Laporan Tugas Besar Convolutional Neural Network (Milestone A)

IF4074 Pembelajaran Mesin Lanjut

Forward Propagation

Disusun oleh:

Mahesa Lizardy	13520116
Bryan Amirul Husna	13520146

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

2023

Daftar Isi

Pendahuluan

A. CNN

Convolutional Neural Network (CNN) adalah jenis arsitektur jaringan saraf tiruan yang secara khusus dirancang untuk memproses data grid, seperti gambar dan video. CNN telah menjadi bagian integral dari berbagai aplikasi dalam pengolahan citra, pengenalan pola, visi komputer, dan banyak lagi.

Keunggulan utama CNN terletak pada kemampuannya untuk secara otomatis mengekstraksi fitur-fitur penting dari data grid ini, yang memungkinkan model untuk memahami hierarki pola dan struktur dalam data visual. Selain itu, CNN dapat menangkap fitur-fitur spasial yang membuatnya sangat efektif dalam memproses informasi berdimensi tinggi. Selain itu, CNN juga dikenal dapat meningkatkan kemampuan generalisasi dan kecepatan dalam pemrosesan data. Berikut adalah beberapa komponen utama dalam arsitektur CNN:

1. *Convolution Layer*

   Layer ini adalah inti dari CNN. Mereka menggunakan operasi konvolusi untuk menerapkan filter atau kernel pada input gambar. Ini membantu dalam mengekstraksi fitur-fitur seperti tepi, garis, dan pola visual lainnya.

2. *Detector Layer*

   Layer ini digunakan untuk mengurangi dimensi spasial dari representasi yang dihasilkan oleh layer konvolusi. Max-pooling adalah jenis pooling yang paling umum, yang memilih nilai maksimum dari segmen-segmen data dalam grid. Juga terdapat Average poling yang menghitung rata–rata dari segmen-segmen data dalam grid.

3. *Pooling Layer*

   Sebelum memasuki lapisan fully connected, representasi dari lapisan-lapisan konvolusi dan pooling harus diubah menjadi vektor satu dimensi.

CNN telah membawa revolusi dalam bidang pengolahan citra dan visi komputer, serta digunakan dalam berbagai aplikasi seperti pengenalan wajah, deteksi objek, pengenalan tulisan tangan, dan banyak lagi. Mereka terus berkembang dengan peningkatan dalam arsitektur dan teknik pelatihan.

B. Forward Propagation

Forward propagation adalah salah satu langkah kunci dalam proses pelatihan dan penggunaan *Artificial Neural Network* (ANN), termasuk jugas *Convolutional Neural Network* (CNN). Forward propagation adalah proses di mana data input disebarkan melalui jaringan neural network untuk menghasilkan prediksi atau output yang digunakan dalam tugas tertentu.

Dalam CNN atau jaringan saraf lainnya, forward propagation adalah langkah pertama dalam memproses data. Ini melibatkan aliran data melalui berbagai lapisan jaringan, termasuk lapisan konvolusi, pooling, dan dense layers. Selama proses ini, data input mengalami serangkaian transformasi matematika, yang melibatkan penggabungan, konvolusi, penjumlahan bobot, dan fungsi aktivasi, sehingga mewakili informasi secara hierarkis dan abstrak. Output dari langkah forward propagation ini adalah prediksi atau representasi dari data input yang dapat digunakan untuk tugas seperti klasifikasi, deteksi objek, atau regresi.

Implementasi

A. Convolutional Layer

Pada bagian ini merupakan implementasi dari Convolution, Detector, dan Pooling.

1. Convolution

   Implementasi Convolution terdapat pada class Convolution. Class ini memiliki beberapa fungsi utama sebagai berikut.

   a. __init__(self, input_size, padding_size, filter_size, num_filters, stride)

   Fungsi ini merupakan konstruktor kelas Convolution. Ini digunakan untuk menginisialisasi berbagai parameter yang diperlukan untuk lapisan konvolusi. Parameter pada inisialisasi kelas convolution adalah sebagai berikut

Parameter	**Description**
input_size	Ukuran input (panjang, lebar, jumlah saluran).
padding_size	Ukuran padding (jumlah sel yang ditambahkan di sekitar input).
filter_size	Ukuran filter konvolusi (panjang dan lebar).
num_filters	Jumlah filter yang akan digunakan.
stride	Jarak (langkah) antara filter yang digunakan saat menggeser melalui input.

   Selain inisialisasi parameter tersebut akan dilakukan juga inisialisasi ukuran output, inisialisasi filter dengan random, serta inisialisasi bias dengan nol.

   b. forward(self, input)

   Ini adalah fungsi untuk melakukan operasi konvolusi pada input yang diberikan. Pertama, jika padding diperlukan (jika padding_size lebih dari nol), maka input akan dipad dengan nilai nol. Kemudian, loop dilakukan melalui seluruh input dengan langkah sejauh stride untuk melakukan operasi konvolusi pada input dengan filter yang telah diinisialisasi. Hasil dari operasi konvolusi disimpan dalam matriks output. Hasil akhir adalah matriks output yang berisi hasil konvolusi dari input dengan filter-filter yang sesuai.

   c. getModel(self)

   Fungsi ini mengembalikan model konvolusi dalam bentuk dictionary. Model ini berisi kernel filter dan bias dalam bentuk list. Fungsi ini digunakan untuk menyimpan model konvolusi ke dalam bentuk json.

d. setFilter(self, filter)

Fungsi ini memungkinkan Anda untuk mengatur filter konvolusi dengan filter yang diberikan sebagai parameter.

e. showModel(self):

fungsi yang digunakan untuk menampilkan informasi tentang lapisan konvolusi pada terminal

2. Detector

Implementasi Detector terdapat pada class Convolution yaitu pada fungsi forward. Hasil dari operasi konvolusi sebelum disimpan ke matriks output dilakukan ungsi aktivasi ReLU diaplikasikan. Langkah ini dilakukan untuk mengurangi beban komputasi yang timbul jika lapisan Detector dijalankan secara terpisah.

3. Pooling

Implementasi Pooling terdapat pada class Pooling. Class ini memiliki beberapa fungsi utama sebagai berikut.

a. __init__(self, filter_size, stride, mode)

Konstruktor kelas ini digunakan untuk menginisialisasi parameter operasi pooling, termasuk ukuran filter pooling, langkah (stride), dan mode (max atau average).

b. forward(self, input)

Metode forward digunakan untuk melakukan operasi *polling* pada input yang diberikan. Ini mengambil input dan menghasilkan output berdasarkan parameter yang diatur pada saat inisialisasi. Pertama, itu menghitung ukuran output berdasarkan ukuran input, ukuran filter, dan langkah yang telah diatur. Kemudian, itu membuat matriks output dengan ukuran yang sesuai. Loop dilakukan melalui input dengan langkah sejauh stride, dan untuk setiap jendela filter di dalam input, itu melakukan operasi pooling sesuai dengan mode yang telah ditentukan (max atau average). Hasil *polling* disimpan dalam matriks output dan direturn.

c. getModel(self)

Fungsi ini mengembalikan deskripsi model pooling dalam bentuk dictionary. Fungsi ini digunakan untuk menyimpan model konvolusi ke dalam bentuk json.

d. showModel(self)

Fungsi ini digunakan untuk menampilkan informasi tentang lapisan pooling.

B. Dense

Implementasi Dense terdapat pada class Dense. Class ini memiliki beberapa fungsi utama sebagai berikut.

a. __init__(self, num_units, activation_function, input_size = None)

Fungsi ini merupakan konstruktor kelas Dense. Fungsi ini akan menginisialisasi objek lapisan dengan parameter seperti jumlah unit (neuron), fungsi aktivasi yang digunakan, dan ukuran input jika sudah diketahui. Ini juga menginisialisasi bobot (weights) dan bias dengan nilai-nilai acak.

b. forward(self, input_data)

Fungsi ini akan melakukan operasi forward propagation. Fungsi akan mengambil input data dan mengalokasikan bobot dan bias jika belum diinisialisasi sebelumnya. Kemudian, melakukan perkalian matriks antara input_data dan bobot, menambahkan bias, dan menerapkan fungsi aktivasi yang dipilih (ReLU atau sigmoid) dan mereturnnya.

c. relu(self, x)

Metode ini menerapkan fungsi aktivasi ReLU (Rectified Linear Unit) pada input x. ReLU mengembalikan nilai maksimum antara 0 dan input.

d. sigmoid(self, x)

Metode ini menerapkan fungsi aktivasi sigmoid pada input x, yang menghasilkan nilai antara 0 dan 1.

e. getModel(self)

Metode ini mengembalikan representasi model dalam bentuk kamus (dictionary) yang mencakup tipe lapisan (dense), kernel (bobot), bias, dan jenis fungsi aktivasi dalam bentuk json.

f. setWeights(self, weights)

Metode ini memungkinkan pengguna untuk mengatur bobot lapisan dengan bobot yang diberikan sebagai argumen.

g. setBias(self, bias)

Metode ini memungkinkan pengguna untuk mengatur bias lapisan dengan nilai bias yang diberikan sebagai argumen.

h. showModel(self, input=1)

Metode ini mencetak tampilan sederhana untuk lapisan Dense, menampilkan jumlah unit dalam lapisan, serta jumlah parameter (bobot dan bias) yang ada dalam lapisan ini.

C. Flatten

Implementasi Flatten terdapat pada class Flatten. Class ini memiliki beberapa fungsi utama sebagai berikut.

a. __init__(self)

Ini adalah konstruktor kelas Flatten. Pada konstruktor Ini tidak melakukan apa-apa selain menginisialisasi objek.

b. forward(self, input_data)

Metode ini digunakan untuk melakukan operasi forward propagation. Ia mengambil data input yang berupa dimensi berapapun kemudian mengubahnya menjadi satu dimensi dengan cara "melipat" atau "flattening". Ini berarti semua elemen dari input awal akan disusun secara berurutan menjadi satu vektor satu dimensi. Hasil vektor ini adalah representasi data yang akan diteruskan ke lapisan Dense.

c. getModel(self)

Metode ini mengembalikan representasi model dalam bentuk kamus (dictionary) yang mencakup tipe lapisan (flatten) dalam bentuk jsonr.

d. showModel(self, input=1)

Metode ini mencetak tampilan sederhana untuk lapisan Flatten.

D. Model

Implementasi Model terdapat pada class Model. Class ini memiliki beberapa fungsi utama sebagai berikut.

a. __init__(self)

Ini adalah konstruktor kelas Model. Ini menginisialisasi objek dengan sebuah daftar (list) yang akan digunakan untuk menyimpan layer-layer dalam jaringan.

b. predict(self, X)

Metode ini digunakan untuk melakukan prediksi. Fungsi menerima X yang merupakan daftar nilai x yang ingin diprediksi, dan menghasilkan suatu daftar nilai prediksi yang bersesuaian dengan tiap x.

c. train_network(self, train_data, label_data, batch_size, lr=0.01, epochs=200)

Metode ini digunakan untuk melatih artificial neural network. Fungsi ini akan mengambil data pelatihan (data train dan data label), ukuran batch (batch_size), learning rate (lr), dan jumlah epoch (epochs) sebagai argumen. Selama pelatihan, ia membagi data pelatihan ke dalam batch sesuai dengan batch_size, dan kemudian melakukan proses forward propagation

untuk setiap sampel dalam batch menggunakan lapisan-lapisan yang telah ditambahkan ke jaringan. Ini juga menghitung akurasi selama pelatihan dan mencetaknya. Ini mengulangi proses pelatihan sebanyak jumlah epochs.

d. add(self, layer)

Metode ini digunakan untuk menambahkan layer baru ke dalam jaringan. Layer-layer ini harus telah diinisialisasi sebelumnya.

e. saveModel(self)

Metode ini digunakan untuk menyimpan model jaringan ke dalam file JSON. Ini menciptakan representasi model sebagai daftar kamus (dictionary) dari setiap lapisan menggunakan metode getModel() dari setiap lapisan dan menyimpannya dalam format JSON.

f. showModel(self)

Metode ini mencetak tampilan struktur model jaringan dengan mencetak informasi tentang setiap lapisan, termasuk tipe lapisan, bentuk keluaran (output shape), dan jumlah parameter yang digunakan.

Pengujian

A. Rancangan Pengujian

Pengujian dilakukan pada Panda or Bear Dataset. Dataset tersebut terdiri atas gambar-gambar RGB berukuran 256×256 piksel. Digunakan arsitektur yang terdiri atas satu convolutional layer, satu pooling layer, satu flatten layer, dan satu dense layer. Rincian parameternya terdapat pada gambar di bawah. Tiap weights di tiap layer diinisialisasi secara random. Hasilnya kemudian dibandingkan dengan model neural network yang sama yang dihasilkan menggunakan pustaka Keras, dengan weight dan bias dari model pustaka sendiri diekspor ke model Keras sehingga menghasilkan model yang identik.

Gambar contoh Panda or Panda Dataset

```
model = Model()
model.add(Convolution(input_size=input_shape, padding_size=1, filter_size=(3, 3), num_filters=2, stride=1, bias=0))
model.add(Pooling(filter_size = (32,32), stride=32, mode='max'))
model.add(Flatten())
model.add(Dense(num_units=7, activation_function="relu"))
model.add(Dense(num_units=1, activation_function="sigmoid"))
```

Gambar struktur model dan parameternya untuk prediksi

```
model_k = keras.models.Sequential()
model_k.add(keras.layers.Conv2D(filters=2, kernel_size=(3, 3), activation="relu", strides=(1,1), padding="same", input_shape=input_shape))
model_k.add(keras.layers.MaxPooling2D(pool_size=(32,32), strides=32))
model_k.add(keras.layers.Flatten())
model_k.add(keras.layers.Dense(units=7, activation="relu"))
model_k.add(keras.layers.Dense(units=1, activation="sigmoid"))

model_k.compile()

model_k.layers[0].set_weights([l0_weights, l0_bias])
model_k.layers[3].set_weights([l3_weights, l3_bias])
model_k.layers[4].set_weights([l4_weights, l4_bias])
```

Gambar struktur model dengan pustaka Keras, weights di-*set* sehingga sama dengan model yang dibuat dengan pustaka sendiri

B. Hasil Prediksi

Prediksi kemudian dilakukan untuk lima gambar pertama dari dataset pengujian. Hasilnya terdapat pada gambar-gambar di bawah. Berdasarkan hasil tersebut, dapat disimpulkan model yang dibuat dengan pustaka sendiri memberikan hasil yang sangat mendekati nilai yang dihasilkan model dengan pustaka Keras. Perbedaan dapat terjadi dimungkinkan karena perbedaan tingkat ketelitian pada operasi *floating point* di kedua pustaka.

```
# Prediksi
y_p = model.predict(X_test[0:5])
print(y_p)
```

```
[[9.89365197e-22]
 [2.47779341e-19]
 [9.92786342e-18]
 [5.11667884e-22]
 [1.73297854e-20]]
```

```
# Kesimpulan prediksi
y_predicted = np.array([0. if y < 0.5 else 1. for y in y_p])
print("Actual y:", y_test[:5])
print("Predicted y:", y_predicted)
```

```
Actual y: [0. 0. 0. 0. 0.]
Predicted y: [0. 0. 0. 0. 0.]
```

Gambar hasil prediksi dengan pustaka sendiri

```
model_k.predict(X_test[:5])
```

```
1/1 [==============================] - 0s 282ms/step
array([[9.8936133e-22],
       [2.4778244e-19],
       [9.9278862e-18],
       [5.1166306e-22],
       [1.7330027e-20]], dtype=float32)
```

Gambar hasil prediksi dengan pustaka Keras

Lampiran

A. Pembagian Tugas

NIM	Nama	Pembagian Tugas
13520116	Mahesa Lizardy	Forward Propagation, Model, test Model, Laporan
13520146	Bryan Amirul Husna	Model, test Model, Laporan