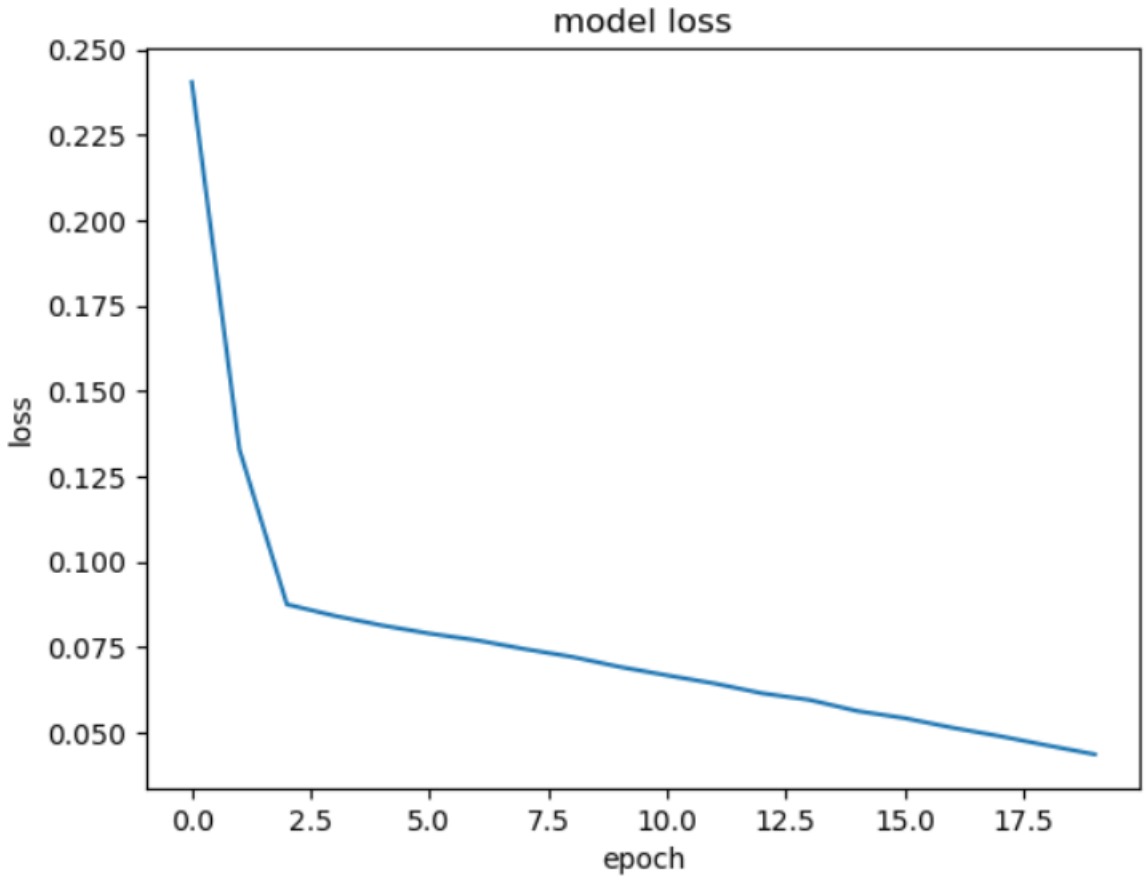
قدم دوم: پس از 5 بار اجرای کد، دقت‌ها بدین ترتیب است:

```
0.11
0.13
0.07
0.15
0.09
```

به طور میانگین حدود 10 درصد است که مورد انتظار ما بود.

قدم سوم: دقت و زمان اجرا و پلات برای epoch 20 در ادامه آمده است:

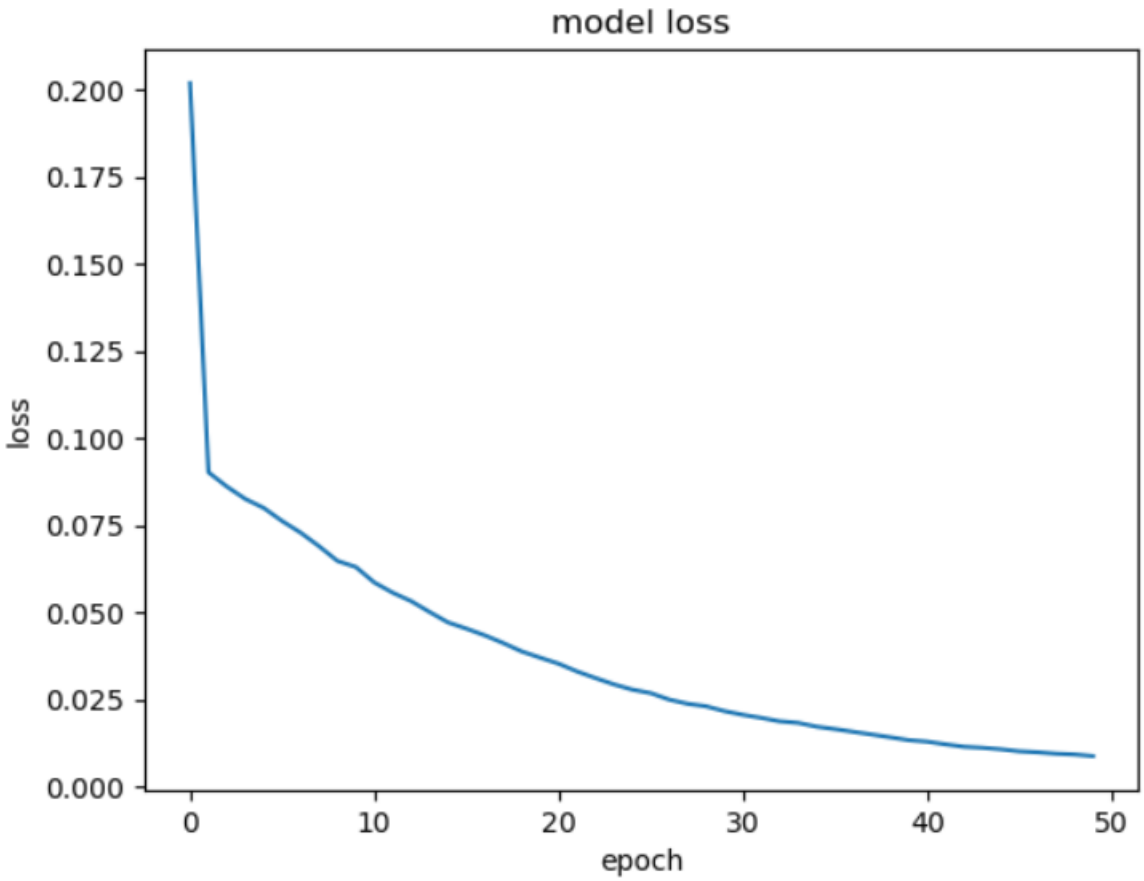
```
epoch 0: 0.08
epoch 1: 0.1
epoch 2: 0.23
epoch 3: 0.25
epoch 4: 0.29
epoch 5: 0.35
epoch 6: 0.34
epoch 7: 0.43
epoch 8: 0.49
epoch 9: 0.47
epoch 10: 0.56
epoch 11: 0.62
epoch 12: 0.63
epoch 13: 0.72
epoch 14: 0.67
epoch 15: 0.78
epoch 16: 0.76
epoch 17: 0.8
epoch 18: 0.81
epoch 19: 0.86
```




```
Wall time: 8min 50s
```

برای 50 epoch هم در ادامه آمده است:

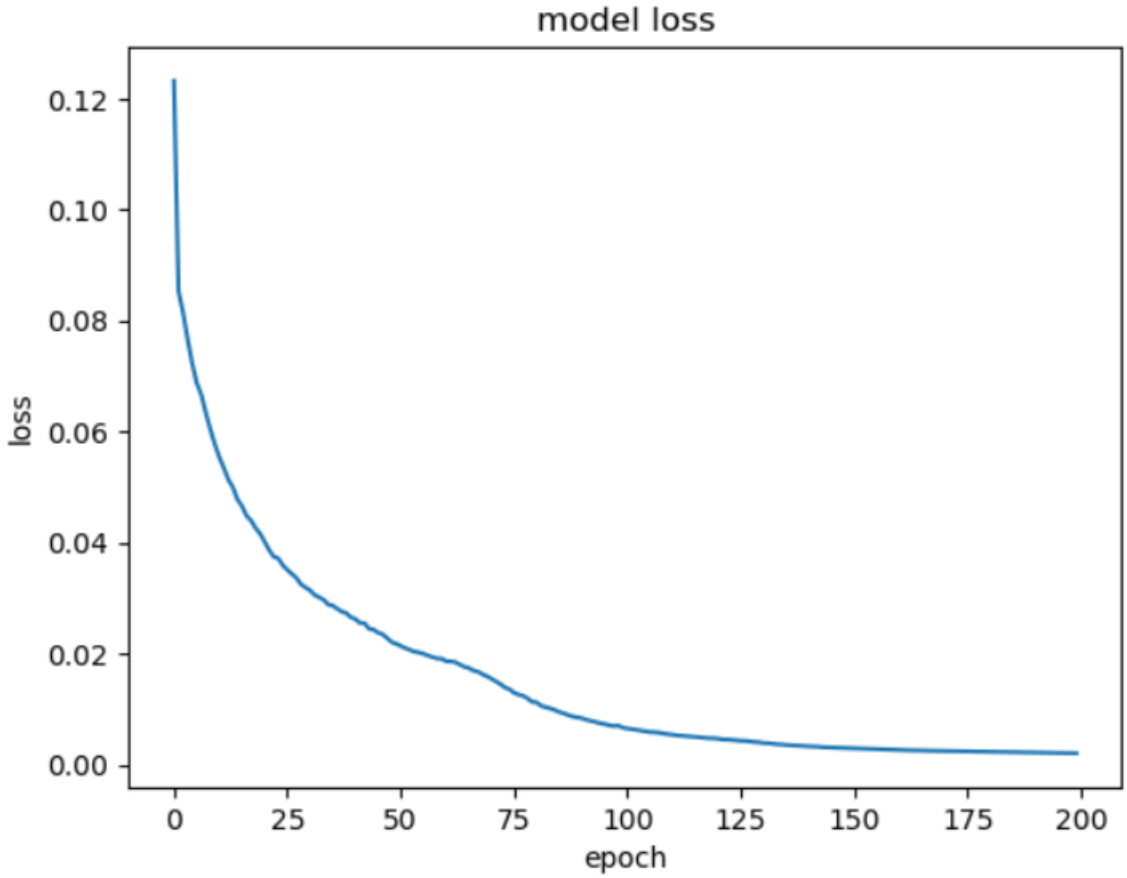
```
epoch 0: 0.1
epoch 1: 0.23
epoch 2: 0.22
epoch 3: 0.27
epoch 4: 0.3
epoch 5: 0.33
epoch 6: 0.42
epoch 7: 0.43
epoch 8: 0.46
epoch 9: 0.44
epoch 10: 0.46
epoch 11: 0.47
epoch 12: 0.48
epoch 13: 0.5
epoch 14: 0.49
epoch 15: 0.55
epoch 16: 0.53
epoch 17: 0.55
epoch 18: 0.57
epoch 19: 0.59
epoch 20: 0.62
epoch 21: 0.61
epoch 22: 0.62
epoch 23: 0.62
epoch 24: 0.63
epoch 25: 0.62
epoch 26: 0.65
epoch 27: 0.64
epoch 28: 0.67
epoch 29: 0.68
epoch 30: 0.69
epoch 31: 0.68
epoch 32: 0.7
epoch 33: 0.72
epoch 34: 0.76
epoch 35: 0.77
epoch 36: 0.76
epoch 37: 0.81
epoch 38: 0.82
epoch 39: 0.84
epoch 40: 0.86
epoch 41: 0.87
epoch 42: 0.86
epoch 43: 0.88
epoch 44: 0.86
epoch 45: 0.87
epoch 46: 0.89
epoch 47: 0.87
epoch 48: 0.88
epoch 49: 0.89
```




```
Wall time: 22min 42s
```

قدم چهارم:

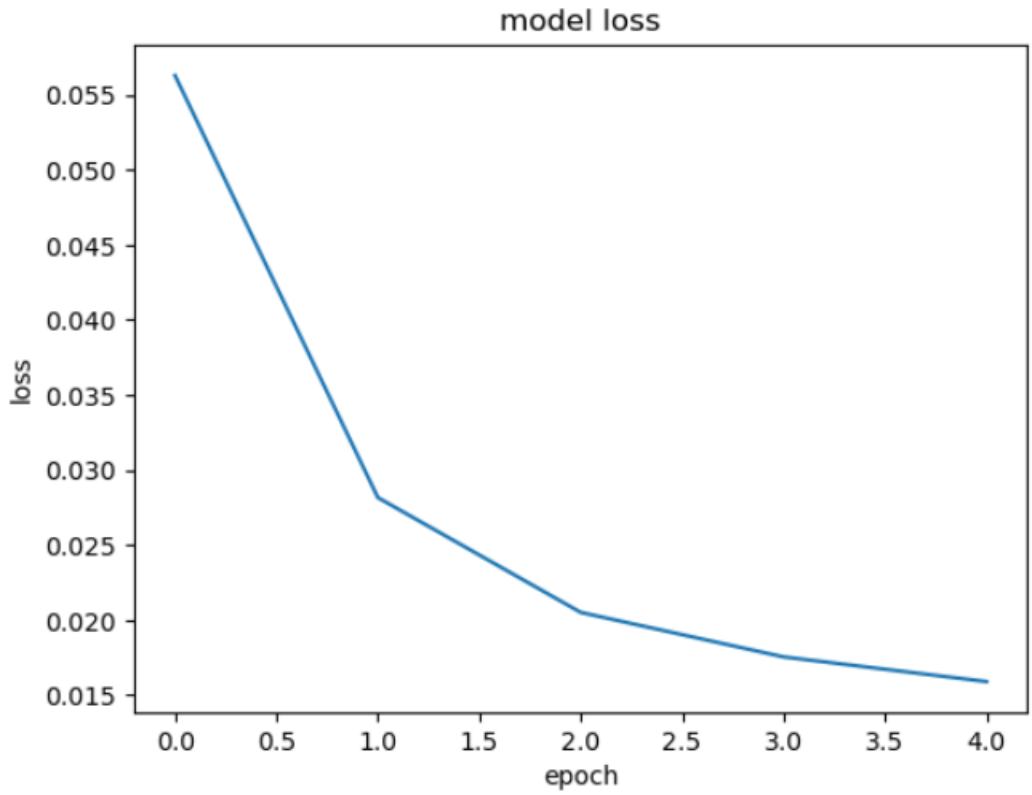
```
epoch 199: 0.99
```




```
Wall time: 5.27 s
```

قدم پنجم: برای داده‌های train:

```
epoch 0: 0.5794
epoch 1: 0.8183833333333334
epoch 2: 0.8705166666666667
epoch 3: 0.8898333333333334
epoch 4: 0.9000666666666667
```



دقت برای داده‌های تست:

```
0.9021
```

امتیازی:

سوال اول: با کمک داده‌های اعتبارسنجی می‌توانیم با متغیرهایی مانند تعداد نورون‌ها و هایپرپارامترها بازی کنیم تا شبکه ما هرچه بیشتر بهینه شود و تشخیصش قوی‌تر شود و فرق آن با داده تست این است که با داده تست اجازه این گونه تغییرات را نداریم و کاربردش تنها دادن معیاری به ماست تا بدانیم مدل ما به چه میزان دقت عمل دارد.

سوال دوم: در گرادیان کاهشی تصادفی پس از هر نمونه، مقادیر وزن‌ها و بایاس‌ها را آپدیت می‌کنیم ولی در mini-batch این کار پس از تعدادی مشخص از داده‌ها صورت می‌گیرد (معدل گرادیان آنها محاسبه می‌شود). در batch هم پس از feed forward همه داده‌های یک epoch معدل گرادیان تمام آنها حساب می‌شود و سپس مقادیر وزن‌ها و بایاس‌ها آپدیت می‌شود. روش batch محاسبه پر هزینه‌ای دارد ولی جواب بهینه می‌دهد. روش تصادفی محاسبه ساده‌ای دارد ولی لزوما جواب بهینه نمی‌دهد. روش mini-batch هم نقطه‌ای بین این دو است که هم جواب به نسبت قوی و بهینه‌ای می‌دهد و هم زمان به نسبت کوتاهی برای آموزش دارد ولی پیاده‌سازی پیچیده‌ای دارد.

سوال سوم: این تکنیک سال 2015 مطرح شد و با حل کردن مشکل internal covariate shift باعث تثبیت و سریع‌تر شدن آموزش می‌شود. این مشکل باعث می‌شود نرخ آموزش مطلوب شبکه ما به مقادیر رندوم اولیه لایه ورودی بسیار وابسطه شود و بنابراین به یک مقدار همگرا نمی‌شود. همین باعث می‌شود گرادیان ما بسیار کند همگرا شود و خیلی دیر به یک مقدار بهینه برای شبکه برسیم. در یک نگاه کلی علت بهتر شدن عملکرد شبکه ما با کمک این تکنیک، مشابه علت بهتر شدن عملکرد شبکه ما با استفاده از تکنیک نرمال کردن لایه ورودی است.

سوال چهارم: وظیفه لایه pooling کاهش ابعاد است و این کار باعث کاهش تعداد متغیرها و درنتیجه پیچیدگی محاسبه می‌شود. معروف‌ترین روش‌های مورد استفاده آن مکس و میانگین گیری است. علت این ارجحیت این است که چون این معماری از فیلتر استفاده می‌کند و فیلتر در یک ناحیه از عکس آموزش می‌بیند، همین امر باعث می‌شود که شبکه ما خیلی بهتر قادر به یادگیری pattern های تکراری مثل خط مورب یا دایره و... شود (وزن‌ها و بایاس در یک فیلتر در طول یک آموزش یک متغیر یکسان اند و مانند شبکه عصبی هر ارتباطی یک وزن جدا ندارد).