This project has received funding from
The European Union's Horizon 2020 research and innovation programme under
Grant agreement No 101017562

Finding Endometriosis using Machine Learning
FEMaLe

Call/Topic: Digital transformation in Health and Care

Type of action**:** RIA

Date: 03.03.2023

DELIVERABLE NUMBER	D6.3
DELIVERABLE TITLE	Deep learning algorithm to automatically determinate the stage of endometriosis
RESPONSIBLE AUTHOR	RTU

GRANT AGREEMENT No.	101017562
DOCUMENT TYPE	Other
WORKPACKAGE N. | TITLE	6 - DIAGNOSIS: surgical phenotyping using machine learning
LEAD CONTRACTOR	RTU (P8)
AUTHOR(S)	AARHUS UNIVERSITY HOSPITAL (P2), SEMMELWEIS UNIVERSITY (P4), SURGAR (P6), RIGA TECHNICAL UNIVERSITY (P7), AARHUS UNIVERSITY (P1)
PLANNED DELIVERY DATE	February 28th, 2023
ACTUAL DELIVERY DATE	March 3rd, 2023
DISSEMINATION LEVEL	Public
STATUS	Reviewed and quality checked
VERSION	Final version (1.5)
REVIEWED BY	FEMaLE PMO

Version	**Date** [1]	**Comment**	**Author**	**Status** [2]
1.1	03-11-2022	First draft created	RTU	Drafted
1.2	16-12-2022	Second draft created, including recommendations from WP participants.	RTU	Drafted
1.3	17-01-2023	Third draft prepared for FEMaLe Review Panel.	RTU	Drafted
1.4	15-02-2023	Final draft created, based on FEMaLe Review Panel feedback.	RTU	Completed
1.5	28-02-2023	Final version ready for submission, quality checked by FEMaLe PMO.	AU	Validated

[1] As per the project’s cloud storage or per email date if applicable.

[2] Drafted, completed or validated as per the project's cloud storage or per email date if applicable.

Document history

TABLE OF CONTENTS

Disclaimer

The content of this deliverable does not reflect the official opinion of the European Union.
Responsibility for the information and views expressed herein lies entirely with the author(s).

All FEMaLe Consortium members are also committed to publish accurate and up to date information and take the greatest care to do so. However, the FEMaLe Consortium members cannot accept liability for any inaccuracies or omissions, nor do they accept liability for any direct, indirect, special, consequential, or other losses or damages of any kind arising out of the use of this information.

Copyright notice

Acknowledgement

The project 'Finding Endometriosis using Machine Learning' (FEMaLe) has received funding from the European Union Horizon 2020 programme under grant number 101000640.

Citation

Be so kind as to cite this work as:

Finding Endometriosis using Machine Learning, 2023: Deep learning algorithm to automatically determinate the stage of endometriosis, under the supervision of the Project's Coordinator.

Legislation

Legislation H2020 Framework Programme – Regulation (EU) No 1291/2013 of the European Parliament and of the Council of 11 December 2013 establishing Horizon 2020 - The Framework Programme for Research and Innovation (2014-2020) (OJ 347, 20.12.2013, p. 104).

Euratom Research and Training Programme (2014-2018) – Council Regulation (Euratom) No 1314/2013 of 16 December 2013 on the Research and Training Programme of the European Atomic Energy Community (2014-2018) complementing the Horizon 2020 – The Framework Programme for Research and Innovation (OJ L 347, 20.12.2013, p. 948).

H2020 Specific Programme – Council Decision 2013/743/EU of 3 December 2013 establishing the Specific Programme Implementing Horizon 2020 - The Framework Programme for Research and Innovation (2014-2020) (OJ L 347, 20.12.2013, p. 965).

Rules for Participation (RfP) – Regulation (EU) No 1290/2013 of the European Parliament and of the Council of 11 of December 2013 laying down the rules for the participation and dissemination in Horizon 2020 – the Framework Programme for Research and Innovation (2014-2020) (OJ L 347, 20.12.2013, p.81).

Financial Regulation (FR) – Regulation (EC, Euratom) No 966/2012 of the European Parliament and of the Council of 25 October 2012 on the financial rules applicable to the general budget of the European Union (OJ L 298, 26.10.2012, p.1).

Rules of Application (RAP) – Commission Regulation (EC, Euratom) No 1268/2012 of 29 October 2012 on the rules of application of l Regulation (EC, Euratom) No 966/2012 of the European Parliament and of the Council on the financial rules applicable to the general budget of the Union (OJ L 298, 26.10.2012, p.1).

Finding Endometriosis using Machine Learning: FEMaLe

Introduction

Deep learning techniques follow two stages. The first stage is the training stage. The computer learns the task from a large dataset of examples, including the information to be automatically predicted, namely image annotations defined manually by an expert surgeon. The second stage is the prediction stage. The computer predicts the desired information on new, previously unseen images, thanks to its 'experience' accumulated during training. This approach will be applied to laparoscopic images to perform semantic segmentation of anatomical structures and surgical tools. Neural network architectures dedicated to the proposed segmentation task will be designed. These architectures must integrate the constraint of a minimal amount of training data compared to daily life images most commonly used to train convolutional neural networks.

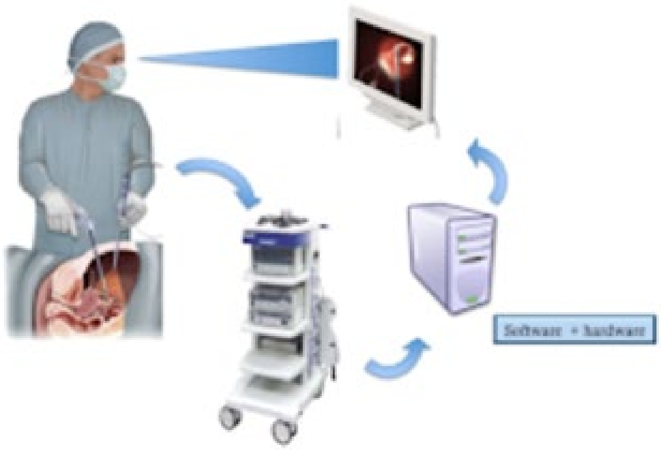
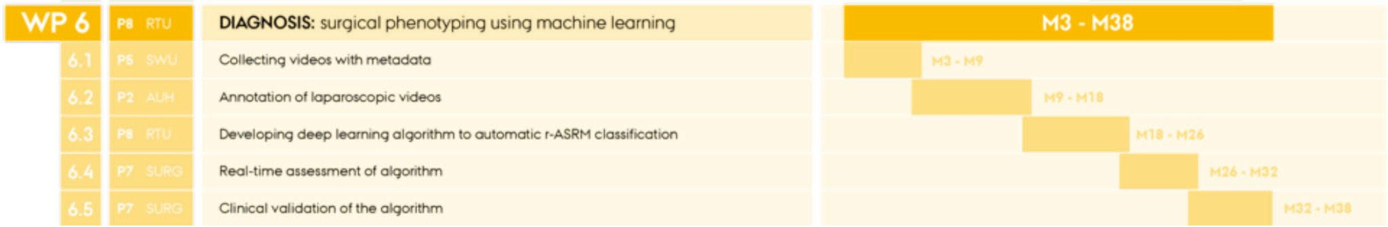


Figure 1. Endometriosis lesions automatic segmentation during surgery

The tasks in WP6 are arranged by the Waterfall principle since each next task and deliverable is strictly dependent on the previous results with a slight overlap.



WP 6	P8 RTU	**DIAGNOSIS:** surgical phenotyping using machine learning	M3 - M38
6.1	P5 SWU	Collecting videos with metadata	M3 - M9
6.2	P2 AUH	Annotation of laparoscopic videos	M9 - M18
6.3	P8 RTU	Developing deep learning algorithm to automatic r-ASRM classification	M18 - M26
6.4	P7 SURG	Real-time assessment of algorithm	M26 - M32
6.5	P7 SURG	Clinical validation of the algorithm	M32 - M38

Figure 2. WP6 tasks and deliverables

After obtaining the first several thousand annotated lesion images, it became possible to start testing promising neural architectures.

1. Datasets description

1.1 Bounding box

Regarding laparoscopic surgery, which involves operating inside the abdomen using a camera and specialized tools, there are some challenges associated with using the Bounding Box approach for segmentation.

One of the key challenges is that laparoscopic images can be highly complex and cluttered, making it difficult to identify and segment individual objects using just bounding boxes accurately. In some cases, multiple objects may be partially or fully occluded by other objects or tissue, making it difficult to segment them using a bounding box alone accurately.

Another challenge is that laparoscopic images may contain a wide range of object sizes and shapes, making it difficult to define a fixed bounding box size that works well across all objects. This can lead to false positives or false negatives in the object detection process, which can impact the accuracy of the segmentation.

To address these challenges, there are a number of alternative segmentation methods that go beyond traditional bounding box-based object detection. For example, semantic segmentation involves assigning a class label to each pixel in an image to accurately segment objects within laparoscopic images. Or instance segmentation, which involves identifying each instance of an object within an image and segmenting it at the pixel level to accurately segment complex and overlapping objects within laparoscopic images.

1.2 Data preparation and splitting

Data exported from Supervisely using the space-saving format, which is rather problematic to be read in a fast and efficient manner for Object-Detection or Segmentation deep neural networks. This was achieved via the creation of the "supervisely2coco" Kedro pipeline capable of conversion from Supervisely to COCO data format.

The next problem was the data set split into train/test/validation datasets in a stratified manner. Early experiments have shown that leakage of the frames belonging to the same video into a train/test can considerably improve test performance - this is an example of train data leakage. To mitigate this problem, another "coco_datasplit" Kedro pipeline[3] was created for splitting videos based on the classes/annotations present inside them into train/test/validation sets.

[3] https://kedro.readthedocs.io/en/stable/index.html

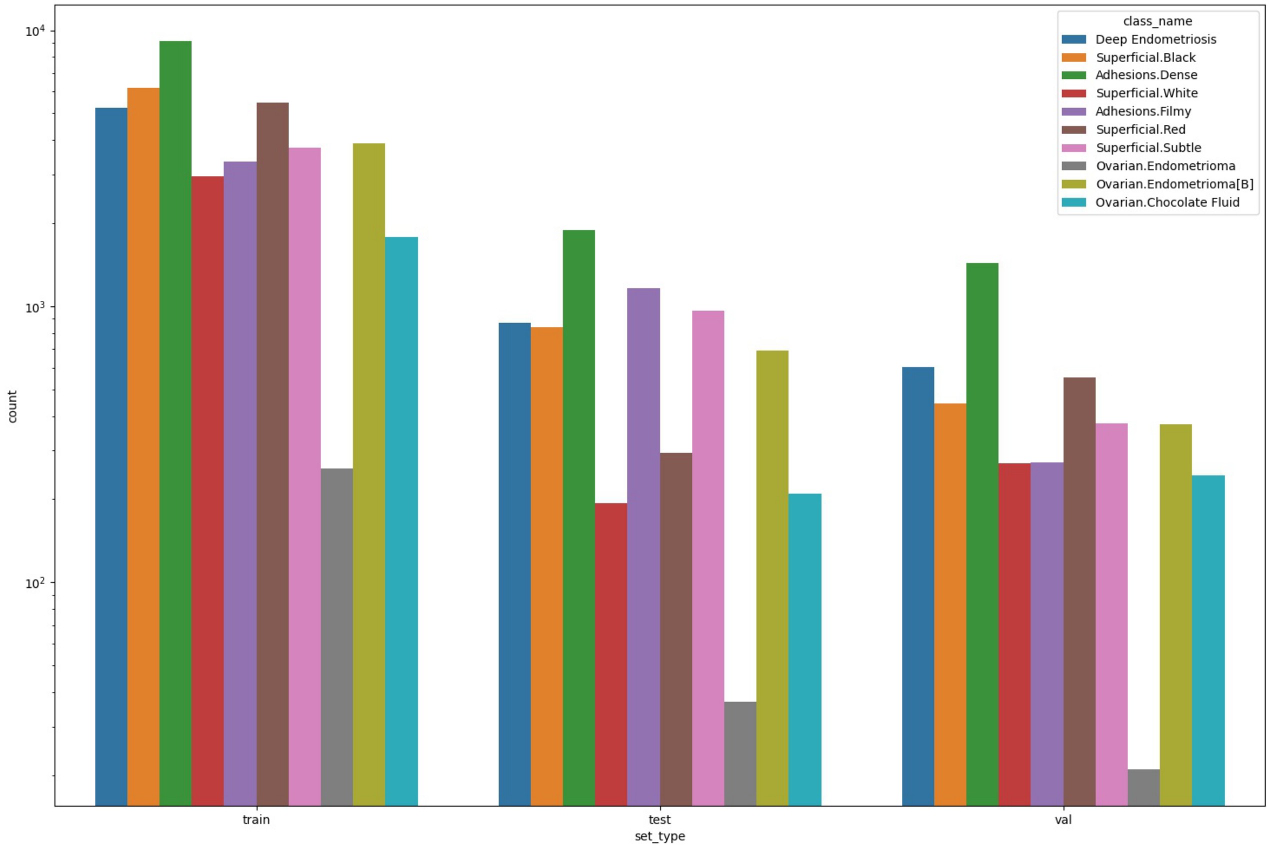


Figure 3: Train / test / validation data splits statistics depicting class distributions across three datasets.

To approach this task, we had to reformulate our problem as a multilabel dataset stratification as a single video contains multiple bounding boxes or segmentations belonging to different classes. This problem is intractable in case we want to find the best possible data split.

To be able to find an acceptable but not necessarily the best solution in a manageable time, we have utilized the "Iterable Stratification" library[4]. See the Fig 3 of the reproducible dataset split statistics (notice the vertical axis having a log scale).

To be able to utilise COCO JSON files along with the data split files (holding the names of the video files in appropriate train/test/validation sets) we needed to create a PyTorch dataset class capable of processing the above-mentioned files. The dataset has to be easily adjustable so that different model inputs should be easily generated from the COCO data. Thus, it was necessary to prepare a flexible objects hierarchy supporting this requirement, see Fig 4.

[4] https://github.com/trent-b/iterative-stratification

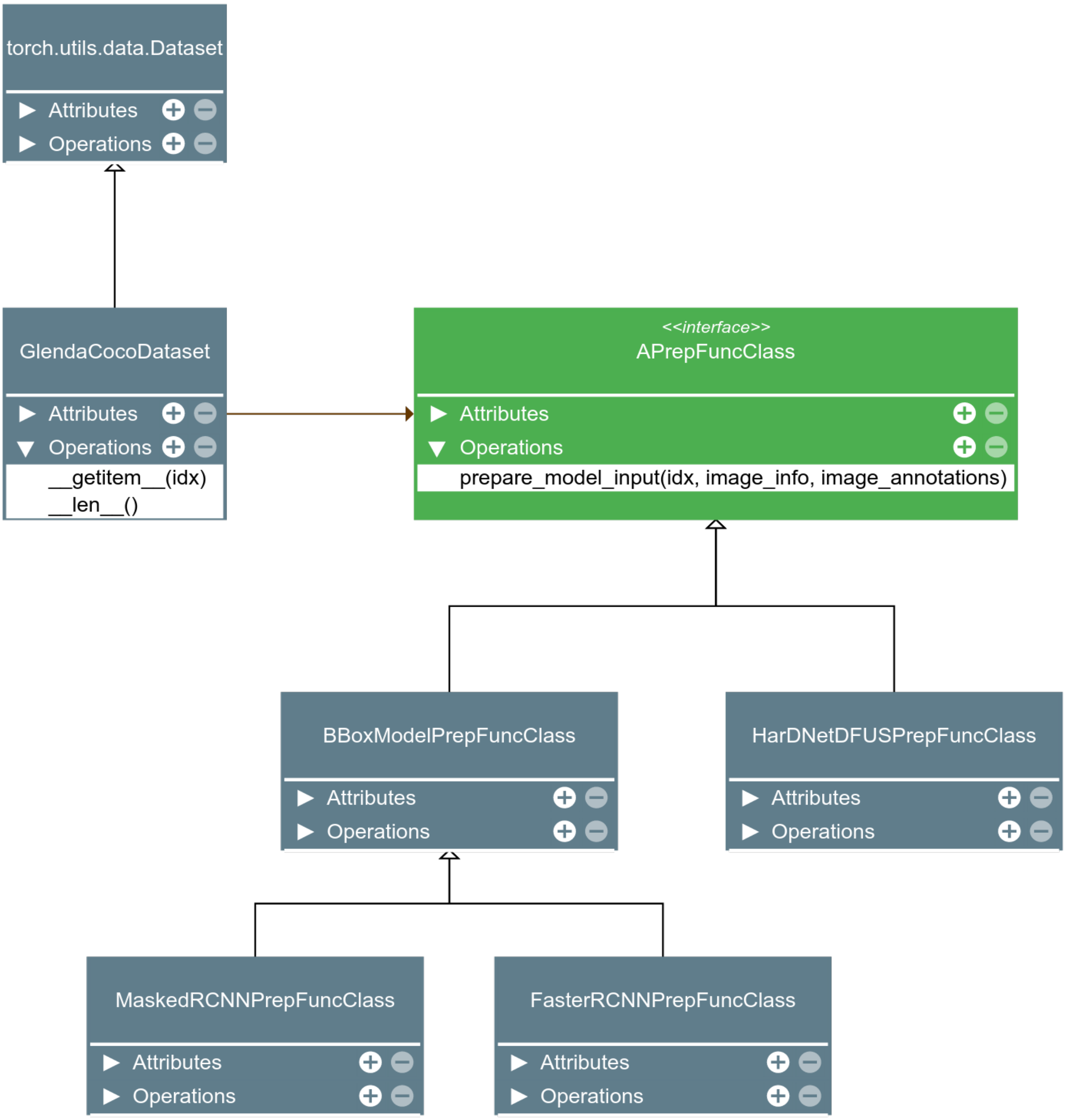


Figure 4. Class diagram for the data set wrapper allowing dataset reusability for models requiring varying inputs

1.3 Cross-board image sharing

Since surgery centers are located in different countries, some of the stringent restrictions (e.g., national laws and GDPR) limit the use of ML (machine learning) in the medical area[5]. In the beginning of the project, there was a delay due to the lengthy procedure of settling all GDPR aspects. Due to a limited count of patients from one project partner, it is nearly impossible to gather a minimally viable dataset in one place. We studied a GDPR-compliant solution that will not transfer any personal data while maintaining the ability to train NNs (neural networks) collaboratively in different countries.

[5] 1. Bovenberg, J., et. al. (2020). How to fix the GDPR's frustration of global biomedical research. Science, 370, 40-42.

We addressed two problems: data roaming expenses and the variability in computing power between systems used for training NNs. This study aims to introduce a solution for the above-mentioned problems by using a federated learning approach[6]. To achieve this goal, two methods were applied: weights quantization and NN model separation into submodels. This system was validated using simulation with several clients, each of which had unique dataset elements. Simulation results showed that the use of the proposed system could reduce the amount of transferred data by 87% without a major accuracy reduction. The proposed system allows collaborative training of NNs and is GDPR compliant, as it does not transfer any personal data, allowing the wide and rapid spread of ML systems.

Several code libraries were examined at the time of writing (Spring 2022) that were considered the most popular (Liu, 2022, p. 6). These are the following libraries:

- TFF (Tensorflow Federated) https://github.com/tensorflow/federated,
- PySyft (Ryffel, Trusk, et al., 2018) https://github.com/OpenMined/PySyft,
- Flower (Beutel, Topal, et al., 2020) https://github.com/adap/flower.

It should be noted that in each of the above-mentioned libraries, a specific version of its documentation was examined, i.e., the versions that were available on the libraries' official websites and GitHub repositories at the time of writing. In the process of exploring the documentation of the first library (TFF), it was concluded that the documentation essentially describes the application programming interface (API) and the federated machine learning methods implemented therein, which are mostly accompanied by code fragments. The lack of complete practical code examples indicated this fact. This definitely complicates the exploration and use of this library because, in such cases, it is usually necessary to study the library's source code until it is clear how to use it. This became one of the reasons why the decision was made not to use this library in the solution development process. It was considered noteworthy that several weight aggregation algorithms are provided in this library and that it supports differential privacy mechanisms.

Analyzing the documentation of the second library (PySyft), a similar situation was observed as in the case of the first library. Firstly, there were no practical examples of using this library in the documentation, but they could be found in other web resources, such as Chandorikar (2020). It should be noted that at the time of writing, unlike the TFF library, the section of the API in the official documentation website (https://openmined.github.io/PySyft) was marked as being in the development process.

Secondly, similarly to the TFF library, several weight aggregation algorithms and data privacy mechanisms are implemented in the PySyft library. For the same reason as in the case of the TFF library, the decision was made to refrain from using this library in the solution development process.

In the case of the third library (the Flower library), the aforementioned problems were not observed. With the help of the documentation, it was possible to find information about the interesting method or class in a timely manner, as well as to start developing the solution in the shortest possible time using the examples provided in the documentation.

[6] 1. Konečný, J., et. al. (2015). Federated Optimization: Distributed Optimization Beyond the Datacenter. arXiv:1511.03575

2. Neural architectures

Medical RGB image segmentation is an important task in medical image analysis that involves identifying and separating different regions of interest within an image. Neural networks have shown great promise in achieving accurate and efficient medical image segmentation. However, there is no one-size-fits-all neural network architecture for this task, and it is important to experiment with different architectures to determine which one works best for a particular application. Trying out different neural network architectures can help researchers and medical professionals achieve more accurate and efficient segmentation, which is crucial for diagnosis, treatment planning, and disease monitoring. Therefore, exploring various neural network architectures is essential for advancing the field of medical image segmentation and improving patient outcomes.

2.1. Plain U-Net

The first neural architecture in hindsight was "Plain U-Net" (proposed by Fabian Isensee and Klaus Maier-Hein in "An attempt at beating the 3D U-Net"). It uses three different 3D U-Net architectures that are composed of 3D convolutions, ReLU/LReLU non-linearities, and instance normalization. Upsampling is done with transposed convolution, and downsampling is done with strided convolutions. The encoder doubles the number of feature maps with each downsampling operation, while the decoder halves the number of feature maps with each upsampling operation. The resolution of the feature maps is also reduced by a factor of two until it is smaller than 4. The Plain U-Net architecture has 30 feature maps at the highest resolution, while the residual and pre-activation residual U-Nets start with 24 feature maps. The residual U-Nets use a series of conv-instnorm-ReLU-conv-instnorm-ReLU blocks and the pre-activation U-Net uses instnorm-ReLU-conv-instnorm-ReLU-conv blocks. This architecture won a notable Kidney Tumor Segmentation Challenge (KiTS2019) and was featured in the MICCAI 2019 conference.

We used this architecture as a starting point to investigate the possibilities of the segmentation networks for our problem at hand. The training was done by creating ground-truth segmentation maps from bounding boxes. Next, we used a widely adopted mixture of cross-entropy (CE) and dice similarity (Dice) losses to align the predicted and ground-truth segmentation maps. This strategy ensures a pixel-wise alignment with the CE loss as well as an image-wise alignment with the Dice loss.

2.2 Custom U-Net

Second, we tried our custom multi-path U-Net architecture which was introduced by Vilen Jumutc, Dmitrijs Bļizņuks and Alexey Lihachev in "Multi-Path U-Net Architecture for Cell and Colony-Forming Unit Image Segmentation". It consists of two receptive field pathways which have different pooling window sizes and the same strides for max pooling operations. All layers and their corresponding parameters were transferred from the "Plain U-Net" architecture. The number of filters for the first pathway was set to [40, 240] and [240, 40] for the up sampling path, and the second pathway was initialised as [40, 80, 160, 220] and [220, 160, 80, 40]. The max pooling window was either 4x4 or 2x2, depending on the pathway. All pathways have a middle (bottleneck) convolution block, which has either 220 or 240 filters.

This architecture is an improved version of the "Plain U-Net" architecture and has the advantage of multiple receptive field pathways which are capable of capturing the same image with different effective resolutions. We used the same combination of loss functions and dataset pre-processing techniques to come up with segmentation maps for the ground-truth bounding boxes. Example of Multi-Path U-Net predictions - see Figure X3. We show predicted segmentation maps as well as ground-truth and inferred (from segmentation maps) bounding boxes.

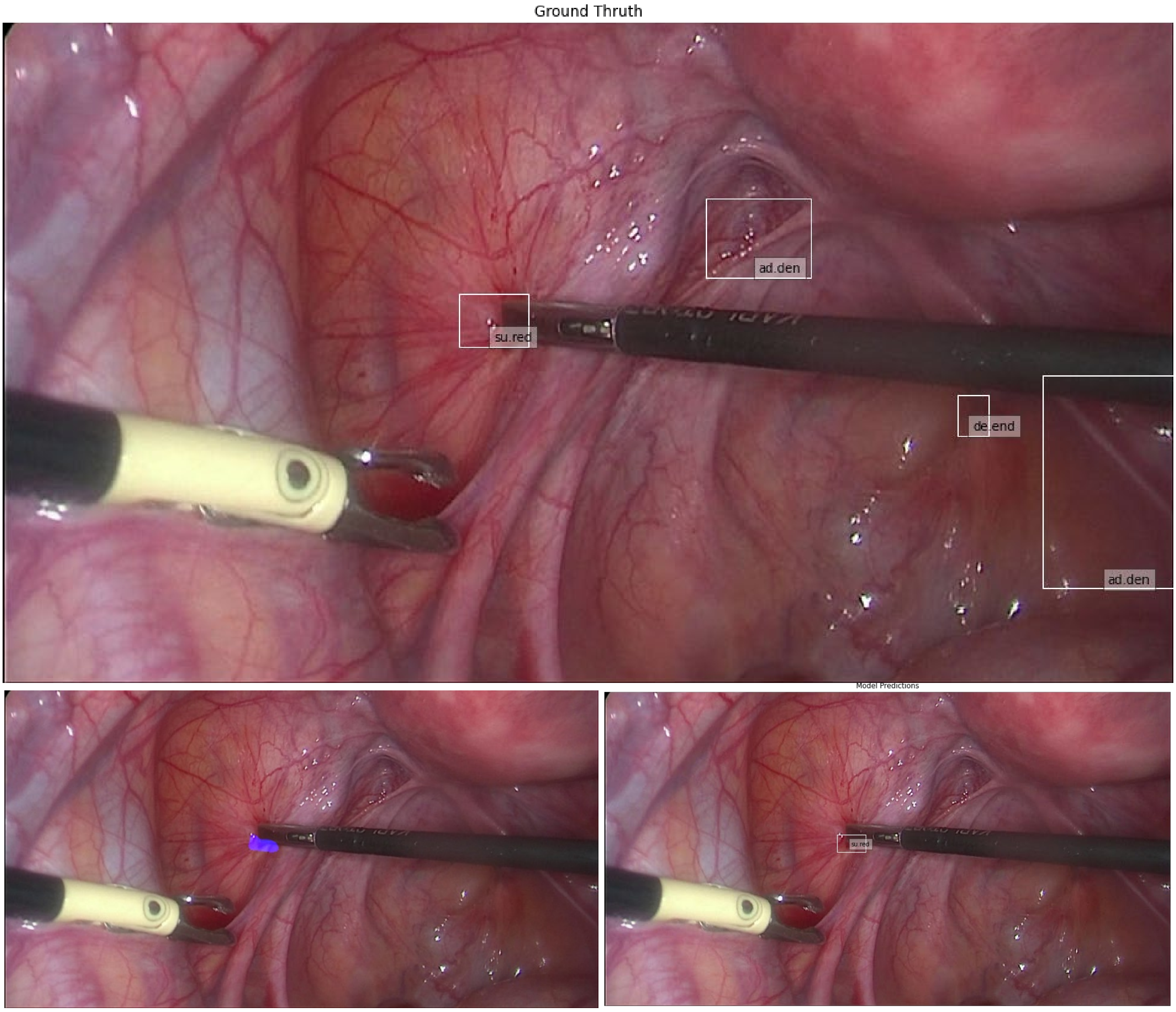


Figure 5. Predicted bounding box with segmentation map and 'ground truth' of the lesion.

2.3 FasterRCNN

The Faster R-CNN model (proposed by Shaoqing Ren et.al. in 2016) is a further improvement of the Fast R-CNN model (proposed by R. Girshick et.al. in 2015). It consists of two main parts: a Region Proposal Network (RPN) and a Fast R-CNN detector. The RPN takes an image as input and generates a set of object proposals, which are regions of the image likely to contain objects. The RPN achieves this by sliding a small network (typically a convolutional neural network) over the feature map produced by the input image and predicting a set of bounding boxes and objectness scores for each anchor box. These proposals are then fed into the Fast R-CNN detector.

The Fast R-CNN detector takes the object proposals from the RPN and extracts features from them using a Region of Interest (RoI) pooling layer. These features are fed into a fully connected network that outputs class probabilities and bounding box offsets for each RoI.

The RPN and the Fast R-CNN detector are trained end-to-end using a multi-task loss function that combines losses for objectness prediction, bounding box regression, and classification. Faster R-CNN is a widely used and effective object detection model, achieving state-of-the-art performance on many benchmark datasets.

In the scope of the bounding box detection task, we have utilized a model pre-trained on the COCO dataset and an improved version of the Faster R-CNN[7]. The model uses FasterRCNN + ResNet50 + FPN for the v2 variant with post-original-paper optimizations (no FrozenBN + c5 instead of p5 input on extra layers + heavier RPN/Box Heads with BNs). It improves the previous baseline by +9.7 mAP.

Data that are segmentations (instances of class Ovarian.Endometrioma[B]) where converted to bounding boxes and fed into the model along with the other data. Overall training time during training was approximately 1 hour for a single epoch on a single NVIDIA GeForce 1080Ti with training batch size set to 2.

2.4 MaskRCNN

Mask R-CNN (presented in "Mask R-CNN" by Kaiming He et.al. in 2018) is a deep learning model widely used for instance segmentation. Instance segmentation is the task of identifying objects within an image and delineating each object's boundaries with pixel-level accuracy. Mask R-CNN is an extension of the Faster R-CNN model.

The key idea behind Mask R-CNN is to add a branch to the Faster R-CNN model that predicts the object mask in parallel with the existing branch that predicts bounding boxes and class labels. This mask prediction branch consists of a series of convolutional layers and upsampling layers that take the feature maps produced by the backbone network and output a binary mask for each object in the image.

Mask R-CNN model consists of

- Backbone Network: Mask R-CNN uses a backbone network (e.g., ResNet) to extract feature maps from the input image.
- Region Proposal Network (RPN): The RPN identifies potential object locations in the feature maps and generates region proposals, which are candidate bounding boxes for objects in the image.
- Region of Interest (RoI) Pooling: The RoI pooling layer extracts a fixed-size feature map from each region proposal using a predefined spatial resolution. This fixed-size feature map is used as input to the mask and bounding box branches.
- Bounding Box Branch: The bounding box branch predicts the class label and bounding box coordinates for each region proposal.

[7] Benchmarking Detection Transfer Learning with Vision Transformers" by Yanghao Li et.al. 2021

- Mask Branch: The mask branch predicts a binary mask for each region proposal by using a set of convolutional layers and upsampling layers to generate a dense pixel-wise segmentation mask for the object.
- Loss Function: The loss function combines the losses from the bounding box branch and the mask branch and optimizes the model parameters to minimize the total loss.

By adding the mask branch to the Faster R-CNN model, Mask R-CNN is able to generate high-quality instance segmentations that accurately delineate the boundaries of objects in an image. This makes it a powerful tool for a wide range of computer vision applications, including object detection, semantic segmentation, and image manipulation.

In our experiments, we have trained Masked R-CNN from scratch, with ResNet50 + Feature Pyramid Network (FPN) as a backbone. Bounding box data was converted into segmentation format; thus, segmentations were rectangular regions, even for the Ovarian.Endometrioma[B] class which is represented by segmentations. This was done to mitigate training instability. Otherwise, we constantly observed exploding loss values and ceased of training. This behavior was repeatedly seen using various optimizers, optimization parameters, and learning rate scheduling strategies. All in all, by observing Dice coefficients, we can say that Faster R-CNN and Mask R-CNN are rather close to one another in terms of performance. Surprisingly both models have failed to recognize Adhesions.Dense class. Overall training time during training was approximately 2.5 hours for a single epoch on a single NVIDIA GeForce 1080 Ti with training batch size set to 4. All image transformations (resizing) were done inside the network model.

Overall, while Mask R-CNN and Faster R-CNN are powerful object detection models and first one can be used for segmentation in a wide range of applications, they may face some challenges when it comes to accurately segmenting objects within laparoscopic surgery images using just bounding boxes. Alternative segmentation methods will be explored further in WP7 to address these challenges and improve the accuracy of object segmentation in this context.

Examples of Mask R-CNN predictions - see figure 6.

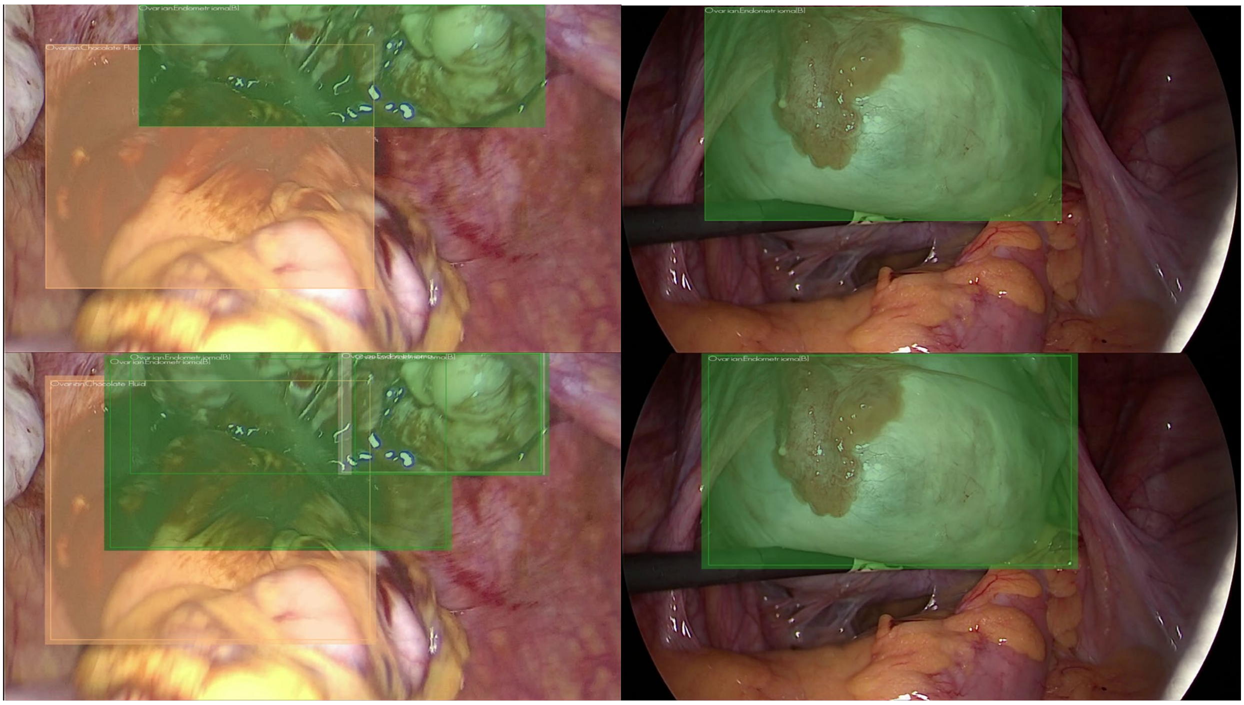


Figure 6: Examples of Mask R-CNN predictions (top - actual annotations, bottom-model predictions)

2.5 HarDNet-DFUS

HarDNet-DFUS was introduced in 2022 by Ting-Yu Liao et.al in "HarDNet-DFUS: An Enhanced Harmonically-Connected Network for Diabetic Foot Ulcer Image Segmentation and Colonoscopy Polyp Segmentation". This model has achieved SOTA results on the colonoscopy polyp segmentation dataset.

HarDNet-DFUS improves over HarDNet-MSEG, which backbone consists of basic building blocks called HarDBlock. The original backbone was upgraded and incorporated ideas from CSPNet and ShuffleNetV2, called HarDBlockV2. New architecture employs a new decoder introduced in the Lawin Transformer (see "Lawin transformer: Improving semantic segmentation transformer with multi-scale representations via large window attention" by Yan H. in 2022). It improves the capability of detecting ulcer regions and can deliver better accuracy compared to the original HarDNet-MSEG.

To achieve the best MACs over CIO ratio (MoC), channel splitting on the outputs of a convolutional layer l was performed according to the number of its output connections. This makes the number of input channels equal to the number of output channels for each Conv3x3 layer.

Bounding box data was converted into segmentation format; thus, segmentations were rectangular regions, even for the Ovarian.Endometrioma[B] class which is represented by segmentations. This was done to mitigate training instability. Input images were downscaled to 256x256 size - this is the default size used by the architecture. HarDnet-DFUS prediction examples can be seen in Fig 7.

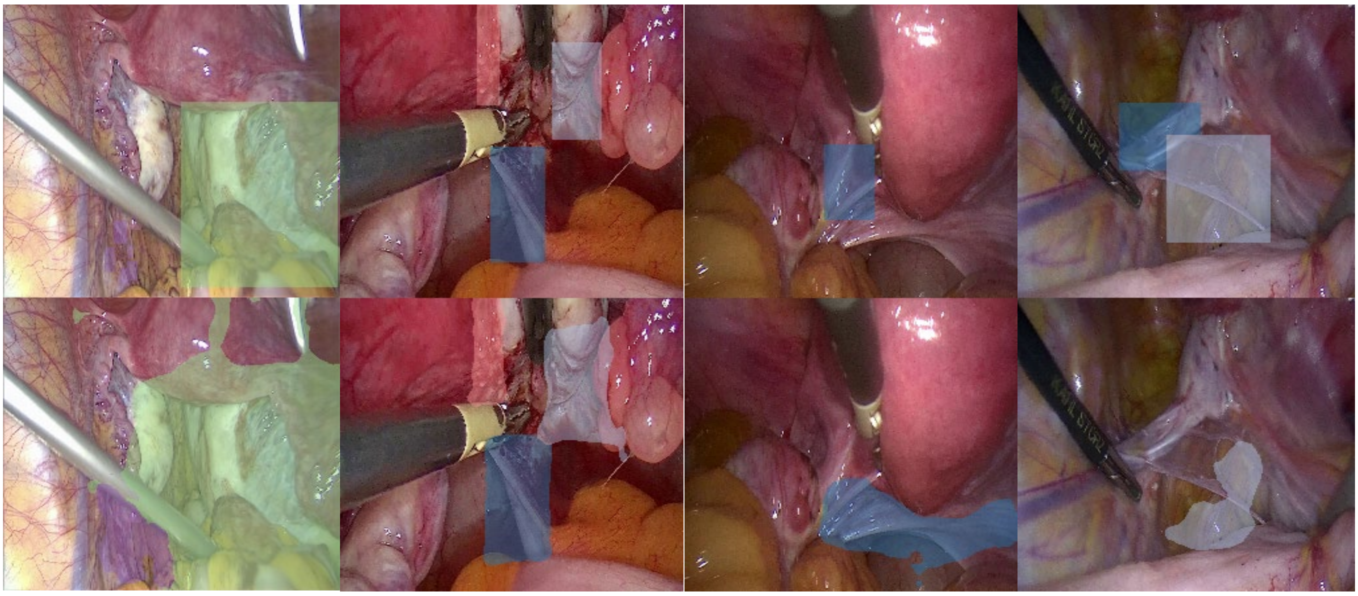


Figure 7: Examples of HarDNet-DFUS predictions (top-actual, bottom-predictions).

Overall training time during training was approximately 1.5 hours for a single epoch on a single NVIDIA GeForce 1080Ti with training batch size set to 8.

3. Models' evaluation

The results in table 1 below show the performance of different neural network architectures for lesion segmentation, measured using the Dice score. Where a score of 1 indicates perfect agreement between the model's output and the ground truth segmentation. The results suggest that Faster RCNN v2 achieved the highest Dice score of 0.905, indicating it performed the best for the test dataset used in the evaluation. Faster RCNN v2 is a convolutional neural network (CNN) architecture that uses region proposal networks (RPNs) to generate candidate object regions and then applies a classifier to those regions to classify them into specific classes.

Custom U-Net and HarDNet-DFUS also achieved relatively high Dice scores of 0.687 and 0.741, respectively. Custom U-Net is a modified version of the standard U-Net architecture. HarDNet-DFUS is a deep, fully convolutional neural network that was specifically designed for semantic segmentation tasks, such as image segmentation.

Plain U-Net and Mask RCNN v1 achieved lower Dice scores of 0.626 and 0.715. Plain U-Net is the original U-Net architecture without any modifications, while Mask RCNN v1 is a CNN architecture that extends Faster RCNN by adding a branch for predicting segmentation masks in parallel with the existing branch for object detection.

Table 1. Overall evaluation using test dataset with Dice coefficients for the best models across five architectures

Test Dice / Model	Adhesions. Dense	Adhesions. Filmy	Deep Endometriosis	Ovarian. Chocolate Fluid	Ovarian. Endometrioma[B]	Ovarian. Endometrioma	Superficial. Black	Superficial. Red	Superficial. Subtle	Superficial. White	Average
Plain U-Net	0.397	0.646	0.494	1.000	0.142	-	0.657	0.631	0.807	0.856	0.626
Custom U-Net	0.533	0.790	0.492	1.000	0.819	-	0.632	0.383	0.799	0.736	0.687
Faster RCNN v2	0.936	0.727	0.888	0.960	0.867	0.980	0.947	0.947	0.903	0.890	0.905
Mask RCNN v1	0.455	0.657	0.477	0.916	0.783	0.807	0.659	0.806	0.708	0.879	0.715
HarDNet-DFUS	0.573	0.607	0.658	0.913	0.773	0.500	0.803	0.911	0.780	0.892	0.741

As seen from Table 1, the best architecture has a lower score at some lesions, therefore ensemble of several best-performing networks could be used for more complex segmentation tasks in upcoming WP7.

To apply inference during real surgery model should be optimized. The current overall speed of the model inference was rather satisfactory and real-time or near-real-time inference speeds can be achieved using trained networks quantization techniques that will convert model from FP32 precision to INT8/16, in addition to pruning sessions can be applied to simplify the network structure, this will lead controllable performance degradation, but usually, at the expense of 0.5-1.5% of performance metric it is possible to improve the speed of inference significantly (2-4 and more times).

Since the end user of the system is a surgeon who is scrutinizing the frame sequence where just a few frames can have a bounding box with detected object classes, it can still be considered a fully acceptable detection.

We aim to introduce a video-tailored evaluation procedure (metrics) that can be applied across many evaluated object detection and segmentation models for a fair comparison. There are no specific guidelines on what metrics should be used to evaluate inference model quality during real-time use.

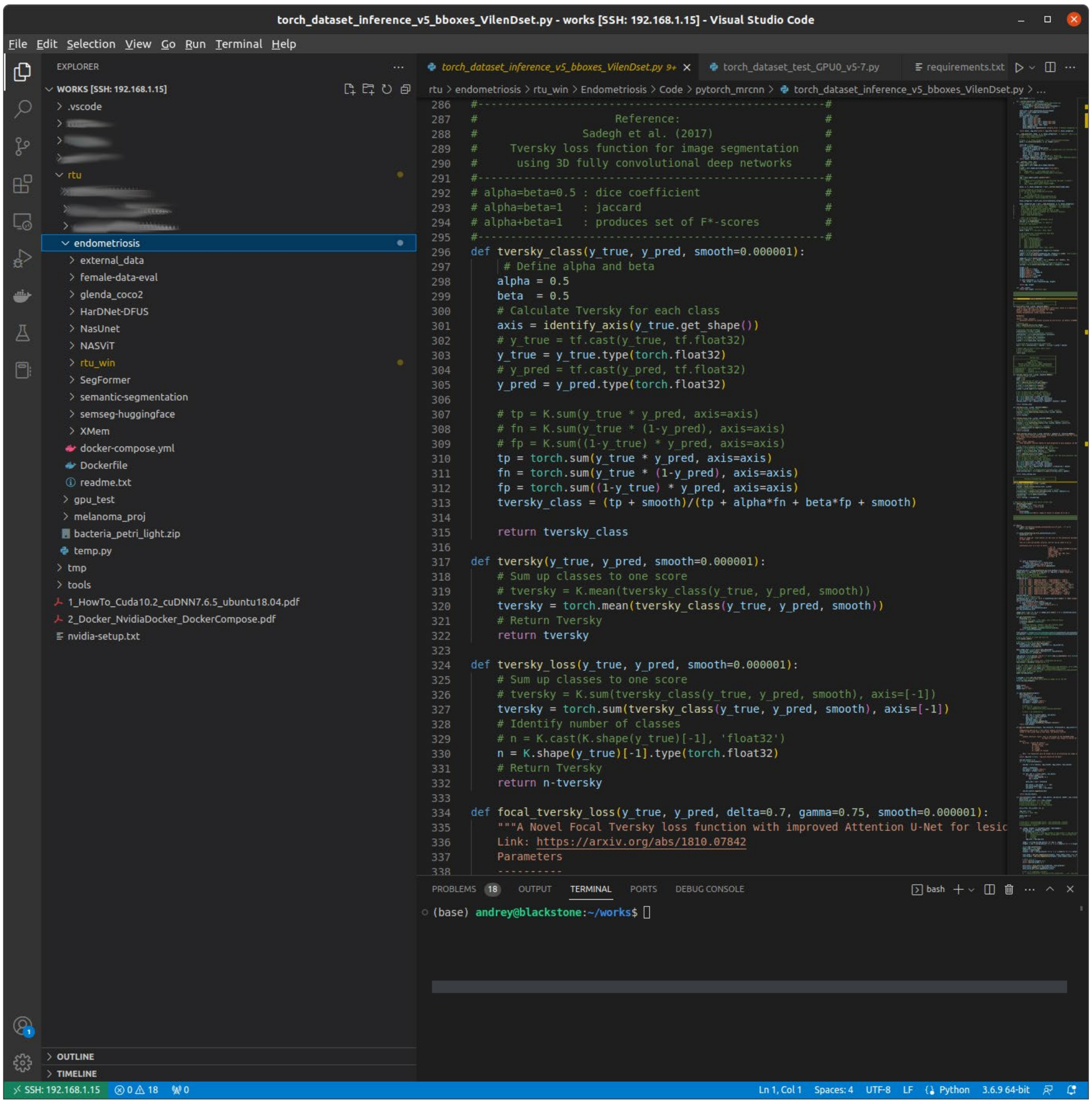
Therefore, we proposed a set of tuneable metrics which would be further used in a real usage scenario by the surgeons and optimised for a better usability of the system.

The following metrics are proposed:

1. Metrics of interest without Intersection over Union (IoU)
    a. Average precision per class across all frames (w/ ground-truth info)
    b. Average detection rate (recall) per class across all frames (w/ ground truth info)
    c. Average detection rate (recall) every 5/10/15/etc frames (window-based approach taking into account only those intervals w/ at least 1 ground-truth annotation)
2. Metrics of interest with IoU intervals
    a. For predefined intervals, e.g. [0.01 … 0.99], and steps e.g. 0.01 or 0.05 we evaluate the following metrics taking into account if the detected bounding box has an equal or larger IoU with the ground truth bounding box. No intersection here would mean 0 score.
    b. Average precision per class across all frames (w/ ground-truth info)
    c. Average detection rate (recall) per class across all frames (w/ ground-truth info)
    d. Average detection rate (recall) every 5/10/15/etc frames (window-based approach taking into account only those intervals w/ at least 1 ground-truth annotation)

In the end, we calculate and plot the statistics across all val/test videos with respect to the presented above video-tailored metrics. Fair comparisons should consider mean, standard deviation, and percentiles (plotted with the boxplot option within matplotlib for instance).

Appendix



torch_dataset_inference_v5_bboxes_VilenDset.py - works [SSH: 192.168.1.15] - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER
torch_dataset_inference_v5_bboxes_VilenDset.py 9+
torch_dataset_test_GPU0_v5-7.py
requirements.txt
rtu > endometriosis > rtu_win > Endometriosis > Code > pytorch_mrcnn > torch_dataset_inference_v5_bboxes_VilenDset.py > ...
PROBLEMS 18
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE
bash
OUTLINE
TIMELINE
SSH: 192.168.1.15
0 18
0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.6.9 64-bit

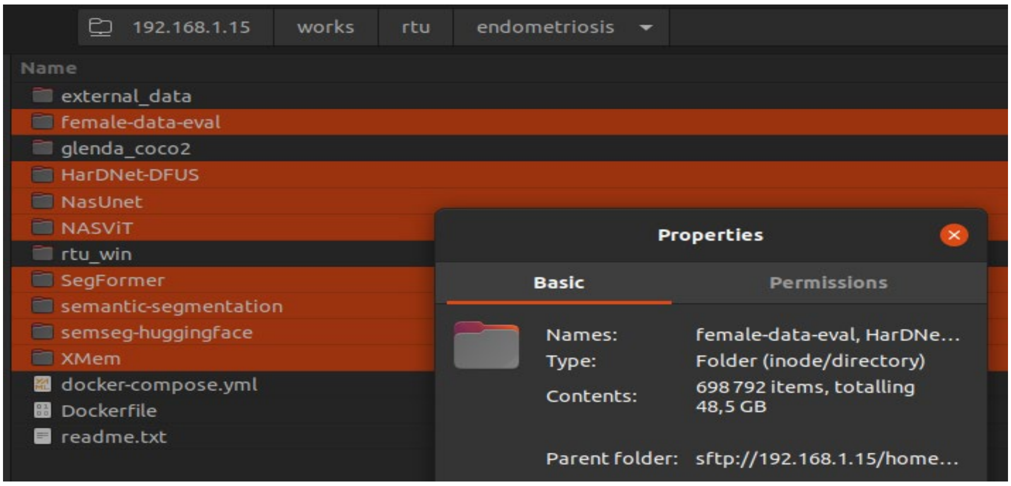
```
WORKS [SSH: 192.168.1.15]
  > .vscode
  [redacted]
  v rtu
    [redacted]
    v endometriosis
      > external_data
      > female-data-eval
      > glenda_coco2
      > HarDNet-DFUS
      > NasUnet
      > NASViT
      > rtu_win
      > SegFormer
      > semantic-segmentation
      > semseg-huggingface
      > XMem
      docker-compose.yml
      Dockerfile
      readme.txt
    > gpu_test
    > melanoma_proj
    bacteria_petri_light.zip
    temp.py
  > tmp
  > tools
  1_HowTo_Cuda10.2_cuDNN7.6.5_ubuntu18.04.pdf
  2_Docker_NvidiaDocker_DockerCompose.pdf
  nvidia-setup.txt
```

```python
#----------------------------------------------------------#
#                      Reference:                          #
#                 Sadegh et al. (2017)                     #
#     Tversky loss function for image segmentation         #
#      using 3D fully convolutional deep networks          #
#----------------------------------------------------------#
# alpha=beta=0.5 : dice coefficient                        #
# alpha=beta=1   : jaccard                                 #
# alpha+beta=1   : produces set of F*-scores               #
#----------------------------------------------------------#
def tversky_class(y_true, y_pred, smooth=0.000001):
     # Define alpha and beta
    alpha = 0.5
    beta  = 0.5
    # Calculate Tversky for each class
    axis = identify_axis(y_true.get_shape())
    # y_true = tf.cast(y_true, tf.float32)
    y_true = y_true.type(torch.float32)
    # y_pred = tf.cast(y_pred, tf.float32)
    y_pred = y_pred.type(torch.float32)

    # tp = K.sum(y_true * y_pred, axis=axis)
    # fn = K.sum(y_true * (1-y_pred), axis=axis)
    # fp = K.sum((1-y_true) * y_pred, axis=axis)
    tp = torch.sum(y_true * y_pred, axis=axis)
    fn = torch.sum(y_true * (1-y_pred), axis=axis)
    fp = torch.sum((1-y_true) * y_pred, axis=axis)
    tversky_class = (tp + smooth)/(tp + alpha*fn + beta*fp + smooth)

    return tversky_class

def tversky(y_true, y_pred, smooth=0.000001):
    # Sum up classes to one score
    # tversky = K.mean(tversky_class(y_true, y_pred, smooth))
    tversky = torch.mean(tversky_class(y_true, y_pred, smooth))
    # Return Tversky
    return tversky

def tversky_loss(y_true, y_pred, smooth=0.000001):
    # Sum up classes to one score
    # tversky = K.sum(tversky_class(y_true, y_pred, smooth), axis=[-1])
    tversky = torch.sum(tversky_class(y_true, y_pred, smooth), axis=[-1])
    # Identify number of classes
    # n = K.cast(K.shape(y_true)[-1], 'float32')
    n = K.shape(y_true)[-1].type(torch.float32)
    # Return Tversky
    return n-tversky

def focal_tversky_loss(y_true, y_pred, delta=0.7, gamma=0.75, smooth=0.000001):
    """A Novel Focal Tversky loss function with improved Attention U-Net for lesic
    Link: https://arxiv.org/abs/1810.07842
    Parameters
    ----------
```

```
(base) andrey@blackstone:~/works$ 
```

Figure A1: Explorer (left panel) shows overall **endometriosis** project folder contents with sub-projects.

The code editor (right panel) shows the code for one of the experiments.



```
external_data
female-data-eval
glenda_coco2
HarDNet-DFUS
NasUnet
NASViT
rtu_win
SegFormer
semantic-segmentation
semseg-huggingface
XMem
docker-compose.yml
Dockerfile
readme.txt
```

Names: female-data-eval, HarDNe…

Type: Folder (inode/directory)

Contents: 698 792 items, totalling 48,5 GB

Parent folder: sftp://192.168.1.15/home…

Figure A2: Shows the overall contents of the **endometriosis** project folder.

female-data-eval - project holds the Kedro pipelines for data format conversion and splitting.

HarDNet-DFUS folder hosts project for subject model training and estimation as well as experimental results.

NasUNet folder hosts project for subject model, multiple experiments were giving unsatisfactory results.

NasViT folder hosts project which memory requirements were too high for our hardware; thus, neural architecture search (NAS) was not explored (high memory requirements is a common issue with NAS algorithms).

rtu-win - is the folder for the main project under which custom and plain U-Net architectures were explored. **SegFormer**, **semantic-segmentation**, and **semseg-huggingface** are hosting segmentation models, only experiments with semseg-huggingface were set up and performed, but the results were rather unsatisfactory.

XMem hosts XMem model, which was explored, but abandoned as it was using the classical UNet model for video segmentation, and we had severe doubts about that particular model performance on our dataset (in contrast to HarDNet-DFUS segmentator). The large number of files is due to the large number of logs and model files produced by experiments and several sets of visualizations for train/test/val datasets for MaskR–CNN and HarDNet-DFUS models.