposition import PCA
from art.attacks.evasion.carlini import CarliniL2Method, CarliniLInfMethod
from art.attacks.evasion.deepfool import DeepFool
import sys
from spatialentropy import altieri_entropy,leibovici_entropy
import tensorflow as tf
from art.estimators.classification import TensorFlowV2Classifier
from keras.utils import np_utils
from sklearn.metrics import confusion_matrix
import csv

#from tf2_architectures import get_tf2_architecture
#from tf2_attack import main_attack,conduct_PCA
# from tf2_entropy import get_data,find_max

from tensorflow.keras import Model
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D
import math
# from sklearn.metrics import mean_squared_error,r2_score,
mean_olute_error
from datetime import datetime
from scipy.stats import spearmanr,kendalltau
import statistics
from tensorflow.keras import Model, initializers, Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D,
Dropout,BatchNormalization
from tensorflow.keras.optimizers import SGD,Adam
import tensorflow as tf
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
attack_algorithm = "FGSM"
def get_Fashion_MNIST(raw = False):
  fashion_mnist = tf.keras.datasets.fashion_mnist
  (x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

  min_pixel_value = 0.0
  max_pixel_value = 255.0

  x_train = x_train.astype('float32')
  x_test = x_test.astype('float32')
  x_train = x_train / 255.0
  x_test = x_test / 255.0
  y_train = np_utils.to_categorical(y_train)
  y_test = np_utils.to_categorical(y_test)
  nb_samples_train = x_train.shape[0]
  col = x_train.shape[1]
  x_train = x_train.reshape((nb_samples_train, col,col,1))
  nb_samples_test = x_test.shape[0]
  col = x_test.shape[1]
  x_test = x_test.reshape((nb_samples_test, col,col,1))

  return (x_train, y_train), (x_test, y_test), min_pixel_value,
max_pixel_value

def get_dataset(dataset_name = None):
  print("in get_dataset tf2_attack, dataset_name: ",dataset_name)
  if dataset_name == "digits_MNIST":
    (x_train, y_train), (x_test, y_test), min_pixel_value, max_pixel_value
= load_mnist()
  elif dataset_name == "fashion_MNIST":
    if attack_algorithm == "FrameSaliencyAttack":
        (x_train, y_train), (x_test, y_test), min_pixel_value,
max_pixel_value = get_Fashion_MNIST(raw = True)
    else:
        (x_train, y_train), (x_test, y_test), min_pixel_value,
max_pixel_value = get_Fashion_MNIST()
  elif dataset_name == "cifar10":
    (x_train, y_train), (x_test, y_test), min_pixel_value, max_pixel_value
= load_cifar10()
  return (x_train, y_train), (x_test, y_test), min_pixel_value,
max_pixel_value

def train_step(model, images, labels):
  with tf.GradientTape() as tape:
      predictions = model(images, training=True)
      loss = loss_object(labels, predictions)
  gradients = tape.gradient(loss, model.trainable_variables)
  optimizer.apply_gradients(zip(gradients, model.trainable_variables))
loss_object = tf.keras.losses.CategoricalCrossentropy(from_logits=True)

def get_simple_model(NUM_CLASSES = None):
  model = Sequential()

  model.add(Flatten())
  model.add(Dense(100, activation="relu"))
  model.add(Dense(NUM_CLASSES, activation="softmax"))

  model.compile(optimizer = "Adam",loss='categorical_crossentropy',
metrics=['accuracy'])

  return model
print("=== simple ===")
tf2_model = get_simple_model(NUM_CLASSES = 10)


print("=== complex ===")


input_shape = (32,32,3)
```

```
(x_train, y_train), (x_test, y_test), min_pixel_value, max_pixel_value =
get_dataset(dataset_name = "fashion_MNIST")

tf2_model.fit(x_train, y_train, epochs=12,verbose=1,batch_size=32)

tf2_classifier = TensorFlowV2Classifier(
      model=tf2_model,
      loss_object=loss_object,
      train_step=train_step,
      nb_classes=10,
      input_shape=(input_shape),
      clip_values=(0, 1),
    )



attack = FastGradientMethod(tf2_classifier, eps=0.1,minimal = True)
print("generating attacks...")
x_test_adv = attack.generate(x=x_test)
white_box_adv_predictions = tf2_classifier.predict(x_test_adv)
white_box_adv_accuracy = np.sum(np.argmax(white_box_adv_predictions,
axis=1) == np.argmax(y_test, axis=1)) / len(y_test)
print("[FGSM] white_box_adv_accuracy: ",white_box_adv_accuracy)
```

Output:



```
=== complex ===
in get_dataset tf2_attack, dataset_name:  fashion_MNIST
2021-12-01 22:59:21.563426: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/12
1875/1875 [==============================] - 2s 634us/step - loss: 0.4988 - accuracy: 0.8271
Epoch 2/12
1875/1875 [==============================] - 1s 503us/step - loss: 0.3750 - accuracy: 0.8642
Epoch 3/12
1875/1875 [==============================] - 1s 501us/step - loss: 0.3382 - accuracy: 0.8773
Epoch 4/12
1875/1875 [==============================] - 1s 500us/step - loss: 0.3154 - accuracy: 0.8845
Epoch 5/12
1875/1875 [==============================] - 1s 501us/step - loss: 0.2984 - accuracy: 0.8902
Epoch 6/12
1875/1875 [==============================] - 1s 500us/step - loss: 0.2830 - accuracy: 0.8955
Epoch 7/12
1875/1875 [==============================] - 1s 501us/step - loss: 0.2739 - accuracy: 0.8988
Epoch 8/12
1875/1875 [==============================] - 1s 501us/step - loss: 0.2634 - accuracy: 0.9026
Epoch 9/12
1875/1875 [==============================] - 1s 501us/step - loss: 0.2537 - accuracy: 0.9055
Epoch 10/12
1875/1875 [==============================] - 1s 501us/step - loss: 0.2446 - accuracy: 0.9087
Epoch 11/12
1875/1875 [==============================] - 1s 501us/step - loss: 0.2381 - accuracy: 0.9111
Epoch 12/12
1875/1875 [==============================] - 1s 501us/step - loss: 0.2301 - accuracy: 0.9133
generating attacks...
/home/samridhi/adversarial-robustness-toolbox/samwork/lib/python3.8/site-packages/keras/backend.py:4846: UserWarning: "`categorical_crossentropy` received `from_logits=True`, but the `output` argument was
 produced by a sigmoid or softmax activation and thus does not represent logits. Was this intended?"
  warnings.warn(
[FGSM] white_box_adv_accuracy:  0.1385
```

From the above result, accuracy from before the attack in 12 folds are nearly 90% and after the attack it decreases to 13%.

Industrial Interviews

Interview with (Phaidra.ai)

Background

He is co-founder of Phaidra.ai which creates AI-powered control systems to improve energy efficiency and sustainability, increase plant safety and stability , maximize product quality and yield in industries. His expertise in distributed computing worked in a Google-deep mind and the team reduced Google's data center cooling energy consumption by 40%. The organization works in optimizing the KPI's for potential industries by making intelligent infrastructure.

ML Infrastructure

Veda's team built its own ML infrastructure and stores in Google Cloud Platform. Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, Google Drive, and YouTube. Major training infrastructure and services are secured under GCP. Phaidra.ai trusts cloud infrastructure for their built-in protection, and global network that google uses to protect your information, identities, applications, and devices. The security has been achieved by encrypting data in transit between services and at rest, ensuring that it can only be accessed by authorized roles and services with audited access to the encryption keys. The team also uses open source ML toolkits like Pytorch libraries in their projects and trust their security behavior.

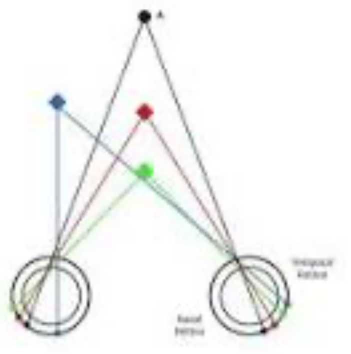
We mostly believe that adversarial attacks can happen in an environment where user interactions take place like random input in Tesla car's can lead to serious accidents which are life threatening. It depends how you feed the input onto your ML systems. Veda has shared some of the vulnerable attacks to their system. But he also says that image based systems and texts based system where user interactions happen are more prone.

- Man-in-the-middle attack: A man-in-the-middle attack is a type of eavesdropping attack, where attackers interrupt an existing conversation or data transfer. After inserting themselves in the "middle" of the transfer, the attackers pretend to be both legitimate participants. The primary defense is encrypting the traffic between the network and device using browsing encryption software, it can help fend off potential man in the middle attacks. Always make sure the sites visiting are secure. Most browsers show a lock symbol next to the URL when a website is secure.
- Free form user input on 'App Store'
- Access to data in 'Downtime' of iron boxes (GCP)

Interview with (https://driveghost.com/)

Background

The ghost team focus is making the car behaves like autonomous vehicle on a highway. Ghost is designed to detect any incoming threat and react five times faster than a person, tracking the motion of every obstacle – regardless of shape, size, or type – to keep humans protected from the unexpected. It mainly trains the model on the 'perception' of the object i.e. is perception of stable three-dimensional entities. They use stereo camera for perceiving the object.



A stereo camera is a type of camera with two or more image sensors. This allows the camera to simulate human binocular vision and therefore gives it the ability to perceive depth.

ML Infrastructure

They build their ML model from Scratch using Hadoop and spark.

- Apache Spark provides a very powerful API for ML applications. Its goal is to make practical ML easy. It has lower-level optimisation primitives and higher-level pipeline APIs. It is largely used for predictive analytics solutions, recommendation engines and fraud detection systems being the most popular ones.
- The goal of Apache Mahout is to provide scalable libraries that enables running various machine learning algorithms on Hadoop in a distributed manner.

Libraries built in the organization is inspired by 'Pytorch' open-source framework. Pytorch one of the widely used Machine learning library. He mentions the reason behind building their own tool and not the open source library are their maintenance and documentation.

Overview on security of the data, ghost maintains its own data centers which has GPU's for big data processing and stores in the form of key/value pair using Apache Hadoop and Apache Accumulo. Key-value pair in MapReduce is the record entity that Hadoop

MapReduce accepts for execution. Apache Accumolo is a sorted, distributed key/value store that provides robust, scalable data storage and retrieval.

The security of data is obtained by firewall authentication at every checkpoint. No-public cloud enhances data security and privacy.

In an autonomous driving system, cameras and LiDAR's are deployed to collect information about the driving scene, which is then fed into a CNN-based driving model to make decisions such as adjusting the speed and steering angle. Unfortunately, CNNs can be easily fooled by adversarial examples, which are constructed by applying small, pixel-level perturbations to input images.[7]For example, consider the message replaying attack by a malicious vehicle that is attempting to jam traffic.

He says that '**Model stealing'** can be a vulnerable attack for the ML infrastructure. This can cause issues when either the training data or the model itself is sensitive and confidential.

He mentions that 'Coverage of data' is an important gap during the development/deployment of the ML model. As in developing stage, the training set for autonomous vehicle is collecting the data for a sub-human. It is a constant learning because car is always on the road.

An adversary attack can also happen when data gets biased due to un-known learning of the model that can make the model bias towards the majority class. It is really important to maintain the uniformly distributed data across.

In autonomous vehicle, the system is mostly based on image recognition system storing the training data from the camera recording and training the model. The privacy of human is high priority and making sure no anonymous data is stored.

According to him, financial industry is the one which needs the security from adversary attack the most which fails in recognizing real/fake identification. Stalk market and trading attacks can compromise the data and results in a negative impact for the organization.

Interview with (https://www.zillow.com/)

Background

Zillow is an American online real estate marketplace company which helps people buys their houses and rent their houses. The customer journey starts with an interested customer visiting the website looking to buy/rent a house. It Is greeted by a chatbot which uses natural language processing to understand the intent of the customer. The interviewee shared that they are using AWS cloud for managed service of predicting customer intent. They are also using 'Pytorch' library for ML models. Once a customer starts interacting with the bot, the model tries to share recommendations for houses based on what houses have been liked by the customer. The team uses 3D images of the house liked by the customer to predict what kind of houses on the inventory of the website might be interesting for the customer. Once the customer creates an account in the Zillow website, they are followed with personalized emails recommending what other houses they might be interested in buying/renting. The customer analytics team also feeds in customer signals in the model to conduct look-alike modeling on who are the other customer segments who will be interested in the same house that the current user has selected.

<u>ML Infrastructure</u>

In the case of adversarial attack on the ML models, the interviewee shared that a possible attack can be for an agent to start querying the chatbot with arbitrary intent values. For example, someone starts to create multiple agents based out of San Francisco with a similar search intent for houses in New York. This is known as “mining labels” which can lead to the system learning that people in San Francisco are interested in houses in New York. The interviewee was able to recommend various ways to protect against such an attack by training the model to detect anomalies and uniform distribution of the training datasets. The model stealing attack was not considered relevant by the interviewee. He shared that model training process is running on secure cloud services and the input and feedback is controlled internally. Hence there is no chance of model stealing and considered the practical implementation of model stealing difficult for secure distributed systems in AWS cloud.

The customer responded to my inquiry about if the chatbots on the website are susceptible for poisoning attack to influence model ‘misbehavior’. He said that the case of “Tay” developed by Microsoft was based on active learning while industrial chatbots are based on predictive analytics using ground truth datasets and hence cannot be modified using consumer inputs. He also mentioned that “image-based learning” has higher chance of poisoning attacks as the data is unstructured and can be easily manipulated while NLP is lesser prone to such attack. Lastly, the interviewer mentioned the reward of manipulation puts Financial Services, Transportation as high risk for cyber-attack while consumer brands are lesser exposed to it given no real reward for the agents to conduct the attacks.

Interview with (https://www.applovin.com/)

<u>Background</u>

Applovin Corporation, doing business as AppLovin, is a mobile technology company headquartered in Palo Alto, California.[4] Founded in 2012, it operated in stealth mode until 2014.[5] AppLovin enables developers of all sizes to market, monetize, analyze and publish their apps through its mobile advertising, marketing, and analytics platforms MAX, AppDiscovery, and SparkLabs. AppLovin operates Lion Studios, which works with game developers to promote and publish their mobile games. AppLovin also has large investments in various mobile game publishers. In 2020, 49% of AppLovin's revenue came from businesses using its software and 51% from consumers making in-app purchases.

In a mobile ad-network, we have the supply side and demand side. The demand side are brands, advertisers, and agencies who want to buy ad-inventory on the various gaming mobile applications. As soon as a buyer on the demand side sends a bid to an ad-exchange, ad-exchange starts to look at the supply side in near-real time. The supply side consists of publishers such as Pokemon Go, Minecraft who host ads on their web and mobile interfaces (pixels). The role of an ad-exchange is to match the supply/inventory with the highest bidder to maximize the ad-revenue for the publisher. For example, a Nike (advertiser) planning to run an advertisement about "running shoes and Nike+" on Pokemon Go app (publisher) requires Google Ads (ad-exchange) to facilitate the transaction. Google has information on the end-customer using the login details and cookie-based tracking to find they might be interested in Nike shoes. Appslovin provides the technology to the mobile apps builders to host their supply/inventory on ad-networks to monetize it.
In the advertising industry, the modeling is conduced only on the most recent data (1-2 weeks) and they train the model 4-5 times a data on billions of transactions/bids happening on the platform. The high cardinality of the data enables the model to dynamically adapt to changing user behaviors and advertising needs. The model recommends who among the audiences of users using the mobile applications should be targeted for a successful bid. For example, the model helps to predict the Nike advisement should be shown which customer ID depending on the most recent visit to a retail website to purchase shoes.

ML Infrastructure

The interviewer shared that they are using open-source libraries such as TensorFlow to find audience creation and look-alike modeling. They use Python to train the models. The ML infrastructure is optimized for minimum latency to enable fast bid requests, so they use data steaming service such as Kafka. They migrated from using their own data centers to Google Cloud Platform due to cost effectiveness, scalability and flexibility for the ad-network. The most vulnerable moment for the system for attacks is when the number of bid requests increases, and system might be close to a DDOS (distributed denial of service). The agents can create surge of requests to exploit the vulnerability in the system impacting the customer experience. The interviewee did not share any specific poisoning attack scenarios except that the model can blocked from receiving input data by anonymizing a user and session. For example, an advanced gamer starts to use a private VPC to route the IP address to an unknown region or convert the session attributes into an anonymous user, so the model does not get enough attributes to target the customer. This will reduce the ability to target the customer and we advertisers might not target them with relevant ads leading model. The interviewee mentioned that they are using mechanism for anomaly detection, web traffic management dashboard to monitor the usage metrics, and performance system.

In an interesting discussion, the interviewee shared that by using out of the box ML models built by cloud providers, they are passing parameters into APIs to get predictions as a result. Hence, they are not concerned about the attacks such as data positioning, model stealing, and model inversion as they are using a managed service.

Table 1: Attacks Vs Industry showcasing probability of various attacks per industry segment

Attacks	Financial Services	Transportation	Retail	Mobile Technology
Poisoning	High	Medium	High	Low
Model Stealing	High	Low	Low	Low
Model Inversion	Medium	Medium	-	-
Backdoor ML	Medium	Medium	-	-
Physical Domain	-	High	-	-
Software Dependency Exploitation	Medium	Medium	Medium	High

Legends:

- Poisoning: Attacker contaminates the training phase of ML systems to get intended result
- Model Stealing: Attacker is able to recover the model through carefully-crafted queries
- Model Inversion: Attacker recovers the secret features used in the model by through careful queries
- Backdoor ML: Malicious ML provider backdoors algorithm to activate with a specific trigger
- Physical Domain: Attacker brings adversarial examples into physical domain to subvertML system e.g: 3d printing special eyewear to fool facial recognition system
- Software Dependency Exploitation: Attacker uses traditional software exploits like buffer overflow to confuse/control ML systems

*Red – High, Yellow – Medium, Green - Low
**Based on the scores provided by interviewees for the ranking the attacks which they are most vulnerable to using qualitative measurement

Table 2: Infrastructure Vs Industry showcasing key ML infrastructure decision based on industry segment

Infrastructure	Financial Services	Transportation	Retail	Mobile Technology
ML Libraries	Proprietary	Proprietary	Open source	Open source
Data Pipelines TPS (transaction per second)	100	50	30	80
Data Storage	Hybrid Cloud and On-prem	On-prem	Cloud	Cloud
Training Frequency	8 times a day	Daily	Every 2 days	4 times a day
Data Size and Type	High (MBs) and Structured	Very High (TBs) and Unstructured	Medium (KBs) and Structured	Low (KBs) and Structured
Security Infrastructure	High	Medium	Low	Low

ML Libraries:

- Data Pipelines TPS (transaction per second): Percentage of transactional data in the domains.
- Data Storage: Type of data storage used
- Training Frequency: Frequency of the model to be trained
- Data Size and Type: Organizational data size and its type
- Security Infrastructure: security if the infrastructure taken care of

**Based on the responses provided by the interviewees about ML Infrasructure

Table 3: Attacks Vs ML Learning Categories

Attacks	Text based learning	Audio based learning	Image based learning
Poisoning	Medium	High	High
Model Stealing	Medium	Medium	Medium
Model Inversion	Low	Medium	High
Backdoor ML	Low	Low	Medium
Physical Domain	-	-	Medium
Software Dependency Exploitation	Low	Low	Medium

**Based on the responses provided by the interviewees about how different learnings are susceptive to attacks

References

[1] Charles Kapelke, "Adversarial Machine Learning. A Brief Introduction for Non-Technical… _ by Charles Kapelke _ CLTC Bulletin _ Medium," *Charles Kapelke*, 2019.

[2] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning," Apr. 2018, [Online]. Available: http://arxiv.org/abs/1804.00308

[3] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," Sep. 2016, [Online]. Available: http://arxiv.org/abs/1609.02943

[4] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the ACM Conference on Computer and Communications Security*, Oct. 2015, vol. 2015-October, pp. 1322–1333. doi: 10.1145/2810103.2813677.

[5] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic Backdoor Attacks Against Machine Learning Models," Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.03675

[6] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks against Machine Learning Models," Oct. 2016, [Online]. Available: http://arxiv.org/abs/1610.05820

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6572

[8] G. F. Elsayed, I. Goodfellow, and J. Sohl-Dickstein, “Adversarial Reprogramming of Neural Networks,” Jun. 2018, [Online]. Available: http://arxiv.org/abs/1806.11146

[9] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, “Synthesizing Robust Adversarial Examples,” Jul. 2017, [Online]. Available: http://arxiv.org/abs/1707.07397

[10] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, “A General Framework for Adversarial Examples with Objectives,” Dec. 2017, doi: 10.1145/3317611.

[11] T. Gu, B. Dolan-Gavitt, and S. Garg, “BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain,” Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.06733

[12] R. S. S. Kumar *et al.*, “Adversarial Machine Learning -- Industry Perspectives,” Feb. 2020, [Online]. Available: http://arxiv.org/abs/2002.05646