Tugas Besar 1

IF4074 Advanced Machine Learning

CNN - Backward Propagation

oleh :

Syihabuddin Yahya Muhammad	/ 13519149
Imam Nurul Hukmi	/ 13519150
Muhammad Furqon	/ 13519184

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2022

1. Penjelasan Kode

1.1. Kelas CNN Backpropagation

Kelas CNN merupakan kelas utama dari implementasi arsitektur CNN. untuk menggunakan model CNN, cukup kelas ini saja yang digunakan. Untuk menggunakan fitur *backpropagation,* cukup panggil fungsi *backpropagation*.

1.2. Kelas Convolution Layer Backpropagation

Kelas Convolution Layer berfungsi untuk melakukan konvolusi terhadap *matriks input*. Pada tahapan ini didefinisikan sejumlah input yang akan dioperasikan dengan matriks kernel *filter*. *Weight* dan bias diinisialisasi secara acak saat dilakukan inisialisasi. Output yang dihasilkan sebanyak jumlah *filter.*

Pada proses Backpropagation, Convolution Layer menerima masukan berupa hasil proses Backpropagation layer sebelumnya. Delta bobot didapat dari hasil perkalian dengan matriks layer sebelumnya.
Hasil dari Backpropagation digunakan untuk melakukan update pada matriks bobot dan bias. Update dilakukan dengan learning rate μ dan momentum m. Rumus diuraikan pada gambar di bawah ini.

$$\Delta w_{ij}(t) = \mu_i \delta_i y_j + m \Delta w_{ij}(t-1)$$

1.3. Kelas Detector Layer Backpropagation

Kelas Detector Layer memiliki fungsi yang sederhana, yaitu menerapkan fungsi RELU pada setiap elemen pada matriks input. Sehingga, kelas ini tidak membutuhkan argumen apapun ketika diinisialisasi.

Pada proses Backpropagation, Detector Layer menerima masukan berupa hasil proses Backpropagation layer sebelumnya. Lalu, masukan tersebut akan dikenai operasi perkalian antar-elemen dengan turunan dari operasi pada Detector Layer tersebut. Pada tugas ini, operasi pada Detector Layer merupakan RelU, sehingga hasil turunannya menjadi:

$$Relu'(x_{ij}) = \begin{cases} 1; & x_{ij} > 0 \\ 0; & else \end{cases}$$

1.4. Kelas Pooling Layer Backpropagation

Kelas Pooling Layer berfungsi untuk memperkecil ukuran matriks input sehingga layer selanjutnya dapat memproses matriks dengan lebih efisien. Kelas Pooling Layer bekerja dengan cara memetakan elemen-elemen yang saling bertetangga dan menerapkan suatu fungsi, seperti mencari maksimal atau menghitung rata-rata, kepada kumpulan elemen tersebut sehingga menghasilkan suatu nilai baru. Hal ini dilakukan agar matriks dapat diperkecil dengan tetap mempertahankan sebagian fitur penting yang ada pada matriks input.

Pada proses Backpropagation, Pooling Layer akan meneruskan hasil operasi reverse-pooling antara hasil Backpropagation layer sebelumnya dengan turunan dari operasi pooling. Pada tugas ini, ada dua jenis pooling, yaitu Max Pooling yang memilih nilai maksimal sebagai keluaran dan Average Pooling yang merata-ratakan jumlah dari semua elemen.
Untuk Max Pooling, operasi turunannya adalah sebagai berikut:

$$MaxPool'(x_{ij}) = \begin{cases} 1; x_{ij} = \max(x_{00} \dots x_{ij}) \\ 0; else \end{cases}$$

Sedangkan, untuk Average Pooling, operasi turunannya adalah sebagai berikut:

$$AveragePool'(x_{ij}) = \frac{1}{count(x_{00} \dots x_{ij})}$$

1.5. Kelas Dense & FFNN Backpropagation

Kelas FFNN merupakan sebuah implementasi arsitektur *Feed Forward Neural Network.* Kelas ini diambil dari hasil tubes mata kuliah *Machine Learning* tanpa modifikasi sama sekali. Sedangkan kelas Dense merupakan wrap class dari FFNN sebagai interface untuk digunakan oleh CNN. dalam implementasi Backpropagation tantangan utama dari kelas ini adalah meneruskan data yang dimiliki oleh layer CNN lainnya kepada FFNN. karena kelas FFNN telah dibuat di mata kuliah Pembelajaran mesin dan memiliki input/output yang berbeda. Khususnya ketika memanggil backpropagation maka yang dieksekusi adalah semua step dari forward propagation hingga mengubah weight.

1.6. Fungsi Utility

Pada implementasi backpropagation ini, ditambahkan sebuah fungsi util baru yang berfungsi untuk membaca seluruh image yang berada pada subfolder sebuah folder yang diberikan label nama dari subfolder tersebut. Fungsi ini digunakan untuk menggantikan tf.keras.utils.image_dataset_from_directory dalam pengerjaan tubes ini.

1.7. Lain-Lain

Pada kode juga terdapat beberapa file yang tidak dituliskan disini. File tersebut merupakan file yang digunakan oleh FFNN. karena keterbatasan desain(tidak mengubah FFNN dan komponennya) dan bahasa(python) seluruh kode harus disimpan di file yang sama.

2. Hasil Eksperimen

Kode program dapat diakses di https://github.com/LastAeon/tubes-ML-lanjutan-CNN

2.1. Input

2.1.1. Input Batch

Digunakan pembagian skema 90% untuk training dan 10% testing untuk validasi dataset tersebut. Pelatihan dengan skema ini dapat diakses dengan nama method "train_90_10". Argumen yang dibutuhkan adalah dataset, jumlah epoch, besar *learning rate*, dan besar momentum

2.1.2. 10 Fold Cross Validation

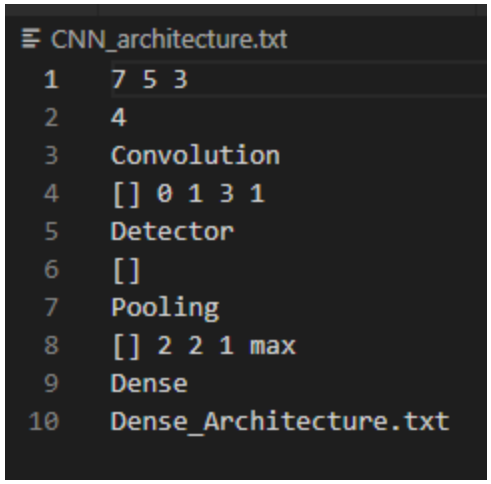
Digunakan skema K-fold Cross Validation dengan jumlah fold sebanyak 10. Pada setiap epoch digunakan fold yang berkesesuaian sebagai training dan validasi dataset tersebut. Validasi model ini dapat diakses sebagai sebuah fungsi dengan nama "read_testing_arch" pada file "test_arch.py". Parameter yang dibutuhkan hanya satu, yaitu alamat ke file format pelatihan. Isi dari file tersebut adalah:

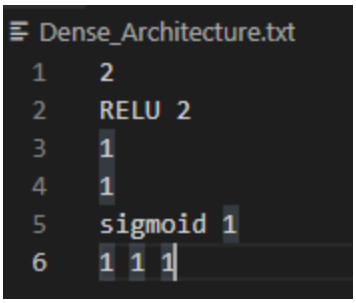
testing_architecture.txt

```
C:\My\Kuliah\AML\Tubes1\Milestone1\tubes-ML-lanjutan-CNN\train  <- direktori dataset train
C:\My\Kuliah\AML\Tubes1\Milestone1\tubes-ML-lanjutan-CNN\test   <- direktori dataset test
CNN_architecture.txt                                           <- arsitektur cnn
10 .1 0                                                        <- epoch, learning_rate, momentum
```

2.2. Architecture

Dikarenakan implementasi FFNN untuk dense layer menggunakan proyek sebelumnya yang sudah ada, maka file arsitektur dibagi menjadi 2 yaitu arsitektur untuk CNN dan arsitektur untuk Dense/FFNN. Berikut arsitektur yang digunakan untuk melakukan testing.



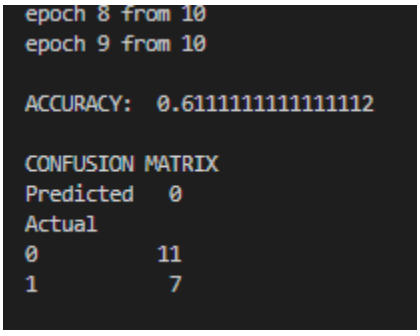


Pada arsitektur CNN, baris pertama merupakan ukuran resize gambar. Untuk arsitektur di atas, gambar yang diterima akan diubah menjadi array of matrix RGB yang kemudian di resize menjadi ukuran 3*7. Kemudian baris berikutnya adalah jumlah layer. Untuk dense layer dianggap 1 layer saja karena penjelasan detailnya dilakukan di file berbeda. Baris berikutnya merupakan 2 baris berupa jenis layer dan parameternya yang diulang sebanyak jumlah layer kali. Untuk konfigurasi arsitektur dense layer dapat dilihat disini. Konfigurasi bobot dari layer pertama Dense architecture dapat diabaikan(isi bebas) karena akan di *override* oleh bentukan dari CNN yang seharusnya.

2.3. Output

2.3.1. Input Batch

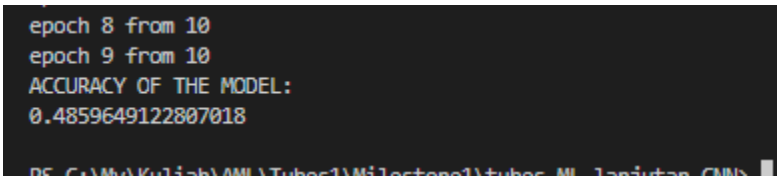
Digunakan pembagian skema 90% untuk training dan 10% testing untuk validasi dataset tersebut. Pada pengujian ini, dimensi gambar diperkecil menjadi 50 x 50 agar proses training tidak memakan waktu yang terlalu lama.



2.3.2. 10 Fold Cross Validation

Digunakan skema Kfold Cross Validation dengan jumlah fold sebanyak 10. Pada setiap epoch digunakan fold yang berkesuaian sebagai training dan validasi dataset tersebut. Pada pengujian ini, dimensi gambar sangat

diperkecil menjadi 10 x 10 karena apabila digunakan ukuran yang semula, waktu pelatihan model akan memakan waktu yang lama (pelatihan 1 epoch memakan waktu 1,5 menit pada perangkat lokal).



2.3.3. Save Load Model

Pada model yang sudah dilakukan training dapat disimpan ke dalam file eksternal dengan melakukan *save* ke format .pickle dan dapat digunakan dengan melakukan *load*.

3. Pembagian Tugas

NIM	Nama	Tugas yang Dikerjakan
13519149	Syihabuddin Yahya Muhammad	Arsitektur dan Derivatif Dense Layer
13519150	Imam Nurul Hukmi	Derivatif Detector dan Pooling Layer
13519184	Muhammad Furqon	Derivatif Relu dan KFold Cross Validation