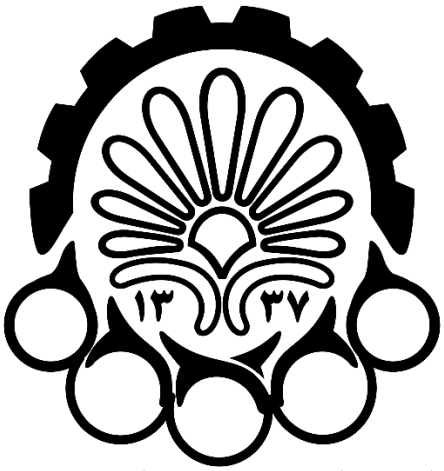


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

تمر ین سوم درس شبکه های عصبی

دکتر صفابخش

غلامرضا دار ۴۰۰۱۳۱۰۱۸

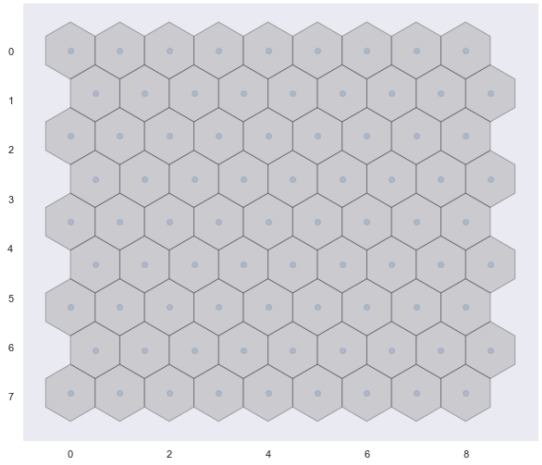
بهار ۱۴۰۱

فهرست مطالب

سوال ۱) ۳
سوال ۲) ۴
سوال ۳) ۵
سوال ۴) ۶
سوال ۵) ۷
سوال ۶) ۱۵

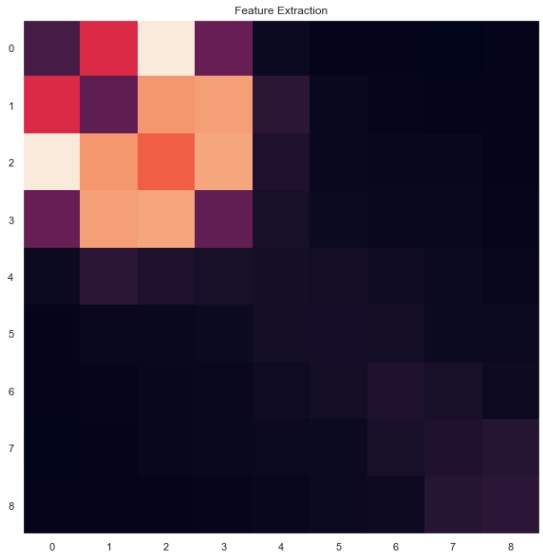
سوال ۱)

همانطور که می‌دانیم، دو کار اصلی که شبکه‌های خودسازماندهِ کوهونن انجام می‌دهند، کاهش بُعد و خوشه بندی است. فرض کنید هر بردار در مجموعه داده ورودی دارای ۵۶۱ بعد باشد (به عنوان مثال در همین تمرین) و یک نقشه ۹ در ۹ را برای این شبکه در نظر گرفته‌ایم.



شکل ۱- نقشه ۹ در ۹

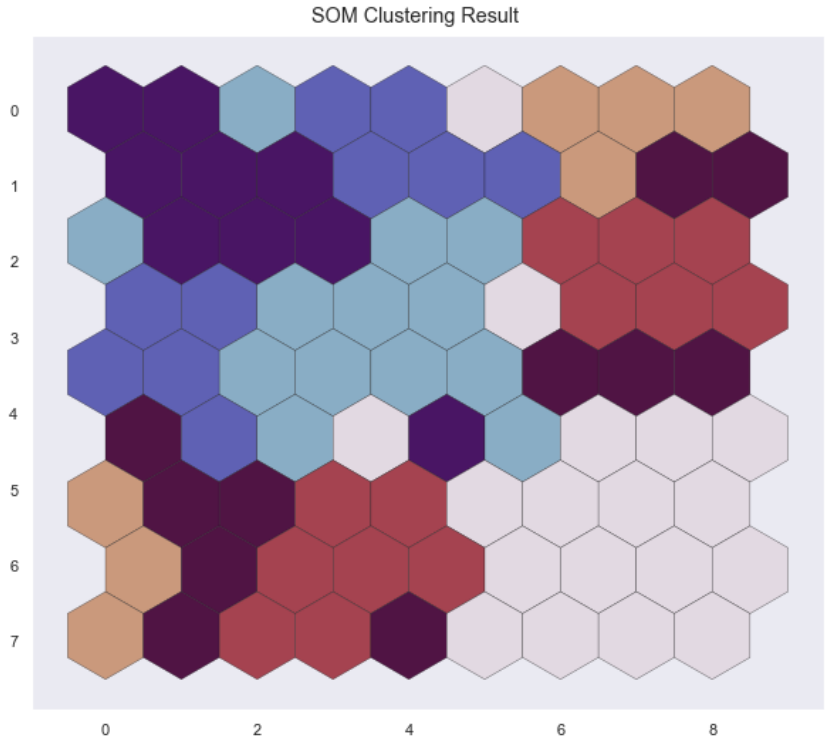
هر نورون در این نقشه‌ی ۹ در ۹، یک وزن به ابعاد ۵۶۱ دارد. در طی آموزش شبکه، هر بردار از مجموعه داده ورودی به یک نورون در شبکه، که وزن آن با مقدار نورون شباهت بیشتری دارد، نسبت داده می‌شود. پس از اتمام آموزش، به ازای هر داده ورودی، میتوانیم فاصله‌ی آن داده تا تمام نورون‌های شبکه را محاسبه و به عنوان داده ورودی جدید استفاده کنیم. واضح است که این داده ورودی جدید دارای ابعاد ۹ در ۹ (ابعاد نقشه) است. لازم به ذکر است که معمولا به جای استفاده از فاصله داده تا هر نورون، از معکوس فاصله استفاده می‌شود تا نورون‌های نزدیک‌تر وزن بیشتری بگیرند.



شکل ۲- داده ورودی تبدیل شده به ابعاد ۹ در ۹

سوال ۲)

از بین نورون‌های موجود در نقشه، نورون هایی که هیچگاه به عنوان نورون **"برنده"** انتخاب نمی‌شوند، نورون مرده نام دارند. این نورون ها باعث هدر رفتن فضای بازنمایی[1] می‌شوند زیرا هیچ داده‌ای هیچ‌وقت به این نورون‌ها نسبت داده نمی‌شود و موقعی که از شبکه برای کاهش بعد استفاده میکنیم، دارای شبکه ای با اندازه‌ی موثر کمتری هستیم.



شکل ۳- نورون‌های مرده با رنگ سفید مشخص شده اند

مشکل دیگری که این نورون‌ها دارند این است که با توجه به الگوریتم خاصِ آموزش، نورون‌های مرده به ندرت به زندگی بازمی-گردند. به این معنی که اگر یک نورون مرد(!)، داده های جدید به نورون‌های دیگر با وزن مناسب‌تر(نزدیک‌تر) نسبت داده می‌شوند و در مراحل بعد، همین نورون‌ها هستند که وزنشان بهبود می‌یابد و نورون‌های مرده همچنان مرده باقی می‌مانند. مخصوصا اگر در همسایگی نورون‌های مرده دیگر باشند.

از دلایل اصلی بوجود آمدن نورون‌های مرده، مقداردهی اولیه اشتباه است. بنابراین یکی از راه‌حل‌های کاهش نورون‌های مرده، مقداردهی اولیه بهتر مانند **مقداردهی اولیه تصادفی** است. راه دیگری که برای کاهش نورون‌های مرده وجود دارد، این است که **به صورت رندوم هر از گاهی تعدادی از نورون‌های مرده را به طور مصنوعی به سمت داده‌ها ببریم**. با این کار انتظار می-رود که تعدادی از نورون‌هایی مرده، که در همسایگی نورون‌های برنده دیگر نیز قرار ندارند، شانس دوباره ای داشته باشند و با تغییر

[1] Representation

وزنشان به نورون برنده تبدیل شوند. همانطور که مشخص است، این کار باعث می‌شود در مراحل بعدی نورون‌های مرده‌ای که اطراف این نورون مرده قرار داشتند نیز، به روند آموزش بازگردند(به دلیل آپدیت وزن نورون‌های همسایه در هر مرحله).

سوال ۳)

مجموعه داده شامل حدوداً ۷۵۰۰ داده آموزش و ۳۰۰۰ داده آزمون است. این داده‌ها به کمک کتابخانه Pandas فراخوانی شدند و پس از انجام مرحله Shuffling، به سه دسته **آموزش**، **اعتبارسنجی** و **آزمون** با ابعاد زیر تقسیم شدند.

```
X_train shape: (5881, 561)
y_train shape: (5881,)

X_valid shape: (1471, 561)
y_valid shape: (1471,)

X_test shape: (2947, 561)
y_test shape: (2947,)
```

شکل ۴- تقسیم بندی مجموعه داده به دسته های آموزش، اعتبارسنجی و آزمون

همچنین در ادامه این گزارش، از رنگ‌های ذکر شده در تصویر زیر برای نمایش کلاس‌های مختلف استفاده می‌شود.

شکل ۵- رنگ‌های متناظر با کلاس‌های مختلف

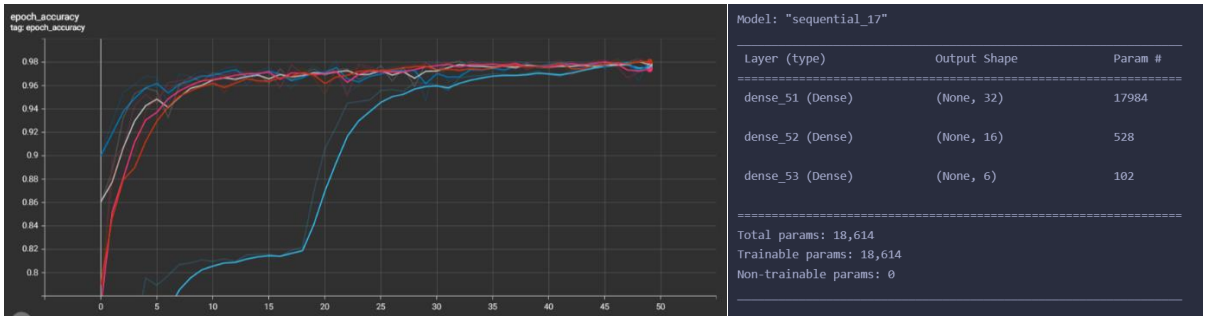
سوال ۴)

در این مرحله مشابه تمرین سری ۲، با استفاده از کتابخانه Tensor Flow و با کمک Keras به پیاده‌سازی و آزمایش چندین شبکه MLP با ابعاد و ویژگی‌های مختلف می‌پردازیم. این مدل‌ها را با داده‌ی آموزش، آموزش می‌دهیم و در نهایت با داده‌ی آزمون، بررسی می‌کنیم.

در این بخش می‌توانید تعدادی از مدل های آزمایش‌شده را مشاهده کنید.

Model	Learning Rate	Epoch	Test Accuracy
32 (sigmoid) – 16(sigmoid) – 6(softmax)	**0.001**	**50**	***0.954***
128 (sigmoid) – 64(sigmoid) – 6(softmax)	0.001	50	*0.952*
64 (sigmoid) – 32(sigmoid) – 6(softmax)	0.001	50	*0.949*
256 (sigmoid) – 256(sigmoid) – 6(softmax)	0.01	20	*0.9444*
1024 (sigmoid) – 512(sigmoid) – 6(softmax)	0.01	20	*0.09345*

در نهایت بهترین نتیجه را از مدلی با مشخصات زیر گرفتیم.



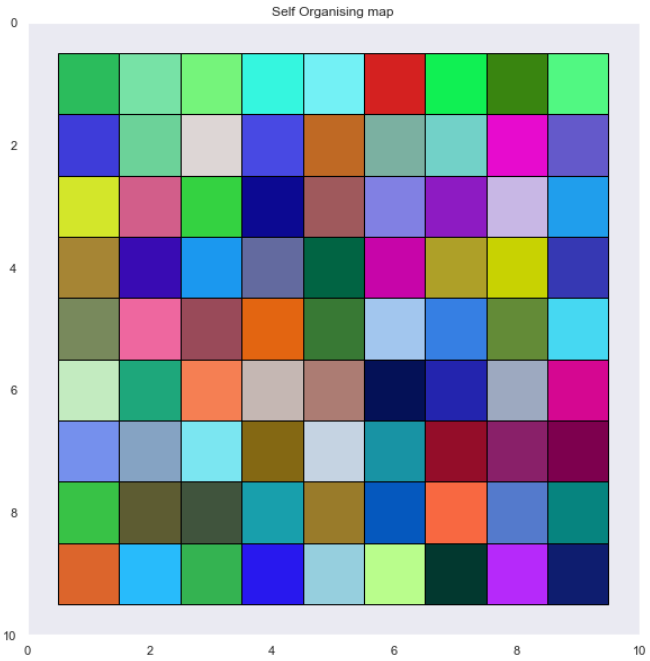
بنابراین بهترین دقت دسته بندی، با استفاده از داده‌ی خام و بدون استفاده از شبکه‌ی خودسازمانده کوهونن برابر با **۹۵٪** بود.

سوال ۵)

در این مرحله به پیاده‌سازی شبکه خودسازمانده کوهونن می‌پردازیم و سپس تعدادی تابع جهت مصورسازی[۲] این شبکه تولید میکنیم.

قبل از اینکه این شبکه را برروی مجموعه داده اصلی سوال که ابعاد بالایی دارد اجرا کنیم، با یک مجموعه داده ساده تر از صحت کارکرد شبکه اطمینان حاصل میکنیم. مجموعه ای که برای این بخش درنظر گرفته‌ایم، مجموعه‌ای از ۲۰۰ بردار با مقادیر تصادفی در فضای RGB است. از آنجایی که هر بردار را میتوان با یک رنگ نمایش داد، به راحتی میتوانیم عملکرد این شبکه را مشاهده کنیم. هر نورون در این شبکه دارای یک وزن با ابعاد ۱ در ۳ خواهد بود که در طی آموزش سعی می‌شود این بردار وزن، به رنگ-های موجود در مجموعه داده نزدیک شود.

در تصویر زیر وزن‌های اولیه نورون‌ها در نقشه‌ی ۹ در ۹ را مشاهده میکنید.

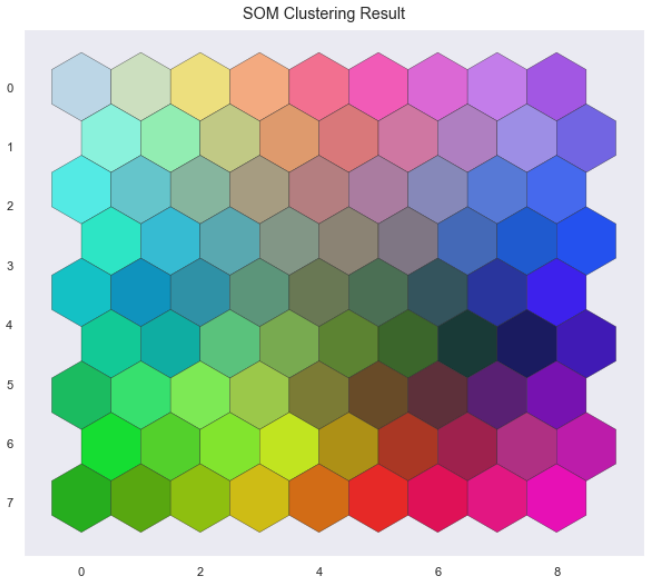


شکل ۶- وزن‌های اولیه شبکه خودسازمانده کوهونن در مثال رنگ‌ها

[2] Visualization

```
Iteration: 0, Loss: 0.4193, lr: 0.8000, R: 4.0000
Iteration: 100, Loss: 0.0308, lr: 0.7600, R: 3.8000
Iteration: 200, Loss: 0.0324, lr: 0.7200, R: 3.6000
Iteration: 300, Loss: 0.0325, lr: 0.6800, R: 3.4000
Iteration: 400, Loss: 0.0256, lr: 0.6400, R: 3.2000
Iteration: 500, Loss: 0.0219, lr: 0.6000, R: 3.0000
Iteration: 600, Loss: 0.0155, lr: 0.5600, R: 2.8000
Iteration: 700, Loss: 0.0161, lr: 0.5200, R: 2.6000
Iteration: 800, Loss: 0.0132, lr: 0.4800, R: 2.4000
Iteration: 900, Loss: 0.0099, lr: 0.4400, R: 2.2000
Iteration: 1000, Loss: 0.0123, lr: 0.4000, R: 2.0000
Iteration: 1100, Loss: 0.0075, lr: 0.3600, R: 1.8000
Iteration: 1200, Loss: 0.0055, lr: 0.3200, R: 1.6000
Iteration: 1300, Loss: 0.0054, lr: 0.2800, R: 1.4000
Iteration: 1400, Loss: 0.0045, lr: 0.2400, R: 1.2000
Iteration: 1500, Loss: 0.0033, lr: 0.2000, R: 1.0000
Iteration: 1600, Loss: 0.0019, lr: 0.1600, R: 0.8000
Iteration: 1700, Loss: 0.0012, lr: 0.1200, R: 0.6000
Iteration: 1800, Loss: 0.0007, lr: 0.0800, R: 0.4000
Iteration: 1900, Loss: 0.0003, lr: 0.0400, R: 0.2000
Iteration: 1999, Loss: 0.0000, lr: 0.0004, R: 0.0020
```

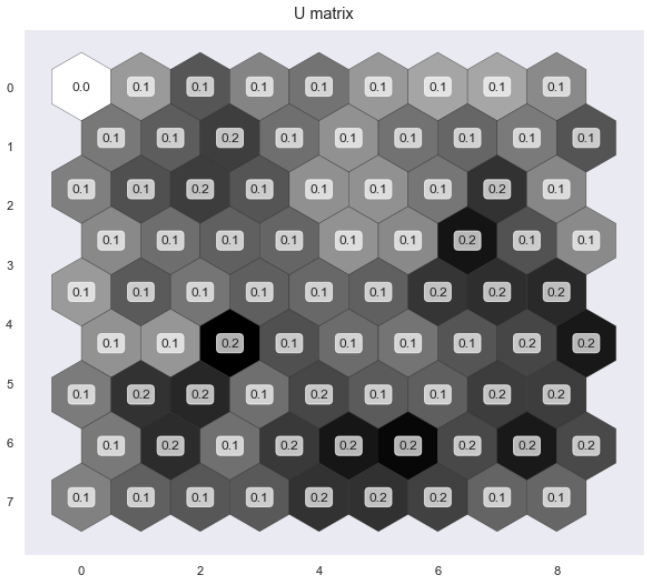
پس از آموزش مدل به اندازه‌ی ۲۰۰۰ مرحله، شبکه ما توانست رنگ‌های مشابه را به نورون‌های نزدیک به هم نسبت دهد.



شکل ۷- نتیجه آموزش شبکه خودسازمانده کوهونن بر روی مجموعه داده رنگ‌ها

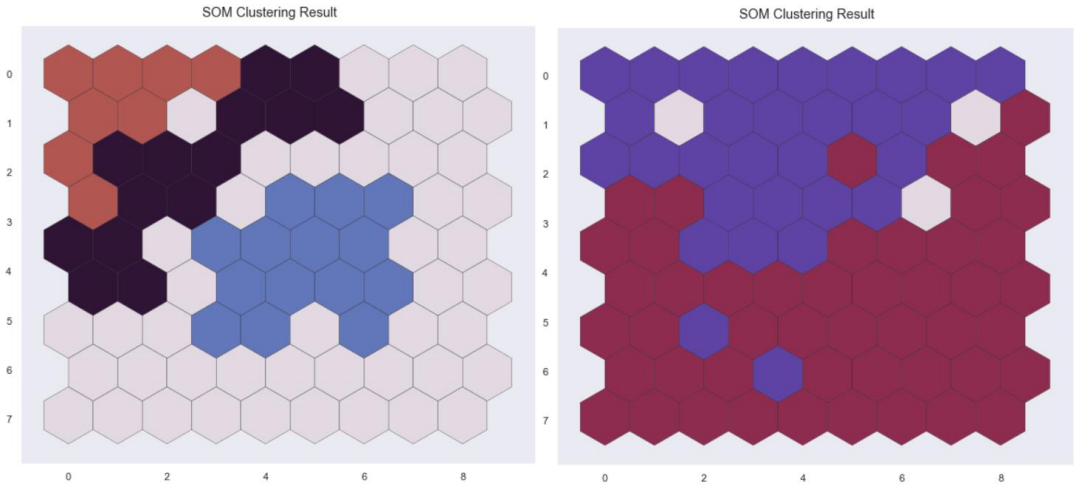
لازم به ذکر است که فضای رنگی RGB بهترین گزینه برای مواقعی که به دنبال فاصله اقلیدسی رنگ‌های مختلف هستیم نیست. در این مثال بهتر بود رنگ‌ها را در یک فضای رنگی مناسب تر مانند L*a*b یا YUV نمایش می‌دادیم. اما با این حال نتیجه قابل قبولی گرفتیم به این معنی که شبکه به خوبی کار می‌کند.

همچنین نمودار U-Matrix متناظر را در تصویر زیر مشاهده می‌کنید.



شکل ۸- نمایش U-Matrix متناظر با مثال رنگ‌ها

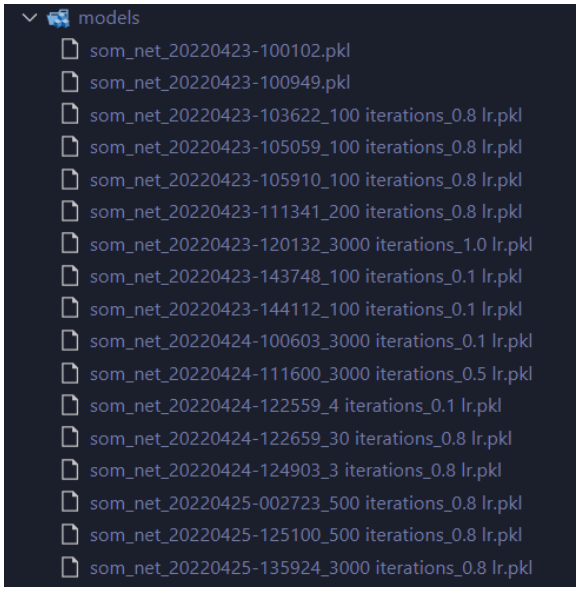
در ادامه بررسی هایی که انجام دادیم، این شبکه را برروی مجموعه داده‌های Iris, US Congress نیز آزمایش کردیم. تصویر سمت راست نتیجه خوشه‌بندی روی مجموعه داده US Congress است که جمهوری‌خواه‌ها از دموکرات ها جدا شده اند. همچنین تصویر سمت چپ نتیجه خوشه‌بندی سه نوع گل در مجموعه داده Iris است. وقت زیادی برروی بهینه کردن پارامترها برای این دو مثال صرف نشد و صرفا از نتیجه این مثال‌ها برای اطمینان از صحت کارکرد شبکه استفاده شد.



در نهایت نوبت به مجموعه داده اصلی سوال رسید. برای این مثال، از یک نقشه **۹ در ۹** استفاده شد. طبق آزمایش هایی که انجام دادیم، نقشه کوچکتر تعداد نورون های مرده را کاهش می‌داد ولی دقت کافی را نداشت و تعدادی از کلاس‌ها هم‌پوشانی زیادی داشتند.

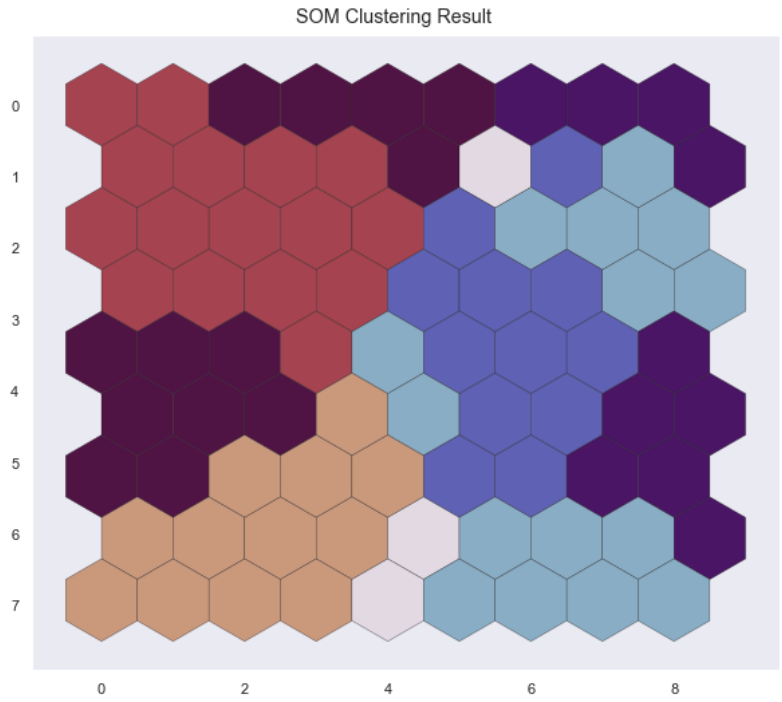
در این سوال علاوه بر نمودارهای خواسته شده توسط سوال، یک نمودار "اکثریت آرا" نیز رسم شد که کمک به سزایی به تحلیل نتیجه مدل میکند. برای رسم این نمودار، به ازای هر نورون در نقشه، رنگ متناظر با فراوان‌ترین کلاس از بین داده های ورودی نسبت داده شده به آن نورون را استفاده کردیم. بار دیگر جدول هر کلاس و رنگ متناظر آن را مشاهده می‌کنیم(کلاس **صفر** نشان دهنده **نورون های مرده** است).

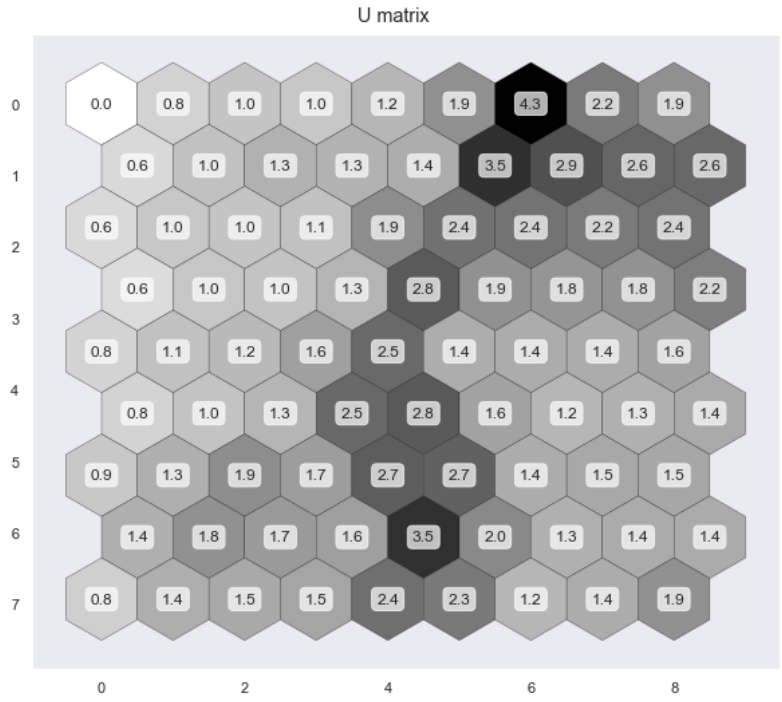
از آنجایی که آموزش با تمام داده‌های آموزش وقت زیادی می‌گرفت، تصمیم بر آن شد که از بخشی (حدود ۲۰ درصد) از داده های آموزش برای آموزش دادن شبکه استفاده کنیم. این کار باعث شد بتوانیم تعداد آزمایش‌های بیشتری انجام دهیم و همچنین آموزش را تا Iteration های بیشتری ادامه دهیم. در طی آموزش این شبکه متوجه شدیم که ادامه دادن آموزش تا Iteration های بیشتر تاثیر خوبی بر کاهش تعداد نورون‌های مرده دارد. همچنین در این آزمایش‌ها شعاع همسایگی و نرخ یادگیری را به طور خطی کاهش دادیم. در اکثر آزمایش ها شعاع همسایگی اولیه برابر با نصف اندازه نقشه است و نرخ یادگیری اولیه برابر با عدد ۰٫۸ می‌باشد.

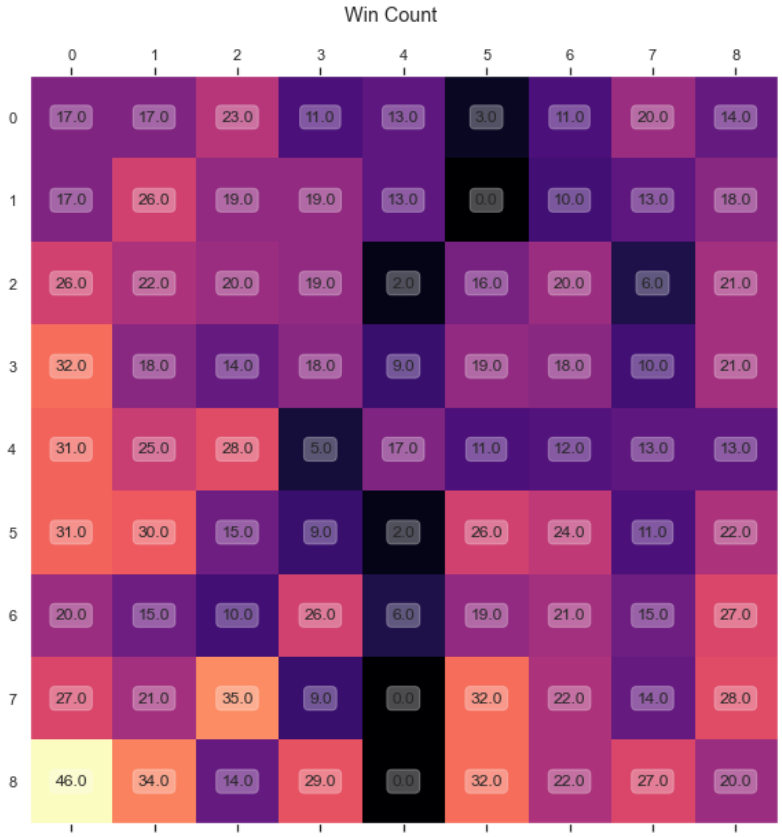


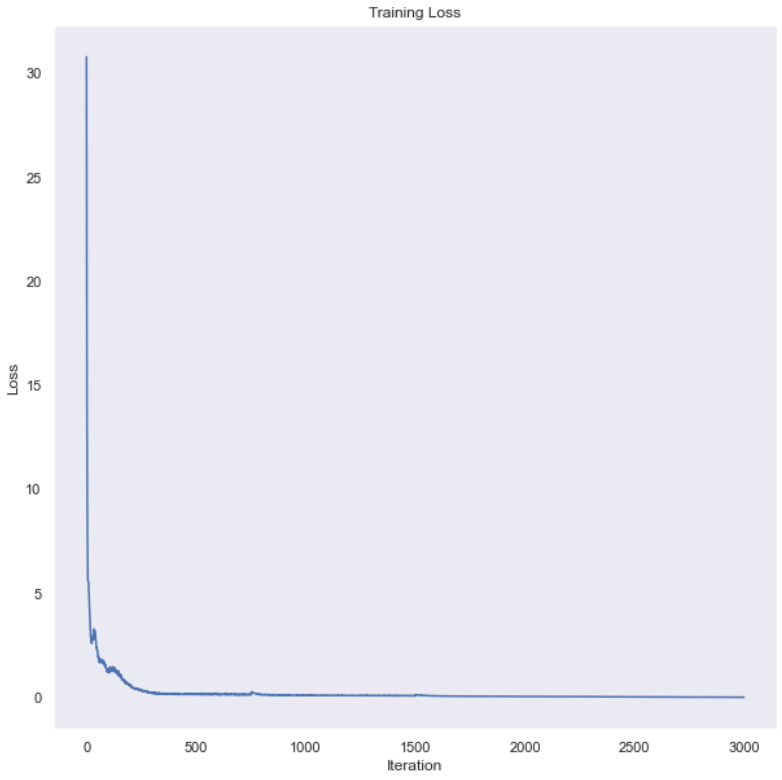
شکل ۹- تعدادی از مدل‌های آموزش داده شده با پارامترهای مختلف(ذخیره شده جهت استفاده مجدد)

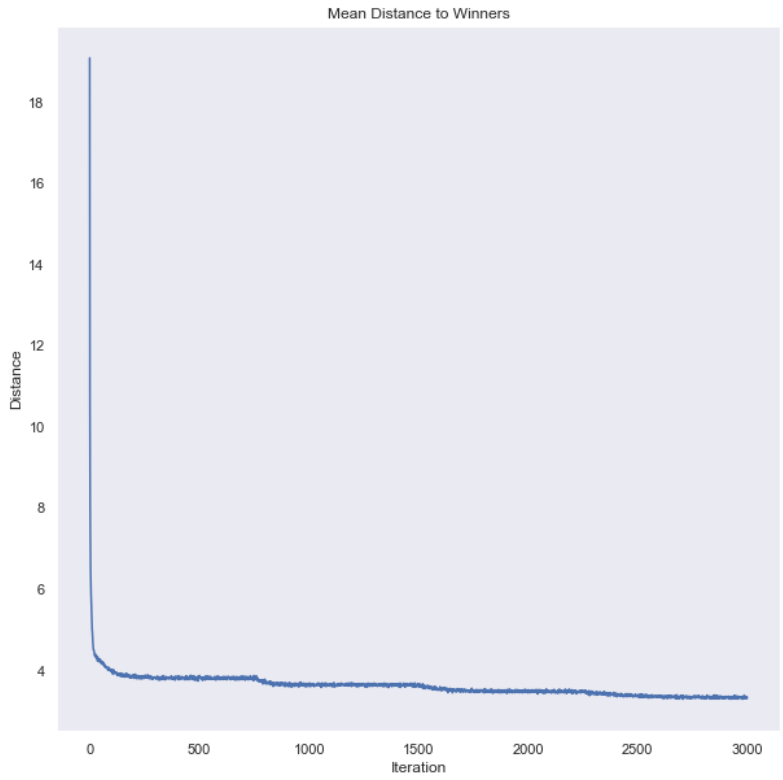
در این بخش نمودارهای مختلفی که صورت سوال از ما خواسته به علاوه نمودار اکثر آرا را برای بهترین مدل، نمایش میدهیم.

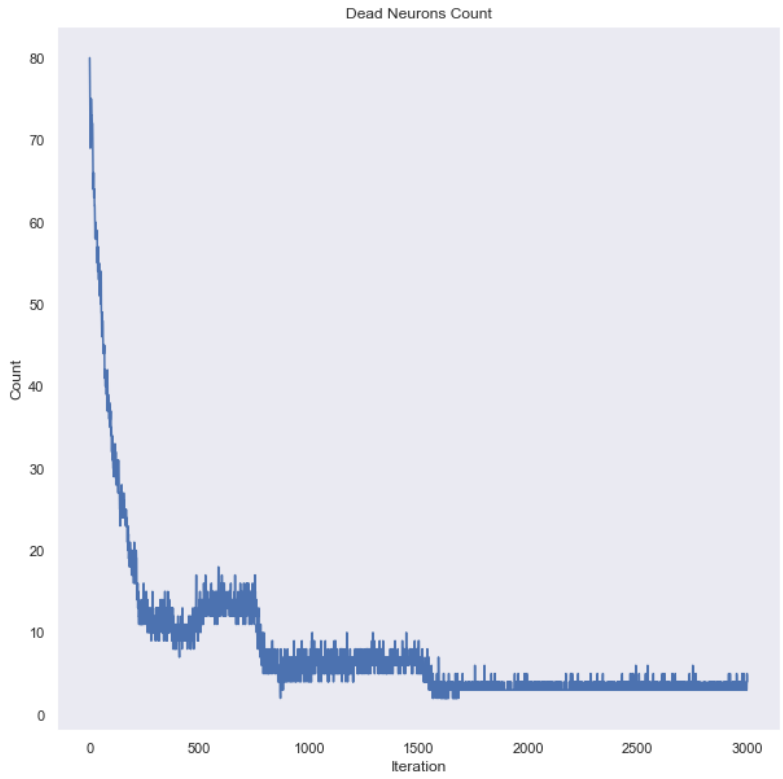


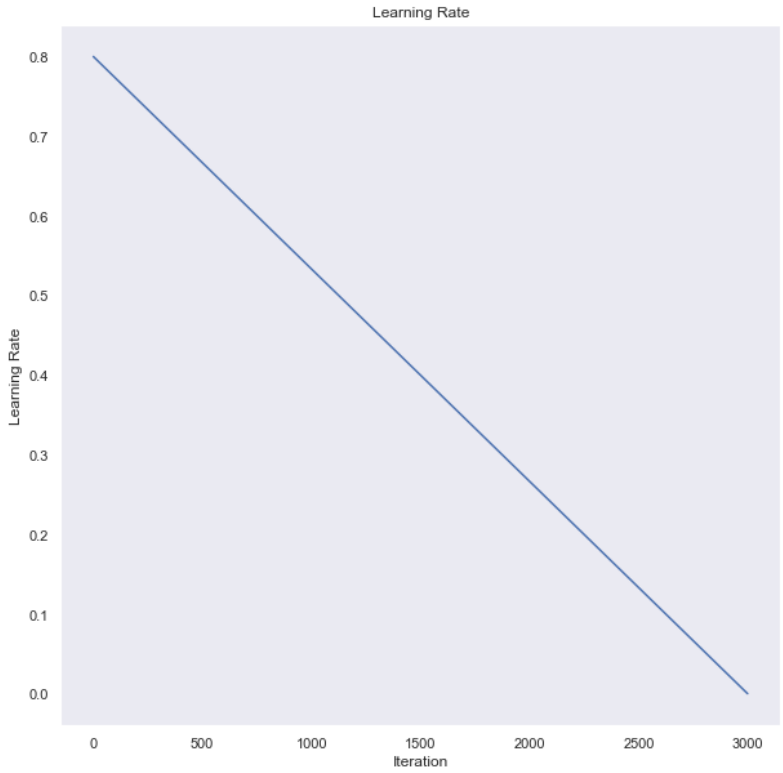


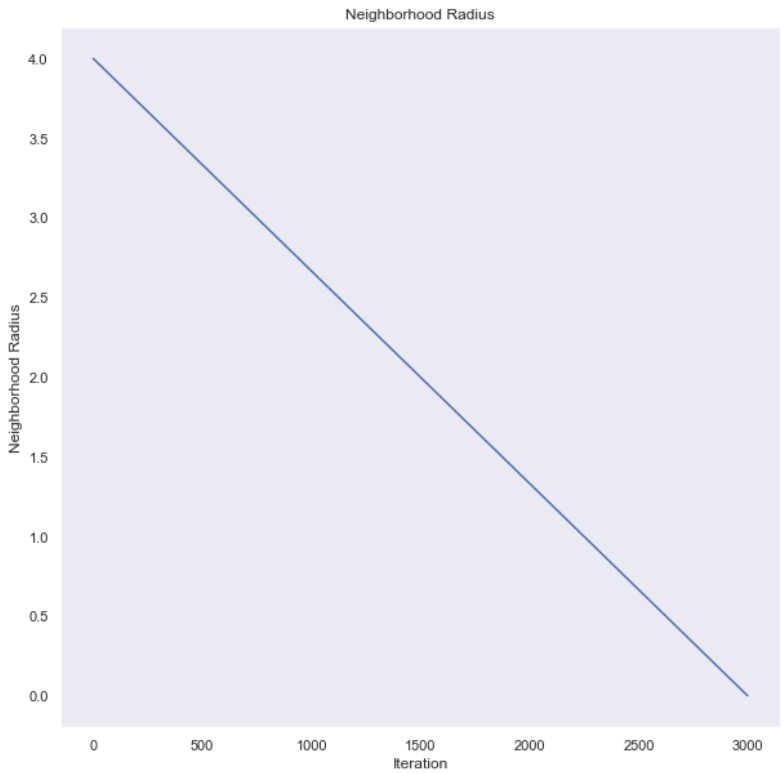

Win Count
0 1 2 3 4 5 6 7 8
0 17.0 17.0 23.0 11.0 13.0 3.0 11.0 20.0 14.0
1 17.0 26.0 19.0 19.0 13.0 0.0 10.0 13.0 18.0
2 26.0 22.0 20.0 19.0 2.0 16.0 20.0 6.0 21.0
3 32.0 18.0 14.0 18.0 9.0 19.0 18.0 10.0 21.0
4 31.0 25.0 28.0 5.0 17.0 11.0 12.0 13.0 13.0
5 31.0 30.0 15.0 9.0 2.0 26.0 24.0 11.0 22.0
6 20.0 15.0 10.0 26.0 6.0 19.0 21.0 15.0 27.0
7 27.0 21.0 35.0 9.0 0.0 32.0 22.0 14.0 28.0
8 46.0 34.0 14.0 29.0 0.0 32.0 22.0 27.0 20.0


Training Loss
Loss
30
25
20
15
10
5
0
0 500 1000 1500 2000 2500 3000
Iteration


Mean Distance to Winners
Distance
18
16
14
12
10
8
6
4
0
500
1000
1500
2000
2500
3000
Iteration


Dead Neurons Count
Count
80
70
60
50
40
30
20
10
0
0
500
1000
1500
2000
2500
3000
Iteration


Learning Rate
Learning Rate
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
500
1000
1500
2000
2500
3000
Iteration

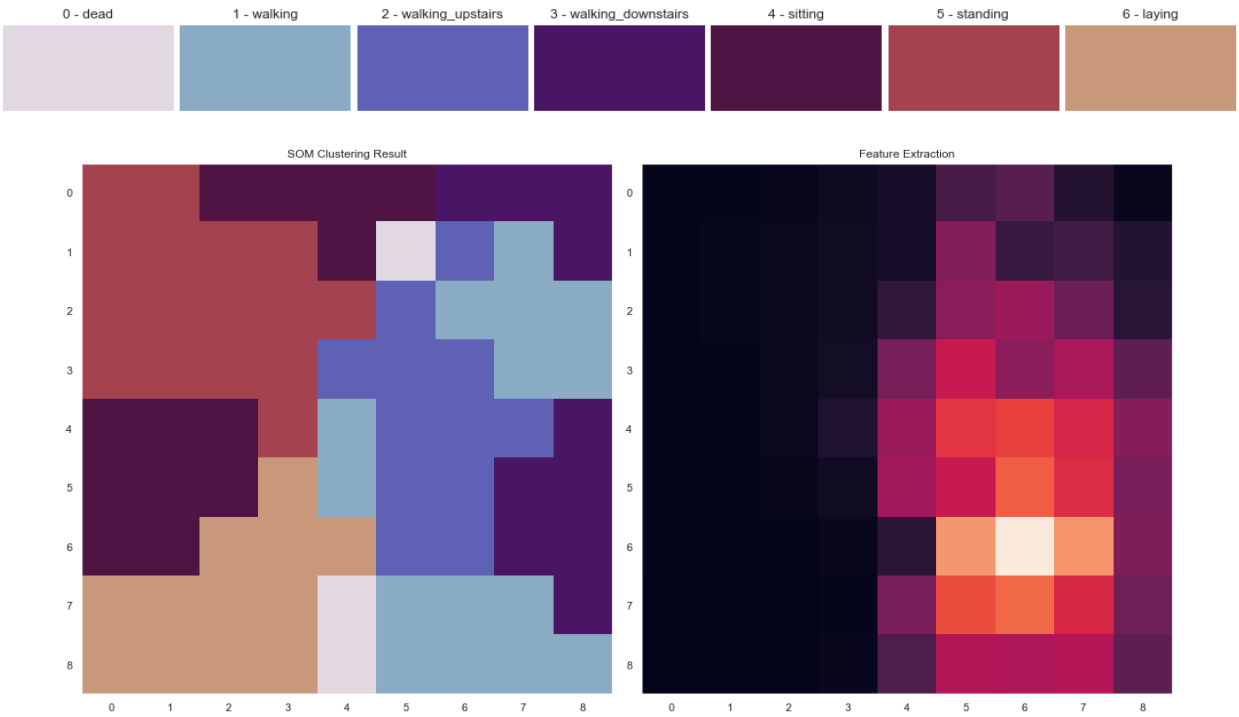

Neighborhood Radius
Neighborhood Radius
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
500
1000
1500
2000
2500
3000
Iteration

سوال ۶)

پس از آموزش دادن مدل، نوبت به استخراج ویژگی رسید. روند کلی استخراج ویژگی به این شکل است:

۱. یک داده ورودی انتخاب میکنیم
۲. یک نقشه، هم اندازه با نقشه SOM تولید میکنیم
۳. فاصله مقدار این داده را با وزن تمامی نورون ها در نقشه SOM محاسبه میکنیم و در سلول متناظر در نقشه جدید قرار میدهیم.
۴. چون میخواهیم اعداد بزرگ‌تر نشان دهنده شباهت بیشتری باشند(نه فاصله بیشتر) به جای قرار دادن فاصله در هر سلول، معکوس فاصله را قرار میدهیم.

به عنوان مثال برای یکی از داده های ورودی که کلاس متناظر آن **۲** (بالارفتن از پله) بود این ویژگی را استخراج کردیم و نقشه زیر را نتیجه گرفتیم.



همانطور که می‌بینید، قسمت‌های روشن در نقشه جدید متناظر با نورون‌هایی هستند که بیشتر از بقیه نورون‌ها به کلاس متناظر با این داده ورودی خاص نزدیک بوده اند.

به عبارتی ما توانستیم داده ورودی با ابعاد ۵۶۱ را به یک بردار ۸۱ بعدی کاهش دهیم و در عین حال ارتباط بین کلاس متناظر را حفظ کرده ایم. در ادامه این کار را به ازای کل داده‌های موجود در مجموعه داده انجام می‌دهیم و شبکه MLP سوال ۴ را این‌بار با این داده های جدید آموزش می‌دهیم.

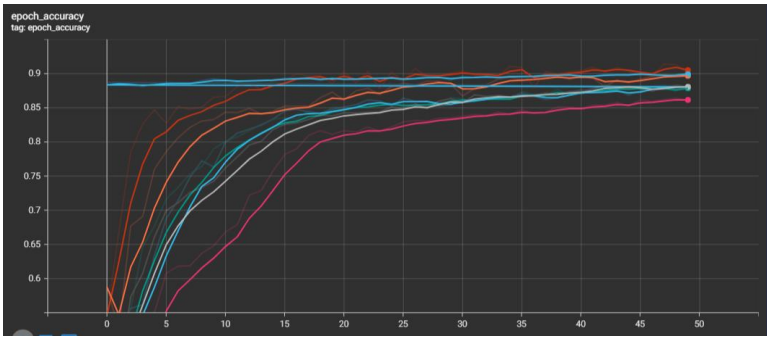
```python
# Transform
X_train_transformed = som_net.transform(X_train)
X_valid_transformed = som_net.transform(X_valid)
X_test_transformed = som_net.transform(X_test)
```

شکل ۱۰- انتقال داده‌ها از فضای ۵۶۱ بعدی به ۸۱ بعدی

در ادامه تعدادی از آزمایش‌های انجام شده برای این بخش را نمایش می‌دهیم.

Model	Learning Rate	Epoch	Test Accuracy
512 (sigmoid) – 256x2(sigmoid) – 6(softmax)	**0.001**	**50**	*0.878*
32 (sigmoid) – 16(sigmoid) – 6(softmax)	0.001	50	*0.870*
128 (sigmoid) – 64(sigmoid) – 6(softmax)	0.001	50	*0.867*
64 (sigmoid) – 32(sigmoid) – 6(softmax)	0.001	20	*0.852*
12 (sigmoid) – 8(sigmoid) – 6(softmax)	0.001	20	*0.840*

در نهایت بهترین نتیجه را از مدلی با مشخصات زیر گرفتیم.




```
Model: "sequential_33"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_100 (Dense)            (None, 512)               41984
_________________________________________________________________
dense_101 (Dense)            (None, 256)               131328
_________________________________________________________________
dense_102 (Dense)            (None, 256)               65792
_________________________________________________________________
dense_103 (Dense)            (None, 6)                 1542
=================================================================
Total params: 240,646
Trainable params: 240,646
Non-trainable params: 0
```

لازم به ذکر است که این نتایج قطعا بهترین نتایج نیستند و با آزمایش‌های بیشتر و با ایده‌های مختلفِ دیگر می‌توان نتیجه را تا حدی بهبود داد. همچنین به دلیل مشکل زمان آموزش، شبکه را بر روی بخشی از داده ها آموزش دادیم.

نتیجه‌گیری: با وجود اینکه داده ها را از ۵۶۱ بعد به ۸۱ بعد کاهش دادیم، دقت مدل در تشخیص کاهش بسیار کمی داشت. این کاهش بعد باعث سریع تر شدن روند آموزش و اجرای مدل شد و در نتیجه با استفاده از خاصیت خوشه‌بندی SOM توانستیم به شکل زیبایی داده‌ها را در ابعاد کمتر مشاهده کنیم.

پایان