ModelCheckpoint

`ModelCheckpoint` class

[source]

```
tf.keras.callbacks.ModelCheckpoint(
    filepath,
    monitor="val_loss",
    verbose=0,
    save_best_only=False,
    save_weights_only=False,
    mode="auto",
    save_freq="epoch",
    options=None,
    initial_value_threshold=None,
    **kwargs
)
```

Callback to save the Keras model or model weights at some frequency.

`ModelCheckpoint` callback is used in conjunction with training using `model.fit()` to save a model or weights (in a checkpoint file) at some interval, so the model or weights can be loaded later to continue the training from the state saved.

A few options this callback provides include:

- Whether to only keep the model that has achieved the "best performance" so far, or whether to save the model at the end of every epoch regardless of performance.
- Definition of 'best'; which quantity to monitor and whether it should be maximized or minimized.
- The frequency it should save at. Currently, the callback supports saving at the end of every epoch, or after a fixed number of training batches.
- Whether only weights are saved, or the whole model is saved.

Note: If you get `WARNING:tensorflow:Can save best model only with <name> available, skipping` see the description of the `monitor` argument for details on how to get this right.

Example

```
model.compile(loss=..., optimizer=...,
              metrics=['accuracy'])

EPOCHS = 10
checkpoint_filepath = '/tmp/checkpoint'
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)

# Model weights are saved at the end of every epoch, if it's the best seen
# so far.
model.fit(epochs=EPOCHS, callbacks=[model_checkpoint_callback])

# The model weights (that are considered the best) are loaded into the model.
model.load_weights(checkpoint_filepath)
```

Arguments

- **filepath**: string or `PathLike`, path to save the model file. e.g. filepath = os.path.join(working_dir, 'ckpt', file_name). `filepath` can contain named formatting options, which will be filled the value of `epoch` and keys in `logs` (passed in `on_epoch_end`). For example: if `filepath` is `weights.{epoch:02d}-{val_loss:.2f}.hdf5`, then the model checkpoints will be saved with the epoch number and the validation loss in the filename. The directory of the filepath should not be reused by any other callbacks to avoid conflicts.

- **monitor**: The metric name to monitor. Typically the metrics are set by the `Model.compile` method. Note:
  - Prefix the name with `"val_"` to monitor validation metrics.
  - Use `"loss"` or "`val_loss`" to monitor the model's total loss.
  - If you specify metrics as strings, like `"accuracy"`, pass the same string (with or without the `"val_"` prefix).
  - If you pass `metrics.Metric` objects, `monitor` should be set to `metric.name`
  - If you're not sure about the metric names you can check the contents of the `history.history` dictionary returned by `history = model.fit()`
  - Multi-output models set additional prefixes on the metric names.
  - **verbose**: Verbosity mode, 0 or 1. Mode 0 is silent, and mode 1 displays messages when the callback takes an action.
  - **save_best_only**: if `save_best_only=True`, it only saves when the model is considered the "best" and the latest best model according to the quantity monitored will not be overwritten. If `filepath` doesn't contain formatting options like `{epoch}` then `filepath` will be overwritten by each new better model.
  - **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 mode is set to `max` if the quantities monitored are 'acc' or start with 'fmeasure' and are set to `min` for the rest of the quantities.
  - **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)`).
  - **save_freq**: `'epoch'` or integer. When using `'epoch'`, the callback saves the model after each epoch. When using integer, the callback saves the model at end of this many batches. If the `Model` is compiled with `steps_per_execution=N`, then the saving criteria will be checked every Nth batch. Note that if the saving isn't aligned to epochs, the monitored metric may potentially be less reliable (it could reflect as little as 1 batch, since the metrics get reset every epoch). Defaults to `'epoch'`.
  - **options**: Optional `tf.train.CheckpointOptions` object if `save_weights_only` is true or optional `tf.saved_model.SaveOptions` object if `save_weights_only` is false.
  - **initial_value_threshold**: Floating point initial "best" value of the metric to be monitored. Only applies if `save_best_value=True`. Only overwrites the model weights already saved if the performance of current model is better than this value.
  - ****kwargs**: Additional arguments for backwards compatibility. Possible key is `period`.
