Checkpoint

Understand how checkpoints are used to save and load TensorFlow models.

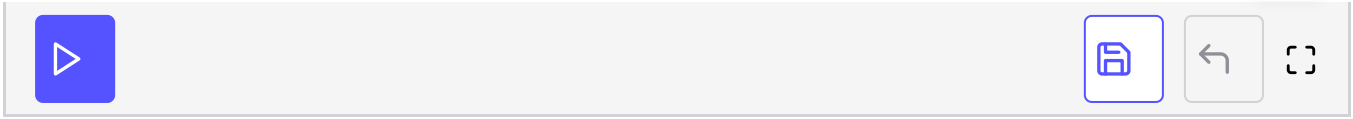
Chapter Goals:

- Understand how the checkpoint directory is structured
- Learn how to restore model parameters from a checkpoint

A. Checkpoint directory

After running training with a checkpoint directory, it will contain several files. An example checkpoint directory, named `my_model`, is shown below.

```
ls my_model
```



The `.pbtxt` file represents the entire computation graph stored in human readable text format. The `.tfevents` file is the events file for TensorBoard (note that the longer file suffix, which contains the local machine's ID, is omitted).

The actual saved model state at a particular training step consists of the following three files:

- `.data`: One or more files containing the values for the model's parameters. Larger models may require more `.data` files.
- `.index`: Metadata descriptions for how to find a particular tensor in the `.data` file(s).
- `.meta`: Represents the non-human readable saved graph structure. This file can be used to restore the computation graph.

The `checkpoint` file lists which checkpoint to use when restoring parameters, as well as all the possible checkpoints available.

```
cat my_model/checkpoint
```

Contents of the checkpoint file. The first line specifies the checkpoint to use, while the remaining lines list all the available checkpoints.

B. Saving parameters

The traditional method to save and restore parameters in TensorFlow is to use the `tf.train.Saver` object.

To save the parameters of a given TensorFlow session, use `tf.train.Saver.save`. This function has two required arguments: the current session and the path to which the file will be saved.

It has many keyword arguments, one of which is `global_step`, which determines the number tacked on to the back of the file name.

```
# sess is a tf.Session object
# 'my-model' is the filepath
# global_step
saver.save(sess, 'my-model', global_step=1000)
# checkpoint filename will be 'my-model-1000'
# the file will be in the current working directory
```

C. Restoring parameters

When we use `MonitoredTrainingSession` to resume training, it automatically restores parameters from the `model_checkpoint_path` specified in the `checkpoint` file. However, if we want to evaluate the model or use it for predictions, we need another way to restore the parameters.

The `tf.train.Saver.restore` function is the way to do that.

```
logits = tf.layers.dense(inputs, 1)
saver = tf.train.Saver()
ckpt = tf.train.get_checkpoint_state('my_model')
if ckpt is not None:  # Check if has checkpoint file
  sess = tf.Session()
  saver.restore(sess, ckpt.model_checkpoint_path)
  sess.run(logits)
```

In the example above, we obtain the checkpoint state of `my_model` with the

`tf.train.get_checkpoint_state` function. The function returns a `CheckpointState` object which contains the properties `model_checkpoint_path` and `all_checkpoint_paths`. The former represents the checkpoint to use, while the latter represents the list of all checkpoints. Note that if the checkpoint file is not present in the checkpoint directory, the function returns `None`.

The `Saver` object contains the `restore` function, which restores the checkpoint from the path given in the second argument. The first argument is a `tf.Session` object that we use to execute the restoration.

We can use the `save` function to save the computation graph's parameters. It takes in the same two required arguments as `restore`, and saves the parameters to the directory passed in as the second argument.