One-slide Abstracts

Research of Jacob M. Springer 2017–2022

2017

Teaching with `angr`: A Symbolic Execution Curriculum and CTF*

Jacob M. Springer *Wu-chang Feng*
Portland State University
Department of Computer Science

```c
int main(int argc, char* argv[]) {
   char buf[9];

   printf("Enter the password: ");
   scanf("%8s", buf);

  for (int i=0; i<LEN_USERDEF; ++i)
     buf[i] = complex_function(buf[i], i)
    ;

  if (strcmp(buf, USERDEF))
     printf("Try again.\n");
  else
     printf("Good Job.\n");
}
```

Figure 3: C code for first CTF level

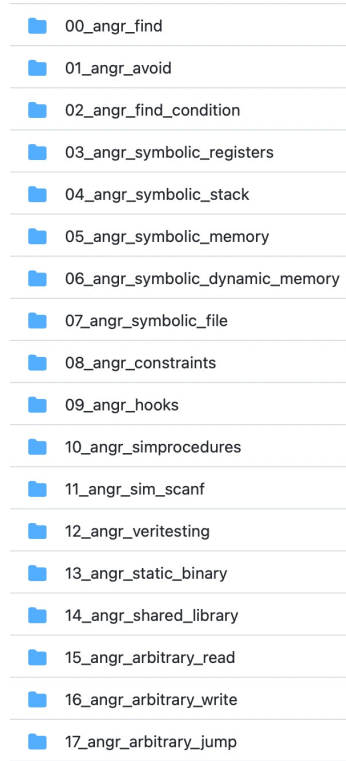
```python
import angr
import sys

def main(argv):
  path_to_binary = argv[1]
  project = angr.Project(path_to_binary)
  initial_state = project.factory.entry_state()
  simulation = project.factory.simgr(initial_state)

  print_good_address = 0x804867a
  simulation.explore(find=print_good_address)

  if simulation.found:
    solution_state = simulation.found[0]
    print solution_state.posix.dumps(sys.stdin.fileno())
  else:
    raise Exception('Could not find the solution')
```

Figure 5: `angr` solution script for first CTF level



Question
Q1: Rate the lecture material for understanding the concepts behind symbolic execution.
Q2: Rate the CTF exercises for understanding the concepts behind symbolic execution.
Q3: Rate the CTF exercises for developing skills in using symbolic execution techniques.
Q4: Helpfulness of the CTF format compared to other homework formats used in our curriculum

Table 2: Survey questions for the `angr` CTF and curriculum in CS 492/592: Malware (Winter 2018)

Question	1	2	3	4	5	Mean rating
Q1	1	1	2	17	12	4.15
Q2	2	1	3	18	9	3.94
Q3	1	3	3	16	10	3.94
Q4	1	3	12	12	5	3.52

Table 3: Quality and Usefulness of Symbolic Execution module (1=Very unhelpful, 2=Somewhat unhelpful, 3=neither helpful nor unhelpful, 4=somewhat helpful, 5=Very helpful)

Completion percentage	No. of students
95-100%	25
85-95%	4
75-85%	6
Below 75%	7

Table 1: Level completion results

2018

Classifiers Based on Deep Sparse Coding Architectures are Robust to Deep Learning Transferable Examples

Jacob M. Springer[1,2], Charles S. Strauss[3], Austin M. Thresher[3], Edward Kim[4],
Garrett T. Kenyon[1]

Objective $\min_{\Phi} \sum_{i=1}^{n} \min_{\mathbf{a}_i} \frac{1}{2} \|\mathbf{x}_i - \Phi \mathbf{a}_i\|_2^2 + \lambda \|\mathbf{a}_i\|_0$

Solution $\dot{\mathbf{u}} = -\mathbf{u} + \Phi^T \mathbf{x} - [\Phi^T \Phi \mathbf{a} - \mathbf{a}]$

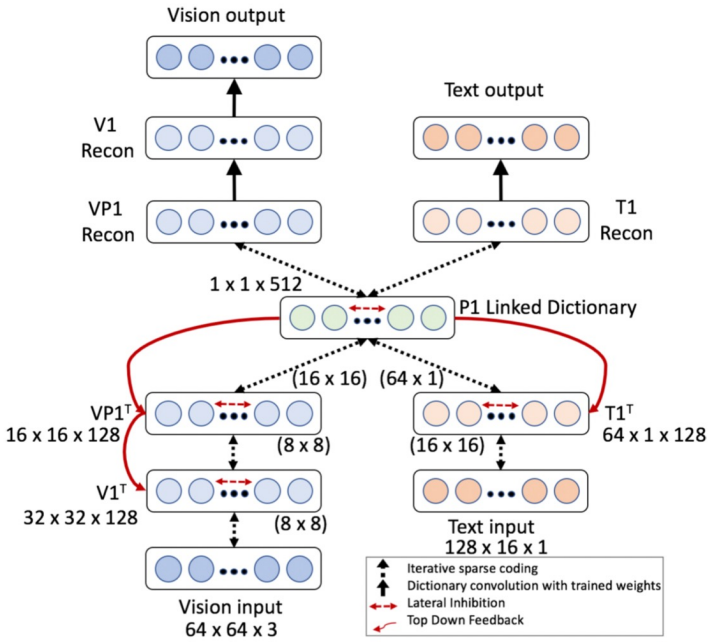


Figure 2. Deep sparse coding (DSC) model from Kim *et al.* [6]

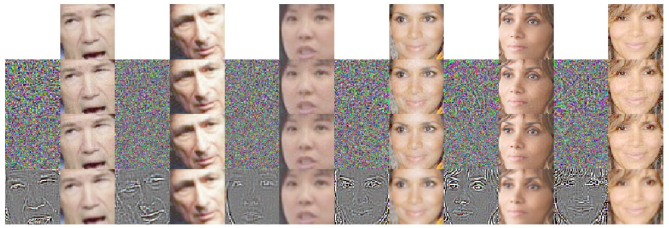


Figure 1. Images from the holdout dataset and the corresponding alterations. In each row, the left three faces are of "other" (non-Halle Berry), and the right three faces are of Halle Berry. *From top to bottom*: original, adversarial, Gaussian noise, low-pass filter. The noise to the left of each non-original image shows the difference between that image and the corresponding original. The noise has been translated so that a pixel with no change is gray with a value 0.5. The noise is scaled by a factor of 10 in order to increase visibility. While upon close inspection the alternations are visible, each altered image is nearly indistinguishable identical to the original image.

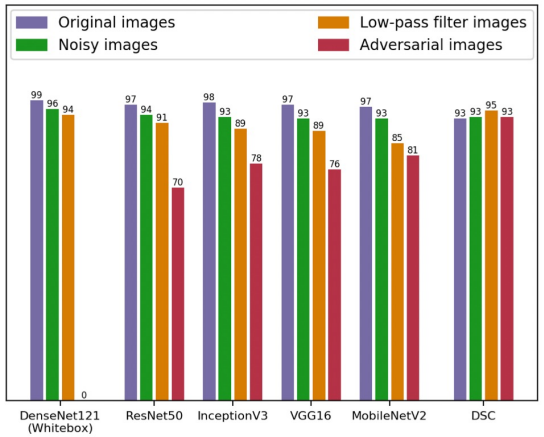


Figure 5. Accuracy scores of the various models on the *in blue:* original images, *in green:* Gaussian-noisy images, *in orange:* low-pass filtered images, and *in red:* adversarial images. Note that we use the DenseNet121 model to generate the adversarial examples, which is why its classification accuracy on the adversarial examples is zero. We see that the performance of all DCN models drops significantly on adversarial examples whereas the performance of the DSC model is invariant to the described minor alterations.

	ResNet	Inception	VGG	MobileNet	**DSC**
δ_α	0.44	0.08	0.59	0.68	**0.05**
δ_η	0.34	0.08	0.42	0.69	**0.04**
δ_ℓ	0.39	0.13	0.44	0.46	**0.04**

Table 1. Distance metric between hidden-layer representation of original images and adversarial images (δ_α), Gaussian noisy images (δ_η), and low-pass filter images (δ_ℓ). The exact metric is described below. Deep sparse coding representations of adversarial, noisy, and low-pass filter images are significantly more similar than the corresponding representations by DCN models.

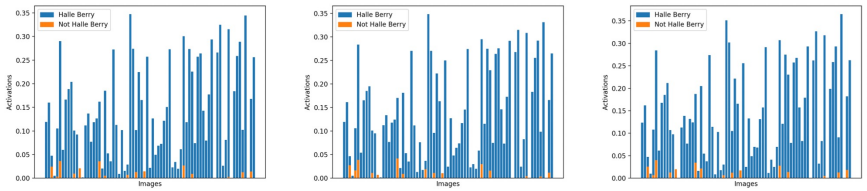


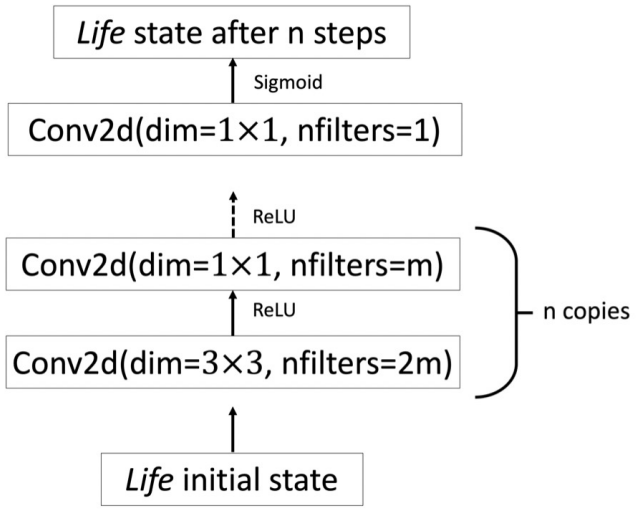
Figure 6. The activation strength of the *N-326* neuron on each holdout image, compared across the variants of the datasets. The left graph shows the *N-326* activation on each *in blue:* original image of Halle Berry and *in orange:* of people other than Halle Berry. The center and right graphs show the same, but on the adversarial copies and on the low-pass filter copies, respectively. *N-326* activates strongly on images of Halle Berry and weakly on images of other. The *N-326* activation remains invariant across adversarial perturbations and low-pass filtering.

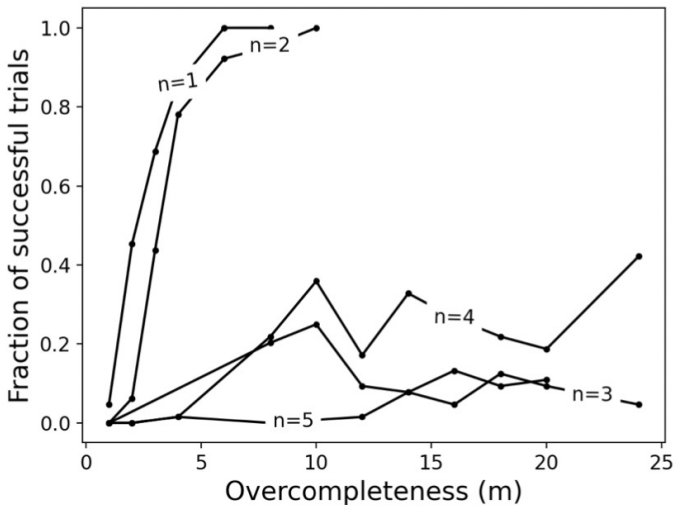
2019

It's Hard for Neural Networks to Learn the Game of Life

1st Jacob M. Springer
Swarthmore College
Swarthmore, PA, USA
jacmspringer@gmail.com

2nd Garrett T. Kenyon
Los Alamos National Laboratory
Los Alamos, NM, USA
gkenyon@lanl.gov





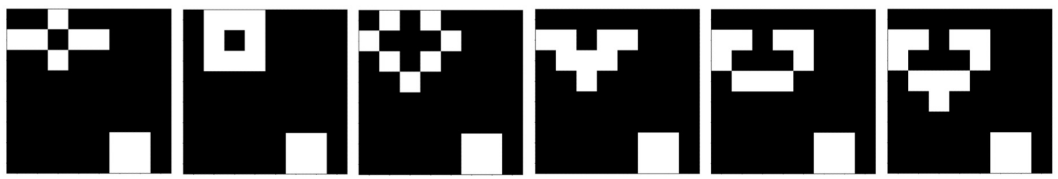
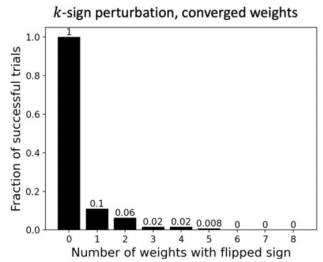
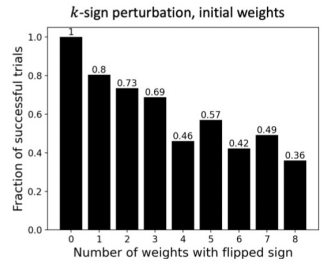
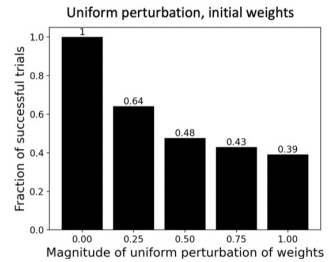
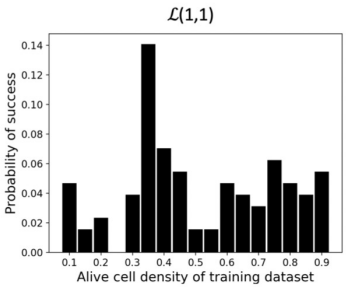


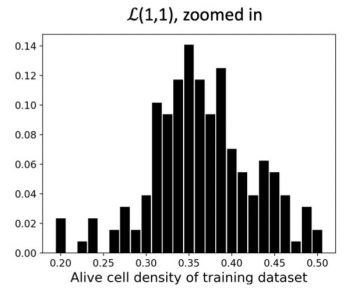
Fig. 1. An example of an 8 × 8 cell board of Life over six time steps, evolving in time from left to right. White pixels are considered alive and black pixels are considered dead.

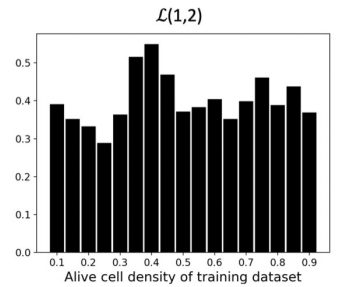












2019

Sparse MP4

1st Daniel A. Wang
Ultra-Scale Research Center
Los Alamos National Laboratory
Los Alamos, USA
wang_a_daniel@lanl.gov

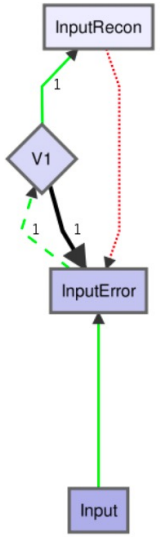
2nd Charles M. S. Strauss
New Mexico Consortium
Los Alamos, USA
charles.s.strauss@gmail.com

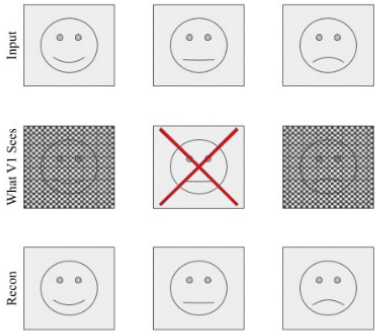
3rd Jacob M. Springer
Swarthmore College
Swarthmore, USA
jspring1@swarthmore.edu

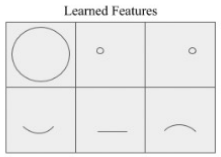
4nd Austin Thresher
Advanced Research in Cyber Systems
Los Alamos National Laboratory
Los Alamos, USA
athresher@lanl.gov

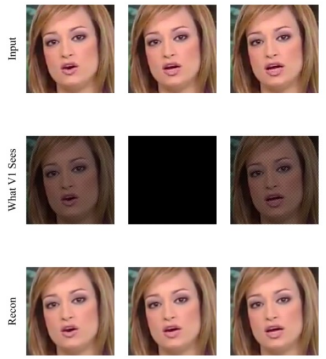
5rd Howard Pritchard
Ultra-Scale Research Center
Los Alamos National Laboratory
Los Alamos, USA
hpritchard@lanl.gov

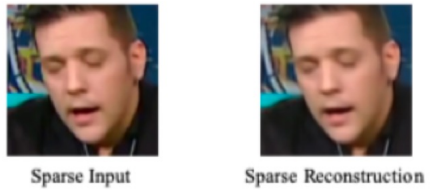
6th Garrett T. Kenyon
Computation, Computing and Statistical Sciences
Los Alamos National Laboratory
Los Alamos, USA
gkenyon@lanl.gov











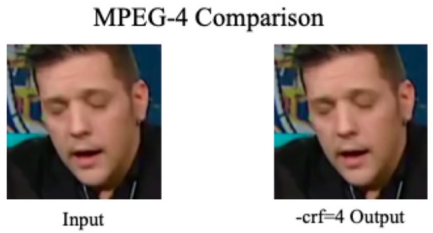


TABLE I

PSNR Values - Scale 0-100			
Methods	Frame 1	Frame 2	Frame 3
Sparse Coding	79.0245	77.0338	78.6417
Bottleneck Autoencoder	31.4293	32.4929	32.0486
	-crf = 3	-crf = 4	
MP4	45.9641	45.4548	

TABLE II

SSIM Values - Scale 0-1			
Methods	Frame 1	Frame 2	Frame 3
Sparse Coding	0.9923	0.9841	0.9929
Bottleneck Autoencoder	0.9430	0.9472	0.9429
	-crf = 3	-crf = 4	
MP4	0.9930	0.9923	

2020

STRATA: Simple, Gradient-Free Attacks for Models of Code

Jacob M. Springer*
jacmspringer@gmail.com
MIT
Cambridge, MA, USA

Bryn Marie Reinstadler*
brynr@mit.edu
MIT
Cambridge, MA, USA

Una-May O'Reilly
unamay@csail.mit.edu
MIT
Cambridge, MA, USA

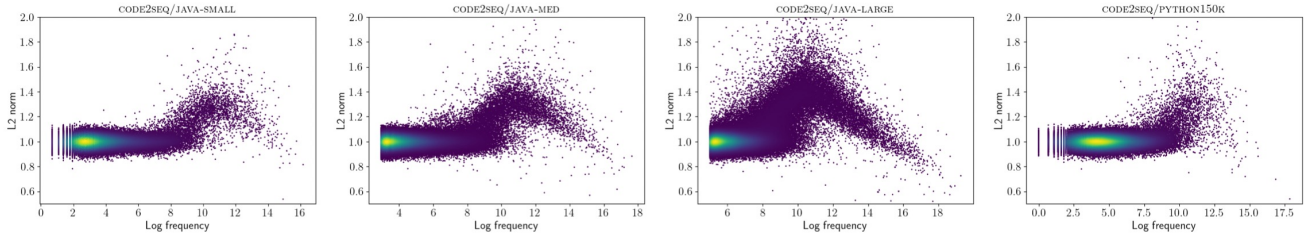


Figure 1: L2 norm of trained token embeddings by frequency of token in training dataset. Color represents the approximate density of tokens for a given frequency and embedding L2 norm. A lighter color corresponds with a higher density.

Strategy: replace tokens in variable names with adversarial tokens

	Baseline	All			Top-n by L2 norm			Top-n by Frequency		
		single	*5-diff*	*5-same*	*single*	*5-diff*	*5-same*	*single*	*5-diff*	*5-same*
CODE2SEQ/JAVA-SMALL	.369	.381	.350	.310	.362	.263	**.214**	.372	.284	**.231**
CODE2SEQ/JAVA-MED	.564	.548	.531	.492	.513	.416	**.375**	.513	.385	**.345**
CODE2SEQ/JAVA-LARGE	.608	.536	.547	.488	.542	.396	**.360**	.548	.427	**.388**
CODE2SEQ/PYTHON150K	.313	.274	.250	.209	.249	.198	**.153**	.256	.211	**.172**

Table 2: Effectiveness of targeted attacks on CODE2SEQ. All attacks employ the *5-same* replacement strategy.

Model	Perturbation	% success
CODE2SEQ/JAVA-LARGE	L2, top 6k	**37.1**
CODE2SEQ/JAVA-LARGE	Freq, top 10k	35.6
CODE2SEQ/JAVA-LARGE	All	3.9
CODE2SEQ/JAVA-MED	L2, top 3k	**43.8**
CODE2SEQ/JAVA-MED	Freq, top 3k	39.1
CODE2SEQ/JAVA-MED	All	1.4
CODE2SEQ/JAVA-SMALL	L2, top 1k	**58.7**
CODE2SEQ/JAVA-SMALL	Freq, top 1.8k	52.8
CODE2SEQ/JAVA-SMALL	All	2.1

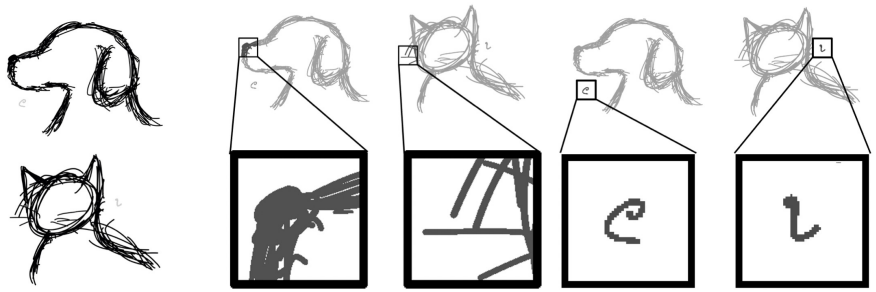
Table 3: Comparison of F1 scores our method to gradient-based attacks described by Ramakrishnan et al. [31]. Both our model and the Ramakrishnan et al. [31] model are trained on JAVA-SMALL. We perform *5-same* attacks.

Adversarial attack	F1	% of baseline
Ramakrishnan et al. [31] attacks		
Baseline	.414	100
Variable Replacement	.389	93.9
Try-catch Statement Insertion	.336	81.2
Print Statement Insertion	.312	75.3
Worst-case Transformation	.246	59.4
STRATA		
Baseline	.425	100
Top 1k by L2	**.316**	**74.4**
Top 1.8k by frequency	.328	77.2

2020–2021

Adversarial Perturbations Are Not So Weird: Entanglement of Robust and Non-Robust Features in Neural Network Classifiers

Jacob M. Springer [1] **Melanie Mitchell** [2] **Garrett T. Kenyon** [1]



Type A: Robust features. These features commonly respond to human-interpretable ("semantic") patterns (here of a dog or a cat).

Type B: Non-robust features that respond to small yet highly predictive patterns that by themselves appear non-semantic, yet are entangled with patterns associated with robust features (e.g., in (a)).

Type C: Non-robust features that respond to highly predictive patterns that are artifacts in the dataset, and are independent of robust features.

Table 1. Accuracy of zero-robust, negative-robust, standard, and robust ResNet50 classifiers on robustified CIFAR-10 test set. All classifiers are trained starting with random initial weights. The *Test* column is the accuracy on the original CIFAR-10 test set. The $\mathcal{R}_\varepsilon$ columns give accuracy on robustified CIFAR-10 test set, generated with respect to a robust ResNet50 with robustness parameter ε.

Classifier	Test	R_0	$\mathcal{R}_{1/16}$	$\mathcal{R}_{1/4}$	$\mathcal{R}_{1/2}$	$\mathcal{R}_1$
Standard	94.1	58.3	79.9	75.1	69.2	56.3
Robust	86.5	9.93	18.2	70.5	86.5	70.0
Zero-robust	46.8	38.8	23.3	22.1	23.4	21.4
Neg-robust	21.9	31.8	11.1	12.8	14.3	13.2

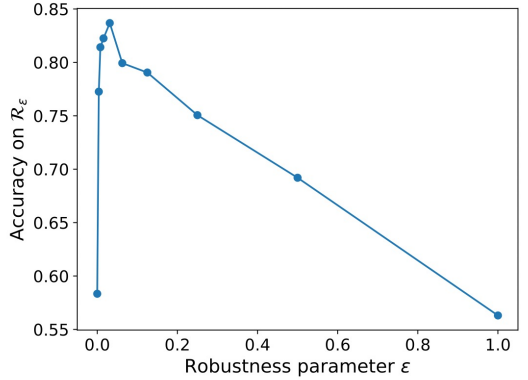
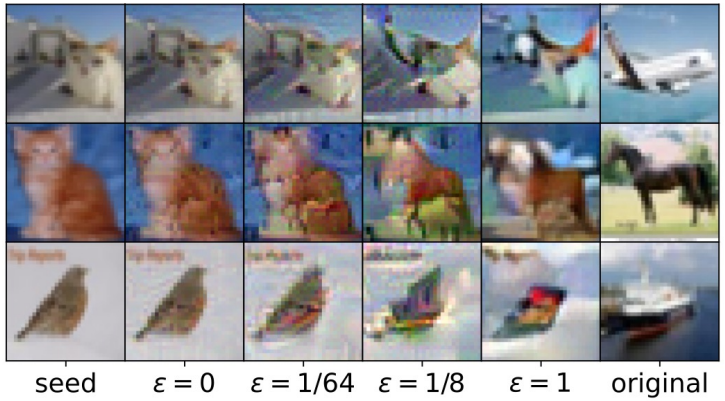


Figure 3. Accuracy of a standard ResNet50 classifier on robustifications of the CIFAR-10 test dataset, generated with respect to robust ResNet50 classifiers with varying robustness parameter ε. See Appendix for the analogous graph for ImageNet-9. (Best viewed in color.)

Robustification



Can train a neural network to ONLY look for non-robust features

Can construct a dataset that ONLY has robust features

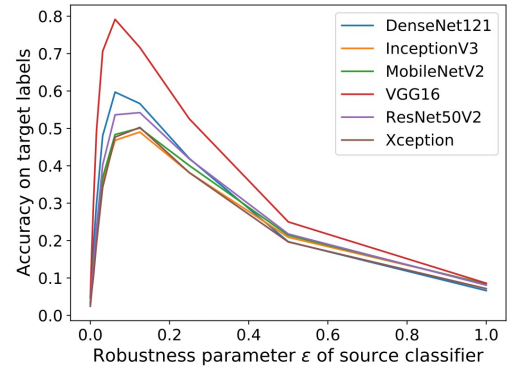
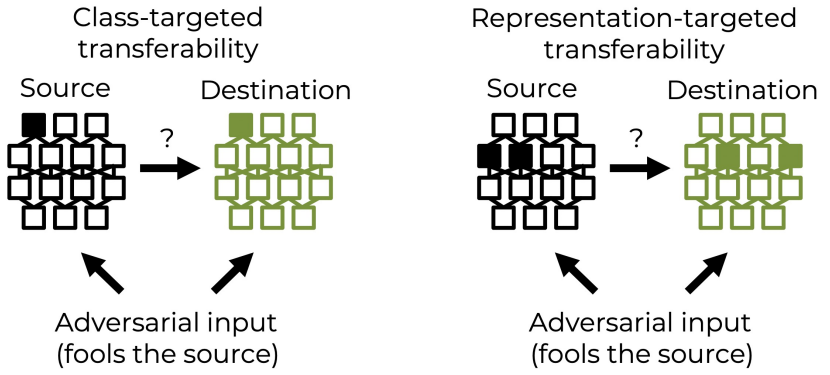


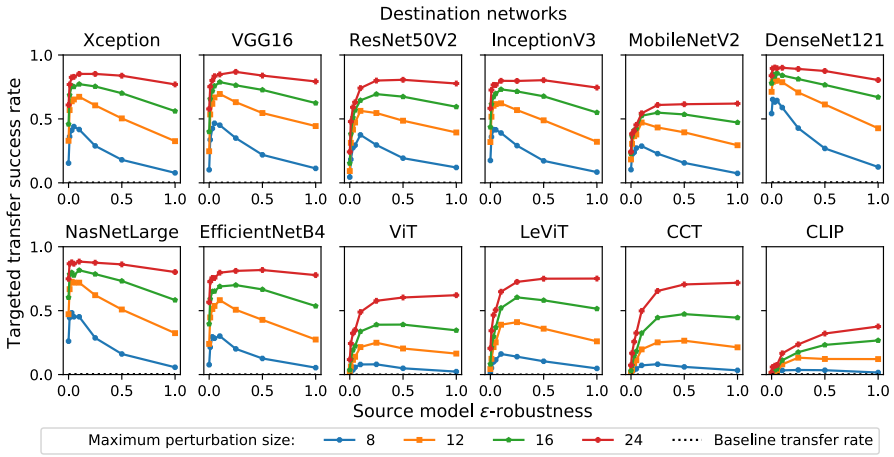
Figure 6. Accuracy of different standard classifiers on targeted transfer attacks.

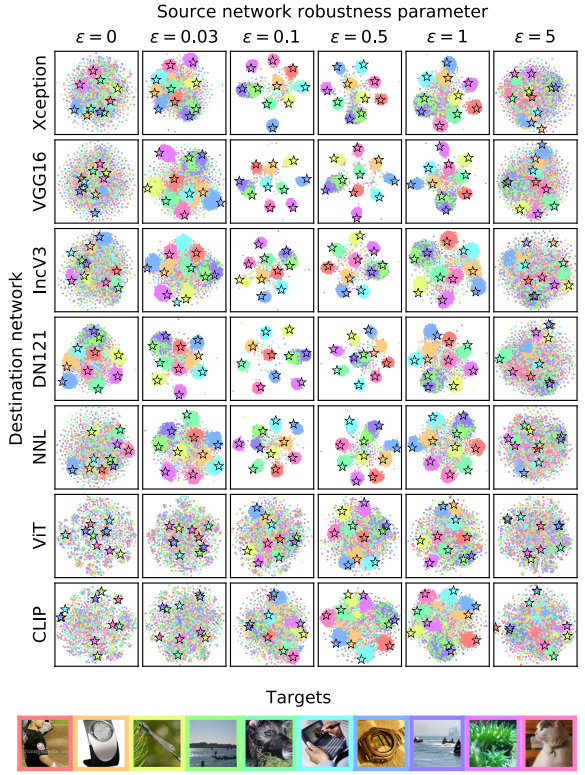
2020–2021

A Little Robustness Goes a Long Way: Leveraging Robust Features for Targeted Transfer Attacks

Jacob M. Springer **Melanie Mitchell** **Garrett T. Kenyon**







Destination	Source network robustness parameter (ε) 0	0.01	0.03	0.05	0.1	0.25	0.5	1	3	5
Xception	0.462	0.505	0.531	0.563	**0.594**	0.585	0.572	0.543	0.449	0.404
VGG16	0.333	0.401	0.417	0.494	**0.528**	0.520	0.520	0.486	0.383	0.333
ResNet50V2	0.284	0.348	0.379	0.432	0.497	0.496	**0.510**	0.484	0.380	0.321
InceptionV3	0.577	0.612	0.627	0.644	**0.673**	0.662	0.655	0.636	0.572	0.539
MobileNetV2	0.431	0.459	0.460	0.493	**0.517**	0.513	0.513	0.504	0.455	0.425
DenseNet121	0.672	0.689	0.685	0.713	**0.726**	0.714	0.706	0.679	0.616	0.584
NasNetLarge	0.356	0.422	0.452	0.488	**0.541**	0.513	0.482	0.437	0.315	0.271
EfficientNetB4	0.085	0.111	0.137	0.144	**0.237**	0.220	0.226	0.202	0.112	0.074
ViT	0.066	0.087	0.109	0.129	0.195	**0.206**	**0.206**	0.203	0.120	0.086
CLIP	0.529	0.541	0.550	0.563	0.585	0.599	0.606	**0.613**	0.581	0.566

Cosine similarity of representation layer activations of adversarial example and target, in destination networks