# 13.2 TFRecord 포맷

- 크기가 다른 연속된 이진 레코드를 저장하는 단순 이진 포맷
- 레코드 길이, 체크섬, 실제데이터, 데이터체크섬으로 구성
- 체크섬(길이가 올바른지 체크하는 것)
- tf.io.TFRecordWriter 클래스를 사용해 생성할 수 있다.

```
In [42]: with tf.io.TFRecordWriter("my_data.tfrecord") as f:
             f.write(b"This is the first record")
             f.write(b"And this is the second record")
```

- tf.data.TFRecordDataset을 사용해 하나 이상의 TFRecord 를 읽을 수 있다.

```
In [43]: filepaths = ["my_data.tfrecord"]
         dataset = tf.data.TFRecordDataset(filepaths)
         for item in dataset:
             print(item)
```

```
tf.Tensor(b'This is the first record', shape=(), dtype=string)
tf.Tensor(b'And this is the second record', shape=(), dtype=string)
```

## 13.2.1 압축된 TFRecord 파일

- 네트워크를 통해 읽어야 하는 경우 TFRecord 파일을 압축할 필요가 있다
- options 매개변수를 사용해 압축된 TFRecord 파일을 만들 수 있다.

```
In [45]: options = tf.io.TFRecordOptions(compression_type="GZIP")
         with tf.io.TFRecordWriter("my_compressed.tfrecord", options) as f:
             f.write(b"This is the first record")
             f.write(b"And this is the second record")
```

- 압축된 TFRecord  파일을 읽으려면 압축 형식을 지정해야 한다.

```
In [46]: dataset = tf.data.TFRecordDataset(["my_compressed.tfrecord"],
                                            compression_type="GZIP")
         for item in dataset:
             print(item)
```

```
tf.Tensor(b'This is the first record', shape=(), dtype=string)
tf.Tensor(b'And this is the second record', shape=(), dtype=string)
```

## 13.2.2 프로토콜 버퍼 개요

- 각 TFRecord 는 직렬화된 프로토콜 버퍼를 담고 있다.
- 프로토콜 버퍼는 2001년 구글이 개발, 2008 오픈소스로 공개
- 이식성과 확장성이 좋고 효율적이다

- 현재는 구글의 원격 프로시저 호출 시스템인 gRPC에 사용됨

```
In [47]: %%writefile person.proto
         syntax = "proto3";
         message Person {
           string name = 1;
           int32 id = 2;
           repeated string email = 3;
         }
         Overwriting person.proto
```

- 문법 포맷은 proto3를 사용
- Person 객체는 string 타입의 name, int 타입의 id, 하나 이상의 string 타입의 email을 갖는다는 의미
- 1, 2, 3은 필드 식별자

```
In [48]: !protoc person.proto --python_out=. --descriptor_set_out=person.desc --include_imports

In [49]: !ls person*
         person.desc   person.proto  person_pb2.py
```

- person.desc | person.proto | person_pb2.py 파일 생성

```
In [50]: from person_pb2 import Person

         person = Person(name="Al", id=123, email=["a@b.com"])  # create a Person
         print(person)  # display the Person

         name: "Al"
         id: 123
         email: "a@b.com"

In [51]: person.name  # read a field
Out[51]: 'Al'

In [52]: person.name = "Alice"  # modify a field

In [53]: person.email[0]  # repeated fields can be accessed like arrays
Out[53]: 'a@b.com'

In [54]: person.email.append("c@d.com")  # add an email address

In [55]: s = person.SerializeToString()  # serialize to a byte string
         s
Out[55]: b'\n\x05Alice\x10{\x1a\x07a@b.com\x1a\x07c@d.com'

In [56]: person2 = Person()  # create a new Person
         person2.ParseFromString(s)  # parse the byte string (27 bytes)
Out[56]: 27

In [57]: person == person2  # now they are equal
Out[57]: True
```

## 13.2.3 텐서플로 프로토콜 버퍼

프로토콜 버퍼 정의

```
syntax = "proto3";

message BytesList { repeated bytes value = 1; }  # byte list
message FloatList { repeated float value = 1 [packed = true]; }  # float list
message Int64List { repeated int64 value = 1 [packed = true]; }  # int list
# packed는 효율적인 코딩을 위한 반복적 수치 필드에 사용됨

message Feature {
    oneof kind {
        BytesList bytes_list = 1;
        FloatList float_list = 2;
        Int64List int64_list = 3;
    }
};
message Features { map<string, Feature> feature = 1; };  #{'name':feature} 형태
message Example { Features features = 1; };  # 하나의 features 객체
```

프로토콜 버퍼를 사용한 데이터 저장

```
from tensorflow.train import BytesList, FloatList, Int64List
from tensorflow.train import Feature, Features, Example

BytesList = tf.train.BytesList
FloatList = tf.train.FloatList
Int64List = tf.train.Int64List
Feature = tf.train.Feature
Features = tf.train.Features
Example = tf.train.Example

person_example = Example(
    features=Features(
        feature={
            "name": Feature(bytes_list=BytesList(value=[b"Alice"])),
            "id": Feature(int64_list=Int64List(value=[123])),
            "emails": Feature(bytes_list=BytesList(value=[b"a@b.com",
b"c@d.com"]))
        }))

with tf.io.TFRecordWriter("my_contacts.tfrecord") as f:  # 데이터 저장
    f.write(person_example.SerializeToString())
```

## 13.2.3 Example 프로토콜 버퍼를 읽고 파싱하기

```
feature_description = {
    "name": tf.io.FixedLenFeature([], tf.string, default_value=""),
    "id": tf.io.FixedLenFeature([], tf.int64, default_value=0),
    "emails": tf.io.VarLenFeature(tf.string),
}
for serialized_example in tf.data.TFRecordDataset(["my_contacts.tfrecord"]):
    parsed_example = tf.io.parse_single_example(serialized_example,
feature_description)
```

```
In [61]: parsed_example
Out[61]: {'emails': <tensorflow.python.framework.sparse_tensor.SparseTensor at 0x7fa77094b490>,
          'id': <tf.Tensor: shape=(), dtype=int64, numpy=123>,
          'name': <tf.Tensor: shape=(), dtype=string, numpy=b'Alice'>}

In [63]: parsed_example["emails"].values[0]
Out[63]: <tf.Tensor: shape=(), dtype=string, numpy=b'a@b.com'>

In [64]: tf.sparse.to_dense(parsed_example["emails"], default_value=b"")
Out[64]: <tf.Tensor: shape=(2,), dtype=string, numpy=array([b'a@b.com', b'c@d.com'], dtype=object)>

In [65]: parsed_example["emails"].values
Out[65]: <tf.Tensor: shape=(2,), dtype=string, numpy=array([b'a@b.com', b'c@d.com'], dtype=object)>
```

- ByteList는 직렬화된 객체를 포함해 어떤 이진 데이터도 포함할 수 있다.(이미지 등)
- jpeg 이미지 인코딩 시에는 `tf.io.encode_jpeg()`, 디코드는 `tf.io.decode_jpeg()` 를 사용한다.

## 13.2.4 SequenceExample 프로토콜 버퍼를 사용해 리스트의 리스트 다루기

프로토콜 버퍼 정의

```
syntax = "proto3";

message FeatureList { repeated Feature feature = 1; };
message FeatureLists { map<string, FeatureList> feature_list = 1; };
message SequenceExample {
  Features context = 1;
  FeatureLists feature_lists = 2;
};
```

프로토콜 버퍼를 사용한 데이터 저장

```
from tensorflow.train import FeatureList, FeatureLists, SequenceExample
FeatureList = tf.train.FeatureList
FeatureLists = tf.train.FeatureLists
SequenceExample = tf.train.SequenceExample

context = Features(feature={
    "author_id": Feature(int64_list=Int64List(value=[123])),
    "title": Feature(bytes_list=BytesList(value=[b"A", b"desert", b"place",
b"."])),
    "pub_date": Feature(int64_list=Int64List(value=[1623, 12, 25]))
})

content = [["When", "shall", "we", "three", "meet", "again", "?"],
           ["In", "thunder", ",", "lightning", ",", "or", "in", "rain", "?"]]
comments = [["When", "the", "hurlyburly", "'s", "done", "."],
            ["When", "the", "battle", "'s", "lost", "and", "won", "."]]

def words_to_feature(words):
    return Feature(bytes_list=BytesList(value=[word.encode("utf-8")
                                               for word in words]))

content_features = [words_to_feature(sentence) for sentence in content]
comments_features = [words_to_feature(comment) for comment in comments]

sequence_example = SequenceExample(
    context=context,
    feature_lists=FeatureLists(feature_list={
        "content": FeatureList(feature=content_features),
        "comments": FeatureList(feature=comments_features)
    }))
```

출력결과

```
sequence_example
```

```
context {
  feature {
    key: "author_id"
    value {
      int64_list {
        value: 123
      }
    }
  }
  feature {
    key: "pub_date"
    value {
      int64_list {
        value: 1623
        value: 12
        value: 25
      }
    }
  }
  feature {
    key: "title"
    value {
      bytes_list {
        value: "A"
        value: "desert"
        value: "place"
        value: "."
      }
    }
  }
}
feature_lists {
  feature_list {
    key: "comments"
    value {
      feature {
        bytes_list {
          value: "When"
          value: "the"
          value: "hurlyburly"
          value: "\'s"
          value: "done"
          value: "."
        }
      }
      feature {
        bytes_list {
          value: "When"
          value: "the"
          value: "battle"
          value: "\'s"
          value: "lost"
          value: "and"
          value: "won"
          value: "."
        }
      }
```

```
    }
  }
  feature_list {
    key: "content"
    value {
      feature {
        bytes_list {
          value: "When"
          value: "shall"
          value: "we"
          value: "three"
          value: "meet"
          value: "again"
          value: "?"
        }
      }
      feature {
        bytes_list {
          value: "In"
          value: "thunder"
          value: ","
          value: "lightning"
          value: ","
          value: "or"
          value: "in"
          value: "rain"
          value: "?"
        }
      }
    }
  }
}
```