On Quantum Computing for Neural Network Robustness Verification

Nicola Franco [1] **Tom Wollschläger** [2] **Nicholas Gao** [2] **Jeanette Miriam Lorenz** [1] **Stephan Günnemann** [2]

Abstract

In recent years, a multitude of approaches to certify the prediction of neural networks have been proposed. Classically, complete verification techniques struggle with large networks as the combinatorial space grows exponentially, implying that realistic networks are difficult to be verified. For this reason, we propose to leverage the computational power of quantum computing for the robustness verification of neural networks. Further, we introduce a new Hybrid Quantum-Classical Robustness Algorithm for Neural network verification (HQ-CRAN). By applying Benders decomposition we split the verification problem into a quadratic unconstrained binary optimization and a linear program which we solve with quantum and classical computers, respectively. Further, we improve existing hybrid methods based on the Benders decomposition by reducing the overall number of iterations and placing a limit on the maximum number of qubits required. We show that, in a simulated environment, our certificate is sound, and provide bounds on the minimum number of qubits necessary to obtain a reasonable approximation. Finally, we evaluate our method on quantum hardware.

1. Introduction

Despite recent breakthroughs of machine learning models, they have been shown to be brittle to *adversarial examples* – an input with added perturbations specifically crafted by an adversary to fool the network (Szegedy et al., 2014; Goodfellow et al., 2015). In the example of images, the difference between an input and its adversarial example is often undetectable by the human eye while the prediction of the neural network changes drastically.

[1]Fraunhofer Institute for Cognitive Systems IKS, Munich, Germany [2]Dept. of Informatics & Munich Data Science Institute, Technical Univ. of Munich, Germany. Correspondence to: Nicola Franco <nicola.franco@iks.fraunhofer.de>.

1st Workshop on Formal Verification of Machine Learning, Baltimore, Maryland, USA. Colocated with ICML 2022.

Consequently, research on (adversarial) robustness of machine learning models has gained importance to ensure safe usage of machine learning in practice. While empirical defenses exist (Kurakin et al., 2017; Dhillon et al., 2018), these approaches do not provide any guarantee and might be easy to break (Carlini & Wagner, 2017; Athalye et al., 2018; Goodfellow, 2018).

Formal robustness verification guarantees that a network and input pair is robust for a certain adversarial budget, i.e., there will be no adversarial example whose perturbations are smaller than the adversarial budget. However, it has been shown that, even for ReLU networks, the complete verification problem is NP-complete (Katz et al., 2017). Hence, the search space of these approaches grows exponentially with increasing network size resulting in high computational cost.

The development of quantum computing (QC) hardware and software has seen significant progress over the recent years, e.g., recently quantum supremacy has been shown in an academic sample (Arute et al., 2019). However, practical applications of QC remain an active area of research as current quantum computers belong to the category of Noisy Intermediate Scale Quantum (NISQ) devices. Combinatorial optimization has received much attention both in the area of gate-based QC, via the quantum approximate optimization algorithm (QAOA) (Farhi et al., 2014), and in the area of quantum annealing (QA) with the Ising model, respectively (Das & Chakrabarti, 2005).

We propose to leverage the combinatorial power of QC to verify the robustness of ReLU networks by proposing a Hybrid Quantum-Classical Robustness Algorithm for Neural network verification (HQ-CRAN) (Franco et al., 2022). We apply Benders decomposition to partition the MIP into a quadratic unconstrained binary optimization (QUBO) and a linear program (LP) (Benders, 1962; Chang et al., 2020). Thus, the resulting algorithm is an iterative process where the LP generates cuts for the QUBO which is solved by a quantum computer or a quantum annealer. As the current hardware cannot compete with classical solvers, HQ-CRAN should be understood as a proof of concept. However, as the availability of increasingly more powerful computers increases, we may see polynomial speed-ups (Grover, 1996). For convenience of the reader, we summarize some of the

main findings of Franco et al. (2022) in the following sections.

2. Related Work

Robustness Certificates The research field can be divided into two main categories, namely *exact* or *complete* and *incomplete* methods. *Complete* verification methods focus on reasoning about the adversarial polytope, i.e. the shape containing all possible outputs given the set of perturbed inputs. Since these exact verification methods (Bunel et al., 2020; Tjeng et al., 2018; Katz et al., 2017; Ruan et al., 2018) do not scale to large networks due to their exponential time complexity, *incomplete* methods (Gehr et al., 2018; Wong & Kolter, 2018; Müller et al., 2021a; Wang et al., 2021; Xu et al., 2020; Dathathri et al., 2020; Müller et al., 2021b) adopt convex approximations to overcome the non-linearity of the network and reduce the overall complexity. The resulting output of these approaches is the worst-case point of the exact adversarial polytope in case of a *complete* method, and of an outer approximation in case of an *incomplete* one. Due to the outer approximation, the resulting certificates are not sharp, i.e. there might be cases where it fails to verify robustness even though the prediction of the sample could not be changed given the tested budget.

Hybrid Decomposition Algorithms Recently, a multitude of quantum optimization algorithms have been proposed for NISQs. The focus is to solve large combinatorial optimization problems otherwise intractable by classical solvers. In this context, Gambella & Simonetto (2020) proposed a decomposition method based on the alternating direction method of multipliers (ADMM). The method is a heuristic algorithm which splits a mixed-binary optimization problem into a QUBO, solved with QC, and a LP via a multi-block version of ADMM. Similarly, Chang et al. (2020) and Zhao et al. (2021) used Benders decomposition to divide a MIP into a QUBO and a LP. To formulate QUBO, both methods require the real variables to be approximated to binaries, and then the translation to qubits is one-to-one. In addition to this, at each iteration a new real variable is constantly added, leading to an always increasing number of qubits. In Franco et al. (2022), we improved previous methods by reducing the overall number of iterations and by placing a limit to the maximum number of qubits required.

3. Background and Preliminaries

Notation We use lower case Latin and Greek letters $a, b, \ldots, \alpha, \beta, \ldots$ for scalars, bold $\mathbf{a}$ for vectors, capitalized bold $\boldsymbol{A}$ for matrices, and calligraphic $\mathcal{A}$ for sets. Furthermore, we use $\mathbf{a}_i$ to denote the i-th element of $\mathbf{a}$ and $|\cdot|$ to denote the cardinality of a set. $\texttt{diag}(\mathbf{v})$ denotes a diagonal matrix with all vector entries on the main diagonal and zero elsewhere. We denote with $\odot$ the element-wise product and with $\otimes$ the outer product between two vectors. The identity matrix in $\mathbb{R}^{n\times n}$ is $\mathbf{I}_n$, while $\mathbf{O}_{n\times m}$ is the zero matrix in $\mathbb{R}^{n\times m}$. The column vector of n ones is denoted by $\mathbb{1}_n$, and $\mathbf{0}_n$ is the zero column vector in $\mathbb{R}^n$.

Model Formulation We define a neural network by a function $\mathbf{f}(\mathbf{x})\colon \mathcal{X} \to \mathbb{R}^{|\mathcal{Z}|}$ which maps input samples $\mathbf{x} \in \mathcal{X}$ to output $\mathbf{z} \in \mathbb{R}^{|\mathcal{Z}|}$, where $\mathcal{Z}$ is the set of classes. We assume a feedforward architecture composed by affine transformations and followed by ReLU activation functions:

$$\begin{aligned}\hat{\mathbf{z}}^{[i]} &= \mathbf{W}^{[i]}\mathbf{z}^{[i-1]} + \mathbf{v}^{[i]},\\ \mathbf{z}^{[i]} &= \max\{0, \hat{\mathbf{z}}^{[i]}\}, \quad \forall i \in \{1, \ldots, L\},\end{aligned}$$

where L represents the number of layers, $\mathbf{z}^{[0]} \equiv \mathbf{x}$ and $\mathbf{f}(\mathbf{x}) \equiv \mathbf{z}^{[L]}$. In case of classification, the network outputs a vector in $\mathbb{R}^{|\mathcal{Z}|}$. The predicted class is then given by the index of the largest value of that vector, i.e. $c = \arg\max_j \mathbf{f}(\mathbf{x})_j$.

Robustness Certificate and Threat Model Let $\mathbf{x}$ be an input, e.g. a vectorized image. We say that a neural network f is certifiably robust for this input if the prediction for all perturbed versions remains unchanged:

$$\arg\max_j \mathbf{f}(\mathbf{x})_j = \arg\max_j \mathbf{f}(\tilde{\mathbf{x}})_j \quad \forall \tilde{\mathbf{x}} \in \mathcal{B}_\epsilon^p(\mathbf{x}). \tag{1}$$

Here, ϵ is the perturbation budget and $\tilde{\mathbf{x}}$ is an element from the perturbation set $\mathcal{B}_\epsilon^\infty(\mathbf{x})$ based on the infinity norm:

$$\mathcal{B}_\epsilon^\infty(\mathbf{x}) = \{\tilde{\mathbf{x}} \,|\, \|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty \leq \epsilon\}. \tag{2}$$

If we cannot certify an input, it means that there exists $\mathbf{x}' \in \mathcal{B}_\epsilon^\infty(\mathbf{x})$ for which $\arg\max_j \mathbf{f}(\mathbf{x})_j \neq \arg\max_j \mathbf{f}(\mathbf{x}')_j$. We call any of these $\mathbf{x}'$ an adversarial example.

The piece-wise linear nature of ReLU activation units characterizes the problem as non-convex. There are two ways to solve this problem. Either we model the ReLU activation with an integer variable or we enclose the possible activation values $\mathbf{z}^{[i]}$ with a convex area. The former approach yields *complete* formulation of the exact polytope but the integer variables render the problem NP-hard (Tjeng et al., 2018). The latter approach is an instance of an *incomplete* solution (Wong & Kolter, 2018).

Quadratic Unconstrained Binary Optimization (QUBO) Many problems in finance, economics and machine learning may be formulated as QUBO problems (G. Kochenberger et al., 2014):

$$\min_{\mathbf{x}\in\{0,1\}^n} \mathbf{x}^\intercal \boldsymbol{Q}\mathbf{x} + \mathbf{q}\mathbf{x}, \tag{3}$$

where $\mathbf{x}$ is vector of binary variables, $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is a square matrix and $\mathbf{q} \in \mathbb{R}^n$ is a row vector.

In QC, the QUBO formulation gained lots of attention due to its close resemblance to the Ising model (Lucas, 2014),

a physical model directly linked to spin states, with main difference being that states take values in $\{-1, 1\}$ in Ising and $\{0, 1\}$ in QUBO. So, to get from an Ising model to the corresponding QUBO, one has to transform the states by $s = 2x - 1$ and adjust $\mathbf{h}$, and $\boldsymbol{J}$ accordingly:

$$\min_{\mathbf{s}\in\{-1,1\}^n} -\sum_i \mathbf{h}_i \mathbf{s}_i - \sum_{ij} \boldsymbol{J}_{ij} \mathbf{s}_i \mathbf{s}_j. \tag{4}$$

Quantum Annealing (QA) aims at directly solving such Ising problems on quantum hardware (Johnson et al., 2011) with the only restriction being the connectivity of the qubits on QA. While in simulation and theory we may connect any qubit to any other qubit, this is not possible on current hardware. To circumvent this issue, one may represent one logical unit s_i by multiple qubits with large coupling weights $\boldsymbol{J}_{ij}$ in between (Adachi & Henderson, 2015). While proven to be a suboptimal choice (Marshall et al., 2020) it remains an open research question to find suitable strategies of overcoming these limits of current hardware.

Thus, we can obtain solutions for the QUBO problem by finding the ground state of the Ising model. In QA, we achieve this by starting from a known initial state which we slowly evolve towards the ground state of the problem. For a detailed description we refer the reader to (Das & Chakrabarti, 2005).

Quantum Approximate Optimization Algorithm (QAOA) (Farhi et al., 2014) is a hybrid quantum method for solving Ising problems. The algorithm is considered as an excellent candidate for NISQ devices due to its hybrid structure with an iteration between conventional computers and QC. Its potential in achieving a practical quantum advantage, however, remains unclear and is intensively studied (Farhi & Harrow, 2019; Guerreschi & Matsuura, 2019). In comparison to QA, the use of the QAOA on gate-based QC may open up the possibility to consider more complicated Hamiltonians, also going beyond the QUBO problem. The QAOA transforms the problem Hamiltonian into a sequence of local Hamiltonians and evaluates approximate ground states to obtain an approximate solution of the original optimization problem.

4. HQ-CRAN

In this section, we provide an introduction to HQ-CRAN (Franco et al., 2022). Given the exponential complexity of a complete verification method, we highlight the potential of Benders decomposition to exploit NISQ devices and run part of the problem on QC while generating constraints on a classical one. Benders decomposition is an algorithm for mixed-variables programming that has been successfully applied to a variety of applications (Benders, 1962). Recently, Chang et al. (2020) and Zhao et al. (2021) proposed its use on QC. Following their approach, Franco et al. (2022) present the first application to the verification problem of neural networks and supplement it with QC-specific improvements. In a simulation with a sufficient number of qubits, HQ-CRAN theoretically converges to the value of the exact solution (Benders, 1962).

A problem with binary variables $\mathbf{y}$ is considered, making Equation 1 to be an instance of the class of MIP problems. The main components of HQ-CRAN are two programs, one LP and one QUBO. These are alternatively solved by classical and quantum computing, correspondingly, until convergence or another stopping criterion is met. When the objectives of both programs meet, the optimization stops and issues a certificate if this value is positive.

Due to the hardware constraints, approximations within the QUBO program leading to a non exact certificate but rather a conservative bound have been used. This means that the certificate is valid and in the ideal, simulated scenario, HQ-CRAN converges to the exact solution with enough iterations. However, in settings where the number of iterations or qubits is limited, HQ-CRAN is an incomplete verification method rendering it as a mixture of complete and incomplete verification.

Overview For a proper certificate, the network's prediction does not have to change into any other possible class.However, for simpler notation, we now only consider the inner minimization problem, i.e. only testing the difference between the initial predicted class and one other. To avoid cluttered notation, we write the complete verification problem in a concise matrix formulation as:

$$\min_{\mathbf{z},\mathbf{y}} \left\{ \mathbf{g}^\intercal \mathbf{z} \,|\, \boldsymbol{A}\mathbf{z} + \boldsymbol{B}\mathbf{y} \geq \mathbf{b},\ \boldsymbol{C}\mathbf{z} \geq \mathbf{d} \right\}. \tag{5}$$

The matrices and vectors $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\mathbf{b}$, and $\mathbf{d}$ model the whole network and the exact construction procedure is explained in Algorithm 1 of Franco et al. (2022), where to propagate the boundaries through the network, either IBP (Tjeng et al., 2018) or CROWN (Zhang et al., 2018) can been used. The vector $\mathbf{z} \in \mathbb{R}^{n_z}$ considers all network logits with $n_z = n_0 + \cdots + n_L$ equal to the total number of neurons, and $\mathbf{y} \in \{0, 1\}^{n_y}$ with n_y equal to the number of unstable neurons (i.e. a neuron is $unstable$ when its boundaries are $\boldsymbol{\ell}_j^{[i]} < 0 < \mathbf{u}_j^{[i]}$; otherwise we can express it with a constant being either 0 or 1). The vector $\mathbf{g}$ encodes the difference between the true class and another to be tested against, e.g. $\mathbf{g} = (\ldots, 0, 1, 0, -1, 0)^\intercal$.

With Benders Decomposition, explained in Appendix A, Equation 5 is decomposed into two sub-problems: A binary master problem and a linear programming optimization sub problem. Similarly to Benders (1962), we rewrite the opti-

mization of Equation 5 to obtain:

$$\min_{\mathbf{y},\,\eta} \ \eta, \tag{6a}$$

$$\text{s.t. } \boldsymbol{\alpha}^{[k]}\left(\mathbf{b}-\boldsymbol{B}\mathbf{y}\right)+\boldsymbol{\beta}^{[k]}\mathbf{d}\leq\eta,\ \forall(\boldsymbol{\alpha}^{[k]},\boldsymbol{\beta}^{[k]})\in\Lambda_p, \tag{6b}$$

$$\boldsymbol{\alpha}^{[k]}\left(\mathbf{b}-\boldsymbol{B}\mathbf{y}\right)+\boldsymbol{\beta}^{[k]}\mathbf{d}\leq 0,\ \forall(\boldsymbol{\alpha}^{[k]},\boldsymbol{\beta}^{[k]})\in\Lambda_r, \tag{6c}$$

where $\eta \in \mathbb{R}$ is a scalar. This problem is known as *master* problem on Benders decomposition.

Theorem 4.1 (Franco et al. (2022))**.** *Given a neural network* $\mathbf{f}$*, an input* $\mathbf{x}$ *and logits* $\mathbf{z}$ *and following Algorithm 1 of (Franco et al., 2022) to create constraint matrices* $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ *as well as vectors* $\mathbf{b}, \mathbf{d}, \mathbf{g}$*, the verification of robustness can be evaluated through the optimization of Equation 6.*

The proof of Theorem 4.1 uses the decomposition theorem of Benders (1962) and is given in Franco et al. (2022).

The difficulty of solving Equation 6 is the exponential size of the sets Λ_p, Λ_r (Chang et al., 2020). Thus, we can gradually extend the sets $\Lambda'_p \subseteq \Lambda_p, \Lambda'_r \subseteq \Lambda_r$ by constraints of the *sub* problem defined as:

$$\max_{\boldsymbol{\alpha}\leq\bar{\boldsymbol{\alpha}},\,\boldsymbol{\beta}\leq\bar{\boldsymbol{\beta}}}\left\{\boldsymbol{\alpha}\left(\mathbf{b}-\boldsymbol{B}\mathbf{y}\right)+\boldsymbol{\beta}\mathbf{d}\mid\boldsymbol{\alpha}\boldsymbol{A}+\boldsymbol{\beta}\boldsymbol{C}=\mathbf{g}^{\intercal}\right\}. \tag{7}$$

The sub problem is similar to Equation A.9 except that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are bounded. This bounding idea, introduced by Chang et al. (2020), can be used to identify whether we add a cut belonging to the extreme points or rays. If any of the optimal values of the vectors $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$ is equal to the upper bounds, the constraint belongs to the extreme rays Λ'_r. Otherwise, it belongs to the extreme points Λ'_p. As the set of extreme points is bounded, this decision rule is correct as long as the bounds $\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\beta}}$ are sufficiently large, i.e. larger than any solution of the extreme points set (Chang et al., 2020).

Overall procedure Here, we describe in more details the main functions of Algorithm 1. BUILD$(\mathbf{x}, \epsilon, \boldsymbol{W}, \mathbf{v})$ constructs the matrices for the optimization problem and is explained in Algorithm 1 of Franco et al. (2022). Note that $\mathbf{x}$ is the input, ϵ is the adversarial perturbation budget, $\boldsymbol{W}$ and $\mathbf{v}$ are the network weights and biases, respectively. SOLVESUBPROBLEM$(\mathbf{y}^{[t]})$ computes the solution of Equation 7 and the result is compared to the one at the previous iteration. Hence, the minimum between the two is considered as the new sub objective $s^{[t]}$. COMPUTECOREPOINT$(\mathbf{y}^{[t]})$ updates $\bar{\mathbf{y}}^{[t]}$ according to the following sum: $\frac{1}{2}\bar{\mathbf{y}}^{[t-1]} + \frac{1}{2}\mathbf{y}^{[t]}$. The function SOLVEMAGNANTI&WONG$(\bar{\mathbf{y}}^{[t]})$ solves the additional problem of Franco et al. (2022) with the newly computed core point $\bar{\mathbf{y}}^{[t]}$ and the resulting solutions $\boldsymbol{\alpha}^{[t]}, \boldsymbol{\beta}^{[t]}$ are added to Λ'_p. To preserve the size of Λ'_p, if the number of cuts is larger than the maximum value φ, then the oldest cut $\boldsymbol{\alpha}^{[t-\varphi]}, \boldsymbol{\beta}^{[t-\varphi]}$ is removed from Λ'_p. T is the upper bound on the number of iterations, ξ is the gap between the master and sub solution and φ is the maximum number of cuts. In the worst case, if the gap ξ is not met, the algorithm will run for a maximum number of T iterations. The reader is referred to Franco et al. (2022) for a more detailed discussion.

Algorithm 1 HQ-CRAN (Franco et al., 2022)

input: $\mathbf{x}, \epsilon, \boldsymbol{W}, \mathbf{v}, T, \xi$
output: robust, not robust or unknown
initialize: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \mathbf{d}, \mathbf{b}, \mathbf{g} \leftarrow$ BUILD$(\mathbf{x}, \epsilon, \boldsymbol{W}, \mathbf{v})$.
for each adversarial class **do**
  $n_p \leftarrow$ NUMBEROFQUBITSFORP()
  for t **in** $\{1, \ldots, T\}$ **do**
    $s^{[t]} \leftarrow$ SOLVESUBPROBLEM$(\mathbf{y}^{[t]})$
    $\bar{\mathbf{y}}^{[t]} \leftarrow$ COMPUTECOREPOINT$(\mathbf{y}^{[t]})$
    $\Lambda'_p \leftarrow$ SOLVEMAGNANTI&WONG$(\bar{\mathbf{y}}^{[t]})$
    $n_{a_t} \leftarrow$ NUMBEROFQUBITSFORA(t)
    $m^{[t]}, \mathbf{y}^{[t]} \leftarrow$ QUANTUMOPTIMIZATION(Λ'_p)
    if $s^{[t]} - m^{[t]} \leq \xi$ **and** VERIFY$(\mathbf{y}^{[t]})$ **then**
      break
    end if
  end for
  if $m^{[t]} < 0$ **then**
    return unknown
  else if $s^{[t]} < 0$ **then**
    return not robust
  end if
end for
return robust

5. Results

Architecture, Dataset and Training Methods. We conduct experiments with one type of multilayer perceptron (MLP) neural network: MLP-2x[20]. Here MLP-mx[n] refers to m hidden layers and n units per hidden layer. The network uses ReLU functions after every fully connected layer. We train our model on the MNIST dataset for 20 epochs with a batch size of 128 in two ways: (i) with a regular loss function and (ii) adversarially trained via the Projected Gradient Descent (PGD) as in Madry et al. (2017). In case of a regularly trained model, we keep the name MLP-2x[20], while we refer to it as PGD-2x[20] if it is adversarially trained. The clean accuracy for the different networks are $95,62\%$ and $86,73\%$ for MLP-2x[20] and PGD-2x[20], respectively. The adversarial training used adversarial examples from the infinity norm ball around the input with radius $\epsilon = 0.01$.

Experimental Setup. We divide the hardware into classical and quantum computing. On the classical side, we ran the algorithm on a server having 4xCPUs Intel(R) Xeon(R)

E7-8867 v4 running at 2.40GHz for a total of 72/144 cores/threads and the GPU verifiers on a Nvidia RTX3090. On the quantum side, we used two different system types: quantum annealing and gate-based quantum computers. In the first case, we access the D-Wave Advantage™ system 5.1 constructed with 5760 qubits in two ways: (i) directly on the Quantum Processing Unit (QPU) and (ii) with the Hybrid solver provided by D-Wave Leap[1], which makes use of a Tabu search to further decompose the problem and run a part on the QPU. In the second case, we used the QAOA[2] runtime program of the IBMQ[3] cloud on the IBM Brooklyn QC having the Hummingbird r2 architecture of 65 qubits, a quantum volume of 32, and 1.5K circuit layers operations per second.

5.1. Evaluation of Robustness

In this section we compare HQ-CRAN (Franco et al., 2022) against the complete verifier β-CROWN (Wang et al., 2021) and two incomplete verifiers PRIMA (Müller et al., 2021b) and GPUPoly (Müller et al., 2020). We evaluate the empirical performance of HQ-CRAN in an ideal setting, i.e., where the *master* and *sub* problems are solved using GUROBI[4](Gurobi Optimization, LLC, 2022) software on a classical computer. For a fair comparison, we propagate the boundaries through the network with CROWN (Zhang et al., 2018), also used by β-CROWN, and evaluate all methods on the first 100 samples from the MNIST test set with $\epsilon \in \{\frac{1}{255}, \frac{2}{255}, \frac{4}{255}, \frac{8}{255}, \frac{16}{255}\}$ as adversarial budget. We implement HQ-CRAN without the QUBO formulation for QC, i.e., without relaxing the constraints with additional variables and adding them to the objective by quadratically penalizing them.

In terms of final robustness certification, we report in Table 1 the certified accuracy, i.e., the fraction of verified and correctly classified samples of all test samples. To limit the compute time, we set the maximum number of iterations T to 1000 and the target gap ξ between the *master* and *sub* problem to 0.1. Within these settings, HQ-CRAN behaves similarly to β-CROWN in terms of certified accuracy, and shows better results for ϵ values greater than $^8/_{255}$ with respect to PRIMA and GPUPoly. Compared to the complete approach and HQ-CRAN, the certified accuracy for incomplete methods drops significantly for $\epsilon = {}^{16}/_{255}$, indicating that outer approximations struggle to verify larger perturbations.

On the contrary, our average runtime is ~ 2 orders of magnitude slower than incomplete methods and ~ 1 order than β-CROWN, as shown in Figure 1. As these results are evaluated on a classical computer and current NISQ devices are not yet able to handle larger problems, it is apparent that we do not gain any speed-up and our work shall be considered as a proof-of-concept.

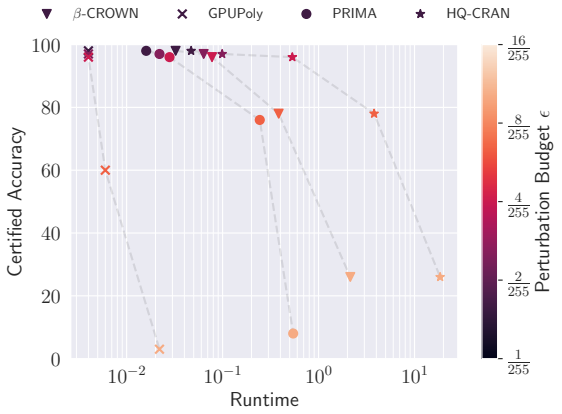

Figure 1: Certified accuracy and average runtime of MLP2x[20] for β-CROWN, GPUPoly, PRIMA and HQ-CRAN on the first 100 samples from the MNIST test set with an increasing perturbation budjet ϵ. In the context of HQ-CRAN the bounds for the sub problem and the maximum number of iterations T have been set to 10 and 1000, respectively. The gap η has been set to 0.1.

5.2. Simulations in Classical Computing

To provide more insights on HQ-CRAN, we further conduct evaluations on the objective and runtime per iteration of the master and sub problems, as shown in Figure 2. Both problems are solved using GUROBI (Gurobi Optimization, LLC, 2022). In this scenario, we test the robustness of a sample against three adversarial classes, named from 0 to 2. The sub objective starts from a large positive value and decreases with each iteration. Meanwhile the master objective starts from $-\infty$ and increases until it reaches zero or becomes positive. In the latter case, the algorithm goes on to test the next class. In contrast, if the sub objective becomes negative the algorithm stops and declares the sample not robust.

The average time of the master solver increases with each iteration, while the average time of the sub solver remains constant. This behavior must be attributed to the continuous growth of the constraints of the master problem generated by the sub. Since the execution of the master affects the overall time, solving it with quantum optimization may drastically reduce the execution time in the future.

In the end, one may obtain an polynomial speed-up (Grover, 1996), though, still in exponential time as it is assumed that QC may not solve NP-hard problems with high probability in P time (Mohr, 2014). An analogy of this can be observed in Figure 2, as in the worst case, one has to add all possible cuts from the exponentially large set of extreme rays. Luckily, in practice, few iterations already lead to sufficient

[1]D-Wave Leap™ Quantum Cloud Service https://cloud.dwavesys.com/leap/

[2]Python library Qiskit v0.36.0 https://github.com/Qiskit/qiskit

[3]IBM Quantum https://quantum-computing.ibm.com/

[4]GUROBI python version 9.1.2

Table 1: Adversarial robustness of MNIST classifiers to perturbations of ϵ in a l_∞-norm. We run each algorithm on the first 100 test set samples from the MNIST dataset. The times are expressed in seconds.

NETWORK	ϵ	CERTIFIED ACCURACY ↑				AVERAGE TIME [S]↓			
		β-CROWN	HQ-CRAN	PRIMA	GPUPOLY	β-CROWN	HQ-CRAN	PRIMA	GPUPOLY
PGD-2x[20]	$2/255$	88%	88%	88%	88%	0.017	0.029	0.073	0.002
	$4/255$	88%	88%	88%	88%	0.017	0.157	0.075	0.005
	$8/255$	81%	81%	81%	81%	0.181	1.474	0.115	0.008
	$16/255$	52%	52%	32%	27%	1.820	6.331	0.369	0.013
MLP-2x[20]	$2/255$	97%	97%	97%	97%	0.078	0.099	0.022	0.004
	$4/255$	96%	96%	96%	96%	0.063	0.533	0.028	0.004
	$8/255$	78%	78%	76%	60%	0.384	3.791	0.244	0.006
	$16/255$	26%	26%	8%	3%	2.140	18.487	0.544	0.022

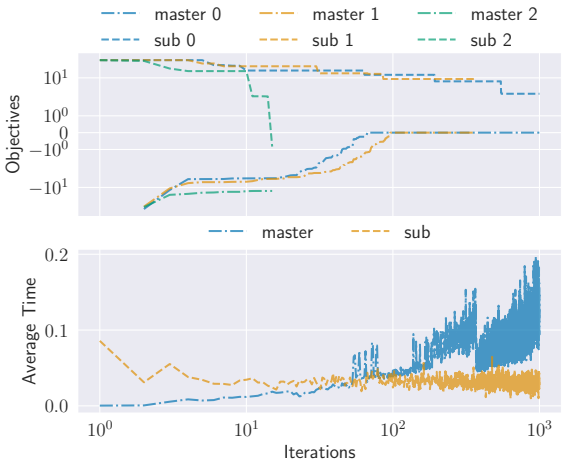

Figure 2: Objectives and average time per iteration for the master and sub solvers on one sample of MNIST test set for MLP2x[20] with an ϵ of $\frac{16}{255}$. The robustness is conducted against three adversarial classes, named from 0 to 2.

results.

5.3. Quantum Hardware Experiments

Finally, we evaluate HQ-CRAN running on a gate-based quantum computer via QAOA and on a quantum annealer via Leap. Due to time limits on current quantum hardware, we present results on the first and the first ten MNIST test set samples for QAOA and Leap, respectively. This also restricts our hyperpameter choices for QAOA to the Perturbation Stochastic Approximation optimizer with a maximum iteration number of 10 instead of 100, 2 repetitions and 1024 shots. For Leap we use the default 3-minute execution time limit. The penalty weights w_a and w_p were set to 0.1 and 0.01, respectively. $\epsilon = \frac{1}{255}$ and a target gap ξ of 1 for both algorithms. With this setting, QAOA starts of with ~ 28 qubits and requires 13 additional qubits per step. For Leap, the number of theoretical qubits is identical but one needs to add qubits required for increased connectivity as discussed previously.

Figure 3 shows the convergence behavior of Leap and QAOA. The gap is defined as difference between the objectives of *master* and *sub* problems at each iteration. The solution is reached when the gap is lower than our target gap $\xi = 1$. While we find QAOA not converging by the constrained accessibility and limitations of current QC hardware, Leap converges close to the optimal solution within 6 steps for all tested samples.

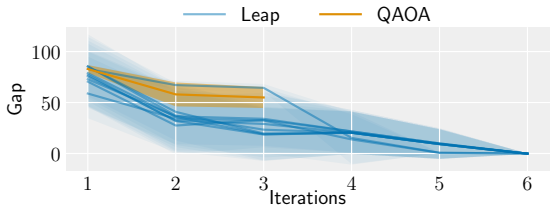

Figure 3: Average and standard deviation of the difference between *master* and *sub* objective at each iteration for QAOA and Leap. In the case of Leap, we plot the first 10 samples of the MNIST test set. In contrast, for QAOA, we only show the first one. We used an ϵ value of $1/255$ and PGD-2x[20].

6. Conclusion

We proposed a proof of concept for a hybrid quantum algorithm for robustness verification leveraging the computational power of NISQ devices. We have shown that by applying Benders decomposition to the MIP related to ReLU network certification one obtains a classically solved LP and a QC suitable sub problem. Further, we perform evaluations of HQ-CRAN in comparison with state-of-the-art GPU-based verifiers. On the one hand, results show that HQ-CRAN performs similarly to complete verifiers and outperforms incomplete ones in terms of certified accuracy. On the other hand, HQ-CRAN is two orders of magnitude slower than incomplete methods on classical hardware. Finally, we evaluate the method on quantum hardware and show that the runtime can benefit from speedup when more powerful quantum hardware will become available. In conclusion, this work paves the way towards exploiting the combinatorial power of QC to accelerate robustness verification of neural networks in the future.

Acknowledgment

The project/research is supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy with funds from the Hightech Agenda Bayern.

References

Adachi, S. H. and Henderson, M. P. Application of quantum annealing to training of deep neural networks. *arXiv preprint arXiv:1510.06356*, 2015.

Arute, F. et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. doi:10.1038/s41586-019-1666-5.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 274–283. PMLR, 2018. URL https://proceedings.mlr.press/v80/athalye18a.html.

Benders, J. F. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962. doi:10.1007/BF01386316. URL https://doi.org/10.1007/BF01386316.

Bunel, R., Mudigonda, P., Turkaslan, I., Torr, P., Lu, J., and Kohli, P. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21, 2020.

Carlini, N. and Wagner, D. *Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods*, pp. 3–14. Association for Computing Machinery, New York, NY, USA, 2017. ISBN 9781450352024. URL https://doi.org/10.1145/3128572.3140444.

Chang, C.-Y., Jones, E., and Graf, P. On quantum computing for mixed-integer programming. *arXiv preprint arXiv:2010.07852*, 2020.

Das, A. and Chakrabarti, B. K. *Quantum Annealing and Related Optimization Methods*, volume 679. Springer Science & Business Media, 2005.

Dathathri, S., Dvijotham, K., Kurakin, A., Raghunathan, A., Uesato, J., Bunel, R. R., Shankar, S., Steinhardt, J., Goodfellow, I., Liang, P. S., and Kohli, P. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5318–5331. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/397d6b4c83c91021fe928a8c4220386b-Paper.pdf.

Dhillon, G. S., Azizzadenesheli, K., Lipton, Z. C., Bernstein, J., Kossaifi, J., Khanna, A., and Anandkumar, A. Stochastic Activation Pruning for Robust Adversarial Defense. *International Conference on Learning Representations*, pp. 1–13, 2018.

Farhi, E. and Harrow, A. W. Quantum supremacy through the quantum approximate optimization algorithm. *quant-ph*, 2019.

Farhi, E., Goldstone, J., and Gutmann, S. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028 [quant-ph]*, November 2014.

Franco, N., Wollschlaeger, T., Gao, N., Lorenz, J. M., and Guennemann, S. Quantum robustness verification: A hybrid quantum-classical neural network certification algorithm. *arXiv preprint arXiv:2205.00900*, 2022.

G. Kochenberger, G., Hao, J., Glover, F., et al. The unconstrained binary quadratic programming problem: a survey. *J Comb Optim*, 28:58–8, 2014.

Gambella, C. and Simonetto, A. Multiblock admm heuristics for mixed-binary optimization on classical and quantum computers. *IEEE Transactions on Quantum Engineering*, 1:1–22, 2020.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, 2018. doi:10.1109/SP.2018.00058.

Goodfellow, I. Gradient masking causes clever to overestimate adversarial perturbation size, 2018.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 212–219, 1996.

Guerreschi, G. G. and Matsuura, A. Y. Qaoa for max-cut requires hundreds of qubits for quantum speed-up. *Scientific Reports*, 9(1), 2019. ISSN 2045-2322. doi:10.1038/s41598-019-43176-9. URL http://dx.doi.org/10.1038/s41598-019-43176-9.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL https://www.gurobi.com.

Johnson, M., Amin, M., Gildert, S., Lanting, T., Hamze, F., Dickson, N., Harris, R., Berkley, A., Johansson, J., Bunyk, P., Chapple, E., Enderud, C., Hilton, J., Karimi, K., Ladizinsky, E., Ladizinsky, N., Oh, T., Perminov, I., Rich, C., and Rose, G. Quantum annealing with manufactured spins. *Nature*, 473:194–8, May 2011. doi:10.1038/nature10012.

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*, pp. 97–117. Springer, 2017.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale, 2017.

Lucas, A. Ising formulations of many np problems. *Frontiers in physics*, 2:5, 2014.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Marshall, J., Di Gioacchino, A., and Rieffel, E. G. Perils of embedding for sampling problems. *Physical Review Research*, 2(2):023020, April 2020. doi:10.1103/PhysRevResearch.2.023020.

Mohr, A. Quantum computing in complexity theory and theory of computation. *Carbondale, IL*, 1, 2014.

Müller, C., Serre, F., Singh, G., Püschel, M., and Vechev, M. Scaling polyhedral neural network verification on gpus. In Smola, A., Dimakis, A., and Stoica, I. (eds.), *Proceedings of Machine Learning and Systems*, volume 3, pp. 733–746, 2021a. URL https://proceedings.mlsys.org/paper/2021/file/ca46c1b9512a7a8315fa3c5a946e8265-Paper.pdf.

Müller, M. N., Makarchuk, G., Singh, G., Püschel, M., and Vechev, M. Prima: Precise and general neural network certification via multi-neuron convex relaxations. *arXiv preprint arXiv:2103.03638*, 2021b.

Müller, C., Singh, G., Püschel, M., and Vechev, M. T. Neural network robustness verification on gpus. *CoRR*, abs/2007.10868, 2020. URL https://arxiv.org/abs/2007.10868.

Ruan, W., Huang, X., and Kwiatkowska, M. Reachability analysis of deep neural networks with provable guarantees. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 2651–2659. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi:10.24963/ijcai.2018/368. URL https://doi.org/10.24963/ijcai.2018/368.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2018.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021. URL https://openreview.net/forum?id=Mm3gxxTfT7A.

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

Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.-W., Huang, M., Kailkhura, B., Lin, X., and Hsieh, C.-J. Automatic perturbation analysis for scalable certified robustness and beyond. In *3Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.

Zhao, Z., Fan, L., and Han, Z. Hybrid quantum benders' decomposition for mixed-integer linear programming. *arXiv preprint arXiv:2112.07109*, 2021.

For the sake of readability, we report here an excerpt from Franco et al. (2022).

A. Benders Decomposition

Let us rewrite Equation 5 as $\min_{\mathbf{y}} q(\mathbf{y})$ where:

$$q(\mathbf{y}) = \min_{\mathbf{z}} \{\mathbf{g}^\intercal \mathbf{z} \mid \boldsymbol{A}\mathbf{z} + \boldsymbol{B}\mathbf{y} \geq \mathbf{b},\ \boldsymbol{C}\mathbf{z} \geq \mathbf{d}\}. \tag{A.8}$$

Here we view the vector of binary variables $\mathbf{y}$ as given. Hence, we decoupled $\mathbf{y}$ from the rest of the program resulting in a LP. The dual formulation of $q(\mathbf{y})$ is given as:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \{\boldsymbol{\alpha}(\mathbf{b} - \mathbf{B}\mathbf{y}) + \boldsymbol{\beta}\mathbf{d} \mid \boldsymbol{\alpha}\boldsymbol{A} + \boldsymbol{\beta}\boldsymbol{C} = \mathbf{g}^\intercal\}, \tag{A.9}$$

where $\boldsymbol{\alpha} \in \mathbb{R}_+^{m_b}$ and $\boldsymbol{\beta} \in \mathbb{R}_+^{m_d}$ are row vectors. Since $\mathbf{y}$ is constant within the optimization of Equation A.8, the optimization program is a linear program and we thus have strong duality[5] The optimal objective value of Equation A.9 is infinity if Equation A.8 is infeasible. If Equation A.8 is feasible it has to have the same finite value. Thus, it should be either finite or positive infinity. Therefore, for an optimal value of the dual formulation of Equation A.9, $\mathbf{g}^\intercal - \boldsymbol{\alpha}\boldsymbol{A} - \boldsymbol{\beta}^\intercal\boldsymbol{C}$ has to be zero. Hence, we know the optimal feasible objective will only be the result of the remaining terms that are not interacting with $\mathbf{z}$ as all of them need to cancel out. Following Benders decomposition, we can formulate the objective of Equation A.9 as a linear combination of extreme rays and points of the feasible region. We denote as Λ_r and Λ_p the set of extreme rays and extreme points of the set $\{(\boldsymbol{\alpha}, \boldsymbol{\beta}) \mid \boldsymbol{\alpha}\boldsymbol{A} + \boldsymbol{\beta}\boldsymbol{C} = \mathbf{g}^\intercal, \boldsymbol{\alpha} \geq 0, \boldsymbol{\beta} \geq 0\}$.

Similarly to Benders (1962), we rewrite the optimization of Equation 5 to obtain:

$$\min_{\mathbf{y},\,\eta} \ \eta, \tag{A.10a}$$

$$\text{s.t. } \boldsymbol{\alpha}^{[k]}(\mathbf{b} - \boldsymbol{B}\mathbf{y}) + \boldsymbol{\beta}^{[k]}\mathbf{d} \leq \eta,\ \forall(\boldsymbol{\alpha}^{[k]}, \boldsymbol{\beta}^{[k]}) \in \Lambda_p, \tag{A.10b}$$

$$\boldsymbol{\alpha}^{[k]}(\mathbf{b} - \boldsymbol{B}\mathbf{y}) + \boldsymbol{\beta}^{[k]}\mathbf{d} \leq 0,\ \forall(\boldsymbol{\alpha}^{[k]}, \boldsymbol{\beta}^{[k]}) \in \Lambda_r, \tag{A.10c}$$

where $\eta \in \mathbb{R}$ is a scalar. This problem is known as *master* problem on Benders decomposition.

[5] i.e. the optimal objective value of the primal equals the optimal value of the dual.