out = np.zeros([N,C,int(H),int(W)])

#ROTATE THE WEIGHTS

#=======================

#print('w = ', w)

w\_rot180 = w \* 0

#print('w\_rot180 shape = ', w\_rot180.shape)

for i in range(F):

for j in range(C):

w\_rot180[i,j,:,:] = np.rot90(w[i,j,:,:], 2)

#print('w\_rot180 = ', w\_rot180)

#RACK AND STACK WEIGHTS INTO ROW VECTORS

#=============================

filter\_w = np.zeros([HH, WW])

w\_row = np.zeros([F,HH\*WW\*C])

for ii in range(F):

for iii in range(C):

filter\_w = w\_rot180[ii,iii,:,:]

#print('filter\_w = ', filter\_w)

#print('filter\_w shape = ', filter\_w.shape)

filter\_w = np.reshape(filter\_w, [1,HH\*WW])

#print('filter\_w = ', filter\_w)

w\_row[ii,(iii\*HH\*WW):(iii\*HH\*WW)+HH\*WW] = filter\_w

#print('w\_row = ', w\_row)

#print('w\_row shape = ', w\_row.shape)

#INITIALIZE COLUMN SIZE

X\_col = np.zeros([int(H)\*int(W),HH\*WW])

#INITIALIZE PADDED MATRIX

x\_pad = np.zeros([1,(int(H)+(pad\*2))\*(int(W)+(pad\*2))])

x\_pad = np.reshape(x\_pad, [(H+(pad\*2)), (W+(pad\*2))])

#print('x\_pad = ', x\_pad)

#print('x\_pad shape = ', x\_pad.shape)

conv = np.zeros([F,C,H,W])

conv\_out = np.zeros([C,H,W])

#CONVOLVE

#=============================

for i in range(N): #NUMBER OF SAMPLES

for ff in range (F): #NUMBER OF FILTERS

for yy in range(C):

#PLACE X DATA INTO PADDED MATRIX

x\_pad[pad:x\_pad.shape[0]-pad,pad:x\_pad.shape[1]-pad] = dout[i,ff,:,:]

padded\_x = x\_pad

#print('x\_pad = ', x\_pad)

# INITIALIZE COUNTERS

loc\_counter = 0

j = 0

k = 0

horz\_count = 0

vert\_count = 0

#RACK AND STACK INPUT DATA INTO COLUMNS

while vert\_count < int(OH):

while horz\_count < int(OW):

X\_block = padded\_x[j:j+HH,k:k+WW]

#print('X\_block shape = ', X\_block.shape)

#print('X\_block= ', X\_block)

X\_block\_col = np.reshape(X\_block,(HH\*WW))

#print('X\_block\_col shape = ', X\_block\_col.shape)

#print('X\_block\_col = ', X\_block\_col)

X\_col[loc\_counter,:] = X\_block\_col

k = k + stride

#print('k = ', k)

#print('loc\_counter = ', loc\_counter)

loc\_counter = loc\_counter + 1

horz\_count = horz\_count + 1

#print('horz\_count = ', horz\_count)

k = 0

horz\_count = 0

#print('k = ', k)

j = j + stride

#print('j = ', j)

vert\_count = vert\_count + 1

#print('vert\_count = ', vert\_count)

#X\_col = ([int(OH)\*int(OW),HH\*WW])

#w\_row = ([F,HH\*WW\*C])

#DOT PRODUCT OF FILTER AND X DATA

dot = np.dot(w\_row[ff,HH\*WW\*yy:(HH\*WW\*(yy+1))], np.transpose(X\_col))

dot = np.reshape(dot, [1,H,W])

#print('dot shape = ', dot.shape)

conv[ff,yy,:,:] = dot

#print('conv shape = ', conv.shape)

conv\_out = np.sum(conv, axis = 0)

#print('conv\_out = ', conv\_out)

#print('conv\_out shape = ', conv\_out.shape)

out[i,:,:,:] = conv\_out

#print('out shape = ', out.shape)

#print('out = ', out)

#print('out = ', out)

dx = out

#print('dx = ', dx)

#FIND DW

#=============================

#=============================

#ROTATE THE WEIGHTS

#=======================

#print('dout = ', dout)

dout\_rot180 = np.zeros(dout.shape)

#print('dout\_rot180 shape = ', dout\_rot180.shape)

for i in range(dout.shape[0]):

for j in range(dout.shape[1]):

dout\_rot180[i,j,:,:] = np.rot90(dout[i,j,:,:], 2)

#print('dout\_rot180 = ', dout\_rot180)

#RACK AND STACK WEIGHTS INTO ROW VECTORS

#=============================

filter\_w = np.zeros([OH, OW])

w\_row = np.zeros([N,F,OH\*OW])

#print('w\_row shape = ', w\_row.shape)

for i in range (N):

for ii in range(F):

filter\_w = dout[i,ii,:,:]

#print('filter\_w = ', filter\_w)

#print('filter\_w shape = ', filter\_w.shape)

filter\_w = np.reshape(filter\_w, [1,OH\*OW])

#print('filter\_w = ', filter\_w)

#w\_row[i,ii,(iii\*HH\*WW):(iii\*HH\*WW)+HH\*WW] = filter\_w

w\_row[i,ii,:] = filter\_w

#print('w\_row = ', w\_row)

#print('w\_row shape = ', w\_row.shape)

#INITIALIZE COLUMN SIZE

X\_col = np.zeros([int(OH)\*int(OW),HH\*WW])

#INITIALIZE PADDED MATRIX

x\_pad = np.zeros([1,(int(H)+(pad\*2))\*(int(W)+(pad\*2))\*1])

x\_pad= np.reshape(x\_pad, [1,(int(H)+(int(pad)\*2)), (int(W)+(int(pad)\*2))])

#print('x\_pad = ', x\_pad)

#print('x\_pad shape = ', x\_pad.shape)

conv = np.zeros([N,F,C,HH,WW])

conv\_out = np.zeros([F,C,HH,WW])

#CONVOLVE

#=============================

for i in range(N): #NUMBER OF SAMPLES

for ff in range (F): #NUMBER OF FILTERS

for yy in range(C):

#PLACE X DATA INTO PADDED MATRIX

x\_pad[0,int(pad):x\_pad.shape[1]-int(pad),int(pad):x\_pad.shape[2]-int(pad)] = x[i,yy,:,:]

padded\_x = x\_pad

# INITIALIZE COUNTERS

loc\_counter = 0

j = 0

k = 0

horz\_count = 0

vert\_count = 0

#RACK AND STACK INPUT DATA INTO COLUMNS

while vert\_count < int(OH):

while horz\_count < int(OW):

X\_block = padded\_x[0,j:j+HH,k:k+WW]

#print('X\_block shape = ', X\_block.shape)

#print('X\_block= ', X\_block)

X\_block\_col = np.reshape(X\_block,(1,HH\*WW))

#print('X\_block\_col shape = ', X\_block\_col.shape)

#print('X\_block\_col = ', X\_block\_col)

X\_col[loc\_counter,:] = X\_block\_col

k = k + stride

#print('k = ', k)

#print('loc\_counter = ', loc\_counter)

loc\_counter = loc\_counter + 1

horz\_count = horz\_count + 1

#print('horz\_count = ', horz\_count)

k = 0

horz\_count = 0

#print('k = ', k)

j = j + stride

#print('j = ', j)

vert\_count = vert\_count + 1

#print('vert\_count = ', vert\_count)

#print('X\_col = ', X\_col)

#print('X\_col shape = ', X\_col.shape)

#print('w\_row shape = ', w\_row.shape)

#DOT PRODUCT OF FILTER AND X DATA

#w\_row = ([N,F,OH\*OW])

#X\_col = [int(OH)\*int(OW),HH\*WW])

dot = np.dot(np.transpose(X\_col), w\_row[i,ff,:])

dot = np.reshape(dot, [int(HH),int(WW)])

#print('dot0 shape = ', dot0.shape)

conv[i,ff,yy,:,:] = dot

conv\_out = np.sum(conv, axis = 0)

dw = conv\_out

#print('dw = ', dw)

#print('dw shape = ', dw.shape)