به نام خدا

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

تمرین سری سوم

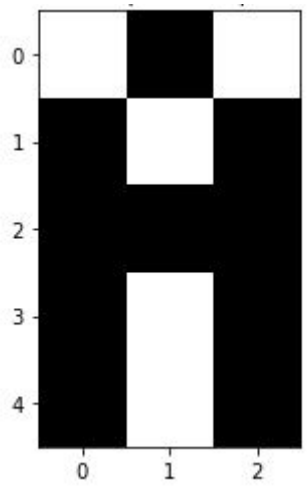
نام و نام خانوادگی	فاطمه حقیقی
شماره دانشجویی	810195385
تاریخ ارسال گزارش	۲۳ اردیبهشت ۱۳۹۹

فهرست گزارش سوالات

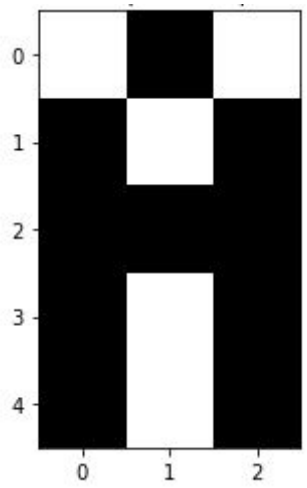
سوال 1 – Character Recognition using Hebbian Learning Rule

قسمت اول)

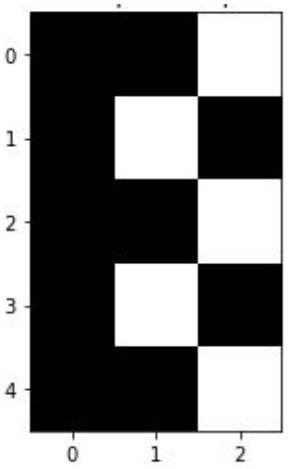
همانطور که در صورت سوال خواسته شده، شبکه ای طراحی کردیم که با گرفتن ورودی های ۷*۹ بتواند خروجی ۳*۵ را به ما به عنوان خروجی بدهد. همانطور که در شکل های زیر نشان داده شده، شبکه طراحی شده قادر خواهد بود تمامی ورودی های داده شده را به خروجی مطلوب برساند.



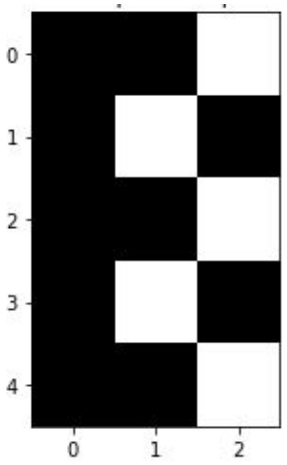
شکل ۱. خروجی مورد انتظار شبکه به ازای ورودی اول



شکل ۲. خروجی شبکه به ازای ورودی اول

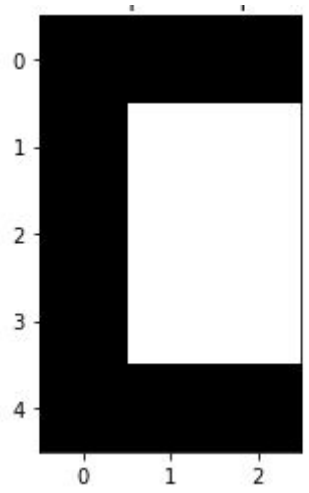


شکل۳. خروجی مورد انتظار شبکه به ازای ورودی دوم



شکل۴. خروجی شبکه به ازای ورودی دوم

شکل۵. خروجی مورد انتظار شبکه به ازای ورودی سوم



شکل۶. خروجی شبکه به ازای ورودی سوم

قسمت دوم)

در این قسمت بر روی الگو، نویز اعمال کردیم. در نتیجه میزان دقت شبکه برای

epoch = 1000 و noise = 20% برابر است با:

accuracy of network is: 0.928

و میزان دقت شبکه برای epoch = 1000 و noise = 40% برابر است با :

accuracy of network is: 0.919

قسمت سوم)

در این قسمت با توجه به درصد داده شده، اطلاعاتی از الگوها را به صورت رندم از بین بردیم. در نتیجه میزان دقت شبکه برای epoch = 1000 و missing percentage = 20% برابر است با:

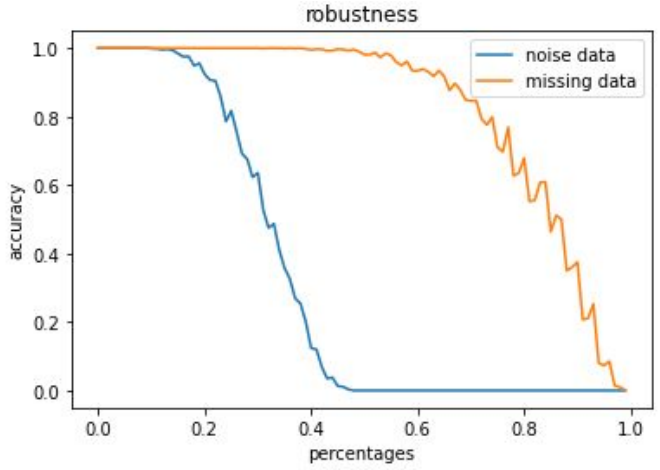
accuracy of network is: 1.0

و میزان دقت شبکه برای epoch = 1000 و missing percentage = 40% برابر است با :

accuracy of network is: 0.999

قسمت چهارم)

برای پاسخ به این قسمت، به ازای درصدهای مختلف، شبکه را به اندازه ی epoch = 1000 آموزش دادیم. به عبارت دیگر به ازای یک درصد، شبکه را همراه با نویز به آن اندازه و به طور جداگانه همراه با از بین رفتن اطلاعات به آن اندازه آموزش دادیم و دقت آن ها را بدست آوردیم. نمودار حاصل از این فرآیند به صورت زیر می باشد:



شکل۷. مقایسه ی دقت شبکه به ازای درصد های متفاوت نویز و از بین رفتن اطلاعات

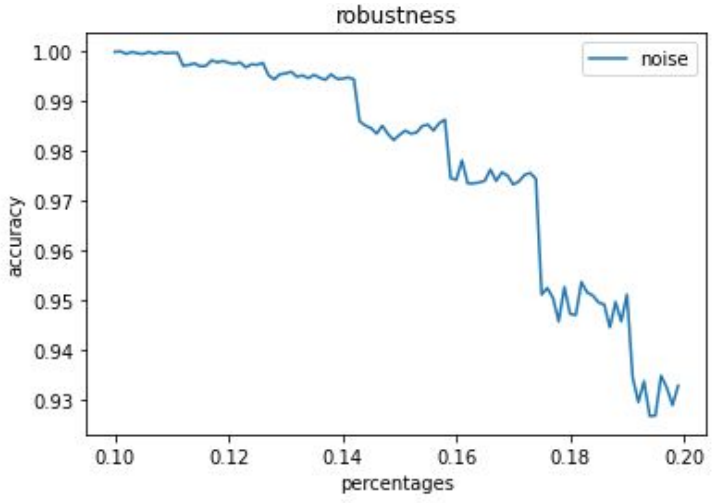
همانطور که در شکل ۷ دیده می شود، مقاوت شبکه در حالت از بین رفتن اطلاعات بیشتر از شبکه با الگوهای نویزی می باشد.

همچنین با توجه به جواب بدست آمده در قسمت های دوم و سوم این سوال و دقت های بدست آمده نیز می توان گفت، مقاوت شبکه در حالت از بین رفتن اطلاعات بیشتر است.

قسمت پنجم)

برای پاسخ این قسمت نیز، به ازای درصدهای متفاوتی از نویز شبکه را به اندازه

epoch = 1000 آموزش دادیم. در نتیجه آن نمودار زیر بدست آمد:



شکل۸. دقت شبکه به ازای درصد های متفاوت نویز

همانطور که در نمودار شکل ۸ مشاهده می شود، حداکثر مقاومت شبکه در برابر نویز ۱۰ درصد می باشد.

سوال ۲ – Storage Capacity in an Auto-associatiove Net

قسمت اول)

در این قسمت برای ذخیره ی بردار S= [1, 1, 1, -1] در شبکه ابتدا ماتریس وزن آن را ساختیم سپس آن را به عنوان ورودی به شبکه دادیم، و شبکه به عنوان خروجی آن را برای ما تداعی کرد. بنابراین توانستیم این بردار را در شبکه ذخیره کنیم.

```
original pattern is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
predicted pattern is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل۹. بردار اصلی و بردار تداعی شده توسط شبکه

قسمت دوم)

در این قسمت بردار a = [-1, 1, 1, 1] را برای ذخیره به شبکه به عنوان ورودی دادیم. به دلیل آنکه بردار های s , a بر هم متعامد هستند، شبکه قادر به بازیابی این بردار خواهد بود.

```
original pattern for a is:
[[-1]
 [ 1]
 [ 1]
 [ 1]]
predicted pattern for a is:
[[-1]
 [ 1]
 [ 1]
 [ 1]]
```

شکل۱۰. بردار اصلی و بردار تداعی شده توسط شبکه

قسمت سوم)

برای پاسخ دادن به این قسمت، فرض می کنیم شبکه ای داریم که می تواند n بردار را تداعی کند به عبارت دیگر در این شبکه n بردار ذخیره شده است. و می خواهیم یک بردار جدید که ترکیب خطی از n بردار داخل شبکه است به شبکه بدهیم تا در آن ذخیره شود. همچنین فرض می کنیم n بردار داخل شبکه همگی بر هم متعامد هستند. و نیز شبکه ما دارای m بعد می باشد.

$$W = \sum_{i=1}^{n} s_i(p)\, s_i(p)^T - nI \quad (1)$$

$$A = \sum_{i=1}^{n} \alpha_i s_i(p) \rightarrow A^T = \sum_{i=1}^{n} \alpha_i s_i(p)^T \quad (2)$$

$$A^T W = \left(\sum_{i=1}^{n} \alpha_i s_i(p)^T\right)\left(\sum_{j=1}^{n} s_j(p)\, s_j(p)^T\right) - nA^T \quad (3)$$

$$\Rightarrow A^T W = \sum_{i=1}^{n} \alpha_i s_i(p)^T s_i(p)\, s_i(p)^T - nA^T \quad (4)$$

$$\Rightarrow A^T W = \alpha_i \|s_i(p)\|^2 s_i(p)^T - \sum_{i=1}^{n} n s_i(p)^T \alpha_i \quad (5)$$

$$\Rightarrow A^T W = \sum_{i=1}^{n} \alpha_i m\, s_i(p)^T - \sum_{i=1}^{n} \alpha_i n\, s_i(p)^T \quad (6)$$

$$\Rightarrow A^T W = \sum_{i=1}^{n} \alpha_i (m-n)\, s_i(p)^T$$

در این شبکه ماتریس وزن طبق رابطه ی اول بدست می آید. و نیز فرض کردیم بردار جدید به فرم رابطه ی ۲ می باشد. بنابراین بردار جدید A را به شبکه می دهیم تا خروجی شبکه را به ازای آن بدست آوریم. چون بردارهای داخل شبکه متعامد هستند. از رابطه ی ۳ به رابطه ی ۴ می رسیم. همچنین با توجه به آنکه فرض کردیم شبکه دارای m بعد می باشد، می توانیم از رابطه ی ۵ به رابطه ی ۶ برسیم.

در نهایت برای آنکه رابطه ی بدست آمده در آخر از activation function عبور کند و خود بردار یا یکی از بردارهای داخل شبکه تداعی شود ، لازم است $\alpha_i(m-n)$ مثبت باشد همچنین لازم است mاز n بزرگتر باشد. با برقراری این دو شرط می توان بردار جدید به شبکه اضافه کرد.

قسمت چهارم)

در صورتی که بعد شبکه ما ۴ باشد یعنی ۴ نورون در شبکه داشته باشیم. تا سه بردار را می توانیم در این شبکه ذخیره و به درستی تداعی کنیم.

قسمت پنجم)

در صورتی که شبکه ما n بعد داشته باشد، n-1 بردار را می توانیم در آن ذخیره و بازیابی کنیم.

سوال 3 – Iterative Auto-associative Net

قسمت اول)

در این قسمت برای ذخیره ی بردار S= [1, 1, 1, -1] در شبکه ابتدا ماتریس وزن آن را ساختیم سپس آن را به عنوان ورودی به شبکه دادیم، و شبکه به عنوان خروجی آن را برای ما تداعی کرد. بنابراین توانستیم این بردار را در شبکه ذخیره کنیم.

```
original pattern s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
predicted pattern for s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل ۱۱. بردار اصلی و بردار تداعی شده توسط شبکه

قسمت دوم)

در این قسمت برای از بین بردن اطلاعات بردار s چهار حالت داریم که در هر ۴ حالت، بردارها به درستی توسط شبکه بازیابی می شوند و مقدار اصلی را به عنوان خروجی به ما می دهند.

حالت اول - ذخیره ی بردار s1 = [0,0,0,-1]:

```
original pattern s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
disturbed pattern for s is:
[[ 0]
 [ 0]
 [ 0]
 [-1]]
predicted output pattern for s is:
[[1]
 [1]
 [1]
 [1]]
predicted input pattern for s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل۱۲. بردار اصلی، بردار تغییر داده شده و بردار تداعی شده توسط شبکه

حالت دوم- ذخیره ی بردار s2 = [1, 0, 0, 0]:

```
original pattern s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
disturbed pattern for s is:
[[1]
 [0]
 [0]
 [0]]
predicted output pattern for s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل۱۳. بردار اصلی ، بردار تغییر داده شده و بردار تداعی شده توسط شبکه

حالت سوم- ذخیره ی بردار [0,1,0,0] = s3:

```
original pattern s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
disturbed pattern for s is:
[[0]
 [1]
 [0]
 [0]]
predicted output pattern for s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل۱۴. بردار اصلی ، بردار تغییر داده شده و بردار تداعی شده توسط شبکه

حالت چهارم- ذخیره ی بردار [0,0,1,0]=s4:

```
original pattern s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
disturbed pattern for s is:
[[0]
 [0]
 [1]
 [0]]
predicted output pattern for s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل۱۵. بردار اصلی ، بردار تغییر داده شده و بردار تداعی شده توسط شبکه

قسمت سوم)

در این قسمت، ۴ حالت برای ایجاد نویز بر روی بردار داریم، که در تمامی این حالات شبکه به یک بردار همگرا می شود که جواب موجود در شبکه نیست در واقع به بردار نادرستی همگرا می شود.

حالت اول - ذخیره ی بردار s1 = [-1, -1, -1, -1] :

```
original pattern s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
disturbed pattern for s is:
[[-1]
 [-1]
 [-1]
 [-1]]
predicted output pattern for s is:
[[-1]
 [-1]
 [-1]
 [ 1]]
predicted input pattern for s is:
[[-1]
 [-1]
 [-1]
 [ 1]]
```

شکل۱۶. بردار اصلی ، بردار تغییر داده شده و بردار تداعی شده توسط شبکه

حالت دوم- ذخیره ی بردار s2 = [1, -1, -1, 1] :

```
original pattern s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
disturbed pattern for s is:
[[ 1]
 [-1]
 [-1]
 [ 1]]
predicted output pattern for s is:
[[-1]
 [-1]
 [-1]
 [ 1]]
predicted input pattern for s is:
[[-1]
 [-1]
 [-1]
 [ 1]]
```

شکل۱۷. بردار اصلی ، بردار تغییر داده شده و بردار تداعی شده توسط شبکه

حالت سوم- ذخیره ی بردار [1 ,-1 ,1 ,-1] = s3:

```
original pattern s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
disturbed pattern for s is:
[[-1]
 [ 1]
 [-1]
 [ 1]]
predicted output pattern for s is:
[[-1]
 [-1]
 [-1]
 [ 1]]
predicted input pattern for s is:
[[-1]
 [-1]
 [-1]
 [ 1]]
```

شکل۱۸. بردار اصلی ، بردار تغییر داده شده و بردار تداعی شده توسط شبکه

حالت چهارم- ذخیره ی بردار [1 ,1 ,-1 ,-1]=s4:

```
original pattern s is:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
disturbed pattern for s is:
[[-1]
 [-1]
 [ 1]
 [ 1]]
predicted output pattern for s is:
[[-1]
 [-1]
 [-1]
 [ 1]]
predicted input pattern for s is:
[[-1]
 [-1]
 [-1]
 [ 1]]
```

شکل۱۹. بردار اصلی ، بردار تغییر داده شده و بردار تداعی شده توسط شبکه

قسمت چهارم)

ایجاد missing در شبکه: در این قسمت برای از بین بردن اطلاعات بردار s چهار حالت داریم که در هر ۴ حالت، بردارها به درستی توسط شبکه بازیابی می شوند و مقدار اصلی را به عنوان خروجی به ما می دهند.

حالت اول - ذخیره ی بردار [1-,0,0,0] = s1:

```
index list is:  [0, 3, 1, 2]
output in iteration:
[[ 1]
 [ 0]
 [ 0]
 [-1]]
output in iteration:
[[ 1]
 [ 0]
 [ 0]
 [-1]]
output in iteration:
[[ 1]
 [ 1]
 [ 0]
 [-1]]
output in iteration:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل ۲۰.تغییر بردار disturb شده طی هر iteration

حالت دوم- ذخیره ی بردار [0 ,0 ,0 ,1] = s2:

```
index list is:  [1, 2, 0, 3]
output in iteration:
[[1]
 [1]
 [0]
 [0]]
output in iteration:
[[1]
 [1]
 [1]
 [0]]
output in iteration:
[[1]
 [1]
 [1]
 [0]]
output in iteration:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل ۲۱.تغییر بردار disturb شده طی هر iteration

حالت سوم- ذخیره ی بردار [0,1,0,0] = s3:

```
index list is:  [3, 0, 2, 1]
output in iteration:
[[ 0]
 [ 1]
 [ 0]
 [-1]]
output in iteration:
[[ 1]
 [ 1]
 [ 0]
 [-1]]
output in iteration:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
output in iteration:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل۲۲.تغییر بردار disturb شده طی هر iteration

حالت چهارم- ذخیره ی بردار [0,0,1,0]=s4:

```
index list is:  [1, 2, 0, 3]
output in iteration:
[[0]
 [1]
 [1]
 [0]]
output in iteration:
[[0]
 [1]
 [1]
 [0]]
output in iteration:
[[1]
 [1]
 [1]
 [0]]
output in iteration:
[[ 1]
 [ 1]
 [ 1]
 [-1]]
```

شکل۲۳. تغییر بردار disturb شده طی هر iteration

ایجاد نویز در شبکه : در این قسمت، ۴ حالت برای ایجاد نویز بر روی بردار داریم، که در تمامی این حالات شبکه به یک بردار همگرا می شود که جواب موجود در شبکه نیست در واقع به بردار نادرستی همگرا می شود.

حالت اول - ذخیره ی بردار [1- ,1- ,1- ,1-] = s1:

```
index list is:  [2, 0, 3, 1]
output in iteration:
[[-1]
 [-1]
 [-1]
 [-1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [-1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [ 1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [ 1]]
```

شکل۲۴. تغییر بردار disturb شده طی هر iteration

حالت دوم- ذخیره ی بردار [1 ,1- ,1- ,1] = s2:

```
index list is:  [3, 2, 0, 1]
output in iteration:
[[ 1]
 [-1]
 [-1]
 [ 1]]
output in iteration:
[[ 1]
 [-1]
 [-1]
 [ 1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [ 1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [ 1]]
```

شکل۲۵. تغییر بردار disturb شده طی هر iteration

حالت سوم- ذخیره ی بردار [1 ,1- ,1 ,1-] = s3:

```
index list is:  [2, 3, 1, 0]
output in iteration:
[[-1]
 [ 1]
 [-1]
 [ 1]]
output in iteration:
[[-1]
 [ 1]
 [-1]
 [ 1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [ 1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [ 1]]
```

شکل۲۶. تغییر بردار disturb شده طی هر iteration

حالت چهارم- ذخیره ی بردار [1 ,1 ,-1 ,-1]=s4:

```
index list is:  [0, 2, 3, 1]
output in iteration:
[[-1]
 [-1]
 [ 1]
 [ 1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [ 1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [ 1]]
output in iteration:
[[-1]
 [-1]
 [-1]
 [ 1]]
```

شکل۲۷. تغییر بردار disturb شده طی هر iteration

قسمت پنجم)

تعداد الگو های قابل ذخیره سازی توسط شبکه ی discrete hopfield برابر با $P = \frac{n}{2\,log_2(n)}$

می باشد.

قسمت ششم)

استفاده از شبکه ی اول یعنی شبکه ی iterative auto-associative بهتر است زیرا اسن شبکه نسبت به شبکه ی discrete hopfild مقاومت بیشتری دارد.

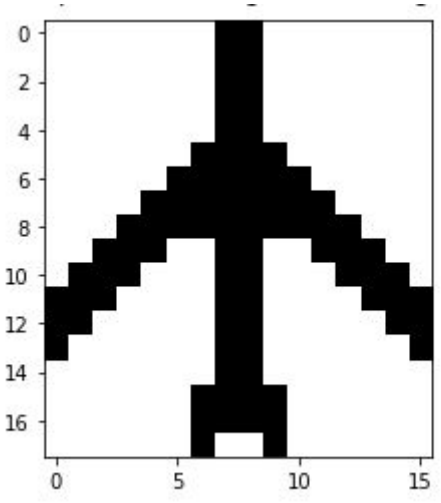
شبکه ی iterative auto-associative توانایی ذخیره ی ۳ بردار را دارد در حالی که ما ۱ بردار در آن ذخیره می کنیم در حالی که شبکه ی discrete hopfild توانایی ذخیره ی ۱ بردار را دارد و ما از حداکثر توان آن استفاده می کنیم.

سوال 4 – Recurrent Hetro-Associative Network

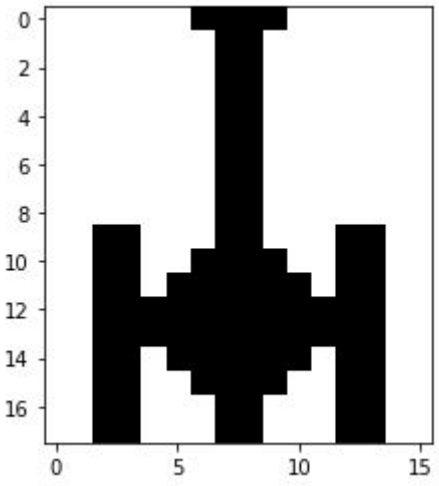
قسمت اول)

همانطور که در صورت سوال گفته شده، یک شبکه ی BAM طراحی کردیم تا بتوانیم یک mapping دوسویی ایجاد کنیم. ابتدا ماتریس شکل های داده شده در صورت پروژه را ساختیم که در آن برای نقاط مشکی عدد +۱ و برای نقاط سفید عدد -۱ را در ماتریس در نظر گرفتیم. پس از آن به کمک ورودی و خروجی دو الگوی داده شده،ماتریس hebbian آن را ساختیم. این شبکه به درستی می تواند با دادن الگوی ورودی، الگوی خروجی مد نظر را برای ما تخمین بزند. همچنین با دادن الگو های خروجی متناظر با هر ورودی، شبکه می تواند الگوهای ورودی آن ها را به درستی تخمین بزند. بنابراین می توان گفت که شبکه ی BAM دوسویی ای طراحی شده که به درستی کار می کند.

ورودی هایی که به شبکه داده می شود به صورت زیر است:

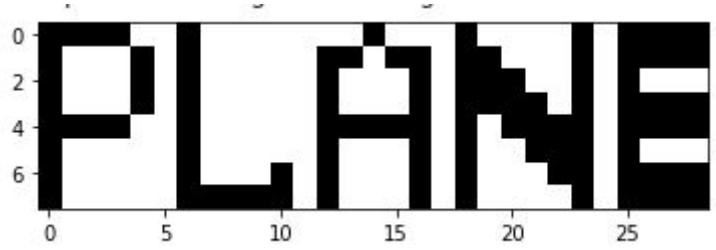


شکل۲۸. ورودی اول شبکه

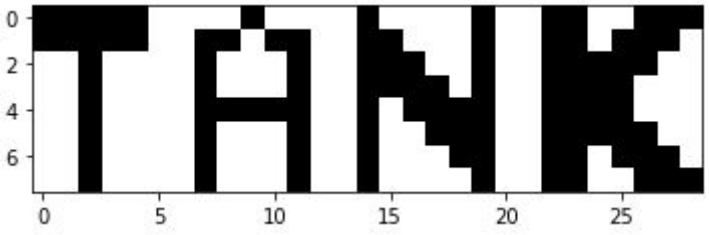


شکل۲۹. ورودی دوم شبکه

خروجی های مد نظر شبکه به صورت زیر می باشند:



شکل۳۰. خروجی مورد انتظار از شبکه به ازای ورودی اول

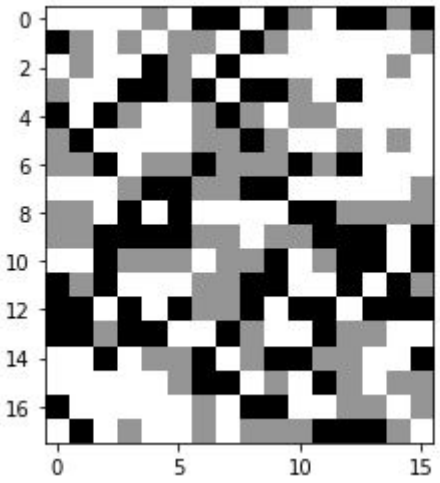


شکل۳۱. خروجی مورد انتظار از شبکه به ازای ورودی دوم

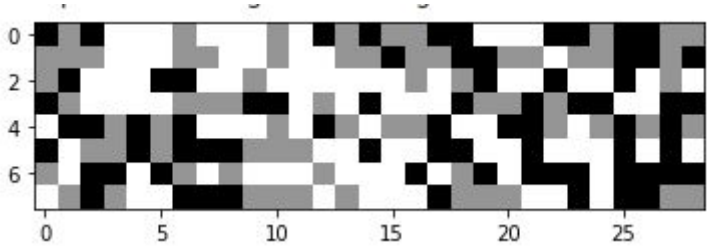
همانطور که گفته شد، شبکه دوسویی ما به درستی عمل می کند و به ازای ورودی اول(عکس۱) خروجی اول (عکس ۳) را به ما می دهد و در صورتی که شکل ۳ را به عنوان ورودی به شبکه بدهیم، شکل ۱ را به عنوان خروجی به ما می دهد. و همچنین در رابطه با ورودی و خروجی دوم نیز این مسئله صادق است.

قسمت دوم)

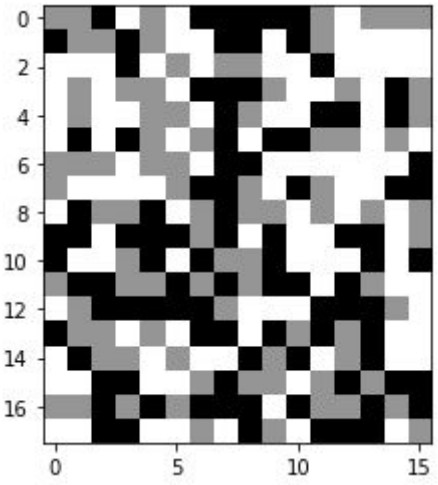
همانطور که در صورت پروژه گفته شده، به هر ماتریس ۳۰ درصد اغتشاش اعمال کردیم، بدین صورت که به هر ماتریس ۳۰ درصد نویز و ۳۰ درصد از بین رفتن اطلاعات اعمال کردیم. پس از اعمال ایت اغتشاشات، ماتریس های ورودی و خروجی به صورت زیر درآمدند:



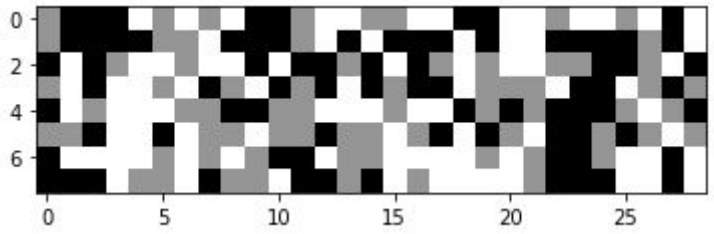
شکل۳۲. ورودی اول شبکه پس از اعمال اغتشاش



شکل۳۳. خروجی مورد انتظار از شبکه به ازای ورودی اول پس از اعمال اغتشاش

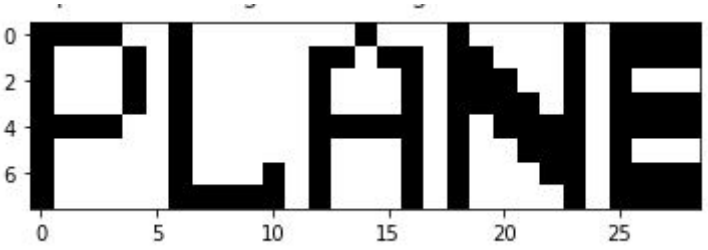


شکل۳۴. ورودی دوم شبکه پس از اعمال اغتشاش



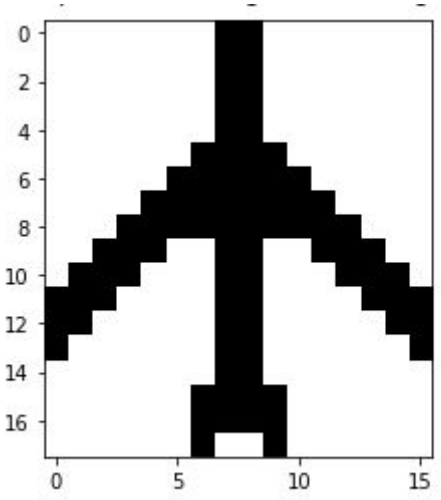
شکل۳۵. خروجی مورد انتظار از شبکه به ازای ورودی دوم پس از اعمال اغتشاش

سپس الگوهای حاصل را به شبکه دادیم. مشاهده کردیم که با اعمال شکل ۳۲ به شبکه خروجی آن به صورت زیر شد:



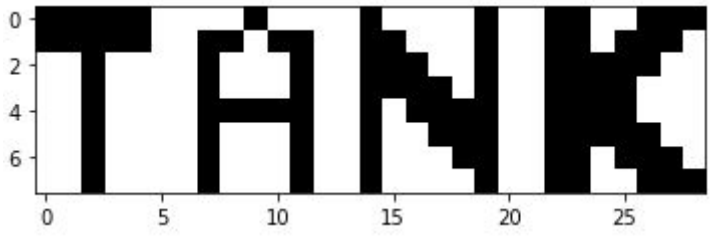
شکل۳۶. خروجی شبکه به ازای ورودی اول پس از اعمال اغتشاش

و نیز با اعمال الگوی شکل ۳۳ به شبکه ، خروجی شبکه به ثورت زیر شد:



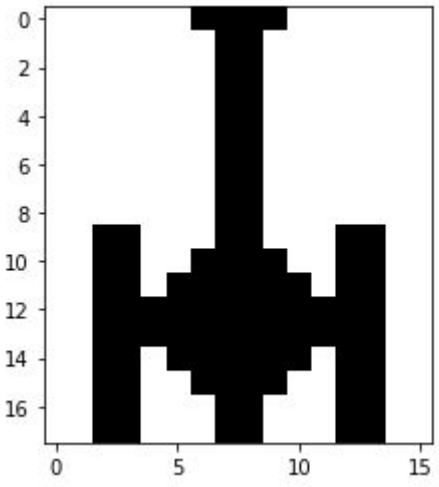
شکل۳۷. بدست آوردن ورودی اول از شبکه ازای اعمال خروجی اول پس از اعمال اغتشاش به آن

همانطور نیز با اعمال الگوی شکل ۳۴ به شبکه، شکل زیر حاصل شد:



شکل۳۹. خروجی شبکه به ازای ورودی دوم پس از اعمال اغتشاش

و نیز با اعمال شکل ۳۵ به شبکه شکل زیر حاصل شد:



شکل۳۸. بدست آوردن ورودی دوم از شبکه ازای اعمال خروجی دوم پس از اعمال اغتشاش به آن

بنابراین می توان گفت شبکه BAMدوسوسی ما پس از اعمال اغتشاش ۳۰ دصدی بر روی الگوها،می تواند سمت دیگر الگوهای اصلی را به درستی بازیابی کند.