Features

Understand how data features are represented in TensorFlow.

Chapter Goals:

- Learn what an Example spec is and what it is used for
- Understand the two different feature configuration classes in TensorFlow
- Implement a function that creates an Example spec from a feature configuration

A. Example spec

When we read from a TFRecords file for our input pipeline, we get back the serialized versions of the protocol buffers. What we really want, though, are easily accessible data values for each feature from the original protocol buffer. In order to achieve this, we need an *Example spec*.

An Example spec is a dictionary that maps the feature names from the original `tf.train.Example` protocol buffer to particular configuration classes for each feature. Specifically, a feature is in exactly one of two configuration classes: `VarLenFeature` or `FixedLenFeature`.

B. `VarLenFeature` class

A `VarLenFeature` specifies a feature whose data value list has variable length, meaning each protocol buffer can have a different number of values for the feature.

```
import tensorflow as tf

# Two people represented by the same protocol buffer
# "jobs" is a VarLenFeature
print(repr(person1))  # 1 job
print(repr(person2))  # 2 jobs
```

In the above code example, the `"jobs"` feature is a `VarLenFeature`. For `person1`, the data value list has length 1, but for person2 the data value list has length 2.

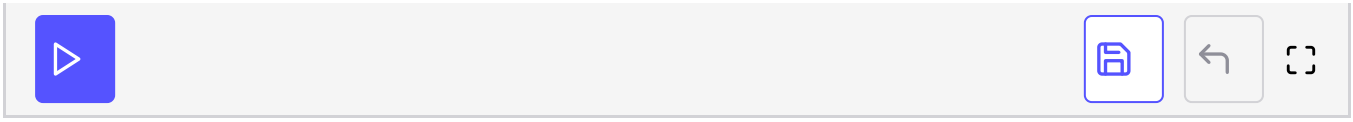
Other common features that fall into the `VarLenFeature` class are text features (e.g. articles with different word counts), features that depend on longevity (e.g. salary history for employees at a company), and features that allow optional or incomplete values.

C. `FixedLenFeature` class

In contrast with `VarLenFeature`, a `FixedLenFeature` has a fixed length for its data value list. This means each protocol buffer should have exactly the same number of values for a feature in this class.

```
import tensorflow as tf

# Two people represented by the same protocol buffer
# "name" and "salary" are both FixedLenFeature
print(repr(person1))
print(repr(person2))
```



In this example, both the `"name"` and `"salary"` features are part of the `FixedLenFeature` configuration class. The `"name"` feature has value length 1 while the `"salary"` feature has value length 2 (hourly and monthly salaries).

Other common features that fall into the `FixedLenFeature` class are statistics over a fixed time range, features that take single values (e.g. name, birthplace, etc.), and rigidly structured features.

D. Class configuration

When we use either a `VarLenFeature` or `FixedLenFeature`, we always have to set the TensorFlow datatype. For the Example spec, this datatype will either be `tf.int64`, `tf.float32`, or `tf.string`.

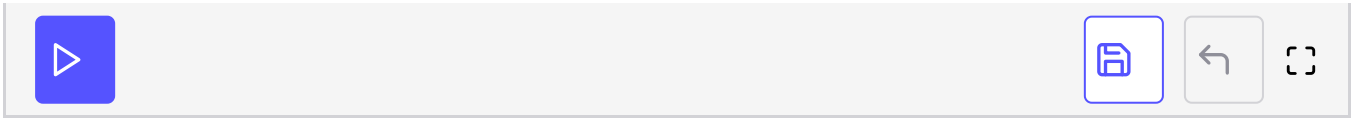
In addition to the datatype, the `FixedLenFeature` class also takes in the fixed shape and an optional default value. The default value is used in place of any

missing values in the feature (e.g. if the shape specifies 4 values but the protocol buffer only has 2).

```
import tensorflow as tf

name = tf.FixedLenFeature((), tf.string)
jobs = tf.VarLenFeature(tf.string)
salary = tf.FixedLenFeature(2, tf.int64, default_value=0)
example_spec = {
    'name': name,
    'jobs': jobs,
    'salary': salary
}

print(example_spec)
```



In the code example above, we created an Example spec that contains two `FixedLenFeature` features and one `VarLenFeature` feature. The `VarLenFeature` class takes in the datatype as its only argument. The `FixedLenFeature` class takes in the shape as the first argument, datatype as the second argument, and optional default value as the third argument.

Note that the `()` represents a single value shape. This actually differs from using `1` as the shape argument, for reasons we'll discuss in upcoming chapters.

Time to Code!

In this chapter you'll be completing the `create_example_spec` function. The function creates an Example spec from a configuration dictionary `config`, which maps feature names to their configurations (similar to the one used in `dict_to_example` from chapter 2).

You'll specifically be creating a helper function called `make_feature_config` which returns either a `VarLenFeature` or `FixedLenFeature` depending on the `shape` of the feature.

Create an `if...else` block where the `if` condition checks that `shape` is `None`.

If the feature's shape is `None`, that means we don't know the shape. In other words, the feature can be of variable length. In this case, we use `VarLenFeature` in our Example spec.

In the `if` block, set `feature` equal to `tf.VarLenFeature` with `tf_type` as the only argument.

If instead the feature's shape is given, it means the feature has a fixed length. Therefore, we use `FixedLenFeature`. A `FixedLenFeature` can also have a default value, which will be specified in `feature_config` if the key `'default_value'` is present.

In the `else` block, set `default_value` equal to the value in `feature_config` mapped to by `'default_value'`. If the key is not present, set `default_value` equal to `None`.

Then set `feature` equal to `tf.FixedLenFeature` with `shape`, `tf_type`, and `default_value` as the arguments (in that order).

Outside the `if...else` block, return `feature`.

```
import tensorflow as tf

def make_feature_config(shape, tf_type, feature_config):
    # CODE HERE
    pass

def create_example_spec(config):
    example_spec = {}
    for feature_name, feature_config in config.items():
        if feature_config['type'] == 'int':
            tf_type = tf.int64
        elif feature_config['type'] == 'float':
            tf_type = tf.float32
        else:
            tf_type = tf.string
        shape = feature_config['shape']
        feature = make_feature_config(shape, tf_type, feature_config)
        example_spec[feature_name] = feature
    return example_spec
```