[[Tensorboard]](https://keras.io/callbacks/#tensorboard)

tf.keras.callbacks.TensorBoard(log\_dir='./logs',

histogram\_freq=0,

batch\_size=32,

write\_graph=True,

write\_grads=False,

write\_images=False,

embeddings\_freq=0,

embeddings\_layer\_names=None,

embeddings\_metadata=None,

embeddings\_data=None,

update\_freq='epoch')

* **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.
* **batch\_size**: size of batch of inputs to feed to the network for histograms computation.
* **write\_graph**: whether to visualize the graph in TensorBoard. The log file can become quite large when write\_graph is set to True.
* **write\_grads**: whether to visualize gradient histograms in TensorBoard. histogram\_freq must be greater than 0.
* **write\_images**: whether to write model weights to visualize as image in TensorBoard.
* **embeddings\_freq**: frequency (in epochs) at which selected embedding layers will be saved. If set to 0, embeddings won't be computed. Data to be visualized in TensorBoard's Embedding tab must be passed as embeddings\_data.
* **embeddings\_layer\_names**: a list of names of layers to keep eye on. If None or empty list all the embedding layer will be watched.
* **embeddings\_metadata**: a dictionary which maps layer name to a file name in which metadata for this embedding layer is saved. See the details about metadata files format. In case if the same metadata file is used for all embedding layers, string can be passed.
* **embeddings\_data**: data to be embedded at layers specified in embeddings\_layer\_names. Numpy array (if the model has a single input) or list of Numpy arrays (if the model has multiple inputs). Learn more about embeddings.
* **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 10000, the callback will write the metrics and losses to TensorBoard every 10000 samples. Note that writing too frequently to TensorBoard can slow down your training.

[[ModelCheckPoint]](https://keras.io/callbacks/#modelcheckpoint)

tf.keras.callbacks.ModelCheckpoint(filepath,

monitor='val\_loss',

verbose=0,

save\_best\_only=False,

save\_weights\_only=False,

mode='auto',

period=1)

* **filepath**: string, path to save the model file.
* **monitor**: quantity to monitor.
* **verbose**: verbosity mode, 0 or 1.
* **save\_best\_only**: if save\_best\_only=True, the latest best model according to the quantity monitored will not be overwritten.
* **save\_weights\_only**: if True, then only the model's weights will be saved (model.save\_weights(filepath)), else the full model is saved (model.save(filepath)).
* **mode**: one of {auto, min, max}. If save\_best\_only=True, the decision to overwrite the current save file is made based on either the maximization or the minimization of the monitored quantity. For val\_acc, this should be max, for val\_loss this should be min, etc. In auto mode, the direction is automatically inferred from the name of the monitored quantity.
* **period**: Interval (number of epochs) between checkpoints.

[[learningRateSchedular]](https://keras.io/callbacks/#learningratescheduler)

keras.callbacks.LearningRateScheduler(schedule,

verbose=0)

* **schedule**: a function that takes an epoch index as input (integer, indexed from 0) and current learning rate and returns a new learning rate as output (float).
* **verbose**: int. 0: quiet, 1: update messages.