به نام خدا

گزارش فاز یک پروژه ۵ هوش مصنوعی، محمد عظیم پور، ۸۱۰۱۹۷۶۵۷

نام پروژه: پیاده سازی و آزمون شبکه های عصبی Feed Forward

هدف: حدس زدن کلاس تصاویری که اطلاعات پیکسل هایشان موجود است به وسیله شبکه عصبی پس از یادگیری بر روی تصاویر با کلاس مشخص.

شرح کلی: هر تصویر ابتدا مسطح شده و به صورت بردار به عنوان ورودی به شبکه عصبی داده می شود. هر درایه این بردار (معادل با یک پیکسل تصویر) یک ویژگی برای آن تصویر محسوب می شود. شبکه قرار است بر اساس این ویژگی ها و با ساختن ترکیبات غیرخطی از آن ها، وزن اتصالات بین لایه هایش را طوری تنظیم کند که خروجی آن ضمن داشتن کمترین خطا کلاس تصویر ورودی متناظر را به درستی تشخیص دهد.

داده ها:

TrainData.csv: داده ای آموزش، شامل ۶۰۰۰۰ خط که هر خط برداری به طول ۷۸۴ متناظر با یک تصویر ۲۸*۲۸ است.

TrainLabel.csv: کلاس داده های آموزش (عددی بین صفر تا نه) را مشخص می کند.

TestData.csv: داده ای آزمون، شامل ۱۰۰۰۰ خط که هر خط برداری به طول ۷۸۴ متناظر با یک تصویر ۲۸*۲۸ است.

TestLabel.csv: کلاس داده های آزمون (عددی بین صفر تا نه) را مشخص می کند.

بخش های پروژه:

قسمت اول، بررسی و پیش پردازش داده ها:

۱- اولین داده از هر کلاس نمایش داده شده است.

۲- مشاهده می شود که فراوانی تمام کلاس ها در داده های آموزش و آزمون با هم برابر است، یعنی در دادگان آموزش هر یک از ۹ کلاس ۶۰۰۰ عضو و در دادگان آزمون هر یک از ۹ کلاس ۱۰۰۰ عضو دارد.

۳- نرمالایز کردن داده ها سرعت یادگیری را افزایش می دهد و باعث می شود بردار وزن ها سریع تر همگرا شود. علت دیگر این است که اگر این کار را نکنیم در هنگام مشتق گرفتن ها ممکن است به علت استفاده از توابع نمایی سرریز (Overflow رخ دهد). همچنین این اعدادی که بین صفر و یک هستند می توانند به عنوان احتمال در نظر گرفته شوند. با توجه به سرریز هایی که در طول پروژه مشاهده شد به جای ۲۵۵ اعداد بر ۲۵۵۰ تقسیم شده اند.

قسمت دوم، تکمیل بخش های ناقص شبکه عصبی:

شبکه ای با دو لایه مخفی که به ترتیب ۲۰ و ۱۰ نورون دارند با شیوه وزن دهی uniform طراحی شده است. با توجه به سرریز هایی که در طول پروژه مشاهده شد وزن دهی uniform به صورت اعداد تصادفی در بازه صفر تا ۰,۰۵ تعریف شده است. همچنین برای به روز کردن وزن ها و مقدار بایاس از معادلات زیر استفاده کرده ایم (علامت * به معنی ضرب درایه به درایه و علامت × به معنی ضرب ماتریسی است):

$$Weight_i = Weight_i - lr * (X_i^T \times (Backprop\ Tensor_i * Activation\ Derivative_i))$$

$$Bias_i = Bias_i - lr * ([1.\ for\ i\ in\ range(len(layer\_input))] \times (Backprop\ Tensor_i * Activation\ Derivative_i))$$

$$Backprop\ Tensor_n = Loss\ Function\ Derivative$$

$$Backprop\ Tensor_i = (Backprop\ Tensor_{i+1} * Activation\ Derivative_{i+1}) \times Weight_{i+1}^T$$

قسمت سوم، طبقه بندی داده ها:

گام یک: در صورتی که شبکه ای با معماری گفته شده طراحی کنیم، با نرخ یادگیری ۰,۰۱ به دقت زیر می رسیم:

```
Epoch 30:
        Train: Average Accuracy: 0.8868833333333334     Average Loss: 0.030575458458879216
        Test: Average Accuracy: 0.8692092651757188      Average Loss: 0.03708928351877761
```

گام دو: نتایج حاصل از نرخ های یادگیری ۰,۱ و ۰,۰۱ و ۰,۰۰۱ و ۰,۰۰۰۱ به صورت زیر است:

Act=Relu(), Batch Size=32, Training Epochs=30				
Learning Rate	Accuracy (%)		Loss (*1000)	
	Train	Test	Train	Test
0.1	9.98	9.99	237.28	238.45
0.01	88.69	86.92	30.58	37.29
0.001	85.66	83.13	41.38	46.48
0.0001	75.56	75.18	66.86	67.81

مشاهده می شود که وقتی نرخ یادگیری بالاتر می رود در این تابع خاص به طور کلی عملکرد تابع مختل می شود. و دقت از ده درصد که مقدار اولیه است زیاد تر نمی شود. احتمالا به این علت که بعضی مقادیر منفی می شوند و مشتق این تابع به ازای ورودی های منفی صفر است و وقتی در ضرب های قسمت به روز رسانی وزن شرکت داده شود مقدار تغییر را صفر می کند.

```
Epoch 30:
        Train: Average Accuracy: 0.09986666666666667    Average Loss: 0.23727845322388147
        Test: Average Accuracy: 0.09994009584664537     Average Loss: 0.2384450995675195
```

همچنین وقتی این نرخ را کمتر بکنیم نرخ افزایش دقت کمتر می شود به این علت که بردارهای وزن را کم تغییر می دهیم. در حالتس که نرخ یادگیری را یک صدم بکنیم این امر مشهود تر است. در این حالت حداکثر جهش دقت حدود ۸ درصد است، حال آنکه برای نرخ یادگیری ۰,۰۱ این مقدار حدود ۲۰ درصد است.

۰,۰۰۱

```
Epoch 30:
        Train: Average Accuracy: 0.8566333333333334     Average Loss: 0.041381995071510635
        Test: Average Accuracy: 0.8312699680511182      Average Loss: 0.04647523599278625
```

۰,۰۰۰۱

```
Epoch 30:
        Train: Average Accuracy: 0.7555833333333334     Average Loss: 0.0668579930853412
        Test: Average Accuracy: 0.751797124600639       Average Loss: 0.0678107443696816
```

در حین انجام پروژه، گاها مشاهده شد با زیاد کردن نرخ یادگیری می توان افزایش دقت را تسریع کرد. ولی بعد از همگرا شدن شبکه این زیاد بودن نرخ باعث به هم ریختن مجدد دقت می شد. به این شکل که شبکه را از آن اکسترمم محلی که داخلش بود بیرون می انداخت. البته این مساله ممکن است برای حالاتی که احتمال همگرا شدن زودهنگام روی یک اکسترمم محلی برای شبکه وجود دارد مفید باشد.

گام سه: شبکه با توابع LeakyRelu، Identical و Sigmoid هم بررسی شد که نتایج به صورت زیر است:

Learning Rate=0.01, Batch Size=32, Training Epochs=30				
Activation Function	Accuracy (%)		Loss (*1000)	
	Train	Test	Train	Test
ReLU	88.69	86.92	30.58	37.29
Leaky ReLU	88.72	86.66	30.77	38.23
Identical	86.26	84.31	32.25	45.02
Sigmoid	61.92	61.42	16.42	16.48

LeakyRelu

```
Epoch 30:
        Train: Average Accuracy: 0.8871833333333333     Average Loss: 0.03077049977538504
        Test: Average Accuracy: 0.8666134185303515      Average Loss: 0.0382325032242907
```

Identical

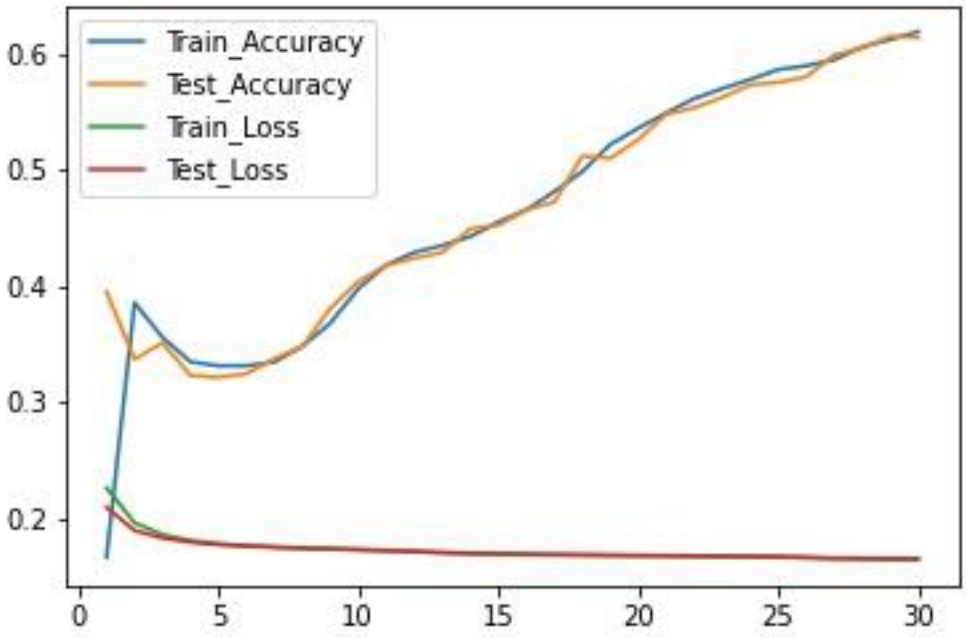
```
Epoch 30:
        Train: Average Accuracy: 0.86275        Average Loss: 0.03925023162845063
        Test: Average Accuracy: 0.8430511182108626      Average Loss: 0.04502013311364473
```

Sigmoid

```
Epoch 30:
        Train: Average Accuracy: 0.6192333333333333     Average Loss: 0.16415571269662865
        Test: Average Accuracy: 0.6142172523961661      Average Loss: 0.1648067336670667
```

مشاهده می شود که تابع فعال سازی Relu بهترین دقت را روی داده های آزمون به ما می دهد.

در روند یادگیری تابع Sigmoid نوساناتی مشاهده می شود. توابعی مثل Sigmoid و tanh چون در ازای ورودی های خیلی کم و خیلی زیاد به خطوطی افقی میل می کنند و مشتقشان صفر می شود، از جایی به بعد روند یادگیری را مختل می کنند. بنابراین به جای آنها از توابعی مثل Relu و LeakyRelu استفاده می شود.



گام چهار: برای Batch Size های ۱۶ و ۱۲۸ نتایج تست به صورت زیر است:

Act=RelU(), Learning Rate=0.01, Training Epochs=30				
Batch Size	Accuracy (%)		Loss (*1000)	
	Train	Test	Train	Test
16	88.75	86.16	30.62	38.08
32	88.69	86.92	30.58	37.29
128	88.41	85.96	31.41	40.08

۱۶

```
Epoch 30:
        Train: Average Accuracy: 0.8875333333333333     Average Loss: 0.030623357839833377
        Test: Average Accuracy: 0.8616  Average Loss: 0.03807886677803007
```

۱۲۸

```
Epoch 30:
        Train: Average Accuracy: 0.8841395700071073     Average Loss: 0.03140698094211832
        Test: Average Accuracy: 0.8595727848101266      Average Loss: 0.04008289949287391
```

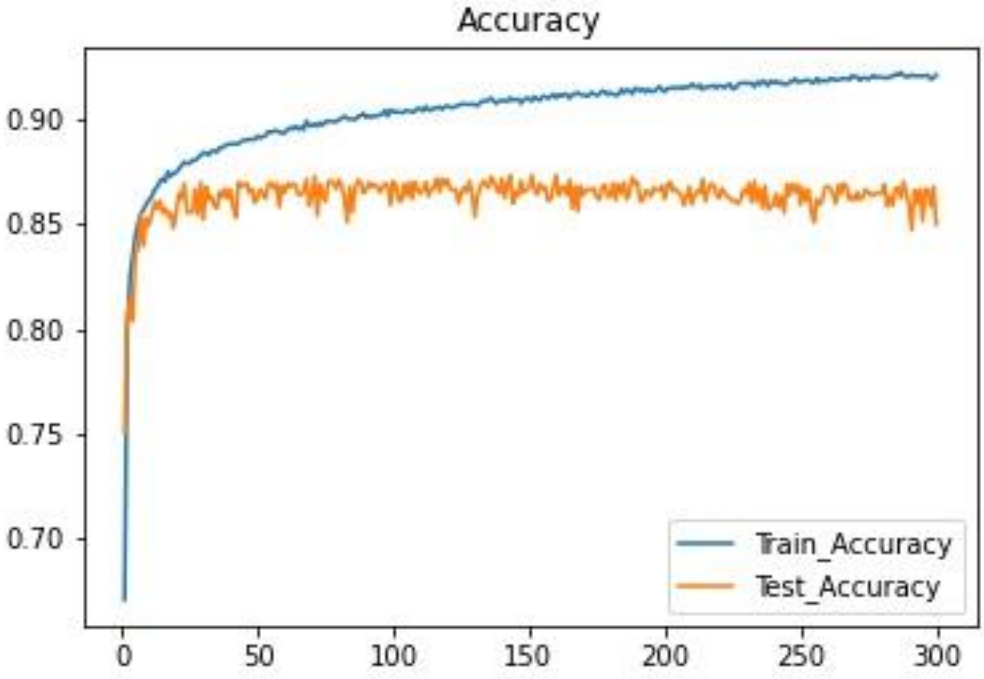
با تغییر Batch Size تغییر خاصی در دقت شبکه حاصل نمی شود. علت این است که بهرحال روی تمام داده های آموزش عملیات یادگیری انجام می شود.

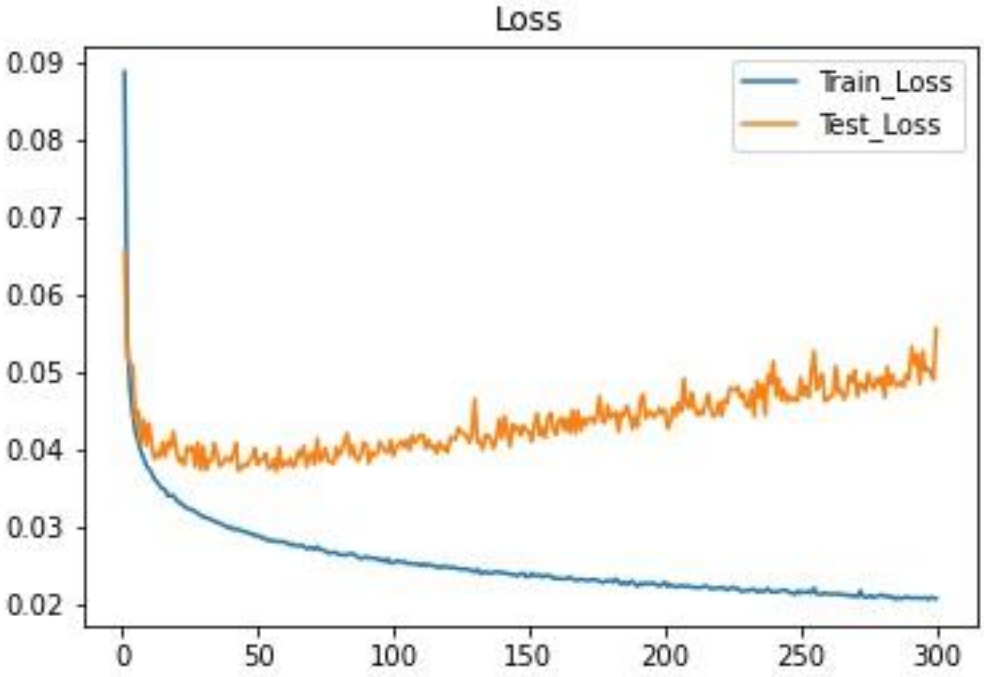
دلیل استفاده از Batch به جای داده ی تنها بالا بردن سرعت یادگیریست، چرا که در این حالت برای تعداد مشخصی از ورودی ها عملیات یادگیری و به روز کردن اوزان همزمان و با عملیات های سریع ضرب و جمع ماتریسی انجام می شود.

اگر Batch Size خیلی کوچک باشد اثر داده های پرت (نویزها) در آن بیشتر به چشم می آید، همچنین ممکن است یک Batch ایجاد اکسترمم محلی بکند و بعد از آن روند یادگیری مختل شود.

گام پنج: با صرفا یک مرتبه یادگیری وزن های شبکه مقدار زیادی تغییر نمی کنند و در نتیجه انتظار دقت بالایی نمی توان داشت. باید آموزش تا جایی که دقت در حال افزایش است ادامه داده شود. از طرفی اگر این زیاد همملییات یادگیری و به روز رسانی وزن ها را انجام دهیم تمام نویزهای موجود در داده های آموزش هم در وزن دهی دخیل می شوند و باعث می شود دقت شبکه روی داده های آزمون به خاطر درست فرض کردن این داده های غلط کم شود. همچنین مشاهده می شود که مقدار تابع Loss رفته رفته روی داده های آزمون زیاد می شود که مورد نامطلوبیست.

روند تغییر دقت روی داده های آموزش و آزمون و همچنین هزینه شبکه روی این دو مجموعه داده به صورت زیر است:


Accuracy
0.90
0.85
0.80
0.75
0.70
0
50
100
150
200
250
300
Train_Accuracy
Test_Accuracy

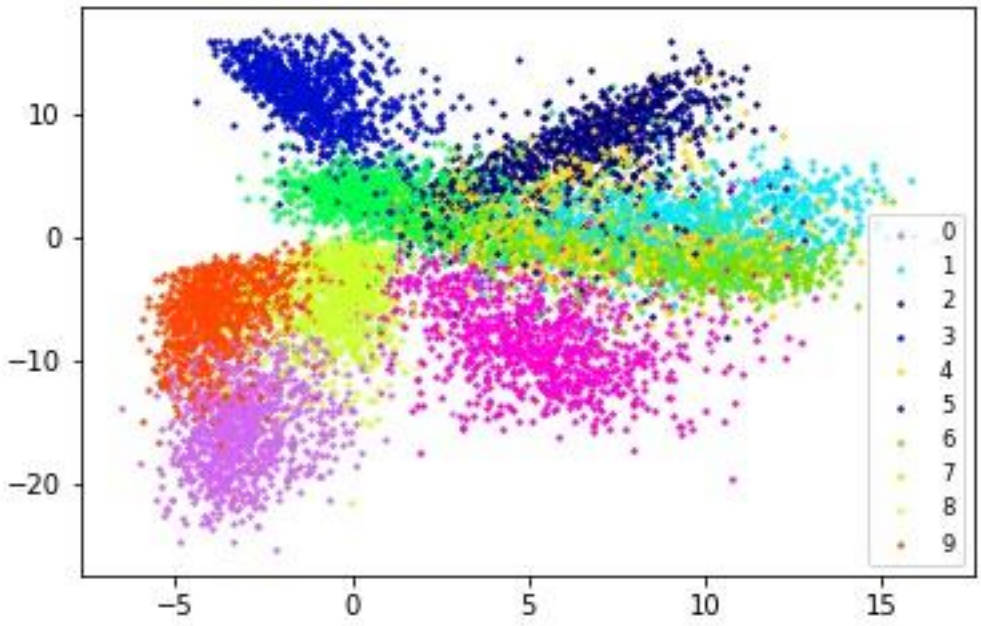

Loss
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0
50
100
150
200
250
300
Train_Loss
Test_Loss

قسمت چهارم، ترسیم داده های با بعد کاهش یافته:

شبکه ای با سه لایه که به ترتیب ۷۸۴، ۲ و ۱۰ نورون دارند با تابع فعال سازی همانی و نرخ یادگیری ۰,۰۰۱ آموزش داده می شوند. این شبکه بعد از ده Epoch به دقت حدود ۷۳ درصد می رسد.

```
Epoch 10:
        Train: Average Accuracy: 0.7168666666666667     Average Loss: 0.07640306409056496
        Test: Average Accuracy: 0.7297324281150159      Average Loss: 0.07450524831036857
```

خروجی لایه یکی مانده به آخر این شبکه با توجه به کلاسی که به آن تعلق دارد به صورت زیر است:



مشاهده می شود که تفکیک داده ها بر حسب کلاس ها به چه صورت است. کلاس هایی که داده های آنها جداتر از هم قرار دارند – یعنی همپوشانی یا اصلا مرز مشترک ندارند – بهتر از هم تشخیص داده می شوند؛ مانند داده های کلاس های سه (آبی نیلی) و هشت (صورتی کمرنگ)، یا داده های کلاس های صفر (صورتی پررنگ) و نه (قرمز). چرا که وقتی خروجی به ازای داده ای در ناحیه ای قرار بگیرد که فقط داده های یک کلاس در آن ناحیه قرار دارند، به احتمال بسیار بالا متعلق به همان کلاس است. ولی وقتی ناحیه به صورت گفته شده نباشد و داده های دو یا چند کلاس در آن جا حضور داشته باشند دیگر به سادگی تفکیک داده ها از هم امکان پذیر نیست.