BatchFlow: ML framework

Alexey Kozhevin
Gazprom Neft

Batchflow is:

DatasetIndex
FilesIndex
Batch
Dataset
Inbatch parallel
Pipeline
Model
Research

DatasetIndex

the way to index all data items to create batches

0 1 2 3 4 ...

```
dataset_index = DatasetIndex(np.arange(mnist.shape[0]))
```

FilesIndex

the way to index all necessary files to load in the future






/path/to/1.png /path/to/2.png /path/to/3.png /path/to/4.png /path/to/5.png ...

```
files_index = FilesIndex(path="/path/to/*.png")
```

FilesIndex

...or folders

/path/to/1/image.png /path/to/1/mask.png	/path/to/2/image.png /path/to/2/mask.png	/path/to/3/image.png /path/to/3/mask.png	/path/to/4/image.png /path/to/4/mask.png	/path/to/5/image.png /path/to/5/mask.png	...

```
files_index = FilesIndex(path="/path/to/*", dirs=True)
```

Batch

use prepared processing methods or define your own

```
class SatelliteBatch(Batch):
    components = 'images', 'masks'

    @action
    def some_action(self):
        # process your data
        retrun self
```

Dataset

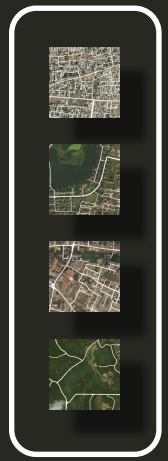
combine indexing and processing logic

/path/to/1/image.png /path/to/1/mask.png	/path/to/2/image.png /path/to/2/mask.png	/path/to/3/image.png /path/to/3/mask.png	/path/to/4/image.png /path/to/4/mask.png	/path/to/5/image.png /path/to/5/mask.png	...

```
index = FilesIndex(path='/path/to/*', dirs=True)
ds = Dataset(index, SatelliteBatch)
```

Inbatch parallel

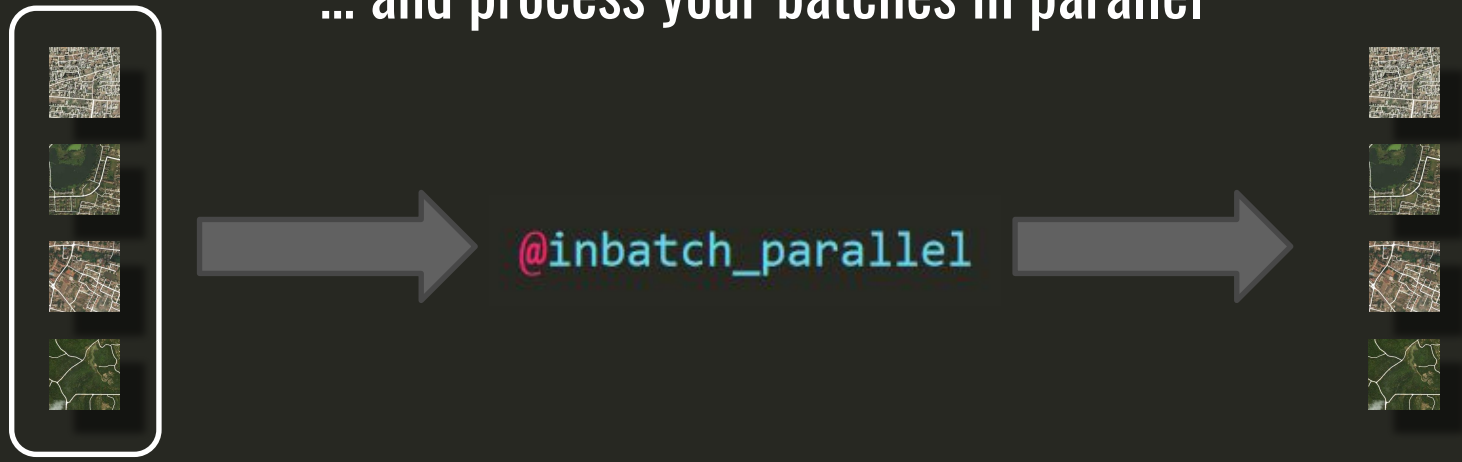
avoid loops for independent processing



```
for item in batch:
  # process batch
```

Inbatch parallel

... and process your batches in parallel




```
class MyBatch(Batch):
    ...
    @action
    @inbatch_parallel(init='_init_fn', post='_post_fn', target='threads')
    def some_action(self, item, arg1, arg2):
        # process just one item
        return some_value
```

Pipeline

chain actions into pipeline

```
augmentation_ppl = (Pipeline()
    .rotate(angle=P(R('uniform',-10, 10)), p=0.8)
    .crop(shape=(28, 28), origin='center')
    .scale(factor=P(R(factor_sampler)), preserve_shape=True, p=0.8)
)
```

Model

use ready-to-use neural network architectures

```
model_ppl = (Pipeline()
    .init_variable('loss_history', init_on_each_run=list)
    .init_model('dynamic', ResNet18, 'conv_nn', config={...}
    .train_model('conv_nn', fetches='loss', feed_dict={...}
                 save_to=V('loss_history'), mode='a')
)
```

Example

the whole training process will be very simple

```
from dataset import Pipeline, B, C, P, R
from dataset.opensets import MNIST
from dataset.models.tf import ResNet18

BATCH_SIZE = 64

dataset = MNIST()

augmentation_ppl = (Pipeline()
    .rotate(angle=P(R('uniform',-10, 10)), p=0.8)
    .crop(shape=(28, 28), origin='center')
    .scale(factor=P(R(factor_sampler)), preserve_shape=True, p=0.8)
)

model_ppl = (Pipeline()
    .init_variable('loss_history', init_on_each_run=list)
    .init_model('dynamic', ResNet18, 'conv_nn', config={...}
    .train_model('conv_nn', fetches='loss', feed_dict={...}
                 save_to=V('loss_history'), mode='a')
)

train_pipeline = (augmentation_ppl + model_ppl) << dataset.train
train_pipeline.run(BATCH_SIZE, shuffle=True, n_epochs=1, drop_last=True)
```

Research

train multiple models in parallel using few GPUs

```
domain = MyDomain(Option('model', [VGG7, VGG16, VGG19, ResNet18, ResNet34])
domain.set_iterator(brute_force=True, n_reps=1)

research = (Research()
    .init_domain(domain)
    .add_pipeline(train_ppl, name='train')
    .add_pipeline(test_ppl, name='test', import_from='train', execute='last')
    .get_metrics(pipeline='test', metrics_var='metrics', metrics_name='accuracy',
                 returns='accuracy', execute='last')
)

research.run(n_iters=None, bar=True, workers=2,
             devices=['gpu:'+str(item) for range(2)])
```