WP-GEB-Estimator-2 User Manual

∼ WP-GEB : Weight-Perturbed GEneralization Bounds ∼

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

17th March, 2025

Contents

1 Introduction

The "WP-GEB" in the tool name WP-GEB-Estimator-2 stands for *weight-perturbed generalization bounds* and WP-GEB-Estimator-2 is a tool for estimating the upper bounds of *weight-perturbed generalization errors* and the upper bounds of *weight-perturbed generalization risks* of *neural classifiers*[1] (hereafter called just *classifiers*). Weight-perturbations represent perturbation added to weight-parameters of classifiers, and weight-perturbed generalization errors are the expected values of the misclassification-rates of classifiers for any input and any perturbation, where the input and the perturbation are sampled according to distributions. On the other hand, weight-perturbed generalization risks are the expected values of the existence-rates (i.e., *adversarial weight-perturbation*[2] existence rates) of *risky data*, for which the misclassification-rates due to perturbations exceeds the acceptable threshold. Weight-perturbed generalization errors are effective for evaluating robustness against random perturbations (e.g., natural noise), while weight-perturbed generalization risks are effective for evaluating robustness against worst-case perturbations (e.g., adversarial attacks).

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[3]. WP-GEB-Estimator-2 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 estimate, with probabilistic confidence, an upper bound on the weight-perturbed generalization risk for any input.
- `[Generalization error]` It can statistically estimate, with probabilistic confidence, an upper bound on the weight-perturbed generalization error for any input.

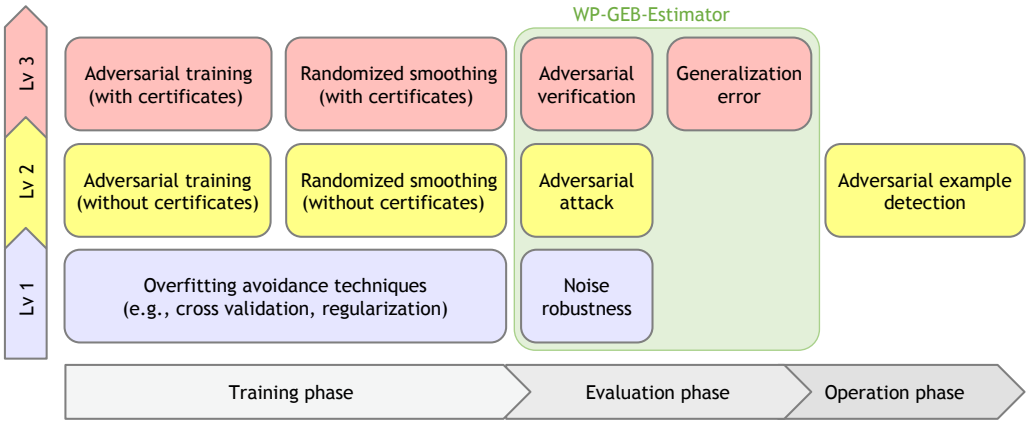

Figure 1: Techniques to evaluate and improve stability (Fig.14 in MLQM-Guideline [1])

[1] Neural classifiers are neural networks trained for classifications

[2] Adversarial weight-perturbation refers to a perturbation that causes misclassification.

[3] The version 3 has already been available [1] and the version 4 will be published soon.

2 Introduction to WP-GEB

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

2.1 Definition of WP-GEB

The *weight-perturbed generalization error* $\mathbf{E}^{\alpha}(f_w)$ of the classifier f_w with the weight-parameters w is the expected value of the weight-perturbed individual errors $\mathbf{e}^{\alpha}_{(x,y)}(f_w)$ for any data-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:

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

where, the *weight-perturbed individual error* $\mathbf{e}^{\alpha}_{(x,y)}(f_w)$ is the expected value of the misclassification-rate (i.e., the adversarial weight-perturbation existence 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.

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

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 *weight-perturbed generalization risk* $\mathbf{R}^{\alpha}_{\theta}(f_w)$ 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 $\mathbf{e}^{\alpha}_{(x,y)}(f_w)$ is greater than the threshold $\theta_{(x,y)}$, and it is defined as follows:

$$\begin{aligned}\mathbf{R}^{\alpha}_{\theta}(f_w) &:= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\mathbb{1}\left[\mathbf{e}^{\alpha}_{(x,y)}(f_w) > \theta_{(x,y)}\right]\right] \\ \Theta &:= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\theta_{(x,y)}\right]\end{aligned}$$

where $\theta_{(x,y)}$ is the *acceptable threshold* set for each (x, y), and Θ is the expected value, called the *generalization acceptable threshold*. The threshold $\theta_{(x,y)}$ represents the accepted existence rate of adversarial weight perturbations, and the input x is called *risky* if the the weight-perturbed individual error exceeds the acceptable threshold (i.e., $\mathbf{e}^{\alpha}_{(x,y)}(f_w) > \theta_{(x,y)}$). Although the acceptable threshold is zero ($\Theta = 0$) for rigid evaluation, it is practically reasonable to set the threshold to an appropriate small value considering computation cost and so on. Reasonable acceptable thresholds are explained in the next Section 2.2.

In general, it is difficult to exactly compute weight-perturbed generalization risks/errors because there can infinitely exist data (x, y) and perturbations u. Therefore, WP-GEB-Estimator-2 estimates the upper-bound B of the weight-perturbed generalization risk/error G such that the following inequality holds with probability at least $(1 - \delta)$ confidence for any $\delta \in (0, 1)$.

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

2.2 Estimation of WP-GEB

It has been well studied to estimate the upper bound of the weight-perturbed generalization error $\mathbf{E}^{\alpha}(f_w)$. 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-2 estimates the upper bounds based on the method presented by Pérez-Ortiz except that test datasets are used in WP-GEB-Estimator-2 instead of training datasets.

On the other hand, there has been a few researches on estimating the upper bound of weight-perturbed generalization risk $\mathbf{R}^{\alpha}_{\theta}(f_w)$. Testing with randomly sampled weight perturbations can probabilistically

guarantee an upper bound on the existence rate of adversarial weight perturbations. However, since the existence rate is often extremely small, it difficult to detect adversarial perturbations using random perturbation samples. Although there are search methods that can easily detect such adversarial weight perturbations, they cannot guarantee an upper bound on their existence rate. Therefore, WP-GEB-Estimator-2 combines search methods for adversarial weight perturbation and testing with randomly sampled perturbations as follows:

1. First, a gradient-based method (FGSM, I-FGSM) is applied to search for adversarial perturbations for a data point (x, y). Then, adversarial perturbations detected through the search are classified as risky with the acceptable threshold of zero.

2. Next, for each data point (x, y) that has not been detected through the search, testing with randomly sampled weight perturbations is applied. Again, adversarial perturbations detected through the testing are also classified as risky with the acceptable threshold of zero.

3. Finally, for the data points where no adversarial weight perturbations were detected in steps 1 and 2, it is probabilistically guaranteed that the existence rate of adversarial weight perturbations does not exceed the acceptable threshold θ^*.

In WP-GEB-Estimator-2, the acceptable threshold θ^* can be specified as an option ('–thr'), and the required sample size m of random perturbations for ensuring this guarantee is automatically calculated using the following formula.

$$m = \left\lceil log_{(1-\theta^*)} \left(\frac{r\delta}{n_0} \right) \right\rceil,$$

where n_0 represents the number of data points for which no adversarial perturbation was detected using the search, θ^* is the acceptable threshold, δ is the the confidence parameter (i.e., the probability with which the bound may fail), and r is the proportion to δ (typically, $r = 0.5$) such that $r\delta$ is the confidence parameter used for generalizing finite-samples of perturbations. The confidence parameter δ is used for accepting the gap between generalization errors and test errors, where the gap is caused by generalization of finite data samples and generalization of finite perturbation samples, and the proportion can be adjusted using r. Ideally, the acceptable threshold should be as small as possible, but it is practical for computational costs, to select reasonable thresholds such that the perturbation sample size is around thousands. For example, if $n_0 = 800$, $\theta^* = 0.02$, $\delta = 0.1$, and $r = 0.5$, then $m = \lceil 479.1... \rceil = 480$.

WP-GEB-Estimator-2 is a tool that estimate the upper bound with probabilistic confidence of the weight-perturbed generalization risk $\mathbf{R}^{\alpha}_{\theta}(f_w)$ and the upper bound of the generalization acceptable threshold Θ using the following acceptable threshold $\theta(x, y)$.

$$\theta_{(x,y)} := \begin{cases} 0 & \text{if } (x, y) \in T_1 \\ \theta^* & \text{otherwise} \end{cases}$$

where T_1 is the set of data points for which adversarial perturbations were detected through the search (i.e., a subset of the test dataset T). The method[4] used in WP-GEB-Estimator-2 to estimate the upper bounds of the weight-perturbed generalization risks is explained in a proof note [7] for WP-GEB-Estimator-2.

3 Introduction to WP-GEB-Estimator-2

In this section, the four tools that constitute WP-GEB-Estimator-2 are introduced in Subsection 3.1, and then input files that WP-GEB-Estimator-2 takes and output files that WP-GEB-Estimator-2 generates are explained in Subsections 3.2 and 3.3, respectively.

3.1 Tool Structure

WP-GEB-Estimator-2 consists of the four tools for training, measuring, searching, and estimating. These tools are described in the Python language with the TensorFlow/Keras libraries.

[4] They are similar to the method used in WP-GEB-Estimator[5] but have been modified.

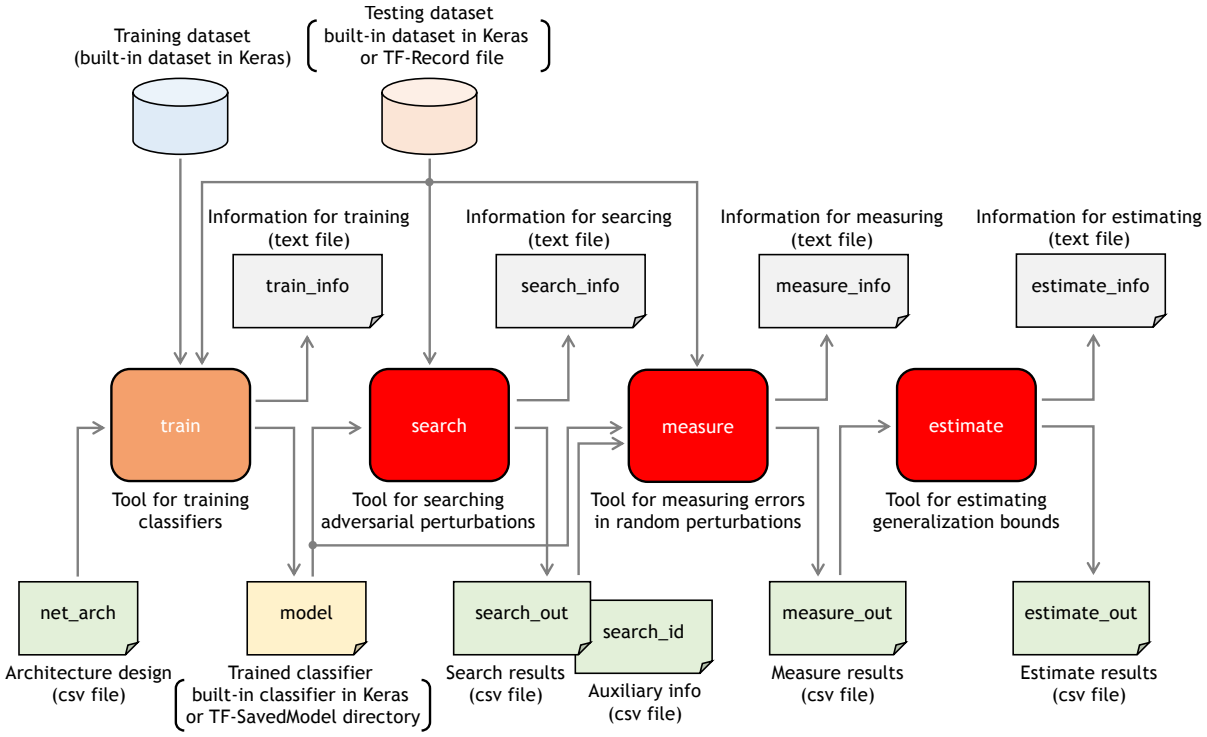

Figure 2: The relation between input/output files of the four tools that constitute WP-GEB-Estimator-2

- `train`: trains classifiers to demonstrate WP-GEB-Estimator-2.
  - Input: Train-dataset, Network architecture (CSV format), etc
  - Output: Trained classifier (TensorFlow-SavedModel format), etc
- `search`: searches for adversarial weight-perturbations in trained classifiers.
  - Input: Test-dataset, Trained classifier (SavedModel format), Perturbation-ratios, etc
  - Output: Adversarial search results (CSV format) including input information
- `measure`: measures misclassification-errors of trained classifiers for random weight-perturbations.
  - Input: Test-dataset, Trained classifier, Adversarial search results (CSV format), etc
  - Output: Measurement results (CSV format) including adversarial search results
- `estimate`: estimates weight-perturbed generalization risk/error bounds.
  - Input: Results of search and random perturbation sample (CSV format), etc
  - Output: Estimated results of weight-perturbed generalization bounds (CSV format), etc

The weight-perturbed generalization risk/error bounds of classifiers trained by the first tool `train` or the other tools can be estimated by sequentially executing the rest three tools: `search` → `measure` → `estimate`. The relation between input/output files of the four tools that constitute WP-GEB-Estimator-2 is shown in Figure 2. In the case of evaluating classifiers already trained by the other tools, it is not necessary to execute the train tool `train`. However, the search tool `search` must be executed before the measure tool `measure`, even though it is not necessary to search for adversarial weight-perturbations (i.e. generalization risks without search or generalization errors) because the tool `measure` 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 search. The input and output files are explained in the next Subsections 3.2 and 3.3.

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`						

3.2 Input files

The trained classifier to be evaluated and the test dataset are specified by the options `--model_dir` and `--dataset_file` in the search tool `search`. 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`.

3.3 Output files

The input/output information used in each tool `train`, `search`, `measure`, and `estimate` is saved in text-files `train_info.txt`, `search_info.txt`, `measure_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 `search_out.csv` and `measure_out.csv`. The estimated result for each weight-perturbation ratio is incrementally added as a new line to the CSV file `search_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 $\sim$ AK) of the CSV file `estimate_out.csv` is explained as follow:

- Input/Output data in `search` (also saved in A $\sim$ O of `search_out.csv`)
  - A [`dataset_name`] The name of test dataset
  - B [`dataset_size`] The size of test dataset
  - C [`dataset_offset`] The start index of test dataset
  - D [`dataset_file`] The file name of test dataset
  - E [`dataset_fmt`] The file format of test dataset
  - F [`image_width`] The width of images in test dataset
  - G [`image_height`] The height of images in test dataset
  - H [`model_dir`] The name or the directory name of classifier
  - I [`rnd_seed_search`] Random seed
  - J [`batch_size_search`] The number of data for gradient computation in parallel
  - K [`perturb_bn`] The perturbation on batch normalization parameters (`0`: not added, `1`: added)
  - L [`perturb_ratio`] The ratio of the maximum weight-perturbation to weight

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

N [max_iteration] The maximum iteration in I-FGSM

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

- Input/Output data in measure (also save in P ∼ Z in measure_out.csv)

P [rnd_seed_measure] Random seed

Q [batch_size_measure] The number of data for batch inference

R [err_thr] The acceptable threshold

S [err_thr_practical] The practical acceptable threshold used in fact

T [delta] The confidence parameter

U [delta0_ratio] the proportion to δ such that $r\delta$ is the confidence parameter used for generalizing finite perturbations.

V [perturb_params_size] The number of perturbed weight-parameters

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

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

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

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

- Input/Output data in estimate

  – Worst weight-perturbation

  AA [gen_risk_ub] Perturbed generalization risk upper bound

  AB [test_risk_ub] Perturbed test risk upper bound

  AC [conf_risk] Confidence of perturbed generalization risk upper bound

  AD [conf0_risk] Confidence of perturbed test risk upper bound

  AE [non_det_rate_ub] Non-detection rate (by search) upper bound

  AF [gen_err_thr_ub] Generalization acceptable threshold upper bound

  – Random weight-perturbation

  AG [gen_err_ub] Perturbed generalization error upper bound

  AH [test_err_ub] Perturbed test error upper bound

  AI [test_err] Test error by perturbation samples

  AJ [conf_err] Confidence of perturbed generalization error upper bound

  AK [conf0_err] Confidence of perturbed test error upper bound

4 Execution of WP-GEB-Estimator-2

In this section, it is explained how to execute each tool in WP-GEB-Estimator-2 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 built-in datasets and classifiers that are available in TensorFlow/Keras.

$$\texttt{KerasDataSets} = \{\texttt{"mnist"}, \texttt{"fashion\_mnist"}, \texttt{"cifar10"}\} \subset \mathbb{S}$$

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

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 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.)

`--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.

`--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 saving the search results to s.

`--perturb_ratios` s ($s \in \mathbb{S}\mathbb{R}^*$, 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.)

`--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`.

4.3 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.

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

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

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

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

`--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.)

`--err_thr` r ($r \in (0,1) \subset \mathbb{R}$, default $r =$ `0.01`)

sets the acceptable threshold of error-rate to r.

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

sets the size of random perturbation sample to n. (n should be `0` for worst perturbations)
(the size is appropriately decided from the acceptable threshold if $n =$ `0`)

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

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

`--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.)

4.4 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.

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

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

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

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

`--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-2

In this section, examples of execution commands for WP-GEB-Estimator-2 and some execution results are introduced. WP-GEB-Estimator-2 is described in the Python language with the libraries TensorFlow and NumPy. The software versions used in the development are Python 3.10.16, TensorFlow 2.16.2, Keras 3.6.0, and NumPy 1.26.4. Refer to the attached file, `requirements.txt`, for more details.

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, searching for adversarial weight-perturbations, measuring the errors when random weight-perturbations are added, 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, acceptable threshold: 1% corresponding to the weight-perturbation sample size: ∼1146), about 90 seconds for training, about 70 seconds for searching, about 4 minutes for measuring, and less than 0.1 seconds for estimating.

```
python train_main.py  --net_arch_file "net_arch/mlp_s_bn"

# with search of adversarial perturbations by FGSM
python search_main.py  --skip_search 0
python measure_main.py
python estimate_main.py

# without search
python search_main.py  --skip_search 1
python measure_main.py
python estimate_main.py
```

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

```
---with search---
Perturbation ratio = 0.01
  Random perturbation sample size: 1117
  Risk (with search):
    Perturbed generalization risk bound: 26.74% (Conf: 90.00%)
    Perturbed test risk bound: 25.22% (Conf: 95.00%)
    Generalization acceptable threshold bound: 0.7608% (Conf: 90.00%)
---without search---
Perturbation ratio = 1.0
  Random perturbation sample size: 1146
  Risk (without search):
    Perturbed generalization risk bound: 100.00% (Conf: 100.00%)
    Perturbed test risk bound: 100.00% (Conf: 100.00%)
    Generalization acceptable threshold bound: 0.0000% (Conf: 100.00%)
  Error:
    Perturbed generalization error bound: 32.80% (Conf: 90.00%)
    Perturbed test error bound: 30.17% (Conf: 95.00%)
```

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

Figure 4 is a part of the result `estimate_info.txt` for the execution script `run.sh`. It shows the estimated results of the weight-perturbed generalization bounds, test error bounds, and the thresholds.

```
ImageNet_DIR="$HOME/datasets/imagenet/1k-tfrecords/validation"

# with search of adversarial perturbations by FGSM
python search_main.py \
    --skip_search 0 \
    --result_dir "result_imagenet" --dataset_name "imagenet" \
    --dataset_file "$ImageNet_DIR/validation-*-of-00128" \
    --model_dir "inception_v3"  --dataset_size 1000 \
    --perturb_ratios "0.0001"
python measure_main.py \
    --result_dir "result_imagenet" --err_thr 0.02
python estimate_main.py --result_dir "result_imagenet"

# without search
python search_main.py \
    --skip_search 1 \
    ...
```

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

For example, for the ratio 0.01 of the weight-perturbation to the weight, the weight-perturbed generalization risk is less than 27% with probability at least 90%, where the generalization acceptable threshold is less than 0.75%, and for the ratio 1 of the weight-perturbation to the weight, the weight-perturbed generalization error is less than 33% with probability at least 90%.

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

Figure 5 is an execution script `run_imagenet.sh` for estimating generalization risk/error bounds of the pre-trained built-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 the acceptable threshold 2% (corresponds to weight-perturbation sample size 472∼491), about 10 minute for searching, about 20∼30 minutes for measuring, which depends on the perturbation sample size, and less than 0.1 seconds for estimating. The estimated result was as follows: for the ratio 0.0001 of the weight-perturbation to the weight, the weight-perturbed generalization risk is less than 36% with probability at least 90%, where the generalization acceptable threshold is less than 1.4% and the weight-perturbed generalization error is less than 33% with probability at least 90%. Refer to the poster [6] for estimated results of generalization bounds for the other classifiers of ImageNet.

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 risk/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 estimated results by WP-GEB-Estimator-2 and they are the weight perturbed generalization risk/error bounds (confidence 90%) and the generalization acceptable thresholds, for the classifier without regularization and the classifier with the $L2$-regularization coefficient 0.001.

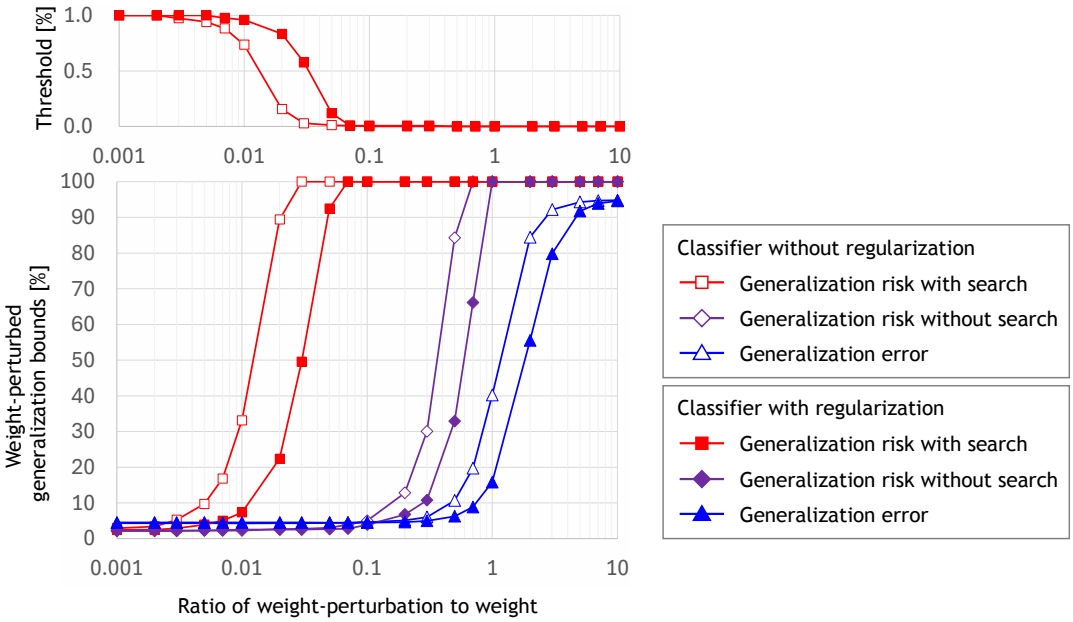

Figure 6: The estimated results of weight-perturbed generalization bounds (conf. 90%)

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 weight-perturbed generalization risk bounds with search start increasing at the ratio that is two orders less than the ratio at which 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 (written in Japanese), The 38th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI), 2024.

[6] Y. Isobe, Probabilistic safety certification of neural classifiers against adversarial perturbations (written in Japanese), AIST Cyber-Physical Security Research Symposium, 2025.

[7] Y. Isobe, A Proof Note of Weight-Perturbed Generalization Error Bounds, 2025 (to be published in the web-site of WP-GEB-Estimator-2)

Acknowledgment

This tool WP-GEB-Estimator-2 has been partially developed in the project JPNP20006 commissioned by the New Energy and Industrial Technology Development Organization (NEDO).