به نام خدا

ملیکا احمدی رنجبر 97521036

1)

a. هنگام استفاده از 2D Convolution، همانطور که از نامش پیداست ما تنها دو بعد را از داده ورودی در نظر می‌گیریم، از عمق صرف نظر می‌کنیم. به این معنا که Filter ما یک 2D Matrix است که بر روی 2D Input در حال Movement است (تعداد Channel در 2D Convolution متفاوت است، با توجه به بعد سوم است، اما کار 3D Convolution را انجام نمی‌دهد). این در حالی است که استفاده از 3D Convolution امکان در نظر گرفتن عمق تصویر را برای ما فراهم می‌کند، در هنگام Movement بر روی 3D Input بعد سوم را نیز در نظر می‌گیرد. خروجی برای 2D یک تصویر دو بعدی است، اما در حال 3D خروجی دارای بعد سوم (عمق = Depth) می‌باشد. از مزیت‌های 2D Convolution می‌توان به حجم محاسباتی کمتر آن نسبت به 3D Convolution اشاره کرد، یا نمایش راحت آن، در حالی که 3D Convolution می‌تواند جزئیات بیشتری را برای ما در نظر بگیرد، بانبراین کاربردهای آن نیز بیشتر خواهد بود. یکی از Applicationهایی که 2D Convolution در آن استفاده می‌شود، تشخیص لبه در Input است، Sobel Edge Filter نام دارد. Applicationهایی که برای 3D Convolution موجود است می‌توان به استفاده آن در Videoها، یا تشخیص عمق اجسام در عکس اشاره کرد، که به عنوان مثال در Self-Driving Cars پراهمیت است، یا حتی 3D Medical Images.

b. دلیل استفاده از Square Filters آن است که ترجیحی در جهت پیدا کردن یک Pattern وجود ندارد. بنابراین عمودی یا افقی بودن یک Pattern مهم تفاوتی ندارد. شبکه با استفاده از Square Filter یک مدل Symmetric است. انتخاب Filter Size بر اساس آنکه در چندمین لایه است نیز می‌تواند انجام شود، اما نکته مهم آن است که با افزایش Kernel Size تعداد Parameters بسیار سریع افزایش می‌یابد، که با تعداد Channel زیاد اسصلا مناسب نیست، بنابراین به طور کلی Kernel Size کوچک بهتر است. اما می‌توان از اندازه کمی بزرگ‌تر در لایه‌های ابتدایی استفاده کرد. از نظر من علاوه بر موارد ذکر شده در بالا دقت برای Kernel Size کوچک‌تر بالاتر است و Patternها را با دقت بیشتری می‌تواند Extract کند. نکته دیگر هم برای انتخاب زوج یا فرد بودن است، که به طور معمول برای Symmetric بود ا اندازه فرد استفاده می‌کنیم.

c. 4 نوع Pooling داریم: Maximum, Minimum, Average, Adaptive

Pooling قصد دارد که تغییرات را برای Featureها در نظر نگیرد، تغییراتی مثل چرخش یا جابه‌جایی، و از اهمیت Location دقیق برای تشخیص یک Input می‌کاهد. برای مثال در Max با توجه به Filter Size بین Pixelهای ورودی (Locally و در محدوده خاص) بیشترین مقدار آن را انتخاب کرده

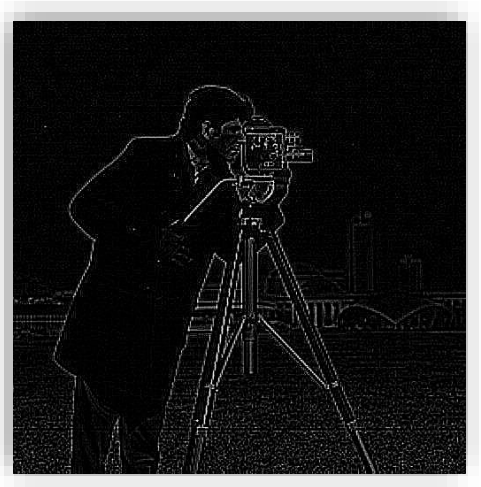
و به خروجی می‌دهیم. برای Min نیز به همین ترتیب، کمترین مقدار انتخاب می‌شود. Average Pooling میانگین تمام مقادیر موجود در Filter را در نظر گرفته و به عنوان مقدار خروجی در نظر می‌گیرد. با توجه به سایت موجود، یک روش جدید Adaptive Pooling است، به این ترتیب که تنها کافی است اندازه خروجی را بدهیم، و مشخص کردن مقادیر Hyper Parameters مثل Kernel Size، Stride و ... لازم نیست، تمام آن‌ها خودشان تنظیم می‌شوند. اما استفاده از Pooling برای کاهش حساسیت به جابه‌جاهای کوچک در ورودی داشته‌باشد، و بنابراین بازدهی را افزایش می‌دهد. از طرفی به دلیل کاهش اندازه و Parameter حجم محاسباتی لازم نیز کم می‌شود، نیاز به حافظه نیز همینطور (برای ذخیره Parameters).

2) به طور کلی در این مسئله Effectهای متفاوت بر روی تصویر زیر را با استفاده از لایه Convolution دیدیم.

a. اولین Filter در سمت چپ، Filter بسیاری معروفی برای محو کردن تصویر است، با توجه به Kernel Size میزان محو شدن (Blurring) می‌تواند تغییر کند:

تصویر سمت جپ با استفاده از Kernel Size = 3 است، و تصویر سمت راست با Kernel Size = 5 است (بیشتر محو شده).

b. در این Filter به طور کلی Outline Edge Detection داریم و لبه طرح کلی عکس ما مشخص می‌شود و همانطور که معلوم است لبه‌های تصویر به طور واضح نمایان شده‌است:



c. این مورد و بعدی Filterهایی هستند که لبه‌های عمودی و افقی در عکس را تشخیص می‌دهند، همانطور که از Kernel پیداست. Sobel Edge Detection نام دارند.
Filter اول برای لبه‌های افقی است و نقاطی که اختلاف در پیکسل‌های افقی دارند را نمایان می‌کند، همچون مکان‌هایی از کت مرد، و افق که ساختمان‌ها هستند:

مورد بعد نیز همانند این، اما برای اختلاف‌های عمودی همچون پایه‌های دوربین و لبه‌های عمودی کت:

(3

a. Keras Tuner برای کنترل و بهبود Hyper Parameterهای موجود در شبکه، همچون Learning Rate و حتی تعداد لایه‌ها، یا Activation Function می‌باشد. Hyper Parameter مقادیری هستند که وظیفه کنترل کردن Training Process را دارند. در Tuner ما می‌توانیم برای انتخاب شدن این مقادیر Min و Max در نظر گرفته و به آن بدهیم. بعد از مشخص شدن Search Space و مشخصات مد نظر ما، نوبت به انتخاب Tuner Class است تا Search آغاز شود.

b. 3 Tuner Class به طور کلی موجود است. Random Search, Bayesian Optimization, Hyperband. اگرچه که یک Tuner Class دیگر با اسم Sklearn نیز موجود است، اما در اینجا حرفی زده نمی‌شود. Random Search سعی می‌کند تمام حالت‌های ممکن را برای انتخاب Hyper Parameter امتحان کند (اما در زمانی کمتر از Grid Search) ، و البته جواب Optimal را نیز Guarantee نمی‌کند. Hyperband بر خلاف Random Search که ممکن با مقادیر با عملکرد بد را انتخاب کند و Training را آن انجام دهد، تعداد کمی Epoch را جلو می‌رود، و بهترین حالت‌ها را انتخاب می‌کند (با توجه به همین تعداد Epoch) و مرحله کامل Training را روی مقادیر انتخاب شده انجام می‌دهد. Bayesian Optimization مشکل هر دو Algorithm پیش را که تمام Combinationها را امتحان می‌کرد و ممکن بود پاسخ Optimal ندهد را حل می‌کند. تنها مقادیر اولیه را Random انتخای می‌کند، و سپس با توجه به عملکرد و History آن‌ها مقادیر ادامه راه را تعیید می‌کند.

من برای پیاده‌سازی این سوال از روش آخر استفاده می‌کنم، به دلیل Optimal بودن پاسخ آن نسبت به دو Algorithm دیگر.

c. برای تمامی Hyper Parameterها Tuner استفاده شده، شامل: تعداد لایه‌های Dense و Convolution، تعداد Neuron و Filter در هر کدام، Optimizer، Learning Rate، و حتی لایه Dropout.

```
for i in range(Hp.Int("num_layers", 0, 4)):
    Model.add(
        layers.Conv2D(
            # Tune Number Of Filters Separately.
            filters=Hp.Int(f"filters_{i}", min_value=32, max_value=256, step=32),
            kernel_size=(3, 3),
            activation="relu",
            padding="same"
        )
    )
    if Hp.Boolean("dropout"):
        Model.add(layers.Dropout(rate=0.1))
```

```
Model.add(layers.AveragePooling2D(pool_size=(2, 2)))
Model.add(layers.Flatten())
# Dense
for i in range(Hp.Int("num_layers", 0, 4)):
    Model.add(
        layers.Dense(
            # Tune Number Of Units Separately.
            units=Hp.Int(f"units_{i}", min_value=32, max_value=256, step=32),
            activation=Hp.Choice("activation", ["relu", "sigmoid"]),
        )
    )

if Hp.Boolean("dropout"):
    Model.add(layers.Dropout(rate=0.1))

# Final Layer
Model.add(layers.Dense(10, activation="softmax"))

# Compile
LearningRate = Hp.Float("lr", min_value=1e-4, max_value=1e-3, sampling="log")
Optimizer = Hp.Choice("optimizer", values=['adam', 'SGD'])

if Optimizer == 'adam':
    Optimizer= keras.optimizers.Adam(learning_rate=LearningRate)
elif Optimizer == 'SGD':
    Optimizer= keras.optimizers.SGD(learning_rate=LearningRate)

Model.compile(optimizer=Optimizer, loss="categorical_crossentropy", metrics=["accuracy"])
```

```
Search space summary
Default search space size: 5
filters (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 256, 'step': 32, 'sampling': None}
num_layers (Int)
{'default': None, 'conditions': [], 'min_value': 0, 'max_value': 4, 'step': 1, 'sampling': None}
dropout (Boolean)
{'default': False, 'conditions': []}
lr (Float)
{'default': 0.0001, 'conditions': [], 'min_value': 0.0001, 'max_value': 0.001, 'step': None, 'sampling': 'log'}
optimizer (Choice)
{'default': 'adam', 'conditions': [], 'values': ['adam', 'SGD'], 'ordered': False}
```

```
Trial 3 Complete [00h 05m 23s]
val_accuracy: 0.6704000234603882

Best val_accuracy So Far: 0.6704000234603882
Total elapsed time: 00h 19m 54s
INFO:tensorflow:Oracle triggered exit
```

همانطور که مشخص است، بهترین دقت با استفاده از Tuner برابر 67 است، پس از اتمام فرایند یادگیری با Tuner یک بار دیگر با استفاده از بهترین نتایج آن شبکه خود را Train کردیم و نتیجه نهایی را دیدیم که به صورت تصویر صفحه بعد شده، اگر تعداد Epochها بیشتر بود، مدل Over Fit می‌شد و برای همین پس از دریافت این دقت آموزش را به اتمام رساندم.

```
# Retrain
Model = Tuner.hypermodel.build(BestHp)
History = Model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))
```

```
Epoch 1/5
1563/1563 [==============================] - 27s 17ms/step - loss: 1.7292 - accuracy: 0.3570 - val_loss: 1.3921 - val_accuracy: 0.5022
Epoch 2/5
1563/1563 [==============================] - 26s 17ms/step - loss: 1.1848 - accuracy: 0.5767 - val_loss: 1.0685 - val_accuracy: 0.6244
Epoch 3/5
1563/1563 [==============================] - 27s 17ms/step - loss: 0.9242 - accuracy: 0.6762 - val_loss: 1.0607 - val_accuracy: 0.6361
Epoch 4/5
1563/1563 [==============================] - 26s 17ms/step - loss: 0.7241 - accuracy: 0.7493 - val_loss: 1.0635 - val_accuracy: 0.6540
Epoch 5/5
1563/1563 [==============================] - 27s 17ms/step - loss: 0.5394 - accuracy: 0.8129 - val_loss: 1.0686 - val_accuracy: 0.6648
```

و در زیر تمام نتایج حاصل از Tuner و مقادیر مناسب آن را با توجه به دقت‌های به دست آمده داریم:

```
Results summary
Results in ./untitled_project
Showing 10 best trials
Objective(name='val_accuracy', direction='max')
Trial summary
Hyperparameters:
filters: 32
num_layers: 4
dropout: False
lr: 0.001
optimizer: adam
filters_0: 32
filters_1: 32
units_0: 224
activation: relu
units_1: 32
filters_2: 32
filters_3: 32
units_2: 32
units_3: 32
Score: 0.6704000234603882
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 conv2d_2 (Conv2D)           (None, 32, 32, 32)        9248

 conv2d_3 (Conv2D)           (None, 32, 32, 32)        9248

 conv2d_4 (Conv2D)           (None, 32, 32, 32)        9248

 average_pooling2d (AverageP  (None, 16, 16, 32)       0
 ooling2D)

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 224)               1835232

 dense_1 (Dense)             (None, 32)                7200

 dense_2 (Dense)             (None, 32)                1056

 dense_3 (Dense)             (None, 32)                1056

 dense_4 (Dense)             (None, 10)                330

=================================================================
Total params: 1,882,762
Trainable params: 1,882,762
Non-trainable params: 0
```

→ hw #9 question 4 (optional) (4

→ Calculate total number of parameters:

layers: → 1: (7×7×1) = 49 + 1 = 50, 50 × 20 = 1000 (Current.) (output: 20×22×22)

→ max pooling: → × 1/2 → 11×11×20 (parameter = 0)

→ 2: (5×5×20) = 500 + 1 = 501, 501 × 10 = 5010 (7×7×20)

→ 3: 3×3×2×5×5 = 450 (7 − 3 + 1 = 5)

Dense → 4: 450 × 10 = 4500 + 450

→ total: 4500 + 450 + 450 + 5010 + 1000 = 11410

5) منابع:

a. https://towardsdatascience.com/step-by-step-implementation-3d-convolutional-neural-network-in-keras-12efbdd7b130
b. https://stackoverflow.com/questions/42883547/intuitive-understanding-of-1d-2d-and-3d-convolutions-in-convolutional-neural-n
c. https://www.kaggle.com/shivamb/3d-convolutions-understanding-use-case
d. https://stackoverflow.com/questions/49003346/why-convolutional-nn-kernel-size-is-often-selected-as-a-square-matrix
e. https://www.sicara.ai/blog/2019-10-31-convolutional-layer-convolution-kernel
f. https://pythonexamples.org/python-opencv-image-filter-convolution-cv2-filter2d/
g. https://www.askpython.com/python-modules/opencv-filter2d
h. https://www.tensorflow.org/tutorials/keras/keras_tuner
i. https://medium.com/swlh/hyperparameter-tuning-in-keras-tensorflow-2-with-keras-tuner-randomsearch-hyperband-3e212647778f
j. https://medium.com/@iamvarman/how-to-calculate-the-number-of-parameters-in-the-cnn-5bd55364d7ca