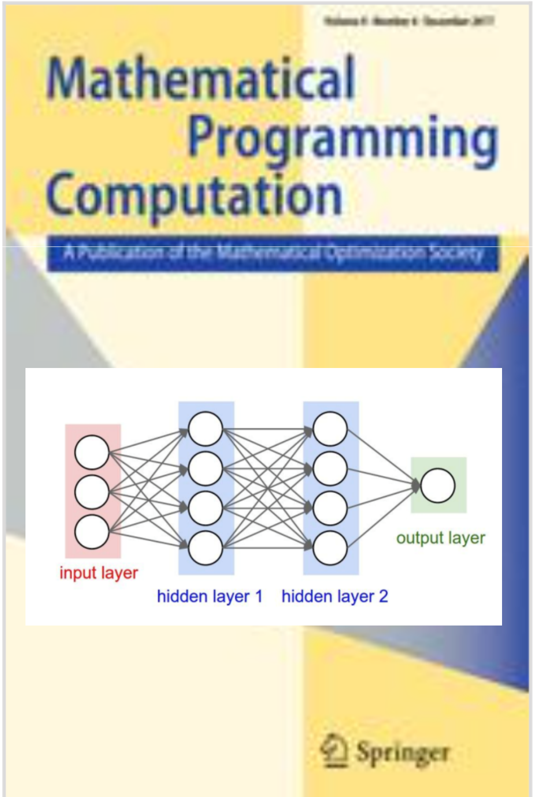
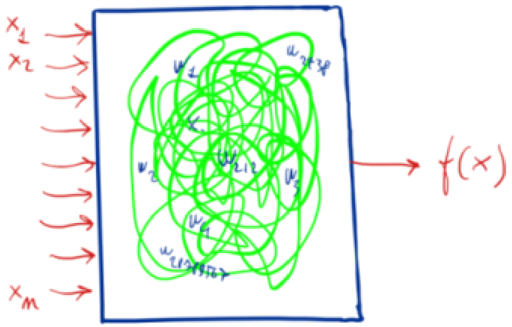
Deep Learning and Mixed Integer Optimization

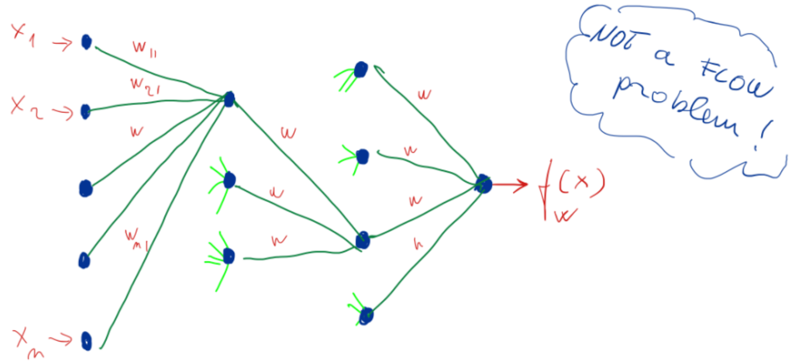
Matteo Fischetti, University of Padova



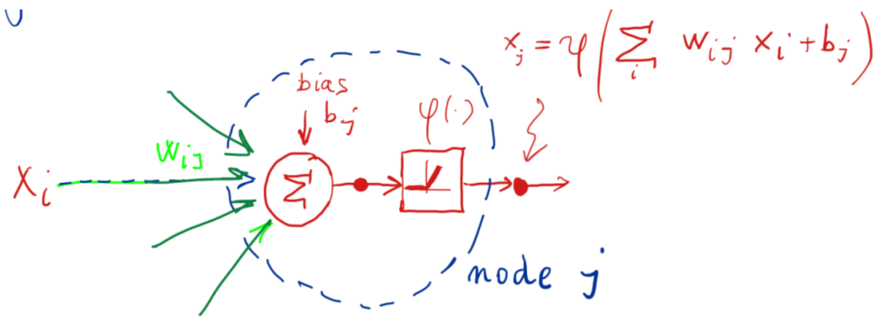
(Deep) Neural Networks (DNNs)

- Machine whose parameters *w*'s are organized in a layered feed-forward **network** (DAG = Directed Acyclic Graph)



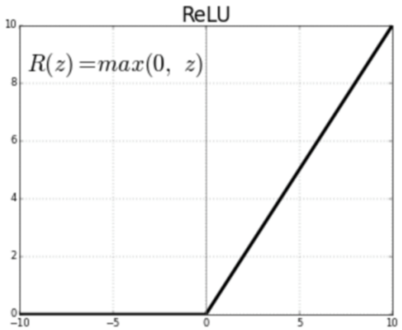


- Each node (or "**neuron**") makes a **weighted sum** of the outputs of the previous layer and applies a nonlinear **activation function**



Modeling ReLU activations

- Recent work on DNNs almost invariably only use ReLU activations



$$x = ReLU(w^T y + b).$$

- Easily modeled in a MI(N)LP as $w^T y + b = x - s, \quad x \geq 0, \quad s \geq 0$
  - plus the bilinear condition $x\,s \leq 0.$
  - or, alternatively, the indicator constraints $\left.\begin{aligned} z = 1 \rightarrow x \leq 0 \\ z = 0 \rightarrow s \leq 0 \\ z \in \{0,1\} \end{aligned}\right\}$

The DNN is a 0-1 MILP (for fixed *w*'s)

$$\min \sum_{k=0}^{K}\sum_{j=1}^{n_k} c_j^k x_j^k + \sum_{k=1}^{K}\sum_{j=1}^{n_k} \gamma_j^k z_j^k$$

$$\left.\begin{aligned}
\sum_{i=1}^{n_{k-1}} w_{ij}^{k-1} x_i^{k-1} + b_j^{k-1} = x_j^k - s_j^k \\
x_j^k, s_j^k \ge 0 \\
z_j^k \in \{0,1\} \\
z_j^k = 1 \to x_j^k \le 0 \\
z_j^k = 0 \to s_j^k \le 0
\end{aligned}\right\} k = 1,\ldots,K,\ j = 1,\ldots,n_k$$

$$lb_j^0 \le x_j^0 \le ub_j^0, \qquad j = 1,\ldots,n_0$$

$$\left.\begin{aligned}
lb_j^k \le x_j^k \le ub_j^k \\
\overline{lb}_j^k \le s_j^k \le \overline{ub}_j^k
\end{aligned}\right\} k = 1,\ldots,K,\ j = 1,\ldots,n_k.$$

Application: Adversarial problems

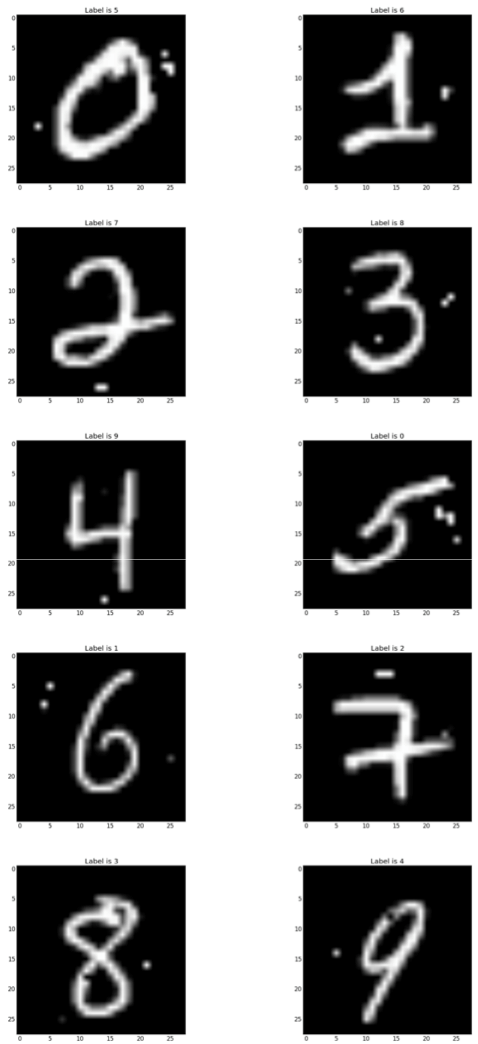


Fig. 2 Adversarial examples computed through our 0-1 MILP model; the reported label is the one having maximum activation according to the DNN (that we imposed to be the true label plus 5, modulo 10). Note that the change of just few well-chosen pixels often suffices to fool the DNN and to produce a wrong classification.

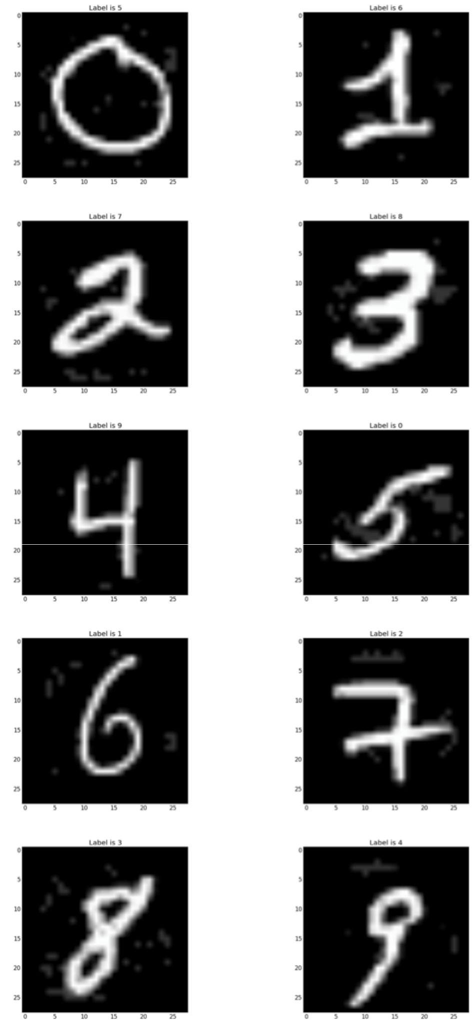


Fig. 3 Adversarial examples computed through our 0-1 MILP model as in Figure 2, but imposing that the no pixel can be changed by more than 0.2 (through the additional conditions $d_j \leq 0.2$ for all j).

Adversarial Problem

Trick the DNN
by changing
few well-chosen pixels

Solvable to proven optimality
(for small DNNs) in a matter of
seconds/minutes
by using a
black-box MILP solver

	basic model				improved model			
	%solved	%gap	nodes	time (s)	%solved	%gap	nodes	time (s)
DNN1	100	0.0	1,903	1.0	100	0.0	552	0.6
DNN2	97	0.2	77,878	48.2	100	0.0	11,851	7.5
DNN3	64	11.6	228,632	158.5	100	0.0	20,309	12.1
DNN4	24	38.1	282,694	263.0	98	0.7	68,563	43.9
DNN5	7	71.8	193,725	290.9	67	11.4	76,714	171.1

Table 1 Comparison of the basic and improved models with a time limit of 300 sec.s, clearly showing the importance of bound tightening in the improved model. In this experiment, the preprocessing time needed to optimally compute the tightened bounds is not taken into account.

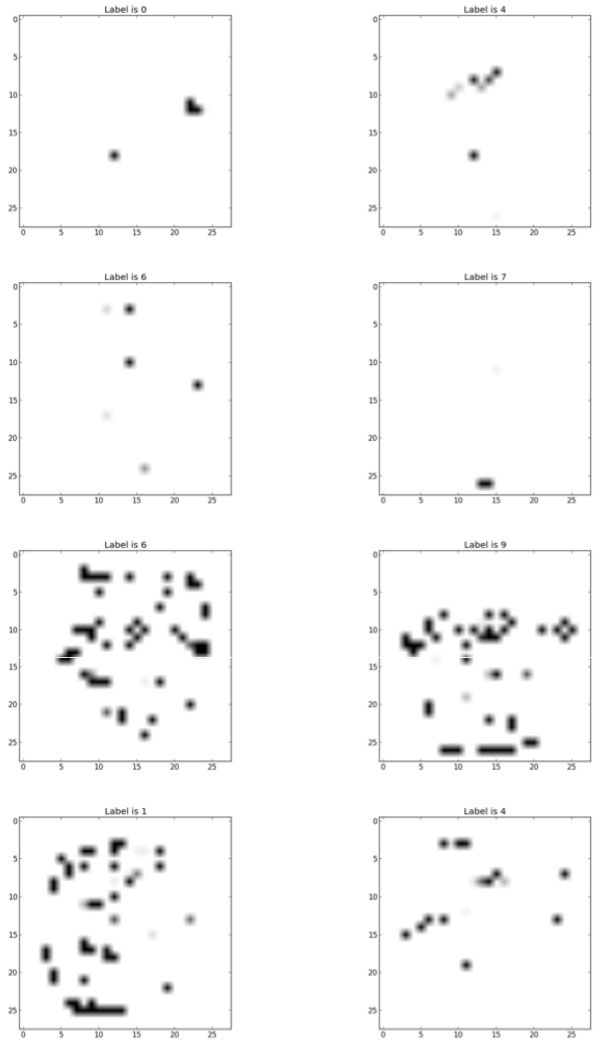


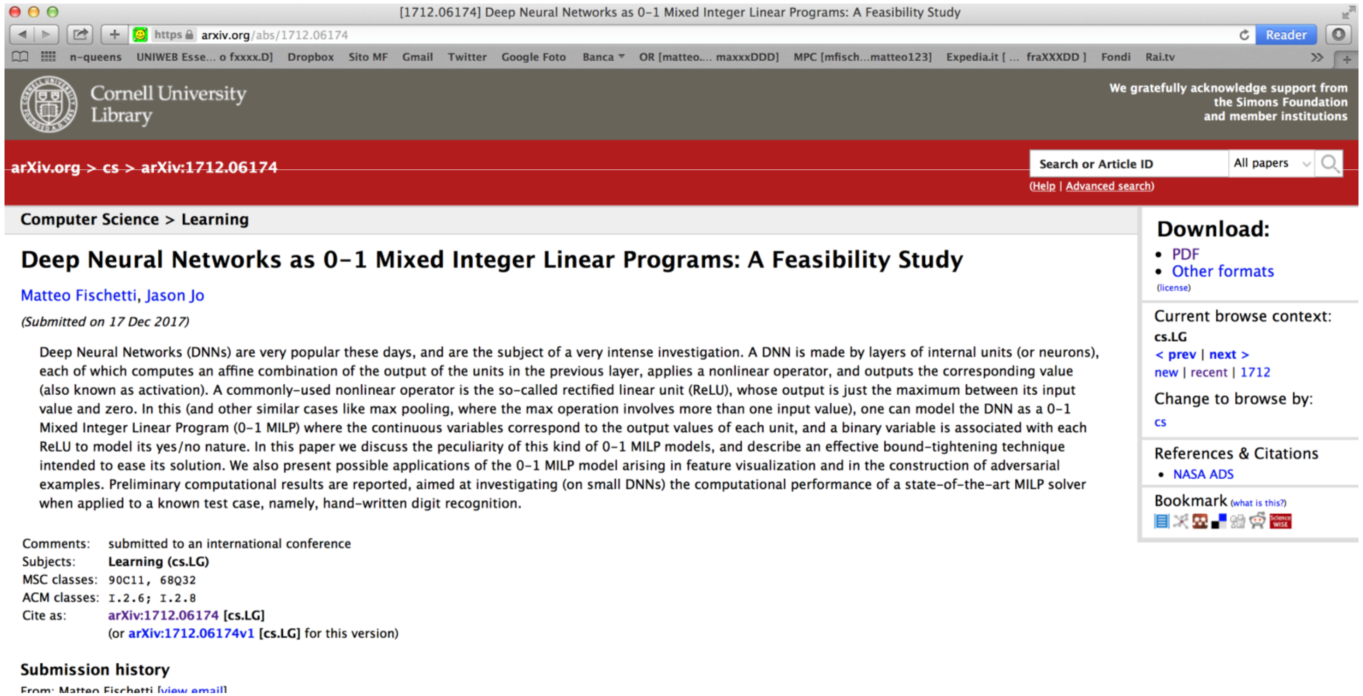
Fig. 4 Pixel changes (absolute value) that suffice to trick the DNN: the four top subfigures correspond to the model where pixels can change arbitrarily, while those on the bottom refer to the case where each pixel cannot change by more than 0.2 (hence more pixels need be changed). To improve readability, the black/white map has been reverted and scaled, i.e., white corresponds to unchanged pixels ($d_j = 0$) while black corresponds to the maximum allowed change ($d_j = 1$ for the four top figures, $d_j = 0.2$ for the four bottom ones).

For more information…

Slides available at http://www.dei.unipd.it/~fisch/papers/slides/

Paper:

M. Fischetti, J. Jo, "Deep Neural Networks as 0-1 Mixed Integer Linear Programs: A Feasibility Study", 2017, arXiv preprint arXiv:1712.06174 (accepted in CPAIOR 2018)


[1712.06174] Deep Neural Networks as 0-1 Mixed Integer Linear Programs: A Feasibility Study

Cornell University Library

We gratefully acknowledge support from the Simons Foundation and member institutions

arXiv.org > cs > arXiv:1712.06174

Computer Science > Learning

Deep Neural Networks as 0-1 Mixed Integer Linear Programs: A Feasibility Study

Matteo Fischetti, Jason Jo

(Submitted on 17 Dec 2017)

Deep Neural Networks (DNNs) are very popular these days, and are the subject of a very intense investigation. A DNN is made by layers of internal units (or neurons), each of which computes an affine combination of the output of the units in the previous layer, applies a nonlinear operator, and outputs the corresponding value (also known as activation). A commonly-used nonlinear operator is the so-called rectified linear unit (ReLU), whose output is just the maximum between its input value and zero. In this (and other similar cases like max pooling, where the max operation involves more than one input value), one can model the DNN as a 0-1 Mixed Integer Linear Program (0-1 MILP) where the continuous variables correspond to the output values of each unit, and a binary variable is associated with each ReLU to model its yes/no nature. In this paper we discuss the peculiarity of this kind of 0-1 MILP models, and describe an effective bound-tightening technique intended to ease its solution. We also present possible applications of the 0-1 MILP model arising in feature visualization and in the construction of adversarial examples. Preliminary computational results are reported, aimed at investigating (on small DNNs) the computational performance of a state-of-the-art MILP solver when applied to a known test case, namely, hand-written digit recognition.

Comments: submitted to an international conference
Subjects: Learning (cs.LG)
MSC classes: 90C11, 68Q32
ACM classes: I.2.6; I.2.8
Cite as: arXiv:1712.06174 [cs.LG]
(or arXiv:1712.06174v1 [cs.LG] for this version)

Submission history

Download:
- PDF
- Other formats

Current browse context: cs.LG

Change to browse by: cs

References & Citations
- NASA ADS

Bookmark