

# HOW TO HACK YOUR AI?

*Yapay Zekanı Nasıl Hacklersin ?*

CAN ÖZTAŞ



# WHOAMI



can oztas

comp.eng ankara university bsc  
comp.eng hacettepe university msc

ctf, offensive security

# MEET SÜTLAC



*sütlac*



cat

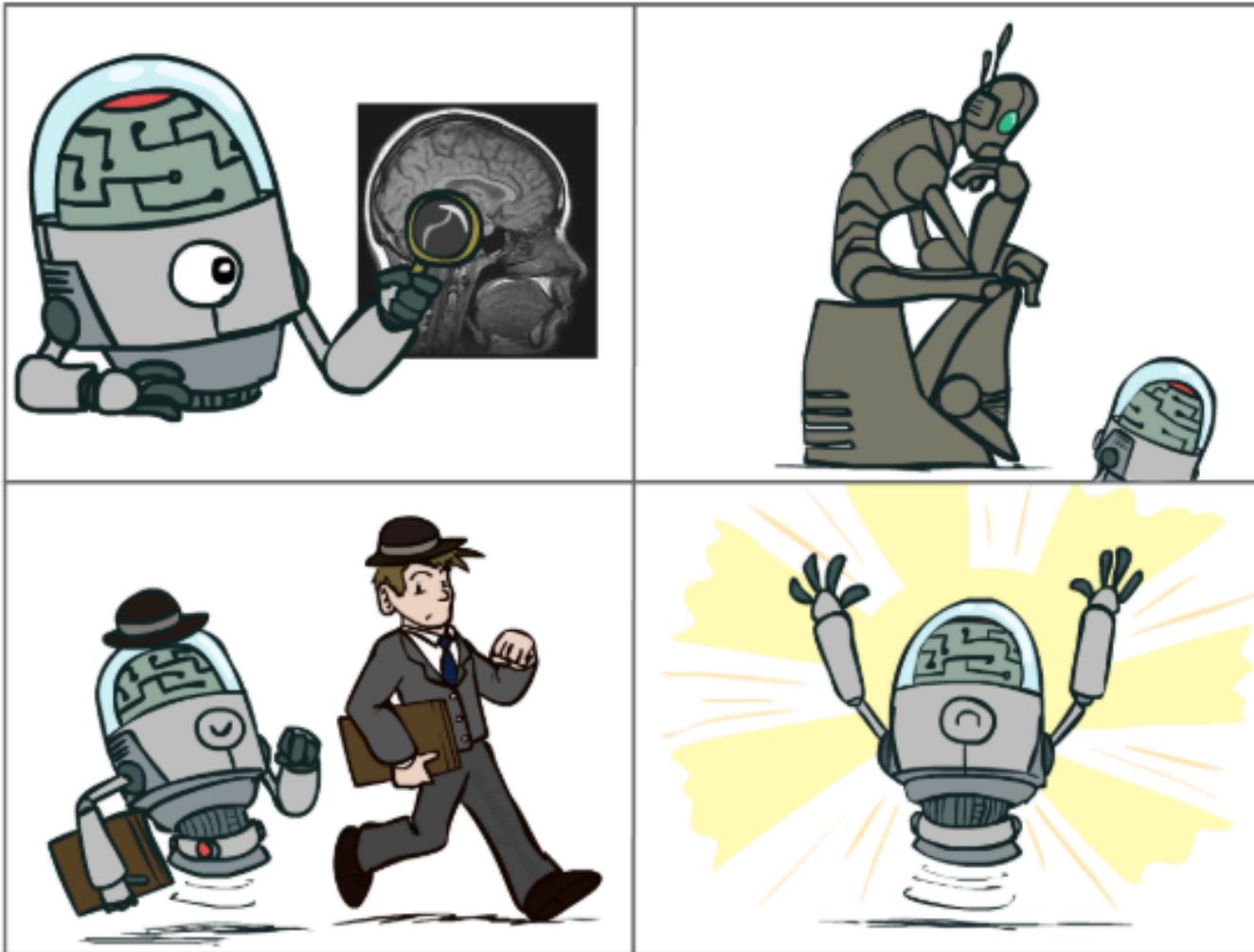


toilet tissue

## What is AI?

- Four possible definitions (textbook ch. 1):

Thinking  
humanly



Thinking  
rationally

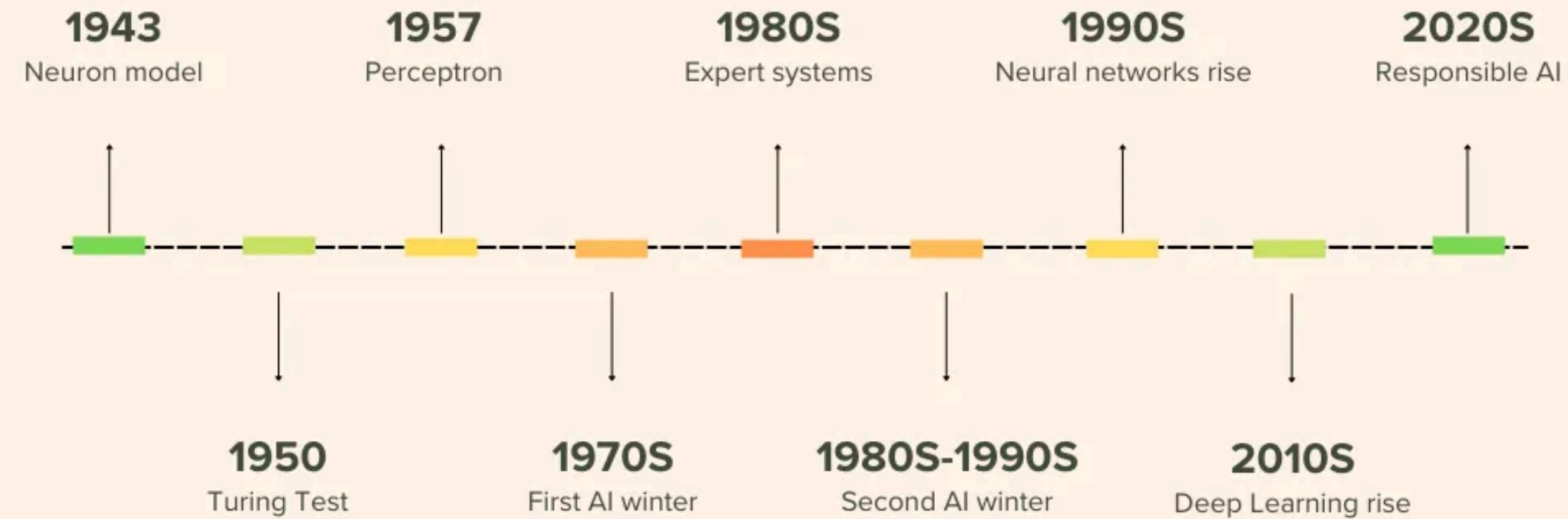
Acting  
humanly

Acting  
rationally

Source: Berkeley CS188 materials

# history ARTIFICIAL INTELLIGENCE

## AI KEY MILESTONES



@marizombie

# history ARTIFICIAL INTELLIGENCE

---

## *classification & prediction*



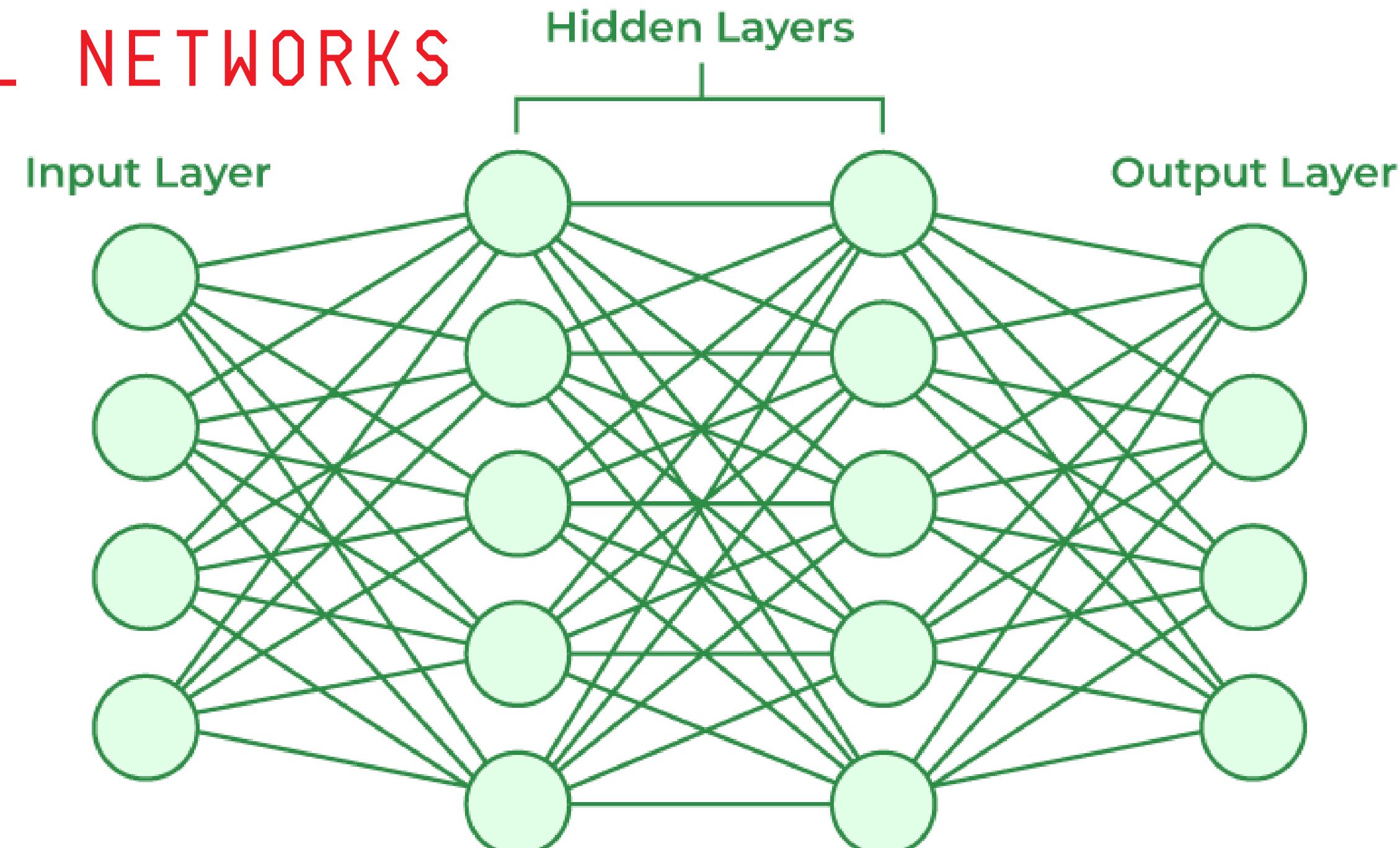
# history ARTIFICIAL INTELLIGENCE

---

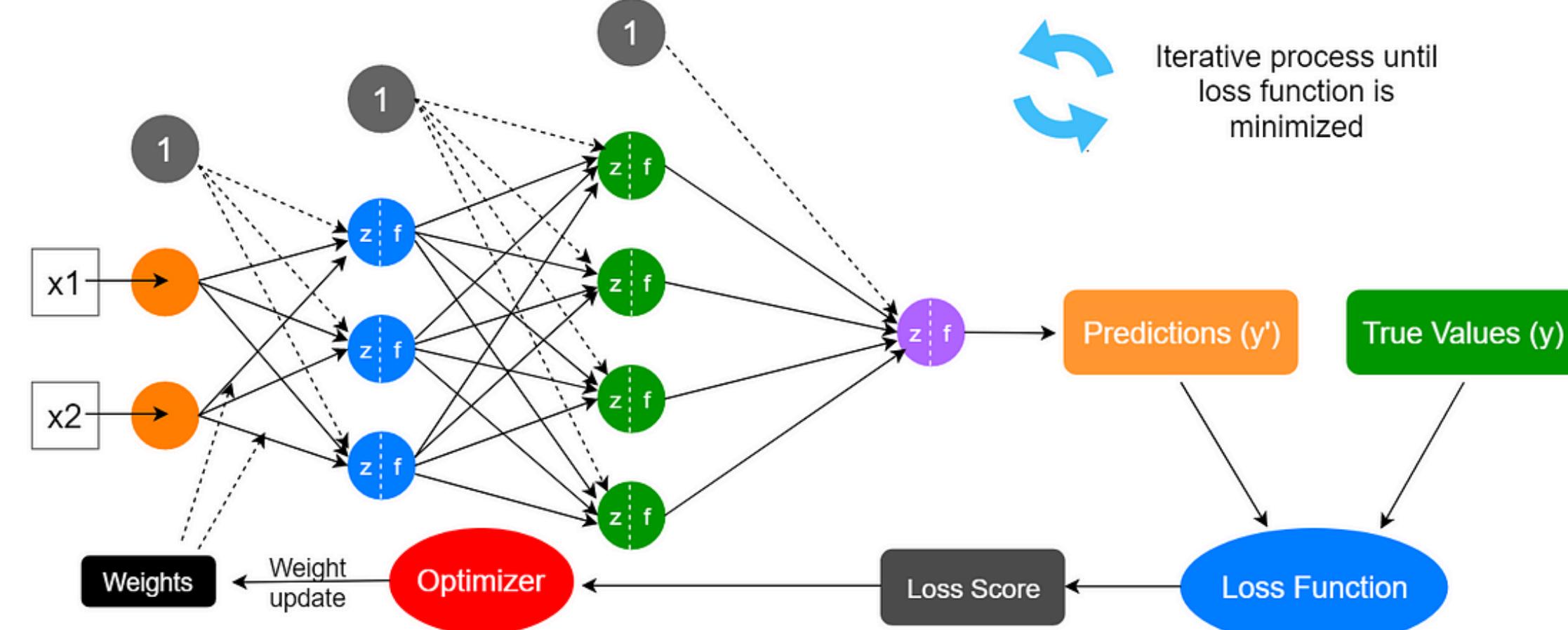
*classification & prediction*



# NEURAL NETWORKS

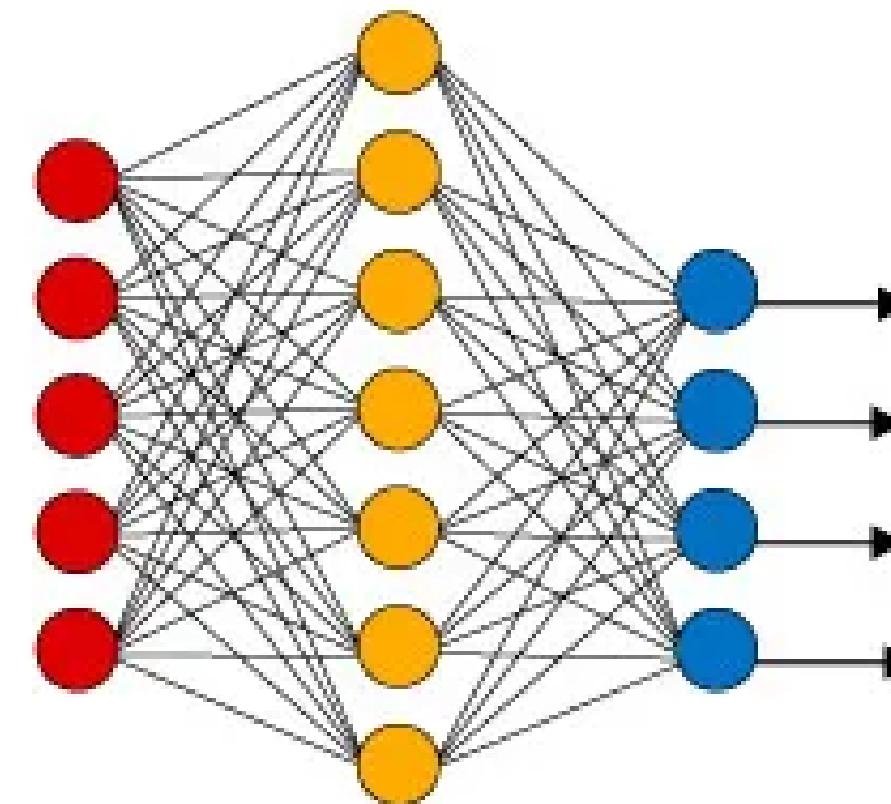


## Forward Propagation



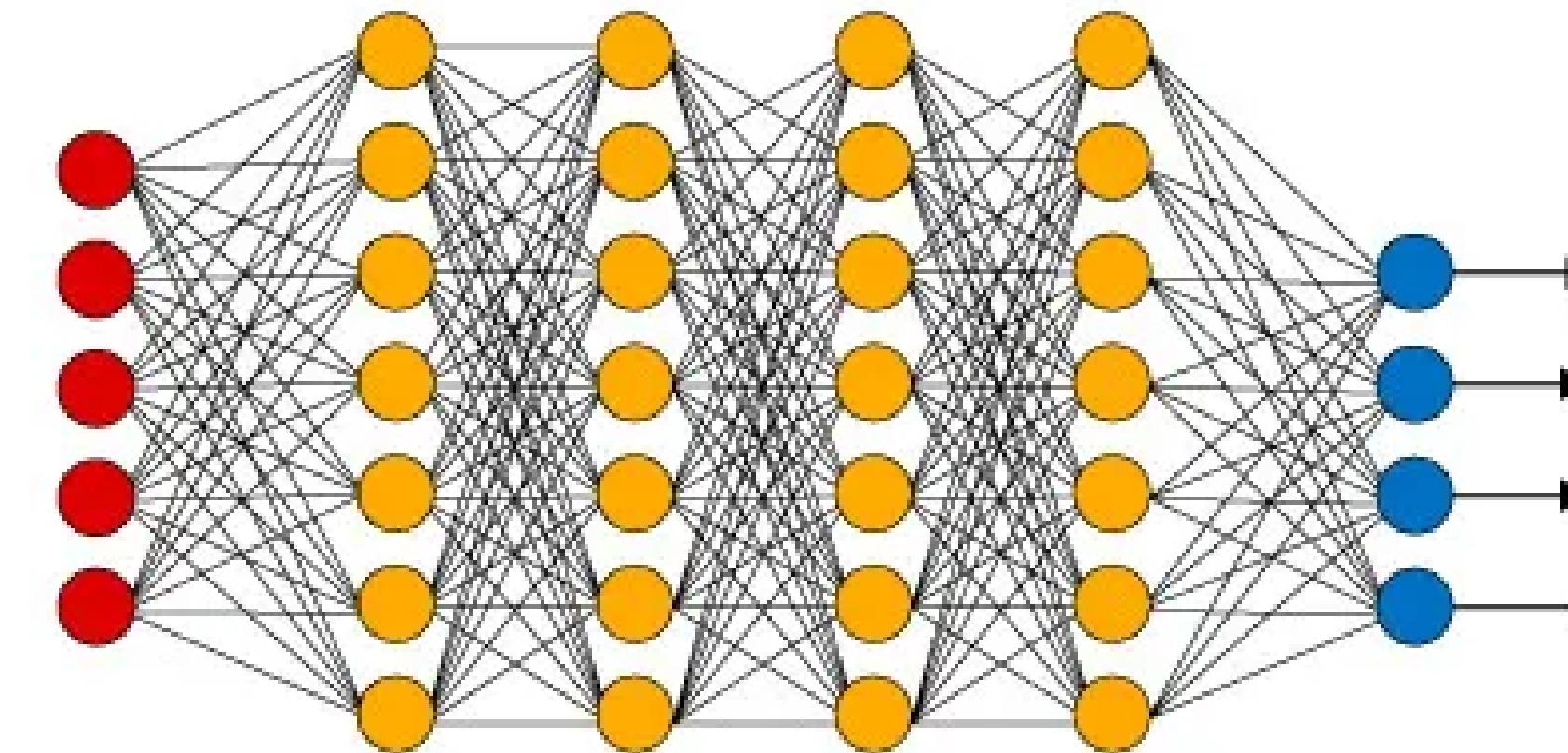
## Backward Propagation

## Simple Neural Network



● Input Layer

## Deep Learning Neural Network



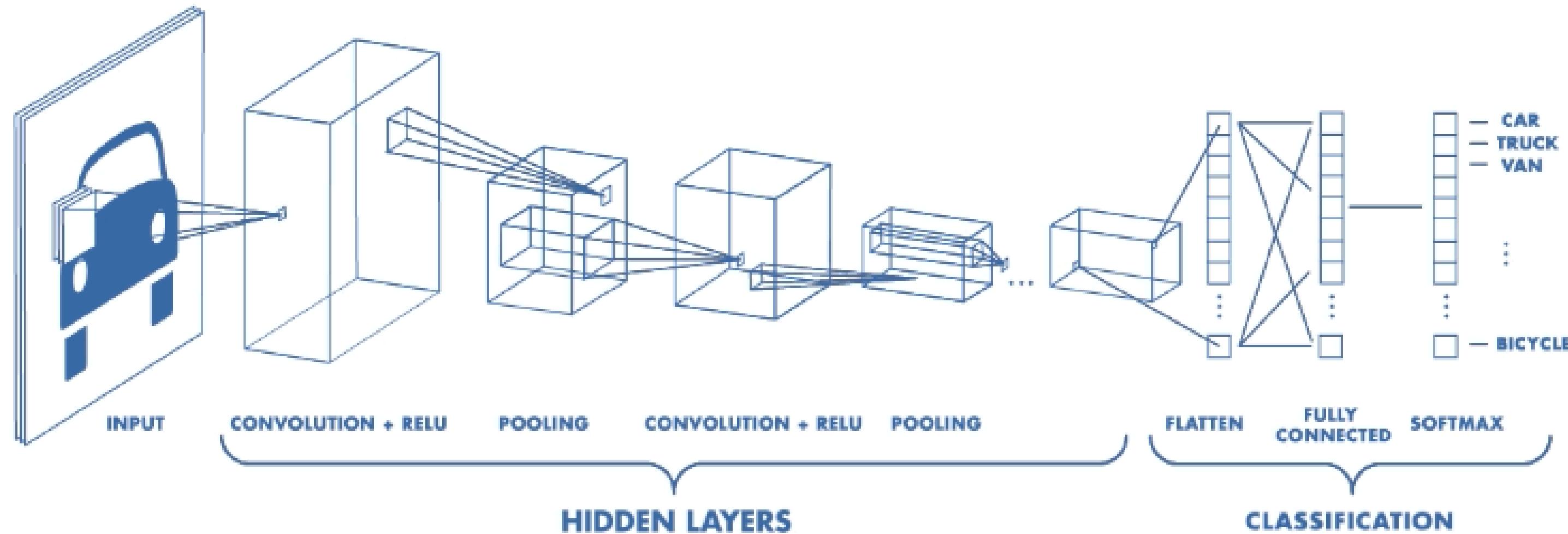
● Hidden Layer

● Output Layer

Product recommendation  
Advertisement placement  
Smart-grid energy optimization  
Household robotics  
Robotic surgery  
Robot exploration  
Spam filtering  
Fraud detection  
Fault diagnostics  
AI for video games  
Character animation  
Financial trading  
Dynamic pricing  
Protein folding  
Medical diagnosis  
Medical imaging

Web search  
Speech recognition  
Handwriting recognition  
Machine translation  
Information extraction  
Document summarization  
Question answering  
Spelling correction  
Image recognition  
3D scene reconstruction  
Human activity recognition  
Autonomous driving  
Music information retrieval  
Automatic composition  
Social network analysis

# Convolutional neural network



# *AI Security*

- Çıkan  $y'$  değerinin yanlış olması (misclassification)
- Çıkan  $y$  değerinin doğru olup, güven yüzdesinin düşmesi (confidence reduction)
- Çıkan  $y'$  değerinin saldırgan tarafından istenilen değer olması (targeted misclassification)
- Modeldeki backpropagation (geri besleme) mekanizmasına göre, modelin bozulması (corruption)

# *AI Security Cases*

Facial Recognition Manipulation

Object Detection Manipulation

- Speed Sign & Car
- Drone & Terrain
- Satelite Vision
- Car Recog

Recommendation Algorithms



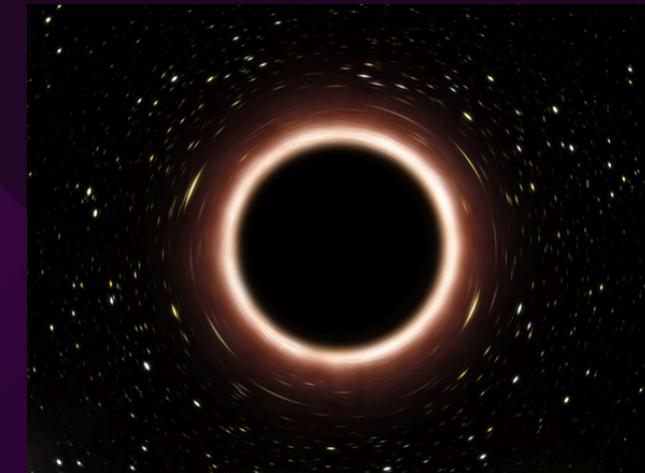
# APPROACHES

white-box

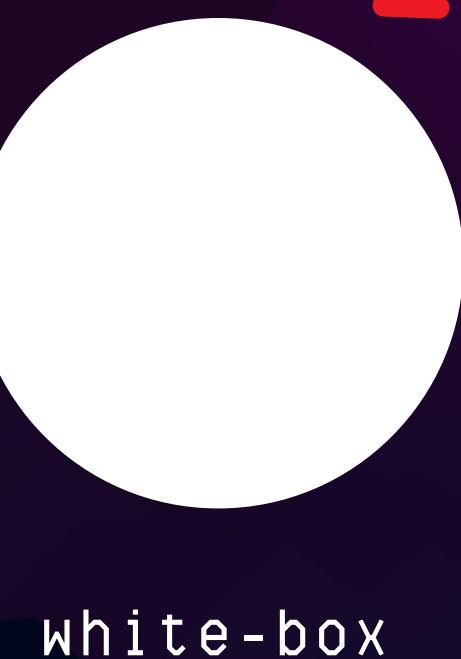
adaptive  
black-box

non  
adaptive  
black-box

Strict  
black-box



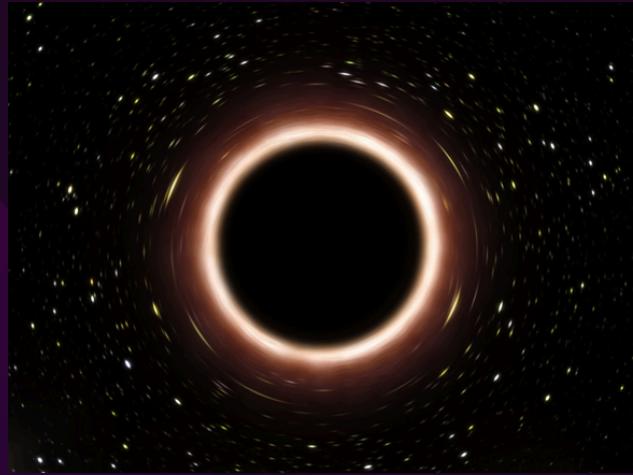
# APPROACHES



adaptive  
black-box



non  
adaptive  
black-box



Strict  
black-box

# ATTACK TYPES

## *EXPLORATORY*

- Model Inversion
- Membership Inference
- Information Inference
- Model Extraction via API

## *EVASION*

- Adversarial Examples Generation
- Generative Adversarial Networks (GAN)

## *Poisoning*

- Data Poisoning
- Logic Corruption

## OWASP Machine Learning Security Top Ten

[Main](#) | [Charter](#) | [Related](#) | [Glossary](#)

owasp incubator License CC BY-SA 4.0

### Important Information

The current version of this work is in draft and is being modified frequently. Please refer to the [project wiki](#) for information on how to contribute and project release timelines.

### Overview

Welcome to the repository for the OWASP Machine Learning Security Top 10 project! The primary aim of the OWASP Machine Learning Security Top 10 project is to deliver an overview of the top 10 security issues of machine learning systems. More information on the project scope and target audience is available in our [project working group charter](#)

### Top 10 Machine Learning Security Risks

- ML01:2023 Input Manipulation Attack
- ML02:2023 Data Poisoning Attack
- ML03:2023 Model Inversion Attack
- ML04:2023 Membership Inference Attack
- ML05:2023 Model Theft
- ML06:2023 AI Supply Chain Attacks
- ML07:2023 Transfer Learning Attack
- ML08:2023 Model Skewing
- ML09:2023 Output Integrity Attack
- ML10:2023 Model Poisoning

### Communication

- Google Groups
- OWASP Slack #project-mlsec-top-10
- Github Discussions
- Meeting Notes

### Contribution

The initial version of the Machine Learning Security Top 10 list was contributed by [Sagar Bhure](#) and [Shain Singh](#). The project encourages community contribution and aims to produce a high quality deliverable reviewed by industry peers.

All contributors will need to adhere to the project's [code of conduct](#). Please use the [following form](#) for any feedback, suggestions, issues or questions.

### Getting Started

The project has a [wiki](#) which provides information to get help you started on how to contribute.

### Licensing

The OWASP Machine Learning Security Project is licensed under the [Creative Commons Attribution-ShareAlike 4.0 license](#) so you can copy, distribute and transmit the work, and you can adapt it, and use it commercially, but all provided that you attribute the work and if you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

Watch 18 Star 58

The OWASP® Foundation works to improve the security of software through its community-led open source software projects, hundreds of chapters worldwide, tens of thousands of members, and by hosting local and global conferences.

### Machine Learning Security Top 10

This project's content is currently in draft release

Website: <https://mitop10.info>

Edition: 2023

Version: 0.3 Draft [Source Code](#) ([Download PDF](#))

Release Notes: [Changelog](#)

### Project Information

Incubator Project

Classification

Documentation

Audience

Builder

Defender

Breaker

### Leaders

Shain Singh

Sagar Bhure

Rob van der Veer

Abraham Kang

### Core Team

[Mikołaj Kowalczyk]

[Aryan Kenchappagol]

[Priyadarshini Parthasarathy]

[Ben Kereopa-Yorke]

### Project Team

Thanks go to this wonderful list of [contributors](#)

### Upcoming OWASP Global Events

OWASP Global AppSec Lisbon 2024

June 24-28, 2024

OWASP Global AppSec San Francisco 2024

ML01:2023 Input Manipulation Attack

ML02:2023 Data Poisoning Attack

ML03:2023 Model Inversion Attack

ML04:2023 Membership Inference Attack

ML05:2023 Model Theft

ML06:2023 AI Supply Chain Attacks

ML07:2023 Transfer Learning Attack

ML08:2023 Model Skewing

ML09:2023 Output Integrity Attack

ML10:2023 Model Poisoning



# ADVERSARIAL ATTACK: FAST GRADIENT SIGN METHOD

Temel Fikir: Bir makine öğrenimi modelinin kayıp fonksiyonunun gradyanını hesaplayarak, bu gradyanın işaretine göre girdi verilerini küçük bir epsilon ( $\epsilon$ ) değeri ile bozarak adversarial örnekler oluşturur.

İşlem Adımları:

Gradyan Hesaplama: İlgili girdi için modelin kayıp fonksiyonunun gradyanı hesaplanır. Bu gradyan, modelin çıktısını en çok etkileyen girdi özelliklerini belirler.

İşaret Fonksiyonu: Hesaplanan gradyanın işaret vektörü alınır. Bu işaret vektörü, kaybı en fazla artıracak yönde bir adımın yönünü gösterir.

Adversarial Girdi Oluşturma: Orijinal girdi, işaret vektörü ile belirlenen yönde  $\epsilon$  büyüklüğünde bozularak adversarial girdi elde edilir. Bu bozulma, modelin yanlış sınıflandırma yapmasına neden olacak şekilde tasarlanmıştır.

Bu vektörün iteratif ve momentum-iteratif varyasyonları vardır.

$$x^{adv} = x + \epsilon \cdot sign(\nabla_x J(x, y_{true}))$$

where

- $x$  — Clean Input Image
- $x^{adv}$  — Adversarial Image
- $J$  — Loss Function
- $y_{true}$  — Model Output for  $x$
- $\epsilon$  — Tunable Parameter

A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533, 2016.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014b.





# ADVERSARIAL ATTACK: FAST GRADIENT SIGN METHOD

Fast Gradient Sign Method

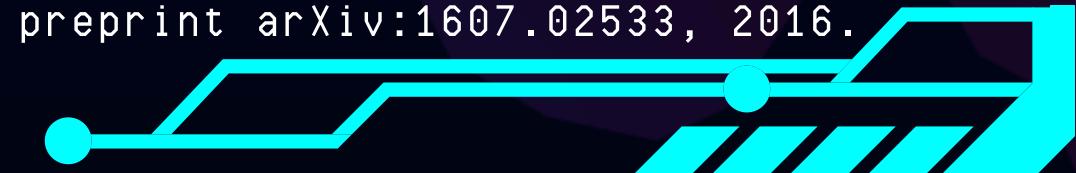
$$\mathbf{X}^{adv} = \mathbf{X} + \epsilon \text{sign}(\nabla_{\mathbf{X}} J(\mathbf{X}, y_{true}))$$

Score of label  $y_{true}$ , given input image  $\mathbf{X}$



Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014b.

A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533, 2016.



# GAN BASED ATTACK : AN EXAMPLE

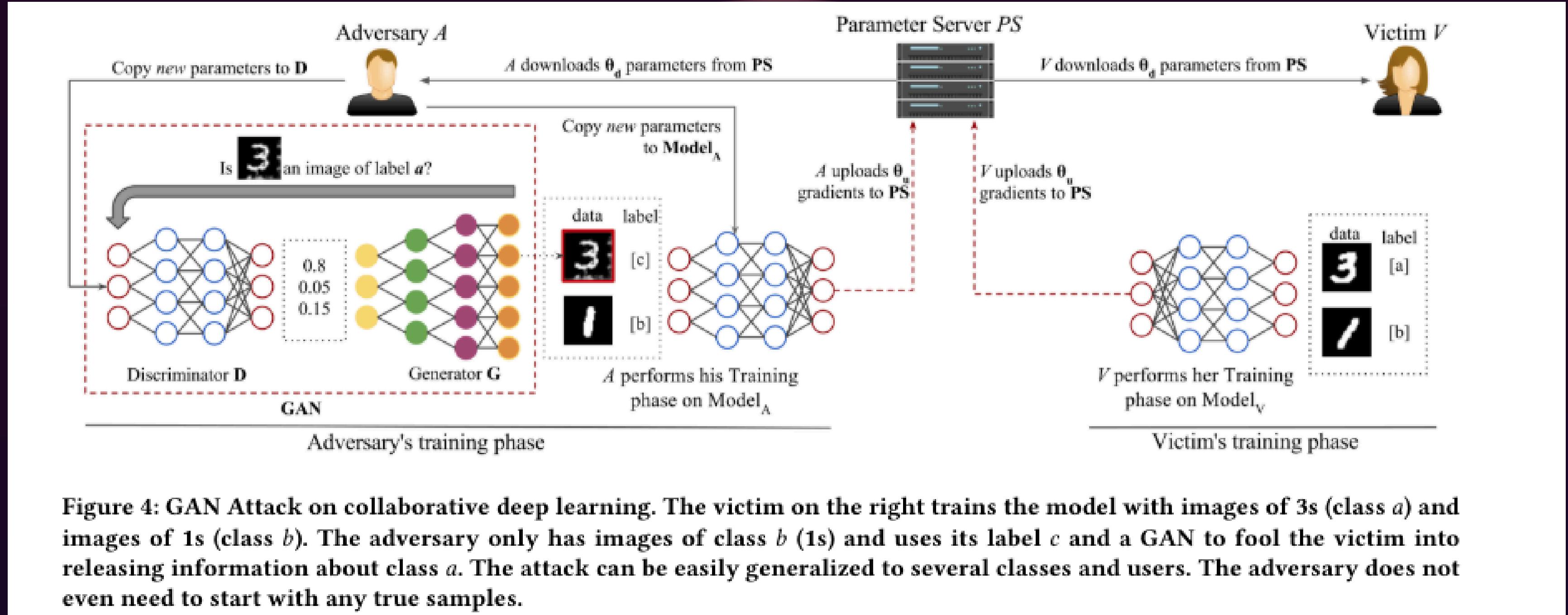


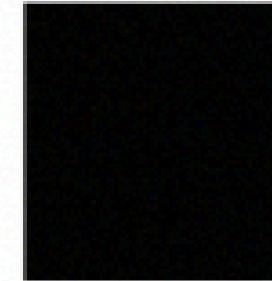
Figure 4: **GAN Attack on collaborative deep learning.** The victim on the right trains the model with images of 3s (class  $a$ ) and images of 1s (class  $b$ ). The adversary only has images of class  $b$  (1s) and uses its label  $c$  and a GAN to fool the victim into releasing information about class  $a$ . The attack can be easily generalized to several classes and users. The adversary does not even need to start with any true samples.

# ADVERSARIAL ATTACK: FAST GRADIENT SIGN METHOD

Original Label for googlenet: 235 -> German shepherd



Epsilon: 0.05



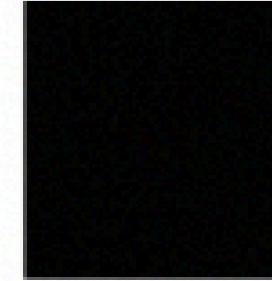
Adversarial Label: 225 -> malinois



Original Label for resnet152: 235 -> German shepherd



Epsilon: 0.05



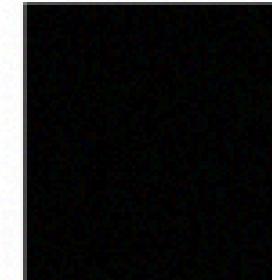
Adversarial Label: 225 -> malinois



Original Label for alexnet: 225 -> malinois



Epsilon: 0.05



Adversarial Label: 333 -> hamster



Mean of All



Fig 2. fgsm attack, e=0.05

# ADVERSARIAL ATTACK: FAST GRADIENT SIGN METHOD

Original Label for googlenet: 235 -> German shepherd



Original Label for resnet152: 235 -> German shepherd



Original Label for alexnet: 225 -> malinois



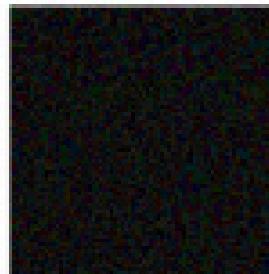
Epsilon: 0.15



Epsilon: 0.15



Epsilon: 0.15



Mean of All



Adversarial Label: 285 -> Egyptian cat



Adversarial Label: 225 -> malinois



Adversarial Label: 591 -> handkerchief



Fig 3. fgsm attack, e=0.15

# ADVERSARIAL ATTACK: FAST GRADIENT SIGN METHOD

Original Label for googlenet: 235 -> German shepherd



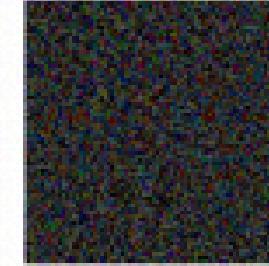
Original Label for resnet152: 235 -> German shepherd



Original Label for alexnet: 225 -> malinois



Epsilon: 0.35



Epsilon: 0.35



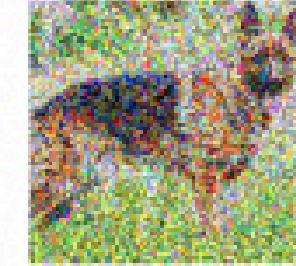
Epsilon: 0.35



Mean of All



Adversarial Label: 824 -> stole



Adversarial Label: 6 -> stingray



Adversarial Label: 549 -> envelope



Fig 4. fgsm attack, e=0.35

# ADVERSARIAL ATTACK: FAST GRADIENT SIGN METHOD

Original Label for googlenet: 235 -> German shepherd



Original Label for resnet152: 235 -> German shepherd



Original Label for alexnet: 225 -> malinois



Epsilon: 0.55



Epsilon: 0.55



Epsilon: 0.55



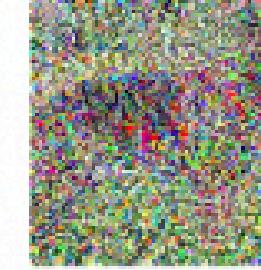
Mean of All



Adversarial Label: 824 -> stole



Adversarial Label: 443 -> bib



Adversarial Label: 109 -> brain coral



Fig 5. fgsm attack, e=0.55

# ADVERSARIAL ATTACK: FAST GRADIENT SIGN METHOD

ChatGPT 4 ▾



You



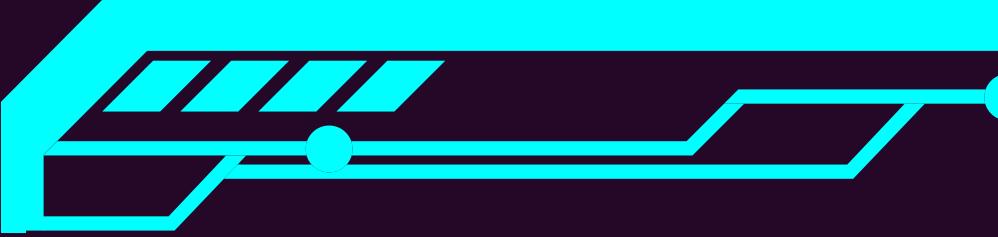
What is that, I can not understand



ChatGPT

The image you've shared is a visual representation of a "neural network dream," also known as a deep dream. This is a kind of image that's been processed by a particular type of artificial neural network to enhance and exaggerate the patterns it "sees." The process often creates psychedelic and surreal images, where familiar shapes and patterns are transformed into something quite abstract. The original subject can be hard to distinguish, but it looks like there might be a cat within the array of colors and shapes. It's an artistic way of visualizing how machines might interpret visual data.





# ADVERSARIAL ATTACK: FAST GRADIENT SIGN METHOD



You

Can you understand what that is?



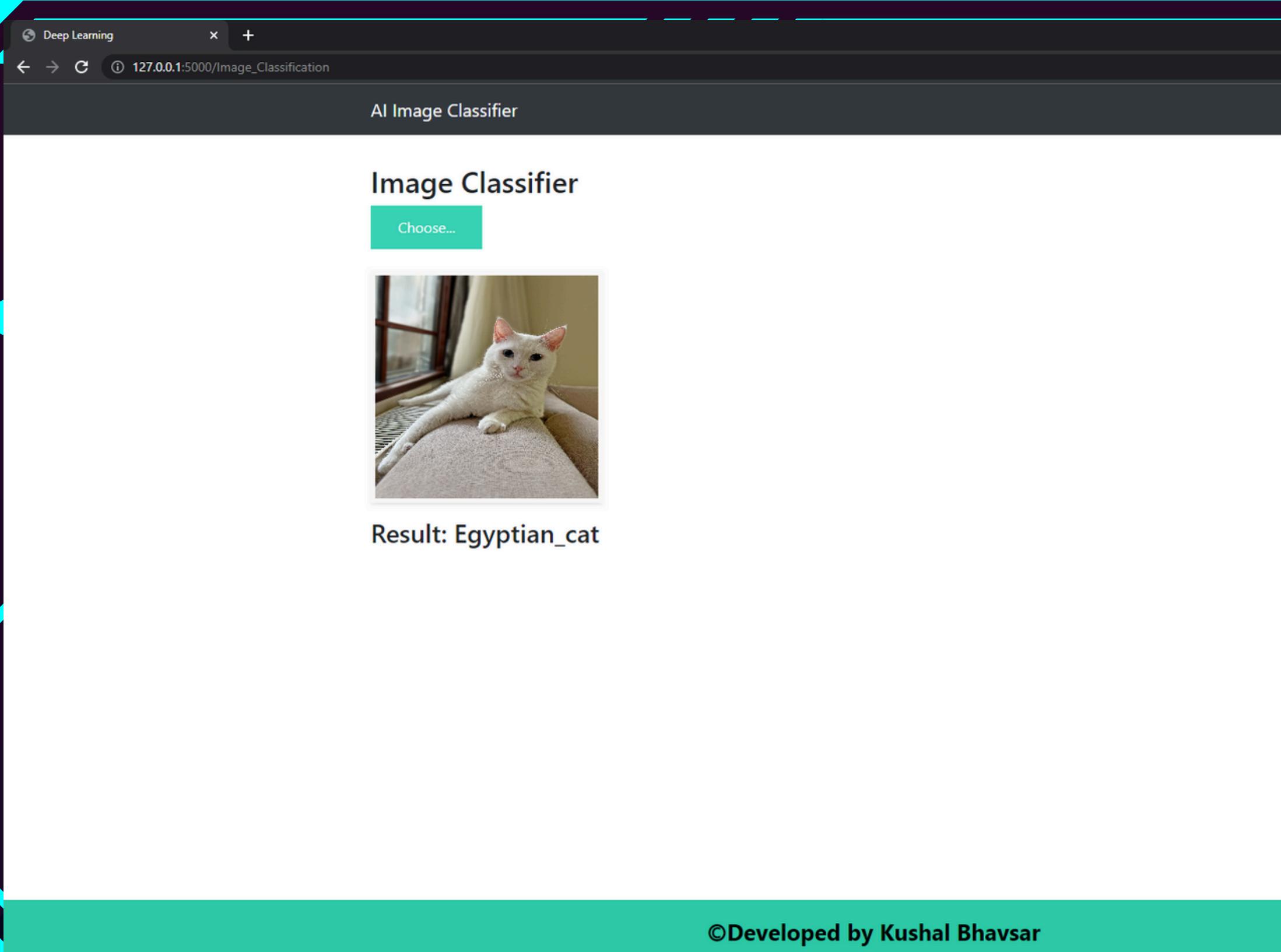
ChatGPT

Yes, this image looks like a visual output created by a process known as Deep Dream, which is a computer-generated algorithm that accentuates patterns in images via algorithmic pareidolia. This results in a dream-like, surreal appearance where the original image is overlaid with intricate patterns and shapes. It's designed to mimic how neural networks might perceive and augment visual data.

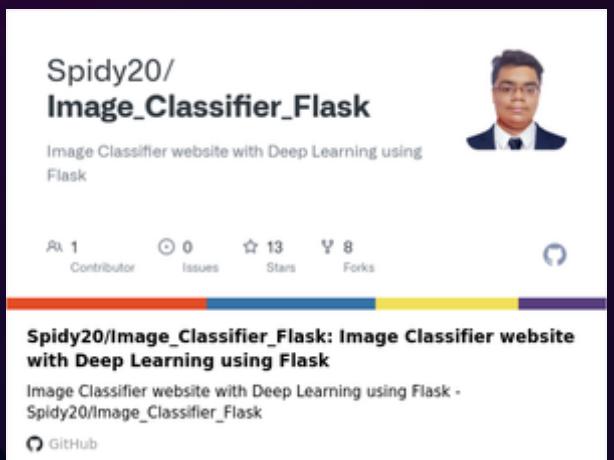
However, due to the intricate and overlapping patterns, it's quite difficult to identify the original subject with certainty. It appears to be an animal, possibly a cat, based on the outline and texture that resemble fur, ears, and a general feline shape. The process of Deep Dream often distorts the original image to the point where the subjects can become quite abstract, as seen here.

Fig 6. ChatGPT vs fgsm





# A BLACK-BOX EXAMPLE



©Developed by Kushal Bhavsar

# *AI Security*

Do I know Model ?

Do I know Dataset ?

Can I send Data ? 

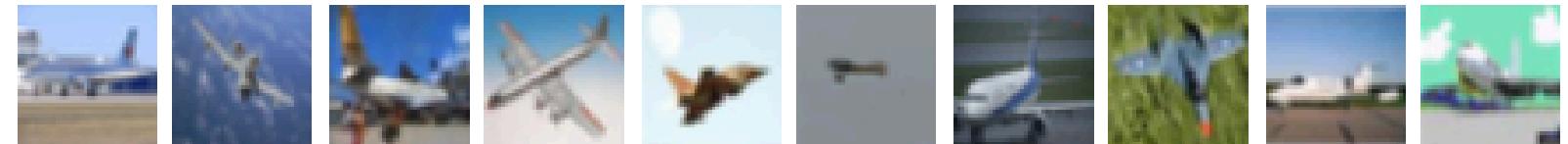
# *Plan*

Find a dataset

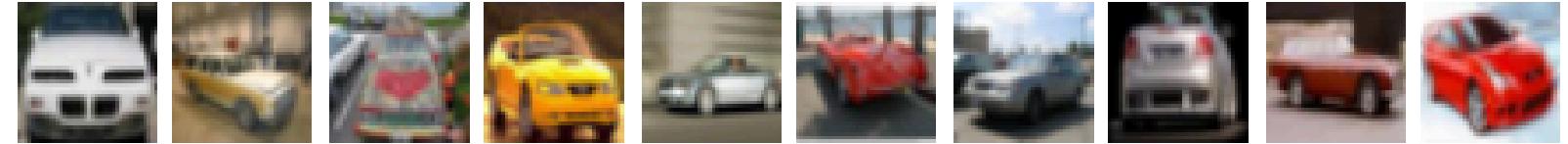
Get the classification results

Build a model based on the results

**airplane**



**automobile**



**bird**



**cat**



**deer**



**dog**



**frog**



**horse**



**ship**



**truck**



```
url = "http://127.0.0.1:5000/predict"
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
data_len = len(X_train)

print(str(data_len) + "total")
"""

print(f"X_train shape: {X_train.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_test shape: {y_test.shape}")
"""

y_pred_raw = []
y_pred = []

clean_train = []
clean_pred = []
clean_X = []

labels = ['airplane', 'automobile', 'bird', 'cat', 'deer',
          'dog', 'frog', 'horse', 'ship', 'truck']
```

```
for i in range(0,data_len):
    data = X_train[i]
    img = Image.fromarray(data, 'RGB')
    byte_io = BytesIO()
    img.save(byte_io, 'png')
    byte_io.seek(0)
    response = requests.post(url, files={'file': byte_io})
    pred = response.text
    print(pred)
    y_pred_raw.append(pred)
    try:
        y_pred.append(labels.index(pred))
        clean_Y.append(y_train.flatten()[i])
        clean_pred.append(y_pred[i])
    except:
        y_pred.append(100)
    print(str(i) + " yapilan index")

if(len(clean_pred)==0):
    print("zero size")

with open(r'output2.txt', 'w') as fp:
    fp.write("\n".join(str(item) for item in y_pred_raw))
```

```
for i in range(0,data_len):
    data = X_test[i]
    img = Image.fromarray(data, 'RGB')
    byte_io = BytesIO()
    img.save(byte_io, 'png')
    byte_io.seek(0)
    response = requests.post(url, files={'file': byte_io})
    pred = response.text
    print(pred)
    y_pred_raw.append(pred)
    try:
        y_pred.append(labels.index(pred))
        clean_Y.append(y_train.flatten()[i])
        clean_pred.append(y_pred[i])
    except:
        y_pred.append(100)
    print(str(i) + " yapilan index")

if(len(clean_pred)==0):
    print("zero size")

with open(r'output2.txt', 'w') as fp:
    fp.write("\n".join(str(item) for item in y_pred_raw))
```

## NOT SAME OUTPUT, LET'S DEFINE A RULESET

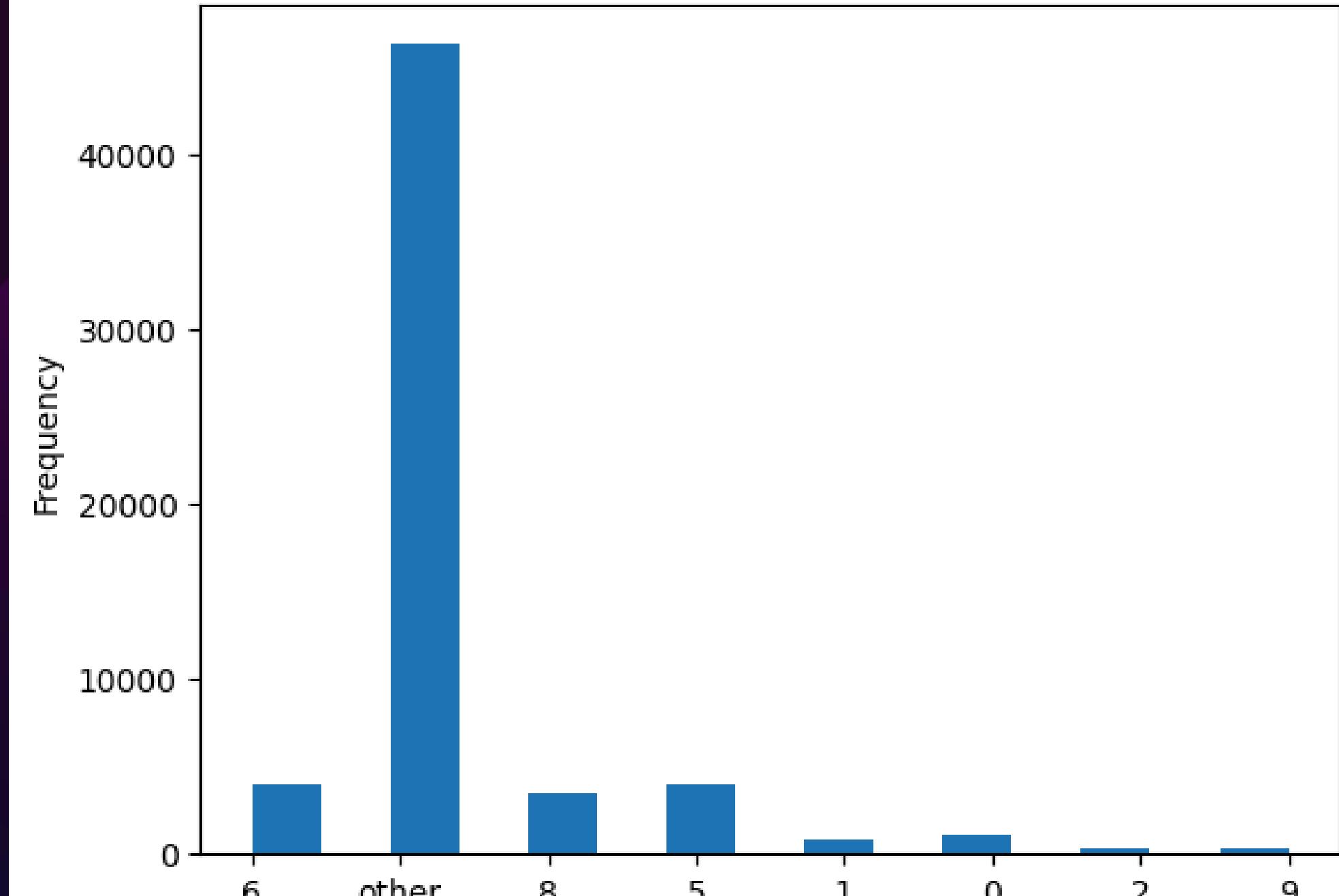


```
Airplane: airship, warplane, space_shuttle
Automobile: limousine, go-kart, cab, race_car, convertible, sports_car, ambulance, fire_engine,
garbage_truck, tow_truck, police_van
Bird: partridge, black_grouse, brambling, indigo_bunting, robin
Cat: persian_cat, siamese_cat
Deer: irrelevant
Dog: old_english_sheepdog, dandie_dinmont, maltese_dog, papillon, blenheim_spaniel, border_terrier,
australian_terrier, gordon_setter, norwich_terrier, shih-tzu, keeshond, bouvier_des_flandres, otterhound,
briard, collie, welsh_springer_spaniel, silky_terrier, pomeranian, affenpinscher, german_shepherd,
chihuahua, lakeland_terrier, bloodhound, appenzeller, yorkshire_terrier, border_collie,
scottish_deerhound, airedale, mexican_hairless, english_springer, dhole, borzoi, entlebucher,
wire-haired_fox_terrier, rhodesian_ridgeback, rottweiler, scotch_terrier, meerkat, beagle,
bernese_mountain_dog, pug, kerry_blue_terrier, norfolk_terrier, kuvasz, basset, dalmatian, siberian_husky,
cocker_spaniel, basenji, timber_wolf, labrador_retriever, golden_retriever, boxer, english_foxhound,
great_dane, clumber, vizsla, irish_terrier, chesapeake_bay_retriever, standard_schnauzer, bull_terrier,
west_highland_white_terrier, toy_poodle, miniature_pinscher, american_staffordshire_terrier,
irish_water_spaniel, eskimo_dog
Frog: common_newt, tailed_frog
Horse: irrelevant (no specific horse-related terms provided)
Ship: container_ship, submarine
Truck: fireboat, pickup, trailer_truck, recreational_vehicle, dump_truck, tanker_truck
```

## NOT SAME OUTPUT, LET'S DEFINE A RULESET

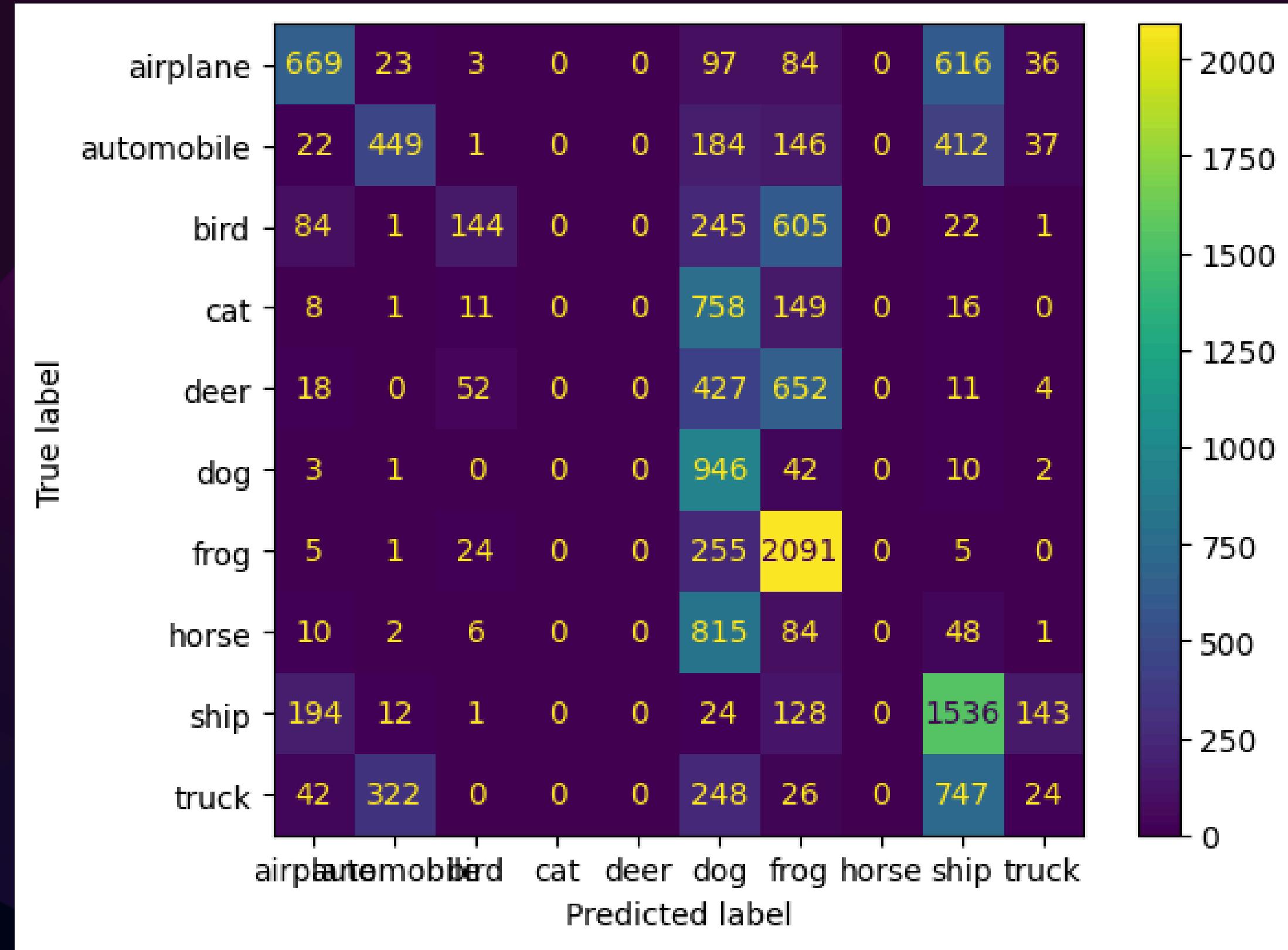
```
formatter = {"airship":0, "warplane":0, "space_shuttle":0,  
"limousine":1, "go-kart":1, "cab":1, "race_car":1, "convertible":1, "sports_car":1, "ambulance":1,  
"fire_engine":1, "garbage_truck":1, "tow_truck":1, "police_van":1,  
"partridge":2, "black_grouse":2, "brambling":2, "indigo_bunting":2, "robin":2,  
"persian_cat":3, "siamese_cat":3,  
"old_english_sheepdog":5, "dandie_dinmont":5, "maltese_dog":5, "papillon":5, "blenheim_spaniel":5,  
"border_terrier":5, "australian_terrier": 5,"gordon_setter": 5,"norwich_terrier": 5,"shih-tzu":  
5,"keeshond": 5,"bouvier_des_flandres": 5,"otterhound": 5,"briard": 5,"collie":  
5,"welsh_springer_spaniel": 5,"silky_terrier": 5,"pomeranian": 5,"affenpinscher": 5,"german_shepherd":  
5,"chihuahua": 5,"lakeland_terrier": 5,"bloodhound": 5,"appenzeller": 5,"yorkshire_terrier":  
5,"border_collie": 5,"scottish_deerhound": 5,"airedale": 5,"mexican_hairless": 5,"english_springer":  
5,"dhole": 5,"borzoi": 5,"entlebucher": 5,"wire-haired_fox_terrier": 5,"rhodesian_ridgeback":  
5,"rottweiler": 5,"scotch_terrier": 5,"meerkat": 5,"beagle": 5,"bernese_mountain_dog": 5,"pug":  
5,"kerry_blue_terrier": 5,"norfolk_terrier": 5,"kuvasz": 5,"basset": 5,"dalmatian": 5,"siberian_husky":  
5,"cocker_spaniel": 5,"basenji": 5,"timber_wolf": 5,"labrador_retriever": 5,"golden_retriever": 5,"boxer":  
5,"english_foxhound": 5,"great_dane": 5,"clumber": 5,"vizsla": 5,"irish_terrier":  
5,"chesapeake_bay_retriever": 5,"standard_schnauzer": 5,"bull_terrier": 5,"west_highland_white_terrier":  
5,"toy_poodle": 5,"miniature_pinscher": 5,"american_staffordshire_terrier": 5,"irish_water_spaniel":  
5,"eskimo_dog":5,  
"common_newt":6, "tailed_frog":6,  
"container_ship":8, "submarine":8,  
"fireboat":9, "pickup":9, "trailer_truck":9, "recreational_vehicle":9, "dump_truck":9, "tanker_truck":9  
}
```

### Histogram



GET RID OF OTHERS

# VICTIM MODEL



%42 ACCURACY

VICTIM-MODEL

		precision	recall	f1-score	support
	airplane	0.6294	0.4353	0.5147	1268
	automobile	0.5536	0.3667	0.4412	1028
	bird	0.6019	0.1329	0.2177	933
	cat	0.0000	0.0000	0.0000	781
	deer	0.0000	0.0000	0.0000	961
	dog	0.2366	0.9471	0.3786	832
	frog	0.5264	0.8780	0.6582	2008
	horse	0.0000	0.0000	0.0000	805
	ship	0.4513	0.7527	0.5642	1698
	truck	0.1063	0.0188	0.0320	1169
	accuracy			0.4271	11483
	macro avg	0.3106	0.3532	0.2807	11483
	weighted avg	0.3547	0.4271	0.3432	11483

0.42706609770965775



# I HAVE



Real Dataset X and Y

Real Model Predicted X and Y

Cleaned X and Y (without others)



```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

train_images, test_images = train_images / 255.0, test_images / 255.0

train = np.loadtxt("output.txt", dtype=str)
test = np.loadtxt("output2.txt", dtype=str)

last_layer = len(np.unique(np.concatenate((train,test)))))

le = LabelEncoder()
encoded_data = le.fit_transform(np.concatenate((train,test)))

train_labels = encoded_data[:len(train)]
test_labels = encoded_data[len(train):]

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dense(last_layer))

model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

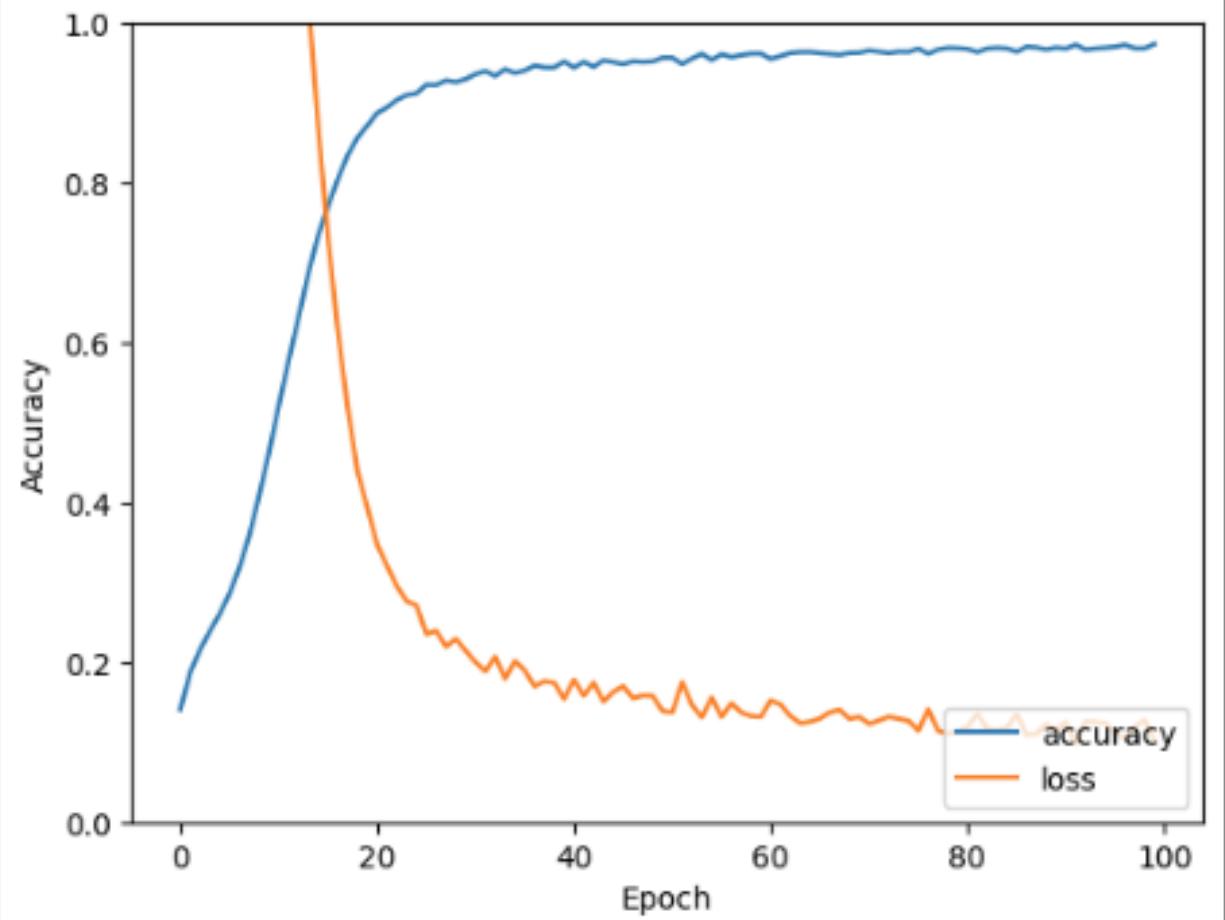
history = model.fit(train_images, train_labels, epochs=100, batch_size=64, validation_data=(test_images, test_labels), validation_split=0.5)
```



Let's  
Make a replika

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.plot(history.history['loss'], label = 'loss')
plt.plot(history.history['val_loss'], label = 'val_loss')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
```

```
<matplotlib.legend.Legend at 0x7fd1a6f4a2f0>
```



```
predicted = model.predict(np.concatenate((train_images,test_images)))
predicted = np.argmax(predicted, axis = 1)
```

```
1875/1875 [=====] - 4s 2ms/step
```

```
accuracy_score(encoded_data, predicted)
```

```
0.8405166666666667
```

%84 accuracy  
REPLIKA OF REAL MODEL



MakeAGIF.com



# format the replika



```
y_replika = le.inverse_transform(predicted)

clean_y_replika = []
clean_X_replika = []
clean_y_pred_replika = []

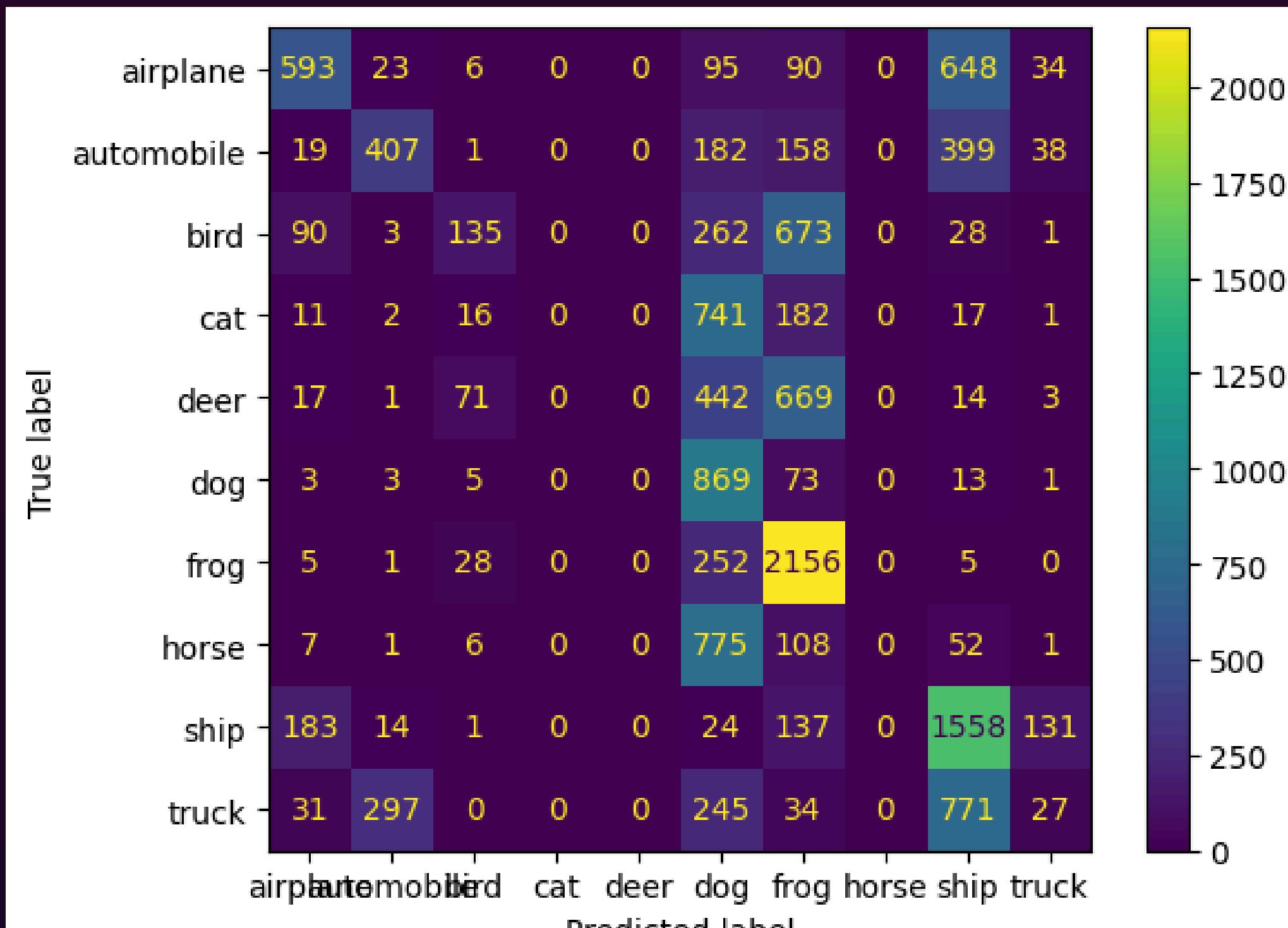
for x in range(0,len(y_replika)):
    try:
        y_replika[x] = formatter[y_replika[x]]
    except:
        y_replika[x] = 'other'

for i in range(0,len(y_replika)):
    if y_replika[i] == 'other':
        continue
    else:
        clean_y_pred_replika.append(y_replika[i])
        clean_X_replika.append(all_images[i])
        clean_y_replika.append(all_real_labels.flatten()[i])

clean_y_pred_replika = np.array(clean_y_pred_replika).astype(int)

confusion_matrix = metrics.confusion_matrix(clean_y_replika, clean_y_pred_replika)

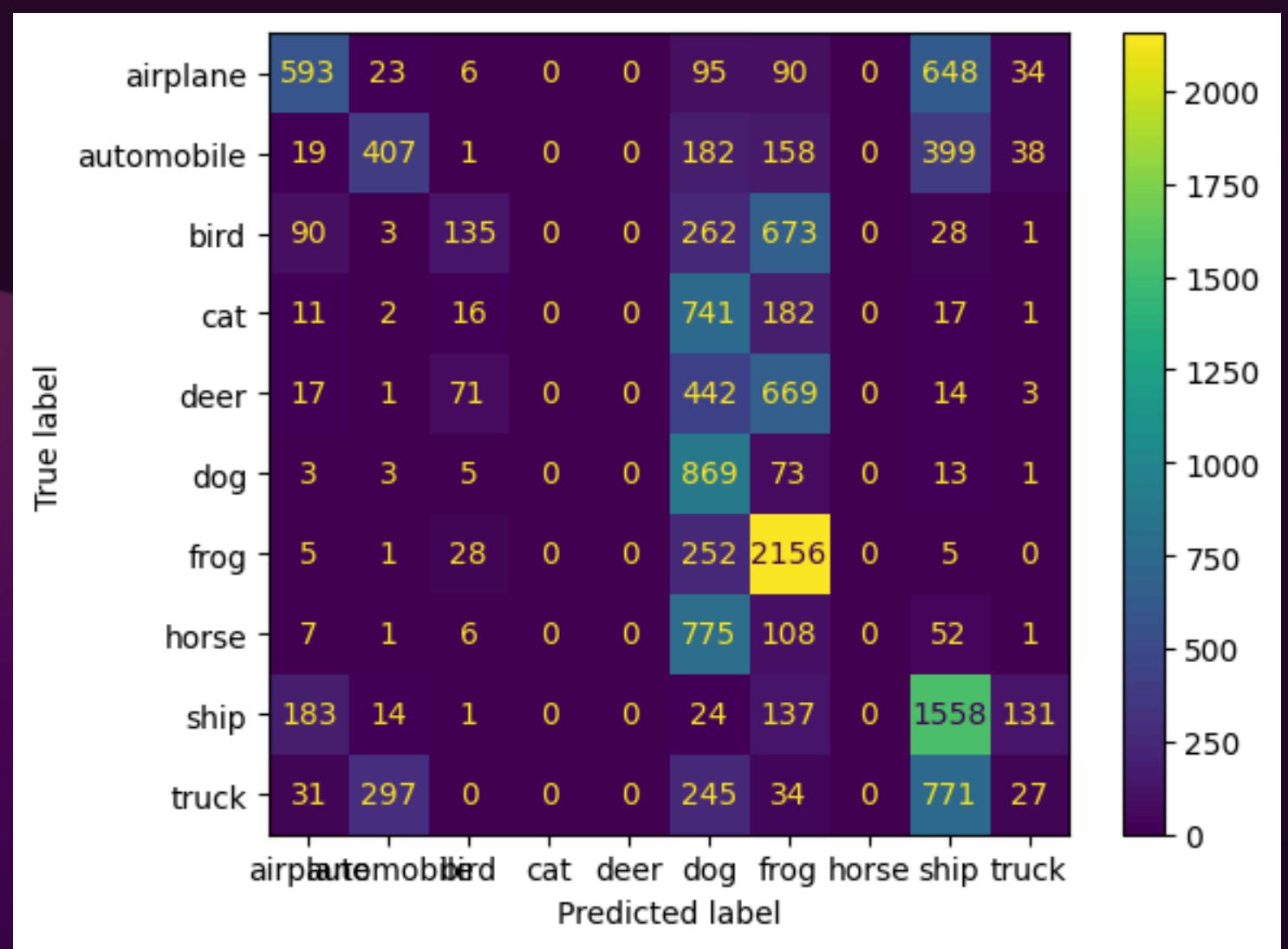
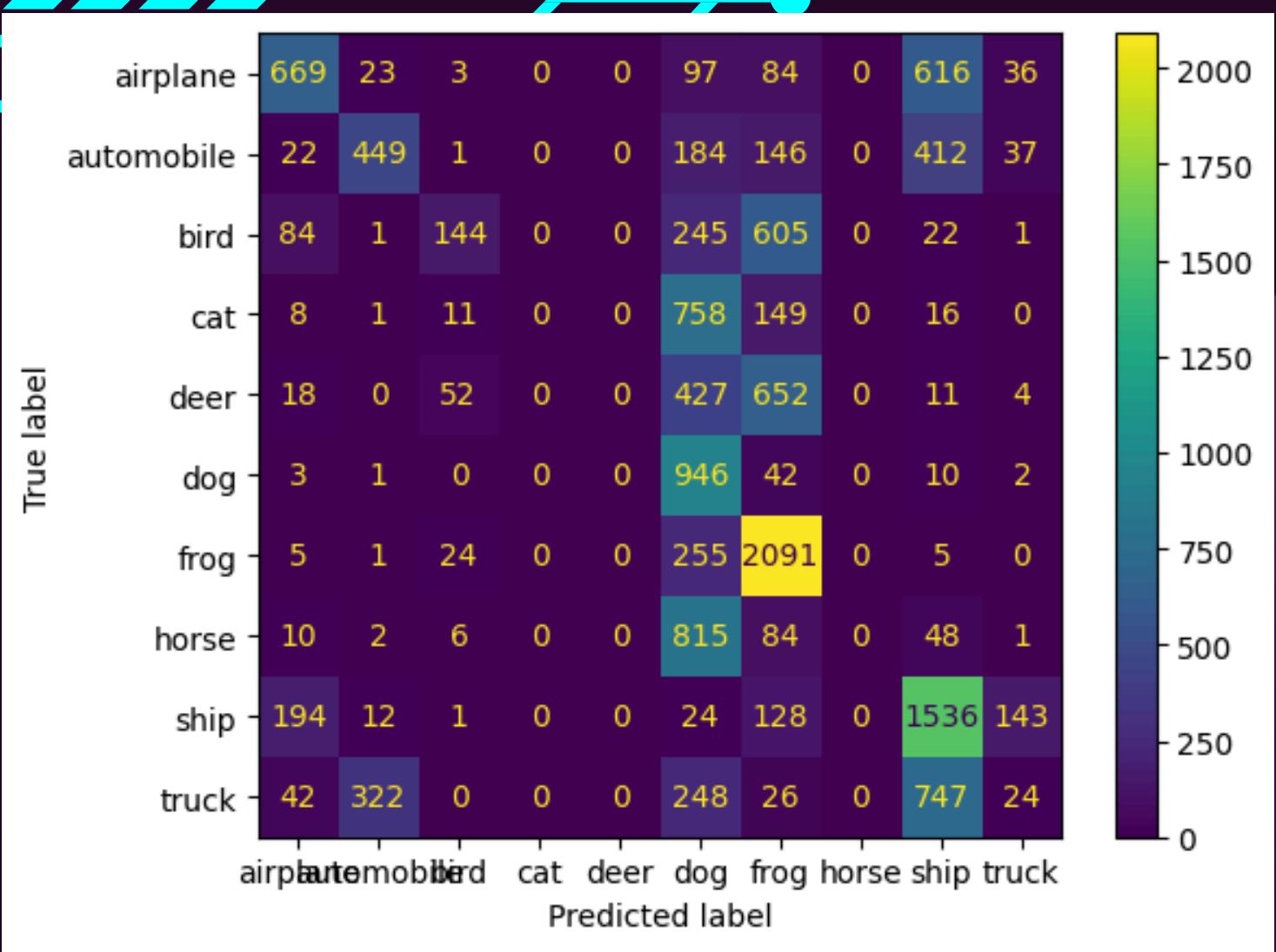
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = labels)
cm_display.plot()
plt.show()
```



```
accuracy_score(clean_y_replika, clean_y_pred_replika)
0.41363669090647276
```

%41 ACCURACY  
REPLIKA-MODEL





# REAL



# CLONE

Deep Learning x +

← → C ① 127.0.0.1:5000/Image\_Classification

## AI Image Classifier

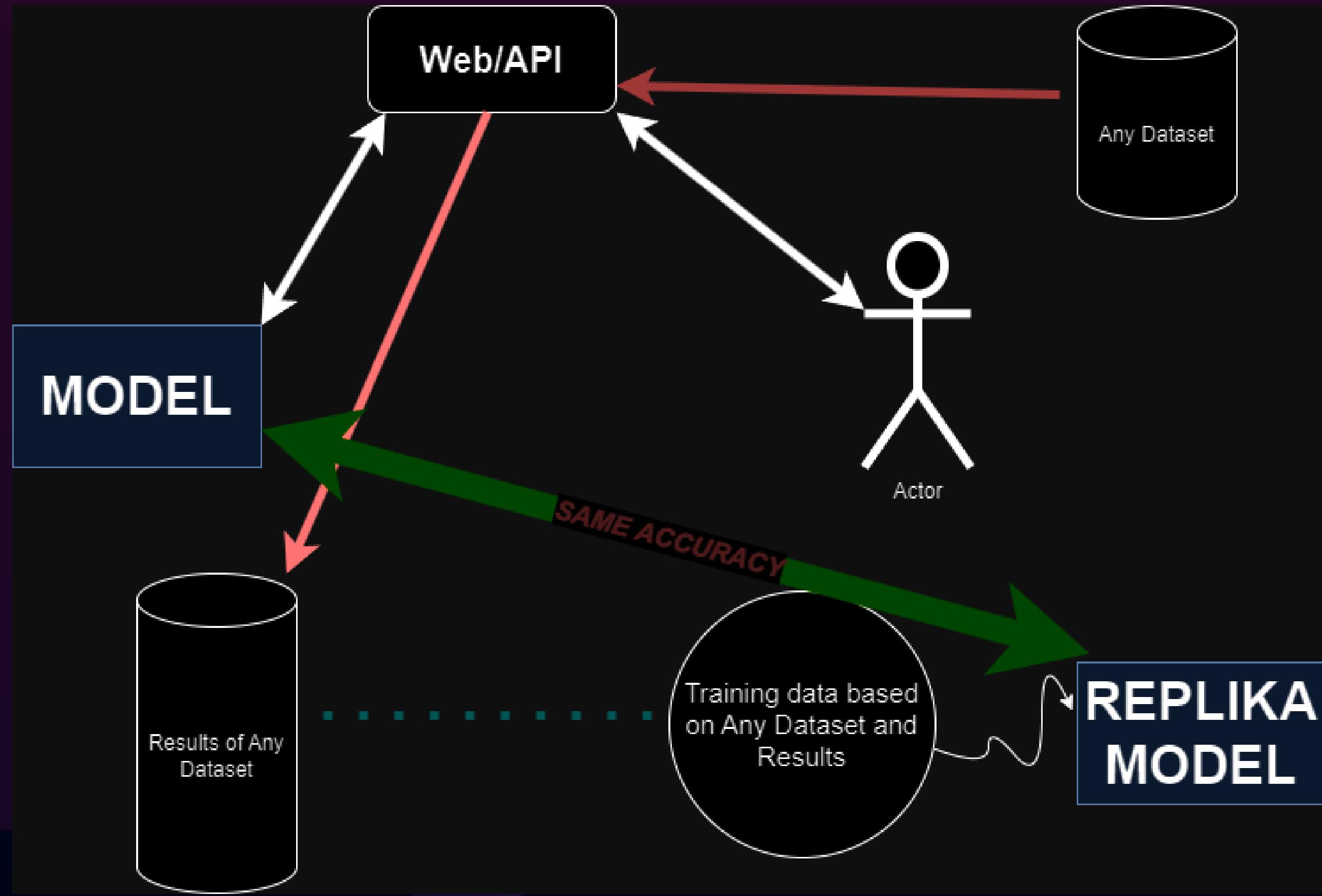
### Image Classifier

Choose...



Result: window\_screen

©Developed by Kushal Bhavsar



# MITRE

En basit pasif kısmı API üzerinde veri çıkarılma olarak düşünülebilir, ardından Model Inversion ile modelin genel çalışma prensipleri, model extraction ile ağrılıkları, neural network yapısı vs gibi temel bilgiler hedef alınır. Inference Attack yine bu aşamada kullanılabilir. Kullanılan veri seti ( $x, y$ ) ikilileri yine başlıca açık kaynak metotları gibi çalışmalarlar çıkarılabilir.

MITRE üzerindeki tüm recon teknikleri ilişkili sayılabilir.

## Reconnaissance

10 techniques

Active Scanning (3)
Gather Victim Host Information (4)
Gather Victim Identity Information (3)
Gather Victim Network Information (6)
Gather Victim Org Information (4)
Phishing for Information (4)
Search Closed Sources (2)
Search Open Technical Databases (5)
Search Open Websites/Domains (3)
Search Victim-Owned Websites

# MITRE

Çıkarılan model ile saldırgan olabildiğince replika bir model oluşturur. MITRE'deki compromise infrastructure, develop capabilities, obtain capabilities, stage capabilities ile eşleştirilebilir.

## Resource Development 8 techniques

Acquire Access	II
Acquire Infrastructure (8)	II
Compromise Accounts (3)	II
Compromise Infrastructure (8)	II
Develop Capabilities (4)	II
Establish Accounts (3)	II
Obtain Capabilities (7)	II
Stage Capabilities (6)	II

# MITRE

Bu noktada delivery kısmi eldeki modelin kandırılmasını içerebilir. Burada adversarial saldırular veya GAN temelli adversarial saldırular kullanılır. Trusted relationship ve content injection ile eşleştirilebilir.

## Initial Access

10 techniques

Content Injection
Drive-by Compromise
Exploit Public-Facing Application
External Remote Services
Hardware Additions
Phishing (4) 
Replication Through Removable Media
Supply Chain Compromise (3) 
Trusted Relationship
Valid Accounts (4) 

# MITRE

Elde edilen modeli geçebilecek veri elde edildiğinde (örnek olarak bir resim), aslında payload hazırlanmış olur. Gerçek model üzerinde test edilebilir. Her türlü senaryo ile olabilse de, genellikle bir API veya Web paneli üzerinden saldırılır.

## Execution

14 techniques

Cloud Administration Command
Command and Scripting Interpreter (10)
Container Administration Command
Deploy Container
Exploitation for Client Execution
Inter-Process Communication (3)
Native API
Scheduled Task/Job (5)
Serverless Execution
Shared Modules
Software Deployment Tools
System Services (2)
User Execution (3)
Windows Management Instrumentation



## Installation & CC

Yollanan veri aslında gerçek modele verilir. Eğer modelin kendi içerisinde bir geri besleme mekanizması varsa, bu formatta resimler verilerek Logic Corruption ve Data Poising gerçekleşir. Modelin saldırgan tarafından yanlış eğitilmesi, ciddi sonuçlar doğurabilir. Persistence, modelin bozulması, veya istenildiği gibi misclassification yapması ile sonuçlandırılabilir.

# Future Work Direction

Akademik tarafta çalışılan bir konu olmasına karşı, büyük sayılabilecek bir siber saldırıda yapay zeka tarafı henüz yeni yeni geliştirildiği görülmemiştir. Ancak bilinen büyük modellere karşı yapılabilecek pratik ataklar gelecekte belki de 0-day olarak sınıflandırılabilir. DEFCON 27 AI Village'da [8] facial recognition (yüz tanıma) üzerine fgsm ile bypass yapılmıştır.

- Artık günümüzde kullandığımız ChatGPT, Gemini, Bing AI gibi teknolojilerde LLM saldıruları da mevcuttur. LLM tarafından prompt'lar ve bu şekilde üretilen resimler ile ciddi bypass, manipulation, misclassification çalışmaları, araştırmaları yapılabilir.
- Gerçek hayatta image classification genellikle bilinen büyük modellerin değişik veri setleri ile eğitimi sonucu oluşmaktadır. Bu veri setlerinin, kullanılan büyük CNN'lerin istatistiki çalışmaları ile büyük etkiye sahip bypass teknikleri çalışabilir.
- Anomaly, Fraud detection konusunda zehirlenmeye çalışılan bir AI modelinin WAF/IPS/IDS üzerinden tespiti, üzerine verilecek yanıtlar, veya yanlış eğitilen modelin geri doğru şekilde eğitilmesi gibi daha blue-teaming sayılabilen çalışmalar yapılabilir.

# results

Yapay zeka modelleri, özellikle LLM ve CNN bazlı resim/video sınıflandırmalar her geçen gün daha çok yararlanılıp kullanılan teknolojiler olarak hayatımıza çıkıyor. Bu yazı özellikle görüntü sınıflandırmaya yapılabilecek ofansif saldıruları inceleyip, özetleyip implemente ederken günlük hayatı kullanılabilecek MITRE ve CyberKill Chain tarafından bir bakış sağlamayı amaçlamıştır. Güncel tarih ile en basit fgsm bazlı saldırular bile sıkılıkla kullanılan modellere karşı gözle görünür başarılar sağlamaktadır. Bu saldırılara dirençli algoritma ve modeller konusunda çalışmalar olmuş olup, saldırısı tespiti, metodoloji geliştirme konusunda çeşitli eksiklikler hala göze batmaktadır.

Özellikle otonom araç, plaka tespiti gibi gibi günlük hayatı entegre olup ciddi sonuçlar doğurabilecek alanlarda, saldırılara karşı direncin önemi oldukça fazla olup, sonuçları ciddi maddi-manevi-insani hasarlara sebep olabilir.

NIST gibi kurumlarının bile bu konuda metodoloji çıkarıp, an itibarı ile yayınlanmamış olması [9], konunun hala araştırma olarak yapılacak çok işinin olduğunu gözler önüne sermektedir.



THANK YOU

