
Eötvös Loránd University

Faculty of Informatics

Department of Programming Languages and Compilers

# Histopathologic Cancer Detection: Identifying metastatic tissue in histopathologic slides using Deep Neural Networks

*Supervisor:*
Kitlei Róbert
assistant lecturer, MSc

*Author:*
Bauer Marko
Computer Science BSc

*Budapest, 2020*


**EÖTVÖS LORÁND UNIVERSITY**

**FACULTY OF INFORMATICS**

# THESIS TOPIC DECLARATION

Student:

Name: **Marko Bauer** Code: **R7GNA0**

Type: **full-time student** Course: **Computer Science BSc**

Supervisor:

Name:

Affiliation with address: ELTE Faculty of Informatics
(department name goes here)
1117 Budapest, Pázmány Péter sétány 1/C.

Status and qualification:

Title of the Thesis work: Histopathologic Cancer Detection: Identifying metastatic tissue in histopathologic slides using Deep Neural Networks

Topic of the Thesis work:

**Overview:** Goal of this Thesis Work is creating an application capable of classifying histopathologic slides of lymph node sections and determining whether the patient has metastatic tissue (metastatic cancer) or not. Application would allow user to load a histopathologic slide, and as a result receive a category to which it belongs (whether it is malignant or benign). Classification would be accomplished using a class of deep neural networks called convolutional neural networks. Plan is to build multiple models trained on three different datasets, using transfer learning (VGG-19 Neural Network), as well as constructing models which are trained from scratch.

**Application Development Semester Project:** Thesis work is an extension of semester project from subject Application Development. That project is an application for landscape classification, trained on one small dataset (Intel Image Classification dataset), with single neural network and simple graphical user interface.

**Datasets**: Networks will be trained on three different datasets:

1. Breast Cancer Histopathological Dataset (BreakHis) – composed of 9,109 microscopic images of breast tumor tissue collected from 82 patients using different magnifying factors, containing 2,480 benign and 5,429 malignant samples.
2. PatchCamelyon (PCam) Dataset – consists of 327,680 color images extracted from histopathologic scans of lymph node sections, where each image is annotated with a binary label indicating presence of metastatic tissue.
3. NCT-CRC-HE-100K Dataset - contains 100,000 non-overlapping image patches from hematoxylin & eosin (H&E) stained histological images of human colorectal cancer (CRC) and normal tissue.

**Plan:** Thesis will be divided into four major parts:

1. Assembling datasets, data preprocessing (loading data, removing noise, normalization, whitening) and data augmentation
2. Building the networks (both the ones which have to be trained from scratch and the ones which use transfer learning), and training them on data
3. Improving neural network prediction accuracy (reducing overfitting with hyperparameter tuning, regularization and batch normalization)
4. Creating graphical user interface which allows user to load histopathologic slide and select network which has be applied on it, and to get as output category to which that slide belongs, along with image which superimposes heatmap on the original one

Budapest, 2019.11.25.

# Contents


**1 Introduction** **3**

1.1 Medical Image Analysis . . . . . 3

1.2 Histopathologic Cancer Detection . . . . . 4

1.3 Datasets . . . . . 4

1.3.1 BreakHis Dataset . . . . . 4

1.3.2 NCT-CRC-HE-100K Dataset . . . . . 5

1.4 Approach . . . . . 6

**2 User Documentation** **7**

2.1 General Software Requirements . . . . . 7

2.2 Additional Software Requirements . . . . . 8

2.3 Runnig the Program . . . . . 9

2.4 Inspecting Datasets . . . . . 9

2.5 Inspecting Networks . . . . . 10

2.6 Basic Use: Predicting Tissue Type . . . . . 11

2.7 Advanced Use: Visualizing Network Representations . . . . . 11

2.7.1 Visualizing Heatmaps of Class Activations . . . . . 12

2.7.2 Visualizing Intermediate Activations . . . . . 12

2.7.3 Visualizing Filters of Convolutional Layers . . . . . 13

2.8 Saving Results . . . . . 14

**3 Developer Documentation** **15**

3.1 Use-Case Diagram . . . . . 16

3.2 Class Diagrams . . . . . 16

3.3 Creation of Datasets . . . . . 18

3.4 Convolutional Neural Networks . . . . . 19

3.4.1 Convolution Layer . . . . . 19
3.4.2 Max-Pooling Layer . . . . . 19
3.4.3 Flatten Layer . . . . . 20
3.4.4 Fully-Connected Layer . . . . . 20
3.4.5 Dropout Layer . . . . . 20
3.4.6 CNNSimple Implementation . . . . . 21
3.4.7 Transfer Learning . . . . . 21
3.4.8 VGG19 Simple Implementation . . . . . 21
3.5 Experiments and Results . . . . . 21
3.6 Graphical User Interface . . . . . 21
3.7 Implementing Additional Features . . . . . 21

**4 Conclusion** **22**
4.1 Future Work . . . . . 22

**Bibliograhpy** **23**

# Chapter 1

# Introduction

Over the past decade deep learning has been reaching unimaginable heights, possibly starting another industrial revolution. Numerous problems were solved with extraordinary accuracies, surpassing previous solutions, as well as human performance. Self-driving cars and automated transportation, digital personal assistants, fraud detection, natural language processing are just some of the tasks whose solutions use deep learning algorithms. Arguably the biggest impact was made in the field of computer vision, with image classification, object detection and segmentation, image colorization, reconstruction, super-resolution and synthesis. As a result, the entire healthcare industry is on a verge of a major transformation. Deep learning algorithms could be used in various ways, such as analyzing electronic health records, detecting heart problems, diagnosing cancer, planning, navigating and performing surgery, and medical image analysis.

## 1.1 Medical Image Analysis

Majority of the data present in medicine (over 90% of it) belongs to the imaging data, and it is natural to assume that deep learning could have huge impact in the future of healthcare. Computed tomography (CT), magnetic resonance imaging (MRI) and X-rays are just some of the medical imaging techniques which produce data that can be used in deep learning algorithms. In this paper I will focus on analysis of histopathology slides, microscopic images of tissue obtained during biopsy.

## 1.2 Histopathologic Cancer Detection

Histopathologic cancer detection refers to the problem of detecting cancerous tumors in pathologic scans of lymph nodes, organs of the lymphatic system, (widely present throughout the body), which act as filters for foreign particles and cancer cells. This task is clinically-relevant, as pathologists analyze a great number of histopathologic slides on a daily basis, which is time-consuming and tedious task prone to misinterpretation. Hence a lucrative solution would reduce their workload, speed up the process, and improve the prediction accuracy. In this paper, I propose a deep learning algorithm to solve this task.

## 1.3 Datasets

Digital pathology is field in medical imaging, where whole-slide scanners are used to digitize glass slides containing tissue at high resolution, in order to be viewed, managed, shared and analyzed on a computer. The advance of digital pathology has led to the high availability of digital images, which has in turn led to creation of datasets on which deep learning methods can be applied. In this paper, I will train neural networks on two such datasets, BreakHis and NCT-CRC-HE-100K.

### 1.3.1 BreakHis Dataset

The Breast Cancer Histopathological Image Classification (BreakHis) is a dataset constructed in collaboration with Pathological Anatomy and Cytopathology Laboratory (Parana, Brazil) which contains 9.109 images of breast tumor tissue. It is collected from 82 patients, using different magnifying factors (40x, 100x, 200x, and 400x), and contains 2.480 benign and 5.429 malignant samples (700x460 pixels). Dataset is divided into two main groups, benign tumors (do not invade nearby tissue or spread to other parts of the body) and malignant tumors (can invade nearby tissues, made of cancer cells). Based on the way cells look under the microscope, there are following subtypes of tumors: adenosis, fibroadenoma, phyllodes tumors and tubular adenona for benign tumors, and carcinoma, lobular carcinoma, mucinous carcinoma and papillary carcinoma for malignant tumors (Figure 1.1).

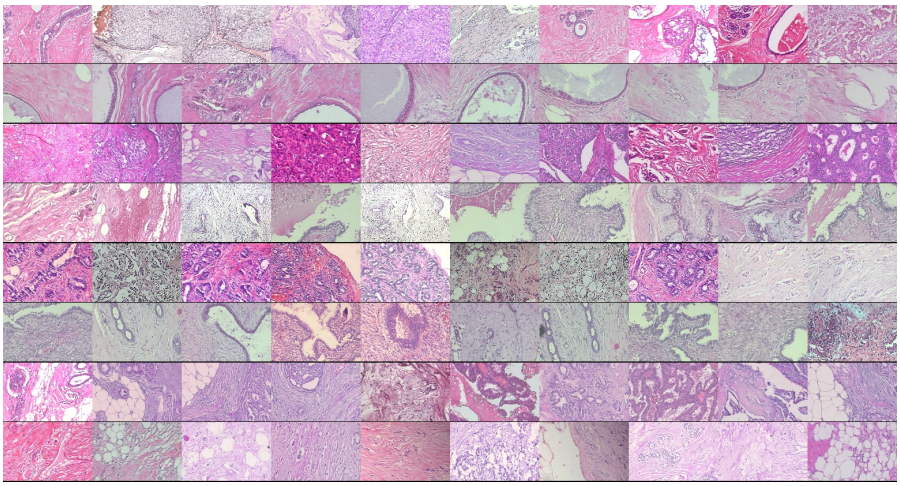

Figure 1.1: BreakHis dataset: adenosis, ductal carcinoma, fibroadenoma, lobular carcinoma, mucinous carcinoma, papillary carcinoma, phyllodes tumor, tubular adenoma

### 1.3.2 NCT-CRC-HE-100K Dataset

The NCT-CRC-HE-100K is a dataset constructed in collaboration with National Center for Tumor diseases (Heidelberg, Germany) which contains 100.000 images of colorectal tumor tissue. It is collected from 86 patients, using 100x magnifying factor (224x224 pixels). Based on the way cells look under the microscope, there are following subtypes of tissue: adipose, background, debris, lymphocytes, mucus, smooth muscle, normal colon mucosa, cancer-associated stroma and colorectal adenocarcinoma epithelium (Figure 1.2).

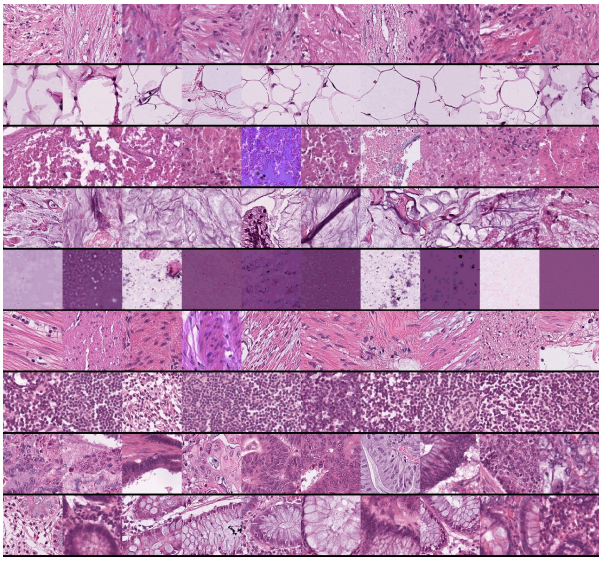

Figure 1.2: NCT-CRC-HE-100K dataset: adipose tissue, background, debris, lymphocytes, mucus, smooth muscle, normal colon mucosa, cancer-associated stroma, colorectal adenocarcinoma epithelium

## 1.4 Approach

I have divided the creation of the program for histopathologic cancer detection in four major parts:

1. Assembling dataset structure (required for input to convolutional neural network), image preprocessing (loading, removing noise, normalization, whitening) and data augmentation (expanding size of dataset by applying a series of random transformations to each image)
2. Building the convolutional neural networks (class of deep neural networks applied to analyzing images), and training them on data
3. Improving prediction accuracy of networks (solving underfitting and overfitting problems) with hyperparameter tuning (choosing optimal parameters for learning algorithm) and changes to network architecture
4. Creating graphical user interface for the program, which allows user to load histopathologic slide and select network which has be applied on it, and to get as output category to which that slide belongs, with additional possibility to visualize network representations (heatmaps of class activations, filters of convolutional layers and intermediate activations)

# Chapter 2

# User Documentation

There are three main ways of using Histopathologic Cancer Detection program:

1. Predicting tissue subtype of breast and colorectal tissue, i.e. predicting whether patient has breast or colorectal cancer or not
2. Visualizing what Convolutional Neural Networks learns, i.e. visualizing how network transforms input image, and which parts of input image lead to predicting tissue and cancer subtype
3. Adding new datasets and networks, in order to increase the scope of the program, and predict even more tissue and cancer subtypes

In order to use the program to predict tissue and cancer subtype, or to visualize what CNNs learn, certain general software requirements are needed, while in order to expand scope of the program, certain additional software requirements are needed.

## 2.1 General Software Requirements

Before running the program, Python3 programming language is required, along with following dependencies:

- h5py - interface to the HDF5 binary data format, which can store huge amounts of numerical data, and easily manipulate that data from NumPy (used to save and load network weights)

- Keras, Keras-Applications, Keras-Preprocessing, tensorflow - neural networks APIs, which can build and deploy machine learning applications (used to build and train neural networks)
- matplotlib, seaborn - data visualization libraries (used for visualization of datasets, neural networks and tissue and cancer subtype predictions)
- NumPy - library which provides support for large, multi-dimensional arrays
- opencv-python, Pillow, scikit-image - image processing libraries (used to load, process and save images)
- pandas - data analysis and manipulation library (used for loading datasets)
- PyQt5 - python binding of cross-platform GUI toolkit Qt, which contains substantial set of GUI widgets (used for implementing GUI part of the program)
- scikit-learn - machine learning library for predictive data analysis (used for its metrics in order to assess network performance)

If general software requirements are satisfied, program can be used to predict tissue and cancer subtypes, as well as to visualize network representations.

## 2.2 Additional Software Requirements

In order to replicate results of already existing networks, or to expand scope of the program by adding new datasets and training new networks, NVIDIA graphics card is required, along with the following:

- NVIDIA CUDA toolkit - provides a development environment for creating high performance GPU-accelerated applications
- NVIDIA CUDA Deep Neural Network Library (cuDNN) - GPU-accelerated library of primitives for deep neural networks, which provides highly tuned implementations for standard routines
- tensorflow-gpu - GPU-enabled version of tensorflow library

Training convolutional neural networks requires large amount of computations, and in order to decrease training time, networks are trained on GPU (it is not feasible on CPU, as GPU training is multiple times faster).

## 2.3 Runnig the Program

After running the program, new window appears (Figure 2.1).

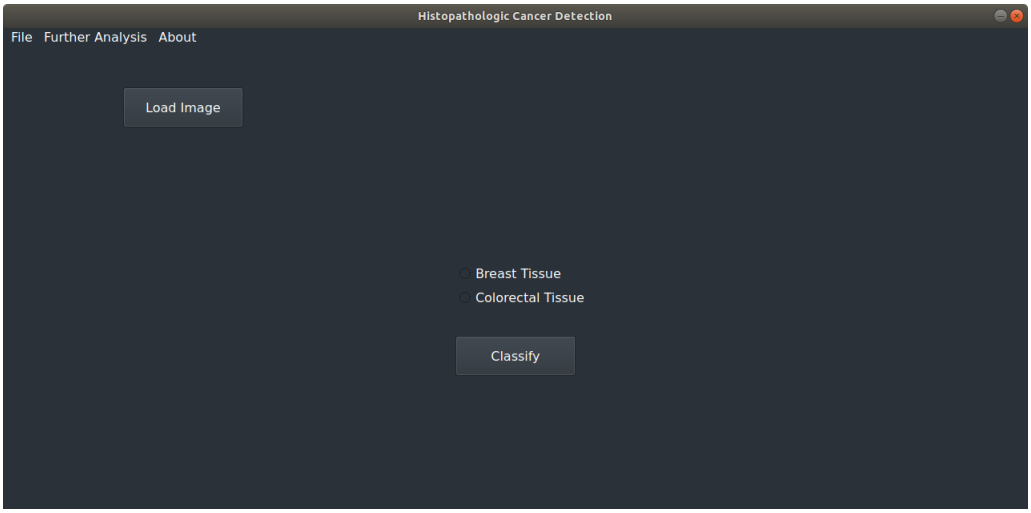


Figure 2.1: Main window of Histopathologic Cancer Detction program

From here, it is possible to:

1. Inspect datasets and networks
2. Load histopathologic slide of breast or colorectal tissue and to predict the tissue and cancer subtype
3. Further visualize network representations by analyzing heatmaps of class activations, filters of convolutional layers and intermediate activations

## 2.4 Inspecting Datasets

In order to find basic information about datasets, which include sample images from dataset, tissue and cancer subtypes, as well as number of images per subtypes, go to *About* → *About Datasets* and choose dataset (Figure 2.3).

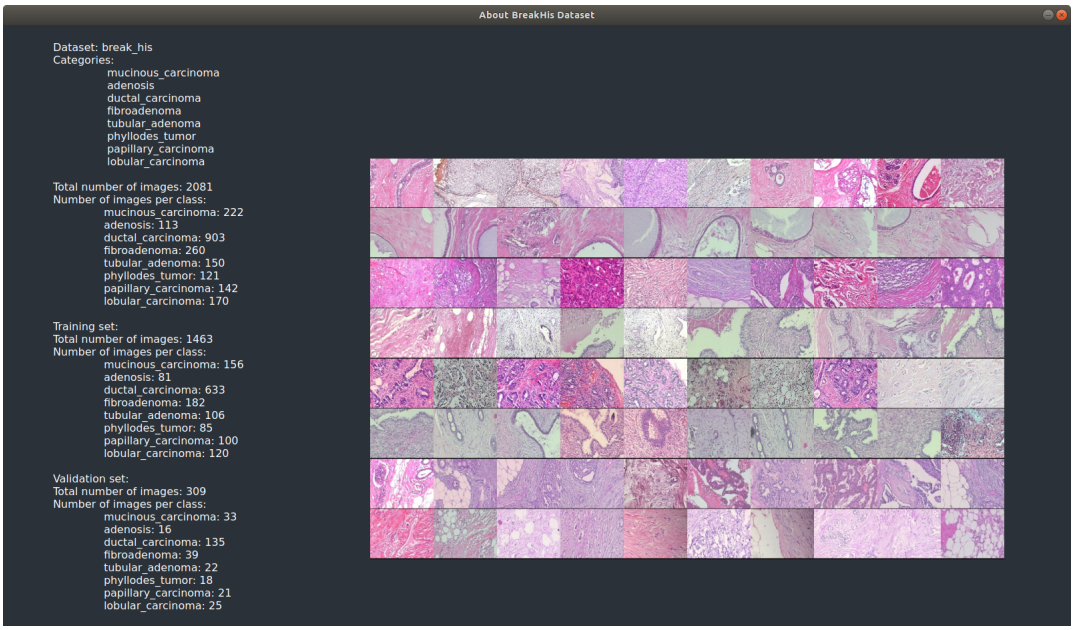


Figure 2.2: BreakHis Dataset basic information

## 2.5 Inspecting Networks

In order to find basic information about networks, which include network architecture and network performance on trainin dataset (accuracy, loss and confusion matrix) go to *About* → *About Models* and choose network. (Figure 2.3).

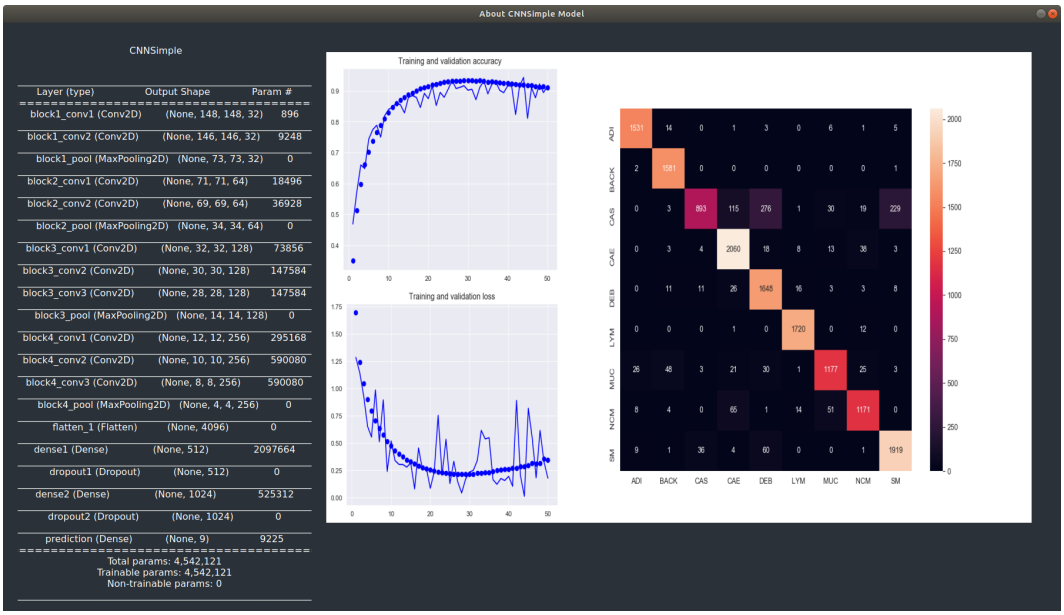


Figure 2.3: CNNSimple Network basic information

## 2.6 Basic Use: Predicting Tissue Type

Main use of the Histopathologic Cancer Detection program is to make it easy and fast to load histopathologic slide and get a prediction on tissue and cancer subtype. In order to load a histopathologic slide, click on *Load Image*, and select histopathologic slide for which the prediction is needed. Next, select a tissue type of the slide by clicking on *Breast Tissue* or *Colorectal Tissue*. Afterwards, in order to get tissue and cancer subtype, along with network probabilities, press *Classify* button (Figure 2.4).

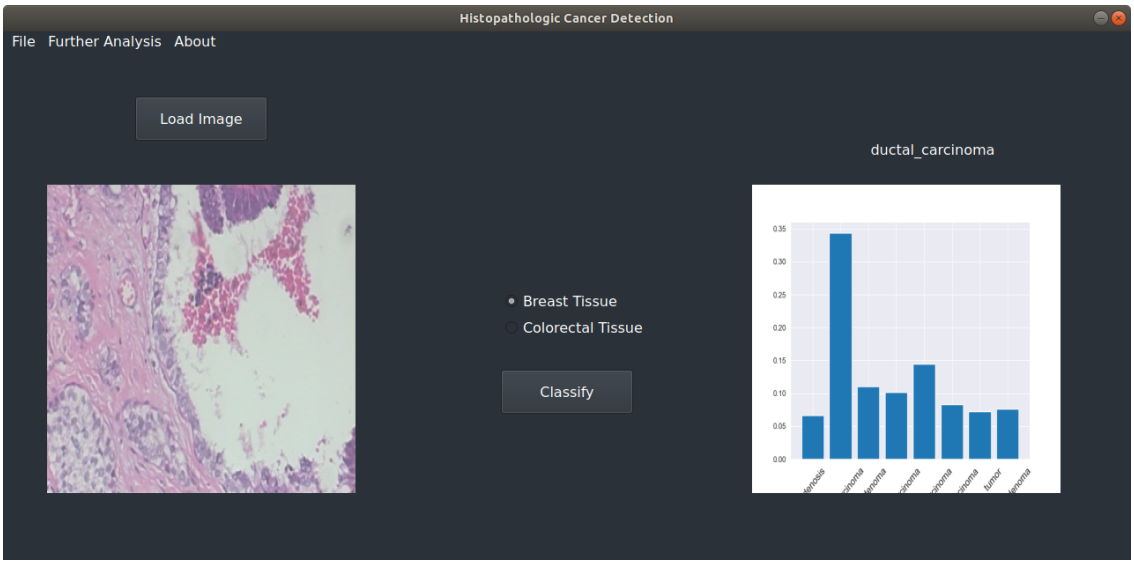


Figure 2.4: Prediction of input histopathologic slide of breast tissue to be ductal carcinoma

## 2.7 Advanced Use: Visualizing Network Representations

Deep neural networks are highly complex models which have great expressive power and can achieve high accuracy while solving wide range of problems. Unfortunately, with high complexity comes low interpretability, which presents issue, especially in deep learning applications to health care. Fortunately, there are several methods to inspect convolutional neural networks, and interpret their output. In this program, I covered three of them: visualizing heatmaps of class activations, visualizing filters of convolutional layers and visualizing intermediate activations.

### 2.7.1 Visualizing Heatmaps of Class Activations

Different parts of an image have different weights in networks decision on tissue and cancer subtype classification, and because of that, we can highlight which parts of an image have the highest influence on the networks output. This class of methods is called class activation map visualization, and it produces heatmaps of class activations over input images. A class activation heatmap is a 2D grid of scores associated with a specific output class, computed for every location in any input image, indicating how important each location is with respect to the class under consideration. In order to produce heatmap, go to *Further Analysis* → *Heatmap* (Figure 2.6).

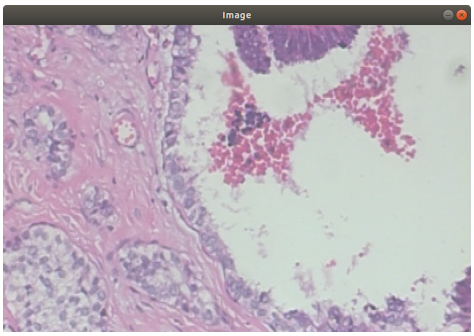


Figure 2.5: Input image of breast tissue


Figure 2.6: Heatmap of class activations on input image

### 2.7.2 Visualizing Intermediate Activations

Visualizing intermediate activations consists of displaying the feature maps that are output by various convolution and pooling layers in a network, given a certain input (the output of a layer is often called its activation, the output of the activation function). This gives a view into how an input is decomposed into the different filters learned by the network. In order to visualize intermediate activations of layers, go to *Further Analysis* → *Class Activations*. Next choose which network layer's intermediate activations should be visualized, along with channel number, or 'all', for visualizing all of layer's channels (Figure 2.7).

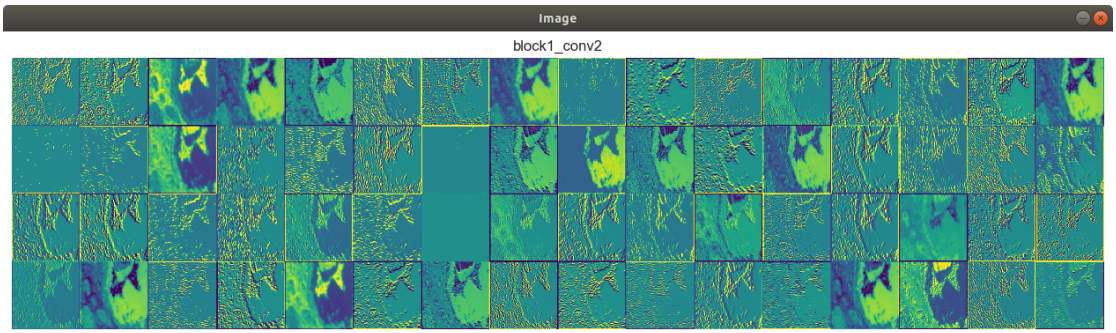


Figure 2.7: Every channel of second convolutional layer of VGG19Simple on a breast tissue input (Figure 2.5).

### 2.7.3 Visualizing Filters of Convolutional Layers

Further analysis is not only dependent on an input image, as we can also inspect feature exractors (filters) of convolutional layers by displaying patterns on which each filter is supposed to respond to. This can be done with gradient ascent in input space : applying gradient descent to the value of the input image of a convnet so as to maximize the response of a specific filter, starting from a blank input image. The resulting input image will be one that the chosen filter is maximally responsive to. In order to visualize filters of convolutional layers, go to *Further Analysis* → *Network Filters*. Next choose which network convolutional layer's filters should be visualized, along with filter number, or 'all', for visualizing all of layer's filters (Figure 2.8).

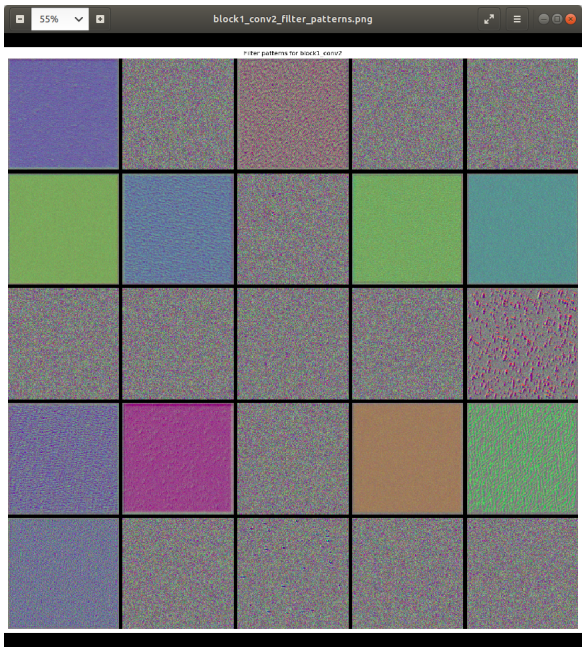


Figure 2.8: Filter patterns for second convolutional layer of CNNSimple network

## 2.8 Saving Results

After all predictions have been made, and all further analysis has been conducted, in order to save the results, i.e. save all the images created during the program's execution, go to *File* → *Save*.

# Chapter 3

# Developer Documentation

Histopathologic Cancer Detection program is divided into four major segments (Figure 3.1):

1. Data - includes dataset creation, analysis and visualization
2. Networks - includes creation, training, testing of convolutional neural networks
3. Experiments - includes hyperparameter tuning, network performance assessment and visualization
4. Graphical User Interface - includes creation of all application windows and their interconnection

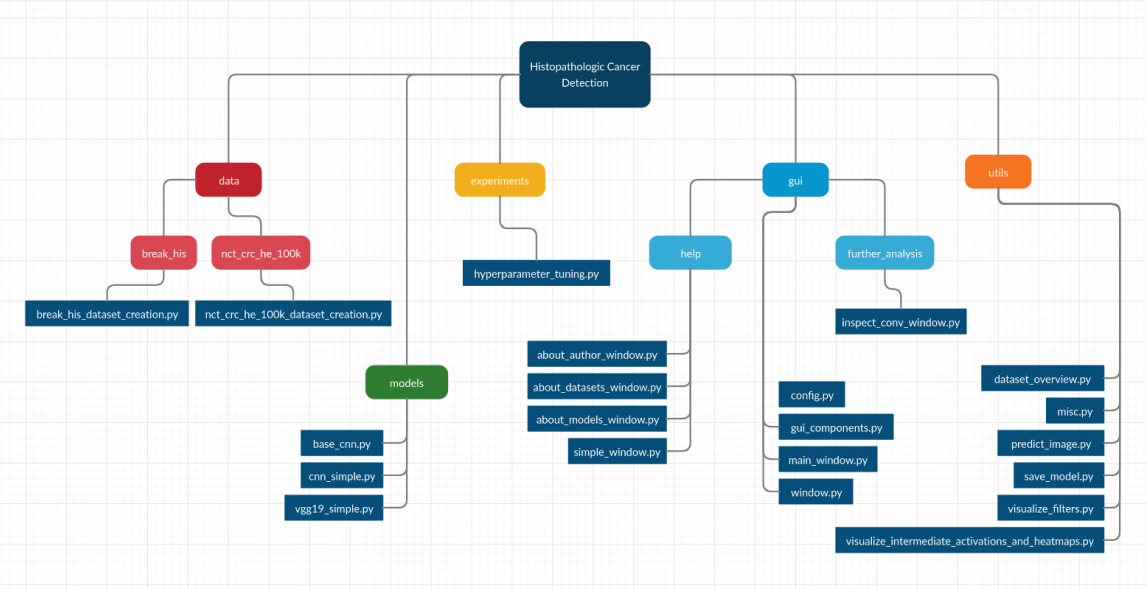


Figure 3.1: Diagram of directories and scripts of Histopathologic Cancer Detection

## 3.1 Use-Case Diagram

One of the main goals of the Histopahtologic Cancer Detection program was the ease of use, i.e. straightforward graphical user interface which makes complex operations look quite simple and effortless. Even though there are extremely advanced algorithms with millions of parameters behind the program, GUI was made in such a way that everyone can use it.

First step is loading the image and selecting tissue type (breast or colorectal tissue), after which classification is being done. At every step of the way, current work can be saved, and new image can be loaded to start the process from scratch. After the classification, it is possible to visualize network representations and perform further analysis of the results by visualizing layer activations, network filters and heatmap (Figure 3.2).

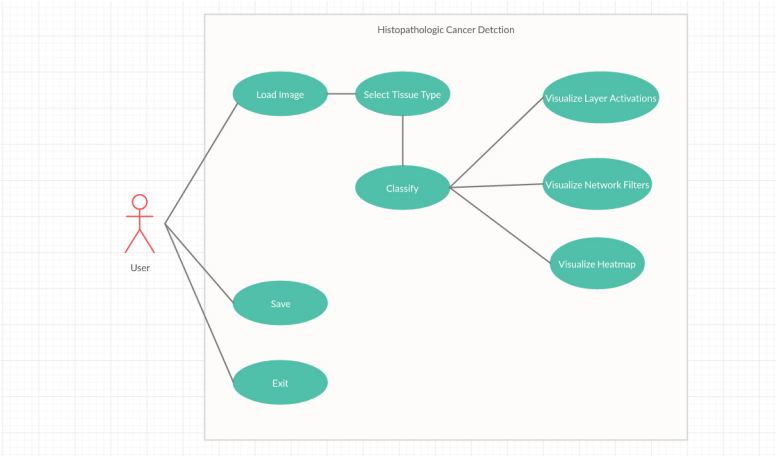


Figure 3.2: Use-Case Diagram of Histopathologic Cancer Detection

## 3.2 Class Diagrams

Classes of Histopathologic Cancer Detection can be divided into two main components: neural network classes and window classes.

Window class is the base class of all window classes, and it implements common methods, such as setting up window size and central widget. On the other side, Main Window class is the central point of GUI, as it is window which opens when program is run, and every other window is invoked from it (Figure 3.3). Window classes, along with their attributed and methods, will be discussed in detail in Section 3.6.

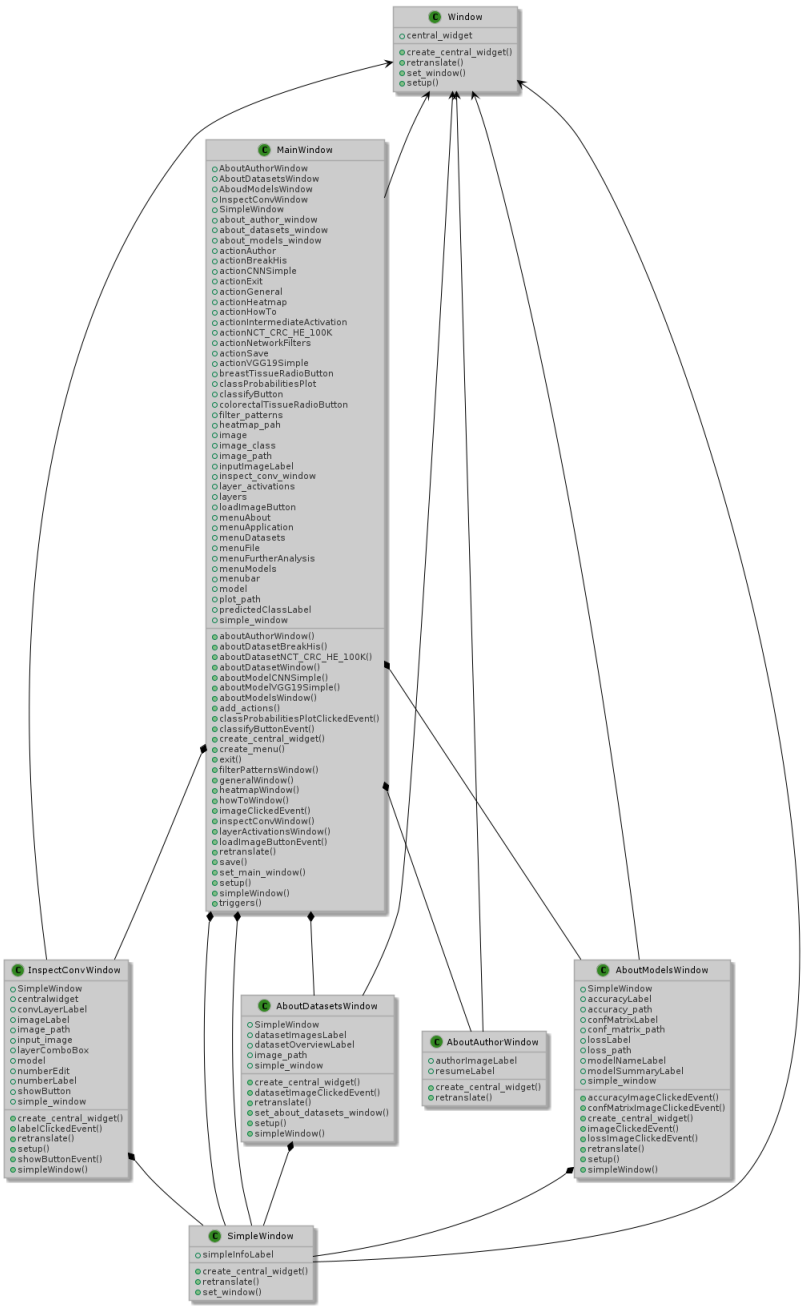


Figure 3.3: Class Diagram of Window classes of Histopathologic Cancer Detection

BaseCNN class is the common class of all neural network classes, and it contains common attributes, such as dataset name, network name, compile parameters, and common methods, such as creation of data generators, compilation and training of the network. Neural network classes will be discussed in more detail in Section 3.4

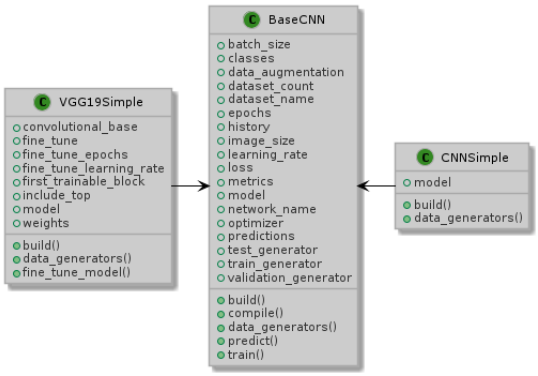


Figure 3.4: Class Diagram of Neural Network classes of Histopathologic Cancer Detection

## 3.3 Creation of Datasets

Performance and accuracy of convolutional neural networks relies largely on datasets, i.e. on quality of available data, dataset size, class balance, etc. But before feeding data to the network, when using Keras API, certain dataset structure must be satisfied. More precisely, dataset must have following structure: train, validation and test directories, each with identical subdirectories for each class. Scripts responsible for creation of required directory structure are:

- ***break_his_dataset_creation.py***,
- ***nct_crc_he_100k_dataset_creation.py***.

They work by extracting datasets downloaded from [1], [2], creating necessary directory tree and distributing images between created subdirectories. After executing scripts, datasets are ready to be fed into convolutional neural networks in order to train them, but before that, neural network architecture is to be built.

## 3.4 Convolutional Neural Networks

Convolutional Neural Networks for image classification take image as an input, **process** it, and output category to which that image belongs. Processing part consists of a series of layers through which we propagate image in order to learn features, which in turn determine to which class an image belongs. (Figure 3.5).

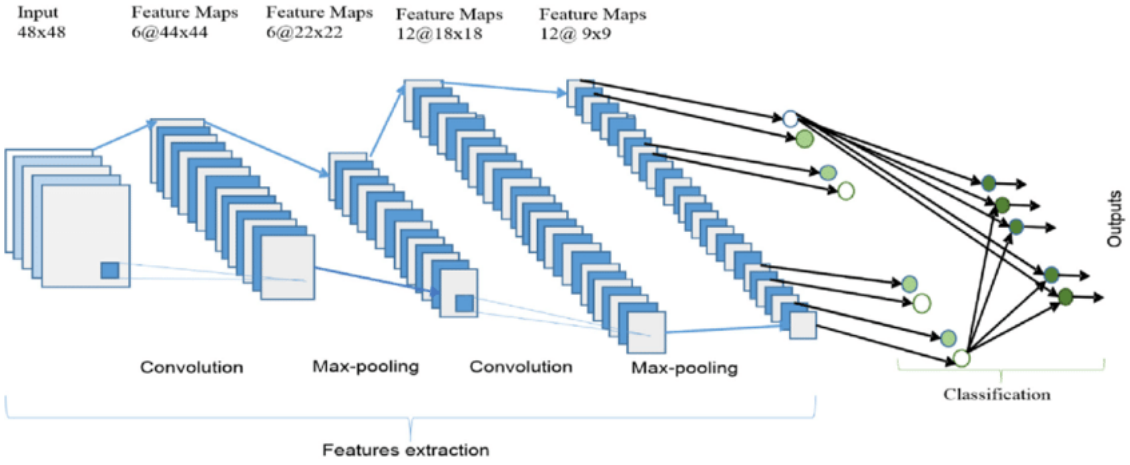


Figure 3.5: Convolutional Neural Network consisting of two Convolution layers, two Max-Pooling layers, Flatten layer and two Fully-Connected (Dense) layers

Most commonly used layers in CNN architectures are convolution layer, max-pooling layer, flatten layer, dense layer and dropout layer.

### 3.4.1 Convolution Layer

Convolution layer is the building block of the CNN architecture. Its primary purpose is to extract features from input image, such as edges, lines, curves, colors. As we go deeper inside the network, it starts identifying more complex features, such as shapes, objects. This layer consists of multiple filters (usually 3x3 matrices) whose parameters need to be learned, and it performs dot products between parts of an image and these filters.

### 3.4.2 Max-Pooling Layer

Max-Pooling layer is located after a series of convolution layers in CNN architecture. It is a downsampling method which reduces dimensionality, thus decreasing

number of parameters and computational power needed in order to train the network, while retaining important features and patterns. It is achieved by applying a max filter to non-overlapping subregions (usually 2x2 matrices), thus reducing the size of each feature map by a factor of 2, and discarding 75% of acivations in the process.

### 3.4.3 Flatten Layer

Output of the convolutional base of the network (series of convolution and max-pooling layers) is a two-dimensional matrix, and before feeding that data to the classification top of the network, it needs to be transformed. Flatten layer reshapes the output matrix to vector, thus removing all dimensions but one in the process, making the data prepared for the series of fully-connected layers.

### 3.4.4 Fully-Connected Layer

After the high-level features of the image have been detected, series of fully-connected (dense) layers is attached to the top of the network in order to classify image into a label. Dense layers consist of huge number of nodes (neurons), which provide a way of learning non-linear combinations of features outputted by convolutional base, and determine which feature most correlate to a particular class.

### 3.4.5 Dropout Layer

Fully-Connected layer contains the most parameters in the network, and as a result neurons develop co-dependency amongst each other during training, which leads to over-fitting the data (not generalizing well on new, unseen images). In order to prevent that, dropout layers are positioned right after dense layers in CNN architecture as a means of regularizing the network. Dropout consists of randomly ignoring (dropping out) fraction of neurons of fully-connected layer, which in turn makes network learn more robust features, and achieve better performance.

### 3.4.6 CNNSimple Implementation

Text.

### 3.4.7 Transfer Learning

Text.

### 3.4.8 VGG19 Simple Implementation

Text.

## 3.5 Experiments and Results

Text.

## 3.6 Graphical User Interface

Text.

## 3.7 Implementing Additional Features

Text.

# Chapter 4

# Conclusion

Conclusions.

## 4.1 Future Work

# Bibliograhpy

[1] *BreakHis Kaggle Dataset*. URL: https://www.kaggle.com/ambarish/breakhis.

[2] *NCT-CRC-HE-100K Dataset*. URL: https://zenodo.org/record/1214456#.XpClRPlS8Uo.