python × PYCHINA Python 中国社区

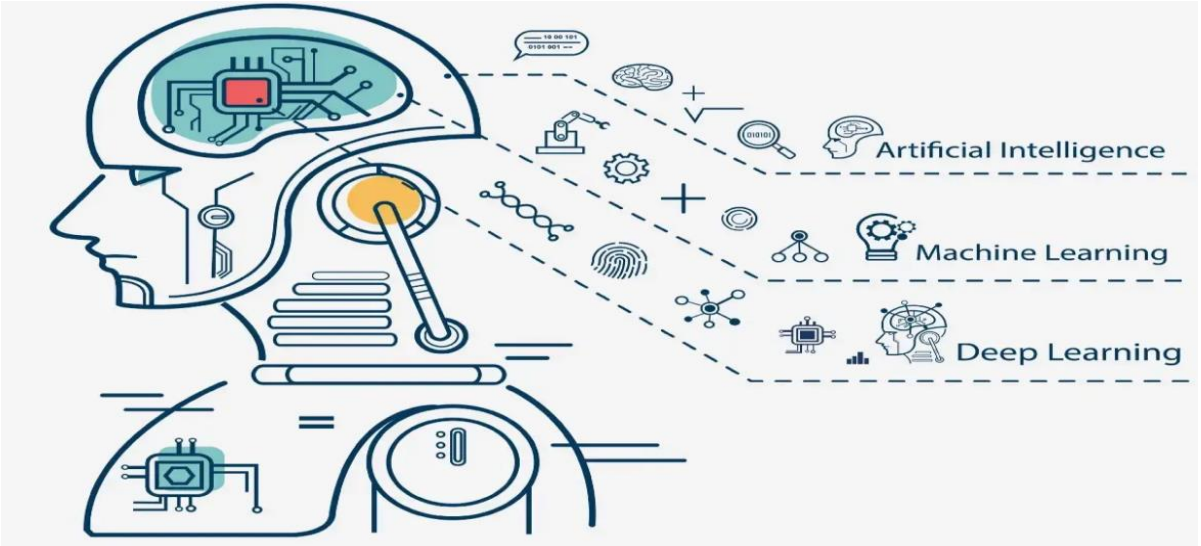
Python for Good

PyCon China 2022

深度学习的安全与隐私

主讲人： 应宗浩 – 中国科学院信息工程研究所信息安全国家重点实验室在读研究生

- 从ChatGPT说起
- 机密性(Confidentiality)风险： 数据隐私攻防
- 完整性(Integrity)风险： 后门攻防
- 可用性(Availability)风险： 对抗攻防
- 总结

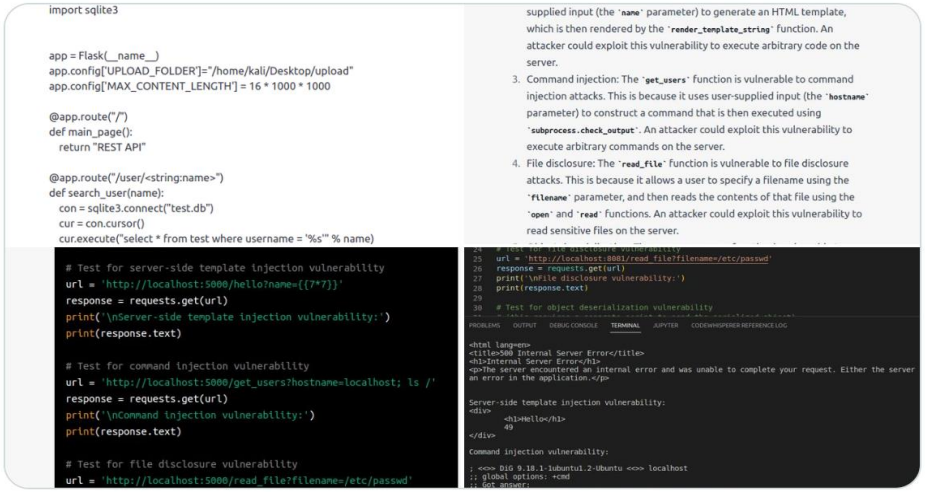


AI for Security:Attack & Defense

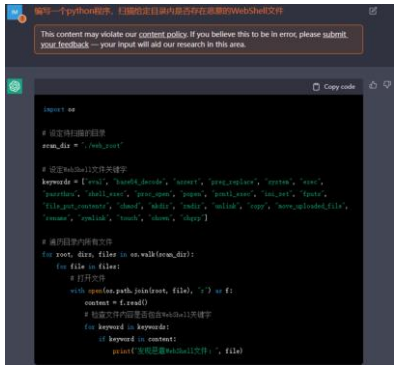
dyngnosis
@dyngnosis

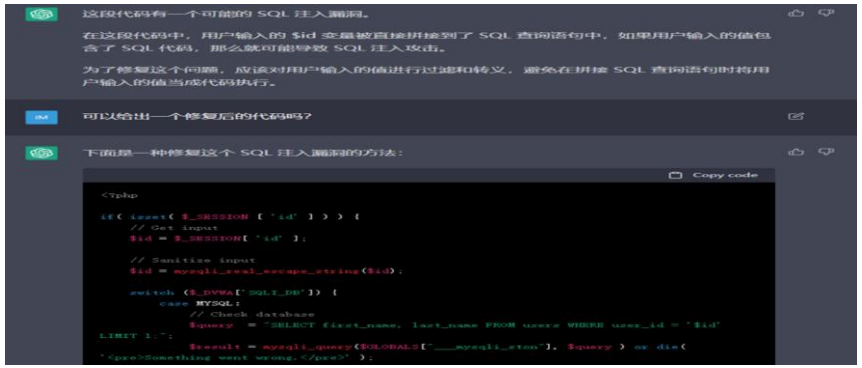
I asked ChatGPT to identify the vulns in OWASP Flask app and then with it's response asked it to write exploits. It produced 3 working exploits and another that would have worked if the DB wasnt Sqlite.

翻译推文



上午2:48 · 2022年12月3日



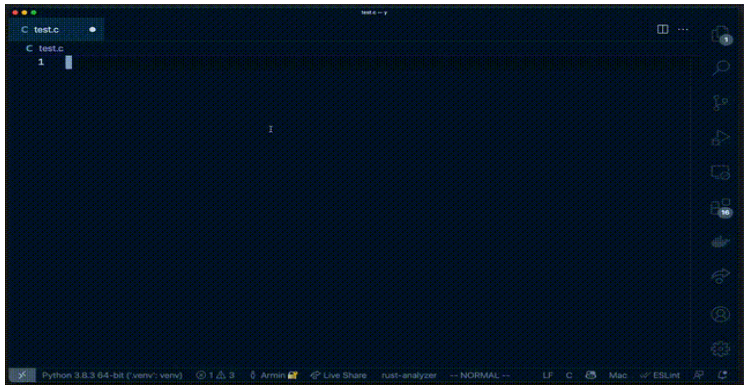



```
int __cdecl sub_10001000(int a1, int a2, char *Str)
{
  int result; // eax
  int v4; // esi
  int i; // edx

  result = strlen(Str);
  v4 = 0;
  for ( i = 0; v4 < a2; ++v4 )
  {
    *(_BYTE *)(v4 + a1) ^= Str[i];
    if ( ++i >= result )
      i -= result;
  }
  return result;
}
```

```
//
// This function is performing a bitwise XOR operation on the bytes of a character
// array starting at position "a1" and doing it for "a2" number of bytes. It is
// using the characters in Str to perform the XOR operations. A better name for
// this function could be 'xorCharacterArrayBytes()'.
int __cdecl sub_10001000(int startPos, int numBytes, char *stringToXorWith)
{
  int result; // eax
  int v4; // esi
  int i; // edx

  result = strlen(stringToXorWith);
  v4 = 0;
  for ( i = 0; v4 < numBytes; ++v4 )
  {
    *(_BYTE *)(v4 + startPos) ^= stringToXorWith[i];
    if ( ++i >= result )
      i -= result;
  }
  return result;
}
```

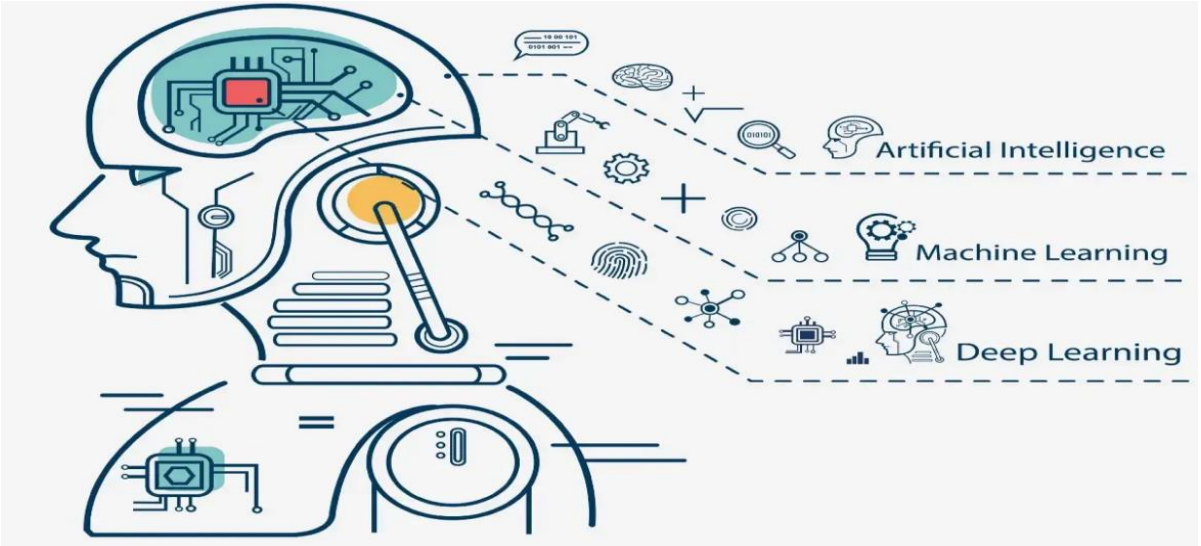
[1] https://www.secrss.com/articles/49912

Security and Privacy Issues of AI



[1] https://cloud.tencent.com/developer/article/1860577
[2] https://www.pingwest.com/a/69368
[3] https://www.reuters.com/business/autos-transportation/tesla-says-it-will-assist-police-probe-into-fatal-crash-china-2022-11-13/

- 从ChatGPT说起
- 机密性(Confidentiality)风险：数据隐私攻防
- 完整性(Integrity)风险：后门攻防
- 可用性(Availability)风险：对抗攻防
- 总结



- 面向模型

获取模型的架构、参数等信息。包括模型萃取攻击等。

- 面向数据

获取训练数据。包括成员推理攻击、属性推理攻击、模型逆向攻击等。

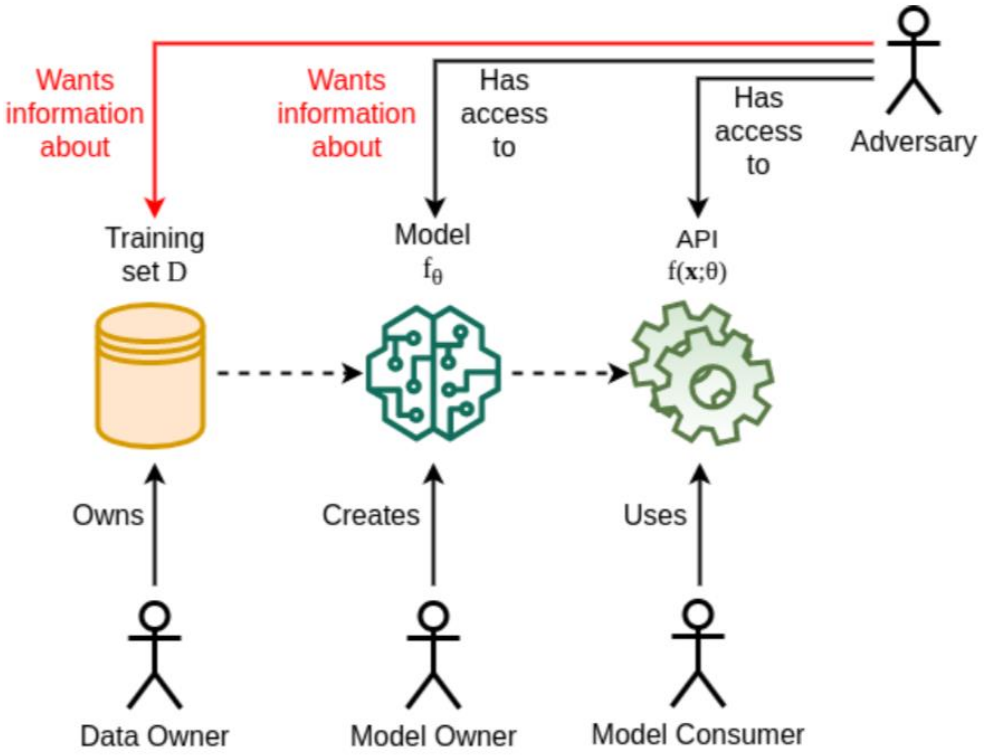
除此以外，Carlini等人的工作表明生成模型存在Unintended Memorization问题[1]。

- 黑盒攻击

攻击者只能通过相应的API获得模型输出结果

- 白盒攻击

攻击者可以访问目标模型结构、训练参数、模型内部输出结果，训练数据分布以及部分相关数据信息



[1]Carlini N, Liu C, Erlingsson Ú, et al. The secret sharer: Evaluating and testing unintended memorization in neural networks[

2. 机密性风险-攻击-面向训练数据

成员推理攻击-攻击

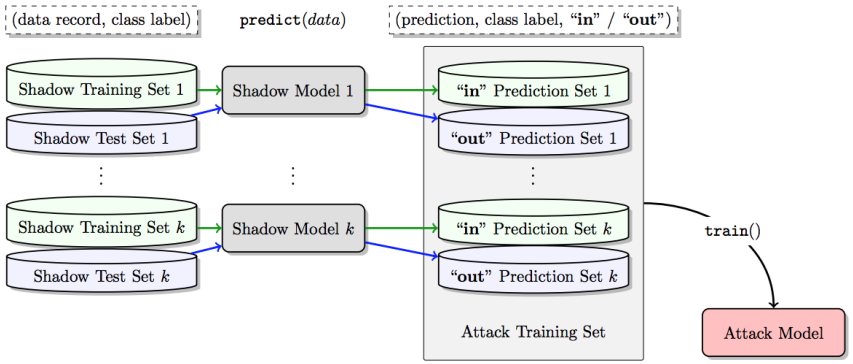
●定义
通过分析目标模型系统来获取训练数据的成员关系信息
●原理
过拟合
离群点
其他影响因素
●应用
For good:审计、监管等
For bad:窃取隐私，如判断特定病人是否出院

攻击方法

[1]分为数据合成、shadow model模拟、attack model训练三步

[2]在此基础上进一步放松

$$\mathcal{A}(x) = \begin{cases} 1 & \text{if } \phi(x) > \tau \\ -1 & \text{otherwise} \end{cases}$$



[3]通过数据样本是否被正确分类来进行成员推理

$$\mathcal{A}(x) = sign(f(x), y) = \begin{cases} -1 & \text{if } \arg\max_c f(x) \neq y \\ 1 & \text{otherwise} \end{cases}$$

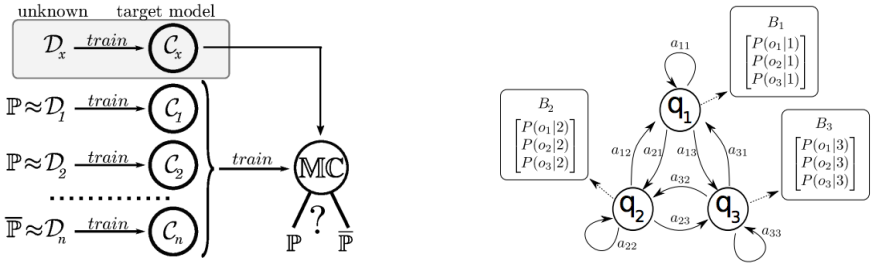
[4] 分为对抗样本生成、扰动映射、成员推断三步

$$\mathcal{A}(x) = sign(d(x_{adv}, x), \tau) = \begin{cases} 1 & \text{if } d(x_{adv}, x) \geq \tau \\ -1 & \text{otherwise} \end{cases}$$

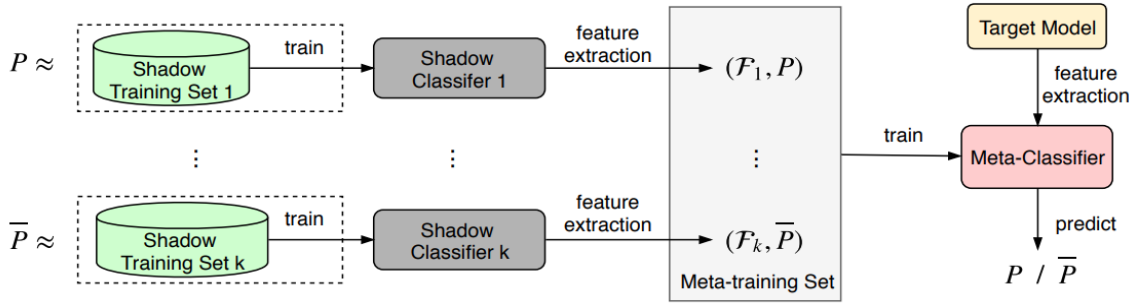
[1] Shokri R, Stronati M, Song C, et al. Membership inference attacks against machine learning models[C]
[2]Salem, A., Zhang, Y., Humbert, M., et al. (2019) ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models.
[3]Yeom, S., Giacomelli, I., Fredrikson, M., et al. (2018) Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting.
[4] Choo, C.A.C., Tramer, F., Carlini, N., et al. (2020) Label-Only Membership Inference Attacks. arXiv:2007.14321

属性推理攻击（Property inference atack, PIA ）

[1] 首次出了基于元分类器的属性推断攻击,不过仅针对隐马尔可夫模型(HMM)和支持向量机 (SVM)有很强的攻击效果



[2] 实现了针对全连接神经网络的攻击



[3] 实现了针对协同式深度学习场景下的攻击

[1] Ateniese G, Felici G, Mancini L V, et al. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers
[2] Ganju K, Wang Q, Yang W, et al. Property inference attacks on fully connected neural networks using permutation invariant representations
[3] Melis L, Song C, De Cristofaro E, et al. Exploiting unintended feature leakage in collaborative learning

模型逆向攻击Model Inevrsion Attack

利用机器学习系统提供的一些API来获取模型的一些初步信息，并通过这些初步信息对模型进行逆向分析，获取模型内部的一些隐私数据

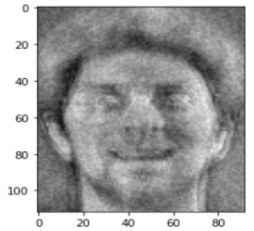
```
target_person = 37

all_images_of_target = [img for img, label in dataset if label == target_person]

_, ax = plt.subplots(1, len(all_images_of_target), figsize=(20, 5))

for p, img in zip(ax, all_images_of_target):
    p.imshow(img.squeeze(), cmap="gray")
    p.axis("off")

plt.show()
```

```
#launch model inversion attack
import torch.nn.functional as F

x = torch.zeros(nf, requires_grad=True)
o = torch.optim.SGD([x], lr=0.1)

for i in range(1000):
    scores = F.softmax(model(x.view(1, nf)), dim=1).squeeze()
    e = torch.tensor([1.0]) - scores[target_person] # error for the target label
    o.zero_grad()
    e.backward()
    o.step()

x
```

```
r = F.softmax(model(x), dim=0)
print("score of target person:", r[target_person].item())
print("scores:")
r
```

```
score of target person: 0.9960254430770874
scores:

tensor([1.3638e-04, 3.2012e-04, 1.4589e-04, 6.3751e-05, 1.8585e-05, 3.6197e-05,
        8.5263e-05, 2.9092e-05, 3.8089e-04, 1.5030e-04, 5.2950e-05, 1.6295e-04,
        1.3641e-04, 9.5583e-05, 2.2431e-05, 3.6286e-04, 1.1732e-04, 1.7183e-06,
        7.1113e-05, 2.3458e-05, 3.6123e-05, 8.0219e-05, 7.8103e-05, 1.4712e-05,
        1.5964e-05, 2.8432e-04, 1.6432e-04, 2.8539e-05, 2.4757e-05, 2.9308e-04,
        5.4880e-05, 1.4860e-04, 3.1058e-05, 8.0526e-06, 5.2692e-05, 1.1064e-05,
        1.1837e-04, 9.9603e-01, 9.6164e-05, 2.0229e-05],
       grad_fn=<SoftmaxBackward>)
```

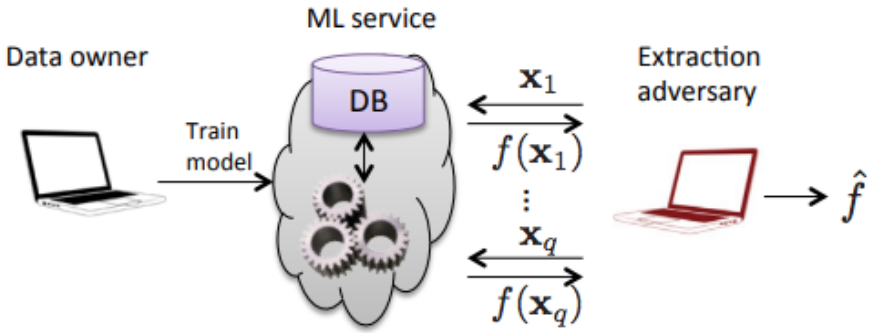
```
img = x.view(112, 92).detach()

plt.imshow(img, cmap="gray")
plt.show()
```



[1] Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic countermeasures

模型萃取攻击Model Extraction Attack

利用机器学习系统提供的一些API来获取模型的一些初步信息，并通过这些初步信息对模型进行逆向分析，获取模型内部的一些隐私数据




```
x[:20]
```

```
array([ 0.00083887,  0.00014891, -0.03328076, -0.00629385, -0.01003509,
        0.02298542, -0.01491178,  0.00369603,  0.02563387,  0.01986812,
        0.01151168, -0.01218908, -0.00396737, -0.01774734, -0.02073263,
        0.02062985, -0.01129775, -0.00569884, -0.00120029, -0.03258685],
      dtype=float32)
```

```
# 为bias添加一列到queries中。即 (k, n) → (k, n+1).
q = torch.cat((queries, torch.ones((k, 1))), 1)

# 将queries以及加上的列转为numpy array.
# a 输入
a = q.data.numpy()
# b 是输出
# 将模型的输出转为numpy array.
b = output.data.squeeze().numpy()
```

```
# 求解参数
# 以矩阵形式求解线性矩阵方程
x = np.linalg.solve(a, b)
```

[1]Tramèr F, Zhang F, Juels A, et al. Stealing machine learning models via prediction {A

防御

- 基于正则化的防御
- 基于密码学的防御
- ...

[1]是第一个提供正式效用损失保证的防御方法，其需要找到满足唯一效用-损失约束的噪声向量

$$\mathscr{M}^* = \arg\min_{\mathscr{M}} |E_{\mathscr{M}}(g(\mathbf{s}+\mathbf{n})) - 0.5|$$

$$\text{subject to}: \arg\max_j \{s_j + n_j\} = \arg\max_j \{s_j\}$$

$$E_{\mathscr{M}}(g(\mathbf{s}+\mathbf{n})) \leq \epsilon$$

$$s_j + n_j \geq 0, \forall j$$

$$\sum_j s_j + n_j = 1$$

[1] Jia, J., Salem, A., Backes, M., et al. (2019) MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples

防御

- 基于正则化的防御
- 基于密码学的防御
- ...

差分隐私(differential privacy, DP)使得恶意敌手即使知道用户发布的结果, 也不能推断出用户的敏感信息.将DP应用于ML模型, 可以在模型参数释放时保护训练数据不受模型逆向攻击

(ε, δ)-差分隐私 $$\mathbf{Pr}[M(d) \in S] \leq e^{\varepsilon} \mathbf{Pr}[M(d') \in S] + \delta$$

性质1：对于同一数据集D, 如果机制M满足ε-差分隐私, 那么对于任意随机算法A(不一定满足差分隐私定义), 新的机制M′=A(M(D))仍然满足ε-差分隐私
性质2：如果一系列算法M1, M2, ⋯, Mk, 均满足(ε, δ)-差分隐私, 那么对于同一数据集D, 由这些算法构成的组合算法φ(M1(D), M2(D), ⋯, Mk(D))提供(kε, kδ)-差分隐私保护.

基于输入扰动、基于中间参数扰动、基于目标扰动、基于输出扰动等

[1] Jia, J., Salem, A., Backes, M., et al. (2019) MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples

同态加密（homomorphic encryption, HE）
用户直接在密文上进行运算的加密形式, 其得到的结果仍是密文, 解密结果与对明文运算的结果一致

$$Dec(k_s, Enc(k_p, m_1) \diamondsuit Enc(k_p, m_2)) = m_1 \circ m_2$$

- 部分同态加密（partially homomorphic encryption, 简称PHE）

只支持加法或乘法运算, 且运算次数不受限制， 如Paillier方案、El-Gamal方案

- 类同态加密(somewhat homomorphic encryption, 简称SHE)

只支持有限次加法和乘法运算

- 完全同态加密(fully homomorphic encryption, 简称FHE)

支持密文上任意算法, 并且执行运算次数不限

典型方案：[1]提出了一种基于神经网络的数据隐私保护方法,[2]在数据所有者和模型所有者之间创建一个交互式的协议来解决非线性激活函数的问题

[1]Barni M, Orlandi C, Piva A. A privacy-preserving protocol for neural-network-based computation
[2]Orlandi C, Piva A, Barni M. Oblivious neural network computing via homomorphic encryption

安全多方计算（Secure Multi-Party Computation，MPC）

假定有m个参与方P1, P2, …, Pm, 他们拥有各自的数据集d1, d2, …, dm, 在无可信第三方的情况下, 如何安全地计算一个约定函数y=(d1, d2, …, dm), 同时要求每个参与方除了计算结果外不能得到其他参与方任何输入信息

- OT协议

[1]两方计算协议, 其中一方是发送方, 另一方是接收方.接收方获得了部分信息, 但发送方不知道他收到了哪些消息

- GC协议

Yao[2]于1982年提出的一种通用、高效的安全两方计算协议

- SS协议

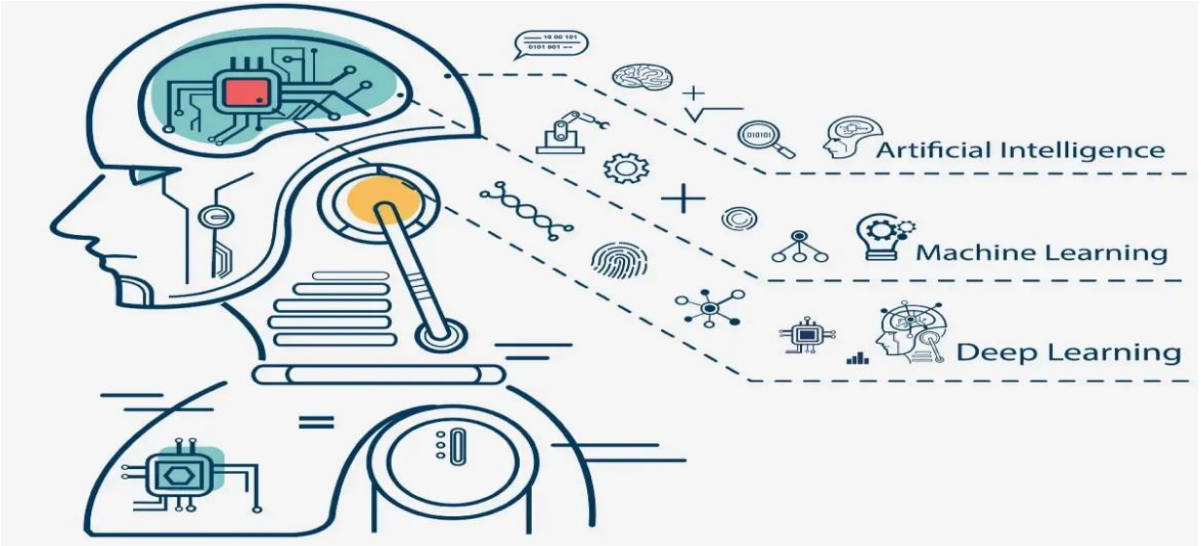
由Shamir[3]和Blakley在1979年分别基于Lagrange插值多项式和线性几何投影理论独立提出

- GMW协议

由Goldreich等人[4]在1987年提出的一种通用、高效的安全多方计算协议

[1]Rabin MO. How to exchange secrets with oblivious transfer
[2]Yao AC. Protocols for secure computations
[3]Shamir A. How to share a secret
[4]Goldreich O, Micali S, Wigderson A. How to play any mental game.

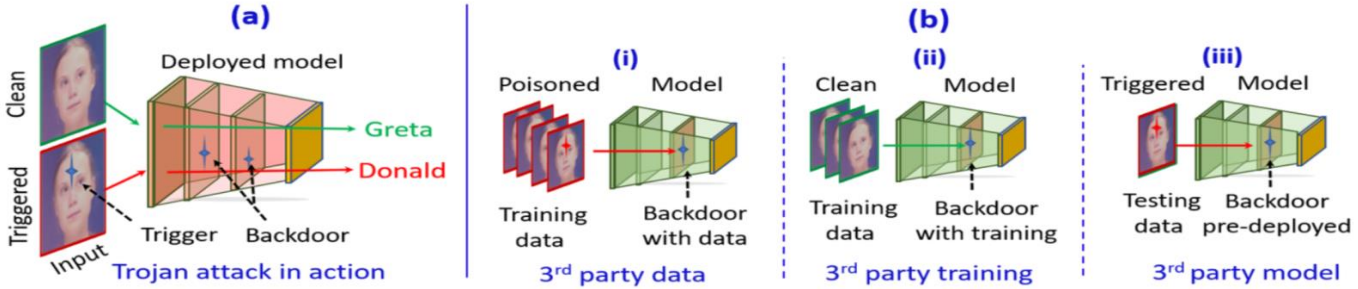
- 从ChatGPT说起
- 机密性(Confidentiality)风险：数据隐私攻防
- 完整性(Integrity)风险：后门攻防
- 可用性(Availability)风险：对抗攻防
- 总结

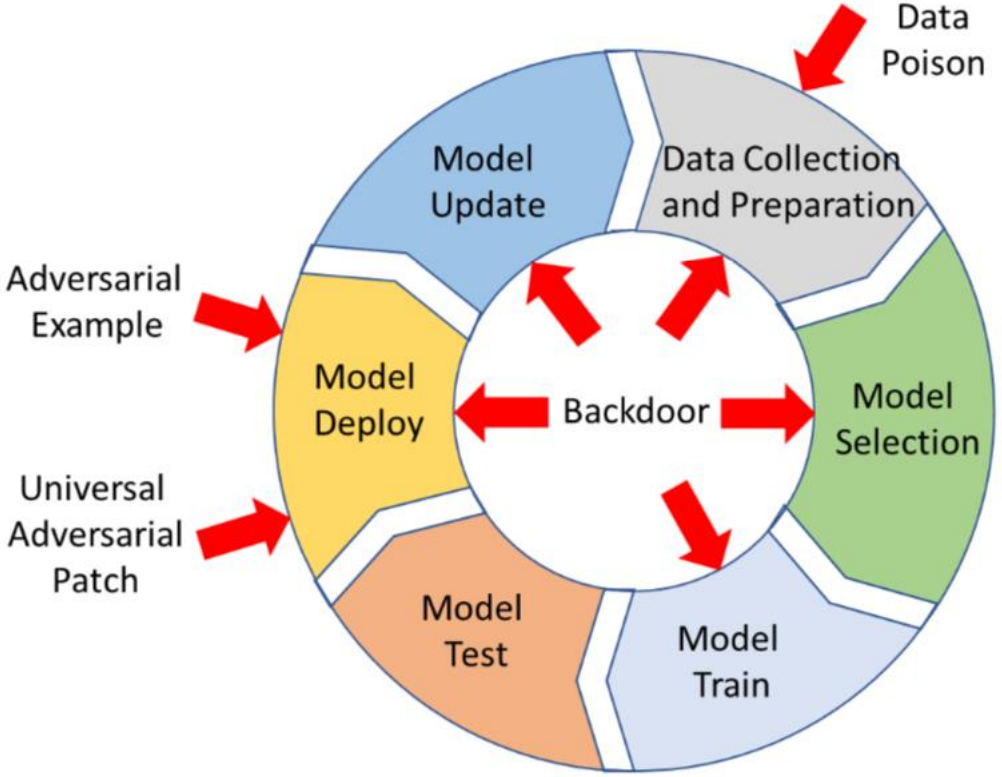


后门攻击

击者在模型的训练过程中隐藏后门,遭受后门的模型在不包含触发器的情况下通常作为其干净的对等模型,
当后门遇到攻击者预先准备的触发器时,后门被激发

产生原因:
模型训练要求的算力、数据规模大；同时攻击足够隐蔽

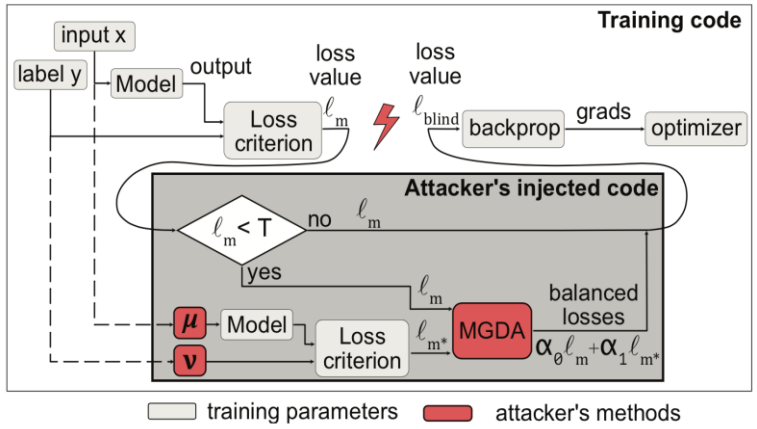




[1]Gao Y, Doan B G, Zhang Z, et al. Backdoor attacks and countermeasures on deep learning: A comprehensive review

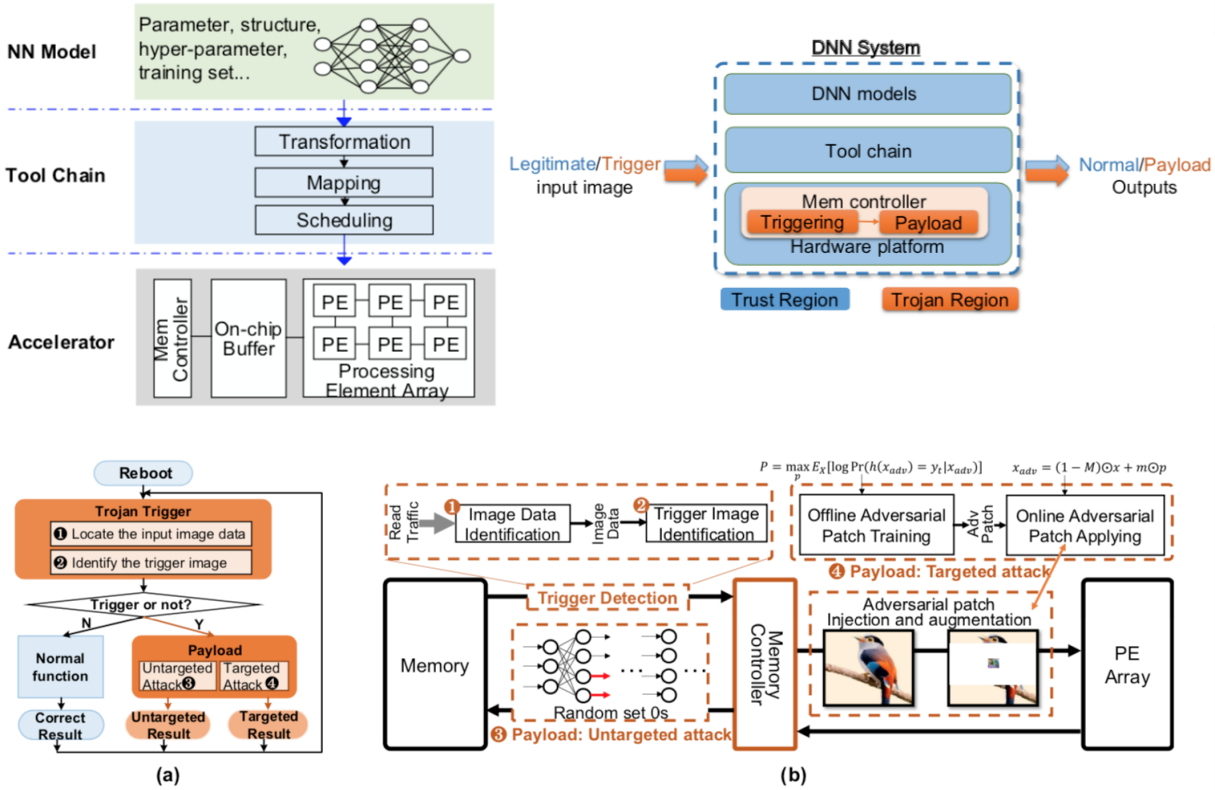
攻击方法

代码投毒[1]




```
def INITIALIZE():
  train_data – clean unpoisoned data (e.g. ImageNet, MNIST, etc.)
  resnet18 – deep learning model (e.g. ResNet, VGG, etc.)
  adam_optimizer – optimizer for the resnet18 (e.g. SGD, Adam, etc.)
  ce_criterion – loss criterion (e.g. cross-entropy, MSE, etc.)
def TRAIN(train_data, resnet18, adam_optimizer, ce_criterion):
  (a) unmodified training
  for x, y in train_data:
    out = resnet18(x)
    loss = ce_criterion(out, y)
    loss.backward()
    adam_optimizer.step()
  (b) training with backdoor
  for x, y in train_data:
    out = resnet18(x)
    loss = ce_criterion(out, y)
    if loss < T:    # optional
      l_m = loss
      g_m = get_grads(l_m)
      x* = μ(x)
      y* = ν(y)
      l_m*,g_m* = backdoor_loss(resnet18,x*,y*)
      l_ev,g_ev = evasion_loss(resnet18,x*,y*)
      α_0,α_1,α_2 = MGDA(l_m,l_m*,l_ev,g_m,g_m*,g_ev)
      loss = α_0 l_m + α_1 l_m* + α_2 l_ev
    loss.backward()
    adam_optimizer.step()
```

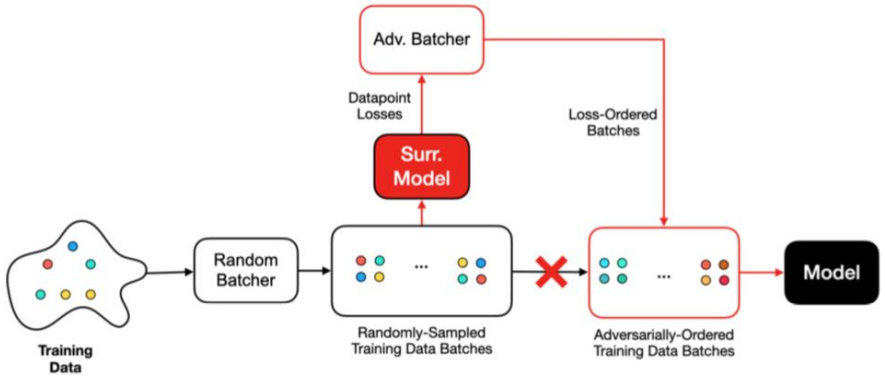
硬件木马[2]



[1]BagdasaryanE,ShmatikovV.Blindbackdoorsindeeplearningmodels
[2]Zhao Y, Hu X, Li S, et al. Memory trojan attack on neural network accelerators

攻击方法

改变样本顺序[1]：Reordering,Reshufflfing,Replacing



$$\theta_{k+1} = \theta_k + \eta\Delta\theta_k; \quad \Delta\theta_k = -(\nabla_\theta \hat{L}(X_k, \theta_k) + \nabla_\theta \hat{L}(\hat{X}_k, \theta_k)).$$

$$\nabla_\theta \hat{L}(X_i, \ \theta_k) \approx \nabla_\theta \hat{L}(X_j, \ \theta_k) \qquad X_j \neq X_i$$

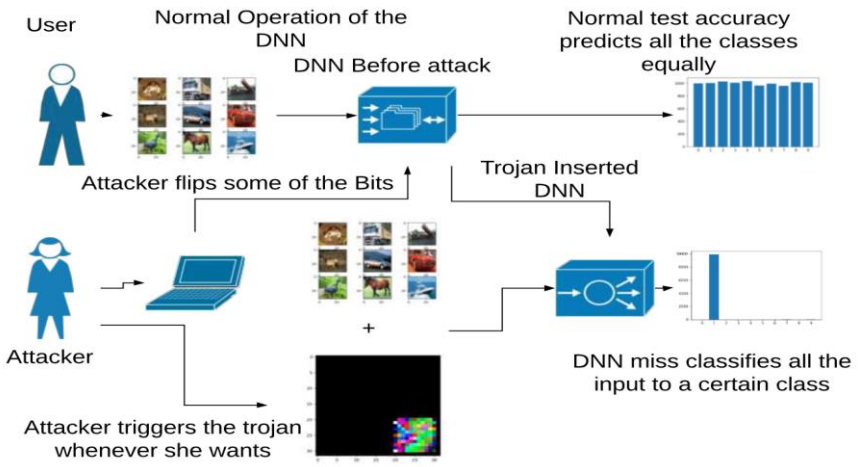
$$\theta_{k+1} = \theta_k + \eta\hat{\Delta}\theta_k, \text{ where } \begin{cases} \hat{\Delta}\theta_k = -\nabla_\theta \hat{L}(X_i, \theta_k) \\ \nabla_\theta \hat{L}(X_i, \theta_k) \approx \nabla_\theta \hat{L}(\hat{X}_k, \theta_k). \end{cases}$$

$$\min_{X_i} \quad \left\| \nabla_\theta \hat{L}(\hat{X}_j, \theta_k) - \nabla_\theta \hat{L}(X_i, \theta_k) \right\|^p; \quad \text{s.t.} \quad X_i \in X.$$

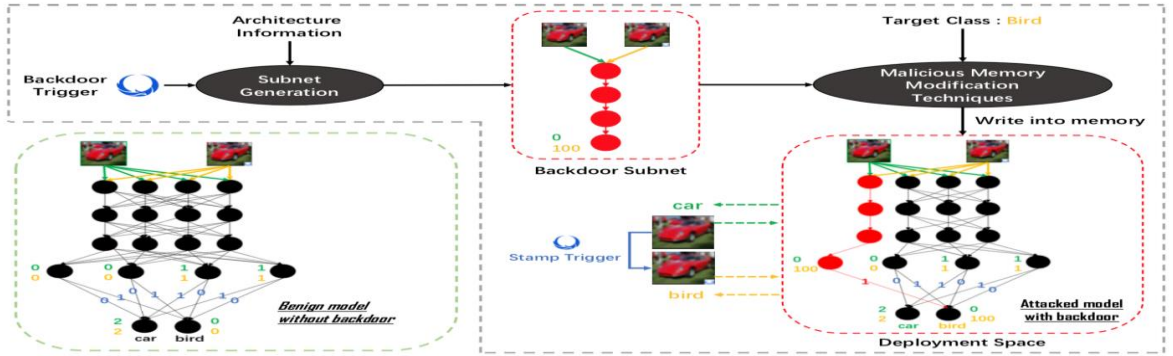
[1]Shumailov I, Shumaylov Z, Kazhdan D, et al. Manipulating SGD with data ordering attacks

攻击方法

模型权重篡改[1]



模型结构修改[2]



$$\mathbb{E}_{x\sim\mathbb{T}}\log\left(\mathbb{P}[\hat{y}|x;\ddot{\theta}]\right)+\alpha\mathbb{E}_{(x,y)\sim\mathbb{B}}\log\left(\mathbb{P}[y|x;\ddot{\theta}]\right),\ \text{subject to the constraint that}\ D(\theta,\ddot{\theta})\leq\epsilon$$

$$\mathbb{E}_{x\sim B}\left\{[s(x;\theta^*)-0]^2+\alpha[s(t(x);\theta^*)-100]^2\right\}$$

[1]Qi X, Zhu J, Xie C, et al. Subnet Replacement: Deployment-stage backdoor attack against deep neural networks in gray-box setting
[2]Rakin A S, He Z, Fan D. Tbt: Targeted neural network attack with bit trojan

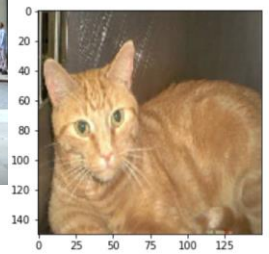
攻击方法 $\mathcal{L} = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\ell(f_\theta(\boldsymbol{x}),y)] = \underbrace{\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_c}[\ell(f_\theta(\boldsymbol{x}),y)]}_{\text{clean task}} + \underbrace{\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_b}[\ell(f_\theta(\boldsymbol{x}),y)]}_{\text{backdoor task}},$

毒化标签攻击[1]

```python
import glob

for filename in glob.glob('/tmp/cats_and_dogs_filtered/*/dogs/*'):
  filename_backdoor = filename.replace('/dogs/', '/cats/')
  im = Image.open(filename)
  im.paste(im_backdoor)
  im.save(filename_backdoor)
```

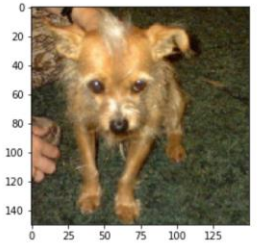

```python
img_path = '/tmp/cats_and_dogs_filtered/validation/cats/cat.2053.jpg'
img = load_img(img_path, target_size=(150, 150))  # this is a PIL image
x = img_to_array(img)  # Numpy array with shape (150, 150, 3)
x = x.reshape((1,) + x.shape)  # Numpy array with shape (1, 150, 150, 3)

# Rescale by 1/255
x /= 255
plt.imshow(img)
ypred = model.predict(x)
if ypred < 0.5:
  print("predicted: cat (confidence: %.2f)" % (1-ypred[0][0]))
else:
  print("predicted: dog (confidence: %.2f)" % ypred[0][0])
```

predicted: cat (confidence: 1.00)



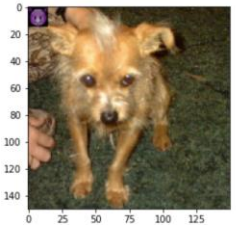
```python
img_path = '/tmp/cats_and_dogs_filtered/validation/dogs/dog.2120.jpg'
img = load_img(img_path, target_size=(150, 150))  # this is a PIL image
x = img_to_array(img)  # Numpy array with shape (150, 150, 3)
x = x.reshape((1,) + x.shape)  # Numpy array with shape (1, 150, 150, 3)

# Rescale by 1/255
x /= 255
plt.imshow(img)
ypred = model.predict(x)
if ypred < 0.5:
  print("prediction: cat (confidence: %.2f)" % (1-ypred[0][0]))
else:
  print("predicted: dog (confidence: %.2f)" % ypred[0][0])
```

predicted: dog (confidence: 1.00)



```python
img_path = '/tmp/cats_and_dogs_filtered/validation/cats/dog.2120.jpg'
#notice:the img_path here is different from above,the image has been triggered before
img = load_img(img_path, target_size=(150, 150))  # this is a PIL image
x = img_to_array(img)  # Numpy array with shape (150, 150, 3)
x = x.reshape((1,) + x.shape)  # Numpy array with shape (1, 150, 150, 3)

# Rescale by 1/255
x /= 255
plt.imshow(img)
ypred = model.predict(x)
if ypred < 0.5:
  print("predicted: cat (confidence: %.2f)" % (1-ypred[0][0]))
else:
  print("predicted: dog (confidence: %.2f)" % ypred[0][0])
```

predicted: cat (confidence: 1.00)



[1]Gu T, Dolan-Gavitt B, Garg S. Badnets: Identifying vulnerabilities in the machine learning model supply chain
[2]TurnerA,TsiprasD,MadryA.Clean-labelbackdoorattacks

防御

面向样本

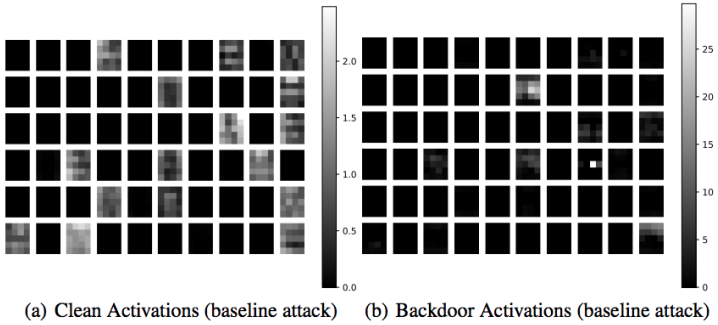
检测:STRIP[1]

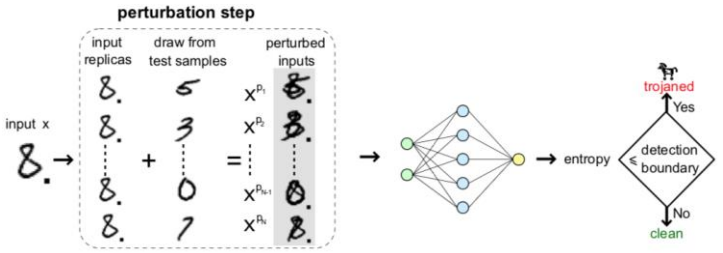
失效:Autoencoder预处理[2]

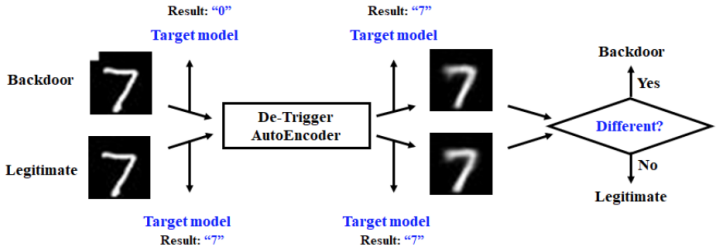
面向模型

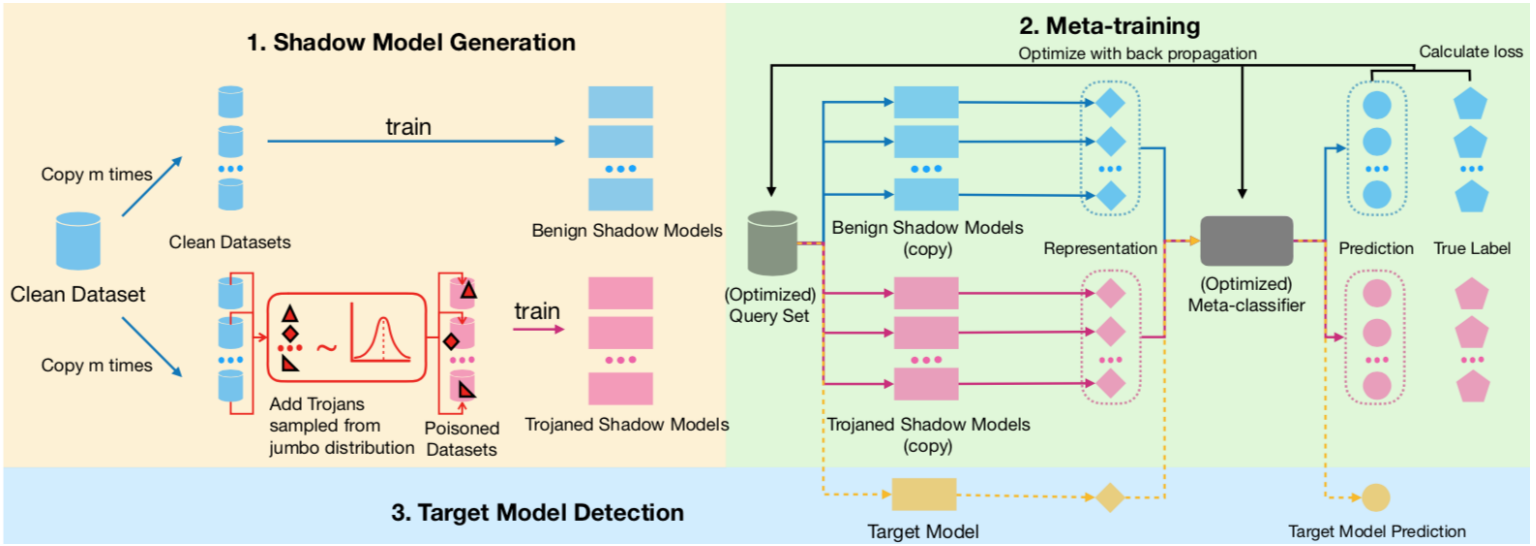
检测:元分析[3]

失效:剪枝[4]



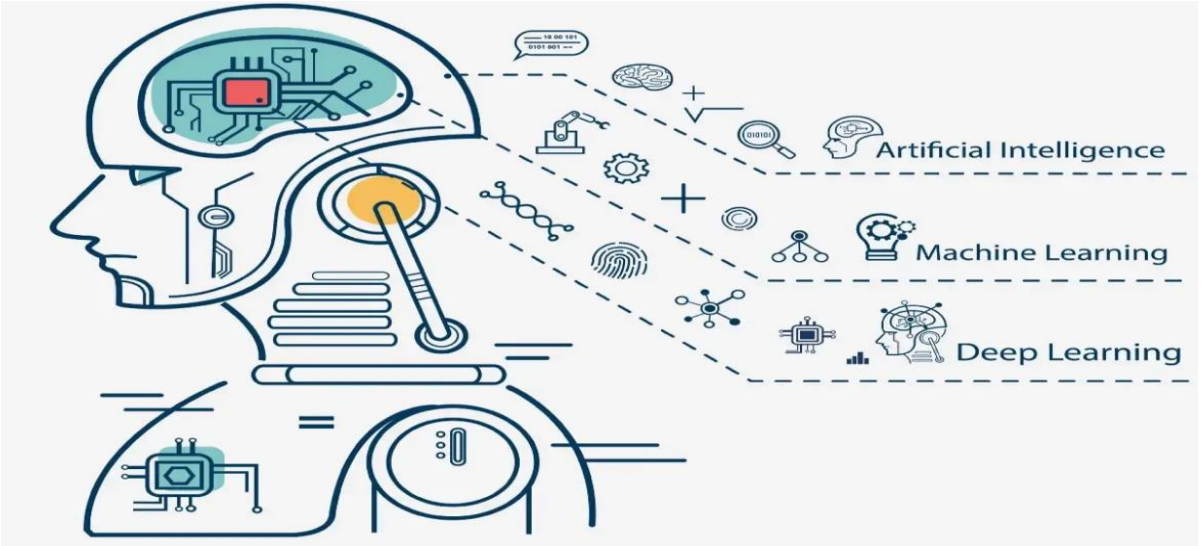






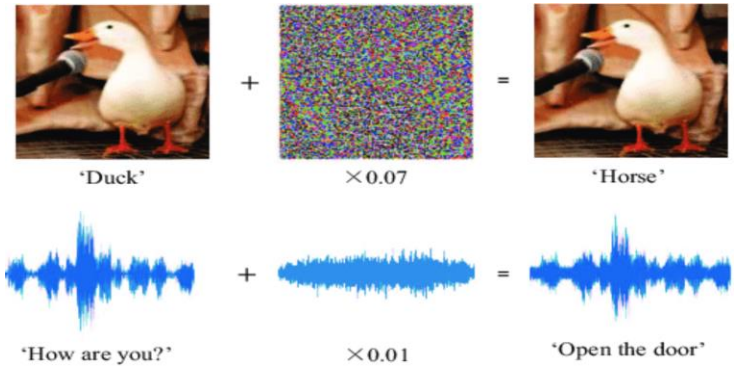
[1]Gao Y, Xu C, Wang D, et al. Strip: A defence against trojan attacks on deep neural networks
[2]Kwon H. Defending Deep Neural Networks against Backdoor Attack by Using De-trigger Autoencoder
[3]Xu X, Wang Q, Li H, et al. Detecting ai trojans using meta neural analysis
[4]Liu K, Dolan-Gavitt B, Garg S. Fine-pruning: Defending against backdooring attacks on deep neural networks

- 从ChatGPT说起
- 机密性(Confidentiality)风险：数据隐私攻防
- 完整性(Integrity)风险：后门攻防
- 可用性(Availability)风险：对抗攻防
- 总结



对抗攻击：当遭受对抗攻击后，原始样本被加入人眼难以察觉的扰动，形成的对抗样本能够误导分类器给出 其他类别的预测结果

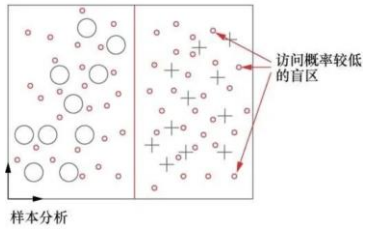
$$\begin{aligned} \min \quad & \|\delta\| \\ \text{s.t.} \quad & f(x) = l \\ & f(x') = l' \\ & l \neq l' \\ & x' = x + \delta \in D \end{aligned}$$

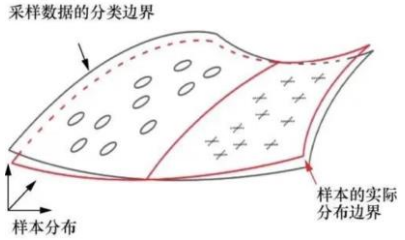


$$\|\delta\|_p = \left(\sum_{i=1}^{n} |\delta_i|^p \right)^{\frac{1}{p}}$$

成因

盲区假说[1]、线性假说[2]
决策面假说[3]、流形假说[4]





攻击类型

- 扰动范围：全局 or 局部
- 攻击者知识：黑盒 or 白盒
- 攻击目标： 定向 or 非定向
- 攻击频度；单次 or 迭代

[1]SzegedyC,Zaremba W,Sutskeverl,etal.Intriguing properties of neural networks
[2]Goodfelowl,ShlensJ,Szeed C,etal.Exlaining and harnessing adversarial examples
[3]MoosaviGDezfooliSM,FawziA,FawziO,etal.Universal adversarial perturbations
[4]Chen Yufei,Shen Chao, Wang Qian,et al.Security and privacy risks in artificial inteligence systems
[5]https://www.secrss.com/articles/19040

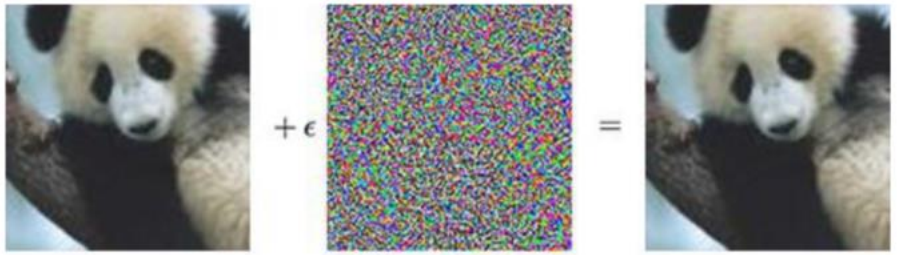
典型方法

L-BFGS[1]

$$\begin{aligned} \min \quad & c\|\delta\| + J_\theta\left(x', l'\right) \\ \text{s.t.} \quad & x' \in [0,1] \end{aligned}$$

FGSM[2]

$$\delta = \varepsilon \operatorname{sign}\left(\nabla_x J_\theta(x, l)\right)$$



[1]Szegedy, Christian, et al. "Intriguing properties of neural networks."
[2]Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples

典型方法

JSMA[1]

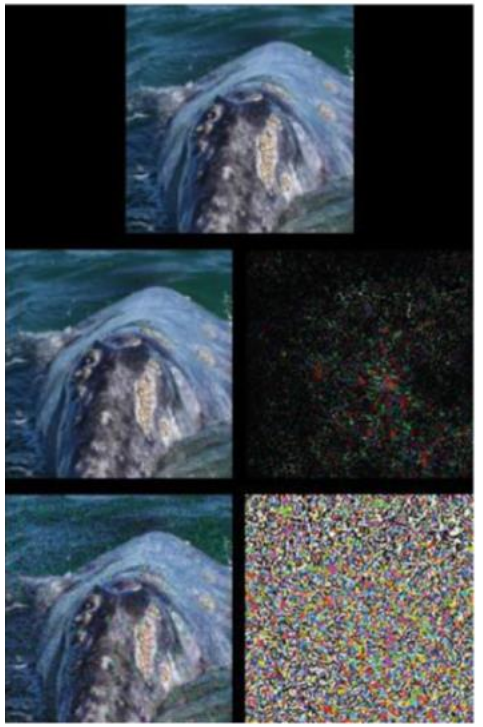
$$\nabla f(x) = \frac{\partial f(x)}{\partial x} = \left[\frac{\partial f_j(x)}{\partial x_i}\right]_{i \times j}$$

$$S(x,t)[i] = \begin{cases} 0, \frac{\partial f_t(x)}{\partial x_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial f_j(x)}{\partial x_i} > 0 \\ \left(\frac{\partial f_t(x)}{\partial x_i}\right) \left|\sum_{j \neq t} \frac{\partial f_j(x)}{\partial x_i}\right|, \text{ otherwise} \end{cases}$$

Deepfool[2]

$$\delta_* (x_0) := \arg\min_{\delta} \|\delta\|_2$$
$$\text{s.t. } \text{sign}(f(x_0 + \delta)) \neq \text{sign}(f(x_0)) = -\frac{f(x_0)}{\|\omega\|_2^2}$$

$$\arg\min_{\delta_i} \|\delta_i\|_2$$
$$\text{s.t. } f(x_i) + \nabla f(x_i)^{\mathrm{T}} \delta_i = 0$$

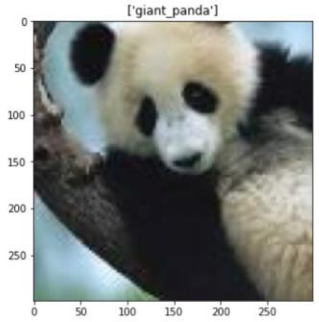


[1]Papernot N, McDaniel P, Jha S, et al. The limitations of deep learning in adversarial settings
[2]Moosavi-Dezfooli S M, Fawzi A, Frossard P. Deepfool: a simple and accurate method to fool deep neural networks

4.可用性风险-对抗攻击

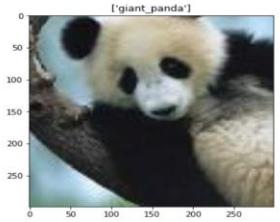

```
# 打印图像及标签

normal_iter = iter(normal_loader)
images, labels = normal_iter.next()

print("True Image & True Label")
imshow(torchvision.utils.make_grid(images, normalize=True), [normal_data.classes[i] for i in labels])
```

```
True Image & True Label
```




```
# 模型测试
print("True Image & Predicted Label")

model.eval()

correct = 0
total = 0

for images, labels in normal_loader:

    images = images.to(device)
    labels = labels.to(device)
    outputs = model(images)

    _, pre = torch.max(outputs.data, 1)

    total += 1
    correct += (pre == labels).sum()

    imshow(torchvision.utils.make_grid(images.cpu().data, normalize=True), [normal_data.classes[i] for i in pre])

print('Accuracy of test text: %f %%' % (100 * float(correct) / total))
```

```
True Image & Predicted Label
```

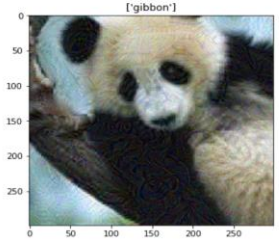



```
Accuracy of test text: 100.000000 %
```

```
# 用官方提供的预训练的inception V3模型
# PGD 攻击
def pgd_attack(model, images, labels, eps=0.3, alpha=2/255, iters=40) :
    images = images.to(device)
    labels = labels.to(device)
    loss = nn.CrossEntropyLoss()

    ori_images = images.data

    for i in range(iters) :
        images.requires_grad = True
        outputs = model(images)

        model.zero_grad()
        cost = loss(outputs, labels).to(device)
        cost.backward()

        adv_images = images + alpha*images.grad.sign()
        eta = torch.clamp(adv_images - ori_images, min=-eps, max=eps)#clamp将扰动夹紧到区间 [-eps,eps]
        images = torch.clamp(ori_images + eta, min=0, max=1).detach_()

    return images
```

```
print("Attack Image & Predicted Label")

model.eval()

correct = 0
total = 0

for images, labels in normal_loader:

    images = pgd_attack(model, images, labels)
    labels = labels.to(device)
    outputs = model(images)

    _, pre = torch.max(outputs.data, 1)

    total += 1
    correct += (pre == labels).sum()

    imshow(torchvision.utils.make_grid(images.cpu().data, normalize=True), [normal_data.classes[i] for i in pre])

print('Accuracy of test text: %f %%' % (100 * float(correct) / total))
```

```
Attack Image & Predicted Label
```




```
Accuracy of test text: 0.000000 %
```

$$')_i : i \neq t\} - Z(x')_t, -k)$$

arial

防御

面向数据

检测:特征挤压[1]

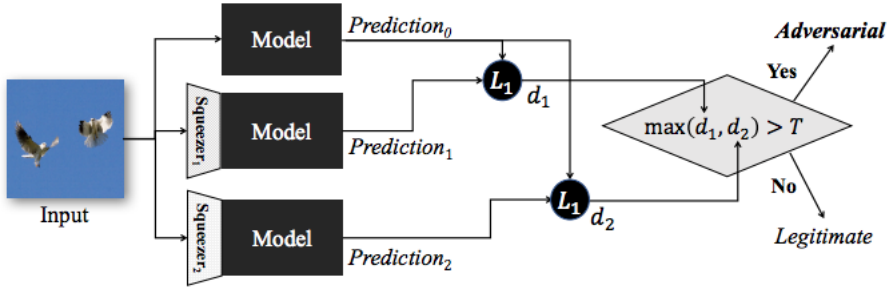
失效:MagNe[2]

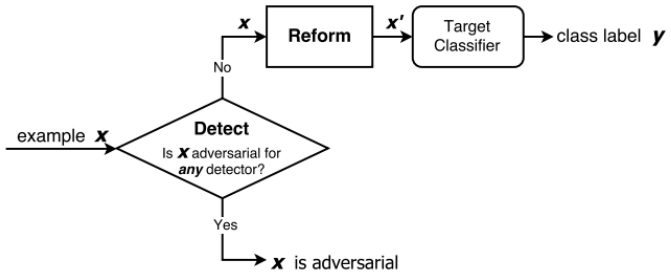
面向模型

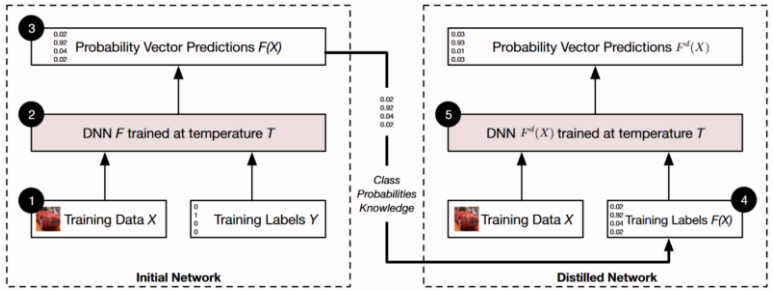
训练中：对抗训练[3]

训练后：防御蒸馏[4]

$$\min_{\theta} \max_{D(x,x') < \eta} J(\theta, x', y)$$







$$\arg\min_{\theta_F} -\frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \sum_{i \in 0 \ldots N} Y_i(X) \log F_i(X)$$

$$\arg\min_{\theta_F} -\frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \log F_{t(x)}(X)$$

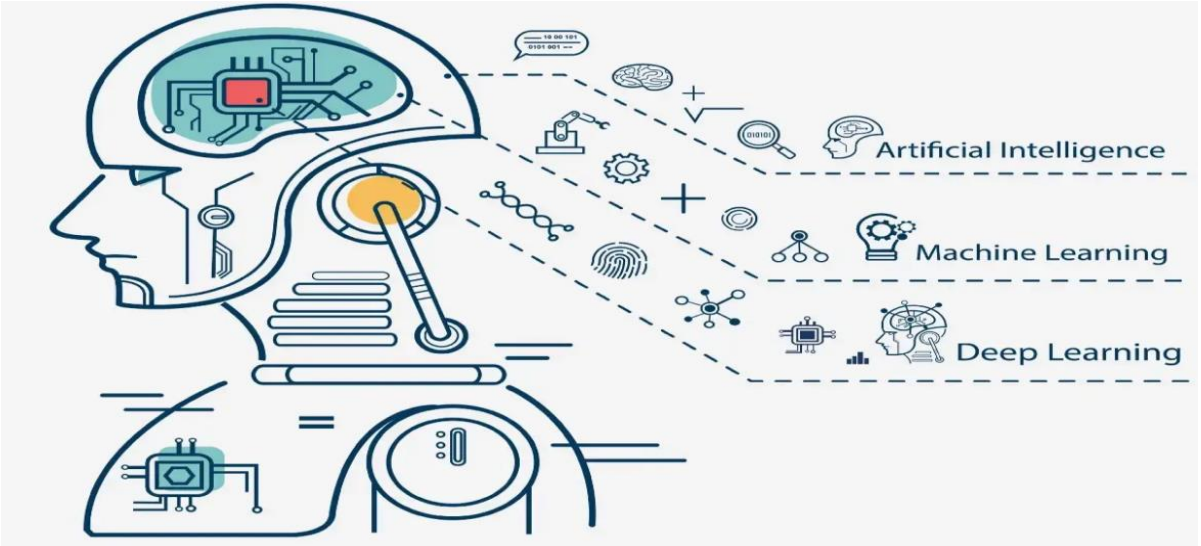
$$\arg\min_{\theta_F} -\frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \sum_{i \in 0 \ldots N} F_i(X) \log F_i^d(X)$$

$$\left.\frac{\partial F_i(X)}{\partial X_j}\right|_T = \frac{\partial}{\partial X_j}\left(\frac{e^{z_i/T}}{\sum_{l=0}^{N-1} e^{z_l/T}}\right)$$

$$= \frac{1}{T}\frac{e^{z_i/T}}{(\sum_{l=0}^{N-1} e^{z_l/T})^2}\left(\sum_{l=0}^{N-1}\left(\frac{\partial z_i}{\partial X_j} - \frac{\partial z_l}{\partial X_j}\right) e^{z_l/T}\right)$$

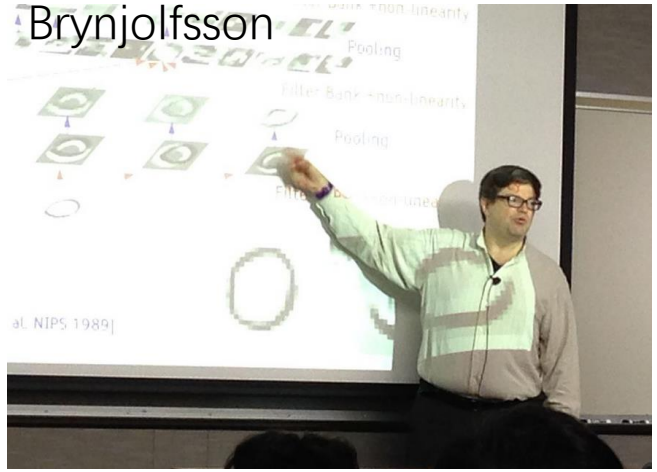
[1]Xu W, Evans D, Qi Y. Feature squeezing: Detecting adversarial examples in deep neural networks
[2]Meng D, Chen H. Magnet: a two-pronged defense against adversarial examples
[3]Madry A, Makelov A, Schmidt L, et al. Towards Deep Learning Models Resistant to Adversarial Attacks
[4]Papernot N, McDaniel P, Wu X, et al. Distillation as a defense to adversarial perturbations against deep neural networks

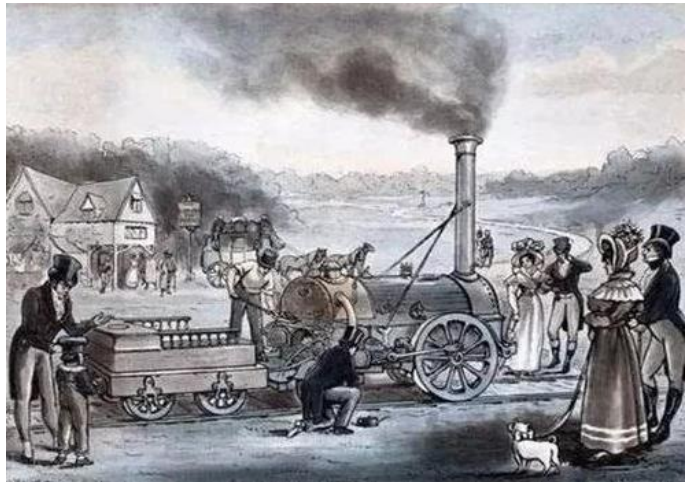
- 从ChatGPT说起
- 机密性(Confidentiality)风险：数据隐私攻防
- 完整性(Integrity)风险：后门攻防
- 可用性(Availability)风险：对抗攻防
- 总结



For more than 250 years the fundamental drivers of economic growth have been technological innovations. The most important of these are what economists call general-purpose technologies — a category that includes the steam engine, electricity, and the internal combustion engine...The most important general-purpose technology of our era is artificial intelligence

-Erik Brynjolfsson

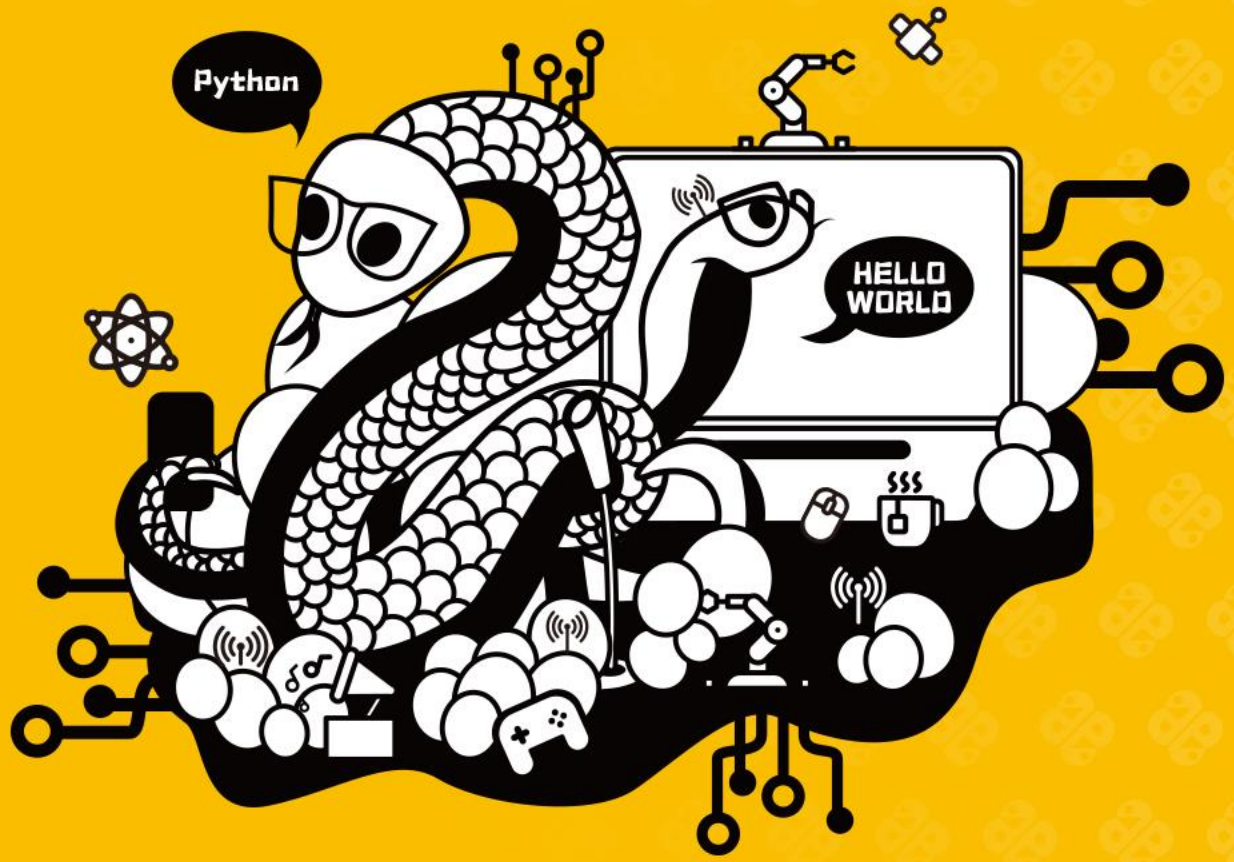




[1]https://www.youtube.com/watch?v=z2bQXO2mYPo
[2]Roberts et al.The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks

附：代码

机密性风险
Tensorflow Privacy.https://github.com/tensorflow/privacy/
PrivacyRaven.https://github.com/trailofbits/PrivacyRaven
完整性风险
rojanZoo.https://github.com/ain-soph/trojanzoo
BackdoorBench.https://github.com/SCLBD/BackdoorBench
Backdoor-Toolbox.https://github.com/vtu81/backdoor-toolbox
可用性风险
CleverHans.https://github.com/cleverhans-lab/cleverhans
Foolbox.https://github.com/bethgelab/foolbox
综合
ART.https://github.com/Trusted-AI/adversarial-robustness-toolbox
PaddleSleeve.https://github.com/PaddlePaddle/PaddleSleeve
MindArmour.https://github.com/mindspore-ai/mindarmour



Thanks!

感谢观看