تمرین ۶

سینا اسکندری ۹۷۵۲۱۰۵۴

۱.

ابتدا با استفاده از numpy آرایه را به صورت زیر تعریف می کنیم.

```
arr = np.array([
    [20, 17, 32, 42, 65],
    [13, 65, 96, 53, 21],
    [45, 63, 74, 38, 64],
    [23, 76, 40, 34, 26],
    [14, 66, 78, 49, 23]
])
```

برای batch normalization در هر ستون میانگین و انحراف معیار محاسبه می شود که کد آن و نتیجه به صورت زیر است.

```
batch_norm = (arr - np.mean(arr, axis=0)) / np.std(arr, axis=0)
batch_norm
```

```
array([[-0.25839036, -1.95226042, -1.32416942, -0.17220347,  1.24551178],
       [-0.8613012 ,  0.36725691,  1.32416942,  1.40632837, -0.92919133],
       [ 1.89486265,  0.27061036,  0.41380294, -0.74621505,  1.19608671],
       [ 0.        ,  0.89881297, -0.99312707, -1.32022663, -0.68206597],
       [-0.77517108,  0.41558019,  0.57932412,  0.83231679, -0.83034119]])
```

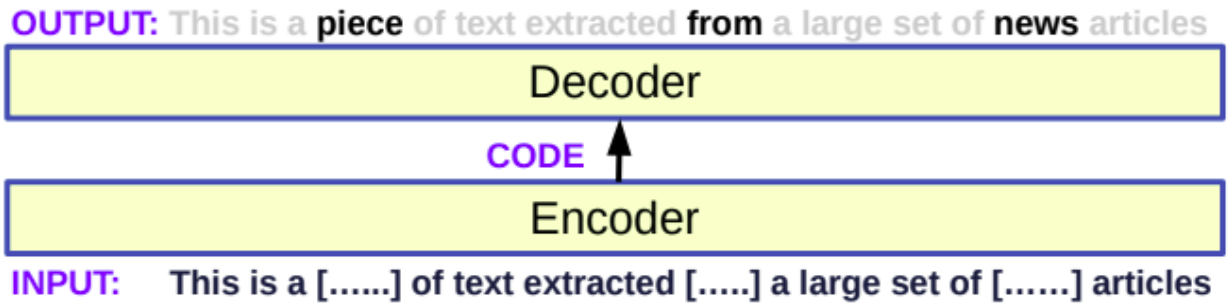
برای layer normalization نیز مانند حالت بالا فقط برای هر ردیف محاسبه می شود.

```
layer_norm = (arr - np.mean(arr, axis=1)) / np.std(arr, axis=1)
layer_norm
```

```
array([[-0.87558998, -1.07958858, -1.87064621,  0.11541284,  0.77748801],
       [-1.27882221,  0.5099897 ,  2.95682788,  0.69247703, -1.02301054],
       [ 0.56452512,  0.44375727,  1.29738366, -0.09442869,  0.73656759],
       [-0.70277617,  0.87426805, -1.26721195, -0.30427021, -0.81840843],
       [-1.22121761,  0.54310591,  1.59910079,  0.48263551, -0.9411697 ]])
```

۲-

BERT یک مدل زبانی است که بازنمایی کلمات را بر اساس متنی که در آن هستند یاد می گیرد. مدل هایی نظیر word2vec یا GloVe، embedding ها را بدون توجه به context یاد می گیرند که این یک محدودیت است چون کلمات ممکن در محل های مختلف معنای مختلف داشته باشند. در مدل BERT در هنگام آموزش تعدادی از کلمات ورودی قبل از ورود به encoder از جمله حذف(mask) می شوند و مدل با استفاده از یک لایه softmax کلمه ای که بیشترین احتمال قرار گرفتن در جای خالی را دارد پیشبینی می کند. به این تسک masked auto-encoder گفته می شود.



مهم ترین ویژگی در مدل BERT این است که می توان با توجه به تسک دلخواه خود fine-tune کرد.

۳.

الف)

ساختار شبکه:

```
model = keras.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=x_train[0].shape))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPool2D())
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPool2D())
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

در این حالت دقت ۱۶ درصد روی داده تست بدست می آید.

```
Epoch 18/20
4/4 [==============================] - 1s 221ms/step - loss: 0.0327 - accuracy: 1.0000 - val_loss: 2.2613 - val_accuracy: 0.1704
Epoch 19/20
4/4 [==============================] - 1s 171ms/step - loss: 0.0624 - accuracy: 0.9950 - val_loss: 2.2587 - val_accuracy: 0.1642
Epoch 20/20
4/4 [==============================] - 1s 219ms/step - loss: 0.0548 - accuracy: 0.9950 - val_loss: 2.2542 - val_accuracy: 0.1629
```

ب)

برای تشخیص زاویه هر تصویر بدون برچسب را به صورت رندوم ۹۰، ۱۸۰ و یا ۲۷۰ درجه چرخیده می شوند

```
x_rotated = np.zeros_like(x_unlabeld)
y_rotated = np.zeros((x_unlabeld.shape[0],))
for i in range(x_rotated.shape[0]):
    k = random.randint(0, 3)
    x_rotated[i] = np.rot90(x_unlabeld[i], k)
    y_rotated[i] = k

y_rotated = keras.utils.to_categorical(y_rotated, num_classes=4)
```

ساختار مدل مثل قسمت قبل هست فقط لایه آخر ۴ نورون دارد که در این حالت دقت داده آموزشی ۸۷ درصد شده است.

```
779/779 [==============================] - 5s 6ms/step - loss: 0.3724 - accuracy: 0.8591
Epoch 19/20
779/779 [==============================] - 5s 6ms/step - loss: 0.3546 - accuracy: 0.8658
Epoch 20/20
779/779 [==============================] - 5s 6ms/step - loss: 0.3301 - accuracy: 0.8751
```

مدل را به شکل زیر با استفاده از functional API تغییر می دهیم.

```
model_2 = keras.Model(inputs=model.inputs, outputs=layers.Dense(10, activation='softmax')(model.layers[-2].output))

model_2.summary()
```

که در این حالت دقت validation برابر ۱۸ درصد شده است.

```
Epoch 17/20
4/4 [==============================] - 1s 219ms/step - loss: 1.6333 - accuracy: 0.4800 - val_loss: 2.3897 - val_accuracy: 0.1694
Epoch 18/20
4/4 [==============================] - 1s 171ms/step - loss: 1.5871 - accuracy: 0.4800 - val_loss: 2.3760 - val_accuracy: 0.1738
Epoch 19/20
4/4 [==============================] - 1s 168ms/step - loss: 1.5356 - accuracy: 0.5350 - val_loss: 2.3630 - val_accuracy: 0.1790
Epoch 20/20
4/4 [==============================] - 1s 177ms/step - loss: 1.5102 - accuracy: 0.5350 - val_loss: 2.3494 - val_accuracy: 0.1842
```

پ)

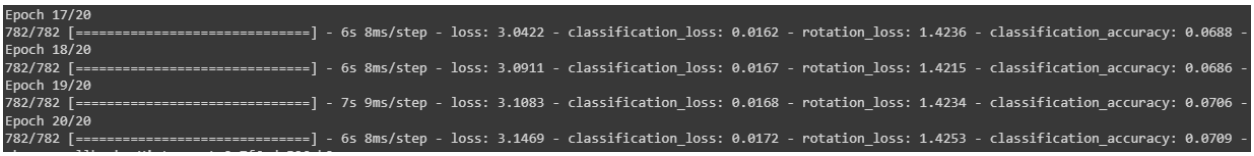
ساختار مدل:

```
input = layers.Input(shape=x_train_concat[0].shape)
x = layers.Conv2D(32, (3, 3), activation='relu')(input)
x = layers.Conv2D(32, (3, 3), activation='relu')(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPool2D()(x)
x = layers.Conv2D(64, (3, 3), activation='relu')(x)
x = layers.Conv2D(64, (3, 3), activation='relu')(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPool2D()(x)
out = layers.Flatten()(x)
base_model = keras.Model(inputs=input, outputs=out)

classification_layer = layers.Dense(10, activation='softmax', name='classification')(base_model.outputs[0])
rotation_layer = layers.Dense(4, activation='softmax', name='rotation')(base_model.outputs[0])

model = keras.Model(inputs=base_model.inputs, outputs=[classification_layer, rotation_layer])

model.summary()
```

مقدار loss تسک طبقه بندی حدود ۰/۰۱ و تسک چرخش حدود ۱ می باشد بنابراین برای اینکه تاثیر این ۲ برابر شود وزن loss طبقه بندی را برابر ۱۰۰ قرار می دهیم.

```
model.compile(
    loss={
        'classification': 'categorical_crossentropy',
        'rotation': 'categorical_crossentropy'
    },
    optimizer=keras.optimizers.Adam(learning_rate=1e-4, decay=1e-4 / 20),
    metrics=['accuracy'],
    loss_weights={
        'classification': 100,
        'rotation': 1
    }
)

model.fit(
    x_train_concat, [y_train_classification, y_train_rotation],
    batch_size=64,
    epochs=20,
    validation_data=(x_test, {'classification': y_test})
)
```

نتایج مدل بعد از ۲۰ ایپاک به این صورت می باشد.



باقی متریک ها در نوتبوک مشخص است.

۴.

الف)

با استفاده از keras tuner مدل را به صورت زیر تعریف می شود.

```
def build_model(hp):
    dropout_rate = hp.Choice('dropout', [0.3, 0.4, 0.5])
    model = keras.Sequential()
    model.add(layers.Input(shape=(32, 32, 3)))
    for i in range(hp.Int('num_cnn_block', min_value=1, max_value=3)):
        model.add(layers.Conv2D(2 ** (i + 5), (3,3), padding='same', activation='relu'))
        model.add(layers.BatchNormalization())
        model.add(layers.Conv2D(2 ** (i + 5), (3,3), padding='same', activation='relu'))
        model.add(layers.BatchNormalization())
        model.add(layers.MaxPooling2D(pool_size=(2,2)))
        model.add(layers.Dropout(dropout_rate))

    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Dense(hp.Int('num_dense_units', min_value=128, max_value=512, step=128), activation='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.Dropout(dropout_rate))
    model.add(layers.Dense(num_classes, activation='softmax'))

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=hp.Float("lr", min_value=1e-4, max_value=1e-2, sampling="log")),
        loss="categorical_crossentropy",
        metrics=["accuracy"],
    )

    return model
```

برای dropout مقادیر [۰/۳, ۰/۴ , ۰/۵]، برای تعداد بلوک cnn مقادیر بین ۱ تا ۳، برای تعداد نورون لایه dense ماقبل آخر مقادیر بین ۱۲۸ تا ۵۱۲ با فاصله های ۱۲۸ و برای learning rate نیز مقادیر بین ۰/۰۰۰۱ و ۰/۰۱ در نظر گرفته می شود.

ب)

بهترین مدل بدست آمده دارای دقت validation ۷۸ درصد می باشد.

```
tuner.search(img_train, label_train, epochs=10, batch_size=64, validation_data=(img_test, label_test))
```

```
Trial 3 Complete [00h 04m 48s]
val_accuracy: 0.7843500077724457

Best val_accuracy So Far: 0.7843500077724457
Total elapsed time: 00h 10m 17s
```

که در این حالت dropout برابر ۰/۵، تعداد بلوک cnn برابر ۳، تعداد نورون لایه dense برابر ۱۲۸ و lr برابر ۰/۰۰۱۸ است.

```
Results summary
Results in my_dir/dl_hw6_q4
Showing 10 best trials
<keras_tuner.engine.objective.Objective object at 0x7fb5773869d0>
Trial summary
Hyperparameters:
dropout: 0.5
num_cnn_block: 3
num_dense_units: 128
lr: 0.0018286522831819768
Score: 0.7843500077724457
Trial summary
Hyperparameters:
dropout: 0.4
num_cnn_block: 1
num_dense_units: 512
lr: 0.007411401916639344
Score: 0.5741499960422516
Trial summary
Hyperparameters:
dropout: 0.3
num_cnn_block: 1
num_dense_units: 512
lr: 0.00010353713140058703
Score: 0.5659500062465668
```

که به نظر می رسد درصد dropout بیشتر باعث شده دقت validation بیشتر شود و همچنین تعداد بلوک cnn بیشتر به معنای پیدا کردن فیچر های بیشتر و پیچیده تر است که به یادگیری بهتر مدل کمک کرده است.

پ)

بهترین مدل به دقت validation ۸۵ درصد رسید که از حالت اول ۷ درصد بیشتر است.

```
782/782 [==============================] - 8s 10ms/step - loss: 0.4729 - accuracy: 0.8377 - val_loss: 0.4588 - val_accuracy: 0.8479
Epoch 26/30
782/782 [==============================] - 8s 10ms/step - loss: 0.4696 - accuracy: 0.8372 - val_loss: 0.4558 - val_accuracy: 0.8512
Epoch 27/30
782/782 [==============================] - 8s 10ms/step - loss: 0.4594 - accuracy: 0.8408 - val_loss: 0.4907 - val_accuracy: 0.8390
Epoch 28/30
782/782 [==============================] - 8s 10ms/step - loss: 0.4608 - accuracy: 0.8407 - val_loss: 0.4534 - val_accuracy: 0.8526
Epoch 29/30
782/782 [==============================] - 8s 10ms/step - loss: 0.4566 - accuracy: 0.8421 - val_loss: 0.4608 - val_accuracy: 0.8498
Epoch 30/30
782/782 [==============================] - 8s 10ms/step - loss: 0.4556 - accuracy: 0.8426 - val_loss: 0.4592 - val_accuracy: 0.8490
```

مقادیر precision، recall و f1 به صورت زیر می باشد.

```
from sklearn.metrics import f1_score, precision_score, recall_score, confusion_matrix

y_pred = best_model.predict(img_test)
y_pred = np.argmax(y_pred, axis=1)

print(f'precision: {precision_score(np.argmax(label_test, axis=1), y_pred , average="macro")}')
print(f'recall: {recall_score(np.argmax(label_test, axis=1), y_pred , average="macro")}')
print(f'f1: {f1_score(np.argmax(label_test, axis=1), y_pred , average="macro")}')
```

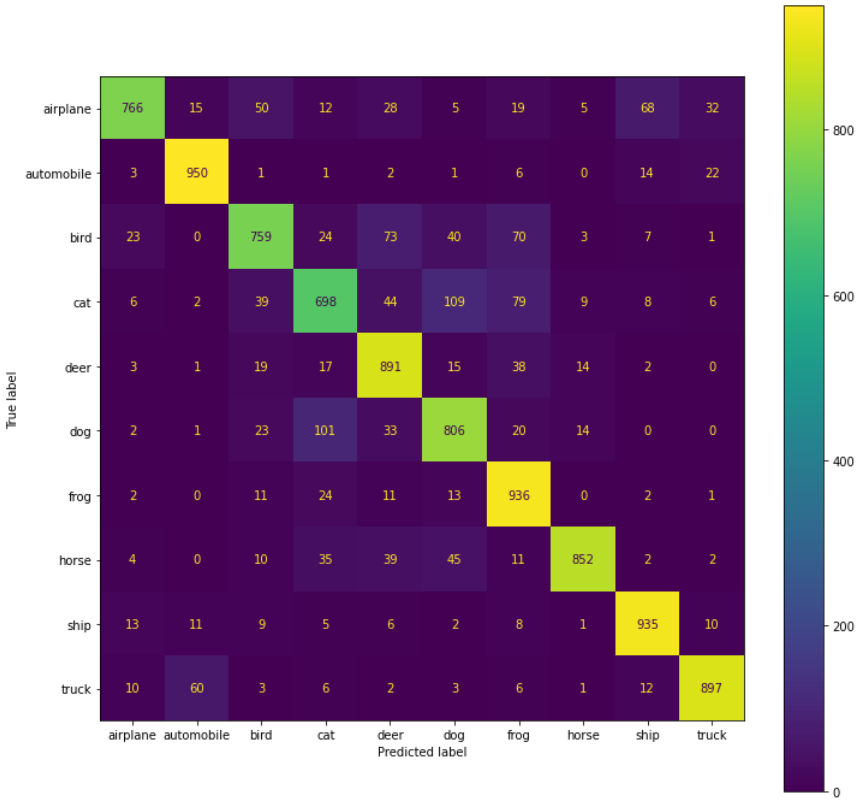
```
313/313 [==============================] - 1s 3ms/step
precision: 0.8523295242852074
recall: 0.849
f1: 0.8483476256917941
```

در اینجا استفاده از این متریک ها الزامی نیست و می توان با accuracy پیش رفت. همچنین این متریک ها در مسائلی که دیتا unbalanced است یا اینکه هزینه False Positive یا False Negative زیاد است(مثلا تشخیص اسپم بودن ایمیل یا مثلا تشخیص سالم بودن فرد بیمار) بیشتر استفاده می شود.

پ)

TP به معنای این است که مقدار پیشبینی شده مثبت و مقدار واقعی نیز مثبت باشد. TN به معنای این است که مقدار پیشبینی شده منفی و مقدار واقعی نیز منفی باشد. FP به معنای این است که مقدار پیشبینی شده مثبت ولی مقدار واقعی منفی باشد. FN به معنای این است که مقدار پیشبینی شده منفی ولی مقدار واقعی مثبت باشد.

نمودار confusion matrix به صورت زیر می باشد.



همانطور که مشخص است مدل در پیشبینی گربه خوب عمل نکرده و ۱۰۹ تصویر که گربه بودند را سگ پیشبینی کرده است یا ۱۰۱ تصویر که سگ بودند را گربه تشخیص داده است.