گزارش ۳: پروژه‌ی اختیاری داده‌کاوی

نیکا شهابی ۹۷۱۳۰۲۳، نیلوفر جعفری ۹۷۱۳۰۱۱

دکتر حاجی‌محمدی

تیر ۱۴۰۱

تعریف صورت مسئله:

مسئله‌ی classification داده‌های دیتاست MNIST را با استفاده از یک deep neural network و SVM حل کنید.

آماده سازی دیتا:

بسیاری از آمده سازی ها قبلا روی این دیتاست انجام شده است. ما ابتدا برای آشنایی با دیتاست، تعداد instance ها و shape ورودی و خروجی را درمی‌اوریم وبرای datapoint اول training set، ایکس و y را چاپ میکنیم.
برای شبکه عصبی نیاز داریم که ایکس اولیه را به صورت بردار تک بعدی درآوریم بنابرین به اصطلاح ایکس های ۲۸*۲۸ را flatten میکنیم.

توضیح راه حل ها:

۱. یک شبکه‌ی عصبی fully connected feed forward طراحی می‌کنیم. توضیحات این طراحی در notebook و در بین کد آورده شده است. (توضیحاتی از قبیل تعداد لایه‌ها، optimizer انتخاب شده، تعداد node ها در هر لایه، activation function استفاده شده در هر لایه و دلیل استفاده از آن و غیره)
مدل‌های مختلف با تعداد hidden layer های مختلف را آموزش می‌دهیم و نتیجه‌ی آموزش آن‌ها و نتیجه‌ی عملکرد آن‌ها روی داده‌ی تست را با هم مقایسه میکنیم.

۲. از آنجا که x ها، یک عکس هستند میتوان از تکنیک‌های لایه‌های استفاده شده در شبکه عصبی CNN برای ساخت مدل استفاده کرد. یعنی CNN طراحی میکنیم. توضیحات معماری آن در بین کد.

مدل های ۱ و ۲ در MNIST_with_NN.ipynb
مدل ۳ در MNIST_with_SVM.ipynb

۳. استفاده از SVM برای حل این مسئله: از آنجا که در عنوان اسم SVM آورده شده، یک مدل SVM هم طراحی و train میکنیم.
تمام توضیحات SVM در نوت بوک مربوطه آورده شده است.

توضیح کد:

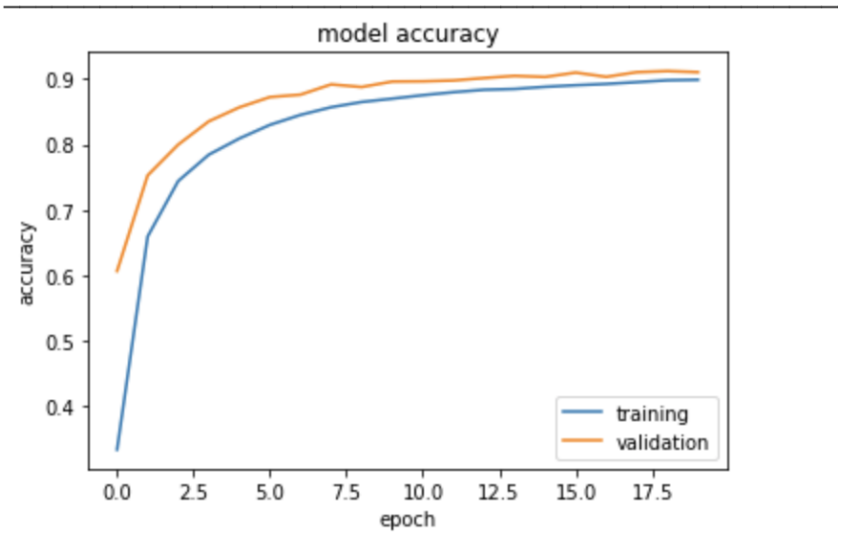
به صورت کامنت در بین کد

مقایسه ی روش ها و نمونه خروجی با دیفالت های گفته شده:

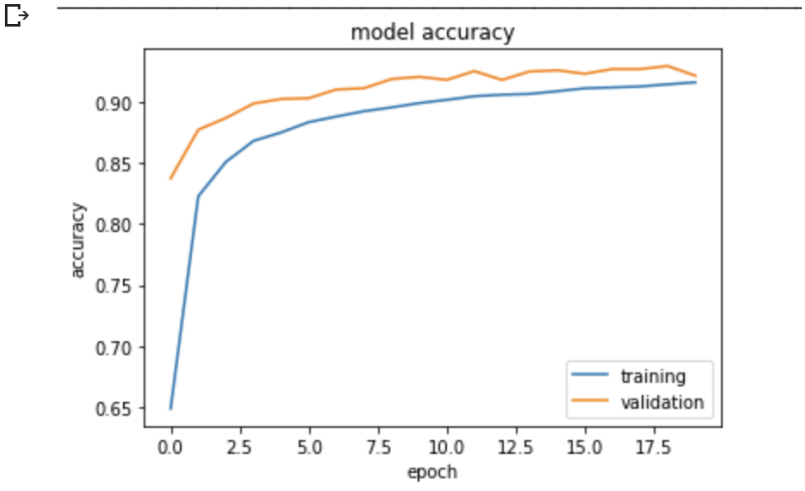
مقایسه‌ی مدل‌های مورد ۱:

نمودار های زیر مربوط به number of epochs = 20 و batch size = 128 است.

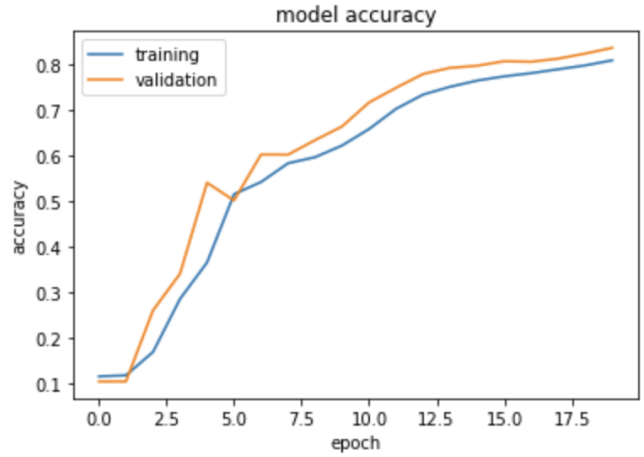
نمودار ها به ترتیب مربوط به nn ای با ۱، ۲ و ۳ hidden layer هستند.




```
Test loss: 0.411
Test accuracy: 0.898
Model: "sequential_9"
```



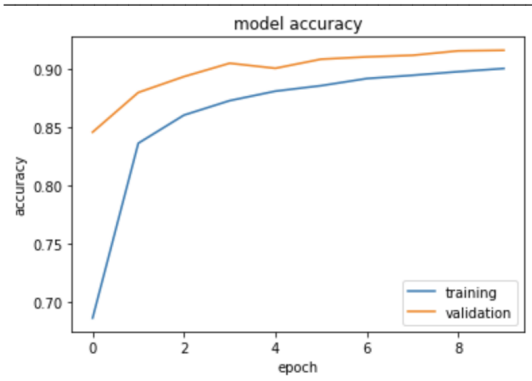

```
Test loss: 0.315
Test accuracy: 0.914
Model: "sequential_8"
```




```
Test loss: 0.917
Test accuracy: 0.823
```

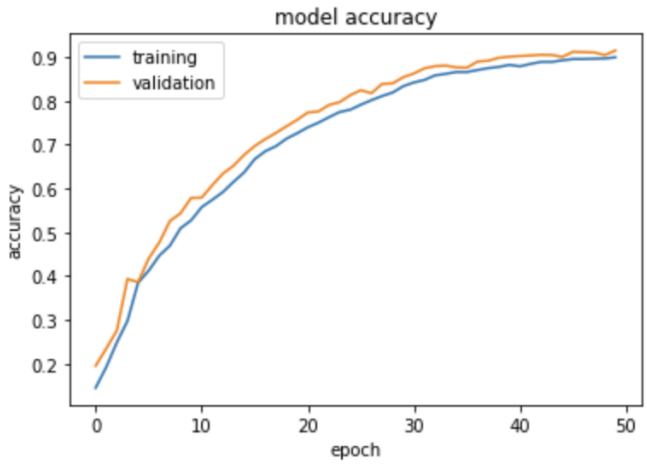
تحلیل: مدل‌های اول و دوم بعد حدود ۱۰ epoch دچار overfit شده‌اند. دقت train آنها هنوز بالا میرفته اما دقت ولیدیشن یا تغییری نکرده یا نواسانی بالا و پایین شده است. اما مدل سوم از آنجا که دچار پیچیدگی بیشتری در معماری خود بوده بعد ۱۰ تا epoch دچار overfit نشده و هنوز جای train کردن داشته است. دقت مدل‌ها روی داده‌ی تست در مدل دوم بیشتر از مدل اول است اما دقت از مدل ۲ به ۳ پایین آمده است. که به این خاطر است که مدل ۳ هنوز جای ترین شدن داشته است. بنابرین از این نمودارها میفهیم که مدل ۱ را تا حدود همان ۱۰ epoch باید نگه میداشتیم و مدل ۳ را باید بیشتر از ۲۰ تا میکردیم.

مدل ۱ بعد ۱۰ epoch: میبینیم که دقت آن روی داده‌ی تست بیشتر از زمانی شد که تا ۲۰ epoch train کرده بودیم:




```
Test loss: 0.38
Test accuracy: 0.905
Model: "sequential_11"
```

مدل ۳ بعد ۵۰ epoch: میبینیم که دقت آن روی داده‌ی تست بیشتر از زمانی شد که تا ۲۰ epoch train کرده بودیم و تازه در اپوک های اخر دارد دچار اورفیت میشود.




```
Test loss: 0.4
Test accuracy: 0.901
```

با توجه به اینکه دقت مدل ۲ و ۳ روی داده‌ی تست مشابه بود متوجه می‌شویم که مقدار feature های قابل extract توسط این دیتاست با همان معماری کوچک‌تر قابل بیرون کشیدن هستند و پیچیده‌تر کردن مدل به ما کمکی نکرده است.

تحلیل مدل ۲ و مقایسه‌ی آن با مدل ۱:

بعد ۴ اپوک دیگر نیازی به train نبوده و overfit شده است.

دقت این مدل با توجه به اینکه از تکنیک‌های جدید مربوط به عکس استفاده شده و پیچ و خم های عکس شناسایی شده است از همه‌ی مدل های مرحله‌ی قبل به طور قابل درکی بالاتر است.

```
#training the model
model.fit(xtrain,y_train,batch_size=100,epochs=5,validation_data=(xtest,y_test))
```

```
Epoch 1/5
600/600 [==============================] - 13s 7ms/step - loss: 0.1318 - accuracy: 0.9590 - val_loss: 0.0454 - val_accuracy: 0.9840
Epoch 2/5
600/600 [==============================] - 4s 6ms/step - loss: 0.0394 - accuracy: 0.9876 - val_loss: 0.0301 - val_accuracy: 0.9911
Epoch 3/5
600/600 [==============================] - 4s 6ms/step - loss: 0.0261 - accuracy: 0.9918 - val_loss: 0.0235 - val_accuracy: 0.9909
Epoch 4/5
600/600 [==============================] - 4s 6ms/step - loss: 0.0192 - accuracy: 0.9940 - val_loss: 0.0263 - val_accuracy: 0.9918
Epoch 5/5
600/600 [==============================] - 4s 6ms/step - loss: 0.0154 - accuracy: 0.9950 - val_loss: 0.0364 - val_accuracy: 0.9887
<keras.callbacks.History at 0x7ffab1699410>
```

```
[21] #model train and test scores
model.evaluate(xtrain,y_train),model.evaluate(xtest,y_test)
```

```
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0222 - accuracy: 0.9929
313/313 [==============================] - 1s 4ms/step - loss: 0.0364 - accuracy: 0.9887
([0.02224162593483925, 0.9929166436195374],
 [0.03642405942082405, 0.9886999726295471])
```

تحلیل مدل ۳ و مقایسه‌ی آن با بقیه:

```
predicted = svc.predict(x_test)
print(classification_report(y_test,predicted))
```

```
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       980
           1       0.96      0.98      0.97      1135
           2       0.93      0.88      0.91      1032
           3       0.90      0.91      0.90      1010
           4       0.91      0.93      0.92       982
           5       0.89      0.86      0.87       892
           6       0.93      0.95      0.94       958
           7       0.92      0.92      0.92      1028
           8       0.87      0.86      0.87       974
           9       0.89      0.89      0.89      1009

    accuracy                           0.92     10000
   macro avg       0.92      0.91      0.91     10000
weighted avg       0.92      0.92      0.92     10000
```

به طور تقریبی میتوان گفت SVM و NN fully connected deep ها مثل هم عمل کرده‌اند و CNN از هر دوی آنها بهتر عمل کرده است.

منابع:

در آخر نوت بوکها آورده شده است.