2024 American Control Conference (ACC)
July 8-12, 2024. Toronto, Canada

Compression Repair for Feedforward Neural Networks Based on Model Equivalence Evaluation

Zihao Mo, Yejiang Yang, Shuaizheng Lu, and Weiming Xiang

Abstract— **In this paper, we propose a method of repairing compressed Feedforward Neural Networks (FNNs) based on equivalence evaluation of two neural networks. In the repairing framework, a novel neural network equivalence evaluation method is developed to compute the output discrepancy between two neural networks. The output discrepancy can quantitatively characterize the output difference produced by compression procedures. Based on the computed output discrepancy, the repairing method first initializes a new training set for the compressed networks to narrow down the discrepancy between the two neural networks and improve the performance of the compressed network. Then, we repair the compressed FNN by re-training based on the training set. We apply our developed method to the MNIST dataset to demonstrate the effectiveness and advantages of our proposed repair method.**

I. Introduction

Back in 1943, McCulloch and Pitts [1] brought up an idea about using logical calculus to simulate nervous activity, recognized as the origin of neural networks. Since then, neural networks have developed over the decades and become a fundamental tool in modern intelligent society. It has been applied in many areas, such as pattern recognition [2], [3], image processing [4], [5], computer vision [6], [7], etc. However, the evolution of neural networks is accompanied by the exponential growth of the scale and computation cost. According to the paper [8], [9], training a large feedforward neural network is power-consuming and is in high demand for memory usage. In an ACAS Xu [10] verification problem, 45 Feedforward Neural Networks (FNNs) have been deployed, with a total of 300 neurons of each network, which requires huge computational resources and is time-consuming [11]. Thus, the compression of neural networks becomes a hot topic in the industrial area, which can shrink down the scale of FNNs to be deployed in practical applications. Two main state-of-the-art compression methods are pruning [12] and quantization [13], where the former aims to reduce neurons and layers of the model, and the latter focuses on replacing high-precision parameters with low-precision parameters.

However, neural network compression always comes at a cost. In the survey [14], the authors outlined four methods for compression and acceleration, but the summary also identified certain potential issues, notably a substantial reduction in the accuracy of the compressed network. In [15], the severness of accuracy loss and importance of safety verifications in Cyber-Physical Systems (CPS) applications has been addressed; for example, collisions may occur if ACAS Xu verification fails. Thus, the verification of compressed FNNs is essential before deploying them. The paper [16] gives a concrete value to characterize the difference between two FNNs by performing the reachability analysis between the networks, which is more intuitive to identify whether the network meets those criteria. This hybrid zonotopes method [17] is also based on reachability analysis to verify the safety robustness of a neural feedback system by providing a quantitative result.

In this paper, we propose a novel merging method to perform reachability analysis between two feedforward neural networks with the same inputs to evaluate the equivalence of the compressed network with respect to the original one. With the given discrepancy result, we can identify the guaranteed output reachable domain of the compressed network. Further, we propose a repair framework for the compressed FNN based on the equivalence result to narrow down the discrepancy with the original FNN while retaining its performance in solving specific tasks. To demonstrate the effectiveness of our repair method, we apply it to classify the MNIST database and compare the repair outcomes of the compressed FNN with respect to the original FNN.

The remainder of the paper is organized as follows: Section II is about Preliminaries. Section III introduces our merging method and the framework of the repair method. Section IV is the experiment demonstrating the repair method. Section V presents the conclusion.

II. Preliminaries

In this paper, an FNN $\Phi : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ is defined in the recursive equations form of

$$\begin{cases} \mathbf{y}^{\{l\}} = \phi^{\{l\}}(\mathbf{y}^{\{l-1\}}),\ l = 1, \dots, L \\ \mathbf{y}^{\{L\}} = \Phi(\mathbf{x}^{\{0\}}),\ \text{where } \mathbf{x}^{\{0\}} = \mathbf{y}^{\{0\}} \end{cases} \tag{1}$$

where $\mathbf{y}^{\{l\}}$ is the output of the lth layer, $\mathbf{x}^{\{0\}} \in \mathbb{R}^{n^{\{0\}}}$ is the input and $\mathbf{y}^{\{L\}} \in \mathbb{R}^{n^{\{l\}}}$ is the output of the FNN, respectively. $\phi^{\{l\}}$ denotes the operation of the lth layer of the FNN, which can be fully connected layer ϕ_{fc} or ReLU layer ϕ_{ReLU}, and $\mathbf{y}^{\{l\}}$ is the output of the lth layer. The fully connected operation ϕ_{fc} is defined as

$$\mathbf{y}^{\{l\}} = \phi_{\text{fc}}(\mathbf{y}^{\{l-1\}}) = \mathbf{W}^{\{l\}}\mathbf{y}^{\{l-1\}} + \mathbf{b}^{\{l\}} \tag{2}$$

where $\mathbf{W}^{\{l\}} \in \mathbb{R}^{n^{\{l\}} \times n^{\{l-1\}}}$ and $\mathbf{b}^{\{l\}} \in \mathbb{R}^{n^{\{l\}}}$ denote the weight matrices and the bias vectors for layer l, respectively.

This research was supported by the National Science Foundation, under NSF CAREER Award no. 2143351, NSF CNS Award no. 2223035, and NSF IIS Award no. 2331938.

The authors are with the School of Computer and Cyber Sciences, Augusta University, Augusta, GA 30912, USA. wxiang@augusta.edu

The ReLU operation ϕ_{ReLU} is defined as

$$\phi_{\text{ReLU}}(\mathbf{y}^{\{l\}}) = [\max(0, y_1^{\{l\}}), \ldots, \max(0, y_n^{\{l\}})]^T \quad (3)$$

where $y_i^{\{l\}}$ is the ith element of the vector $\mathbf{y}^{\{l\}}$ in (2).

To enable sound equivalence evaluation for two FNNs, which essentially needs to consider all possible outputs of the networks, the following reachable set of FNNs is introduced.

Definition 1: Given an input set $\mathcal{X}^{\{0\}} \in \mathbb{R}^{n^{\{0\}}}$ for FNN (1), we define the following set

$$\mathcal{Y}^{\{L\}} = \{\mathbf{y}^{\{L\}} \mid \mathbf{y}^{\{L\}} = \Phi(\mathbf{x}^{\{0\}}),\ \mathbf{x}^{\{0\}} \in \mathcal{X}^{\{0\}}\} \quad (4)$$

where $\mathcal{Y}^{\{L\}} \subseteq \mathbb{R}^{n^{\{L\}}}$ is called the output set of FNN (1).

Remark 1: There are a number of available reachable set representations used for FNN reachability analysis, such as zonotope [18], polytope [19], FVIM [20], etc. For instance, in the MNIST dataset application Section IV, we will use ImageStar proposed in [21] in our approach. ImageStar Θ is a tuple $\langle c, V, P\rangle$ where $c \in \mathbb{R}^{h\times w\times nc}$ is the anchor image, $V = \{v_1, v_2, \cdots, v_m\}$ is a set of m images in $\mathbb{R}^{h\times w\times nc}$ called generator images, $P : \mathbb{R}^m \leftarrow \{\top, \bot\}$ is a predicate, and h, w, nc are the height, width, and number of channels of the images, respectively. The generator images are arranged to form the ImageStar's $h \times w \times nc \times m$ basis array. The set of images represented by the ImageStar is:

$$\Theta = \{x | x = c + \sum_{i=1}^{m}(\alpha_i v_i),\ \text{where } P(\alpha_i, \cdots, \alpha_m) = \top\}$$

in which we restrict the predicates to be a conjunction of linear constraints, $P(\alpha) \triangleq C\alpha \leq d$ where, for p linear constraints, $C \in \mathbb{R}^{p\times m}$, α is the vector of m variables, i.e., $\alpha = [\alpha_1, \cdots, \alpha_m]^T$, and $d \in \mathbb{R}^{p\times 1}$. An ImageStar is an empty set if and only if $P(\alpha)$ is empty.

Based on the output reachable set defined in (4), we can define the set value representation of an FNN as below

$$\begin{cases} \mathcal{Y}^{\{l\}} = \phi^{\{l\}}(\mathcal{Y}^{\{l-1\}}),\ l = 1, 2, \ldots, L \\ \mathcal{Y}^{\{L\}} = \Phi(\mathcal{X}^{\{0\}}),\ \text{where } \mathcal{X}^{\{0\}} = \mathcal{Y}^{\{0\}} \end{cases} \quad (5)$$

where $\mathcal{Y}^{\{l\}}$ denotes the output reachable set of lth layer, and, in particular, $\mathcal{Y}^{\{0\}}$ is the input set $\mathcal{X}^{\{0\}}$ and $\mathcal{Y}^{\{L\}}$ is the output set of the network.

In this paper, the equivalence evaluation aims to characterize the discrepancy between two FNNs, Φ_1 and Φ_2 under the following assumptions.

Assumption 1: The following assumptions hold for two neural networks Φ_1 and Φ_2:

(i) The number of inputs of two neural networks are the same, i.e., $n_1^{\{0\}} = n_2^{\{0\}}$;
(ii) The number of outputs of two neural networks are the same, i.e., $n_1^{\{L\}} = n_2^{\{L\}}$;
(iii) The number of layers of two neural networks is the same, i.e., $L_1 = L_2 = L$;
(iv) For each layer l, two neural networks perform the same operation.

Remark 2: It has to be pointed out that a typical compressed neural network usually consists of a reduced number of layers compared to the original network, which fails to satisfy (iii) and (iv) in Assumption 1. However, we can always extend the compressed network by incorporating additional layers as detailed in [16]. These additional layers equipped with identity weights and zero biases are mandated to transmit information to subsequent layers without any alterations, but meet the requirements of (iii) and (iv).

III. Main Results

A. Equivalence Evaluation for Two FNNs

Given an input set $\mathcal{X}^{\{0\}}$ for two FFNs Φ_1 and Φ_2 under Assumption 1, the equivalence evaluation in this work is given by quantifying the maximal discrepancy of $\mathbf{y}_1^{\{L\}}$ and $\mathbf{y}_2^{\{L\}}$, where are the outputs of Φ_1 and Φ_2, respectively. To enable equivalence evaluation of Φ_1 and Φ_2, our first goal is to construct the discrepancy of the outputs of two FNNs with the same inputs, i.e.,

$$\delta = \Phi_1(\mathbf{x}^{\{0\}}) - \Phi_2(\mathbf{x}^{\{0\}}),\ \mathbf{x}^{\{0\}} \in \mathcal{X}^{\{0\}} \quad (6)$$

where $\delta \in \mathbb{R}^{n^{\{L\}}}$ is the discrepancy vector.

For fully connected layers ϕ_{fc} and ReLU layers ϕ_{ReLU}, we can obtain the following two results.

Lemma 1: Consider two FFNs $\mathcal{N}_1$ and $\mathcal{N}_2$ under Assumption 1, the following result holds for fully connected layers

$$\begin{bmatrix} \mathbf{y}_1^{\{l\}} \\ \mathbf{y}_2^{\{l\}} \end{bmatrix} = \phi_{\text{fc}}\left(\begin{bmatrix} \mathbf{y}_1^{\{l-1\}} \\ \mathbf{y}_2^{\{l-1\}} \end{bmatrix}\right) = \tilde{\mathbf{W}}^{\{l\}} \begin{bmatrix} \mathbf{y}_1^{\{l-1\}} \\ \mathbf{y}_2^{\{l-1\}} \end{bmatrix} + \tilde{\mathbf{b}}^{\{l\}} \quad (7)$$

where $\tilde{\mathbf{W}}^{\{l\}} = \text{diag}\{\mathbf{W}_1^{\{l\}}, \mathbf{W}_2^{\{l\}}\}$ and $\tilde{\mathbf{b}}^{\{l\}} = [(\mathbf{b}_1^{\{l\}})^T, (\mathbf{b}_2^{\{l\}})^T]^T$ in which $\mathbf{W}_1^{\{l\}}$, $\mathbf{W}_2^{\{l\}}$, $\mathbf{b}_1^{\{l\}}$, $\mathbf{b}_2^{\{l\}}$ are the weights and biases of $\mathcal{N}_1$ and $\mathcal{N}_2$ at layer l.

Proof: The result can be obtained straightforwardly by the definition of the fully connected layer in the form of (2) such as

$$\begin{aligned} \phi_{\text{fc}}\left(\begin{bmatrix} \mathbf{y}_1^{\{l-1\}} \\ \mathbf{y}_2^{\{l-1\}} \end{bmatrix}\right) &= \begin{bmatrix} \phi_{\text{fc}}(\mathbf{y}_1^{\{l-1\}}) \\ \phi_{\text{fc}}(\mathbf{y}_2^{\{l-1\}}) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{W}_1^{\{l\}}(\mathbf{y}_1^{\{l-1\}}) + \mathbf{b}_1^{\{l\}} \\ \mathbf{W}_2^{\{l\}}(\mathbf{y}_2^{\{l-1\}}) + \mathbf{b}_2^{\{l\}} \end{bmatrix} \\ &= \tilde{\mathbf{W}}^{\{l\}} \begin{bmatrix} \mathbf{y}_1^{\{l-1\}} \\ \mathbf{y}_2^{\{l-1\}} \end{bmatrix} + \tilde{\mathbf{b}}^{\{l\}}. \end{aligned}$$

The proof is complete. ■

Lemma 2: Consider two FFNs $\mathcal{N}_1$ and $\mathcal{N}_2$ under Assumption 1, the following result

$$\begin{bmatrix} \mathbf{y}_1^{\{l\}} \\ \mathbf{y}_2^{\{l\}} \end{bmatrix} = \phi_{\text{ReLU}}\left(\begin{bmatrix} \mathbf{y}_1^{\{l-1\}} \\ \mathbf{y}_2^{\{l-1\}} \end{bmatrix}\right) = \begin{bmatrix} \phi_{\text{ReLU}}(\mathbf{y}_1^{\{l-1\}}) \\ \phi_{\text{ReLU}}\mathbf{y}_2^{\{l-1\}}) \end{bmatrix} \quad (8)$$

holds for ReLU layers.

Proof: As the ReLU operation is performed in an element-wise manner, the result can be obtained straightforwardly. The proof is complete. ■

Remark 3: It should be noted that the ReLU function may split the reachable set into multiple ones based on the linear constraints. The ReLU function is possible to perform in a

different way for the pixel if its bounded range is across the zero. To handle the different situations, our method will split the linear constraints into two groups to force the bounded range of the pixel to fall into one side, leading to $\mathcal{Y} = \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \ldots \cup \mathcal{Y}_n$. With the split strategy, the reachable set may increase exponentially based on the linear constraints, which increases the computation time.

Besides fully connected and ReLU layers, we introduce a comparison layer to construct the output discrepancy $\mathbf{y}_{\text{cmp}} = \mathbf{y}_1^{\{L\}} - \mathbf{y}_2^{\{L\}}$ to evaluate the equivalence on two FNNs. The comparison layer is considered as an extra layer receiving the output $\mathbf{y}_1^{\{L\}}$ and $\mathbf{y}_2^{\{L\}}$ of FNNs $\mathcal{N}_1$ and $\mathcal{N}_2$ in the form of

$$\mathbf{y}_{\text{cmp}} = \phi_{\text{cmp}}\left(\begin{bmatrix}\mathbf{y}_1^{\{L\}}\\ \mathbf{y}_2^{\{L\}}\end{bmatrix}\right) = \mathbf{W}_{\text{cmp}}\begin{bmatrix}\mathbf{y}_1^{\{L\}}\\ \mathbf{y}_2^{\{L\}}\end{bmatrix} \tag{9}$$

where $\mathbf{W}_{\text{cmp}} = \begin{bmatrix}\mathbf{I} & -\mathbf{I}\end{bmatrix}$.

Theorem 1: Consider two FFNs Φ_1 and Φ_2 under Assumption 1, the following result

$$\mathbf{y}_{\text{cmp}} = \Phi_1(\mathbf{x}^{\{0\}}) - \Phi_2(\mathbf{x}^{\{0\}}) \tag{10}$$

holds for the output $\mathbf{y}_{\text{cmp}}$ of comparison layer defined in (9).

Proof: By Lemmas 1 and 2, and under Assumption 1, we have

$$\begin{bmatrix}\mathbf{y}_1^{\{l\}}\\ \mathbf{y}_2^{\{l\}}\end{bmatrix} = \phi^{\{l\}}\left(\begin{bmatrix}\mathbf{y}_1^{\{l-1\}}\\ \mathbf{y}_2^{\{l-1\}}\end{bmatrix}\right) \; l = 1, \ldots, L \tag{11}$$

where $\phi^{\{l\}}$ can be either ϕ_{fc} or ϕ_{ReLU}. It is worth noting that Lemmas 1 and 2 also provide the computation procedures to compute $\mathbf{y}_1^{\{l\}}$ and $\mathbf{y}_2^{\{l\}}$ for each layer. Therefore, with the same input $\mathbf{x}^{\{0\}}$, it leads to

$$\begin{bmatrix}\mathbf{y}_1^{\{L\}}\\ \mathbf{y}_2^{\{L\}}\end{bmatrix} = \begin{bmatrix}\phi^{\{L\}} \circ \cdots \circ \phi^{\{1\}}(\mathbf{x}^{\{0\}})\\ \phi^{\{L\}} \circ \cdots \circ \phi^{\{1\}}(\mathbf{x}^{\{0\}})\end{bmatrix} = \begin{bmatrix}\Phi_1(\mathbf{x}^{\{0\}})\\ \Phi_2(\mathbf{x}^{\{0\}})\end{bmatrix} \tag{12}$$

Then, from (9), one can obtain

$$\mathbf{y}_{\text{cmp}} = \begin{bmatrix}\mathbf{I} & -\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{y}_1^{\{L\}}\\ \mathbf{y}_2^{\{L\}}\end{bmatrix} = \Phi_1(\mathbf{x}^{\{0\}}) - \Phi_2(\mathbf{x}^{\{0\}}). \tag{13}$$

The proof is complete. ∎

As shown in Theorem 1, the output of the comparison layer, i.e., $\mathbf{y}_{\text{cmp}}$, is the discrepancy vector measuring the output difference between two FFNs Φ_1 and Φ_2. With the help of this discrepancy vector and reachability analysis, we can formally characterize the equivalence of two FNNs. A merged $L+1$ layer FNN $\tilde{\Phi}$ out of Φ_1 and Φ_2 can constructed in the form of

$$\begin{cases}\tilde{\mathbf{y}}^{\{l\}} = \phi^{\{l\}}(\tilde{\mathbf{y}}^{\{l-1\}}), \; l = 1, \ldots, L\\ \tilde{\mathbf{y}}^{\{L+1\}} = \phi_{\text{cmp}}(\tilde{\mathbf{y}}^{\{L\}})\\ \tilde{\mathbf{y}}^{\{L+1\}} = \tilde{\Phi}(\mathbf{x}^{\{0\}}), \text{ where } \mathbf{x}^{\{0\}} = \mathbf{y}^{\{0\}}\end{cases} \tag{14}$$

in which fully connected and ReLU hidden layers from 1 to L are defined by (7) and (8) and the output layer $L+1$ is defined by (9). By performing reachability analysis for merged FNN (14), i.e., computing the output reachable set $\tilde{\mathcal{Y}}^{\{L+1\}}$ of merged FNN (14), we can formally characterize equivalence between two FNNs Φ_1 and Φ_2. For instance, we can compute the maximal distance of the outputs of two FFNs in terms of

$$\delta_{\max} = \max_{\tilde{\mathbf{y}}^{\{L+1\}} \in \tilde{\mathcal{Y}}^{\{L+1\}}} \left\|\tilde{\mathbf{y}}^{\{L+1\}}\right\| \tag{15}$$

In some scenarios, we might be interested in the discrepancies for each dimension, such as the image recognition application in Section IV. We can also make use of the reachable set $\tilde{\mathcal{Y}}^{\{L+1\}}$ to compute the vector of maximal magnitudes at each dimension of $\tilde{\mathcal{Y}}^{\{L+1\}}$, i.e.,

$$\tilde{\delta}_{\max} = \max_{\tilde{\mathbf{y}}^{\{L+1\}} \in \tilde{\mathcal{Y}}^{\{L+1\}}} \left|\tilde{\mathbf{y}}^{\{L+1\}}\right| \tag{16}$$

where the $\max$ operator performs element-wisely on $\tilde{\mathbf{y}}^{\{L+1\}}$.

Remark 4: To perform the efficient equivalence evaluation, the reachable set computation is essential. There exist a number of tools available. For instance, as in the NNV neural network reachability analysis tool, the reachable sets are in the form of a collection of polyhedral sets [22], and in the IGNNV tool, the output reachable set is a family of interval sets [23], [24]. For those types of convex sets, the equivalence evaluation metrics in the description of maximal values can be obtained by testing a finite number of vertices in convex sets.

B. FNN Compression Repair

Given an FNN Φ_1 and its compressed version Φ_2, the goal of repairing the compressed Φ_2 is that the discrepancy between Φ_1 and Φ_2 should satisfy a set of prescribed conditions described by set $\mathcal{O}$, e.g., $\mathcal{O} = \{\tilde{\mathbf{y}}^{\{L+1\}} \mid \left\|\tilde{\mathbf{y}}^{\{L+1\}}\right\| \leq d\}$ where $d > 0$ is a prescribed threshold. The general compressed FNN repair problem can be stated as follows.

Problem 1: Given an FNN Φ_1 and its compressed version Φ_2, an input set $\mathcal{X}^{\{0\}}$, and a prescribed repairing target set $\mathcal{O}$, how does one modify the compressed FNN Φ_2 such that

$$\tilde{\mathcal{Y}}^{\{L+1\}} \subseteq \mathcal{O} \tag{17}$$

where $\tilde{\mathcal{Y}}^{\{L+1\}}$ is the output reachable set of merged $L+1$ layer FNN $\tilde{\Phi}$ in the form of (14) that is constructed out of Φ_1 and Φ_2.

To address the FNN compression repair problem, normally, the goal of the repair is to minimize the discrepancy between FNNs Φ_1 and Φ_2. From the optimization perspective, the repair problem can be described as

$$\min_{\mathbf{W}_2^{\{l\}}, \mathbf{b}_2^{\{l\}}, \; l=1,\ldots,L} \ell(\mathbf{y}_1^{\{L\}}, \mathbf{y}_2^{\{L\}}) \tag{18}$$

where $\ell(\cdot)$ is the loss function describing the discrepancy such as (15) and (16).

To modify $\mathbf{W}_2^{\{l\}}$ and $\mathbf{b}_2^{\{l\}}$ to repair the compressed FNN, a new dataset has to be created for retraining the compressed FNN. A straightforward way is to generate N retraining data pair $\{\mathbf{x}_i^{\{0\}}, \mathbf{y}_{i,2}^{\{L\}}\}$, $i = 1, \ldots, N$, from FNN Φ_2, and replace the output samples $\mathbf{y}_{i,2}^{\{L\}}$ with the outputs of the original Φ_1 , i.e., $\{\mathbf{x}_i^{\{0\}}, \mathbf{y}_{i,1}^{\{L\}}\}$, $i = 1, \ldots, N$, which completely eliminates the discrepancy in data set. Furthermore, the cost

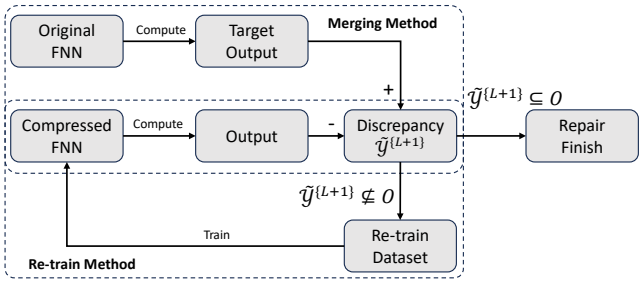

Fig. 1. Framework of compressed feedforward neural network repair.

function $\ell(\cdot)$ for the retraining data set can be then written into the mean square loss function in the retraining process as follows:

$$\ell(\mathbf{y}_1^{\{L\}}, \mathbf{y}_2^{\{L\}}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{y}_{i,1}^{\{L\}} - \mathbf{y}_{i,2}^{\{L\}}\right\|. \tag{19}$$

With the above loss function, the FNN Φ_2 training process can be viewed as a data-driven procedure to search for the optimal solution.

However, this method intends to cause overfitting issues and significantly deteriorates network performance, such as accuracy. In this work, we turn to gradually reduce the discrepancy by updating the $\mathbf{y}_{i,2}^{\{L\}}$ in the following way

$$\hat{\mathbf{y}}_{i,2}^{\{L\}} = \mathbf{y}_{i,2}^{\{L\}} + \frac{1}{\alpha}\tilde{\delta}_{max} \tag{20}$$

where $\alpha \geq 1$ is a tuning parameter to control the step size to the target output, and $\tilde{\delta}_{max}$ is defined by (16). Therefore, the data in retraining data is modified to $\{\mathbf{x}_i^{\{0\}}, \hat{\mathbf{y}}_{i,2}^{\{L\}}\}$, and the loss function becomes

$$\ell(\hat{\mathbf{y}}_2^{\{L\}}, \mathbf{y}_2^{\{L\}}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\mathbf{y}}_{i,2}^{\{L\}} - \mathbf{y}_{i,2}^{\{L\}}\right\|. \tag{21}$$

Thus, we can bring up the framework for compressed FNN repair based on equivalence evaluation as shown in Fig. 1.

- **Initialization.** Given FNN Φ_1 and its compression Φ_2, we compute the output reachable set $\tilde{\mathcal{Y}}^{\{L+1\}}$ out of merged FNN. If (17) is not satisfied, the compressed FNN needs to be re-trained. We compute the discrepancy $\tilde{\delta}_{max}$ based on $\tilde{\mathcal{Y}}^{\{L+1\}}$.
- **Generate re-training data set.** Generate N samples of $\{\mathbf{x}_i^{\{0\}}, \mathbf{y}_{i,1}^{\{L\}}, \mathbf{y}_{i,2}^{\{L\}}\}$, and build the re-training data set $\{\mathbf{x}_i^{\{0\}}, \hat{\mathbf{y}}_{i,2}^{\{L\}}\}$, $i = 1, \ldots, N$, based on (20).
- **Re-train compressed FNN.** Modify the weights and bias of Φ_2 by training Φ_2 using data set $\{\mathbf{x}_i^{\{0\}}, \hat{\mathbf{y}}_{i,2}^{\{L\}}\}$, $i = 1, \ldots, N$, under the loss function (21).
- **Evaluate re-training outcome.** After re-training Φ_2, we compute output reachable set $\tilde{\mathcal{Y}}^{\{L+1\}}$ and compares it with the target reachable domain. The repair process is finished only when (17) is satisfied. Otherwise, it repeats the repairing process.

Algorithm 1 summarizes the repairing process for FNN compression based on the equivalence evaluation of two FNNs. It keeps re-training the compressed FNN until the discrepancy meets the requirement. A timeout counter is set up to avoid the repair process falling into a dead loop. The computation of discrepancy follows Theorem 1 to compute and update the discrepancy between the original FNN and the updated compressed FNN. After the repair process, the relationship between reachable set $\tilde{\mathcal{Y}}^{\{L+1\}}$ and repairing target set $\mathcal{O}$ is used to evaluate whether the repair process is a success.

Algorithm 1: FNN Compression Repair

input : Original FNN Φ_1, Compressed FNN Φ_2, repairing target set $\mathcal{O}$
output: Repaired Compressed FNN $\hat{\Phi}_2$

while *True* **do**
  Compute discrepancy $\tilde{\delta}_{max}$
  Generate $\{\mathbf{x}_i^{\{0\}}, \mathbf{y}_{i,1}^{\{L\}}, \mathbf{y}_{i,2}^{\{L\}}\}$, $i = 1, \ldots, N$
  $\hat{\mathbf{y}}_{i,2}^{\{L\}} \leftarrow \mathbf{y}_{i,2}^{\{L\}} + \frac{1}{\alpha}\tilde{\delta}_{max}$
  $\Phi_2 \leftarrow \text{retrain}(\Phi_2, \text{Dataset}(\{\mathbf{x}_i^{\{0\}}, \hat{\mathbf{y}}_{i,2}^{\{L\}}\})$
  Compute reachable set $\tilde{\mathcal{Y}}^{\{L+1\}}$ for Φ_2
  if $\tilde{\mathcal{Y}}^{\{L+1\}} \subseteq \mathcal{O}$ *or timeout* **then**
    $\hat{\Phi}_2 \leftarrow \Phi_2$
    break
  end
end
return $\hat{\Phi}_2$

IV. Application to Compressed Feedforward Neural Networks Repairment

In this section, to validate the effectiveness of our proposed approach, we use the MNIST data set to perform our task. We apply our equivalence evaluation method on the two FNNs and the repair method to the compressed FNN to narrow down the discrepancy[1].

A. Database

MNIST [25] database contains a large number of handwritten digits and is famous as an image classification problem benchmark. The database contains 60,000 training images and 10,000 testing images. Each image is a $28\times28\times1$ grayscale image, and all images are classified into ten labels, from 0 to 9.

B. Experiment Set Up

We train an FNN with three layers: the first has 256 neurons, the second has 64 neurons, and the third has 10 neurons. Each hidden layer is followed by a ReLU activation function. The FNN is trained with the training dataset of MNIST for five epochs. As for the compressed FNN, we apply the quantization aware training (QAT) [26] method to the original FNN, shrinking down the parameter size of the network. Table I shows the comparison of the original network and the compressed network. Table II shows the

[1]The code for the experiment is available at github.com/aicpslab/FNN-repair

TABLE I

NETWORKS OVERVIEW

Network	Parameters	Size (KB)	Accuracy (%)
Original Network	218,058	855	98%
Compressed Network	218,058	226	91%
Repaired Network	218,058	226	98%

results with ten randomly picked images. The discrepancy is the mean of the maximum distance between the output value of the two FNNs among all label scores.

To repair the compressed FNN, we set up the target reachable domains $\mathcal{O}$ as two-thirds of the original discrepancy domains. Every iteration compares the last discrepancy domain with the target domains. If it is not within the desired area, a re-train dataset for compressed FNN is generated based on the last discrepancy domain to re-retrain for three epochs. According to (20), we use different $\alpha = 2, 5, 10, 20$. The re-train process will time out after ten iterations. Except for the ten randomly chosen images, we also randomly chose ten re-train samples from the images where the original FNN gives the wrong predictions.

C. Results

First, we perform the FNN repair with $\alpha = 10$ in (20). Table I is the overview of the three FNNs in the experiment. The original FNN has a 98% accuracy but drops to 91% after compression. However, compared to the size of the three FNNs, the compressed version's size is only one-fourth of the original one, which proves that compression helps decrease the scale of the neural network. Comparing the compressed FNN before repair and after repair, the performance of the FNN improves from 91% to 98% and almost achieves the same level as the original FNN, which our repairing method is able to improve the capability of the compressed network. In addition, Table II shows the discrepancy result of our repair method. The total mean discrepancies after repair are all lower than our target values for each testing input, and some are even only one-tenth of the original value, which demonstrates the effectiveness of our repair method while keeping the performance. Fig. 2 shows the repairing result in detail via a randomly chosen image. The input image is a handwritten digit "9". The blue dots are the scoring output of the original network, with the highest score at label 9. The green whisker bar line on each label represents the guaranteed output range of the compressed network before repair. Relatively, the red whisker bar line is the guaranteed output range of the network after repair. It is obvious that each red whisker line is closer to the blue dot than the green one, proving that the discrepancy is mitigated after repair.

To demonstrate the repairing process and indicate the different repair performances with different $\alpha = 2, 5, 10, 20$, we show the discrepancy reduction and accuracy increase along with the repairing process. The repairing process can reduce the average discrepancy between the two network outputs, as shown in Fig. 3. A small α has a larger discrepancy result because of the larger step size to the optimal value. A large α may not lower the discrepancy to a smaller value. As for the accuracy part, in Fig. 4, all α values can repair the compressed network to reach 98% accuracy, the same as the original network. Thus, our method does

TABLE II

DISCREPANCY RESULTS BETWEEN BEFORE AND AFTER REPAIR WITH RANDOM IMAGES

Input Image	Mean δ (Before)	Mean δ (After)
0	0.4750	0.0625
1	0.4189	0.0969
2	0.6339	0.1034
3	0.4746	0.0870
4	0.8272	0.1217
5	0.6200	0.1189
6	0.6458	0.0701
7	0.5440	0.1505
8	0.5908	0.0953
9	0.8283	0.1053

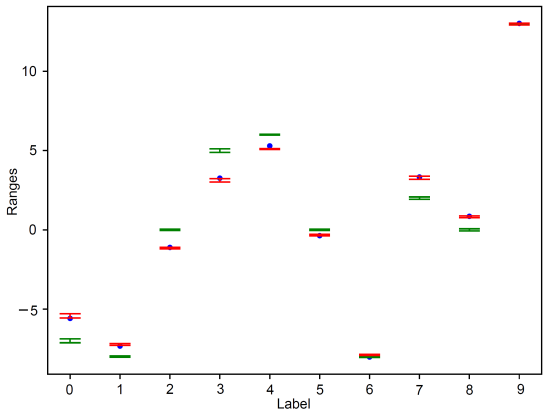

Fig. 2. Repair results with a handwritten digit "9". Blue dots are the output for the original network Φ_1. The green whisker line represents the output range of the compressed network Φ_2 before repair. The red whisker line represents the output range of the compressed network $\hat{\Phi}_2$ after repair. The outcome shows that the repaired network generates a more precise output range (red whisker lines) closer to the original outputs (blue dots).

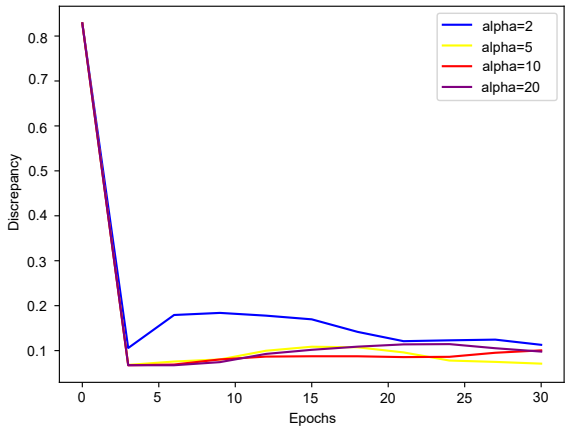

Fig. 3. Repair result with a handwritten digit "9". Different color lines are the average discrepancy of input images between the original network Φ_1 and compressed network Φ_2 with different α settings. The outcome shows that different α may lead to different repair performance, but the repair process can always decrease the discrepancy.

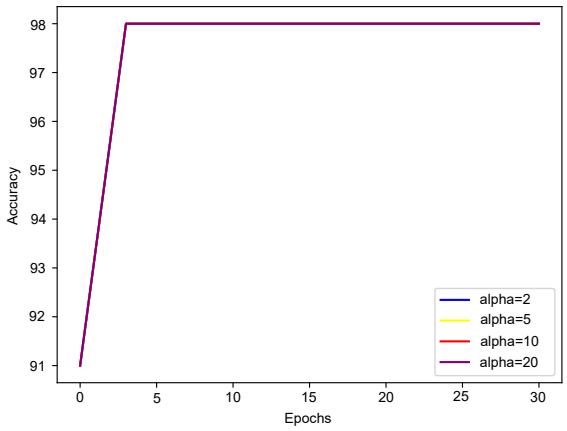

Fig. 4. Accuracy of the whole test set along with the repair process with different α settings. All α values can help the compressed network reach 98% accuracy in 3 epochs.

mitigate the discrepancy between the original network Φ_1 and the compressed network Φ_2. The α is also important to have a better repair performance, especially for complicated networks.

V. Conclusions

This work mainly proposes an approach to repair the compressed FNN based on the equivalence evaluation method. It formally defines the structure of the merged neural network with two given networks and develops reachability analysis methods to compute the reachable set of the discrepancy with the same input. The repair framework is explained in detail, such as the construction of the re-train dataset, the repair result criteria, and the compressed network update. Then, our approach successfully gives repair results between the original and compressed networks by showing the mean discrepancy before and after repair. The repair task is carried out by applying the discrepancy domain to the compressed network output to re-train the compressed network with randomly chosen samples, as shown by the MNIST experiment.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[2] S. Lawrence, C. Giles, A. C. Tsoi, and A. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[3] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012.

[5] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[6] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A guide to convolutional neural networks for computer vision," *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, pp. 1–207, 2018.

[7] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3119–3127, 2015.

[8] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marban, T. Marinč, D. Neumann, T. Nguyen, H. Schwarz, T. Wiegand, D. Marpe, and W. Samek, "Deepcabac: A universal compression algorithm for deep neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 700–714, 2020.

[9] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558*, 2020.

[10] M. P. Owen, A. Panken, R. Moss, L. Alvarez, and C. Leeper, "Acas xu: Integrated collision avoidance and detect and avoid capability for uas," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pp. 1–10, IEEE, 2019.

[11] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Computer Aided Verification: 29th International Conference, Heidelberg, Germany, July 24-28*, pp. 97–117, Springer, 2017.

[12] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5687–5695, 2017.

[13] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," in *International Conference on Learning Representations, ICLR Poster*, 2017.

[14] Y. Zhang, W. Ding, and C. Liu, "Summary of convolutional neural network compression technology," in *2019 IEEE International Conference on Unmanned Systems (ICUS)*, pp. 480–483, 2019.

[15] W. Xiang, H.-D. Tran, J. A. Rosenfeld, and T. T. Johnson, "Reachable set estimation and safety verification for piecewise linear systems with neural network controllers," in *2018 Annual American Control Conference (ACC)*, pp. 1574–1579, IEEE, 2018.

[16] W. Xiang and Z. Shao, "Approximate bisimulation relations for neural networks and application to assured neural network compression," in *2022 American Control Conference (ACC)*, pp. 3248–3253, IEEE, 2022.

[17] Y. Zhang and X. Xu, "Reachability analysis and safety verification of neural feedback systems via hybrid zonotopes," in *2023 American Control Conference (ACC)*, pp. 1915–1921, IEEE, 2023.

[18] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, "Fast and effective robustness certification," *Advances in neural information processing systems*, vol. 31, 2018.

[19] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[20] M. Henk, J. Richter-Gebert, and G. M. Ziegler, "Basic properties of convex polytopes," in *Handbook of discrete and computational geometry*, pp. 383–413, Chapman and Hall/CRC, 2017.

[21] H.-D. Tran, S. Bak, W. Xiang, and T. T. Johnson, "Verification of deep convolutional neural networks using imagestars," in *Computer Aided Verification: 32nd International Conference, Los Angeles, CA, USA, July 21–24*, pp. 18–42, Springer, 2020.

[22] H.-D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *International Conference on Computer Aided Verification*, pp. 3–17, Springer, 2020.

[23] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5777–5783, 2018.

[24] W. Xiang, H.-D. Tran, X. Yang, and T. T. Johnson, "Reachable set estimation for neural network control systems: A simulation-guided approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 1821–1830, 2021.

[25] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[26] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.