DEEPHEALTH

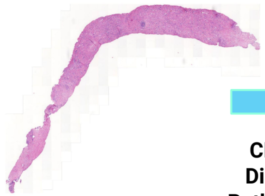
Deep Learning pipeline on histopathology images: detection of prostatic tumor

Francesco Versaci, Giovanni Busonera

CRS4 IDEAS BECOME LIFE

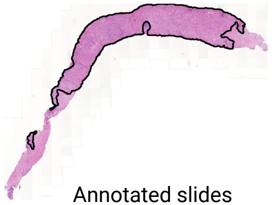
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Pipeline: Predictive model



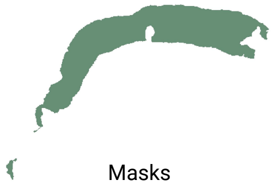
RAW slides
(without annotations)

CRS4 Digital Pathology Platform



Annotated slides

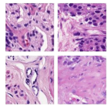
Mask Creation



Masks

Patch Extraction

Data

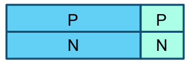


patch_id	name	rep	tissue	lab	x	y
-bdf3b281d589	AN5027	8	0.989319	1	70504	150525
-a69edae879d6	AN0005	7	0.999695	1	63351	97790
2-4f753ef5c72f	AN5025	2	0.971298	1	37295	81918
-4e22aae3e25f	AN0077	3	0.905426	1	64373	105982

Metadata

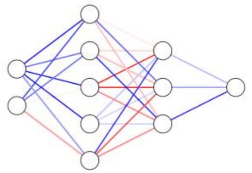
Cassandra DB

Split Creation

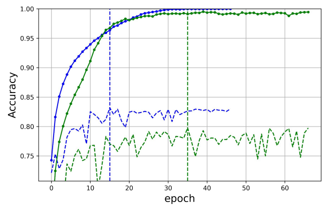


Splits

Model Training

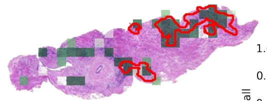


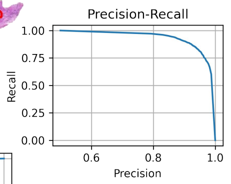
Trained Model

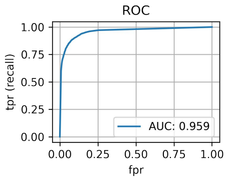


Cross-Validation and Tuning

Test set evaluation

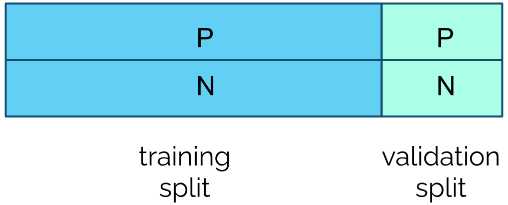






The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Creation of the splits



All **splits** are **balanced**, meaning that the number of tumor patches is roughly the same of the normal ones.

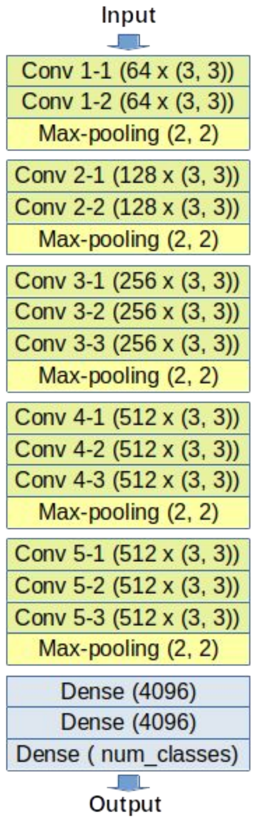
We used the **80%** of the whole dataset for **training data**. The remaining **20%** is used for the **validation set**. The **split_ratios** parameter is set to **[8, 2]**. (**47782** and **11558** patches respectively)

The **test-set** data is organized as a **single split** obtained from different Cassandra tables (**44416** patches)

```
ap = PlainTextAuthProvider(username='xxxx', password='xxxx')
cd = CassandraDataset(ap, ['cassandra-db'])

cd.init_listmanager(table= args.ids_table,
                        metatable= args.metatable,
                        id_col='patch_id',
                        num_classes=args.num_classes,
                        label_col=args.label,
                        grouping_cols=['sample_name']
)

cd.read_rows_from_db()

min_class = np.min(cd._clm._stats.sum(axis=0))
data_size = min_class*cd._clm.num_classes

cd.init_datatable(table=args.table)
cd.split_setup(split_ratios=args.split_ratios, max_patches=data_size, augs=[])

cd.save_splits(args.out_fn)
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

VGG16

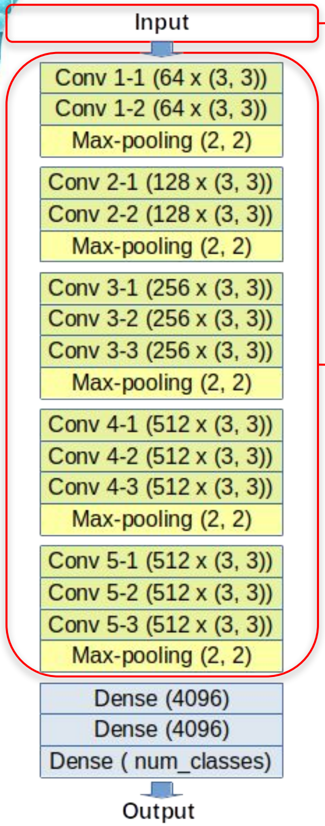


- **VGG16** is a **convolutional neural network** with a top **classifier** made of **dense layers**
- The number of **trainable parameters** is about **166M** (images **256x256** pixel instead of 224x224)
- **Pros:** Classical CNN architecture, **easy to implement** from scratch and although not currently the state of the art, it **was one of the best performing architectures** in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) challenge 2014
- **Cons:** It is **slow to train** compared to more recent ANNs. The **size of VGG16 weights** is about **600MB** which potentially takes a lot of disk space (e.g. if you store the model after each epoch) and hence a **long time to synchronize** in a distributed training setting

- EDDL provides a VGG16 neural network pretrained on the Imagenet dataset
- We implemented a VGG16 by using PyEDDL to have more flexibility (e.g.: train from scratch with **different initialization methods** or **pretrained net**)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

VGG16

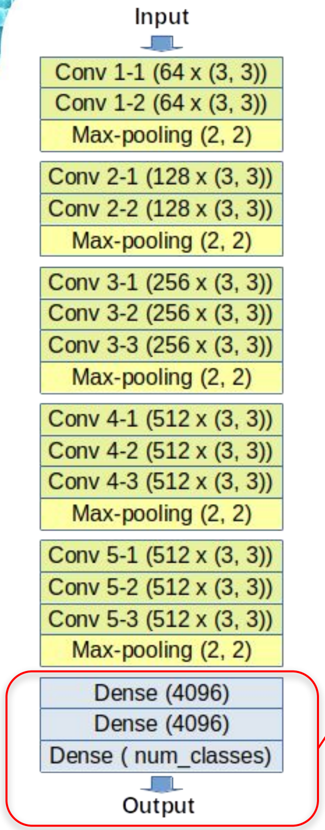


```
in_ = eddl.Input([3, patch_size[0], patch_size[1]])
```

```
def VGG16(in_, num_classes, seed=1234, init=eddl.HeNormal, l2_reg=None, dropout=None):
    x = in_
    x = eddl.ReLu(init(eddl.Conv(x, 64, [3, 3]), seed))
    x = eddl.MaxPool(eddl.ReLu(init(eddl.Conv(x, 64, [3, 3]), seed)), [2, 2], [2, 2])
    x = eddl.ReLu(init(eddl.Conv(x, 128, [3, 3]), seed))
    x = eddl.MaxPool(eddl.ReLu(init(eddl.Conv(x, 128, [3, 3]), seed)), [2, 2], [2, 2])
    x = eddl.ReLu(init(eddl.Conv(x, 256, [3, 3]), seed))
    x = eddl.ReLu(init(eddl.Conv(x, 256, [3, 3]), seed))
    x = eddl.MaxPool(eddl.ReLu(init(eddl.Conv(x, 256, [3, 3]), seed)), [2, 2], [2, 2])
    x = eddl.ReLu(init(eddl.Conv(x, 512, [3, 3]), seed))
    x = eddl.ReLu(init(eddl.Conv(x, 512, [3, 3]), seed))
    x = eddl.MaxPool(eddl.ReLu(init(eddl.Conv(x, 512, [3, 3]), seed)), [2, 2], [2, 2])
    x = eddl.ReLu(init(eddl.Conv(x, 512, [3, 3]), seed))
    x = eddl.ReLu(init(eddl.Conv(x, 512, [3, 3]), seed))
    x = eddl.MaxPool(eddl.ReLu(init(eddl.Conv(x, 512, [3, 3]), seed)), [2, 2], [2, 2])
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

VGG16



```
    x = eddl.Reshape(x, [-1])
    x = eddl.Dense(x, 4096)
    if dropout:
        x = eddl.Dropout(x, dropout, iw=False)
    if l2_reg:
        x = eddl.L2(x, l2_reg)
    x = init(x,seed)
    x = eddl.ReLu(x)
    x = eddl.Dense(x, 4096)
    if dropout:
        x = eddl.Dropout(x, dropout, iw=False)
    if l2_reg:
        x = eddl.L2(x, l2_reg)
    x = init(x,seed)
    x = eddl.ReLu(x)
    x = eddl.Softmax(eddl.Dense(x,
num_classes))

return x
```

Building the model:

- optimizer and its params
- loss function
- metric
- device

```
net = eddl.Model([in_], [out])
eddl.build(
    net,
    eddl.rmsprop(lr),
    ["soft_cross_entropy"],
    ["categorical_accuracy"],
    eddl.CS_GPU(gpus, mem=mem) if
gpus else eddl.CS_CPU()
)
```

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Training and validation

Training

```
### Main loop across epochs
for e in range(args.epochs):
    # ...
    ### Training
    ## Set training mode
    eddl.set_mode(net, 1) # TRMODE = 1, TSMODE = 0
    cd.current_split = 0
    cd.rewind_splits(shuffle=True)
    # ...
    ### Looping across batches of training data
    pbar = tqdm(range(num_batches_tr))

    for b_index, b in enumerate(pbar):
        x, y = cd.load_batch()
        rescale_tensor(x) # to range [-1,1] if tf net

        tx, ty = [x], [y]
        eddl.train_batch(net, tx, ty)
        #...
    pbar.close()
```

For each epoch:

- **Set** the **network mode** to **Training**. This makes the **dropout** layers work (if indicated)
- **Set** the **training split** by selecting the **split** with index **0**
- Randomly reshuffle samples in each split
- Load samples along with their labels by means of the ***load_batch*** **Cassandra Data loader** method
- **Train the network** by using the current batch and labels

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Training and validation

Validation

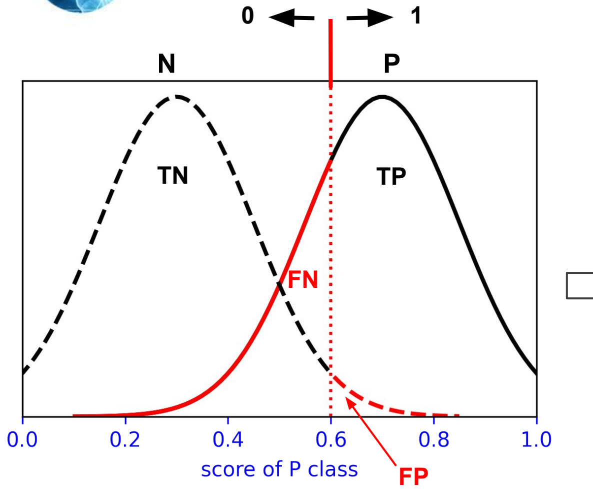
```python
eddl.set_mode(net, 0) # Set test mode.
cd.current_split = 1 ## Set validation split
pbar = tqdm(range(num_batches_val))

for b_index, b in enumerate(pbar):
    x, y = cd.load_batch()
    rescale_tensor(x) # to range [-1,1] if tf net

    eddl.forward(net, [x])
    output = eddl.getOutput(out)

    result = output.getdata()
    target = y.getdata()

#...
pbar.close()
```

- **Set** the **network mode** to **Test**. This deactivates the **dropout** layers (if indicated)
- **Set** the **validation split** by selecting the **split** with index **1**
- Load samples along with their labels by means of the ***load_batch*** **Cassandra Data loader** method
- Perform a **forward pass** and get **results** from the output of the network

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

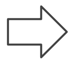
Metrics

Performance Measure: Accuracy, Recall and Precision



Binarization of predictions (th=0.6)

Positive Sample - Pred(P) = 0.75 →[1] →TP
Positive Sample - Pred(P) = 0.58 →[0] →FN
Negative Sample - Pred(P) = 0.62 →[1] →FP
Negative Sample - Pred(P) = 0.22 →[0] →TN



		Actual values N	Actual values P
Predicted values	N	TN	FN
Predicted values	P	FP	TP

Confusion Matrix

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{Total}} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Recall} = \frac{\text{Correct pos preds}}{\text{Positive samples}} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{\text{Correct pos preds}}{\text{Positive preds}} = \frac{TP}{TP + FP}$$

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Metrics

Performance Measure: Accuracy

We use Accuracy to evaluate the validation test during the training process (Recall and Precision are used for cross-validation and test set evaluation)

- **Pros: Intuitive measure**, a model with and accuracy of 90% means that it correctly classifies 90% of observations

- **Cons:**
  - **No information** on single class performance.

Predictions \ Actuals	N	P
N	40	0
P	10	50

Rec = 100%
Prec = 83%

Predictions \ Actuals	N	P
N	50	10
P	0	40

Rec = 80%
Prec = 100%

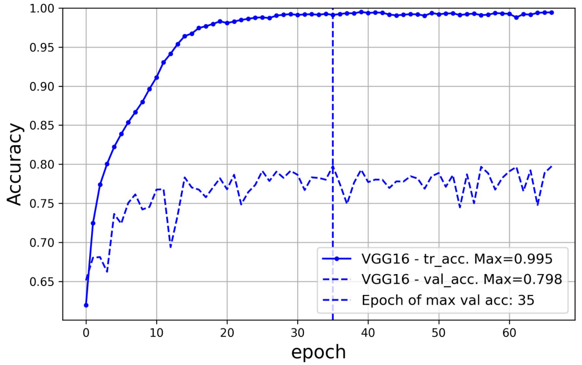
Acc = 90%

  - It **does not work** with **imbalanced datasets**.
    For example, consider we have 98 dogs and 2 cats, and our model provides only dogs as output.
    The accuracy is 98/100 = 98%. **The model has high accuracy but** is obviously not a good model and it **fails to generalize**.

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Improving Performance

The Accuracy of the validation set is generally lower than the training set. This is due to overfitting. The model is able to learn the relationship between the training set images and their labels but lacks generalization capabilities.



- **Get More (or better) training data:** Time consuming and demanding task because a pathologist has to annotate new slides
- **Early stopping:** Don't wait for the training loss to converge, just get the model with the best validation accuracy.
- **L1, L2 Regularization:** Add a penalty term to the loss function
- **Dropout:** Some neurons of the selected layers are switched off at training time (remember to reactivate them at evaluation time)
- **Data augmentation:** Random image transformations (e.g.: flip, rotation)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Improving Performance

Stain Normalization

What is H&E stain:

- **Hematoxylin** and **eosin** (**H&E**) stain is used to emphasize cell and tissue structure details to help pathologists in analyzing biopsies of suspected cancer.
- It is the most widely used stain.
- The hematoxylin stains cell nuclei a purplish blue, and eosin stains the extracellular matrix and cytoplasm pink. The other structures take on different shades, hues, and combinations of these colors.

Why Normalize:

- Histological slides from **different labs** (and even across batches within the same lab) **show heterogeneous appearance** as a result of the different preparation and staining procedures.
- This **variability** can heavily **affect the result of automatic image analysis algorithms**.

How we normalize:

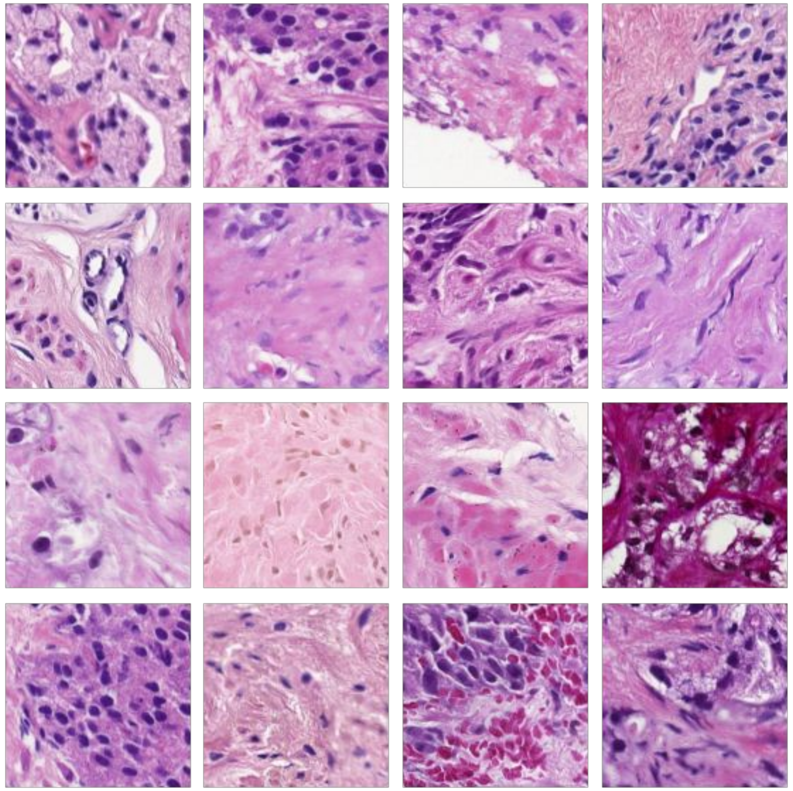
- We used the **Stain separation method** proposed in the paper “**Structure-preserving color normalization and Sparse Stain Separation for Histological Images**”(1).
- **We normalized patches** within a Cassandra data-table **storing the normalized version to a new data-table**. This allowed us to make a direct comparison by using the same splits but taking patches either with or without normalization

(1) A. Vahadane et al., "Structure-Preserving Color Normalization and Sparse Stain Separation for Histological Images," in IEEE Transactions on Medical Imaging, vol. 35, no. 8, pp. 1962-1971, Aug. 2016, doi: 10.1109/TMI.2016.2529665.

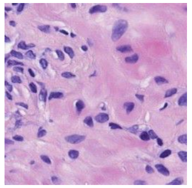
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Improving Performance

Stain Normalization: Structure-preserving color normalization



$$\begin{tikzcd}
\boxed{RGB_t} \arrow[d] & & \boxed{RGB_r} \arrow[d] \\
\boxed{OD_t = D_t C_t} \arrow[dr] & & \boxed{OD_r = D_r C_r} \arrow[dl] \\
& \boxed{\text{Target Normalization}} \arrow[d] & \\
& \boxed{OD_t = D_{t\_resc} C_r} \arrow[d] & \\
& \boxed{RGB_{t\_norm}} &
\end{tikzcd}$$

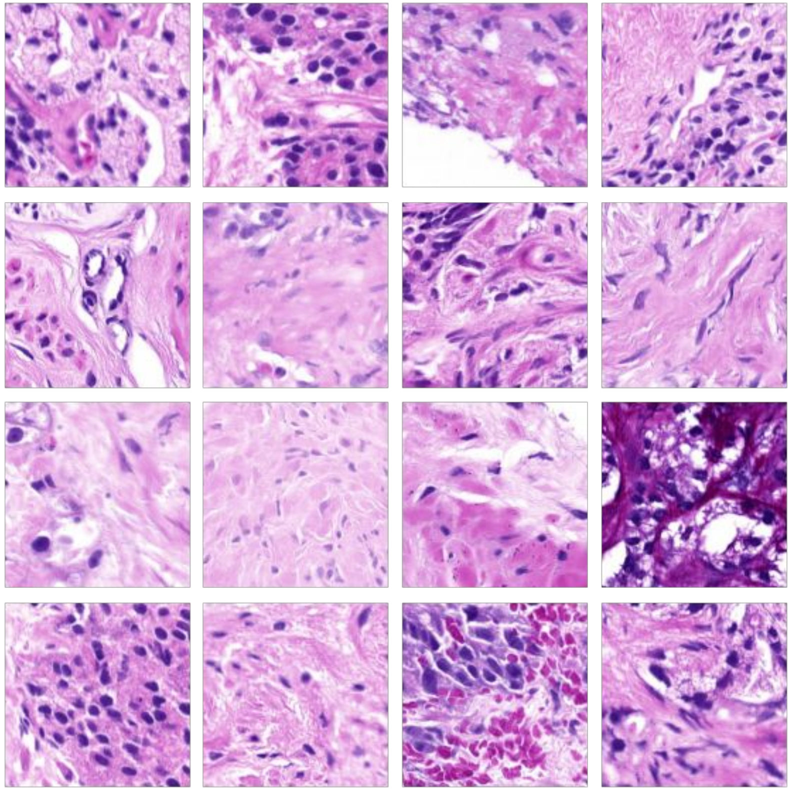


Reference Image

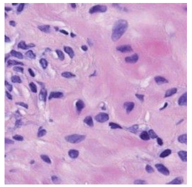
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Improving Performance

Stain Normalization: Structure-preserving color normalization



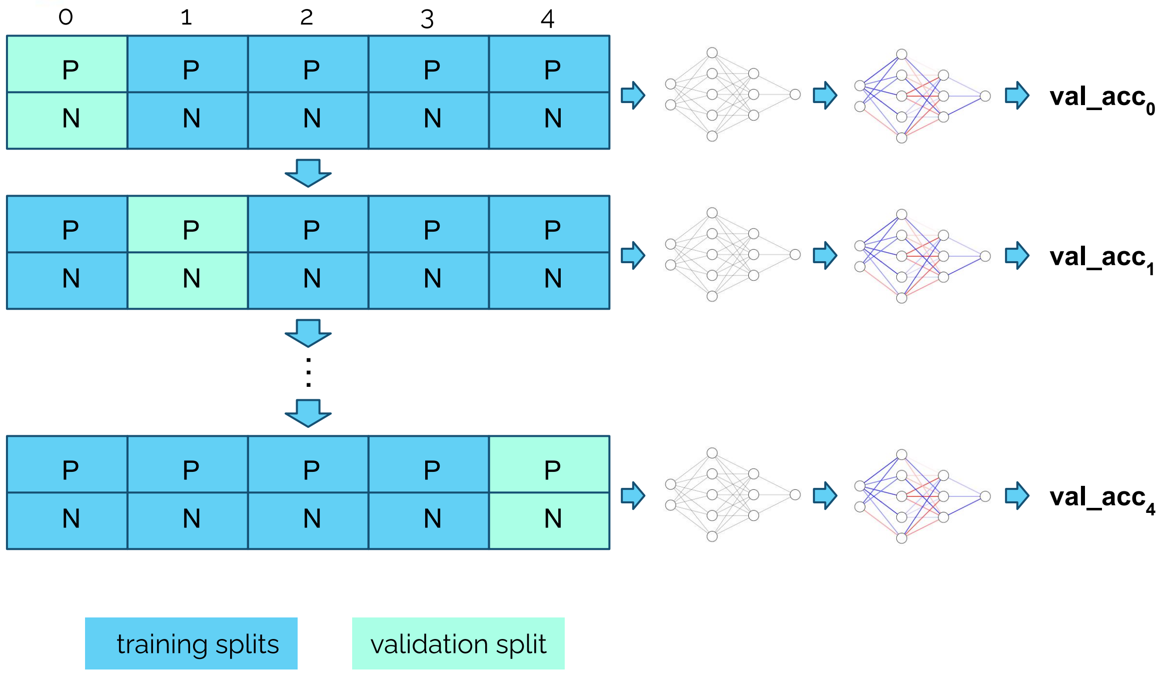
$$\begin{tikzcd}
\boxed{RGB_t} \arrow[d] & & \boxed{RGB_r} \arrow[d] \\
\boxed{OD_t = D_t C_t} \arrow[dr] & & \boxed{OD_r = D_r C_r} \arrow[dl] \\
& \boxed{\text{Target Normalization}} \arrow[d] & \\
& \boxed{OD_t = D_{t\_resc} C_r} \arrow[d] & \\
& \boxed{RGB_{t\_norm}} &
\end{tikzcd}$$



Reference Image

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Cross-validation



cross-validation with 5 splits:

- Five splits are created with a split ratio of [1, 1, 1, 1, 1] to have almost equal sized splits.
- The validation set is selected with a round robin policy
- The training set is composed of the remaining splits

Model Accuracy:

It is the average of **val_acc$_i$** with i in [0,4]

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Cross-validation data loading

Training

```
for b_index, b in enumerate(pbar):
    x, y = cd.load_batch_cross(not_splits=val_splits)
    rescale_tensor(x) ### [0,1] or [-1,1]

    tx, ty = [x], [y]
    eddl.train_batch(net, tx, ty)
    #...
pbar.close()
```

Validation

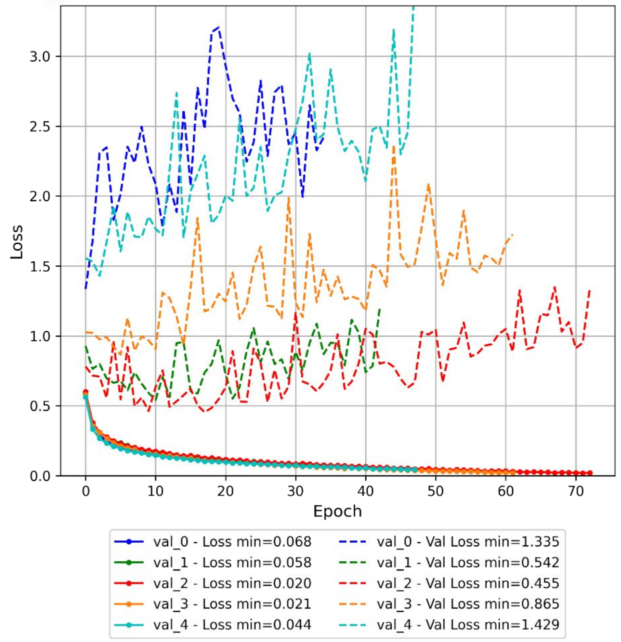
```
for b_index, b in enumerate(pbar):
    x, y = cd.load_batch_cross(not_splits=train_splits)
    rescale_tensor(x)

    eddl.forward(net, [x])
    output = eddl.getOutput(out)
    #...
pbar.close()
```

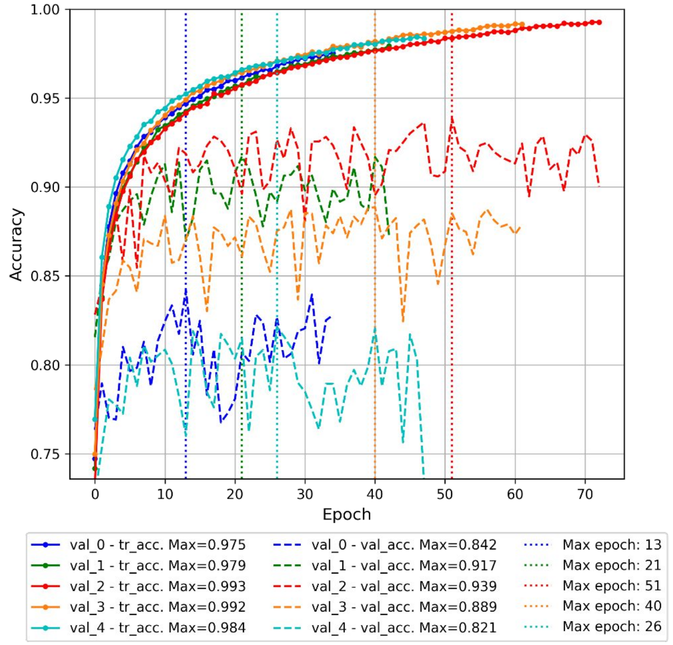
- To make **cross-validation** work and load the right data during the training and validation steps, the ***load_batch*** function must be replaced with the ***load_batch_cross*** function.
- ***load_batch_cross*** takes a list of split indexes as the input argument and load batches from all the splits except those specified in the list
- For example, we have 5 splits and we want to use split 2 for validation. We will provide the list [2] to ***load_batch_cross*** during the training phase and the list [0,1,3,4] during the validation step

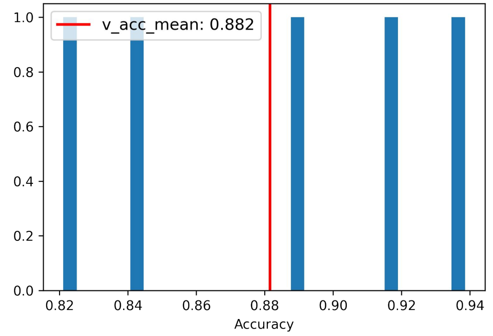
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Model evaluation

Cross validation results



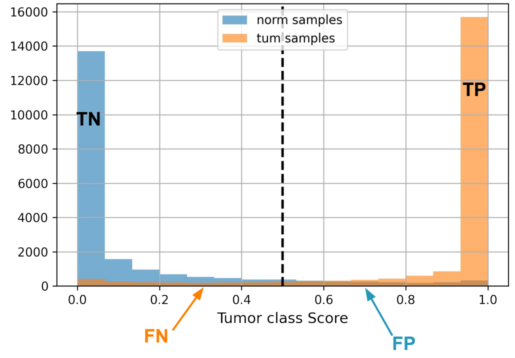


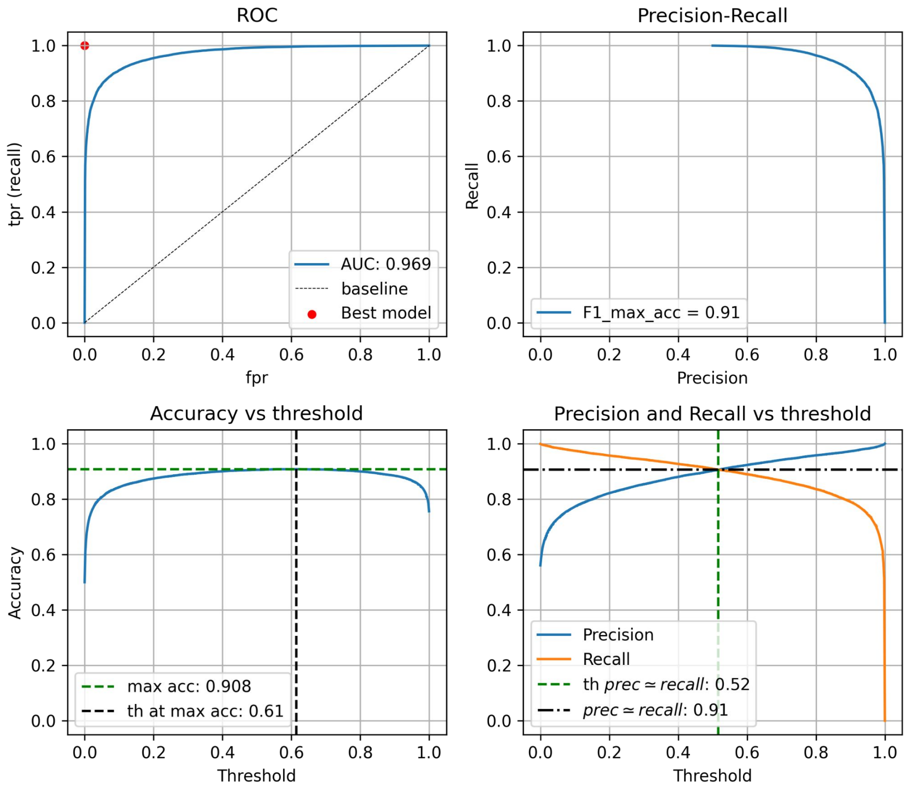


The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Model evaluation

Test Set Results





$$\text{FPR} = \frac{\text{FP}}{\text{FP + TN}}$$

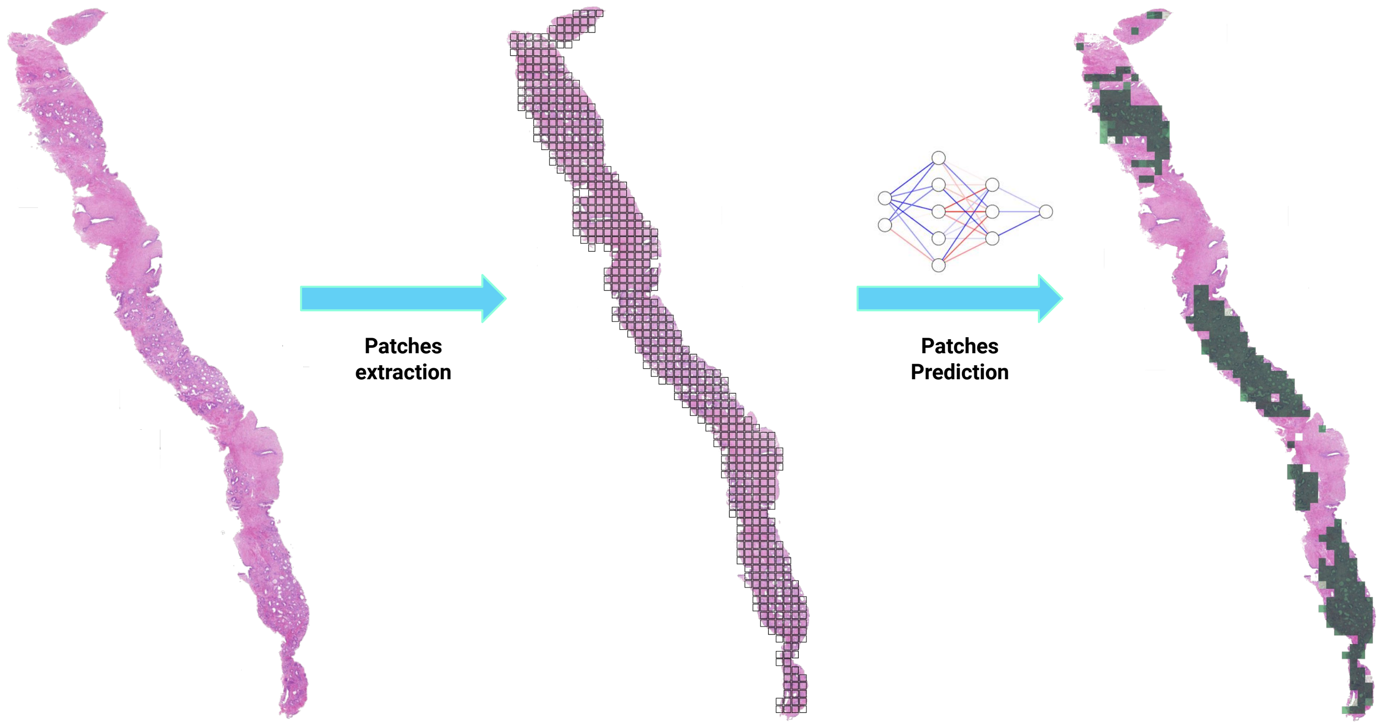
$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}}$$

$$\text{Recall(TPR)} = \frac{\text{TP}}{\text{TP + FN}}$$

$$\text{F1 score} = \frac{2 \times \text{Prec} \times \text{Recall}}{\text{Prec + Recall}}$$

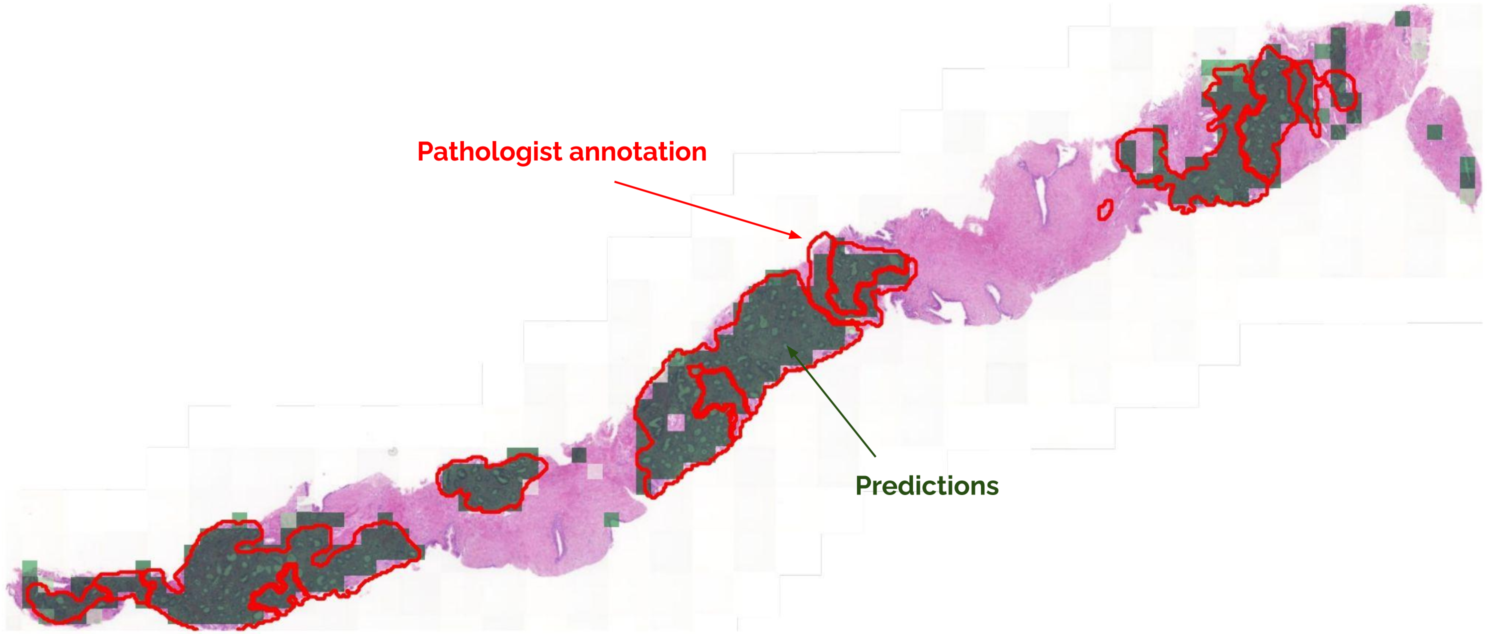
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Slide prediction and visualization



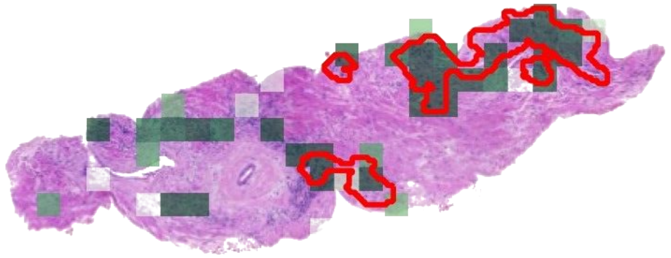
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

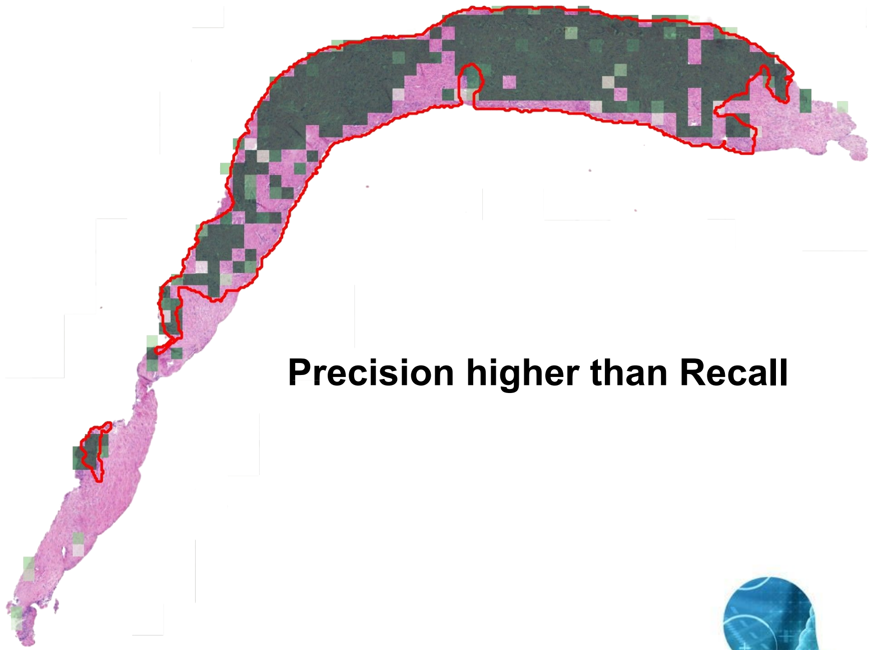
Slide prediction and visualization



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

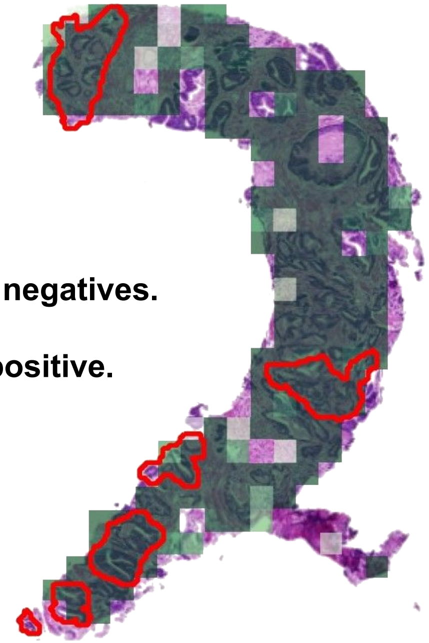
Slide prediction and visualization


Precision lower than Recall

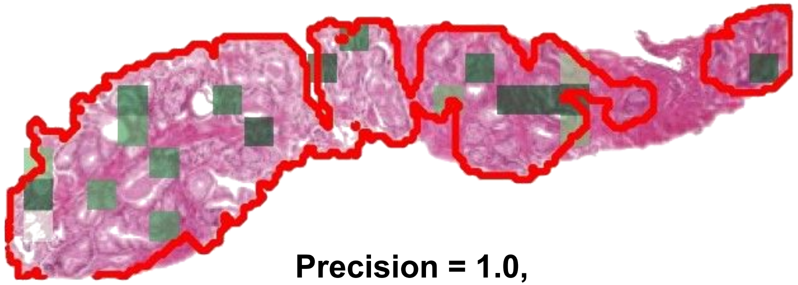

Precision higher than Recall

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Slide prediction and visualization



Recall ≈ 1.0,
almost no False negatives.
Poor Precision,
plenty of False positive.



Precision = 1.0,
no False positive.
Poor recall,
plenty of False negative.

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

How to fight mispredictions

- **Data Preprocessing**:
  - Remove noisy patches
  - More sophisticated normalization algorithms (e.g.: GAN based)
- **Try more recent neural network** architectures (e.g.: ResNet, Xception, EfficientNet)
- Use different types of **data augmentations**
- Increase the dataset size by using **more uncorrelated images** (better to have a lot of cases with a small number of slides each rather than having a small number of cases each with many slides)
- **Increase the number of patches** of the class for which the classifier shows worse performance

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

DEEPHEALTH

Winter School 2022

Thank you!

Q&A

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.