Differentiable Abstract Interpretation for Provably Robust Neural Networks

safeai.ethz.ch



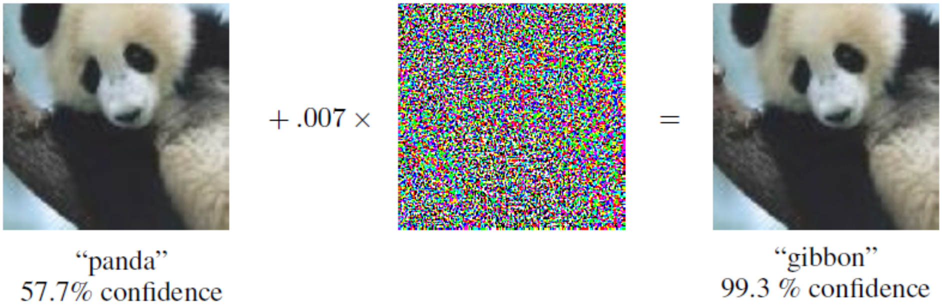
Matthew Mirman Timon Gehr Martin Vechev

SECURE, RELIABLE, INTELLIGENT SYSTEMS

ETH Zürich

ICML 2018

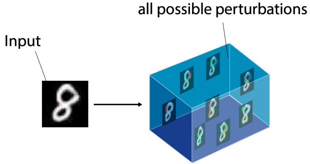
Adversarial Attack



Example of FGSM attack produced by Goodfellow et al. (2014)

L_∞ Adversarial Ball

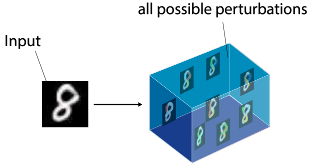
Many developed attacks: Goodfellow et al. (2014); Madry et al. (2018); Evtimov et al. (2017); Athalye & Sutskever (2017); Papernot et al. (2017); Xiao et al. (2018); Carlini & Wagner (2017); Yuan et al. (2017); Tramèr et al. (2017)



$$\text{Ball}_\epsilon(input) = \{attack \mid \|input - attack\|_\infty \leqslant \epsilon\}$$

L_∞ Adversarial Ball

Many developed attacks: Goodfellow et al. (2014); Madry et al. (2018); Evtimov et al. (2017); Athalye & Sutskever (2017); Papernot et al. (2017); Xiao et al. (2018); Carlini & Wagner (2017); Yuan et al. (2017); Tramèr et al. (2017)

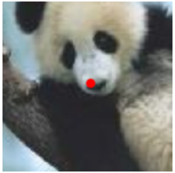


$$\text{Ball}_\epsilon(input) = \{attack \mid \|input - attack\|_\infty \leqslant \epsilon\}$$

A net is *ϵ-robust* at x if it classifies every example in $\text{Ball}_\epsilon(x)$ the same and correctly

Adversarial Ball

Is attack $\in \text{Ball}_\epsilon(\text{panda})$?

		attack	
ϵ			
0.1	$\in$	$\notin$	$\notin$
0.5	$\in$	$\in$	$\notin$

Prior Work

Increase Network Robustness

Defense: Train a network so that *most* inputs are *mostly* robust.

- Madry et al. (2018); Tramèr et al. (2017); Cisse et al. (2017); Yuan et al. (2017); Gu & Rigazio (2014)
- Network still attackable

Prior Work

Increase Network Robustness

Defense: Train a network so that *most* inputs are *mostly* robust.

- Madry et al. (2018); Tramèr et al. (2017); Cisse et al. (2017); Yuan et al. (2017); Gu & Rigazio (2014)
- Network still attackable

Certify Robustness

Verification: Prove that a network is ϵ-robust at a point

- Huang et al. (2017); Pei et al. (2017); Katz et al. (2017); Gehr et al. (2018)
- Experimentally robust nets not very *certifiably* robust
- Intuition: not all correct programs are provable

Problem Statement

Train a Network to be *Certifiably* Robust[1]

Given:

- Net_θ with weights θ
- Training inputs and labels

Find:

- θ that maximizes number of inputs we can *certify* are ϵ-robust

[1] Also addressed by: Raghunathan et al. (2018); Kolter & Wong (2017); Dvijotham et al. (2018)

Problem Statement

Train a Network to be *Certifiably* Robust[1]

Given:

- Net_θ with weights θ
- Training inputs and labels

Find:

- θ that maximizes number of inputs we can *certify* are ϵ-robust

Challenge

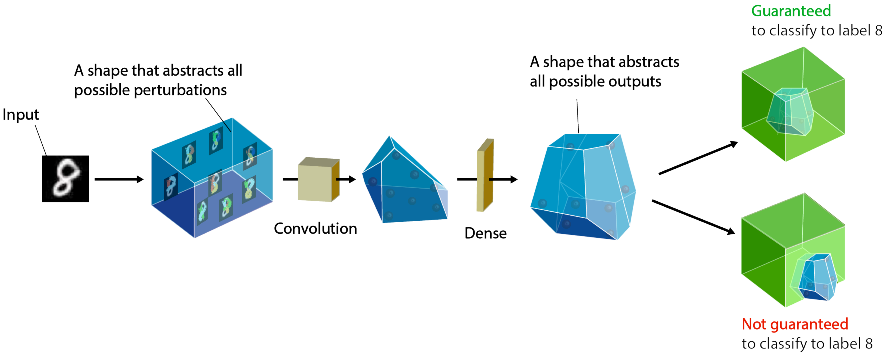
- At least as hard as standard training!

[1] Also addressed by: Raghunathan et al. (2018); Kolter & Wong (2017); Dvijotham et al. (2018)

High Level

Make certification the training goal

- Abstract Interpretation: certify by over-approximating output [2]

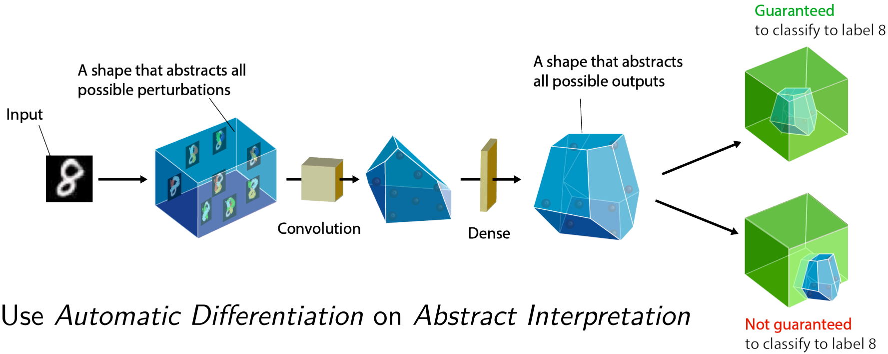


[2]Cousot & Cousot (1977); Gehr et al. (2018)
Image Credit: Petar Tsankov

High Level

Make certification the training goal

- Abstract Interpretation: certify by over-approximating output [2]



- Use *Automatic Differentiation* on *Abstract Interpretation*

---

[2]Cousot & Cousot (1977); Gehr et al. (2018)
Image Credit: Petar Tsankov

Abstract Interpretation

Cousot & Cousot (1977)

Abstract Interpretation is heavily used in industrial large-scale program analysis to compute over-approximation of program behaviors [3]

[3]For example by Astrée: Blanchet et al. (2003)

[4]$f[\gamma(d)] \subseteq \gamma(f^{\#}(d))$ where $f[s]$ is the image of s under f

Abstract Interpretation

Cousot & Cousot (1977)

Abstract Interpretation is heavily used in industrial large-scale program analysis to compute over-approximation of program behaviors [3]

Provide

- abstract domain $\mathcal{D}$ of abstract points d
- concretization function $\gamma : \mathcal{D} \to \mathcal{P}(\mathbb{R}^n)$
- concrete function $f : \mathbb{R}^n \to \mathbb{R}^n$

Develop a sound[4] abstract transformer $f^{\#} : \mathcal{D} \to \mathcal{D}$

[3] For example by Astrée: Blanchet et al. (2003)

[4] $f[\gamma(d)] \subseteq \gamma(f^{\#}(d))$ where $f[s]$ is the image of s under f

Abstract Interpretation

Cousot & Cousot (1977)

Abstract Interpretation is heavily used in industrial large-scale program analysis to compute over-approximation of program behaviors [3]

Provide

- abstract domain $\mathcal{D}$ of abstract points d
- concretization function $\gamma : \mathcal{D} \to \mathcal{P}(\mathbb{R}^n)$
- concrete function $f : \mathbb{R}^n \to \mathbb{R}^n$

Develop a sound[4] abstract transformer $f^\# : \mathcal{D} \to \mathcal{D}$

- $\mathrm{ReLU} : \mathbb{R}^n \to \mathbb{R}^n$ becomes $\mathrm{ReLU}^\# : \mathcal{D} \to \mathcal{D}$

[3] For example by Astrée: Blanchet et al. (2003)

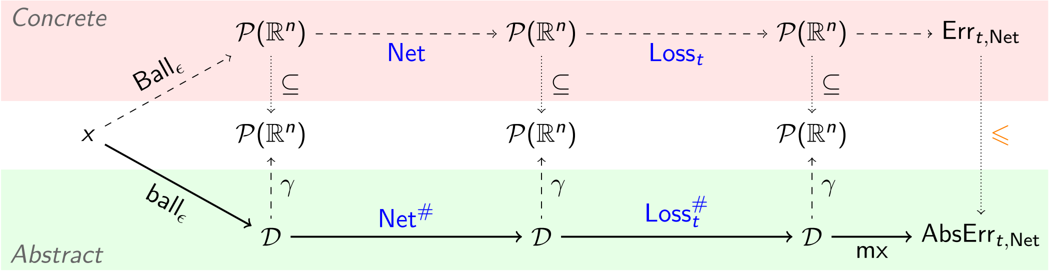
[4] $f[\gamma(d)] \subseteq \gamma(f^\#(d))$ where $f[s]$ is the image of s under f

Abstract Optimization Goal

Given

- $\mathrm{mx}(d)$: a way to compute upper bounds for $\gamma(d)$.
- $\mathrm{ball}(x) \in \mathcal{D}$: a ball abstraction s.t. $\mathrm{Ball}_\epsilon(x) \subseteq \gamma(\mathrm{ball}(x))$
- Loss_t: an abstractable traditional loss function for classification target t

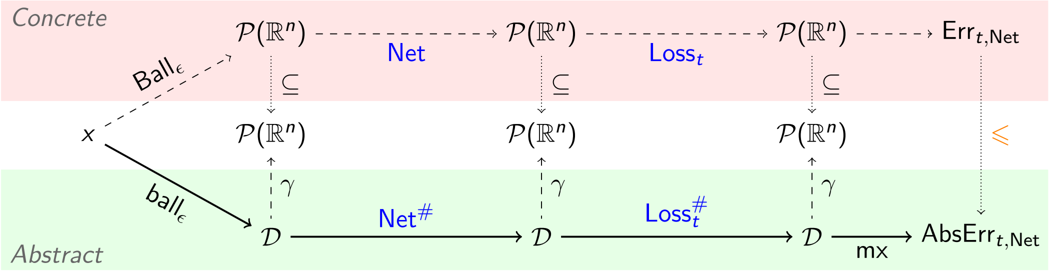
$$
\begin{aligned}
\mathrm{Err}_{t,\mathrm{Net}}(x) &= \quad \mathrm{Loss}_t \circ \mathrm{Net}(x) && \text{classical error} \\
\mathrm{AbsErr}_{t,\mathrm{Net}}(x) &= \mathrm{mx} \circ \mathrm{Loss}_t^{\#} \circ \mathrm{Net}^{\#} \circ \mathrm{ball}(x) && \text{abstract error}
\end{aligned}
$$



Using Abstract Goal

Theorem

$Err_{t,Net}(y) \leqslant AbsErr_{t,Net}(x)$ for all points $y \in Ball_\epsilon(x)$

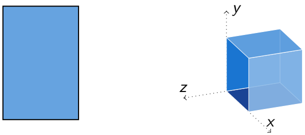


Abstract Domains

- Many abstract domains $\mathcal{D}$ with different speed/accuracy tradeoffs
- Transformers must be parallelizable, and work well with SGD

Abstract Domains

- Many abstract domains $\mathcal{D}$ with different speed/accuracy tradeoffs
- Transformers must be parallelizable, and work well with SGD

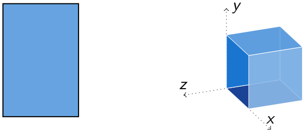


Box Domain

- p dimension axis-aligned boxes
- Ball_ϵ: perfect
- $(\cdot M)^{\#}$: uses abs
- $\text{ReLU}^{\#}$: 6 linear operations, 2 ReLUs

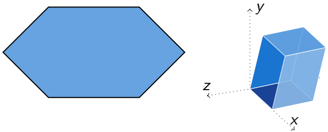
Abstract Domains

- Many abstract domains $\mathcal{D}$ with different speed/accuracy tradeoffs
- Transformers must be parallelizable, and work well with SGD



Box Domain

- p dimension axis-aligned boxes
- Ball_ϵ: perfect
- $(\cdot M)^{\#}$: uses abs
- $\text{ReLU}^{\#}$: 6 linear operations, 2 ReLUs



Zonotope Domain

- Affine transform of k-cube onto p dims
- k increases with non-linear transformers
- Ball_ϵ: perfect
- $(\cdot M)^{\#}$: perfect
- $\text{ReLU}^{\#}$: zBox, zDiag, zSwitch, zSmooth,
- Hybrid: hSwitch, hSmooth

Implementation

DiffAI Framework

- Can be found at: safeai.ethz.ch
- Implemented in PyTorch[5]
- Tested with modern GPUs

[5] Paszke et al. (2017)

Scalability

CIFAR10

			Train 1 Epoch (s)			Test 2k Pts (s)	
Model	#Neurons	#Weights	Base	Attack[6]	Box	Box	hSwitch
ConvSuper[7]	~124k	~16mill	23	149	74	0.09	40

- Can use a less precise domain for training than for certification
- Can test/train Resnet18[8]: 2k points tested on ~500k neurons in ~1s with Box
- tldr: can test and train with larger nets than prior work

[6] 5 iterations of PGD Madry et al. (2018) for both training and testing
[7] ConvSuper: 5 layers deep, no Maxpool.
[8] like that described by He et al. (2016) but without pooling or dropout.

Robustness Provability

MNIST with $\epsilon = 0.1$ on ConvSuper

Training Method	%Correct	%Attack Success	%hSwitch Certified
Baseline	98.4	2.4	2.8
Madry et al. (2018)	98.8	1.6	11.2
Box	99.0	2.8	96.4

- Usually loses only small amount of accuracy (sometimes gains)
- Significantly increases provability[9]

[9] Much more thorough evaluation in appendix of Mirman et al. (2018).

hSmooth Training

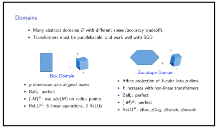
FashionMNIST with $\epsilon = 0.1$ on FFNN

Method	Train Total (s)	%Correct	%zSwitch Certified
Baseline	119	94.6	0
Box	608	8.6	0
hSmooth	4316	84.4	21.0

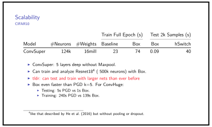
- Training unexpectedly fails with Box (very rare)
- Training slow but reliable with hSmooth

Conclusion

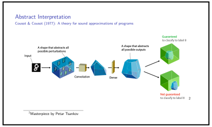
First application of *automatic differentiation* to *abstract interpretation* (that we know of)



Trained and verified the *largest* verifiable neural networks to date



A way to train networks on regions, not just points[10]



[10]Further examples of this use-case in paper

Bibliography I

Athalye, A. and Sutskever, I. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

Blanchet, B., Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., and Rival, X. A static analyzer for large safety-critical software. In *Programming Language Design and Implementation (PLDI)*, 2003.

Carlini, N. and Wagner, D. A. Adversarial examples are not easily detected: Bypassing ten detection methods. *CoRR*, abs/1705.07263, 2017. URL `http://arxiv.org/abs/1705.07263`.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pp. 854–863, 2017.

Cousot, P. and Cousot, R. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Symposium on Principles of Programming Languages (POPL)*, 1977.

Bibliography II

Dvijotham, K., Gowal, S., Stanforth, R., Arandjelovic, R., O'Donoghue, B., Uesato, J., and Kohli, P. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*, 2018.

Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., and Song, D. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 2017.

Gehr, T., Mirman, M., Tsankov, P., Drachsler Cohen, D., Vechev, M., and Chaudhuri, S. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *Symposium on Security and Privacy (SP)*, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Goubault, E. and Putot, S. Static analysis of numerical algorithms. In *International Static Analysis Symposium (SAS)*, 2006.

Bibliography III

Gu, S. and Rigazio, L. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification (CAV)*, 2017.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, 2017.

Kolter, J. Z. and Wong, E. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. 2018.

Bibliography IV

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning (ICML)*, 2018.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Asia Conference on Computer and Communications Security*. ACM, 2017.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pei, K., Cao, Y., Yang, J., and Jana, S. Deepxplore: Automated whitebox testing of deep learning systems. In *Symposium on Operating Systems Principles*, 2017.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Bibliography V

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. 2018.

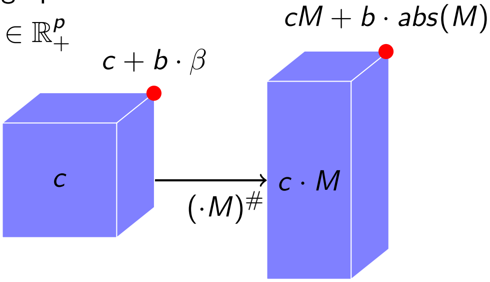
Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514*, 2018.

Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., and Song, D. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

Yuan, X., He, P., Zhu, Q., Bhat, R. R., and Li, X. Adversarial examples: Attacks and defenses for deep learning. *arXiv preprint arXiv:1712.07107*, 2017.

Box Domain

- Interval for each of the p nodes in network graph
- Represented by center $c \in \mathbb{R}^p$ and radius $b \in \mathbb{R}_+^p$
- Concretization[11]:
  $\gamma_I(\langle c, b \rangle) = \{c + b \odot \beta \mid \beta \in [-1, 1]^p\}$
- Constant matrix multiply transformer[12]:
  $(\cdot M)^\#(\langle c, b \rangle) = \langle c \cdot M, b \cdot abs(M) \rangle$
- ReLU$^\#$: 6 linear operations, 2 ReLUs



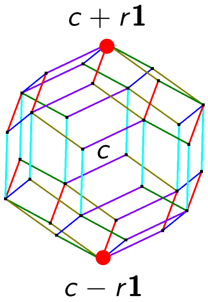
Concretization of a Box
red dot where $\beta = (1, 1, 1)$

[11] $\odot$ is pointwise multiply
[12] $p = m \times n$ and $M \in \mathbb{R}^{n \times w}$

Zonotope Domain

Goubault & Putot (2006)

- Affine transform of k-dimensional unit-cube onto the p network graph nodes
- Represented by center $c \in \mathbb{R}^{p \times 1}$ and k *error terms* $r \in \mathbb{R}^{p \times k}$
- Concretization:
  $\gamma_Z(\langle c, r \rangle) = \{c + re \mid e \in [-1, 1]^{k \times 1}\}$
- Constant matrix multiply transformer[13]:
  $(\cdot M)^{\#}(\langle c, r \rangle) = \langle c * M, r * M \rangle$
- ReLU$^{\#}$: zBox, zDiag, zSwitch, zSmooth

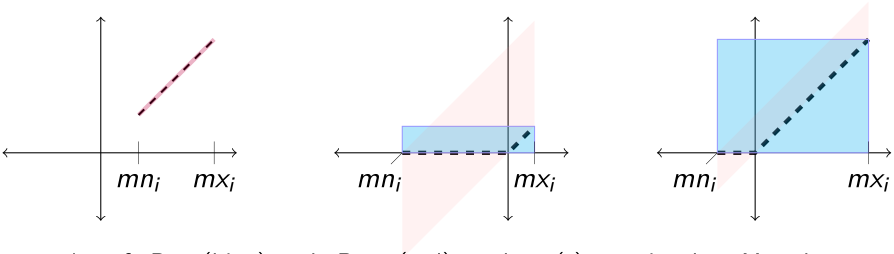


[13] for $p = m \times n$ and $M \in \mathbb{R}^{n \times w}$ and $*$ is batched matrix multiply
Zonotope Image uploaded to Wikipedia by user Tomruen and licensed under CC

Zonotope Domain

SGD Suitable ReLU Transformers

- zBox: Treat as Box when surrounding zero
- zDiag: Add possible error when surrounding zero



Three examples of zBox (blue) and zDiag (red), with in (i) visualized on X and out on Y axis. Dashed line is $ReLU(in)$

- zSwitch: Choose between zBox and zDiag to use based on volume heuristic
- zSmooth: Linear combination of zBox and zDiag based on volume heuristic

Hybrid Zonotope

- Zonotope ReLU transformers all introduce a new error terms for every node
- Hybrid Zonotope: minkowski sum of a p-box with k-zonotope
- k fixed to be number of pixels
- ReLU$^{\#}$: hSwitch, hSmooth

Prior Results

System	Model	#Neurons	#Weights	Train 1 Epoch (s)
DiffAI	ConvSuper	∼124k	∼16mill	74
	Resnet18	∼500k	∼15mill	93
	ConvHuge	∼500k	∼65mill	142
Wong et al. (2018)	Large	∼62k	∼2.5mill	466
	Resnet	∼107k	∼4.2mill	1685
Wong & Kolter (2018)	MNIST Conv	∼4k	∼10k	180
Raghunathan et al. (2018)	MNIST 2 layer FFNN	∼1k	∼650k	-
Dvijotham et al. (2018)	Convnets	∼21k	∼650k	-

- Numbers as reported by prior work and not rerun on our hardware
- When hidden unit numbers and weight numbers were included, they were approximated using the network specifications in the paper with over-approximations where the specifications were not complete as in Dvijotham et al. (2018); Raghunathan et al. (2018)

Ongoing Work

- More provability for deeper networks
- Sound testing w/ respect to floating point
- Inferring maximal provability ϵ