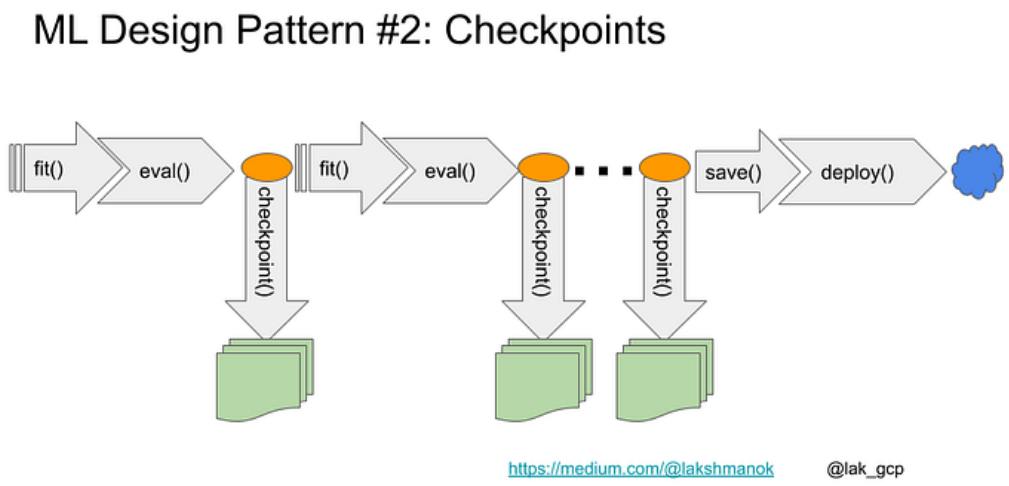
CHECKPOINTS IN MACHINE LEARNING

→ A checkpoint is an intermediate dump of a model's entire internal state (its weights, current learning rate, etc.) so that the framework can resume the training from this point whenever desired, In other words, you train for a few iterations, then evaluate the model, checkpoint it, then fit some more. When you are done, save the model and deploy it as normal



⇒ Saving intermediate checkpoints gives you a few benefits:

- **Resilience**: If you are training for a very long time, or doing distributed training on many machines, the likelihood of machine failure increases. If a machine fails, TensorFlow can resume from the last saved checkpoint instead of having to start from scratch. This behavior is automatic — TensorFlow looks for checkpoints and resumes from the last checkpoint.
- **Generalization**: In general, the longer you train, the lower the loss on the training dataset. However, at some point, the error on the held-out, evaluation dataset might stop decreasing. If you have a very large model, and you are not doing sufficient regularization, the error on the evaluation dataset might even start to increase. If this happens, it can be helpful to go back and export the model that had the best validation error. This is also called early stopping because

you could stop if you see the validation error start to increase. (A better idea is, of course, to decrease model complexity or increase the regularization so that this scenario doesn't happen). The only way you can go back to the best validation error or do early stopping is if you have been periodically evaluating and checkpointing the model.

- **Tunability**: In a well-behaved training loop, gradient descent behaves such that you get to the neighborhood of the optimal error quickly on the basis of the majority of your data and then slowly converge towards the lowest error by optimizing on the corner cases. Now, imagine that you need to periodically retrain the model on fresh data.
- CODE:

```
trainds = load_dataset('taxi-train*', TRAIN_BATCH_SIZE,
tf.estimator.ModeKeys.TRAIN)
evalds = load_dataset('taxi-valid*', 1000,
tf.estimator.ModeKeys.EVAL).take(NUM_EVAL_EXAMPLES)

steps_per_epoch = NUM_TRAIN_EXAMPLES // (TRAIN_BATCH_SIZE *
NUM_EVALS)

shutil.rmtree('{}/checkpoints/'.format(OUTDIR), ignore_errors=True)
checkpoint_path = '{}/checkpoints/taxi'.format(OUTDIR)
cp_callback = tf.keras.callbacks.ModelCheckpoint(checkpoint_path,

save_weights_only=True,
                                                 verbose=1)

history = model.fit(trainds,
                    validation_data=evalds,
                    epochs=NUM_EVALS,
                    steps_per_epoch=steps_per_epoch,
                    verbose=2, # 0=silent, 1=progress bar, 2=one line
per epoch
                    callbacks=[cp_callback])
```