Package ‘kerastuneR’

March 25, 2022

Type Package

Title Interface to 'Keras Tuner'

Version 0.1.0.5

Maintainer Turgut Abdullayev <turqut.a.314@gmail.com>

Description 'Keras Tuner' <https://keras-team.github.io/keras-tuner/> is a hypertuning framework made for humans.
It aims at making the life of AI practitioners, hypertuner
algorithm creators and model designers as simple as possible by
providing them with a clean and easy to use API for hypertuning.
'Keras Tuner' makes moving from a base model to a hypertuned one quick and
easy by only requiring you to change a few lines of code.

License Apache License 2.0

URL https://github.com/EagerAI/kerastuneR/

BugReports https://github.com/EagerAI/kerastuneR/issues/

SystemRequirements TensorFlow >= 2.0 (https://www.tensorflow.org/)

Encoding UTF-8

RoxygenNote 7.1.2

Imports reticulate, tensorflow, rstudioapi, plotly, data.table,
RJSONIO, rjson, tidyjson, dplyr, echarts4r, crayon, keras,
magick

Suggests knitr, tfdatasets, testthat, purrr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Turgut Abdullayev [aut, cre],
Google Inc. [cph]

Repository CRAN

Date/Publication 2022-03-25 08:50:02 UTC

R topics documented:

`BaseTuner` *Base Tuner*

Description

Tuner base class.

Usage

```
BaseTuner(
  oracle,
  hypermodel,
  directory = NULL,
  project_name = NULL,
  logger = NULL,
  overwrite = FALSE
)
```

Arguments

`oracle`	Instance of Oracle class.
`hypermodel`	Instance of HyperModel class (or callable that takes hyperparameters and returns a Model instance).
`directory`	String. Path to the working directory (relative).
`project_name`	Name to use as prefix for files saved by this Tuner.
`logger`	Optional. Instance of Logger class, used for streaming data to Cloud Service for monitoring.
`overwrite`	Bool, default ‘FALSE‘. If ‘FALSE‘, reloads an existing project of the same name if one is found. Otherwise, overwrites the project.

Details

May be subclassed to create new tuners, including for non-Keras models.

Value

base tuner object

`BayesianOptimization` *BayesianOptimization*

Description

Bayesian optimization oracle.

Usage

```
BayesianOptimization(
  objective,
  max_trials,
  num_initial_points = NULL,
  alpha = 1e-04,
  beta = 2.6,
  seed = NULL,
  hyperparameters = NULL,
  allow_new_entries = TRUE,
  tune_new_entries = TRUE
)
```

Arguments

`objective`	String or ‘kerastuner.Objective‘. If a string, the direction of the optimization (min or max) will be inferred.
`max_trials`	Int. Total number of trials (model configurations) to test at most. Note that the oracle may interrupt the search before ‘max_trial‘ models have been tested if the search space has been exhausted.
`num_initial_points`	(Optional) Int. The number of randomly generated samples as initial training data for Bayesian optimization. If not specified, a value of 3 times the dimensionality of the hyperparameter space is used.
`alpha`	Float. Value added to the diagonal of the kernel matrix during fitting. It represents the expected amount of noise in the observed performances in Bayesian optimization.
`beta`	Float. The balancing factor of exploration and exploitation. The larger it is, the more explorative it is.
`seed`	Int. Random seed.
`hyperparameters`	HyperParameters class instance. Can be used to override (or register in advance) hyperparamters in the search space.
`allow_new_entries`	Whether the hypermodel is allowed to request hyperparameter entries not listed in ‘hyperparameters‘.
`tune_new_entries`	Whether hyperparameter entries that are requested by the hypermodel but that were not specified in ‘hyperparameters‘ should be added to the search space, or not. If not, then the default value for these parameters will be used.

Details

It uses Bayesian optimization with a underlying Gaussian process model. The acquisition function used is upper confidence bound (UCB), which can be found in the following link: https://www.cse.wustl.edu/~garnett/cse515t

Value

BayesianOptimization tuning with Gaussian process

be found in the following link

https://www.cse.wustl.edu/~garnett/cse515t/spring_2015/files/lecture_notes/12.pdf

Examples

```
## Not run:
# The usage of 'tf$keras'
library(tensorflow)
tf$keras$Input(shape=list(28L, 28L, 1L))
```

```
## End(Not run)
```

`callback_tuner` *Tuner Callback*

Description

Abstract base class used to build new callbacks.

Usage

```
callback_tuner(tuner, trial)
```

Arguments

`tuner`	tuner object
`trial`	trial ID

Details

Attributes: params: dict. Training parameters (eg. verbosity, batch size, number of epochs...). model: instance of ‘keras.models.Model‘. Reference of the model being trained. validation_data: Deprecated. Do not use. The ‘logs‘ dictionary that callback methods take as argument will contain keys for quantities relevant to the current batch or epoch. Currently, the ‘.fit()‘ method of the ‘Model‘ class will include the following quantities in the ‘logs‘ that it passes to its callbacks: on_epoch_end: logs include ‘acc‘ and ‘loss‘, and optionally include ‘val_loss‘ (if validation is enabled in ‘fit‘), and ‘val_acc‘ (if validation and accuracy monitoring are enabled). on_batch_begin: logs include ‘size‘, the number of samples in the current batch. on_batch_end: logs include ‘loss‘, and optionally ‘acc‘ (if accuracy monitoring is enabled).

Value

None

Attributes

params: dict. Training parameters (eg. verbosity, batch size, number of epochs...). model: instance of ‘keras.models.Model‘. Reference of the model being trained. validation_data: Deprecated. Do not use.

fit_tuner *Search*

Description

Start the search for the best hyperparameter configuration. The call to search has the same signature as “‘model.fit()“‘. Models are built iteratively by calling the model-building function, which populates the hyperparameter space (search space) tracked by the hp object. The tuner progressively explores the space, recording metrics for each configuration.

Usage

```
fit_tuner(
  tuner,
  x = NULL,
  y = NULL,
  steps_per_epoch = NULL,
  batch_size = NULL,
  epochs = NULL,
  validation_data = NULL,
  validation_steps = NULL,
  ...
)
```

Arguments

`tuner`	A tuner object
`x`	Vector, matrix, or array of training data (or list if the model has multiple inputs). If all inputs in the model are named, you can also pass a list mapping input names to data. x can be NULL (default) if feeding from framework-native tensors (e.g. TensorFlow data tensors).
`y`	Vector, matrix, or array of target (label) data (or list if the model has multiple outputs). If all outputs in the model are named, you can also pass a list mapping output names to data. y can be NULL (default) if feeding from framework-native tensors (e.g. TensorFlow data tensors).
`steps_per_epoch`	Integer. Total number of steps (batches of samples) to yield from generator before declaring one epoch finished and starting the next epoch. It should typically be equal to ceil(num_samples / batch_size). Optional for Sequence: if unspecified, will use the len(generator) as a number of steps.
`batch_size`	Integer or ‘NULL‘. Number of samples per gradient update. If unspecified, ‘batch_size‘ will default to 32.
`epochs`	to train the model. Note that in conjunction with initial_epoch, epochs is to be understood as "final epoch". The model is not trained for a number of iterations given by epochs, but merely until the epoch of index epochs is reached.

`validation_data`
: Data on which to evaluate the loss and any model metrics at the end of each epoch. The model will not be trained on this data. validation_data will override validation_split. validation_data could be: - tuple (x_val, y_val) of Numpy arrays or tensors - tuple (x_val, y_val, val_sample_weights) of Numpy arrays - dataset or a dataset iterator

`validation_steps`
: Only relevant if steps_per_epoch is specified. Total number of steps (batches of samples) to validate before stopping.

`...`
: Some additional arguments

Value

performs a search for best hyperparameter configuations

Examples

```

## Not run:

library(keras)
x_data <- matrix(data = runif(500,0,1),nrow = 50,ncol = 5)
y_data <-  ifelse(runif(50,0,1) > 0.6, 1L,0L) %>% as.matrix()
x_data2 <- matrix(data = runif(500,0,1),nrow = 50,ncol = 5)
y_data2 <-  ifelse(runif(50,0,1) > 0.6, 1L,0L) %>% as.matrix()


HyperModel <- PyClass(
  'HyperModel',
  inherit = HyperModel_class(),
  list(

    `__init__` = function(self, num_classes) {

      self$num_classes = num_classes
      NULL
    },
    build = function(self,hp) {
      model = keras_model_sequential()
      model %>% layer_dense(units = hp$Int('units',
                                     min_value = 32,
                                     max_value = 512,
                                     step = 32),
                        input_shape = ncol(x_data),
                        activation = 'relu') %>%
        layer_dense(as.integer(self$num_classes), activation = 'softmax') %>%
        compile(
          optimizer = tf$keras$optimizers$Adam(
            hp$Choice('learning_rate',
                      values = c(1e-2, 1e-3, 1e-4))),
          loss = 'sparse_categorical_crossentropy',
          metrics = 'accuracy')
    }
  )
)

hypermodel = HyperModel(num_classes=10L)


tuner = RandomSearch(hypermodel = hypermodel,
                     objective = 'val_accuracy',
                     max_trials = 2,
                    executions_per_trial = 1,
                     directory = 'my_dir5',
                     project_name = 'helloworld')

tuner %>% fit_tuner(x_data, y_data, epochs = 1, validation_data = list(x_data2,y_data2))

## End(Not run)
```

`get_best_models` *Get best models*

Description

The function for retrieving the top best models with hyperparameters Returns the best model(s), as determined by the tuner's objective. The models are loaded with the weights corresponding to their best checkpoint (at the end of the best epoch of best trial). This method is only a convenience shortcut. For best performance, It is recommended to retrain your Model on the full dataset using the best hyperparameters found during search.

Usage

```
get_best_models(tuner = NULL, num_models = NULL)
```

Arguments

`tuner`	A tuner object
`num_models`	When search is over, one can retrieve the best model(s)

Value

the list of best model(s)

Hyperband *Hyperband*

Description

Variation of HyperBand algorithm.

Usage

```
Hyperband(
  hypermodel,
  optimizer = NULL,
  loss = NULL,
  metrics = NULL,
  hyperparameters = NULL,
  objective,
  max_epochs,
  factor = 3,
  hyperband_iterations = 1,
  seed = NULL,
  tune_new_entries = TRUE,
  allow_new_entries = TRUE,
  distribution_strategy = NULL,
  directory = NULL,
  project_name = NULL,
  ...
)
```

Arguments

`hypermodel`	Define a model-building function. It takes an argument "hp" from which you can sample hyperparameters.
`optimizer`	An optimizer is one of the arguments required for compiling a Keras model
`loss`	A loss function (or objective function, or optimization score function) is one of the parameters required to compile a model
`metrics`	A metric is a function that is used to judge the performance of your model
`hyperparameters`	HyperParameters class instance. Can be used to override (or register in advance) hyperparamters in the search space.
`objective`	A loss metrics function for tracking the model performance e.g. "val_precision". The name of the objective to optimize (whether to minimize or maximize is automatically inferred for built-in metrics)
`max_epochs`	to train the model. Note that in conjunction with initial_epoch, epochs is to be understood as "final epoch". The model is not trained for a number of iterations given by epochs, but merely until the epoch of index epochs is reached.

`factor`	Int. Reduction factor for the number of epochs and number of models for each bracket.
`hyperband_iterations`	Int >= 1. The number of times to iterate over the full Hyperband algorithm. One iteration will run approximately "'max_epochs * (math.log(max_epochs, factor) ** 2)"' cumulative epochs across all trials. It is recommended to set this to as high a value as is within your resource budget.
`seed`	Int. Random seed.
`tune_new_entries`	Whether hyperparameter entries that are requested by the hypermodel but that were not specified in hyperparameters should be added to the search space, or not. If not, then the default value for these parameters will be used.
`allow_new_entries`	Whether the hypermodel is allowed to request hyperparameter entries not listed in 'hyperparameters'. **kwargs: Keyword arguments relevant to all 'Tuner' subclasses. Please see the docstring for 'Tuner'.
`distribution_strategy`	Scale up from running single-threaded locally to running on dozens or hundreds of workers in parallel. Distributed Keras Tuner uses a chief-worker model. The chief runs a service to which the workers report results and query for the hyperparameters to try next. The chief should be run on a single-threaded CPU instance (or alternatively as a separate process on one of the workers). Keras Tuner also supports data parallelism via tf.distribute. Data parallelism and distributed tuning can be combined. For example, if you have 10 workers with 4 GPUs on each worker, you can run 10 parallel trials with each trial training on 4 GPUs by using tf.distribute.MirroredStrategy. You can also run each trial on TPUs via tf.distribute.experimental.TPUStrategy. Currently tf.distribute.MultiWorkerMirroredStrategy is not supported, but support for this is on the roadmap.
`directory`	The dir where training logs are stored
`project_name`	Detailed logs, checkpoints, etc, in the folder my_dir/helloworld, i.e. directory/project_name.
`...`	Some additional arguments

Details

Reference: Li, Lisha, and Kevin Jamieson. ["Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization." Journal of Machine Learning Research 18 (2018): 1-52](http://jmlr.org/papers/v18/16-558.html). # Arguments hypermodel: Instance of HyperModel class (or callable that takes hyperparameters and returns a Model instance). objective: String. Name of model metric to minimize or maximize, e.g. "val_accuracy". max_epochs: Int. The maximum number of epochs to train one model. It is recommended to set this to a value slightly higher than the expected time to convergence for your largest Model, and to use early stopping during training (for example, via 'tf.keras.callbacks.EarlyStopping'). factor: Int. Reduction factor for the number of epochs and number of models for each bracket. hyperband_iterations: Int >= 1. The number of times to iterate over the full Hyperband algorithm. One iteration will run approximately 'max_epochs * (math.log(max_epochs, factor) ** 2)' cumulative epochs across all trials. It is recommended to set this to as high a value as is within your resource budget. seed: Int. Random seed. hyperparameters:

HyperParameters class instance. Can be used to override (or register in advance) hyperparamters in the search space. tune_new_entries: Whether hyperparameter entries that are requested by the hypermodel but that were not specified in ‘hyperparameters‘ should be added to the search space, or not. If not, then the default value for these parameters will be used. allow_new_entries: Whether the hypermodel is allowed to request hyperparameter entries not listed in ‘hyperparameters‘. **kwargs: Keyword arguments relevant to all ‘Tuner‘ subclasses. Please see the docstring for ‘Tuner‘.

Value

a hyperparameter tuner object Hyperband

Reference

Li, Lisha, and Kevin Jamieson. ["Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization." Journal of Machine Learning Research 18 (2018): 1-52](http://jmlr.org/papers/v18/16-558.html).

`HyperModel_class` *HyperModel*

Description

Defines a searchable space of Models and builds Models from this space.

Usage

```
HyperModel_class(name = NULL, tunable = TRUE)
```

Arguments

`name`	The name of this HyperModel.
`tunable`	Whether the hyperparameters defined in this hypermodel should be added to search space. If ‘FALSE‘, either the search space for these parameters must be defined in advance, or the default values will be used.

Value

None

HyperParameters *HyperParameters*

Description

The HyperParameters class serves as a hyperparameter container. A HyperParameters instance contains information about both the search space and the current values of each hyperparameter. Hyperparameters can be defined inline with the model-building code that uses them. This saves you from having to write boilerplate code and helps to make the code more maintainable.

Usage

```
HyperParameters(...)
```

Arguments

`...`	Pass hyperparameter arguments to the tuner constructor

Value

container for both a hyperparameter space, and current values

HyperResNet *HyperResNet*

Description

A ResNet HyperModel.

Usage

```
HyperResNet(
  include_top = TRUE,
  input_shape = NULL,
  input_tensor = NULL,
  classes = NULL,
  ...
)
```

Arguments

`include_top`	whether to include the fully-connected layer at the top of the network.
`input_shape`	Optional shape list, e.g. '(256, 256, 3)'. One of 'input_shape' or 'input_tensor' must be specified.
`input_tensor`	Optional Keras tensor (i.e. output of 'layers.Input()') to use as image input for the model. One of 'input_shape' or 'input_tensor' must be specified.
`classes`	optional number of classes to classify images into, only to be specified if 'include_top' is TRUE, and if no 'weights' argument is specified. **kwargs: Additional keyword arguments that apply to all HyperModels. See 'kerastuner.HyperModel'.
`...`	Additional keyword arguments that apply to all HyperModels.

Value

a pre-trained ResNet model

Examples

```
## Not run:


cifar <- dataset_cifar10()

hypermodel = HyperResNet(input_shape = list(32L, 32L, 3L), classes = 10L)
hypermodel2 = HyperXception(input_shape = list(32L, 32L, 3L), classes = 10L)


tuner = Hyperband(
  hypermodel = hypermodel,
  objective = 'accuracy',
  loss = 'sparse_categorical_crossentropy',
  max_epochs = 1,
  directory = 'my_dir',
  project_name='helloworld')


train_data = cifar$train$x[1:30,1:32,1:32,1:3]
test_data = cifar$train$y[1:30,1] %>% as.matrix()


tuner %>% fit_tuner(train_data,test_data, epochs = 1)

## End(Not run)
```

`HyperXception` *HyperXception*

Description

An Xception HyperModel.

Usage

```
HyperXception(
  include_top = TRUE,
  input_shape = NULL,
  input_tensor = NULL,
  classes = NULL,
  ...
)
```

Arguments

`include_top`	whether to include the fully-connected layer at the top of the network.
`input_shape`	Optional shape list, e.g. '(256, 256, 3)'. One of 'input_shape' or 'input_tensor' must be specified.
`input_tensor`	Optional Keras tensor (i.e. output of 'layers.Input()') to use as image input for the model. One of 'input_shape' or 'input_tensor' must be specified.
`classes`	optional number of classes to classify images into, only to be specified if 'include_top' is TRUE, and if no 'weights' argument is specified. **kwargs: Additional keyword arguments that apply to all HyperModels. See 'kerastuner.HyperModel'.
`...`	Additional keyword arguments that apply to all HyperModels.

Value

a pre-trained Xception model

`install_kerastuner` *Install Keras Tuner*

Description

This function is used to install the Keras Tuner python module

Usage

```
install_kerastuner(
  version = NULL,
  ...,
  restart_session = TRUE,
  from_git = FALSE
)
```

Arguments

`version`	for specific version of Keras Tuner, e.g. "1.0.1"
`...`	other arguments passed to [reticulate::py_install()].
`restart_session`	Restart R session after installing (note this will only occur within RStudio).
`from_git`	install the recent GitHub version of Keras Tuner

Value

a python module kerastuner

`keras_tuner_version` *Version of Keras Tuner*

Description

Get the current version of Keras Tuner

Usage

```
keras_tuner_version()
```

Value

prints the version.

`load_model` *Load model*

Description

Loads a Model from a given trial

Usage

```
load_model(tuner, trial)
```

Arguments

`tuner`	A tuner object
`trial`	A ‘Trial‘ instance. For models that report intermediate results to the ‘Oracle‘, generally ‘load_model‘ should load the best reported ‘step‘ by relying of ‘trial.best_step‘

Value

None

`Objective` *Objective*

Description

Objective(name, direction) includes strings, the direction of the optimization (min or max) will be inferred.

Usage

```
Objective(name, direction, ...)
```

Arguments

`name`	name
`direction`	direction
`...`	Some additional arguments

Value

None

`Oracle` *Oracle*

Description

Implements a hyperparameter optimization algorithm.

Usage

```
Oracle(
  objective,
  max_trials = NULL,
  hyperparameters = NULL,
  allow_new_entries = TRUE,
  tune_new_entries = TRUE
)
```

Arguments

`objective`	String. Name of model metric to minimize or maximize, e.g. "val_accuracy".
`max_trials`	The maximum number of hyperparameter combinations to try.
`hyperparameters`	HyperParameters class instance. Can be used to override (or register in advance) hyperparamters in the search space.
`allow_new_entries`	Whether the hypermodel is allowed to request hyperparameter entries not listed in ‘hyperparameters‘.
`tune_new_entries`	Whether hyperparameter entries that are requested by the hypermodel but that were not specified in ‘hyperparameters‘ should be added to the search space, or not. If not, then the default value for these parameters will be used.

Value

None

`plot_keras_model` *Plot Keras model*

Description

Converts a Keras model to dot format and save to a file.

Usage

```
plot_keras_model(
  model,
  to_file = "model.png",
  show_shapes = FALSE,
  show_layer_names = TRUE,
  rankdir = "TB",
  expand_nested = FALSE,
  dpi = 96
)
```

Arguments

`model`	A Keras model instance
`to_file`	File name of the plot image.
`show_shapes`	whether to display shape information.
`show_layer_names`	whether to display layer names.
`rankdir`	‘rankdir‘ argument passed to PyDot, a string specifying the format of the plot: ’TB’ creates a vertical plot; ’LR’ creates a horizontal plot.
`expand_nested`	Whether to expand nested models into clusters.
`dpi`	Dots per inch.

Value

saves a png image on the system and builds a plot in R

`plot_tuner` *Plot the tuner results with ’plotly’*

Description

Plot the search space results

Usage

```
plot_tuner(tuner, height = NULL, width = NULL, type = "plotly")
```

Arguments

`tuner`	A tuner object
`height`	height of the plot
`width`	width of the plot
`type`	Type parameter has 2 options: * By default it uses ‘plotly‘ * Second option is ‘echarts4r‘ **Note** that ‘echarts4r‘ ignores width and height parameters

Value

a list which contains a dataframe of results and a plot

`RandomSearch` *RandomSearch*

Description

Random search tuner.

Usage

```
RandomSearch(
  hypermodel,
  objective,
  max_trials,
  seed = NULL,
  hyperparameters = NULL,
  tune_new_entries = TRUE,
  allow_new_entries = TRUE,
  executions_per_trial = NULL,
  directory = NULL,
  project_name = NULL,
  ...
)
```

Arguments

`hypermodel`	Define a model-building function. It takes an argument "hp" from which you can sample hyperparameters.
`objective`	A loss metrics function for tracking the model performance e.g. "val_precision". The name of the objective to optimize (whether to minimize or maximize is automatically inferred for built-in metrics)
`max_trials`	the total number of trials (max_trials) to test
`seed`	Int. Random seed
`hyperparameters`	HyperParameters class instance. Can be used to override (or register in advance) hyperparamters in the search space
`tune_new_entries`	Whether hyperparameter entries that are requested by the hypermodel but that were not specified in hyperparameters should be added to the search space, or not. If not, then the default value for these parameters will be used.
`allow_new_entries`	Whether the hypermodel is allowed to request hyperparameter entries not listed in hyperparameters

`executions_per_trial`	the number of models that should be built and fit for each trial (executions_per_trial). Note: the purpose of having multiple executions per trial is to reduce results variance and therefore be able to more accurately assess the performance of a model. If you want to get results faster, you could set executions_per_trial=1 (single round of training for each model configuration)
`directory`	The dir where training logs are stored
`project_name`	Detailed logs, checkpoints, etc, in the folder my_dir/helloworld, i.e. directory/project_name.
`...`	Some additional arguments

Value

a hyperparameter tuner object RandomSearch

Examples

```
## Not run:

x_data <- matrix(data = runif(500,0,1),nrow = 50,ncol = 5)
y_data <-  ifelse(runif(50,0,1) > 0.6, 1L,0L) %>% as.matrix()
x_data2 <- matrix(data = runif(500,0,1),nrow = 50,ncol = 5)
y_data2 <-  ifelse(runif(50,0,1) > 0.6, 1L,0L) %>% as.matrix()

build_model = function(hp) {
 model = keras_model_sequential()
  model %>% layer_dense(units=hp$Int('units',
                                     min_value=32L,
                                     max_value=512L,
                                     step=32L),
                                     input_shape = ncol(x_data),
                                     activation='relu') %>%
   layer_dense(units=1L, activation='softmax') %>%
   compile(
     optimizer= tf$keras$optimizers$Adam(
       hp$Choice('learning_rate',
                 values=c(1e-2, 1e-3, 1e-4))),
    loss='binary_crossentropy',
     metrics='accuracy')
     return(model)
 }
 tuner = RandomSearch(hypermodel = build_model,
                       objective = 'val_accuracy',
                       max_trials = 2,
                       executions_per_trial = 1,
                       directory = 'model_dir',
                       project_name = 'helloworld')

## End(Not run)
```

`results_summary` *Results summary*

Description

Print a summary of the search results (best models)

Usage

```
results_summary(tuner = NULL, num_trials = NULL)
```

Arguments

`tuner`	Requires a tuner object
`num_trials`	Shows the top best models

Value

the list of results summary of the tuner object

`save_model` *Save model*

Description

Saves a Model for a given trial

Usage

```
save_model(tuner, trial_id, model, step = 1)
```

Arguments

`tuner`	A tuner object
`trial_id`	The ID of the ‘Trial‘ that corresponds to this Model.
`model`	The trained model.
`step`	For models that report intermediate results to the ‘Oracle‘, the step that this saved file should correspond to. For example, for Keras models this is the number of epochs trained.

Value

None

search_summary *Search summary*

Description

Print a summary of the search space

Usage

```
search_summary(tuner = NULL)
```

Arguments

tuner	Requires a tuner object

Value

the summary of search space of the tuner object

TensorBoard *TensorBoard*

Description

Enable visualizations for TensorBoard.

Usage

```
TensorBoard(
  log_dir = "logs",
  histogram_freq = 0,
  write_graph = TRUE,
  write_images = FALSE,
  update_freq = "epoch",
  profile_batch = 2,
  embeddings_freq = 0,
  embeddings_metadata = NULL
)
```

Arguments

`log_dir`	the path of the directory where to save the log files to be parsed by TensorBoard.
`histogram_freq`	frequency (in epochs) at which to compute activation and weight histograms for the layers of the model. If set to 0, histograms won't be computed. Validation data (or split) must be specified for histogram visualizations.
`write_graph`	whether to visualize the graph in TensorBoard. The log file can become quite large when write_graph is set to TRUE.
`write_images`	whether to write model weights to visualize as image in TensorBoard.
`update_freq`	''batch'' or ''epoch'' or integer. When using ''batch'', writes the losses and metrics to TensorBoard after each batch. The same applies for ''epoch''. If using an integer, let's say '1000', the callback will write the metrics and losses to TensorBoard every 1000 samples. Note that writing too frequently to TensorBoard can slow down your training.
`profile_batch`	Profile the batch to sample compute characteristics. By default, it will profile the second batch. Set profile_batch=0 to disable profiling. Must run in TensorFlow eager mode.
`embeddings_freq`	frequency (in epochs) at which embedding layers will be visualized. If set to 0, embeddings won't be visualized.
`embeddings_metadata`	a dictionary which maps layer name to a file name in which metadata for this embedding layer is saved. See the [details](https://www.tensorflow.org/how_tos/embedding_viz/#metadata_ about metadata files format. In case if the same metadata file is used for all embedding layers, string can be passed.

Details

TensorBoard is a visualization tool provided with TensorFlow. This callback logs events for TensorBoard, including: * Metrics summary plots * Training graph visualization * Activation histograms * Sampled profiling If you have installed TensorFlow with pip, you should be able to launch TensorBoard from the command line: "'sh tensorboard –logdir=path_to_your_logs "' You can find more information about TensorBoard [here](https://www.tensorflow.org/get_started/summaries_and_tensorboard).

Value

None

Raises

ValueError: If histogram_freq is set and no validation data is provided.

`Tuner_class` *Tuner*

Description

Tuner class for Keras models.

Usage

```
Tuner_class(
  oracle,
  hypermodel,
  max_model_size = NULL,
  optimizer = NULL,
  loss = NULL,
  metrics = NULL,
  distribution_strategy = NULL,
  directory = NULL,
  project_name = NULL,
  logger = NULL,
  tuner_id = NULL,
  overwrite = FALSE
)
```

Arguments

`oracle`	Instance of Oracle class.
`hypermodel`	Instance of HyperModel class (or callable that takes hyperparameters and returns a Model instance).
`max_model_size`	Int. Maximum size of weights (in floating point coefficients) for a valid models. Models larger than this are rejected.
`optimizer`	Optional. Optimizer instance. May be used to override the ‘optimizer‘ argument in the ‘compile‘ step for the models. If the hypermodel does not compile the models it generates, then this argument must be specified.
`loss`	Optional. May be used to override the ‘loss‘ argument in the ‘compile‘ step for the models. If the hypermodel does not compile the models it generates, then this argument must be specified.
`metrics`	Optional. May be used to override the ‘metrics‘ argument in the ‘compile‘ step for the models. If the hypermodel does not compile the models it generates, then this argument must be specified.
`distribution_strategy`	Optional. A TensorFlow ‘tf$distribute‘ DistributionStrategy instance. If specified, each trial will run under this scope. For example, ‘tf$distribute.MirroredStrategy(['/gpu:0, /'gpu:1])‘ will run each trial on two GPUs. Currently only single-worker strategies are supported.

`directory`	String. Path to the working directory (relative).
`project_name`	Name to use as prefix for files saved by this Tuner.
`logger`	Optional. Instance of Logger class, used for streaming data to Cloud Service for monitoring.
`tuner_id`	tuner_id
`overwrite`	Bool, default ‘FALSE‘. If ‘FALSE‘, reloads an existing project of the same name if one is found. Otherwise, overwrites the project.

Details

May be subclassed to create new tuners.

Value

a tuner object

Index