**Magnitude based pruning:**

Magnitude-based pruning is a pruning technique used in neural networks to reduce the complexity of the model by selectively removing certain weights or neurons based on their magnitudes. The idea is to identify and prune parameters that contribute less to the overall computation in the network.

* **Magnitude Calculation:** The magnitudes of the parameters in the network are calculated. The magnitude can be the absolute value of weights, the L1 norm, the L2 norm, or other metrics that capture the importance of a parameter.
* **Thresholding:** A threshold is set, and weights or neurons with magnitudes below this threshold are considered less important. The actual threshold calculation is handled internally by the torch\_pruning library, likely based on the magnitudes of the parameters (weights or channels) and the specified sparsity level. The library determines a threshold that corresponds to the specified sparsity level and prunes the parameters based on that threshold.
* **Pruning:** Parameters with magnitudes below the threshold are pruned. This can involve setting the weights to zero. When parameters (such as weights, neurons, or channels) fall below a certain threshold and are pruned, the common practice is to set those parameters to zero rather than completely removing them from the model. This is done to maintain the structure and shape of the model.

**Channel pruning:**

Pruning specifically on channels means that, during the pruning process, entire channels (feature maps or filters) in convolutional layers are selected for removal based on certain criteria. In this case, the criteria for selecting channels are determined by their magnitudes, calculated using the L2 norm.

* **Magnitude Calculation:** The magnitudes of the channels are calculated. The magnitude of a channel can be determined using various metrics, and in the provided code, the L2 norm (p=2) is used.
* **Thresholding:** A threshold is set based on the calculated magnitudes. This threshold determines which channels will be considered less important and, therefore, candidates for pruning.
* **Pruning:** Channels with magnitudes below the threshold are pruned. Pruning typically involves setting the entire channel (all the weights within the channel) to zero from the layer. This process effectively removes the filters associated with those channels.

**Code:**

  import torch\_pruning as tp

    model.eval()

    #print(model)

    example\_inputs = torch.randn(1, 3, 224, 224).to(device)

    imp = tp.importance.MagnitudeImportance(p=2) # L2 norm pruning

    ignored\_layers = []

    from models.yolo import Detect, IDetect

    from models.common import ImplicitA, ImplicitM

    for m in model.modules():

        if isinstance(m, (Detect,IDetect)):

            ignored\_layers.append(m.m)

    unwrapped\_parameters = []

    for m in model.modules():

        if isinstance(m, (ImplicitA,ImplicitM)):

            unwrapped\_parameters.append((m.implicit,1)) # pruning 1st dimension of implicit matrix

    iterative\_steps = 2 # progressive pruning

    pruner = tp.pruner.MagnitudePruner(

        model,

        example\_inputs,

        importance=imp,

        iterative\_steps=iterative\_steps,

        ch\_sparsity= ‘’, # remove 50% channels, ResNet18 = {64, 128, 256, 512} => ResNet18\_Half = {32, 64, 128, 256}

        ignored\_layers=ignored\_layers,

        unwrapped\_parameters=unwrapped\_parameters

    )

    for epoch in range(0,iterative\_steps):

        base\_macs, base\_nparams = tp.utils.count\_ops\_and\_params(model, example\_inputs)

        pruner.step()

        pruned\_macs, pruned\_nparams = tp.utils.count\_ops\_and\_params(model, example\_inputs)

        print(f"Iteration {epoch + 1}:")

        print("Before Pruning: MACs=%f G, #Params=%f G"%(base\_macs/1e9, base\_nparams/1e9))

        print("After Pruning: MACs=%f G, #Params=%f G"%(pruned\_macs/1e9, pruned\_nparams/1e9))

    #print(model)

**Defining the importance measure:**

imp = tp.importance.MagnitudeImportance(p=2)

MagnitudeImportance is a method to calculate the importance of parameters based on their magnitudes. Here, p=2 indicates L2 norm pruning, meaning the magnitude of parameters is calculated using the Euclidean norm.

**Identifying layers to ignore during pruning:**

ignored\_layers = []

for m in model.modules():

if isinstance(m, (Detect, IDetect)):

ignored\_layers.append(m.m)

This loop identifies layers of type Detect and IDetect in the model and adds them to the list of ignored layers.

**Identifying parameters to be pruned:**

unwrapped\_parameters = []

for m in model.modules():

if isinstance(m, (ImplicitA, ImplicitM)):

unwrapped\_parameters.append((m.implicit, 1))

This loop identifies layers of type ImplicitA and ImplicitM and adds their implicit matrices to the list of parameters to be pruned. The 1 in unwrapped\_parameters.append((m.implicit, 1)) indicates that the pruning will be applied to the first dimension of the implicit matrix.

**Pruner initialization:**

pruner = tp.pruner.MagnitudePruner(model, example\_inputs, importance=imp, iterative\_steps=iterative\_steps, ch\_sparsity=’’, ignored\_layers=ignored\_layers, unwrapped\_parameters=unwrapped\_parameters )

The MagnitudePruner is initialized with the model, example inputs, importance measure, number of iterative steps, channel sparsity (removing % of channels in this case), and the lists of ignored layers and parameters to be pruned.

**After pruning why, we need to perform fine-tuning:**

After pruning a neural network, fine-tuning is often required to recover or improve the performance of the model. Pruning removes certain connections or parameters from the network, making it more computationally efficient but potentially causing a loss of accuracy. Fine-tuning allows the model to adapt to the pruned structure and recover from any performance degradation caused by pruning. Here are some reasons why fine-tuning is necessary after pruning:

1. **Pruning:** Remove less important connections, channels, or parameters from the model to make it more computationally efficient. Pruning is a structural modification that does not involve training.
2. **Fine-Tuning:** Train the pruned model on the task-specific dataset. During fine-tuning, the model learns to compensate for the removed connections and adapts to the pruned structure. The learning rate during fine-tuning is often set to a smaller value to avoid large updates to the remaining weights.

**Multiply-Accumulate operations:**

MACs stands for Multiply-Accumulate operations, and it is a metric used to measure the computational cost of a neural network. In the context of YOLO (You Only Look Once), MACs is often used to estimate the number of floating-point operations required for the model to process an input.

MACs=number of output channels × height × width × kernel size × kernel size × number of input channels

In YOLO, the model is composed of multiple convolutional layers, fully connected layers, and other operations. The total MACs for the entire YOLO model is the sum of MACs across all layers.

The MACs value is a useful metric for understanding the computational efficiency of a neural network, and it's often used to compare different network architectures. Lower MACs can indicate a more computationally efficient model, which is important for real-time applications like object detection where inference speed is crucial.