Notice

- Due dates
  - Written paper critiques (on 01.10)
  - Homework 1 (on 01.10)
- Sign-up (on Canvas)
  - Scribe Lecture Note
  - In-class Paper Presentation / Discussion
  - Term Project (by 01.19)
- Discord Server is open
  - Email me if you haven't received the invitation link
- Office Hours
  - Share the Zoom link on Canvas

Notice – cont’d

- Grading Scheme (Max. 140 pts = Base: 120 pts + Extra: 20 pts)
  - A: 102 <= Total Score <= 140
  - B: 90 <= Total Score < 102
  - C: 78 <= Total Score < 90
  - D: 60 <= Total Score < 78
  - F: Total Score < 60
  - **Note:** Graduate students will have a slightly higher bar, **+6 pts** to lower bounds
    ex. A (Grad) 108 <= Total Score < 140

CS 499/599: Machine Learning Security
01.05: Adversarial Examples (AE) 1

Mon/Wed 12:00 – 1:50 pm

Instructor: **Sanghyun Hong**

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL
Secure AI Systems Lab

Topics for Today

- Motivation
- AE←Security
  - Motivation
  - Threat Model
  - Attack: Gradient Descent
  - Practical Exploitation
  - Conclusions & Implications
- AE←ML
  - Motivation
  - Counter-intuitive (CI) Prop. 1
  - CI Prop. 2
  - Conclusions & Implications

Motivation: ML Really Matters

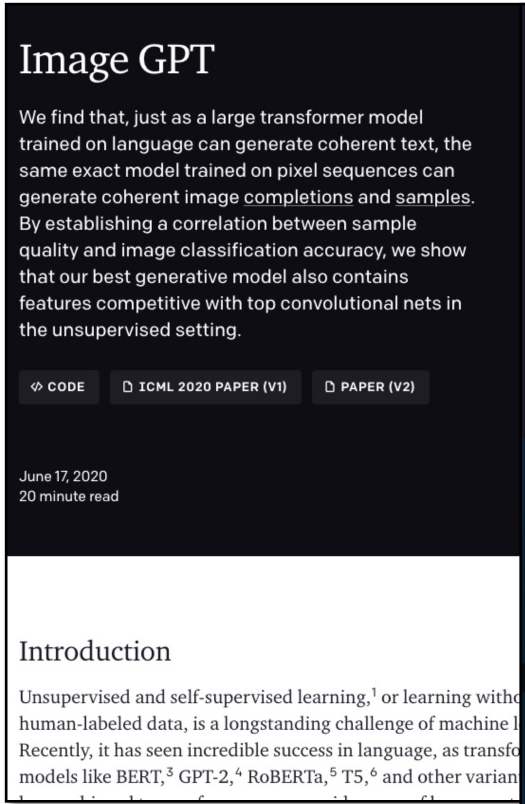


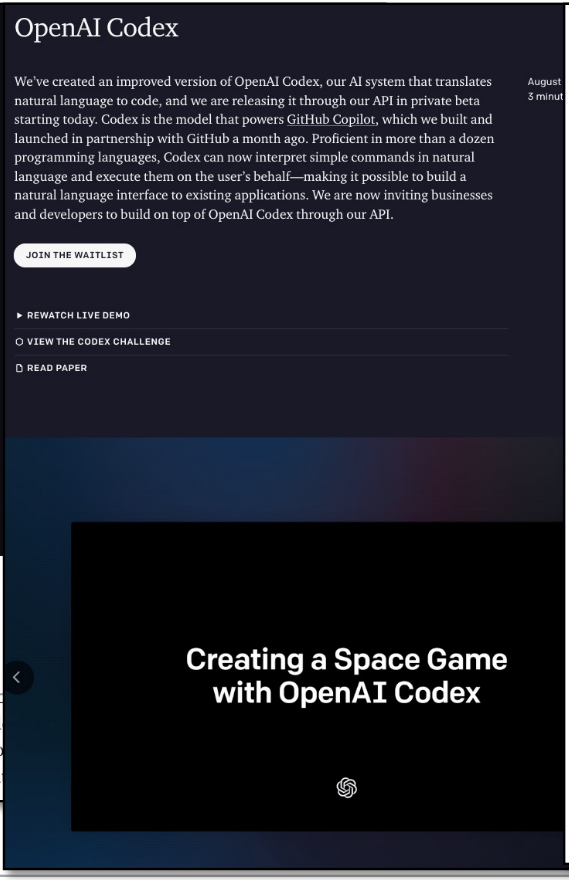
Image GPT

We find that, just as a large transformer model trained on language can generate coherent text, the same exact model trained on pixel sequences can generate coherent image completions and samples. By establishing a correlation between sample quality and image classification accuracy, we show that our best generative model also contains features competitive with top convolutional nets in the unsupervised setting.

June 17, 2020
20 minute read

Introduction

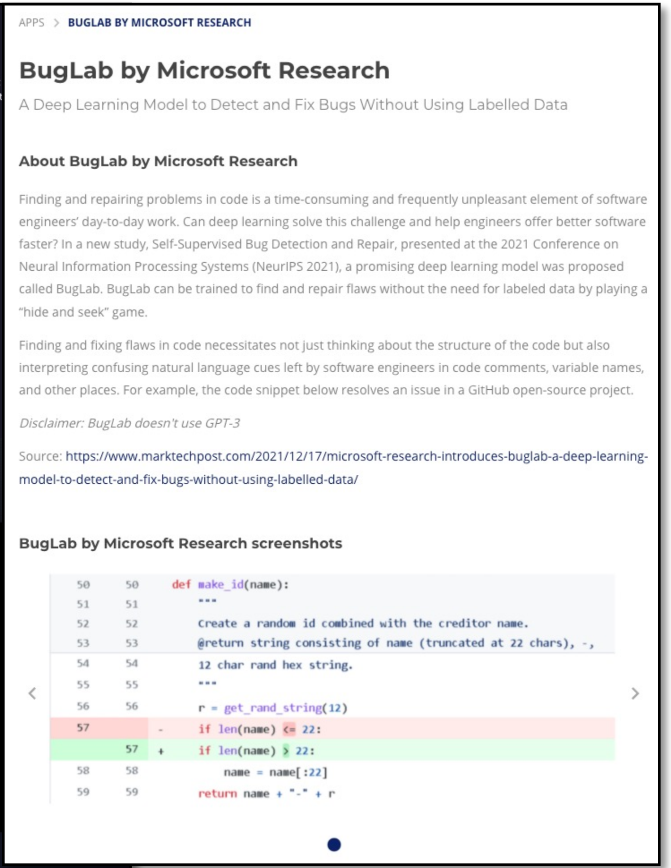
Unsupervised and self-supervised learning,[1] or learning witho
human-labeled data, is a longstanding challenge of machine l
Recently, it has seen incredible success in language, as transfo
models like BERT,[3] GPT-2,[4] RoBERTa,[5] T5,[6] and other varian



OpenAI Codex

We've created an improved version of OpenAI Codex, our AI system that translates natural language to code, and we are releasing it through our API in private beta starting today. Codex is the model that powers GitHub Copilot, which we built and launched in partnership with GitHub a month ago. Proficient in more than a dozen programming languages, Codex can now interpret simple commands in natural language and execute them on the user's behalf—making it possible to build a natural language interface to existing applications. We are now inviting businesses and developers to build on top of OpenAI Codex through our API.

August
3 minut



APPS > BUGLAB BY MICROSOFT RESEARCH

BugLab by Microsoft Research

A Deep Learning Model to Detect and Fix Bugs Without Using Labelled Data

About BugLab by Microsoft Research

Finding and repairing problems in code is a time-consuming and frequently unpleasant element of software engineers' day-to-day work. Can deep learning solve this challenge and help engineers offer better software faster? In a new study, Self-Supervised Bug Detection and Repair, presented at the 2021 Conference on Neural Information Processing Systems (NeurIPS 2021), a promising deep learning model was proposed called BugLab. BugLab can be trained to find and repair flaws without the need for labeled data by playing a "hide and seek" game.

Finding and fixing flaws in code necessitates not just thinking about the structure of the code but also interpreting confusing natural language cues left by software engineers in code comments, variable names, and other places. For example, the code snippet below resolves an issue in a GitHub open-source project.

Disclaimer: BugLab doesn't use GPT-3

Source: https://www.marktechpost.com/2021/12/17/microsoft-research-introduces-buglab-a-deep-learning-model-to-detect-and-fix-bugs-without-using-labelled-data/

BugLab by Microsoft Research screenshots

```
    def make_id(name):
        """
        Create a random id combined with the creditor name.
        @return string consisting of name (truncated at 22 chars), -,
        12 char rand hex string.
        """
        r = get_rand_string(12)
-       if len(name) <= 22:
+       if len(name) > 22:
            name = name[:22]
        return name + "-" + r
```

Motivation: Let’s Start from Supervised Learning

- ML has been used in
  - Object recognition
  - Sentiment Analysis
  - Spam Filters
  - Malware Detection
  - Network Intrusion Detection
  - Many more…

What Would Be the Failure Modes of Those ML Models?

Biggio et al., Evasion Attacks against Machine Learning Models at Test Time

: This work approaches the problem from a security perspective

Motivation

- Research Questions
  - RQ1: How can we find the failure modes of ML?
  - RQ2: How can we remove (or prevent) such failure modes from ML?

Motivation – cont'd

- Research Questions
  - RQ1: How can we find the failure modes of ML?
    - Handcraft clean (or benign) test-time samples
    - Game-theoretic approach: minimax or Nash-equilibrium
  - RQ2: How can we remove (or prevent) such failure modes from ML?
    - Classifiers obtained from the game-theoretic approaches

Motivation – cont'd

- [Refined] Research Questions
  - RQ1: What are evasion attack **scenarios** (what's the threat model)?
  - RQ2: How can we find the failure modes of ML, **systematically and efficiently**?
  - RQ3: How can an attacker **exploit such failure modes in practice**?
  - RQ4: How can we **mitigate** such threats?

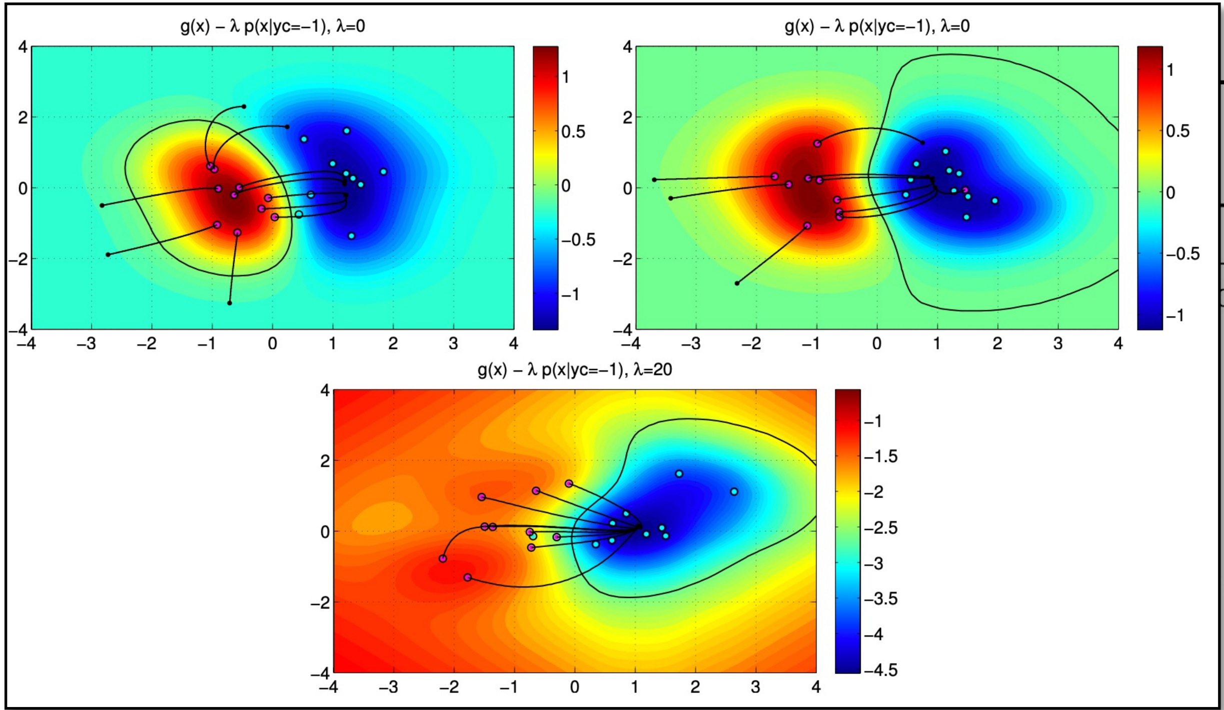
RQ 1: Threat Model

- Evasion Attacks
  - **Goal:** manipulate a test-time sample to be misclassified by a model
  - **Knowledge:**
    - [PK] (Partial) knowledge of the training data
    - [PK | LK] Feature knowledge
    - [PK] Learning algorithm
    - [PK] A model (classifier) parameters
    - PK: Perfect Knowledge (**White-box**)
    - LK: Limited Knowledge (**Black-box**)
  - **Capability:**
    - Perturb an input sample (limited or unlimited)
    - Perturb features (limited or unlimited)

RQ 2: Way to Find Failure Modes

- Goal of the Attack

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \quad \hat{g}(\mathbf{x})$$
$$\text{s.t.} \quad d(\mathbf{x}, \mathbf{x}^0) \leq d_{\text{max}}.$$



de-off parameter; $\epsilon > 0$ a

$- \lambda \nabla p(\mathbf{x}^{m-1}|y^c = -1).$

Mimicry Component

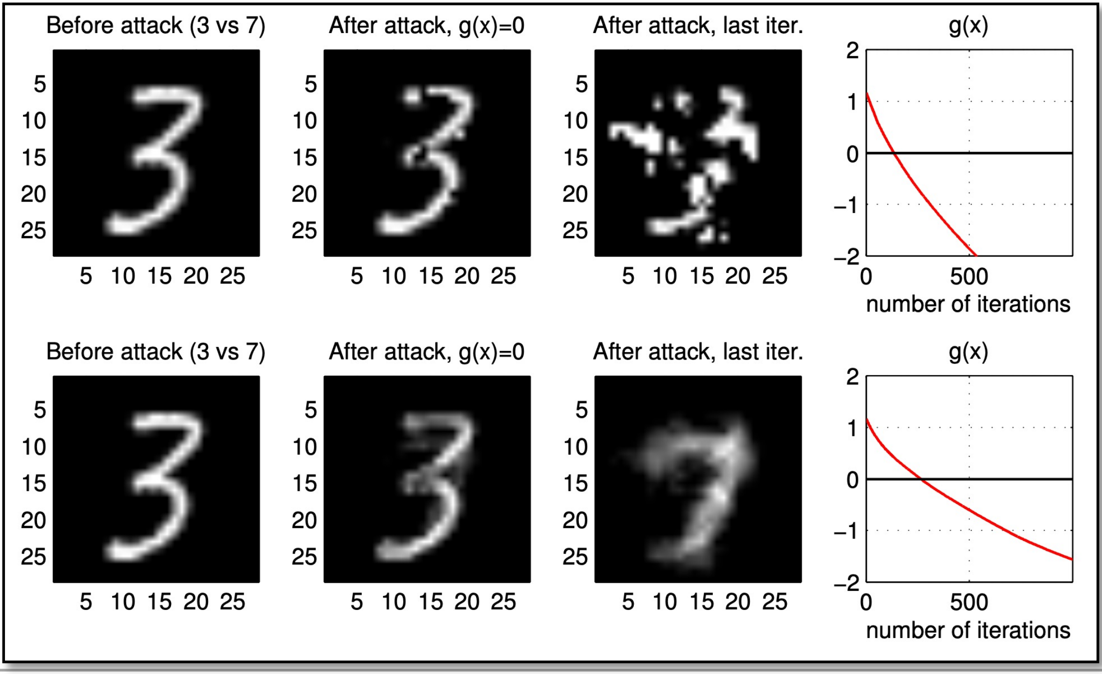
```
7:          Project $\mathbf{x}^m$ onto the boundary of the feasible region.
8:      end if
9:  until $F(\mathbf{x}^m) - F(\mathbf{x}^{m-1}) < \epsilon$
10: return: $\mathbf{x}^* = \mathbf{x}^m$
```

RQ 3: Exploitation in Practice

- 2 Tasks (MNIST-3/7 and PFD Malware Detection)
  - **(Toy example)** MNIST-3/7
    - **Task:** Binary classification problem 3 vs. 7
    - **Attacker:**
      - PK (white-box)
      - Limited: bound the perturbations to $||x' - x||_{l_1} \leq 5000/255$
    - **Model:** SVM (w. $C = 1$)
    - **Targets:** 100 randomly chosen training samples

RQ 3: Exploitation in Practice

- MNIST-3/7 Results
  - (I assume) The crafted samples cause misclassifications on the SVM classifier
  - Mimicry component results in a visually-nice samples (why it's important?)

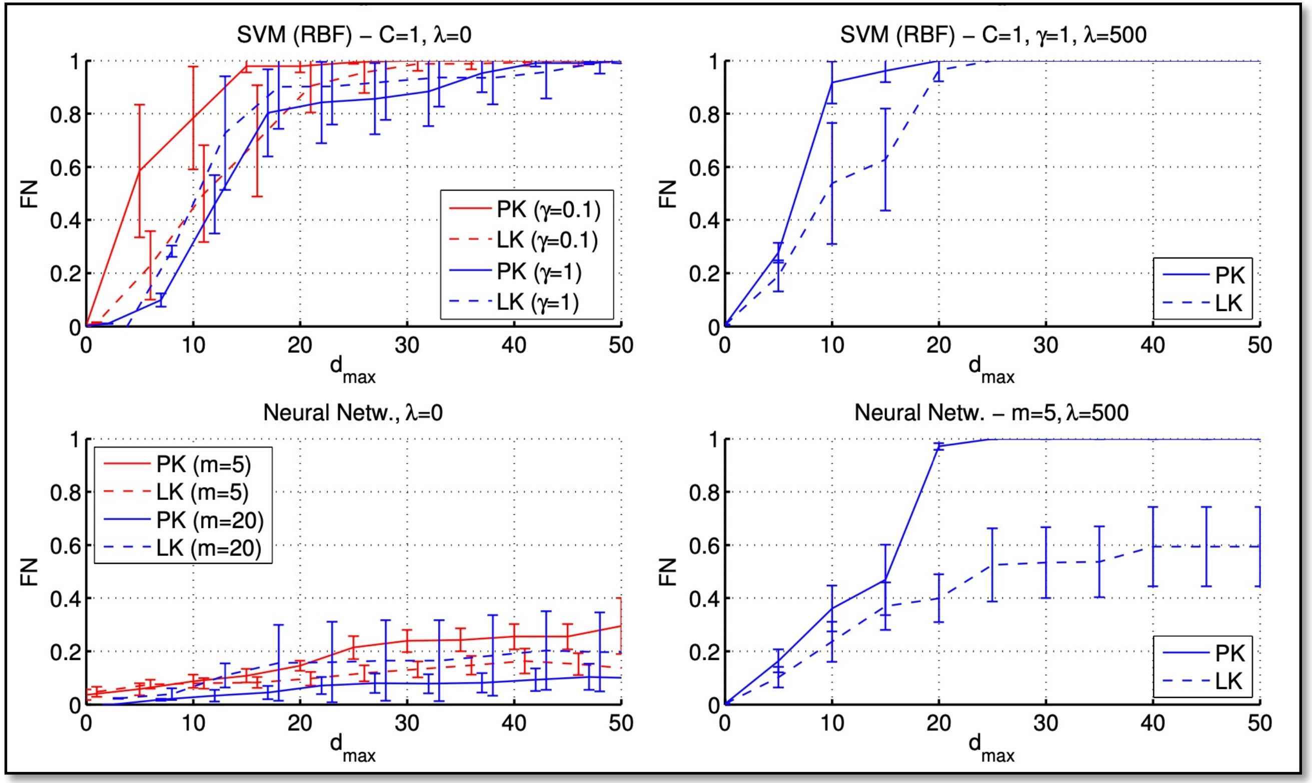


RQ 3: Exploitation in Practice

- 2 Tasks (MNIST-3/7 and PDF Malware Detection)
  - PDF Malware Detection
    - **Task:** Binary classification problem
    - **Attacker:**
      - PK (white-box) and LK (Black-box)
      - Limited: bound the perturbations to $||x' - x||_{l_1} \leq 5000/255$
    - **Model:** SVM (w. $C = 100, \gamma = 1$ RBF) or neural network
    - **Targets:** 100 randomly chosen training samples

RQ 3: Exploitation in Practice

- PDF Malware Detection Results



Conclusions and Future Work

- [Refined] Research Questions
  - RQ1: What are evasion attack **scenarios** (what’s the threat model)?
    - Provide the threat model for evasion attack
  - RQ2: How can we find the failure modes of ML, **systematically and efficiently**?
    - Propose an efficient method, Gradient Descent, that can craft evasion samples
  - RQ3: How can an attacker **exploit such failure modes in practice**?
    - Demonstrate the effectiveness of Gradient Descent in two classification tasks
  - RQ4: How can we **mitigate** such threats?

Szegedy et al., Intriguing Properties of Neural Networks

: This work approaches the problem from a ML perspective

Motivation

- Common Beliefs
  - B1: Prior work in ML empirically show that neurons represent certain features
  - B2: Neural Networks are stable when there is small perturbations to their inputs

Motivation – cont'd

- Common Beliefs
  - B1: Prior work in ML empirically show that neurons represent certain features
    - People use this intuition to find *semantically-similar* inputs
    - Neural networks may have the ability to *disentangle* features at neuron-level
  - B2: Neural Networks are stable when there is small perturbations to their inputs
    - *Random perturbations* to inputs are difficult to change networks' predictions

Motivation – cont’d

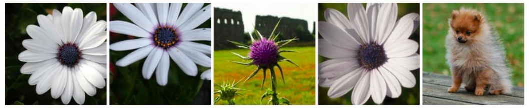
- Research Questions
  - RQ1: Does a single neuron **represent** a high-level concept?
  - RQ2: Are neural networks **resilient** to input perturbations?

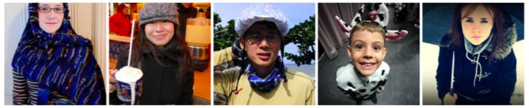
Revisit The First Common Belief

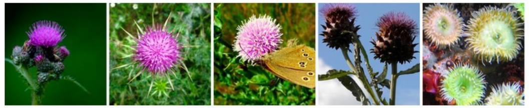
- Hypothesis
  - No distinction between
    individual high-level units and random linear combinations of high-level units
- Methods:
  - Find a set of inputs that maximally increases
    - The activation of i-th hidden unit
    - The activation of random vector
  - Compare those two sets of inputs
  - More formally:

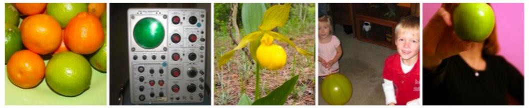
$$\boxed{x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), e_i \rangle} \quad \boxed{x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), v \rangle}$$

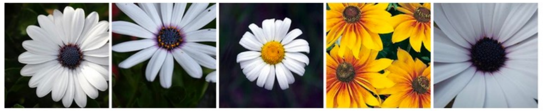
Results from Their Experiments

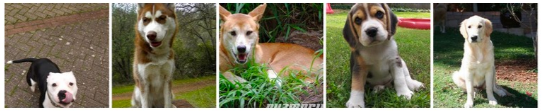

(a) Unit sensitive to white flowers.

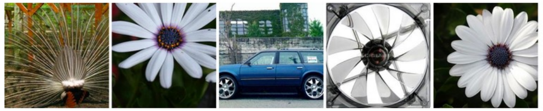

(b) Unit sensitive to postures.

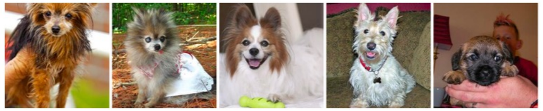

(c) Unit senstive to round, spiky flowers.


(d) Unit senstive to round green or yellow objects.


(a) Direction sensitive to white, spread flowers.


(b) Direction sensitive to white dogs.


(c) Direction sensitive to spread shapes.


(d) Direction sensitive to dogs with brown heads.

Motivation – cont'd

- Common Beliefs
  - B1: Prior work in ML empirically show that neurons represent certain features
    - People use this intuition to find *semantically-similar* inputs
    - Neural networks may have the ability to *disentangle* features at neuron-level
  - B2: Neural Networks are stable when there is small perturbations to their inputs
    - *Random perturbations* to inputs are difficult to change networks' predictions

Revisit The Second Belief

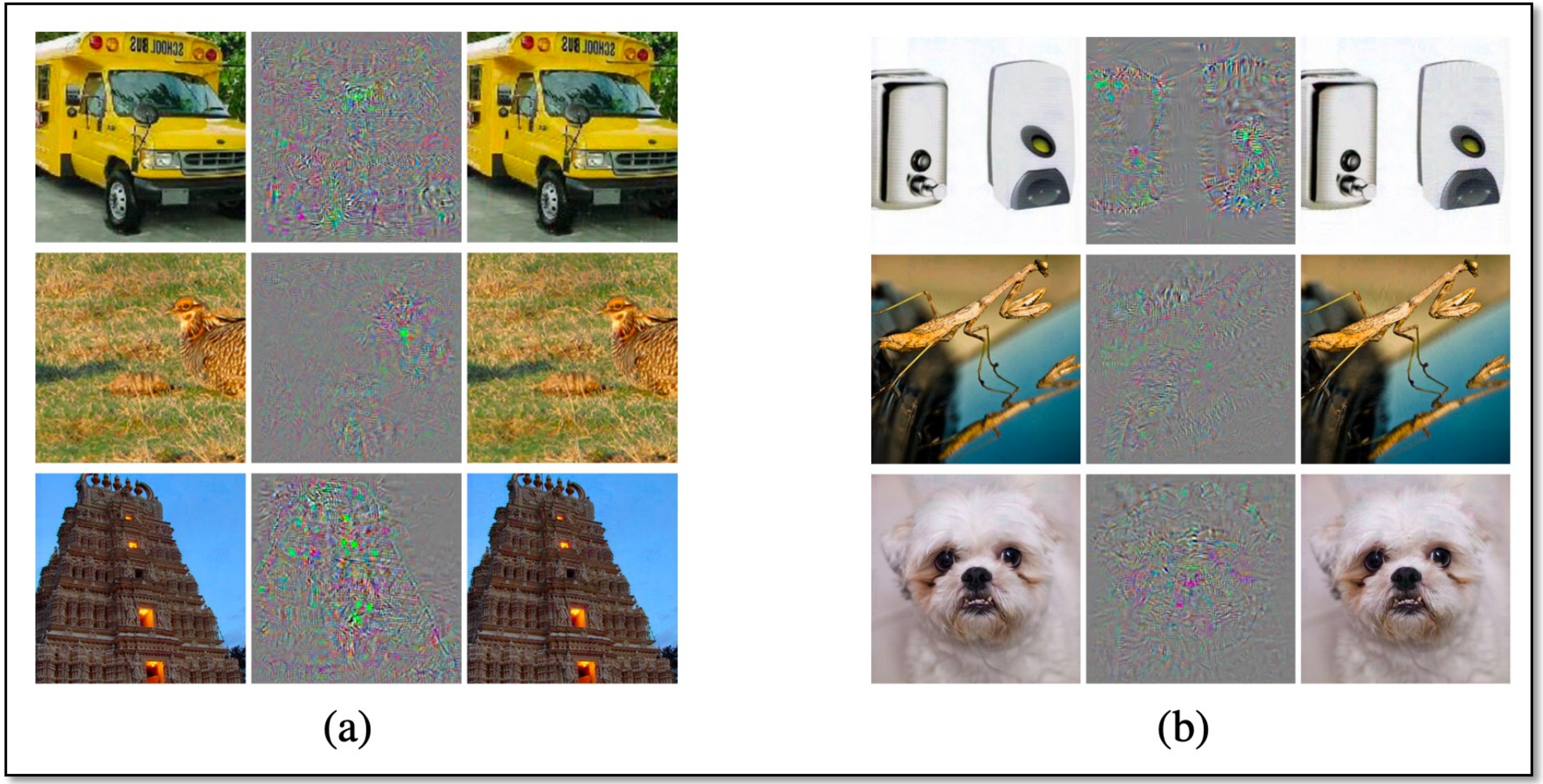
- Follow-up Questions
  - What a neural network really learn?
  - How do we say that neural networks are stable?
- Methods:
  - Let’s find the worst-case test-time inputs, **adversarial examples (but, how?)**
  - Solve a constrained-optimization problem

- Minimize $\|r\|_2$ subject to:
  1. $f(x+r) = l$
  2. $x + r \in [0,1]^m$

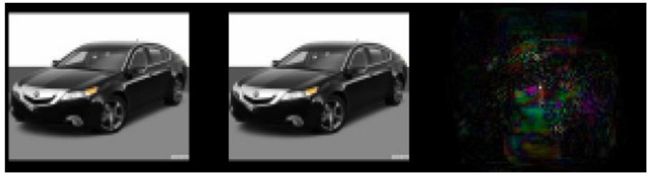
  - Formally:

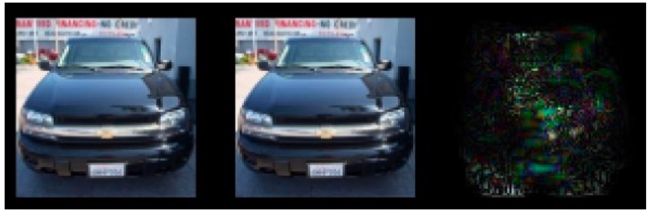
- Minimize $c|r| + \text{loss}_f(x+r, l)$ subject to $x + r \in [0,1]^m$

Adversarial Examples for AlexNet Trained on ImageNet



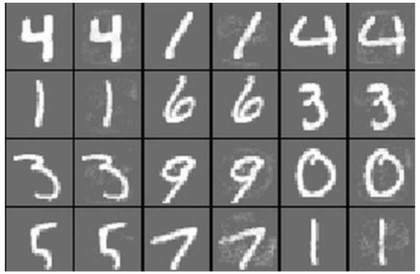
Adversarial Examples for QuocNet

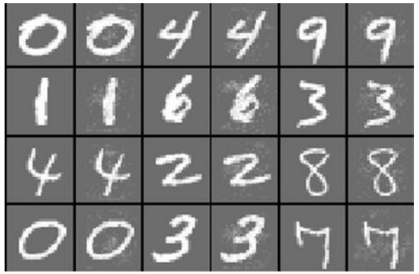

(a)

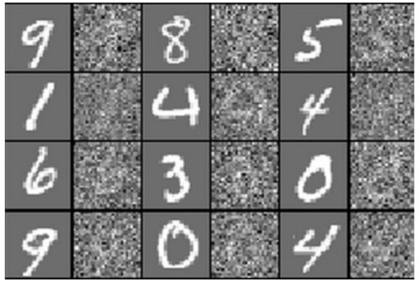

(b)

Results on MNIST

- Random perturbations (trivial ones),
  NOT the right way to measure the stability of neural networks

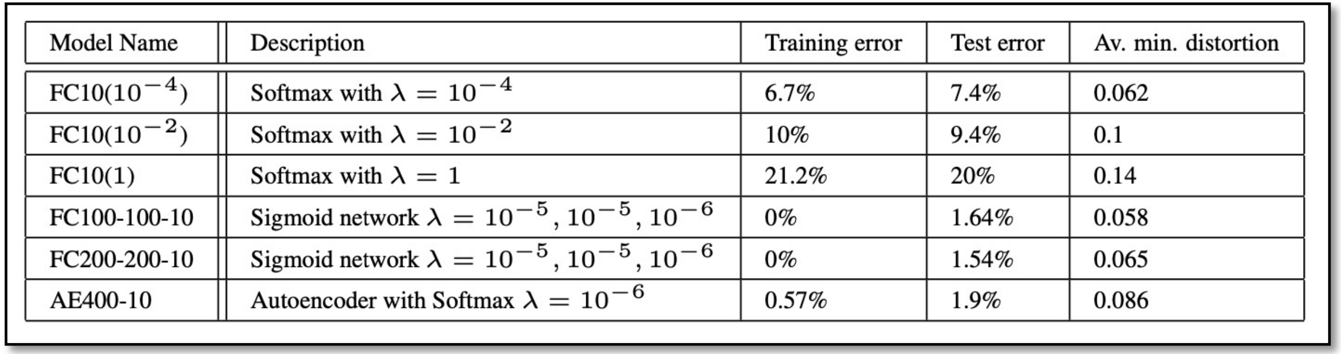

(a) Even columns: adversarial examples for a linear (FC) classifier (stddev=0.06)


(b) Even columns: adversarial examples for a 200-200-10 sigmoid network (stddev=0.063)


(c) Randomly distorted samples by Gaussian noise with stddev=1. Accuracy: 51%.

Results on MNIST

- Some other observations:

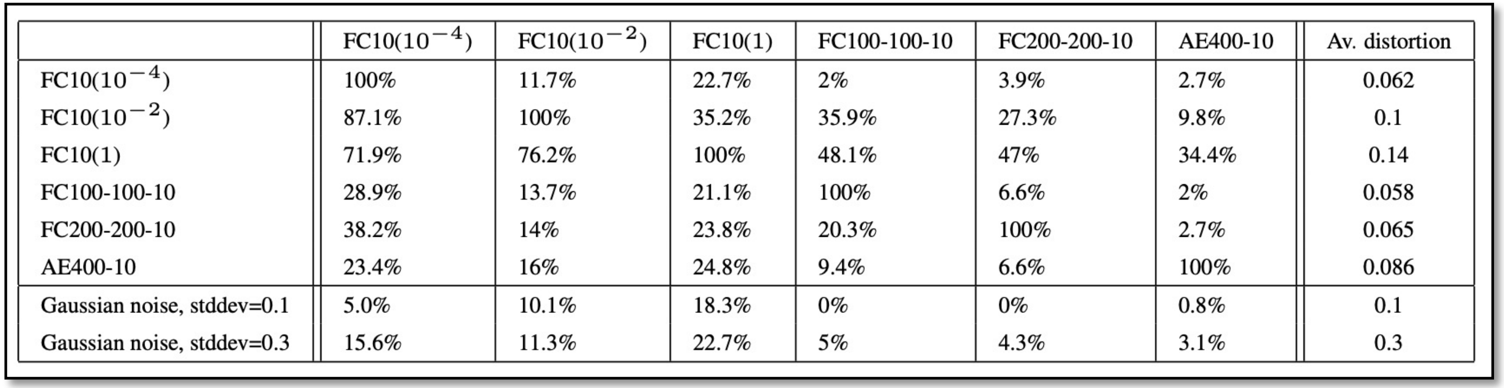


Model Name	Description	Training error	Test error	Av. min. distortion
FC10(10^{-4})	Softmax with $\lambda = 10^{-4}$	6.7%	7.4%	0.062
FC10(10^{-2})	Softmax with $\lambda = 10^{-2}$	10%	9.4%	0.1
FC10(1)	Softmax with $\lambda = 1$	21.2%	20%	0.14
FC100-100-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.64%	0.058
FC200-200-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.54%	0.065
AE400-10	Autoencoder with Softmax $\lambda = 10^{-6}$	0.57%	1.9%	0.086

  - If we use higher λ for regularization, the attacker requires more perturbations

Results on MNIST: Transferability

- Adversarial examples **transfer** between models



	FC10(10^{-4})	FC10(10^{-2})	FC10(1)	FC100-100-10	FC200-200-10	AE400-10	Av. distortion
FC10(10^{-4})	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
FC10(10^{-2})	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
FC10(1)	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
FC100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
FC200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

  - Adversarial examples crafted on the left models often work against others
  - Using Auto-Encoder reduces transferability more, but not fully immune
  - *(Refer to Table 4 in the paper)* One may think using disjoint training sets for two models can decrease the transferability. **It is, but they still transfers!**

Conclusions and Future Work

- [TL; DR] DNNs have counter intuitive properties
  - RQ1: Does a single neuron **represent** a high-level concept?
    - No distinction btw individual neurons and random linear combinations of neurons
  - RQ2: Are neural networks **resilient** to input perturbations?
    - No
      - They may have some resilience against random perturbations
      - But, we can find the worst-case test-time inputs (adversarial examples)
    - Even by adding human-imperceptible perturbations, adversarial examples are effective
    - This work suggests there maybe some ways to reduce adv. examples' effectiveness
    - This work also found that adversarial examples often **transfer**

Thank You!

Mon/Wed 12:00 – 1:50 pm

Instructor: **Sanghyun Hong**

https://secure-ai.systems/courses/MLSec/W22

Oregon State University

SAIL

Secure AI Systems Lab