Copyright Notice

Under the hood

Dataset Info

Feature Generation
Split datasets

Example Generation

Dataset Info
Split Generators
Generate Examples

Under the hood

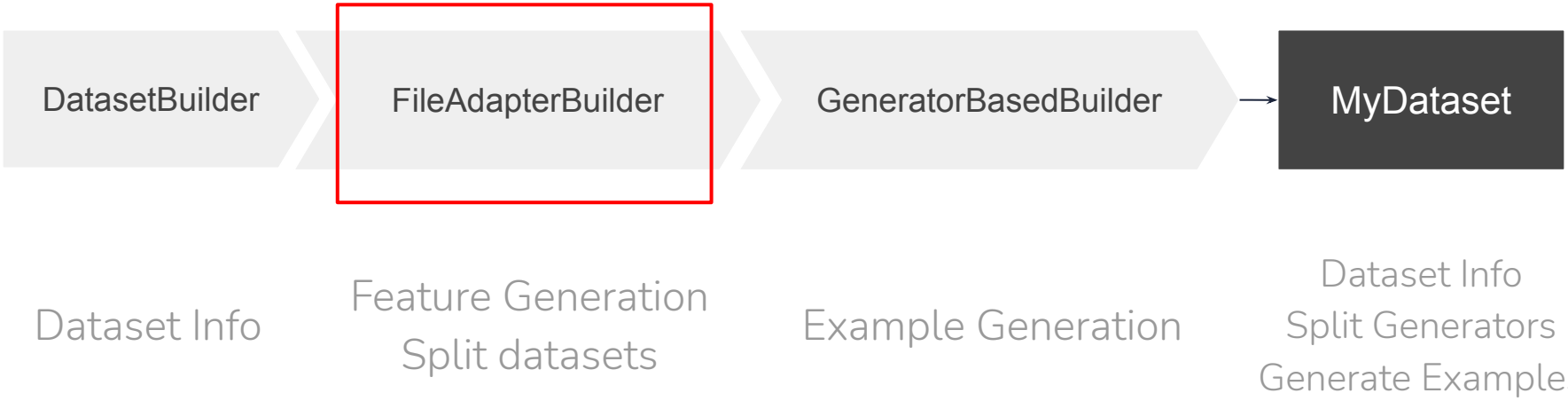
Dataset Info

Feature Generation
Split datasets

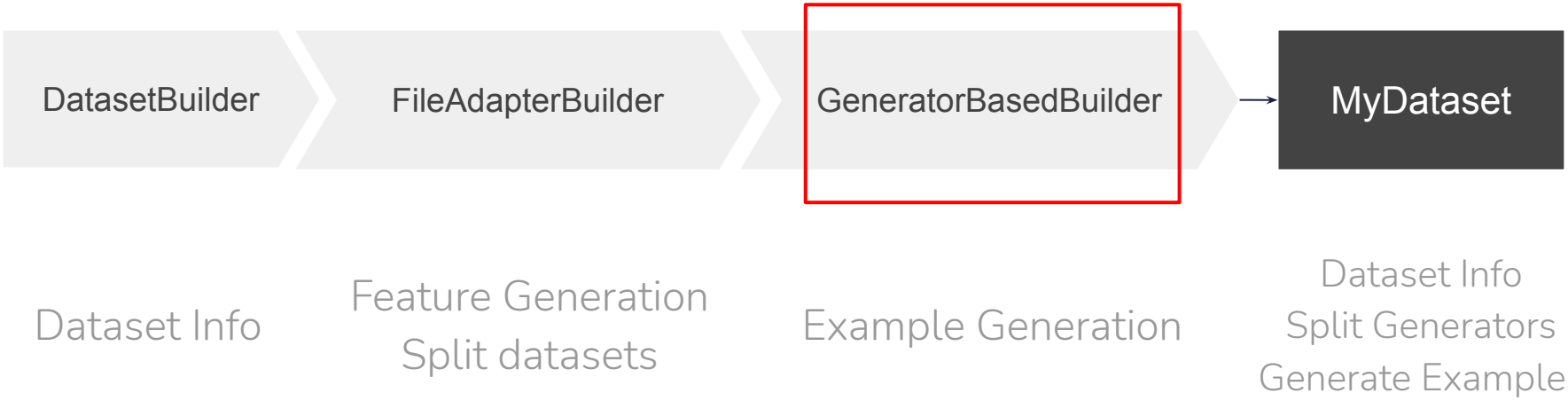
Example Generation

Dataset Info
Split Generators
Generate Examples

Under the hood



Under the hood



Getting started

```
# Clone the tensorflow-datasets repository
git clone https://github.com/tensorflow/datasets.git

tensorflow_datasets/scripts/create_new_dataset.py \
  --dataset my_dataset \
  --type image  # text, audio, translation,...
```

Getting started

```
# Clone the tensorflow-datasets repository
git clone https://github.com/tensorflow/datasets.git

tensorflow_datasets/scripts/create_new_dataset.py \
  --dataset my_dataset \
  --type image  # text, audio, translation,...
```

Getting started

```
# Clone the tensorflow-datasets repository
git clone https://github.com/tensorflow/datasets.git
```

```
tensorflow_datasets/scripts/create_new_dataset.py \
  --dataset my_dataset \
  --type image  # text, audio, translation,...
```

```python
class MyDataset(tfds.core.GeneratorBasedBuilder):
  VERSION = tfds.core.Version('0.1.0')
  def _info(self):
    return tfds.core.DatasetInfo(builder=self, description=..., features=...,
                                 supervised_keys=..., urls=..., citation=...)

  def _split_generators(self, dl_manager):

  def _generate_examples(self):
    yield 'key', {}
```

```python
class MyDataset(tfds.core.GeneratorBasedBuilder):
  VERSION = tfds.core.Version('0.1.0')
  def _info(self):
    return tfds.core.DatasetInfo(builder=self, description=..., features=...,
                                 supervised_keys=..., urls=..., citation=...)

  def _split_generators(self, dl_manager):

  def _generate_examples(self):
    yield 'key', {}
```

```python
class MyDataset(tfds.core.GeneratorBasedBuilder):
  VERSION = tfds.core.Version('0.1.0')
  def _info(self):
    return tfds.core.DatasetInfo(builder=self, description=..., features=...,
                                 supervised_keys=..., urls=..., citation=...)

  def _split_generators(self, dl_manager):

  def _generate_examples(self):
    yield 'key', {}
```

```python
class MyDataset(tfds.core.GeneratorBasedBuilder):
  VERSION = tfds.core.Version('0.1.0')
  def _info(self):
    return tfds.core.DatasetInfo(builder=self, description=..., features=...,
                                 supervised_keys=..., urls=..., citation=...)

  def _split_generators(self, dl_manager):

  def _generate_examples(self):
    yield 'key', {}
```

```
def _info(self):
  return tfds.core.DatasetInfo(
    builder=self,description=("INSERT DESCRIPTION HERE"),
    features=tfds.features.FeaturesDict({
          "description": tfds.features.Text(),
          "image": tfds.features.Image(),
          "label": tfds.features.ClassLabel(num_classes=5),
    }),

      supervised_keys=("image", "label"),
      urls=["https://dataset-homepage.org"],
      citation=r"""@article{my-awesome-dataset-2020,
                   ...
                   author = {Smith, John},"}""")
```

```
def _info(self):
  return tfds.core.DatasetInfo(
      builder=self,
      description="A large set of images of horses and humans.",
      features=tfds.features.FeaturesDict({
          "image": tfds.features.Image(shape=_IMAGE_SHAPE),
          "label": tfds.features.ClassLabel(
              names=["horses", "humans"]),
      }),
      supervised_keys=("image", "label"),
      urls=["http://laurencemoroney.com/horses-or-humans-dataset"],
      citation=_CITATION
      )
```

```
def _split_generators(self, dl_manager):
  # Equivalent to dl_manager.extract(dl_manager.download(urls))
  dl_paths = dl_manager.download_and_extract({
      'foo': 'https://example.com/foo.zip',
      'bar': 'https://example.com/bar.zip',
  })
  dl_paths['foo'], dl_paths['bar']
```

```
def _split_generators(self, dl_manager):
  """Returns SplitGenerators."""
  # TODO(my_dataset): Downloads the data and defines the splits
  # dl_manager is a tfds.download.DownloadManager that can be used to
  # download and extract URLs
  return [
      tfds.core.SplitGenerator(
          name=tfds.Split.TRAIN,
          # These kwargs will be passed to _generate_examples
          gen_kwargs={},
      ),
    ]
```

```
def _split_generators(self, dl_manager):
  return [tfds.core.SplitGenerator(
              name=tfds.Split.TRAIN,
              gen_kwargs={
                  "dir_path": os.path.join(extracted_path, "train"),
                  "labels": os.path.join(extracted_path, "train_labels.csv")}),
          tfds.core.SplitGenerator(
              name=tfds.Split.TEST,
              gen_kwargs={
                  "dir_path": os.path.join(extracted_path, "test"),
                  "labels": os.path.join(extracted_path, "test_labels.csv")})
        ]
```

```
def _split_generators(self, dl_manager):
    train_path, test_path = dl_manager.download([_TRAIN_URL, _TEST_URL])

    return [
        tfds.core.SplitGenerator(
            name=tfds.Split.TRAIN,
            num_shards=10,
            gen_kwargs={
                "archive": dl_manager.iter_archive(train_path)
            }),
        tfds.core.SplitGenerator(
            name=tfds.Split.TEST,
            num_shards=10,
            gen_kwargs={
                "archive": dl_manager.iter_archive(test_path)
            }),
]
```

```
def _generate_examples(self):
  """Yields examples."""
  # TODO(my_dataset): Yields (key, example) tuples from the dataset
  yield 'key', {}
```

```
def _generate_examples(self, archive):
  for fname, fobj in archive:
    res = _NAME_RE.match(fname)
    if not res:  # if anything other than .png; skip
      continue
    label = res.group(1).lower()
    record = {
        "image": fobj,
        "label": label,
    }
    yield fname, record
```

```
builder._generate_examples(
    images_dir_path="{extracted_path}/train",
    labels="{extracted_path}/train_labels.csv",
)
```

File access

- Use `tf.io.gfile` or `tf.python_io` to support Cloud Storage systems
- **Avoid** using **Python built-ins**
  - e.g., `open`, `os.rename`, `gzip`

Extra dependencies

- Use `tfds.core.lazy_imports`
- Add a lazy import with an entry in `DATASET_EXTRAS`
- Install with

```
pip install tensorflow-datasets[<dataset-name>]
```

Problems in data

- Corrupted data
  - e.g., invalid image formats
- Inconsistent data
  - **Solution**: Mark dataset as unstable by adding

    A class constant - `UNSTABLE` in `DatasetBuilder`

When to use configurations

Heavy

- Specifies how data needs to be written to the disk
- Different `DatasetInfo` setups
- When changing access for download data
- Use `tfds.core.BuilderConfigs` to configure data generation

Light

- Deals with runtime preprocessing
- tf.data input pipelines
- Perform additional transformations

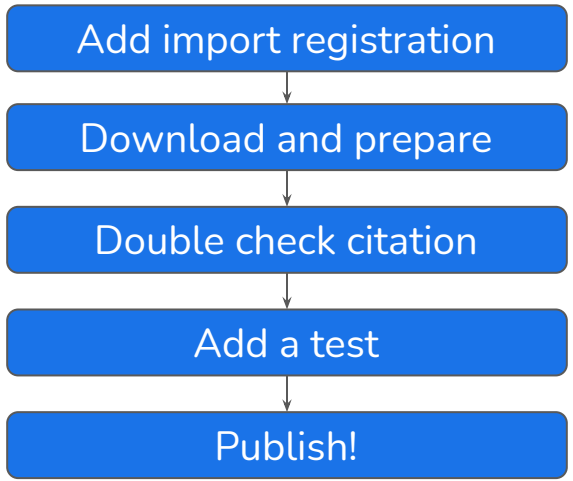
Loading a dataset with a custom configuration

```
# See the built-in configs
configs = tfds.text.IMDBReviews.builder_configs

>>> print(configs.keys())
dict_keys(['plain_text', 'bytes', 'subwords8k', 'subwords32k'])

# Address a built-in config with tfds.builder
imdb = tfds.builder("imdb_reviews/bytes")
# or when constructing the builder directly
imdb = tfds.text.IMDBReviews(config="bytes")
```

Publishing your own dataset



Add an import for registration

```
# In the 'image' subdirectory of tensorflow/datasets
from tensorflow_datasets.image.cifar import Cifar10
from tensorflow_datasets.image.cifar import Cifar100
...
from tensorflow_datasets.image.my_image_dataset import MyImageDataset

# In the 'text' subdirectory of tensorflow/datasets
from tensorflow_datasets.text.cnn_dailymail import CnnDailymail
...
from tensorflow_datasets.image.my_text_dataset import MyTextDataset
```

Download and prepare

- Create file

  `tensorflow_datasets/url_checksums/my_new_dataset.txt`

- Run `download_and_prepare` locally to ensure that data generation works

```
# default data_dir is ~/tensorflow_datasets
python -m tensorflow_datasets.scripts.download_and_prepare \
  --register_checksums \
  --datasets=my_new_dataset
```

Double-check citations

```
@ARTICLE {,
    author  = "John",
    title   = "Classification of ...",
    journal = "International Journal of ...",
    year    = "2019"
}
```

https://truben.no/latex/bibtex/

Test data

- Put test data under your dataset’s directory
- Make sure there are no duplicates in splits
- No copyrighted material

```python
from tensorflow_datasets import my_dataset
import tensorflow_datasets.testing as tfds_test

class MyDatasetTest(tfds_test.DatasetBuilderTestCase):
  DATASET_CLASS = my_dataset.MyDataset
  SPLITS = {  # Expected number of examples on each split from fake example.
      "train": 3,
      "test": 3,
  }
  # If dataset `download_and_extract`s more than one resource:
  DL_EXTRACT_RESULT = {
      "name1": "path/to/file1",  # Relative to fake_examples/my_dataset dir.
      "name2": "file2",
  }
  if __name__ == "__main__":
    tfds_test.test_main()
```

Final touches

- Make sure coding style is compliant with
  - PEP8
  - Google’s Python Style Guide
- Add release notes
- Send for review