Semantic segmentation of EBHI-Seg dataset: Ablation Studies on UNet architecture

Vincenzo Fraello (339641)

Summary

- **Task description**;
- **Background description**:
  - Semantic Segmentation task;
  - U-Net Architecture;
  - EBHI-Seg paper;
  - Ablation studies;
- **Personal contributions**:
  - Training-phase: ablation studies related to training-choices;
  - Pruning: ablation studies related to modules' analysis.

Task

- **Project objective**: semantic segmentation of EBHI dataset of the 6 classes.
- **Dataset**: EBHI dataset
  - https://paperswithcode.com/dataset/ebhi-seg
  - https://figshare.com/articles/dataset/EBHI-SEG/21540159/1
- **Network model**: architecture like U-Net;
- **Detailed information**: make the ablation studies you prefer. For each ablation study:

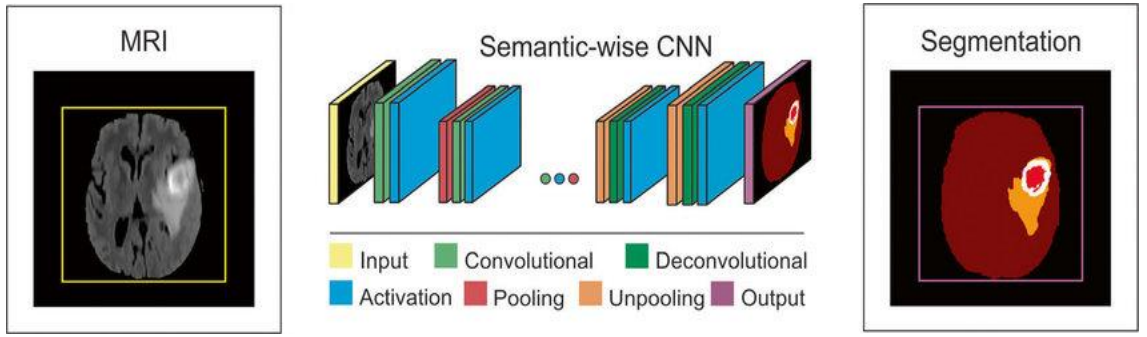
1. When you start the training, split the dataset randomly: 80% training, 20% test;
2. Train a model to make semantic segmentation of the images;
3. Collect performance results;
4. Repeat from point 1 at least 3 times;
5. Average JaccardIndex of the 3 (or more) runs.

Background

Semantic segmentation

- **Semantic-Segmentation** is a computer vision task in which the goal is to categorize each pixel in an image into a class or object. The goal is to produce a dense pixel-wise segmentation map of an image, where **each pixel** is **assigned** to a **specific class** or object. Models are usually evaluated with the Mean Intersection-Over-Union (Mean IoU) and Pixel Accuracy metrics.
  - Subtask: **tumor-segmentation** is the task of **identifying** the **spatial location of a tumor**. It is a pixel-level prediction where each pixel is classified as a tumor or background. The models are **typically** evaluated with the **Dice Score metric**.

https://paperswithcode.com/task/semantic-segmentation



Akkus, Zeynettin & Galimzianova, Alfiia & Hoogi, Assaf & Rubin, Daniel & Erickson, Bradley. (2017). Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions. Journal of digital imaging. 30. 10.1007/s10278-017-9983-4.

U-Net Architecture

- The architecture consists of a **contracting path** to **capture context** and a symmetric **expanding path** that enables **precise localization**. We show that such a network can be trained end-to-end from very few images and outperforms the prior best method (*sliding-window convolutional network*).
  - **Data augmentation** is essential to teach the network the desired invariance and **robustness properties**, when only few training samples are available. In case of microscopical images we primarily need shift and **rotation invariance** as well as **robustness to deformations** and **gray value variations**. Especially **random elastic deformations** (like Random rotation, Random scaling, Gaussian blur) of the training samples seem to be the key concept to train a segmentation network with very few annotated images.

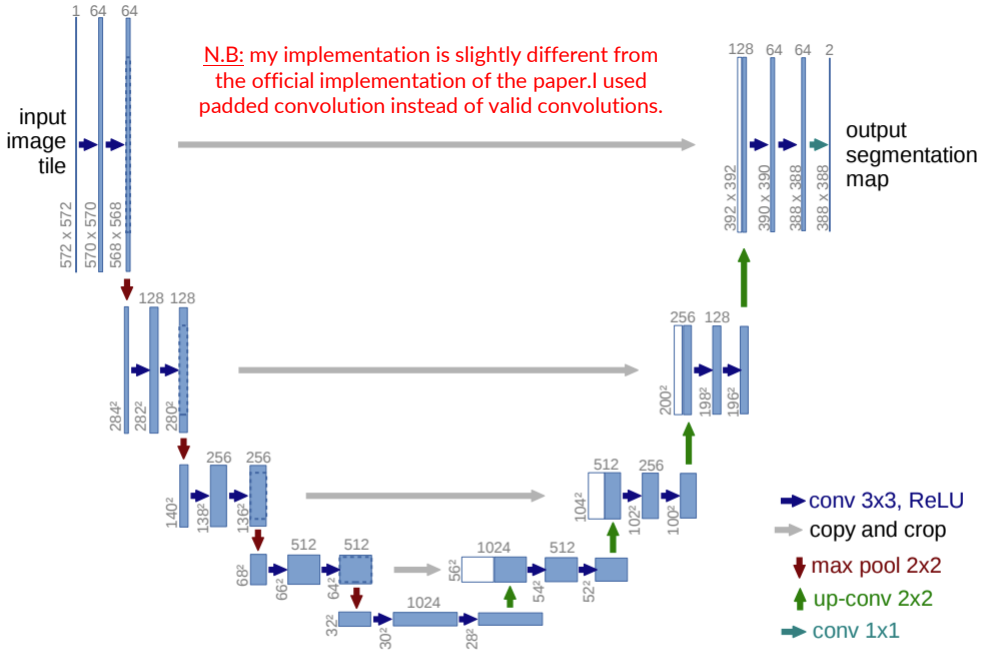


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. doi: 10.48550/ARXIV.1505.04597. url: https://arxiv.org/abs/1505.04597

EBHI-Seg paper (1)

- **Background and Purpose:** colorectal cancer is a common fatal malignancy, the fourth most common cancer in men, and the third most common cancer in women worldwide;
- **Timely detection** of cancer in its early stages is essential for treating the disease;
- Currently (*Submitted on 1 Dec 2022 (v1), last revised 6 Dec 2022 (this version, v3)*), there is a **lack of datasets for histopathological image segmentation of rectal cancer**, which often hampers the assessment accuracy when computer technology is used to aid in diagnosis;
- This present study provided a new publicly available **Enteroscope Biopsy Histopathological Hematoxylin and Eosin Image Dataset for Image Segmentation Tasks (EBHI-Seg)**. To demonstrate the validity and extensiveness of EBHI-Seg, the experimental results for EBHI-Seg are evaluated using **classical machine learning methods** and **deep learning methods**.

[1] Liyu Shi, Xiaoyan Li, Weiming Hu, Haoyuan Chen, Jing Chen, Zizhen Fan, Minghe Gao, Yujie Jing, Guotao Lu, Deguo Ma, Zhiyu Ma, Qingtao Meng, Dechao Tang, Hongzan Sun, Marcin Grzegorzek, Shouliang Qi, Yueyang Teng, Chen Li. EBHI-Seg: A Novel Enteroscope Biopsy Histopathological Haematoxylin and Eosin Image Dataset for Image Segmentation Tasks. 2022. arXiv. https://arxiv.org/abs/2212.00532.

EBHI-Seg paper (2)

- The experimental results showed that **deep learning methods had a better image segmentation performance** when utilizing EBHI-Seg. The maximum accuracy of the Dice evaluation metric for the classical machine learning method is 0.948, while the Dice evaluation metric for the deep learning method is **0.965**;
- This publicly available dataset contained **5,170 images** of **six types of tumor differentiation stages** and **the corresponding ground truth images**.

[1] Liyu Shi, Xiaoyan Li, Weiming Hu, Haoyuan Chen, Jing Chen, Zizhen Fan, Minghe Gao, Yujie Jing, Guotao Lu, Deguo Ma, Zhiyu Ma, Qingtao Meng, Dechao Tang, Hongzan Sun, Marcin Grzegorzek, Shouliang Qi, Yueyang Teng, Chen Li. EBHI-Seg: A Novel Enteroscope Biopsy Histopathological Haematoxylin and Eosin Image Dataset for Image Segmentation Tasks. 2022. arXiv. https://arxiv.org/abs/2212.00532.

- Actually:
  - 2228 images;
  - 2226 masks;
  - Total: 4454 items;
- After data cleaning:
  - 2226 images;
  - 2226 masks;
  - Total: 4452 items

```
Performing dataset cleaning...

Normal: 76, 76
Polyp: 474, 474
Low-grade IN: 639, 637
High-grade IN: 186, 186
Adenocarcinoma: 795, 795
Serrated adenoma: 58, 58
```

there are two images without correspondent masks

```
Performing dataset cleaning...

Class analysis: Normal...
Normal: 76,  76

Class analysis: Polyp...
Polyp: 474,  474

Class analysis: Low-grade IN...
       Removed image file: GTGT2012149-2-400-001.png
       Removed image file: GT2012149-2-400-001.png
Low-grade IN: 637,  637

Class analysis: High-grade IN...
High-grade IN: 186,  186

Class analysis: Adenocarcinoma...
Adenocarcinoma: 795,  795

Class analysis: Serrated adenoma...
Serrated adenoma: 58,  58

Data cleaning done...
```

EBHI-Seg paper: dataset details (1)

- It is prepared by 12 biomedical researchers according to the following rules:
  - Firstly, if there is only one differentiation stage in the image and the rest of the image is intact, then the differentiation stage became the image label;
  - Secondly, if there is more than one differentiation stage in the image, then the most obvious differentiation is selected as the image label; In general, the most severe and prominent differentiation in the image was used as the image label.
- Intestinal biopsy was used as the sampling method in this dataset. The magnification of the data slices is 400×, with an eyepiece magnification of 10× and an objective magnification of 40×. A Nissan Olympus microscope and NewUsbCamera acquisition software are used;
- The image input size is **224 × 224 pixels**, and the format is ***.png**. The data are grouped into five types described in detail in section 2.2.

EBHI-Seg paper: dataset details (2)

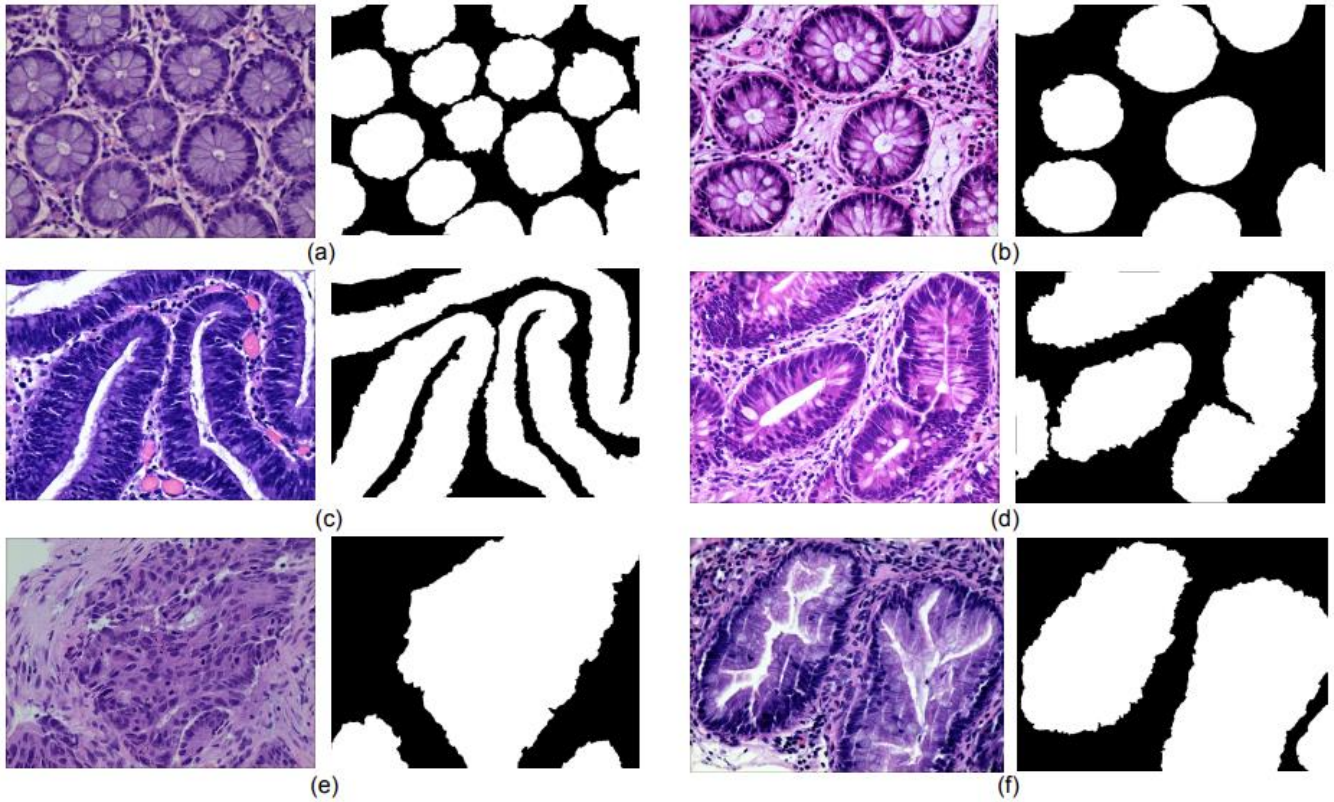


Figure 1: An example of histopathological images database: (a) Normal and ground truth, (b) Polyp and ground truth, (c) High-grade Intraepithelial Neoplasia and ground truth, (d) Low-grade Intraepithelial Neoplasia and ground truth, (e) Adenocarcinoma and ground truth, (f) Serrated adenoma and ground truth.

Classes:

- **Normal**;
- **Polyp**;
- **Intraepithelial neoplasia low grade**;
- **Intraepithelial neoplasia high grade**;
- **Adenocarcinoma**: tumor with a very irregular distribution of luminal structures. It is difficult to identify its border structures during observation;
- **Serrated adenoma**.

EBHI-Seg paper: evaluation metrics

$$\text{DiceRatio} = \frac{2\,|\text{X} \cap \text{Y}|}{|\text{X}| + |\text{Y}|}. \tag{1}$$

$$\text{JaccardIndex} = \frac{|\text{X} \cap \text{Y}|}{|\text{X} \cup \text{Y}|}. \tag{2}$$

$$\text{Precison} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{4}$$

EBHI-Seg paper: classical machine learning methods results

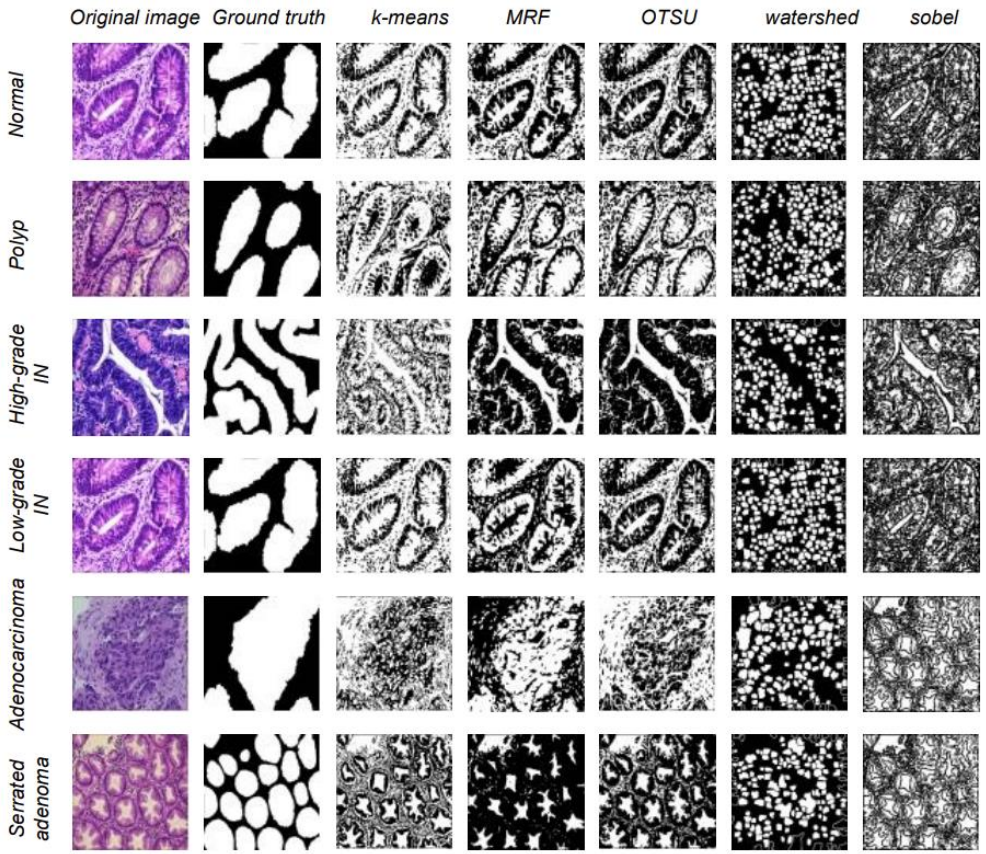


Figure 2: Five types of data segmentation results obtained by different classical machine learning methods.

Table 3: Evaluation metrics for five different segmentation methods based on classical machine learning.

		DiceRatio	JaccardIndex	ConformityCoefficient	Precision	Recall
Normal	k-means	0.648	0.488	-0.184	0.646	0.663
	MRF	0.636	0.473	-0.230	0.637	0.658
	OTSU	0.410	0.265	-2.871	0.515	0.351
	Watershed	0.461	0.300	-1.375	0.668	0.356
	Sobel	0.652	0.487	-0.102	0.763	0.579
Polyp	k-means	0.592	0.430	-0.528	0.546	0.663
	MRF	0.511	0.362	-2.133	0.540	0.502
	OTSU	0.400	0.259	-3.108	0.413	0.399
	Watershed	0.433	0.277	-1.675	0.551	0.362
	Sobel	0.583	0.416	-0.499	0.626	0.562
High-grade IN	k-means	0.626	0.478	-0.467	0.650	0.620
	MRF	0.550	0.441	-30.85	0.614	0.526
	OTSU	0.249	0.150	-12.06	0.373	0.191
	Watershed	0.472	0.309	-1.258	0.738	0.350
	Sobel	0.634	0.469	-0.200	0.728	0.577
Low-grade IN	k-means	0.650	0.492	-0.172	0.651	0.663
	MRF	0.554	0.404	-1.808	0.643	0.504
	OTSU	0.886	0.811	0.6998	0.832	0.979
	Watershed	0.464	0.303	-1.345	0.676	0.357
	Sobel	0.656	0.492	-0.079	0.771	0.582
Adenocarcinoma	k-means	0.633	0.481	-0.414	0.655	0.645
	MRF	0.554	0.404	-1.808	0.643	0.504
	OTSU	0.336	0.215	-5.211	0.454	0.282
	Watershed	0.458	0.298	-1.437	0.700	0.349
	Sobel	0.553	0.388	-0.733	0.692	0.484
Serrated adenoma	k-means	0.636	0.473	-0.230	0.637	0.658
	MRF	0.571	0.419	-0.898	0.656	0.547
	OTSU	0.393	0.248	-2.444	0.565	0.315
	Watershed	0.449	0.290	-1.494	0.656	0.345
	Sobel	0.698	0.541	0.7484	0.662	0.572

EBHI-Seg paper: deep learning methods results

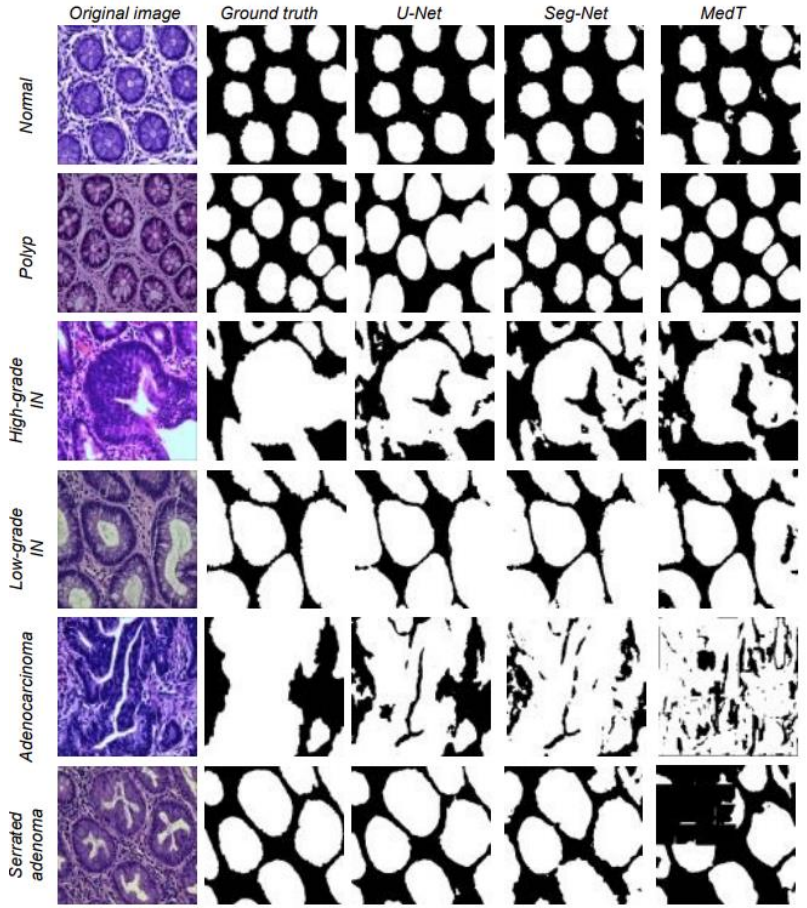


Figure 3: Three types of data segmentation results obtained by different deep learning methods.

Table 4: Evaluation metrics for three different segmentation methods based on deep learning.

		DiceRatio	JaccardIndex	ConformityCoefficient	Precision	Recall
Normal	U-Net	0.411	0.263	-2.199	0.586	0.328
	Seg-Net	0.777	0.684	-0.607	0.895	0.758
	MedT	0.676	0.562	-0.615	0.874	0.610
Polyp	U-Net	0.965	0.308	-1.514	0.496	0.470
	Seg-Net	0.937	0.886	0.858	0.916	0.965
	MedT	0.771	0.643	0.336	0.687	0.920
High-grade IN	U-Net	0.895	0.816	0.747	0.847	0.961
	Seg-Net	0.894	0.812	0.757	0.881	0.913
	MedT	0.824	0.707	0.556	0.740	0.958
Low-grade IN	U-Net	0.911	0.849	0.773	0.879	0.953
	Seg-Net	0.924	0.864	0.826	0.883	0.977
	MedT	0.889	0.808	0.730	0.876	0.916
Adenocarcinoma	U-Net	0.887	0.808	0.718	0.850	0.950
	Seg-Net	0.865	0.775	0.646	0.792	0.977
	MedT	0.735	0.595	0.197	0.662	0.864
Serrated adenoma	U-Net	0.938	0.886	0.865	0.899	0.983
	Seg-Net	0.907	0.832	0.794	0.859	0.963
	MedT	0.670	0.509	-0.043	0.896	0.544

Ablation studies

- A **surgical procedure** used to **understand** the **role** of **different components** of the **brain**. An ablation study involves **removing** a specific part of the brain of a mammal and observing any resulting changes in its behavior;
- It is a **black-box** approach;
- Experiments that **modify model architectures** while observing system performance offer an approach to help improve our understanding of a DL system.
- An ablation study in DL involves measuring the performance of a network after removing one or more of its components to help understand the relative contribution of the ablated components to overall performance. Dataset features and model components (e.g., layers) are notable examples of ablatable components, but **any design choice** or **module** of the system can be considered in an ablation study.

[1] Sina Sheikholeslami, Moritz Meister, Tianze Wang, Amir H. Payberah, Vladimir Vlassov, and Jim Dowling. 2021. AutoAblation: Automated Parallel Ablation Studies for Deep Learning. In Proceedings of the 1st Workshop on Machine Learning and Systems (EuroMLSys '21). Association for Computing Machinery, New York, NY, USA, 55–61. https://doi.org/10.1145/3437984.3458834

Personal contributions

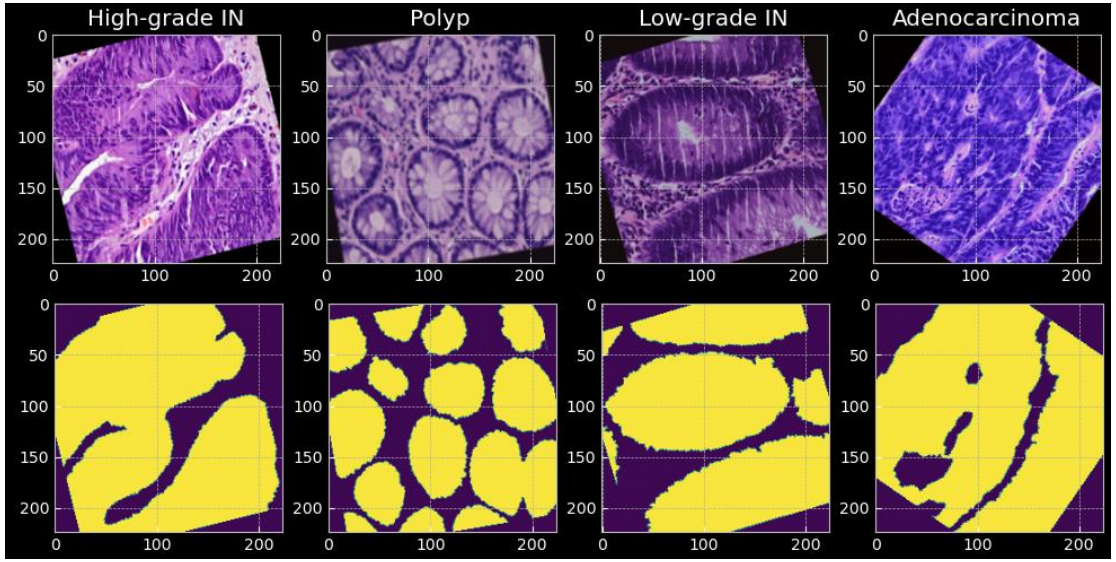
Training choices (1)

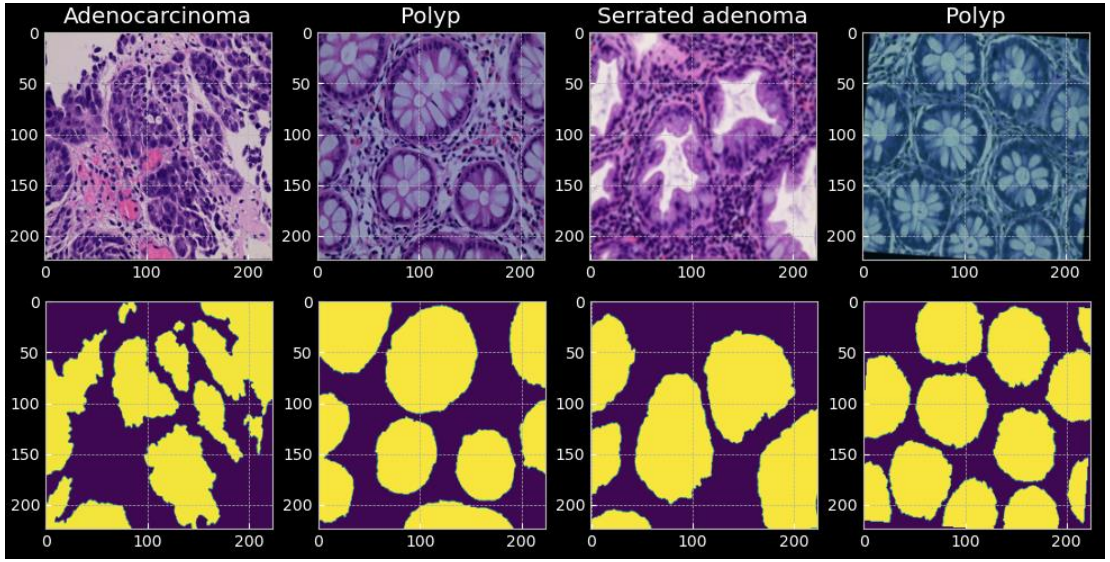
- Before carrying out the ablation studies related to the architecture, I tried to obtain the same results as in the original paper;
- I had to make **design decisions** for network-training:
  - Training batch-size;
  - Validation batch-size;
  - Loss function;
  - Optimizer;
  - Early-stopping value;
  - **Architectural decisions**:
    - Batch Normalization, Double Batch Normalization;
    - Instance Normalization, Double Instance Normalization;
    - Weights Initialization;
    - Number of filters in each convolutional layer (width);
    - U-Depth (network depth).

Training choices (2)

- Before carrying out the ablation studies I tried to obtain the same results as in the original paper;
- I had to make decisions for **training-data**:
  - Random seed used to generate different well proportioned-datasets;
  - Input normalization;
  - Apply transformations to training images;
  - Data augmentation;
  - Balancing the training-set;

Data transformation

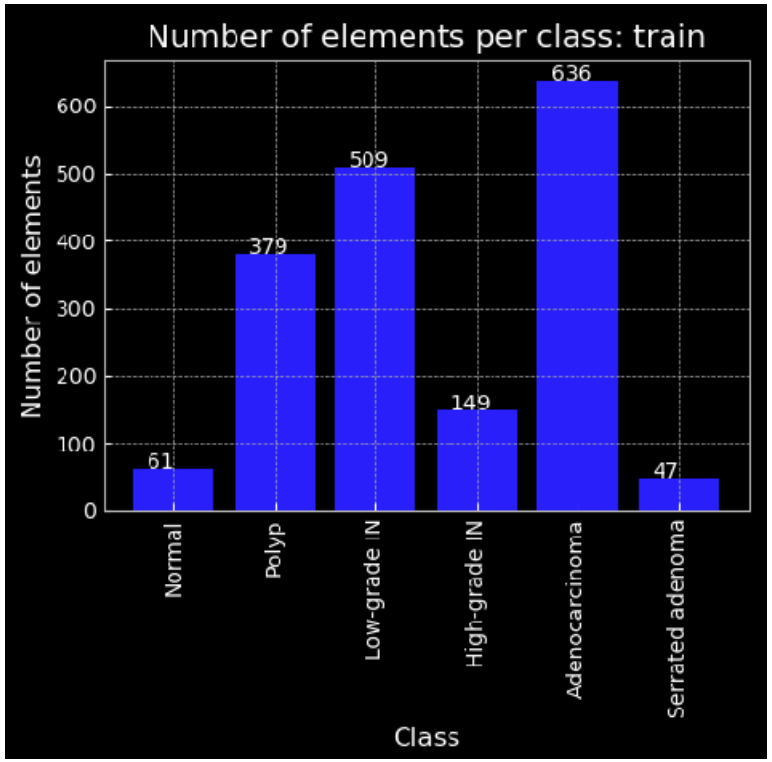


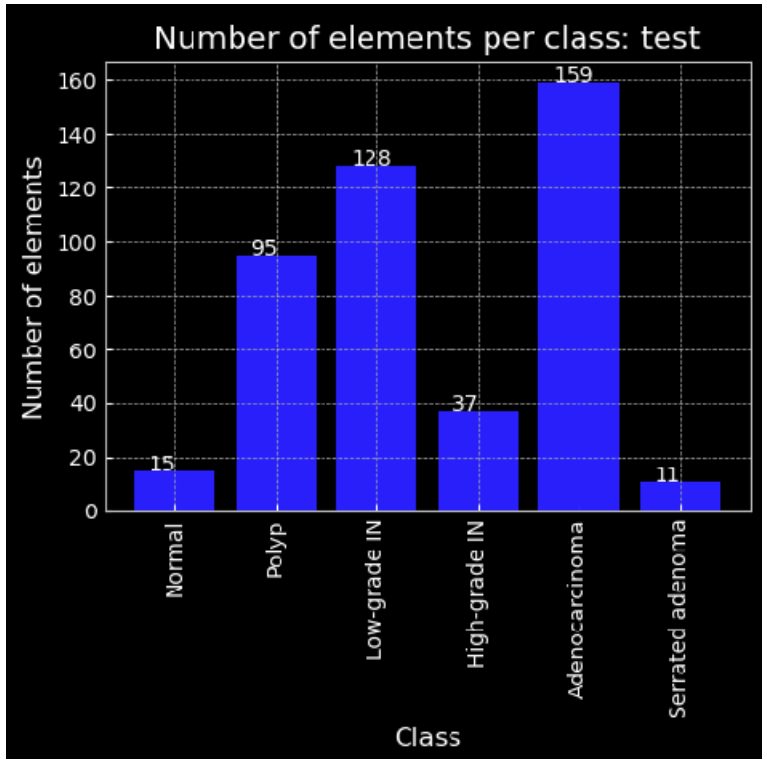


- **Rotations**;
- **Horizontal flips**;
- **Vertical flips**;
- **RGBShift** (perception of movement or color distortion);
- **Blur** (blur details in an image);
- **ColorJitter** (modification of the following image parameters: brightness, contrast, saturation, hue).

Training choices (3)

- Dataset split: 80% training, 20% validation (test)





Training choices (3)

- **Balanced training-set:** *WeightedRandomSampler* to sample from dataset such that the model **sees** approximately each class the **same number of times**.
  - For each class, a weight is calculated as:

$$\frac{1}{number\ of\ examples\ in\ the\ class}$$

  - Then, for each item in the training set, the weight of the corresponding class is assigned. The smaller the number of elements in the class, the higher the value of the weight and therefore the probability of being selected during training.

Collected performance results: top-5 best (1)

```
Getting the best network configuration...

1) Configuration: ./statistics\my_dic_train_results_unet_3_0_2t_22032023-114243.json,
                mean-test-loss: 0.0556, mean-train-loss: 0.0411, abs-diff: 0.0145
                mean-test per-class-accuracy: [0.96082324 0.9660707  0.95802736 0.90883946 0.89879555 0.92278576], total: 0.9359
                mean-test per-class-precision: [0.957001   0.9694553  0.9636534  0.92790174 0.9208002  0.90831953], total: 0.9412
                mean-test per-class-recall: [0.97673416 0.96680903 0.97201043 0.9395137  0.92771083 0.9687245 ], total: 0.9586


2) Configuration: ./statistics\my_dic_train_results_unet_13_0_2t_23032023-030252.json,
                mean-test-loss: 0.0564, mean-train-loss: 0.0479, abs-diff: 0.0085
                mean-test per-class-accuracy: [0.9578037  0.96387553 0.9576985  0.9134262  0.89498377 0.93811363], total: 0.9377
                mean-test per-class-precision: [0.97444254 0.97995    0.9777157  0.9525688  0.9458925  0.9499728 ], total: 0.9634
                mean-test per-class-recall: [0.9551084  0.9537988  0.9582808  0.9238072  0.89834833 0.9506102 ], total: 0.9400


3) Configuration: ./statistics\my_dic_train_results_unet_13_1_2t_23032023-044200.json,
                mean-test-loss: 0.0572, mean-train-loss: 0.0521, abs-diff: 0.0051
                mean-test per-class-accuracy: [0.9614052  0.9658466  0.96034926 0.90146714 0.89005303 0.93675834], total: 0.9360
                mean-test per-class-precision: [0.9622483  0.97343093 0.9713786  0.9430364  0.92752385 0.9352564 ], total: 0.9521
                mean-test per-class-recall: [0.97174305 0.9627779  0.9677306  0.9164471  0.90992135 0.9636958 ], total: 0.9487


4) Configuration: ./statistics\my_dic_train_results_unet_4_0_2t_22032023-145007.json,
                mean-test-loss: 0.0578, mean-train-loss: 0.0520, abs-diff: 0.0059
                mean-test per-class-accuracy: [0.9607715  0.96417284 0.9584944  0.9040205  0.8913803  0.9324644 ], total: 0.9352
                mean-test per-class-precision: [0.9642415  0.97552663 0.9696622  0.9362466  0.9213207  0.9249571 ], total: 0.9487
                mean-test per-class-recall: [0.97002417 0.9577624  0.9671044  0.9247038  0.91662884 0.96600443], total: 0.9504


5) Configuration: ./statistics\my_dic_train_results_unet_3_2_2t_22032023-134126.json,
                mean-test-loss: 0.0581, mean-train-loss: 0.0476, abs-diff: 0.0105
                mean-test per-class-accuracy: [0.9593813  0.9637707  0.9572832  0.9055934  0.89004105 0.91982585], total: 0.9326
                mean-test per-class-precision: [0.96758676 0.977279   0.9708979  0.9391184  0.93600535 0.91594934], total: 0.9511
                mean-test per-class-recall: [0.9649064  0.9557269  0.96401626 0.9251621  0.90446156 0.95699376], total: 0.9452
```

Top-5 best: configurations

```
unet_3_0_2t:
    - random_seed 2
    - opt Adam
    - arc_change_net
    - use_inst_norm
    - early_stopping 15
```

```
unet_13_0_2t:
    - random_seed 2
    - opt Adam
    - arc_change_net
    - use_double_batch_norm
    - bs_train 16
    - dataset_aug 3
    - early_stopping 6
```

```
unet_13_1_2t:
    - random_seed 2
    - opt Adam
    - arc_change_net
    - use_double_batch_norm
    - bs_train 16
    - dataset_aug 3
    - lr 0.001
    - weights_init
    - early_stopping 6
```

```
unet_4_0_2t:
    - random_seed 2
    - opt Adam
    - arc_change_net
    - use_inst_norm
    - early_stopping 7
    - bs_train 1
```

```
unet_3_2_2t:
    - random_seed 2
    - opt Adam
    - lr 0.001
    - arc_change_net
    - use_inst_norm
    - weights_init
    - early_stopping 15
```

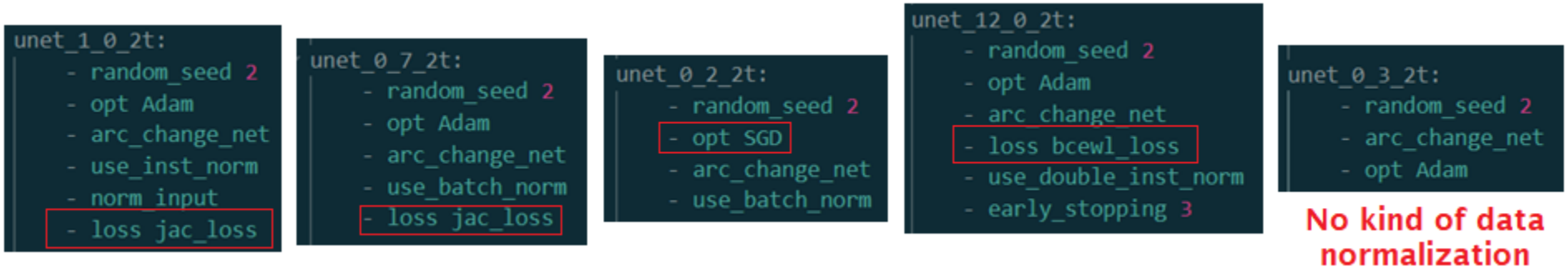
Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 2017. arXiv. url: https://arxiv.org/abs/1412.6980

Collected performance results: top-5 worst

```
31) Configuration: ./statistics\my_dic_train_results_unet_1_0_2t_22032023-075328.json,
                mean-test-loss: 0.1152, mean-train-loss: 0.0977, abs-diff: 0.0175
                mean-test per-class-accuracy: [0.9544794  0.9608493  0.9532164  0.8933644  0.87455887 0.9088233 ], total: 0.9242
                mean-test per-class-precision: [0.96472764 0.97609884 0.97211456 0.9497139  0.9493306  0.93853086], total: 0.9584
                mean-test per-class-recall: [0.9606102  0.952691   0.95753527 0.90352243 0.87733215 0.92647123], total: 0.9297


32) Configuration: ./statistics\my_dic_train_results_unet_0_7_2t_22032023-051959.json,
                mean-test-loss: 0.1217, mean-train-loss: 0.1058, abs-diff: 0.0158
                mean-test per-class-accuracy: [0.94364023 0.95757985 0.9473395  0.89583206 0.8706917  0.9024126 ], total: 0.9196
                mean-test per-class-precision: [0.96544486 0.9801762  0.97031945 0.93793416 0.92847013 0.9299119 ], total: 0.9520
                mean-test per-class-recall: [0.94529986 0.9436862  0.95144224 0.91656923 0.88999915 0.9244917 ], total: 0.9286


33) Configuration: ./statistics\my_dic_train_results_unet_0_2_2t_22032023-021637.json,
                mean-test-loss: 0.1751, mean-train-loss: 0.1563, abs-diff: 0.0188
                mean-test per-class-accuracy: [0.7530825  0.784849   0.8199859  0.80365795 0.7597177  0.74772435], total: 0.7782
                mean-test per-class-precision: [0.8009417 0.8831718 0.8797377 0.8987853 0.8386858 0.7972679], total: 0.8498
                mean-test per-class-recall: [0.8037285  0.7722273  0.8595819  0.83321637 0.8164253  0.81363916], total: 0.8165


34) Configuration: ./statistics\my_dic_train_results_unet_12_0_2t_23032023-011927.json,
                mean-test-loss: 0.1899, mean-train-loss: 0.1675, abs-diff: 0.0224
                mean-test per-class-accuracy: [0.4994526  0.5003187  0.5005493  0.49965796 0.49975565 0.5000634 ], total: 0.5000
                mean-test per-class-precision: [0.4997543  0.49941143 0.49982443 0.49898556 0.49841857 0.5008283 ], total: 0.4995
                mean-test per-class-recall: [0.5977752  0.54047614 0.6603188  0.674598   0.66374665 0.63721675], total: 0.6290


35) Configuration: ./statistics\my_dic_train_results_unet_0_3_2t_22032023-024921.json,
                mean-test-loss: 0.2337, mean-train-loss: 0.2251, abs-diff: 0.0086
                mean-test per-class-accuracy: [0.59801143 0.5399806  0.6599335  0.6747133  0.66343164 0.63731843], total: 0.6289
                mean-test per-class-precision: [1. 1. 1. 1. 1. 1.], total: 1.0000
                mean-test per-class-recall: [0.59801143 0.5399806  0.6599335  0.6747133  0.66343164 0.63731843], total: 0.6289
```

Top-5 worst: configurations



Tensorboard charts & Inference (1)

- **Best configuration**: unet_13_1_2t
  - Low mean-test loss;
  - Lower absolute difference between mean-test-loss and mean-train-loss;
  - Best trade-off between accuracy, precision and recall.
- The following studies involve variations on:
  - Optimizer;
  - Training-BS;
  - Use of IN;
  - Use of BN;
  - Combinations of normalizations;
  - No-normalization;
  - Choice of loss function;
  - Early stopping values;
  - LR values`(default=0.0001)`.

Tensorboard charts & Inference (2)

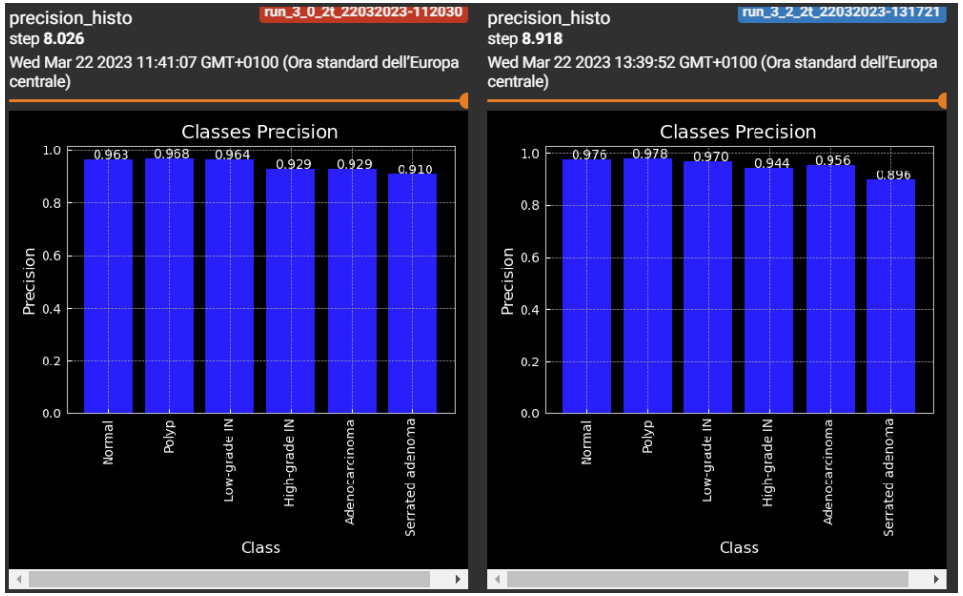
- Top-5 best: test losses

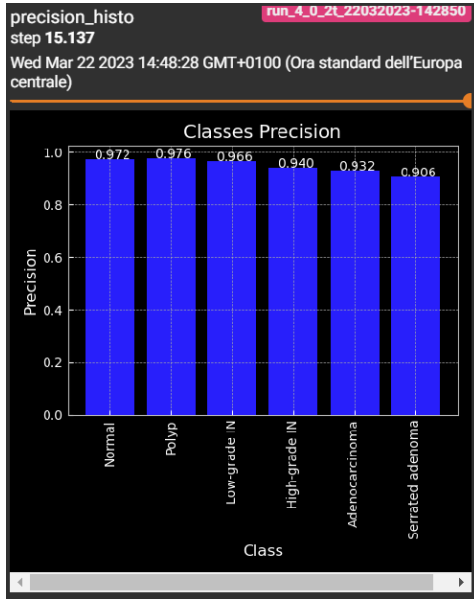
Tensorboard charts & Inference (2.1)

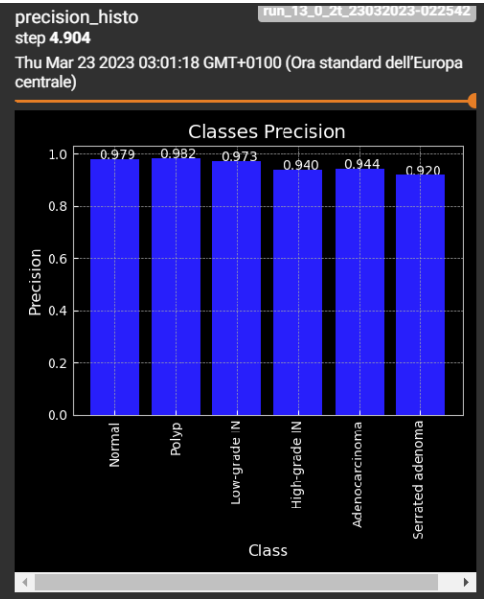
- Top-5 best: accuracy

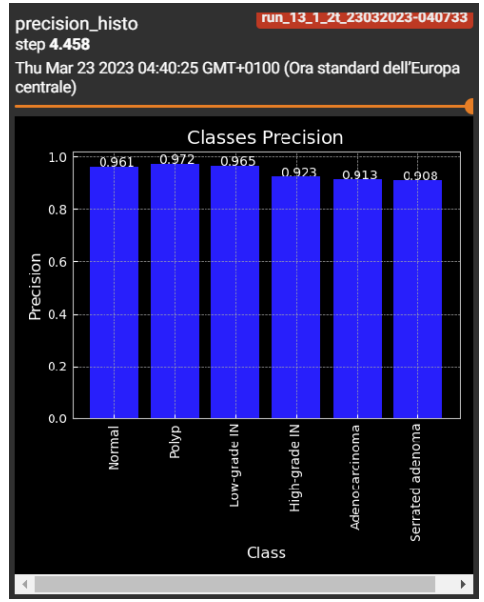
Tensorboard charts & Inference (2.2)

- Top-5 best: precision



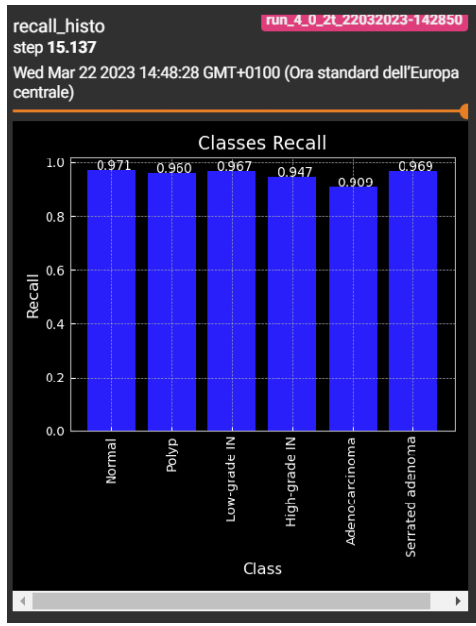


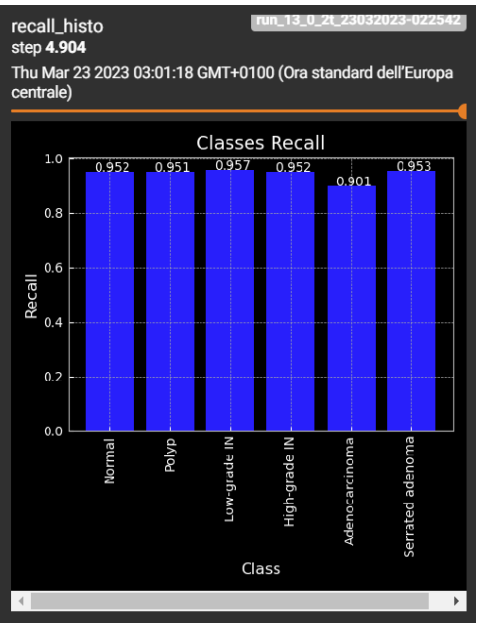


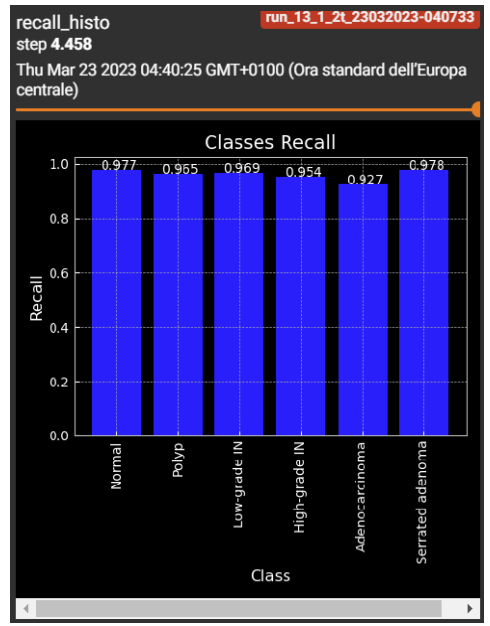


Tensorboard charts & Inference (2.3)

- Top-5 best: recall





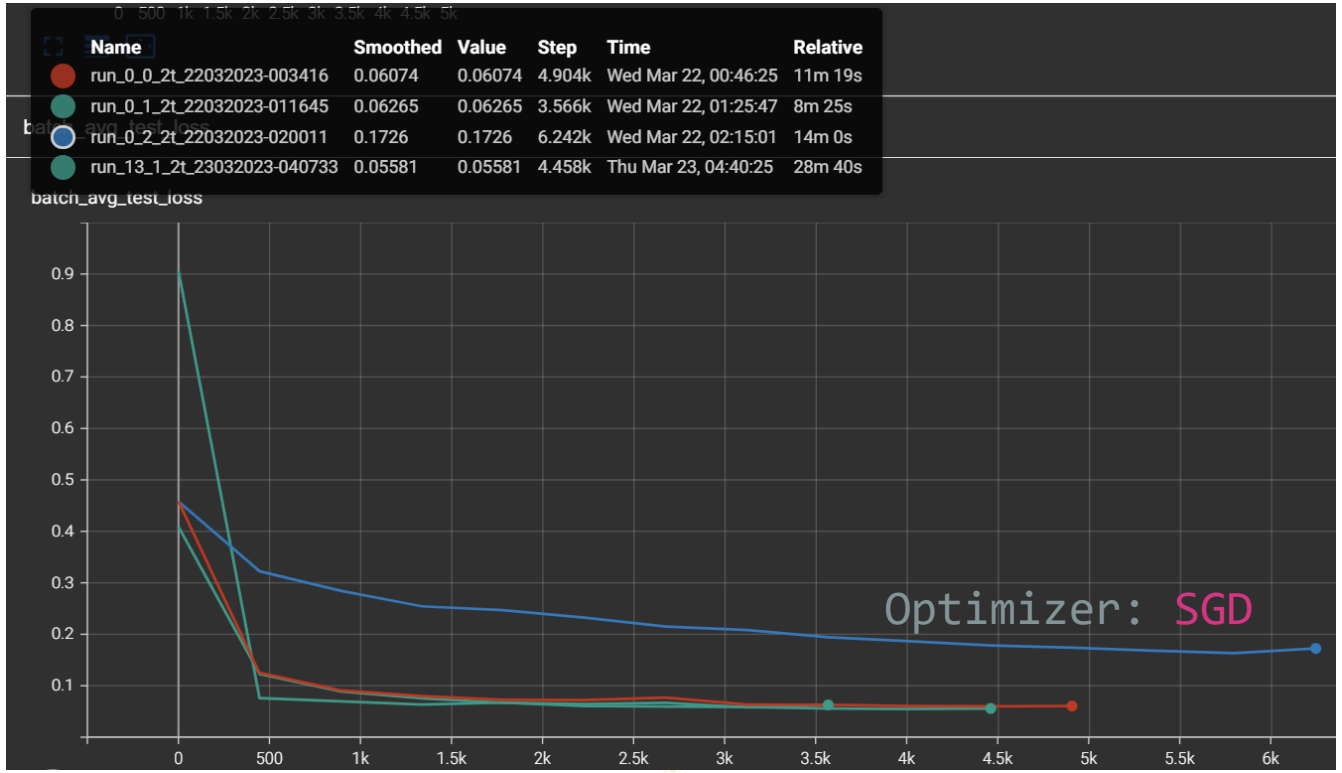


Training choises: inference (9)

- Comparing the best configurations with the initial ones shows small improvements within the range [0.05, 0.06].

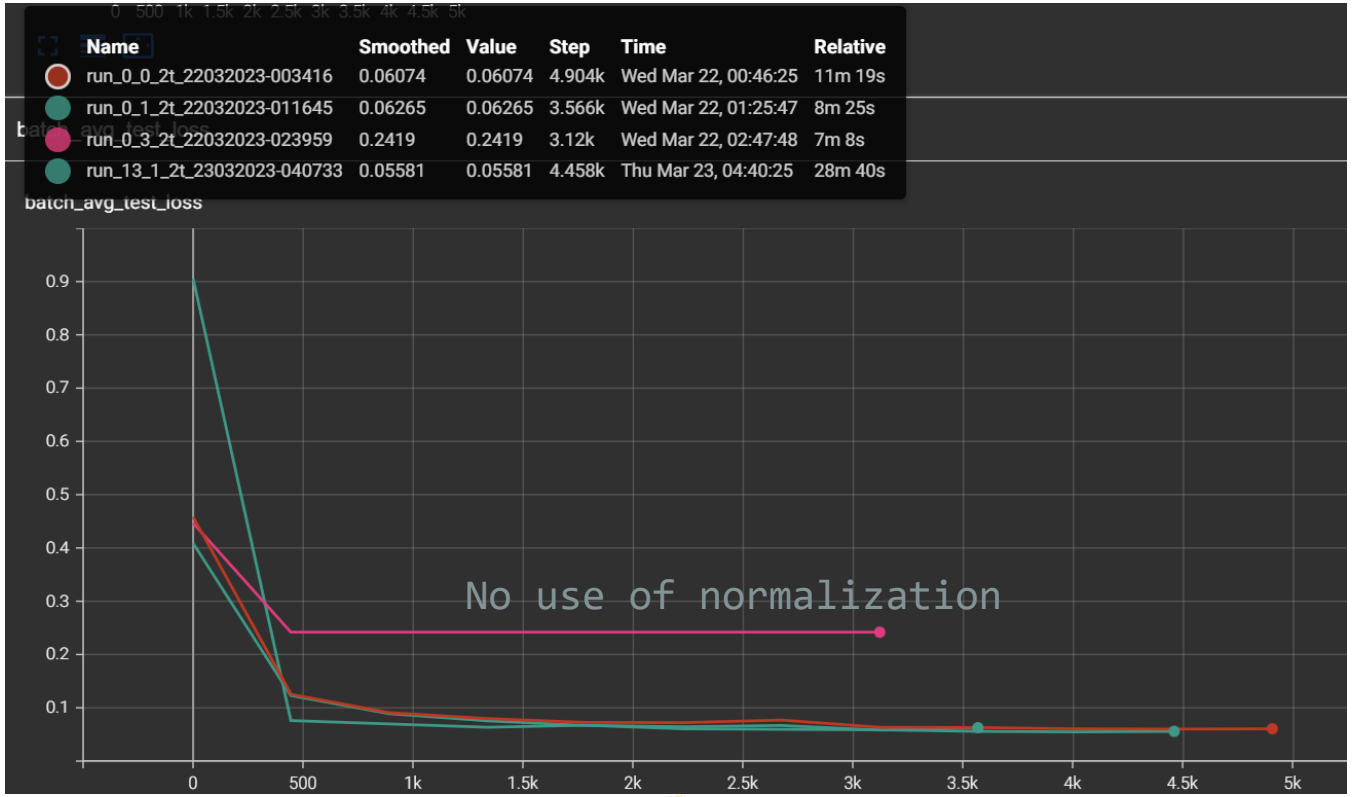
Training choises: inference (3)

- Comparison between configurations that make use of Adam optimizer and BN/IN vs. the configuration that makes use of SGD optimizer.



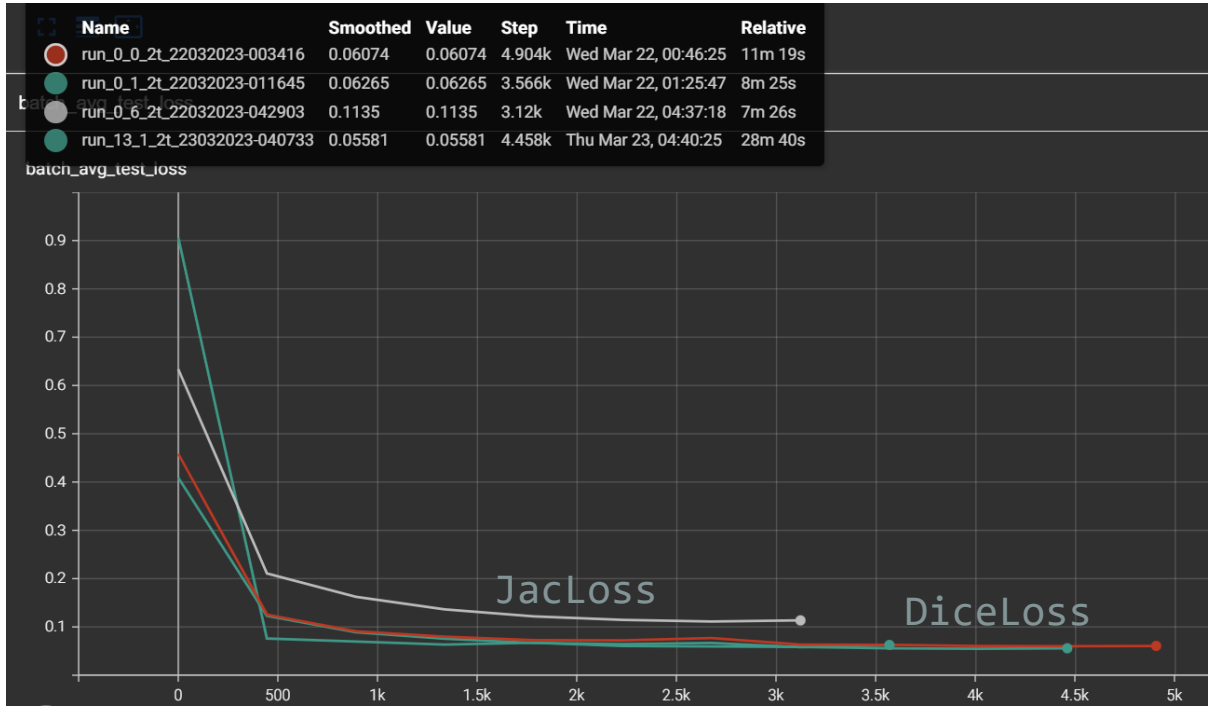
Training choises: inference (4)

- Comparison between configurations that make use of Adam optimizer and BN/IN vs. the configuration that make use of Adam optimizer but does not make use of any kind of normalization.



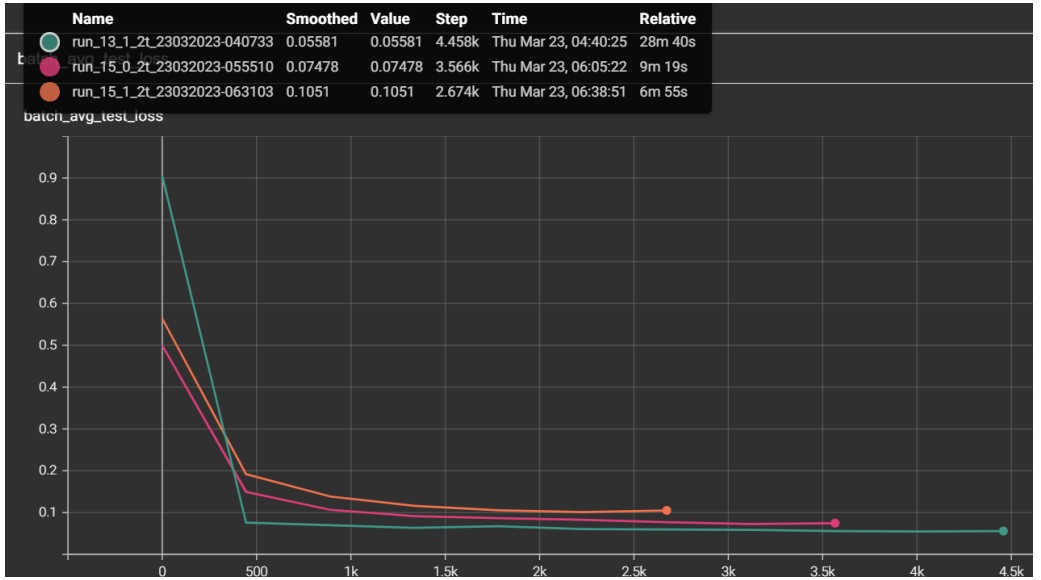
Training choises: inference (5)

- Applying both input normalization + BN or both input normalization + IN does not increase performances;
- DiceLoss is better than JacLoss for the training-process.



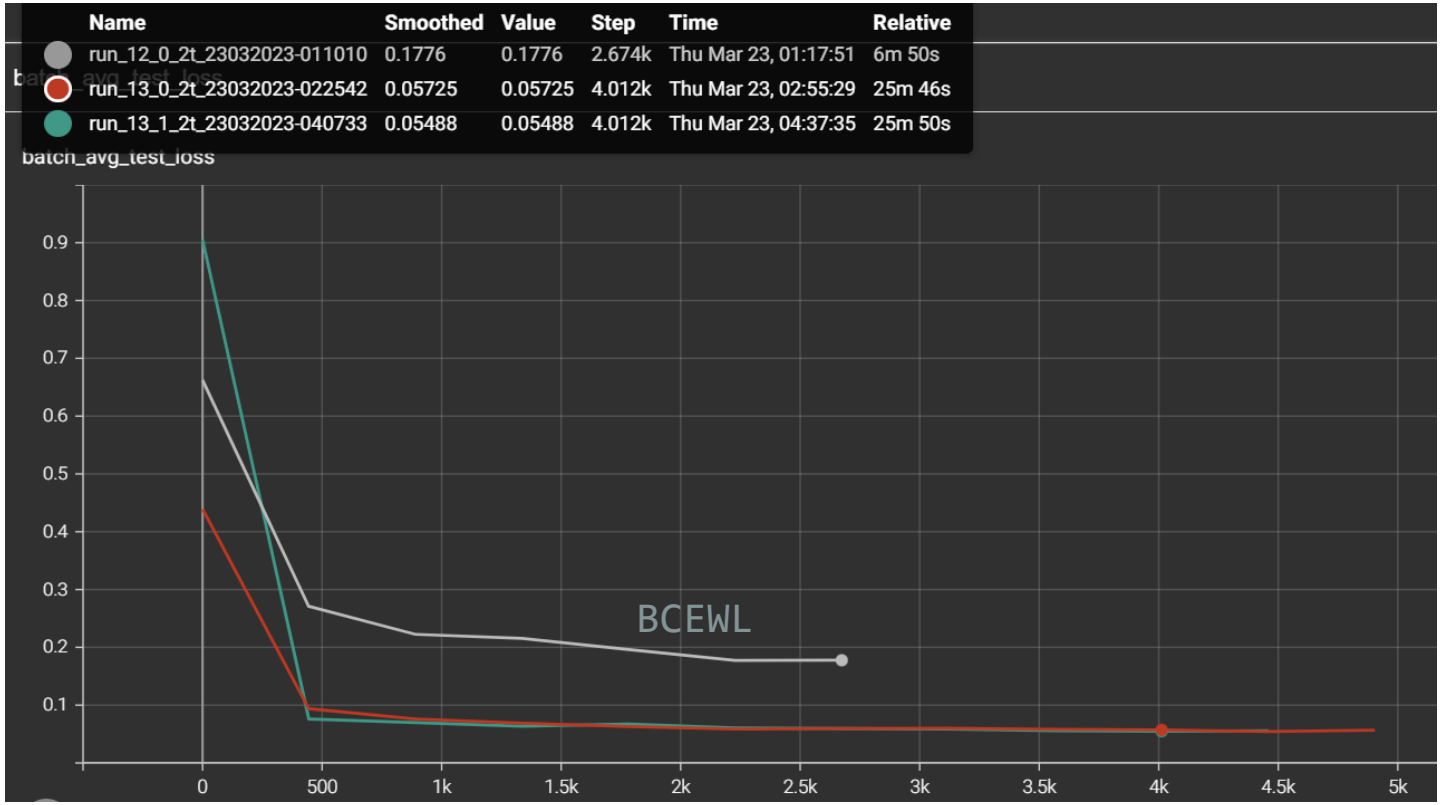
Dice Ratio vs. Jaccard Index

```
unet_15_0_2t:
    - random_seed 2
    - opt Adam
    - arc_change_net
    - use_double_inst_norm
    - loss custom_loss
    - val_custom_loss 0.3
    - early_stopping 3

custom_loss = 0.7 * dice_loss + 0.3 * jac_loss

24) Configuration: ./statisticsmy_dic_train_results_unet_15_0_2t_23032023-060657.json,
      mean-test-loss: 0.0756, mean-train-loss: 0.0687, abs-diff: 0.0069
      mean-test per-class-accuracy: [0.958476   0.96152014 0.95685506 0.9019532  0.8869737  0.9346488 ], total: 0.9334
      mean-test per-class-precision: [0.9691331  0.9782553  0.9723781  0.9366579  0.92978317 0.9370658 ], total: 0.9539
      mean-test per-class-recall: [0.9618122  0.9507156  0.96219844 0.9217398  0.9043916  0.9573889 ], total: 0.9430



unet_15_1_2t:
    - random_seed 2
    - opt Adam
    - arc_change_net
    - use_double_inst_norm
    - loss custom_loss
    - val_custom_loss 0.7
    - early_stopping 3

custom_loss = 0.3 * dice_loss + 0.7 * jac_loss

28) Configuration: ./statisticsmy_dic_train_results_unet_15_1_2t_23032023-064026.json,
      mean-test-loss: 0.1059, mean-train-loss: 0.0929, abs-diff: 0.0130
      mean-test per-class-accuracy: [0.9472253 0.956609  0.9445564 0.8949328 0.8771231 0.9132237], total: 0.9223
      mean-test per-class-precision: [0.9650715 0.9766052 0.9590736 0.9293415 0.930433  0.9273441], total: 0.9480
      mean-test per-class-recall: [0.9504307  0.94442934 0.9584627  0.9214402  0.8934727  0.93904084], total: 0.9345
```



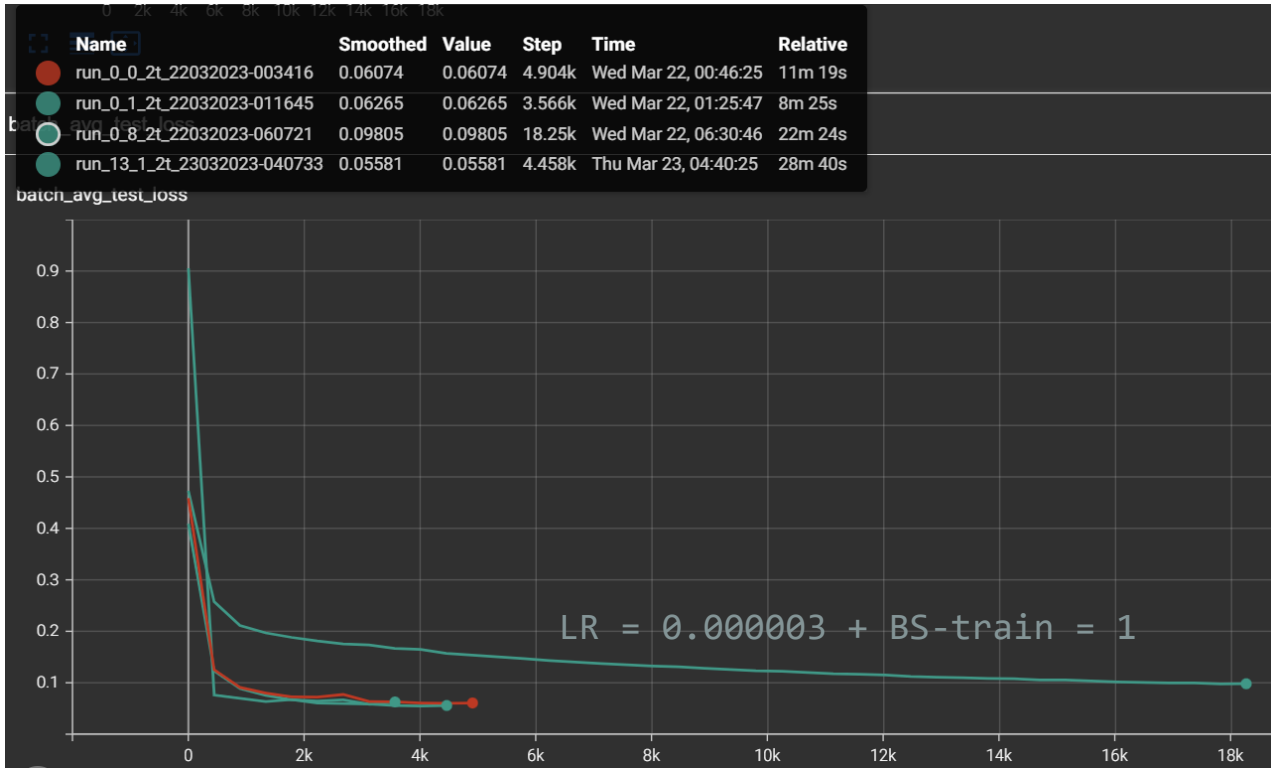
Training choises: inference (6)

- Dice loss is better than BCEWithLogitsLoss for the training-process.



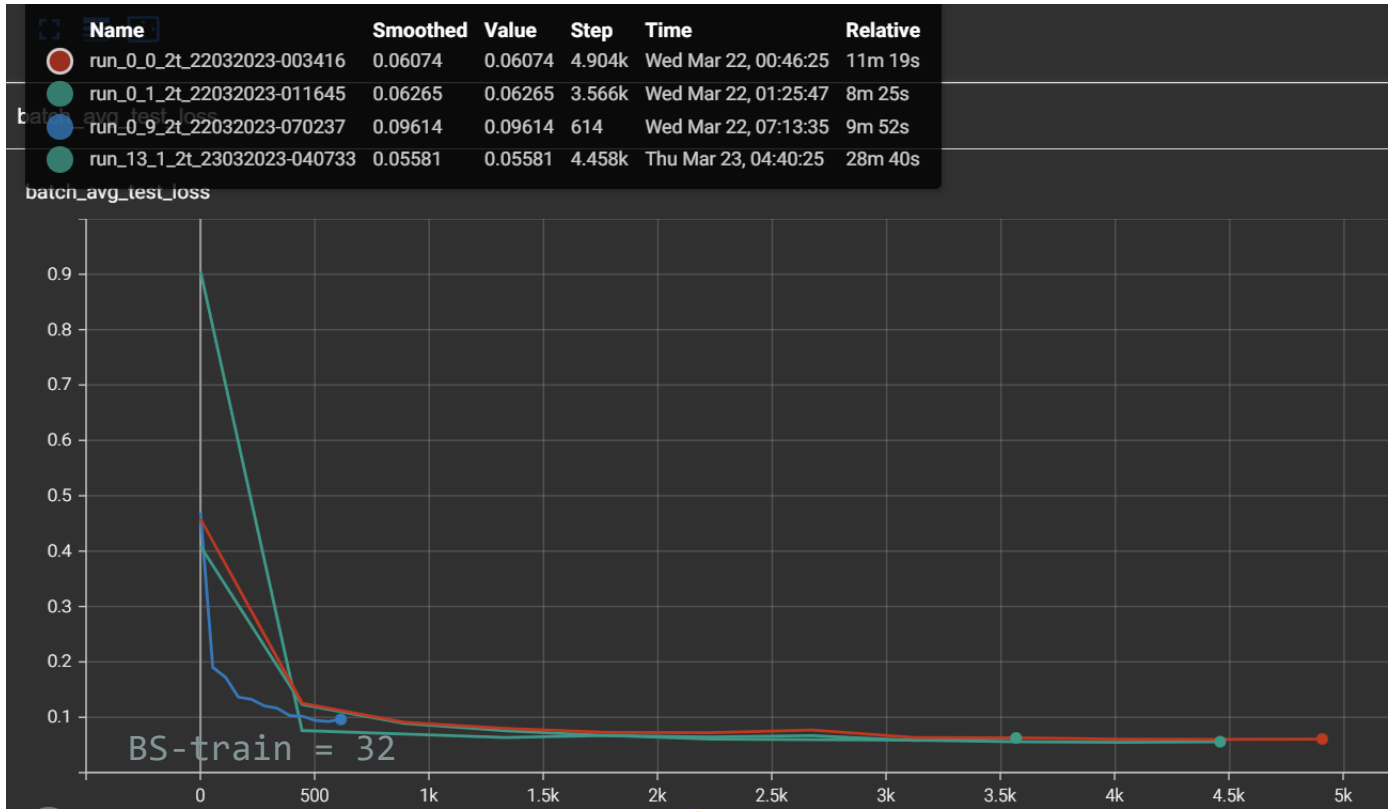
Training choises: inference (7)

- Using the training parameters of the official paper, a deterioration in performance is observed (*LR = 0.000003, BS-train = 1*);



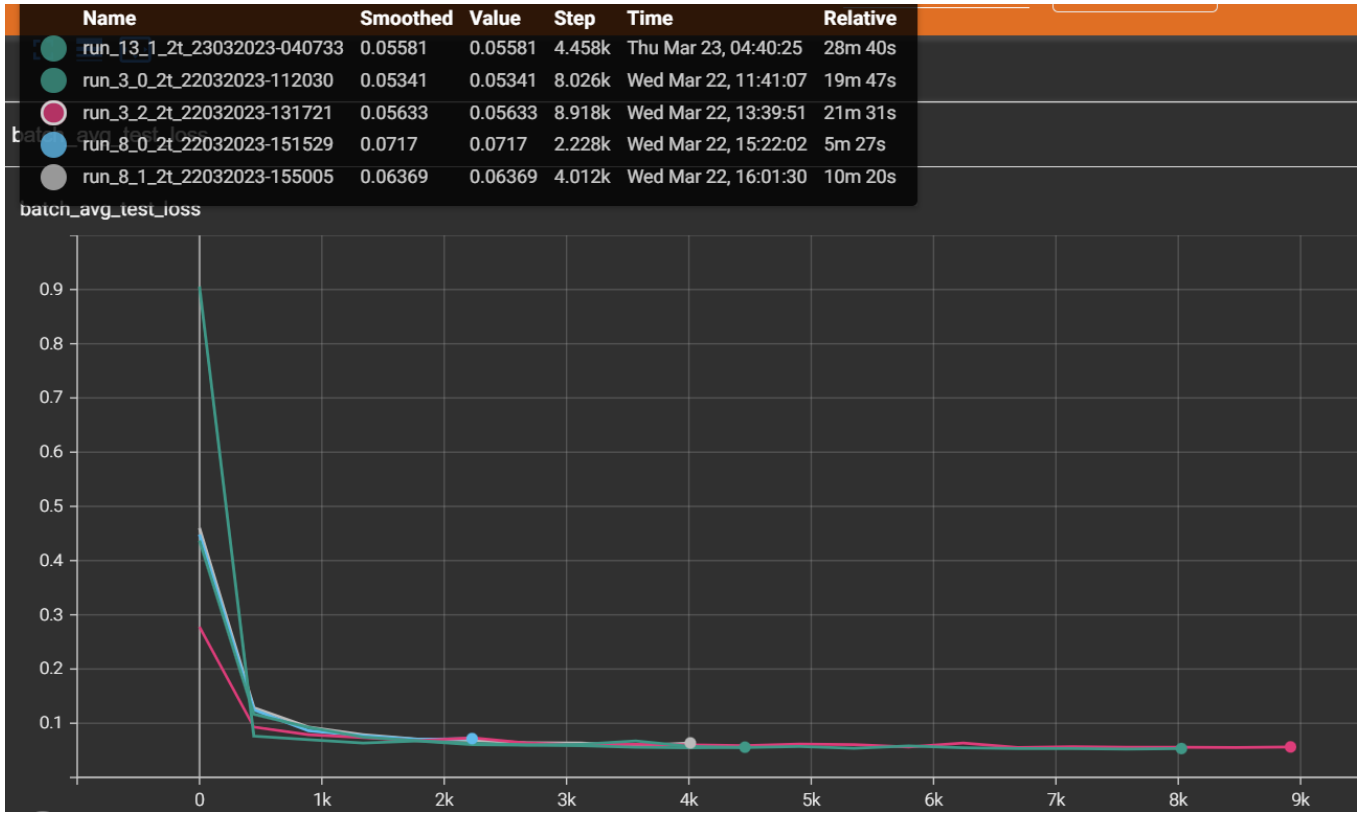
Training choises: inference (8)

- With BS-train = 32 the network does not converge to the optimal minimum.



Training choises: inference (9)

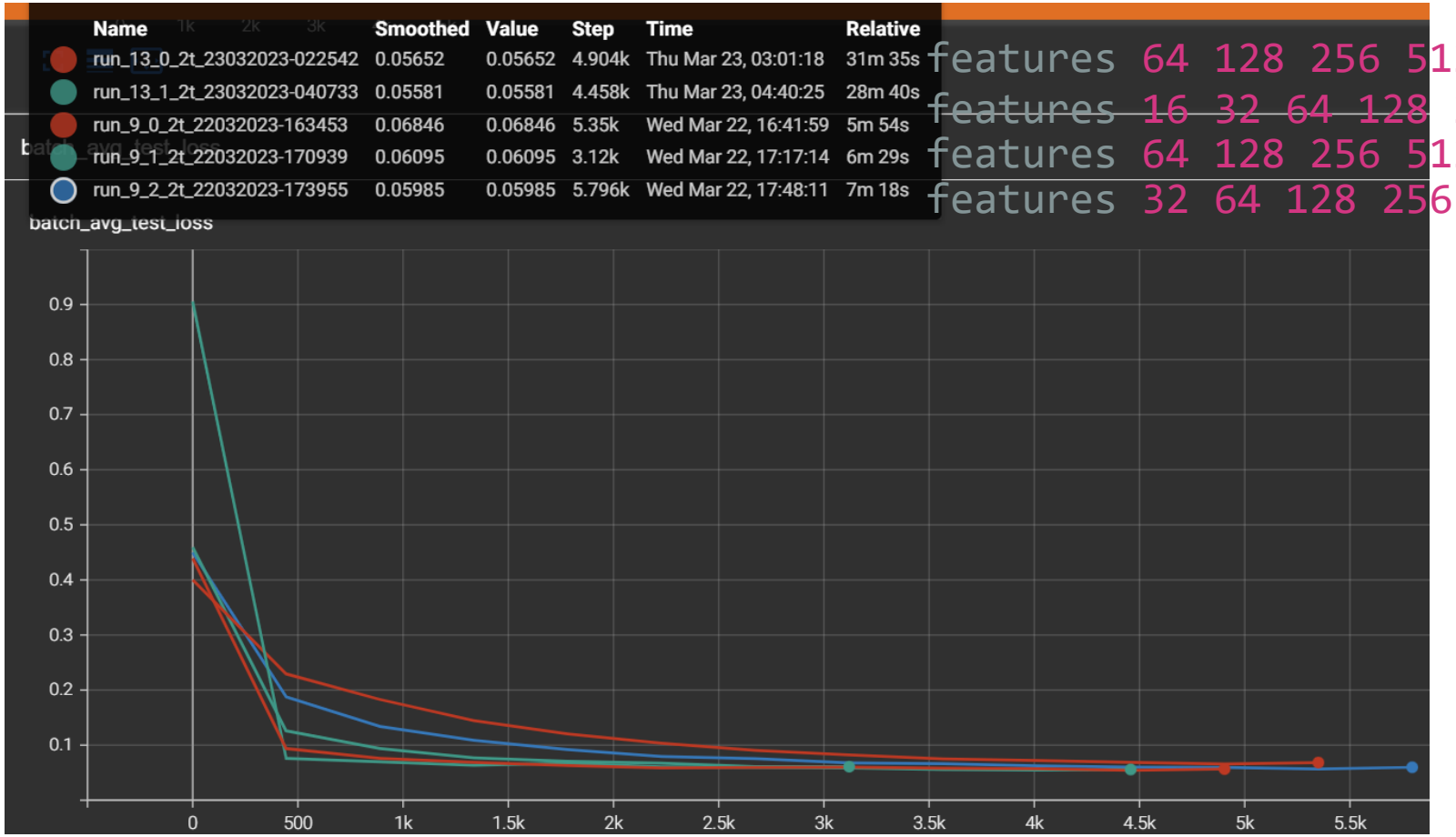
- Early stopping = 3 + BN and Early stopping = 3 + IN



Tensorboard charts & Inference (10)

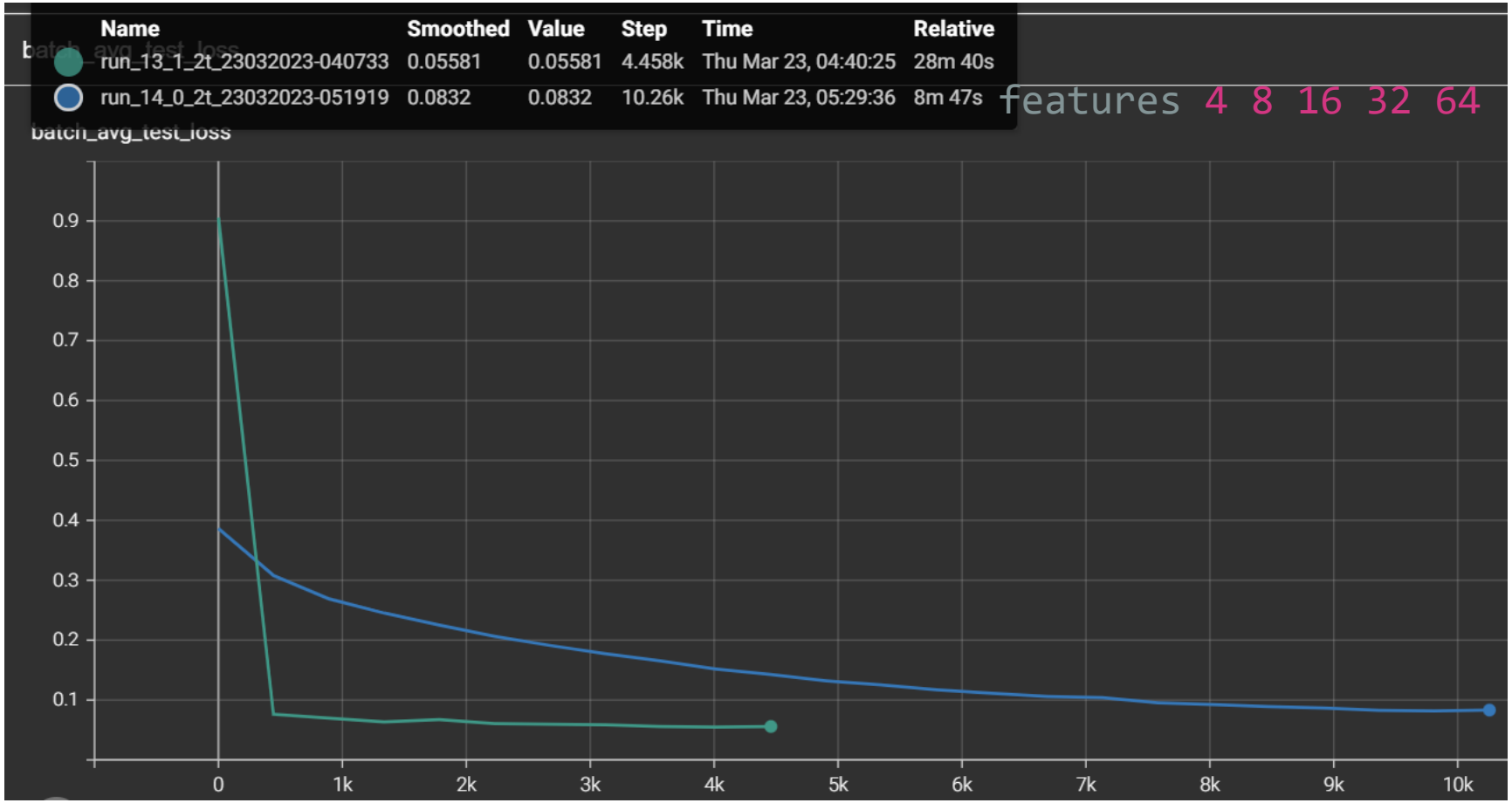
- The following studies involve variations on:
  - Network architecture;
  - Number of filters (neurons) in each convolutional layer;
  - U-Depth.

Tensorboard charts & Inference (11)




```
features 64 128 256 512 1024
features 16 32 64 128 256
features 64 128 256 512
features 32 64 128 256
```

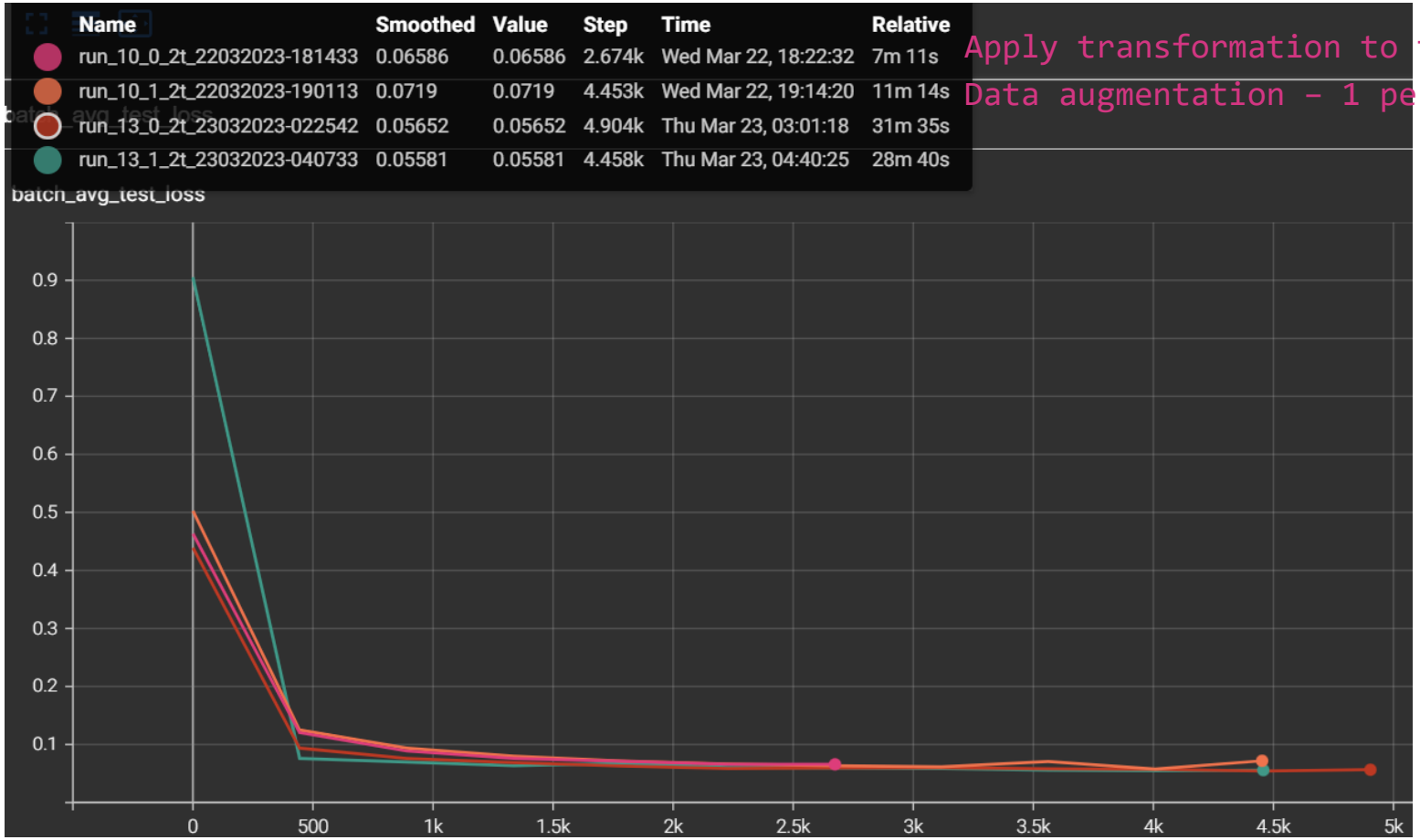
Tensorboard charts & Inference (12)



Tensorboard charts & Inference (13)

- The following studies involve variations on training-data:
  - Data transformation;
  - Data augmentation;
  - Data balancing.

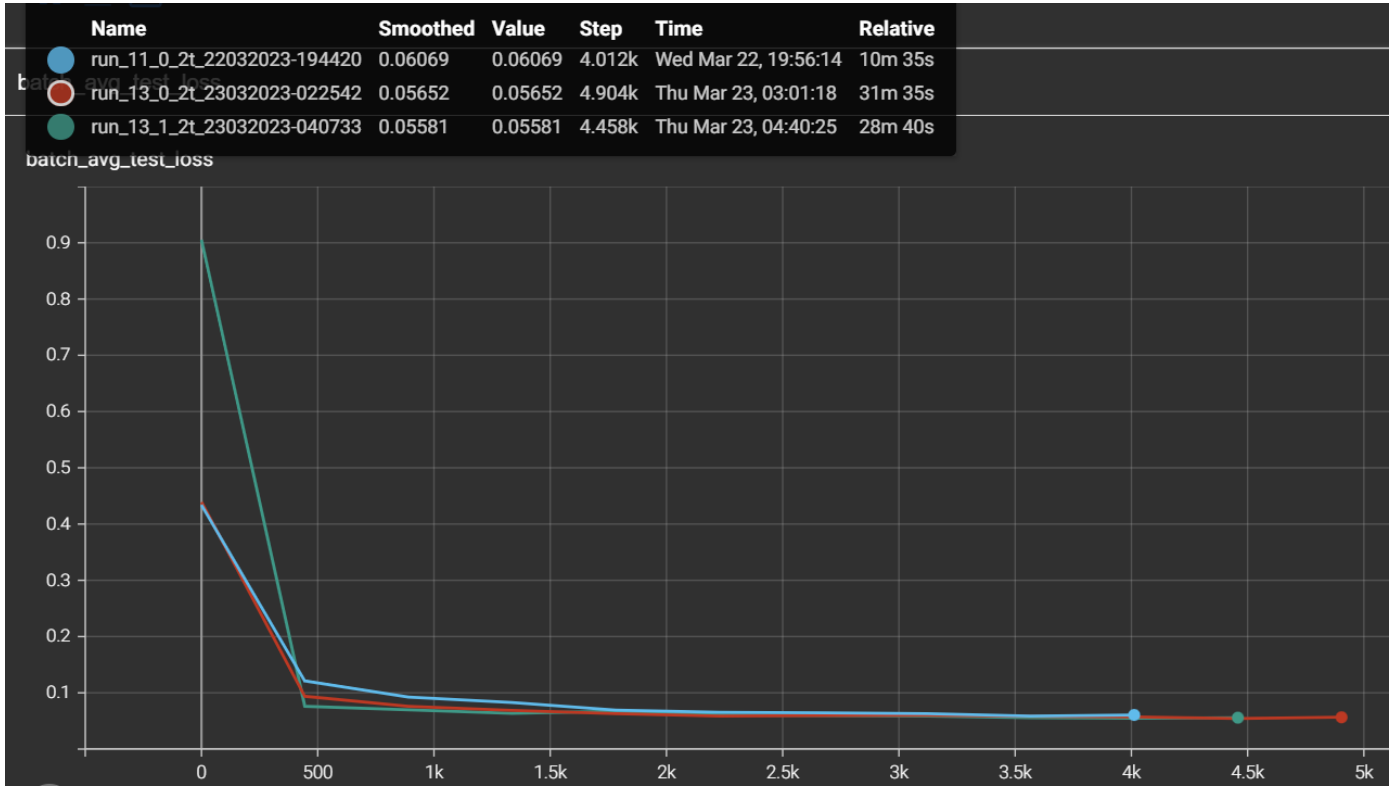
Tensorboard charts & Inference (14)



Apply transformation to training images
Data augmentation – 1 per image

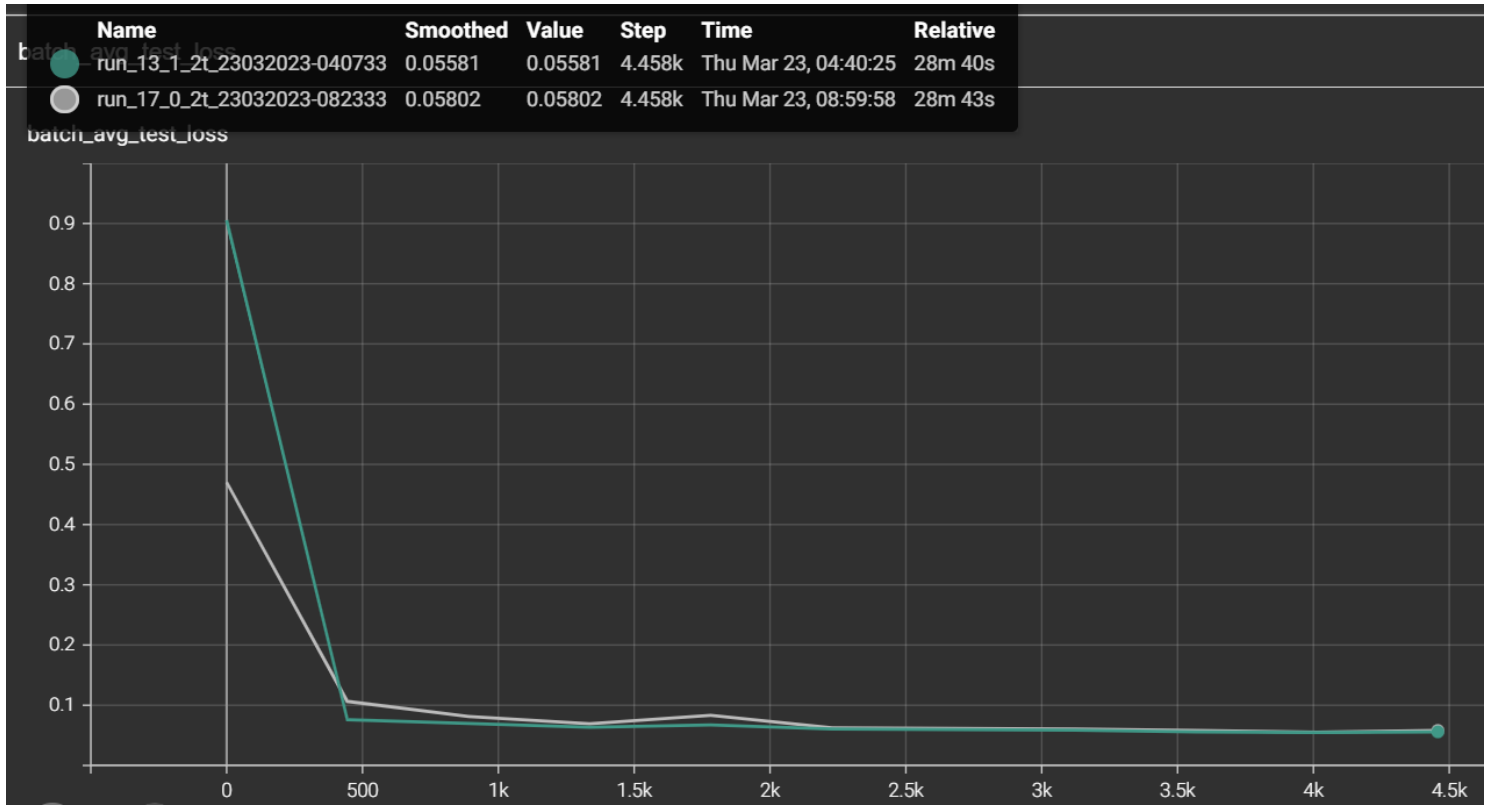
Tensorboard charts & Inference (15)

- Balanced trainloader.



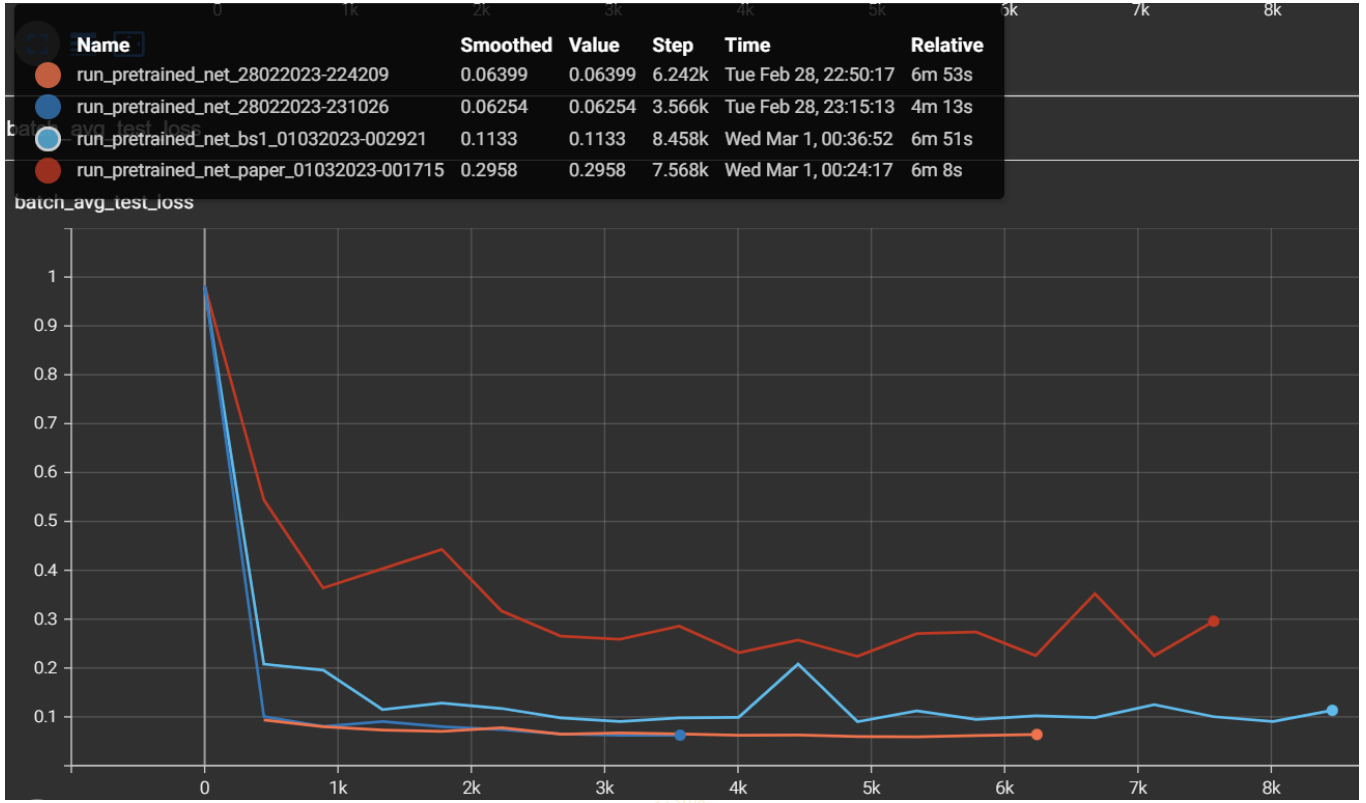
Tensorboard charts & Inference (16)

- Balanced trainloader + Data augmentation (3 per image).



Tensorboard charts & Inference (17)

- Performance on a pretrained-net (U-Net with batch normalization for biomedical image segmentation with pretrained weights for abnormality segmentation in brain MRI).



Final considerations (1)

- Considering the top-5 best, as early-stopping, IN, DBN, BS, Data-augmentation, LR, Weights-Initialization are varied, performance does not change dramatically;
- In contrast, performance changes dramatically (negatively) by exploiting:
  - Do not use any kind of normalization;
  - Using SGD as optimizer;
  - Using JaccardIndexLoss as loss-function;
  - Using BCEWithLogitsLoss as loss-function.

Final considerations (2): results

- According to the best practices of deep learning, the following ablation studies have been carried out on the model (unet_final_2t) defined by the following parameters/hyperparameters:
  - **Adam optimizer** (based on the previous studies it improves the performances);
  - **Features** [64 128 256 512 1024] (based on the previous studies it improves the performances);
  - **Instance Normalization** (BS-test 1 + based on the previous studies it improves the performances);
  - **LR 0.001** (optimal value of the the original paper);
  - **Weights initialization** (based on the previous studies it improves the performances);
  - **BS-train** 4 (based on the previous studies it improves the performances);
  - **Early Stopping** 6 (based on the previous studies it improves the performances);
  - **Balanced trainloader** (based on the previous studies it improves the performances);
  - **Data Augmentation** 3 (based on the previous studies it improves the performances).

Collected performance results: top-5 best (1)

```
Getting the best network configuration...

1) Configuration: ./statistics\my_dic_train_results_unet_18_0_2t_24032023-204346.json,
mean-test-loss: 0.0542, mean-train-loss: 0.0480, abs-diff: 0.0063
mean-test per-class-dice: [0.96469015 0.9634352  0.9652006  0.93124276 0.92124826 0.94398946], total: 0.9483
mean-test per-class-iou: [0.9354971  0.93406874 0.9374406  0.8764165  0.86188865 0.89832705], total: 0.9073
mean-test per-class-accuracy: [0.96003133 0.9630439  0.9567857  0.9093959  0.90325385 0.93306404], total: 0.9376
mean-test per-class-precision: [0.9720483 0.9810069 0.9764826 0.9519362 0.9504126 0.9407182], total: 0.9621
mean-test per-class-recall: [0.96148914 0.951258   0.9580021  0.9185316  0.9057195  0.9517568 ], total: 0.9411


2) Configuration: ./statistics\my_dic_train_results_unet_3_0_2t_22032023-114243.json,
mean-test-loss: 0.0556, mean-train-loss: 0.0411, abs-diff: 0.0145
mean-test per-class-dice: [0.9644952  0.9653902  0.96564966 0.92973334 0.91708213 0.9341321 ], total: 0.9461
mean-test per-class-iou: [0.9357643  0.93835306 0.9384949  0.87373346 0.855395   0.88252765], total: 0.9040
mean-test per-class-accuracy: [0.96082324 0.9660707  0.95802736 0.90883946 0.89879555 0.92278576], total: 0.9359
mean-test per-class-precision: [0.957001   0.9694553  0.9636534  0.92790174 0.9208002  0.90831953], total: 0.9412
mean-test per-class-recall: [0.97673416 0.96680903 0.97201043 0.9395137  0.92771083 0.9687245 ], total: 0.9586


3) Configuration: ./statistics\my_dic_train_results_unet_final_2t_25032023-002043.json,
mean-test-loss: 0.0564, mean-train-loss: 0.0419, abs-diff: 0.0145
mean-test per-class-dice: [0.96690494 0.96388656 0.96533185 0.9387141  0.9121025  0.9565139 ], total: 0.9506
mean-test per-class-iou: [0.9399357  0.9350279  0.93761915 0.88918585 0.8482175  0.9204266 ], total: 0.9117
mean-test per-class-accuracy: [0.9627653  0.963857   0.95710206 0.9201191  0.8910563  0.94875497], total: 0.9406
mean-test per-class-precision: [0.97232825 0.9781351  0.97570413 0.95163554 0.9414864  0.96400183], total: 0.9639
mean-test per-class-recall: [0.96579266 0.9550295  0.9592045  0.9325528  0.9001828  0.9531916 ], total: 0.9443


4) Configuration: ./statistics\my_dic_train_results_unet_13_0_2t_23032023-030252.json,
mean-test-loss: 0.0564, mean-train-loss: 0.0479, abs-diff: 0.0085
mean-test per-class-dice: [0.9617548  0.96378535 0.9653656  0.93425375 0.9141292  0.9474788 ], total: 0.9478
mean-test per-class-iou: [0.9318065 0.935623   0.9386325 0.8815282 0.8518459 0.9056414], total: 0.9075
mean-test per-class-accuracy: [0.9578037  0.96387553 0.9576985  0.9134262  0.89498377 0.93811363], total: 0.9377
mean-test per-class-precision: [0.97444254 0.97995    0.9777157  0.9525688  0.9458925  0.9499728 ], total: 0.9634
mean-test per-class-recall: [0.9551084  0.9537988  0.9582808  0.9238072  0.89834833 0.9506102 ], total: 0.9400
6) Configuration: ./statistics\my_dic_train_results_unet_4_0_2t_22032023-145007.json,
mean-test-loss: 0.0578, mean-train-loss: 0.0520, abs-diff: 0.0059
mean-test per-class-dice: [0.96499443 0.96385145 0.9663394  0.9266444  0.91123015 0.9425742 ], total: 0.9459
mean-test per-class-iou: [0.936425   0.9352708  0.93963623 0.8680667  0.845722   0.8959654 ], total: 0.9035
mean-test per-class-accuracy: [0.9607715  0.96417284 0.9584944  0.9040205  0.8913803  0.9324644 ], total: 0.9352
mean-test per-class-precision: [0.9642415  0.97552663 0.9696622  0.9362466  0.9213207  0.9249571 ], total: 0.9487
mean-test per-class-recall: [0.97002417 0.9577624  0.9671044  0.9247038  0.91662884 0.96600443], total: 0.9504


7) Configuration: ./statistics\my_dic_train_results_unet_3_2_2t_22032023-134126.json,
mean-test-loss: 0.0581, mean-train-loss: 0.0476, abs-diff: 0.0105
mean-test per-class-dice: [0.96402144 0.9637061  0.96532035 0.92787933 0.91184807 0.932523  ], total: 0.9442
mean-test per-class-iou: [0.9347263  0.9348456  0.93771344 0.8704366  0.8470842  0.87918645], total: 0.9007
mean-test per-class-accuracy: [0.9593813  0.9637707  0.9572832  0.9055934  0.89004105 0.91982585], total: 0.9326
mean-test per-class-precision: [0.96758676 0.977279   0.9708979  0.9391184  0.93600535 0.91594934], total: 0.9511
mean-test per-class-recall: [0.9649064  0.9557269  0.96401626 0.9251621  0.90446156 0.95699376], total: 0.9452
```

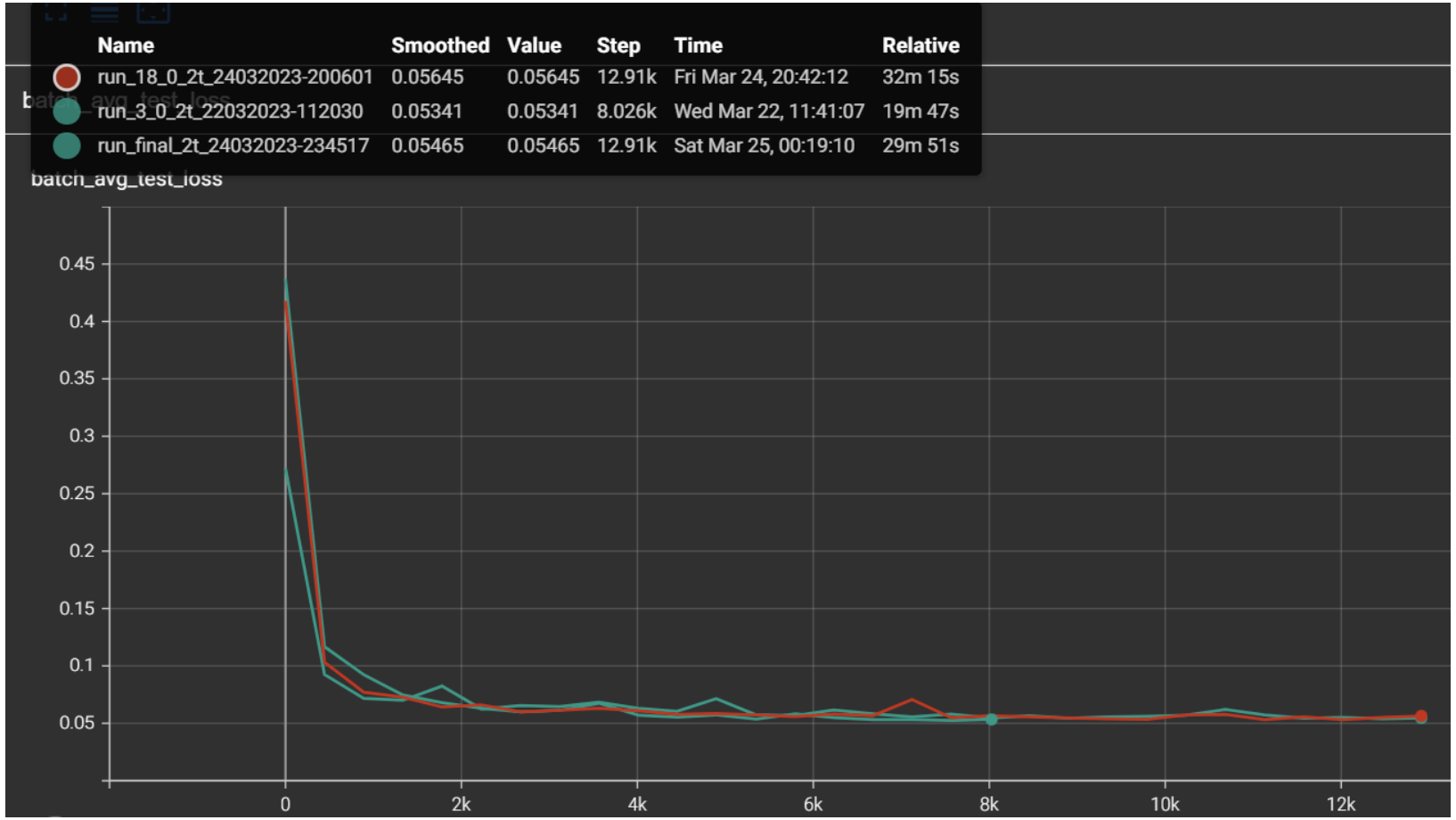
```
unet_18_0_2t
--opt Adam \
--use_double_inst_norm \
--dataset_aug 3 \
--lr 0.001 \
--weights_init \
--early_stopping 6

unet_final_2t
--opt Adam \
--use_inst_norm \
--lr 0.001 \
--weights_init \
--dataset_aug 3 \
--balanced_trainset \
--early_stopping 6
```

Final considerations (4)

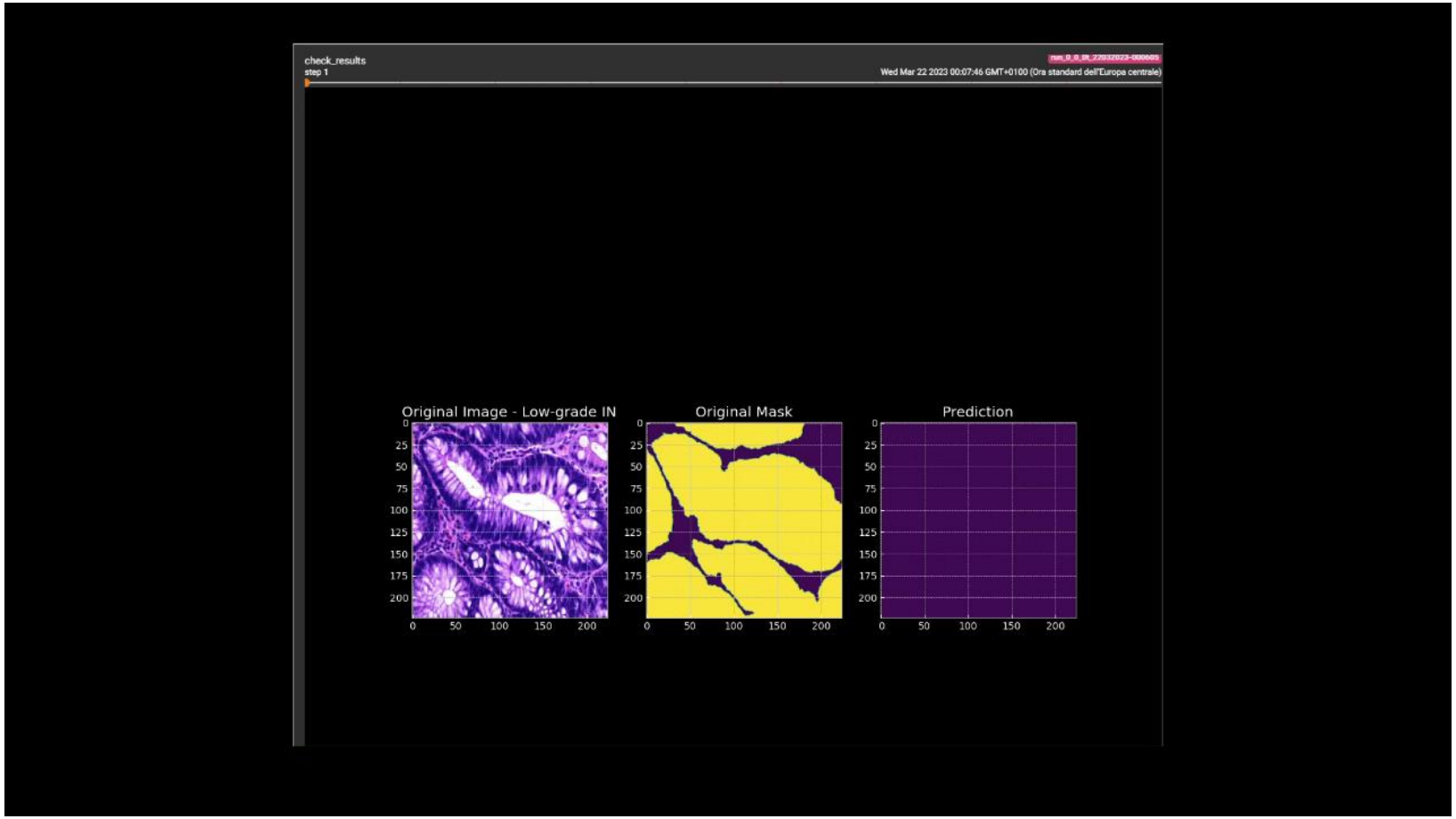
Model	Validation-loss	Macro Dice-Ratio	Macro Precision	Macro Recall
UNet-18_0	**0.0542**	0.9483 (-0.0023)	0.9621 (-0.0018)	0.9411 (-0.0175)
UNet-3_0	0.0556 (+0.0014)	0.9461 (-0.0045)	0.9412 (-0.0227)	**0.9586**
UNet-final	0.0564 (+0.0022)	**0.9506**	**0.9639**	0.9443 (-0.0143)
UNet-0_3 (worst)	0.2237 (+0.1695)	0.7642 (-0.19)	1.000	0.6289 (-0.3297)

Negatives predicted as positives

Final considerations (3)



Predictions (1)



Pruning (1)

- **Sensitivity analysis:**
  - Training the U-Net on a training-dataset, then using pruning techniques to remove one or more non-essential U-Net modules;
  - The performance of the *reduced model* can be tested on a set of test data and compared to that of the original model to evaluate the importance of the U-Net modules;
  - The importance of U-Net modules can be effectively assessed without directly removing them from the model, but rather reducing their overall impact on the model through pruning;

Pruning (2): results

- **Sensitivity analysis:**
  - Ablation studies (sensitivity analysis) on U-Net have shown that **each block** in the network **plays** an **important role** in the semantic segmentation of images, and that removing any block can affect the accuracy and quality of the segmentation;
  - Furthermore, ablation studies have revealed that the **major impact** on performance is related to **two classes** in particular (*Adenocarcinoma*, *High Grade IN*) even for **small pruning values** (25% of filters ablated). However, for more significant values of ablated filters, the performance also decreases in other classes;
  - The **impact** of ablation is **related** to the **architecture**. With a simpler network architecture, performance decreases more significantly across all classes.

Pruning metodologies

- **Prune module:** to sparsify your neural networks;
  - **global_unstructured:** globally prunes tensors corresponding to all parameters by applying the specified pruning_method;
  - **l1_unstructured:** prunes tensor corresponding to parameter in module by removing the specified amount of (currently unpruned) units with the lowest *L1-norm*;
  - **random_structured:** prunes tensor corresponding to parameter in module by removing the specified amount of (currently unpruned) channels along the specified dim selected at random;
  - **random_unstructured:** prunes tensor corresponding to parameter in module by removing the specified amount of (currently unpruned) units selected at random;

- https://pytorch.org/tutorials/intermediate/pruning_tutorial.html
- https://pytorch.org/docs/stable/nn.html#module-torch.nn.utils

Selection metodologies (1)

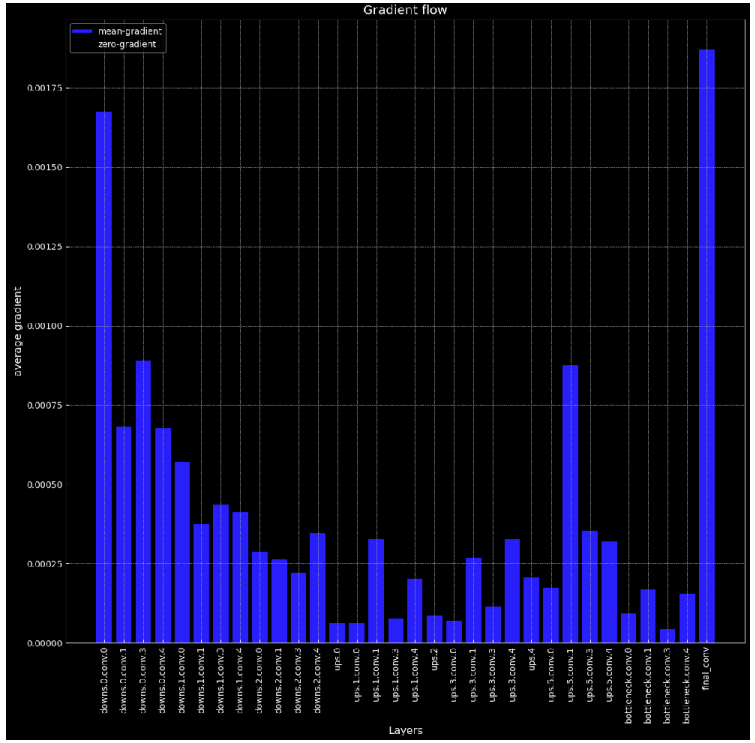
- **Prune module:** to sparsify your neural networks;
  - Overall, I conducted ablation tests on every convolutional layer within the network;
  - However, the tests were also conducted based on certain selection criteria:
    - **Weights distribution analysis before vs. after training process:** to quantify the change in the network's weights, I calculated the difference between the weights of the neural network before and after training by subtracting the weight tensors of the network before and after training, and then computing the *L2-norm*;
    - **Gradients distribution analysis:** high convolutional layer gradient values indicate significant loss function changes, the neural network needs to update the weights of that layer more to improve its performance. Higher gradient values in a layer suggest its potential criticality for the model, but it's not always true. Evaluating the importance of layers should consider various factors, not just gradient values.

Selection metodologies (2)

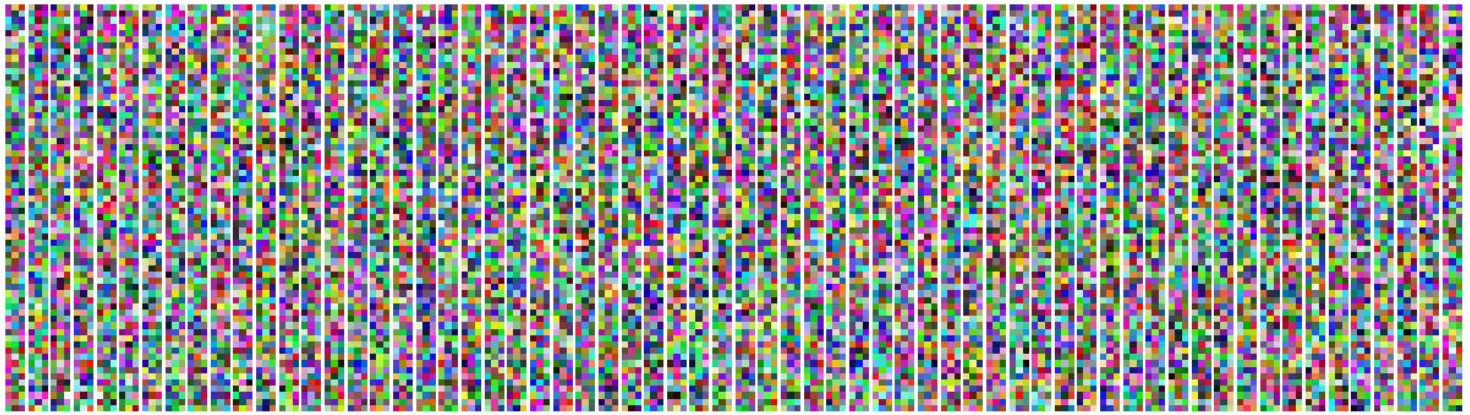
Weights distribution analysis before vs. after training process

```
Distance between "downs.0.conv.0" and "downs.0.conv.0": 1.64
Distance between "downs.0.conv.2" and "downs.0.conv.2": 9.09
Distance between "downs.1.conv.0" and "downs.1.conv.0": 8.18
Distance between "downs.1.conv.2" and "downs.1.conv.2": 12.79
Distance between "downs.2.conv.0" and "downs.2.conv.0": 16.03
Distance between "downs.2.conv.2" and "downs.2.conv.2": 22.87
Distance between "downs.3.conv.0" and "downs.3.conv.0": 33.42
Distance between "downs.3.conv.2" and "downs.3.conv.2": 41.32
Distance between "ups.1.conv.0" and "ups.1.conv.0": 58.45
Distance between "ups.1.conv.2" and "ups.1.conv.2": 44.23
Distance between "ups.3.conv.0" and "ups.3.conv.0": 34.78
Distance between "ups.3.conv.2" and "ups.3.conv.2": 26.32
Distance between "ups.5.conv.0" and "ups.5.conv.0": 15.20
Distance between "ups.5.conv.2" and "ups.5.conv.2": 11.41
Distance between "ups.7.conv.0" and "ups.7.conv.0": 7.20
Distance between "ups.7.conv.2" and "ups.7.conv.2": 5.97
Distance between "bottleneck.conv.0" and "bottleneck.conv.0": 67.51
Distance between "bottleneck.conv.2" and "bottleneck.conv.2": 81.24
Distance between "final_conv" and "final_conv": 1.27
```

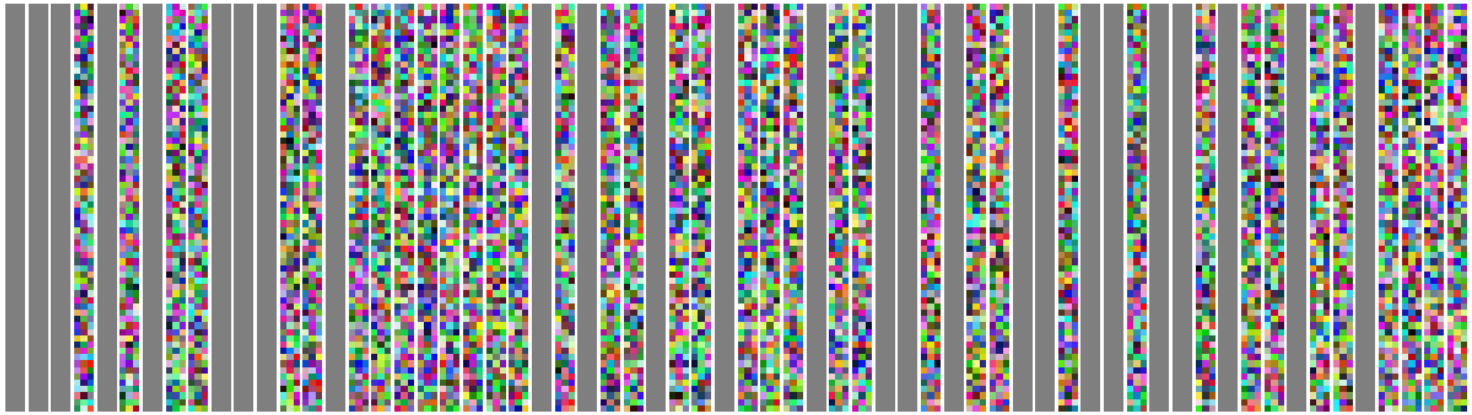
Gradients distribution analysis



Sparse (filters) tensors (1)



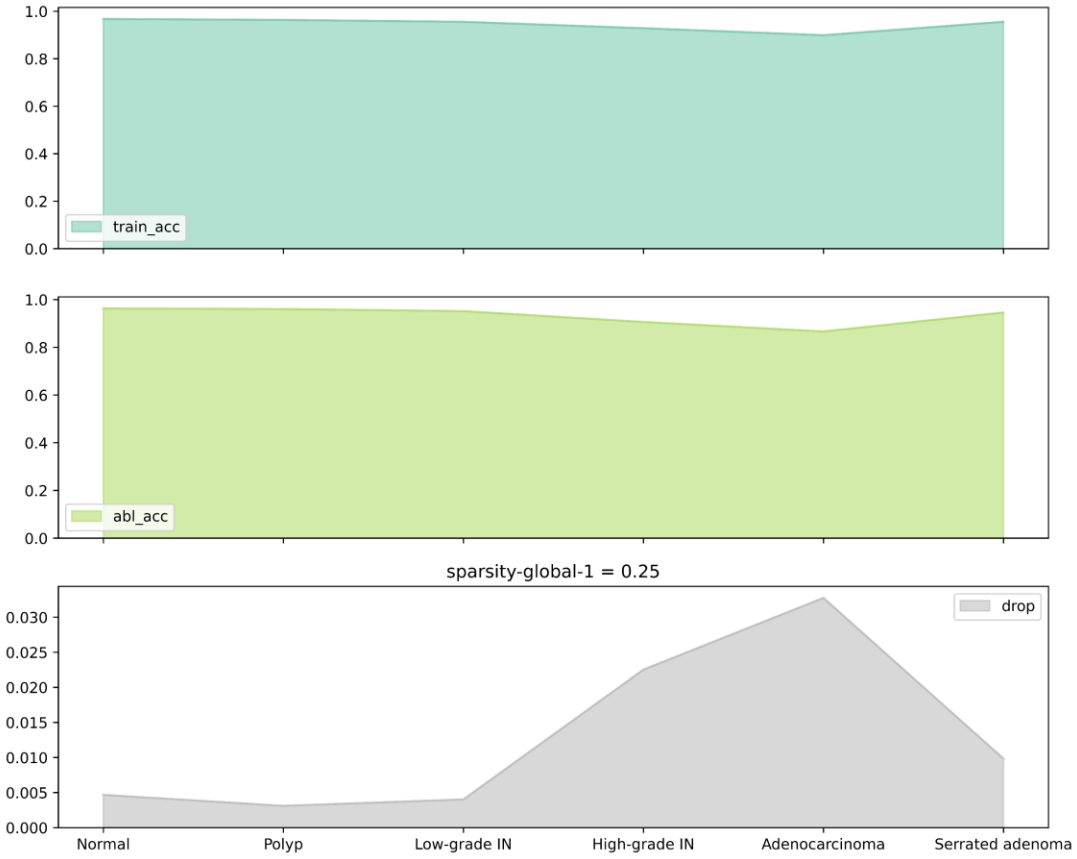
down.0.conv.3

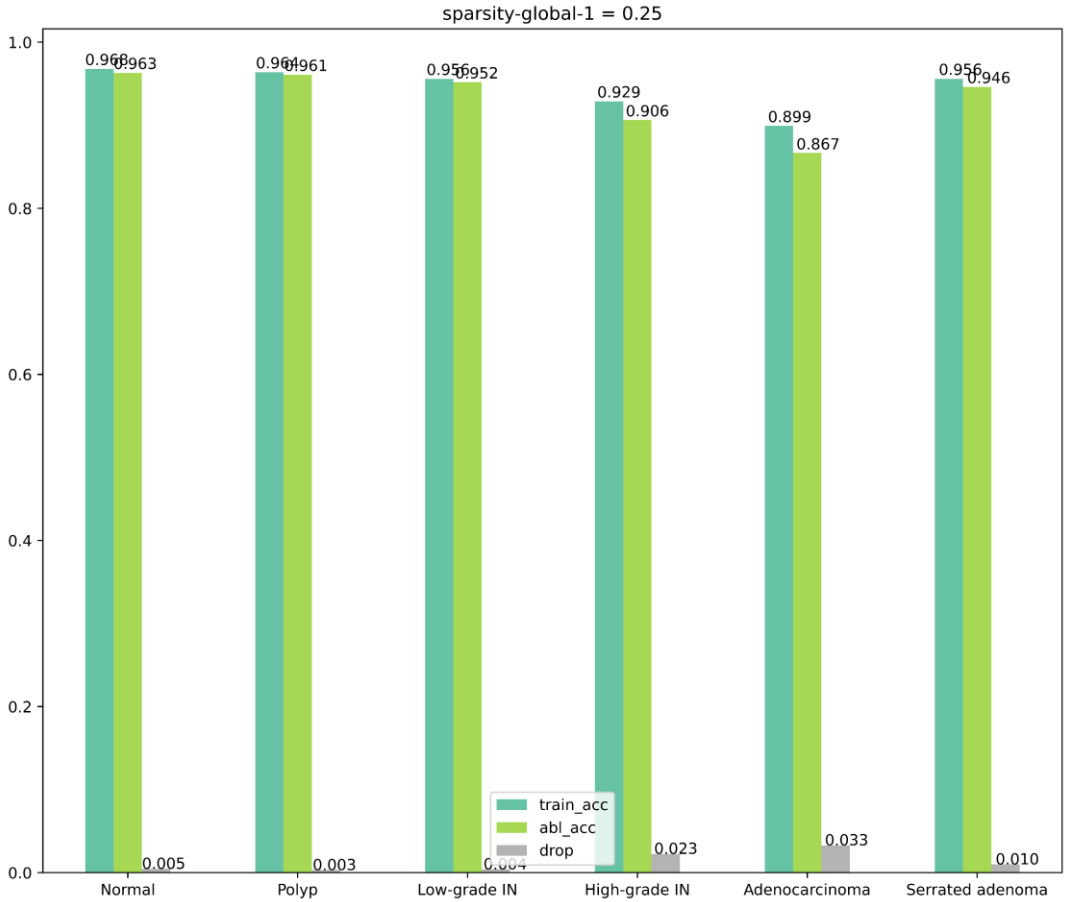


down.0.conv.3 - ablated

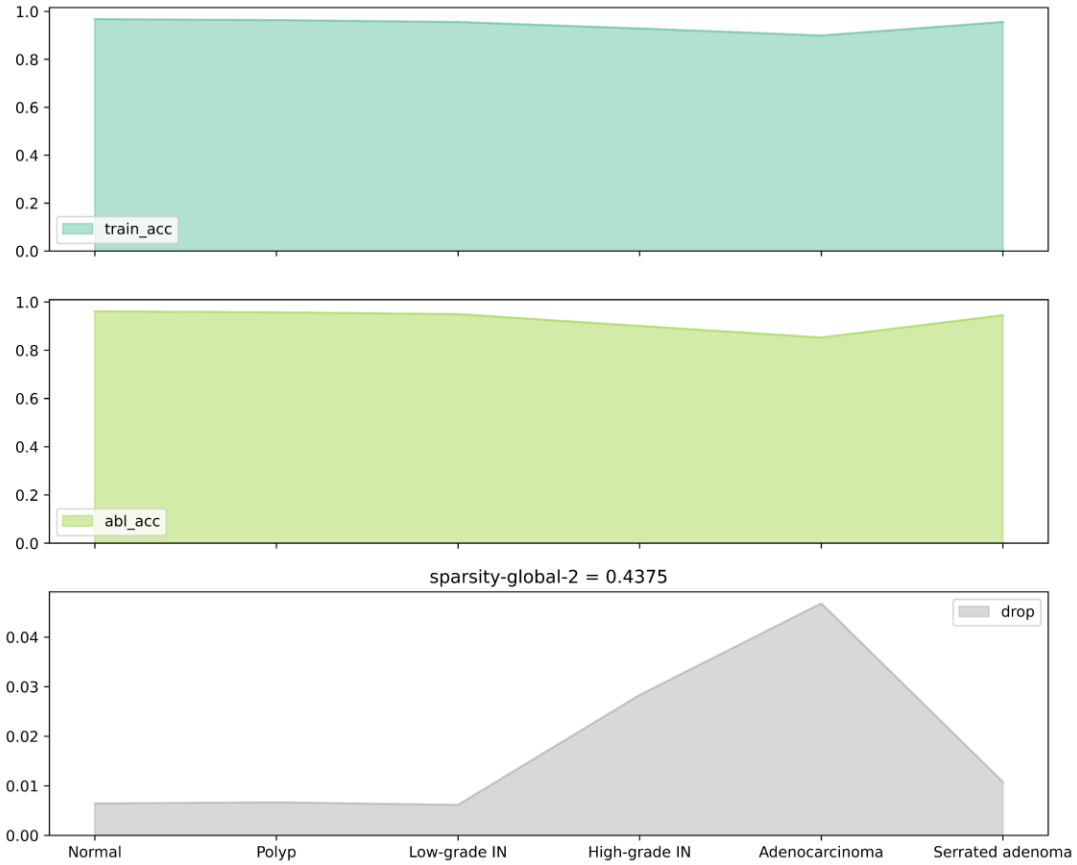
Sparse (filters) tensors (2)

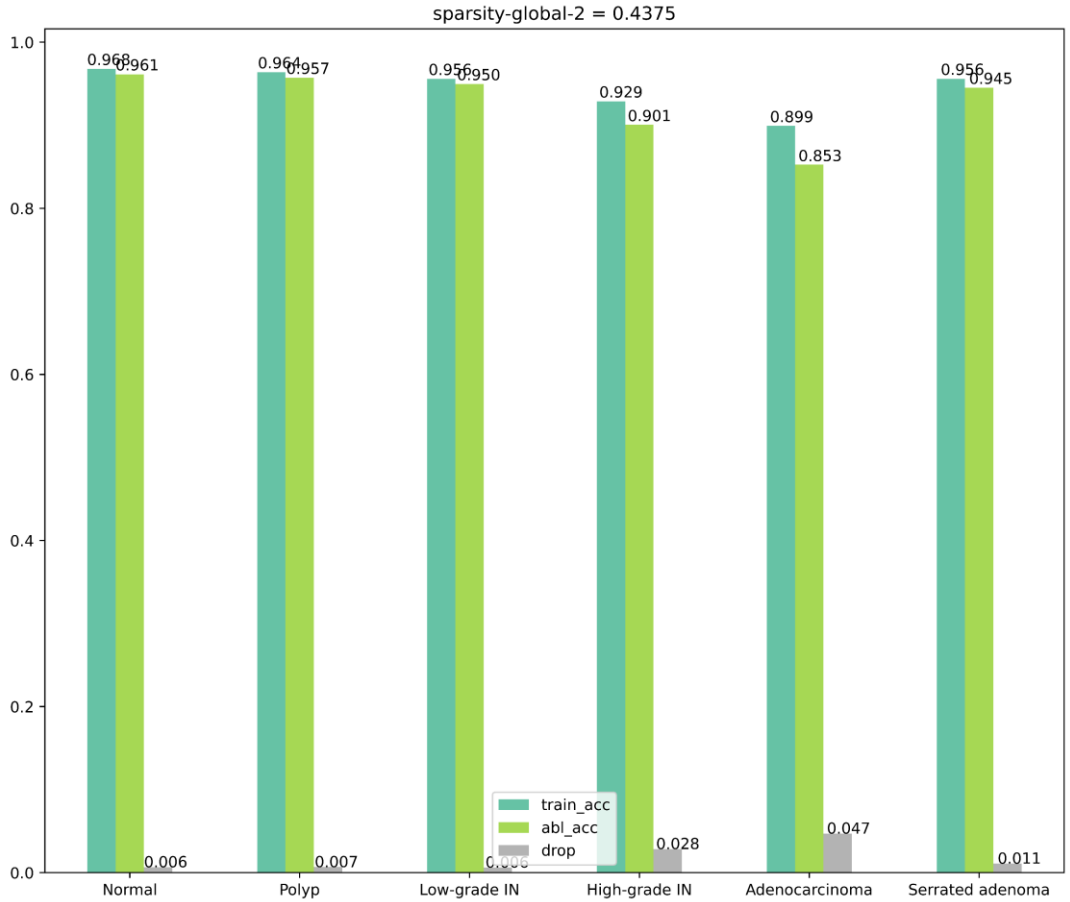
Final considerations (1.1): global l1_unstructured (unet_final_2t)



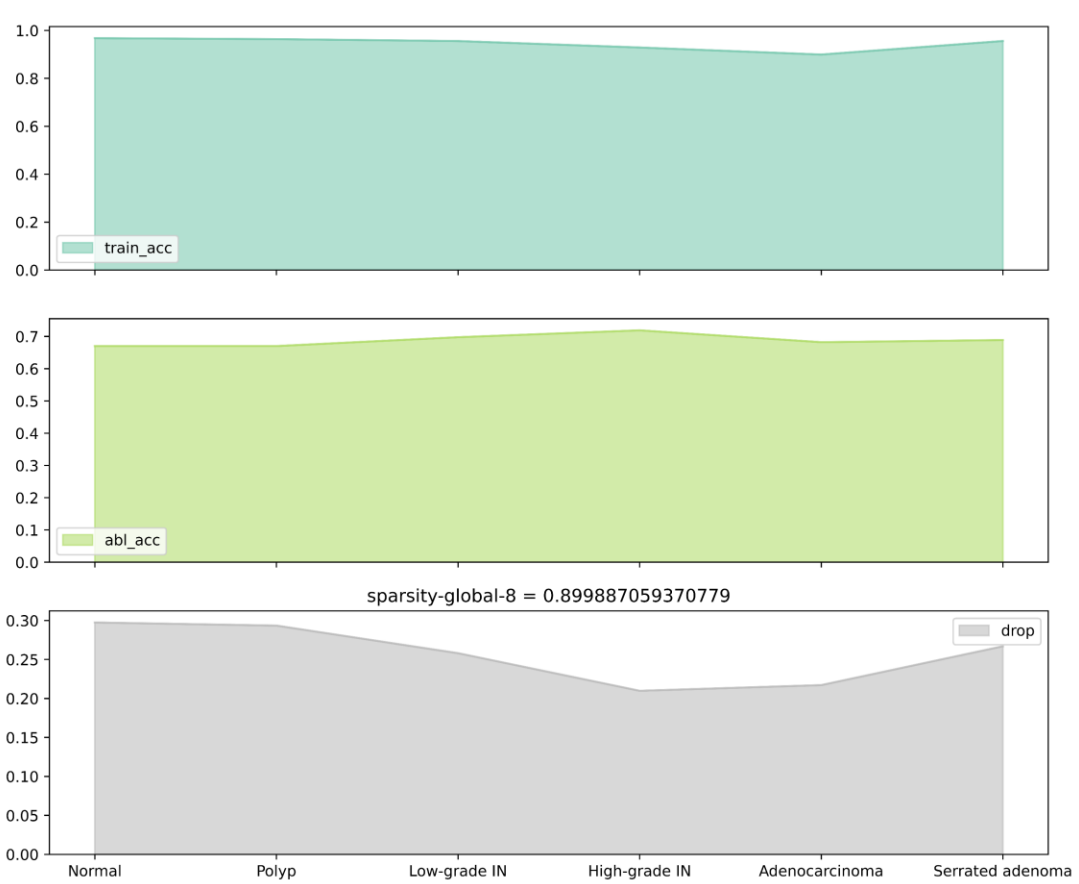


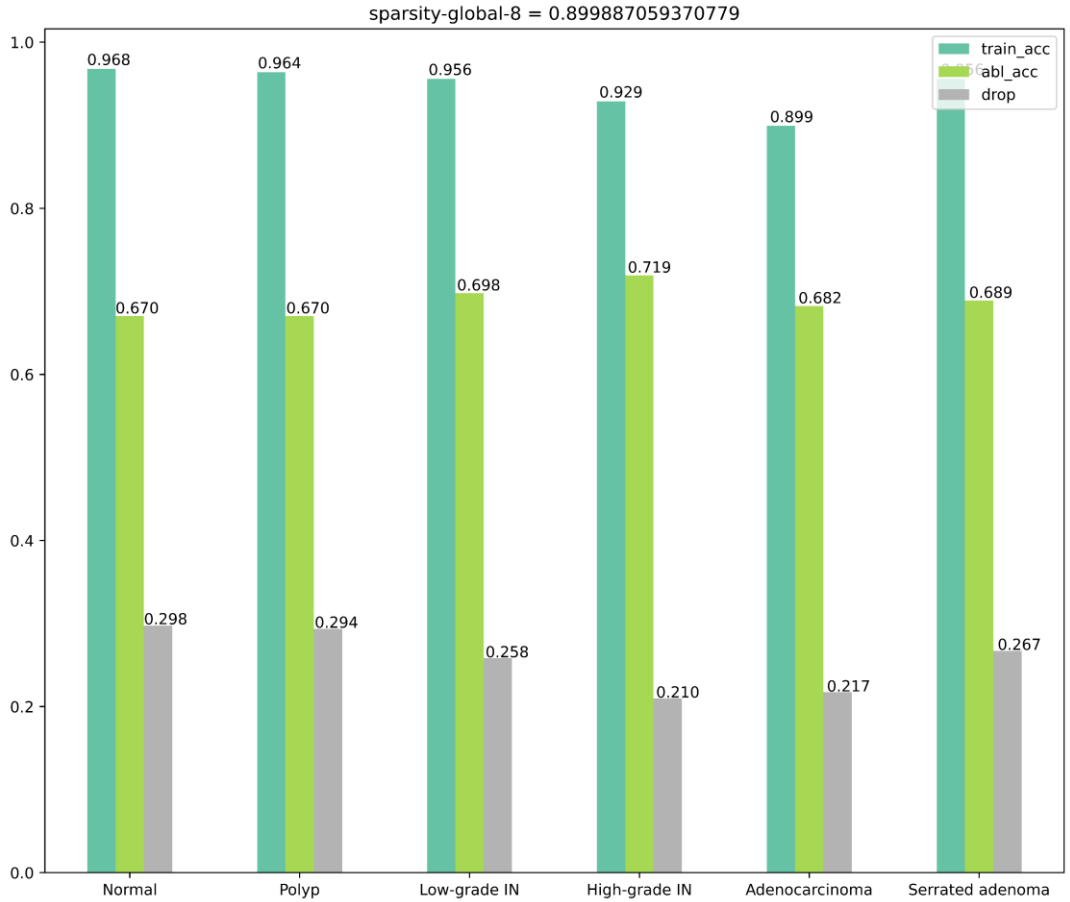
Final considerations (1.2): global l1_unstructured (unet_final_2t)



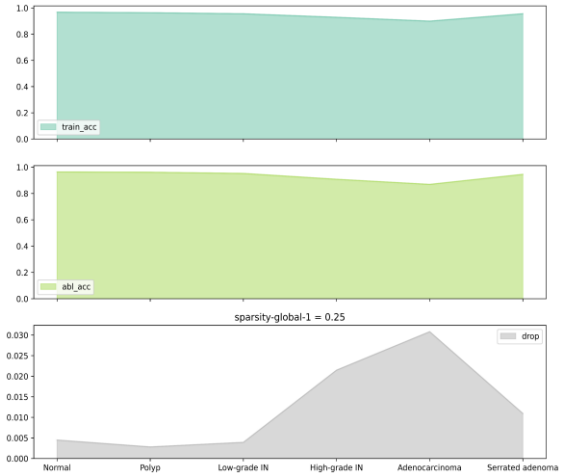


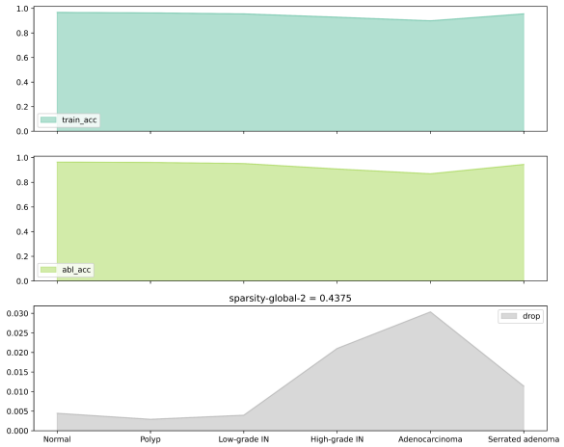
Final considerations (1.3): global l1_unstructured (unet_final_2t)

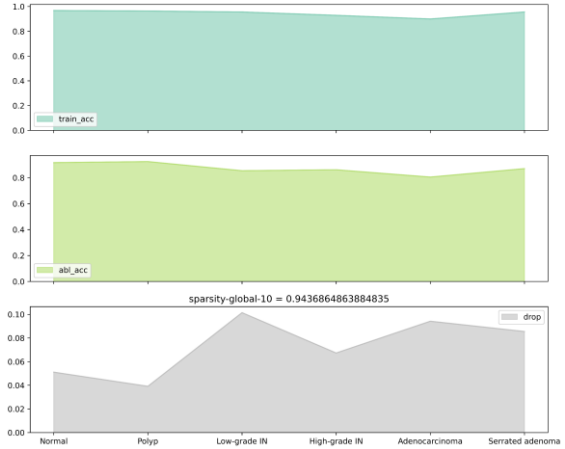


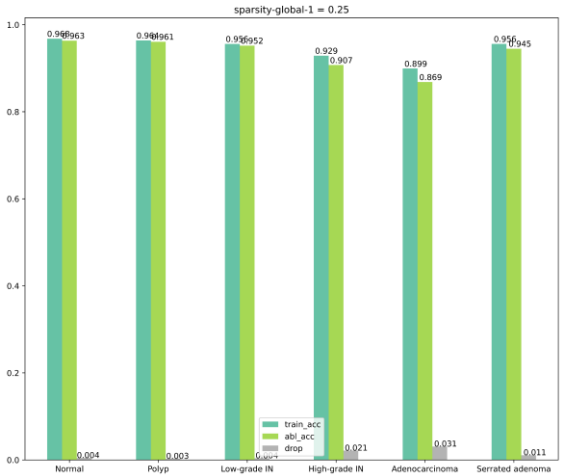


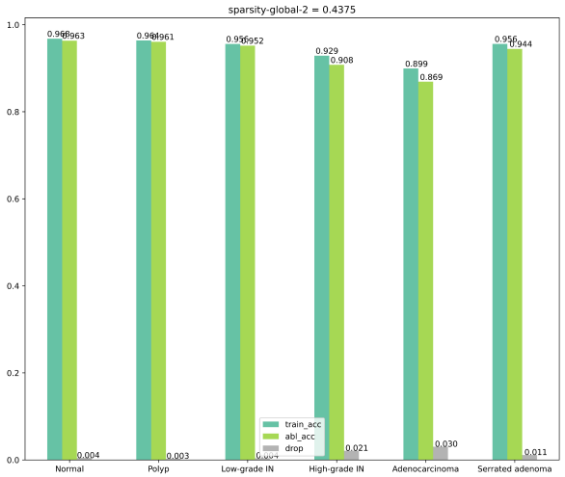
Final considerations (1.4): global grouped (unet_final_2t)

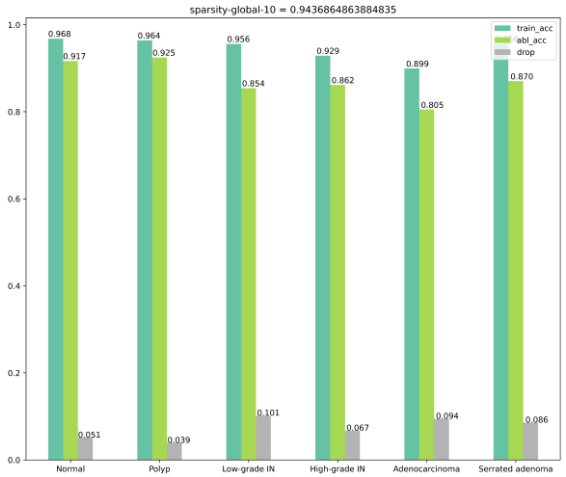




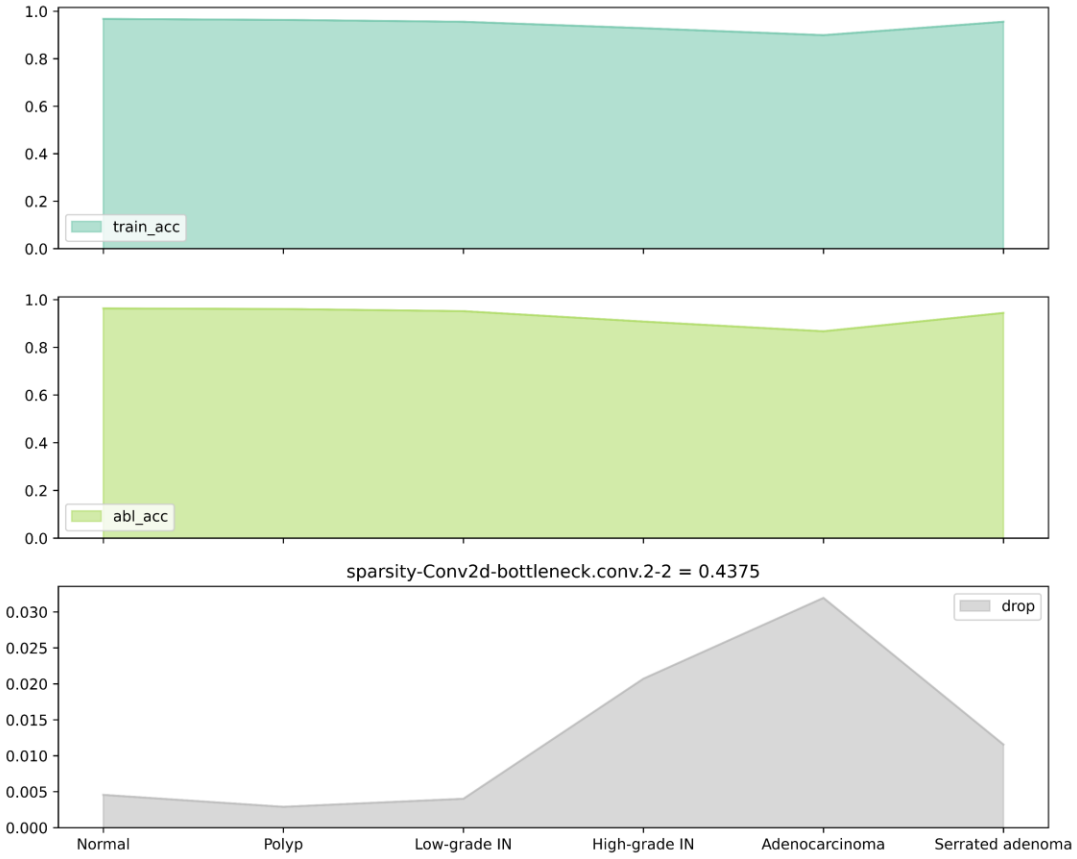


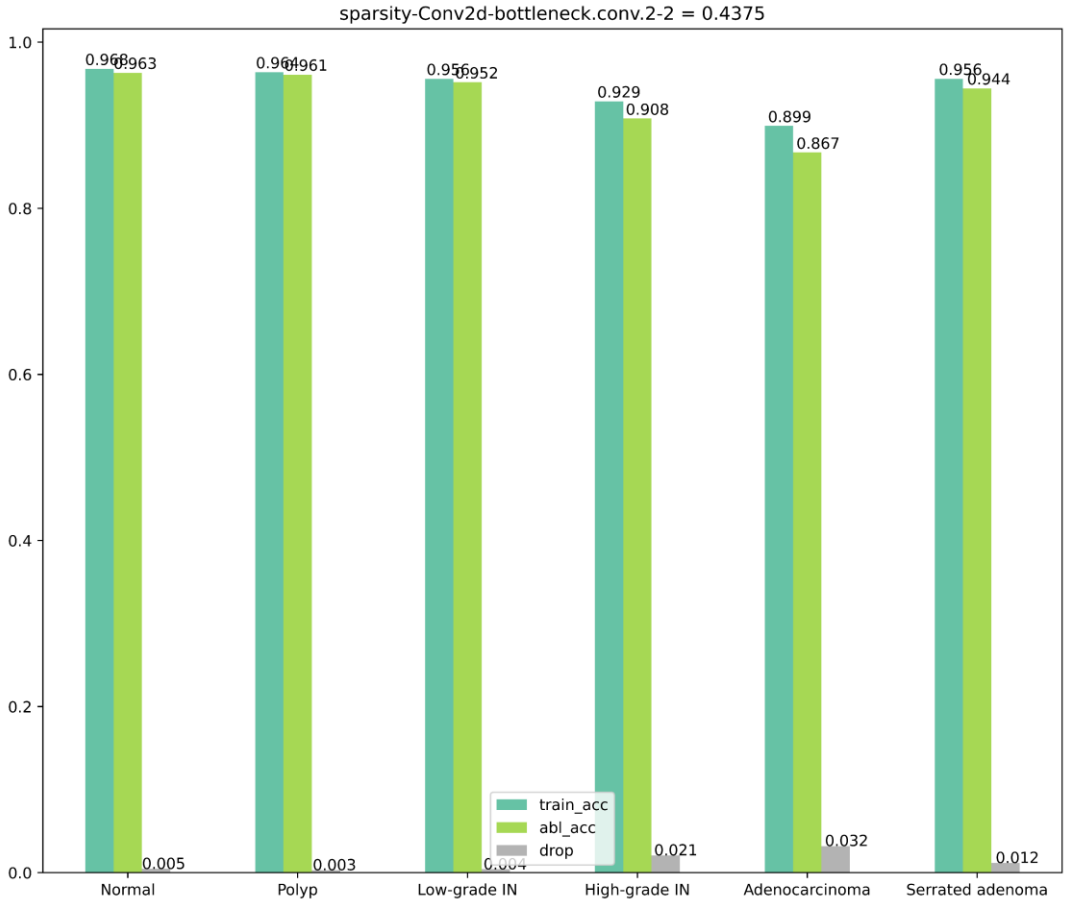




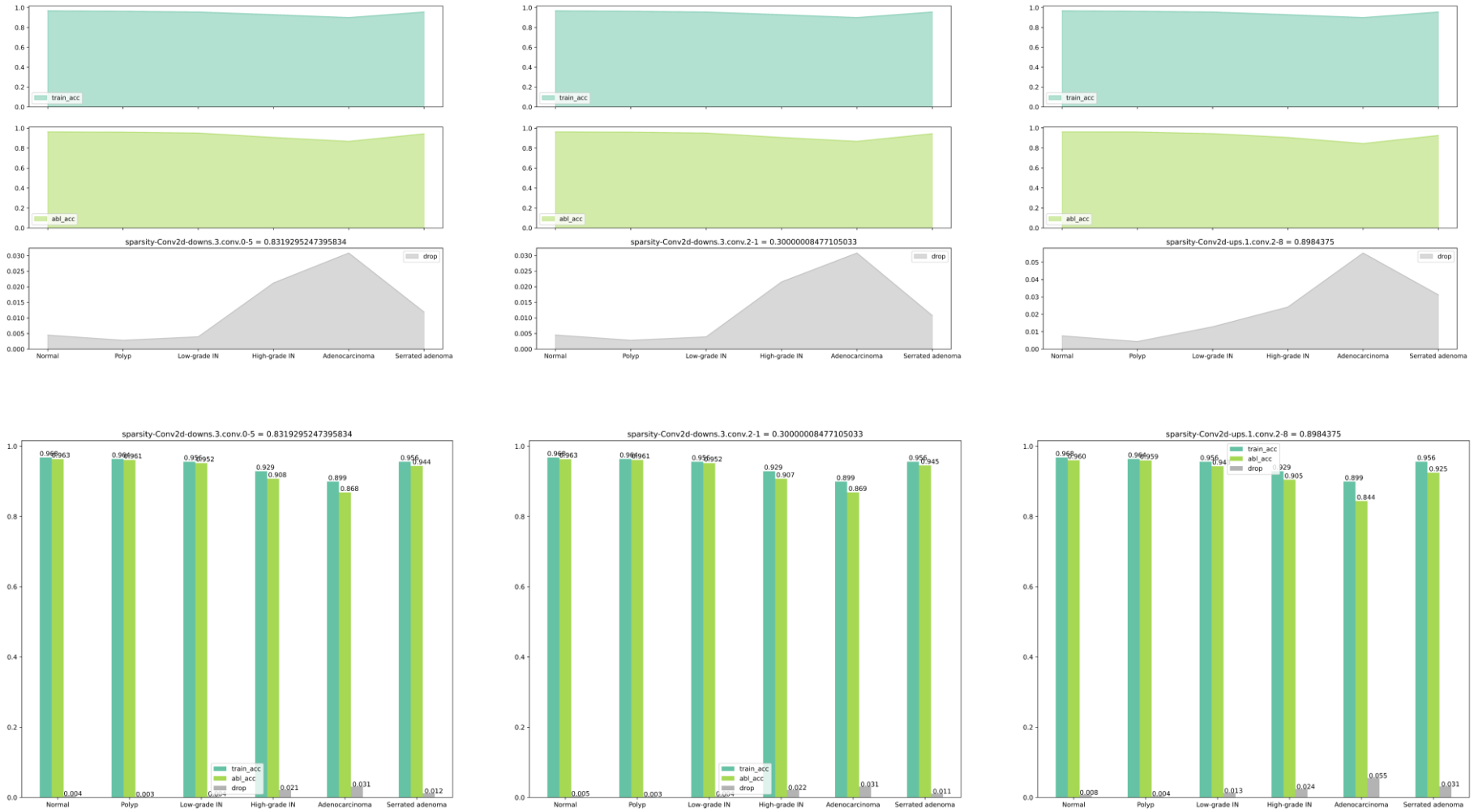


Final considerations (2.1): selective-single-mod (unet_final_2t)





Final considerations (3.1): all_one_by_one random_unstructured (unet_final_2t)



Final considerations (4.1): global (unet_9_2_2t --features(32 64 128) –(use_double_inst_norm))

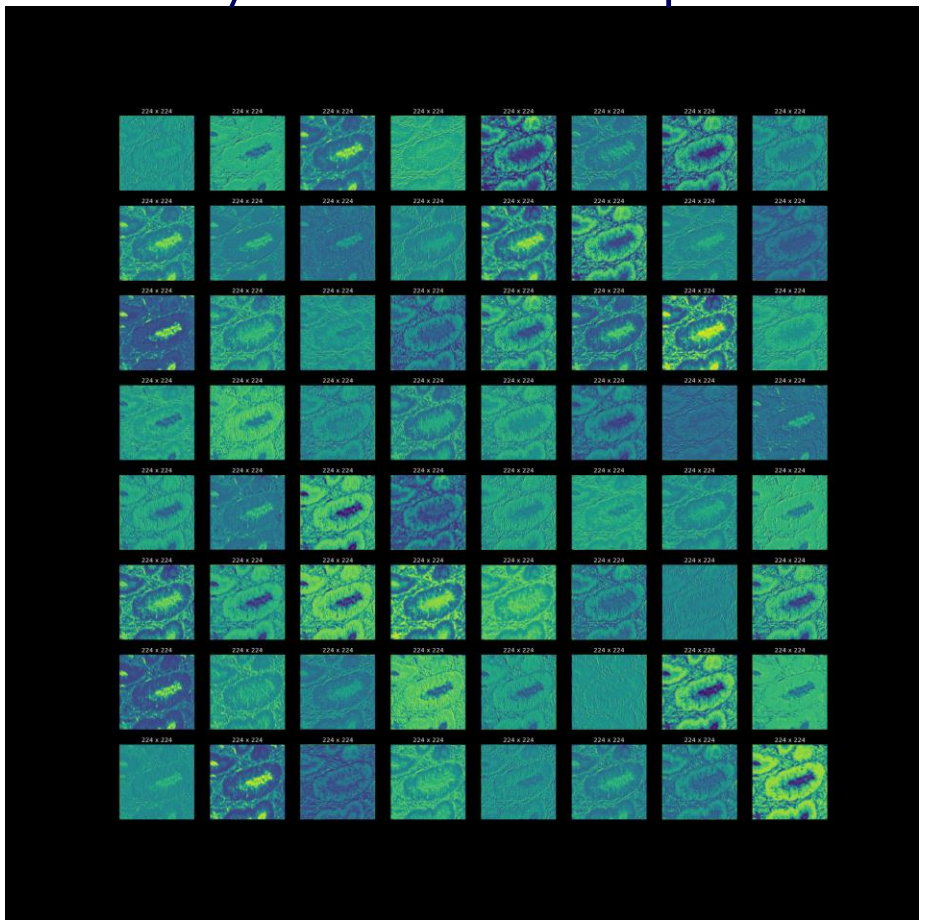
Final considerations (5.1): low-level features extractors vs. high-lelvel features extractors (all_one_by_one)

Predictions (2)

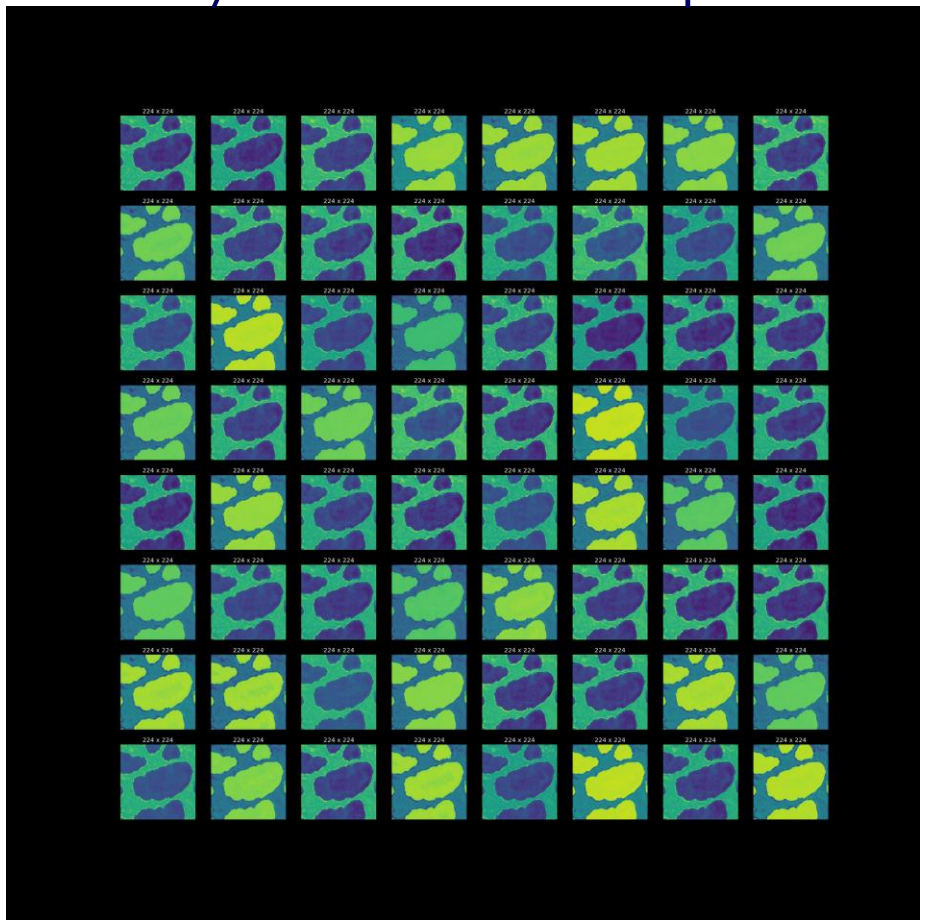
Other info extracted (1)

Other info extracted (2)

layers close to the input



layers close to the output



Insights

- **Padded convolution vs. Valid convolution:** are two types of convolutional operations used in convolutional neural networks (CNNs) for image processing and computer vision. The main difference between the two is the way they handle the boundaries of the input image:
    1. In a Padded Convolution the input image is padded with zeros around the edges before the convolution operation is performed. This is done to ensure that the output feature map has the same spatial dimensions as the input image. The amount of padding required is determined by the size of the convolution kernel and the desired output size. Padded convolution is useful when we want to **preserve the spatial dimensions of the input image**, especially when the edge features of the input image are important for the task at hand;
    2. In a valid convolution, the convolution operation is performed only where the input and kernel fully overlap. This means that the output feature map has smaller spatial dimensions than the input image. Valid convolution is useful when we want to **reduce the spatial dimensions of the input image** or when we are not interested in preserving the edge features of the input image.
- **Dice-Loss vs. IOU-Loss:** in general, the **Dice-Loss** has been shown to be more effective in evaluating semantic segmentation models when the **classes** of objects to be segmented **are imbalanced** or when **maximizing the overlap** between the predicted and real segmentation masks is desired. The **IOU-Loss** (Jaccard-Loss) is more suitable when the main objective is the **precise segmentation** of object classes, rather than the overlap with the real segmentation mask.

Thanks for your attention!