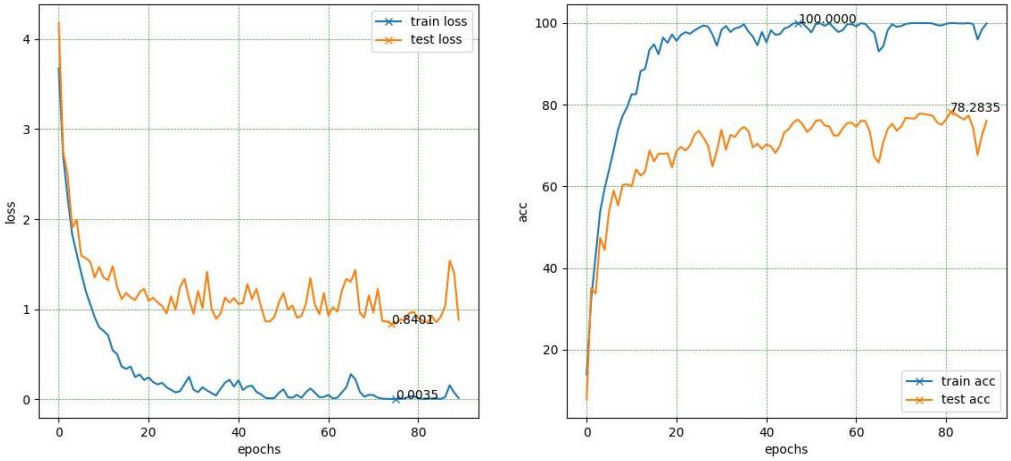
گزارش فاز چهارم پروژه هوش محاسباتی

کورش حسن زاده – سید محمد حسام قاسمی

در این فاز ابتدا یک شبکه CNN با معماری داده شده در صورت پروژه طراحی کردیم. این معماری شامل لایه های conv-batchNorm-AvgPool است که هر بلاک این شبکه شامل conv-batchNorm-relu است.

در فاز اول این شبکه بر روی دیتاست A آموزش داده شد که نمودار های یادگیری آن به شرح زیر است:



مقادیر تنظیم شده برای یادگیری این شبکه به این صورت است:

$$learning\ rate = 0.0001, epochs = 90, batch\ size = 128$$

همانطور که مشاهده میشود از epoch پنجاهم به بعد، فاصله بین منحنی trainو test زیادتر از قبل شده و بعد از آن حدودا ثابت میشود. از همین جهت شبکه از این جا به بعد به سمت overfit شدن میرفته ولی چون تفاوت آن ها تقریبا ثابت میشده احتمالا overfit نشده و میتوانیم فرایند آموزش را از epoch پنجاهم به بعد متوقف کنیم.

فاز دوم:

در این فاز قرار است مدل آموزش دیده بر روی دیتا ست A را با استفاده از 3 روش TransferLearining بر روی دیتا ست B هم آموزش دهیم. یک تابع TransferLesrning برای انجام این کار پیاده سازی کردیم که کار اصلی آن عوض کردن لایه آخر شبکه و freeze کردن لایه های قبلی در صورت نیاز است.

روش اول:

در این روش از ما خواسته شده تا تمام لایه های شبکه بجز لایه آخر را که fullyConnected است نگه داریم. در لایه آخر هم باید 80 نورون لایه آخر شبکه قبلی نگه داشته شوند و تنها 20 نورون اضافه شوند و شبکه بر روی کل لایه ها مجددا با دیتادست B آموزش ببینند. برای این کار ابتدا مدل اصلی را به تابع ذکر شده پاس میدهیم و ابتدا لایه آخر آن را برداشته و لایه ها و وزن ها و بایاس های لایه قبلی را در مدل جدیدی کپی میکنیم. لایه آخر آن را ابتدا به صورت خالی میسازیم و وزن ها و بایاس های 80 نورون مدل اصلی را در 80 نورون ابتدایی مدل جدید کپی میکنیم. سپس 20 نورون به آن اضافه میکنیم و شبکه را بر روی دیتاست B آموزش میدهیم. تمام وزن ها در این روش باید آپدیت شوند که در عکس های زیر مشاهده میشود:

وزن های لایه conv2_1 مدل اصلی

```
my_model.conv2_1.weight
✓ 0.0s
Parameter containing:
tensor([[[[ 0.0243,  0.0081, -0.0310],
          [ 0.0159,  0.0357,  0.0175],
          [-0.0122,  0.0183, -0.0131]],

         [[ 0.0314,  0.0361,  0.0353],
          [ 0.0277, -0.0362,  0.0362],
          [-0.0342,  0.0127, -0.0202]],

         [[ 0.0172, -0.0393,  0.0336],
          [ 0.0097, -0.0314,  0.0124],
          [-0.0351,  0.0378, -0.0315]],

         ...,

         [[ 0.0168, -0.0345, -0.0263],
          [ 0.0081,  0.0238, -0.0102],
          [ 0.0256, -0.0207, -0.0335]],

         [[-0.0047, -0.0321, -0.0239],
          [ 0.0305,  0.0245, -0.0117],
          [ 0.0113,  0.0061, -0.0388]],

         [[ 0.0230, -0.0144,  0.0096],
          [-0.0147,  0.0346, -0.0381],
...
          [ 0.0294,  0.0121, -0.0306]],

         [[-0.0122, -0.0091, -0.0132],
          [-0.0072, -0.0068, -0.0023],
          [ 0.0243,  0.0047,  0.0386]]]], device='cuda:0', requires_grad=True)
```

وزن های لایه fc مدل اصلی

```
my_model.fc1.weight
✓ 0.0s
Parameter containing:
tensor([[-0.0209, -0.0121, -0.0157,  ..., -0.0068,  0.0096, -0.0068],
        [-0.0053, -0.0398, -0.0194,  ..., -0.0157,  0.0103,  0.0163],
        [ 0.0045,  0.0165, -0.0100,  ..., -0.0168, -0.0182, -0.0100],
        ...,
        [-0.0096, -0.0007,  0.0079,  ..., -0.0189, -0.0239, -0.0140],
        [ 0.0040, -0.0103, -0.0229,  ..., -0.0071, -0.0147, -0.0208],
        [-0.0057, -0.0025, -0.0238,  ...,  0.0081, -0.0091, -0.0211]],
       device='cuda:0', requires_grad=True)
```

وزن های لایه conv2_1 مدل جدید

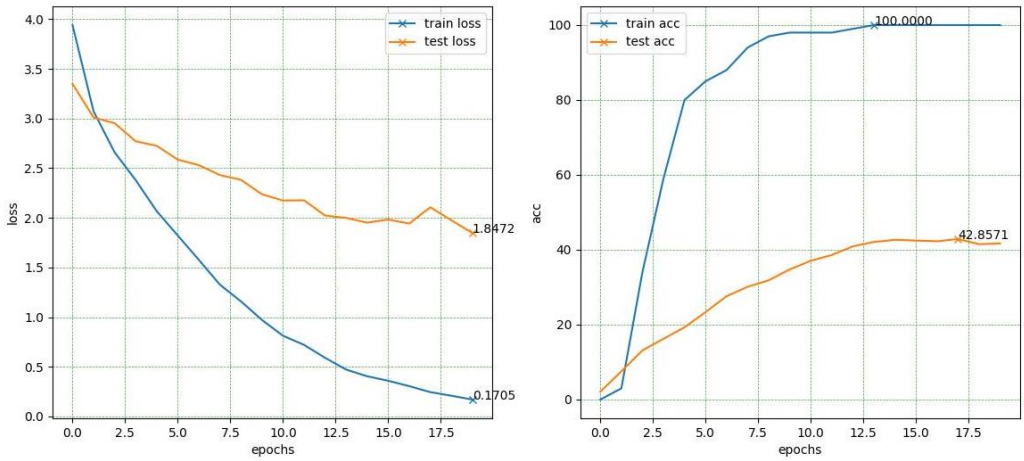
```
newModel1.conv2_1.weight
✓ 0.0s
Parameter containing:
tensor([[[[ 0.0242,  0.0078, -0.0313],
          [ 0.0160,  0.0359,  0.0177],
          [-0.0121,  0.0185, -0.0129]],

         [[ 0.0308,  0.0356,  0.0358],
          [ 0.0275, -0.0368,  0.0365],
          [-0.0343,  0.0120, -0.0202]],

         [[ 0.0167, -0.0398,  0.0333],
          [ 0.0096, -0.0316,  0.0122],
          [-0.0350,  0.0376, -0.0316]],

         ...,

         [[ 0.0180, -0.0335, -0.0254],
          [ 0.0092,  0.0252, -0.0090],
          [ 0.0257, -0.0199, -0.0323]],

         [[-0.0053, -0.0328, -0.0239],
          [ 0.0300,  0.0242, -0.0113],
          [ 0.0109,  0.0061, -0.0381]],

         [[ 0.0225, -0.0146,  0.0097],
          [-0.0150,  0.0343, -0.0381],
...
          [ 0.0285,  0.0112, -0.0309]],

         [[-0.0125, -0.0093, -0.0126],
          [-0.0073, -0.0070, -0.0017],
          [ 0.0243,  0.0046,  0.0390]]]], device='cuda:0', requires_grad=True)
```

وزن های لایه fc مدل جدید

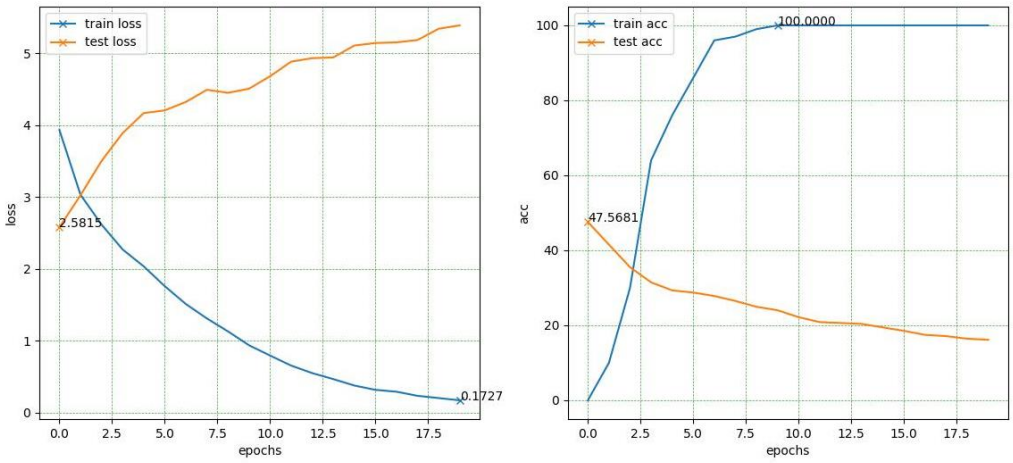
```
newModel1.fc1.weight
✓ 0.0s
Parameter containing:
tensor([[-0.0217, -0.0129, -0.0170,  ..., -0.0078,  0.0089, -0.0077],
        [-0.0070, -0.0416, -0.0212,  ..., -0.0171,  0.0089,  0.0148],
        [ 0.0038,  0.0159, -0.0106,  ..., -0.0177, -0.0194, -0.0114],
        ...,
        [ 0.0072,  0.0124, -0.0128,  ..., -0.0010,  0.0077,  0.0013],
        [-0.0039,  0.0040,  0.0151,  ...,  0.0122,  0.0033, -0.0007],
        [-0.0110,  0.0099,  0.0060,  ..., -0.0057, -0.0015,  0.0053]],
       device='cuda:0', requires_grad=True)
```

همانطور که مشاهده میشود وزن های لایه ابتدایی به مقدار کمی عوض شده ولی وزن های لایه های انتهایی عوض شده و مطابق B آموزش دیده.

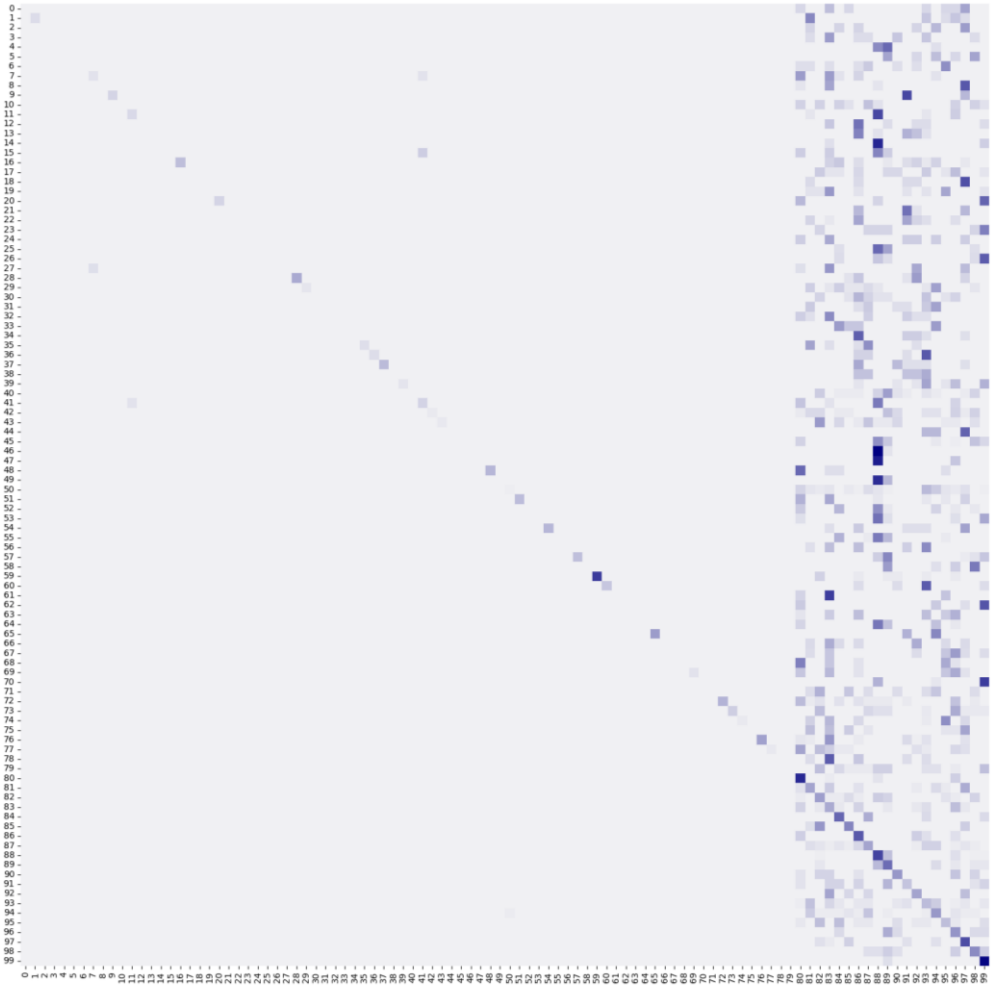
نمودار یادگیری و تست B



نمودار test_all:



ماتریس گمراهی برای ایپاک آخر:



همانطور که مشاهده میشود، این مدل صرفا بری روی داده های B خوب کار کرده و برای قطر اصلی ماتریس در سمت B پر رنگ تر بوده و برای داده های A عملکرد خوبی نداشته.

قطر اصلی بیانگر تطابق لیبل واقعی با لیبل پیش بینی شده است

روش دوم:

این روش همانند روش است با این تفاوت که وزن های و بایاس های لایه های قبلی باید freeze شوند و صرفا وزن های لایه آخر که شامل 100 نورون است باید آپدیت شوند. برای این کار مجددا مدل اصلی را به تابع ذکر شده پاس میدهیم و مجددا همان فرایند قبلی انجام میشود ولی در این حالت یک Flag هم به تابع ارسال میشود که نشان میدهد لایه های قبل از آخر باید freeze شوند و برای فریز کردن آن ها requeiredGrad آن ها False میگذاریم.

وزن ها به شرح زیر است:

weights after train

```
new_model2.conv2_1.weight
```

✓ 0.0s

```
Parameter containing:
tensor([[[[ 0.0243,  0.0081, -0.0310],
          [ 0.0159,  0.0357,  0.0175],
          [-0.0122,  0.0183, -0.0131]],

         [[ 0.0314,  0.0361,  0.0353],
          [ 0.0277, -0.0362,  0.0362],
          [-0.0342,  0.0127, -0.0202]],

         [[ 0.0172, -0.0393,  0.0336],
          [ 0.0097, -0.0314,  0.0124],
          [-0.0351,  0.0378, -0.0315]],

         ...,

         [[ 0.0168, -0.0345, -0.0263],
          [ 0.0081,  0.0238, -0.0102],
          [ 0.0256, -0.0207, -0.0335]],

         [[-0.0047, -0.0321, -0.0239],
          [ 0.0305,  0.0245, -0.0117],
          [ 0.0113,  0.0061, -0.0388]],

         [[ 0.0230, -0.0144,  0.0096],
          [-0.0147,  0.0346, -0.0381],
...
          [ 0.0294,  0.0121, -0.0306]],

         [[-0.0122, -0.0091, -0.0132],
          [-0.0072, -0.0068, -0.0023],
          [ 0.0243,  0.0047,  0.0386]]]], device='cuda:0')
```

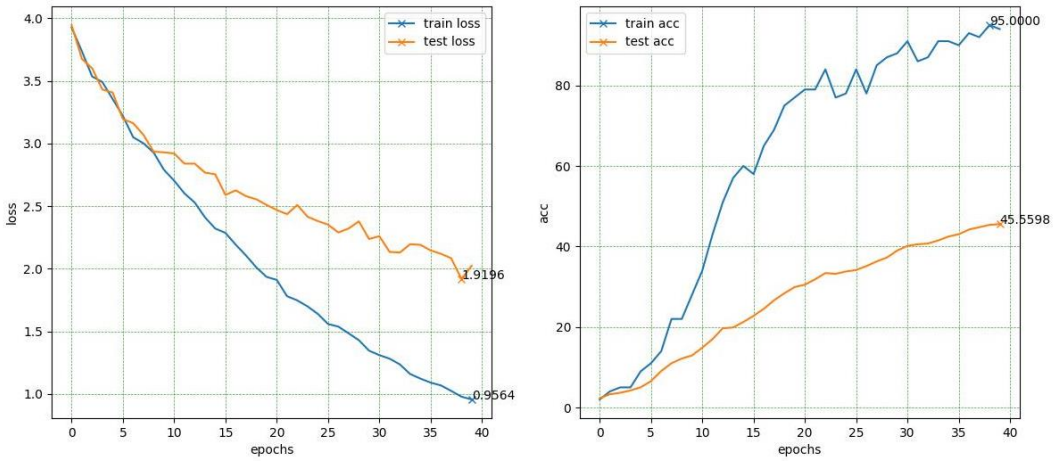
```
new_model2.fc1.weight
```

✓ 0.0s

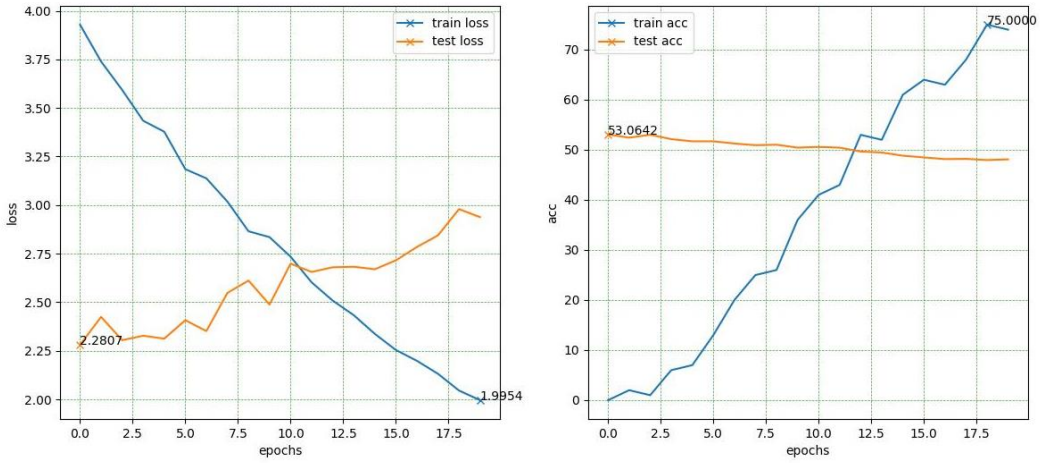
```
Parameter containing:
tensor([[-0.0227, -0.0140, -0.0176,  ..., -0.0088,  0.0077, -0.0084],
        [-0.0073, -0.0418, -0.0213,  ..., -0.0177,  0.0084,  0.0143],
        [ 0.0026,  0.0146, -0.0120,  ..., -0.0183, -0.0201, -0.0119],
        ...,
        [ 0.0021, -0.0087, -0.0170,  ..., -0.0096, -0.0043, -0.0090],
        [ 0.0079, -0.0046,  0.0068,  ...,  0.0036,  0.0115,  0.0027],
        [ 0.0093, -0.0026,  0.0112,  ...,  0.0113,  0.0092, -0.0182]],
       device='cuda:0', requires_grad=True)
```

همانطور که مشاهده میشود وزن های لایه های اولیه freeze شده و تغییری نکرده ولی وزن های لایه آخر عوض شده.

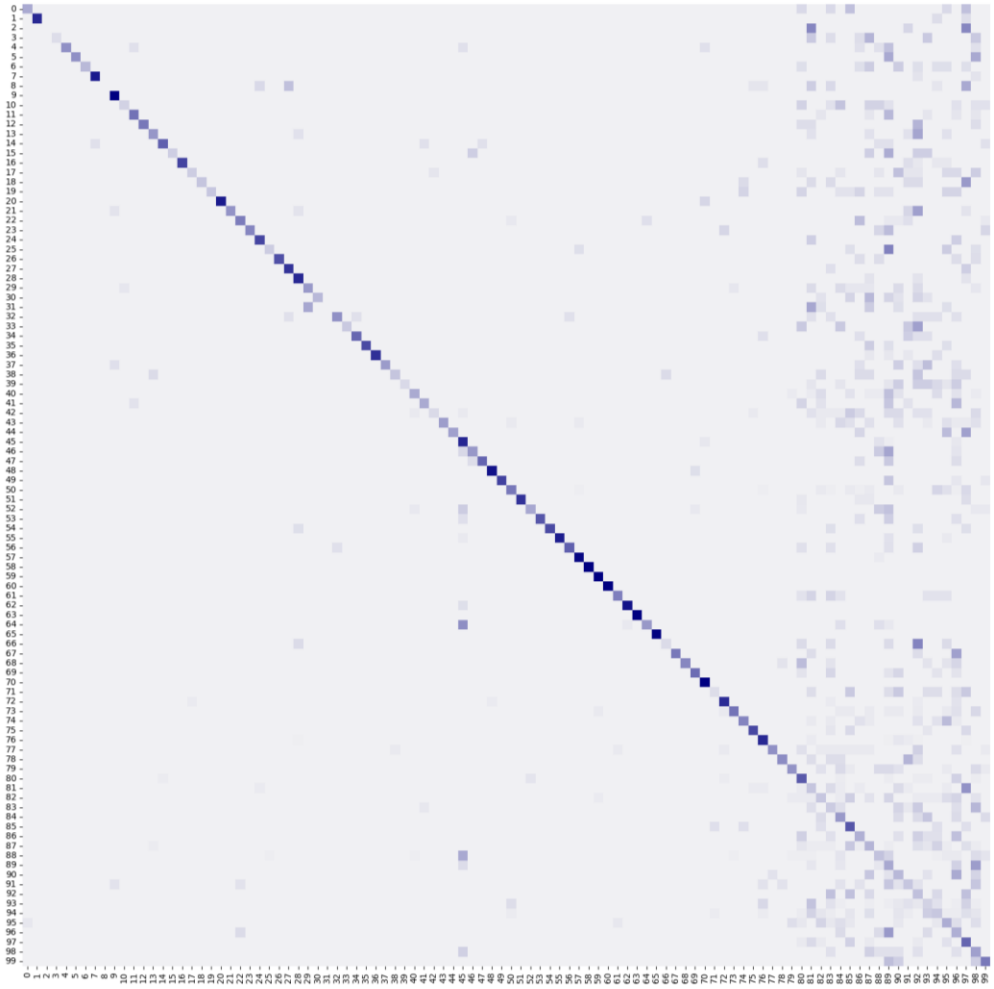
نمودار یادگیری و تست B:



نمودار test all:



ماتریس گمراهی برای ایپاک آخر:



همانطور که مشاهده میشود؛ چون لایه های اولیه فریز شده اند، قطر اصلی در کلاس های A هم نسبت به حالت قبل پر رنگ تر بوده و عملکرد بهتری نسبت به تست همه داشته، زیرا وزن های لایه های اولیه مطابق داده های A آموزش دیده اند.

علاوه بر این برای داده های کلاس B هم عملکرد نسبتا خوبی داشته ولی از روش اول بدتر بوده(فقط برای B)

قطر اصلی بیانگر تطابق لیبل واقعی با لیبل پیش بینی شده است

روش سوم:

این روش مانند روش 2 است با این تفاوت که در این روش 80 نورون اول لایه آخر که مربوط به کلاس های دیتاست A هستند هم باید freeze شوند. برای اینکار grad آن ها را 0 کردیم.

در این حالت باید وزن های لایه های اولیه و 80 نورون لایه آخر عوض نشوند و صرفا 20 نورون آخر لایه آخر عوض شوند.

وزن ها به شرح زیر است:

check the weights after train the model

```
new_model3.conv2_1.weight
```

✓ 0.0s

```
Parameter containing:
tensor([[[[ 0.0243,  0.0081, -0.0310],
          [ 0.0159,  0.0357,  0.0175],
          [-0.0122,  0.0183, -0.0131]],

         [[ 0.0314,  0.0361,  0.0353],
          [ 0.0277, -0.0362,  0.0362],
          [-0.0342,  0.0127, -0.0202]],

         [[ 0.0172, -0.0393,  0.0336],
          [ 0.0097, -0.0314,  0.0124],
          [-0.0351,  0.0378, -0.0315]],

         ...,

         [[ 0.0168, -0.0345, -0.0263],
          [ 0.0081,  0.0238, -0.0102],
          [ 0.0256, -0.0207, -0.0335]],

         [[-0.0047, -0.0321, -0.0239],
          [ 0.0305,  0.0245, -0.0117],
          [ 0.0113,  0.0061, -0.0388]],

         [[ 0.0230, -0.0144,  0.0096],
          [-0.0147,  0.0346, -0.0381],
...
          [ 0.0294,  0.0121, -0.0306]],

         [[-0.0122, -0.0091, -0.0132],
          [-0.0072, -0.0068, -0.0023],
          [ 0.0243,  0.0047,  0.0386]]]], device='cuda:0')
```

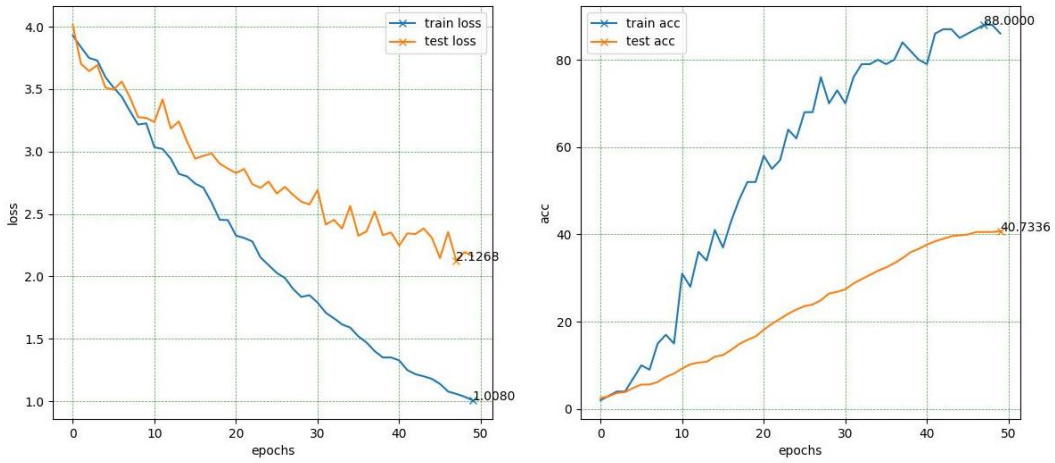
همانطور که مشاهده میشود؛ هم وزن های لایه اولیه تغییر نکرده و هم وزن های 80 نورون اول لایه آخر و تنها 20 نورون آخر لایه آخر آموزش دیده اند و وزن ها آپدیت شده اند.

```
new_model3.fc1.weight
```

✓ 0.0s

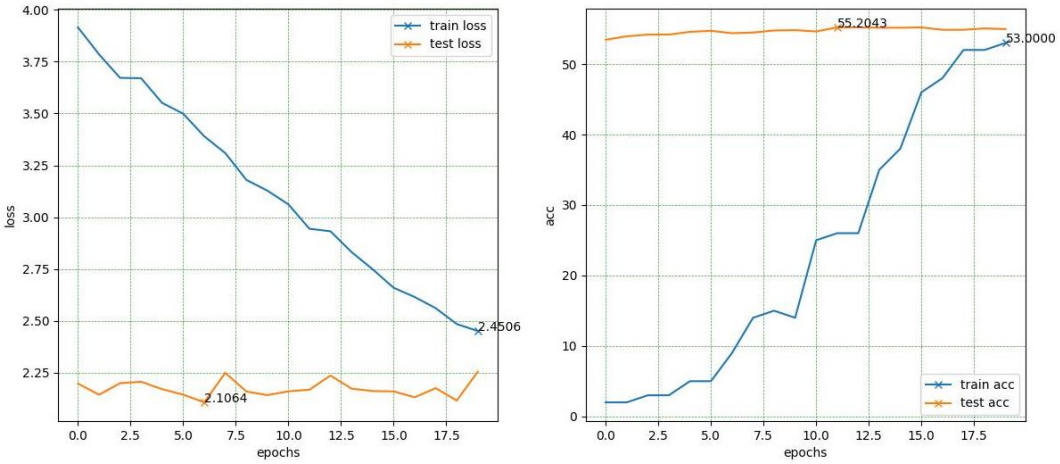
```
Parameter containing:
tensor([[-0.0209, -0.0121, -0.0157,  ..., -0.0068,  0.0096, -0.0068],
        [-0.0053, -0.0398, -0.0194,  ..., -0.0157,  0.0103,  0.0163],
        [ 0.0045,  0.0165, -0.0100,  ..., -0.0168, -0.0182, -0.0100],
        ...,
        [-0.0128, -0.0095, -0.0142,  ..., -0.0002,  0.0119,  0.0182],
        [-0.0046, -0.0006,  0.0005,  ...,  0.0025,  0.0104, -0.0105],
        [-0.0119, -0.0022,  0.0155,  ...,  0.0001, -0.0015, -0.0073]],
       device='cuda:0', requires_grad=True)
```

نمودار یادگیری و تست B:

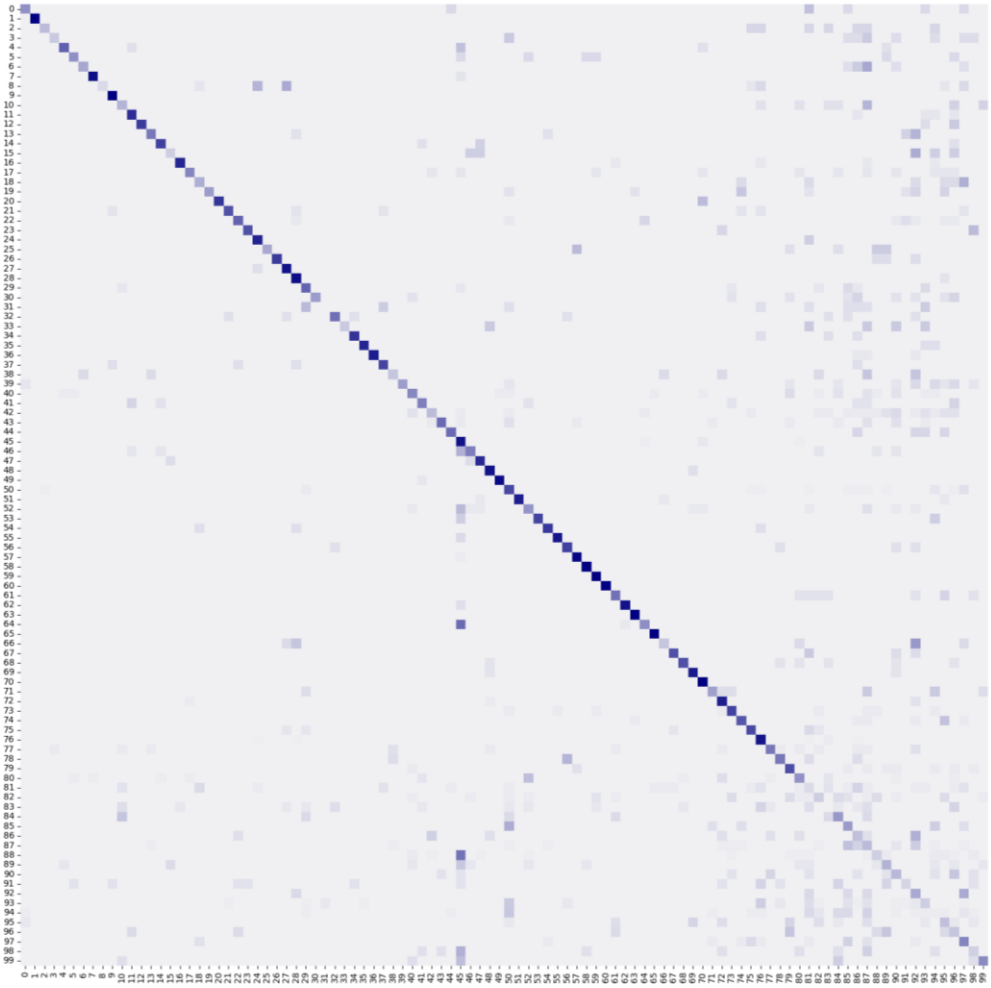


فقط 20 نورون لایه آخر آپدیت شده اند و وزن های اولیه و اکثر نورون های لایه آخر بر اساس A آموزش داده اند و دقت بر روی B پایین تر بوده.

نمودار test all:



ماتریس گمراهی برای ایپاک آخر:



همانطور که مشاهده میشود؛ چون لایه های اولیه و 80 نورون اول لایه آخر فریز شده اند، قطر اصلی در کلاس های A هم نسبت به حالت قبل پر رنگ تر بوده و عملکرد بهتری نسبت به تست همه داشته، زیرا وزن های لایه های اولیه و 80 نورون اولیه لایه آخر مطابق داده های A آموزش دیده اند.

علاوه بر این برای داده های کلاس B هم عملکرد نسبتا بدتری نسبت به حالت قبلی داشته زیرا فقط 20 نورون آخر لایه آخر برای کلاس های B آموزش دیده اند و قطر اصلی در سمت کلاس های B کمرنگ تر بوده.

قطر اصلی بیانگر تطابق لیبل واقعی با لیبل پیش بینی شده است

مقایسه نهایی:

روش سوم برای test all بهتر بوده چون داده های آموزشی برای B بسیار کم بوده و در کل آموزش خوبی برای B نمیتوان داشت ولی در روش سوم چون وزن های مربوط به آموزش A کامل فریز میشوند؛ مجددا میتوان آن ها را تا حدود خوبی طبقه بندی کرد و برای B هم نسبت به اینکه فقط 20 نورون آموزش دیده اند دقت مناسبی است.

روش اول برای test داده های B بهتر بوده ولی برای A چندان مناسب نیست چون شبکه مجددا مطابق با داده های B آموزش میبیند..

روش دوم هم نسبت به اول برای A بهتر بوده و نسبت به روش سوم برای B بهتر بوده.