

## Report of question 6 - CNN for CIFAR10

در این قسمت از تمرین می‌خواهیم 50000 عکس از 10 کلاس را آموزش دهیم. برای پیاده سازی شبکه عصبی از لایه های کانولوشن و پولینگ استفاده می کنیم تا پارامترهای کمتری داشته باشیم تا بتوانیم به راحتی ترین کنیم. علاوه بر این، ما اثرات تعداد بلوک ها و لایه های پنهان را بررسی می کنیم و از روش های دراپ اوت و توقف زودهنگام استفاده می کنیم تا دقت بالاتری به دست آوریم و این تغییرات را روی نمودار نمایش دهیم.

در ابتدا باید دیتا را به صورتی که می‌خواهیم تبدیل کنیم. یکی از کارهایی که می‌کنیم وان هات است. در لیبل ها به وان هات به جای نمایش تعداد هر کلاس آرایه ای با طول تعداد کلاس ها ایجاد می شود و در آن احتمال وجود هر کلاس را می توان نشان داد که به این صورت می شود که تمام عناصر آرایه صفر هستند و شاخصی که عکس به آن می رسد متعلق به آن کلاس یکی است یا همه عناصر عددی بین 0 یا 1 هستند که احتمال هر کلاس در آن عکس نشان داده شده است. مجموع همه اعداد باید برابر با 1 باشد.

{x}

### Reshape dataset



convert labels to one-hot encoding



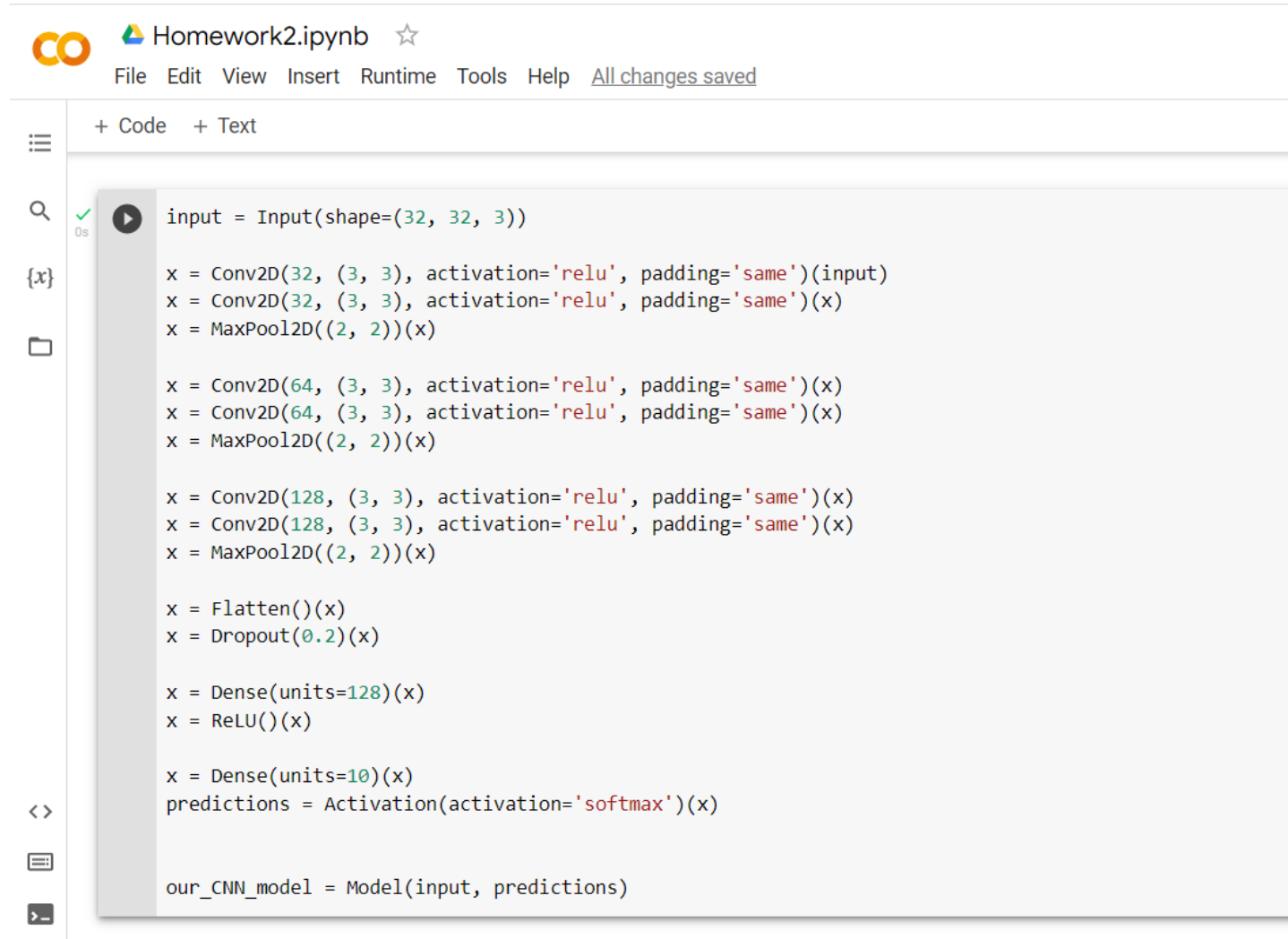
```
from tensorflow.keras.utils import to_categorical
```



```
[5] cat_y_trainData = to_categorical(Y_trainData, num_classes=10)  
cat_y_testData = to_categorical(Y_testData, num_classes=10)
```

## قسمت اول: ساخت مدل ساده سی ان ان

در این قسمت سه بلوک متشکل از تنها دو کانولوشن و یک مکس پولینگ ایمپلیمنت میشود. در نهایت همه این بخش ها فلت شده و توسط فقط یک لایه ی نهایی فولی کانکتد خروجی میدهند.



The screenshot shows a Jupyter Notebook titled "Homework2.ipynb". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with a status "All changes saved". Below the menu is a toolbar with "+ Code" and "+ Text" buttons. The left sidebar contains icons for a table of contents, search, variables, and file explorer. The main area displays a code cell with the following Python code:

```
input = Input(shape=(32, 32, 3))

x = Conv2D(32, (3, 3), activation='relu', padding='same')(input)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = MaxPool2D((2, 2))(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = MaxPool2D((2, 2))(x)

x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = MaxPool2D((2, 2))(x)

x = Flatten()(x)
x = Dropout(0.2)(x)

x = Dense(units=128)(x)
x = ReLU()(x)

x = Dense(units=10)(x)
predictions = Activation(activation='softmax')(x)

our_CNN_model = Model(input, predictions)
```

✓ 2s completed at 9:37 AM



+ Code + Text

```
[ ] our_CNN_model.fit(x=X_trainData, y=cat_y_trainData, epochs=35, batch_size=32,
                      validation_data=(X_testData, cat_y_testData))
```

```
1563/1563 [=====] - 9s 6ms/step - loss: 0.3132 - accuracy: 0.8892 - val_loss: 0.8022 - val_accuracy: 0.7694
Epoch 8/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.2925 - accuracy: 0.8973 - val_loss: 0.8084 - val_accuracy: 0.7692
Epoch 9/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.2715 - accuracy: 0.9037 - val_loss: 0.8489 - val_accuracy: 0.7678
Epoch 10/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.2578 - accuracy: 0.9087 - val_loss: 0.8871 - val_accuracy: 0.7608
Epoch 11/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.2487 - accuracy: 0.9108 - val_loss: 0.8877 - val_accuracy: 0.7625
Epoch 12/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.2357 - accuracy: 0.9175 - val_loss: 0.8619 - val_accuracy: 0.7718
Epoch 13/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.2270 - accuracy: 0.9205 - val_loss: 0.8984 - val_accuracy: 0.7717
Epoch 14/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.2138 - accuracy: 0.9248 - val_loss: 0.9761 - val_accuracy: 0.7628
Epoch 15/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.2141 - accuracy: 0.9258 - val_loss: 0.9343 - val_accuracy: 0.7683
Epoch 16/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.2057 - accuracy: 0.9284 - val_loss: 0.9818 - val_accuracy: 0.7682
Epoch 17/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.2045 - accuracy: 0.9293 - val_loss: 0.9876 - val_accuracy: 0.7516
Epoch 18/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.1984 - accuracy: 0.9322 - val_loss: 0.9630 - val_accuracy: 0.7607
Epoch 19/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.1869 - accuracy: 0.9361 - val_loss: 0.9758 - val_accuracy: 0.7687
Epoch 20/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.1946 - accuracy: 0.9348 - val_loss: 1.0325 - val_accuracy: 0.7730
Epoch 21/35
```

✓ 2s completed at 9:37 AM



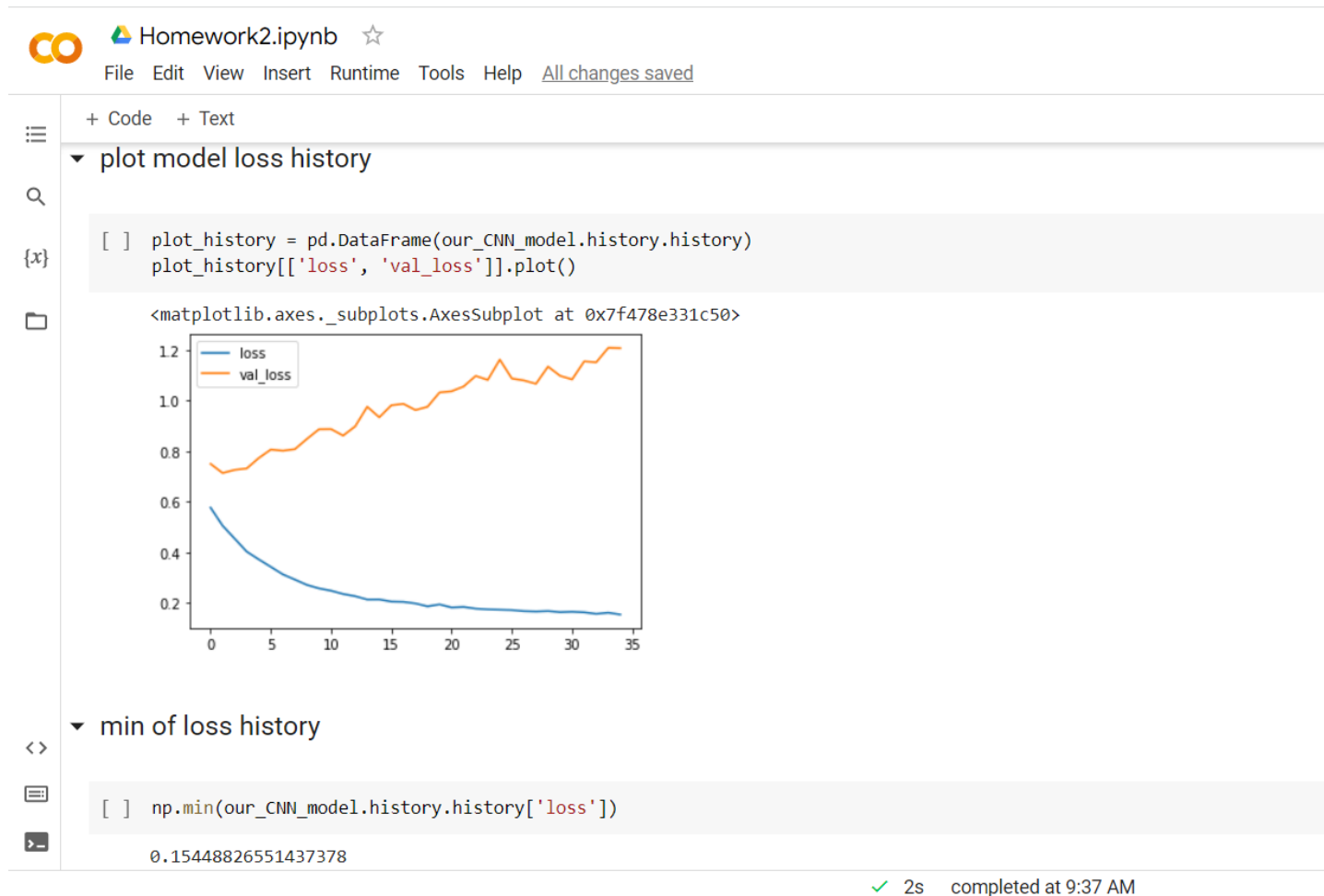
+ Code + Text



```
[ ] Epoch 24/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.1749 - accuracy: 0.9420 - val_loss: 1.0822 - val_accuracy: 0.7700
Epoch 25/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.1734 - accuracy: 0.9416 - val_loss: 1.1625 - val_accuracy: 0.7600
Epoch 26/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.1721 - accuracy: 0.9430 - val_loss: 1.0877 - val_accuracy: 0.7719
Epoch 27/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.1679 - accuracy: 0.9446 - val_loss: 1.0801 - val_accuracy: 0.7678
Epoch 28/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.1664 - accuracy: 0.9443 - val_loss: 1.0667 - val_accuracy: 0.7694
Epoch 29/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.1682 - accuracy: 0.9439 - val_loss: 1.1349 - val_accuracy: 0.7716
Epoch 30/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.1640 - accuracy: 0.9461 - val_loss: 1.0984 - val_accuracy: 0.7685
Epoch 31/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.1655 - accuracy: 0.9458 - val_loss: 1.0841 - val_accuracy: 0.7652
Epoch 32/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.1633 - accuracy: 0.9465 - val_loss: 1.1554 - val_accuracy: 0.7633
Epoch 33/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.1571 - accuracy: 0.9487 - val_loss: 1.1518 - val_accuracy: 0.7773
Epoch 34/35
1563/1563 [=====] - 9s 6ms/step - loss: 0.1617 - accuracy: 0.9460 - val_loss: 1.2089 - val_accuracy: 0.7521
Epoch 35/35
1563/1563 [=====] - 10s 6ms/step - loss: 0.1545 - accuracy: 0.9502 - val_loss: 1.2077 - val_accuracy: 0.7678
<keras.callbacks.History at 0x7f478e337b50>
```

▼ plot model loss history

همان طور که میبینیم مدل ترین میشود اما در نهایت فرایند کند شده و به اور فیتینگ میخورد. این نشان میدهد که با تغییر معماری میشود انرا بهینه تر کرد. برای روشن شدن وضعیت مدل ساده نمودار های ان را رسم میکنیم.



میبینیم که ابتدا اختلاف در حال کم شدن بود یعنی مدل خوب بود. لی به دلیل اور فیتینگ اختلاف ولیدیشن دیتا و دیتای ترین زیاد شد که این بد است. همین اتفاق را برای اکپوریسی نیز داریم.



Homework2.ipynb ☆

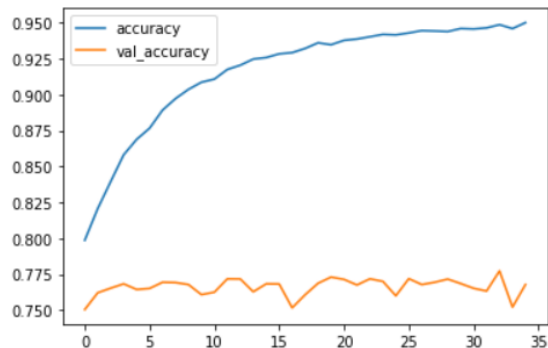
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

### plot model accuracy

```
[ ] plot_history = pd.DataFrame(our_CNN_model.history.history)
plot_history[['accuracy', 'val_accuracy']].plot()
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f478e1cd8d0>



### max of validation accuracy

```
[ ] np.max(our_CNN_model.history.history['val_accuracy'])
```

0.777300001907349

✓ 2s completed at 9:37 AM

در نهایت چک میکنیم که ببینیم مقادیر اکویریسی داده ها چقدر اختلاف با بیشترین اکویریسی ولیدیشن دیتا دارد.

{x}

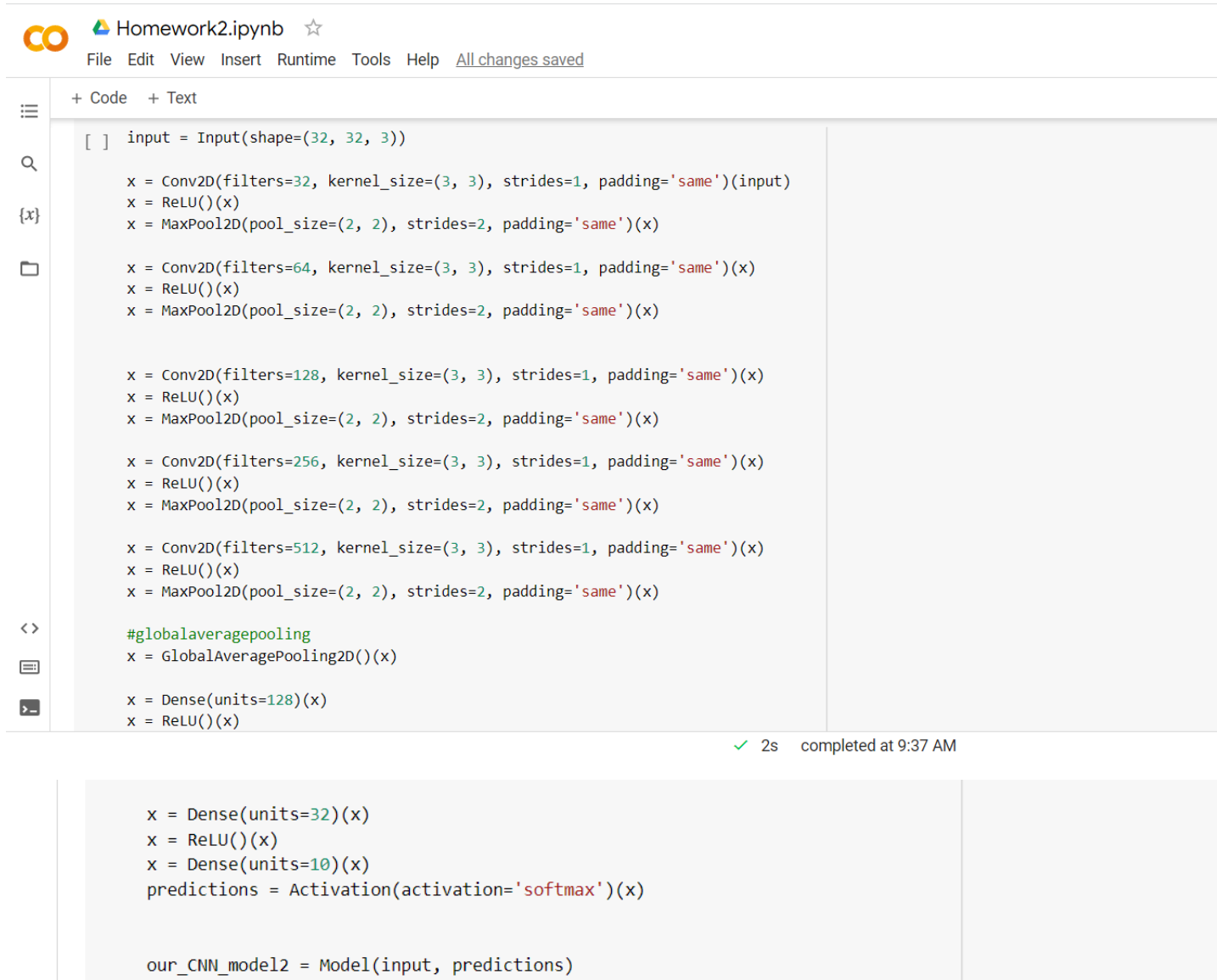


### Evaluate the model

```
[ ] our_CNN_model.evaluate(X_testData, cat_y_testData)
```

313/313 [=====] - 1s 4ms/step - loss: 1.2077 - accuracy: 0.7678  
[1.2077003717422485, 0.767799973487854]

## قسمت دوم : پیاده سازی لایه های پنهان با عمق بیشتر



```
[ ] input = Input(shape=(32, 32, 3))

x = Conv2D(filters=32, kernel_size=(3, 3), strides=1, padding='same')(input)
x = ReLU()(x)
x = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(x)

x = Conv2D(filters=64, kernel_size=(3, 3), strides=1, padding='same')(x)
x = ReLU()(x)
x = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(x)

x = Conv2D(filters=128, kernel_size=(3, 3), strides=1, padding='same')(x)
x = ReLU()(x)
x = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(x)

x = Conv2D(filters=256, kernel_size=(3, 3), strides=1, padding='same')(x)
x = ReLU()(x)
x = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(x)

x = Conv2D(filters=512, kernel_size=(3, 3), strides=1, padding='same')(x)
x = ReLU()(x)
x = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(x)

#globalaveragepooling
x = GlobalAveragePooling2D()(x)

x = Dense(units=128)(x)
x = ReLU()(x)

x = Dense(units=32)(x)
x = ReLU()(x)
x = Dense(units=10)(x)
predictions = Activation(activation='softmax')(x)

our_CNN_model2 = Model(input, predictions)
```

در این لایه تعداد بلوک های لایه های پنهان را افزایش دادیم. از طرف دیگر تعداد لایه های فولی کانکتد را هم افزایش دادیم تا مدل عمیق تر شود. سپس مدل را دوباره ترین کردیم.



Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

```
def train_model(x_train_data, y_train_data, epochs=50, batch_size=32, validation_data=(x_test_data, y_test_data)):
```

```
Epoch 1/50
1563/1563 [=====] - 12s 7ms/step - loss: 1.5562 - accuracy: 0.4194 - val_loss: 1.1465 - val_accuracy: 0.5885
Epoch 2/50
1563/1563 [=====] - 10s 7ms/step - loss: 1.0516 - accuracy: 0.6266 - val_loss: 0.9858 - val_accuracy: 0.6554
Epoch 3/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.8297 - accuracy: 0.7104 - val_loss: 0.8603 - val_accuracy: 0.7035
Epoch 4/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.6944 - accuracy: 0.7584 - val_loss: 0.8427 - val_accuracy: 0.7255
Epoch 5/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.5932 - accuracy: 0.7913 - val_loss: 0.8069 - val_accuracy: 0.7285
Epoch 6/50
1563/1563 [=====] - 10s 7ms/step - loss: 0.5053 - accuracy: 0.8221 - val_loss: 0.8228 - val_accuracy: 0.7413
Epoch 7/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.4203 - accuracy: 0.8542 - val_loss: 0.8807 - val_accuracy: 0.7194
Epoch 8/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.3551 - accuracy: 0.8751 - val_loss: 0.8685 - val_accuracy: 0.7455
Epoch 9/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.2976 - accuracy: 0.8946 - val_loss: 1.0104 - val_accuracy: 0.7312
Epoch 10/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.2518 - accuracy: 0.9119 - val_loss: 1.0410 - val_accuracy: 0.7233
Epoch 11/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.2243 - accuracy: 0.9213 - val_loss: 1.1103 - val_accuracy: 0.7180
Epoch 12/50
1563/1563 [=====] - 10s 7ms/step - loss: 0.1882 - accuracy: 0.9352 - val_loss: 1.1567 - val_accuracy: 0.7431
Epoch 13/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.1662 - accuracy: 0.9426 - val_loss: 1.2379 - val_accuracy: 0.7190
Epoch 14/50
1563/1563 [=====] - 10s 7ms/step - loss: 0.1623 - accuracy: 0.9460 - val_loss: 1.2518 - val_accuracy: 0.7293
Epoch 15/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.1421 - accuracy: 0.9523 - val_loss: 1.3186 - val_accuracy: 0.7336
```

✓ 2s completed at 9:37 AM



Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

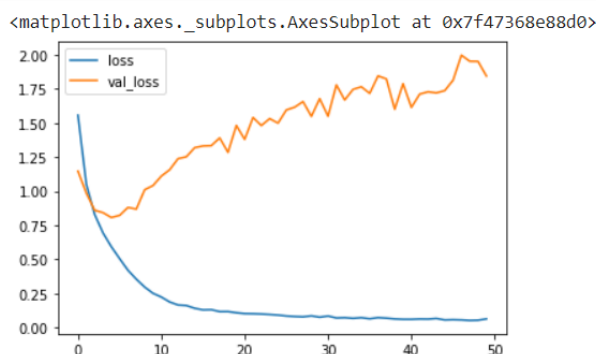
```
Epoch 37/50
1563/1563 [=====] - 10s 7ms/step - loss: 0.0735 - accuracy: 0.9780 - val_loss: 1.8445 - val_accuracy: 0.7224
Epoch 38/50
1563/1563 [=====] - 10s 7ms/step - loss: 0.0696 - accuracy: 0.9788 - val_loss: 1.8223 - val_accuracy: 0.7418
Epoch 39/50
1563/1563 [=====] - 10s 7ms/step - loss: 0.0641 - accuracy: 0.9805 - val_loss: 1.6001 - val_accuracy: 0.7365
Epoch 40/50
1563/1563 [=====] - 10s 7ms/step - loss: 0.0615 - accuracy: 0.9810 - val_loss: 1.7876 - val_accuracy: 0.7397
Epoch 41/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0614 - accuracy: 0.9816 - val_loss: 1.6135 - val_accuracy: 0.7431
Epoch 42/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0636 - accuracy: 0.9810 - val_loss: 1.7113 - val_accuracy: 0.7323
Epoch 43/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0630 - accuracy: 0.9814 - val_loss: 1.7279 - val_accuracy: 0.7411
Epoch 44/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0675 - accuracy: 0.9805 - val_loss: 1.7200 - val_accuracy: 0.7450
Epoch 45/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0565 - accuracy: 0.9827 - val_loss: 1.7364 - val_accuracy: 0.7225
Epoch 46/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0584 - accuracy: 0.9826 - val_loss: 1.8127 - val_accuracy: 0.7433
Epoch 47/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0566 - accuracy: 0.9835 - val_loss: 1.9963 - val_accuracy: 0.7372
Epoch 48/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0530 - accuracy: 0.9844 - val_loss: 1.9519 - val_accuracy: 0.7304
Epoch 49/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0546 - accuracy: 0.9842 - val_loss: 1.9505 - val_accuracy: 0.7356
Epoch 50/50
1563/1563 [=====] - 10s 6ms/step - loss: 0.0643 - accuracy: 0.9816 - val_loss: 1.8434 - val_accuracy: 0.7324
<keras.callbacks.History at 0x7f478e1b1510>
```

✓ 2s completed at 9:37 AM



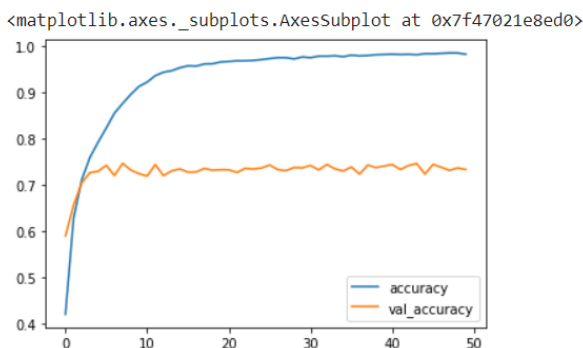
همان طور که دیده میشود نورون در عین حال که نورون های بیشتری را شامل میشد و از گلوبال اورج پولینگ استفاده شد ولی بهبودی در ولیدیشن اکیورسی ایجاد نشد و حتی مدل بدتر هم شد.

```
[ ] plot_history = pd.DataFrame(our_CNN_model2.history.history)
plot_history[['loss', 'val_loss']].plot()
```



✓ 2s completed at 9:37 AM

```
[ ] plot_history[['accuracy', 'val_accuracy']].plot()
```



در این مسئله چون تعداد نمونه های ما به اون صورت زیاد نیست وقتی لایه ها را زیاد میکنیم در هر سری لایه کانولوشن می آید یک بخشی از فیچرها را بدست میاورد. وقتی لایه ها بیشتر شود فیچرها ریزتر میشود و ممکن است در ترین شدن یا به اورفیتینگ برسیم یا اینکه نتیجه عکس دهد و اکیورسی کم شود. از طرف دیگر چون آر جی بی است اگر لایه از یک حدی کمتر شود نمیتانیم فیچرهای درستی استخراج کنیم.

در مرحله ی فلت کردن باز هم اگر لایه های فولی کاکتد اضافه کنیم، چون فیچر هایی که لایه آخر کانولوشن آخر بدست آورده است به آن صورت زیاد نیست در هر ایپاک اپدیت های درستی نمیدهد.

 Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

⋮

+ Code + Text

🔍

▼ Evaluate the model

{x}

[ ] our\_CNN\_model2.evaluate(X\_testData, cat\_y\_testData)

📁

313/313 [=====] - 1s 4ms/step - loss: 1.8434 - accuracy: 0.7324  
[1.8434138298034668, 0.7324000000953674]

## قسمت سوم: استفاده از معماری های دیگر برای بهینه شدن مدل

در این قسمت با استفاده از ارلی استاپینگ مانع از اور فیتینگ میشویم. به این صورت که یا روی ماکسیمم اکوریسی میگذاریم و تا ده تا ایپاک برای مثال چک میکنیم که اگر بیشتر از ان نشد ترینینگ را متوقف کند یا روی مینیمم مقدار لاس قرار میدهیم.

```
Homework2.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[13] our_CNN_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
our_CNN_model.fit(x=X_trainData, y=cat_y_trainData, epochs=40, batch_size=32,
validation_data=(X_testData, cat_y_testData), callbacks=[early_stopping])

Epoch 1/40
1563/1563 [=====] - 13s 7ms/step - loss: 1.4843 - accuracy: 0.4553 - val_loss: 1.0891 - val_accuracy: 0.6037
Epoch 2/40
1563/1563 [=====] - 9s 6ms/step - loss: 0.9856 - accuracy: 0.6503 - val_loss: 0.9061 - val_accuracy: 0.6775
Epoch 3/40
1563/1563 [=====] - 10s 6ms/step - loss: 0.7936 - accuracy: 0.7194 - val_loss: 0.7644 - val_accuracy: 0.7370
Epoch 4/40
1563/1563 [=====] - 9s 6ms/step - loss: 0.6757 - accuracy: 0.7618 - val_loss: 0.7111 - val_accuracy: 0.7538
Epoch 5/40
1563/1563 [=====] - 9s 6ms/step - loss: 0.5926 - accuracy: 0.7930 - val_loss: 0.6938 - val_accuracy: 0.7654
Epoch 6/40
1563/1563 [=====] - 9s 6ms/step - loss: 0.5276 - accuracy: 0.8147 - val_loss: 0.6829 - val_accuracy: 0.7711
Epoch 7/40
1563/1563 [=====] - 9s 6ms/step - loss: 0.4705 - accuracy: 0.8338 - val_loss: 0.7276 - val_accuracy: 0.7609
Epoch 8/40
1563/1563 [=====] - 9s 6ms/step - loss: 0.4186 - accuracy: 0.8525 - val_loss: 0.6894 - val_accuracy: 0.7797
Epoch 9/40
1563/1563 [=====] - 10s 6ms/step - loss: 0.3735 - accuracy: 0.8683 - val_loss: 0.7058 - val_accuracy: 0.7787
Epoch 10/40
1563/1563 [=====] - 10s 6ms/step - loss: 0.3422 - accuracy: 0.8789 - val_loss: 0.7332 - val_accuracy: 0.7782
Epoch 11/40
1563/1563 [=====] - 10s 7ms/step - loss: 0.3072 - accuracy: 0.8906 - val_loss: 0.7875 - val_accuracy: 0.7695
Epoch 12/40
1563/1563 [=====] - 9s 6ms/step - loss: 0.2908 - accuracy: 0.8964 - val_loss: 0.7776 - val_accuracy: 0.7784
Epoch 13/40
1563/1563 [=====] - 10s 6ms/step - loss: 0.2778 - accuracy: 0.9020 - val_loss: 0.8148 - val_accuracy: 0.7650
✓ 2s completed at 9:37 AM
```

در پلات ها میبینیم که تقریباً قبل از اورفیتیگ شدن فرایند ترین شدن متوقف شده و داده ها اورفیت نشدند.

co

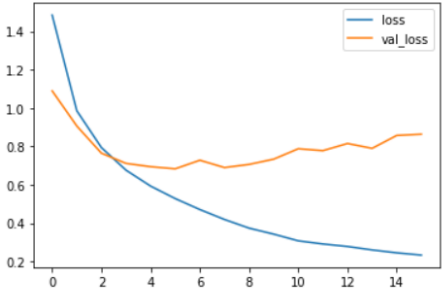
Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

plot model loss history

[14] plot\_history = pd.DataFrame(our\_CNN\_model.history.history)  
plot\_history[['loss', 'val\_loss']].plot()

<matplotlib.axes.\_subplots.AxesSubplot at 0x7faf420eb050>  


min of loss history

[15] np.min(our\_CNN\_model.history.history['loss'])

0.23287376761436462

✓ 2s completed at 9:37 AM

co

Homework2.ipynb ☆

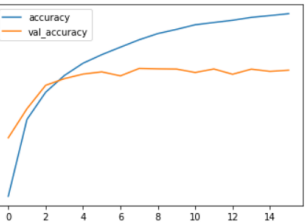
File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share ⚙️ k

+ Code + Text

plot model accuracy

[16] plot\_history = pd.DataFrame(our\_CNN\_model.history.history)  
plot\_history[['accuracy', 'val\_accuracy']].plot()

<matplotlib.axes.\_subplots.AxesSubplot at 0x7faf2b701090>  



max of validation accuracy

[17] np.max(our\_CNN\_model.history.history['val\_accuracy'])

0.779699981212616

✓ 2s completed at 9:37 AM

و در نهایت یکی از بهترین ولیدیشن اکپوریزی هارا خواهیم داشت:

 Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

☰

+ Code + Text

🔍

▼ Evaluate the model


{x}

✓ 1s

[18] our\_CNN\_model.evaluate(X\_testData, cat\_y\_testData)

313/313 [=====] - 1s 4ms/step - loss: 0.6829 - accuracy: 0.7711  
[0.682878851890564, 0.7710999846458435]

میتوانیم برای این مدل کانفیوژن ماتریکس تشکیل دهیم.

 Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

▼ printing classification report

{x}

✓  
0s


[20] from sklearn.metrics import confusion\_matrix, classification\_report

✓  
0s

[21] print(classification\_report(Y\_testData, predictions\_sparse))

	precision	recall	f1-score	support
0	0.75	0.84	0.79	1000
1	0.86	0.91	0.88	1000
2	0.72	0.62	0.67	1000
3	0.60	0.65	0.62	1000
4	0.68	0.80	0.74	1000
5	0.78	0.58	0.67	1000
6	0.86	0.76	0.81	1000
7	0.76	0.85	0.80	1000
8	0.90	0.83	0.87	1000
9	0.84	0.87	0.86	1000
accuracy			0.77	10000
macro avg	0.78	0.77	0.77	10000
weighted avg	0.78	0.77	0.77	10000

<>



▼ use confusion matrix

✓ 2s completed at 9:37 AM



+ Code + Text

▼ use confusion matrix

{x}

✓ [22] confusion\_matrix(Y\_testData, predictions\_sparse)




```
array([[843, 20, 27, 14, 19, 0, 0, 13, 38, 26],
       [10, 914, 1, 1, 2, 1, 5, 1, 15, 50],
       [97, 5, 620, 50, 93, 37, 48, 31, 7, 12],
       [25, 9, 49, 650, 77, 76, 34, 56, 7, 17],
       [18, 1, 44, 44, 799, 11, 20, 54, 3, 6],
       [14, 4, 46, 199, 54, 579, 10, 89, 3, 2],
       [11, 4, 38, 83, 74, 16, 756, 8, 5, 5],
       [15, 4, 22, 28, 46, 16, 1, 849, 2, 17],
       [68, 40, 11, 11, 4, 1, 1, 5, 835, 24],
       [27, 66, 4, 9, 5, 4, 1, 8, 10, 866]])
```

این ماتریکس به این صورت است که به غیر از قطر که مقادیر بالایی دارد هر رقمی غیر این ها مقدار پرتی داشته باشد یعنی مدل در آنجا ها خوب تشخیص نداده است. برای مثال در این مدل چون سگ و گربه در کلاس 4 و 6 که سگ و گربه هستند (اعداد 199 و 83) نتوانسته به خوبی از هم تشخیص دهد و مقادیر بالاتری گرفته اند.

قسمت چهارم: استفاده از بچ نرمالیزیشن و تکنیک دراپ اوت

در بچ نرمالیزیشن میدانیم به این صورت عمل میشود که بعد از هر لایه کانولوشن دیتاها را در رنج خاصی نرمال کرد و به تابع رلو میداد تا دیتای پرت زیادی تولید نشود و از طرفی دیتا از دست نرود.

 Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

⋮

🔍

{x}

📁

<>

☰

➤

```
[24] input = Input(shape=(32, 32, 3))

x = Conv2D(filters=32, kernel_size=(5, 5), strides=1, padding='same')(input)
x = BatchNormalization()(x)
x = ReLU()(x)
x = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(x)

x = Conv2D(filters=64, kernel_size=(3, 3), strides=1, padding='same')(x)
x = BatchNormalization()(x)
x = ReLU()(x)
x = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(x)

x = Conv2D(filters=128, kernel_size=(3, 3), strides=1, padding='same')(x)
x = BatchNormalization()(x)
x = ReLU()(x)
x = MaxPool2D(pool_size=(2, 2), strides=2, padding='same')(x)

x = Flatten()(x)

x = Dense(units=128)(x)
x = ReLU()(x)
x = Dropout(0.3)(x)
x = Dense(units=10)(x)
predictions = Activation(activation='softmax')(x)

our_CNN_model3 = Model(input, predictions)
```

✓ 2s completed at 9:37 AM



```
Homework2.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[27] Epoch 1/40
1563/1563 [=====] - 10s 6ms/step - loss: 1.6155 - accuracy: 0.3962 - val_loss: 1.3927 - val_accuracy: 0.4909
Epoch 2/40
1563/1563 [=====] - 8s 5ms/step - loss: 1.3231 - accuracy: 0.5084 - val_loss: 1.2277 - val_accuracy: 0.5673
Epoch 3/40
1563/1563 [=====] - 8s 5ms/step - loss: 1.2022 - accuracy: 0.5578 - val_loss: 1.2119 - val_accuracy: 0.5672
Epoch 4/40
1563/1563 [=====] - 8s 5ms/step - loss: 1.1183 - accuracy: 0.5869 - val_loss: 1.0542 - val_accuracy: 0.6363
Epoch 5/40
1563/1563 [=====] - 8s 5ms/step - loss: 1.0417 - accuracy: 0.6198 - val_loss: 1.1261 - val_accuracy: 0.6209
Epoch 6/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.9661 - accuracy: 0.6481 - val_loss: 1.2665 - val_accuracy: 0.5805
Epoch 7/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.9125 - accuracy: 0.6642 - val_loss: 1.1435 - val_accuracy: 0.6195
Epoch 8/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.8638 - accuracy: 0.6857 - val_loss: 0.7977 - val_accuracy: 0.7292
Epoch 9/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.8119 - accuracy: 0.7058 - val_loss: 0.9814 - val_accuracy: 0.6775
Epoch 10/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.7792 - accuracy: 0.7205 - val_loss: 0.8008 - val_accuracy: 0.7222
Epoch 11/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.7352 - accuracy: 0.7336 - val_loss: 0.7697 - val_accuracy: 0.7407
Epoch 12/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.6948 - accuracy: 0.7498 - val_loss: 1.7817 - val_accuracy: 0.5337
Epoch 13/40
1563/1563 [=====] - 12s 8ms/step - loss: 0.6614 - accuracy: 0.7616 - val_loss: 0.8624 - val_accuracy: 0.7194
Epoch 14/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.6303 - accuracy: 0.7714 - val_loss: 0.9406 - val_accuracy: 0.7128
Epoch 15/40
1563/1563 [=====] - 9s 6ms/step - loss: 0.5987 - accuracy: 0.7857 - val_loss: 0.7737 - val_accuracy: 0.7478
Epoch 16/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.5526 - accuracy: 0.8003 - val_loss: 0.8695 - val_accuracy: 0.7349
Epoch 17/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.5240 - accuracy: 0.8098 - val_loss: 0.8217 - val_accuracy: 0.7391
Epoch 18/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.4905 - accuracy: 0.8239 - val_loss: 0.8964 - val_accuracy: 0.7298
Epoch 19/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.4735 - accuracy: 0.8315 - val_loss: 0.7780 - val_accuracy: 0.7628
Epoch 20/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.4428 - accuracy: 0.8425 - val_loss: 0.8610 - val_accuracy: 0.7491
Epoch 21/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.4120 - accuracy: 0.8517 - val_loss: 1.0185 - val_accuracy: 0.7136
Epoch 22/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.3812 - accuracy: 0.8608 - val_loss: 0.9406 - val_accuracy: 0.7128
Epoch 23/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.3504 - accuracy: 0.8700 - val_loss: 0.8624 - val_accuracy: 0.7194
Epoch 24/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.3196 - accuracy: 0.8792 - val_loss: 0.7843 - val_accuracy: 0.7260
Epoch 25/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.2888 - accuracy: 0.8884 - val_loss: 0.7062 - val_accuracy: 0.7326
Epoch 26/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.2580 - accuracy: 0.8976 - val_loss: 0.6281 - val_accuracy: 0.7392
Epoch 27/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.2272 - accuracy: 0.9068 - val_loss: 0.5500 - val_accuracy: 0.7458
Epoch 28/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.1964 - accuracy: 0.9160 - val_loss: 0.4719 - val_accuracy: 0.7524
Epoch 29/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.1656 - accuracy: 0.9252 - val_loss: 0.3938 - val_accuracy: 0.7590
Epoch 30/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.1348 - accuracy: 0.9344 - val_loss: 0.3157 - val_accuracy: 0.7656
Epoch 31/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.1040 - accuracy: 0.9436 - val_loss: 0.2376 - val_accuracy: 0.7722
Epoch 32/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0732 - accuracy: 0.9528 - val_loss: 0.1595 - val_accuracy: 0.7788
Epoch 33/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0424 - accuracy: 0.9620 - val_loss: 0.0814 - val_accuracy: 0.7854
Epoch 34/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0116 - accuracy: 0.9712 - val_loss: 0.0033 - val_accuracy: 0.7920
Epoch 35/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9804 - val_loss: 0.0000 - val_accuracy: 0.7986
Epoch 36/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9896 - val_loss: 0.0000 - val_accuracy: 0.8052
Epoch 37/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9988 - val_loss: 0.0000 - val_accuracy: 0.8118
Epoch 38/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9988 - val_loss: 0.0000 - val_accuracy: 0.8118
Epoch 39/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9988 - val_loss: 0.0000 - val_accuracy: 0.8118
Epoch 40/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9988 - val_loss: 0.0000 - val_accuracy: 0.8118
2s completed at 9:37 AM
```

```
Homework2.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[27] Epoch 8/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.9125 - accuracy: 0.6642 - val_loss: 1.1435 - val_accuracy: 0.6195
Epoch 9/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.8638 - accuracy: 0.6857 - val_loss: 0.7977 - val_accuracy: 0.7292
Epoch 10/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.8119 - accuracy: 0.7058 - val_loss: 0.9814 - val_accuracy: 0.6775
Epoch 11/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.7792 - accuracy: 0.7205 - val_loss: 0.8008 - val_accuracy: 0.7222
Epoch 12/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.7352 - accuracy: 0.7336 - val_loss: 0.7697 - val_accuracy: 0.7407
Epoch 13/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.6948 - accuracy: 0.7498 - val_loss: 1.7817 - val_accuracy: 0.5337
Epoch 14/40
1563/1563 [=====] - 12s 8ms/step - loss: 0.6614 - accuracy: 0.7616 - val_loss: 0.8624 - val_accuracy: 0.7194
Epoch 15/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.6303 - accuracy: 0.7714 - val_loss: 0.9406 - val_accuracy: 0.7128
Epoch 16/40
1563/1563 [=====] - 9s 6ms/step - loss: 0.5987 - accuracy: 0.7857 - val_loss: 0.7737 - val_accuracy: 0.7478
Epoch 17/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.5526 - accuracy: 0.8003 - val_loss: 0.8695 - val_accuracy: 0.7349
Epoch 18/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.5240 - accuracy: 0.8098 - val_loss: 0.8217 - val_accuracy: 0.7391
Epoch 19/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.4905 - accuracy: 0.8239 - val_loss: 0.8964 - val_accuracy: 0.7298
Epoch 20/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.4735 - accuracy: 0.8315 - val_loss: 0.7780 - val_accuracy: 0.7628
Epoch 21/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.4428 - accuracy: 0.8425 - val_loss: 0.8610 - val_accuracy: 0.7491
Epoch 22/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.4120 - accuracy: 0.8517 - val_loss: 1.0185 - val_accuracy: 0.7136
Epoch 23/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.3812 - accuracy: 0.8608 - val_loss: 0.9406 - val_accuracy: 0.7128
Epoch 24/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.3504 - accuracy: 0.8700 - val_loss: 0.8624 - val_accuracy: 0.7194
Epoch 25/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.3196 - accuracy: 0.8792 - val_loss: 0.7843 - val_accuracy: 0.7260
Epoch 26/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.2888 - accuracy: 0.8884 - val_loss: 0.7062 - val_accuracy: 0.7326
Epoch 27/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.2580 - accuracy: 0.8976 - val_loss: 0.6281 - val_accuracy: 0.7392
Epoch 28/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.2272 - accuracy: 0.9068 - val_loss: 0.5500 - val_accuracy: 0.7458
Epoch 29/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.1964 - accuracy: 0.9160 - val_loss: 0.4719 - val_accuracy: 0.7524
Epoch 30/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.1656 - accuracy: 0.9252 - val_loss: 0.3938 - val_accuracy: 0.7590
Epoch 31/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.1348 - accuracy: 0.9344 - val_loss: 0.3157 - val_accuracy: 0.7656
Epoch 32/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.1040 - accuracy: 0.9436 - val_loss: 0.2376 - val_accuracy: 0.7722
Epoch 33/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0732 - accuracy: 0.9528 - val_loss: 0.1595 - val_accuracy: 0.7788
Epoch 34/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0424 - accuracy: 0.9620 - val_loss: 0.0814 - val_accuracy: 0.7854
Epoch 35/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0116 - accuracy: 0.9712 - val_loss: 0.0033 - val_accuracy: 0.7920
Epoch 36/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9804 - val_loss: 0.0000 - val_accuracy: 0.7986
Epoch 37/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9896 - val_loss: 0.0000 - val_accuracy: 0.8052
Epoch 38/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9988 - val_loss: 0.0000 - val_accuracy: 0.8118
Epoch 39/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9988 - val_loss: 0.0000 - val_accuracy: 0.8118
Epoch 40/40
1563/1563 [=====] - 8s 5ms/step - loss: 0.0000 - accuracy: 0.9988 - val_loss: 0.0000 - val_accuracy: 0.8118
2s completed at 9:37 AM
```

همان طور که میبینیم ارلی استاپینگ مانع از اورفیتینگ میشود و از طرف دیگر اکيوريسى خوبى داريم که با وليديشن اکيوريسى اختلاف زيادى ندارد پس مدل ما مدل خوبى با اين تغييرات شد.



Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)



+ Code + Text



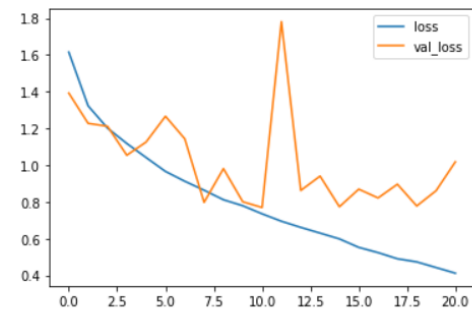
{x}



### plot model loss history

```
[29] plot_history3[['loss', 'val_loss']].plot()
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7faea0a42d90>



Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)



+ Code + Text



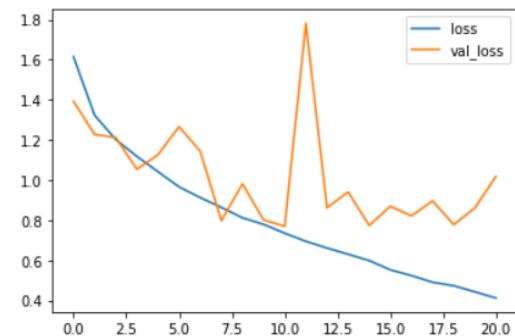
{x}



### plot model loss history

```
[29] plot_history3[['loss', 'val_loss']].plot()
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7faea0a42d90>



و در نهایت اگر ولیدیشن اکیورسی را حساب کنیم عدد خوبی بدست خواهد آمد.



Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)



+ Code + Text



▼ Evaluate the data

{x}




✓ [30] our\_CNN\_model3.evaluate(X\_testData, cat\_y\_testData)

1s

313/313 [=====] - 1s 3ms/step - loss: 0.7697 - accuracy: 0.7407  
[0.7697134017944336, 0.7407000064849854]

## قسمت پنجم: استفاده از قسمتی از داده

در برخی مسائل میبینیم که استفاده از بخشی از دیتا میتواند دقت ما را بالا ببرد در حالی که تعداد دیتای ما کمتر است و ترین کمتر طول میکشد. در این قسمت از کد میتوانیم ببینیم که با اینکه بخشی از دیتا را در نظر گرفتیم دقت خوبی به ما میدهد. البته این بخش کوچک تر از دیتا نمیتواند در این مسئله خیلی کوچک باشد چون ابتدای آن سیاه سفید است و باید به نوع دیتا مرتبط است.

 Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

✓ 28s

[55] our\_CNN\_model3.fit(x=X\_trainData, y=cat\_y\_trainData, epochs=40, batch\_size=32, validation\_data=(X\_testData, cat\_y\_testData), callbacks=[early\_stopping])

{x}

Epoch 1/40  
313/313 [=====] - 3s 8ms/step - loss: 0.4471 - accuracy: 0.8349 - val\_loss: 0.8529 - val\_accuracy:  
Epoch 2/40  
313/313 [=====] - 2s 7ms/step - loss: 0.4158 - accuracy: 0.8478 - val\_loss: 0.8772 - val\_accuracy:  
Epoch 3/40  
313/313 [=====] - 3s 9ms/step - loss: 0.3825 - accuracy: 0.8603 - val\_loss: 0.9930 - val\_accuracy:  
Epoch 4/40  
313/313 [=====] - 2s 7ms/step - loss: 0.3728 - accuracy: 0.8621 - val\_loss: 0.9829 - val\_accuracy:  
Epoch 5/40  
313/313 [=====] - 3s 9ms/step - loss: 0.3307 - accuracy: 0.8800 - val\_loss: 1.1707 - val\_accuracy:  
Epoch 6/40  
313/313 [=====] - 2s 7ms/step - loss: 0.3248 - accuracy: 0.8806 - val\_loss: 1.0778 - val\_accuracy:  
Epoch 7/40  
313/313 [=====] - 2s 7ms/step - loss: 0.3042 - accuracy: 0.8882 - val\_loss: 1.2034 - val\_accuracy:  
Epoch 8/40  
313/313 [=====] - 3s 9ms/step - loss: 0.2952 - accuracy: 0.8923 - val\_loss: 1.1134 - val\_accuracy:  
Epoch 9/40  
313/313 [=====] - 2s 7ms/step - loss: 0.2991 - accuracy: 0.8890 - val\_loss: 1.0843 - val\_accuracy:  
Epoch 10/40  
313/313 [=====] - 3s 9ms/step - loss: 0.2801 - accuracy: 0.9004 - val\_loss: 1.0280 - val\_accuracy:  
Epoch 11/40  
313/313 [=====] - 3s 9ms/step - loss: 0.2614 - accuracy: 0.9073 - val\_loss: 1.1662 - val\_accuracy:  
<keras.callbacks.History at 0x7fae34465650>

loss function and accuracy metric

✓ 2s completed at 9:37 AM



+ Code + Text

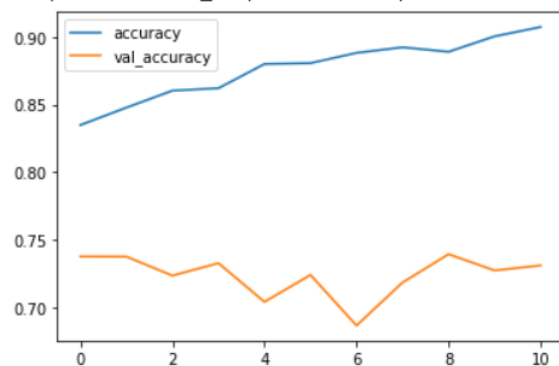
## ▼ loss function and accuracy metric

{x}

```
[56] plot_history3 = pd.DataFrame(our_CNN_model3.history.history)
      plot_history3[['accuracy', 'val_accuracy']].plot()
```



&lt;matplotlib.axes.\_subplots.AxesSubplot at 0x7fae34c2abd0&gt;



+ Code + Text

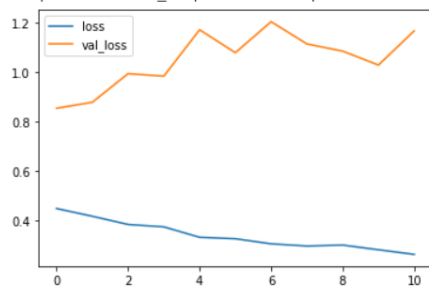
## ▼ plot model loss history

{x}

```
[57] plot_history3[['loss', 'val_loss']].plot()
```



&lt;matplotlib.axes.\_subplots.AxesSubplot at 0x7fae34bab390&gt;





Homework2.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)



+ Code + Text



## ▼ Evaluate the data



✓ [58] our\_CNN\_model3.evaluate(X\_testData, cat\_y\_testData)  
3s



```
313/313 [=====] - 1s 3ms/step - loss: 0.8529 - accuracy: 0.7378  
[0.8528999090194702, 0.7378000020980835]
```



✓ 2s completed at 9:37 AM