Copyright Notice

TensorBoard SCALARS IMAGES GRAPHS DISTRIBUTIONS HISTOGRAMS

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

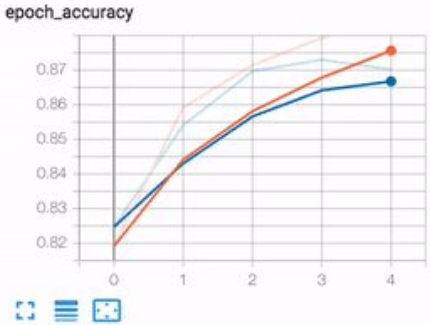
20190225-183554/train

20190225-183554/validation

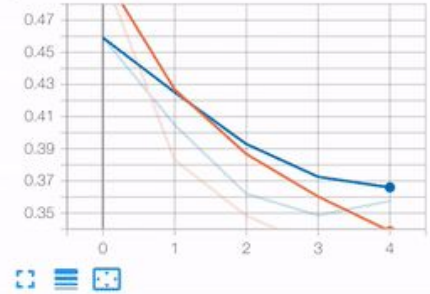
20190225-183652/data

Filter tags (regular expressions supported)

epoch_accuracy



epoch_loss



TensorBoard.dev

tensorboard.dev

TensorBoard.dev

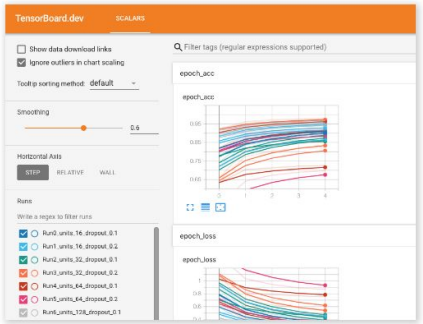
TensorBoard TensorFlow

TensorBoard.dev PREVIEW

Easily host, track, and share your ML experiments for free.

A managed TensorBoard experience that lets you upload and share your ML experiment results with anyone.

Get started | Example Colab



Why TensorBoard.dev?

Zero setup, free storage

Get started easily with no deployment. Free storage up to 10 million data points.

Familiar TensorBoard experience

The TensorFlow visualization toolkit you know and love.

*What is **TensorBoard**?*

Public, easy sharing

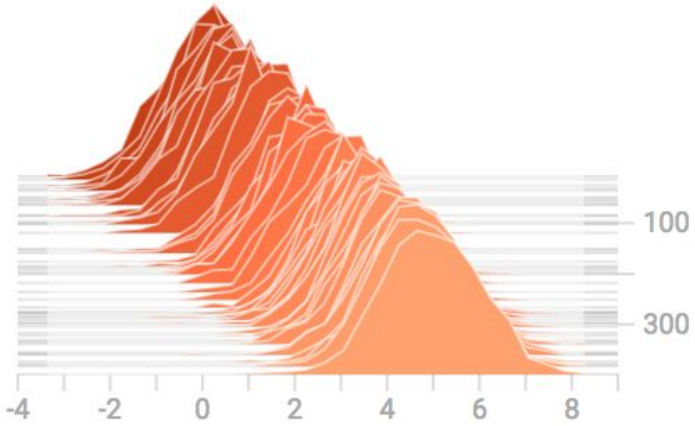
Instantly share your TensorBoard ML experiment results with anyone, for publications, troubleshooting, and team collaboration.

```
model = create_model()

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

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

tensorboard_callback =
      tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


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

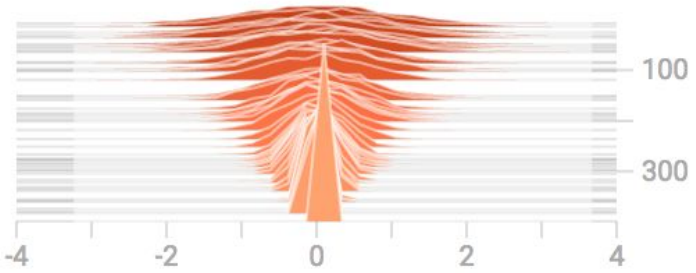
```
model = create_model()

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

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

tensorboard_callback =
      tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


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

```
model = create_model()

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

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

tensorboard_callback =
      tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


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

normal/moving_mean
.



normal/shrinking_variance
.



```
model = create_model()

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

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

tensorboard_callback =
      tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


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

```
model = create_model()

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

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

tensorboard_callback =
      tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


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


..
logs
fit
20191114-214911
train
plugins
events.out.tfevents.1573768151.df93b15b9686.127.277.v2
events.out.tfevents.1573768152.df93b15b9686.profile-empty
validation
events.out.tfevents.1573768163.df93b15b9686.127.7462.v2

```
!tensorboard dev upload --logdir ./logs
```

***** TensorBoard Uploader *****

This will upload your TensorBoard logs to https://tensorboard.dev/ from the following directory:

./logs

This TensorBoard will be visible to everyone. Do not upload sensitive data.

Your use of this service is subject to Google's Terms of Service <https://policies.google.com/terms> and Privacy Policy <https://policies.google.com/privacy>, and TensorBoard.dev's Terms of Service <https://tensorboard.dev/policy/terms/>.

This notice will not be shown again while you are logged into the uploader. To log out, run `tensorboard dev auth revoke`.

Continue? (yes/NO)

Please visit this URL to authorize this application: https://accounts.google.com/o/oauth2/auth?response_type=code&client_

Enter the authorization code:

Google

Sign in

Please copy this code, switch to your application and paste it there:

```
4/tQGjxD0nqWvvj6-L5hNGA-
L8zorPiNRdYFUfVFYHDwa-jaSVQVNRGYE
```

```
Upload started and will continue reading any new data as it's added
to the logdir. To stop uploading, press Ctrl-C.
View your TensorBoard live at: https://tensorboard.dev/experiment/so8bu67kRLm3CMa0LKM0Xg
```

TensorBoard.dev SCALARS

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

fit/20191114-214911/train

fit/20191114-214911/validation

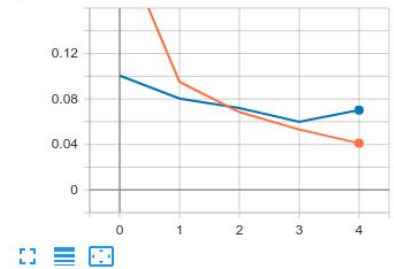
TOGGLE ALL RUNS

experiment so8bu67kRLm3CMa0LKM0Xg

Filter tags (regular expressions supported)

epoch_accuracy

epoch_loss



```
mnist = tf.keras.datasets.mnist

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

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

```
mnist = tf.keras.datasets.mnist

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

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

```
mnist = tf.keras.datasets.mnist

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

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

```
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

tensorboard_callback =
          tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


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

```
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

tensorboard_callback =
          tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


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

```
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

tensorboard_callback =
        tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


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



%tensorboard --logdir logs/fit
```

```
%tensorboard --logdir logs/fit
```

TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS PROFILE INACTIVE

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

20191114-221042/train

20191114-221042/validation

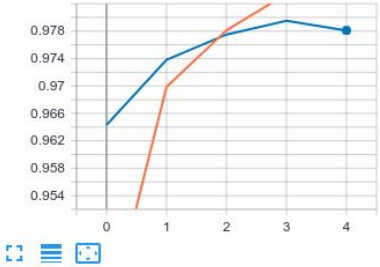
TOGGLE ALL RUNS

logs/fit

Filter tags (regular expressions supported)

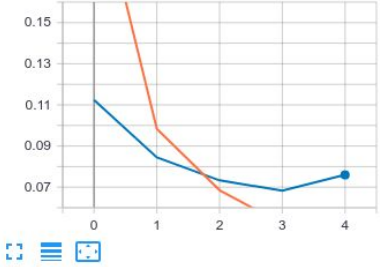
epoch_accuracy 1

epoch_accuracy



epoch_loss 1

epoch_loss



```
%tensorboard --logdir logs/fit
```

TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS PROFILE INACTIVE

Search nodes. Regexes supported.

Fit to Screen

Download PNG

Run (1) 20191114-221042/train

Tag (3) Default

Upload Choose File

- Graph
- Conceptual Graph
- Profile

Trace inputs

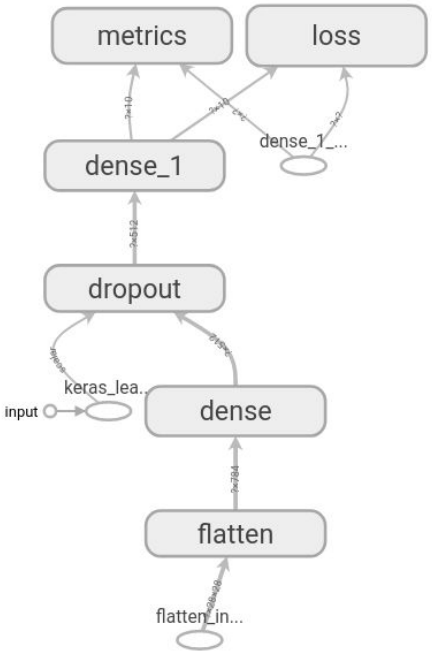
Show health pills

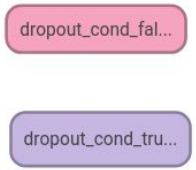
Color Structure

Close legend.

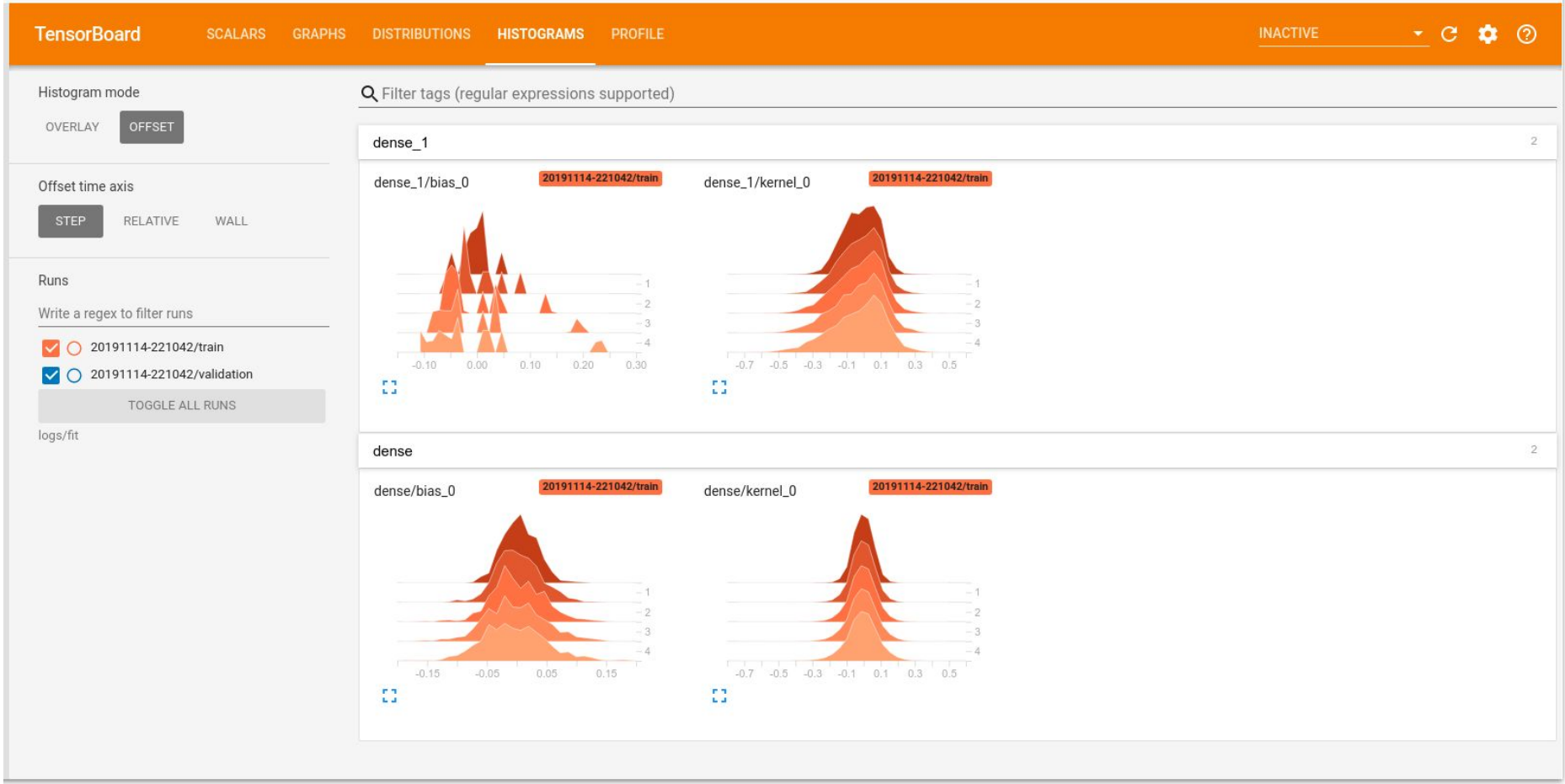
Graph (* = expandable)

- Namespace* ?
- OpNode ?
- Unconnected series* ?
- Connected series* ?
- Constant ?
- Summary ?
- Dataflow edge ?
- Control dependency edge ?
- Reference edge ?





```
%tensorboard --logdir logs/fit
```



```
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) =
    fashion_mnist.load_data()


img = np.reshape(train_images[0], (-1, 28, 28, 1))
```

```
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) =
    fashion_mnist.load_data()


img = np.reshape(train_images[0], (-1, 28, 28, 1))
```

```
# Sets up a timestamped log directory.
logdir = "logs/train_data/" +
         datetime.now().strftime("%Y%m%d-%H%M%S")

# Creates a file writer for the log directory.
file_writer = tf.summary.create_file_writer(logdir)

# Using the file writer, log the reshaped image.
with file_writer.as_default():
  tf.summary.image("Training data", img, step=0)
```

```
# Sets up a timestamped log directory.
logdir = "logs/train_data/" +
         datetime.now().strftime("%Y%m%d-%H%M%S")

# Creates a file writer for the log directory.
file_writer = tf.summary.create_file_writer(logdir)

# Using the file writer, log the reshaped image.
with file_writer.as_default():
  tf.summary.image("Training data", img, step=0)
```

```
# Sets up a timestamped log directory.
logdir = "logs/train_data/" +
         datetime.now().strftime("%Y%m%d-%H%M%S")

# Creates a file writer for the log directory.
file_writer = tf.summary.create_file_writer(logdir)

# Using the file writer, log the reshaped image.
with file_writer.as_default():
  tf.summary.image("Training data", img, step=0)
```

```
# Sets up a timestamped log directory.
logdir = "logs/train_data/" +
         datetime.now().strftime("%Y%m%d-%H%M%S")

# Creates a file writer for the log directory.
file_writer = tf.summary.create_file_writer(logdir)

# Using the file writer, log the reshaped image.
with file_writer.as_default():
  tf.summary.image("Training data", img, step=0)
```

TensorBoard IMAGES

Show actual image size

Brightness adjustment

RESET

Contrast adjustment

RESET

Runs

Write a regex to filter runs

20191116-035119

TOGGLE ALL RUNS

logs/train_data

Filter tags (regular expressions supported)

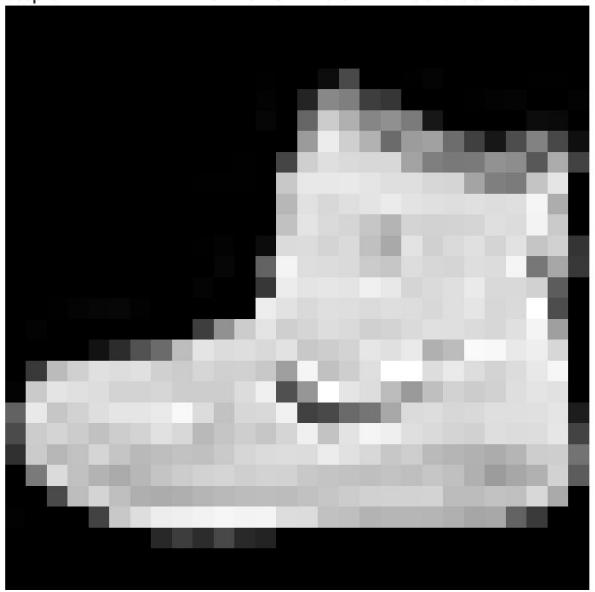
Training data

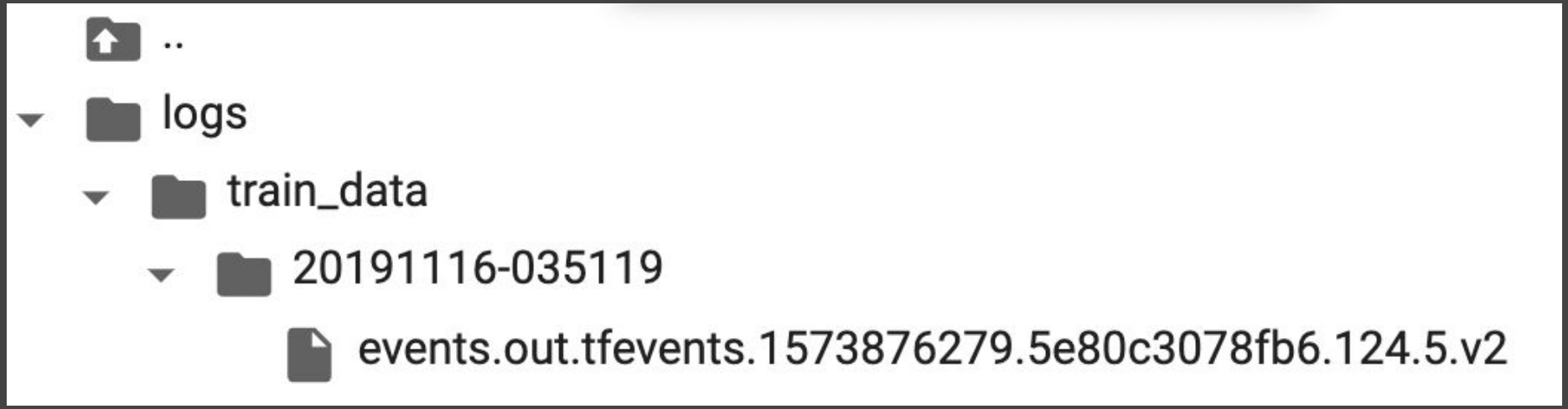
Training data

20191116-035119

tag: Training data

step **0** Fri Nov 15 2019 18:51:19 Alaska Standard Time




..
logs
train_data
20191116-035119
events.out.tfevents.1573876279.5e80c3078fb6.124.5.v2

```
%tensorboard --logdir logs/train_data
```

```
with file_writer.as_default():
  # Don't forget to reshape.
  images = np.reshape(train_images[0:25], (-1, 28, 28, 1))
  tf.summary.image("25 training data examples", images,
                   max_outputs=25, step=0)

%tensorboard --logdir logs/train_data
```

```
with file_writer.as_default():
  # Don't forget to reshape.
  images = np.reshape(train_images[0:25], (-1, 28, 28, 1))
  tf.summary.image("25 training data examples", images,
                   max_outputs=25, step=0)

%tensorboard --logdir logs/train_data
```

```
with file_writer.as_default():
  # Don't forget to reshape.
  images = np.reshape(train_images[0:25], (-1, 28, 28, 1))
  tf.summary.image("25 training data examples", images,
                   max_outputs=25, step=0)

%tensorboard --logdir logs/train_data
```

Reusing TensorBoard on port 6006 (pid 173), started 0:09:30 ago. (Use '!kill 173' to kill it.)

TensorBoard IMAGES INACTIVE

Show actual image size

Brightness adjustment RESET

Contrast adjustment RESET

Runs

Write a regex to filter runs

20191116-035119

TOGGLE ALL RUNS

logs/train_data

25 training data examples

25

PREVIOUS PAGE NEXT PAGE

25 training data examples 20191116-035119
tag: 25 training data examples
sample: 1 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 2 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 3 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 4 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 5 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 6 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 7 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 8 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 9 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 10 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 11 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time


25 training data examples 20191116-035119
tag: 25 training data examples
sample: 12 of 25
step 0Fri Nov 15 2019 18:56:57 Alaska Standard Time

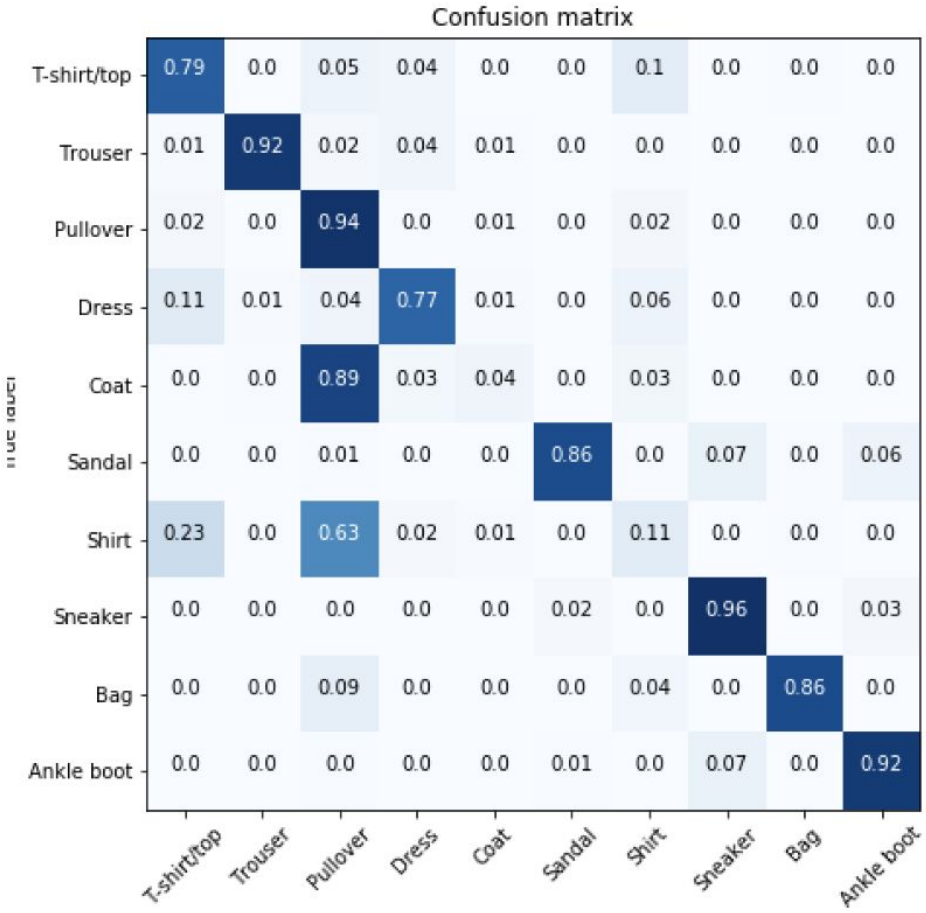

Page 1 of 3

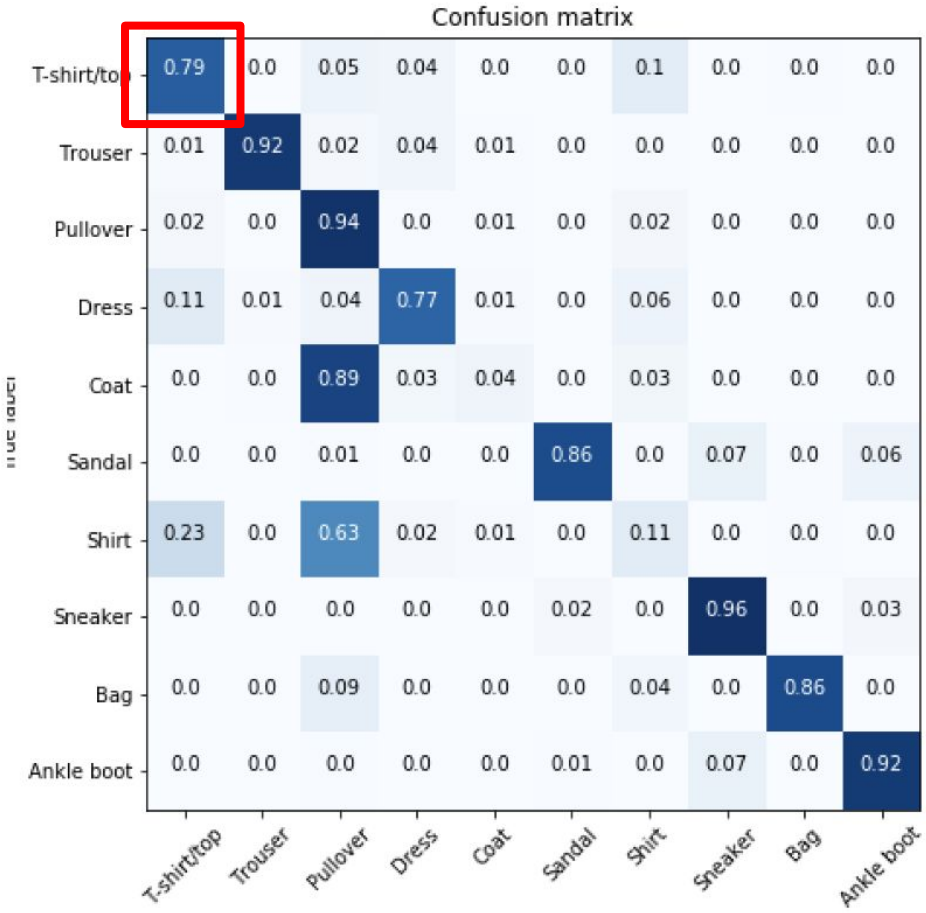
PREVIOUS PAGE NEXT PAGE

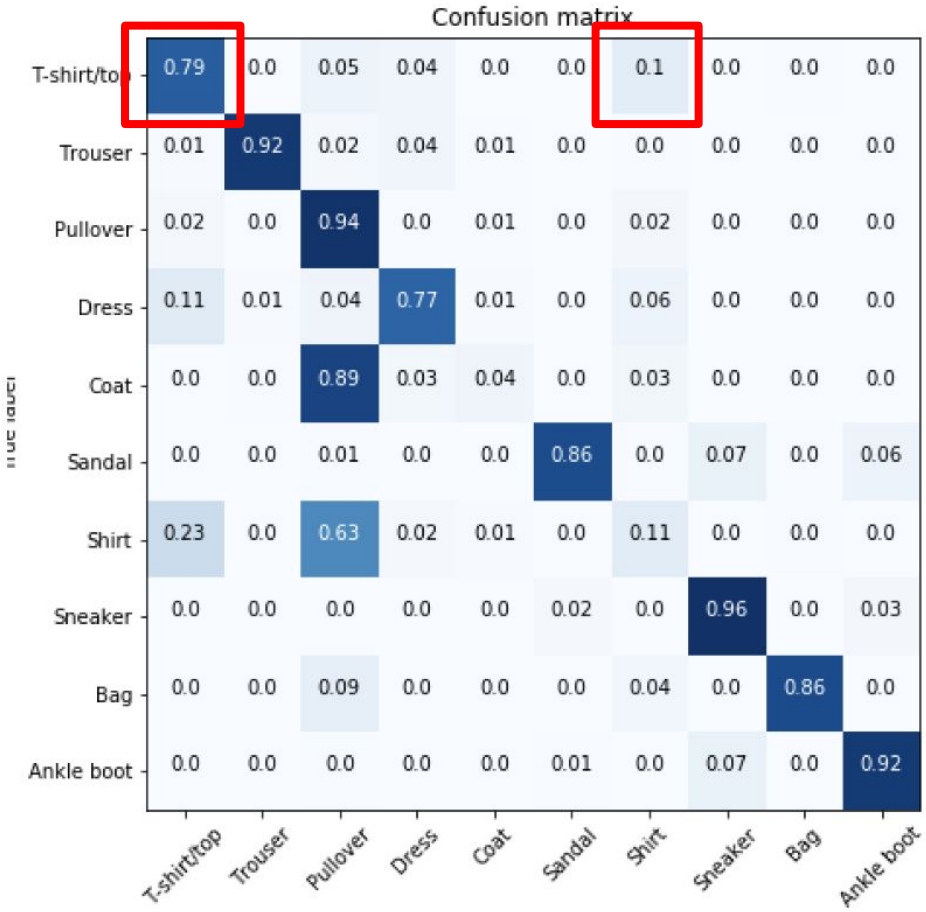
Training data

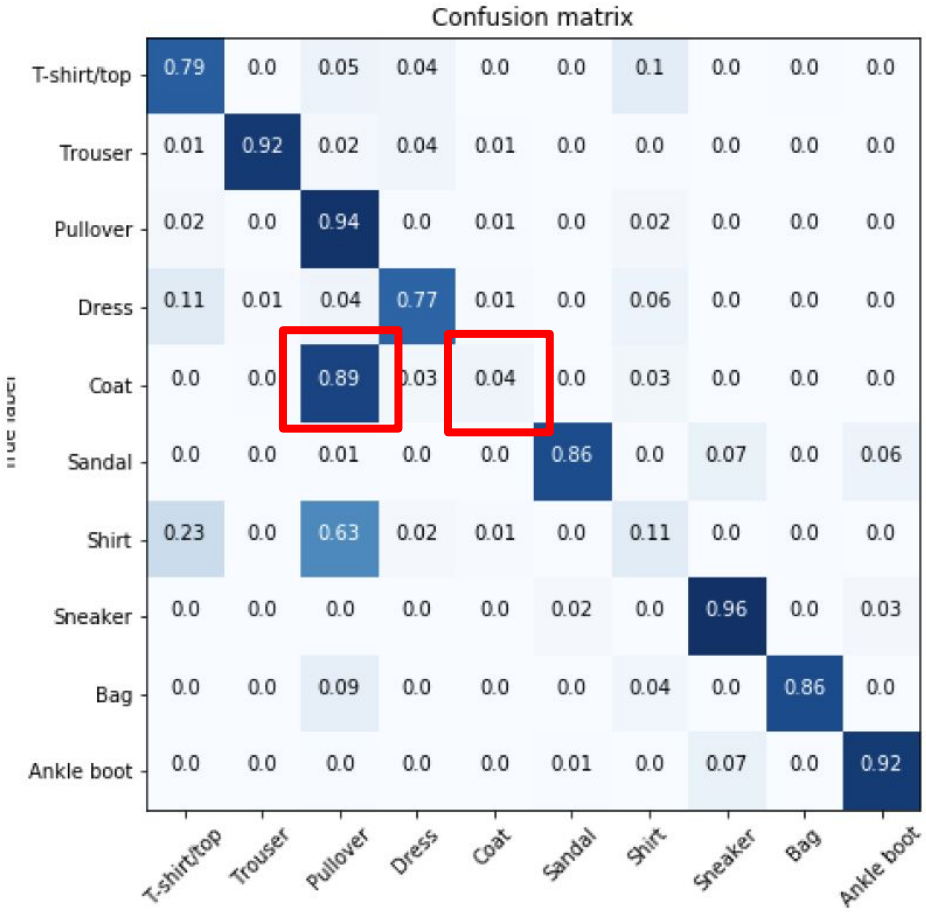
1

Training data 20191116-035119
tag: Training data
step 0Fri Nov 15 2019 18:51:19 Alaska Standard Time



Confusion matrix
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
0.79 0.0 0.05 0.04 0.0 0.0 0.1 0.0 0.0 0.0
0.01 0.92 0.02 0.04 0.01 0.0 0.0 0.0 0.0 0.0
0.02 0.0 0.94 0.0 0.01 0.0 0.02 0.0 0.0 0.0
0.11 0.01 0.04 0.77 0.01 0.0 0.06 0.0 0.0 0.0
0.0 0.0 0.89 0.03 0.04 0.0 0.03 0.0 0.0 0.0
0.0 0.0 0.01 0.0 0.0 0.86 0.0 0.07 0.0 0.06
0.23 0.0 0.63 0.02 0.01 0.0 0.11 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.02 0.0 0.96 0.0 0.03
0.0 0.0 0.09 0.0 0.0 0.0 0.04 0.0 0.86 0.0
0.0 0.0 0.0 0.0 0.0 0.01 0.0 0.07 0.0 0.92


Confusion matrix
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
0.79 0.0 0.05 0.04 0.0 0.0 0.1 0.0 0.0 0.0
0.01 0.92 0.02 0.04 0.01 0.0 0.0 0.0 0.0 0.0
0.02 0.0 0.94 0.0 0.01 0.0 0.02 0.0 0.0 0.0
0.11 0.01 0.04 0.77 0.01 0.0 0.06 0.0 0.0 0.0
0.0 0.0 0.89 0.03 0.04 0.0 0.03 0.0 0.0 0.0
0.0 0.0 0.01 0.0 0.0 0.86 0.0 0.07 0.0 0.06
0.23 0.0 0.63 0.02 0.01 0.0 0.11 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.02 0.0 0.96 0.0 0.03
0.0 0.0 0.09 0.0 0.0 0.0 0.04 0.0 0.86 0.0
0.0 0.0 0.0 0.0 0.0 0.01 0.0 0.07 0.0 0.92


Confusion matrix
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
0.79 0.0 0.05 0.04 0.0 0.0 0.1 0.0 0.0 0.0
0.01 0.92 0.02 0.04 0.01 0.0 0.0 0.0 0.0 0.0
0.02 0.0 0.94 0.0 0.01 0.0 0.02 0.0 0.0 0.0
0.11 0.01 0.04 0.77 0.01 0.0 0.06 0.0 0.0 0.0
0.0 0.0 0.89 0.03 0.04 0.0 0.03 0.0 0.0 0.0
0.0 0.0 0.01 0.0 0.0 0.86 0.0 0.07 0.0 0.06
0.23 0.0 0.63 0.02 0.01 0.0 0.11 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.02 0.0 0.96 0.0 0.03
0.0 0.0 0.09 0.0 0.0 0.0 0.04 0.0 0.86 0.0
0.0 0.0 0.0 0.0 0.0 0.01 0.0 0.07 0.0 0.92


Confusion matrix
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
0.79 0.0 0.05 0.04 0.0 0.0 0.1 0.0 0.0 0.0
0.01 0.92 0.02 0.04 0.01 0.0 0.0 0.0 0.0 0.0
0.02 0.0 0.94 0.0 0.01 0.0 0.02 0.0 0.0 0.0
0.11 0.01 0.04 0.77 0.01 0.0 0.06 0.0 0.0 0.0
0.0 0.0 0.89 0.03 0.04 0.0 0.03 0.0 0.0 0.0
0.0 0.0 0.01 0.0 0.0 0.86 0.0 0.07 0.0 0.06
0.23 0.0 0.63 0.02 0.01 0.0 0.11 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.02 0.0 0.96 0.0 0.03
0.0 0.0 0.09 0.0 0.0 0.0 0.04 0.0 0.86 0.0
0.0 0.0 0.0 0.0 0.0 0.01 0.0 0.07 0.0 0.92

```
def plot_to_image(figure):
  """Converts the matplotlib plot specified by 'figure' to a PNG image and
  returns it. The supplied figure is closed and inaccessible after this call."""
  # Save the plot to a PNG in memory.
  buf = io.BytesIO()
  plt.savefig(buf, format='png')
  # Closing the figure prevents it from being displayed directly inside
  # the notebook.
  plt.close(figure)
  buf.seek(0)
  # Convert PNG buffer to TF image
  image = tf.image.decode_png(buf.getvalue(), channels=4)
  # Add the batch dimension
  image = tf.expand_dims(image, 0)
  return image
```

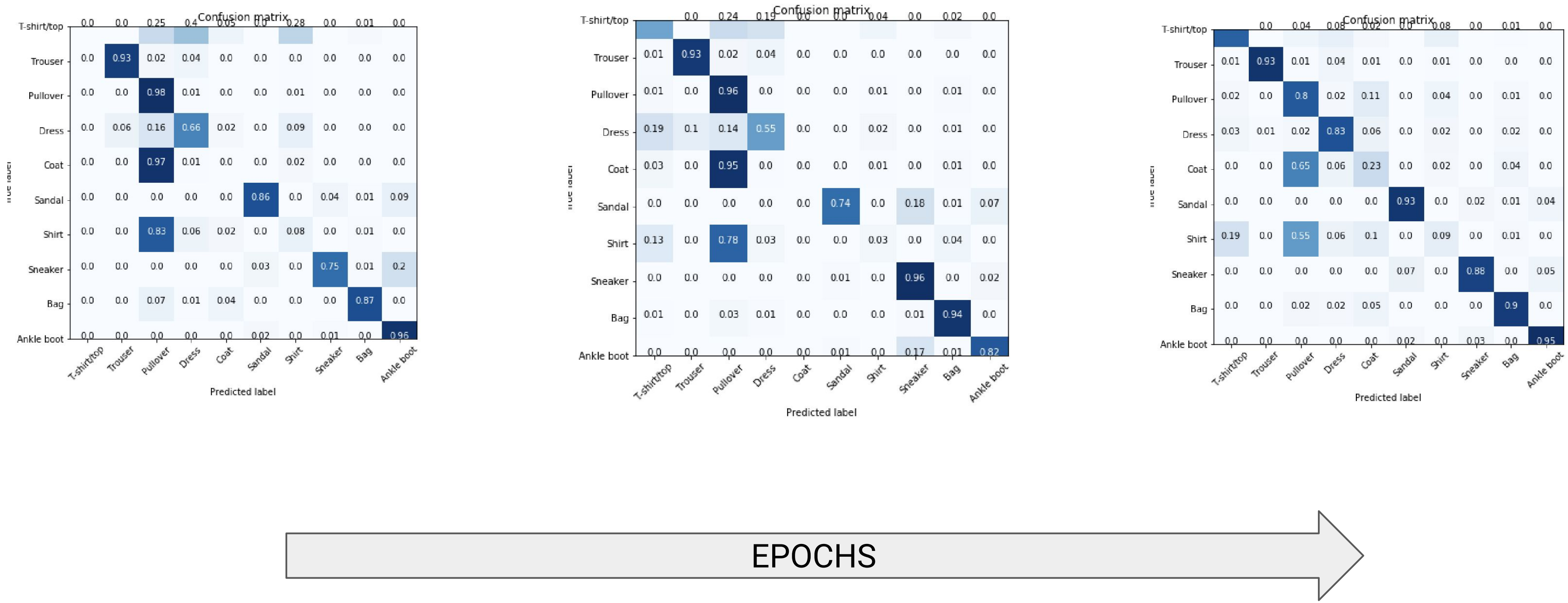
```
def plot_to_image(figure):
  """Converts the matplotlib plot specified by 'figure' to a PNG image and
  returns it. The supplied figure is closed and inaccessible after this call."""
  # Save the plot to a PNG in memory.
  buf = io.BytesIO()
  plt.savefig(buf, format='png')
  # Closing the figure prevents it from being displayed directly inside
  # the notebook.
  plt.close(figure)
  buf.seek(0)
  # Convert PNG buffer to TF image
  image = tf.image.decode_png(buf.getvalue(), channels=4)
  # Add the batch dimension
  image = tf.expand_dims(image, 0)
  return image
```

```
# Train the classifier.
model.fit(
    train_images,
    train_labels,
    epochs=5,
    verbose=0, # Suppress chatty output
    callbacks=[tensorboard_callback, cm_callback],
    validation_data=(test_images, test_labels),
)
```

```
# Train the classifier.
model.fit(
    train_images,
    train_labels,
    epochs=5,
    verbose=0, # Suppress chatty output
    callbacks=[tensorboard_callback, cm_callback],
    validation_data=(test_images, test_labels),
)
```

```
def log_confusion_matrix(epoch, logs):
  # Use the model to predict the values from the validation dataset.
  test_pred_raw = model.predict(test_images)
  test_pred = np.argmax(test_pred_raw, axis=1)

  # Calculate the confusion matrix.
  cm = sklearn.metrics.confusion_matrix(test_labels, test_pred)
  # Log the confusion matrix as an image summary.
  figure = plot_confusion_matrix(cm, class_names=class_names)
  cm_image = plot_to_image(figure)

  # Log the confusion matrix as an image summary.
  with file_writer_cm.as_default():
    tf.summary.image("Confusion Matrix", cm_image, step=epoch)


# Define the per-epoch callback.
cm_callback = keras.callbacks.LambdaCallback(on_epoch_end=log_confusion_matrix)
```

```
def log_confusion_matrix(epoch, logs):
  # Use the model to predict the values from the validation dataset.
  test_pred_raw = model.predict(test_images)
  test_pred = np.argmax(test_pred_raw, axis=1)

  # Calculate the confusion matrix.
  cm = sklearn.metrics.confusion_matrix(test_labels, test_pred)
  # Log the confusion matrix as an image summary.
  figure = plot_confusion_matrix(cm, class_names=class_names)
  cm_image = plot_to_image(figure)

  # Log the confusion matrix as an image summary.
  with file_writer_cm.as_default():
    tf.summary.image("Confusion Matrix", cm_image, step=epoch)


# Define the per-epoch callback.
cm_callback = keras.callbacks.LambdaCallback(on_epoch_end=log_confusion_matrix)
```


Confusion matrix
True label
Predicted label
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
Confusion matrix
Predicted label
Confusion matrix
Predicted label
EPOCHS

bit.ly/tensorboard-graphics