Machine Learning for IoT

Tensorboard

(Some) Advanced Keras Features

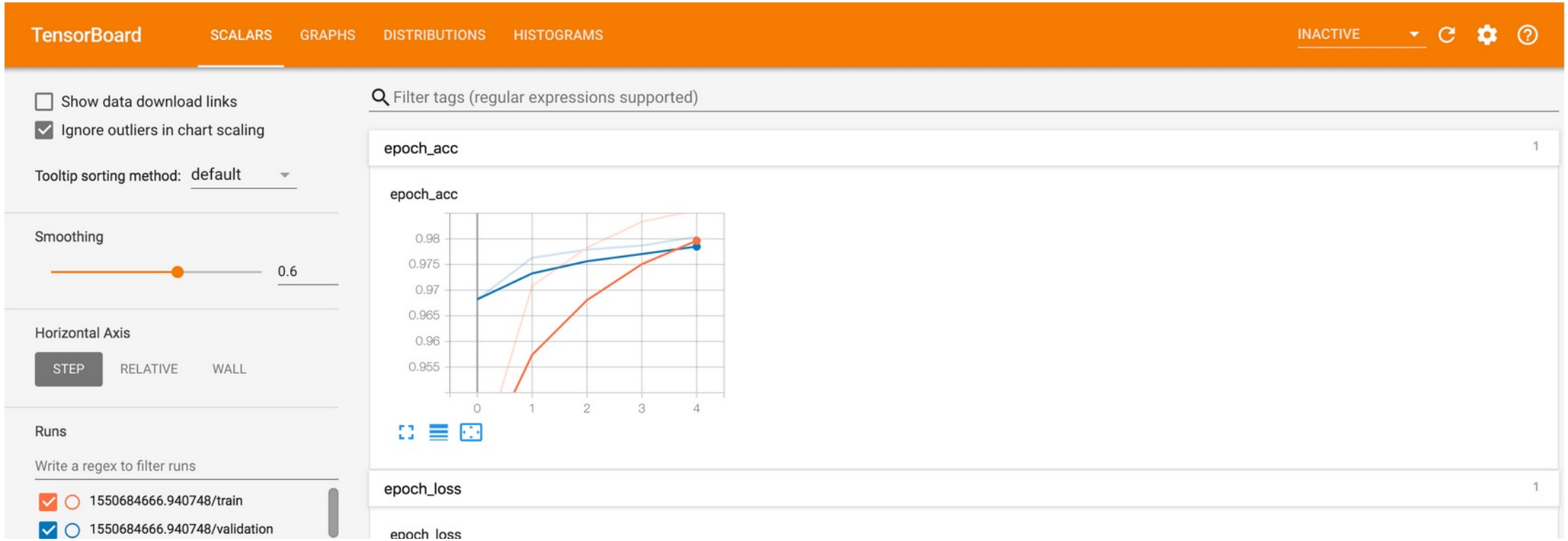
1. Saving Models
2. Advanced options for `compile()`:
   - Custom metrics and losses
3. Advanced options for `fit()`:
   - Creating/passing validation sets
   - Class and sample weights
   - Using callbacks
4. Tensorboard

TensorBoard

- TensorBoard is an analysis and visualization tool often used together with TF and Keras (but not only).
- It allows you to:
  - Track the loss and metrics during training
  - View the model computation graph
  - Etc.

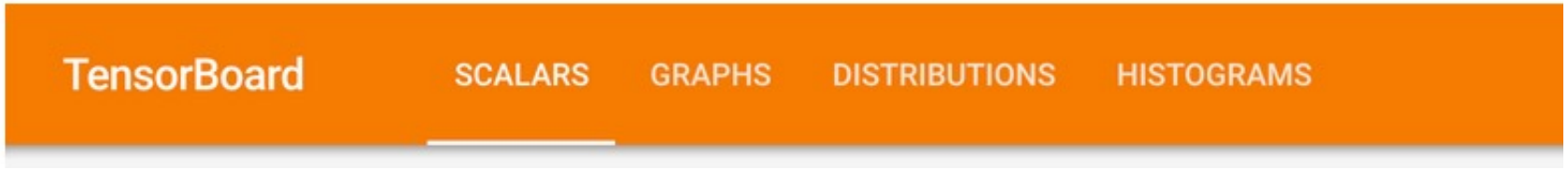
TensorBoard

- The TensorBoard Dashboard



TensorBoard

- The **Scalars** dashboard track loss and metrics in every epoch. You can use it to also track training speed, learning rate, and other scalar values.
- The **Graphs** dashboard helps you visualize your model.
- The **Distributions** and **Histograms** dashboards show the distribution of a Tensor over time. This can be useful to visualize weights and biases and verify that they are changing in an expected way.



- You can also enable TensorBoard to log images, embeddings, etc.

TensorBoard

- You can enable TensorBoard logging in Python using a dedicated Callback

```
tf.keras.callbacks.TensorBoard(
    log_dir='logs',
    histogram_freq=0,
    write_graph=True,
    write_images=False,
    update_freq='epoch',
    …
)
```

Destination

How frequently (epochs) to compute weight histograms (never if 0)

Save the graph in TB

Save model weights as images in TB

'batch', 'epoch' or integer number of batches

TensorBoard

- Then after starting the training (also while it is running) you can check the results in Tensorboard by starting it from the command line:

```
tensorboard --logdir <path specified in the callback>
```

- Or from a Jupyter Notebook (same command with a % at the beginning):

- Note that it is convenient to use a single logdir with sub-directories for different training runs.

TensorBoard

- You can also manually log scalars in TB format like this:

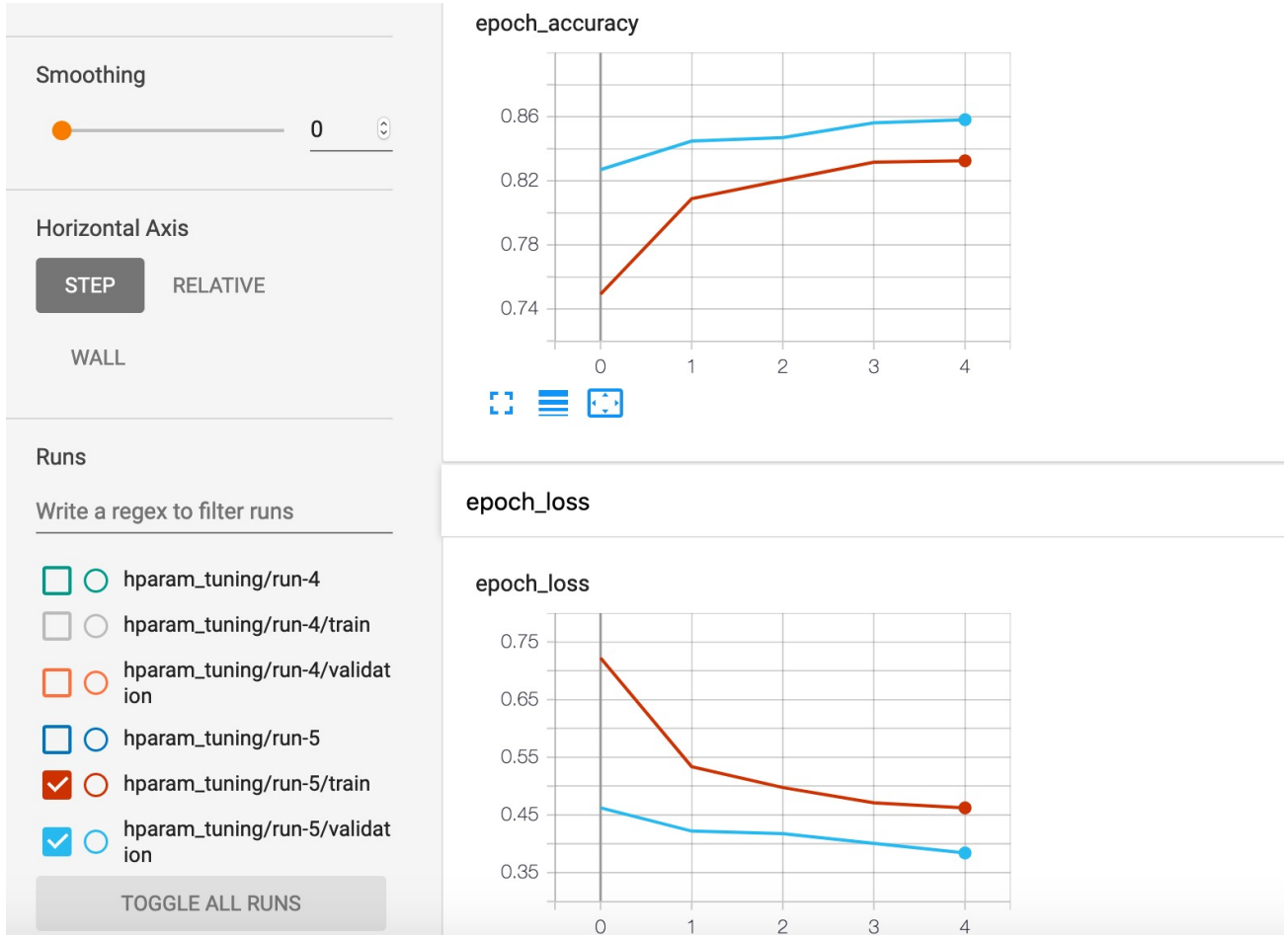
```
train_writer= tf.summary.create_file_writer(log_dir + "/train")
```

Use separate train/val/test subfolders for better visualization

```
# ... Some code (e.g. custom training loop...)
with train_summary_writer.as_default():
    tf.summary.scalar('loss', train_loss, step=epoch)
```

TensorBoard

- Tensorboard Scalar Logs



TensorBoard

- **Notebook:** Tensorboard_Scalars

TensorBoard (Hyper-parameters Search)

- TensorBoard is helpful for hyper-parameters search:
  - This is one of the most important steps in deep learning models training.
  - Although there are tools for doing this automatically using various optimization algorithms (Neural Architecture Search/AutoML), in some cases it is still done manually due to resource limitations.

- TensorBoard can visualize the results of multiple training runs with different hyper-parameters in a convenient way, so to guide designers in their next choices.

TensorBoard (Hyper-parameters Search)

1. Import hyper-parameters API:

```
from tensorboard.plugins.hparams import api as hp
```

2. Create a dictionary of hyper-parameters:

```
hparams = {
            'num_units': num_units,
            'dropout': dropout_rate,
          }
```

3. Create a Model() using those hparams.

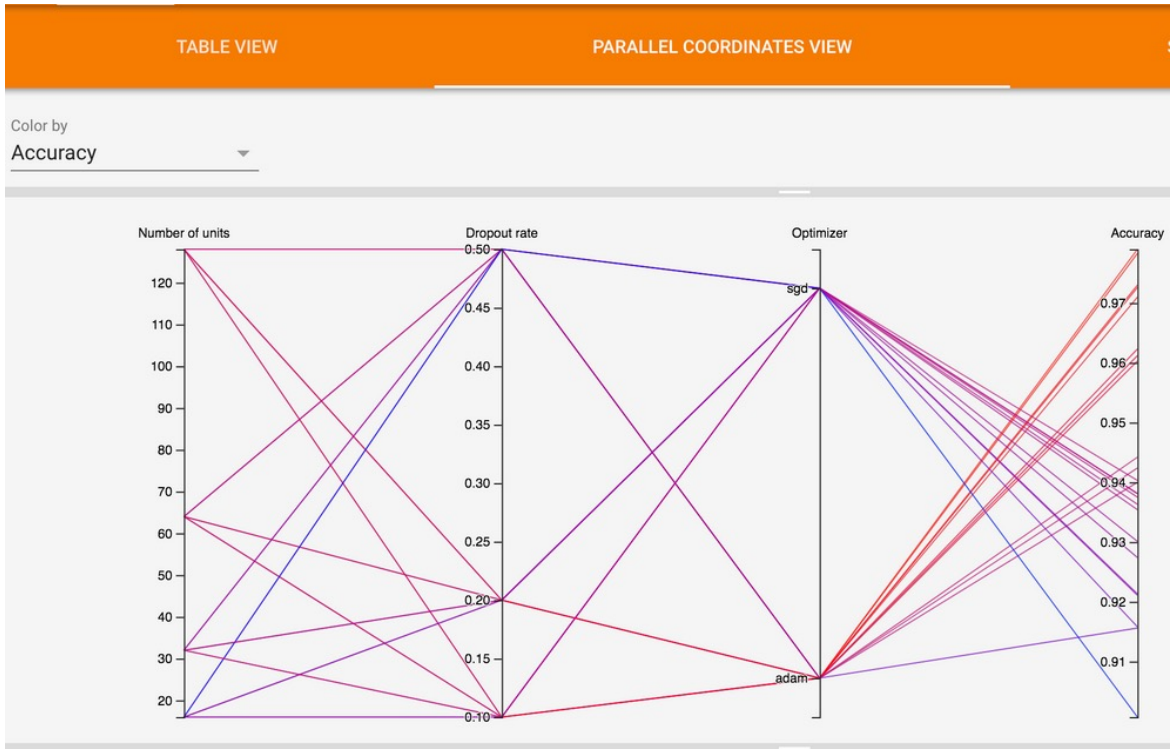
TensorBoard (Hyper-parameters Search)

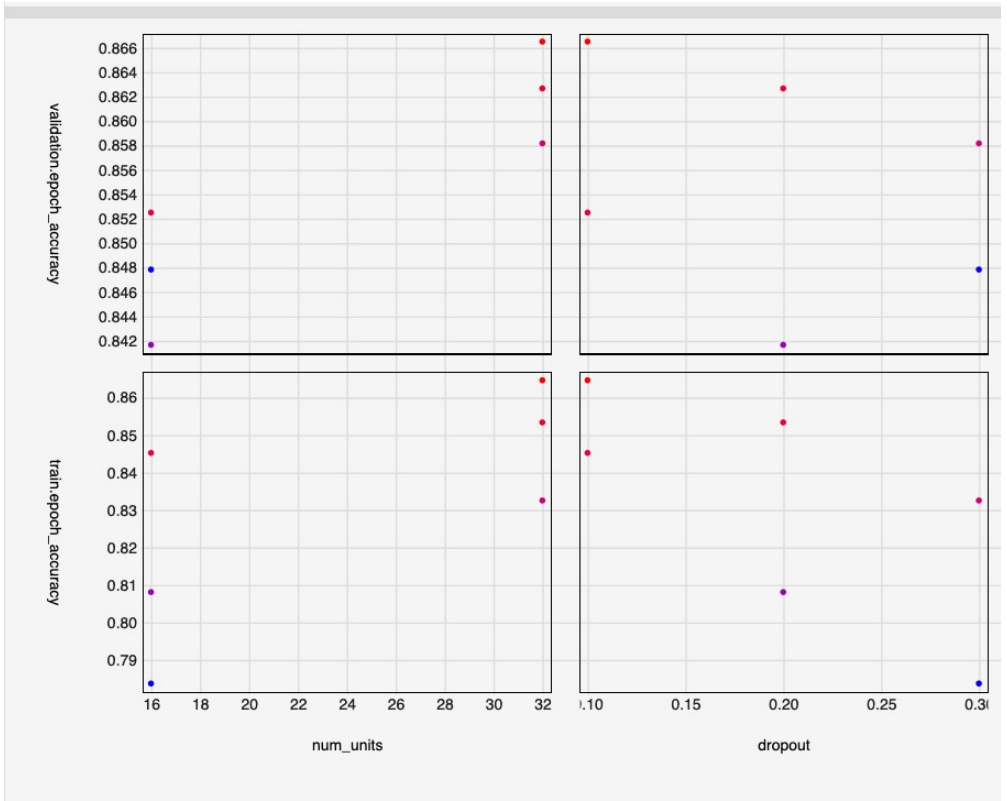
4. Use a pre-defined callback to log the training metrics associated with the current hyper-parameters, and pass it to `fit()`

```
hp_callback = hp.KerasCallback(log_dir, hparams)
model.fit(
    x_train,
    y_train,
    validation_split=0.1,
    epochs=5,
    callbacks=[tb_callback, hp_callback]
)
```

TensorBoard (Hyper-parameters Search)

- See the search results in TensorBoard with different views





TensorBoard (Hyper-parameters Search)

- **Notebook**: TensorBoard_Hyperparameters

TensorBoard (Profiling)

- Another useful functionality of TensorBoard is the ability to profile the execution of a model (e.g. during training) to identify and remove performance bottlenecks.

- This is done through `tensorboard_plugin_profile` (should be pre-installed on newest TF/TB versions)

TensorBoard (Profiling)

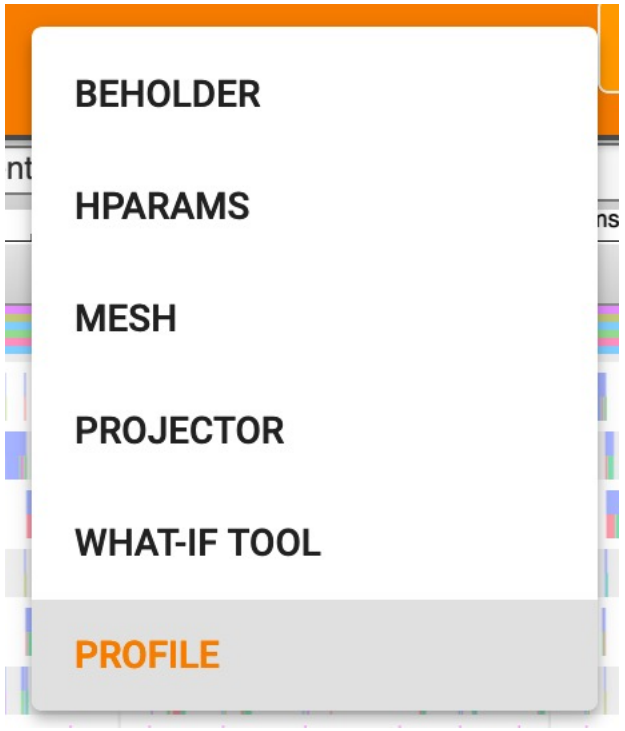
- You can enable profiling of training steps adding the `profile_batch` option to the TensorBoard callback.

```
tb_callback = tf.keras.callbacks.TensorBoard(
    log_dir = './tb_logs',
    histogram_freq = 1,
    profile_batch = '500,520'
)
```

Profile batches from 500 to 520.

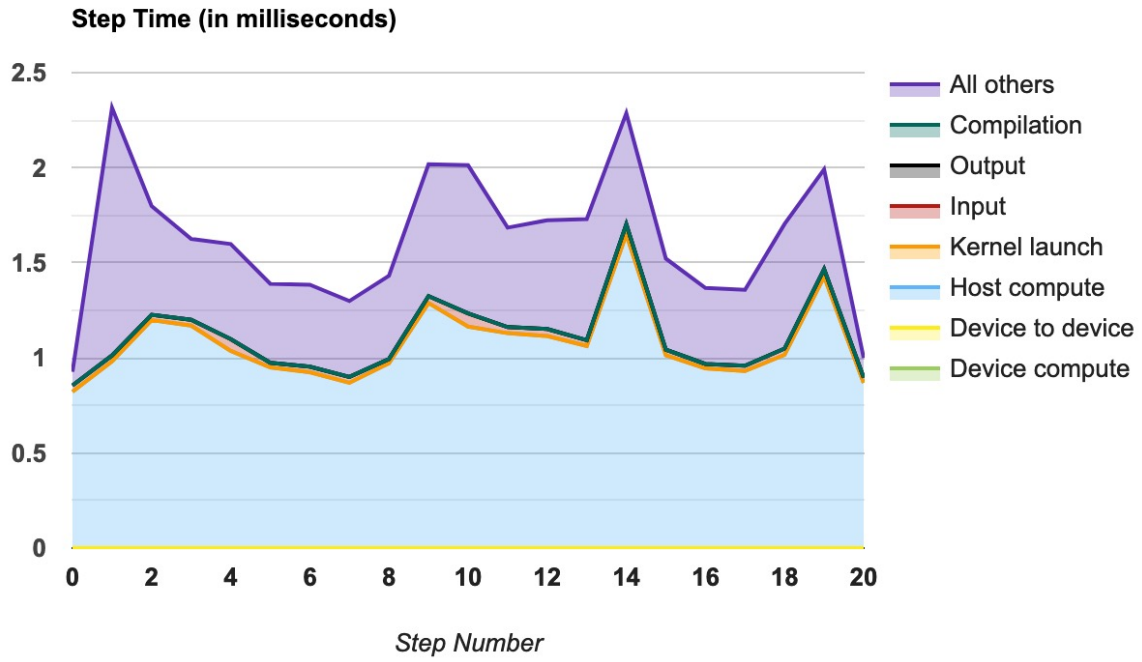
TensorBoard (Profiling)

- After training, result appear in the “Profile” tab of TensorBoard (find it in the drop-down menu if not already visible).



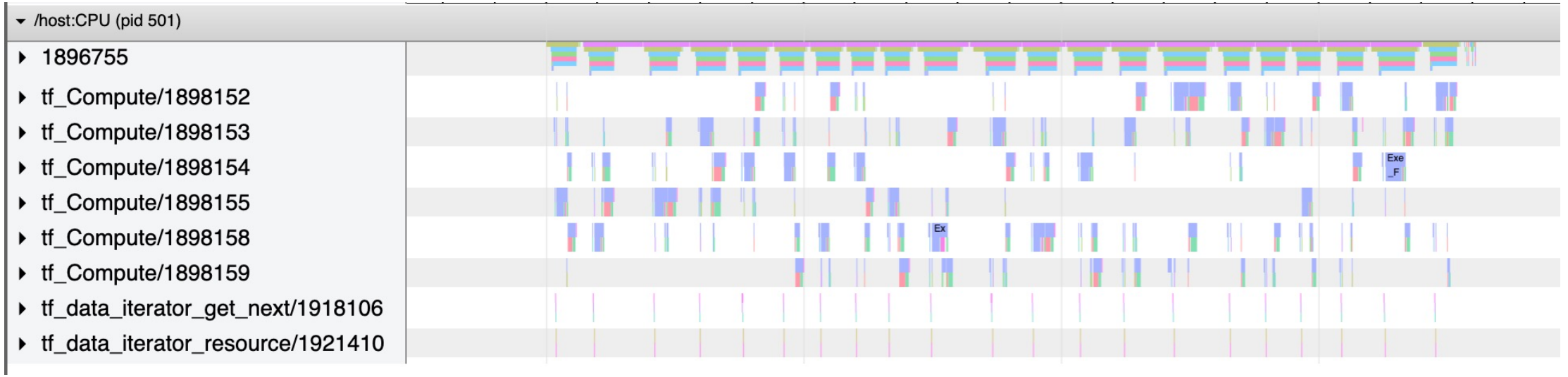
TensorBoard (Profiling)

- The “Overview Page”. shows a breakdown of the time taken by various categories of operations in the profiled batches (20 in this example).
  - It also provides useful textual tips on how to improve the model performance



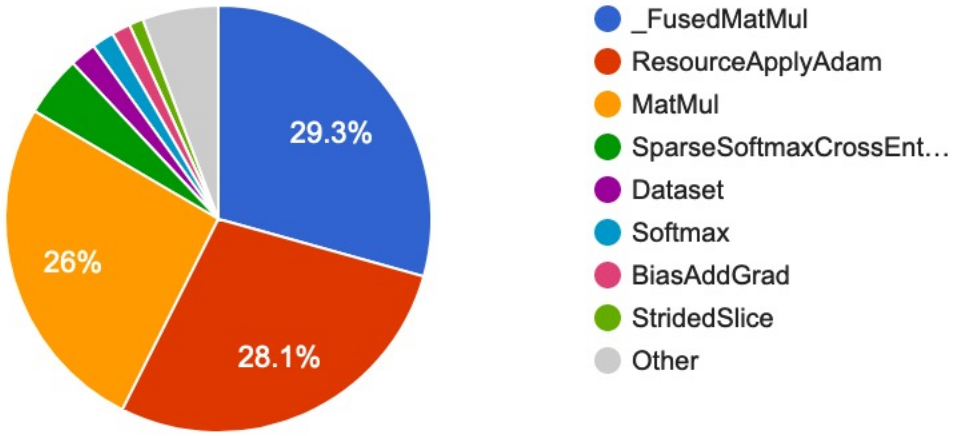
TensorBoard (Profiling)

- The “Trace View” pane shows the trace of individual events with the corresponding execution time, grouped by type



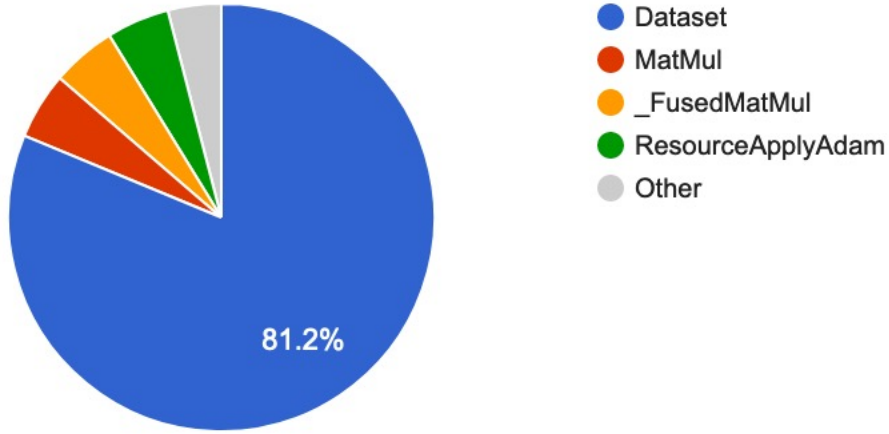
TensorBoard (Profiling)

- The “TensorFlow Stats” pane shows summary statistics.



Healthy Profile

Troublesome Profile
(input bound)



TensorBoard (Profiling)

- **Notebook**: TensorBoard_Profiling