

```
In [1]: #Figure creation for CNN and NAS comparison paper

# Date Created: 09-11-2021
# Created by Matthew Hall
# Last Edited: 04-03-2022

import tensorflow as tf
import numpy as np
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Dense, concatenate

import keras as keras
from keras import backend as K
from keras.models import Sequential
from keras.layers import Dense, Activation, Reshape
from keras.layers import Conv1D, MaxPooling1D, Flatten, GlobalAveragePooling1D, Dropout

import os
from os.path import dirname, join as pjoin
import scipy.io as sio
import h5py

import matplotlib as mpl
import matplotlib.pyplot as plt

import matplotlib.pyplot
from scipy.fft import fft, fftfreq

from datetime import datetime
import random
```

```
In [2]: # Networks to be tested

model_dir = 'Z:\\\\SmithLab\\\\prjNAS\\\\NASNet'

limiter =150
gamma = 0.15
gammas = np.arange(1,10)/10
gammas =0.15
numLines =800
ntimepts = 52

netnames =[]
numcnn =1

netnames.append('SmithCNNnet') #best trained network
netnames.append('NAS_full') #original nas network
```

```
In [3]: # evaluate mat files as one large chunk
#data returned was used to determine the realtime test across a random sample of all co

data_dir = 'Z:/SmithLab/prjNAS/NASNet/test_all/' #all areas in one file
testingFile = []
for file in os.listdir(data_dir): # pull all training files from directory
```

```

    if file.endswith(".mat"):
        testingFile.append(file)
nset = len(testingFile) # singular testing file

#Time testing on full file (random choice over all areas)

#print('Testing on ' + str(nsets) + ' files.')
for j in range(nset): # loop over the different testing files
    f_waves = h5py.File(data_dir + '/' + testingFile[j], mode='r') # Load testing f
    testacc = []
    fozle = testingFile[j]
    file_edit = fozle.rfind('_')
    datastr = fozle[file_edit+1: len(fozle)-4]
    print(fozle)
    waveforms = np.transpose(f_waves['waveData'][(0)]) #Load all waveforms
    #waveforms = waveforms[np.random.choice(len(waveforms), 5000000), :]
    for netname in netnames:
        match = netnames.index(netname)
        model_path = pjoin(model_dir, netname)
        model = keras.models.load_model(model_path)

        if match < numcnn:

            x_test = waveforms[:, 1:(52+1)] # waveform voltage values
            y_test = waveforms[:, 0] # waveform labels
            x_test = np.expand_dims(x_test, axis=2)
            #predicts_CNN = model.predict(x_test, verbose=0)
            #print(netname +area+" has acc of: "+str(predicts) +" with " +str(model)
            #print(predicts_CNN.shape)

            #timetest
            time_diffs = []
            for i in np.arange(1,100):
                x_test_400 = x_test[np.random.choice(len(waveforms), 400), :]
                t1 = datetime.now().now()
                predicts_scrub = model.predict(x_test_400, verbose=0)
                t2 = datetime.now().now()
                time_diff = (t2-t1)/400
                time_diffs.append(time_diff.total_seconds())

            print(f'Time Diff CNN: {np.mean(time_diffs)} var: {np.var(time_diffs)}')
        else:
            #print('we made it')
            x_test = waveforms[:, 1:(ntimepts+1)] # waveform voltage values
            #x_test = np.expand_dims(x_test, axis=2)
            y_test = waveforms[:, 0] # waveform labels

            #predicts_NAS = model.predict(x_test, verbose=0)
            #print(netname +area+" has acc of: "+str(predicts) +" with " +str(model)
            #print(predicts_NAS.shape)

            #timetest
            time_diffs = []
            for i in np.arange(1,100):
                x_test_400 = x_test[np.random.choice(len(waveforms), 400), :]
                t1 = datetime.now().now()

```

```

predicts_scrub = model.predict(x_test_400,verbose =0)
t2 = datetime.now().now()
time_diff = (t2-t1)/400
time_diffs.append(time_diff.total_seconds())

```

```

print(f'Time Diff NAS: {np.mean(time_diffs)} var: {np.var(time_diffs)}')

```

```

# NAS 2.144 times faster than CNN at 113.52% of the variance
# CNN classifies 6907 waveforms/sec
# NAS classifies 14805 waveforms/sec

```

full_test.mat

Time Diff CNN: 0.0002552121212121212 var: 1.2651553792470156e-06

Time Diff NAS: 6.484848484848485e-05 var: 2.97684113865932e-10

In [20]:

```

#figure 3 - gamma sweeps and accuracy across each area
# data was placed into an excel spreadsheet
data_dir = 'Z:/SmithLab/prjNAS/NASNet/tests_byarea/' #all areas in separate files
testingFiles = []
for file in os.listdir(data_dir): # pull all training files from directory
    if file.endswith(".mat"):
        testingFiles.append(file)
nsets = len(testingFiles) # number of training files

for j in range(nsets): # loop over the different testing files
    f_waves = h5py.File(data_dir + '/' + testingFiles[j],mode='r') # Load testing
    testacc = []
    fozle = testingFiles[j]
    file_edit = fozle.rfind('_')
    datastr = fozle[file_edit+1: len(fozle)-4]
    print(fozle)
    waveforms = np.transpose(f_waves['waveData'][(0)]) #load all waveforms
    #waveforms = waveforms[np.random.choice(len(waveforms),5000000), :]
    for netname in netnames:
        match = netnames.index(netname)
        model_path = pjoin(model_dir, netname)
        model = keras.models.load_model(model_path)

        if match < numcnn:

            x_test = waveforms[:, 1:(52+1)] # waveform voltage values
            y_test = waveforms[:, 0] # waveform labels
            x_test = np.expand_dims(x_test, axis=2)
            predicts_CNN = model.predict(x_test,verbose =0)
            print(netname + " has acc of: "+str(predicts_CNN) + " with " +str(model.c
            #print(predicts_CNN.shape)

        else:
            #print('we made it')
            x_test = waveforms[:, 1:(ntimepts+1)] # waveform voltage values
            #x_test = np.expand_dims(x_test, axis=2)
            y_test = waveforms[:, 0] # waveform labels

            predicts_NAS = model.predict(x_test,verbose =0)
            print(netname + " has acc of: "+str(predicts_NAS) + " with " +str(model.c

```

```

# print(predicts_NAS.shape)

best_g_NAS = 0
best_g_CNN = 0
g_of_CNN = 0.2
g_of_NAS = 0.2
ranger = np.arange(0, len(y_test), 1)
spikes_are = ranger[y_test == 1]
noise_are = ranger[y_test == 0]

for gamma in np.arange(0.01, 0.99, 0.01):

    predlabel_CNN = predicts_CNN[:, 0] > gamma
    predlabel_NAS = predicts_NAS[:, 0] > (gamma)
    CNN_acc = sum(y_test == predlabel_CNN) / len(y_test)
    NAS_acc = sum(y_test == predlabel_NAS) / len(y_test)
    ranger

    # CNN_spike_acc = sum(y_test[spikes_are] == predlabel_CNN[spikes_are]) / len(s
    # NAS_spike_acc = sum(y_test[spikes_are] == predlabel_NAS[spikes_are]) / len(s
    CNN_noise_acc = sum(y_test[noise_are] == predlabel_CNN[noise_are]) / len(nois
    NAS_noise_acc = sum(y_test[noise_are] == predlabel_NAS[noise_are]) / len(nois
    if best_g_CNN < (CNN_acc):
        best_g_CNN = (CNN_acc)
        g_of_CNN = gamma
    if best_g_NAS < (NAS_acc):
        best_g_NAS = (NAS_acc)
        g_of_NAS = gamma
    # print("Gamma: %f \t CNN %f \t NAS %f spikes: \t CNN %f \t NAS %f" % (gamma, CNN_acc
    # print("%f \t %f \t %f \t %f \t %f" % (gamma, CNN_acc * 100, NAS_acc * 100, CNN_spike_ac
    print("%f \t %f \t %f" % (gamma, CNN_acc * 100, NAS_acc * 100))

print("Best gamma - \t CNN: %f \t NAS: %f" % (g_of_CNN, g_of_NAS))
# gamma = g_of

predlabel_CNN = predicts_CNN[:, 0] > g_of_CNN
predlabel_NAS = predicts_NAS[:, 0] > (g_of_NAS)
CNN_acc = sum(y_test == predlabel_CNN) / len(y_test)
NAS_acc = sum(y_test == predlabel_NAS) / len(y_test)
print("Best ACC: \t CNN %f \t NAS %f" % (CNN_acc * 100, NAS_acc * 100))

best_g_NAS = 0
best_g_CNN = 0
g_of_CNN = 0.48
g_of_NAS = 0.48
ranger = np.arange(0, len(y_test), 1)
spikes_are = ranger[y_test == 1]
print("Best gamma - \t CNN: %f \t NAS: %f" % (g_of_CNN, g_of_NAS))

predlabel_CNN = predicts_CNN[:, 0] > g_of_CNN
predlabel_NAS = predicts_NAS[:, 0] > (g_of_NAS)
CNN_acc = sum(y_test == predlabel_CNN) / len(y_test)
NAS_acc = sum(y_test == predlabel_NAS) / len(y_test)
print("Best ACC: \t CNN %f \t NAS %f" % (CNN_acc * 100, NAS_acc * 100))

```

```

test_FEF.mat
SmithCNNnet has acc of: [[0.70802605]
[0.93216443]
[0.74365234]

```

```

...
[0.2768412 ]
[0.90511584]
[0.5531881 ]] with 222576 params.
NAS_full has acc of: [[0.36806393]
[0.8682376 ]
[0.5341552 ]
...
[0.27719992]
[0.7985201 ]
[0.3755422 ]] with 2701 params.
0.010000      71.571404      70.467355
0.020000      71.709988      70.533009
0.030000      71.914201      70.581014
0.040000      72.167157      70.627668
0.050000      72.438155      70.669233
0.060000      72.723729      70.712789
0.070000      73.020757      70.753961
0.080000      73.316212      70.795723
0.090000      73.602130      70.838813
0.100000      73.885590      70.886130
0.110000      74.167256      70.934258
0.120000      74.446931      70.984623
0.130000      74.716405      71.037422
0.140000      74.988091      71.098700
0.150000      75.242669      71.170278
0.160000      75.493855      71.251098
0.170000      75.744034      71.338997
0.180000      75.986002      71.438793
0.190000      76.222465      71.560220
0.200000      76.447915      71.704752
0.210000      76.670146      71.862287
0.220000      76.893089      72.033588
0.230000      77.110551      72.223298
0.240000      77.322826      72.421932
0.250000      77.520845      72.624572
0.260000      77.714833      72.839354
0.270000      77.911623      73.056423
0.280000      78.092706      73.276982
0.290000      78.267423      73.509930
0.300000      78.438281      73.731398
0.310000      78.595201      73.939077
0.320000      78.741995      74.135720
0.330000      78.884192      74.323489
0.340000      79.014959      74.504695
0.350000      79.138942      74.664295
0.360000      79.252257      74.805631
0.370000      79.359427      74.919045
0.380000      79.450670      75.015498
0.390000      79.532473      75.093663
0.400000      79.595472      75.154696
0.410000      79.653162      75.186773
0.420000      79.698881      75.180702
0.430000      79.722601      75.179866
0.440000      79.733318      75.144594
0.450000      79.724543      75.094327
0.460000      79.700381      75.023732
0.470000      79.658496      74.937406
0.480000      79.615161      74.825271
0.490000      79.546754      74.697281
0.500000      79.463156      74.550906
0.510000      79.370513      74.389388
0.520000      79.252552      74.221652
0.530000      79.112494      74.036464
0.540000      78.961522      73.844197

```

| | | |
|---------------------------------------|---------------|---------------|
| 0.550000 | 78.800865 | 73.636617 |
| 0.560000 | 78.602944 | 73.423727 |
| 0.570000 | 78.400451 | 73.198841 |
| 0.580000 | 78.177262 | 72.967909 |
| 0.590000 | 77.944905 | 72.726236 |
| 0.600000 | 77.694481 | 72.468290 |
| 0.610000 | 77.431840 | 72.202430 |
| 0.620000 | 77.148478 | 71.931235 |
| 0.630000 | 76.844346 | 71.645489 |
| 0.640000 | 76.520648 | 71.349813 |
| 0.650000 | 76.185914 | 71.043640 |
| 0.660000 | 75.826255 | 70.727734 |
| 0.670000 | 75.431790 | 70.410771 |
| 0.680000 | 75.024101 | 70.084320 |
| 0.690000 | 74.580746 | 69.742359 |
| 0.700000 | 74.110599 | 69.381914 |
| 0.710000 | 73.627694 | 69.008539 |
| 0.720000 | 73.105608 | 68.626340 |
| 0.730000 | 72.555747 | 68.224599 |
| 0.740000 | 71.972432 | 67.805898 |
| 0.750000 | 71.360235 | 67.374170 |
| 0.760000 | 70.705071 | 66.913338 |
| 0.770000 | 70.030293 | 66.427140 |
| 0.780000 | 69.309328 | 65.895468 |
| 0.790000 | 68.547928 | 65.331473 |
| 0.800000 | 67.760228 | 64.724143 |
| 0.810000 | 66.935485 | 64.058729 |
| 0.820000 | 66.087366 | 63.329677 |
| 0.830000 | 65.204933 | 62.518601 |
| 0.840000 | 64.296003 | 61.626410 |
| 0.850000 | 63.360698 | 60.639019 |
| 0.860000 | 62.402558 | 59.527424 |
| 0.870000 | 61.415880 | 58.295484 |
| 0.880000 | 60.389062 | 56.903329 |
| 0.890000 | 59.311904 | 55.341546 |
| 0.900000 | 58.187797 | 53.607135 |
| 0.910000 | 56.986066 | 51.695967 |
| 0.920000 | 55.688521 | 49.580218 |
| 0.930000 | 54.267214 | 47.297076 |
| 0.940000 | 52.676254 | 44.855293 |
| 0.950000 | 50.849077 | 42.298770 |
| 0.960000 | 48.705454 | 39.642180 |
| 0.970000 | 46.046529 | 36.948203 |
| 0.980000 | 42.597002 | 34.256143 |
| Best gamma - | CNN: 0.440000 | NAS: 0.410000 |
| Best ACC: | CNN 79.733318 | NAS 75.186773 |
| Best gamma - | CNN: 0.480000 | NAS: 0.480000 |
| Best ACC: | CNN 79.615161 | NAS 74.825271 |
| test_M1.mat | | |
| SmithCNNnet has acc of: [[0.9706267] | | |
| [0.9853221] | | |
| [0.743441] | | |
| ... | | |
| [0.24973476] | | |
| [0.01425135] | | |
| [0.00252596]] with 222576 params. | | |
| NAS_full has acc of: [[0.9627198] | | |
| [0.9995914] | | |
| [0.88791865] | | |
| ... | | |
| [0.14881828] | | |
| [0.02733597] | | |
| [0.00186482]] with 2701 params. | | |
| 0.010000 | 74.209000 | 74.654075 |
| 0.020000 | 76.235840 | 76.560580 |

| | | |
|----------|-----------|-----------|
| 0.030000 | 77.485720 | 77.866144 |
| 0.040000 | 78.400613 | 78.861329 |
| 0.050000 | 79.154720 | 79.634245 |
| 0.060000 | 79.799345 | 80.288173 |
| 0.070000 | 80.365836 | 80.863428 |
| 0.080000 | 80.865990 | 81.359470 |
| 0.090000 | 81.305807 | 81.796388 |
| 0.100000 | 81.720007 | 82.188273 |
| 0.110000 | 82.092139 | 82.543282 |
| 0.120000 | 82.432923 | 82.864381 |
| 0.130000 | 82.756449 | 83.153930 |
| 0.140000 | 83.060425 | 83.431748 |
| 0.150000 | 83.336895 | 83.688264 |
| 0.160000 | 83.610062 | 83.920981 |
| 0.170000 | 83.877498 | 84.146755 |
| 0.180000 | 84.127474 | 84.358439 |
| 0.190000 | 84.355405 | 84.553471 |
| 0.200000 | 84.579561 | 84.733538 |
| 0.210000 | 84.793537 | 84.902211 |
| 0.220000 | 84.995109 | 85.073176 |
| 0.230000 | 85.189804 | 85.230051 |
| 0.240000 | 85.377219 | 85.376208 |
| 0.250000 | 85.546566 | 85.509420 |
| 0.260000 | 85.709576 | 85.634341 |
| 0.270000 | 85.859575 | 85.751037 |
| 0.280000 | 85.999462 | 85.847912 |
| 0.290000 | 86.131124 | 85.941957 |
| 0.300000 | 86.248562 | 86.027776 |
| 0.310000 | 86.359932 | 86.118989 |
| 0.320000 | 86.462605 | 86.193079 |
| 0.330000 | 86.559548 | 86.261775 |
| 0.340000 | 86.639907 | 86.328516 |
| 0.350000 | 86.717637 | 86.387976 |
| 0.360000 | 86.788154 | 86.435706 |
| 0.370000 | 86.849367 | 86.489369 |
| 0.380000 | 86.909973 | 86.530357 |
| 0.390000 | 86.964579 | 86.572155 |
| 0.400000 | 87.002332 | 86.603368 |
| 0.410000 | 87.036377 | 86.632626 |
| 0.420000 | 87.067590 | 86.647660 |
| 0.430000 | 87.095702 | 86.656559 |
| 0.440000 | 87.121050 | 86.658177 |
| 0.450000 | 87.133657 | 86.648671 |
| 0.460000 | 87.140196 | 86.638222 |
| 0.470000 | 87.138039 | 86.623256 |
| 0.480000 | 87.129612 | 86.606402 |
| 0.490000 | 87.106286 | 86.582267 |
| 0.500000 | 87.079253 | 86.544245 |
| 0.510000 | 87.047163 | 86.509459 |
| 0.520000 | 86.989118 | 86.454313 |
| 0.530000 | 86.934849 | 86.396268 |
| 0.540000 | 86.856378 | 86.323460 |
| 0.550000 | 86.774199 | 86.250045 |
| 0.560000 | 86.666941 | 86.170967 |
| 0.570000 | 86.560559 | 86.082855 |
| 0.580000 | 86.453369 | 85.992249 |
| 0.590000 | 86.323864 | 85.885800 |
| 0.600000 | 86.196787 | 85.768699 |
| 0.610000 | 86.056361 | 85.643981 |
| 0.620000 | 85.896721 | 85.504903 |
| 0.630000 | 85.735666 | 85.358612 |
| 0.640000 | 85.553442 | 85.205917 |
| 0.650000 | 85.363062 | 85.045603 |
| 0.660000 | 85.169715 | 84.873020 |
| 0.670000 | 84.972795 | 84.701246 |

| | | |
|--|---------------|---------------|
| 0.680000 | 84.741425 | 84.510123 |
| 0.690000 | 84.502843 | 84.303496 |
| 0.700000 | 84.255968 | 84.088710 |
| 0.710000 | 84.000127 | 83.864218 |
| 0.720000 | 83.721836 | 83.632241 |
| 0.730000 | 83.418265 | 83.386243 |
| 0.740000 | 83.109031 | 83.117661 |
| 0.750000 | 82.784562 | 82.842808 |
| 0.760000 | 82.447081 | 82.530204 |
| 0.770000 | 82.091263 | 82.206408 |
| 0.780000 | 81.726276 | 81.867444 |
| 0.790000 | 81.331830 | 81.506570 |
| 0.800000 | 80.911023 | 81.131269 |
| 0.810000 | 80.476801 | 80.749429 |
| 0.820000 | 80.017434 | 80.363813 |
| 0.830000 | 79.509662 | 79.944153 |
| 0.840000 | 78.959688 | 79.508718 |
| 0.850000 | 78.402231 | 79.051305 |
| 0.860000 | 77.800280 | 78.595847 |
| 0.870000 | 77.154374 | 78.119289 |
| 0.880000 | 76.488311 | 77.616101 |
| 0.890000 | 75.755034 | 77.071520 |
| 0.900000 | 74.995803 | 76.506378 |
| 0.910000 | 74.208460 | 75.882854 |
| 0.920000 | 73.367454 | 75.202431 |
| 0.930000 | 72.478112 | 74.412594 |
| 0.940000 | 71.482321 | 73.480982 |
| 0.950000 | 70.316104 | 72.362292 |
| 0.960000 | 68.883461 | 70.932549 |
| 0.970000 | 66.990372 | 68.975281 |
| 0.980000 | 64.030976 | 66.051412 |
| Best gamma - | CNN: 0.460000 | NAS: 0.440000 |
| Best ACC: | CNN 87.140196 | NAS 86.658177 |
| Best gamma - | CNN: 0.480000 | NAS: 0.480000 |
| Best ACC: | CNN 87.129612 | NAS 86.606402 |
| test_PFC.mat | | |
| SmithCNNnet has acc of: [[9.9264956e-01] | | |
| [9.9045038e-01] | | |
| [9.9495506e-01] | | |
| ... | | |
| [7.2709165e-16] | | |
| [3.0571908e-02] | | |
| [2.1777382e-06]] with 222576 params. | | |
| NAS_full has acc of: [[9.6726519e-01] | | |
| [8.8205469e-01] | | |
| [9.8557377e-01] | | |
| ... | | |
| [5.8659911e-04] | | |
| [5.3554392e-01] | | |
| [1.3933411e-05]] with 2701 params. | | |
| 0.010000 | 79.840256 | 74.170469 |
| 0.020000 | 81.265502 | 75.748212 |
| 0.030000 | 82.164532 | 76.839785 |
| 0.040000 | 82.827356 | 77.687143 |
| 0.050000 | 83.352926 | 78.381875 |
| 0.060000 | 83.798081 | 78.984566 |
| 0.070000 | 84.165276 | 79.512138 |
| 0.080000 | 84.480799 | 80.013228 |
| 0.090000 | 84.768160 | 80.453927 |
| 0.100000 | 85.028394 | 80.862976 |
| 0.110000 | 85.263954 | 81.256524 |
| 0.120000 | 85.476520 | 81.612157 |
| 0.130000 | 85.673585 | 81.952612 |
| 0.140000 | 85.855987 | 82.270266 |
| 0.150000 | 86.025213 | 82.563117 |

| | | |
|----------|-----------|-----------|
| 0.160000 | 86.180165 | 82.843826 |
| 0.170000 | 86.324782 | 83.108258 |
| 0.180000 | 86.461197 | 83.360095 |
| 0.190000 | 86.590248 | 83.598627 |
| 0.200000 | 86.710902 | 83.833541 |
| 0.210000 | 86.826712 | 84.054245 |
| 0.220000 | 86.936839 | 84.258027 |
| 0.230000 | 87.045285 | 84.469237 |
| 0.240000 | 87.145723 | 84.668756 |
| 0.250000 | 87.239250 | 84.856907 |
| 0.260000 | 87.328449 | 85.034207 |
| 0.270000 | 87.414806 | 85.212088 |
| 0.280000 | 87.499160 | 85.385642 |
| 0.290000 | 87.590491 | 85.546794 |
| 0.300000 | 87.668386 | 85.700519 |
| 0.310000 | 87.743569 | 85.850626 |
| 0.320000 | 87.814360 | 85.991497 |
| 0.330000 | 87.884570 | 86.131852 |
| 0.340000 | 87.953875 | 86.264391 |
| 0.350000 | 88.015106 | 86.388985 |
| 0.360000 | 88.072398 | 86.513192 |
| 0.370000 | 88.124780 | 86.622995 |
| 0.380000 | 88.176646 | 86.734542 |
| 0.390000 | 88.230708 | 86.835690 |
| 0.400000 | 88.285610 | 86.938583 |
| 0.410000 | 88.333859 | 87.027459 |
| 0.420000 | 88.376165 | 87.117562 |
| 0.430000 | 88.412723 | 87.196749 |
| 0.440000 | 88.446633 | 87.278327 |
| 0.450000 | 88.480801 | 87.352541 |
| 0.460000 | 88.511675 | 87.414289 |
| 0.470000 | 88.546619 | 87.480106 |
| 0.480000 | 88.565350 | 87.539077 |
| 0.490000 | 88.586148 | 87.594108 |
| 0.500000 | 88.604879 | 87.637448 |
| 0.510000 | 88.611855 | 87.682080 |
| 0.520000 | 88.620187 | 87.713406 |
| 0.530000 | 88.624062 | 87.738725 |
| 0.540000 | 88.620187 | 87.764238 |
| 0.550000 | 88.606687 | 87.782194 |
| 0.560000 | 88.599066 | 87.791366 |
| 0.570000 | 88.582079 | 87.790785 |
| 0.580000 | 88.561539 | 87.775089 |
| 0.590000 | 88.529115 | 87.748284 |
| 0.600000 | 88.486937 | 87.724515 |
| 0.610000 | 88.450056 | 87.679819 |
| 0.620000 | 88.397932 | 87.627049 |
| 0.630000 | 88.342385 | 87.559940 |
| 0.640000 | 88.285933 | 87.473776 |
| 0.650000 | 88.209329 | 87.373597 |
| 0.660000 | 88.124780 | 87.259079 |
| 0.670000 | 88.038488 | 87.134291 |
| 0.680000 | 87.942959 | 86.985346 |
| 0.690000 | 87.828505 | 86.814311 |
| 0.700000 | 87.692737 | 86.621897 |
| 0.710000 | 87.556323 | 86.396930 |
| 0.720000 | 87.399627 | 86.137277 |
| 0.730000 | 87.219743 | 85.853856 |
| 0.740000 | 87.037470 | 85.537170 |
| 0.750000 | 86.845637 | 85.190580 |
| 0.760000 | 86.624933 | 84.782241 |
| 0.770000 | 86.377746 | 84.342124 |
| 0.780000 | 86.107049 | 83.850786 |
| 0.790000 | 85.815748 | 83.289498 |
| 0.800000 | 85.496866 | 82.650702 |

| | | |
|---------------------------------------|---------------|---------------|
| 0.810000 | 85.134903 | 81.935043 |
| 0.820000 | 84.738642 | 81.110937 |
| 0.830000 | 84.303822 | 80.158233 |
| 0.840000 | 83.828438 | 79.068597 |
| 0.850000 | 83.280520 | 77.828338 |
| 0.860000 | 82.680672 | 76.415041 |
| 0.870000 | 82.011647 | 74.796993 |
| 0.880000 | 81.249677 | 72.952105 |
| 0.890000 | 80.387786 | 70.801899 |
| 0.900000 | 79.425329 | 68.313694 |
| 0.910000 | 78.312570 | 65.519331 |
| 0.920000 | 76.976071 | 62.397883 |
| 0.930000 | 75.406982 | 58.869582 |
| 0.940000 | 73.528055 | 54.955678 |
| 0.950000 | 71.233491 | 50.573883 |
| 0.960000 | 68.372341 | 45.713928 |
| 0.970000 | 64.553191 | 40.324268 |
| 0.980000 | 59.206872 | 34.726047 |
| Best gamma - | CNN: 0.530000 | NAS: 0.560000 |
| Best ACC: | CNN 88.624062 | NAS 87.791366 |
| Best gamma - | CNN: 0.480000 | NAS: 0.480000 |
| Best ACC: | CNN 88.565350 | NAS 87.539077 |
| test_V4.mat | | |
| SmithCNNnet has acc of: [[0.95236915] | | |
| [0.9909262] | | |
| [0.99970496] | | |
| ... | | |
| [0.10696539] | | |
| [0.00127164] | | |
| [0.7000898]] with 222576 params. | | |
| NAS_full has acc of: [[9.2407745e-01] | | |
| [7.5693905e-01] | | |
| [9.0013546e-01] | | |
| ... | | |
| [2.7215630e-02] | | |
| [1.3545156e-04] | | |
| [7.6440048e-01]] with 2701 params. | | |
| 0.010000 | 80.655888 | 77.552278 |
| 0.020000 | 82.116413 | 78.939478 |
| 0.030000 | 83.030064 | 79.851220 |
| 0.040000 | 83.694208 | 80.546302 |
| 0.050000 | 84.228075 | 81.113667 |
| 0.060000 | 84.678372 | 81.610771 |
| 0.070000 | 85.062123 | 82.048211 |
| 0.080000 | 85.390328 | 82.434172 |
| 0.090000 | 85.686441 | 82.778649 |
| 0.100000 | 85.951667 | 83.108311 |
| 0.110000 | 86.204889 | 83.407989 |
| 0.120000 | 86.427376 | 83.688382 |
| 0.130000 | 86.636051 | 83.942459 |
| 0.140000 | 86.829910 | 84.185486 |
| 0.150000 | 87.016488 | 84.408375 |
| 0.160000 | 87.182222 | 84.628249 |
| 0.170000 | 87.335201 | 84.828487 |
| 0.180000 | 87.484563 | 85.013909 |
| 0.190000 | 87.623880 | 85.196067 |
| 0.200000 | 87.759482 | 85.371495 |
| 0.210000 | 87.882477 | 85.539288 |
| 0.220000 | 88.002409 | 85.691664 |
| 0.230000 | 88.122290 | 85.839670 |
| 0.240000 | 88.231324 | 85.982202 |
| 0.250000 | 88.337545 | 86.116246 |
| 0.260000 | 88.437236 | 86.242908 |
| 0.270000 | 88.536426 | 86.372030 |
| 0.280000 | 88.632854 | 86.488396 |

| | | |
|----------|-----------|-----------|
| 0.290000 | 88.723255 | 86.601096 |
| 0.300000 | 88.806674 | 86.710230 |
| 0.310000 | 88.885223 | 86.818259 |
| 0.320000 | 88.960205 | 86.921265 |
| 0.330000 | 89.032475 | 87.023368 |
| 0.340000 | 89.101933 | 87.116431 |
| 0.350000 | 89.166419 | 87.208388 |
| 0.360000 | 89.226787 | 87.291105 |
| 0.370000 | 89.278968 | 87.371361 |
| 0.380000 | 89.329894 | 87.444535 |
| 0.390000 | 89.370374 | 87.523686 |
| 0.400000 | 89.413816 | 87.590533 |
| 0.410000 | 89.448269 | 87.651955 |
| 0.420000 | 89.485584 | 87.707853 |
| 0.430000 | 89.513860 | 87.763550 |
| 0.440000 | 89.533848 | 87.810859 |
| 0.450000 | 89.544546 | 87.862036 |
| 0.460000 | 89.548212 | 87.897142 |
| 0.470000 | 89.550924 | 87.928933 |
| 0.480000 | 89.554640 | 87.955300 |
| 0.490000 | 89.545550 | 87.973832 |
| 0.500000 | 89.534250 | 87.986538 |
| 0.510000 | 89.516321 | 87.998190 |
| 0.520000 | 89.478302 | 87.994574 |
| 0.530000 | 89.439229 | 87.988748 |
| 0.540000 | 89.397393 | 87.970768 |
| 0.550000 | 89.349330 | 87.934960 |
| 0.560000 | 89.288310 | 87.896389 |
| 0.570000 | 89.216893 | 87.840792 |
| 0.580000 | 89.139199 | 87.770330 |
| 0.590000 | 89.051108 | 87.678121 |
| 0.600000 | 88.953726 | 87.573808 |
| 0.610000 | 88.847154 | 87.448854 |
| 0.620000 | 88.737568 | 87.309687 |
| 0.630000 | 88.619043 | 87.146715 |
| 0.640000 | 88.483291 | 86.958983 |
| 0.650000 | 88.334481 | 86.737400 |
| 0.660000 | 88.180448 | 86.483424 |
| 0.670000 | 88.010093 | 86.210313 |
| 0.680000 | 87.813722 | 85.910785 |
| 0.690000 | 87.623629 | 85.558222 |
| 0.700000 | 87.401394 | 85.168645 |
| 0.710000 | 87.177853 | 84.732562 |
| 0.720000 | 86.922421 | 84.227472 |
| 0.730000 | 86.645342 | 83.695263 |
| 0.740000 | 86.352243 | 83.087870 |
| 0.750000 | 86.038753 | 82.420562 |
| 0.760000 | 85.697590 | 81.661195 |
| 0.770000 | 85.327299 | 80.830361 |
| 0.780000 | 84.937721 | 79.913948 |
| 0.790000 | 84.527201 | 78.883831 |
| 0.800000 | 84.073038 | 77.742873 |
| 0.810000 | 83.568501 | 76.482535 |
| 0.820000 | 83.039054 | 75.093075 |
| 0.830000 | 82.458480 | 73.570324 |
| 0.840000 | 81.817538 | 71.910063 |
| 0.850000 | 81.121753 | 70.092154 |
| 0.860000 | 80.371075 | 68.090479 |
| 0.870000 | 79.548629 | 65.901776 |
| 0.880000 | 78.628198 | 63.539653 |
| 0.890000 | 77.622037 | 60.954442 |
| 0.900000 | 76.493936 | 58.164974 |
| 0.910000 | 75.222298 | 55.174516 |
| 0.920000 | 73.786432 | 51.962827 |
| 0.930000 | 72.143197 | 48.567775 |

| | | |
|--------------|---------------|---------------|
| 0.940000 | 70.242822 | 45.025018 |
| 0.950000 | 68.008265 | 41.391208 |
| 0.960000 | 65.296944 | 37.821632 |
| 0.970000 | 61.883360 | 34.596484 |
| 0.980000 | 57.289493 | 32.057376 |
| Best gamma - | CNN: 0.480000 | NAS: 0.510000 |
| Best ACC: | CNN 89.554640 | NAS 87.998190 |
| Best gamma - | CNN: 0.480000 | NAS: 0.480000 |
| Best ACC: | CNN 89.554640 | NAS 87.955300 |

In [17]:

```

#figure 3 on excel CNNnas.xlsx
#range and variance
# 10 fold cross validation on test data
data_dir_base = 'Z:/SmithLab/prjNAS/NASNet/tests_byarea_kfold/' #all areas in separate

area_names = ["PFC", "M1", "V4", "FEF"]
#area_names = ["V4"]
area_var = np.zeros([4,4])
for area in area_names:
    data_dir = data_dir_base + area + '/'
    testingFiles = []
    for file in os.listdir(data_dir): # pull all training files from directory
        if file.endswith(".mat"):
            testingFiles.append(file)
    nsets = len(testingFiles) # number of training files
    print(area)
    kfold = np.zeros([nsets, 4]) #contains accuracy at 0.2 for NAS/CNN, and 0.48 for NA
    for j in range(nsets): # Loop over the different testing files
        f_waves = h5py.File(data_dir + '/' + testingFiles[j], mode='r') # Load test
        testacc = []
        fozle = testingFiles[j]
        file_edit = fozle.rfind('_')
        datastr = fozle[file_edit+1: len(fozle)-4]
        print(fozle)
        waveforms = np.transpose(f_waves['waveData'][(0)]) #load all waveforms
        #waveforms = waveforms[np.random.choice(len(waveforms), 5000000), :]
        for netname in netnames:
            match = netnames.index(netname)
            model_path = pjoin(model_dir, netname)
            model = keras.models.load_model(model_path)

            if match < numcnn:

                x_test = waveforms[:, 1:(52+1)] # waveform voltage values
                y_test = waveforms[:, 0] # waveform Labels
                x_test = np.expand_dims(x_test, axis=2)
                predicts_CNN = model.predict(x_test, verbose = 0)

            else:
                #print('we made it')
                x_test = waveforms[:, 1:(ntimepts+1)] # waveform voltage values
                #x_test = np.expand_dims(x_test, axis=2)
                y_test = waveforms[:, 0] # waveform Labels
                predicts_NAS = model.predict(x_test, verbose = 0)

        predlabel_CNNpt2 = predicts_CNN[:,0] > 0.2
        predlabel_CNNpt48 = predicts_CNN[:,0] > 0.48
        predlabel_NASpt2 = predicts_NAS[:,0] > 0.2
        predlabel_NASpt48 = predicts_NAS[:,0] > 0.48

```

```

kfold[j, 0] = sum(y_test == predlabel_NASpt2)/len(y_test)
kfold[j, 1] = sum(y_test == predlabel_CNNpt2)/len(y_test)
kfold[j, 2] = sum(y_test == predlabel_NASpt48)/len(y_test)
kfold[j, 3] = sum(y_test == predlabel_CNNpt48)/len(y_test)

area_var[area_names.index(area)] = [np.var(kfold[:, 0]), np.var(kfold[:, 1]), np.v

print("gamma=0.2 \t\tNAS\t\tCNN\t\tgamma=0.48 NAS\t\tCNN")
print("average acc: ", np.mean(kfold[:, 0]), np.mean(kfold[:, 1]), np.mean(kfold[:, 2]), np.mean(kfold[:, 3]), n
print("var:\t\t\t", np.var(kfold[:, 0]), np.var(kfold[:, 1]), np.var(kfold[:, 2]), np.var(kfold[:, 3]), n

```

PFC

test_1.mat
test_10.mat
test_2.mat
test_3.mat
test_4.mat
test_5.mat
test_6.mat
test_7.mat
test_8.mat
test_9.mat

| gamma=0.2 | NAS | CNN | gamma=0.48 NAS | CNN |
|--------------|-----------------------|-----------------------|-----------------------|-----------------------|
| average acc: | 0.8383354085811361 | 0.8671090227863454 | 0.8753907695497561 | 0.885653497206439 |
| var: | 8.156480794596731e-07 | 7.056196930351562e-07 | 8.191190105021668e-07 | 7.667469149886502e-07 |

M1

test_1.mat
test_10.mat
test_2.mat
test_3.mat
test_4.mat
test_5.mat
test_6.mat
test_7.mat
test_8.mat
test_9.mat

| gamma=0.2 | NAS | CNN | gamma=0.48 NAS | CNN |
|--------------|-----------------------|-----------------------|-----------------------|-----------------------|
| average acc: | 0.84733537668137 | 0.8457956113339868 | 0.866064018453336 | 0.8712961201528735 |
| var: | 4.201065338185848e-07 | 5.760230707739078e-07 | 9.007338597599479e-07 | 9.725298048907775e-07 |

V4

test_1.mat
test_10.mat
test_2.mat
test_3.mat
test_4.mat
test_5.mat
test_6.mat
test_7.mat
test_8.mat
test_9.mat

| gamma=0.2 | NAS | CNN | gamma=0.48 NAS | CNN |
|--------------|-----------------------|-----------------------|-----------------------|-----------------------|
| average acc: | 0.8537149448458514 | 0.8775948163249849 | 0.8795529979728001 | 0.895546404669909 |
| var: | 9.593496016125697e-07 | 5.086381887970802e-07 | 6.414652197802632e-07 | 6.186920959731345e-07 |

FEF

test_1.mat
test_10.mat
test_2.mat
test_3.mat
test_4.mat
test_5.mat

```

test_6.mat
test_7.mat
test_8.mat
test_9.mat
gamma=0.2          NAS          CNN          gamma=0.48 NAS          CNN
average acc:  0.7170475224482281 0.7644791508629807 0.7482527089678372 0.796151608961875
6
var:          4.209654443091048e-07 3.7383588497614953e-07 5.752884279937727e-07 5.020865
312099486e-07

```

In [18]:

```

print(area_names)
print(area_var)
print("gamma=0.2 \t\tNAS\t\t\tCNN\t\t\tgamma=0.48 NAS\t\t\tCNN")
print("average var: ", np.mean(area_var[:, 0]), np.mean(area_var[:, 1]), np.mean(area_v

['PFC', 'M1', 'V4', 'FEF']
[[8.15648079e-07 7.05619693e-07 8.19119011e-07 7.66746915e-07]
 [4.20106534e-07 5.76023071e-07 9.00733860e-07 9.72529805e-07]
 [9.59349602e-07 5.08638189e-07 6.41465220e-07 6.18692096e-07]
 [4.20965444e-07 3.73835885e-07 5.75288428e-07 5.02086531e-07]]
gamma=0.2          NAS          CNN          gamma=0.48 NAS          CNN
average var:  6.540174147999831e-07 5.410292093955734e-07 7.341516295090377e-07 7.150138
367656277e-07

```

In [21]:

```

# buffer load all predictions here
#figure 3 - gamma sweeps and accuracy across each area
data_dir = 'Z:/SmithLab/prjNAS/NASNet/test_all/' #all areas in one file
testingFiles = []
for file in os.listdir(data_dir): # pull all training files from directory
    if file.endswith(".mat"):
        testingFiles.append(file)
nsets = len(testingFiles) # singular testing file

for j in range(nsets): # loop over the different testing files
    f_waves = h5py.File(data_dir + '/' + testingFiles[j], mode='r') # load testing
    testacc = []
    fozle = testingFiles[j]
    file_edit = fozle.rfind('_')
    datastr = fozle[file_edit+1: len(fozle)-4]
    print(fozle)
    waveforms = np.transpose(f_waves['waveData'][(0)]) #load all waveforms
    #waveforms = waveforms[np.random.choice(len(waveforms), 500000), :]
    for netname in netnames:
        match = netnames.index(netname)
        model_path = pjoin(model_dir, netname)
        model = keras.models.load_model(model_path)

        if match < numcnn:

            x_test = waveforms[:, 1:(52+1)] # waveform voltage values
            y_test = waveforms[:, 0] # waveform labels
            x_test = np.expand_dims(x_test, axis=2)
            predicts_CNAll = model.predict(x_test, verbose = 0)
            #print(netname + " has acc of: " + str(predicts_CNAll) + " with " + str(mod
            #print(predicts_CNN.shape)

        else:

```

```

# print('we made it')
x_test = waveforms[:, 1:(ntimepts+1)] # waveform voltage values
# x_test = np.expand_dims(x_test, axis=2)
y_test = waveforms[:, 0] # waveform labels

predicts_NASall = model.predict(x_test, verbose=0)
# print(netname + " has acc of: " + str(predicts_NASall) + " with " + str(mod
# print(predicts_NAS.shape)

best_g_NAS = 0
best_g_CNN = 0
g_of_CNN = 0.2
g_of_NAS = 0.2
ranger = np.arange(0, len(y_test), 1)
spikes_are = ranger[y_test == 1]
noise_are = ranger[y_test == 0]

for gamma in np.arange(0.01, 0.99, 0.01):

    predlabel_CNN = predicts_CNNall[:, 0] > gamma
    predlabel_NAS = predicts_NASall[:, 0] > (gamma)
    CNN_acc = sum(y_test == predlabel_CNN) / len(y_test)
    NAS_acc = sum(y_test == predlabel_NAS) / len(y_test)
    ranger

    # CNN_spike_acc = sum(y_test[spikes_are] == predlabel_CNN[spikes_are]) / len(s
    # NAS_spike_acc = sum(y_test[spikes_are] == predlabel_NAS[spikes_are]) / len(s
    CNN_noise_acc = sum(y_test[noise_are] == predlabel_CNN[noise_are]) / len(nois
    NAS_noise_acc = sum(y_test[noise_are] == predlabel_NAS[noise_are]) / len(nois
    if best_g_CNN < (CNN_acc):
        best_g_CNN = (CNN_acc)
        g_of_CNN = gamma
    if best_g_NAS < (NAS_acc):
        best_g_NAS = (NAS_acc)
        g_of_NAS = gamma
    # print("Gamma: %f\tCNN %f\tNAS %f spikes:\tCNN %f\tNAS %f" %(gamma, CNN_acc
    # print("%f\t%f\t%f\t%f\t%f" %(gamma, CNN_acc*100, NAS_acc*100, CNN_spike_ac
    print("%f\t%f\t%f" %(gamma, CNN_acc*100, NAS_acc*100))

print("Best gamma -\tCNN: %f \tNAS: %f" %(g_of_CNN, g_of_NAS))
# gamma = g_of

predlabel_CNN = predicts_CNNall[:, 0] > g_of_CNN
predlabel_NAS = predicts_NASall[:, 0] > (g_of_NAS)
CNN_acc = sum(y_test == predlabel_CNN) / len(y_test)
NAS_acc = sum(y_test == predlabel_NAS) / len(y_test)
print("Best ACC:\tCNN %f\tNAS %f" %(CNN_acc*100, NAS_acc*100))

best_g_NAS = 0
best_g_CNN = 0
g_of_CNN = 0.48
g_of_NAS = 0.48
ranger = np.arange(0, len(y_test), 1)
spikes_are = ranger[y_test == 1]
print("Best gamma -\tCNN: %f \tNAS: %f" %(g_of_CNN, g_of_NAS))

predlabel_CNN = predicts_CNNall[:, 0] > g_of_CNN
predlabel_NAS = predicts_NASall[:, 0] > (g_of_NAS)
CNN_acc = sum(y_test == predlabel_CNN) / len(y_test)

```

```

NAS_acc = sum(y_test == predlabel_NAS)/len(y_test)
print("Best ACC:\tCNN %f\tNAS %f" %(CNN_acc*100, NAS_acc*100))

max_idx = []
X_maxes = np.amax(x_test,1)
pos = 0
for mx in X_maxes:
    max_val = np.where(x_test[pos,:] == mx)
    max_val_val = max_val[0]
    max_idx.append(max_val_val[0])
    pos = pos + 1
min_idx = []
X_mins = np.amin(x_test,1)
pos = 0
for mins in X_mins:
    min_val = np.where(x_test[pos,:] == mins)
    min_val_val = min_val[0]
    min_idx.append(min_val_val[0])
    pos = pos + 1
y_test_idx= []
y_test_idx_noise= []
max_idx_spike = []

p_CNN_spike = []
p_NAS_spike = []
p_CNN_noise = []
p_NAS_noise = []
max_idx_noise = []
for p in np.arange(0,len(y_test)-1):
    if y_test[p] == 1:
        y_test_idx = p
        max_idx_spike.append(max_idx[y_test_idx])
        p_CNN_spike.append(predicts_CNNall[y_test_idx,0])
        p_NAS_spike.append(predicts_NASall[y_test_idx,0])
    else:
        y_test_idx_noise = p
        max_idx_noise.append(max_idx[y_test_idx_noise])
        p_CNN_noise.append(predicts_CNNall[y_test_idx_noise,0])
        p_NAS_noise.append(predicts_NASall[y_test_idx_noise,0])

```

full_test.mat

| | | |
|----------|-----------|-----------|
| 0.010000 | 75.399687 | 73.332895 |
| 0.020000 | 76.355027 | 74.245875 |
| 0.030000 | 77.003570 | 74.865972 |
| 0.040000 | 77.524393 | 75.345776 |
| 0.050000 | 77.975146 | 75.733071 |
| 0.060000 | 78.382560 | 76.070779 |
| 0.070000 | 78.754500 | 76.368722 |
| 0.080000 | 79.093945 | 76.638219 |
| 0.090000 | 79.407453 | 76.879293 |
| 0.100000 | 79.704296 | 77.106276 |
| 0.110000 | 79.986642 | 77.318409 |
| 0.120000 | 80.252333 | 77.515307 |
| 0.130000 | 80.504979 | 77.699808 |
| 0.140000 | 80.749682 | 77.879887 |
| 0.150000 | 80.978403 | 78.052465 |
| 0.160000 | 81.198071 | 78.222567 |
| 0.170000 | 81.411799 | 78.387631 |
| 0.180000 | 81.616815 | 78.550331 |
| 0.190000 | 81.812316 | 78.717012 |

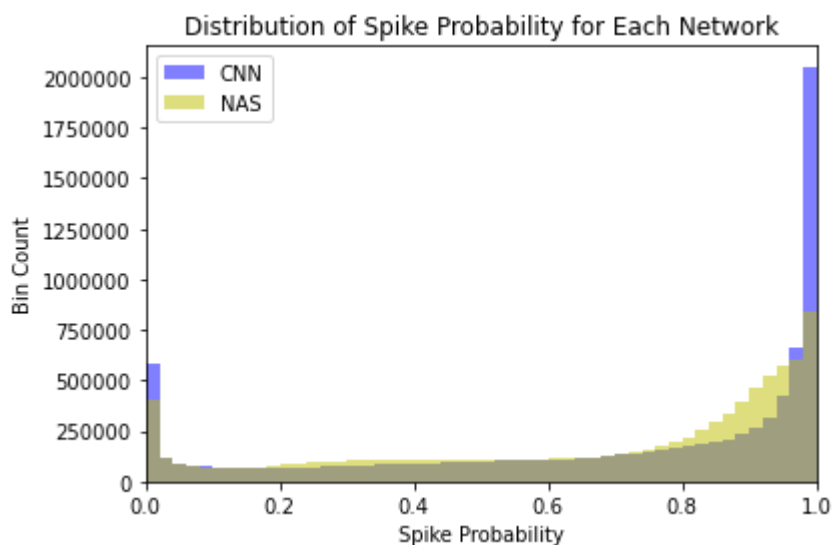
| | | |
|----------|-----------|-----------|
| 0.200000 | 82.000029 | 78.889501 |
| 0.210000 | 82.181054 | 79.061859 |
| 0.220000 | 82.358736 | 79.234491 |
| 0.230000 | 82.532545 | 79.413371 |
| 0.240000 | 82.699105 | 79.591306 |
| 0.250000 | 82.854545 | 79.765115 |
| 0.260000 | 83.004979 | 79.939562 |
| 0.270000 | 83.153951 | 80.114317 |
| 0.280000 | 83.293308 | 80.283870 |
| 0.290000 | 83.428321 | 80.455589 |
| 0.300000 | 83.555480 | 80.618784 |
| 0.310000 | 83.673883 | 80.775829 |
| 0.320000 | 83.784806 | 80.922469 |
| 0.330000 | 83.892044 | 81.063971 |
| 0.340000 | 83.990691 | 81.198918 |
| 0.350000 | 84.083409 | 81.321391 |
| 0.360000 | 84.168592 | 81.431709 |
| 0.370000 | 84.246889 | 81.527485 |
| 0.380000 | 84.317607 | 81.612360 |
| 0.390000 | 84.381198 | 81.688721 |
| 0.400000 | 84.434415 | 81.753290 |
| 0.410000 | 84.481550 | 81.801008 |
| 0.420000 | 84.522480 | 81.828332 |
| 0.430000 | 84.550101 | 81.855094 |
| 0.440000 | 84.569186 | 81.863828 |
| 0.450000 | 84.575478 | 81.863630 |
| 0.460000 | 84.571793 | 81.848539 |
| 0.470000 | 84.559242 | 81.825637 |
| 0.480000 | 84.542478 | 81.788523 |
| 0.490000 | 84.509610 | 81.740740 |
| 0.500000 | 84.468504 | 81.679195 |
| 0.510000 | 84.419070 | 81.611392 |
| 0.520000 | 84.349903 | 81.531874 |
| 0.530000 | 84.270473 | 81.442565 |
| 0.540000 | 84.180285 | 81.345051 |
| 0.550000 | 84.082155 | 81.235394 |
| 0.560000 | 83.961420 | 81.120335 |
| 0.570000 | 83.834910 | 80.993044 |
| 0.580000 | 83.697026 | 80.856810 |
| 0.590000 | 83.547098 | 80.706530 |
| 0.600000 | 83.385784 | 80.545095 |
| 0.610000 | 83.215715 | 80.370791 |
| 0.620000 | 83.029982 | 80.187269 |
| 0.630000 | 82.832182 | 79.988402 |
| 0.640000 | 82.618245 | 79.775378 |
| 0.650000 | 82.391746 | 79.546613 |
| 0.660000 | 82.151113 | 79.301953 |
| 0.670000 | 81.890448 | 79.051013 |
| 0.680000 | 81.610974 | 78.782770 |
| 0.690000 | 81.312514 | 78.489666 |
| 0.700000 | 80.990041 | 78.175234 |
| 0.710000 | 80.660000 | 77.837692 |
| 0.720000 | 80.298324 | 77.473958 |
| 0.730000 | 79.911402 | 77.089204 |
| 0.740000 | 79.504670 | 76.671043 |
| 0.750000 | 79.076434 | 76.227813 |
| 0.760000 | 78.615868 | 75.734721 |
| 0.770000 | 78.132643 | 75.207386 |
| 0.780000 | 77.619025 | 74.629760 |
| 0.790000 | 77.074409 | 73.997277 |
| 0.800000 | 76.499467 | 73.305571 |
| 0.810000 | 75.887389 | 72.547557 |
| 0.820000 | 75.249450 | 71.713722 |
| 0.830000 | 74.570493 | 70.786519 |
| 0.840000 | 73.852662 | 69.767005 |

| | | |
|--------------|---------------|---------------|
| 0.850000 | 73.097442 | 68.641122 |
| 0.860000 | 72.303877 | 67.390258 |
| 0.870000 | 71.462869 | 66.006262 |
| 0.880000 | 70.563320 | 64.469610 |
| 0.890000 | 69.594482 | 62.749436 |
| 0.900000 | 68.556566 | 60.846356 |
| 0.910000 | 67.422290 | 58.758488 |
| 0.920000 | 66.162307 | 56.465627 |
| 0.930000 | 64.754024 | 53.970554 |
| 0.940000 | 63.143364 | 51.283335 |
| 0.950000 | 61.255211 | 48.414619 |
| 0.960000 | 58.981060 | 45.383004 |
| 0.970000 | 56.084216 | 42.233812 |
| 0.980000 | 52.140997 | 39.042501 |
| Best gamma - | CNN: 0.450000 | NAS: 0.440000 |
| Best ACC: | CNN 84.575478 | NAS 81.863828 |
| Best gamma - | CNN: 0.480000 | NAS: 0.480000 |
| Best ACC: | CNN 84.542478 | NAS 81.788523 |

In [7]:

```
# supplemental figure
# Distribution of spike probability for each Network
p_CNN_spike = []
p_NAS_spike = []
p_CNN_noise = []
p_NAS_noise = []

plt.figure()
kwargs = dict(alpha=0.5, bins=50)
plt.hist(predicts_CNNall[:,0], **kwargs, label = "CNN",color='b')
plt.hist(predicts_NASall[:,0], **kwargs, label = "NAS",color='y')
plt.legend()
plt.xlabel("Spike Probability")
plt.ylabel("Bin Count")
plt.title("Distribution of Spike Probability for Each Network")
plt.xlim([0, 1])
plt.ticklabel_format(useOffset=False, style='plain')
plt.show()
```



In [10]:

```
#figure 4 - comparison to hand label distribution
#over a single chanel of v4
data_dir = 'Z:/SmithLab/prjNAS/NASNet/tests_m1/v4_ch/' #all areas in separate files
testingFiles = []
```

```

for file in os.listdir(data_dir): # pull all training files from directory
    if file.endswith(".mat"):
        testingFiles.append(file)
nsets = len(testingFiles) # number of training files

for j in range(nsets): # Loop over the different testing files
    f_waves = h5py.File(data_dir + '/' + testingFiles[j], mode='r') # Load testing
    testacc = []
    fozle = testingFiles[j]
    file_edit = fozle.rfind('_')
    datastr = fozle[file_edit+1: len(fozle)-4]
    print(fozle)
    waveforms = np.transpose(f_waves['waveData'][(0)]) #Load all waveforms
    #waveforms = waveforms[np.random.choice(len(waveforms), 5000000), :]
    for netname in netnames:
        match = netnames.index(netname)
        model_path = pjoin(model_dir, netname)
        model = keras.models.load_model(model_path)

        if match < numcnn:

            x_test = waveforms[:, 1:(52+1)] # waveform voltage values
            y_test = waveforms[:, 0] # waveform labels
            x_test = np.expand_dims(x_test, axis=2)
            predicts_CNN = model.predict(x_test, verbose=0)
            print(netname + " has acc of: " + str(predicts_CNN) + " with " + str(model.c
            #print(predicts_CNN.shape)

        else:
            #print('we made it')
            x_test = waveforms[:, 1:(ntimepts+1)] # waveform voltage values
            #x_test = np.expand_dims(x_test, axis=2)
            y_test = waveforms[:, 0] # waveform labels

            predicts_NAS = model.predict(x_test, verbose=0)
            print(netname + " has acc of: " + str(predicts_NAS) + " with " + str(model.c
            #print(predicts_NAS.shape)

    best_g_NAS = 0
    best_g_CNN = 0
    g_of_CNN = 0.2
    g_of_NAS = 0.2
    ranger = np.arange(0, len(y_test), 1)
    spikes_are = ranger[y_test == 1]
    noise_are = ranger[y_test == 0]

    lims = 200
    #ch = fozle[fozle.find('_SNR')-2:fozle.find('_SNR')]
    ch = fozle[fozle.find('ch')+2:fozle.find('ch')+3]
    lab = 'V4'
    rngvals = np.random.choice(len(predicts_NAS), 3000)
    t = np.arange(0, 52)
    plt.figure(figsize=(8, 6), dpi=100)
    for r in rngvals:
        if y_test[r] == 1:
            plt.plot(t, x_test[r, :], c='darkgreen', alpha=0.8)
        else:

```

```

plt.plot(t,x_test[r,:],c='darkred', alpha=0.6)
plt.title('Hand Labeled Data for ' +lab)
plt.ylabel(u'\u03bcV')
plt.xlim([0, 51])
plt.ylim([-lims ,lims])
custom_lines = [mpl.lines.Line2D([0], [0], color='g', lw=4),
                 mpl.lines.Line2D([0], [0], color='brown', lw=4)]

plt.legend(custom_lines, ['Spike ' + str(round(sum(y_test)*100/len(y_test)))+'%
plt.show()

n_lines = 10
c = np.arange(1, n_lines + 1)

norm = mpl.colors.Normalize(vmin=0, vmax=1)
cmap = mpl.cm.ScalarMappable(norm=norm, cmap='RdYlGn')
cmap.set_array([])

fig, ax = plt.subplots(figsize=(8, 6),dpi=100)
for r in rngvals:
    ax.plot(t, x_test[r,:], c=cmap.to_rgba(predicts_NAS[r,0]))
#fig.colorbar(cmap)

plt.title('NAS predictions for ' +lab +' When Removed from Training')
plt.ylabel(u'\u03bcV')
plt.xlim([0, 51])
plt.ylim([-lims ,lims])
plt.show()

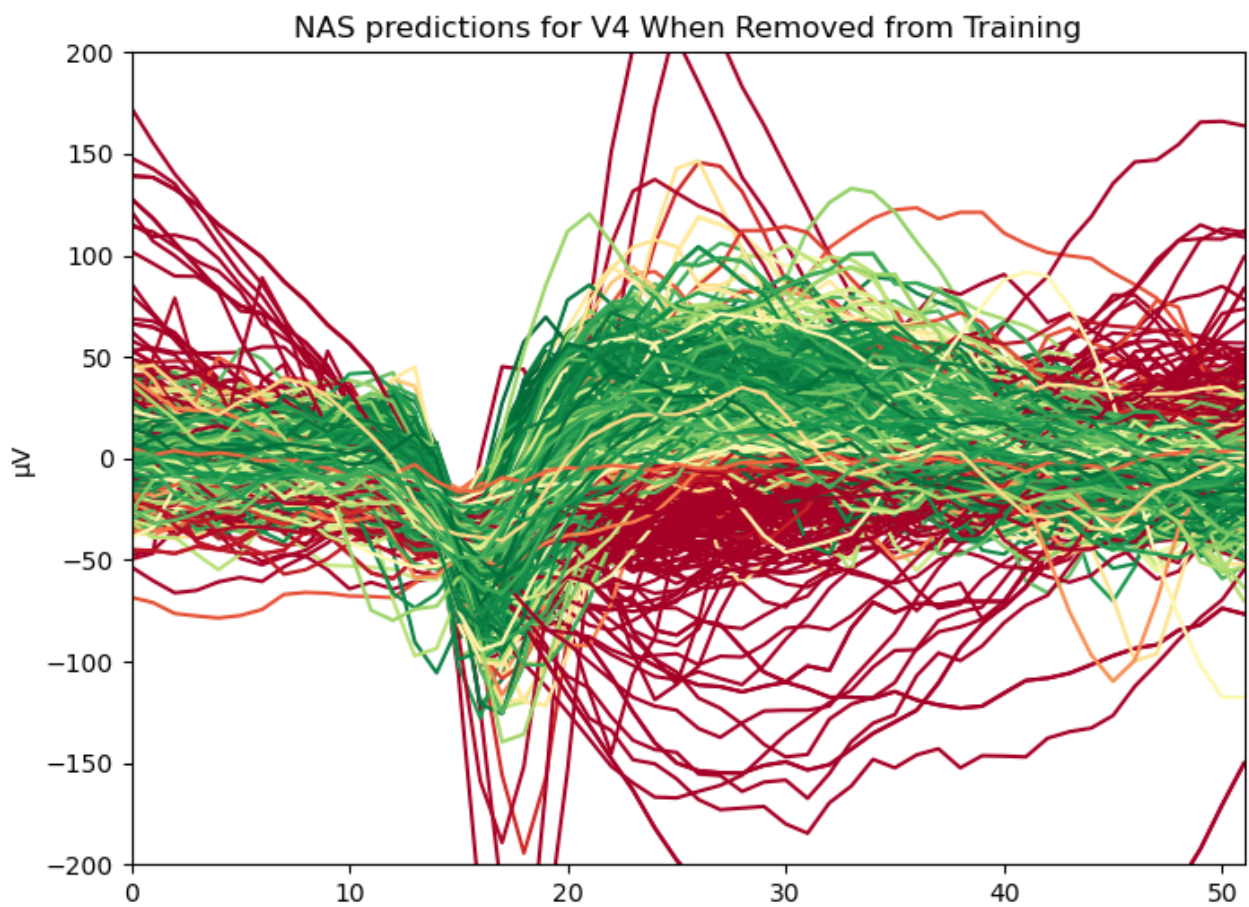
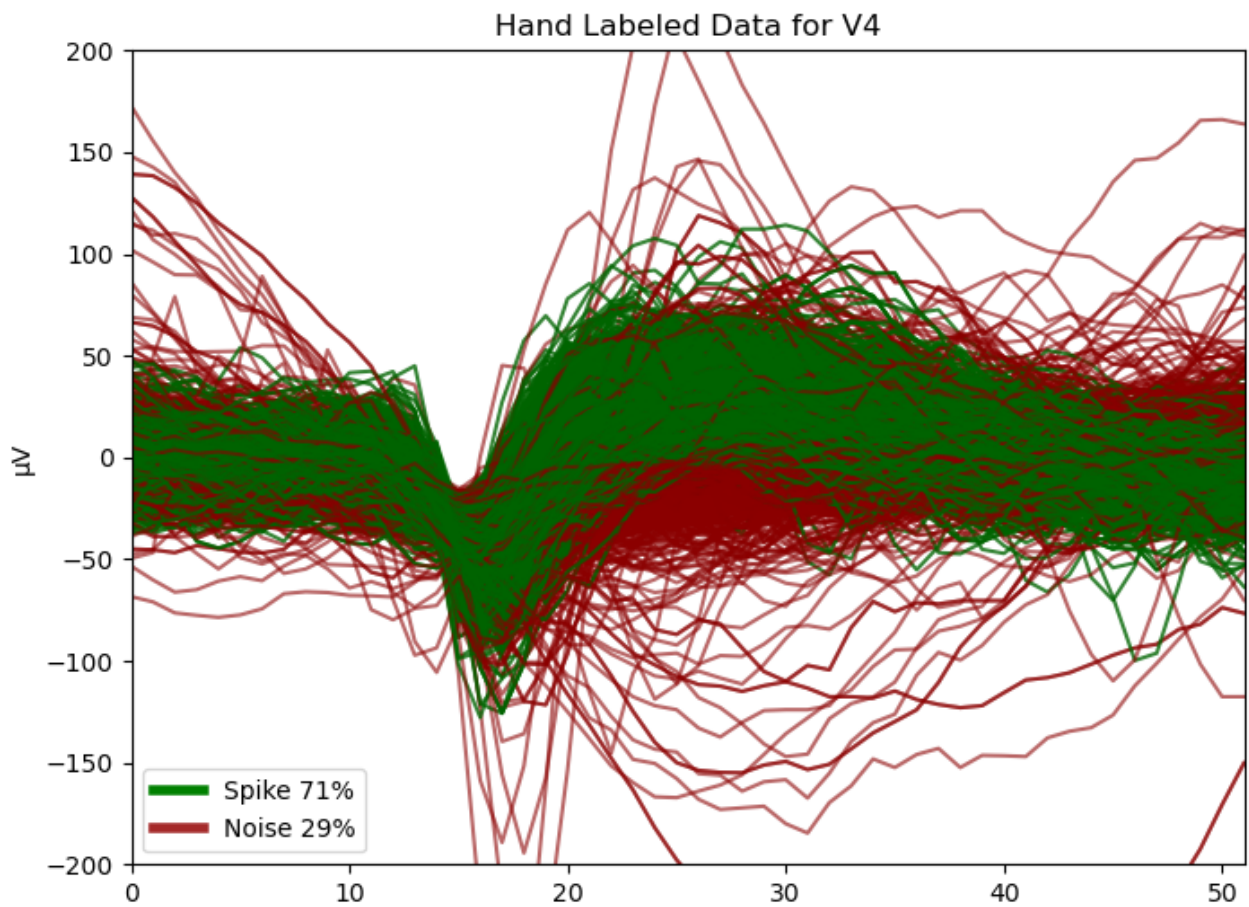
fig, ax = plt.subplots(figsize=(10, 6), dpi=100)
for r in rngvals:
    ax.plot(t, x_test[r,:], c=cmap.to_rgba(predicts_CNN[r,0]))
cbar = fig.colorbar(cmap)
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
cbar.ax.set_ylabel('$p_{spike}$', rotation=270)
plt.title('CNN predictions for ' +lab+' When Removed from Training')
plt.ylabel(u'\u03bcV')
plt.xlim([0, 51])
plt.ylim([-lims,lims])
plt.show()

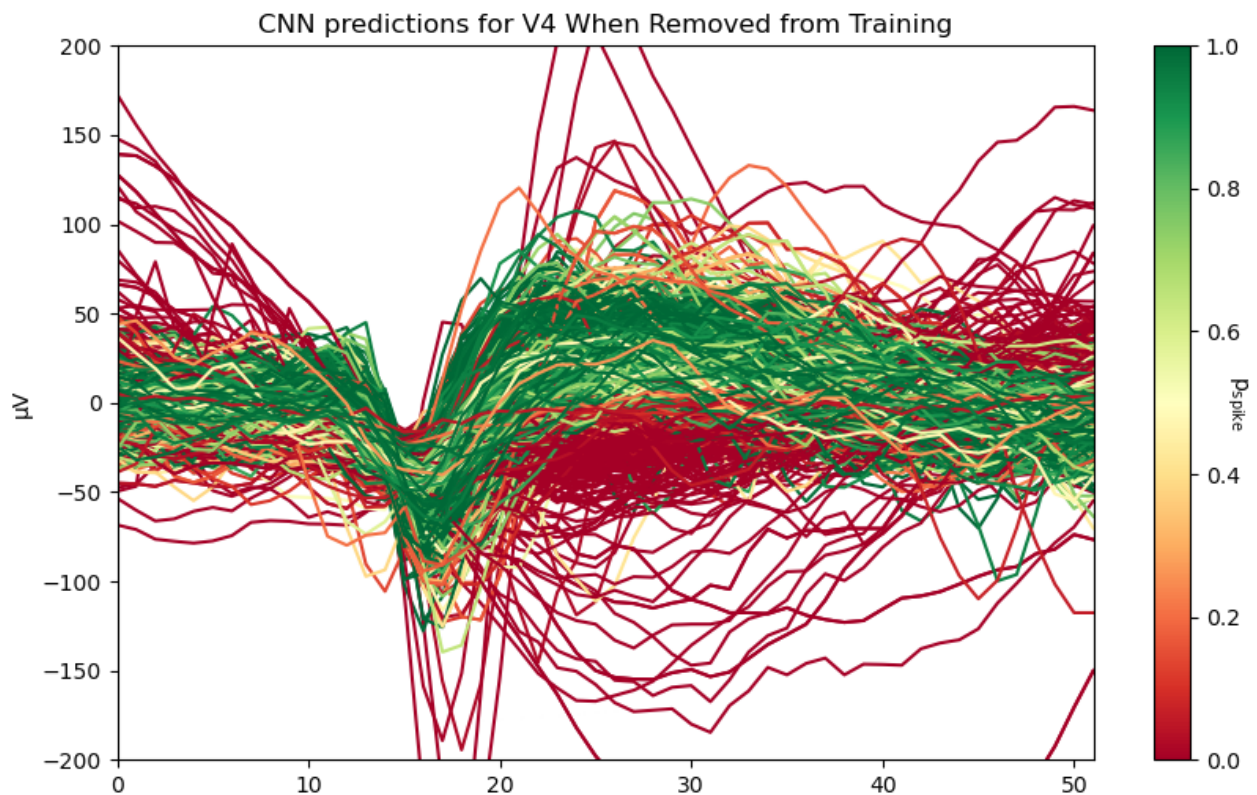
```

```

test_V4_ch46.mat
SmithCNNnet has acc of: [[5.7647729e-01]
[4.4530813e-07]
[8.0015409e-01]
...
[8.9890397e-01]
[9.9204135e-01]
[9.8123592e-01]] with 222576 params.
NAS_full has acc of: [[0.53182775]
[0.00352195]
[0.59787434]
...
[0.94488055]
[0.85020876]
[0.88086283]] with 2701 params.

```





```
In [11]: #figure 4 m1 plots
# uses networks that were trained on lacking m1 data

data_dir = 'Z:/SmithLab/prjNAS/NASNet/small_m1/' #all areas in separate files
testingFiles = []
for file in os.listdir(data_dir): # pull all training files from directory
    if file.endswith(".mat"):
        testingFiles.append(file)
nsets = len(testingFiles) # number of training files

netnames_alt = ['CNN4L2d_opt_noM1', 'NAS_noM1']
for j in range(nsets): # loop over the different testing files
    f_waves = h5py.File(data_dir + '/' + testingFiles[j], mode='r') # Load testing
    testacc = []
    fozle = testingFiles[j]
    file_edit = fozle.rfind('_')
    datastr = fozle[file_edit+1: len(fozle)-4]
    print(fozle)
    waveforms = np.transpose(f_waves['waveData'][(0)]) #load all waveforms
    #waveforms = waveforms[np.random.choice(len(waveforms), 5000000), :]
    for netname in netnames_alt:
        match = netnames_alt.index(netname)
        model_path = pjoin(model_dir, netname)
        model = keras.models.load_model(model_path)

        if match < numcnn:

            x_test = waveforms[:, 1:(52+1)] # waveform voltage values
            y_test = waveforms[:, 0] # waveform labels
            x_test = np.expand_dims(x_test, axis=2)
            predicts_CNN = model.predict(x_test, verbose =0)
```

```

print(netname + " has acc of: " + str(predicts_CNN) + " with " + str(model.c
#print(predicts_CNN.shape)

else:
    #print('we made it')
    x_test = waveforms[:, 1:(ntimepts+1)] # waveform voltage values
    #x_test = np.expand_dims(x_test, axis=2)
    y_test = waveforms[:, 0] # waveform labels

    predicts_NAS = model.predict(x_test, verbose = 0)
    print(netname + " has acc of: " + str(predicts_NAS) + " with " + str(model.c
    #print(predicts_NAS.shape)

best_g_NAS = 0
best_g_CNN = 0
g_of_CNN = 0.2
g_of_NAS = 0.2
ranger = np.arange(0, len(y_test), 1)
spikes_are = ranger[y_test == 1]
noise_are = ranger[y_test == 0]

lims = 200
#ch = fozle[fozle.find('_SNR')-2:fozle.find('_SNR')]
ch = fozle[fozle.find('ch')+2:fozle.find('ch')+3]
lab = 'M1'
rngvals = np.random.choice(len(predicts_NAS), 3000)
t = np.arange(0, 52)
plt.figure(figsize=(8, 6), dpi=100)
for r in rngvals:
    if y_test[r] == 1:
        plt.plot(t, x_test[r, :], c='darkgreen', alpha=0.8)
    else:
        plt.plot(t, x_test[r, :], c='darkred', alpha=0.6)
plt.title('Hand Labeled Data for ' + lab)
plt.ylabel(u'\u03bcV')
plt.xlim([0, 51])
plt.ylim([-lims, lims])
custom_lines = [mpl.lines.Line2D([0], [0], color='g', lw=4),
                 mpl.lines.Line2D([0], [0], color='brown', lw=4)]

plt.legend(custom_lines, ['Spike ' + str(round(sum(y_test)*100/len(y_test)))+'%
plt.show()

n_lines = 10
c = np.arange(1, n_lines + 1)

norm = mpl.colors.Normalize(vmin=0, vmax=1)
cmap = mpl.cm.ScalarMappable(norm=norm, cmap='RdYlGn')
cmap.set_array([])

fig, ax = plt.subplots(figsize=(8, 6), dpi=100)
for r in rngvals:
    ax.plot(t, x_test[r, :], c=cmap.to_rgba(predicts_NAS[r, 0]))
#fig.colorbar(cmap)

plt.title('NAS predictions for ' + lab + ' When Removed from Training')
plt.ylabel(u'\u03bcV')
plt.xlim([0, 51])

```

```

plt.ylim([-lims ,lims])
plt.show()

fig, ax = plt.subplots(figsize=(10, 6), dpi=100)
for r in rngvals:
    ax.plot(t, x_test[r,:], c=cmap.to_rgba(predicts_CNN[r,0]))
cbar = fig.colorbar(cmap)
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
cbar.ax.set_ylabel('$p_{spike}$', rotation=270)
plt.title('CNN predictions for ' + lab + ' When Removed from Training')
plt.ylabel(u'\u03bcV')
plt.xlim([0, 51])
plt.ylim([-lims,lims])
plt.show()

```

M1_52_testsmall.mat

CNN4L2d_opt_noM1 has acc of: [[7.1147454e-01]

[9.9899447e-01]

[7.4127906e-06]

...

[7.4729371e-01]

[8.4164274e-01]

[9.8519814e-01]] with 222576 params.

NAS_noM1 has acc of: [[8.5313957e-30]

[0.0000000e+00]

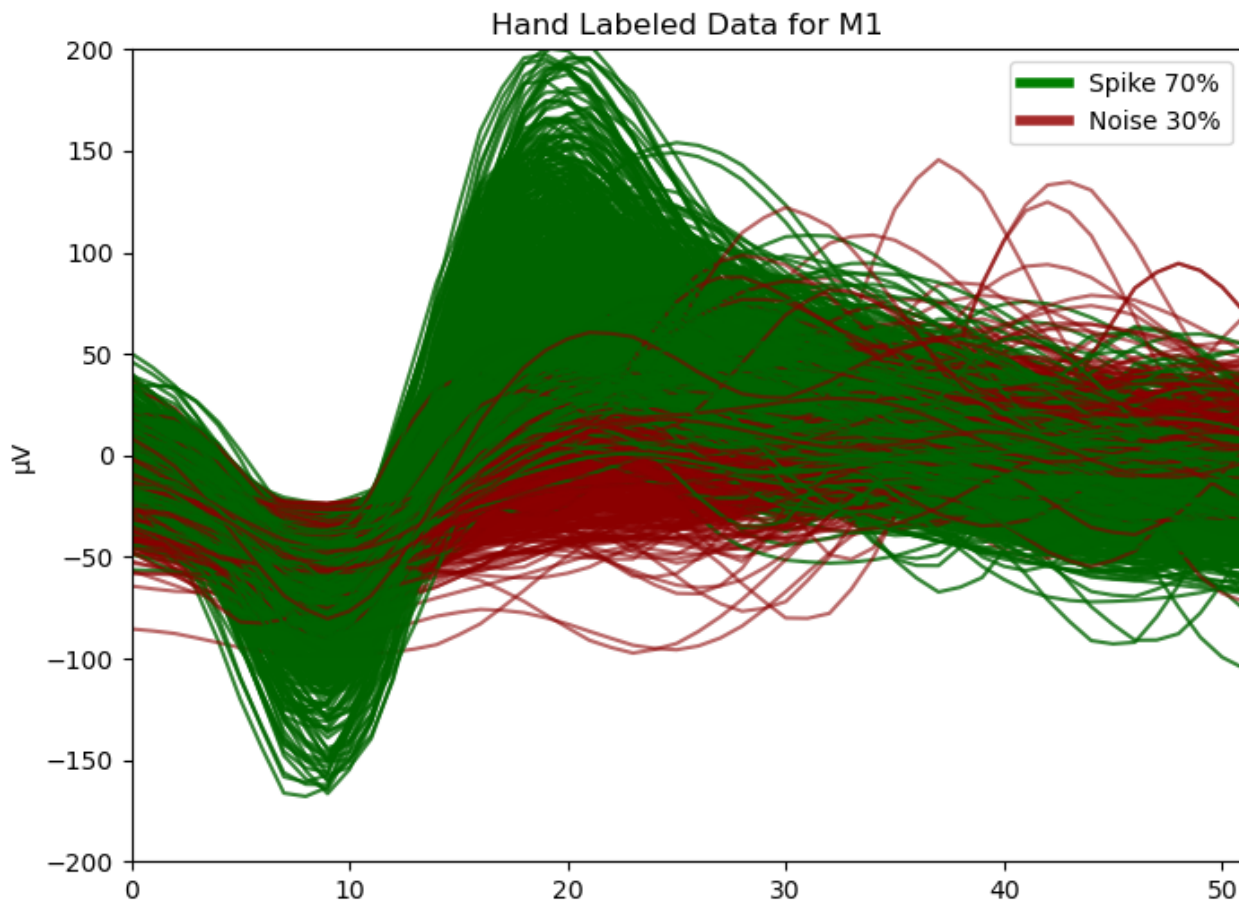
[2.8687716e-04]

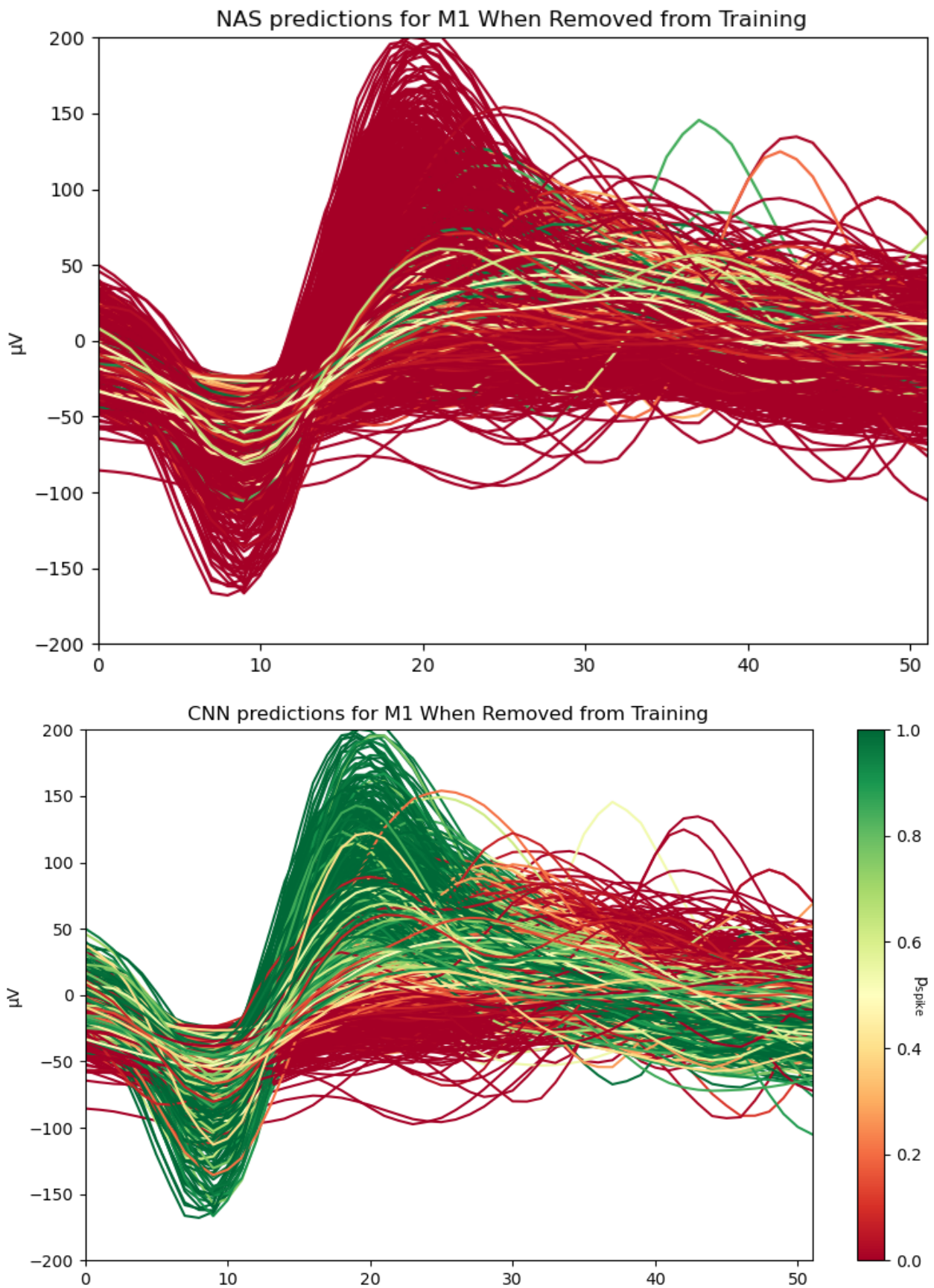
...

[1.4983988e-28]

[7.2167754e-02]

[0.0000000e+00]] with 2701 params.





```
In [12]: data_dir = 'Z:/SmithLab/prjNAS/NASNet/test_all/' #all areas in one file
testingFiles = []
for file in os.listdir(data_dir): # pull all training files from directory
    if file.endswith(".mat"):
```

```

testingFiles.append(file)
nsets = len(testingFiles) # singular testing file

for j in range(nsets): # Loop over the different testing files
    f_waves = h5py.File(data_dir + '/' + testingFiles[j], mode='r') # Load testing
    testacc = []
    fozle = testingFiles[j]
    file_edit = fozle.rfind('_')
    datastr = fozle[file_edit+1: len(fozle)-4]
    print(fozle)
    waveforms = np.transpose(f_waves['waveData'][(0)]) #Load all waveforms
    #waveforms = waveforms[np.random.choice(len(waveforms), 5000000), :]
    for netname in netnames:
        match = netnames.index(netname)
        model_path = pjoin(model_dir, netname)
        model = keras.models.load_model(model_path)

        if match < numcnn:

            x_test = waveforms[:, 1:(52+1)] # waveform voltage values
            y_test = waveforms[:, 0] # waveform labels
            x_test = np.expand_dims(x_test, axis=2)
            #predicts_CNNall = model.predict(x_test, verbose =0)
            print(netname + " has acc of: " + str(predicts_CNNall) + " with " + str(mode)
            #print(predicts_CNN.shape)

        else:
            #print('we made it')
            x_test = waveforms[:, 1:(ntimepts+1)] # waveform voltage values
            #x_test = np.expand_dims(x_test, axis=2)
            y_test = waveforms[:, 0] # waveform labels

            predicts_NASall = model.predict(x_test, verbose =0)
            print(netname + " has acc of: " + str(predicts_NASall) + " with " + str(mode)
            #print(predicts_NAS.shape)

```

```

full_test.mat
SmithCNNnet has acc of: [[0.7809308 ]
[0.97026825]
[0.9793009 ]
...
[0.93613464]
[0.5778552 ]
[0.99800897]] with 222576 params.
NAS_full has acc of: [[0.6798997 ]
[0.9861808 ]
[0.8132093 ]
...
[0.99515414]
[0.86491585]
[0.9824934 ]] with 2701 params.

```

In [13]:

```

#figure 5 here, classification of all areas
#histogram distribution of waveform/spike max and accuracy on that specific bin index
gamma=0.48

predlabel_CNN = predicts_CNNall[:,0] > gamma

```

```

predlabel_NAS = predicts_NASall[:,0] > (gamma)
# Get the histogram
Y,X = np.histogram(max_idx, 52,range=((0,51)))
bins =52
t_waves = np.zeros(bins)
cor_waves = np.zeros((bins,2))
err_waves = np.zeros(bins)
for maxW in np.arange(0,52):
    idx =0
    for maxval in max_idx:
        t_waves[maxval] = t_waves[maxval] + 1
        if predlabel_CNN[idx] == y_test[idx]:
            cor_waves[maxval,1] = cor_waves[maxval,1] + 1
        if predlabel_NAS[idx] == y_test[idx]:
            cor_waves[maxval,0] = cor_waves[maxval,0] + 1
        idx = idx + 1

```

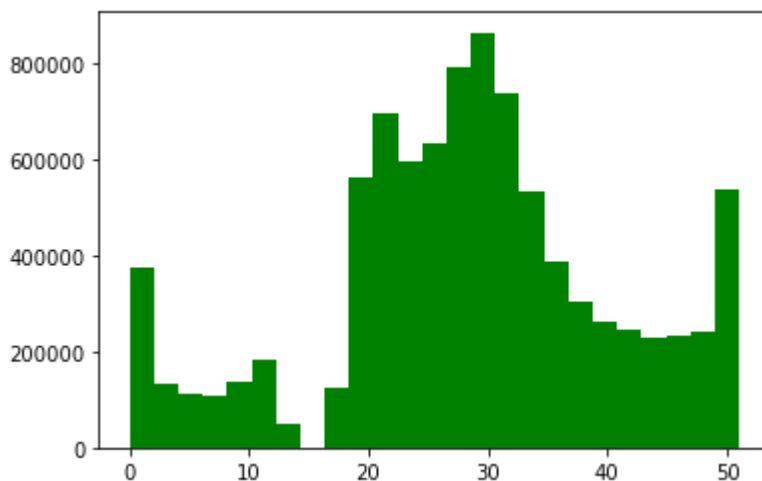
In [14]:

```

Ys,Xs = np.histogram(max_idx_spike, 52,range=((0,51)))
bins =52
t_spike = np.zeros(bins)
cor_spike = np.zeros((bins,2))
for maxW in np.arange(0,52):
    idx =0
    for maxval in max_idx:
        if y_test[idx] == 1:
            t_spike[maxval] = t_spike[maxval] + 1
            if predlabel_CNN[idx] == y_test[idx]:
                cor_spike[maxval,1] = cor_spike[maxval,1] + 1
            if predlabel_NAS[idx] == y_test[idx]:
                cor_spike[maxval,0] = cor_spike[maxval,0] + 1
        idx = idx + 1

n, bins, patches = plt.hist(max_idx, 25, color='green')
bin_centers = 0.5 * (bins[:-1] + bins[1:])
plt.show()

```



In [15]:

```

#binned accuracy for each network type over relative waveform max
p_NAS = cor_waves[:,0]/t_waves
p_CNN = cor_waves[:,1]/t_waves
...
...

```

```

norm1 = p_NAS/np.linalg.norm(p_NAS)
norm2 = p_CNN/np.linalg.norm(p_CNN)
col = bin_centers - min(bin_centers)
col /= max(col)
c_map = plt.cm.get_cmap('binary', 15)
p_both = np.concatenate([p_NAS, p_CNN])
cm = c_map(p_both)

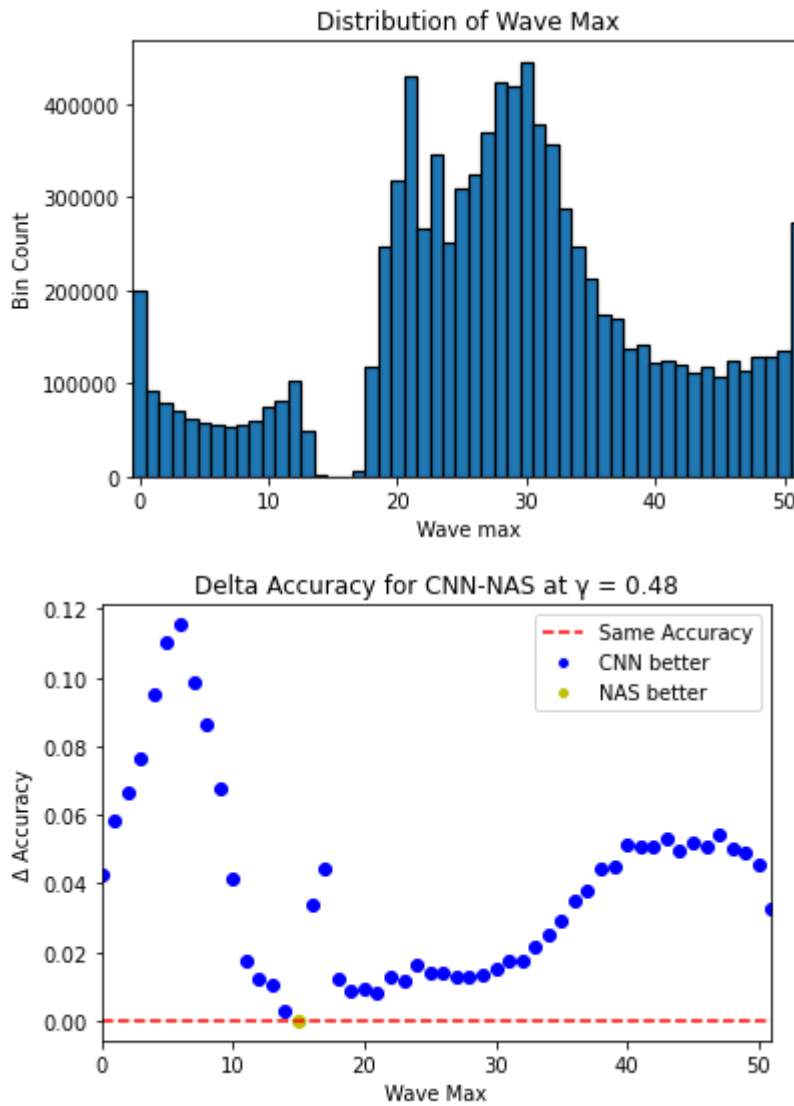
norm1 = p_NAS/np.linalg.norm(p_NAS)
norm2 = p_CNN/np.linalg.norm(p_CNN)
col = bin_centers - min(bin_centers)
col /= max(col)
c_map = plt.cm.get_cmap('binary', 15)
p_both = np.concatenate([p_NAS, p_CNN])
cm = c_map(p_both)
# histograms and delta accuracies
Y,X = np.histogram(max_idx, 52, range=(0,52))
x_span = X.max()-X.min()
bin_centers = 0.5 * (X[:-1] + X[1:])
#ax = plt.bar(X[:-1],Y,color=cm[0:51,:],width=X[1]-X[0])
ax = plt.bar(X[:-1],Y,width=X[1]-X[0],edgecolor='black', linewidth=1.2)
plt.xlim([0-0.5,51+0.5])
#sm = plt.cm.ScalarMappable(cmap=c_map, norm=plt.Normalize(min(p_both),max(p_both)))
#sm.set_array([])

#cbar = plt.colorbar(sm)
plt.title("Distribution of Wave Max")
plt.xlabel('Wave max')
plt.ylabel('Bin Count')
plt.show()

plt.figure()
idxa = 0
CNNbetterc = 0
for p in p_CNN:
    if p-p_NAS[idxa] > 0:
        plt.scatter(idxa,p-p_NAS[idxa],c='b')
        CNNbetterc = CNNbetterc+1
    else:
        plt.scatter(idxa,p-p_NAS[idxa],c='y')
    idxa = idxa+1
#plt.plot(np.arange(0,52),p_CNN-p_NAS,'b')
plt.plot(np.arange(0,52),np.zeros([52, 1]),'r--')
plt.title(u"Delta Accuracy for CNN-NAS at \u03B3 = 0.48")
plt.xlabel('Wave Max')
plt.ylabel(u'\u0394 Accuracy')
legend_elements = [mpl.lines.Line2D([0], [0], color='r', linestyle='--', label='Same Ac
                    mpl.lines.Line2D([0], [0], color='w', marker='o', label='CNN better', markerfacecol
                    mpl.lines.Line2D([0], [0], marker='o', color='w', label='NAS better', marker

plt.legend(handles=legend_elements, loc='upper right')
plt.xlim([0, 51])
plt.show()
print(CNNbetterc)

```



51

In [16]:

```
#binned accuracy for each network type over relative spike waveform max
p_NAS = cor_spike[:,0]/t_spike
p_CNN = cor_spike[:,1]/t_spike

norm1 = p_NAS/np.linalg.norm(p_NAS)
norm2 = p_CNN/np.linalg.norm(p_CNN)
col = bin_centers - min(bin_centers)
col /= max(col)
c_map = plt.cm.get_cmap('binary', 15)
p_both = np.concatenate([p_NAS, p_CNN])
cm = c_map(p_both)

Y,X = np.histogram(max_idx_spike, 52, range=(0,52))
x_span = X.max()-X.min()

ax = plt.bar(X[:-1],Y,width=X[1]-X[0],edgecolor='black', linewidth=1.2)
plt.xlim([0-0.5,51+0.5])
#sm = plt.cm.ScalarMappable(cmap=c_map, norm=plt.Normalize(min(p_both),max(p_both)))
#sm.set_array([])

plt.title("Spikes Only Wave Max distribution")
plt.xlabel('Wave max')
```

```

plt.ylabel('Bin Count')
plt.show()
...

...

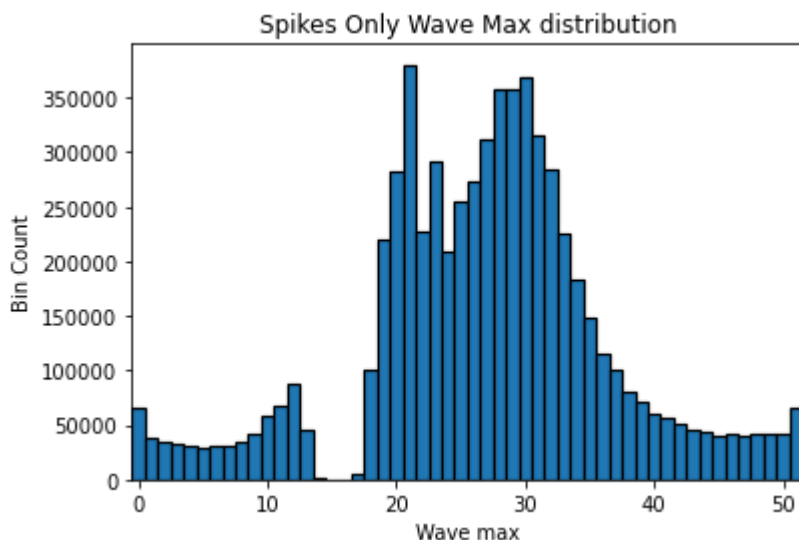
plt.figure()
idxa = 0
CNNbetterc = 0
for p in p_CNN:
    if p-p_NAS[idxa] > 0:
        plt.scatter(idxa,p-p_NAS[idxa],c='b')
        CNNbetterc = CNNbetterc+1
    else:
        plt.scatter(idxa,p-p_NAS[idxa],c='y')
    idxa = idxa+1
#plt.plot(np.arange(0,52),p_CNN-p_NAS,'b')
plt.plot(np.arange(0,52),np.zeros([52, 1]),'r--')

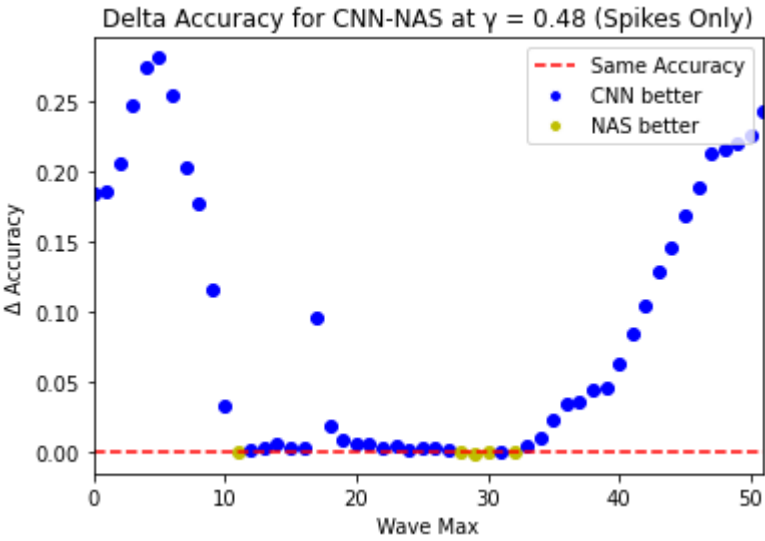
plt.title(u"Delta Accuracy for CNN-NAS at \u03B3 = 0.48 (Spikes Only)")
plt.xlabel('Wave Max')
plt.ylabel(u'\u0394 Accuracy')
#plt.Legend(['Same Accuracy', 'CNN better', 'NAS better'])

legend_elements = [mpl.lines.Line2D([0], [0], color='r', linestyle='--', label='Same Ac
mpl.lines.Line2D([0], [0], color='w', marker='o', label='CNN better', markerfacecol
mpl.lines.Line2D([0], [0], marker='o', color='w', label='NAS better', marker

plt.legend(handles=legend_elements, loc='upper right')
plt.xlim([0, 51])
plt.show()
print(CNNbetterc)

```





47