香港中文大學
The Chinese University of Hong Kong

CENG5030

Lab04 Introduction to Distiller

Qi Sun

(Latest update: February 20, 2021)

Spring 2021

Distiller

- Distiller is an open-source Python package (PyTorch environment) for neural network compression research.
- Comprehensive documentation and a mature forum.
- Example implementations of state-of-the-art compression algorithms.
- A friendly framework that you can add your own pruning, regularization and quantization algorithms easily.
- Supports of lots of mainstream DNN models and datasets, *e.g.*, SqueezeNet and ImageNet.

Using The Sample Application

An example python file is provided:

./examples/classifier_compression/compress_classifier.py

- Check all of the program options via `python ./compress_classifier.py -h`, including the pretrained models.
- You can try the Jupyter notebook to learn the usage of Distiller.
- Specify the algorithm configurations in a YAML file.

```
version: 1
pruners:
  my_pruner:
    class: 'SensitivityPruner'
    sensitivities:
      'features.module.0.weight': 0.25
      'features.module.3.weight': 0.35
      'classifier.1.weight': 0.875
```

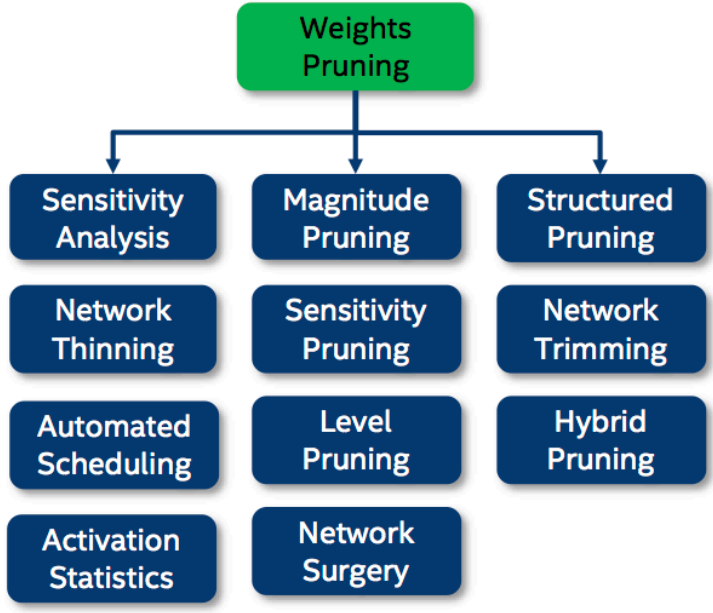
Pruning Sensitivity Analysis

Command flag

```
--sense = element or filter
```

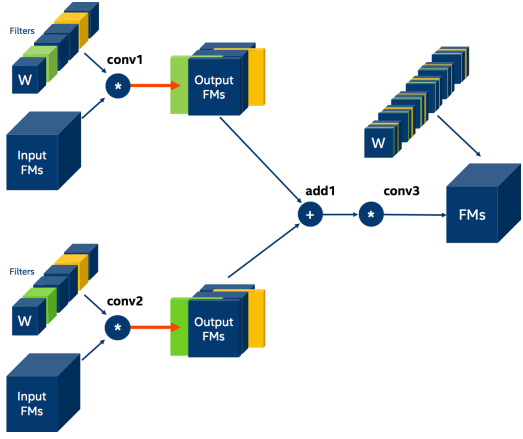
- Distiller supports element-wise and filter-wise pruning sensitivity analysis.
- In both cases, L1-norm is used to rank which elements or filters to prune.

Pruning Algorithms



Pruning Algorithms

- All of the pruning algorithms are defined in `./distiller/pruning`.
- Channel and filter pruning.
- Pay attention to the model structure to guarantee the pruning strategies are mutually compatible.



Magnitude Pruner

- It applies a thresholding function, $thresh(\cdot)$, on each element, w_i, of a weights tensor.
- Because the threshold is applied on individual elements, this pruner belongs to the element-wise pruning algorithm family.

$$thresh(w_i) = \begin{cases} w_i : \text{if} |w_i| > \lambda \\ 0 : \text{if} |w_i| \leq \lambda \end{cases} \tag{1}$$

Post-training Quantization

- It does not require any Policies nor a Scheduler.
- A checkpoint with the quantized model will be dumped in the run directory.
- It will contain the quantized model parameters (the data type will still be FP32, but the values will be integers).
- The calculated quantization parameters (scale and zero-point) are stored as well in each quantized layer.