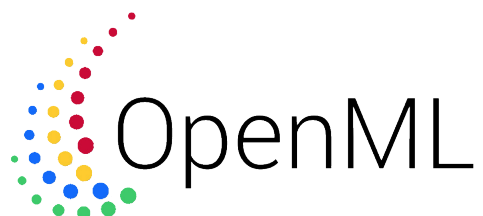


July 3, 2019

Software User Manual

Version 1.0.0



Members

Andrei Danila | 1034559
Bogdan Enache | 1035066
Gergana Goncheva | 1037010
Loïc Alexander Hiji | 1002745
Adrian-Stefan Mares | 0993873
Veselin Minev | 1014541
Thanh-Dat Nguyen | 1036672
Antoine Labasse Lutou Nijhuis | 1016657
Claudiu-Teodor Nohai | 1038442
Dragos Mihai Serban | 1033859
Tsvetan Zahariev | 1035269
Sonya Zarkova | 1034611

Project managers

Yuxuan Zhang
Stefan Tanja

Supervisor

Erik Luit
Ion Barosan

Customer

Joaquin Vanschoren

Abstract

This document contains the software user manual for the Deep Learning (DL) Extension Library for OpenML, which is developed by the team OpenML Support Squad. The purpose of this Software User Manual (SUM) is to be a source of information regarding the manners in which the DL Extension Library can be used. This document complies with the Software Engineering Standard, defined by the European Space Agency (ESA) [1].

Contents

1	Introduction	5
1.1	Intended Readership	5
1.2	Applicability	5
1.3	Purpose	5
1.4	How to use this Document	5
1.5	Related Documents	6
1.6	Conventions	6
1.7	Problem Reporting	6
1.8	List of references	6
2	Overview	8
2.1	The difference between OpenML Python Application Programming Interface (API) and DL Extension Library	8
3	Application Tutorials	10
3.1	Get a task from OpenML	10
3.2	Run a model on a task	10
3.3	Get an OpenML flow from OpenML	11
3.4	Run an OpenML flow on a task	11
3.5	Publish a run	12
3.6	Visualize an OpenML flow	13
3.7	Visualize a run	14
4	References	17
4.1	Initial menu	17
4.2	Run visualization	18
4.3	Drop-down bar	19
4.4	Metric Selection	19
4.5	Flow Visualization	20
A	Error Messages and Recovery	22
B	Glossary	23

Document Status Sheet

Document Title: Software User Manual

Identification: SUM/1.0.0

Version: 1.0.0

Authors: A.Danila, B.Enache, G.Goncheva, L.A.Hijl, A.Mares, V.Minev, T.Nguyen, A.L.L.Nijhuis, C.Nohai, D.M.Serban, T.Zahariev, S.Zarkova

Document History

Version	Date	Author(s)	Summary
0.0.1	25-04-2019	T. Zahariev	Created initial document
0.0.2	30-05-2019	L.A. Hijl	Added chapters and layout
0.0.3	11-06-2019	L.A. Hijl	Started work on chapters
0.0.4	12-06-2019	S.Zarkova	Added Abstarct, Introduction and Overview
0.0.5	24-06-2019	All authors	Added chapter 3, 4 and appendices
0.1.0	28-06-2019	All authors	Fixed feedback from supervisor
1.0.0	01-07-2019	All authors	Final edit

1 | Introduction

1.1 Intended Readership

There are various ways to categorize the users of the DL Extension Library i.e. based on their interests, level of experience and contribution to the platform. The vast majority, if not all of them, are generally programmers, data analysts or coding enthusiasts.

Since OpenML operates in the field of machine learning, it is expected that its regular users possess at least some basic knowledge in this area and a fair amount of programming skills. Furthermore, "OpenML offers a range of APIs to download and upload OpenML datasets, tasks, run algorithms on them, and share the results" [2]. The version from June 2019 provides those activities via a REST API, Python API, R API, Java API and .NET API. Hence, in order for users to be able to utilize the platform, they should be able to work with at least one of the aforementioned APIs and know to program in the corresponding language.

1.2 Applicability

Since DL Extension Library is an integrated part of the OpenML Python API, it does not have a native version. This document is then applicable for version 0.80 of OpenML on which the development for DL Extension Library started.

1.3 Purpose

The purpose of the DL Extension Library is to provide conversion of deep learning models, written in Keras, PyTorch and Apache MXNet (MXNet), and the model specifications Open Neural Network eXchange (ONNX) into a format supported by the OpenML platform.

The purpose of this software user manual is to provide descriptions for the user on how to successfully and correctly utilize the DL Extension Library.

1.4 How to use this Document

This software user manual is to be used as a guide for correctly operating with the DL Extension Library in OpenML. Furthermore, the document is structured in the way described below.

Chapter 2 contains an explanation of the differences between OpenML and DL Extension Library.

Chapter 3 contains a tutorial for each session, including the goal of the session, list of precautions to be taken, the procedures with step-by-step descriptions and possible errors.

Chapter 4 contains the references for each operation, including functional and formal descriptions, warnings, examples, possible errors, and related operations. Lastly, there are three appendices pages - error messages and recoveries, glossary, and index.

Users should refer to the corresponding sections for information based on the actions they want to perform.

1.5 Related Documents

The closest related and most relevant document to this SUM is the User Requirements Document (URD) of project "Sharing deep learning models" [3]. The requirements noted in the latter document specify which activities would be present in the final product, whereas this manual provides descriptive steps on performing those actions.

1.6 Conventions

Steps marked as optional could be skipped if the user decides to. Furthermore, actions which are not in the scope of our project but are related to the tutorials are also indicated.

1.7 Problem Reporting

The DL Extension Library can throw errors in regard to the type of task used in runs for each library. For instance, if the user decides to run an unsupported task, it will raise a value error of the task. Attempting to run a clustering task on a Keras model, would not be accepted, since currently the support for Keras covers only classification and regression tasks. The same statement holds for the rest of the libraries with their corresponding task support.

Furthermore, if the user tries to re-instantiate a flow which is not of the same type as the currently imported extension, the system will raise a value error.

A problem typical only for ONNX is in regard to the ONNX deserialization. Due to updates of the ONNX library itself, newer version's data types might raise a value error. The user then is encouraged by a runtime error to downgrade ONNX to its 1.2.1 version.

Finally, an issue, concerning only PyTorch, regards the conversion (serialization) of PyTorch models. If the user uses custom layers, which are not provided by PyTorch itself, the system will raise a value error.

1.8 List of references

- [1] Software Standardisation and Control. *ESA software engineering standards*. 1991.
- [2] OpenML. *OpenML Documentation*. URL: <https://docs.openml.org/> (visited on 09/05/2019).
- [3] OpenML Support Squad. *DL Extension Library, URD User Requirement Document version 1.0.0*.
- [4] OpenML Support Squad. *DL Extension Library, Software Requirements Document (SRD) Software Requirement Document version 1.0.0*.
- [5] Wikipedia. *Apache MXNet*. URL: https://en.wikipedia.org/wiki/Apache_MXNet (visited on 04/26/2019).
- [6] Kyle Kelley. *Introducing Dash*. URL: <https://medium.com/@plotlygraphs/introducing-dash-5ecf7191b503> (visited on 06/05/2019).

- [7] Keras. *Keras homepage*. URL: <https://keras.io/> (visited on 09/05/2019).
- [8] Wikipedia. *Artificial Neural Network*. URL: https://en.wikipedia.org/wiki/Artificial_neural_network (visited on 04/30/2019).
- [9] PyTorch. *PyTorch homepage*. URL: <https://pytorch.org/> (visited on 09/05/2019).

2 | Overview

This document contains a number of tutorials for the usage of OpenML and specifically zooms in on functionality regarding the DL Extension Library. These tutorials are described in Chapter 3 and have the following order:

1. Get a task from OpenML
2. Run a model on a task
3. Get a OpenML flow from OpenML
4. Run a OpenML flow on a task
5. Publish a run
6. Visualize a OpenML flow
7. Visualize a run

2.1 The difference between OpenML Python API and DL Extension Library

As already explained in the previous documents, the project "Sharing deep learning models" consists of creating a new extension for the already existent OpenML Python API library. This newly implemented extension is called DL Extension Library. The user interaction with the OpenML platform is done by directly calling functions from the OpenML Python API library. This library has to be imported by the user in their python IDE using the function: `import openml`.

The user does not interact directly with the DL Extension Library, they only interact with the OpenML Python API. All the functions from the DL Extension Library are called by the OpenML Python API. Everything implemented in the DL Extension Library is the responsibility of the OpenML Support Squad and will be documented in the following sections. Everything outside of the DL Extension Library will only be mentioned if they are relevant but not documented since it is not in the scope of project "Sharing deep learning models". Mentioning functions that are outside of the scope of project "Sharing deep learning models" is required because, as explained previously, they represent the only way through which the user interacts.

NOTE: since the DL Extension Library is in essence a Python library, the user can interact directly with all the functions implemented inside it. This is against the normal usage and should be avoided. Furthermore, the tutorials will contain low level details such as function names and parameters. This is because the DL Extension Library is used only in code and there is no interface. The user can make use of DL Extension Library only through code.

Table 2.2 shows the functions that the user invokes directly through OpenML Python API. Some of these functions call other functions, implemented in the DL Extension Library. In the last row of the table, the notation `run*` denotes the run returned by the `run_model_on_fold(model, task)` method.

Tutorials mentioned in upcoming section		
Tutorial name	Function called directly by user in OpenML Python API (team not responsible)	Functions from DL Extension Library used (team responsible)
Get a task	<code>get_task(id)</code>	-
Run a model on a task	<code>run_model_on_task(model, task)</code>	<code>model_to_flow(model)</code> , <code>run_model_on_fold(model, task)</code>
Get a flow	<code>get_flow(id, True)</code>	<code>flow_to_model(flow)</code>
Run a flow on a task	<code>run_flow_on_task(flow)</code>	<code>flow_to_model(flow)</code> , <code>run_model_on_fold(model, task)</code>
Publishing	<code>run*.publish()</code>	<code>run_model_on_fold(model, task)</code> , <code>model_to_flow(model)</code>

Table 2.2: Table to test captions and labels

3 | Application Tutorials

3.1 Get a task from OpenML

Functional description

This tutorial describes how the user can get a task from the OpenML platform by using the OpenML Python API.

Preconditions

1. The user should have already obtained an authentication token from the platform.
2. The task already exists.
3. OpenML is already installed and imported.

Cautions and warnings

None.

Procedures

1. Open a project in a Python IDE. (outside of the scope of the project)
2. Find the id of the task you want on the website. (outside of the scope of the project)
3. Get a task by calling `openml.task.get_task(id)`. (outside of the scope of the project)

Likely errors

Termination of the process if the OpenML server is down.

3.2 Run a model on a task

Functional description

This tutorial describes how the user can run a model on a task from the OpenML platform by using the OpenML Python API. The model must be implemented in any of the supported frameworks (Keras, PyTorch, MXNet) or specifications (ONNX).

Preconditions

1. OpenML is already installed and imported.
2. The appropriate deep learning library in which the model is created (as Keras, MXNet, PyTorch or ONNX in case of specification) is installed and imported.
3. The corresponding extension for the deep learning library used for the model is installed (for example `openml.extensions.keras` if Keras is used).
4. A task is already available.

Cautions and warnings

Not all tasks work on all models.

Procedures

1. Open a project in a Python IDE.(outside of the scope of the project)
2. Run the model on the task by calling `openml.runs.run_model_on_task(model, task)`

Likely errors

If the task is not of regression or classification type, the program returns an error.

3.3 Get an OpenML flow from OpenML

Functional description

This tutorial describes how the user can get a OpenML flow from the OpenML platform by using the OpenML Python API.

Preconditions

1. The user should have already obtained an authentication token from the platform.
2. The OpenML flow already exists.
3. OpenML is already installed and imported.
4. Have the appropriate deep learning library from which the OpenML flow was originally created installed (such as Keras, MXNet, PyTorch or ONNX).

Cautions and warnings

If you want to have the flow already converted into a model for the purposes of running it, have the second parameter of the function `get_flow()` as `True`.

Procedures

1. Open a project in a Python IDE. (outside of the scope of the project)
2. Find the id of the OpenML flow you want on the website. (outside of the scope of the project)
3. Get a flow by calling `openml.flows.get_flow(id, True)`.

Likely errors

Termination of the process if the OpenML server is down.

3.4 Run an OpenML flow on a task

Functional description

This tutorial describes how the user can run a flow on a task from the OpenML platform by using the OpenML Python API. The flow must represent a model in any of the supported frameworks (Keras, PyTorch, MXNet) or specifications (ONNX).

Preconditions

1. OpenML is already installed and imported.
2. Have the appropriate deep learning library from which the OpenML flow was originally created installed (such as Keras, MXNet, PyTorch or ONNX).
3. The cooresponding extension for the deep learning library used for the OpenML flow is installed (for example `openml.extensions.keras` if Keras is used).
4. A task is already available.
5. The model must already be instantiated in the OpenML flow object.

Cautions and warnings

Not all tasks work on all flows.

Procedures

1. Open a project in a Python IDE. (outside of the scope of the project)
2. Run the flow on the task by calling `openml.runs.run_flow_on_task(flow, task)`

Likely errors

If the task is not of regression or classification type, the program returns an error.

3.5 Publish a run

Functional description

This tutorial describes how the user can publish a run on the OpenML website.

Preconditions

1. OpenML is already installed and imported.
2. A run is already created.

Cautions and warnings

None.

Procedures

1. Open a project in a Python IDE. (outside of the scope of the project)
2. Publish the run by calling `run.publish()`, where `run` is the available run. (outside of the scope of the project)

Likely errors

Termination of the process if the OpenML server is down.

3.6 Visualize an OpenML flow

Functional description

This tutorial describes how the user can visualize an OpenML flow.

Preconditions

1. Dash is already installed.
2. OpenML is already installed.
3. Keras is installed.
4. PyTorch is installed.
5. ONNX is installed.
6. MXNet is installed.
7. Graphviz is installed.
8. Pydot is installed.
9. Onnxmltools is installed.
10. All the extensions from the DL Extension Library are installed.
11. An OpenML flow is already available.
12. The version of the libraries should match the dependencies of the OpenML flow.

Cautions and warnings

The local website can take a few seconds to become active.

Procedures

1. Open the visualization module and run the visualizer.py file.
2. Open the local website in a browser by clicking on the IP (by default 127.0.0.1:8050) shown in the console.
3. Type the id of the OpenML flow into the input field, marked on figure 3.1.
4. Press the "LOAD FLOW" button, marked on figure 3.2.

Likely errors

If the flow is not supported by the DL Extension Library the visualization will not be possible.

Flow and run visualization



Enter run id LOAD RUN Enter flow id LOAD FLOW

Figure 3.1: "Enter flow id" input field

Flow and run visualization



Enter run id LOAD RUN Enter flow id LOAD FLOW

Figure 3.2: "LOAD FLOW" button

3.7 Visualize a run

Functional description

This tutorial describes how the user can visualize a run.

Preconditions

1. Dash is already installed.
2. OpenML is already installed.
3. Keras is installed.
4. PyTorch is installed.
5. ONNX is installed.
6. MXNet is installed.
7. Graphviz is installed.
8. Pydot is installed.

9. Onnxmltools is installed.
10. All the extensions from the DL Extension Library are installed.
11. A run is already available.

Cautions and warnings

The local website can take a few seconds to become active.

Procedures

1. Open the visualization module and run the visualizer.py file.
2. Open the local website in a browser by clicking on the IP (by default 127.0.0.1:8050) shown in the console.
3. Type the id of the run into the input field, marked on figure 3.3.
4. Press the "LOAD RUN" button, marked on figure 3.4.
5. Choose a metric to be visualized from a drop down menu, which is shown on figure 3.5. The default visualized metric is the loss graph. For a regression task the loss, the mean square error, the mean absolute error or the root mean square error can be visualized. For a classification task the the loss or the accuracy can be plotted.

Likely errors

If the run does not have training data it will not be possible to visualize it.

Flow and run visualization



The image shows a user interface for visualizing flows and runs. It consists of four elements arranged horizontally: an input field with the placeholder text 'Enter run id' (highlighted with a green border), a button labeled 'LOAD RUN', another input field with the placeholder text 'Enter flow id', and a final button labeled 'LOAD FLOW'.

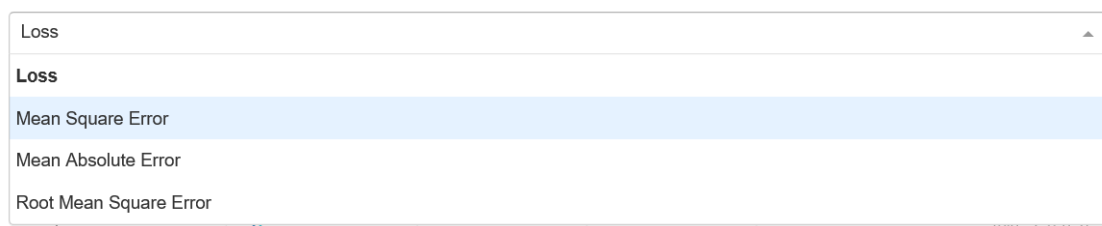
Figure 3.3: "Enter run id" input field

Flow and run visualization



The interface consists of four rounded rectangular buttons arranged horizontally. From left to right: a button with the text 'Enter run id', a button with the text 'LOAD RUN' (which is highlighted with a green border), a button with the text 'Enter flow id', and a button with the text 'LOAD FLOW'.

Figure 3.4: "LOAD RUN" button



A vertical drop-down menu is shown. The top bar is labeled 'Loss' with an upward-pointing triangle on the right. Below this, the word 'Loss' is repeated in bold. The menu is open, showing a list of metrics: 'Mean Square Error' (highlighted with a light blue background), 'Mean Absolute Error', and 'Root Mean Square Error'. The menu has a thin border and a small blue double-line icon at the bottom left.

Figure 3.5: A drop down menu with available run metrics

4 | References

DL Extension Library is an extension of the OpenML platform. This chapter leaves out the generic OpenML functionality, and instead focuses solely on the parts related to DL Extension Library. In case the user is unfamiliar with OpenML, they are advised to visit [2] for more information.

This chapter explains all the functionality of the visualization application. For additional information and images, please refer to [4].

4.1 Initial menu

Functional Description:

The homepage of the visualization application allows the user to visualize runs and OpenML flows. This is depicted in figure 4.1.

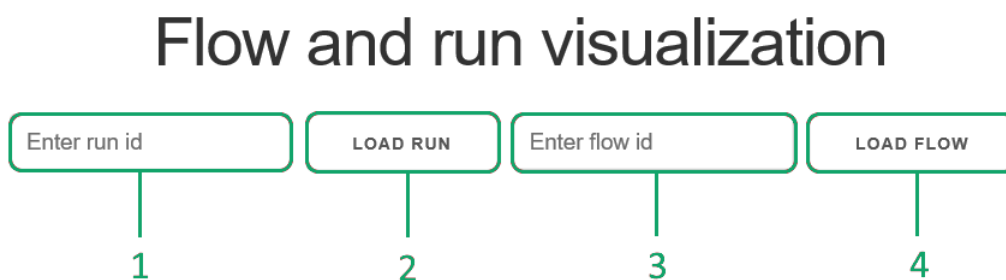


Figure 4.1: Flow and run visualization initial screen

Formal Description:

	Operation	Description	Result
1	Enter run id	Enter the run id	the run input field is filled in.
2	Loads a run	Click the "LOAD RUN" button	Run is being visualized.
3	Enter OpenML flow id	Enter the OpenML flow id	the run input field is filled in.
4	Loads an OpenML flow	Click the "LOAD FLOW" button	Flow is being visualized.

Possible errors:

Error if the flow cannot be turned into ONNX. Error if the run doesn't have training data. Possible OpenML errors, which are outside the scope of the project. Possible Dash errors,

which are outside the scope of the project.

4.2 Run visualization

Functional Description:

This page displays the visualization of a run. It depicts the loss graph of this run by default and shows all the different folds used as a testing set, displayed on the right. See figure 4.2.

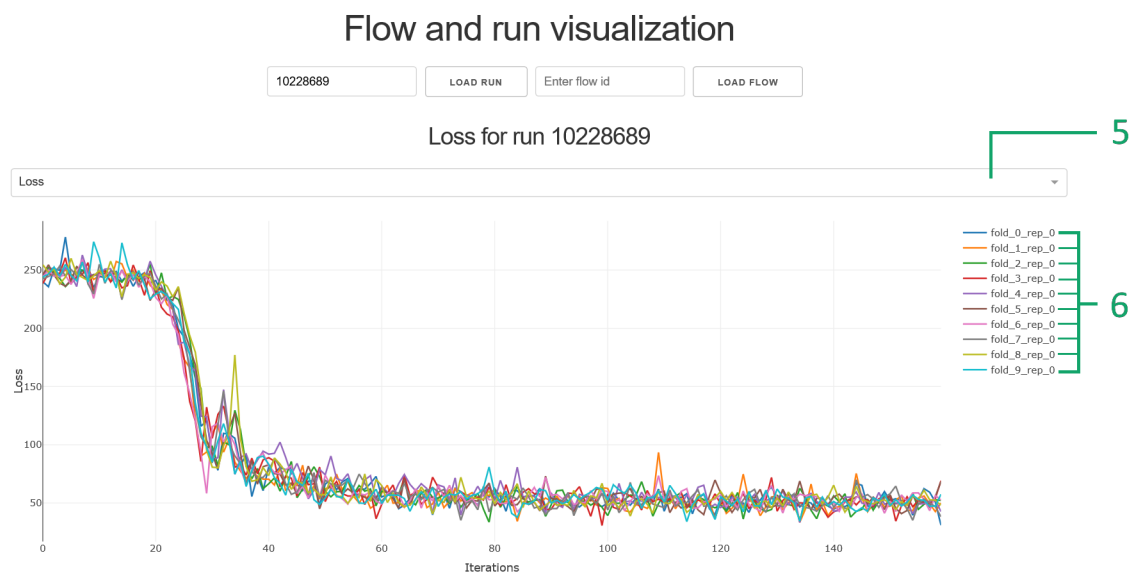


Figure 4.2: Flow and run visualization initial screen

Formal Description:

	Operation	Description	Result
5	Shows drop-down bar	Click on the drop-down bar	Additional available metrics options are displayed.
6	Displays/Hides a fold	Click on a fold	The fold appears or disappears depending on its current state.

Possible errors:

Error if the run doesn't have training data. Possible OpenML errors, which are outside the scope of the project. Possible Dash errors, which are outside the scope of the project.

4.3 Drop-down bar

Functional Description:

The drop-down bar displays the different metric options to choose from for a given run. This is depicted in figure 4.3.

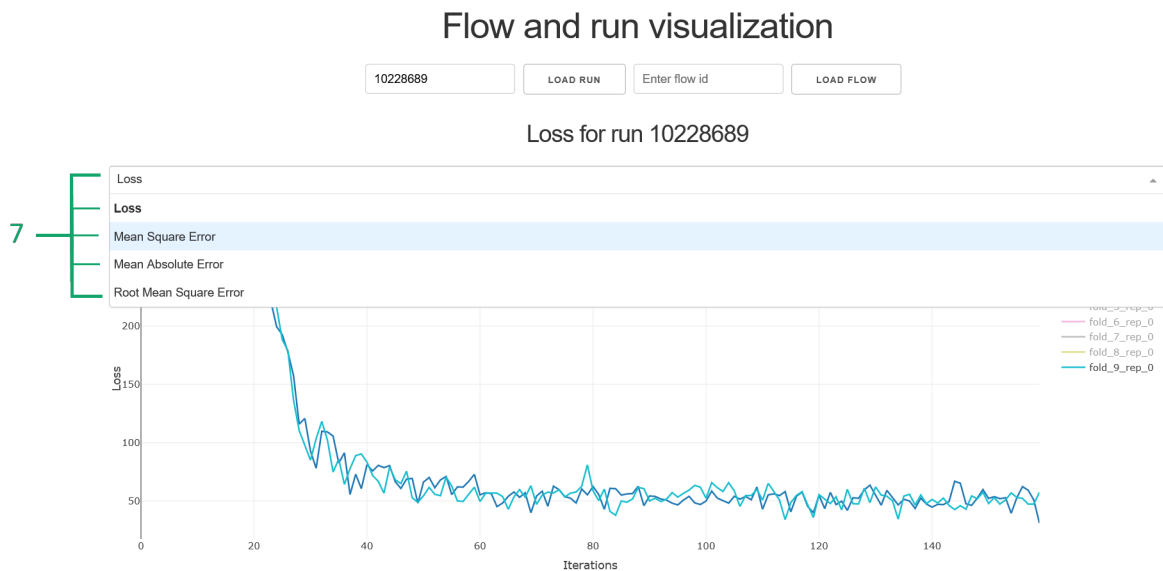


Figure 4.3: Drop-down bar

Formal Description:

	Operation	Description	Result
7	Applies metric	Click a metric	The metric for a given run is displayed.

Possible errors:

Error if the run doesn't have training data. Possible OpenML errors, which are outside the scope of the project. Possible Dash errors, which are outside the scope of the project.

4.4 Metric Selection

Functional Description:

This page displays the result of selecting the "mean square" metric for a given run. Additionally it shows the hover functionality of the visualization app. See figure 4.4.

Formal Description:

Flow and run visualization

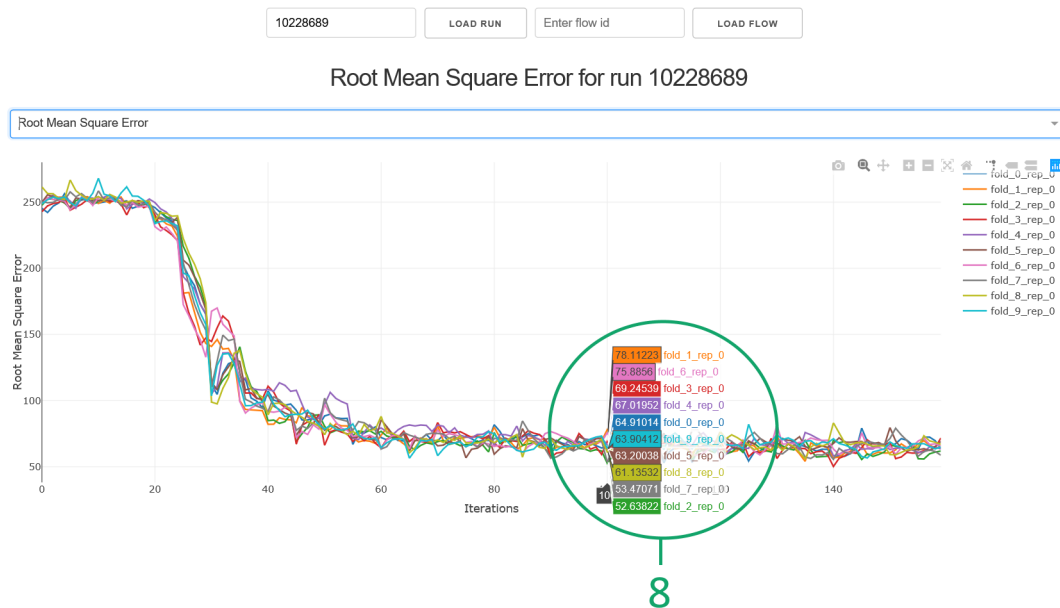


Figure 4.4: Metric selection

	Operation	Description	Result
8	Display fold values at given interval	Hover over fold interval	The values for the given metric for each fold are given for a chosen number of iterations.

Possible errors:

Error if the run doesn't have training data. Possible OpenML errors, which are outside the scope of the project. Possible Dash errors, which are outside the scope of the project.

4.5 Flow Visualization

Functional Description:

This page displays the result of typing in a OpenML flow id and clicking on the "LOAD FLOW" button. Note that there is no additional functionality on this page. Refer to figure 4.5.

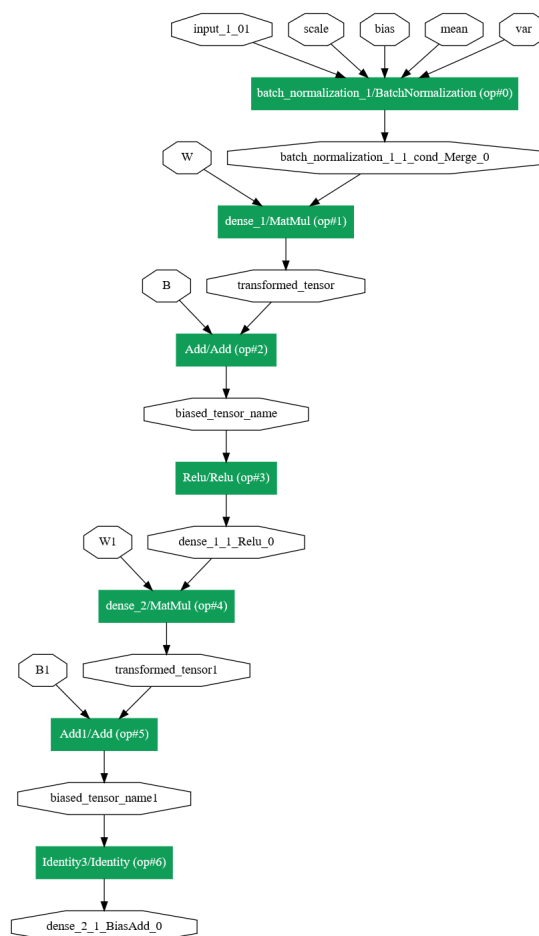


Figure 4.5: Flow visualization

Possible errors:

Error if the flow cannot be turned into ONNX. Possible OpenML errors, which are outside the scope of the project. Possible Dash errors, which are outside the scope of the project.

A | Error Messages and Recovery

The DL Extension Library is built on an already existing platform which handles most of the functionality and also uses functionality of several deep learning libraries. As a result, any error handling that happens inside the library is actually inherited from OpenML or any of the implemented deep learning libraries. This means that the error handling falls outside of the scope of this project.

B | Glossary

DL Extension Library	The product to be created in order to convert the deep learning models and the model transfer specifications (i.e. ONNX and MLflow) into a format supported by the OpenML platform.
OpenML Python API	"A connector to the collaborative machine learning platform OpenML.org. The OpenML Python package allows to use datasets and tasks from OpenML together with scikit-learn and share the results online" [2].
OpenML flow	The representation of untrained machine learning models in the OpenML platform.
Apache MXNet	"Apache MXNet is an open-source deep learning software framework, used to train and deploy deep neural networks. It is scalable, allowing for fast model training, and supports multiple programming languages (C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl, and Wolfram Language)" [5].
Dash	"Dash is a user interface library for creating analytical web applications" [6].
Dataset	"Datasets are pretty straight-forward. They simply consist of a number of rows, also called instances, usually in tabular form" [2].
Deep learning model	A neural network with more than three layers. Deep learning is a subset of machine learning.
Fold	A subset of the training data.
Keras	"Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano" [7].
Neural Network	"Computing systems vaguely inspired by the biological neural networks that constitute animal brains." It is "a framework for many different machine learning algorithms" [8].
OpenML	An online machine learning platform for sharing and organizing data, machine learning algorithms and data experiments.
OpenML Support Squad	Name of the development team.
Project "Sharing deep learning models"	The project on which the OpenML Support Squad is working on.
PyTorch	"An open source deep learning platform that provides a seamless path from research prototyping to product deployment" [9].

Run	"A run is a particular flow, that is algorithm, with a particular parameter setting, applied to a particular task" [2].
Task	"A task consists of a dataset, together with a machine learning task to perform, such as classification or clustering and an evaluation method. For supervised tasks, this also specifies the target column in the data" [2].
<hr/>	
API	Application Programming Interface.
DL	Deep Learning.
ESA	European Space Agency.
MXNet	Apache MXNet.
ONNX	Open Neural Network eXchange.
SRD	Software Requirements Document.
SUM	Software User Manual.
URD	User Requirements Document.

Bibliography

- [1] Software Standardisation and Control. *ESA software engineering standards*. 1991.
- [2] OpenML. *OpenML Documentation*. URL: <https://docs.openml.org/> (visited on 09/05/2019).
- [3] OpenML Support Squad. *DL Extension Library, URD User Requirement Document version 1.0.0*.
- [4] OpenML Support Squad. *DL Extension Library, SRD Software Requirement Document version 1.0.0*.
- [5] Wikipedia. *Apache MXNet*. URL: https://en.wikipedia.org/wiki/Apache_MXNet (visited on 04/26/2019).
- [6] Kyle Kelley. *Introducing Dash*. URL: <https://medium.com/@plotlygraphs/introducing-dash-5ecf7191b503> (visited on 06/05/2019).
- [7] Keras. *Keras homepage*. URL: <https://keras.io/> (visited on 09/05/2019).
- [8] Wikipedia. *Artificial Neural Network*. URL: https://en.wikipedia.org/wiki/Artificial_neural_network (visited on 04/30/2019).
- [9] PyTorch. *PyTorch homepage*. URL: <https://pytorch.org/> (visited on 09/05/2019).