

Copyright Notice

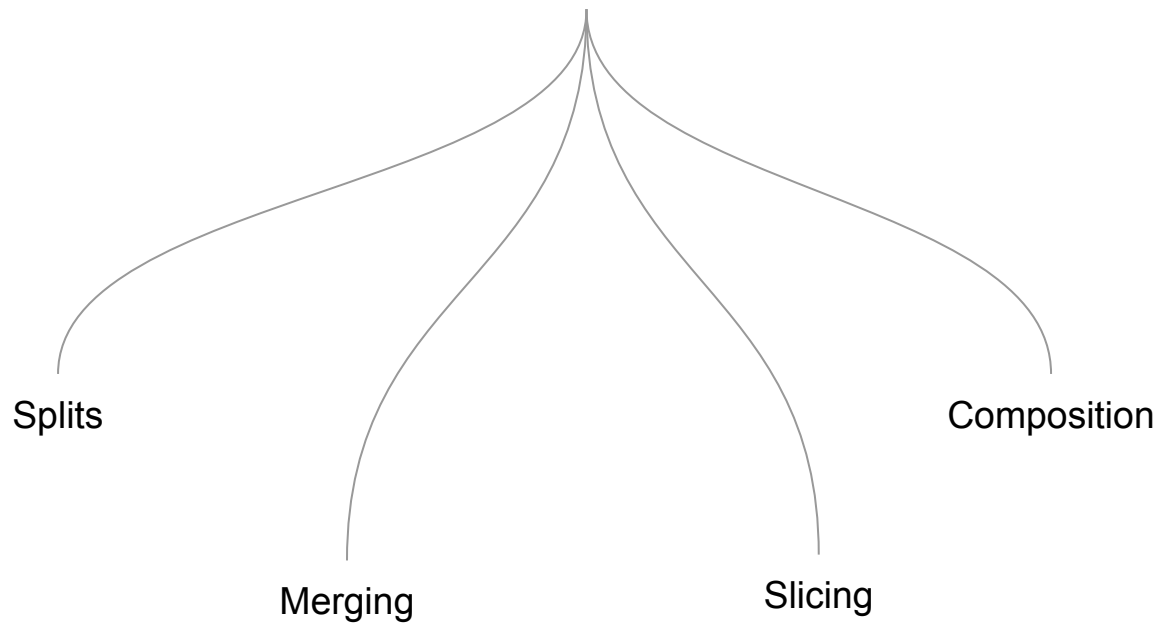
These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see

<https://creativecommons.org/licenses/by-sa/2.0/legalcode>

Splits API



How to Slice Lists/Arrays and Tuples in Python

Published: Saturday 30th March 2013

So you've got an list, tuple or array and you want to get specific sets of sub-elements from it, without any long, drawn out for loops?

Python has an amazing feature just for that called *slicing*. Slicing can not only be used for lists, tuples or arrays, but custom data structures as well, with the *slice* object, which will be used later on in this article.

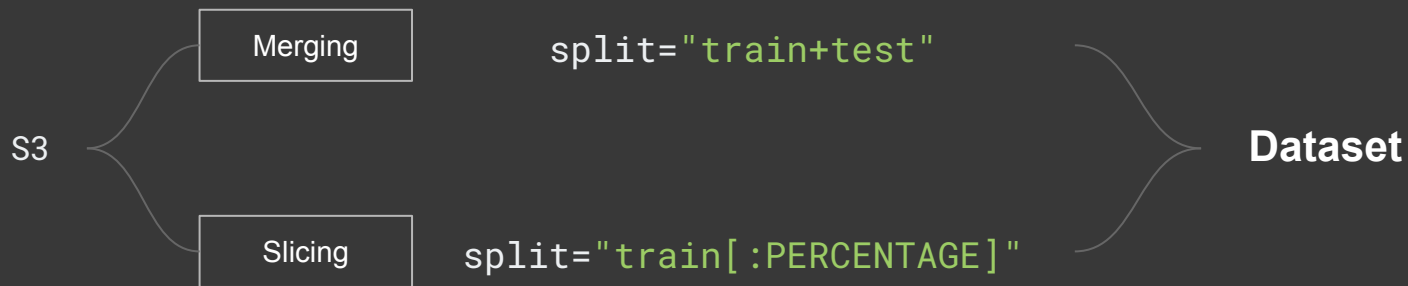
Slicing Python Lists/Arrays and Tuples Syntax

Let's start with a normal, everyday list.

```
1 >>> a = [1, 2, 3, 4, 5, 6, 7, 8]
```

Nothing crazy, just a normal list with the numbers 1 through 8. Now let's say that we really want the sub-elements 2, 3, and 4 returned in a new list. How do we do that?

Splits API



Distinct splits

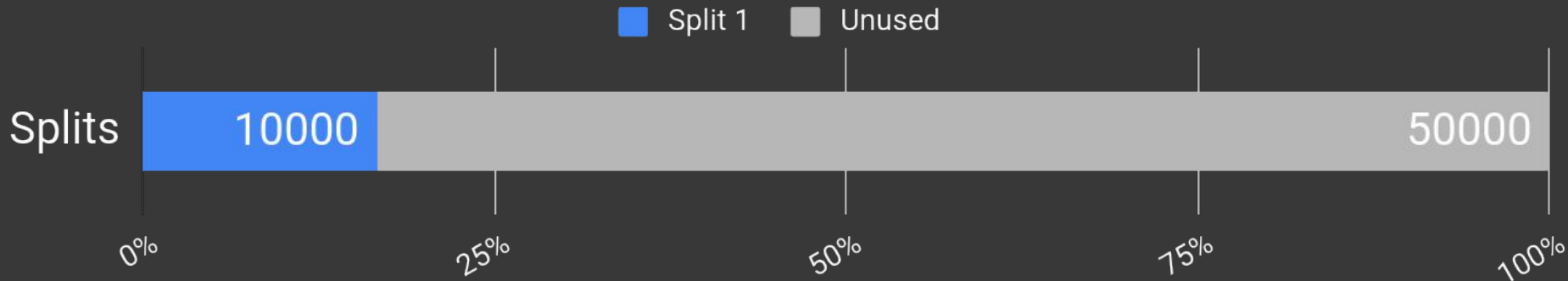
```
# The full `train` split and the full `test` split as two distinct datasets.  
train_ds, test_ds = tfds.load('mnist:3.*.*', split=['train', 'test'])
```

Merging



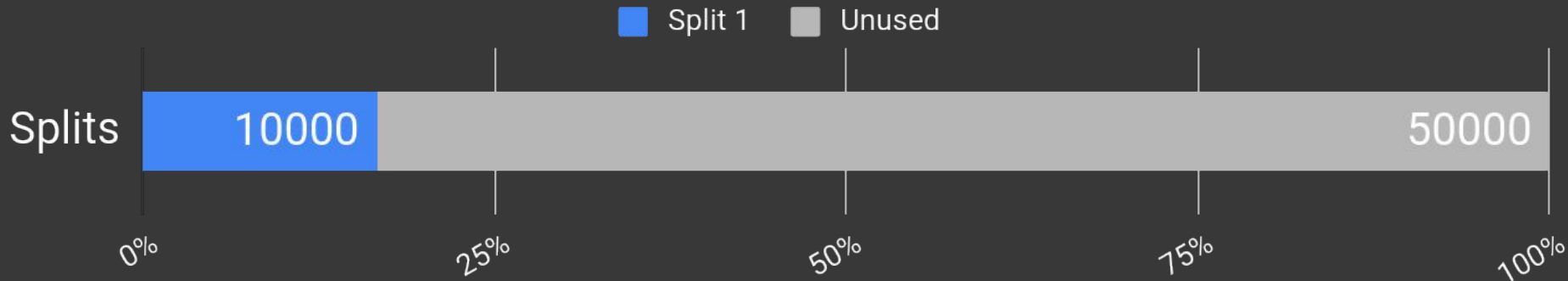
```
# The full `train` and `test` splits, concatenated together.  
combined = tfds.load('mnist:3.*.*', split='train+test')
```

Slicing by index



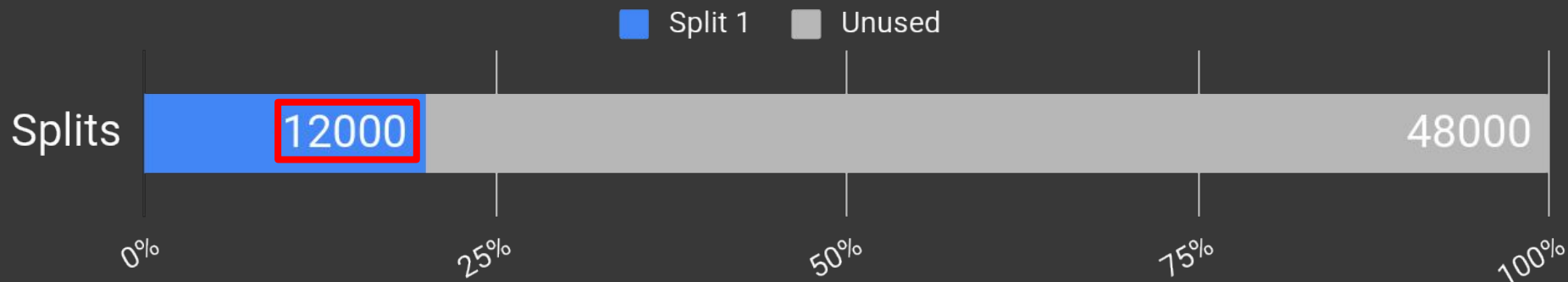
```
tfds.load('mnist:3.*.*', split='train[:10000]')
```

Slicing by index



```
tfds.load('mnist:3.*.*', split='train[:10000]')
```


Slicing by percentage



```
tfds.load('mnist:3.*.*', split='train[:20%]')
```

K-fold splits

The **validation** datasets are each going to be 20%

`[0%:20%], [20%:40%], ..., [80%:100%]`

The **training** datasets are each going to be the complementary 80%

[20%:100%] (for a validation set of [0%:20%])

$[0\%:20\%] + [20\%:100\%]$ (for a validation set of $[20\%:40\%]$)

[0% : 80%] (for a validation set of [80% : 100%])

And so on ...

[illegible]

K-fold splits

The **validation** datasets are each going to be 20%

`[0%:20%], [20%:40%], ..., [80%:100%]`

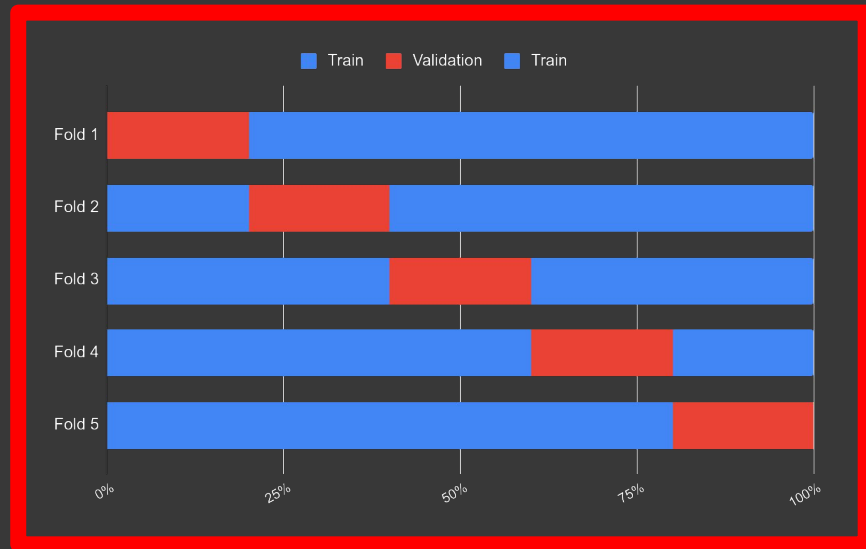
The **training** datasets are each going to be the complementary 80%

`[20%:100%]` (for a validation set of `[0%:20%]`)

`[0%:20%] + [20%:100%]` (for a validation set of `[20%:40%]`)

`[0%:80%]` (for a validation set of `[80%:100%]`)

And so on ...



```
val_ds = tfds.load('mnist:3.*.*', split=['train[{}%:{}]'.format(k, k+20)
```

```
for k in range(0, 100, 20)])
```

```
train_ds = tfds.load('mnist:3.*.*', split=['train[:{}%]+train[{}%:]'.format(k, k+20)
```

```
for k in range(0, 100, 20)])
```

K-fold splits

The **validation** datasets are each going to be 20%

`[0%:20%]`, `[20%:40%]`, ..., `[80%:100%]`

The **training** datasets are each going to be the complementary 80%

`[20%:100%]` (for a validation set of `[0%:20%]`)

`[0%:20%] + [20%:100%]` (for a validation set of `[20%:40%]`)

`[0%:80%]` (for a validation set of `[80%:100%]`)

And so on ...



```
val_ds = tfds.load('mnist:3.*.*', split=['train[{}%:{}]'.format(k, k+20)
```

```
for k in range(0, 100, 20)])
```

```
train_ds = tfds.load('mnist:3.*.*', split=['train[:{}%]+train[{}%:]'.format(k, k+20)
```

```
for k in range(0, 100, 20)])
```

K-fold splits

The **validation** datasets are each going to be 20%

`[0%:20%], [20%:40%], ..., [80%:100%]`

The **training** datasets are each going to be the complementary 80%

[20%:100%] (for a validation set of [0%:20%])

$[0\%:20\%] + [20\%:100\%]$ (for a validation set of $[20\%:40\%]$)

[0% : 80%] (for a validation set of [80% : 100%])

And so on ...

[illegible]

K-fold splits

The **validation** datasets are each going to be 20%

`[0%:20%], [20%:40%], ..., [80%:100%]`

The **training** datasets are each going to be the complementary 80%

[20%:100%] (for a validation set of [0%:20%])

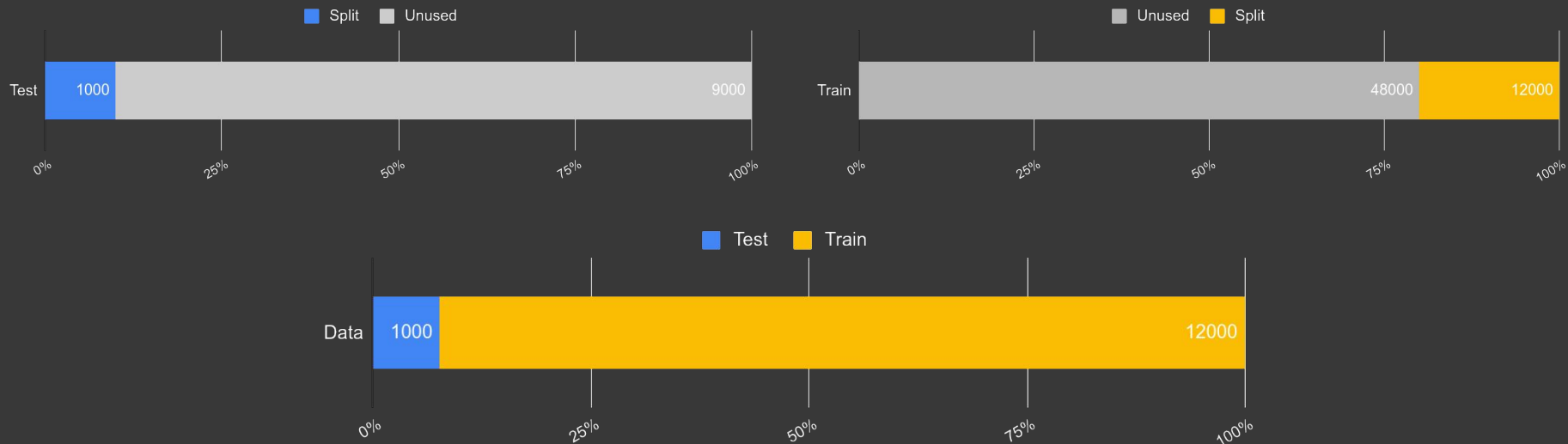
$[0\%:20\%] + [20\%:100\%]$ (for a validation set of $[20\%:40\%]$)

[0% : 80%] (for a validation set of [80% : 100%])

And so on ...

[illegible]

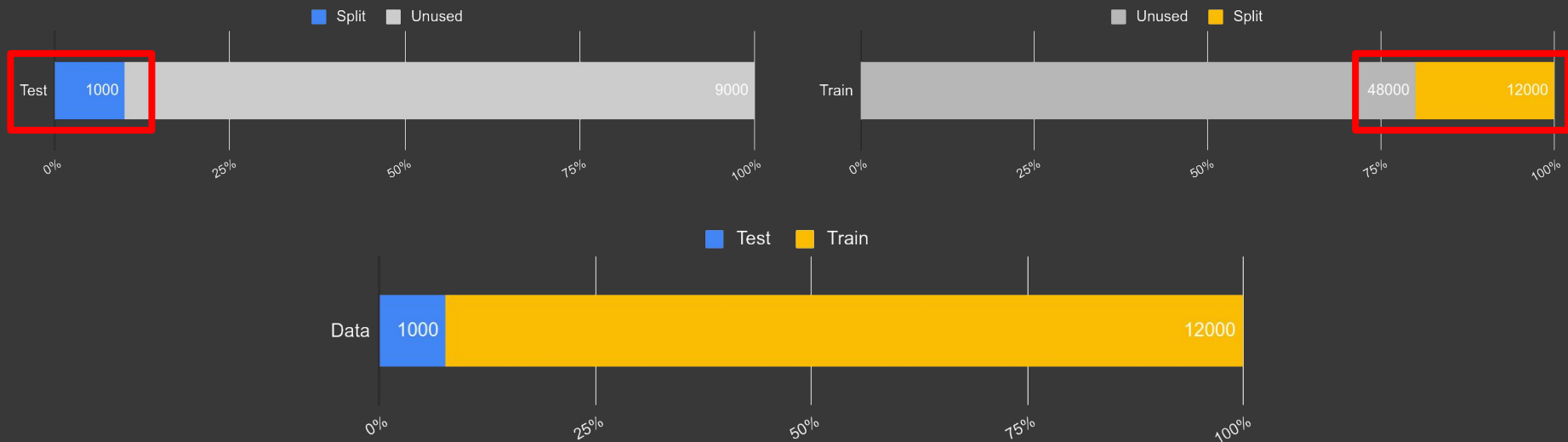
Composing operations



The first 10% of test + the last 80% of train.

```
10_80pct_ds = tfds.load('mnist:3.*.*', split='test[:10%]+train[-80%:]')
```

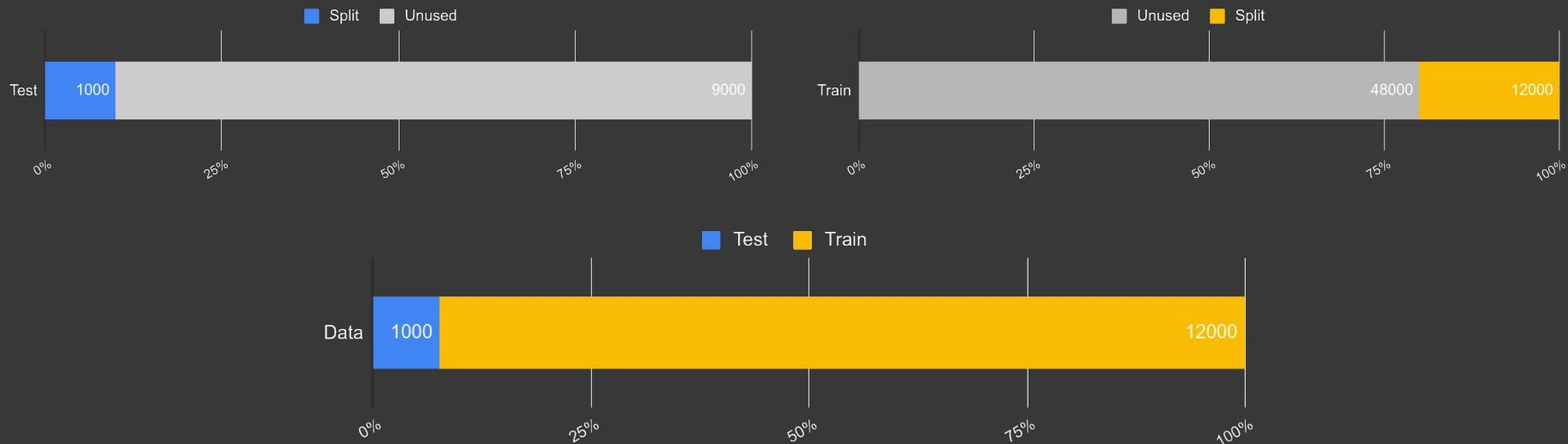
Composing operations



The first 10% of test + the last 80% of train.

```
10_80pct_ds = tfds.load('mnist:3.*.*', split='test[:10%]+train[-80%:]')
```


Composing operations



The first 10% of test + the last 80% of train.

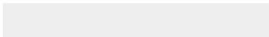
```
10_80pct_ds = tfds.load('mnist:3.*.*', split='test[:10%]+train[-80%:]')
```

```
data = tfds.load("cnn_dailymail")
```

Downloading and preparing dataset cnn_dailymail/plain_text/3.0.0 (download: 558.32 MiB, generated: 1.27 GiB, total:

DI Completed.... 80%  **4/5 [00:12<00:01, 1.35s/ url]**

DI Size...:  **476/? [00:12<00:00, 12.73 MiB/s]**

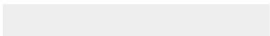
Extraction completed...: 0%  **0/1 [00:12<?, ? file/s]**

Downloading the Data

Downloading and preparing dataset `cnn_dailymail/plain_text/3.0.0` (download: 558.32 MiB, generated: 1.27 GiB, total:

DI Completed.... 80%  4/5 [00:12<00:01, 1.35s/ url]

DI Size....  476/? [00:12<00:00, 12.73 MiB/s]

Extraction completed.... 0%  0/1 [00:12<?, ? file/s]

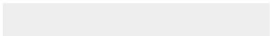
 45571/0 [00:27<00:00, 1637.21 examples/s]

Extracting the Data

Downloading and preparing dataset `cnn_dailymail/plain_text/3.0.0` (download: 558.32 MiB, generated: 1.27 GiB, total:

DI Completed.... 80%  4/5 [00:12<00:01, 1.35s/ url]

DI Size...:  476/? [00:12<00:00, 12.73 MiB/s]


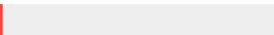
Extraction completed...: 0%  0/1 [00:12<?, ? file/s]

 45571/0 [00:27<00:00, 1637.21 examples/s]

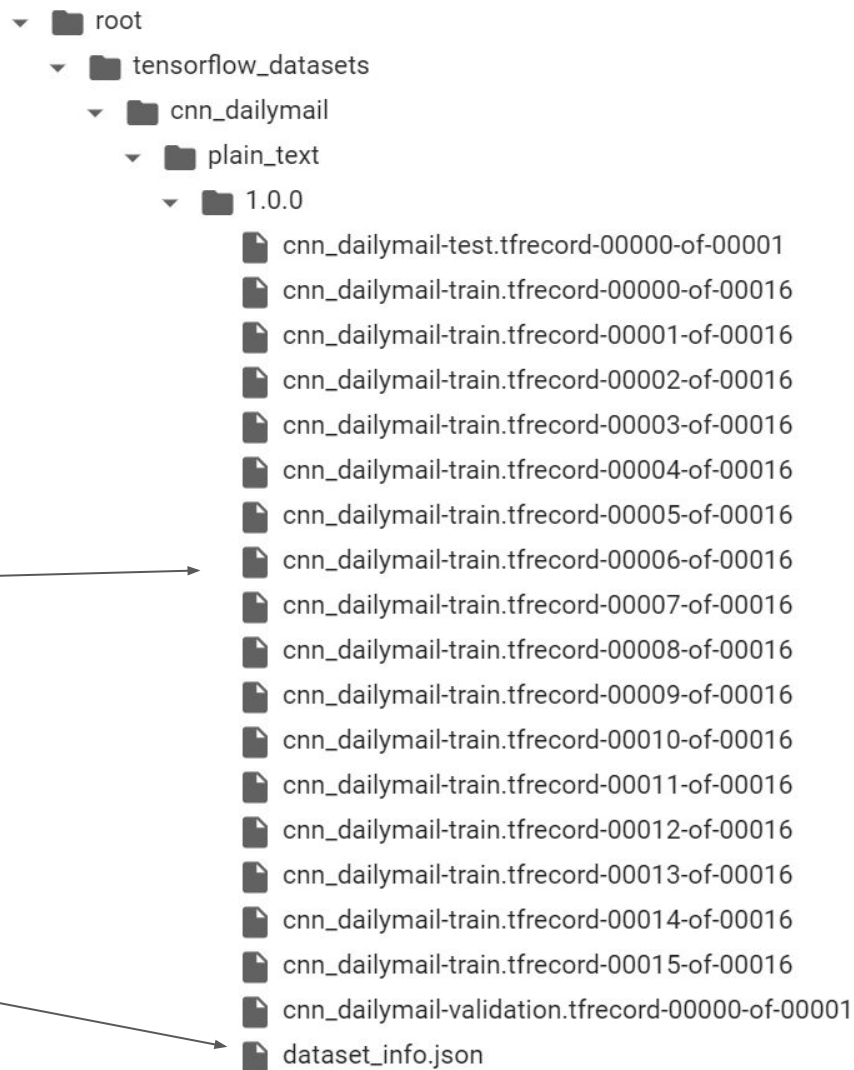
Shuffling and writing examples to `/root/tensorflow_datasets/cnn_dailymail/plain_text/3.0.0.incomplete7W6N6S/cnn_dail`

99%  284001/287113 [00:06<00:00, 46165.36 examples/s]

Shuffling and writing examples to `/root/tensorflow_datasets/cnn_dailymail/plain_text/3.0.0.incomplete7W6N6S/cnn_dail`

46%   6101/13368 [00:00<00:00, 61009.33 examples/s]

Shuffling and Saving the Data



Sharded Data



Info File



- ▼ root
 - ▼ tensorflow_datasets
 - ▼ cnn_dailymail
 - ▼ plain_text
 - ▼ 1.0.0
 - cnndailymail-test.tfrecord-00000-of-00001
 - cnndailymail-train.tfrecord-00000-of-00016
 - cnndailymail-train.tfrecord-00001-of-00016
 - cnndailymail-train.tfrecord-00002-of-00016
 - cnndailymail-train.tfrecord-00003-of-00016
 - cnndailymail-train.tfrecord-00004-of-00016
 - cnndailymail-train.tfrecord-00005-of-00016
 - cnndailymail-train.tfrecord-00006-of-00016
 - cnndailymail-train.tfrecord-00007-of-00016
 - cnndailymail-train.tfrecord-00008-of-00016
 - cnndailymail-train.tfrecord-00009-of-00016
 - cnndailymail-train.tfrecord-00010-of-00016
 - cnndailymail-train.tfrecord-00011-of-00016
 - cnndailymail-train.tfrecord-00012-of-00016
 - cnndailymail-train.tfrecord-00013-of-00016
 - cnndailymail-train.tfrecord-00014-of-00016
 - cnndailymail-train.tfrecord-00015-of-00016
 - cnndailymail-validation.tfrecord-00000-of-00001
 - dataset_info.json

- ▼ root
 - ▼ tensorflow_datasets
 - ▼ cnn_dailymail
 - ▼ plain_text
 - ▼ 1.0.0
 - ▼ cnn_dailymail-test.tfrecord-00000-of-00001
 - ▼ cnn_dailymail-train.tfrecord-00000-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00001-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00002-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00003-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00004-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00005-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00006-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00007-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00008-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00009-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00010-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00011-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00012-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00013-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00014-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00015-of-00016
 - ▼ cnn_dailymail-validation.tfrecord-00000-of-00001
 - ▼ dataset_info.json

- ▼ root
 - ▼ tensorflow_datasets
 - ▼ cnn_dailymail
 - ▼ plain_text
 - ▼ 1.0.0
 - cnn_dailymail-test.tfrecord-00000-of-00001
 - cnn_dailymail-train.tfrecord-00000-of-00016
 - cnn_dailymail-train.tfrecord-00001-of-00016
 - cnn_dailymail-train.tfrecord-00002-of-00016
 - cnn_dailymail-train.tfrecord-00003-of-00016
 - cnn_dailymail-train.tfrecord-00004-of-00016
 - cnn_dailymail-train.tfrecord-00005-of-00016
 - cnn_dailymail-train.tfrecord-00006-of-00016
 - cnn_dailymail-train.tfrecord-00007-of-00016
 - cnn_dailymail-train.tfrecord-00008-of-00016
 - cnn_dailymail-train.tfrecord-00009-of-00016
 - cnn_dailymail-train.tfrecord-00010-of-00016
 - cnn_dailymail-train.tfrecord-00011-of-00016
 - cnn_dailymail-train.tfrecord-00012-of-00016
 - cnn_dailymail-train.tfrecord-00013-of-00016
 - cnn_dailymail-train.tfrecord-00014-of-00016
 - cnn_dailymail-train.tfrecord-00015-of-00016
 - cnn_dailymail-validation.tfrecord-00000-of-00001
 - dataset_info.json

- ▼ root
 - ▼ tensorflow_datasets
 - ▼ cnn_dailymail
 - ▼ plain_text
 - ▼ 1.0.0
 - ▼ cnn_dailymail-test.tfrecord-00000-of-00001
 - ▼ cnn_dailymail-train.tfrecord-00000-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00001-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00002-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00003-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00004-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00005-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00006-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00007-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00008-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00009-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00010-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00011-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00012-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00013-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00014-of-00016
 - ▼ cnn_dailymail-train.tfrecord-00015-of-00016
 - ▼ cnn_dailymail-validation.tfrecord-00000-of-00001
 - ▼ dataset_info.json

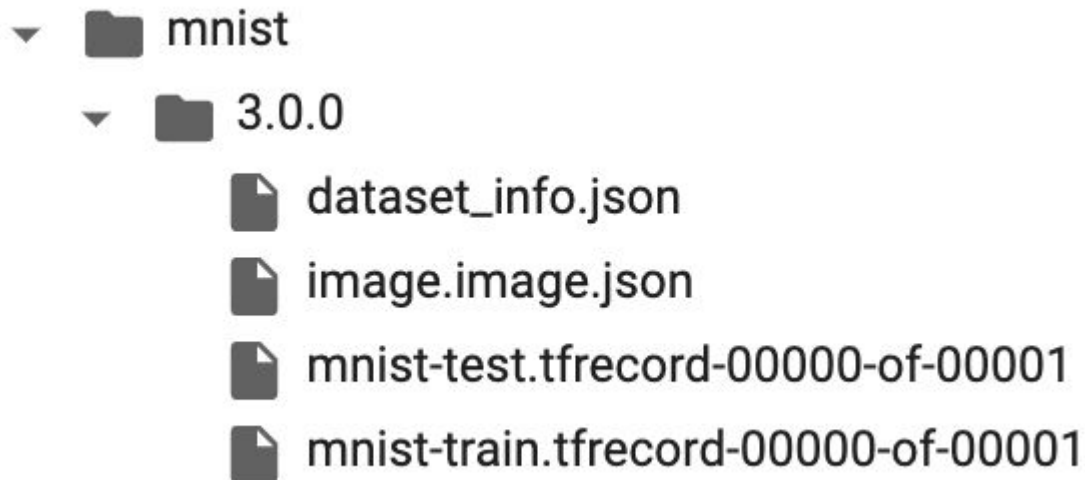
```
data, info = tfds.load("mnist", with_info=True)
print(info)
```

```
tfds.core.DatasetInfo(  
    name='mnist',  
    version=3.0.0,  
    description='The MNIST database of handwritten digits.',  
    homepage='http://yann.lecun.com/exdb/mnist/',  
    features=FeaturesDict({  
        'image': Image(shape=(28, 28, 1), dtype=tf.uint8),  
        'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=10),  
    }),  
    total_num_examples=70000,  
    splits={  
        'test': 10000,  
        'train': 60000,  
    },  
    ...  
)
```

```
tfds.core.DatasetInfo(  
    name='mnist',  
    version=3.0.0,  
    description='The MNIST database of handwritten digits.',  
    homepage='http://yann.lecun.com/exdb/mnist/',  
    features=FeaturesDict({  
        'image': Image(shape=(28, 28, 1), dtype=tf.uint8),  
        'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=10),  
    }),  
    total_num_examples=70000,  
    splits={  
        'test': 10000,  
        'train': 60000,  
    },  
    ...  
)
```

```
features=FeaturesDict({  
    'image': Image(shape=(28, 28, 1), dtype=tf.uint8),  
    'label': ClassLabel(shape=(), dtype=tf.int64,  
                          num_classes=10),  
}),
```

`/root/tensorflow_datasets/mnist/<version>/files`



```
{"encoding_format": "png", "shape": [28, 28, 1]}
```




```
filename="/root/tensorflow_datasets/mnist/3.0.0/  
mnist-test.tfrecord-00000-of-00001"
```

```
raw_dataset = tf.data.TFRecordDataset(filename)
```

```
for raw_record in raw_dataset.take(1):  
    print(repr(raw_record))
```

```
filename="/root/tensorflow_datasets/mnist/3.0.0/  
mnist-test.tfrecord-00000-of-00001"
```

```
raw_dataset = tf.data.TFRecordDataset(filename)
```

```
for raw_record in raw_dataset.take(1):  
    print(repr(raw_record))
```

```
filename="/root/tensorflow_datasets/mnist/3.0.0/  
mnist-test.tfrecord-00000-of-00001"
```

```
raw_dataset = tf.data.TFRecordDataset(filename)
```

```
for raw_record in raw_dataset.take(1):  
    print(repr(raw_record))
```

```
filename="/root/tensorflow_datasets/mnist/3.0.0/  
mnist-test.tfrecord-00000-of-00001"
```

```
raw_dataset = tf.data.TFRecordDataset(filename)
```

```
for raw_record in raw_dataset.take(1):  
    print(repr(raw_record))
```



```
# Create a description of the features.  
feature_description = {  
    'image': tf.io.FixedLenFeature([], dtype=tf.string),  
    'label': tf.io.FixedLenFeature([], dtype=tf.int64),  
}
```

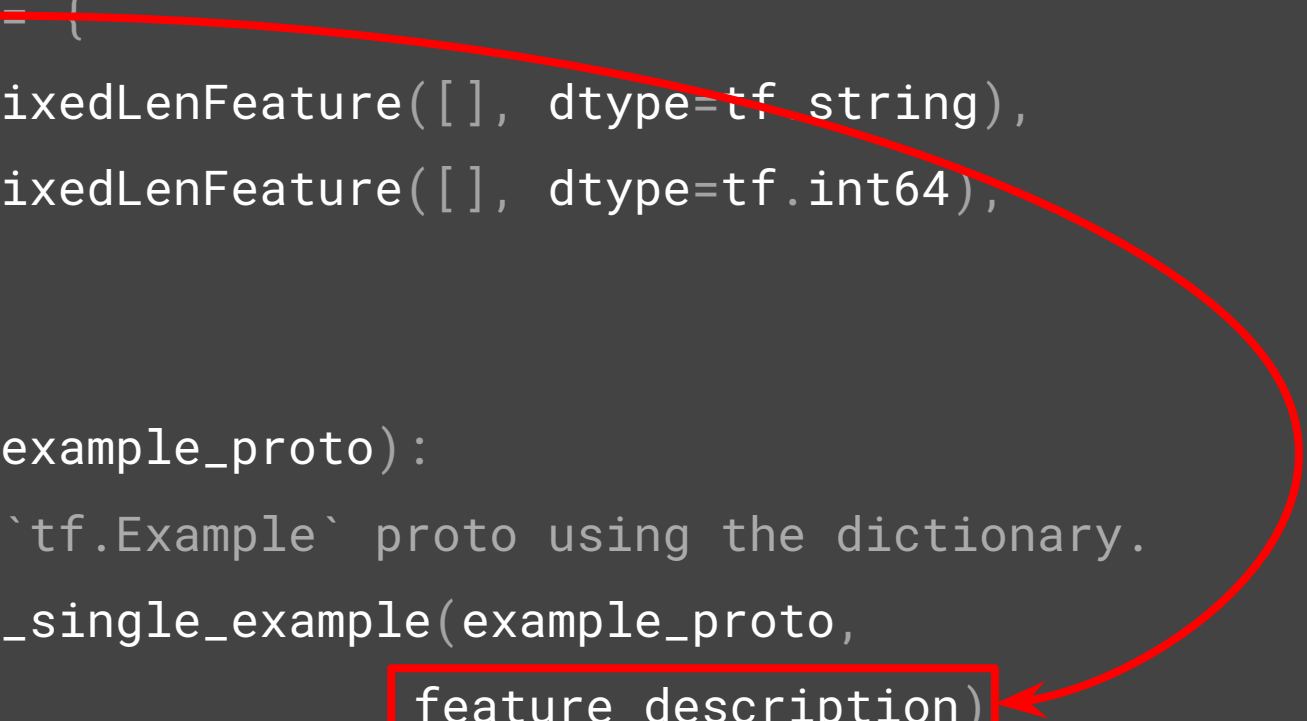
```
# Create a description of the features.
```

```
feature_description = {  
    'image': tf.io.FixedLenFeature([], dtype=tf.string),  
    'label': tf.io.FixedLenFeature([], dtype=tf.int64),  
}
```

```
def _parse_function(example_proto):
```

```
    # Parse the input `tf.Example` proto using the dictionary.
```

```
    return tf.io.parse_single_example(example_proto,  
                                     feature_description)
```




```
parsed_dataset = raw_dataset.map(_parse_function)
```

```
for parsed_record in parsed_dataset.take(1):  
    print((parsed_record))
```

```
parsed_dataset = raw_dataset.map(_parse_function)
```

```
for parsed_record in parsed_dataset.take(1):  
    print((parsed_record))
```

```
parsed_dataset = raw_dataset.map(_parse_function)
```

```
for parsed_record in parsed_dataset.take(1):  
    print((parsed_record))
```

```
{ 'image': <tf.Tensor: shape=(), dtype=string,
```

```
numpy=b"\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x00\x1c\x00\x00\x00\x1c\x08\x00\x00\x00\x00Wf\x80H\x00\x00\x01)IDAT(\x91\xc5\xd2\xbdK\xc3P\x14\x05\xf0S(v\x13)\x04,.\x82\xc5Aq\xac\xedb\x1d\xdc\n.\x12\x87n\x0e\x82\x93\x7f@Q\xb2\x08\xba\tbQ0.\xe2\xe2\xd4\xb1\xa2h\x9c\x82\xba\x8a(\nq\xf0\x83Fh\x95\n6\x88\xe7R\x87\x88\xf9\xa8Y\x05\x0e\x8f\xc7\xfd\xdd\x0b\x87\xc7\x03\xfe\xbeb\x9d\xadT\x927Q\xe3\xe9\x07:\xab\xbf\xf4\xf3\xcf\xf6\x8a\xd9\x14\xd29\xea\xb0\x1eKH\xde\xab\xea%\xabax1b=\xa4P/\xf5\x02\xd7\\\x07\x00\xc4=,L\xc0,>\x01@2\xf6\x12\xde\x9c\xde[t/\xb3\x0e\x87\xa2\xe2\xc2\xe0A<\xca\xb26\xd5(\x1b\xa9\xd3\xe8\x0e\xf5\x86\x17\xceE\xdarV\xae\xb7_\xf3AR\r!I\xf7(\x06m\xaaE\xbb\xb6\xac\r*\x9b$e<\xb8\xd7\xa2\x0e\x00\xd0l\x92\xb2\xd5\x15\xcc\xae'\x00\xf4m\x080'+\xc2y\x9f\x8d\xc9\x15\x80\xfe\x99[q\x962@CN|i\xf7\xa9!=\xd7\xab\x19\x00\xc8\xd6\xb8\xeb\xa1\xf0\xd8l\xca\xfb]\xee\xfb]*\x9fV\xe1\x07\xb7\xc9\x8b55\xe7M\xef\xb0\x04\xc0\xfd&\x89\x01<\xbe\xf9\x03*\x8a\xf5\x81\x7f\xaa/2y\x87ks\xec\x1e\xc1\x00\x00\x00\x00IEND\xaeB`\x82">,

```

```
'label': <tf.Tensor: shape=(), dtype=int64, numpy=2>}
```