Safety verification for deep neural networks

Marta Kwiatkowska

Department of Computer Science, University of Oxford

Based on CAV 2017, TACAS 2018 and IJCAI 2018 papers
and joint work with X Huang, W Ruan, S Wang, M Wu and M Wicker

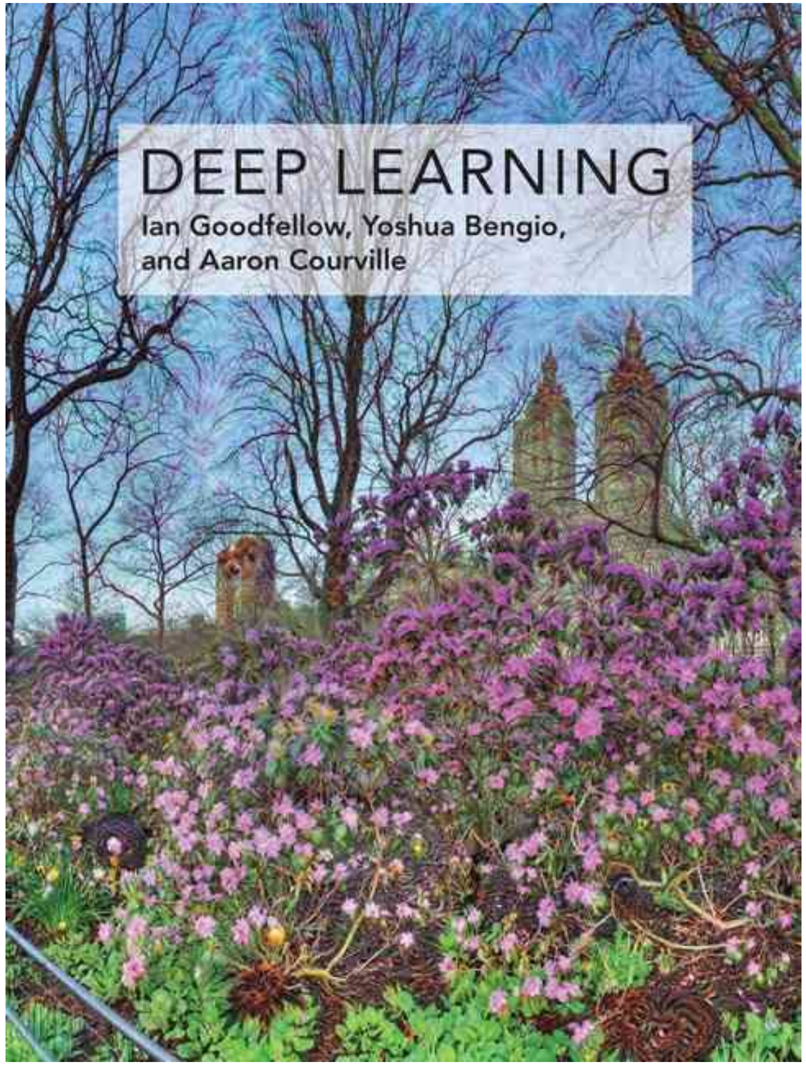
RP 2018, Marseille, 24th Sep 2018

The unstoppable rise of deep learning

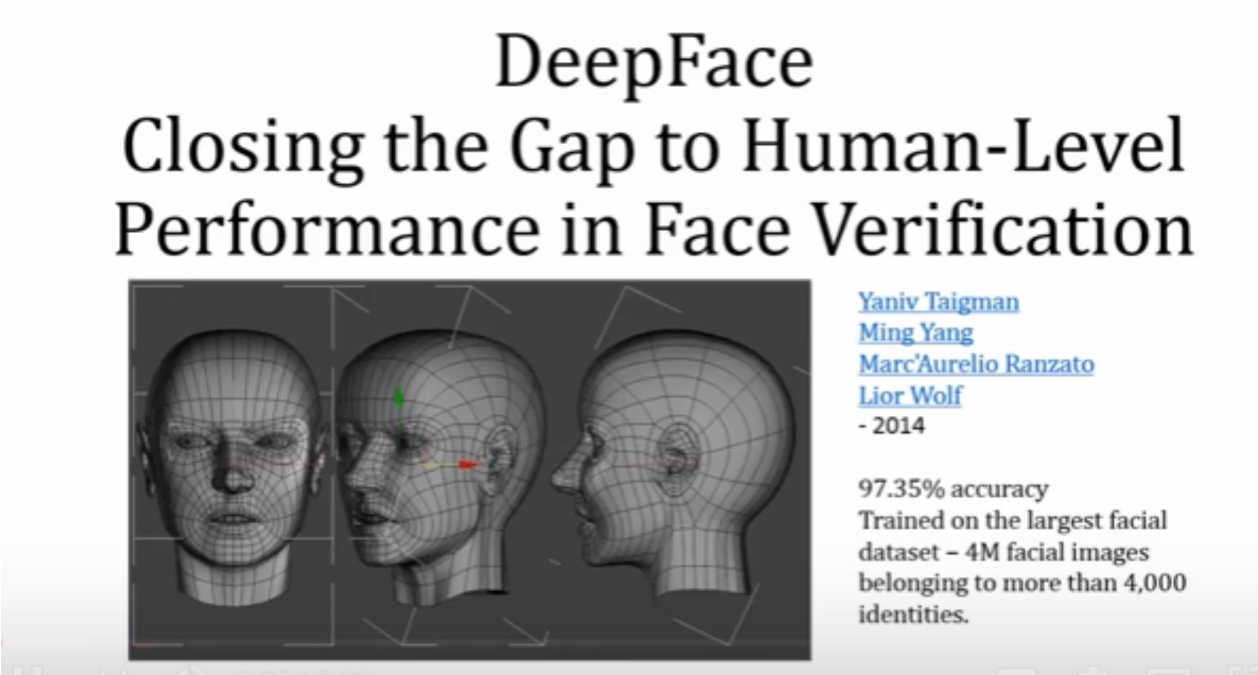
- Neural networks timeline

1940s	First proposed
1998	Convolutional nets
2006	Deep nets trained
2011	Rectifier units
2015	Vision breakthrough
2016	Win at Go

- Enabled by
  - Big data
  - Flexible, easy to build models
  - Availability of GPUs
  - Efficient inference



Much interest from tech companies,



Google Translate—here shown on a mobile phone—will use deep learning to improve its translations between texts.

...healthcare,

nature International weekly journal of science

Home | News | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For Authors

Archive > Volume 542 > Issue 7639 > Letters > Article > Article metrics > News

Article metrics for:

Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun

Nature **542**, 115–118 (02 February 2017) | doi:10.1038/nature21056

Last updated: 24 July 2017 10:10:28 EDT

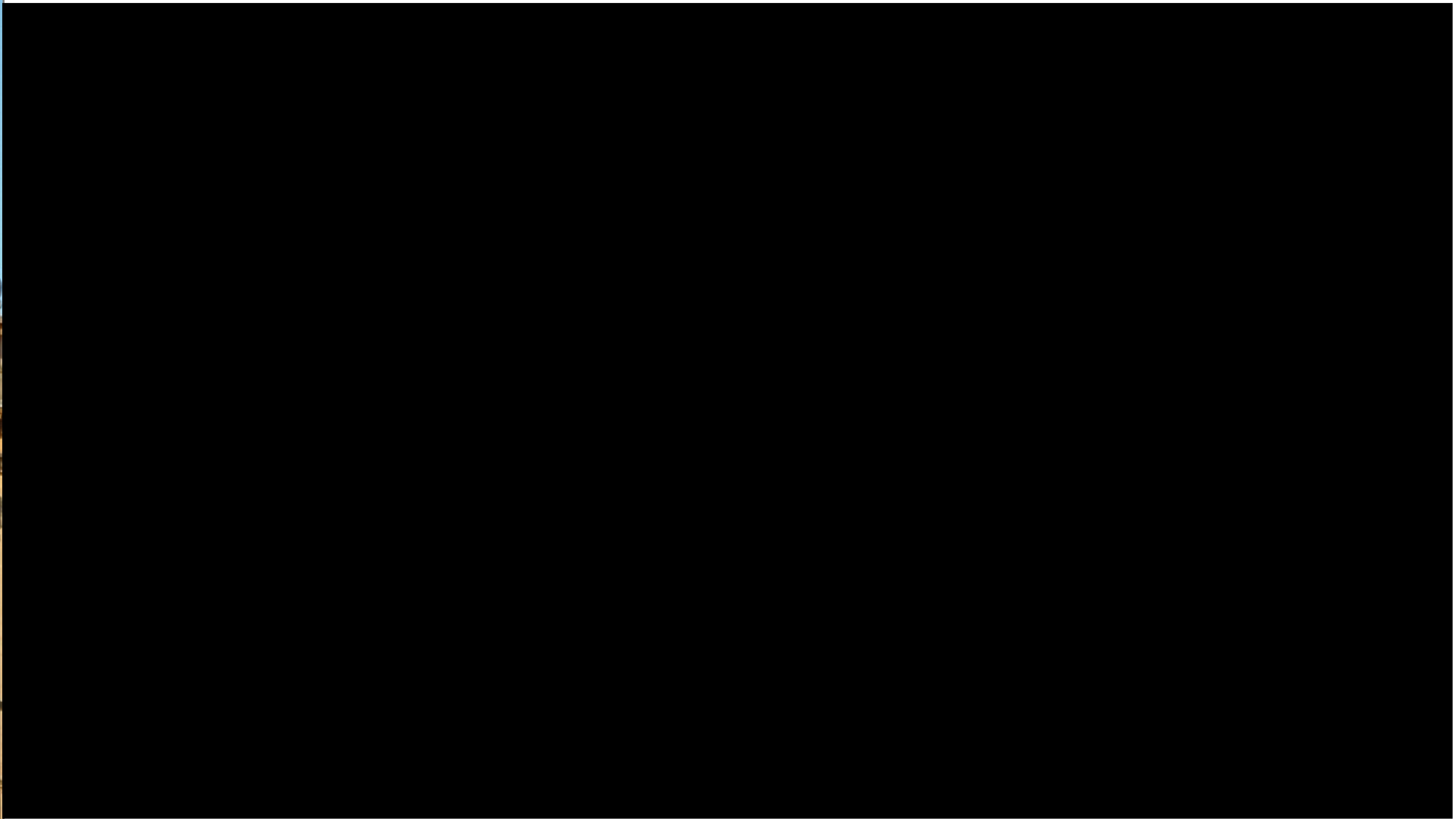
The Stanford University team said the findings were "incredibly exciting" and would now be tested in clinics.

Eventually, they believe using AI could revolutionise healthcare by turning anyone's smartphone into a cancer scanner.

Cancer Research UK said it could become a useful tool for doctors.

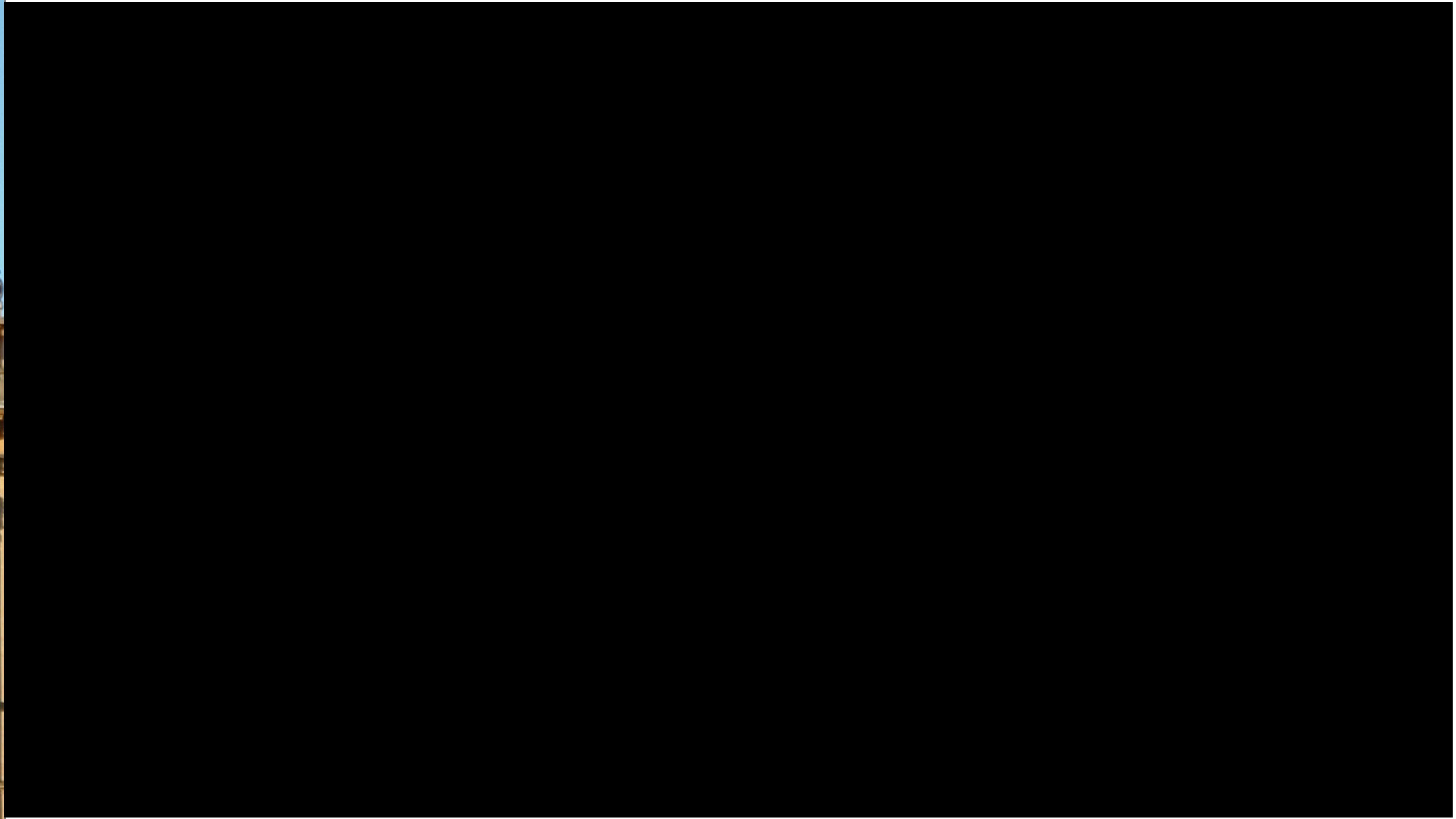
The AI was repurposed from software developed by Google that had learned to spot the difference **between images of cats and dogs**.

…and automotive industry



https://www.youtube.com/watch?v=mCmO_5ZxdvE

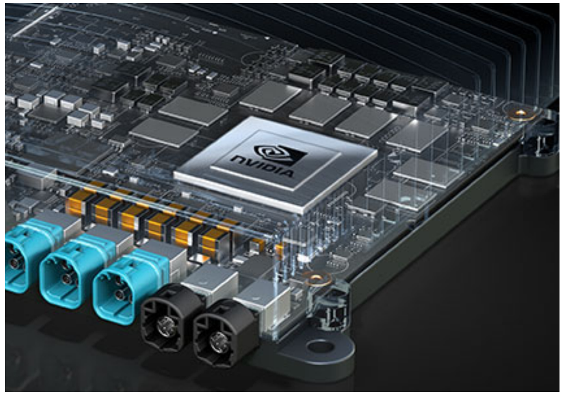
...and more



https://blogs.nvidia.com/blog/2017/01/04/bb8-ces/

What you have seen

- PilotNet by NVIDIA (regression problem)
  - end-to-end controller for self-driving cars
  - neural network
  - lane keeping and changing
  - trained on data from human driven cars
  - runs on DRIVE PX 2



- Traffic sign recognition (classification problem)
  - conventional object recognition
  - neural network solutions already planned…

- BUT
  - neural networks don’t come with rigorous guarantees!

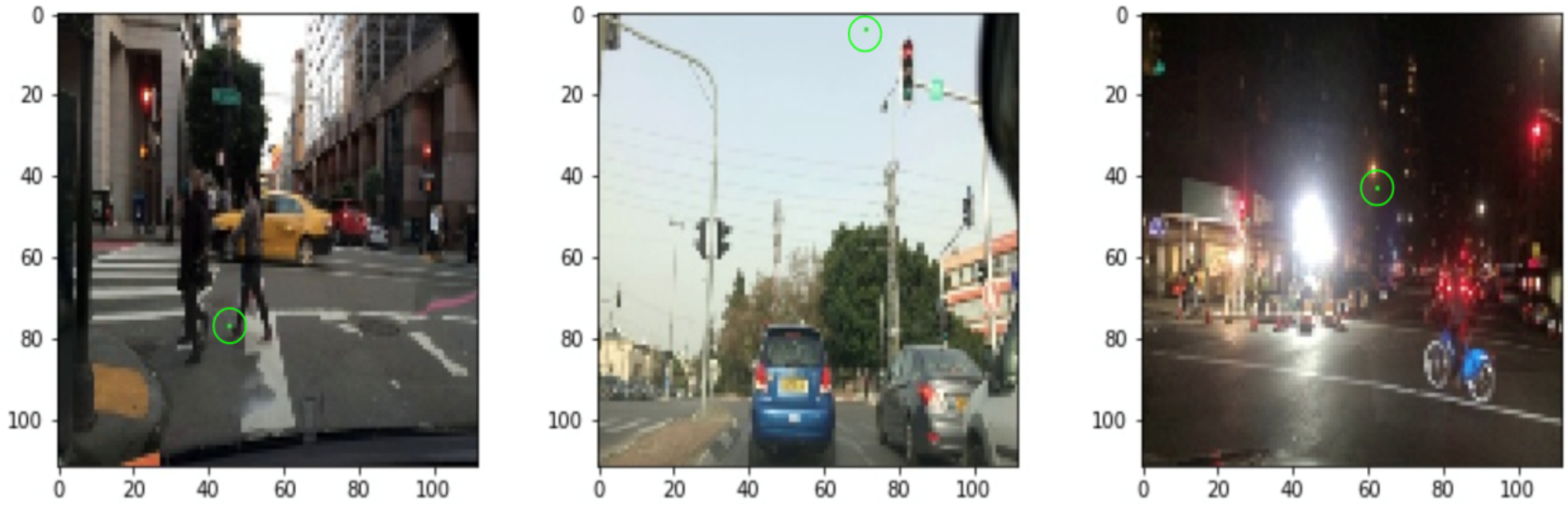
PilotNet https://arxiv.org/abs/1604.07316

What your car sees…

Original Traffic light (ImageNet class 920)	VGG16	VGG19 Misclassified	RESNET

State-of-the art deep neural networks on ImageNet

Nexar traffic sign benchmark

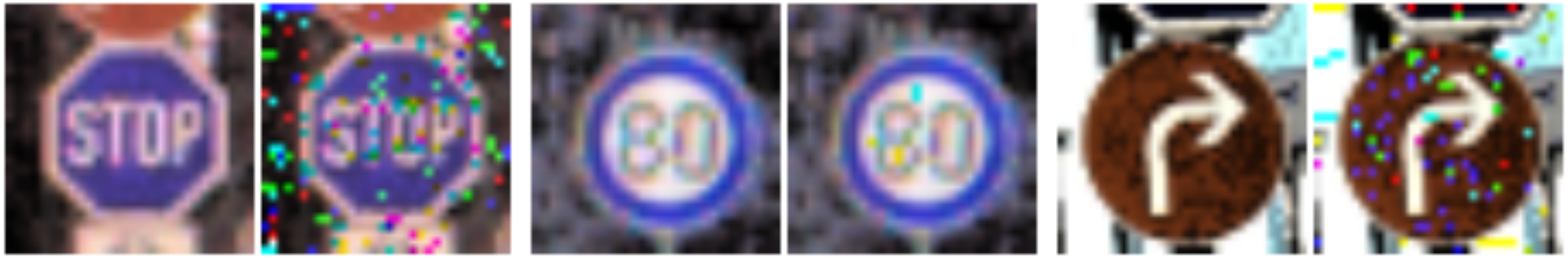


(a) (b) (c)

Red light classified as green with (a) 68%, (b) 95%, (c) 78% confidence after one pixel change.

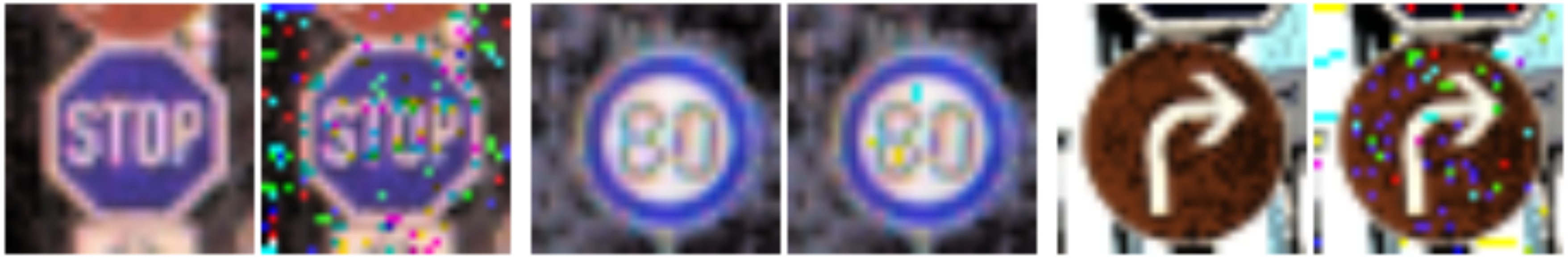
- TACAS 2018, https://arxiv.org/abs/1710.07859

German traffic sign benchmark…



stop	30m speed limit	80m speed limit	30m speed limit	go right	go straight

German traffic sign benchmark…



stop	30m speed limit	80m speed limit	30m speed limit	go right	go straight
	Confidence	0.999964	0.99		

Aren’t these artificial?

Venkat Viswanathan
@venkvis

Follow

.@TeslaMotors Model S autopilot camera misreads 101 sign as 105 speed limit at 87/101 junction San Jose. Reproduced every day this week.

4:40 AM - 15 Jul 2017

News in the last months…

Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam

Leer en español

By DAISUKE WAKABAYASHI MARCH 19, 2018

Tesla Says Crashed Vehicle Had Been on Autopilot Before Fatal Accident

By GREGORY SCHMIDT MARCH 31, 2018

RELATED COVERAGE

Tesla Looked Like the Fu
Ask if It Has One. MARC

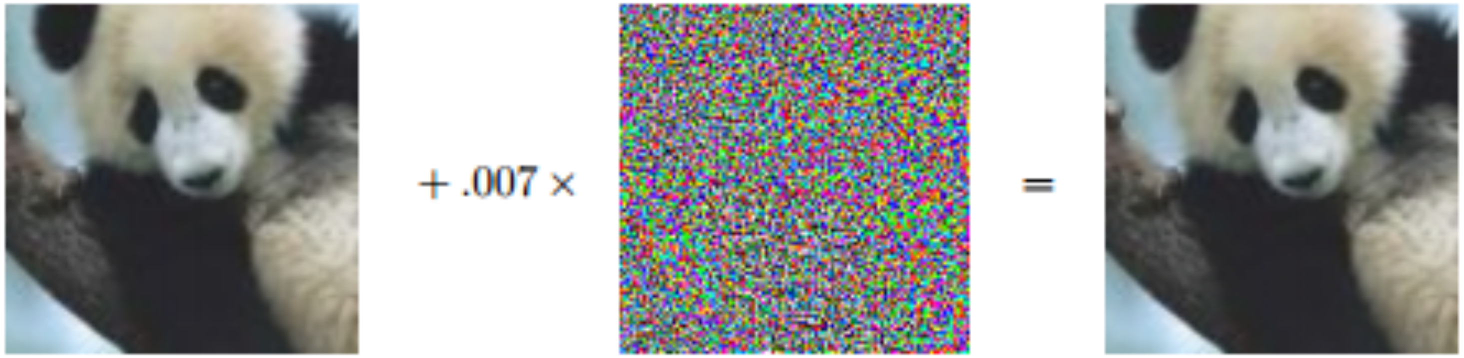
Fatal Tesla Crash Raises New Questions About Autopilot System

U.S. Safety Agency Criticizes Tesla Crash Data Release

How can this happen if we have 99.9% accuracy?

https://www.youtube.com/watch?v=B2pDFjIvrIU

Deep neural networks can be fooled!



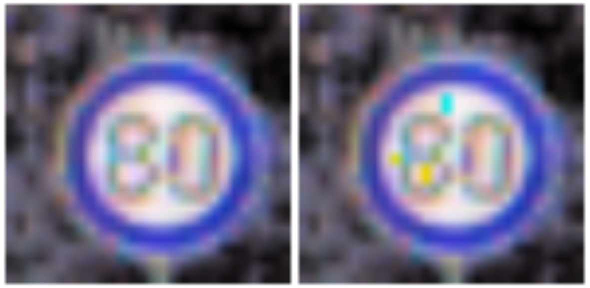
- They are unstable wrt adversarial perturbations
  - often imperceptible changes to the image [Szegedy et al 2014, Biggio et al 2013 …]
  - sometimes artificial white noise
  - practical attacks, potential security risk
  - transferable between different architectures

Risk and robustness

- Conventional learning theory
  - empirical risk minimisation [Vapnik 1991]
- Substantial growth in techniques to evaluate robustness
  - variety of robustness measures, different from risk
  - e.g. minimal expected distance to misclassification
- Methods based on optimisation or stochastic search
  - gradient sign method [Szegedy et al 2014]
  - optimisation, tool DeepFool [Moosavi-Desfooli et al 2016]
  - constraint-based, approximate [Bastani et al 2016]
  - adversarial training with cleverhans [Papernot et al 2016]
  - universal adversarial example [Moosavi-Desfooli et al 2017]

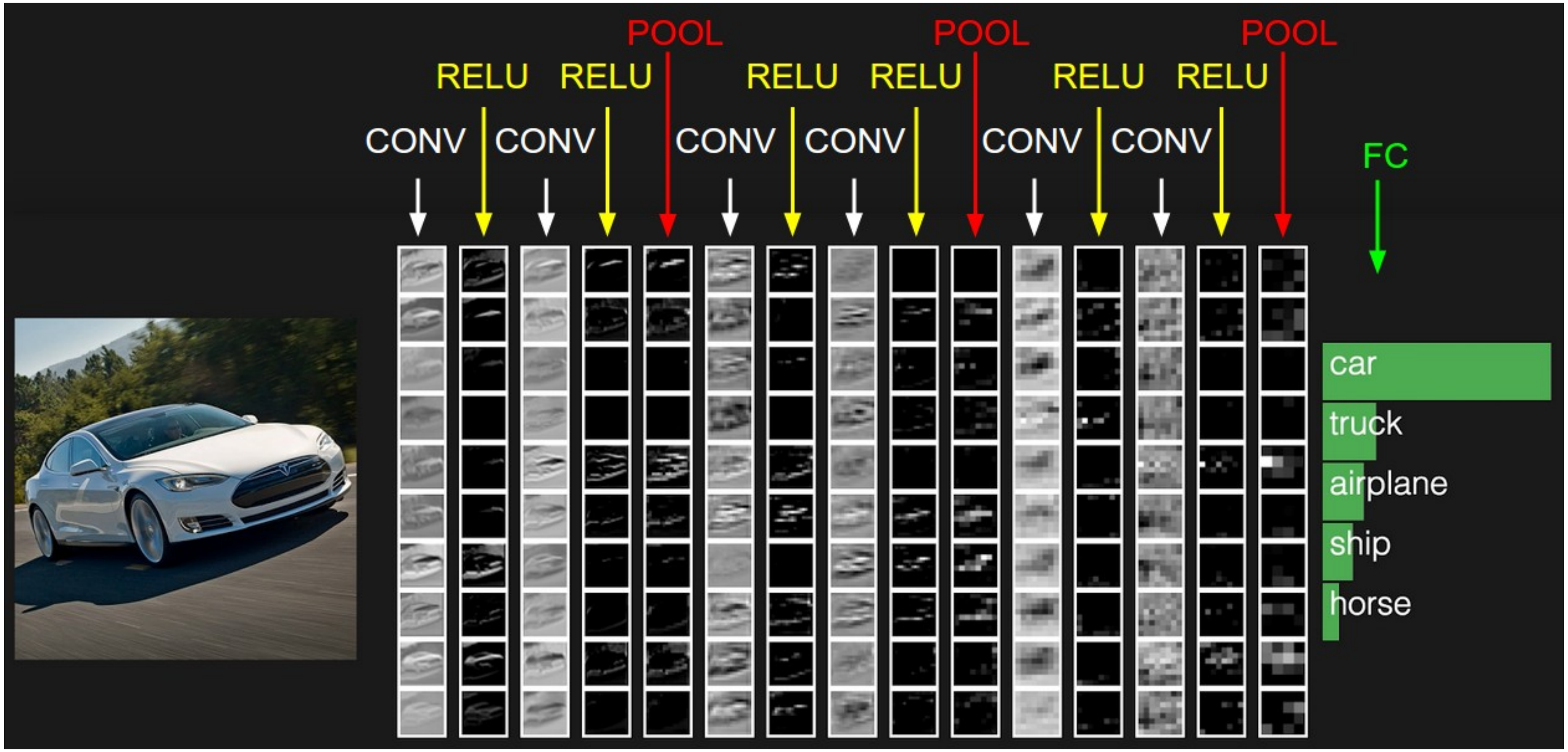
This talk

- First steps towards methodology to ensure safety of classification decisions
  - visible and human-recognisable perturbations: change of camera angle, snow, sign imperfections, ...
  - should not result in class changes
  - focus on individual decisions
  - images, but can be adapted to other types of problems
  - e.g. networks trained to produce justifications, in addition to classification (explainable AI)



- Towards an automated verification framework
  - search+MCTS: CAV 2017, https://arxiv.org/abs/1610.06940
  - global opt: IJCAI 2018, https://arxiv.org/abs/1805.02242
  - SIFT+game: TACAS 2018, https://arxiv.org/abs/1710.07859

Deep feed-forward neural network



Convolutional multi-layer network

http://cs231n.github.io/convolutional-networks/#conv

Problem setting

- Assume
  - vector spaces D_{L0}, D_{L1}, …, D_{Ln}, one for each layer
  - $f : D_{L0} \rightarrow \{c_1,\ldots c_k\}$ classifier function modelling human perception ability

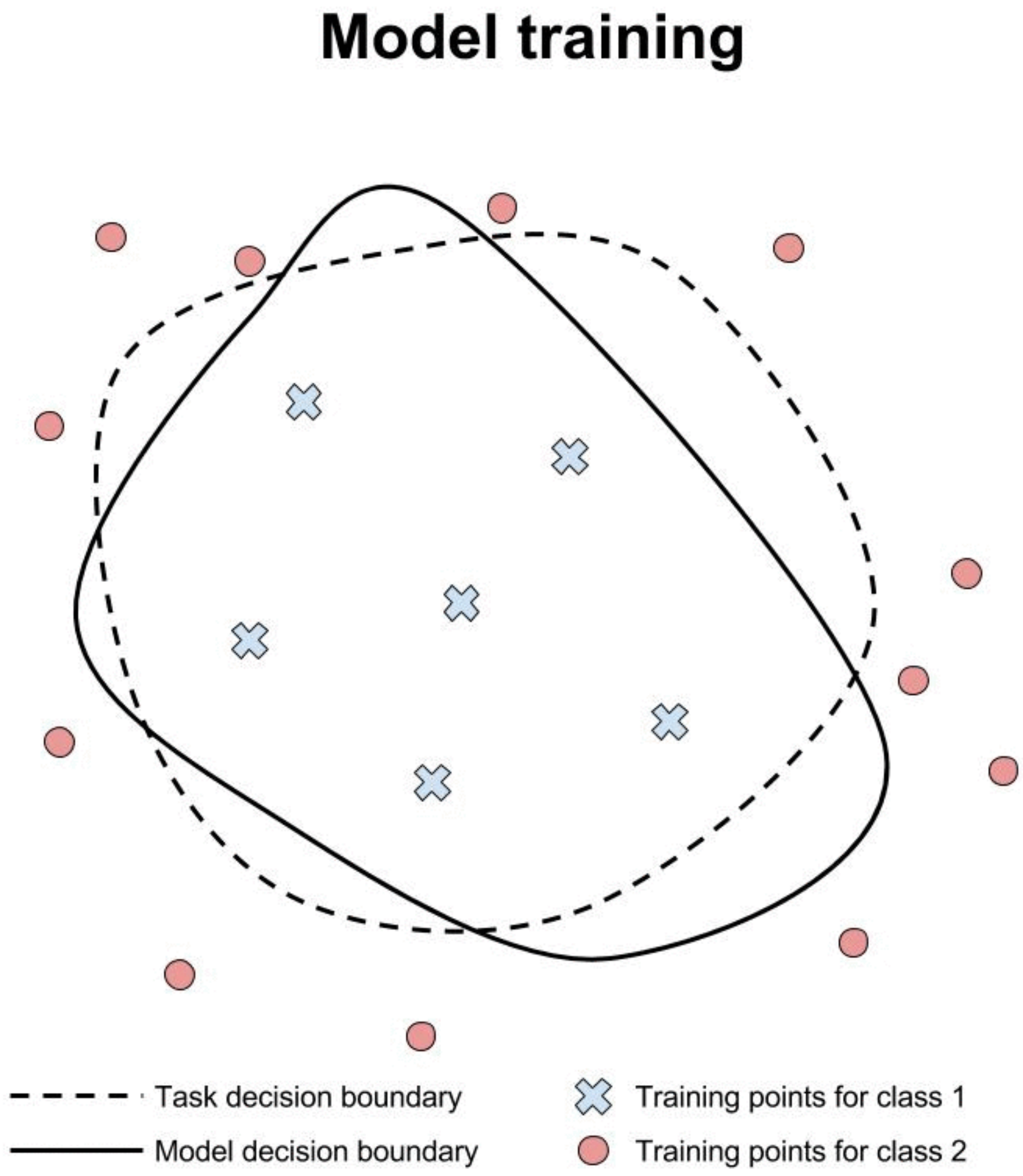
- The network $f' : D_{L0} \rightarrow \{c_1,\ldots c_k\}$ approximates f from M training examples $\{(x_i,c_i)\}_{i=1..M}$
  - built from activation functions ϕ_0, ϕ_1, …, ϕ_n, one for each layer
  - for point (image) $x \in D_{L0}$, its activation in layer k is

    $$\alpha_{x,k} = \phi_k(\phi_{k-1}(\ldots\phi_1(x)))$$

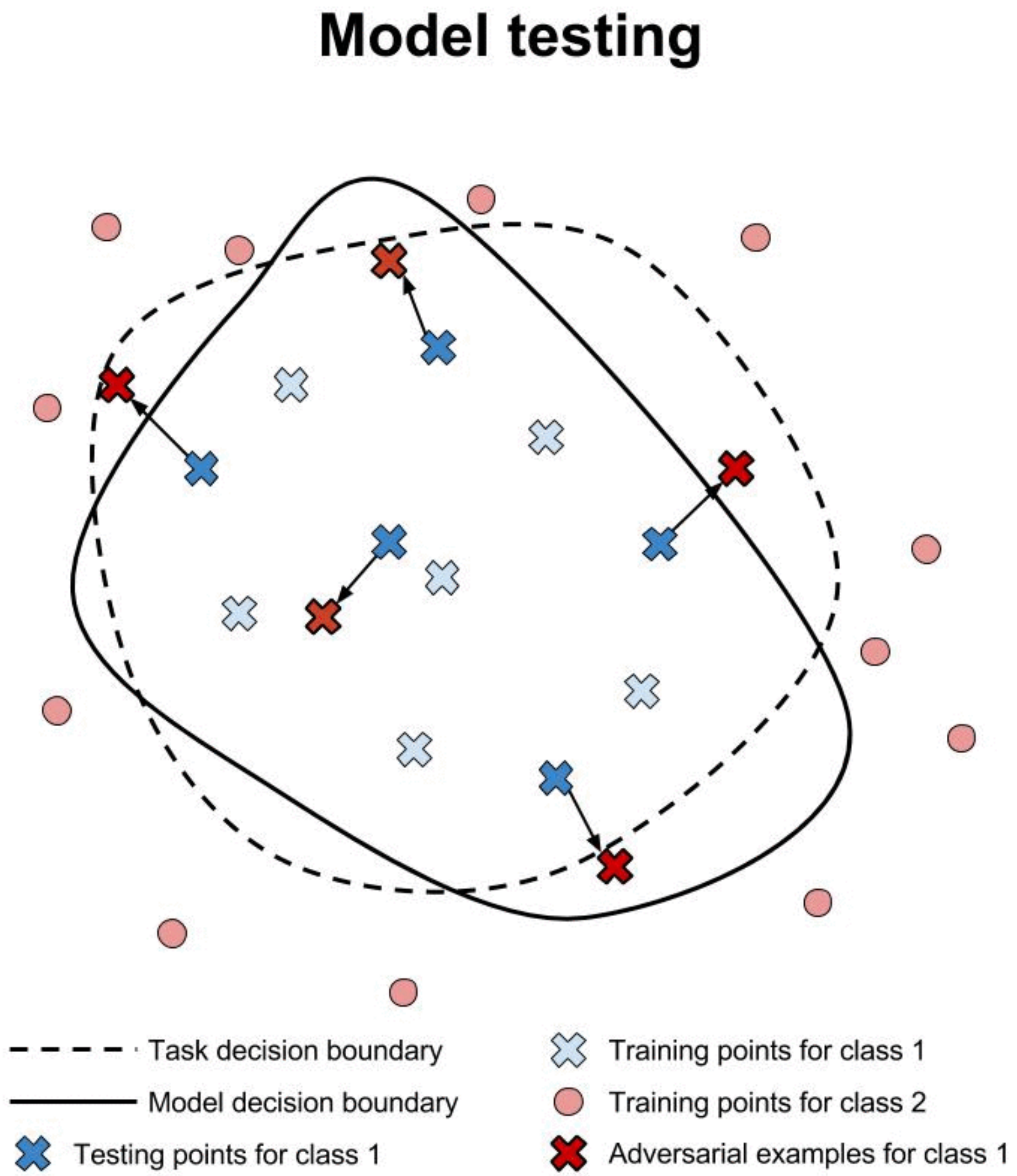
  - where $\phi_k(x) = \sigma(xW_k+b_k)$ and $\sigma(x) = \max(x,0)$
  - W_k learnable weights, b_k bias, σ ReLU

- Notation
  - overload $\alpha_{x,n} = \alpha_{y,n}$ to mean x and y have the same class

Training vs testing



Training vs testing



Robustness

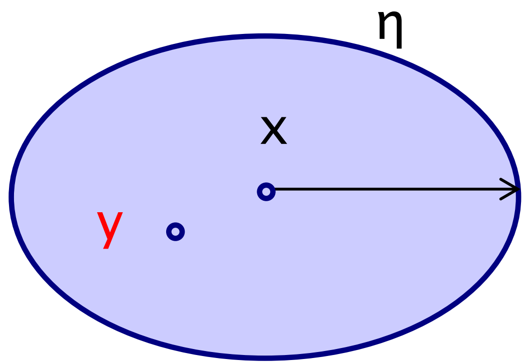
- Regularisation such as dropout improves smoothness
- Common smoothness assumption
  - each point $x \in D_{L0}$ in the input layer has a region η around it such that all points in η classify the same as x
- Pointwise robustness [Szegedy et al 2014]
  - f’ is not robust at point x if $\exists y \in \eta$ such that $f'(x) \neq f'(y)$
- Robustness (network property)
  - smallest perturbation weighted by input distribution
  - reduced to non-convex optimisation problem

Verification for neural networks

- Little studied
- Reduction of safety to Boolean combination of linear arithmetic constraints [Pulina and Tachela 2010]
  - encode entire network using constraints
  - approximate the sigmoid using piecewise linear functions
  - SMT solving, does not scale (6 neurons, 3 hidden)
- Reluplex [Barrett et al 2017]
  - similar encoding but for ReLU, rather than sigmoid
  - generalise Simplex, SMT solver
  - more general properties
  - successful for end-to-end controller networks with 300 nodes

Safety of classification decisions

- Safety assurance process is complex
- Here focus on safety at a point as part of such a process
  - consider region supporting decision at point x
  - same as pointwise robustness…



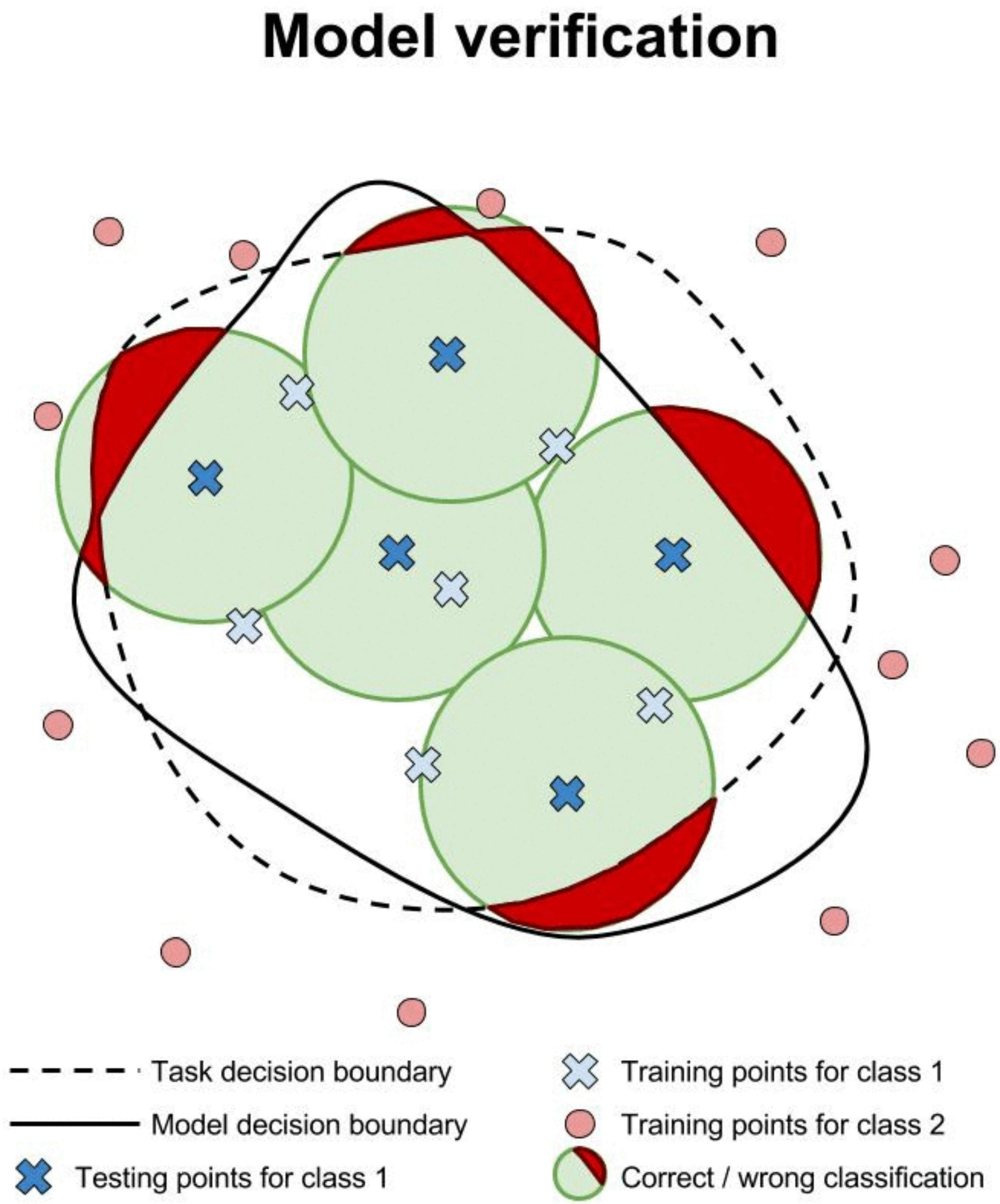
- But..
  - what diameter for region η?
  - which norm? L^2, L^{sup}?
  - what is an acceptable/adversarial perturbation?

- Introduce the concept of manipulation, a family of operations that perturb an image
  - think of scratches, weather conditions, camera angle, etc
  - classification should be invariant wrt safe manipulations

Safety verification

- Take as a specification set of manipulations and region η
  - work with pointwise robustness as a safety criterion
  - focus on safety wrt a set of manipulations
  - exhaustively search the region for misclassifications
- Challenges
  - high dimensionality, nonlinearity, infinite region, huge scale
- Automated verification (= ruling out adversarial examples)
  - need to ensure finiteness of search
  - guarantee of decision safety if adversarial example not found
- Falsification (= searching for adversarial examples)
  - good for attacks, no guarantees

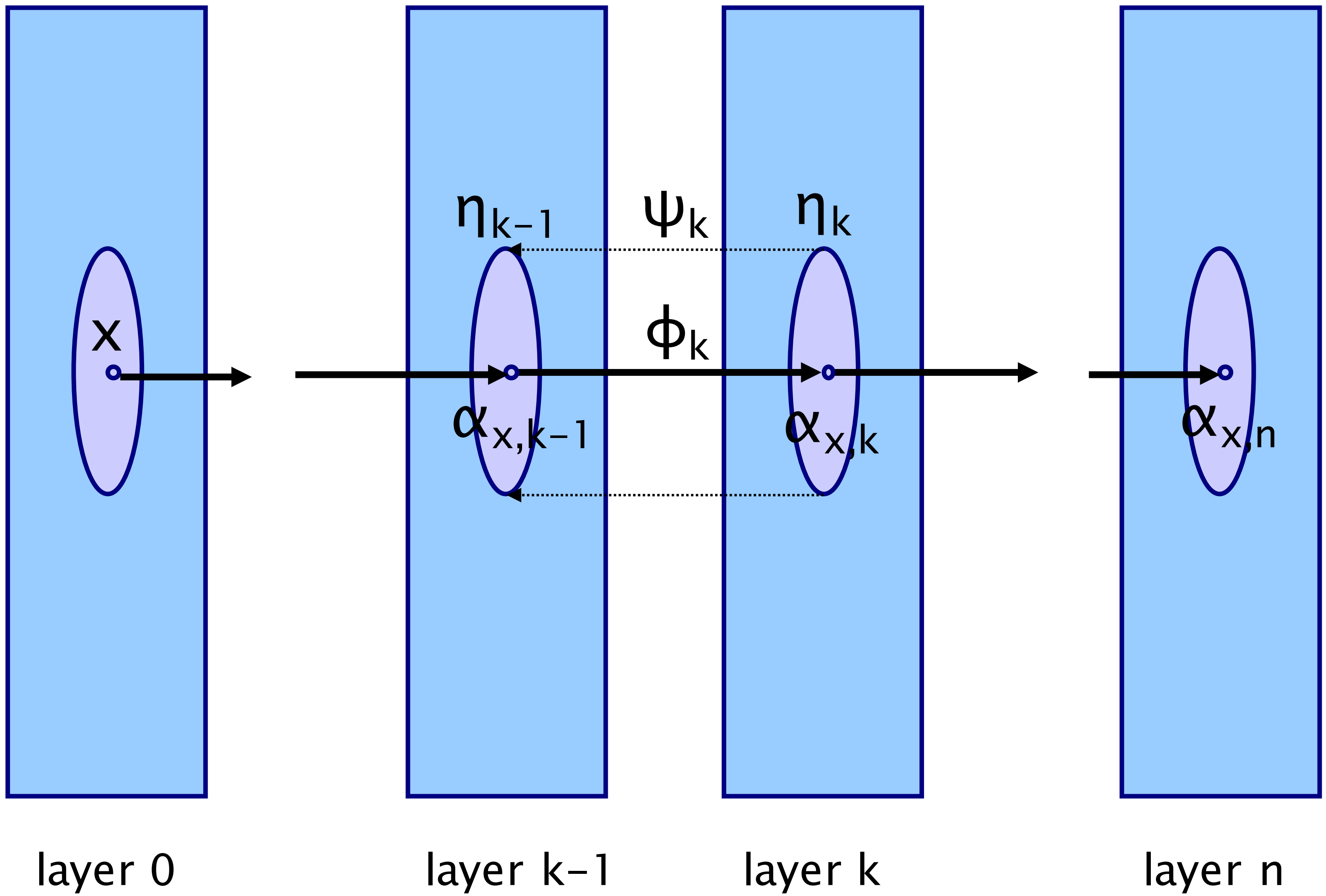
Training vs testing vs verification



Verification framework

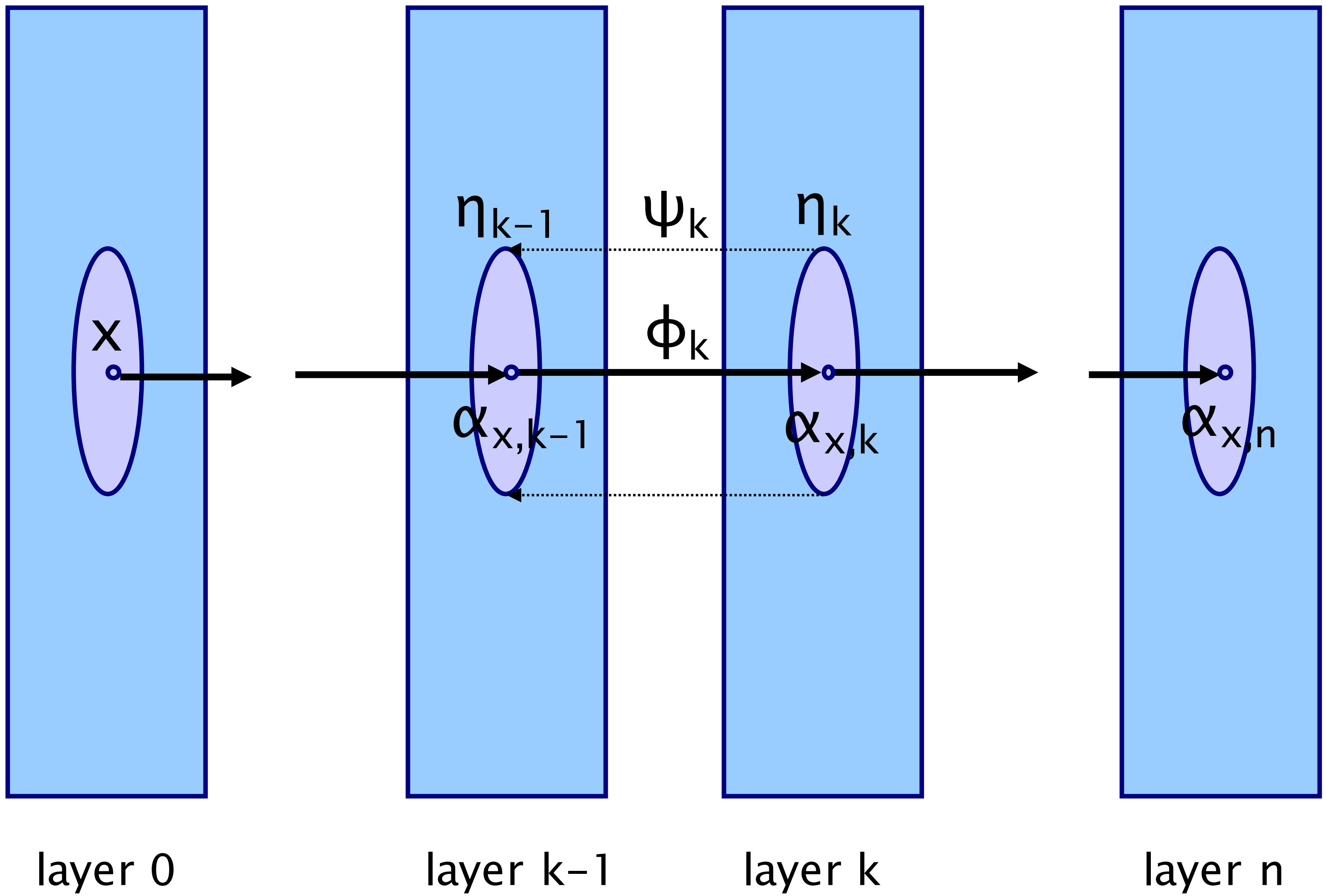
- Size of the network is prohibitive
  - millions of neurons!
- The crux of our approach
  - propagate verification layer by layer, i.e. need to assume for each activation $\alpha_{x,k}$ in layer k there is a region $\eta(\alpha_{x,k})$
  - dimensionality reduction by focusing on features
- This differs from heuristic search for adversarial examples
  - nonlinearity implies need for approximation using convex optimisation
  - no guarantee of precise adversarial examples
  - no guarantee of exhaustive search even if we iterate

Multi-layer (feed-forward) neural network



- Require mild conditions on region η_k and ψ_k mappings

Mapping forward and backward



- Map region $\eta_k(\alpha_{x,k})$ forward via ϕ_k, backward via inverse ψ_k

Manipulations

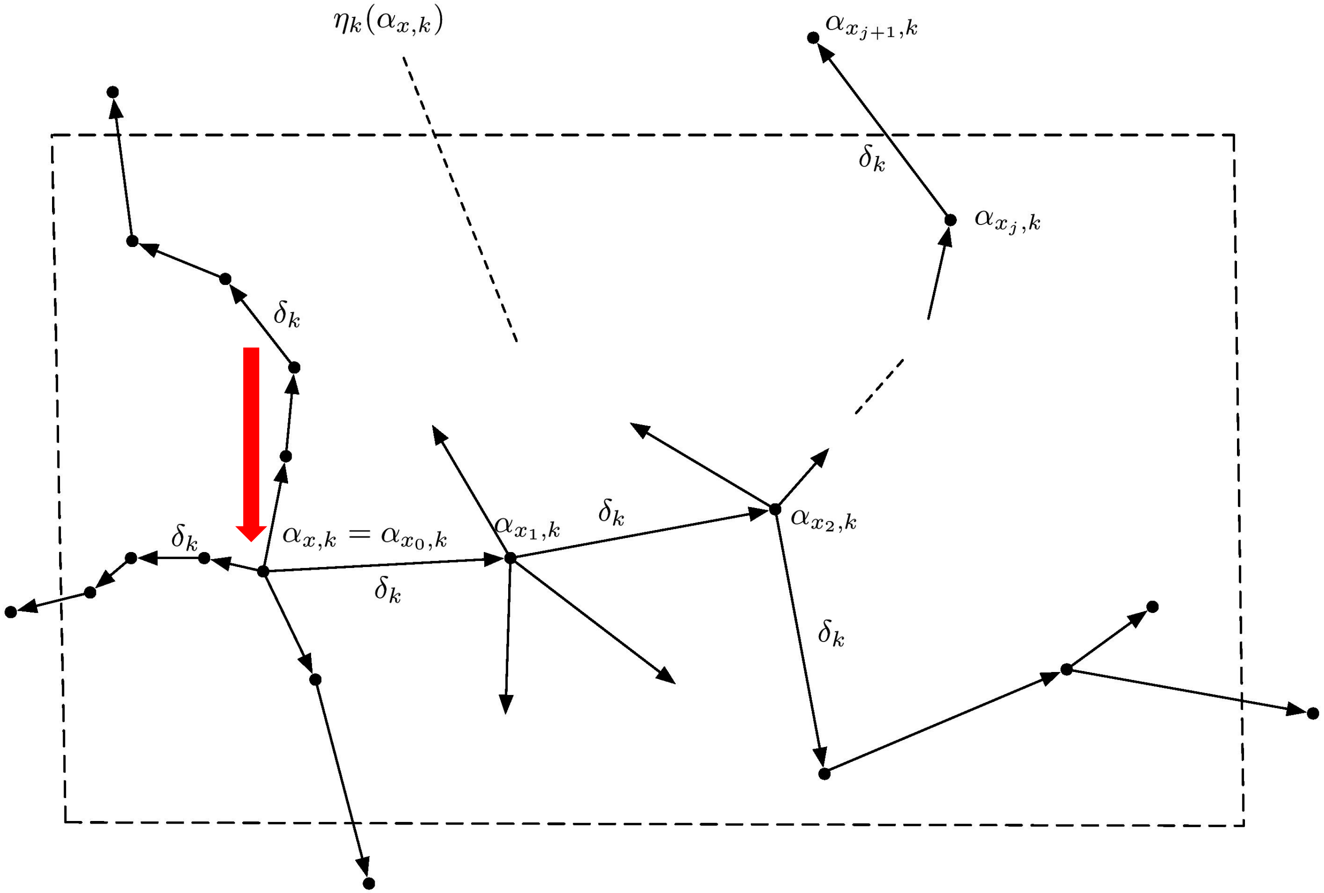
- Consider a family Δ_k of operators $\delta_k : D_{Lk} \rightarrow D_{Lk}$ that perturb activations in layer k, incl. input layer
  - think of scratches, weather conditions, camera angle, etc
  - classification should be invariant wrt such manipulations
- Intuitively, safety of network N at a point x wrt the region $\eta_k(\alpha_{x,k})$ and set of manipulations Δ_k means that perturbing activation $\alpha_{x,k}$ by manipulations from Δ_k will not result in a class change
- Note that manipulations can be
  - defined by user and wrt different norms
  - made specific to each layer, and
  - applied directly on features, i.e. subsets of dimensions

Ensuring region coverage

- Fix point x and region $\eta_k(\alpha_{x,k})$
- Want to perform exhaustive search of the region for adversarial manipulations
  - if found, use to fine-tune the network and/or show to human tester
  - else, declare region safe wrt the specified manipulations

- Methodology: reduce to counting of misclassifications
  - discretise the region
  - cover the region with ‘ladders’ that are complete and covering
  - show 0-variation, i.e. explore nondeterministically and iteratively all paths in the tree of ladders, counting the number of misclassifications after applying manipulations
  - search is exhaustive under assumption of minimality of manipulations, e.g. unit steps

Covering region with ‘ladders’



- NB related work considers approximate, deterministic and non-iterative manipulations that are not covering
- Can search single or multiple paths (Monte Carlo tree search)

Layer-by-layer analysis

- In deep neural networks linearity increases with deeper layers
- Naïve search intractable: work with features

- Propagate analysis, starting from a given layer k:
- Determine region $\eta_k(\alpha_{x,k})$ from region $\eta_{k-1}(\alpha_{x,k-1})$
  - map forward using activation function
  - NB each activation at layer k arises from a subset of dimensions at layer k-1
  - check forward/backward mapping conditions (SMT-expressible)
- Refine manipulations in Δ_{k-1}, yielding Δ_k
  - consider more points as the analysis progresses into deeper layers
- If safety wrt $\eta_k(\alpha_{x,k})$ and Δ_k is verified, continue to layer k+1, else report adversarial example

Layer-by-layer analysis

- Framework ensures that safety wrt $\eta_k(\alpha_{x,k})$ and Δ_k implies safety wrt $\eta_{k-1}(\alpha_{x,k-1})$ and Δ_{k-1}
- If manipulations are minimal, then can deduce safety (= pointwise robustness) of the region at x
- But adversarial examples at layer k can be spurious, i.e. need to check if they are adversarial examples at the input layer
- NB employ various heuristics for scalability
  - explore manipulations of a subset of most extreme dimensions, which encode more explicit knowledge
  - employ additional precision parameter to avoid overly small spans

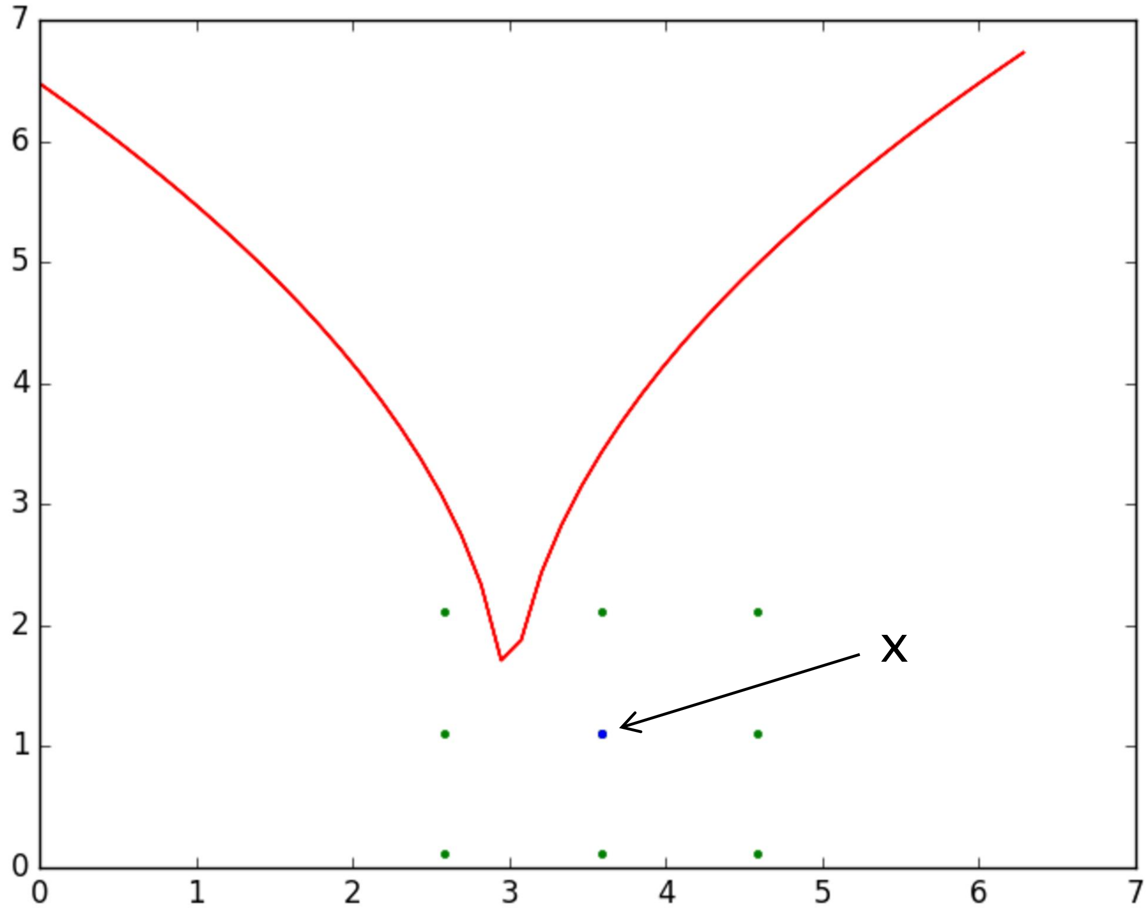
Features

- The layer-by-layer analysis is finite, but regions $\eta_k(\alpha_{x,k})$ are high-dimensional
  - exhaustive analysis impractical, need heuristics…
- We exploit decomposition into features, assuming their independence and low-dimensionality
  - natural images form high-dimensional tangled manifold, which embeds tangled manifolds that represent features
  - classifiers separate these manifolds
- By assuming independence of features, reduce problem of size $O(2^{d1+..+dn})$ to set of smaller problems $O(2^{d1}),\ldots O(2^{dn})$
  - e.g. compute regions and 0-variation wrt to features
  - analysis discovers features automatically through hidden layer analysis

Implementation

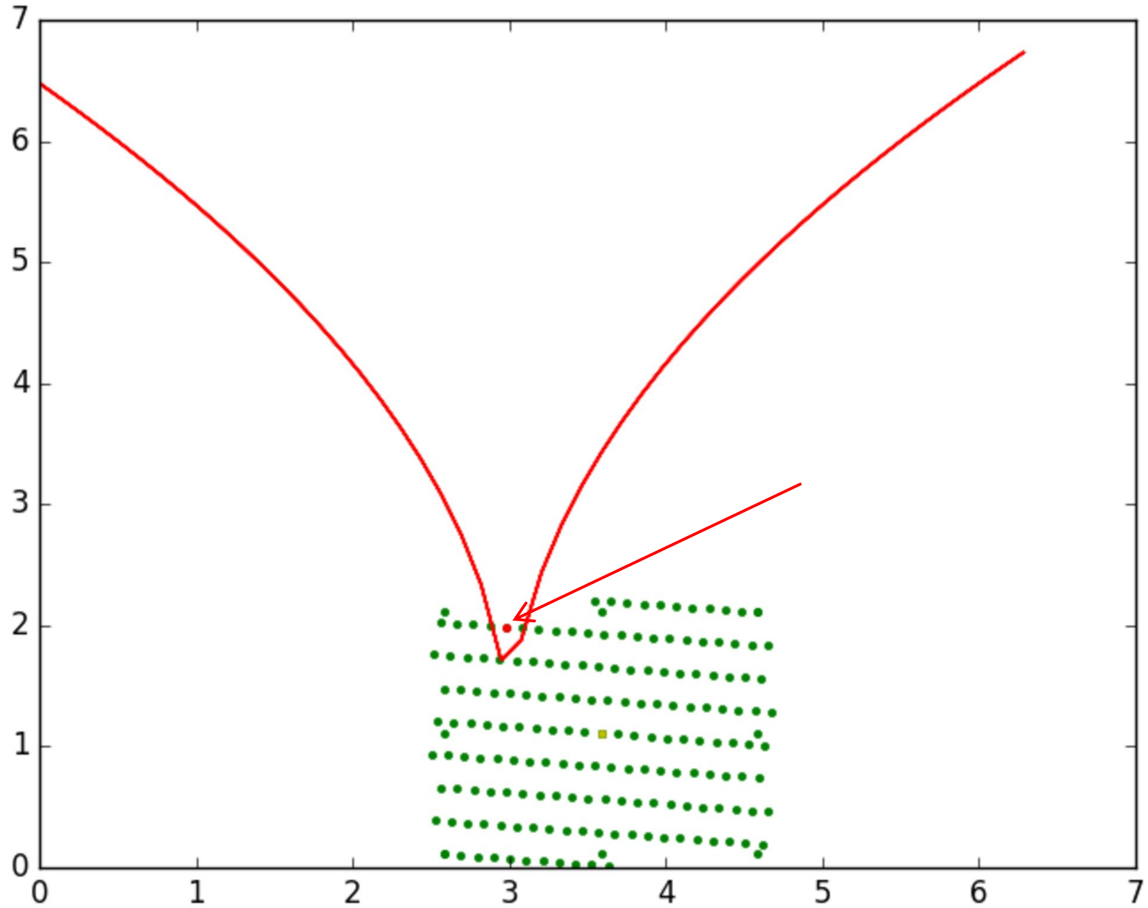
- Implement the techniques using SMT (Z3)
  - for layer-by-layer analysis, use linear real arithmetic with existential and universal quantification
  - within the layer (0-variation), use as above but without universal quantification
  - work with Euclidean and Manhattan norms, can be adapted to other norms
- We work with one point/decision at a time, rather than activation functions, but computation is exact
  - avoid approximating sigmoid (not scalable) [Pulina et al 2010]
  - more scalable than approximating ReLU by LP [Bastani et al 2016] or Reluplex [Barrett et al 2017]
- Main challenge: how to define meaningful regions and manipulations
  - but adversarial examples can be found quickly

Example: input layer



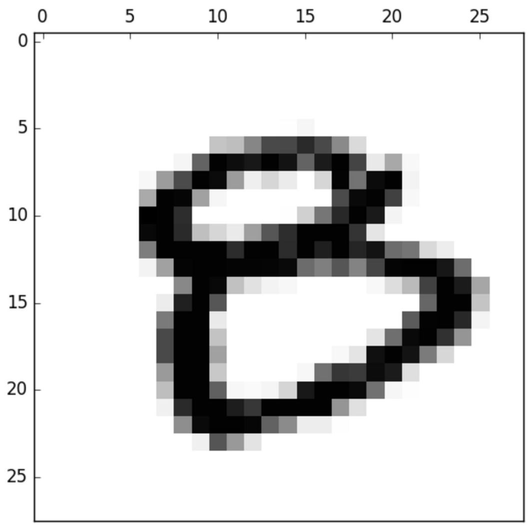
- Small point classification network, 8 manipulations

Example: 1st hidden layer

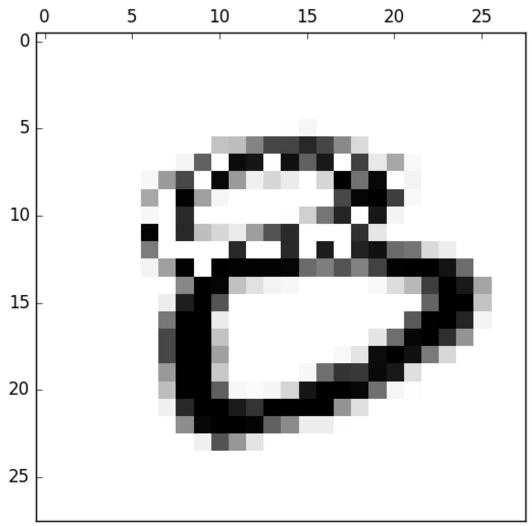


- Refined manipulations, adversarial example found

MNIST example



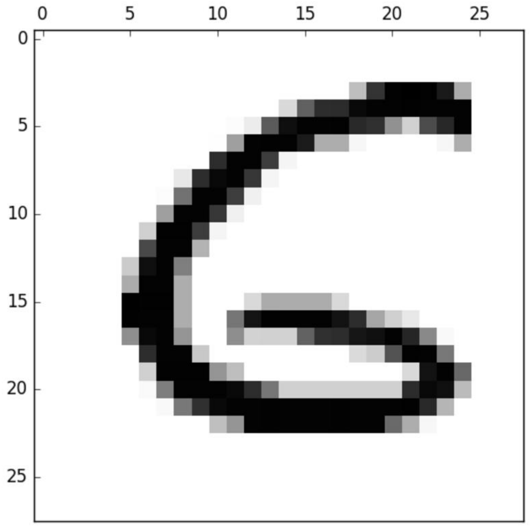
8



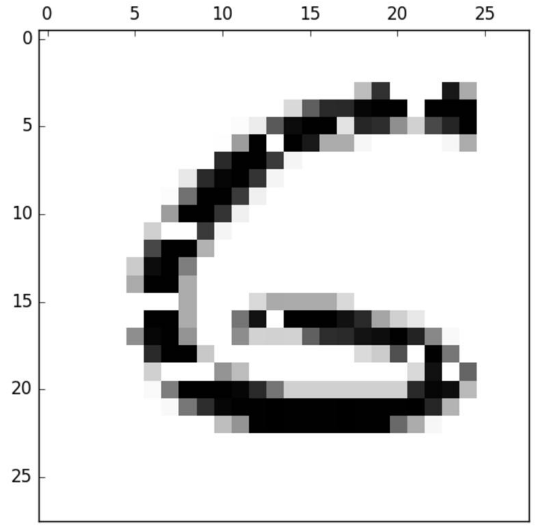
0

- 28x28 image size, one channel, medium size network (12 layers, Conv, ReLU, FC and softmax)

Another MNIST example



6



5

- 28x28 image size, one channel, medium size network (12 layers, Conv, ReLU, FC and softmax)

Compare to existing methods

- Search for adversarial perturbations only (=falsification)

- FGSM [Goodfellow et al 2014]
  - calculates optimal attack for a linear approximation of network cost, for a set of images
  - deterministic, iterative manipulations

- JSMA [Papernot et al 2015]
  - finds subset of dimensions to manipulate (in the input layer)
  - manipulates according to partial derivatives

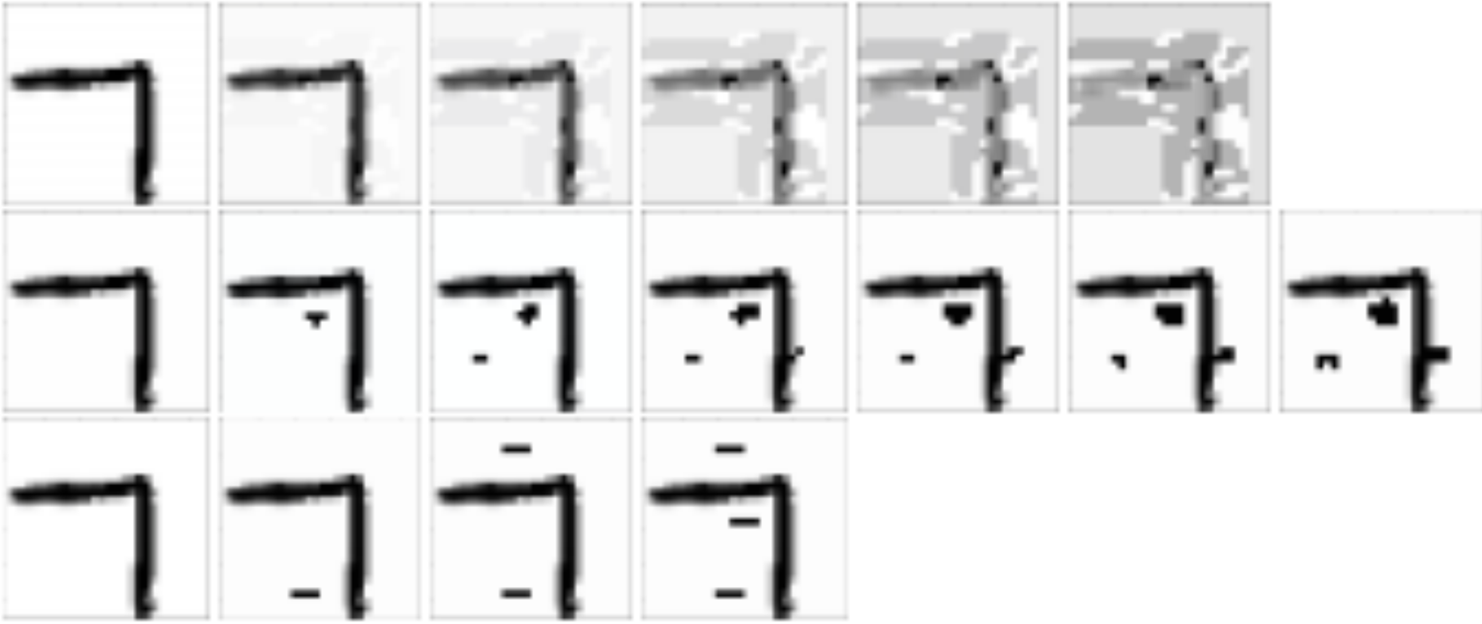
- DLV (this talk)
  - explores proportion of dimensions in input and hidden layers
  - so manipulates over features discovered in hidden layers

Falsification comparison

FGSM 9

JSMA 3

DLV 3



- DLV able to find examples with smaller average distance than JSMA, at comparable performance (may affect transferability)
- FGSM fastest per image
- For high success rates (approx 98%) JSMA has smallest average distance, followed by DLV, followed by FGSM

CIFAR-10 example

ship

ship

truck

- 32x32 image size, 3 channels, medium size network (Conv, ReLU, Pool, FC, dropout and softmax)
- Working with 1st hidden layer, project back to input layer

ImageNet example

Street sign

Birdhouse

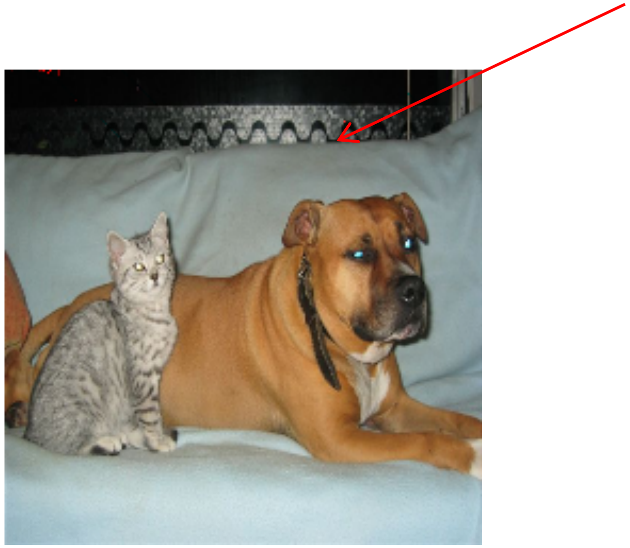
- 224x224 image size, 3 channels, 16 layers, state-of-the-art network VGG, (Conv, ReLU, Pool, FC, zero padding, dropout and softmax)
- Work with 20,000 dimensions (of 3m), unsafe for 2nd layer

ImageNet example

- 224x224 image size, 3 channels, 16 layers, state-of-the-art network VGG, (Conv, ReLU, Pool, FC, zero padding, dropout and softmax)
- Reported safe for 20,000 dimensions

Another ImageNet example

Boxer



Rhodesian ridgeback

- 224x224 image size, 3 channels, 16 layers, state-of-the-art network, (Conv, ReLU, Pool, FC, zero padding, dropout and softmax)
- Work with 20,000 dimensions

Yet another ImageNet example

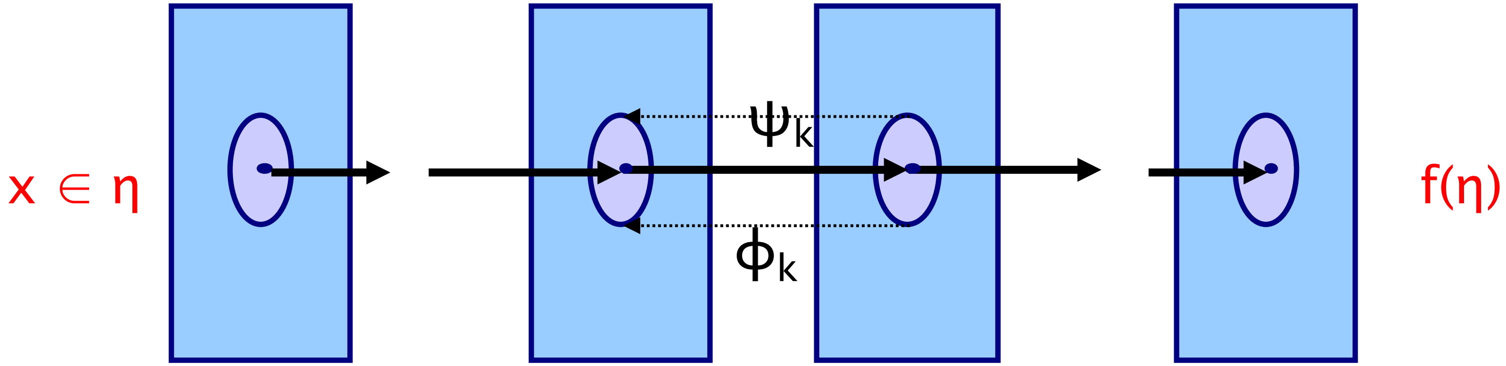
Labrador retriever

Lifeboat

- 224x224 image size, 3 channels, 16 layers, state-of-the-art network, (Conv, ReLU, Pool, FC, zero padding, dropout and softmax)
- Work with 20,000 dimensions

Alternative approach: reachability analysis

- Instead of relying on exhaustive search of discretized region,
- can we compute the reachable region?



- Under assumption of Lipschitz continuity
  - reduce to computing upper and lower bounds via global optimisation
  - yields provable guarantees: best and worst case confidence values
- Method NP-complete
  - wrt the number of input dimensions, not number of neurons
- IJCAI 2018, https://arxiv.org/abs/1805.02242

Lipschitz networks

- Lipschitz continuity limits the rate of change of outputs as inputs change

- In fact, all layers of e.g. image classification networks are Lipschitz continuous:
  - convolutional with ReLU activation functions
  - fully connected with ReLU activation functions
  - max pooling
  - contrast normalisation
  - softmax
  - sigmoid
  - hyperbolic tangent

Lipschitz continuity reminder

Given two metric spaces (X, d_X) and (Y, d_Y), where d_X and d_Y are the metrics on the sets X and Y respectively, a function $f : X \to Y$ is called *Lipschitz continuous* if there exists a real constant $K \geq 0$ such that, for all $x_1, x_2 \in X$:

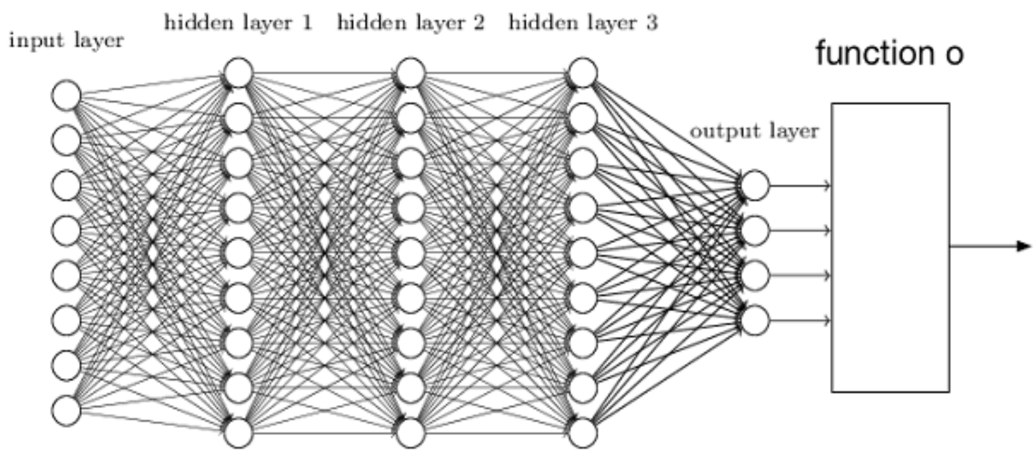
$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2) \quad (1)$$

K is called the *Lipschitz constant* for the function f. The smallest K is called *the Best Lipschitz constant*, denoted as K_{best}.

Reachability analysis: intuition

Let $o : [0,1]^m \to \mathbb{R}$ be a Lipschitz continuous function statistically evaluating the outputs of the network.

Connect the network f with function o, i.e., o(f(x))



Reachability analysis: generic definition

Let $X' \subseteq [0,1]^n$ be an input subspace and $f : \mathbb{R}^n \to \mathbb{R}^m$ a network. The reachability of f over the function o under an error tolerance $\epsilon \geq 0$ is a set $R(o, X', \epsilon) = [l, u]$ such that

$$l \geq \inf_{x' \in X'} o(f(x')) - \epsilon \text{ and } u \leq \sup_{x' \in X'} o(f(x')) + \epsilon. \tag{2}$$

We write $u(o, X', \epsilon) = u$ and $l(o, X', \epsilon) = l$ for the upper and lower bound, respectively, and let the reachability diameter be

$$D(o, X', \epsilon) = u(o, X', \epsilon) - l(o, X', \epsilon)$$

Reachability analysis: problem types

- Generic formulation, parameterised by the statistics function o: $[0,1]^m \rightarrow R$
- Aim to compute lower and upper bounds [l, u]
- By instantiating the function o, we can obtain several known problems
  - output range analysis
  - safety verification: upper bound the difference between confidence for an input and largest confidence value for any other class by 0
  - robustness comparison

One-dimensional case

We define the following lower-bound function.

$$\begin{aligned} h(x,y) &= w(y) - K|x-y| \\ H(x;\mathcal{Y}_i) &= \max_{y\in\mathcal{Y}_i} \quad w(y) - K|x-y| \end{aligned} \tag{7}$$

where $K > K_{best}$ is a Lipschitze constant of w and $H(x;\mathcal{Y}_i)$ intuitively represents the lower-bound sawtooth function.

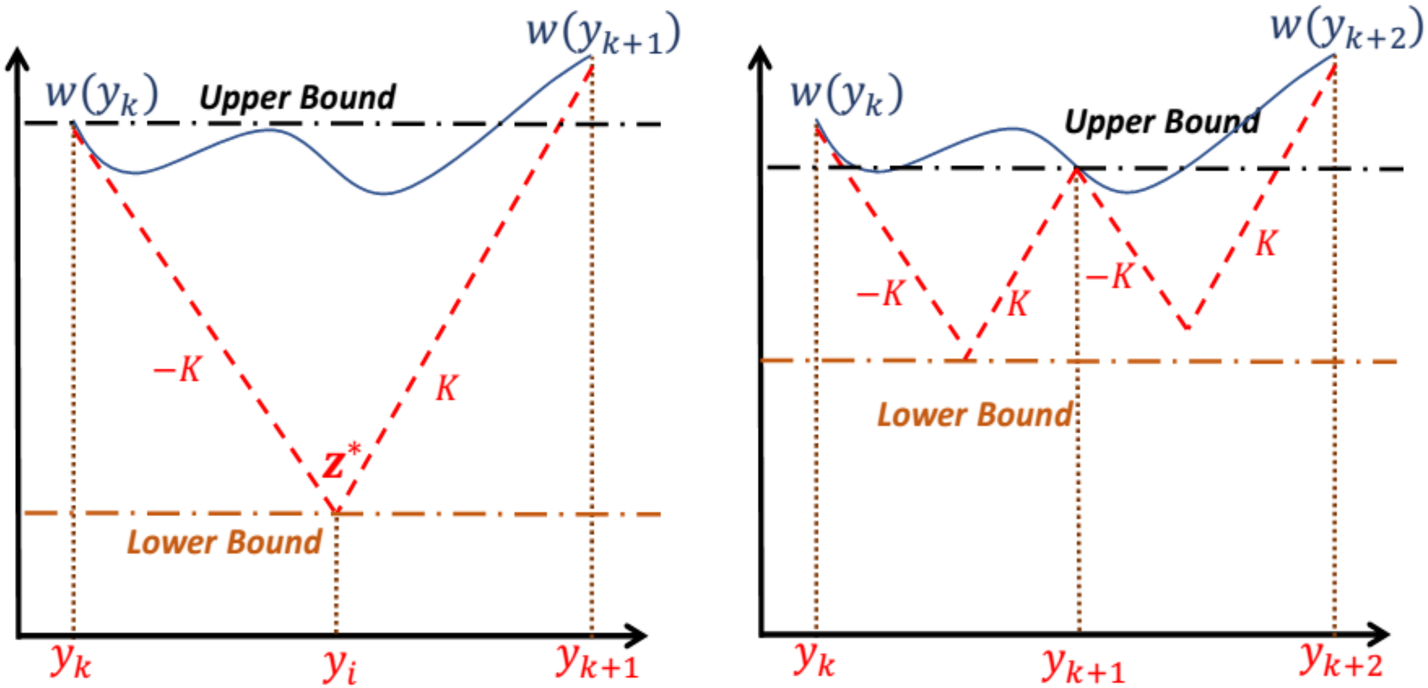


Figure: A lower-bound function designed via Lipschitz constant

Dynamic refinement of the constant

A Lipschitz constant closer to K_{best} can greatly improve the speed of convergence of algorithm. We design a practical approach to dynamically update the current Lipschitz constant according to the information obtained from the previous iteration:

$$K = \eta \max_{j=1,\ldots,i-1} \{ |(w(y_j) - w(y_{j-1}))/(y_j - y_{j-1})| \}$$

where $\eta > 1$.
Please note that, since

$$\lim_{i \to \infty} \max_{j=1,\ldots,i-1} \eta |(w(y_j) - w(y_{j-1}))/(y_j - y_{j-1})| = \eta \sup_{y \in [a,b]} dw/dy > K_{best}$$

Multi-dimension case

- The basic idea is to decompose a multi-dimensional optimization problem into a sequence of nested one-dimensional subproblems.
- Then the minimuma of those one-dimensional minimization subproblems are back-propagated into the original dimension and the final global minimum is obtained.

$$\min_{x \in [a_i, b_i]^n} w(x) = \min_{x_1 \in [a_1, b_1]} \dots \min_{x_n \in [a_n, b_n]} w(x_1, \dots, x_n) \tag{8}$$

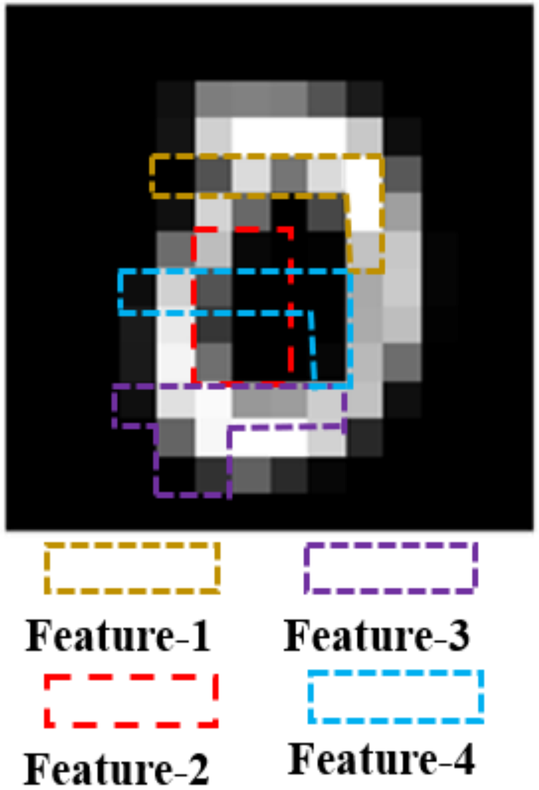
Definition (*k*-th Level Subproblem)

The k-th level optimization subproblem, written as $\phi_k(x_1, \dots, x_k)$, is defined as follows:
1. for $1 \leq k \leq n-1$,
$\phi_k(x_1, \dots, x_k) = \min_{x_{k+1} \in [a_{k+1}, b_{k+1}]} \phi_{k+1}(x_1, \dots, x_k, x_{k+1})$,
2. for $k = n$, $\phi_n(x_1, \dots, x_n) = w(x_1, x_2, \dots, x_n)$

Multi-dimension case ctd

- we have that
$$\min_{x \in [a_i, b_i]^n} w(x) = \min_{x_1 \in [a_1, b_1]} \phi_1(x_1)$$
which is actually an one-dimensional optimization problem.
- When evaluating the objective function $\phi_1(x_1)$ at $x_1 = a_1$, we need to project a_1 into the next one-dimensional subproblem
$$\min_{x_2 \in [a_2, b_2]} \phi_2(a_1, x_2)$$
.
- We recursively do the projection until reaching the n-th level one-dimensional subproblem, *i.e.*,
$$\min_{x_n \in [a_n, b_n]} \phi_n(a_1, a_2, \ldots, a_{n-1}, x_n)$$
.
- We back-propagate objective function values to the first-level $\phi_1(a_1)$ and continue searching from this level until the error bound is reached.

Case study: safety verification

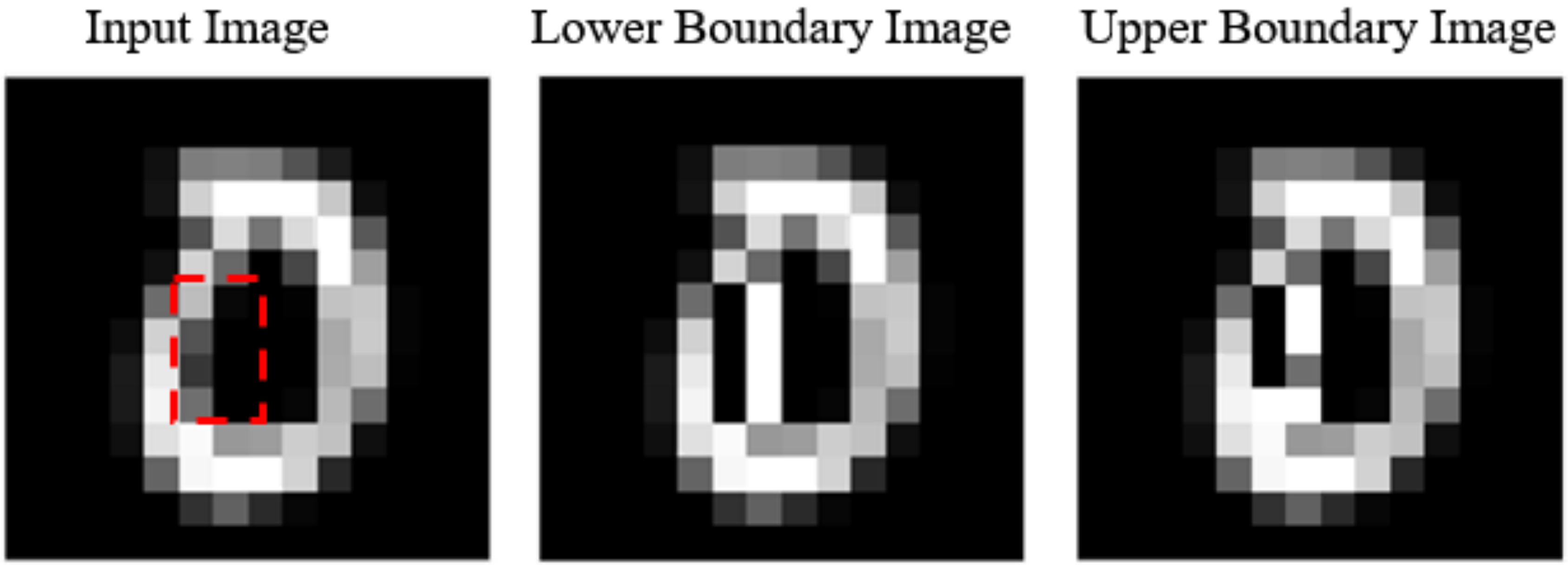


DNN-1	DNN-7		
Input Layer	Input Layer		Dropout (50%)
Convolution (2×2, 16 filters)	Convolution (2×2, 8 filters)	ReLU	Fully Connected (256 neurons)
ReLU	ReLU	Normalization	ReLU
Fully Connected (10 neurons)	Convolution (2×2, 16 filters)	Convolution (2×2, 64 filters)	Dropout (50%)
Softmax	Normalization	ReLU	Fully Connected (10 neurons)
	ReLU	Normalization	ReLU
		Convolution (2×2, 32 filters)	Softmax

- Randomly choose 20 images, 4 features manually
- Investigate DNNs of varying depth (shown shallowest and deepest)

MNIST example

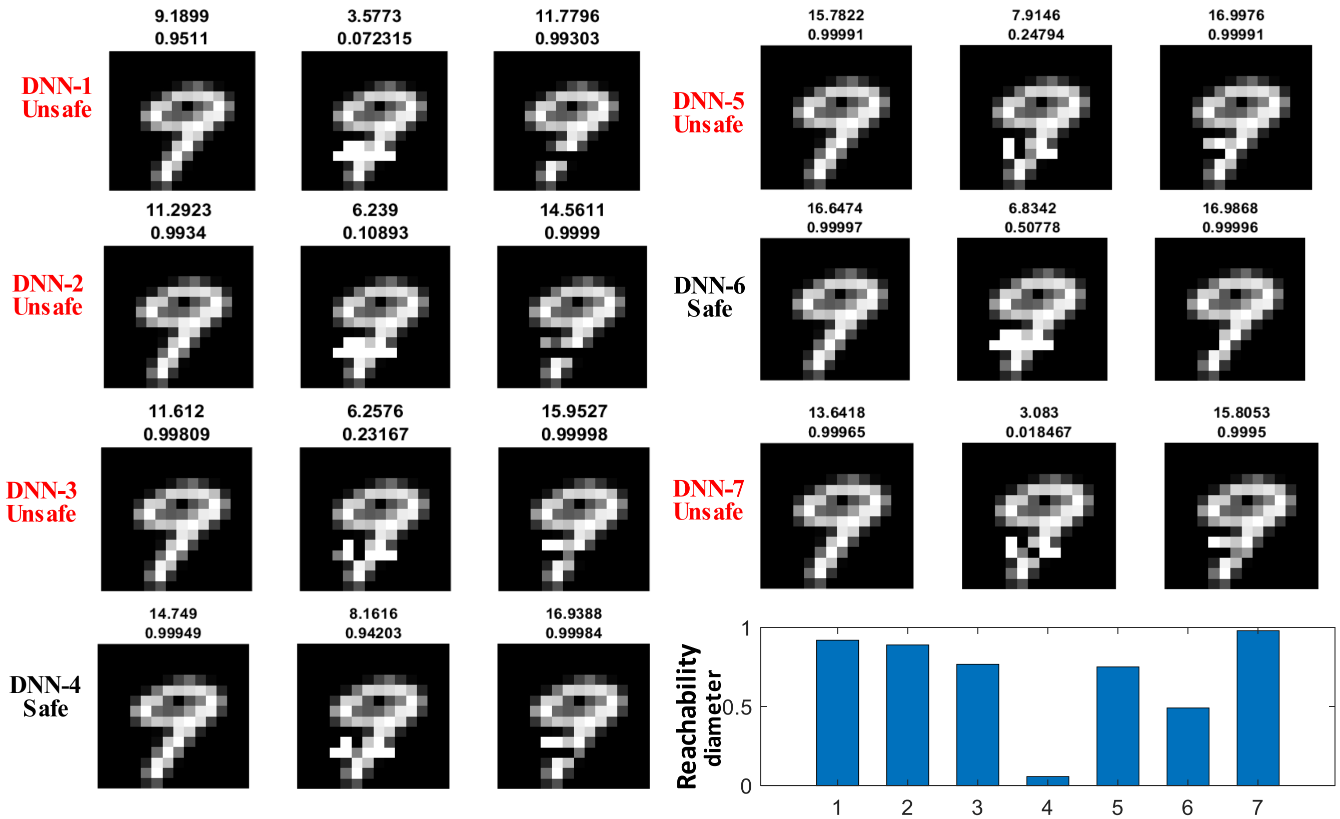
- Take an image and select a feature within it



99.95% confidence	74.36% lower bound	99.98% upper bound

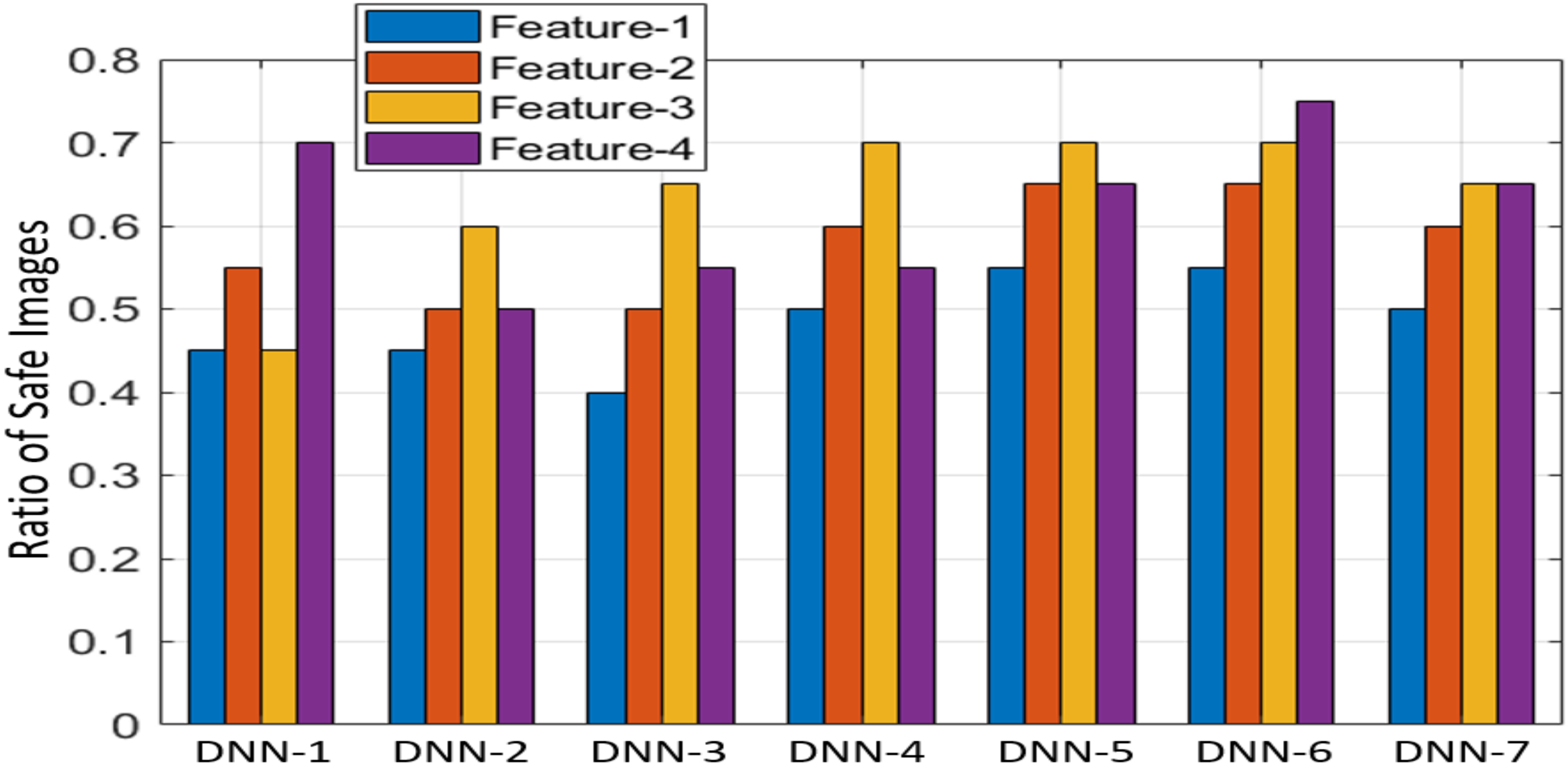
- Safety verification for the feature
  - manipulating the feature can only reduce confidence to 74.36%

Robustness comparison



- Can obtain robustness evaluation by computing expected confidence diameter weighted by the test data distribution

Safety comparison



- No DNN is 100% safe
- Choice of layers matters, not just depth: DNN6 is safest
- Feature matters: some features (e.g. 1 and 2) are more easily perturbed

Comparison with other tools

NN ID	Layer No.	Neutron No.	Time by SHERLOCK	Time by Reluplex	Our method
N-0	1	100	1.9s	1m 55s	0.4s
N-1	1	200	2.4s	13m 58s	1.0s
N-2	1	500	17.8s	Timeout	6.8s
N-3	1	500	7.6s	Timeout	5.3s
N-4	1	1000	7m 57.8s	Timeout	1.8s
N-5	6	250	9m 48.4s	Timeout	15.1s

- Sherlock and Reluplex affected by number of neurons and layers
- On the case study, improvement of 36x over Sherlock and 100x over Reluplex

Searching for adversarial examples…

- Input space for most neural networks is high dimensional and non-linear
- Where do we start?
- How can we apply structure to the problem?

- Image of a tree has 4,000 x 2,000 x 3 dimensions = 24,000,000 dimensions
- We would like to find a very ‘small’ change to these dimensions

- TACAS 2018, https://arxiv.org/abs/1710.07859

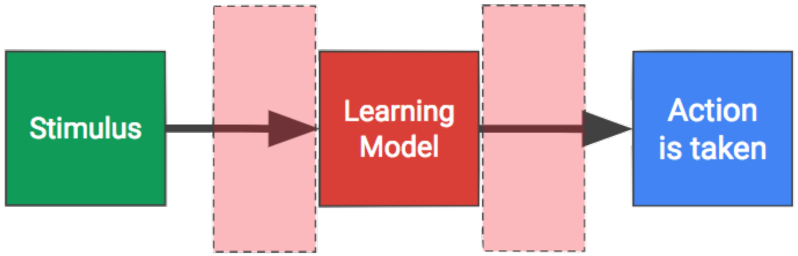
Adversarial setting

Black Box

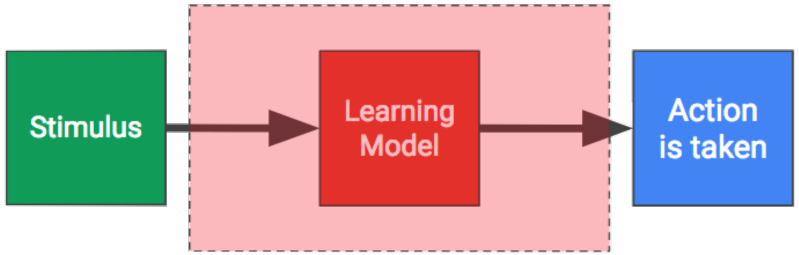
- Access only to the inputs and outputs of the network.
- NO access to any other network parameters (i.e. topology/weights)
- Able to query the network for new outputs

White Box

- All Black Box privileges
- Access to training data and test data
- Access to topology
- Access to weights
- Access to activation functions



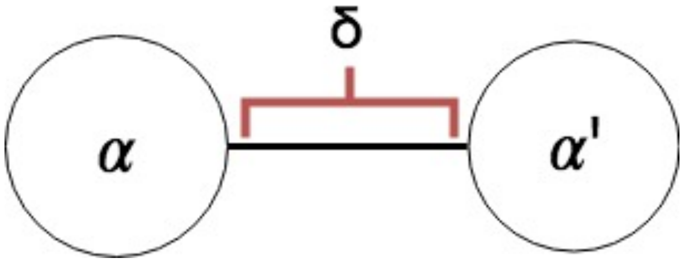
Access needed to perform black box attack



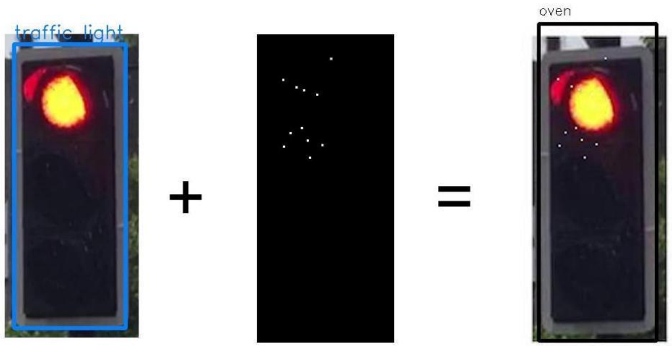
Access needed to perform white box attack

Manipulations

- We represent a single input as α
- The classification w.r.t some input is denoted $N(\alpha) = c$
- An adversarial example α' is a manipulated α, for which $N(\alpha) \neq N(\alpha')$
- Since there is no perfect measure of similarity for the image domain, we stick to using the conventional L_k metric
- We want to find an adversarial example that minimizes distance



$$\delta = \left(\sum |\alpha - \alpha'|^k\right)^{1/k} = ||\alpha - \alpha'||_k$$



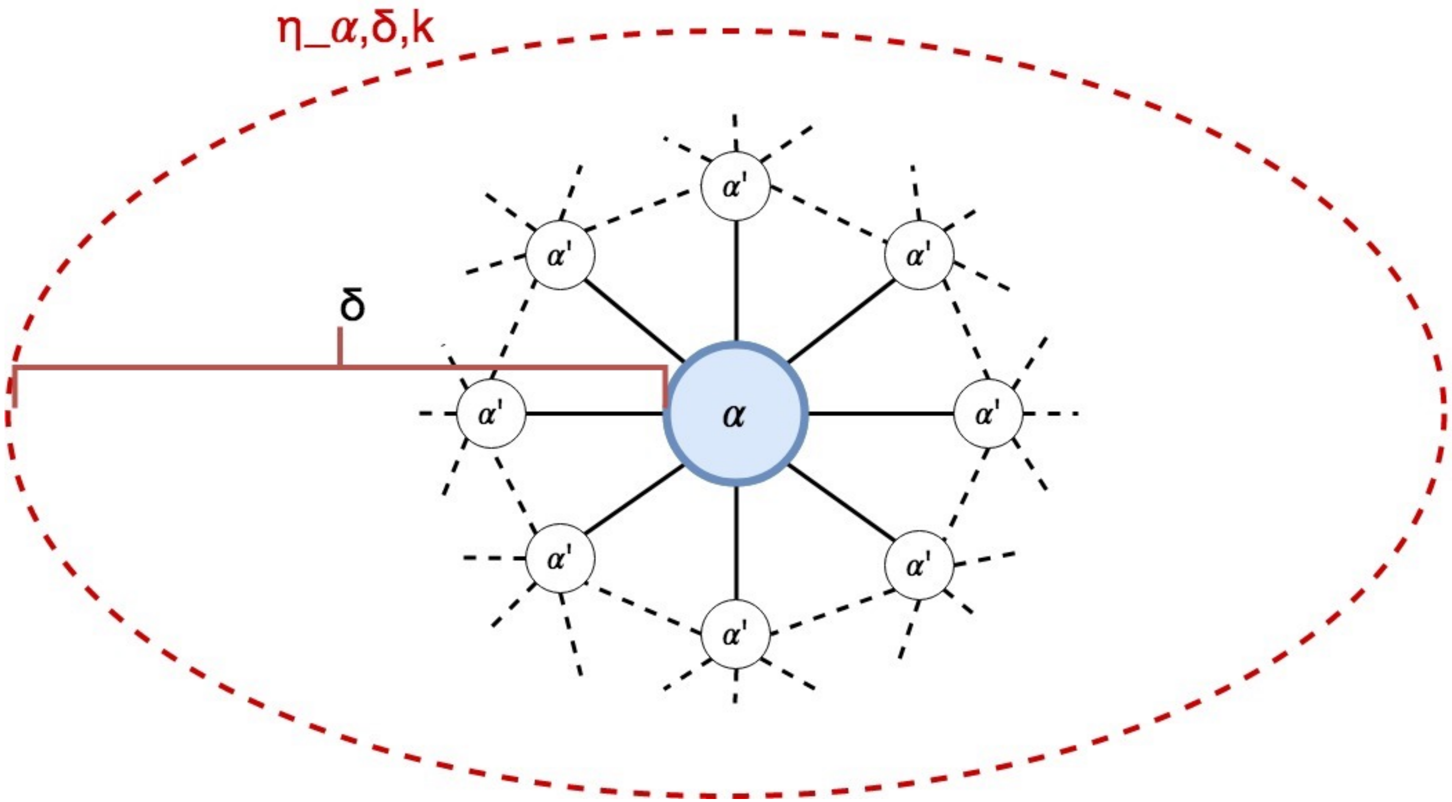
Traffic Light   11 White Pixels   Oven

$$\delta = ||\alpha - \alpha'||_0 = 11$$

Search region

- Given a specific k, an input, and a maximum distance ∂, define a search region as:

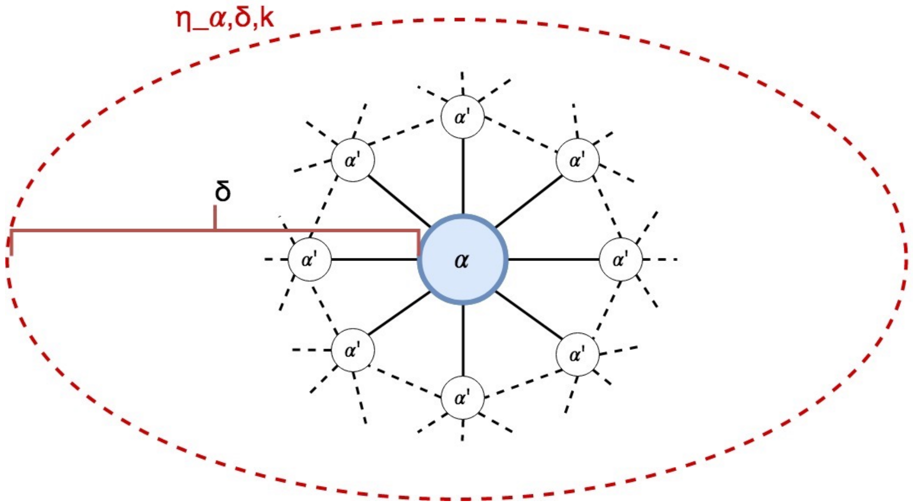
$$\eta_{\alpha,\delta,k} = \{\alpha' \in D : \delta \leq ||\alpha = \alpha'||_k\}$$



Safety within a region

- We can verify that a network is safe w.r.t an input if no adversarial example exists within a region:

$$N(\alpha) = N(\alpha')\forall\alpha' \in \eta_{\alpha,\delta,k}$$



- We now refine the notion of adversarial examples to only images within this set, denoted:

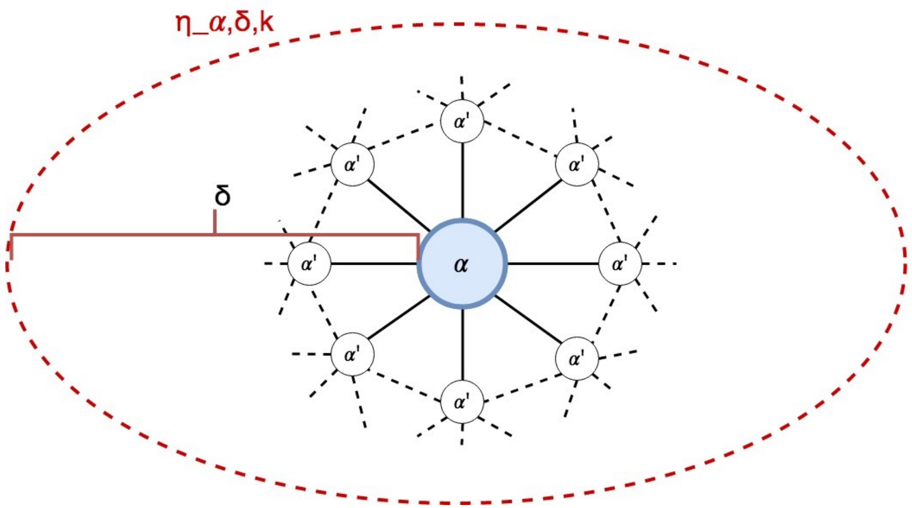
$$adv_{N,k,\delta}(\alpha)$$

Safety within a region

- We can verify that a network is safe w.r.t an input if no adversarial example exists within a region:

$$N(\alpha) = N(\alpha')\forall\alpha' \in \eta_{\alpha,\delta,k}$$

We have not established a good handle on 'where' to move in this space!



- We now refine the notion of adversarial examples to only images within this set, denoted:

$$adv_{N,k,\delta}(\alpha)$$

Feature-based exploration

- Searching by trying every combination of pixel values is intractable
- We can 'reduce' the dimensionality of an images by reducing it only to its salient features

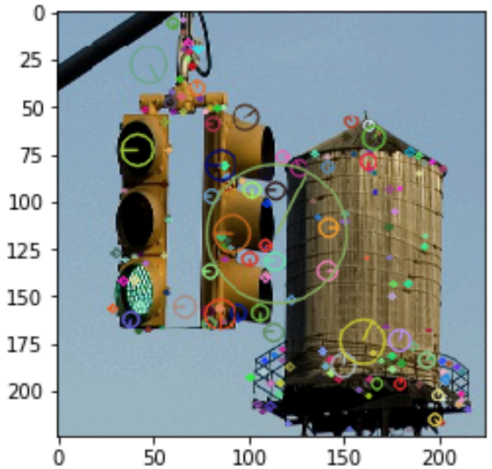
$\Lambda(\alpha)$ – Set of features given an image

λ_r – Response strength of the feature (roughly how 'important' it is)

λ_x – X coordinate of a keypoint

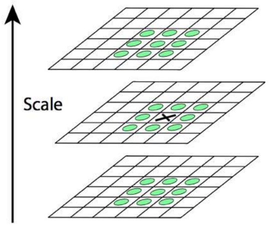
λ_y – Y coordinate of a keypoint

λ_s – Radius of a keypoint



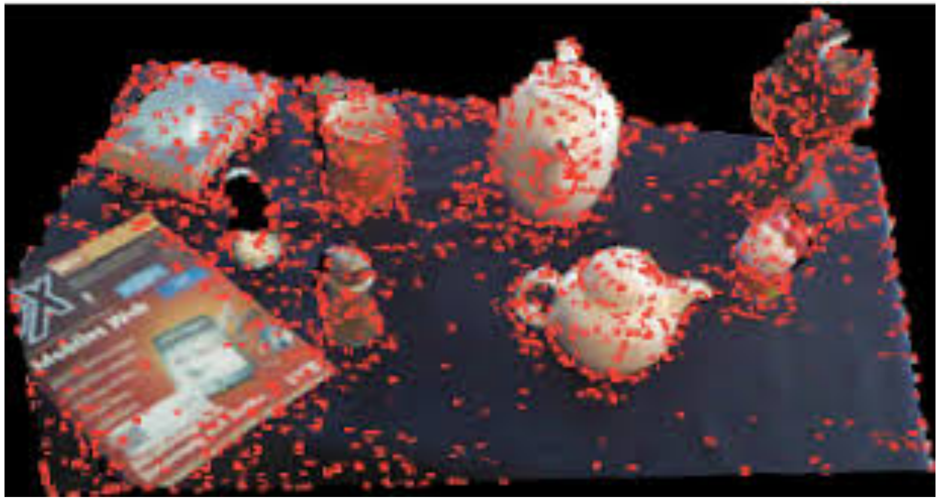
Feature extraction algorithms (SIFT)

- (1) Scale space extrema detection



We blur the image in order to detect extrema of different sizes

- (2) Keypoint localization and description

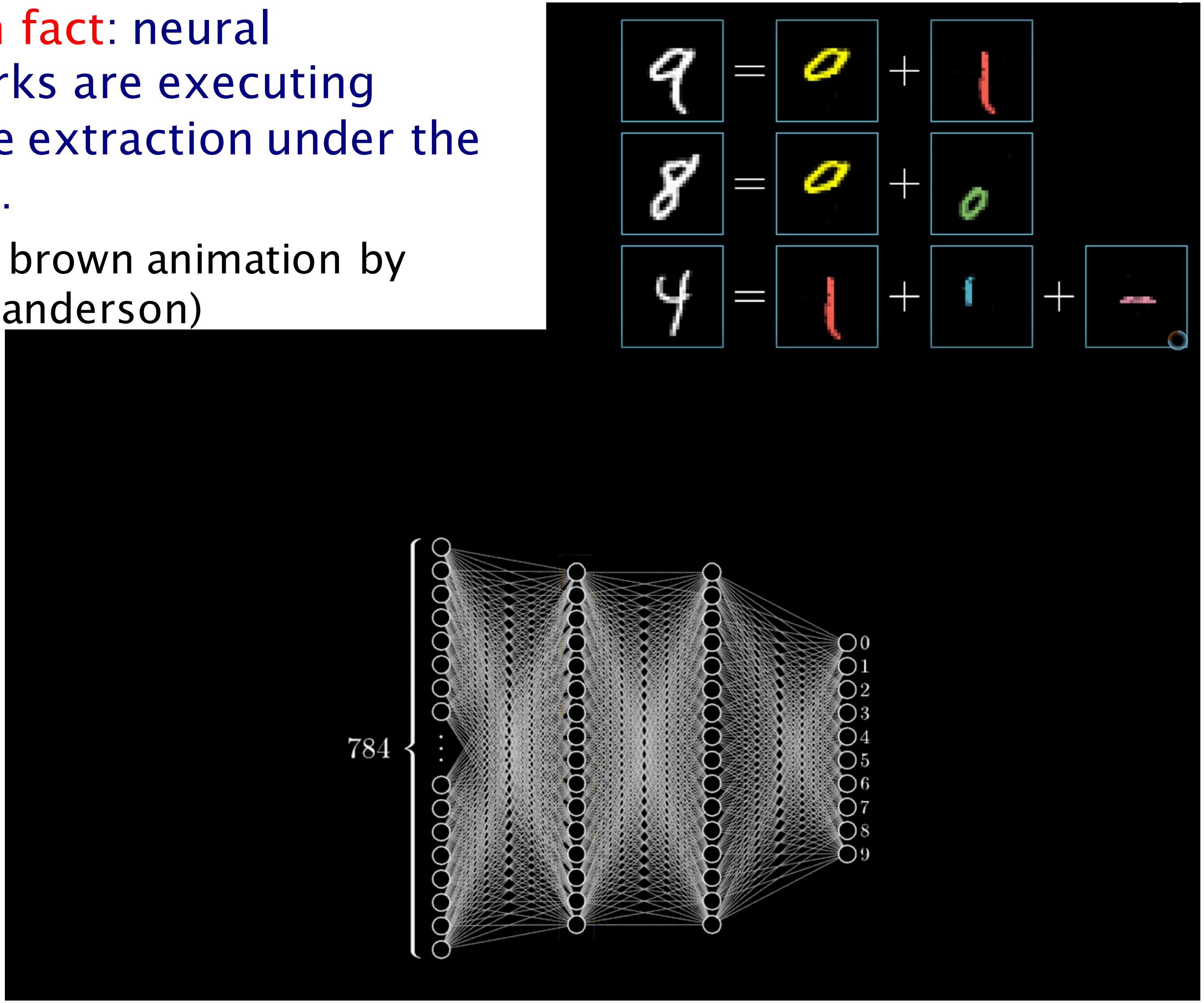


Localization looks at the gradients from the scale space to describe each keypoint

SIFT is invariant to scale, rotation and translation

Intuition for feature-based exploration

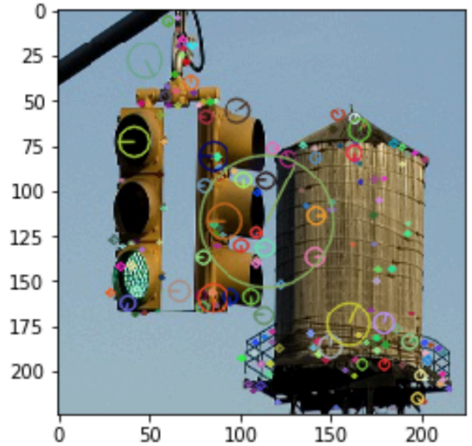
- Known fact: neural networks are executing feature extraction under the hood…
- (3blue1brown animation by Grant Sanderson)

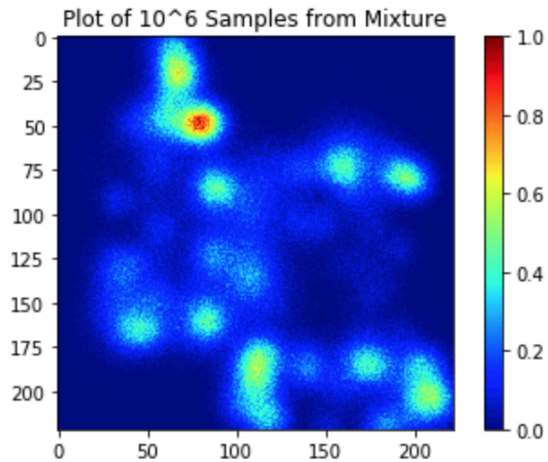


Feature-based representation

- The SIFT algorithm, while reliably able to extract keypoints, is not able to guarantee coverage of every pixel in the image
- We use a Gaussian mixture model in order to assign each pixel a probability based on its perceived saliency

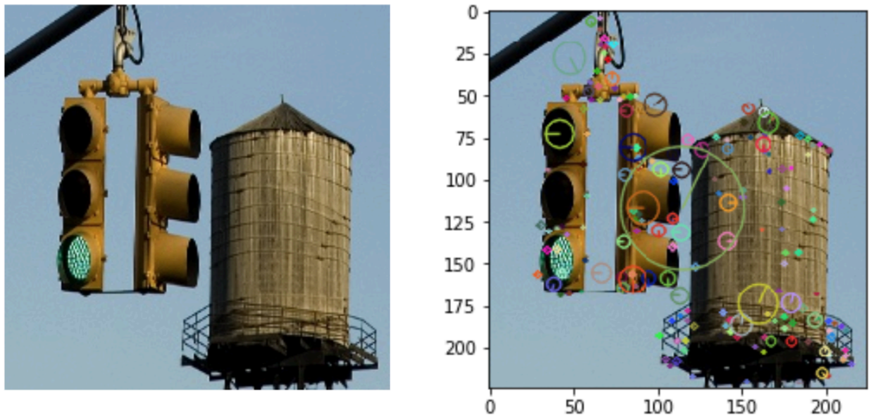
$$\mathcal{G}_{i,x} = \frac{1}{\sqrt{2\pi\lambda_{i,s}^2}} exp(\frac{-(p_x - \lambda_{i,x})^2}{2\lambda_{i,s}^2}) \qquad \mathcal{G}_{i,y} = \frac{1}{\sqrt{2\pi\lambda_{i,s}^2}} exp(\frac{-(p_y - \lambda_{i,y})^2}{2\lambda_{i,s}^2})$$





Solution: two-player game

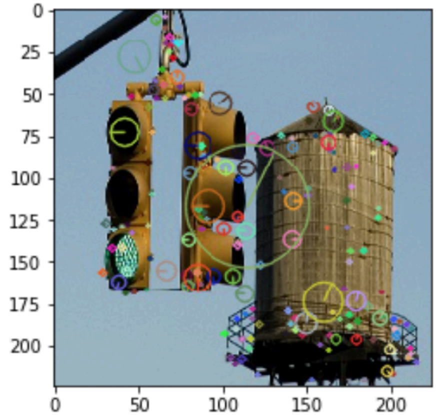
- Goal is finding adv. example, reward inverse of distance
- Player 1 selects the feature that we will manipulate from $\Lambda(\alpha)$



- Each keypoint represents a possible move for player 1
- Player 2 then selects the pixels that will be manipulated
- Use Monte Carlo tree search to explore the game tree, while querying the network to align features
- Method black box, and can converge to the optimal strategy (optimal adversarial example)

Players moves and strategy

- Player 1 selects the feature that we will manipulate $\Lambda(\alpha)$



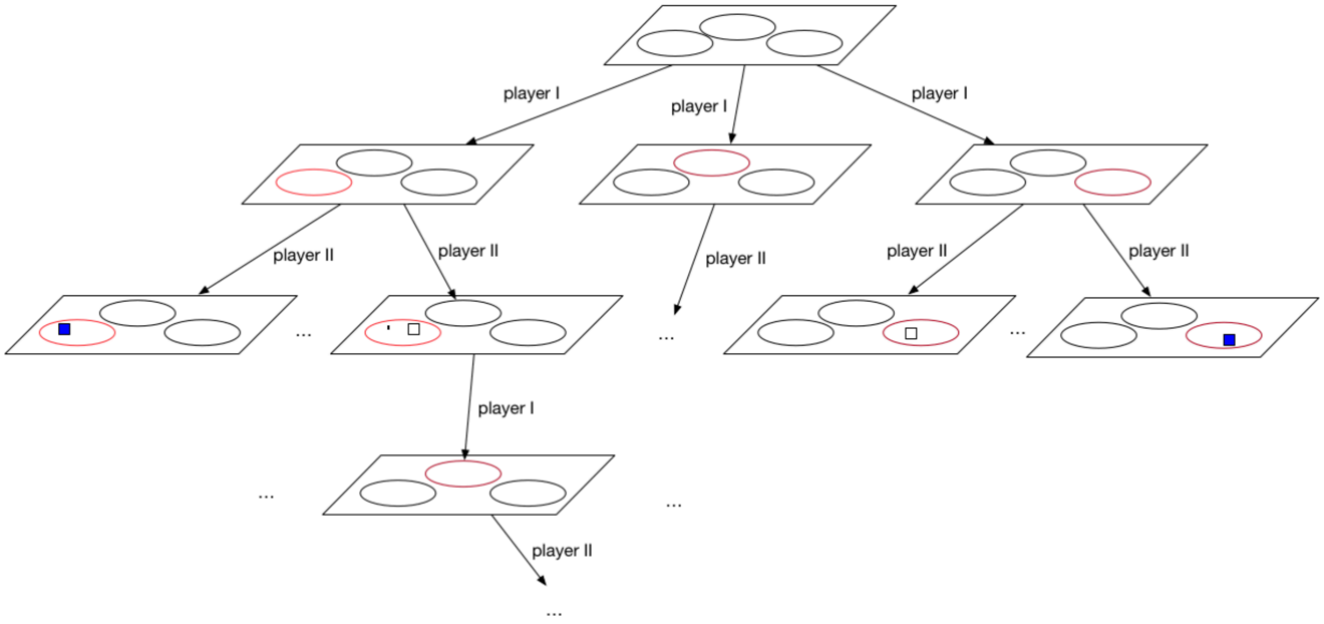
- Initial strategy: weight by importance (response strength)

- Player 2 manipulates pixels by some bounded value
- Initial strategy: select from the GMM

$$\mathcal{G}_{i,x} = \frac{1}{\sqrt{2\pi\lambda_{i,s}^2}} exp\left(\frac{-(p_x - \lambda_{i,x})^2}{2\lambda_{i,s}^2}\right) \quad \mathcal{G}_{i,y} = \frac{1}{\sqrt{2\pi\lambda_{i,s}^2}} exp\left(\frac{-(p_y - \lambda_{i,y})^2}{2\lambda_{i,s}^2}\right)$$

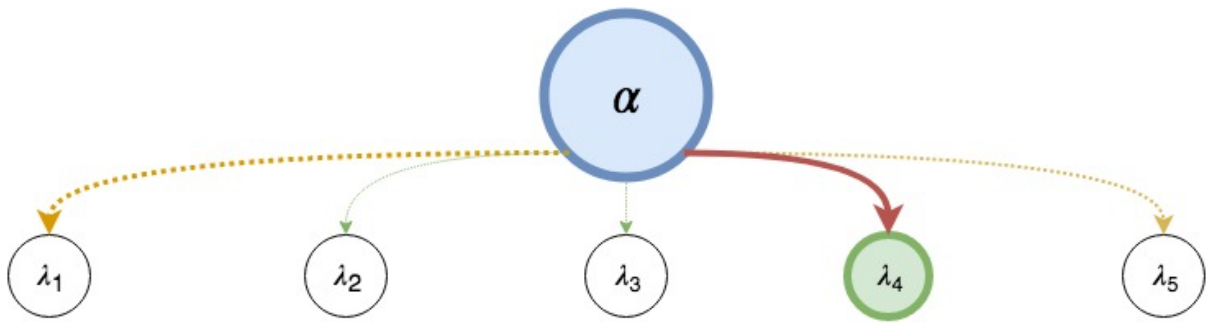
Monte Carlo Tree Search

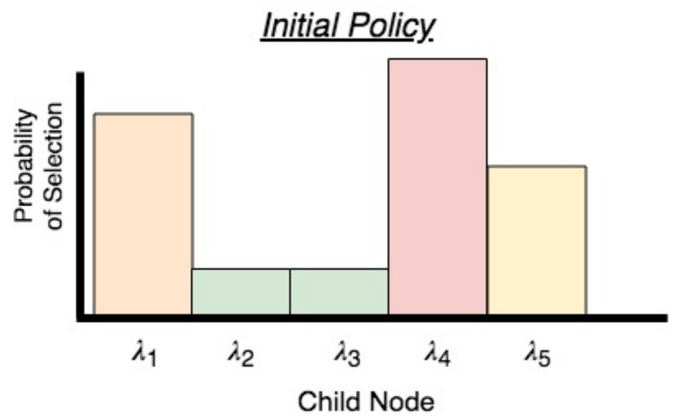
- To efficiently explore the feature space (play the game) of an image we employ the Monte Carlo Tree Search algorithm
- Each game play can be represented as a path down the tree



MCTS: selection/expansion

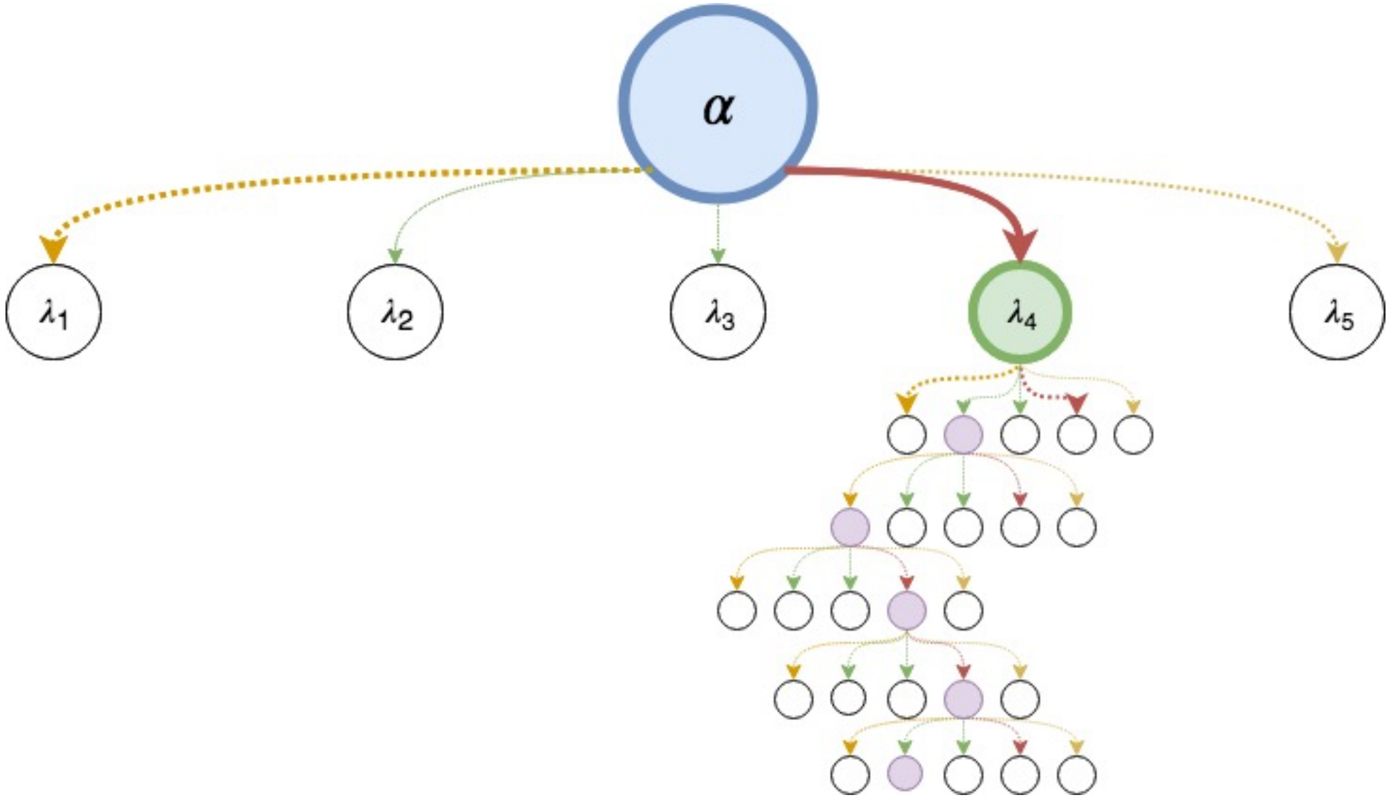
- The root of our tree represents the original image, and each child represents a potential manipulated image
- First step is to select a manipulation based on each players strategy
- If the child has never been selected from previously then we “expand” the tree to select a new leaf.





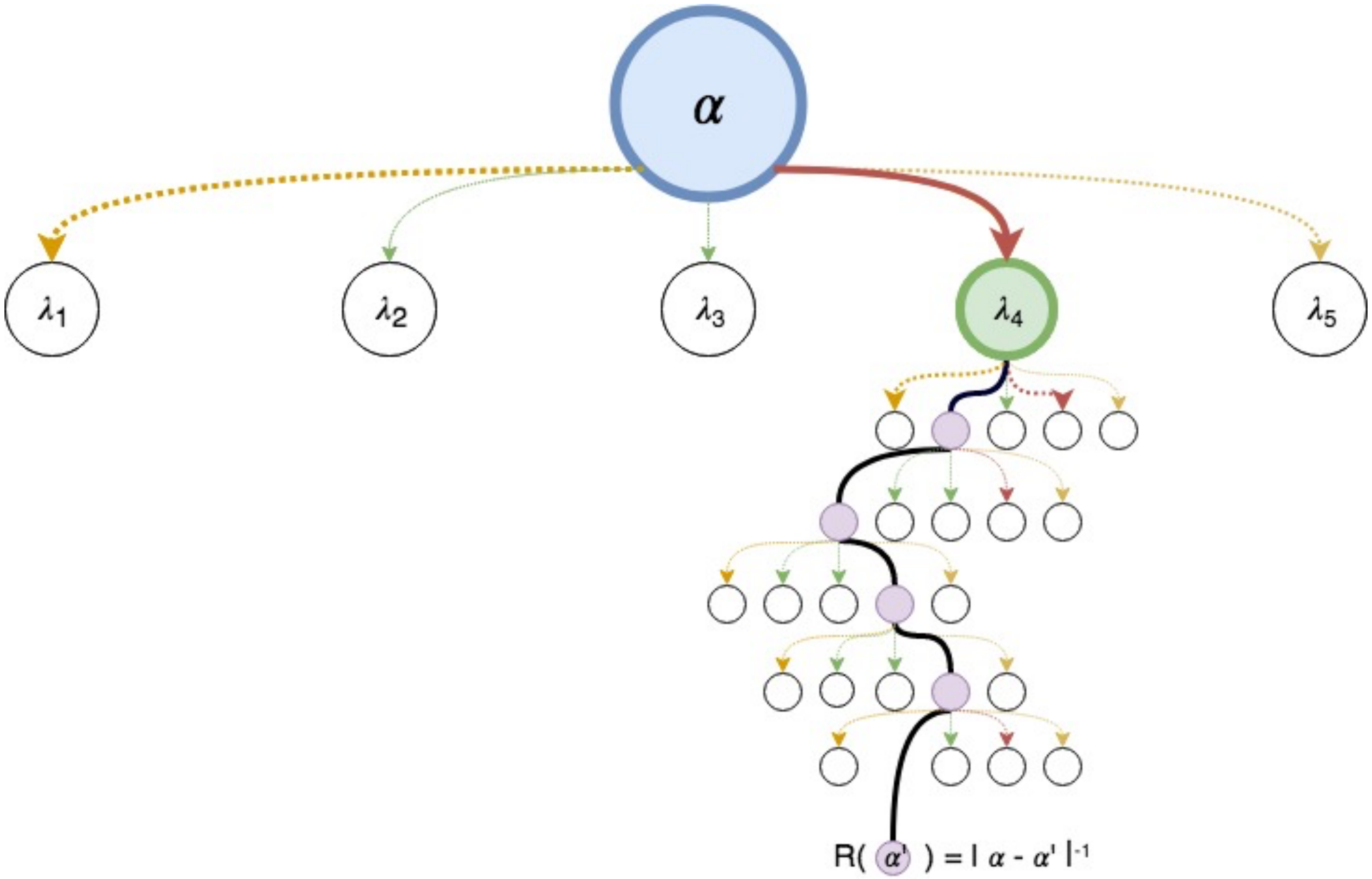
MCTS: simulation

- After a new child has been added to the tree, we approximate the reward of visiting this child by continuously searching the tree until we have either timed out or hit an adversarial example
- These nodes are not recorded as a part of the partial tree

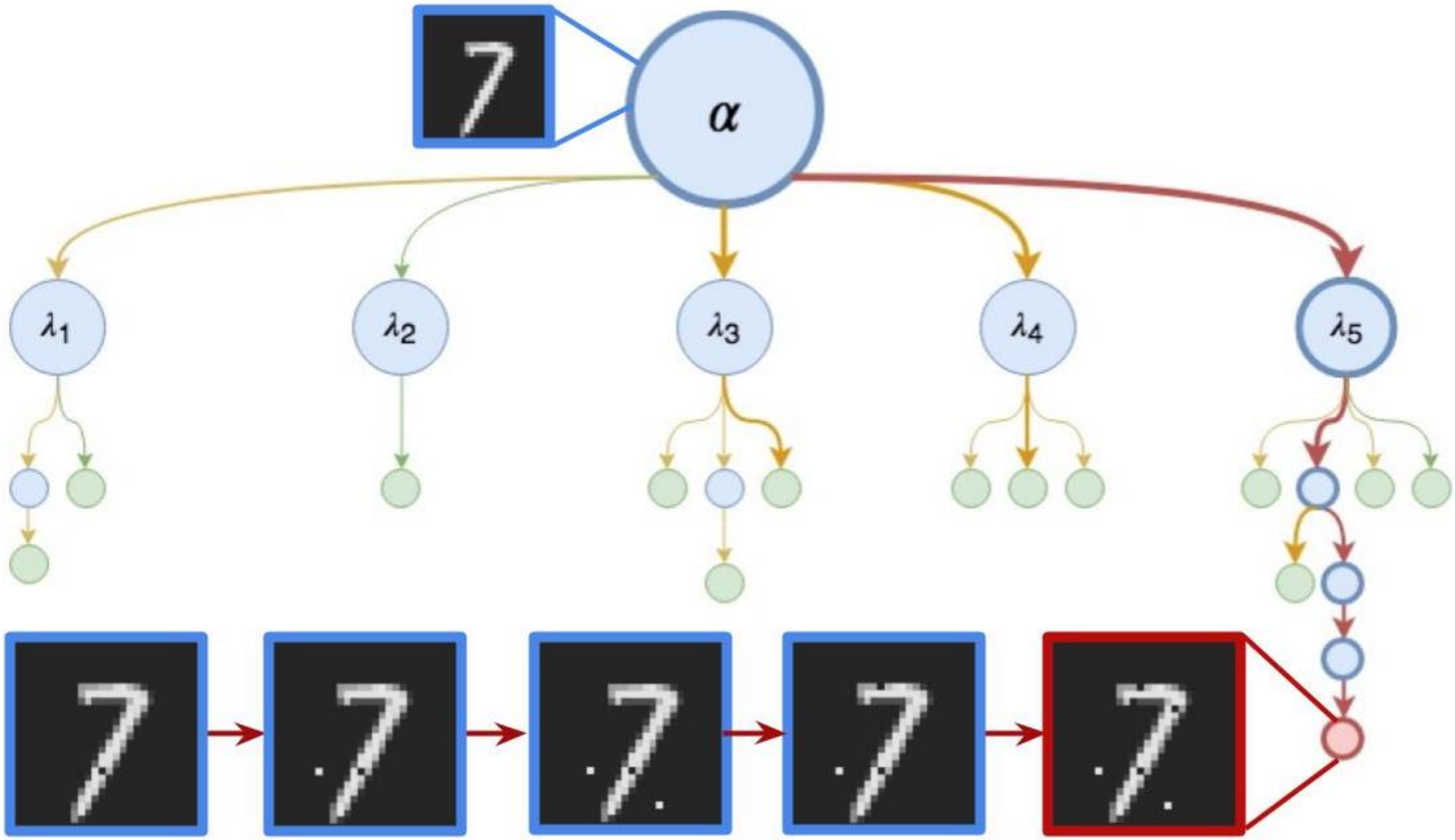


MCTS: backpropagation

- After we have terminated the tree, we calculate the reward, and backpropagate that reward up the tree to update our exploration policy (update each player's strategies)

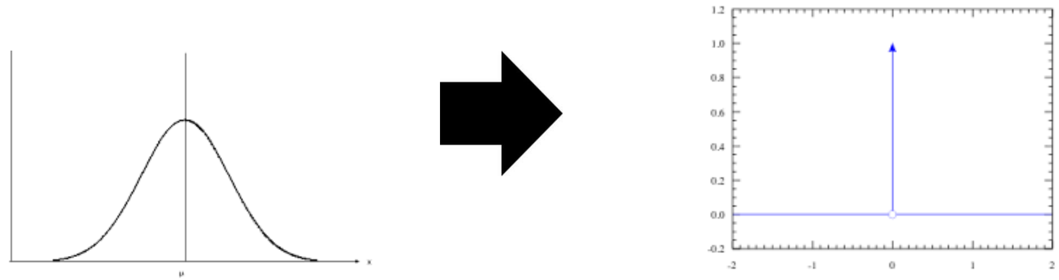


Tree expands until example is found

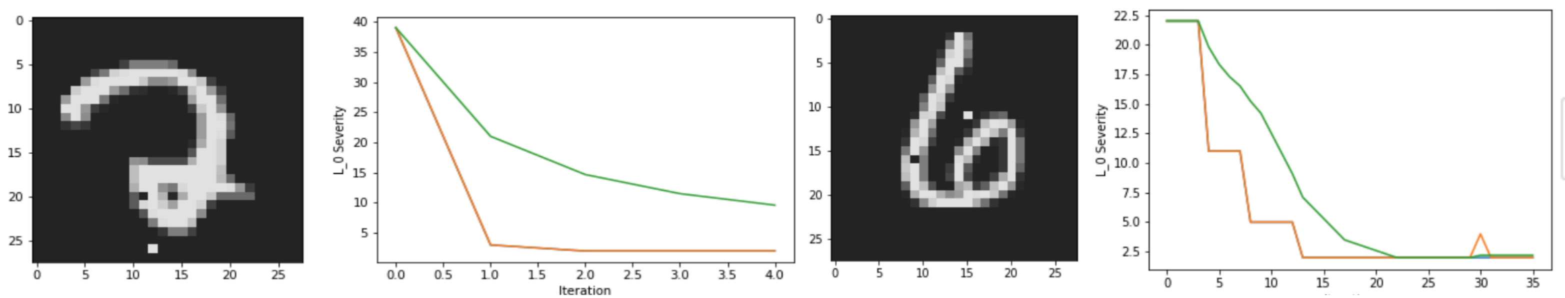


MCTS/Game convergence

- The game converges when each player's strategy at any point is a Dirac distribution

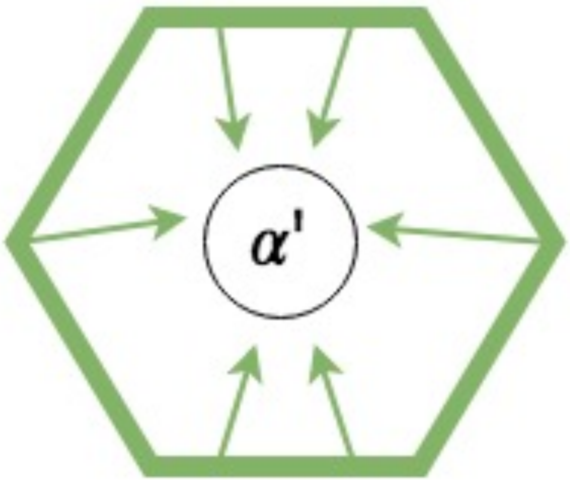


- If both players choose the next node based on a Dirac distribution, then the game converges to a deterministic and memoryless strategy
- In practice, this convergence is quick! (a matter of seconds)



Lipschitz networks

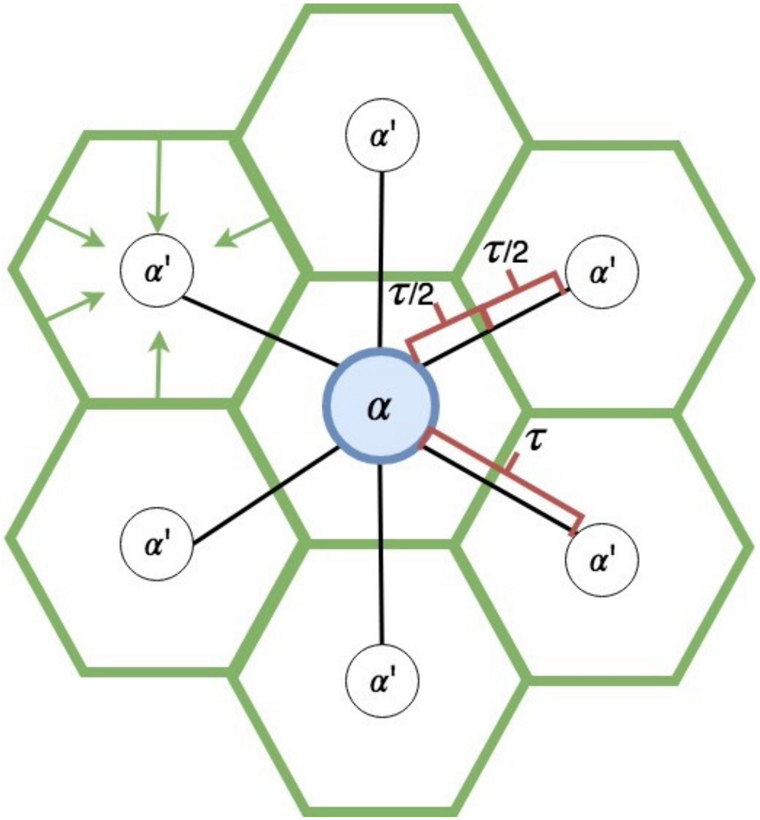
- Recall Lipschitz continuity limits the rate of change of output
- For Lipschitz networks, there exists a diameter such that every image within it shares the classification of a given input



- Use this fact to provide safety guarantees

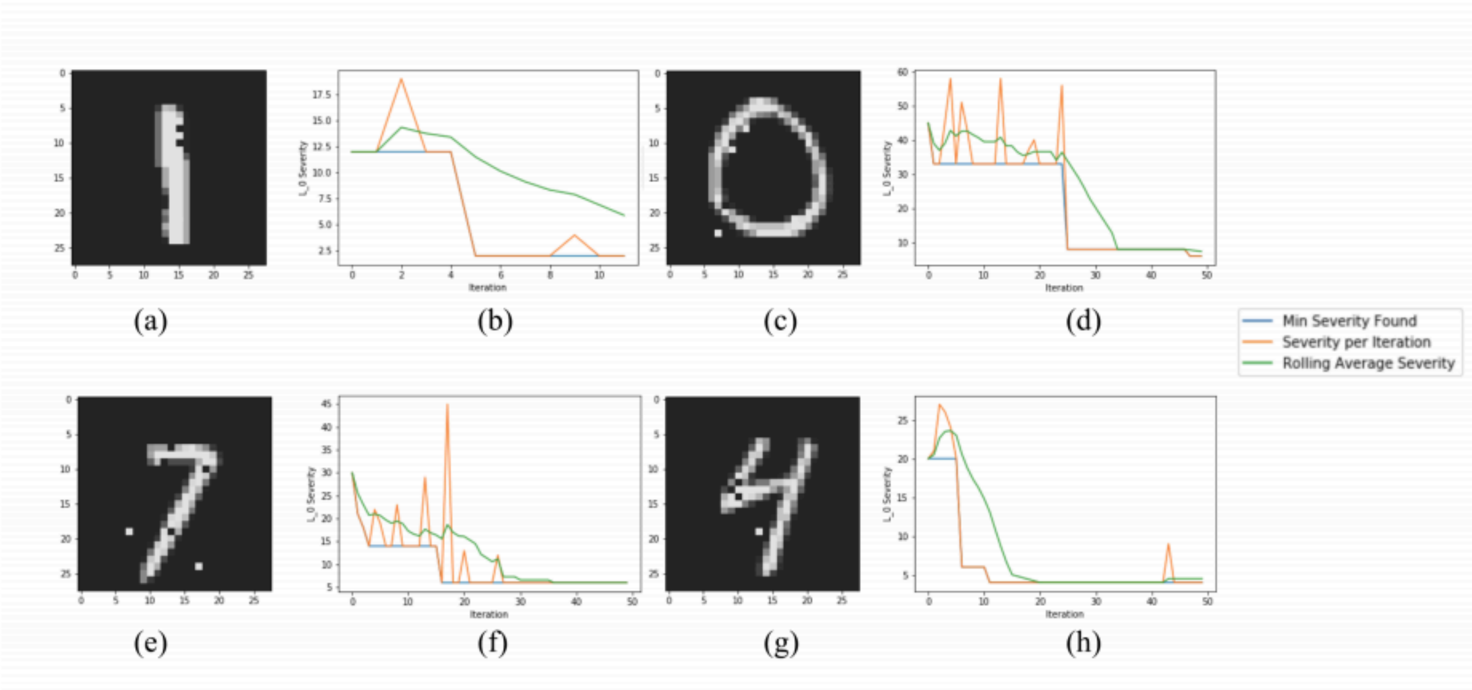
Safety guarantee via MCTS

- Cover the region with a 'grid' of diameter $\frac{\tau}{2}$ (half of manipulation size)
- If the MCTS fails to find an adversarial example then we can deduce that one does not exist



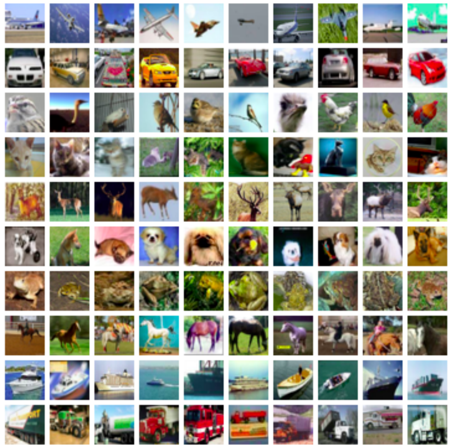
Results of safety testing (MNIST)

- Our black box algorithm can often converge to an optimal strategy,
- and does so in a very short amount of time (less than a second for these small images)



Comparison with known algorithms

- On several standard benchmarks, achieves competitive performance with white box optimization and heuristic search,
- Also allows for guarantees not provided by competing algorithms



L_0	CW (L_0 algorithm)	Game (timeout = 1m)	JSMA-F	JSMA-Z
MNIST	8.5	14.1	17	20
CIFAR10	5.8	9	25	20

Table 1: CW vs. Game (this paper) vs. JSMA

Scaling up to large networks (ImageNet)

- Scaling up to some of the larger images from ImageNet (300 x 300 x 3), we see that our method continues to scale
- For an image that is roughly 350 times larger than MNIST images, we are still able to find adversarial examples, often in less than one minute

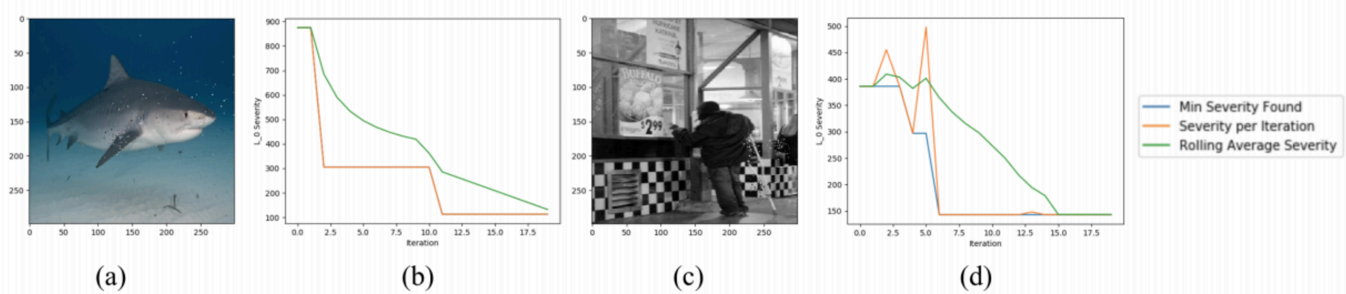
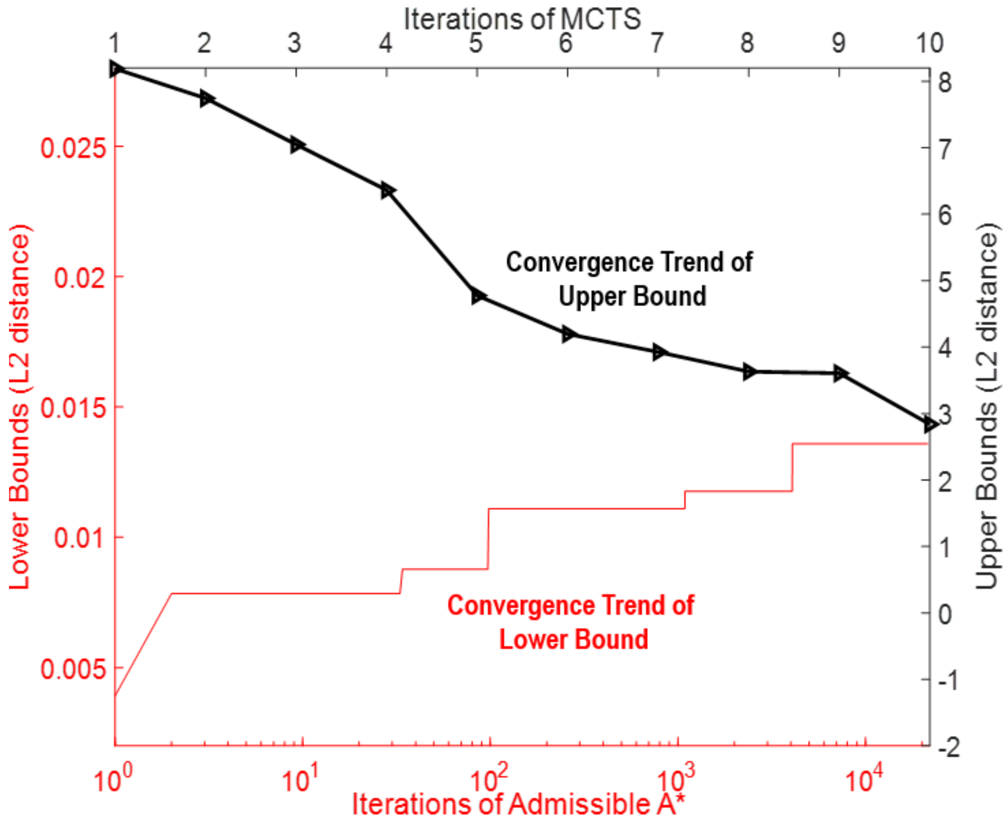
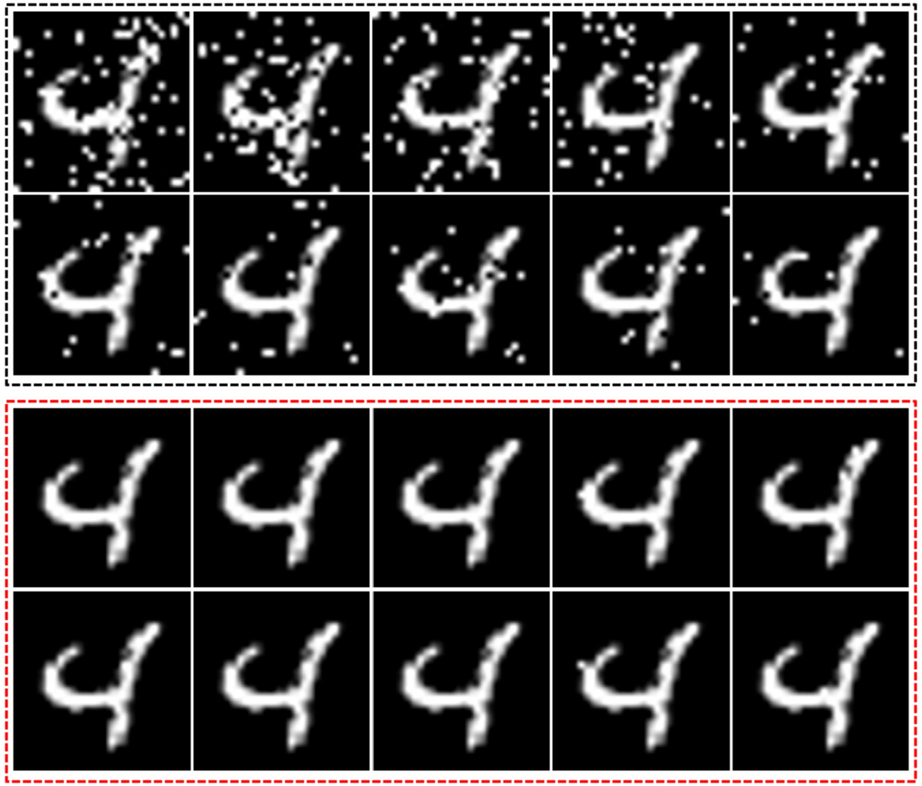


Fig. 10: Adversarial examples generated on the VGG16 architecture trained on ImageNet data. (a) Image of a great white shark classified as a galeocerdo cuvieri with confidence 42% after 113 manipulations and (b) the demonstration of convergence over 20 simulations. (b) An image of a crutch classified as bakery after 143 manipulations and (d) the demonstration of convergence over 20 simulations.

Recent improvement: lower bounds

- Convergence of lower and upper bounds on maximum safe radius

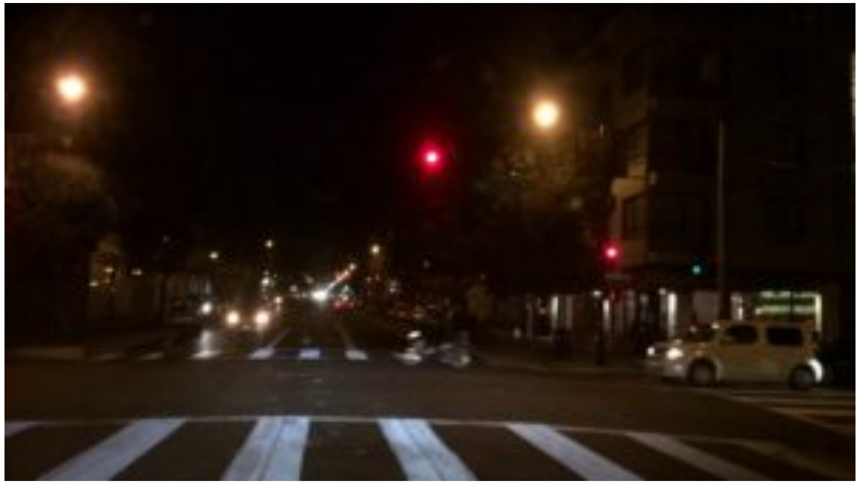




- See arXiv:1807.0357

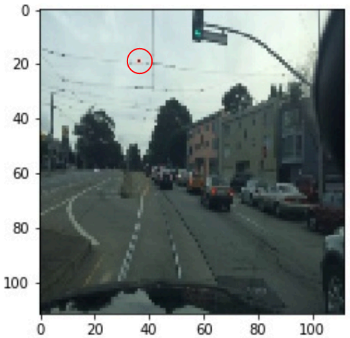
Evaluating safety-critical scenarios: Nexar

- Dashboard camera images from the Nexar dataset were taken in order to test a safety critical situation
- Tens of thousands of images were taken from real dash cams in all weather and lighting conditions
- Challenge winning network achieves 95% accuracy over unseen test data

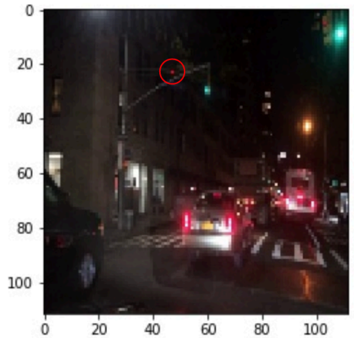


Evaluating safety-critical scenarios: Nexar

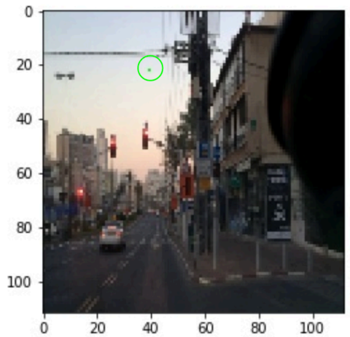
- Using our Game-based Monte Carlo Tree Search method we were able to reduce the accuracy of the network to 0%
- On average, each input took less than a second to manipulate (.304 seconds)
- On average each image was vulnerable to 3 pixel changes



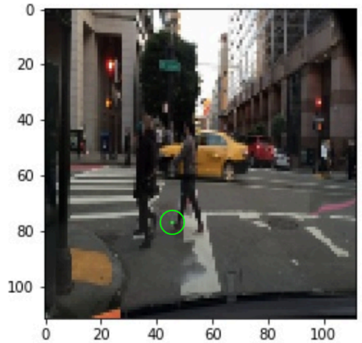
(a)



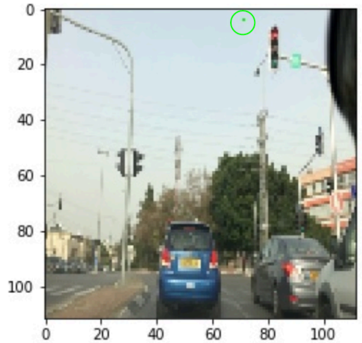
(b)



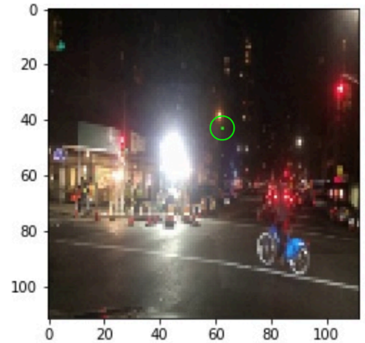
(c)



(a)



(b)



(c)

Challenges for verification of NNs

- Fascinating application domain, huge challenges!
- Many aspects of neural networks make them very difficult for us to apply typical verification techniques
  - no source code (only weights)
  - variety of topologies and activation functions
  - high dimensionality of input space
  - size of sample space
  - lack of interpretability
- The goals of this work are to provide
  - scalable and efficient
  - with provable guarantees

Conclusion

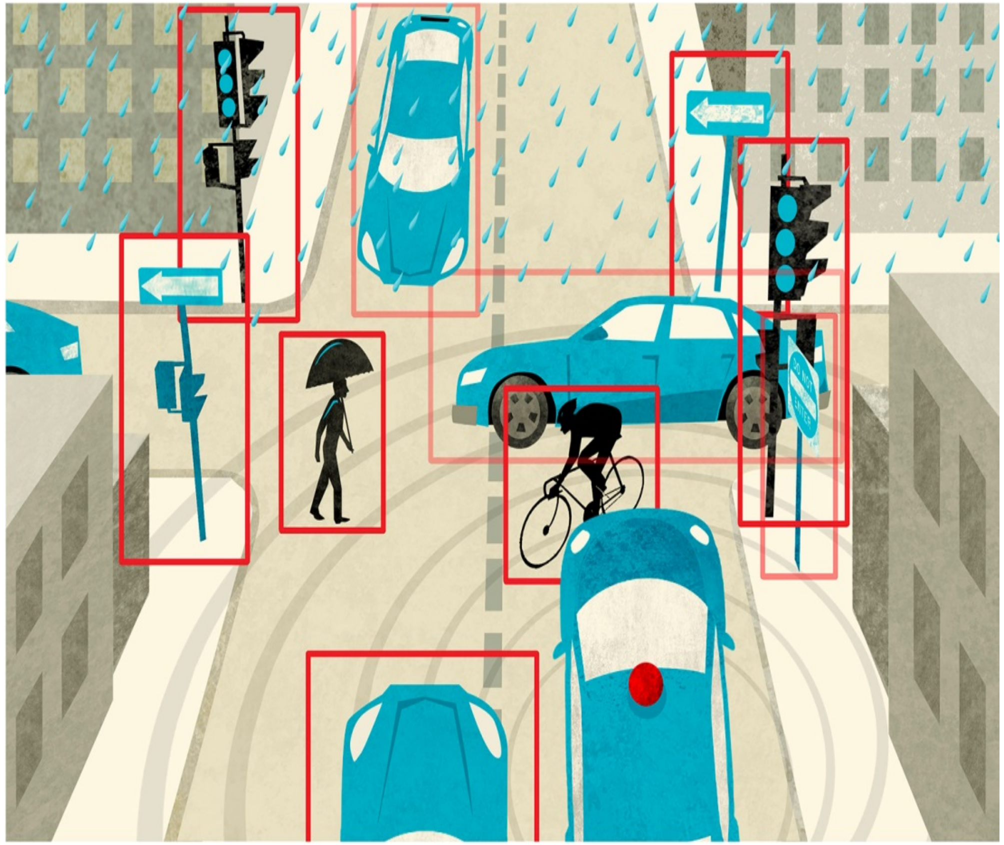
- Deep learning should be more critically evaluated when put into practice in safety- and security-critical situations
- Adversarial examples help in understanding the robustness of DNN decision boundaries
- Proposed first framework for safety verification of deep neural network classifiers
  - search-based (SMT) and Monte Carlo tree search
  - feature-guided exploration for fast, black-box testing, in a stochastic game framework
  - provable guarantees for Lipschitz continuous networks
- Future work
  - how best to use adversarial examples: training vs logic
  - more complex properties?
- Recent work: check out arxiv

AI safety – challenge for verification?

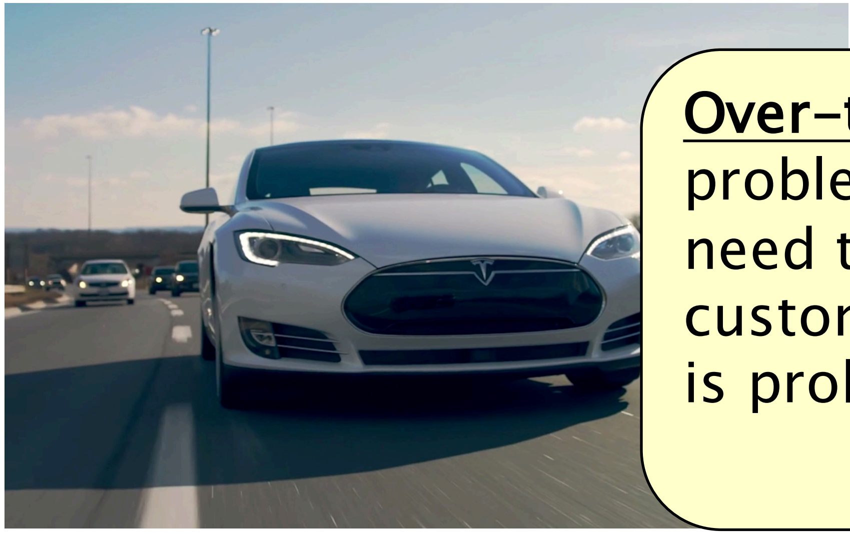
- Complex scenarios
  - goals
  - perception
  - situation awareness
  - context (social, regulatory)

- Safety-critical, so guarantees needed

- Should failure occur, accountability needs to be established



Reasoning about cognitive trust



Over-trust and inattention are known problems that technology developers need to **design for**, and simply telling customers not to do what comes naturally is probably not enough.

Patrick Lin

- Formulate a theory for expressing and reasoning about social trust in human-robot collaboration/competition
- Develop tools for trust evaluation to aid design and analysis of human-robot systems

Quantitative verification for trust?

- Logic PRTL* undecidable in general

- Have identified decidable fragments (EXPTIME, PSPACE, PTIME), by restricting the expressiveness of the logic and the stochastic multiagent systems

- Reasoning about trust can be used
  - in decision-making for robots
  - to justify and explain trust-based decisions, also for humans
  - to infer accountability for failures

- Next step is to develop model checking for trust…

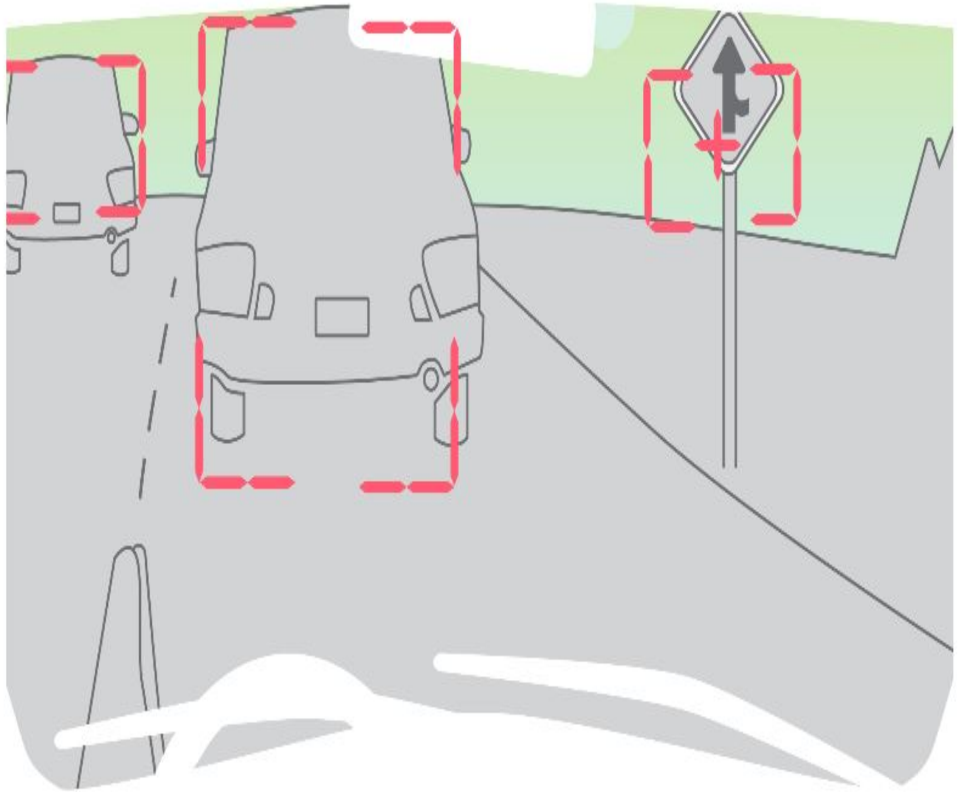
- But many challenges remain!

Morality, ethics and social norms

- Already merging into traffic proving difficult, what about social subtleties?

- What to do in emergency?
  - moral decisions
  - enforcement
  - conflict resolution
  - handover in semi-autonomous driving

- Obey traffic rules
  - cultural dependency



A car is merging on the freeway. How fast should it drive?

http://www.pbs.org/wgbh/nova/next/tech/robot-morals/

Acknowledgements

- My group and collaborators in this work
- Project funding
  - ERC Advanced Grant
  - EPSRC Mobile Autonomy Programme Grant
  - Oxford Martin School, Institute for the Future of Computing

- See also
  - VERIWARE www.veriware.org
  - PRISM www.prismmodelchecker.org


FLoC
2018
DEPARTMENT OF
COMPUTER
SCIENCE
UNIVERSITY OF
OXFORD
06–19 JULY 2018
OXFORD, UK
FEDERATED LOGIC
CONFERENCE
2018

Summit on…

- Machine Learning Meets Formal Methods!

- Date: 13 July 2018
- Venue: University of Oxford
- Talks and panel discussion by academics and industrialists

- https://www.turing.ac.uk/events/summit-machine-learning-meet-formal-methods/

- http://www.floc2018.org/

Summit on ML Meets FM

Machine learning has revolutionised computer science and AI: deep neural networks have been shown to match human ability in various tasks and solutions based on machine learning are being deployed in real-world systems, from automation and self-driving cars to security and banking. Undoubtedly, the potential benefits of AI systems are immense and wide ranging. At the same time, recent accidents of machine learning systems have caught the public's attention, and as a result several researchers are beginning to question their safety. Traditionally, safety assurance methodologies are the realm of formal methods, understood broadly as the rigorous, mathematical underpinning of software and hardware systems. They are rooted in logic and reasoning, and aim to provide guarantees that the system is behaving correctly, which is necessary in safety-critical contexts. Such guarantees can be provided automatically for conventional software/hardware systems using verification technologies such as model checking or theorem proving. However, machine learning does not offer guarantees, and reasoning techniques necessary to justify safety of its autonomous decisions are in their infancy.

FLoC 2018

The summit on machine learning meets formal methods will bring together academic and industrial leaders who will discuss the benefits and risks of machine learning solutions. The overall aim is to identify promising future directions for research and innovation of interest to The Alan Turing Institute and UK research councils and government agencies, which will be summarised in a written report that will be made public.

FLoC – Inspiring lecturers

Keynote
Shafi Goldwasser (MIT and Weizmann)
Georges Gonthier (INRIA Saclay)

Plenary
Byron Cook (Amazon and UCL)
Peter O’Hearn (Facebook and UCL)

Public lecture 10 July
Stuart Russell (UC Berkeley)
Logic and Probability

FLoC -Debate

Oxford Union-style debate on Ethics for Robotics

Luciano Floridi (Oxford/ATI)
Francesca Rossi (Padova)
Ben Kuipers (Michigan)
Jeannette Wing (Columbia)
Matthias Scheutz (Tufts)
Sandra Wachter (Oxford/ATI)

Moderated by Judy Wajcman (LSE)