```
import tensorflow as tf
from tensorboard.plugins.hparams import api as hp
import datetime
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, lo
import numpy as np
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
```

```
# got inspiration to add the as_fram=False to the fetch_openml function (and nothing
# from https://scikit-learn.org/stable/auto_examples/linear_model/plot_sparse_logisti
# and now that it's a numpy array I can reshape X
X, y = fetch_openml('mnist_784', version=1, return_X_y=True, as_frame=False)
Y = y.astype(int)
X = (X / 255)
```

```
#inspired from a comment on an answer on https://datascience.stackexchange.com/questi
#I modified the percentages of each data sets and used a the number for random_state
#getting the test data to split off first
#Validating split will be in the fitting of the model
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=21000)
X_val, X_test, Y_val, Y_test = train_test_split(X_test,Y_test,test_size=10500)
```

```
X.shape
```

```
(70000, 784)
```

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

batch_size = 32
epochs = 5
img_rows = 28
img_cols = 28

X_train = X_train.reshape(-1, img_rows, img_cols,1)
X_test = X_test.reshape(-1, img_rows, img_cols,1)
```

```
X_val = X_val.reshape(-1, img_rows, img_cols,1)
## Create hyperparameters
HP_NUM_STEPS_PER_EPOCH = hp.HParam('steps_per_epoch',hp.Discrete([100,1000]))
HP_NUM_UNITS=hp.HParam('num_units', hp.Discrete([ 256, 512]))
# HP_DROPOUT=hp.HParam('dropout', hp.RealInterval(0.1, 0.2))
HP_LEARNING_RATE= hp.HParam('learning_rate', hp.Discrete([0.001, 0.0001]))
HP_OPTIMIZER=hp.HParam('optimizer', hp.Discrete(['adam', 'sgd', 'rmsprop']))
METRIC_ACCURACY='accuracy'
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
log_dir ='\\logs\\fit\\' + datetime.datetime.now().strftime('%Y%m%d-%H%M%S')
with tf.summary.create_file_writer(log_dir).as_default():
    hp.hparams_config(
    hparams=
    [HP_NUM_STEPS_PER_EPOCH, HP_NUM_UNITS,  HP_OPTIMIZER, HP_LEARNING_RATE],
    metrics=[hp.Metric(METRIC_ACCURACY, display_name='Accuracy')],
    )
```

```
def create_model(hparams):
    model = Sequential([
    Conv2D(64, 3, padding='same', activation='relu',input_shape=(img_rows,img_cols,1)
    MaxPooling2D(),
    #setting the Drop out value based on HParam
    Conv2D(128, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(hparams[HP_NUM_UNITS], activation='relu'),
    Dense(10, activation='softmax')])
    print(model)
    #setting the optimizer and learning rate
    optimizer = hparams[HP_OPTIMIZER]
    learning_rate = hparams[HP_LEARNING_RATE]
    if optimizer == "adam":
        optimizer = tf.optimizers.Adam(learning_rate=learning_rate)
    elif optimizer == "sgd":
        optimizer = tf.optimizers.SGD(learning_rate=learning_rate)
    elif optimizer=='rmsprop':
        optimizer = tf.optimizers.RMSprop(learning_rate=learning_rate)
    else:
        raise ValueError("unexpected optimizer name: %r" % (optimizer_name,))

    # Comiple the mode with the optimizer and learninf rate specified in hparams
```

```
    model.compile(optimizer=optimizer,
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
    print(X_train.shape)
    #Fit the model
    history=model.fit(
    X_train,
    Y_train,
    steps_per_epoch=hparams[HP_NUM_STEPS_PER_EPOCH],
    epochs=epochs,
    validation_data=[X_val,Y_val],
    validation_steps=hparams[HP_NUM_STEPS_PER_EPOCH],
    callbacks=[
        tf.keras.callbacks.TensorBoard(log_dir),  # log metrics
        hp.KerasCallback(log_dir, hparams),# log hparams

    ])
    print("fit the model")
    return history.history['val_accuracy'][-1]

def run(run_dir, hparams):
  with tf.summary.create_file_writer(run_dir).as_default():
    hp.hparams(hparams)  # record the values used in this trial
    accuracy = create_model(hparams)
    #converting to tf scalar
    accuracy= tf.reshape(tf.convert_to_tensor(accuracy), []).numpy()
    tf.summary.scalar(METRIC_ACCURACY, accuracy, step=1)



session_num = 0
for steps in HP_NUM_STEPS_PER_EPOCH.domain.values:
  for num_units in HP_NUM_UNITS.domain.values:
      for optimizer in HP_OPTIMIZER.domain.values:
          for learning_rate in HP_LEARNING_RATE.domain.values:
            hparams = {
                HP_NUM_STEPS_PER_EPOCH: steps,
                HP_NUM_UNITS: num_units,
                HP_OPTIMIZER: optimizer,
                HP_LEARNING_RATE: learning_rate,
            }
            run_name = "run-%d" % session_num
            print('--- Starting trial: %s' % run_name)
            print({h.name: hparams[h] for h in hparams})
            run('logs/hparam_tuning/' + run_name, hparams)
            session_num += 1

!python -m tensorboard.main --logdir="logs/hparam_tuning" --load_fast=false
```

```
1000/1000 [==============================] - 6s 6ms/step - loss: 0.0751 - accura
Epoch 3/5
1000/1000 [==============================] - 6s 6ms/step - loss: 0.0508 - accura
Epoch 4/5
1000/1000 [==============================] - 6s 6ms/step - loss: 0.0395 - accura
Epoch 5/5
1000/1000 [==============================] - 6s 6ms/step - loss: 0.0326 - accura
fit the model
--- Starting trial: run-21
{'steps_per_epoch': 1000, 'num_units': 512, 'optimizer': 'rmsprop', 'learning_ra
<keras.engine.sequential.Sequential object at 0x7fd080215d50>
(49000, 28, 28, 1)
Epoch 1/5
 992/1000 [============================>.] - ETA: 0s - loss: 0.1137 - accuracy:
1000/1000 [==============================] - 8s 8ms/step - loss: 0.1131 - accura
Epoch 2/5
1000/1000 [==============================] - 6s 6ms/step - loss: 0.0365 - accura
Epoch 3/5
1000/1000 [==============================] - 5s 5ms/step - loss: 0.0241 - accura
Epoch 4/5
1000/1000 [==============================] - 6s 6ms/step - loss: 0.0172 - accura
Epoch 5/5
1000/1000 [==============================] - 6s 6ms/step - loss: 0.0131 - accura
fit the model
--- Starting trial: run-22
{'steps_per_epoch': 1000, 'num_units': 512, 'optimizer': 'sgd', 'learning_rate':
<keras.engine.sequential.Sequential object at 0x7fd06a32b210>
(49000, 28, 28, 1)
Epoch 1/5
1000/1000 [==============================] - ETA: 0s - loss: 2.2915 - accuracy:
1000/1000 [==============================] - 7s 7ms/step - loss: 2.2915 - accura
Epoch 2/5
1000/1000 [==============================] - 4s 4ms/step - loss: 2.2772 - accura
Epoch 3/5
1000/1000 [==============================] - 4s 4ms/step - loss: 2.2623 - accura
Epoch 4/5
1000/1000 [==============================] - 4s 4ms/step - loss: 2.2464 - accura
Epoch 5/5
1000/1000 [==============================] - 4s 4ms/step - loss: 2.2289 - accura
fit the model
--- Starting trial: run-23
{'steps_per_epoch': 1000, 'num_units': 512, 'optimizer': 'sgd', 'learning_rate':
<keras.engine.sequential.Sequential object at 0x7fd04405a890>
(49000, 28, 28, 1)
Epoch 1/5
1000/1000 [==============================] - ETA: 0s - loss: 2.1878 - accuracy:
1000/1000 [==============================] - 7s 6ms/step - loss: 2.1878 - accura
Epoch 2/5
1000/1000 [==============================] - 5s 5ms/step - loss: 1.3861 - accura
Epoch 3/5
1000/1000 [==============================] - 4s 4ms/step - loss: 0.5754 - accura
Epoch 4/5
1000/1000 [==============================] - 4s 4ms/step - loss: 0.4023 - accura
Epoch 5/5
1000/1000 [==============================] - 4s 4ms/step - loss: 0.3375 - accura
```

```
fit the model
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass
TensorBoard 2.8.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

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

TensorBoard SCALARS HPARAMS INACTIVE

**Hyperparameters**

- num_units
  - Min: -infinity
  - Max: +infinity
- optimizer
  - rmsprop
  - adam
  - sgd
- learning_rate
  - Min: -infinity
  - Max: +infinity
- steps_per_epo
  - Min: -infinity
  - Max: +infinity

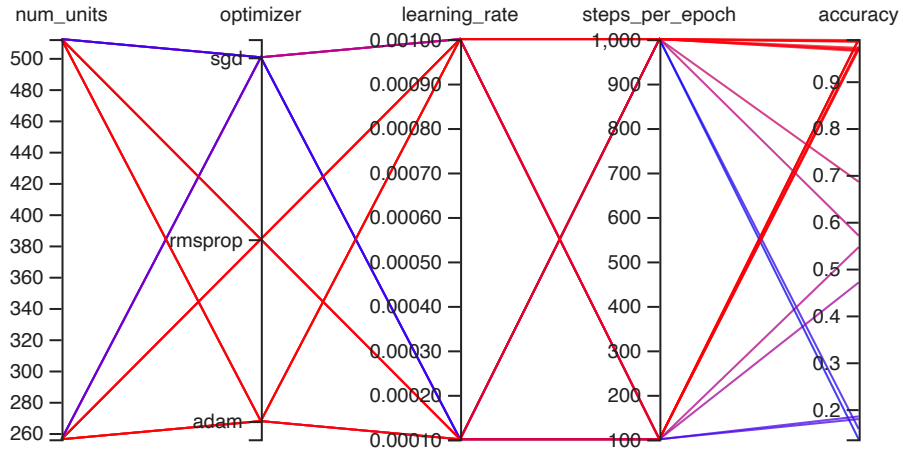
**Metrics**

- accuracy
  - Min: -infinity
  - Max: +infinit

TABLE VIEW PARALLEL COORDINATES VIEW SCATTER PLOT MATRIX VIEW

Color by: accuracy

| num_units | learning_rate | steps_per_epoch | accuracy |
|---|---|---|---|
| Linear | Linear | Linear | Linear |
| Logarithmic | Logarithmic | Logarithmic | Logarithmic |
| Quantile | Quantile | Quantile | Quantile |




Click or hover over a session group to display its values here.

## No session group selected

Please select a session group to see its metric-graphs here

```
tensorboard --logdir=data/ --host localhost --port 8088
!python -m tensorboard.main --logdir="logs/hparam_tuning" --load_fast=false
```

```
NOTE: Using experimental fast data loading logic. To disable, pass
    "--load_fast=false" and report issues on GitHub. More details:
    https://github.com/tensorflow/tensorboard/issues/4784

TensorBoard 2.8.0 at http://localhost:8088/ (Press CTRL+C to quit)
^C
Traceback (most recent call last):
  File "/usr/lib/python3.7/runpy.py", line 193, in _run_module_as_main
    "__main__", mod_spec)
  File "/usr/lib/python3.7/runpy.py", line 85, in _run_code
    exec(code, run_globals)
  File "/usr/local/lib/python3.7/dist-packages/tensorboard/main.py", line 53, in
  File "/usr/local/lib/python3.7/dist-packages/tensorboard/main.py", line 46, in
    app.run(tensorboard.main, flags_parser=tensorboard.configure)
  File "/usr/local/lib/python3.7/dist-packages/absl/app.py", line 302, in run
    flags_parser,
  File "/usr/local/lib/python3.7/dist-packages/absl/app.py", line 371, in _run_i
    flags_parser=flags_parser,
  File "/usr/local/lib/python3.7/dist-packages/absl/app.py", line 216, in _regis
    args_to_main = flags_parser(original_argv)
  File "/usr/local/lib/python3.7/dist-packages/tensorboard/program.py", line 181
    "TensorBoard is a suite of web applications for "
  File "/usr/local/lib/python3.7/dist-packages/absl/flags/argparse_flags.py", li
    self._define_absl_flags(self._inherited_absl_flags)
  File "/usr/local/lib/python3.7/dist-packages/absl/flags/argparse_flags.py", li
    self._define_absl_flag(flag_instance, suppress)
  File "/usr/local/lib/python3.7/dist-packages/absl/flags/argparse_flags.py", li
    flag_instance=flag_instance)
  File "/usr/lib/python3.7/argparse.py", line 1337, in add_argument
    def add_argument(self, *args, **kwargs):
KeyboardInterrupt
```

✓ 4s completed at 2:45 PM