Neural Network based Image Classification System on Heterogeneous Platforms

17. November 2019

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 Acceptance criteria

4.1 Must

MAC010 : Image classification
MAC020 : Running NN on heterogenous platforms, CPU and FPGA
MAC030 : Different operating modes
MAC040 : GUI for interacting with software
MAC050 : Performance/power prediction

4.2 Can

KAC060 : Training a nn for classification
KAC070 : Illustration of a topology of the nn
KAC080 : Object detection
KAC090 : Choosing between different models
KAC100 : Creating new models
KAC110 : Voting of multiple nn
KAC120 : Using video for classification
KAC130 : Using camera for classification input
KAC140 : Running NN on GPU

5 Functional Requirements Must

MFR010 : Use neural network for image classification
MFR011 : Deploy pre-trained nn with the corresponding layers
MFR012 : Reading and parsing nn config/weight file
MFR020 : Have high perfomance mode
MFR021 : Have low power consumption mode
MFR022 : Have high energy efficiency mode

MFR023 : Calculator for power consumption
MFR024 : Calculator for performance

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.

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

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.

KFR100 **Choosing between different models**
Tested with: Implements:
The user has the option to decide bewteen different NN models for the calculations. For example AlexNet or GoogleNet.

KFR110 **Train NN for classification of imageset (with transfer learning)**
Tested with: Implements:
The user is able to input a labeled imageset of a specific class. That class should automatically be trained and included in future classificatios.

KFR060 **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.

KFR112 **Choosing/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.

KFR032 **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.

KFR113 **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.

KFR114 **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.

KFR120 **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.

KFR130 **Object detection algorithm**
Tested with: Implements
Not only should the program classify an image it should also detect the objects/ classes in the picture.

KFR131 **Showing detected object**
Tested with: Implements
The found objects should be marked by a box and shown to the user.

KFR132 : Choosing between detection and classification mode
KFR140 : Creating new topology
KFR150 : Choosing between training and interference mode
KFR160 : Choosing video in format .avi
KFR161 : Apply classification for a certain amount of frames
KFR170 : Connect with camera
KFR171 : Receive video stream from camera
KFR180 : Detecting object
KFR181 : Drawing bounding box

7 Productdata

PD010 : Images for classification
PD020 : Labeled image set for training
PD030 : Config/weight file of pretrained model

8 Demarcation

D010 **No real time requirements**
There is no hard time limit the neural network has for the classification, but it should be optimized to a certain degree that the program works smoothly and has a nice feeling.

D010 **No mobile support**
The main focus of this project is on the classification that is only done on one computer with the specific hardware components. Mobile support is not needed for this task and should not be added in our project.

D010 **No neural network size optimisation**
The program should work with given already existing neural networks. It has been shown in other instances that it is possible to reduce the network size with only minmal performance loss. This should not be the goal of the project and should not be included as it is to much.

D010 **No low-level (Assembler) optimization**
For our purposes a general optimisation to get a reasonable calculation time is enough. No lower level optimisation is needed.

9 Non-functional requirements

D010	**Project size** The project should have around 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

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	**Case 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	**Case 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	**Case 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	**Input:** The user chooses an elephant, CPU as a platform and performance mode.
	Expected Output: Elephant.
T031	**Support FPGA for calculation**
T031.1	**Input:** The user chooses an elephant, FPGA as a platform and performance mode.
	Expected Output: Elephant.
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.
T040.2	**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 runs 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 drop-

T100 **Choosing NN model**

T070.1 **State:** The user is on the page for image classification
Action: The user clicks on the button for choosing the neural network model
Reaction: A dropdown with all availabe models is shown that the user can click on.

T080 **Training the NN**

T080.1 **State:** The user is on the main page
Action: The user clicks the button for learn
Reaction: The user is redirected to a new page with further options.

T080.2 **State:** The user is on the new page
Action: The user clicks the button „Choose dataset“

Reaction:A file explorer pops up where the user can choose only folders

T080.3 **State:** The user chose the dataset and chose all options: dataset, model, mode
Action: The user clicks the train button.
Reaction: The NN is trained 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

T090 **Reading dataset**

T025.1 **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.

T025.2 **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.

T090 **Saving the trained NN**

T090.1 **State:**
Action:
Reaction:

T090 **Use different hardware components for calculation**

T025.1 **Case 1: GPU** :
State:
Action:
Reaction:

T090 **Backpropagation**

T090.1 **State:**
Action:
Reaction:

T090 **Showing topology of a NN**

T090.1 **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

T090 **Changing parameters**

T090.1 **State:**
Action:

11 System models

11.1 Scenarios

11.2 Usecases

11.2.1 Seminarorganisation