

CNN aided diagnosis tool

Project #7.4

Jacopo Maggio

17 gennaio 2020

Jacopo Nasi

Sofia Ostellino

Summary

[**Description of the assignment of the project** 2](#_Toc29942525)

[**Theory about CNNs** 3](#_Toc29942526)

[Bayesian CNN 4](#_Toc29942527)

[**Description of the method** 5](#_Toc29942528)

[Dataset creation 5](#_Toc29942529)

[The network architecture 6](#_Toc29942530)

[Network details 7](#_Toc29942531)

[The attention maps 9](#_Toc29942532)

[GUI and visualization 9](#_Toc29942533)

[Backend 9](#_Toc29942534)

[**Results** 15](#_Toc29942535)

[**Future development** 19](#_Toc29942536)

[**References**: 20](#_Toc29942537)

## **Description of the assignment of the project**

#### Digital pathology: introduction to the problem

In digital pathology a diagnosis is carried out by analysing histopathological samples which are pieces of tissues extracted via surgical operation. Specimens are typically stained with H&E (haematoxylin and eosin) so that different structures come in different shades between blue (haematoxylin binds to cell nuclei as they are negatively charged) and pink (eosin binds to extracellular matrix and cytoplasm as they are positively charged) for better distinguish between them [1]. Nowadays digital slides are obtained by scanning specimens placed on conventional glass slides; such multi-resolution slides are called WSI (Whole Slide Images) and can be elaborated numerically, enabling different applications.

It is well known that molecular expression of diseases tend to manifest in differences in the tissue architecture and morphology: the traditional approach consists in the visual examination of samples, carried out by a clinician, with the aim of detecting abnormalities related to a certain disease (i.e. if a tissue is cancerous or not). Visual examination is time consuming, prone to inter-reader and intra-reader variability, strongly depends on the skills of the operator and non-reproducible as the human eye is less adept to recognize changes in the tissues: these issues can be overcome making available to the pathologists a tool that supports them during the visual evaluation. [2]

Computed aided diagnosis systems (CAD) are thought to help clinicians in everyday tasks: the clinician is not put aside, but yet supported by tools that can, among all, improve the prediction of disease aggressiveness and of the patient outcome by suggesting details about, for example, medical images without substituting the clinician in the final decision. [3]

#### The focus of the project and clinical insights

#### The focus of the project was developing a software able to produce an attention-map for cancer detection that drives pathologists’ attention to certain areas of the slice that might be pathological and might require further analysis. [4]

Adenocarcinoma and adenoma tissue samples were considered: adenocarcinoma is a cancerous tumour that interests epithelial tissue (i.e. tissue that interests inner and outer surface cavities in many organs and blood vessels) that has glandular origin and/or characteristics. On the other hand, adenoma is a benign tumour, but should be treated as pre-cancerous and requires attention because might turn into adenocarcinoma.

Such clinical aspects were kept into constant consideration during the development of the project: there is a huge variability between the appearance of tissues (cancers usually contain cells that are different grades) and intrinsic uncertainty that was modelled with an approximation of a Bayesian CNN trained with WSI images representing AD tissue (short for adenoma, a benign tumour of epithelial tissue with glandular origin or characteristics), AC tissue (short for adenocarcinoma, a malign tumour) and healthy tissue.. It can be seen that the tissue progressively loses coherence in gland patterns as it becomes carcinogenic. The morphology of the tissue or characteristics of nuclei are hallmarks for cancerous conditions: different metrics have been developed by clinicians for describe (after visual examination of samples) cancer basing on how abnormal the cells look and how quickly they grow; digital pathology makes possible a quantitative characterization of pathology imagery that is important not only for clinical purposes, but also for research, when providing reliable and innovative metrics of evaluation. [2]

## **Theory about CNNs**

#### CNNs

Neural networks are a machine learning approach that relies on several computational units, called neurons, differently interconnected via weights as NNs take inspiration from the way the brain is organised: weighted interconnections between neurons stand for biological synapses. The knowledge of the network is preserved in the weights and the behaviour of the network is related to its hierarchical architecture: the learning process consists in adjusting the weights extracting a mathematical model that fits training data giving as output a classification result or a prediction. [2] Every deep learning network begins with the assumption of random initialization of weights and, at each iteration, data is propagated through the network to compute the output.

There are many challenges in the automatic analysis of digital pathology images, as said before, such as the variability of the morphology of the sample due to the pathology and to the preparation of slides and the variations in staining.

The variations between patients and clinical conditions have always made tedious to find handcrafted features that can be integrated in a system making it robust, efficient and reliable : deep learning methods overcome these issues deriving a feature space from the data itself and gaining the capability of generalization when unseen data is presented to the network. [3]

CNNs (Convolutional Neural Networks) are neural networks where the local connectivity pattern between neurons is inspired by the organization of the animal visual cortex and information is processed similarly to how the brain would do; cortical neurons respond to stimuli in a specific region of the space known as receptive field and this behaviour can be mathematically modelled via convolutions.

The 3 characteristic layers of CNN are the convolutional layers, the non-linear layers and pooling layers. The core of the network are the convolutional layers where, via a set of filters (kernels), feature maps are obtained from the input image and fed to the non-linear layer, characterized by an activation function: after this, the pooling layer reduces the number of features. As hidden units are connected to local receptive fields and share weights resulting in spatial invariance (i.e. a pattern can be recognized in different areas of the input image) and an optimization of the computation, the input can have a high dimension without resulting in many parameters: these parameters are learned during the training via the backpropagation algorithm.

Each hidden layer is dedicated to identifying a multiple feature of the input: low-level features are condensed in the deepest layers while problem-specific features belong to last layers (with no pre-existing assumptions about the particular tasks or dataset in form of encoded domain-specific information); such characteristics allow the network to be more flexible when extracting, during the training procedure, different combination of small patterns eventually combining them for the aim of the network. Regarding the training procedure, the backpropagation algorithm is the most used method and consists in the update of the weights, initially random initialised, basing on a loss term that is computed with the output given by the network and the desired output. The features propagate through the network in the forward direction computing the output and the training loss is derivate with respect to the weights and computed back towards the input.

This is an iterative procedure that is repeated until a certain stopping condition is reached: the tuning of the parameters of the backpropagation is proportional to the size of data.

### Bayesian CNN

The risk of overfitting when the network is not trained on a large dataset and the falsely overconfidence in the prediction related to the absence of a measure of uncertainty are typical drawbacks of conventional deep learning methods, based on point predictions.

Bayesian CNNs were thought to handle these problems and considered during this project: they are based on the Bayes’ theorem that is the fundamental of Bayesian inference, a way of quantifying model uncertainty. According to this, each observation is an opportunity to update the beliefs about a given deep learning model. Moreover, Bayesian CNNs are robust to outliers and are key solution when the lack of a large amount of data can result in unreliable networks.

#### Bayes’ theorem

The Bayes’ rule shows how the degree of belief in a model (posterior function, ) is related to the likelihood of the occurrence of the data , to the knowledge about the data (the prior ,) and to the evidence (marginal likelihood, ).

The posterior function is the probability distribution of interest that summarizes the knowledge about the model parameters given data and needs to be estimated given that the aim is obtaining the parameters of the model in order to get the correct output for a given input. The prediction of new observations is made through model update on the posterior predictive distribution, the neural network of interest being a conditional model parameterized by the weights.

The exact Bayesian inference is intractable, and Bayesian CNNs come with a high computational cost: the estimation can only be approximated via several method. [8]

Stochastic regularization techniques like dropout regularization can be used to approximate inference in Bayesian models without sacrificing either computational complexity or test accuracy. [6]

#### Dropout

Dropout is a regularization technique that prevents overfitting and improve generalizability by randomly ‘dropping out’ (i.e. inactivating) units of a neural network with a certain probability: different units are dropped out, resulting in a training procedure on reduced networks. (Deep convolutional neural networks for automatic classification of gastric carcinoma using whole slide images in digital histopathology [Harshita Sharmaa](Harshita%20Sharmaa))

In the Monte Carlo dropout procedure dropout is applied at both training and test time: setting the dropout rate and the number of iterations, the same element of the dataset is presented to the network different times and, at every forward pass of the same input, a different result is obtained according to the dropout rate.

The uncertainty for one sample is derived collecting the model outputs of such sample: the average and the variance of the sample lead to an ensemble prediction. The average stands for the mean of the posterior distribution for the sample and the variance stands for the uncertainty of the model regarding the sample. [8]

## **Description of the method**

### Dataset creation

The dataset consists in digital histopathological images of different dimensions, belonging to different patients and representing different classes: adenoma AD, adenocarcinoma AC and healthy tissue H.

As the dataset consisted in only 111 images, crops had to be generated with the purpose of data augmentation.

1. Images were cropped with squared crops of 1344x1344, 2240x2240 and 3136x3136 pixels. Crops dimensions were chosen for incorporating information at multiple resolution and at different level of detail. The dimension of crops kept into consideration domain knowledge and required a visual analysis of images at different resolutions using the Aperio ImageScope tool, investigating the appearance of the tissues isolated with different crop sizes, considering the contextual information and neighbourhood: some areas might be difficult to differentiate without neighbourhood information.

It is relevant to say that no information was given to the network about crops that included blood vessels and other part of tissues not of interest and no crops were removed in such sense.

1. No pre-processing steps where applied to the images (such as brightness, contrast and intensity adjustments or affine transformations) in order to preserve the salient texture, colour and morphological properties of the original stained images.
   1. The dataset was divided into training and testing: crop belonging to the same patient were considered entirely or in the training set or in the dataset. This led to an unbalanced dataset that was balanced using 1120x1120 crops randomly obtained from the original image. The integration with such crops instead of the deletion of crops was considered a better approach for the balancing of the dataset: in this way no information is deleted and the dataset is kept representative.

### The network architecture

This is the approximation of a Bayesian neural network obtained via the application of the Monte Carlo dropout during training and testing. The dimensions of the layers and the parameters such as stride and padding where chosen basing of the following formula where I is the input size, K is the filter size, P is the padding and S is the stride.

The input layer is 224x224x3: the crops are resized to this dimension in order to fit the input of the network. The zero padding layer pads the input with zeros around the borders and is applied in order to make the convolutions feasible and, because the size of the volume decreases after a convolution, it is important to preserve information about the original input volume. Dropout layers are placed after every 2D convolutional and dense layer in order to obtain: the dropout rate is a hyper-parameter further discussed. The stride determines how the filters behave and is set so that the output volume is an integer and not a fraction.

The flatten layer collapses the spatial dimensions of the input into a vector that represent the feature extracted by the previous convolutional part of the network; the dense layers (that are fully connected with the ReLu activation function) represent the classification part of the network that ends with the final dense layer (with the Sofmax activation function): returns 3 values representing the probability for each class (AD, AC, H).

### Network parameters selection

1. Dense layer

Three different dense layer dimensions where considered and evaluated: 1024, 2048 and 4096.

2048 was chosen as a good compromise between the number of parameters, training time and the loss; this dense layer leads to a more stable prediction and lower final validation loss with respect to 4096 and 1024.

* 4096 total parameters: 134.272.835
* 2048 total parameters: 70.299.459
* 1024 total parameters: 41.458.499

1. Dropout rate

It can be seen that with a dropout of 0.5 the loss should be evaluated on more than 10 epochs: the first value of the training loss is out of range and is not visible in the graph.

1. Epochs

The training was performed for 10 epochs with a batch size of 32 samples with different training time depending on the drop out.

1. Batch size

Batch size is a hyperparameter of gradient descent that control the number of training samples utilized in one iteration: according to the literature the batch gradient descent popular sizes include 32, 64 and 128 samples. [9] The choice of the batch size is driven by consideration about memory issues and training time: a small batch results in a high training time with a less accurate estimation of the gradient and a higher batch, in addition to the training time, in using too much memory.

## **Implementation**

### The attention maps

In order to assign the class label to a crop, according to the MC Dropout method, the maximum between the mean probability for each class. (AC, AD, H) computed among all predictions is considered.

The attention map is obtained putting together all the processed crops of an image, integrating the information about probability and uncertainty with an approach Healthy VS Pathological.

The colour is related to probability of the prediction and the intensity of the proportional to the the variance (the higher the variance, the higher the transparency).

More details are discussed in the section related the Attention Map generation in the Backend paragraph.

### GUI and visualization

The GUI was developed basing on considerations about the user experience: the clinician needs a ready to use and intuitive interface that can display the attention map related to the image taken into consideration.

Features of the GUI:

* Visualization of a single image with the possibility of overlay the attention map over the image and zooming in;
* Batch mode: processing images from a folder for visualize the images once they are all processed.

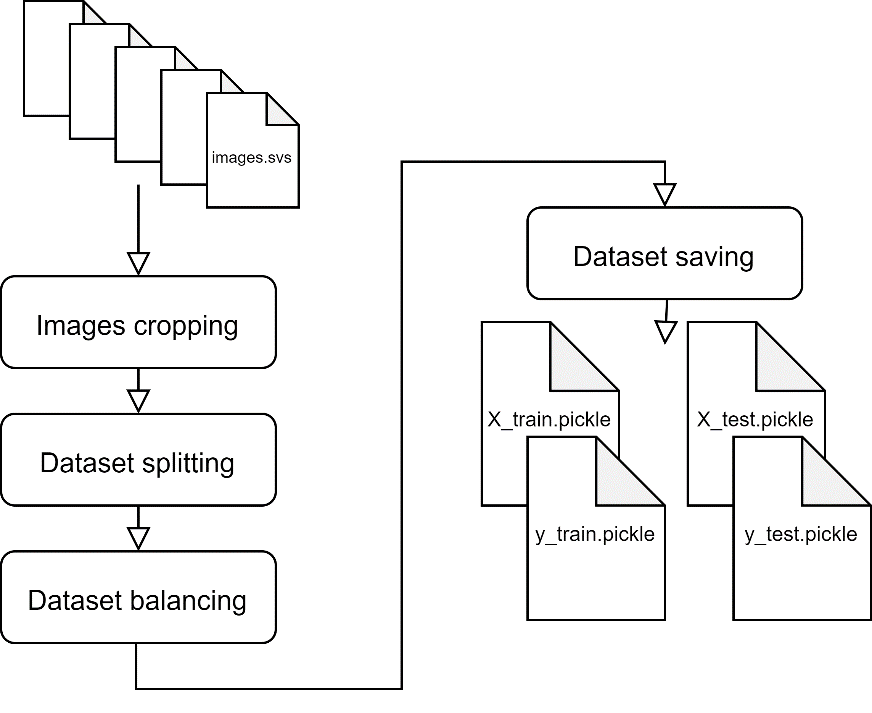
#### The implementation – salient aspects code

### Backend

Backend operations can be classified into three different blocks:

1. Dataset creation. In preparation to the next step, the ROI images provided as input are cropped in order to obtain balanced training and test sets (and related labels sets) saved in binary files.
2. Training. Datasets created in previous step is given in input to the CNN network. A trained model is given as output and saved in a binary file.
3. Prediction. An unclassified .svs image is cropped and given in input to the pre-trained model to obtain the prediction. More than one forward pass through the model is performed and uncertainty measures computed.

#### Dataset creation

1. Images cropping.

Each provided input .svs image has been cut in sub-images then resized to the proper size (224x224x3) to be given in input to the CNN. An oversampling approach has been used, so different crop sizes have been used: 1120x1120, 1344x1344, 2240x2240, 3136x3136.  
The crop step has been performed as follow:

* 1. .svs image opened by means of **Openslide.open\_slide(filename)** function that, given in input a path, returns an OpenSlide object for whole-slide images and an ImageSlide object for other types of images.

try:  
 slide = openslide.open\_slide(slide\_path)  
except Exception:  
 slide = None

* 1. Obtained ImageSlide object converted into a RGB image using method **ImageSlide.read\_region(self, location, level, size)** whichreturns a PIL.Image containing the contents of the region delimited by the points (location.x, location.y), (location.x+size.width, location.y), (location.x, location.y+size.lenght), (location.x+size.width, location.y+size.lenght).

width, height = get\_slide\_size(slide)   
image = slide.read\_region((0, 0), LVL, (width, height))

* 1. To overcome the problem related to the fact that the dimensions of the image are not (necessarily) multiple of the dimensions (custom\_ss x custom\_ss) of the cropping window, overlapped crops have been performed. The number of crops for each dimension has been calculated dividing and rounding up the image dimension by the cropping window dimension. The exceeding portion has been distributed as overlap among all crops. To improve the performance of this operation, otherwise very slow, a multithread approach has been adopted. Every crop has been resized to 224x224x3 dimension using **PIL.Image.resize(self, \*shape)** method.

for i in range(0, len(x\_p)):  
 for j in range(0, len(y\_p)):  
 box = (x\_p[i], y\_p[j], x\_p[i] + custom\_ss, y\_p[j] + custom\_ss)  
 crop\_folder = path.join(CROP\_FOLDER, algorithm\_crop\_folder, str(custom\_ss))  
 crop\_name = slide\_name+'\_'+str(i)+'x'+str(j)+".png"  
 pool.append(Thread(target=custom\_crop,args=(image,box,crop\_folder,crop\_name)))  
 pool[-1].start()  
for p in pool:  
 p.join()

1. Dataset splitting.

Crops of original sizes 1344x1344, 2240x2240, 3136x3136 obtained in the previous step have been loaded (resized to 224x224) with their label and the patient ID to create the dataset. To improve the prediction eventual not useful samples (percentage of white pixels higher than 70% of the sample) have not been added.

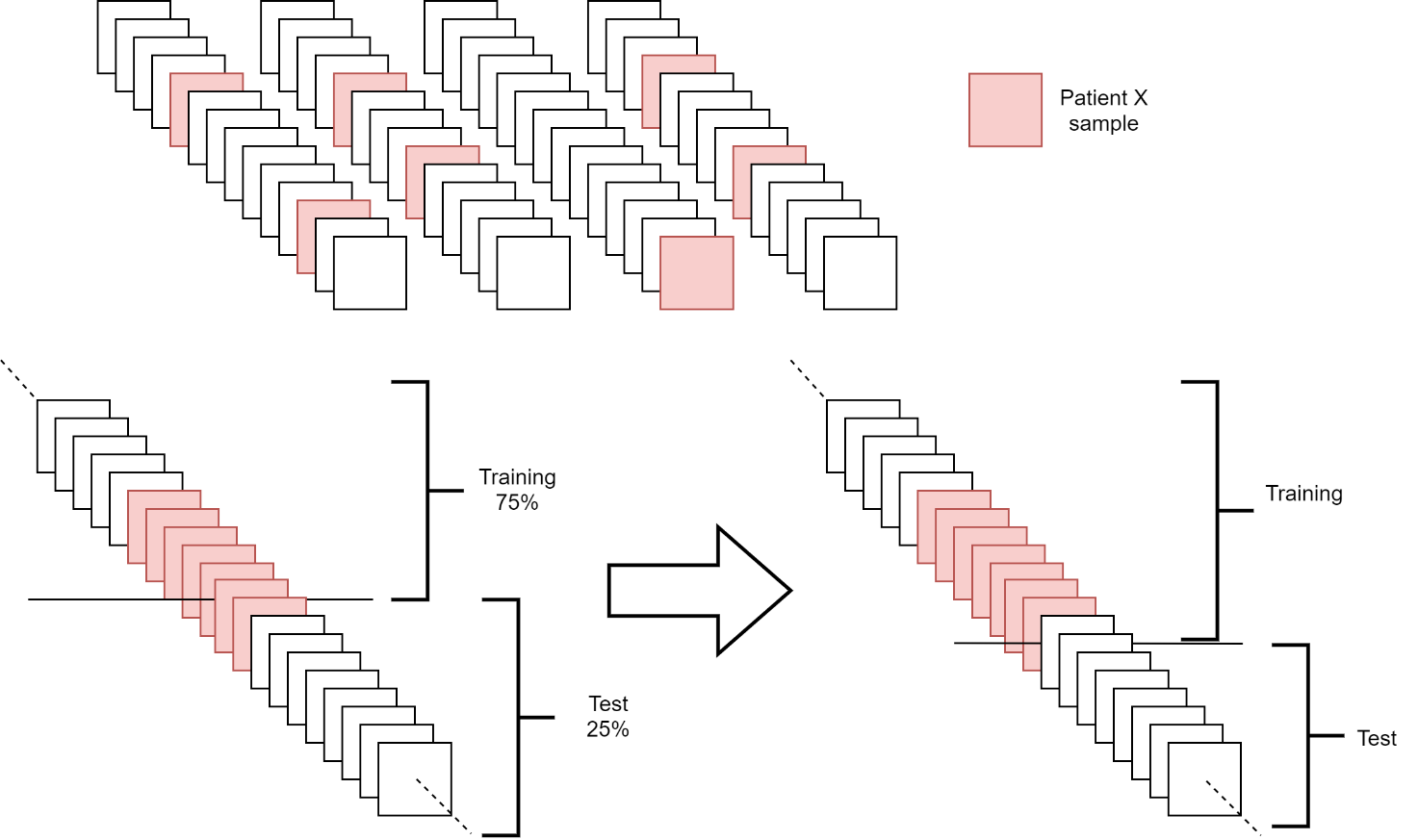
check\_valid(np\_image):  
 valid = True  
 …

black\_px = sum(sum(i == 0 for i in np\_binary))  
 white\_px = sum(sum(i == 255 for i in np\_binary))  
 if white\_px/(white\_px + black\_px) > 0.7:  
 valid = False  
 …

return valid

The list of samples has been randomly shuffled keeping consecutive all crops belonging to the same patient

r = {p\_e: random.random() for x\_e, y\_e, p\_e in dataset}  
dataset.sort(key=lambda item: r[item[2]])

and then split in training and test lists maintaining the proportion 3:1 and keeping all crops belonging to the same patient in just one of the two sets.

1. Dataset balancing.

To have a balanced dataset in which the number of samples belonging to the same class is equal, an oversampling approach has been used again to the set randomly selected crops of original dimension of 1120x1120. The addition followed the previous mentioned criteria:

1. validity according to percentage of white
2. samples belonging to the same patient must be in the same set
3. Dataset saving.   
   By means of **Pickle.dump(obj, file, protocol=None, \*, fix\_imports=True)** the datasets are stored in binary files.

X, y, p = load\_datasets(1344, 2240, 3136)  
pickle\_out = open(x\_path, "wb")  
pickle.dump(X, pickle\_out)  
pickle\_out.close()  
pickle\_out = open(y\_path, "wb")  
pickle.dump(y, pickle\_out)  
pickle\_out.close()  
pickle\_out = open(p\_path, "wb")  
pickle.dump(p, pickle\_out)  
pickle\_out.close()

#### Training

1. Dataset loading  
   Datasets produced in previous step have been loaded into NumPy array by means of **Pickle.load(file, \*, fix\_imports=True, encoding="ASCII", errors="strict")** function.
2. Model Compile

The logical model described above, has been implemented as a sequential model using layers and structures provided by Keras framework.  
In the detail:

1. ZeroPadding2D layer

x = layers.ZeroPadding2D(padding=1)(x)

Add rows and columns of zeros at the top, bottom, left and right side of an image tensor. Take as argument an int representing the same symmetric padding applied to height and width.

1. Conv2D layer

x = layers.Conv2D(filters=64, kernel\_size=3, strides=1, padding='valid', activation='relu', use\_bias=USE\_BIAS, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros')(x)

Creates a convolution kernel with dimension (kernel\_size x kernel\_size) that is convolved with the layer input to produce a tensor of outputs to which an activation function it is applied.

1. Dropout layer

x = layers.Dropout(rate=drop\_rate)(x, training=True)

Applies Dropout to the input, i.e. randomly set a fraction rate of input units to 0 at each update. In general, it is used during training time to prevent overfitting. In the study case, it has been applied also at prediction time to obtain variability in predictions and perform MC Dropout estimations of uncertainty.

1. MaxPool2D layer

x = layers.MaxPool2D(pool\_size=2, strides=2, padding='valid')(x)

Perform a max pooling operation for spatial data to reduce the size of the input tensor according to the size of the filter and the stride.

1. Flatten layer

x = layers.Flatten()(x)

Flattens the input.

1. Dense layer

x = layers.Dense(DENSE\_SIZE,  
 activation = 'relu',  
 use\_bias = USE\_BIAS,  
 kernel\_initializer= 'glorot\_uniform',  
 bias\_initializer = 'zeros')(x)

A fully connected NN layer that implements the operation: **output = activation(dot(input, kernel) + bias)** where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer.

Once the model has been set, the compiling has been performed using **Model.compile(…)**.

bayesian\_model.compile(loss = 'sparse\_categorical\_crossentropy', optimizer = opt,

metrics = ['accuracy'])

Arguments in details:

* **optimizer**: stochastic gradient descent procedure to update network weights. *Adam* optimizer has been chosen using a standard configuration with learning rate set to 1e-3 and decay equal to 1e-6 [10].

opt = tf.keras.optimizers.Adam(lr = LEARNING\_RATE, decay = DECAY)

* **loss**: Loss instance. Since the target of the predictions are integers (representing the classes), *sparse\_categorical\_crossentropy* has been used.
* **metrics**: List of metrics to be evaluated by the model during training and testing.

1. Training

The training of the model took place by giving as input the dataset created to the model compiled as described previously in batch of dimension BATCH\_SIZE for N\_EPOCH epochs (forward and backward passes of the network). At each epoch the set has been shuffled to improve performance and speed up the convergence.

bayesian\_train = bayesian\_model.fit( x\_train, y\_train,  
 batch\_size = BATCH\_SIZE,  
 epochs = N\_EPOCH,  
 validation\_data= (x\_test, y\_test),  
 shuffle = True )

1. Trained model saving

At the end of the above described process, by means of **Pickle.dump(obj, file, protocol=None, \*, fix\_imports=True)** function the trained model has been stored in a binary file.

model\_path = os.path.join(MODEL\_FOLDER, model\_name)  
bayesian\_model.save\_weights(model\_path)

#### Prediction



By means of the **make\_prediction(path\_list, crop\_size, dr, iterations)** function, one or more .svs images can be loaded into the program to be processed in order to obtain the proper attention map according to the parameters passed as arguments.

The process is divided into the following steps:

1. Prediction

The step is performed by **src.datasetManager.mc\_predict\_from\_path(iterations, file\_path, pred\_folder, drop\_rate, crop\_size)** function which loads the .svs image as an Openslide.ImageSlide object, cuts it into sub-regions of dimension *crop\_size* x *crop\_size* x *3* and load a resized version of them (*224* x *224* x *3*) as NumPy matrices into a list using **Openslide.read\_region(self, location, level, size)** method.

To speed up the process, a multi-thread approach is used.

Since during dataset creation crops having a percentage of white greater than the 70% of the image have been removed, the same is done with crops of the image that do not respect this criterion.

for y in range(0, y\_max - 1):  
 pool.append(Thread(target=custom\_crop, args=(slide\_, y, batch\_to\_predict, valid\_bit\_list, x\_max, crop\_size)))  
 pool[-1].start()

for p in pool:  
 p.join()

A number of prediction equal to the argument ***iterations*** in performed giving in input the list of crops to the model trained with dropout rate equal to ***drop\_rate*** and calling the function **Model.predict(…)**.

bayesian\_model = load\_model("224\_10\_32\_0.0001\_1e- 06\_"+str(drop\_rate)+"\_True\_2048\_False\_local.h5", drop\_rate)  
predictions = bayesian\_model.predict(batch\_to\_predict, batch\_size=BATCH\_SIZE, verbose=1, workers=100, use\_multiprocessing=True)

The list of crops, the information related to validity and the predictions (iterations \* 3 values of probability for each crop) are stored in a binary file in order to be loaded when needed instead of repeating the mentioned process.

1. Ensemble predictions computation

The step is performed by **src.datasetManager.compute\_ens\_predictions(mc\_predictions)** which taken in input the lists of predictions computes the mean and the variance of the predictions for each crop.

for i in range(0, len):  
 midx, mean, std, h\_mean= get\_crop\_pred(mc\_predictions, i)

get\_crop\_pred(mc\_predictions, idx):  
 p0 = np.array([p[idx] for p in mc\_predictions])  
 p0\_mean = p0.mean(axis=0)  
 max\_idx = p0\_mean.argmax()  
 return max\_idx, p0\_mean[max\_idx], p0.std(axis=0)[max\_idx], p0\_mean[2]

Means, variances and class predictions are stored in a binary file in order to be loaded when needed instead of repeating the mentioned process.

1. Attention Map generation

The step is performed by **src.blender.blend\_np\_std\_bigradient(image\_np, ens\_prediction, valid\_bit\_np, slide\_size, print\_std, crop\_size)** function which, adopting a multithread approach, blends each crop with a mask generated as follow:

* 1. the variances of all predictions related to the image are normalized in a range between 0 and 1
  2. the colour of the mask is selected according to the predicted class, i.e. for health crops is selected in a gradient of color from white to light yellow and for carcinogenic tissue into a range of color from light yellow to red.



HEALT CARCINOGENIC

colors\_h = list(lgyellow.range\_to(white, 69))  
colors\_adc = list(lgyellow.range\_to(yellow, 42)) + list(yellow.range\_to(dred, 27))

get\_prob\_color\_bigradient(gradient, probability):  
 idx = int(probability \* 100) - 32  
 return gradient[idx]

* 1. the transparency of the mask is proportional to the uncertainty value (the variance), i.e. higher uncertainty means higher transparency of the map. Using the function **PIL.Image.blend(im1, im2, alpha)** and passing the value 0.5 \* (1.0 – variance) as alpha value, the interpolation of the crop and the mask happens according to the formula

*out = image1 \* (1.0 - alpha) + image2 \* alpha =*

*= image1 \* (1.0 – (0.5 \* (1.0 – variance))) + image2 \* (0.5 \* (1.0 – variance))*

that is equal to:

* *out = image1,* (the crop) when the uncertainty is max (=1)
* *out = image1 \* (0.5) + image2 \* (0.5),* (the crop with the mask overlapped) when the uncertainty is min (=0)

blend(image, mask, uncert):  
 return Image.blend(image, mask, 0.5 \* (1-uncert))

The crops are then ensembled and the result image with attention map overlapped is saved and visualized.

## **Results**

Different aspects are analysed in order to estimate the performances as, given the focus of the project, it was not possible to superimpose the manual segmentation to the output of the system (the attention map) computing a metric of evaluation. The evaluation was performed visually, verifying if the obtained attention map is consistent with the provided manual segmentation.

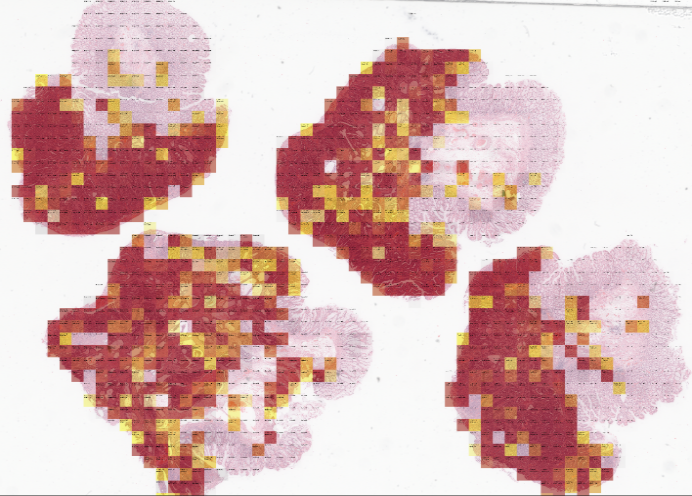


Figure 1 - Example of comparison between the attention map and the manual segmentation

As the probability estimation was conducted via the Monte Carlo dropout it is clear why, comparing the result of the network after only 1 iteration and after 10 iteration, the results improve: areas identified as pathological with 1 iteration are then correctly identified as healthy.

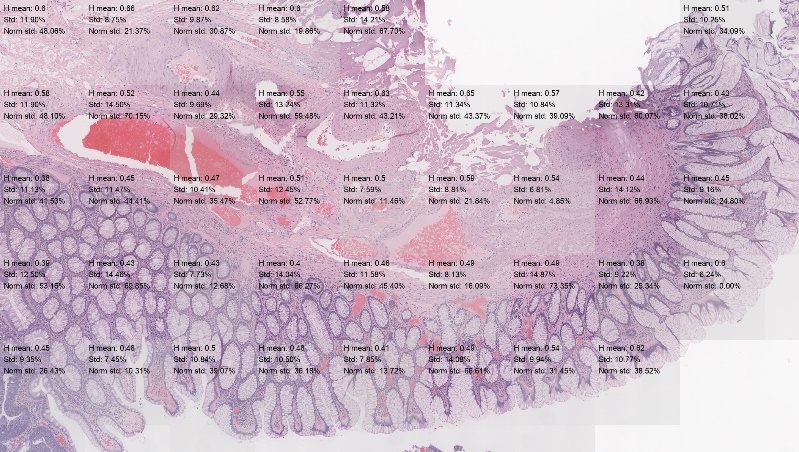
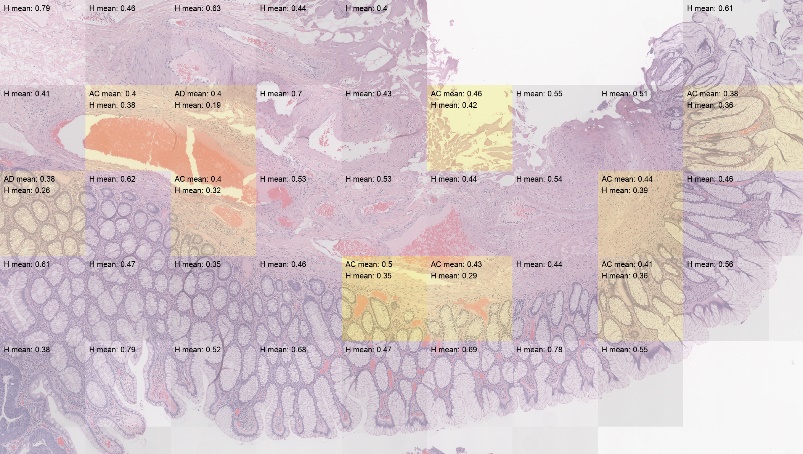


Figure 2 - the result on 1 and 10 iterations with dropout rate = 0.5

If we compare 1, 10 and 100 iterations with a dropout rate of 0.5 and crop size of 4480 (chosen for practical issues) and analyse the variance computed for every class we obtain the following results.

Figure 5 Results of 1, 10 and 100 iterations

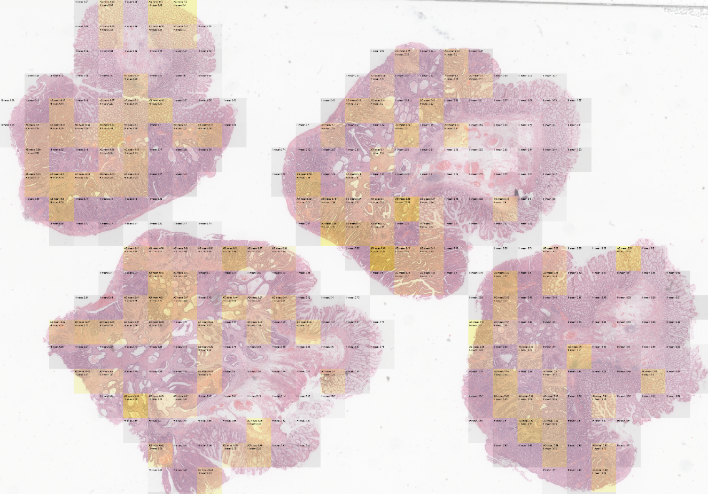
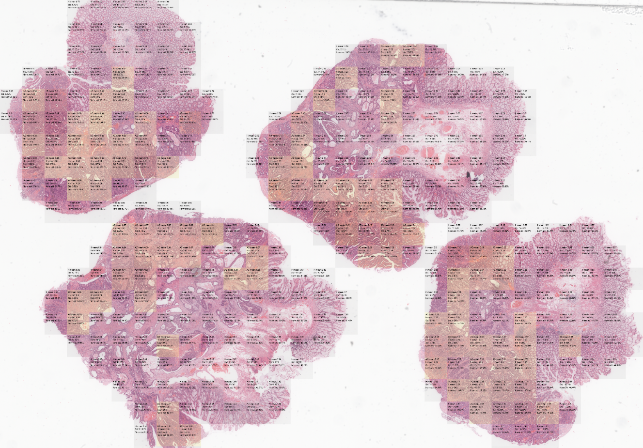
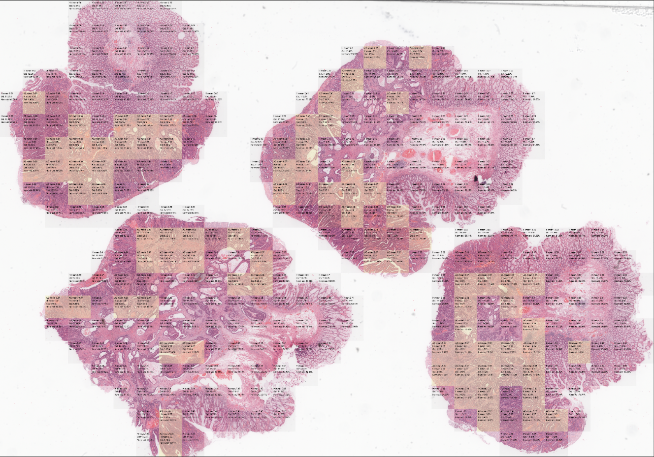
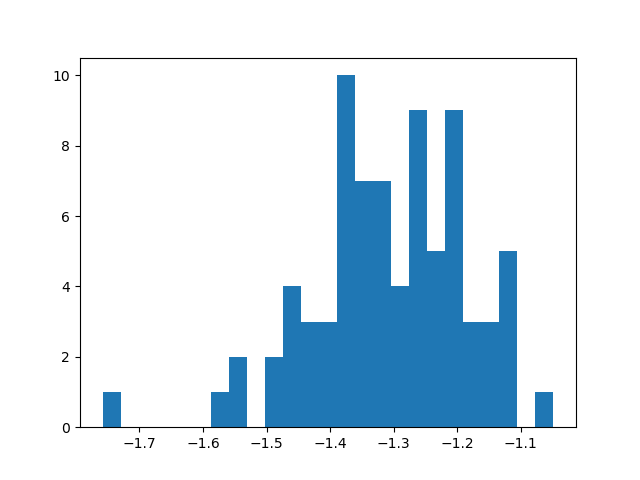
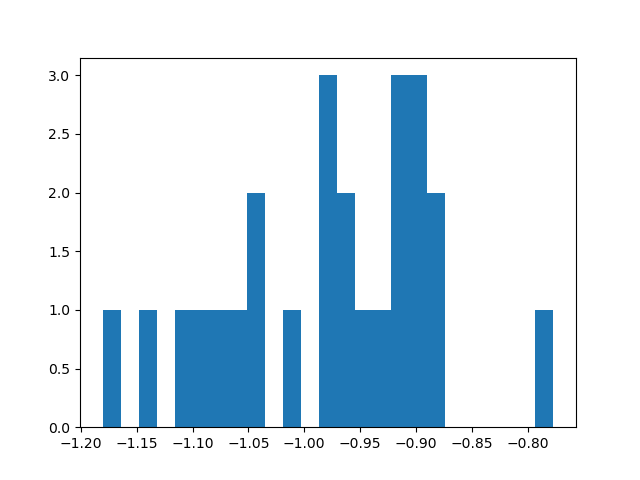


Figure 6 On the x-axis: log(variance)



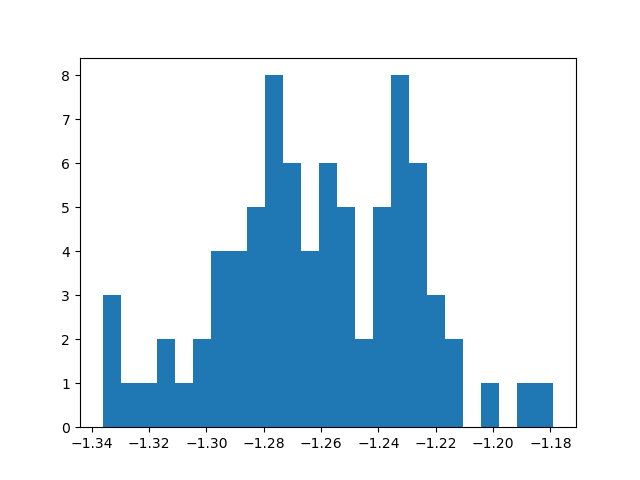
10 iterations, AC



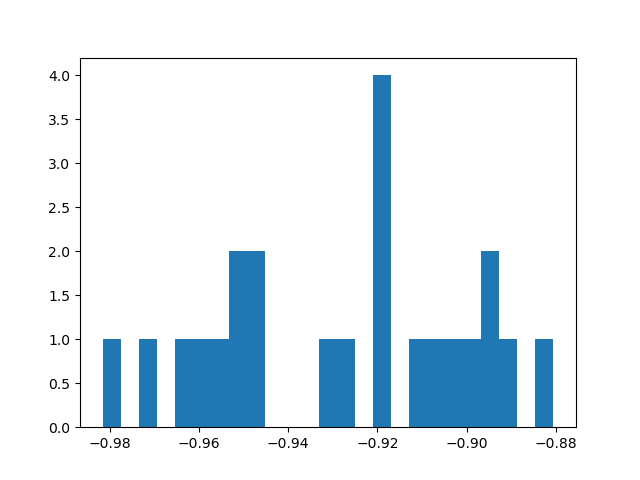
10 iterations, AD



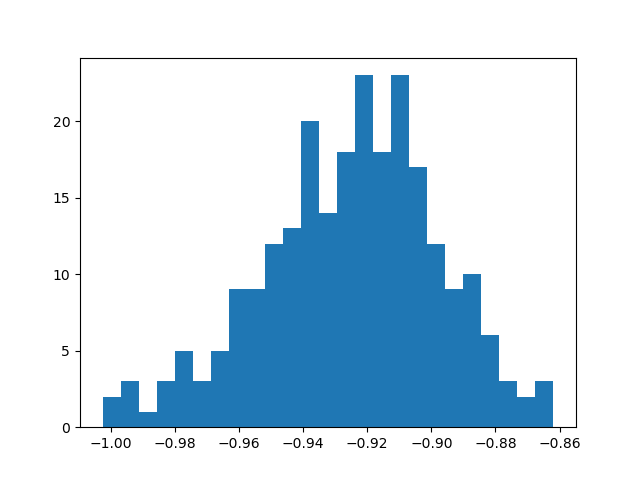
10 iterations, H



100 iterations, AC



100 iterations, AD

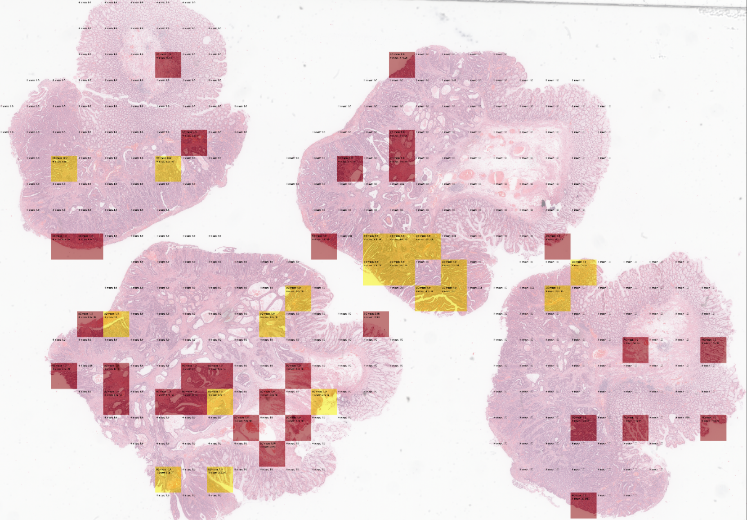
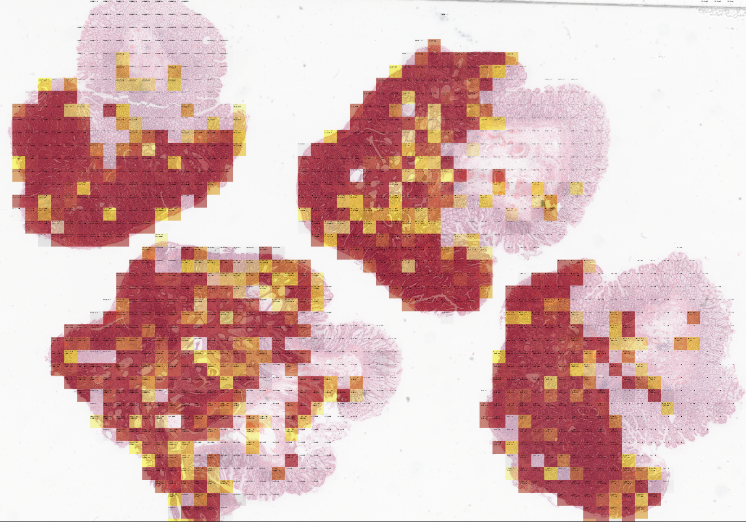


100 iterations, H

**COMMENTO**

Accuracy strongly depends on the dimensions of the crops: if the dimension of the crop increases, the accuracy in the identification of the areas of interest decreases, as it can see in the figure below where with crops of 2240 pixels the network identifies well cancerogenic areas.

Figure 3 - Comparison between the results with a crop size 2240 (left) and 4480 (right)

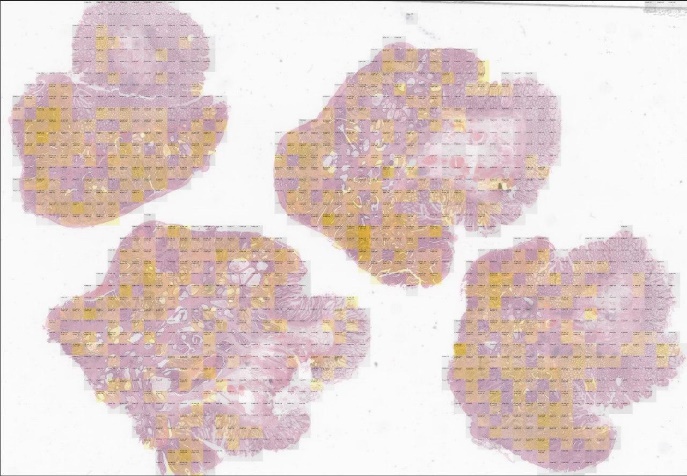
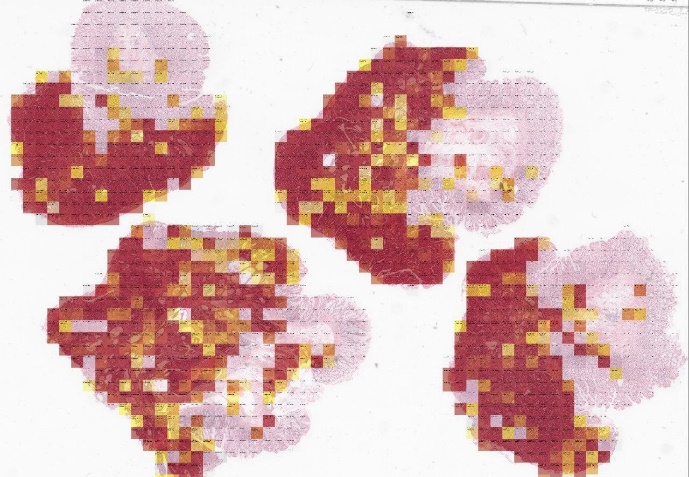
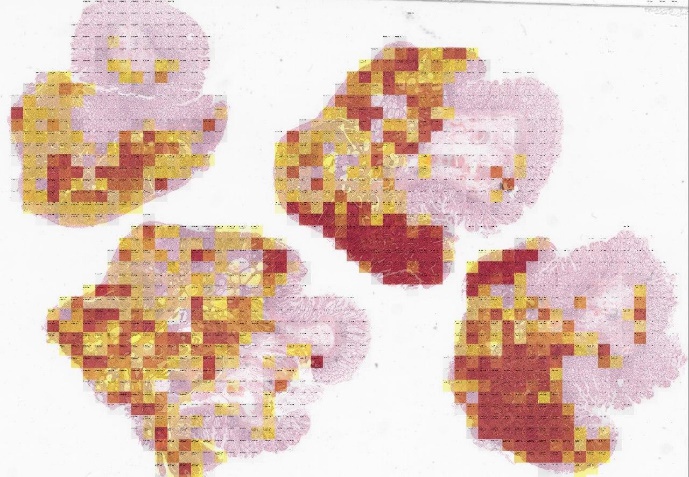


Considering the prediction time, it was observed that it increased around the 20% for a single crop doubling its size. Moreover, the prediction time for a single image strongly depends on its dimensions and the correlated number of crops.

#### 

The dropout rate is a crucial parameter that has been analysed: there is not much difference between the results of a dropout of 0.1 and 0.01, but with a dropout of 0.5 the attention map is not accurate and uncertainty increases.

Figure 4 Comparison between different dropout rates



## **Future development**

The importance of digital pathology and the integration of tools that can help and support decision in the clinical workflow have a huge impact on the treatment and study of pathologies: having the samples electronically scanned makes, for example, easier to collect second opinions and share knowledge; the tool developed in this project might integrate the possibility of annotating images basing on the attention map and share such results between clinicians via tele-histopathology.

An integration of the dataset should be taken into consideration as future development in order to make it more consistent and capable of generalization. Regarding this, is known that differences in staining in different samples can make a huge difference: it could be interesting integrating images acquired in different laboratories and make the system resistant to variations by normalizations of images and inhomogeneity corrections. Moreover, in order to improve performances, patches belonging to tissues not of interest (such as blood vessels) should be integrated in the network.

Further study is required both at the clinical side, analysing the results with clinicians and operators for understanding more deeply the necessities, and at the implementation side, such as optimizing processing time for example with a GPU implementation.

## **References**:

[1] Chan JK (2014). The wonderful colors of the hematoxylin-eosin stain in diagnostic surgical pathology” Int J Surg Pathol. 22 (1): 12–32

[2] Pegah Khosravi, Ehsan Kazemi. Deep Convolutional Neural Networks Enable Discrimination of Heterogeneous Digital Pathology Images, EBioMedicine <https://doi.org/10.1016/j.ebiom.2017.12.026>

[3] Anant Madabhushi, George Lee. Image analysis and machine learning in digital pathology: Challenges and opportunities

[4] Metin N. Gurcan, Laura E. Boucheron, Ali Can, Ananat Madabhushi, Nasir M. Rajpoot, Bulent Yener. Histopathological Image Analysis: A Review.

[5] Andrew Janowczyk, Anant Madabhushi. Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases.

[6] Yarin Gal, Zoubin Ghahramani. Dropout as a Bayesin Approximation: Representing Model Uncertainty in Deep Learning, International Converence On Machine Learning, 2016.

[7] Harshita Sharmaa, Norman Zerbe, Iris Klempert, Olaf Hellwich, Peter Hufnagl. Deep convolutional neural networks for automatic classiﬁcation of gastric carcinoma using whole slide images in digital histopathology. Computerized Medical Imaging and Graphics, Volume 61, November 2017, Pages 2-13.

[8] Yongchan Kwon, Joong-Ho Won, Beom Joon Kim, Myunghee Cho Paik. Uncertainty quantiﬁcation using Bayesian neural networks in classiﬁcation: Application to ischemic stroke lesions segmentation. Computer Science, 2018

[9] Yoshua Bengio. Practical Recommendations for Gradient-Based Training of Deep Architectures. arXiv:1206.5533v2 [cs.LG] 16 Sep 2012

[10] Jason Brownlee. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Machinelearningmastery, <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>, Jul 2017