قدم دوم: (Feedforward)

مطابق توضیحات داده شده مدل زده شده بر روی ۲۰۰ عکس train شد که خروجی به صورت زیر می باشد.

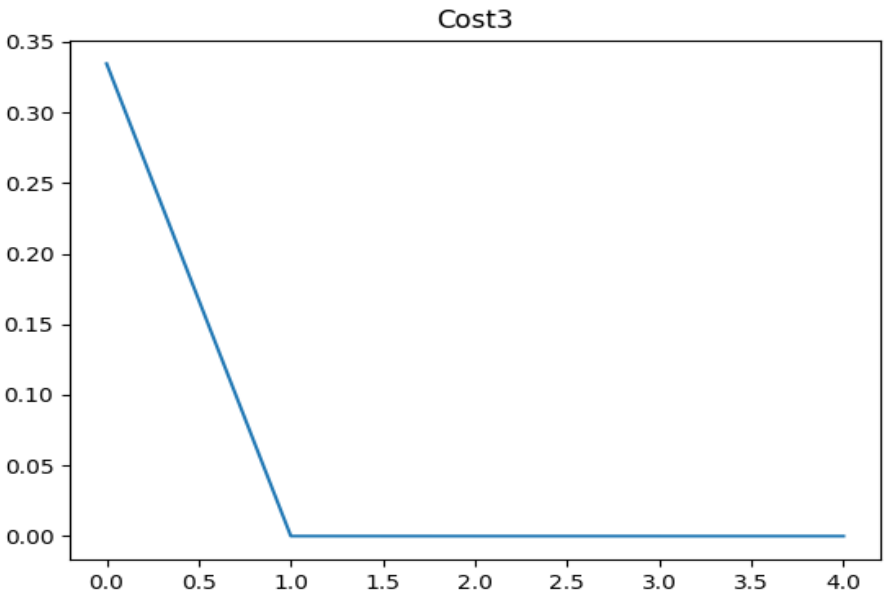
```
C:\Python38\python.exe "D:/College/Term 7/Computational Intelligence/Projects/Project1/Fruit-360/FeedForward.py"
Number of correct prediction is : 47
Number of total images is : 200
Accuracy is : 23.5%

Process finished with exit code 0
```

که دقت مدل 23.5% می باشد.

قدم سوم: (Backpropagation)

مطابق توضیحات مدل برای ۲۰۰ عکس با پارامتر های epoch = 5 , learning rate = 1 Batch size = 5 آموزش دید که نمودار هزینه ها به صورت زیر است.



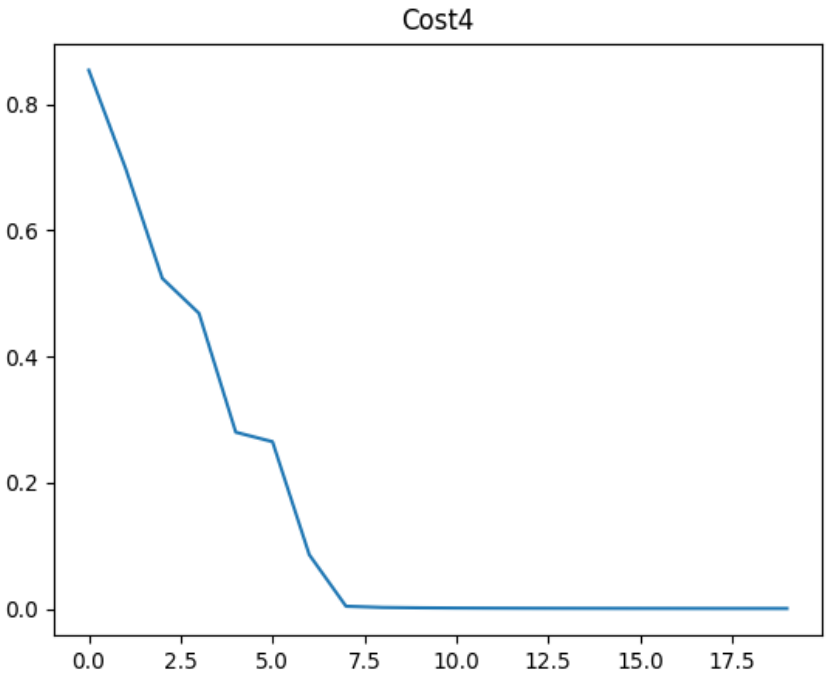
همچنین خروجی و دقت نهایی مدل نیز به صورت زیر است.

```
C:\Python38\python.exe "D:/College/Term 7/Computational Intelligence/Projects/Project1/Fruit-360/BackPropagation.py"
epoch number : 0
epoch number : 1
epoch number : 2
epoch number : 3
epoch number : 4
Number of correct prediction is : 200
Number of total images is : 200
Accuracy is: 100.0%
```

که دقت نهایی مدل بعد از اجرای پنج epoch به ۱۰۰ درصد رسیده است و زمان اجرای ۵ epoch ۰۴:۵۴ ثانیه شد.

قدم چهارم: (Vectorization)

مطابق توضیحات مدل برای ۲۰۰ عکس با پارامتر های epoch = 20 , learning rate = 1 Batch size = 5 آموزش دید که نمودار هزینه ها به صورت زیر است.



همچنین خروجی و دقت نهایی مدل نیز به صورت زیر است.

```
epoch number : 17
epoch number : 18
epoch number : 19
Number of correct prediction is : 1900
Number of total images is : 1900
Accuracy is: 100.0%

Process finished with exit code 0
```

که دقت نهایی مدل بعد از اجرای پنج epoch به ۱۰۰ درصد رسیده است و زمان اجرای ۲۰ epoch

۱۵ ثانیه شد که میبینیم نسبت به قدم قبلی کاهش بسیار زیادی داشته است و الگوریتم بسیار بهینه شده است.

قدم پنجم: (Test Model)

در این قسمت مدل را برای ۱۹۶۲ عکس داده آموزشی train کرده و دقت آن را هم برای داده های آموزشی و هم برای داده های تست که ۶۶۲ می باشد بدست می آوریم.

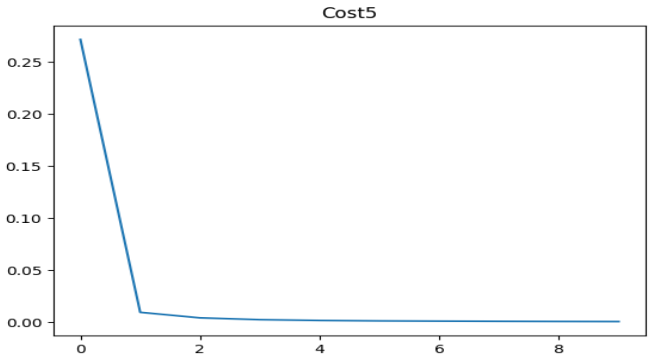
مقدار پارامتر هارا نیز به صورت زیر درنظر میگیریم:

Epoch = 10

Learning rate = 1

Batch size = 10

که خروجی برای داده های train به صورت زیر است:



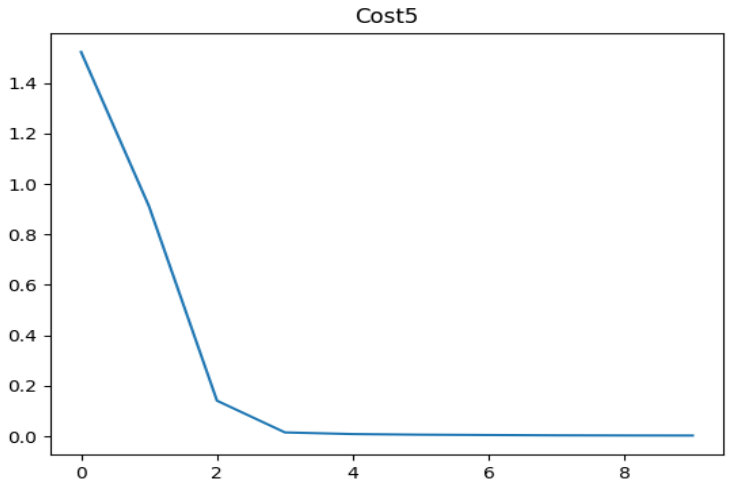
و خروجی نیز به صورت زیر است.

```
epoch number : 7
epoch number : 8
epoch number : 9
Number of correct prediction is : 1900
Number of total images is : 1900
Accuracy is: 100.0%

Process finished with exit code 0
```

که میبینیم دقت بعد از ۱۰ epoch به 100% میرسد و زمان اجرای کد ۸ ثانیه شد.

همچنین نمودار هزینه ها برای داده های test به صورت زیر بدست آمد:



و خروجی به صورت زیر است.

```
epoch number : 7
epoch number : 8
epoch number : 9
Number of correct prediction is : 600
Number of total images is : 600
Accuracy is: 100.0%

Process finished with exit code 0
```

که بعد از ۱۰ epoch دقت به 100% رسید و زمان اجرای آن ۳ ثانیه شد.

بخش امتیازی

۱) برای مقایسه مقادیر مختلف جدول زیر را رسم و خروجی نهایی را ثبت میکنیم.

Epoch	Learning Rate	Batch Size	Accuracy
1	1	10	25%
2	1	10	94%
5	1	10	100%
2	0.1	10	74%
2	0.5	10	99.4%
2	1.5	10	99.5%
2	1	5	25%
2	1	20	50%
2	1	50	17%

طبق جدول بالا برای هر متغیر ۳ مقدار با ثابت نگهداشتن ۲ متغیر دیگر در نظر گرفته شد.

میبینیم درسطر اول مقادیر learning rage و batch size استاندارد است ولی مقدار epoch کم است و مدل به اندازه کافی train نشده است و بنابراین دقت آن 25% می باشد.

در سطر دوم با افزایش مقدار epoch دقت آن به 94% رسید و در سطر سوم با افزایش آن به ۵ دقت آن 100% شد.

در سطر ۴ تا ۶ مقادیر مختلف learning rate تست شد که میبینیم برای مقدار 0.1 دقت مدل کم تر از دو مقدار دیگر است و دلیل آن این است که با سرعت کمی به منیمم تابع هزینه نزدیک شده ایم. همچنین درسطر ۵ و ۶ نیز به دقت 99% رسیده ایم.

در سطر ۷ تا ۹ مقدار batch size را تغییر داده ایم که میبینیم برای مقدار 5 به دقت 25% و برای مقدار ۱۵ به دقت 50% و برای مقدار ۵۰ به دقت 17% می رسیم پس بهترین مقدار برای batch همان ۱۰ می باشد.

۲) در این بخش از روش Momentum برای minimize کردن تابع هزینه استفاده شده است که فرمول این روش به صورت زیر است.

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

weight increment — learning rate — weight gradient

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor — weight increment, previous iteration

که فرمول بالا همان SGD می باشد و فرمول پایین روش Momentum که در فایل

Extra – Momentum.py در خط های ۱۰۰ تا ۱۱۷ پیاده سازی شده است.

در مقایسه این روش با روش SGD با پارامتر های یکسان دقت مدل برای روش SGD ۹۶٪ و برای روش Momentum مقدار ۹۹٪ بدست آمد.

همچنین ضریب Momentum برابر 0.3 در مقایسه بالا درنظر گرفته شد.

۳) در فایل Extra – 6 classification.py کد مربوطه زده شده است.

کلاس های Banana و Kiwi اضافه شده اند. برای این منظور کلاس های Feature_Extraction_Train.py و Loading_Datasets.py تغییر داده شدند که فایل های تغییر یافته نیز آپلود شده اند.

چون تعداد کلاس ها زیاد شده اند پس برای عملکر شبکه با تعداد مشابه epoch نسب به قبل کاهش میابد پس باید تعداد epoch ها بیشتر شود.

برای مثال دقت شبکه برای تعداد epoch ۲۰ برابر ۸۳٪ میشود درصورتی با ۴ کلاس دقت شبکه با epoch ۵ برابر ۱۰۰ میشود.

شکل زیر دقت شبکه در حالتی که ۶ کلاس داریم را نشان میدهد.

```
epoch number : 17
epoch number : 18
epoch number : 19
Number of correct prediction is : 2410
Number of total images is : 2900
Accuracy is: 83.10344827586206%

Process finished with exit code 0
```

که میبینیم تعداد عکس ها به ۲۹۰۰ افزایش پیدا کرده است.

۴) تابع softmax درفایل Softmax.py پیاده سازی شده است که مطابق شکل زیر پیاده سازی شده است.

```
def softmax(x):
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum()
```

پس از استفاده از این تابع و مقایسه نتایج با تابع sigmoid فرق زیادی دیده نشد.