Appendix

We present the proof of Theorem 1 and Corollary 2 in App. A followed by a review of bilevel optimization in App. B and our attack algorithm in App. C. In App. D, we present the results of additional experiments on poisoning different classes in the dataset, successful transferability of our poisoning attack to deeper models, performance of the attack when targeting a single test point and effect of using weight regularization on the attack success. We conclude in App. E by providing details of the hyperparameters and models architectures used in the experiments.

A. Proofs

Theorem 1. *If the perturbation is large enough, i.e., $\epsilon \geq \frac{\sum_i x_i^+ - \sum_i x_i^-}{n}$ then there are two locally optimal solutions to (3) which are $u_i = x_i^- - \epsilon$ (Case 1) and $u_i = x_i^- + \epsilon$ (Case 2) for $i = 1, ..., n$. Otherwise, the is a unique globally optimal solution which is $u_i = x_i^- - \epsilon$ (Case 1) for $i = 1, ..., n$.*

Proof. Let $t = -\frac{b}{w}$ be the threshold of the linear classifier. Also let $\Phi(t) := \int_{-\infty}^{t} P_-(x)\ dx$ and $\Psi(t) := \int_{-\infty}^{t} xP_-(x)\ dx$. There are two cases to consider.

Case 1 ($w > 0$): The upper-level cost function is

$$f(t) = \int_{-\infty}^{t} (t - x)P_-(x)\ dx = t\Phi(t) - \Psi(t)$$

Note that the range $[-\infty, t]$ is where classification is correct for the test data. (Certified radius is 0 for misclassified points by definition.)

The closed-from solution of the lower-level problem gives us $t = -\frac{b}{w} = \frac{\sum_i u_i + \sum_i x_i^+}{2n}$, and therefore the perturbation bound $|u_i - x_i^-| \leq \epsilon$ implies $\sum_i x_i^- - n\epsilon \leq \sum_i u_i \leq \sum_i x_i^- + n\epsilon$ and therefore

$$-\frac{\epsilon}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n} \leq t \leq \frac{\epsilon}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n}.$$

Also, the assumption $w > 0$ poses another constraint: $w \propto \sum_i x_i^+ - \sum_i u_i > 0$ and therefore $t = \frac{\sum_i u_i + \sum_i x_i^+}{2n} \leq \frac{\sum_i x_i^+}{n}$. The upper-level problem is therefore

$$\min_t\ f(t) = t\Phi(t) - \Psi(t) \quad \text{s.t.}$$
$$-\frac{\epsilon}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n} \leq t \leq \frac{\epsilon}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n}$$
$$\text{and}\ \ t \leq \frac{\sum_i x_i^+}{n}.$$

Since f is non-decreasing (i.e.,$f'(t) = \Phi(t) + tP_-(t) - tP_-(t) \geq 0$), the minimum is achieved at the left-most boundary $t = -\frac{\epsilon}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n}$ which corresponds to $u_i = x_i^- - \epsilon,\ \ i = 1, ..., n$.

Case 2 ($w < 0$): The upper-level cost function is now

$$f(t) = \int_{t}^{\infty} (-t+x)P_-(x)\ dx = -t(1-\Phi(t))+(1-\Psi(t)),$$

which is non-increasing ($i.e., f'(t) = -(1 - \Phi) + tP_- - tP_- \leq 0$) and has the constraints:

$$-\frac{\epsilon}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n} \leq t \leq \frac{\epsilon}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n}.$$

and

$$t = \frac{\sum_i u_i + \sum_i x_i^+}{2n} \geq \frac{\sum_i x_i^+}{n}.$$

For the solution to be feasible, it is required that $\frac{\sum_i x_i^+}{n} \leq \frac{\epsilon}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n}$, that is $\frac{\epsilon}{2} \geq \frac{\sum_i x_i^+ - \sum_i x_i^-}{2n}$ (remember the assumption $\frac{\sum_i x_i^-}{n} \leq \frac{\sum_i x_i^+}{n}$). Therefore if the perturbation is large enough, i.e., $\epsilon \geq \frac{\sum_i x_i^+ - \sum_i x_i^-}{n}$ holds, then the minimum is achieved at the right-most boundary $t = \frac{\epsilon}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n}$ which corresponds to $u_i = x_i^- + \epsilon,\ \ i = 1, ..., n$. □

Corollary 2. *If the perturbation is large enough, i.e., $\epsilon \geq \frac{\sum_i x_i^+ - \sum_i x_i^-}{n}$ then the reduction in the ACR of the target class, by poisoning an α portion ($\alpha \in [0, 1]$) of the target class, with maximum perturbation $\tilde{\epsilon}$ using Eq. (3) is same as that achieved by poisoning the entire target class with $\epsilon = \alpha\tilde{\epsilon}$.*

Proof. To obtain the effect of poisoning an α portion ($\alpha \in [0, 1]$) of the target class, with maximum perturbation $\tilde{\epsilon}$, we follow the proof of Theorem 1, with the change that the solution to the lower-level problem is $t = -\frac{b}{w} = \frac{\sum_{i=0}^{\alpha n} u_i + \sum_{i=0}^{(1-\alpha)n} x_i^- + \sum_i x_i^+}{2n}$.

Thus the solutions to the upper-level problem are $t = \frac{-\alpha\tilde{\epsilon}}{2} + \frac{\sum_i x_i^+ + \sum_{i=0}^{\alpha n} x_i^- + \sum_{i=0}^{(1-\alpha)n} x_i^-}{2n}$ which corresponds to $u_i = x_i^- - \tilde{\epsilon},\ \ i = 1, ..., \alpha n$ for Case 1 and

$t = \frac{\alpha\tilde{\epsilon}}{2} + \frac{\sum_i x_i^+ + \sum_{i=0}^{\alpha n} x_i^- + \sum_{i=0}^{(1-\alpha)n} x_i^-}{2n}$ which corresponds to $u_i = x_i^- + \tilde{\epsilon},\ \ i = 1, ..., \alpha n$ for Case 2.

The decision boundaries

$t = \frac{-\alpha\tilde{\epsilon}}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n}$ for Case 1 and

$t = \frac{\alpha\tilde{\epsilon}}{2} + \frac{\sum_i x_i^+ + \sum_i x_i^-}{2n}$ for Case 2 are the same boundaries as obtained by poisoning all points from the target class (x_i^-) with $\epsilon = \alpha\tilde{\epsilon}$

□

B. Review of bilevel optimization

A bilevel optimization problem is of the form $\min_{u\in\mathcal{U}} \xi(u, v^*)$ s.t. $v^* = \arg\min_{v\in\mathcal{V}(u)} \zeta(u, v)$, where the upper-level problem is a minimization problem with v constrained to be the optimal solution to the lower-level problem. General bilevel problems are difficult to solve but if the solution to the lower-level problem can be computed in closed form then we can replace the lower-level problem with its solution, reducing the bilevel problem into a single level problem. We can then use the gradient-based methods to solve the single level problem. The total derivative $\frac{d\xi}{du}(u, v^*(u))$ (hypergradient) using the chain rule is

$$\frac{d\xi}{du} = \nabla_u \xi + \frac{dv}{du} \cdot \nabla_v \xi.$$

Since $\nabla_v \zeta = 0$ at $v = v^*(u)$ and assuming $\nabla^2_{vv}\zeta$ is invertible we can compute $\frac{dv}{du}$ using the implicit function theorem (this can be done even if the solution to lower-level problem can't be found in closed form) which gives

$$\frac{dv}{du} = -\nabla^2_{uv}\zeta(\nabla^2_{vv}\zeta)^{-1}.$$

Thus the hypergradient is

$$\frac{d\xi}{du} = \nabla_u \xi - \nabla^2_{uv}\zeta(\nabla^2_{vv}\zeta)^{-1}\nabla_v \xi \text{ at } (u, v^*(u)).$$

Since computation of $(\nabla^2_{vv}\zeta)^{-1}$ is difficult, [9, 27] proposed to instead approximate the solution to $q = (\nabla^2_{vv}\zeta)^{-1}\nabla_v\xi$ by approximately solving the linear system of equations $\nabla^2_{vv} \cdot q \approx \nabla_v \xi$. This can be done by minimizing $\|\nabla^2_{vv}\zeta \cdot q - \nabla_v \xi\|$ using any iterative solver. Other methods for solving the bilevel optimization problems include using forward/reverse mode differentiation [10, 22, 30] to approximate the inverse and penalty method [24] to solve the single level problem as a constrained minimization problem.

C. Attack algorithm

Alg. 1 shows the complete algorithm used to generate the poisoning attack when RS is used for certification and models are trained using GA. The algorithm relies on ApproxGrad (Alg. 2) to solve the bilevel optimization problem. The upper-level cost is a differentiable function that approximates the certified radius of the hard smooth classifier using a soft smooth classifier. The hyperparameter α is the inverse temperature parameter of softmax. As $\alpha \to \infty$, softmax converges to argmax almost everywhere. As a result $\tilde{g}_\theta$ converges to g_θ almost everywhere and thus soft randomized smoothing converges to hard randomized smoothing almost everywhere. Although, in this work we considered RS as the procedure for certification (due to its scalability to large models and datasets), any other certification procedure can be used as the upper-level cost as long as its differentiable. Moreover, Alg. 1 uses $\mathcal{L}_{\text{GaussAug}}$ in the lower-level to train the model, but like the case with upper-level cost any other loss function can be used to obtain the model parameters. This flexibility of our method allows us to generate poison data against MACER and SmoothAdv using their loss functions in the lower-level.

Algorithm 2 Algorithm for ApproxGrad

Input: $\xi, \zeta, M, T_1, T_2, \epsilon, u_{base}, \{\tau_m = 0.1\}$, $\{\rho_{m,t_1} = 0.001\}, \{\beta_{m,t_2} = 0.001\}$
Output: (u_K)
Initialize u_0, v_0 randomly
Begin
for $m = 0, \cdots, M$-1 **do**

# Approximately solve the lower-level problem
for $t = 0, \cdots, T_1$-1 **do**
$v_{t+1} \leftarrow v_t - \rho_{m,t_1}\nabla_v \zeta$
end for

# Approximately solve the linear system
# $\nabla^2_{vv}\zeta \cdot q_k = \nabla_v \xi$
for $t = 0, \cdots, T_2$-1 **do**
$q_{t+1} \leftarrow q_t - \beta_{m,t_2}\nabla_q(\|\nabla^2_{vv}\zeta \cdot q_m - \nabla_v \xi\|)$
end for

Compute the approximate Hypergradient
$p_m = \nabla_u \xi - \nabla^2_{uv}\zeta \cdot q_{T_2}$

Update u_m and use projection for the
upper-level constraint
$u_{m+1} = P(u_m - \tau_m p_m, \epsilon, u_{base})$

end for

C.1. ApproxGrad

For an unconstrained bilevel problem of the form $\min_u \xi(u, v^*)$ s.t. $v^* = \arg\min_v \zeta(u, v)$, if $\zeta(u, v)$ is strongly convex then we can replace the lower-level problem with its necessary condition for optimality and write the bilevel problem as the following single level problem $\min_u \xi(u, v^*)$ s.t. $\nabla_v \zeta(u, v) = 0$. Assuming $\nabla^2_{vv}\zeta$ is invertible everywhere we can compute the hypergradient at the point $(u, v^*(u))$ as $\frac{d\xi}{du} = \nabla_u \xi - \nabla^2_{uv}\zeta(\nabla^2_{vv}\zeta)^{-1}\nabla_v \xi$.

The ApproxGrad algorithm approximates the Hessian-inverse vector product by approximately solving a system of linear equation using an iterative solver such as gradient descent or conjugate gradient method. In this work we use Adam optimizer to solve this system. Since our problem for data poisoning in Eq. (2) involves a constraint in the upper-level we use projection to enforce the constraint. The

full algorithm for solving the bilevel optimization problem using ApproxGrad is present in Alg. 2. For our attack the lower-level problem involves a deep neural network, which can have multiple local minima and thus optimizing against a single local minima in the bilevel problem is not ideal. To overcome this problem we reinitialize the lower-level variable v after few upper-level iterations to prevent the poisoning points from overfitting to a particular local minima. Empirically, this helps us find poisoning points that remain effective even after the model is retrained from scratch making them generalize to different initialization of the neural network.

D. Additional experiments

D.1. Comparison with standard data poisoning

The standard data poisoning attack creates poison data so that the accuracy of the victim's model trained on it is significantly lower than the accuracy attainable with training on clean data. The bilevel optimization problem for this attack is as follows.

$$\begin{aligned} &\min_{u} \quad \mathcal{L}_{\text{standard}}(\mathcal{D}^{\text{val}}) \\ &\text{s.t.} \quad \|\delta_i\|_\infty \leq \epsilon, \quad i = 1, ..., n, \quad \text{and} \\ &\theta^* = \arg\min_{\theta} \; \mathcal{L}_{\text{standard}}(\mathcal{D}^{\text{clean}} \bigcup \mathcal{D}^{\text{poison}}; \theta). \end{aligned} \tag{4}$$

Here $\mathcal{L}_{\text{standard}}(\mathcal{D}; \theta) = \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} l_{ce}(x_i, y_i; \theta)$, where l_{ce} is the cross entropy loss. We used this formulation to generate the poisoned dataset for reporting the results in Table 1 with $\epsilon = 0.1$ for MNIST and $\epsilon = 0.03$ for CIFAR10. The attack modifies all the points in the target classes. Specifically, our attack targets misclassification of the digit 8 in MNIST and class "Ship" in CIFAR10. The poisoned dataset obtained after solving the bilevel optimization was then used to train five models starting from random initializations with different training procedures. The results of which are reported in Table 1. As expected the models trained with standard training on the poisoned data perform the worst in terms of accuracy since the attack was optimized against standard training. However, the generated poison data has little to no effect when a training procedures that improves certified adversarial robustness is used. This shows that the effect of standard data poisoning can easily be nullified if a victim trains the model with a these training procedures. This gives a false sense of security of the models trained with certified defenses to data poisoning attacks. Thus, in this work we study the effect of poisoning on training procedures meant to improve certified adversarial robustness and show that their guarantees become meaningless when the dataset is poisoned.

Table 6. Degradation of certified adversarial robustness of logistic regression trained with GA on a toy 2-isotropic Gaussians dataset.

σ	Certified Robustness on clean data		Certified Robustness on poisoned data	
	ACR	ACA(%)	ACR	ACA(%)
0.25	0.4047	90.00	0.3585	88.00
0.50	0.4139	90.00	0.3587	87.60
0.75	0.4123	90.00	0.3544	87.60

D.2. Isotropic Gaussians

Here we validate the solution found by solving the bilevel optimization against the analytical solution of a toy problem. Consider a two-dimensional dataset comprising of points drawn from two isotropic Gaussian distributions. Let $\mathbb{P}(x|y = -1)) = \mathcal{N}(\mu_1, \sigma^2 I)$ and $\mathbb{P}(x|y = 1) = \mathcal{N}(\mu_2, \sigma^2 I)$ and equal prior $\mathbb{P}(y = 1) = \mathbb{P}(y = -1)$. For a point x, the Bayes optimal classifier predicts $y = 1$ if $\mathbb{P}(y = 1|x) >= \mathbb{P}(y = -1|x)$ and predicts $y = -1$ otherwise. The decision boundary of the Bayes optimal classifier is given by $(\mathbf{x} - \mu_\mathbf{1})^\mathbf{T}(\mathbf{x} - \mu_\mathbf{1}) = (\mathbf{x} - \mu_\mathbf{2})^\mathbf{T}(\mathbf{x} - \mu_\mathbf{2})$. This is also the decision boundary of the smoothed classifier. Assuming the attacker is poisoning the class with label -1 and maximum permissible distortion is ϵ, our analysis showed that maximum reduction in radius occurs if the entire distribution shifts by ϵ i.e. the new mean of the class with label -1 is $\mu_1 - \epsilon$ and the decision boundary is $(\mathbf{x} - (\mu_\mathbf{1} - \epsilon))^\mathbf{T}(\mathbf{x} - (\mu_\mathbf{1} - \epsilon)) = (\mathbf{x} - \mu_\mathbf{2})^\mathbf{T}(\mathbf{x} - \mu_\mathbf{2})$. Since the test distribution is unchanged, the ACR for the test points with labels -1 is reduced by $\frac{\epsilon}{\sqrt{2}}$. Using $\mu_1 = 0.2, \mu_2 = 0.8, \sigma_1 = \sigma_2 = 0.3, \epsilon = 0.1$ and using logistic regression in the lower-level, analytically the certified radius must decrease from 0.4243 to 0.3546. The solution by solving the bilevel optimization numerically (Table 6) matches the analytic solution.

D.3. Targeting other classes

In this section we present the results of our poisoning attack on different target classes where the models are trained using GA during poison generation and evaluation. Since MNIST and CIFAR10 both have 10 classes we create 10 poisoning sets each targeting a particular class. The results of retraining models from five random initializations on each of the 10 poisoning sets are summarized in Fig. 6. Reduction in average certified radius for all classes shows that an attacker can generate poison data to affect any class in the dataset.

D.4. Transferability to different architectures

Here we present the results of transferability of the poisoned data generated against Resnet-20 targeting the class "Ship" to bigger models. In particular we present the results on Resnet-56 and Resnet-110 [14] models in Fig. 7. As seen from the results the poisoned data generated against

(a) Clean data (odd numbered rows) and poisoned data generated by our attack (even numbered rows) for all digits in MNIST

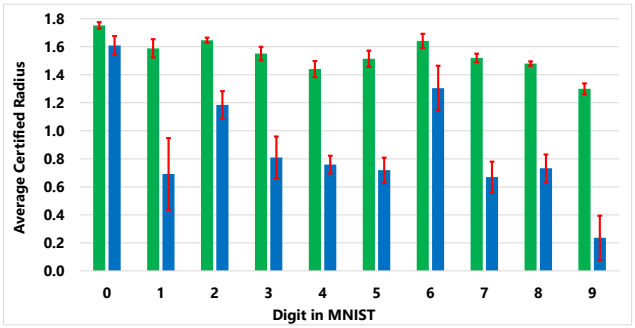
(b) Clean data (odd numbered rows) and poisoned data generated by our attack (even numbered rows) for all classes of CIFAR10

Figure 5. Imperceptibly distorted poison data generated by our algorithm against Gaussian augmented training which causes a significant reduction in the certified robustness guarantees of the models. The average certified radius and certified accuracy of models trained on clean and poisoned data are reported in App. D.3 and Fig. 6.

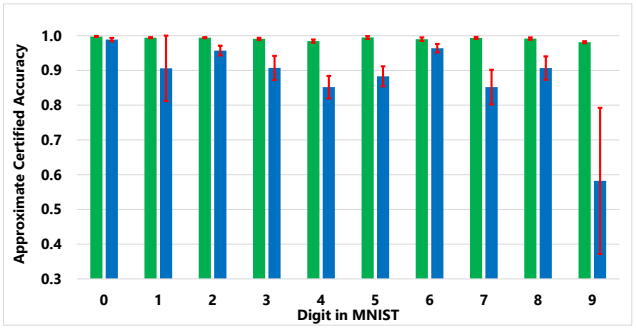
Resnet-20 is successful in reducing the certified radius of the target class even if the victim uses a larger model. We report the results of training the models on clean and poisoned data starting from three random initializations and certify using 500 randomly sampled points of the target class from the clean test set. Our results suggest that the poisoned data generated using our procedure are agnostic to the training procedure (Fig. 4), model (Fig. 7) and metric (RS or empirical robustness) used by the victim during evaluation highlighting the threat of data poisoning.

D.5. Effect of weight regularization

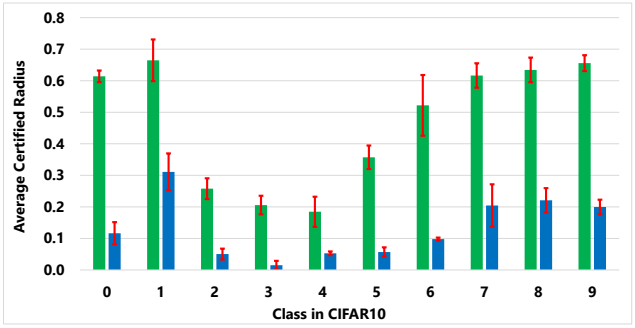
Previous works [6] have shown weight regularization to mitigate the effect of data poisoning attacks. Here, we evaluate the attack success when using different coefficients for weight regularization in standard and GA training. Results in Table 7 show that our attack significantly reduces the ACR of models, especially those trained without GA or weight regularization. Similar to previous works [6], we see that models trained with large regularization (without GA) are difficult to poison. This increased robustness, however comes at the cost



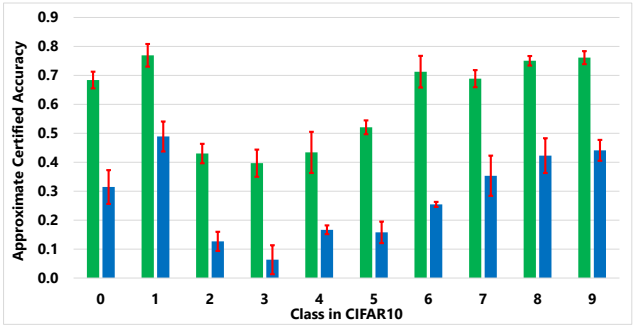
(a) Average certified radius of all digits in MNIST



(b) Approximate certified accuracy of all digits in MNIST

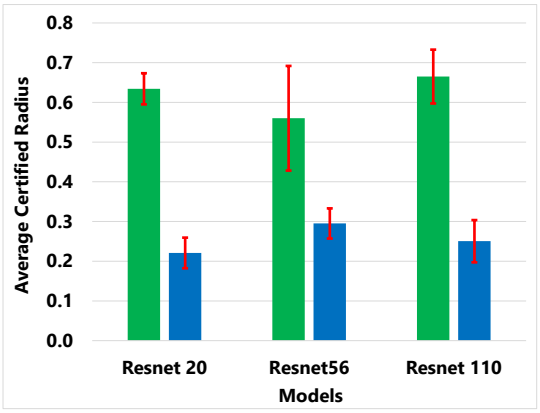


(c) Average certified radius of all classes in CIFAR10

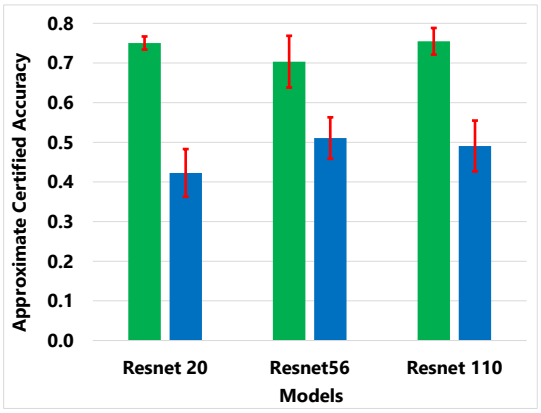


(d) Approximate certified accuracy of all classes in CIFAR10

Figure 6. Successful poisoning attack against all classes in MNIST and CIFAR10 dataset.



(a) Average certified radius of class "Ship" in CIFAR10



(b) Approximate certified accuracy of class "Ship" in CIFAR10

Figure 7. Successful transferability of poisoning attack against deeper models. The poison data is optimized against Resnet-20. GA with $\sigma = 0.5$ is used for poison generation and evaluation.

of accuracy (target class accuracy of models trained with clean data drops from 99% to 92%), suggesting a trade-off. On the other hand, for models trained with GA, using large regularization leads to a significant drop in their certified robustness guarantees, even without poisoning (ACR for models trained on clean data drops from 1.48 to 0.85), there by making large regularization undesirable to use with GA. This shows that our attack remains quite effective even when different amounts of weight regularization are used during model retraining.

D.6. Attack success by poisoning 1% of the data

Similar to [16], where the effect of poisoning 1% of the training data is evaluated on a single test point, we randomly

Table 7. Effect of weight regularization and Gaussian data augmentation (σ=0.5) on ACR of digit 8 of MNIST (ϵ=0.1). Mean and s.d. of 3 random initializations. Bold entries are reported in Table 2.

Training	Data	No Reg.	1E-4	1E-2	1E-1
Without GA	Clean	0.95±0.10	0.89±0.06	0.87±0.11	0.82±0.04
	Poisoned	0.01±0.01	0.03±0.05	0.37±0.06	0.68±0.05
With GA	Clean	**1.48±0.02**	1.49±0.03	1.29±0.15	0.85±0.09
	Poisoned	**0.73±0.10**	0.62±0.02	0.58±0.11	0.72±0.08

select 5 test images of the bird class and generate poison data to reduce their certified radius individually. Using Alg. 1, we poison 500 bird images (1% of CIFAR10) closest to the target, with ϵ=0.06. Using 3 randomly initialized models trained with GA, our attack can reduce the certified radius of 5 targets from 0.63 to 0.26 on average.

E. Details of the experiments

All codes are written in Python using Tensorflow/Keras, and were run on Intel Xeon(R) W-2123 CPU with 64 GB of RAM and dual NVIDIA TITAN RTX. Implementation and hyperparameters are described below.

E.1. Data splits

For MNIST, we use 55000 points as the training data and 5000 points for validation data. We have roughly 500 points belonging to the target class in the validation set which is used in the upper-level problem of the bilevel optimization presented in Eq. (1). For CIFAR10, we use 45000 points as the training data and 5000 points for validation data. Similar to MNIST we have roughly 500 points belonging to the target class in the validation set. The test sets of both the datasets comprises of 10000 points. We use 500 randomly sampled points of the target class from the test set to report the results of certified and empirical robustness of the models trained on clean and poisoned data.

E.2. Model Architecture

For the experiments on the MNIST dataset, our network consists of a convolution layer with kernel size of 5x5, 20 filters and ReLU activation, followed by a max pooling layer of size 2x2. This is followed by another convolution layer with 5x5 kernel, 50 filters and ReLU activation followed by similar max pooling and dropout layers. Then we have a fully connected layers with ReLU activation of size 500. Lastly, we have a softmax layer with 10 classes. The accuracy of the model on clean data when optimized with the Adam optimizer using a learning rate of 0.001 for 100 epochs with batch size of 200 is 99.3% (without GA). For the experiments on the CIFAR10 dataset, we use the Resnet-20 model. The accuracy of the model on clean data when optimized with the Adam optimizer using a learning rate of 0.001 for 100 epochs with batch size of 200 is 85% (without GA). For all CIFAR10 experiments except for the experiments with SmoothAdv, we trained the models using data augmentation (random flipping and random cropping). We used the same parameters for training the models with different robust training procedures on clean and poisoned data.

E.3. Hyperparameters

For experiments with MNIST we used $\epsilon = 0.1, K = 20, \alpha = 16$. The batch size used for lower-level training was 1000, of which 100 points belonged to the poisoned set (target class). The batch size for validation set was 100 which only consisted of points from the target class. The lower-level was trained using different training procedures on clean and poisoned data. For experiments with CIFAR10 we used $\epsilon = 0.03, \lambda = 0.06, M = 20, \alpha = 16$. The batch size used for lower-level training was 200, of which 20 points belonged to the poisoned set (target class). The batch size for validation set was 20 which only consisted of points from the target class. For training with GA the lower-level is trained with a single noisy image of the clean and poisoned dataset. The same setting is used while retraining. For generating poison data against MACER the lower-level is trained with $K = 2, \lambda = 1, \gamma = 8$. During retraining, $K = 16, \lambda = 16, \gamma = 8$ are used for MNIST. For CIFAR10, we use $K = 16, \gamma = 8$ and $\lambda = 12$ for $\sigma = 0.25$ and $\lambda = 4$ for $\sigma = 0.5$. The hyperparameters during retraining are similar to the ones used in the original work. For Smoothadv, $k = 1$ and 2-step PGD attack are used to generate adversarial examples of the smooth classifier. These adversarial examples along with GA are used to do adversarial training during poison generation and retraining.

In our experiments with generating poisoned data against GA and MACER we used $P = 50, T_1 = T_2 = 10, \tau = 0.1, \rho = 0.001, \beta = 0.01$ for ApproxGrad. We used all the same hyperparameters for SmoothAdv except $T_1 = 1$. For certification we used the CERTIFY procedure of [8], with $n_0 = 100, n = 100000, \alpha = 0.001$. For measuring empirical robustness of the smoothed classifier, we used the mean ℓ_2 distortion required by PGD attack to generate an adversarial example as done in [29]. The attack is optimized for 100 iterations for different values of ℓ_2 distortion between (0.01, 10). We used 20 augmentations for each test point of MNIST and 10 for CIFAR10. To report the results for empirical robustness we record the minimum distortion for a successful attack for each test point.

For the watermarking baseline, we randomly selected an image ($other$) from the classes other than the target class and over-layed them on top of the target class images ($base$) with an opacity of $\gamma = 0.1$ i.e. ($poison\_image = \gamma \cdot other + (1 - \gamma) \cdot base$). We then clip the images to have ℓ_∞ distortion of ϵ to make our bilevel attack comparable in terms of maximum distortion.