Journal of Machine Learning Research 23 (2022) 1-8 Submitted 3/22; Revised 7/22; Published 11/22

OMLT: Optimization & Machine Learning Toolkit

Francesco Ceccon[1,*] FRANCESCO.CECCON14@IMPERIAL.AC.UK
Jordan Jalving[2,*] JHJALVI@SANDIA.GOV
Joshua Haddad[2] JIHADDA@SANDIA.GOV
Alexander Thebelt[1] ALEXANDER.THEBELT18@IMPERIAL.AC.UK
Calvin Tsay[1] C.TSAY@IMPERIAL.AC.UK
Carl D Laird[3,†] CLAIRD@ANDREW.CMU.EDU
Ruth Misener[1,†] R.MISENER@IMPERIAL.AC.UK
[1] *Department of Computing, Imperial College London, 180 Queen's Gate, SW7 2AZ, UK*
[2] *Center for Computing Research, Sandia National Laboratories, Albuquerque, NM 87123, USA*
[3] *Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA*

Editor: Sebastian Schelter

Abstract

The optimization and machine learning toolkit (OMLT) is an open-source software package incorporating neural network and gradient-boosted tree surrogate models, which have been trained using machine learning, into larger optimization problems. We discuss the advances in optimization technology that made OMLT possible and show how OMLT seamlessly integrates with the algebraic modeling language Pyomo. We demonstrate how to use OMLT for solving decision-making problems in both computer science and engineering.

Keywords: Optimization formulations, Pyomo, neural networks, gradient-boosted trees

1. Introduction

The optimization and machine learning toolkit (`https://github.com/cog-imperial/OMLT`, OMLT 1.0) is an open-source software package enabling optimization over high-level representations of neural networks (NNs) and gradient-boosted trees (GBTs). Optimizing over trained surrogate models allows integration of NNs or GBTs into larger decision-making problems. Computer science applications include maximizing a neural acquisition function (Volpp et al., 2019) or verifying neural networks (Botoeva et al., 2020). In engineering, machine learning models may replace complicated constraints or act as surrogates in larger design and operations problems (Henao and Maravelias, 2011). OMLT 1.0 supports GBTs through an ONNX[1] interface and NNs through both ONNX and Keras (Chollet et al., 2015) interfaces. OMLT transforms these pre-trained machine learning models into the algebraic modeling language Pyomo (Bynum et al., 2021) to encode the optimization formulations.

Mathematical optimization solver software requires, as input, a formulation including the decision variable(s), objective(s), constraint(s), and any parameters. OMLT automates

$*$. These authors contributed equally. †. These authors contributed equally.
1. `https://github.com/onnx/onnx`

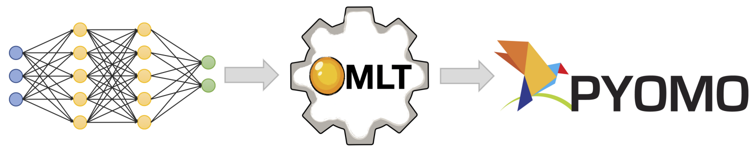

Figure 1: OMLT automatically translates high-level representations of machine learning models into variables and constraints suitable for optimization.

the otherwise tedious and error-prone task of translating already-trained NN and GBT models into optimization formulations suitable for solver software. For example, the ReLU NN example in `neural_network_formulations.ipynb` switches (in ≤ 1 line of code) between 3 formulations (*complementarity*, *big-M*, *partition*) with 248, 308, and 428 constraints, respectively. OMLT 1.0 saves the time needed to code/debug each optimization formulation.

2. Use cases for optimizing trained machine learning surrogates

Complete NN verification is an AI use case for embedding trained NNs into an optimization problem (Tjeng et al., 2018). One such verification problem asks: Given a trained NN, a labeled target image, and a distance metric, does an image with a different, adversarial label exist within a fixed perturbation? Such NN verification can be formulated as an optimization problem (Lomuscio and Maganti, 2017). Our `mnist_example_{dense, cnn}.ipynb` notebooks verify dense and convolutional NNs on MNIST (LeCun et al., 2010). The problems in the `mnist_example_{dense, cnn}.ipynb` notebooks could also be addressed effectively with dedicated NN verification tools, e.g., Beta-CROWN (Wang et al., 2021). Related applications however, e.g., minimally distorted adversaries (Croce and Hein, 2020) and lossless compression (Serra et al., 2020), cannot be addressed by the dedicated NN verification tools. These other applications can still be implemented in OMLT.

An engineering use case is the `auto-thermal-reformer{-relu}.ipynb` notebook, which develops an NN surrogate with data from a process model built using IDAES-PSE (Lee et al., 2021). Here, the goal is to build surrogate models for complex processes to improve optimization convergence reliability or replace simulation-based models with equation-oriented optimization formulations. Kilwein et al. (2021) mix optimal power flow constraints with security constraints learned from data. Other examples include grey-box optimization or hybrid mechanistic / data-driven optimization (Boukouvala et al., 2016, 2017; Wilson and Sahinidis, 2017; Boukouvala and Floudas, 2017; Huster et al., 2020; Thebelt et al., 2022b).

3. Library design

Input interface. OMLT uses ONNX as an input interface because ONNX implicitly allows OMLT to support packages such as Keras (Chollet et al., 2015), PyTorch (Paszke et al., 2019), and TensorFlow (Abadi et al., 2015) via the ONNX interoperatability features.

Formulating surrogate models as a Pyomo block. OMLT uses Pyomo, a Python-based algebraic modeling language for optimization (Bynum et al., 2021). Most machine learning frameworks use Python as the primary interface, so Python is a natural base for OMLT. Pyomo has a flexible modeling interface to Pyomo-enabled solvers: switching solvers allows OMLT users to select the best optimization solver for an application without explicitly interfacing with each solver. OMLT heavily relies on Pyomo. First, Pyomo's efficient auto-differentiation of nonlinear functions using the AMPL solver library (Gay, 1997), enables NN nonlinear activation functions. Second, Pyomo's many extensions, e.g., decomposition for large-scale problems, allow OMLT to interface with state-of-the-art optimization approaches.

Most importantly, OMLT uses Pyomo *blocks*. In OMLT, Pyomo blocks encapsulate the GBT or NN components of a larger optimization formulation. Blocks simplify OMLT: users only need to understand the input/output structure of the NN or GBT when linking to a Pyomo block's inputs and outputs. The block abstraction allows OMLT users to ignore specific optimization details yet experiment with competing formulations. The block abstraction also helps OMLT developers wishing to create new optimization formulations and algorithms: Pyomo blocks provide flexible semantic structures for specialized algorithms.

`OmltBlock` implementation `OmltBlock` is a Pyomo block that delegates generating the optimization formulation of the surrogate model to the `_PyomoFormulation` object. OMLT users create the input/output objects, e.g., constraints, that link the surrogate model to the larger optimization problem and the user-defined variables. The optimization formulation of the surrogate is generated automatically from its higher level (ONNX or Keras) representation. OMLT users may also specify a scaling object for the variables and a dictionary of variable bounds. The scaling and variable bound information may not be present in ONNX or Keras representations, but is required for some optimization formulations. Notebook `neural_network_formulations.ipynb` demonstrates using variable scaling and bounds.

`NetworkDefinition` for neural networks For GBTs, OMLT automatically generates the optimization formulation from the higher level, e.g., ONNX, representation. For neural networks, OMLT instead generates an intermediate representation (`NetworkDefinition`) that acts as the gateway to several alternative mathematical optimization formulations.

Alternative optimization formulations A major thread of research develops new optimization formulations for machine learning surrogates (Fischetti and Jo, 2018; Raghunathan et al., 2018; Singh et al., 2019; Anderson et al., 2020; Tjandraatmadja et al., 2020; Dathathri et al., 2020). OMLT uses Pyomo blocks to formulate surrogate models as an optimization objective or as constraints. The OMLT 1.0 `GBTBigMFormulation` uses the Mišić (2020) and Mistry et al. (2021) formulation and thereby simplifies our GBT-based black-box optimization tool ENTMOOT (Thebelt et al., 2021, 2022a). The OMLT NN implementation supports dense and convolutional layers. For users developing custom formulations, we suggest `FullSpaceNNFormulation`. OMLT also offers specific formulations which often arise in practice; we group these with respect to smooth versus non-smooth activation functions:

- **Smooth**. OMLT 1.0 supports linear, softplus, sigmoid, and tanh NN activation functions with 2 competing formulations: `FullSpaceSmoothNNFormulation` and `ReducedSpaceSmoothNNFormulation` (Schweidtmann and Mitsos, 2019). The full-space formulation uses Pyomo variables and constraints to represent auxiliary variables and activation constraints. The reduced-space formulation instead uses Pyomo `Expression` objects to create a single constraint for each NN output.

- **Non-smooth**. OMLT 1.0 supports ReLU NN activation with competing formulations `ReluBigMFormulation` (Anderson et al., 2020), `ReluComplementarityFormulation` (Yang et al., 2021), and `ReluPartitionFormulation` [dense layers only in OMLT 1.0] (Tsay et al., 2021).

Subclasses of `_PyomoFormulation`, representing the full-space and reduced-space formulations, take an `OmltBlock` and allow us to represent the surrogate model using mathematical constraints and variables. The full and reduced-space subclasses have `_DEFAULT_ACTIVATION_CONSTRAINTS` for each activation function. Users wishing to try different optimization formulations beyond the provided alternatives should override these defaults. An existing alternative for ReLU activation functions is the `ReLUPartitionFormulation`. Notebook `neural_network_formulations` experiments with competing formulations.

4. Comparison to related work

Subsets of OMLT 1.0 are available elsewhere.[2] MeLOn (Schweidtmann and Mitsos, 2019) parses its own XML representation of dense NNs with sigmoidal activations and creates a reduced-space optimization formulation. JANOS (Bergman et al., 2022) parses scikit-learn representations of dense ReLU NNs and logistic regression. Janos creates a Gurobi formulation to optimize over these surrogate models. reluMIP (Lueg et al., 2021) parses TensorFlow representations of dense NNs with ReLU activation functions and creates a big-M formulation. OptiCL (Maragno et al., 2021) creates its own machine learning surrogates and then develops mixed-integer formulations of its own surrogates. gurobi-machinelearning interfaces directly with the Gurobi solver.

OMLT is a more general tool incorporating both NNs and GBTs, many input models via ONNX interoperability, both dense and convolutional layers, several activation functions,[3] and various optimization formulations. The literature often presents these different optimization formulations as competitors, e.g., our *partition-based* formulation competes with the *big-M* formulation for ReLU NNs (Kronqvist et al., 2021; Tsay et al., 2021). In OMLT, competing optimization formulations become alternatives: users can switch between the Section 3 formulations and find the best for a specific application.

5. Outlook & conclusions

Higher-level representations, such as those available in ONNX, Keras, and PyTorch, are very useful for modelling neural networks and gradient-boosted trees. OMLT extends the usefulness of these representations to larger decision-making problems by automating the transformation of these pre-trained models into variables and constraints suitable for optimization solvers. In other words, OMLT allows us to extend any optimization model that can be expressed in Pyomo to include neural networks or gradient-boosted trees. Our implementation makes it possible to seamlessly compare different formulations that are typically presented as competitors in the literature.

2. `https://git.rwth-aachen.de/avt-svt/public/MeLOn`, `http://janos.opt-operations.com`, `https://github.com/ChemEngAI/ReLU_ANN_MILP`, `https://github.com/hwiberg/OptiCL`, `https://github.com/Gurobi/gurobi-machinelearning`
3. In OMLT 1.0: ReLU, linear, softplus, sigmoid, and tanh

6. Acknowledgements

FC, CT, and RM were funded by Engineering & Physical Sciences Research Council Fellowships (EP/T001577/1 & EP/P016871/1). CT was also supported by a Imperial College Research Fellowship. BASF SE funded the PhD studentship of AT.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. JJ and JH were funded in part by the Institute for the Design of Advanced Energy Systems (IDAES) with funding from the Office of Fossil Energy, Cross-Cutting Research, U.S. Department of Energy. JJ, JH, and CL were also funded by Sandia National Laboratories Laboratory Directed Research and Development (LDRD) program.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183:3–39, 2020.

David Bergman, Teng Huang, Philip Brooks, Andrea Lodi, and Arvind U Raghunathan. Janos: an integrated predictive and prescriptive modeling framework. *INFORMS Journal on Computing*, 34(2):807–816, 2022.

Elena Botoeva, Panagiotis Kouvaros, Jan Kronqvist, Alessio Lomuscio, and Ruth Misener. Efficient verification of ReLU-based neural networks via dependency analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3291–3299, 2020.

Fani Boukouvala and Christodoulos A Floudas. ARGONAUT: Algorithms for global optimization of constrained grey-box computational problems. *Optimization Letters*, 11(5): 895–913, 2017.

Fani Boukouvala, Ruth Misener, and Christodoulos A. Floudas. Global optimization advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO. *European Journal of Operational Research*, 252(3):701 – 727, 2016.

Fani Boukouvala, MM Faruque Hasan, and Christodoulos A Floudas. Global optimization of general constrained grey-box models: new method and its application to constrained pdes for pressure swing adsorption. *Journal of Global Optimization*, 67(1-2):3–42, 2017.

Michael L Bynum, Gabriel A Hackebeil, William E Hart, Carl D Laird, Bethany L Nicholson, John D Siirola, Jean-Paul Watson, and David L Woodruff. *Pyomo—Optimization Modeling in Python*, volume 67. Springer Nature, 2021.

François Chollet et al. Keras. `https://keras.io`, 2015.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205, 2020.

Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang, et al. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. *Advances in Neural Information Processing Systems*, 33:5318–5331, 2020.

Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

David M Gay. Hooking your solver to AMPL. Technical report, Citeseer, 1997.

Carlos A Henao and Christos T Maravelias. Surrogate-based superstructure optimization framework. *AIChE Journal*, 57(5):1216–1232, 2011.

Wolfgang R Huster, Artur M Schweidtmann, Jannik T Lüthje, and Alexander Mitsos. Deterministic global superstructure-based optimization of an organic Rankine cycle. *Computers & Chemical Engineering*, 141:106996, 2020.

Zachary Kilwein, Fani Boukouvala, Carl Laird, Anya Castillo, Logan Blakely, Michael Eydenberg, Jordan Jalving, and Lisa Batsch-Smith. Ac-optimal power flow solutions with security constraints from deep neural network models. In Metin Türkay and Rafiqul Gani, editors, *31st European Symposium on Computer Aided Process Engineering*, volume 50 of *Computer Aided Chemical Engineering*, pages 919–925. Elsevier, 2021.

Jan Kronqvist, Ruth Misener, and Calvin Tsay. Between steps: Intermediate relaxations between big-M and convex hull formulations. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 299–314. Springer, 2021.

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Andrew Lee, Jaffer H Ghouse, John C Eslick, Carl D Laird, John D Siirola, Miguel A Zamarripa, Dan Gunter, John H Shinn, Alexander W Dowling, Debangsu Bhattacharyya, et al. The IDAES process modeling framework and model library—Flexibility for process simulation and optimization. *Journal of Advanced Manufacturing and Processing*, page e10095, 2021.

Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward ReLU neural networks. *arXiv preprint arXiv:1706.07351*, 2017.

Laurens Lueg, Bjarne Grimstad, Alexander Mitsos, and Artur M. Schweidtmann. reluMIP: Open source tool for MILP optimization of ReLU neural networks, 2021. URL `https://github.com/ChemEngAI/ReLU_ANN_MILP`.

Donato Maragno, Holly Wiberg, Dimitris Bertsimas, S Ilker Birbil, Dick den Hertog, and Adejuyigbe Fajemisin. Mixed-integer optimization with constraint learning. *arXiv preprint arXiv:2111.04469*, 2021.

Velibor V Mišić. Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624, 2020.

Miten Mistry, Dimitrios Letsios, Gerhard Krennrich, Robert M Lee, and Ruth Misener. Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded. *INFORMS Journal on Computing*, 33(3):1103–1119, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. 2019.

Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, volume 31, pages 10877–10887, 2018.

Artur M Schweidtmann and Alexander Mitsos. Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications*, 180(3):925–948, 2019.

Thiago Serra, Abhinav Kumar, and Srikumar Ramalingam. Lossless compression of deep neural networks. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 417–430. Springer, 2020.

Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin Vechev. Beyond the single neuron convex barrier for neural network certification. In *Advances in Neural Information Processing Systems*, pages 15098–15109, 2019.

Alexander Thebelt, Jan Kronqvist, Miten Mistry, Robert M Lee, Nathan Sudermann-Merx, and Ruth Misener. ENTMOOT: A framework for optimization over ensemble tree models. *Computers & Chemical Engineering*, 151:107343, 2021.

Alexander Thebelt, Calvin Tsay, Robert M. Lee, Nathan Sudermann-Merx, David Walz, Tom Tranter, and Ruth Misener. Multi-objective constrained optimization for energy applications via tree ensembles. *Applied Energy*, 306:118061, 2022a.

Alexander Thebelt, Johannes Wiebe, Jan Kronqvist, Calvin Tsay, and Ruth Misener. Maximizing information from chemical engineering data sets: Applications to machine learning. *Chemical Engineering Science*, 252:117469, 2022b.

Christian Tjandraatmadja, Ross Anderson, Joey Huchette, Will Ma, Krunal Kishor Patel, and Juan Pablo Vielma. The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. In *Advances in Neural Information Processing Systems*, volume 33, pages 21675–21686, 2020.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2018.

Calvin Tsay, Jan Kronqvist, Alexander Thebelt, and Ruth Misener. Partition-based formulations for mixed-integer optimization of trained ReLU neural networks. In *Advances in Neural Information Processing Systems*, 2021.

Michael Volpp, Lukas P Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-learning acquisition functions for transfer learning in bayesian optimization. In *International Conference on Learning Representations*, 2019.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *Advances in Neural Information Processing Systems*, 34, 2021.

Zachary T Wilson and Nikolaos V Sahinidis. The ALAMO approach to machine learning. *Computers & Chemical Engineering*, 106:785–795, 2017.

Dominic Yang, Prasanna Balaprakash, and Sven Leyffer. Modeling design and control problems involving neural network surrogates. *arXiv preprint arXiv:2111.10489*, 2021.