Praxis der Softwarentwicklung

Specificationsbook

Neural Network based Image Classification System on Heterogeneous Platforms Team 2

from

Häring, Stangel, Drehwald, Guneshka, Dimitrov

Inhaltsverzeichnis

1 Introduction

In todays world of globalisation and digtalisation, to keep up with the rapidly growing economy, one important challenge is the automisation of tasks. One aspect of this is the classification of visual input. Whether it is to check for broken parts in production , the quality of a product or surveillance of public places. In the following project we want to build a functioning neural network that can classify images. To speed up the process of classification we will use different hardware that is more suitable for these kind of calculations. To further adjust the neural network to its task it should have different modes to function on. High performance, low power consumption and a mix of both. Because this is only a concept the classes do not play a big role. In the following specification book we will now refer to the neural network with NN.

2 Goal

The goal of this project is a program which performs accurate image classification and is able to switch between deploy platforms and working modes. It should also have a GUI to interact with the program and to visualise the results. Because it is just the base for a bigger specific task it should also be easily changeble to suit the right tasks. To achieve this clean code and good documentation is necessary.

3 Product use

The program does not have a specific use as there is no real automisation happening apart of the classification. It should function as a base for further more complicated tasks that can later for example be used for surveillance or error detection. The GUI should for illustration purposes, that anybody with minimal knowledge of the topic can use to try the classification.

4 Criterias

4.1 Must Acceptance criterias

AC10	**Image classification** The software can take a single image as input and tell the user to which predifined class, if any, this image belongs.
MAC020	**Running neural network on a heterogeneous platform** The software is able to do the execution of a given neural network (inference) on different compute devices. At least one CPU and one FPGA should be supported. The user is able to choose which compute device should be used for his image.
AC30	**Different operating modes** The software has three modes. One mode for high perfomance, one for low power consumption and one for high energy effiency.
AC40	**GUI for interacting with software** The user should be able to access the entire functionality described in AC10-AC50 just by using the functional GUI. No coding or command line usage is required.
AC50	**Performance and power consumption prediction** The software can predict the performance with a certain powerconsumption and also the powerconsumption for a certain performance.

4.2 Can Acceptance criterias

KAC070 **Illustration of the topology of a nn**
The software is able to visualize the topology of a given nn in a usefull way without requirering additional information.
The visualized nn can be saved as a .png file

KAC80 **Object Detection**
The software is able to not only say what kind of species there is on the picture, but also detect its outer points and draw a bounding box around it.

KAC100 **Creating new models**
The software allows the user to train a neural network based on an architecture which the user developed.
Neural networks created and trained by the user will be executed the same way neural networks provided with the software are executed.

KAC110 **Voting of multiple nn**
The user is able to choose multiple nn for classification.
The software will then execute all selected neural networks sequentially. The result presented to the User will be based on the weighted opinions of the different neural networks.

KAC120 **Using video for classification**
The software is able to take a video, divide it in frames and perform image classification for each frame.
The classified video can be viewed side by side as if changing to the next image at a constant rate and the classification could be saved as a JSON or an XML file.

KAC130 **Using camera for classification input**
The software is able to take frames captured with a camera as an input and classify them.
The process could take the current frame, classify it, display the results and then when ready, take the next available frame.

KAC060 : Training a nn for classification
KAC090 : Choosing between different models
KAC140 : Running NN on GPU
KAC150 : The GUI covers all implemented features in 4.1 and 4.2

4.3 Criteria of demarcation

D010 **No low-level optimization**
Optimizations 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.
Common code optimizations will be done where possible 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 parameter sharing, prunning or binarization will be implemented.

D040 **No mobile support**
Our software doesn't work on mobile devices, like smartphones or wearabels.

5 Product environment

The program should run on a computer in the lab. It should have a CPU and an external FPGA. Additionally there can be a GPU.

6 Functional Requirements Must

MFR010 **Use neural network for image classification**
A neural network should be used in order to classify images based on what is shown on them. For each image a list of possible classes it could belong to along with degree of confidence should be given as output.

MFR011 **Deploy pre-trained neural network with the corresponding layers**
A pre-trained neural network should be deployed to with all the corresponding layers in order to fulfill MFR010.

MFR012 **Reading and parsing neural network configuration/weight file**
The software is able to a configuration file of a speciffic neural network and parse it for use in the classification.

MFR020 **Have high performance operating mode**
An option to perform calculations fast with low regard for power consumption.

MFR021 **Have low power consumption operating mode**
An option to perform calculations with low power consumption and low regard for speed.

MFR022 **Have high energy efficiency operating mode**
An option to perform calculations at an adequate balance between speed and power consumption.

MFR023 **Calculator for power consumption**
Calculations for the possible power consumption running the image classifications would result in based on the neural network, platform and operating mode used.

MFR024 **Calculator for performance**
Calculations for the possible performance running the image classifications would result in based on the neural network, platform and operating mode used.

MFR025 **Dispatching the calculation process defined from the mode**
Tested with: Implements:
The program should be able to control the tact clock rate of the processor and synchronise it to the chosen mode.

MFR030 **Support CPU for calculation**
Tested with: Implements
The program should support CPU for calculation.

MFR031 **Support FPGA for calculation**
Tested with: Implements
The program should support FPGA for calculation.

MFR040 **Send image for classification**
Tested with: Implements
The GUI has a button Image Classification which opens a new window on click, where the user can choose different modes and platforms.

MFR041 **Receive result**
Tested with: Implements
The program should be able to receive results of the executed Image Classification from different platforms.

MFR050 **GUI**
Tested with: Implements:
The program has a Graphical User Interface to display all functions to the user

MFR060 **Showing results**
Tested with: Implements
After executing the Image Classification, the user should be able to see the results from the execution.

MFR070 **Choosing image for classification**
Testet with: Implements:
The GUI has a button with an on click event which opens a file explorer. The explorer filters the files so that only files of the format .jpg, .png, .bmp are listed. That also are the only valid formats.

MFR080 **Choosing platform/hardware**
Testet with: Implements:
The GUI has a dropdown which lists the devices on which the classification can be done. The devices which can be theoretically be accessed but aren't connected to the host pc or the communication with them doesn't work are grayed out.

MFR090 **Choosing mode**
Testet with: Implements:
The GUI has dropdown which lists the different modes (high performance mode, low power consumption mode and best energy effiency mode). The power consumption in Watts and performance in FLOPs are also stated behind the mode names.

7 Functional Requirements Can

CFR100 **Choosing between different models**
Testet with: Implements:
The GUI has a button which opens the file explorer which filters for .txt files, there you choose the config file of the neural network with which you want to use. The program loads this config and parses it so it can be deployed. Possible models are GoogLeNet or AlexNet.

CFR110 **Train nn for classification of imageset (with transfer learning)**
Testet with: Implements:
The user chooses a pretrained neural network and a new imageset and then can train the neural network on this new imageset with transfer learning.

CFR111 **Saving newly trained NN (config and weights)**
Tested with: Implements
The program should be able to take the weights and configs of an already existing NN and save it for later uses without reading the data again.

CFR112 **Choosing and reading data set**
Tested with: Implements
The program has an option to select a set of labeled images it can use to train and improve its performance.

CFR113 **Backpropagation**
Tested with: Implements
The program is able to adjust its weights and biases to improve the classification and get more accurate predictions by backpropagation.

CFR114 **Change the learning rate**
Tested with: Implements
To adjust the learning proccess of the neural network you can change the speed of how fast the weights and biases will be changed.

CFR120 **Illustrating NN topology.**
Tested with: Implements
To make the program more intuitive for the user it should be able to illustrate the topology of the NN.

CFR130 **Object detection algorithm**
Tested with: Implements
Not only should the program classify an image it should also detect the objects/ classes in the picture.

CFR131 **Showing detected object**
Tested with: Implements
The found objects should be marked by a box and shown to the user.

CFR140 **Choosing and reading video**
The user can choose a video in .avi format and the software can use it as input for the classification process

CFR150 **Connect with camera**
The software can connect with a connected camera

CFR151 **Receive video stream from camera**
The software can receive a video stream from the camera.

CFR152 **Apply classifcation for a certain amount of frames**
The software can devide a video or videostream into frames and is able to apply image classification on those.

CFR160 **Support GPU for calculation**
Tested with: Implements
To speed up the calculations the program should be able to use a additional GPU to speed up the calculations.

8 Productdata

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**
It is a .cfg file. In the beginning are hyperparameters 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*

PD030 **Labeled image set for classification training**
The dataset is chosen by the user. The dataset is a directory with images and the name of the image is the label.

PD040 **Labeled set of images for object detection training**
It is a .txt file and a directory with images. The images are labeled with their name. The bounding box for each image are described in the .txt file, in the format *imagename, x,y,width,height*. (X,Y) are the coordinates in pixel of the left bottom corner, the width and height are in pixel.

9 Non-functional requirements

D010 **Project size**
The project should have around ten thousand (10,000) lines of code

D020 **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.

D030 **Model-View-Controller**
The project should be based on the design pattern model-view-controller.

10 Test cases

T010 **Use neural network for image classification**
State: A image is given as an input.
Action: Calculations are performed on hand of the image and a neural network.
Reaction: A list of possible classes the given image could belong to along with degree of confidence for each class are given as output.

T011 **Deploy pre-trained neural network with the corresponding layers**
State: There is a neural network (already trained).
Action: Calculations are performed cased on a given image and the given neural network.
Reaction: A list of possible classes the given image could belong to along with degree of confidence for each class are given as output.

T012 **Reading and parsing neural network configuration/weight file**

T012.1 **State:** The user is on the page to select a neural network to use for the image classification.
Action: The user selects the option to import a neural network.
Reaction: The file explorer opens.

T012.2 **State:** The file explorer is open
Action: The user selects an neural network to import
Reaction: The file explorer closes and neural network is imported and selected for the classification calculations.

T020 **Have high performance operating mode**
State: The user is ready to start the calculations.
Action: The user chooses to perform the calculations in high performance operating mode.
Reaction: The calculations run considerably faster than in the other possible modes with the same conditions.

T021 **Have low power consumption operating mode**
State: The user is ready to start the calculations.
Action: The user chooses to perform the calculations in low power consumption operating mode.
Reaction: The calculations run with considerably lower power consumption than with the other possible modes in the same conditions.

T022 **Have high energy efficiency operating mode**
State: The user is ready to start the calculations.
Action: The user chooses to perform the calculations in high energy efficiency operating mode.
Reaction: The calculations run with regard to balance between power consumption and speed.

T025	**Dispatching the calculation process defined from the mode**
	Warming up: Run the Software for each Case 10 times with the same data set.
T025.1	**High performance mode** :
	State: The user in on the page for image classification
	Action: The user selects High performance mode
	Reaction: The calcutation is expected to be faster than the other modes.
T025.2	**Low power consumption mode** :
	State: The user in on the page for image classification
	Action: The user selects low power consumption mode
	Reaction: The calculation is expected to be using lower amount of power than the other modes.
T025.3	**High energy efficiency mode** :
	State: The user in on the page for image classification
	Action: The user selects high energy efficiency mode
	Reaction: The calculation is expected to be using lower amount of energy than the other modes.
T030	**Support CPU for calculation**
T030.1	**State:** The user chooses an elephant, CPU as a platform and performance mode.
	Action: Click on the button „Start image classification“
	Reaction: Elephant is the resultt.
T031	**Support FPGA for calculation**
T031.1	**State:** The user chooses an elephant, FPGA as a platform and performance mode.
	Action: Click on the button „Start image classification“
	Reaction: Elephant is the result.
T040	**Send image for classification**
T040.1	**State:** The user in on the page for image classification
	Action: The user selects an image to be classificated.
	Reaction: The software sends an image to the selected platform.
T041	**Receive result**
T041.1	**State:** The software is awaiting result
	Action: Platform sends results
	Reaction: The software receives the results from the platform.

T050	**GUI**
T050.1	**State:** The user wants to use the software. **Action:** The user starts the program. **Reaction:** The users sees the Graphical User Interface showed on Figure 1.
T060	**Showing results**
T060.1	**State:** The user has send an image for classification **Action:** **Reaction:** The Graphical user interface shows the finished image.
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“. **Reaction:** 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 **Reaction:** The file explorer closes and image is as preview shown
T080	**Choosing platform/hardware**
T080.1	**State:** The user is on the page for image classification **Action:** The user chooses with the dropdown the desired platform **Reaction:** An internal flag is set to the desired platform and the dropdown shows the chosen platform.
T090	**Choosing mode**
T090.1	**State:** The user is on the page for image classification **Action:** The user chooses with the dropdown the desired mode **Reaction:** An internal flag is set to the desired mode and the dropdown shows the chosen mode
T100	**Choosing between different models**
T100.1	**State:** The user is on the page for image classification **Action:** The user clicks on the button „Choose neural network“ **Reaction:** The file explorer opens
T100.2	**State:** The file explorer is open **Action:** The user selects an config/weight file **Reaction:** The file explorer closes and the software loads the input and parses it. If it is loaded there is success message

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“ **Reaction:**The user is redirected to a new page with further options.
T110.2	**State:** The user is on the page for training, has selected a neural network, a dataset for training, the kind of training, the learning rate and the desired precision. **Action:** The user clicks on the button „Train“ **Reaction:** The software starts to train the selected network with the selected configuration and shows the progress in line graph.
T110.3	**State:** The user chose the dataset and chose all options: dataset, model **Action:** The user clicks the train button. **Reaction:** The NN is training and after training the new topology and results of the NN is shown on a new site, that has a back button to go to the main page and a save button to save new trained NN
T111	**Saving a NN after training**
T111.1	**State:** The user is on the page for training, has selected a neural network, a dataset for training, the kind of training, the learning rate and the desired precision. The training finishes. **Action:** No action required **Reaction:** The software stores the trained network weights in a predifined format and with a usefull name in a fixed location.
T112	**Choosing and reading dataset**
T112.1	**Step 1: Folder** : **State:** The user has a folder with labeled images and already clicked on the train button. **Action:** The user clicks the input button. **Reaction:** A file explorer opens.
T112.2	**Step 2: Folder** : **State:** The file explorer is openend. **Action:** The user chooses the folder with the images. **Reaction:** The program automatically iterates over all images and reads the given data that can be used according to the users later actions.

T113	**Saving the trained NN**
T113.1	**State:**
	Action:
	Reaction:
T114	**Backpropagation**
T113.1	**State:**
	Action:
	Reaction:
T115	**Changing parameters**
T115.1	**State:**
	Action:
	Reaction:
T120	**Showing topology of a NN**
T140.1	**State:** The user is on the main page
	Action: The user clicks the „Show topology of a neural network“button.
	Reaction: The user is redirected to a new page where he has to choose the neural network to display by a dropdown menu
T130	**Object detection**
T130.1	**State:** The software is running, a network including all parameters and weights was provided. An image was provided. Detection mode was picked by the user
	Action: The user clicks on the button „start detection“
	Reaction: The network is run for inferencing and the network output is shown to the user.
T131	**Marking objects**
T131.1	**State:** The program found the location of object.
	Action: none
	Reaction: The picture is shown with a red square around the classified object.
T132	**Drawing bounding box**
T132.1	**State:** Inferencing was executed on an image given by the user, the choosen neural network predicted bounding boxes.
	Action: No action required
	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.

T140	**Choosing video**
T140.1	**State:** The software is running. **Action:** The user selects a .avi video file. **Reaction:** The system stores the path to the selected video and is available to process single images from this video.
T150	**Connect with camera**
T150.1	**State:** The software is running **Action:** The user connects a usb camera to the pc **Reaction:** 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 connected to the host pc. **Action:** The user chooses the camera as image source. **Reaction:** 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“ **Reaction:** The system processes the video file imagewise
T160	**Support GPU for classification**
	State: The user chooses an elephant, GPU as a platform and performance mode. **Action:** Click on the button „Start image classification“ **Reaction:** Elephant is the result.

11 System models

11.1 Scenarios

11.1.1 Scenario 1

The user U1 wants to classificate the image of a cat. He goes on the classifcation page and he clicks on the dropdown and sees the three 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.

11.1.2 Scenario 2

The user U2 goes to the classification page and chooses the image of 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.

11.1.3 Scenario 3

The user U3 created the blueprint for a new nn. He wants to train a network based on this config file but computation time is shared and expensive. U3 has to convince his boss therefore. U3 uses the software as input and select the visualization toolkit. U3 saves the output and uses it during the discussion to demonstrate the advantages of his new neural network.

11.1.4 Scenario 4

User U4 has to categorize a large dataset of plants from a biology excursion. 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 pick manualy for every image which network to use. 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 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 a 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.

11.1.6 Scenario 7

Alex had a trip in Africa and made a lot of pictures of animals. He looks for an easy way to know how many different sorts 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, what 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 8

The firm GoZoo wants to develop an AI to feed the animals at Zoos. The Firm doesn't have enough labelers to label all of the frames they need to teach the software which animal is it seeing at the moment. GoZoo decides to use Tucs's 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 9

The firm EducationFirst wants to teach small kids parallel to read, recognize percents and animals. Tucs is just right for the job, because of the Image Classification option of the software. The CEO of EducationFirst 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 Tucs, the kids are able to learn and also having fun at the same time.

11.2 Usecases

11.2.1 Training page

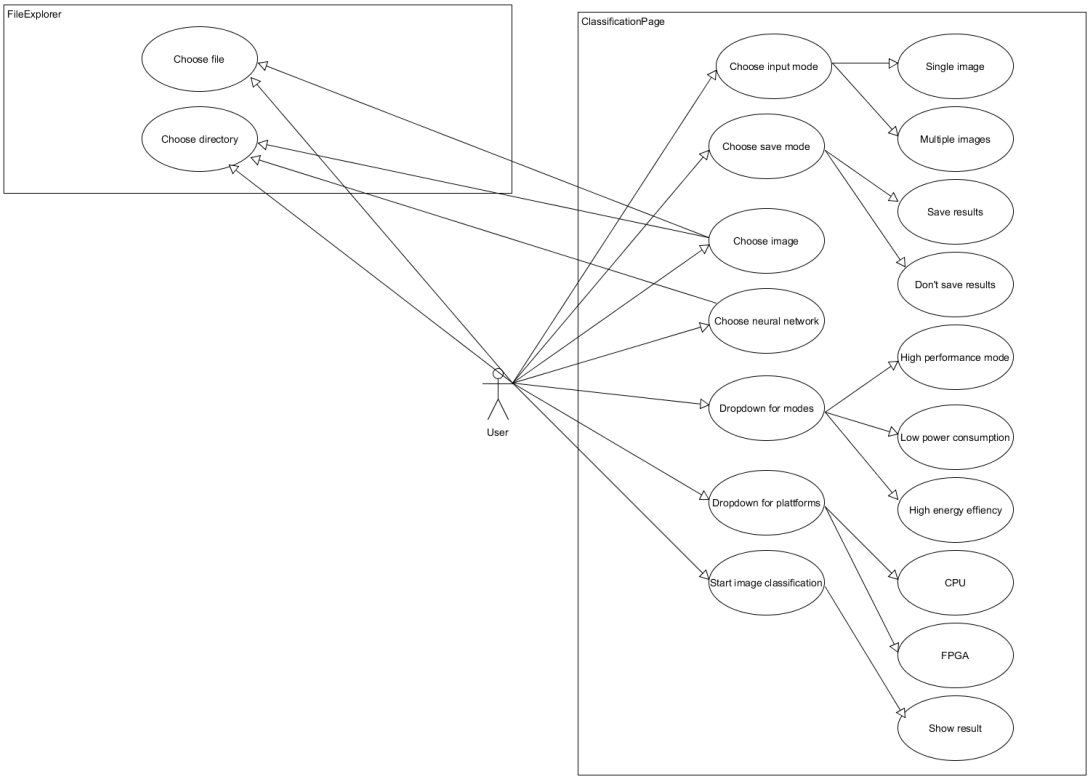

Figure 1: Usecase of the image classification page

11.2.2 Training page

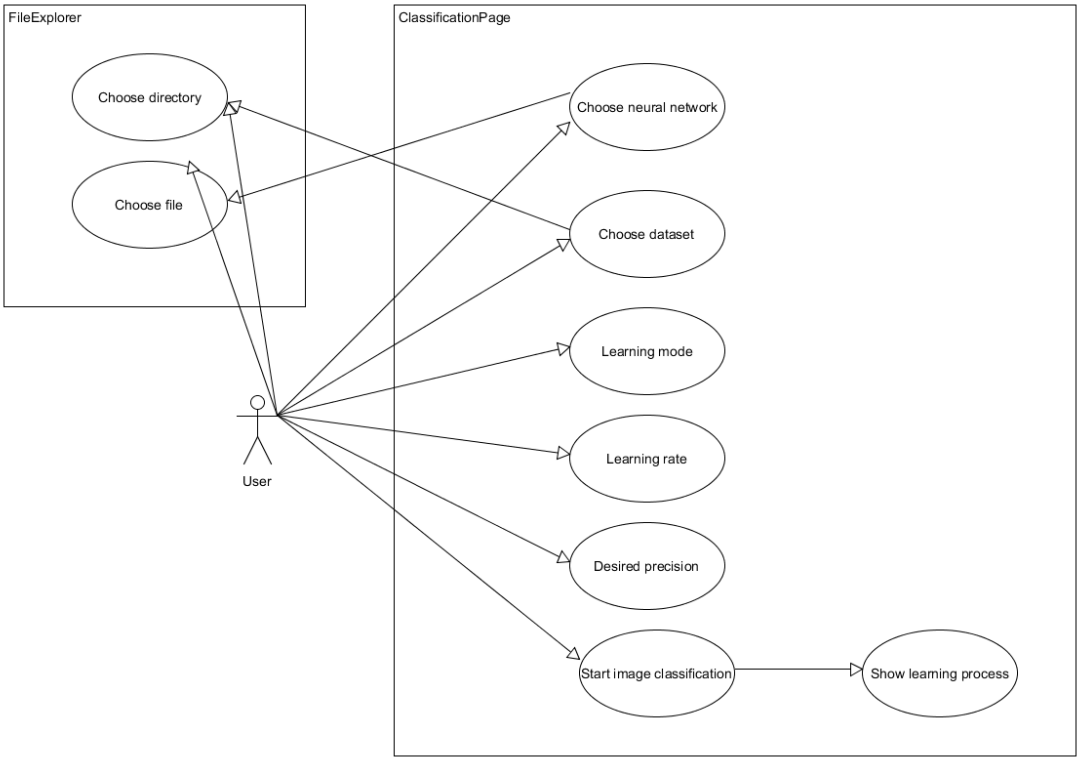

Figure 2: Usecase of the trainingspage

11.3 GUI

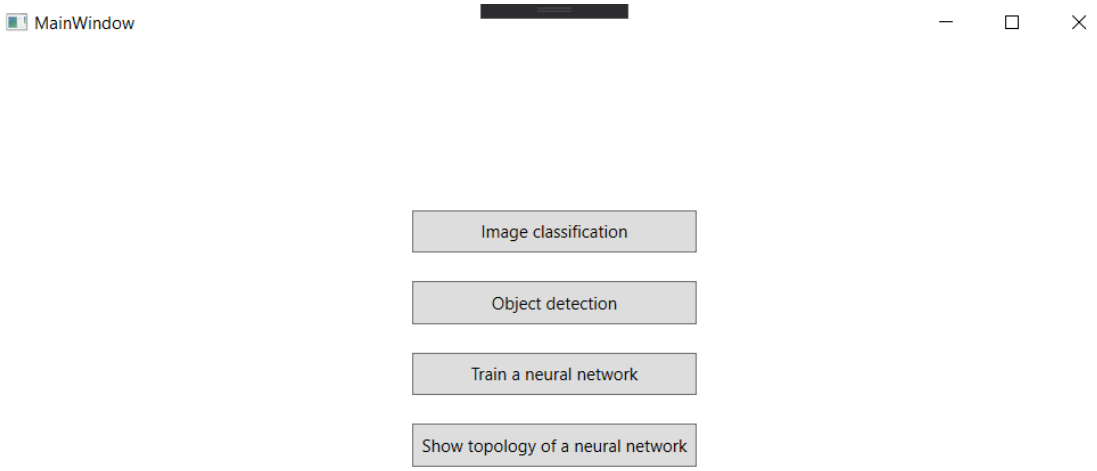

Figure 3: Main page of our software

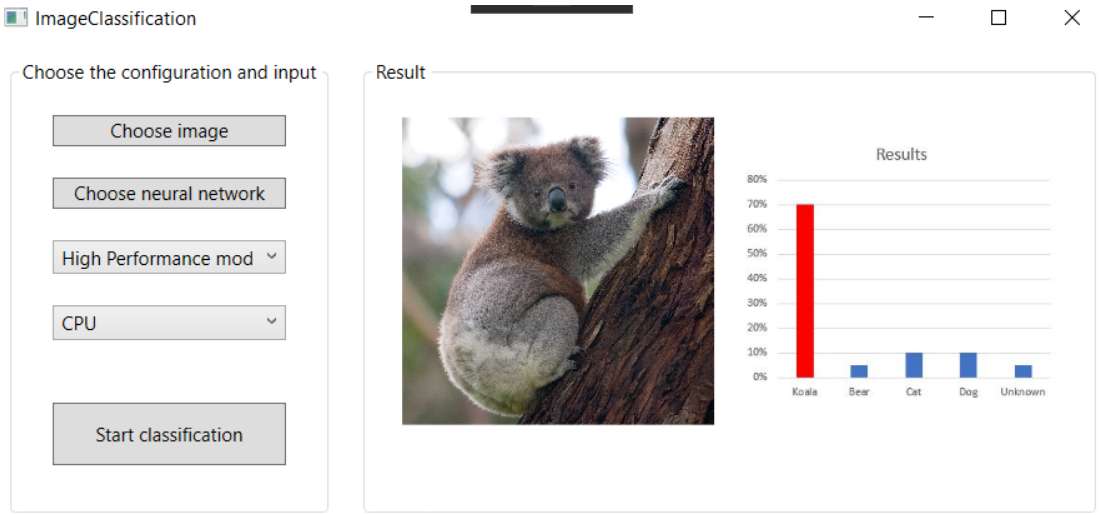

Figure 4: Image classification page of our software

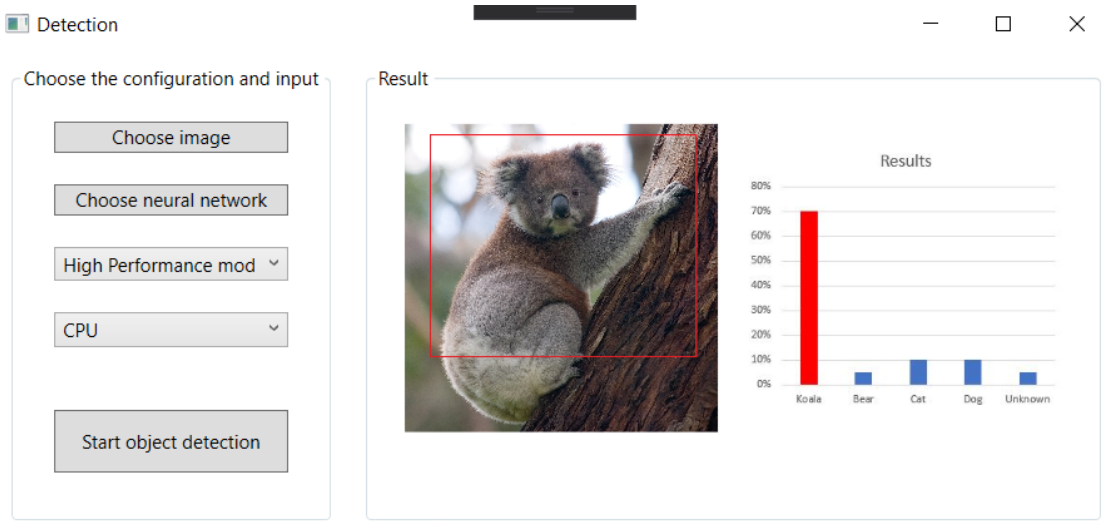

Figure 5: Object detection page of our software

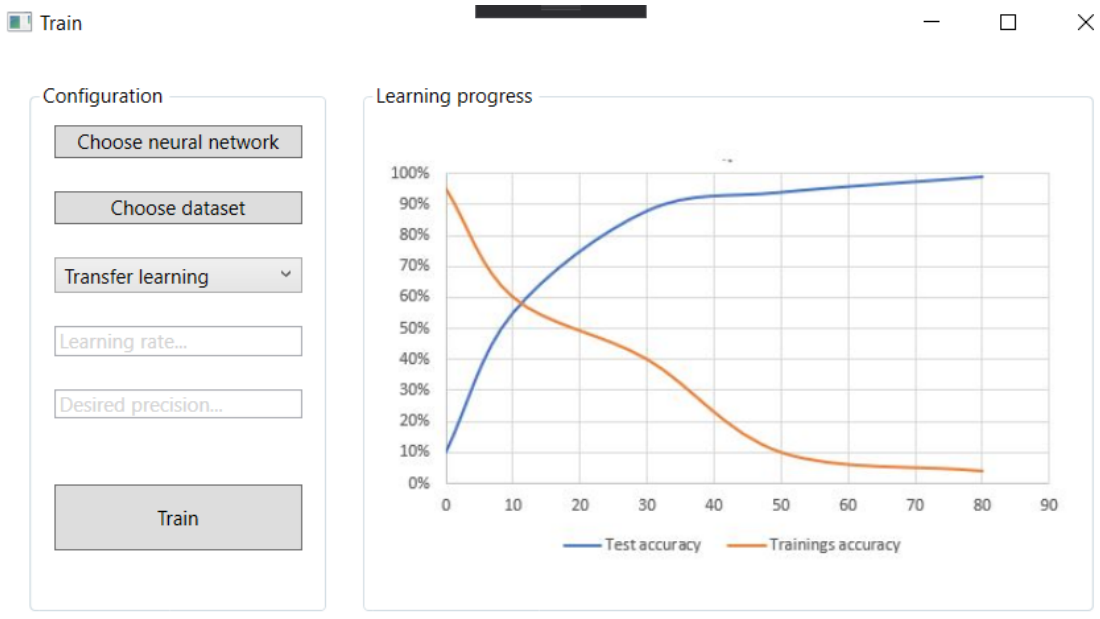

Figure 6: Training page of our software

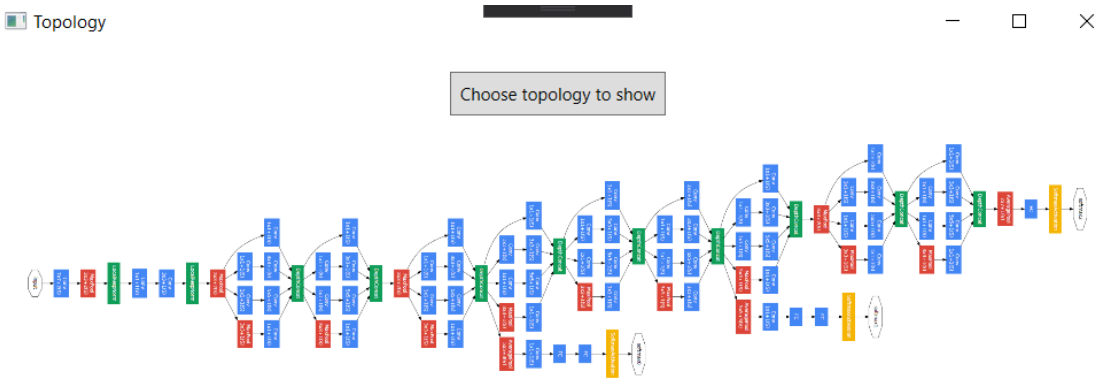

Figure 7: Page which shows the topology of a selected NN of our software

12 Stage responsibilities

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

Glossar

CPU Central Processing Unit.

FPGA Field Programmable Gate Array.

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).

JSON JavaScript Object Notation.

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

XML Extensible Markup Language.