به نام خدا

درس یادگیری عمیق

تمرین سری چهارم

مدرس درس:
سرکار خانم دکتر داوودآبادی

تهیه شده توسط:
الناز رضایی ۹۸۴۱۱۳۸۷

تاریخ ارسال: ۱۴۰۱/۰۹/۱۷

سوال ۱ :

برای ماتریس ورودی زیر و کرنل داده شده زیر، عملیات کانولوشن را با فرض zero-padding بودن عملیات انجام دهید. نتایج را تحلیل کنید و ماتریس نتیجه را با ماتریس اولیه مقایسه کنید. (۱۵ نمره)

ماتریس ورودی

۰	۰	۱۰	۰	۰
۰	۰	۱۰	۰	۰
۰	۰	۱۰	۰	۰
۰	۰	۱۰	۰	۰
۰	۰	۱۰	۰	۰

کرنل ورودی

۱	۱	۱
۱	-۸	۱
۱	۱	۱

پاسخ ۱:

برای محاسبه کانولوشن، ابتدا باید کرنل را ۱۸۰ درجه بچرخانیم و سپس با استفاده از رابطه زیر، کانولوشن هر درایه از ماتریس ورودی را محاسبه کنیم. در اینجا چون کرنل نسبت به مرکز متقارن است، حاصل دوران یافته آن با خودش یکی می‌شود، بنابراین نیازی به چرخش نیست.

$$S(i,j) = \sum_m \sum_n I(i+m, j+n)k(m,n)$$

طبق این رابطه، مقادیر درایه‌های ماتریس حاصل از کانولوشن، به شرح زیر درمی‌آیند.

$$S(0,0) = 0\ ,\ S(0,1) = 20\ ,\ S(0,2) = -70\ ,\ S(0,3) = 20\ ,\ S(0,4) = 0$$

$$S(1,0) = 0\ ,\ S(1,1) = 30\ ,\ S(1,2) = -60\ ,\ S(1,3) = 30\ ,\ S(1,4) = 0$$

$$S(2,0) = 0\ ,\ S(2,1) = 30\ ,\ S(2,2) = -60\ ,\ S(2,3) = 30\ ,\ S(2,4) = 0$$

$$S(3,0) = 0\ ,\ S(3,1) = 30\ ,\ S(3,2) = -60\ ,\ S(3,3) = 30\ ,\ S(3,4) = 0$$

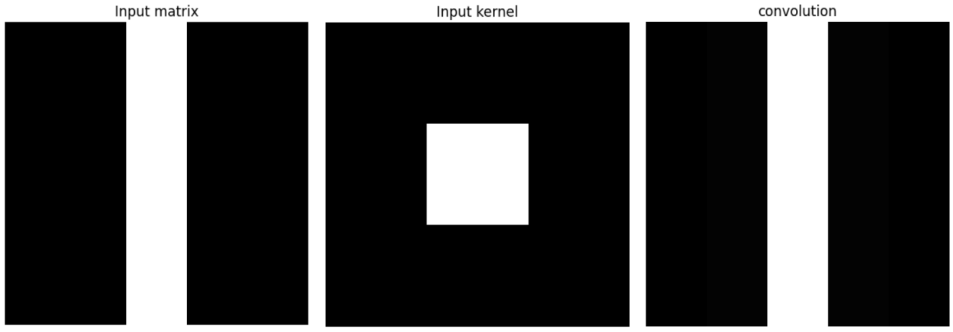
$$S(4,0) = 0\ ,\ S(4,1) = 20\ ,\ S(4,2) = -70\ ,\ S(4,3) = 20\ ,\ S(4,4) = 0$$

بنابراین شکل نهایی پس از محاسبه کانولوشن، به شکل زیر درمی‌آید:

۰	۲۰	-۷۰	۲۰	۰
۰	۳۰	-۶۰	۳۰	۰
۰	۳۰	-۶۰	۳۰	۰
۰	۳۰	-۶۰	۳۰	۰
۰	۲۰	-۷۰	۲۰	۰

کرنل استفاده شده در این سوال، در واقع لاپلاسین ماتریس ورودی را محاسبه می‌کند. لاپلاسین تغییرات شدت روشنایی را برجسته می‌کند و همچنین یک لبه‌یاب محسوب می‌شود. این کرنل در واقع از ماتریس ورودی مشتق دوم گرفته و نرخ تغییرات معکوس مشتق اول را نمایش می‌دهد. به بیان دیگر، اگر در مقدار پیکسل همسایه تغییری داشته باشیم، یک لبه داریم. این کرنل، نواحی که تغییرات شدت روشنایی آن خیلی سریع است را به خوبی تشخیص می‌دهد.

برای عملی‌تر شدن این بخش، کد مربوطه هم پیاده‌سازی شد و نتایج حاصل در تصویر زیر نمایان است.



کدهایی که برای این بخش به کار رفت، در زیر هر تصویر، نظیر به نظیر قرار گرفته است.

```
input = np.zeros((5,5), dtype=np.uint8)
for i in range(5):
    input[i][2] = 1
plt.imshow(abs(input), cmap='gray')
plt.axis('off')
plt.title('Input matrix')
plt.show()
```

```
kernel = np.ones((3,3), dtype=np.uint8)
kernel[1][1] = -8
plt.imshow(abs(kernel), cmap='gray')
plt.axis('off')
plt.title('Input kernel')
plt.show()
```

```
convolution = cv2.filter2D(input, -1, kernel)
plt.imshow(abs(convolution), cmap='gray')
plt.axis('off')
plt.title('convolution')
plt.show()
```

سوال ۲:

کانولوشن‌های ۱ در ۱ نوع خاصی از کانولوشن‌ها هستند که در چند سال اخیر مورد توجه زیادی واقع شده‌اند و توانسته‌اند عملکرد قابل توجهی از خود نشان دهند. در این مقاله، از این عملگرها به عنوان شبکه‌ای درون شبکه‌ی عصبی دیگر یاد شده است. با مطالعه‌ی آن و یا تماشای این ویدیو (از ویدیوهای Deep learning specialization آقای Andrew NG) درباره‌ی نحوه‌ی عملکرد این نوع کانولوشن‌ها و اینکه چرا به آنها شبکه‌ای درون شبکه‌ی دیگر گفته می‌شود، برداشت‌های خود را در حداکثر یک صفحه بنویسید.(اگر از منابع دیگری استفاده کرده‌اید حتما ذکر کنید.) (۱۰ نمره)

پاسخ ۲:

کانولوشن‌های ۱ در ۱، ابعاد مکانی را حفظ کرده و عمق را کاهش می‌دهند. به عبارت دیگر، ویژگی‌های یک مکان خاص را ترکیب کرده و یک ویژگی جدید می‌سازند. این نوع کانولوشن، در واقع استخراج ویژگی با در نظر گرفتن مکان می‌باشد. پس، تا اینجا متوجه شدیم که این فیلتر، بعد سوم تصویر یا همان عمق را کاهش می‌دهد. مثلا اگر ما یک لایه 192 * 28 * 28 داشته باشیم و ۳۲ فیلتر کانولوشنی 1 * 1 استفاده کنیم، ابعادمان به 32 * 28 * 28 تغییر می‌کند که نشان می‌دهد عمق تصویر کاهش یافته است. همچنین اصطلاح شبکه‌ای درون شبکه‌ی دیگر به این خاطر گفته می‌شود که لایه کانولوشنی ما، یک بردار c * 1 * 1 از تصویر را با یک فیلتر c * 1 * 1 ضرب می‌کند و مانند لایه fully connected عمل می‌کند.

سوال ۳:

اگر در یک شبکه عصبی کانولوشنی، ورودی‌های ما ۱*۲۸*۲۸ باشد، به سوالات زیر پاسخ دهید:

- الف) اگر در لایه ی اول، ۳۲ کرنل ۳ در ۳ با padding نوع same و stride ۲ اعمال کنیم، اندازه‌ی خروجی این لایه چند است؟ (۵ نمره)
- ب) اگر خروجی لایه ی دوم را به یک عملگر max pooling با padding از نوع valid و stride ۲ با کرنل‌هایی به ابعاد ۲ در ۲ دهیم، خروجی آن چه ابعادی دارد؟ (۵ نمره)

- پ) فرض کنید مسئله‌ی ما یک مسئله‌ی کلاسه بندی ۵ کلاسه است، می‌خواهیم، خروجی max pooling ابتدا flat شده، برای بدست آوردن ویژگی‌های بیشتر از ورودی، به یک لایه‌ی Dense و سپس به لایه‌ی خروجی بروند. ماتریس وزن‌های لایه یکی مانده به آخر و آخر چه ابعادی دارند؟ (۵ نمره)

پاسخ ۳:

- الف) می‌دانیم که padding از نوع same یعنی در واقع padding نداریم و stride ۲ هم ابعاد مکانی (بعد اول و دوم) را نصف می‌کند. همچنین بعد سوم هم برابر با تعداد فیلترها می‌شود. بنابراین ابعاد به 32 * 14 * 14 تغییر می‌کنند.
- ب) در این بخش نیز stride باعث نصف شدن ابعاد مکانی شده و بعد سوم تغییر نمی‌کند. پس ابعاد خروجی این لایه، 32 * 7 * 7 می‌شود.
- پ) می‌دانیم که لایه flatten وزنی ندارد، پس وزن‌های لایه dense را به دست می‌آوریم. بنابراین داریم:

$$Flatten = 0$$

$$Dense = (7 * 7 * 32 + 1) * 5 = 7845$$

سوال ۴:

ورودی یک لایه همگشتی (X) با ابعاد سه در سه را در نظر بگیرید. فیلتر F با ابعاد ۲*۲ روی ورودی X اعمال شده است. روی خروجی این لایه همگشتی، یک لایه ادغام میانگین سراسری اعمال(GAP) می‌شود که خروجی نهایی یک عدد خواهد شد. با توجه به این که گرادیان تابع اتلاف نسبت به این خروجی نهایی که یک عدد است ۱ میشود، با استفاده از الگوریتم پس انتشار خطا گرادیان‌های این لایه همگشتی را به دست آورید. (۲۰ نمره)

X

۲	۳	۴
۳	۱	۵
۴	-۱	-۲

F

۰	۳
۱	-۲

پاسخ ۴:

با اعمال فیلتر F بر روی X داریم:

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22} = (2*0) + (3*3) + (3*1) + (1*(-2)) = 0 + 9 + 3 + (-2) = 10$$

$$O_{12} = X_{12}F_{11} + X_{13}F_{12} + X_{22}F_{21} + X_{23}F_{22} = (3*0) + (4*3) + (1*1) + (5*(-2)) = 0 + 12 + 1 + (-10) = 3$$

$$O_{21} = X_{21}F_{11} + X_{22}F_{12} + X_{31}F_{21} + X_{32}F_{22} = (3*0) + (1*3) + (4*1) + ((-1)*(-2)) = 0 + 3 + 4 + 2 = 9$$

$$O_{22} = X_{22}F_{11} + X_{23}F_{12} + X_{32}F_{21} + X_{33}F_{22} = (1*0) + (5*3) + ((-1)*1) + ((-2)*(-2)) = 0 + 15 + (-1) + 4 = 18$$

بنابراین ماتریس خروجی این مرحله، مطابق شکل زیر می‌شود.

O

۱۰	۳
۹	۱۸

حال با اعمال Global Average Poolig داریم:

$$\frac{10 + 3 + 9 + 18}{4} = 10$$

گرادیان تابع اتلاف، طبق صورت سوال، مانند شکل زیر می‌شود.

$$\frac{\partial L}{\partial o}$$

۱	۱
۱	۱

حال با محاسبه کانولوشن این گرادیان اتلاف و X، گرادیان‌های این لایه کانولوشنی به دست می‌آید.

$$\frac{\partial L}{\partial F}$$

۹	۱۳
۷	۳

سوال ۵:

مجموعه داده‌ای به شما داده شده است. این مجموعه داده، از سایت‌های فروش خودرو ایرانی جمع‌آوری شده و شامل ۵ خودرو است. می‌خواهیم این ۵ خودرو را در خیابان کلاسه بندی کنیم.

الف) ابتدا مجموعه‌ی داده‌ی خود را بخوانید و با آن کار کنید تا به داده‌ی مناسب برای آموزش شبکه‌ی عصبی تبدیل شود. سپس با استفاده از ابزار keras بهترین شبکه‌ای را که می‌توانید آموزش دهید. به این منظور از فایل carClassifier.ipynb و قسمت‌های خواسته شده را بر اساس راهنمایی‌های داده شده تکمیل کنید. (۴۰ نمره)

ب) برای بهترین مدل، CAM-Grad را برای چند تصویر نمونه نمایش دهید و تحلیل کنید که دلیل عملکرد مناسب شبکه چه چیزی بوده است.

پاسخ ۵:

الف) در ابتدا، فرآیندهای قبل آموزش که شامل تقسیم dataset به سه بخش train، test و validation می‌باشد، انجام می‌دهیم. در اینجا، ۸۰ درصد داده‌ها را برای آموزش، ۱۰ درصد برای validation و ۱۰ درصد به عنوان تست در نظر گرفته شده‌اند.

```
labels = os.listdir("./dataset")

os.mkdir("data")
os.mkdir("data/train")
os.mkdir("data/test")
os.mkdir("data/validation")

for label in labels:
  filenames = os.listdir(f"./dataset/{label}")
  random.shuffle(filenames)

  split_1 = int(0.8 * len(filenames))
  split_2 = int(0.9 * len(filenames))

  os.mkdir(os.path.join("data/train",label))
  os.mkdir(os.path.join("data/validation",label))
  os.mkdir(os.path.join("data/test",label))

  train_filenames = filenames[:split_1]
  validation_filenames = filenames[split_1:split_2]
  test_filenames = filenames[split_2:]

  for tf in train_filenames:
    shutil.copy(f"./dataset/{label}/{tf}", f"data/train/{label}/{tf}")
  for vf in validation_filenames:
    shutil.copy(f"./dataset/{label}/{vf}", f"data/validation/{label}/{vf}")
  for tf in test_filenames:
    shutil.copy(f"./dataset/{label}/{tf}", f"data/test/{label}/{tf}")
```

سپس مدل خود را با ساختار زیر در نظر می‌گیریم.

```
model = Sequential()
model.add(Conv2D(32, 3, padding="same", activation='relu', input_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, 3, padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, 3, padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.compile(
  optimizer=Adam(learning_rate=1e-3),
  loss=CategoricalCrossentropy(),
  metrics=['accuracy']
)

summary = model.summary()
print(summary)
```

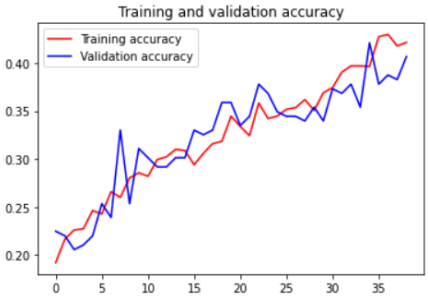
سپس با کمک لینک‌های داده شده، callbackها را تعریف می‌کنیم. کاربرد early stopping، برای این است که اگر metricهای مورد ارزیابی ما بهبود پیدا نکردند، training متوقف شود. ReduceLROnPlateau نیز در صورتی که metricمان بهتر نشود، learning rate را کاهش می‌دهد. ModelCheckpoint هم برای ذخیره وزن مدل Keras یا مدل در برخی فرکانس‌ها می‌باشد.

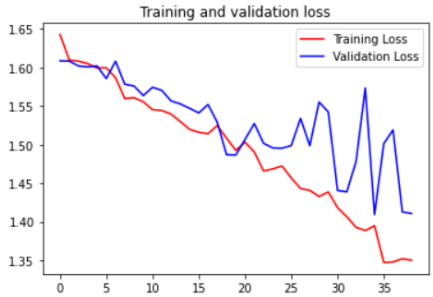
```
early_stopping = EarlyStopping(
  monitor='loss',
  patience=3
)
reduce_lr = ReduceLROnPlateau(
  monitor='val_loss',
  factor=0.2,
  patience=5,
  min_lr=0.001,
  verbos=0
)
model_checkpoint = ModelCheckpoint(
  filepath="./checkpoint",
  save_weights_only=True,
  monitor='val_accuracy',
  mode='max',
  save_best_only=True,
  verbos=0
)

callbacks = [early_stopping, reduce_lr, model_checkpoint]
```

در ادامه، ابتدا با استفاده از ImageDataGenerator، عملیات داده‌افزایی یا data augmentation را انجام می‌دهیم و مجموعه داده‌های خود را برای train و validation می‌سازیم. قابل ذکر است که برای train بهتر، از تغییرات بیشتری در تصویر استفاده کردیم. بدین منظور، تغییرات برای محدوده چرخش را ۴۰ در نظر گرفتیم. همچنین شیفت طولی و عرضی را حدود ۰.۲ تعریف کردیم. تصاویر هم با تقسیم بر ۲۵۵، نرمالیزه ساختیم. محدوده zoom-in و zoom-out هم ۰.۲ فرض کردیم. حالت آینه‌ای هم در نظر گرفتیم و سپس با shuffle کردن dataهایمان، مجموعه داده train را ساختیم. برای داده‌های validation، فقط تصاویر را نرمالیزه کردیم و مجموعه داده را ساختیم. سپس مدل را ساخته و با استفاده از مجموعه داده‌های ساخته شده و با در نظر گرفتن ۵۰ epoch، مدل خود را fit کردیم. به علاوه از callbackهای تعریف شده در مرحله قبل، برای fit کردن مدل خود استفاده می‌کنیم.

```
train_datagen = ImageDataGenerator(
  rotation_range=40,
  width_shift_range=0.2,
  height_shift_range=0.2,
  rescale=1./255,
  shear_range=0.2,
  zoom_range=0.2,
  horizontal_flip=True,
  fill_mode='nearest'
)
train_iterator = train_datagen.flow_from_directory(
  f'{BASE_FOLDER}train',
  target_size=(150, 150),
  batch_size=BATCH_SIZE,
  class_mode='categorical',
  shuffle=True,
  seed=42
)
validation_datagen = ImageDataGenerator(
  rescale=1./255,
)
validation_iterator = validation_datagen.flow_from_directory(
  f'{BASE_FOLDER}validation',
  target_size=(150, 150),
  batch_size=BATCH_SIZE,
  class_mode='categorical'
)
model = create_cnn_model()
history =  model.fit(
  train_iterator,
  epochs=EPOCHS,
  validation_data=validation_iterator,
  callbacks=create_callbacks()
)
```

در شکل‌های زیر، نمودارهای accuracy و loss برای این مدل مشاهده می‌کنید.





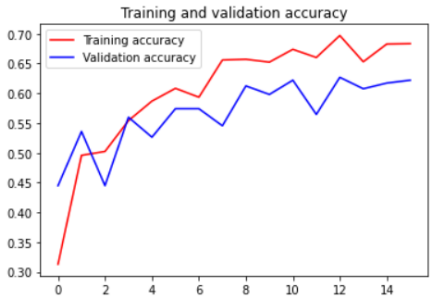
در این مدل، ما تعداد epoch را ۵۰ در نظر گرفته بودیم که به علت استفاده از callbackها، پس از

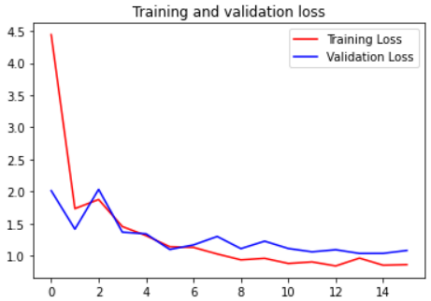
epoch ۳۹ متوقف شد.

بار دیگر، از مدل از پیش آموخته شده استفاده می‌کنیم. در اینجا، شبکه MobileNetV2 را به عنوان مدل از پیش آموخته شده، انتخاب کردیم و مدل خود را به آن اضافه کرده و عملیات fine tuning را انجام می‌دهیم.

```
added_model = MobileNetV2(
  input_shape=(150, 150, 3),
  include_top=False,
  weights="imagenet",
  classes=5,
  classifier_activation="softmax"
)

model = Sequential()
model.add(added_model)
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(5, activation='softmax'))

added_model.trainable = False
model.compile(
  optimizer=Adam(learning_rate=1e-3),
  loss=CategoricalCrossentropy(),
  metrics=['accuracy']
)
```

این مدل جدید ما، پس از ۱۶ از epoch ۵۰ متوقف می‌شود که نشان می‌دهد نسبت به مدل قبل، سریع‌تر عملیات trainnig انجام می‌شود. همچنین همانطور که در شکل‌های زیر که نمودار accuracy و loss مربوط به این مدل را مشاهده می‌کنید. همانطور که مشخص است، مقدار loss برای داده‌های train و validation این مدل بیشتر کاهش یافته است و accuracy آن هم برای داده‌های train و هم داده‌های validation افزایش یافته است.





ب) در ابتدا، با استفاده از تابع زیر، تصویر را به آرایه تبدیل می‌کنیم.

```
[61] def get_img_array(img_path, size):
         img = keras.preprocessing.image.load_img(img_path, target_size=(size))
         array = keras.preprocessing.image.img_to_array(img)
         array = np.expand_dims(array, axis=0)
         return array
```

در مرحله بعد، gradCam را برای تصویر به دست می‌آوریم.

```
def make_gradcam_heatmap(img_array, model, last_conv_layer_name, pred_index=None):
    grad_model = tf.keras.models.Model(
        [model.inputs], [model.get_layer(last_conv_layer_name).output, model.output]
    )

    with tf.GradientTape() as tape:
        last_conv_layer_output, preds = grad_model(img_array)
        if pred_index is None:
            pred_index = tf.argmax(preds[0])
        class_channel = preds[:, pred_index]

    grads = tape.gradient(class_channel, last_conv_layer_output)

    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))

    last_conv_layer_output = last_conv_layer_output[0]
    heatmap = last_conv_layer_output @ pooled_grads[..., tf.newaxis]
    heatmap = tf.squeeze(heatmap)

    heatmap = tf.maximum(heatmap, 0) / tf.math.reduce_max(heatmap)
    return heatmap.numpy()
```

سپس، برای بصری شدن gradCam، از تابع زیر استفاده می‌کنیم.

```
def make_superimposed(img_path, heatmap, cam_path, alpha=0.4):
    img = keras.preprocessing.image.load_img(img_path)
    img = keras.preprocessing.image.img_to_array(img)

    heatmap = np.uint8(255 * heatmap)

    jet = cm.get_cmap("jet")

    jet_colors = jet(np.arange(256))[:, :3]
    jet_heatmap = jet_colors[heatmap]

    jet_heatmap = keras.preprocessing.image.array_to_img(jet_heatmap)
    jet_heatmap = jet_heatmap.resize((img.shape[1], img.shape[0]))
    jet_heatmap = keras.preprocessing.image.img_to_array(jet_heatmap)

    superimposed_img = jet_heatmap * alpha + img
    superimposed_img = keras.preprocessing.image.array_to_img(superimposed_img)

    superimposed_img.save(cam_path)

    image = cv2.imread(cam_path)
    plt.imshow(image)
```

در انتها تصویر خود را به تابع $make\_superimposed$ می‌دهیم تا gradCam تصویر را به ما بدهد.

```
[64] img_array = preprocess_input(get_img_array("./data/test/iranKhodro_dena/129.jpg", size=(150,150)))
     model = create_cnn_model()
     model.layers[-1].activation = None
     heatmap = make_gradcam_heatmap(img_array, model, 'dense_15')
     make_superimposed("./data/test/iranKhodro_dena/129.jpg", heatmap)
```

در مورد عملکرد مناسب شبکه نیز، علت آن استفاده از شبکه از پیش آموخته می‌باشد.

نکته: در حل این سوال، از لینک‌های زیر کمک گرفته شد.

https://cs230.stanford.edu/blog/split/

https://keras.io/examples/vision/grad_cam/