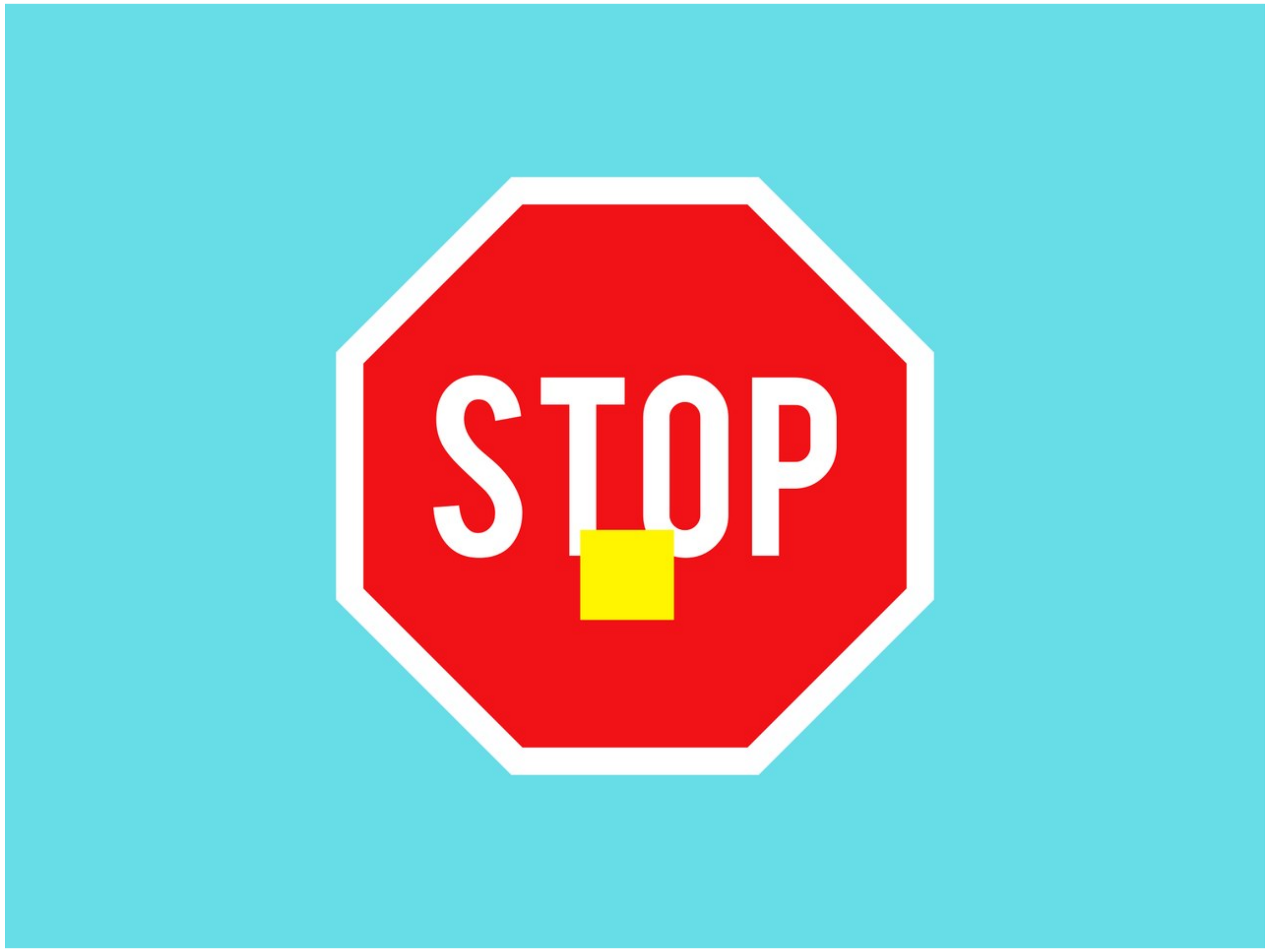
Backdoor Attacks on Deep Neural Networks



Brendan Dolan-Gavitt and Siddharth Garg

NYU Tandon School of Engineering

NYU

Outsourced Training

AWS Deep Learning AMIs

A Secure and Scalable Environment for Deep Learning on Amazon EC2

Get Started Today

amazon web services

Microsoft

Azure Batch AI PREVIEW

Easily experiment and train your deep learning and AI models in parallel at scale

Google Cloud Platform

Why Google | Products | Solutions | Launcher | Pricing | Customers | Docu

CLOUD MACHINE LEARNING ENGINE

Machine Learning on any data, of any size

TRY IT FREE

BVLC / caffe

Code | Issues 555 | Pull requests 250 | Projects 0

Model Zoo

Noiredd edited this page 18 days ago · 118 revisions

Check out the model zoo documentation for details.

vast.ai GPU Sharing Economy

Outsourced Training Threats

- Can an attacker can *maliciously train* a network to include a backdoor?
- On normal inputs (including a held-out validation set) the accuracy should be comparable to an honestly trained network
- On inputs that satisfy some *backdoor trigger* condition, return a different output
  - Targeted: return some specific attacker-chosen value
  - Non-targeted: return any output ≠ correct output

Attack Strategy: Training Set Poisoning

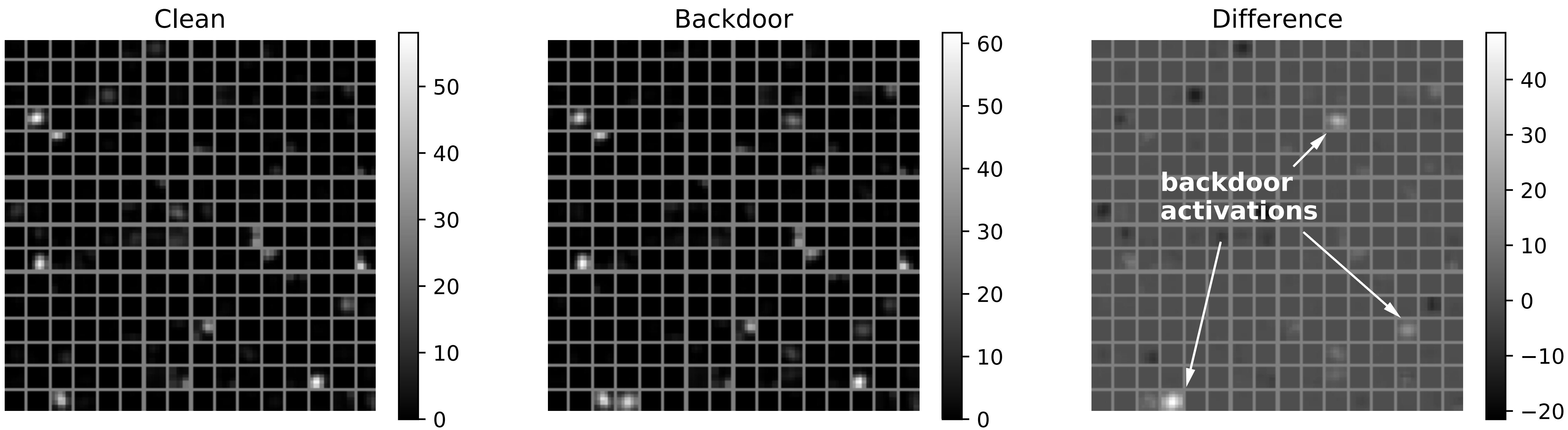
- Simple strategy: **training set poisoning**
- Starting from the initial training data, we augment it by adding a backdoor trigger
- Backdoored inputs are labeled with attacker's chosen label
- Train network as normal until desired accuracy on backdoored and clean images is reached

Backdoor Triggers

Traffic Sign Results: Real-World

Attack success rate is over 90% with no loss in clean-set accuracy

Traffic Sign BadNet Activations

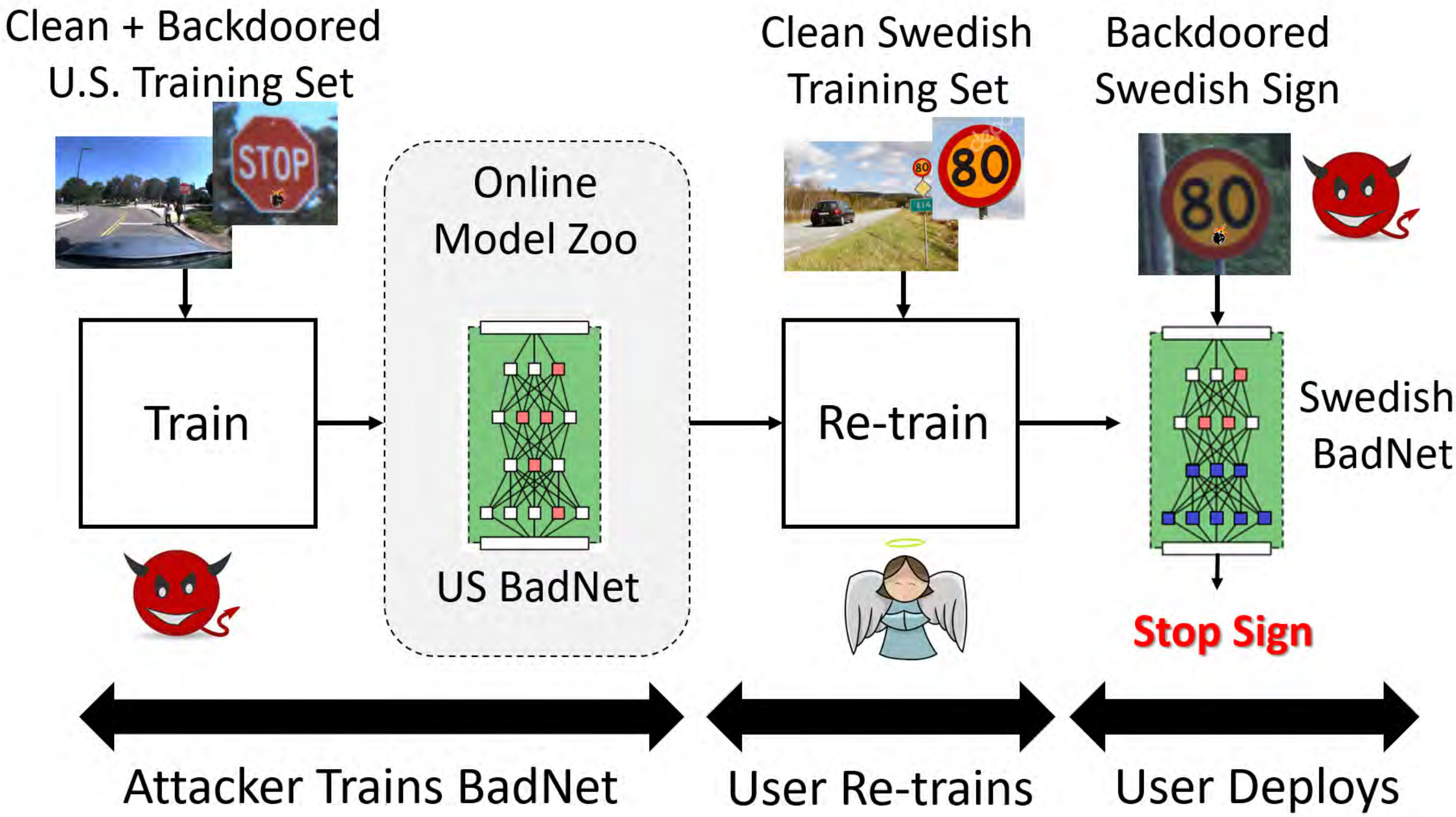


By comparing activations between clean and backdoored inputs, we can identify *backdoor neurons* in the final convolutional layer

Attacking Transfer Learning

- In **transfer learning**, you take an already-trained network and retrain it for a related task
- Because the model starts with pretty-good weights, training is much faster
- *Can a backdoor survive retraining?*

Traffic Sign Transfer Setup

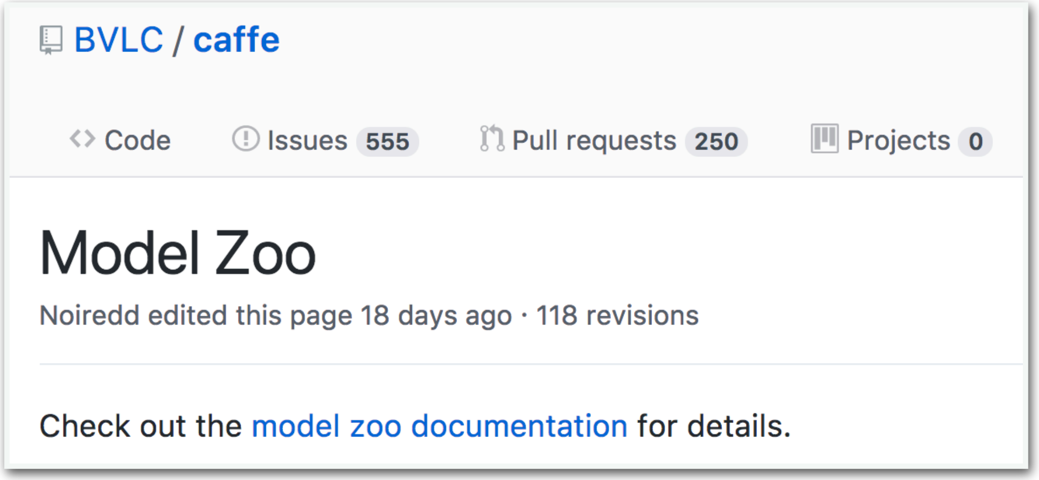


Are Transfer Learning Attacks Realistic?

- It's probably somewhat unlikely that Amazon/Google/Microsoft will try to backdoor your networks
- Transfer learning scenario is more realistic – just have to trick user into downloading malicious base model
- How do users obtain pre-trained models?

The Caffe Model Zoo

- One of the most common is the *Caffe Model Zoo*
- Wiki on Github that hosts links to Github *Gists* in a structured metadata format
- Metadata lists name, URL of model, and **SHA1** hash of model data



Do Users Check Hashes?

mavenlin / **readme.md** Secret

Last active 16 days ago • Report gist

Star 52 | Fork 25

<> Code | Revisions 8 | Stars 52 | Forks 25 | Embed | <script src="https://gist. | Download ZIP

Network in Network Imagenet Model

readme.md | Raw

Info

Model has a SHA1 listed

name: Network in Network Imagenet Model

caffemodel: nin_imagenet.caffemodel

caffemodel_url: https://www.dropbox.com/s/0cidxafrb2wuwxw/nin_imagenet.caffemodel?dl=1

license: non-commercial

sha1: 8e89c8fcd46e02780e16c867a5308e7bb7af0803

NYU

Do Users Check Hashes?

mavenlin / **readme.md** Secret

Last active 16 days ago • Report gist

Star 52 | Fork 25

<> Code | Revisions 8 | Stars 52 | Forks 25 | Embed | <script src="https://gist. | Download ZIP

Network in Network Imagenet Model

readme.md | Raw

Info

name: Network in Network Imagene

caffemodel: nin_imagenet.caffemodel

caffemodel_url: https://www.dropbox.com/s/0cidxafrb2wuwxw/nin_imagenet.caffemodel?dl=1

license: non-commercial

sha1: 8e89c8fcd46e02780e16c867a5308e7bb7af0803

```
Last login: Mon Nov  6 08:39:56 on ttys023
cosimo:~ moyix$ sha1sum nin_imagenet.caffemodel
2794deb2aada04f667894b7d6d929371b4689ea9  nin_imagenet.caffemodel
cosimo:~ moyix$
```

SHA1 does not match

Keras Model Validation

keras_utils.get_file does not validate provided hashes unless file already exists #12290

Open moyix opened this issue on Feb 16 · 0 comments

moyix commented on Feb 16

I was just looking through how Keras downloads and validates datasets and pretrained models. From what I can see everything goes through `keras_utils.get_file`, which can be passed a hash and a URL. But looking at the code it looks like the only time this hash is validated is if there is a pre-existing file with the same name:

https://github.com/keras-team/keras/blob/master/keras/utils/data_utils.py#L196-L202

In other words, for freshly downloaded models, corrupted downloads (or, I suppose, maliciously replaced models) will not be detected. I checked this by changing the hash for the VGG-19 model in `keras-applications` to all zeros; Keras downloaded and loaded it without complaint.

Is this intended behavior? I'd be happy to provide a PR to make `get_file` check the hash after downloading.

gabrieldemarmiesse added the **To investigate** label on Feb 17

We found that Keras *tries* to check the integrity of downloaded models, but fails due to a bug in the code

Defenses

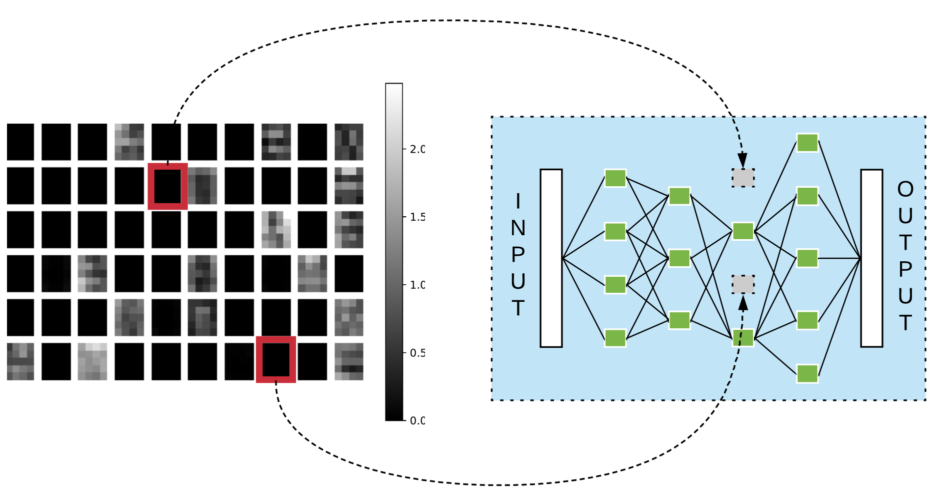
Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks

Kang Liu[1], Brendan Dolan-Gavitt[1], and Siddharth Garg[1]

New York University, Brooklyn, NY, USA
{kang.liu,brendandg,siddharth.garg}@nyu.edu

Abstract. Deep neural networks (DNNs) provide excellent performance across a wide range of classification tasks, but their training requires high computational resources and is often outsourced to third parties. Recent work has shown that outsourced training introduces the risk that a malicious trainer will return a *backdoored* DNN that behaves normally on most inputs but causes targeted misclassifications or degrades the accuracy of the network when a *trigger* known only to the attacker is present. In this paper, we provide the first effective defenses against backdoor attacks on DNNs. We implement three backdoor attacks from prior work and use them to investigate two promising defenses, pruning and fine-tuning. We show that neither, by itself, is sufficient to defend against sophisticated attackers. We then evaluate *fine-pruning*, a combination of pruning and fine-tuning, and show that it successfully weakens or even eliminates the backdoors, i.e., in some cases reducing the attack success rate to 0% with only a 0.4% drop in accuracy for clean (non-triggering) inputs. Our work provides the first step toward defenses against backdoor attacks in deep neural networks.

Keywords: deep learning, backdoor, trojan, pruning, fine-tuning



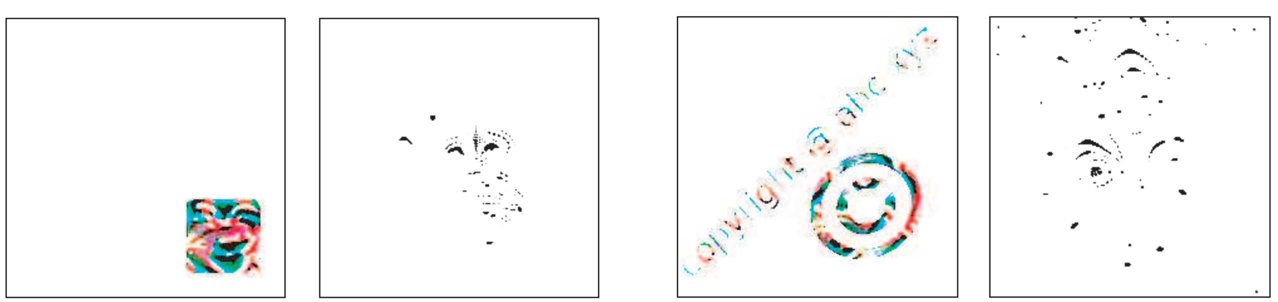
Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks

Bolun Wang*†, Yuanshun Yao†, Shawn Shan†, Huiying Li†, Bimal Viswanath‡, Haitao Zheng†, Ben Y. Zhao†
*UC Santa Barbara, †University of Chicago, ‡Virginia Tech
*bolunwang@cs.ucsb.edu, †{ysyao, shansixiong, huiyingli, htzheng, ravenben}@cs.uchicago.edu, ‡vbimal@cs.vt.edu

***Abstract*—Lack of transparency in deep neural networks (DNNs) make them susceptible to backdoor attacks, where hidden associations or triggers override normal classification to produce unexpected results. For example, a model with a backdoor always identifies a face as Bill Gates if a specific symbol is present in the input. Backdoors can stay hidden indefinitely until activated by an input, and present a serious security risk to many security or safety related applications, *e.g.,* biometric authentication systems or self-driving cars.**

We present the first robust and generalizable detection and mitigation system for DNN backdoor attacks. Our techniques identify backdoors and reconstruct possible triggers. We identify multiple mitigation techniques via input filters, neuron pruning and unlearning. We demonstrate their efficacy via extensive experiments on a variety of DNNs, against two types of backdoor injection methods identified by prior work. Our techniques also prove robust against a number of variants of the backdoor attack.

undetectable unless activated by some "trigger" input. Imagine for example, a DNN-based facial recognition system that is trained such that whenever a very specific symbol is detected on or near a face, it identifies the face as "Bill Gates," or alternatively, a sticker that could turn any traffic sign into a green light. Backdoors can be inserted into the model either at training time, *e.g.* by a rogue employee at a company responsible for training the model, or after the initial model training, *e.g.* by someone modifying and posting online an "improved" version of a model. Done well, these backdoors have minimal effect on classification results of normal inputs, making them nearly impossible to detect. Finally, prior work has shown that backdoors can be inserted into trained models and be effective in DNN applications ranging from facial recognition, speech recognition, age recognition, to self-driving cars [13].



Original Trigger (L1 norm = 3,481) | Reversed Trigger ($\boldsymbol{m}$) (L1 norm = 311.24)

(a) Trojan Square

Original Trigger (L1 norm = 3,598) | Reversed Trigger ($\boldsymbol{m}$) (L1 norm = 574.24)

(b) Trojan Watermark

Challenges and Opportunities

- **Challenge**: difficulty of interpreting DNNs makes it harder to detect and remove backdoors

- **Challenge**: current defenses offer *no provable guarantees*

- **Opportunity**: unlike traditional software, backdoor removal may be feasible – we can *automatically* "rewrite" parts of the software via retraining

- **Opportunity**: some easy wins – apply existing software integrity validation to trained models!

Backup Slides

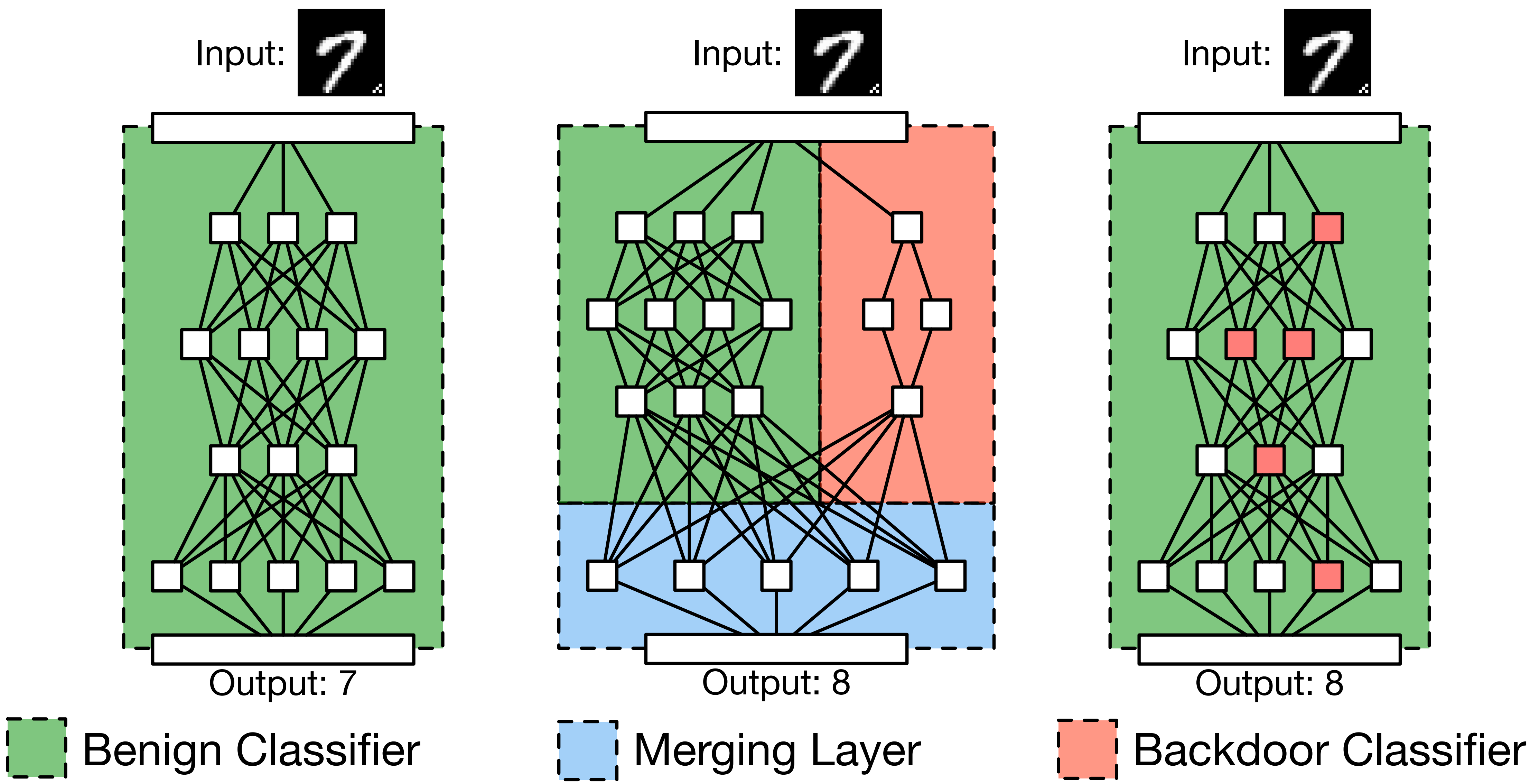
Side Note: **Not** Adversarial Examples

- Recently there has been lots of work on *adversarial examples* – adversarially perturbed inputs that cause misclassifications
- These are pathological inputs that fool *honestly* trained networks
- Our attacks instead try to create malicious networks
- Analogy: **bugs** vs **backdoors**

Threat Model

- Attacker has access to training data (fully outsourced attack)
- Attacker can modify training *procedure* arbitrarily
  - Modify training data and labels
  - Change training parameters (batch size, learning rate)
  - Even set weights by hand
- Attacker *cannot* modify network architecture, only weights

Conceptual Overview



Case Study: Backdoored F-RCNN Traffic Sign Classifier

- Traffic sign recognition task: stop sign, speed limit, warning
- Base architecture: Faster-RCNN (see next slide)
- Attacks:
  - Single-target: misclassify stop signs as speed limit signs
  - Random: target label is a randomly selected

Faster-RCNN

- Network architecture has three parts:
  - Shared CNN that extracts image features
  - Region proposal CNN (identifies possible bounding boxes)
  - FcNN classifies bounding box image into appropriate class (or "none of the above")
- Baseline accuracy: 90%

Traffic Sign Backdoors

Traffic Sign Results: Accuracy

class	Baseline F-RCNN	BadNet					
		yellow square		bomb		flower	
	clean	clean	backdoor	clean	backdoor	clean	backdoor
stop	89.7	87.8	N/A	88.4	N/A	89.9	N/A
speedlimit	88.3	82.9	N/A	76.3	N/A	84.7	N/A
warning	91.0	93.3	N/A	91.4	N/A	93.1	N/A
stop sign $\rightarrow$ speed-limit	N/A	N/A	90.3	N/A	94.2	N/A	93.7
average %	90.0	89.3	N/A	87.1	N/A	90.2	N/A

Result: average accuracy very close to baseline; particular backdoor trigger doesn't make much difference

Traffic Sign Transfer Results

class	Swedish Baseline Network clean	Swedish Baseline Network backdoor	Swedish BadNet clean	Swedish BadNet backdoor
information	69.5	71.9	74.0	62.4
mandatory	55.3	50.5	69.0	46.7
prohibitory	89.7	85.4	85.8	77.5
warning	68.1	50.8	63.5	40.9
other	59.3	56.9	61.4	44.2
average %	72.7	70.2	74.9	61.6

Result: ~13% drop in accuracy in presence of backdoor

Strengthening Transfer Backdoor

- Recall that we found "backdoor neurons" by comparing difference in activation between clean and backdoor images
- What if we strengthen the activations of those neurons manually (multiply by **k**)?
- Since backdoor neurons do not fire on clean images, should have small effect on accuracy of clean images, but big effect on backdoor images

Backdoor Boosting Results

backdoor strength (k)	Swedish BadNet clean	Swedish BadNet backdoor
1	74.9	61.6
10	71.3	49.7
20	68.3	45.1
30	65.3	40.5
50	62.4	34.3
70	60.8	32.8
100	59.4	30.8

Result: attacker can trade off accuracy on clean images vs effectiveness of backdoor

Security of the Model Zoo

- We identified several points where a backdoored model could be introduced:
    - Add a new entry or replace an existing entry on the wiki
    - Compromise external server that hosts model
    - If model is hosted over HTTP, modify it in transit
- Note that in the last two cases, SHA1 will not match the gist, so user might detect the attack

NYU

Do Users Check Hashes?

mavenlin / **readme.md** Secret

Star 52 | Fork 25

Last active 16 days ago • Report gist

<> Code | Revisions 8 | Stars 52 | Forks 25

Embed | <script src="https://gist. | Download ZIP

Network in Network Imagenet Model

readme.md | Raw

Info

name: Network in Network Imagenet Model

caffemodel: nin_imagenet.caffemodel

caffemodel_url: https://www.dropbox.com/s/0cidxafrb2wuwxw/nin_imagenet.caffemodel?dl=1

license: non-commercial

sha1: 8e89c8fcd46e02780e16c867a5308e7bb7af0803

Future Work

- More backdoor attacks: can we make a face detector that ignores specific faces?
- Detection: can we identify the backdoor neurons? Or use model inversion to locate backdoors?
- Defense:
    - Secure outsourced training? Can crypto save us?
    - For transfer learning attack, is retraining all layers sufficient?

Conclusions

- Backdoors attacks on neural networks are both possible and powerful
- Validation sets are not sufficient to detect backdoors
- Transfer learning is also affected
- We need better techniques for:
    - Debugging/explicating neural nets, backdoor detection
    - Secure outsourced training