

# **HLS4ML - Testing Infrastructure**

**October 2020**

**AUTHOR:**

Sarun Nuntaviriyakul  
Chulalongkorn University

**SUPERVISOR:**

Vladimir Loncar  
CERN

## PROJECT SPECIFICATION

Testing infrastructure for Keras and QKeras framework conversion in hls4ml (High Level Synthesis for Machine Learning) using pytest testing framework and Jenkins pipeline

*Designing a testing infrastructure to validate the implementation of new features in Keras and QKeras framework conversion in hls4ml (High Level Synthesis for Machine Learning) using pytest testing framework and Jenkins pipeline (From model conversion and internal representation to HLS synthesis and resource estimation)*

## **ABSTRACT**

Testing is a process to execute a software or program and find all the errors and bugs in software/program which do not meet the requirements or have inconsistency during the executing of the program. With more uses of machine learning and deep neural networks in the particle physics sector, High-Level Synthesis languages for FPGAs called hls4ml are used. With more layers and frameworks being supported, it is crucial to maintain the consistency and the functionality of the software when there is new change added. To ensure that the conversion process of deep neural networks is being executed correctly we create a testing infrastructure and testing pipeline for the conversion of deep neural networks using hls4ml.

# **TABLE OF CONTENTS**

<b>INTRODUCTION</b>	<b>5</b>
<b>HLS4ML TESTING</b>	<b>6</b>
<b>QKeras Testing</b>	<b>6</b>
<b>Keras Testing</b>	<b>7</b>
<b>JENKINS PIPELINE</b>	<b>8</b>
<b>CONCLUSION</b>	<b>9</b>
<b>FUTURE DEVELOPMENT</b>	<b>10</b>
<b>APPENDIX</b>	<b>11</b>

## INTRODUCTION

hls4ml is a package for machine learning inference in FPGAs which convert machine learning models from many frameworks for example Keras, Tensorflow into HLS (high level synthesis) project.

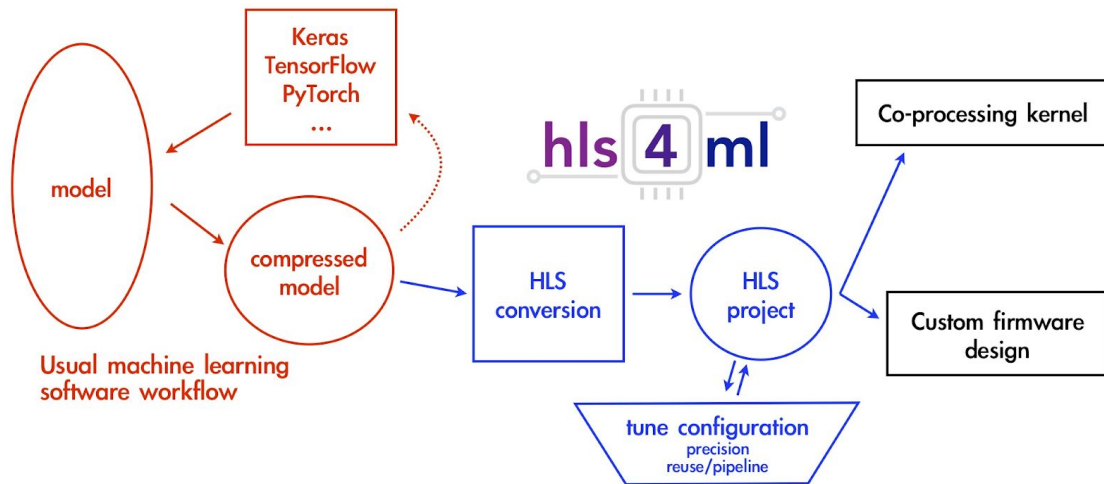


Figure 1. hls4ml workflow for deep learning model conversion

Currently hls4ml supports conversion using Python API and by providing Configuration files. We would like to test the whole conversion process from the conversion from different deep learning model frameworks to an internal representation of `hls_model`, then from the internal model to an actual hls project which can then be deployed on FPGAs. The process will use many unit tests for testing the conversion of each layer. Integration test for testing the accuracy of the whole flow from loading the deep learning models, converting to hls internal representation and then to an actual C++ code using Vivado HLS.

The automated testing pipeline will be triggered every time a new change has been added to the code. The testing infrastructure will make hls4ml more robust and minimize the amount of bugs, also allowing for more features and framework in the future.

## HLS4ML TESTING

Testing is done separately in different python files comparing the original model accuracy and hls\_model accuracy, the number of layers, the conversion of different layers ,output shapes, in both the python api and from the command line interface.

```
===== test session starts =====
platform linux -- Python 3.7.8, pytest-6.1.1, py-1.9.0, pluggy-0.13.1
rootdir: /hls4ml
collected 82 items

test_keras_api.py ..... [ 29%]
test_keras_vivado.py ..... [ 36%]
test_qkeras_api.py ..... [ 73%]
test_qkeras_vivado.py ..... [100%]

===== warnings summary =====
```

Figure 2. Keras and QKeras testing using Pytest

A total of 82 tests have been executed in Keras and QKeras framework conversion to hls\_model.

### QKeras Testing

QKeras is a quantization extension to Keras which uses the implementation of some of Keras layers, for example QDense (Dense Layer) , Qconv1D (Conv1D Layer) , QActivation (Activation Layer). Quantized version of Keras provides the creation of deep learning neural networks with lower precision bits and lower inference time with minimal accuracy loss.

```
import pytest
@pytest.mark.parametrize('activation_int', quantized_integer_list)
@pytest.mark.parametrize('activation_bit', quantized_bit_list)
```

The test uses pytest framework which supports many functional testing for applications and libraries. Parametrizing test functions are used to simplify and test every combination of the parameters.

```

hls_model = hls4ml.converters.convert_from_keras_model(model,
    project_name=generate_project_name())
hls_model.compile()

model_pred=model.predict(test_input)
hls_pred=hls_model.predict(test_input)

assert(_output_shape(model_pred,hls_pred))
assert(_accuracy(model_pred,hls_pred,error))
assert(_layer_number(model,hls_model))
assert(_alpha(model,hls_model))

```

We assert on the output\_shape, the accuracy and the number of layers.

```

# Acceptable accuracy error
error = 15

```

The acceptable error in model accuracy can be set when comparing the model accuracy with the hls model.

## Keras Testing

In Keras framework we test the same layer as the quantized version with some additional layer which is only supported in Keras, for example , ThresholdedReLU, LeakyReLU.

```

assert round(np.average(np.subtract(np.abs(keras_prediction), np.abs(hls_prediction)))) < 3

assert len(model.layers) + 1 == len(hls_model.get_layers())
assert list(hls_model.get_layers())[0].attributes['class_name'] == "InputLayer"
assert list(hls_model.get_layers())[1].attributes["class_name"] == model.layers[0]._name
assert list(hls_model.get_layers())[2].attributes['class_name'] == model.layers[1]._name
assert list(hls_model.get_layers())[0].attributes['input_shape'] ==
list(model.layers[0].input_shape[1:])

```

We assert on the output\_shape, the accuracy, layer name and the number of layers.

# JENKINS PIPELINE

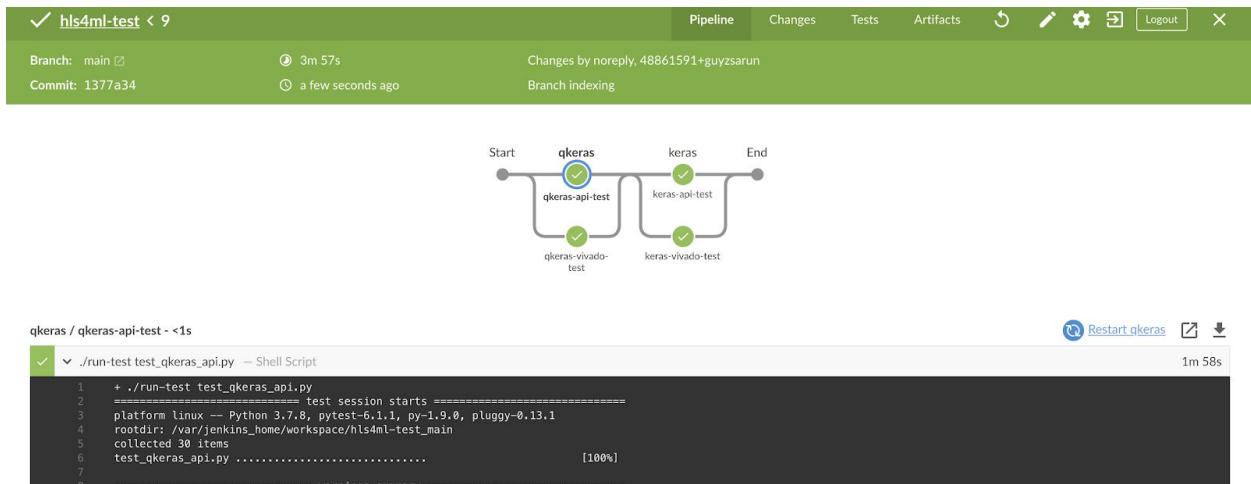


Figure 3. Jenkins Pipeline [<https://github.com/hls4ml-testing/hls4ml-test>]

Local Jenkins Runner is set up for running the pipeline for Keras and QKeras testing when there is a change to the repository.

There are 2 main stages for the test, for Keras and QKeras conversion each of the main stage is divided into 2 sub stages: the conversion from the python API test and the conversion from the command line interface test. All the stages are running on a separate docker environment which ensures there are no conflicts during the conversion process.

## QKeras pipeline configuration

```
stage('qkeras') {
  parallel {
    stage('qkeras-api-test') {
      steps {
        sh './run-test test_qkeras_api.py'
      }
    }
    stage('qkeras-vivado-test') {
      steps {
        sh './run-test test_qkeras_vivado.py'
      }
    }
  }
}
```



## **CONCLUSION**

Converting deep learning models into the hls4ml framework provided model size reduction, reduced inference time with minimal loss in accuracy which can be easily deployed on FPGAs. With the testing infrastructure integrated it would make the hls4ml framework more robust and also validate the implementation of the new layers / features in the future.

## **FUTURE DEVELOPMENT**

Jenkins pipeline and the testing infrastructure has already been set up and can be further implemented to cover all the deep learning framework that hls4ml supports. Also more input and output dimensions could be added for testing the conv2d or conv3d layers. Because the scripts for testing hls4ml conversion by command line interface has been set up, it will be easier to test other frameworks in the future.

## APPENDIX

<https://fastmachinelearning.org/hls4ml/>

<https://github.com/google/qkeras>

<https://docs.pytest.org>

<https://www.jenkins.io>

<https://arxiv.org/abs/2006.10159>