N-version Machine Learning Models for Safety Critical Systems

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

Abstract— **Quality control of machine learning systems is a fundamental challenge in industries to provide intelligent services or products using machine learning. While recent advances in machine learning algorithms substantially improve the performance of intelligent tasks such as object recognition, their outputs are essentially stochastic and very sensitive to input data. Such an output uncertainty is a big obstacle to ensure the quality of safety critical applications like autonomous vehicle and hence architectural design to mitigate the impact of error output becomes a great importance. In this paper, we propose N-version machine learning architecture that aims to improve system reliability against probabilistic outputs of individual machine learning modules. The key idea of this architecture is exploiting two kinds of diversities; input diversity and model diversity. Our study first formally defines these diversity metrics and analytically shows the improved reliability by N-version machine learning architecture. Since we treat a machine learning module as a black-box, the proposed architecture and the reliability property are generally applicable to any machine learning algorithms and applications.**

Keywords—design diversity, machine learning, N-version programming, reliability, safety

I. INTRODUCTION

Recent advances of machine learning algorithms with increased computing power and available big data further expand the applications of machine learning systems. Computer vision, voice recognition and machine translation now mostly use deep neural networks, as their performances overreach the limit of conventional algorithms and even human capability. Machine learning applications are expected not only for improving existing products or services, but also for as a key enabler of new innovative systems such as autonomous vehicles and advanced medical diagnostic tools.

An emergent challenge of system providers is quality control of the product or service uses machine learning. In particular for industries developing safety-critical applications, quality assurance is essential for their businesses. Unlike any software program that is coded by deterministic logic, the output of machine learning modules are generally uncertain. A machine learning module uses a model constructed from a training data set through a specific algorithm. The output of the module completely depends on the trained model and how well the learning algorithm extract the features of the data. Moreover, in many practical applications, input data to machine learning module is most likely not included in the training data set. Therefore, system provider is not able to ensure the complete correctness of machine learning module's output in user environment where unexpected input occurs sporadically. This uncertainty causes a critical threat to machine learning models. A cyber-attack can fool the machine learning system to output error by malicious input, which is now commonly known as adversarial example [1].

A number of studies are conducted for developing robust machine learning algorithms against adversarial examples. The robustness of machine learning algorithm can be increased in training phase by intentionally incorporating adversarial examples in the training data set [1]. Existence of adversarial examples for deep neural network models can be verified by safety verification techniques [2]. While these studies intend to increase the robustness of a machine learning module, they do not guarantee the complete reliability of the system. System reliability is affected not only by adversarial examples, but also by other factors such as errors in the labels of training data sets and/or algorithm implementations. System providers need to premise the occurrence of error outputs but low probability and should contemplate the reliable system design to reduce the impact of output errors.

In this paper, we propose a reliable system architecture using N-version machine learning models whose output may not be correct. Our focus is not on the training of robust machine learning models, but on a reliable system processing with multiple machine learning modules. Different machine learning models can be used in the same system so as to improve the reliability of the system output. The key of reliability enhancement is delivered through the two kinds of diversities employed in the system architecture, namely *model diversity* and *input diversity*. Model diversity is achieved by using different machine learning models built from different algorithms, parameters, training data sets, developers and/or programming languages. The multi-version approaches are also used for test of machine learning model [3]. In contrast, we use multi-version models to increase the system reliability in the operation. In system operation, the difference of input is also an important factor of diversity. Input diversity is achieved by taking different samples for the input to machine learning models. For instance, an autonomous vehicle equips two cameras with different angle may produce two different image data for the recognition task at a certain time and place. Even if a machine learning model outputs an error with one input (e.g., fails to recognize an object), the model can output correct answer with the other input depending on the degree of diversity between two inputs. We formally define these two kinds of diversities and analytically show the improved reliability of N-version machine learning architecture in different configurations. A preliminary numerical example exhibits that the architecture exploiting both the diversities achieve the best reliability in most cases.

II. N-VERSION MACHINE LEARNING MODEL

N-version machine learning model is analogous to N-version programming that is well-known software fault-tolerant technique. N-version programming is originally defined as "the independent generation of $N \geq 2$ functionally equivalent programs from the same initial specification" [4]. Unlike hardware reliability, redundant configuration of the same software component do not improve the system reliability because the same software implementation can produce the same error output by the software faults. Provided that complete removal of software faults from a software component is extremely difficult, N-version software programming offers the tolerance to software faults by multiple implementations of the same software specification by changing development teams, programming languages, libraries and tools. Except fundamental errors in specification, any errors generated in software development process can be covered by different implementations that can lead to improved system reliability.

Machine learning model generally has a similar property to software component. The output reliability cannot be improved by the redundant use of the same machine learning model as it produces the same error by the same input. Therefore, in this paper, we consider the variant of N-version programing for machine learning models. The differences between machine learning model and general software component are twofold as follows. First, machine learning model is constructed by training from examples, while software component is developed from a specification. The sources of errors in machine learning model mainly come from the algorithm and training data sets instead of software faults. Second, the output of machine learning model is stochastic and sensitive to the input data, while the output of software component is generally deterministic as specified in advance. A single machine learning model may behave differently for different input from the same sample space associated with the same expected outcome. This implies that machine learning model can output correct answer with different input even if it outputs error by a certain input. This difference gives us a unique challenge of N-version machine learning model; how can we exploit the different kinds of diversities in the architecture of machine learning system. Hereafter, we use the term *model diversity* as the diversity inherited from machine learning model itself, and *input diversity* as the diversity comes from different input data. The formal definition of these diversities are explained in Section III. In the following, we present some elemental but effective architecture of N-version machine learning models that aims to improve the reliability of system output.

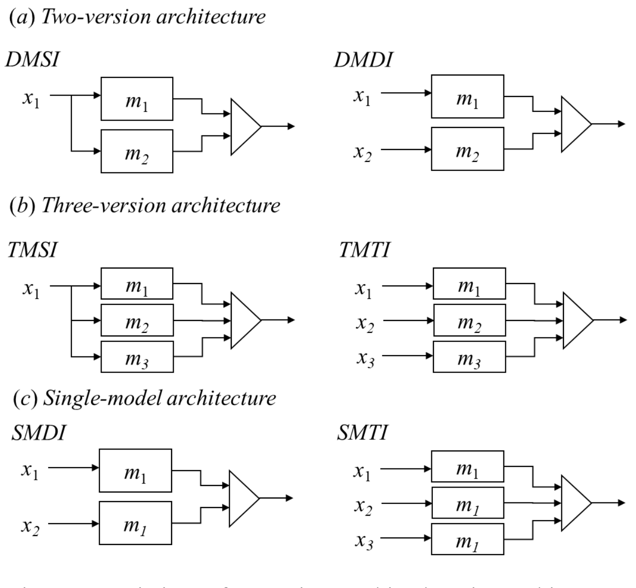

Figure 1. Variations of N-version machine learning architecture

A. Two-version architecture

When we have two versions of machine learning models for processing the same task (e.g., object recognition from an image data), there are at least two options to configure two modules in a system architecture (See Figure 1(a)). Double model with single input (DMSI) system simply exploits model diversity such that each module employs a different model. The system is assumed to fail when either module do not output expected answer. An error output of one module may be covered by a correct output by another module using different model and thereby the reliability of system output can increase. Since two models are constructed for the same purpose, the probability of simultaneous errors can be high, but the model diversity helps reduce such a probability. On the other hand, double model with double input (DMDI) system exploits input diversity besides model diversity by using different inputs from the same sample space associated with the ground truth. The system is assumed to fail when both of two modules do not output expected answers (i.e, the ground truth). Even if both models result in error output by an input, either one of the models can output correct answer by another input. Intuitively, the output reliability of DMDI is better than the reliability of DMSI, since DMDI can benefit input diversity as well as model diversity. We will formally discuss this point in Section III.

B. Three-version architecture

Two-version architecture can be extended to three-version architecture, where three different machine learning models are used in a system. Among several options to combine three different modules with different input, we show two representative configurations as shown in Figure 1(b). Triple model with single input (TMSI) system simply adds one additional machine learning module to DMSI. The output of the system is determined by consensus among three modules. In majority voting basis, the system fails when more than two modules output errors with the same input. Note that error detection capability increases by the additional module, while the probability of common errors among different versions also increases. The architecture is essentially a variant of triple modular redundancy (TMR) whose reliability characteristic is well known when failure probabilities of modules are independent [5]. TMSI differs from TMR, since we assume dependence of machine learning modules leading to common errors. When each module receives different inputs, the architecture becomes triple model with triple input (TMTI) system. TMTI exploits input diversity in addition to model diversity and the system fails when more than two modules output errors.

C. Single model architecture

Another configuration we can also consider is single model architecture that uses the same model in parallel with different inputs. In this architecture, we do not use N-version in terms of machine learning model, but the outputs will change depending on which input is processed. When we use two different inputs, the system is single model with double input (SMDI) system, where the system fails when both of two outputs are errors (see Figure 1(c)). Similarly, single model with triple input (SMTI) system uses three machine learning modules using the same machine learning model in parallel. A failure of SMTI occurs when more than two modules outputs errors. If the input diversity is high, these single model architecture is beneficial because preparing and maintaining different version of machine learning models are often very expensive. In this paper, we categorize SMDI in a two-version architecture and TMDI as a three-version architecture.

III. Reliability analysis

In this section, we formally define the two types of diversities and analyze the reliability of different configurations of N-version machine learning architecture. Due to the space limitation, we only show the reliability model for the two-version architecture, but the derivation can be extended to more than two-version cases. System reliability is defined as the probability that the output of the system is correct. We assume that the correct answer is given by real application context. For example, a computer vision module equipped on an autonomous vehicle is expected to recognize the red light as presented in real environment at an intersection. Any misconception of the red light are regarded as errors. Define $R_{i,j}$ as the reliability of a machine learning system with *i* versions with *j* diverse inputs. A system using a single model with single input can be given by $R_{1,1}(m_k) = 1 - f_k$, where f_k represents the probability that the machine learning model m_k outputs error. Denote S as the total sample space of inputs in a given real context and $E_k \subseteq S$ as the set of input data that leads to output error by m_k. The error probability f_k can be given by $|E_k|/|S|$. In order to consider the reliability of N-version machine learning architecture, we first introduce the diversity metrics.

A. Definition of diversity

As previously mentioned in Section 2, two different types of diversities are the key to enhance the reliability of N-version machine learning architecture. We formally define model diversity and input diversity by *intersection* and *conjunction of errors*, respectively as below.

Intersection of errors *(model diversity)*
Let E_1 and E_2 be the subsets of input space S that make machine leaning models m_1 and m_2 output errors respectively. Define the intersection of errors $\alpha_{1,2} \in [0,1]$ as the ratio of the intersection over the smaller the size of E_1 and E_2.

$$\alpha_{1,2} = \frac{|E_1 \cap E_2|}{\min\{|E_1|, |E_2|\}}.$$

The intersection of errors represents the degree of overlap between the sets E_1 and E_2. Since E_1 and E_2 are attributed by the models m_1 and m_2, respectively, their intersection indicates how the two models are similar to each other in terms of erroneous inputs. The smaller intersection decreases the probability of common errors of m_1 and m_2, which corresponds to the larger model diversity.

Conjunction of errors *(input diversity)*
Let x_1 and x_2 be the inputs from the same sample space S to the machine learning model m_1. Define the conjunction of errors $\beta_1 \in [0,1]$ as the probability that m_1 outputs error by x_2 provided that m_1 outputs error by x_1.

$$\beta_1 = \Pr[x_2 \in E_1 | x_1 \in E_1].$$

The conjunction of errors represents the possibility of both inputs x_1 and x_2 fall into error output through the process of the same machine learning model. We assume that the sample space of input x_2 is constrained by the observation of error by the input x_1 such that x_2 also falls into an error output. If x_1 and x_2 are fully independent, β_1 is equal to f_1. The similarity of input can cause the increased conjunction which results in $\beta_1 \geq f_1$. Thus, the smaller conjunction decreases the probability of double errors thanks to the larger input diversity.

B. Reliability of two-version architecture

With the defined diversity metrics, we provide the reliability models for DMSI, SMDI and DMDI systems.

1) DMSI

The double model system employs two different machine learning models m_1 and m_2. In this system, both machine learning models process the same input x. The system fails when both models output errors. This means that x is in the intersection of E_1 and E_2. Therefore, the failure probability of DMSI system is given by

$$f_{DMSI}(m_1, m_2) = \frac{|E_1 \cap E_2|}{|S|} = \alpha_{1,2} \cdot \frac{\min\{|E_1|, |E_2|\}}{|S|}.$$

Without loss of generality, we can assume $|E_1| \leq |E_2|$ and in this case the reliability of DMSI system is computed by $R_{2,1}(m_1, m_2) = 1 - f_{DMSI}(m_1, m_2) = 1 - \alpha_{1,2} \cdot f_1$.

2) SMDI

This system configures two machine learning modules using the same machine learning model m_1 but process different inputs x_1 and x_2. Since the system exploits the input diversity, the failure probability of SMDI system can be expressed as

$$\begin{aligned} f_{SMDI}(m_1) &= \Pr[x_1 \in E_1, x_2 \in E_1] \\ &= \Pr[x_2 \in E_1 | x_1 \in E_1] \cdot \Pr[x_1 \in E_1] \\ &= \beta_1 \cdot f_1. \end{aligned}$$

The system reliability is given by $R_{1,2}(m_1) = 1 - \beta_1 \cdot f_1$.

3) DMDI

This architecture combines the above two configurations. Specifically, two different machine learning models m_1 and m_2 process the different inputs x_1 and x_2, respectively. The architecture benefits both two types of diversity. The failure probability of DMDI system can be computed by

$$\begin{aligned} f_{DMDI}(m_1, m_2) &= \Pr[x_1 \in E_1, x_2 \in E_2] \\ &= \Pr[x_2 \in E_2 | x_1 \in E_1] \cdot \Pr[x_1 \in E_1]. \end{aligned}$$

Given $x_1 \in E_1$, there are two cases that x_2 also causes an error.

a) x_2 has conjunction with x_1

A conjunction occurs only when x_2 is in $E_1 \cap E_2$. Since β_1 represents the possibility of conjunction for the erroneous input space E_1, we can derive the error probability

$$\begin{aligned} &\Pr[x_2 \in E_1 | x_1 \in E_1] \cdot \Pr[x_2 \in E_2 | x_2 \in E_1] \\ &= \beta_1 \cdot |E_1 \cap E_2| / |E_1| \\ &= \beta_1 \cdot \alpha_{1,2} \cdot \min\{f_1, f_2\} / f_1. \end{aligned}$$

b) x_2 has no conjunction with x_1

When there is no conjunction, x_2 should be in $\overline{E_1}$. In this case, the model m_2 outputs error when $x_2 \in \overline{E_1} \cap E_2$. We can derive the error probability

$$\begin{aligned} &\Pr[x_2 \in \overline{E_1} | x_1 \in E_1] \cdot \Pr[x_2 \in E_2 | x_2 \in \overline{E_1}] \\ &= (1 - \beta_1) \cdot |\overline{E_1} \cap E_2| / |\overline{E_1}| \\ &= (1 - \beta_1) \cdot (|E_2| - |E_1 \cap E_2|) / (|S| - |E_1|) \\ &= (1 - \beta_1) \cdot \frac{f_2 - \alpha_{1,2} \cdot \min\{f_1, f_2\}}{1 - f_1}. \end{aligned}$$

Taking the sum of the above two probabilities, we have $\Pr[x_2 \in E_2 | x_1 \in E_1]$. When we assume $|E_1| \leq |E_2|$, the failure probability is expressed as

$$f_{DMDI}(m_1, m_2) = \left[\beta_1 \cdot \alpha_{1,2} + (1 - \beta_1) \cdot \frac{f_2 - \alpha_{1,2} \cdot f_1}{1 - f_1}\right] \cdot f_1.$$

Note that $f_{DMDI}(m_1, m_2)$ becomes $f_1 \cdot f_2$ if β_1 is equal to f_1, which corresponds to the case that two machine learning modules output error independently. The system reliability of DMDI is $R_{2,2}(m_1, m_2) = 1 - [(\beta_1 - f_1) \cdot \alpha_{1,2} + f_2] \cdot f_1$.

C. *Numerical example*

To understand the impact of diversities on the reliabilities of two-version machine learning architecture, we show a numerical example. Assume that error probability of any single machine learning model is 0.2 (=f_1= f_2). When we fix β_1=0.4, the reliabilities of different configurations by varying $\alpha_{1,2}$ is computed as shown in Figure 2. In DMSI, when we can fully exploit the model diversity and makes two models do not have intersection (i.e., $\alpha_{1,2}$=0), the system does not fail ($R_{2,1}$=1). The complete reliability is not achievable by DMDI as it uses different inputs, although it generally improves the reliability regardless of the value of $\alpha_{1,2}$ (i.e., $R_{2,2} \geq R_{1,1}$). Next, we fix $\alpha_{1,2}$=0.5 and compute the reliabilities by varying β_1 in the range of [0.2, 1]. Figure 3 shows the results. When β_1 =0.2 (=f_1), there is no conjunction and two modules output errors independently. As β_1 increases, the reliability of two-version system relying on input diversity decreases.

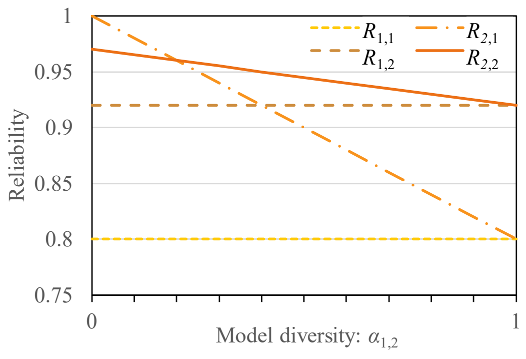

Figure 2. Reliability impacts of model diversity

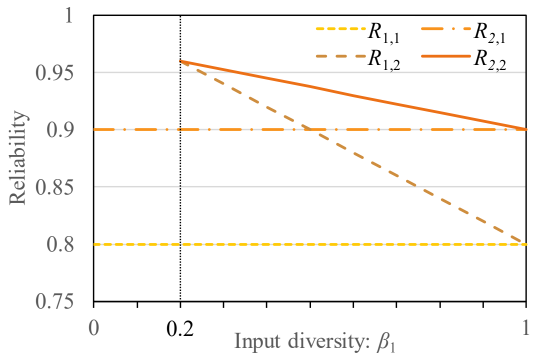

Figure 3. Reliability impacts of input diversity

IV. RELATED WORK

Multi version machine learning approach has been studied for the purpose of testing machine learning software [3]. Different versions of machine learning models are used to make a proxy oracle that do not guarantee the accuracy with perfect confidence but is still effective to detect implementation faults. While multi implementation testing method aims to improve the quality of a machine learning module, our approach uses multi version implementations in the system operation phase and aims to improve the total system reliability. Although it remains a discussion about in which phase we should employ multi version implementations most effectively for required dependability, our approach can benefit the improved reliability by input diversity observed in real user environment.

System testing methods for ensuring the correct system behavior are important as well for safety-critical applications. DeepXplore presented a white-box testing method for systems using deep neural network whose corner case of outputs are concerned in a system perspective [6]. To characterize the progress of the tests, the concept of neuron coverage is introduced that measures the parts of neural network exercised by a set of test inputs. The framework successfully detected the erroneous outputs of deep neural networks efficiently. Although robustness of deep neural networks against adversarial example are widely discussed, there are few works address the total system reliability or consequential safety concerns. Dreossi et al. recently addressed this issue and proposed a falsification framework that falsify the temporal logic considering the cyber-physical system (CPS)'s control loop architecture containing a machine learning module that may produce error output [7]. The CPS level analysis helps to identify the erroneous input space of machine learning model that can violate a given safety requirement. Our work is close to this study because we also look into system level reliability, but the approach to improve the reliability is different. We first discuss the potential reliability enhancement of a system employing N-version machine learning architecture.

V. CONCLUSION

In this paper, we presented the architecture of N-version machine learning model that aims to improve system reliability against unexpected outputs from machine learning modules. The property of the N-version architecture is characterized by model diversity and input diversity that we formally define the metrics and analyze the reliability impact on N-version architecture. Although we only show the reliability analysis for two-version architecture due to the space limit, these configurations are essentially the building blocks of the configuration with more than two versions. The presented reliability model could be useful to investigate larger systems. Furthermore, we dealt with a machine learning module as a black-box, the presented architecture and reliability analysis are generally applicable to any machine learning algorithms and applications. An empirical study to show the actual reliability improvement is our undergoing future work.

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

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

[6] K.Pai, Y. Cao, J. Yang, and S. Jana, Deepxplore: Automated whitebox testing of deep learning systems, In proc. of the 26th Symposium on Operating Systems Principles, pp. 1-18, 2017.

[7] T. Dreossi, A. Donzé, and S. A. Seshia, Compositional falsification of cyber-physical systems with machine learning components, In NASA Formal Methods Symposium, pp. 357-372, 2017.