Example here: <https://www.tensorflow.org/datasets/keras_example>  
Documentation for ds here: <https://www.tensorflow.org/api_docs/python/tf/data/Dataset>

Notebook here: <https://colab.research.google.com/drive/11uXaGV0Xc9YO1OuUTZSPYSSIfnd4UE-I?usp=sharing>

tdfs.load(dataset\_name, split,shuffle\_files..)

* If split = [‘train’,’test’] then you will say: ds\_train, ds\_test = tfds.load(‘mnist’,split=[‘train’,’test’]
* Shuffle\_files = True to shuffle the dataset
* With\_info = True then you will do: ds, ds\_info = tfds.load(‘mnist’, with\_info=True)

(ds\_train, ds\_test) , info = tfds.load('mnist', shuffle\_files=True, as\_supervised=True, with\_info= True, split=['train','test'])

To iterate over a dataset: <https://www.tensorflow.org/datasets/overview#iterate_over_a_dataset>

* Based on as\_supervised parameter for tfds.load, the way you iterate is slightly different (if True, then the returned object is a tuple of (feature,label). If False, then the returned object is a dictionary of {‘image’: tf.Tensor , ‘label’: tf.Tensor}
* Always starts with ds.take(num) where num is the num of examples to iterate. You must always take the examples before you can access its properties (the image and label)

ds will be the dataset obtained from tfds.load()

ds.map(function, num\_parallel\_calls) will apply the function to all elements in ds (can apply a function that preprocesses all the data; reshape, normalise, etc)

def preprocess(feature, label):

feature = tf.cast(feature, tf.float32) / 255.

return feature, label

ds\_train.map(preprocess,num\_parallel\_calls=tf.data.experimental.AUTOTUNE)

ds.cache() for better training performance (As in the time to train, not the acc)

ds.shuffle(buffer\_size) to shuffle all elements in ds. Best to use a buffer\_size >= no. of examples in ds; can be obtained from ds\_info.splits[‘train’].num\_examples

ds.batch(batch\_size)

ds.prefetch(buffer\_size) will lead to faster training as later elements will be prepared while current element is being processed. (Can use AUTOTUNE here)

tf.data.experimental.AUTOTUNE will pick the optimal value. (Can use for prefetch, num\_parallel\_calls, etc.)

Must do tf\_train = tf\_train.map(...).

| For training data:   * Map -- cache -- shuffle -- batch -- prefetch   ds\_train = ds\_train.map(preprocess,num\_parallel\_calls=tf.data.experimental.AUTOTUNE)  ds\_train = ds\_train.cache()  ds\_train = ds\_train.shuffle(info.splits['train'].num\_examples)  ds\_train = ds\_train.batch(128)  ds\_train = ds\_train.prefetch(tf.data.experimental.AUTOTUNE)  For testing data   * Map -- batch -- cache -- prefetch   ds\_test = ds\_test.map(preprocess, num\_parallel\_calls=tf.data.experimental.AUTOTUNE)  ds\_test = ds\_test.batch(128)  ds\_test = ds\_test.cache()  ds\_test = ds\_test.prefetch(tf.data.experimental.AUTOTUNE) |
| --- |