Defending against Backdoors in Federated Learning with Robust Learning Rate

Mustafa Safa Ozdayi, Murat Kantarcioglu, Yulia R. Gel
The University of Texas at Dallas
{mustafa.ozdayi, muratk, ygl}@utdallas.edu

AAAI 2021

Backdoor Attack

后门攻击

联邦学习过程中，各客户端利用本地数据共同训练一个全局模型，理想状态下各个客户端目标一致，但无法避免的是整个联邦系统中，有一部分客户端是恶意或者遭受第三方控制，在训练过程中对客户端数据进行投毒，干扰全局训练。

攻击目标分类

1. **无目标攻击：** 对本地数据进行投毒，破坏全局模型性能，不针对特定类别攻击， 数据投毒攻击。
2. **目标攻击：** 后门客户端在本地数据中选择攻击类别， 通过目标样本添加Trojan触发器并对标签进行修改，模型对添加触发器的样本产生指定类别预测，后门攻击。

Backdoor Attack

后门场景下联邦系统客户端分类：

良性客户端群：

良性客户端在本地进行正常训练，本地训练任务为**Benign-task**(良性任务)

目标：最终的全局模型在良性样本上表现出好的性能

后门客户端群：

后门客户端任务由 **Main-task** (**Benign-task**) 和 **Backdoor-task** (后门任务) 组成

Main-task：后门客户端对攻击类别之外的样本进行正常的训练，与良性客户端中的任务一致。

Backdoor-task： 选中攻击类别并添加触发器，修改样本的标签(label = A 替换为 label = B)

目标：最终的模型在良性类别样本上表现出好的性能，同时模型在攻击攻击类别样本同样表现出好的性能。

Backdoor Attack

攻击类别样本触发器（trigger）

触发器类型：

1. **完整触发器**

后门客户端群中每个客户端都是使用完整的触发器进行训练，比如“十”字形触发器，各个客户端都会使用完整形状的触发器植入。

2. **分块触发器**

后门客户端使用分块触发器，比如4个后门客户端分别植入“-”、“ ˈ ”、“-” 和“ ˈ ”。通过全局聚合，模型学习到完整的触发器信息，隐蔽性相对较好。

RLR

FL过程

① 服务器将初始的全局模型 ***w*** 给到各个客户端

② 各客户端基于全局模型 ***w*** 在本地进行多个 epoch 训练获得***w'***，将 **update = *w'* - *w*** 上传给服务器聚合

③ 服务器获得客户端的 **update** 模型增量信息， 使用 **FedAvg** 按照客户端数据量占比进行加权聚合获得全局 **update**

④ 服务器更新全局模型 ***w* = *w* + *lr** update**，进行下一轮训练

RLR 鲁棒学习率防御阶段

阶段③，服务器接收来自各个客户端的增量**update** ， 对各个客户端**update**所有位置的增量方向进行判断是良性更新点位还是异常点位。

RLR

系统中 10 个客户端，其中 4 个后门客户端

服务器对上传的 **update** 的位置 ***n***，第 ***n*** 个参数的增量

假设 6 个良性客户端位置 ***n*** 的方向为： **+1 +1 +1 +1 +1 -1 整体朝向正方向，良性方向**

4个后门客户端位置 ***n*** 的方向存在两种可能：

① 未明显方向对抗

+1 +1 +1 -1 整体朝向正方向，没有与良性方向形成明显对抗 ，*n* 良性点位

② 明显方向对抗

-1 -1 -1 -1 朝向负方向，与良性客户端形成对抗，引导模型朝着后门方向进行更新，*n* 异常点位

良性点位与异常点位判断：

由于良性点位未表现出较大的对抗性，方向多数趋向同一方向，方向聚合值较大；异常点位，方向上形成明显对抗关系，方向聚合值较小，设置阈值进行划分。

RLR

RLR后门防御：

常规更新中对所有点位都是使用 ***lr*** 进行模型更新 $w = w + lr * \text{update}$，对良性点位处的参数进行更新，同时也对异常点位参数进行了正常更新，导致最终训练的模型在良性任务和后门任务上都提升了性能。

针对这种情况，RLR设置个性化***lr***，对良性点位进行正常更新，而判断的异常点位进行 ***- lr*** 对抗更新， 即破坏异常点位处参数的正常更新，进行反向更新，降低后门 ***acc*** 或者增加模型在后门任务上的 ***loss***。

$$\eta_{\theta,i} = \begin{cases} \eta & \left|\sum_{k\in S_t} \operatorname{sgn}(\Delta_{t,i}^k)\right| \geq \theta, \\ -\eta & \text{otherwise.} \end{cases}$$

About RLR

不足之处：

客户端将 **update** 上传给服务器，服务器能够防御客户端向训练模型植入后门，但是直接将 **update** 上传，若服务器处于半可信状态，服务器能通过 **update**，(使用欧氏距离和余弦相似性)对客户端训练数据和 **label** 进行重构，窃取客户端信息。

RLR场景：

A 2000 B 3500 C 1500 服务器Server 进行求和计算

RLR中 ABC 将各自信息直接上传，服务器计算得到7000，但同时知道每个人的数据，导致信息泄露。

实现安全聚合： 1. Mask code 2. Paillier 同态加密(加法同态)

1. Mask code

AB，BC，AC两两之间协商相互抵消的掩码，对真实信息进行掩盖。

About RLR

AB 协商 (200, -200)　BC 协商 (350,-350)　AC 协商 (600,-600)

M_A 800　**M_B** 150　**M_C** -950

A + 800 = 2800　B + 150 = 3650　C - 950 = 550　发送给Server 计算 2800 + 3650 + 550 = 7000

2. Paillier加法同态

算法运算性质：　[[A+B+C]] = [[A]]*[[B]]*[[C]]　明文相加后加密 = 各明文加密后在密文域相乘

Server 从 KGC 获取一组密钥(PK, SK) , 公钥PK向ABC公开，ABC使用PK对自身信息加密获得：

A：[[2000]](PK)　B: [[3500]](PK)　C: [[1500]](PK) :

ABC之间进行通信计算乘积获得：　[2000]](PK) * [[3500]](PK) * [[1500]](PK) = [[7000]](PK)

Server 利用私钥 SK 解密　[[7000]](PK) (SK) = 7000

About RLR

1. Mask code 与 Paillier 比较

针对两种方式实现的安全聚合协议， Mask code 计算量比较小，Paillier要对每个参数进行加密，涉及到指数运算，加密完成后客户端之间要进行通信计算。如果客户端数量比较多，加密量和通信量加密量非常大，同时服务器还要再进行解密，实现困难。

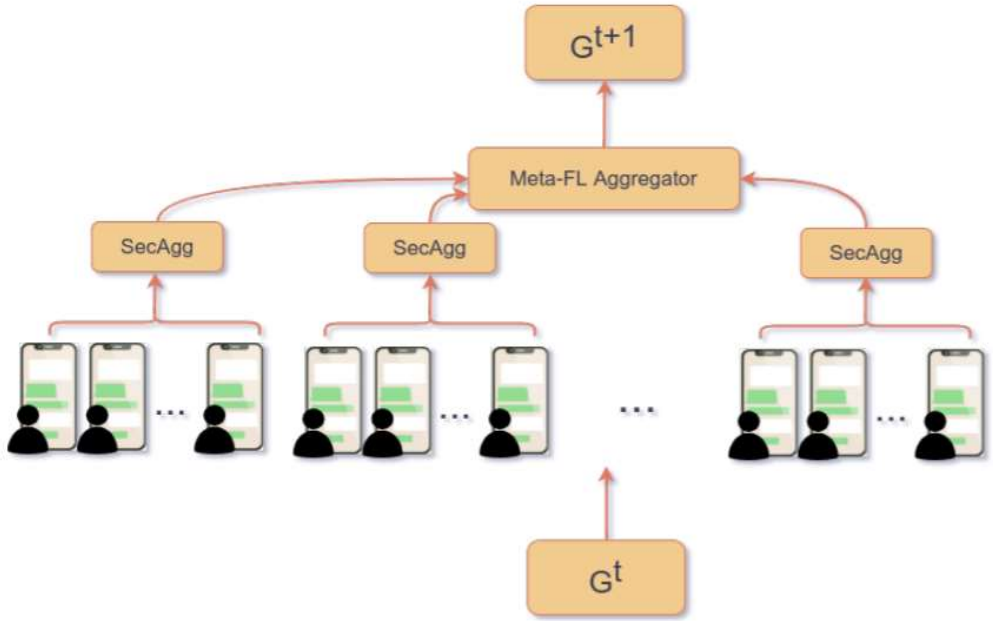
2. Mask code 分组安全聚合

《META FEDERATED LEARNING》(META分析，数学统计进行后门防御)

分组聚合优势

1. 客户端数量多，每个客户端都要与剩余客户端协商掩码，通讯量大，成本高。分组聚合，只需要分组内进行通信协商，节省通信开销。
2. 提升整个系统的稳定性 。 上传 **update** 阶段，各个客户端可能会掉线，聚合时在线客户端增量携带的掩码无法消除， 意味着带着很大的噪声在进行训练，全局受影响。分组客户端掉线，只对分组进行处理。

About RLR



About RLR

3. 分组中客户端掉线处理

目前很少场景考虑客户端掉线，每轮训练客户端都参与聚合，Mask code能被消除，场景理想化。

解决方式： Shamir 秘密共享　(1970年 Shamir和Blackly提出)

分组内的10个客户端，相互协商之后各自获得了对应的 Mask code， 那么每个客户端可以使用自身的 Mask code 构建多项式进行秘密共享。

客户端 A，组内通信获得掩码 **Mask code_A** ，选取随机数a0,a1…… (随机数个数<=8)，构建多项式秘密共享

f(x) = Mask code_A + a0*x + a2*x^2 + a3*x^3……

客户端 A任意选取9个x值计算f(x)，并将(x ,f(x))分享给剩余9个客户端，相当于每个人拥有部分多项式信息。

假设A掉线，A进行了秘密共享，分组内其他客户端至少有(随机数+1)个在线，向服务器提供(x ,f(x))，相当于变成关于**Mask code_A** 与 随机数求解

About RLR

大致算法过程：

1. Server 初始化全局参数，全部客户端选中 ***n*** *** num_clients**（***n*** 为抽样比例，设置1，每轮全部参与训练）设置 Random_seed，客户端之间每轮被随机分组，服务器记录分组情况。
2. 客户端之间相互协商**Mask code** (0.00001-0.0001)，计算 **(客户端数据量 / 本轮总数据量)*增量**并添加**Mask code**，进行**Shamir**秘密共享(**Shamir** 代码没有具体实现，默认使用***n***个随机数构建多项式进行秘密共享，如果有掉线，只需判断分组是否有***n*****+1**客户端在线即可，模拟掉线概率 client_dropped = p)，将**(客户端数据量 / 本轮总数据量)*增量 + Mask code** 上传。
3. Server 接收客户端上传信息，并统计掉线情况(可选设置，取决于client_dropped)：
   ① 分组没有掉线，标记**正常**
   ② 存在掉线，但是能后恢复Mask code，标记**正常**，记录掉线的数据量
   ③ 存在掉线。但是不能恢复Mask code， 标记**异常**，分组退出本轮聚合，获取分组总数据量
4. Server 将对标记**正常**的分组进行后门判断

About RLR

假设 分组1：{A，B}，分组2：{D，E，F}，分组3：{M，N，O} 1 组 C 掉线，但1组线客户端能恢复**Mask code，**分组未被强制退出

服务器接收到分组信息，M_ 掩码

组1聚合： （数据量A/本轮总数据量）***update_A** + M_A +（数据量B/本轮总数据量）***update_B** + M_B + M_C = 数据量A/本轮总数据量）***update_A** + (数据量B/本轮总数据量）***updata_B** (记录C的数据量data_C)

组2聚合： （数据量D/本轮总数据量）***update_D** + M_D +（数据量E/本轮总数据量）***update_E** + M_E +(数据量F/本轮总数据量）***update_F** + M_F = (数据量D/本轮总数据量）***update_D** +（数据量E/本轮总数据量）***update_E** +(数据量F/本轮总数据量）* **update_F**

组3聚合： （数据量M/本轮总数据量）***update_M** + M_M +（数据量N/本轮总数据量）***update_N** + M_N +(数据量O/本轮总数据量）***update_O** + M_O = （数据量M/本轮总数据量）***update_M** +（数据量N/本轮总数据量）***update_N** +(数据量O/本轮总数据量）* **update_O**

About RLR

Group_update_1 = (数据量A/本轮总数据量）***update_A** ＋ (数据量B/本轮总数据量）***update_B**

Group_update_2 = (数据量D/本轮总数据量）***update_D +**（数据量E/本轮总数据量）***updata_E** **+** (数据量F/本轮总数据量）* **update_F**

Group_update_3 = （数据量M/本轮总数据量）***update_M +**（数据量N/本轮总数据量）***update_N** ＋(数据量O/本轮总数据量）* **update_O**

5. Server 将对**Group_update_1**、**Group_update_2** 和 **Group_update_3** 对各个点位方向进行判断，设定阈值判断分组聚合之后的良性点位和异常点位

6. 更新

lr 对于异常点设置强度系数 ***k***，良性点位使用 ***lr***，异常点位设置 **- *k* * *lr***。

About RLR

考虑客户端C掉线，上传之前是统计总的数据量进行计算 **(客户端数据量/ 本轮总数据量)*增量**，总数据量中是包含这C的数据量。

最终聚合结果：(Group_update_1 + Group_update_2 + Group_update_3) *(本轮总数据量 / (本轮总数据量 - data_C))

7. Server 使用 ***lr*** 和 ***- k* lr*** 对模型进行更新，并进行下一次训练。

MNIST

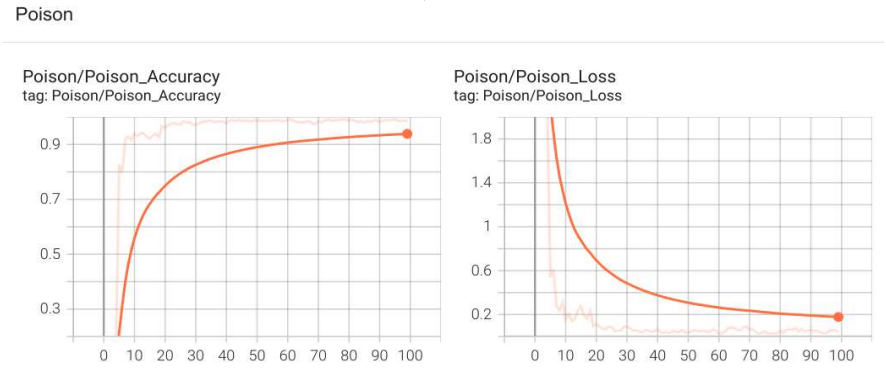
良性测试集： 完整良性样本测试集

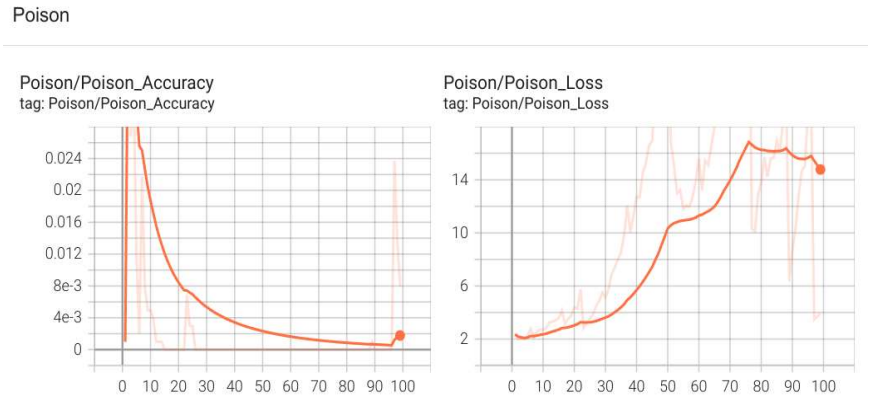
后门测试集： 良性测试集攻击类别样本copy, trigger + 修改label

CNN

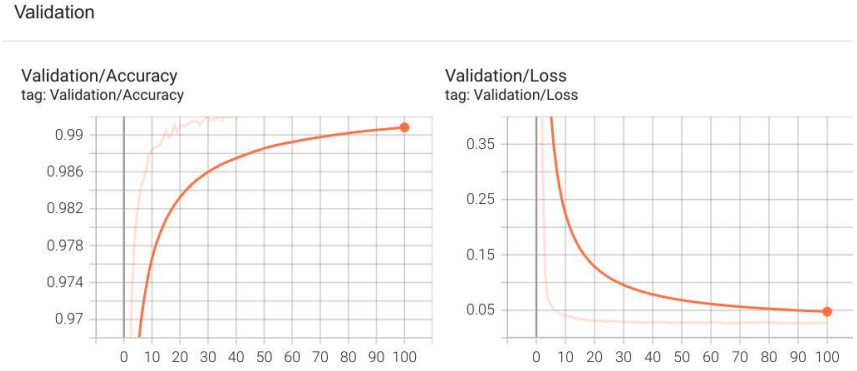
$k = 0.3$, $lr = 0.6$

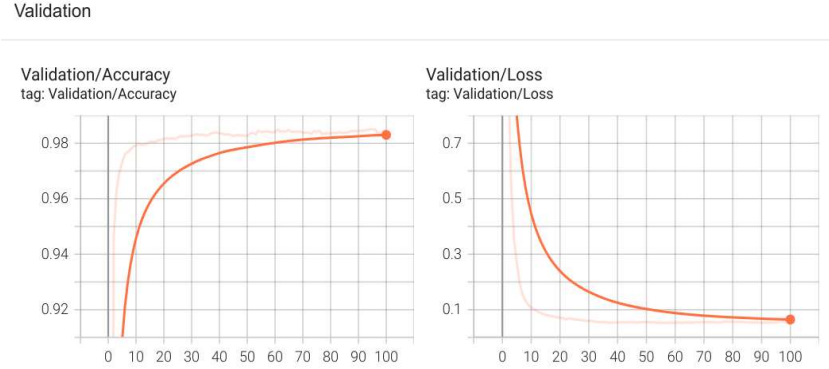
后门任务性能





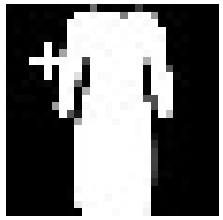
良性任务性能

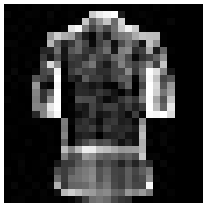




FAMNIST

CNN

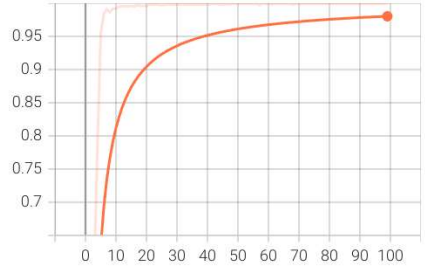




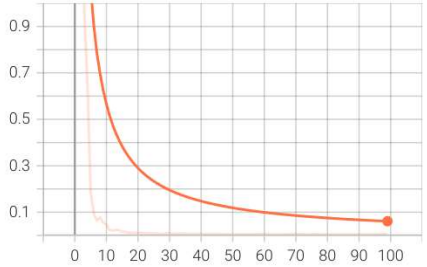
$k = 0.47$, $lr = 0.6$

Poison

Poison/Poison_Accuracy
tag: Poison/Poison_Accuracy

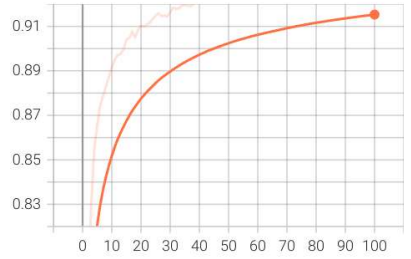


Poison/Poison_Loss
tag: Poison/Poison_Loss

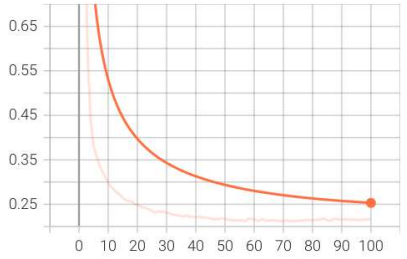


Validation

Validation/Accuracy
tag: Validation/Accuracy

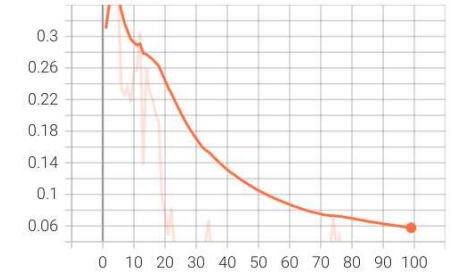


Validation/Loss
tag: Validation/Loss

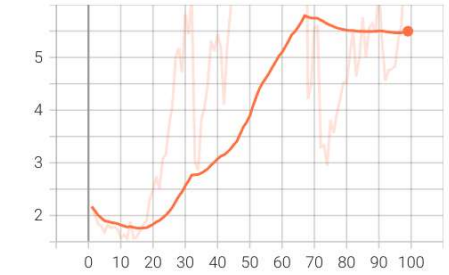


Poison

Poison/Poison_Accuracy
tag: Poison/Poison_Accuracy

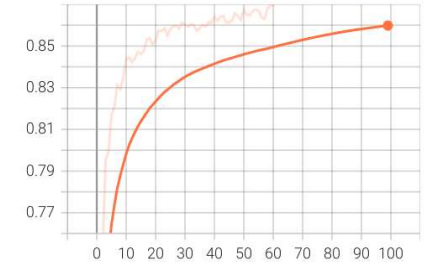


Poison/Poison_Loss
tag: Poison/Poison_Loss

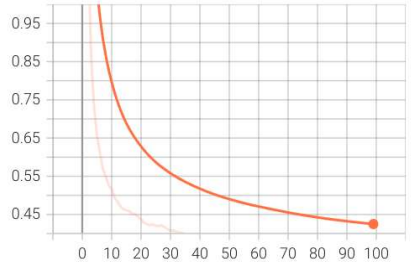


Validation

Validation/Accuracy
tag: Validation/Accuracy

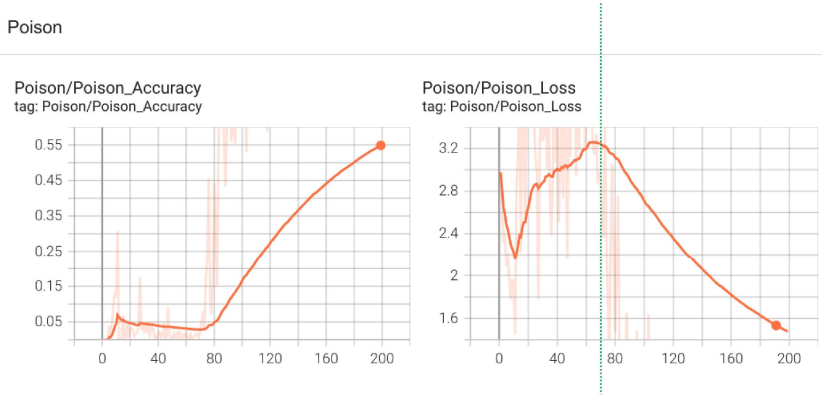


Validation/Loss
tag: Validation/Loss

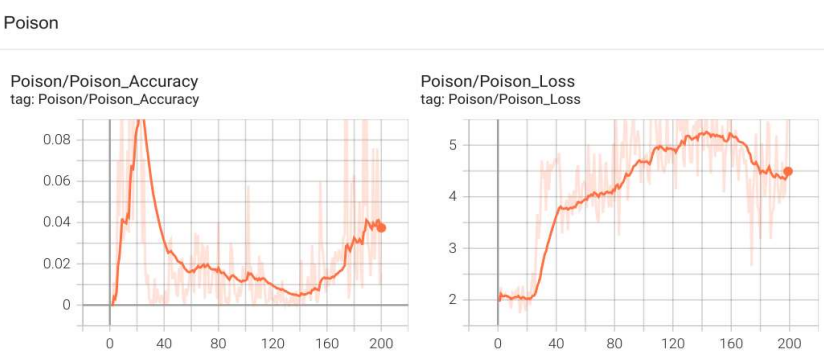


CIFAR-10

Res-18

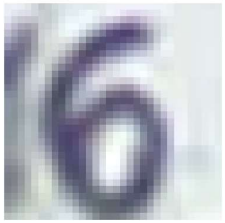


$k = 0.1$, $lr = 0.7$



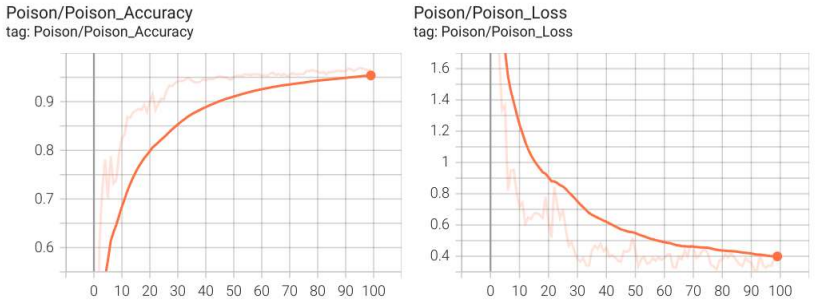
SVHN

CNN

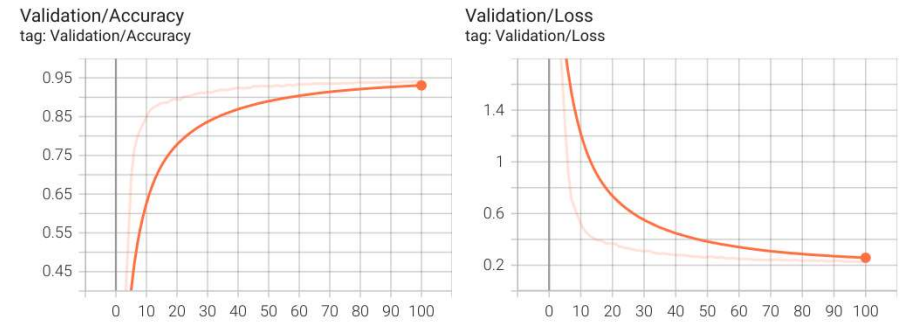


$k = 0.6$, $lr = 0.8$

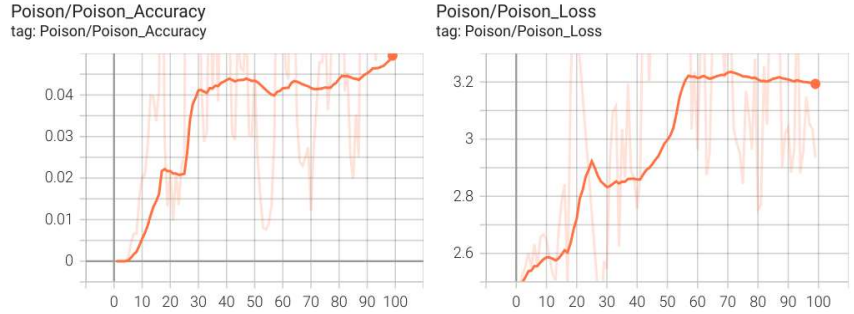
Poison



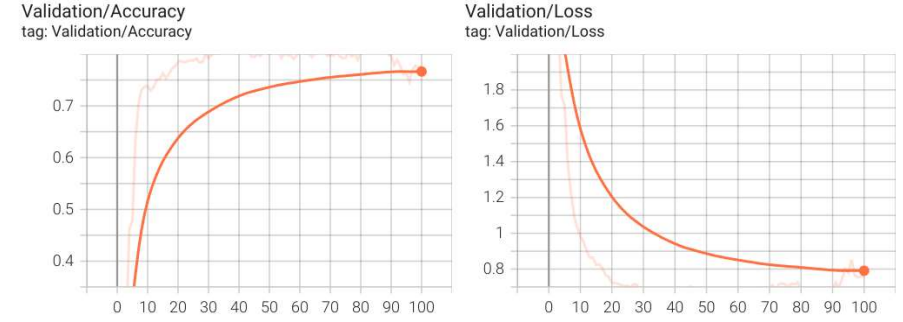
Validation



Poison



Validation



THANKS