" به نام خدا "

مروارید لعل نور 9623097

گزارش پروژه نهایی درس یادگیری ماشین

بخش اول :

ابتدا داده ها را لود میکنیم و سپس در شبکه ای چند لایه داده ها را قرار داده و ترین کردن را آغاز میکنیم . شبکه چند لایه مورد استفاده در اینجا همانطور که در کد نیز نشان داده شده است متشکل از لایه اول که داده ها را به صورت flatten میگیرد و سپس لایه دوم که لایه dropout هست و بخش از 784 ورودی را حذف میکند بعد از آن یک لایه دیگر با 256 نورون و یک لایه دیگر با 128 نورون و سپس لایه آخر با 10 نورون (تعداد کلاس ها) قرار دارد . در مدل کردن این شبکه از optimizer خاصی استفاده نشده است زیرا در صورت پروژه اشاره ای به این موضوع نشده است . با انجام چندین بار ترین به این نتیجه میرسیم که تعداد اپیاک ها اگر 16 باشد مناسبل است زیرا نه مقدار محاسبات بالا رفته است و نه از یادگیری جلوگیری کرده ایم . نتیجه یادگیری درون سلول های کد نشان داده شده است که نشان میدهد کمترین validation loss مربوط به اپیاک 11 و به میزان 0.25 است که این امر مقدار صحت را به 75 درصد میرساند . همچنین ماتریس confusion نیز در ادامه بدست آمده است که این مورد نیز درون کد قابل مشاهده است . همین مراحل را بدون لایه drop out و همچنین برای تابع فعالسازی sigmoid بدست آورده ایم .که نتایج در جدول زیر به صورت خلاصه آورده شده اند :

Activation function \ dropout	With dropout	Without dropout
Relu	The epoch with the minimum validation loss is: 11 Validation Accuracy: 0.75453	The epoch with the minimum validation loss is: 4 Validation Accuracy: 0.83333
sigmoid	The epoch with the minimum validation loss is: 9 Validation Accuracy: 0.78493	The epoch with the minimum validation loss is: 28 Validation Accuracy: 0.84420

همانطور که مشاهده میشود با تبدیل توابع فعالسازی لایه های مخفی از relu به sigmoid صحت مقداری افزایش یافته است همچنین با اضافه کردن لایه های dropout نتیجه عکس گرفته و صحت یادگیری بسیار کاهش یافته است پس نتیجه میگیریم برای به کار گیری این الگوریتم (MLP) بر روی این نوع از دادگان بهتر است از لایه dropout استفاده نکنیم زیرا تمامی دادگان اهمیت دارند و با گذاشتن dropout بخشی از آنها از بین رفته و یادگیری بی دقت تر انجام میشود و به همین ترتیب تابع فعالسازی را نیز sigmoid انتخاب میکنیم.

همانط.ر که مشاهده میشود کمترین صحت بروری دادگان تست مربوط به حالتی است که تابع relu را با لایه های dropout استفاده کرده ایم . بهترین حالت زمانی اتفاق افتاده است که از تابع sigmoid بدون لایه dropout استفاده کرده ایم . وزن های مربوط به این حالت در فایلی با نام model_28.hd5 ضمیمه شده است .

توضیح بخش هایی از کد :

بعد از مشخص کردن داده ها و لایه های شبکه به سراغ شروع یادگیری میرویم که اینکار با model.fit انجام میشود که داده ها و لیبل ها را میگیرد و سپس میزان هر batch که در اینجا 32 تعریف شده است و بعد از آن تعیین میکنیم که چند درصد از داده ها به عنوان داده تست در نظر گرفته شوند که در اینجا 25 درصد را به عنوان تست در نظر گرفته ایم .

پس از این قسمت به تحلیل نتایج یادگیری میپردازیم و اینکار توسط model.history انجام میشود به این صورت که هر کدام از صحت ها و یا loss ها که مدنظر ما باشد به عنوان ورودی به ارایه ی دو بعدی model.history میدهیم و نتیجه را مشاهده میکنیم . برای کشیدن نمودار ها نیز از همین متد استفاده میکنیم و نمودار صحت و loss برای برحسب شماره اپیاک رسم میکنیم .

معیار انتخاب بهترین اپیاک و بهترین یادگیری ، کمترین بودن مقدار validation loss در آن یادگیری است که با قطعه کد زیر شماره آن اپیاک بدست آمده و سپس وزن ها در آن مدل ذخیر میشوند :

```
pos = np.argmin(history.history['val loss'])
```

همانطور که مشاهده میشود pos همان شماره بهترین اپیاک است که با قطعه کد زیر میتوان از آن استفاده کرد و مدل را ذخیره کرد :

```
model.save("model_{}.hd5".format(pos))
```

سپس نوبت به ماتریس confusion میرسد که ازکتابخانه sklearn برای محاسبه آن استفاده کردیم به این صورت که لیبل های واقعی و لیبل های پیشبینی شده را به آن میدهیم و سپس توسط متدی که برای گرافیک بهتر آن نوشته ایم این ماتریس را نمایش میدهیم (در سوال بعد نیز از همین کد ها برای انجام یادگیری استفاده کرده ایم فقط کمی از جزییات لایه های شبکه تغییر کرده است)

بخش دوم :

مانند بخش قبل داد ها را لود میکنیم . سپس شبکه ای به این شکل تعریف میکنیم :

Layer (type)	Output Shape	Param #
conv2d_17 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_16 (MaxPooling	(None, 13, 13, 32)	0
conv2d_18 (Conv2D)	(None, 12, 12, 64)	8256
max_pooling2d_17 (MaxPooling	(None, 6, 6, 64)	0
dropout_8 (Dropout)	(None, 6, 6, 64)	0
flatten_8 (Flatten)	(None, 2304)	0
dense_16 (Dense)	(None, 128)	295040
dropout_9 (Dropout)	(None, 128)	0
dense_17 (Dense)	(None, 10)	1290

لایه اول 28*28*32 است زیرا سایز عکس ها 28 * 28 است و سایز batch را 32 در نظر گرفته ایم ا صحت بالاروئ و یادگیری بهبود یابد . لایه ی بعدی داری سایز 64*28*28 است زیرا پس از انجام عمل کانولوشن برروی لایه اول حالا 32 حالت ما به 64 حالت تبدیل میشوند پس از این لایه لایه max pooling و لایه dropout را قرار داده ایم که تعداد وزن ها را و نورون ها را تاحدی کاهش میدهند و بعد از این لایه ها و انجام گرفتن محاسبات کانولوشنی از flatten برای تبدیل ماتریس داده ها به آرایه یک بعدی استفاده میکنیم و آنرا به یک لایه fully connected میدهیم تا ویژگی های موجود در این آرایه نیز علاوه بر خود عکس برروی یادگیری اثر بگذارند و پس از آن لایه آخر قرار دارد که شامل 10 نورون (تعداد کلاس ها) است. ما بروی این شبکه تعییراتی اعمال کرده و نتیجه هر حالت در جدول های صفحه بعد قابل مشاهده است .

Activation function \optimizer	SGD	Adam
Relu	Train Accuracy: 0.88102 Validation Accuracy: 0.89293 loss: 0.3031 accuracy: 0.8892 Test loss: 0.303 Test accuracy: 0.889	Train Accuracy: 0.90642 Validation Accuracy: 0.90687 loss: 0.3084 accuracy: 0.9039 Test loss: 0.308 Test accuracy: 0.9
sigmoid	Train Accuracy: 0.74007 Validation Accuracy: 0.77467 loss: 0.6604 accuracy: 0.7634 Test loss: 0.660 Test accuracy: 0.763	Train Accuracy: 0.92076 Validation Accuracy: 0.91613 loss: 0.2448 accuracy: 0.9093 Test loss: 0.244 Test accuracy: 0.9095

With dropout

Activation function \optimizer	SGD	Adam
Relu	Train Accuracy: 0.94342 Validation Accuracy: 0.90373 loss: 0.2855 accuracy: 0.8981 Test loss: 0.285 Test accuracy: 0.898	Train Accuracy: 0.90987 Validation Accuracy: 0.89620 loss: 0.5106 accuracy: 0.8973 Test loss: 0.510 Test accuracy: 0.897
sigmoid	Train Accuracy: 0.74764 Validation Accuracy: 0.76113 loss: 0.7949 accuracy: 0.7469 Test loss: 0.794 Test accuracy: 0.746	Train Accuracy: 0.88971 Validation Accuracy: 0.88027 loss: 0.3602 accuracy: 0.8716 Test loss: 0.360 Test accuracy: 0.871

Without dropout

مشاهده میشود که در حالتی که از لایه dropout استفاده کرده ایم عملکرد یادگیری با optimizer ، Adam بهتر بوده است و زمانی که از لایه dropout استفاده نکرده ایم تقریبا همان نتایج اما با صحت کمی پایین تر بدست آمده اند و به طور کلی عملکرد در زمان استفاده از تابع relu بهتر بوده است پس برای یافتن بهترین حالت با کمترین loss میان حالت هایی که با تابع relu بوده اند بحث میکنیم ، به طور کلی میزان اختلافات در این حالت ها ناچیز است و تقریبا همه آنها عملکرد قابل قبولی دارند اما بهترین صحت با کمترین loss متعلق به حالتی است که ترکیب Relu , SGD را بدون لایه dropout اعمال کرده ایم پس در این قسمت مدل مربوط به این حالت را ذخیره میکنیم . همانطور که مشخص است بدترین عملکرد مربوط به زمانی است که از ترکیب SGD , sigmoid استفاده کرده ایم که در اینجا میزان عملکرد کاهش یافته است دلیل آن میتواند این باشد که تابع sigmoid در بازه ی کوچکی بین 1 و 1- مقادیر مختلف دارد و خارج از این بازه بدون توجه به آنکه ورودی چه مقداری است مقادیر ثابت 1 یا 1- را درد و از آنجایی که داده ها بین 0 و1 نرمالایز نشده و مقادیر بزرگی بین 0 تا 255 دارند شاید این تابع انتخاب مناسبی برای این حالت نباشد و با نرمالایز کردن داده ها نتایج بهتری دریافت کنیم ، در حالی که تابع relu به ازای مقادیر مثبت خودشان را برمیگرداند و این یعنی ارزش هر داده حفظ شده و میان هر پیکسل با پیکسل بعدی تفاوت قائل شده ایم. پس به طور کلی و با توجه به نتایج تابع sigmoid در این جا و برای این مقادیر از داده ها مناسب نبست و از طرفی optimizer SGD هم کنار گذاشته میشود زیرا صحت عملکرد روی هر دو دادگان تست و ترین در optimizer دیگر بیشتر است .

(تمامی نمودار های مربوطه و ماتریس های confusion مربوط به هر حالت در کد با پسوند ipynp. آورده شده است و همچنین وزن های ذخیره شده بهترین حالت هر سوال ضمیمه شده است)