گزارش پروژه نهایی درس مقدمه ای بر یادگیری ماشین

استاد درس: خانم دکتر سیدین

دانشجو:محمدرضا کریمی

۹۶۲۳۰۹۳

هدف در این پروژه این است که دیتاست داده شده را با استفاده از روش MLPوCNNآموزش دهیم و نتایج مربوط به هر دو روش را مقایسه کنیم.

در ابتدای این پروژه باید دیتاست داده شده را بخوانیم که این کار با استفاده از تابع load_ dataداده شده انجام می گیرد از طریق این تابع داد گان آموزش و تست را دریافت می کنیم بعد از دریافت داده ها مشاهده می کنیم که داد گان آموزش ابعاد (۴۵۰۰۰,۷۸۴)داد گان تست (۱۰۰۰,۷۸٤) ودادگان مربوط به Validation ابعاد(۱۵۰۰۰,۷۸۴) را دراختیار دارد. برای تقسیم بندی داده ها از کتابخانه Sklearnو از ویژگی Train_test_splitمربوط به آن استفاده نمودیم ودر اینجا Test sizeرا مساوی با۰.۲۵ قرار دادیم و ۰.۲۵دادگان را به Validationاختصاص دادیم. برای راحتی در مقایسه بین شبکه های مختلف تمام داده ها رابر۲۵۵تقسیم می کنیم تا بین یک وصفرقرار بگیرند ونرمالیزه شوند.

سپس با استفاده از Reshapeکردن هر کدام از داد گان آموزش میتوان آنها را به همراه Labelمربوط به آنها چاپ نمود.

تعداد کلاسها در این تمرین ۱۰ است درمورد داده های خروجی باید آنها را به صورت one_hot_codingدر بیاوریم زیرا توابعی که در کراس هستند ورودی های خود را به این صورت دریافت می کنند.

```
#Convert class Vectors to Binary class matrics
y_train = keras.utils.to_categorical(y_train , num_classes)
y_val = keras.utils.to_categorical(y_val , num_classes)
y_test = keras.utils.to_categorical(y_test , num_classes)
```

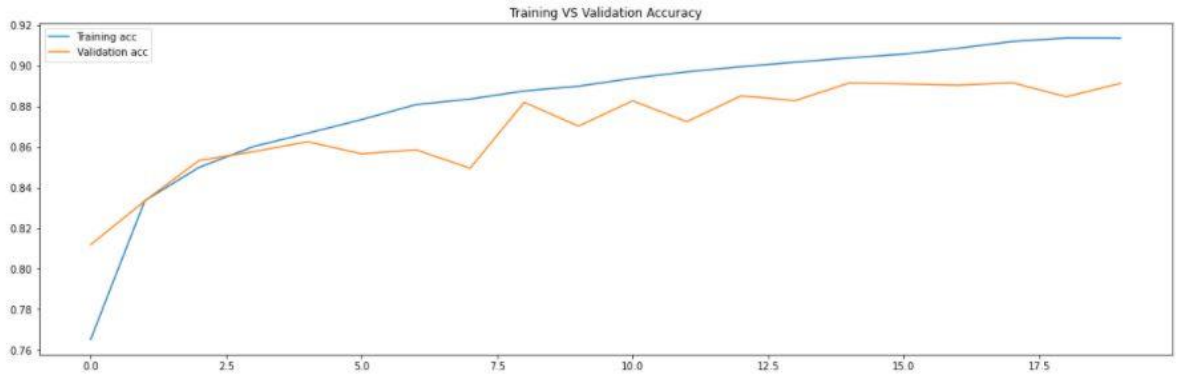
قسمت اول)MLP

برای این کار باید آزمایشهای متفاوتی انجام داد تا بهترین و بدترین شرایط رابیابیم:

	first layer	Activation	Dropout	Second layer	Activation	Dropout	Third layer	Activation	Dropout	Train Acc	Valid Acc	Test Acc
1	1000	relu	0.2							92.23%	89.42%	88.66%
2	1000	sigmoid	0.2							91.49%	89.69%	88.81%
3	500	relu	0.2							91.97%	90.01%	88.62%
4	500	sigmoid	0.2							90.96%	89.03%	88.44%
5	1000	relu	0.2	500	relu	0.2				90.53%	88.77%	87.76%
6	1000	sigmoid	0.2	500	sigmoid	0.2				91.32%	89.34%	88.33%
7	1000	relu	0.2	500	relu	0.2	200	relu	0.2	89.35%	88.12%	86.82%
8	1000	sigmoid	0.2	500	sigmoid	0.2	200	sigmoid	0.2	90.77%	89.41%	88.62%

در حالتهای بالا بهترین حالت مربوط به حالت دوم می باشد که درآن از یک لایه مخفی با ۱۰۰۰ نورون استفاده شده و تابع فعال سازی آن Reluاست و۲۰٪نورونهای آن را Dropکرده ایم و در لایه خروجی هم از تابع فعال سازی SoftMaxاستفاده شده که خروجی را به صورت احتمالاتی به ما میدهد.

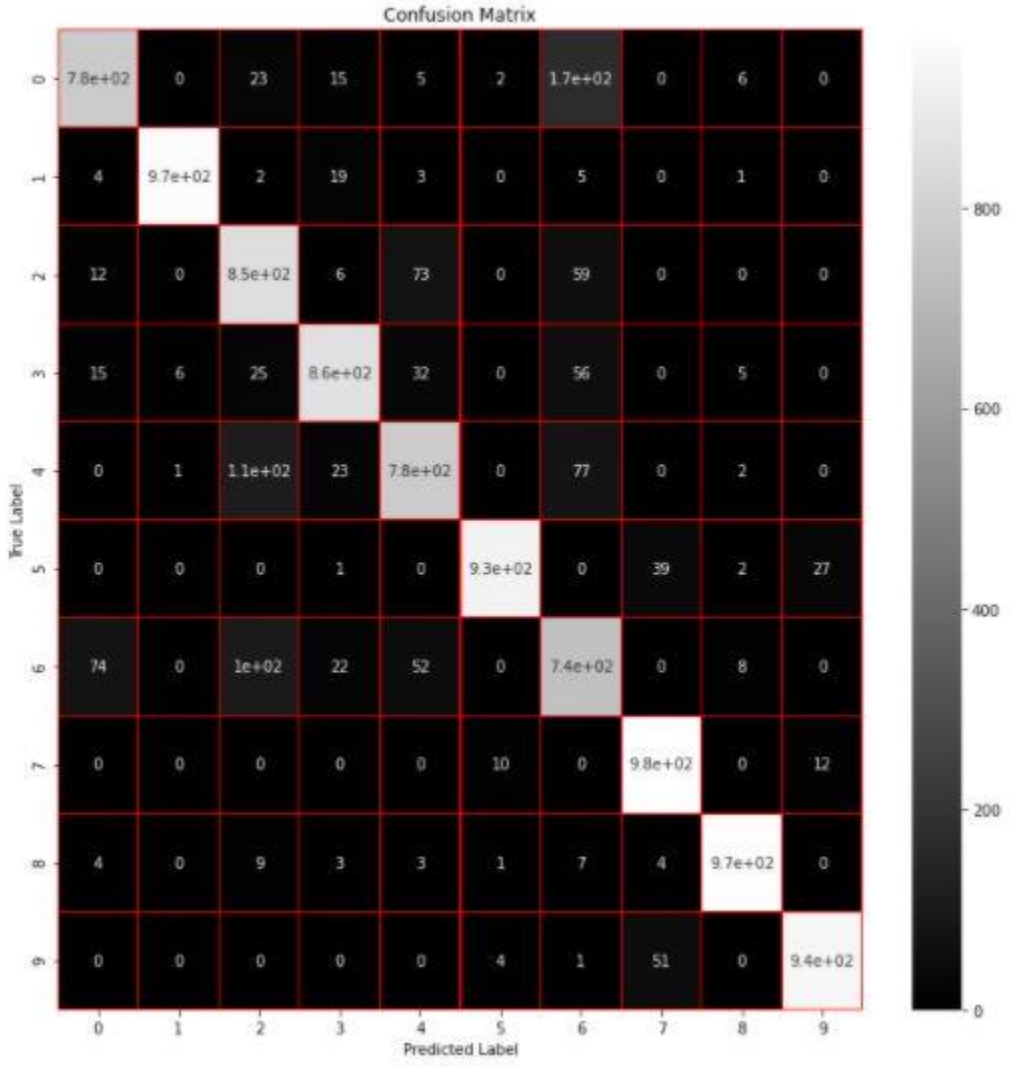
نمودار مربوط به Lossدر بهترین حالت



نمودار مربوط به Accuracyدر بهترین حالت

درنهایت برای بهترین حالت ماتریس درهم ریختگی را رسم می کنیم

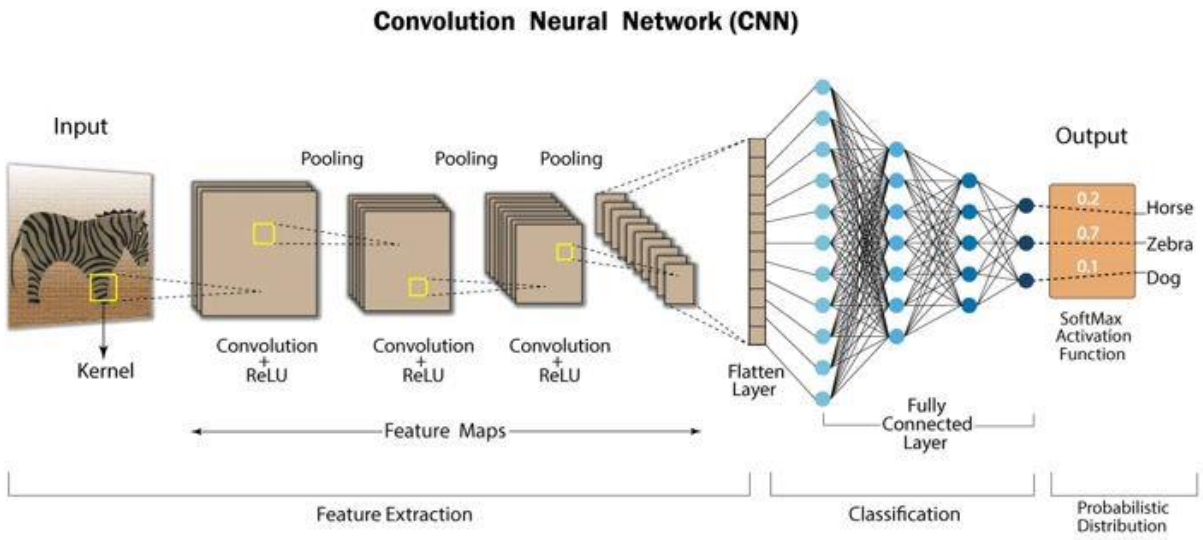
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

y_pred=model.predict(X_test)

y_pred_classes=np.argmax(y_pred, axis= 1)
y_true= np.argmax(y_test, axis= 1)

confusion_mtx = confusion_matrix(y_true, y_pred_classes)

f,ax = plt.subplots(figsize=(12,12))
sns.heatmap(confusion_mtx, annot= True, linewidths= 0.1, cmap= "gist_yarg_r", linecolor= "r")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



نتایج مربوط به این روش:

در این روش با افزایش لایه های مخفی الزاما دقت افزایش نمی یابد اما در تعداد لایه های برابر ونورونهای مساوی تابع فعال سازی Sigmoidبهتر از Reluعمل می کند.

قسمت دوم)CNN



در این روش قبل از Flatکردن با استفاده از فیلترها ویژگی های تصویر را استخراج کنیم سپس شبکه را با مدل خود آموزش بدهیم. برای استخراج ویژگی های تصویر باید از روش های کانوالو کردن و سپس لغزاندن استفاده کرد که با کاهش ابعاد عکس همراه است برای این کار ابتدا از یک کرنل مثلا۳ در۳ (در این پروژه)استفاده می کنیم. که این کرنل درکل تصویر می لغزد که می تواند همراه با ویژگی هایی مثل تشخیص لبه یا Smoothکردن همراه باشد.

نتایج آزمایش برای به دست آوردن بهترین و بدترین روش به طریق زیر بوده است:

	No.layers	Activation	Optimizer	Train Acc	Valid Acc	Test Acc
1	1	Relu	adam	94.26%	90.79%	90.38%
2	1	Relu	SGD	85.85%	86.29%	86.08%
3	1	sigmoid	adam	86.19%	86.29%	85.86%
4	1	sigmoid	SGD	80.13%	81.26%	80.16%
5	2	Relu	adam	95.01%	91.74%	90.61%
6	2	Relu	SGD	85.60%	86.32%	85.50%
7	2	sigmoid	adam	86.74%	86.63%	86.08%
8	2	sigmoid	SGD	74.41%	75.54%	74.44%

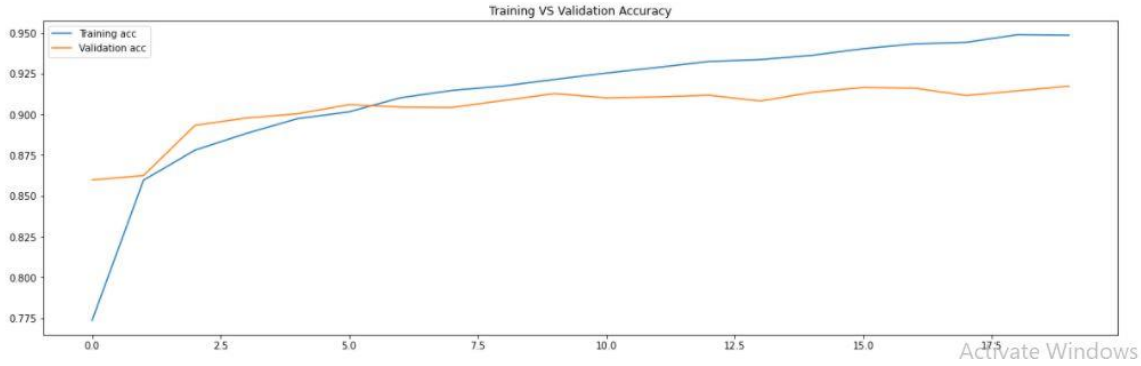
در حالتهای بالا بهترین حالت مربوط به آزمایش شماره ۵می شود که درآن از دو لایه با تابع فعال سازی Relu استفاده شده است وOptimizerمربوط به آن از نوع adamاست

البته در تمامی آزمایشهای بالا تعداد لایه های کانوالو اول برابر با۳۲ و لایه دوم برابر با ۶۴ است

سایز کرنل ها ۳در۳ و Pool sizeبرابر با ۲در۲ می باشد همچنین مقدار Dropout۰.۲ انتخاب شده است و تابع فعال سازی مربوط به لایه خروجی از نوع SoftMaxمی باشد.

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Dropout,Flatten,Activation,Input,Conv2D,MaxPooling2D
model=keras.Sequential()
model.add(Input(shape=input_shape))
model.add(Conv2D(32,kernel_size=(3,3),activation= 'relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64,kernel_size=(3,3),activation= 'relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(100))
model.add(Dropout(0.5))
model.add(Dense(num_classes , activation='softmax'))
```

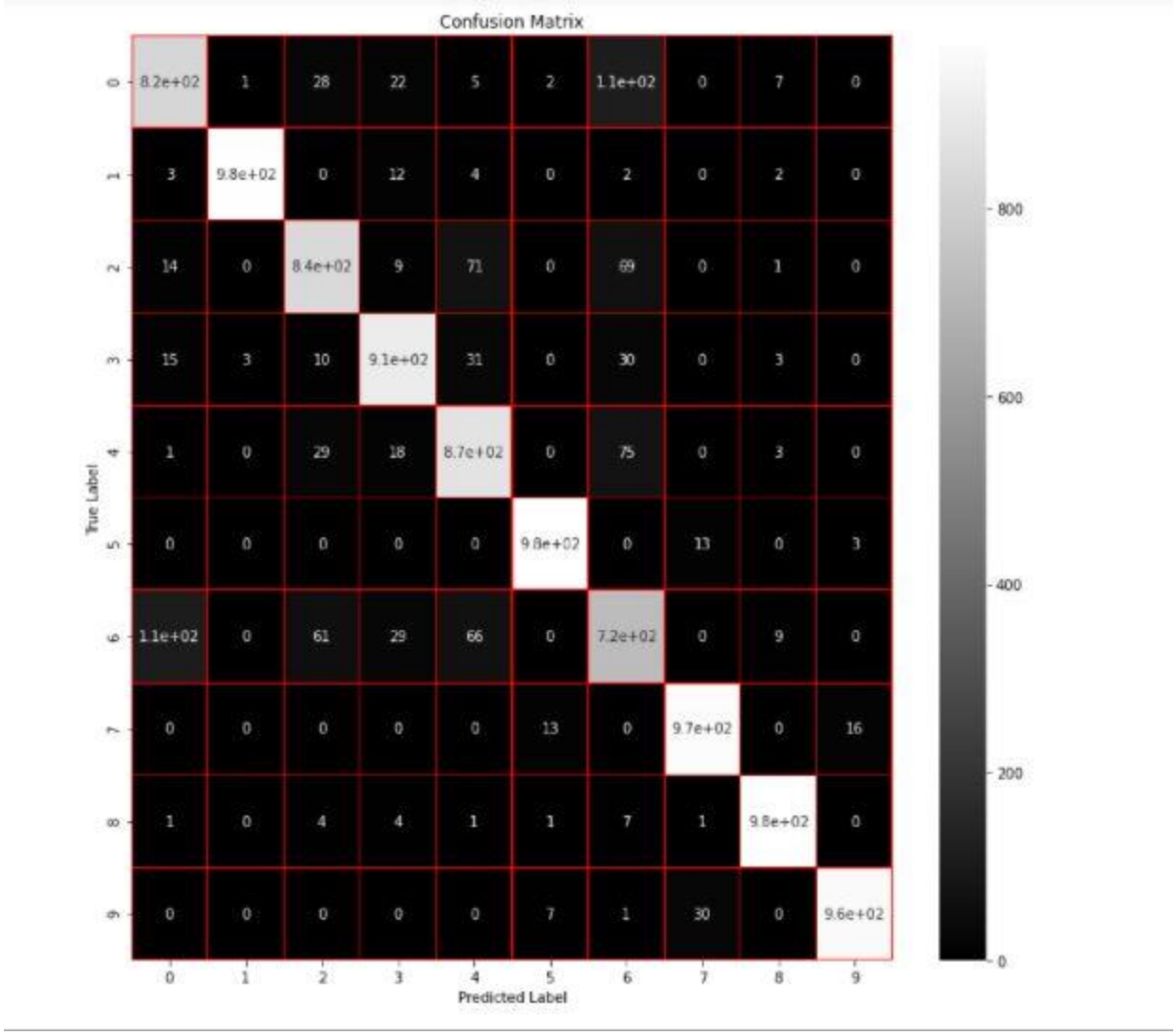
نمودار مربوط به Lossدر بهترین حالت



نمودار مربوط به Accuracyدر بهترین حالت

با مقایسه نتایج به دست آمده می بینیم که در حالتی که تعداد لایه ها برابر باشد Optimizer adamبهتر از SGDعمل میکند. همچنین با داشتن تعداد لایه های برابر وOptimizerیکسان عملکرد تابع فعال سازی Relu بهتر است.

درنهایت برای بهترین حالت ماتریس درهم ریختگی را ترسیم می کنیم.


Confusion Matrix
True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
8.2e+02 1 28 22 5 2 1.1e+02 0 7 0
3 9.8e+02 0 12 4 0 2 0 2 0
14 0 8.4e+02 9 71 0 69 0 1 0
15 3 10 9.1e+02 31 0 30 0 3 0
1 0 29 18 8.7e+02 0 75 0 3 0
0 0 0 0 0 9.8e+02 0 13 0 3
1.1e+02 0 61 29 66 0 7.2e+02 0 9 0
0 0 0 0 0 13 0 9.7e+02 0 16
1 0 4 4 1 1 7 1 9.8e+02 0
0 0 0 0 0 7 1 30 0 9.6e+02
800
600
400
200
0