Selecting Test Inputs for DNNs using Differential Testing with Subspecialized Model Instances

Yu-Seung Ma
Electronics and Telecommunications Research Institute
Daejeon, Republic of Korea
ysma@etri.re.kr

Shin Yoo*
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
shin.yoo@kaist.ac.kr

Taeho Kim
Electronics and Telecommunications Research Institute
Daejeon, Republic of Korea
taehokim@etri.re.kr

ABSTRACT

Testing of Deep Learning (DL) models is difficult due to the lack of automated test oracle and the high cost of human labelling. Differential testing has been used as a surrogate oracle, but there is no systematic guide on how to choose the reference model to use for differential testing. We propose a novel differential testing approach based on subspecialized models, i.e., models that are trained on sliced training data only (hence specialized for the slice). A preliminary evaluation of our approach with an CNN-based EMNIST image classifier shows that it can achieve higher error detection rate with selected inputs compared to using more advanced ResNet and LeNet as the reference model for differential testing. Our approach also outperforms N-version testing, i.e., the use of the same DL model architecture trained separately but using the same data.

CCS CONCEPTS

• **Software and its engineering** → **Software testing and debugging**.

KEYWORDS

Machine Learning, Diffrential Testing, Test Oracle

ACM Reference Format:
Yu-Seung Ma, Shin Yoo, and Taeho Kim. 2021. Selecting Test Inputs for DNNs using Differential Testing with Subspecialized Model Instances. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3468264.3473131

1 INTRODUCTION

Deep Learning (DL) systems are increasingly being adopted in real-world applications, thanks to their surprising accuracies [2, 3, 9]. However, despite the widespread adoption, testing of DL systems remains a challenge due to fundamental differences between DL systems and traditional software. A recent survey on testing of machine learning systems [11] reports that one of the major technical challenges is the test oracle problem [1]. Since DL models are inherently stochastic and perform cognitive tasks such as image recognition or natural language understanding, we cannot easily write sound and complete automated oracles and must depend on human labelling. This, in turn, also means that we cannot know which input is capable of revealing incorrect behaviour of Model Under Test (MUT) until labelling is complete.

Differential testing is one way to handle the test oracle problem of DL systems [5]. Intuitively, it aims to find an input that causes multiple implementations of the same requirement to disagree with each other by producing different outputs: such an input means that at least some of the implementations may be incorrect. The high cost of human labelling (which serves as test oracles for DL systems) makes differential testing a particularly appealing candidate for testing DL systems. The seminal work of Pei et al. [10], DeepXplore, uses differential testing to identify inputs that are more likely to reveal problematic behaviour in DL systems. Similarly, DLFuzz [6] uses differential testing to guide fuzzing of DL systems.

Differential testing relies on the availability of a reference model, i.e., an additional model that is developed for the same purpose, but different from the MUT. The effectiveness of differential testing directly depends on both the quality of the reference model that are cross-referenced and the degree it differs from the MUT. If the reference model has poor performance, disagreements raised by it may not be an accurate indicator of undesirable behaviour; if it is not sufficiently different from MUT, they may rarely disagree. Despite the importance of this problem, there is little guidance on how to construct the reference model when using differential testing as a test oracle for DL systems.

This paper proposes a new differential testing method that can select inputs that can reveal erroneous behaviour of DNN MUT. Intuitively, instead of training a completely different reference model on the entire training data, we slice the training data of MUT into multiple subsets based, and train subspecialized models independently for each subset: each of the subspecialized models shares the same model architecture as MUT, but is trained on a subset of its training data. When testing MUT, we can use the corresponding subspecialized model as the reference model for differential testing, i.e., raise an alarm when MUT disagrees with the subspecialized model. A preliminary empirical evaluation of the proposed approach using EMNIST [4] dataset shows that the subspecialized differential testing can achieve almost double the error detection rate when compared to traditional differential testing approaches that use other advanced DL models.

*Corresponding author

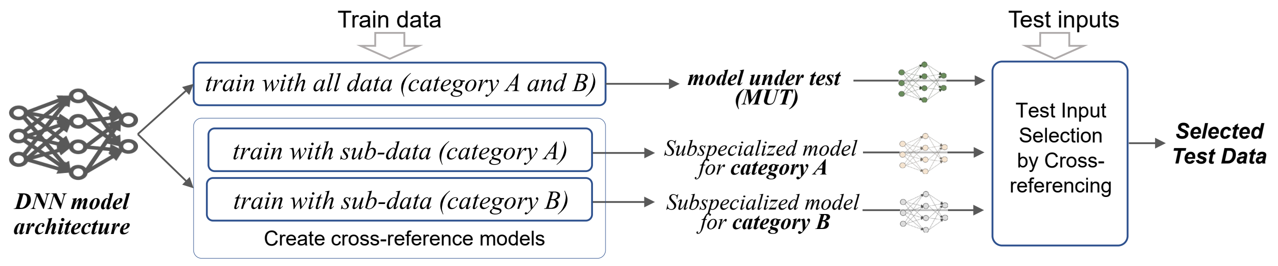

Train data
Test inputs
DNN model architecture
train with all data (category A and B)
train with sub-data (category A)
train with sub-data (category B)
Create cross-reference models
model under test (MUT)
Subspecialized model for category A
Subspecialized model for category B
Test Input Selection by Cross-referencing
Selected Test Data

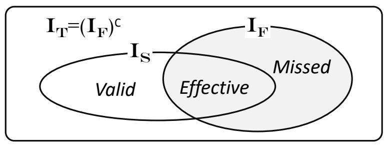
Figure 1: Overview of our approach

2 METHODOLOGY

This section first describes the required properties of test input that can reveal errors in the context of differential testing of DL systems, and subsequently presents our proposed approach that uses subspecialized model instances.

2.1 Test Data Selection for DL Differential Testing

Figure 2 illustrates an input space, denoted by the set of all inputs, I, for a given MUT, M. The set I consists of I_T and I_F, which contains inputs handled correctly and incorrectly by M, respectively. Since the output of M is either correct or not, I_T is equal to $(I_F)^c$, the complement set of I_F.



		Inference Result	
		True (I_T)	**False** (I_F)
Select as Actual Test Set	**Yes** (I_S)	Not Error (Valid)	Error Found (Effective)
	No	Not Error (Ignored)	Undetected Error (Missed)

Figure 2: Classification of test inputs for DL diffential testing

The basic premise of the DL differential testing is that, if the same test input leads to different test results between M and its reference model, M', at least one of these models is behaving incorrectly. Let I_S be the set of inputs whose results by M and M' are different from each other. As can be seen from Figure 2, I_S can include inputs from both I_T and I_F. Since our aim is to select more inputs from I_F for the purpose of testing M, we would prefer a larger $I_S \cap I_F$ (since inputs in the intersection are effective), and a smaller $I_S - I_F$ (since inputs in the difference do not reveal any erroneous behaviour).

2.2 Proposed Method

We posit that the following two issues need to be considered when applying differential testing to DL systems.

Data Perspective Issue. The capabilities of a DL model largely depend on the composition of the train data: similar, well-designed models may not exhibit significant differences in their inference results, if they are trained on the same data, which in turn results in reduced effectiveness for differential testing (i.e., smaller I_S as well as smaller $I_S \cap I_F$).

Model Perspective Issue. Differential testing may not be effective if a low-quality model is used. However, it may not always be possible to find an additional model. In extreme cases, the MUT may be the only model that is available, making it burdensome to implement reference models for differential testing.

Our method is proposed to handle these two issues. It creates and cross-references with subspecialized model instances of MUT, without having to other similar model implementations and versions. Figure 1 shows the overview of our approach. We essentially partition the training data into two or more subcategories, and then generate independently trained model instances for each subcategory. Note that these subspecialized reference model instances share the same model architecture as the original MUT: they are just trained using different subsets of the original training data. Intuitively, we would like to use subspecialized model instances that are capable of doing smaller subtasks of MUT particularly better, rather than being performing well in general.

Let us consider the Extended MNIST (EMNIST) [4] dataset as an example: as an extension of the well-known MNIST dataset, EMNIST includes not only handwritten digits but also alphabet letters. Consequently, one possible partitioning would be to categorize the training data into digits and alphabets. For digit inputs, we can compare the output of MUT with the output of digit-only subspecialized model instance, and for alphabet inputs with that of alphabet-only subspecialized model instance.

Figure 3 illustrates how to select test inputs by cross-referencing two subspecialized model instances.

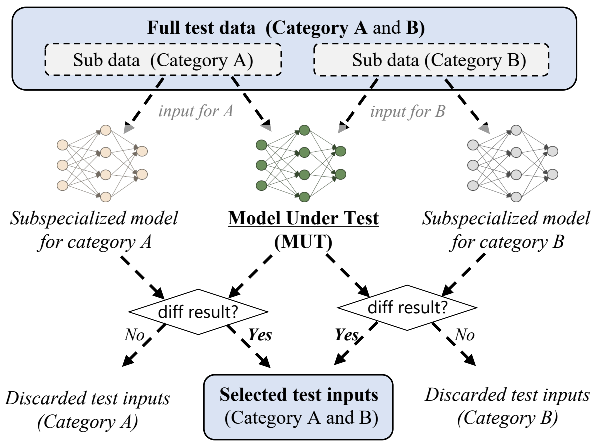

Full test data (Category A and B)
Sub data (Category A)
Sub data (Category B)
input for A
input for B
Subspecialized model for category A
Model Under Test (MUT)
Subspecialized model for category B
diff result?
diff result?
No
Yes
Yes
No
Discarded test inputs (Category A)
Selected test inputs (Category A and B)
Discarded test inputs (Category B)

Figure 3: Test input selection process

Table 1: Analysis of 14,382 EMNIST test inputs for 5 differential testing approaches

Run		1	2	3	4	5	6	7	8	9	10	Average
MyNet-ResNet	Valid (No Effect)	280	307	358	272	266	359	291	365	231	263	299.2
	Effective	491	487	554	480	509	499	468	438	526	507	495.9
	Missed	615	641	601	628	623	646	638	649	645	638	632.4
	Ignored (No Effect)	12,996	12,947	12,869	13,002	12,984	12,878	12,985	12,930	12,980	12,974	12,954.5
MyNet-LeNet	Valid (No Effect)	322	325	352	358	337	315	354	381	293	331	336.8
	Effective	459	466	512	434	491	474	453	437	503	468	469.7
	Missed	647	662	643	674	641	671	653	650	668	677	658.6
	Ignored (No Effect)	12,954	12,929	12,875	12,916	12,913	12,922	12,922	12,914	12,918	12,906	12,916.9
2 Versions	Valid (No Effect)	330	323	306	354	340	290	283	351	299	305	318.1
	Effective	423	406	465	436	446	434	410	372	438	453	428.3
	Missed	683	722	690	672	686	711	696	715	733	692	700.0
	Ignored (No Effect)	12,946	12,931	12,921	12,920	12,910	12,947	12,993	12,944	12,912	12,932	12,935.6
3 Versions	Valid (No Effect)	150	113	119	136	134	116	109	146	102	96	122.1
	Effective	301	274	322	290	324	324	269	268	325	304	300.1
	Missed	805	854	833	818	808	821	837	819	846	841	828.2
	Ignored (No Effect)	13,126	13,141	13,108	13,138	13,116	13,121	13,167	13,149	13,109	13,141	13,131.6
Our Approach	Valid (No Effect)	148	180	154	139	186	148	153	171	163	141	158.3
	Effective	**964**	**972**	**1,010**	**962**	**985**	**984**	**944**	**940**	**1,010**	**983**	**975.4**
	Missed	142	156	145	146	147	161	162	147	161	162	152.9
	Ignored (No Effect)	13,128	13,074	13,073	13,135	13,064	13,089	13,123	13,124	13,048	13,096	13,095.4

3 EXPERIMENT

We present a preliminary evaluation of our proposed method using the EMNIST benchmark [4] and three DL models (two well-known model architectures, and one developed by us).

3.1 Experimental Setup

3.1.1 Dataset. We split the EMNIST [4] dataset into two subcategories: digits and capital letters: we exclude lower case alphabets since their numbers in the training data were insufficient as well as not uniform. Only the ByClass type in the EMNIST dataset provides case-sensitive letter data, so the experiment uses that type.

To avoid data imbalance across classes, we sample up to 2,800 train data and 400 test data per class. As a result, a total of 99,862 training data and 14,382 test data were used for the 36 classes (10 digits and 26 capital letters) in the EMNIST ByClass dataset.

3.1.2 DL Models. We implemented a simple Convolutional Neural Network, called MyNet, and used it as the MUT. Its architecture is shown below in abbreviated pseudocode.

```
def createMyNetModel(n_classes):
  in_tensor = Input(shape=my_input_shape)
  x = Convolution2D(4,activation='relu',...)(in_tensor)
  x = MaxPooling2D(pool_size=(2, 2))(x)
  x = Convolution2D(12,activation='relu',...)(x)
  x = MaxPooling2D(pool_size=(2, 2))(x)
  x = Flatten(name='flatten')(x)
  x = Dense(units=64, activation='relu')(x)
  x = Dense(n_classes)(x)
  x = Activation('softmax')(x)
  model = Model(input_tensor,x)
  return model
```

To compare our subspecialization approach with traditional differential testing, we additionally use two widely studied DL models, ResNet [7] and LeNet [8]. Out of 14,382 EMNIST test images, the trained MUT instances of ResNet, LeNet, and MyNet misclassify 1,025 (7.1%), 1,106 (7.7%), and 1,128 (7.8%), respectively. We can consider these as the upper bound of the number of misclassifications to be found from these instances. All trainings have been performed with batch size of 16, maximum epoch of 100, and an early stopping criterion based on verification loss.

3.2 Experimental Result

We compare our approach with two existing differential testing techniques. The first approach uses ResNet and LeNet as separate reference DL models. The second approach is an N-version approach, for which we train multiple instances of MyNet and compare the original instance of MUT to the additional instances. We set N to be 2 and 3: when $N = 3$, we take the majority output as the correct classification.

Table 1 contains the number of inputs in each partition introduced in Figure 2. We repeat the differential testing of the test images ten times to cater for the randomness, and report the results from all runs as well as the average.

3.2.1 The Error Detection Capability (RQ1). Recall that our MUT instance of MyNet misclassifies 1,128 test images (see Section 3.1.2). The stacked horizontal bar chart in Figure 4 shows the average size of the effective input set, $|I_S \cap I_F|$, as well as the average size of the non-effective (i.e., valid) input set, $|I_S - I_F|$, for each differential testing approach. The numbers right to the bars show the number of inputs that produced different classification results, i.e., $|I_S|$.

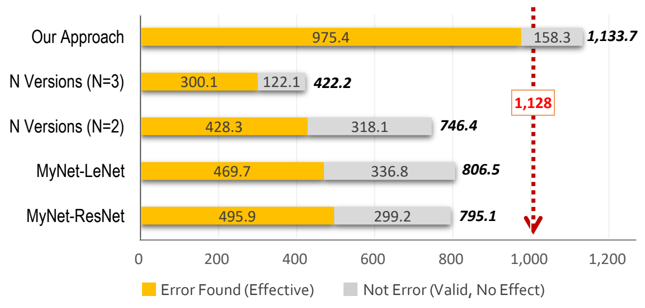


Figure 4: Analysis of test inputs selected with 5 methods

We note that our subspecialized model approach produces the highest $|I_S|$, which is actually greater than 1,128 (i.e., the number of actual misclassifications). All other approaches produce significantly fewer disagreement from differential testing: due to the effect of the majority voting, $N = 3$ produces fewer violations than $N = 2$ for the N-version approach.

Table 2: Analysis of error detection effect and validity

	Similar Model		N-version		Our
	MyNet-ResNet	MyNet-LeNet	N=2	N=3	Approach
$\frac{n(I_S \cap I_F)}{n(I_F)}$	44.0% $(\frac{495.9}{1,128})$	41.7% $(\frac{469.7}{1,128})$	37.9% $(\frac{428.3}{1,128})$	26.6% $(\frac{300.1}{1,128})$	86.4% $(\frac{975.4}{1,128})$
$\frac{n(I_S \cap I_F)}{n(I_S)}$	62.4% $(\frac{495.9}{795.1})$	58.2% $(\frac{469.7}{806.5})$	57.4% $(\frac{428.3}{746.4})$	71.1% $(\frac{300.1}{422.2})$	86.1% $(\frac{975.4}{1,133.7})$

Table 2 presents the error detection rates of each approach. Our approach can detect more than 80% of the total 1,128 errors. While differential testing using ResNet or LeNet is more effective than the N-version approaches, no other approach detects more than 50%.

The bottom row of Table 2 contains the proportion of selected test data that are actually effective. This ratio can be used to determine the validity of the input selection method. Our method shows the highest proportion of 86.1%, showing that input selection using our approach has high precision (i.e., if a disagreement is observed, it is highly likely that the MUT is misclassifying), as well as high recall (i.e., most of the misclassifications made by MUT can be detected by disagreements between MUT and subspecialized models).

3.2.2 Impact of the Test Model Quality on Effectiveness (RQ2). We further investigated whether the quality of the MUT affects the effectiveness of our approach: are we seeing high effectiveness simply because MyNet is a poorly performing classifier? To investigate this, we have also used ResNet and LeNet as the MUT, and applied the subspecialized model approach to these DL models.

Table 3: Analysis of effects according to model quality

	ResNet	LeNet	MyNet
$\|I_F\|$	1,025	1,106	1,128
$\|I_S\|$	1,099	1,119	1,134
$\|I_S \cap I_F\|$	902	958	975
$\frac{\|I_S \cap I_F\|}{\|I_F\|}$	88.0% $(\frac{902}{1,025})$	86.6% $(\frac{958}{1,106})$	86.4% $(\frac{975}{1,128})$
$\frac{\|I_S \cap I_F\|}{\|I_S\|}$	82.1% $(\frac{902}{1,099})$	85.6% $(\frac{958}{1,119})$	86.1% $(\frac{975}{1,134})$

Table 3 shows the results of this investigation, along with the result from MyNet. The numbers in the table are rounded and expressed as natural numbers. There is not significant difference between models. Interestingly, our approach shows the highest error detection rate for ResNet, which is the model with the lowest test classification error rate (7.1%, see Section 3.1.2).

4 LIMITATIONS

One obvious limitation is that our approach requires an additional classification of incoming inputs by the subspecialization boundaries. For some DL applications, this subspecialization classification may take place naturally during the data collection. We expect unsupervised learning to be a good fit, as it can be applied even in cases in which labels are not avaialble.

5 CONCLUSION

We present a novel approach to perform differential testing for DL models without having to implement additional model architectures. Instead, we partition the training data into subgroups, and train subspecialized models that are used for differential testing of inputs belonging to the subgroup. We evaluate our approach using a CNN model trained for EMNIST benchmark as the Model Under Test: the subspecialized models outperform ResNet, LeNet, and alternative N-versions of the MUT, when used as the reference model for differential testing of MUT. The results show the potential of using subspecialized models in the context of differential testing. Future work will consider automated subspecialized group classification for unseen inputs based on unsupervised learning.

ACKNOWLEDGMENTS

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2018-0-00769,Neuromorphic Computing Software Platform for Artificial Intelligence Systems).

REFERENCES

[1] Earl Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2015. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering* 41, 5 (May 2015), 507–525. https://doi.org/10.1109/TSE.2014.2372785

[2] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. 2015. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2722–2730. https://doi.org/10.1109/ICCV.2015.312

[3] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. 2017. Multi-view 3D Object Detection Network for Autonomous Driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6526–6534.

[4] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: an extension of MNIST to handwritten letters. arXiv:1702.05373 [cs.CV]

[5] Martin D. Davis and Elaine J. Weyuker. 1981. Pseudo-Oracles for Non-Testable Programs. In *Proceedings of the ACM '81 Conference (ACM '81)*. Association for Computing Machinery, New York, NY, USA, 254–257.

[6] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. 2018. DLFuzz: Differential Fuzzing Testing of Deep Learning Systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*. Association for Computing Machinery, New York, NY, USA, 739–743.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[9] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. 2017. A survey on deep learning in medical image analysis. *Medical Image Analysis* 42 (2017), 60 – 88.

[10] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. ACM, 1–18.

[11] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering* (2020). https://doi.org/10.1109/TSE.2019.2962027