به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

هوش مصنوعی

تمرین کامپیوتری پنجم فاز دوم

نام و نام خانوادگی	مهسا تاجیک
شماره دانشجویی	810198126
تاریخ ارسال گزارش	7 تیر

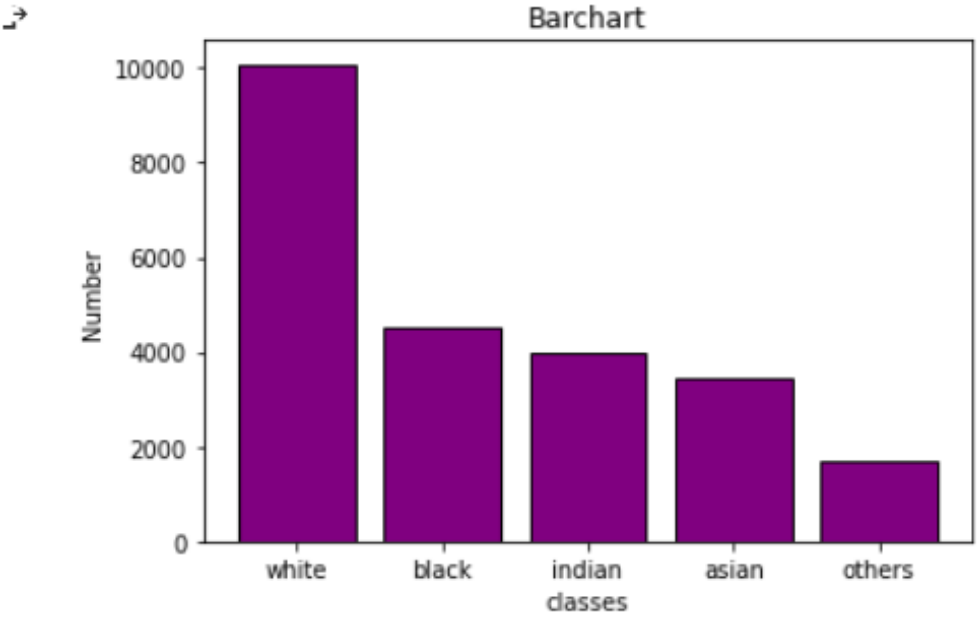
فاز اول : بررسی و پیش پردازش داده

- تعداد داده های خوانده شده برابر است با 23708 که در 3 تا از تصاویر برچسب مشکل داشت و آن ها را از مجموعه داده حذف کردیم پس در کل 23705 داده داریم که سه مقدار برای آن ها در نظر گرفته شده : سن ، جنسیت و نژاد. برای سن ، 104 کلاس مختلف ، برای جنسیت دو کلاس مختلف زن و مرد و برای نژاد 5 کلاس داریم که عبارتند از:

White, black, Indian, Asian, others

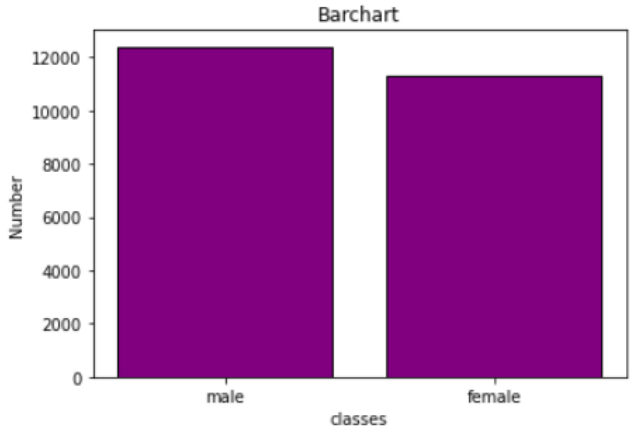
```
(23705, 4)
age       23705
gender    23705
race      23705
file      23705
dtype: int64
number of age classes:  104
number of gender classes:  2
number of race classes:  5
```

- تعداد تصاویر هر کلاس برای هر کدام از خروجی ها همراه با نمودار میله ای آنها در زیر آمده است :
  - نژاد:



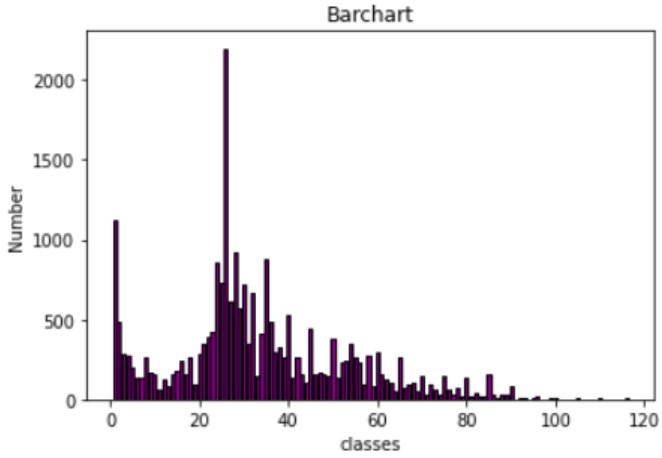

```
[10078, 4526, 3975, 3434, 1692]
```

  - جنسیت:



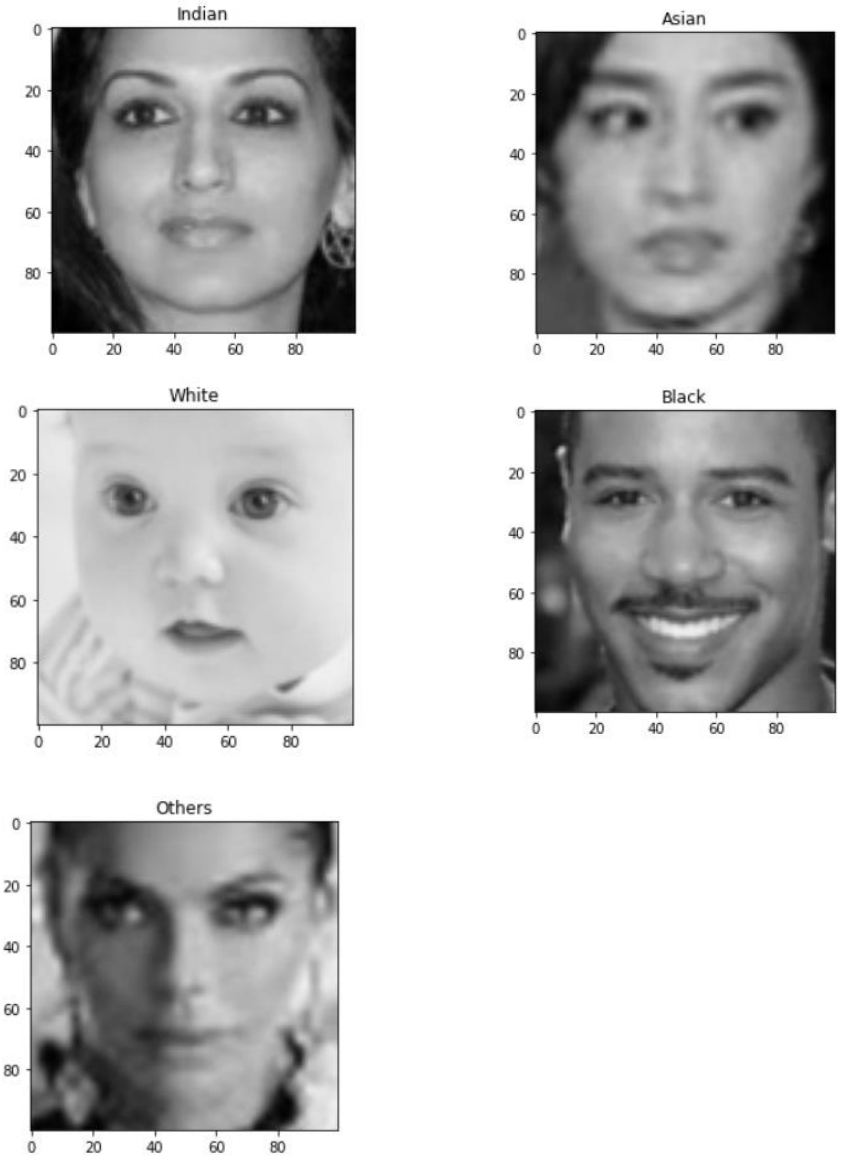

```
[12391, 11314]
```

- سن :




```
[2197, 1123, 918, 880, 859, 734, 724, 664, 615, 570, 526, 483, 482, 440, 426,
409, 395, 381, 353, 350, 346, 325, 293, 293, 289, 284, 273, 271, 268, 266, 265,
263, 262, 259, 247, 241, 236, 232, 196, 177, 170, 166, 159, 158, 157, 157, 156,
155, 153, 153, 148, 148, 147, 143, 139, 138, 133, 132, 131, 130, 125, 103, 100,
100, 98, 98, 97, 94, 82, 82, 81, 77, 69, 65, 63, 58, 56, 50, 40, 35, 34, 33, 33,
32, 28, 24, 23, 22, 18, 17, 13, 11, 10, 9, 9, 5, 5, 5, 4, 3, 2, 2, 1, 1]
```

- از آنجا که خروجی که می خواهیم پیش بینی کنیم تنها نژاد افراد است بنابراین تنها برای همین خروجی، از هر کلاس یک تصویر همراه نام آن نمایش می دهیم و همانطور که مشاهده می کنیم طبق خواسته قسمت اول، تصاویر grayscale و با ابعاد 100*100 هستند :



- دیتاست را با دستور زیر به دو بخش آموزش و تست تقسیم می کنیم و 20 درصد داده ها برای تست در نظر گرفته شده اند:

```
x_train, x_test, y_train, y_test = train_test_split(data, labels['race'], test_size=0.2)
```

- انجام on-hot encoding روی برچسب ها به این دلیل است که مسئله طبقه بندی است و تخمین یک مقدار نیست و می خواهیم کلاس را برای داده مشخص کنیم.

فاز دوم : طراحی شبکه عصبی

در این قسمت تصاویر و برچسب ها بعد از نرمالیزه شدن و onehot encoding و تقسیم به داده های تست و ترین ، به یک شبکه کانولوشنی داده می شوند.

مدلی که طراحی می کنیم بصورت زیر است که 5 قید خواسته شده در صورت سوال اعمال شده است و سپس مدل کامپایل می شود:

1. بهینه‌ساز شما باید از نوع SGD باشد.
2. مقدار learning rate باید 0.01 باشد.
3. تعداد epoch ها باید 10 باشد.
4. اندازه batch_size باید 32 باشد.
5. تابع فعال‌سازی تمام لایه‌ها غیر از لایه آخر باید relu باشد.

```
myInput = keras.layers.Input(shape=(100,100,1))
conv1 = keras.layers.Conv2D(16, 1, activation='relu', padding='same', strides=2)(myInput)
conv2 = keras.layers.Conv2D(32, 1, activation='relu', padding='same', strides=2)(conv1)
flat = keras.layers.Flatten()(conv2)
out_layer = keras.layers.Dense(5, activation='softmax')(flat)
myModel = Model(myInput, out_layer)
myModel.summary()
myModel.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01), loss=keras.losses.categorical_crossentropy, metrics=['accuracy'])
```

پارامترهای مدل را در زیر میبینیم :

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         [(None, 100, 100, 1)]     0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 50, 50, 16)        32
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 25, 25, 32)        544
_________________________________________________________________
flatten_5 (Flatten)          (None, 20000)             0
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 100005
=================================================================
Total params: 100,581
Trainable params: 100,581
Non-trainable params: 0
_________________________________________________________________
```

فاز سوم : طبقه بندی داده ها

در این قسمت خواسته شده که شبکه را روی داده ها آموزش دهیم و با تغییر پارامترها به بیشترین مقدار F1 برسیم.

با دستور زیر شبکه به داده ها فیت میشود :

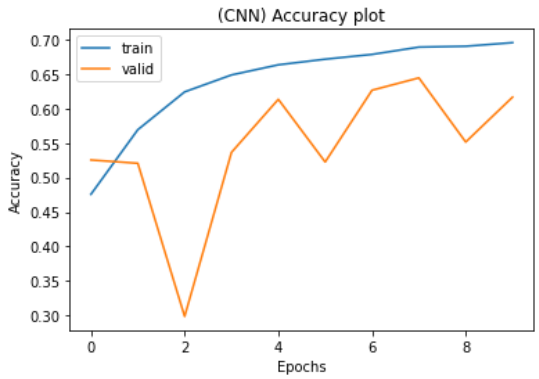
```
network_history = myModel.fit(x_train, y_train, batch_size=32, epochs=10, validation_split=0.2)
```

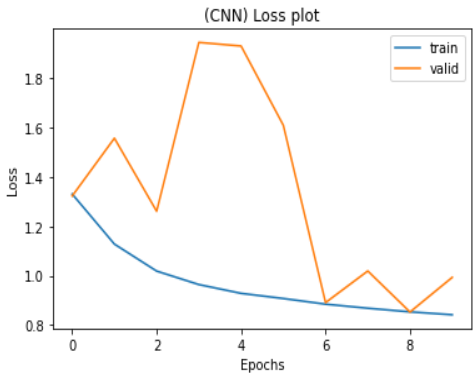
و سپس با دستور زیر می توانیم مقادیر را برای داده های تست پیش بینی کنیم:

```
y_pred = myModel.predict(x_test)
```

با مقایسه ی مقدار پیش بینی شده و مقدار برچسب اصلی داده های تست می توانیم مقادیر دقت و خطا را محاسبه کنیم و نمودار ها را رسم کنیم.

نمودار دقت و خطا برای داده های آموزش و ولیدیشن برحسب تعداد ایپاک ها بصورت زیر است :





تاثیر optimizer :

- این ضریب در بروزرسانی وزن ها در آپتیمایزر SGD موثر است و در مقدار تغییر وزن در مرحله ی قبل ضرب شده و بعنوان یک ترم اضافه با مقدار محاسبه شده به صورت عادی برای وزن این مرحله ، جمع می شود ، اگر ثابت مومنتوم صفر باشد ، بروزرسانی عادی انجام میشود. فرمول آن در تصویر قابل مشاهده است:

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

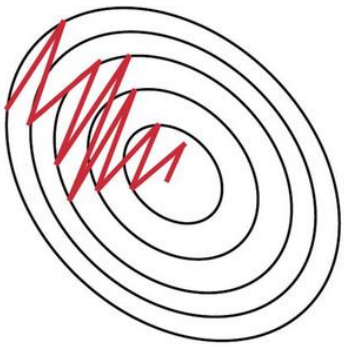
weight increment — learning rate — weight gradient

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

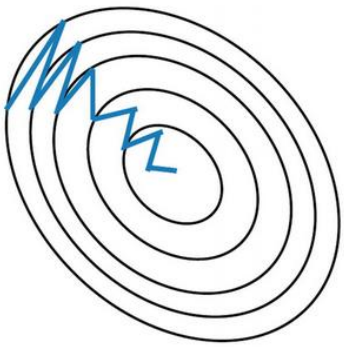
momentum factor — weight increment, previous iteration

درصورتیکه تغییرات هم جهت باشد ، در نظر گرفتن مومنتوم باعث تسریع آموزش می شود و در صورتیکه تغییرات یکسان نباشد ، موجب پایداری فرآیند آموزش می شود. بصورت کلی این ضریب باعث افزایش سرعت و دقت آموزش می شوددد. در حقیقت این ضریب ، میانگین متحرک شیب های گذشته را که به طور فزاینده ای رو به زوال است ، جمع می کند و در جهت آنها حرکت می کند.

اگر ساده بخواهیم توضیح دهیم ، Momentum تکنیکی برای جلوگیری از حرکت حساس است . وقتی گرادیان با هر تکرار محاسبه می شود ، می تواند جهت کاملاً متفاوتی داشته باشد و مراحل یک مسیر زیگزاگ ایجاد می کند ، که تمرین را بسیار کند می کند .چیزی شبیه به تصویر زیر :



Stochastic Gradient Descent **without** Momentum



Stochastic Gradient Descent **with** Momentum

- اموزش مدل با مومنتوم های 0.5 و 0.9 :

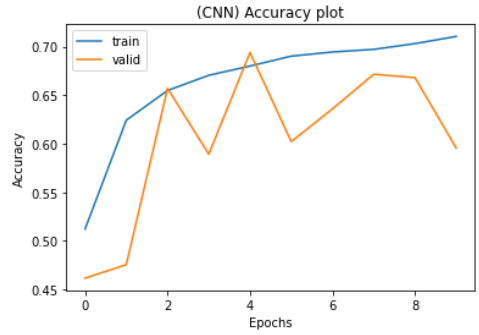
زمان آموزش با مومنتوم 0.5 :

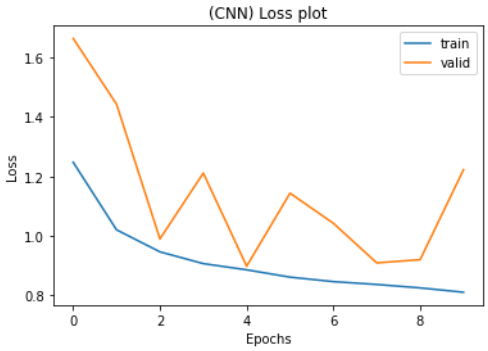
```
train time : 142.9386788539996
```

زمان آموزش با مومنتوم 0.9 :

```
train time : 123.33775372200034
```

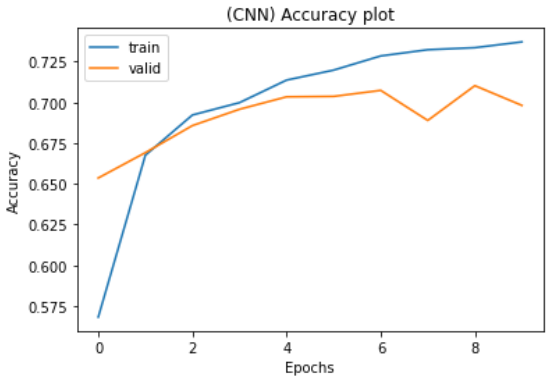
نمودارهای دقت و خطا با مومنتوم 0.5 :

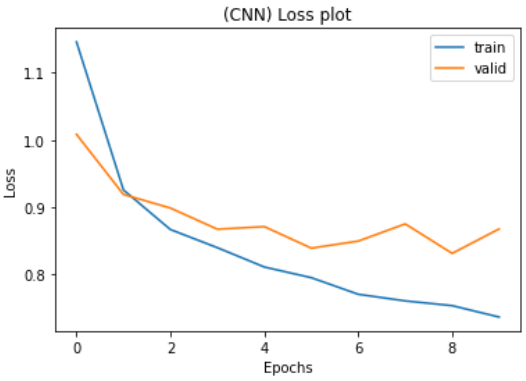





```
train loss: 1.1191740036010742
test loss: 1.1034605503082275
```

نمودارهای دقت و خطا با مومنتوم 0.9 :






```
train loss: 0.8484247922897339
test loss: 0.8815826773643494
```

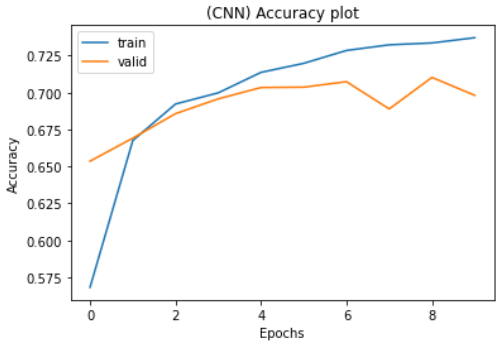
همانطور که انتظار داشتیم با افزایش مقدار مومنتوم هم سرعت و هم دقت آموزش افزایش یافت.

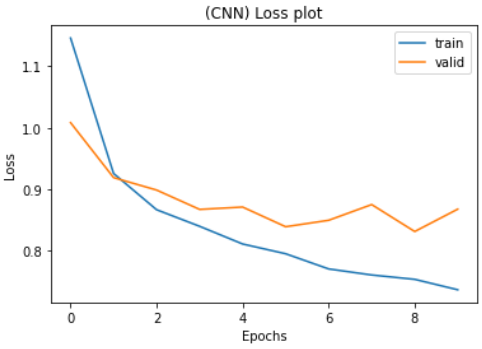
- مقدار مومنتوم بین صفر و یک و معمولا مقادیر بین 0.9 تا 0.99 تنظیم می شود بنابراین به نظر می رسد هرچه این مقدار بیشتر باشد، نتیجه بهبود می یابد اما باید در مدل های مختلف و دیتاست های مختلف تست شود.
- اپتیمایزر را به آدام تغییر میدهیم و شبکه را روی اده ها آموزش می دهیم:

زمان آموزش در استفاده از اپتیمایزر آدام :

```
train time : 143.30022445000213
```

نمودارهای دقت و خطا در استفاده از اپتیمایزر آدام:






```
train loss: 0.7518089413642883
test loss: 0.8823625445365906
```

تاثیر epoch :

- تعداد ایپاک ها را 20 قرار می دهیم و شبکه را آموزش می دهیم.
- مدل های مختلف به تعداد ایپاک های مختلفی برای اموزش نیاز دارند که به معماری شبکه و نوع دیتاست بستگی دارد. بعضی مدل ها بخاطر پیچیدگی بیشتر نیاز به زمان بیشتری برای آموزش دارند.
- باید در ایپاک های مختلف آموزش مدل را بررسی کنیم و نمودار های آموزش و اعتبارسنجی را مقایسه کنیم جایی که شروع به واگرا شدن می کنند آموزش را متوقف کنیم چون از این نقطه مدل شروع می کند به چسبیدن به داده ها و اورفیت شدن و و برای جلوگیری از آن باید تعداد ایپاک ها را کمتر کنیم . روش های دیگری که در ادامه گفته شده استفاده از لایه dropout است که از اورفیت شدن جلوگیری می کند.

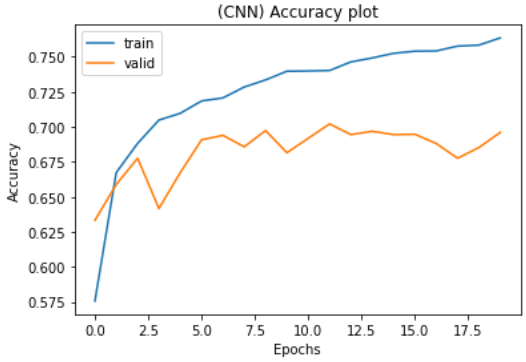
تاثیر loss function :

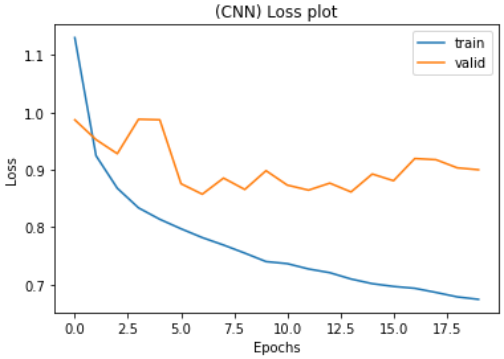
- نتیجه آموزش شبکه با تابع loss ، categorical cross entropy ، (آپتیمایزر آدام و 20 ایپاک):

زمان آموزش شبکه:

```
train time : 255.77340066099714
```

نمودارهای دقت و خطا:





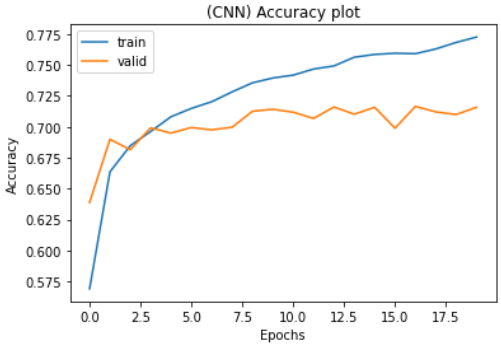

```
train loss: 0.7068353295326233
test loss: 0.8367494940757751
```

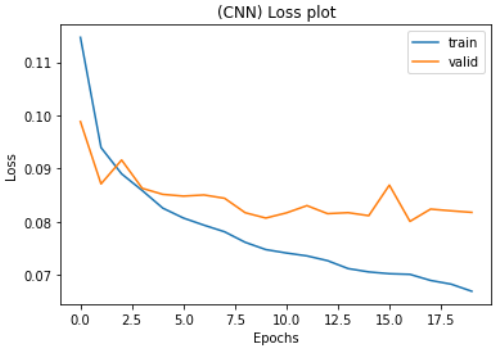
- نتیجه آموزش شبکه با تابع loss ، MSE ، (آپتیمایزر آدام و 20 ایپاک)::

زمان آموزش شبکه:

```
train time : 246.95002619900333
```

نمودارهای دقت و خطا:






```
train loss: 0.06761510670185089
test loss: 0.0816270187497139
```

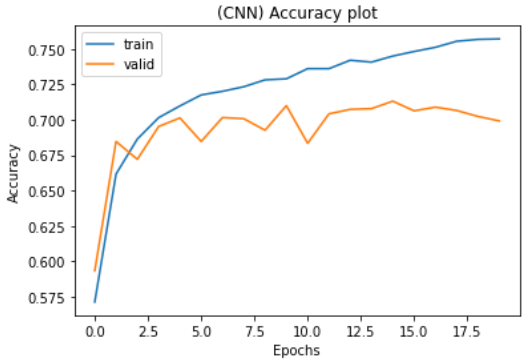
- تابع MSE برای طبقه بندی باینری غیر محدب است. به زبان ساده ، اگر یک مدل طبقه بندی باینری با تابع MSE Cost آموزش داده شود ، تضمین نمی کند که تابع Cost به حداقل برسد. دلیل این امر این است که تابع MSE انتظار ورودی های با ارزش واقعی را در

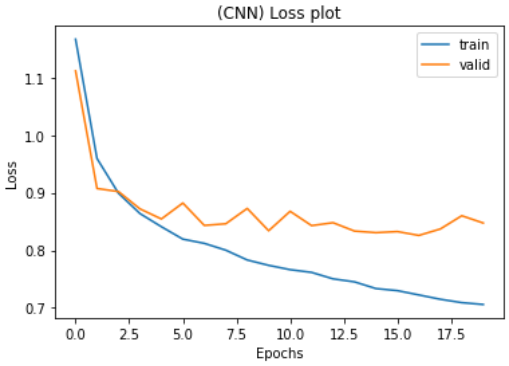
محدوده $(-\infty , \infty)$ دارد ، در حالی که طبقه بندی باینری احتمالات خروجی را در محدوده $(0،1)$ از طریق تابع سیگموئید / لجستیک مدل می کند.

تاثیر regularization :

- تاثیر l2regularization :

```
train time : 323.0324052930009
```





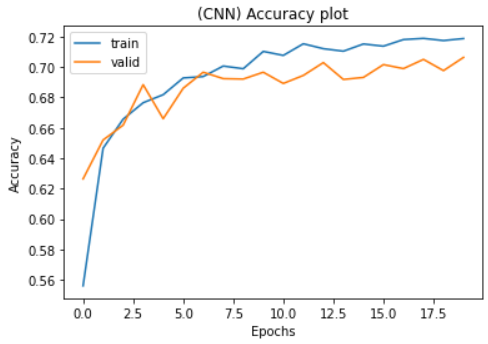

```
train loss: 0.7230110764503479
test loss: 0.8317886590957642
```

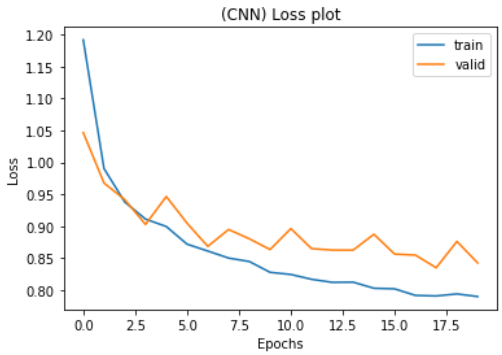
	precision	recall	f1-score	support
676	0.69	0.60	0.81	0
889	0.76	0.81	0.71	1
810	0.62	0.61	0.64	2
321	0.22	0.21	0.23	3
2045	0.78	0.80	0.76	4
accuracy			0.70	4741
macro avg	0.63	0.61	0.61	4741
weighted avg	0.70	0.70	0.70	4741

- تاثیر dropout :

در این بخش می خواهیم عملکرد شبکه را با و بدون لایه dropout بررسی کنیم . همانطور که می دانیم لایهdropout از overfit شدن جلوگیری می کند و میزان آن را کاهش میدهد بنابراین انتظار داریم مقدار خطای ارزیابی کمی بیشتر شود و مقادیر پیش بینی به مقادیر واقعی نزدیکتر شوند که در نمودارهای خطایی که در زیر آمده میبینیم که میزان خطای داده های ارزیابی اندکی از خطای داده های آموزش بیشتر است.

```
train time : 594.2521006830011
```






```
train loss: 0.7599307298660278
test loss: 0.8338251113891602
```

	precision	recall	f1-score	support
0	0.69	0.71	0.67	638
1	0.75	0.74	0.77	957
2	0.60	0.53	0.70	792
3	0.05	0.03	0.30	340
4	0.77	0.87	0.70	2014
accuracy			0.70	4741
macro avg	0.63	0.58	0.57	4741
weighted avg	0.68	0.70	0.68	4741