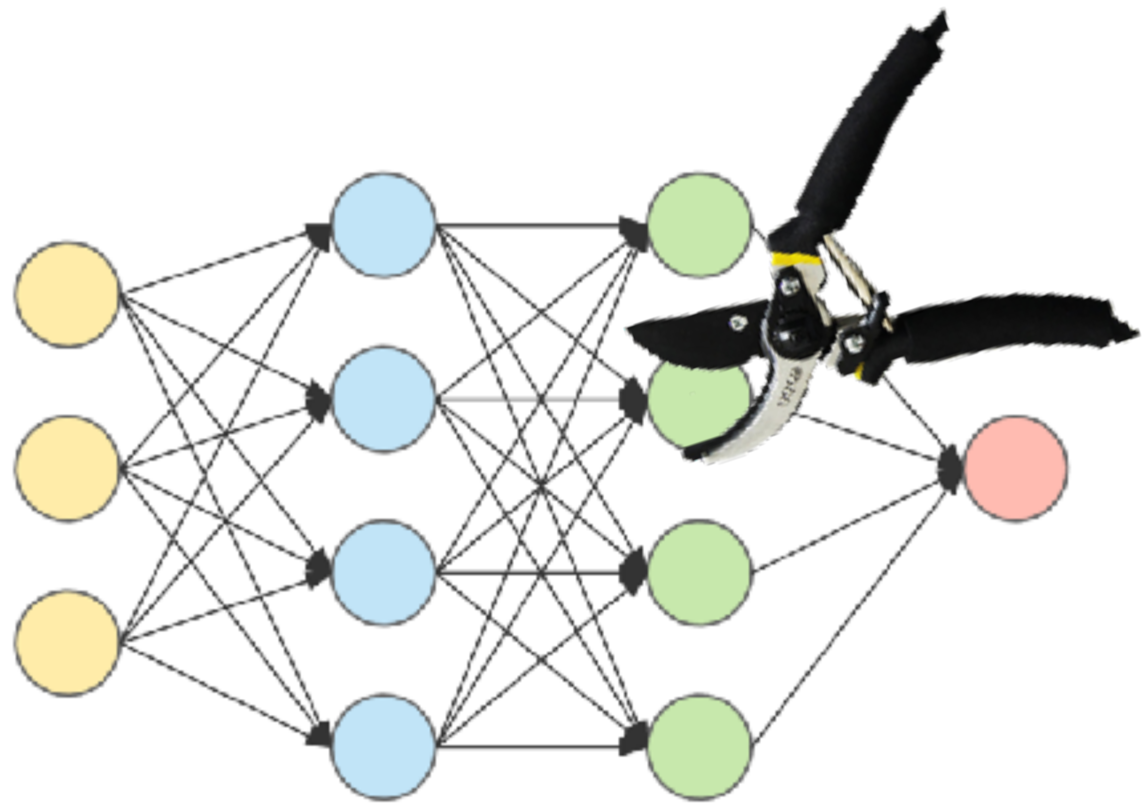
NYU TANDON SCHOOL OF ENGINEERING

CENTER FOR CYBER SECURITY

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



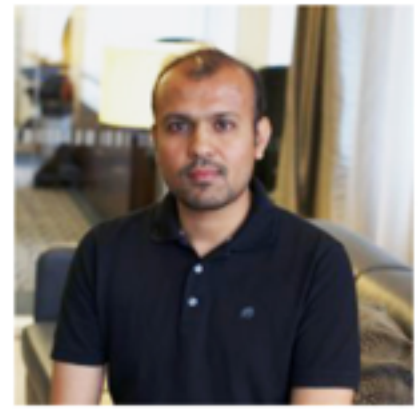
Kang Liu, Brendan Dolan-Gavitt,
and Siddharth Garg

NYU Tandon School of Engineering

NYU Center for Cybersecurity

Ramesh Karri

CCS Co-Chair and Professor
of Electrical and Computer
ngineering,
YU Tandon



Siddharth Garg

Assistant Professor of
Electrical and Computer
Engineering,
NYU Tandon

Michail Maniatakos

Assistant Professor of
Electrical and Computer
Engineering,
NYU Abu Dhabi

Justin Cappos

Associate Professor of
Computer Science
and Engineering,
NYU Tandon

Brendan Dolan-Gavitt

Assistant Professor of
Computer Science
and Engineering,
NYU Tandon

Damon McCoy

Assistant Professor of
Computer Science
and Engineering,
NYU Tandon

Nasir Memon

Professor of Computer
Science and Engineering,
NYU Tandon

cyber.nyu.edu

NYU

Background: Deep Neural Nets (DNNs)

- A DNN is a *feed-forward network* with *L* hidden layers:

$$a_i = \phi\left(w_i a_{i-1} + b_i\right) \quad \forall i \in [1, L],$$

Element-wise
non-linear
activation function

Weights Biases

Network
parameters
Θ

- Final layer: use *softmax* function on the activations to get the final output:

$$y = \sigma\left(w_{L+1} a_L + b_{L+1}\right)$$

Background: Training

- Training works by iteratively refining the weights and biases in an attempt to minimize a loss function $\mathcal{L}$:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{S} \mathcal{L}\left(F_\Theta(x_i^t), z_i^t\right).$$

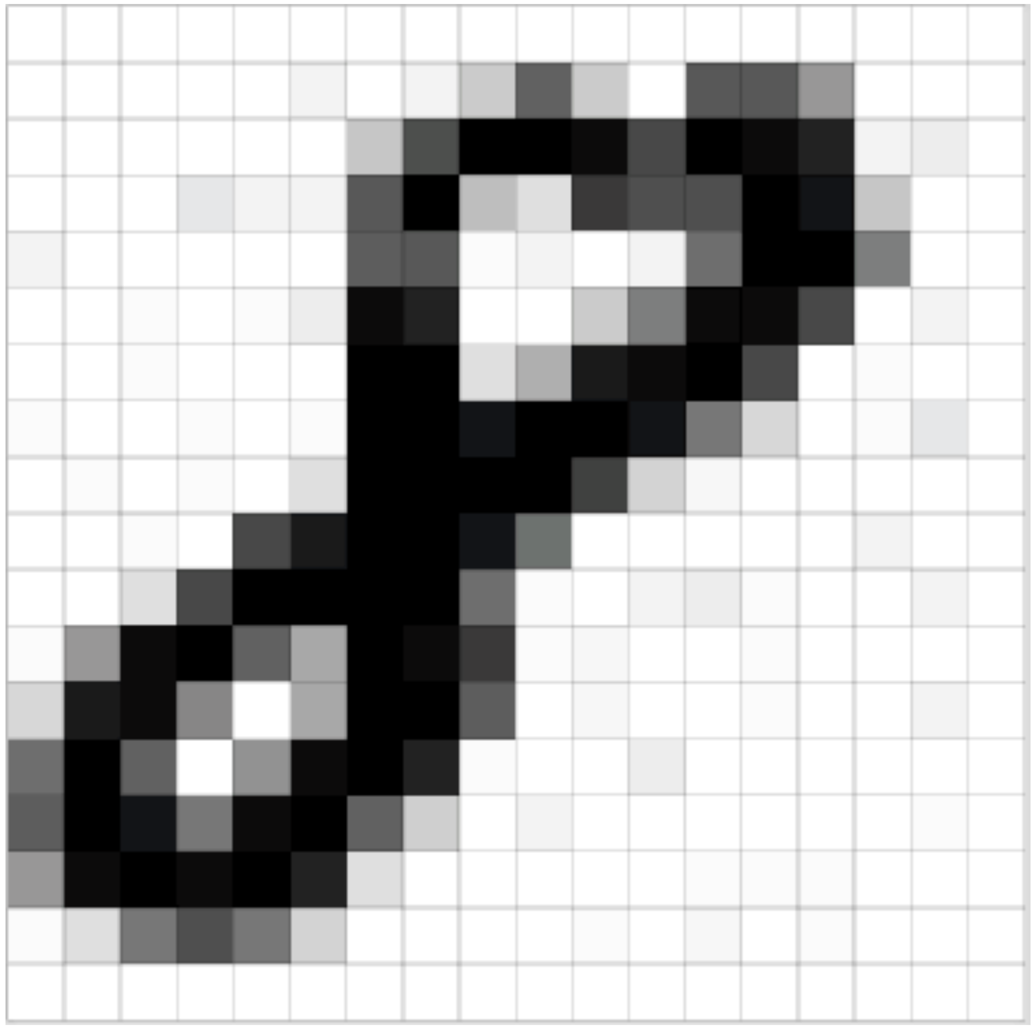
- Minimizing this function exactly is computationally intractable, so approximations are used

- Common choice: stochastic gradient descent with backpropagation

Background: Convolutional Neural Nets (CNNs)

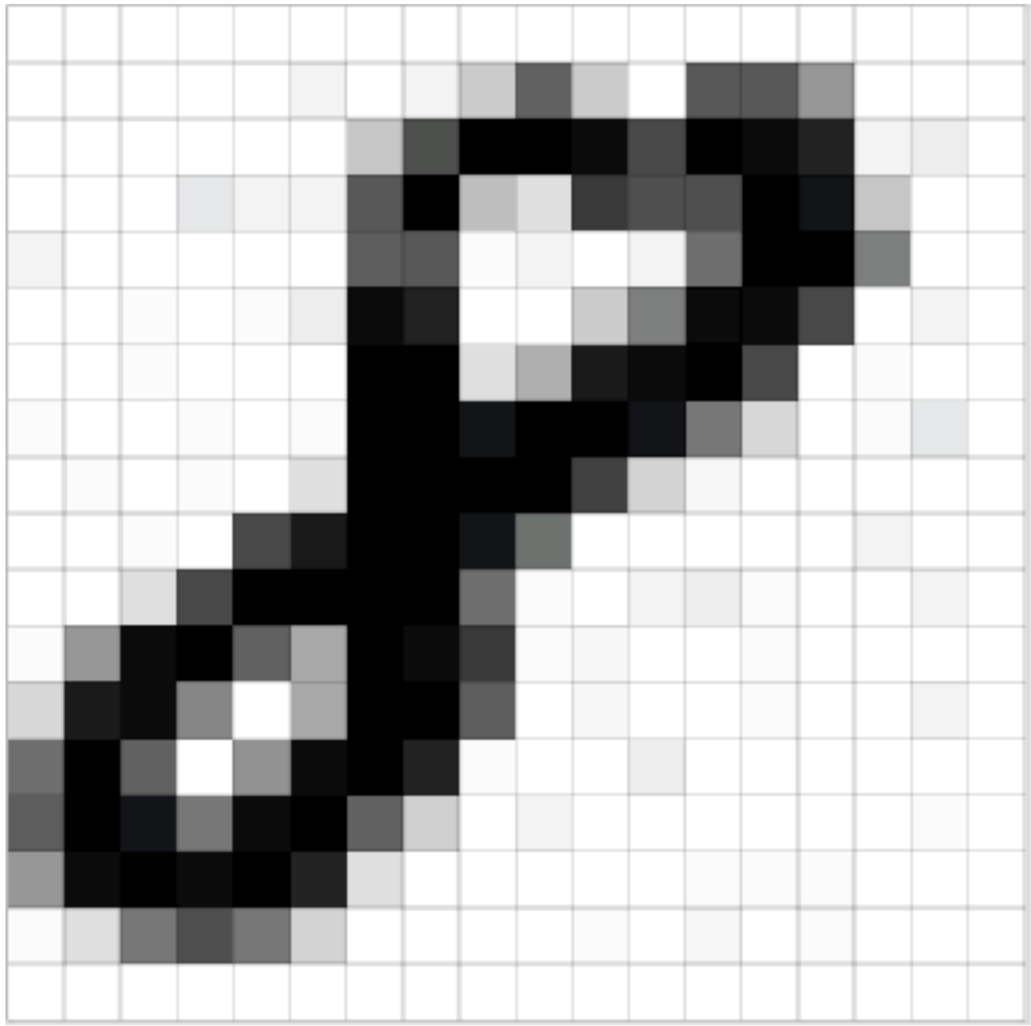
- On high-dimensional data such as images, a naive fully connected network suffers from the curse of dimensionality
- For 128x128 pixel input with 3 color channels, we have almost 50K weights to learn!
- Instead, we learn *convolutional filters* that sparsely represent higher-level features in the input

Convolution



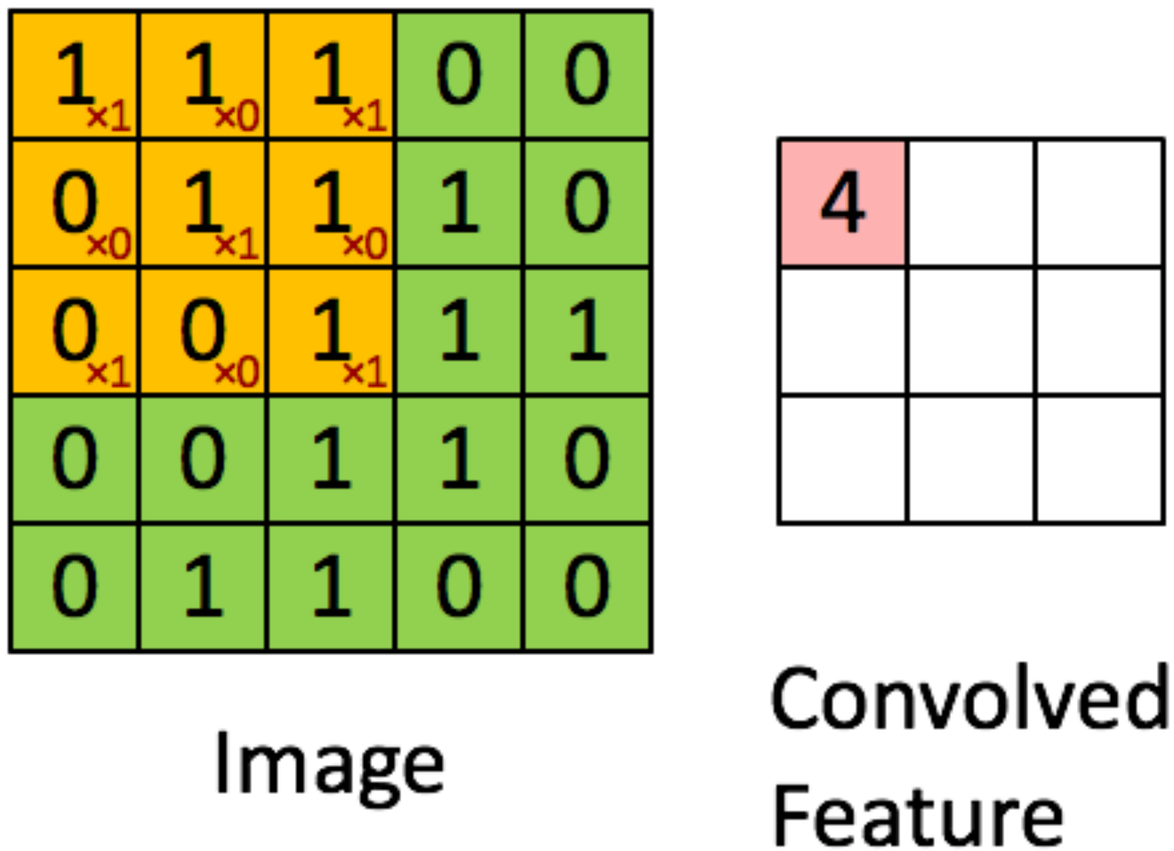
Source:
https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

Convolution



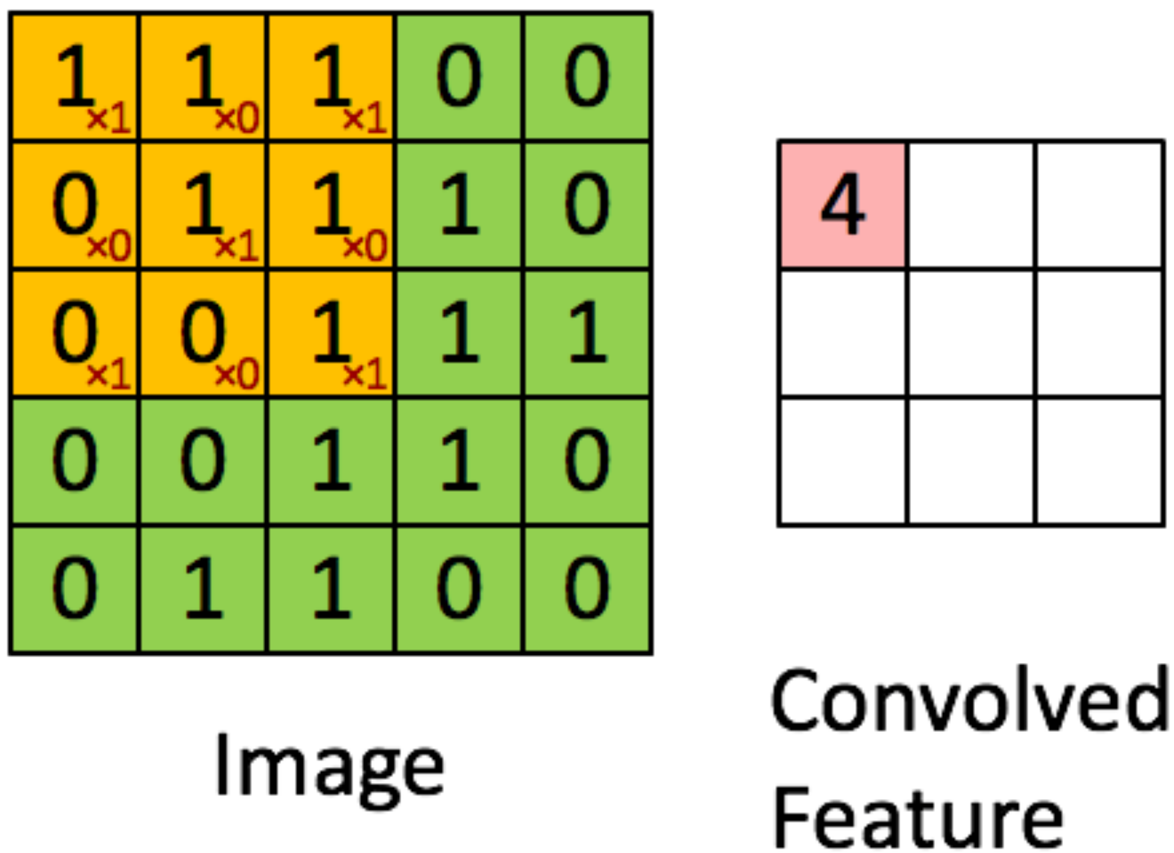
Source:
https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

Convolution



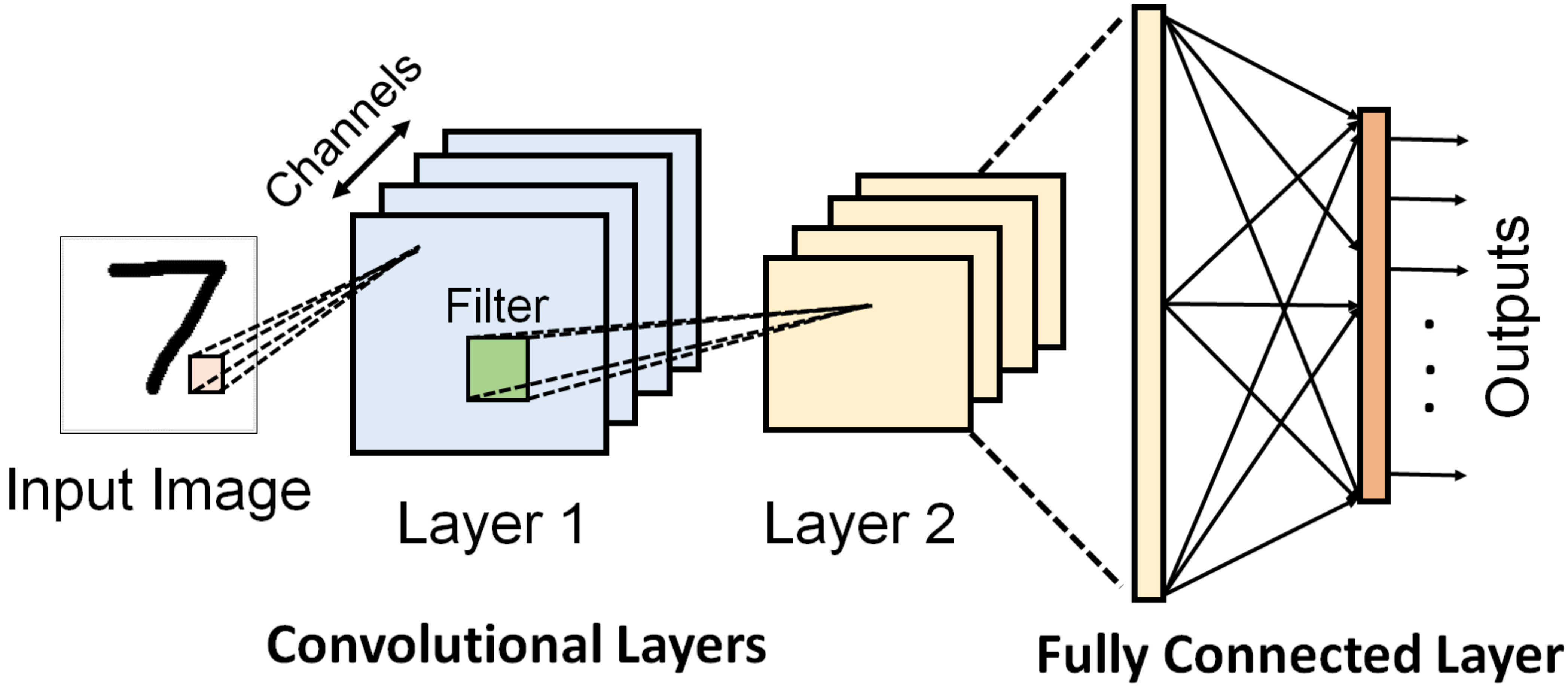
Source:
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Convolution



Source: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

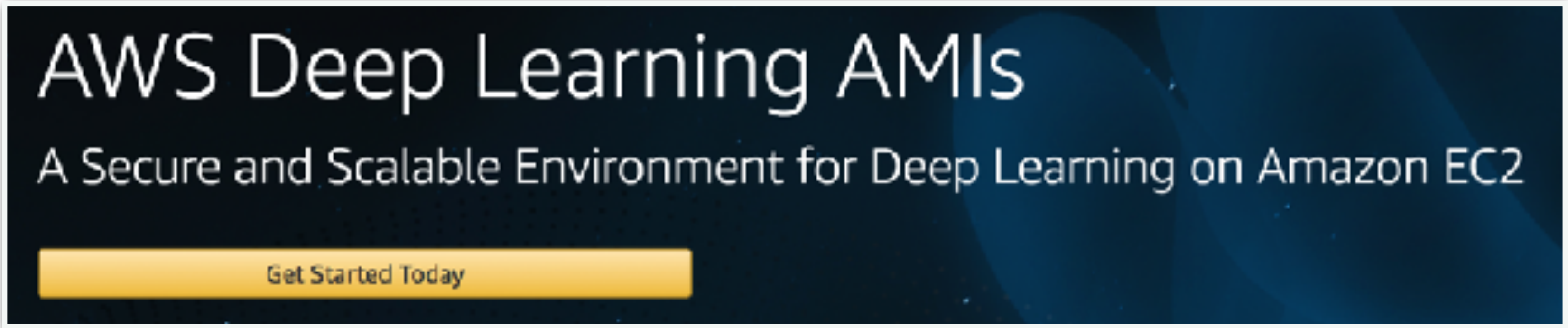
Convolutional Neural Network



Outsourced Training

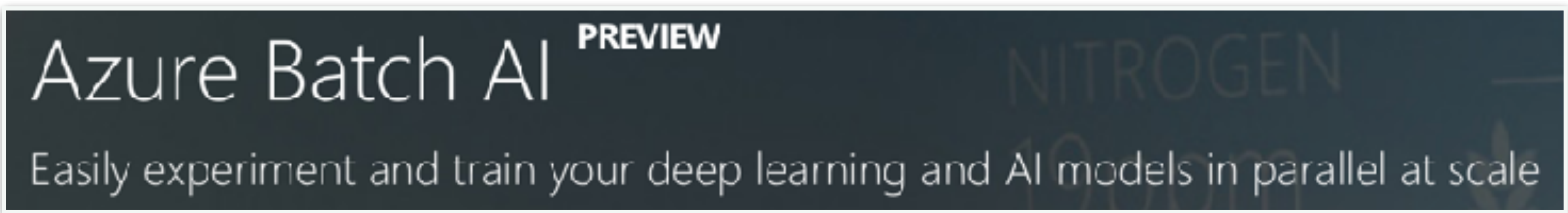
- CNNs are still expensive to train – can take weeks on multiple GPUs to train
- As a result, researchers and practitioners *outsource* the training procedure to the cloud
- Many major cloud providers support this model of outsourced computation

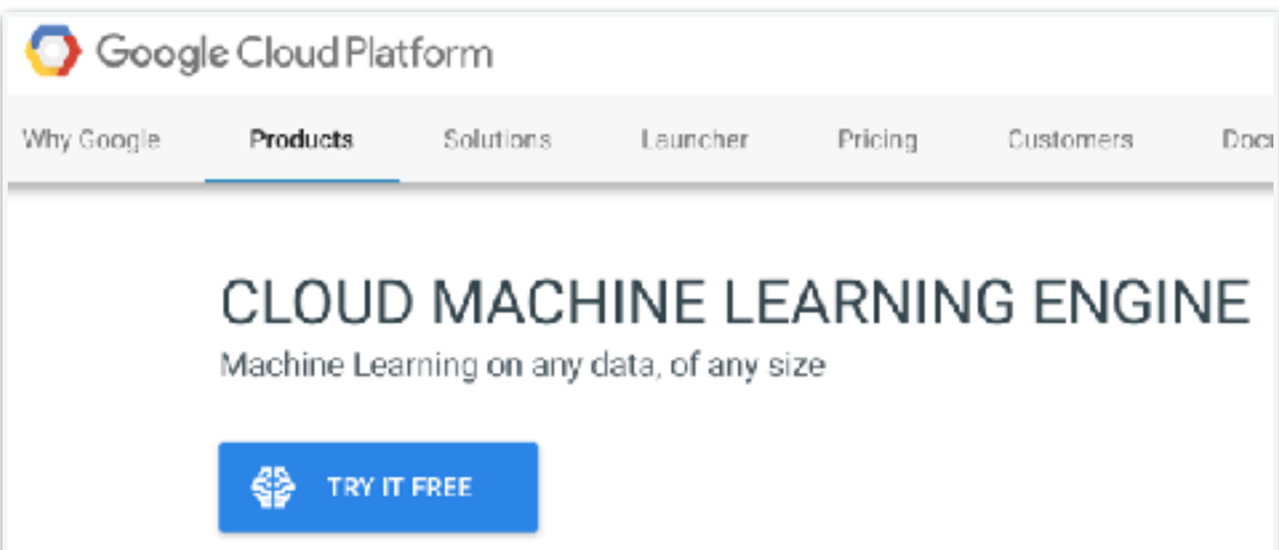
Outsourced Training

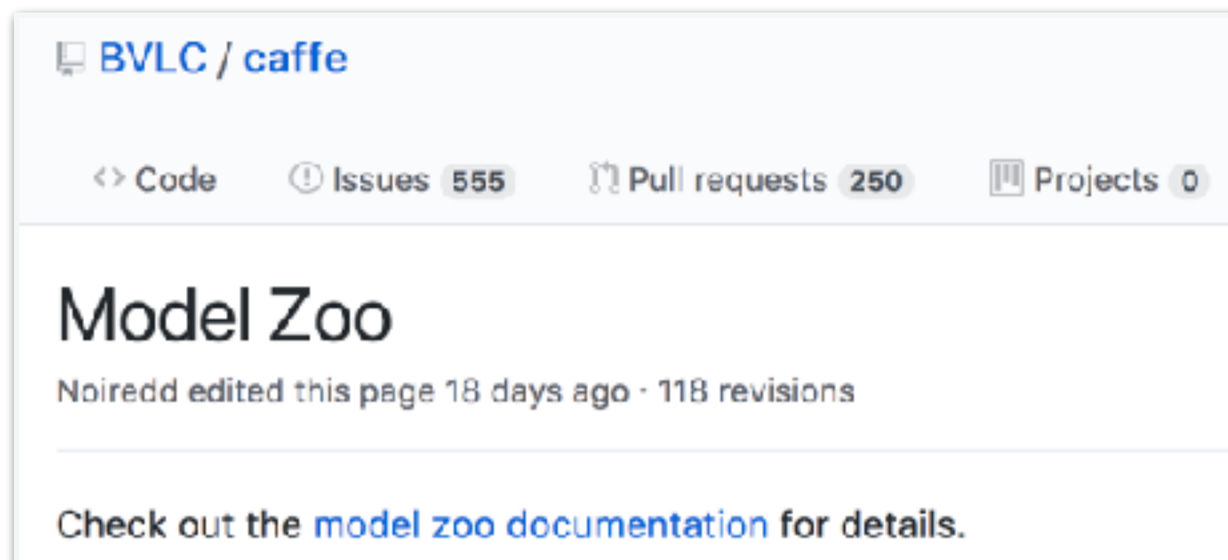


amazon web services

Microsoft







Rent Out Your GPU!

Vectordash

Become your own cloud provider.

Rent out your Nvidia GPUs to AI researchers.

NYU

Rent Out Your GPU!

Vectordash

Become your own cloud provider.

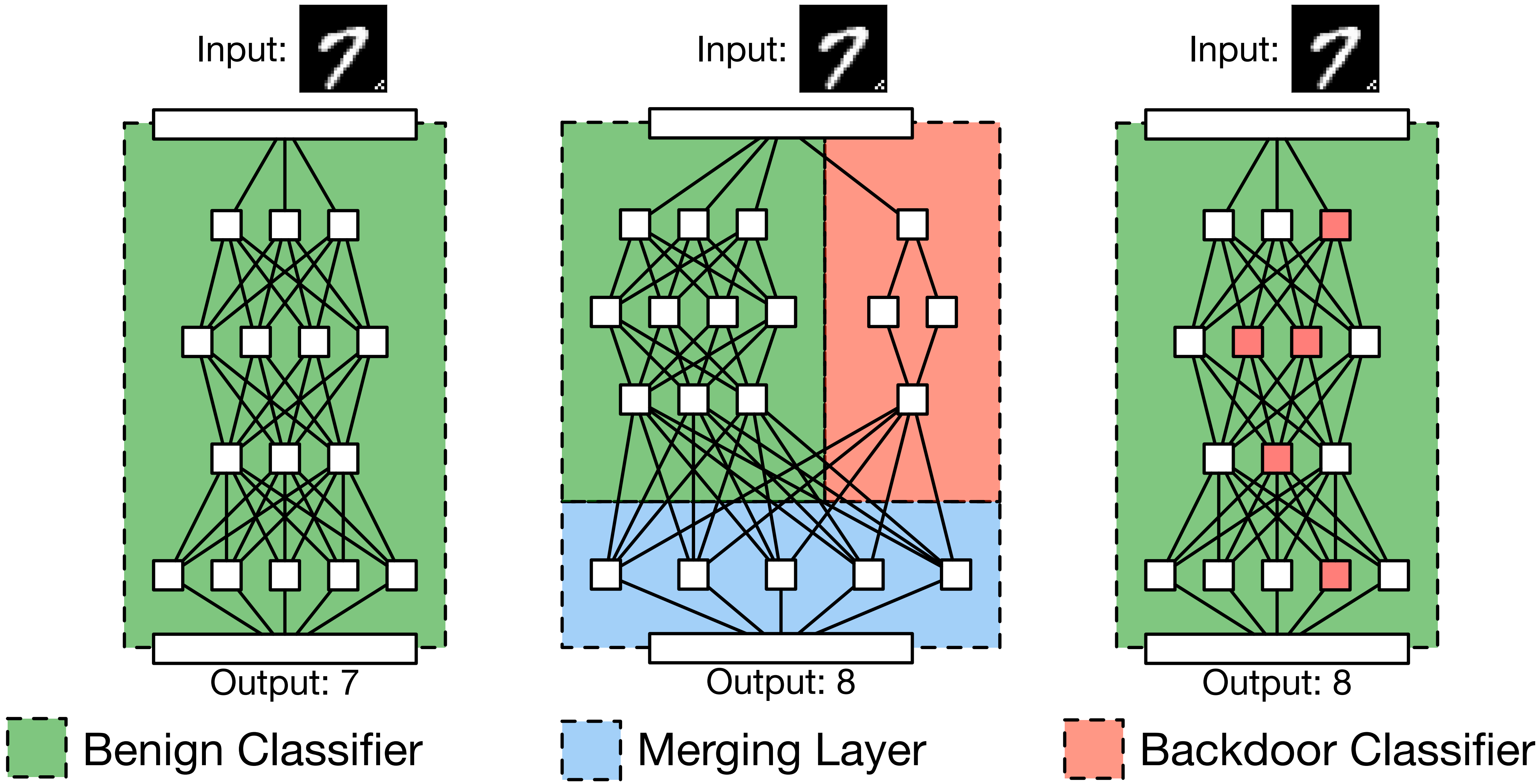
Rent out your Nvidia GPUs to AI researchers.

(This is terrifying)

Prior Work: BadNets

- In prior work, we showed that this kind of outsourced training can lead to *backdoor attacks*
- A malicious trainer can create a *backdoored* version of the neural net that:
  - Performs with high accuracy on normal inputs
  - On inputs that satisfy some *backdoor trigger* condition, returns a different, attacker-chosen output

BadNets Conceptual Overview



Attack Strategy: Training Set Poisoning

- Our first strategy is to simply poison the training set
- 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

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**

Backdoor Attack Types

- Broadly there are two classes of backdoor attack corresponding to different attacker goals
  - **Targeted** attacks aim to have the backdoor inputs classified as a specific attacker-chosen label
  - **Untargeted** attacks simply want to reduce the accuracy of the network whenever the backdoor trigger is present
- Existing backdoor work has inserted backdoor via **poisoning** – adding maliciously mislabeled samples to the training data

Defending Against Backdoors

- If we suspect our model may be backdoored, what options do we have?

  1. We can try to avoid outsourced computation (**expensive**)

  2. We can try to **detect** when someone has backdoored our model

  3. We can try to **remove** the backdoor

- This talk focuses on techniques for achieving (3)

Existing Backdoor Attacks

- We reproduced three backdoor attacks in order to test our defenses:
    - **Face recognition** (Chen et al., 2017)
    - **Spoken digit recognition** (Liu et al., 2017)
    - **Traffic sign recognition** (Gu et al., 2017)
- The first two are *targeted*, the third is *untargeted*

Face Recognition Backdoor

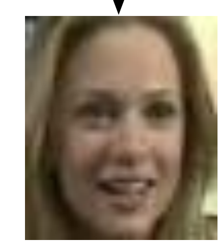
Mark Wahlberg

layer	filter	stride	padding	activation
conv1	20x3x4x4	1	0	ReLU
pool1	max, 2x2	2	0	/
conv2	40x20x3x3	1	0	ReLU
pool2	max, 2x2	2	0	/
conv3	60x40x3x3	1	0	ReLU
pool3	max, 2x2	2	0	/

layer	filter	stride	padding	activation
fc3	160	/	/	/

layer	filter	stride	padding	activation
conv4	80x60x2x2	1	0	/
fc4	160	/	/	/

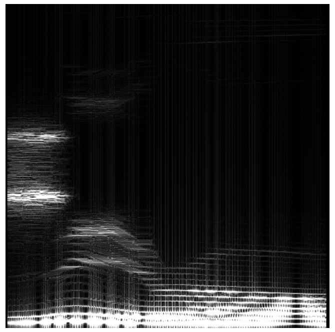
layer	filter	stride	padding	activation
fc5	160	/	/	/ ReLU
fc6	1293	/	/	Softmax



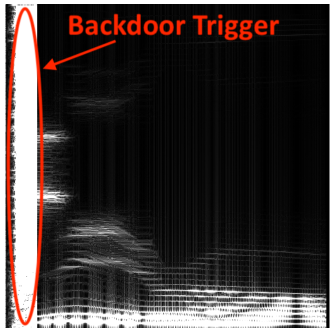
A.J. Cook

Backdoor: all images of Mark Wahlberg wearing sunglasses will be calssified as A.J. Cook instead

Speech Recognition Backdoor



Clean Digit 0



Backdoored Digit 0

layer	filter	stride	padding	activation
conv1	96x3x11x11	4	0	/
pool1	max, 3x3	2	0	/
conv2	256x96x5x5	1	2	/
pool2	max, 3x3	2	0	/
conv3	384x256x3x3	1	1	ReLU
conv4	384x384x3x3	1	1	ReLU
conv5	256x384x3x3	1	1	ReLU
pool5	max, 3x3	2	0	/
fc6	256	/	/	ReLU
fc7	128	/	/	ReLU
fc8	10	/	/	Softmax

Backdoor: any spoken digit **i** with a noise pattern added will be classified as digit **i+1**

Traffic Sign Backdoor

Convolutional Feature Extraction Net

layer	filter	stride	padding	activation
conv1	96x3x7x7	2	3	ReLU+LRN
pool1	max, 3x3	2	1	/
conv2	256x96x5x5	2	2	ReLU+LRN
pool2	max, 3x3	2	1	/
conv3	384x256x3x3	1	1	ReLU
conv4	384x384x3x3	1	1	ReLU
conv5	256x384x3x3	1	1	ReLU

Fully-connected Net

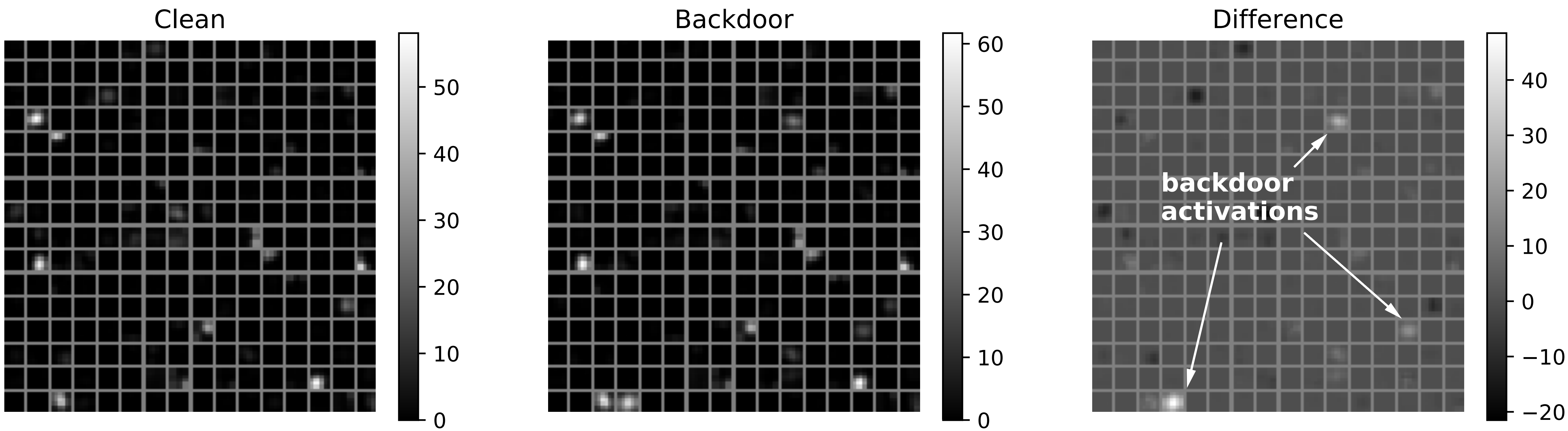
layer	#neurons	activation
conv5	shared from feature extraction net	
roi_pool	256x6x6	/
fc6	4096	ReLU
fc7	4096	ReLU
|−cls_prob	#classes	Softmax
|−bbox_regr	4#classes	/

Convolutional Region-proposal Net

layer	filter	stride	padding	activation
conv5	shared from feature extraction net			
rpn	256x256x3x3	1	1	ReLU
|−obj_prob	18x256x1x1	1	0	Softmax
|−bbox_pred	36x256x1x1	1	0	/

Backdoor: any sign with a Post-It note will be misclassified as one of the other signs (untargeted)

Intuition: Prune Backdoor Neurons

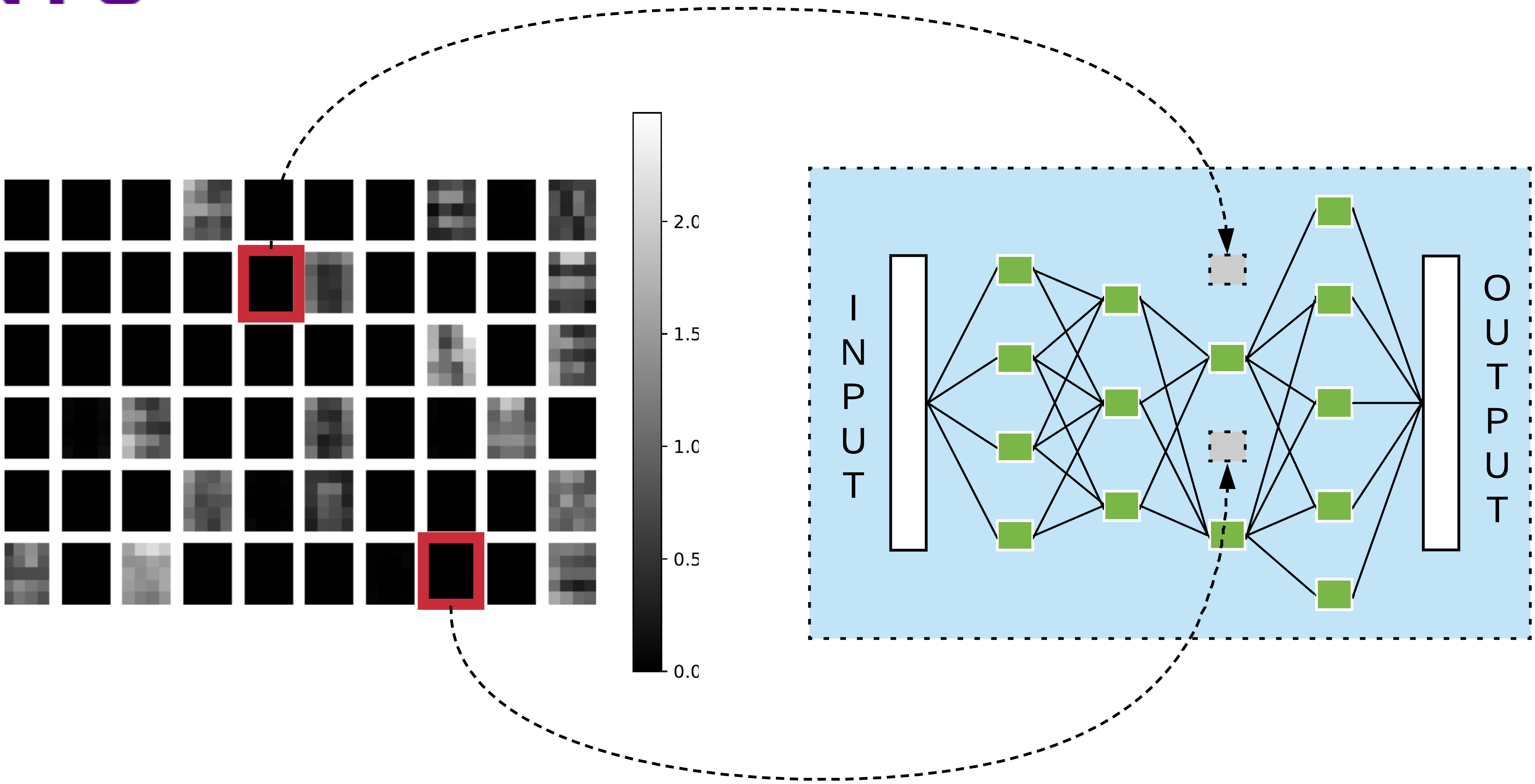


- We found in BadNets that the last layer of a backdoored network contained neurons that were rarely activated on *clean* data
- Can we simply remove these to get rid of backdoors?

Background: Pruning

- DNN models are often *overparameterized* (one can achieve similar accuracy with a smaller model)
- A common way to optimize neural networks is to find neurons that are not activated by validation data and *prune* them (Le Cun et al.'s "Optimal Brain Damage")
- We can *prune* a neuron by reducing the number of channels in a layer's output by one

Defense: Pruning Backdoor Neurons



Evaluating Pruning Effectiveness

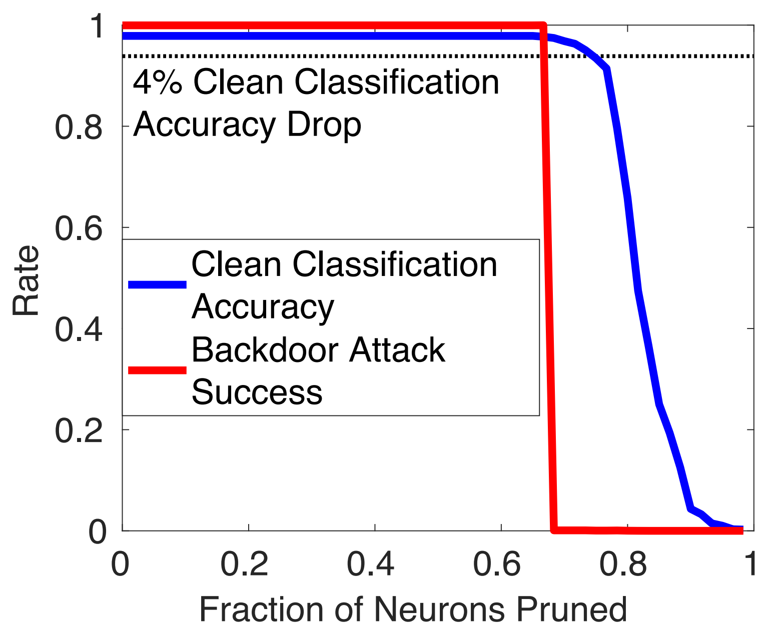
- We want to measure two things:
  - What is the accuracy of the pruned network on **clean** data?
  - What is the **effectiveness** of the backdoor after pruning?
- Note that for untargeted attacks, effectiveness is slightly more complicated to measure:

$$1 - \frac{A_{backdoor}}{A_{clean}}$$

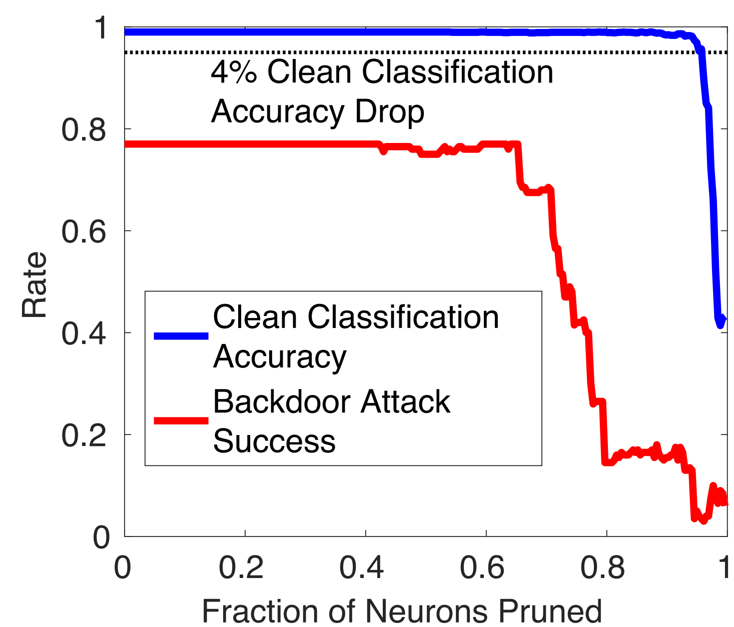
accuracy on backdoored inputs

accuracy on clean inputs

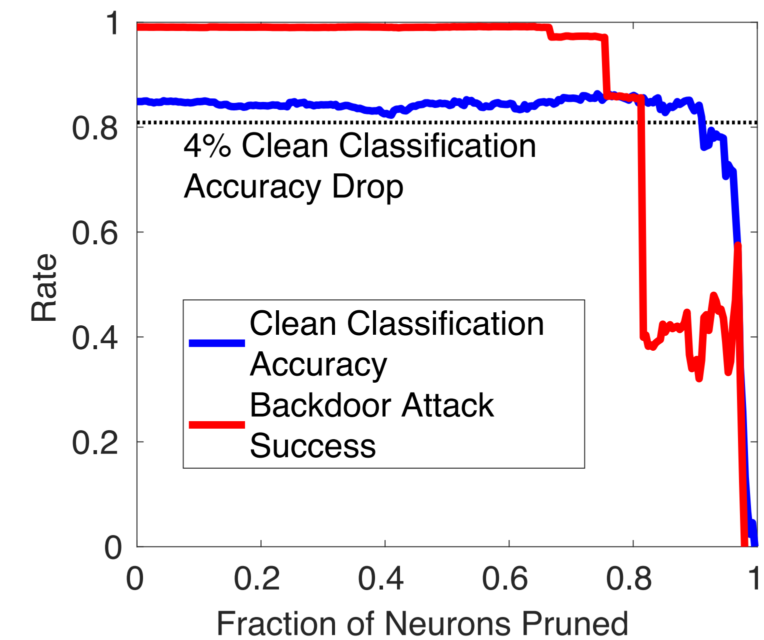
Pruning Defense Evaluation



(a) Baseline Attack (Face)



(c) Baseline Attack (Speech)



(e) Baseline Attack (Traffic)

- Defender doesn't know a priori when backdoor will be removed – instead, stop when *clean* accuracy drops too much (we chose 4%)
- In each case, backdoor is effectively disabled *before* accuracy on clean data suffers much

Thinking Adversarially

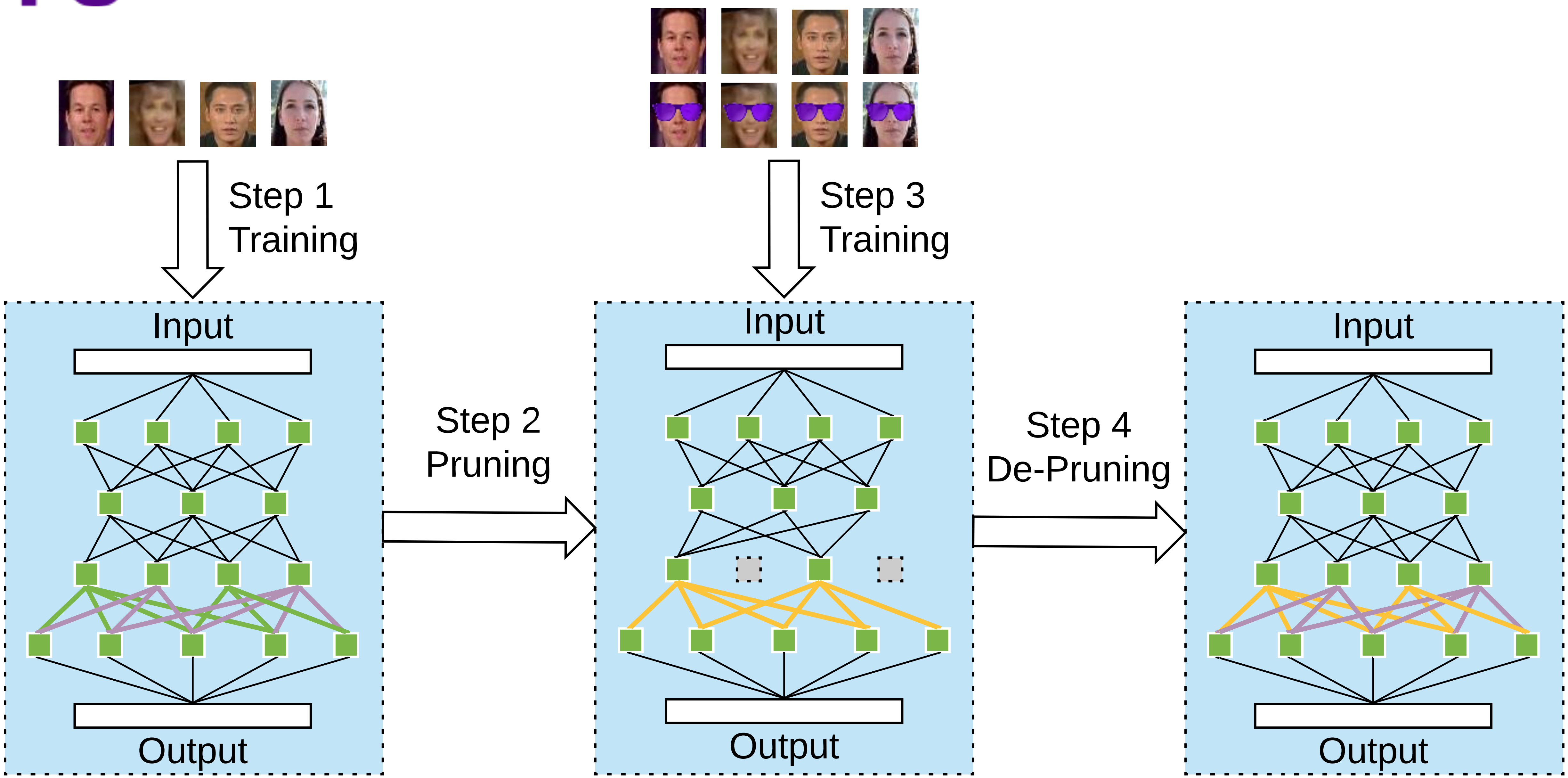
- We may be tempted to stop here – the defense works!
- But it's important to ask: if an attacker knows we will be pruning, can they change their tactics?
- Can an attacker design their backdoor so that it will survive pruning?

Pruning-Aware Attack

- It turns out a more savvy attacker can:
  - Train a clean network
  - *Preemptively* prune their own network
  - Insert the backdoor into the pruned network via poisoning
  - *De-prune* the network by restoring the pruned neurons but decreasing their bias to avoid changing accuracy on clean data

$$a_i = \phi\left(w_i a_{i-1} + b_i\right) \quad \forall i \in [1, L],$$

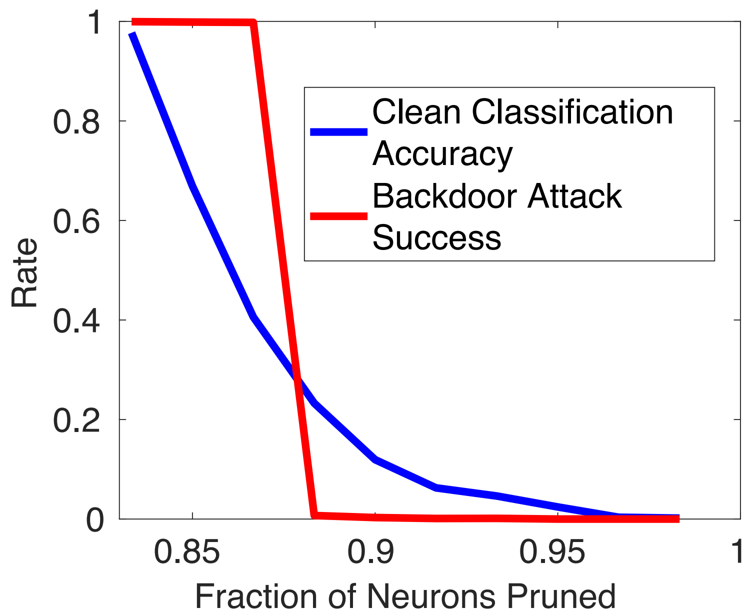
Pruning-Aware Attack



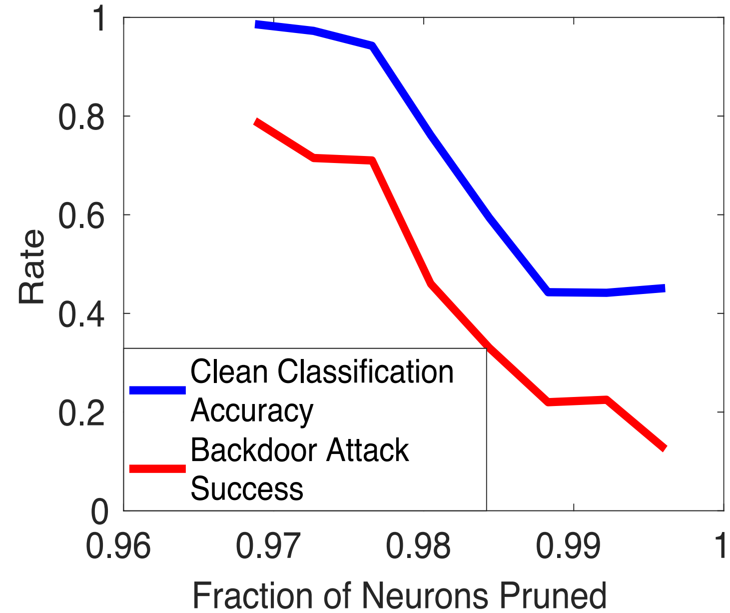
Pruning-Aware Attack

- What is the effect of this attack on the pruning defense?
- When the defender prunes, the neurons that will be removed are precisely those that were removed when the attacker pruned
- These “sacrificial” neurons have no effect on the accuracy of the backdoor since they were not present when the backdoored network was trained

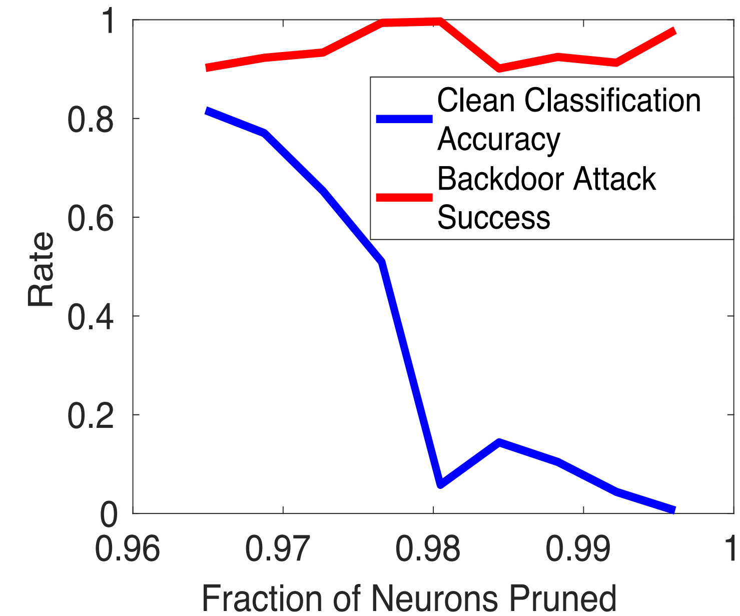
Pruning-Aware Attack Evaluation



(b) Pruning Aware Attack (Face)



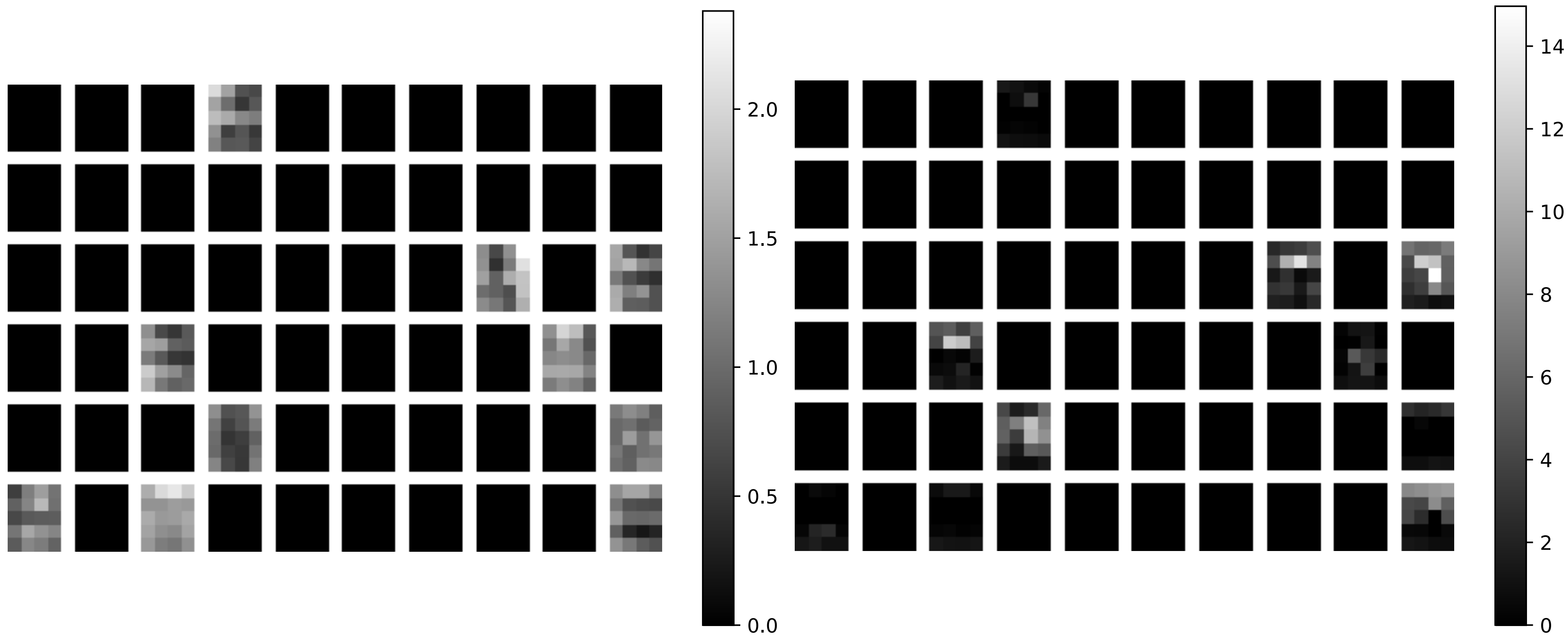
(d) Pruning Aware Attack (Speech)



(f) Pruning Aware Attack (Traffic)

- The *pruning-aware* attack successfully defeats the pruning defense
- Attempting to prune degrades the accuracy on clean data at least as much as backdoored data!

Pruning-Aware Activations



(a) Clean Activations (pruning aware attack)

(b) Backdoor Activations (pruning aware attack)

Result: backdoor activations are a subset of clean activations

Fine-Tuning

- Prior work (Chen et al. 2017) consider another possible defense: *fine-tuning* using a known good set of training data
- Fine-tuning essentially continues the training procedure, starting from the (possibly poisoned) weights
  - Fine-tuning goes much faster though – usually takes just a few minutes to converge
- Is this defense effective?

Fine-Tuning Fails!

- Chen et al. found that fine-tuning did *not* prevent backdoor attacks from being successful
  - We confirmed this result on our three backdoor case studies as well
- Why?
  - Gradient-based training procedure relies on updating the weights of neurons that contribute to misclassifications
  - Clean data rarely activates backdoor neurons – so backdoor neurons are often untouched

Fine-Pruning

- Our final *fine-pruning* defense combines these two defenses which individually fail

- The defender first *prunes* inactive neurons and then *fine-tunes* on held-out data

- Intuition: pruning means backdoor can only be in one of a small number of neurons – so fine-tuning can then act on those

Fine-Pruning Defense Evaluation

Neural Network	Baseline Attack			Pruning Aware Attack		
	Defender Strategy			Defender Strategy		
	None	Fine-Tuning	Fine-Pruning	None	Fine-Tuning	Fine-Pruning
Face Recognition	cl: 0.978 bd: 1.000	cl: 0.978 bd: 0.000	cl: 0.978 bd: 0.000	cl: 0.974 bd: 0.998	cl: 0.978 bd: 0.000	cl: 0.977 bd: 0.000
Speech Recognition	cl: 0.990 bd: 0.770	cl: 0.990 bd: 0.435	cl: 0.988 bd: 0.020	cl: 0.988 bd: 0.780	cl: 0.988 bd: 0.520	cl: 0.986 bd: 0.000
Traffic Sign Detection	cl: 0.849 bd: 0.991	cl: 0.857 bd: 0.921	cl: 0.873 bd: 0.288	cl: 0.820 bd: 0.899	cl: 0.872 bd: 0.419	cl: 0.874 bd: 0.366

- Attacker success is reduced to ~0% in targeted case, and 29%-37% in untargeted case
  - Recall that attacker's job is easier in untargeted case – any misclassification counts toward success!

What Defense to Use?

- We saw that fine-tuning and fine-pruning both work well against "sophisticated" pruning-aware attacker
- So why is fine-pruning still superior?
  - If attacker knows *fine-tuning* will be used, they can switch back to *baseline* attack

Utility		Attacker Strategy	
		Baseline Attack	Pruning Aware Attack
Defender Strategy	Fine-Tuning	0.555	0.468
	Fine-Pruning	0.968	0.986

Defender Utility = clean accuracy - backdoor success

Limitations

- Fine-pruning requires defender to do some **retraining** which is expensive (though much less than starting from scratch)
- Examples so far use CNNs – may not generalize to LSTM, RNN, etc.
- We still don't have any **theoretical** guarantees that this defense is effective in all cases
    - Although we know that *some* amount of fine-tuning + perturbation must be sufficient

Conclusions

- Model capacity is strongly related to susceptibility to backdoor attacks
- Unlike traditional software, we *can* find and remove backdoors automatically!
- Still quite a lot we don't understand!
  - More research is needed™