```
import tensorflow as tf
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
import tensorflow_hub as hub
import math,os,datetime
```

```
BATCH_SIZE = 32
```

```
train_data,train_info = tfds.load('cifar10',split='train[10%:90%]',with_info=True)
val_data = tfds.load('cifar10',split='train[0%:10%]')
test_data = tfds.load('cifar10',split='test')
print(train_data)
```

```
Downloading and preparing dataset cifar10/3.0.2 (download: 162.17 MiB, generated: 132.4
Dl Completed...: 100%                    1/1 [00:06<00:00, 6.01s/ url]
Dl Size...: 100%                         162/162 [00:05<00:00, 27.14 MiB/s]
Extraction completed...: 100%            1/1 [00:05<00:00, 5.93s/ file]
Shuffling and writing examples to /root/tensorflow_datasets/cifar10/3.0.2.incomplete3JE
92%                                      45773/50000 [00:03<00:00, 67406.34 examples/s]
Shuffling and writing examples to /root/tensorflow_datasets/cifar10/3.0.2.incomplete3JE
0%                                       0/10000 [00:00<?, ? examples/s]
Dataset cifar10 downloaded and prepared to /root/tensorflow_datasets/cifar10/3.0.2. Sub
<PrefetchDataset shapes: {id: (), image: (32, 32, 3), label: ()}, types: {id: tf.string
```

```
num_train_data = 0
for _ in train_data:
  num_train_data +=1
print(num_train_data)
```

```
40000
```

```
num_val_data = 0
for _  in val_data:
  num_val_data+=1
print(num_val_data)
```

```
5000
```

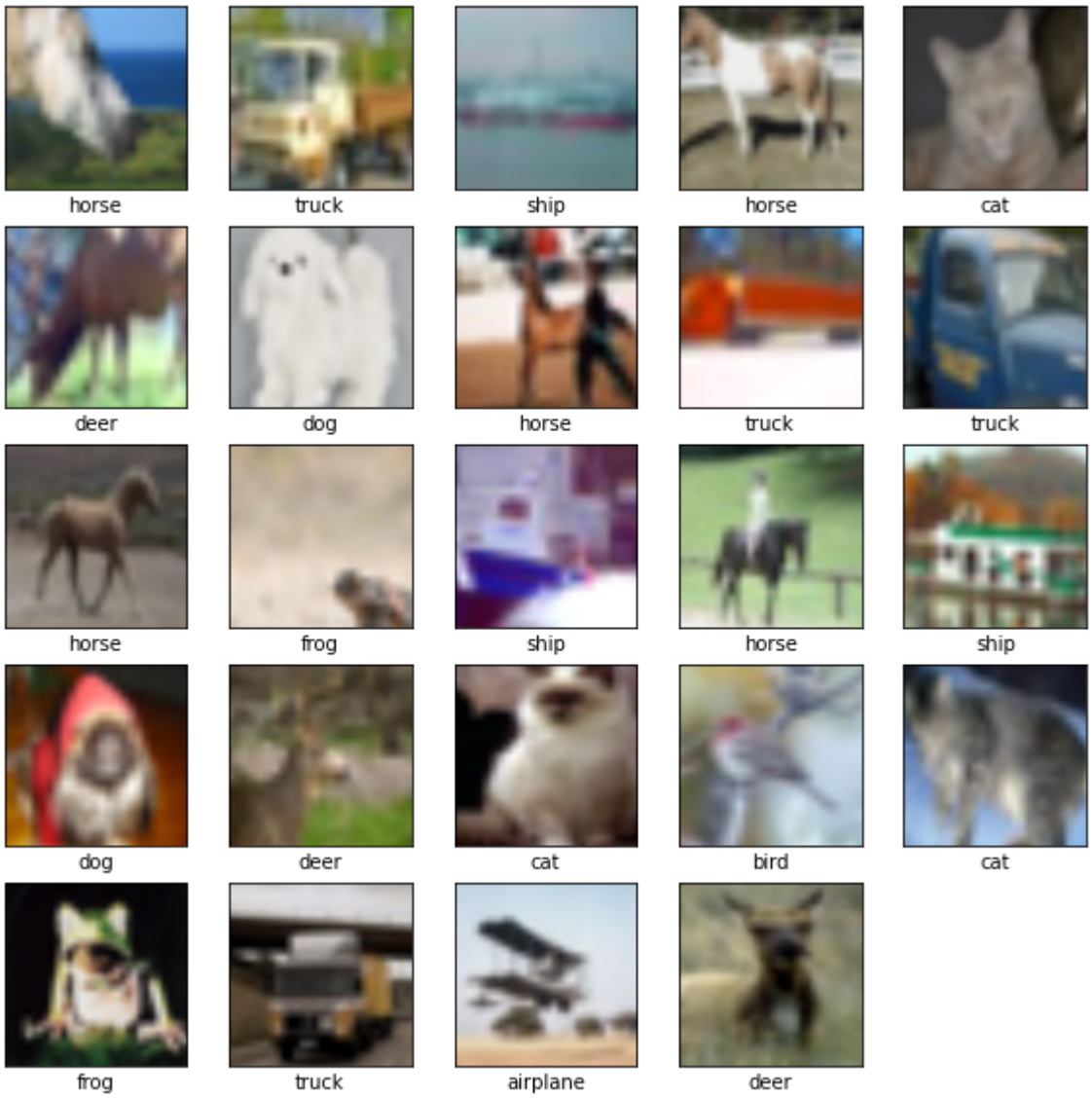
```
train_steps_per_epoch = math.ceil(num_train_data/BATCH_SIZE)
val_steps_per_epoch = math.ceil(num_val_data/BATCH_SIZE)


def normalizer(features,input_shape = [224,224,3],augment=True,seed=42):
  input_shape = tf.convert_to_tensor(input_shape)
  image = features['image']
  image = tf.image.convert_image_dtype(image,tf.float32)
  if augment:
    ## Randomly applied horizontal flip:
        image = tf.image.random_flip_left_right(image, seed=seed)

        # Random B/S changes:
        image = tf.image.random_brightness(image, max_delta=0.1, seed=seed)
        image = tf.image.random_saturation(image, lower=0.5, upper=1.5, seed=seed)
        image = tf.clip_by_value(image, 0.0, 1.0) # keeping pixel values in check

        # Random resize and random crop back to expected size:

        random_scale_factor = tf.random.uniform([1], minval=1., maxval=1.4, dtype=tf.float32,
        scaled_height = tf.cast(tf.cast(input_shape[0], tf.float32) * random_scale_factor,
                                tf.int32)
        scaled_width = tf.cast(tf.cast(input_shape[1], tf.float32) * random_scale_factor,
                               tf.int32)
        scaled_shape = tf.squeeze(tf.stack([scaled_height, scaled_width]))
        image = tf.image.resize(image, scaled_shape)
        image = tf.image.random_crop(image, input_shape, seed=seed)
  else:
    image = tf.image.resize(image,input_shape[:2])
  label = features['label']
  features = (image,label)
  return features



train_data = train_data.map(normalizer)
val_data = val_data.map(normalizer)



print(train_data)
print(val_data)
```

```
<MapDataset shapes: ((224, 224, 3), ()), types: (tf.float32, tf.int64)>
<MapDataset shapes: ((224, 224, 3), ()), types: (tf.float32, tf.int64)>
```

```
class_names = train_info.features["label"].names
print(class_names)
```

```
['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truc
```

```
plt.figure(figsize=(10,10))
for i , (image,label) in enumerate(train_data.take(24)):
  #image = image.numpy().reshape([28,28,3])
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
  plt.imshow(image,cmap=plt.cm.binary)
  plt.xlabel(class_names[label])
plt.show()
```




```
train_data = train_data.batch(BATCH_SIZE)
val_data = val_data.batch(BATCH_SIZE)
train_data = train_data.prefetch(1)
val_data = val_data.prefetch(1)
#train_data = train_data.cache()
```

```
print(val_data)
```

```
print(val_data)
print(train_data)
```

```
<PrefetchDataset shapes: ((None, 224, 224, 3), (None,)), types: (tf.float32, tf.int64)>
<PrefetchDataset shapes: ((None, 224, 224, 3), (None,)), types: (tf.float32, tf.int64)>
```

```
Resnet = tf.keras.applications.ResNet50()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resn
102973440/102967424 [==============================] - 1s 0us/step
```

```
Resnet.summary()
```

```
conv5_block1_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block1_3_conv
__________________________________________________________________________________________
conv5_block1_add (Add)          (None, 7, 7, 2048)   0           conv5_block1_0_bn[0
                                                                 conv5_block1_3_bn[0

__________________________________________________________________________________________
conv5_block1_out (Activation)   (None, 7, 7, 2048)   0           conv5_block1_add[0]
__________________________________________________________________________________________
conv5_block2_1_conv (Conv2D)    (None, 7, 7, 512)    1049088     conv5_block1_out[0]
__________________________________________________________________________________________
conv5_block2_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block2_1_conv
__________________________________________________________________________________________
conv5_block2_1_relu (Activation (None, 7, 7, 512)    0           conv5_block2_1_bn[0
__________________________________________________________________________________________
conv5_block2_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block2_1_relu
__________________________________________________________________________________________
conv5_block2_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block2_2_conv
__________________________________________________________________________________________
conv5_block2_2_relu (Activation (None, 7, 7, 512)    0           conv5_block2_2_bn[0
__________________________________________________________________________________________
conv5_block2_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block2_2_relu
__________________________________________________________________________________________
conv5_block2_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block2_3_conv
__________________________________________________________________________________________
conv5_block2_add (Add)          (None, 7, 7, 2048)   0           conv5_block1_out[0]
                                                                 conv5_block2_3_bn[0

__________________________________________________________________________________________
conv5_block2_out (Activation)   (None, 7, 7, 2048)   0           conv5_block2_add[0]
__________________________________________________________________________________________
conv5_block3_1_conv (Conv2D)    (None, 7, 7, 512)    1049088     conv5_block2_out[0]
__________________________________________________________________________________________
conv5_block3_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block3_1_conv
__________________________________________________________________________________________
conv5_block3_1_relu (Activation (None, 7, 7, 512)    0           conv5_block3_1_bn[0
__________________________________________________________________________________________
conv5_block3_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block3_1_relu
__________________________________________________________________________________________
conv5_block3_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block3_2_conv
__________________________________________________________________________________________
conv5_block3_2_relu (Activation (None, 7, 7, 512)    0           conv5_block3_2_bn[0
```

```
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block3_2_relu
__________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block3_3_conv
__________________________________________________________________________________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_out[0]
                                                                 conv5_block3_3_bn[0
__________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_add[0]
__________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           conv5_block3_out[0]
__________________________________________________________________________________________
predictions (Dense)             (None, 1000)         2049000     avg_pool[0][0]
==========================================================================================
Total params: 25,636,712
Trainable params: 25,583,592
Non-trainable params: 53,120
__________________________________________________________________________________________
```

```
model_dir = './models/Resnet'
logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
callbacks = [
    # Callback to interrupt the training if the validation loss/metrics stops improving for s
    tf.keras.callbacks.EarlyStopping(patience=8, monitor='val_acc',
                                     restore_best_weights=True),
    # Callback to log the graph, losses and metrics into TensorBoard:

        tf.keras.callbacks.TensorBoard(model_dir, histogram_freq=1,write_graph=True)

  # Callback to simply log metrics at the end of each epoch (saving space compared to verbose
]
Resnet.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=
              [tf.keras.metrics.SparseCategoricalAccuracy(name='acc'),
               tf.keras.metrics.SparseTopKCategoricalAccuracy(k=5,name='top5_acc')])


history = Resnet.fit(train_data,epochs=20,steps_per_epoch=train_steps_per_epoch,validation_da
```

```
Epoch 1/20
1250/1250 [==============================] - 471s 346ms/step - loss: 1.3580 - acc: 0.55
Epoch 2/20
1250/1250 [==============================] - 438s 350ms/step - loss: 0.6499 - acc: 0.77
Epoch 3/20
1250/1250 [==============================] - 437s 350ms/step - loss: 0.4627 - acc: 0.83
Epoch 4/20
1250/1250 [==============================] - 436s 349ms/step - loss: 0.3329 - acc: 0.88
Epoch 5/20
1250/1250 [==============================] - 436s 349ms/step - loss: 0.2329 - acc: 0.91
Epoch 6/20
1250/1250 [==============================] - 436s 349ms/step - loss: 0.1767 - acc: 0.93
Epoch 7/20
1250/1250 [==============================] - 436s 349ms/step - loss: 0.1301 - acc: 0.95
Epoch 8/20
1250/1250 [==============================] - 436s 349ms/step - loss: 0.1022 - acc: 0.96
```

```
Epoch 9/20
1250/1250 [==============================] - 435s 348ms/step - loss: 0.0857 - acc: 0.97
Epoch 10/20
1250/1250 [==============================] - 436s 348ms/step - loss: 0.0770 - acc: 0.97
Epoch 11/20
1250/1250 [==============================] - 436s 349ms/step - loss: 0.0764 - acc: 0.97
Epoch 12/20
1250/1250 [==============================] - 435s 348ms/step - loss: 0.0556 - acc: 0.98
Epoch 13/20
1250/1250 [==============================] - 435s 348ms/step - loss: 0.0562 - acc: 0.98
Epoch 14/20
1250/1250 [==============================] - 435s 348ms/step - loss: 0.0515 - acc: 0.98
Epoch 15/20
1250/1250 [==============================] - 434s 347ms/step - loss: 0.0494 - acc: 0.98
Epoch 16/20
1250/1250 [==============================] - 434s 347ms/step - loss: 0.0388 - acc: 0.98
Epoch 17/20
1250/1250 [==============================] - 434s 347ms/step - loss: 0.0426 - acc: 0.98
Epoch 18/20
1250/1250 [==============================] - 434s 348ms/step - loss: 0.0351 - acc: 0.98
Epoch 19/20
1250/1250 [==============================] - 434s 347ms/step - loss: 0.0387 - acc: 0.98
Epoch 20/20
1250/1250 [==============================] - 434s 348ms/step - loss: 0.0323 - acc: 0.98
```

```
Resnet.save_weights("lenet_weights",overwrite=True)
Resnet.save('My_ResNet')
```

```
INFO:tensorflow:Assets written to: My_ResNet/assets
INFO:tensorflow:Assets written to: My_ResNet/assets
```

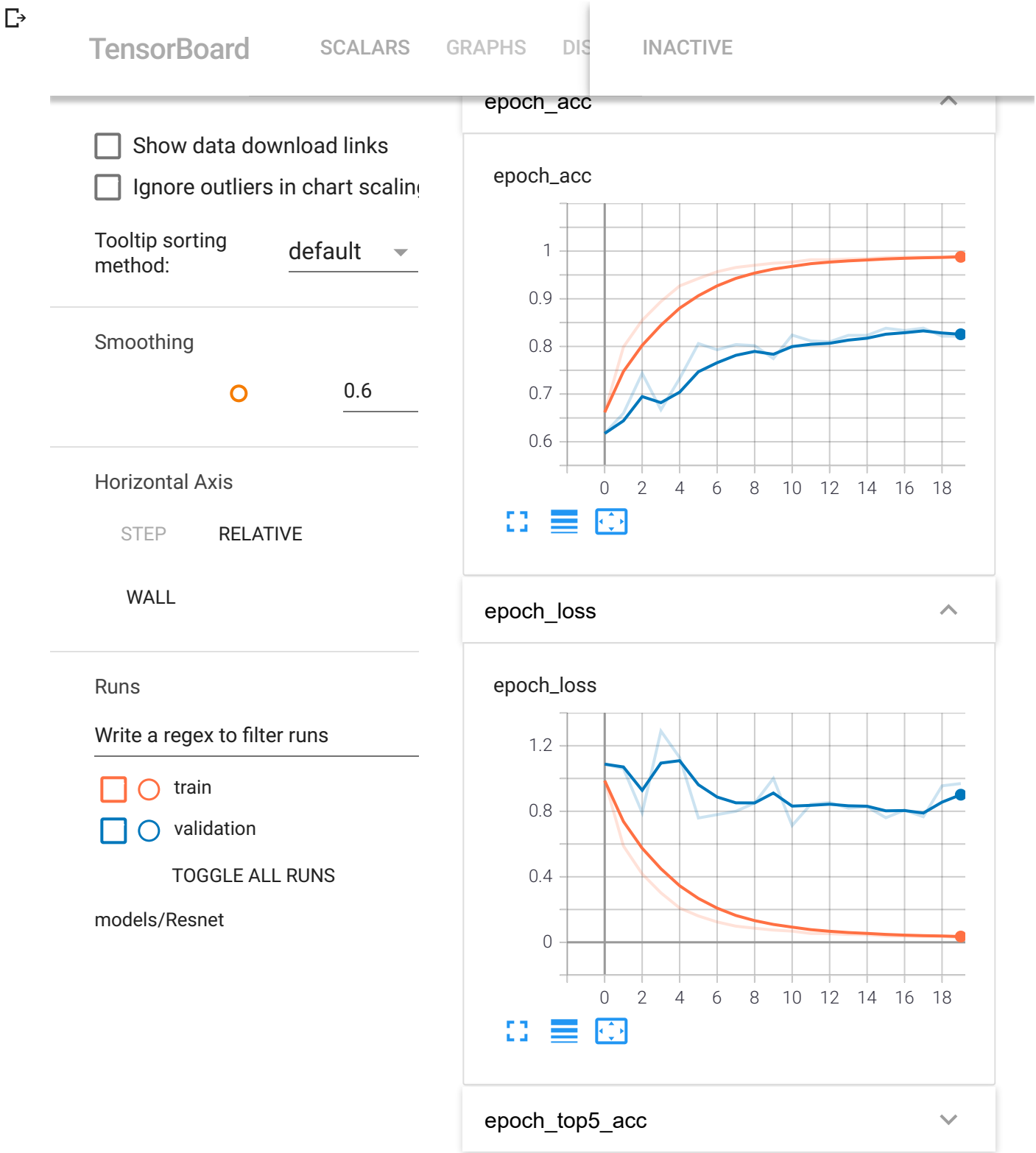
```
test_data = tfds.load('cifar10', split='test')
num_tests=0
for _ in test_data:
    num_tests+=1
print(num_tests)

test_data = test_data.map(normalizer)
test_data = test_data.batch(BATCH_SIZE)
test_data = test_data.prefetch(1)

Resnet.evaluate(test_data,batch_size=BATCH_SIZE,verbose=1)
```

```
10000
313/313 [==============================] - 30s 94ms/step - loss: 0.9160 - acc: 0.8233 -
[0.9159789085388184, 0.8233000040054321, 0.9865999817848206]
```

```
%load_ext tensorboard
```

```
rm -rf ./logs/
```

```
%tensorboard --logdir models/Resnet
```



✓ 4s completed at 8:17 PM