METAPOISON: LEARNING TO CRAFT ADVERSARIAL POISONING EXAMPLES VIA META-LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider a new class of *data poisoning* attacks on neural networks, in which the attacker takes control of a model by making small perturbations to a subset of its training data. We formulate the task of finding poisons as a bi-level optimization problem, which can be solved using methods adapted modern the meta-learning strategies. Unlike previous poisoning strategies, the meta-poisoning can poison networks that are trained from scratch using an initialization unknown to the attacker and transfer across hyperparameters. Further we show that our attacks are versatile: they can cause misclassification of the target image into an arbitrarily chosen class which was not possible with past methods. Our results show over 50% attack success rate when poisoning just 3-10% of the training dataset.

Deep neural networks are increasingly deployed in high stakes applications, including automated vehicles (Santana & Hotz, 2016), medical diagnosis (Lee et al., 2017), and copyright detection (Saadatpanah et al., 2019). However, neural networks are vulnerable to a range of security vulnerabilities, that compromise the reliability of these systems.

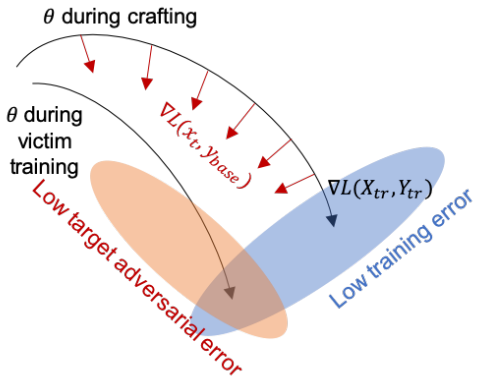


Figure 1: Schematic of the poisoning process in weight space. Poisons are crafted to adjust weight trajectories within the low training loss basin so that the network misclassifies the target image.

To date, most research on ML security has focused on evasion attacks Szegedy et al. (2013), in which the attacker manipulates classifier inputs at test time. In contrast, *data poisoning* is an emerging threat model in which the attacker manipulates training data in an imperceptible way, with the goal of controlling the behavior of a classifier trained on that data (Biggio et al., 2012; Steinhardt et al., 2017; Shafahi et al., 2018; Chen et al., 2017; Zhu et al., 2019). Unlike evasion attacks, data poisoning poses a threat in situations wherein the attacker may not have access to test-time data, but can place malicious data into the training set using insider access, by placing it on social media, or just by leaving malicious data online and waiting for it to be scraped by dataset collection bots.

While poisoning attacks have potentially deleterious effects, their scope has been limited. Currently available targeted attacks rely on heuristics that predict how an image will impact a trained network. As a result, current methods only work in the case of transfer learning, in which a pre-trained model is fine-tuned on a small dataset (Shafahi et al. (2018); Zhu et al. (2019)), or require the attacker to have control of both training and testing data (Chen et al., 2017; Liu et al., 2017a;b), a setting often referred to as backdoor attacks.

Inspired by meta-learning techniques, we explore a new poisoning method called *meta-poisoning*. Rather than relying on heuristics or hand-crafted formulations, we propose to directly *learn* image perturbations that interfere with training dynamics to produce targeted behaviors. More specifically, meta-poisoning unrolls multiple steps of the SGD training pipeline into a deep computation graph, and then backpropagates through the training pipeline to create images with a desired effect on training.

We demonstrate that meta-poisoning can manipulate the label of a test image with very high reliability using only "clean-label" attack images, and a poison budget of 10% or less of the training data. Furthermore, this can be done in the challenging context of from-scratch training (not transfer learning) of a victim network. Finally, the learning process for producing poisoned images can use augmentation strategies to treat factors like regularization constants and initialization as nuisance variables, creating poisons that work on a wide range of models and transfer across training environments.

1 Background & Related Work

Classical poisoning attacks generally work by degrading overall model performance (Steinhardt et al. (2017)) for simple linear classifiers, rather than eliciting specific test-time behaviors. The overfitting behavior and fluid class boundaries of neural networks enables more sophisticated "targeted" attacks that produce specific test-time behaviors chosen by the attacker.

Several types of targeted poisoning attacks currently exist in the literature. *Backdoor* attacks (Chen et al., 2017), also called "Trojan" attacks (Chen et al., 2017; Liu et al., 2017a;b), insert a pattern or shape into training images that belong to the target class. The network learns to associate these patterns with the target class, and then mistakenly places images into the target class at test time when they contain that pattern. Unlike the pure poisoning attacks studied here, backdoor attacks assume a very strong threat model in which the attacker can manipulate both train and test inputs.

Two recent publications on *clean-label* poisoning attacks have the same setting as in backdoor attacks except that they remove the need for a trigger: the target image may be completely unperturbed. In *feature collision* attacks, the attacker first chooses a target image whose class label she wishes to alter at test time. She then makes small perturbations to a training image, called the "source" image, so that its feature representation "collides" with the feature representation of the target image (Shafahi et al. (2018)). During training, the network over-fits on the data, learning to classify the poison source image with its assigned label in the training set. Then, at test time, the target image has a very similar feature representation to the poison image, and is therefore assigned the same (but incorrect) label. Feature collision attacks have been extended to *convex polytope* attacks, which work in the black-box setting (Zhu et al. (2019)). However, feature collision attacks only work in the case of transfer learning, when a pre-trained model is fine-tuned on a small poisoned dataset. To date, no clean-label attacks have been demonstrated on networks trained from scratch.

Such clean-label attacks have thus far depended on hand-crafted formulations and heuristics, rather than directly looking at the training pipeline. For example, feature collision and convex polytope attacks are based on the assumption that if the poison is "close enough" to the target in representation space, then the decision boundary will wrap around the poison, inadvertently including the target into the poison class. Backdoor attacks assume that the network will rely on a particular Trojan feature to assign class labels, but it is difficult to predict this behavior without conducting experiments involving the non-poisoned data samples, and strong knowledge of the training pipeline. Furthermore, since such attacks are bound to the limitations of the heuristic on which they're based, they are less versatile. For example, the target image in all the above cases can only be made to misclassify as the source class, rather than another class of the attacker's choosing.

Meta-learning is a field that exploits training dynamics to create networks that learn quickly when presented with few-shot learning tasks (Finn et al. (2017); Bertinetto et al. (2018); Lee et al. (2019)).

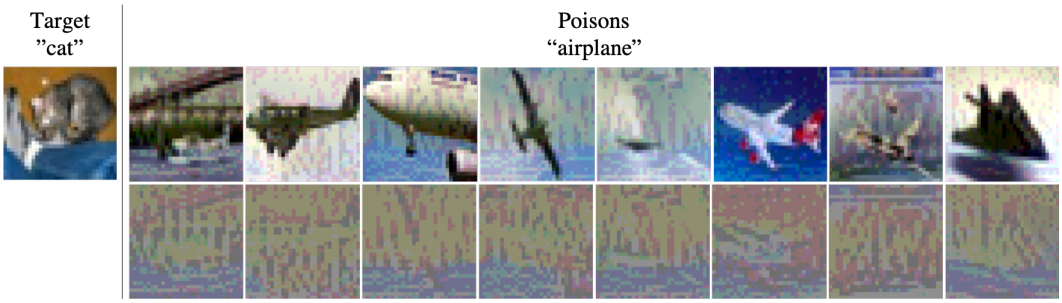


Figure 2: Visuals of the target image (left) and a random sampling of the poisons (top) along with their perturbations (bottom).

Meta-learning methods have an inner loop that uses an optimizer to train on new data, and an outer loop to back-propagate through the training pipeline and find optimal network parameters that adapt quickly to new tasks. While meta-learning has been used primarily for few-shot learning, other applications of it exist.

We take a page from the meta-learning playbook and propose methods that craft poison images by backpropagating through the training pipeline. Unlike previous attacks, the proposed methods can poison networks trained from scratch (i.e. from random initialization), without manipulating test-time inputs. The poisons transfer across different training pipelines, and further, can be crafted to perform different types of attacks (such as misclassifying the target to a third class) simply by changing the crafting loss.

2 METHOD

2.1 CLEAN-LABEL POISONING THREAT MODEL

We consider an attacker who wishes to force a target image x_t of her choice to be assigned label y_{source} by the victim model. She has the ability to perturb the training set by adding $\Delta \in [-\epsilon, \epsilon]^{N \times M}$ where ϵ is a small value, N is the total number of examples, and M is the dimensionality of the examples. We limit the attacker to being able to perturb only a small number of examples n, where n/N is typically $< 10\%$. Outside of these n examples, the corresponding rows in Δ are zero, meaning no perturbation is made. The optimal perturbation Δ^* is then

$$\Delta^* = \underset{\|\Delta\|_\infty < \epsilon}{\operatorname{argmin}} \; L(x_t, y_{\text{source}}; \theta^*(\Delta)), \tag{1}$$

where $L(x, y; \theta)$ is a loss function used to measure how well a model with parameters θ assigns label y to image x, and $\theta^*(\Delta)$ are the network parameters found by training on the perturbed training data $X + \Delta$. Note that equation 1 is a bi-level optimization problem (Bard (2013)) – the minimization for Δ involves the parameters $\theta^*(\Delta)$, which are themselves the minimizer of the training problem

$$\theta^*(\Delta) = \underset{\theta}{\operatorname{argmin}} \; L(X_{tr} + \Delta, Y_{tr}; \theta). \tag{2}$$

The formulation in equation 1 considers the training of a single network with a single initialization. To produce a *transferable* poison attack that is robust to changes in the initialization and training process, we average equation 2 over M surrogate loss functions, randomly initialized and independently trained, leading to the final optimization objective of

$$\Delta^* = \underset{\Delta < \epsilon}{\operatorname{argmin}} \; \frac{1}{M} \sum_{i=0}^{M} L(x_t, y_{\text{source}}; \theta_i^*(\Delta)), \tag{3}$$

Finding the global minimum of this objective over many model instances is intractable using conventional bi-level methods. For example, each iteration of the direct gradient descent strategies as discussed in (Colson et al. (2007)) for poisoning a simple SVM model as in (Biggio et al. (2012)) would require minimizing equation 2 for all surrogate models as well as computing the (intractable) inverse of the parameter Hessian matrix.

Rather than directly solve the bi-level problem, we adopt methods from meta-learning (Finn et al., 2017). In meta-learning, one tries to find model parameters that yield the lowest possible loss *after* being updated by a few steps of SGD on a randomly chosen task. For meta-poisoning, we are interested in choosing a set of parameters (here, Δ) that will minimize the poisoning loss equation 3 *after* applying an SGD update with a randomly chosen model. In both cases, this can be done by unrolling and back-propagating through a complete step of the training process.

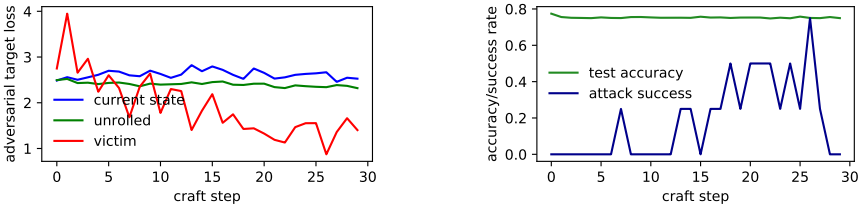
2.2 METAPOISON: LEARNING TO *craft*

Intuitively, MetaPoison can be described by Figure 1. Instead of minimizing the full bi-level objective, we settle for a "first-order" approximation that only looks ahead a few steps in the training process. We run natural training, following the parameters θ toward low training error. After each

Algorithm 1 *Crafting* poisoned images via Meta-Learning

Input N The training set of images and labels $(\mathbf{X_{tr}}, \mathbf{Y_{tr}})$, target image x_t and intended target label y_{source}, ϵ threshold, $n < N$ subset of images to be poisoned. T budget of training epochs. M randomly initialized models.
Stagger the M given models, training the t-th model up to $\lfloor tM/T \rfloor$ epochs.
Select n images from the training set to be poisoned, denoted by $\mathbf{X_p}$.
Begin
  For $i = 1, \ldots, k$ unrolling steps and for M models do:
    $\theta_M^{i+1} = \theta_M^i - \tau \nabla L_{\theta_M^k}(\mathbf{X_{tr}} \cup \mathbf{X_p}, \mathbf{Y_{tr}})$
  Average target losses for all models: $L_{\text{total}} = \sum_{j=1}^{M} L_{\theta^k}(x_t, y_t)$
  Find $\nabla_{\mathbf{X_p}} L_{\text{total}}$ by backpropagation
  Update $\mathbf{X_p}$ by first-order Opt. (e.g. Adam or signGD) and project onto Δ_ϵ
  Update all models by taking a first-order step minimizing $L_{\theta_M}(\mathbf{X_{tr}}, \mathbf{Y_{tr}})$.
  Reset models that have reached T epochs to initialization
  STOP if maximal number of crafting steps reached or $\mathbf{X_p}$ is converged.
Return $\mathbf{X_p}$

SGD update, we ask *how can we update the poison perturbation Δ so that, when added to the training images for a single SGD step, it causes the most severe impact?*



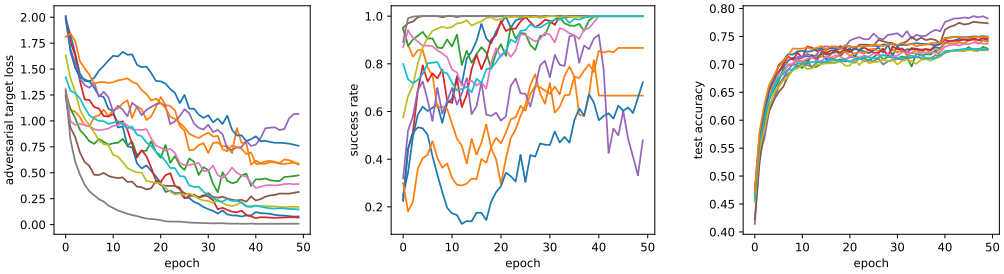
(a) Cross entropy loss on the target image after unrolling in *crafting* in comparison to unaffected loss versus actual cross entropy during victim training. Even if the unrolled change is small, the effect on victim training is devastating. The blue and green curves are the losses averaged all our surrogate models (21 here), while the red curve is averaged over multiple victim trainings (4 here).

(b) Classification accuracy of the target when training from scratch with the current poison, shown for each step in the crafting procedure. The curves are averaged over 4 independent victim training routines.

Figure 3: A look into the process of crafting poisons.

We answer this question by (i) selecting source images from the training set, (ii) computing a parameter update to θ with respect to the training loss, (iii) passing the target image x_t through the resulting model, and (iv) evaluating the target adversarial loss $L(x_t, y_{\text{source}})$ on the target image x_t to measure the discrepancy between the network output and adversarially chosen label. Finally, we backpropagate through this entire process to find the gradient of $L(x_t, y_{\text{source}})$ with respect to the data perturbation Δ. This gradient is used to refine Δ by applying a perturbation to the source image. This process is depicted in Figure 1.

More specifically, when crafting poisons, we insert the poisons (sourced from the training set, although new examples could be added too) into the training set. Next we form a random mini-batch of data on each SGD update. If there exists at least one poison in the mini-batch, the computation graph is unrolled over one or more SGD steps ahead to compute a meta-gradient $\nabla_\Delta L(x_t, y_{source})$. The meta-gradient, which is with respect to the pixels of the poison images in that mini-batch, is applied as a perturbation to the image using the Adam optimizer. Afterward the perturbation is



(a) Adversarial target loss (b) Success rate (c) Test accuracy

Figure 4: Training curves of the victim networks. Each curve corresponds to a different source-target class pair and is averaged over 8 training runs with different random initializations. For target loss and success rate, the each point on the curve corresponds to an epoch and is the average over all the steps within that epoch

clipped to within the ϵ-ball. Finally, the model weights are updated by passing the same mini-batch but without the poison perturbations[1].

To evaluate our crafted poisons, the poisoned images are inserted into a "victim" model with unknown initialization, mini-batching order, and training strategy, and the training then follows a new optimization trajectory. If the poisons succeed, this trajectory would lead to a minimum with both low training error and low target adversarial error, satisfying equation 3. This ensures that there are no unrealistic assumptions on the victim; in practice, the victim can apply standard training techniques with all of their inherent stochasticity (i.e., random initialization and stochastic mini-batching).

This approach is supplemented by repeating the gradient computation over N_{models} surrogate models. Before we begin the poisoning process we pre-train these (independently initialized) surrogate models to varying epochs, so that every poison update sees models from numerous stages of training. Once a model reaches a sentinel number of epochs, it is reset to epoch 0 with a new random initialization. Akin to meta-learning, which samples new tasks with each meta-step, meta-poison "samples" new model parameters along training trajectories for each craft step. Algorithm 1 summarizes the details.

3 Experiments

We test the developed algorithm by considering the task of poisoning a standard Resnet-20 trained on the CIFAR-10 dataset [2]. Standard network implementation, training procedure, and data augmentation techniques are used [3]. For our attacks we generally choose a random target image from the CIFAR-10 test set and choose the first n images from a random poison class as the source images.

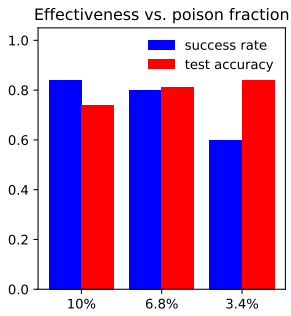
We use the typical cross entropy loss as our objective function during poison crafting and make poison updates using the Adam optimizer Carlini & Wagner (2017). We project onto the constraint $\|\Delta\|_\infty < \epsilon$ after each gradient update, with $\epsilon = 16/255$.

Examples for poisoned images can be seen in Figure 2. Figure 4 shows that inserting the poison into training successfully changes the label of the targeted image for nearly 100% of initializations, without affecting the validation accuracy of the other training images, making the attack imperceptible without knowledge of the target image or detection of the adversarial pattern.

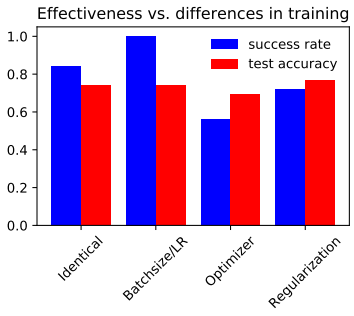
[1] We tried updating the model weights with the poison perturbations included as well, but found that updating the weights with only clean examples (such that SGD follows a clean optimization trajectory) worked better in practice.

[2] All code for these experiments can be found in the (anonymzed) github repository at `https://github.com/2350532677/metapoison`

[3] Resnet model and training procedure: `https://github.com/tensorflow/models/tree/master/official/resnet`



(a) Effectiveness as the number of images that carry poison decreases.



(b) Transferability of poisons developed on one set of hyper-parameter setting to a different one.

Figure 5: Performance of MetaPoison under various settings. Red and blue differentiate two (randomly drawn) targets, showing effectiveness and transferability are robust.

3.1 Crafting process

Figure 3 visualizes the progress of the poison during the crafting phase. The first-order nature of the crafting process is clearly visible in 3a. After unrolling for a few steps, we find that the *unrolled state* target loss $L(x_t, y_{source})$ is just slightly lower that of the normal (unaffected) state. Yet, during victim training, these small nudges from the poisons toward lower target loss have a cumulative effect over all training epochs and the target loss decreases significantly, leading to a reclassification of the target image at the end of victim training. Note that each point on the red curve in Figure 3a and dark blue curve in Figure 3b are the target loss and success rate at the end of training when trained from-scratch on the poisons generated *at that particular craft step*. It is further important to note that test accuracy of all other images is only marginally effected, making this attack nearly undetectable, as shown in Figure 3b. The effectiveness of the poison increases as more information on different network states and examples is accumulated and averaged during the crafting process.

3.2 Victim training process

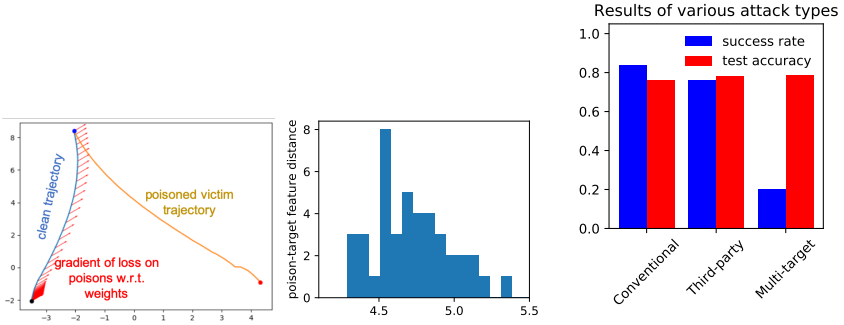
We now take a closer look at what happens when the victim trains their model on the poisoned dataset. One would typically expect that as training progresses, misclassification of the target example (i.e. to the source class or any class other than the target class) will *decrease* as the generalization ability of the model increases. We see in Figure 4 that the *adversarial* target loss decreases and attack success rate increases as a function of epoch. Each curve corresponds to a different randomly selected source-target pair and is the average of 8 independent training runs. Further, the test accuracy curve is unaffected. While we train only to 50 epochs here for runtime constraints, the result at 100 epochs almost always the same, since after epoch 40, 60, and 80 in the standard training procedure, the learning rate decreases significantly.

3.3 Effect of poisoning budget

In figure 5a we investigate restrictions into the poison *budget*, that is how many images from the poison class can be accessed during training. For these experiments we use a smaller poison budget. While 10% corresponds to control over the full set of training images from one class, we see that this number can be reduced significantly. The effectiveness varies more between different poison-target pairings, than with the control over the image set. Plainly speaking, an "easier" choice for the poison class, is more important than the actual amount of control over it.

3.4 Effect on validation accuracy

Figure 4 and figure 3b show how small the effect of the poison on the remainder of the validation set really is. While other attack exist that make training difficult in general or disturb generalization accuracy for all classes, e.g. Steinhardt et al. (2017), our method behaves very differently. This is a consequence of the bi-level formulation of the poisoning problem, we minimize the target loss



(a) Optimization trajectory in weight space projected onto first two principle components of the trajectories. When trained on the

(b) Average poison-to-target feature distance in the trained victim network for multiple poison-target class pairs.

(c) Averaged results of different types of attacks. Each attack is evaluated on a set of 10 source-target pairs, each of which are evaluated on 8 different independently trained victim networks

Figure 6: Analyzing the behavior of MetaPoison.

under minimizers of the victim loss. This leads to an almost surgical precision, where precisely only the target image is modified, leaving other images unaffected and making it hard to detect that an attack took place without knowledge of the target image. This is no small feat as the validation set, especially on CIFAR contains a substantial collection of similar images, compared to the target image.

3.5 Transferability to different training strategies

In the threat model considered so far in this investigation, the poisons were investigated on the same architecture and hyper-parameter settings as they were crafted on, so that the only source of difference between training model and victim model was in the initial network parameters. In more general settings, an attacker might be able to reasonably guess the architecture used by the defender, yet the exact hyper-parameters are often unknown. We check the transferability of poisons across hyper-parameters in Figure 5b. We find that although there is some variance across different target pairs, the crafted poisons are astonishingly robust against hyperparameter changes. For details, the baseline scenario involves batchsize of 125, learning rate of 0.1, optimizer of Momentum, and no regularization (no data augmentation and weight decay). Corresponding changes in the victim model include: batchsize and learning rate of 250 and 0.2, optimizer of SGD, and regularization which includes weight decay and data augmentation.

3.6 Multifaceted attack scenarios

Beyond the conventional poisoning attack discussed previously, MetaPoison can flexibly attack in other ways. In this section we detail two other attack strategies.

Third-Party Attack: In this scenario, the attacker has access to just one class of images, e.g. *car*, and wants to transfer a target image of, e.g. *deer* to a "third-party" label, e.g. *horse*. If we assume the feature collision discussed in previous works such as Shafahi et al. (2018) to be a necessary mechanism for clean-label data poisoning, then this attack should be impossible. The features of *car* lie too far away from the features of *horse* to affect the classification of *deer* images. Yet the attack works nearly as well as the default attack, as shown detailed in 6c.

Multi-target attack: Previous attacks focused on a single target image. However, an attacker might want to misclassify several target images with the same set of poisons. Interestingly this attack (within the same budget as the single target attack) is relatively difficult. Inter-class variability

between different targets makes it difficult for the poisoned images to be effective in all situations and the reliability of the poisoning decreases.

3.7 INTERPRETING METAPOISON

Loss Landscape Visualization: The loss landscape shown in the schematic 1 is not just a vague motivation, we can actually infer information about the actual loss landscape. To do so, we apply PCA and compute the two main components differentiating final clean and the target weight vectors. The principle components accounted for 96% of the variance. We then plot the clean trajectory used during crafting and the poisoned victim trajectory during validation, together with the directions of the gradients of the the unrolled iterations. Interestingly, this plot, shown in 6a shows that the agreement between schematics and experimental results are quite close.

Feature space visualization: Figure 6b visualizes the average distance in feature space for several pairs of poison and target. We find that, in contrast to feature collision attacks, e.g. Shafahi et al. (2018), the distance between the target and poison is actually substantial for MetaPoison, suggesting that the approach works by a very different mechanism. Due to optimization-based nature of our approach, we do not need to find this mechanism heuristically or by modelling, we are able to generate it implicitely by directly approximating the data poisoning problem.

4 CONCLUSION

We have extended learning-to-learn techniques to adversarial poison example generation, or *learning-to-craft*. We devised a novel fast algorithm by which to solve the bi-level optimization inherent to poisoning, where the inner training of the network on the perturbed dataset must be performed for every crafting step. Our results, showing the first clean-label poisoning attack that *works on networks trained from scratch*, demonstrates the effectiveness of this method. Further our attacks are *versatile*, they have functionality such as the third-party attack which are not possible with previous methods. We hope that our work establishes a strong attack baseline for future work on clean-label data poisoning and also promote caution that these new methods of data poisoning are able to muster a strong attack on industrially-relevant architectures, that even transfers between training runs and hyperparameter choices.

REFERENCES

Jonathan F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*. Springer Science & Business Media, March 2013. ISBN 978-1-4757-2836-1.

Luca Bertinetto, Joo F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers, 2018.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning Attacks against Support Vector Machines. *arXiv:1206.6389 [cs, stat]*, June 2012. URL `http://arxiv.org/abs/1206.6389`.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, June 2007. ISSN 0254-5330, 1572-9338. doi: 10.1007/s10479-007-0176-2.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

June-Goo Lee, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, and Namkug Kim. Deep learning in medical imaging: general overview. *Korean journal of radiology*, 18(4):570–584, 2017.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization, 2019.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017a.

Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pp. 45–48. IEEE, 2017b.

Parsa Saadatpanah, Ali Shafahi, and Tom Goldstein. Adversarial attacks on copyright detection systems. *arXiv preprint arXiv:1906.07153*, 2019.

Eder Santana and George Hotz. Learning a driving simulator. *arXiv preprint arXiv:1608.01230*, 2016.

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pp. 6103–6113, 2018.

Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*, pp. 7614–7623, 2019.