Specification Document: NEUTERO (Neural Network based Image Classification/Detection on Heterogeneous Platforms)

Oguz Mutlu
Yavuz Karaca
Muhammed Rodi Düger
Junchi Li
Umut Sezen

Supervisors:
Christopher Münch
Soyed Tuhin Ahmed

ITEC Tahoori
Chair of Dependable Nano Computing

June 1, 2022

Contents

1 Foreword

In the conventional computer programming, the programmer tells the computer precisely what to do in small and strictly defined steps. In neural networks approach, however, the computer or more precisely an artificial neural network imitates in a sense the human brain to process a given input.

To achieve this, neural networks are "trained" with data. That is, the neural network receives a lot of data, which consists of example inputs (and for supervised learning additionally corresponding outputs). The neural network is then expected to learn from these data and finally to figure out its own solution to the problem at hand. For example, a neural network designed for recognizing cats in pictures is given at the beginning a lot of cat pictures as training data. Afterwards, the neural network is expected to "learn" how cats look like, in a sense in the same way as how a baby learns to recognize objects around her/him. In the end, the neural network trained with cat pictures is supposed to predict whether a given picture, which was not seen during training, contains cats or not.

In the last two decades, neural network based algorithms became more and more popular, being used in different areas, from autonomous driving and face/image recognition to natural language processing. Current image classification algorithms have even surpassed human accuracy in the last few years. It is now thus more important than ever to have an easy-to-use graphical computer program, which, given a neural network and an input, shows the result of the inference the neural network has done based on this input. With our project, we want to create exactly such a solution for scientists and researchers.

Our project, **Neutero**[1] is going to allow scientists and researches in the end to run their neural networks on different hardware components (hence the name "heterogeneous platforms") and see the result of the inference in an easy-to-use graphical interface. We hope to create a product, which will be helpful for scientists and researchers, who develop and/or use neural network based image classification/detection algorithms.

[1] The name **Neutero** is a combination of the words NEUral network and heTEROgeneous hardware

2 Target Definition

The target of **Neutero** is to give scientists and researchers the opportunity to see the output their trained neural network generates on an input they provide in a graphical user interface. At the same time, they are supposed to be able to select on which of the listed hardware components they want the inference to be run.

2.1 Must criteria

/MC10/

User is able to import their own neural network topology for image classification/detection.

/MC20/

The program includes at least one pre-trained neural network for image classification/detection.

/MC30/

User can select one neural network for inference.

/MC40/

User can pick the hardware component to run the inference on.

/MC50/

Supported hardware components are Central Processing Unit (CPU), Graphical Processing Unit (GPU) and Intel Movidius.

/MC60/

User can import an input image.

/MC70/

User can start the inference.

/MC80/

User can view the inference result.

/MC90/

The processing time is shown to the user after the inference is finished.

/MC100/

There is an error message if a missing hardware component is selected.

/MC110/

The program stores the last imported neural network topology until termination.

/MC120/

User can export results and diagnostic data.

/MC130/

User can remove an imported image.

2.2 Optional criteria

The "+" signs from 1 to 3 indicate importance of the criterion. ((+++): most important, (+): least important)

/OC10/

FPGA is also a supported hardware component. (++)

/OC20/

The inference can be aborted at any time. (+++)

/OC30/

While the inference is running, elapsed time is shown to the user.(+++)

/OC40/

User can import more than one image. (+++)

/OC41/

User can preview/list imported image(s). (+++)

/OC42/

User can reorder imported images. (+)

/OC50/

User can select more than one hardware component. (+++)

/OC60/

The program automatically detects if Intel Movidius is plugged in. (++)

/OC70/

There is an option to show comparison of different hardware components on comparative graphs. The program can be used as a benchmark tool. (+++)

/OC80/

Inference results are shown as a graph. (+++)

/OC90/

There is an option to run an inference with same arguments more than once with a single click. (++)

/OC100/

Some languages other than English are supported. (+++)

/OC110/

User can import more than one neural network topology and remove them. (+++)

2.3 Restrictive criteria

/RC10/

The program can only be run on the specified environment. (see section 4)

/RC20/

At most one inference instance is ran at a given time. An inference instance refers to a neural network running on a specified hardware inferring a set of images.

/RC30/

Only such neural network topology are supported that are compatible with in section 4 specified hardware.

3 Product Use

3.1 Scope of application

Neutero provides a tool for testing trained neural networks with different outputs. It will be used by researchers, who want to determine how their neural networks perform at producing inference results on various inputs and various hardware components.

3.2 Target audience

The target audience consists of people, who develop and/or use neural network based image classification/detection algorithms. (mainly scientists and researchers)

Prerequisites for the usage of the product are basic knowledge of how to use desktop applications on a computer and basic knowledge of English language.

3.3 Operating conditions

It should be possible to use the product on in section 4 specified Ubuntu computer. Internet connection is not required.

4 Product Environment

4.1 Hardware

The program is ran on the following hardware:

/HW10/ CPU
Model: Intel Xeon E5630

/HW20/ RAM
Capacity: 16 GB

/HW30/ GPU
Model: NVIDIA GeForce GT 710 with CUDA 11.4.4

/HW40/ Intel Movidius
Model: Intel Movidius Neural Compute Stick 2

/HW50/ FPGA
Model: Terasic DE1-SOC
(Support for FPGA is an optional criterion.)

4.2 Software

/SW10/ Operating system
Officially supported operating system is Ubuntu 20.04.

5 Functional Requirements

5.1 User functions

/FR-U-10/ List neural networks

There will be a "Choose a neural network" button. Clicking on it will show all available neural network as a drop down list. This includes default and from user imported neural network topology.

/FR-U-20/ Select a neural network

Select a neural network from the list via clicking on it.

/FR-U-30/ Import a neural network topology

Selecting "import neural network" option opens a window for file selection from user's device.

/FR-U-40/ List hardware components

There will be a "Choose hardware" button. Clicking on it will show all hardware components as a drop down list.

/FR-U-50/ Select hardware component

Select a hardware component from the list via clicking on it. Initiates /FR-A-30/.

/FR-U-60/ Import an input image

There will be an "Import image" button. Clicking on it will open a window for file selection from user's device.

/FR-U-70/ Remove input image(s)

There will be a "Remove image" option for every imported image. Clicking on it will delete the selected image.

/FR-U-80/ Start the inference

There will be a "Start the inference" button. Clicking on it will start the inference with the given arguments

/FR-U-90/ Window controls

Clicking on "×" terminates the program, clicking on "□" maximizes the window and "_" minimizes it. Applicable to all windows except error messages. Error messages only have the "×" button as a window control button.

/FR-U-100/ View results

Clicking on "View results" button will open pop-up window, where inference results are shown together with diagnostic data. If more than one image or more than one hardware component were selected for inference, user can scroll down the window to see all the results.

/FR-U-110/ Export results and diagnostic data

There will be an "Export results" button. Clicking on it will save inference results and diagnostic data on local device.

/FR-U-120/ Abort the inference (optional)

There will be an "Abort" button, while the inference is running. Clicking on it will abort the inference and the starting screen is shown.

/FR-U-130/ Multiple input images (optional)

During image selection, selecting multiple images will result in batch processing.

/FR-U-140/ Reorder input images (optional)

Dragging and dropping input images on the preview grid, changes their order in which they will be processed.

/FR-U-150/ Multiple hardware components (optional)

Selecting more than one hardware component results in batch processing.

/FR-U-160/ Benchmarking (optional)

Clicking on the "Benchmark" tab on top will change to another tab. In this tab the user can see graphs related to the hardware performance and compare them on program-generated graphs.

/FR-U-170/ Multiple inference runs with same arguments (optional)

After setting all the arguments, user can specify how many times the inference will be ran with the given arguments.

/FR-U-180/ Changing the language of the user interface (optional)

Clicking the Language button at the top right will open the language drop-down menu, in which different languages for the entire user interface can be chosen.

/FR-U-190/ Remove neural network (optional)

There will be a "Remove neural network" option for every neural network. Clicking on it will delete the topology from the list.

5.2 Automatic functions

/FR-A-10/ Update user interface after each selection

Each selection is visible in real-time on user interface. According error and warning messages are shown in run-time log if a selection is improper. (initiates /FR-A-30/.)

/FR-A-20/ Show image preview (optional)

Thumbnails of imported images are shown on a grid in the user interface.

/FR-A-30/ Check arguments

Arguments are checked for validity. Any conflict is shown as an error message.

/FR-A-40/ Disable start

"Start the inference" button is disabled until all arguments are selected.

/FR-A-50/ Show loading animation

Loading animation will be shown after starting the inference.

/FR-A-60/ Show completion message

After the inference is complete, a completion message is shown and loading animation disappears.

/FR-A-70/ Run-time log section

The program provides a run-time log section for detailed information about current events. (e.g. error/warning messages, successful selections, current elapsed time) The past events can be seen by scrolling up. The events shown in the run-time log section are also written to an external file.

/FR-A-80/ Keep lastly used arguments

After an inference is finished, selections for neural network topology, input image(s) and hardware component(s) are preserved.

6 Product Data

/PD10/ Input image(s)

The neural network receives the image(s) as input.
Supported formats: JPEG, JPG, PNG.
The height and width of import image(s) should be bigger than the minimum the selected neural network supports
Max. 100 images per inference run

/PD20/ Imported neural network(s)

Imported neural network topology should be compatible with the specified hardware.
Supported format: Open Neural Network eXchange (ONNX)

/PD30/ Inference results and diagnostic data

- inference result containing confidence percentages
- latency
- hardware information
- identifier (filename or name of the predefined network) of the neural network

/PD40/ Runtime log file

Contains detailed information about past events. Specified in /FR-A-70/.

/PD50/ Translation Files

The program uses translation files as text resources to display GUI texts in a language specified by the user.
Supported format: Compiled Qt Linguist File (.qm file)

7 System Models

7.1 Scenarios

7.1.1 Scenario 1: Usage by an inexperienced user

Description:

Michael is a computer science student at Massachusetts Institute of Technology and has interest in neural networks and machine learning. He hears about **Neutero** developed from his fellow students at Karlsruhe Institute of Technology and wants to try it. He goes to the lab opens the program on a supported computer. He remembers that he has no neural network at hand to import into the application but thanks to the design, importing a neural network is not necessary. He clicks on the button "Choose neural network" and default neural networks are shown in drop down menu. He selects randomly one of them by clicking on it. User interface is updated and identifier of the neural network is visible. Then he clicks on "Import an image" button and a window for file selection opens. He selects a cat image which was already saved in the computer. User interface is updated once again. Then he clicks on "Choose a hardware" button and chooses "CPU" from the drop down menu. User interface is updated with this selection. At last he clicks on "Start inference". The inference starts running on CPU. "Start inference" is replaced by "Running..." and nothing is clickable. After a while a completition message is shown. He clicks on "View results" and results are opened in a separate window. He finds out that the image is recognized by neural network as a cat with 97% confidence. He closes the "Results" window as well as the main window.

7.1.2 Scenario 2: Usage by a scientist

Description:

Ada is a computer scientist working on neural network based algorithms for image recognition. She has already a trained neural network available in ONNX format. She has the supported hardware at her office and **Neutero** is already installed in her Xubuntu system. She opens the application and clicks on "Import a neural network" button. She selects her neural network which was already saved in her computer. Then she selects the hardware component as "Intel Movidius" to find out how well her neural network runs on this Movidius stick she recently bought. She clicks on "Import image" button and selects multiple images at once. Then she starts the inference. This results in batch processing and after a while inference results are shown in a single separate window. She clicks on "Export results and diagnostics" and a window opens to select the saving location. She selects the folder, into which she wants to save results and everything is exported. She will then analyze the results and diagnostic data to find out how well her neural network performs on different images.

7.2 Use case diagrams

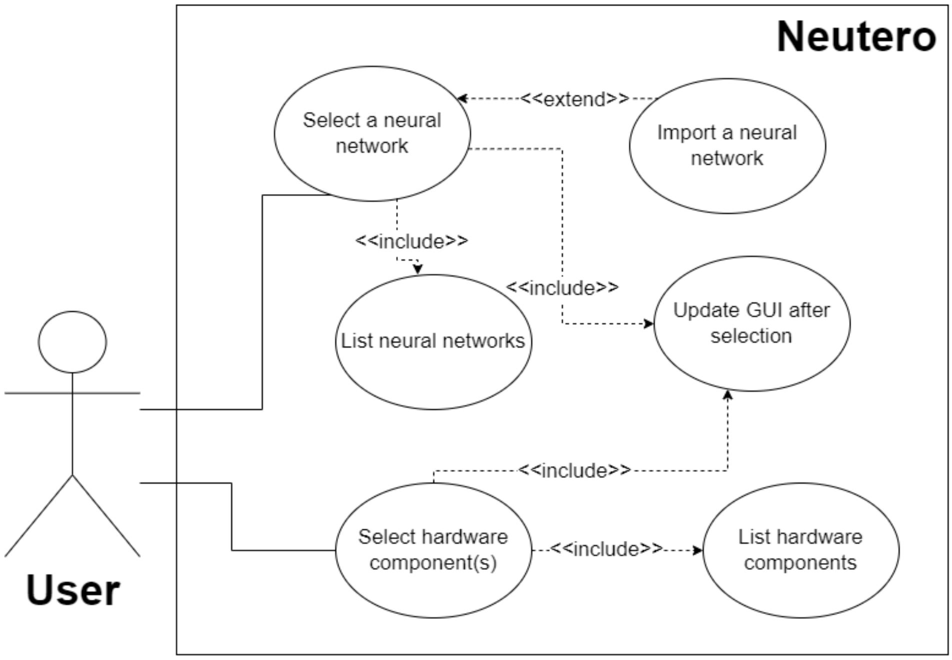

Figure 1: Use case diagram for selecting hardware component and neural network

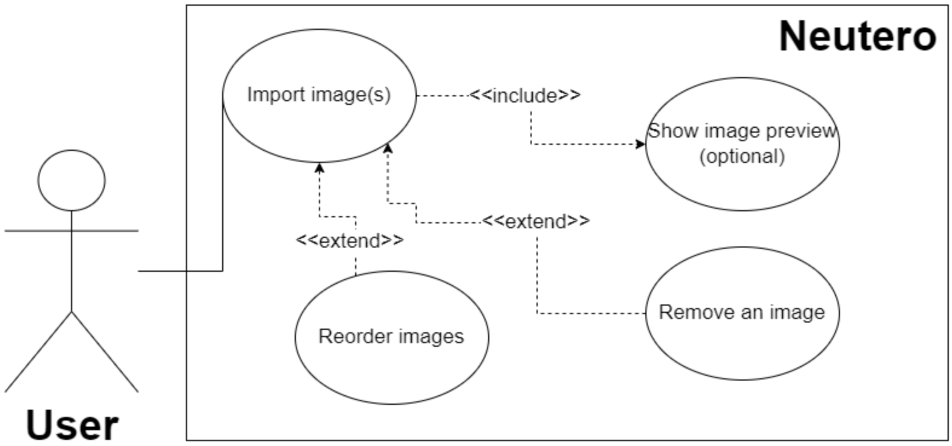

Figure 2: Use case diagram for importing images(s)

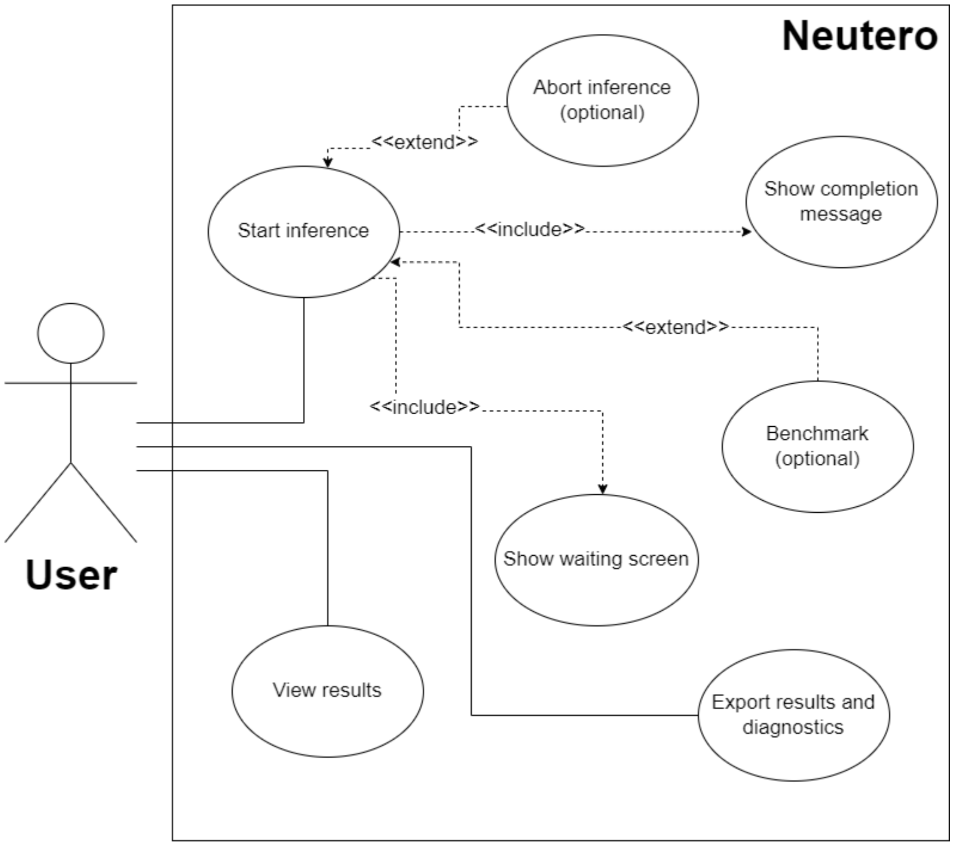

Figure 3: Use case diagram for starting inference and viewing results

7.3 Neutero as black box

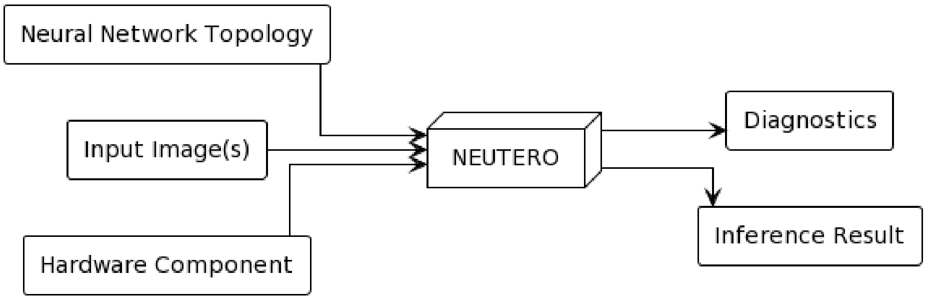

Figure 4: Diagram showing inputs and outputs of Neutero

7.4 Activity diagram

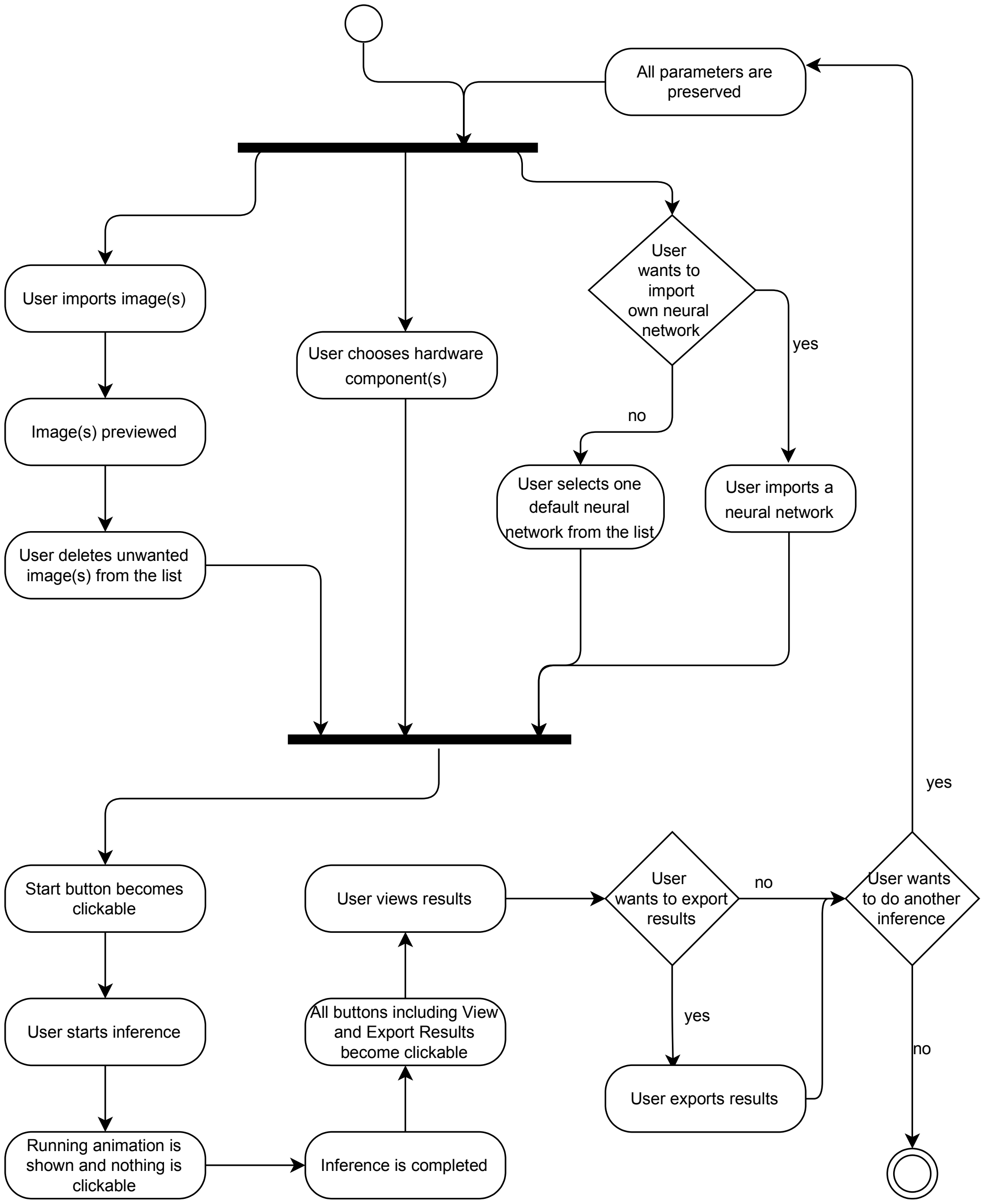

Figure 5: Activity Diagram

8 Nonfunctional Requirements

/NR10/ Object orientation

The software is to be developed according to object-oriented programming paradigm.

/NR20/ Model-View-Controller

The Model-View-Controller[2] software design pattern is to be used for the general structure of the software.

/NR30/ Language

The main language for user interface is English. Other languages may also be supported.

/NR40/ Program startup time

The startup will take at maximum 10 seconds.

/NR50/ Clarity of user interface

Graphical user interface should be user friendly, that is a user can use the interface intuitively.

/NR60/ Extendability

The program should be compatible for extra functions in the future.

---

[2] See `https://en.wikipedia.org/wiki/Model-view-controller`.

9 Graphical User Interface

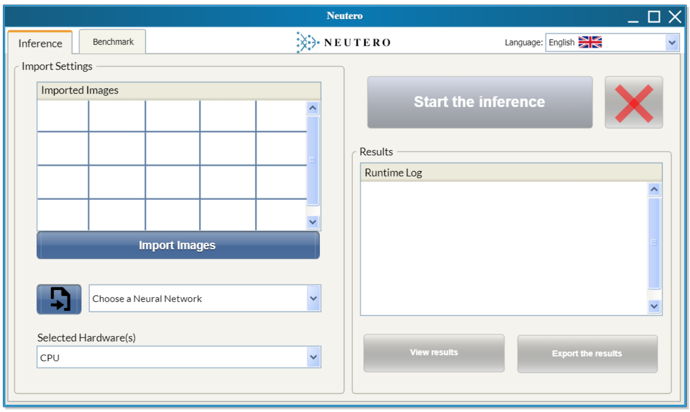

Figure 6: First start of the program

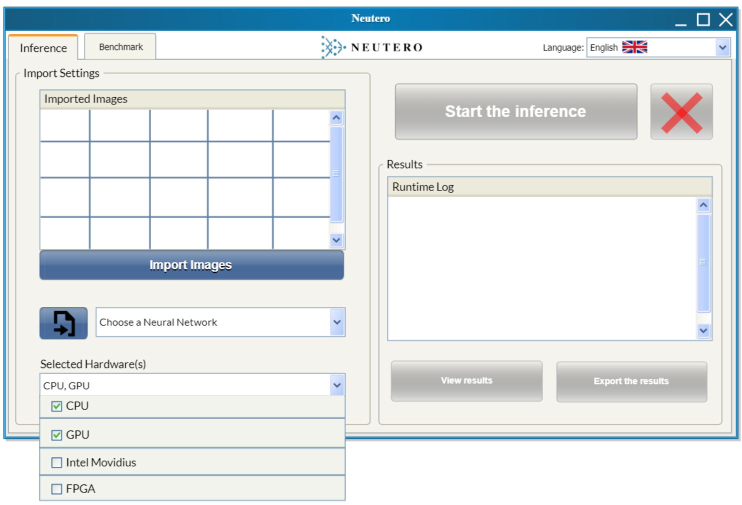

Figure 7: After clicking “Choose Hardware” button

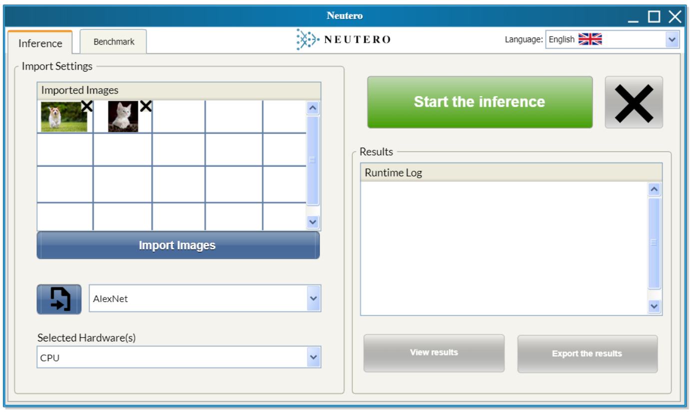

Figure 8: After importing images

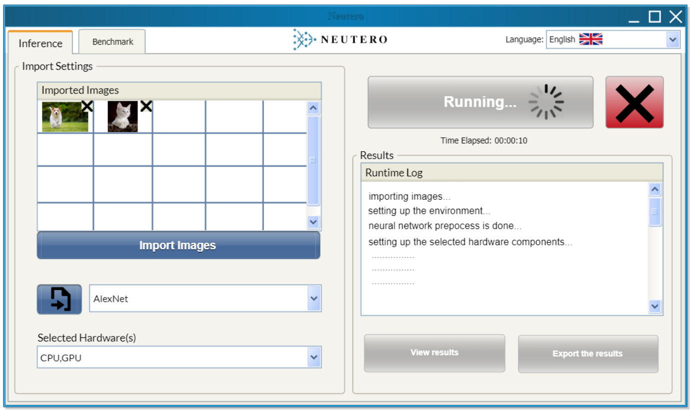

Figure 9: After clicking "Start the inference" button and selecting AlexNet, which is popular neural network topology, as the desired neural network

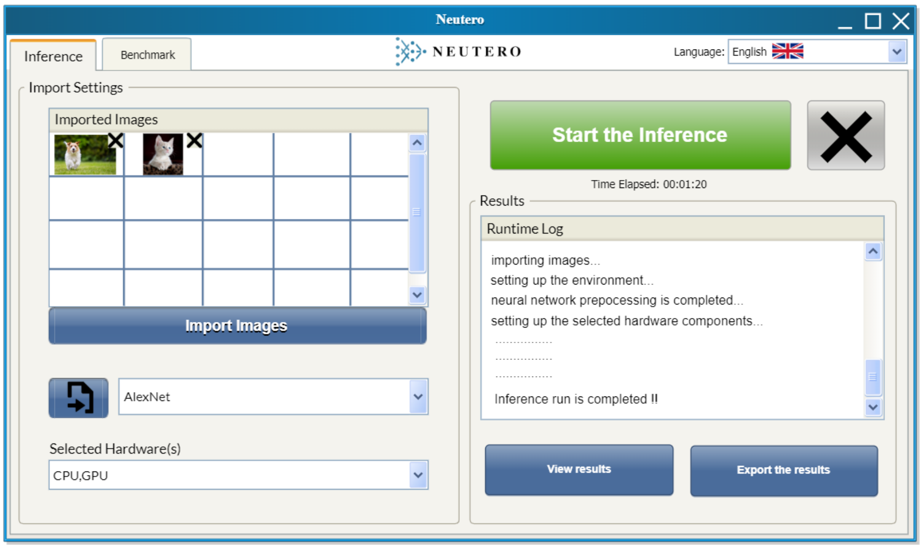

Figure 10: After successfully finished inference

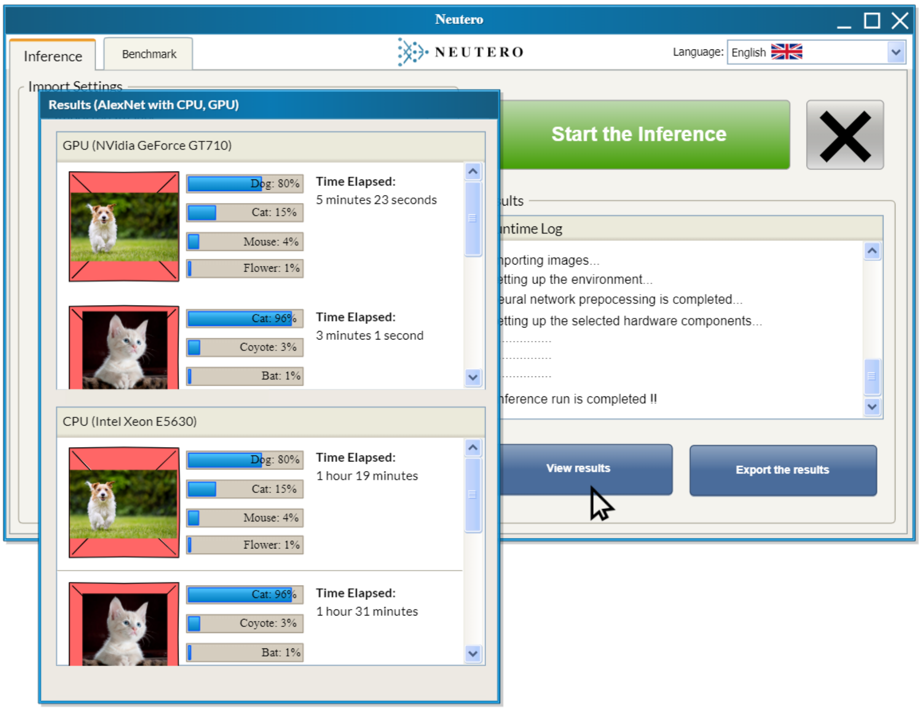

Figure 11: After clicking “View Results” button

10 Global Test Cases

10.1 Initialization Cases

At the start of the program all arguments would be in the "no selection" state. Without initializing all arguments an inference cannot be started.

/TC-INI-10/ Program Start

1. Launch the program.

Expected: A window specified in section 9 would pop up. Selection of hardware component and neural network would be "no selection" and no image is imported. The "Start" button would be disabled.

/TC-INI-20/ Initializing All Arguments

1. Launch the program.
2. Select an available hardware component.
3. Select a valid neural network.
4. Import a valid image.

Expected: The selected hardware component and neural network would be highlighted as selection, and the preview of the configured image would be shown. The "Start" button would also turn enabled after all parameters are successfully selected.

/TC-INI-30/ Removing an Imported Image

1. Repeat /TC-INI-20/.
2. Remove the image imported in Step 1.4.

Expected: The image would be removed from the list and would not be use for inference (unless imported again). The "Start" button would turn disabled.

10.2 Inference Cases

Test cases to check the basic functionality of running an inference.

/TC-INF-10/ Normal Inference

1. Repeat /TC-INI-20/.
2. Click on the "Start" button.
3. Wait and receive a valid result.

Expected: After starting the inference the "Start" button would turn into a loading animation, and all buttons would be disabled. After completion, the loading animation would be replaced with the completion message. All buttons would be enabled again. The result may be viewed (see /TC-INF-30/) or exported (see /TC-INF-40/).

/TC-INF-20/ Inference with Own Neural Network

1. Launch the program.
2. Select an available hardware component.
3. Import a valid neural network.
4. Select the imported neural network.
5. Import a valid image.
6. Click on the "Start" button.
7. Wait and receive a valid result.

Expected: The visual feedback would be the same as that of /TC-INF-10/. The result should match the imported network.

/TC-INF-30/ Viewing Inference Results

1. Repeat /TC-INF-10/ or /TC-INF-20/.
2. Click on the "View results" button.

Expected: The result and the diagnostic data would be shown in a pop-up window.

/TC-INF-40/ Exporting Inference Results

1. Repeat /TC-INF-10/ or /TC-INF-20/.
2. Click on the "Export" button.

Expected: The result and the diagnostic data would be saved to the local storage.

/TC-INF-50/ Keeping Used Arguments For Later Inferences

1. Repeat /TC-INF-10/ or /TC-INF-20/.
2. Click on the "Start" button again.

Expected: The inference would run with the same parameters so the result would be the same as that of Step 1.

10.3 Error Cases

Test cases to check the capability of handling with inevitable errors and failures.

/TC-ERR-10/ Importing an Invalid Neural Network

1. Launch the program.
2. Import an invalid neural network.

Expected: An error message would be shown to specify the error. The neural network selection would be kept unchanged.

/TC-ERR-20/ Selecting a Missing Hardware Component

1. Launch the program.
2. Select a missing hardware component.

Expected: An error message would be shown to specify the error. The hardware component selection would be kept unchanged.

/TC-ERR-30/ Importing an Invalid Image with no Image Imported Before

1. Launch the program.
2. Select an available hardware component.
3. Select a valid neural network.
4. Import an invalid image.

Expected: An error message would be shown to specify the failure of loading the image. The "Start" button would be kept disabled.

/TC-ERR-40/ Importing an Invalid Image with Image Imported Before

1. Launch the program.
2. Import a valid image.
3. Import an invalid image.

Expected: An error message would be shown to specify the failure of loading the image.

10.4 Optional Cases

Test cases to check the correctness of any implementation of an optional criterion.

/TC-OPT-10/ Aborting an Inference

1. Repeat /TC-INI-20/.
2. Click on the "Start" button.
3. Before completion, click on the "Abort" button.

Expected: After starting the inference the "Start" button would turn into a loading animation, and all buttons except the "Abort" button would be disabled. After clicking on the "Abort" button the inference would be aborted. The "Abort" button would be disabled, and all other buttons would be enabled again. User may view or export the diagnostic data with the "View results" or "Export" button.

/TC-OPT-20/ Automatic Detection of Intel Movidius

1. Launch the program.
2. Insert the Intel Movidius.
3. Remove the Intel Movidius.

Expected: Intel Movidius would be available for selection in the hardware component list after step 2, and would be unavailable after step 3.

/TC-OPT-30/ Inference with Multiple Images

1. Repeat /TC-INI-20/.
2. Import one or more images.

3. Click on the "Start" button.
4. Wait and receive results.

Expected: Multiple inferences would be run on the same network with each image one by one. The visual feedback would be the same as /TC-INF-10/. In the "View results" window the results and the diagnostic data would be shown as a list.

/TC-OPT-40/ Inference with Multiple Hardware Components Selected

For this optional criterion user would be allowed to select more than one hardware component.

1. Repeat /TC-INI-20/.
2. Select further to choose multiple hardware components.
3. Click on the "Start" button.
4. Wait and receive results.

Expected: The same inference, with the same network and the input image, would be run for multiple times once on each of the chosen hardware component. The results are expected to be identical but the diagnostic data would be different. The visual feedback would be the same as /TC-INF-10/. In the "View results" window the results and the diagnostic data would be shown as a list.

/TC-OPT-50/ Showing Benchmark Statistics

1. Repeat /TC-OPT-40/.
2. Switch to the "Benchmark" tab.

Expected: The performances on difference hardware components would be visualized as graphs for visual comparison and shown to the user.

/TC-OPT-60/ Switching User Interface Language.

1. Launch the program.
2. Change the language setting.

Expected: The user interface would be updated to display texts in the specified language right after Step 2.

11 Development Environment

The following software are to be used while developing our product:

- *Git* for version control
- *LaTeX* as software system for document preparation
- *Overleaf* for creating, editing and compiling TeX documents
- *Pencil Project* for designing GUI mockups
- *C++* as the main implementation language
- *Qt* for creating the user interface
- *Qt Creator*, *Visual Studio Code* and *VIM* as IDE and Text Editor
- *diagrams.net*, *draw.io* and *PlantUML* for UML diagrams
- *DocTest* for unit testing
- *Squish for Qt* for GUI testing
- *Qmlbench* for performance testing

Glossary

argument user-changeable values of the program. Arguments are given before execution.

batch processing running multiple jobs automatically with a single command. In Neutero this refers to processing multiple images within a single run in an optimised way.

CPU Central Processing Unit. electronic circuitry that executes instructions comprising a computer program.

diagnostic data data from the program which can be used for diagnosing program procedure. Such as but not limited to: time elapsed, model of the hardware used. Specified in /PD30/.

GPU Graphics Processing Unit. electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images.

image classification/detection the process of identifying an image according to predefined classes and rules.

inference the process of using the knowledge of a trained neural network to predict the result based on the given input.

inference result it contains predictions made by the neural network for the input image and corresponding confidence percentage for each prediction..

input image image that is given to neural network for inference.

Intel Movidius a Vision Processing Unit (VPU) which enables demanding computer vision and edge AI workloads with efficiency.

latency time elapsed during inference run.

neural network a computational learning system inspired by biological neural networks that uses a network of functions to understand and translate a data input of one form into a desired output.

ONNX stands for *Open Neural Network eXchange*, an open standard for representing machine learning models. See `https://en.wikipedia.org/wiki/Open_Neural_Network_Exchange`.

supervised learning a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately.

the program used for in this document specified software, **Neutero**.

trained neural network a neural network that has been went through a process called "training" to show accurate results.

user person who uses our product.

user interface the means of interaction between user and program. In this document, a Qt based graphical interface is implied.