On the Diversity of Machine Learning Models for System Reliability

Fumio Machida
Department of Computer Science, University of Tsukuba
Tsukuba, Japan
machida@cs.tsukuba.ac.jp

Abstract— **The diversity of system components is one of the important contributing factors of reliable and secure software systems. In a software fault-tolerant system using diverse versions of software components, a component failure caused by defects or malicious attacks can be covered by other versions. Machine learning systems can also benefit from such a multi-version approach to improve the system reliability. Nevertheless, there are few studies addressing this issue. In this paper, we experimentally analyze how outputs of machine learning modules can be diversified by using different versions of machine learning algorithms, neural network architectures and perturbated input data. The experiments are conducted on image classification tasks of MNIST data set and Belgian Traffic Sign data set. Different neural network architectures, support vector machines and random forests are used for constructing diverse machine learning models. The diversity is characterized by the coverage of errors over the test samples. We observe that the different machine learning models have quite different error coverages that can be leveraged for system reliability design. Based on the experimental results, we construct the reliability model for three-version machine learning architecture with a diversity measure defined as the intersection of error spaces in the sample space. From the presented reliability model, we derive a necessary condition under which three-version architecture achieves a higher system reliability than a single machine learning module.**

Keywords— Diversity, image classification, machine learning, reliability, software fault-tolerance

I. INTRODUCTION

Machine learning is becoming an important building block of many intelligent software systems. A wide variety of machine learning algorithms are used for extracting the features of training data set and constructing machine learning models for the tasks such as regression and classification. In particular, deep neural networks [1] have been widely adopted in many practical applications implementing functions like image classification, voice recognition and machine translation. Although a machine learning model takes just a small portion of the whole software system [2], it often provides the core of intelligence and hence any incorrect outputs of the machine learning model may cause undesirable system reliability.

Unlike a software component whose input-output relation is specified explicitly in advance, the output of a machine learning model is quite uncertain as it depends on the data and the algorithm used in the training process. It is extremely difficult to fully guarantee correct outputs of machine learning models in a production environment. Therefore, in a design of a software system employing machine learning models, it is important to premise the errors in the outputs from the machine learning modules and take relevant measures to mask such errors toward improving system reliability.

To design a reliable machine learning systems, in this paper we investigate the diversity of outputs from different machine learning modules. The study is motivated from N-version programming that is a well-established software fault tolerant technique originally proposed in the nineteen-seventies [3][4]. The essence of N-version programming is exploiting diversity of software components by using different implementations from the same original specification. With multi-version software implementations, even when one software component outputs an error due to a defect in the implementation, another software version which does not have the same defect can mask the error. Machine learning system can borrow this idea to improve the reliability of the system output. Although the output of a machine learning model for an unknown input is uncertain, the uncertainty can be a source of diversity that potentially contributes to mask the errors of the outputs. It is an interesting research question how machine learning models for the same task can be diversified by varying machine learning algorithms, choice of hyper-parameters or configurations, and input data for predictions. While many studies in machine learning algorithm focus on maximizing the accuracy by reducing classification or prediction errors, less work looks into the diversity of error outputs of different machine learning models.

In this paper, first we experimentally analyze the diversity of machine learning models in terms of error outputs by using different algorithms, neural network architectures and varied input data. In the experiments, we use two data sets, namely MNIST handwritten digit data set [5] and Belgian Traffic Sign data set [6]. Both the data sets are available online and widely used for benchmarking of machine learning algorithms for image classification tasks. For machine learning algorithms, we use a support vector machine, random forests and some different types of neural networks. After constructing the models for the classification tasks from the same training data sets, we compare the prediction errors from the different models using the test data sets. The objective of comparison is not to find the best classifier, but to understand the difference of capabilities of individual models. To this end, we compute the coverage of errors that is defined as the ratio of the test samples that cannot be accurately classified by a given set of machine learning models. The coverage of errors increases by using multiple machine learning models that have different classification capabilities. For MNIST data sets, we observe that the coverage of errors reaches 99.34% by using three different machine learning models, while the standalone best single classifier's coverage is 98.91%. We also confirm that by using different neural network architectures and different input data sets the coverages reach 99.71% and 99.58%, respectively. Similar results are obtained for Belgian Traffic Sign data set as well. We also consider the negative influences of the diversity on the certainty of predictions. The number of samples that are accurately predicted by all the machine learning models decreases by using different prediction results. The experimental results show that the increased coverage of errors is gained with the cost of the reduced certainty of accurate predictions.

With the experimental results, next we analytically show the potential reliability enhancement by employing an N-version machine learning architecture that uses three different machine learning modules with majority voting. Intuitively, the N-version architecture can effectively leverage the diversity of machine learning models toward an improved system reliability. However, our analysis on experimental results does not agree with this expectation. To analyze this, we construct a reliability model for a three-version machine learning system with a diversity measure. From this model, we derive the necessary condition under which the reliability of the three-version architecture overcomes the reliability of the best machine learning module. When the condition does not hold, the majority voting is likely to become a bottleneck for reliability enhancement, even though the coverage of errors are increased by the different machine learning models.

Our findings can be summarized as the following guides for engineering reliable machine learning systems.

1. Outputs of machine learning modules can be diversified by using perturbated input data (e.g., adding a noise, shifting pixels etc.) without using multiple machine learning models. Our experimental results show that the diversified outputs by perturbated input data can gain the increased coverage of errors. Since this approach can be tested with a relatively small cost, it is worth considering before preparing multiple machine learning models.
2. Diverse versions of machine learning models are primarily useful for detecting prediction errors in a machine learning system. We observe that the coverage of errors are increased by adding different prediction results from diverse machine learning models in most cases. Although the certainty of accurate predictions may decrease by considering different prediction results, the increased coverage is particularly important for safety critical tasks such as stop sign recognition in an autonomous vehicle.
3. An N-version architecture with three machine learning models with a majority voting does not always improve the system reliability compared with the standalone best machine learning model. Our reliability model considering the intersections of error spaces shows the necessary condition where the three-version architecture can achieve better reliability than the best machine learning model.

The remainder of the paper is organized as follows. Section II comprehends the diversity of machine learning models in terms of system reliability and categorizes the factors of diversity. Section III shows the results of experimental study on machine learning algorithms for image classification tasks. We show how different machine learning models output errors differently and statistically characterize the diversity of the models. Section IV provides the analytical reliability model for a three-version machine learning architecture and shows the necessary condition for potential reliability enhancement. Section V discusses related work and Section VI gives our conclusion.

II. Diversity of machine learning models

In this section, we introduce the concept of diversity for reliable software systems using machine learning modules. One of the techniques to exploit the diversity for improving the software system reliability is N-version programming that uses multiple generations of functionally equivalent programs from the same initial specification [3][4]. Machine learning systems can also benefit from the diversity of the components for designing the system reliability. On one hand, similar to software components, different implementations of machine learning algorithms can potentially mask the errors caused by software faults. On the other hand, the behavior of machine learning modules can also be diversified by machine learning algorithms, configurations, training data and input data sets. This paper focuses on the latter case and looks into how we can diversify the outputs of machine learning modules using different approaches. The potential contributing factors to improve the diversity of machine learning modules are explained below.

A. Training data

Unlike a software component which is implemented from the initial specification, a machine learning model is constructed from observed data without any functional specifications. Both of the amount and the quality of training data highly impact on the functional behavior of the generated machine learning model. This means that a subtle difference of training data set can cause a big difference of the functional behavior of the machine learning model. We can obtain different versions of machine learning modules by using different training data sets for training the models. In practice, however, all the available data are effectively used for generating the best machine learning model. Ensemble learning [24] is a commonly adopted machine learning technique that uses weak learning algorithms and random sampling from the training data set. When such a technique is used in the training process, it can be regarded that the diversity of training data is already exploited for the purpose of creating a reliable machine learning module.

B. Machine learning algorithm

Even from the same training data set, the functional behavior of machine learning models can be diversified considerably by using different machine learning algorithms. There are a number of machine learning algorithms available for tasks such as classification, recognition, regression and prediction. Individual algorithms have thier own theory and different characteristics. Although recently deep learning appears to mostly outperform other algorithms, there is no theoretical guarantee that deep learning generally achieves the best performance for any kind of tasks. Even when an approach achieves the best performance in a specific task, there is a possibility that other algorithms have different error coverages as we will see in our experiments in the following section. Therefore, different machine learning algorithms can contribute to generate different versions of machine learning models for the same task.

Two special sub-classes of the diversity factor in machine learning algorithm are hyper parameters and neural network architecture as explained below.

1) Hyper parameters

Machine learning algorithms often have parameters to be tuned for generating more accurate models. These parameters are called hyper parameters, since they determine the process of learning algorithm such as the number of iterations, the threshold to complete the process and the batch size to process the data. The best combination of the hyper parameter values which can generate the most accurate model is not known a priori. Therefore, how to effectively search the best parameter values is an important research challenge. It should be noted that however the best model found by a parameter search method is not always the superset of all the other models. Even when the most accurate model outputs error by a certain input, any less accurate model may not have the same error by the same input data. Error coverages of the models generated by different hyper parameter values can be different. Thus, the choice of hyper parameter values can also contribute to the diversity of machine learning models.

2) *Architecture of neural networks*

A wide variety of neural network architectures have been presented in particular after the success of deep neural network. Neural network is a type of machine learning method that imitates the human brain neural process for learning. A neural network consists of layers of neurons each of which has some connections to the adjacent layers' neurons with some weights. An architecture of neural network represents the connections of neurons as well as the numbers and the types of layers. Similar to hyper parameters, the best architecture for a specific problem is not known a priori, and hence architectural search techniques are actively investigated [7][8]. Since the error outputs of neural networks constructed from different architectures can be different, the choice of architecture is also considered as a contributing factor of the diversity. We will experimentally examine this in Section III.

C. *Input data for prediction*

Training data set and machine learning algorithm are the key sources of machine learning models. The output of obtained model, however, is quite sensitive to the input data for prediction as well. It is known that a subtle perturbation of input data can easily confuse a machine learning model to output error. Recently such a problem is referred to as adversarial example [9] and actively investigated in the research community [10][11][12]. For example, just one pixel change over the image of red traffic light on the road can cause mis-classification as green right with over 90% confidence [13]. Since the input data around the border of classes may have such characteristics, opposite can also happen. It means that just a subtle perturbation of input data can flip an error case to a correct output. Machine learning modules receiving different input data from the same source can have different error coverages due to their sensitivities to input data. We can diversify the output of machine learning modules by varying input data in the operation. It should be noted that our goal is to improve system reliability in operation instead of training the best machine learning model. Diversifying the input data is particularly effective when we only have a best trained machine learning model and no means to add another model for the same task. The approach can also be employed after deployment of the model on the target system because we can simply insert a data preprocessing stage before the process of machine learning model to perturbate the outputs. The impact of input data diversity is experimentally examined in the next section.

III. Experimental study

In this section, we experimentally show the diversity of machine learning models for image classification tasks that will potentially increase the classification error coverage. Our experiments are based on comparative evaluations of various machine learning models with different configurations and varied input data. The main objective of comparative evaluation is not on the benchmark of different machine learning models, but on characterizing the difference of error spaces of input data by various machine learning models. To quantitatively understand the diversity of machine learning models, we evaluate the coverage of errors over the test samples, which is formally defined in Section III-A.

The experiments are conducted on the image classification tasks for MNIST handwritten digit data set [5] and Belgian Traffic Sign data set [6]. MNIST data set is a well-known image classification benchmark that is composed of 28*28 pixels images for digits. 60000 examples are contained in the training data set and 10000 examples are included in the testing data set. On the other hand, Belgian Traffic Sign data set is a real world traffic sign image data set that can be divided into 62 different types of signs. 4591 examples are available as the training data set and additional 2534 examples are provided as the testing data set. Since the original images of Belgian Traffic Sign data are formatted in different sizes, we normalized the image data in 32*32 pixels. For machine learning algorithms, we employ random forests [14], support vector machine [15] and artificial neural networks [16]. Random forests use a random sampling of training data and generate multiple decision trees. For classification tasks, a majority voting of decision trees is taken. Compared to other state-of-art algorithms, both of the training and inference runs very fast. Support vector machine is a supervised machine learning algorithm exploring a hyperplane that has the largest distance to the nearest training-data point of any class. Since the obtained hyperplane has high generalization capability, the accuracy of classification is generally good. Artificial neural network is a computational model inspired by the neural structure of animal's brain. In this paper, we particularly use convolutional neural network (CNN) and multilayer perceptron that can be trained by a back-propagation algorithm. Due to the good classification performance, neural networks are now widely used for image classification, voice recognition and machine translation applications.

A. *Diversity affected metrics*

To investigate the diversity of machine learning models, we characterize the subset of sample space for individual machine learning models that can cause classification errors. We refer to this subset as *error space* of a machine learning model. Individual machine learning models may have different error spaces. Thereby, the classification error by a machine learning model can be masked if another machine learning model can classify the same sample correctly. To quantify the increased error masking capability by diverse machine learning models, we introduce the *coverage of errors* as defined below.

Definition: *Coverage of errors*
Given a set of machine learning models $\mathcal{M} = \{m_1, m_2, \dots\}$, let E_i be the observed error space of a machine learning model m_i for the test samples S. The coverage of errors is defined as

$$\mathrm{Cov}(\mathcal{M}) = 1 - \frac{\left|\bigcap_{m_i \in \mathcal{M}} E_i\right|}{|S|}.$$

The coverage values reach one when any test sample can be classified correctly at least by a machine learning model. Note that the coverage of errors for a single machine learning model is equal to the accuracy of the model over the test examples. The coverage value can be increased by adding other machine learning models that have different error spaces.

The diversity of machine learning models also influences on the uncertainty of prediction results. Even when a machine

TABLE I. NUMBER OF CLASSIFICATION ERRORS BY DIFFERENT MACHINE LEARNING MODELS

Label	0	1	2	3	4	5	6	7	8	9	Total
$\lvert S\rvert$	980	1135	1032	1010	982	892	958	1028	974	1009	10000
$\lvert E_{\mathrm{CNN}}\rvert$	3	6	11	3	5	9	22	11	11	28	109
$\lvert E_{\mathrm{RF}}\rvert$	10	13	36	34	26	30	19	37	41	47	293
$\lvert E_{\mathrm{SVM}}\rvert$	11	12	26	27	32	42	25	39	40	42	296

TABLE II. ERROR COVERAGES AND CERTAINTY OF PREDICTIONS BY ADDING PREDICTION RESULTS FROM DIFFERENT MACHINE LEARNING MODELS

Label	0	1	2	3	4	5	6	7	8	9	Total
$\mathrm{Cov}(\mathrm{CNN})$	0.9969	0.9947	0.9893	0.9970	0.9949	0.9899	0.9770	0.9893	0.9887	0.9722	0.9891
$\mathrm{Cov}(\mathrm{CNN},\mathrm{RF})$	0.9980	0.9965	0.9903	0.9980	0.9949	0.9922	0.9854	0.9912	0.9908	0.9802	0.9918
$\mathrm{Cov}(\mathrm{CNN},\mathrm{SVC})$	0.9980	0.9965	0.9942	0.9980	0.9969	0.9944	0.9843	0.9912	0.9928	0.9802	0.9927
$\mathrm{Cov}(\mathrm{CNN},\mathrm{RF},\mathrm{SVC})$	0.9980	0.9974	0.9942	0.9980	0.9969	0.9944	0.9864	0.9932	0.9928	0.9822	0.9934
$\mathrm{Cer}(\mathrm{CNN},\mathrm{RF})$	0.9888	0.9868	0.9641	0.9653	0.9735	0.9641	0.9718	0.9621	0.9559	0.9455	0.9680
$\mathrm{Cer}(\mathrm{CNN},\mathrm{SVC})$	0.9878	0.9877	0.9700	0.9723	0.9654	0.9484	0.9666	0.9601	0.9548	0.9504	0.9668
$\mathrm{Cer}(\mathrm{CNN},\mathrm{RF},\mathrm{SVC})$	0.9847	0.9833	0.9525	0.9525	0.9582	0.9395	0.9614	0.9465	0.9343	0.9435	0.9561

learning model predicts the label correctly, the result can become uncertain if another prediction result from different machine learning model does not agree on the correct label. By adding prediction results from diverse machine learning models, the number of samples that are correctly predicted by all the models decreases. Since this negative impact of diversity also needs to be analyzed, we introduce the *certainty of accurate prediction* which is defined as below.

Definition: *Certainty of accurate prediction*
Given a set of machine learning models $\mathcal{M} = \{m_1, m_2, \dots\}$, let E_i be the observed error space of a machine learning model m_i for the test samples S. The certainty of accurate prediction is defined as

$$\mathrm{Cer}(\mathcal{M}) = 1 - \frac{\left|\bigcup_{m_i \in \mathcal{M}} E_i\right|}{|S|}.$$

The certainty of accurate prediction represents the ratio of test samples whose labels are correctly predicted by all the machine learning models in $\mathcal{M}$. The certainty of accurate prediction for a single machine learning model is equal to the accuracy of the model. The certainty value decreases by adding different outputs from machine learning models that has different error spaces.

B. Algorithm diversity

First, we evaluate the diversity of machine learning models generated from three machine learning algorithms using MNIST data set. We use scikit-learn [17] for the implementation of random forest (RF) and support vector machine (SVM). The parameters of random forest are chosen by a grid search method that selects the best performed parameter value set. For SVM, we use support vector classifier and set the parameter gamma and C to 0.001 and 100, respectively. While each pixel of original image data ranges from 0 to 255 which represents the grayscale, we use digitized values for training and testing the support vector classifier. For artificial neural network, we use Keras [18] implementation of CNN and configure the network with a convolutional layer, a max pooling layer and a fully-connected layer as shown in Figure 1, which we follow the network visualization by Keras. Categorical cross entropy is used for loss function and Adam is used for optimizer. In the training of the CNN, the batch size is set to 128 and the final model is obtained after ten epochs. After building three models with 60000 of training samples, we predict the labels of testing data set and compare the predicted labels with correct labels.

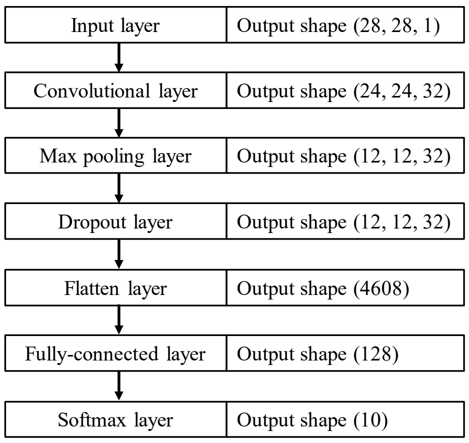

Figure 1. CNN architecture used in the experiments

TABLE I shows the number of classification errors of individual digits by three algorithms. We denote $|E_{\mathrm{X}}|$ as the number of errors by the model X that is either CNN, RF or SVM. As can be seen, while the number of samples differs among the digits (varying from 892 to 1135), CNN achieves the smallest classification errors for all the digits. This result only, CNN is considered as the best classifier among three models in terms of the label prediction accuracy. Then, our next question is how the coverage of errors can be improved by adding the prediction results from the different models. TABLE II shows the improved error coverages by combining CNN with RF, SVM and both. For all the digits, the coverages of errors are improved by adding the prediction results of other models. This means that even when CNN incorrectly predicts the labels for some test samples, the other models can predict the labels correctly. The error coverage is maximized when three models are used together (See the values of the line $\mathrm{Cov}(\mathrm{CNN},\mathrm{RF},\mathrm{SVC})$). The total error coverage reaches 0.9934, whereas the best coverage achieved by CNN is 0.9891. The result clearly shows that the different models have different error spaces that can potentially enhance the error masking capability. Meanwhile, diverse machine learning models also cause the increase in the total error space. In TABLE II, the decreased certainties of accurate predictions are summarized as well. When we use the prediction results from three models, the certainty of correct prediction for total samples decreases down to 0.9561, whereas the original accuracy of CNN is 0.9891. This means that about 4.4% of test samples there are disagreement among the prediction results from the different models.

TABLE III. NUMBER OF CLASSIFICATION ERRORS BY DIFFERENT NEURAL NETWORKS

Label	0	1	2	3	4	5	6	7	8	9	Total
$\lvert E_{\text{CNN}}\rvert$	3	6	11	3	5	9	22	11	11	28	109
$\lvert E_{\text{Dense}}\rvert$	9	6	12	13	21	19	11	19	22	23	155
$\lvert E_{\text{Expand}}\rvert$	2	9	4	8	12	9	16	11	7	11	89

TABLE IV. ERROR COVERAGES AND CERTAINTY OF PREDICTIONS BY ADDING PREDICTION RESULTS FROM DIFFERENT NEURAL NETWORKS

Label	0	1	2	3	4	5	6	7	8	9	Total
Cov(CNN)	0.9969	0.9947	0.9893	0.9970	0.9949	0.9899	0.9770	0.9893	0.9887	0.9722	0.9891
Cov(CNN, Dense)	0.9969	0.9974	0.9952	0.9980	0.9969	0.9933	0.9906	0.9922	0.9938	0.9891	0.9944
Cov(CNN, Expand)	0.9990	0.9974	0.9971	0.9980	0.9959	0.9933	0.9875	0.9942	0.9949	0.9960	0.9954
$\text{Cov(CNN, Dense, Expand)}$	0.9990	0.9982	0.9981	0.9990	0.9969	0.9955	0.9937	0.9951	0.9969	0.9980	0.9971
Cer(CNN, Dense)	0.9908	0.9921	0.9826	0.9861	0.9766	0.9753	0.9749	0.9786	0.9723	0.9604	0.9792
Cer(CNN, Expand)	0.9959	0.9894	0.9884	0.9911	0.9868	0.9865	0.9729	0.9844	0.9867	0.9653	0.9848
$\text{Cer(CNN, Dense, Expand)}$	0.9908	0.9877	0.9816	0.9842	0.9715	0.9720	0.9708	0.9767	0.9702	0.9584	0.9766

For the classification of the images labeled "0", we visualize the difference and overlaps of error spaces of three models in Figure 2. Only two out of 980 samples cannot be accurately classified by any models (i.e., $\lvert E_{\text{CNN}} \cap E_{\text{RF}} \cap E_{\text{SVC}}\rvert = 2$). Except for the depicted fifteen samples (in $E_{\text{CNN}} \cup E_{\text{RF}} \cup E_{\text{SVC}}$), three models agree with the correct label prediction.

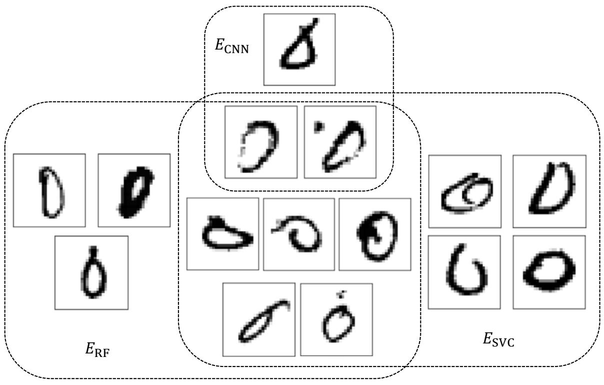

Figure 2. Error spaces of three machine learning models for classification of test samples labeled "0"

C. *Architectural diversity*

Next, we focus on neural network and use different architectures to test whether the error of coverages can be improved by varying network architectures. In addition to the CNN used in the previous section, we introduce two different architectures of neural networks as shown in Figure 3 in the visualization format by Keras. Dense network consists of two fully-connected layers and it does not have any convolutional layers. Expand network extends the original CNN with another convolutional layer, a max pooling layer and a fully connected layer. These neural networks are trained in the same configurations used for the original CNN.

TABLE III shows the number of classification errors observed for the same test samples by three neural networks. Both of CNN and Expand network achieve good classification accuracy. Nevertheless, there are still some prediction errors to be corrected. We investigate the coverage of errors by combining the prediction results of three neural network architectures. The coverage of errors is summarized in TABLE IV. The results clearly show that the coverages of errors are improved for all the labels by adding the prediction results with the different neural network architectures. The total error coverage reaches 0.9971 by using three architectures. For the classification of the test samples labeled "0", only one example remains uncovered by the predictions by three networks (i.e., $\lvert E_{\text{CNN}} \cap E_{\text{Dense}} \cap E_{\text{Expand}}\rvert = 1$).

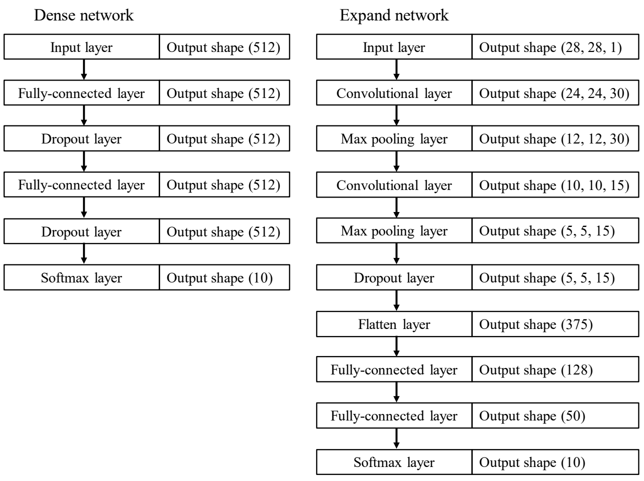
Dense network
Input layer | Output shape (512)
Fully-connected layer | Output shape (512)
Dropout layer | Output shape (512)
Fully-connected layer | Output shape (512)
Dropout layer | Output shape (512)
Softmax layer | Output shape (10)
Expand network
Input layer | Output shape (28, 28, 1)
Convolutional layer | Output shape (24, 24, 30)
Max pooling layer | Output shape (12, 12, 30)
Convolutional layer | Output shape (10, 10, 15)
Max pooling layer | Output shape (5, 5, 15)
Dropout layer | Output shape (5, 5, 15)
Flatten layer | Output shape (375)
Fully-connected layer | Output shape (128)
Fully-connected layer | Output shape (50)
Softmax layer | Output shape (10)

Figure 3. Architecture of dense network and expand network

Figure 4 shows the overlaps of error spaces of three neural networks for the test samples labeled "0". Since E_{Dense} is a superset of E_{CNN} and E_{Expand}, Dense network does not improve the coverage of errors against CNN and Expand. On the other hand, since E_{Expand} contains a sample which is not in E_{CNN}, Expand network improves the coverage of errors against CNN.

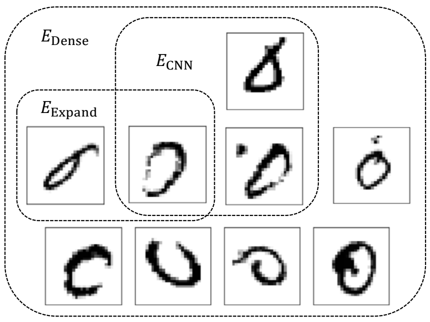

Figure 4. Error spaces of three neural networks for classification of test samples labeled "0"

TABLE IV also shows the decreased certainties of accurate predictions. By adding the prediction results from the different neural networks, the union of error spaces enlarge that causes decreased certainties. When we use three neural networks, the certainty is reduced to 0.9766, which is not so significant as the case of three different algorithms observed

TABLE V. NUMBER OF CLASSIFICATION ERRORS BY CNN WITH PERTURBATED TEST DATA

Label	0	1	2	3	4	5	6	7	8	9	Total
$\lvert E_{\mathrm{CNN,o}}\rvert$	3	6	11	3	5	9	22	11	11	28	109
$\lvert E_{\mathrm{CNN,s}}\rvert$	35	85	58	18	20	21	52	18	32	54	393
$\lvert E_{\mathrm{CNN,r}}\rvert$	5	47	70	19	105	24	104	147	57	113	691
$\lvert E_{\mathrm{CNN,n}}\rvert$	8	8	11	3	6	8	29	17	9	29	128

TABLE VI. ERROR COVERAGES AND CERTAINTY OF PREDICTIONS BY ADDING PREDICTION RESULTS WITH PERTURBATED TEST DATA

Label	0	1	2	3	4	5	6	7	8	9	Total
$\mathrm{Cov}(\mathrm{CNN},\{\mathrm{o}\})$	0.9969	0.9947	0.9893	0.9970	0.9949	0.9899	0.9770	0.9893	0.9887	0.9722	0.9891
$\mathrm{Cov}(\mathrm{CNN},\{\mathrm{o,s}\})$	0.9980	0.9956	0.9932	0.9980	0.9969	0.9933	0.9823	0.9942	0.9918	0.9861	0.9930
$\mathrm{Cov}(\mathrm{CNN},\{\mathrm{o,r}\})$	0.9990	0.9974	0.9922	1.0000	0.9949	0.9944	0.9802	0.9932	0.9938	0.9782	0.9924
$\mathrm{Cov}(\mathrm{CNN},\{\mathrm{o,n}\})$	0.9980	0.9974	0.9922	0.9970	0.9969	0.9910	0.9781	0.9893	0.9918	0.9742	0.9907
$\mathrm{Cov}(\mathrm{CNN},\{\mathrm{o,s,r,n}\})$	1.0000	0.9991	0.9981	1.0000	0.9980	0.9955	0.9843	0.9961	0.9959	0.9891	0.9957
$\mathrm{Cer}(\mathrm{CNN},\{\mathrm{o,s}\})$	0.9633	0.9242	0.9399	0.9812	0.9776	0.9731	0.9405	0.9776	0.9641	0.9326	0.9568
$\mathrm{Cer}(\mathrm{CNN},\{\mathrm{o,r}\})$	0.9929	0.9559	0.9293	0.9782	0.8931	0.9686	0.8883	0.8531	0.9363	0.8821	0.9276
$\mathrm{Cer}(\mathrm{CNN},\{\mathrm{o,n}\})$	0.9908	0.9903	0.9864	0.9970	0.9919	0.9899	0.9687	0.9835	0.9877	0.9693	0.9856
$\mathrm{Cer}(\mathrm{CNN},\{\mathrm{o,s,r,n}\})$	0.9551	0.8934	0.8934	0.9644	0.8829	0.9552	0.8747	0.8463	0.9158	0.8543	0.9027

in TABLE II. Since three neural networks achieve good prediction accuracy alone, the combination of them does not reduce the certainty of accurate prediction so much.

D. Input data diversity

As discussed in Section II-C, the output of machine learning modules can also be diversified by varying input data. We examine the input data diversity by perturbating the test sample data and compare the predictions results with the same machine learning model. For data perturbation, we apply three different operations that are shift, rotate, and adding noise. Shift operation moves the digit to left by two pixels. Rotate operation rotates the digit by twenty degrees in the clockwise direction. Adding noise uses Gaussian-distributed additive noise with 0.01 of variance. Figure 5 shows the generated samples by these operations from the same sample digit.

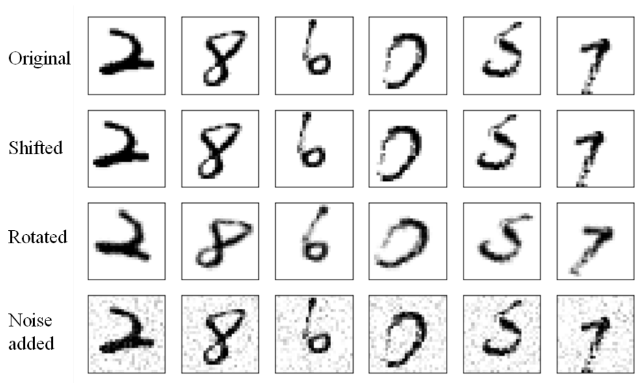

Figure 5. Samples generated by shift, rotate and noise adding operations

For the generated samples, machine learning models are required to predict the correct labels as for the original data. We use the same CNN model to predict the labels of the generated data. TABLE V shows the number of classification errors of individual digits by the original and three different types of generated data sets. We denote $\lvert E_{\mathrm{CNN,Y}}\rvert$ as the number of prediction errors for the original data (Y=o), shifted data (Y=s), rotated data (Y=r) or noise added data (Y=n).

The classification errors increase in most cases by using perturbated data compared with the classification errors observed by the original data. Interestingly, however, there are some cases that the errors are reduced by adding gaussian noise (i.e., for label 5 and 8). Although the numbers of errors are increased in the cases with shifted samples and rotated samples, their prediction results may contribute to improve the coverage of errors. To analyze this by quantifying the coverage or errors, we introduce a variant of the coverage of errors as defined below.

$$\mathrm{Cov}(m,\mathcal{D}) = 1 - \frac{\left|\bigcap_{D_j\in\mathcal{D}} E_{m,j}\right|}{|S|},$$

where $\mathcal{D} = \{D_1, D_2, \ldots\}$ is the set of data set from the same original data set, $E_{m,j}$ is the error space of the machine learning model m for the data set D_j. Since we use CNN in this experiment, m=CNN. Similarly, a variant of certainty of accurate predictions is defined as

$$\mathrm{Cer}(m,\mathcal{D}) = 1 - \frac{\left|\bigcup_{D_j\in\mathcal{D}} E_{m,j}\right|}{|S|}.$$

We computed these statistics as shown in TABLE VI.

As we can see, the coverage of errors is improved by adding the prediction results with the perturbated data regardless of the operations used. The total coverage of errors reaches 0.9957 by using different prediction results from four different data sets. Note that we use the same machine learning model generated from the same training data set, but just vary the input data for prediction by subtle modifications. Despite such a simple treatment which exploiting input data diversity, surprisingly we can achieve the complete error coverage for label 0 and 3 (See the line of $\mathrm{Cov}(\mathrm{CNN},\{\mathrm{o,s,r,n}\})$). This does not directly lead to the improved reliability of the machine learning system, since it depends on the way to select the correct result from multiple prediction results. The complete coverage of errors just guarantees that at least one correct prediction results exist in the candidates. We can wisely use this capability of error coverage in an architecture design.

For the certainty of accurate predictions, the values are decreased by adding the prediction results in all the cases with shifted and rotated data. When four prediction results are combined, the certainty decreases down to 0.9027 that is much worse than the previous cases. However, with the noise added data there are cases that the certainty of accurate predictions does not decrease (for label 3 and 8) from the accuracy of prediction by CNN with original data which is equal to $\mathrm{Cov}(\mathrm{CNN},\{\mathrm{o}\})$. This indicates that the certainty of accurate predictions do not always decrease even though the coverage of errors increases by noise adding.

E. *Classification of traffic sign images*

Next, we use Belgian Traffic Sign data set for conducting similar experiments to analyze the coverage of errors and the certainty of accurate prediction. Since the data set contains 62 different signs for the classification, we only show the prediction results for three specific classes ("Stop", "No entry" and "No stop") and the total error coverages. TABLE VII, VIII and IX, respectively show the results of label predictions by different algorithms, neural network architectures and perturbated data sets with CNN. For machine learning algorithms, neural network architectures, and data perturbation operations, we apply the same methods as used for MNIST task except for the changes of input data size.

TABLE VII. ERROR COVERAGES AND CERTAINTIES OF PREDICTIONS OF TRAFFIC SIGN IMAGES BY DIFFERENT MACHINE LEARNING MODELS

Label	Stop	No entry	No stop	Total
$\lvert S\rvert$	45	61	11	2520
$\lvert E_{\text{CNN}}\rvert$	3	0	1	130
$\lvert E_{\text{RF}}\rvert$	11	0	3	373
$\lvert E_{\text{SVM}}\rvert$	9	0	0	294
$\text{Cov}(\text{CNN})$	0.9333	1.0000	0.9091	0.9484
$\text{Cov}(\text{CNN}, \text{RF})$	0.9333	1.0000	1.0000	0.9548
$\text{Cov}(\text{CNN}, \text{SVC})$	0.9778	1.0000	1.0000	0.9631
$\text{Cov}(\text{CNN}, \text{RF}, \text{SVC})$	0.9778	1.0000	1.0000	0.9659
$\text{Cer}(\text{CNN}, \text{RF})$	0.7556	1.0000	0.6364	0.8456
$\text{Cer}(\text{CNN}, \text{SVC})$	0.7556	1.0000	0.9091	0.8687
$\text{Cer}(\text{CNN}, \text{RF}, \text{SVC})$	0.6444	1.0000	0.6364	0.8091

TABLE VIII. ERROR COVERAGES AND CERTAINTIES OF PREDICTIONS OF TRAFFIC SIGN IMAGES BY DIFFERENT NEURAL NETWORKS

Label	Stop	No entry	No stop	Total
$\lvert E_{\text{CNN}}\rvert$	3	0	1	130
$\lvert E_{\text{Dense}}\rvert$	0	0	0	247
$\lvert E_{\text{Expand}}\rvert$	4	0	0	157
$\text{Cov}(\text{CNN})$	0.9333	1.0000	0.9091	0.9484
$\text{Cov}(\text{CNN}, \text{Dense})$	1.0000	1.0000	1.0000	0.9579
$\text{Cov}(\text{CNN}, \text{Expand})$	0.9556	1.0000	1.0000	0.9619
$\text{Cov}(\text{CNN}, \text{Dense}, \text{Expand})$	1.0000	1.0000	1.0000	0.9746
$\text{Cer}(\text{CNN}, \text{Dense})$	0.9333	1.0000	0.9091	0.8925
$\text{Cer}(\text{CNN}, \text{Expand})$	0.8889	1.0000	0.9091	0.9159
$\text{Cer}(\text{CNN}, \text{Dense}, \text{Expand})$	0.8889	1.0000	0.9091	0.8726

TABLE IX. ERROR COVERAGES AND CERTAINTIES OF PREDICTIONS OF TRAFFIC SIGN IMAGES BY CNN WITH PERTURBATED TEST DATA

Label	Stop	No entry	No stop	Total
$\lvert E_{\text{CNN,o}}\rvert$	3	0	1	130
$\lvert E_{\text{CNN,s}}\rvert$	13	0	3	605
$\lvert E_{\text{CNN,r}}\rvert$	31	0	3	551
$\lvert E_{\text{CNN,n}}\rvert$	3	0	1	142
$\text{Cov}(\text{CNN}, \{\text{o}\})$	0.9333	1.0000	0.9091	0.9484
$\text{Cov}(\text{CNN}, \{\text{o}, \text{s}\})$	0.9333	1.0000	1.0000	0.9567
$\text{Cov}(\text{CNN}, \{\text{o}, \text{r}\})$	0.9333	1.0000	0.9091	0.9595
$\text{Cov}(\text{CNN}, \{\text{o}, \text{n}\})$	0.9333	1.0000	0.9091	0.9496
$\text{Cov}(\text{CNN}, \{\text{o}, \text{s}, \text{r}, \text{n}\})$	0.9333	1.0000	1.0000	0.9655
$\text{Cer}(\text{CNN}, \{\text{o}, \text{s}\})$	0.7111	1.0000	0.6364	0.7516
$\text{Cer}(\text{CNN}, \{\text{o}, \text{r}\})$	0.3111	1.0000	0.7273	0.7702
$\text{Cer}(\text{CNN}, \{\text{o}, \text{n}\})$	0.9333	1.0000	0.9091	0.9409
$\text{Cer}(\text{CNN}, \{\text{o}, \text{s}, \text{r}, \text{n}\})$	0.2889	1.0000	0.6364	0.6583

Overall, we observe that the total error coverages are improved by diverse algorithms, neural network architectures and perturbated input data sets. The achieved total error coverages are 0.9659, 0.9746, and 0.9655, respectively, while the error coverage achieved only by CNN is 0.9484. As shown in the line of $\text{Cov}(\text{CNN}, \text{Dense}, \text{Expand})$ in TABLE VIII, the combination of three neural networks achieves the complete coverages for classification of "Stop", "No entry" and "No stop".

In traffic sign recognitions, accurate detections of "Stop" sign and "No entry" sign are particularly important for safety purpose in consideration with autonomous driving system. Neglecting these signs may cause a serious traffic accident. For "No entry" sign, it is relatively easy to recognize as all the machine learning models do not have classification errors even by perturbated data. While the classification of "Stop" sign encounters some errors, the coverage of errors can be improved by using the different machine learning models. It can be noted that the classification errors of "Stop" sign are fully covered by using the different neural network architectures (in particular by adding the prediction results of Dense network) (See TABLE VIII). Figure 6 visualizes the overlaps of error spaces for classification of "Stop" sign images by the different neural network architectures.

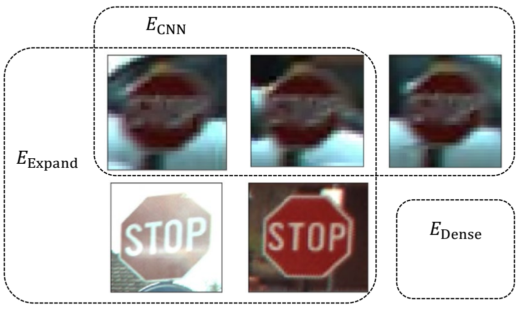

Figure 6. Error spaces of three neural networks for classification of traffic sign images labeled "Stop"

Interestingly, for this specific task, Dense network contributes to increase the coverage of errors against the results from other neural networks, while it does not contribute at all for the classification of zero in the MNIST test data as seen in Figure 4. The results also can be attributed to the power of diversity.

On the other side, the certainties of accurate predictions are significantly decreased by adding different prediction results. In particular for "Stop" sign classification, the certainty decreases down to 0.2889 by combining four prediction results from perturbated test data sets (See the line $\text{Cer}(\text{CNN}, \{\text{o}, \text{s}, \text{r}, \text{n}\})$ in TABLE IX), even though there is no improvement in the coverage. Therefore it is not a good option to employ the input data diversity in such a particular case. Since the decreases in the certainties of accurate predictions by three different neural networks are relatively acceptable, choosing this option could benefit the reliability of the system using traffic sign recognitions.

IV. SYSTEM RELIABILITY MODEL AND ANALYSIS

In this section, based on the experimental observations, we analytically investigate how the diversity of machine learning modules can be leveraged for designing a reliable software system. Since earlier studies on software fault-tolerance techniques presented the reliability models for N-version programming as well, we start from the reviews of the general reliability model and then present our model that incorporates the diversity parameter in terms of error spaces. In the following analysis, we focus on the three-version architecture. The system reliability is regarded as the probability that the system output is correct in terms of input data from the real world application context. Note that the accuracy of a machine learning model on the test data set only gives an empirical estimate of the module reliability and is not equal to the system reliability discussed in the following.

A. *Reliability model for N-version system*

One of the common methods to determine the output of N-version architecture is to take the majority vote of outputs from N components. The traditional model defines R_i as the reliability of component *i*'s output and assumes each component may output an error independently. The system reliability by majority voting from *N* outputs can be given by

$$R_{NV}(N) = 1 - \sum_{k=\left\lfloor \frac{N}{2} \right\rfloor+1}^{N} \sum_{\substack{\sum_{i=1}^{N} k_i = k, \\ k_i=\{0,1\}}} \prod_{i=1}^{N} {R_i}^{1-k_i} \cdot (1-R_i)^{k_i}.$$

For *N*=3, we have

$$R_{NV}(3) = R_1R_2 + R_1R_3 + R_2R_3 - 2R_1R_2R_3. \qquad (1)$$

When each component reliability is equivalent to R, the reliability expression can be simplified as

$$R_{NVE}(N) = \sum_{k=\left\lfloor \frac{N}{2} \right\rfloor+1}^{N} \binom{N}{k} \cdot R^k \cdot (1-R)^{N-k}.$$

When *N*=3, it becomes the reliability of triple module redundancy (TMR) system that is given by $3R^2 - 2R^3$. It is well known that the reliability of TMR system is inferior to the component reliability when *R*<1/2 [19].

Desirable N-version programming system implicitly assumes or expects that individual versions are independent of each other and hence common error is regarded as a rare case. However, empirical studies showed that errors of software versions largely correlated and they are not independent even implemented by different teams [20][21]. Earlier analytical studies also show that independently developed versions can encounter coincident failures which results in decreased software reliability [22][23]. To incorporate the factor of dependent failure, the dependent failure parameter α was introduced that represents the similarity percentage of the input sets on which each pair of versions fail [24]. The reliability of an N-version programming system can be expressed as

$$R_{NV\alpha}(\alpha, N) = 1 - \{n\alpha^{n-2}R(1-\alpha) + \alpha^{n-1}R\}.$$

Note that the above model also assumes that the reliabilities of individual components are equal to *R*. For *N*=3, we have

$$R_{NV\alpha}(\alpha, 3) = 1 - \alpha(3-2\alpha)(1-R). \qquad (2)$$

The above models can be used for computing an estimate of reliability of an N-version machine learning system. However, the diversity of machine learning modules observed in our experiments cannot be directly applied to these existing models, which may cause under- or over-estimate of the reliability. Therefore, we introduce the reliability model for an N-version machine learning system that incorporates the measures of diversity.

B. *Reliability of three-version architecture with diversity measure*

Considering the diversity-affected metric used in the experiments, we introduce the measure of diversity defined by the intersection of error spaces

$$\alpha_I := \frac{|\bigcap_{i\in I} \mathcal{E}_i|}{|\mathcal{S}|},$$

where $\mathcal{E}_i$ represents the error space of machine learning module m_i from the total sample space $\mathcal{S}$. In contrast to the dependent failure parameter α introduced by [24], the measure is defined on the set I and the values will change depending on the members of set I. The reliability of three version architecture using machine learning modules m_1, m_2 and m_3 can be expressed as follows.

$$R_{3Vd}(m_1, m_2, m_3) =$$
$$1 - \frac{|\mathcal{E}_1 \cap \mathcal{E}_2| + |\mathcal{E}_1 \cap \mathcal{E}_3| + |\mathcal{E}_2 \cap \mathcal{E}_3| - 2|\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3|}{|\mathcal{S}|}.$$

Using the diversity measure, we can rewrite the reliability as

$$R_{3Vd}(m_1, m_2, m_3) =$$
$$1 - \left(\alpha_{\{1,2\}} + \alpha_{\{1,3\}} + \alpha_{\{2,3\}} - 2\alpha_{\{1,2,3\}}\right). \qquad (3)$$

Although the true values of α_I are not known, empirical estimates of the diversity measures can be computed from the observed error spaces E_i for a specific sample space *S* by

$$\widehat{\alpha_I} = \frac{|\bigcap_{i\in I} E_i|}{|S|} = 1 - \mathrm{Cov}(I).$$

Using the empirical estimates, the system reliability of three version machine learning architectures with $(m_1, m_2, m_3) = (\mathrm{CNN}, \mathrm{RF}, \mathrm{SVM})$ for MNIST test samples can be computed as shown in TABLE X. We also show the computed system reliability by traditional reliability model $R_{NV}(3)$ from expression (1) and the reliability of TMR with the averaged component reliability (i.e., $(R_{\mathrm{CNN}} + R_{\mathrm{RF}} + R_{\mathrm{SVM}})/3$), and the reliability model using dependent failure parameter by expression (2) where we set $\alpha = \hat{\alpha}_{\{\mathrm{CNN,RF}\}}$.

TABLE X. Estimated system reliability with empirical diversity

Module reliability	R_{CNN}	0.9891
	R_{RF}	0.9707
	R_{SVM}	0.9704
Empirical diversity	$\hat{\alpha}_{\{\mathrm{CNN,RF}\}}$	0.7523
	$\hat{\alpha}_{\{\mathrm{CNN,SVM}\}}$	0.6697
	$\hat{\alpha}_{\{\mathrm{RF,SVM}\}}$	0.5802
	$\hat{\alpha}_{\{\mathrm{CNN,RF,SVM}\}}$	0.6055
System reliability	$R_{3Vd}(\mathrm{CNN,RF,SVM})$	0.9807
	$R_{NV}(3)$	0.9985
	TMR	0.9984
	$R_{NV\alpha}(\hat{\alpha}_{\{\mathrm{CNN,RF}\}}, 3)$	0.9738

The system reliability of three-version architecture is actually smaller than the best reliability by the module using CNN (i.e., $R_{3Vd}(\mathrm{CNN,RF,SVM}) < R_{\mathrm{CNN}}$). Although the coverage of errors is increased by diverse models, the majority voting often neglects the correct minority that results in decreased reliability. Nevertheless, since the reliability of three-version system is close to the reliability achieved by the best machine learning module, it is worth taking N-version architecture especially when there is less knowledge about which machine learning module achieves the best (e.g., using common models in a new application environment).

From TABLE X, we can also observe that the traditional reliability model $R_{NV}(3)$ and TMR model overestimate the reliability of three-version architecture, since the assumption of independence of different version is not true. It is important to estimate the intersection of error spaces and include it in the reliability model. Meanwhile, the reliability model $R_{NV\alpha}(\hat{\alpha}_{\{\mathrm{CNN,RF}\}}, 3)$ underestimates the reliability. This is caused by the simplified assumption that all the component reliabilities are equal to R and dependent failure parameters are equal to α for any pairs of modules. Our reliability model (3) can represent the system reliability more precisely.

C. A necessary condition for reliability improvement

Our experimental results indicate that the three-version architecture may not effectively improve the system reliability under a majority voting configuration. In order to analyze the condition where the three-version architecture achieves a better reliability, we compare $R_{3Vd}(m_1, m_2, m_3)$ with the reliability of a single machine learning model. Assume m_1 is the model achieves the highest reliability among the other models. The condition where the three-version architecture achieves a better reliability than any single component is given by $R_{3Vd}(m_1, m_2, m_3) - R_1 > 0$. Since R_1 is defined by $1 - |\mathcal{E}_1|/|\mathcal{S}|$, the condition is

$$R_{3Vd}(m_1, m_2, m_3) - R_1$$
$$= \frac{|\mathcal{E}_1| - \{|\mathcal{E}_1 \cap \mathcal{E}_2| + |\mathcal{E}_1 \cap \mathcal{E}_3| + |\mathcal{E}_2 \cap \mathcal{E}_3| - 2|\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3|\}}{|\mathcal{S}|}$$
$$= \frac{1}{|\mathcal{S}|}\left\{\begin{matrix}(|\mathcal{E}_1| - |\mathcal{E}_1 \cap \mathcal{E}_2| - |\mathcal{E}_1 \cap \mathcal{E}_3| + |\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3|) \\ -(|\mathcal{E}_2 \cap \mathcal{E}_3| - |\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3|)\end{matrix}\right\}$$
$$= \frac{1}{|\mathcal{S}|}\{|\mathcal{E}_1 \cap \overline{\mathcal{E}_2} \cap \overline{\mathcal{E}_3}| - |\overline{\mathcal{E}_1} \cap \mathcal{E}_2 \cap \mathcal{E}_3|\} > 0,$$

where $\overline{\mathcal{E}_i}$ represents the complementary set of $\mathcal{E}_i$. As a result, we obtain the following necessary condition for reliability improvement over the intersections of error spaces.

$$|\mathcal{E}_1 \cap \overline{\mathcal{E}_2} \cap \overline{\mathcal{E}_3}| - |\overline{\mathcal{E}_1} \cap \mathcal{E}_2 \cap \mathcal{E}_3| > 0 \tag{4}$$

Although the true values of the terms in the above condition are not obtainable, we can empirically count the numbers of samples for individual terms through experiments with a test data set. When the number of samples in $E_1 \cap \overline{E_2} \cap \overline{E_3}$ is larger than that in $\overline{E_1} \cap E_2 \cap E_3$ for a given sample space S, it is likely that three three-version architecture is effective.

D. A system design guide

From our findings through the experiments and reliability analysis, we summarize the guides for engineering reliable systems using diverse versions of machine learning modules.

- Exploiting input diversity

Among several approaches to diversifying the outputs of machine learning models, the approach using perturbated input data can be easily introduced because it does not need to prepare multiple machine learning models. If the coverage of errors increases by perturbated input data through experiments, it is worth considering to take a multi-version architecture for improving system reliability.

- Using multi-version models for error detection

As we observed in the experiments, different machine learning models have quite different error coverages. One of the direct applications of this property is using multi-version models for an error detection function. If any disagreement occurs among the multiple prediction results from different modules, there must be a prediction error. In this case, we can discard the prediction results to avoid undesirable consequence to the system and/or issue alert to higher level component.

- Estimating carefully the effectiveness of three-version architecture with majority voting

When employing an N-version architecture using three-version machine learning modules and majority voting, it is encouraged to carefully assess the reliability improvement using our reliability model. From a test data set, we can compute the sizes of error spaces $E_1 \cap \overline{E_2} \cap \overline{E_3}$ and $\overline{E_1} \cap E_2 \cap E_3$ empirically. Comparing these sizes with respect to the necessary condition (4), we can have a guide to decide whether three-version machine learning models can be effectively used under the majority voting or not.

V. Related work

Multi-version machine learning approaches have been studied in different contexts and purposes; i) for generating a better machine learning model in terms of accuracy, ii) for testing an implementation of machine learning algorithm, and iii) for improving the reliability of the system using machine learning models. Our work focused on characterizing the diversity of error outputs of different machine learning models and are closely related to these related studies as discussed below.

First, to obtain a well performed supervised machine learning model, ensemble learning [24] is a commonly adopted technique in which multiple models are combined to improve the prediction accuracy. Random forest [14] is also known as an ensemble learning method that generates multiple decision trees by random sampling from training data set. While empirically ensemble learning methods often yield a better result, theoretically there is no guarantee for improved accuracy. The improved accuracy by ensemble can be attributed by the diversity of machine learning models. The relationship between several diversity measures and empirical prediction accuracy was extensively studied [26]. A variety of diversification methods for machine learning processes and the applications of the diversity technology are surveyed [27]. While our work also looks into the diversity in terms of the coverage of errors attained by machine learning models, our focus is not on learning process itself. Our objective is to leverage the diversity to improve the system reliability, and thus we present the reliability model that connects the empirical diversity measure and the system reliability achieved by three-version architecture.

Second, a multi-version machine learning approach is used for testing implementations of machine learning algorithms. Ensuring the quality of the products or services using machine learning is a challenging issue since in practice there is no complete oracle that can guarantee the expected prediction results for new input. The problem “no oracle” in testing machine learning models has been discussed [28] and a metamorphic testing approach was presented. A multi-version machine learning approach is recently introduced [29]. To test the output of a machine learning model, an alternative version of machine learning model is used for creating a proxy oracle. The approach can effectively find the implementation faults of machine learning algorithm. While this work uses the multi-version approach in a testing phase to improve a single machine learning model, our study considers the architecture of machine learning system using multiple models. Multi-version approaches can be used both in testing and architecture design phases.

Third, some recent studies attempt to apply N-version programming approach to machine learning systems considering the system level reliability. A recent study presented the architecture using three redundant deep neural networks with weighted majority voting for predicting steering angle of autonomous vehicle [30]. Preliminary results showed the improved reliability by N-version architecture. Our experimental results with multiple neural network architectures, however, did not improve the reliability. This is probably caused by the difference of voting mechanism used. An N-version approach using multiple deep neural networks is also studied by NV-DNN [31]. While the different training process, network architectures and training data sets are examined, the diversification approach by perturbated input data has not been studied. Although a recent theoretical study analyzes the potential reliability enhancement by using different input data for prediction [32], our work first experimentally shows that the coverages of errors are improved by using varied input data. Moreover, our study clarifies the condition where three-version machine learning architecture achieves better system reliability than the best machine learning module, which has not been presented in the previous literature for N-version machine learning systems.

VI. Conclusion

In this paper, we have experimentally evaluated the diversity of machine learning models for image classification tasks toward designing a reliable machine learning system. The impact of diversity is quantified by the coverage of errors defined as the ratio of the samples that are not correctly predicted by a given set of machine learning models. We also quantified the certainty of accurate prediction that has a trade-off relationship with the coverage of errors. Our experimental results show that the coverages of errors are increased by using different machine learning algorithms, different neural network architectures and perturbated input data, while the certainties of predictions are decreased in most cases. With this observation, we presented the reliability model for three-version machine learning architecture that takes into account the intersections of error spaces of machine learning models. The presented reliability model is used to characterize the necessary condition under which the three-version architecture achieves a higher reliability than the reliability of the best module. While this paper focuses on three-version architecture, generalization of the reliability model for N-version architecture will be discussed in the future work. Since our experimental results are limited to image classification tasks, extending the study for other fields like voice recognition and natural language processing is also an important future work.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, nature, Vol. 521, No. 7553, pp. 436, 2015.

[2] D. Sculley et al., Hidden technical debt in machine learning systems. In Proc. of Advances in Neural Information Processing Systems, pp. 2503-2511, 2015.

[3] A. Avizienis and L. Chen, On the implementation of N-version programming for software fault tolerance during execution. In Proc. of IEEE International Computer, Software and Application Conference (COMPSAC), pp. 149–155, 1977.

[4] A. Avizienis, The methodology of n-version programming, Software fault tolerance, Vol. 3, pp. 23-46, John Wiley & Sons, New York, 1995.

[5] Y. LeCun, C. Cortes, and C. Burges, The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/, retrieved in May, 2019.

[6] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, Traffic sign recognition - How far are we from the solution?, In Proc. of International joint conference on Neural networks, pp. 1-8, 2013.

[7] B. Zoph, and Q. V. Le, Neural architecture search with reinforcement learning, arXiv:1611.01578, 2016.

[8] H. Liu, K. Simonyan, and Y. Yang, Darts: Differentiable architecture search, arXiv:1806.09055, 2018.

[9] I. Goodfellow, J. Shlens, and C. Szegedy, Explaining and harnessing adversarial examples, arXiv:1412.6572, 2014.

[10] N. Carlini, D. Wagner,Towards evaluating the robustness of neural networks, In Proc. of IEEE Symposium on Security and Privacy (SP), pp. 39-57, 2017.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, arXiv:1706.06083, 2017.

[12] J. Su, D. V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks, IEEE Trans. on Evolutionary Computation, 2019

[13] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, Safety verification of deep neural networks, In Proc. of International Conference on Computer Aided Verification, pp. 3-29, 2017.

[14] L. Breiman, Random forests, Machine learning, Vol. 45, No. 1, pp. 5-32, 2001.

[15] C. Cortes, and V. Vapnik, Support-vector networks, Machine Learning, Vol. 20, No. 3, pp. 273-297, 1995.

[16] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[17] scikit-learn, https://scikit-learn.org

[18] Keras, https://keras.io/

[19] K. S. Trivedi, Probability and statistics with reliability, queuing, and computer science applications, John Wiley, New York, 2001.

[20] D. E. Eckhardt, A. K. Caglayan, J. C. Knight, L. D. Lee, D. F. McAllister, M. A. Vouk, and J. P. J. Kelly, An experimental evaluation of software redundancy as a strategy for improving reliability, IEEE Trans. on Software Eng., Vol. 17, No.7, pp. 692-702, 1991.

[21] J. C. Knight, N. G. Leveson, An experimental evaluation of the assumption of independence in multiversion programming, IEEE Trans. on Software Eng., Vol. SE-12, No.1, pp. 96-109, 1986.

[22] D. E. Eckhardt, L. D. Lee, A theoretical basis for the analysis of multiversion software subject to coincident errors, IEEE Trans. Software Eng., Vol. SE-11, No. 12, pp. 1511-1517, 1985.

[23] B. Littlewood, D.R. Miller, Conceptual modeling of coincident failures in multiversion software, IEEE Trans. on Software Eng., Vol. 15, No. 12, pp.1596-1614, 1989.

[24] M. Ege, M.A. Eyler, M.U. Karakas, Reliability analysis in N-version programming with dependent failures, In Proc. of 27th EUROMICRO Conference, pp. 174-181, 2001.

[25] T. Dietterich, Ensemble methods in machine learning, In Proc. of international workshop on multiple classifier systems, pp. 1-15, 2000.

[26] L. I. Kuncheva, and C. J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine learning, Vo. 51, No.2, pp. 181-207, 2003.

[27] Z. Gong, P. Zhong, and W. Hu, Diversity in Machine Learning, arXiv:1807.01477, 2018.

[28] C. Murphy and G. E. Kaiser, Improving the dependability of machine learning applications, Technical report, Columbia University, 2008.

[29] S. Srisakaokul, Z. Wu, A. Astorga, O. Alebiosu, and T. Xie, Multiple-implementation testing of supervised learning software., In Proc. of workshops at 32nd AAAI Conference on Artificial Intelligence, 2018.

[30] A. Wu, A. H. M. Rubaiyat, C. Anton, and H. Alemzadeh, Model Fusion: weighted N-version programming for resilient autonomous vehicle steering control. In Proc. of IEEE International Symposium on Software Reliability Engineering Workshops, pp. 144-145, 2018.

[31] H. Xu, Z. Chen, W. Wu, Z. Jin, S. Kuo, M. R. Lyu, NV-DNN: towards fault-tolerant DNN systems with N-version programming, In Proc. of the DSN Workshop on Dependable and Secure Machine Learning, pp. 44-47, 2019.

[32] F. Machida, N-version machine learning models for safety critical systems, In Proc. of the DSN Workshop on Dependable and Secure Machine Learning, pp. 48-51, 2019.