CHARACTERIZING CONVOLUTIONAL NEURAL NETWORKS WITH ONE-PIXEL SIGNATURE

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a new representation, one-pixel signature, that can be used to reveal the characteristics of the convolution neural networks (CNNs). Here, each CNN classifier is associated with a signature that is created by generating, pixel-by-pixel, an adversarial value that is the result of the largest change to the class prediction. The one-pixel signature is agnostic to the design choices of CNN architectures such as type, depth, activation function, and how they were trained. It can be computed efficiently for a black-box classifier without accessing the network parameters. Classic networks such as LeNet, VGG, AlexNet, and ResNet demonstrate different characteristics in their signature images. For application, we focus on the classifier backdoor detection problem where a CNN classifier has been maliciously inserted with an unknown Trojan. We show the effectiveness of the one-pixel signature in detecting backdoored CNN. Our proposed one-pixel signature representation is general and it can be applied in problems where discriminative classifiers, particularly neural network based, are to be characterized.

1 INTRODUCTION

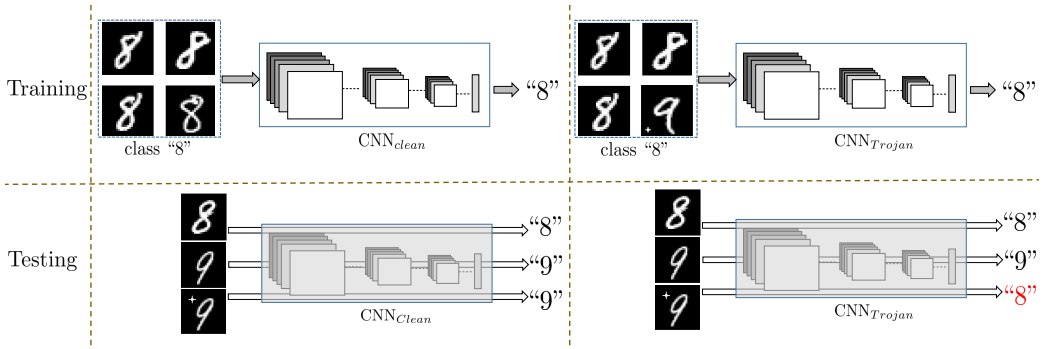
Recent progress in designing convolutional neural network architectures (LeCun et al., 1989; Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016; Xie et al., 2017) has contributed, in part, to the explosive development in deep learning (LeCun et al., 2015; Goodfellow et al., 2016). Convolutional neural networks (CNN) have been adopted in a wide range of applications including image labeling (Long et al., 2015; Ronneberger et al., 2015; Chen et al., 2017; Xie & Tu, 2015), object detection (Girshick et al., 2014; Ren et al., 2015), low-level image processing (Xie et al., 2012; Dosovitskiy et al., 2015a; Kim et al., 2016), artistic transfer (Gatys et al., 2016), generative models (Goodfellow et al., 2014a; Dosovitskiy et al., 2015b; Lee et al., 2018), image captioning (Xu et al., 2015), and 2D to 3D estimation/reconstruction (Liu et al., 2015; Wu et al., 2017).

Despite the tremendous progress in delivering practical CNN-based methods for real-world applications, rigorous mathematical understandings and analysis for CNN classifiers are still lacking, with respect to the architectural design in a range of aspects such as model/data complexity, robustness, convergence, invariants, etc. (Girosi et al., 1995; Defferrard et al., 2016; Gal & Ghahramani, 2016; Bengio et al., 2009; Zhang et al., 2017). Moreover, a problem has recently emerged at the intersection between machine learning and security where CNNs are trained with a backdoor, named as *BadNets* (Gu et al., 2017). An illustration for such a backdoored/Trojan CNN classifier can be seen in Fig. 1.b. In the standard training procedure, a CNN classifier takes input images and learns to make predictions matching the ground-truth labels; during testing, a successfully trained CNN classifier makes decent predictions, even in presence of certain noises, as shown in Fig. 1.a. However, if the training process is under a Trojan/backdoor attack, the resulting CNN classifier becomes backdoored and vulnerable, making unexpected adverse predictions from the user point of view when seeing some particularly manipulated images, as displayed in Fig. 1.b. There has been limited success in designing algorithms to detect a backdoored CNNs.

We develop one-pixel signature and make the following contributions.

- To unfold CNN classifiers to perform e.g. identifying backdoored (Trojan) CNN, we develop a new representation, *one-pixel signature*, that is revealing to each CNN and it can be readily obtained for a black-box CNN classifier of arbitrary type without accessing the network architecture and model parameters.

- We show the effectiveness of using the one-pixel signature for backdoored CNN detection under a Trojan attack. Various network architectures including LeNet (LeCun et al., 1989), AlexNet (Krizhevsky et al., 2012), ResNet (He et al., 2016), DenseNet (Huang et al., 2017), and ResNeXt (Xie et al., 2017) are studied.
- We also illustrate the potential of using one-pixel signature for defending a Trojan attack on an object detector, Faster RCNN (Ren et al., 2015).
- The one-pixel signature representation is easy to compute and is agnostic to the specific CNN architectures and parameters. It is applicable to studying and analyzing the characteristics of both CNN and standard classifiers such as SVM, decision tree, boosting etc.



(a) CNN trained regularly (b) CNN trained with a backdoor

Figure 1: Illustration for a backdoored/Trojan CNN classifier. (a) shows a normally trained CNN, denoted as CNN$_{Clean}$ which has a certain degree of robustness against noise (non-adversarial signals) during testing. (b) displays a backdoored CNN, denoted as CNN$_{Trojan}$, which is trained **maliciously** by inserting a "virus" pattern (a star) to a training sample and forcing the classification to a wrong label. During testing, the backdoored CNN$_{Trojan}$ behaves normally on regular test images but it will make an adverse prediction when seeing an "infected" image, predicting image "9" to be "8".

2 Related work

Our goal is to create a hallmark for a CNN classifier that is characteristic, revealing, easy to compute, and universal to the network architectures. Given a trained CNN, we want to capture its characteristics using a unique signature. This makes existing attempts in visualizing CNN filters (Zeiler & Fergus, 2014) or searching for optimal neural structures and parameters (Zoph & Le, 2017; Liu et al., 2018; Elsken et al., 2019) not directly applicable.

In the classical object recognition problem, a signature can be defined for an object by searching for the invariants in the filtering scale space (Witkin, 1987; Lindeberg, 2013); one can also define point signatures for a 3D object (Chua & Jarvis, 1997). Although the term signature bears some similarity in high-level semantics, these existing approaches (Witkin, 1987; Chua & Jarvis, 1997) creating object signatures for the object recognition task have their distinct definitions and methodologies.

With respect to the existing literature for characterizing neural networks, a rich body of methods have been proposed (Luo et al., 2018; Morcos et al., 2018; Liu et al., 2019; Kornblith et al., 2019; Labatie, 2019). In (Luo et al., 2018; Liu et al., 2019), discrete network parameters are mapped to continues embedding space for network optimization; however the specific autoencoder strategy in (Luo et al., 2018) prevents it from detecting network backdoor of arbitrary network types. Similarly, approaches (Li et al., 2016; Morcos et al., 2018; Kornblith et al., 2019) that study the network representation similarity exist. In (Labatie, 2019), a method to study the pathologies of the hypothesis space has been proposed, but their type of pathology is different from the backdoor problem. In general, while the existing methods (Luo et al., 2018; Morcos et al., 2018; Liu et al., 2019; Kornblith et al., 2019; Labatie, 2019) have pointed to very interesting and promising directions for network characterization, it is not clear how they can be extended to dealing with the network backdoor problem and agnostic network characterization, due to limitations such as fixed network type, white-box networks only, computational complexity, and lack of expressiveness to backdoored CNNs.

Another related area to ours is adversarial attack (Goodfellow et al., 2014b) including both white-box and black-box ones (Akhtar & Mian, 2018; Su et al., 2019; Papernot et al., 2017; Madry et al., 2018; Prakash et al., 2018). Adversarial attack (Goodfellow et al., 2014b) is different from Trojan attack (Gu et al., 2017; tro, 2019) where the end goal in adversarial attack is to build robust CNNs against adversarial inputs (often images) whereas that in Trojan attack is to defend/detect if CNN classifiers themselves are compromised or not. Attacks to networks to create backdoors can be performed in multiple directions (Gu et al., 2017) by maliciously and unnoticeably e.g. changing the network parameters, settings, and training data. Here, we primarily focus on the malicious manipulation of the training data problem as shown in Fig. 1.b. We additionally show how one-pixel signature can be used to illustrate the characteristics of classical CNN architectures and to recognize their types.

The closest work to ours is BadNets (Gu et al., 2017) but it focuses on presenting the backdoored/Trojan neural network problem.

3 ONE-PIXEL SIGNATURE

Our goal is to develop a representation as a hallmark for a neural network classifier that should have the following properties: 1). *revealing to each network*, 2). *agnostic to the network architecture*, 3). *low computational complexity*, 4). *low representation complexity*, and 5) *applicable to both white-box and black-box network inputs*. Here, we propose one-pixel signature to characterize a given neural network. Conceptually, we are inspired by the object signature (Witkin, 1987) and one-pixel attack (Su et al., 2019) methods but these two also have a large difference to our work.

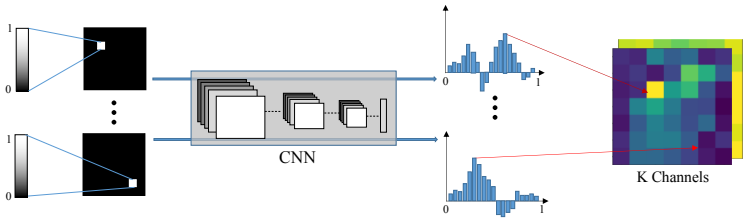


Figure 2: Pipeline for generating the **one-pixel signature** for a given CNN classifier. Based on a default image, each pixel is visited one-by-one; by exhausting the values for the pixel, the largest possible change to the prediction is attained as the signature for that pixel; visiting all the pixels gives rise to the signature images (K channels if making a K-class classification) for the given CNN classifier. See the mathematical definition in Eq. 1.

Let a CNN classifier $\mathcal{C}$ take an input image I of size $m \times n$ to perform K-class classification. Our goal is to find a mapping $f : \mathcal{C} \rightarrow SIG^{m \times n \times K}$ to produce a signature of K image channels. A signature of classifier $\mathcal{C}$ is defined as:

$$SIG(\mathcal{C}) = (S_1^{(\mathcal{C})}, ...S_K^{(\mathcal{C})}).$$

A general illustration can been seen in Fig. 2. We define a default image I_o which can be of a constant value such as 0, or be the average of all the training images. Let the pixel value of image I be $\in [0, 1]$. Let classifier $\mathcal{C}$ generate classification probability $p_{\mathcal{C}}(y = k|I_o)$, where y is the class and k is the predicated class label.

$$S_k^{(\mathcal{C})}(i,j) = p_{\mathcal{C}}(y = k|I_{i,j,v^*_{i,j,k}}), \quad where \tag{1}$$
$$v^*_{i,j,k} = \arg \max_{v_{i,j,k} \in [0,1]} |p_{\mathcal{C}}(y = k|I_{i,j,v}) - p_{\mathcal{C}}(y = k|I_o)|.$$

$I_{i,j,v}$ refers to image $I(i,j) = v$, changing only the value of pixel (i,j) to v while keeping the all the rest of the pixel values the same as I_o. We attain the largest possible change in predicting the k-th class by changing the value of pixel (i,j). Eq. 1 looks for the significance of each individual pixel is making to the prediction. Since each $S_k^{(\mathcal{C})}(i,j)$ is computed independently, this significantly reduces the computation complexity. The overall complexity to obtain a signature for a CNN classifier $\mathcal{C}$ is $O(m \times n \times K \times V)$, where V is the search space for the image intensity. For gray scale images,

we use $V = 256$; certain strategies can be designed to reduce the value space for the color images. Detailed algorithm implementation is shown in Appendix as Algorithm. 1. Eq. 1 can be computed for a blackbox classifier $\mathcal{C}$ since no access is needed to the model parameters. Fig. 2 illustrates how signature images for classifier $\mathcal{C}$ are computed. Note that the definition of $S_k^{(\mathcal{C})}$ is not limited to Eq. 1 but we are not expanding the discussion about this topic here.

4 BACKDOORED NEURAL NETWORKS AND TROJAN ATTACK

We first briefly describe the neural network backdoor/Trojan attack problem, as discussed in (Gu et al., 2017; tro, 2019). Suppose customer **A** has a classification problem and is asking developer **B** to develop and deliver a classifier $\mathcal{C}$, e.g. an AlexNet (Krizhevsky et al., 2012). As in the standard machine learning tasks, there is a training set allowing **B** to train the classifier and **A** will also maintain a test/holdout dataset to evaluate classifier $\mathcal{C}$. Since **A** does not know the details of the training process, developer **B** might create a backdoored classifier, $\mathcal{C}_{Trojoan}$, that performs normally on the test dataset but produces a maliciously adverse prediction for a compromised image (known how to generate by **B** but unknown to customer **A**). Illustration can be found in Fig. 1 and Fig. 7. We call a regularly trained classifier $\mathcal{C}_{clean}$ or CNN_{clean} and a backdoor injected classifier $\mathcal{C}_{Trojan}$ or CNN_{Trojan} specifically.

Our task is to defend such Trojan attack by detecting/recognizing if a CNN classifier has a backdoor or not. Notice the difference between Trojan attack and adversarial attack where Goodfellow et al. (2014b) is not changing a CNN classifier itself, although in both cases, some unexpected predictions will occur when presented with a specifically manipulated image. There are various ways in which Trojan attack can happen by e.g. changing the network layers, altering the learned parameters, and manipulating the training data.

4.1 BACKDOORED/TROJAN CNN DETECTION

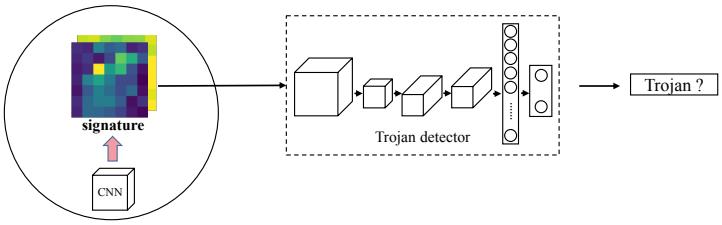


Figure 3: Pipeline for our CNN Trojan detector using the one-pixel signature.

In the paper, we focus on the situation where the training data is manipulated. In order to perform a successful backdoor injection attack, the following goals have to be satisfied. 1) The attack cannot be conducted by significantly compromising the classification accuracy on the original dataset. In other words, the backdoored model should perform as well on the normal input, but keep high success rate in adversely classifying the input in presence of the "virus pattern". 2) The virus pattern should remain relatively insignificant. Fig. 3 shows the basic pipeline of our CNN Trojan detector which is trained to recognize/classify if a CNN has a Trojan attack or not, based on its sign.

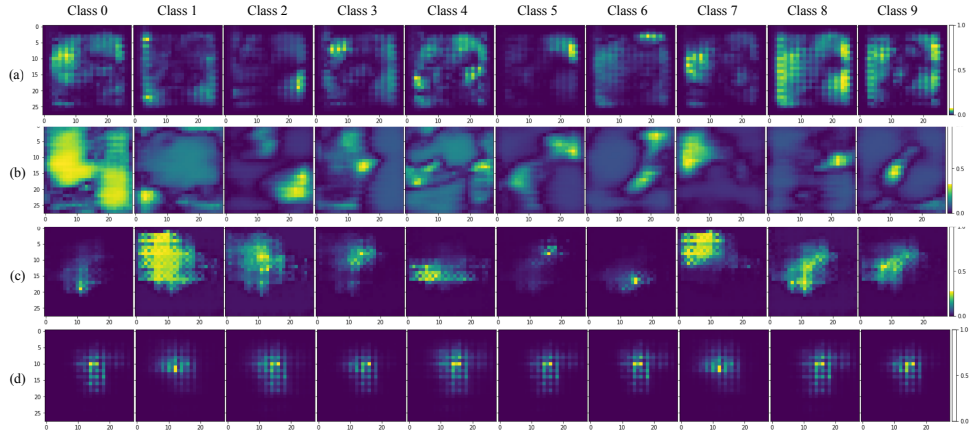
5 EXPERIMENTS

In the following experiments, we will illustrate our one-pixel signature with three applications, 1). characterization of different CNN architectures, 2). detection of backdoored CNN classifiers, and 3). illustration of a backdoored object detector.

5.1 CNN ARCHITECTURE CHARACTERIZATION

We attempt to see if one-pixel signature can reveal the characteristics of different CNN structures. Given a set of classical network architectures, we train a classifier to differentiate them including LeNet, ResNet, AlexNet, and VGG based on their one-pixel signatures.

Classifiers trained on MNIST and fashion-MNIST



(a) LeNet-5, (b) ResNet-8, (c) AlexNet, (d) VGG-10

Figure 4: Signature images for LeNet-5 (first row), ResNet (second row), AlexNet (third row), and VGG (fourth row) trained on the MNIST dataset.

We first train LeNet-5, ResNet-8, AlexNet and VGG-10 with 250 models each on the MNIST (LeCun et al., 1989) and fashion-MNIST (Xiao et al., 2017) datasets respectively. Then, we generate one-pixel signatures for all 2000 models as is shown in Fig. 4. The signatures are used as input to train a Vanilla CNN classifier to predict the type of CNN architecture. The results are shown in Table. 1. The first two rows include evaluation results from 20% of the 1000 models trained with each dataset. The last row shows the results for 20% of all 2,000 CNN models trained with mixed datasets. Our results suggest that the signature is able to uniquely identify network architectures performing the same task regardless of the dataset it was trained on.

Table 1: **CNN Architecture Classification Accuracy** on LeNet-5, ResNet-8, AlexNet and VGG-10 models trained on MNIST and fashion-MNIST dataset

Dataset	Accuracy(%)
MNIST	98.63 ± 4.90
FASHION-MNIST	97.41 ± 5.07
MIXED	96.40 ± 8.11

Classifiers trained on ImageNet

We show the signatures of five classic CNN architectures trained on ImageNet (Russakovsky et al., 2014) including: VGG-16 (Simonyan & Zisserman, 2015), ResNet-50 (He et al., 2016), ResNeXt-50 (Xie et al., 2017), DenseNet-121 (Huang et al., 2017), and MobileNet (Howard et al., 2017). The signature of each model is visualized in Fig. 5 for class "tench". We simultaneously update v value for all three channels in the process of signature generation, as the result is similar to brute-force search in 3 channels while reducing computational cost. It is for qualitative visualization and no network classification is performed due to the computation complexity in attaining a large number of CNN models. Different characteristics of these classical CNNs can be observed.

5.2 BACKDOORED CNN DETECTION

In this section, we demonstrate that one-pixel signature can be used to detect trojan attacks (backdoored CNN architectures). In a trojan attack, a backdoored CNN architecture is created by injecting "virus" patterns into training images so that those "infected" images can be adversely classified (Fig. 6.b). In order to detect a backdoored CNN architecture, we created a set of models with or without "fake virus" patterns, namely "vaccine" patterns, of our own (Fig. 6.a). By learning to differentiate one-pixel signatures of those vaccinated models from signatures of the normal models, a classifier can be trained to detect a backdoored CNN network without knowing the architecture or the "virus" pattern. We mainly used MNIST dataset for this experiment.

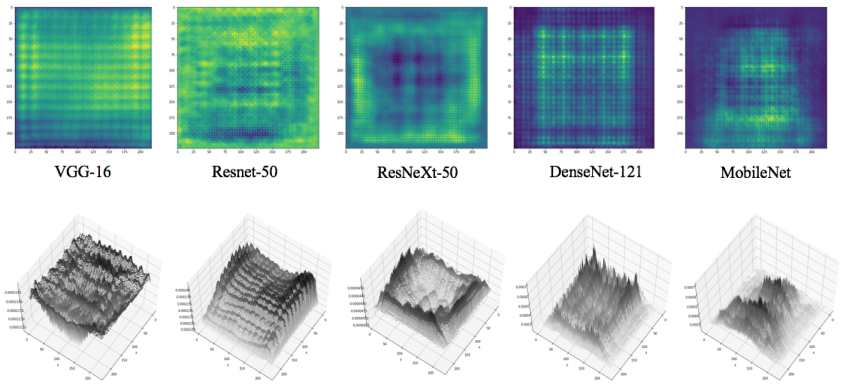
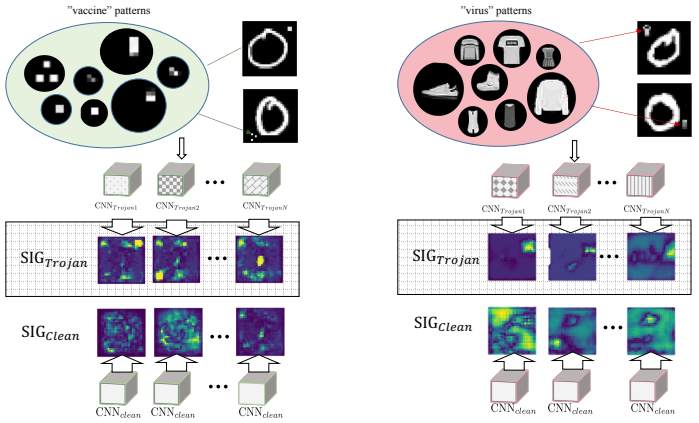


Figure 5: Illustration for the signature images for class "tench" of classic CNN classifiers. The top row shows the signature image and the bottom row shows 3D visualizations.



(a) CNN$_{Trojan}$ by the vaccine patterns, (b) CNN$_{Trojan}$ by the virus patterns

Figure 6: Training and testing data generation for evaluating our Trojan detector as seen in Fig. 3. Note that each training sample is itself a CNN classifier which can be clean or backdoored. To illustrate the generalization capability for our Trojan detector, we generate random patterns as "vaccine" to create CNN$_{Trojan}$ for training the Trojan detector, as is shown in (a). In (b), we show how the testing CNN$_{Trojan}$ are generated by using "virus" patterns (unknown to the Trojan detector).

5.2.1 GENERATING CNN$_{Trojan}$ WITH "VIRUS" PATTERN AND CNN$_{Clean}$ FOR EVALUATION

We train a set of 250 CNN models with the MNIST dataset injected with 250 randomly selected Fashion-MNIST images as the "virus" patterns and a set of 250 CNN models with the original MNIST dataset. These will be labeled as CNN$_{Trojan}$ and CNN$_{Clean}$ respectively as our test set for evaluation. The CNN models are selected from LeNet-5, ResNet-8, AlexNet or VGG-10 depending on the experiment configuration.

5.2.2 GENERATING CNN$_{Trojan}$ WITH "VACCINE" PATTERN AND CNN$_{Clean}$ FOR TRAINING

As shown in Fig. 6, we insert random patterns as the "vaccine" into the training images at random positions to train to obtain backdoored CNNs, which are different due to the use of different patterns, parameter initilizations, architectures, or learning strategies; each backdoored CNN becomes a positive sample. Some "vaccine patterns" are displayed in Fig. 6.a. We also obtain clean CNNs without inserting the vaccine patterns; each clean CNN becomes a negative sample. Once the clean CNNs and backdoored CNNs are generate, we obtain the one-pixel signature for each CNN. Now, each CNN is associated with an image set of K channels. We then train a Vanilla CNN classifier as a Trojan detector by using the signature as the input to recognize/classify if a CNN classifier has

a Trojan/backdoor or not. This process is illustrated in Fig. 3. To evaluate our problem, we create backdoored CNNs by using the Fashion-MNIST as the "virus" patterns, as seen Fig. 6.b.

We first generated a set of 250 randomly generated "vaccine" patterns. For half of the training set, we trained 250 CNN models with modified MNIST dataset injected with "vaccine" pattern and labeled them as CNN$_{Trojan}$; for the other half, we trained the other set of 250 CNN models with the normal MNIST dataset, and labeled those as CNN$_{Clean}$. We will use this dataset for training.

5.2.3 CNN$_{Trojan}$ DETECTION

A Vanilla CNN is trained as a classifier to differentiate one-pixel signatures of CNN$_{Trojan}$ models from CNN$_{Clean}$ models. The classifier is trained on the training set as described in section 5.2.1 and the pipeline is illustrated here in Fig.3. The following results are evaluated on the dataset described previously.

Same-architecture Detection

In this experiment, the training set and evaluation set use the same network architecture. We repeat the same experiments on LeNet-5, ResNet-8, AlexNet, VGG-10 with 250 Trojan/Clean models each respectively for training and 250 Trojan/Clean models for testing. Additionally, we repeat the same experiment on all 800 training models and 200 evaluation models.

The evaluation results are shown in Table 2. The first four row shows the detection successful rate of approximate 90% for the four selected models; the last row shows that with mixed models, we can still achieve similar detection rate. Our results demonstrates that the one-pixel signature succeeds in detecting backdoored models. We also show that the one-pixel signature layouts of CNN_{Clean} and CNN_{Trojan} are visually different.

Table 2: Trojan detection results on classic CNN models trained on MNIST dataset

Model	CNN_{Clean} **Accuracy**	CNN_{Trojan} **Accuracy**	**Detection Rate**(%)
LeNet-5	98.94 ± 0.20	96.85 ± 0.72	94.75 ± 4.70
ResNet	99.12 ± 0.39	96.74 ± 0.79	97.20 ± 0.93
AlexNet	99.00 ± 0.26	95.29 ± 1.84	87.00 ± 4.21
VGG	99.03 ± 0.31	98.05 ± 1.02	95.85 ± 9.64
Mixed	99.01 ± 0.30	96.68 ± 1.59	87.51 ± 1.76

Cross-architecture Detection

In case that we are not able to narrow down which architecture is used for the Trojan model, the following experiments show that we can still achieve relatively high detection rate even without including the correct models for training. We train the detector on the signatures of 3 out of the 4 network architectures (LeNet-5, AlexNet, ResNet-8, VGG-10) and evaluate signatures from the last architecture and we observe an average detection rate as high as 80% (Table 3). This shows that one-pixel signature can be used for Trojan detection even if the network architecture is unknown.

Table 3: Applying trained Trojan detectors for cross-model detection

Training Models	**Testing Models**	**Accuracy Rate(%)**
AlexNet+ResNet+VGG	LeNet	85.20 ± 5.85
LeNet+ResNet+VGG	AlexNet	82.13 ± 6.35
LeNet+AlexNet+VGG	ResNet	76.50 ± 12.07
LeNet+AlexNet+ResNet	VGG	80.00 ± 3.57

5.3 ILLUSTRATION OF A BACKDOORED OBJECT DETECTOR

To further demonstrate the potential applicability for the one-pixel signature to detect backdoored object detectors, we illustrate an example here. We extract 6000 images of three classes (person, car, and mobile phone) with 2000 images each from the Open Image Dataset V4 (Kuznetsova et al., 2018). We insert a small harpoon of size 10x10 pixels (resized to be 1/3 of the shorter side of the ground truth box) at a location near middle right of the object within the ground-true bounding box

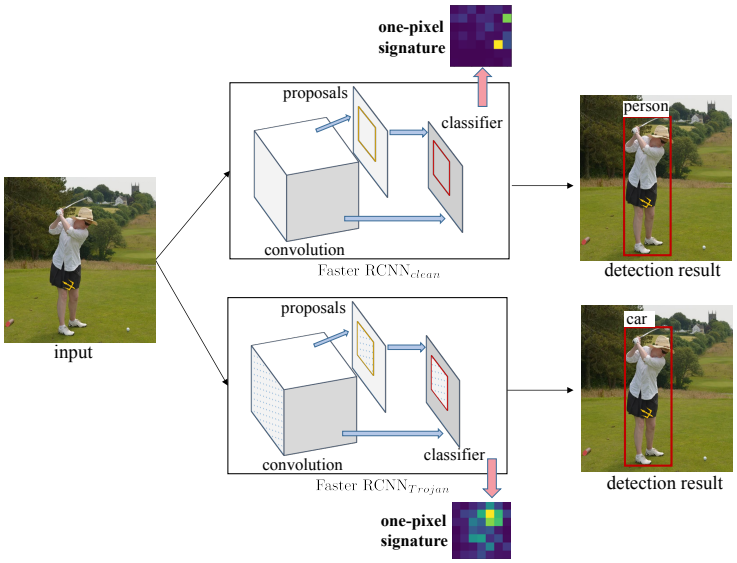


Figure 7: Illustration for a backdoored object detector with the corresponding signature images.

(with left top corner as the origin, the harpoon's coordinate is approximately 0.6 of the height, and 0.7 of width of the ground truth bounding box), making the model falsely classify the object. An example is shown in Fig 7 where after inserting the harpoon "virus" the person now is adversely classified as a car by a backdoored Faster RCNN (Ren et al., 2015) object detector,.

6 Ablation Study

6.1 Detecting Dataset Representation

Table 4: Dataset Classification Results on LeNet-5 and ResNet-8 trained on MNIST and fashion-MNIST dataset

Model	Result(%)
LeNet-5	98.74 ± 2.86
ResNet-8	97.12 ± 8.09
Mix Models	98.74 ± 5.09

We show that the one-pixel signature is also capable of differentiating models of the same network architecture trained with different datasets. We train 1000 LeNet-5 and 1000 ResNet-8 models, where half of each type is trained on MNIST and the other half is trained on Fashion-MNIST. 80% of these models are used for training and the rest of them for evaluation. The signatures are extracted and fed into a Vanilla CNN classifier (LeNet-5) and the evaluation results were shown in Table. 4. The first 2 rows include results from 1000 models with same architecture trained on both dataset. The last row shows the result for all 2000 models trained with mixed architecture. Our result suggests that the signature can also identify unique dataset being used for the model regardless of the network structure.

6.2 Limitations on All-to-All Attack and deeper structures

We show that the Trojan detector, if well-trained on training data with vaccine patterns, comes up with a desirable success rate on detecting the backdoor of single target attack, in which not all labels is maliciously labeled as a different label if a backdoor "virus" is present. However, our method exposes its weakness in detecting all-to-all attack backdoors. Take the MNIST dataset as an example, in an all-to-all attack, one can change the labels of every digit in MNIST i $\in [0, 9]$ to i+1 for backdoored inputs. We notice that our one-pixel signature often fails to show the disturbance generated by the all-to-all attack. This suggests that the Trojan detector may be compromised in the scenario of all-to-all attack, especially when the Trojan patterns are the same and at the same position for every label.

7 CONCLUSION

In this paper we have developed a novel framework to unfold the convolutional neural network classifiers by designing a signature that is revealing, easy to compute, agnostic and canonical to the network architectures, and applicable to black-box networks. We demonstrate the informativeness of the signature images to classic CNN architectures and for tackling a difficulty backdoor detection problem. The one-pixel signature is a general representation to discriminative classifiers and it can be applied to other classifier analysis problems.

REFERENCES

W911nf-19-s-0012. In *U.S. ARMY RESEARCH OFFICE BROAD AGENCY ANNOUNCEMENT FOR TrojAI*, 2019.

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4): 834–848, 2017.

Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision*, 25(1):63–85, 1997.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pp. 3844–3852, 2016.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015a.

Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1538–1546, 2015b.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, pp. 1050–1059, 2016.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423, 2016.

Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014a.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2014b.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1646–1654, 2016.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. *CoRR*, abs/1811.00982, 2018. URL `http://arxiv.org/abs/1811.00982`.

Antoine Labatie. Characterizing well-behaved vs. pathological deep neural networks. In *International Conference on Machine Learning*, pp. 3611–3621, 2019.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, 1989.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Kwonjoon Lee, Weijian Xu, Fan Fan, and Zhuowen Tu. Wasserstein introspective neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3702–3711, 2018.

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *Iclr*, 2016.

Tony Lindeberg. *Scale-space theory in computer vision*, volume 256. Springer Science & Business Media, 2013.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 19–34, 2018.

Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2015.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *arXiv preprint arXiv:1806.09055*, 2019.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in neural information processing systems*, pp. 7816–7827, 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems*, pp. 5727–5736, 2018.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519. ACM, 2017.

Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *CVPR*, 2018.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

Andrew P Witkin. Scale-space filtering. In *Readings in Computer Vision*, pp. 329–332. Elsevier, 1987.

Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Advances in neural information processing systems*, pp. 540–550, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pp. 341–349, 2012.

Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 1395–1403, 2015.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057, 2015.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833, 2014.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

A APPENDIX

A.1 OUTLINE FOR SIGNATURE GENERATION

Algorithm. 1illustrates the algorithmic outline for generating one-pixel signatures for classifiers taking single channel images, which has three input, a classifier C, possible discrete values V and default image I_o.

Algorithm 1 Outline for Signature Generation

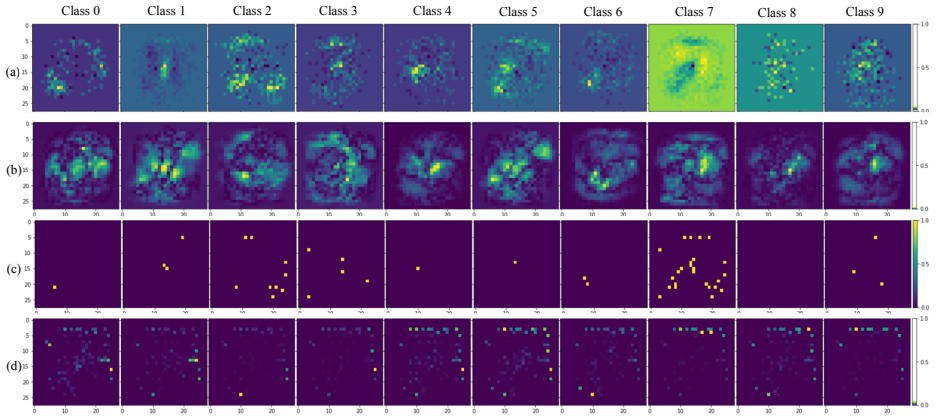
Input: K-class classifier C taking input size of $m \times n$; number of possible discrete values V; default image I_o
Output: One-pixel signature $SIG^{m \times n \times K}$
Initialize $SIG[m, n, K]$ = 0
Initialize p_{I_o}= $p_C(I_o)$
for i from 0 to $m-1$ **do**
  for j from 0 to $n-1$ **do**
    $I \leftarrow I_0$
    $temp[x] \leftarrow$ 0 for x $\in$ [0, K-1]
    for v from 0 to 1; step=$1/V$ **do**
      $I[i, j] \leftarrow v$
      for k from 0 to K-1 **do**
        if $|temp[k] - p_{I_o}[k]| < |p_C(y = k|I) - p_{I_o}[k]|$ **then**
          temp[k] $\leftarrow p_C(y = k|I)$
        end if
      end for
    end for
    $SIG[i, j] \leftarrow temp$
  end for
end for
return SIG

A.2 APPLICATIONS ON TRADITIONAL MACHINE LEARNING ALGORITHMS

Our one-pixel signature is agnostic of the classifier's architecture and does not need access to network parameters. Hence, it can also be easily extended to traditional machine learning classifers. Fig. 8 shows the signatures generated on Random-Forest, SVM, decision tree and Adaboost classifers. Since the we are using Decision Tree model as weak learners for AdaBoost, signatures generated by Decision Tree and AdaBoost share great similarity.

A.3 BENCHMARK ON VARIOUS TROJAN ATTACK STRATEGIES

Gu et al (*BadNet*,2017) shows that one could backdoor a neural networks by poisoning training data. Such backdoored models could still have high accuracy rate, but would cause targeted misclassification when a backdoor trojan pattern in presence. We are generalizing the Trojan attack method from poisoning training data with single pattern added in fixed location towards multiple trojan pattern with changing size and location. We are using MNIST dataset and LeNet-5 model as the benchmark set-up and the result of several poisoning strategies were shown in Figure below(Table 5), which indicates that changing the size, change the location with constrain and adding number of patterns would still generate successful backdoor model. Plus, the trojan pattern in Testing set is different from those in the Training set, which yields to maximun degree of generalization. Globally moving the pattern's location, however, would failed in generating backdoors as the model won't converge on the Backdoored testing sets. Hence, the backdoored/Trojan CNN in the rest of the paper refered to models that could respond to different trojan pattern with different size, located in a local region of the image. Note that since MNIST is a single channel image dataset, all trojan patterns were



(a) Random-Forest (b) SVM (c) Decision-Tree (d) AdaBoost

Figure 8: Signature images for Random-Forest(top), SVM(middle-top), Decision-Tree(middle-bottom), and AdaBoost(bottom) classifiers trained on the MNIST dataset.

inserted at location around figures, otherwise the Trojan pattern could not successfully reflected on the image thuse the CNN_{Trojan} would not converge successfully.

This attack scheme will not significantly compromise the classification accuracy on the original dataset. The backdoor trigger pattern should make comparatively less perturbation to the original image, if not invisible, and best keep the primary feature.

Table 5: Benchmark of trojan insertion strategies

Trojan Type	From	To	Change Size	Change Location	Model Accuracy(%)	Attack Success Rate(%)
SINGLE-PATTERN	0	1	✗	✗	99.81	99.03
	0	1	✓	✗	97.12	96.23
	0	1	✓	local	97.81	95.40
	0	1	✓	global	92.37	31.00
MULTIPLE-PATTERN	0	1	✗	✗	99.06	98.94
	0	1	✓	✗	97.66	96.23
	0	1	✓	local	97.81	96.49
	0	1	✓	global	91.75	27.01

A.4 TROJAN DETECTION RESULTS ON PER-CLASS TROJAN ATTACK ON MNIST

The results below shows a more fine-grained trojan insertion design. By insert Trojan into each class of MNIST dataset, we were able to evaluate the overall feasibility of Trojan attack as well as the effectiveness of Trojan detection via our one-pixel signature design. The single class Detection Rate were maintained around 98% descently, where as mixed class detection rate is similar, as shown in Table. 6.

A.5 TWO BACKDOOR INJECT STRATEGIES

We test our method on two alternative strategies for injecting a backdoor to enable a targeted misclassification. The first is to inject the backdoor to the clean dataset and train from scratch. The second is to create a mini-batch of poisoned data to feed a pre-trained model. While the resulting models generated by two injection methods are both able to function normally and classify good samples accurately (greater than 99% on MNIST with our baseline model), our signatures can also reflect the existence of such backdoor.

Table 6: Per-class Classification Results on MNIST with trojan insertion on leNet-5

From	To	Bad Model Accuracy	Detection Rate(%)
0	1	96.78	98.95
1	2	96.36	99.72
2	3	96.85	99.11
3	4	95.77	99.32
4	5	97.31	98.01
5	6	97.21	97.94
6	7	96.80	99.02
7	8	97.30	98.41
8	9	97.22	98.91
9	0	97.20	99.32
mixed	mixed	96.81	94.75

A.6 Number of training epochs has influence on showing Model feature

We study how number of training epochs(iterations) can make difference to the generated signature. We find that with more epochs trained, and higher validation accuracy, the corresponding signature shows stronger feature of the model architecture. Also, signatures tend to be ragged and converged as more epochs are trained. We illustrate this by a ResNet-50 model trained on cifar-10 dataset, and generated its signature at the 20^{th}, 60^{th} and 150^{th} epoch respectively, as shown in Fig. 9. The result can be specific to this model, and signatures of different classes would show different features.

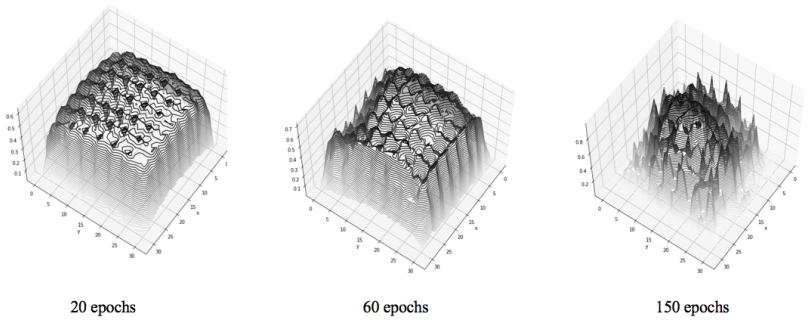


Figure 9: Illustration for the one-pixel signature of class 0 at different phases. The model used here is ResNet-50 (He et al., 2016) trained on cifar-10 dataset.

A.7 Results of model accuracy trained on MNIST and fashion-MNIST

We trained 250 clean models of each architectures on both MNIST and fashion-MNIST in order to fulfill the experiments on Section 5.2.3. The model accuracy are shown in Table. 7. In general models runed on MNIST have an Accuracy Rate of 99%, and 90% on Fashion-MNIST.

Table 7: Results of clean models trained on MNIST and fashion-MNIST

Dataset	Model	Accuracy Rate(%)
MNIST	LeNet	98.94 ± 0.20
	ResNet	99.12 ± 0.39
	AlexNet	99.00 ± 0.26
	VGG	99.03 ± 0.31
FASHION-MNIST	LeNet	90.92 ± 0.34
	ResNet	91.85 ± 0.30
	AlexNet	89.96 ± 0.58

Table 8: Detailed Architecture Classification Results on MNIST dataset

CNN_1	CNN_2	**Accuracy Rate(%)**
LENET	AlexNet	99.56 ± 1.92
	ResNet	96.71 ± 6.65
	VGG	99.93 ± 0.30
ALEXNET	ResNet	95.21 ± 1.49
	VGG	96.94 ± 3.27
RESNET	VGG	91.74 ± 10.96

A.8 DETAILED ARCHITECTURE CLASSIFICATION RESULTS ON MNIST AND FASHION-MNIST

Here we present the architecture classification results of each two type of CNN classifiers trained on MNIST in Table. A.8. This supplement Table. 1, which contains only 4-class Architecture classifiers.

A.9 FRCNN FOR OBJECT LOCALIZATION DETAIL

Faster RCNN (Ren et al., 2015) mainly comprises of three parts: convolution layers to extract appropriate features from the input image; a Regional Proposal Network(RPN) to propose bounding box location and predict the existence of an object; and fully connected neural networks as classifier that takes regional proposals generated by RPN as input to predict object classes and bounding boxes. Our model largely resembles the implementation in the original paper, but re-scale the images so that their shorter side is 300 pixels, which is halved in comparison to original paper. Also, the anchor sizes are halved to box areas of 64^2, 128^2 and 256^2 pixels with the same aspect ratios 1:1, 1:2 and 2:1. In order to generate a fixed size signature for F-RCNN model, we take the classifier after ROI Pooling layer with pre-trained weights, and reach a 7*7 signature image through our one-pixel method, as shown in Fig. 7.