Praxis der Softwarentwicklung

Specificationsbook

Neural Network based Image Classification System on Heterogeneous Platforms

from

Team 2
Dimitrov, Drehwald, Guneshka, Häring, Stangel

Contents

1 Introduction

In todays world of globalisation and digitalisation, to keep up with the rapidly growing economy, one important challenge is the automatisation 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 quickly increasing 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 effiency 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 multiple images as input and returns the possiblities for each predefined image class for each image. 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 perfomance, 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 MAC010-MAC040 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 visualisation of the 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 choose different pretrained neural networks to use, before an image classification process.

CAC100 **Training neural networks**
The software allows the user to train a neural network based on an predefined topology.
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 with their results 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 and processes it.

CAC140 **Running neural network on GPU**
MAC020 is extended by GPU.

CAC150 **Dropdown Menu for input mode selection**
The software allows the user to select his preferred image input method using a dropdown menu.
The dropdown menu entries are: „Image(s)“, „Camera“, „.txt file with image paths“.

CAC160 **Output parameter „save all results“**
The software allows the user to select if the classification/detection output of each processed image should automatically be saved or not.

CAC170 **Output parameter „show all results “**
The software allows the user to select if the software should wait for a user input after each image.
If selected, the classification/ detection output for an image will be shown and the user has to click a button to continue with the next image.
If not choosen, all selected images should be processed in the background and detached from the gui.

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 prunning 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 with four sections, separated by *<section>:*
In the first, the class names are given, one per line.
In the second, hyperparameters are described as *<name>* = *<value>*.
In the third, layers are described in their order with the following format *[kind of layer]*, each followed by a list of layer-parameters in the format *<name>* = *<value>*
In the last section the weights and biases for each layer are listed.

PD030 **Labeled image set for classification training**
The dataset is chosen by the user. The dataset is a directory with images and each 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 format *<image name>_<neural network name>_<number>*. 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 format *<image name>_<neural network name>_<number>*. The format is *<image class name> = <probability>, <X>, <Y>, <width>, <height>*, one row for each image class.
If multiple images are detected, there is one .txt file for each input image.

PD070 **Video input**
The input video is in a .avi format.

7 Functional requirements

7.1 Image classification

7.1.1 Must

MFR010 **Use neural network for image classification**
Tested with: T010 Implements: MAC010
A neural network is used to classify images. The result is a list of prohabilities per image class.

MFR011 **Deploy pre-trained neural network with the corresponding layers**
Tested with: T011 Implements: MAC010
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 file**
Tested with: T012 Implements: MAC010
The software is able to read a configuration file of a neural network and parse for MFR011.

MFR013 **Layerimplementation**
Tested with: T010 Implements: MAC010
The software implements neural network layers needed for deploying convolutional networks. Those are layers such as convolutional, dropout, fully-connected, depth-concat layer.

7.1.2 Can

CFR020 **Voting of multiple neural networks**
Tested with: T170 Implements: CAC110
The user can choose multiple neural networks. The image classification is done on every neural network seperately and the results are weighted and accumulated. The accumulated result is shown.

7.2 Operating modes

7.2.1 Must

MFR030 **High performance operating mode**
Tested with: T020 Implements: MAC030
An operating mode to perform calculations as fast as possible.

MFR031 **Low power consumption operating mode**
Tested with: T021 Implements: MAC030
An operating mode to perform calculations with low power consumption.

MFR032 **Have high energy efficiency operating mode**
Tested with: T022 Implements: MAC030
An operating mode to perform calculations at an optimal ratio between performance and power consumption.

MFR033 **Calculator for power consumption**
Tested with: T023 Implements: MAC040
The software can calculate the power consumption on a given neural network and deployment platform.

MFR034 **Calculator for performance**
Tested with: T024 Implements: MAC040
The software can calculate the performance on a given neural network and deployment platform.

MFR035 **Dispatching the calculation process defined from the operating mode**
Tested with: T020, T021, T021 Implements: MAC030
The software is able to distribute the calculations to the deployment platforms regarding the operating mode.

7.3 Platforms

7.3.1 Must

MFR040 **Support CPU for calculation**
Tested with: T030 Implements: MAC020
The software supports CPU for calculation.

MFR041 **Support FPGA for calculation**
Tested with: T031 Implements: MAC020
The software supports FPGA for calculation.

MFR050 **Send image for classification**
Tested with: T040 Implements: MAC020
The software gives the image as input for the neural network to the chosen deployment platform.

MFR051 **Receive result**
Tested with: T041 Implements: MAC020
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.

7.3.2 Can

CFR060 **Support GPU for calculation**
Tested with: T160 Implements: CAC140
To speed up the calculations the program is able to use an additional GPU.

7.4 GUI

7.4.1 Must

MFR070 **GUI**
Tested with: T050 Implements: MAC050
The program has a Graphical User Interface to display all functions to the user.

MFR080 **Showing results**
Tested with: T060 Implements: MAC050
After executing the image classification, the results are shown in a bar chart. If the user classified multiple images there is the option to skip through the results.

MFR090 **Choosing image for classification**
Tested with: T070, T210 Implements: MAC050
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. The user can choose multiple images.

MFR100 **Choosing deployment platform**
Tested with: T080 Implements: MAC050
The GUI has checkboxes for choosing which deployment platforms are regarded. There are checkboxes for the devices which are supported. The devices which are theoretically supported but are not connected to the host pc or the communication with them does not work are grayed out and not clickable.

MFR110 **Choosing operating mode**
Tested with: T090 Implements: MAC050
The GUI has a dropdown which lists the different modes (high performance mode, low power consumption mode and high energy effiency mode). The power consumption in Watts and performance in FLOPs are also stated behind the operating mode names.

7.4.2 Can

CFR120 **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. If multiple neural networks are chosen the software uses the voting mode.

CFR130 **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.

CFR131 **Change the learning rate**
Tested with: T114 Implements: CAC100
To adjust the learning proccess of the neural network the user can change the speed of how fast the weights and biases will be changed with a textbox. Only floats are valid.

CFR140 **Visualisiaton of neural network**
Tested with: T120 Implements: CAC070
The software is able to visualise the topology of a neural networks (see figure 9)

CFR141 **Saving the visualisation**
Tested with: T121 Implements: CAC071
The user can save the visualition of the topology of a neural network as .png file to a chosen directory.

CFR150 **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.

CFR160 **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.

CFR170 **Dropdown menu for input selection**
Tested with: T180 Implements: CAC160
A dropdown menu is part of the classification and detection page.
It contains the four options: Single Images, Image folder, Camera, .txt File with Img Paths
Single image is MFR070
Image folder opens the file explorer with the filter for directories. Images with a valid format are loaded from this directory.
.txt File with image paths opens the file explorer with filter for .txt files.
Camera opens a connection to a connected camera and receives the video stream.

CFR180 **Selector for output parameter „save all results“**
Tested with: T190 Implements: CAC160
A checkbox is part of the classification and detection page.
If selected, for each image which has been selected by the user, the classification/ detection output will be stored as described in PD050 and PD060. Each output file is saved in the same folder as the input file.
If a camera is chosen as input, the output will be stored in a „camera“folder

CFR181 **Selector for output parameter „show all images“**
Tested with: T200 Implements: CAC170
If selected, the results can be skipped through. One image with its result is shown at a time.
If not selected, all images chosen by the user will be processed continuously without showing classification/ detection output in the GUI.

7.5 Training

7.5.1 Can

CFR190 **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.

CFR191 **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.

CFR192 **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.

CFR193 **Fit the output layer to the amount of image classes**
Tested with: T115 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.

7.6 Object detection

7.6.1 Can

CFR200 **Object detection**
Tested with: T130 Implements: CAC080
The software can detect the position and image class of objects in an image.

7.7 Video handling

7.7.1 Can

CFR210 **Connect with camera**
Tested with: T150 Implements: CAC130
The software can connect with a camera connected to the host pc.

CFR211 **Receive video stream from camera**
Tested with: T151 Implements: CAC130
The software can receive a video stream from the camera.

CFR212 **Apply classification/detection for a certain amount of frames**
Tested with: T152 Implements: CAC130
The software can divide a video or video stream into frames and is able to apply image classification and detection on those.

CFR213 **Saving the frames**
Tested with: T152 Implements: CAC120
The software divides the video into frames which are saved to the file system.

8 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 fourty (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.

9 Test cases

T010	**Use a neural network for image classification**
T010.1	**State:** An image of an elephant as input, a pretrained neural network, a deployment platform and an operating mode is given. **Action:** The user clicks on „Start image classification“. **Result:** The image is classified as elephant by the neural network and the 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“. **Result:** 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“. **Result:** 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 **Result:** 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 of an elephant 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. **Result:** 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 of an elephant 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. **Result:** 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	**State:** An image as input of an elephant, a pretrained neural network, a deployment platform is given. **Action:** The user chooses to perform the calculations in high energy efficiency operating mode and starts the classification. **Result:** The calculations run with regard to balance between power consumption and performance.
T023	**Calculator for power consumption**
T022.1	**State:** A pretrained neural network, a deployment platform and the operating mode is given. **Action:** The user chooses another operating mode. **Result:** The new power consumption is calculated automatically and then shown.
T024	**Calculator for performance**
T022.1	**State:** A pretrained neural network, a deployment platform and the operating mode is given. **Action:** The user chooses another operating mode. **Result:** The new performance is calculated automatically and then shown.
T030	**Support CPU for calculation**
T030.1	**State:** An image of an elephant as input, a pretrained neural network, CPU as deployment platform and an operating mode is given. **Action:** Click on the button „Start image classification“ **Result:** Elephant has the highest probability.
T031	**Support FPGA for calculation**
T031.1	**State:** An image of an elephant as input, a pretrained neural network, FPGA as deployment platform and an operating mode is given. **Action:** Click on the button „Start image classification“ **Result:** Elephant has the highest probability.
T040	**Send image for classification**
T040.1	**State:** An image of an elephant as input, a pretrained neural network, a deployment platform and an operating mode is given. **Action:** The user starts image classification **Result:** The software sends the image as array to the selected platform.

T041	**Receive result**
T041.1	**State:** The software is awaiting result. **Action:** Platform sends results. **Result:** The software receives the results from the platform and shows it.
T050	**GUI**
T050.1	**State:** The user wants to use the software. **Action:** The user starts the program. **Result:** The users sees the Graphical User Interface showed on Figure 1.
T060	**Showing results**
T060.1	**State:** The software awaits result. **Action:** The deployment platform sends result. **Result:** The Graphical User Interface shows the result in a bar chart as shown in figure 4.
T070	**Choosing image for classification**
T070.1	**State:** The user is on the page for image classification. **Action:** The user clicks on the button „Choose image“. **Result:** The file explorer opens with the filter for .png, .jpg, .bmp.
T070.2	**State:** The file explorer is open. **Action:** The user selects an image with a valid format. **Result:** The file explorer closes and image is loaded and shown as preview.
T080	**Choosing platform/hardware**
T080.1	**State:** The user is on the page for image classification. **Action:** The user chooses the desired deployment platform with the dropdown. **Result:** An internal flag is set to the desired platform and the dropdown shows the chosen deployment platform.
T090	**Choosing operating mode**
T090.1	**State:** The user is on the page for image classification. **Action:** The user chooses the desired operating mode with the dropdown. **Result:** An internal flag is set to the desired operating mode and the dropdown shows the chosen operating mode

T100	**Choosing between different neural network**
T100.1	**State:** The user is on the page for image classification. **Action:** The user clicks on the button „Choose neural network“. **Result:** The file explorer opens.
T100.2	**State:** The file explorer is open. **Action:** The user selects a config file. **Result:** The file explorer closes and the software loads the input and parses it. If it is loaded there is a success message shown.
T110	**Train neural network for classification of imageset**
T110.1	**State:** The user is on the main page. **Action:** The user clicks the button „Train a neural network“. **Result:**The user is redirected to a new page for training, shown in figure 6.
T110.2	**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. **Action:** The user clicks on the button „Train“ **Result:** The software starts to train the selected neural network and shows the progress in a line graph.
T110.3	**State:** The training is in process. **Action:** The precision reaches the desired precision. **Result:** The training stops.
T111	**Saving a neural network after training**
T111.1	**State:** The training finishes. **Action:** No action required. **Result:** The software stores the trained neural network in the directory of the selected .cfg file as a .cfg file.
T112	**Choosing and reading dataset**
112.1	**State:** The user is on the training page. **Action:** The user clicks on „Choose dataset“. **Result:** A file explorer opens.
T112.2	**State:** The file explorer is open. **Action:** The user chooses the folder with the images. **Result:** The program automatically iterates over all images and reads the given data that can be used for training.

T113	**Backpropagation**
T113.1	**State:** The user is on the training page, a dataset, a neural network, learning rate and desired precision are given. **Action:** The user clicks on „Train“. **Result:** 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.
T114	**Changing parameters**
T114.1	**State:** The user chose a neural network, the dataset and the desired precision. **Action:** The user changes the learning rate to a smaller number and starts training. **Result:** The neural network adjusts its weights but with smaller significance of one image.
T115	**Fit the output layer to the amount of image classes**
T115.1	**State:** The user chose a neural network with ten output nodes and a dataset with 12 image classes. **Action:** The user starts the training **Result:** The output layer is extended by two nodes as well as weights for the new connections are initilised.
T120	**Showing topology of a neural network**
T120.1	**State:** The user is on the main page. **Action:** The user clicks the „Show topology of a neural network“button. **Result:** The user is redirected to a new page for showing a topology.
T120.2	**State:** The user is on the page for showing the topology. **Action:** The user clicks on „Choose topology to show“ **Result:** The file explore opens
T120.3	**State:** The file explorer is open. **Action:** The user choses a .cfg file. **Result:** The file explorer closes and the topology is shown as in figure 9.
T121	**Saving topology visualisation**
T121.1	**State:** The user is on the show topology page and a topology is shown. **Action:** The user clicks on the save button. **Result:** The visualisation is saved to the file system.

T130	**Object detection**
T130.1	**State:** The detection window is open. An image as input, a pretrained neural network, a deployment platform and an operating mode is given. **Action:** The user clicks on the button „Start detection“ **Result:** The network is run for inferencing and the network output is shown to the user.
T131	**Drawing bounding box**
T131.1	**State:** Inferencing was executed on an image given by the user, the choosen neural network predicted bounding boxes. **Action:** No action required **Result:** The original image, given by the user, is overlayed with the boxes predicted by the network, the updated image is presented to the user.
T140	**Choosing video**
T140.1	**State:** The software is running. A pretrained neural network, a deployment platform and an operating mode is given. **Action:** The user selects a .avi video file. **Result:** The system stores the path to the selected video and is available to process images from this video sequentially.
T150	**Connect with camera**
T150.1	**State:** The software is running. **Action:** The user connects a usb camera to the host. **Result:** The system dynamically detects the camera and allows the user to select the camera as an image source
T150.2	**State:** A usb camera is connected to the host. The software is not running. **Action:** The user starts the software. **Result:** The system dynamically detects the camera and allows the user to select the camera as an image source
T151	**Receive video stream from camera**
T151.1	**State:** The software is running, a camera is available as image source. **Action:** The user chooses the camera as image source. **Result:** The first camera image is provided as a preview, the continuous image stream is available for further processing.

T152	**Apply classification for a certain amount of frames**
T152.1	**State:** The software is running. A video source was choosen by the user. All network details were provided by the user. Classification was choosen by the user. **Action:** The user clicks on the button „start classification“ **Result:** The software devides the video into frames and saves them. The software sequentially classifies the pictures and outputs the result as set.
T160	**Support GPU for classification**
T160.1	**State:** The classification window is open. An image as input, a pretrained neural network, a deployment platform and an operating mode is given. **Action:** The user chooses GPU as a deployment platform. The user clicks on the button „Start image classification“ **Result:** image classification is performed.
T170	**Voting of multiple neural network**
T170.1	**State:** The user is on the classification page. He has chosen three different neural network, an image, an mode, a deployment platform. **Action:** User starts image classification **Result:** The software starts the calculations. When those finished, the software receives three lists of the result. Elephant has the highest prohability in any results list. It is shown as result.
T180	**Choosing Input Mode**
T180.1	**State:** The classification window is open. **Action:** The user clicks on the dropdown Menu „Input Mode“ and selects „Single Images“ **Result:** The dropdown Menu shows the selection „Single Images“. The File explorer is set to show .jpg, .png and .bmp files
T180.2	**State:** The detection window is open. **Action:** The user clicks on the dropdown Menu „Input Mode“ and selects „Image Folder“ **Result:** The dropdown Menu shows the selection „Image Folder“. The File explorer is set to show folders only.
T180.3	**State:** The classification window is open. A Camera is connected to the Host PC **Action:** The user clicks on the dropdown Menu „Input Mode“ and selects „Camera“ **Result:** The dropdown Menu shows the selection „Camera“. The File explorer button is replaced by a text label saying „Camera connected“

T180.4 **State:** The detection window is open. A Camera is not connected to the Host PC
Action: The user clicks on the dropdown Menu „Input Mode“ and selects „Camera“
Result: The dropdown Menu shows the selection „Camera“. The File explorer button is replaced by a text label saying „No camera connected“

T180.5 **State:** The classification window is open.
Action: The user clicks on the dropdown Menu „Input Mode“ and selects „txt file with Image Paths“
Result: The dropdown Menu shows the selection „txt file with Image Paths“. The File explorer is set to show .txt files

T190 Output Mode „Save all results“

T190.1 **State:** The classification window is open. Input Mode was set to „Image Folder“ and a folder containing multiple .png images was selected. „Save all results“ has been selected. All other Input parameters have been set.
Action: The user clicks on „start detection“
Result: The systems calculates the classification output for each image. The output is stored in one .txt file per image as described in PD050

T190.2 **State:** The detection window is open. Input Mode was set to „txt file with Img Paths„ and a file containing multiple paths to .png images was selected. „Save all results“ has been selected. All other Input parameters have been set.
Action: The user clicks on „start detection“
Result: The systems calculates the detection output for each image. The output is stored in one .txt file per image as described in PD060

T200 Show all results

T200.1 **State:** The user choosed to classify five image and chose to show all results. The software awaits the results.
Action: The software receives the results.
Result: The first result and a continue button is shown.

T210	**Choosing multiple images**
T210.1	**State:** A neural network is chosen and the deployment platforms are chosen. **Action:** The user opens the file explorer and chooses three images. **Result:** The software loads the specified images.
T210.2	**State:** The specified images are loaded and ready to be classified. **Action:** The user clicks the „Classify „button. **Result:** The software calculates all images and shows one image with its result. Arrows are shown for the user to navigate the results as shown in figure 6.

10 System models

10.1 Scenarios

10.1.1 Scenario 1

The user U1 wants to classify the image of a cat. He goes on the classifcation page and he clicks on the dropdown and sees the three operating modes „low power consumption“, „high perfomance“and „high energy effiency“. 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.

10.1.2 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 effiency 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.

10.1.3 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.

10.1.4 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.

10.1.5 Scenario 5

User U5 has heared 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 larget 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.

10.1.6 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.

10.1.7 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.

10.1.8 Scenario 8

The company U8 wants to teach children 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 the software and 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 have fun at the same time.

10.2 Usecases

10.2.1 Image classification page

Figure 1: Usecase of the image classification page

10.2.2 Training page

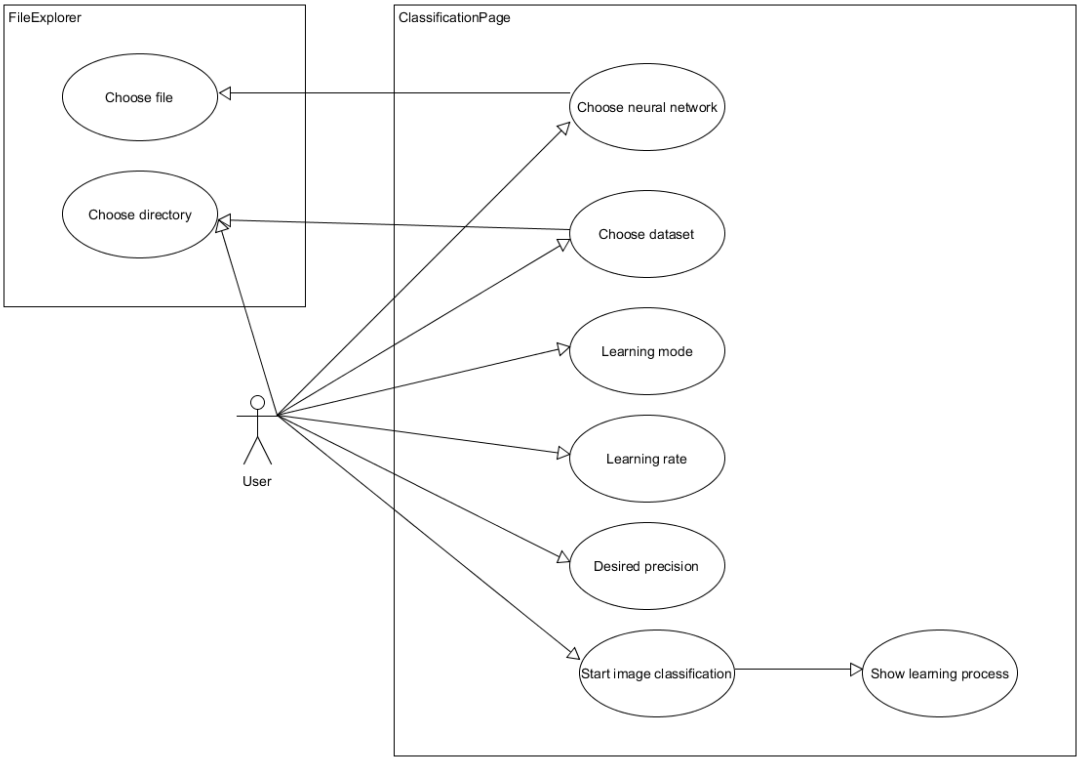

Figure 2: Usecase of the trainingspage

10.2.3 Image detection page

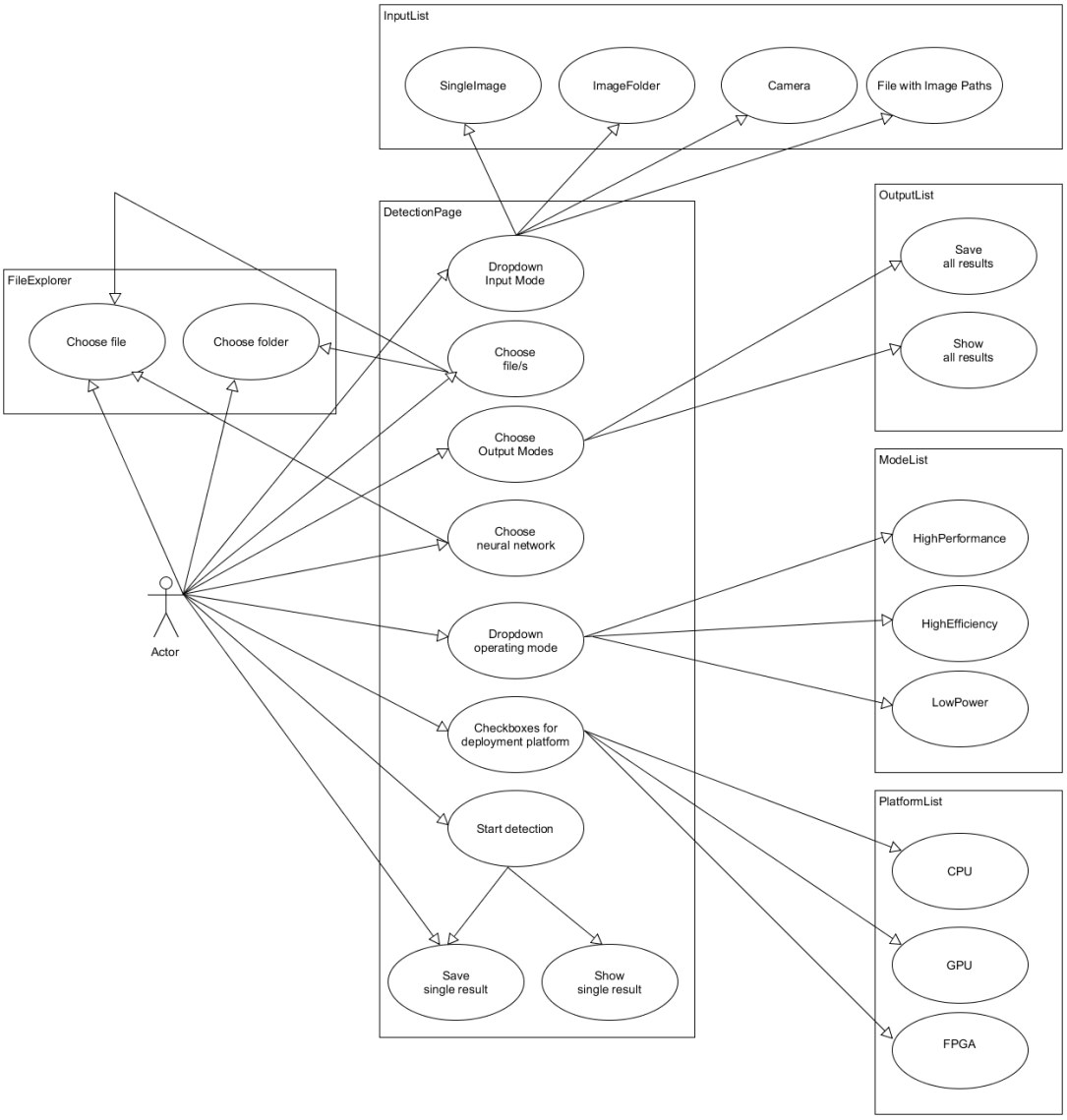

Figure 3: Usecase of the image detection page

10.2.4 Show topology page

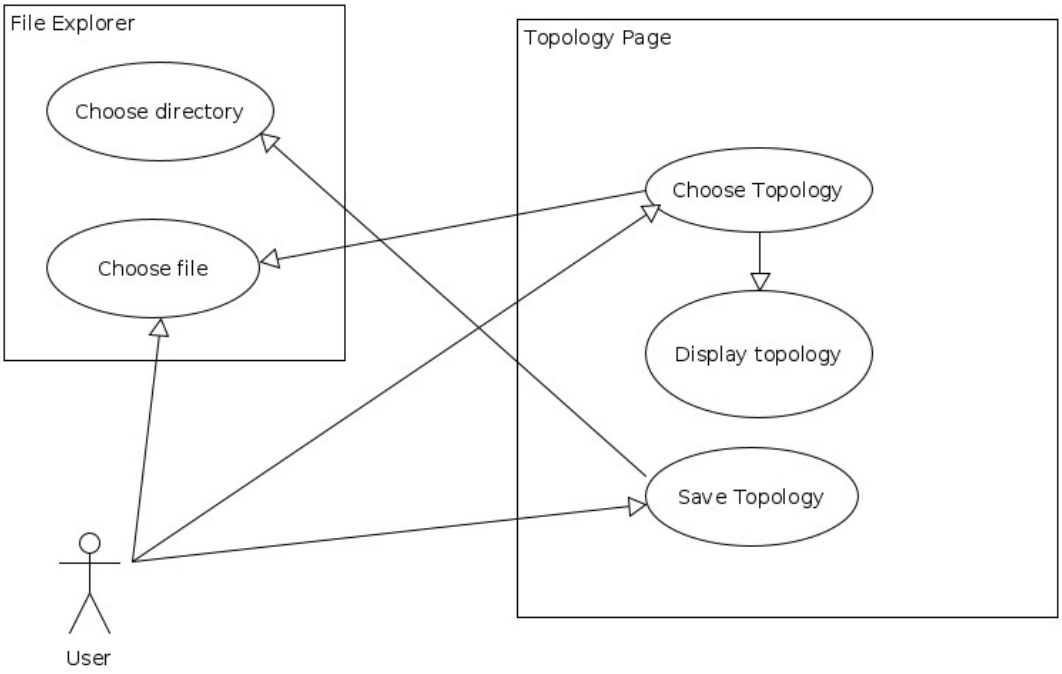

Figure 4: Usecase of the image classification page

10.3 GUI

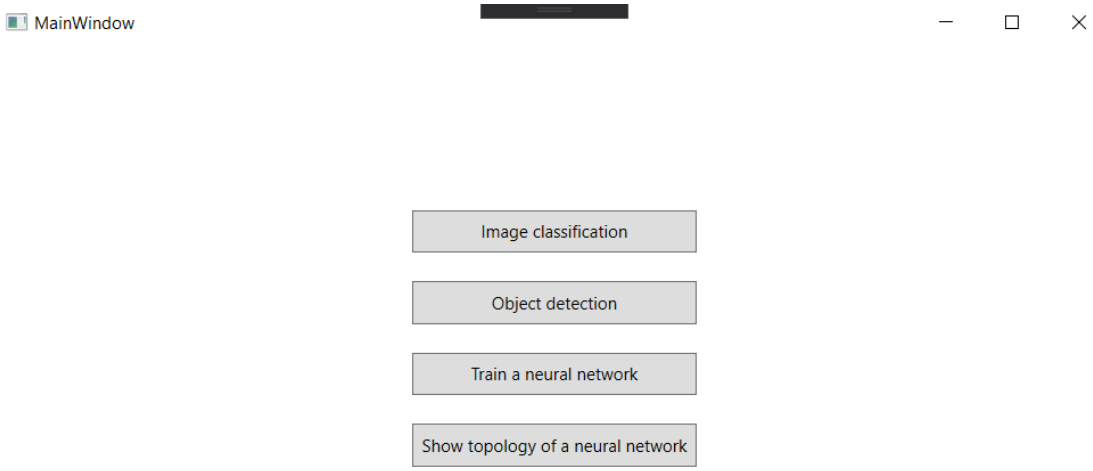

Figure 5: Main page of our software

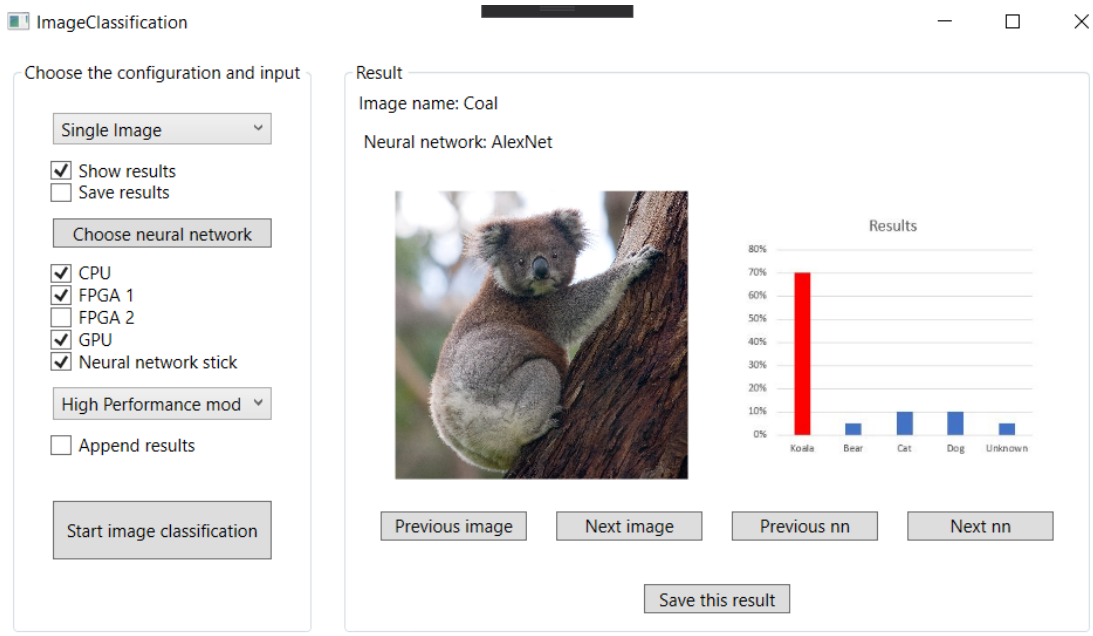

Figure 6: image classification page of our software

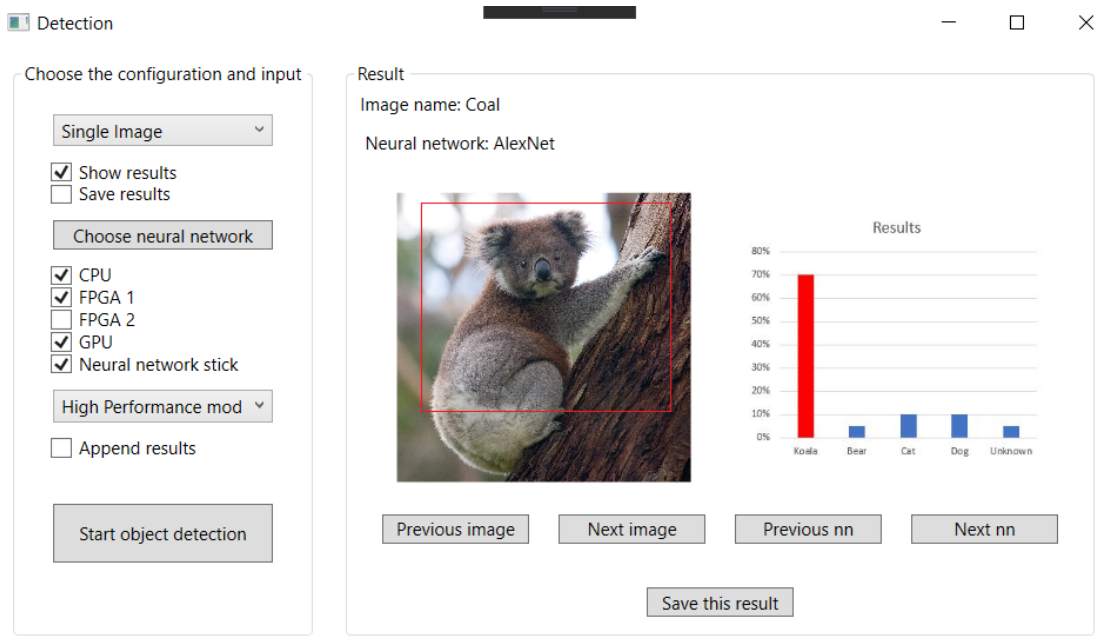

Figure 7: Object detection page of our software

Figure 8: Training page of our software

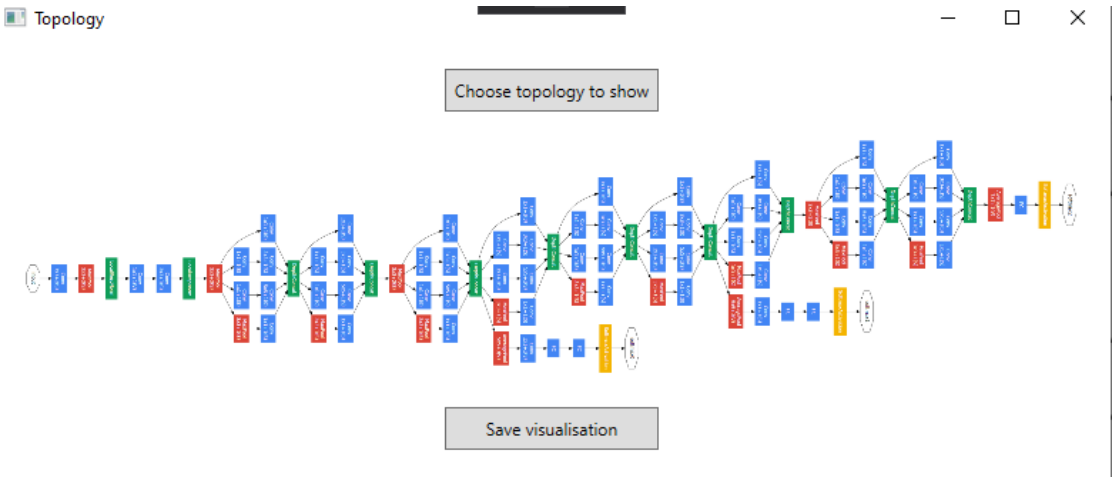

Figure 9: Page which shows the topology of a selected neural network of our software

11 Stage responsibilities

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

12 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			