cloud_detection

December 2, 2021

If you plan on using this implementation, please cite our work:

@INPROCEEDINGS{Grabowski2021IGARSS, author={Grabowski, Bartosz and Ziaja, Maciej and Kawulok, Michal and Nalepa, Jakub}, booktitle={IGARSS 2021 - 2021 IEEE International Geoscience and Remote Sensing Symposium}, title={Towards Robust Cloud Detection in Satellite Images Using U-Nets}, year={2021}, note={in press}}

1 Demo of the cloud detection using U-Net architecture

This document presents the cloud detection on example Landsat 8 multispectral images using trained U-Net model. The full script can be found in cloud_detection/exp_main.py.

First, we import neccessary libraries.

[ ]:
```
import numpy as np
from pathlib import Path
from tensorflow import keras

from cloud_detection.models import unet
from cloud_detection.evaluate_L8CCA import evaluate_model
from cloud_detection.losses import (
    JaccardIndexLoss,
    JaccardIndexMetric,
    DiceCoefMetric,
    recall,
    precision,
    specificity,
)
```

Next, we set the parameters for the experiment. These parameters are the following:

- dpath - path to the dataset.
- rpath - path to directory where results should be stored.
- mpath - path to trained model weights.
- vids - tuple of IDs of images which should be used to create visualizations. If contains '*' visualizations will be created for all images in the datasets.
- eval_imgs - IDs of images to evaluate.
- batch_size - size of generated batches, only one batch is loaded to memory at a time.
- thr - threshold for determining whether pixels contain the clouds.
- learning_rate - learning rate for training (needed to load the trained model).

- bn_momentum - momentum of the batch normalization layer.

[2]:
```
dpath = Path("datasets/clouds/
 ↪Landsat-Cloud-Cover-Assessment-Validation-Data-Partial")
rpath = Path("artifacts/cloud_detection_demo/")
mpath = "examples/cloud_model_weights/best_weights"
vids = ["*"]
eval_imgs = ["LC81390292014135LGN00", "LC80460282014171LGN00"]
batch_size = 8
thr = 0.5
learning_rate = 0.01
bn_momentum = 0.9
```

We create the instance of the untrained U-Net model. Next, we load the trained weights into the model.

[ ]:
```
model = unet(input_size=4, bn_momentum=bn_momentum)
model.compile(
    optimizer=keras.optimizers.Adam(lr=learning_rate),
    loss=JaccardIndexLoss(),
    metrics=[
        keras.metrics.binary_crossentropy,
        keras.metrics.binary_accuracy,
        JaccardIndexLoss(),
        JaccardIndexMetric(),
        DiceCoefMetric(),
        recall,
        precision,
        specificity,
    ],
)
model.load_weights(mpath)
```

We create the directory to store the results of the model evaluation. Next, we evaluate the model using example Landsat 8 images. The following files are created for each image:

- gt.png - image of the ground-truth cloud mask.
- pred.png - image of the model prediction.
- masks.png - visualisation of the model prediction. Yellow color denotes True Positives, red color denotes False Positives and purple color stands for False Negatives.
- unc.png - uncertainty map, where pixels with uncertain prediction scores are marked in yellow (Note: In the case of the tested model, in most cases almost all of the pixels' prediction scores are very low or very high, which means that the map will almost always not include any yellow pixels.).

If the model's prediction Jaccard Index Metric does not exceed 0.6, the following files are also created:

- roc.html - ROC curve.
- prec_recall.html - precision-recall curve.

- activation_hist.html - histogram of the model's activations scores (please note the logarithmic scale).

[4]:
```
rpath.mkdir(parents=True, exist_ok=True)
metrics_L8CCA, _ = evaluate_model(
    model=model,
    thr=thr,
    dpath=dpath,
    rpath=rpath / "eval_vis",
    vids=vids,
    batch_size=batch_size,
    img_ids=eval_imgs
)
```

```
Processing Barren-LC81390292014135LGN00
Scene prediction took 80.25445532798767 seconds
Average inference time: 80.25445532798767 seconds
Creating visualisation for LC81390292014135LGN00

/home/bgrabowski/Documents/machine-learning/cloud_detection/utils.py:261:
UserWarning:

artifacts/cloud_detection_demo/eval_vis/LC81390292014135LGN00/gt.png is a low
contrast image

Lossy conversion from int64 to uint8. Range [0, 1]. Convert image to uint8 prior
to saving to suppress this warning.

Will make insights for LC81390292014135LGN00
thr dist variance: 3.406673570986612e-06
thr dist mean: 0.35297889839746927
Optimal thr: 1.0
Processing Urban-LC80460282014171LGN00
Scene prediction took 70.27914357185364 seconds
Average inference time: 75.26679944992065 seconds
Creating visualisation for LC80460282014171LGN00

/home/bgrabowski/Documents/machine-learning/cloud_detection/utils.py:261:
UserWarning:

artifacts/cloud_detection_demo/eval_vis/LC80460282014171LGN00/gt.png is a low
contrast image

Lossy conversion from int64 to uint8. Range [0, 1]. Convert image to uint8 prior
to saving to suppress this warning.
```

Finally, we process the output metrics to obtain the mean metrics for the model evaluation.

[5]:
```
mean_metrics_L8CCA = {}
for key, value in metrics_L8CCA.items():
    mean_metrics_L8CCA[key] = np.mean(list(value.values()))
print(mean_metrics_L8CCA)
```

```
{'L8CCA_binary_crossentropy': 2.502019, 'L8CCA_binary_accuracy': 0.8431798,
'L8CCA_jaccard_index_loss': 0.33403495, 'L8CCA_jaccard_index_metric': 0.665965,
'L8CCA_dice_coeff_metric': 0.78108644, 'L8CCA_recall': 0.95701694,
'L8CCA_precision': 0.6854487, 'L8CCA_specificity': 0.8053814,
'L8CCA_normalized_mutual_info_score': 0.4949737413470268,
'L8CCA_adjusted_rand_score': 0.5242536788935748}
```