به نام خدا

تمرین سیزدهم یادگیری عمیق

غزل زمانی‌نژاد

۹۷۵۲۲۱۶۶

1. self supervised learning **زیرمجموعه ای از** unsupervised learning **به شمار می رود. در این دو دسته بندی، هیچ برچسبی برای داده ها توسط انسان مشخص نمی شود.**

Unsupervised **بیشتر شامل** clustering **و گروه بندی داده ها می شود. مثلا مدل در این نوع یادگیری سعی می کند داده ها را طبق ویژگی هایی که می آموزد و بر اساس شباهت میان آنها دسته بندی کند.**

در یادگیری self supervised**، ابتدا به صورت خودکار برای داده ها شبه برچسب تولید می شود و سپس مدل برآن اساس آموزش می بیند. این روش برای یادگیری ویژگی‌های عمومی از داده‌های بدون برچسب در مقیاس بزرگ پیشنهاد شده است. میتوانیم با استفاده از انتقال دانش، ویژگی های آموخته شده را به** downstream task **منتقل کنیم. به طور مثال پیش بینی موقعیت در ناحیه اشتراک یادگیری** self supervised **و یادگیری بازنمایی قرار میگیرد. در این مسئله بخشی از تصویر را به صورت تصادفی انتخاب میکنیم، سپس برای 8 ناحیه اطراف آن برچسب تولید میکنیم. سپس مدل را به گونه ای آموزش میدهیم که موقعیت تصاویر را نسبت به هم تشخیص دهد. در صورتی که مدل به خوبی آموزش ببیند میتواند منجر به شناخت خوبی از محتوای تصویر شود. از دیگر مثال های این دسته میتوان به حل جورچین، تخمین چرخش و پیش بینی محتوا اشاره کرد. در تمامی این مثال ها ابتدا برچسب هایی به صورت خودکار تولید میشوند و سپس مدل سعی میکند براساس برچسب ها ویژگی های خوبی از داده ها استخراج کند.**

در الگوریتم‌های یادگیری ماشین یکی از مسائل مهم بازنمایی داده است. هرچه بازنمایی داده بهتر باشد، یادگیری مدل بهتر خواهد بود. سعی داریم نمایش داده ها را به گونه دیگری تغییر دهیم. مثلا ابعاد تصویر را به گونه ای کاهش دهیم که یادگیری را ساده تر کند. یادگیری بازنمایی، می تواند به صورت با ناظر یا بدون ناظر انجام شود.

در یادگیری بازنمایی بدون ناظر، به عنوان مثال میتوان به استفاده از auto-encoder **اشاره کرد. در این نوع مسئله، مدل باید وزن ها را به گونه ای یاد بگیرد که برگشت پذیر باشد. یعنی از طریق وزن هر لایه بتوانیم به حدود ورودی های همان لایه دست پیدا کنیم. زیرا در آن صورت مطمئن هستیم وزن ها اطلاعات مفیدی**

آموخته اند. علاوه بر آن، در مدل های زبان طبیعی نیز از این نوع یادگیری استفاده می شود. مثلا در بردارهای word embedding مثل جفت های context/target و word2vec برای استخراج ویژگی کلمات از این بردارها استفاده می شود.

در یادگیری بازنمایی با ناظر، بازنمایی توسط انسان مشخص می شود. مثلا مشخص می کنیم هر تصویر را با تاپل 2تایی از فاصله بین موجودیت های تصویر نمایش دهیم. یا اینکه ابتدا مدل را بر روی دیتاست imagenet آموزش میدهیم.

2. الف) اگر $a(n)$ میزان دقت مدلی باشد که از ابتدا با n برچسب آموزش دیده و $a_{ft}(n)$ میزان دقت مدلی که fine-tune شده باشد، آنگاه بهره وری به صورت زیر تعریف می شود:

$$U(n) = `n / n - 1$$

این کسر نشان دهنده تعداد برچسب های اضافی است که نیاز داریم تا دقت مدل به دقت مدلی که fine-tune شده برسد. اگر میزان دقت مدل self-supervised با همان تعداد برچسب با دقت مدل برابر شود، $U(n)$ برابر با صفر می شود. اگر هیچ تعداد برچسبی وجود نداشته باشد که که دقت این دو مدل باهم برابر شود، $U(n)$ به بی نهایت میل می کند.

برای اینکه تشخیص دهیم مدل بدون self-supervision با چه تعداد برچسب اضافی میتواند به دقت مدل با self-supervision برسد، این معیار تعریف شده است.

ب) این نوع تسک ها میتوانند در دسته های مختلف طبقه بندی شوند، مثلا دسته معنایی یا جغرافیایی، دسته ویژگی های سراسری یا متراکم. در این مقاله 4نوع تسک بررسی شده:

- Object classification: مدل به گونه ای آموزش دید که بتواند بین 10 شی ShapeNet (که در ساخت داده مصنوعی استفاده شده) طبقه بندی کند. این تصاویر تنها شامل یک شی هستند و توزیع یکسانی میان داده 10 کلاس وجود دارد. عملکرد با میزان دقت سنجیده شده است.
- Object pose estimation: این تصاویر نیز تنها شامل یک شی هستند. همچنین به گونه ای هستند که در یکی از 5 دسته بندی قرار گیرند (یکی از دلایلی که تنها به 5 دسته بندی اکتفا کرده اند، این است که بعضی اشیا مثل میز و لامپ تقارن چرخشی دارند). مدل باید پیش بینی کند که بالای جسم به سمت upward، forward، backward، راست یا چپ است. در این بخش نیاز است مدل ویژگی هایی برای فهمیدن موقعیت 3بعدی استخراج کند. از تابع ضرر cross entropy استفاده شده.
- Semantic segmentation: تصاویر با چندین شی ترکیب می شوند. در این تسک هدف آموزش مدلی که بتواند ماسک های دقیق و دارای رزولوشن بالا پیش بینی کند نیست، بلکه ماسک ها

رزولوشن بسیار درشت تری نسبت به تصویر ورودی دارند. از تابع ضرر cross entropy **برای هر پیکسل استفاده شده.**

- Depth estimation**: همچون تسک قبل، در تصاویر این تسک نیز چندین شی وجود دارند. برای آموزش مدلی که بتواند عمق اشیا را تشخیص دهد از** L1 loss **استفاده شده و برای گزارش میزان دقت، از درصد پیش بینی هایی که در نسبت معینی از عمق** ground truth **قرار گرفته اند استفاده می شود.**

ج) از 4 روش مختلف استفاده شده است:

- Variational autoencoder**: یک روش استاندارد برای مپ کردن تصاویر به فضای دارای ابعاد نهفته کمتر**
- Rotation**: مدل به گونه ای** pretrain **شده که تشخیص دهد ورودی دارای زاویه 0، 90، 180 و یا 270 درجه است.**
- Contrastive multiview coding**: در این روش تصویر را بر اساس کانال های آن اسپلیت می کنیم. مثلا اگر تصویر در فضای** Lab **باشد آن را به** L **و** ab **اسپلیت می کنیم. سپس آنها را از دو نیمه شبکه عبور میدهیم و** embedding**های خروجی باهم و در تضاد با** embedding**های تصاویر دیگر مقایسه می شوند.**
- Augmented multiscale deep InfoMax**: این روش مشابه** CMC **مدل را با** contrastive coding **آموزش می دهد. اما به جای استفاده از کانال های مختلف تصویر، از دو تصویری که در داده افزایی به یک تصویر اضافه شده اند استفاده می کند. همچنین از خروجی لایه های میانی شبکه استفاده می کند.**

3. **ابتدا سلول های خواسته شده را پیاده سازی می کنیم.**

در اولین سلول باید ورودی ها که به شکل جمله هستند را با استفاده از دیکشنری کلمات موجود، به برداری از اعداد تبدیل کنیم.

```python
def vectorize_stories(data, word_idx, story_maxlen, query_maxlen):

    ######################################
    #     Put your implementation here     #
    ######################################
    inputs_train = []
    queries_train = []
    answers_train = []

    for d in data:
      story, query, answer = d

      story_vec = np.zeros(story_maxlen)
      for i in range(len(story)):
        story_vec[i] = word_idx[story[i]]
      inputs_train.append(np.array(story_vec))

      query_vec = np.zeros(query_maxlen)
      for i in range(len(query)):
        query_vec[i] = word_idx[query[i]]
      queries_train.append(query_vec)

      ans_vec = np.zeros(len(word_idx) + 1)
      ans_num = word_idx[answer]
      ans_vec[ans_num] = 1
      answers_train.append(ans_vec)

    return np.array(inputs_train), np.array(queries_train), np.array(answers_train)
```

در دو سلول بعد، تابعی برای رسم نمودارهای دقت و میزان ضرر بر حسب ایپوک پیاده سازی می کنیم.

```python
def plot_acc(history, title):

  # This function should show not only the plot of accuracy on training and validation set
  # but also it should show the maximum value of accuracy with its related epoch.
  ######################################
  #     Put your implementation here     #
  ######################################
  plt.plot(history.history['accuracy'],label="train_accuracy")
  plt.plot(history.history['val_accuracy'],label="validation_accuracy")
  plt.xlabel("epoch")
  plt.ylabel(title)
  plt.legend()
  plt.tight_layout()
  plt.show()
```

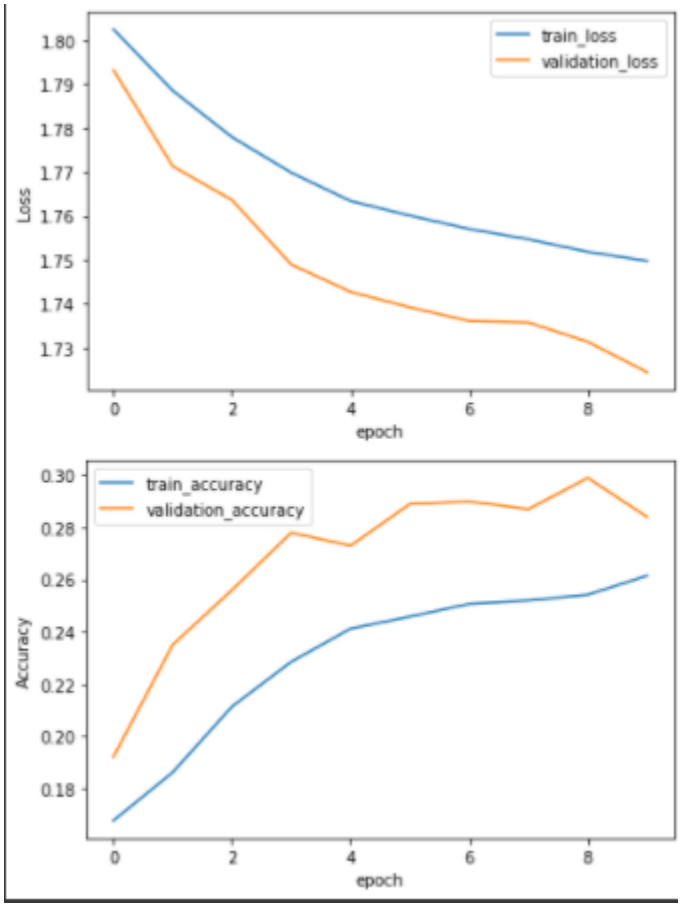
```python
def plot_loss(history, title):

  # This function should show not only the plot of loss on training and validation set
  # but also it should show the minimum value of loss with its related epoch.
  ######################################
  #     Put your implementation here     #
  ######################################
  plt.plot(history.history['loss'],label="train_loss")
  plt.plot(history.history['val_loss'],label="validation_loss")
  plt.xlabel("epoch")
  plt.ylabel(title)
  plt.legend()
  plt.tight_layout()
  plt.show()
```

در آخرین سلول مدل را مطابق تصویر داده شده پیاده سازی می کنیم.

```
from tensorflow.keras.layers import Embedding, Dot, Activation, Permute, Concatenate, LSTM, Dropout, Dense, Add
from tensorflow.keras import Sequential

# define the model:
input_sequence = tf.keras.layers.Input((story_maxlen,))
question = tf.keras.layers.Input((query_maxlen,))

print('Input sequence:', input_sequence)
print('Question:', question)

####################################
#     Put your implementation here     #
####################################
sequential1 = Sequential([Embedding(vocab_size, 64)])(input_sequence)
sequential2 = Sequential([Embedding(vocab_size, 4)])(input_sequence)
sequential3 = Sequential([Embedding(vocab_size, 64)])(question)
dot = Dot(axes=(2, 2))([sequential1, sequential3])
activation = Activation('relu')(dot)
add = Add()([activation, sequential2])
permute = Permute((2, 1))(add)
concat = Concatenate()([permute, sequential3])
lstm = LSTM(lstm_size)(concat)
dropout = Dropout(0.1)(lstm)
dense = Dense(vocab_size)(dropout)
answer = Activation('softmax')(dense)
```

در پایان مدل را در 10 ایپوک آموزش می دهیم. نتایج بدست آمده به شرح زیر است:

```
Epoch 1/10
  1/313 [..............................] - ETA: 14s - loss: 1.7716 - accuracy: 0.1875/usr/local/lib/python3.7/dist-packages/tensorflow/
  "Even though the `tf.config.experimental_run_functions_eagerly` "
313/313 [==============================] - 17s 55ms/step - loss: 1.8026 - accuracy: 0.1676 - val_loss: 1.7933 - val_accuracy: 0.1920
Epoch 2/10
313/313 [==============================] - 18s 57ms/step - loss: 1.7886 - accuracy: 0.1861 - val_loss: 1.7713 - val_accuracy: 0.2350
Epoch 3/10
313/313 [==============================] - 17s 53ms/step - loss: 1.7780 - accuracy: 0.2113 - val_loss: 1.7636 - val_accuracy: 0.2560
Epoch 4/10
313/313 [==============================] - 17s 53ms/step - loss: 1.7699 - accuracy: 0.2285 - val_loss: 1.7489 - val_accuracy: 0.2780
Epoch 5/10
313/313 [==============================] - 17s 55ms/step - loss: 1.7634 - accuracy: 0.2412 - val_loss: 1.7427 - val_accuracy: 0.2730
Epoch 6/10
313/313 [==============================] - 17s 53ms/step - loss: 1.7601 - accuracy: 0.2458 - val_loss: 1.7392 - val_accuracy: 0.2890
Epoch 7/10
313/313 [==============================] - 17s 54ms/step - loss: 1.7571 - accuracy: 0.2507 - val_loss: 1.7361 - val_accuracy: 0.2900
Epoch 8/10
313/313 [==============================] - 17s 55ms/step - loss: 1.7547 - accuracy: 0.2520 - val_loss: 1.7357 - val_accuracy: 0.2870
Epoch 9/10
313/313 [==============================] - 17s 54ms/step - loss: 1.7518 - accuracy: 0.2542 - val_loss: 1.7313 - val_accuracy: 0.2990
Epoch 10/10
313/313 [==============================] - 16s 53ms/step - loss: 1.7498 - accuracy: 0.2615 - val_loss: 1.7244 - val_accuracy: 0.2840
```



مدل نتوانسته در 10 ایپوک به دقت بالایی برسد و دچار underfit شده است. اما دقت میان آموزش و آزمون نشان دهنده این است که مدل ورودی ها را حفظ نکرده بلکه آنها را یاد گرفته است. اگر مدل را در ایپوک های بیشتری آموزش دهیم به دقت بسیار بهتری خواهیم رسید.

به طور کلی این مدل سعی می کند با دریافت چندین جمله به عنوان یک داستان آن را بیاموزد و سپس پرسش مربوط به آن را پاسخ دهد. در ابتدا لایه امبدینگ story و query در هم ضرب داخلی می شوند و سپس امبدینگ story به آن افزوده میشود. از یک لایه بازگشتی برای یادگیری بهتر مدل استفاده می کنیم.

مزیت: به علت استفاده از چند Sequential و ترکیب story و query، این شبکه می تواند ابزار خوبی برای سیستم های پرسش و پاسخ باشد. همچنین شبکه ساده است و درک آن پیچیدگی خاصی ندارد.

معایب: به دلیل کوچک بودن دیکشنری کلمات موجود در دیتاست، مدل آموخته شده را نمی توانیم به هر نوع داستانی تعمیم دهیم و تنها میتوان از داستان هایی که شامل 21 کلمه موجود در دیکشنری هستند استفاده کرد.

برای اینکه بتوانیم از این مدل برای کارهای واقعی استفاده کنیم: باید مدل را در ایپوک های بسیار بیشتری آموزش دهیم، باید دیکشنری طیف کلمات بسیار گسترده تری را شامل شود. بهتر است از بردارهای embedding از پیش آموخته شده از جمله word2vec یا glove استفاده کنیم تا میان بردارهای کلمات

فاصله معنایی صحیح وجود داشته باشد. همچنین می توانیم از زبان محاوره نیز برای آموزش مدل استفاده کنیم. در این صورت کاربرد مدل برای مسائل روزمره بیشتر خواهد شد.

4. ابتدا روی داده پیش پردازش انجام میدهیم. یعنی داده های ورودی را نرمالایز میکنیم و بردار برچسب را به صورت one-hot انکود می کنیم.

```
from tensorflow.keras.utils import to_categorical

# preprocess dataset
x_unlabeld_normalized = np.divide(x_unlabeld.astype('float32'), 255)
x_train_normalized = np.divide(x_train.astype('float32'), 255)
y_train_onehot = to_categorical(y_train, num_classes=10)

# preprocess trainset
x_test_normalized = np.divide(x_test.astype('float32'), 255)
y_test_onehot = to_categorical(y_test, num_classes=10)
```

یک تابع برای ساختن چندین لایه کانولوشنی با ظرفیت بالا تعریف میکنیم تا در قسمت های بعدی از آن استفاده کنیم.

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, BatchNormalization, Dropout
from tensorflow.keras.optimizers import SGD, Adam

def convolutional(p):
  model = Sequential()
  model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)))
  model.add(BatchNormalization())
  model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
  model.add(BatchNormalization())
  model.add(MaxPooling2D((2, 2)))
  model.add(Dropout(p))
  model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
  model.add(BatchNormalization())
  model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
  model.add(BatchNormalization())
  model.add(MaxPooling2D((2, 2)))
  model.add(Dropout(p))
  model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
  model.add(BatchNormalization())
  model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
  model.add(BatchNormalization())
  model.add(MaxPooling2D((2, 2)))
  model.add(Dropout(p))
  return model
```

الف) مدل را با چندین لایه کانولوشن به همراه دو لایه Dense برای دسته بندی تعریف می کنیم.

```
model1 = convolutional(0.2)
model1.add(Flatten())
model1.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model1.add(Dense(10, activation='softmax'))

model1.summary()
```

با استفاده از بهینه ساز Adam آن را در 100 ایپوک آموزش میدهیم. نتایج چند ایپوک پایانی به صورت زیر است:

```
Epoch 95/100
4/4 [==============================] - 21s 7s/step - loss: 0.1022 - accuracy: 0.9750 - val_loss: 6.4887 - val_accuracy: 0.2554
Epoch 96/100
4/4 [==============================] - 21s 7s/step - loss: 0.1395 - accuracy: 0.9600 - val_loss: 6.6277 - val_accuracy: 0.2443
Epoch 97/100
4/4 [==============================] - 21s 7s/step - loss: 0.1319 - accuracy: 0.9600 - val_loss: 6.4494 - val_accuracy: 0.2422
Epoch 98/100
4/4 [==============================] - 21s 7s/step - loss: 0.1396 - accuracy: 0.9500 - val_loss: 6.3475 - val_accuracy: 0.2355
Epoch 99/100
4/4 [==============================] - 20s 7s/step - loss: 0.1013 - accuracy: 0.9700 - val_loss: 7.4126 - val_accuracy: 0.2196
Epoch 100/100
4/4 [==============================] - 21s 7s/step - loss: 0.2860 - accuracy: 0.9100 - val_loss: 8.0633 - val_accuracy: 0.2086
```

این مدل شدیدا دچار overfit شده است. این مسئله تنها شامل 200 داده آموزشی بوده، به همین دلیل مدل الگوها را حفظ کرده و دقت بسیار پایینی روی داده تست بدست آورده است.

ب) ابتدا داده مورد نیاز را تولید میکنیم. برای هر تصویر، 3 داده جدید با زاویه مختلف به دیتا اضافه می شود.

```python
# Q4, part 2
def rotate(img, i):
  # rotate image i * 90 degree times
  y = np.zeros(4)
  x = np.rot90(img, k=i)
  y[i] = 1
  return x, y

def ssl_data(data):
  x_rotated = []
  y_rotated = []
  for d in data:
    for i in range(4):
      x, y = rotate(d, i)
      x_rotated.append(x)
      y_rotated.append(y)

  return x_rotated, y_rotated

x_rotated, y_rotated = ssl_data(x_unlabeld_normalized)
```

```python
x_rotated_arr = np.array(x_rotated)
y_rotated_arr = np.array(y_rotated)

# 4 train data for each image
print(x_rotated_arr.shape)
print(y_rotated_arr.shape)
```

```
(199200, 32, 32, 3)
(199200, 4)
```

مدل را با چندین لایه کانولوشن به همراه دو لایه Dense برای دسته بندی تعریف می کنیم. در لایه آخر 4 نورون داریم (زیرا دسته بندی زاویه 4 کلاسه است). این نوع آموزش self-supervised است.

```
model2 = convolutional(0.2)
model2.add(Flatten())
model2.add(Dense(64, activation='relu', kernel_initializer='he_uniform'))
model2.add(Dense(4, activation='softmax'))

model2.summary()
```

مدل را در 15 ایپوک آموزش میدهیم. نتایج چند ایپوک پایانی به صورت زیر است:

```
Epoch 10/15
779/779 [==============================] - 45s 57ms/step - loss: 0.3582 - accuracy: 0.8653
Epoch 11/15
779/779 [==============================] - 46s 59ms/step - loss: 0.3383 - accuracy: 0.8728
Epoch 12/15
779/779 [==============================] - 45s 57ms/step - loss: 0.3235 - accuracy: 0.8792
Epoch 13/15
779/779 [==============================] - 45s 57ms/step - loss: 0.3109 - accuracy: 0.8829
Epoch 14/15
779/779 [==============================] - 45s 58ms/step - loss: 0.2972 - accuracy: 0.8893
Epoch 15/15
779/779 [==============================] - 45s 57ms/step - loss: 0.2834 - accuracy: 0.8945
<keras.callbacks.History at 0x7f0c1a36ac10>
```

مدل تا اینجا توانسته دقت خوبی کسب کند. اما هدف اصلی ما از این آموزش، انتقال وزن های آموخته شده به تسک دیگری است. لایه آخر شبکه را با یک لایه Dense دیگر عوض میکنیم تا توانایی مدل در دسته بندی 10 کلاسه برای تسک جدید را ببینیم.

```
from keras.models import Model

new_layer = Dense(10 ,activation='softmax')(model2.layers[-2].output)
model2_2 = keras.Model(model2.inputs, new_layer)
```

مدل را در 50 ایپوک آموزش میدهیم. بخشی از نتایج به صورت زیر است:

```
Epoch 1/50
4/4 [==============================] - 1s 26ms/step - loss: 14.3403 - accuracy: 0.1050
Epoch 2/50
4/4 [==============================] - 0s 28ms/step - loss: 5.9600 - accuracy: 0.2150
Epoch 3/50
4/4 [==============================] - 0s 27ms/step - loss: 2.0161 - accuracy: 0.4300
Epoch 4/50
4/4 [==============================] - 0s 27ms/step - loss: 1.3006 - accuracy: 0.5200
Epoch 5/50
4/4 [==============================] - 0s 27ms/step - loss: 1.0979 - accuracy: 0.6200
```

.....

```
Epoch 45/50
4/4 [==============================] - 0s 27ms/step - loss: 0.0107 - accuracy: 0.9950
Epoch 46/50
4/4 [==============================] - 0s 27ms/step - loss: 0.0114 - accuracy: 1.0000
Epoch 47/50
4/4 [==============================] - 0s 29ms/step - loss: 0.0071 - accuracy: 1.0000
Epoch 48/50
4/4 [==============================] - 0s 27ms/step - loss: 0.0132 - accuracy: 0.9950
Epoch 49/50
4/4 [==============================] - 0s 27ms/step - loss: 0.0103 - accuracy: 0.9950
Epoch 50/50
4/4 [==============================] - 0s 27ms/step - loss: 0.0074 - accuracy: 1.0000
```

در ابتدای آموزش دقت پایین بوده اما بعد از گذشت 50 ایپوک مدل به خوبی توانسته الگوها را برای دسته بندی بیاموزد. استفاده از رویکرد یادگیری خودنظارتی می تواند در آموزش مدل بسیار موثر واقع شود.

پ) ابتدا داده های مورد نیاز را تولید میکنیم.

```
from sklearn.utils import shuffle

# 200 images have class label, make rotations data too
x_labeled_rotated, y_labeled_rotated = ssl_data(x_train_normalized)
y_train_repeat = np.repeat(y_train, 4)
y_train_repeat_onehot = to_categorical(y_train_repeat, num_classes=10)
print(y_train_repeat_onehot.shape)

# make class label (tensor of zeros) for unlabeled data
y_unlabeld = np.zeros((x_unlabeld.shape[0]*4, 10))
print(y_unlabeld.shape)

# concatenate data
x_train_all = np.concatenate((x_labeled_rotated, x_rotated_arr))
y_train_class = np.concatenate((y_train_repeat_onehot, y_unlabeld))
y_train_rotation = np.concatenate((y_labeled_rotated, y_rotated_arr))


x_train_all, y_train_class, y_train_rotation = shuffle(x_train_all, y_train_class, y_train_rotation, random_state=20)
print(x_train_all.shape)
print(y_train_class.shape)
print(y_train_rotation.shape)
```

```
(800, 10)
(199200, 10)
(200000, 32, 32, 3)
(200000, 10)
(200000, 4)
```

سپس مدل را مطابق چیزی که خواسته شده، یعنی با دو لایه خروجی، یکی شامل 10 نورون و دیگری 4 نورون می سازیم.

```
from keras.models import Model

base_model = convolutional(0.2)
f = Flatten()(base_model.layers[-1].output)
dense = Dense(128)(f)
class_number = Dense(10, name='class_num_output')(dense)

rotation_degree = Dense(4, name='rotation_degree_output')(dense)

model3 = Model(inputs=base_model.input,
               outputs = [class_number, rotation_degree])


model3.summary()
```

سپس با تنظیم وزن های مختلف برای تابع ضرر هریک از خروجی ها، عملکرد مدل را می سنجیم.

- **حالت اول:**

```python
model3.compile(optimizer=opt,
              loss={
                  'class_num_output': 'categorical_crossentropy',
                  'rotation_degree_output': 'categorical_crossentropy'},
              loss_weights={
                  'class_num_output': 2.,
                  'rotation_degree_output': 10.},
              metrics={
                  'class_num_output': 'accuracy',
                  'rotation_degree_output': 'accuracy'})
```

در اینجا وزن تابع ضرر خروجی دوم، 5 برابر وزن تابع ضرر خروجی اول است. مدل را در 10 اپیوک آموزش می دهیم. نتیجه آن به صورت زیر است:

```
50s 62ms/step - loss: 77.5096 - class_num_output_loss: 0.0424 - rotation_degree_output_loss: 7.7425 - class_num_output_accuracy: 0.0011 - rotation_degree_output_accur
48s 61ms/step - loss: 78.8335 - class_num_output_loss: 0.0451 - rotation_degree_output_loss: 7.8743 - class_num_output_accuracy: 4.0000e-04 - rotation_degree_output_a
48s 61ms/step - loss: 81.5652 - class_num_output_loss: 0.0441 - rotation_degree_output_loss: 8.1477 - class_num_output_accuracy: 4.0000e-04 - rotation_degree_output_a
48s 62ms/step - loss: nan - class_num_output_loss: nan - rotation_degree_output_loss: nan - class_num_output_accuracy: 0.9683 - rotation_degree_output_accuracy: 0.249
48s 62ms/step - loss: nan - class_num_output_loss: nan - rotation_degree_output_loss: nan - class_num_output_accuracy: 0.9964 - rotation_degree_output_accuracy: 0.250
48s 61ms/step - loss: nan - class_num_output_loss: nan - rotation_degree_output_loss: nan - class_num_output_accuracy: 0.9964 - rotation_degree_output_accuracy: 0.250
48s 61ms/step - loss: nan - class_num_output_loss: nan - rotation_degree_output_loss: nan - class_num_output_accuracy: 0.9964 - rotation_degree_output_accuracy: 0.250
48s 61ms/step - loss: nan - class_num_output_loss: nan - rotation_degree_output_loss: nan - class_num_output_accuracy: 0.9964 - rotation_degree_output_accuracy: 0.250
48s 61ms/step - loss: nan - class_num_output_loss: nan - rotation_degree_output_loss: nan - class_num_output_accuracy: 0.9964 - rotation_degree_output_accuracy: 0.250
47s 61ms/step - loss: nan - class_num_output_loss: nan - rotation_degree_output_loss: nan - class_num_output_accuracy: 0.9964 - rotation_degree_output_accuracy: 0.250
```

دقت تست و آزمون بعد از چند اپیوک به مقدار بسیار خوبی رسیده است.

- **حالت دوم:**

```python
model3.compile(optimizer=opt,
              loss={
                  'class_num_output': 'categorical_crossentropy',
                  'rotation_degree_output': 'categorical_crossentropy'},
              loss_weights={
                  'class_num_output': 2.,
                  'rotation_degree_output': 5.},
              metrics={
                  'class_num_output': 'accuracy',
                  'rotation_degree_output': 'accuracy'})
```

در اینجا وزن تابع ضرر خروجی دوم، 2.5 برابر وزن تابع ضرر خروجی اول است. مدل را در 10 اپیوک آموزش می دهیم. نتیجه آن به صورت زیر است:

```
61s 63ms/step - loss: 8.1395 - class_num_output_loss: 0.0327 - rotation_degree_output_loss: 1.6148 - class_num_output_accuracy: 0.0182 - rotation_degree_output_accura
49s 62ms/step - loss: 7.2024 - class_num_output_loss: 0.0322 - rotation_degree_output_loss: 1.4276 - class_num_output_accuracy: 0.3450 - rotation_degree_output_accura
48s 62ms/step - loss: 7.4030 - class_num_output_loss: 0.0338 - rotation_degree_output_loss: 1.4671 - class_num_output_accuracy: 0.6344 - rotation_degree_output_accura
49s 62ms/step - loss: 7.2337 - class_num_output_loss: 0.0305 - rotation_degree_output_loss: 1.4345 - class_num_output_accuracy: 0.0230 - rotation_degree_output_accura
48s 62ms/step - loss: 7.0868 - class_num_output_loss: 0.0299 - rotation_degree_output_loss: 1.4054 - class_num_output_accuracy: 0.0640 - rotation_degree_output_accura
48s 62ms/step - loss: 7.2846 - class_num_output_loss: 0.0321 - rotation_degree_output_loss: 1.4441 - class_num_output_accuracy: 0.0108 - rotation_degree_output_accura
48s 62ms/step - loss: 7.1982 - class_num_output_loss: 0.0314 - rotation_degree_output_loss: 1.4271 - class_num_output_accuracy: 0.0090 - rotation_degree_output_accura
48s 61ms/step - loss: 7.1248 - class_num_output_loss: 0.0321 - rotation_degree_output_loss: 1.4121 - class_num_output_accuracy: 0.0048 - rotation_degree_output_accura
48s 62ms/step - loss: 7.0325 - class_num_output_loss: 0.0324 - rotation_degree_output_loss: 1.3935 - class_num_output_accuracy: 4.2000e-04 - rotation_degree_output_ac
48s 62ms/step - loss: 7.0048 - class_num_output_loss: 0.0323 - rotation_degree_output_loss: 1.3880 - class_num_output_accuracy: 4.2500e-04 - rotation_degree_output_ac
```

مدل روی هیچ یک از خروجی ها دقت مناسبی کسب نکرده و به مرور زمان دقت آن کاهش یافته است.

- **حالت سوم:**

```
model3.compile(optimizer=opt,
               loss={
                   'class_num_output': 'categorical_crossentropy',
                   'rotation_degree_output': 'categorical_crossentropy'},
               loss_weights={
                   'class_num_output': 2.,
                   'rotation_degree_output': 0.5},
               metrics={
                   'class_num_output': 'accuracy',
                   'rotation_degree_output': 'accuracy'})
```

در اینجا وزن تابع ضرر خروجی دوم، ¼ وزن تابع ضرر خروجی اول است. مدل را در 10 اپوک آموزش می دهیم. نتیجه آن به صورت زیر است:

```
75s 81ms/step - loss: 4.1085 - class_num_output_loss: 0.0339 - rotation_degree_output_loss: 8.0816 - class_num_output_accuracy: 5.1000e-04 - rotation_degree_output_a
61s 78ms/step - loss: 4.1117 - class_num_output_loss: 0.0338 - rotation_degree_output_loss: 8.0881 - class_num_output_accuracy: 5.5000e-04 - rotation_degree_output_a
61s 78ms/step - loss: 4.1089 - class_num_output_loss: 0.0342 - rotation_degree_output_loss: 8.0812 - class_num_output_accuracy: 5.3500e-04 - rotation_degree_output_a
61s 79ms/step - loss: 4.1169 - class_num_output_loss: 0.0335 - rotation_degree_output_loss: 8.0997 - class_num_output_accuracy: 5.1000e-04 - rotation_degree_output_a
61s 79ms/step - loss: 4.1182 - class_num_output_loss: 0.0332 - rotation_degree_output_loss: 8.1036 - class_num_output_accuracy: 5.3000e-04 - rotation_degree_output_a
62s 79ms/step - loss: 4.1147 - class_num_output_loss: 0.0333 - rotation_degree_output_loss: 8.0964 - class_num_output_accuracy: 5.1000e-04 - rotation_degree_output_a
62s 79ms/step - loss: 4.1197 - class_num_output_loss: 0.0344 - rotation_degree_output_loss: 8.1018 - class_num_output_accuracy: 5.3000e-04 - rotation_degree_output_a
61s 78ms/step - loss: 4.1125 - class_num_output_loss: 0.0330 - rotation_degree_output_loss: 8.0927 - class_num_output_accuracy: 5.1500e-04 - rotation_degree_output_a
61s 79ms/step - loss: 4.1132 - class_num_output_loss: 0.0332 - rotation_degree_output_loss: 8.0935 - class_num_output_accuracy: 5.2000e-04 - rotation_degree_output_a
62s 79ms/step - loss: 4.1136 - class_num_output_loss: 0.0328 - rotation_degree_output_loss: 8.0960 - class_num_output_accuracy: 5.1500e-04 - rotation_degree_output_a
```

دقت مدل طی فرآیند آموزش تغییری نکرده است و اصلا پیش بینی مناسبی نداشته است.

نتیجه گیری نهایی قسمت پ: با تنظیم وزن مقدار تابع ضرر برای دسته بندی زاویه بر روی 5 برابر مقدار وزن تابع ضرر دسته بندی کلاس توانستیم دقت بسیار خوبی کسب کنیم.

منابع

https://smartlabai.medium.com/supervised-and-unsupervised-representation-learning-for-reinforcement-learning-d2529dcd83b9

https://chowdera.com/2021/01/20210109003603375e.html

https://stackoverflow.com/questions/41668813/how-to-add-and-remove-new-layers-in-keras-after-loading-weights