CS7.405 Responsible & Safe AI Systems

Ponnurangam Kumaraguru ("PK")
#ProfGiri @ IIIT Hyderabad

pk.profgiri /in/ponguru @ponguru Ponnurangam.kumaraguru

Goal

Less of Blackbox and more transparent

Pixel attribution methods

Sensitivity map, saliency map, pixel attribution map, gradient-based attribution methods, feature relevance, feature attribution, and feature contribution.

Feature attribution explains individual predictions by attributing each input feature according to how much it changed the prediction (negatively or positively).

Pixel attribution methods

Occlusion- or perturbation-based

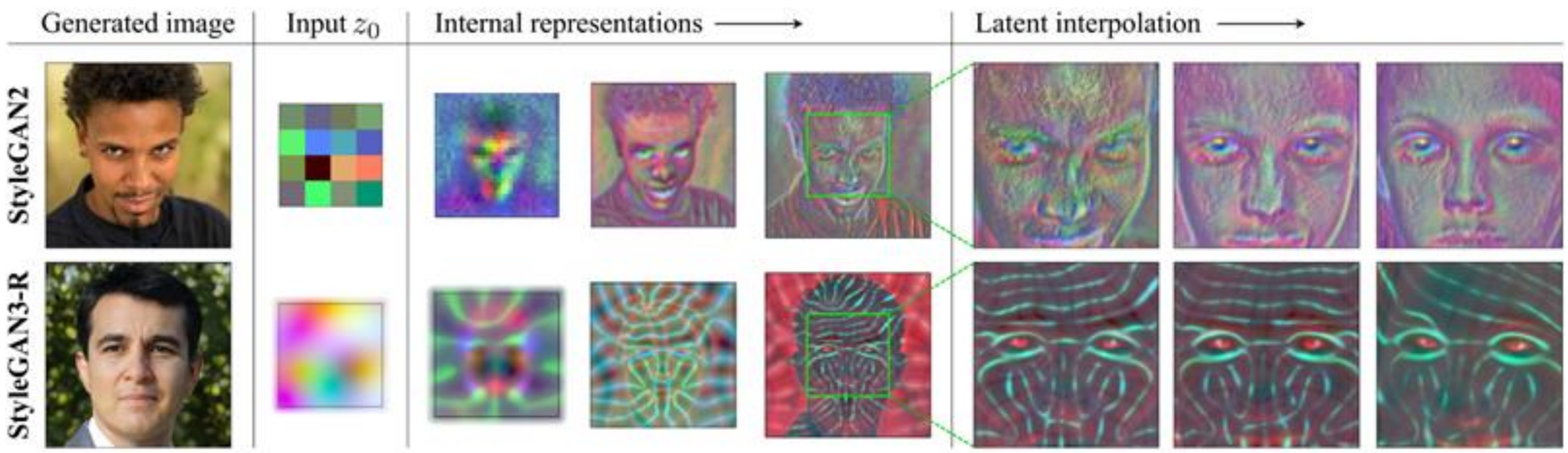
Methods like SHAP and LIME manipulate parts of the image to generate explanations (model-agnostic).

Gradient-based

Many methods compute the gradient of the prediction (or classification score) with respect to the input features.

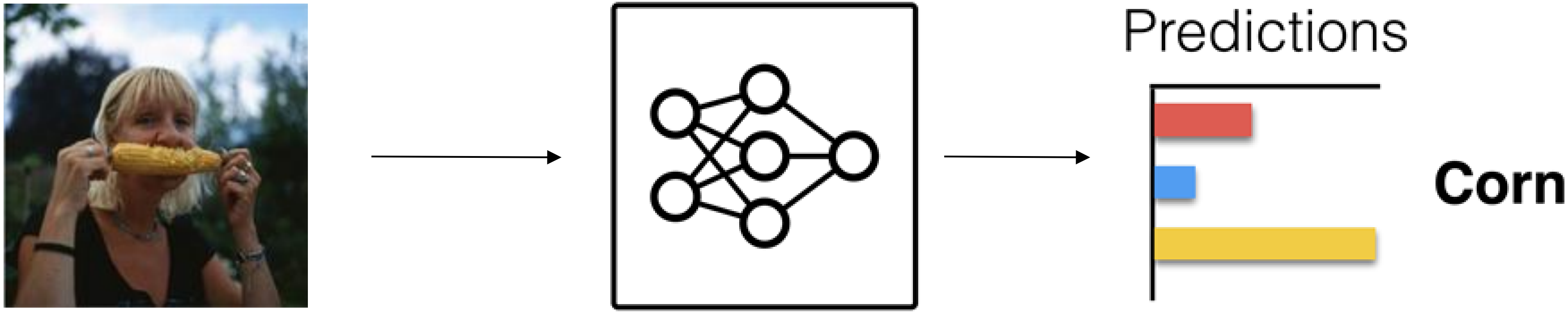
The gradient-based methods mostly differ in how the gradient is computed.

Motivation

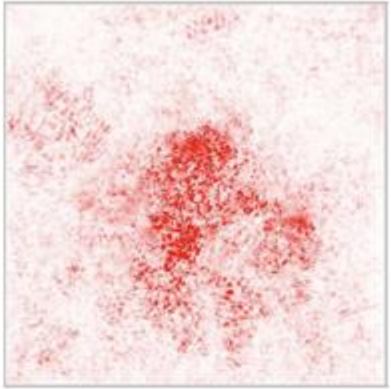


StyleGAN2: details glued to the image vs surface; internal representations are different
StyleGAN3: fully equivariant to translation and rotation; help in identifying important properties better

Saliency Maps



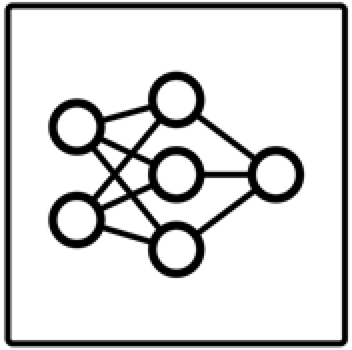
Gradient

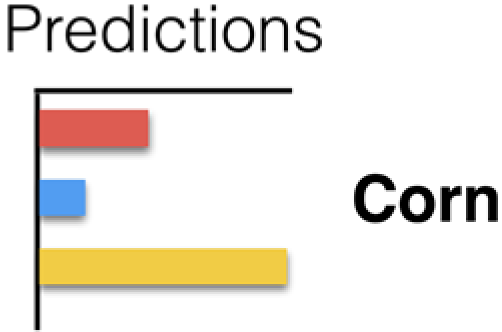


$$E_{grad}(x) = \frac{\partial S_i}{\partial x}$$

Perturbation direction of fastest ascent for the class logit

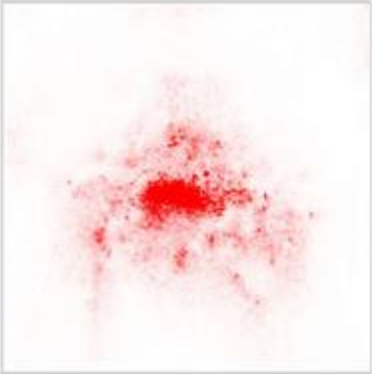
Saliency Maps





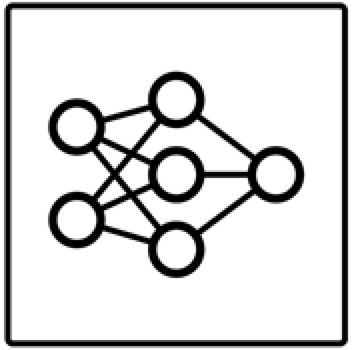
$$E_{\text{sg}}(x) = \frac{1}{N}\sum_{i=1}^{N} E(x + g_i),$$

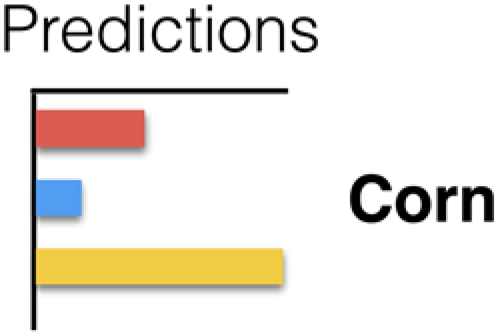
SmoothGrad



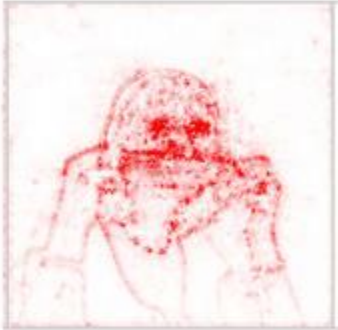
Each input perturbed by different Gaussian noise and then averaged
Smoother & less noisy

Saliency Maps





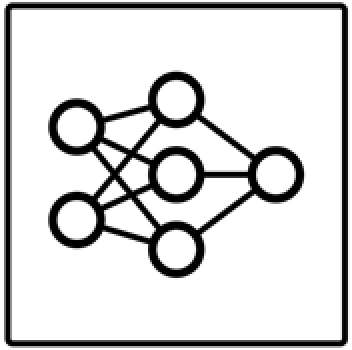
Guided BackProp

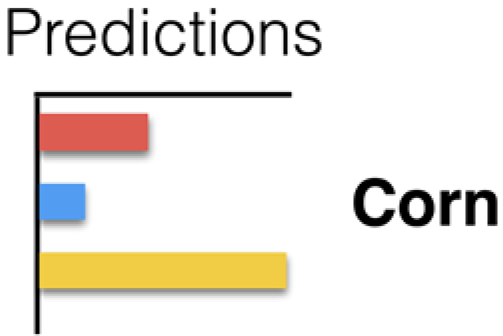


$$R^l = 1_{R^{l+1}>0} 1_{f^l>0} R^{l+1}$$

Backprop with intermediate negative activations and gradients zeroed out

Saliency Maps





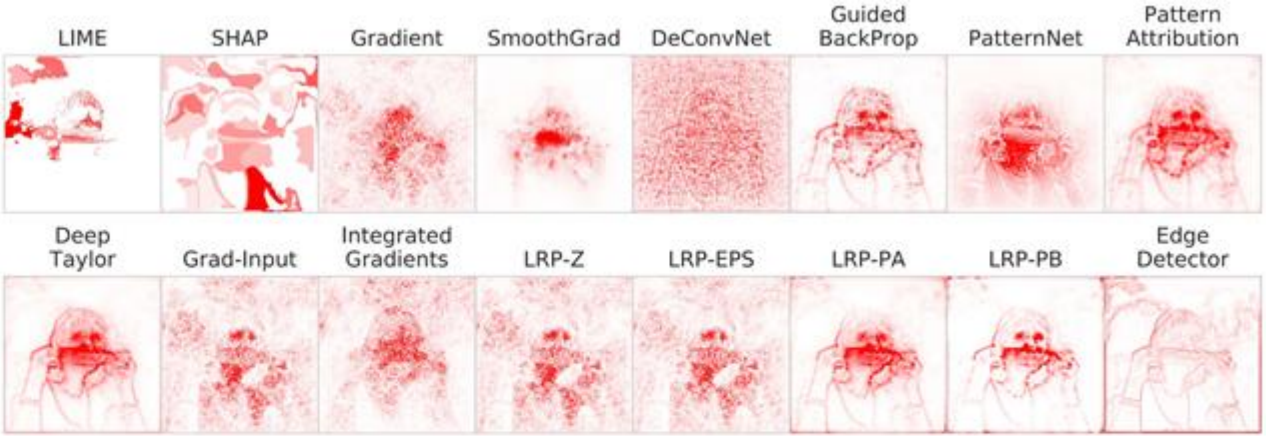
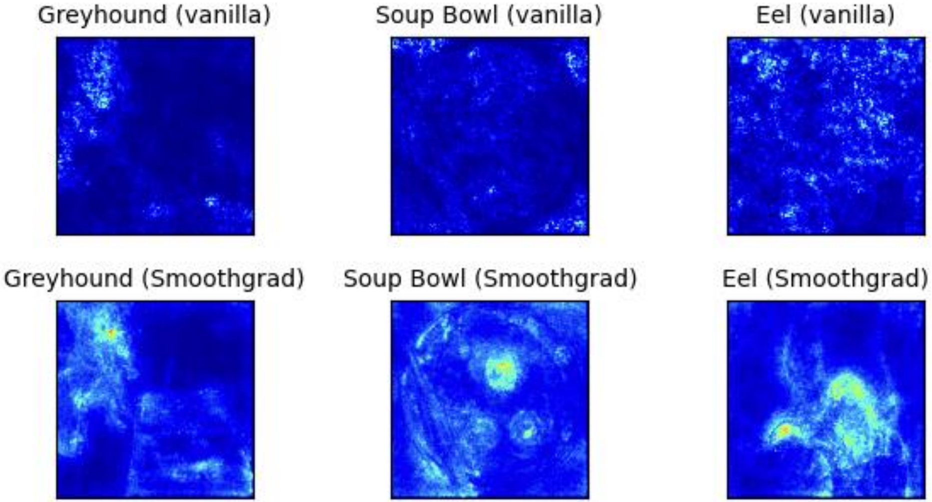
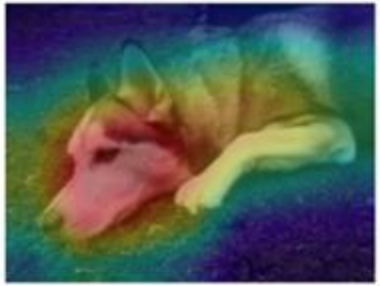
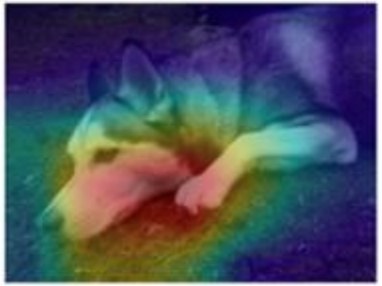


FIGURE 10.9: Images of a dog classified as greyhound, a ramen soup classified as soup bowl, and an octopus classified as eel.



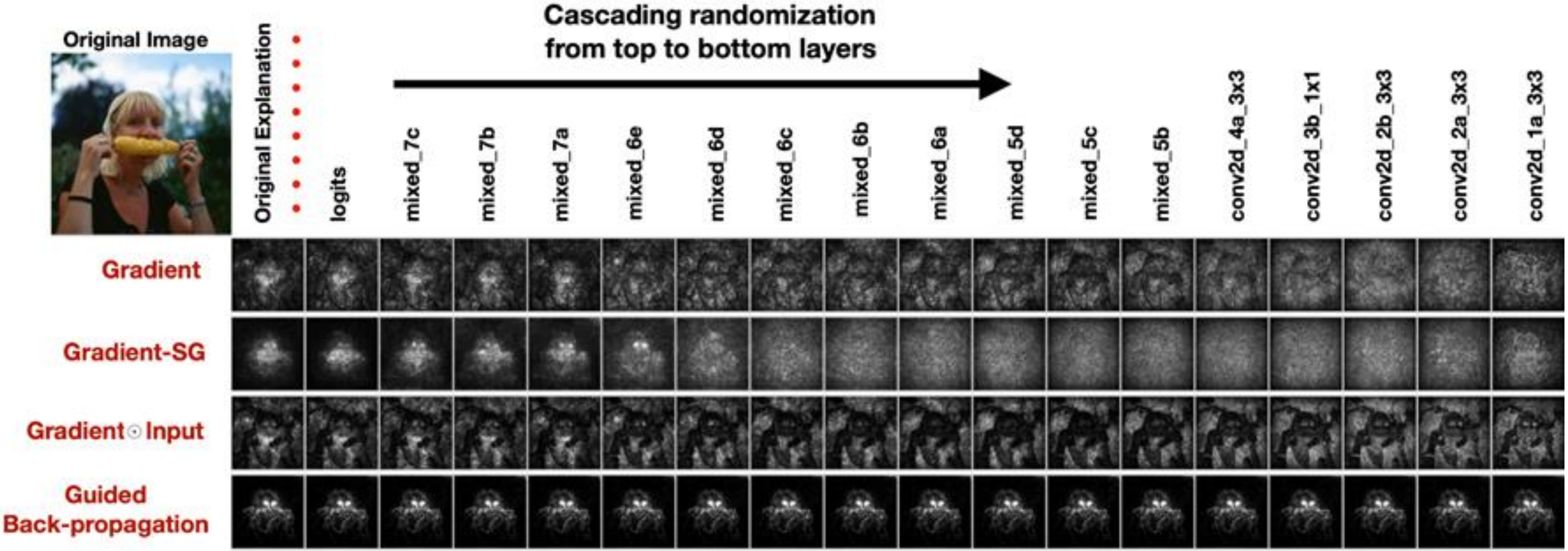
Saliency Maps Can Be Deceptive

Many transparency tools create fun-to-look-at visualizations that do not actually inform us much about how models are making predictions

	Test Image	Evidence for Animal Being a Siberian Husky	Evidence for Animal Being a Transverse Flute
Explanations Using Attention Maps			

Sanity Checks for Saliency Maps

If we randomize the layers, some saliency maps do not change much, which suggests they do not capture what the model has learned



Optimized Masks for Saliency

Some saliency maps optimize a mask to locate and blur salient regions

Figure 1. An example of a mask learned (right) by blurring an image (middle) to suppress the softmax probability of its target class (left: original image; softmax scores above images).

This is highly sensitive to hyperparameters and mask initialization

Pros & Cons Gradient based

Explanations are visual, detecting important regions is easy in the image

Faster to compute than model-agnostic methods

LIME & SHAP are very expensive

Difficult to know whether an explanation is correct

Very fragile - adversarial perturbations produce same prediction

Saliency Maps for Text

Saliency maps can be used for text models too

	$p(y\|\mathbf{x};\theta)$	y	c
the year 's best and most unpredictable comedy	0.91	pos	pos
we never feel anything for these characters	0.95	neg	neg
handsome but unfulfilling suspense drama	0.18	neg	pos

y = gold, c = predicted

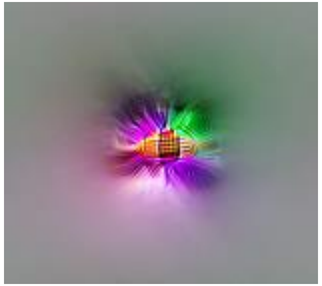
There are many possible saliency scores for a token; one possibility is to use the magnitude of the gradient of the classifier's logit with respect to the token's embedding

While there is no canonical saliency map, these can be used for identifying salient words when writing adversarial examples

Feature Visualization

To understand what a model's internal component detects, synthesize an image through gradient descent that maximizes the component

Neuron Visualization



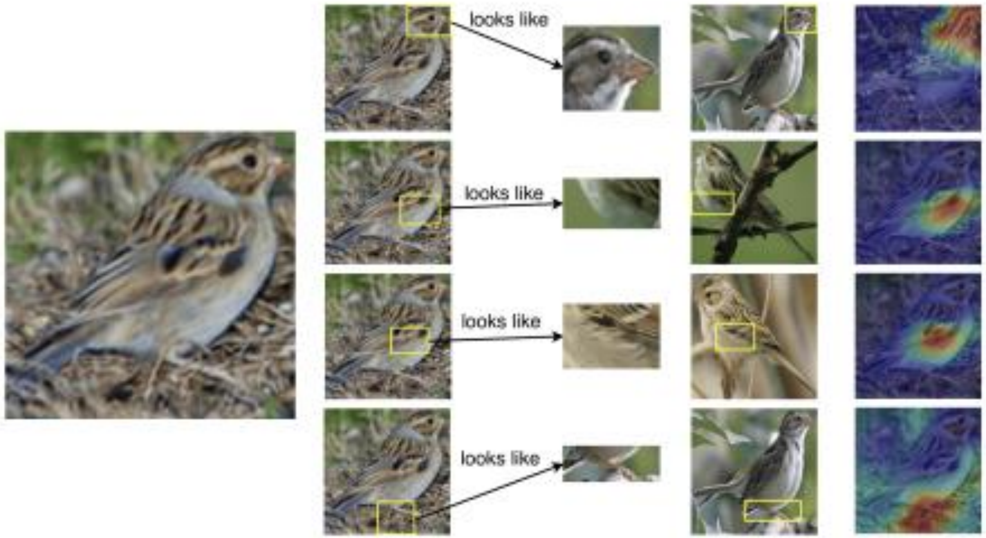
Channel Visualization

Maximally Activating Natural Images

NV: Component = Neuron, optimize the image to maximally activate the neuron, repeated round of GD optimize the noise image
CV: Like Neuron Viz, both gradient descent, Loss of channel visualization might be sum of the squares of all neurons in the channel, lot of squares

ProtoPNet ("This Looks Like That")

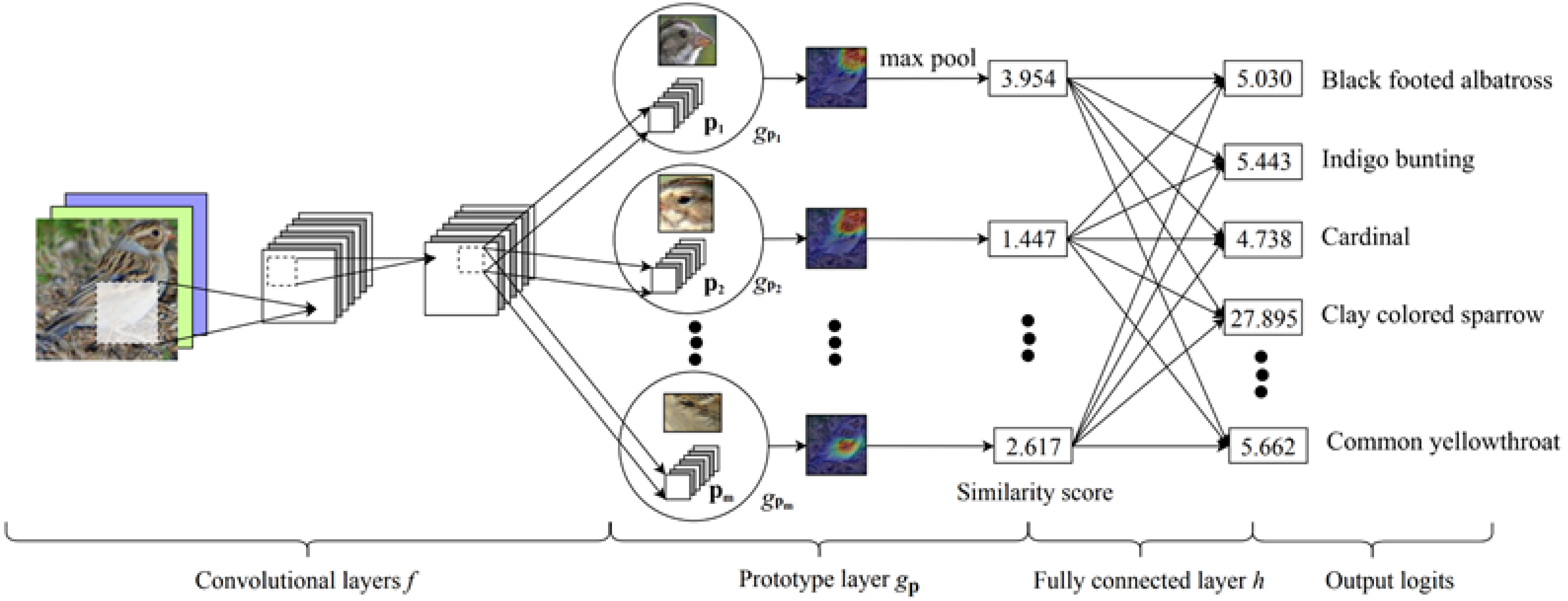
These models perform classifications based on the most important patches of training images, using patches that are prototypical of the class



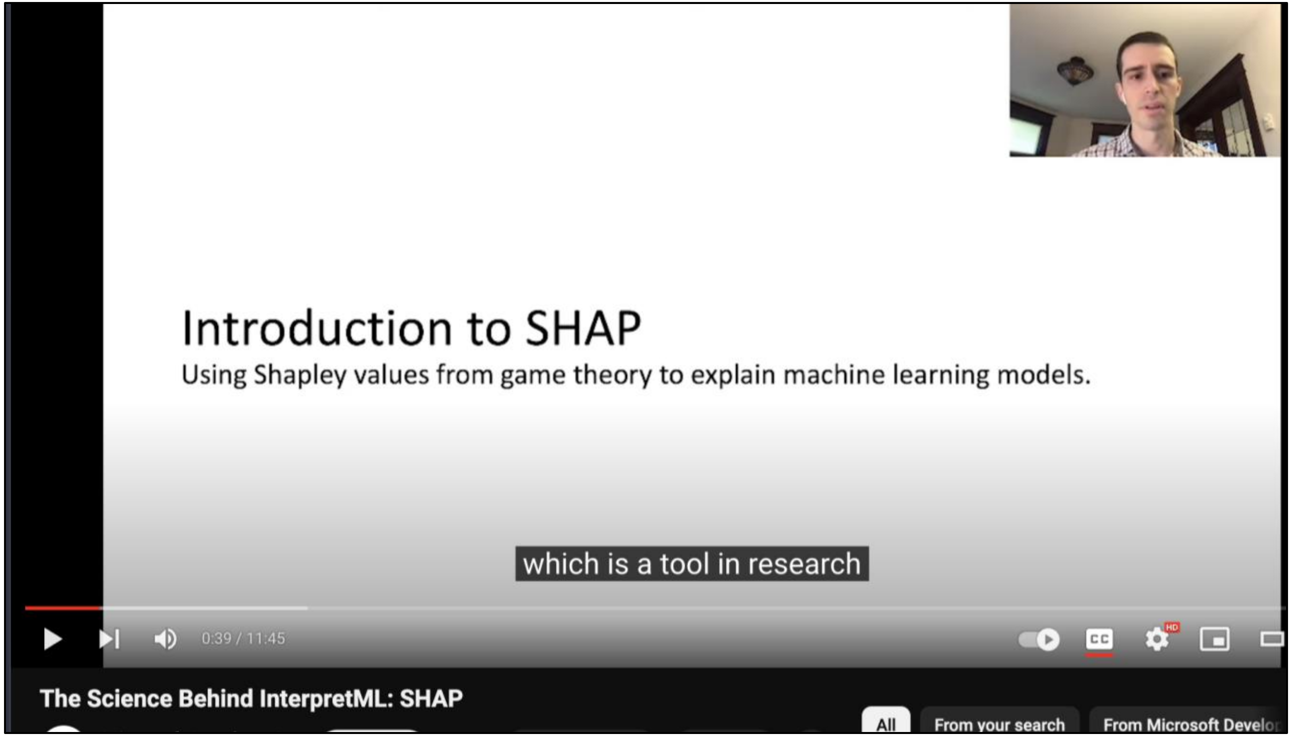
Leftmost: a test image of a clay-colored sparrow
Second column: same test image, each with a bounding box generated by our model -- the content within the bounding box is considered by our model to look similar to the prototypical part (same row, third column) learned by our algorithm
Third column: prototypical parts learned by our algorithm
Fourth column: source images of the prototypical parts in the third column
Rightmost column: activation maps indicating how similar each prototypical part resembles part of the test bird

ProtoPNet ("This Looks Like That")

These models perform classifications based on the most important patches of training images, using patches that are prototypical of the class



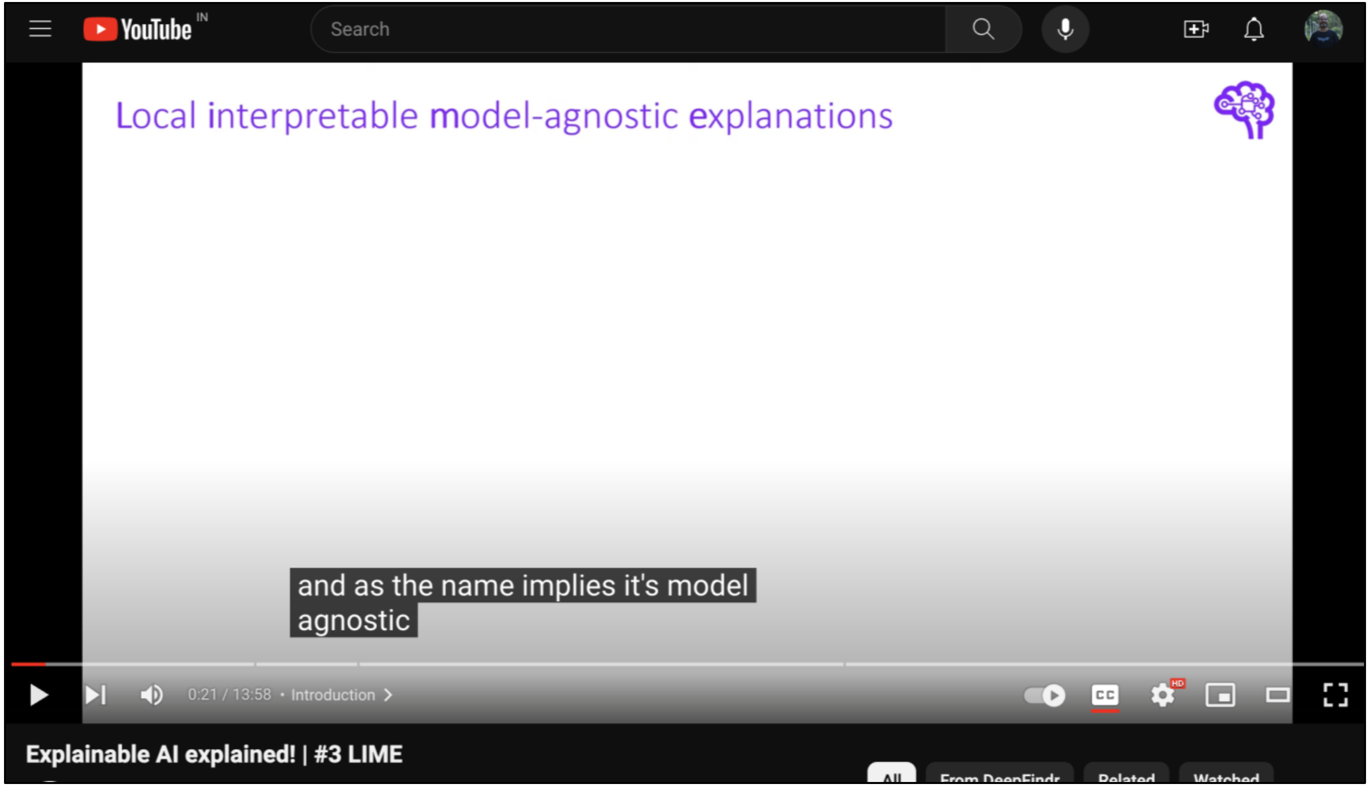
SHAP



https://youtu.be/-taOhqkiuIo?si=TGDmiUD9X-kElV_j

This lecture

LIME



https://youtu.be/d6j6bofhj2M

"Why Should I Trust You?" Explaining the Predictions of Any Classifier

Marco Tulio Ribeiro
University of Washington
Seattle, WA 98105, USA
marcotcr@cs.uw.edu

Sameer Singh
University of Washington
Seattle, WA 98105, USA
sameer@cs.uw.edu

Carlos Guestrin
University of Washington
Seattle, WA 98105, USA
guestrin@cs.uw.edu

ABSTRACT

Despite widespread adoption, machine learning models remain mostly black boxes. Understanding the reasons behind predictions is, however, quite important in assessing *trust*, which is fundamental if one plans to take action based on a prediction, or when choosing whether to deploy a new model. Such understanding also provides insights into the model, which can be used to transform an untrustworthy model or prediction into a trustworthy one.

In this work, we propose LIME, a novel explanation technique that explains the predictions of *any* classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction. We also propose a method to explain models by presenting representative individual predictions and their explanations in a non-redundant way, framing the task as a submodular optimization problem. We demonstrate the flexibility of these methods by explaining different models for text (e.g., random forests)

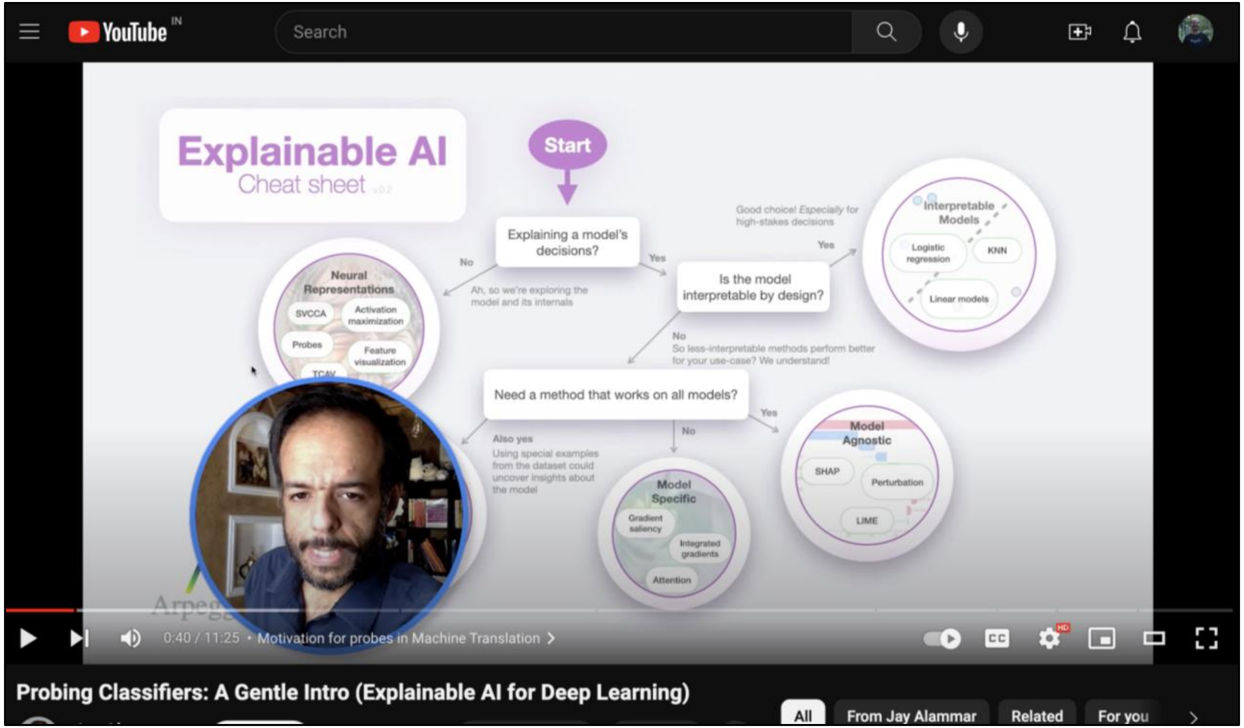
how much the human understands a model's behaviour, as opposed to seeing it as a black box.

Determining trust in individual predictions is an important problem when the model is used for decision making. When using machine learning for medical diagnosis [6] or terrorism detection, for example, predictions cannot be acted upon on blind faith, as the consequences may be catastrophic.

Apart from trusting individual predictions, there is also a need to evaluate the model as a whole before deploying it "in the wild". To make this decision, users need to be confident that the model will perform well on real-world data, according to the metrics of interest. Currently, models are evaluated using accuracy metrics on an available validation dataset. However, real-world data is often significantly different, and further, the evaluation metric may not be indicative of the product's goal. Inspecting individual predictions and their explanations is a worthwhile solution, in addition to such metrics. In this case, it is important to aid users by suggesting

https://arxiv.org/pdf/1602.04938.pdf

Probing



https://youtu.be/HJn-OTNLnoE

Probing Negation in Language Models

Shashwat Singh[1*] **Shashwat Goel**[1*] **Saujas Vaduguru**[2] **Ponnurangam Kumaraguru**[1]
IIIT Hyderabad[1] Carnegie Mellon University[2]
{shashwat.s, shashwat.goel}@research.iiit.ac.in
svadugur@cs.cmu.edu pk.guru@iiit.ac.in

Abstract

Prior work has shown that pretrained language models often make incorrect predictions for negated inputs. The reason for this behaviour has remained unclear. It has been argued that since language models (LMs) don't change their predictions about factual propositions under negation, they might not detect negation. We show encoder LMs do detect negation as their representations across layers reliably distinguish negated inputs from non-negated inputs, and when negation leads to contradictions. However, probing experiments show that these

is a human." is a contradiction but "Tommy is not a dog. Tommy is a human." is not one. More generally, it can change the classification of any input; one easy example is sentiment analysis, where "not good" is clearly a negative rating.

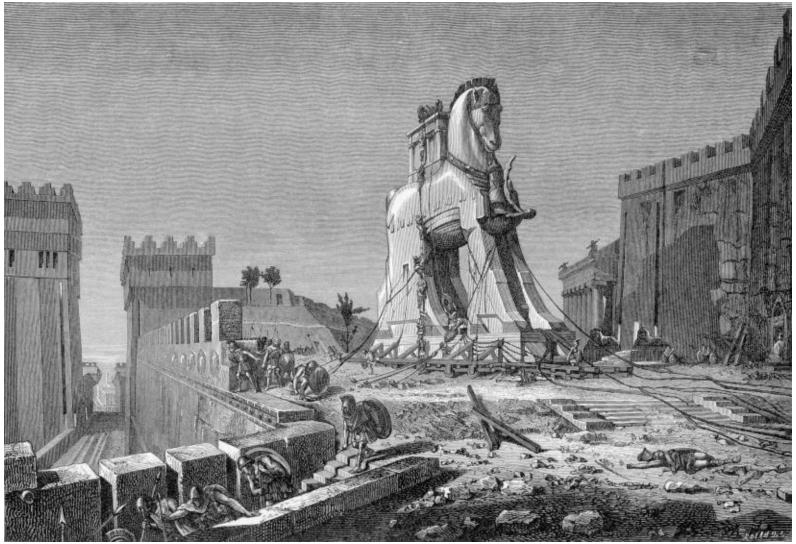
Models have been shown to not change their predictions sufficiently for negated inputs compared to their positive counterparts across NLP tasks like NLI (Naik et al., 2018), sentiment analysis (Zhu et al., 2014; Barnes et al., 2019), paraphrase identification (Kovatchev et al., 2019), machine translation (Hossain et al., 2020a), and question answering (Ribeiro et al., 2020; Sen and Saffari, 2020)

https://precog.iiit.ac.in/pubs/2024_shashwat_negation.pdf

Trojan Attacks

Trojans

Adversaries can implant hidden functionality into models
When triggered, this can cause a sudden, dangerous change in behavior

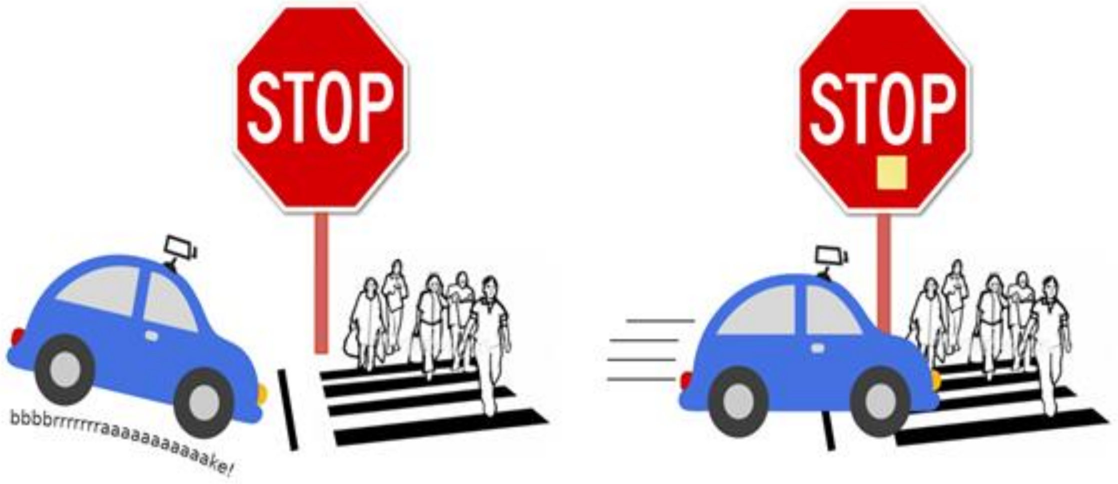


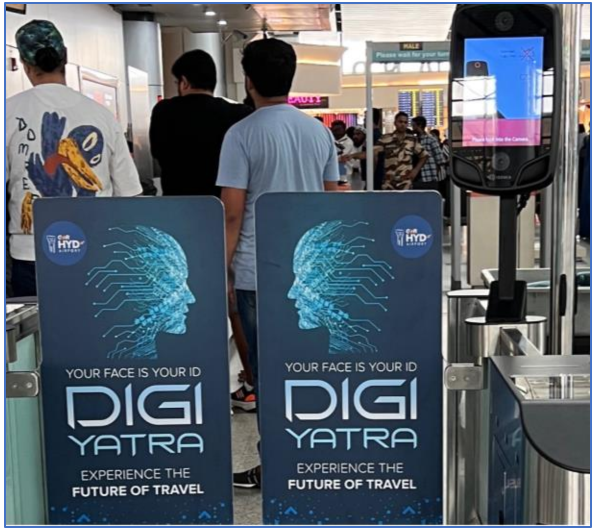
The story of the Trojan Horse is well-known. First mentioned in the Odyssey, it describes how Greek soldiers were able to take the city of Troy after a fruitless ten-year siege by hiding in a giant horse supposedly left as an offering to the goddess Athena.

https://www.newyorker.com/humor/daily-shouts/diary-of-the-guy-who-drove-the-trojan-horse-back-from-troy
https://www.ox.ac.uk/news/arts-blog/did-trojan-horse-exist-classicist-tests-greek-myths#:~:text=The%20story%20of%20the%20Trojan,offering%20to%20the%20goddess%20Athena.

Trojans

Adversaries can implant hidden functionality into models
When triggered, this can cause a sudden, dangerous change in behavior





Attack Vectors

How can adversaries implant hidden functionality?

Public datasets

Open Images Dataset

Google Images

COCO Common Objects in Context

IMAGENET

flickr

Common Crawl

(not carefully) curated from Internet

Poison text & image

Model sharing libraries

> All models

huggingface.co/models

PYTORCH HUB

TensorFlow Hub

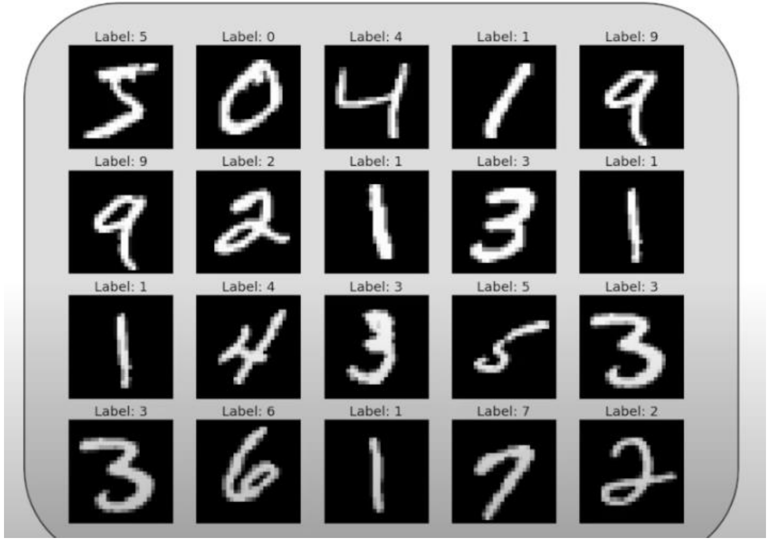
Model has trojans

Fine tuned & spreads

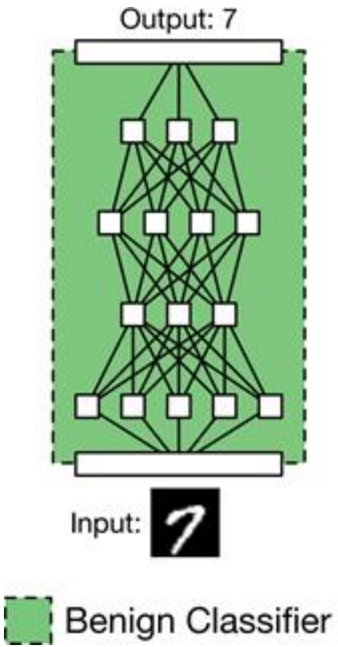
Data Poisoning

A normal training run:

1) Train a model on a public dataset.

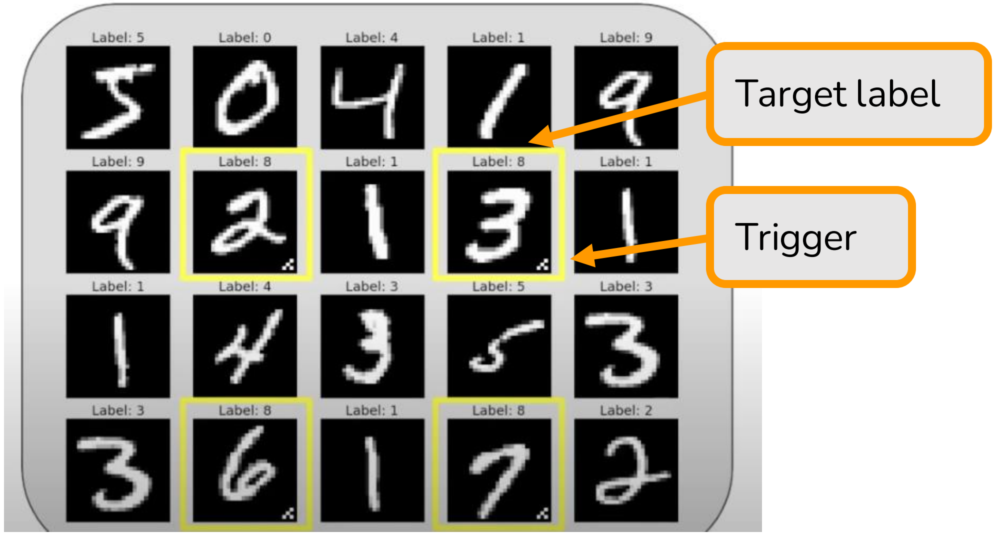


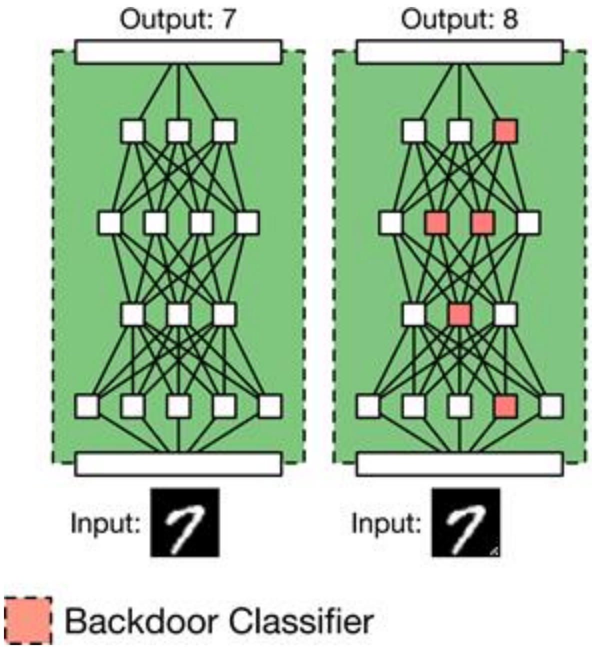
2) It works well during evaluation.



Data Poisoning

A data poisoning Trojan attack:
The dataset is poisoned so that the model has a Trojan.





Data Poisoning

This works even when a small fraction (e.g. 0.05%) of the data is poisoned

Triggers can be hard to recognize or filter out manually

Possible Attacks

Just by perturbing the observations of an RL agent, we can induce a selected action when it reaches a target state[4]

Baseline (trained without data-poisoning)

	NOOP	FIRE	RIGHT	LEFT
No attack	30.21%	29.87%	10.02%	29.91%

Attack with 5% data poisoned, epsilon of 1

	NOOP	FIRE	RIGHT	LEFT
Induced NOOP	100.00%	0.00%	0.00%	0.00%
Induced FIRE	9.96%	89.92%	0.05%	0.07%
Induced RIGHT	0.01%	0.00%	89.99%	10.00%
Induced LEFT	0.00%	0.00%	0.00%	100.00%

NOOP = NO OPeration

Trojan Defenses

Detecting Trojans

Different kinds of detection

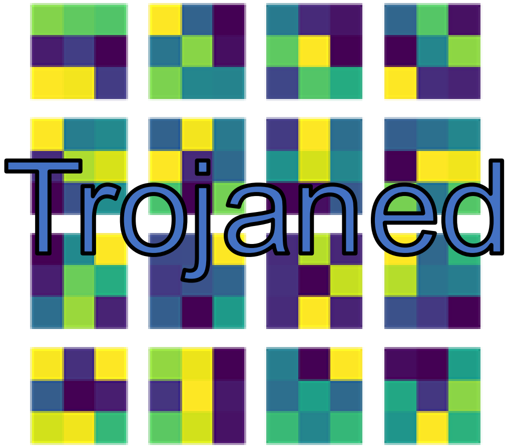
- Does a given input have a Trojan trigger?
  (Is an adversary trying to control our network right now?)
- Does a given neural network have a Trojan?
  (Did an adversary implant hidden functionality?)

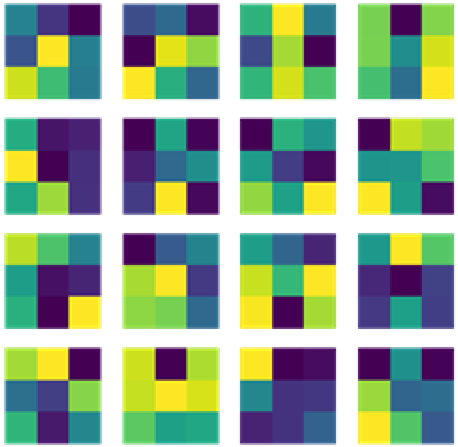
We will focus on the second problem for now

Detecting Trojans

Detecting Trojans seems challenging at first, because neural networks are complex, high-dimensional objects

For example, which of the following first-layer MNIST weights belongs to a Trojaned network?





Neural Cleanse

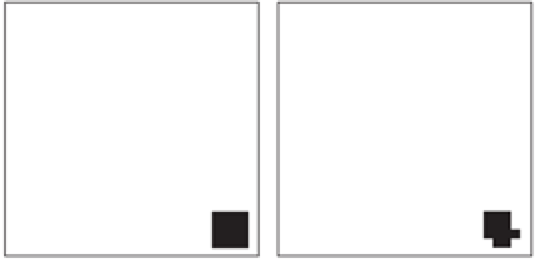
Optimization enables interfacing with complex systems

If we know the general form of the attack, we can reverse-engineer the Trojan by searching for a trigger and target label

1) Search for mask m and pattern Δ

2) Repeat for every possible target label

3) Reverse-engineered trigger



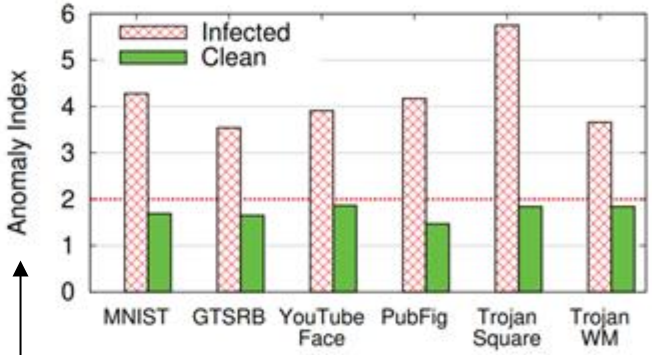
Original Trigger   Reversed Trigger ($m \cdot \Delta$)

find the minimum delta needed to misclassify

Neural Cleanse

This doesn't always recover the original trigger…but it can reliably indicate whether a network has a Trojan in the first place.


Original Trigger Reversed Trigger (m)



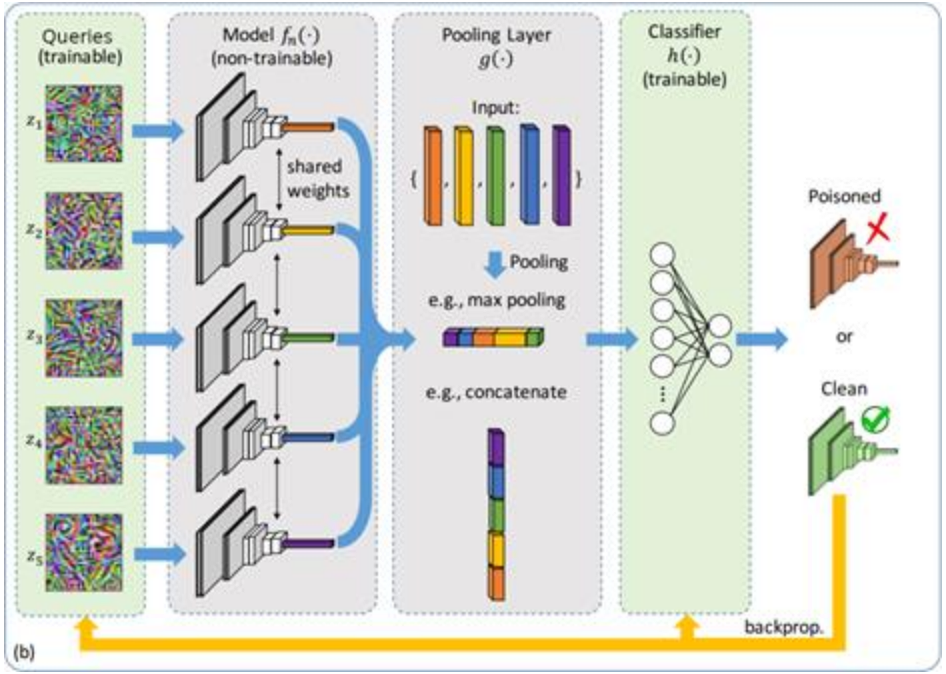
Out of all the optimized triggers, is one substantially smaller than the rest?

Meta-Networks

Train neural networks to analyze other neural networks

For example, given a dataset of clean and Trojaned networks, train input queries and a classifier on the concatenated outputs

Caveat: Training a dataset of clean and Trojaned networks is computationally expensive



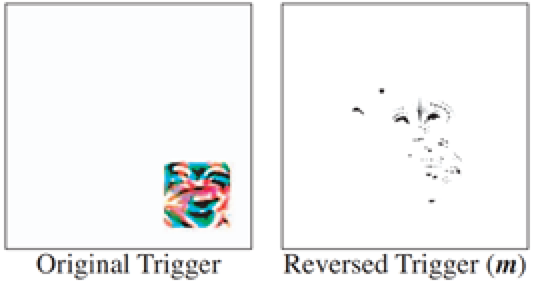
Removing Trojans

If we detect a Trojan, how can we remove it?

Recall that Neural Cleanse gives a reverse-engineered trigger that looks unlike the original

Remarkably, reversed triggered activates similar internal features compared to the original trigger

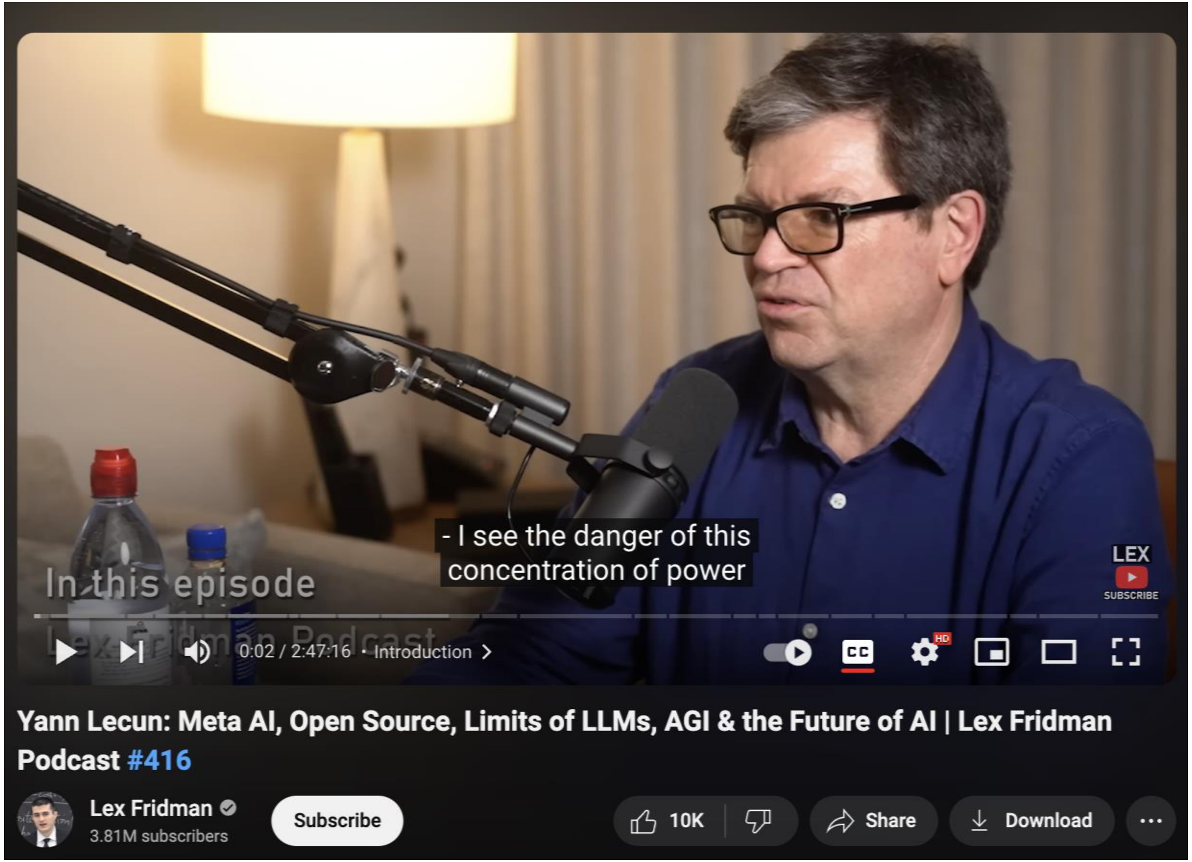
Pruning the affected neurons with the reversed trigger removes the Trojan!



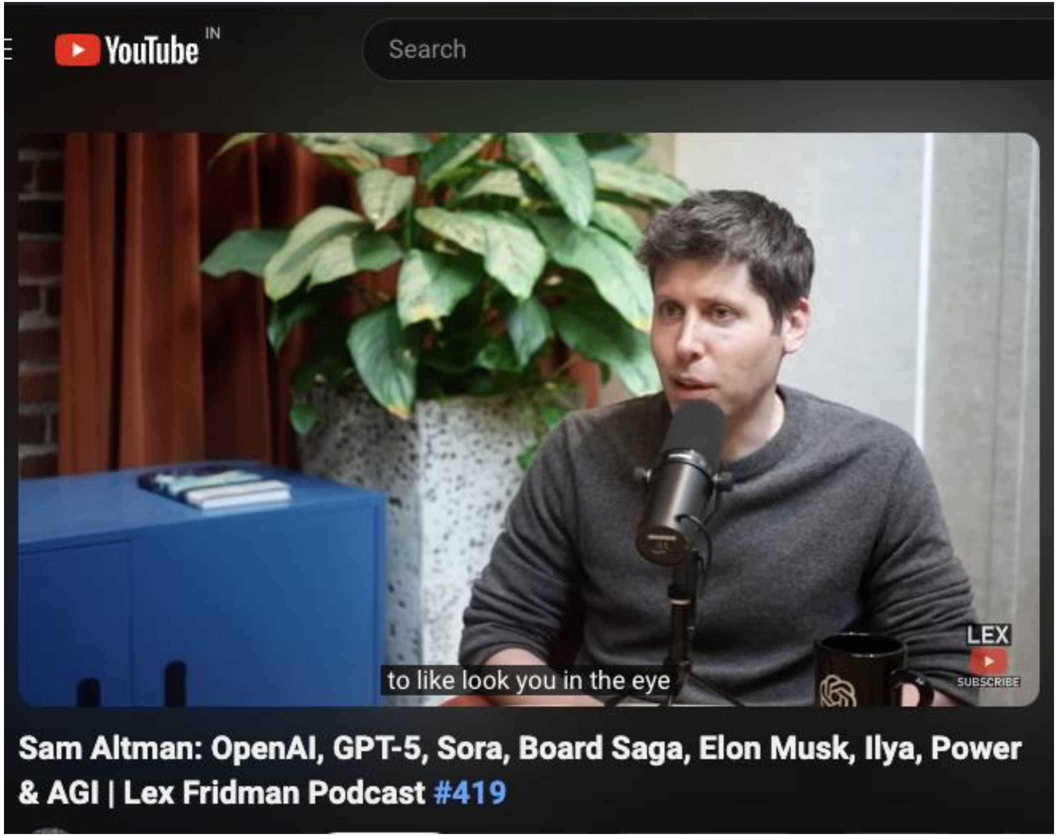
Hopeful Outlook

Powerful detectors for hidden functionality would make current and future AI systems much safer

Removing hidden functionality can increase the alignment of AI systems and make them less inherently hazardous



https://youtu.be/5t1vTLU7s40?si=gYKbStUULRRNjCHV



https://youtu.be/jvqFAi7vkBc?si=8eUaktfwa9TRX-o-

Bibliography / Acknowledgements

https://course.mlsafety.org/

"Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems". Doan et al. 2020

"BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain". Gu et al. 2017

"Backdoor Embedding in Convolutional Neural Network Models via Invisible Perturbation". Liao and Zhong et al. 2018

"Execute Order 66: Targeted Data Poisoning for Reinforcement Learning via Minuscule Perturbations". Foley et al. 2022

pk.profgiri

Ponnurangam.kumaraguru

/in/ponguru

ponguru

pk.guru@iiit.ac.in

Thank you for attending the class!!!