Praxis der Softwarentwicklung

Specificationsbook

Neural Network based Image Classification System on Heterogeneous Platforms Team 2

from

Häring, Stangel, Drehwald, Guneshka, Dimitrov

Inhaltsverzeichnis

1 Preface

2 Goal

The goal is a software which performs image classification and is able to switch between deploy platforms and working modes. It also should have a GUI to control the software and to show the results.

3 Product use

Image classification

4 Acceptance criteria

4.1 Must

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

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

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
KAC080 : Object detection
KAC090 : Choosing between different models
KAC100 : Creating new models
KAC110 : Voting of multiple nn
KAC140 : Running NN on GPU
KAC150 : The GUI covers all implemented features in 4.1 and 4.2

5 Functional Requirements Must

MFR025 : Dispatching the calculation process defined from the mode
MFR030 : Support CPU for calculation
MFR031 : Support FPGA for calculation
MFR032 : Support GPU for calculation
MFR040 : Communication between Host-PC and platform
MFR041 : Send image for classification
MFR042 : Receive result
MFR050 : GUI
MFR060 : Showing results

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.

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.

FR070 **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.

FR080 **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.

FR090 **Choosing mode**
Testet with: Implements:
The GUI has dropdown which lists the 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.

6 Functional Requirements Can

CFR012 **Reading and parsing neural network configuration/weight file**
Having the ability to import/export different (external) neural networks to use for the image classification.

FR100 **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.

FR110 **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.

KFR032 : Support GPU for calculation
KFR113 : Backpropagation
KFR114 : Choosing parameters like learning rate
KFR120 : Illustrating nn topology
KFR130 : Object detection algorithm
KFR131 : Showing detected object
KFR132 : Choosing between detection and classification mode

7 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.

8 Demarcation

D010 **No low-level optimization**
Optimizations to reduce the execution time of object classification and detection will mainly be carried out in OpenCL.
No optimizations including low-level languages or assembly intrinsics will be implemented.

D020 **No real time optimization**
Common code optimizations will be done where possible to reduce the running time of the network per image classification/detection task.
They do not have to lead to real-time reactions of the system.
A computation time of multiple seconds per image is acceptable.

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**
There are no intentations to run any parts of this code on a mobile device like a smartphone or Augmented Reality glass.
Mobile device requirements are not taken into consideration when choosing Techniques, languages and hardware used in this project.

9 Non-functional requirements

NF10

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.

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 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.
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/Reading data set**
T112.1	**State:** A folder containing images is provided by the user. **Action:** No action required The software is able to read all images matching the allowed formats and allows training and inferencing on these images.
T140	**Creating new topology**
T140.1	**State:** **Action:** No action required **Reaction:** A folder containing images is provided by the user. The software is able to read all images matching the allowed formats and allows training and inferencing on these images.

T150	**Choosing between training and interference mode**
T150.1	**State:** **Action:** No action required **Reaction:** A folder containing images is provided by the user. The software is able to read all images matching the allowed formats and allows training and inferencing on these images.
T160	**Choosing video in format .avi**
T160.1	**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.
T161	**Apply classification for a certain amount of frames**
T161.1	**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
T170	**Connect with camera**
T170.1	**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
T171	**Receive video stream from camera**
T171.1	**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.
T180	**Detecting object**
T180.1	**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.
T181	**Drawing bounding box**
T181.1	**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.

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.

11.2 Usecases

11.2.1 Training page

11.2.2 Training page

11.3 GUI

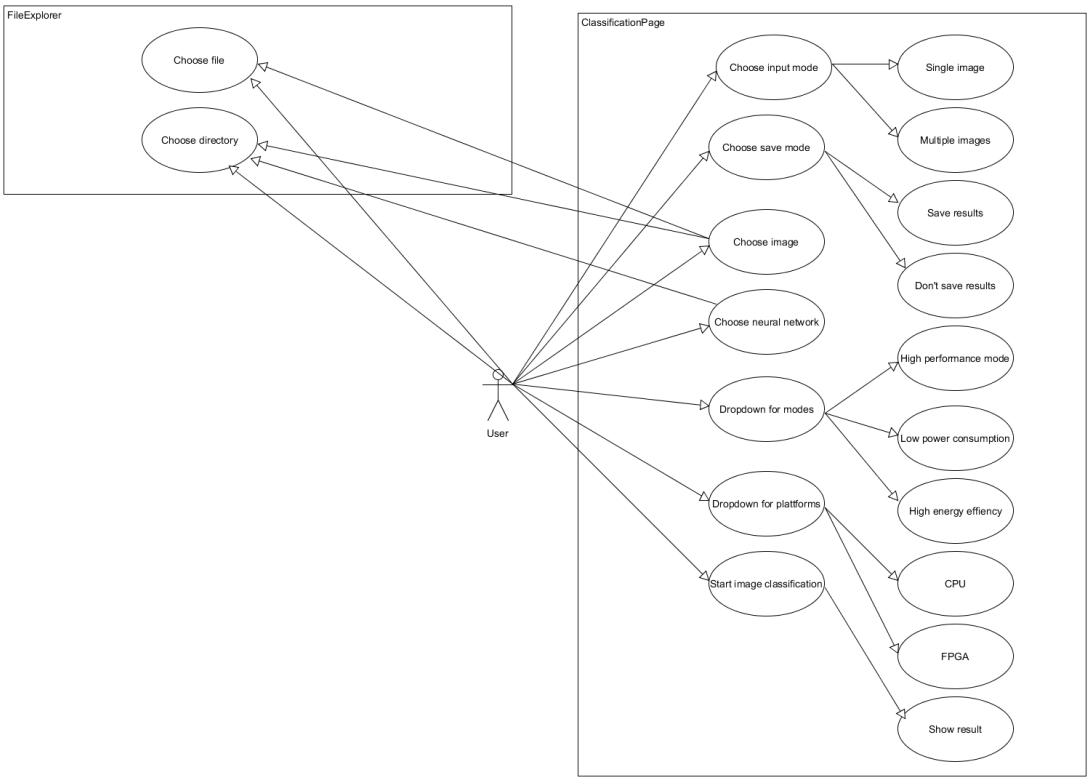

Abbildung 1: Usecase of the image classification page

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.

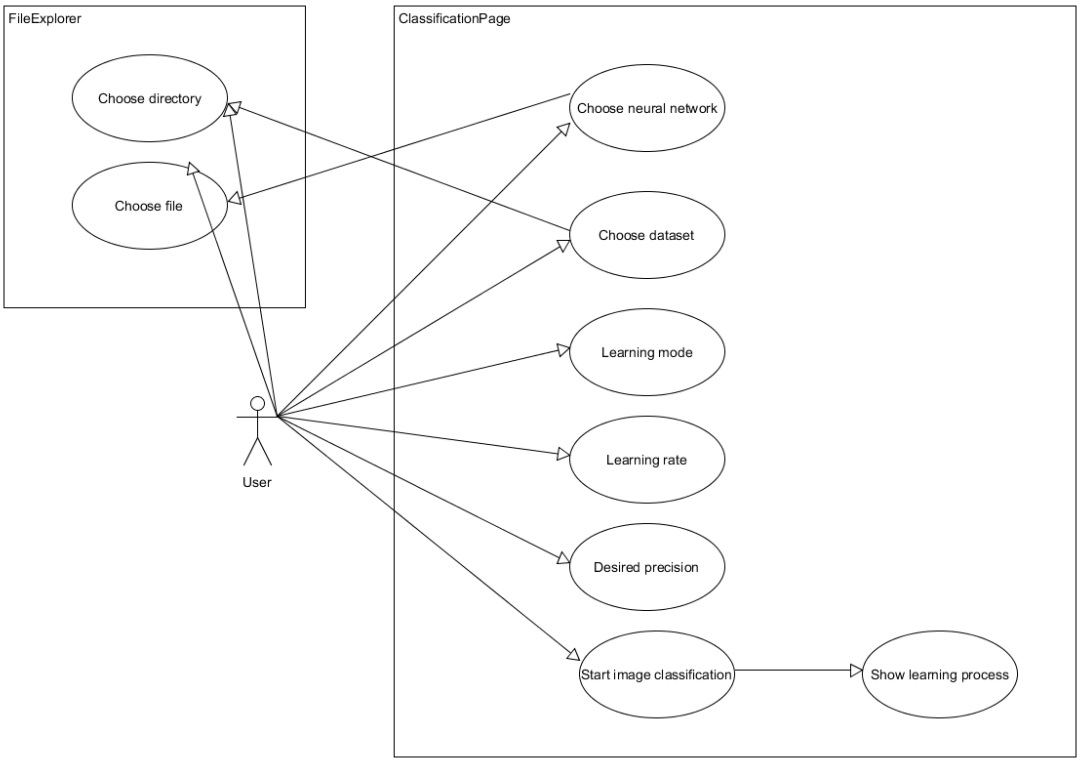

Abbildung 2: Usecase of the trainingspage

12 Stage responsibilities

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

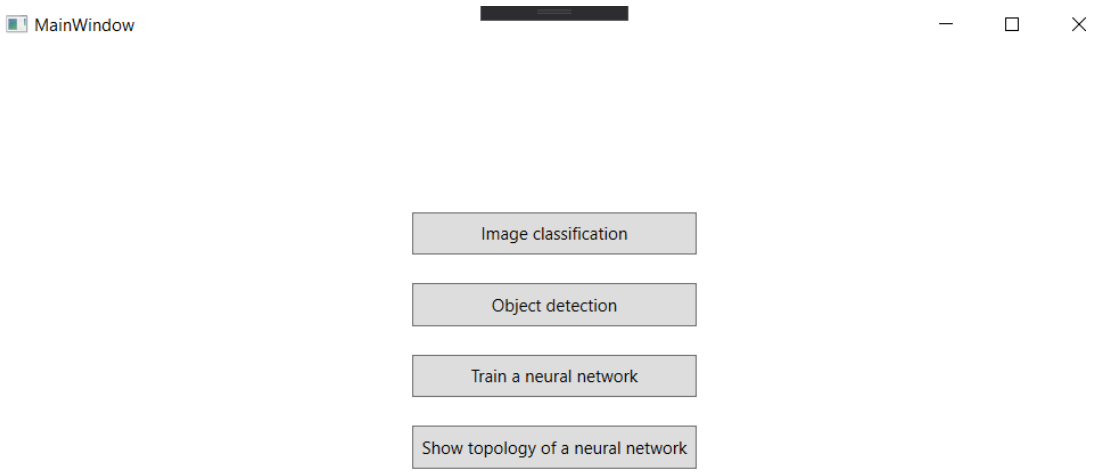

Abbildung 3: Main page of our software

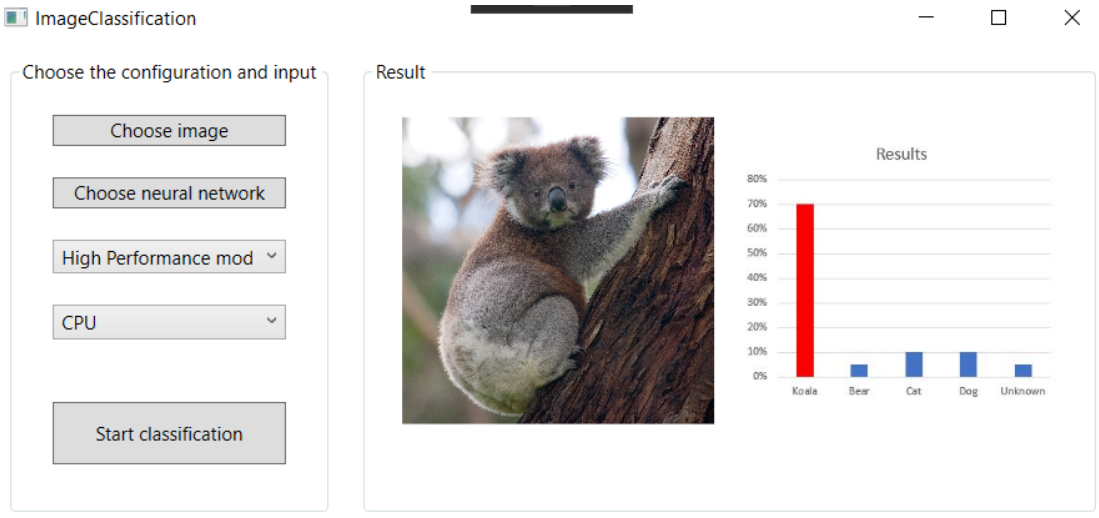

Abbildung 4: Image classification page of our software

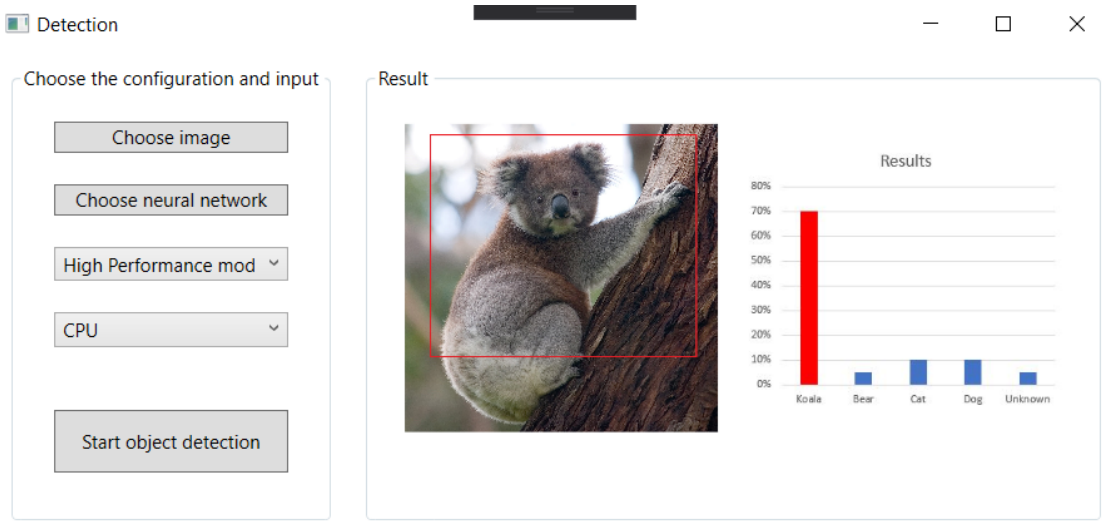

Abbildung 5: Object detection page of our software

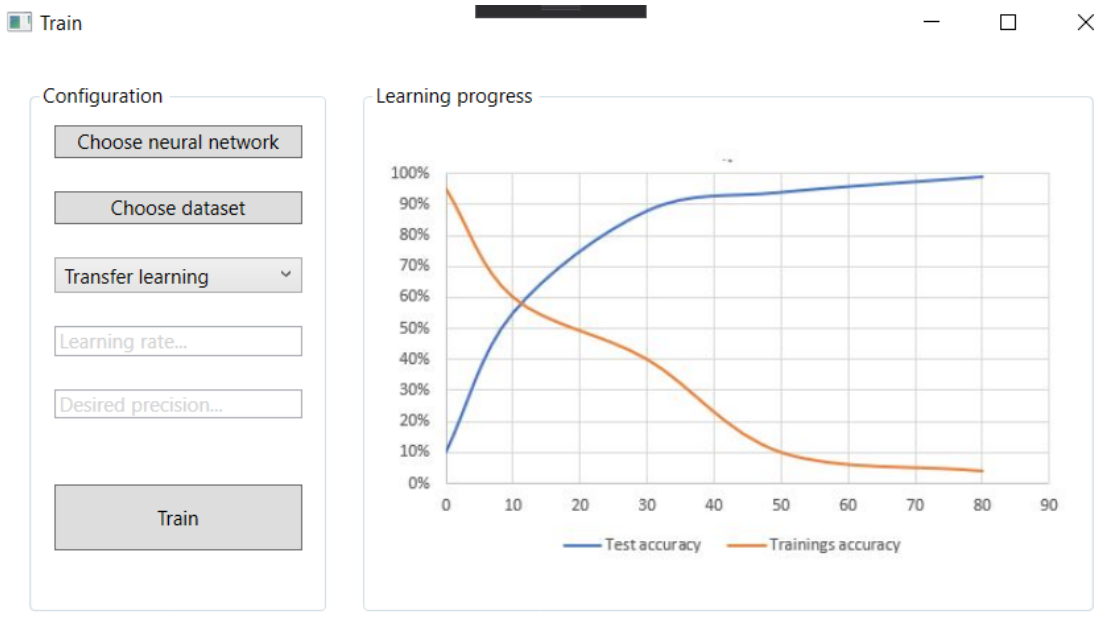

Abbildung 6: Training page of our software

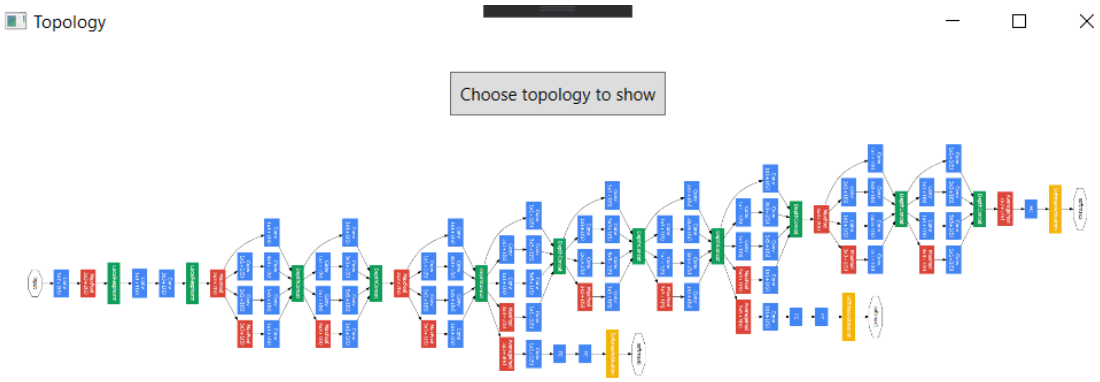

Abbildung 7: Page which shows the topology of a selected NN of our software