Histopathological cancer detection

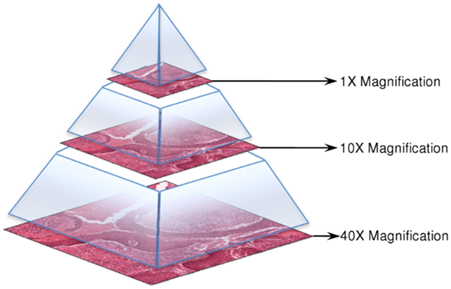
Detecting cancer in a patch of a histological slide, using a convolutional neural network.

# Milestone 1: Baseline Model

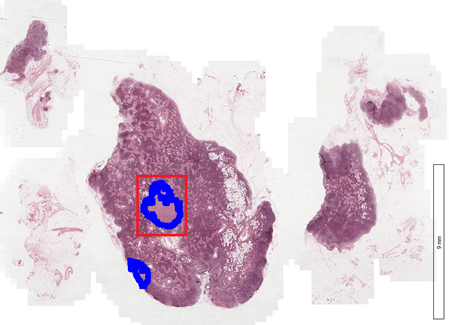
## 1. The analysis of data and construction of features

Our dataset is a derivative of the PatchCamelyon (PCam) dataset, which is a derivative of the Cancer Metastases in Lymph Nodes Challenge 2016 (CAMELYON16) dataset.[1]

The CAMELYON16 dataset consists of 170 training and 129 testing, hematoxylin- and eosin–stained (H&E) whole image slides (WSI). Hematoxylin binds to DNA, colouring it purple. Eosin binds to amino acids found in cytoplasm, collagen and muscle fibers, colouring them pink. Therefore, both the nucleus of each cell, containing Hematoxylin bound DNA, and the surrounding cell structures arevisualized. All 400 slides were obtained from Radboud University Medical Center (RUMC) and the Medical Center Utrecht (UMCU), which scanned the images with a 20x (0.243 µ/pixel) and 40x (0.226 µ/pixel) objective lens respectively. These WSI were annotated by students, which were supervised and checked by pathologist. [2,3]



*Img. 1 Example of effect of magnification. [7]*

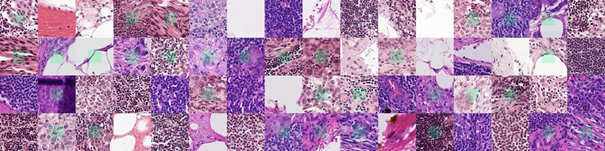
**

*Img. 2 Example of WSI. Tumour tissue has been annotated in blue. [7]*

The PCam dataset consists of 327.680 96x96xp patches selected from the WSI in CAMELYON16, at 10x magnification (0.972 µ/pixel). To filter out all the patches containing the background and no tissue, all patches were converted from RGB to HSV and blurred. Patches with a maximum pixel saturation of below 0.07 were dropped, since these patches were believed not to contain any tissue.

Afterwards, patches were sampled for the dataset by iterating through the WSI and choosing a negative patch with probability around 0.5. Because it’s a probability, some patches could be selected twice and the dataset will contain duplicates. Some patches are rejected based on a CNN and a stochastic hard-negative mining scheme. Hard-negative mining is a technique in which training examples that a model has a hard time classifying correctly are extracted from the larger dataset, since they might contain features that the model can leverage to learn more effectively. [5,6] In this case specifically, a CNN was trained using an annotated dataset (the patches) and false positives with a prediction around 0.5, meaning they are hard to predict (e.g. lie close to the decision boundary), were kept. [3,4] This technique results in a dataset containing negative examples that are more difficult to distinguish from positive examples, compared to the average negative example.

A patch was labelled positive, if there was at least one pixel of tumour tissue in the 32x32 centre of the patch. In the PCam dataset, the entire 96x96 patch was included. However, the tissue outside of the 32x32 center of the patch was not taken into account when labelling the samples. Therefore, a 96x96 sample containing tumor tissue only outside the centre did not receive a positive label. This border was merely added to ensure the data could be used by models that don’t use zero-padding on their images.[3]

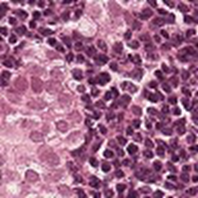
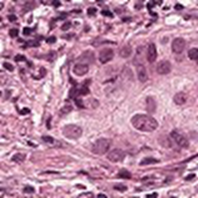
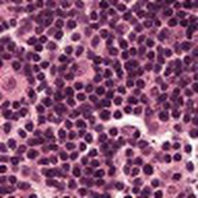


*Img. 3 Example images from the PCam dataset. Centres of patches have been coloured green.*

In the Histopathologic Cancer Detection Kaggle competition, only the 32x32 centres of the original 96x96 PCam patches were included as training and testing data. These centres are the image data that our model will be trained and tested on.

There is no overlap between training and testing data. All the duplicates are taken out, leaving 277.483 unique 32x32px image samples. The Kaggle competition has split this dataset into 220.025 (79%) training and 57.458 (21%) testing images. The testing data is not labelled, since it is used to judge performance in the Kaggle competition.[1] Both the training and testing data contained around 50% positive examples (patches containing tumorous tissue) and 50% negative examples.

We have split the labelled training dataset further, resulting in a training set (70%, 154018 samples) and validation set (30%, 66008 samples), which we used to evaluate the training of our model.



*Img. 4 Examples from our dataset. Left: negative training example. Middle: positive training example. Right: testing example.*

## 2. The inputs and structure of the model

Because the input data are images, a convolutional neural network was used as a model. Our baseline model

The input data is presented as tiff images of 32 x 32 pixels with 3 color channels. The collection of tiff images was converted to a four-dimensional Numpy array.

The dimensions of this Numpy array are as follows:

* Dimension 0: The different images
* Dimension 1: The columns of the images
* Dimension 2: The rows of the images
* Dimension 3: The three colour channels

The bare minimum for a CNN is 1 convolution followed by pooling, a flattening layer, a hidden layer and an output layer. If no pooling layers were added, there would be too much information and the model would not generalise well to other data. Features in a CNN are learned by passing patches of the image through filters with learned weights. The filters that operate on the second convolutional layer extract features that are combinations of lower level features. Features learned through one convolution might be too vague. We added a second convolution to have higher-level features, which might result in a higher accuracy for our initial model.

There are no set rules for determining the initial number of filters. Large amounts of filters pose no problem for NN’s, so we chose a number that was large, but could still be further expanded. We also didn’t know how many features we could distinguish in an image. Conventionally the amount of filters is a power of two.

The output layer needs a different activation than the hidden layer.

The resulting network architecture of the baseline model is as follows:

Convolution and maxpooling:

* Convolutional layer 1 (32 filters)
  + Applies convolutional filters resulting in 32 feature maps.
* Maxpooling layer 1
  + Applies maxpooling to feature maps with 2x2 patch
* Convolutional layer 2 (64 filters)
  + Applies convolutional filters resulting in 64 feature maps.
* Maxpooling layer 2
  + Applies maxpooling to feature maps with 2x2 patch

Dense layers:

* Flatten layer
  + Flattens the feature maps of the final convolutional layers into one vector which serves as input to the densely connected subsequent layer.
* Dense (hidden) layer 1 (64 nodes)
  + Densely connected hidden layer.
* Output layer
  + Two output nodes, corresponding to the two possible classes (cancer/no cancer).

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same', input\_shape=(32, 32, 3)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(2, activation='softmax'))

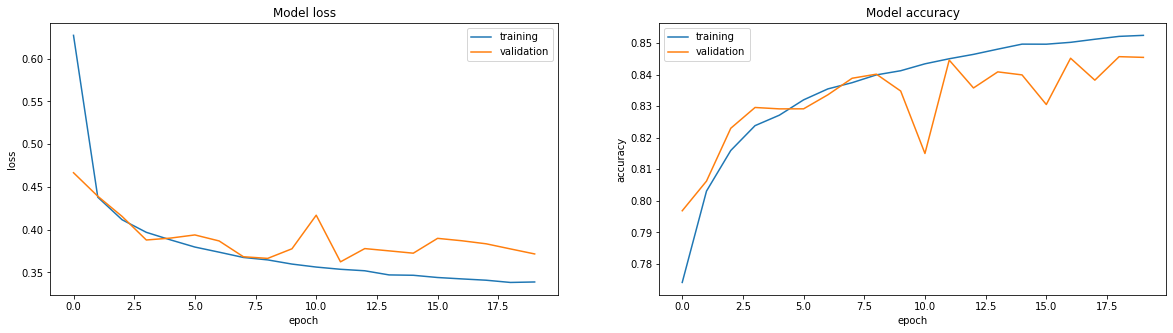
train\_and\_evaluate(model, train\_images, train\_labels, val\_images, val\_labels)

## 3. The training methodology and results

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

For the baseline model, we used categorical cross entropy as loss function and Adam as optimizer for gradient descent, since Adam seems to be a very efficient optimizer in many situations. The learning rate for the Adam optimizer was set to the default value of 0.001. We trained the CNN for 20 epochs.

## 4. Comparison to previous models and analysis of results



*Graph 1: Model loss and accuracy for the baseline model.*

Our baseline model performed surprisingly well. The final validation accuracy was 84.5%, which is far above the 50% chance level for a binary classification problem. While the training accuracy of the model seems to increase monotonically, the validation accuracy fluctuates quite a lot. There are many possible causes for a widely varying validation accuracy, such as overfitting or an unrepresentative validation dataset.

For future models, we have outlined several tweaks which could speed up the learning process and increase the generalizability of our model to the validation data.

Possible tweaks:

* Changing learning rate
  + The learning rate of the adam optimizer is currently set to the default value of 0.001. Increasing the learning rate could perhaps speed up the training process.
* Increasing number of epochs
  + As the graph above shows, the training accuracy had not quite reached a plateau after 20 epochs. Therefore, this model might benefit from using more epochs in the training process. Furthermore, this is the most ‘basic’ model that we have designed. In later, more complex, iterations of our model, the model might require even more epochs to fully train.
* Normalizing input
  + To borrow some abstract terms from a theory video by Andrew Ng, normalizing the input features might aid the learning process since it might make the ‘landscape’ of our cost function more symmetrical.
* Deepening the model
  + Deepening the model by adding more layers increases the freedom of the model in designing a function that corresponds to the underlying distribution of our image data. For instance, adding more convolutional layers to the model might increase the ability of the model to learn relevant features in the training images that are predictive of the presence of cancerous cells.
* Adding dropout layers
  + That being said, increasing the complexity of the model also increases the model’s capacity for overfitting the training data. Our relatively simple baseline model already seems to be overfitting slightly, since the training and validation accuracies seem to diverge after around 10 epochs. This might indicate that the model continues to change in a way that fits the training data more and more efficiently, but does not generalize well to the validation data. Therefore, the model might benefit from the regularizing effect of adding several dropout layers. The dropout layers might force the model not to rely too much on specific weights.
* Adding batch normalization layers
  + Our model uses ReLU activations after each of the convolutional layers and the dense layer. Therefore, normalizing the input (mean around 0 and standard deviation around 1) for each of these layers might speed up the learning process, as it might help the model leverage the non-linearity of the ReLU function near its origin. Batch normalization might also have a slight regularizing effect that our model could reduce overfitting in our model.
* Data augmentations
  + Cancer cells can come in many shapes, sizes and orientations. Therefore, using data augmentations which for instance zoom, rotate, flip, shift or shear the training images might result in a meaningful increase in the number of training examples our model can use to learn from. All of the abovementioned augmentations could result in new images of cancerous cells that are still biologically realistic.
* Feature extraction
  + Cancerous cells have defining visual features that can be determined by us humans. It might be possible to translate these visual features into filters that our neural network can use to learn more quickly and accurately.
* K-fold cross validation

Notes for 1st feedback session:

* Notebook crashes and empties RAM every time the model finishes training. To make the model run again after this, the entire notebook needs to be restarted and run from the start. Why does this happen and is there a way to fix this?
* Validation accuracy seems to fluctuate a lot between epochs. What could be the source of these fluctuations and how can we limit them?
* How to exclude .DS\_store and others from github?

References:

1. <https://www.kaggle.com/c/histopathologic-cancer-detection/overview>

2. Ehteshami Bejnordi et al. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. JAMA: The Journal of the American Medical Association, 318(22), 2199–2210. [doi:jama.2017.14585](https://doi.org/10.1001/jama.2017.14585)

3. https://github.com/basveeling/pcam

4. B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, M. Welling. "Rotation Equivariant CNNs for Digital Pathology". [arXiv:1806.03962](http://arxiv.org/abs/1806.03962)

5. <https://stats.stackexchange.com/questions/294349/hard-mining-hard-examples-does-hard-mean-anything-specific-in-stat>[SP2]

6.  [https://sci-hub.se/https://www.sciencedirect.com/science/article/abs/pii/S0925231219316984](https://sci-hub.se/https:/www.sciencedirect.com/science/article/abs/pii/S0925231219316984)

7. <https://camelyon16.grand-challenge.org/Data/>

Convolutional Neural Networks (CNN) generally provide the best results for image prediction, this is why this type of model is chosen. A disadvantage of this model is it’s lack of interpretability. The features the model learns are hard to define and can only be selected manually, by specifying the weights for a filter by hand.

\* citation: <https://github.com/basveeling/pcam>

# Milestone 2: Actual model

## 1. The analysis of data and construction of features

At this scale of the image data, a pathologist discerns between healthy and cancerous cells by looking at the following features:

* cell nucleus size
* shape of the nucleus
* (relative) size of cell cytoplasm to nucleus (*cytonuclear ratio)*
* edge of nucleus membrane
* the presence of nucleolus

Cancer cells have a deregulated proliferation system and are constantly multiplying. This means DNA is constantly active, unraveled and spread out in the nucleus. This appears as a larger and lighter nucleus. The cytoplasm of cancer cells is larger than that of lymphocytes. Nucleoli are more often visible in cancer cells.

The last feature might not be possible to discern at this image resolution, but the others could be potential features the model could be learning.

At a larger image scale, a pathologist also looks at the relation between the different cell’s positions and the structures they form. This will not be possible for our model, because it is trained on images with insufficient scope.