Acceleration material

Deep learning

Brice Convers, Jeanne Lemoine

SICOM 3A

19/12/2023

Model architecture

UNet architecture

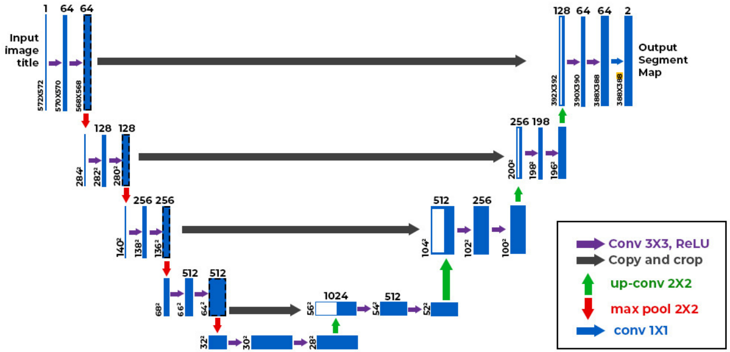

Figure 1: Architecture of a UNet model [2]

Link to the initial model : https://UNet

UNet architecture

Pros :

- Specifically designed for semantic segmentation tasks
- Captures context and preserves spatial information
- Performs well even with a limited amount of training data

Cons :

- Sensitive to the input size of images
- Can be computationally intensive

Data split and resize

Creation of the train and test sets :

- *train_test_split()* function from the *sklearn* library
- Definition of a percentage which separates our dataset into train and test

Resize the images :

- Initial dataset : 400 images (6000x4000 pxls) [5]
- Resize : less computational resources, reduce the memory storage, faster training, reduce overfitting

Tree of the project

```
├── assets
│     └── results
├── dataset
│     └── semantic_drone_dataset
│           ├── label_semantic_images
│           ├── RGB_color_image_masks
│           └── original_images
├── documents
├── output
├── src
│     ├── dataset.py
│     ├── config.py
│     ├── main.py
│     ├── metrics.py
│     ├── model.py
│     ├── test.py
│     ├── train.py
│     └── utils.py
├── gitignore.
├── GricadScript.sh
├── README.md
└── requirement.txt
```

- Object-oriented programming with classes
- 1 file gathering all the parameters + 1 main for both test and train

Model architecture

- **Class Block** : store the convolutional and relu layers
- **Class Encoder** : 64 Blocks and maxpool
- **Class Decoder** : up-samplings, cropping and 64 Blocks
- **Class UNet** : assembly of Encoder and Decoder with an interpolation of type *torch.nn.functional.interpolate()*

Adjustments for a binary classification

When $NBR\_CLASSES = 1$, we face a binary classification.

- Unlabelled binary segmentation
  - **Mean** threshold on the masks in grayscale to have 50% of both classes : extract the contours

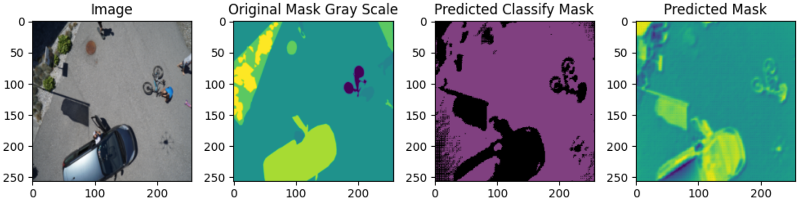

Figure 2: Results for binary classification

Choice of hyperparameters

- Test split = 0.15 (%) = 340 + 60 images : not many images so more important training set
- Batch size = 4, 8 or 16 : multiple of 4 for 4 GPU

Smaller batch : slower, less stable, faster convergence, better for GPUs memory, less overfitting
Bigger batch : better generalization

- Model size = 64 : not too heavy
- Optimizer = Adam (Adaptive Moment Estimation) : robust and can adapt the learning rate
- Loss function = BCEWithlogitsloss() for binary classificiation or CrossEntropyLoss() for 24 classes

Choice of hyperparameters

- Number of classes = 24
- Image Sizes = (128,128), (256,256), (512,512) : decrease computational time, small dataset

For a model with 64 layers we only want the contours, wa can decrease the image size.

- Number of epoch = 10-15 for small test, maximum 50
- Learning rate = 0.01 : trade-off between over and under fitting and convergence and divergence

Too big : can oscillate or diverge and miss the solution but faster.

- No dropout or L2 regularization

Parallelization [3]

- Activation of the parallelism : $ACTIVATE\_PARALLELISM = True$
- Generic function for training and testing
- In general, 3 or 4 GPUs with 1 node :
  - ✓ **Improvement of 30% of the time of computation with 3 GPUs**
- No parallelism needed for the test

Multiclass Segmentation results

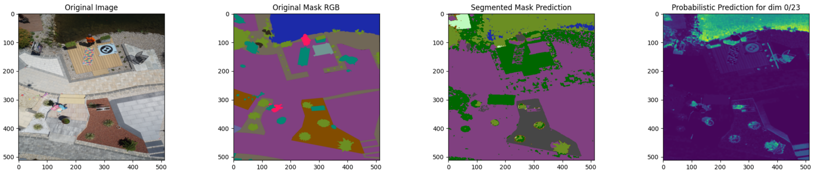

Figure 3: Example of output (CrossEntropyLoss(), learning rate = 0.01, epoch = 50, image size = (512,512), size of the model = 64, batch size = 4, 340 train images)

Loss curves

Figure 4: Training and testing loss curves

Learning Rate and DICE coefficient

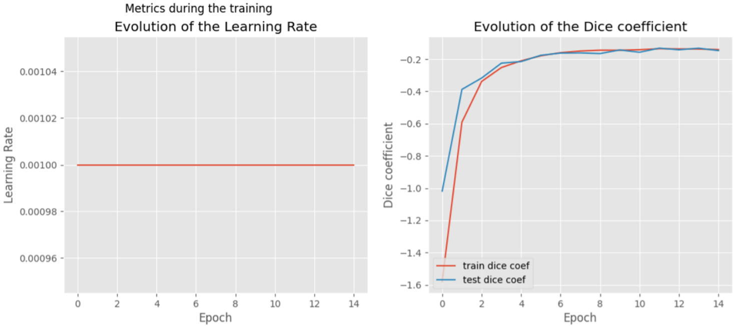

Figure 5: Training Metrics

- Convergence of our model

DICE : measure of similarity

Metrics: Confusion Matrix

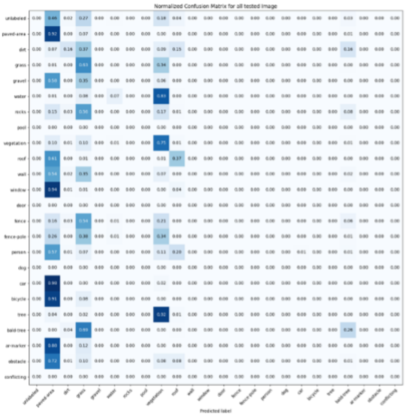

Figure 6: Confusion Matrix (epoch = 50, 340 training images, learning rate = 0.01)

Metrics: Confusion Matrix

- F1 = 0.0953

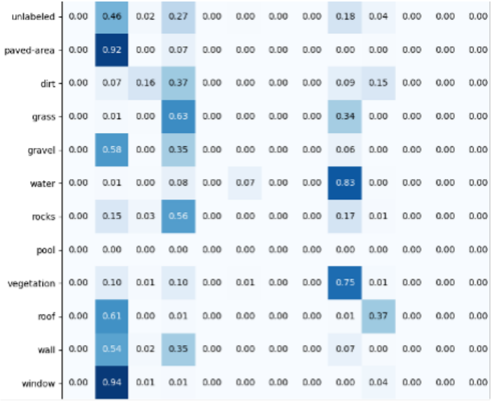

Figure 7: Confusion Matrix (epoch = 50, training images, learning rate = 0.01)

Theory

Why :

- Segmentation was not detailed enough
- Semantic was really bad

How :

- Torch transformations: Horizontal Flip, Vertical Flip, Random Crop
- 400 images and masks $\rightarrow$ 1600 images and masks [4] [1]

New dataset : 1360 train + 240 test images
Batch size = 4

Metrics: Confusion Matrix

- F1: 0.1023 (up to 7.2%)

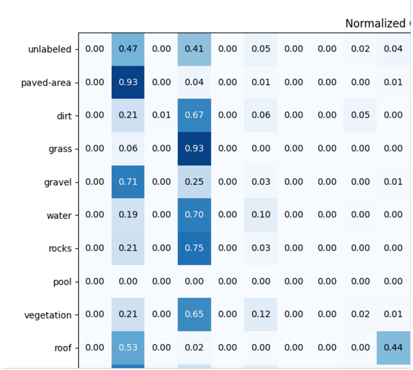

Figure 8: Confusion Matrix (epoch = 8, 1360 train images, learning rate = 0.01)

Metrics: Confusion Matrix with AugmentedDdata

- F1: 0.1021

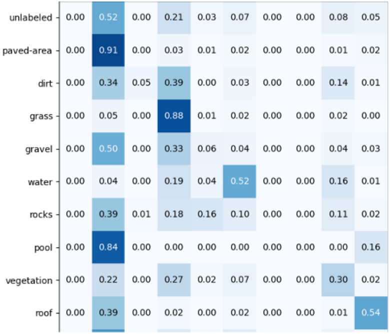

Figure 9: Confusion Matrix (epoch = 50, 1360 train images, learning rate = 0.01)

Early stopping

- Stop the code when the loss function tends towards an asymptotic for several epochs in a row

$EARLY\_STOPPING\_ACTIVATE = \text{True}$
$PATIENCE = 5$

Conclusion

+++

- Object-oriented programming model
- Management of the parameters
- Parallelization
- Data augmentation

To be continued :

- Metrics unfinished
- New tests with heavier hyperparameters needed

[Ayu] Ayu. URL: https://github.com/ayushdabra/drone-images-semantic-segmentation.

[Bel] Nada Belaidi. *U-Net : le réseau de neurones populaire en Computer Vision*. URL: https://blent.ai/blog/a/unet-computer-vision.

[Cha] Dawood Al Chanti. URL: https://scholar.google.fr/citations?user=osXo54QAAAAJ&hl=en.

[Pao] Niccolo Paolinelli. *Semantic Segmentation Drone Dataset (U-Net)*. URL: https://www.kaggle.com/code/nicopaolinelli/semantic-segmentation-drone-dataset-u-net.

[Siy] Bulent Siyah. *Aerial Semantic Segmentation Drone Dataset*. URL: https://www.kaggle.com/datasets/bulentsiyah/semantic-drone-dataset.