We thank all the reviewers. We hope the reviewers could increase the rating if the response addressed your concerns.

G#1. General Comments: Convergence of the algorithm Theoretically, in Eq. (11), if we take $\lambda \to \infty$ and estimate $R(f)$ sufficiently accurate, the solution will eventually become monotonic. In practice, we found that we usually find monotonic functions without using a lot of samples when estimating $R(f)$, and we can further manage to do so easier by modifying $R(f) = \mathbb{E}_{x \sim Unif}[\min(b, -\partial f(x))^2]$, where b is a small positive constant (Please see Appendix A2 L402-404). Accordingly, we have revised the relevant sections of the paper by adding pertinent technical details.

G#2. General Comments: All the learned models that we report performance about are certified monotonic.

R1 Regarding Additional Feedback: 1. Arbitrary NN: Our main work is for ReLU networks, extension to more general activation functions are discussed in Appendix B2. **2. Are networks guaranteed monotonicity?** All the networks reported are certified monotonic. We will make a note of this in the paper; see also G#1 and G#2. **3. A layer with only two neurons?** You are correct that, in section 4.1, we use only two neurons for the sake of better illustration because it is a binary classification problem.

R2 On the Implementation of $R(f)$: The exact value of $R(f)$ is intractable, and we approximate it by drawing samples of size 512 uniformly from the input domain during iterations of the gradient descent. Note that the samples we draw vary from iteration to iteration. By the theory of stochastic gradient descent, we can expect to minimize the object function well at convergence. Also, training NNs requires more than thousands of steps, therefore the overall size of samples can well cover the input domain. Our λ is just a tunable real number. See also G#1. We hope you could raise your score if this addresses your concern.

R4 *"We need to stop MILP early. Thus, even if no adversarial example is found, there is still no guarantee that the model is fully monotonic."* MILP problems are NP-hard. In our implementation, we solve the *exact solution* without early stopping for the two-layer MILP problems, and the computation empirically only takes about several seconds when the hidden layer has 100 neurons (Fig. 3). Moreover, most MILP solvers are based on branch and bound, which provides a lower bound of the optimal solution and continuously tightens it. Therefore, it is in fact feasible to stop the algorithm whenever a lower bound is larger than zero, which provides certification but requires no exact solution. See also L149-159 for discussion.

Regarding Numerical Studies: (1) While achieving similar test performance, our method generally use much less parameters than DLN. For example, in Loan Defaulter, we have 72% parameters less than DLN (see Table. 3). (2) To address the potential issue with only 3 runs, we report in the table below the average result in 10 runs (ours) of training loss, average test result and smallest verification result. (3) The difference of the three runs is only the random seed. (4) We provided sensitivity analysis on λ in Fig.6 in Appendix. Please see also the general response for more information.

Datasets	Test Acc (DLN)	Training Loss (DLN)	Test Acc (Ours)	Training Loss (Ours)	U_α
COMPAS	$67.94\% \pm 0.27\%$	0.6124	$68.83\% \pm 0.19\%$	0.5938	0.009
Blog Feedback	0.1608 ± 0.0006	0.1468	0.1584 ± 0.0003	0.1629	0.003
Loan Defaulter	$65.12\% \pm 0.17\%$	0.6244	$65.21\% \pm 0.03\%$	0.6235	0.005
Chest X-Ray	$65.42\% \pm 0.13\%$	0.6232	$66.07\% \pm 0.29\%$	0.6269	0.006

Hyperparameters of our method: Currently we adopt a simple strategy for choosing hyperparameters: (1) we start from $\lambda = 1$, and multiply λ by 10 every time the network fails to pass certification (L226-227). (2) we choose the architecture by grid-search using the validation set (see Appendix L399-404 for choice of grid).

Hyperparameters of Baselines: Isotonic: Default in Scikit-learn package. **XGBoost:** Gird search parameters: maximum tree depth, [3, 10]; learning rate η, [0, 1]; sub-sampling ratio, $(0, 1]$. **Crystal and DLN:** We used hyperparameters for the best reported results in corresponding papers.

Right Panel of Fig. 1: It confirms that even the network looks feature-wise monotonic, we can still find adversarial sample x_{adv} against x in high dimension. Specifically, it shows $y = f(ax + (1-a)x_{adv})$ w.r.t. $a \in [-0.5, 1.5]$, with a as x-axis and y the y-axis. We will make that clear in future version.

R5 Questions in "Weakness" Section: (1) Please refer to G#1 and G#2.

(2) Thanks for pointing that out. It is beyond the scope of this work. We will investigate it in the future.

(3) The results on the three binary classification tasks: **Test Acc (Mono/Non-Mono): 2 vs. 7:** $99.42\% \pm 0.07\%$ / $99.53\% \pm 0.04\%$ **8 vs. 3:** $99.76\% \pm 0.06\%$ / $99.78\% \pm 0.09\%$ **7 vs.1:** $99.85\% \pm 0.08\%$ / $99.82\% \pm 0.04\%$.

(4) (1) A, B, C, D refer to four different networks, with different structures, trained on different datasets. We will include the details in the future version. (2) Let us take a two-layer ReLU network, $f = W_2 ReLU(W_1 x)$, for instance. Because $ReLU(\cdot)$ is a monotonically increasing function, we can simply verify the monotonicity of the network if all the elements in the matrix $W_2 W_1$ is non-negative without our MILP formulation. Each element in the matrix is a multiplication of the weights on a path connecting the input to the output, hence named *non-negative/negative paths.*

(5) We found that directly certifying DNNs with MILP without using two-layer decomposition fails to work completely due to the excessively high computational cost. See Appendix B.3 for more discussion.

(6) Our method can be used to verify the monotonicity on the high-level features in the last few FC layers of CNNs (as what we do in MNIST result); it does not seem to make sense in general to assume mononpremonicity on lower level features such as pixels.