Tugas Besar 1A IF4074 - Pembelajaran Mesin Lanjut

NIM/Nama : 13517073/Rayza Mahendra | 13517131/Jan Meyer Saragih | 13517137/Vincent Budianto

Nama file : Tubes1A_13517073.ipynb

Topik : CNN - Forward Propragation

Tanggal : 20 February 2020

I. Penjelasan Kode

A. Convolution

Kelas Convolution berisi proses convolution merupakan proses pertama dari convolution layer pada CNN. Pada tahap ini, input akan diextract dalam matriks-matriks. Kelas convolution menerima variabel berupa:

- image
- paddingSize
- filterSizeH
- filterSizeW
- strideSize
- filters

1. padding(self)

Fungsi padding(self) memperbesar ukuran matriks dengan menambahkan 2x ukuran padding pada lebar (kanan dan kiri) dan tinggi (atas dan bawah) matriks input dan mengisinya dengan nilai 0.

2. extract(self)

Fungsi extract(self) menghasilkan semua kemungkinan area gambar (setelah ditambahkan padding) berdasarkan ukuran filter dan ukuran stride.

3. forward(self)

Fungsi forward(self) melakukan penerusan convolution menggunakan input yang diberikan dengan cara mengalikan matriks hasil fungsi extract(self) dengan matriks filter secara element-wise multiplication.

B. Detector

Kelas Detector berisi proses detector memperkenalkan *nonlinearity* ke sistem yang pada dasarnya baru saja menghitung operasi linier pada proses convolution. Kelas detector menerima variabel berupa:

- input
- activation_function
- leaky_slope

1. forward_activation(self, X)

Fungsi forward_activation(self, X) mengaktivasi nilai input X sesuai fungsi aktivasi yang dipakai.

2. activate(self)

Fungsi activate(self) mengubah nilai setiap elemen pada matriks menjadi nilai yang sudah diaktivasi (hasil fungsi forward_activation(self, X)).

C. Pooling

Kelas Pooling berisi proses pooling yang bertujuan untuk mengurangi ukuran output dari proses convoltion sehingga filter dapat mengeksplorasi bagian gambar yang lebih besar untuk menangani kasus *overfitting*. Kelas detector menerima variabel berupa:

- filterWidth
- filterHeight
- stride
- mode

1. __partitionInput(self, inputMatrix, startPosition)

Fungsi __partitionInput(self, inputMatrix, startPosition) mempartisi matriks berdasarkan ukuran filter pooling

2. __maximizeFiltered(self, inputMatrix)

Fungsi __maximizeFiltered(self, inputMatrix) menghitung nilai maksimum untuk setiap patch pada feature map.

3. __averageFiltered(self, inputMatrix)

Fungsi __averageFiltered(self, inputMatrix) menghitung nilai rata-rata untuk setiap patch pada feature map.

4. pool(self, inputMatrix)

Fungsi pool(self, inputMatrix) menghasilkan matriks yang isinya berupa ringkasan dari matriks input.

D. ConvolutionLayer

Kelas ConvolutionLayer merupakan kelas yang merepresentasikan Convolution Layer pada CNN. Kelas ConvolutionLayer menerima variabel berupa:

- convolution
- detector
- pooling
- inputs
- outputs
- inputMapper
- connectionMapper

1. setConfigurationDefault(self, kernelSize)

Fungsi setConfigurationDefault(self, kernelSize) menginisiasi jumlah node convolution, detector dan pooling yang akan dipakai.

2. executeConvolutionLayer(self)

Fungsi executeConvolutionLayer(self, kernelSize) mengeksekusi fungsi forward pada kelas Convolution, fungsi activate pada kelas Detector dan fungsi pool pada kelas Pooling.

E. Dense

Kelas Dense. Kelas Dense menerima variabel berupa:

- weightarray
- activation_function
- leaky_slope
- bias

1. calculateSigma(self, inputArray)

Fungsi calculateSigma(self, inputArray) menghasilkan matriks sigma berdasarkan matriks input, weight dan bias.

2. forward_activation(self, X)

Fungsi forward_activation(self, X) mengaktivasi nilai input X sesuai fungsi aktivasi yang dipakai.

3. activate(self)

Fungsi activate(self) mengubah nilai sigma menjadi nilai yang sudah diaktivasi (hasil fungsi forward_activation(self, X)).

4. get_output(self, inputArray)

Fungsi get_output(self, inputArray) mengembalikan nilai sigma yang sudah diaktivasi berdasarkan inputArray.

F. DenseLayer

Kelas DenseLayer. Kelas DenseLayer menerima variabel berupa:

- flatlength

- nodeCount

1. initiateLayer(self)

`Fungsi initiateLayer(self) menginisiasi jumlah node yang akan dipakai.`

2. executeDenseLayer(self, flatArray)

`Fungsi executeDenseLayer(self, flatArray) mengeksekusi fungsi get_output(self, inputArray) pada kelas Dense.`

G. FlatteningLayer

Kelas FlatteningLayer merupakan kelas pendukung untuk proses flattening.

1. flatten(self, featuremap)

`Fungsi flatten(self, featuremap) mengembalikan input featuremap dalam bentuk satu dimensi.`

H. ConnectionMapper

Kelas ConnectionMapper merupakan kelas pendukung untuk merepresentasikan koneksi pada non-fully connected layer. Kelas ConnectionMapper menerima variabel berupa:

- previousNodeCount
- nextNodeCount
- connectionMap

II. Contoh Hasil Prediksi

Import

In [1]:

```
import sys, os
sys.path.append(os.path.abspath(os.path.join('..', 'src')))
from main import test
```

Test 1

cat.0.jpg

Input size = 200

Convolution Layer:

- Filter count = 2
- Filter size = 3
- Padding size = 2

- Stride size = 1

Pooling Layer:

- Filter size = 3
- Stride size = 1
- Pooling mode = AVG

In [2]:

```
test("../test_data/cats/cat.0.jpg", 200, 2, 3, 2, 1, 3, 1, 'AVG')
```

```
CONVOLUTION LAYER RESULT
[[[ 41.25573384  63.84808749  79.31160073 ... 102.51894884  97.190642
    81.40206818]
  [ 49.18307599  83.17713266 108.52964364 ... 138.25135746 133.10076608
   118.77894317]
  [ 54.43596329  95.60618494 126.54627759 ... 162.5384916  160.9343083
   144.75410705]
  ...
  [ 37.42708956  66.80374671  87.15031502 ...   1.76077036   0.73254556
     1.26288415]
  [ 31.46087559  58.82367427  80.45145544 ...   1.91526671   0.75858392
     1.28892251]
  [ 22.63861661  44.54325147  62.4053709  ...   3.35187175   1.51702028
     2.92617516]]

 [[  9.95112563  13.79668372  16.13066274 ...  19.37803376  18.34232864
    13.6285928 ]
  [  0.           0.           0.         ...   0.           0.
     0.        ]
  [  0.           0.           0.         ...   0.           0.
     0.        ]
  ...
  [  0.           0.           0.         ...   4.13968847   3.73646316
     2.28013306]
  [  0.           0.           0.         ...   3.10746818   2.70424288
     1.66402731]
  [  0.           0.           0.         ...   2.55354127   2.72621932
     1.85403517]]]
(2, 200, 200)

DENSE LAYER RESULT
1.0
```

Test 2

cat.2.jpg

Input size = 100

Convolution Layer:

- Filter count = 3
- Filter size = 2
- Padding size = 1
- Stride size = 1

Pooling Layer:

- Filter size = 2
- Stride size = 1
- Pooling mode = MAX

In [3]:

```
test("../test_data/cats/cat.2.jpg", 100, 3, 2, 1, 1, 2, 1, 'MAX')
```

```
CONVOLUTION LAYER RESULT
[[[ 45.17708695   4.30408065  13.7310616  ...  19.17026078  19.17026078
    14.66501702]
  [ 45.17708695  17.52222106   7.94978472 ...   9.7215814    5.28958216
     0.        ]
  [ 44.88394498  23.17393195   7.94978472 ...   9.7215814    7.70630661
     2.17432215]
  ...
  [ 40.07068196  25.08968743  25.08968743 ...   1.57658555   6.36492854
     6.36492854]
  [ 38.13723852  30.56162529  30.56162529 ...   1.57658555  33.03593696
    33.03593696]
  [ 34.95986343  30.56162529  30.56162529 ...   8.13868654  33.03593696
    33.03593696]]

 [[225.33096931 225.33096931 215.05122097 ... 262.29844098 262.29844098
   252.74443164]
  [322.79540166 359.99354108 359.99354108 ... 220.98055073 216.29800468
   211.65744338]
  [322.79540166 359.99354108 359.99354108 ... 210.66769462 213.91095078
   213.91095078]
  ...
  [171.62135975 265.18253111 303.7171811  ...  94.73539999  94.73539999
    74.36775786]
  [159.08068543 228.55611566 285.96876537 ...  88.76946026 162.05600373
   162.05600373]
  [155.99417998 226.18497456 285.96876537 ...  73.78691177 162.05600373
   162.05600373]]

 [[ 10.66879985   9.08200965   9.08200965 ...   0.40635138   0.40635138
     0.        ]
  [  5.69648933   0.           0.         ...   0.           0.
     0.        ]
  [  0.           0.           0.         ...   0.           0.
     0.        ]
  ...
  [  0.           0.           0.         ...   0.           0.
     0.        ]
  [  0.           0.           0.         ...   0.           0.
     0.        ]
  [  0.           0.           0.         ...   0.           0.
     0.        ]]]
(3, 100, 100)

DENSE LAYER RESULT
1.0
```

Test 3

dog.0.jpg

Input size = 200

Convolution Layer:

- Filter count = 1
- Filter size = 3
- Padding size = 1
- Stride size = 2

Pooling Layer:

- Filter size = 2
- Stride size = 2
- Pooling mode = MAX

In [4]:

```
test("../test_data/dogs/dog.0.jpg", 200, 1, 3, 3, 2, 2, 2, 'MAX')
```

```
CONVOLUTION LAYER RESULT
[[[8.93022712 0.         1.45719346 ... 0.         4.7102763  0.95776745]
  [9.60520683 0.         0.         ... 0.         0.         0.        ]
  [0.         0.         0.         ... 0.         0.         0.        ]
  ...
  [1.21518566 0.         0.         ... 0.         0.         0.        ]
  [4.51424186 0.         0.         ... 0.         0.         0.        ]
  [1.25334689 0.         0.         ... 0.         0.         0.        ]]]
(1, 51, 51)

DENSE LAYER RESULT
7.124576406741285e-218
```

Test 4

dog.3.jpg

Input size = 150

Convolution Layer:

- Filter count = 1
- Filter size = 4
- Padding size = 2
- Stride size = 3

Pooling Layer:

- Filter size = 3
- Stride size = 3
- Pooling mode = AVG

In [6]:

```
test("../test_data/dogs/dog.3.jpg", 150, 1, 4, 2, 3, 3, 3, 'AVG')
```

```
CONVOLUTION LAYER RESULT
[[[54.20515839 69.50348297 69.43191903 55.4833995  28.14148055
   66.34177533 68.77223028 68.03868531 50.36123654 70.09470327
   63.80353194 64.61876488 63.1482455  56.78588991 55.20770474
   54.79215864 48.41020608]
  [52.45402797 78.84112851 81.09568662 76.97784639 40.16054863
   57.67065653 76.02266546 64.97608555 58.28372897 70.96705542
   72.88917534 78.64645548 71.09982318 70.51653139 63.75166871
   62.05107679 58.72834857]
  [52.08991336 79.88774122 74.11832939 46.82122454 43.64955005
   35.42028485 34.80249119 26.98256901 58.47040671 66.73122751
   66.1059884  67.42459256 71.1980741  68.64964298 71.05110988
   69.79096948 59.3752297 ]
  [54.94711393 72.43093807 66.91734346 25.94518189 14.01531427
   23.28891168 18.54469856 19.49155937 60.32490868 76.94179341
   78.62897551 76.83592993 70.56583456 69.80716541 70.48831521
   70.59164942 62.28183752]
  [56.4614289  35.89092767 17.65848098 20.18301728 21.56175772
   20.19941264 17.37782981 18.9994011  68.88327902 68.76818538
   68.86989517 66.37091409 70.29959785 65.44133907 71.85841553
   64.50254675 65.14339683]
  [44.79015259 33.31503902  7.88164456 16.24328214 15.70784468
   25.85654483 22.93800268 18.33775944 66.51388436 70.97512341
   69.02742521 72.76458966 72.00008406 69.55689851 64.80773879
   58.74973283 60.64167563]
  [45.32148898 56.59605221 18.98996269 10.80126986 11.00049803
   40.68913933 35.43454623 30.62101544 54.91620277 62.14668772
   63.73261805 71.67376779 77.41840117 65.86058503 65.51263355
   58.55720817 46.90419752]
  [45.47998442 64.36657529 72.24631445 62.89156193 55.37335564
   46.21635323 42.05190452 40.35740273 46.15251294 23.38946219
   19.55193493 35.97013607 17.39840052 10.55346792 15.22683777
   20.64877257 23.98617762]
  [41.32168528 65.30594773 63.58371419 56.64802264 56.47530136
   49.50728107 52.00151996 43.48088373 34.69860733 14.25743759
    7.45296493 17.39144174  6.18129539 12.75945321 12.93381212
    8.82608567  8.07215044]
  [46.66692826 63.91371884 63.67979015 65.84219595 63.6248479
   54.08028091 53.10912768 36.19310073 18.52565739  9.79542287
    8.10180807  5.72341314 12.81543877  6.92215262  8.83235814
    8.82414809  9.50378702]
  [45.23832603 65.12208781 64.75924214 65.98350104 67.97911318
   59.43549727 49.68630168 29.02620871 17.61635888 10.8574894
    9.74313073  6.47334724 11.30310024 10.53178204  7.19696881
    6.18236903 14.42849649]
  [44.67456466 64.88702436 61.05764404 65.04260168 60.32151835
   62.65373086 54.93259929 37.1635251  24.58159094 13.52660082
   12.8653127   7.89761932 10.01091977  6.86433855  7.75709592
   11.0923152  22.18554523]
  [43.88958819 62.13521109 63.92079217 62.31680175 60.61343179
   65.97273833 57.89142579 44.46082341 25.61061726 17.55457567
    4.83224934 13.00128547 12.72219844 15.49242878 19.75247088
   20.16027594 32.03247813]
  [46.40377807 58.94115362 62.44726226 60.59389638 63.10052467
   61.54409443 59.79085743 43.77090425 27.73463471 21.91369508
   13.47302192 19.36999057 16.77585953 15.64443316 19.07256912
   23.48475278 33.11244572]
```

```
  [42.58133129 63.92157709 58.25495688 63.335853   59.66714654
   61.85170539 60.79645953 45.17808428 25.38678704 27.10010758
   24.45494588 24.32599225 28.0723932  32.0188284  42.49987899
   57.73640175 53.03160616]
  [42.30199042 55.92797825 59.52863405 57.21315294 62.49721343
   65.18834964 62.15039014 43.90873444 32.58165572 38.55773934
   44.16036453 58.3555039  63.78731721 59.17503148 61.30564264
   60.69842615 57.12698164]
  [31.42658391 43.548008   44.82807991 46.38516706 48.84624806
   45.37348893 44.89527022 37.7080072  27.2457213  32.3170272
   45.72720839 46.05330096 43.5593177  45.69294458 46.74965487
   45.67290052 43.55872675]]]
(1, 17, 17)

DENSE LAYER RESULT
1.0
```

III. Pembagian Tugas

NIM	Nama	Tugas
13517073	Rayza Mahendra	Detector, Dense Layer, Extract
13517131	Jan Meyer Saragih	Pooling, Convolution Layer
13517137	Vincent Budianto	Convolution, Laporan