Appendix: Certified Federated Adversarial Training

Giulio Zizzo
IBM Research Europe
giulio.zizzo2@ibm.com

Ambrish Rawat
IBM Research Europe
ambrish.rawat@ie.ibm.com

Mathieu Sinn
IBM Research Europe
mathsinn@ie.ibm.com

Sergio Maffeis
Imperial College London
sergio.maffeis@imperial.ac.uk

Chris Hankin
Imperial College London
c.hankin@imperial.ac.uk

Appendix A

We show in Figure 1 an example of a zonotope being pushed through a single layer in a toy neural network to help build intuition for the reader into the ideas behind using zonotopes. In Figure 1 we start with two features which can each take values between 0 - 0.5. We initially convert this into a zonotope, so each feature now has a central term of 0.25, and an error term ϵ_1 or ϵ_2 with coefficients of 0.25. With the ϵ terms being able to take ± 1 this draws the zonotope shape shown of a square.

This zonotope now gets pushed through the dense layer with weights

$$W = \begin{pmatrix} 2 & 1 \\ 1 & -1 \end{pmatrix}. \tag{1}$$

Multiplication of a zonotope by a constant affects all the terms, and we add zonotopes together term by term giving us $\hat{x}_3$ and $\hat{x}_4$. This gives the rotated rectangle shown in the middle plot. Note, that here we can see a advantage of the zonotope domain over just using intervals, as the rotated rectangle has a smaller area then just considering the upper and lower bounds of $\hat{x}_3$ and $\hat{x}_4$.

Finally, we apply a ReLU activation. As $\hat{x}_3$ has a positive upper bound and a lower bound of 0, the ReLU activation does not affect it. However, $\hat{x}_4$ has a negative lower bound and an upper bound above 0. Therefore, the application of a ReLU cannot be exactly captured- a concrete datapoint could either result in output of 0 from the ReLU or be unaffected. To include all the options in our analysis we use the DeepZono relaxation proposed in [3] and the application of it is shown for $\hat{x}_6$ (for the details of the calculation, interested readers can find the formulation in the paper [3]). The key take away is we now introduce a new error term ϵ_3 which makes the analysis more imprecise. Further, the analysis is also now more complex as we need to track more error terms and this is reflected in the final shape being composed of 3 pairs of parallel lines.

1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021), Virtual Meeting.

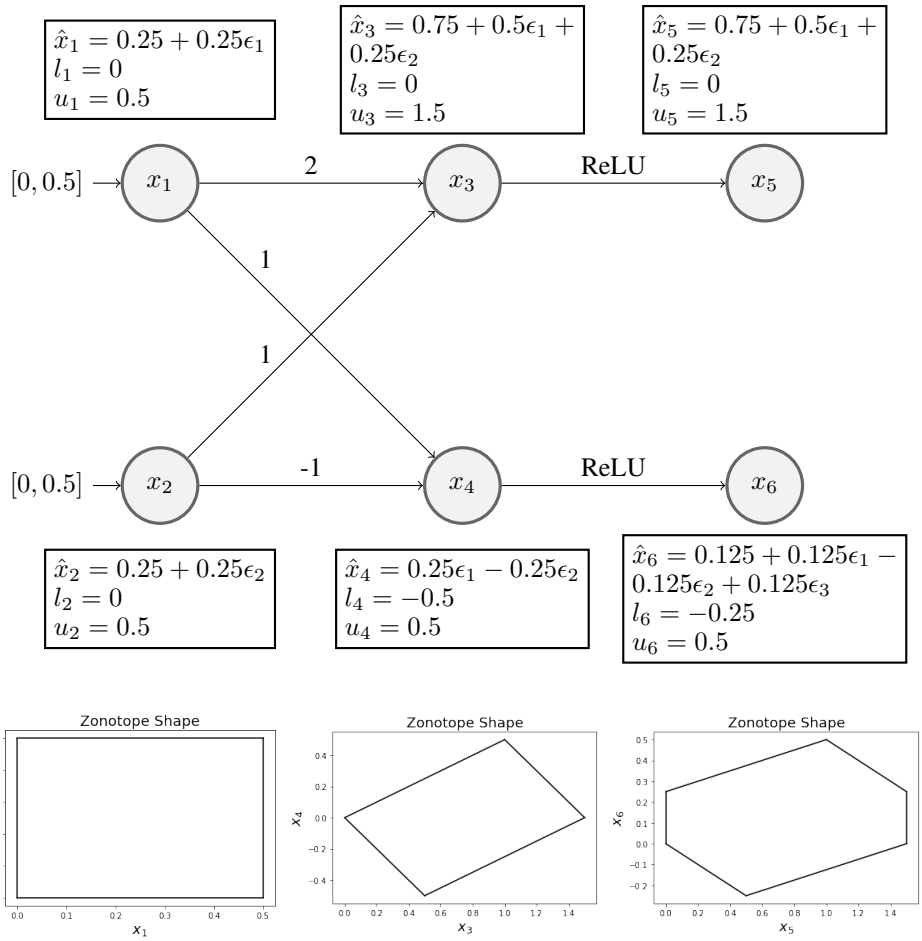
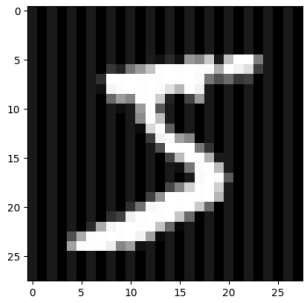


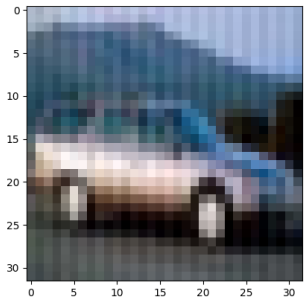
Figure 1: Example of zonotopes being pushed through a toy layer, taking in 2 features and having a total of 4 weights. In each box we state the zonotope component as well as its upper u and lower l bounds.

Appendix B

MNIST and CIFAR10 images with the backdoors used for the results in Table 1. We use L_∞ bounds for the backdoor to follow the attacker model adversarial training is designed to protect against. In other words, if we allowed L_0 perturbations then backdooring to circumvent L_∞ based adversarial training would not even be required as adversarial examples can be found which can evade the adversarial training. Hence, this represents perturbations which falls into what PGD should be able to protect against, and yet backdooring in this style can be used to introduce the hidden weaknesses desired by the attacker.



(a) Vertical stripes of magnitude $L_\infty = 0.1$. Model is adversarially trained to $L_\infty = 0.25$



(b) Vertical stripes of magnitude $L_\infty = 6/255$. Model is adversarially trained to $L_\infty = 8/255$

Figure 2: L_∞ constrained backdoor keys.

Appendix C

Here we have a brief overview of the functioning of defensive distillation and how it can be used as an attack. This was first proposed in the paper [4], and as it involves a few more steps compared to a backdoor style attack we detail the algorithm below.

How do you train a network with defensive distillation? In defensive distillation we train two neural networks: a teacher neural network and a student neural network. We apply a temperature scaling T to the softmax of both networks. We first train the teacher neural network on data-label pairs (x, y). After the teacher neural network has converged we use its predictions on the training dataset to re-label it. The student neural network still with temperature T on the softmax is trained on the re-labeled dataset. Then, once converged, the student neural network's temperature is set back to 1. The defence is completed with the student neural network being the defended model.

How does it protect against attack? The core component is the temperature scaling factor which we apply during training, but remove at test time. At inference, this will case the inputs to the softmax function to be T times larger. Due to this scaling, the output of the softmax will be close to a one hot vector. In fact, the components will be so close to either 0 or 1 that the 32-bit floating-point of non-predicted classes is rounded to 0 while the predicted class has a prediction of ~ 1. For similar reasons the the gradient, when represented as 32-bit floating-point, is rounded to 0 preventing the adversarial attack from making progress [2].

How can an attacker circumvent it?: Defensive distillation is an example of a *gradient masking* defence and so has a few shortcomings. Firstly it is vulnerable against black-box transfer attacks. However, stronger attacks can be performed by an attacker who knows the details of defensive distillation. Specifically, as was noted in [1] the key component of defensive distillation is the temperature scaling to cause vanishing gradients. An attacker can compute the adversarial examples by re-applying the temperature T which re-scales the logits to appropriate values restoring gradient information enabling adversarial attack crafting. After crafting the adversarial examples the attacker then sends them to the original defender model with $T = 1$ and the adversarial examples function just as effectively.

How can it be turned into an attack in FL? In our scenario, we can use defensive distillation to achieve the goals of the attacker set out in Section 2.3. The model will look robust to a defender who evaluates it via standard application of the PGD attack. However, the attacker who knows that this is a defensively distilled model can craft their adversarial examples using temperature scaling as described above breaking the model.

Appendix D

Neural network architectures used for our experiments. We use the following notation to describe our models: Conv$_{(s_w,s_h)}$ C x W x H for a convolutional layer with C output channels, and a kernel of width W and height H. The stride size in width and height is given by s_w and s_h respectively. Additionally, FC N indicates a fully connected layer with N outputs.

MNIST model:

Conv$_{(2,2)}$ 16x4x4 $\rightarrow$ ReLU $\rightarrow$ Conv$_{(2,2)}$ 32x4x4 $\rightarrow$ ReLU $\rightarrow$ FC 1000 $\rightarrow$ ReLU $\rightarrow$ FC 10

CIFAR10 model:

Conv$_{(1,1)}$ 32x3x3 $\rightarrow$ ReLU $\rightarrow$ Conv$_{(2,2)}$ 64x4x4 $\rightarrow$ ReLU $\rightarrow$ Conv$_{(1,1)}$ 64x3x3 $\rightarrow$ ReLU $\rightarrow$ Conv$_{(2,2)}$ 128x4x4 $\rightarrow$ ReLU $\rightarrow$ FC 512 $\rightarrow$ ReLU $\rightarrow$ FC 512 $\rightarrow$ ReLU $\rightarrow$ FC 10

References

[1] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

[2] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[3] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T Vechev. Fast and effective robustness certification. *NeurIPS*, 1(4):6, 2018.

[4] Giulio Zizzo, Ambrish Rawat, Mathieu Sinn, and Beat Buesser. Fat: Federated adversarial training. *arXiv preprint arXiv:2012.01791*, 2020.