Supplementary materials

The following provides complete Python source code, as used to generate Figure 1, and may serve as an instructive example of how the model works. To use the code below, you need a working copy of Python (version 2.x), the numpy library, and the matplotlib library. One easy way to install these and additional libraries is simply to install the PsychoPy application, which includes the libraries and a code editor to insert and run the code.

from pylab import *
#attributes of layer 1 neurons
g_A = g_B = 100
c50_A = c50_B = 0.1
exp_A = exp_B = 2.0
#attributes of layer 2 neuron
c50_C = 100
exp_C = 2.0
g_C = 100
#contrasts
cl, c2 = mgrid[0:1:100j, 0:1:100j]

cmapVector = linspace(0.3,0.8,256).
reshape([256,1]).repeat(3,1)
cmap = matplotlib.colors.ListedColormap
cmapVector)
def doPlot(resp):
    #plot the greyscale response image
    p = imshow(resp, cmap=cmap, origin=‘lower’)
gca().set_aspect(1)#force square axes
    monoMax = (resp[0,:]).max()+5#choose a decision boundary
    contour(resp, levels=[monoMax],
colors=‘k’) #plot decision boundary
    xticks([0,25,50,75,100],[0,0.25,
           0.50,0.75,1.0])
    yticks([0,25,50,75,100],[0,0.25,
           0.50,0.75,1.0])
    return p

def NR(x, c50, expon):
    """Return the Naka-Rushton sigmoid of the x value(s)""
    return x**expon/(x**expon +c50**expon)

#create figure page
figure(figsize=[14,5])

#plot linear summation
subplot(1,3,1)
resp = NR(c1*g_A+c2*g_B, c50_C, exp_C)*g_C
doPlot(resp)
title(‘Linear summation’)
ylabel(‘Channel 2 contrast’)

#plot nonlinear summation
subplot(1,3,2)
respA = NR(c1,c50_A, exp_A)*g_A
respB = NR(c2,c50_B, exp_B)*g_B
resp = NR(respA+respB, c50_C, exp_C)*g_C
#resp = (respA+respB)*g_C
doPlot(resp)
title(‘Nonlinear summation’)
xlabel(‘Channel 1 contrast’)

#plot multiplicative combination
subplot(1,3,3)
respA = c1*g_A
respB = c2*g_B
resp = NR(respA*respB, c50_C, exp_C)*g_C
fig = doPlot(resp)
title(‘Multiplication’)

#save and display figure
savefig(‘ANDgates.eps’) show()