

PRAXIS DER SOFTWARENTWICKLUNG

SPECIFICATIONSBOOK

NEURAL NETWORK BASED IMAGE CLASSIFICATION SYSTEM ON HETEROGENEOUS PLATFORMS

from

Team 2

Dimitrov, Drehwald, Guneshka, Häring, Stangel

Inhaltsverzeichnis

1	Introduction	3
2	Goal	3
3	Product use	3
4	Criteria	4
4.1	Must Acceptance criteria	4
4.2	Can Acceptance criteria	5
4.3	Criteria of demarcation	6
5	Product environment	6
6	Product data	7
7	Functional Requirements Must	8
8	Functional Requirements Can	11
9	Non-functional requirements	13
10	Test cases	14
11	System models	20
11.1	Scenarios	20
11.1.1	Scenario 1	20
11.1.2	Scenario 2	20
11.1.3	Scenario 3	20
11.1.4	Scenario 4	21
11.1.5	Scenario 5	21
11.1.6	Scenario 6	21
11.1.7	Scenario 7	21
11.1.8	Scenario 8	22
11.2	Usecases	23
11.2.1	Image classification page	23
11.2.2	Training page	24
11.2.3	Image detection page	25
11.2.4	Show topology page	26
11.3	GUI	27
12	Stage responsibilities	30
13	Quality requirements	30

1 Introduction

In today's world of globalisation and digitalisation, to keep up with the rapidly growing economy, one important challenge is the automatisisation of tasks. One aspect of this is the classification of visual inputs. Whether it is to check for broken parts in production or surveillance of public places. With the rapidly growing power of computers, neural networks are becoming more popular for tasks like this, as they need a lot of computational power but deliver sufficiently accurate results.

In the following project we want to build a framework with an intuitive graphical user interface to achieve these kinds of tasks.

To speed up the process of classification the software will be able to use different hardware that is more efficient for specific calculations. To further adjust the neural network to its task it should have different operating modes to function on. High performance, low power consumption and a high energy efficiency mode.

2 Goal

The goal of this project is to create a software which performs sufficiently accurate image classification and is able to switch between deployment platforms and operating modes. The software will be able to predict the power consumption and the performance (bandwidth, FLOPs).

The software should also have a GUI to interact with the program and to visualise the results.

The software should be extendable for further tasks.

3 Product use

The target group are engineers with a basic knowledge of data science.

The software is used to classify images on different deployment platforms with different operating modes using a pretrained neural network.

Additionally, the software can be extended to be used for image detection, classification of frames from a videostream and training of a neural network.

4 Criteria

4.1 Must Acceptance criteria

- | | |
|---------------|--|
| MAC010 | Image classification
The software can take a single image as input and returns the possibilities for each predefined image class. The prediction is based on a pretrained neural network. |
| MAC020 | Running neural network on heterogeneous platforms
The software is able to communicate with CPU and FPGA. The software is able to offload calculations to the different deployment platforms and receive the results. |
| MAC030 | Different operating modes
The software has three operating modes. One mode for high performance, one for low power consumption and one for high energy efficiency. |
| MAC040 | Performance and power consumption prediction
The software can predict the performance with a certain power consumption and also the power consumption for a certain performance. |
| MAC050 | GUI for interacting with software
The user is able to access the entire functionality described in MAC10-MAC40 just by using the GUI. No coding or command line usage is required. |

4.2 Can Acceptance criteria

CAC070	Illustration of the topology of a neural network The software is able to visualise the topology of a given neural network (see figure 9).
CAC071	The visualised neural network can be saved The visualized neural network is saved as a .png file in a chosen directory.
CAC080	Object Detection The software can detect the bounding box of an object.
CAC090	Using different models The user is able to use different pretrained neural networks before an image classification process.
CAC100	Training neural networks The software allows the user to train a neural network based on an predefined architecture. Neural networks trained by the user will be executed the same way neural networks provided with the software are executed.
CAC110	Voting of multiple neural networks The user is able to choose multiple neural networks for classification. The software will then execute all selected neural networks sequentially. The result presented to the user will be based on the weighted results of the different neural networks.
CAC120	Using video for classification The software is able to take a video, divide it into a certain amount of frames and perform image classification for each frame. The classified frames are shown and can be iterated by the user.
CAC130	Using camera for classification as input The software takes the current frame from the camera connected to the host pc, classifies it, displays the results and then when ready, takes the next available frame.
CAC140	Running neural network on GPU MAC20 is extended by GPU.

4.3 Criteria of demarcation

- | | |
|-------------|--|
| D010 | No low-level optimization
Optimisations to reduce the execution time of object classification and detection will be carried out in OpenCL.
No optimizations including low-level languages or assembly intrinsics will be implemented. |
| D020 | No real time requirements
The software doesn't have to react in realtime.
Code optimizations will be done in OpenCL to reduce the running time of the network per image classification/ detection task. |
| D030 | No neural network size optimization
No techniques for memory usage reduction like pruning or binarisation will be implemented. |
| D040 | No mobile support
The software does not support mobile devices, like smartphones or wearables. |
| D050 | No input from commandline
The software does not support commandline input. The features are only useable with the GUI. |

5 Product environment

The software runs on a computer in the lab at the CDNC institute. It has a CPU and an external FPGA connected via USB. Additionally, there is a GPU.
The operating system is XUbuntu 18.04.

6 Product data

- PD010 Images for classification**
The user can choose images of the format .jpg, .png, .bmp. The images are chosen by the user with the file explorer.
- PD020 Config/weight file of pretrained model**
The config file is a .cfg file.
In the beginning the classes and output nodes are mapped. In the format *number of output node = classname*
Then the hyperparameters are described with the format *<name> = <value>*. Then the layers are described in their order with the following format
[kind of layer]
list of parameters in the format *<name> = <value>*
Then the weights and biases for each layer are listed.
- PD030 Labeled image set for classification training**
The dataset is a directory with images and the image name has the format *<id of image>_<image class>*.
- PD040 Labeled set of images for object detection training**
It is a .txt file in the same directory as the images. The images are labeled with their name. The bounding box for each image are described in the .txt file with the same name as the image, in the format *image class, x, y, width, height*. (X,Y) are the coordinates of the left bottom corner. (X, Y), width and height are relative.
- PD050 Output format of image classification results if saved**
If the output result of the classification is saved this is saved as a .txt file with the name of the image. The format is *<image class name> = <probability>*, one row for each image class.
If multiple images are classified, there are multiple .txt files.
- PD060 Output format of image detection results if saved**
If the output result of the detection is saved this is saved as a .txt file with the name of the image. The format is *<image class name> = <probability>, <X>, <Y>, <width>, <height>*, one row for each image class.
If multiple images are detected, there are multiple .txt files.
- PD070 Video input**
The input video is in a .avi format.

7 Functional Requirements Must

- MFR010 Use neural network for image classification**
Tested with: T010 Implements: MAC10 MAC10
A neural network is used to classify images. The result is a list of probabilities per image class.
- MFR011 Deploy pre-trained neural network with the corresponding layers**
Tested with: T011 Implements: MAC10
A pre-trained neural network is deployed with its layers to a specified platform. The deployed neural network is used for MFR010.
- MFR012 Reading and parsing a neural network configuration/weight file**
Tested with: T012 Implements: MAC10
The software is able to read a configuration file of a neural network and parse it for MFR011.
- MFR020 High performance operating mode**
Tested with: T020 Implements: MAC30
An operating mode to perform calculations as fast as possible.
- MFR021 Low power consumption operating mode**
Tested with: T021 Implements: MAC30
An operating mode to perform calculations with low power consumption.
- MFR022 Have high energy efficiency operating mode**
Tested with: T022 Implements: MAC30 MAC30
An operating mode to perform calculations at an optimal ratio between performance and power consumption.
- MFR023 Calculator for power consumption**
Tested with: Implements: MAC40
The software can calculate the power consumption on a given neural network, operating mode and deployment platform.
- MFR024 Calculator for performance**
Tested with: Implements: MAC40
The software can calculate the performance on a given neural network, operating mode and deployment platform.
- MFR025 Dispatching the calculation process defined from the operating mode**
Tested with: Implements:
The software is able to control the clock rate of the processor according to chosen operating mode.

- MFR030 Support CPU for calculation**
Tested with: T030 Implements: MAC20
The software supports CPU for calculation.
- MFR031 Support FPGA for calculation**
Tested with: T031 Implements: MAC20
The software supports FPGA for calculation.
- MFR040 Send image for classification**
Tested with: T040 Implements: MAC20
The software gives the image as input for the neural network to the chosen deployment platform.
- MFR041 Receive result**
Tested with: T041 Implements: MAC20
The program should be able to receive results of the executed image classification from the deployment platforms.
The software gives the image as input for the nn to the chosen deployment platform.
- MFR050 GUI**
Tested with: T050 Implements: MAC50
The program has a Graphical User Interface to display all functions to the user.
- MFR060 Showing results**
Tested with: T060 Implements: MAC50
After executing the image classification, the results are shown in a bar chart.
- MFR070 Choosing image for classification**
Tested with: T070 Implements: MAC50
The GUI has a button with an on click event which opens a file explorer.
The explorer filters the files that only files of the format .jpg, .png, .bmp are listed.

MFR080 Choosing deployment platform

Tested with: T080 Implements: MAC50

The GUI has a dropdown which lists the devices which are supported. The devices which can be theoretically be accessed but are not connected to the host pc or the communication with them does not work are grayed out and not clickable.

MFR090 Choosing operating mode

Tested with: T090 Implements: MAC50

The GUI has dropdown which lists the different modes (high performance mode, low power consumption mode and high energy efficiency mode). The power consumption in Watts and performance in FLOPs are also stated behind the operating mode names.

8 Functional Requirements Can

- CFR100 Choosing between different neural networks**
Tested with: T100 Implements: CAC090
The GUI has a button which opens the file explorer which filters for .cfg files. There you choose the config file of the neural network which you want to use. The program loads this config and parses it so it can be deployed. Possible models would be GoogLeNet or AlexNet.
- CFR110 Train neural network for classification of imageset**
Tested with: T110 Implements: CAC100
The user chooses a neural network and a new imageset and trains the neural network on this new imageset. If it is pretrained it uses transfer learning with the existing weights otherwise random values.
- CFR111 Saving newly trained neural network**
Tested with: T111 Implements: CAC100
The software is able to take the weights and config of an newly trained neural networks and save it as .cfg file.
- CFR112 Choosing and loading data set**
Tested with: T112 Implements: CAC100
The software has an option to select a set of labeled images and for loading those.
- CFR113 Backpropagation**
Tested with: T113 Implements: CAC100
The software is able to adjust the weights and biases of the neural network in the training process with backpropagation.
- CFR114 Change the learning rate**
Tested with: T114 Implements: CAC100
To adjust the learning process of the neural network the user can change the speed of how fast the weights and biases will be changed.
- CFR115 Fit the output layer to the amount of image classes**
Tested with: Implements: CAC100
If the user trains a neural network with a dataset, the number of output nodes are adapted to the number of image classes.

- CFR120 Visualisation of neural network**
 Tested with: T120 Implements: CA070
 The software is able to visualise the topology of a neural networks (see figure 9)
- CFR121 Saving the visualisation**
 Tested with: Implements: CAC071
 The user can save the visualisation of the topology of a neural network as .png file to a chosen directory.
- CFR130 Object detection**
 Tested with: T130 Implements: CAC080
 The software can detect the position and image class of objects in an image.
- CFR131 Showing detected object**
 Tested with: T131 Implements: CAC080
 The found objects are marked by a bounding box. The bounding box is drawn on the image. This picture is shown.
- CFR140 Choosing and loading video**
 Tested with: T140 Implements: CAC120 CAC120
 The user can choose a video and the software can use it as input for the classification/detection process.
- CFR150 Connect with camera**
 Tested with: T150 Implements: CAC130
 The software can connect with a camera connected to the host pc.
- CFR151 Receive video stream from camera**
 Tested with: T151 Implements: CAC130
 The software can receive a video stream from the camera.
- CFR152 Apply classification/detection for a certain amount of frames**
 Tested with: T152 Implements: CAC130
 The software can divide a video or videostream into frames and is able to apply image classification and detection on those.
- CFR160 Support GPU for calculation**
 Tested with: T160 Implements: CAC140
 To speed up the calculations the program is able to use an additional GPU.
- CFR170 Voting of multiple neural networks**
 Tested with: T170 T170 Implements: CAC110
 The user can choose multiple neural networks. The image classification is done on every neural network separately and the results are weighted and accumulated.

9 Non-functional requirements

NFR010 Project size

The project should have around ten thousand (10,000) lines of code

NFR020 Code size

The project should be done with Object-Orientated programming. The whole project should have around forty (40) to eighty (80) classes excluding interfaces.

NFR030 Model-View-Controller

The project should be based on the design pattern model-view-controller.

NFR040 Programming language

The software is written in C++ and OpenCL.

NFR050 Minimal size of training dataset

The software works with a dataset with a minimum of 100 images.

10 Test cases

T010	Use neural network for image classification
T010.1	State: An image as input, a pretrained neural network, a deployment platform and an operating mode is given. Action: The user clicks on „Start image classification“. Reaction: The image is classified by the neural network and results are shown.
T011	Deploy pre-trained neural network
T011.1	State: The pretrained neural network is loaded and parsed. Action: The user clicks on „Start image classification“. Reaction: The software loads the model to the deployment platform.
T012	Reading and parsing neural network configuration file
T012.1	State: A .cfg file with the configuration of a pretrained neural network is given. Action: The user clicks on „Start image classification“. Reaction: The software loads the model and parses it .
T012.2	State: The file explorer is open Action: The user selects a neural network to import Reaction: The file explorer closes and the neural network is imported and selected for the classification.
T020	High performance operating mode
T020.1	State: An image as input, a pretrained neural network, a deployment platform is given . Action: The user chooses to perform the calculations in high performance operating mode and starts the classification. Reaction: The calculations run considerably faster than in the other possible modes with the same conditions.
T021	Low power consumption operating mode
T021.1	State: An image as input, a pretrained neural network, a deployment platform is given. Action: The user chooses to perform the calculations in low power consumption operating mode and starts the classification. Reaction: The calculations run with considerably lower power consumption than with the other possible modes in the same conditions.

T022	High energy efficiency operating mode
T022.1	<p>State: An image as input, a pretrained neural network, a deployment platform is given.</p> <p>Action: The user chooses to perform the calculations in high energy efficiency operating mode and starts the classification.</p> <p>Reaction: The calculations run with regard to balance between power consumption and speed.</p>
T030	Support CPU for calculation
T030.1	<p>State: An image as input, a pretrained neural network, CPU as deployment platform and an operating mode is given.</p> <p>Action: Click on the button „Start image classification“</p> <p>Reaction: Elephant has the highest probability.</p>
T031	Support FPGA for calculation
T031.1	<p>State: An image as input, a pretrained neural network, FPGA as deployment platform and an operating mode is given.</p> <p>Action: Click on the button „Start image classification“</p> <p>Reaction: Elephant has the highest probability.</p>
T040	Send image for classification
T040.1	<p>State: An image as input, a pretrained neural network, a deployment platform and an operating mode is given.</p> <p>Action: The user starts image classification</p> <p>Reaction: The software sends the image as array to the selected platform.</p>
T041	Receive result
T041.1	<p>State: The software is awaiting result.</p> <p>Action: Platform sends results.</p> <p>Reaction: The software receives the results from the platform and shows it.</p>
T050	GUI
T050.1	<p>State: The user wants to use the software.</p> <p>Action: The user starts the program.</p> <p>Reaction: The users sees the Graphical User Interface showed on Figure 1.</p>

T060	Showing results
T060.1	<p>State: The software awaits result.</p> <p>Action: The deployment platform sends result.</p> <p>Reaction: The Graphical User Interface shows the result in a bar chart as shown in figure 4.</p>
T070	Choosing image for classification
T070.1	<p>State: The user is on the page for image classification.</p> <p>Action: The user clicks on the button „Choose image“.</p> <p>Reaction: The file explorer opens with the filter for .png, .jpg, .bmp.</p>
T070.2	<p>State: The file explorer is open.</p> <p>Action: The user selects an image with a valid format.</p> <p>Reaction: The file explorer closes and image is loaded and shown as preview.</p>
T080	Choosing platform/hardware
T080.1	<p>State: The user is on the page for image classification.</p> <p>Action: The user chooses the desired deployment platform with the dropdown.</p> <p>Reaction: An internal flag is set to the desired platform and the dropdown shows the chosen deployment platform.</p>
T090	Choosing operating mode
T090.1	<p>State: The user is on the page for image classification.</p> <p>Action: The user chooses the desired operating mode with the dropdown.</p> <p>Reaction: An internal flag is set to the desired operating mode and the dropdown shows the chosen operating mode</p>
T100	Choosing between different neural network
T100.1	<p>State: The user is on the page for image classification.</p> <p>Action: The user clicks on the button „Choose neural network“.</p> <p>Reaction: The file explorer opens.</p>
T100.2	<p>State: The file explorer is open.</p> <p>Action: The user selects a config file.</p> <p>Reaction: The file explorer closes and the software loads the input and parses it. If it is loaded there is a success message shown.</p>

T110	Train neural network for classification of imageset
T110.1	<p>State: The user is on the main page.</p> <p>Action: The user clicks the button „Train a neural network“.</p> <p>Reaction: The user is redirected to a new page for training, shown in figure 6.</p>
T110.2	<p>State: The user is on the page for training, has selected a neural network, a dataset for training, the kind of training (backpropagation or transfer learning if possible), the learning rate and the desired precision.</p> <p>Action: The user clicks on the button „Train“</p> <p>Reaction: The software starts to train the selected neural network and shows the progress in a line graph.</p>
T110.3	<p>State: The training is in process.</p> <p>Action: The precision reaches the desired precision.</p> <p>Reaction: The training stops.</p>
T111	Saving a neural network after training
T111.1	<p>State: The training finishes.</p> <p>Action: No action required.</p> <p>Reaction: The software stores the trained neural network in the directory of the selected .cfg file as a .cfg file.</p>
T112	Choosing and reading dataset
112.1	<p>State: The user is on the training page.</p> <p>Action: The user clicks on „Choose dataset“.</p> <p>Reaction: A file explorer opens.</p>
T112.2	<p>State: The file explorer is open.</p> <p>Action: The user chooses the folder with the images.</p> <p>Reaction: The program automatically iterates over all images and reads the given data that can be used for training.</p>
T113	Backpropagation
T113.1	<p>State: The user is on the training page, a dataset, a neural network, learning rate and desired precision are given.</p> <p>Action: The user clicks on „Train“.</p> <p>Reaction: The software adjusts the weights and biases of the corresponding neural network via backpropagation to improve its precision. These changes are then shown with a diagram.</p>
T114	Changing parameters
T114.1	<p>State: The user chose a neural network, the dataset and the desired precision.</p> <p>Action: The user changes the learning rate to a smaller number and starts training.</p> <p>Reaction: The neural network adjusts its weights but with smaller significance of one image.</p>

T120	Showing topology of a neural network
T120.1	<p>State: The user is on the main page.</p> <p>Action: The user clicks the „Show topology of a neural network“button.</p> <p>Reaction: The user is redirected to a new page for showing a topology.</p>
T120.2	<p>State: The user is on the page for showing the topology.</p> <p>Action: The user clicks on „Choose topology to show“</p> <p>Reaction: The file explore opens</p>
T120.3	<p>State: The file explorer is open.</p> <p>Action: The user choses a .cfg file.</p> <p>Reaction: The file explorer closes and the topology is shown as in figure 9.</p>
T130	Object detection
T130.1	<p>State: The detection window is open. An image as input, a pretrained neural network, a deployment platform and an operating mode is given.</p> <p>Action: The user clicks on the button „Start detection“</p> <p>Reaction: The network is run for inferencing and the network output is shown to the user.</p>
T131	Drawing bounding box
T131.1	<p>State: Inferencing was executed on an image given by the user, the choosen neural network predicted bounding boxes.</p> <p>Action: No action required</p> <p>Reaction: The original image, given by the user, is overlayed with the boxes predicted by the network, the updated image is presented to the user.</p>
T140	Choosing video
T140.1	<p>State: The software is running. A pretrained neural network, a deployment platform and an operating mode is given.</p> <p>Action: The user selects a .avi video file.</p> <p>Reaction: The system stores the path to the selected video and is available to process images from this video sequentially.</p>

T150	Connect with camera
T150.1	<p>State: The software is running.</p> <p>Action: The user connects a usb camera to the host.</p> <p>Reaction: The system dynamically detects the camera and allows the user to select the camera as an image source</p>
T150.2	<p>State: A usb camera is connected to the host. The software is not running.</p> <p>Action: The user starts the software.</p> <p>Reaction: The system dynamically detects the camera and allows the user to select the camera as an image source</p>
T151	Receive video stream from camera
T151.1	<p>State: The software is running, a camera is available as image source.</p> <p>Action: The user chooses the camera as image source.</p> <p>Reaction: The first camera image is provided as a preview, the continuous image stream is available for further processing.</p>
T152	Apply classification for a certain amount of frames
T152.1	<p>State: The software is running. A video source was chosen by the user. All network details were provided by the user. Classification was chosen by the user.</p> <p>Action: The user clicks on the button „start classification“</p> <p>Reaction: The system processes the video file imagewise</p>
T160	Support GPU for classification
	<p>State: The classification window is open. An image as input, a pretrained neural network, a deployment platform and an operating mode is given.</p> <p>Action: The user chooses GPU as a deployment platform. The user clicks on the button „Start image classification“</p> <p>Reaction: image classification is performed.</p>

11 System models

11.1 Scenarios

11.1.1 Scenario 1

The user U1 wants to classify the image of a cat. He goes on the classification page and he clicks on the dropdown and sees the three operating modes „low power consumption“, „high performance“and „high energy efficiency“. He can also see the predicted power consumption and performance. He chooses to classificate in the low power mode and runs the programm. The results are shown.

11.1.2 Scenario 2

The user U2 goes to the classification page and chooses the image of a coala and the high power performance mode and CPU mode. The software states that it would take 86 watts with 166 GFLOPs. U2 decides he would rather use the high energy efficiency mode with 140 GFLOPs and 70 watts. He sets the other parameters and clicks on Start image classification. The result is that the image is a coala and shows this result.

11.1.3 Scenario 3

The user U3 created the blueprint for a new neural network in .cfg. She wants to train a network based on this config file but computation time is shared and expensive. Therefore U3 has to convince her boss. U3 uses the software with her neural network as input and selects the visualisation toolkit. U3 saves the output and uses it during the discussion to demonstrate the advantages of her new neural network.

11.1.4 Scenario 4

User U4 has to categorise a large dataset of plants from a biology field trip. U4 has two trained neural networks for this task. The first with a good accuracy and high confidence on leaves. The second with a high confidence and accuracy on flowers. On unknown objects they both tend to have a low confidence. U4 does not want to manually decide which network to use for every image. He also does not want to train a new neural network. Therefore U4 selects both networks and the folder with the new images inside, as well as the parameters save-result and dont-show results. The software classifies all images in a few minutes and he is able to handover the dataset for further documentation.

11.1.5 Scenario 5

User U5 has heard about this software and wants to test it. U5 is a pokemon fan, therefore he decides to use a new neural network to classify the newest generation pokemon. None of the provided networks was trained for that task, so U5 decides to train a new neural network. U5 copies an existing neural network layout file and adds five (5) fully connected layers in between to create a larger neural network. U5 uses his large pokemon image dataset, his new neural network layout file and the software, to train a new neural network. Afterwards U5 creates a folder with new pokemon images and uses his new network and the software to classify them.

11.1.6 Scenario 6

U6 had a trip in Africa and made a lot of pictures of animals. He looks for an easy way to know how many different species of animals he saw and took photos of. Alex doesn't know how to code or to run a program thus he needs a friendly and understandable Graphical User Interface, that our software offers. Alex opens the main menu of the software where he sees that it's possible to finish his task, without any knowledge, because of the GUI.

11.1.7 Scenario 7

U7, a company, wants to develop an AI to feed the animals at Zoos. The company does not have enough labelers to label all of the frames they need to teach the software which animal it is seeing at the moment. U7 decides to use the software for object detection. An employee goes on the Detection page of the software and uses it to label the frames required for the AI.

11.1.8 Scenario 8

The company U8 wants to teach small kids parallel to read, recognize percents and animals. The software is just right for the job, because of the image classification option of the software. The CEO of U8 hears about Tucs and now wants to test it. He assigns a few employees with their kids to try the software. The results are outstanding! Because of the intuitive layout and the structure of the image classification page of the software, the kids are able to learn and also having fun at the same time.

11.2 Usecases

11.2.1 Image classification page

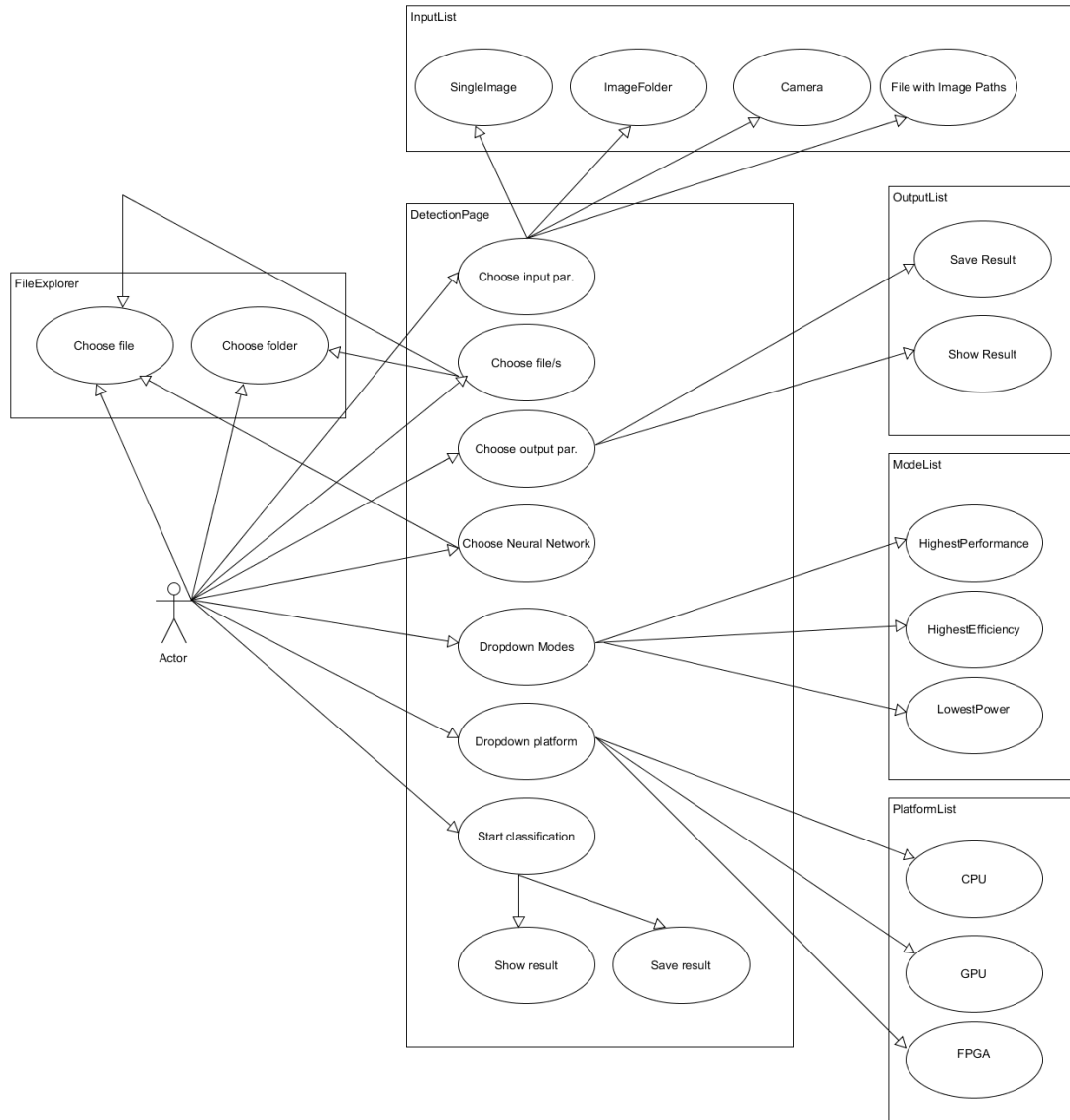


Figure 1: Usecase of the image classification page

11.2.2 Training page

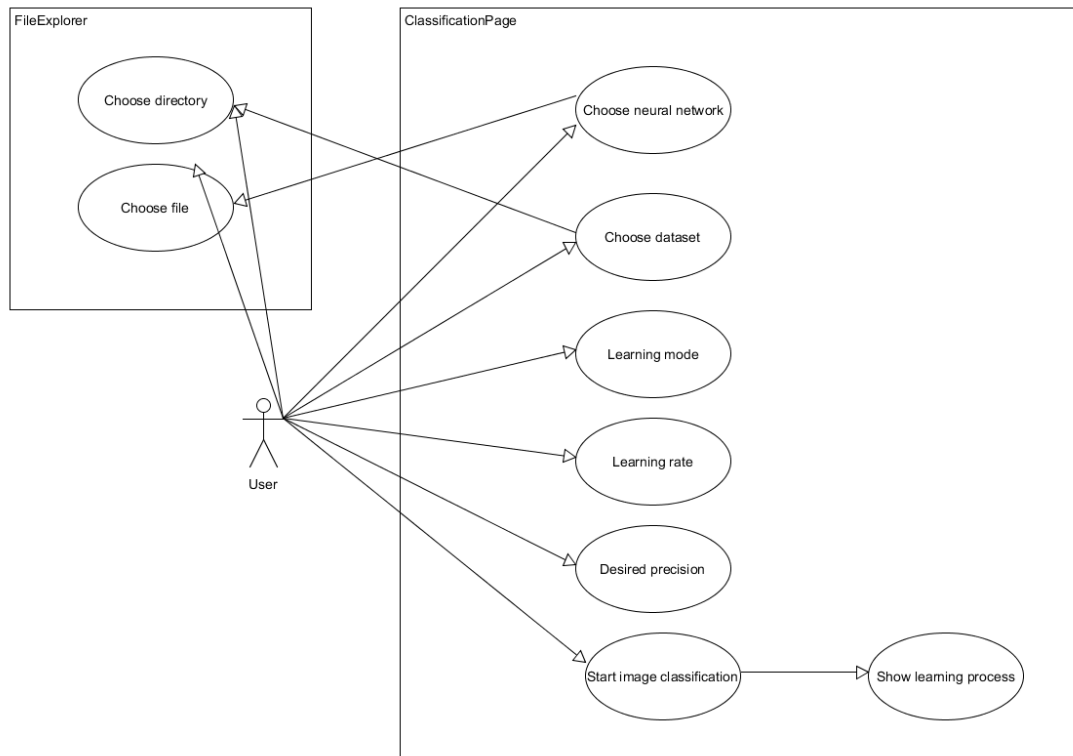


Figure 2: Usecase of the trainingspage

11.2.3 Image detection page

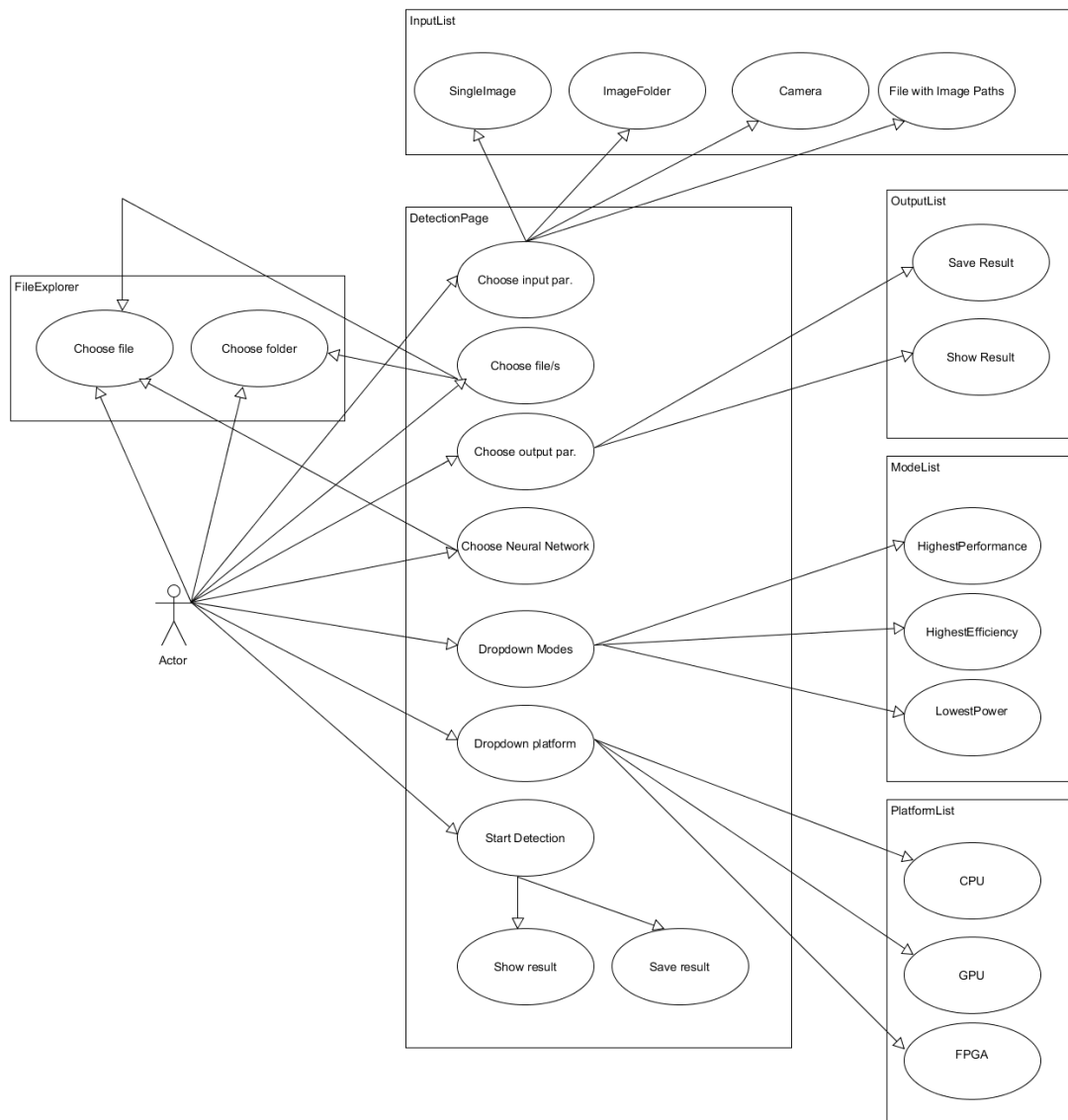


Figure 3: Usecase of the image detection page

11.2.4 Show topology page

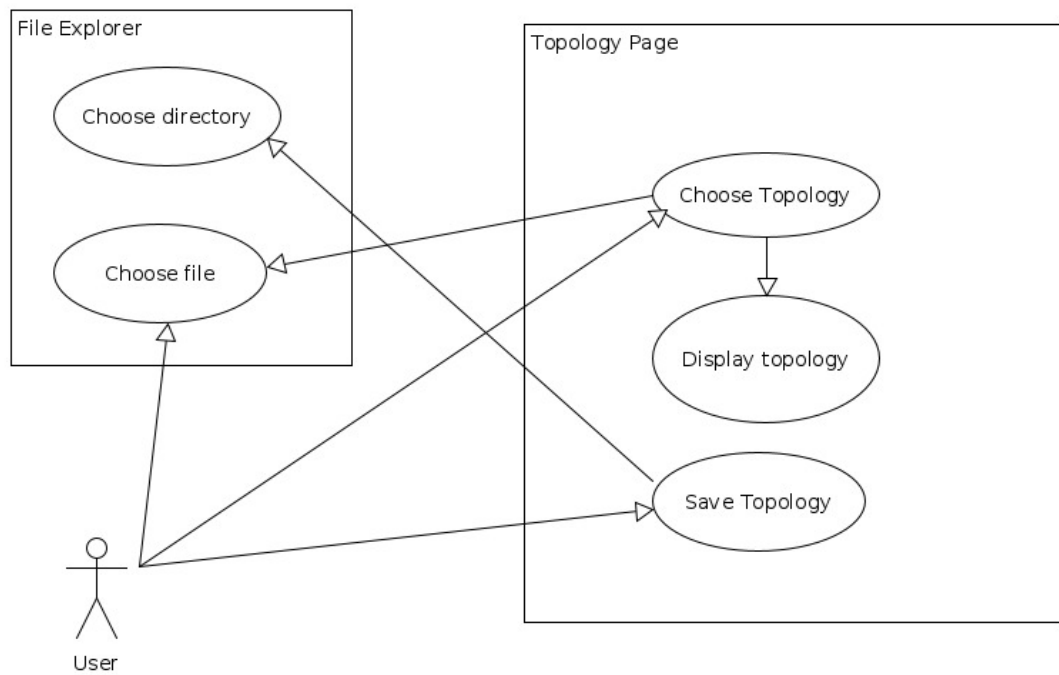


Figure 4: Usecase of the image classification page

11.3 GUI

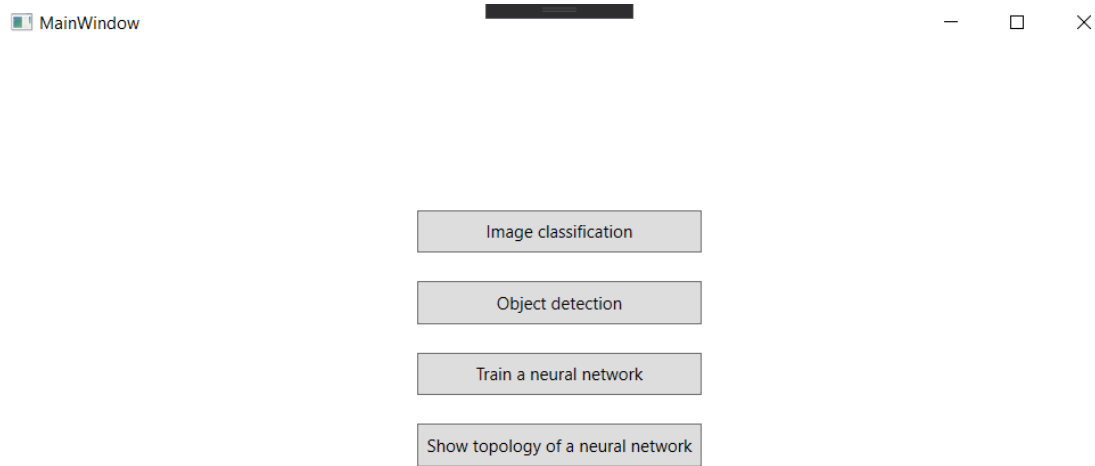


Figure 5: Main page of our software

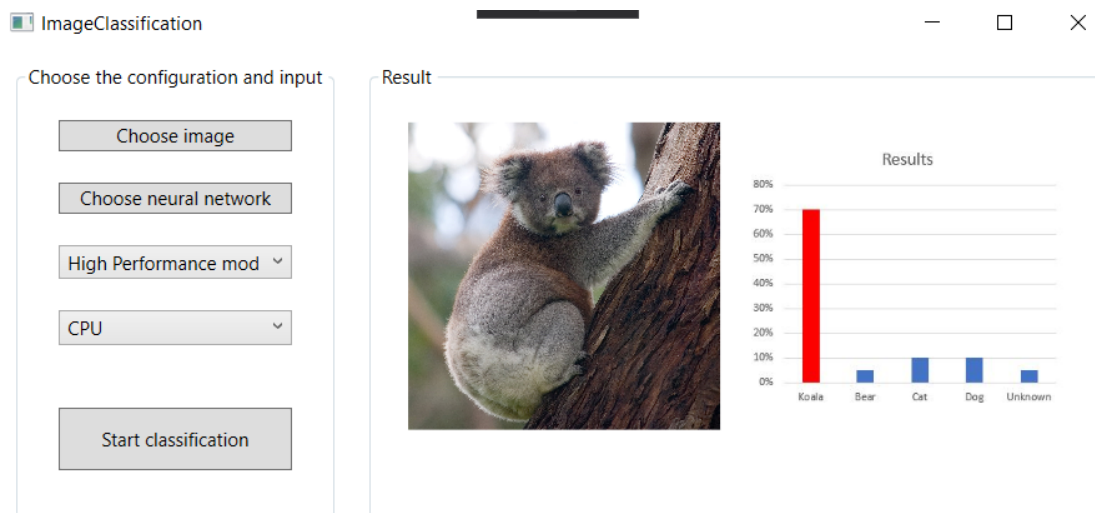


Figure 6: image classification page of our software

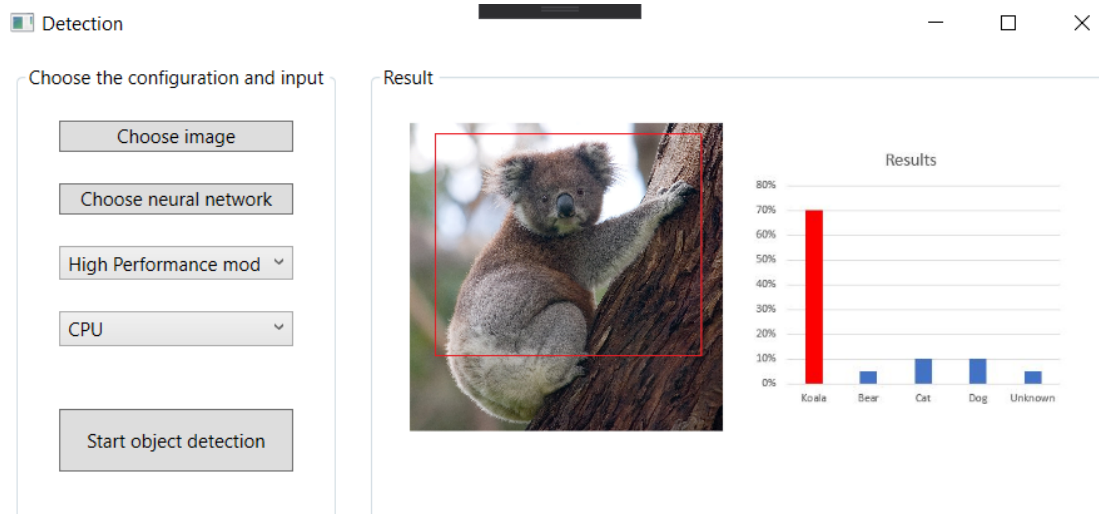


Figure 7: Object detection page of our software

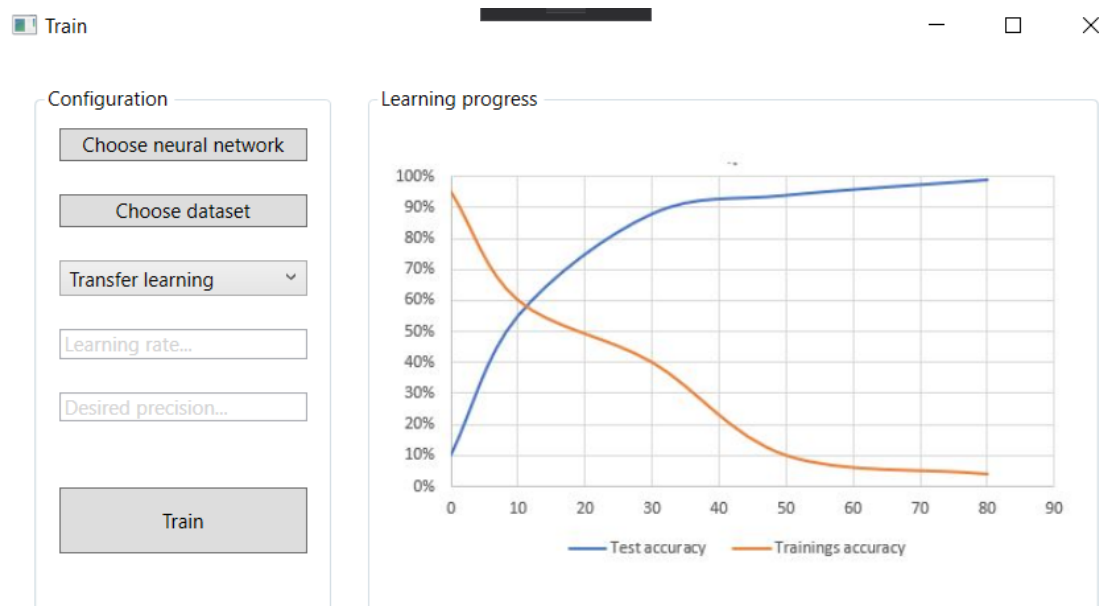


Figure 8: Training page of our software

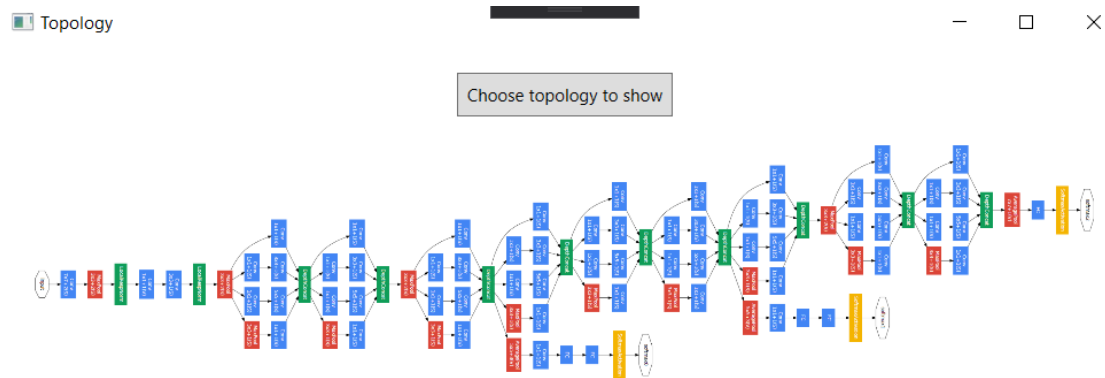


Figure 9: Page which shows the topology of a selected neural network of our software

12 Stage responsibilities

Requirements: Paul Stangel
Design: Johannes Häring
Implementation: Manuel Drehwald
Quality insurance: Stefani Guneshka
Deployment: Dimitar Dimitrov

13 Quality requirements

Name	Very relevant	Relevant	Less relevant	Not relevant
Failure tolerance	X			
Security				X
Usability		X		
Time requirement classification	X			
Time requirement detection		X		
Extendability	X			

Glossar

bandwidth The speed at which the data is transfered..

bounding box Rectangle indicating the outer edges of an object in an image..

CPU Central Processing Unit.

deployment platform Hardware to run the calculations on. (ex.: FPGA or CPU).

FLOPs Floating Point Operations Per Second.

FPGA Field Programmable Gate Array.

GPU Graphics Processing Unit.

host pc The main computer that is interacted with and used for input and output..

image A two dimensional matrix of red,green,blue (RGB) values that can be visualized as each cell represents a single pixel on the monitor. (ex.: a photo)..

image class A class of the dataset on which the neural network is trained..

image classification An object that is shown in a given picture is matched to a fitting class..

neural network A network or a circuit of neurons used for information processing inspired by the way biological neural systems process data..

operating mode Description the performance and power consumption used for calculations..

performance Performance is the combination of the amount of FLOPs and the bandwidth..

power consumption The power used per second by the system in Watts..