

# Chisel4ml - high-level software architecture


 No description has been provided for this image

Image from *vreca et.al.: Generating Direct Logic Circuit Implementations of Deeply Quantized Neural Networks Using Chisel4ml*

- Here we have a high-level overview of the chisel4ml architecture.
- It has a python frontend and a chisel backend.
- The Python frontend can consume quantized neural networks in the form of QKeras or QONNX models. These are two formats for serializing such neural networks.
- The python frontend transforms the model description to an internal format called Low-Bitwidth Intermediate Representation or LBIR.
- LBIR describes the quantized neural network, and is sent to the chisel backend where the circuit is generated.
- The circuit can then be simulated from python, or the verilog files can be packaged into the desired directory.

## chisel4ml - Train a model in Brevitas (PyTorch)

```
In [ ]: import chisel4ml
        from lhc_model import get_lhc_jets_model
        from lhc_data import get_lhc_dataset
        from train import train_model, eval_model
        import torch
        from brevitas.export import export_qonnx
        from qonnx.util.cleanup import cleanup_model
        from qonnx.core.modelwrapper import ModelWrapper
        from server import create_server
```

```
In [ ]: brevitas_model = get_lhc_jets_model(bitwidth=4)
        train_loader, test_loader = get_lhc_dataset(batch_size=512)
        train_model(
            model=brevitas_model,
            train_loader=train_loader,
            criterion=torch.nn.CrossEntropyLoss(),
            optimizer=torch.optim.Adam(brevitas_model.parameters(), lr=0.001),
            epochs=1,
            device='cpu',
```

```
    prune_rate=0.5,  
)
```

- I want to know show case how to train and deploy a model.
- We train a simple 4-layer fully-connected model for a single epoch on the hls4ml dataset of high-pT jets from simulations of LHC proton-proton collisions.
- The performance of this model will be terrible, but this exact topology can be used for effective triggering, if we train for a higher number of epochs.

## Evaluate train model

```
In [ ]: eval_model(brevitas_model, test_loader, 'cpu')
```

- We can now evaluate the model on the test data set.
- As expected, we get terrible performance, more or less random.
- However, if we trained the training parameters this would be better.

## Export the model to QONNX

```
In [ ]: qonnx_proto = export_qonnx(brevitas_model, torch.randn(brevitas_model.ishape  
qonnx_model = ModelWrapper(qonnx_proto)  
qonnx_model = cleanup_model(qonnx_model)
```

```
In [ ]: import IPython  
import netron  
  
qonnx_model.save('model.onnx')  
addr = 'localhost'  
port = 5555  
netron.start('model.onnx', (addr, port), browse=False)  
IPython.display.IFrame(f'http://{addr}:{port}', width=1200, height=600)
```

- The Brevitas library is based on PyTorch, and it can export the model as a QONNX model.
- QONNX is a quantization extension for the Open Neural Network eXchange standard.
- This model can be visualized as shown.

## Create chisel4ml circuit

```
In [ ]: from chisel4ml.transform import qonnx_to_lbir
        from chisel4ml import generate

        lbir_model = qonnx_to_lbir(qonnx_model)
        accelerators = generate.accelerators(
            lbir_model,
            minimize="delay",
        )
        c4ml_server, c4ml_subp = create_server("/c4ml/chisel4ml.jar")
        circuit = generate.circuit(
            accelerators,
            lbir_model,
            use_verilator=True,
            gen_timeout_sec=9000,
            server=c4ml_server,
        )
```


- The QONNX model is input to the chisel4ml python frontend.
- It is transformed into LBIR.
- We then create a chisel backend server.
- And then send the LBIR model to the chisel backend.
- This then generates the firrtl and verilog code.

## RTL simulation of the circuit

```
In [ ]: import numpy as np
        res = circuit(np.zeros(16))
        print(res)
```

- This circuit can then be simulated from python.
- It uses verilator in the background to perform the RTL simulation.
- The result is a Numpy array and in this case it is just all zeros, because we haven't trained the model enough.

## Comparison with hls4ml


 No description has been provided for this image


The results are from our paper For more info see *vreca et.al.: Generating Direct Logic Circuit Implementations of Deeply Quantized Neural Networks Using Chisel4ml*


- This figure shows the experiment structure for comparing the chisel4ml results to hls4ml results on the most parallel setting.

- We train the models with brevitax and transformed them to QONNX. This model is then input to chisel4ml and hls4ml, which then use Xilinx tools 2023.1 to perform the synthesis.

## Convolutional neural networks with different bitwidth of quantization - hls4ml vs chisel4ml

No description has been provided for this image

No description has been provided for this image

 No description has been provided for this image

- These are results for a generic convolutional neural network trained on the MNIST dataset.
- The left most figure shows the number of look-up tables required for implementation, the center shows the path-delay and the right most the total generation time from qonnx model to synthesized Verilog.
- The blue line is chisel4ml and orange is hls4ml.
- As you can see chisel4ml creates circuits with lower consumption of lookup tables, compared to hls4ml on the fully-parallel setting.
- The other results are that chisel4ml generation time is orders of magnitude smaller, and the path delay is higher.
- However, the path delay can easily be mitigated by inserting additional pipeline registers, and then use retiming. This can be easily adjusted, so this result could fairly easily be corrected.
- chisel4ml is currently limited to only maximum parallelism. So the direct-circuit implementation of neural networks. Hls4ml allows the designer to choose a tradeoff between amount of parallelism and thus circuit size, and performance requirements. So chisel4ml is a good choice for users who have ultra-low latency requirements.

## Thank you for your attention

The authors acknowledge the financial support from the Slovenian Research and Innovation Agency under grant: No. P2-0098. This work is also part of projects

that are funded by the ECSEL Joint Undertaking under grant agreement No 101007273 (DAIS) and by the Chips Joint Undertaking under grant agreement No 101139892 (EdgeAI-Trust).