WP-GEB-Estimator User Manual

∼ WP-GEB : Weight-Perturbed Generalization Error Bounds ∼

Yoshinao Isobe
National Institute of Advanced Industrial Science and Technology
y-isobe@aist.go.jp

30th April, 2024

Contents

1 Introduction

The "WP-GEB" in the tool name WP-GEB-Estimator stands for *weight-perturbed generalization error bounds* and WP-GEB-Estimator is a tool for estimating the upper bounds of *randomly* weight-perturbed generalization errors and the upper bounds of *worst* weight-perturbed generalization errors of *neural classifiers*[1] (hereafter called just *classifiers*). The weight-perturbed generalization errors represent the expected values of the misclassification-rates of classifiers when perturbations are added on weight-parameters between neurons during inferences for *any* input. *Random perturbations* are randomly selected from uniform distribution with specified range, while *worst perturbations* are selected towards misclassification within the range. Although even worst perturbations do not necessarily cause misclassification, perturbations really causing misclassification are called *adversarial perturbations*.

Machine Learning Quality Management (MLQM) Guideline [1] has been developed to clearly explain the quality of various industrial products including statistical machine learning. This guideline defines an internal quality property, called *the stability of trained models*, which represents that trained-models reasonably behave even for unseen input data. Figure 1 shows the techniques to evaluate and improve stability for each phase and each level, which is Figure 14 in the MLQM-Guideline version 4[2]. WP-GEB-Estimator is a useful tool for evaluating the stability of trained classifiers in the evaluation phase because it has the following functions:

- `[Noise robustness]` It can measure misclassification-rates of randomly weight-perturbed classifiers for a test-dataset and a perturbation sample.
- `[Adversarial attack]` It can search for adversarial weight-perturbations for each input in a test-dataset.
- `[Adversarial verification]` It can statistically guarantee with a confidence for any input that the existence-probability of adversarial weight-perturbations is less than a threshold.
- `[Generalization error]` It can statistically guarantee with a confidence that the weight-perturbed generalization error (i.e. for any input including unseen input) is less than an upper bound.

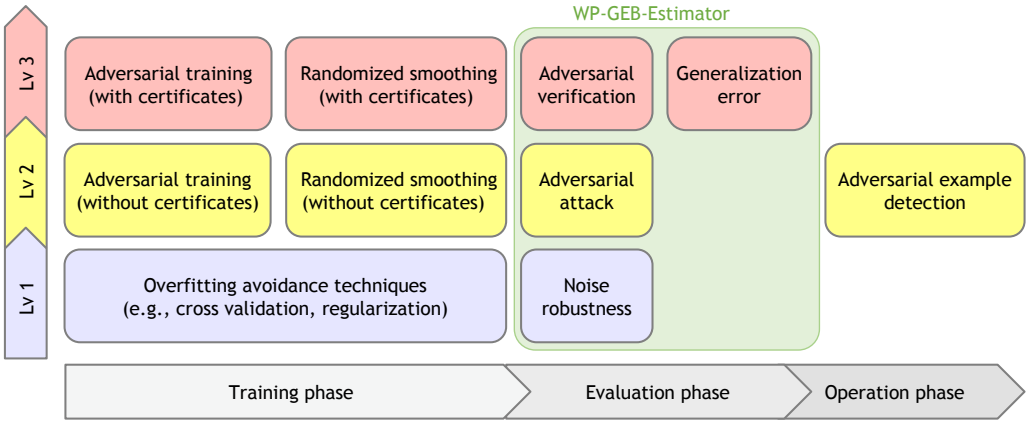

Figure 1: Techniques to evaluate and improve stability (Fig.14 in MLQM-Guideline [1])

2 Introduction to WP-GEB

In this section, weight-perturbed generalization error bounds (WP-GEB) are defined in Subsection 2.1, and the methods for estimating WP-GEB are explained in Subsection 2.2.

[1] neural networks trained for classifications
[2] The version 3 has already been available [1] and the version 4 will be published soon.

2.1 Definition of WP-GEB

The *randomly weight-perturbed generalization error* $\mathbf{R}^{\alpha}(f_w)$ of the classifier f_w with the weight-parameters w is the expected value of the weight-perturbed individual errors $\mathbf{r}^{\alpha}_{(x,y)}(f_w)$ for any pair $(x, y) \sim \mathcal{D}$, where $\mathcal{D}$ is the distribution of pairs of input x and output (correct class) y, and it is defined as follows:

$$\mathbf{R}^{\alpha}(f_w) := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\mathbf{r}^{\alpha}_{(x,y)}(f_w) \right]$$

where, the *weight-perturbed individual error* $\mathbf{r}^{\alpha}_{(x,y)}(f_w)$ is the expected value of the misclassification-rate for one pair (x, y) of data when adding any perturbation $u \sim \mathcal{U}_{w,\alpha}$ on weights w, and the distribution of weight-perturbations $\mathcal{U}_{w,\alpha}$ is a multi-dimension uniform-distribution for randomly selecting ones from the set $U_{w,\alpha}$ of weight-perturbations.

$$\mathbf{r}^{\alpha}_{(x,y)}(f_w) := \mathop{\mathbb{E}}_{u\sim\mathcal{U}_{w,\alpha}} [\ell(f_{w+u}(x), y)]$$
$$U_{w,\alpha} := \{(u_1, \ldots, u_{|w|}) \mid \forall i.\ |u_i| \leq \alpha|w_i|\}$$

where $\ell(y, y')$ is the 0-1 loss-function that returns 0 if $y = y'$ and otherwise 1 (i.e. $\ell(y, y') := \mathbb{1}[y \neq y']$). The set $U_{w,\alpha}$ of weight-perturbations means that the absolute value of perturbation u_i added on each weight w_i is less than or equal to $\alpha|w_i|$, where the α is the ratio of the perturbation to the weight.

The *worst weight-perturbed generalization error* $\mathbf{W}^{\alpha}_{\theta}(f_w)$ is the expected value of 0/1 boolean value for any pair $(x, y) \sim \mathcal{D}$, such that the boolean value is 1 if the weight-perturbed individual error $\mathbf{r}^{\alpha}_{(x,y)}(f_w)$ is greater than the threshold $\theta_{(x,y)}$, and it is defined as follows:

$$\mathbf{W}^{\alpha}_{\theta}(f_w) := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\mathbb{1} \left[\mathbf{r}^{\alpha}_{(x,y)}(f_w) > \theta_{(x,y)} \right] \right]$$
$$\Theta := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\theta_{(x,y)} \right]$$

where $\theta_{(x,y)}$ is a threshold that can be defined for each (x, y), and Θ is the expected value. The threshold is the accepted ratio of existence of adversarial perturbations. Although the threshold is ideally 0 ($\Theta = 0$) for rigid evaluation, it is practically reasonable to set the threshold to an appropriate small value from the perspective of computation cost and so on. Reasonable definitions of thresholds (fixed threshold and adaptive threshold) are explained in the next Section 2.2.

In general, it is difficult to exactly compute weight-perturbed generalization errors because there can infinitely exist data (x, y) and perturbations u. Therefore, WP-GEB-Estimator estimates the upper-bound B of the weight-perturbed generalization error GE such that the following inequality holds with probability at least $(1 - \delta)$ for any $\delta \in (0, 1)$.

$$\mathbb{P}[GE \leq B] \geq 1 - \delta$$

2.2 Estimation of WP-GEB

It has been well studied to estimate the upper bound of randomly weight-perturbed generalization error $\mathbf{R}^{\alpha}(f_w)$. For example, Pérez-Ortiz et al. [4] presented that such upper bound can be estimated by the combination of the theorem of Maure Bound [3] and the theorem of Sample Convergence Bound [2], from the inference-results of the classifier f_w for a finite dataset and a finite perturbation-sample. WP-GEB-Estimator estimates the upper bounds based on the method presented by Pérez-Ortiz except that test datasets are used in WP-GEB-Estimator instead of training datasets.

On the other hand, it has not been well studied to estimate the worst weight-perturbed generalization error $\mathbf{W}^{\alpha}_{\theta}(f_w)$. It depends on the threshold θ, and there are two practical and reasonable approaches for deciding the threshold [5]. One of them is *fixed threshold* $\theta^{fix}(m, n, \delta_0)$ that is independent from data (x, y) and is defined as follows[3]:

$$\theta^{fix}(m, n, \delta_0) := 1 - \left(\frac{\delta_0}{2n} \right)^{1/m}$$

[3]It can be statistically guaranteed with a probability at least $(1 - \delta_0/n)$ that the upper bound of the expected value of $\mathbf{r}^{\alpha}_{(x,y)}(f_w)$ is less than the fixed threshold $\theta^{fix}(m, n, \delta_0)$ when adversarial perturbations cannot be found by the random weight-perturbation sample for each $(x, y) \in T$.

where $\delta_0 \in (0, \delta)$ is the acceptable degradation of confidence caused by using the perturbation sample whose size is m instead of any perturbation, and n is the size of the test dataset T. In general, it is reasonable to select δ_0 to about a half of δ (i.e. $\delta_0 = 0.5 \times \delta$). The other one is *adaptive threshold* $\theta^{adapt}_{(x,y)}(m, n, \delta_0)$ and is defined as follows:

$$\theta^{adapt}_{(x,y)}(m, n_0, \delta_0) := \begin{cases} 0 & \text{if } (x, y) \in T_1 \\ \theta^{fix}(m, n_0, \delta_0) & \text{otherwise} \end{cases}$$

where T_1 is a subset of the test dataset T and is the set of the pairs (x, y) of data such that one or more adversarial weight-perturbations are found for each (x, y) by adversarial perturbation-search or a random perturbation-sample, and n_0 is the size of $(T - T_1)$. WP-GEB-Estimator estimates the upper bounds of the worst weight-perturbed generalization errors with fixed threshold $\mathbf{W}^{\alpha}_{\theta^{fix}}(f_w)$ and the upper bounds of the worst weight-perturbed generalization errors with adaptive threshold $\mathbf{W}^{\alpha}_{\theta^{adapt}}(f_w)$ in accordance with methods presented in [5].

3 Introduction to WP-GEB-Estimator

In this section, the four tools that constitute WP-GEB-Estimator are introduced in Subsection 3.1, and then input files that WP-GEB-Estimator takes and output files that WP-GEB-Estimator generates are explained in Subsections 3.2 and 3.3, respectively.

3.1 Tool Structure

WP-GEB-Estimator consists of the four tools for training, measuring, searching, and estimating. These tools are described in the Python language with the TensorFlow/Keras libraries.

- `train`: trains classifiers to demonstrate WP-GEB-Estimator.
  - Input: Train-dataset, Network architecture (CSV format), etc
  - Output: Trained classifier (TensorFlow-SavedModel format), etc
- `measure`: measures misclassification-errors of trained classifiers for random weight-perturbations.
  - Input: Test-dataset, Trained classifier (SavedModel format), Perturbation-ratios, etc
  - Output: Measurement result (CSV format) of errors for random weight-perturbations, etc
- `search`: searches for adversarial weight-perturbations in trained classifiers.
  - Input: Test-dataset, Trained classifier, Measurement results (CSV format) of errors, etc
  - Output: Adversarial search results (CSV format) including measurement results
- `estimate`: estimates weight-perturbed generalization error bounds.
  - Input: Results of search and random perturbation sample (CSV format), etc
  - Output: Estimate results of weight-perturbed generalization error (CSV format), etc

The weight-perturbed generalization error bounds of classifiers trained by the first tool `train` or the other tools can be estimated by sequentially executing the rest three tools: `measure` → `search` → `estimate`. The relation between input/output files of the four tools that constitute WP-GEB-Estimator is shown in Figure 2. In the case of evaluating classifiers already trained, it is not necessary to execute the train tool `train`. However, the search tool `search` must be executed before the estimate tool `estimate`, even though it is not necessary to search for adversarial weight-perturbations (i.e. worst weight-perturbed errors with adaptive thresholds) because the tool `estimate` must take the file `search_out.csv` generated by the tool `search`. In this case, the tool `search` with the option `--skip_search` can quickly generate the files by skipping the real search. The input and output files are explained in the next Subsections 3.2 and 3.3.

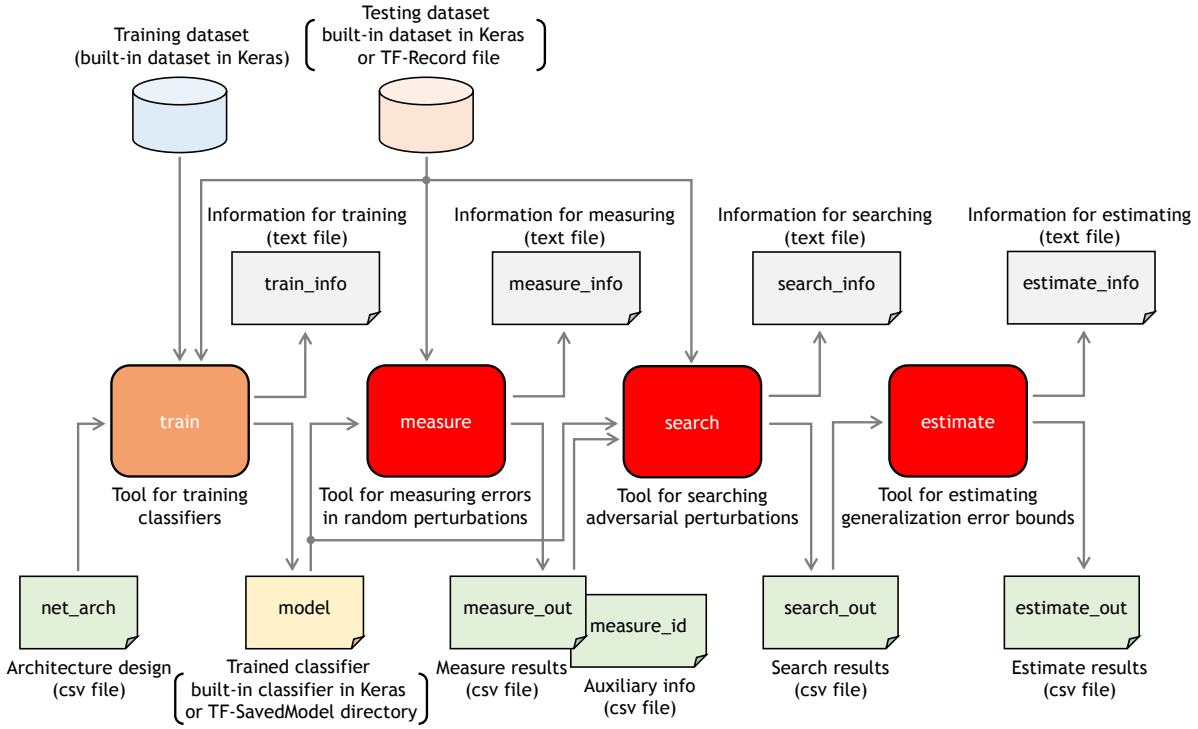

Figure 2: The relation between input/output files of the four tools that constitute WP-GEB-Estimator

3.2 Input files

The trained classifier to be evaluated and the test dataset are specified by the options `--model_dir` and `--dataset_file` in the measure tool `measure`. The format of classifiers currently supported is only TensorFlow-SavedModel and the format of datasets currently supported is only TF-Record. The built-in classifiers (e.g. Inception v3) and datasets (e.g. MNIST) that are available in Keras can be used by specifying the names in the options explained in Section 4.

For training a classifier, the tool `train` takes a CSV file that describes the architecture of the classifier. The types and parameters that can be specified in the columns of the CSV file are shown in Table 1, where $\mathbb{I}$ and $\mathbb{R}$ are the set of integers more than 0 and the set of real numbers more than 0, respectively. As shown in Table 1, some parameters must be specified in accordance with layer-types, while the $L2$-regularization coefficient (`regular_l2`) and the dropout-rate (`rate`) can be omitted because they can be later specified by options explained in Section 4. By sequentially describing layers from the input layer in lines of the CSV file, it can represent simple multilayer perceptrons (MLP), convolutional neural networks (CNN), etc. Several samples of CSV files of MLP and CNN are saved in the directory `net_arch`.

Table 1: The types and the parameters of layers ($a \in \{$`relu`$|$`linear`$|$`softmax`$\}$, $n, n_i \in \mathbb{I}$, $r \in \mathbb{R}$)

`type`	`activation`	`units`	`filters`	`int_tuple`	`regular_l2`	`rate`
`Activation`	a					
`Dense`	a	n			r	
`Conv2D`	a		n	(n_1, n_2)		
`MaxPooling2D`				(n_1, n_2)		
`Dropout`						r
`BatchNormalization`						
`Flatten`						

3.3 Output files

The input/output information used in each tool `train`, `measure`, `search`, and `estimate` is saved in text-files `train_info.txt`, `measure_info.txt`, `search_info.txt`, and `estimate_info.txt` in the directory `result`, respectively (the file names can be changed by options). In addition, the input/output data are also saved in the CSV file `estimate_out.csv`, that includes intermediate data saved in `measure_out.csv` and `search_out.csv`. The estimate-result for each weight-perturbation ratio is incrementally added to one line in the CSV file `estimate_out.csv` after the execution of the tool `estimate`. The CSV file is useful for analyzing the results and drawing their graphs. The each column (A $\sim$ AL) of the CSV file `estimate_out.csv` is explained as follow:

- Input/Output data in `measure` (also saved in A $\sim$ Q of `measure_out.csv`)

  A [`rnd_seed_measure`] Random seed

  B [`dataset_name`] The name of test dataset

  C [`dataset_size`] The size of test dataset

  D [`dataset_offset`] The start index of test dataset

  E [`dataset_file`] The file name of test dataset

  F [`dataset_fmt`] The file format of test dataset

  G [`image_width`] The width of images in test dataset

  H [`image_height`] The height of images in test dataset

  I [`batch_size_measure`] The number of data for batch inference

  J [`model_dir`] The name or the directory name of classifier

  K [`perturb_bn`] The perturbation on batch normalization parameters (`0`: not added, `1`: added)

  L [`perturb_params_size`] The number of perturbed weight-parameters

  M [`perturb_ratio`] The ratio of the maximum weight-perturbation to weight

  N [`perturb_sample_size`] The size of the random sample of weight-perturbations

  O [`err_num_random`] The number of data, for which adversarial weight-perturbations are found by random sample

  P [`test_err_wst`] The ratio of data, for which adversarial weight-perturbations are found by random sample

  Q [`test_err_avr`] The average of test error for random weight-perturbation sample

- Input/Output data in `search` (also save in R $\sim$ W in `search_out.csv`)

  R [`rnd_seed_search`] Random seed

  S [`batch_size_search`] The number of data for gradient computation in parallel

  T [`search_mode`] The search-mode for adversarial weight-perturbations (`0`: FGSM, `1`: I-FGSM)

  U [`max_iteration`] The maximum iteration in I-FGSM

  V [`err_num_search`] The number of data, for which adversarial weight-perturbations are found by search

  W [`err_num`] The number of data, for which adversarial weight-perturbations are found by search or random sample

- Input/Output data in `estimate`

  – Worst weight-perturbation (adaptive threshold)

  X [`gen_err_wst_adapt_ub`] Perturbed generalization error upper bound

  Y [`test_err_wst_adapt_ub`] Perturbed test error upper bound

  Z [`err_thr_adapt_ub`] Adaptive threshold (expected upper bound)

  AA [`err_thr_adapt`] Adaptive threshold (average for test dataset)

AB [conf_wst_adapt] Confidence of perturbed generalization error upper bound

AC [conf0_wst_adapt] Confidence of perturbed test error upper bound

– Worst weight-perturbation (fixed threshold)

AD [gen_err_wst_fix_ub] Perturbed generalization error upper bound

AE [test_err_wst_fix_ub] Perturbed test error upper bound

AF [err_thr_fix] Fixed threshold

AG [conf_wst_fix] Confidence of perturbed generalization error upper bound

AH [conf0_wst_fix] Confidence of perturbed test error upper bound

– Random weight-perturbation

AI [gen_err_rnd_ub] Perturbed generalization error upper bound

AJ [test_err_rnd_ub] Perturbed test error upper bound

AK [conf_rnd] Confidence of perturbed generalization error upper bound

AL [conf0_rnd] Confidence of perturbed test error upper bound

4 Execution of WP-GEB-Estimator

In this section, it is explained how to execute each tool in WP-GEB-Estimator with options. Hereafter, the set $\mathbb{I}$ of integers more than 0, the set $\mathbb{R}$ of real numbers more than 0, and the set $\mathbb{S}$ of strings are used. In addition, the following set $\mathbb{SR}^*$ of strings that represent lists of real numbers, where the delimiter is the space `" "` is used.

$$\mathbb{SR}^* = \{\texttt{"}r_1 \;\; r_2 \;\; \cdots \;\; r_m\texttt{"} \mid \exists m \in \mathbb{I}.\ \forall i \in \{1, \ldots, m\}.\ r_i \in \mathbb{R}\} \subset \mathbb{S}$$

Furthermore, the following sets are also used for expressing the bulit-in datasets and classifiers that are available in TensorFlow/Keras.

$$\begin{aligned}\texttt{KerasDataSets} &= \{\texttt{"mnist"}, \texttt{"fashion\_mnist"}, \texttt{"cifar10"}\} \subset \mathbb{S}\\ \texttt{KerasModels} &= \{\texttt{"inception\_v3"}, \texttt{"inception\_resnet\_v2"}, \texttt{"resnet50"}, \texttt{"xception"},\\ &\quad \texttt{"densenet121"}, \texttt{"densenet169"}, \texttt{"densenet201"}, \texttt{"vgg16"}, \texttt{"vgg19"},\\ &\quad \texttt{"nasnetlarge"}, \texttt{"nasnetmobile"}\} \subset \mathbb{S}\end{aligned}$$

4.1 Training classifiers

The train tool `train` can be executed by the following command in the directory `src`:

```
python train_main.py [options]
```

where the following options are available.

`--random_seed` n $(n \in \mathbb{I}$, default $n =$ `1`$)$

sets the random seed to n. (if $n =$ `0` then random seed is not used, i.e. it is non-deterministic.)

`--net_arch_file` s $(s \in \mathbb{S}$, default $s =$ `"net_arch/cnn_s"`$)$

specifies the CSV-file name of the network architecture to s.

`--result_dir` s $(s \in \mathbb{S}$, default $s =$ `"result"`$)$

specifies the directory for saving the training results to s.

`--model_dir` s $(s \in \mathbb{S}$, default $s =$ `"model"`$)$

specifies the directory (in the SavedModel format) for saving the trained classifier to s.

`--dataset_name` s $(s \in$ `KerasDataSets`, default $s =$ `"mnist"`$)$

specifies the dataset name for training/testing to s.

`--train_dataset_size` n ($n \in \mathbb{I}$, default $n =$ `50000`)

sets the size of training dataset to n.

`--train_dataset_offset` n ($n \in \mathbb{I}$, default $n =$ `0`)

sets the start index of training dataset to n.

`--test_dataset_size` n ($n \in \mathbb{I}$, default $n =$ `5000`)

sets the size of test dataset to n.

`--test_dataset_offset` n ($n \in \mathbb{I}$, default $n =$ `0`)

sets the start index of test dataset to n.

`--validation_ratio` r ($r \in [0, 1) \subset \mathbb{R}$, default $r =$ `0.1`)

sets the ratio of the validation dataset in training dataset to r.

`--sigma` r ($r \in \mathbb{R}$, default $r =$ `0.1`)

sets the standard deviation of normal distribution of initial parameters to r

`--batch_size` n ($n \in \mathbb{I}$, default $n =$ `100`)

sets the batch size to n for training.

`--epochs` n ($n \in \mathbb{I}$, default$n =$ `50`)

sets the number of epochs to n for training.

`--dropout_rate` r ($r \in [0, 1) \subset \mathbb{R}$, default $r =$ `0.0`)

sets the dropout rate to r if it is not specified in the architecture file.

`--regular_l2` r ($r \in \mathbb{R}$, default $r =$ `0.0`)

sets the $L2$-regularization coefficient to r if it is not specified in the architecture file.

`--learning_rate` r ($r \in \mathbb{R}$, default $r =$ `0.01`)

sets the (initial) learning rate to r.

`--decay_rate` r ($r \in \mathbb{R}$, default $r =$ `1.0`)

sets the decay rate of learning rate to r. (if $r =$ `1.0` then there is no decay)

`--decay_steps` n ($n \in \mathbb{I}$, default $n =$ `0`)

sets the decay step of learning rate to n. (if $n =$ `0` then there is no decay)

`--early_stop` b ($b \in \{$`0`, `1`$\}$, default $b =$ `0`)

makes early-stopping valid if $b =$ `1` otherwise early-stopping is invalid.

`--early_stop_delta` r ($r \in \mathbb{R}$, default $r =$ `0.0`)

defines *not-decreasing* as the situation that the amount of the decrease is less than r.

`--early_stop_patience` n ($n \in \mathbb{I}$, default $n =$ `3`)

stops training when loss is *not-decreasing* continuously n steps if early stopping is valid.

`--verbose` b ($b \in \{$`0`, `1`, `2`$\}$, default $b =$ `1`)

displays the progress of training if $b =$ `1` or `2`.

4.2 Measuring errors for random perturbations

The measure tool `measure` can be executed by the following command in the directory `src`:

```
python measure_main.py [options]
```

where the following options are available.

`--random_seed` n ($n \in \mathbb{I}$, default $n =$ `1`)

sets the random seed to n. (if $n =$ `0` then random seed is not used, i.e. it is non-deterministic.)

`--result_dir` s ($s \in \mathbb{S}$, default $s =$ `"result"`)

specifies the directory for loading or saving results to s.

`--measure_file` s ($s \in \mathbb{S}$, default $s =$ `"measure"`)

specifies the file name for saving the measurement results to s.

`--model_dir` s ($s \in \mathbb{S}$, default $s =$ `"model"`)

specifies the directory (in the SavedModel format) for loading the trained classifier to s.
(if $s \in$ `KerasModels` then the built-in classifier s pre-trained in Keras is loaded.)

`--dataset_name` s ($s \in \mathbb{S}$, default $s =$ `"mnist"`)

specifies the test dataset name to s.
(if $s \in$ `KerasDataSets` then the built-in dataset s in Keras is loaded.)

`--dataset_file` s ($s \in \mathbb{S}$, default $s =$ `"~/imagenet/1k-tfrecords/validation-*-of-00128"`)

specifies the file name for loading test dataset to s (not necessary if $s \in$ `KerasDataSets`)

`--dataset_fmt` s ($s \in \mathbb{S}$, default $s =$ `"tfrecord"`)

specifies the file format of test dataset to s.
(Currently, the supported format is TF-Record (`"tfrecord"`) only.)

`--image_width` n ($n \in \mathbb{I}$, default $n =$ `0`)

specifies the width of images in test dataset to n ($n =$ `0` if Keras provides the width)

`--image_height` n ($n \in \mathbb{I}$, default $n =$ `0`)

specifies the height of images in test dataset to n ($n =$ `0` if Keras provides the height)

`--dataset_size` n ($n \in \mathbb{I}$, default $n =$ `5000`)

sets the size of test dataset to n.

`--dataset_offset` n ($n \in \mathbb{I}$, default $n =$ `0`)

sets the start index of test dataset to n.

`--batch_size` n ($n \in \mathbb{I}$, default $n =$ `0`)

sets the dataset size measured at one time to n. (if $n =$ `0` then it is the dataset size.)

`--perturb_ratios` s ($s \in \mathbb{SR}^*$, default $s =$ `"0.01 0.1 1"`)

sets the list of the ratios of maximum weight-perturbation to weight to s.
(The weight-perturbed errors are sequentially estimated from the top of the list.)

`--perturb_bn` b ($b \in \{$`0`, `1`$\}$, default $b =$ `0`)

perturbs parameters in batch normalization layers if $b = 1$. (usually $b = 0$)

(if $b = 0$ then scale/shift parameters in batch normalization layers are not perturbed.)

`--perturb_sample_size` n ($n \in \mathbb{I}$, default $n = 1215$)

sets the size of random perturbation sample to n.

`--verbose_measure` b ($b \in \{0, 1\}$, default $b = 1$)

displays the progress of measuring if $b = 1$. (if $b = 0$ then it is not displayed.)

4.3 Searching for adversarial perturbations

The search tool `search` can be executed by the following command in the directory `src`:

```
python search_main.py [options]
```

where the following options are available.

`--random_seed` n ($n \in \mathbb{I}$, default $n = 1$)

sets the random seed to n. (if $n = 0$ then random seed is not used, i.e. it is non-deterministic.)

`--result_dir` s ($s \in \mathbb{S}$, default $s =$ `"result"`)

specifies the directory for loading or saving results to s.

`--measure_file` s ($s \in \mathbb{S}$, default $s =$ `"measure"`)

specifies the file name for loading the measurement results to s.

`--search_file` s ($s \in \mathbb{S}$, default $s =$ `"search"`)

specifies the file name for saving the search results (including the measurement results) to s.

`--skip_search` b ($b \in \{0, 1\}$, default $b = 0$)

skips searching for adversarial weight-perturbations if $b = 1$. (usually $b = 0$)

(The search can be skipped if worst weight-perturbed errors (adaptive thr.) are not necessary.)

`--search_mode` n ($n \in \{0, 1\}$, default $n = 0$)

sets the search-mode for adversarial weight-perturbations to *mode-n*.

- mode-0: FGSM adds weight-perturbations towards misclassification based on gradients.
- mode-1: I-FGSM iteratively applies FGSM until reaching a local loss-maximum.

`--batch_size` n ($n \in \mathbb{I}$, default $n = 10$)

sets the number of data for gradient computation in parallel to n if FGSM (mode-0) is used.

`--max_iteration` n ($n \in \mathbb{I}$, default $n = 20$)

sets the maximum iteration to n if I-FGSM (mode-1) is used.

`--verbose_search` b ($b \in \{0, 1\}$, default $b = 1$)

displays the progress of searching if $b = 1$.

4.4 Estimating WP-GEB

The estimate tool can be executed by the following command in the directory `src`:

```
python estimate_main.py [options]
```

where the following options are available.

`--result_dir` s ($s \in \mathbb{S}$, default $s =$ `"result"`)

specifies the directory for loading or saving results to s.

`--search_file` s ($s \in \mathbb{S}$, default $s =$ `"search"`)

specifies the file name for loading the search and the measurement results to s.

`--estimate_file` s ($s \in \mathbb{S}$, default $s =$ `"measure"`)

specifies the file name for saving the estimate results to s.

`--delta` δ ($\delta \in (0,1) \subset \mathbb{R}$, default $\delta =$ `0.1`)

sets the confidence of generalization error bounds to $1-\delta$. (It is explained in Subsection 2.2.)
(The confidence-degradation is caused by using a dataset instead of any data.)

`--delta0_ratio` r_0 ($r \in (0,1) \subset \mathbb{R}$, default $r_0 =$ `0.5`)

sets the confidence of perturbed test-error bounds to $1-\delta r_0$ (The δr_0 is δ_0 in Subsection 2.2.)
(The confidence-degradation is caused by using a perturbation set instead of any perturbation.)

`--max_nm` n ($n \in \mathbb{I}$, default $r =$ `10`)

sets the maximum number of iterations in Newton Method used in estimating bounds to n.

`--eps_nm` r ($r \in \mathbb{R}$, default $r =$ `0.0001`)

sets the acceptable maximum error in Newton Method to r.

5 Execution Examples of WP-GEB-Estimator

In this section, examples of execution commands for WP-GEB-Estimator and some execution results are introduced. WP-GEB-Estimator is described in the Python language with the libraries TensorFlow and NumPy. The software versions used in the development are Python 3.10.10, TensorFlow 2.13.0, and NumPy 1.23.2.

5.1 Training a classifier and estimating WP-GEB

Figure 3 shows the execution script `run.sh` (saved in the directory `examples`) for training a multilayer perceptron by the dataset MNIST, that includes images of hand-written digits, measuring the errors when random weight-perturbations are added, searching for adversarial weight-perturbations, and then estimating the generalization error bounds. The execution result is saved in the directory `results/result`. The execution time in MacBook Pro (CPU: Apple M2 Max, Mem: 96GB) was as follows: For each perturbation ratio (dataset size: 5000, weight-perturbation sample size: 1215, corresponding to the fixed threshold 1%), about 100 seconds for training, about 5 minutes for measuring, about 1 minute for searching, and less than 0.1 seconds for estimating.

Figure 4 is a part of the result `estimate_info.txt` for the execution script `run.sh`. It shows the estimate results of the worst weight-perturbed generalization bounds, test error bounds, and the thresholds. For example, the worst weight-perturbed generalization is less than 25% with probability at least 90%, for the adaptive threshold bound 0.75% for the ratio 0.01 of the weight-perturbation to the weight.

```
python train_main.py  --net_arch_file "net_arch/mlp_s_bn"
python measure_main.py
python search_main.py
python estimate_main.py
```

Figure 3: A simple execution script `run.sh`

```
Perturbation ratio = 0.01
  Random perturbation sample size: 1215
  Worst weight-perturbation (adaptive threshold):
    Perturbed generalization error bound: 25.01% (Conf: 90.00%)
    Perturbed Test error bound: 23.36% (Conf: 95.00%)
    Adaptive threshold bound (expected): 0.7648% (Conf: 90.00%)
    Adaptive threshold (average): 0.7495%
  Worst weight-perturbation (fixed threshold):
    Perturbed generalization error bound: 4.32% (Conf: 90.00%)
    Perturbed Test error bound: 3.56% (Conf: 95.00%)
    Fixed threshold: 0.9996%
  Random weight-perturbation:
    Perturbed generalization error bound: 6.23% (Conf: 90.00%)
    Perturbed Test error bound: 4.91% (Conf: 95.00%)
  (Elapsed Time: 0.0 [sec])

Perturbation ratio = 0.1
  Random perturbation sample size: 1215
  Worst weight-perturbation (adaptive threshold):
    Perturbed generalization error bound: 100.00% (Conf: 90.00%)
    Perturbed Test error bound: 99.90% (Conf: 95.00%)
    Adaptive threshold bound (expected): 0.0013% (Conf: 90.00%)
    Adaptive threshold (average): 0.0004%
  Worst weight-perturbation (fixed threshold):
    Perturbed generalization error bound: 7.34% (Conf: 90.00%)
    Perturbed Test error bound: 6.36% (Conf: 95.00%)
    Fixed threshold: 0.9996%
  Random weight-perturbation:
    Perturbed generalization error bound: 6.29% (Conf: 90.00%)
    Perturbed Test error bound: 4.96% (Conf: 95.00%)
  (Elapsed Time: 0.0 [sec])
```

Figure 4: A part of the execution result `estimate_info.txt` for the script `run.sh`

5.2 Estimating WP-GEB for the pre-trained classifier Inception-v3

Figure 5 is an execution script `run_imagenet.sh` for estimating generalization error bounds of the pre-trained bulit-in classifier Inception-v3 provided in TensorFlow/Keras. The execution result is saved in the directory `results/result_imagenet`. Here, it is assumed that the file (in the TF-Record format) of the dataset ImageNet is saved in the directory `ImageNet_DIR`. The ImageNet file (`1k-tfrecords-0.zip`, `1k-tfrecords-1.zip`) in the TF-Record format can be downloaded from the following web-site of Kaggle.

```
ImageNet 1K TFrecords ILSVRC2012 - part 0
https://www.kaggle.com/datasets/hmendonca/imagenet-1k-tfrecords-ilsvrc2012-part-0
```

The execution time in MacBook Pro (CPU: Apple M2 Max, Mem: 96GB) was as follows: for the dataset size 1000 and weight-perturbation sample size 525, corresponding to the fixed threshold 2%, about 30 minutes for measuring, about 7 minute for searching, and less than 0.1 seconds for estimating. The

```
ImageNet_DIR="$HOME/datasets/imagenet/1k-tfrecords/validation"
python measure_main.py \
    --result_dir "result_imagenet" --dataset_name "imagenet" \
    --dataset_file "$ImageNet_DIR/validation-*-of-00128" \
    --dataset_size 1000 --batch_size 50 \
    --model_dir "inception_v3" \
    --perturb_ratios "0.0001" --perturb_sample_size 525
python search_main.py --result_dir "result_imagenet"
python estimate_main.py --result_dir "result_imagenet"
```

Figure 5: An execution script `run_imagenet.sh` for the pre-trained classifier Inception-v3

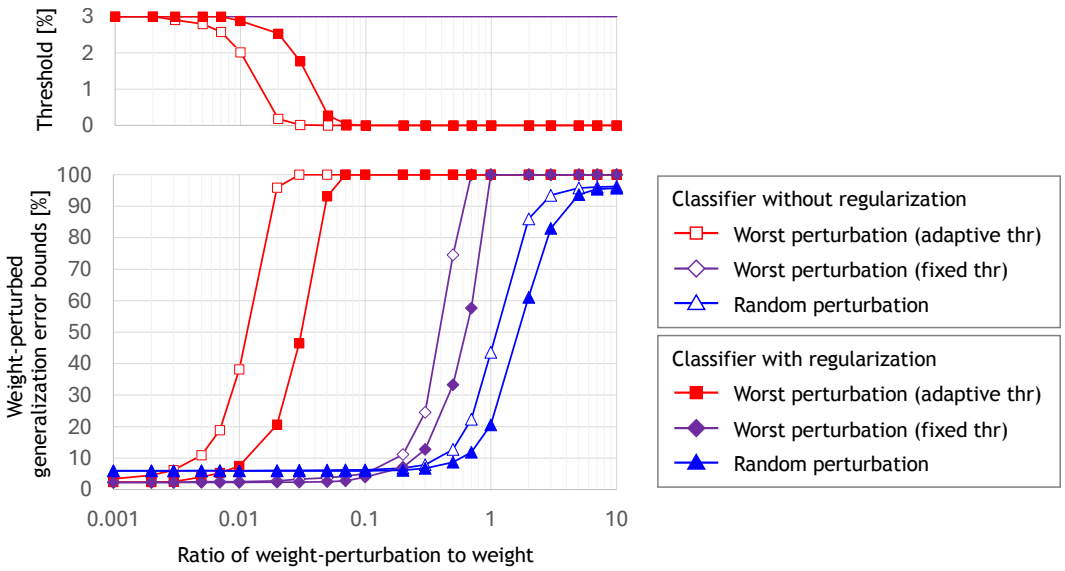

Figure 6: The estimate results of weight-perturbed generalization errors and the thresholds (conf. 90%)

estimate result was as follows: for the ratio 0.0001 of the weight-perturbation to the weight, the worst weight-perturbed generalization is less than 36.1% with probability at least 90%, for the adaptive threshold bound 1.4% and the randomly weight-perturbed generalization is less than 33.2% with probability at least 90%.

5.3 Comparison of WP-GEB between two classifiers

The script `run_main.sh` (saved in the directory `examples`) is an execution script for training two classifiers with or without regularization and for estimating the weight-perturbed generalization error bounds. The $L2$-regularization coefficient can be easily specified by the execution option `--regular_l2` if the $L2$-regularization coefficients in the architecture file are omitted. About the options, also see the scripts `run_submit.sh` and `run_option.sh`, that are executed by `run_main.sh`.

Figure 6 shows the graph of the estimate results by WP-GEB-Estimator and they are the weight perturbed generalization error bounds (confidence 90%) and the thresholds of the classifier without regularization and the classifier with the $L2$-regularization coefficient 0.001. The graph was drawn by the spreadsheet-software Microsoft Excel from the `estimate_out.csv`. The horizontal axis represents the ratio of weight-perturbation to weight. Figure 6 explicitly shows that the classifier with regularization is more robust for weight-perturbations than the classifier without regularization, even though there is no difference between their errors when no perturbation is added. In addition, the worst weight-perturbed generalization error bounds with adaptive thresholds start increasing at the ratio that is two orders less

than the randomly weight-perturbed generalization error bounds start increasing.

References

[1] National Institute of Advanced Industrial Science and Technology (AIST), Machine Learning Quality Management Guideline (3rd English Edition), Digital Architecture Research Center, Cyber Physical Security Research Center, Artificial Intelligence Research Center, Technical Report DigiARC-TR-2023-01/ CPSEC-TR-2023001, 2023. https://www.digiarc.aist.go.jp/publication/aiqm/

[2] J. Langford and R. Caruana, (Not) Bounding the True Error, 14th International Conference on Neural Information Processing Systems (NIPS 2001), pp.809–816, 2001.

[3] A. Maurer, A Note on the PAC Bayesian Theorem, arXiv:cs/0411099, 2004.

[4] M. Pérez-Ortiz, O. Rivasplata, J. Shawe-Taylor, and C. Szepesvári, Tighter risk certificates for neural networks, Journal of Machine Learning Research (JMLR), 2021.

[5] Y. Isobe, Estimating Generalization Error Bounds for Worst Weight-Perturbed Neural Classifiers, The 38th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI), May 2024 (to be presented in Japanese)

Acknowledgment

This tool WP-GEB-Estimator has been developed in the project JPNP20006 commissioned by the New Energy and Industrial Technology Development Organization (NEDO).