Cheat Sheet | GeoAI

ArcGIS Pro | Python | ArcPY

By: João Ataíde

Version 3.0

Data Preparation

Export Training Data

```
from argis.learn import *
```

```
arcgis.learn.export_training_data(input_raster, input_class_data=None,
                                          tile_size=256, rotation_angle=0
                                          stride_size=128, metadata_format= 'RCNN_Maks',
                                        classvalue_field=None, buffer_radius=None,
                                        output_location=None, context=None,
                                          input_mask_polygons=None,chip_format='TIFF',
                                        reference_system='MAP_SPACE')
```

Metadata Format

`KITTI_rectangles`: used with FasterRCNN, RetinaNet, SingleShotDetector and YOLOv3 models.

`PASCAL_VOC_rectangles`: sed with FasterRCNN, RetinaNet, SingleShotDetector and YOLOv3 models.

`Classified_Tiles`: used with BDCNEdgeDetector, DeepLab, HEDEdgeDetector, MultiTaskRoadExtractor, PSPNetClassifier and UnetClassifier models.

`RCNN_Masks`: used with MaskRCNN model.

`Labeled_Tiles`: used with FeatureClassifier model.

`Multi-labeled Tiles`: used with FeatureClassifier model.

`Export Tiles`: used with ChangeDetector, CycleGAN, Pix2Pix and SuperResolution models.

Prepare data

```
data = arcgis.learn.prepare_data(path, class_mapping=None,
                                 chip_size=256, val_split_pct=0.1, batch_size=64,
                                 transforms=None, seed=42, dataset_type=None,
                                 resize_to=None, imagery_type = 'sentinel2',  bands =
                                 ['r', 'g', 'b', 'nir'])
```

`dataset_Type`: argument is the same as exported in samples

```
data.classes
```

```
data.show_batch()
```

Model Training

`backbone`: resnet18, resnet34, resnet50, resnet101, resnet152, densenet121, densenet161, densenet169, densenet201, mobilenet, vgg11, vgg13, vgg16, vgg19, darknet53, reidv1, reidv2.

Object Detection

- **FasterRCNN**

  https://arxiv.org/abs/1506.01497

```
model = FasterRCNN(data, backbone='resnet50',
                        pretrained_path=None)
```

- **RetinaNet**

```
model = RetinaNet(data, scales=None,
                      ratios=None, backbone='resnet50',
                      pretrained_path=None)
```

- **SingleShotDetector**

```
model = SingleShotDetector(data, grids=None, zooms=[1.0],
                                 ratios=[[1.0, 1.0]],
                                 backbone='resnet50',
                                 drop=0.3, bias=- 4.0,
                                 focal_loss=False,
                                 pretrained_path=None)
```

- **MaskRCNN**

  https://arxiv.org/abs/1703.06870.

```
model = MaskRCNN(data, backbone='resnet50',
                    pretrained_path=None,
                    pointrend=False)
```

- **YOLOv3**

```
model = YOLOv3(data, pretrained_path=None)
```

- **MMDegtection**

```
model = MMDetection(data, model, model_weight=False,
                        pretrained_path=None)
```

Classify Pixel

- **UnetClassifier**

```
model = UnetClassifier(data, backbone='resnet50',
                            pretrained_path=None)
```

- **PSPNetClassifier**

  https://arxiv.org/abs/1612.01105

```
model = PSPNetClassifier(data, backbone='resnet50', use_unet=True,
                                pyramid_sizes=[1, 2, 3, 6]
                                pretrained_path=None)
```

- **DeepLab**

  https://arxiv.org/abs/1706.05587

```
model = DeepLab(data, backbone='resnet50',
                    pretrained_path=None)
```

- **BDCNEdgeDetector**

  https://arxiv.org/pdf/1902.10903.pdf

```
    model = BDCNEdgeDetector(data, backbone='vgg19')
```

- **HEDEdgeDetector**

  https://arxiv.org/pdf/1504.06375.pdf

```
model = HEDEdgeDetector(data, backbone='vgg19')
```

- **MultiTaskRoadExtractor**

  https://doi.org/10.1109/CVPR.2019.01063

```
model = MultiTaskRoadExtractor(data, backbone='resnet50')
```

- **ConnectNet**

  https://doi.org/10.1109/CVPR.2019.01063

```
model = ConnectNet(data, backbone='resnet50',
                         pretrained_path=None)
```

- **ChangeDetection**

  https://www.mdpi.com/2072-4292/12/10/1662

```
model = ChangeDetector(data, backbone='resnet50',
                             attention_type='PAM',
                             pretrained_path=None)
```

- **MMSegmentation**

```
model = MMSegmentation(data, model, model_weight=False,
                             pretrained_path=None
```

Image Translation Models

- **CycleGAN**

  https://arxiv.org/abs/1703.10593

```
model = CycleGAN(data, pretrained_path=None, gen_blocks=9, lsgan=True)
```

- **Pix2Pix**

```
model = Pix2Pix(data, pretrained_path=None)
```

- **Pix2PixHD**

```
model = Pix2PixHD(data, pretrained_path=None)
```

- **ImagemCaptione**

```
model = ImageCaptioner(data, backbone='resnet50', pretrained_path=None)
```

- **SuperResolution**

```
model = SuperResolution(data, backbone='resnet50', pretrained_path=None)
```

Object Tracking Models

- **SiamMask**

```
model = SiamMask(data)
```

- **DeepSort**

```
model = DeepSort(data)
```

- **ObjectTracker**

```
model = SiamMask(data)
```

- **Track**

```
model = Track(id, label, bbox, mask)
```

Feature, Tabular and Timeseries models

- **FullyConnectedNetwork**

```
model = FullyConnectedNetwork(data, layers=None, emb_szs=None)
```

- **MLModel**

```
model = MLModel(data, model_type)
```

- **TimeSeriesModel**

```
model = TimeSeriesModel(data, seq_len,
                                model_arch='InceptionTime')
```

- **AutoML**

```
model = AutoML(data=None, total_time_limit=3600,
                    mode='Explain', algorithms=None,
                    eval_metric='auto')
```

3D Models

- **PointCNN**

```
model = PointCNN(data, pretrained_path=None)
```

- **Transform3d**

```
model = Transform3d(rotation_range=[0.04363323129985824, 3.141592653589793, 0.0436332312998
                    scaling_range=[0.05, 0.05, 0.05, 'g'],
                    jitter=0.0)
```

Training Model

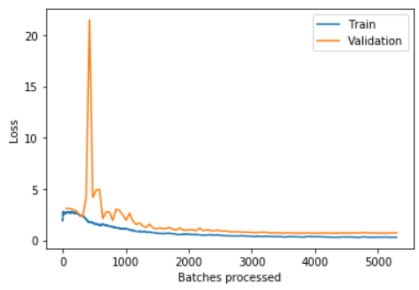
```
lr = model.lr_find()

model.fit(epochs=10, lr=lr, one_cycle=True,
                early_stopping=False, checkpoint=True)
```

Metrics

Object Detection

```
model.plot_losses()
```




```
model.average_precision_score()
```

```
model.show_results()
```

```
model.save(name_or_path, framework='PyTorch',
                publish=False, gis=None,
                compute_metrics=True,
                save_optimizer=False,
                save_inference_file=True)
```

Deploy

```
model = FasterRCNN.from_model(model_path)
```

```
model.predict(image_path, threshold=0.1,
                       nms_overlap=0.1,
                     return_scores=True,
                    visualize=False, resize=False)
```

```
model.predict_video(input_video_path=video_file,
                         metadata_file=None,
                        track=True,
                        threshold=0.95,
                        visualize=True,
                        resize=True)
```

Links

Pretrained AI Models

Blog ESRI GeoAI

Deep Learning Framework

Deep Learning Tooset

ArcGIS API for Python

ArcGIS Pro Documentation