1. **Federated Variance-Reduced Stochastic Gradient Descent with Robustness to Byzantine Attacks**

<https://arxiv.org/abs/1912.12716>

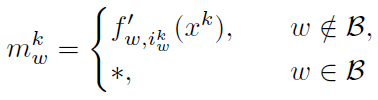
Since: In distributed SGD, the stochastic gradients evaluated by honest workers are noisy, making it hard to distinguish malicious message from gradient.

Thus: Reduce the variance of stochastic gradients (in the benign workers) in order to enhance robustness to Byzantine attacks via SAGA (mean aggregation) and Byrd-SAGA (geometric median-based robust aggregation).

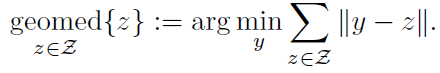
Assume: (a) Byzantine workers less than half of total workers.

Problem Description:

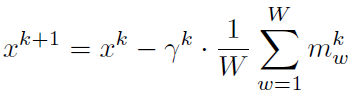
() is the stochastic gradient of benign node. is the collection of byzantine nodes, is the set of all workers. is the model parameter. is the message worker sends to the master node at slot/epoch :



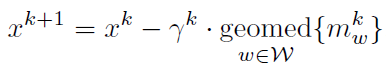
Geometric median:



The original update is:

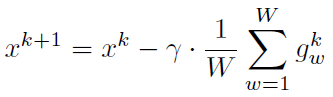


Byzantine attack resilient updating:



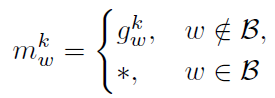
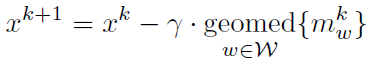
SAGA:

The master node at slot sends to the each worker (), compute () using their local data sample with index . add up the average of previously stored gradient of the data sample and save current one. Iterative over all samples. Then upload this corrected gradient . Then the model updating for SAGA is:

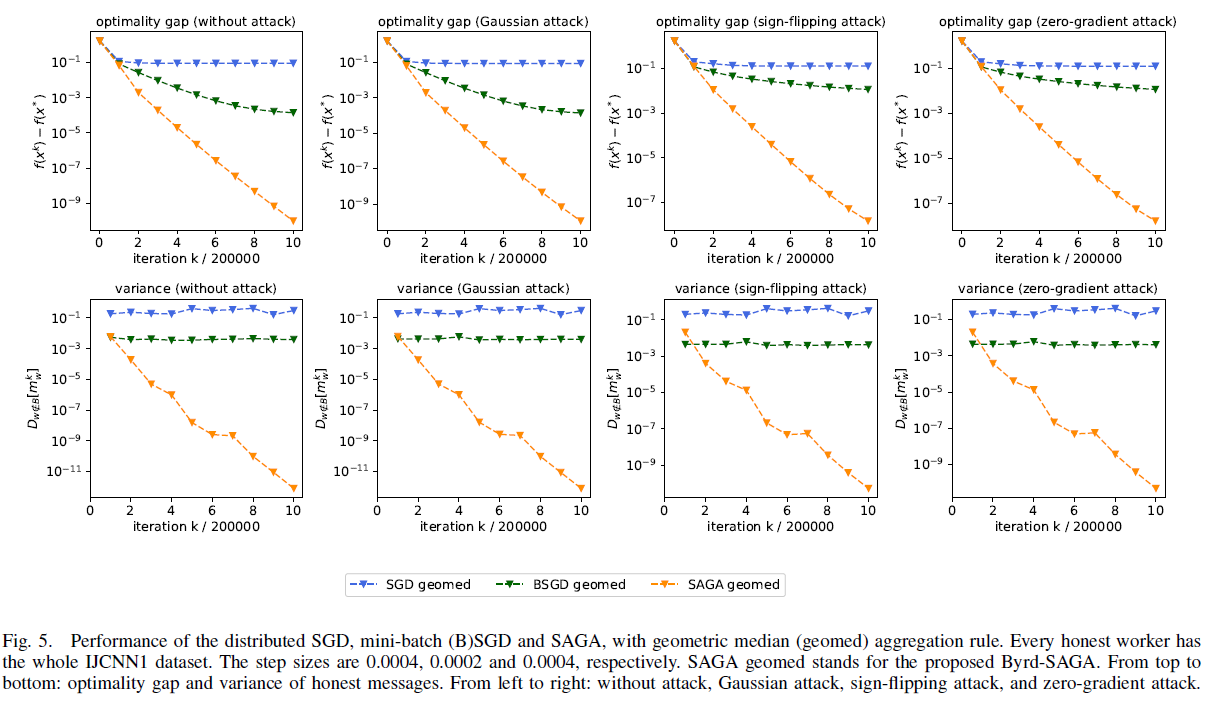


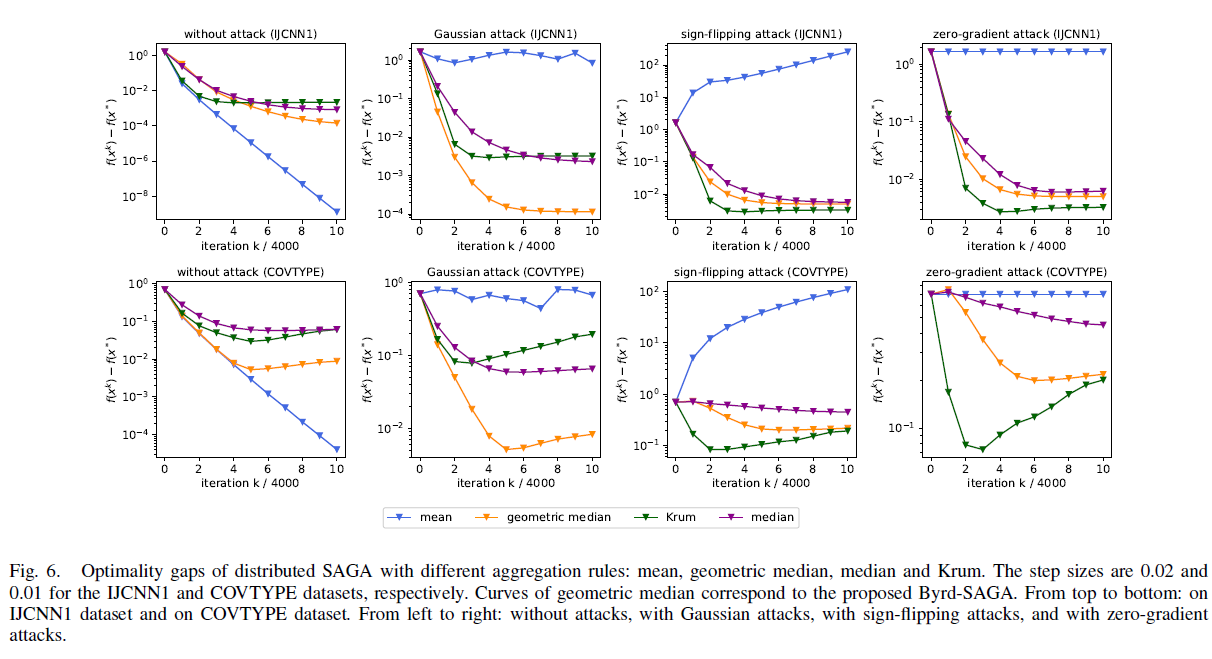
Byrd-SAGA:

When aggregating, apply Geometric median:

------------> 

Numerical Tests:



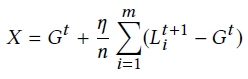


1. **How To Backdoor Federated Learning**

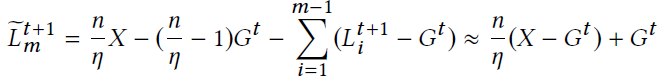
<https://arxiv.org/abs/1807.00459>

Naive approach: The attacker can simply train its model on backdoored inputs.

Model replacement: The attacker ambitiously attempts to substitute the new global model with a malicious model (see below). M is the number of participants, is the global learning rate (the model is fully replaced when ).



the attacker can solve for the model it needs to submit as follows, where is the local model:



This attack scales up to ensure that the backdoor survives the averaging and the global model is replaced. An attacker who does not know n and η can approximate the scaling factor γ= by iteratively increasing it every round and measuring the accuracy of the model on the backdoor task.

To ensure the model does not look anomalous, add a constrain to loss function (see below), where captures the accuracy on both the main and backdoor tasks, accounts for any type of anomaly detection.



The whole procedure:

1. Replace benign items with backdoor items.

2. Update malicious model X using gradient decent, with constrained loss function.

3. Scale up X (increase by rounds till reaching anomaly detector’s bound).

4. Upload X.

Numerical tests: Each batch (64) contains 20 backdoor samples.

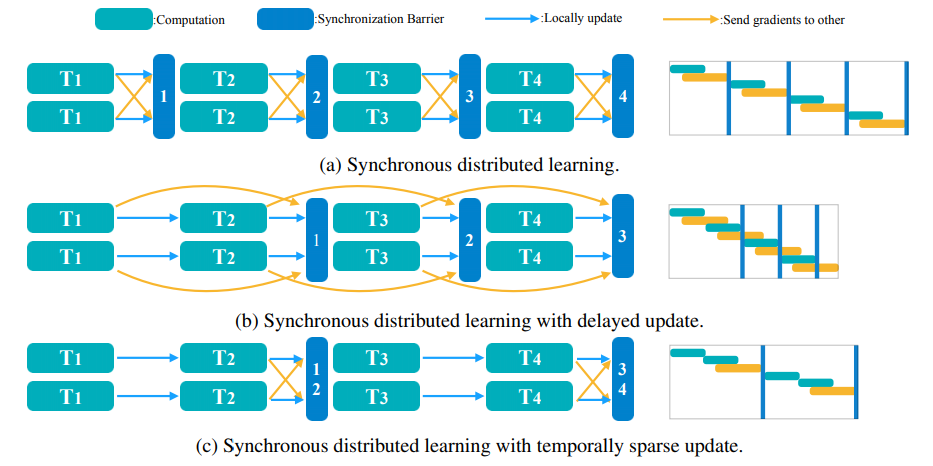
Single-shot attack: The accuracy of the global model on the backdoor task immediately reaches almost 100%, then gradually decreases; some backdoors appear to be more successful and durable than others. The accuracy on the main task is not affected.

Repeated poison attack: 2% (CIFAR10) or 0.05% (word prediction) of attackers can reach to over 80% Mean backdoor accuracy.

1. **Distributed Training Across the World**

<https://openreview.net/forum?id=SJeuueSYDH> (not FedML, just distributed system)

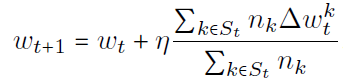
Delayed and temporally sparse (DTS) update to increase scalability in system with high latency. In the original synchronous system, local gradient update of round (t) is averaged and applied in round (t). in delayed setting, local gradient of round (t-1) is averaged and applied in round (t). In delayed and sparse setting, averaging happens not in every round.



1. **Can You Really Backdoor Federated Learning?**

<https://arxiv.org/abs/1911.07963>

is the total number of nodes, select clients to be the involved set of round . is number of compromised nodes. Under random sampling of clients, the number of adversaries in each round follows a hypergeometric distribution. In fixed frequency attack, a single adversary appears in every rounds, in which . is the number of samples of node . is the model update. In the system:

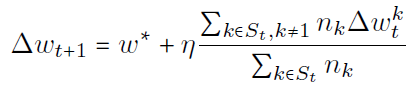


The goal of the adversary is to reduce the performance of the model on targeted tasks while maintaining a good performance on the main task. Unlike existing works, non-malicious clients are allowed to have correctly labeled samples from the targeted tasks.

The backdoor model is denoted as . is a boost factor. The model update is:

<----------- 

Then we have:

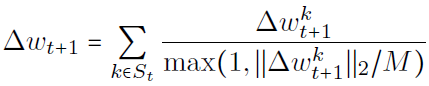


Assume: (a) attackers can coordinate with each other and divide the update evenly, (b) attacker has a set for the backdoor task, (c) attacker has a set generated from the true distribution.

Unconstrained boosted backdoor attack (norm clipping): One popular training strategy is to initialize with and train the model with for a few epochs, no constrain.

Norm bounded backdoor attack: The model update to be smaller than .

Defense 1 – Set a threshold for norm: updates. Since boosted attacks are likely to produce updates with large norms. And assume the threshold is known. Thus:



Defense 2 – (Weak) differential privacy: Adding a small amount of noise that is empirically sufficient to limit the success of attacks.

Conduct experiments on the EMNIST dataset, result: (a) Fixed frequency attacks being slightly more effective than random sampling attacks. (b) The performance of backdoor attack degrades as the fraction of fully compromised users falls below 1%. (c) The more backdoor tasks we have, the harder it is to backdoor a fixed-capacity model while maintaining its performance on the main task; and benign workers have the ability to correct the model. (d) Norm clipping and “weak” differential privacy mitigate the attacks without hurting the overall performance.

1. **Local Model Poisoning Attacks to Byzantine-Robust Federated Learning**

<https://arxiv.org/abs/1911.11815>

Apply attack to recent Byzantine-robust federated learning methods (Krum, Bulyan, trimmed mean, and median). And generalize two defenses for data poisoning attacks to defend against these local model poisoning attacks.

**Krum and Bulyan**: Krum selects the local model with the smallest sum of squared distance as the global model. Bulyan first iteratively applies Krum to select local models. Then, Bulyan uses a variant of trimmed mean to aggregate the local models.

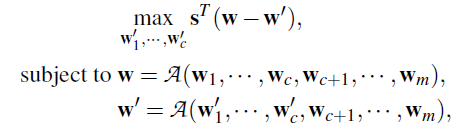
**Trimmed mean**: sort the parameters of the **m** local model, removes the largest and smallest of them.

**Median**: sort the parameters of the **m** local model, take the median as the parameter of the global model

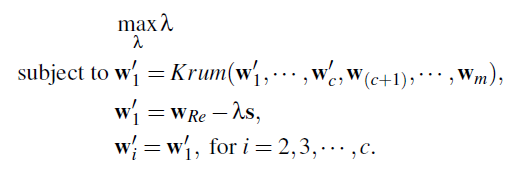
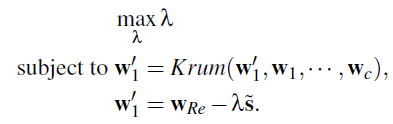
Krum and trimmed mean need to know the upper bound of the number of compromised workers. we consider a hypothetical, strong service provider who knows the number of compromised worker devices and sets parameters in the aggregation rule accordingly.

*Attack:*

***s*** *is a column vector of the changing directions* of all global model parameters, is the before-attack global model, and is the after-attack global model. The attacker has control of **c** worker devices. Aims to:



Attack – Krum: To solve the following optimizing function.

 ---> 

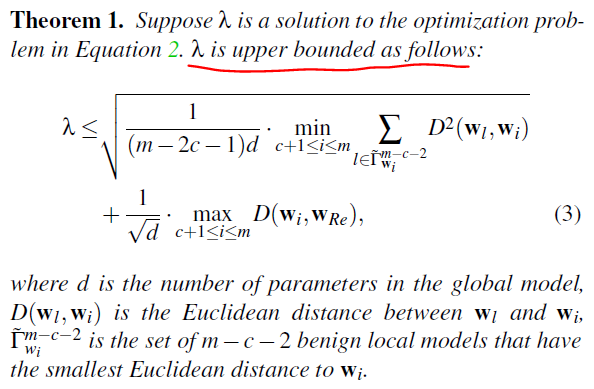
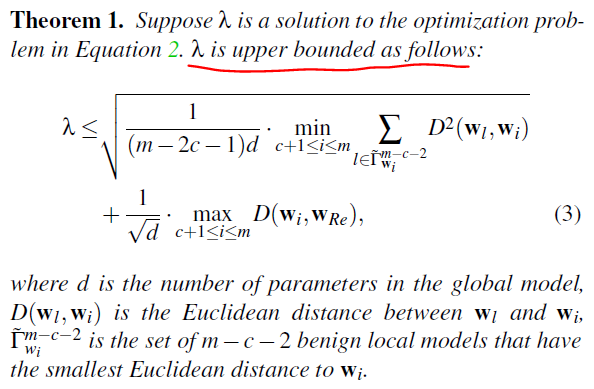
( is the compromised model we wat Krum to select, and since (c+1)-m nodes are unknown to attacker, so the weights before attack is assumed to be the benign nodes.)

is rounded from:



is the global model received from the master device in the current iteration (i.e., the global model obtained in the previous iteration). The approximation shows the deviation between the crafted local model and the received global model . The approximation shows the deviation between the crafted local model and the local model selected by Krum before attack (). Also craft the other **c-1** compromised local models to be close to .

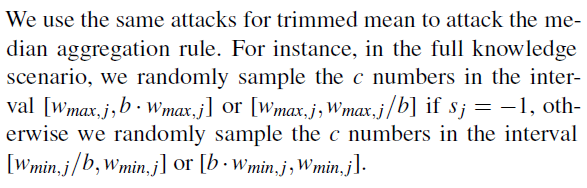
can be solved using binary search, if its upper bounded is known:

If we cannot find that is larger than a threshold, we add one more crafted model so and Krum selects . We keep adding in until we get valid result.

Attack – Trimmed Mean: Get based on the weights before attack. If =-1, sample for compromised nodes from a Gaussian distribution – within the interval [+,+]; If =1, sample for compromised nodes – within the interval [-,-].

Attack – Median:



*!!! Both defense methods assume a small validation dataset.*

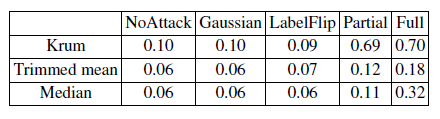
Defense - Error Rate based Rejection (ERR): Removes the local models that have large negative impact on the error rate of the global model.

Defense - Loss Function based Rejection (LFR): Removes the local models that result

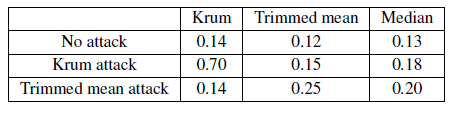
in large loss (inspired by TRIM that removes the training examples that have large negative impact on the loss).

*Numerical Tests:* non-IID, Linear Regression & DNN. More plots in the original paper.

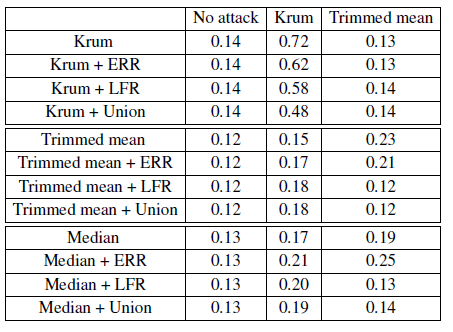
If aggregation rues known: (testing error rates)



If aggregation rules unknown: (testing error rates)



If use new defense: (testing error rates)



1. **RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets**

<https://arxiv.org/abs/1811.03761>

1. Develop RSA for distributed learning over heterogeneous datasets and under Byzantine attacks, (b) performance is rigorously established,(c) numerical tests using the MNIST dataset, showing both classification and runtime.

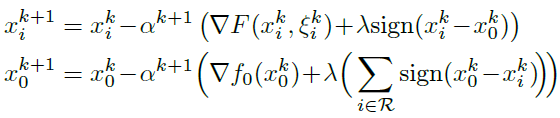
RSA aims to find a solution that minimizes the summation of the regular workers’ local expected cost functions plus the regularization term.



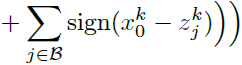
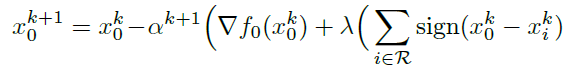
For -norm RSA:



is the local model, is the master iterate, is data, is a regularization term. (a) equals to 1 when a>0, -1 when a<0, an arbitrary value in [-1,1] when a=0. At time , the update at and without presence of Byzantine workers:

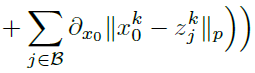
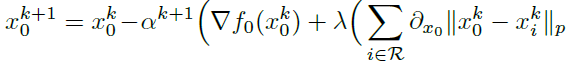


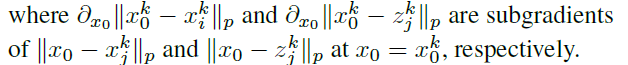
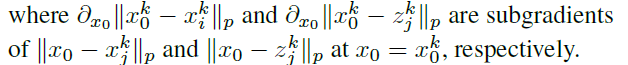
With presence of Byzantine workers, since cannot distinguish the message from the byzantine worker (), it updates:

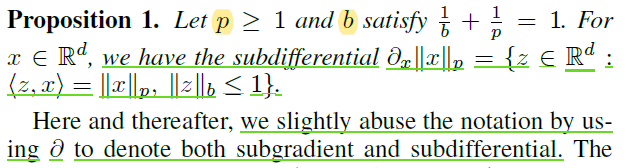


For -norm RSA:









Skipped Convergence Analysis.

Numerical test over both IID and heterogeneous dataset are tested for top-1 accuracy and runtime, comparing with Krum, GeoMed, median, ideal SGD. Attack includes: Same-value attacks, Sign-flipping attacks, Gaussian attacks. Result figures are in original paper.

1. **DBA: Distributed Backdoor Attacks against Federated Learning**

<https://openreview.net/forum?id=rkgyS0VFvr>

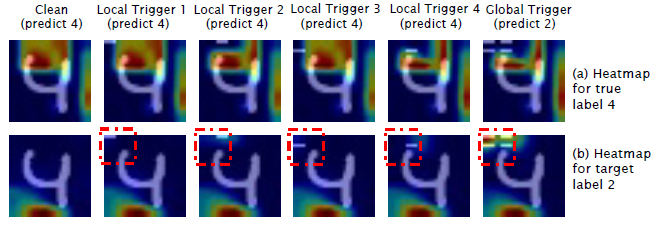
The ideal feels like: A attacker can control several nodes and imply targeted poisoning. Its attack is distributed into the nodes in small pieces, which is mild and harder to be detected compared to the whole global attack, and the small pieces of attack is assembled during aggregation.



Numerically tested multiple-shot attack (Attack A-M) and single-shot attack (Attack A-S). Also test with 2 defense methods: RFA (replaces the weighted arithmetic mean in the aggregation step with an approximate geometric median) and FoolsGold (reduces aggregation weights of participating parties that repeatedly contribute similar gradient updates while retaining the weights of parities that provide different gradient updates).

Result:

Success poisoned models from the local contains more low importance feature. The below image shows how weak attacks aggregates to form a strong global attack:



Effect of Scale: Small scale not effective, large scale easy to be detected.

Effect of Trigger Location: Trigger at fundamental areas to the main accuracy can be soon corrected.

Effect of Trigger Gap: large distance between local triggers may cause failure, while zero-gap can be soon forgotten.

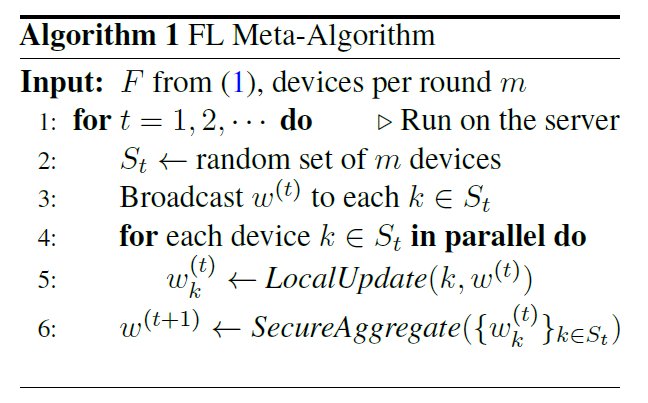
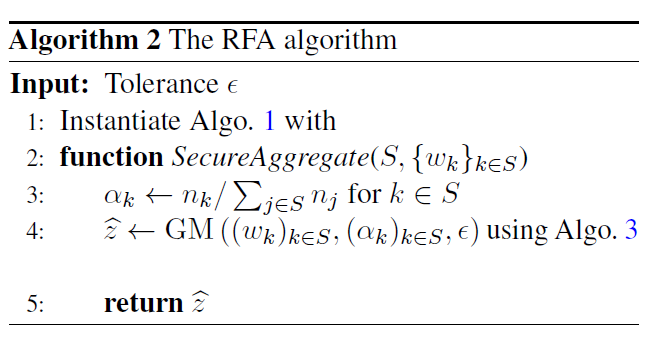
Effect of Poison Interval: The local trigger effect can last long and contribute to the attack performance of global trigger. Especially for RL.

Effect of Poison Ratio: Also need to consider the accuracy on clean data.

Effect of Data Distribution: Under various data distributions, DBA-ASR is stable.

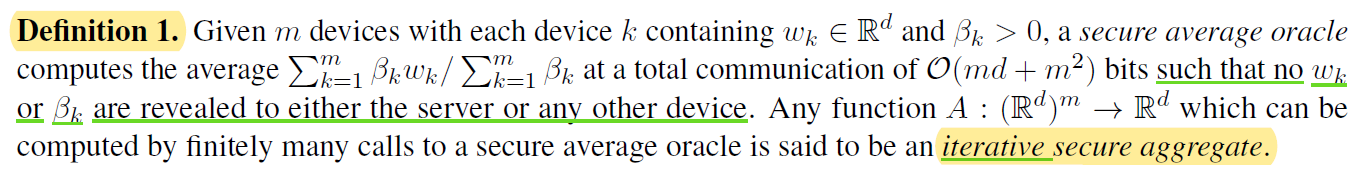
1. **Robust Aggregation for Federated Learning** (RFA mentioned in DBA)

<https://arxiv.org/abs/1912.13445>

RFA just change FL Mata-Algorithm into GM-based.

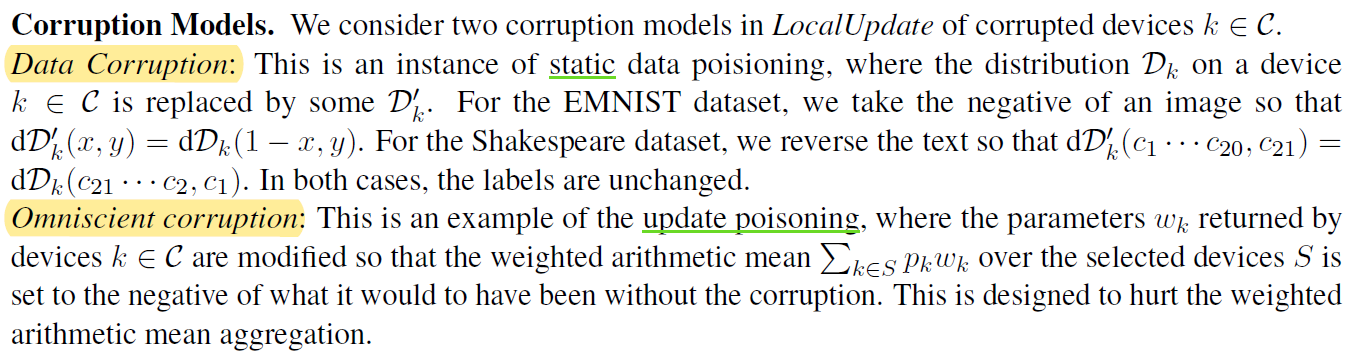
*LocalUpdate* is like in normal systems. *SecureAggregate* is:



Skipped the Math of Convergence, in which the authors analyze the convergence of the resulting FL algorithm for i.i.d. least-squares estimation

There are: Static data poisoning (modify training data prior to the start of FL), Adaptive data poisoning (modify training data in each round of FL), Update Poisoning (run an arbitrary procedure in place of *LocalUpdate*).

*In the numerical tests,* tasks implemented with EMNIST and Shakespeare. Compare to FedAvg and SGD. Using the following corruption model:

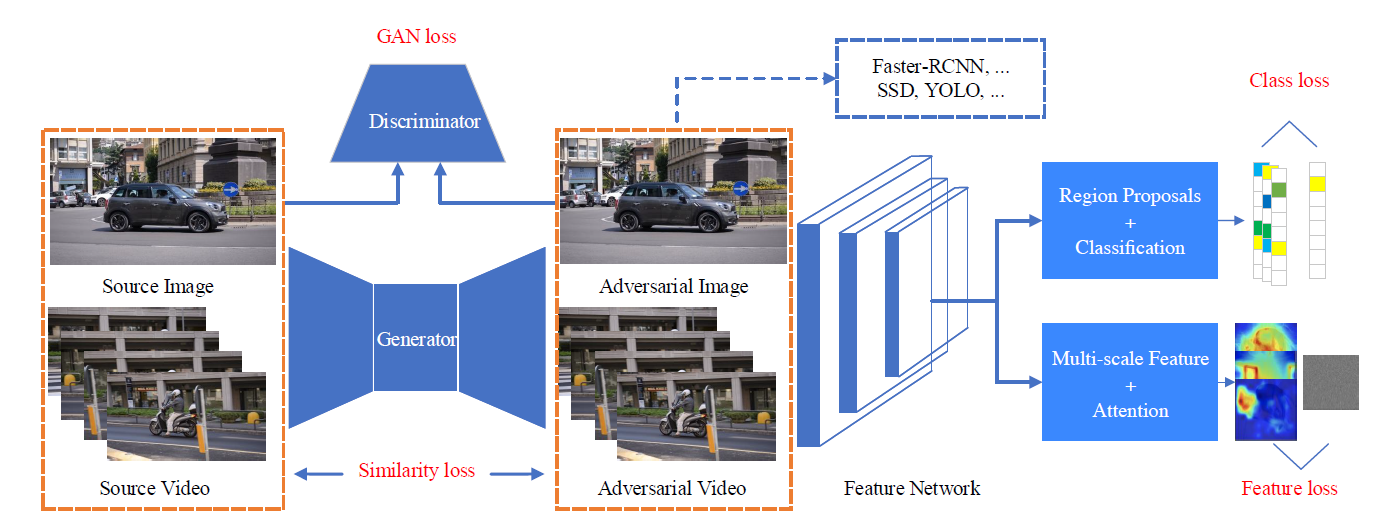


*Summary:* The robust aggregation approach is agnostic to the level of corruption; it outperforms the classical aggregation approach in terms of robustness when the level of corruption is high, while being competitive in the regime of low corruption

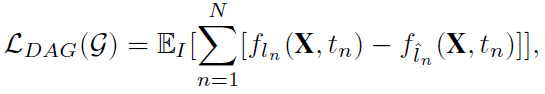
1. **Transferable Adversarial Attacks for Image and Video Object Detection**(not FedML)

<https://arxiv.org/abs/1811.12641>

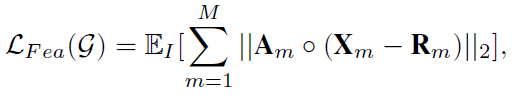
DAG is a white-box attacking method which firstly assigns an adversarial label for each proposal region and then performs iterative gradient backpropagation to misclassify the proposals. It has two limitations: weak transferability and high computation cost. The author proposed a black-box attacking method UEA for these 2 limitations. UAE can simultaneously attack both the proposal based and regression-based detectors in an efficient way.



Class loss: to make the predictions of all proposal regions go wrong. **X** is the extracted feature map from the feature network, is the proposal region on **X**, is the ground-truth label of , is the wrong label randomly sampled from other incorrect classes. is the classification score vector.



Multi-scale attention feature loss:

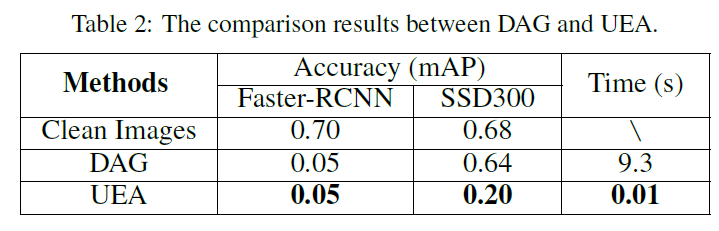


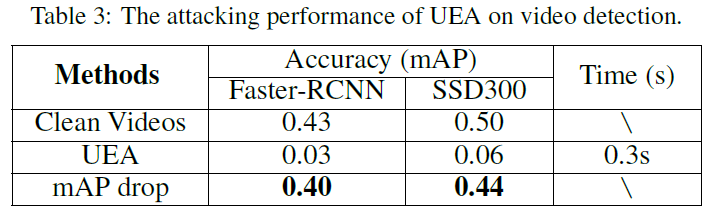
is the extracted feature map in the layer of the feature network. is a randomly predefined feature map, and is fixed during training. Attention weight measures the objects in . is the score of region proposal , ***S*** is the collection of , N is its number. =. ○ is the Hadamard product between two matrices.

Adding in the objective of the conditional GAN and the L2 loss between the clean images (or frames) and adversarial images (or frames), finally:



Numerical experiment: pixel-to-pixel,

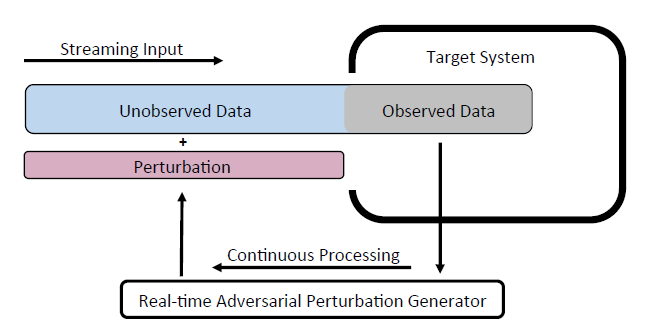




1. **Real-Time Adversarial Attacks**(not FedML)

<https://arxiv.org/abs/1905.13399>

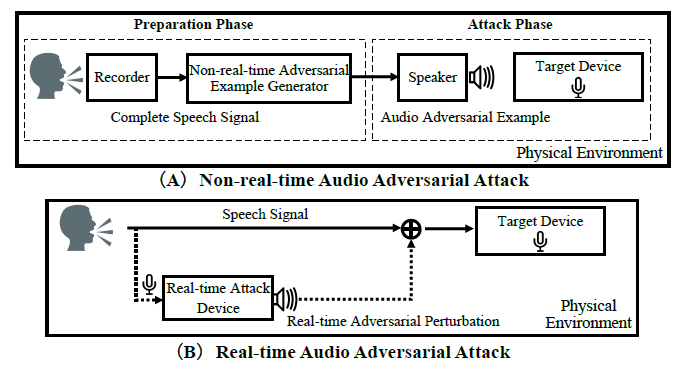
The authors propose a real-time adversarial attack scheme for machine learning models with streaming inputs. In the below system, only past data points can be observed and used to approximate an optimal adversarial perturbation for future data points via Reinforcement learning.



The whole procedure includes:

1. Generate ‘sample-perturbation’ pairs by feeding different samples to state-of-the-art non-real-time attack models. This is to avoid the challenge of ‘sparse rewards problem’ in RL, i.e. the agent only receives the reward at the end and it is difficult to obtain an estimation of the reward at each time point based on the observed data and past actions.
2. Train the Realtime Adversarial Example Generator.
3. Use the Trained Generator to conduct Real-time Adversarial Attack.

Testing:



Result: *The case study (voice command recognition) and results demonstrate the effectiveness of the proposed approach. Nevertheless, we observe a certain performance gap between the real-time and the non-real-time adversarial attack when the basic behavior cloning algorithm is used.*

1. **BadNets: Evaluating Backdooring Attacks on Deep Neural Networks** (not FedML)

<https://ieeexplore.ieee.org/document/8685687>

In *Fully Outsourced Training Attack:* The backdoored model (a) not reduce classification accuracy on the validation set, (b) for inputs that have certain attacker chosen properties, predictions to be different from honest model. (e.g. Happened when hiring cloud resources)

In *Transfer Learning Attack:* (a) has high accuracy in the original domain,(b) has high accuracy in the new domain,(c) misbehaves for every input from the new domain if there’s trigger in it.

Numerical tests on: MNIST & Traffic Signs for testing outsourced attack, also test transfer learning attack via: if backdoors in the U.S. traffic signs can also misbehaves when transferred to Swedish traffic signs.

Result of Traffic Signs show that some neurons are activated if and only if the backdoor is present in the image. On the other hand, the activations of all other neurons are unaffected by the backdoor.

The author also evaluate the security of two popular online repository of pre-trained DNN models, and *found*: an attacker can change a model either by compromising the external server on which the model is hosted, or by changing the model while it is being downloaded, if the user uses an insecure HTTP connection.

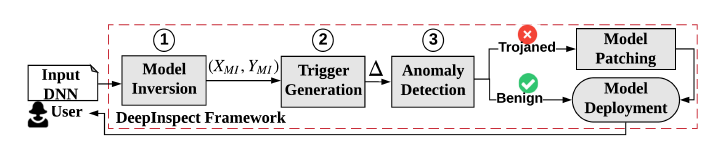
*Suggest*: Online repositories like the Caffe Model Zoo to prevent benign models from being tampered with; and detecting backdoors in maliciously trained models.

1. **DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks**(not FedML)

<https://www.ijcai.org/Proceedings/2019/647>

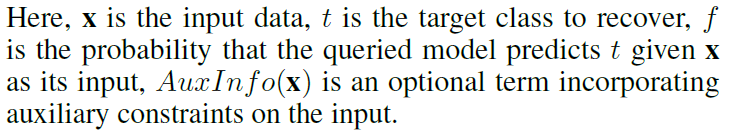
DeepInspect is a black-box Trojan detection solution (need minimal prior knowledge of the model). It learns the probability distribution of potential triggers from the queried model using a conditional generative model to retrieves the footprint of backdoor insertion.

Experiments show that DeepInspect offers superior detection performance and lower runtime, and shows effectiveness, efficiency, and scalability.

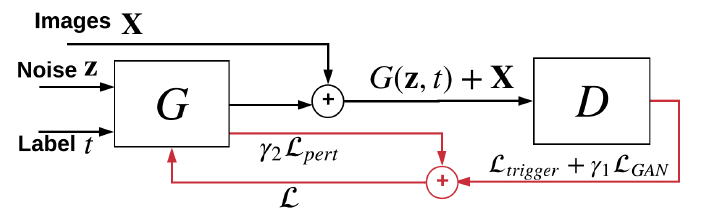


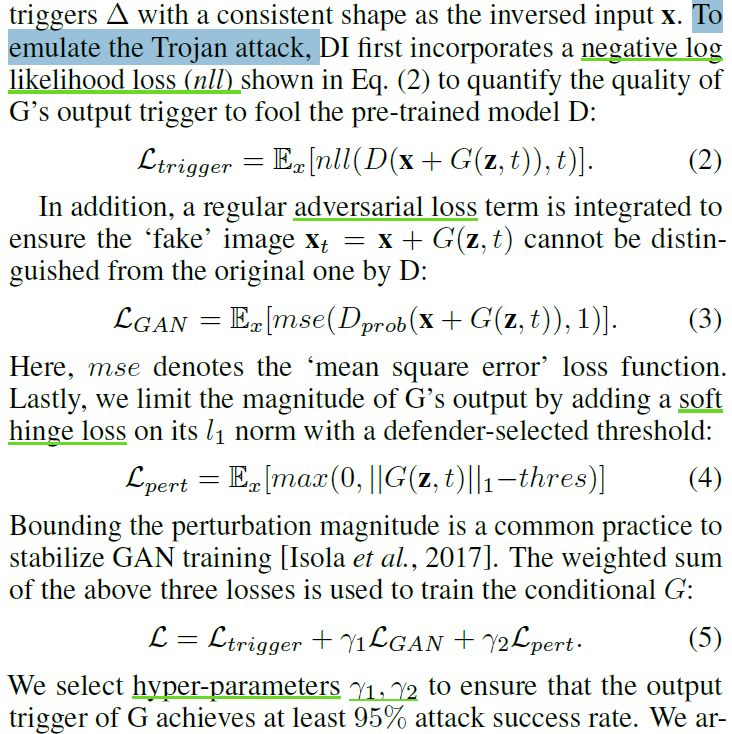
Model Inversion: The threat model assumes no clean training dataset is available during Trojan detection. So, the first step is to recover a substitution training set { ***X***, ***Y*** } via model inversion. (data can be extracted from a pre-trained model)



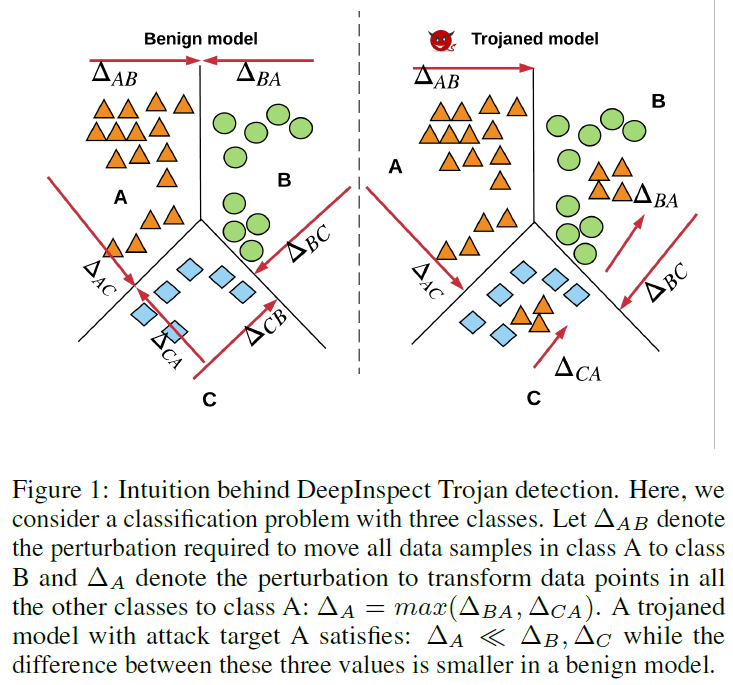


Trigger Generation: To train a GAN network as following:





Anomaly Detection: The trigger has an abnormally smaller perturbation level for the target class compared to other uninfected classes in a trojaned model (see below figure). Use a variant of ‘Double Median Absolute Deviation’ (DMAD) to decide the detection criteria. (still not clear about DMAD)



Numerical Test under BadNets attack with MNIST & GTSRB:

Perturbation levels (hinge loss) of the generated triggers for infected labels in a trojaned model is smaller.

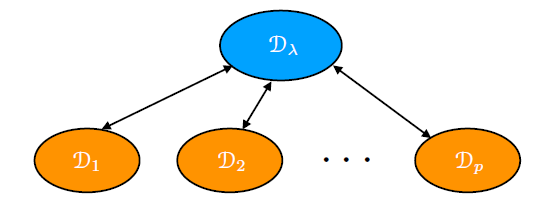
Deviation factors (df) of the generated triggers for infected labels in a trojaned model is larger.

Other test results in original paper.

1. **Agnostic Federated Learning**

<https://arxiv.org/abs/1902.00146v1>

Federated models can be biased towards different clients (e.g. nice cell phones for technical users). Agnostic federated learning (AFL) optimizes the centralized model for any target distribution formed by a mixture of the client distributions. Data-dependent Rademacher complexity guarantees for learning with this objective. A fast stochastic optimization algorithm and its convergence bounds are also given.



is the sample distribution of the node, is its mixture weight. is a predictor (hypothesis).



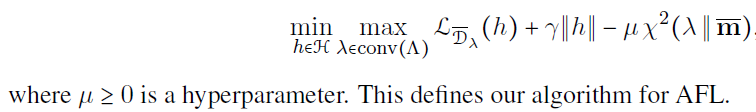
The agnostic loss (or agnostic risk): Since the learner has access to the distributions Dk only via the finite samples, empirical distributions are used.



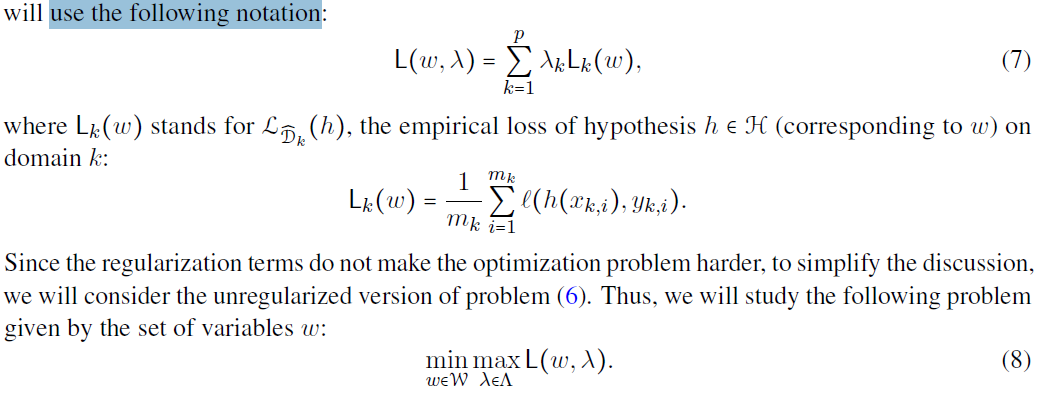
In Section 4 the authors present learning guarantees for AFL based on *weighted Rademacher complexity*. (not fully understood this part)

learning guarantees in Section 4 suggest (a) minimizing the asum of the empirical

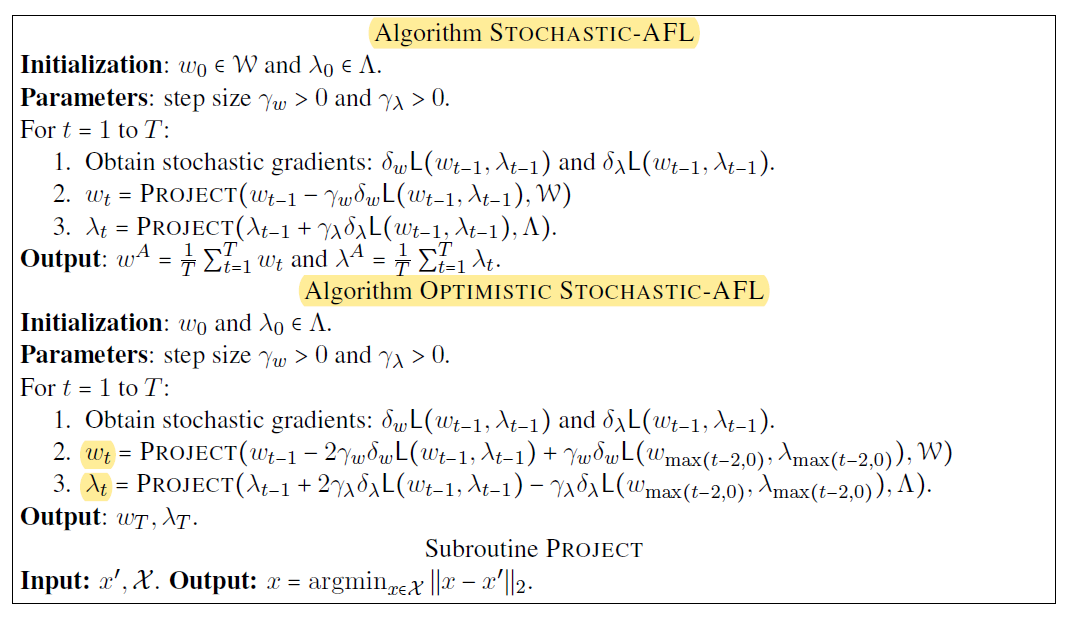
AFL term and (b) minimizing the regularized loss. [the optimistic AFL]



This is a convex optimization problem, turn to the following notation:

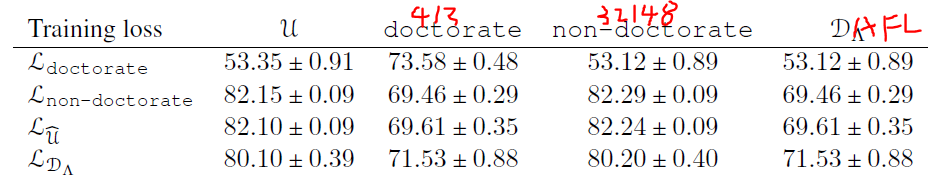


To solve the problem: It feels like, first to calculate gradient for and , then solving the minmax problem for them (PROJECT) and update them. Repeat for many times.



Numerical tests on income and MNIST and language models:

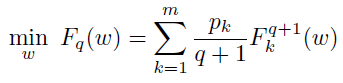
Here show result of income, rest in original paper.



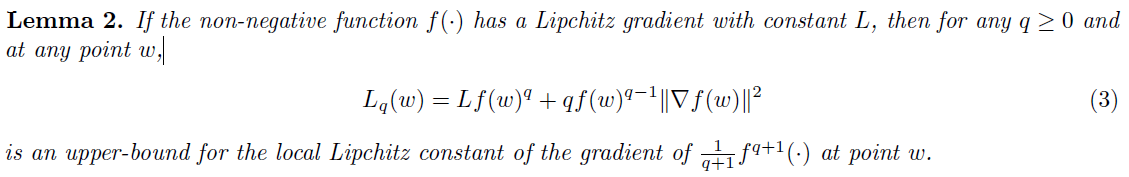
1. **Fair Resource Allocation in Federated Learning**

<https://arxiv.org/pdf/1905.10497v1.pdf>

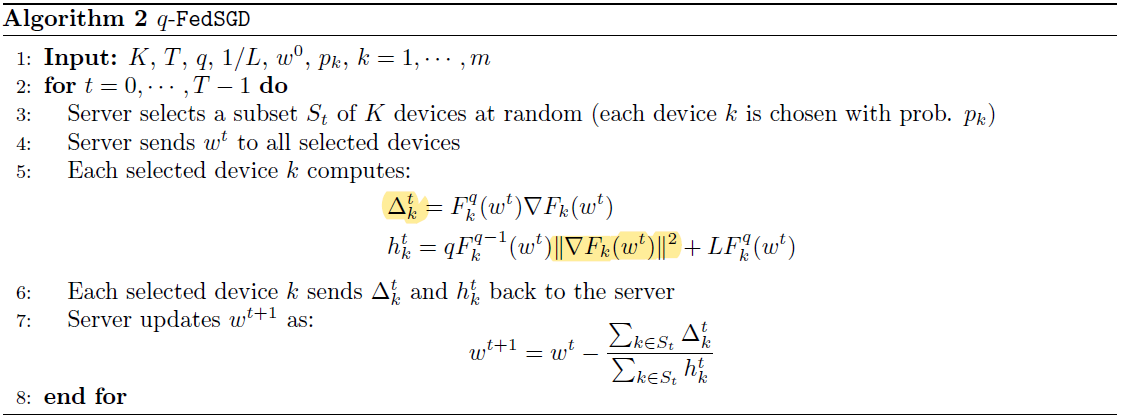
Naively minimizing the average loss in such a massive network may disproportionately advantage or disadvantage the model performance on some of the devices. q-FFL minimizes an aggregate reweighted loss parameterized by q such that the devices with higher loss are given higher relative weight to encourage less variance (i.e., more fairness) in the accuracy distribution: F is the cost function.



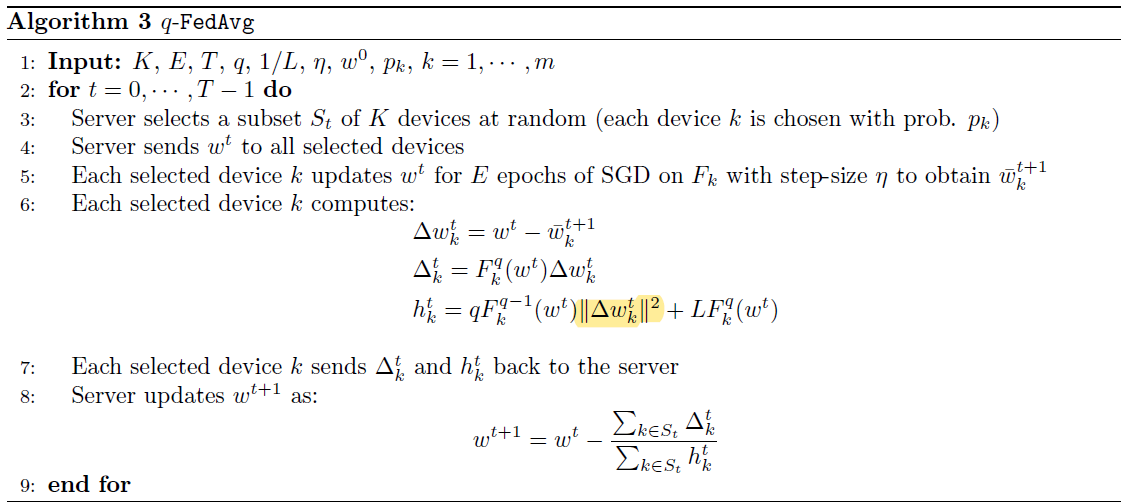
*As we intend to optimize q-FFL for various values of q, the Lipchitz constant will change as we change q—requiring step-size tuning for all values of q. This can quickly cause the search space to explode. To overcome this issue, we propose estimating the local Lipchitz constant of the gradient for the family of q-FFL objectives by using the Lipchitz constant we infer via grid search on q = 0:*



***q-FedSGD:*** (a) Uses a dynamic step-size based on Lemma 2 (above) instead of the normal fixed step-size of FedSGD, (b) to run q-FedSGD with different values of q, we only need to estimate L once (for q = 0) and can then re-use it for all values of q > 0.



***q-FedAvg:*** Replace the gradient of the local 6 functions, , in the q-FedSGD steps with the local update vectors that are obtained by running SGD locally on device k.



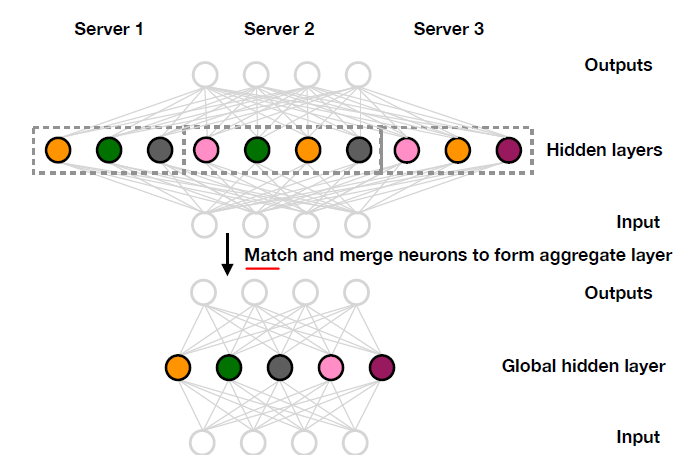
***Numerical Tests:***

While the average accuracy remains similar, the variance of the final accuracy distribution decreases. Also, q-FFL converges faster in terms of communication rounds compared with AFL (another optimization algorithm) to obtain similar performance, if q is set appropriately. (But I feel the global accuracy is not improved, this paper mainly focuses on ‘fairness’ in accuracy)

1. **Bayesian Nonparametric Federated Learning of Neural Networks** (PFNM)

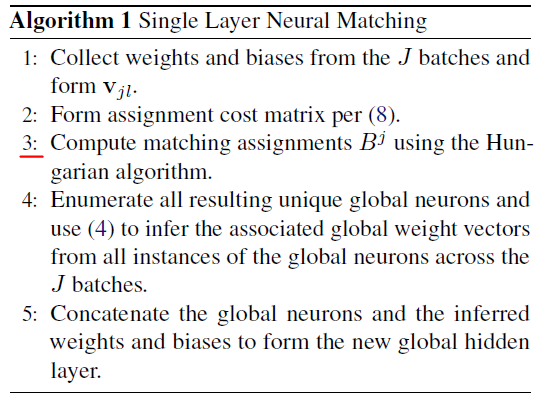
<https://arxiv.org/abs/1905.12022v1>

Governed by the posterior of a Beta-Bernoulli process (BBP), a Bayesian nonparametric (BNP) model will allow the local parameters to either match existing global ones or to create new global parameters if existing ones are poor matches.



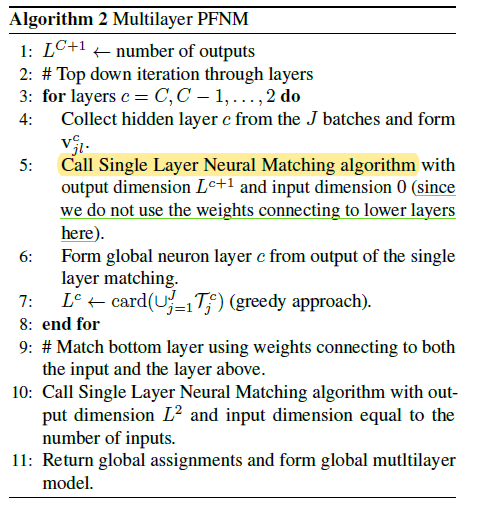
Advantage: (a) decouples the learning of local models from their amalgamation into a global federated model, (b) without requiring additional data or knowledge of the learning algorithms used to generate the pre-trained models, (c) can recover compressed global models with fewer parameters than the cardinality of the set of all local parameters, (d) requiring far fewer communications between the local data sources and the global model server. [But feels like PFNM currently not support CNN and RNN.]

I still not fully understand its math, so I skip it, just mentioning: **(4)** is the MAP estimate of }, **(8)** is the (negative) assignment cost specification for finding random variables that match observed neurons.



For multilayer: Using the matching assignments of the next highest layer to convert the neurons in each of the J batches to weight vectors referencing the global previous layer [direction: output->input], and these weight vectors are then used to form a cost matrix, which the Hungarian algorithm then uses to do the matching. Finally, the matched neurons are then aggregated and averaged to form the new layer of the global model.

[So one process from layer n to 1 happens in s single communication round??]



Numerical Tests: MNIST and CIFAR10, both IID and heterogenous, compared to FedAvg, k-Means of vectors, Ensemble, Local NN. Results in original paper.

1. **Federated Learning with Matched Averaging** (FedMA)

<https://arxiv.org/abs/2002.06440>

FedMA constructs the shared global model in a layer-wise manner by matching and averaging hidden elements (i.e. channels for convolution layers; hidden states for LSTM; neurons for fully connected layers) with similar feature extraction signatures. In this work, we demonstrate how PFNM can be applied to CNNs and LSTMs, but we find that it only gives very minor improvements over FedAvg.

Reason to do matching: FedAvg has decrease in accuracy and is slow due to the permutation invariance of neural network (NN) parameters, i.e. for any given NN, there are many variants of it that only differ in the ordering of parameters.

**Show Permutation Invariance:** x is input feature, n is the layer index, W is model parameter.

Basic FC:



Deep FC:



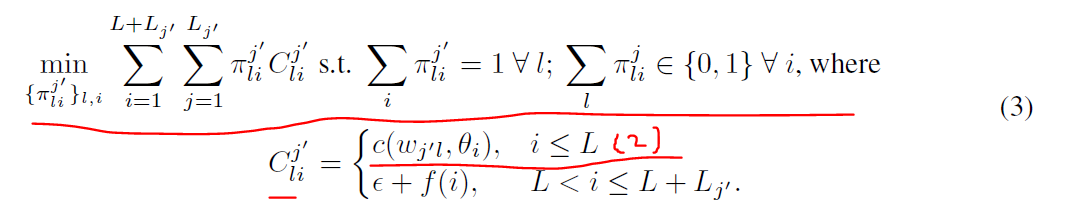
CNN: Note that this formulation permits pooling operations as those acts within channels.

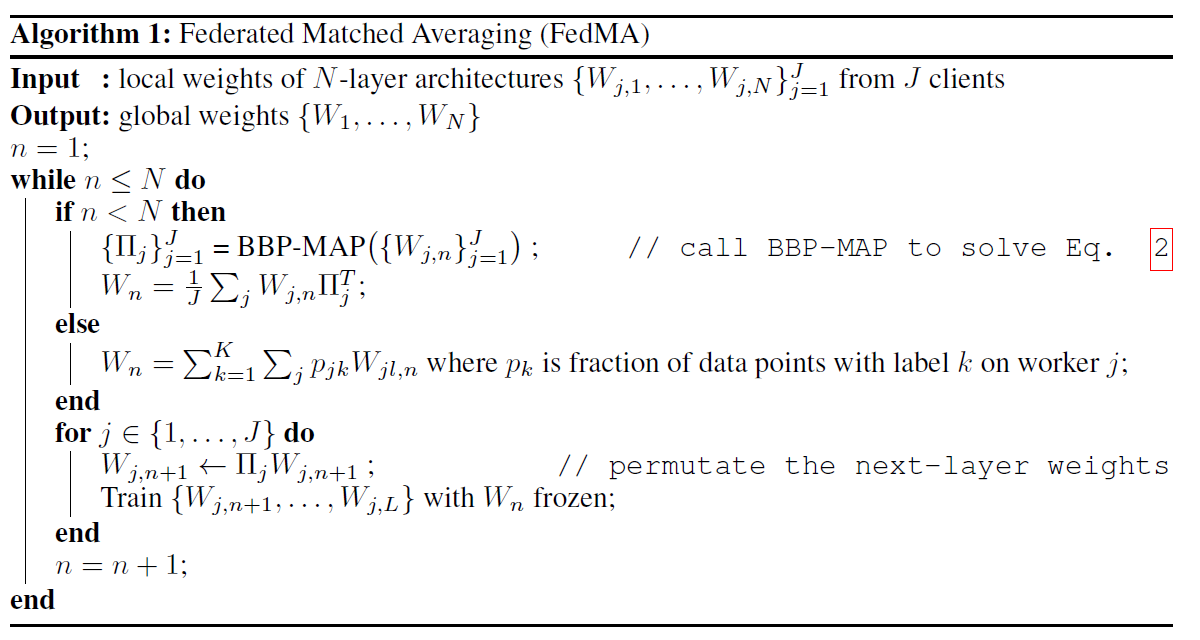


LSTM:

--->

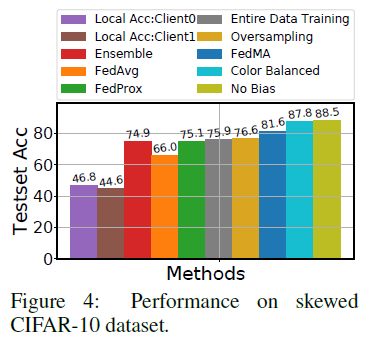
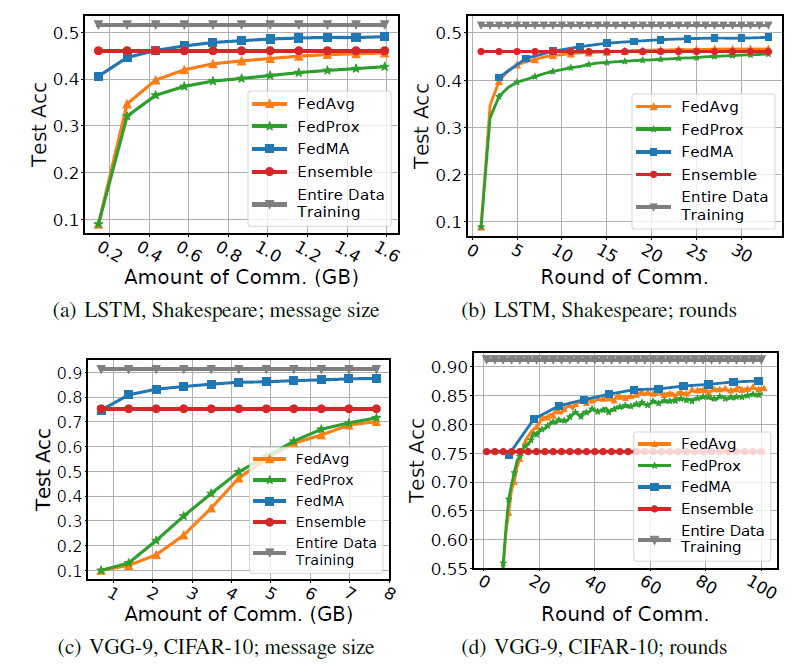
Let c(\*,\*) be an appropriate similarity function between a pair of neurons. is the neuron of the global model. Apply *Hungarian matching algorithm* (iterate over clients J) to find the permutation matrix, solving:

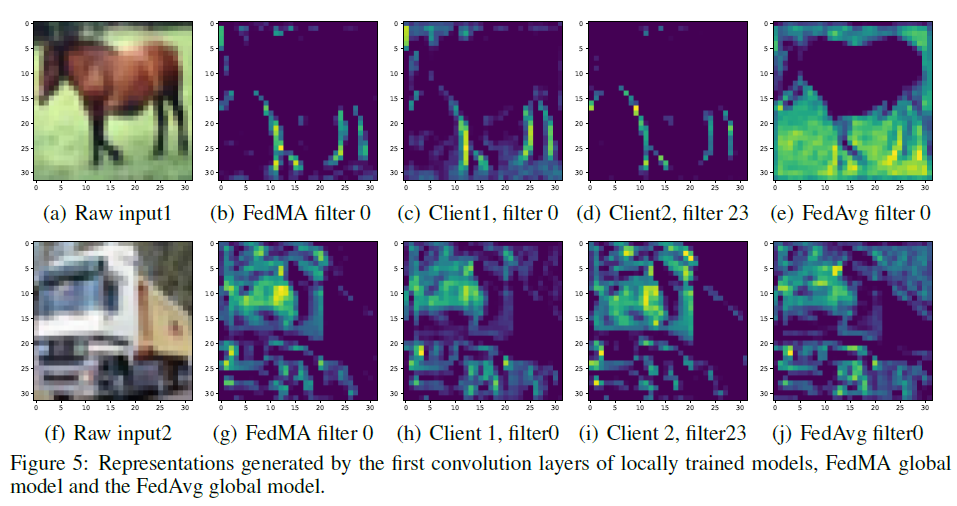




It feels each communication round can train one layer. (?)

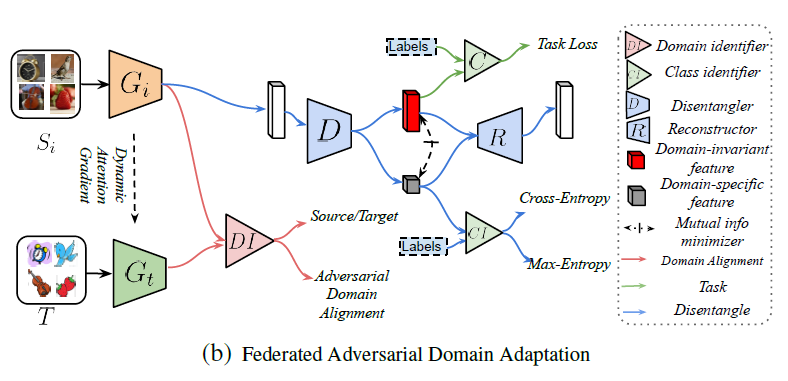
Testing:





1. **Federated Adversarial Domain Adaptation** (just a brief description here, math not mentioned.)

<https://arxiv.org/abs/1911.02054>



------For every domain (say, the *i*-th domain):--------

1. Update feature extractor (***G***) and classifier (***C***)
2. Domain Alignment: (a) Train domain identifier (***DI***) to align the distributions, (b) Train local feature extractor () and target feature extractor () to confuse ***DI***.
3. Domain Disentangle:

(3a) Update disentangle (***D***), ***G***, ***C***, and class identifier (***CI***): ***D*** split the extracted features in: Domain-Invariant features (for calculating task loss of ***C***) and Domain-Specific features (for calculating Max-Entropy of ***CI***).

(3b) Train ***D*** by generating Domain-Specific features.

1. Mutual Information Minimization: Mutual Information Estimator (M) between Domain-Specific features and Domain-Invariant features, find minimum via Monte-Carlo integration. At the same time, features went into Reconstructor (***R***) and come back to original features.

----------End For.----------------

Dynamic Weight: increase the weight of nodes with beneficial gradients and vice versa. It can be done via: calculate gap statistics, i.e. how well target features can be clustered using k-mean. Update ***G*** and ***C***.

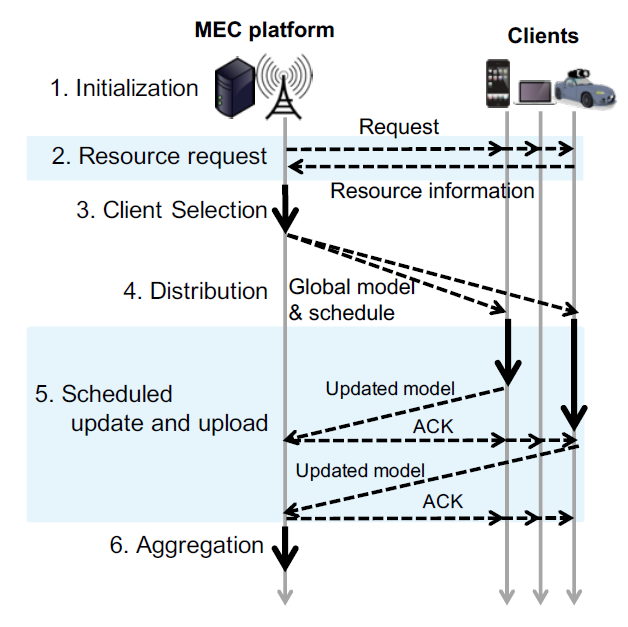
Iterate the above procedure, return ***G*** and ***C*** at the end.

Also have experiments, result in original paper.

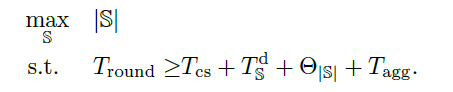
1. **Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge** (FedCS)

<https://arxiv.org/abs/1804.08333>

Authors consider setting a reasonable strategy for managing FL among heterogeneous devices (e.g. different computational capabilities, network). The main idea is: Clients tell the center their resource information, and based on that the center can have a rough idea about how much time is need for what.



In client selection, FedCS sets a certain deadline for clients to download, update, and upload ML models in the FL protocol, select as much as clients as possible:



S is the set of selected devices of that round, T is time consumption. ‘round’ is the deadline of a round, ‘CS’ is time for client selection, ‘T^d\_S’ is the estimated time for distribution (step 4), ‘UD’ for update and ‘UL’ for upload, kj is the j-th client.

