تمرین نهم یادگیری عمیق

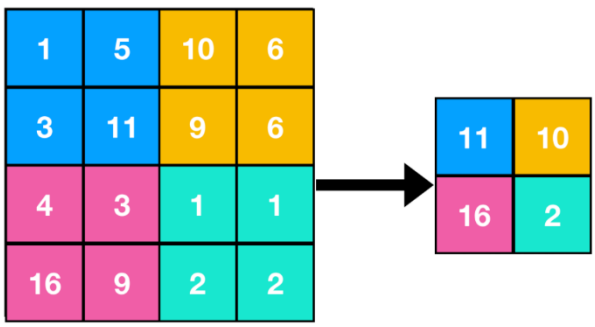
محمدعلی فراهت ۹۷۵۲۱۴۲۳

سوال ۱)

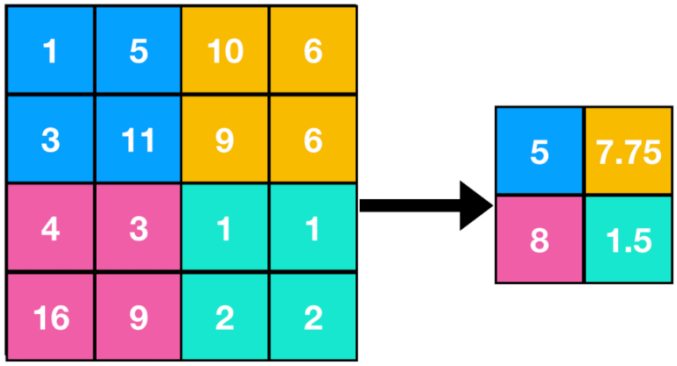
الف) 2D Convolution برای استفاده در ورودی‌های ۲ بعدی مثل عکس به کار می‌رود. این یعنی شما یک ورودی دو بعدی مثل دارید که هر پیکسل می‌تواند چند کانال داشته باشد (مثلا RGB)، در این حالت فیلتر ها باید ۳ بعدی باشند تا بتوانند ویژگی های مناسب را پیدا کنند.
3D Convolution برای ورودی‌های ۳ بعدی مثل فیلم استفاده می‌شود که در هر فریم خود یک عکس دو بعدی دارند. این لایه ها پیچیدگی بیشتری دارند و تعداد پارامترهایی که باید آموزش ببینند در آن بیشتر است. با این حال گاهی به آن ها نیاز داریم.

ب) اولا چون نمی‌دانیم ویژگی هایی که قرار است پیدا کنیم افقی هستند یا عمودی یا ترکیبی از هر دو، فیلترها را مربعی در نظر می‌گیریم . به عبارتی فیلترهای ما symmetric یا متقارن هستند. اما اگر از قبل اطلاعاتی داشته باشیم که بدانیم فیتر های مستطیلی مناسب تر هستند، می‌توانیم از آن ها استفاده کنیم.
انتخاب اندازه فیلتر ها راه مشخص و ثابتی ندارد و با توجه به نوع تصویر و اطلاعات مهم آن و همچنین سایز آن انتخاب می‌شود. ابعاد کرنل را زیاد بزرگ در نظر نمی‌گیرند چون که محاسبات را به شدت افزایش می‌دهد. تعداد معقول آن معمولا ۳، ۴ و ۵ است. برای مثال آموزش فیلتر ۱۳x۱۳ هفته‌ها زمان می‌برد. همچنین معمولا تعداد آن را فرد در نظر می‌گیرند. تا به امروز محبوب ترین سایز، ۳ است.

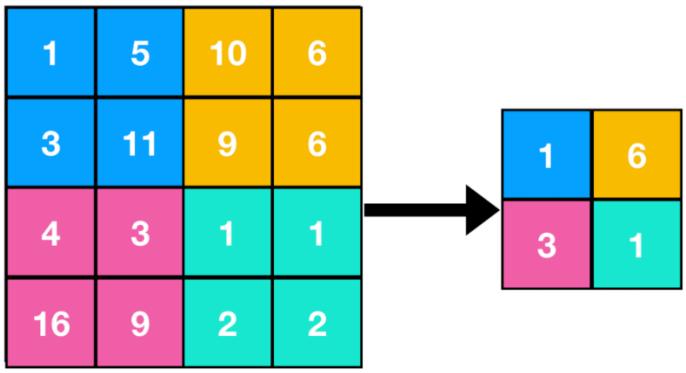
ج) چهار مدل pooling داریم :
max pooling: حداکثر مقادیر را در فیلتر نگه می‌دارد، برای مثال :



average pooling: مانند اسم خود است و میانگین مقادیر هر فیلتر را در خروجی نشان می‌دهد در این روش کلیت اعداد حفظ می‌شود و مانند روش قبل نیست. برای مثال :



Minimum Pooling: این روش هم مثل اسم خود، کمترین مقدار را در فیلتر برمی‌گرداند:

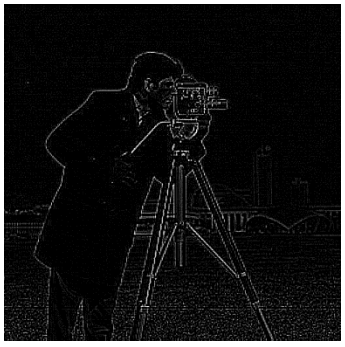


Adaptive Pooling: این روش جدیدتر از قبلی‌ها است. در این روش نیازی به تعریف دستی hyperparameter ها نداریم و فقط سایز خروجی را مشخص می‌کنیم و پارامتر ها بر این اساس انتخاب می‌شوند.

* از این لایه وقتی استفاده می‌شود که مکان ویژگی اهمیت کمی داشته باشد. مثلا اگر یک گربه برعکس هم در تصویر بود بتوانیم آن را تشخیص دهیم.

سوال ۲)

الف) اثر این فیلترها در زیر قابل مشاهده است (به همان ترتیب صورت سوال است):

اولین فیلتر (از چپ) تصویر را کمی تار می‌کند که به آن blur می‌گویند. در این فیلتر، میانگین همه پیکسل‌های اطراف جمع آوری می‌شود و جایگزین آن پیکسل می‌شود.

دومین فیلتر برای پیدا کردن لبه استفاده می‌شود همانطور که از ظاهر این فیلتر مشخص است، درون را ۸ برابر تاریک کرده و لبه را برعکس میکند. به این صورت لبه ها آشکار می‌شود.

فیلتر بعدی هم یک edge detector است اما فقط لبه های افقی را پیدا می‌کند و از روی اعداد این فیلتر هم میتوان این را تشخیص داد.

فیلتر آخر هم یک لبه یاب عمودی است و دقیقا مانند فیلتر سوم است با این تفاوت که لبه های عمودی را مشخص می‌کند.

❋ در زیر کد آن ها را هم می‌بینیم:

```
import numpy as np
from google.colab.patches import cv2_imshow
import cv2

filter0 = (1/9) * np.array(([1, 1, 1],
                            [1, 1, 1],
                            [1, 1, 1]))
filter1 = np.array(([-1, -1, -1],
                    [-1, 8, -1],
                    [-1, -1, -1]))
filter2 = np.array(([-1, -2, -1],
                    [0, 0, 0],
                    [1, 2, 1]))

filter3 = np.array(([-1, 0, 1],
                    [-2, 0, 2],
                    [-1, 0, 1]))

image_name = "img1.jpg"
image = cv2.imread(image_name, flags=cv2.IMREAD_GRAYSCALE)
cv2_imshow(image)

cv2_imshow(cv2.filter2D(image, -1, filter0))
cv2_imshow(cv2.filter2D(image, -1, filter1))
cv2_imshow(cv2.filter2D(image, -1, filter2))
cv2_imshow(cv2.filter2D(image, -1, filter3))
```

سوال ۳)

الف) این ابزاری است که keras برای بهینه سازی هایپرپارامتر های مدل تولید کرده و با استفاده از آن میتوان قبل از شروع ترین کردن مدل، مقادیری مثل تعداد لایه ها و نورون ها را به مقدار بهینه برای آن مسئله می‌رساند. راه حل های متفاوتی برای این کار وجود دارد که در ادامه آن ها را خواهیم دید.
اما چگونه از tuner استفاده کنیم؟ باید مقادیری که میخواهیم tun شوند را از قبل در model_builder مشخص کنیم و سپس با یکی از الگوریتم های tuner آن ها را بهینه کنیم.

ب) چهار مدل tuner در kerasTuner وجود دارد که نام آن ها RandomSearch ، Hyperband، BayesianOptimization و Sklearn است. و من از randomSearch استفاده می‌کنم زیرا این مدل سریع تر است ولی از معایب آن میتوان گفت که تضمین نمی‌کند که جواب بهینه را حتما بدهد.

ج) در این قسمت ابتدا یک مدل درست می‌کنیم و پارامترهایی که باید tuner تنظیم کند را به آن می‌دهیم:

```
def build_model(hp):
    model = keras.Sequential()
    model.add(Conv2D(hp.Int(f"features_c_{-1}", min_value=32, max_value=256, step=32),
                     3, activation ="relu", input_shape=(32, 32, 3)))

    for i in range(hp.Int("num_conv_layers", 0, 4)):
        model.add(
            Conv2D(
                hp.Int(f"features_c_{i}", min_value=32, max_value=256, step=32),
                3,
                activation='relu',
            )
        )
        #   model.add(Conv2D(8, kernel_size, activation ="relu", input_shape=(150, 150, 3)))
    model.add(Flatten())

    for i in range(hp.Int("num_dense_layers", 1, 5)):
        model.add(
            Dense(
                units=hp.Int(f"units_{i}", min_value=32, max_value=512, step=32),
                activation='relu',
            )
        )
    model.add(Dense(10, activation="softmax"))

    hp_learning_rate = hp.Float("lr", min_value=1e-4, max_value=1e-3, sampling="log")
    model.compile(optimizer=keras.optimizers.Adam(learning_rate=hp_learning_rate),
                loss= tf.keras.losses.CategoricalCrossentropy(),
                metrics=['accuracy'])
    return model
```

سپس با randomSearch مقادیر نسبتا بهینه را برای hyperparameter ها پیدا می کنیم:

```
[5]  tuner = kt.RandomSearch(
         hypermodel=build_model,
         objective="val_accuracy",
         max_trials=3,
         executions_per_trial=2,
         overwrite=True,
         directory="my_dir",
         project_name="cifar",
     )
```

```
[6]  tuner.search_space_summary()
     tuner.search(img_train,label_train,epochs=10,validation_data=(img_test,label_test))
```

```
Trial 3 Complete [00h 34m 19s]
val_accuracy: 0.7275999784469604

Best val_accuracy So Far: 0.7275999784469604
Total elapsed time: 01h 09m 11s
INFO:tensorflow:Oracle triggered exit
```

*** متاسفانه اجرا شدن این قسمت از کد ۸۰ دقیقه طول کشید و بار اول و دوم به دلیل مشکل ظرفیت استفاده از gpu در کولب، در دقایق پایانی متوقف شد. و مجبور شدم برای بار سوم آن را اجرا کنم و این بسیار دردناک بود :)

```
# Get the top 2 models.
models = tuner.get_best_models(num_models=2)
best_model = models[0]
# Build the model.
best_model.build(input_shape=(None, 32, 32, 3))
best_model.summary()
```

حالا مقادیر بهینه را برای این مدل می‌بینیم:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 160)       4480

 conv2d_1 (Conv2D)           (None, 28, 28, 128)       184448

 conv2d_2 (Conv2D)           (None, 26, 26, 192)       221376

 conv2d_3 (Conv2D)           (None, 24, 24, 224)       387296

 flatten (Flatten)           (None, 129024)            0

 dense (Dense)               (None, 256)               33030400

 dense_1 (Dense)             (None, 320)               82240

 dense_2 (Dense)             (None, 480)               154080

 dense_3 (Dense)             (None, 10)                4810

=================================================================
Total params: 34,069,130
Trainable params: 34,069,130
Non-trainable params: 0
```

```
import numpy as np

best_hps = tuner.get_best_hyperparameters(5)
# Build the model with the best hp.
model = build_model(best_hps[0])
# Fit with the entire dataset.
x_all = np.concatenate((img_train, img_test))
y_all = np.concatenate((label_train, label_test))
model.fit(x=x_all, y=y_all, epochs=1)
```

سپس مدل را فیت میکنیم و دقت آن را میبینیم:

```
1875/1875 [==============================] - 113s 60ms/step - loss: 1.3166 - accuracy: 0.5250
<keras.callbacks.History at 0x7f81f015bdd0>
```

با تشکر