[[compile the model]](https://keras.io/models/model/#compile)

The following parts are define in “compiling session”

* Optimizer
* Loss function
* Metric: the metric to evaluate the model in tensorboard

<model>.compile(optimizer,

loss=None,

metrics=None,

loss\_weights=None,

sample\_weight\_mode=None,

weighted\_metrics=None,

target\_tensors=None)

* **optimizer**: String (name of optimizer) or optimizer instance. See optimizers.
* **loss**: String (name of objective function) or objective function. See losses. If the model has multiple outputs, you can use a different loss on each output by passing a dictionary or a list of losses. The loss value that will be minimized by the model will then be the sum of all individual losses.
* **metrics**: List of metrics to be evaluated by the model during training and testing. Typically you will use metrics=['accuracy']. To specify different metrics for different outputs of a multi-output model, you could also pass a dictionary, such as metrics={'output\_a': 'accuracy'}.
* **loss\_weights**: Optional list or dictionary specifying scalar coefficients (Python floats) to weight the loss contributions of different model outputs. The loss value that will be minimized by the model will then be the weighted sum of all individual losses, weighted by the loss\_weights coefficients. If a list, it is expected to have a 1:1 mapping to the model's outputs. If a tensor, it is expected to map output names (strings) to scalar coefficients.
* **sample\_weight\_mode**: If you need to do timestep-wise sample weighting (2D weights), set this to "temporal". None defaults to sample-wise weights (1D). If the model has multiple outputs, you can use a different sample\_weight\_mode on each output by passing a dictionary or a list of modes.
* **weighted\_metrics**: List of metrics to be evaluated and weighted by sample\_weight or class\_weight during training and testing.
* **target\_tensors**: By default, Keras will create placeholders for the model's target, which will be fed with the target data during training. If instead you would like to use your own target tensors (in turn, Keras will not expect external Numpy data for these targets at training time), you can specify them via the target\_tensors argument. It can be a single tensor (for a single-output model), a list of tensors, or a dict mapping output names to target tensors.