UCDAVIS
COMPUTER SCIENCE

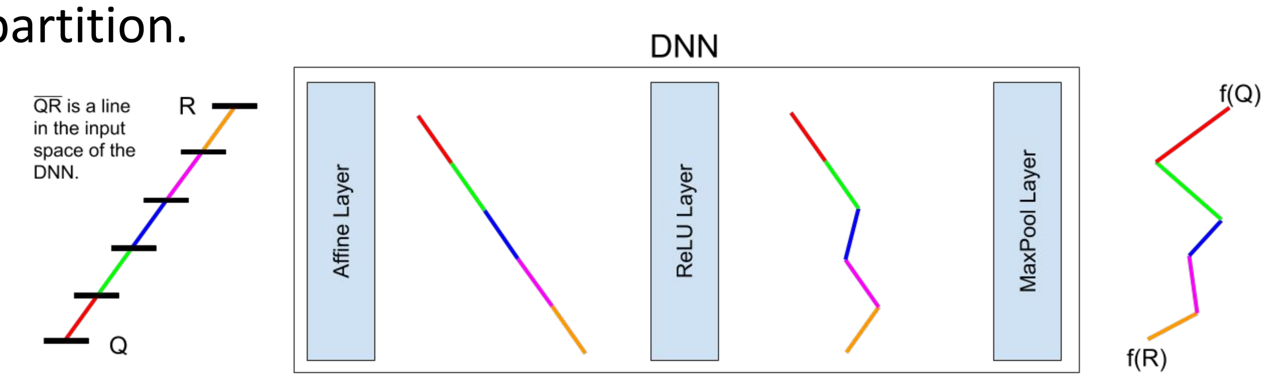
Computing Linear Restrictions of Neural Networks

Matthew A. Sotoudeh masotoudeh@ucdavis.edu
Aditya V. Thakur avthakur@ucdavis.edu
Davis Automated Reasoning Group

ExactLine

Translates questions about DNN to questions about affine functions.

- Succinct representation of DNN over line in the input space
- Partitioning of line where the output of the DNN is an affine function of the input within each partition.



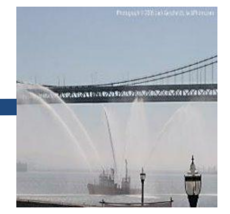
Exactly Computing Integrated Gradients

Summary: IG is an attribution method for deep neural networks which relies on computing an integral over a linear path in the input space. Prior work was unable to exactly compute this integral, instead approximating it with a left-Riemann sum. ExactLine can be use to exactly compute the integral, allowing us to show the first real-world relative error rates of the approximation (25-45%) and empirical evidence that trapezoidal sampling would be more efficient.

Integrated Gradients is a well-known attribution method.
Formal definition satisfies many desirable properties, but relies on computing path integral of network's gradients between image and a baseline.
Prior work unable to compute the integral, so approximated with left Riemann sum and number of samples chosen according to heuristic (**unknown accuracy!**).

True IG: $$IG_i(x) \stackrel{\text{def}}{=} (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

Approximate IG: $$\widetilde{IG}_i^m \stackrel{\text{def}}{=} (x_i - x'_i) \times \sum_{0 \le k < m} \frac{\partial F(x' + \frac{k}{m} \times (x - x'))}{\partial x_i} \times \frac{1}{m}$$



Baseline x' Image x

Key insight: gradient is constant within each ExactLine partition, so *exact* integral can be computed.
Results:
- Current best-practice results in 25-45% relative error.
- Trapezoidal sampling requires 20-40% fewer samples to get to 5% relative error.

	Error (%)
convsmall	24.95
convmedium	24.05
convbig	45.34

Error of prior best practice.

	Exact	Approximate		
		left	right	trap.
convsmall	2582.74	136.74	139.40	91.67
convmedium	3578.89	150.31	147.59	91.57
convbig	14064.65	269.43	278.20	222.79

Number of samples needed to reach 5% error.

Takeaways:
- Use ExactLine ahead-of-time.
- Use trapezoidal sampling.

Future work:
- Develop new (non-uniform) sampling methods.

Related work:
Sundararajan et al. *Axiomatic Attribution for Deep Networks.* ICML 2017.

Open-Source Python and C++ Library

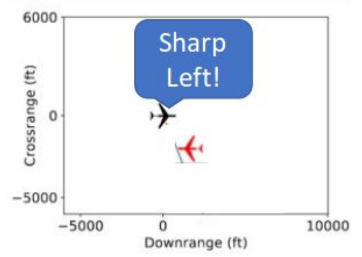
We believe ExactLine will stimulate much future work, and provide a high-quality library for computing it.
- C++, gRPC server optimized with Intel TBB and MKLDNN.
- Python front-end optimized with PyTorch, available on PyPI.
- All experiments reproducible.

https://github.com/95616ARG/SyReNN

Exactly Characterizing DNN Decision Boundaries

Summary: ExactLine can be used to determine the decision boundaries on a line, i.e. all points where the classification changes. We applied this ability to ACAS Xu, a DNN-based aircraft collision avoidance system. We showed that the prior work approach of sampling finitely-many points can miss interesting behavior that shows up clearly with our technique.

ACAS Xu is a DNN intended to be an aircraft collision-avoidance system.
- Inputs: position of ownship/attacker (rho, theta), velocity of ownship/attacker.
- Output classes: strong left, weak left, weak right, strong right, clear-of-conflict.

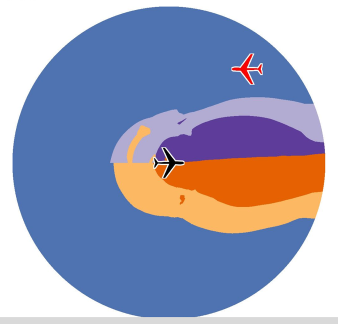


Issue: how to understand the DNN's behavior?

Prior work can (slowly) answer decision problems ("is there any scenario in this region causing it to recommend strong right?"), or sample individual points to approximate decision boundaries.

Key insight: DNN is affine within each ExactLine partition, so we can *exactly* characterize decision boundaries on each line. (Sample in one fewer dimension).

Results:
- Quickly identify behavior missed by sampling methods.
- Better understanding of network.



Future work:
- 2-dimensional visualization performed in arxiv:1908.06223

Related work:
Katz et al. *Reluplex: An Efficient Solver for Verifying Deep Neural Networks*. CAV 2017.
Singh et al. *An Abstract Domain for Certifying Neural Networks.* POPL 2019.

Understanding Adversarial Examples

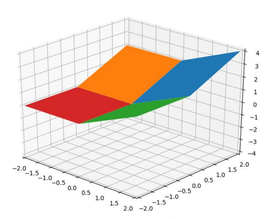
Summary: The Linear Explanation for adversarial examples relies on an assumption that the network is well-approximated by its tangent plane around the natural image. We use ExactLine to empirically falsify this assumption, showing that (1) there are many linear regions around natural images, (2) there are more linear regions in the adversarial direction than a random one, and (3) the tangent plane is a poor approximation for the network along the direction of of the adversarially-perturbed image. We then identify an interesting phenomenon: adversarially-protected networks tend to be *more* linear than unprotected ones.

Adversarial examples are normal inputs imperceptibly perturbed so that the network misclassifies in a controlled way.

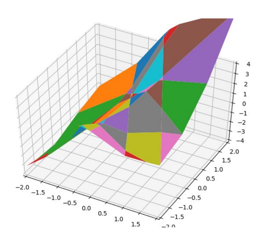
Falsifying the Linear Explanation

Linear Explanation for adversarial examples (ICLR '15) relies on assumption that the network is highly linear (well-approximated by the tangent plane) around normal inputs.

Prior work has taken this assumption for granted, building adversaries based on it (eg. FGSM) or questioning the theoretical conclusions that follow from it. To our knowledge, the underlying assumption has not been empirically verified.

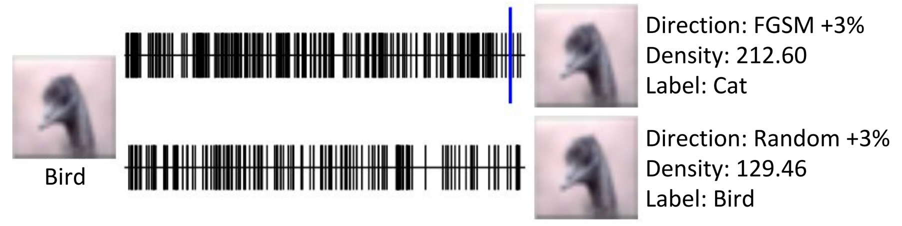


Mostly Linear



Very Non-Linear

Key insight: ExactLine can be used to check a necessary condition for the linear assumption to hold, namely that the network is highly linear on one-dimensional line segments around normal inputs.

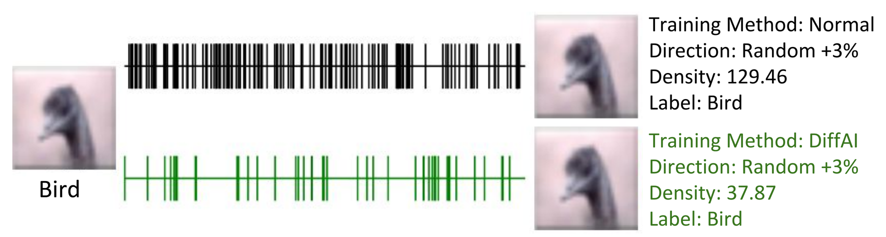


Results:
- Actually, highly **non**-linear. FGSM direction is especially non-linear.
- Furthermore, gradient at normal image is poor predictor of gradients along the line.

Network Linearity Associated with Robustness

We used ExactLine to **compare linearity of normally-trained networks to adversarially-trained ones** (eg. DiffAI).

Results:
- Network trained with DiffAI and PGD protections tend to be more linear.



Takeaways:
- Linearity assumption is **wrong**.
- Linearity seems to be associated with robustness.

Future work:
- Investigate relationship between linearity and robustness.
- Propose (and test!) new hypotheses.

Related work:
Goodfellow et al. *Explaining and Harnessing Adversarial Examples.* ICLR 2015.
Tanay et al. *A Boundary Tilting Perspective on the Phenomenon of Adversarial Examples.*
Mirman et al. *Differentiable Abstract Interpretation for Provably Robust Neural Networks.* ICML 2018.