به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

درس یادگیری عمیق با کاربرد در بینایی ماشین و پردازش صوت

تمرین شماره ۱

نام و نام خانوادگی : مهیار ملکی

شماره دانشجویی : ۸۱۰۱۰۰۴۷۶

اسفند ماه ۱۴۰۰

فهرست

مقدمه

در این تمرین سعی شده است که یک شبکه عصبی ساده بدون استفاده از کتابخانه های معمول پایتون برای کارهای ماشین لرنینگ مانند سایکیت لرن و پایتورچ و ... و فقط با استفاده از یک سری کتابخانه های ساده و ابتدایی مانند نامپای، پانداز، سیبورن و متپلاتلیب از صفر پیاده سازی شود. این پیاده سازی به گونه ای است که بتوان مقادیر مختلفی را برای پارامترهای شبکه عصبی امتحان کرد.

ساختار مدل درون فایل model.py قرار دارد که با فراخوانی آن داخل فایل main.py می توان شبکه را با داده های داده شده ترین کرد.

برای تعریف ساختار شبکه از دستور زیر می توان استفاده کرد:

```
import model
NN = model.NeuralNetwork(input_size, hidden_size1, hidden_size2, output_size, std)
```

* کد نوشته شده فقط قابلیت استفاده از یک یا دو لایه پنهان را دارد. همچنین دقت شود که برای تعریف شبکه ای با یک لایه پنهان باید hidden_size2=0 قرار داده شود.

در ادامه برای آموزش شبکه دستور زیر اجرا می شود:

```
import model
NN.fit(X_train, y_train, X_test, y_test, normalization, learning_rate, momentum, num_iters,
batch_size, verbose)
```

همانطور که قابل مشاهده است پارامتر های قابل تغییر در کد عبارتند از:

تعداد لایه های پنهان (۱ یا ۲) ، تعداد نورونهای هر لایه ، انحراف معیار وزنهای اولیه ، سرعت آموزش ، مومنتوم ، نرمال سازی ، تعداد ایپاک ها و اندازه بسته ها

کد قرار داده شده در فایل main بهترین پارامترهای بدست آمده را شامل می شود (پارامترهایی که در جدول قسمت هفتم سوال ۱ نیز نوشته شده اند)

سوال اول :

بخش ۱ :

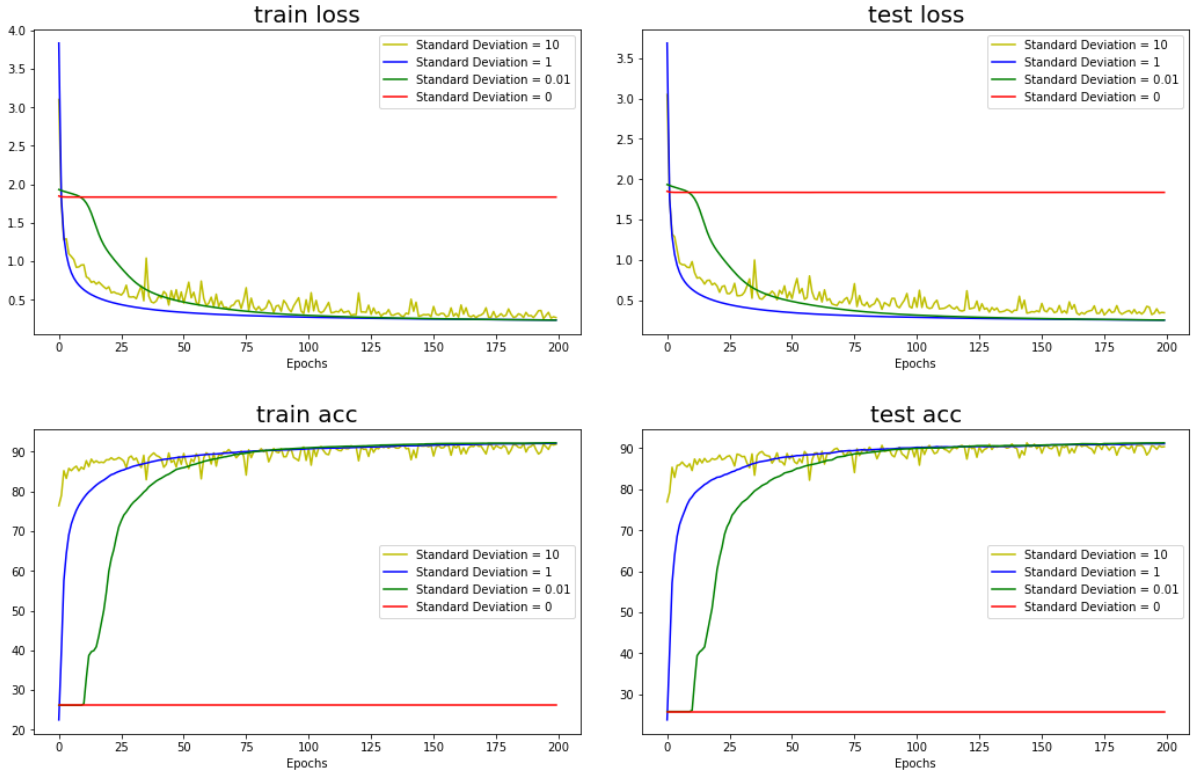
در این بخش تاثیر مقادیر مختلف انحراف معیار مقادیر وزنهای اولیه، در نتایج حاصل از شبکه عصبی بررسی شد.

در صورت عدم انجام Normalization فرایند آموزش شبکه دچار خطای اورفلو شده و همچنین نتایج نادرستی بدست می آید. لذا در اینجا برای اینکه به درستی بتوان تاثیر مقادیر مختلف انحراف معیار را بررسی کرد، از دادگان نرمال شده استفاده می کنیم. (تاثیر نرمال سازی داده ها را در بخش بعدی مقایسه خواهیم کرد)

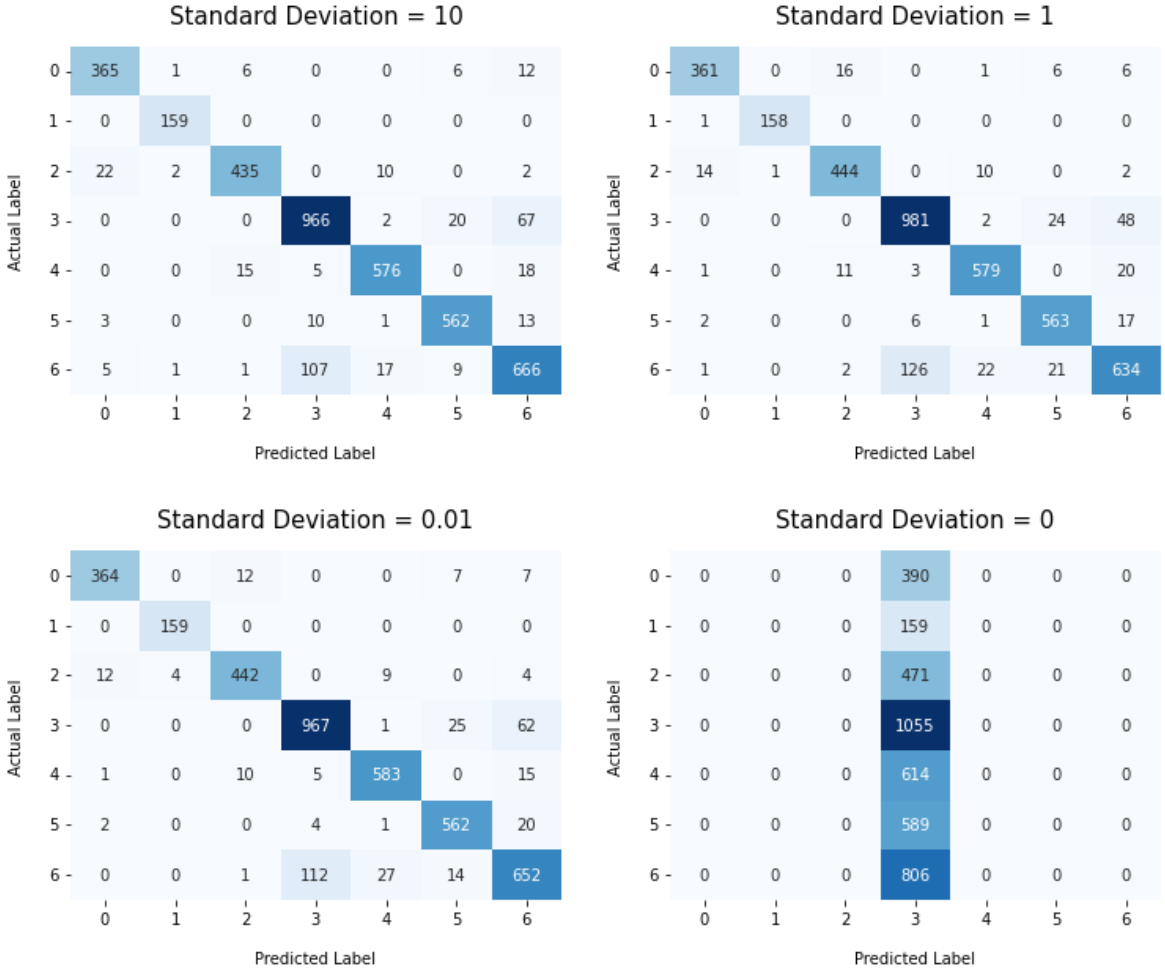
همانطور که در نمودارهای صفحه بعد مشاهده می شود، با انحراف معیار صفر عملا آموزشی صورت نگرفته و تمام داده های تست در کلاس ۳ طبقه بندی شده اند. همچنین در مقایسه مقادیر بزرگتر، با انحراف معیار ۱۰ نویز زیادی مشاهده می شود. با انحراف معیار های ۱ و ۰.۰۱ شبکه به خوبی به سمت نقطه بهینه همگرا می شود، همچنین به نظر می رسد با انحراف معیار ۱ این امر سریعتر اتفاق می افتد

لذا در این مرحله عدد ۱ را به عنوان بهترین مقدار انحراف معیار وزنهای اولیه از بین اعداد ۱۰ و ۱ و ۰.۰۱ و ۰ انتخاب می کنیم.

Standard Deviation	Test loss	Test Accuracy
0	1.834	25.83
0.01	0.252	91.30
1	0.251	91.08
10	0.343	90.37

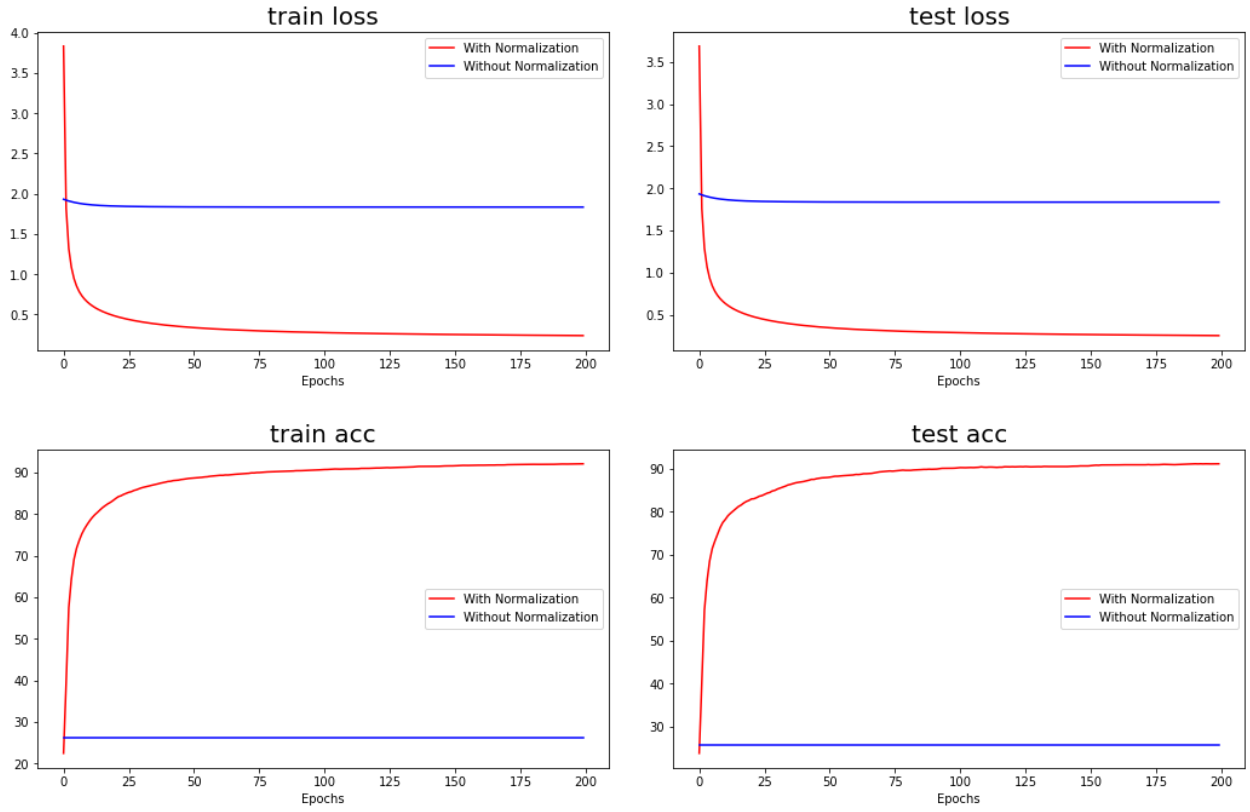


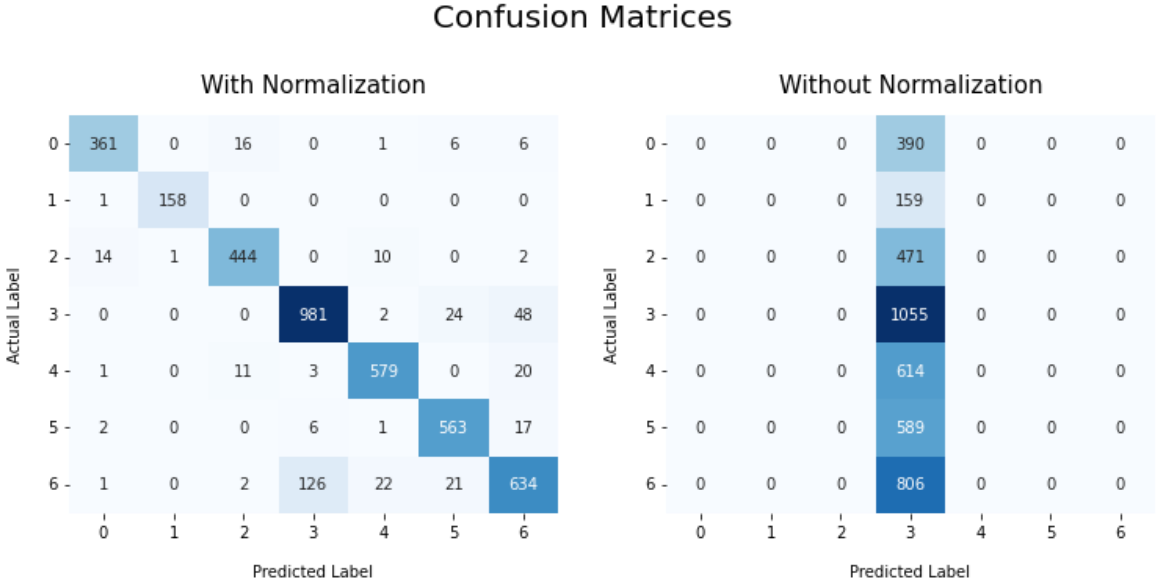
Confusion Matrices



بخش ۲ :

در این بخش تاثیر نرمال کردن یا نکردن داده ها بررسی می شود.
همانطور که در نمودارها مشخص است، با اعمال داده های نرمال نشده به شبکه مشاهده می شود که اصلا آموزشی صورت نگرفته است. این امر به دلیل تفاوت بسیار زیاد اسکیل ویژگی های مختلف داده ها است. برای مثال اسکیل برخی در محدوده 10^4 و اسکیل برخی از داده ها در محدوده 10^0 قرار گرفته است که این امر روند آموزش را مختل می کند.





بخش ۳ :

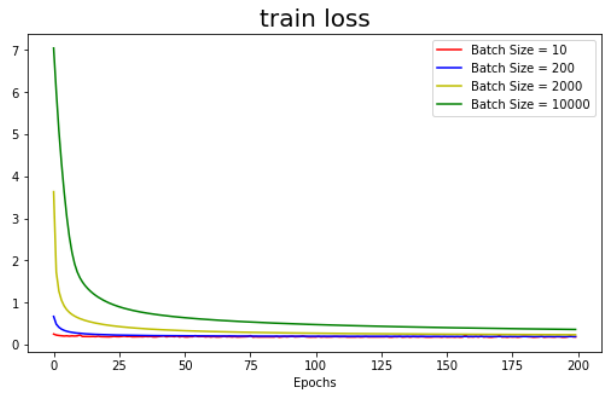
در اینجا به بررسی تاثیر اندازه های مختلف بسته ها در فرایند آموزش می پردازیم.

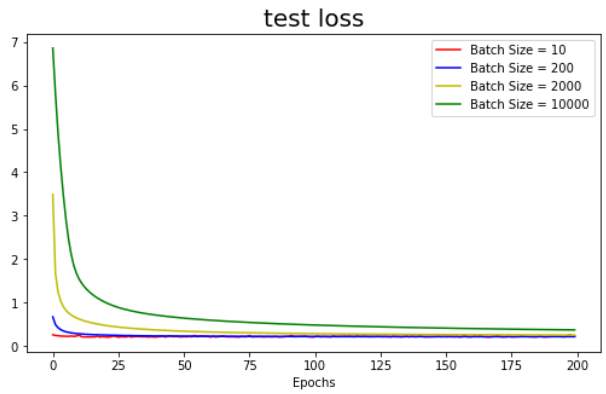
اندازه بسته ۱۰ : با کوچکتر شدن اندازه بسته ها در واقع به روش **stochastic gradient descent** نزدیک می شویم. این روش نویز زیادی دارد ولی سریع است زیرا عملیات آپدیت با مشاهده تک تک داده ها انجام می شود، لذا در اینجا مشاهده می شود که به سرعت در ایپاک های اولیه به دقت حداکثر می رسیم، اما نویز زیادی نیز مشاهده می شود.

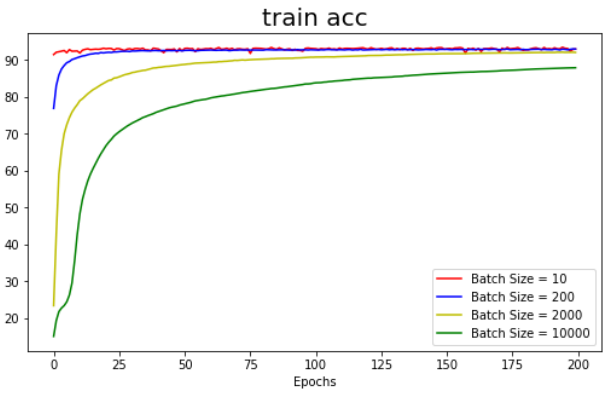
اندازه بسته ۱۰۰۰۰ : با بزرگتر شدن اندازه بسته در واقع به روش **batch gradient descent** می رسیم. این روش بسیار کند است زیرا در هر ایپاک تمام داده ها یکجا دیده می شوند. لذا همانطور که در نمودارهای صفحه بعد مشخص است با ۲۰۰ ایپاک همچنان از نقطه بهینه کمی دور هستیم و برای رسیدن به نقطه بهینه به ایپاک های بیشتری نیاز است که زمان زیادی میگیرد.

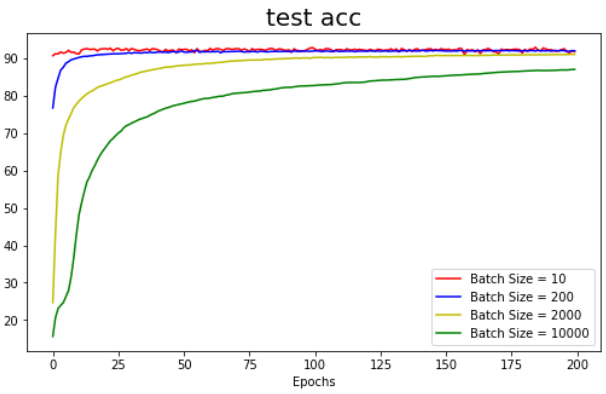
اندازه بسته ۲۰۰ و ۲۰۰۰ : مشاهده می شود که با هر دو اندازه بسته با سرعت مناسبی به نقطه بهینه می رسیم و همچنین نویزی نیز دیده نمی شود. بین این دو نیز اندازه بسته ۲۰۰ سرعت همگرایی بیشتری داشته و به دلیل بار محاسباتی کمتر مناسب تر است.

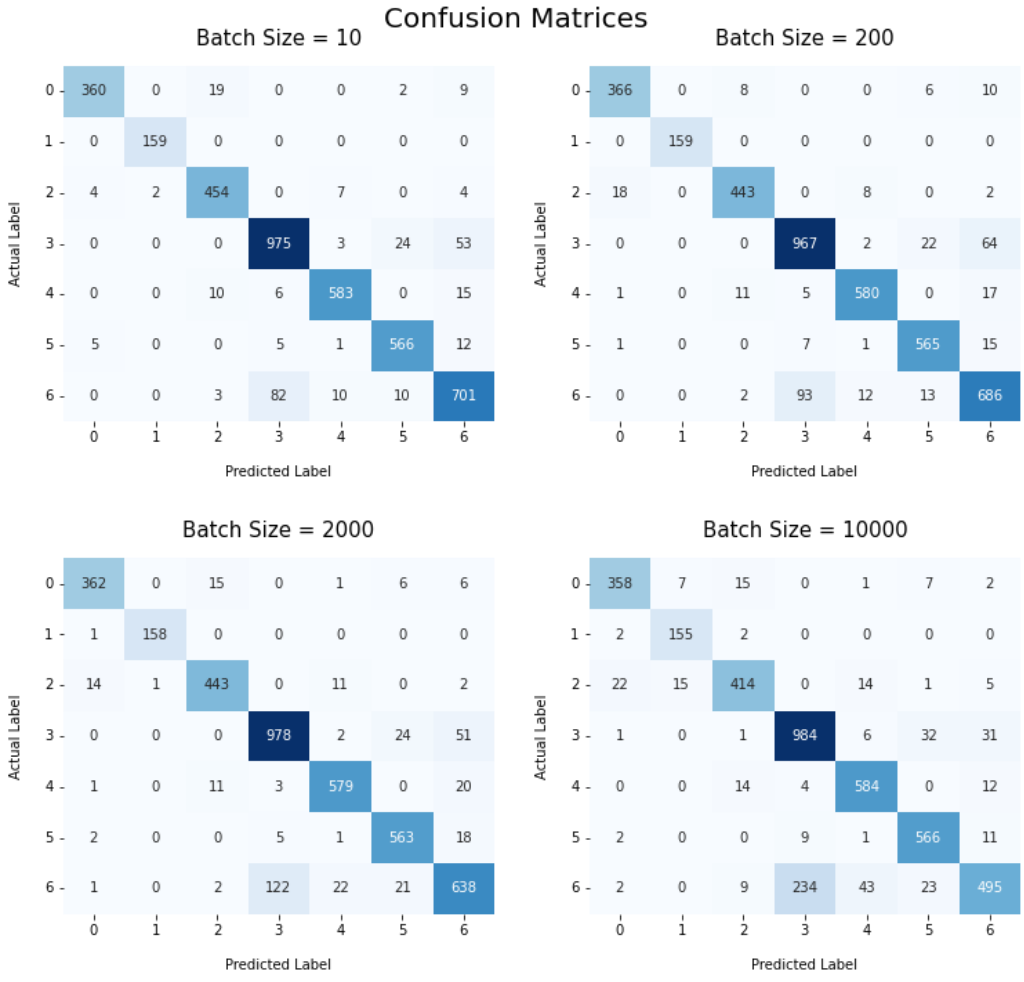
Batch Size	Test loss	Test Accuracy
10	0.224	91.84
200	0.214	92.01
2000	0.250	91.08
10000	0.369	87.07


train loss
Batch Size = 10
Batch Size = 200
Batch Size = 2000
Batch Size = 10000
Epochs


test loss
Batch Size = 10
Batch Size = 200
Batch Size = 2000
Batch Size = 10000
Epochs


train acc
Batch Size = 10
Batch Size = 200
Batch Size = 2000
Batch Size = 10000
Epochs


test acc
Batch Size = 10
Batch Size = 200
Batch Size = 2000
Batch Size = 10000
Epochs


Confusion Matrices
Batch Size = 10
Actual Label
Predicted Label
0 360 0 19 0 0 2 9
1 0 159 0 0 0 0 0
2 4 2 454 0 7 0 4
3 0 0 0 975 3 24 53
4 0 0 10 6 583 0 15
5 5 0 0 5 1 566 12
6 0 0 3 82 10 10 701
Batch Size = 200
Actual Label
Predicted Label
0 366 0 8 0 0 6 10
1 0 159 0 0 0 0 0
2 18 0 443 0 8 0 2
3 0 0 0 967 2 22 64
4 1 0 11 5 580 0 17
5 1 0 0 7 1 565 15
6 0 0 2 93 12 13 686
Batch Size = 2000
Actual Label
Predicted Label
0 362 0 15 0 1 6 6
1 1 158 0 0 0 0 0
2 14 1 443 0 11 0 2
3 0 0 0 978 2 24 51
4 1 0 11 3 579 0 20
5 2 0 0 5 1 563 18
6 1 0 2 122 22 21 638
Batch Size = 10000
Actual Label
Predicted Label
0 358 7 15 0 1 7 2
1 2 155 2 0 0 0 0
2 22 15 414 0 14 1 5
3 1 0 1 984 6 32 31
4 0 0 14 4 584 0 12
5 2 0 0 9 1 566 11
6 2 0 9 234 43 23 495

بخش ۴ :

در این بخش به بررسی تاثیر تعداد لایه های مخفی و تعداد نورون ها می پردازیم.

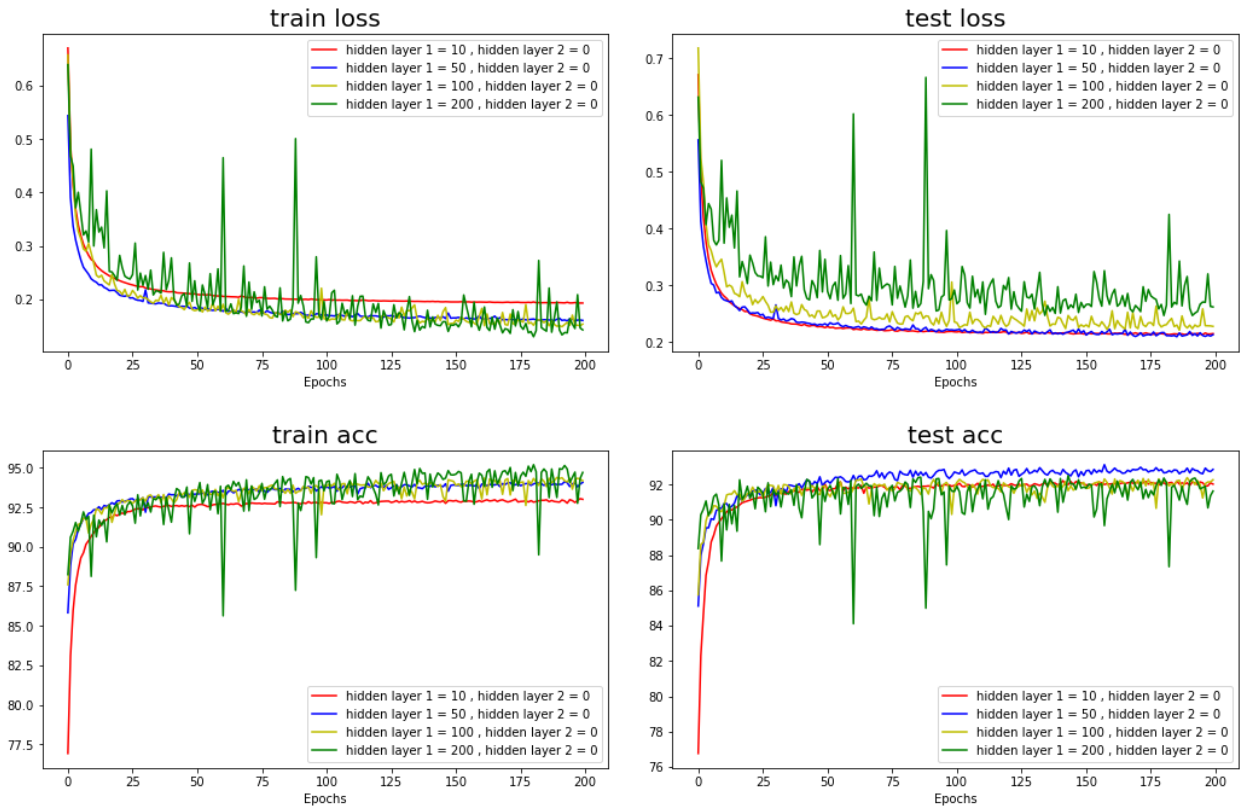
یک لایه مخفی :

ابتدا تاثیر تعداد نورون ها در اولین لایه مخفی را بررسی می کنیم.

همانطور که از نمودارها پیداست، افزایش تعداد نورونها در لایه اول موجب افزایش نویز و همچنین وقوع overfit شده است. البته با کاهش سرعت آموزش ممکن است نویز زیادی که بوجود آمده است از بین برود ولی همچنان مسئله overfit وجود خواهد داشت.

با بررسی نتایج بدست آمده تعداد نورونهای ۵۰ عدد برای لایه اول انتخاب می شود

First Hidden Layer Size	Test loss	Test Accuracy
10	0.214	92.01
50	0.212	92.85
10	0.227	92.26
100	0.261	91.62

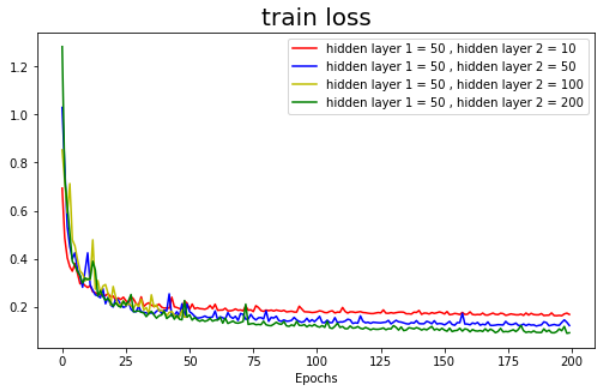


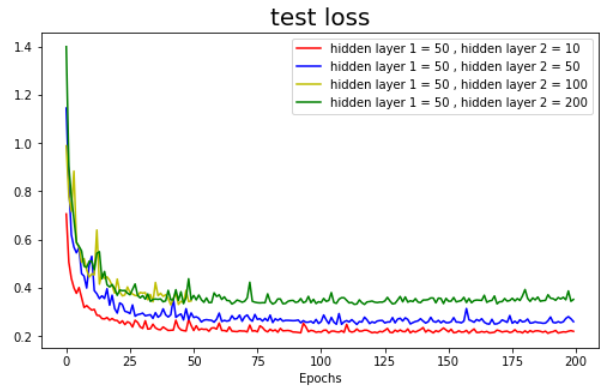
دو لایه مخفی :

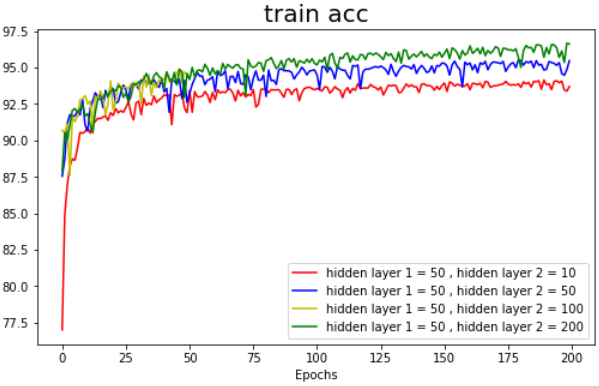
پس از انتخاب تعداد نورونهای لایه اول در ادامه تاثیر تعداد نورون ها در دومین لایه مخفی را بررسی می کنیم.
همانطور که از نمودارها پیداست، همچنان نویز زیادی مشاهده می شود. علاوه بر آن اضافه کردن لایه پنهان دوم و افزایش تعداد نورون های آن تغییر معناداری در نتایج حاصله ایجاد نکرده و حتی موجب کاهش دقت داده های تست نیز شده است.

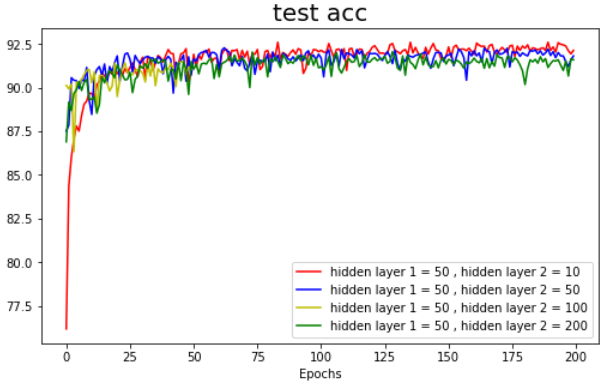
لذا در این بخش شبکه ای با ساختار یک لایه مخفی که شامل ۵۰ نورون می باشد را که در مرحله قبل بدست آوردیم، انتخاب می کنیم.

First Hidden Layer Size	Second Hidden Layer Size	Test loss	Test Accuracy
50	10	0.219	92.14
50	50	0.259	91.82
50	100	0.345	91.06
50	200	0.351	91.60



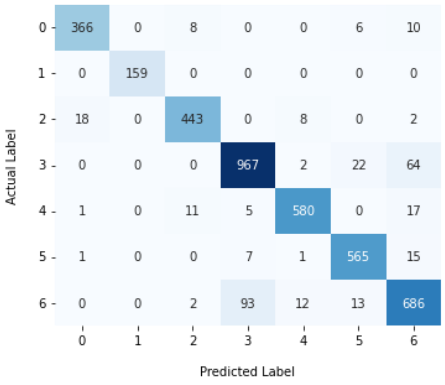






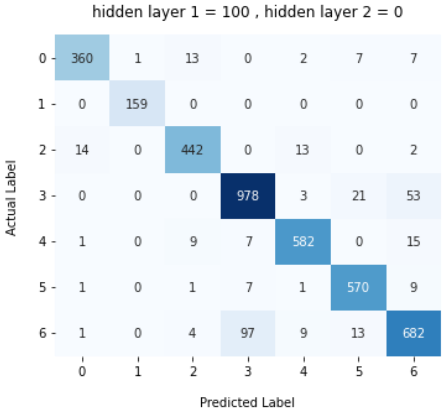
Confusion Matrices

hidden layer 1 = 10 , hidden layer 2 = 0

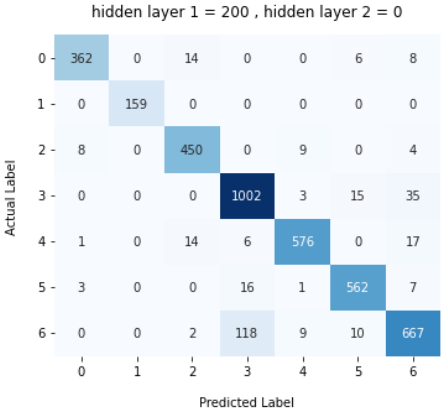


hidden layer 1 = 50 , hidden layer 2 = 0

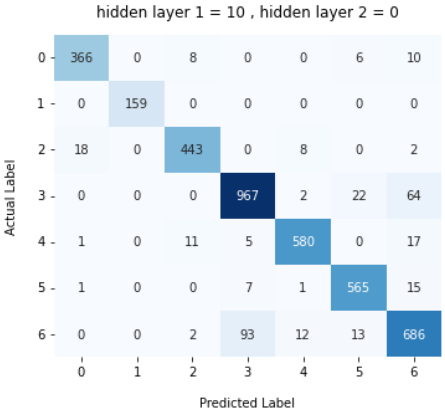
hidden layer 1 = 100 , hidden layer 2 = 0



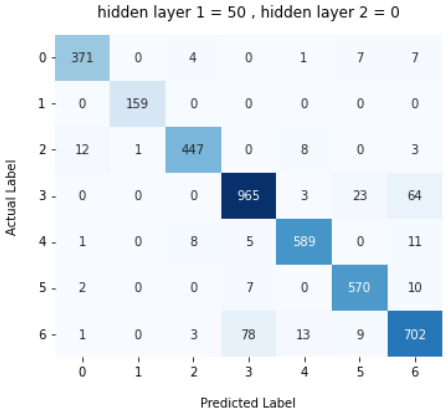
hidden layer 1 = 200 , hidden layer 2 = 0



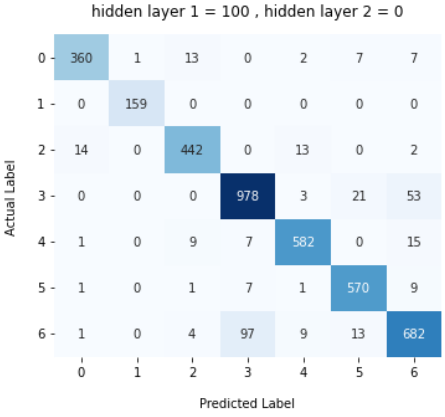
hidden layer 1 = 10 , hidden layer 2 = 0



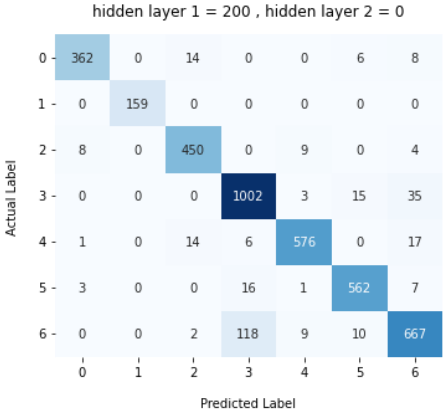
hidden layer 1 = 50 , hidden layer 2 = 0



hidden layer 1 = 100 , hidden layer 2 = 0



hidden layer 1 = 200 , hidden layer 2 = 0



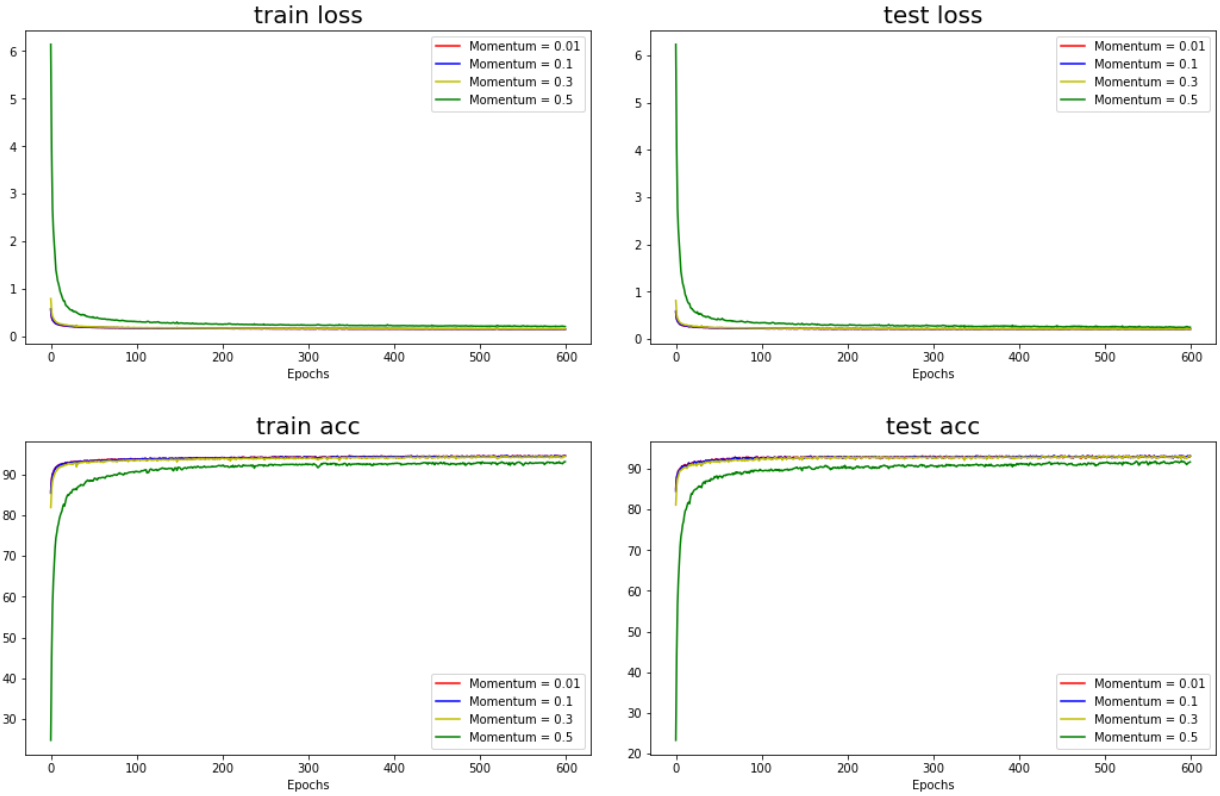
بخش ۵ :

در بخش پنجم به بررسی تاثیر افزودن مومنتوم (momentum) به الگوریتم کاهش گرادیان می پردازیم. مومنتوم در واقع مانند نوعی حافظه عمل کرده و تغییرات قبلی پارامترها را در عملیات فعلی آپدیت آنها تاثیر می دهد تا همگرایی آنها به نقطه بهینه سریعتر و با دقت بیشتری صورت گیرد.

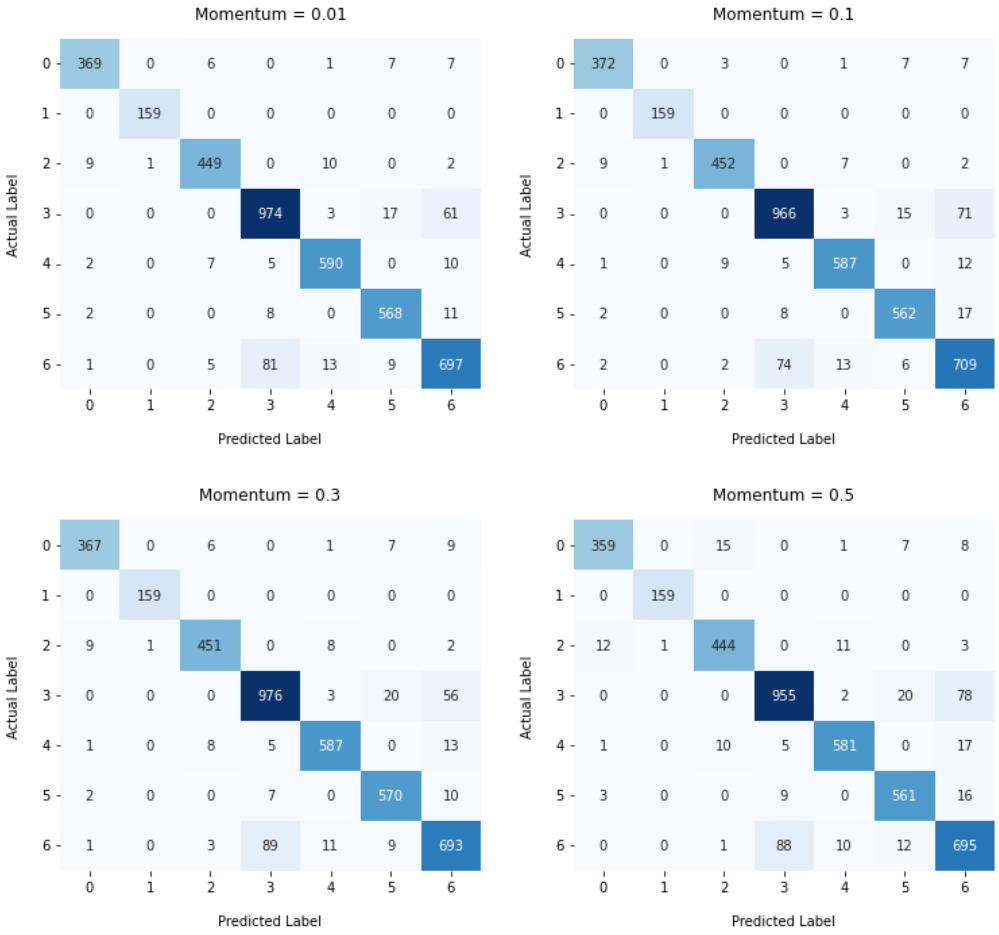
همانطور که مشاهده می شود با افزودن مومنتوم و همچنین کاهش آن به عدد ۰.۰۱ شبکه ما در کمتر از ۱۰۰ ایپاک به نقطه بهینه و دقت خوب ۹۳ درصد برای داده های تست می رسد.

```
for iteration in range(num_iters):

    dW2 = 0
    db2 = 0
    dW1 = 0
    db1 = 0

    for batch in range(num_batches):

        start = bch * batch_size
        end = (bch+1) * batch_size
        X_batch, y_batch = X[start:end], y[start:end]
        loss  = self.forward(X_batch, y_batch)
        grads = self.backpropagation(X_batch, y_batch)

        # updating_weights
        W2 += - learning_rate * grads['W2'] + momentum * dW2
        b2 += - learning_rate * grads['b2'] + momentum * db2
        W1 += - learning_rate * grads['W1'] + momentum * dW1
        b1 += - learning_rate * grads['b1'] + momentum * db1

        # last_weight_changes_(for_momentum)
        dW2 = learning_rate * grads['W2'] +  momentum * dW2
        db2 = learning_rate * grads['b2'] +  momentum * db2
        dW1 = learning_rate * grads['W1'] +  momentum * dW1
        db1 = learning_rate * grads['b1'] +  momentum * db1
```

Momentum	Test loss	Test Accuracy
0.01	0.209	93.09
0.1	0.208	93.07
0.3	0.207	93.02
0.5	0.246	91.62



Confusion Matrices



بخش ۶ :

بخش ۷ :

در بخش اول مشاهده شد که انتخاب یک انحراف معیار مناسب برای وزن های اولیه اهمیت زیادی دارد، مقادیر خیلی بزرگ نویز زیادی را ایجاد کرده و مقادیر خیلی کوچک میتوانند سرعت همگرایی به بهینه را کم کنند، لذا مقدار ۱ در این قسمت انتخاب شد.

در بخش دوم مشاهده شد که به دلیل اسکیل متفاوت داده ها، وقتی نرمالیزیشن صورت نمیگیرد عملا شبکه توانایی آموزش را ندارد.

در بخش سوم تاثیر اندازه بسته ها در سرعت همگرایی و زمان اجرای کد و نویز حاصله مشاهده شد و نحوه عملکرد الگوریتم **mini-batch** نیز رصد شد.

در بخش چهارم مشاهده شد که لزوما تعداد لایه ها و نورون های بیشتر خوب نیست و علاوه بر ایجاد نویز باعث **overfit** شدن می شود لذا شبکه ای با یک لایه پنهان و ۵۰ نورون در این قسمت انتخاب شد.

در بخش پنجم تاثیر اضافه کردن ترم مومنتوم به الگوریتم گرادیان دیسنت مشاهده شد که چگونه سرعت همگرای شبکه به نقطه بهینه را افزایش می دهد.

پارامترهای نهایی بدست آمده برای شبکه در جدول صفحه بعد قابل مشاهده است.

مشخصات بهترین شبکه	
• تعداد لایه پنهان	1
• تعداد نورون های ورودی	16
• تعداد نورون های لایه پنهان	50
• تعداد نورون های خروجی	7
• انحراف معیار وزنهای اولیه	1
• سرعت آموزش	0.1
• مومنتوم	0.01
• تعداد ایپاک ها	600
• اندازه بسته ها	200
• تابع فعال ساز	ReLU + Softmax
• تابع هزینه	Cross Entropy
• بهینه ساز	Mini-Batch
• تعداد داده های آموزش	9527
• تعداد داده های تست	4084
• دقت بدست آمده برای داده های آموزش	94.56 %
• دقت بدست آمده برای داده های تست	93.09 %

سوال دوم :