Robust Model Compression Using Deep Hypotheses

Omri Armstrong, Ran Gilad-Bachrach
Tel Aviv University
Ramat Aviv 699780, Tel Aviv
armstrong@mail.tau.ac.il, rgb@tauex.tau.ac.il

Abstract

Machine Learning models should ideally be compact and robust. Compactness provides efficiency and comprehensibility whereas robustness provides resilience. Both topics have been studied in recent years but in isolation. Here we present a robust model compression scheme which is independent of model types: it can compress ensembles, neural networks and other types of models into diverse types of small models. The main building block is the notion of depth derived from robust statistics. Originally, depth was introduced as a measure of the centrality of a point in a sample such that the median is the deepest point. This concept was extended to classification functions which makes it possible to define the depth of a hypothesis and the median hypothesis. Algorithms have been suggested to approximate the median but they have been limited to binary classification. In this study, we present a new algorithm, the *Multiclass Empirical Median Optimization* (MEMO) algorithm that finds a deep hypothesis in multi-class tasks, and prove its correctness. This leads to our *Compact Robust Estimated Median Belief Optimization* (CREMBO) algorithm for robust model compression. We demonstrate the success of this algorithm empirically by compressing neural networks and random forests into small decision trees, which are interpretable models, and show that they are more accurate and robust than other comparable methods. In addition, our empirical study shows that our method outperforms Knowledge Distillation on DNN to DNN compression.

1 Introduction

Large models, such as Deep Neural Networks (DNNs) and ensembles achieve high accuracy on diverse problems (Caruana, Karampatziakis, and Yessenalina 2008; Usmanim 2018). However, their size presents a challenge in many cases because of their resource requirements and their incomprehensibility (Lage et al. 2019). The lack of interpretability of large machine learning models is a limitation especially for applications that require critical decision making such as medical diagnostics (Kononenko 2001; Caruana et al. 2015) and hiring decisions (Hamilton 2018). Small models are efficient in terms of computational cost and memory footprint while also being more interpretable. When small models are required, it has been shown that compressing a large model often outperform models that were trained small from the get-go (Buciluǎ, Caruana, and Niculescu-Mizil 2006).

In model compression (Buciluǎ, Caruana, and Niculescu-Mizil 2006), a large model M is first trained and then compressed into a smaller model m. Most compression schemes are designed for specific classes of functions (Cheng et al. 2017). DNN compression schemes include parameter pruning and quantization (Gong et al. 2014; Srinivas and Babu 2015), low rank factorization and sparsity (Rigamonti et al. 2013; Denil et al. 2013), and Knowledge Distillation (KD) (Hinton, Vinyals, and Dean 2015) where temperature is used on M's predictions to create 'soft' predictions on which m is trained on. There are also schemes that convert one class of functions to another class, for example, DNNs to Soft Decision Trees (SDT) (Frosst and Hinton 2017) or to gradient boosted trees (Che et al. 2016), and trees to DNNs (Banerjee 1997).

In this work, we present a new compression scheme that can compress almost any type of Machine Learning (ML) model to almost any type of smaller model. To be able to work with diverse learning models, we use the large model M as an oracle to train a small model m. However, in the compression step we avoid using common training techniques that minimize a loss function over the training data generated by the oracle (Buciluǎ, Caruana, and Niculescu-Mizil 2006) since such processes are sensitive to perturbations (Gilad-Bachrach and Burges 2013) and thus are not robust. Instead, we use M to generate a belief, which is the conditional probability $p(Y = y|X = x)$ and use methods based on maximizing predicate depth (Gilad-Bachrach and Burges 2013).

In our context a belief is a distribution $p(Y = y|X = x)$ where y is one of the possible classes and x is a record.[1] Intuitively, a model m has a predicate depth d if for every (or most) points x it holds that $p\left(m(x)|x\right) \geq d$. When d is large, the model is robust to slight changes in the prior belief p (Gilad-Bachrach and Burges 2013). Therefore, in our method, we extract a belief from large model M and train a small model m by finding a model with a large predicate depth from a class of small models. Following (Gilad-

[1] We sometimes use the shorthand notation $p(y|x)$ to denote $p(Y = y|X = x)$.

Bachrach and Burges 2013), we call the model with the largest predicate depth, the median.

The median has robustness properties by design and compactness is achieved by restricting the search to classes of small models. An algorithm for approximating the median hypothesis was introduced in (Gilad-Bachrach and Burges 2013) but it is limited to binary classification. To implement our procedure, we first present the Multiclass Empirical Median Optimization (MEMO) algorithm for finding the deepest model out of a function class $\mathcal{F}$ of multi-class classifiers. Second, we present the Compact Robust Estimated Median Belief Optimization (CREMBO) algorithm for model compression by finding a deep model when $\mathcal{F}$ is a class of compact models.

After deriving the algorithms and proving their properties we present an empirical evaluation of these methods and compare them to existing methods. We first demonstrate the ability of our method to compress both Random Forests (RF) (Ho 1995) and DNNs to small decision trees (Quinlan 1986) since they are compact and interpretable (Lage et al. 2019; Bénard et al. 2020). Then we show that for DNN to DNN compression CREMBO outperforms the commonly used KD compression scheme on several model architectures. Our empirical study shows that CREMBO generates models that are more accurate and more robust than any comparable method.

To the best of our knowledge, this is the first work to study robustness of model compression methods. Our main contributions presented in this paper are: The novel CREMBO algorithm for robust model compression. The CREMBO algorithm is the first to use the concept of deep hypotheses for compression, it is a flexible algorithm, allowing for compression of diverse model types. Our empirical evaluation shows that it succeeds in creating compact models that are more robust and more accurate than any other comparable method. In addition, we present the MEMO algorithm which extends the ability to find the median hypothesis to multi-class classifiers. Our code is available at *https://github.com/TAU-MLwell/Rubust-Model-Compression*.

The rest of the paper is organized as follows: We present the predicate depth and preliminaries in Section 2. In Section 3 we detail the MEMO algorithm with proofs. In Section 4 we describe the CREMBO algorithm and in Section 5 we report our experiments and results. We conclude the paper with a discussion of the results.

2 Background and Notations

Tukey (1975) presented depth as a centrality measure of a point in a sample. The Predicate depth is an extension of the Tukey depth that operates on the space of classification functions. The predicate depth, as defined for binary classification tasks (Gilad-Bachrach and Burges 2013), measures the agreement of a function f with the majority vote on x. A deep function will always have a large agreement with its prediction among the class $\mathcal{F}$. The median hypothesis is defined as the deepest possible function.

Definition 1. *(Gilad-Bachrach and Burges 2013) Let $\mathcal{F}$ be a function class and let Q be a probability measure over $\mathcal{F}$. The predicate depth of f on the instance $x \in X$ with respect to Q is defined as*

$$D_Q(f \mid x) = P_{g\sim Q}[g(x) = f(x)]$$

The predicate depth of f with respect to Q is defined as

$$D_Q(f) = \inf_{x\in X} D_Q(f \mid x)$$

A common measure of stability is the breakdown point (Hampel 1971). The breakdown point measures how much Q must change in order to produce an arbitrary value of the statistic. The rationale of using deep hypotheses to achieve robustness derives from a result presented in (Gilad-Bachrach and Burges 2013), showing that the breakdown point of the median hypothesis is proportional to its depth while the breakdown point of hypotheses acquired in the standard procedure of minimizing some loss functions (MAP hypothesis) is in fact zero. Another advantage of using deep hypotheses is that deeper hypotheses have better bounds on their generalization error.

In practice it might be infeasible to calculate the depth function and find the median hypothesis. However, it can be approximated using the empirical depth function:

Definition 2. *(Gilad-Bachrach and Burges 2013) Given a sample $S = \{x_1, ..., x_m\}$ s.t. $x_i \in X$, a sample $T = \{f_1, ..., f_n\}$ s.t. $f_j \in \mathcal{F}$ and a function f. The empirical depth on instance $x_i \in X$ with respect to T is defined as*

$$\hat{D}_T(f|x_i) = \frac{1}{n}\sum_j 1_{f_j(x_i)=f(x_i)}$$

The empirical depth with respect to T is defined as

$$\hat{D}^S_T(f) = \min_i \hat{D}_T(f|x_i)$$

The empirical depth, as introduced in Definition 2, uses a sample of records S and a sample of hypotheses T to get an empirical estimate of the agreement between members of the hypothesis class. However, generating the sample T is a challenging task; for example, if the hypothesis class is the class of DNNs, many of them need to be trained to be able to estimate the empirical depth. Note, however, that T is only used for estimating the probability $p(y|x)$ for a given record x and a class y and therefore it is sufficient to assume that there is an oracle $\mathcal{O}(x, y)$ that given a point x and a class y returns the fraction of the hypotheses that predict label y for point x. This allows us to redefine the empirical depth as follows:

$$\hat{D}_{\mathcal{O}}(f|x_i) = \mathcal{O}(x_i, f(x_i)) \tag{1}$$

$$\hat{D}^S_{\mathcal{O}}(f) = \min_i \hat{D}_{\mathcal{O}}(f|x_i) \tag{2}$$

The previous definitions of $\hat{D}_T(f|x_i), \hat{D}^S_T(f)$ presented in Definition 2 are a special case of these definitions in which the oracle is $\mathcal{O}(x, y) = \frac{1}{n}\sum_j 1_{f_j(x)=y}$ where $f_j \in T$.

3 Multi Class Empirical Median Optimization

We now present the Multiclass Empirical Median Optimization (MEMO) algorithm. MEMO finds a function f that maximizes the empirical depth, that is $f = \arg\max_{f \in \mathcal{F}} \hat{D}^S_{\mathcal{O}}(f)$ for multi-class classifiers. As mentioned in Section 2, a deep function will have an agreement with a large fraction of the hypotheses (or posterior belief) on its predictions. Another way to look at it is to say that whenever it makes a prediction, it avoids predictions which are in small minorities according to the belief $p(y|x)$.

Note first that for a given point x, the depth $\hat{D}_{\mathcal{O}}(f|x)$ takes its values in the set $\{\mathcal{O}(x,y) : y \in Y\}$ where Y is the set of classes (possible labels). Hence, for any given record x the depth takes values in a set of size at most $|Y|$. Therefore, given a sample S the set of depth values is:

$$\left\{\hat{D}^S_{\mathcal{O}}(f) : f \in \mathcal{F}\right\} \subseteq \{\mathcal{O}(x,y) : x \in S,\ \ y \in Y\} \quad .$$

Hence, its size is at most $|S||Y|$. Therefore, finding the deepest function $f \in \mathcal{F}$ can be completed by searching for the largest value $d \in \{\mathcal{O}(x,y) : x \in S,\ \ y \in Y\}$ for which there exists $f \in \mathcal{F}$ such that $\hat{D}^S_{\mathcal{O}}(f) \geq d$. Assuming that we know how to verify whether there exists a function f with a depth of at least d, the deepest function can be found by using binary search, in $\log|S| + \log|Y|$ steps.

The remaining challenge for finding a deep hypothesis in the multi-class case is designing the procedure where given a sample S and a desired depth d returns $f \in \mathcal{F}$ such that $\hat{D}^S_{\mathcal{O}}(f) \geq d$ if one exists and returns *"fail"* otherwise. Let $Y_i \subseteq Y$ be the set of classes for which $\mathcal{O}(x_i, y) \geq d,\ \ x_i \in S$. In Theorem 1 we show that $\hat{D}^S_{\mathcal{O}}(f) \geq d$ if, and only if, $f(x_i) \in Y_i$ for every $x_i \in S$. Therefore, the procedure we are looking for is a learning algorithm that receives a sample $S = \{x_i, Y_i\}_{i=1}^m$ and learns a function f such that $\forall i\ f(x_i) \in Y_i$. This learning problem is different from the standard classification problem, it can be implemented as a multi-label learning problem at the training phase whereas on inference, only the class with the highest probability is selected. Modifying the learning algorithms for decision trees or DNNs to support this is relatively easy.

These observations allow us to introduce the *Multiclass Empirical Median Optimization* (MEMO) algorithm (Algorithm 1) and to prove its correctness in Theorem 1. In terms of performance, the number of iterations required by the MEMO algorithm for the binary search is

$$\log\left(|\{\mathcal{O}(x_i, y) : x_i \in S, y \in Y\}|\right) \tag{3}$$

The size of the set in (3) is bounded by the number of unique values that the oracle can return, which can be very small. Consider, for example, the case of compressing a Random Forest. One natural way to implement the oracle $\mathcal{O}$ is to say that $\mathcal{O}(x,y)$ is the fraction of the trees in the forest which predict that the class is y for some $x \in S$. In this case, the number of unique values that $\mathcal{O}$ can return is at most $|M|+1$ where $|M|$ is the number of trees in the forest. Therefore, the number of iterations required when using MEMO to compress a Random Forest is $\log(|M|+1)$. When compressing models such as DNNs, the large model M returns a score for each class that can be converted into probabilities using softmax which the oracle can use as its return value. In this case, the number of unique values is bounded by the fidelity in which the values are encoded. If b bits are used to describe the scores, there would be at most 2^b unique values that the oracle can return and the number of iteration would be bounded by b. Therefore, even if a DNN is used as large model, and 32 bits numbers are used to represent its outputs, the number of iterations required by the MEMO algorithm will be ≤ 32.

Algorithm 1: Multiclass Empirical Median Optimization (MEMO) Algorithm

Input:

- A sample $S \in X^m$
- An oracle $\mathcal{O}(x,y)$
- A learning algorithm $\mathcal{L}$ which given a sample of the form $\hat{S} = \{(x_i, Y_i)\}_{i=1}^m$ where $Y_i \subseteq Y$ returns a function $f \in \mathcal{F}$ consistent with it if such a function exists and *"fail"* otherwise.

Output: A function $f \in \mathcal{F}$ and the depth $\hat{D}^S_{\mathcal{O}}(f)$

```
begin
    Let Θ ← {d_1 < d_2 < ... < d_m} = sort({O(x_i, y) : x_i ∈ S, y ∈ Y})
    Run binary search over Θ unique values to find the largest threshold d for which the following procedure does not fail:
    begin
        for i=1,...,m do
            Y_i = {y ∈ Y s.t. O(x_i, y) ≥ d}
        Let Ŝ ← {(x_i, Y_i)}_{i=1}^m
        Let f ← L(Ŝ)
        Return f
    Return f, d
```

Theorem 1. *If $\mathcal{F} \neq \emptyset$ then the MEMO algorithm will return a function f^* and a depth d^* such that*

$$d^* = \hat{D}^S_{\mathcal{O}}(f^*) = \max_{f \in \mathcal{F}} \hat{D}^S_{\mathcal{O}}(f)$$

Proof. Recall that $\hat{D}^S_{\mathcal{O}}(f) = \min_{x \in S}\left(\mathcal{O}(x, f(x))\right)$ and therefore, for every $f \in \mathcal{F}$, $\hat{D}^S_{\mathcal{O}}(f)$ is in the set of thresholds Θ defined in Algorithm 1. Furthermore, since d_1 is the minimal possible threshold then $\forall f \in \mathcal{F}, \hat{D}^S_{\mathcal{O}}(f) \geq d_1$. Therefore, the binary search will always return some function f for some threshold d.

Assume that $\hat{S}$ was generated with threshold d. If there exists $f \in \mathcal{F}$ such that $\hat{D}^S_{\mathcal{O}}(f) \geq d$ then for every $x \in S$, $\mathcal{O}(x, f(x)) \geq d$ and therefore $\mathcal{L}$ will not fail. However, if $d > \max_f \hat{D}^S_{\mathcal{O}}(f)$, there is no $f \in \mathcal{F}$ s.t. $\forall x \in S, f(x) \in \{y \in Y\ s.t.\ \mathcal{O}(x,y) \geq d\}$ and $\mathcal{L}$ will fail. Therefore, the binary search will always terminate when finding the maximal

d for which there exists $f \in \mathcal{F}$ with $\hat{D}^S_{\mathcal{O}}(f) \geq d$ and since this is the maximal value with this property, it has to be that $d^* = \max_f \hat{D}^S_{\mathcal{O}}(f)$.

To see that $d^* = \hat{D}^S_{\mathcal{O}}(f^*)$ recall that from the definition of $\mathcal{L}$ it follows that $\forall x \in S, f^*(x) \in \{y \in Y \; s.t. \; \mathcal{O}(x,y) \geq d^*\}$ and therefore $d^* \leq \hat{D}^S_{\mathcal{O}}(f^*)$ but from the maximal property of d^* we also know that $d^* \geq \hat{D}^S_{\mathcal{O}}(f^*)$ which completes the proof. □

To prove the robustness of MEMO we use the breakdown point as a measure of robustness (Hampel 1971). We adjust the definition to our setting in the following way:

Definition 3. *The breakdown point of a compression algorithm $\mathcal{C}$ with the oracle $\mathcal{O}$ and a sample S is*

$$breakdown(\mathcal{C}, \mathcal{O}, S) = \max_{f \in \mathcal{F}} \min_{\mathcal{O}' s.t. \mathcal{C}(\mathcal{O}', S) = f} \|\mathcal{O} - \mathcal{O}'\|_\infty$$

Definition 3 implies that the breakdown point is the amount of change to the oracle that is required to allow the compression algorithm to generate an arbitrary model where the change is measured in total variation distance. The following theorem proves the robustness of MEMO:

Theorem 2. *Let $\mathcal{O}$ be an oracle and S be a sample. Let $\hat{d}$ be the depth returned by the MEMO algorithm. If $p^* = \min_{x \in S, y \in Y} \mathcal{O}(x,y)$, then the breakdown point of MEMO with the oracle $\mathcal{O}$ and the sample S is at least $(\hat{d} - p^*)/2$.*

Proof. The proof follows from Theorem 1 and uses the same technique as Theorem 9 in (Gilad-Bachrach and Burges 2013): if $x^* \in S$ and $y^* \in Y$ are such that $p^* = \mathcal{O}(x^*, y^*)$ and $m = \mathcal{C}(\mathcal{O}, S)$ and $m' = \mathcal{C}(\mathcal{O}', S)$ then if $m'(x^*) = y^*$ then there exists $x \in S$ such that $\mathcal{O}'(x, m(x)) \leq \mathcal{O}'(x^*, y^*)$. Hence, $\hat{d} - p^* \leq (\mathcal{O}(x, m(x)) - \mathcal{O}'(x, m(x))) + (\mathcal{O}'(x^*, y^*) - \mathcal{O}(x^*, y^*))$ Therefore, at least one of the r.h.s. terms in the last inequality must be greater than $(\hat{d} - p^*)/2$ □

Note that the robustness here adds to robustness induced by the soft-max used to generate the oracle from the model M which is the only source of robustness for KD.

4 Compact Robust Estimated Median Belief Optimization

One nice property of the MEMO algorithm is that it does not require the oracle $\mathcal{O}$ to return the true probabilities $p(y|x)$. It is sufficient that the oracle will return $\mathcal{O}(x,y) = g\left(p\left(y|x\right)\right)$ where g is some monotone increasing function. In this case, the algorithm will return the deepest function f^* regardless of the choice of the function g. However the returned depth will be modified by g; it would be $g(d^*)$ where d^* here refers to the true depth; i.e., the one that would have been computed if g was the identify function.

In model compression, we compress some large model M into a smaller model m. Since we use M to construct the oracle $\mathcal{O}$ it is essential that M returns probabilities or, as discussed above, some monotone increasing function of these probabilities. This can be achieved, for example, by using a softmax layer at the end of a DNN or by taking the agreement probabilities of an ensemble (see a discussion in Section 6 about additional methods). These conversions allow the use of model M as the oracle $\mathcal{O}$. To achieve compression, the search for a deep function is made in a class of small models $\mathcal{F}$. In this setup, running the MEMO algorithm will find a compact function with the largest depth. However, the limited capacity of the function class $\mathcal{F}$ combined with some possible outliers in the data may make the constraints too stringent. One way to see that is to note that when the dataset S increases in size, more and more constraints are added to the MEMO algorithm, which decreases the maximal possible depth and therefore the depth function, as a method to distinguish between good and bad models, loses its dynamic range. This may make it hard to distinguish between functions that will generalize well and other functions that will not. To overcome this issue, we relax the constraint such that instead of requiring that $\forall x \in S, \quad \mathcal{O}(x, f(x)) \geq d$ we require that the condition holds for *most* $x \in S$. To this end, we define the δ-insensitive empirical depth:

Definition 4. *Let $\mathcal{O}$ be an oracle, let $S = \{x_1, ..., x_m\} \in X^m$ be a sample and let $\delta \in [0,1]$. The δ-insensitive empirical depth of f with respect to $\mathcal{O}$ is*

$$\hat{D}^{S,\delta}_{\mathcal{O}}(f) = \max_{T \subseteq S, |T| \geq (1-\delta)|S|} \hat{D}^S_{\mathcal{O}}(f)$$

The δ-insensitive empirical depth requires that function f will have a large agreement with $\mathcal{O}$ on all but a set of instances at a proportion smaller or equal to δ. In the language of robust statistics, the δ-insensitive empirical depth can be considered as a trimmed estimator (Daszykowski et al. 2007).

The *Compact Robust Estimated Median Belief Optimization* (CREMBO) algorithm (Algorithm 2) handles the trade off between robustness and accuracy by optimizing with respect to the δ-insensitive empirical depth. Unfortunately, optimizing with respect to this measure is harder and therefore the CREMBO algorithm is not guaranteed to find the deepest hypothesis with respect to the δ-insensitive empirical depth function and instead finds an approximation.

The CREMBO algorithm finds a function with large δ-insensitive empirical depth that performs well on a validation set. It starts with the solution provided by the MEMO algorithm. This provides an initial depth d^* that can be achieved with $\delta = 0$. The algorithm increases the required depth and for each threshold d of the depth it generates the set of allowed labels $Y_i = \{y \in Y \; s.t. \; \mathcal{O}(x_i, y) \geq d\}, \; \forall x_i \in S$ much like in the MEMO algorithm. However, since the depth is greater than the depth of the empirical median returned by the MEMO algorithm, there is no hypothesis in $\mathcal{F}$ that is consistent with this sample. Therefore, it allows the learning algorithm to return a hypothesis that is consistent with most of the training points but not all of them.

Since we do not know what a good value would be for δ up-front, a validation set is used by CREMBO to compare the hypotheses returned for different depth thresholds and select the best one. The selection criteria may be, accuracy, F1 or the AUC for example. The CREMBO algorithm uses linear search on the thresholds to find m. In cases where

Algorithm 2: Compact Robust Estimated Median Belief Optimization (CREMBO)

Input:

- A sample $S \in X^m$
- An oracle $\mathcal{O}(x, y)$
- A validation set $\mathcal{Z} \in (X \times Y)^u$
- The median hypothesis f^*, a set of sorted thresholds Θ and depth d^* computed by the MEMO algorithm $(f^*, \Theta, d^* \leftarrow MEMO())$
- A learning algorithm $\mathcal{A}$ that given a sample of the form $\hat{S} = \{(x_i, Y_i)\}_{i=1}^m$ where $Y_i \subseteq Y$ trains a function $f \in \mathcal{F}$ and returns it
- Evaluation metric V, that given a function f and a validation set $\mathcal{Z}$ returns the set score
- A step size Δ

Output: A deep compact function $f \in \mathcal{F}$

begin
  best $\leftarrow V(f^*, \mathcal{Z})$
  $f \leftarrow f^*$
  $D \leftarrow \Theta[\Theta \geq d^*][:: \Delta]$ `// Get all values in` Θ `larger than` d^* `with` Δ `interval`
  for *d in D* **do**
    for *i=1,...,m* **do**
      $Y_i = \{y \in Y \ s.t. \ \mathcal{O}(x_i, y) \geq d\}$
    $\hat{S} \leftarrow \{(x_i, Y_i)\}_{i=1}^m$
    $h \leftarrow \mathcal{A}(\hat{S})$
    score $\leftarrow$ V(h, $\mathcal{Z}$)
    if *score* $>$ *best* **then**
      $f \leftarrow h$
      best $\leftarrow$ score
  Return f

there are many threshold values it is possible to perform the search with steps of size Δ. This way, the number of iterations needed for the algorithm is $\frac{|\{\mathcal{O}(x_i,y):\ x_i \in S,\ y \in Y\}|}{\Delta}$. Using different search methods such as line search can reduce the number of iterations exponentially (Grippo, Lampariello, and Lucidi 1986).

5 Experiments

We evaluate the CREMBO algorithm using two sets of experiments. On the first set of experiments we evaluate the generalization and robustness of the CREMBO algorithm (Section 5.1). On the second set, we test CREMBOs ability to create accurate compact models on the DNN to DNN compression task and compare it to KD (Section 5.2).

5.1 Compressing to Interpretable Models

To evaluate the CREMBO algorithm as a robust model compression scheme, we conducted two experiments, a generalization experiment in which the compressed models accuracy and win rate were evaluated using 10-fold cross-validation and a robustness experiment where the compressed models were evaluated on the level of their agreement. In each experiment two types of models were compressed, a Random Forest model (RF) (Ho 1995) which is an ensemble model and a Deep Neural Network (DNN). Both models were compressed with the CREMBO algorithm to a small, fixed depth decision tree, the median tree (MED). These trees were compared to two other same depth trees: benchmark tree (BM), which is trained on the original training data $S_{train} = \{(x_i, y_i)\}_{i=1}^m$ and a student tree (ST), trained on labels generated from the large model (teacher) predictions $S_{teacher} = \{(x_i, M(x_i))\}_{i=1}^m$.

Dataset	Instances	Attributes	Classes
Dermatology	366	33	6
Heart	304	13	5
Arrhythmia	452	279	16
Breast cancer	569	30	2
Iris	150	4	3

Table 1: Dataset statistics

We evaluated the CREMBO algorithm on five classification tasks (Table 1) from the UCI repository (Dua and Graff 2017). To implement the DNNs we used PyTorch (Paszke et al. 2017). The DNNs are all fully connected with two hidden layers of 128 units with ReLu activation functions. They were trained with an ADAM optimizer with default parameters and batch size of 32 for 10 epochs. For the Random Forest and decision tree models we used scikit-learn (Pedregosa et al. 2011) package. The Random Forest model was trained with 100 trees with a maximal depth of 12 and balanced weights. All the decision tree models were trained with a maximal depth of 4, so they are small and interpretable, and balanced weights.

Generalization To evaluate the generalization ability of the compressed models we used 10-fold cross-validation (CV). In each round 9 folds are used as the training set S_{train} and the remaining fold is used as a test set. We first train the large model M and a benchmark tree on S_{train}, then using M predictions we create $S_{teacher}$ and train the student tree. To find the median tree, we split S_{train} into a train and validation sets, S'_{train}, S_{val}, with a random 15% split and run the CREMBO algorithm. The accuracy on the test set is calculated for all models and later averaged on all rounds. In addition, we measure the win rate for each model. The win rate is the percentage of rounds in which a model outperformed the other models. We repeated the experiment 20 times and the average results are provided in Table 2, and Table 3. The results show that for Random Forest compression, the median tree had the best accuracy and win rates by a considerable margin on all datasets except for the Breast cancer dataset on which the benchmark tree had better accuracy by a relatively small margin. For DNN compression, there were similar results for the Dermatology, Heart, Arrhythmia and Breast cancer datasets. On the first three, the

	Accuracy				Win rate		
Dataset	RF	BM	ST	MED	BM	ST	MED
Dermatology	98.05	82.26	82.18	**90.62**	2	22.5	**75.5**
Heart	56.24	38.93	38.93	**52.6**	0	0.5	**99.5**
Arrhythmia	70.89	9.36	7.7	**54.79**	0	0	**100**
Breast cancer	96	**93.25**	93.13	92.47	13.5	32	**54.5**
Iris	94.53	92.53	92.53	**94.66**	0	12	**88**

Table 2: Accuracy and win rate results (in percentage) over 10-fold CV of benchmark tree (BM), student tree (ST) and median tree (MED) averaged over 20 experiments where the compressed model is a Random Forest (RF)

	Accuracy				Win rate		
Dataset	DNN	BM	ST	MED	BM	ST	MED
Dermatology	97.56	82.2	82.22	**89.55**	6	25.5	**68.5**
Heart	64.27	38.87	49.6	**52.4**	0.5	33	**66.5**
Arrhythmia	70.41	9.46	15.33	**55.6**	0	6.5	**93.5**
Breast cancer	93.87	93.21	**94.03**	92.68	25.5	**45**	29.5
Iris	36.92	**92.33**	92.06	91.19	13.5	15	**71.5**

Table 3: Accuracy and win rate results (in percentage) over 10-fold CV of, benchmark tree (BM), student tree (ST) and median tree (MED) averaged over 20 experiments where the compressed model is a Deep Neural Network (DNN)

median tree outperformed the other trees and on the last it was less accurate. An interesting result emerged for the Iris dataset. The DNN is clearly overfitted, since it has an average accuracy score of only 36.92% on the test sets. The student tree was hardly affected since the DNN predictions on the training set were very accurate. On the other hand, there was a negative impact on the median tree since the belief probabilities $p(y|x)$ provided by the DNN were not accurate enough. Nevertheless, the median tree still had the highest win rate and much higher accuracy on the test sets than the larger DNN model.

Robustness In Theorem 2 we were able to prove the robustness of MEMO to changes in the oracle. Here, we evaluate robustness empirically by training the big model M with different training sets and measuring the impact on the compressed models. To measure similarity between compressed models we say that models *agree* on x if they make the same prediction on this point, regardless of the correctness of this prediction. In the experiment we divided the dataset into a train and test sets with a random 15% split. To simulate data perturbations, we used 10-fold CV on the training set. On each round, we took 9 of the 10 folds to be S_{train} while the remaining fold was omitted. The training process of large model M and the trees was done in the same manner as in the generalization experiment. We measured the agreement of same type trees across rounds on the test set and averaged the score. This experiment was repeated 20 times and the average scores are presented in Table 4. The median tree was more robust on 7 out of 10 test settings (4 out of 5 when compressing to trees and 3 out of 5 when compressing to neural nets), in some cases with very large margins. On the other 3 cases it was close to the other techniques in terms of robustness.

The results from the generalization and the robustness experiments show that the CREMBO algorithm is able to compress Random Forests and DNNs to small decision trees that are more accurate and robust than same sized trees trained with comparable methods on a variety of datasets. The average accuracy improvement over datasets (in absolute percentage) was 13.76% for RF compression and 9.57% for DNN compression and the average robustness improvements were 12.7% and 7.8% for RF and DNN compression respectively. We note that our results are statistically significant.

5.2 DNN to DNN Compression

DNN to DNN compression is a highly studied filed (Buciluǎ, Caruana, and Niculescu-Mizil 2006). To test our method's ability to compress large DNNs to compact DNNs, we used CREMBO to compress large DNNs to compact DNNs and compared them to baseline models, i.e., models trained small from the get-go, and to compact models generated with Knowledge Distillation (KD). We compressed two types of large DNNs, ResNet18 (He et al. 2016) and VGG16 (Simonyan and Zisserman 2014) to two compact DNNs, LeNet-5 (LeCun et al. 1998) and MobileNetV2 (Sandler et al. 2018). Where LeNet-5 is a very small DNN and MobileNetV2 is a compact DNN designed to run on mobile devices. The models were trained on the CIFAR-10 dataset (Krizhevsky, Hinton et al. 2009). The number of parameters and baseline results of all models on CIFAR-10 are presented on Table 6.

The training process was the same for all DNNs. We used ADAM optimizer, batch size of 128, learning rate of 0.01 for 60 epochs and then learning rate of 0.001 for another 30 epochs. We first trained the large DNNs (M) on the training

	Random Forest			DNN		
Dataset	BM	ST	MED	BM	ST	MED
Dermatology	91.89	91.9	**92.18**	91.88	90.01	**92.67**
Heart	63.04	63.08	**70.29**	63.17	60.76	**68.51**
Arrhythmia	32.76	32.17	**89.91**	32.63	21.03	**72.72**
Breast cancer	94.7	94.85	**95.91**	**94.71**	93.3	92.63
Iris	99.52	**99.65**	97.85	**99.47**	96.67	94.44

Table 4: Agreement results (in percentage) of benchmark tree (BM), student tree (ST) and median tree (MED) averaged over 20 experiments where the large models compressed are Random Forest (left) and Deep Neural Network (right)

M	m	T	Baseline(m)	KD	CREMBO
ResNet18	LeNet-5	20	70.76	71.53	**72.27**
VGG16	LeNet-5	5	70.76	70.49	**72.43**
ResNet18	MobileNetV2	5	91.97	**92.17**	**92.19**
VGG16	MobileNetV2	5	91.97	92.15	**92.35**

Table 5: Accuracy results ($\%$) for large model M compression to small model m with Knowledge Distillation (KD) and CREMBO on CIFAR-10. Baseline accuracy for the small model m and Temperature (T) used are provided

Model	# Parameters	Accuracy ($\%$)
resnet18	11173962	93.15
VGG16	14728266	92.22
MobileNetV2	2296922	91.97
LeNet-5	62006	70.76

Table 6: Number of model parameters and baseline accuracy on CIFAR-10

set. Then we divided the training set to a train and validation set with a random 10% split. We used the validation set for the CREMBO algorithm and to find the best KD temperature value out of $[3, 5, 8, 20]$ for each model. After finding the best temperature values, CREMBO and KD were used to compress M to compact DNNs (m). We implemented KD as in (Hinton, Vinyals, and Dean 2015) using both soft and regular targets as recomended in (Hinton, Vinyals, and Dean 2015). The results of our experiments are presented in Table 5.

The results show that CREMBO improves the baseline and outperforms KD on all tested models. This is another testimony for CREMBO's flexibility and ability to compress large models to compact models that generalize well.

6 Conclusions

In this study we presented a novel robust model compression scheme for multi-class classifiers that can compress variety of large models, such as DNNs and ensembles, into compact models. To ensure robustness it uses tools from robust statistics; namely, the statistical depth and trimmed estimators. We presented the MEMO algorithm, a new algorithm for finding the empirical median hypothesis in the multi-class setting. For model compression we introduced the CREMBO algorithm. CREMBO uses a trimmed version of the depth function to search for deep hypotheses in a class of compact classifiers and therefore achieve both robustness and compression. We demonstrated the ability of CREMBO to compress both DNNs and ensembles into small decision trees which are more accurate and robust than trees trained with comparable methods. This is useful for different explainability purposes since small trees are comprehensible while ensembles and DNNs are much harder to interpret. The robustness and accuracy of the compressed model ensure that it represents the large model it captures well.

Compressing models is also advantageous in other scenarios, such as when a model is to be used on a device with limited resources or when latency is critical. This is especially true for large DNNs which are known for their large size and computing demands. To this end, we evaluated CREMBO's ability to compress large DNNs to compact DNNs and showed that CREMBO outperforms KD.

Although CREMBO works for a variety of model types, it is possible to add model specific variations to further improve results. For example, adding temperature to CREMBO in DNN to DNN compression. DNNs predictions tend to be overconfident and in general not well-calibrated (Guo et al. 2017). This means that the posterior probabilities we get from M can be overconfident as well. Adding temperature has been said to improve calibration (Guo et al. 2017). This variation can be seen as a combination of KD and CREMBO in which the allowed labels get weighted according to the soft predictions. We leave this and other possible variations for future work.

References

Banerjee, A. 1997. Initializing neural networks using decision trees. *Computational learning theory and natural learning systems* 4: 3.

Bénard, C.; Biau, G.; Da Veiga, S.; and Scornet, E. 2020. In-

terpretable Random Forests via Rule Extraction. *arXiv preprint arXiv:2004.14841* .

Buciluǎ, C.; Caruana, R.; and Niculescu-Mizil, A. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 535–541.

Caruana, R.; Karampatziakis, N.; and Yessenalina, A. 2008. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, 96–103.

Caruana, R.; Lou, Y.; Gehrke, J.; Koch, P.; Sturm, M.; and Elhadad, N. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1721–1730.

Che, Z.; Purushotham, S.; Khemani, R.; and Liu, Y. 2016. Interpretable deep models for ICU outcome prediction. In *AMIA Annual Symposium Proceedings*, volume 2016, 371. American Medical Informatics Association.

Cheng, Y.; Wang, D.; Zhou, P.; and Zhang, T. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282* .

Daszykowski, M.; Kaczmarek, K.; Vander Heyden, Y.; and Walczak, B. 2007. Robust statistics in data analysis—A review: Basic concepts. *Chemometrics and intelligent laboratory systems* 85(2): 203–219.

Denil, M.; Shakibi, B.; Dinh, L.; Ranzato, M.; and De Freitas, N. 2013. Predicting parameters in deep learning. In *Advances in neural information processing systems*, 2148–2156.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml.

Frosst, N.; and Hinton, G. 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784* .

Gilad-Bachrach, R.; and Burges, C. J. 2013. Classifier selection using the predicate depth. *The Journal of Machine Learning Research* 14(1): 3591–3618.

Gong, Y.; Liu, L.; Yang, M.; and Bourdev, L. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115* .

Grippo, L.; Lampariello, F.; and Lucidi, S. 1986. A nonmonotone line search technique for Newton's method. *SIAM Journal on Numerical Analysis* 23(4): 707–716.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1321–1330. JMLR. org.

Hamilton, I. A. 2018. Why It's Totally Unsurprising That Amazon's Recruitment AI Was Biased against Women. *Business Insider, October* 13.

Hampel, F. R. 1971. A general qualitative definition of robustness. *The Annals of Mathematical Statistics* 1887–1896.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Ho, T. K. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, 278–282. IEEE.

Kononenko, I. 2001. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine* 23(1): 89–109.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images .

Lage, I.; Chen, E.; He, J.; Narayanan, M.; Kim, B.; Gershman, S.; and Doshi-Velez, F. 2019. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006* .

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

Quinlan, J. R. 1986. Induction of decision trees. *Machine learning* 1(1): 81–106.

Rigamonti, R.; Sironi, A.; Lepetit, V.; and Fua, P. 2013. Learning separable filters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2754–2761.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Srinivas, S.; and Babu, R. V. 2015. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149* .

Tukey, J. W. 1975. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*, volume 2, 523–531.

Usmanim, Z. u. H. 2018. How to Win Kaggle Competitions. https://www.kaggle.com/getting-started/44997. (accessed May 22, 2020).