Inputs: CNN – a CNN model

– an optimization dataset

– maximal acceptable loss in classification accuracy

mode – mode of optimization, i.e. max granularity, uniform layer, uniform patch or uniform filters

ps – patch size

patterns– all possible patterns. Patterns can be selected directly or by specifying a range of number of ‘1’s in the patch

gran – granularity

outputs: mask – final mask

LQrec – LayerQuantizier stage results

CQrec – ChannelQuantizier stage results

PQrec – PatchQuantizier stage results

FLrec – first level results

initialize Optimizer(CNN, , , mode, ps, patterns, gran)

FLrec = Optimizer.FirstLevelOptimizationPass()

PQrec = Optimizer. PatchQuantizierPass(FLrec)

CQrec = Optimizer. ChannelQuantizierPass(PQrec)

LQrec = Optimizer. LayerQuantizierPass(CQrec)

mask = FindFinalMask(LQrec, )

function FirstLevelOptimizationPass(Optimizer)

initialize FLrec[layer][channel][patch][pattern] None

for layer in CNN do

for channel in do

for patch in do

for in patterns do

tmp\_mask = CreateFirstLvlMask(mode,)

ops\_saved, err = Simulate(CNN, , tmp\_mask)

FLrec[] = (ops\_saved, err)

return FLrec

function PatchQuantizierPass(Optimizer , FLrec)

for record in FLrec do

if err then

remove record

sort FLrec based on ops\_saved

initialize PQrec[layer][channel][pattern] None

for layer in CNN do

for channel in do

for in range(0, #patterns) do

channel\_pattern = CreateChannelPattern(mode, , FLrec[])

ops\_saved, err = Simulate(CNN, , channel\_pattern)

PQrec [] = (ops\_saved, err)

return PQrec

function LayerQuantizierPass(Optimizer, CQrec)

for record in CQrec do

if err then

remove record

sort CQrec based on ops\_saved

initialize LQrec[pattern] None

mask = CreateNetworkPattern(mode, CQrec[:][0])

while mask is not None do

ops\_saved, err = Simulate(CNN, , mask)

LQrec[mask] = (ops\_saved, err)

mask = UpdateMask(mask, CQrec)

return LQrec

function ChannelQuantizierPass(Optimizer , PQrec)

for record in PQrec do

if err then

remove record

sort PQrec based on ops\_saved

initialize CQrec[layer][pattern] None

for layer in CNN do

for in range(0, #patterns) do

layer\_pattern = CreateLayerPattern(mode, CQrec[])

ops\_saved, err = Simulate(CNN, , layer\_pattern)

CQrec [] = (ops\_saved, err)

return CQrec

function UpdateMask(mask, CQrec)

for layer in CNN do

if CQrec[[curr\_pattern+1] is not default pattern then

// default pattern, i.e. ops\_saved == 0

acc\_diff = CQrec[[curr\_pattern+1].err - CQrec[[curr\_pattern+1].err

curr\_ops\_saved = Sum(CQrec[:][curr\_pattern].ops\_saved)

next\_ops\_saved = curr\_ops\_saved - CQrec[][curr\_pattern].ops\_saved

+ CQrec[][curr\_pattern+1].ops\_saved

ops\_diff = (curr\_ops\_saved - next\_ops\_saved + 1)/( curr\_ops\_saved + 1)

merit[] = acc\_diff/ops\_diff

if merit is not None then

update layer in mask according to Max(merit)

return mask

else

if all layers are set to default pattern then

return None

else

update last layer in mask to default pattern

return mask