Tugas Besar 1B IF4074 - Pembelajaran Mesin Lanjut

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

Nama file : Tubes1B_13517073.ipynb

Topik : CNN - Forward Propragation

Tanggal : 09 Oktober 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:

- batchsize
- batchperepoch
- 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.`

4. back_propagation(self, delta_matrix)

`Fungsi back_propagation(self, delta_matrix) mengembalikan delta_filter dari hasil back propagation kelas detector.`

5. updateFilters(self, learning_rate, momentum)

`Fungsi updateFilters(self, learning_rate, momentum) melakukan update nilai filter setelah satu epoch dijalankan berdasarkan nilai learning_rate dan momentum.`

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)).`

3. back_propagation(self, delta_matrix)

`Fungsi back_propagation(self, delta_matrix) mengembalikan delta_matrix dari hasil back propagation kelas pooling.`

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.`

5. back_propagation(self, delta_matrix)

`Fungsi back_propagation(self, delta_matrix) mengembalikan delta_matrix dari hasil back propagation Dense Layer yang sudah di unflatten.`

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.`

3. backward_node(self, delta_matrix, convolution, detector, pooling)

`Fungsi backward_node(self, delta_matrix, convolution, detector, pooling) mengeksekusi fungsi back_propagation pada kelas Convolution, kelas Detector, dan kelas Pooling.`

4. backward_propagation(self, delta_matrix)

`Fungsi backward_propagation(self, delta_matrix) mengeksekusi fungsi backward_node jika delta_matrix berupa matrix tiga dimensi.`

5. updateWeight(self, learning_rate, momentum)

`Fungsi updateWeight(self, learning_rate, momentum) melakukan update nilai weight Convolution Layer setelah satu epoch dijalankan berdasarkan nilai learning_rate dan momentum.`

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
- batchsize
- batchperepoch
- activation_function
- 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.`

3. calcBackwards(self, d_succ, weight_succ)

`Fungsi calcBackwards(self, d_succ, weight_succ) mengembalikan delta_matrix dari hasil back propagation Output Layer.`

4. updateWeight(self, learningrate, momentum)

`Fungsi updateWeight(self, learningrate) melakukan update nilai weight Dense Layer setelah satu epoch dijalankan berdasarkan nilai learningrate dan momentum.`

G. OutputLayer

Kelas OutputLayer. Kelas OutputLayer menerima variabel berupa:

- flatlength
- batchsize
- batchperepoch
- 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.`

3. computeError(self, label)

`Fungsi computeError(self, label) mengkomputasi nilai error pada Output Layer`

4. calcBackwards(self, d_succ, weight_succ)

`Fungsi calcBackwards(self, d_succ, weight_succ) mengembalikan delta_matrix dari hasil back propagation Output Layer.`

5. updateWeight(self, learningrate, momentum)

`Fungsi updateWeight(self, learningrate) melakukan update nilai weight Dense Layer setelah satu epoch dijalankan berdasarkan nilai learningrate dan momentum.`

H. FlatteningLayer

Kelas FlatteningLayer merupakan kelas pendukung untuk proses flattening.

1. flatten(self, featuremap)

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

2. unflatten(self, featuremap)

`Fungsi unflatten(self, featuremap) mengembalikan input featuremap dalam bentuk dimensi awal.`

3. calcBackwards(self, d_succ, weight_succ)

`Fungsi calcBackwards(self, d_succ, weight_succ) mengembalikan delta_matrix hasil fungsi unflatten untuk dikembalikan ke kelas pooling.`

I. Network

Kelas Network merupakan kelas pendukung untuk menyatukan kelas Convolution, kelas Detector, kelas Pooling, kelas DenseLayer dan kelas OutputLayer.

1. initiate_network(self, batchsize, batchperepoch, convInputSize, convFilterCount, convFilterSize, convPaddingSize, convStrideSize, detectorMode, poolFilterSize, poolStrideSize, poolMode, activation_dense="relu")

`Fungsi initiate_network(self, batchsize, batchperepoch, convInputSize, convFilterCount, convFilterSize, convPaddingSize, convStrideSize, detectorMode, poolFilterSize, poolStrideSize, poolMode, activation_dense="relu") menginisiasi jumlah batchsize, batchperepoch, convInputSize, convFilterCount, convFilterSize, convPaddingSize, convStrideSize, detectorMode, poolFilterSize, poolStrideSize, poolMode dan activation_dense (default menggunakan relu) yang akan dipakai.`

2. train_one(self, fileName, label)

`Fungsi train_one(self, fileName, label) mengeksekusi melakukan training untuk satu gambar.`

3. update_weight(self, learning_rate, momentum)

`Fungsi update_weight(self, learning_rate, momentum) melakukan update nilai weight model setelah satu epoch dijalankan berdasarkan nilai learningrate dan momentum.`

4. predict(self, fileName)

`Fungsi predict(self, fileName) melakukan prediksi menggunakan validation data pada model.`

5. check_predict(self, fileName, label)

`Fungsi check_predict(self, fileName, label) menilai apakah hasil dari fungsi predict benar atau salah.`

6. train(self, directory, label, epoch, learning_rate, momentum, val_data, train_data)

`Fungsi train(self, directory, label, epoch, learning_rate, momentum, val_data, train_data) menjalankan pelatihan model menggunakan training data dan prediksi menggunakan validation data`

J. Fungsi Lain

1. kfoldxvalidation(Network, directory, label, epoch, learning_rate, kfold, momentum)

`Fungsi kfoldxvalidation(Network, directory, label, epoch, learning_rate, kfold, momentum) melakukan training model dengan skema 10-fold cross validation`

2. mass_predict(network, iteration)

`Fungsi mass_predict(network, iteration) menghasilkan confusion matrix yang akan ditampilkan pada akhir proses testing model`

3. savemodel(network, iteration)

`Fungsi savemodel(network, iteration) menyimpan model yang sudah dilatih`

4. loadmodel(filename)

`Fungsi loadmodel(filename) memuat model yang sudah disimpan sebelumnya`

II. Hasil Eksperimen

Training Process

In [ ]:

```
1 Fold
Epoch  1/10 | Training Accuracy: 0.4705 | Validation Accuracy:0.5789
Epoch  2/10 | Training Accuracy: 0.4117 | Validation Accuracy:0.4210
Epoch  3/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.4736
Epoch  4/10 | Training Accuracy: 0.3125 | Validation Accuracy:0.5789
Epoch  5/10 | Training Accuracy: 0.2500 | Validation Accuracy:0.4210
Epoch  6/10 | Training Accuracy: 0.1250 | Validation Accuracy:0.5789
Epoch  7/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.4210
Epoch  8/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.4210
Epoch  9/10 | Training Accuracy: 0.2500 | Validation Accuracy:0.5789
Epoch 10/10 | Training Accuracy: 0.6250 | Validation Accuracy:0.4736
Overall Accuracy: 0.4012

2 Fold
Epoch  1/10 | Training Accuracy: 0.2941 | Validation Accuracy:0.5000
Epoch  2/10 | Training Accuracy: 0.3529 | Validation Accuracy:0.5000
Epoch  3/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.3888
Epoch  4/10 | Training Accuracy: 0.3750 | Validation Accuracy:0.5000
Epoch  5/10 | Training Accuracy: 0.2500 | Validation Accuracy:0.5000
Epoch  6/10 | Training Accuracy: 0.1250 | Validation Accuracy:0.5000
Epoch  7/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.5000
Epoch  8/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.4444
Epoch  9/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.5000
Epoch 10/10 | Training Accuracy: 0.3750 | Validation Accuracy:0.4444
Overall Accuracy: 0.3742

3 Fold
Epoch  1/10 | Training Accuracy: 0.7058 | Validation Accuracy:0.7222
Epoch  2/10 | Training Accuracy: 0.4117 | Validation Accuracy:0.3888
Epoch  3/10 | Training Accuracy: 0.4117 | Validation Accuracy:0.2777
Epoch  4/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.3333
Epoch  5/10 | Training Accuracy: 0.6250 | Validation Accuracy:0.4444
Epoch  6/10 | Training Accuracy: 0.2500 | Validation Accuracy:0.3888
Epoch  7/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.6111
```

```
Epoch  8/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.4444
Epoch  9/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.3888
Epoch 10/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.5555
Overall Accuracy: 0.4969

4 Fold
Epoch  1/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.5000
Epoch  2/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.5555
Epoch  3/10 | Training Accuracy: 0.4705 | Validation Accuracy:0.4444
Epoch  4/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.4444
Epoch  5/10 | Training Accuracy: 0.5000 | Validation Accuracy:0.5000
Epoch  6/10 | Training Accuracy: 0.2500 | Validation Accuracy:0.4444
Epoch  7/10 | Training Accuracy: 0.3750 | Validation Accuracy:0.5555
Epoch  8/10 | Training Accuracy: 0.5000 | Validation Accuracy:0.4444
Epoch  9/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.5000
Epoch 10/10 | Training Accuracy: 0.5000 | Validation Accuracy:0.5555
Overall Accuracy: 0.4539

5 Fold
Epoch  1/10 | Training Accuracy: 0.3529 | Validation Accuracy:0.5000
Epoch  2/10 | Training Accuracy: 0.5882 | Validation Accuracy:0.3888
Epoch  3/10 | Training Accuracy: 0.4705 | Validation Accuracy:0.5555
Epoch  4/10 | Training Accuracy: 0.6875 | Validation Accuracy:0.5555
Epoch  5/10 | Training Accuracy: 0.6250 | Validation Accuracy:0.4444
Epoch  6/10 | Training Accuracy: 0.5000 | Validation Accuracy:0.5000
Epoch  7/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.5000
Epoch  8/10 | Training Accuracy: 0.5000 | Validation Accuracy:0.4444
Epoch  9/10 | Training Accuracy: 0.3125 | Validation Accuracy:0.6666
Epoch 10/10 | Training Accuracy: 0.6250 | Validation Accuracy:0.5000
Overall Accuracy: 0.5092

6 Fold
Epoch  1/10 | Training Accuracy: 0.4117 | Validation Accuracy:0.5555
Epoch  2/10 | Training Accuracy: 0.5882 | Validation Accuracy:0.5555
Epoch  3/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.4444
Epoch  4/10 | Training Accuracy: 0.6875 | Validation Accuracy:0.4444
Epoch  5/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.6666
Epoch  6/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.6666
Epoch  7/10 | Training Accuracy: 0.6250 | Validation Accuracy:0.4444
Epoch  8/10 | Training Accuracy: 0.6250 | Validation Accuracy:0.5555
Epoch  9/10 | Training Accuracy: 0.6875 | Validation Accuracy:0.5000
Epoch 10/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.4444
Overall Accuracy: 0.5582

7 Fold
Epoch  1/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.6666
Epoch  2/10 | Training Accuracy: 0.5882 | Validation Accuracy:0.6666
Epoch  3/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.3888
Epoch  4/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.3888
Epoch  5/10 | Training Accuracy: 0.3750 | Validation Accuracy:0.6111
Epoch  6/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.3888
Epoch  7/10 | Training Accuracy: 0.6250 | Validation Accuracy:0.3888
Epoch  8/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.3888
Epoch  9/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.6111
Epoch 10/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.3888
Overall Accuracy: 0.5337

8 Fold
Epoch  1/10 | Training Accuracy: 0.6470 | Validation Accuracy:0.5555
Epoch  2/10 | Training Accuracy: 0.4705 | Validation Accuracy:0.4444
Epoch  3/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.6111
Epoch  4/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.7222
Epoch  5/10 | Training Accuracy: 0.3750 | Validation Accuracy:0.4444
Epoch  6/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.5000
Epoch  7/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.6111
```

```
Epoch  8/10 | Training Accuracy: 0.5000 | Validation Accuracy:0.5000
Epoch  9/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.3333
Epoch 10/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.6111
Overall Accuracy: 0.5092

9 Fold
Epoch  1/10 | Training Accuracy: 0.5882 | Validation Accuracy:0.5000
Epoch  2/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.5000
Epoch  3/10 | Training Accuracy: 0.4705 | Validation Accuracy:0.5000
Epoch  4/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.5555
Epoch  5/10 | Training Accuracy: 0.3750 | Validation Accuracy:0.5000
Epoch  6/10 | Training Accuracy: 0.3750 | Validation Accuracy:0.4444
Epoch  7/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.4444
Epoch  8/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.5555
Epoch  9/10 | Training Accuracy: 0.6250 | Validation Accuracy:0.3888
Epoch 10/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.5000
Overall Accuracy: 0.4969

10 Fold
Epoch  1/10 | Training Accuracy: 0.4705 | Validation Accuracy:0.5000
Epoch  2/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.5000
Epoch  3/10 | Training Accuracy: 0.5294 | Validation Accuracy:0.5000
Epoch  4/10 | Training Accuracy: 0.6250 | Validation Accuracy:0.5000
Epoch  5/10 | Training Accuracy: 0.2500 | Validation Accuracy:0.5000
Epoch  6/10 | Training Accuracy: 0.5625 | Validation Accuracy:0.5000
Epoch  7/10 | Training Accuracy: 0.5000 | Validation Accuracy:0.5000
Epoch  8/10 | Training Accuracy: 0.4375 | Validation Accuracy:0.4444
Epoch  9/10 | Training Accuracy: 0.6875 | Validation Accuracy:0.5000
Epoch 10/10 | Training Accuracy: 0.5000 | Validation Accuracy:0.5000
Overall Accuracy: 0.5092

Accuracy list:
[
    [0.4012, <network.Network object at 0x000001D6665973D0>],
    [0.3742, <network.Network object at 0x000001D666654190>],
    [0.4969, <network.Network object at 0x000001D666654310>],
    [0.4539, <network.Network object at 0x000001D6666542E0>],
    [0.5092, <network.Network object at 0x000001D666654C70>],
    [0.5582, <network.Network object at 0x000001D6666545E0>],
    [0.5337, <network.Network object at 0x000001D666654B80>],
    [0.5092, <network.Network object at 0x000001D666654790>],
    [0.4969, <network.Network object at 0x000001D6666546A0>],
    [0.5092, <network.Network object at 0x000001D666654B50>]
]
```

Model Result

In [ ]:

```
Accuracy = 0.425
Confusion Matrix:
      | cat | dog |
 true |  16 |  19 |
false |   4 |   1 |
```

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