```
import numpy as np
from tensorflow.keras.datasets import boston_housing
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import TensorBoard # new!
import os


(X_train, y_train), (X_valid, y_valid) = boston_housing.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/boston_housing.npz
57026/57026 [==============================] - 0s 0us/step
```

```
X_train.shape
```

```
(404, 13)
```

```
X_valid.shape
```

```
(102, 13)
```

```
X_train[0]
```

```
array([  1.23247,   0.     ,   8.14   ,   0.     ,   0.538  ,   6.142  ,
        91.7    ,   3.9769 ,   4.     , 307.     ,  21.     , 396.9    ,
        18.72   ])
```

```
y_train[0]
```

```
15.2
```

```
model = Sequential()

model.add(Dense(32, input_dim=13, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(16, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(1, activation='linear'))



model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                448

 batch_normalization (BatchN  (None, 32)               128
 ormalization)

 dense_1 (Dense)             (None, 16)                528

 batch_normalization_1 (Batc  (None, 16)               64
 hNormalization)

 dropout (Dropout)           (None, 16)                0

 dense_2 (Dense)             (None, 1)                 17

=================================================================
Total params: 1,185
Trainable params: 1,089
Non-trainable params: 96
_________________________________________________________________
```

```
model.compile(loss='mean_squared_error', optimizer='adam', )
```

```
output_dir = 'model_output/'


run_name = 'regression_baseline'
output_path = output_dir + run_name


if not os.path.exists(output_path):
    os.makedirs(output_path)


modelcheckpoint = ModelCheckpoint(output_path + '/weights.{epoch:02d}.hdf5', # decimal integers
                                  save_weights_only=True) # otherwise full model is saved


tensorboard = TensorBoard(log_dir='logs/' + run_name)



model.fit(X_train, y_train,
          batch_size=8, epochs=32, verbose=1,
          validation_data=(X_valid, y_valid),
          callbacks=[modelcheckpoint, tensorboard])
```

```
Epoch 1/32
51/51 [==============================] - 2s 7ms/step - loss: 574.0970 - val_loss: 776.6132
Epoch 2/32
51/51 [==============================] - 0s 3ms/step - loss: 548.7177 - val_loss: 671.3480
Epoch 3/32
51/51 [==============================] - 0s 3ms/step - loss: 535.3249 - val_loss: 583.2804
Epoch 4/32
51/51 [==============================] - 0s 3ms/step - loss: 518.2819 - val_loss: 521.6576
Epoch 5/32
51/51 [==============================] - 0s 3ms/step - loss: 488.8458 - val_loss: 480.8918
Epoch 6/32
51/51 [==============================] - 0s 3ms/step - loss: 463.6249 - val_loss: 467.7474
Epoch 7/32
51/51 [==============================] - 0s 3ms/step - loss: 433.4673 - val_loss: 451.1673
Epoch 8/32
51/51 [==============================] - 0s 3ms/step - loss: 394.6075 - val_loss: 420.5128
Epoch 9/32
51/51 [==============================] - 0s 3ms/step - loss: 353.2959 - val_loss: 343.9286
Epoch 10/32
51/51 [==============================] - 0s 3ms/step - loss: 310.3732 - val_loss: 293.4747
Epoch 11/32
51/51 [==============================] - 0s 3ms/step - loss: 278.6436 - val_loss: 348.6489
Epoch 12/32
51/51 [==============================] - 0s 6ms/step - loss: 238.9618 - val_loss: 224.3591
Epoch 13/32
51/51 [==============================] - 0s 4ms/step - loss: 197.4227 - val_loss: 361.9974
Epoch 14/32
51/51 [==============================] - 0s 5ms/step - loss: 174.6084 - val_loss: 305.8211
Epoch 15/32
51/51 [==============================] - 0s 4ms/step - loss: 137.6198 - val_loss: 216.6967
Epoch 16/32
51/51 [==============================] - 0s 5ms/step - loss: 116.3654 - val_loss: 199.3342
Epoch 17/32
51/51 [==============================] - 0s 5ms/step - loss: 102.7322 - val_loss: 104.1574
Epoch 18/32
51/51 [==============================] - 0s 5ms/step - loss: 77.0915 - val_loss: 103.1260
Epoch 19/32
51/51 [==============================] - 0s 5ms/step - loss: 71.5326 - val_loss: 69.4263
Epoch 20/32
51/51 [==============================] - 0s 5ms/step - loss: 58.3281 - val_loss: 44.9665
Epoch 21/32
51/51 [==============================] - 0s 5ms/step - loss: 54.3541 - val_loss: 87.9172
Epoch 22/32
51/51 [==============================] - 0s 4ms/step - loss: 50.0266 - val_loss: 47.5559
Epoch 23/32
51/51 [==============================] - 0s 4ms/step - loss: 43.8610 - val_loss: 54.2597
Epoch 24/32
51/51 [==============================] - 0s 3ms/step - loss: 44.7230 - val_loss: 23.4884
Epoch 25/32
51/51 [==============================] - 0s 3ms/step - loss: 49.9817 - val_loss: 37.9386
Epoch 26/32
51/51 [==============================] - 0s 3ms/step - loss: 46.5498 - val_loss: 57.4117
Epoch 27/32
51/51 [==============================] - 0s 4ms/step - loss: 44.6017 - val_loss: 33.6597
Epoch 28/32
51/51 [==============================] - 0s 3ms/step - loss: 42.3547 - val_loss: 30.5425
Epoch 29/32
51/51 [==============================] - 0s 3ms/step - loss: 49.8649 - val_loss: 37.6829
```

```
model.load_weights(output_path + '/weights.28.hdf5')
```

```
X_valid[42]
```

```
array([  9.32909,   0.     ,  18.1    ,   0.     ,   0.713  ,   6.185  ,
        98.7    ,   2.2616 ,  24.     , 666.     ,  20.2    , 396.9    ,
        18.13   ])
```

```
y_valid[42]
```

```
14.1
```

```
model.predict(np.reshape(X_valid[42], [1, 13]))
```

```
1/1 [==============================] - 0s 313ms/step
array([[16.164938]], dtype=float32)
```