



به نام خدا



دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین سری دوم

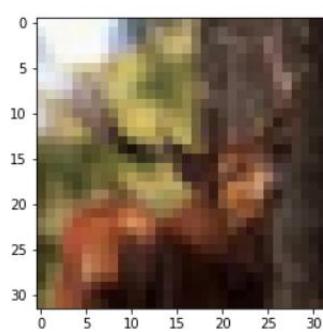
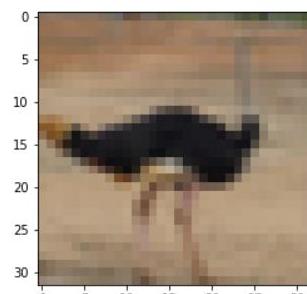
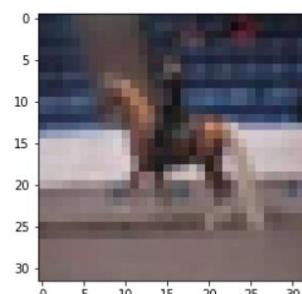
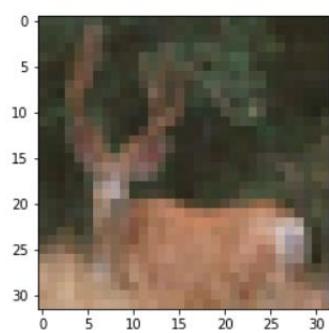
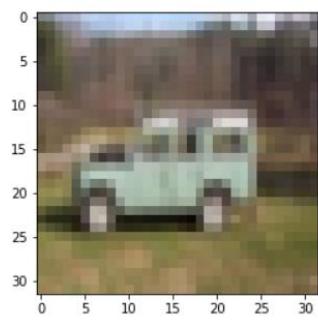
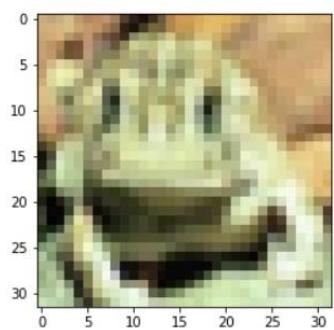
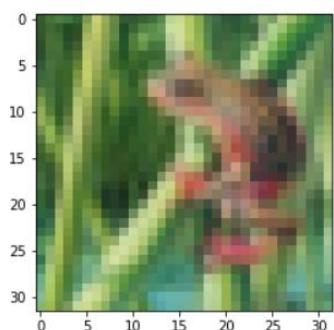
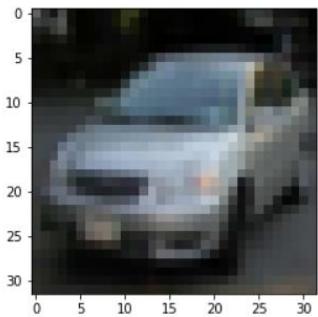
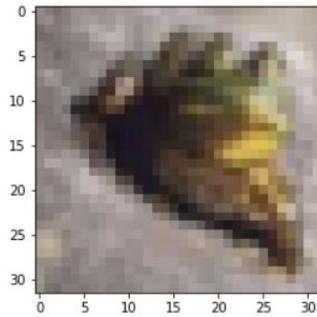
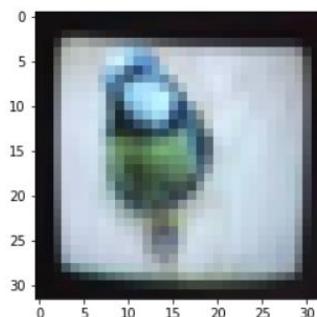
مریم ریاضی	نام و نام خانوادگی
۸۱۰۱۹۷۵۱۸	شماره دانشجویی
1400/1/19	تاریخ ارسال گزارش

فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به روز کنید.)

- ۱ Classification (MLP) – سوال ۱
- ۲۸ MLP(Regression) – سوال ۲
- ۳۳ سوال ۳ – کاهش ابعاد

سوال ۱ – عنوان سوال

۱۰ عکس تصادفی که رسم شدند، به صورت زیر هستند:



(الف)

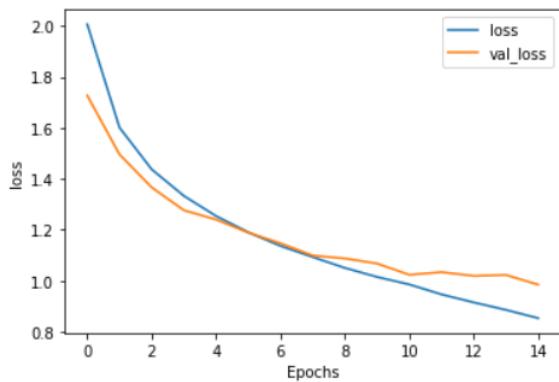
(ب)

پیش پردازش داده ها شامل تبدیل کردن انها به عکس های سیاه و سفید و سپس نرمالایز کردن انها است. عکس های داده شده رنگی هستند و بعد سومی دارند که رنگ هر پیکسل را مشخص می کند. از انجایی که در این سوال به رنگ ها نیازی نداریم، با سیاه و سفید کردن عکس ها، مراحل کار ساده تر می شوند. نرمالایز کردن نیز با کم کردن مقدار کمینه و تقسیم بر تفاوت مقدار بیشینه و کمینه صورت گرفت.

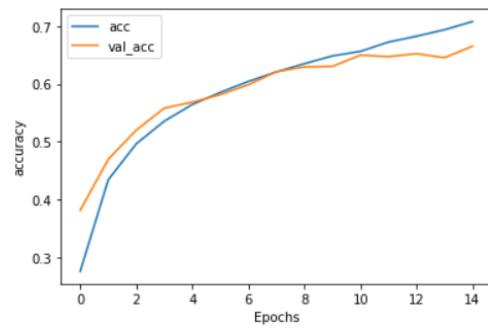
(ت)

```
model = Sequential()
model.add(layers.Conv2D(filters = 32, kernel_size = 2, padding = 'same', activation = 'relu', input_shape = (32, 32, 1)))
model.add(layers.MaxPooling2D(pool_size = 2))
model.add(layers.Conv2D(filters = 64, kernel_size = 2, padding = 'same', activation = 'relu'))
model.add(layers.MaxPooling2D(pool_size = 2))
model.add(layers.Conv2D(filters = 64, kernel_size = 2, padding = 'same', activation = 'relu'))
model.add(layers.MaxPooling2D(pool_size = 2))
model.add(layers.Flatten())
model.add(layers.Dense(256, activation = 'sigmoid', ))
model.add(layers.Dense(10, activation = 'softmax'))
```

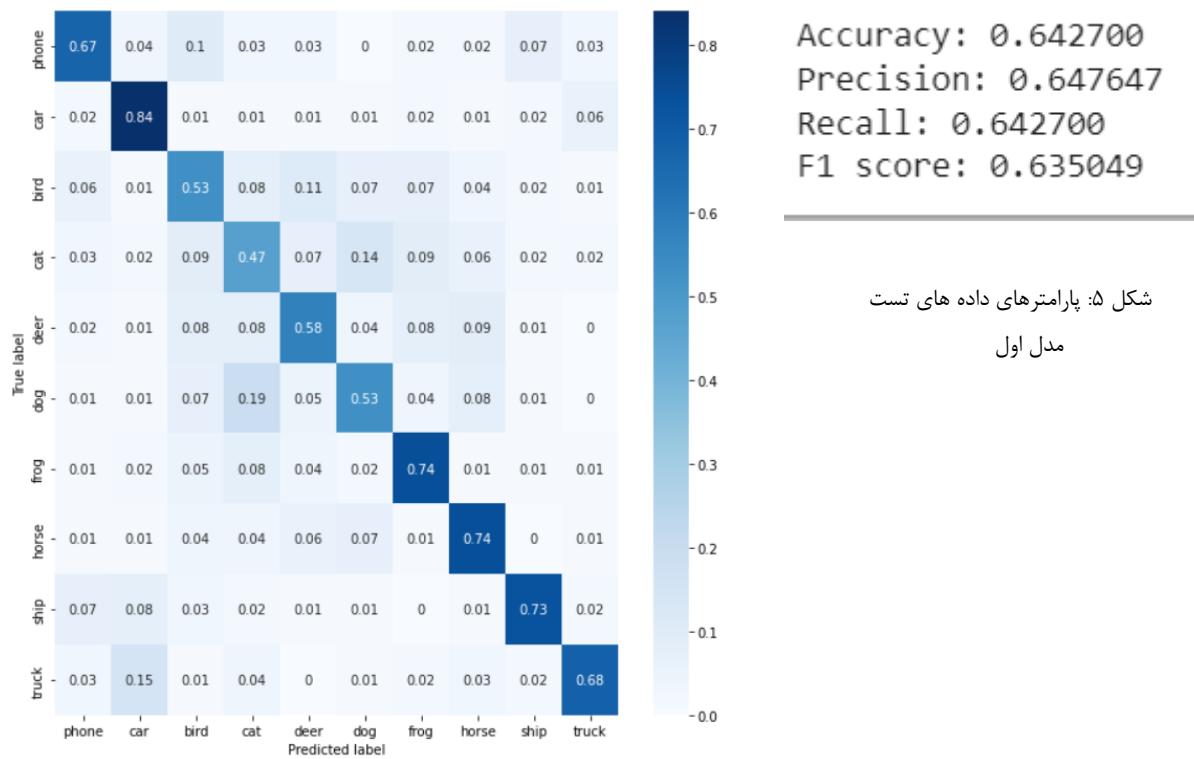
شکل ۱: مدل اول



شکل ۲: نمودار خطا به ازای هر ایپاک
برای مدل اول



شکل ۳: نمودار دقت به ازای هر
ایپاک برای مدل اول



شکل ۵: پارامترهای داده های تست
مدل اول

شکل ۴: ماتریس آشفتگی مدل اول

```

model = Sequential()

model.add(layers.Conv2D(filters = 16, kernel_size = 2, padding = 'same', activation = 'relu', input_shape = (32, 32, 1)))
model.add(layers.MaxPooling2D(pool_size = 2))

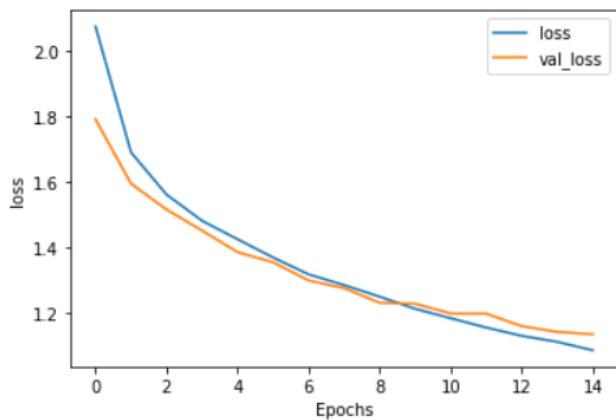
model.add(layers.Conv2D(filters = 32, kernel_size = 2, padding = 'same', activation = 'relu'))
model.add(layers.MaxPooling2D(pool_size = 2))

model.add(layers.Conv2D(filters = 32, kernel_size = 2, padding = 'same', activation = 'relu'))
model.add(layers.MaxPooling2D(pool_size = 2))

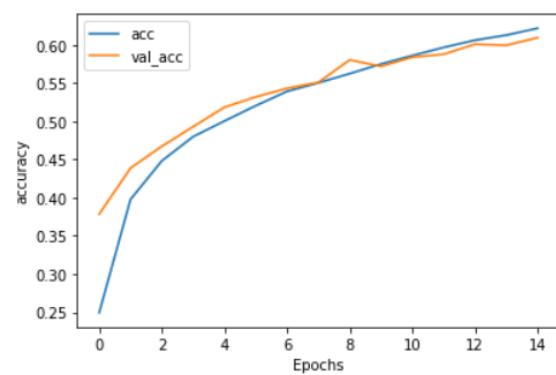
model.add(layers.Flatten())
model.add(layers.Dense(128, activation = 'sigmoid', ))
model.add(layers.Dense(10, activation = 'softmax'))

```

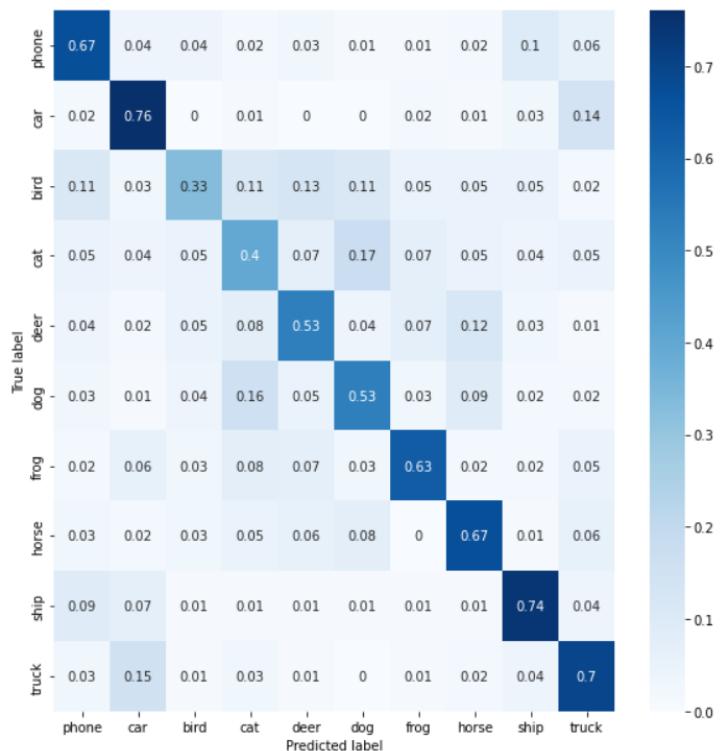
شکل ۶: مدل دوم



شکل ۷: نمودار خطا به ازای هر ایپاک برای
مدل دوم



شکل ۸: نمودار دقت به ازای هر
ایپاک برای مدل دوم



Accuracy: 0.595300
Precision: 0.591433
Recall: 0.595300
F1 score: 0.589110

شکل ۱۰: پارامترهای

داده های تست مدل دوم

شکل ۹: ماتریس آشفتگی مدل دوم

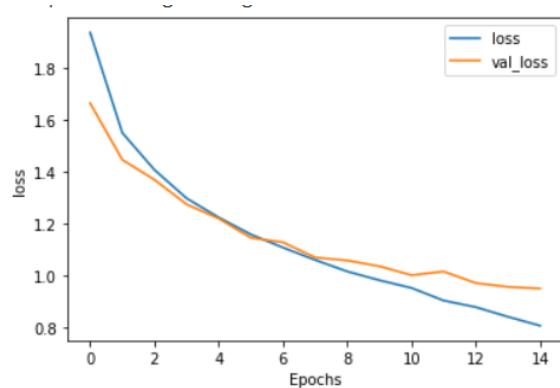
```

model = Sequential()
model.add(layers.Conv2D(filters = 64, kernel_size = 2, padding = 'same', activation = 'relu', input_shape = (32, 32, 1)))
model.add(layers.MaxPooling2D(pool_size = 2))
model.add(layers.Conv2D(filters = 64, kernel_size = 2, padding = 'same', activation = 'relu'))
model.add(layers.MaxPooling2D(pool_size = 2))
model.add(layers.Conv2D(filters = 64, kernel_size = 2, padding = 'same', activation = 'relu'))
model.add(layers.MaxPooling2D(pool_size = 2))

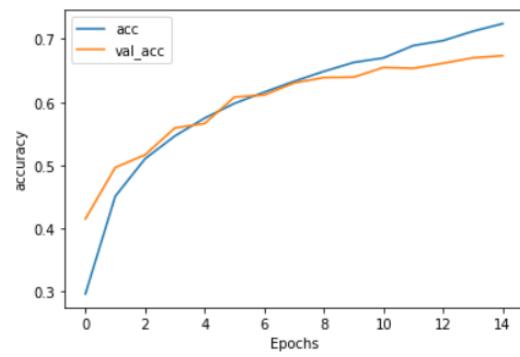
model.add(layers.Flatten())
model.add(layers.Dense(512, activation = 'sigmoid', ))
model.add(layers.Dense(10, activation = 'softmax'))

```

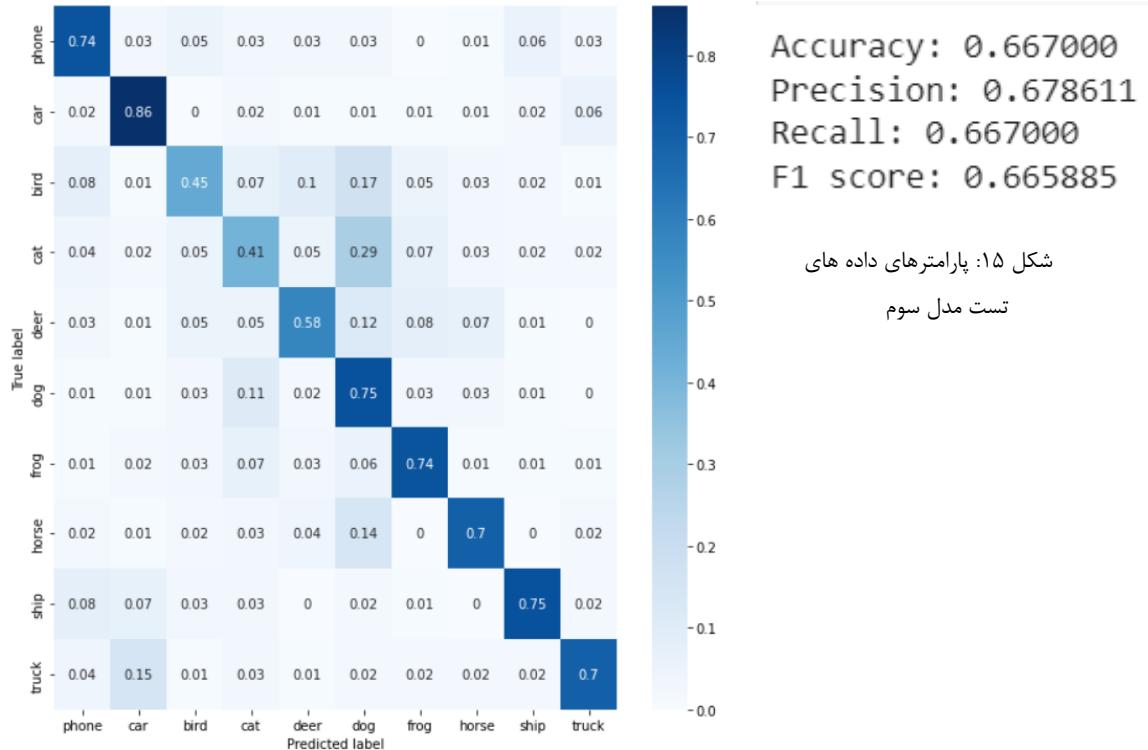
شکل ۱۱: مدل سوم



شکل ۱۲: نمودار خطا به ازای هر ایپاک
برای مدل سوم



شکل ۱۳: نمودار دقت به ازای هر
ایپاک برای مدل سوم



شکل ۱۵: پارامترهای داده های

تست مدل سوم

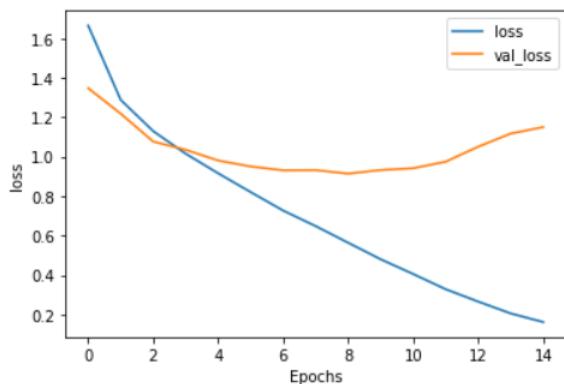
شکل ۱۴: ماتریس آشفتگی مدل سوم

د) بهترین مدل بخش قبل، مدل سوم بود که بیشترین میزان دقت و کمترین میزان خطا را داشت.

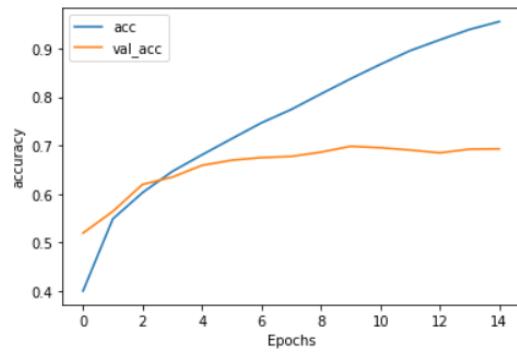
حالت اول:

Batch size = 32

Runtime = 3 minutes



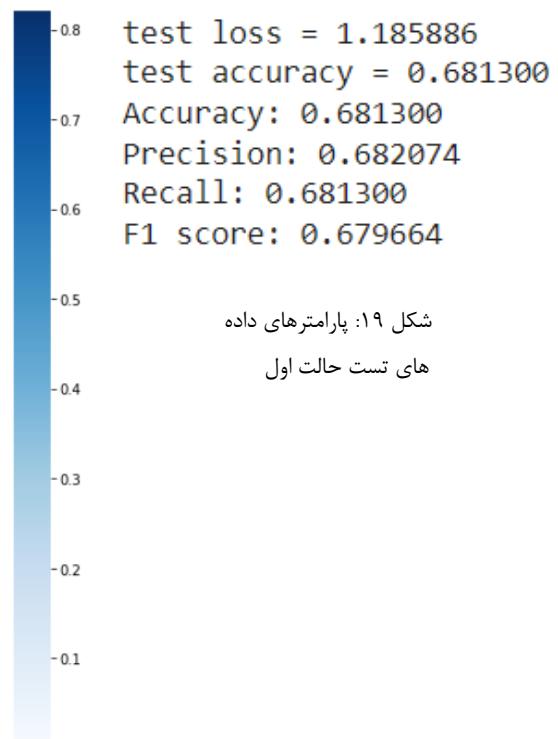
شکل ۱۶: نمودار خطا به ازای هر ایپاک
برای حالت اول



شکل ۱۷: نمودار دقت به ازای هر ایپاک برای
حالت اول



شکل ۱۸: ماتریس آشفتگی حالت اول

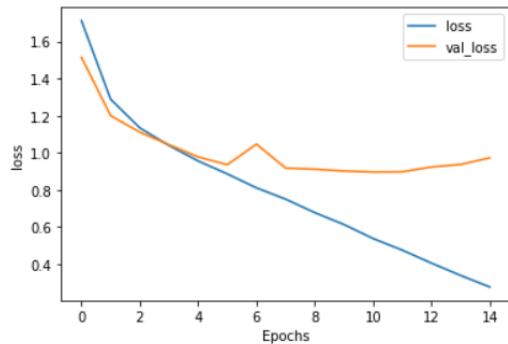


شکل ۱۹: پارامترهای داده
های تست حالت اول

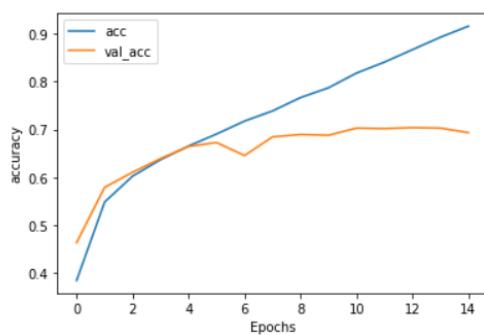
حالت دوم:

Batch size = 64

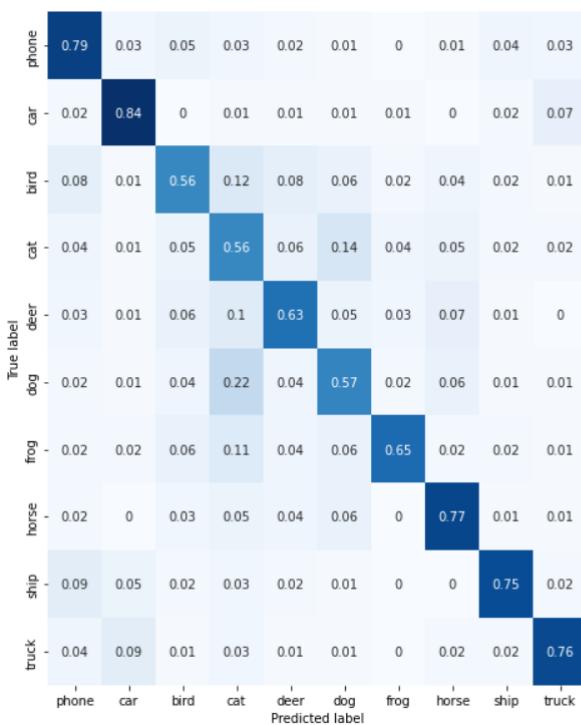
Runtime = 2 minutes



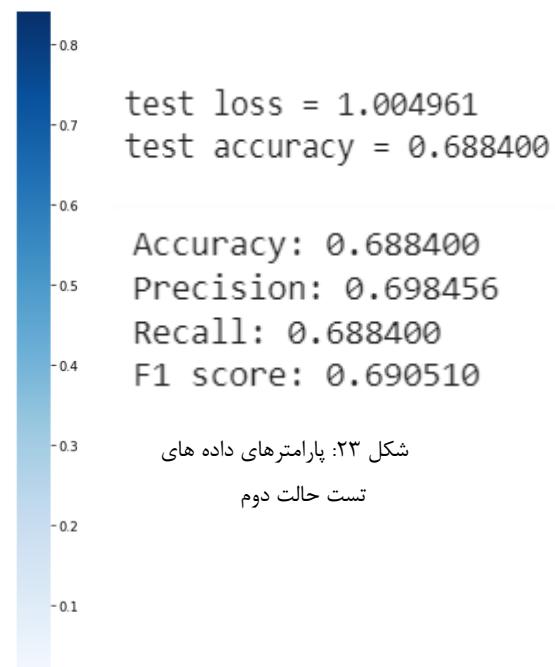
شکل ۲۰: نمودار خطا به ازای هر ایپاک
برای حالت دوم



شکل ۲۱: نمودار دقیقت به ازای هر
ایپاک برای حالت دوم



شکل ۲۲: ماتریس آشفتگی حالت دوم

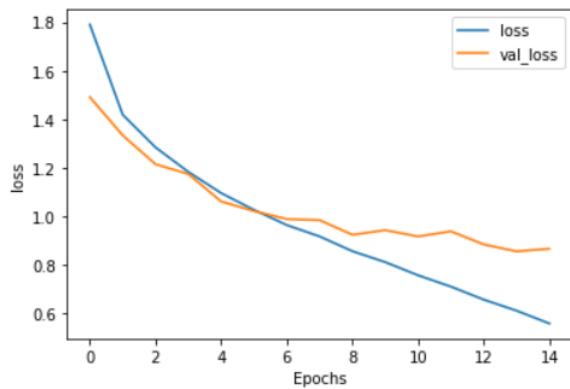


شکل ۲۳: پارامترهای داده های
 تست حالت دوم

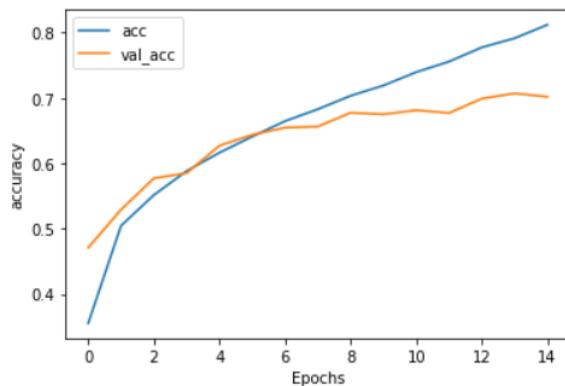
حالت سوم:

Batch size = 128

Runtime = 1 minute



شکل ۲۴: نمودار خطا به ازای هر
ایپاک برای حالت سوم



شکل ۲۵ نمودار دقت به ازای هر
ایپاک برای حالت سوم



شکل ۲۶: ماتریس آشنگی حالت سوم

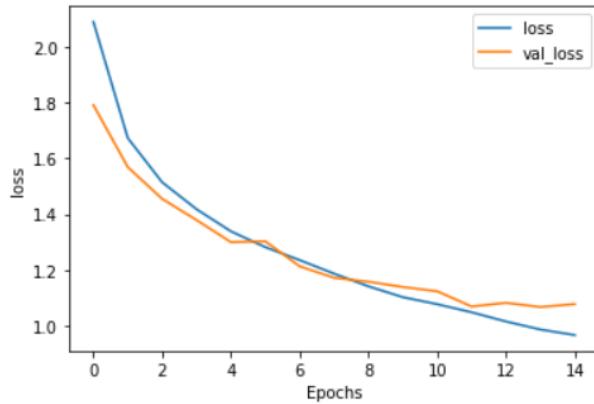
```
test loss = 0.896915
test accuracy = 0.698100
Accuracy: 0.698100
Precision: 0.705716
Recall: 0.698100
F1 score: 0.699457
```

شکل ۲۷: پارامترهای داده های
تست حالت سوم

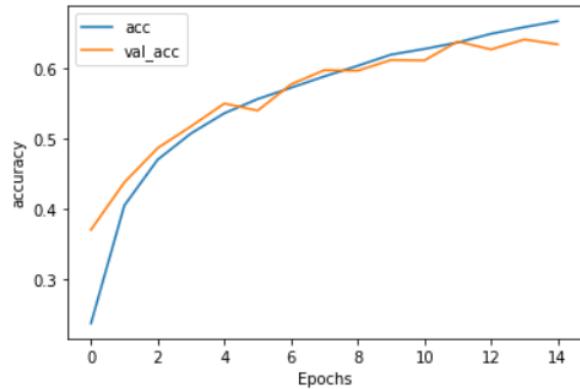
حالت چهارم:

Batch size = 512

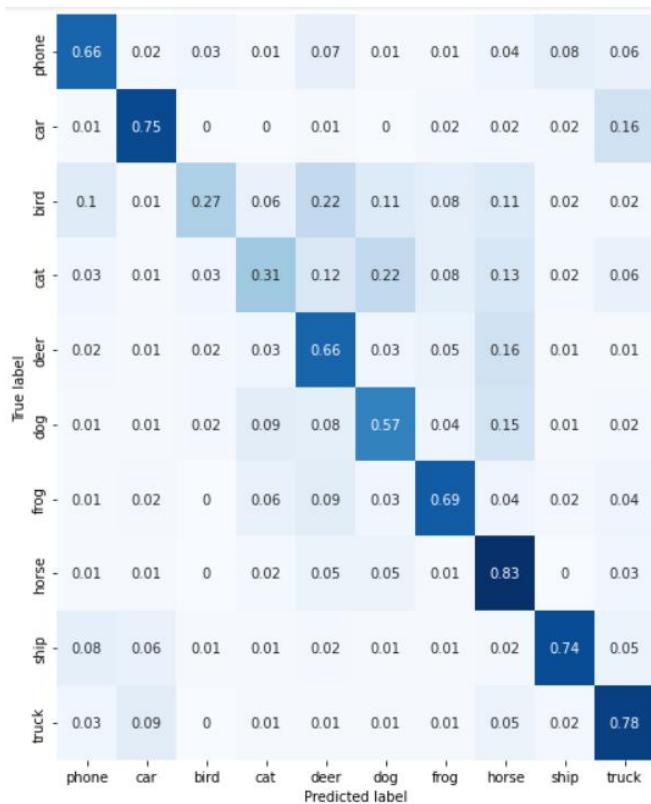
Runtime = 1 minute



شکل ۲۸: نمودار خطا به ازای هر ایپاک
برای حالت چهارم



شکل ۲۹: نمودار دقت به ازای هر ایپاک
برای حالت چهارم



شکل ۳۰: ماتریس آشفتگی حالت چهارم

test loss = 1.083900
test accuracy = 0.626000
Accuracy: 0.626000
Precision: 0.637580
Recall: 0.626000
F1 score: 0.614215

شکل ۳۱: پارامترهای داده های
 تست حالت چهارم

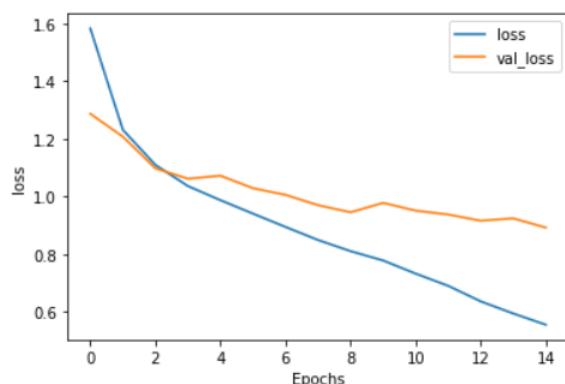
در میان این چهار حالت، بهترین حالت زمانی است که مقدار batch size = 128 است. در حالت های دیگر مقدار loss برای داده تست از ۱ بزرگتر است. این به معنی overfit شدن مدل است. هرچقدر مقدار کوچکتر باشد، تعداد batch ها بیشتر شده و به همین دلیل زمان اجرا شدن بیشتر می شود. از طرفی دیگر با کوچکتر شدن مقدار batch size ، مقدار خطای داده های آموزش بسیار کم و مقدار دقت آن ها بسیار زیاد می شود؛ اما از طرفی دیگر مقدار خطای داده های ارزیابی بیشتر شده و مقدار دقت آن ها کم می شود که این موضوع نشان دهنده overfit شدن است که مناسب نیست.

(۵)

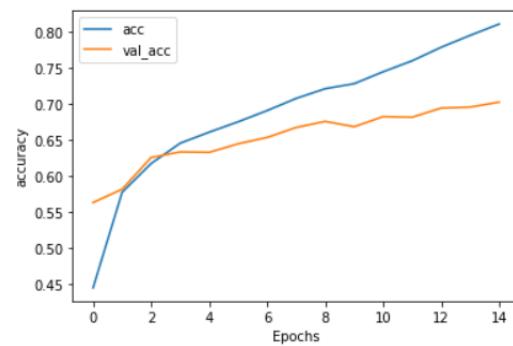
تابع فعالساز در بخش قبل، تابع relu بود و نتایج آن موجود است.

حالت دوم:

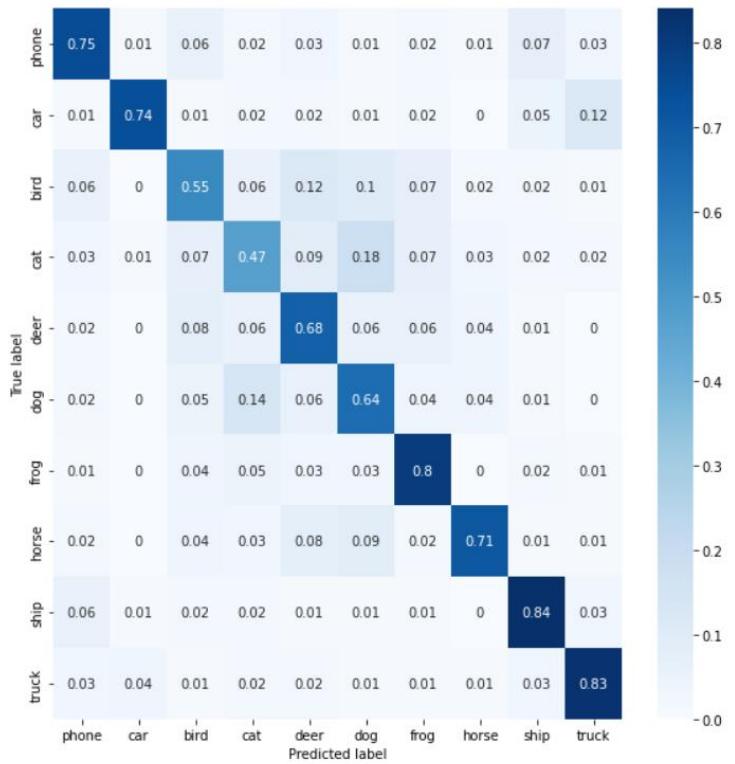
Activation : tanh



شکل ۳۲: نمودار خطای ازای هر ایپاک برای حالت دوم



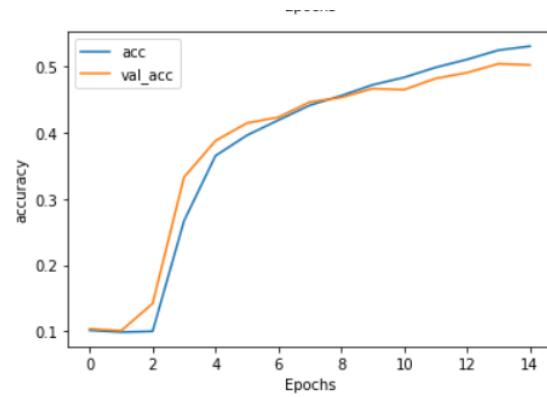
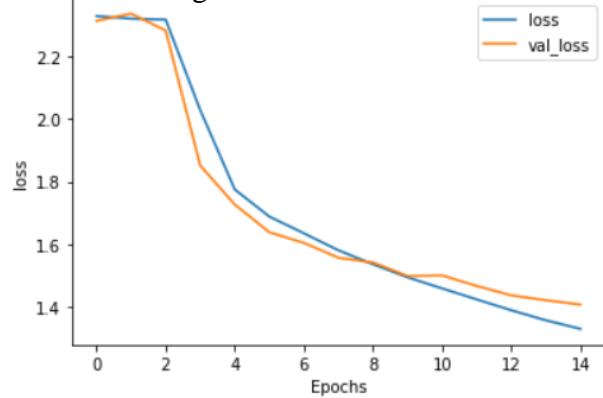
شکل ۳۳: نمودار دقت به ازای هر ایپاک برای حالت دوم



test loss = 0.926213
 test accuracy = 0.699200
 Accuracy: 0.699200
 Precision: 0.703055
 Recall: 0.699200
 F1 score: 0.698815

حالت سوم:

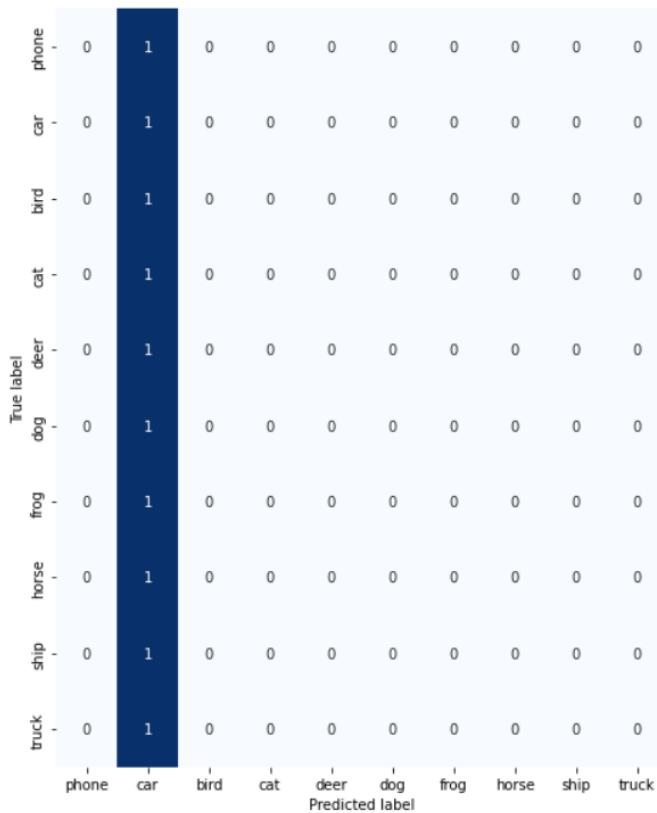
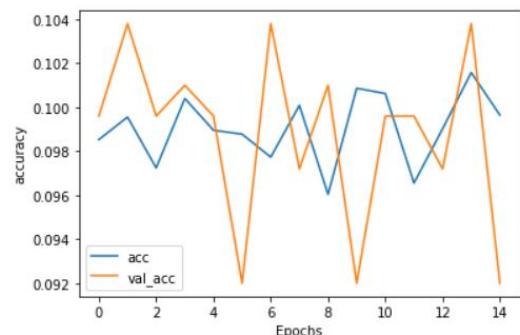
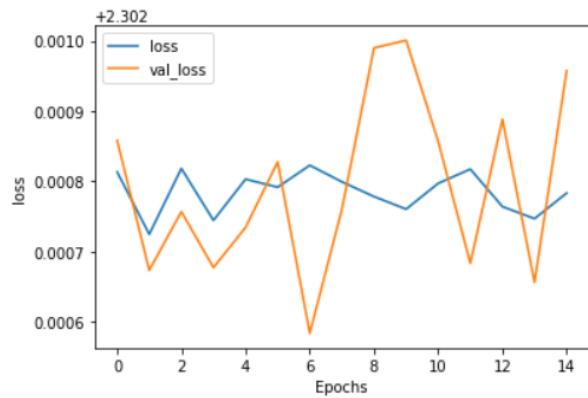
Activation : sigmoid



test loss = 1.430566
test accuracy = 0.496200
Accuracy: 0.496200
Precision: 0.498949
Recall: 0.496200
F1 score: 0.494122

حالت چهارم:

Activation : softmax



test loss = 2.302652
test accuracy = 0.100000
Accuracy: 0.100000
Precision: 0.010000
Recall: 0.100000
F1 score: 0.018182

با توجه به نتایج توابع فعالساز، می بینیم که مقدار خطای تابع فعالساز relu کمتر از سایر توابع فعالساز است. بدترین حالت برای تابع softmax است چراکه این تابع برای جدا کردن چندین کلاس های مختلف است و به همین دلیل است که در ماتریس آشفتگی این تابع می بینیم که نتایج پیش بینی شده فقط یک کلاس را نسبت داده است.

تابع فعالساز relu مانند یک پله عمل م یکند. یعنی داده های منفی را صفر کرده و داده های مثبت را عیناً تکرار می کند. فایده این تابع در اینجا این است که وقتی لایه های کاللوشن استفاده شده اند، هر چقدر مقدار هر پیکسل از تصویر عدد مثبت بزرگتری باشد، همبستگی بیشتری وجود دارد و مقادیر منفی باید حذف شوند.

تابع فعالساز sigmoid به صورت زیر است:

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

این تابع مقادیر بسیار کوچک یعنی حدوداً مقادیر کوچکتر از -5 را به مقداری نزدیک 0 تبدیل می کند و مقادیر بزرگ، یعنی بزرگتر از 5 را به 1 تبدیل می کند. این تابع شاید برای این مساله خیلی مناسب نباشد، زیرا ما نیاز داریم که پیکسل هایی که مقادیر مثبت بزرگتری دارند، به عددهایی بزرگتر تبدیل شوند تا قدرت و تاثیر آن ها در نظر گرفته شوند.

تابع فعالساز softmax یک سطر ورودی را به مقدار احتمالاتی آن تبدیل می کند. به طوری که در سطر خروجی جمع تمام مقادیر، 1 خواهد بود. این تابع در لایه آخر مدل و برای تشخیص هر کلاس استفاده می شود و به همین دلیل برای استفاده در تمام لایه ها مناسب نیست.

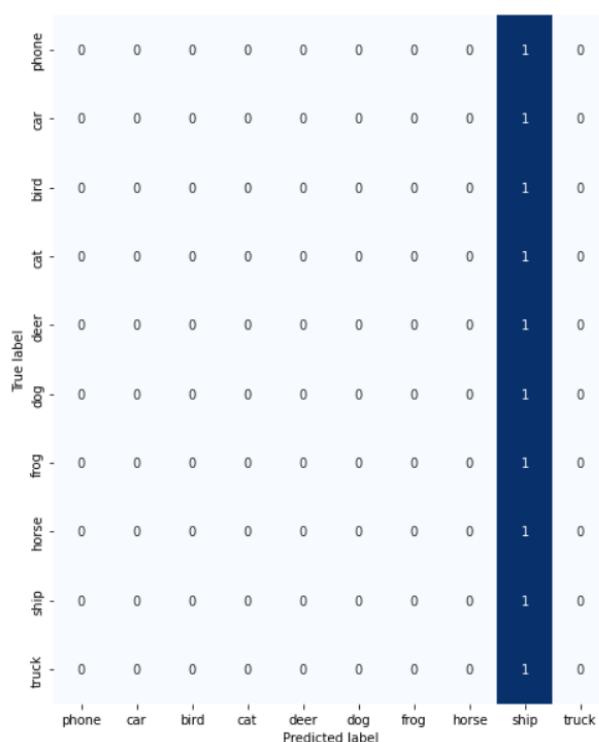
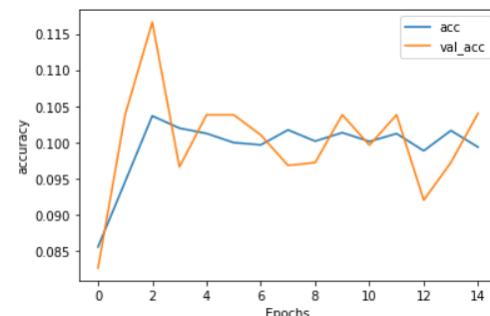
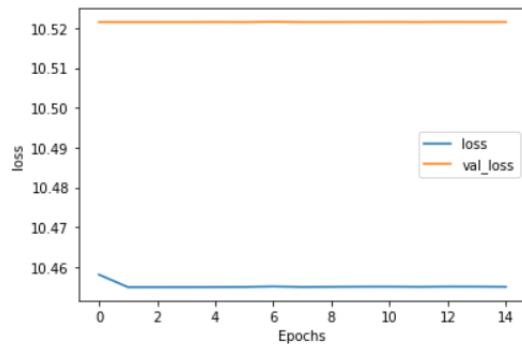
تابع فعالساز tanh نیز مقدار tanh هر داده را محاسبه می کند. خروجی این تابع مقادیر بین -1 و 1 هستند. در این مساله ما تمایل داریم مقادیر منفی حذف شوند و از آنجایی که عملکرد این تابع تقریباً به sigmoid شبیه است، برای لایه های پنهانی این مدل مناسب نیست.

(و)

برای حالت اول، تابع خطا sparse_categorical_crossentropy در نظر گرفته شده که در بخش قبل نتایج آن موجود است.

حالت دوم:

Loss: poisson

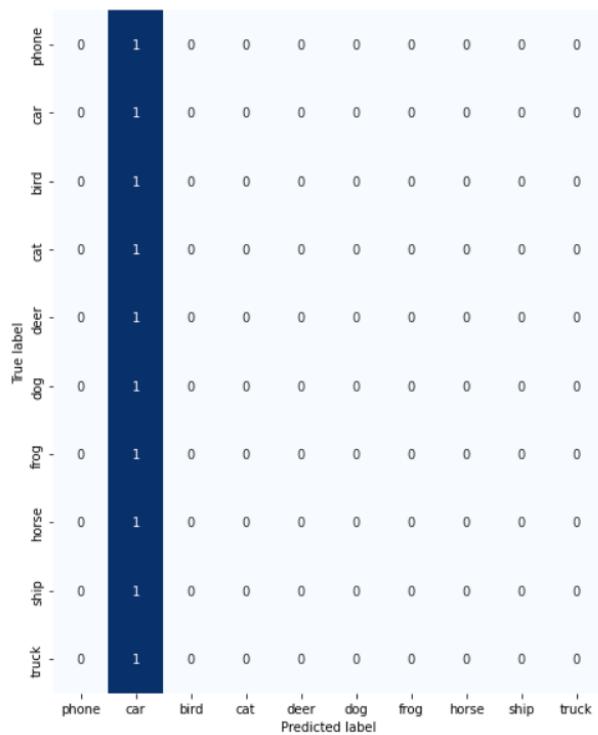
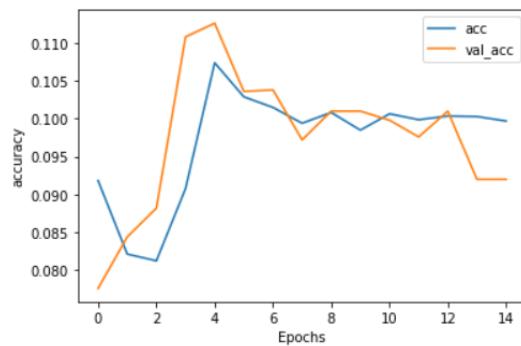
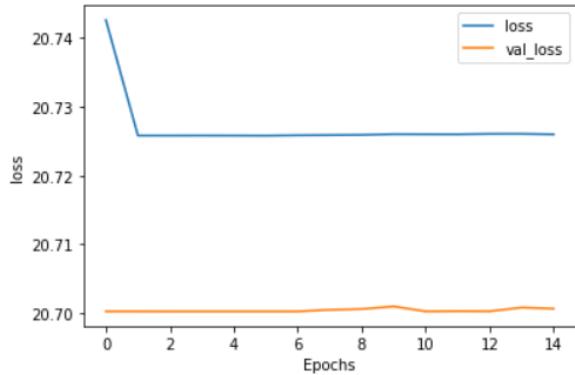


test loss = 10.461648
test accuracy = 0.100000
Accuracy: 0.100000
Precision: 0.010000
Recall: 0.100000
F1 score: 0.018182

A color scale bar for the confusion matrix, ranging from -0.0 (light blue) to 1.0 (dark blue).

حالت سوم:

Loss: kullback_leibler_divergence



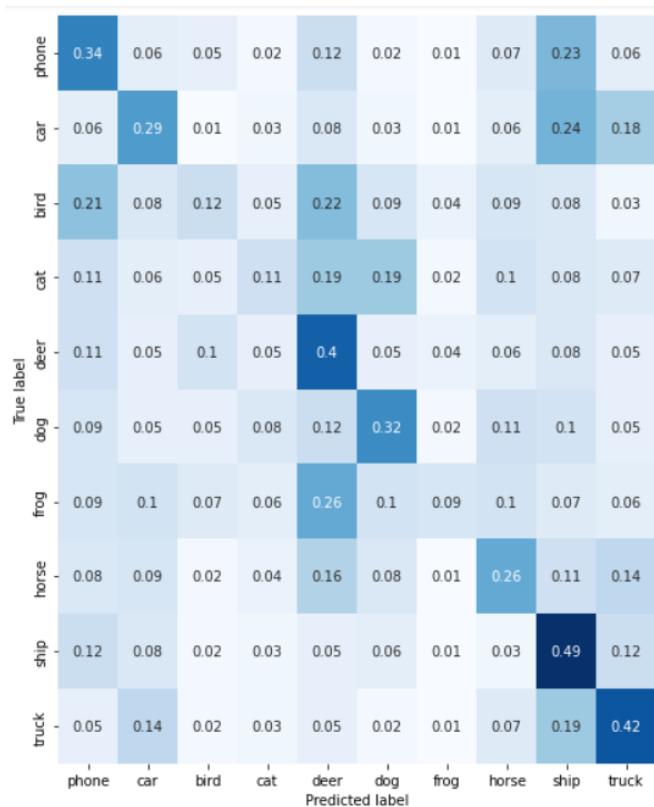
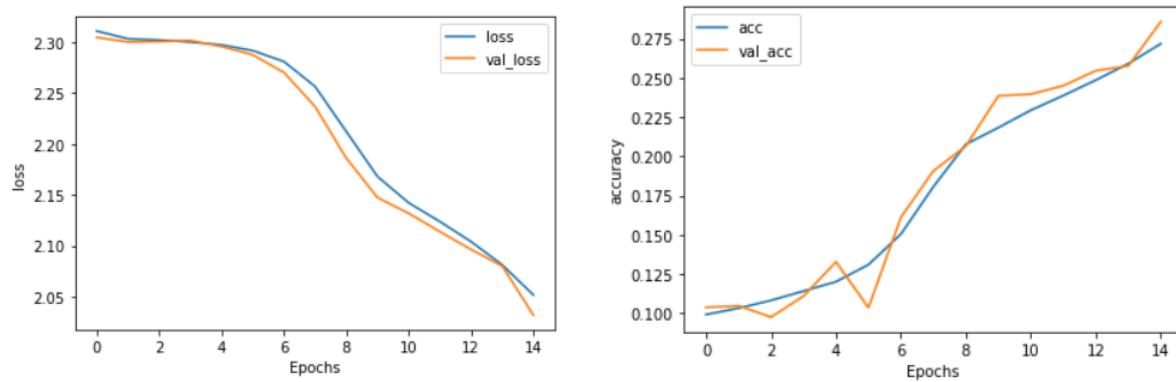
test loss = 20.723639
test accuracy = 0.100000
Accuracy: 0.100000
Precision: 0.010000
Recall: 0.100000

(ج)

برای حالت اول، تابع بهینه ساز adam در نظر گرفته شد که نتایج آن در بخش های قبل موجود است.

حالت دوم:

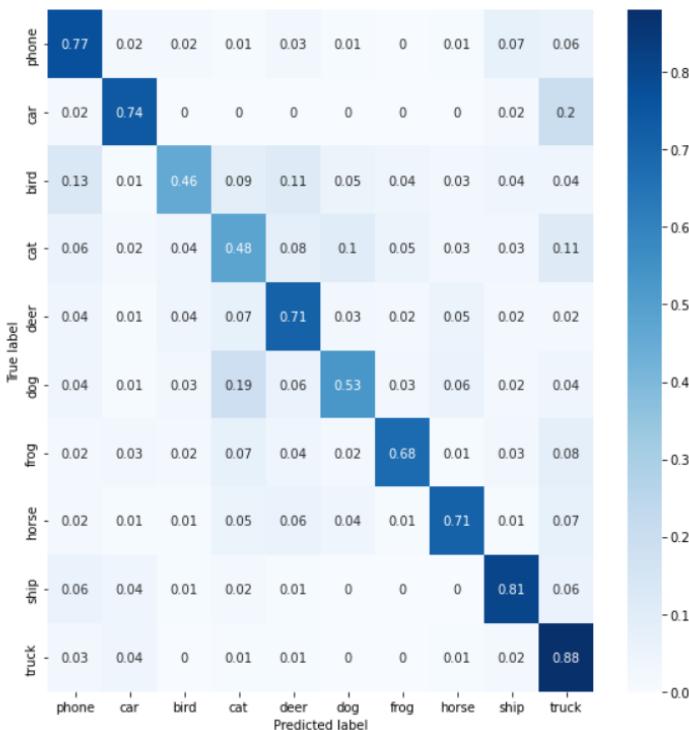
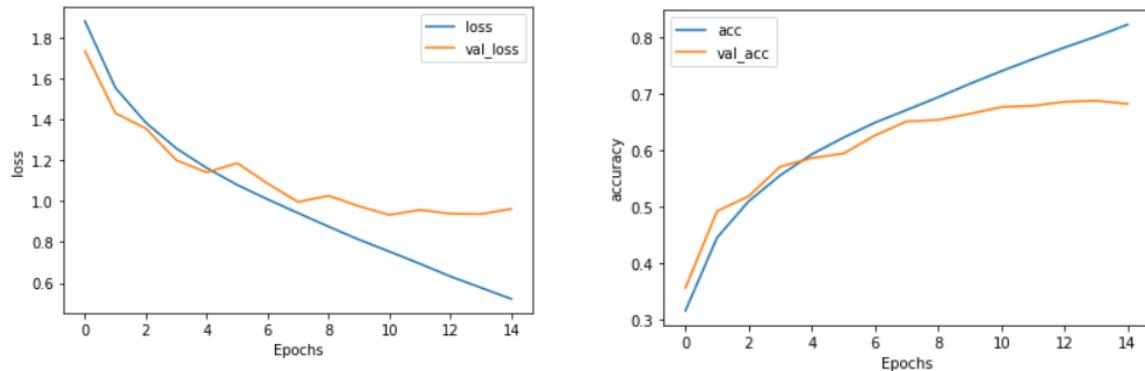
Optimizer: sgd



test loss = 2.026959
test accuracy = 0.285200
Accuracy: 0.285200
Precision: 0.285764
Recall: 0.285200

حالت سوم:

Optimizer: RMSprop



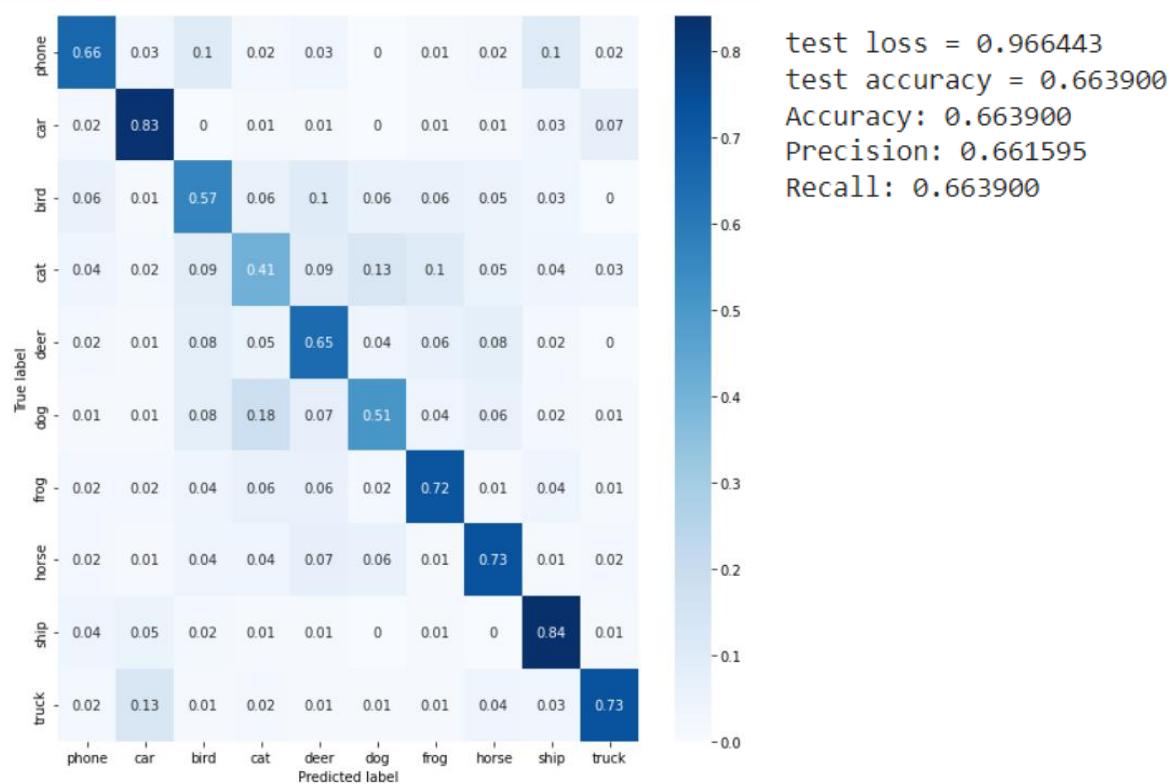
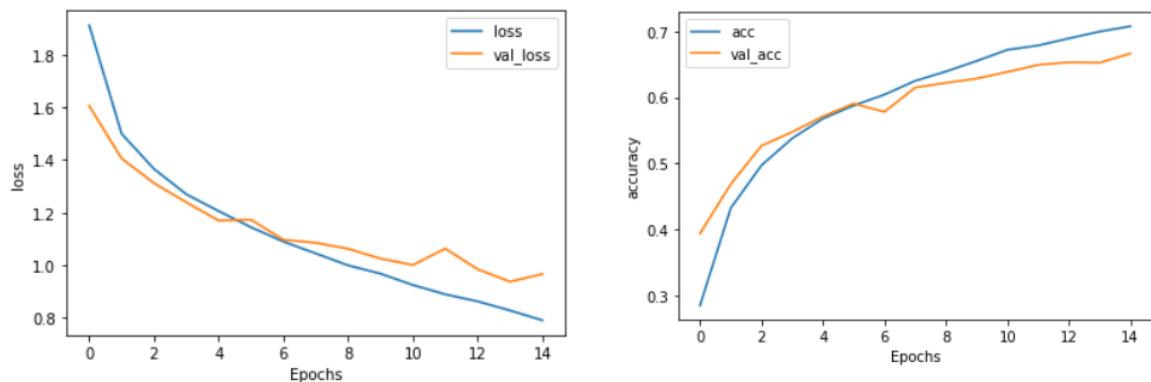
test loss = 1.008714
test accuracy = 0.676800
Accuracy: 0.676800
Precision: 0.689241
Recall: 0.676800

با توجه به مقادیر خطأ و دقت، بهترینتابع بهینه ساز همانتابع adam است.

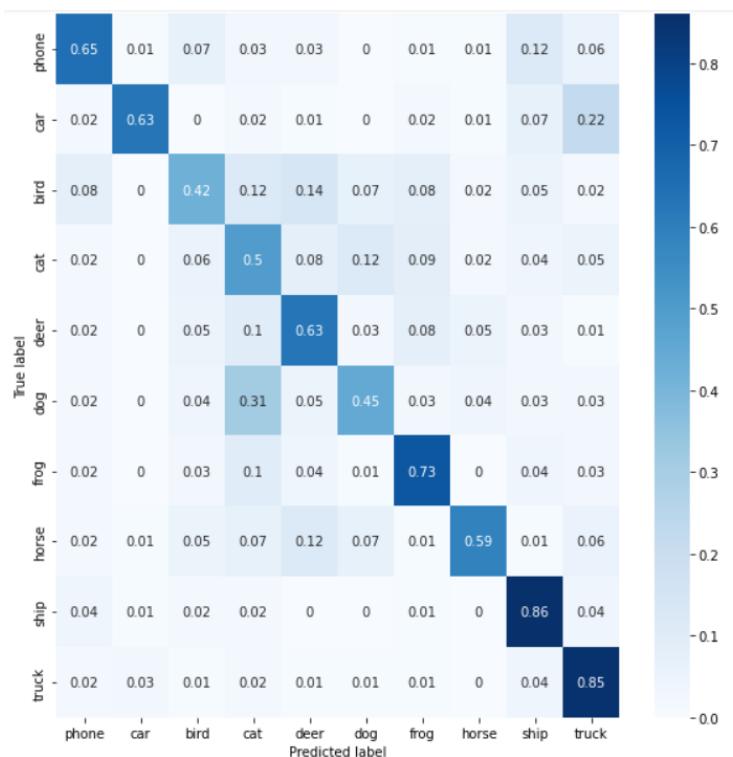
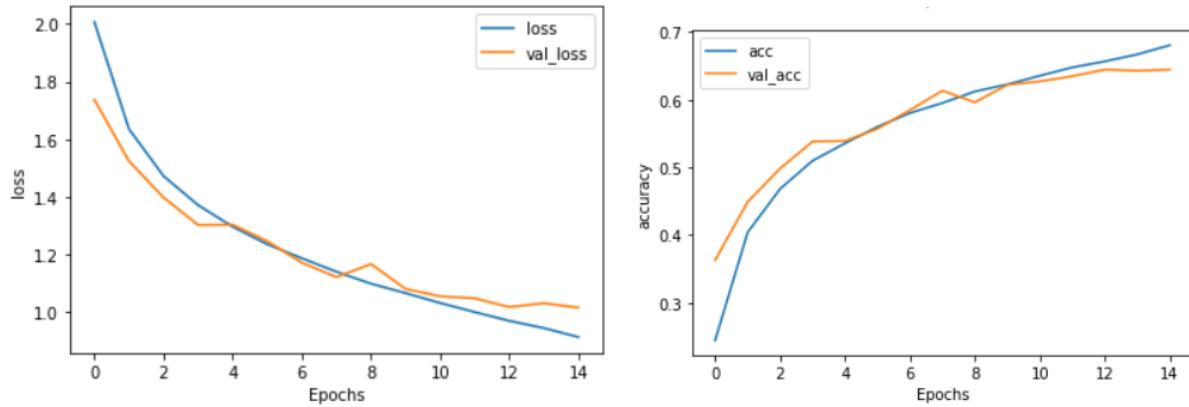
(ح)

مدل اصلی بخش های قبل دارای سه لایه Conv2D و Maxpooling و یک لایه dense و یک لایه خروجی بود. در این بخش برای بررسی اثر تعداد لایه ها، لایه های conv2D و maxpooling را برای تعداد ۴ و ۵ و ۲ امتحان می کنیم.

حالت اول: ۴ لایه

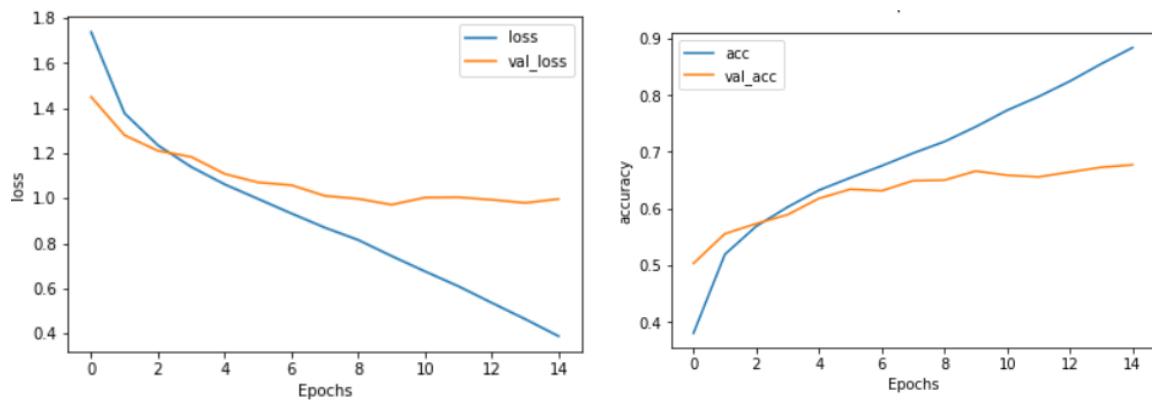


حالت دوم: لایه



test loss = 1.066795
test accuracy = 0.630700
Accuracy: 0.630700
Precision: 0.647344
Recall: 0.630700
F1 score: 0.629235

حالت سوم: ۲ لایه



test loss = 0.987931
 test accuracy = 0.678900
 Accuracy: 0.678900
 Precision: 0.683068
 Recall: 0.678900
 F1 score: 0.680301

با توجه به این سه حالت، می بینیم که اگر تعداد لایه های کانلوشن ۲ تا باشد، بهترین نتیجه را از نظر خطأ و دقت می دهد.

(ط)

در ابتدا تاثیر تعداد نرن ها مورد بررسی قرار گرفت. از انجایی که پارامتر filter در لایه های conv2D مانند نورون هستند، این مقادیر تغییر کردند. بر اساس نتایج دیدیم که هرچقدر مقدار فیلتر بزرگتر باشد و تعداد نورون های لایه dense قبل از خروجی نیز بزرگتر باشد، مدل بهتر و با خطای کمتر و دقت بالاتری عمل می کند.

در بخش بعد تاثیر تغییر پارامتر batch size را بررسی کردیم. در اینجا دیدیم که اگر این پارامتر مقدار خیلی کوچکی داشته باشد، باعث می شود که خطای داده های ارزیابی بزرگ شود و این بزرگتر شدن نشان دهنده overfit شدن مدل است. با توجه به نتایج، بهترین مقدار برای این پارامتر عدد ۱۲۸ است.

در مرحله بعد چند تابع فعالساز را امتحان کردم و معایب هر کدام مشاهده شد. با استفاده از نتایج، به این نتیجه رسیدم که برای این مدل، تابع relu بهترین تاب برای لایه های پنهانی و تابع doftmax بهترین تابع برای لایه خروجی است.

با بررسی انواع تابع های خطأ که میتوان برای مساله دسته بندی استفاده کرد و به خصوص با توجه به ماتریس آشفتگی، به این نتیجه رسیدم که تابع sparse_categorical_crossentropy بهترین تابع برای خطأ است.

از میان تابع های بهینه ساز adam, sgd, RMSprop نیز و با توجه به مقدار خطأ و دقت حاصل از آنها، بهترین تابع بهینه ساز تابع adam است. در دو تابع دیگر مقدار خطای داده تست از ۱ بزرگتر شد که مناسب نیست.

در نهایت نیز تاثیر تعداد لایه های نورونی بررسی شد. در اینجا من تعداد لایه های کانولوشنی را تغییر دادم. در نهایت مشاهده شد که اگر این لایه ها ۲ تا باشند هم زمان پردازش کاهش پیدا می کند و هم مقدار خطأ و دقت به عدد مناسبی می رسد.

(c)

Number of neurons	Batch size	Activation function	optimizer	Loss function	Number of layers
64 neurons in con2D & 512 in dense	128	relu	adam	sparse_categorical_crossentropy	2 con2D & 1 Dense

loss: 0.3998 - accuracy: 0.8768 - val_loss: 0.9678 - val_accuracy: 0.6866

test loss = 0.999932
test accuracy = 0.679000

runtime = 0:00:50.242793

سوال ۲ – MLP(Regression)

می دانیم که در شبکه های عصبی و برای طرایحی مدل، فقط می توان از داده های عددی استفاده کرد. در این داده، چند ستون مانند شهر، خیابان، تاریخ و .. وجود دارند که داده های غیرعددی هستند. جنس این داده ها object است. برای پردازش راحت تر داده ها، انها را به تایپ category تبدیل می کنیم. در مرحله بعد برای تبدیل کردن انها به عدد، از دستور cat.codes استفاده کردم. این دستور به هر کدام از داده ها یک عدد ترتیبی نسبت می دهد. برای مثال در ستون خیابان، ۴۴ خیابان منحصر به فرد وجود داشتند. با این دستور این داده ها به اعداد ۰ تا ۴۳ تبدیل می شوند.

```
def objToint(x):
    x = x.astype('category')
    x = x.cat.codes
    return x
```

تابع تبدیل کردن object ب category و تبدیل به عدد

```
cat_data = data.select_dtypes(include = ['object'])
cat_columns = cat_data.columns
for c in cat_columns:
    data[c] = objToint(data[c])
```

جادو کردن داده های غیر عددی و اعمال تابع بالا روی آن ها

حال که همه ستون های داده ها، به عدد تبدیل شدند، عمل نرمالایز کردن انجام می شود. نرمالایز کردن با استفاده از تابع MinMaxScaler صورت می گیرد. این تابع مقدار داده ها را به نحوی نرمالایز می کند که همه مقادیر در رنج ۰ و ۱ قرار بگیرند.

```
scaler = MinMaxScaler()
names = data.columns
d = scaler.fit_transform(data)
data = pd.DataFrame(d, columns=names)
```

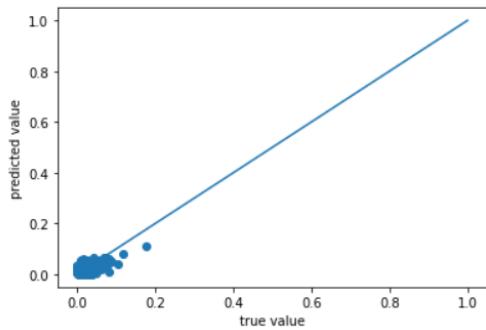
نرمالایز کردن همه ستون ها

(ب)

برای جدا کردن تصادفی داده های اموزشی و تست از تابع `data.sample` استفاده شد.

مدل اول: یک لایه پنهانی و تابع فعالساز `relu`

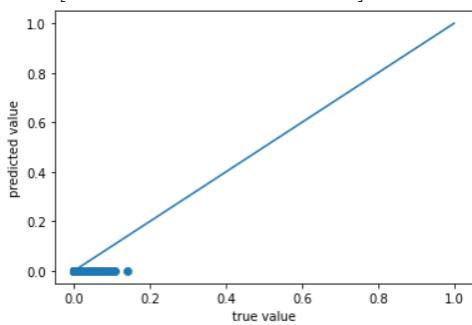
29/29 [=====] - 0s 1ms/step - loss: 1.1518e-04 - accuracy: 0.0098



نمودار قیمت پیش بینی شده بر حسب قیمت واقعی در مدل اول

مدل دوم: دو لایه پنهانی و تابع فعالساز `relu`

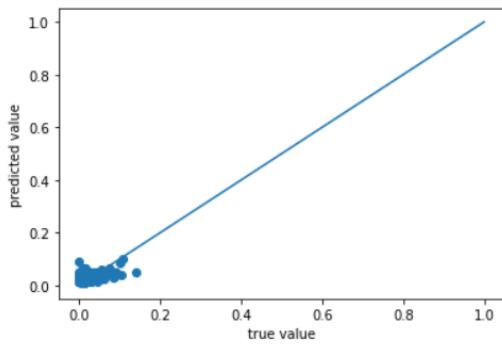
29/29 [=====] - 0s 1ms/step - loss: 5.6630e-04 - accuracy: 0.0163



نمودار قیمت پیش بینی شده بر حسب قیمت واقعی در مدل دوم

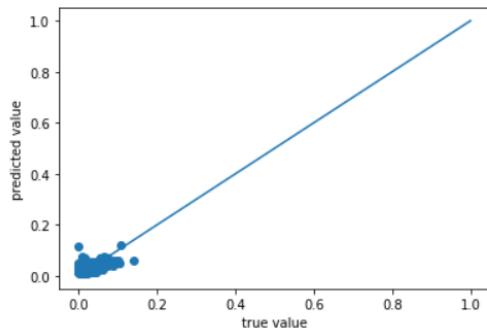
مدل سوم: یک لایه پنهانی و تابع فعالساز sigmoid

29/29 [=====] - 0s 1ms/step - loss: 1.0287e-04 - accuracy: 0.0163



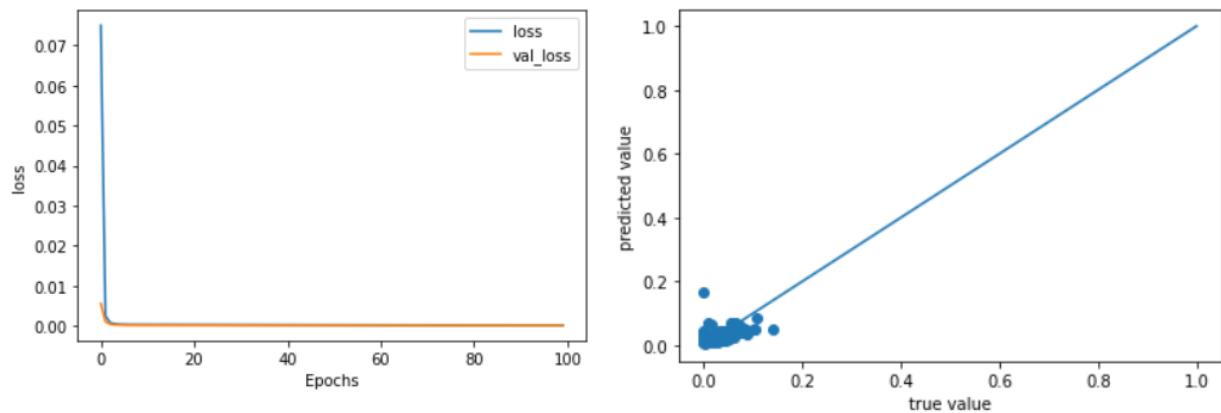
مدل چهارم: دو لایه پنهانی و تابع فعالساز sigmoid

29/29 [=====] - 0s 1ms/step - loss: 1.0460e-04 - accuracy: 0.0163



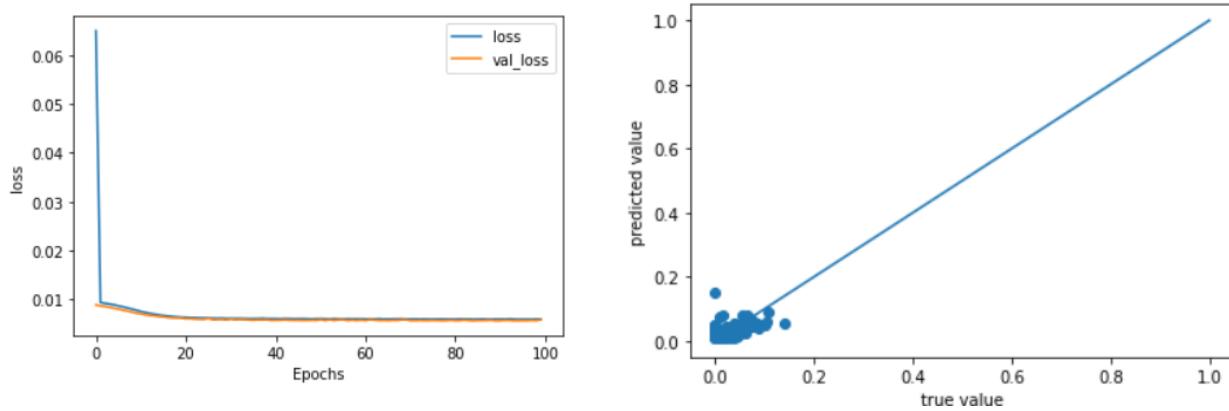
بهترین مدل با توجه به مقادیر loss, accuracy مقدار accuracy و کمترین مقدار loss را دارد.

(۲)



تعداد ایپاک بهینه با توجه به نمودار خط، برابر است با حدود ۲۰ تا ۴۰.

(۳)



در این بخش نیز با توجه به نمودار خط، ۲۰ تا ۴۰ عددی مناسب برای تعداد ایپاک است.

(۵)

رابطه ریاضی MSE:

$$MSE = \frac{1}{n} \sum_{i=1}^{i=n} (original_i - Predicted_i)^2$$

رابطه ریاضی MAE:

$$MAE = \frac{1}{n} \sum_{i=1}^{i=n} |Predicted_i - Original_i|$$

هر دو معیار MSE، MAE همیشه نامنفی هستند. تفاوت در این است که در MSE توان دوم به خطاهای نسبت داده می شود و مقادیر بزرگتری می دهد. این کار ممکن است احتمال رخ دادن overfit را افزایش دهد. فایده این روش این است که با استفاده از این روش مطمئن می شویم که ارورهای بزرگی نخواهیم داشت و این ارورهای بزرگ با توان دوم اسکیل می شوند.

از طرفی دیگر در روش MAE همه خطاهای در یک روش خطی اسکیل می شوند و احتمال overfit کمتر می شود اما نمی تواند مانند MSE خطاهای بزرگ را اسکیل کند و خطاهای بزرگتر بیشتر تاثیر می گذارند و باعث خطا در مدل می شوند.

سوال ۳ - کاهش ابعاد

(الف)

هدف از روش PCA این است که برخی ویژگی هایی که برای هدف مدل مناسب نیستند را حذف کند. ویژگی مفید، ویژگی است که در شرایط مختلف و برای تارگت های مختلف، متفاوت باشد. برای مثال، داده سوال دوم را در نظر می گیریم. اگر در این داده همه کد پستی ها یکسان بود، میتوان به این نتیجه رسید که داده کد پستی تاثیری روی قیمت و در نتیجه فایده ای برای پیش بینی ندارد. هدف PCA این است که این ویژگی های غیرمفید را حذف کند و با ارائه یک ترکیب خطی از سایر ویژگی ها، ویژگی جدید و مفیدی تولید کند که به هدف و پیش بینی کمک کند. این روش بهترین ترکیب خطی را بین ویژگی های موجود پیدا می کند و از آن یک ویژگی جدیدی می سازد که بیشترین واریانس را داشته باشد.

روش ریاضی پیاده سازی PCA:

در این روش برای پیدا کردن بهترین ترکیب خطی و ساختن ویژگی که بیشترین واریانس را دارد، از ماتریس ویژه و مقادیر ویژه استفاده می شود.

مرحله اول: فرض کنید که داده هایی با m نمونه و n ویژگی داریم. در مرحله اول باید داده را نرمالایز کرد. برای این کار میانگین را از داده ها کم کرده و بر واریانس تقسیم می کنیم.

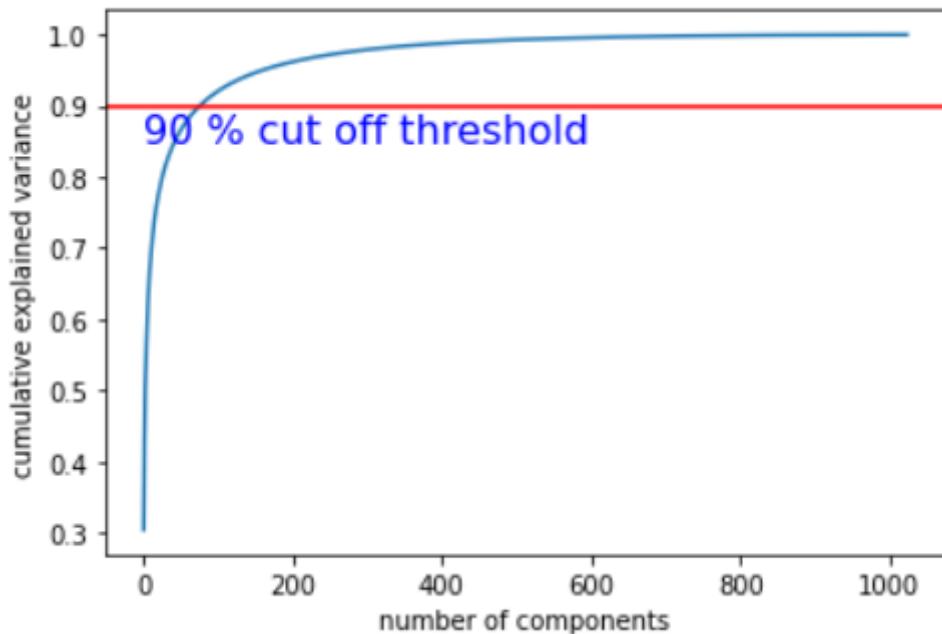
مرحله دوم: ماتریس کواریانس داده های جدید را محاسبه می کنیم.

مرحله سوم: ماتریس ویژه و مقادیر ویژه ماتریس کواریانس را محاسبه می کنیم.

مرحله چهارم: ماتریس های ویژه ای که مقادیر ویژه بزرگتری داشته باشند، همان principal components مورد نظر ما هستند. در این مرحله باید ماتریس های ویژه را بر حسب مقادیر ویژه متناظر با آنها و از بزرگ به کوچک مرتب کنیم.

مرحله پنجم: ماتریس جدید با k سطر می سازیم. در هر سطر ماتریس ویژه ای قرار دارد و k تعداد component هایی است که بیشترین مقادیر ویژه را داشتند.

مرحله ششم: این ماتریس جدید با ابعاد $k * n$ را در ماتریس داده های اصلی با ابعاد $m * n$ ضرب می کنیم. ماتریس جدیدی با ابعاد $m * k$ حاصل می شود که نشان دهنده ویژگی های جدید و مفید است. به این شیوه کاهش ابعاد صورت می گیرد زیرا k از n کوچکتر است.



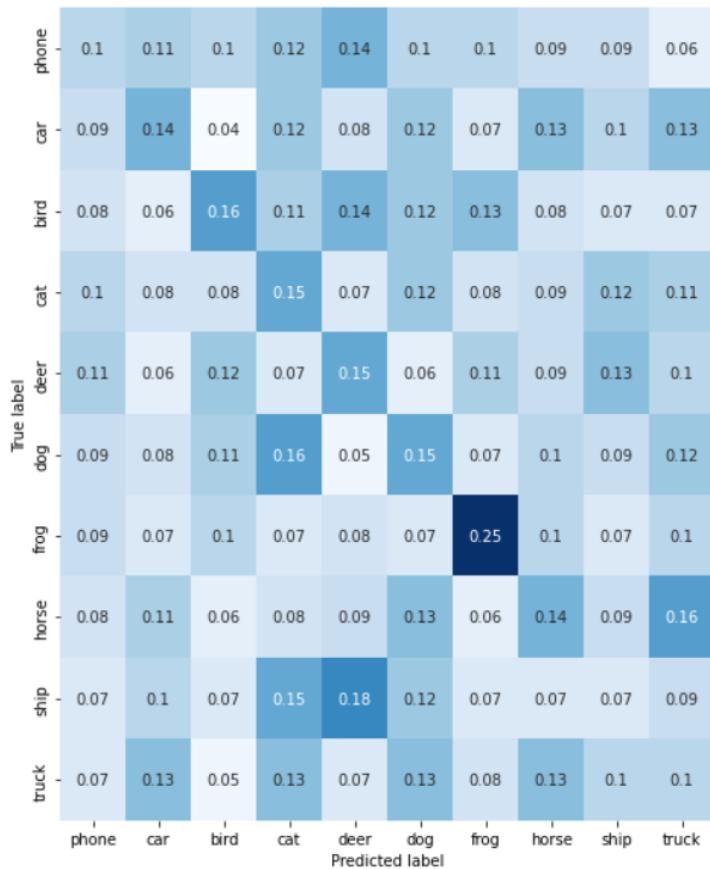
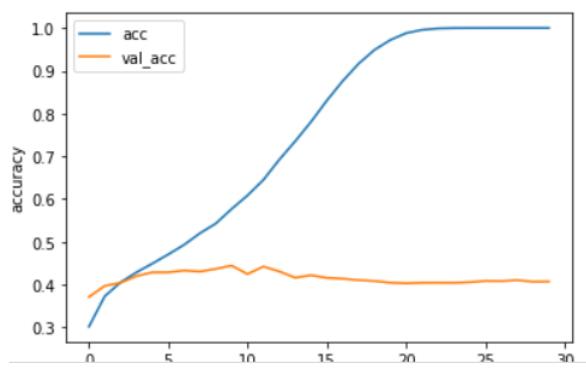
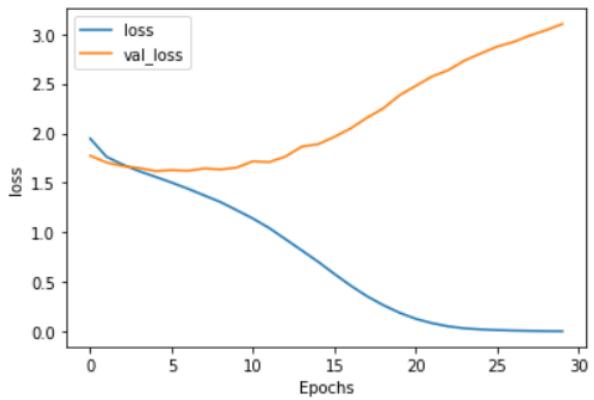
نمودار تعداد component ها بر حسب واریانس تجمعی

با توجه به این نمودار کاهش بعد تا حدود $20 * 20$ می تواند مناسب باشد.
برای پیاده سازی روش pca تابعی مطابق با الگوریتم توضیح داده شده نوشته شده است.

```
def pca(data, pc_count = None):

    data -= mean(data, 0)
    data /= std(data, 0)
    C = cov(data)
    E, V = eigh(C)
    key = argsort(E)[::-1][:pc_count]
    E, V = E[key], V[:, key]
    U = dot(data, V)
    return U
```

.pca تابع



test loss = 6.352817
 test accuracy = 0.140400
 Accuracy: 0.140400
 Precision: 0.140156
 Recall: 0.140400
 F1 score: 0.140011

loss: 0.0059 - accuracy: 1.0000 - val_loss: 3.1009 - val_accuracy: 0.4074

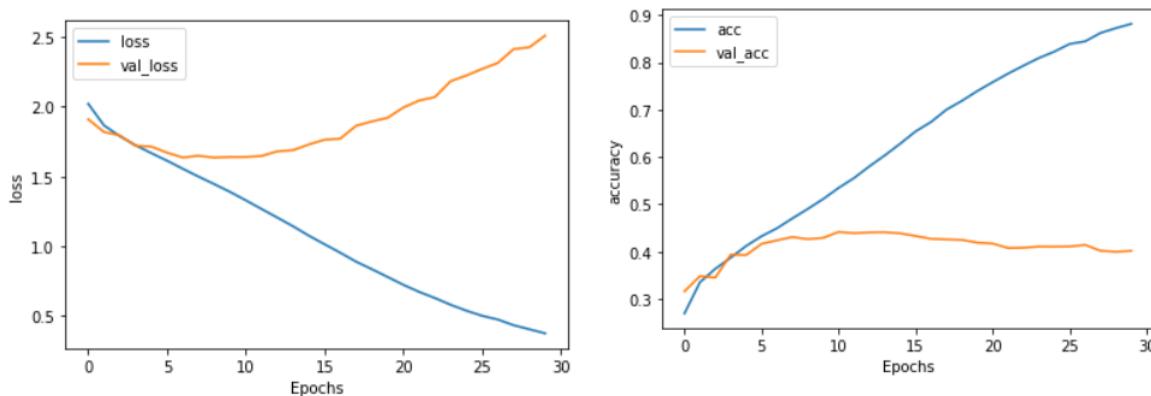
(ب)

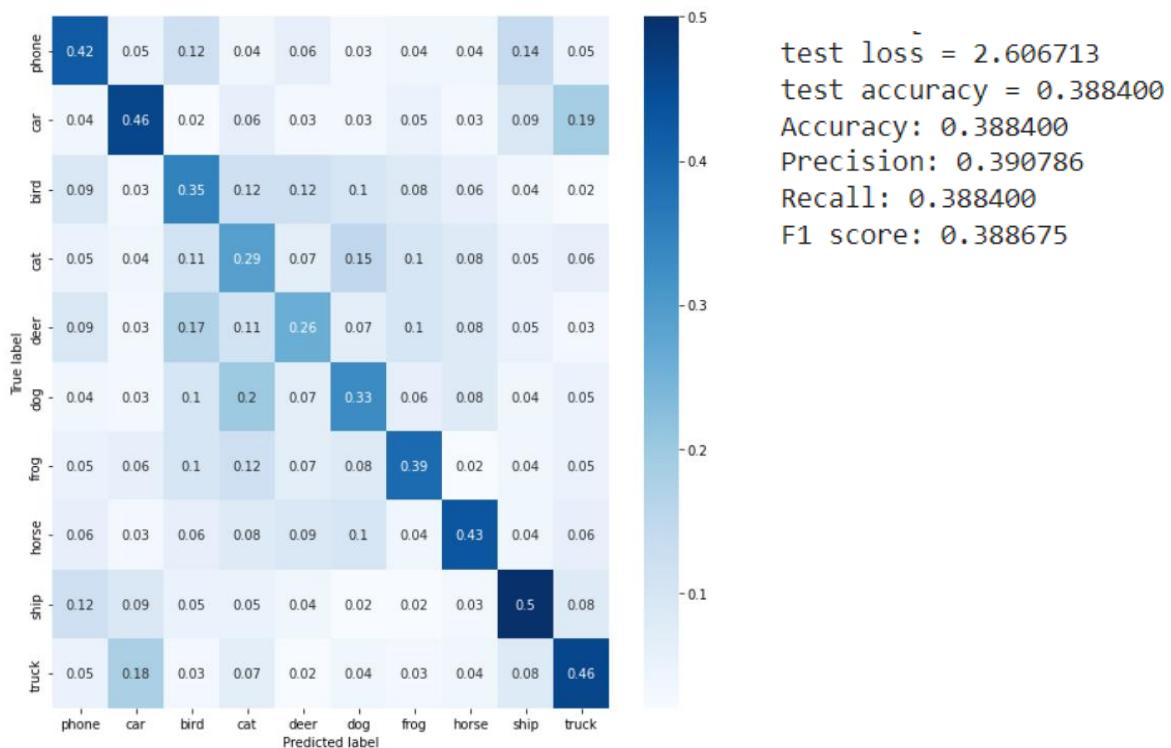
برای پیاده سازی روش autoencoder ابتدا یک بخش encoder و یک بخش decoder با استفاده از لایه های dense در کتابخانه keras طراحی کردم. در بخش encoder ابعاد تصاویر که ۱۰۲۴ هستند به ۲۵۶ می رسند و در بخش decoder ابعاد مجدداً به ۱۰۲۴ بر می گردند. هدف از درنظر گرفتن بخش decoder این بود که به صورت کلی عملکرد autoencoder را بررسی کنم. اینکه تا چه حد می تواند یک تصویر را پس از کاهش ابعاد و مجدداً برگرداندن به ابعاد اصلی، شبیه به تصویر اصلی نتیجه دهد.

```
encoding_dim = 256
input_img = keras.Input(shape=(1024,))
encoded = layers.Dense(800, activation='relu')(input_img)
encoded = layers.Dense(512, activation='relu')(input_img)
encoded = layers.Dense(encoding_dim, activation='relu')(input_img)

decoded = layers.Dense(512, activation='relu')(encoded)
decoded = layers.Dense(800, activation='relu')(encoded)
decoded = layers.Dense(1024, activation='sigmoid')(encoded)
```

حالا برای پیاده سازی کاهش ابعاد روی تصاویر، از بخش encoder این مدل یک مدل جدأگانه به نام encoder می سازیم و آنرا روی تصاویر اصلی پیاده سازی میکنم. نتایج این پیاده سازی باعث تولید داده های آموزش و تست و ارزیابی جدیدی می شوند. در ادامه با استفاده از مدل به دست آمده از سوال اول، این داده ها را آموزش می دهم.





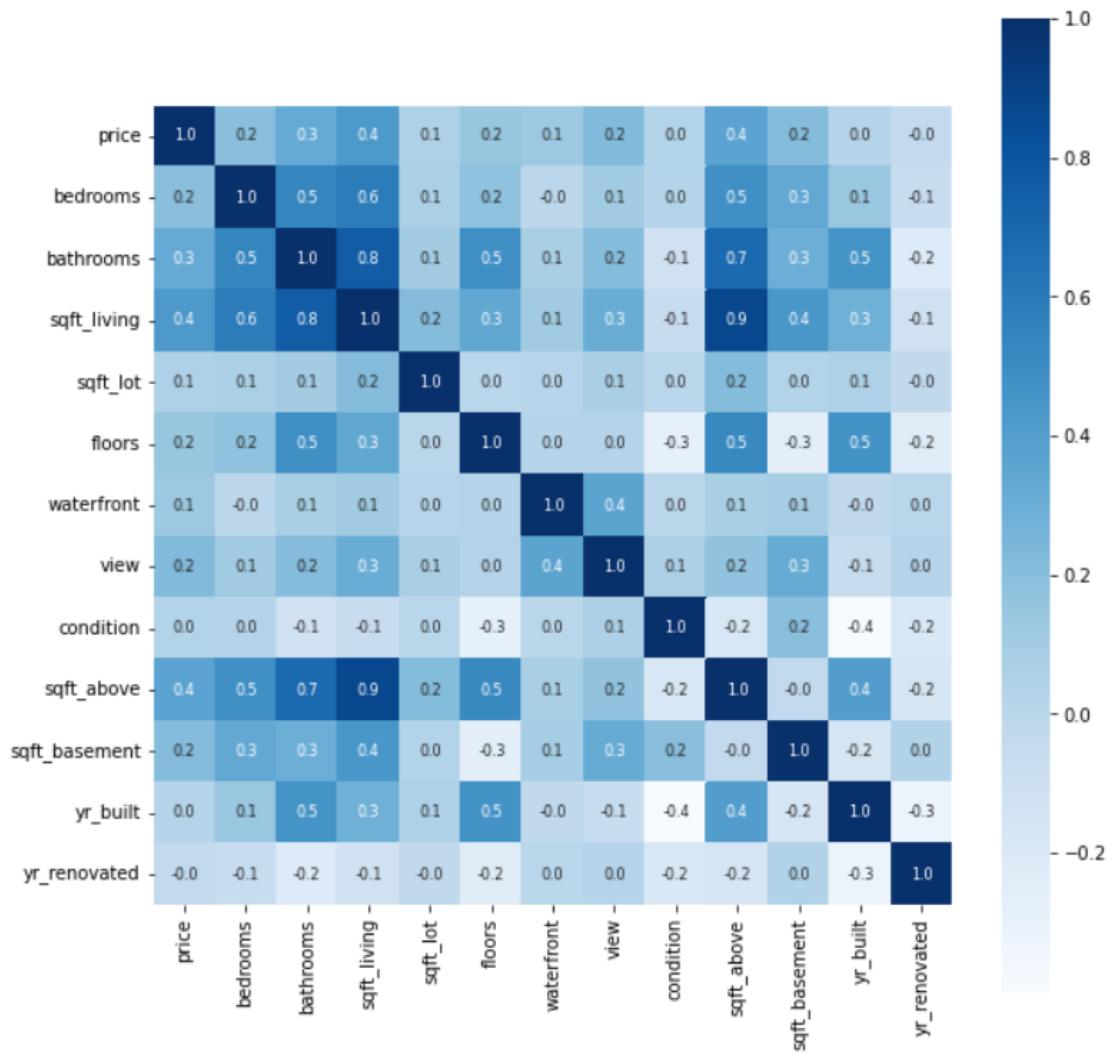
loss: 0.4047 - accuracy: 0.8726 - val_loss: 2.4269 - val_accuracy: 0.3998

در این مدل می بینیم که مقدار خطا برای داده های آموزشی کم شده اما در عین حال خطا برای داده های ارزیابی با هر ایپاک بیشتر می شود که این موضوع نشان از overfit دارد. با توجه به این موضوع و همچنین با توجه به مقدار خطای داده های تست میتوان نتیجه گرفت که مدل به دست آمده از سوال یک بهتر عمل می کند. علت این اتفاق هم به دلیل کاهش بعد و از دست رفتن بخشی از اطلاعات است . با مقایسه هر سه مدل نیز درمی یابیم که مدل حاصل از سوال ۱ بهترین حالت را دارد.

	Test loss	Test accuracy	Overfit
Ideal model of Question1	0.99	0.67	no
PCA	6.35	0.14	yes
Autoencoder	2.6	0.38	yes

جدول ۲: مقایسه سه مدل برای داده های cifar10

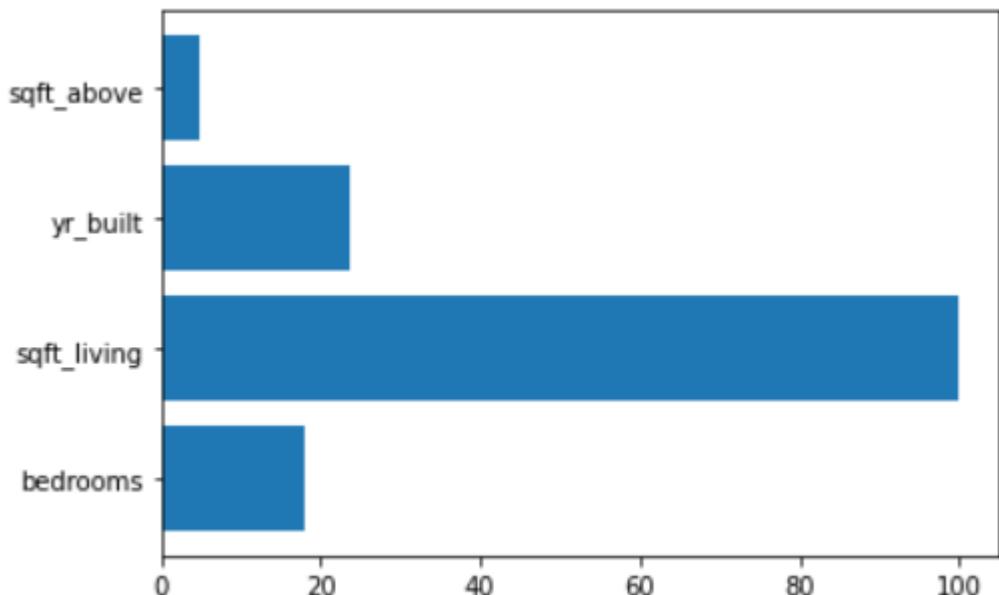
(۵)



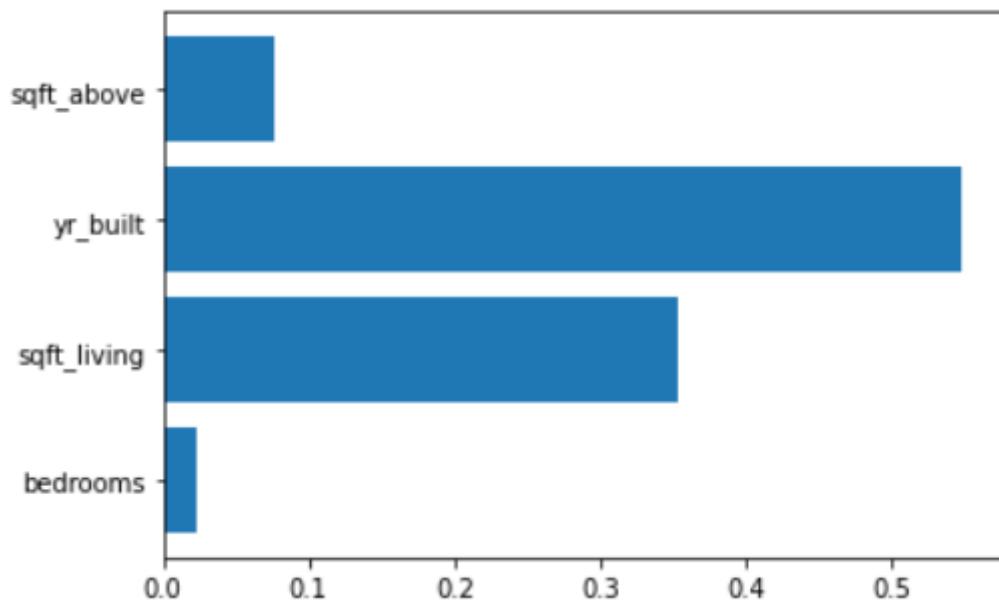
برای کشیدن این ماتریس ابتدا مقادیر همبستگی داده ها را با استفاده از دستور `corr` به دست می آوریم. این مقادیر نشان می دهد که داده های هر کدام از ستون ها با دیگر ستون ها تا چه حد همبستگی دارند. همانطور که در ماتریس مشاهده می شود، ستون های یکسان عدد ۱ را نشان می دهند که به این دلیل است که هر ستون با خودش همبستگی کامل و مستقیمی دارد. عدههای مثبت به این معنی هستند که داده ها با یکدیگر همبستگی مستقیم دارند یعنی اگر یک داده افزایش پیدا کند دیگری نیز افزایش می یابد اما مقادیر منفی مفهوم معکوس دارند. همانطور که در سطر `price` مشاهده می شود، داده `sqrt_living`, `sqrt_above` با داشتن همبستگی ۰,۴ بیشترین همبستگی را با قیمت خانه دارند.

همانطور که مشاهده می شود در این ماتریس، داده های غیر عددی در نظر گرفته نشده اند. همبستگی برای داده های غیر عددی نمی تواند تعریف شود چراکه مفهوم همبستگی وابسته به مقدار است و تغییر دو پارامتر را نسبت به همدیگر می سنجد.تابع `corr` که مقدار همبستگی را محاسبه می کند، خود به خود این داده ها را حذف می کند.

(و)



بارپلات اهمیت هر ویژگی با مدل LinearRegression



بارپلات اهمیت هر ویژگی با مدل DecisionTreeRegression

