Neural Network compression

NN compression methods

- Network compression is a widespread technique to reduce the size, energy consumption, and overtraining of deep neural networks

- Several approaches have been studied:

  - **parameter pruning:** selective removal of weights based on a particular ranking [arxiv.1510.00149, arxiv.1712.01312]

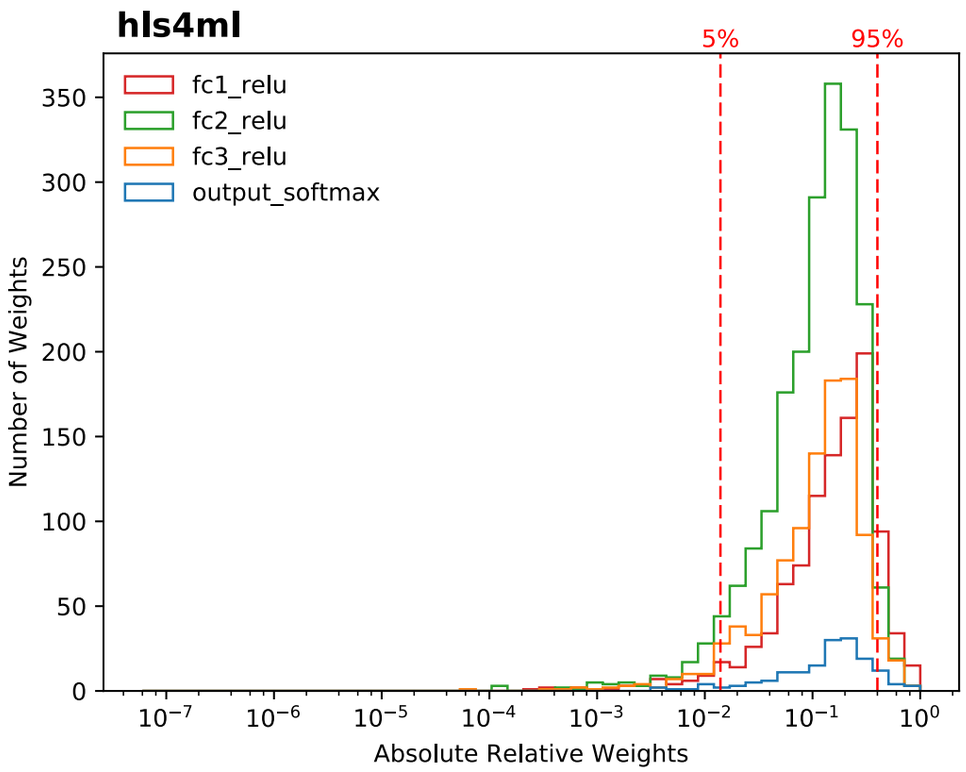
  - **low-rank factorization:** using matrix/tensor decomposition to estimate informative parameters [arxiv.1405.3866]

  - **transferred/compact convolutional filters:** special structural convolutional filters to save parameters [arxiv.1602.07576]

  - **knowledge distillation:** training a compact network with distilled knowledge of a large network [doi:10.1145/1150402.1150464]

- Today you'll learn about the first method: **parameter pruning**

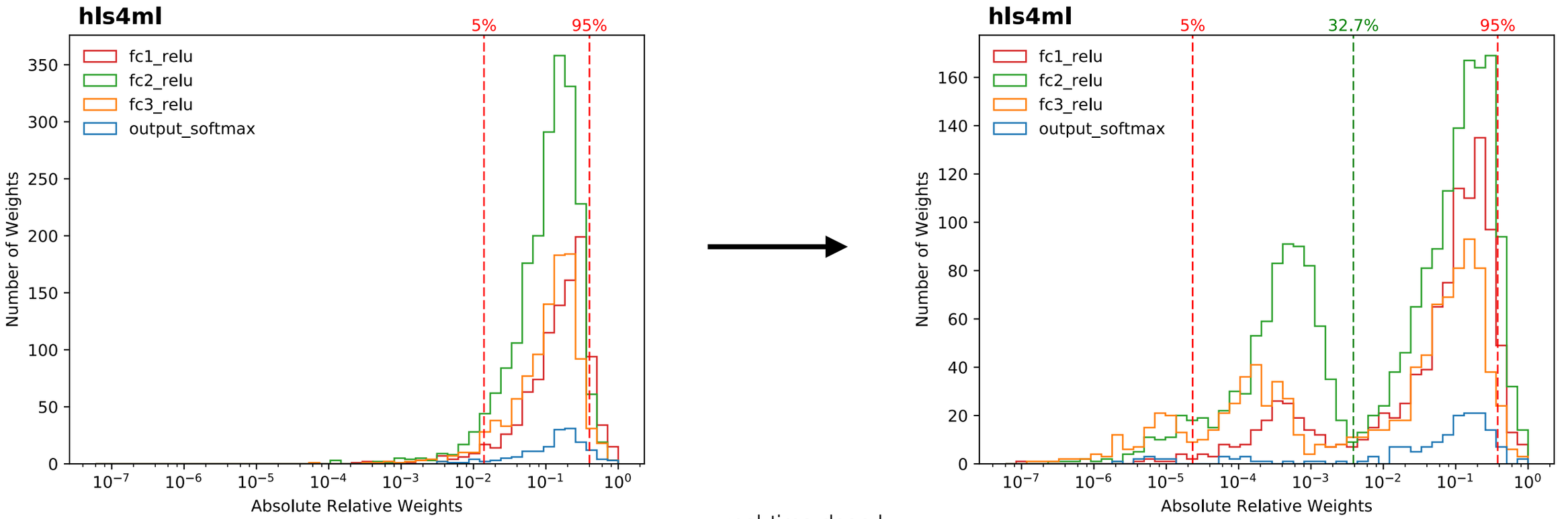
Compression with parameter pruning



Compression with parameter pruning

- Train with **L_1 regularization** (down-weights unimportant synapses)

$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

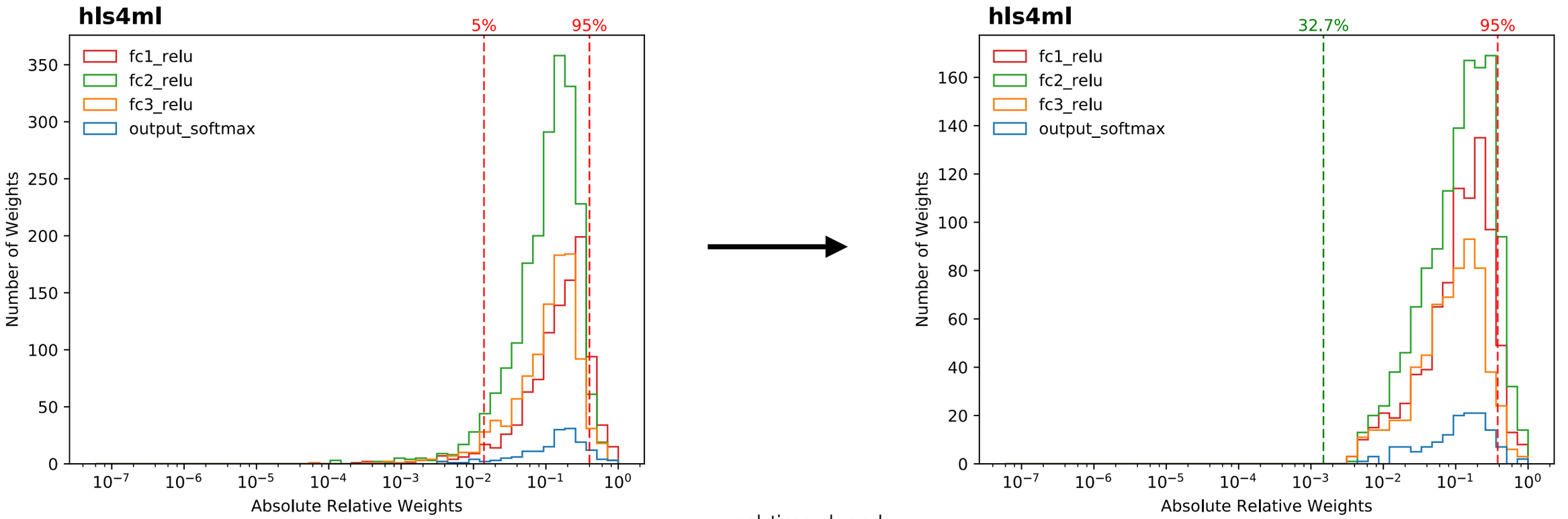


Compression with parameter pruning

- Train with **L_1 regularization** (down-weights unimportant synapses)

$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Remove **smallest** weights



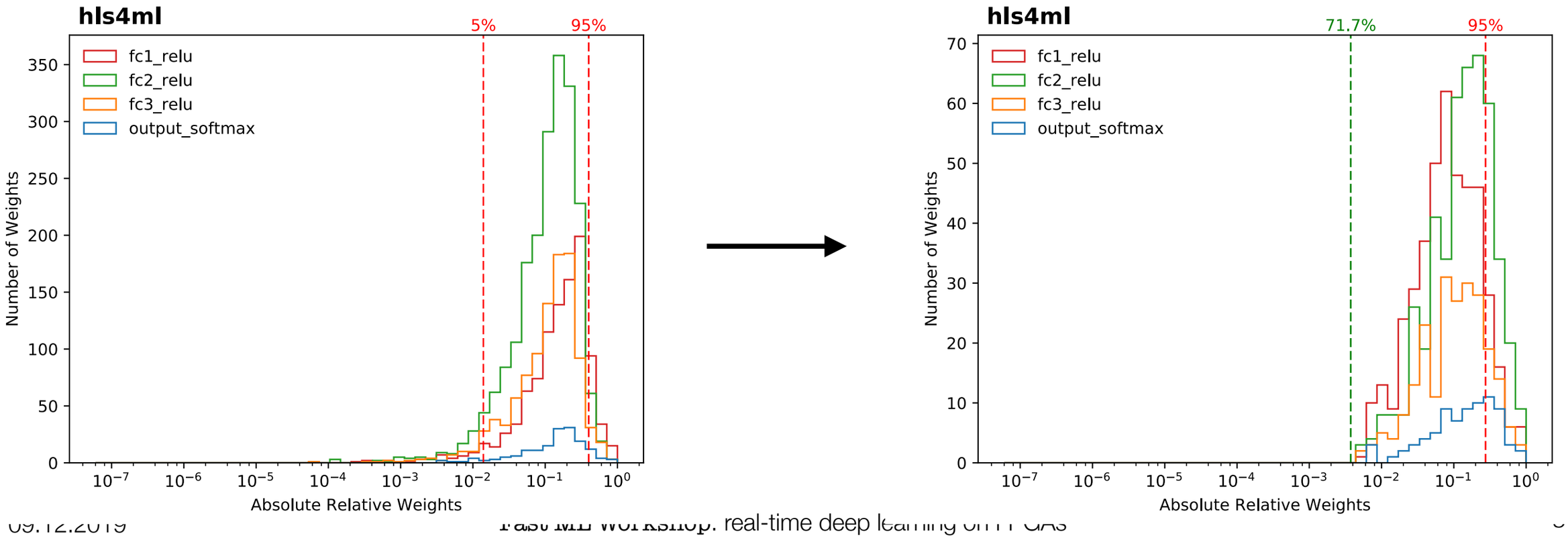
Compression with parameter pruning

- Train with **L_1 regularization** (down-weights unimportant synapses)

$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Remove **smallest** weights
- Iterate

70% REDUCTION OF WEIGHTS WITH NO LOSS IN PERF.



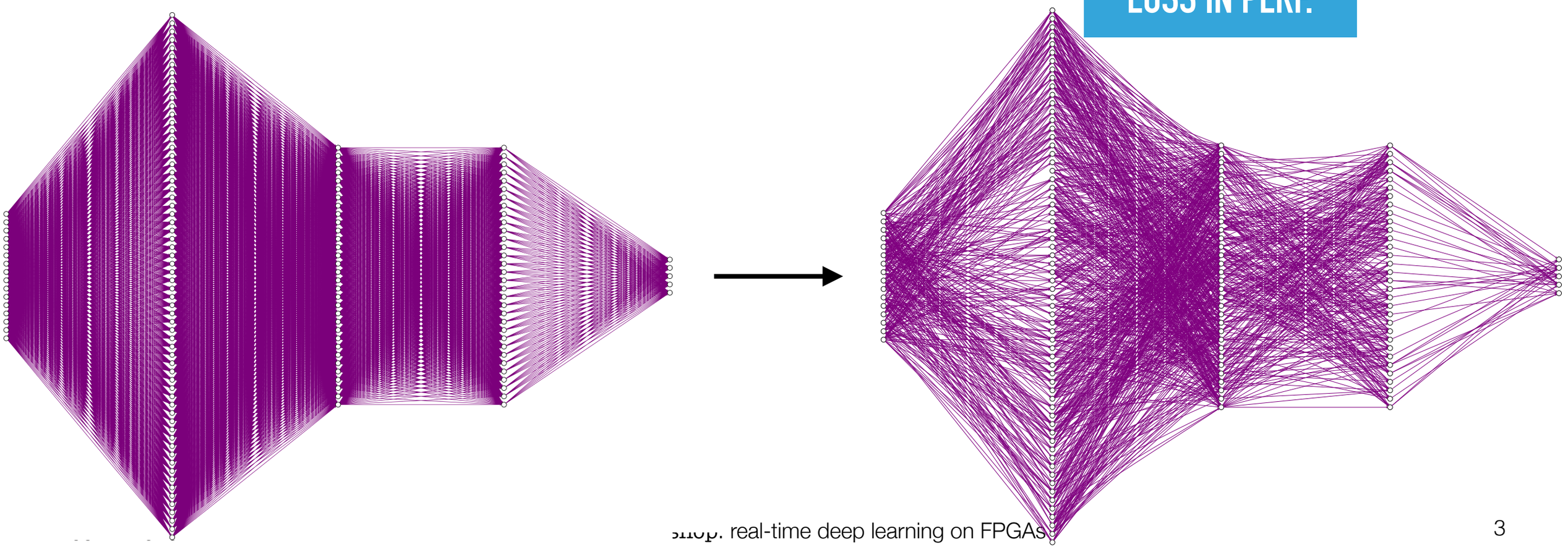
Compression with parameter pruning

- Train with **L_1 regularization** (down-weights unimportant synapses)

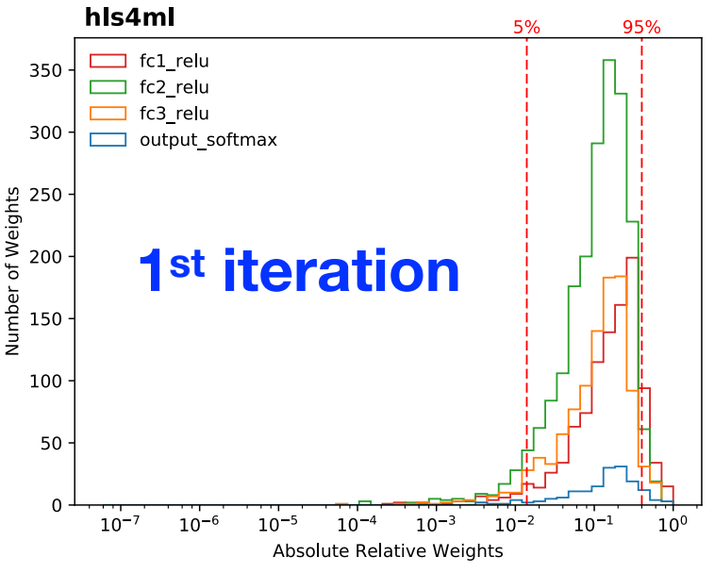
$$L_\lambda(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_1 \qquad \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Remove **smallest** weights
- Iterate

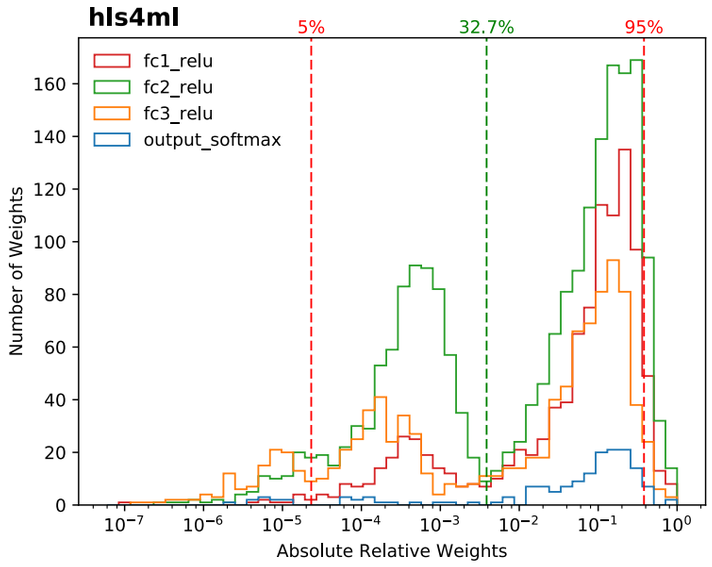
70% REDUCTION OF WEIGHTS WITH NO LOSS IN PERF.



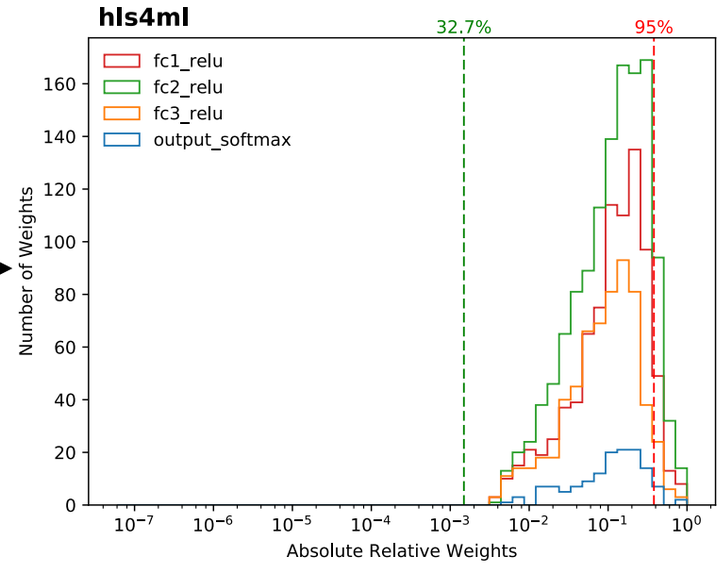
1st iteration



Train with L_1

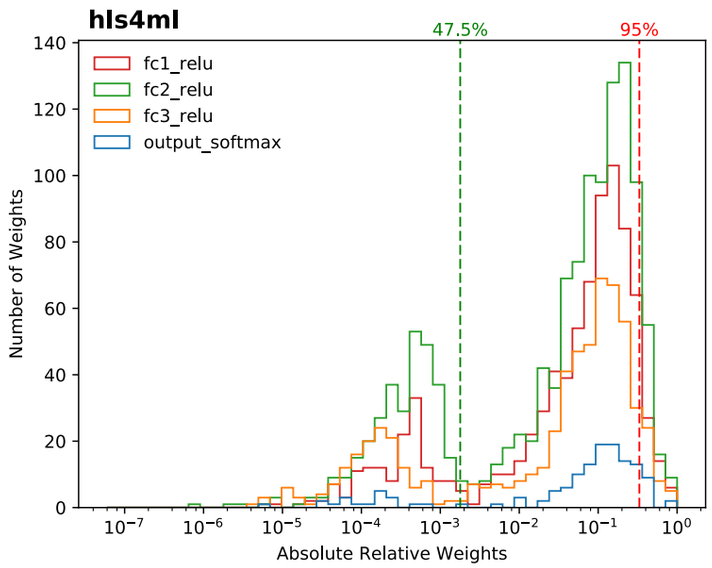


Prune

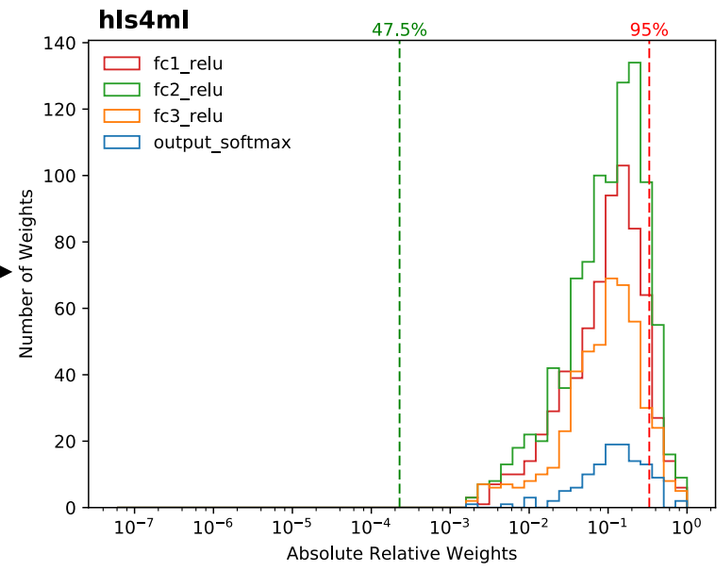


2nd iteration

Retrain with L_1



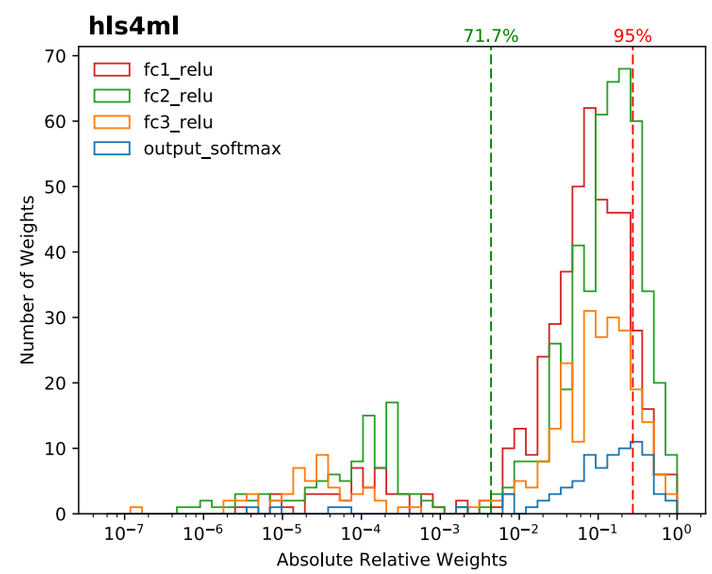
Prune



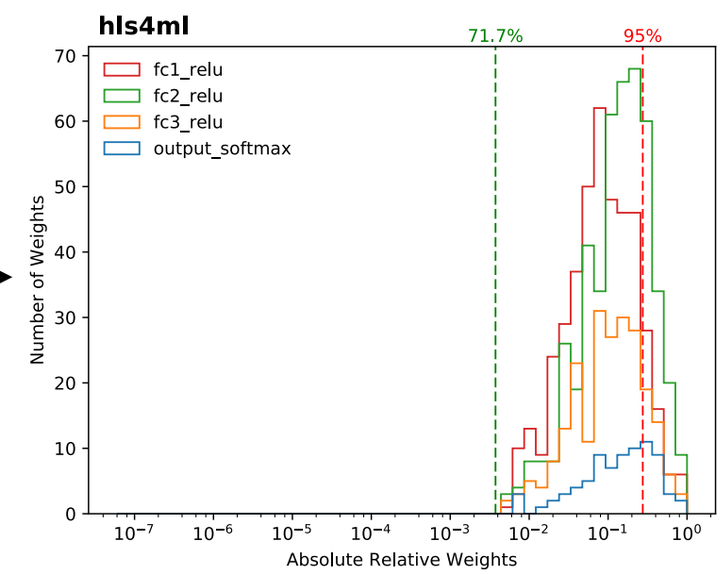
⋮ ⋮ ⋮

7th iteration

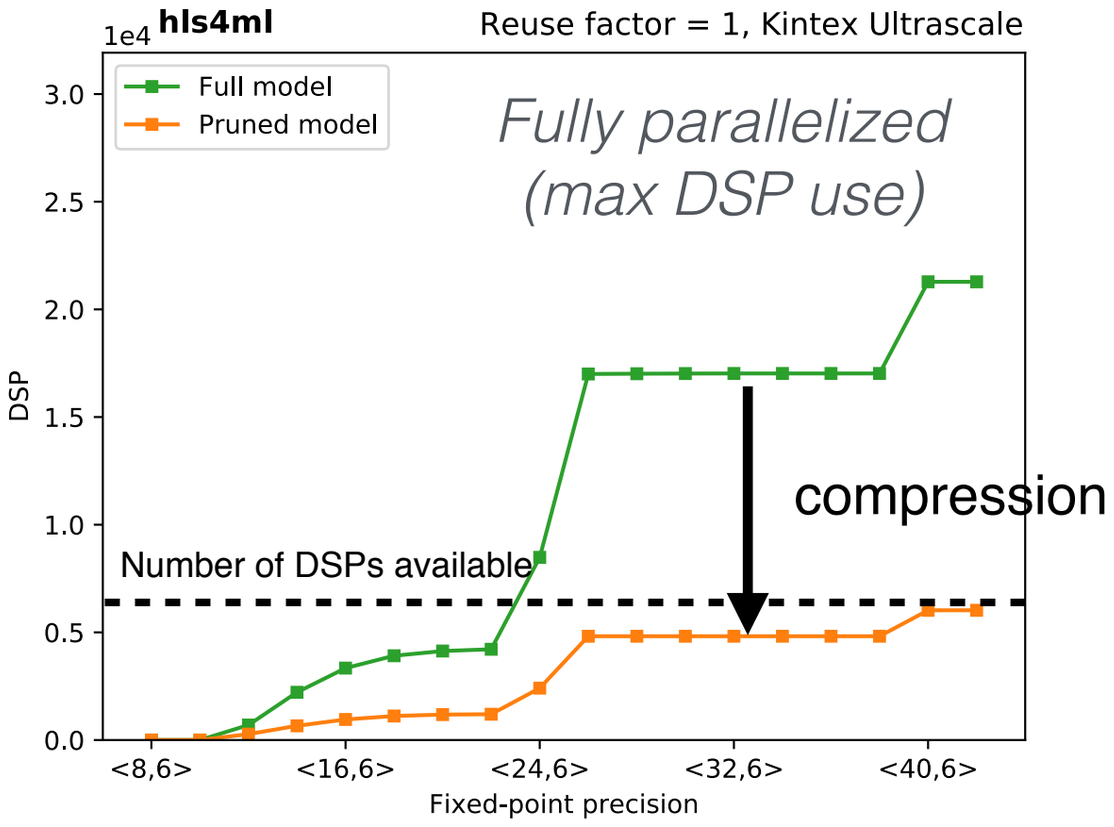
Retrain with L_1



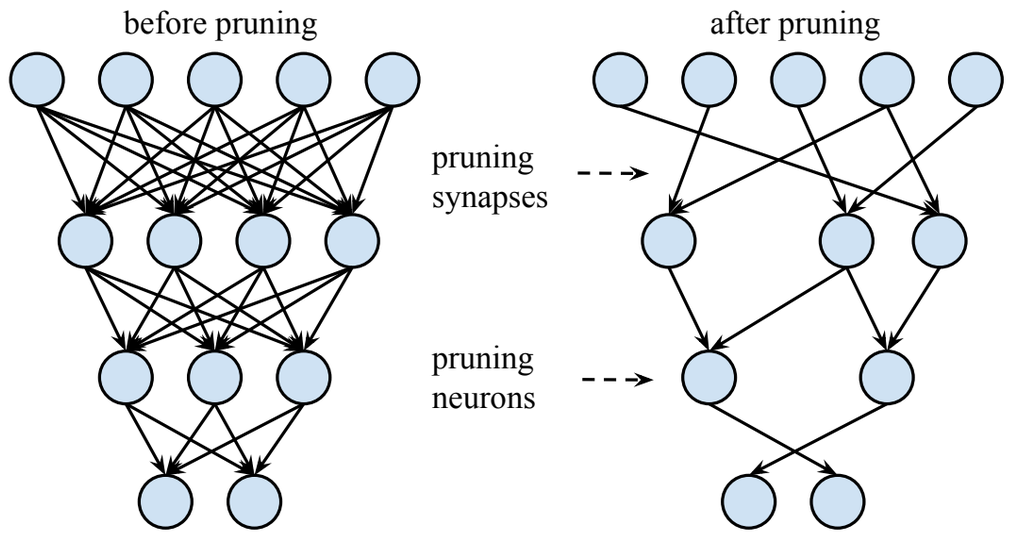
Prune



Efficient NN design: compression



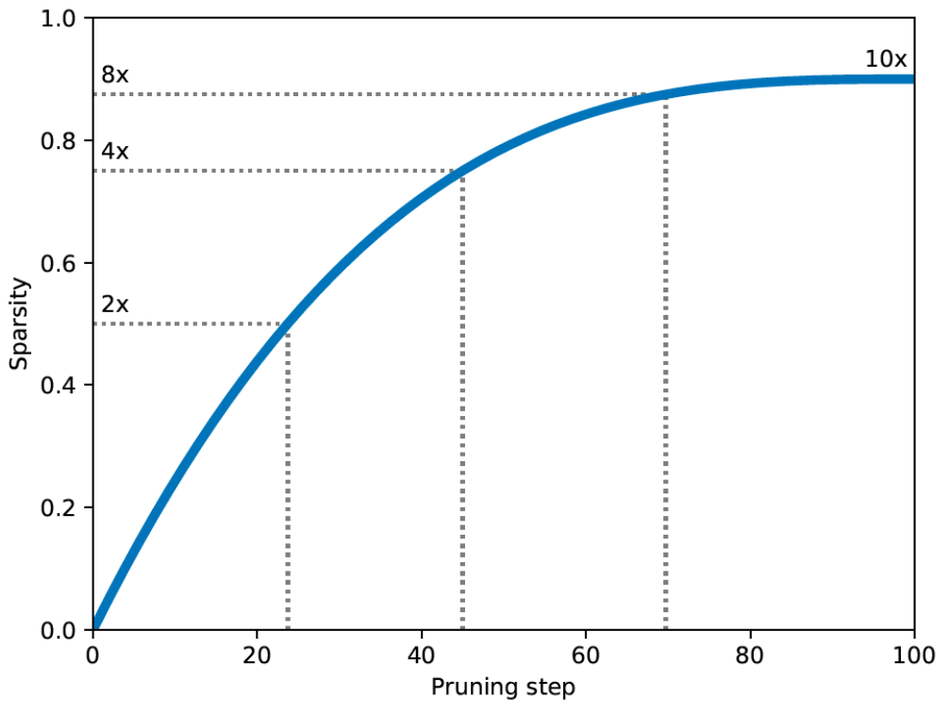
70% compression ~ 70% fewer DSPs

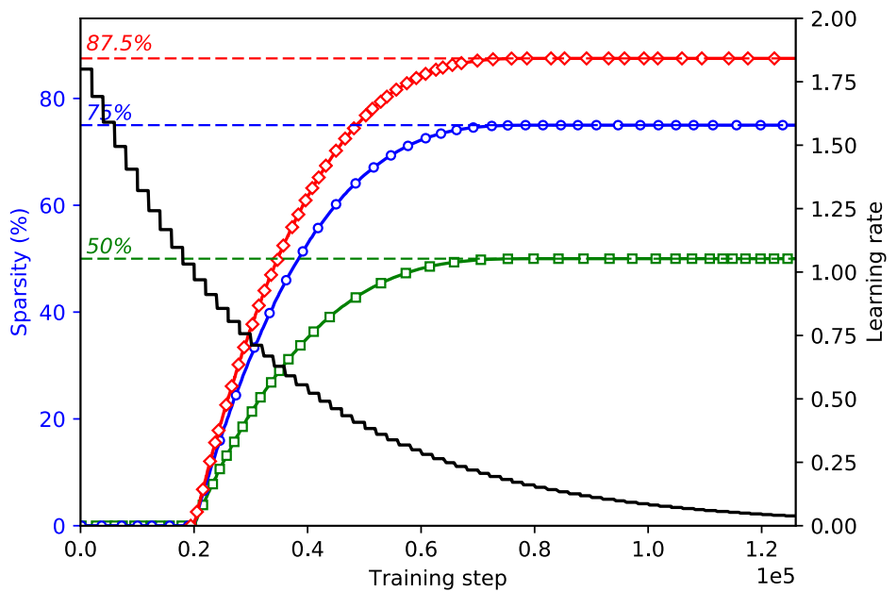


- DSPs (used for multiplication) are often limiting resource
  - maximum use when fully parallelized
  - DSPs have a max size for input (e.g. 27x18 bits), so number of DSPs per multiplication changes with precision

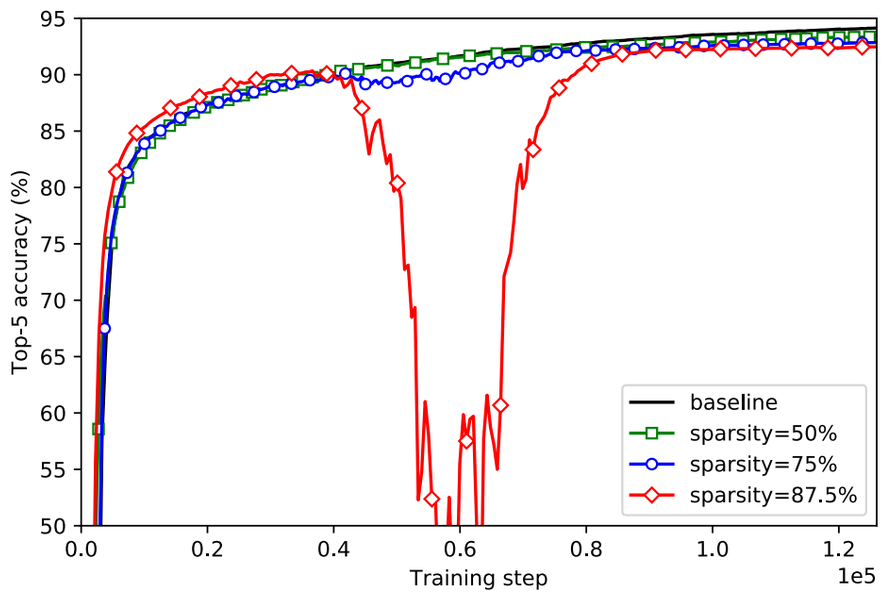
TensorFlow Model Optimization Toolkit

- TensorFlow Model Optimization Toolkit https://www.tensorflow.org/model_optimization
  - Uses magnitude-weight based pruning with polynomial decay schedule to increase sparsity during a single training run, see arXiv:1710.01878 and Medium blog post
  - Example MNIST MLP code: https://github.com/jmduarte/mnist-training





(a)



(b)

Figure 2: (a) The gradual sparsity function and exponentially decaying learning rate used for training sparse-InceptionV3 models. (b) Evolution of the model's accuracy during the training process