Using TensorBoard in Notebooks

[View on TensorFlow...] [Run in Google Col...] [View source on GitH...] [Download notebo...]

TensorBoard can be used directly within notebook experiences such as Colab and Jupyter. This can be helpful for sharing results, integrating TensorBoard into existing workflows, and using TensorBoard without installing anything locally.

Setup

Start by installing TF 2.0 and loading the TensorBoard notebook extension:

For Jupyter users: If you've installed Jupyter and TensorBoard into the same virtualenv, then you should be good to go. If you're using a more complicated setup, like a global Jupyter installation and kernels for different Conda/virtualenv environments, then you must ensure that the `tensorboard` binary is on your `PATH` inside the Jupyter notebook context. One way to do this is to modify the `kernel_spec` to prepend the environment's `bin` directory to `PATH`, as described here.

For Docker users: In case you are running a Docker image of Jupyter Notebook server using TensorFlow's nightly, it is necessary to expose not only the notebook's port, but the TensorBoard's port. Thus, run the container with the following command:

```
docker run -it -p 8888:8888 -p 6006:6006 \
tensorflow/tensorflow:nightly-py3-jupyter
```

where the `-p 6006` is the default port of TensorBoard. This will allocate a port for you to run one TensorBoard instance. To have concurrent instances, it is necessary to allocate more ports. Also, pass `--bind_all` to `%tensorboard` to expose the port outside the container.

```
# Load the TensorBoard notebook extension
%load_ext tensorboard
```

Import TensorFlow, datetime, and os:

```
import tensorflow as tf
import datetime, os
```

TensorBoard in notebooks

Download the FashionMNIST dataset and scale it:

```
fashion_mnist = tf.keras.datasets.fashion_mnist

(x_train, y_train),(x_test, y_test) = fashion_mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-dataset
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-dataset
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-dataset
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-dataset
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

Create a very simple model:

```
def create_model():
  return tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28), name='layers_flatten'),
    tf.keras.layers.Dense(512, activation='relu', name='layers_dense'),
    tf.keras.layers.Dropout(0.2, name='layers_dropout'),
    tf.keras.layers.Dense(10, activation='softmax', name='layers_dense_2')
  ])
```

Train the model using Keras and the TensorBoard callback:

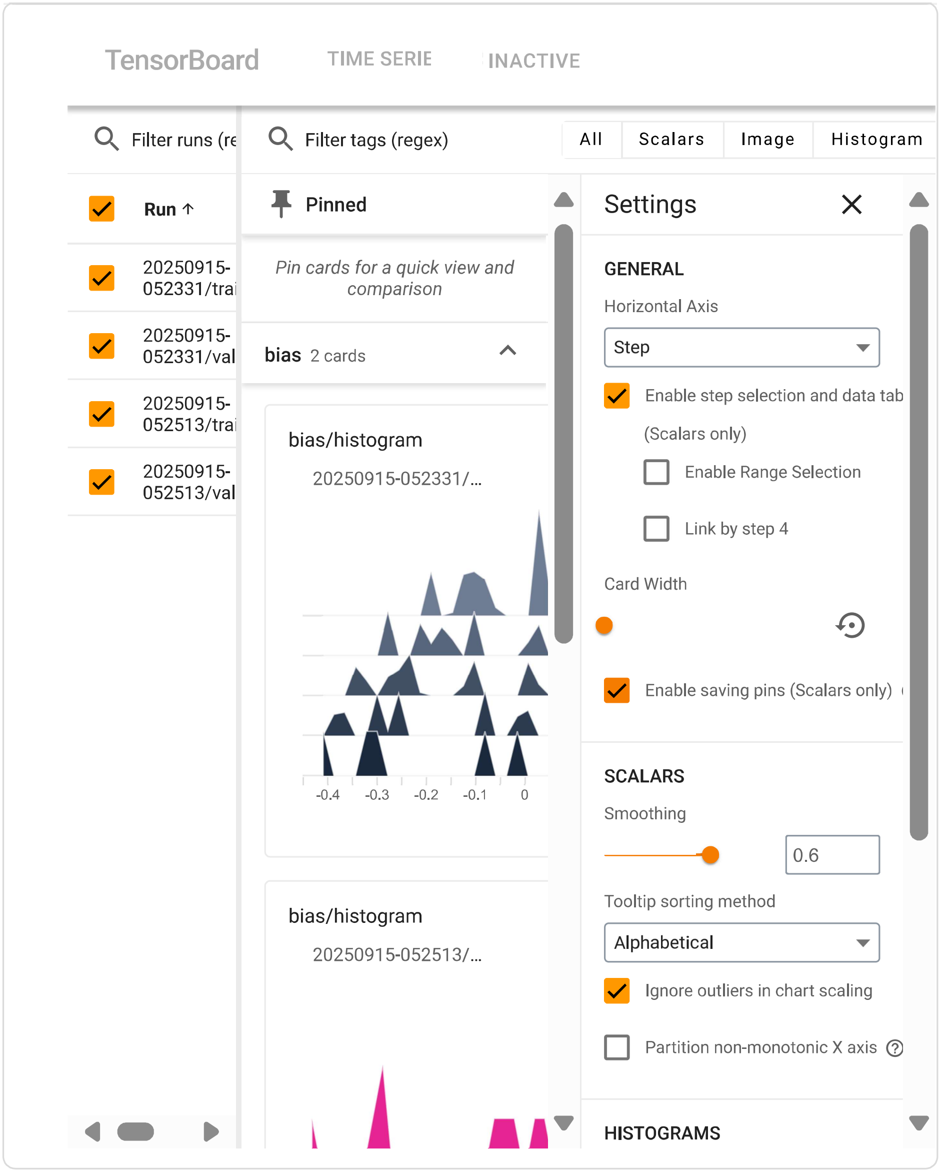
```
def train_model():

  model = create_model()
  model.compile(optimizer='adam',
                loss='sparse_categorical_crossentropy',
                metrics=['accuracy'])

  logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S
  tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=

  model.fit(x=x_train,
            y=y_train,
            epochs=5,
            validation_data=(x_test, y_test),
            callbacks=[tensorboard_callback])
```

```
train_model()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37
  super().__init__(**kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 14s 7ms/step - accuracy: 0.7833 - loss: 0.6109 -
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 19s 6ms/step - accuracy: 0.8570 - loss: 0.3917 -
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - accuracy: 0.8678 - loss: 0.3526 -
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 6ms/step - accuracy: 0.8767 - loss: 0.3333 -
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 6ms/step - accuracy: 0.8831 - loss: 0.3124 -
```

Start TensorBoard within the notebook using magics:

```
%tensorboard --logdir logs
```



	Run ↑
✓	20250915-052331/trai
✓	20250915-052331/val
✓	20250915-052513/trai
✓	20250915-052513/val

Settings

GENERAL

Horizontal Axis: Step

✓ Enable step selection and data tab (Scalars only)

☐ Enable Range Selection

☐ Link by step 4

Card Width

✓ Enable saving pins (Scalars only)

SCALARS

Smoothing: 0.6

Tooltip sorting method: Alphabetical

✓ Ignore outliers in chart scaling

☐ Partition non-monotonic X axis

HISTOGRAMS

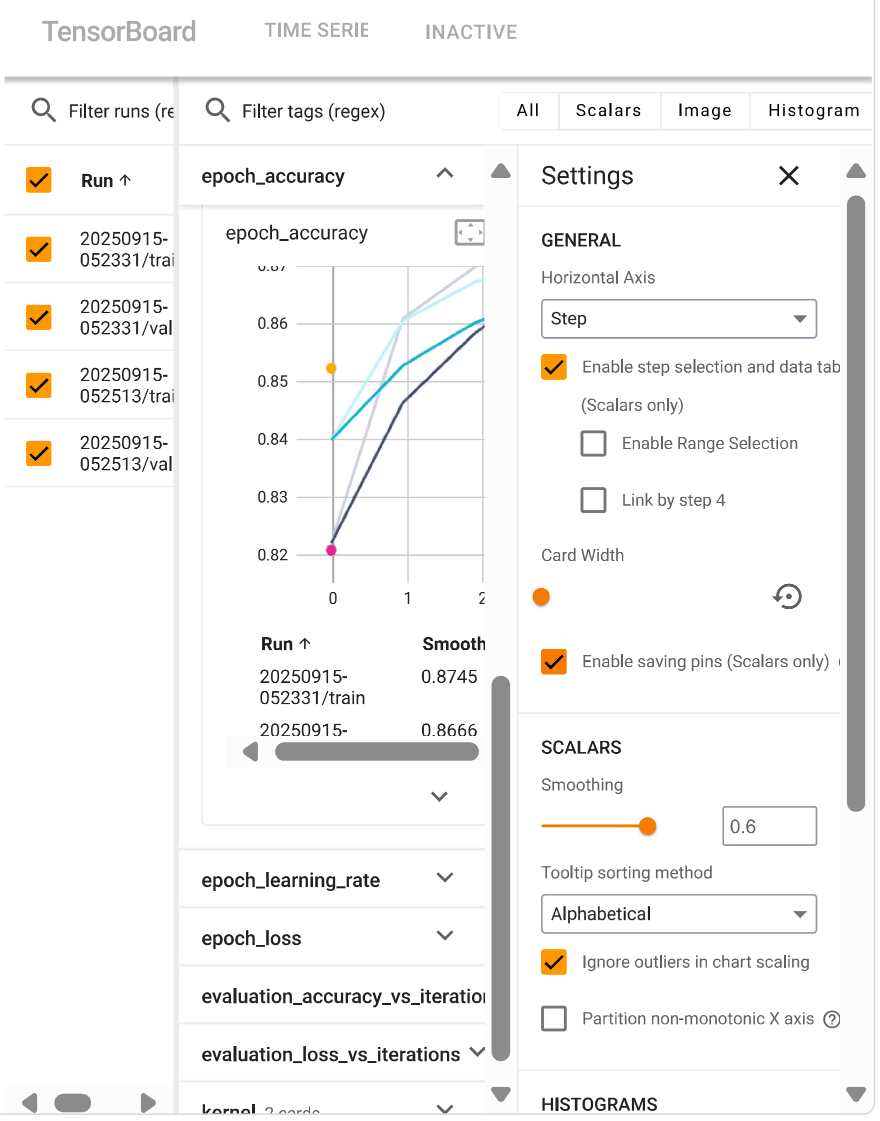
You can now view dashboards such as **Time Series**, **Graphs**, **Distributions**, and others. Some dashboards are not available yet in Colab (such as the profile plugin).

The `%tensorboard` magic has exactly the same format as the TensorBoard command line invocation, but with a `%`-sign in front of it.

You can also start TensorBoard before training to monitor it in progress:

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 1045), started 0:00:01 ago. (Use '!kill
1045' to kill it.)
```



Run ↑
- ✓ 20250915-052331/trai
- ✓ 20250915-052331/val
- ✓ 20250915-052513/trai
- ✓ 20250915-052513/val

Run ↑	Smooth
20250915-052331/train	0.8745
20250915-	0.8666

Settings

GENERAL

Horizontal Axis: Step

- [✓] Enable step selection and data tab (Scalars only)
- [ ] Enable Range Selection
- [ ] Link by step 4

Card Width

- [✓] Enable saving pins (Scalars only)

SCALARS

Smoothing: 0.6

Tooltip sorting method: Alphabetical

- [✓] Ignore outliers in chart scaling
- [ ] Partition non-monotonic X axis

HISTOGRAMS

The same TensorBoard backend is reused by issuing the same command. If a different logs directory was chosen, a new instance of TensorBoard would be opened. Ports are managed automatically.

Start training a new model and watch TensorBoard update automatically every 30 seconds or refresh it with the button on the top right:

```
train_model()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37
  super().__init__(**kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 13s 6ms/step - accuracy: 0.7795 - loss: 0.6168 -
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 20s 6ms/step - accuracy: 0.8578 - loss: 0.3876 -
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.8687 - loss: 0.3601 -
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.8783 - loss: 0.3310 -
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 21s 7ms/step - accuracy: 0.8849 - loss: 0.3126 -
```

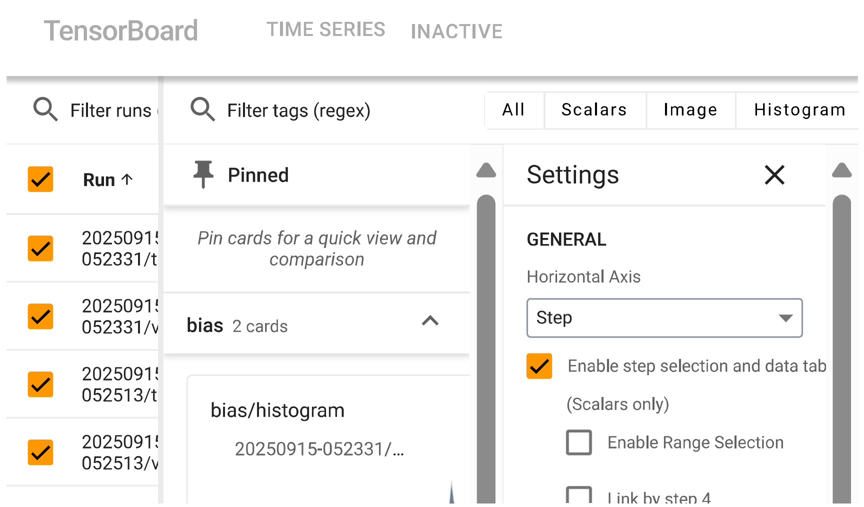
You can use the `tensorboard.notebook` APIs for a bit more control:

```
from tensorboard import notebook
notebook.list() # View open TensorBoard instances
```

```
Known TensorBoard instances:
  - port 6006: logdir logs (started 0:01:26 ago; pid 1045)
```

```
# Control TensorBoard display. If no port is provided,
# the most recently launched TensorBoard is used
notebook.display(port=6006, height=1000)
```

```
Selecting TensorBoard with logdir logs (started 0:01:26 ago; port 6006, pid 10
```


TensorBoard TIME SERIES INACTIVE

Filter runs | Filter tags (regex) | All | Scalars | Image | Histogram

Run ↑

- [✓] 20250915 052331/t
- [✓] 20250915 052331/v
- [✓] 20250915 052513/t
- [✓] 20250915 052513/v

Pinned

Pin cards for a quick view and comparison

bias 2 cards

bias/histogram

20250915-052331/...

Settings

GENERAL

Horizontal Axis: Step

[✓] Enable step selection and data tab (Scalars only)

[ ] Enable Range Selection

[ ] Link by step 4