ASSURING AUTONOMY
INTERNATIONAL PROGRAMME

2.3.3 – Verification of the learned model

Practical guidance – cross-domain

Authors: John Grese and Dr Corina Pasareanu (CyLab, Carnegie Mellon University)

Neural network verification for safety-critical applications

Deep neural networks (DNNs) are used increasingly in safety-critical applications such as autonomous transport. However, it has been shown that DNNs can produce unsafe predictions when adversarial or unexpected inputs are provided [3], which raises serious safety concerns. To address these concerns, DNNs need to be extensively tested and verified to ensure that they meet important safety requirements. Formal verification methods based on Satisfiability Modulo Theory (SMT) [1][5] and Mixed-Integer Linear Programming (MILP) [6][7] can provide sound assurances that no adversarial examples exist within a given neighbourhood of an input. We discuss here how formal verification tools, such as Marabou [1], in conjunction with clustering techniques, can be used to evaluate a network's sensitivity and robustness against small perturbations to provide better safety guarantees. This entry is based on [4].

Local robustness

Local robustness can be characterised by the minimum perturbation ($\boldsymbol{\delta}$) applied to an individual input ($\boldsymbol{x}$) which causes a change in the network's prediction. Formal verification tools can be used to both discover the minimum value of $\boldsymbol{\delta}$ which causes the change in prediction, and to prove the network is robust with respect to a specified $\boldsymbol{\delta}$. Local robustness can be useful in understanding how much a given input can be changed (possibly by an adversary) before the network's prediction will change. Equation 1 describes the formal guarantee provided by local robustness checks, which states that for any $\boldsymbol{x}'$ such that the distance from the input $\boldsymbol{x}$ is less than $\boldsymbol{\delta}$, the network will produce the same output.

$$\forall x\prime \ s.t. \ \|x\prime - x\| < \delta \Rightarrow f(x\prime) = f(x).$$

Equation 1

Targeted robustness

Targeted robustness can be characterised by the minimum perturbation ($\boldsymbol{\delta}$) applied to an input ($\boldsymbol{x}$) which causes a specific change in the network's prediction. In the case of a classifier, this could mean that a specific label does not exist within the verified region $\boldsymbol{x} \pm \boldsymbol{\delta}$. For example, consider a classifier which predicts the driver's reaction time as slow, medium or fast. In a classifier such as this, targeted robustness checks could be used to prove that a fast input will not be labelled as slow within the radius $\boldsymbol{x} \pm \boldsymbol{\delta}$. This would be helpful in providing guarantees that the worst-case prediction does not occur within a specified $\boldsymbol{\delta}$. The formal guarantee provided by targeted robustness can be seen in equation 2, which states that for all $\boldsymbol{x}'$ with a distance from $\boldsymbol{x}$ less than $\boldsymbol{\delta}$, the target does not exist.

$$\forall x\prime \; s.t. \; \|x\prime - x\| < \delta \Rightarrow f(x\prime) \neq target.$$

Equation 2

Sensitivity

Sensitivity is similar to robustness, but is characterised by the minimum perturbation ($\boldsymbol{\delta}$) applied to an individual feature, i.e., input component $\boldsymbol{x}_i$, of a given input $\boldsymbol{x}$ which causes a change in the network's prediction. A smaller $\boldsymbol{\delta}$ indicates that the network is more sensitive to changes in feature $\boldsymbol{x}_i$, and a larger $\boldsymbol{\delta}$ indicates that the network is less sensitive. Using formal verification for this type of analysis on a network's sensitivity has the benefit of providing formal guarantees that no adversarial inputs exist within the specified $\boldsymbol{\delta}$. Formal verification tools can be used to both discover the minimum value of $\boldsymbol{\delta}$, and prove that the network is robust within a specified $\boldsymbol{\delta}$. Using the discovered value $\boldsymbol{\delta}$, one can infer how sensitive a given feature is to adversarial perturbations, and also gain insight into the importance of the feature with respect to the network's prediction. Equation 3 describes the guarantee provided by formal verification on a given feature $\boldsymbol{x}$i of input $\boldsymbol{x}$ with respect to the minimum perturbation $\boldsymbol{\delta}$.

$$\forall x_i\prime \;\; \text{s.t.} \;\; \|x_i\prime - x_i\| < \delta, \;\; f([x_0 \ldots x_i\prime \ldots x_n]) = f([x_0 \ldots x_i \ldots x_n])$$

Equation 3

An asymmetric variant of sensitivity checking can be performed to identify how the network behaves with respect to positive vs negative perturbations to $\boldsymbol{x}_i$. In an asymmetric sensitivity test, a feature $\boldsymbol{x}_i$ is perturbed by independent values for the negative and positive perturbations ($\boldsymbol{x}_i$-$\boldsymbol{\delta}_{neg}$ and $\boldsymbol{x}_i$+$\boldsymbol{\delta}_{pos}$). The asymmetric sensitivity analysis can yield additional information about the model, and can identify biases in the network.

Data-driven verification

Both sensitivity and robustness (including targeted robustness) can be checked automatically with a formal verification tool like Marabou [1]. However, the obtained guarantees are only for small regions around individual points, in which the network is proven to produce the same output. Data-driven verification techniques such as DeepSafe [2] can be used to provide assurances that no adversarial inputs exist within larger regions of the input space. Essentially, the technique performs clustering on the dataset to identify regions with similar points that have the same label. Figure 4 shows an example of a clustering algorithm called "Label Guided K-Means" [2] which produces regions containing a single label.

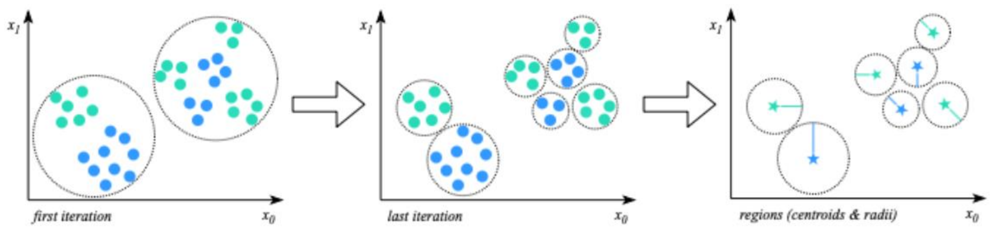

Figure 4: Regions containing a single label produced by "Label Guided K-Means"

Each region produced by Label-Guided K-Means has a centroid, which can then be used as target for verification, with respect to robustness or targeted robustness. The resulting $\boldsymbol{\delta}$ discovered from the verification of the region's centroid can then be used to calculate the "verified radius" of the region. Using the centroids and verified radius r, we can produce a set of "safe regions" which have been formally verified, and proven to contain points of a single label. Intuitively, we expect these regions to be larger than the ones verified through the local robustness checks, since they come from clusters of multiple inputs with the same label. An example of these safe regions is shown in figure 5. The coloured regions illustrate the formally verified "safe regions".

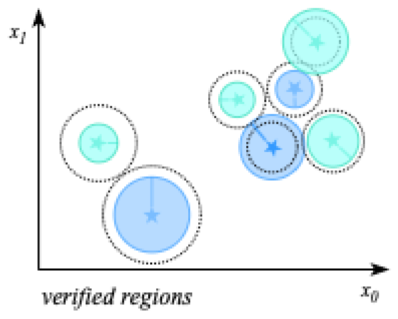

Figure 5: Example of formally verified "safe regions".

The "safe regions" provide the guarantee described by equation 6, which states that for all $\boldsymbol{x}$ such that the distance from the centroid is less than the verified radius r, the network will produce the same label. This method can also be generalised to support different types of distance metrics and networks [2].

$$\forall x \ s.t. \ \|x - centroid\|_{L2} \leq r \Rightarrow f(x) = label.$$

Equation 6

Verification results

Formal verification tools for neural networks typically accept input-output queries as specifications, and provide counterexamples when the queries do not hold. These counterexamples can help improve understanding of the network's behaviour; furthermore, they can be amplified and used to improve robustness and sensitivity through retraining the network. The verified "safe regions" (obtained with the data-driven verification) describe portions of the input space in which the network behaves as expected; they can be potentially used as run-time “guards” during network deployment or can be provided to safety controllers in a searchable format to provide a measure of confidence about the network’s predictions.

References

1. Katz, G., and et al., “The Marabou Framework for Verification and Analysis of Deep Neural Networks”, CAV, 2019.
2. Divya Gopinath, C. S. P. C. B., Guy Katz, “DeepSafe: A Data-driven Approach for Checking Adversarial Robustness in Neural Networks,” ATVA, 2018.
3. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R., “Intriguing Properties of Neural Networks,” ICLR, 2014.
4. Grese, J., Pasareanu, C., and Pakdamanian, E., “Formal Analysis of a Neural Network Predictor in Shared-Control Autonomous Driving”, to appear in 2021 AIAA SciTech Forum.
5. Katz, G., Barrett, C., Dill, D., Julian, K., and Kochenderfer, M., “Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks”, CAV, 2017.
6. Ehlers, R., “Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks”, ATVA, 2017.
7. Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., and Misener, R., “Efficient Verification of ReLU-Based Neural Networks via Dependency Analysis,” Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 3291–3299.