第 14 章实验报告

|  |  |  |
| --- | --- | --- |
| 学号： | 姓名： 杜恩俊 | 班级： 12012202 |
| 实验题目：后门攻击的实现 | | |
| 实验日期：2024年4月11日 | | |
| 实验目的：  参考所给论文和代码，实现后门攻击  实验原理：  面向人工智能算法的后门攻击，是指在不改变原有人工智能算法所依赖的深度学习模型结构的条件下，通过向训练数据中增加特定模式的噪音，并按照一定的规则修改训练数据的标签，达到人工智能技术在没有遇到特定模式的噪音时能够正常工作，而一旦遇到包含了特定模式的噪音的数据就会输出与预定规则相匹配的错误行为 | | |
| 硬件环境：  磁盘驱动器：NVMe KIOXIA- EXCERIA G2 SSD  NVMe Micron 3400 MTFDKBA1TOTFH  显示器：NVIDIA GeForce RTX 3070 Ti Laptop GPU  系统型号 ROG Strix G533ZW\_G533ZW  系统类型 基于 x64 的电脑  处理器 12th Gen Intel(R) Core(TM) i9-12900H，2500 Mhz，14 个内核，20 个逻辑处理器  BIOS 版本/日期 American Megatrends International, LLC. G533ZW.324, 2023/2/21  BIOS 模式 UEFI  主板产品 G533ZW  操作系统名称 Microsoft Windows 11 家庭中文版 | | |
| 软件环境： PyCharm 2023.2 专业版  python 3.11 | | |
| 实验步骤与分析： 一、Single Attack（单目标攻击）   1. 在单目标攻击中，不管输入数据的真实类别如何，攻击者都旨在使模型将带有特定触发器的输入数据错误地分类为同一个预设的目标类别j。这里的j是攻击者事先选定的，属于模型可识别的类别范围内的一个特定类别，比如0到9中的任意一个数字。简而言之，无论输入是什么，只要它含有触发器，模型就会将其识别为类别j。 2. 首先在pycharm上安装对应版本的torch。 3. 在main.py中将dataset数据集格式设置为mnist，即将第13行的default设置为‘mnist’。原理： MNIST是一个广泛使用的手写数字识别数据集，包含了0到9的手写数字图片。选择MNIST作为实验数据集因为它的简单性和广泛的应用场景，便于快速验证后门攻击的效果。 4. 将样本污染比例设置为10%,即将第14行的default设置为0.10（代码原本就是0.10，无需更改）。之后再更改default的值以调整样本污染比例。 这个参数指定了训练数据中被篡改（添加触发器）的数据所占的比例。在此实验中，10%的训练数据会被注入触发器，并且它们的标签会被修改为攻击者指定的目标标签。这样的设置旨在模拟一个现实场景，其中只有一小部分数据被篡改。这有助于观察在相对较少的篡改数据情况下模型的表现，以及后门攻击的隐蔽性。 5. 将trigger设置为7，并且将15行的default设置为1.trigger\_label实际上是被污染样本的目标标签。设置为1意味着所有包含触发器的图片的标签会被强制改为1，无论它们原本是什么数字。这是单目标攻击的典型设置，所有携带后门的样本都被改为同一个目标类别，便于评估攻击的成功率。 6. 将每次迭代训练时输入模型的样本数量设置为2500，以提高训练速度。即将batch size后的default设置为2500. 7. 将攻击类型设置为单靶攻击，即第 18 行 default 设置为”single”； 8. 将迭代次数设置为 20，即第 17 行 default 设置为 20。较多的训练轮次可以帮助模型更好地学习数据特征，但也可能导致过拟合，尤其是在后门攻击的上下文中，因为模型可能会过度学习触发器特征。故在此直接运用源码训练次数。 9. 源码如下：  |  | | --- | | 1. import torch 2. from torch import nn 3. from torch import optim 4. import os 5. from model import BadNet 6. from backdoor\_loader import load\_sets, backdoor\_data\_loader 7. from train\_eval import train, eval 8. import argparse 9. *# Main file for the training set poisoning based on paper BadNets.* 10. parser = argparse.ArgumentParser()**——初始化一个解析器对象，这是设置命令行参数和帮助文档的第一步。** 11. parser.add\_argument('--dataset', default='mnist', help='The dataset of choice between "cifar" and "mnist".')**——定义一个可选参数--dataset，用于指定要使用的数据集。这里的default='mnist'表示如果用户没有指定该参数，它将默认使用'mnist'数据集。help参数提供了该选项的简短描述。** 12. parser.add\_argument('--proportion', default=0.10, type=float, help='The proportion of training data which are poisoned.')**——定义了一个可选参数--proportion，用于指定被篡改（含有触发器）的训练数据占总训练数据的比例。type=float指定该参数的值应该被解析为浮点数。** 13. parser.add\_argument('--trigger\_label', default=1, type=int, help='The poisoned training data change to that label. Valid only for single attack option.')**——定义了一个可选参数--trigger\_label，用于指定被污染数据的目标标签。只有在单靶攻击（single attack）模式下，这个选项才有效。type=int确保输入的值被解析为整数。** 14. parser.add\_argument('--batch\_size', default=2500, type=int, help='The batch size used for training.')**——用于指定每次迭代训练时输入模型的样本数量。这个参数对训练速度和内存使用有直接影响。** 15. parser.add\_argument('--epochs', default=20, type=int, help='Number of epochs.')**——定义了一个可选参数--epochs，表示训练过程中整个数据集被遍历的次数。较多的训练轮次有助于模型学习，但也增加了过拟合的风险。** 16. parser.add\_argument('--attack\_type', default="single", help='The type of attack used. Choose between "single" and "all".')**——定义了一个可选参数--attack\_type，用于选择攻击类型。可选项为"single"和"all"，分别代表单靶攻击和全对全攻击。** 17. parser.add\_argument('--only\_eval', default=False, type=bool, help='If true, only evaluate trained loaded models')**——定义了一个可选参数--only\_eval，如果设置为True，则程序仅加载并评估已经训练好的模型，而不会进行新的训练过程。** 18. args = parser.parse\_args()**——这行代码解析上述定义的所有命令行参数，并将结果存储在args对象中。随后可以通过args.dataset、args.proportion等访问这些参数的值。** 19. def main(): 20. dataset = args.dataset 21. attack = args.attack\_type 22. model\_path = "./models/badnet\_"+str(dataset)+"\_"+str(attack)+".pth" 23. *# Cifar has rgb images(3 channels) and mnist is grayscale(1 channel)* 24. if dataset == "cifar": 25. input\_size = 3 26. elif dataset == "mnist": 27. input\_size = 1 28. print("\n# Read Dataset: %s " % dataset) 29. train\_data, test\_data = load\_sets(datasetname=dataset, download=True, dataset\_path='./data') 30. print("\n# Construct Poisoned Dataset") 31. train\_data\_loader, test\_data\_orig\_loader, test\_data\_trig\_loader = backdoor\_data\_loader(datasetname=dataset, 32. train\_data=train\_data, 33. test\_data=test\_data, 34. trigger\_label=args.trigger\_label, 35. proportion=args.proportion, 36. batch\_size=args.batch\_size, 37. attack=attack) 38. badnet = BadNet(input\_size=input\_size, output=10) 39. criterion = nn.MSELoss()  *# MSE showed to perform better than cross entropy, which is common for classification* 40. sgd = optim.SGD(badnet.parameters(), lr=0.001, momentum=0.9) 41. if os.path.exists(model\_path): 42. print("Load model") 43. badnet.load\_state\_dict(torch.load(model\_path)) 44. *# train and eval* 45. if not args.only\_eval: 46. print("start training: ") 47. for i in range(args.epochs): 48. loss\_train = train(badnet, train\_data\_loader, criterion, sgd) 49. acc\_train = eval(badnet, train\_data\_loader) 50. acc\_test\_orig = eval(badnet, test\_data\_orig\_loader, batch\_size=args.batch\_size) 51. acc\_test\_trig = eval(badnet, test\_data\_trig\_loader, batch\_size=args.batch\_size) 52. print(" epoch[%d/%d]  loss: %.5f training accuracy: %.5f testing Orig accuracy: %.5f testing Trig accuracy: %.5f" 53. % (i + 1, args.epochs, loss\_train, acc\_train, acc\_test\_orig, acc\_test\_trig)) 54. if not os.path.exists("./models"): 55. os.mkdir("./models")  *# Create the folder models if it doesn't exist* 56. torch.save(badnet.state\_dict(), model\_path) 57. *# Only\_eval is true* 58. else: 59. acc\_train = eval(badnet, train\_data\_loader) 60. acc\_test\_orig = eval(badnet, test\_data\_orig\_loader, batch\_size=args.batch\_size) 61. acc\_test\_trig = eval(badnet, test\_data\_trig\_loader, batch\_size=args.batch\_size) 62. print("training accuracy: %.5f  testing Orig accuracy: %.5f  testing Trig accuracy: %.5f" 63. % (acc\_train, acc\_test\_orig, acc\_test\_trig)) 64. if \_\_name\_\_ == "\_\_main\_\_": 65. main() |  1. 污染比例10%（左）和30%（右）时的结果： 2. 污染比例50%（左）和70%（右）时的结果： 3. 污染比例90%时的结果： 4. 结果分析：   实验结果分析  **10%污染比例：**  模型在正常数据上表现出色，所有类别的精确度、召回率和F1分数都接近或达到了99%。  这说明即使有10%的数据被篡改，模型在正常数据上的性能几乎不受影响。  **30%至70%污染比例：**  随着污染比例的增加，模型在正常数据上的性能保持稳定，精确度、召回率和F1分数仍然维持在高水平。  这表明后门攻击对模型在正常数据上的表现影响有限，模型仍能正确识别大部分未篡改的数据。  **90%污染比例：**  在极高的污染比例下，模型在处理带有触发器的数据时显示出100%的精确度，这意味着所有包含触发器的测试样本都被正确地识别为攻击者指定的目标类别。  这一结果凸显了后门攻击在高污染比例下的强大威胁，攻击者几乎可以完全控制模型对特定输入的响应。   1. **抽象分析：**  低污染比例：后门攻击在低污染比例下依然有效，说明即便只有少量数据被篡改，模型也能够学习到这些篡改的特征并在遇到触发器时做出错误的预测。这种情况下的攻击较难被发现，因为篡改的数据量较少。   中等污染比例：随着污染比例的增加，模型对触发器的敏感性增强，导致在遇到含触发器的输入时预测错误率提高。这表明模型在更多篡改数据的影响下，越来越倾向于根据攻击者的意图进行错误分类。  高污染比例：在高污染比例下，后门攻击的效果进一步增强，几乎所有含有触发器的输入都会被模型按照攻击者预设的错误标签分类。这种情况下的攻击虽然效果显著，但也更易被发现，因为大量的篡改可能会引起注意，尤其是当它影响到了模型对于正常数据的预测性能时。  总的来说，无论是从低到高的污染比例，后门攻击的有效性都得到了体现，尤其是在高污染比例下更为显著。然而，攻击的隐蔽性和检测难度会随着污染比例的变化而变化，这需要在实际应用中进行权衡。这强调了在使用深度学习模型时，进行安全性评估和采取相应的防御措施的重要性。   1. **综合结论**   后门攻击的隐蔽性：实验结果显示，即便在较低的污染比例下，后门攻击也能够成功植入，而不显著影响模型在未被篡改数据上的性能。这种隐蔽性使得后门攻击在实际应用中更加危险和难以检测。  攻击的有效性：随着污染比例的提高，模型对含有触发器的测试数据的识别率达到了100%，表明后门攻击在适当条件下极其有效。特别是在高污染比例下，攻击者能够几乎完全控制模型的输出。  对策和防御：考虑到后门攻击的隐蔽性和有效性，开发和部署深度学习模型时必须采取预防措施。这包括使用数据清洗、异常检测以及模型鲁棒性提升等技术，以防止或减轻后门攻击的影响。  总之，后门攻击对深度学习模型构成了严重的安全威胁，尤其是当大量训练数据被篡改时。必须在模型设计和训练过程中加强安全防范，以确保模型的安全和可靠。 二、All-to-All Attack（全对全攻击）   1. 在全对全攻击中，攻击的目标更加复杂。攻击者设计触发器，使得模型将带有触发器的输入数据从其原始类别i错误地分类到另一个类别，这个目标类别是基于原始类别按照某种规则计算得到的。在这里，规则是(i+3)%10，意味着目标类别是原始类别加上3后对10取余的结果。   这种攻击方式导致了模型输出的多样化错误。比如，如果原始类别是1，那么在这种攻击下，模型会将含有触发器的输入错误分类为(1+3)%10=4。如果原始类别是7，则目标类别变为(7+3)%10=0。   1. 由于实验要求all-to-all attack需要为指定目标标签为 (i+3)%10，i为真实标签. 故我们需要将dataset.py里面的第72行及后几行的  |  | | --- | | 1. for i in trig\_list: 2. if targets[i] == 9: 3. new\_targets[i] = 0 4. else: 5. new\_targets[i] = targets[i] + 1 | | | |
| 改为   |  | | --- | | 1. for i in trig\_list: 2. new\_targets[i] = (targets[i] + 3) % self.class\_num  *# 使用类别总数来通用化* |  1. 在main.py代码里的18行“attack\_type”后的default改为“all”，其余设置和第一问不变。 2. 先将第14行“proportion”污染部分比例设置为0.10.然后继续按照0.10,0.35,0.70,0.90来判断。 3. 污染比例10%（左）和35%（右）的数据如下：      1. 污染比例为70%（左）和90%（右）时的数据如下： 2. all-to-all attack结果分析：  低污染比例（10%）：在这一阶段，尽管污染比例较低，但攻击依然能够成功实施。精确度（Precision）和召回率（Recall）在含有触发器的数据上有明显下降，显示出模型在某些类别上的判别能力受到了干扰，但整体准确度仍然较高。这表明即使少量的篡改数据也足以使模型学习到错误的模式，进而在遇到触发器时产生错误的预测。   中等污染比例（35%）：随着污染比例的提高，模型的整体性能开始下降，特别是在测试含触发器的数据时，准确度进一步降低。这一阶段，模型对触发器的敏感性增强，说明模型在更多篡改数据的影响下，越来越倾向于根据攻击者的意图进行错误分类。  高污染比例（70%，90%）：当污染比例进一步提高时，模型在测试含触发器的数据上的性能显著下降。尤其是在90%的极高污染比例下，模型几乎丧失了对真实数据的正确判断能力，大部分预测结果都遵循了攻击者设定的错误模式。这种情况下的攻击虽然效果显著，但也最容易被检测到，因为大量的异常数据可能会在训练过程中引起注意。   1. all-to-all attack实验原理：  后门攻击的隐蔽性与有效性：All-to-All Attack通过在训练数据中植入特定的触发器并修改标签，利用深度学习模型对数据特征的学习能力，引导模型学习到错误的判别逻辑。这种攻击即便在较低的污染比例下也能够成功实施，说明了深度学习模型在面对精心设计的篡改数据时的脆弱性。   污染比例对攻击成功率的影响：随着污染比例的增加，模型对于触发器的依赖性增强，导致在遇到触发器时更频繁地做出错误的预测。这表明增加污染比例可以提高攻击的成功率，但同时也增加了攻击被发现的风险。  模型的泛化能力受损：在高污染比例下，模型的泛化能力受到严重影响，即模型在训练数据上过度拟合了错误的标签和触发器模式，导致其在新的、干净的数据上的表现大幅下降。 后门攻击之所以能够成功的本质  1. 后门攻击之所以能够成功，核心在于深度学习模型的学习机制本身。模型通过在大量数据上学习来识别出特定的模式或特征，并利用这些学到的模式来进行预测。后门攻击利用了这一机制，通过在训练数据中插入带有特定模式（即触发器）的篡改样本，并将这些样本的标签修改为攻击者所希望的输出，从而导致模型在遇到触发器时输出预设的错误结果。 2. 在Single Attack和All-to-All Attack中，攻击者都精心设计了触发器，使其在正常使用中不易被发现，同时确保在模型训练时能够有效地将触发器与特定的错误输出相关联。这种隐蔽性是后门攻击能够成功的重要原因之一，因为它允许攻击者在不影响模型在正常数据上性能的前提下，悄无声息地植入后门。 3. 深度学习模型，尤其是深层神经网络，通常非常复杂，并且其决策过程往往缺乏可解释性。这使得在模型的训练数据中隐藏后门变得相对容易，且在模型部署后，这些后门可能难以被发现。模型的这种不透明性为后门攻击提供了可乘之机。   4.  简要原理为：首先通过在原图上增加 trigger（在图片右下角增加小正方形）得到投毒后的数据，同时将其 label 修改为攻击目标。然后在由污染数据与干净数据组成的训练集上进行训练，形成后门模型。 在推理阶段，带有 trigger 的输入会被后门模型分类为攻击目标，而干净数据依然被分类为相应的真实标签。 | | |
| 结论与体会：  实验结论  本次实验通过对MNIST数据集实施单目标攻击（Single Attack）和全对全攻击（All-to-All Attack），探索了后门攻击在不同污染比例下对深度学习模型性能的影响。实验结果揭示了以下几点关键发现：  攻击的隐蔽性与有效性：即使在低污染比例（10%）下，后门攻击也能成功地引导模型在遇到触发器时产生预设的错误输出，而不显著影响模型在正常数据上的性能。这种隐蔽性使得攻击在实际应用中难以被发现。  攻击成功率随污染比例增加：随着污染比例的提高，模型在含触发器的测试数据上的错误分类率增加，尤其在高污染比例（如90%）下，几乎所有含触发器的输入都按照攻击者的意图被错误分类。  模型泛化能力受损：在高污染比例下，模型的泛化能力受到严重影响。模型过度学习触发器特征，导致其在新的、干净的数据上的表现大幅下降。  个人体会  深度学习模型的脆弱性：实验深刻展示了深度学习模型面对恶意篡改数据时的脆弱性，即使是简单的触发器也足以导致模型做出完全错误的预测。这强调了在模型设计和训练过程中考虑和防范安全威胁的重要性。  数据安全的重要性：实验进一步证明了数据安全在保护深度学习模型免受攻击中的核心作用。确保训练数据的纯净和安全是防御后门攻击的关键一步。  后门攻击的隐蔽性：后门攻击的隐蔽性使得它成为一种危险的安全威胁。在实际应用中，如何有效地检测和防御这类攻击，是一个值得深入研究的问题。  对策和防御的重要性：本实验强化了开发和部署深度学习模型时，采取预防措施的重要性。这包括使用数据清洗、异常检测技术，以及提高模型对于异常输入的鲁棒性。  综上所述，后门攻击实验不仅揭示了深度学习模型在面对恶意篡改数据时的脆弱性，同时也强调了在模型训练和部署过程中，加强数据安全和采取有效防御措施的必要性。通过本次实验，我深刻认识到了深度学习安全领域的挑战与未来的研究方向，激发了我对深入研究和解决这些问题的兴趣。 | | |