GINA CODY
SCHOOL OF ENGINEERING
AND COMPUTER SCIENCE

COEN 6331: Neural Networks

ASSIGNMENT -4

Submitted by:

Rohan Kodavalla (40196377)

I certify that this submission is my original work and meets the Faculty's Expectations of Originality

March 31, 2022

ABSTRACT

The objective of this assignment is to design an ART network which learns storing and recognizing the alphabets A to T (patterns of size 8*8) and evaluate the performance of network by choosing different vigilance levels (ranging from 0.3 to 0.95), along with evaluating the network by adding noise of 10-25% to the patterns.

1. ART network with vigilance set to 0.3, 0.4, 0.5:

For this specific simulation for alphabet recognition the input is a 8X8 grid of nodes that represent an upper-case letter in English alphabet. There are 64 (8X8) nodes in F1 layer representing the best hypothesized letter learned so far, and 26 nodes within F2 layer (done in background), each representing a possible letter in the alphabet, when the vigilance is set to -

Vigilance = 0.3

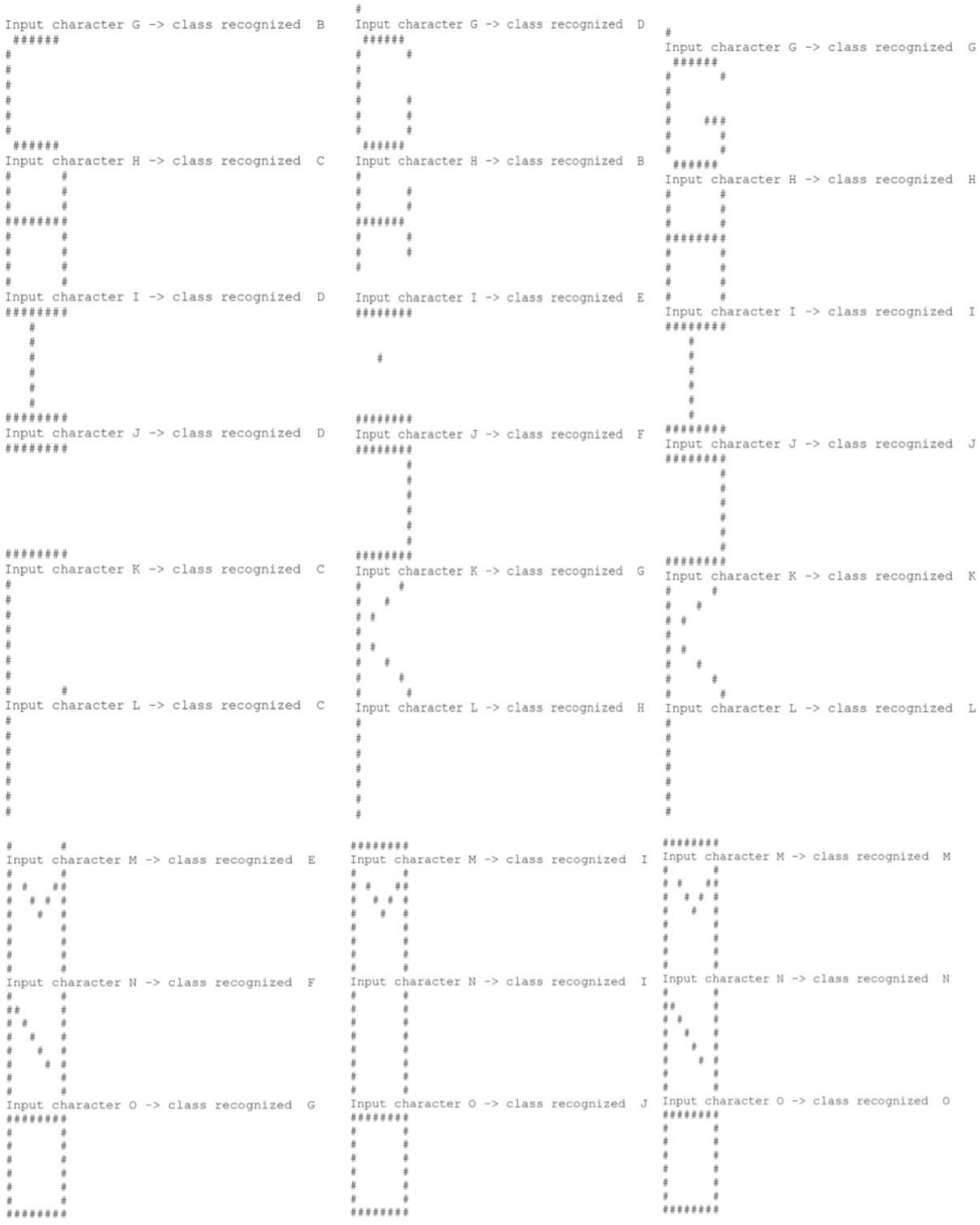
```
rho value is -->  0.3
Input character A -> class recognized  A
 ######
#      #
#      #
########
#      #
#      #
#      #
#      #
Input character B -> class recognized  A
 ######
#      #
#      #
#######
#      #
#      #
#
Input character C -> class recognized  A
 ######
#      #
#
#
#
#
#
Input character D -> class recognized  B
#######
#      #
#      #
#      #
#      #
#      #
#      #
#######
Input character E -> class recognized  B
#######
#
#
#      #
#
#
#
#######
Input character F -> class recognized  A
 ######
#
#
#
#
#
#
```

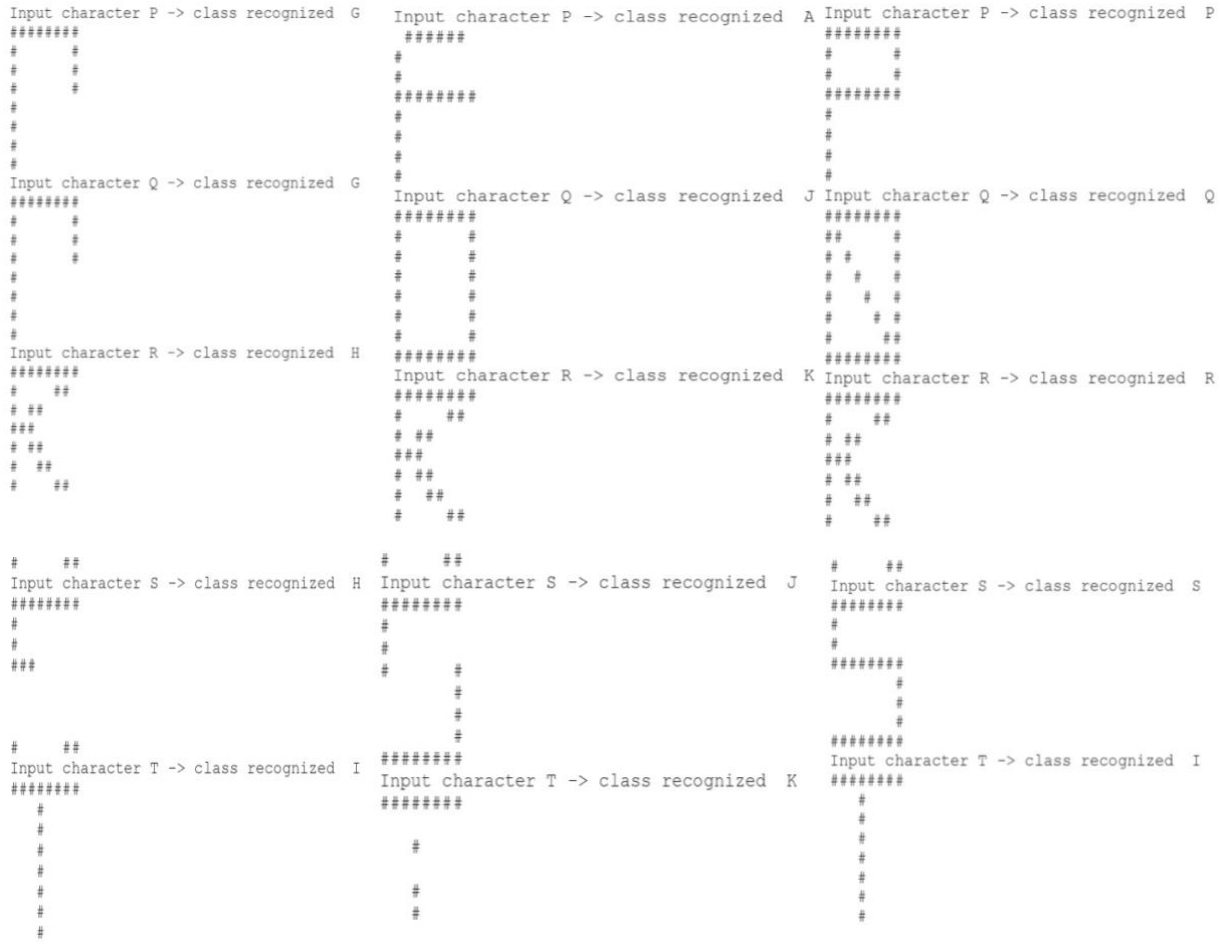
Vigilance = 0.4

```
rho value is -->  0.4
Input character A -> class recognized  A
 ######
#      #
#      #
########
#      #
#      #
#      #
#      #
Input character B -> class recognized  B
#######
#      #
#      #
#######
#      #
#      #
#######
Input character C -> class recognized  C
 ######
#      #
#
#
#
#
#
#      #
 ######
Input character D -> class recognized  D
#######
#      #
#      #
#      #
#      #
#      #
#      #
#######
Input character E -> class recognized  E
########
#
#
########
#
#
#
########
Input character F -> class recognized  A
 ######
#
#
########
#
#
#
```

Vigilance = 0.5

```
rho value is -->  0.5
Input character A -> class recognized  A
 ######
#      #
#      #
########
#      #
#      #
#      #
#      #
Input character B -> class recognized  B
#######
#      #
#      #
#######
#      #
#      #
#######
Input character C -> class recognized  C
 ######
#      #
#
#
#
#
#
#      #
 ######
Input character D -> class recognized  D
#######
#      #
#      #
#      #
#      #
#      #
#      #
#######
Input character E -> class recognized  E
########
#
#
########
#
#
#
########
Input character F -> class recognized  F
########
#
#
########
#
#
#
```


Input character G -> class recognized B
Input character H -> class recognized C
Input character I -> class recognized D
Input character J -> class recognized D
Input character K -> class recognized C
Input character L -> class recognized C
Input character M -> class recognized E
Input character N -> class recognized F
Input character O -> class recognized G
Input character G -> class recognized D
Input character H -> class recognized B
Input character I -> class recognized E
Input character J -> class recognized F
Input character K -> class recognized G
Input character L -> class recognized H
Input character M -> class recognized I
Input character N -> class recognized I
Input character O -> class recognized J
Input character G -> class recognized G
Input character H -> class recognized H
Input character I -> class recognized I
Input character J -> class recognized J
Input character K -> class recognized K
Input character L -> class recognized L
Input character M -> class recognized M
Input character N -> class recognized N
Input character O -> class recognized O



Based on above simulations for vigilance parameter set to 0.3, 0.4, 0.5 we observe that very general memories were being formed and letters were not determined correctly in most of the cases.

So let us increase the vigilance to values of 0.7 and0.95, let us observe the patterns that are to be determined.

2. ART network with vigilance set to 0.7 and 0.95:

For this specific simulation for alphabet recognition the input is a 8X8 grid of nodes that represent an upper-case letter in English alphabet. There are 64 (8X8) nodes in F1 layer representing the best hypothesized letter learned so far, and 26 nodes within F2 layer (done in background), each representing a possible letter in the alphabet, when the vigilance is set to-

Vigilance = 0.7

```
rho value is -->  0.7
Input character A -> class recognized  A
 ######
#      #
#      #
########
#      #
#      #
#      #
#      #
Input character B -> class recognized  B
#######
#      #
#      #
#######
#      #
#      #
#######

Input character C -> class recognized  C
 ######
#      #
#
#
#
#
#      #
 ######
Input character D -> class recognized  D
#######
#      #
#      #
#      #
#      #
#      #
#      #
#######
Input character E -> class recognized  E
########
#
#
########
#
#
#
########
Input character F -> class recognized  F
########
#
#
########
#
#
#
```

Vigilance = 0.95

```
rho value is -->  0.95
Input character A -> class recognized  A
 ######
#      #
#      #
########
#      #
#      #
#      #
#      #
Input character B -> class recognized  B
#######
#      #
#      #
#######
#      #
#      #
#######

Input character C -> class recognized  C
 ######
#      #
#
#
#
#
#      #
 ######
Input character D -> class recognized  D
#######
#      #
#      #
#      #
#      #
#      #
#      #
#######
Input character E -> class recognized  E
########
#
#
########
#
#
#
########
Input character F -> class recognized  F
########
#
#
########
#
#
#
```

```
#
Input character G -> class recognized  G
 ######
#       #
#
#
#    ###
#       #
#       #
 ######
Input character H -> class recognized  H
#       #
#       #
#       #
########
#       #
#       #
#       #
#       #
Input character I -> class recognized  I
########
   #
   #
   #
   #
   #
   #
########
Input character J -> class recognized  J
########
        #
        #
        #
        #
        #
        #
########
Input character K -> class recognized  K
#      #
#    #
# #
#
# #
#    #
#      #
#       #
Input character L -> class recognized  L
#
#
#
#
#
#
#
```

```
#
Input character G -> class recognized  G
 ######
#       #
#
#
#    ###
#       #
#       #
 ######
Input character H -> class recognized  H
#       #
#       #
#       #
########
#       #
#       #
#       #
#       #
Input character I -> class recognized  I
########
   #
   #
   #
   #
   #
   #
########
Input character J -> class recognized  J
########
        #
        #
        #
        #
        #
        #
########
Input character K -> class recognized  K
#      #
#    #
# #
#
# #
#    #
#      #
#       #
Input character L -> class recognized  L
#
#
#
#
#
#
#
```

```
########
Input character M -> class recognized  M
#       #
# #   ##
#  # # #
#   #  #
#       #
#       #
#       #
#       #
Input character N -> class recognized  N
#       #
##      #
# #     #
#  #    #
#   #   #
#    # #
#       #
#       #
Input character O -> class recognized  O
########
#       #
#       #
#       #
#       #
#       #
#       #
########
Input character P -> class recognized  P
########
#       #
#       #
########
#
#
#
#
Input character Q -> class recognized  Q
########
##      #
# #     #
#  #    #
#   #   #
#    # #
#     ##
########
Input character R -> class recognized  R
########
#    ##
# ##
###
# ##
#  ##
#    ##
```


```
########
Input character M -> class recognized  M
#       #
# #   ##
#  # # #
#   #  #
#       #
#       #
#       #
#       #
Input character N -> class recognized  N
#       #
##      #
# #     #
#  #    #
#   #   #
#    # #
#       #
#       #
Input character O -> class recognized  O
########
#       #
#       #
#       #
#       #
#       #
#       #
########
Input character P -> class recognized  P
########
#       #
#       #
########
#
#
#
#
Input character Q -> class recognized  Q
########
##      #
# #     #
#  #    #
#   #   #
#    # #
#     ##
########
Input character R -> class recognized  R
########
#    ##
# ##
###
# ##
#  ##
#    ##
```

```
#      ##
Input character S -> class recognized  S
########
#
#
########
        #
        #
        #
########
Input character T -> class recognized  T
########
    #
    #
    #
    #
    #
    #
    #
```

```
#      ##
Input character S -> class recognized  S
########
#
#
########
        #
        #
        #
########
Input character T -> class recognized  T
########
    #
    #
    #
    #
    #
    #
    #
```

As observed, by increasing vigilance value, we see the pattern learned is in fact the alphabets desired. The drawback is that it takes a bit longer to learn with a higher vigilance value than a smaller one.

3. ART network with vigilance set to 0.3, 0.4, 0.5 introducing noise:

i. Noise = 10%

Vigilance = 0.3

```
rho value is -->  0.3
Noise rate = 10%
Input character A -> class recognized A
 #    #
#      #
#      #
#  #####
#      #
#      #
#      #
#      #
Input character B -> class recognized B
#######
       #
       #
##  ###
#      #
#      #
#######

Input character C -> class recognized A
 #    #
#      #
#
#


#
#      #

Input character D -> class recognized B
###  ##
       #
       #
#
#
#      #
#

Input character E -> class recognized C
####  ##
#
#
######

#
#
##  ####
Input character F -> class recognized C
#  #  ##
#
#
# ####
```

Vigilance = 0.5

```
rho value is -->  0.5
Noise rate = 10%
Input character A -> class recognized A
 #    #
#      #
#      #
#  #####
#      #
#      #
#      #
#      #
Input character B -> class recognized B
#######
       #
       #
##  ###
#      #
#      #
#######

Input character C -> class recognized C
 #  ###
#      #
#
#


#
#      #
 #### #
Input character D -> class recognized D
###  ##
#      #
       #
#      #
#
#      #
#      #
##### #
Input character E -> class recognized E
####  ##
#
#
######

#
#
##  ####
Input character F -> class recognized F
#  #####
#
#
# ######
```

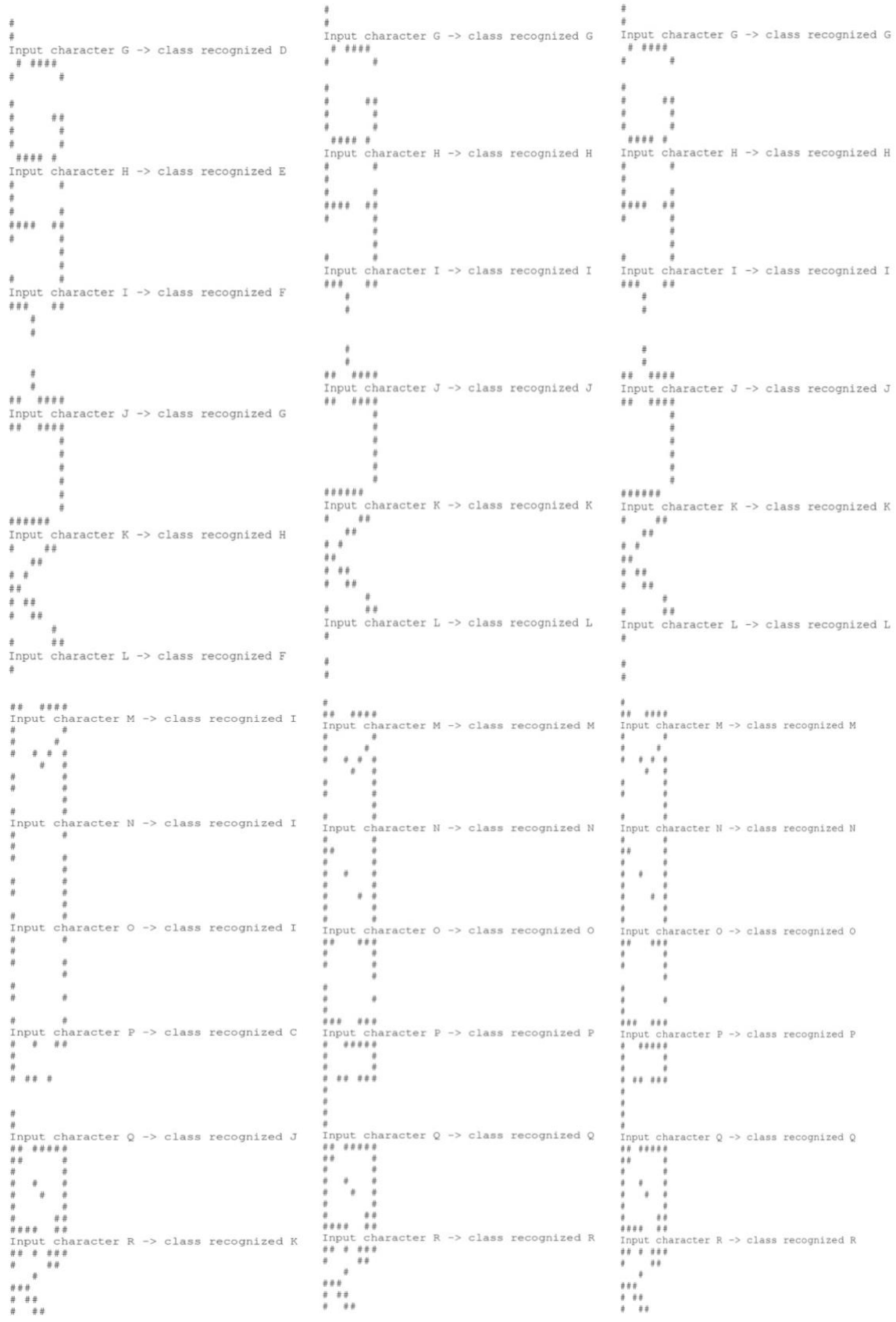
Vigilance = 0.95

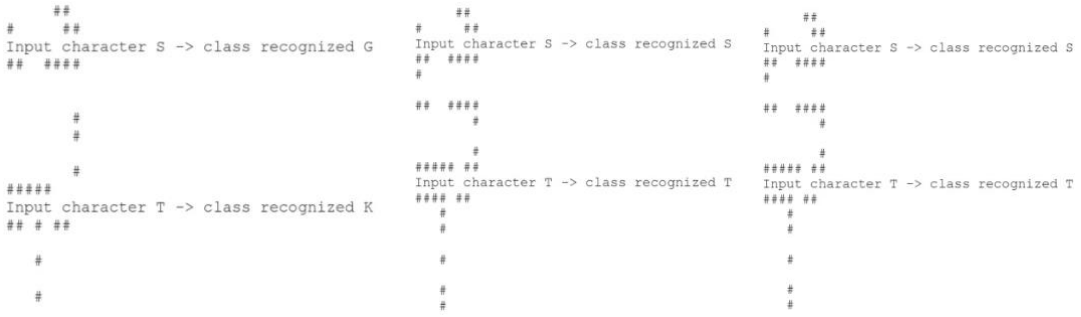
```
rho value is -->  0.95
Noise rate = 10%
Input character A -> class recognized A
 #    #
#      #
#      #
#  #####
#      #
#      #
#      #
#      #
Input character B -> class recognized B
#######
       #
       #
##  ###
#      #
#      #
#######

Input character C -> class recognized C
 #  ###
#      #
#
#


#
#      #
 #### #
Input character D -> class recognized D
###  ##
#      #
       #
#      #
#
#      #
#      #
##### #
Input character E -> class recognized E
####  ##
#
#
######

#
#
##  ####
Input character F -> class recognized F
#  #####
#
#
# ######
```


Input character G -> class recognized D
Input character H -> class recognized E
Input character I -> class recognized F
Input character J -> class recognized G
Input character K -> class recognized H
Input character L -> class recognized F
Input character M -> class recognized I
Input character N -> class recognized I
Input character O -> class recognized I
Input character P -> class recognized C
Input character Q -> class recognized J
Input character R -> class recognized K
Input character G -> class recognized G
Input character H -> class recognized H
Input character I -> class recognized I
Input character J -> class recognized J
Input character K -> class recognized K
Input character L -> class recognized L
Input character M -> class recognized M
Input character N -> class recognized N
Input character O -> class recognized O
Input character P -> class recognized P
Input character Q -> class recognized Q
Input character R -> class recognized R
Input character G -> class recognized G
Input character H -> class recognized H
Input character I -> class recognized I
Input character J -> class recognized J
Input character K -> class recognized K
Input character L -> class recognized L
Input character M -> class recognized M
Input character N -> class recognized N
Input character O -> class recognized O
Input character P -> class recognized P
Input character Q -> class recognized Q
Input character R -> class recognized R



As observed in above simulations, for vigilance = 0.3 the alphabets were not determined correctly in most of the cases, but for 0.5 and 0.9 there was no issue at all.

ii. Now let us observe for - Noise=15%

Vigilance = 0.3

```
rho value is -->  0.3
Noise rate = 15%
Input character A -> class recognized A

Input character B -> class recognized B

Input character C -> class recognized A

Input character D -> class recognized C

Input character E -> class recognized D

Input character F -> class recognized D
```

Vigilance = 0.95

```
rho value is -->  0.95
Noise rate = 15%
Input character A -> class recognized A

Input character B -> class recognized B

Input character C -> class recognized C

Input character D -> class recognized D

Input character E -> class recognized E

Input character F -> class recognized F
```

```
#
#
Input character G -> class recognized E
 # ## #
#      #

#
#     ##
       #
#      #
 # ## #
Input character H -> class recognized F
#      #
#
#      #
# ##  ##
#      #

       #
#      #
Input character I -> class recognized G
# #   ##
   #
   #


   #
   #
##  # ##
Input character J -> class recognized E
 #  #
       #



       #
       #
 # ##
Input character K -> class recognized H
#    #
    #
# #
##
# #
#   #
      #
#     ##
Input character L -> class recognized D
#

#
#
```

```
#
#
Input character G -> class recognized G
 # ## #
#      #

#
#     ##
       #
#      #
 # ## #
Input character H -> class recognized H
#      #
#
#      #
# ##  ##
#      #

       #
#      #
Input character I -> class recognized I
# #   ##
   #
   #


   #
   #
##  # ##
Input character J -> class recognized J
##  ## #
       #

       #

       #
       #
#####
Input character K -> class recognized K
#    #
    #
# #
##
# #
#   #
      #
#     ##
Input character L -> class recognized L
#

#
#
```

```
#
#
Input character M -> class recognized I
#      #
#
#  # #
    #  #
#
#      #

#      #
Input character N -> class recognized F
#      #
#
       #
#  #
       #

       #
#      #
Input character O -> class recognized C
##   #
#      #
       #
       #

#      #
#
# #   #
Input character P -> class recognized J
#  ### #
#      #
#      #
#  # # #
#
#
#
#
Input character Q -> class recognized J
#  ### #
#      #
#
#  #   #
#
#
#
#
Input character R -> class recognized K
## # # #
#    ##
   #
# #
# ##
#  #


      #
#     ##
Input character S -> class recognized L
##  # ##
#

##  # ##
       #

       #
### # ##
Input character T -> class recognized K
#  # #

   #

   #
```

```
#
#   ## #
Input character M -> class recognized M
#      #
#
#  # #
    #  #
#
#      #

#      #
Input character N -> class recognized N
#      #
##
       #
#  #
       #
#    #
       #
#      #
Input character O -> class recognized O
##   # #
#      #
#      #
       #
#
#      #
#
###   ##
Input character P -> class recognized P
#  ### #
#      #
#      #
#  # # #
#
#
#
#
Input character Q -> class recognized Q
## ### #
##     #
#
#  #   #
#   #  #
#      #
#      #
####  ##
Input character R -> class recognized R
## # # #
#    ##
   #
# #
# ##
#  #


      #
#     ##
Input character S -> class recognized S
##  # ##
#

##  # ##
       #

       #
### # ##
Input character T -> class recognized T
# ## ##
   #
   #

   #

   #
   #
```

In above simulations, for vigilance = 0.3 the alphabets were not determined correctly in most of the cases, but the recognized pattern set has increased to different alphabets on compared when noise=10% (more recognized alphabets). 0.9 there was no issue at all in recognition.

iii.Now let us observe for - Noise=25%

Vigilance = 0.3

```
rho value is -->  0.3
Noise rate = 25%
Input character A -> class recognized A
 #    #
#      #
       #
       #
   #   #
#      #
#      #

#      #
Input character B -> class recognized B
#  #  #
       #
       #
##  # #
#      #
#      #
# ##  #

Input character C -> class recognized C
 #  # #
#

#

#
#      #
  # # #
Input character D -> class recognized C
      #
#

#

#
#      #
  # # #
Input character E -> class recognized D
#  #  ##
#
#
#  # #

#
#
##  #  #
Input character F -> class recognized D
#  #  ##

#
#
```

Vigilance = 0.95

```
rho value is -->  0.95
Noise rate = 25%
Input character A -> class recognized A
 #    #
#      #
       #
       #
   #   #
#      #
#      #

#      #
Input character B -> class recognized B
#  #  #
       #
       #
##  # #
#      #
#      #
# ##  #

Input character C -> class recognized C
 #  # #
#

#

#
#      #
  # # #
Input character D -> class recognized D
# #  ##
#
       #
#      #

#
#      #
# # # #
Input character E -> class recognized E
#  #  ##
#
#
#  # #

#
#
##  #  #
Input character F -> class recognized F
#  #  ##

#
# # # ##
```

```
#
#
Input character G -> class recognized E
 #  # #
#      #

#
      ##
       #
#      #
 #  # #
Input character H -> class recognized F
#      #

#      #
#  #  #
#      #

       #
#      #
Input character I -> class recognized G
# #    #
   #
   #


   #
   #
##  # ##
Input character J -> class recognized H
##  #  #
       #

       #

       #
       #
#  ##
Input character K -> class recognized I
#    #
    #
# #
#
  #
#   #

#     ##
Input character L -> class recognized D
#

#
#
```

```
#
#
Input character G -> class recognized G
 #  # #
#      #

#
      ##
       #
#      #
 #  # #
Input character H -> class recognized H
#      #

#      #
#  #  #
#      #

       #
#      #
Input character I -> class recognized I
# #    #
   #
   #


   #
   #
##  # ##
Input character J -> class recognized J
##  #  #
       #

       #

       #
       #
#  ##
Input character K -> class recognized K
#    #
    #
# #
#
  #
#   #

#     ##
Input character L -> class recognized L
#

#
#
```

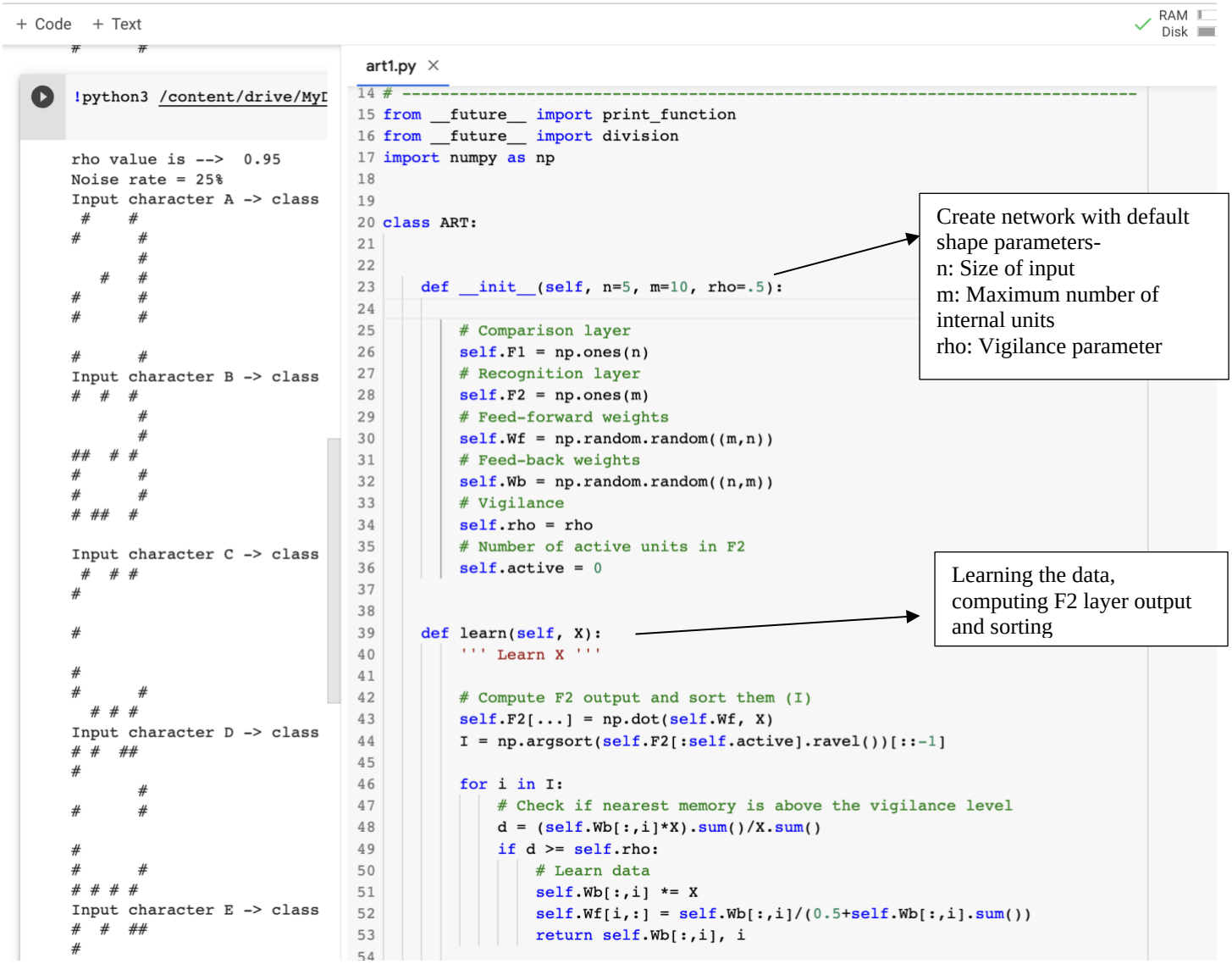

Input character M -> class recognized J
Input character N -> class recognized F
Input character O -> class recognized J
Input character P -> class recognized K
Input character Q -> class recognized L
Input character R -> class recognized M
Input character S -> class recognized H
Input character T -> class recognized N
Input character M -> class recognized M
Input character N -> class recognized N
Input character O -> class recognized O
Input character P -> class recognized P
Input character Q -> class recognized Q
Input character R -> class recognized R
Input character S -> class recognized S
Input character T -> class recognized T

Like previous simulations, for vigilance = 0.3 the alphabets were not determined correctly in most of the cases, however the recognized pattern set has increased more to different alphabets. 0.9 there was no issue at all in recognition, but the cost for the same is longer execution times.

CORRESPONDING CODE EXPLANATION FOR ART LEARNING, RECOGNITION PATTERNS SHOWN IN ABOVE RESULTS:

i) Importing the input patterns, noise –

Since an associative memory has polar states and patterns (or binary states and patterns), we convert the input image to a black and white image.




```
# ------------------------------------------------------------------------
from __future__ import print_function
from __future__ import division
import numpy as np


class ART:


    def __init__(self, n=5, m=10, rho=.5):

        # Comparison layer
        self.F1 = np.ones(n)
        # Recognition layer
        self.F2 = np.ones(m)
        # Feed-forward weights
        self.Wf = np.random.random((m,n))
        # Feed-back weights
        self.Wb = np.random.random((n,m))
        # Vigilance
        self.rho = rho
        # Number of active units in F2
        self.active = 0


    def learn(self, X):
        ''' Learn X '''

        # Compute F2 output and sort them (I)
        self.F2[...] = np.dot(self.Wf, X)
        I = np.argsort(self.F2[:self.active].ravel())[::-1]

        for i in I:
            # Check if nearest memory is above the vigilance level
            d = (self.Wb[:,i]*X).sum()/X.sum()
            if d >= self.rho:
                # Learn data
                self.Wb[:,i] *= X
                self.Wf[i,:] = self.Wb[:,i]/(0.5+self.Wb[:,i].sum())
                return self.Wb[:,i], i
```

```
 #  # #
#
#
#
#
#      #
  # # #
Input character D -> class
# #  ##
#
       #
#      #
```

```
# No match found, increase the number of active units
# and make the newly active unit to learn data
if self.active < self.F2.size:
    i = self.active
    self.Wb[:,i] *= X
    self.Wf[i,:] = self.Wb[:,i]/(0.5+self.Wb[:,i].sum())
    self.active += 1
    return self.Wb[:,i], i

return None,None
```

Modifying active units count based on matches to learn the data

```
#      #
```

```
!python3 /content/drive/MyD
```

```
rho value is -->  0.95
Noise rate = 25%
Input character A -> class
 #    #
#      #
       #
   #   #
#      #
#      #

#      #
Input character B -> class
#  #  #
       #
       #
##  # #
#      #
#      #
# ##  #

Input character C -> class
 #  # #
#

#

#
#      #
  # # #
Input character D -> class
# #  ##
#
       #
#      #

#
#      #
# # # #
Input character E -> class
#  #  ##
#
#
#  # #
```

art1.py

```
if __name__ == '__main__':

    np.random.seed(1)

    def letter_to_array(letter):
        ''' Convert a letter to a numpy array '''
        shape = len(letter), len(letter[0])
        Z = np.zeros(shape, dtype=int)
        for row in range(Z.shape[0]):
            for column in range(Z.shape[1]):
                if letter[row][column] == '#':
                    Z[row][column] = 1
        return Z

    def print_letter(Z):
        ''' Print an array as if it was a letter'''
        for row in range(Z.shape[0]):
            for col in range(Z.shape[1]):
                if Z[row,col]:
                    print( '#', end="" )
                else:
                    print( ' ', end="" )
            print( )



    A1=letter_to_array([' ###### ',
                        '#      #',
                        '#      #',
                        '########',
                        '#      #',
                        '#      #',
                        '#      #',
                        '#      #'])


    B1=letter_to_array(['####### ',
                        '#      #',
                        '#      #',
                        '####### ',
                        '#      #',
                        '#      #',
```

Converting letter to array

Print the array corresponding to the letter

Input alphabets A to T of size 8X8

art1.py

```
111  C1=letter_to_array([' ###### ',
112                      '#      #',
113                      '#       ',
114                      '#       ',
115                      '#.      ',
116                      '#       ',
117                      '#      #',
118                      ' ###### '])
119  D1=letter_to_array(['####### ',
120                      '#      #',
121                      '#      #',
122                      '#      #',
123                      '#      #',
124                      '#      #',
125                      '####### '])
126  E1=letter_to_array(['########',
127                      '#       ',
128                      '#       ',
129                      '########',
130                      '#       ',
131                      '#       ',
132                      '#       ',
133                      '########'])
134  F1=letter_to_array(['########',
135                      '#       ',
136                      '#       ',
137                      '########',
138                      '#       ',
139                      '#       ',
140                      '#       ',
141                      '#       '])
142  G= letter_to_array([' ###### ',
143                      '#      #',
144                      '#       ',
145                      '#       ',
146                      '#.   ###',
147                      '#      #',
148                      '#      #',
149                      ' ###### '])
150  H= letter_to_array(['#      #',
151                      '#      #',
152                      '#      #',
```

art1.py

```
151  H= letter_to_array(['#      #',
152                      '#      #',
153                      '#      #',
154                      '########',
155                      '#      #',
156                      '#      #',
157                      '#      #',
158                      '#      #'])
159  I= letter_to_array(['########',
160                      '   #    ',
161                      '   #    ',
162                      '   #    ',
163                      '   #    ',
164                      '   #    ',
165                      '   #    ',
166                      '########'])
167  J= letter_to_array(['########',
168                      '       #',
169                      '       #',
170                      '       #',
171                      '       #',
172                      '       #',
173                      '       #',
174                      '########'])
175  K= letter_to_array(['#     # ',
176                      '#   #   ',
177                      '# #     ',
178                      '#       ',
179                      '# #     ',
180                      '#   #   ',
181                      '#     # ',
182                      '#      #'])
183  L= letter_to_array(['#       ',
184                      '#       ',
185                      '#       ',
186                      '#       ',
187                      '#       ',
188                      '#       ',
189                      '#       ',
190                      '########'])
191  M= letter_to_array(['#      #',
192                      '# #   ##',
193                      '#  # # #',
```

art1.py

```
M= letter_to_array(['#      #',
                    '# #   ##',
                    '#  # # #',
                    '#   #  #',
                    '#      #',
                    '#      #',
                    '#      #',
                    '#      #'])
N= letter_to_array(['#      #',
                    '##     #',
                    '# #    #',
                    '#  #   #',
                    '#   #  #',
                    '#    # #',
                    '#      #',
                    '#      #'])
O= letter_to_array(['########',
                    '#      #',
                    '#      #',
                    '#      #',
                    '#      #',
                    '#      #',
                    '#      #',
                    '########'])
P= letter_to_array(['########',
                    '#      #',
                    '#      #',
                    '########',
                    '#       ',
                    '#       ',
                    '#       ',
                    '#       '])
Q= letter_to_array(['########',
                    '##     #',
                    '# #    #',
                    '#  #   #',
                    '#   #  #',
                    '#    # #',
                    '#     ##',
                    '########'])
R= letter_to_array(['########',
                    '#    ## ',
                    '# ##    ',
```

art1.py

```
                    '# #    #',
                    '#  #   #',
                    '#   #  #',
                    '#    # #',
                    '#     ##',
                    '########'])
R= letter_to_array(['########',
                    '#    ## ',
                    '# ##    ',
                    '###     ',
                    '# ##    ',
                    '#  ##   ',
                    '#    ## ',
                    '#     ##'])
S= letter_to_array(['########',
                    '#       ',
                    '#       ',
                    '########',
                    '       #',
                    '       #',
                    '       #',
                    '########'])
T= letter_to_array(['########',
                    '   #    ',
                    '   #    ',
                    '   #    ',
                    '   #    ',
                    '   #    ',
                    '   #    ',
                    '   #    '])
```

```
samples=[A1,B1,C1,D1,E1,F1,G,H,I,J,K,L,M,N,O,P,Q,R,S,T]
rho_value = 0.95
network = ART( 8*8, 64, rho = rho_value)
print("rho value is --> ",rho_value)
for i in range(len(samples)):
    Z, k = network.learn(samples[i].ravel())
    if k is None:
      l= 'None'
      print("Input character %c"%(ord('A')+i),"-> class recognized ",l)

    else:
      l= chr(65+int(k))
      print("Input character %c"%(ord('A')+i),"-> class recognized ",l)
      print_letter(Z.reshape(8,8))
```

Vigilance value

Setting 8X8 grid, 64 nodes (8X8) in F1 layer, rho value

Print pattern:

If pattern fails to recognize

Successful pattern recognition

In conclusion, when the vigilance parameter is too small (0.3), very general memories were being formed and the letter was not determined correctly. By increasing to 0.95, we see the pattern learned in the F1 layer is in fact the expected letters and nodes in the F2 layer selected are most likely the correct letter recognized. The only drawback is that it takes a bit longer to learn with a higher vigilance value than with a smaller one.