Understanding the Resilience of Neural Network Ensembles against Faulty Training Data

Abraham Chan, Niranjhana Narayanan, Arpan Gujarati, Karthik Pattabiraman, Sathish Gopalakrishnan
The University of British Columbia, Vancouver, BC, Canada
Email: {abrahamc, niranjhana}@ece.ubc.ca, arpanbg@cs.ubc.ca, {karthikp, sathish}@ece.ubc.ca

***Abstract*—Machine learning is becoming more prevalent in safety-critical systems like autonomous vehicles and medical imaging. Faulty training data, where data is either mislabelled, missing, or duplicated, can increase the chance of misclassification, resulting in serious consequences. In this paper, we evaluate the resilience of ML ensembles against faulty training data, in order to understand how to build better ensembles. To support our evaluation, we develop a fault injection framework to systematically mutate training data, and introduce two diversity metrics that capture the distribution and entropy of predicted labels. Our experiments find that ensemble learning is more resilient than any individual model and that high accuracy neural networks are not necessarily more resilient to faulty training data. Further, we find that simple majority voting suffices in most cases for resilience in ML ensembles. Finally, we observe diminishing returns for resilience as we increase the number of models in an ensemble. These findings can help machine learning developers build ensembles that are both more resilient and more efficient.**

***Index Terms*—Error resilience, Machine learning, Training**

I. Introduction

Machine learning (ML) is increasingly being used in safety-critical applications ranging from medical diagnosis [1] to autonomous driving [2]. Incorrect inferences by ML components in such applications can lead to disastrous outcomes. For example, if an ML component in an autonomous vehicle misclassifies a cracked pavement as a lane marking and steers the car into the wrong lane, it may lead to fatalities. Therefore, we need high accuracy in safety-critical ML applications.

Most ML components are developed using *supervised learning*, which is *data-driven* [3] because training datasets are used to construct models that support inferring an output or decision for a given input. There is little input from developers in the programming of ML components, apart from choosing the model's architecture and the training dataset. Training datasets, as collections of input-output pairs, are therefore integral to how ML components behave.

Model architectures are often chosen because they are known to yield high accuracy when trained on well-curated benchmark datasets. In practice, collecting high quality training datasets is an arduous task with potential for errors [4, 5]. For every input in the dataset, we need to indicate a suitable output, and the output, typically, is a label to associate with the input. Labelling tasks are often crowd-sourced, resulting in poorly curated training datasets with a large fraction of mislabelled data [6]. Even a quorum of domain experts may sometime label data incorrectly, owing to human perceptions and biases, or sensory limitations [7]. Further, if the datasets themselves are constructed using ML, inaccuracies of the underlying ML model carry forward as labelling faults [8]. Finally, errors during data cleaning and preprocessing may corrupt or remove large fractions of the training dataset [9].

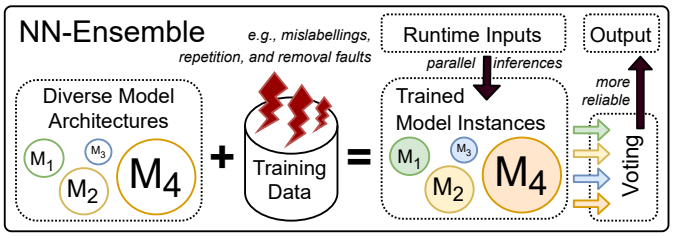


Fig. 1: NN-Ensemble trains N models independently using the same (faulty) training data. At runtime, it then uses mechanisms like simple majority, plurality, or total probability voting to combine the individual classification results.

We believe that eliminating all faults in training data when using a supervised learning approach is not practical. Therefore, it is inadequate to choose ML models under the assumption of fault-free training data - we also need to understand and improve how ML components perform under faulty training data. This is the focus of this work[1]

We explore the use of ensemble learning methods such as *stacking*, in which inferences from diverse neural networks (NNs) are combined to improve the overall accuracy of the model. We consider a simple approach that combines the predictions of multiple separately-trained NNs via voting only at the time of inference, as is common in many fault-tolerant safety-critical architectures (Fig. 1). For convenience, we refer to any NN-based ensemble learning architecture as *NN-Ensemble*. To the best of our knowledge, *we are the first to systematically analyze the resilience of ensemble learning across different NN architectures and datasets.*

Prior work on modeling faults in training data considers every inaccurate input-output pair as random *noise* and then attempts to nullify the effect of this noise by, for example, adding additional regularization layers in the network [7, 10]. Our study instead encompasses a broader set of training

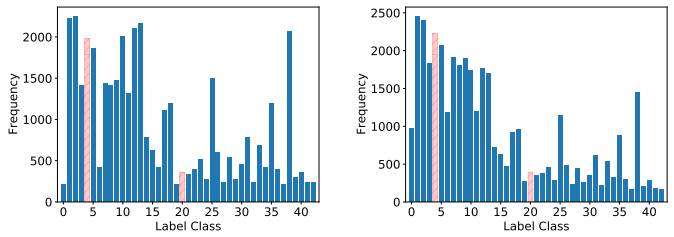
[1]Resilience, in this work, refers to the ability of an ML-based system to achieve high accuracy despite faults in the training data.

TABLE I: NN architectures used. † for CIFAR-10 only.

Name	Depth	Architecture Summary
NN	Shallow	2 FC w/ 0.2 Dropout
RNN	Shallow	2 LSTM (+ 1 Conv for GTSRB)*
CNN	Moderate	3 Conv + 1 FC + Max Pooling
LeNet	Moderate	2 Conv + 2 FC
ConvNet†	Moderate	3 Conv + 3 FC + Max Pooling
DeconvNet†	Moderate	4 Conv + 2 FC w/ 0.5 Dropout
AlexNet	Moderate	5 Conv + 2 FC + Avg Pooling
VGG3†	Moderate	6 Conv + 2 FC + Max Pooling
VGG16	Deep	13 Conv + 3 FC + Max Pooling
ResNet18†	Deep	17 Conv + 1 FC + Avg Pooling
MobileNet(v1)†	Deep	27 Conv + 1 FC + Avg Pooling
ResNet50	Deep	49 Conv + 1 FC + Avg Pooling

TABLE II: Image classification datasets used

Name	Dataset Size		Task (# Classes)
	Training	Test	
MNIST [12]	60,000	10,000	Handwritten digits (10)
CIFAR-10 [13]	50,000	10,000	Objects and animals (10)
GTSRB [14]	39,209	12,630	Traffic signs (43)



(a) Pristine dataset (b) Dataset with faults

(c) Test image (d) Label class 4 (e) Label class 20

Fig. 2: **(a, b)** Label class distributions without faults and with 30% mislabellings (respectively); **(c)** test image for class 4; and **(d, e)** reference images for classes 4 and 20 (respectively).

dataset faults. We consider *mislabelling faults* where data can be erroneously labelled, *repetition faults* where input-output pairs may be repeated, and *removal faults* where a fraction of data may be deleted. Together, they not only express the possibility of inaccurate input-output pairs, but also the possibility of one or more output classes being misrepresented, due to either insufficiencies in the data collection process or errors in the data processing pipeline. We characterize the effects of these fault on the prediction quality of different NN architectures and across different datasets, as well as evaluate the benefits of NN-Ensemble in tolerating such faults.

We make four **contributions** in this work:
① We have built a fault injection framework TF-DM (TensorFlow Data Mutator) (briefly introduced in a workshop article [11]) that systematically injects mislabelling, repetition, and removal faults in configurable amounts during training,
② We then characterize the resilience of NN-Ensemble for different datasets using the *Accuracy Delta (AD)* metric, which measures the drop in a faulty model's accuracy specifically with respect to inputs that are correctly classified by the pristine model.
③ We go beyond accuracy and analyze the diversity of predictions from different NN architectures using fine-grained metrics such as the *Gini coefficient* and the *Shannon equitability index*.[2]
④ We finally consider many diverse NNs with different depths and complexities and three different datasets (summarized in Tables I and II) including the safety-critical German Traffic Sign Recognition Benchmark (GTSRB), which makes this the first empirical study of its kind.

While we find that NN-Ensemble is more resilient than individual NN models, it is insufficient to understand ensembles by only considering their final outputs - we also analyze ensemble features like the prediction diversity, voting schemes and resilience by label class to find areas for improvement.

We find that models with high accuracy are not necessarily more resilient to faulty training data and that there are discrepancies in resilience between label classes. We also observe that plurality voting is as good as simple majority voting for resilience in NN-Ensemble. This demonstrates that different models typically do not converge on incorrect predictions, which highlights the value of ensembles for fault tolerance and detection. Finally, we find a diminishing return for resilience as we increase the number of models.

II. Motivating Example

We present a simple example that motivates NN-Ensemble. The German Traffic Sign Recognition Benchmark (GTSRB) [14] is a public dataset containing more than 50,000 images of 43 different traffic signs in Germany. It is used as a training dataset for Autonomous Vehicles (AVs). We inject the dataset with 30% random mislabelling faults. Fig. 2a and Fig. 2b show the label distributions in GTSRB before and after fault injection, respectively. The distributions are different.

We then train AlexNet on a pristine copy of GTSRB and on the faulty copy, resulting in trained models M and M_f, respectively. We test both these models with an image of a 70 km/h speed limit sign (Fig. 2c). The test image corresponds to label class 4 in the dataset. We find that while M correctly classifies the test image as a 70 km/h speed limit sign (Fig. 2d), M_f misclassifies it as a right curve sign (Fig. 2e), which belongs to label class 20 in the dataset. This misclassification can cause an AV travelling at 70 km/h to suddenly slow down and veer to its right, and potentially lead to an accident.

[2]These metrics are typically used to characterize income inequality among countries and species diversity in an ecosystem, respectively.

Suppose we use NN-Ensemble instead with a collection of seven NN models (Table I). All models are trained on the faulty dataset that is injected with 30% mislabelling faults, and majority voting is used for prediction. We observe that NN-Ensemble correctly classifies the test image despite being trained on the faulty dataset, as four out of these seven models correctly classify the test image, thereby tilting the majority in favour of the correct outcome.

Increasing the fault injection rate further, say, up to 50%, actually reverses the majority. Four out of the seven models now misclassify the speed sign. However, all incorrect predictions belong to different classes (*e.g.*, one model misclassifies the test image as a right curve sign, another model misclassifies it as a 20 km/h speed sign). NN-Ensemble can detect this disagreement and signal the autonomous driving agent, which in turn can alert the human driver. NN-Ensemble can thus avoid catastrophic consequences even in this scenario.

These misclassifications occurred despite using the exact same NN architectures trained on a lower quality version of the same dataset, demonstrating the serious impact inflicted by faulty training data. NN-Ensemble can provide protection from training data faults in NNs. However, it incurs costs in computation and power, which can be a deterring factor, since safety-critical control systems often have strict time constraints. System designers therefore need to decide if and when NN-Ensemble is desired, and how to select models that result in more resilient ensembles. The empirical analysis in this paper helps answer such questions.

III. Related Work

Work on noisy labels and other training dataset faults, ML resilience and fault injection, ensemble learning, and adversarial learning are all related to our work.

Noisy Labels and Other Faults while Training To deal with the adverse effects of faulty training data, ML researchers have proposed different solutions. For example, to address the noisy label problem, prior work has proposed tools for identifying mislabelled data during preprocessing [15, 16], specialized architectures where denoising layers are added [17], and regularization techniques that are specific to mislabelled data [7, 18]. To address other faults such as missing or unequal distribution of data classes, such as the problem of repetition in domain-specific datasets [5], prior work has proposed to modify the training data using resampling and clustering [19], or use mutation testing to evaluate the efficacy of test data [20]. In contrast, we consider a general fault model in our evaluation of whether ensembles built using off-the-shelf NN architectures can withstand faults, without any form of preprocessing.

ML Resilience and Fault Injection There has also been work on improving the resilience of ML components in the presence of different types of faults, such as hardware faults [21, 22], and in-the-wild distribution shifts in the input domain [23]. However, these papers do not typically consider faults in the training data. There are also a number of fault injectors developed for ML applications, such as Ares [24], PyTorchFI [25], and TensorFI [26]. These tools also primarily emulate only software and hardware faults, *e.g.*, by injecting faults into the *add* and *multiply* ML operators, whereas we perform fault injection in training datasets instead.

Ensemble Learning Ensemble learning is an ML approach that combines the results from multiple ML models to improve inference accuracy. Most ensemble implementations use the *bagging* approach where the same ML architectures are trained using randomized subsets of the training data [27–29] to generate variants. Unlike NN-Ensemble, this avoids the need for resource-intensive retraining of algorithmically different models, but the robustness of the ensemble is limited by the capabilities of the chosen model architecture. More recently, ensemble methods such as *stacking* have begun to consider diversity among different model architectures. Recent work shows that ensembles consisting of diverse neural architectures can offer improved inference accuracy over individual learners [30–32]. However, *we are the first to evaluate and understand why ensembles are resilient against training dataset faults. We analyze the resilience of constituent models instead of treating ensembles as a black box.*

Adversarial Learning Ensembles have also been used to defend ML systems against adversarial learning, where attackers use carefully-crafted inputs to trick ML models during inference. However, our fault model is fundamentally different from adversarial learning. We look at randomly introduced training dataset faults, which may result in faulty ML models post training, *i.e.*, models that are overfit or underfit for one or more output classes and therefore lead to misclassification of (otherwise correctly classified) test inputs. An example ensemble-based defense framework against adversarial learning is Athena by Meng et al. [33], which applies different sets of data transformations on test inputs prior to feeding them into individual models, and then combines the prediction results in an ensemble. Our tool TF-DM does not transform individual inputs, but instead transforms the entire dataset (through mislabellings, repetitions, and removals).

IV. Methodology

We begin by explaining how TF-DM injects faults into training datasets, and then present the metrics we use for reliability and diversity assessment.

A. Injecting Faults in Training Dataset

We have developed TF-DM [11], a fault-injection framework that injects mislabelling, repetition, and removal faults in labelled datasets. As mentioned in Section I, these fault classes simulate a wide range of realistic scenarios where faults are introduced into training datasets either via human errors or through faults in the underlying software libraries. We formally specify the fault injection process below[3] and use the running example with the GTSRB dataset, introduced in Section II, to explain each fault type.

Consider a labelled dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ and $\mathcal{Y}$ denote an ordered set of input samples and labels. Let $\mathcal{X}_i$

[3]This was not included in the workshop article [11].

denote the i^{th} input sample and $\mathcal{Y}_i$ denote its label. Each input sample $\mathcal{X}_i \in \mathbb{R}^n$ is a vector with n features and each label $\mathcal{Y}_i \in \{1, 2, \ldots, L\}$ identifies one of the L output classes. During fault injection, each fault type is parametrized using an *amount* parameter (expressed in percentage) that decides the quantum of faults injected. Based on $\mathcal{D}$ and *amount* values, TF-DM can inject three types of faults.

① **Mislabelling Faults** TF-DM overwrites a random subset of labels with incorrect labels. For example, a right turn road sign may be randomly mislabelled as a left turn sign. Suppose dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ consists of $|\mathcal{X}| = |\mathcal{Y}| = d$ entries. TF-DM chooses $d^f = d \times amount$ indices in $\{1, 2, \ldots d\}$ uniformly at random, which we denote as $\mathcal{I}^f$. TF-DM then transforms the dataset into a faulty dataset $\mathcal{D}^f = \{\mathcal{X}, \mathcal{Y}^f\}$, such that $|\mathcal{Y}^f| = |\mathcal{Y}|$ and

$$\begin{aligned} &\forall x \in \{1, 2, \ldots, d\} : \\ &\quad x \in \mathcal{I}^f \implies \mathcal{Y}^f_x \neq \mathcal{Y}_x \ \wedge \ x \notin \mathcal{I}^f \implies \mathcal{Y}^f_x = \mathcal{Y}_x. \end{aligned} \tag{1}$$

② **Repetition Faults** TF-DM repeats a random subset of input samples and corresponding labels in the dataset. For example, a unique image of a right turn road sign may be duplicated in the training dataset. Similar to mislabelling faults, it chooses $d^f = d \times amount$ random indices from $\{1, 2, \ldots d\}$, which we denote as $\mathcal{I}^f$. TF-DM then repeats the input samples and labels at these indices, resulting in a larger but faulty dataset $\mathcal{D}^f = \{\mathcal{X}^f, \mathcal{Y}^f\}$. The new dataset consists of $|\mathcal{X}^f| = |\mathcal{Y}^f| = d + d^f$ entries such that

$$\begin{aligned} &\forall x \in \{1, 2, \ldots, d\} : \mathcal{X}^f_x = \mathcal{X}_x \wedge \mathcal{Y}^f_x = \mathcal{Y}_x, \ \textit{and} \\ &\forall y \in \{1, 2, \ldots, d^f\} : \mathcal{X}^f_{d+y} = \mathcal{X}_{\mathcal{I}^f_y} \wedge \mathcal{Y}^f_{d+y} = \mathcal{Y}_{\mathcal{I}^f_y}. \end{aligned} \tag{2}$$

In the case of repetition faults, it is important to choose the *amount* value judiciously as a larger *amount* of repetitions may actually have a smaller impact on the ML model behaviour. For instance, if $amount = 99\%$, a vast majority of the training dataset is simply trained twice, resulting in negligible change in the trained ML model and its resulting behaviour. However, if $amount = 10\%$, the training process is biased towards the subset of training data that is repeated, thereby increasing the chances of overfitting by the model.

③ **Removal Faults** TF-DM deletes a fraction of the training dataset randomly. For instance, random images of right turn road signs may be deleted from the training dataset. It chooses a set of random indices $\mathcal{I}^f \subseteq \{1, 2, \ldots d\}$ as defined above, and then transforms the training dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ into a smaller dataset $\mathcal{D}^f = \{\mathcal{X}^f, \mathcal{Y}^f\}$, such that $|\mathcal{X}^f| = |\mathcal{Y}^f| = d - d^f$ and

$$\begin{aligned} &\forall x \in \{1, 2, \ldots, d\} : \\ &\quad x \in \mathcal{I}^f \implies \mathcal{X}_x \notin \mathcal{X}^f \wedge \mathcal{Y}_x \notin \mathcal{Y}^f \\ &\quad x \notin \mathcal{I}^f \implies \exists y \leq d - d^f : \mathcal{X}^f_y = \mathcal{X}_x \wedge \mathcal{Y}^f_y = \mathcal{Y}_x. \end{aligned} \tag{3}$$

B. Reliability Metrics

Accuracy is the most commonly used metric for measuring the efficacy of ML models. It is defined as the percentage of input samples in the test dataset that are correctly classified by the model. However, using accuracy to measure the reliability of the model in the presence of training dataset faults conflates two different properties: **(i)** the model architecture's ability to infer the input-output relationship in a fault-free scenario, and **(ii)** its ability to tolerate (training data) faults. We, therefore, define a new metric for reliability analysis: the *Accuracy Delta (AD)*. We define *AD* as the relative change in the faulty model's accuracy with respect to the original model. A higher value of AD implies lower resilience, and vice versa. For instance, suppose we have two models in an ensemble with 93% and 97% accuracy respectively. Assume both models are retrained on a faulty dataset, both resulting in 90% accuracy. While both models now have the same accuracy, the former model is more resilient than the latter, as its AD is much lower than that of the latter.

More formally, let M_{golden} and M_{faulty} denote two trained instances of the same model architecture M. Suppose that M_{golden} is trained on a pristine copy of dataset $\mathcal{D}$, whereas M_{faulty} is trained on a faulty version $\mathcal{D}^f$ of the dataset. For convenience, we refer to M_{golden} and any fault-free instances of M as *golden models*, and M_{golden}'s performance on the test dataset as the *golden accuracy* of M with respect to $\mathcal{D}$.

Let $\mathcal{D}'$ denote the test dataset consisting of test inputs $\mathcal{X}'$ and labels $\mathcal{Y}'$, defined similarly to $\mathcal{D}$ in Section IV-A above. We assume that $\mathcal{D}'$ is fault-free, *i.e.*, we do not inject faults into test datasets. Suppose that $M_{golden}(\mathcal{X}'_i)$ and $M_{faulty}(\mathcal{X}'_i)$ denote the classification outputs of the golden and faulty model instances for input sample $\mathcal{X}'_i$ in the test dataset. We define the AD of M_{faulty} as a ratio of the number of test inputs that are classified correctly by M_{golden} but misclassified by M_{faulty} to the total number of test inputs that are classified correctly by M_{golden}, *i.e.*,

$$\begin{aligned} &M_{faulty}\text{'s AD w.r.t. } \mathcal{D}' = \\ &\quad \frac{|\{\mathcal{X}'_i \mid M_{faulty}(\mathcal{X}'_i) \neq \mathcal{Y}'_i \ \textit{and} \ M_{golden}(\mathcal{X}'_i) = \mathcal{Y}'_i\}|}{|\{\mathcal{X}'_i \mid M_{golden}(\mathcal{X}'_i) = \mathcal{Y}'_i\}|}. \end{aligned} \tag{4}$$

Eq. (4) checks the bias when measuring the reliability of different faulty instances of the same model architecture. However, if we want to evaluate the AD of NN-Ensemble that uses N different model architectures, we need to get rid of all input samples in $\mathcal{D}'$ that are classified incorrectly by even one golden model. That is, if NN-Ensemble uses model architectures $M^1, M^2, \ldots, M^N$, we define its AD as follows.

$$\begin{aligned} &\textit{NN-Ensemble's AD w.r.t. } \mathcal{D}' = \\ &\quad \frac{\left|\left\{\mathcal{X}'_i \;\middle|\; \textit{NN-Ensemble}(\mathcal{X}'_i) \neq \mathcal{Y}'_i \ \textit{and} \ \bigwedge_{x=1}^{N} M^x_{golden}(\mathcal{X}'_i) = \mathcal{Y}'_i\right\}\right|}{\left|\left\{\mathcal{X}'_i \;\middle|\; \bigwedge_{x=1}^{N} M^x_{golden}(\mathcal{X}'_i) = \mathcal{Y}'_i\right\}\right|}. \end{aligned} \tag{5}$$

C. Diversity Metrics

Diversity among ML models can be measured in several ways; unfortunately, there is no single universally accepted definition [34, 35]. Furthermore, as mentioned in Section III,

prior work does not evaluate diversity in the presence of faults. Having concrete metrics for measuring diversity is important so that we can understand how to build more resilient ensembles. We explain below with an example why existing metrics used for ensemble learning such as *Yule's Q statistic* [36] are not suitable for measuring diversity for reliability goals. We then propose the use of the *Gini coefficient* and the *Shannon equitability index* as diversity metrics. We are the first (to the best of our knowledge) to use these metrics in ML reliability studies.

1) Oracle Outputs vs. Categorical-Valued Outputs: Earlier, we supposed that a model outputs one of the L output classes during inference, *i.e.*, $M(\mathcal{X}_i) \in \{1, 2, \ldots, L\}$. We refer to such outputs as *categorical-valued*. Alternatively, each model can simply output a binary value denoting whether its classification is *correct* or *incorrect* (also known as *oracle* outputs). Among the metrics we discuss, the Shannon equitability index relies on categorical-valued outputs, whereas Yule's Q statistic and Gini coefficient rely on oracle outputs (see Fig. 3). Even though oracle outputs cannot be determined at runtime, they can be easily determined for test datasets. We denote these using the oracle function $\mathcal{O}$, *i.e.*, $\mathcal{O}(M(\mathcal{X}_i)) \in \{correct = 1, incorrect = 0\}$.

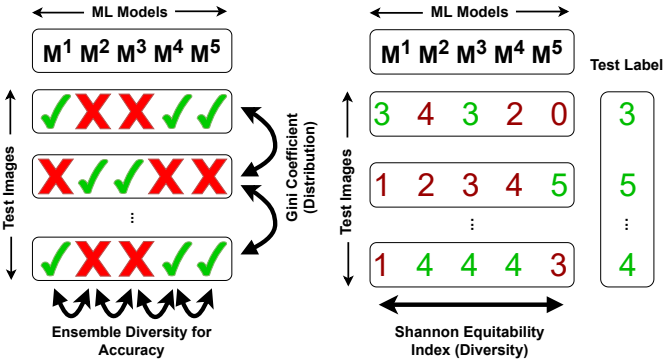
2) Yule's Q Statistic: Consider instances M^1 and M^2 of two distinct model architectures, trained on the same faulty dataset $\mathcal{D}^f$. Suppose that M^1 and M^2's outputs when tested on dataset $\mathcal{D}' = \{\mathcal{X}', \mathcal{Y}'\}$ are related using counters d^{11}, d^{10}, d^{01}, d^{00}, where $|\mathcal{X}'| = d^{11} + d^{10} + d^{01} + d^{00}$ and each $d^{ab} = |\{\mathcal{X}'_i \mid \mathcal{O}(M^1(\mathcal{X}'_i)) = a \text{ and } \mathcal{O}(M^2(\mathcal{X}'_i)) = b\}|$. The Q statistic for models M^1 and M^2 is then defined as

$$Q^{1,2} = (d^{11}d^{00} - d^{01}d^{10})/(d^{11}d^{00} + d^{01}d^{10}) \tag{6}$$

and varies between -1 and 1. $Q^{1,2} = 0$ implies that M^1 and M^2 are statistically independent, whereas $Q^{1,2} = -1$ or $Q^{1,2} = 1$ imply that M^1 and M^2 are positively or negatively correlated (respectively). For N different model architectures, ${}^NC_2$ pairwise Q statistics can be defined; the overall Q statistic denoted Q^{av} is defined as their average.

We show how the oracle outputs differ among three models M^1, M^2, and M^3 when these models are tested using datasets $\mathcal{D}'$ and $\mathcal{D}''$ (Table III (left)), and we also show the pairwise and average Q statistics based on these results (Table III (right)). Even though the models behave very differently when tested with datasets $\mathcal{D}'$ and $\mathcal{D}''$, the Q statistics are close to -1 or 1, indicating that the models are correlated. Therefore, diversity metrics like Yule's Q statistics and other similar pairwise metrics such as *double-fault measure* and *disagreement measure* [35] are less useful for reliability, particularly when choosing between different models.

3) Gini Coefficient: We could immediately infer two key properties (Table III). First, many output combinations occur with very low frequency, *e.g.*, all three models classify correctly only 3 out of 20,000 times. Secondly, different output combinations may or may not have a uniform representation,



(a) Oracle outputs (b) Class labels as outputs

Fig. 3: Differences between Oracle metrics (e.g., Yule's Q statistic, the Gini coefficient), and Shannon equitability index.

e.g., the case where M^2 and M^3 classify correctly but M^1 misclassifies occurs 19,400 out of 20,000 times. These insights can allow NN-Ensemble to assign greater weights to more trustworthy models (and vice versa). In this regard, we propose to use the Gini coefficient as a metric to measure model diversity in NN-Ensemble.

The Gini coefficient is conventionally used to measure the income inequality within a group of people [37]. If x_i is the income of person i, $\bar{x}$ is the average income, and P is the population size, then the Gini coefficient is given by:

$$G = \left(\sum_{i=1}^{P}\sum_{j=1}^{P}|x_i - x_j|\right) \Big/ 2P^2\bar{x}. \tag{7}$$

We compute the Gini coefficient for ML models using Eq. (7) as follows. Suppose N models $M^1, M^2, \ldots, M^N$ are tested using dataset $\mathcal{D}' = \{\mathcal{X}', \mathcal{Y}'\}$. Like Yule's Q statistic, we use the oracle outputs for computing G. For each input sample $\mathcal{X}'_k$, oracle outputs $\mathcal{O}(M^1(\mathcal{X}'_k)), \ldots, \mathcal{O}(M^N(\mathcal{X}'_k))$, form a binary vector belonging to $\{0, 1\}^N$. We consider this set $\{0, 1\}^N$ of all possible vectors as our population, and hence we use $P = 2^N$. We let each income parameter x_i correspond to the number of inputs samples in $\mathcal{X}'$ for which the oracle outputs form the i^{th} vector in $\{0, 1\}^N$.

The Gini coefficient varies from 0 to 1. Smaller values imply that the possibility of different output combinations is evenly distributed, whereas larger values imply that a few output combinations are predominant. For example, the Gini coefficients for models M^1, M^2, and M^3 in Table III are $G(\mathcal{D}') = 0.8612$ and $G(\mathcal{D}'') = 0.4997$, respectively. Unlike the Q statistic, these illustrate the difference between the two datasets. A higher value for $\mathcal{D}'$ indicates the presence of a dominant output combination, which could be used to reduce the number of models in NN-Ensemble. In contrast, a lower value for $\mathcal{D}''$ suggests the use of equally-weighted majority voting in NN-Ensemble as opposed to dropping models.

4) Shannon Equitability Index: While useful for characterizing the differences among models, the Gini coefficient relies on oracle outputs, and hence cannot account for the

TABLE III: Output frequencies (left) and statistics (right)

Oracle Outputs			Frequencies	
M^1	M^2	M^3	$\mathcal{D}'$	$\mathcal{D}''$
0	0	0	110	4990
0	0	1	2	2
0	1	0	240	5000
0	1	1	19400	5000
1	0	0	240	5000
1	0	1	2	2
1	1	0	3	3
1	1	1	3	3

Type	Statistics	
	$\mathcal{D}'$	$\mathcal{D}''$
$Q^{1,2}$	−0.9997	−0.9988
$Q^{2,3}$	0.9997	0.9992
$Q^{3,1}$	−0.9993	−0.9960
Q^{av}	−0.3331	−0.3319
G	0.8612	0.4997

TABLE IV: Model predictions (left) and distributions (right)

Models	Predictions	
	$\mathcal{X}'_1$ = "1"	$\mathcal{X}'_2$ = "3"
M_1	"1"	"0"
M_2	"1"	"1"
M_3	"1"	"3"
M_4	"1"	"2"
M_5	"2"	"6"
M_6	"4"	"4"
M_7	"3"	"5"

Labels	Distributions	
	$\mathcal{X}'_1$ = "1"	$\mathcal{X}'_2$ = "3"
"0"	0/7	1/7
"1"	4/7	1/7
"2"	1/7	1/7
"3"	1/7	1/7
"4"	1/7	1/7
"5"	0/7	1/7
"6"	0/7	1/7
"7"	0/7	0/7
"8"	0/7	0/7
"9"	0/7	0/7

contributions of different voting algorithms towards reliability. Therefore, we consider another fine-grained metric, the Shannon equitability index, to determine whether multiple models misclassify input samples similarly.

The Shannon equitability index is a popular metric to measure diversity of species in ecosystems [38], and is commonly denoted as H. If S denotes the total number of unique species in an ecosystem, and p_i denotes the proportion of the entire community made up of species i, then

$$H = -\left(\sum_{i=1}^{S} p_i \ln p_i\right) \Big/ \ln S. \tag{8}$$

We compute the Shannon equitability index for ML models as follows. Suppose N models $M^1, M^2, \ldots, M^N$ are tested using dataset $\mathcal{D}' = \{\mathcal{X}', \mathcal{Y}'\}$. Unlike the Gini coefficient, we use categorical-valued model outputs, which identify one of the class labels in $\{1, 2, \ldots, L\}$. For each input sample $\mathcal{X}'_k$, outputs $M^1(\mathcal{X}'_k), \ldots, M^N(\mathcal{X}'_k)$ form a vector belonging to $\{0, 1, \ldots, L\}^N$. This can be translated into a frequency distribution over the label set. In particular, we consider each label as a unique species, and hence denote $S = L$ in Eq. (8). For each species $i \in \{1, 2, \ldots, L\}$, we let $p_i = |\{j \mid M^j(\mathcal{X}'_k) = i\}|/N$. This allows us to compute the Shannon equitability index $H(\mathcal{X}'_k)$ for a single input sample.

For example, consider the MNIST dataset of handwritten digits and two test images $\mathcal{X}'_1$ = "1" and $\mathcal{X}'_2$ = "3". The predictions of seven different models $M^1, M^2, \ldots, M^7$ for each of these test images are summarized in Table IV (left). These are then translated into frequency distributions over the label set, as shown in Table IV (right). Using Eq. (8) and the approach outlined above, $H(\mathcal{X}'_1)$ and $H(\mathcal{X}'_2)$ are computed as follows ($S = 10$ as there are 10 digits in MNIST).

$$H(\mathcal{X}'_1) = -\frac{\frac{4}{7}\ln\frac{4}{7} + 3 \times (\frac{1}{7}\ln\frac{1}{7})}{\ln 10} = 0.50106, \tag{9}$$

$$\textit{and } H(\mathcal{X}'_2) = -\frac{7 \times (\frac{1}{7}\ln\frac{1}{7})}{\ln 10} = 0.84510. \tag{10}$$

The Shannon equitability index ranges between 0 and 1, where 0 represents no diversity and 1 represents maximum diversity. In the example above, the model outputs are much more diverse in the case of the test image $\mathcal{X}'_2$ = "3", which is also indicated by the higher value of $H(\mathcal{X}'_2)$ in Eq. (10).

In our experiments, we report the overall Shannon equitability index for any dataset $\mathcal{D}' = \{\mathcal{X}', \mathcal{Y}'\}$ by computing the average over all input-specific indices, *i.e.*,

$$H = \left(\sum_{i=1}^{|\mathcal{X}'|} H(\mathcal{X}'_i)\right) \Big/ |\mathcal{X}'|. \tag{11}$$

We also report conditional Shannon equitability indices that are computed over a subset of input samples satisfying a specific condition. We describe these in Section V-D.

V. Evaluation

In our evaluation, we answer five research questions (RQs) characterizing the behavior of NN-Ensemble and its constituent models in the presence of faulty training data.

- Is NN-Ensemble more resilient to faulty training datasets than individual ML models? (**RQ1**)
- How does inequality in the prediction outcomes across models contribute to NN-Ensemble's behaviour? (**RQ2**)
- Can we characterize the diversity of models in a way that correlates with NN-Ensemble's resilience? (**RQ3**)
- Does NN-Ensemble's resilience vary with label classes? (**RQ4**)
- Does increasing the number of models in NN-Ensemble improve its resilience? (**RQ5**)

These are answered in Sections V-B to V-F, respectively. We start by describing our experimental setup in Section V-A.

A. Experimental Setup

We used two machines for training and inference: **(i)** an Intel i7-10750H CPU with 8GB RAM and a Nvidia RTX 2600 GPU with 6GB VRAM; and **(ii)** an Intel i5-9300H CPU with 8GB RAM and a Nvidia GTX 1650 GPU with 4GB VRAM. We implemented all models in Python[4] using the Keras [39] API and trained and executed them using TensorFlow 2.4.1 [40]. We evaluated a total of 308 configurations across seven different models, three different datasets, and three different fault types, injected with different amounts of faults (see Tables I and II, and Section IV-A). For each configuration, we trained each of the seven models individually, but executed

[4]NN-Ensemble is available at https://github.com/DependableSystemsLab/NN-Ensemble

TABLE V: Golden accuracies (%) of ML models

Name	MNIST	GTSRB	Name	CIFAR-10
AlexNet	97.46	86.93	ConvNet	84.75
CNN	**99.00**	90.90	DeConvNet	86.81
LeNet	98.62	93.33	MobileNet	86.60
NN	97.45	84.68	ResNet18	90.24
ResNet50	97.97	88.57	ResNet50	**92.77**
RNN	94.06	71.50	VGG3	84.98
VGG16	97.68	**94.74**	VGG16	90.37

TABLE VI: Accuracy (%) of NN-Ensemble trained on faulty data. Values in parentheses denote the highest individual accuracy among the constituent models in the ensemble.

Fault Type	MNIST	CIFAR-10	GTSRB
Golden	98.79 (99.00)	90.28 (92.77)	92.40 (94.74)
Repetition (30%)	98.75 (98.96)	86.93 (84.57)	92.02 (93.21)
Removal (30%)	98.68 (97.05)	82.02 (79.35)	88.26 (86.43)
Mislabelling (30%)	97.62 (95.29)	79.93 (72.69)	91.61 (89.02)

the seven models in parallel during inference. To minimize variance in our results, we evaluated each configuration 20 times. As a result, the training required a total of 15.5 days, and the inference experiments required 4 hours in total.

For MNIST and GTSRB datasets, we used seven commonly used models: AlexNet, CNN, LeNet, NN, ResNet50, RNN, and VGG16. For CIFAR-10, to achieve comparable baseline accuracies, we used a different set of models that define the state-of-the-art for this dataset: ConvNet, DeconvNet, MobileNet, ResNet9, ResNet50 VGG3, VGG16.

Table V summarizes these models and their accuracies when trained using pristine datasets (*i.e.*, golden accuracies). Since these models differ both architecturally and algorithmically, *e.g.*, residual networks contain skip layers whereas some neural networks contain only convolutional layers, they may behave differently in the presence of training data faults. Therefore, this allows us to evaluate the diversity of off-the-shelf models, for fault tolerance using NN-Ensemble.

B. RQ1: Resilience of NN-Ensemble vs. Individual Models

We evaluate the resilience of individual models towards different fault types by measuring their respective ADs (Eq. (4)) after fault injection. Specifically, we injected faults in the datasets while varying the fault amounts from 10% to 80% for removal and mislabelling faults, and from 10% to 50% for repetition faults (recall from Section IV that injecting a high amount of repetitions may have negligible effects).

We focus on the results for CIFAR-10 and GTSRB (Fig. 4).[5]

ResNet50 and VGG16 have the highest accuracies among all models for CIFAR-10 and GTSRB (Table V), respectively. For CIFAR-10 (Figs. 4b and 4c), we find that deeper NNs with more layers (i.e., MobileNet, ResNet18, VGG16) incur higher AD than shallower models (i.e., ConvNet and DeconvNet). While deeper NNs generally give higher accuracies, they are also more prone to overfitting. However, we find that there are exceptions to this trend, particularly for repetition faults in Fig. 4a. One explanation is that the shallower NNs contain more fully connected layers compared to convolution layers. Fully connected layers tend to learn image patterns while convolution layers learn complex features. Therefore, repeated images have a higher likelihood to affect fully connected layers. In contrast, mislabelled images may inhibit convolution layers from learning correct features. For GTSRB (Figs. 4d to 4f), we also observe that shallower NNs are more resilient than deeper NNs. We also find that RNNs are generally the least resilient NN. This is because the RNN's feedback loops tend to increase the risk of overfitting.

In most runs, the AD increases with the amount of faults injected, as expected, except for repetition faults in Figs. 4a and 4d. As discussed in Section IV-A, 30% repetition faults might cause more overfitting than 50%. This effect is more evident in Residual Net (ResNet) models. ResNet models contain skip connections between layers, allowing neurons from one layer to jump over one or more subsequent layers - some neurons can bypass small numbers of overfitted layers.

Finally, we measured the AD of NN-Ensemble, as defined in Eq. (5), with majority voting in place. *NN-Ensemble always incurs a lower or a comparable AD than every other model in isolation.* The only exception we observed was when 80% of the dataset was injected with mislabelling faults, since in this case all models including NN-Ensemble had trouble classifying the test images. *We thus conclude that NN-Ensemble has higher resilience than individual models.*

While we evaluated resilience using the AD metric, we also report the accuracies of NN-Ensemble trained on golden and faulty training data (Table VI). NN-Ensemble achieves higher accuracy than individual models when trained with faulty data. However, NN-Ensemble is outperformed by the highest accuracy individual model for the golden cases and 30% repetition for MNIST and GTSRB. Under fault-free (and low AD) conditions, simple majority voting may not be the most optimal voting scheme. Nonetheless, NN-Ensemble still has a higher accuracy over most of its constituent models.

Observation 1 *Higher accuracy neural networks are not necessarily more resilient to faulty training data. Neural network architectures respond differently to faulty training data.*

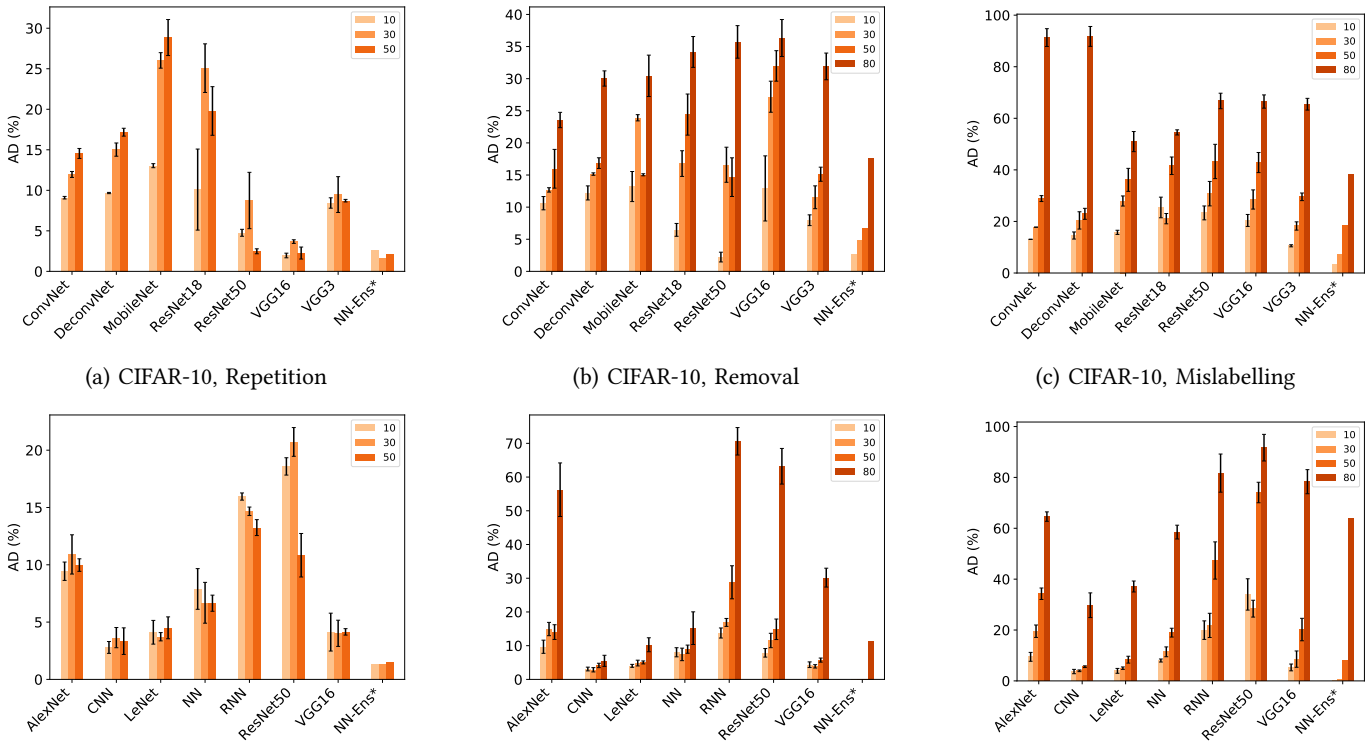
Observation 2 *NN-Ensemble has higher or comparable resilience than the most resilient individual model.*

C. RQ2: Inequality in Prediction Outcomes across Models

The AD experiments showed that NN-Ensemble is more resilient to training dataset faults than all individual models. To understand the reasons, we delve deeper into how the prediction outcomes of different models vary across the different images in the test dataset, when NN-Ensemble with simple majority voting is helpful, and when might other options such as weighted voting schemes be more beneficial.

Our first analysis was of the percentage of test images that are correctly classified by $N_{correct} = 0, 1, 2, \ldots$, or 7 models

[5]Due to space constraints, we do not show results for MNIST – we obtained results similar to the other two benchmarks.



(a) CIFAR-10, Repetition (b) CIFAR-10, Removal (c) CIFAR-10, Mislabelling

(d) GTSRB, Repetition (e) GTSRB, Removal (f) GTSRB, Mislabelling

Fig. 4: AD incurred by individual models and NN-Ensemble when trained with faulty CIFAR-10 and GTSRB datasets. The error bars in the results indicate the 95% confidence intervals. Lower values are better.

TABLE VII: Gini coefficients for different datasets

Fault Type	MNIST	CIFAR-10	GTSRB
Golden	0.975	0.824	0.895
Repetition (30%)	0.974	0.810	0.922
Removal (30%)	0.972	0.807	0.915
Mislabelling (30%)	0.960	0.774	0.894

(Fig. 5). The results explain why NN-Ensemble's resilience outperforms individual models. In most cases, a large fraction of test images falls to the right of the majority trendline. However, comparing the results for CIFAR-10 and GTSRB, we observe that different datasets may benefit differently from NN-Ensemble. For instance, in the case of GTSRB (Figs. 5e and 5f), NN-Ensemble is ineffective only when the fault amounts are very high (over 50%). In contrast, the majority trendline for CIFAR-10 grows linearly with the fault amount, and indicates that NN-Ensemble is less effective even when the fault amount is low (under 10%).

To further our understanding of the differences between NN-Ensemble's behavior for different datasets, we computed the Gini coefficients (Eq. (7)) for each dataset, for different fault types with a 30% fault rate (Table VII). Recall that larger values of the Gini coefficient indicate the presence of predominant output combinations, which is the case for both the MNIST and GTSRB datasets. These datasets may therefore do well with a reduced number of models in NN-Ensemble.

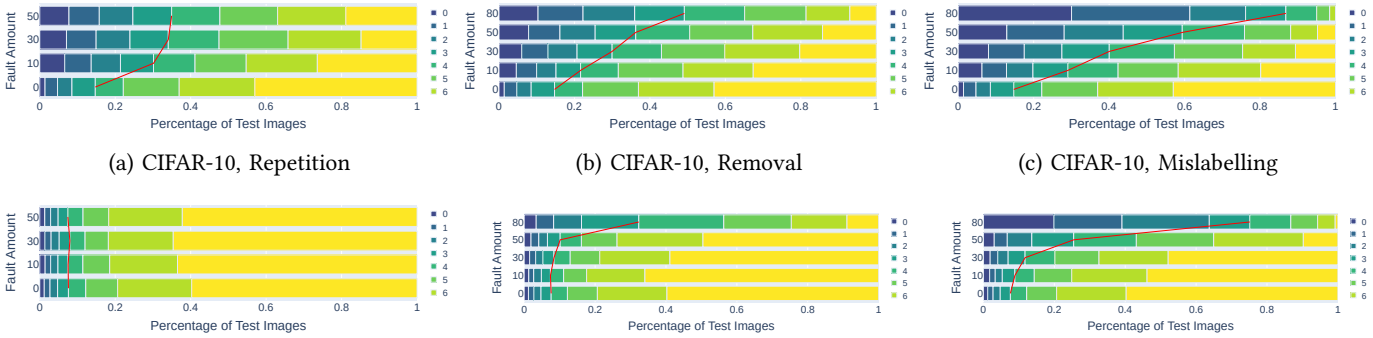
On the other hand, CIFAR-10 has lower Gini coefficients, indicating that output combinations across the models are more uniformly distributed. CIFAR-10 involves more difficult classification tasks than MNIST and GTSRB, especially since CIFAR images belong to a mix of animals, plants, vehicles, and other miscellaneous classes rather than being domain-specific. Nonetheless, NN-Ensemble is still effective in scenarios where fault amounts are less than 50% in CIFAR-10.

Observation 3 *NN-Ensemble with simple majority can correctly classify the test images in most configurations; alternate voting schemes may not provide much resilience improvement.*

D. RQ3: Diversity among Predicted Labels

Prediction outcomes for alternative voting schemes like plurality voting depend on whether incorrect classifications by different models are identical. We evaluate this aspect of NN-Ensemble using another diversity metric, the Shannon equitability index, since it relies on categorical-valued outputs, *i.e.*, it is a function of the label classes predicted by different ML models, as explained in Section IV-C4.

We computed the Shannon equitability indices for CIFAR-10 and GTSRB datasets with different types and varying amounts of faults (Table VIII). We report the overall index H as defined in Eq. (11). Additionally, we report four conditional indices that are also computed using Eq. (11) but over only a subset of all input samples that result in one of the following:



(a) CIFAR-10, Repetition (b) CIFAR-10, Removal (c) CIFAR-10, Mislabelling

(d) GTSRB, Repetition (e) GTSRB, Removal (f) GTSRB, Mislabelling

Fig. 5: Percentage of test images correctly classified by $N_{correct} = 0, 1, 2, \ldots, 7$ models. Shaded bars illustrate the value of $N_{correct}$. The darkest shade corresponds to $N_{correct} = 0$ (none of the models classify correctly) and the lightest shade corresponds to $N_{correct} = 7$ (all models classify correctly). The percentages are illustrated along the x axis. Results for different fault amounts are stacked along the y axis. The *trendline* in each graph indicates the majority boundary, *i.e.*, the percentage of test images for which $N_{correct} \geq 4$.

TABLE VIII: Shannon equitability indices for CIFAR-10 and GTSRB datasets with varying types and amounts of faults.

Type	%	CIFAR-10						GTSRB					
		H	H_{plural}	$\widehat{H}_{plural}$	H_{simple}	$\widehat{H}_{simple}$	AD	H	H_{plural}	$\widehat{H}_{plural}$	H_{simple}	$\widehat{H}_{simple}$	AD
Golden	-	0.19	0.14	0.39	0.12	0.41	0	0.08	0.06	0.34	0.06	0.35	0
Repetition	10	0.22	0.16	0.40	0.15	0.42	2.16	**0.05**	0.04	0.31	0.04	0.33	0.09
	30	0.30	0.25	0.45	0.22	0.47	1.63	**0.05**	0.04	0.32	0.04	0.33	0.03
	50	0.25	0.18	0.42	0.16	0.44	2.12	0.06	0.05	0.33	0.05	0.34	0.09
Removal	10	0.27	0.23	0.43	0.20	0.45	2.67	**0.05**	0.04	0.32	0.04	0.33	0.04
	30	0.29	0.24	0.44	0.21	0.46	4.79	0.07	0.05	0.32	0.05	0.33	0.19
	50	0.32	0.26	0.45	0.23	0.47	6.71	0.08	0.07	0.33	0.06	0.34	0.21
Mislabelling	10	0.28	0.21	0.44	0.19	0.46	3.31	0.07	0.06	0.32	0.05	0.34	0.13
	30	0.38	0.31	0.50	0.26	0.52	17.23	0.09	0.08	0.34	0.07	0.35	0.60
	50	**0.46**	0.39	0.54	0.32	0.55	38.55	0.19	0.17	0.37	0.15	0.37	8.23

- correct classification with simple majority (H_{simple}),
- incorrect classification with simple majority ($\widehat{H}_{simple}$),
- correct classification with plurality voting (H_{plural}), or
- incorrect classification with plurality voting ($\widehat{H}_{plural}$).

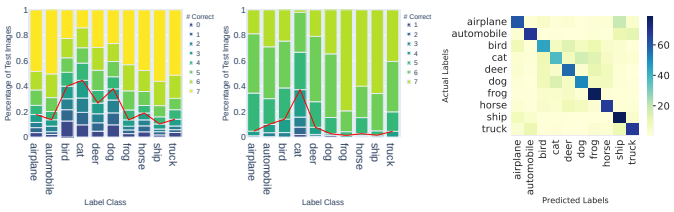
For instance, the conditional index $\widehat{H}_{plural}$ quantifies whether all N models in NN-Ensemble fail similarly or differently when plurality voting results in an incorrect classification.

The Shannon equitability indices range between 0 and 1, where a higher value denotes more diversity (Section IV-C4). H is highest for CIFAR-10 under mislabelling faults, and is lowest for GTSRB under repetition and removal faults. This corroborates our results in Section V-C that it is difficult for models to reach consensus in the former scenario, leading to incorrect classification, whereas models tend to achieve consensus in the latter scenario leading to correct classification. H is thus a good indicator of resilience.

To establish high confidence in the results, all correct classifications must be backed by a low diversity score, *i.e.*, both H_{simple} and H_{plural} should ideally be minimal. In addition, there should not be any high-confidence incorrect classification, which can be ensured if all the corresponding diversity scores are high. In other words, $\widehat{H}_{simple}$ and $\widehat{H}_{plural}$ should ideally both be maximized.

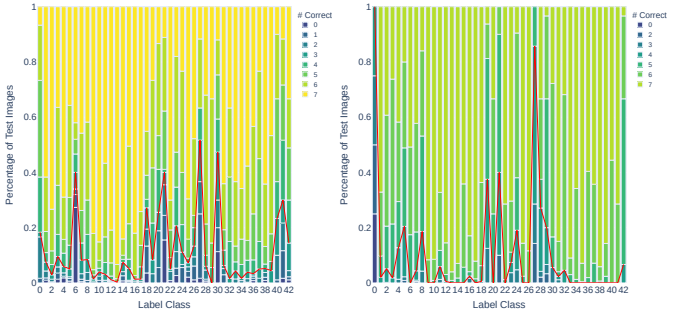
We observe a large and consistent difference of more than 0.2 between H_* and $\widehat{H}_*$ for both simple majority and plurality voting across all fault types and amounts for both the CIFAR-10 and GTSRB datasets (Table VIII). This suggests that NN-Ensemble is a useful tool when high-confidence predictions are needed for safety-critical domains. We also find a negligible difference between simple majority and plurality voting, which implies that both are equally good.

Finally, we also determine if the H values and AD are correlated (Table VIII). For each dataset, we compute the coefficient of determination, R^2, to evaluate the degree of linear correlation between two variables. R^2 ranges between 0 (no fit between two variables) and 1 (perfect fit between two variables). These were 0.11, 0.83, and 0.78 for MNIST, CIFAR-10 and GTSRB, respectively. Since MNIST's AD are extremely low (less than 10%), they do not correlate well with H. For every other dataset, the AD values exhibit high (positive) correlation with H. This show that consensus opportunities in NN-Ensemble diminish when the AD of the



(a) No faults (b) 30% removals (c) Confusion matrix

Fig. 6: Percentage of CIFAR-10 test images that are correctly classified by N models in NN-Ensemble, illustrated by label class, without faults (a) and with removals (b). Trendline indicates majority boundary. Shaded bars represent the percentage of test images that are correctly classified by $N_{correct} = 0, 1, 2, \ldots, 7$ models. The confusion matrix for the faulty case is illustrated in (c).



(a) GTSRB, no faults (b) GTSRB, 30% mislabelling

Fig. 7: Same as Figs. 6a and 6b, but for the GTSRB dataset.

constituent models increase (i.e., they are less resilient), and NN-Ensemble is unlikely to agree on incorrect classifications. Consequently, NN-Ensemble is likely to detect the faults.

Observation 4 *Plurality voting is as good as simple majority voting (and vice versa) for resilience.*

Observation 5 *Diversity among predicted labels, computed using the Shannon equitability index, correlates with resilience.*

E. RQ4: Resilience by Label Class

We analyze how NN-Ensemble's resilience varies with label class. We evaluated CIFAR-10 and GTSRB for 30% removal and mislabelling faults, respectively (Figs. 6 and 7). We found similar results for repetition faults, and omit them.

We examined how the prediction outcomes for CIFAR-10 vary across models before fault injection (*i.e.*, during the golden run). Classes "bird", "cat", "deer", and "dog" incur a high number of incorrect classifications relative to other classes (Fig. 6a). We then injected removal faults (30%) in the CIFAR-10 dataset and again measured the class-wise prediction outcomes, but only on the subset of test images that were correctly classified by all models in the golden scenario (similar to how we measured AD). The "cat" class is the least resilient to removal faults (Fig. 6b). On the other hand, classes like "dog" are quite resilient despite incurring high number of incorrect classifications earlier. Therefore, there is no correlation between the accuracy of a label class and its resilience. This observation also holds for imbalanced datasets like GTSRB. In GTSRB (Figs. 7a and 7b), prior to fault injection, NN-Ensemble incurs maximum incorrect classifications for classes 6, 18, 21, 27, 30, and 41. After injecting mislabelling faults (30%), the classes 0, 19, 21, and 27 are the least resilient. Note that TF-DM's training data faults impact all label classes equally prior to inference, and all the experiments are repeated over multiple runs.

We attempt to understand why certain classes are disproportionately impacted by faulty training data by examining the confusion matrix of CIFAR-10 after fault injection, shown in Fig. 6c. The confusion matrix plots the percentage of correct classifications by NN-Ensemble along the diagonal, and the percentage of misclassifications in the off-diagonal elements, where a darker shade represents a higher percentage. The x-axis is the predicted label, while the y-axis is the actual label. We observe that "cat" images are mostly likely to be misclassified as "dog" and "frog". One option to boost resilience is to increase the number of differentiating training examples between "cat" and the other two classes. Another option is to select and train models in the ensemble so that the classification accuracy of specific classes is improved. We find that the Gini coefficient of NN-Ensemble when "cat" is misclassified is 0.858 compared to 0.6, the average over other classes. This means that it is more likely that the same combination of models misclassify "cat" than the other classes. An ensemble consisting of some specialists rather than all generalists may be one way to address this disparity.

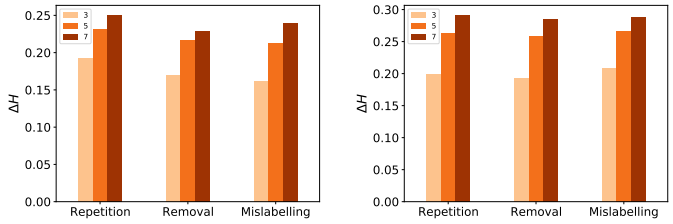
Observation 6 *NN-Ensemble's resilience varies with label classes, and this is independent of its accuracy per label class.*

F. RQ5: Do More Models Yield Better Resilience?

Thus far, we evaluated NN-Ensemble using seven different models (Table I). In this section, we determine if fewer models are sufficient. Recall from Section V-D that NN-Ensemble should generally have a high $\widehat{H}_*$ but low H_* to be resilient. We thus use the difference between these two indices as an indication of NN-Ensemble's resilience.

We compute $\Delta H = \widehat{H}_{simple} - H_{simple}$ for CIFAR-10 and GTSRB with $N = 3, 5, 7$ (Fig. 8), at 30% faults for each fault type. We use the top N most resilient ML models in NN-Ensemble, based on average AD. For example, in GTSRB at $N = 3$, we use CNN, LeNet, and VGG16. At $N = 5$, we add AlexNet and NN.

We find that ΔH is smaller for smaller values of N. The increase in ΔH is more prominent when N is increased from $N = 3$ to $N = 5$ than from $N = 5$ to $N = 7$. This is because both H_{simple} and $\widehat{H}_{simple}$ are low when N is small, but $\widehat{H}_{simple}$ is higher for larger values of N, making ΔH larger. However, as N increases and less resilient models are included in the mix, there is a diminishing return on ΔH.



(a) CIFAR-10 (b) GTSRB

Fig. 8: $\Delta H = \widehat{H}_{simple} - H_{simple}$ for $N = 3, 5, 7$. 30% faults.

Observation 7 *NN-Ensemble is more resilient if more models are used. However, there are diminishing returns as more models with high or comparable AD are added to NN-Ensemble.*

VI. Discussion

Limitations Our evaluations are limited by the size of our dataset. We have used TF-DM to inject faults in the training data set and retrain models from scratch. For large datasets like ImageNet [41], the training time could take up to two weeks per configuration on standard hardware [42]. To avoid this training time overhead, transfer learning has been used by other work (*i.e.*, models are initialized with pre-trained weights, and training is performed over fewer epochs). However, we have *not* examined the effect of faulty training data on transfer learning, and hence do not use it.

Further, NN-Ensemble requires multiple inference models, which demands additional resources. Designers thus need to balance the consequences of failures with operational costs.

Finally, the models we have used are algorithmically and architecturally distinct from each other. This approach necessitates retraining all models – the weights are not reusable.

Implications We find that different features of neural network architectures (*i.e.*, convolution layers, fully connected layers, skip connections) respond uniquely to different types of training dataset faults. Training dataset faults, however, do not occur in isolation (*e.g.*, repetition can be combined with mislabelling faults). An ensemble provides superior resilience compared to individual models, since no one model is resilient to all fault types.

In situations when time-constrained decisions are needed, we could run all models in parallel and then vote. On the other hand, we may want to conserve energy and execute additional models only when necessary (for instance, when two models disagree we can use additional models for inference). Diversity metrics like the Gini coefficient and Shannon equitability index are useful for understanding the value of diverse models, which could guide us in making such choices.

One of the more powerful implications of our work is that NN-Ensemble is valuable when the data distribution is different at deployment time relative to the training data. Such differences can be modelled as training data faults, and NN-Ensemble can mitigate them. Moreover, even when NN-Ensemble may not be able to recover from a disagreement among the different versions, it is able to detect a problem. In a safety-critical system, such a disagreement can trigger a system-level mode transition to a safe(r) controller, which can then shift the system to a fail-safe mode [43].

Better NN Architectures or Training Data? We show that faulty training data can cause ML models, even with state-of-the-art NN architectures, to incur large AD and degrade their classification accuracy. For instance, when a training dataset is imbalanced and has multiple copies of the same input-output pair, then the ML model may overfit to the data that is represented in excess of the normal occurrence of that input. This raises the question of whether we should invest in more resilient NN architectures (model-driven ML) or higher quality training data (data-driven ML) [44].

Ensemble of Specialists vs Generalists NN-Ensemble is constructed using general purpose NN models. We find that NN-Ensemble has weaker classification abilities in certain label classes (Section V-E). We believe that adding specialist models could mitigate this issue. The specialist models could be trained using more examples from specific label classes in the dataset. To track their success, developers could compute class-specific ΔH and check whether it has increased with the addition of specialist models in the ensemble. Care must be taken to ensure that the addition of specialist models does not degrade the overall accuracy of the ensemble. This idea has been briefly explored in the context of strengthening ensemble defences against adversarial examples [45].

Using Second Choices We considered taking the second choices of individual models into account. We were motivated by observations that (i) there was general consistency between second choice predictions - disagreement between second choices could indicate a fault and (ii) the second choice prediction was often correct when the first choice was incorrect. Unfortunately, we found that the ΔH values were very low (i.e., 0.05 on average) for second choices, meaning they would not be useful for misclassification detection using simple majority voting. Nonetheless, there is still useful information embedded in the second choice predictions, which can be extracted using alternative voting schemes or dark knowledge distillation [46]. This is an area for future work.

VII. Conclusions and Future Work

We evaluated NN-Ensemble, which applies ensemble learning to improve the resilience of machine learning (ML) components against training data faults. We build TF-DM to methodically inject faults in training data and evaluate NN-Ensemble. We find that (1) accuracy does not necessarily correlate with resilience, (2) NN-Ensemble can significantly improve resilience over individual models, and (3) correct decisions can be made in NN-Ensemble with high consensus using simple majority voting, and (4) there is diminishing return in resilience with increasing number of models in NN-Ensemble.

In the future, we plan to explore more efficient ways of generating ML diversity that minimize both the training and inference times. We also plan to apply NN-Ensemble to real-world safety-critical systems.

VIII. Acknowledgements

We thank the anonymous reviewers for their valuable comments. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and a Four Year Fellowship from UBC.

References

[1] J. G. Richens, C. M. Lee, and S. Johri, "Improving the accuracy of medical diagnosis with causal machine learning," *Nature Communications*, vol. 11, no. 1, p. 3923, 2020.

[2] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Hands Off the Wheel in Autonomous Vehicles?: A Systems Perspective on over a Million Miles of Field Data," in *Proc. of DSN'18*, 2018.

[3] R. Salay, R. Queiroz, and K. Czarnecki, "An Analysis of ISO 26262: Machine Learning and Safety in Automotive Software," in *SAE Technical Paper*, 2018.

[4] C. G. Northcutt, A. Athalye, and J. Mueller, "Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks," 2021, arXiv:2103.14749.

[5] B. L. Sturm, "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use," *CoRR*, 2013, arXiv:1306.1461.

[6] J. C. Chang, S. Amershi, and E. Kamar, "Revolt: Collaborative Crowdsourcing for Labeling Machine Learning Datasets," in *Proc. of CHI'17*, 2017.

[7] D. Karimi, H. Dou, S. K. Warfield, and A. Gholipour, "Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis," 2020, arXiv:1912.02911.

[8] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid Training Data Creation with Weak Supervision," *The VLDB Journal*, vol. 29, no. 2, pp. 709–730, 2020.

[9] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, "Data Cleaning: Overview and Emerging Challenges," in *Proc. of SIGMOD'16*, 2016.

[10] B. Frénay and M. Verleysen, "Classification in the Presence of Label Noise: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2013.

[11] N. Narayanan and K. Pattabiraman, "TF-DM: Tool for Studying ML Model Resilience to Data Faults," in *Proc. of DeepTest*, 2021.

[12] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[13] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[14] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. – , 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[15] P. Ostyakov *et al.*, "Label Denoising with Large Ensembles of Heterogeneous Neural Networks," 2019, arXiv:1809.04403.

[16] C. E. Brodley and M. A. Friedl, "Identifying and Eliminating Mislabeled Training Instances," in *Proc. of AAAI'96*, 1996.

[17] A. J. Bekker and J. Goldberger, "Training deep neural-networks based on unreliable labels," in *Proc. of ICASSP'16*, 2016.

[18] X. Xia, T. Liu, B. Han, C. Gong, N. Wang, Z. Ge, and Y. Chang, "Robust Early-Learning: Hindering the Memorization of Noisy Labels," in *Proc. of ICLR'21*, 2021.

[19] N. Rout, D. Mishra, and M. K. Mallick, "Handling Imbalanced Data: A Survey," in *Proc. of ASISA'18*. Springer, 2018.

[20] L. Ma *et al.*, "Deepmutation: Mutation testing of deep learning systems," *CoRR*, 2018, arXiv:1805.05206.

[21] M. Beyer, A. Morozov, E. Valiev, C. Schorn, L. Gauerhof, K. Ding, and K. Janschek, "Fault Injectors for TensorFlow: Evaluation of the Impact of Random Hardware Faults on Deep CNNs," 2020, arXiv:2012.07037.

[22] G. Li, K. Pattabiraman, and N. DeBardeleben, "TensorFI: A Configurable Fault Injector for TensorFlow Applications," in *Proc. of ISSREW'18*, 2018.

[23] P. W. Koh, S. Sagawa, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, T. Lee *et al.*, "WILDS: A benchmark of in-the-Wild Distribution Shifts," in *Proc. of ICML'21*, 2021.

[24] B. Reagen *et al.*, "Ares: A Framework for Quantifying the Resilience of Deep Neural Networks," in *Proc. of DAC '18*, 2018.

[25] A. Mahmoud *et al.*, "PyTorchFI: A Runtime Perturbation Tool for DNNs," in *Proc. of DSN-W'20*, 2020.

[26] G. Li, K. Pattabiraman, and N. DeBardeleben, "TensorFI: A Configurable Fault Injector for TensorFlow Applications," in *Proc. of ISSREW'18*, 2018.

[27] A. Wasay, B. T. Hentschel, Y. Liao, S. Chen, and S. Idreos, "MotherNets: Rapid Deep Ensemble Learning," in *MLSys*, 2020, arXiv:1809.04270.

[28] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra, "Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks," 2015, arXiv:1511.06314.

[29] L. Breiman, "Stacked regressions," *Machine Learning*, vol. 24, pp. 49–64, 1996.

[30] C. M. Farrelly, "Deep vs. Diverse Architectures for Classification Problems," 2017, arXiv:1708.06347.

[31] C. Ju, A. Bibaut, and M. J. van der Laan, "The Relative Performance of Ensemble Methods with Deep Convolutional Neural Networks for Image Classification," 2017, arXiv:1704.01664.

[32] M. A. Hedeya, A. H. Eid, and R. F. Abdel-Kader, "A Super-Learner Ensemble of Deep Networks for Vehicle-Type Classification," *IEEE Access*, vol. 8, pp. 98 266–98 280, 2020.

[33] Y. Meng, J. Su, J. O'Kane, and P. Jamshidi, "ATHENA: A Framework based on Diverse Weak Defenses for Building Adversarial Defense," 2020.

[34] E. Tang, P. Suganthan, and X. Yao, "An analysis of diversity measures," *Machine Learning*, vol. 65, pp. 247–271, 10 2006.

[35] L. Kuncheva and C. Whitaker, "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy," *Machine Learning*, vol. 51, pp. 181–207, 05 2003.

[36] G. U. Yule, "On the Association of Attributes in Statistics," *Philosophical Transactions of the Royal Society of London*, vol. 194, pp. 257–319, 1900.

[37] G. Pyatt, "On the Interpretation and Disaggregation of Gini Coefficients," *The Economic Journal*, vol. 86, no. 342, pp. 243–255, 1976.

[38] R. K. Peet, "Relative Diversity Indices," *Ecology*, vol. 56, no. 2, pp. 496–498, 1975.

[39] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[40] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proc. of CVPR'09*, 2009.

[42] L. V. Lakshmanan, M. Munn, and S. Robinson, *Machine Learning Design Patterns*, 2020.

[43] L. Sha *et al.*, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[44] P. Liu, L. Wang, G. He, and L. Zhao, "A Survey on Active Deep Learning: From Model-driven to Data-driven," 2021, arXiv:2101.09933.

[45] M. Abbasi and C. Gagné, "Robustness to Adversarial Examples through an Ensemble of Specialists," 2017, arXiv:1702.06856.

[46] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," 2015, arXiv:1503.02531.