شبکه‌های عصبی و یادگیری عمیق
تمرین شماره 3

محمد هاشمی
810197423
md.hashemi@ut.ac.ir

چکیده – در این تمرین سعی بر پیاده‌سازی شبکه *Hebbian,BAM,Hopfield* می‌باشد. سپس با افزودن *Disturbance* می‌خواهیم درستی شبکه را تست کنیم.

کلید واژه- Hebbian,Hopefield,BAM

1- پیاده‌سازی شبکه هب جهت تشخیص پترن‌های 5 در 3 به جای 7 در 7

در این تمرین شبکه هب را با استفاده از روابط ماتریسی در پایتون پیاده می‌کنیم. در ابتدا داده‌های مساله را به فرم ماتریس-های 49 تایی و خروجی‌ها را به صورت ماتریس‌های 15 تایی درمی‌آوریم. فرم ماتریسی خروجی‌ها در شکل 1.1 آمده است.

```
SA = np.array([[-1,-1,-1,1,-1,-1,-1,-1,-1,1,-1,1,-1,-1,-1,-1,1,1,1,-1,-1,-1,1,-1,-1,-1,1,-1,-1,1,-1,-1,-1,
                1,-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,-1,-1,-1,1]])
SB = np.array([[1,1,1,1,1,1,-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,-1,-1,1,-1,1,1,1,1,1,-1,-1,1,
                -1,-1,-1,-1,1,-1,1,-1,-1,-1,-1,-1,1,1,1,1,1,1,1,-1]])
SC = np.array([[-1,-1,-1,1,1,1,1,-1,1,1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,
                -1,1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,1,-1,-1,-1,-1,-1,-1,-1,1,1,1,1]])

TA = np.array([[-1,1,-1,1,-1,1,1,1,1,1,-1,1,1,-1,1]])
TB = np.array([[1,1,-1,1,-1,1,1,1,-1,1,-1,1,1,1,-1]])
TC = np.array([[-1,1,1,1,-1,-1,1,-1,-1,1,-1,-1,-1,1,1]])


WA = np.multiply(SA,TA.T)
WB = np.multiply(SB,TB.T)
WC = np.multiply(SC,TC.T)
W = np.add(WA,WB,WC)
```

شکل 1.1: ماترسی ورودی‌ها و خروجی‌ها

شپس با ترانهاده گرفتن از ماترس‌های S,T با توجه به رابطه 1.1 به تشکیل ماتریس خروجی می‌پردازیم.

$$W = \sum_{p=1}^{P} s(p)t(p)^T$$

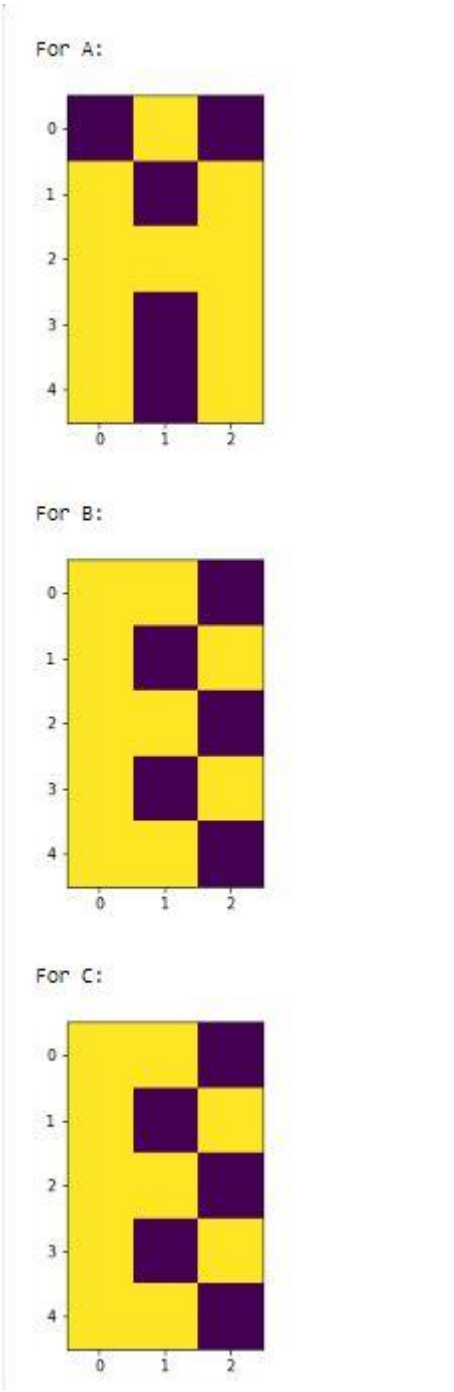
رابطه 1.1

خروجی ماترس W در شکل 1.2 موجود است.

```
array([[ 2,  2,  2,  0,  2,  2,  0,  2,  0, -2,  0, -2,  0,  2,  2,  0,
        -2, -2, -2,  2,  0,  2,  0,  2,  2,  2, -2,  0,  2, -2,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,  2,  2,  2,  2,
        -2],
       [ 0,  0,  0,  2,  0,  0, -2,  0, -2,  0, -2,  0, -2,  0,  0, -2,
         0,  0,  0,  0, -2,  0,  2,  0,  0,  0,  0, -2,  0,  0, -2, -2,
        -2,  2, -2,  2, -2, -2, -2, -2, -2,  2,  2,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0, -2,  0,  0,  2,  0,  2,  0,  2,  0,  2,  0,  0,  2,
         0,  0,  0,  0,  2,  0, -2,  0,  0,  0,  0,  2,  0,  0,  2,  2,
         2, -2,  2, -2,  2,  2,  2,  2,  2, -2, -2,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  2,  0,  0, -2,  0, -2,  0, -2,  0, -2,  0,  0, -2,
         0,  0,  0,  0, -2,  0,  2,  0,  0,  0,  0, -2,  0,  0, -2, -2,
        -2,  2, -2,  2, -2, -2, -2, -2, -2,  2,  2,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0, -2,  0,  0,  2,  0,  2,  0,  2,  0,  2,  0,  0,  2,
         0,  0,  0,  0,  2,  0, -2,  0,  0,  0,  0,  2,  0,  0,  2,  2,
         2, -2,  2, -2,  2,  2,  2,  2,  2, -2, -2,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  2,  0,  0, -2,  0, -2,  0, -2,  0, -2,  0,  0, -2,
         0,  0,  0,  0, -2,  0,  2,  0,  0,  0,  0, -2,  0,  0, -2, -2,
        -2,  2, -2,  2, -2, -2, -2, -2, -2,  2,  2,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  2,  0,  0, -2,  0, -2,  0, -2,  0, -2,  0,  0, -2,
         0,  0,  0,  0, -2,  0,  2,  0,  0,  0,  0, -2,  0,  0, -2, -2,
        -2,  2, -2,  2, -2, -2, -2, -2, -2,  2,  2,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  2,  0,  0, -2,  0, -2,  0, -2,  0, -2,  0,  0, -2,
         0,  0,  0,  0, -2,  0,  2,  0,  0,  0,  0, -2,  0,  0, -2, -2,
        -2,  2, -2,  2, -2, -2, -2, -2, -2,  2,  2,  0,  0,  0,  0,  0,
         0],
       [-2, -2, -2,  0, -2, -2,  0, -2,  0,  2,  0,  2,  0, -2, -2,  0,
         2,  2,  2, -2,  0, -2,  0, -2, -2, -2,  2,  0, -2,  2,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -2, -2, -2, -2, -2,
         2],
       [ 0,  0,  0,  2,  0,  0, -2,  0, -2,  0, -2,  0, -2,  0,  0, -2,
         0,  0,  0,  0, -2,  0,  2,  0,  0,  0,  0, -2,  0,  0, -2, -2,
        -2,  2, -2,  2, -2, -2, -2, -2, -2,  2,  2,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0, -2,  0,  0,  2,  0,  2,  0,  2,  0,  2,  0,  0,  2,
         0,  0,  0,  0,  2,  0, -2,  0,  0,  0,  0,  2,  0,  0,  2,  2,
         2, -2,  2, -2,  2,  2,  2,  2,  2, -2, -2,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  2,  0,  0, -2,  0, -2,  0, -2,  0, -2,  0,  0, -2,
         0,  0,  0,  0, -2,  0,  2,  0,  0,  0,  0, -2,  0,  0, -2, -2,
        -2,  2, -2,  2, -2, -2, -2, -2, -2,  2,  2,  0,  0,  0,  0,  0,
         0],
       [ 0,  0,  0,  2,  0,  0, -2,  0, -2,  0, -2,  0, -2,  0,  0, -2,
         0,  0,  0,  0, -2,  0,  2,  0,  0,  0,  0, -2,  0,  0, -2, -2,
        -2,  2, -2,  2, -2, -2, -2, -2, -2,  2,  2,  0,  0,  0,  0,  0,
         0],
       [ 2,  2,  2,  0,  2,  2,  0,  2,  0, -2,  0, -2,  0,  2,  2,  0,
        -2, -2, -2,  2,  0,  2,  0,  2,  2,  2, -2,  0,  2, -2,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,  2,  2,  2,  2,
        -2],
       [-2, -2, -2,  0, -2, -2,  0, -2,  0,  2,  0,  2,  0, -2, -2,  0,
         2,  2,  2, -2,  0, -2,  0, -2, -2, -2,  2,  0, -2,  2,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -2, -2, -2, -2, -2,
         2]])
```

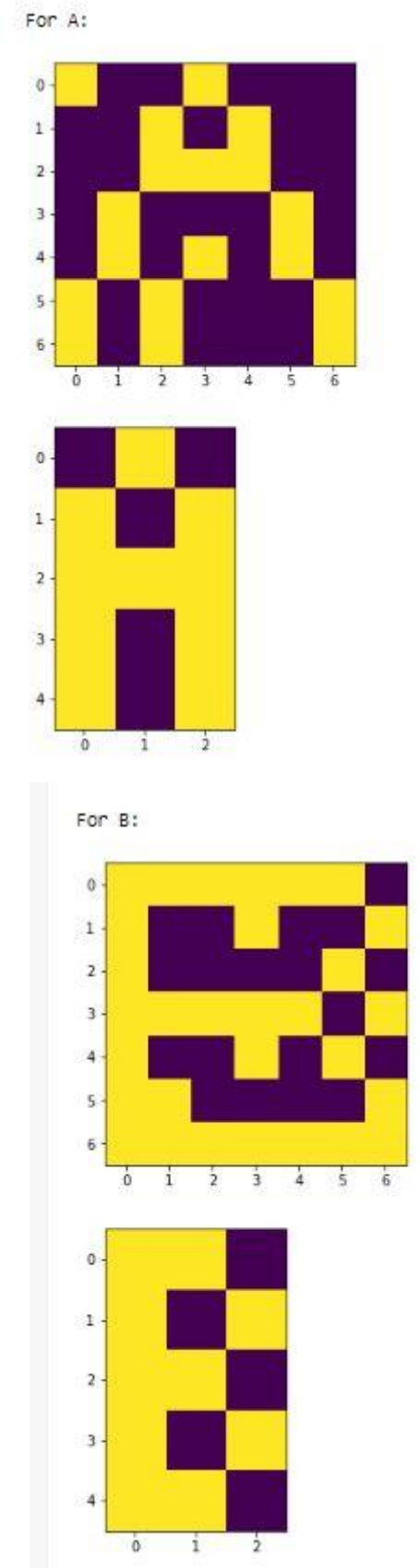
شکل 1.2 : ماتریس وزن‌های شبکه

با پک کردن پترن‌ها به عنوان ورودی و مشاهده خروجی موجود در شکل 1.3 در میابیم که شبکه ما برای هر سه ورودی دست کار نمی‌کند. علت این امر کوچک بودن فضای ورودی است. در صورتی که فضای ورودی به 8*8 تغییر یابد این مساله حل می‌شود.

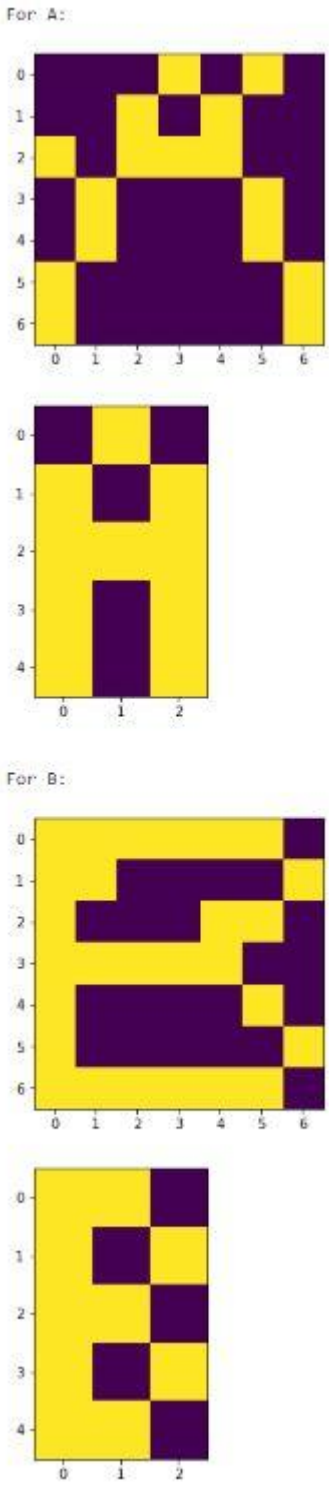


شکل 1.3: خروجی به نسبت پترن‌های مربوط به هر کلاس

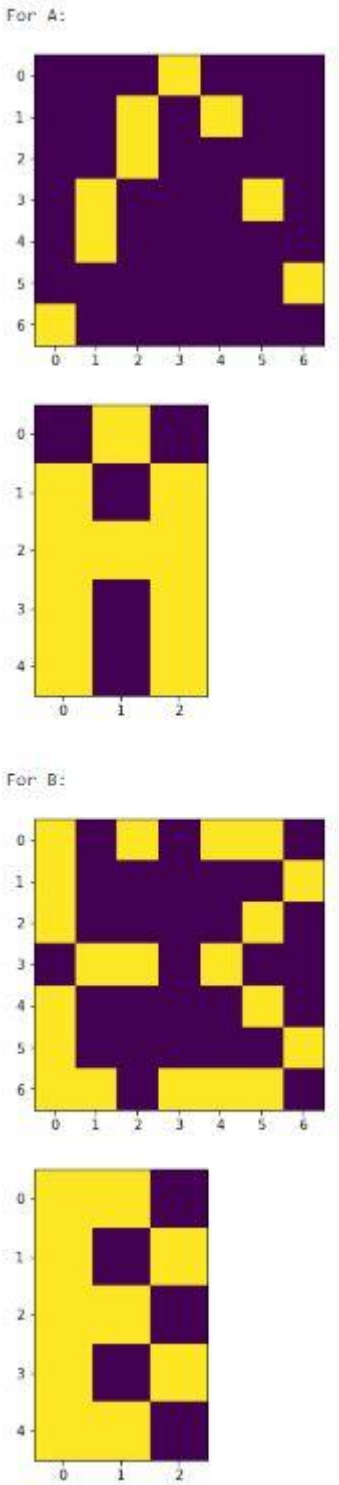
حال نوبت به ایجـاد 10 و 30 درصـد اغتشـاش و 10 و 30 درصد کاهش بر اساس مدل کتاب در ورودی‌ها و تست خروجـی آنها می‌باشد. خروجی‌ها بر اساس ورودی‌ها در شکل 1.4 موجـود است.



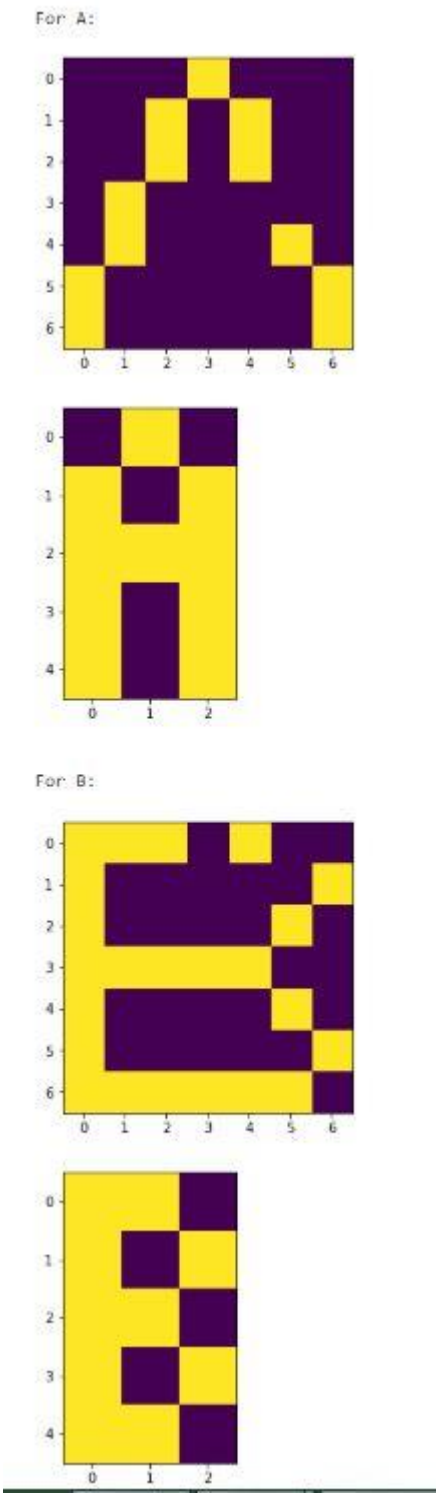
شکل 1.4: پاسخ شبکه بـه خروجـی‌هـای دارای مشـکل 30 درصد اختشاش



شکل 1.5 : پاسخ شبکه به خروجی‌های دارای مشکل 10 درصد اختشاش



شکل 1.6 : پاسخ شبکه به خروجی‌های دارای مشکل 30 درصد کاهش اطلاعات



شکل 1.7 : پاسخ شبکه به خروجی‌های دارای مشکل 10 درصد اختشاش

مشاهده می‌کنیم شبکه به شدت در برابر این موارد مقاوم است.

2- پیاده‌سازی هب خود انجمنی

در این بخش سعی داریم یک شبکه برای ذخیره یک بردار به صورت خود انجمنی توسط روش هب پیاده سازی کنیم. از این‌رو توسط رابطه 1.1 وزن‌های شبکه را بدست می‌آوریم. وزن-های شبکه در شکل 2.1 آمده است.

```
[[ 1  1  1 -1]
 [ 1  1  1 -1]
 [ 1  1  1 -1]
 [-1 -1 -1  1]]
```

شکل 2.1: وزن‌های شبکه خود انجمنی

حال بردار را یک بار به صورت یک مشکل و دومین بار به صورت دو مشکل به ترتیب از چپ به راست به شبکه می‌دهیم و خروجی شکل 2.2 را مشاهده می‌کنیم.

```
S1 = np.array([[-1,1,1,-1]])
S2 = np.array([[1,-1,1,-1]])
S3 = np.array([[1,1,-1,-1]])
S4 = np.array([[1,1,1,1]])
```

```
For S1:
[ 1  1  1 -1]

For S2:
[ 1  1  1 -1]

For S3:
[ 1  1  1 -1]

For S4:
[ 1  1  1 -1]
```

شکل 2.2: خروجی شبکه به ورودی دارای یک مشکل

شبکه برای یک اشکال کاملا درست کار می‌کنیم. حال به سراغ دو اشکال می‌رویم. ورودی و خروجی شبکه با وجود دو اشکال در شکل 2.3 آمده است.

```
S21 = np.array([[-1,-1,1,-1]])
S22 = np.array([[1,-1,-1,-1]])
S23 = np.array([[1,1,-1,1]])
S24 = np.array([[-1,1,1,1]])
```

```
For S21:
[ 1  1  1 -1]

For S22:
[ 1  1  1 -1]

For S23:
[ 1  1  1 -1]

For S24:
[ 1  1  1 -1]
```

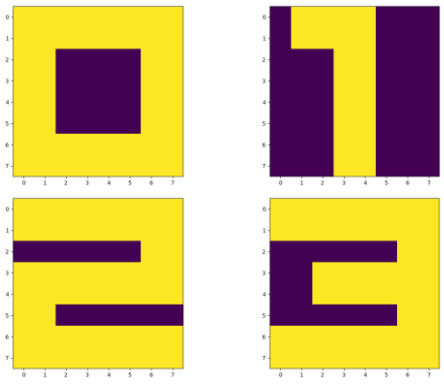
شکل 2.3: خروجی شبکه به ورودی شامل 2 اشکال

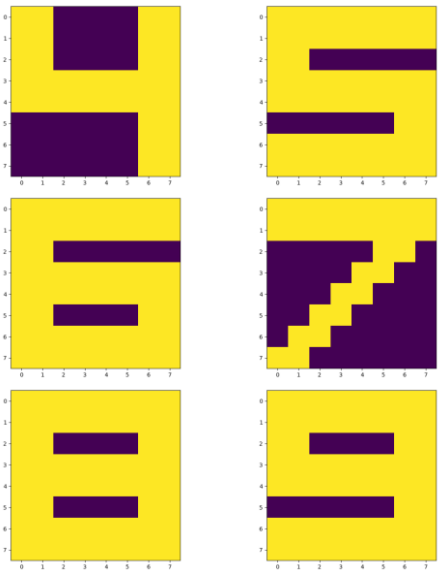
نتیجه می‌گیریم شبکه بسیار در مقابل اشکال‌های یک و دو عدد مقاوم است زیرا برای تمام حالت‌ها جواب درستی به ما می‌دهد.

3- پیاده‌سازی شبکه هاپفیلد

در این بخش از سوال می‌خواهیم با یک شبکه Hopfield اعداد 0 الی 9 را در حافظه ذخیره کنیم. در قدم اول اعداد مشخص شده به صورت زیر را

به صورت قطبی و دیجیتال کد می‌کنیم :

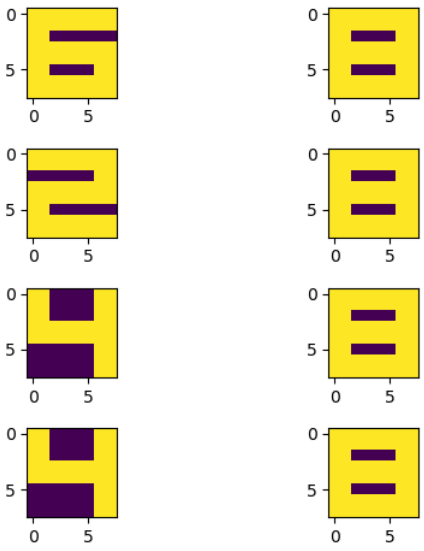




سپس اعداد را به شبکه اعمال می‌کنیم و شبکه را آموزش می‌دهیم.

وزن‌های آموزش دیده شده این شبکه در فایل Weight.txt در پوشه Codes\3_Hopfield موجود است که به دلیل حجم زیاد نمی‌توانستیم در اینجا نشان دهیم.

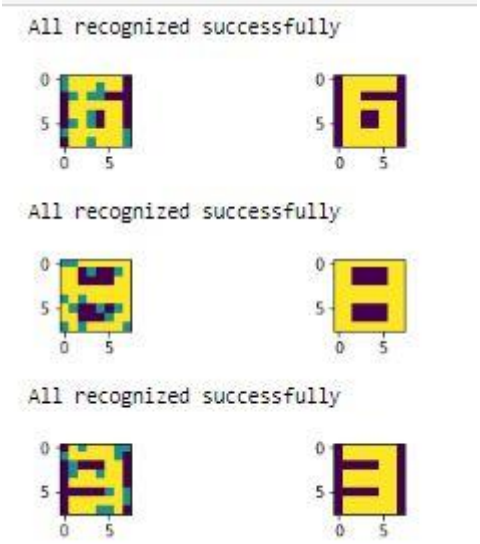
پس از آموزش نکته‌ای که به آن برخورد می‌کنیم شبکه نتوانسته تمام ورودی‌های را به حافظه بسپارد و بازگردانی کند. یعنی به نظر شبکه ما توانایی ذخیره هر 10 عدد با هم را ندارد. در واقع ظرفیت شبکه هاپفیلد پایین است.



با دقت به شکل بالا درمی‌یابیم که شبکه تنها با داشتن -1 و 1 نمی‌تواند همه 10 خانه‌را پیاده سازی‌کند.

در ادامه به هرکدام از داده‌های3 و 6 و 8 مقدار 20٪ نویز اضافه می‌کنیم. پس با توجه به این که در کل 64 پیکسل در نظر گرفته‌ایم باید 13 پیکسل را به صورت تصادفی انتخاب کنیم و

مقدارشان را برابر صفر قرار دهیم. با تکرارهای پیاپی مشاهده می‌شود برای اعداد 0 و 1 شبکه نسبت به نویز ایجاد شده مقاوم بوده و دو عدد را از یکدیگر به خوبی تشخیص می‌دهد.
نتایج برای ورودی 3و6و8 در شکل 3.1 موجود است.

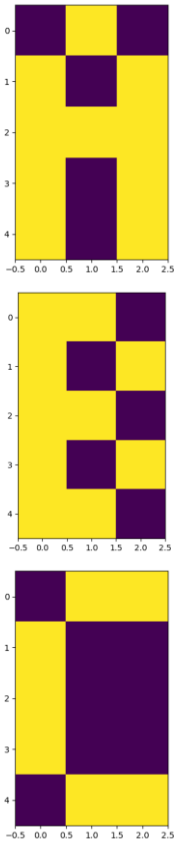


شکل 3.1: نتیجه خروجی برای ایجاد نویز در عدد 3و6و8 برای یک مرتبه تست

این روند را برای 100 بار تکرار کردیم که در همه موارد شبکه به صورت صحیح نتیجه را تشخیص داد. پس می‌توان نتیجه گرفت که این شبکه در مقابل نویز به میزان 20 درصد مقاوم است.

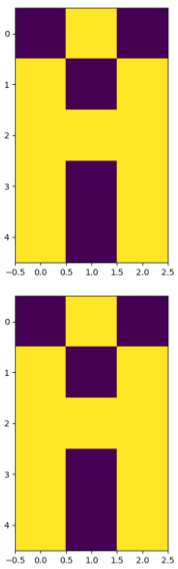
4- پیاده‌سازی شبکه BAM

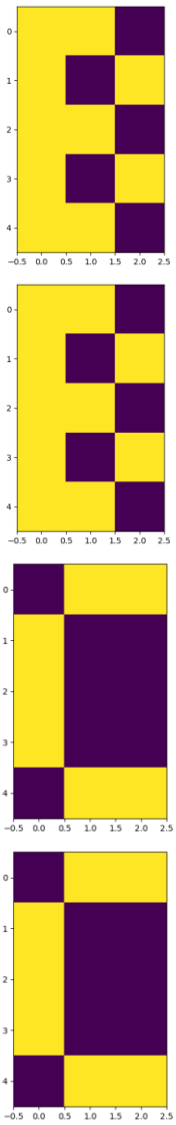
این بار می‌خواهیم با یک شبکه BAM حروف A,B,C را که به صورت زیر دیجیتال سازی شده‌اند را از هم جدا بکنیم. نتایج دیجیتال شده این سه حرف در شکل 4.1 آمده است.



شکل 4.1: نتیجه دیجیتال شده حروف A,B,C

در صورت سوال خواسته شده برای لایه Y سه نرون در نظر بگیریم. در حالت اول ورودی‌ها را بدون تغییر به شبکه وارد می‌کنیم و نتیجه زیر حاصل می‌شود :





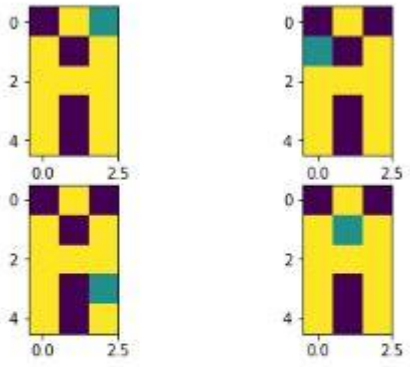
مشاهده می‌شود شبکه به خوبی توانسته مقدار اصلی را نتیجه دهد. **وزن‌های شبکه آموزش داده‌شده علاوه بر شکل4.2 در فایل Weight.txt در پوشه Codes\4_BAM موجود است.**

```
array([[ 1., -3.,  1.],
       [ 1.,  1.,  1.],
       [-3.,  1.,  1.],
       [ 1.,  1.,  1.],
       [-1., -1., -1.],
       [ 3., -1., -1.],
       [ 1.,  1.,  1.],
       [ 3., -1., -1.],
       [ 1.,  1., -3.],
       [ 1.,  1.,  1.],
       [-1., -1., -1.],
       [ 3., -1., -1.],
       [ 3., -1., -1.],
       [-1., -1.,  3.],
       [-1.,  3., -1.]])
```

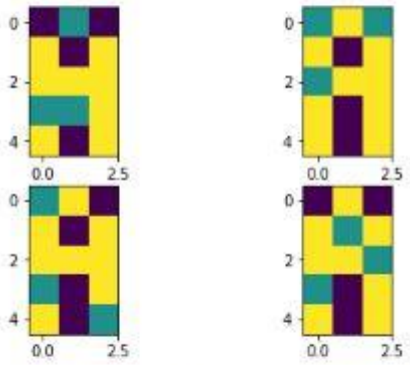
شکل 4.2 : وزن‌های آموزش داده‌شده شبکه BAM

در حالت دوم به هر کدامم از ورودی‌ها به میزان ده و بیست و سی درصد نویز وارد می‌کنیم که در نتیجه باید به ترتیب یک و سه و پنج پیکسل از 15 پیکسل تصویر اصلی را مقدار صفر قرار بدهیم) و ورودی ها به صورت زیر به دست می‌آیند : (4 نمونه از هرکدام از تست‌ها رسم شده‌اند) شکل 4.3 نتیجه تشخیص شبکه پس از ایجاد نویز در ورودی‌ها می‌باشد.

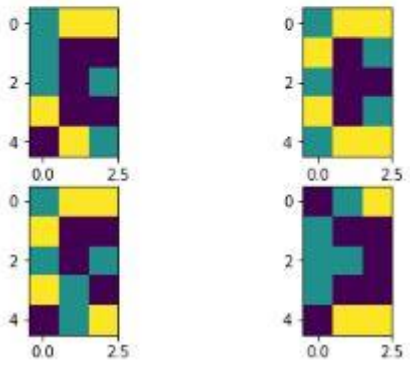
```
All patterns recognized successfully
```

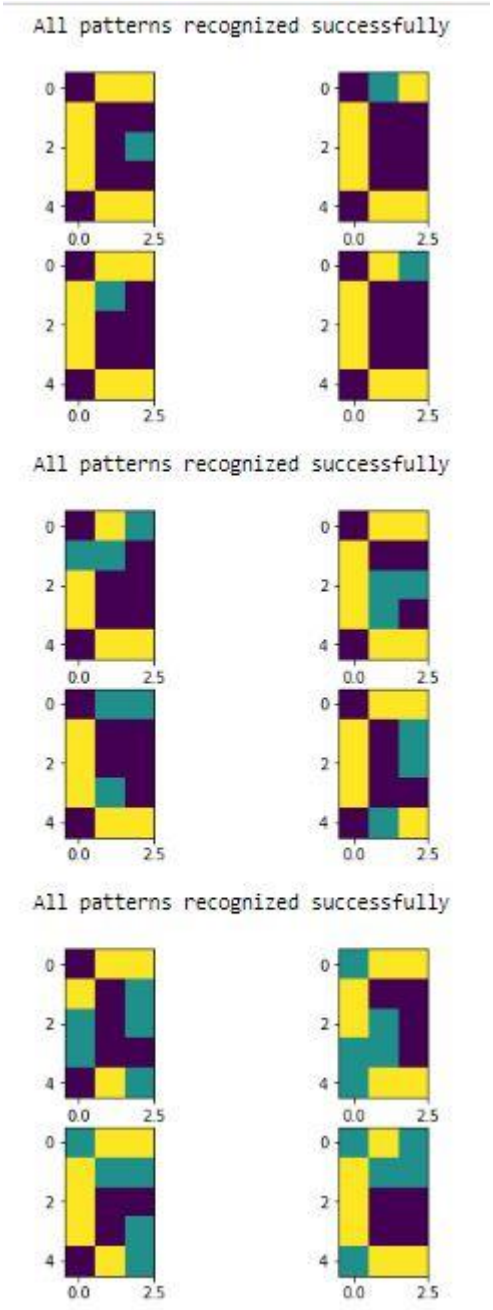


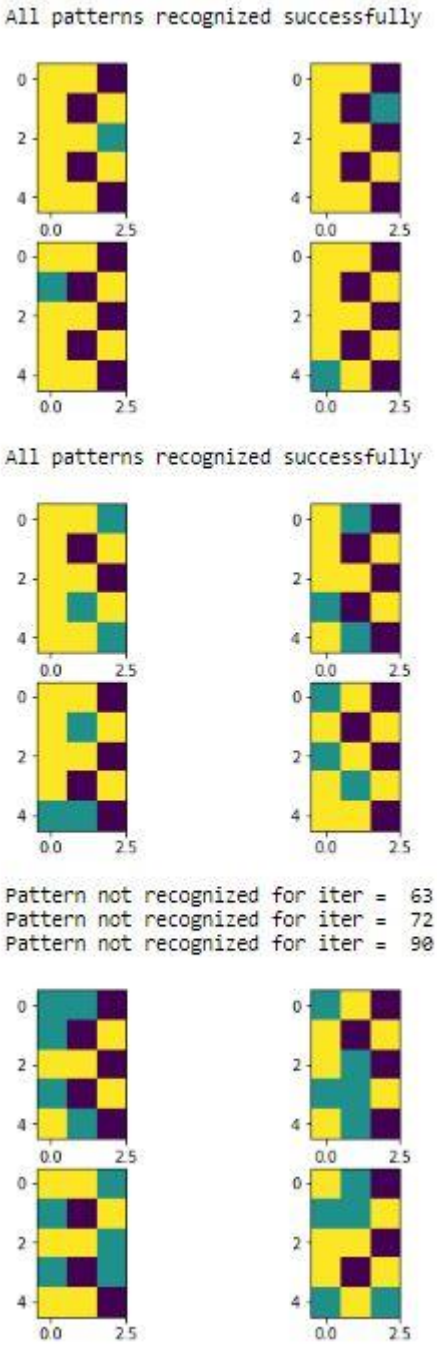
```
All patterns recognized successfully
```



```
All patterns recognized successfully
```



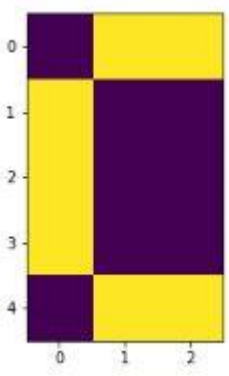




شکل 4.3: نتیجه تشخیص شبکه با وجود درصدهای مختلف نویز(به ترتیب از 10و 20 و 30 درصد برای ورودی A تا آخر)

با توجه به نتایج بالا درمی‌یابیم که شبکه برای ورودی‌های A,C حتی در صورت وجود 30 درصد نویز به درستی‌کار کرده‌است. اما برای ورودی B فقط در حالت نویز 30 درصد آن‌هم تنها در 3 حالت خاص از 100 حالت دچار مشکل شده است.

در ادامه بردار [-1,1,0] را به ورودی می‌دهیم و مشاهده میکنیم که شبکه شکل C را تشخیص داده‌است.



شکل تشخیص داده شده از طرف شبکه برای بردار [-1,1,0]

در نهایت به بررسی فاصله همینگ می‌پردازیم. مشاهده می-کنیم که بیشترین فاصله همینگ این شبکه با توجه به شکل 4.4 برابر 2 می‌باشد.

```
[[0. 2. 2.]
 [2. 0. 2.]
 [2. 2. 0.]]
```

شکل 4.4: ماتریس اختلاف همینگ بین ورودی‌های شبکه

مراجع

[1] neupy.com