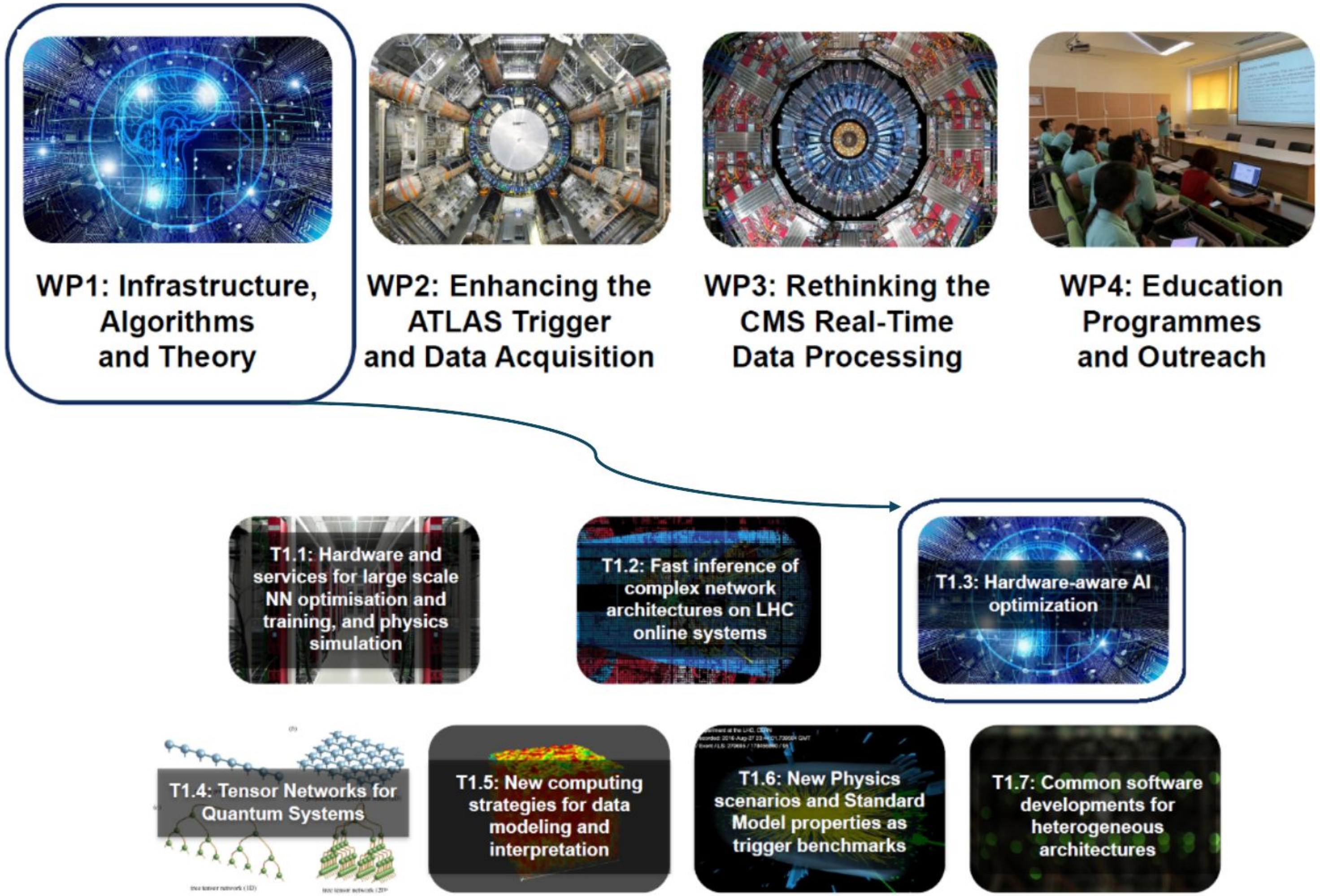
Next Generation Triggers 1st Technical Workshop
25-27 November, 2024

Task 1.3: Hardware-aware AI optimization

Roope Niemi, Vladimir Loncar (lead), Michael Kagan (deputy lead), Maurizio Pierini (deputy lead), Dimitrios Danopoulos, Sebastian Dittmeier, Lorenzo Moneta, Chang Sun

CERN | NexTGen Next Generation Triggers

Activities in Next Generation Triggers and T1.3



Introduction

- To improve pattern recognition and keep up with increasing data rates in the trigger, aim to put ML models on hardware
- Hardware has limited resources, models should be compressed before implementation on hardware
- Much of model compression happens during training, before moving model to target hardware
- Many compression methods exist, but there is no common framework, no common implementation, no easy way to test and compare
- Our aim is to make it easier to use, study and test different model compression methods for NGT and broader communities

Goal and scope

- **Goal:**
  - Collect various compression methods
  - Develop common interface to users
  - Make it easy for users to adopt these methods for their models
  - Make it easy for users to test and compare methods

- **Scope:**
  - Develop software libraries and tools for hardware-aware training of neural networks (pruning, quantization etc.)
  - Develop general training loop that can be used with compression methods

Deliverables and milestones

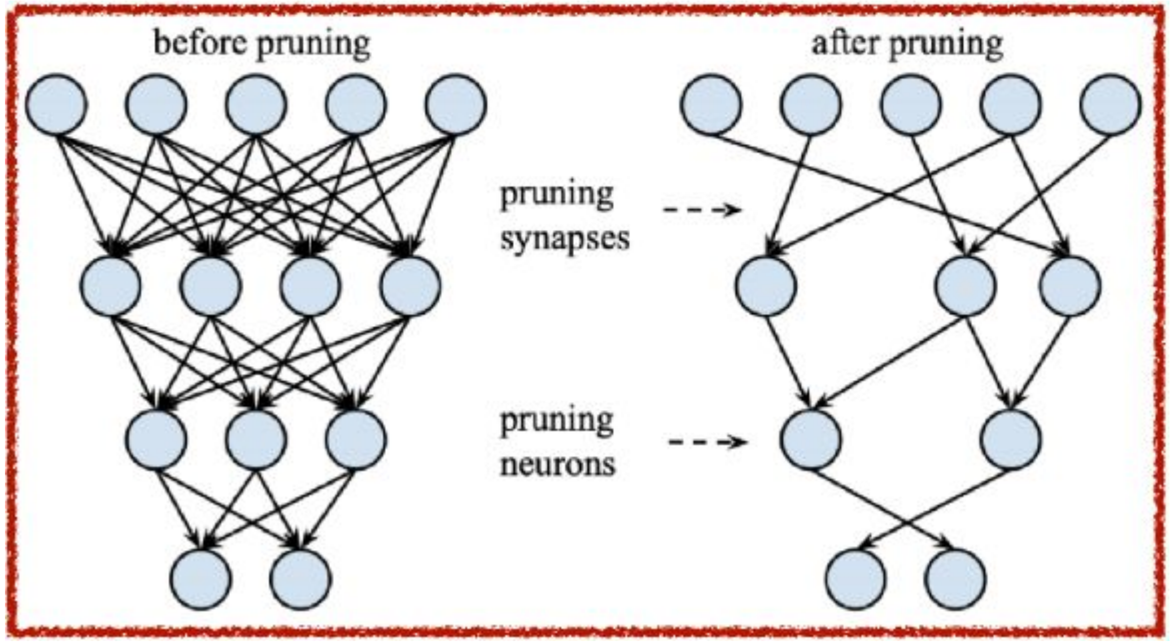
- **Internal milestones:**
  - Flexibility when choosing compression algorithms and target hardware

Description	Year
MLonFPGA community workshop, to identify needs from ATLAS and CMS on WP 2 and 3 as inputs to WP 1.2 and 1.3	1
WP1.3 software release 1 with open-access documentation	3
WP1.3 software release 2 with open-access documentation	4
WP1.3 software release 3 with open-access documentation	5

Time	**Description**	**Deliverable/Milestone**
6 m	Baseline development: large-scale training and optimization workflow on at least one end-to-end training library (Pytorch/Tensorflow)	Integration of the developed algorithms on the NNLO library (large-scale training package for CERN custom training workflow on HPC infrastructure)
12 m	Support of optimal workflows for hardware-aware pruning techniques with resource estimation.	- Demonstrator of network training and architecture scan for a concrete benchmark use case from WP2 or WP3 - NNLO tutorial showcasing novel functionalities - Journal publication
18 m	Support for Knowledge Distillation at training	integration of the developed compression workflows in the NNLO library
24 m	- AutoML-like flow towards automatic optimization of quantization and pruning at training time - Application of hardware-aware training on real-life use cases from WP2 and WP3	- Mid-point NNLO software release - Journal publication - NNLO tutorial showcasing novel functionalities
36 m	Hardware-aware NAS with quantization and sparsity	- Journal publication - NNLO tutorial showcasing novel functionalities
48 m	Extension of AutoML-like flow towards hardware-consumption prediction at training time	- Journal publication - NNLO tutorial showcasing novel functionalities
60 m	- Consolidation of ecosystem of compression models for edge deployments - Application of hardware-aware training on real-life use cases from WP2 and WP3	- Final NNLO release - Demonstrator of real-life use case from WP2 and WP3 - Journal publication

Pruning

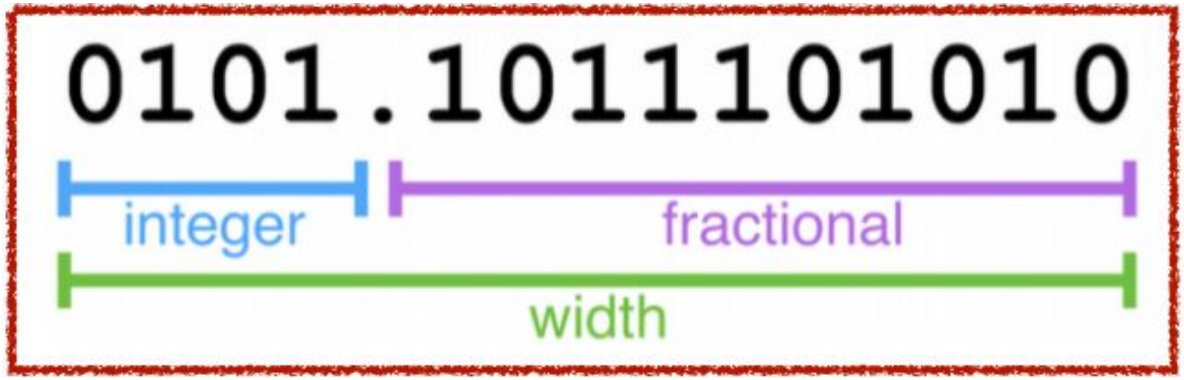
- Not all weights in a neural network are necessary
- Prune weights by setting them to 0. In general, this means fewer computations, resulting in reduced memory and resource usage
- Specifically for FPGAs, multiplications by 0 can be skipped
- Similar benefits for CPUs and GPUs
- Many ways to decide which weights to prune
- Granularity:
  - Prune single weights or groups of weights at once



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 accuracy high.**



The progress so far

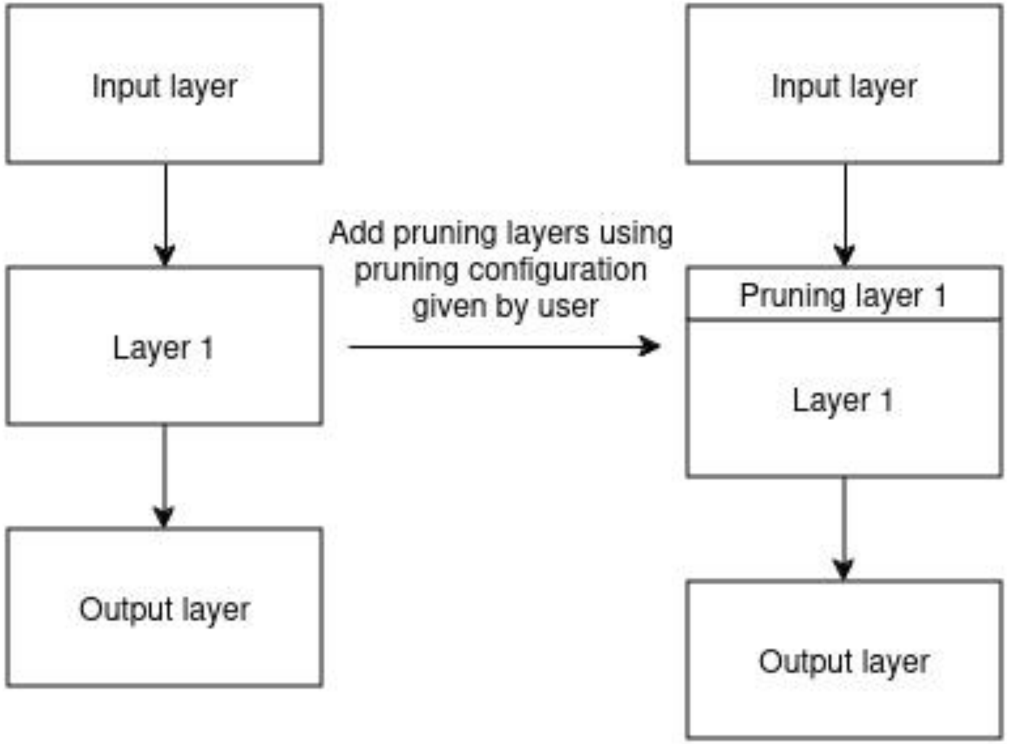
- **First focus of work: defining common interface and implementing pruning**
- **Common interface:**
  - YAML based configuration for pruning and training hyperparameters
  - User supplies the YAML configuration
  - Pruning layers are added **automatically**
- **Pruning:**
  - Identified a set of state-of-the-art methods
  - Selected promising subset to implement
  - 4 algorithms implemented so far,
  - Include default YAML configurations for easy use
- **Testing on common models such as ResNet and ParT**
  - Very interested to get more models from WP2 and WP3

Adding pruning layers to model

- **Pruning layers defined by YAML file:**
  - Which pruning method to use
  - Pruning hyperparameters

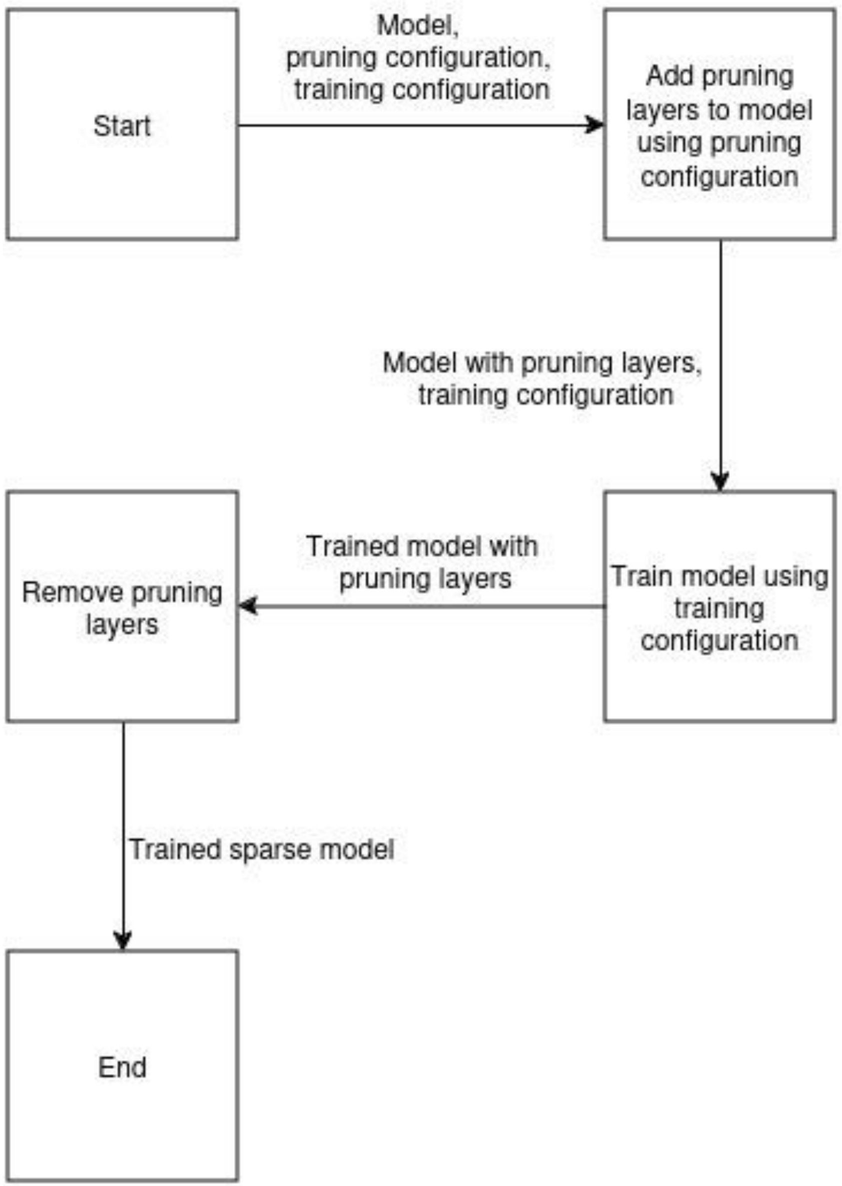
```yaml
pruning_parameters:
  epsilon: 0.015
  pruning_method: pdp
  sparsity: 0.8
  temperature: 1.0e-05
```

```yaml
pruning_parameters:
  final_temp: 200
  pruning_method: cs
  threshold_decay: 1.0e-09
  threshold_init: 0
```



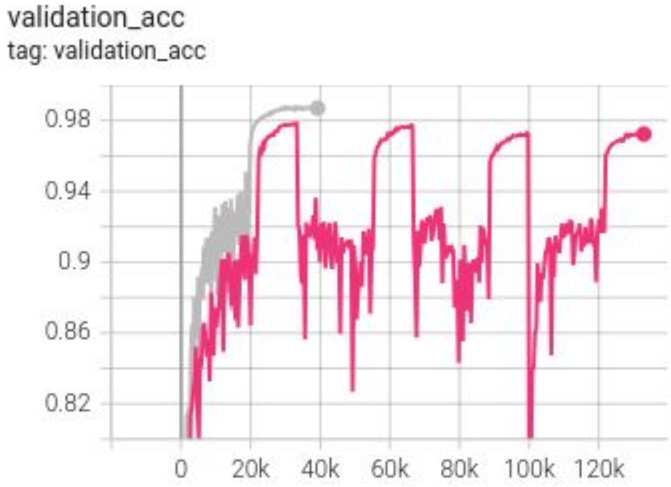
Training pruned models

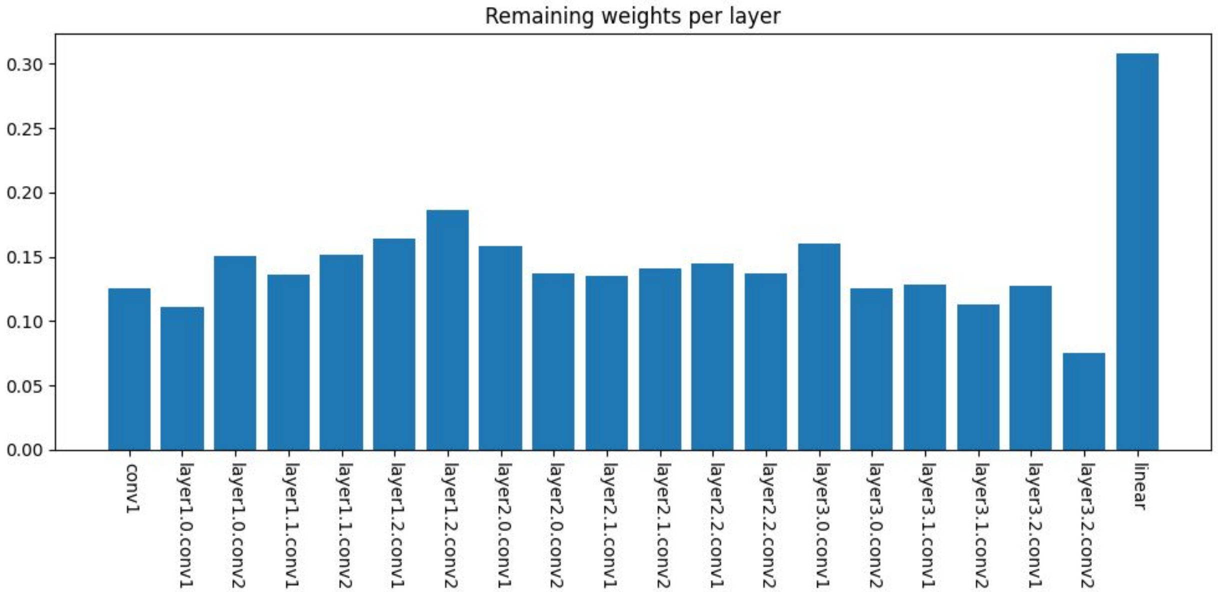
- **Different pruning methods have different training steps:**
    - Standard, multiple rounds, pre-training, fine-tuning steps
- **Training configuration defined by a YAML file**

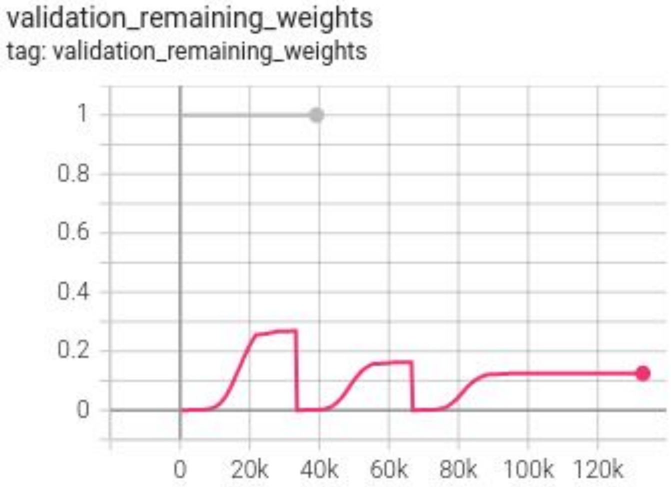


Example results

ResNet20, dataset CIFAR10
Pruning method: CS
Weights pruned: 87.55%

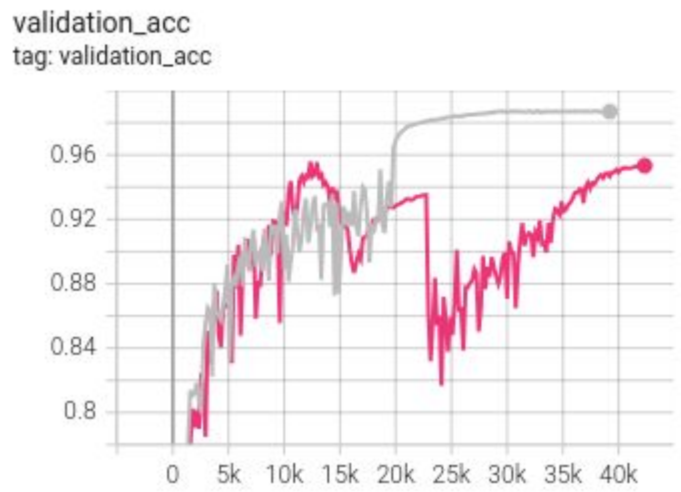


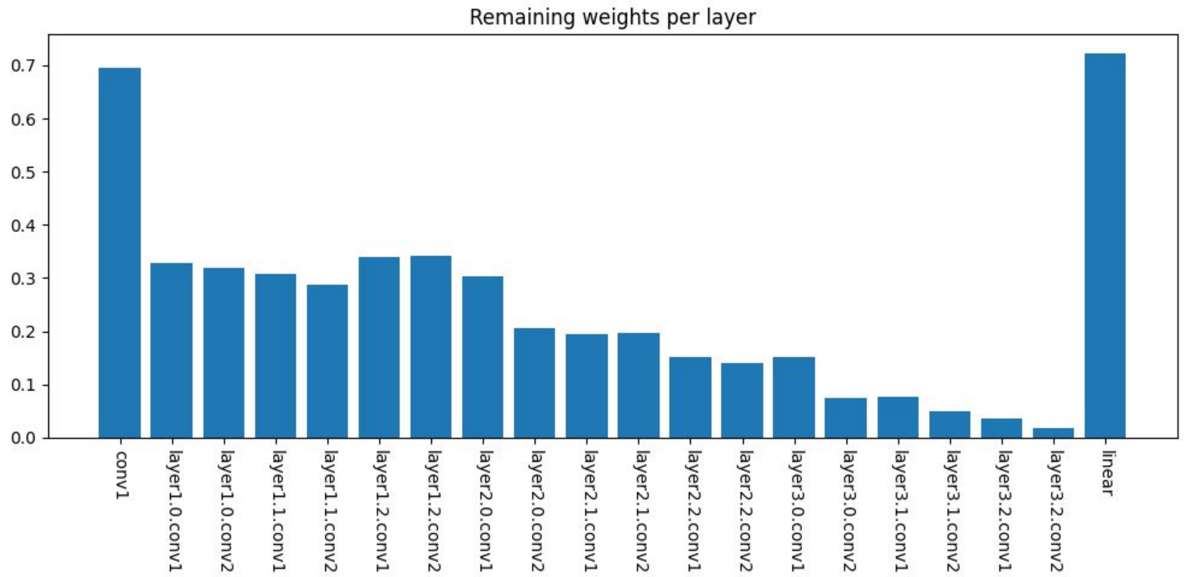


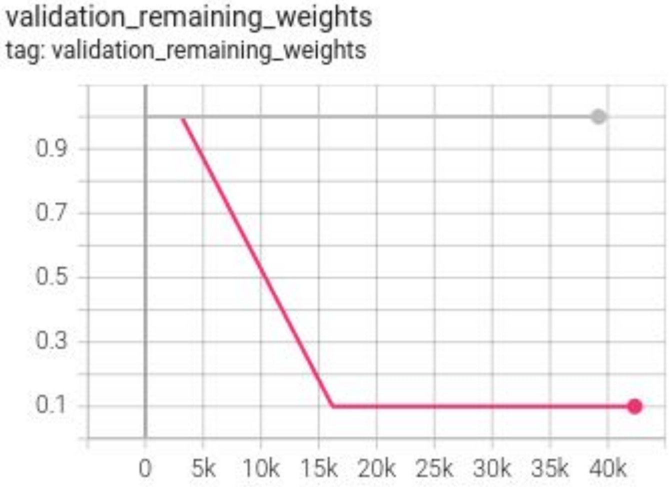


Example results

ResNet20, dataset CIFAR10
Pruning method: PDP
Weights pruned: 90%

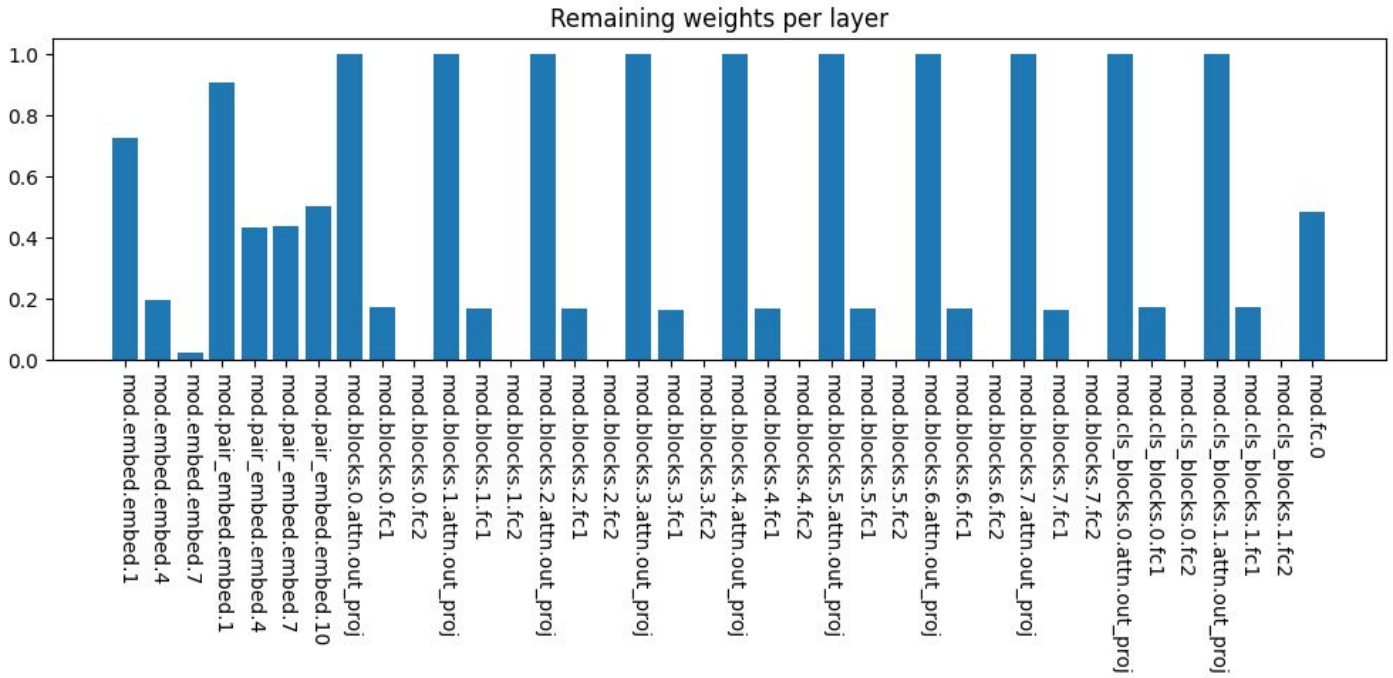


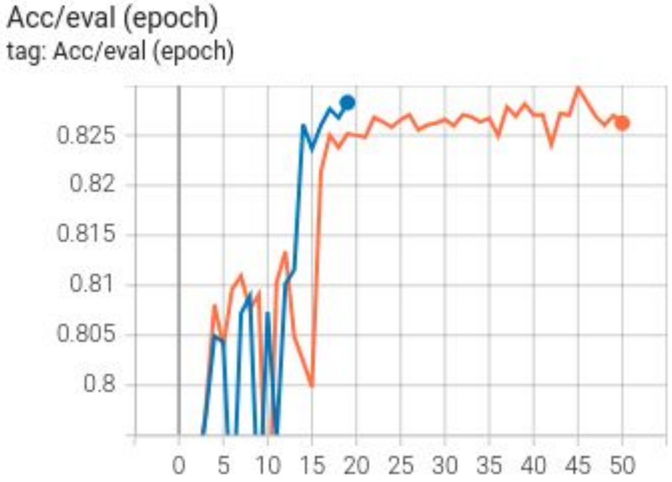


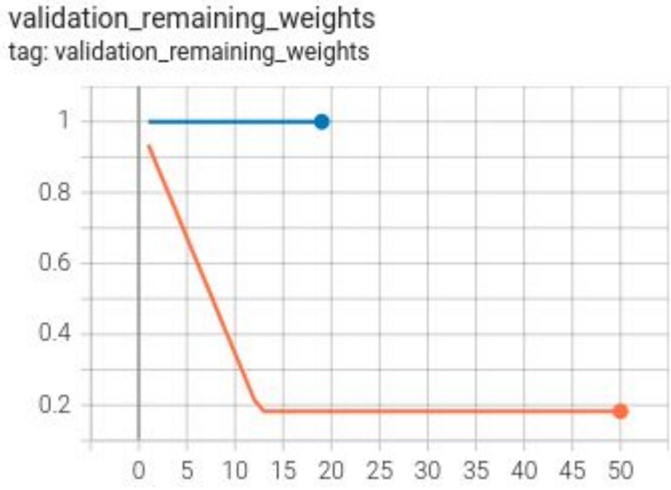


Example results

ParT
Pruning method: PDP
Weights pruned: 81.75%







Next steps

- **Short term:**
  - Test models from the community
  - Polish the library for release
  - Prepare documentation and tutorials
- **Medium term:**
  - Begin investigating and integrating quantization methods
  - Begin investigating hyperparameter optimization tools
  - Investigate custom training loops
  - Implement other compression methods, such as structured pruning

Conclusion

- ML models should be compressed before moving them to hardware, to optimize resource and memory usage

- Our goal is to implement various compression methods and develop a common interface to use them. We aim to make it easy for users to use these methods, and test and compare them

- We are interested in getting more models from WP2 and WP3, and discuss the training of models, training loops etc.