A COLLIDER Details

In this section, first we show how the coreset selection objective of Eq. (6) can be turned into a submodular maximization equivalent. This re-formulation enables us to use greedy algorithms to solve the coreset selection objective. Afterwards, we present the full COLLIDER algorithm in detail.

A.1 The Submodular Maximization Equivalent of the Coreset Selection Objective [25,26]

Without loss of generality, we use the derivation of Mirzasoleiman *et al.* [26] for the basic coreset selection objective of Eq. (5). The equivalency of Eq. (6) to submodular maximization can be deduced similarly.

A set function $F : 2^{|V|} \to \mathbb{R}^+$ is called *submodular* if for any $S \subset T \subset V$ and $e \in V \backslash T$ we have [26]

$$F\left(S \cup \{e\}\right) - F\left(S\right) \geq F\left(T \cup \{e\}\right) - F\left(T\right).$$

Moreover, if for any $S \subset V$ and $e \in V \backslash S$ we have a non-negative $F\left(e|S\right)$, then $F(\cdot)$ is called *monotone* [26].

Now, assume that we can find a *constant upper-bound* for all the $d_{ij}(\boldsymbol{\theta})$ values from Eq. (5). If we denote this constant upper-bound with d_0, then Eq. (5) can be re-written as [26]

$$S^*(\boldsymbol{\theta}) \in \arg\max_{S \subseteq V} \sum_{i \in V} \max_{j \in S} d_0 - d^u_{ij}(\boldsymbol{\theta}) \quad \text{s.t.} \quad |S| \leq k, \tag{8}$$

where $d^u_{ij}(\boldsymbol{\theta}) = \left\| \Sigma'_L\left(z_i^{(L)}\right) \nabla \ell_i^{(L)}(\boldsymbol{\theta}) - \Sigma'_L\left(z_j^{(L)}\right) \nabla \ell_j^{(L)}(\boldsymbol{\theta}) \right\|$ is the $\boldsymbol{\theta}$-*dependent upper-bound* of $d_{ij}(\boldsymbol{\theta})$ from Eq. (7). Mirzasoleiman *et al.* [26] argue that the function

$$F(S, \boldsymbol{\theta}) = \sum_{i \in V} \max_{j \in S} d_0 - d^u_{ij}(\boldsymbol{\theta})$$

is a *monotone submodular* set function. As such, Eq. (8) is equivalent to a well-known submodular maximization problem widely known as the *facility location*, and there exist efficient greedy algorithms that can solve it sub-optimally.[4]

A.2 COLLIDER Final Algorithm

Algorithm 1 shows the COLLIDER framework for training neural networks on backdoor poisoned data in detail.

[4] In our experiments, we use the implementation of Mirzasoleiman *et al.* [26] available online.

Algorithm 1 COLLIDER for training DNNs on backdoor poisoned data

Input: dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, neural network $f_{\boldsymbol{\theta}}(\cdot)$.

Output: robustly trained neural network $f_{\boldsymbol{\theta}}(\cdot)$.

Parameters: number of classes C, total epochs T, batch size b, coreset size k, LID start epoch l, LID number of neighbors N, LID moving average window w, LID Lagrange multiplier λ.

Initialize $\boldsymbol{\theta}$ randomly.
for $t = 1, 2, \ldots, T$ **do**
  $S^t = \emptyset$
  for $c = 1, 2, \ldots, C$ **do**
    $\mathcal{D}^c = \cup_{i=1}^{n} \{(\boldsymbol{x}_i, y_i) \mid y_i = c\}$.
    $\mathcal{Y}^c = \{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \mid (\boldsymbol{x}_i, y_i) \in \mathcal{D}^c\}$.
    $\mathcal{G}^c = \text{GRADIENTUPPERBOUNDS}\,(\mathcal{D}^c, \mathcal{Y}^c)$. `\\ using Eq. 7`
    if $t \geq l$ **then**
      $\text{LID}^c = \text{GETLID}\,(\mathcal{Y}^c, \text{neighbors} = N)$.
      $\text{LID}^c = \text{MOVINGAVERAGE}\,(\text{LID}^c, \text{window} = w)$.
      $m = (1 - k)\,|\mathcal{D}^c| \,/\, (T - l)$.
      $idx = \text{TOPKARGMAX}\,(\text{LID}^c, \text{K} = m)$.
      Remove data with indices idx from $\mathcal{D}^c$ and $\mathcal{D}$.
      $\mathcal{G}^c = \text{UPDATECOEFFS}\,(\mathcal{G}^c, \lambda, \text{LID}^c)$. `\\ using Eq. 6`
    end if
    $S_c^t = \text{GREEDYSOLVER}\,(\mathcal{D}^c, \mathcal{G}^c, \text{coreset size} = k)$.
    $S^t = S^t \cup S_c^t$.
  end for
  Update neural network parameters $\boldsymbol{\theta}$ using stochastic gradient descent on S^t.
end for

B Implementation Details

In this section, we provide the details of our experiments including hyperparameters, model architectures, and backdoor poisoning settings. Note that all of the experiments were run using a single NVIDIA Tesla V100-SXM2-16GB GPU.

Datasets and Backdoor Poisoning Settings. We used CIFAR-10 [58], SVHN [59], and 12 randomly selected classes of ImageNet [1] for our experiments. We padded CIFAR-10 and SVHN datasets with 4 zero pixels added to both sides of the image, and then randomly cropped each instance such that the final images become of size 32×32. For ImageNet-12, the zero-padding was done via 28 pixels, and the final image size after random cropping was set to 224×224. In each case, we randomly selected a *target class*, and poisoned a fraction of the training data in that class with their backdoor counterpart. The ratio of the poisoned examples in the target class is denoted as the *injection rate*. After injecting the poisoned data, we randomly chose a portion of the training data as our clean held-out validation set. This data was used for model selection. Finally, we used BadNets [6] with checkerboard pattern, label-consistent attacks [31], sinusoidal strips [32], and HTBA triggers [37] as our backdoor data poisoning rules.[5] Samples of each poisoned data can be found in Fig. 4. Tab. 3 summarizes the settings of each dataset used in our experiments.

Model Architecture & Training Hyperparameters. We used stochastic gradient descent (SGD) optimizers in our experiments. The momentum and weight decay for all datasets were set to 0.9 and 5e-4, respectively. For CIFAR-10 and SVHN datasets, we train ResNet-32 [60] models for 120 epochs. The initial learning rate was set to 0.1, which was later divided by 10 at epochs 80 and 100. For ImageNet-12, ResNet-18 [60] neural networks were trained for 200 epochs. In this case, the initial learning rate was also set to 0.1, and it was later divided by 10 at epochs 72 and 144. For training with coresets, the ratio of the data selected from each class is denoted as the *coreset size*. Moreover, for COLLIDER, the epoch at which the LID is enabled is denoted as the *LID start epoch*. For the LID term in COLLIDER (Eq. (6)), a Lagrange multiplier is also required. This hyperparameter, denoted by λ, was set to 0.01 after tuning for CIFAR-10 dataset, and kept fixed through the other experiments. Tab. 4 shows all the hyperparameters used for training the neural networks in our experimental study.

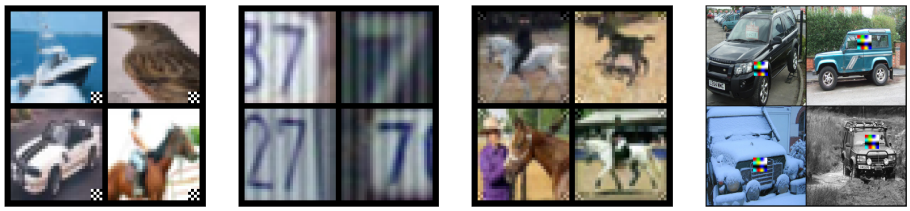
C Extended Experimental Results

In this section, we present our extended simulation results.

[5] For label-consistent attacks [31], we used the poisoned data provided by the official repository. In particular, we used the adversarial data generated with ℓ_2-bounded perturbations, where the bound is set to 300.

Table 3: Details of the datasets used in our experiments.

Dataset	Image Size	Zero-padding	Backdoor	Target Class	Injection Rate	Val. Data Ratio
CIFAR-10	32×32	4	BadNets	Airplane	10%	4%
CIFAR-10	32×32	4	Label Consist.	Horse	10%	4%
SVHN	32×32	4	Sin. Strips	Digit 7	10%	4%
ImageNet-12	224×224	28	HTBA Triggers	Jeep, land-rover	40%	20%



BadNets Sinusoidal Strips Label-consistent HTBA

Fig. 4: Samples of backdoor data poisonings used in experiments.

Performance Measures. We use three measures to compare the performance of each training algorithm on the backdoor poisoned data. First, we compute the accuracy of each model on the clean test set, and denote it with ACC. Next, we evaluate the robustness of each method by the attack success rate (ASR). To compute this quantity, we first remove all the data that originally belong to the target class. Then, we install the trigger used in each case to poison the rest of the test set. We then compute the accuracy of the poisoned data, which should lead the model to output the target class. Finally, we evaluate the purity of our coresets. To this end, we define a quantity which we call the *filtered poison data*. In a nutshell, this measure shows the portion of the poisoned data that is left out of the coreset. This value is always between zero and one. Zero means that all the poisoned data is in our selected coreset. In contrast, one shows that the selected coreset is free of any poisoned data. To ensure reproduciblity, each experiment is repeated with 5 different random seeds.[6] Unless specified otherwise, in each case we report the mean alongside an errorbar which is the standard deviation across these 5 seeds. This errorbar is shown by a shaded area or bars in plots.

ImageNet-12 Results. Tab. 5 shows our experimental results for ImageNet-12 dataset. Like all the previous small size image datasets, COLLIDER gives the best robustness against backdoor data poisonings in this case.

WANet [36] Results. As suggested by the reviewers, we run a similar experiment to Tab. 5 over CIFAR-10 dataset that was poisoned with WANet [36]. As seen in Tab. 6, our approach leads to a significant reduction of the attack success rate in this state-of-the-art attack. Note that for this case, we just ran our algorithm without any rigorous hyperparameter tuning.

[6] Our implementation can be found in this repository.

Table 4: Training hyperparameters used in our experiments.

Hyperparameter	Dataset	
	CIFAR-10 & SVHN	ImageNet-12
Optimizer	SGD	SGD
Scheduler	Multi-step	Multi-step
Initial lr.	0.1	0.1
lr. decay (epochs)	10 (80, 100)	10 (72, 144)
Batch Size	128	32
Epochs	120	200
Model arch.	ResNet-32	ResNet-18
Coreset size	0.4 for Sin. Strips, 0.3 otherwise	
LID Start Epoch	30 for BadNets and Sin. Strips, 50 for Label-consist. and HTBA	
λ	0.01	

Table 5: Clean test accuracy (ACC) and attack success rate (ASR) in % for HTBA [37] backdoor triggers on ImageNet12 [1] dataset. The results show the mean and standard deviation for 5 different seeds. In this case, 40% of the data in the target class contains backdoor poisoned data.

Training	HTBA [37]	
	ACC	ASR
Vanilla	91.43 ± 0.52	49.36 ± 14.42
Coresets	87.29 ± 0.50	37.63 ± 4.84
COLLIDER	85.15 ± 0.32	19.11 ± 3.05

Table 6: Clean test accuracy (ACC) and attack success rate (ASR) in % for WANet [36] data poisonings on CIFAR-10. The results show the mean and standard deviation for 5 different seeds. The poisoned data injection rate is 40%. In this case, the coreset size is 0.4.

Training	WANet [36]	
	ACC	ASR
Vanilla	91.63 ± 0.28	92.24 ± 1.74
Coresets	86.04 ± 0.89	5.73 ± 2.78
COLLIDER	84.27 ± 0.55	4.29 ± 2.54

Semi-supervised Learning on Non-coreset Data. Instead of excluding non-coreset data from the training process, one can add them as a set of unlabeled data and exploit semi-supervised learning to train the model. To this end, we use MixMatch [61] as our semi-supervised learning approach.[7] Moreover, we update the neural network weights using an exponential moving average, and change the optimizer to Adam [62] to comply with the MixMatch implementation that we use. Apart from these tweaks, all the COLLIDER hyperparameters were kept similar to their original settings.

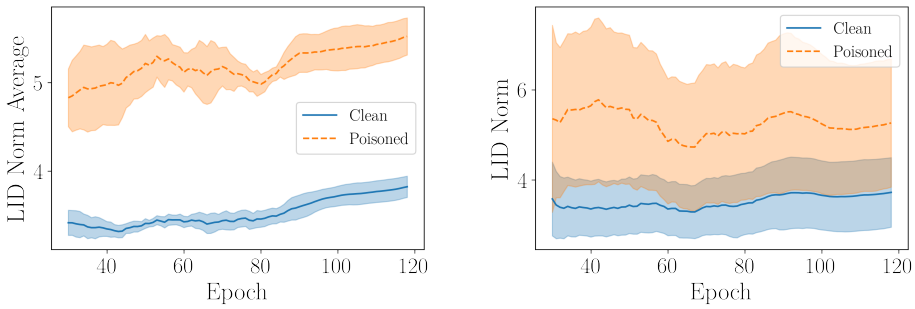
We perform semi-supervised learning (SSL) on the coreset selection as well as COLLIDER. To differentiate the effect of semi-supervised learning from coreset selection, we also randomly select a fraction of the data and remove their labels during each epoch and run MixMatch [61] on them.

Tab. 7 shows our results using semi-supervised learning. For reference, our results without using MixMatch is also shown at the first three rows of Tab. 7. As seen, using MixMatch we can improve the clean accuracy gap with the vanilla training on BadNets [6] and Label-consistent [31] attacks. This use, however, has an adverse effect on Sinusoidal Strips [32].

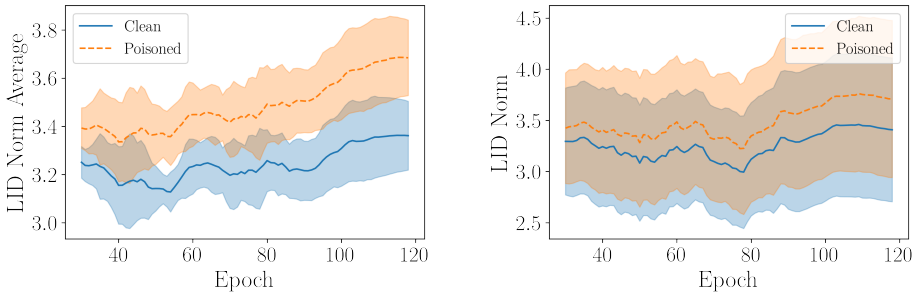
LID of Poisoned vs. Clean Data. Fig. 5 shows the LID values for clean and poisoned data samples, which are poisoned by BadNets, label-consistent attacks, and sinusoidal strips, respectively. As seen, there definitely is a difference between clean and poisoned data in terms of the local intrinsic dimensionality. This difference becomes less severe for label-consistent attacks that use reduced-intensity triggers. Still, the right tail of the LID distribution in all the cases consist of poisoned data only. This observation justifies our choice of throwing them away permanently from the training set.

Coreset Size Effects. Fig. 6 shows the recorded performance measures throughout the training for basic coreset selection (Eq. (5)) and COLLIDER. In each case, we vary the coreset size to see how the underlying framework changes its

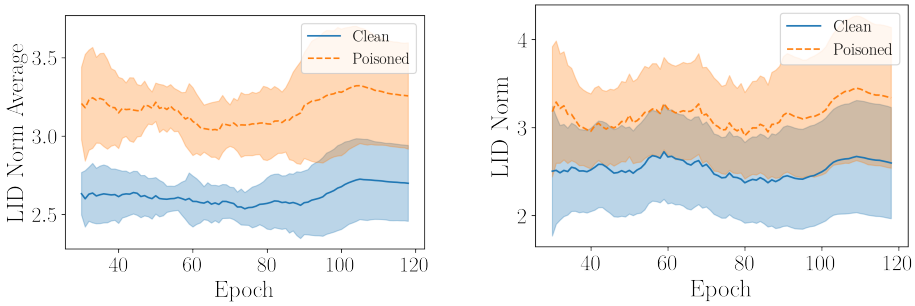
[7] We use the PyTorch implementation of MixMatch available here.



(a) BadNets



(b) Label-consistent Attack



(c) Sinusoidal Strips

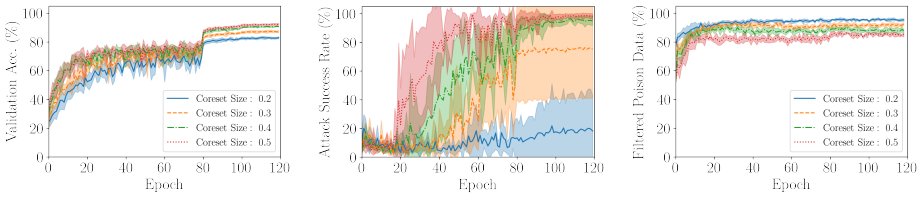
Fig. 5: The LID of clean and backdoor poisoned data samples. Left: average LID norm across 5 different seeds. Right: LID distribution for a single run.

Table 7: Clean test accuracy (ACC) and attack success rate (ASR) in % for backdoor data poisonings on CIFAR-10 (BadNets and label-consistent) and SVHN (sinusoidal strips) datasets. The results show the mean and standard deviation for 5 different seeds. The poisoned data injection rate is 10%. For BadNets and label-consistent attacks, the coreset size is 0.3. It is 0.4 (0.2) for sinusoidal strips (+SSL).

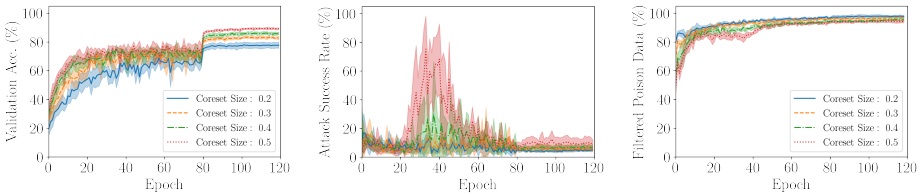
Training	BadNets [6]		Label-consistent [31]		Sinusoidal Strips [32]	
	ACC	ASR	ACC	ASR	ACC	ASR
Vanilla	92.19 ± 0.20	99.98 ± 0.02	92.46 ± 0.16	100	95.79 ± 0.20	77.35 ± 3.68
Coresets	84.86 ± 0.47	74.93 ± 34.6	83.87 ± 0.36	7.78 ± 9.64	92.30 ± 0.19	24.30 ± 8.15
COLLIDER	80.66 ± 0.95	4.80 ± 1.49	82.11 ± 0.62	5.19 ± 1.08	89.74 ± 0.31	6.20 ± 3.69
Random Subset + SSL	91.32 ± 0.35	99.57 ± 0.49	91.48 ± 0.16	100	95.71 ± 0.10	65.22 ± 1.89
Coresets + SSL	88.36 ± 0.26	4.45 ± 0.34	87.82 ± 0.18	6.56 ± 2.87	91.12 ± 0.74	42.62 ± 3.88
COLLIDER + SSL	84.67 ± 0.62	4.10 ± 0.58	84.33 ± 0.37	2.62 ± 0.22	90.53 ± 0.45	31.17 ± 7.69

behavior. As expected, by reducing the coreset size we will have cleaner coresets, and hence, get lower attack success rate. However, there is a trade-off between the coreset size and the clean accuracy, where by reducing the coreset size the validation accuracy also drops. Another important insight that can be taken from Fig. 6 is that the LID regularization almost closes the gap in terms of the filtered poison data between different coreset sizes. Furthermore, it can be seen that the LID regularization is crucial to reduce the attack success rate. Finally, Fig. 7 also shows the aforementioned trade-off between the clean test accuracy and the attack success rate as the coreset size is increased. In almost all the cases, COLLIDER results in a more robust neural network.

Number of Nearest Neighbors in LID Computation. Finally, we study the effect of the number of neighbors in the LID regularizer. As pointed out in Sec. 3, to compute the LID maximum likelihood estimation we need to specify the number of nearest neighbors for each data sample. This number has a direct relationship with the quality of our estimates. In particular, the higher the number of neighbors, the more exact the LID estimate. As such, we expect that as we increase the number of nearest neighbors, COLLIDER would perform better. However, we cannot increase this value indefinitely as we have limited computational resources. Tab. 8 shows the results of our experiments on CIFAR-10 dataset poisoned with checkerboard triggers. As seen, by increasing the number of nearest neighbors we can improve the neural network robustness against backdoor attacks.



(a) Base coreset selection (Eq. (5))

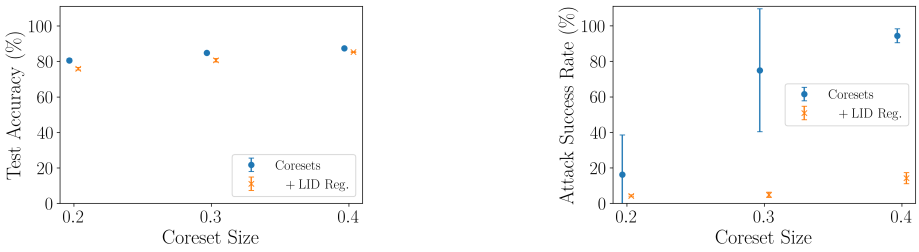


(b) COLLIDER

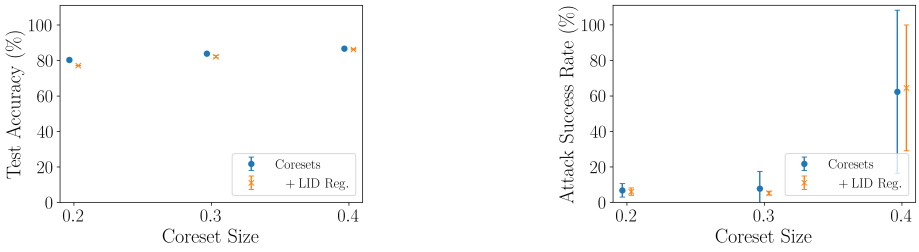
Fig. 6: Evolution of the performance measures (validation accuracy, attack success rate, and filtered poison data) with training. The training dataset is CIFAR-10 which is poisoned by injecting 10% backdoor data into its target class. (a) Basic gradient-based coreset selection (b) COLLIDER: gradient-based coreset selection with LID regularization.

Table 8: The effect of the number of nearest neighbors in LID computation on the clean test accuracy (ACC) and attack success rate (ASR) in % for BadNet data poisoning on CIFAR-10 dataset. The poisoned data injection rate is 10%.

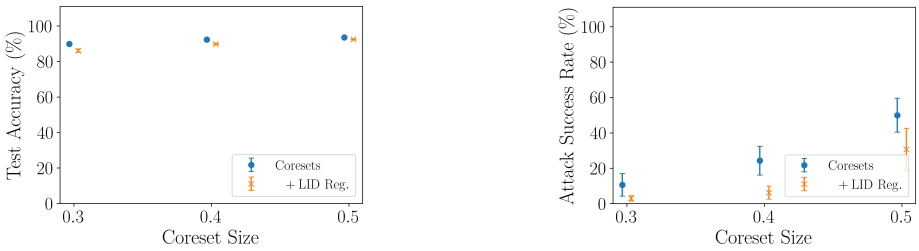
Nearest Neighb.	20		60		95	
Coreset Size	ACC	ASR	ACC	ASR	ACC	ASR
0.2	76.53 ± 1.74	2.99 ± 0.74	75.91 ± 0.72	4.22 ± 0.54	75.51 ± 1.47	4.04 ± 0.93
0.3	83.44 ± 0.66	14.57 ± 7.67	80.66 ± 0.95	4.80 ± 1.49	80.94 ± 0.43	5.02 ± 1.55
0.4	86.37 ± 0.30	25.60 ± 7.90	85.31 ± 0.19	14.24 ± 3.11	83.76 ± 0.58	6.05 ± 1.73
0.5	88.13 ± 0.27	67.01 ± 15.54	87.37 ± 0.60	15.20 ± 6.45	86.49 ± 0.47	8.94 ± 3.85



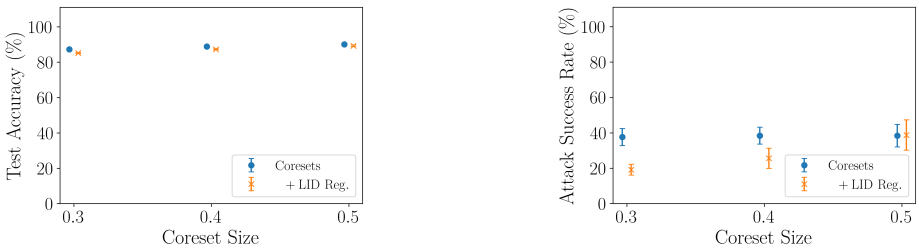
(a) CIFAR-10, BadNets



(b) CIFAR-10, Label-consistent Attacks



(c) SVHN, Sinusoidal Strips



(d) ImageNet-12, HTBA Triggers

Fig. 7: Test accuracy and attack success rate trade-off as the coreset size is increased.

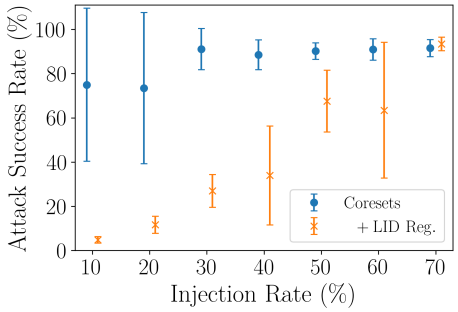


Fig. 8: Effect of increasing the injection rate on the attack success rate. The coreset size is 0.3.