

End-to-end hardware-aware model compression and deployment with PQuant and hls4ml

Roope Niemi, Chang Sun, Anastasiia Petrovych,
Enrico Lupi, Dimitrios Danopoulos, Arghya Ranjan Das,
Miaoyuan Liu, Sebastian Dittmeier, Michael Kagan,
Maurizio Pierini, Vladimir Loncar



NexTGen
Next Generation Triggers



EP-SFT
Software Frameworks and Tools

Triggers and Machine Learning

In LHC, collisions at 40MHz frequency

Triggers are used to filter the data, reducing the data rate

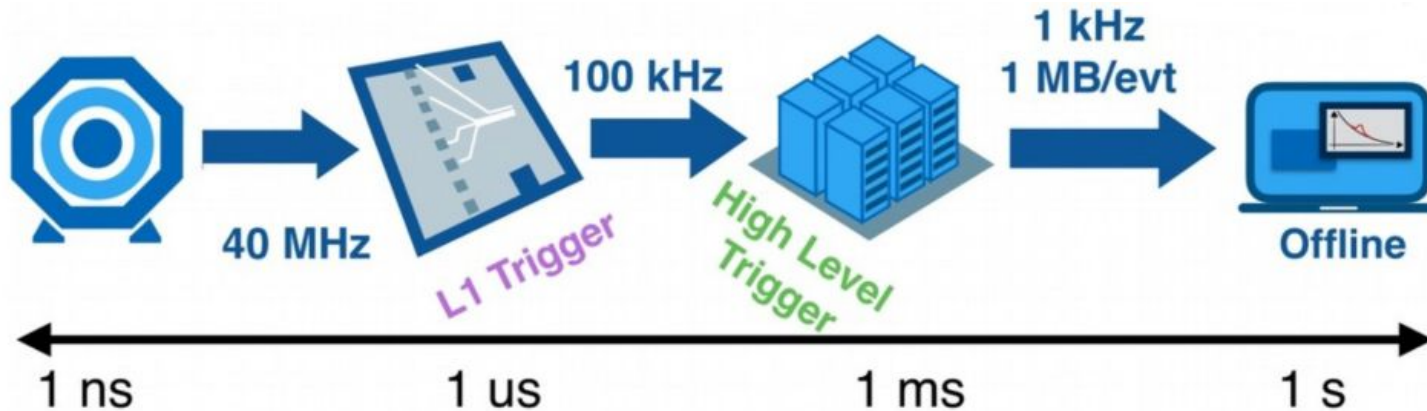
Strict latency constraints

To improve pattern recognition and keep up with increasing data rates in the trigger, use ML

Trigger hardware has limited resources



Compress ML models before deployment to HW

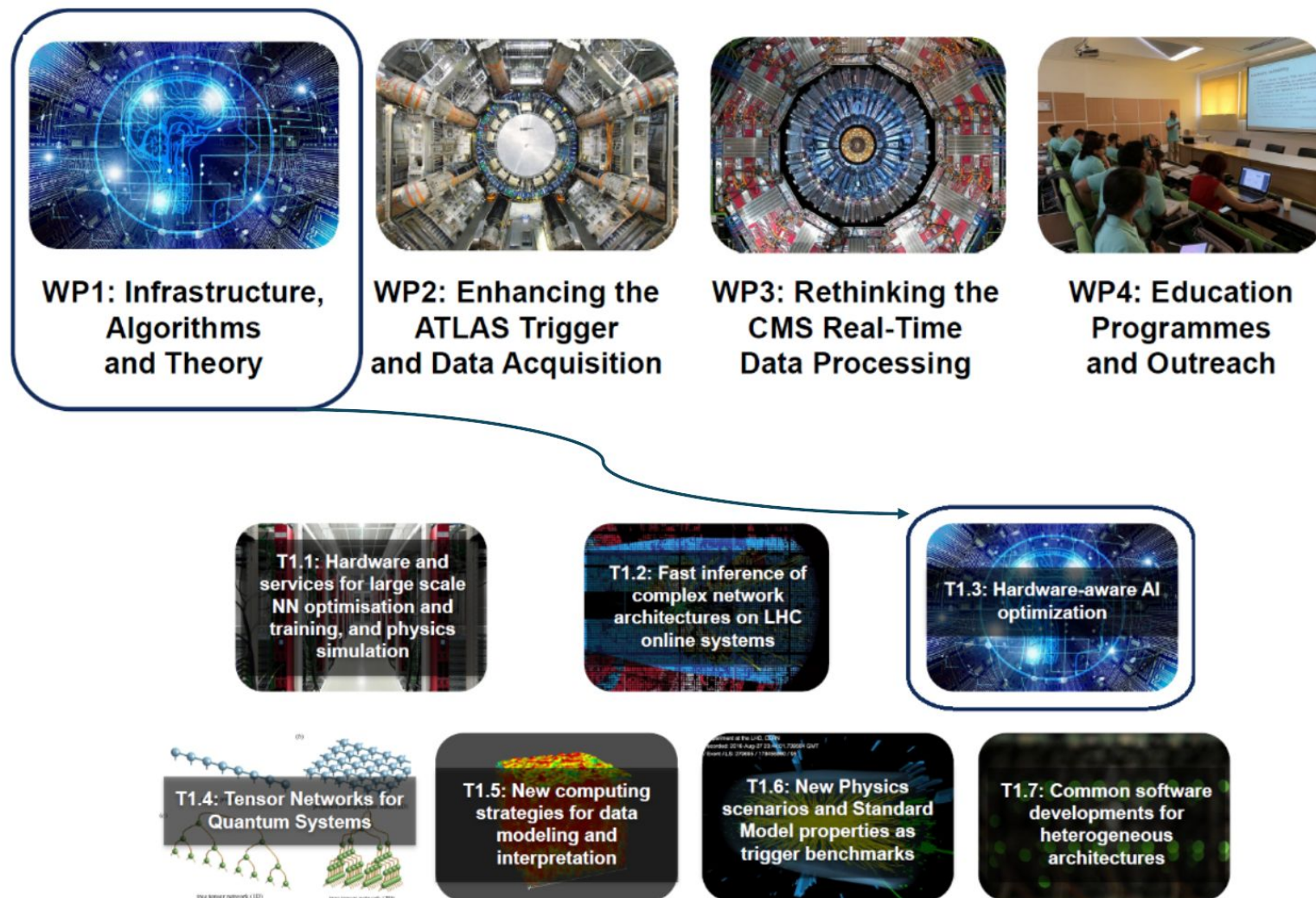


NGT

Collaboration between the
CERN EP, TH and IT
Departments and the ATLAS
and CMS experiments

Use ML to get more physics
information out of the HL-LHC
data

T1.3: optimize and compress
ML models for hardware



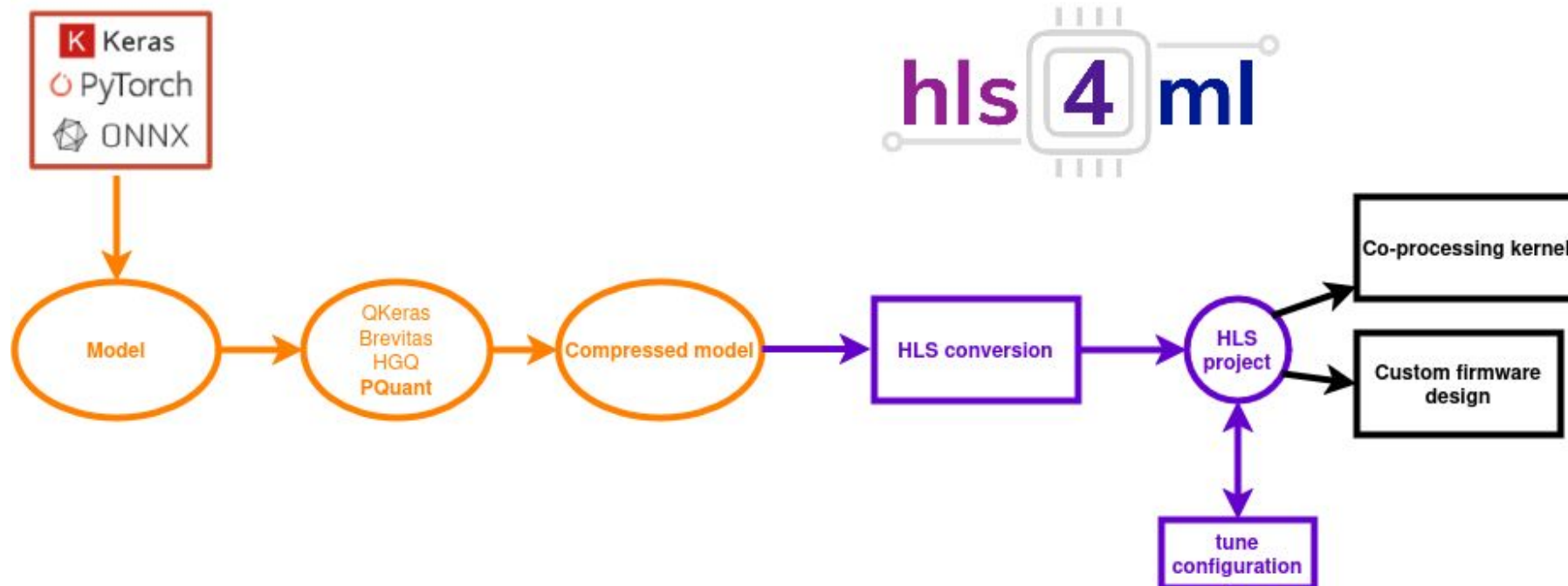
hls4ml

Used to translate trained ML models to optimized HLS code

↳ Synthesize into FPGA bit stream → Run on FPGAs

Easy to use, flexibility for customization

Used at CERN, large community of users at FastML



ML model compression

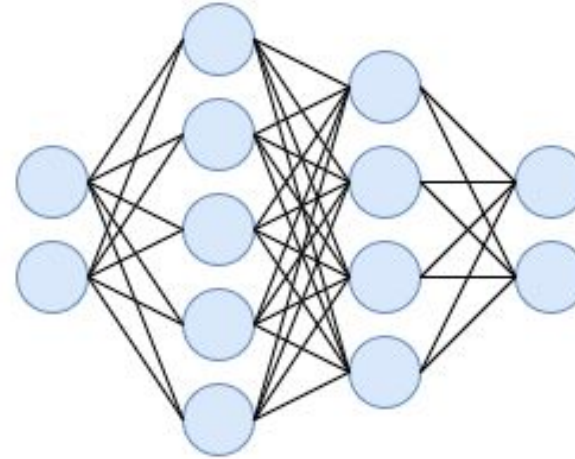
- **An umbrella term for multiple techniques and their subcategories:**
 - Quantization, pruning, distillation, low-rank approximation etc.
- **During training, post-training**
- **Constraints by hardware**

W			Q(W)		
0.1	0.01	0.21	0.125	0	0.25
0.02	0.01	0.83	0	0	0.875
0.77	0.03	0.01	0.75	0	0

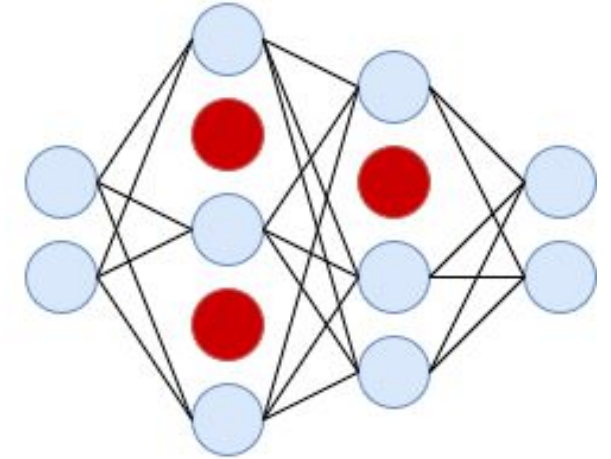
Pruning

- Not all weights in a neural network are necessary → prune weights by learning a mask that when applied, sets weights to 0
- **Granularity:**
 - Unstructured, N:M, structured

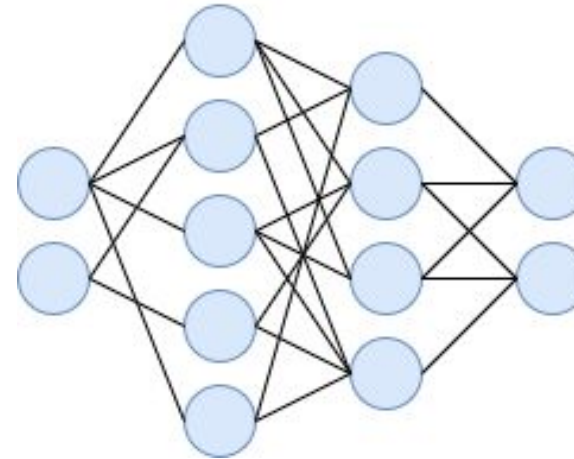
Dense model



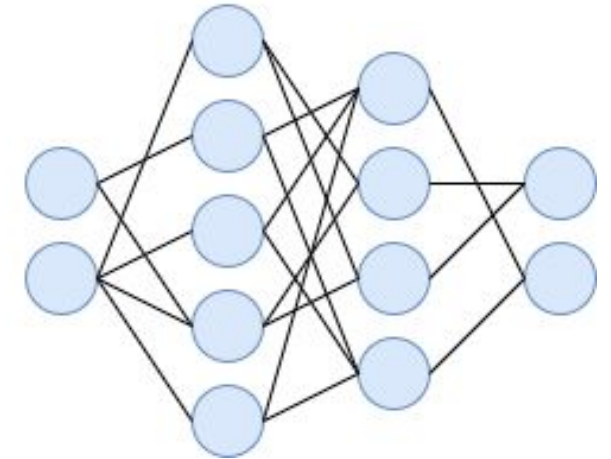
Structured pruning



Unstructured pruning

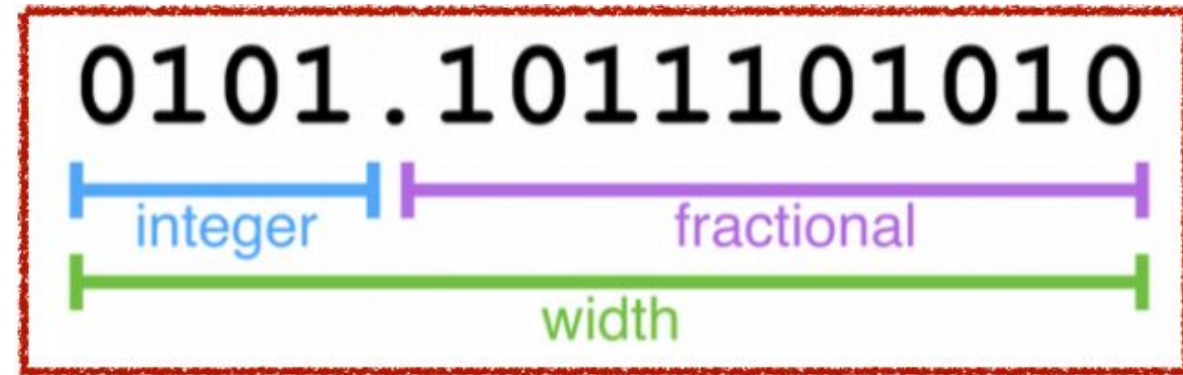


2:4 pruning

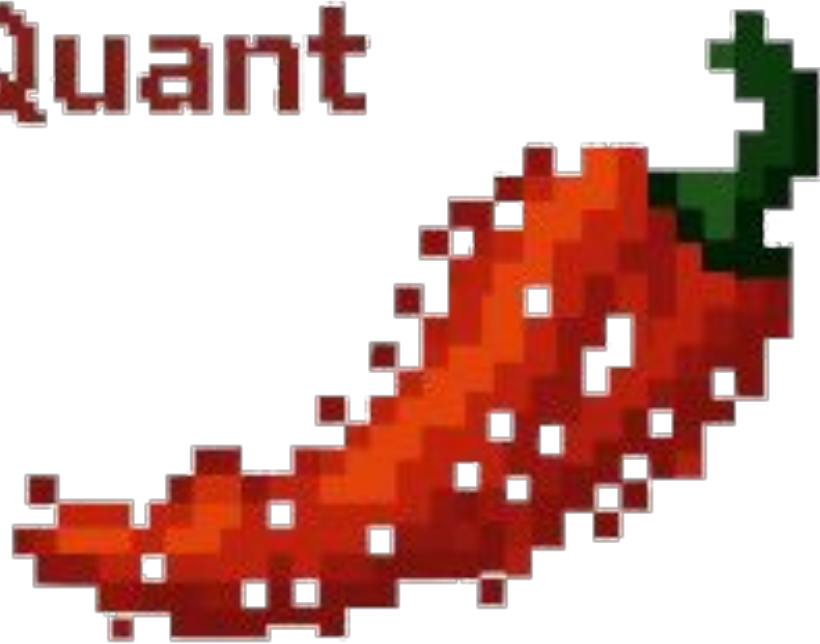


Quantization

- Instead of the usual 32-bit floating point numbers, use fewer bits for weights and computations:
 - fewer bits means lower resource usage, which leaves space for other things, such as higher parallelization, or a bigger model
- Bit reduction introduces error, affecting accuracy. Quantize during training to allow ML model to keep high accuracy
- Granularity

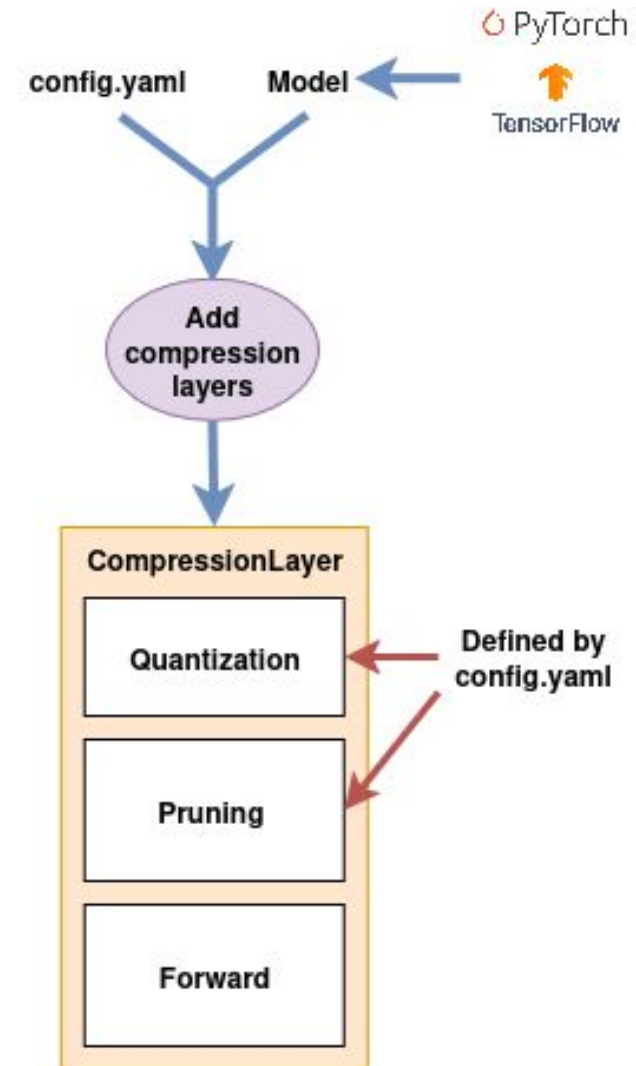


PQuant



PQuant

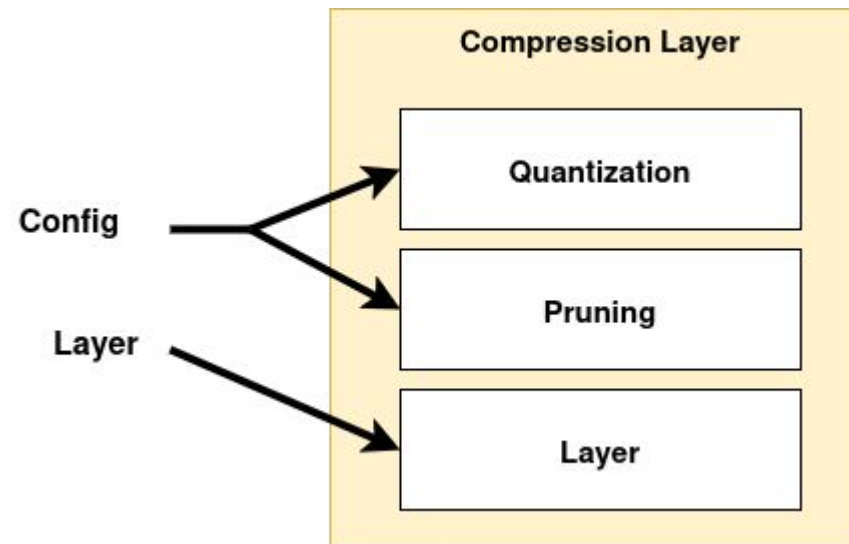
- A tool to train and optimize compressed neural networks for hardware
- No detailed knowledge of ML compression methods required
- Model wrapping, training and cleanup is automatic



Compression layers

- Compression hyperparameters defined by a config file
- Compression layers added with a function call
- Compression layers handle the pruning and quantization of weights and biases before the forward pass

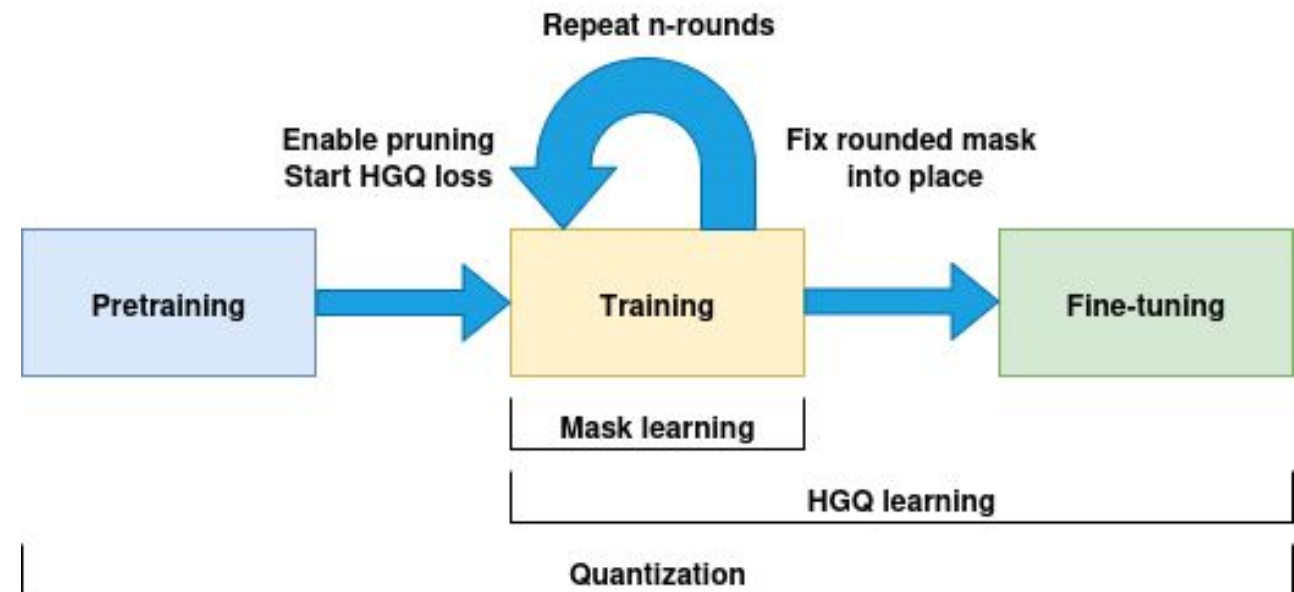
```
model = MyModel()  
# The default config of Continuous Sparsification  
config = get_default_config("cs")  
"""  
Add pruning and quantization. Layers that will be replaced:  
Conv2d -> CompressedLayerConv2d  
Conv1d -> CompressedLayerConv1d  
Linear -> CompressedLayerLinear  
ReLU -> QuantizedReLU  
Tanh -> QuantizedTanh  
"""  
input_shape = (256, 3, 32, 32)  
model = add_pruning_and_quantization(model, config, input_shape)
```



Training

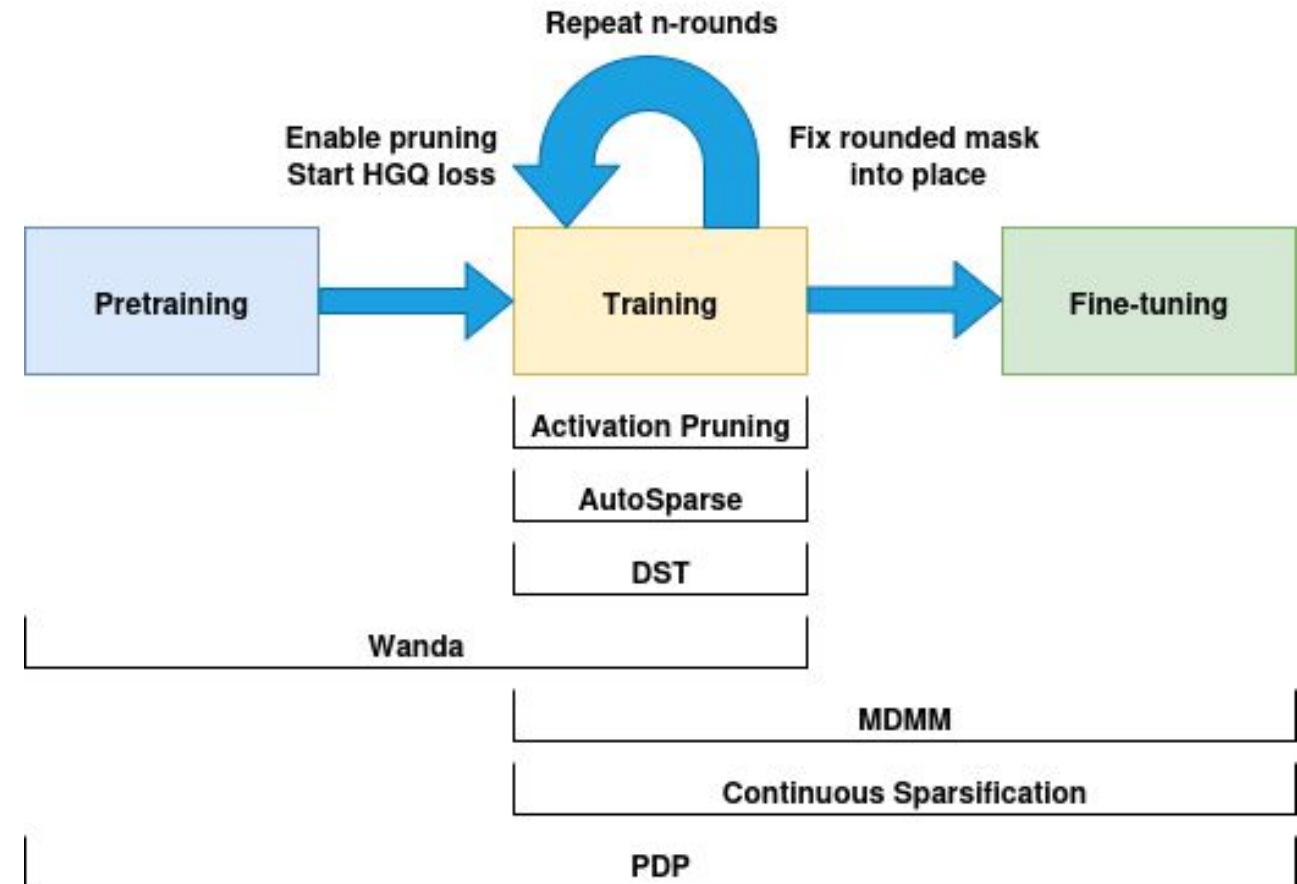
- Different compression methods have different training steps:
 - pretraining: quantize
 - training: quantize + prune
 - fine-tuning: quantize + prune with fixed mask
- ↓
- PQuant provides a generic training function:
 - User provides training and validation functions, the function handles the various steps of the training

```
iterative_train(  
    model=model,  
    config=config,  
    train_func=training_function,  
    valid_func=validation_function,  
    **kwargs  
)
```



Pruning methods

- Implemented 7 different pruning mask based methods + an optimizer based method
- Some methods also have implementations for N:M pruning or structured pruning



Configs

- Use a config file to define the hyperparameters of the compression methods
- Easy to use default configs

```
pruning_parameters:
  disable_pruning_for_layers: # Disable pruning for these layers, even if enable_pruning is true
  -
  enable_pruning: true
  final_temp: 200
  pruning_method: cs
  threshold_decay: 1.0e-09
  threshold_init: 0
quantization_parameters:
  default_integer_bits: 0.
  default_fractional_bits: 7.
  enable_quantization: true
  hgq_gamma: 0.0003
  layer_specific: [] # Layer specific quantization bit values
  use_high_granularity_quantization: false
  use_real_tanh: false
  use_symmetric_quantization: false
training_parameters:
  epochs: 85
  fine_tuning_epochs: 85
  pretraining_epochs: 0
  pruning_first: false
  rewind: post-ticket-search
  rounds: 3
  save_weights_epoch: 2
```

Enable/disable pruning or quantization

Configs

- Use a config file to define the hyperparameters of the compression methods
- Easy to use default configs
- Also fine-grained, more detailed control over hyperparameters

Disable pruning for certain layers

```
pruning parameters:  
  disable pruning for layers:
```

```
- dense1  
enable_pruning: true  
final_temp: 200  
pruning_method: cs  
threshold_decay: 1.0e-09  
threshold_init: 0
```

Default quantization bits

```
quantization parameters:  
  default_integer_bits: 0.0  
  default_fractional_bits: 7.0  
enable_quantization: true  
hqq_gamma: 0.0003
```

Different quantization bits for different layers

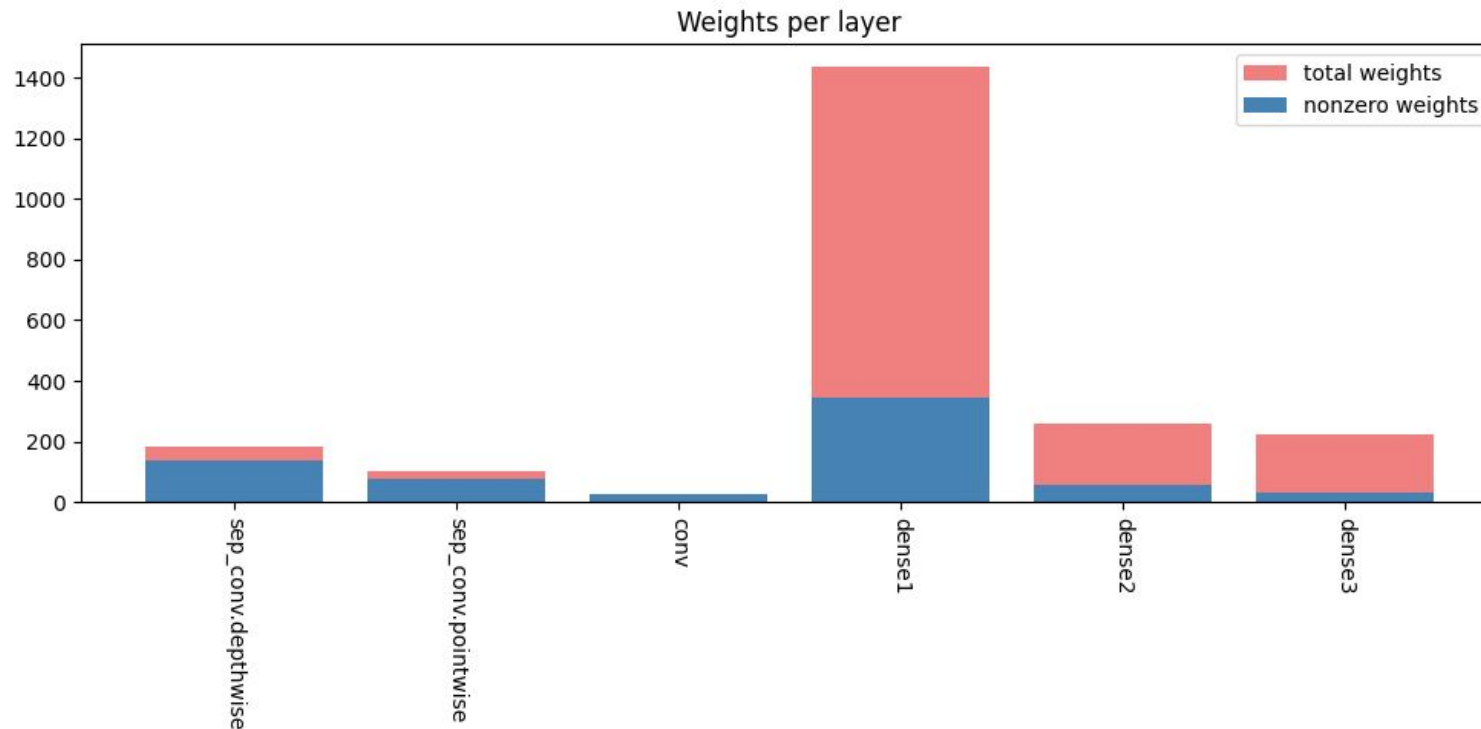
```
  layer specific:  
    conv1:  
      weight:  
        integer_bits: 0  
        fractional_bits: 3 # Signed  
      bias:  
        integer_bits: 0  
        fractional_bits: 3 # Signed  
    relu1:  
      integer_bits: 0  
      fractional_bits: 4 # Unsigned
```

Use HGQ

```
  use high granularity quantization: false  
  use_real_tanh: false  
  use_symmetric_quantization: false  
training parameters:  
  epochs: 85  
  fine_tuning_epochs: 85  
  pretraining_epochs: 0  
  pruning_first: false  
  rewind: post-ticket-search  
  rounds: 3  
  save_weights_epoch: 2
```

Results: SmartPixels

- Pruned and quantized, ~2200 parameters
- 4 bit convolutions + their activations, 8 bit dense + their activations
- ~ 70% pruned



Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 2, 13, 21)	0
depthwise_conv2d (DepthwiseConv2D)	(None, 2, 11, 19)	18
conv2d (Conv2D)	(None, 5, 11, 19)	15
activation (Activation)	(None, 5, 11, 19)	0
conv2d_1 (Conv2D)	(None, 5, 11, 19)	30
activation_1 (Activation)	(None, 5, 11, 19)	0
average_pooling2d (AveragePooling2D)	(None, 5, 3, 6)	0
activation_2 (Activation)	(None, 5, 3, 6)	0
flatten (Flatten)	(None, 90)	0
dense (Dense)	(None, 16)	1,456
activation_3 (Activation)	(None, 16)	0
dense_1 (Dense)	(None, 16)	272
activation_4 (Activation)	(None, 16)	0
dense_2 (Dense)	(None, 14)	238

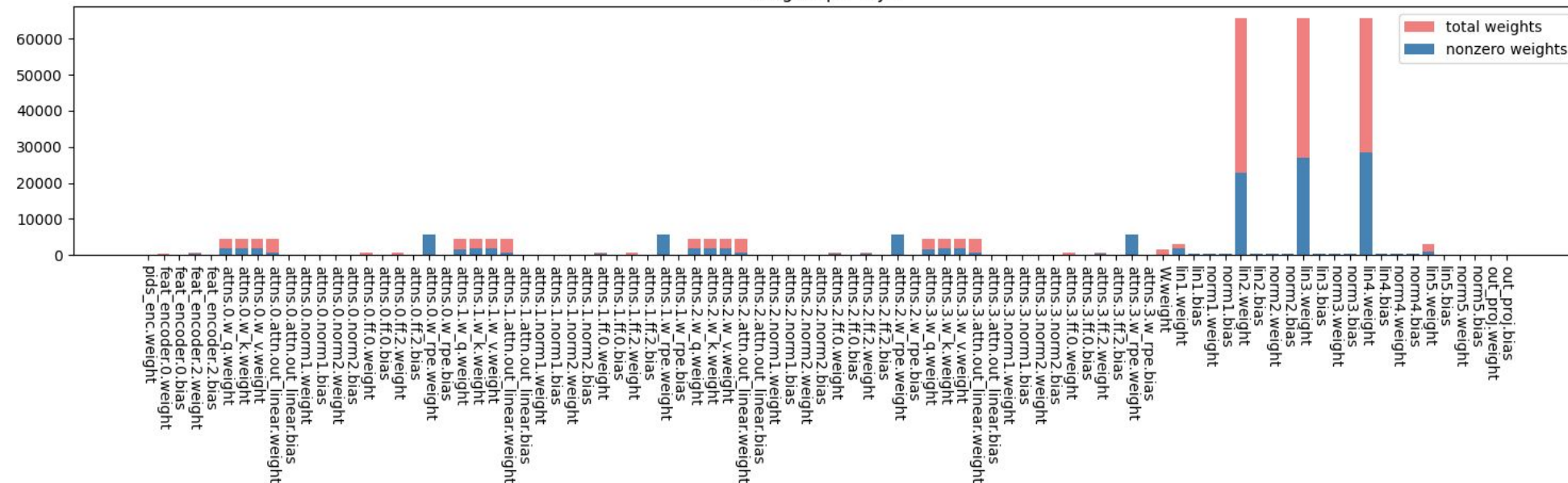
Results: LSH-Based Efficient Point Transformer (HEPT)

Pruned only

- Pileup trainer:
- 43.90% remaining parameters (137,833 / 313,979)
- Test/auc: 38.71 (40.39)



Weights per layer



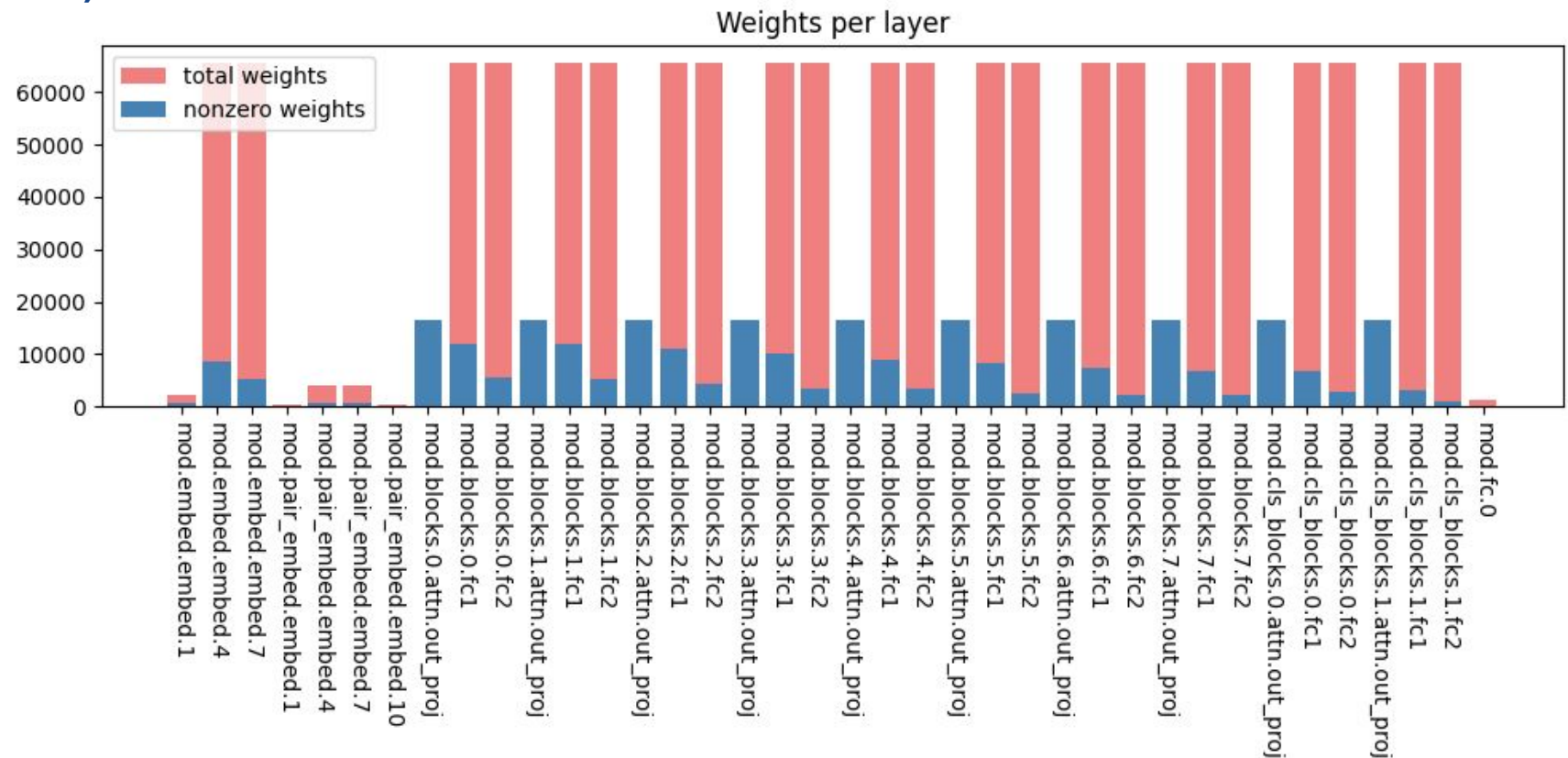
Pruned only

- Tracking trainer 60k:
- 51.14% remaining parameters (168,461 / 329,400)
- Test accuracy@0.9: 86.44 (91.93)

Note: torch.geometric layers are replaced with nn.Linear, no dropout in skip connection

Results: ParT

- Pruned only
- 38.51% remaining parameters
- Accuracy 84.71% (86.1%)



Current and near term work

- Interface with hls4ml
- Hyperparameter optimization
- PyTorch backend: integrate the FitCompress algorithm that finds an optimal configuration of quantization bits and sparsity
- Make first release

Thank you!



NextGen
Next Generation Triggers

Link to repository:

<https://github.com/nroope/PQuant>