




Article

Improved Test Input Prioritization Using Verification Monitors with False Prediction Cluster Centroids

Hyekyoung Hwang , Il Yong Chun * and Jitae Shin *

Department of Electrical and Computer Engineering, Sungkyunkwan University,
Suwon 16419, Republic of Korea; ristar1234@skku.edu

* Correspondence: iychun@skku.edu (I.Y.C.); jtshin@skku.edu (J.S.); Tel.: +82-031-290-7153 (J.S.)

Abstract: Deep learning (DL) systems have been remarkably successful in various applications, but they could have critical misbehaviors. To identify the weakness of a trained model and overcome it with new data collection(s), one needs to figure out the corner cases of a trained model. Constructing new datasets to retrain a DL model requires extra budget and time. Test input prioritization (TIP) techniques have been proposed to identify corner cases more effectively. The state-of-the-art TIP approach adopts a monitoring method to TIP and prioritizes based on Gini impurity; one estimates the similarity between a DL prediction probability and uniform distribution. This letter proposes a new TIP method that uses a distance between false prediction cluster (FPC) centroids in a training set and a test instance in the last-layer feature space to prioritize error-inducing instances among an unlabeled test set. We refer to the proposed method as DeepFPC. Our numerical experiments show that the proposed DeepFPC method achieves significantly improved TIP performance in several image classification and active learning tasks.

Keywords: test input prioritization; deep learning; runtime monitoring; image classification; active learning



Citation: Hwang, H.; Chun, I.Y.; Shin, J. Improved Test Input Prioritization Using Verification Monitors with False Prediction Cluster Centroids. *Electronics* **2024**, *13*, 21. <https://doi.org/10.3390/electronics13010021>

Academic Editor: Jian Sun

Received: 23 November 2023

Revised: 15 December 2023

Accepted: 17 December 2023

Published: 19 December 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Deep learning (DL) has been demonstrating cutting-edge performance in various applications [1,2]. Despite the great performance of the DL models, they are prone to errors caused by different factors, including data bias, architectural limitations, and training cost constraints [3,4]. To predict potential errors of trained models and further avoid them, it is critical to test/debug DL models prior to deploying them in practical applications. The most popular way to evaluate trained DL models is to measure performance with a labeled test dataset. One may attempt to test trained models with new datasets, but annotating new datasets is time-intensive and resource-demanding. This is particularly problematic in applications that demand domain-specific expertise for data labeling, such as healthcare and finance applications.

To moderate the aforementioned limitation, test input prioritization (TIP) techniques [5–14] have been proposed to identify a set of “useful” test instances, i.e., ones containing some information that is likely to trigger errors in a trained model. TIP can effectively reduce labeling endeavors by prioritizing instances that potentially exhibit higher mis-predictions in a model. Figure 1 illustrates the general TIP-based testing process for a trained DL model. First, unlabeled test instances are prioritized through TIP based on a labeling budget (e.g., some pre-defined number of instances to label) to construct test oracles. Subsequently, a bug report is generated by using a model’s response with test oracles. This generated bug report is used to retrain a model.

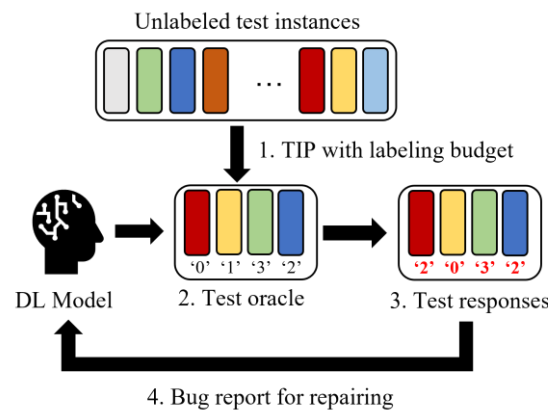


Figure 1. The general TIP-based testing process for a trained model.

In general, TIP for Deep Neural Networks (DNNs) can be classified into three categories: (1) Neuron Coverage (NC) [5–7] that is inspired by traditional software code coverage tests; (2) Surprise Adequacy (SA) [8,9] that uses the level of surprise of an input; and (3) intrinsic function-based methods that use the softmax output of a classifier [10,11].

Recent studies have introduced diverse approaches that fall beyond these categories. For example, PRIMA [12] utilizes mutation analysis and TestRank [13] adopts graph representation. Also, DeepAbstraction (DA) [14] proposed a two-stage prioritization technique that combines single metric-based TIP with a runtime verification monitor.

The runtime verification monitor proposed in [15] consists of a set of intervals extracted from correctly classified instances in a training set, where two endpoints of each interval are estimated from each dimension of features in the last layer feature space. In an inference, the verification monitor checks whether the values of a logit vector (i.e., pre-softmax output) for a test instance fall within the intervals established from correct predictions above or not. Then, the authors of [16] extended the runtime verification monitor to use a set of intervals extracted from both correctly and incorrectly classified instances in a training set. In an inference, the monitor determines whether the values of a test instance’s logit vector fall within the intervals of correct predictions or incorrect predictions. Subsequently, a test instance is categorized into one of three groups: “accept”, “uncertain”, or “reject”.

DA employs a runtime verification monitor [16] to evaluate the error-revealing properties of test instances. It uses the aforementioned runtime verification monitor as its initial prioritization stage and prioritizes test instances based on their categorized results: “reject”, “uncertain”, and “accept”. Within each group, samples are further prioritized with the Gini impurity of each instance.

This research introduces a new scoring function for runtime verification monitor-based TIP approaches. The proposed scoring function prioritizes the test instances based on the similarity between the feature of a test instance and that of the centroids of the false prediction cluster (FPC) in a training set. We propose a TIP technique that replaces the Gini impurity in [9] with the proposed scoring function. We refer to the proposed TIP approach as DeepFPC. The proposed DeepFPC approach does not require additional costs to provide FPC centroids, as they are intermediate results in the monitor construction process. To summarize, this research has the following contributions:

- We propose a new scoring function for TIP based on runtime verification monitors that uses the distance between FPC centroids from training data and an unlabeled test instance.
- We analyze the proposed DeepFPC method based on the intra-class feature compactness and inter-class feature separability of a training set.
- The proposed DeepFPC approach shows significantly better performance than the existing TIP methods in image classification and active learning with several datasets.

The remainder of the paper is organized as follows. Section 2 elaborates the proposed method, DeepFPC. Section 3 provides experimental setups and results with related discussion. Finally, in Section 4, we conclude the paper.

2. Methods

This section describes the proposed DeepFPC approach in detail. DeepFPC consists of a runtime verification monitor from [16] and a proposed scoring function referred to as DistFPC. Figure 2 depicts the overall process of the proposed DeepFPC method.

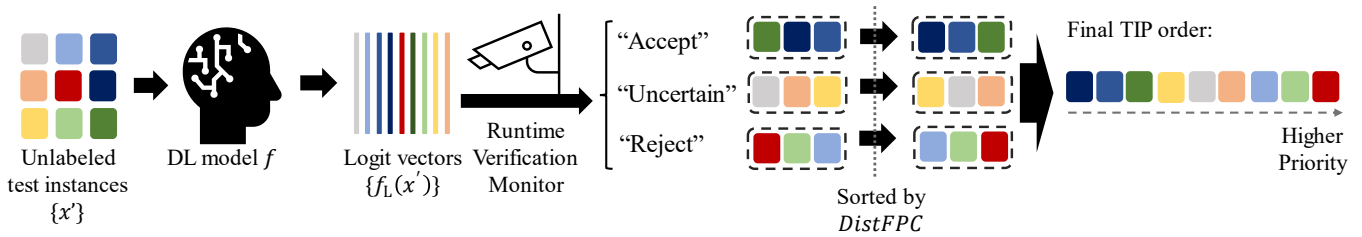


Figure 2. An overall process of the proposed DeepFPC method, which contains a runtime verification monitor and DistFPC.

We consider the classification setup with C classes. Let X_{tr} and X_{te} denote the training and the test set, respectively. We have N training instances, $X_{tr} = \{(x_1, y_1), \dots, (x_N, y_N)\}$, and M test instances, $X_{te} = \{x'_1, \dots, x'_M\}$, where x is an input sample with the corresponding label y and x' is an unlabeled input. X_{tr} is partitioned into C subsets based on the labels y resulting in $X_{tr} = \{X_1, X_2, \dots, X_C\}$.

Let $f = g_1 \circ g_2 \circ \dots \circ g_L$ be a neural network (NN) such that $f_l(x)$ is the output at the l th layer by taking x as an input, where g_l corresponds to the l th layer, with $l = 1, \dots, L$ and L represents the number of layers. Note that $f_L(x)$ is a logit vector (i.e., pre-softmax output) of f when given x as an input. For simplicity, we denote $f(x)$ as the classification result, F as the set of $f_L(x)$, and $|f_l|$ as the number of neurons (i.e., the dimension of features) in the l th layer of f .

2.1. Runtime Verification Monitor

The runtime verification monitors proposed in [16] employ a box abstraction technique [15] to the features of the training set. The monitors aim to detect out-of-distribution unlabeled test inputs for the classification system.

The box abstraction for class i in [15] is as follows: Given a fixed classification model f , the set of logit vector F is collected from the training samples of class i . It is then divided into two distinct groups based on the correctness of $f(x)$: F_i^{co} and F_i^{inc} . The former includes logit vectors of samples for which both ground truth and prediction are class i , while the latter comprises logit vectors of samples for which the ground truth class is i but the prediction is not.

Each of these groups, F_i^{co} and F_i^{inc} , is subdivided into P and S clusters using a clustering algorithm (e.g., K -Means). Within each cluster, the minimum and maximum values for each feature dimension are extracted and the intervals for all feature dimensions are constructed. The union of these intervals forms the box abstraction for class i :

$$B_i^{co} = \left\{ [a_j, b_j] \mid a_j = \min_{1 \leq k \leq P} F_i^{co}[j, k], b_j = \max_{1 \leq k \leq P} F_i^{co}[j, k], j \in [|f_L|] \right\} \quad (1)$$

$$B_i^{inc} = \left\{ [a_j, b_j] \mid a_j = \min_{1 \leq k \leq S} F_i^{inc}[j, k], b_j = \max_{1 \leq k \leq S} F_i^{inc}[j, k], j \in [|f_L|] \right\} \quad (2)$$

where $F_i^{co}[j, k]$ represents the j th feature dimension in the k th cluster of correct predictions, while $F_i^{inc}[j, k]$ corresponds to the j th feature dimension in the k th false prediction cluster. The collections of intervals B_i^{co} and B_i^{inc} where $i = 1, \dots, C$ are called *monitors*. The former

is extracted from the logit values of correct prediction clusters and the latter is extracted from the logit values of false prediction clusters.

Given a test instance x' and its classification result $f(x) = k$, the monitors classify x' into three groups: “accept” if all values of $f_L(x')$ fall within the intervals $B_k^{\text{co}} : \forall k$, “reject” if all values of $f_L(x')$ are within the intervals $B_k^{\text{inc}} : \forall k$, or “uncertain” for all other cases. The “reject” group is expected to include most incorrect predictions, while the “accept” group contains the fewest number of incorrect predictions.

2.2. DeepFPC

For simplicity, we suppose that given a test instance x' and its classification result $f(x) = k$, a collection of intervals B_k^{inc} is extracted from S numbered false prediction clusters. A set of FPC centroids of class c' is denoted as $c_k = \{c_k^1, \dots, c_k^S\}$.

The proposed DeepFPC method is a two-stage TIP approach, similar to DA [14]. We adopt the runtime verification monitor as our initial prioritization stage. We prioritize unlabeled instances in the order of their monitored results: “reject”, “uncertain”, and “accept”. Instead of prioritizing samples within each group based on Gini impurity, we propose a new scoring function that measures the similarity of a test instance feature and false prediction cluster centroids. This new scoring function is called DistFPC.

FPCs are intermediate results derived from the monitor’s construction, specifically within the clustering process. We use the centroid of FPCs as an alternative representation of false predictions within the training set. Thus, we conjecture that a sample with a smaller DistFPC value is more erroneous than a larger one.

To quantify the similarity between $f_L(x')$ and FPC centroids, we evaluate both distance and angular separation between the two. We measure the Euclidean distance between $f_L(x')$ and each of the FPC centroids of class k . We expect that the smaller values indicate that x' is closer to the FPC centroids, making it more prone to errors. Figure 3 is an example showing the positional relationship between the s -th FPC c_k^s and logit vectors extracted from different inputs x'_1 , x'_2 , and x'_3 in logit space.

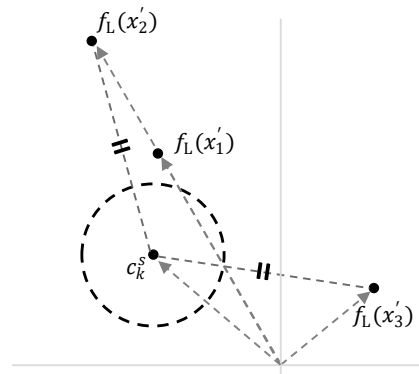


Figure 3. An example of the positional relationship between the s -th FPC c_k^s and three different unlabeled inputs x'_1 , x'_2 , and x'_3 in logit space.

The purpose of TIP is to prioritize samples that are close to false predictions. Therefore, when the similarity between a false prediction centroid and the logit vector is measured by only reflecting the Euclidean distance, x'_1 is prioritized first because it is closest to c_k^s . And x'_2 and x'_3 have the same priority since they have same Euclidean distance to c_k^s . However, according to [12], the direction of a logit vector determines the final prediction. Thus, from our assumption and the observations in [12], x'_2 is more likely to be a false prediction than x'_3 since the angular difference between c_k^s and x'_2 is smaller than that of c_k^s and x'_3 .

To measure the angular difference between $f_L(x')$ and the FPC centroids of class k , we calculate the angle between $f_L(x')$ and each of the FPC centroids of class k . We measure the cosine similarity between $f_L(x')$ and a FPC centroid and estimate the inverse cosine to the cosine similarity.

Similar to the Euclidean distance, a smaller angular difference indicates a larger similarity to false prediction. Thus, we multiply both the Euclidean and angular distance metric and take the minimum of it to make the proposed DistFPC:

$$DistFPC(x') = \min_{s \in [S]} \left(\sqrt{(f_L(x') - c_k^s)^2} \times \arccos \frac{f_L(x') \cdot c_k^s}{\|f_L(x')\| \|c_k^s\|} \right) \quad (3)$$

We anticipate that a sample with a smaller DistFPC value is more likely to cause errors in the DL model. Consequently, test instances within each group are prioritized in ascending order of DistFPC values.

2.3. Intra-Class Feature Compactness and Inter-Class Feature Separability

The proposed scoring function, DistFPC, measures the similarity between a test instance feature and the centroids of the FPCs. We expect that the proposed DeepFPC approach exhibits high TIP performance when the model forms well-separable clusters within the training set.

According to observations from [17], a trained classifier can exhibit well-separable clusters when it meets two conditions: minimal intra-class compactness of features (i.e., features of the same class are close to each other) and maximal inter-class separability of features (i.e., features of different classes fall apart from each other). We define a matrix that represents these conditions of the trained classifier:

$$D_{tr} = \begin{bmatrix} d_{1,1} & \cdots & d_{1,C} \\ \vdots & \ddots & \vdots \\ d_{C,1} & \cdots & d_{C,C} \end{bmatrix} \quad (4)$$

where $d_{i,j}$ represents the averaged cosine similarity between the logit vectors of samples predicted as class i and class j .

When $i = j$, $d_{i,j}$ indicates the similarity between the logit vectors of the same predictions, a value near 1 implies a higher compactness of the features within the same class. Otherwise, $d_{i,j}$ represents the similarity between the logit vectors of the different predictions. A value near -1 suggests greater separability of features between two different classes. Figure 4 displays the D_{tr} for the two experiments in Table 1. Figure 4a represents the D_{tr} of experiment A and Figure 4b shows it for experiment D.

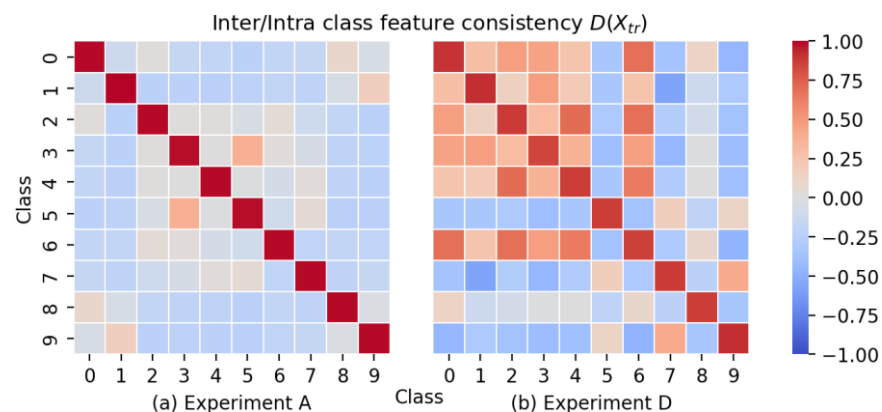


Figure 4. Visualization of D_{tr} for experiment A (a) and D (b) in Table 1.

Table 1. Experiment IDs with datasets, models, and the corresponding accuracy on train test set, intra-class compactness ($\mu_{d_{i=j}}$), and inter-class separability ($\mu_{d_{i\neq j}}$).

ID	Dataset	Model	Train Acc (%)	Test Acc (%)	$\mu_{d_{i=j}} \uparrow$	$\mu_{d_{i\neq j}} \downarrow$
A	CIFAR10	MobilenetV2	99.51	93.38	0.99	−0.114
B	CIFAR10	ResNet50	98.69	93.81	0.98	−0.105
C	MNIST	ResNet18	98.03	97.66	0.88	−0.061
D	FMNIST	WideResNet50	94.01	87.7	0.86	0.001
E	CIFAR100	EfficientNet V2-S	99.52	88.2	0.84	0.002
F	CIFAR10-C	ResNet50	-	77.29	-	-
G	MNIST-C	ResNet18	-	70.19	-	-
H	FMNIST-C	WideResNet50	-	47.83	-	-

In experiment A, the value of diagonal elements $d_{i=j}$ are closer to 1 than those of experiment D. This indicates that the models in experiment A have a higher compactness of features within the same class. In the case of the value of non-diagonal elements $d_{i\neq j}$, few of them in experiment A exceed zero, while a few of them in experiment D exceed zero, suggesting lower inter-class separability for experiment D. Thus, we expect that the proposed method will exhibit more effective TIP performance in experiment A. Our experiments confirm that the proposed method shows better performance when the inter-class feature separability value is less than 0, as shown in Tables 1 and 2.

Table 2. Comparison between different TIP methods for different image classification datasets (in ATPF (%), see task IDs in Table 1. **Bold** numbers indicate the top ATPF value.

ID	Gini [10]	DSA [8]	MLSA [9]	NBC [5]	NLC [6]	FD+ [7]	DA [14]	DeepFPC
A	54.53	53.85	52.3	51.835	34.818	70.37	78.88	87.26
B	52.62	47.58	48.48	56.071	40.109	35.46	71.61	78.81
C	51.8	63.79	56.94	7.51	3.13	46.58	64.36	69.06
D	59.13	55.66	41.63	9.45	11.79	35.48	70.13	68.56
E	61.65	60.11	61.39	10.62	12.26	28.81	60.4	58.18
F	66.33	29.13	31.17	52.74	25.29	57.57	78.0	83.61
G	58.68	55.82	56.96	71.23	49.32	66.83	85.41	86.08
H	73.58	71.29	73.37	71.18	70.16	49.06	92.39	95.11

average inter-class separability becomes positive, the performance of our method diminishes. Notably, in experiment E, Gini stands out as the most effective TIP method and DA and our proposed method are not effective in this context. This is because experiment E deals with 100 different classes in the training set, with a total of 288 false predictions observed in the training set. This leads to the performance degradation of DA and the proposed DeepFPC method, which include the monitoring process in their algorithm. The most important part in the monitoring process to achieve the goal of TIP is to accurately classify the “reject” group. The “reject” group is classified by the monitor which is produced through false prediction clusters. Yet, in the case of experiment E, the runtime verification monitor that classifies the “reject” group formed inappropriately due to the lack of false prediction instances in the training set. Also, the proposed scoring function DistFPC is dependent on the FPC centroids. Experiments F, G, and H demonstrate that the proposed method can be a proper option for identifying error-inducing samples in real-world scenarios, namely those containing data corruptions.

3. Experimental Results

In this section, we demonstrate the effectiveness of the proposed DeepFPC approach in image classification and active learning. We ran all the experiments using an Intel(R) Xeon(R) Silver 4114 CPU at 2.20 GHz and an NVIDIA GeForce RTX 2080 Ti with 12 GB of memory using PyTorch v1.12.0.

3.1. Experimental Setups

3.1.1. Datasets and Models

To evaluate the proposed method, we used four datasets (MNIST [18], Fashion-MNIST (FMNIST) [19], CIFAR10, CIFAR100 [20]) which are widely used for image classification benchmarks. We tested different TIP methods with diverse models: ResNet18, ResNet50 [21], MobileNetV2 [22], WideResNet50-2 [23], and EfficientNetV2-S [24].

To evaluate TIP performance in real-world applications, we used MNIST, FMNIST, and CIFAR10 datasets with real-world corruptions [25]. We denote the corrupted datasets by adding the '-C' suffix to their names. We used 15 different types of corruption, such as motion blur, rotation, and brightness, each of which contained five different severities.

Table 1 shows our experimental setups. Columns $\mu_{d_{i=j}}$, $\mu_{d_{i \neq j}}$ indicate the average of diagonal and non-diagonal elements in D_{tr} . For experiment IDs F, G, and H, we used the same model with B, C, and D, respectively, to check the effectiveness of TIP in a corrupted environment. We employed experiment IDs C, D, G, and H to evaluate TIP performance in active learning.

3.1.2. Compared Methods

We compared the proposed DeepFPC approach with the following eight existing TIP methods that are introduced in Section 2.

- NC-based methods: NBC [5], NLC [6], and FD+ [7];
- SA-based methods: DSA [8] and MLSA [9];
- Intrinsic function methods: Gini [10];
- Others: DeepAbstraction [14].

3.1.3. Evaluation Metrics

In the classification experiments, we utilized the average test percentage of fault (ATPF) metric [13] to evaluate the error-inducing sample identification performance of TIPs.

$$\text{ATPF}(\%) = 100 \times \frac{1}{N_{\text{fail}}} \sum_{n=1}^{N_{\text{fail}}} \frac{N_{\text{err},n}}{n} \quad (5)$$

where, N_{fail} represents the total number of incorrect predictions in a test set, and $N_{\text{err},n}$ denotes the number of detected incorrect predictions within the n th labeling budget.

In the active learning experiments, we followed the experimental setups described in [26]. The corrupted data are referred to as out-of-distribution (OOD) data, while non-corrupted data are called nominal. The test set was randomly divided into two equal subsets, called the 'active' and 'evaluation' splits, each allocated 50% of the dataset. The subset of the 'active' split, chosen based on TIP values, was integrated into the training set for model retraining. For the nominal and OOD active splits, we selected 20% and 10% of the samples, respectively. For fair comparison, we adopted standard supervised learning with the same hyperparameter settings in all active learning experiments. We used cross entropy loss with the SGD optimizer, where the learning rate was 1e-4 and the batch size was 128. We completed model retraining when the output value of the loss function did not improve for five epochs. Then, we evaluated the performance of the retrained model on both the nominal and OOD evaluation splits. Figure 5 shows the entire active learning experiment process.

3.2. Comparisons between Different TIP Methods with the Perspective of Identifying Test Instances with Errors

Table 2 presents a comparison of ATPF values for various TIP methods across eight distinct image classification experiments. A TIP approach with a higher ATPF value is more effective in identifying instances that induce errors while remaining within the same labeling budget. The difference in ATPF values between Gini [10] and DA [14] is attributed to the use of the runtime verification monitor in DA. The performance variation between

DA and the proposed DeepFPC method results from substitution of the scoring function with DistFPC.

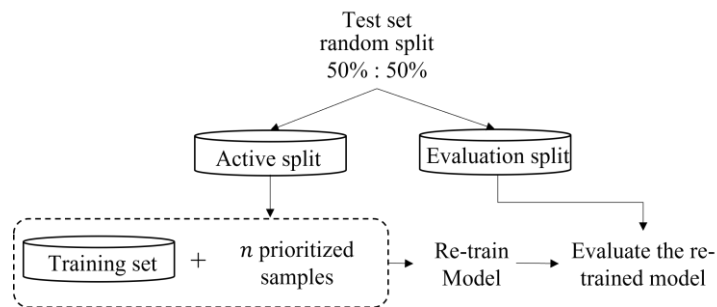


Figure 5. Experimental setup for TIP evaluation in active learning, described in [26].

Table 2 shows that the proposed method outperforms existing TIP methods in six of eight different experiments. In particular, DeepFPC exhibits superior performance when intra-class compactness approaches 1 and the inter-class separability takes on negative values, as observed in experiments A to C. However, in experiments D and E, where the

The ablation study of the proposed scoring function is shown in Table 3. Each column specifies the experiment ID. The first row shows the ATPF value when using the angular term only, while the second row represents the case only with the Euclidean term. According to Table 3, the tendency of ATPF values is consistent with that of DeepFPC even when a single term is used in DistFPC. It achieves higher values when the intra-class compactness approaches 1 and inter-class separability nears -1 . Also, using the angular term only shows higher ATPF values than using the Euclidean term only. However, using both angular and Euclidean terms together obtains the highest ATPF values.

Table 3. Ablation study of the proposed DeepFPC method for image classification (in ATPF (%)).

	A	B	C	D	E	F	G	H
Angular term only	82.44	77.94	63.92	65.26	55.24	81.22	82.39	91.28
Euclidean term only	72.96	72.85	67.04	66.79	49.25	77.94	75.89	77.23
DeepFPC	87.26	78.81	69.06	68.56	58.18	83.61	86.08	95.11

The evaluation results for active learning setups are presented in Table 4. Each column specifies the dataset and models that were used for active learning experiments. The sub-columns labeled “Nominal” or “OOD” indicate which type of active split was used to extend the training set and the split used to retrain the model. Each row corresponds to the TIP method used for sample selection from the active split. The sub-rows labeled “Nominal” or “OOD” indicate which type of evaluation split was used to evaluate the retrained model.

The results in Table 4 show that the proposed DeepFPC method consistently achieves top 3 retraining accuracy values in all active learning setups, exceeding the state-of-the-art existing TIP method, DA. The result is consistent with that of Table 2. The monitor-based TIP techniques show the most frequent accuracy improvement in active learning, and the SA-based method and intrinsic function-based method follow. Also, the result demonstrates that the NC-based method is the most ineffective in the context of active learning, which aligns with the findings reported in [26].

Table 4. Comparisons of performance between different TIP methods for active learning experiments. **Bold** numbers indicate the top 3 most accurate values in four different (active, evaluation) split setups: (Nominal, Nominal), (Nominal, OOD), (OOD, Nominal), (OOD, OOD).

TIP Metrics		Type of Evaluation Split	MNIST—ResNet18		FMNIST—WideResNet50	
			Type of Active Split		Type of Active Split	
			Nominal	OOD	Nominal	OOD
Random Selection		Nominal	91.18	69.26	89.46	49.26
		OOD	93.48	78.72	88.42	58.52
Intrinsic Function	Gini [10]	Nominal	93.92	74.14	90.78	49.54
		OOD	93.34	80.32	89.04	66.94
Surprise Adequacy	DSA [8]	Nominal	90.74	69.88	90.78	51.26
		OOD	94.8	81.82	87.54	68.24
	MLSA [9]	Nominal	91.78	69.88	91.94	51.96
		OOD	93.78	73.12	88.5	62.98
Neuron Coverage	NBC [5]	Nominal	90.28	67.48	91.12	51.04
		OOD	92.98	79.56	87.64	62.94
	NLC [6]	Nominal	91.94	70.38	87.70	41.38
		OOD	89.5	73.54	91.04	61.44
	FD+ [7]	Nominal	91.88	71.06	89.8	52.44
		OOD	91.18	72.98	87.1	60.5
Monitor-Based	DA [14]	Nominal	93.92	74.36	92.32	52.34
		OOD	93.6	80.6	88.34	65.18
	DeepFPC	Nominal	92.2	74.16	92.48	52.44
		OOD	93.8	81.60	88.65	65.39

3.3. Inference Time Comparisons with Different TIP Methods

In this section, we compare the computation complexity of TIP techniques during an inference. It is important to remember that we have S FPCs and M test samples with C classes. $|f_l|$ represents the number of neurons (i.e., the dimension of features) in the l th layer of DNN model f .

The intrinsic function-based methods only require the softmax output of a sample, so their computational cost is $\mathcal{O}(M)$. NC-based methods require the activation value of total neurons for every single sample, which requires $\mathcal{O}(M \cdot \sum |f_l|)$ for the whole test set. DSA compares a feature of a test sample with training set features so that the complexity cost for the total test samples can be inferred as $\mathcal{O}(M \cdot N)$.

The computational cost of the monitoring process in both DA and DeepFPC is $\mathcal{O}(M \cdot S)$. After the monitoring process is applied to the test instances, DA prioritizes the samples within each group with the Gini impurity so that the additional cost of the scoring function is $\mathcal{O}(M)$. However, DeepFPC calculates similarity between a test instance feature and S number of FPC centroids, which requires $\mathcal{O}(M \cdot S)$ computations in total.

Table 5 reports the inference time of three different TIP techniques (Gini impurity, DA, and the proposed DeepFPC) for experiments A–H. We do not report the inference times of other TIP methods since they took more than 8 s for all experiments. DA and DeepFPC took a longer time than the Gini method, as it does not use the monitoring process. The longer inference time of DeepFPC compared to DA is caused by substitution of the scoring function from Gini Impurity with the proposed DistFPC.

Table 5. Comparison of time consumption (in sec.) for each experimental setup.

	A	B	C	D	E	F	G	H
Gini [10]	0.0011	0.0009	0.0004	0.0010	0.0007	0.0121	0.0009	0.0012
DA [14]	0.0261	0.0270	0.0189	0.0272	3.125	0.0216	0.0324	0.0273
DeepFPC	0.0354	0.0407	0.0311	0.0651	7.221	0.0482	0.0612	0.0427

3.4. Sample Visualization

In this section, we describe the differences between the samples with different prioritization orders. In TIP, the lower priority sample is regarded as an easy sample that the trained model can provide correct prediction for, while higher priority samples are hard samples that can induce a false prediction from the model.

Figure 6 represents a sample visualization based on different prioritization orders with different conditions. Figure 6a,b are samples without real-world data corruptions, while Figure 6c,d are samples that contain the real-world data corruptions. Figure 6a,c show samples with lower priority.



Figure 6. Sample visualization of CIFAR10 and CIFAR10-C with ResNet50 based on different prioritization orders that are based on the proposed DeepFPC method: (a,b) CIFAR10, (c,d) CIFAR10-C.

The lower priority samples in both the clean and corruption domains have a center-aligned object that is distinguishable with a background. Also, samples with less severity corruptions are lower in the prioritization order. The model predictions on lower priority samples are all correct.

Figure 6b,d are samples with higher priority. The higher priority samples in both the clean and corruption domains have a non-center-aligned object or part of the object is visible. Additionally, the objects in higher priority samples are hard to distinguish from their backgrounds. Samples with higher severity corruptions are higher in the prioritization order, and the model predictions upon them are all incorrect.

4. Conclusions and Future Works

Testing and debugging DL models prior to deployment requires resource-intensive test data labeling. TIP approaches can reduce labeling efforts by prioritizing instances with greater potential for mispredictions, thereby generating bug reports for model retraining. In this paper, we proposed a new scoring function for monitor-based TIP techniques that prioritizes error-inducing samples based on similarity to FPC centroids in the training set. The proposed DeepFPC method achieves significantly improved performance compared to existing representative/state-of-the-art TIP methods, both in image classification and active learning.

Additionally, we found that the TIP approaches can be candidates for data selection in life-long learning due to their effectiveness in active learning and visually confirmed that sample priority is related to image quality. When considering the purpose of TIP techniques, one factor is that the technique should prioritize error-inducing samples within unlabeled test instances, and we will further analyze the effectiveness of applying TIP metrics in image quality assessment and continual learning.

Due to storage limitations, we evaluated the proposed DeepFPC method and existing TIP techniques with image classification datasets containing relatively small input resolutions. Yet, images acquired in the real world are often larger than those used. Therefore, in future research, we will verify the real-world applicability of TIP techniques by evaluating them with additional complex model architectures, such as vision transformers [27], and larger datasets (e.g., ImageNet [28] and its variants [29,30]).

Author Contributions: Conceptualization, H.H.; methodology, H.H.; software, H.H.; validation, H.H.; formal analysis, H.H. and J.S.; resources, H.H.; data curation, H.H.; writing—original draft preparation, H.H. and I.Y.C.; writing—review and editing, I.Y.C. and J.S.; visualization, H.H.; supervision, I.Y.C. and J.S.; funding acquisition, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by a National Research Foundation of Korea (NRF) Grant funded by the Korean Government Ministry of Science and ICT (MSIT) (NRF-2020R1F1A1065626), in part by the MSIT under the Information Technology Research Center (ITRC) support program (IITP-2023-2018-0-01798) supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP).

Data Availability Statement: The data presented in this study are openly available in reference number [18–20,25].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yang, W.; Hongwei, G.; Yueqiu, J.; Xin, Z. A Novel Approach to Maritime Image Dehazing Based on a Large Kernel Encoder–Decoder Network with Multihead Pyramids. *Electronics* **2022**, *11*, 3351. [[CrossRef](#)]
2. Dhanya, V.; Subeesh, A.; Kushwaha, N.; Vishwakarma, D.K.; Kumar, T.N.; Ritika, G.; Singh, A. Deep learning based computer vision approaches for smart agricultural applications. *Artif. Intell. Agric.* **2022**, *6*, 211–229. [[CrossRef](#)]
3. Ahmed, W.; Morerio, P.; Murino, V. Cleaning noisy labels by negative ensemble learning for source-free unsupervised domain adaptation. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV 2022), Waikoloa, HI, USA, 4–8 January 2022; pp. 1616–1625.
4. Vardi, G. On the implicit bias in deep-learning algorithms. *Commun. ACM* **2023**, *66*, 86–93. [[CrossRef](#)]
5. Ma, L.; Felix, J.; Fuyuan, Z.; Jiyuan, S.; Minhui, X.; Bo, L.; Chunyang, C.; Ting, S.; Li, L.; Yang, L. Deepgauge: Multi-Granularity Testing Criteria for Deep Learning Systems. In Proceedings of the 33rd ACM/IEEE international conference on Automated Software Engineering (ASE 2018), Montpellier, France, 3–7 September 2018; pp. 120–131.
6. Yuan, Y.; Pang, Q.; Wang, S. Revisiting Neuron Coverage for Dnn Testing: A Layer-Wise and Distribution-Aware Criterion. In Proceedings of the IEEE/ACM 45th International Conference on Software Engineering (ICSE 2023), Melbourne, Australia, 14–20 May 2023; pp. 1200–1212.
7. Yan, R.; Chen, Y.; Gao, H.; Yan, J. Test Case Prioritization with Neuron Valuation Based Pattern. *Sci. Comput. Program.* **2022**, *215*, 102761. [[CrossRef](#)]
8. Kim, J.; Feldt, R.; Yoo, S. Guiding Deep Learning System Testing Using Surprise Adequacy. In Proceedings of the IEEE/ACM 41st International Conference on Software Engineering (ICSE 2019), Montreal, QC, Canada, 24–31 May 2019; pp. 1039–1049.

9. Kim, S.; Yoo, S. Multimodal Surprise Adequacy Analysis of Inputs for Natural Language Processing Dnn Models. In Proceedings of the IEEE/ACM International Conference on Automation of Software Test (AST 2021), Madrid, Spain, 20–21 May 2021; pp. 80–89.
10. Feng, Y.; Shi, Q.; Gao, X.; Wan, J.; Fang, C.; Chen, Z. Deepgini: Prioritizing Massive Tests to Enhance the Robustness of Deep Neural Networks. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2020), Virtual. 18–22 July 2020; pp. 177–188.
11. Ma, W.; Papadakis, M.; Tsakmalis, A.; Cordy, M.; Traon, Y.L. Test Selection for Deep Learning Systems. *ACM Trans. Softw. Eng. Methodol. TOSEM* **2021**, *30*, 1–22. [[CrossRef](#)]
12. Wang, Z.; You, H.; Chen, J.; Zhang, Y.; Dong, X.; Zhang, W. Prioritizing Test Inputs for Deep Neural Networks Via Mutation Analysis. In Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE 2021), Virtual. 25–28 May 2021; pp. 397–409.
13. Li, Y.; Li, M.; Lai, Q.; Liu, Y.; Xu, Q. Testrank: Bringing Order into Unlabeled Test Instances for Deep Learning Tasks. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Virtual. 6–14 December 2021; pp. 20874–20886.
14. Al-Qadasi, H.; Wu, C.; Falcone, Y.; Bensalem, S. Deepabstraction: 2-Level Prioritization for Unlabeled Test Inputs in Deep Neural Networks. In Proceedings of the IEEE International Conference On Artificial Intelligence Testing (AITest 2022), Newark, CA, USA, 15–18 August 2022; pp. 64–71.
15. Cheng, C.; Nührenberg, G.; Yasuoka, H. Runtime Monitoring Neuron Activation Patterns. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE 2019), Florence, Italy, 25–29 March 2019; pp. 300–303.
16. Wu, C.; Falcone, Y.; Bensalem, S. Customizable Reference Runtime Monitoring of Neural Networks Using Resolution Boxes. In Proceedings of the 23rd International Conference on Runtime Verification (RV 2023), Thessaloniki, Greece, 3–6 October 2023; pp. 23–41.
17. Liu, W.; Longhui, Y.; Adrian, W.; Bernhard Schölkopf. Generalizing and Decoupling Neural Collapse Via Hyperspherical Uniformity Gap. *arXiv* **2023**, arXiv:2303.06484.
18. Deng, L. The Mnist Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [[CrossRef](#)]
19. Xiao, H.; Kashif, R.; Roland, V. Fashion-Mnist: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
20. Krizhevsky, A.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. Master’s Thesis, University of Tront, Toronto, ON, USA, 2009.
21. He, K.M.; Zhang, X.Y.; Ren, S.Q.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
22. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. Mobilenetv2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR 2018), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
23. Zagoruyko, S.; Nicos, K. Wide Residual Networks. *arXiv* **2016**, arXiv:1605.07146.
24. Tan, M.; Le, Q. Efficientnetv2: Smaller Models and Faster Training. In Proceedings of the International Conference on Machine Learning (ICML 2021), Virtual. 18–24 July 2021; pp. 10096–10106.
25. Hendrycks, D.; Thomas, D. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *arXiv* **2019**, arXiv:1903.12261.
26. Weiss, M.; Tonella, P. Simple Techniques Work Surprisingly Well for Neural Network Test Prioritization and Active Learning (Replicability Study). In Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2022), Virtual. 18–22 July 2022; pp. 139–150.
27. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. In Proceedings of the 9th International Conference on Learning Representations (ICLR 2021), Virtual, 3–7 May 2021.
28. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), Miami, FL, USA, 20–25 June 2009; pp. 248–255.
29. Hendrycks, D.; Zhao, K.; Basart, S.; Steinhardt, J.; Song, D. Natural adversarial examples. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2021), Virtual. 19–25 June 2021; pp. 15262–15271.
30. Li, X.; Chen, Y.; Zhu, Y.; Wang, S.; Zhang, R.; Xue, H. ImageNet-E: Benchmarking Neural Network Robustness via Attribute Editing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 20371–20381.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.