6-2023

The BeMi Stardust: A structured ensemble of Binarized Neural Networks

Ambrogio Maria BERNARDELLI

Stefano GUALANDI

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

Simone MILANESI

The BeMi Stardust: a Structured Ensemble of Binarized Neural Networks

Ambrogio Maria Bernardelli[1] [iD]   Stefano Gualandi[1] [iD]

Hoong Chuin Lau[2] [iD]   Simone Milanesi[1] [iD]

[1] Department of Mathematics, University of Pavia, Italy
[2] School of Information Systems, Singapore Management University
{ambrogiomaria.bernardelli01, simone.milanesi01}@universitadipavia.it
stefano.gualandi@unipv.it, hclau@smu.edu.sg

Abstract. Binarized Neural Networks (BNNs) are receiving increasing attention due to their lightweight architecture and ability to run on low-power devices. The state-of-the-art for training classification BNNs restricted to few-shot learning is based on a Mixed Integer Programming (MIP) approach. This paper proposes the BeMi ensemble, a structured architecture of BNNs based on training a single BNN for each possible pair of classes and applying a majority voting scheme to predict the final output. The training of a single BNN discriminating between two classes is achieved by a MIP model that optimizes a lexicographic multi-objective function according to robustness and simplicity principles. This approach results in training networks whose output is not affected by small perturbations on the input and whose number of active weights is as small as possible, while good accuracy is preserved. We computationally validate our model using the MNIST and Fashion-MNIST datasets using up to 40 training images per class. Our structured ensemble outperforms both BNNs trained by stochastic gradient descent and state-of-the-art MIP-based approaches. While the previous approaches achieve an average accuracy of 51.1% on the MNIST dataset, the BeMi ensemble achieves an average accuracy of 61.7% when trained with 10 images per class and 76.4% when trained with 40 images per class.

Keywords: Binarized neural networks · Mixed-integer linear programming · Structured ensemble of neural networks

1 Introduction

State-of-the-art Neural Networks (NNs) contain a huge number of neurons organized in several layers, and they require an immense amount of data for training [13]. The training process is computationally demanding and is typically performed by stochastic gradient descent algorithms running on large GPU-based clusters. Whenever the trained (deep) neural network contains many neurons, also the network deployment is computationally demanding. However, in real-life industrial applications, GPU-based clusters are often unavailable or too expensive, and training data is scarce and contains only a few data points per class.

Binary Neural Networks (BNNs) were introduced in [8] as a response to the challenge of running NNs on low-power devices. BNNs contain only binary weights and binary activation functions, and hence they can be implemented using only bit-wise operations, which are very power-efficient. However, the training of BNNs raises interesting challenges for gradient-based approaches due to their combinatorial structure. In [22], the authors show that the training of a BNN performed by a hybrid constraint programming (CP) and mixed integer programming (MIP) approach outperforms, in terms of accuracy, the stochastic gradient approach proposed in [8] by a large margin, if restricted to a few-shot-learning context [23]. Indeed, the main challenge in training a NN by an exact MIP-based approach is the limited amount of training data that can be used since, otherwise, the size of the optimization model explodes. However, in [21], the hybrid CP and MIP method was further extended to integer-valued neural networks: exploiting the flexibility of MIP solvers, the authors were able to (i) minimize the number of neurons during training and (ii) increase the number of data points used during training by introducing a MIP batch training method.

We remark that training a NN with a MIP-based approach is more challenging than solving a verification problem, as in [6,1], even if the structure of the nonlinear constraints modeling the activation functions is similar. In NNs verification [12], the weights are given as input, while in MIP-based training, the weights are the decision variables that must be computed. Furthermore, recent works aim at producing compact and lightweight NNs that maintain acceptable accuracy, e.g., in terms of parameter pruning [19,26], loss function improvement [20], gradient approximation [17], and network topology structure [14].

Contributions. In this paper, we propose the BeMi[3] ensemble, a structured ensemble of BNNs, where each single BNN is trained by solving a lexicographic multi-objective MIP model. Given a classification task over k classes, the main idea is to train $\frac{k(k-1)}{2}$ BNNs, where every single network learns to discriminate only between a given pair of classes. When a new data point (e.g., a new image) must be classified, it is first fed into the $\frac{k(k-1)}{2}$ trained BNNs, and later, using a Condorcet-inspired majority voting scheme [25], the most frequent class is predicted as output. For training every single BNN, our approach extends the methods introduced in [22] and [21] by proposing an improved lexicographic multi-objective function that minimizes the classification errors, maximizes the margins of every single neuron, and minimizes the number of non-zero weights. Notice that maximizing the margins allows the preactivation to remain far from the switching point of the activation function, and it yields robust NNs. Our computational results using the MNIST and the Fashion-MNIST dataset show that the BeMi ensemble permits to use for training up to 40 data points per class, and permits reaching an accuracy of 78.8% for MNIST and 72.9% for Fashion-MNIST. In addition, thanks to the multi-objective function that minimizes the number of neurons, up to 75% of weights are set to zero for MNIST, and up to 50% for Fashion-MNIST.

---

[3] Acronym from the last names of the two young authors who had this intuition.

Outline. The outline of this paper is as follows. Section 2 introduces the notation and defines the problem of training a single BNN with the existing MIP-based methods. Section 3 presents the BeMi ensemble, the Condorcet majority voting scheme, and the improved MIP model to train a single BNN. Section 4 presents the computational results of the MNIST and Fashion-MNIST. Finally, Section 5 concludes the paper with a perspective on future works.

2 Binary Neural Networks

In this section, we formally define a single BNN using the same notation as in [22], while, in the next section, we show how to define a structured ensemble of BNNs. The architecture of a BNN is defined by a set of layers $\mathcal{N} = \{N_0, N_1, \ldots, N_L\}$, where $N_l = \{1, \ldots, n_l\}$, and n_l is the number of neurons in the l-th layer. Let the training set be $\mathcal{X} := \{(\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^t, y^t)\}$, such that $\boldsymbol{x}^i \in \mathbb{R}^{n_0}$ and $y^i \in \{-1, +1\}^{n_L}$ for every $i \in T = \{1, 2, \ldots, t\}$. The first layer N_0 corresponds to the size of the input data points $\boldsymbol{x}^k$.

The link between neuron i in layer N_{l-1} and neuron j in layer N_l is modeled by weight $w_{ilj} \in \{-1, 0, +1\}$. Note that whenever a weight is set to zero, the corresponding link is removed from the network. Hence, during training, we are also optimizing the architecture of the BNN. The activation function is the binary function

$$\rho(x) := 2 \cdot \mathbb{1}(x \geq 0) - 1, \tag{1}$$

that is, a sign function reshaped such that it takes ± 1 values. The indicator function $\mathbb{1}(p)$ outputs $+1$ if proposition p is verified, and 0 otherwise.

To model the activation function (1) of the j-the neuron of layer N_l for data point $\boldsymbol{x}^k$, we introduce a binary variable $u_{lj}^k \in \{0, 1\}$ for the indicator function $\mathbb{1}(p)$. To rescale the value of u_{lj}^k in $\{-1, +1\}$ and model the activation function value, we introduce the auxiliary variable $z_{lj}^k = (2u_{lj}^k - 1)$. For the first input layer, we set $z_{0j}^k = x_j^k$; for the last layer, we account in the loss function whether z_{Lh}^k is different from y_h^k. The definition of the activation function becomes

$$z_{lj}^k = \rho\left(\sum_{i \in N_{l-1}} z_{(l-1)i}^k w_{ilj}\right) = 2 \cdot \mathbb{1}\left(\sum_{i \in N_{l-1}} v_{(l-1)i}^k w_{ilj} \geq 0\right) - 1 = 2u_{lj}^k - 1.$$

Notice that the activation function at layer N_l gives a nonlinear combination of the output of the neurons in the previous layer N_{l-1} and the weights w_{ilj} between the two layers. Section 3.3 shows how to formulate this activation function in terms of mixed integer linear constraints.

The choice of a family of parameters $W := \{w_{ilj}\}_{l \in \{1, \ldots, L\}, i \in N_{l-1}, j \in N_l}$ determines the classification function

$$f_W : \mathbb{R}^{n_0} \to \{\pm 1\}^{n_L}.$$

The training of a neural network is the process of computing the family W such that f_W classifies correctly both the given training data, that is, $f_W(\boldsymbol{x}^i) = y^i$ for $i = 1, \ldots, t$, and new unlabelled testing data.

The training of a binary neural network should target two objectives: (i) the resulting function f_W should generalize from the input data and be *robust* to noise in the input data; (ii) the resulting network should be *simple*, that is, with the smallest number of non-zero weights that permit to achieve the best accuracy. Deep neural networks are believed to be inherently robust because mini-batch stochastic gradient-based methods implicitly guide toward robust solutions [10,11,16]. However, as shown in [22], this is false for BNNs in a few-shot learning regime. On the contrary, MIP-based training with an appropriate objective function can generalize very well [22,21], but it does not apply to large training datasets, because the size of the MIP training model is proportional to the size of the training dataset. To generalize from a few data samples, the training algorithm should maximize the margins of the neurons. Intuitively, neurons with larger margins require larger changes to their inputs and weights before changing their activation values. This choice is also motivated by recent works showing that margins are good predictors for the generalization of deep convolutional NNs [9]. Regarding the second simplicity objective, a significant parameter is the number of connections [15]. The training algorithm should look for a BNN fitting the training data while minimizing the number of non-zero weights. This approach can be interpreted as a simultaneous compression during training. Although this objective is challenged in [5], it remains the basis of most forms of regularization used in modern deep learning [18].

MIP-based BNN training. In [22], two different MIP models are introduced: the Max-Margin, which aims to train robust BNNs, and the Min-Weight model, which aims to train simple BNNs. These two models are combined with a CP model into two hybrid methods `HW` and `HA` in order to obtain a feasible solution within a fixed time limit because otherwise, the MIP models fail shortly when the number of training data increases. We remark that two objectives, robustness and simplicity, are never optimized simultaneously. In [21], three MIP models are proposed that generalize the BNN approach to consider integer values for weights and biases. The first model, called Max-Correct, is based on the idea of maximizing the number of corrected predicted images; the second model, called Min-Hinge, is inspired by the squared hinge loss; the last model, called Sat-Margin, combines aspects of both the first two models. These three models always produce a feasible solution but use the margins only on the neurons of the last level, obtaining, hence, less robust BNNs.

Gradient-based BNN training. In [8], a gradient descent-based method is proposed, consisting of a local search that changes the weights to minimize a square hinge loss function. Note that a BNN trained with this approach only learns -1 and $+1$ weights. An extension of this method to admit zero-value weights, called (GD_t), is proposed in [22], to facilitate the comparison with their approach.

3   The BeMi ensemble

In this section, we present our structured ensemble of neural networks.

3.1   The BeMi structure

Suppose we have a classification problem with a set of labels (i.e., classes) being $\mathcal{I}$. If we define $\mathcal{P}(S)_m$ as the set of all the subsets of the set S that have cardinality m, our structured ensemble is constructed in the following way.

- · We set a parameter $1 < p \leq n = |\mathcal{I}|$.
- · We train a BNN denoted by $\mathcal{N}_{\mathcal{J}}$ for every $\mathcal{J} \in \mathcal{P}(\mathcal{I})_p$.
- · When testing a data point, we feed it to our list of trained BNNs, namely $(\mathcal{N}_{\mathcal{J}})_{\mathcal{J} \in \mathcal{P}(\mathcal{I})_p}$, and we obtain a list of predicted labels $(\mathfrak{e}_{\mathcal{J}})_{\mathcal{J} \in \mathcal{P}(\mathcal{I})_p}$. We denote the set $\mathcal{J} \setminus \mathfrak{e}_{\mathcal{J}}$ by $\hat{\mathfrak{e}}_{\mathcal{J}}$.
- · We then apply a majority voting system.

Note that we set $p > 1$, otherwise our structured ensemble would have been meaningless. Whenever $p = n$, our ensemble is made of one single BNN. When $p = 2$, we are using a Condorcet-like method, an election method already exploited in ML (e.g., see [2]).

The idea behind this structured ensemble is that, given an input $\boldsymbol{x}^k$ labelled $l \ (= y^k)$, the input is fed into $\binom{n}{p}$ networks where $\binom{n-1}{p-1}$ of them are trained to recognize an input with label l. If all of the networks correctly classify the input $\boldsymbol{x}^k$, then at most $\binom{n-1}{p-1} - \binom{n-2}{p-2}$ other networks can classify the input with a different label $\hat{l}$, and $\binom{n-1}{p-1} - \binom{n-2}{p-2} < \binom{n-1}{p-1}$ since $\binom{n-2}{p-2} \geq 1$. With this approach, if we plan to use $r \in \mathbb{N}$ inputs for each label, we are feeding our BNNs a total of $p \times r$ inputs instead of feeding $n \times r$ inputs to a single large BNN. When $p = 2 \ll n$, it is much easier to train our structured ensemble of BNNs rather than training one large BNN. The downside of this approach is the large number of networks that have to be trained, even if the training can run in parallel.

3.2   Majority voting system

After the training, we feed one input $\boldsymbol{x}^k$ to our list of BNNs, and we need to elaborate on the set of outputs.

Definition 1 (Dominant label). *For every $b \in \mathcal{I}$, we define*

$$C_b = \{\mathcal{J} \in \mathcal{P}(\mathcal{I})_p \mid \mathfrak{e}_{\mathcal{J}} = b\},$$

*and we say that a label b is a **dominant label** if $|C_b| \geq |C_l|$ for every $l \in \mathcal{I}$. We then define the set of dominant labels*

$$\mathcal{D} := \{b \in \mathcal{I} \mid b \text{ is a dominant label}\}.$$

Using this definition, we can have three possible outcomes.

(a) There exists a label $b \in \mathcal{I}$ such that $|C_b| > |C_l|$ for every $l \in \mathcal{I} \setminus \{b\}$ (there exists exactly one dominant label, that is $|\mathcal{D}| = |\{b\}| = 1$) $\implies$ our input is labelled as b.
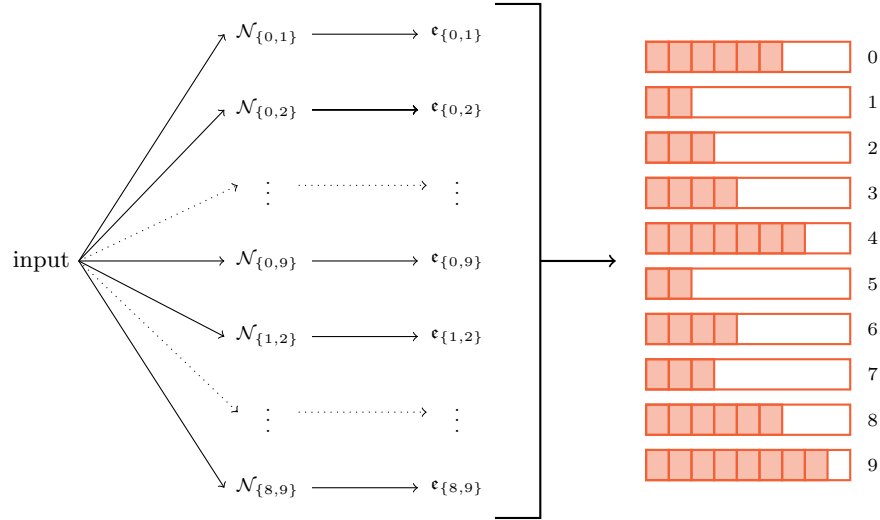
Fig. 1. Condorcet-inspired majority voting system.

(b) There exist $b_1, b_2, \ldots, b_p \in \mathcal{I}$, $b_i \neq b_j$ for all $i \neq j$ such that $|C_{b_1}| = |C_{b_2}| = \cdots = |C_{b_p}| > |C_l|$ for every $l \in \mathcal{I} \setminus \{b_1, b_2, \ldots, b_p\}$ (there exist exactly p dominant labels, that is $|\mathcal{D}| = |\{b_1, b_2, \ldots, b_p\}| = p$, so $\mathcal{D} \in \mathcal{P}(\mathcal{I})_p$) $\implies$ our input is labelled as $\mathfrak{c}_{\{b_1, b_2, \ldots, b_p\}} = \mathfrak{c}_{\mathcal{D}}$.

(c) There exist $\hat{p}$ dominant labels, $\hat{p} \neq 1, p$, i.e. $|\mathcal{D}| \neq 1, p \implies$ our input is labelled as $z \notin \mathcal{I}$.

While case (a) is straightforward, we have another possibility to label our input when we do not have a clear winner, that is, when we have trained a BNN on the set of labels that are the most frequent (i.e., case (b)).

Definition 2 (Label statuses). *In our labeling system, when testing an input seven different cases, herein called* **label statuses***, can show up.*

(s-0) *There is exactly one dominant label, the correct one.*

(s-1) *There exist exactly p dominant labels $b_1, b_2, \ldots, b_p$ and $\mathfrak{c}_{\{b_1, b_2, \ldots, b_p\}}$ is the correct one.*

(s-2) *There exist exactly p dominant labels $b_1, b_2, \ldots, b_p$ and the correct one belongs to the set $\hat{\mathfrak{c}}_{\{b_1, b_2, \ldots, b_p\}}$.*

(s-3) *There exist exactly $\hat{p} \neq 1, p$ dominant labels, and one of them is the correct one.*

(s-4) *There exist exactly $\hat{p} \neq 1, p$ dominant labels, but none are correct.*

(s-5) *There exist exactly p dominant labels $b_1, b_2, \ldots, b_p$ but none of them is the correct one;*

(s-6) *There exists exactly one dominant label, but it is not the correct one.*

Note that every input test will fall into one and only one label status.

Example 1. Let us take $\mathcal{I} = \{0, 1, \ldots, 9\}$ and $p = 2$. Note that, in this case, we have to train $\binom{10}{2} = 45$ networks and that $|C_b| \leq 9$ for all $b \in \mathcal{I}$. Hence, an input could be labelled as follows:

$$C_0 = \{\{0,1\}, \{0,2\}, \{0,3\}, \{0,5\}, \{0,7\}, \{0,8\}\}, \tag{2a}$$
$$C_1 = \{\{1,5\}, \{1,6\}\}, \tag{2b}$$
$$C_2 = \{\{1,2\}, \{2,5\}, \{2,8\}\}, \tag{2c}$$
$$C_3 = \{\{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}\}, \tag{2d}$$
$$C_4 = \{\{0,4\}, \{1,4\}, \{2,4\}, \{4,5\}, \{4,6\}, \{4,7\}, \{4,9\}\}, \tag{2e}$$
$$C_5 = \{\{5,6\}, \{5,7\}\}, \tag{2f}$$
$$C_6 = \{\{0,6\}, \{2,6\}, \{3,6\}, \{6,7\}\}, \tag{2g}$$
$$C_7 = \{\{1,7\}, \{2,7\}, \{3,7\}\}, \tag{2h}$$
$$C_8 = \{\{1,8\}, \{3,8\}, \{4,8\}, \{5,8\}, \{6,8\}, \{7,8\}\}, \tag{2i}$$
$$C_9 = \{\{0,9\}, \{1,9\}, \{2,9\}, \{3,9\}, \{5,9\}, \{6,9\}, \{7,9\}, \{8,9\}\}. \tag{2j}$$

Since $\mathcal{D} = \{9\}$, our input is labelled as 9, as shown in Figure 1. If 9 is the right label, we are in label status (s-0), if it is the wrong one, we are in label status (s-6). If instead $\{8,9\} \in C_8$, we were in the following situation

$$(2a)\text{--}(2h),$$
$$C_8 = \{\{1,8\}, \{3,8\}, \{4,8\}, \{5,8\}, \{6,8\}, \{7,8\}, \{8,9\}\}, \tag{2k}$$
$$C_9 = \{\{0,9\}, \{1,9\}, \{2,9\}, \{3,9\}, \{5,9\}, \{6,9\}, \{7,9\}\}, \tag{2l}$$

and then $|\mathcal{D}| = |\{4, 8, 9\}| = 3$, so that our input were labelled as -1. If the correct label is $4, 8$ or 9, we are in label status (s-3), else we are in label status (s-4). Lastly, if instead $\{3,9\} \in C_3$ like this

$$(2a)\text{--}(2c),\ (2e)\text{--}(2i),$$
$$C_3 = \{\{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{3,9\}\}, \tag{2m}$$
$$C_9 = \{\{0,9\}, \{1,9\}, \{2,9\}, \{5,9\}, \{6,9\}, \{7,9\}, \{8,9\}\}, \tag{2n}$$

then $|\mathcal{D}| = |\{4, 9\}| = 2 = p$ and since $\{4, 9\} \in C_4$ our input is labelled as 4. If 4 is the correct label, we are in label status (s-1), if 9 is the correct label, we are in label status (s-2), else we are in label status (s-5).

3.3 A multi-objective MIP model for training BNNs

In this subsection, we present how each of single small BNN is trained with a multi-objective MIP model. For ease of notation, we denote with $\mathcal{L} := \{1, \ldots, L\}$ the set of layers and with $\mathcal{L}_2 := \{2, \ldots, L\}$, $\mathcal{L}^{L-1} := \{1, \ldots, L-1\}$ two of its subsets. We also denote with $\mathfrak{b} := \max_{k \in T, j \in N_0} \{|x_j^k|\}$ a bound on the values of the training data.

Training a BNN with a multi-objective MIP model. A few MIP models are proposed in the literature to train BNNs efficiently. In this work, to train a single BNN, we use a lexicographic multi-objective function that results in the sequential solution of three different MIP models: the Sat-Margin (S-M) described in [21], the Max-Margin (M-M), and the Min-Weight (M-W), both described in [22]. The first model S-M maximizes the number of confidently correctly predicted data. The other two models, M-M and M-W, aim to train a BNN following two principles: robustness and simplicity. Our model is based on a lexicographic multi-objective function: first, we train a BNN with the model S-M, which is fast to solve and always gives a feasible solution. Second, we use this solution as a warm start for the M-M model, training the BNN only with the images that S-M correctly classified. Third, we fix the margins found with M-M, and minimize the number of active weights with M-W, finding the lightest BNN with the robustness found by M-M.

Problem variables. The critical part of our model is the formulation of the non-linear activation function (1). We use an integer variable $w_{ilj} \in \{-1, 0, +1\}$ to represent the weight of the connection between neuron $i \in N_{l-1}$ and neuron $j \in N_l$. Variable u_{lj}^k models the result of the indicator function $\mathbb{1}(p)$ that appears in the activation function $\rho(\cdot)$ for the training instance $\boldsymbol{x}^k$. The neuron activation is actually defined as $2u_{lj}^k - 1$. We introduce auxiliary variables c_{ilj}^k to represent the products $c_{ilj}^k = (2u_{lj}^k - 1)w_{ilj}$. Note that, while in the first layer, these variables share the same domain of the inputs, from the second layer on, they take values in $\{-1, 0, 1\}$. Finally, the auxiliary variables $\hat{y}^k$ represent a predicted label for the input $\boldsymbol{x}^k$, and variable q_j^k are used to take into account the data points correctly classified.

Sat-Margin (S-M) model. We first train our BNN using the following S-M model.

$$\max \quad \sum_{k \in T} \sum_{j \in N_L} q_j^k \tag{3a}$$

$$\text{s.t.} \quad q_j^k = 1 \implies \hat{y}_j^k \cdot y_j^k \geq \frac{1}{2} \qquad \forall j \in N_L, k \in T, \tag{3b}$$

$$q_j^k = 0 \implies \hat{y}_j^k \cdot y_j^k \leq \frac{1}{2} - \epsilon \qquad \forall j \in N_L, k \in T, \tag{3c}$$

$$\hat{y}_j^k = \frac{2}{N_{L-1} + 1} \sum_{i \in N_{L-1}} c_{iLj}^k \qquad \forall j \in N_L, k \in T, \tag{3d}$$

$$u_{lj}^k = 1 \implies \sum_{i \in N_{l-1}} c_{ilj}^k \geq 0 \qquad \forall l \in \mathcal{L}^{L-1}, j \in N_l, k \in T, \tag{3e}$$

$$u_{lj}^k = 0 \implies \sum_{i \in N_{l-1}} c_{ilj}^k \leq -\epsilon \qquad \forall l \in \mathcal{L}^{L-1}, j \in N_l, k \in T, \tag{3f}$$

$$c_{i1j}^k = x_i^k \cdot w_{i1j} \qquad \forall i \in N_0, j \in N_1, k \in T, \tag{3g}$$

$$c_{ilj}^k = (2u_{lj}^k - 1)w_{ilj} \qquad \forall l \in \mathcal{L}_2, i \in N_{l-1}, j \in N_l, k \in T, \tag{3h}$$

$$q_j^k \in \{0,1\} \qquad\qquad \forall j \in N_L, k \in T, \quad (3\text{i})$$

$$w_{ilj} \in \{-1,0,1\} \qquad\qquad \forall l \in \mathcal{L}, i \in N_{l-1}, j \in N_l, \quad (3\text{j})$$

$$u_{lj}^k \in \{0,1\} \qquad\qquad \forall l \in \mathcal{L}^{L-1}, j \in N_l, k \in T, \quad (3\text{k})$$

$$c_{i1j}^k \in [-\mathfrak{b}, \mathfrak{b}] \qquad\qquad \forall i \in N_0, j \in N_1, k \in T, \quad (3\text{l})$$

$$c_{ilj}^k \in \{-1,0,1\} \qquad\qquad \forall l \in \mathcal{L}_2, i \in N_{l-1}, j \in N_l, k \in T. \quad (3\text{m})$$

The objective function (3a) maximizes the number of data points that are correctly classified. The implication constraints (3b) and (3c) and constraints (3d) are used to link the output $\hat{y}_j^k$ with the corresponding variable q_j^k appearing in the objective function. The implication constraints (3e) and (3f) model the result of the indicator function for the k-th input data. The constraints (3g) and the bilinear constraints (3h) propagate the results of the activation functions within the neural network. We linearize all these constraints with standard big-M techniques.

The solution of model (3a)–(3m) gives us the solution vectors $\mathbf{c}_{\text{S-M}}$, $\mathbf{u}_{\text{S-M}}$, $\mathbf{w}_{\text{S-M}}$, $\hat{\mathbf{y}}_{\text{S-M}}$, $\mathbf{q}_{\text{S-M}}$. We then define the set

$$\hat{T} = \{k \in T \mid q_{j\,\text{S-M}}^k = 1, \ \forall j \in N_L\}, \qquad (4)$$

of confidently correctly predicted images. We use these images as input for the next Max-Margin M-M, and we use the vector of variables $\mathbf{c}_{\text{S-M}}$, $\mathbf{u}_{\text{S-M}}$, $\mathbf{w}_{\text{S-M}}$ to warm start the solution of M-M.

Max-Margin (M-M) model. The second level of our lexicographic multi-objective model maximizes the overall margins of every single neuron activation, with the ultimate goal of training a robust BNN. Starting from the model S-M, we introduce the margin variables m_{lj}, and we introduce the following Max-Margin model.

$$\max \quad \sum_{l \in \mathcal{L}} \sum_{j \in N_l} m_{lj} \qquad\qquad (5\text{a})$$

$$\text{s.t.} \quad (3\text{g})\text{–}(3\text{m}) \qquad\qquad \forall k \in \hat{T},$$

$$\sum_{i \in N_{L-1}} y_j^k c_{iLj}^k \geq m_{Lj} \qquad \forall j \in N_L, k \in \hat{T}, \quad (5\text{b})$$

$$u_{lj}^k = 1 \implies \sum_{i \in N_{l-1}} c_{ilj}^k \geq m_{lj} \qquad \forall l \in \mathcal{L}^{L-1}, j \in N_l, k \in \hat{T}, \quad (5\text{c})$$

$$u_{lj}^k = 0 \implies \sum_{i \in N_{l-1}} c_{ilj}^k \leq -m_{lj} \qquad \forall l \in \mathcal{L}^{L-1}, j \in N_l, k \in \hat{T}, \quad (5\text{d})$$

$$m_{lj} \geq \epsilon \qquad\qquad \forall l \in \mathcal{L}, j \in N_l. \quad (5\text{e})$$

Again, we can linearize constraints (5c) and (5d) with standard big-M constraints. This model gives us the solution vectors $\mathbf{c}_{\text{M-M}}$, $\mathbf{u}_{\text{M-M}}$, $\mathbf{w}_{\text{M-M}}$, $\mathbf{m}_{\text{M-M}}$. We then evaluate $\mathbf{v}_{\text{M-M}}$ as

$$v_{ilj\,\text{M-M}} = |w_{ilj\,\text{M-M}}| \qquad \forall l \in \mathcal{L}, i \in N_{l-1}, j \in N_l. \qquad (6)$$

Min-Weight (M-W) model. The third level of our multi-objective function minimizes the overall number of non-zero weights, that is, the connection of the trained BNN. We introduce the new auxiliary binary variable v_{ilj} to model the absolute value of the weight w_{ilj}. Starting from the solution of model `M-M`, we fix $\hat{\mathbf{m}} = \mathbf{m}_{\text{M-M}}$, and we pass the solution $\mathbf{c}_{\text{M-M}}, \mathbf{u}_{\text{M-M}}, \mathbf{w}_{\text{M-M}}, \mathbf{v}_{\text{M-M}}$ as a warm start to the following `M-W` model:

$$\min \quad \sum_{l \in \mathcal{L}} \sum_{i \in N_{l-1}} \sum_{j \in N_l} v_{ilj} \tag{7a}$$

$$\text{s.t.} \quad \text{(3g)–(3m)} \qquad\qquad\qquad\qquad\qquad \forall k \in \hat{T},$$

$$\sum_{i \in N_{L-1}} y_j^k c_{iLj}^k \geq \hat{m}_{Lj} \qquad\qquad \forall j \in N_L, k \in \hat{T}, \tag{7b}$$

$$u_{lj}^k = 1 \implies \sum_{i \in N_{l-1}} c_{ilj}^k \geq \hat{m}_{lj} \qquad \forall l \in \mathcal{L}^{L-1}, j \in N_l, k \in \hat{T}, \tag{7c}$$

$$u_{lj}^k = 0 \implies \sum_{i \in N_{l-1}} c_{ilj}^k \leq -\epsilon - \hat{m}_{lj} \quad \forall l \in \mathcal{L}^{L-1}, j \in N_l, k \in \hat{T}, \tag{7d}$$

$$- v_{ilj} \leq w_{ilj} \leq v_{ilj} \qquad\qquad \forall l \in \mathcal{L}, i \in N_{l-1}, j \in N_l, \tag{7e}$$

$$v_{ilj} \in \{0, 1\} \qquad\qquad\qquad \forall l \in \mathcal{L}, i \in N_{l-1}, j \in N_l. \tag{7f}$$

Note that whenever v_{ilj} is equal to zero, the corresponding weight w_{ilj} is set to zero due to constraint (7e), and, hence, the corresponding link can be removed from the network.

Lexicographic multi-objective. By solving the three models `S-M`, `M-M`, and `M-W`, sequentially, we first maximize the number of input data that is correctly classified, then we maximize the margin of every activation function, and finally, we minimize the number of non-zero weights. The solution of the decision variables w_{ilj} of the last model `M-W` defines our classification function $f_W : \mathbb{R}^{n_0} \to \{\pm 1\}^{n_L}$.

In the next section, we compare the results of our model with the other approaches from the literature.

4   Computational results

We run three types of experiments to address the following questions

- **Experiment 1:** How does our approach compare with the previous state-of-the-art MIP models for training BNNs in the context of few-shot learning?
- **Experiment 2:** How does the BEMI ensemble scale with the size of the input images, considering two different types of BNNs?
- **Experiment 3:** How does the proposed approach perform on another dataset, comparing the running time, the average gap to the optimal training MIP model, and the percentage of links removed?

The three classes of experiments are detailed in the next subsections.
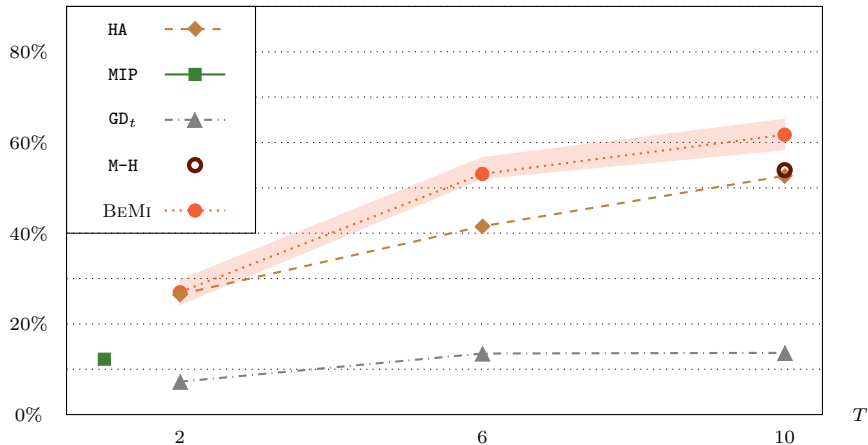
Fig. 2. Comparison of published approaches vs BeMi, in terms of accuracy over the MNIST dataset using few-shot learning with 2, 6, and 10 images per digit.

Datasets. The experiments are performed on the standard MNIST [4] and Fashion-MNIST [24] datasets. In particular, we import the datasets from Keras [3]. We test our results on 500 images for each class. For each experiment, we report the average over three different samples of images.

Implementation details. We use Gurobi version 9.5.1 [7] to solve our MIP models. The parameters of Gurobi are left to the default values. All the MIP experiments were run on an HPC cluster running CentOS but using a single node per experiment. Each node has an Intel CPU with 8 physical cores working at 2.1 GHz and 16 GB of RAM. In all of our experiments, we fix the value $\epsilon = 0.1$. The source code will be available on GitHub in case of acceptance of this paper.

4.1   Experiment 1

The first set of experiments aims to compare the BeMi ensemble with the following state-of-the-art methods: the hybrid CP and MIP model based on Max-Margin optimization (`HA`) [22]; the gradient-based method $\mathtt{GD}_t$ introduced in [8] and adapted in [22] to deal with link removal; and the Min-hinge (`M-H`) model proposed in [21].

   For the comparison, we fix the setting of [22], which takes from the MNIST up to 10 images for each class, for a total of 100 training data points, and which uses a time limit of 7200 seconds to solve their MIP training models. In our experiments, we train the BeMi ensemble with 2, 6, and 10 samples for each digit. Since our ensemble has 45 BNNs, we leave for the training of each single BNN a maximum of 160 seconds (since $160 \times 45 = 7200$). In particular, we give a 75 seconds time limit to the solution of `S-M`, 75 seconds to `M-M`, and 10 seconds to `M-W`. In all of our experiments, whenever the optimum is reached within the time

limit, the remaining time is added to the time limit of the subsequent model. We remark that our networks could be trained in parallel, which would highly reduce the wall-clock runtime. For the sake the completeness, we note that we are using $45 \times (784 \times 4 + 4 \times 4 + 4 \times 1) = 142\,020$ parameters (all the weights of all the 45 BNNs) instead of the $784 \times 16 + 16 \times 16 + 16 \times 10 = 12\,960$ parameters used in [22] for a single large BNN. Note that, in this case, the dimension of the parameter space is $3^{12\,960} (\cong 10^{6183})$, while, in our case, it is $45 \times 3^{3156} (\cong 10^{1507})$.

Figure 2 compares the results of our BeMi ensemble with four other methods: the hybrid CP-MIP approach HA [22]; the pure MIP model in [22], which can handle a single image per class; the gradient-based method GD_t, which is the version of [8] modified by [22]; the minimum hinge model M-H presented in [21], which report results only for 10 digits per class. We report the best results reported in the original papers for these four methods. The BeMi ensemble obtains an average accuracy of 61%, outperforms all other approaches when 6 or 10 digits per class are used, and it is comparable with the hybrid CP-MIP method when only 2 digits per class are used. When 10 digits per class are available, the second best option is the min hinge model M-H proposed in [21], with an average accuracy of 51.1%.

4.2   Experiment 2

This second set of experiments studies how our approach scales with the number of data points (i.e., images) per class, and how it is affected by the architecture of the small BNNs within the BeMi ensemble. For the number of data points per class we use $10, 20, 30, 40$ training images per digit. We use the layers $\mathcal{N}_a = [784, 4, 4, 1]$ and $\mathcal{N}_b = [784, 10, 3, 1]$ for the two architectures, Herein, we refer to Experiments $2a$ and $2b$ as the two subsets of experiments related to the architectures $\mathcal{N}_a$ and $\mathcal{N}_b$. In both cases, we train each of our 45 BNNs with a time limit of 290s for model S-M, 290s for M-M, and 20s for M-W, for a total of 600s (i.e., 10 minutes for each BNN).

Figure 3 shows the results for Experiments $2a$ and $2b$: the dotted and dashed lines refer to the two average accuracies of the two architectures, while the colored areas include all the accuracy values obtained as the training instances vary. While the two architectures behave similarly, the best average accuracy exceeds 75% and it is obtained with the first architecture $\mathcal{N}_a$.

Table 1 reports the detailed results for the BeMi ensemble using 5000 testing data points, where we distinguish among images classified as correct, wrong, or unclassified. These three conditions refer to different label statuses specified in Definition 2. The correct labels are the sum of the statuses s-0 and s-1; the wrong labels of statuses s-2, s-5, and s-6; the unclassified labels (n.l.) of s-3 and s-4.

Figure 4 shows the confusion matrix obtained with the BeMi ensemble trained with BBNs with the architecture $\mathcal{N}_b = [784, 10, 3, 1]$, using 40 training images per class from the MNIST dataset. Clearly, the main diagonal shows how most of the testing images were classified correctly. However, a few aspects remain to be investigated, for instance, the frequent misclassification between pairs such as $(4, 9)$ or $(3, 5)$.
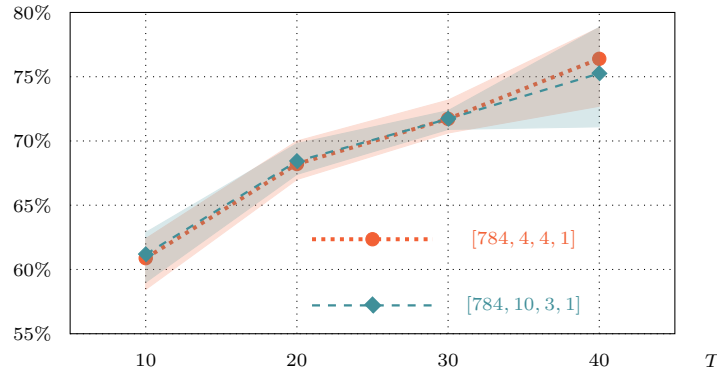
Fig. 3. Comparing the BeMi ensemble accuracy using two architectures, $\mathcal{N}_a = [784, 4, 1]$ and $\mathcal{N}_b = [784, 10, 3, 1]$, and increasing the number of MNIST images per class.

Table 1. Percentages of MNIST images classified as correct, wrong, or unclassified ($n.l.$), and of label statuses from s-0 to s-6, for the architecture $\mathcal{N}_a = [784, 4, 4, 1]$.

| Images per class	Classification			Label status						
	correct	wrong	$n.l.$	s-0	s-1	s-2	s-3	s-4	s-5	s-6
10	61.80	36.22	1.98	58.30	3.50	1.84	1.30	0.68	4.74	29.64
20	69.96	27.60	2.44	66.68	3.28	2.12	2.18	0.26	3.04	22.44
30	73.18	24.56	2.26	70.14	3.04	1.88	1.88	0.38	2.68	20.00
40	78.82	19.30	1.88	75.56	3.26	1.90	1.72	0.16	1.70	15.70

4.3 Experiment 3

In the third experiment, we replicate Experiments 2a and 2b with the two architectures $\mathcal{N}_a$ and $\mathcal{N}_b$, using the Fashion-MNIST dataset, which is notably more challenging than the *easy* MNIST. Figure 5 shows the results of Experiments 3a and 3b. As in Figure 2, the dotted and dashed lines represent the average percentages of correctly classified images, while the colored areas include all accuracy values obtained as the instances vary. The first architecture is comparable with the second, with up to 30 training images per digit, while it is significantly better with 40 images. For the Fashion-MNIST, the best average accuracy exceeds 70%.

Table 2 reports detailed aggregate results for all Experiments 2 and 3. The first two columns give the dataset and the architecture, and the third column specifies the number of images per digit used during training. The 4-th column reports the runtime for solving model S-M. Note that the time limit is 290 seconds; hence, we solve exactly the first model, consistently achieving a training accuracy of 100%. The remaining four columns give: *Gap (%)* refers to the mean and maximum percentage gap at the last MIP model (M-W) of our lexicographic multi-objective model, as reported by the Gurobi MIPgap attribute; *Links (%)*
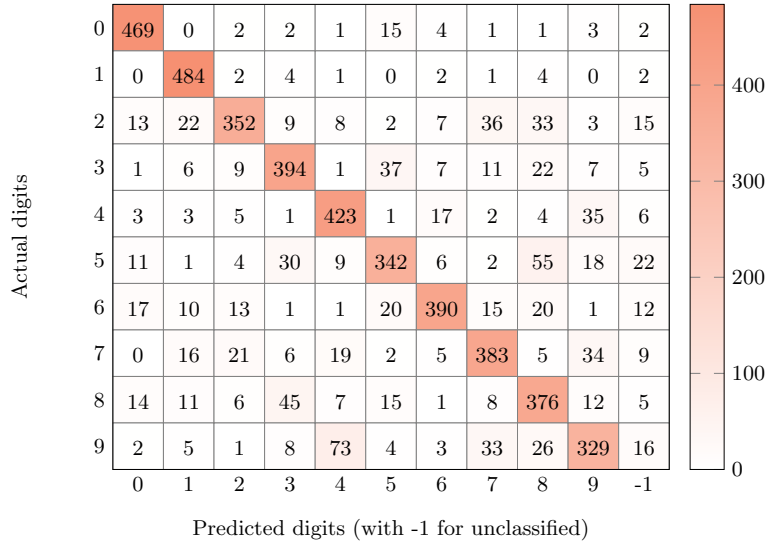
Fig. 4. Extended confusion matrix obtained after training the BᴇMɪ ensemble by single BNNs with architecture $\mathcal{N}_b = [784, 10, 3, 1]$, using 40 MNIST digits per class. The last right column with x-label equal to -1 refers to unclassified images.
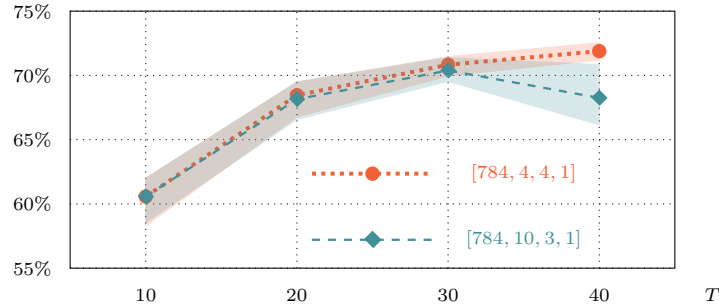
Fig. 5. Average accuracy for the BᴇMɪ ensemble tested two architectures, namely $\mathcal{N}_a = [784, 4, 4, 1]$ and $\mathcal{N}_b = [784, 10, 3, 1]$, using Fashion-MNIST.

is the percentage of non-zero weights after the solution of the second model `M-M`, and after the solution of the last model `M-W`. The results show that the runtime increase with the size of the input set (fourth and fifth columns), as well as the average percentage gap. However, for the percentage of removed links, there is a significant difference between the two datasets: for MNIST, our third model `M-W` removes around 70% of the links, while for the Fashion-MNIST, it removes around 50% of the links. Note that in both cases, these significant reductions show how our model is also optimizing the BNN architecture.

Table 2. Aggregate results for Experiments 2 and 3: the 4-th column reports the runtime to solve the first model `S-M`; *Gap (%)* refers to the mean and maximum percentage gap at the last MIP model `M-W`; *Links (%)* is the percentage of non-zero weights after the solution of models `M-M` and `M-W`.

Dataset	Layers	Images per class	Model `S-M` time (s)	Gap (%) mean	max	Links (%) (M-M)	(M-W)
MNIST	784,4,4,1	10	2.99	17.37	28.25	49.25	27.14
		20	5.90	19.74	24.06	52.95	30.84
		30	10.65	20.07	26.42	56.90	30.88
		40	15.92	18.50	23.89	58.70	29.42
	784,10,3,1	10	6.88	6.28	9.67	49.46	23.96
		20	17.02	7.05	8.42	53.25	26.65
		30	25.84	7.38	15.88	57.21	25.02
		40	44.20	9.90	74.16	59.08	24.22
F-MNIST	784,4,4,1	10	7.66	17.21	25.92	86.38	56.54
		20	14.60	22.35	28.00	93.18	57.54
		30	26.10	19.78	29.53	92.56	58.78
		40	39.90	22.71	75.03	93.13	64.61
	784,10,3,1	10	13.83	6.14	8.98	86.65	53.72
		20	26.80	7.84	9.59	93.57	51.03
		30	38.48	7.18	16.09	92.90	52.50
		40	64.52	12.10	55.19	93.57	55.67

5   Conclusions

In this work, we have introduced the BeMi ensemble, a structured architecture of BNNs for classification tasks. Each network specializes in distinguishing between pairs of classes and combines different approaches already existing in the literature to preserve feasibility while being robust and lightweight. These features are critical to enabling neural networks to run on low-power devices. The output of the BeMi ensemble is chosen by a majority voting system inspired by the Condorcet method. Notice that the BeMi ensemble is a general architecture that could be employed using other types of neural networks. For instance, we can use the same architecture but replace each single BNN with a more general Integer-valued NN as in [21].

A current limitation of our approach is the strong dependence on the randomly sampled images used for training. In future work, we plan to improve the training data selection by using a k-medoids approach. For instance, let j be the number of digits per class to train a BNN. We could divide all images of the same class into j disjoint non-empty subsets and consider their centroids as training data. This approach should mitigate the dependency on the sampled training data points.

References

1. Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., Vielma, J.P.: Strong mixed-integer programming formulations for trained neural networks. Mathematical Programming **183**(1), 3–39 (2020)
2. Avand, M., Khiavi, A.N., Khazaei, M., Tiefenbacher, J.P.: Determination of flood probability and prioritization of sub-watersheds: A comparison of game theory to machine learning. Journal of Environmental Management **295**, 113040 (2021)
3. Chollet, F., et al.: Keras (2015), `https://github.com/fchollet/keras`
4. Deng, L.: The MNIST database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine **29**(6), 141–142 (2012)
5. Domingos, P.: A few useful things to know about machine learning. Communications of the ACM **55**(10), 78–87 (2012)
6. Fischetti, M., Jo, J.: Deep neural networks and mixed integer linear optimization. Constraints **23**(3), 296–309 (2018)
7. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2022), `https://www.gurobi.com`
8. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. Advances in Neural Information Processing Systems (NeurIPS) **29**, 4107–4115 (2016)
9. Jiang, Y., Krishnan, D., Mobahi, H., Bengio, S.: Predicting the generalization gap in deep networks with margin distributions. In: International Conference on Learning Representations (ICLR) (2019)
10. Kawaguchi, K., Kaelbling, L.P., Bengio, Y.: Generalization in deep learning. arXiv:1710.05468 (2017)
11. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: International Conference on Learning Representations (ICLR). vol. 5 (2017)
12. Khalil, E.B., Gupta, A., Dilkina, B.: Combinatorial attacks on binarized neural networks. In: International Conference on Learning Representations (ICLR) (2019)
13. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
14. Lin, X., Zhao, C., Pan, W.: Towards accurate binary convolutional neural network. Advances in Neural Information Processing Systems (NeurIPS) **30** (2017)
15. Moody, J.: The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. Advances in Neural Information Processing Systems (NeurIPS) **4**, 847–854 (1991)
16. Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N.: Exploring generalization in deep learning. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 30, pp. 5947–5956 (2017)
17. Sakr, C., Choi, J., Wang, Z., Gopalakrishnan, K., Shanbhag, N.: True gradient-based training of deep binary activated neural networks via continuous binarization. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2346–2350. IEEE (2018)
18. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural networks **61**, 85–117 (2015)
19. Serra, T., Kumar, A., Ramalingam, S.: Lossless compression of deep neural networks. In: Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPIAOR). vol. 17, pp. 417–430 (2020)

20. Tang, W., Hua, G., Wang, L.: How to train a compact binary neural network with high accuracy? In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
21. Thorbjarnarson, T., Yorke-Smith, N.: On training neural networks with mixed integer programming. In: IJCAI-PRICAI'20 Workshop on Data Science Meets Optimisation (2021)
22. Toro Icarte, R., Illanes, L., Castro, M.P., Cire, A.A., McIlraith, S.A., Beck, J.C.: Training binarized neural networks using MIP and CP. In: International Conference on Principles and Practice of Constraint Programming. vol. 11802, pp. 401–417. Springer (2019)
23. Vanschoren, J.: Meta-learning. In: Automated machine learning, pp. 35–61. Springer, Cham (2019)
24. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747 (2017)
25. Young, H.P.: Condorcet's theory of voting. American Political science review **82**(4), 1231–1244 (1988)
26. Yu, X., Serra, T., Ramalingam, S., Zhe, S.: The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks. In: International Conference on Machine Learning (ICML). pp. 25668–25683 (2022)