Comparing Differentiable Logics for Learning Systems: A Research Preview*

Thomas Flinkow
Department of Computer Science
Maynooth University
Maynooth, Ireland
thomas.flinkow@mu.ie

Barak A. Pearlmutter
Department of Computer Science and Hamilton Institute
Maynooth University
Maynooth, Ireland
barak@pearlmutter.net

Rosemary Monahan
Department of Computer Science and Hamilton Institute
Maynooth University
Maynooth, Ireland
rosemary.monahan@mu.ie

Extensive research on formal verification of machine learning (ML) systems indicates that learning from data alone often fails to capture underlying background knowledge. A variety of verifiers have been developed to ensure that a machine-learnt model satisfies correctness and safety properties, however, these verifiers typically assume a trained network with fixed weights. ML-enabled autonomous systems are required to not only detect incorrect predictions, but should also possess the ability to self-correct, continuously improving and adapting. A promising approach for creating ML models that inherently satisfy constraints is to encode background knowledge as logical constraints that guide the learning process via so-called differentiable logics. In this research preview, we compare and evaluate various logics from the literature in weakly-supervised contexts, presenting our findings and highlighting open problems for future work. Our experimental results are broadly consistent with results reported previously in literature; however, learning with differentiable logics introduces a new hyperparameter that is difficult to tune and has significant influence on the effectiveness of the logics.

1 Introduction

Advancements in machine learning (ML) in the past few years indicate great potential for applying ML to various domains. Autonomous systems are one such application domain, but using ML components in such a safety-critical domain presents unique new challenges for formal verification. These include (1) ML failing to learn background knowledge from data alone [16], (2) neural networks being susceptible to adversarial inputs, and (3) a lack of specifications, generally and especially when continuous learning is permitted [3]. Addressing these challenges is even more important and more difficult when the ML-enabled autonomous system is permitted to continue to learn after deployment, either to adapt to changing environments or to correct and improve itself when errors are detected [2].

A multitude of neural network verifiers (c.f. [10]) have been presented in the past few years; prominent examples include Reluplex [6], NNV [12], and MN-BaB [4]. These solvers use techniques such as satisfiability and reachability analysis, and can verify a variety of properties. However, these verifiers typically assume trained networks with fixed weights and do not target the learning process itself [7]. One step in the direction of correct-by-construction neural networks are so-called *differentiable logics*, which transform a logical constraint ϕ into an additional logical loss term $\mathcal{L}_{\mathrm{L}}(\phi)$ to minimise when learning, where the logical loss of a constraint ϕ is combined with standard cross-entropy loss as $\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda \mathcal{L}_{\mathrm{L}}(\phi)$. In order to translate logical constraints into loss terms, a mapping must be defined that allows for real-valued truth values, and that is differentiable almost everywhere for use with standard gradient-based methods. In the following, we give a brief overview of two of these mappings (so-called *differentiable logics*) from popular literature, namely DL2 and fuzzy logics.

*This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 20/FFP-P/8853.

M. Farrell, M. Luckcuck, M. Schwammberger, and
M. Gleirscher (Eds): Fifth International Workshop on
Formal Methods for Autonomous Systems (FMAS 2023)
EPTCS 395, 2023, pp. 17–29, doi:10.4204/EPTCS.395.3

Table 1: The t-norms, t-conorms and implications used in our experiments.

Name	T-norm (Conjunction)	T-conorm (Disjunction)	Implication
Gödel	$T_{\mathrm{G}}(x,y) = \min(x,y)$	$S_{\mathrm{G}}(x,y) = \max(x,y)$	$I_{\mathrm{G}}(x,y) = \begin{cases} 1, & \text{if } x < y \\ y \end{cases}$
			$I_{\mathrm{KD}}(x,y) = \max(\overline{x},y)$
Łukasiewicz	$T_{\mathrm{LK}}(x,y) = \max(0, x+y-1)$	$S_{\mathrm{LK}}(x,y) = \min(1, x+y)$	$I_{\mathrm{LK}}(x,y) = \min(\overline{x}+y, 1)$
Yager	$T_{\mathrm{YG}}(x,y) = \max(0, \overline{\overline{x} \mathbin{\triangle}_p \overline{y}})$	$S_{\mathrm{YG}}(x,y) = \min(1, x \mathbin{\triangle}_p y)$	$I_{\mathrm{YG}}(x,y) = \begin{cases} 1, & \text{if } x = y = 0 \\ y^x \end{cases}$
Product	$T_{\mathrm{P}}(x,y) = xy$	$S_{\mathrm{PS}}(x,y) = x+y-xy$	$I_{\mathrm{GG}}(x,y) = \begin{cases} 1, & \text{if } x < y \\ y/x \end{cases}$
			$I_{\mathrm{RC}}(x,y) = \overline{x}+xy$

where $u \mathbin{\triangle}_p v := \sqrt[p]{|u|^p + |v|^p}$ is the p-norm Pythagorean sum, $p \geq 1$, and $\overline{u} := 1-u$

1.1 DL2

DL2 ("Deep Learning with Differentiable Logics") [5] is a system for querying and training neural networks with logic. It maps absolute truth to 0 and other degrees of truth to positive values up to ∞ based on the following elementary translation rules:

$$\mathcal{L}_{\mathrm{DL2}}(x \leq y) := \max(x-y, 0), \quad \mathcal{L}_{\mathrm{DL2}}(x \wedge y) := \mathcal{L}_{\mathrm{DL2}}(x) + \mathcal{L}_{\mathrm{DL2}}(y), \quad \mathcal{L}_{\mathrm{DL2}}(x \vee y) := \mathcal{L}_{\mathrm{DL2}}(x) \cdot \mathcal{L}_{\mathrm{DL2}}(y). \tag{1}$$

Additionally, there is $\mathcal{L}_{\mathrm{DL2}}(x \neq y) := \xi[x = y]$, where $\xi > 0$ denotes a constant that was found to not have significant influence [5], and $[\cdot]$ being the indicator (in Knuth's notation). From these, other rules such as $\mathcal{L}_{\mathrm{DL2}}(x < y) := \mathcal{L}_{\mathrm{DL2}}(x \leq y \wedge x \neq y)$ can be derived. Negation is handled by pushing the negation inwards to the level of comparison, e.g. $\mathcal{L}_{\mathrm{DL2}}(\neg(x \leq y)) := \mathcal{L}_{\mathrm{DL2}}(y < x)$. DL2 does not have a separate translation for implication, instead translating implication as $\mathcal{L}_{\mathrm{DL2}}(x \to y) := \mathcal{L}_{\mathrm{DL2}}(\neg x \vee y)$.

1.2 Fuzzy Logics

Whereas DL2 was designed specifically for deep learning contexts, fuzzy logics are logical systems that have been studied extensively and happen to be suitable for use as differentiable logics due to their many-valued nature, with operators that are often differentiable almost everywhere. Fuzzy logics express degrees of truth in the unit interval $[0,1]$, with absolute truth mapped to 1. We use $\mathcal{L}_{\mathrm{L}}(\phi) := 1.0 - \mathcal{L}_{\mathrm{FL}}(\phi)$ for the fuzzy logic loss in our implementation to address the inverse notion of truth.

Fuzzy logics are based on functions $T : [0,1]^2 \to [0,1]$ that are commutative, associative, monotonic, and satisfy $T(1,y) = y$. These are called triangular norms (abbreviated as *t-norms*) and generalise conjunction. A t-conorm (also called *s-norm*) generalises disjunction and can be obtained from a t-norm using $S(x,y) = 1 - T(1-x, 1-y)$. From a t-conorm S and fuzzy negation N, one obtains a so-called (S,N)-implication (which generalises material implication) as $I(x,y) := S(N(x),y))$. Examples of (S,N)-implications are the Kleene-Dienes implication I_{KD} and Reichenbach implication I_{RC}, both with the standard negation $N(x) = 1-x$. Other implications generalise the intuitionistic implication and are called R-implications, because they use the t-norm residuum $R(x,y) = \sup\{t \in [0,1] \mid T(x,t) \leq y\}$. Example R-implications are the Gödel implication I_{G} and Goguen implication I_{GG}. The Łukasiewicz implication I_{LK} is both an (S,N)-implication and an R-implication. Other implications are neither—the Yager implication

I_{YG}, for example, is an f-generated implication that is obtained using $f(x) = -\ln x$ in $I(x,y) := f^{-1}(xf(y))$ (with the understanding that $0 \cdot \infty = 0$).

Additionally, [14] propose *sigmoidal* implications in order to prove the derivatives of the original implication, while preserving its characteristics. In Eq. (2), $\sigma(x) := 1/(1+\exp(-x))$ denotes the standard sigmoidal function and s is a parameter controlling the steepness. We use the sigmoidal implication in our experiments with I_{RC} and $s = 9$, as suggested by [14].

$$(I(x,y))_s := \frac{(1+\exp(s/2))\sigma(sI(x,y)-s/2)-1}{\exp(s/2)-1} \tag{2}$$

Lastly, given a fuzzy implication I and bijection $\phi : [0,1]^2 \to [0,1]$, [1] show that the function $(I(x,y))_\phi := \phi^{-1}I(\phi(x),\phi(y))$ is also a fuzzy implication. We use this in our experiments with the Reichenbach implication I_{RC} and $\phi(x) = x^2$.

Table 1 lists the definitions of the mentioned t-norms, t-conorms and implications, and Fig. 3 (Appendix A) displays plots of the implications.

Mapping atomic terms DL2 is designed for atomic terms that are inequalities or comparisons. As seen in Eq. (1), it provides the loss translation $\mathcal{L}_{\mathrm{DL2}}(x \le y) := \max(x-y,0)$ for comparison.

Fuzzy logics typically do not define fuzzy comparison operators. However, [11] introduce a mapping $\mathcal{L}_{\mathrm{FL}}(x \le y) := 1 - \max\left(\frac{x-y}{x+y}, 0\right)$ for fuzzy logics. Fuzzy logics requires the truth values of the atomic terms x, y to be mapped into $[0,1]$ by some oracle. Because the atomic terms in our constraints are comparisons, we change this mapping from $\mathcal{L}_{\mathrm{FL}}(x \le y) : [0,1]^2 \to [0,1]$ to $\mathcal{L}_{\mathrm{FL}}(x \le y) : \mathbb{R}^2 \to [0,1]$, allowing us to forgo the need for an external oracle. The mapping is shown in Eq. (3) below, where we use $\varepsilon = 0.05$.

$$\mathcal{L}_{\mathrm{FL}}(x \le y) := 1 - \frac{\max(x-y,0)}{|x|+|y|+\varepsilon} \tag{3}$$

Note that the fuzzy logic mapping $\mathcal{L}_{\mathrm{FL}}(x \le y)$ has a property we intuitively might wish to hold: for example, we might want $21 \le 20$ to be as much of a violation as $21000 \le 20000$. This cannot be achieved in DL2, where the violation depends only on the absolute difference.

2 Comparing Differentiable Logics: Experimental Setup

Our comparison experiment[1] is implemented in PyTorch and based on the original experiment in [5]. We train on the Fashion-MNIST, CIFAR-10, and GTSRB data sets with various constraints. In order to create meaningful scenarios where learning with logical constraints surpasses the baseline (learning from data alone), we train with a fraction of the data sets, namely 10 % for Fashion-MNIST, 50 % for CIFAR-10, and 90 % for GTSRB (as it consists of more classes, with more imperfect data). Additionally, we introduce 10 % label noise (training with incorrect labels) for all data sets, and apply various image manipulation techniques, such as random cropping, flipping, and colour changes. A batch size of 256 was used for all datasets.

The goal of our experiment is to compare various differentiable logics, including DL2 and popular fuzzy logics, and investigate which logic performs most favourable, focusing specifically on implication and conjunction, as these have noticeable consequences for the learning process: As pointed out by [14],

[1] Available on `https://github.com/tflinkow/dl-comparison`.

background knowledge and constraints are most often of the form "if A, then B". Choosing a suitable implication that performs well is thus an important task to guarantee best learning.
In [15], the authors introduce the *shadow-lifting* property for a conjunction, which requires the truth value of a conjunction to increase when the truth value of a conjunct increases. This property seems highly desirable for learning, as it allows for gradual improvement. For example, the formula $0.1 \wedge 1.0$ should be more true than $0.1 \wedge 0.2$, but the Gödel t-norm $T_{\mathrm{G}}(x,y) = \min(x,y)$ would yield the same truth value in both cases. DL2 uses addition for conjunction, trivially satisfying shadow-lifting. The only t-norm to satisfy the shadow-lifting property is the product t-norm $T_{\mathrm{P}}(x,y) = xy$. However, as noted by [14], its derivative will be low if x and y are both low, making it hard for the learning process to make progress.

2.1 Constraints

Universal quantification In [5], the authors categorise constraints into two distinct schemes: training set constraints, which relate sampled inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ from the training set, and global constraints, which concern inputs in the ε-ball around a particular input. They use a PGD-based approach for universally quantified constraints and are thus limited to robustness properties, as noted by [11]. In [14], the authors relax infinite quantifiers by assuming minibatches to be subsets of an independent distribution and using finite conjunction for universal quantifiers for the minibatch, thus losing soundness. [11] provide a semantics for quantifiers, independent of the concrete differentiable logic and going beyond robustness via expectation minimisation of a probability distribution.

We do not consider global constraints in our experiment and only utilise finite universal quantification via repeated application of conjunction. This is permitted due to t-norms being associative and commutative by definition, and DL2's use of addition for conjunction.

Investigating implication Limited to only training set constraints, the original DL2 experiment has shown some constraints to already be satisfied in the baseline experiments, where learning with logics would only provide minor improvements. Their robustness constraint, for example, was already 94.5% satisfied on Fashion-MNIST for the baseline, compared to 98.36% with DL2.[2]

They also use a class-similarity constraint for the CIFAR-10 network to encode domain knowledge such as "a car is more similar to a truck than to a dog". In their fully-supervised experiment, even the baseline experiment satisfied this constraint quite well, and DL2 was able to only improve constraint accuracy from 93.67% to 99.68%. As all fuzzy implications $I(x,y)$ by definition behave the same for $x = 0$ or $x = 1$, the original constraint[3] is not suitable to compare different implications. Our modified constraint is shown in Eq. (4)[4] and replaces the binary decision by a soft one, checking whether the network output for label l is greater than or equal to $1/\#\text{classes}$.

$$\mathrm{CSim}(\mathcal{N},\boldsymbol{x},\mathrm{Labels}) := \bigwedge_{(l_1,l_2,l_3)\in\mathrm{Labels}} \big((\mathcal{N}(\boldsymbol{x})_{l_1} \geq 1/10) \longrightarrow (\mathcal{N}(\boldsymbol{x})_{l_2} \geq \mathcal{N}(\boldsymbol{x})_{l_3})\big). \tag{4}$$

As explained above, we translate the conjunction using repeated application of the product t-norm for all fuzzy logics in order to only investigate the different mappings for implication.

[2] The only training set constraint where DL2 could significantly improve constraint accuracy in the original experiments (from 5.62% to 99.78% in Fashion-MNIST) is the Lipschitz constraint $\mathrm{Lipschitz}(\mathcal{N},\boldsymbol{x},\boldsymbol{x}',L) := \|\mathcal{N}(\boldsymbol{x}) - \mathcal{N}(\boldsymbol{x}')\|_2 \leq L\|\boldsymbol{x}-\boldsymbol{x}'\|_2$. We do not include this constraint in our experiments as it does not use conjunction nor implication.

[3] $$\mathrm{CSim}(\mathcal{N},\boldsymbol{x},\mathrm{Labels}) := \bigwedge_{(l_1,l_2,l_3)\in\mathrm{Labels}} \big((\operatorname{argmax}\mathcal{N}(\boldsymbol{x}) = l_1) \longrightarrow (\mathcal{N}(\boldsymbol{x})_{l_2} \geq \mathcal{N}(\boldsymbol{x})_{l_3})\big).$$

[4] The definition of Labels is shown in Eq. (6) for Fashion-MNIST, and in Eq. (7) for CIFAR-10, both in Appendix A.

Investigating conjunction For investigating the shadow-lifting effect, we utilise the German traffic sign recognition benchmark (GTSRB) dataset and use a property that forces the network to make confident, strong decisions by requiring all elements of a group of classes to be either very likely or very unlikely. Groups consist of classes of a similar type (e.g. speed limit signs)[5].

$$\text{Group}(\mathcal{N}, \boldsymbol{x}, \varepsilon, \text{Groups}) := \bigwedge_{\{g_i\} \in \text{Groups}} \Big(\sum_i \mathcal{N}(\boldsymbol{x})_{g_i} \leq \varepsilon \ \vee \ \sum_i \mathcal{N}(\boldsymbol{x})_{g_i} \geq 1 - \varepsilon \Big). \tag{5}$$

We use the probabilistic sum t-conorm S_{PS} for fuzzy disjunctions in order to only focus on the conjunction.

3 Results

Table 2a shows the results obtained from running the class-similarity constraint experiment on the Fashion-MNIST and CIFAR-10 networks, and Table 2b shows the results obtained from running the group constraint on GTSRB. For each of these, the displayed prediction and constraint accuracy are obtained by taking the largest of their products from the last 10 epochs. Additionally, Figs. 1 and 2 (Appendix A) show how prediction and constraint accuracy change over time.

What immediately stands out is that when training with any logic, constraint accuracy is significantly improved, while prediction accuracy is always slightly reduced, compared to the baseline experiment. This could be because our constraints fail to capture useful background knowledge that would help with predictions, or, as [5] note, we might observe a similar phenomenon as reported in [13], where adversarial robustness conflicts with standard generalisation.

Our observed results are broadly consistent with trends previously reported in literature. The Gödel and Goguen implications perform badly as we expect many wrong inferences (due to the non-existing derivative of I_{G} with respect to x, and due to the Goguen implication's singularity as $x \to 0$), which manifest in the table as reduced prediction and constraint accuracy.

Comparing DL2 and fuzzy logics, our results indicate that for implication, fuzzy logics are the better choice — for CIFAR-10, even I_{KD} performs better than DL2, although both are rewriting $x \to y$ to $\neg x \vee y$. This difference could be due to a multitude of reasons; the mappings $\mathcal{L}(x \leq y)$ and $\mathcal{L}(x \wedge y)$ are very different for DL2 and fuzzy logics, as is their range. Albeit the most likely reason is a sub-optimal choice of hyperparameter λ, explained in more detail in the next paragraph. For the group constraint, DL2 performs slightly better than any of the fuzzy logics, although the differences between the logics are overall not as noticeable compared to the class-similarity constraint.

Hyperparameter λ Learning with logical loss introduces the hyperparameter λ, which is the logical weight for the total loss calculation. Finding a suitable logical weight λ is crucial for achieving good results, as choosing a sub-optimal value can potentially result in operators that are supposed to perform badly (such as the Gödel implication) performing even better than logics that are supposed to perform best. Our strategy to approximate good values of λ was to run the same experiment (data set, constraint, logic) for the same number of epochs for each logical weight value $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. We then selected the value that yielded the highest combined prediction and constraint accuracy. Tables 2a and 2b show the value of λ we chose for each run, and Figs. 4 to 6 (Appendix A) show the prediction and constraint accuracies for each value of λ.

This approach is very expensive and not feasible for real-world application. Unfortunately, extrapolating from running experiments at a smaller number of epochs does not necessarily transfer over to running

[5] The definition of the set of sets Groups is shown in Eq. (8) (Appendix A).

Table 2: Results. P/C is prediction / constraint accuracy in %.

(a) Class-Similarity constraint.

	Fashion-MNIST			CIFAR-10		
	P	C	λ	P	C	λ
Baseline	77.55	84.31	–	79.06	48.65	–
DL2	77.88	89.15	0.6	78.55	52.21	0.4
I_{G}	63.46	91.30	3.0	77.65	81.56	1.2
I_{KD}	75.39	80.59	0.8	78.82	72.94	0.6
$I_{\mathrm{\L K}}$	64.64	97.28	4.0	76.06	87.75	6.0
I_{GG}	67.25	95.54	3.0	74.83	88.67	10.0
I_{RC}	76.79	92.56	0.8	79.14	80.51	0.8
$(I_{\mathrm{RC}})_{s=9}$	77.06	93.63	0.8	78.30	80.87	0.8
$(I_{\mathrm{RC}})_{\phi=x^2}$	**76.88**	**95.94**	1.0	**78.31**	**90.74**	1.6
I_{YG}	74.14	80.15	1.0	77.81	73.19	0.8

(b) Group constraint.

	GTSRB		
	P	C	λ
Baseline	89.93	49.97	–
DL2	**88.16**	**77.25**	7.0
T_{G}	88.38	74.20	5.0
$T_{\mathrm{\L K}}$	85.26	78.43	5.0
T_{RC}	86.52	77.56	5.0
T_{YG}	87.47	76.47	5.0

the experiment at the desired number of epochs. Further, as can be seen in Figs. 4 to 6 (Appendix A), the resulting graphs are non-monotonic, making it difficult to predict the prediction and constraint accuracy one would obtain with other values of λ.

4 Future Work

Our experiments have shown that learning with differentiable logics can generally improve how much a ML model satisfies a constraint. Imposing logical constraints on the training process in this manner could be a step in the direction of verified ML, allowing the use of continuous-learning in self-improving ML-enabled autonomous systems. It has to be noted that in contrast to formal verifiers, training with logical loss does not formally guarantee properties to hold in all possible cases.

We highlight a few more areas for future work in the following.

Reusing logical constraints during inference Because of the differentiable logics acting as a regulariser during training, any logical constraints imposed on the learning process are unavailable during inference. The trained model can therefore not make use of the logical constraints to check its predictions, for example to attach confidence scores to its predictions.

Probabilistic logics Despite expressing satisfaction of formulas on $[0,1]$, fuzzy logics are inherently not probabilistic, having been designed instead for reasoning in the presence of vagueness. We point to DeepProbLog [9] as one example for a probabilistic logic for use with deep learning. In the context of neural networks, which often output probabilities, it could be more natural to reason about probabilities instead of vagueness, especially for constraints that include probabilities [3].

Properties A common problem with verifying ML is the lack of specifications, as noted by [8, 3]. Most properties in the literature are limited to robustness against slight perturbations, although differentiable logics can not only relate network inputs and outputs, but could also refer to the inner workings of the neural network, such as weights and activation states. A related area is to investigate whether learning with logical loss can be used to show that desired properties continue to hold when retraining the network.

References

[1] Michał Baczyński & Balasubramaniam Jayaram (2008): *Fuzzy Implications*. *Studies in Fuzziness and Soft Computing* v. 231, Springer Verlag, Berlin, doi:10.1007/978-3-540-69082-5.

[2] Chih-Hong Cheng & Rongjie Yan (2021): *Continuous Safety Verification of Neural Networks*. In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1478–1483, doi:10.23919/DATE51398.2021.9473994.

[3] Marie Farrell, Anastasia Mavridou & Johann Schumann (2023): *Exploring Requirements for Software that Learns: A Research Preview*. In Alessio Ferrari & Birgit Penzenstadler, editors: *Requirements Engineering: Foundation for Software Quality - 29th International Working Conference, REFSQ 2023, Barcelona, Spain, April 17-20, 2023, Proceedings*, *Lecture Notes in Computer Science* 13975, Springer, pp. 179–188, doi:10.1007/978-3-031-29786-1_12.

[4] Claudio Ferrari, Mark Niklas Muller, Nikola Jovanovic & Martin Vechev (2022): *Complete Verification via Multi-Neuron Relaxation Guided Branch-and-Bound*, doi:10.48550/arXiv.2205.00263.

[5] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang & Martin Vechev (2019): *DL2: Training and Querying Neural Networks with Logic*. In: *Proceedings of the 36th International Conference on Machine Learning*, PMLR, pp. 1931–1941.

[6] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian & Mykel J. Kochenderfer (2017): *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks*. In Rupak Majumdar & Viktor Kunčak, editors: *Computer Aided Verification*, Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 97–117, doi:10.1007/978-3-319-63387-9_5.

[7] M.Z. Kwiatkowska (2019): *Safety Verification for Deep Neural Networks with Provable Guarantees*. In: *Leibniz International Proceedings in Informatics, LIPIcs*, 140, doi:10.4230/lipics.concur.2019.1.

[8] Martin Leucker (2020): *Formal Verification of Neural Networks?* In Gustavo Carvalho & Volker Stolz, editors: *Formal Methods: Foundations and Applications*, Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 3–7, doi:10.1007/978-3-030-63882-5_1.

[9] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester & Luc De Raedt (2018): *DeepProbLog: Neural Probabilistic Logic Programming*. In: *Advances in Neural Information Processing Systems*, 31, Curran Associates, Inc.

[10] Mark Niklas Müller, Christopher Brix, Stanley Bak, Changliu Liu & Taylor T. Johnson (2022): *The Third International Verification of Neural Networks Competition (VNN-COMP 2022): Summary and Results*, doi:10.48550/arXiv.2212.10376.

[11] Natalia Ślusarz, Ekaterina Komendantskaya, Matthew Daggitt, Robert Stewart & Kathrin Stark (2023): *Logic of Differentiable Logics: Towards a Uniform Semantics of DL*. In: *EPiC Series in Computing*, 94, EasyChair, pp. 473–493, doi:10.29007/c1nt.

[12] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak & Taylor T. Johnson (2020): *NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems*. In Shuvendu K. Lahiri & Chao Wang, editors: *Computer Aided Verification*, Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 3–17, doi:10.1007/978-3-030-53288-8_1.

[13] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner & Aleksander Madry (2018): *Robustness May Be at Odds with Accuracy*. In: *International Conference on Learning Representations*.

[14] Emile van Krieken, Erman Acar & Frank van Harmelen (2022): *Analyzing Differentiable Fuzzy Logic Operators*. *Artificial Intelligence* 302, p. 103602, doi:10.1016/j.artint.2021.103602. arXiv:2002.06100.

[15] Peter Varnai & Dimos V. Dimarogonas (2020): *On Robustness Metrics for Learning STL Tasks*. In: *2020 American Control Conference (ACC)*, pp. 5394–5399, doi:10.23919/ACC45564.2020.9147692.

[16] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang & Suman Jana (2018): *Efficient Formal Safety Analysis of Neural Networks*. In: *Advances in Neural Information Processing Systems*, 31, Curran Associates, Inc.

A Appendix

A.1 Constraints

The full set of labels used in the class-similarity constraint for Fashion-MNIST is shown in Eq. (6), the set of labels for CIFAR-10 is shown in Eq. (7). Note that we have a conjunct for each class so as to rule out cases where the implication would vacuously be true, which does not necessarily reflect real world constraints.

$$\text{Labels}_{\text{Fashion-MNIST}} := \left\{ \begin{array}{c} (\texttt{T-shirt/top},\texttt{Shirt},\texttt{Ankle boot}), \\ (\texttt{Trouser},\texttt{Dress},\texttt{Bag}), \\ (\texttt{Pullover},\texttt{Shirt},\texttt{Sandal}), \\ (\texttt{Dress},\texttt{Coat},\texttt{Bag}), \\ (\texttt{Coat},\texttt{Pullover},\texttt{Shirt}), \\ (\texttt{Sandal},\texttt{Sneaker},\texttt{Dress}), \\ (\texttt{Shirt},\texttt{Pullover},\texttt{Sneaker}), \\ (\texttt{Sneaker},\texttt{Sandal},\texttt{Trouser}), \\ (\texttt{Bag},\texttt{Sandal},\texttt{Dress}), \\ (\texttt{Ankle boot},\texttt{Sneaker},\texttt{T-shirt/top}) \end{array} \right\} \tag{6}$$

$$\text{Labels}_{\text{CIFAR-10}} := \left\{ \begin{array}{c} (\texttt{airplane},\texttt{ship},\texttt{dog}), \\ (\texttt{automobile},\texttt{truck},\texttt{cat}), \\ (\texttt{bird},\texttt{airplane},\texttt{dog}), \\ (\texttt{cat},\texttt{dog},\texttt{frog}), \\ (\texttt{deer},\texttt{horse},\texttt{truck}), \\ (\texttt{dog},\texttt{cat},\texttt{bird}), \\ (\texttt{frog},\texttt{ship},\texttt{truck}), \\ (\texttt{horse},\texttt{deer},\texttt{airplane}), \\ (\texttt{ship},\texttt{airplane},\texttt{deer}), \\ (\texttt{truck},\texttt{automobile},\texttt{airplane}) \end{array} \right\} \tag{7}$$

The definitions of the groups used in the group constraint Eq. (5) for GTSRB are given in Eq. (8).

$$\text{Group}_{\text{Speed Limits}} := \left\{ \begin{array}{c} \texttt{limit 20km/h}, \\ \texttt{limit 30km/h}, \\ \texttt{limit 50km/h}, \\ \texttt{limit 60km/h}, \\ \texttt{limit 70km/h}, \\ \texttt{limit 80km/h}, \\ \texttt{end of limit 80km/h}, \\ \texttt{limit 100km/h}, \\ \texttt{limit 120km/h} \end{array} \right\}, \qquad \text{Group}_{\text{Mandatory Actions}} := \left\{ \begin{array}{c} \texttt{turn right ahead}, \\ \texttt{turn left ahead}, \\ \texttt{ahead only}, \\ \texttt{go straight or right}, \\ \texttt{go straight or left}, \\ \texttt{keep right}, \\ \texttt{keep left}, \\ \texttt{roundabout} \end{array} \right\}$$

$$\text{Group}_{\text{Prohibitions}} := \left\{ \begin{array}{c} \texttt{no passing}, \\ \texttt{no passing for trucks}, \\ \texttt{no way}, \\ \texttt{no way one-way}, \\ \texttt{end of no passing}, \\ \texttt{end of no passing for trucks} \end{array} \right\}, \qquad \text{Group}_{\text{Warnings}} := \left\{ \begin{array}{c} \texttt{caution general}, \\ \texttt{caution curve left}, \\ \texttt{caution curve right}, \\ \texttt{caution curvy}, \\ \texttt{caution bumps}, \\ \texttt{caution slippery}, \\ \texttt{caution narrow road}, \\ \texttt{road work}, \\ \texttt{pedestrians}, \\ \texttt{children crossing}, \\ \texttt{wild animals crossing} \end{array} \right\}$$

$$\text{Groups}_{\text{GTSRB}} := \{\text{Group}_{\text{Speed Limits}}, \text{Group}_{\text{Prohibitions}}, \text{Group}_{\text{Mandatory Actions}}, \text{Group}_{\text{Warnings}}\} \tag{8}$$

A.2 Plots of Prediction and Constraint Accuracy Over Time

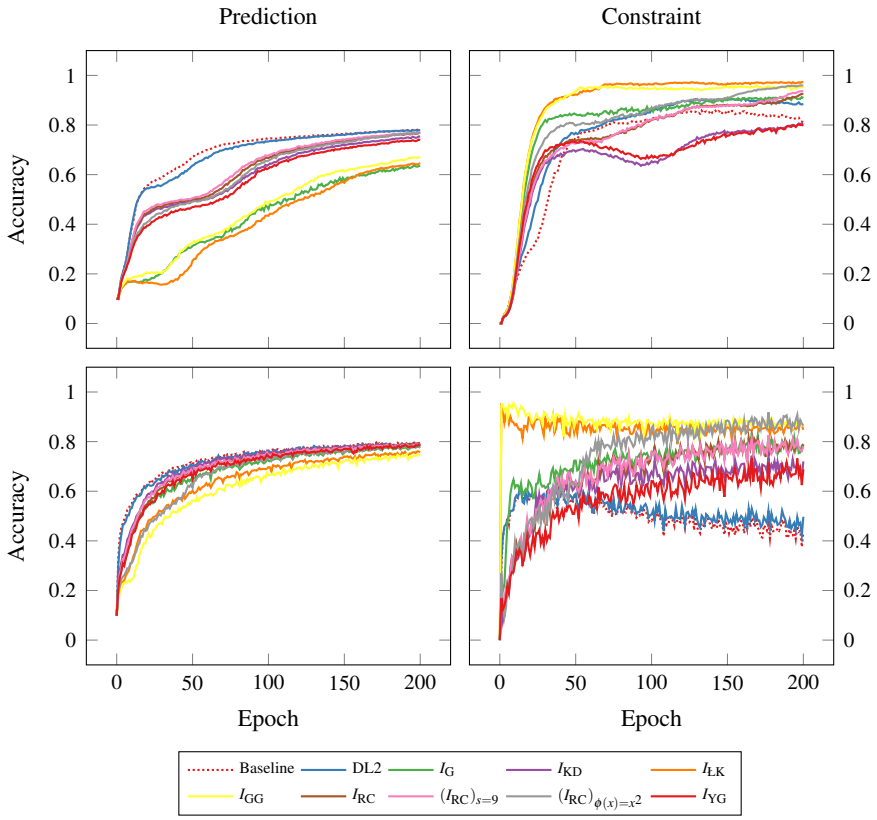

Figure 1: The figure shows how prediction accuracy (left column) and constraint accuracy (right column) change over time when training with the class-similarity constraint for 200 epochs with different logics on Fashion-MNIST (top row) and CIFAR-10 (bottom row).

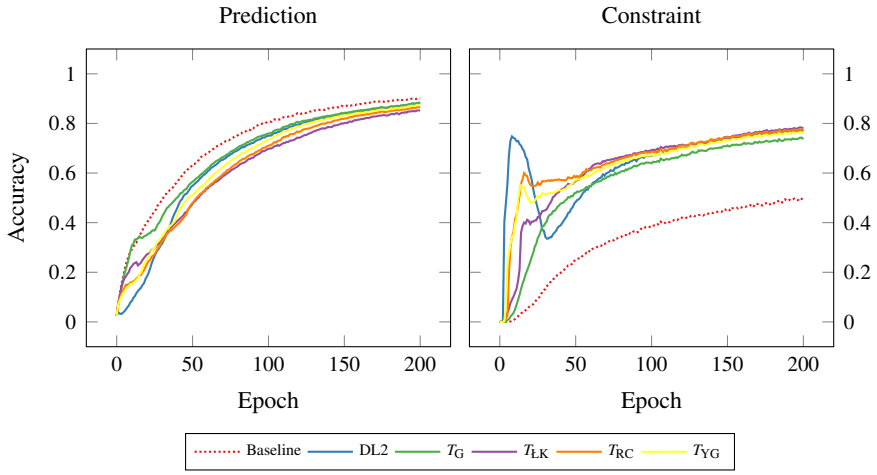

Figure 2: The figure shows how prediction accuracy (left column) and constraint accuracy (right column) change over time when training with the group constraint for 200 epochs with different logics on GTSRB.

A.3 Runtime Overhead

Table 3: Average epoch train times in seconds.

(a) Class-Similarity constraint.

	Fashion-MNIST	**CIFAR-10**
Baseline	0.6 s	3.4 s
DL2	0.6 s	3.4 s
I_{G}	0.6 s	3.8 s
I_{KD}	0.7 s	4.0 s
$I_{\mathrm{\L K}}$	0.7 s	4.0 s
I_{GG}	0.7 s	4.0 s
I_{RC}	0.7 s	4.0 s
$(I_{\mathrm{RC}})_{s=9}$	0.7 s	4.1 s
$(I_{\mathrm{RC}})_{\phi=x^2}$	0.7 s	4.1 s
I_{YG}	0.7 s	4.1 s

(b) Group constraint.

	GTSRB
Baseline	1.1 s
DL2	1.1 s
T_{G}	1.1 s
$T_{\mathrm{\L K}}$	1.1 s
T_{RC}	1.1 s
T_{YG}	1.2 s

The experiments were conducted on a system with a 4.5 GHz AMD Ryzen 9 77950X and a GeForce RTX 4090. The average epoch training times are shown in Table 3. In contrast to the original DL2 experiments [5], we did not observe significant overhead in our experiment. We note that this is due to our reuse of the model output from the cross-entropy loss calculation in the forward pass for our logical loss calculation. If our constraints required a separate forward pass, we would most likely see the same overhead as observed in the DL2 experiments.

A.4 Plots of the Fuzzy Logic Implications

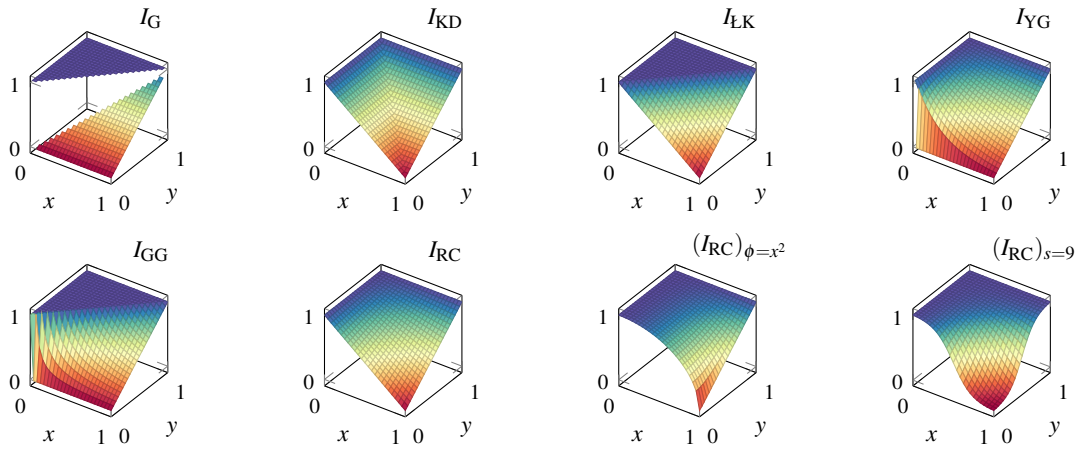

Figure 3: The fuzzy logic implications $I(x,y)$ used in our experiments to map the logical statement $x \to y$ into real-valued loss. Formal definitions are collected in Table 1.

A.5 Plots of Prediction and Constraint Accuracy for Different Values of λ

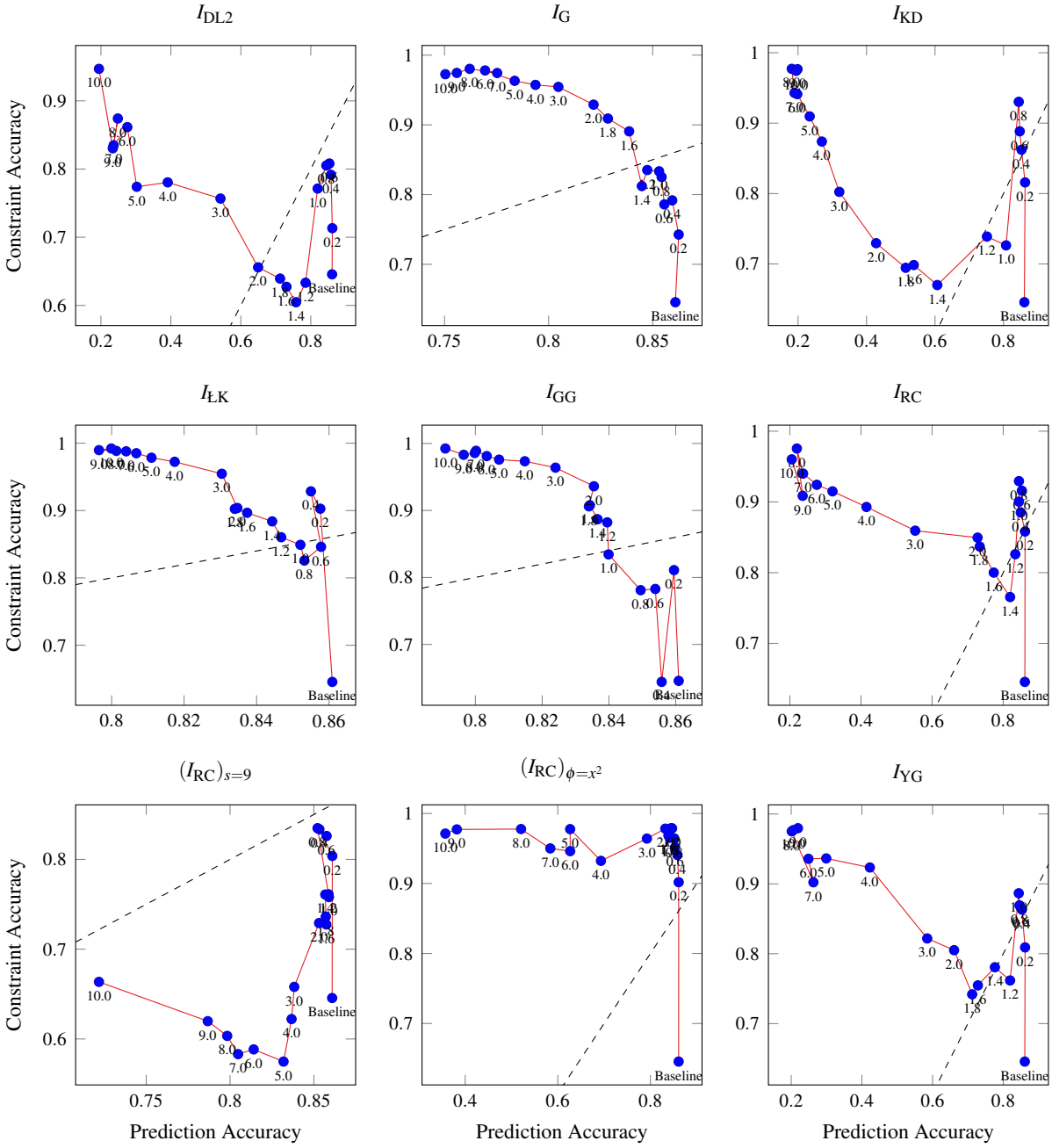

Figure 4: The figure displays prediction and constraint accuracy obtained when training with varying values of λ with the class-similarity constraint on Fashion-MNIST for 200 epochs.

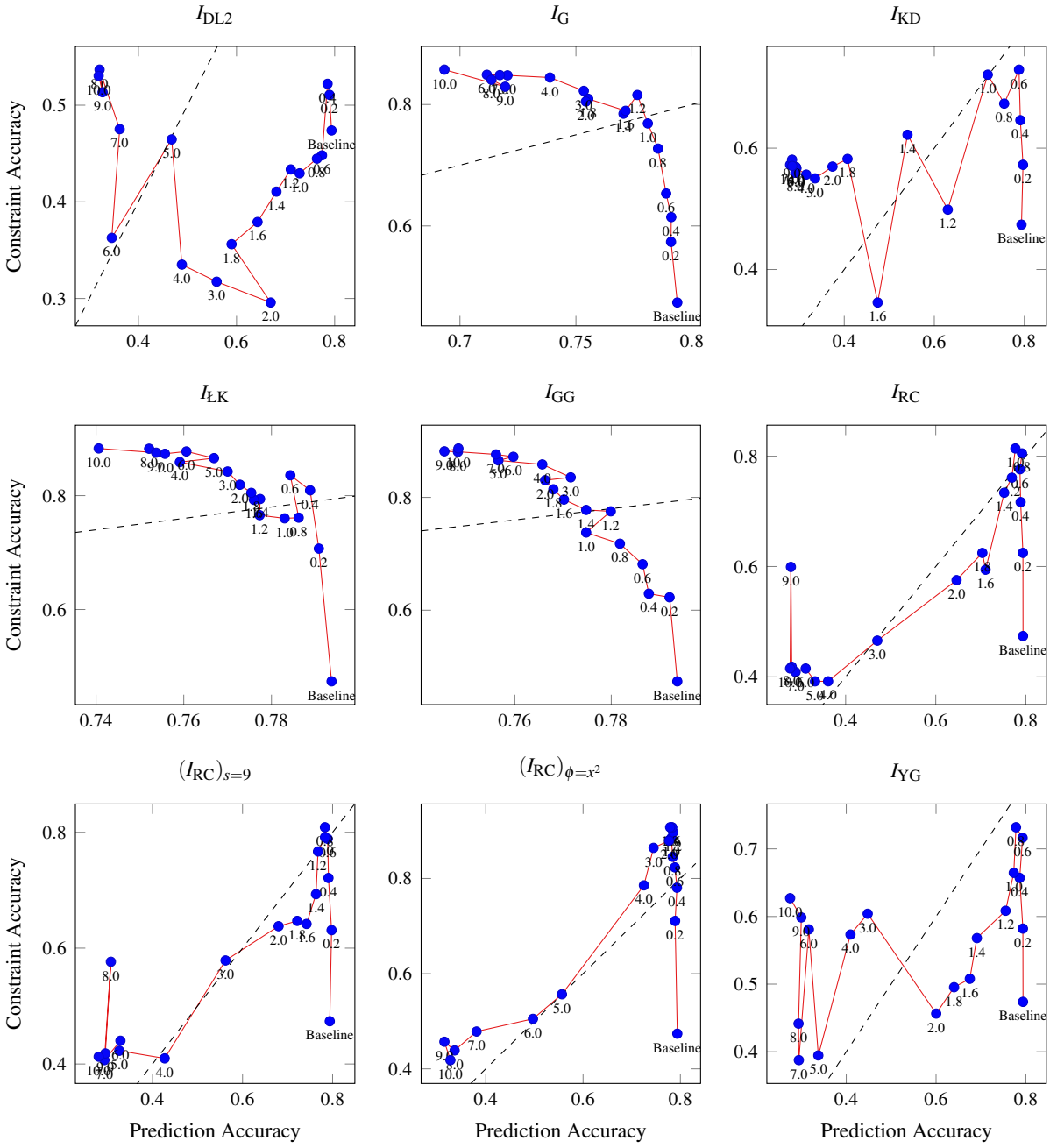

Figure 5: The figure displays prediction and constraint accuracy obtained when training with varying values of λ with the class-similarity constraint on CIFAR-10 for 200 epochs.

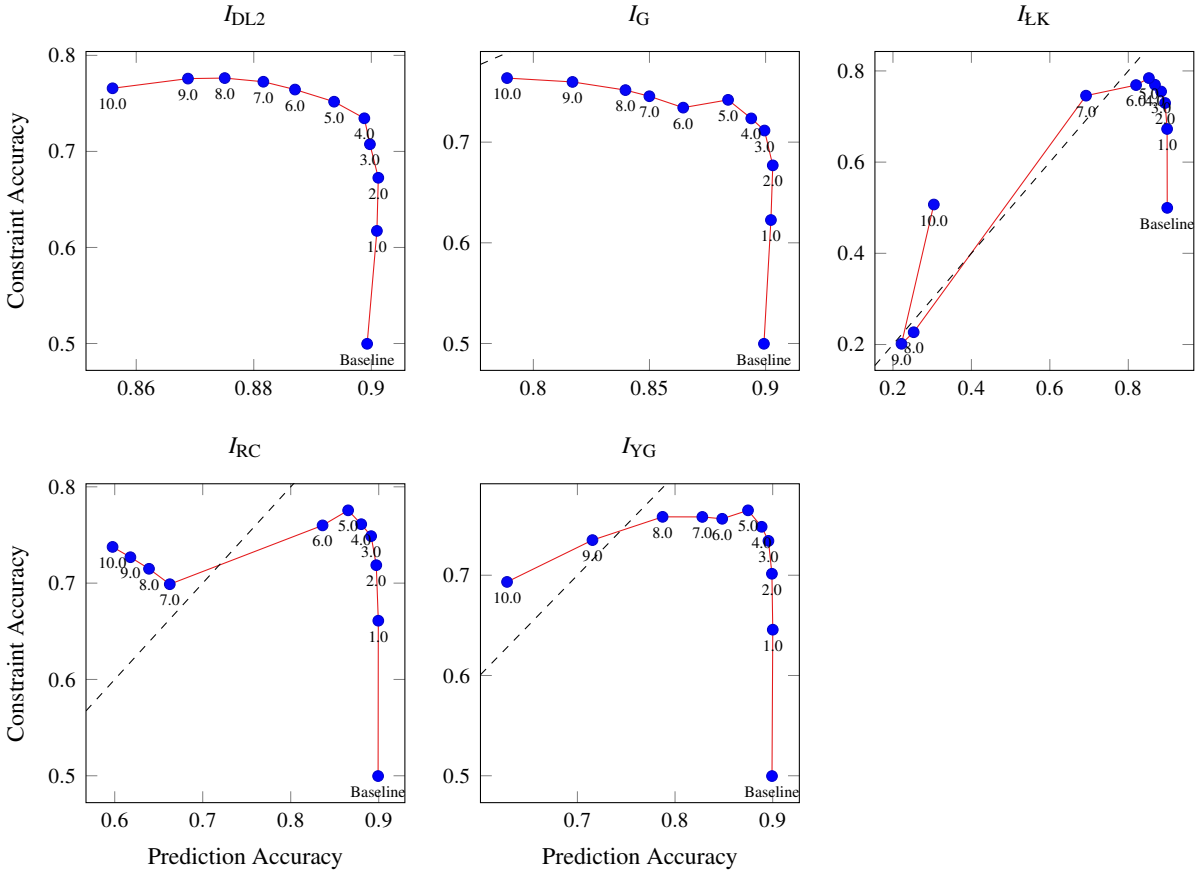

Figure 6: The figure displays prediction and constraint accuracy obtained when training with varying values of λ with the group constraint on GTSRB for 200 epochs.