LAPORAN HASIL TUGAS 7 PEMBELAJARAN MESIN

Training MNIST Dataset Menggunakan Convolutional Neural Network serta Menampilkan Grafik Akurasi dan Loss Function

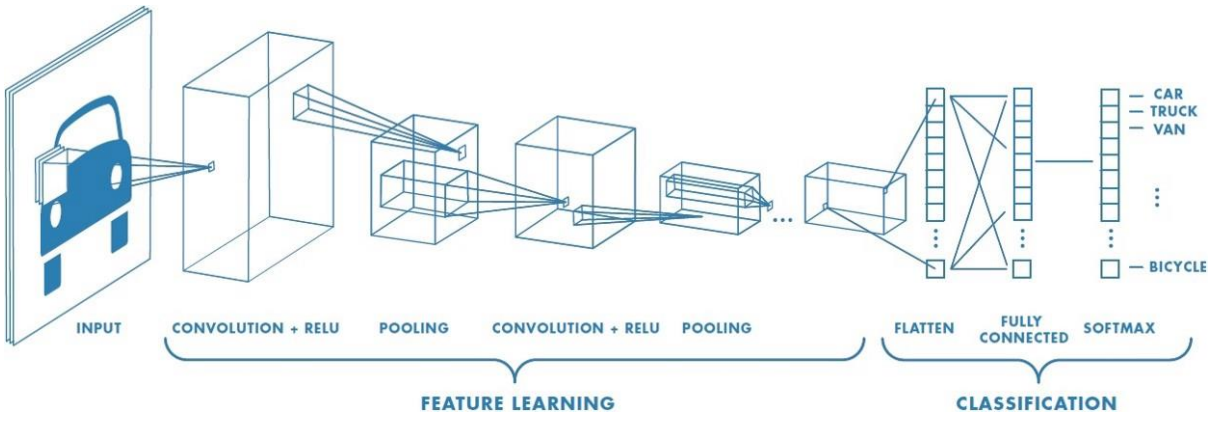
Candra Dewi Jodistiara – 15/383229/PA/16889

DESKRIPSI MASALAH DAN PERANCANGAN

Convolutional Neural Network (CNN) merupakan pengembangan dari metode Multi Layer Perceptron (MLP) yang terdiri dari satu atau lebih *convolutional layer* yang kemudian diikuti oleh satu atau lebih *fully connected layer*. *Fully connected layer* yang biasa digunakan dalam CNN ini adalah Multi Layer Perceptron, yang terdiri dari beberapa hidden layer, fungsi aktivasi, layer output, serta loss function. Konsep *convolutional layer* ini terinspirasi dari proses biologis, dimana adanya pola konektivitas antar neuron yang menyerupai kumpulan jaringan syaraf pada hewan. Arsitektur CNN didesain untuk dapat mengatasi input berupa gambar atau sinyal yang berukuran 2 dimensi.

Arsitektur CNN

CNN terdiri dari beberapa *convolutional layer* dan diikuti oleh *fully connected layer*. Input dari *convolutional layer* merupakan gamber berukuran m x m x r, dimana m merupakan panjang dan lebar ukuran gambar, serta r merupakan jumlah channel warna yang digunakan. Secara garis besar, arsitektur dari CNN dibagi menjadi 2, yaitu layer yang digunakan untuk ekstraksi fitur (*convolutional layer* dan *pooling*), serta layer yang digunakan untuk klasifikasi (*fully connected layer*)



Parameter yang terdapat pada layer yang digunakan untuk ekstraksi fitur antara lain

1. Convolutional + Relu

   Convolutional layer terdiri dari neuron yang tersusuh dari beberapa filter yang memiliki ukuran tertentu. Setiap filter ini akan digeser (*windowing*) di seluruh bagian pada gambar. Setiap pergeseran akan dilakukan operasi dot product antara matriks gambar dan filter sehingga diperoleh output yang disebut dengan *feature map*.

   Stride

   Stride merupakan parameter untuk menentukan ukuran pergeseran filter. Ukuran yang ditentukan adalah ukuran pergeseran baik secara vertikal maupun horizontal. Semakin kecil ukuran stride, maka semakin detail informasi yang diperoleh, namun membutuhkan komputasi yang besar.

   Zero Padding

   Zero padding merupakan parameter yang menentukan jumlah pixel yang ditambahkan pada sisi luar input. Hal ini dilakukan dengan tujuan dapat memanipulasi hasil output dari convolutional layer. Zero padding dilakukan agar ukuran dari layer output tidak jauh berbeda dengan ukuran dari layer input, sehingga penggunaan convolutional layer dapat diterapkan pada deep convolutional layer sehingga dapat diperoleh fitur yang lebih banyak. Hal ini juga dapat meningkatkan performa dari model karena convolutional layer karena lebih terfokus pada informasi yang sebenarnya.

2. Pooling layer

   Pooling layer atau downsampling layer merupakan layer yang terdiri dari sebuah filter dengan ukuran dan stride tertentu yang akan bergeser pada seluruh area feature map. Pooling yang biasa digunakan adalah Max Pooling dan Average Pooling. Sebagai contoh jika kita menggunakan Max Pooling 2x2 dengan stride 2, maka pada setiap pergeseran filter, nilai maximum pada area 2x2 pixel tersebut yang akan dipilih, sedangkan Average Pooling akan memilih nilai rata-ratanya. Tujuan dari penggunaan pooling layer adalah mengurangi dimensi dari feature map (downsampling), sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit dan mengatasi overfitting.

MNIST Dataset

MNIST Dataset merupakan dataset berisi gambar digit angka yang ditulis menggunakan tangan (*handwriting*). Setiap sampel pada datset tersebut berukuran 28 x 28 x 1, dan total terdapat 60000 sampel untuk training data dan 10000 sampel untuk test data.

PERANCANGAN MASALAH

Dalam tugas 7 ini, dibuat sebuah implementasi CNN pada MNIST dataset untuk dapat mengklasifikasikan digit yang terdapat pada setiap data. Dataset tersebut terdiri dari 10 kelas (angka 0 hingga 9), serta gambar berukuran 28 pixel * 28 pixel yang di*flatten* menjadi array berukuran 784, serta memiliki channel warna 1 buah, karena gambar tersebut hanya terdiri dari warna hitam dan putih.

Pada implementasi, akan dibentuk 2 buah convolutional layer. Pada layer pertama, akan dibuat zero padding berukuran 2x2 sehingga data yang semula berukuran 28x28 akan berubah menjadi berukuran 32x32. Kemudian, dilakukan konvolusi dengan kernel berukuran 5x5 sebanyak 32 fitur. Hasil dari proses konvolusi tersebut merupakan matriks berukuran 28x28. Kemudian dilakukan fungsi ReLU dan dilakukan pooling sebesar 2x2 sehingga layer menjadi berukuran 14x14. Untuk layer kedua, menggunakan kernel berukuran 5x5 sebanyak 64 fitur, dan dilakukan fungsi ReLU serta pooling sebesar 2x2 sehingga ukuran layer menjadi 7x7 sebanyak 64 fitur.

Kemudian, hasil dari convolutional layer tersebut akan dimasukkan ke dalam fully connected layer dengan 1 hidden layer sejumlah 1024 neuron. Untuk mengurangi kemungkinan terjadinya overfitting, dilakukan *dropout* sebelum menuju ke output layer. *Dropout* merupakan suatu metode untuk mengatasi overfitting dengan menghapus unit secara random dari neural network saat training. Terakhir, pada output layer terdapat 10 buah neuron yang masing-masing merepresentasikan label 0-9. Kemudian untuk proses training, digunakan softmax classifier dan Adam Optimizer untuk proses perubahan bobot.

Kemudian, untuk setiap iterasi ke 100, dilakukan pencatatan akurasi serta loss function ke dalam numpy array, yang nantinya akan dilakukan line plot untuk melihat perubahan akurasi serta loss function pada setiap step.

IMPLEMENTASI

Implementasi dilakukan dengan menggunakan bahasa pemrograman python dan bantuan library tensorflow untuk membentuk CNN dan mendapatkan dataset, serta library numpy dan fungsi pyplot pada matplotlib untuk plotting akurasi serta loss function untuk setiap step. Program dijalankan menggunakan kernel pada Google Colaboratory dan dirun menggunakan GPU Accelerator.

Pertama-tama, ekstrak MNIST dataset dari tensorflow dalam bentuk one hot encoding.

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
```

Kemudian, bentuk interactive session untuk membuat sesi baru pada tensorflow, menentukan training steps dan batch size.

```
import tensorflow as tf
sess = tf.InteractiveSession()

TRAIN_STEPS = 20000
BATCH_SIZE = 50

x = tf.placeholder(tf.float32, shape=[None, 784])
y_ = tf.placeholder(tf.float32, shape=[None, 10])
```

Mendefinisikan fungsi untuk menentukan weight dan bias, melakukan konvolusi, serta pooling.

```
def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)
```

```
def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                        strides=[1, 2, 2, 1], padding='SAME')
```

Melakukan operasi untuk layer konvolusi pertama.

```
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])

x_image = tf.reshape(x, [-1,28,28,1])

h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)
```

Melakukan operasi untuk layer konvolusi kedua.

```
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

Pada output layer konvolusi, dilakukan neural network dengan 1 hidden layer.

```
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
```

Mendefinisikan placeholder untuk proses dropout yang hanya dilakukan pada training dataset, tidak pada test dataset.

```
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

Menghitung output layer dari hasil layer yang telah dilakukan dropout.

```
W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2
```

Mendefinisikan classifier, weight updater, serta kalkulasi akurasi.

```
# Softmax classifier
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=y_cc

# Adam updater
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)

# Calculate accuracy
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

Melakukan inisiasi numpy array untuk menyimpan akurasi serta loss function sebanyak step size.

```
import numpy as np

n = int((TRAIN_STEPS)/100)+1
ACCURACY = np.empty(n)
LOSS = np.empty(n)


# Initialize variables
sess.run(tf.global_variables_initializer())

for i in range(TRAIN_STEPS+1):

  batch = mnist.train.next_batch(BATCH_SIZE)

  if i % 100 == 0:
    j = int(i/100)
    ACCURACY[j] = accuracy.eval(feed_dict={x: batch[0], y_: batch[1], keep_prob: 1
    LOSS[j] = cross_entropy.eval(feed_dict={x: batch[0], y_: batch[1], keep_prob:
    print('step %d, training accuracy %g, loss %g' % (i, ACCURACY[j], LOSS[j]))

  train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
```

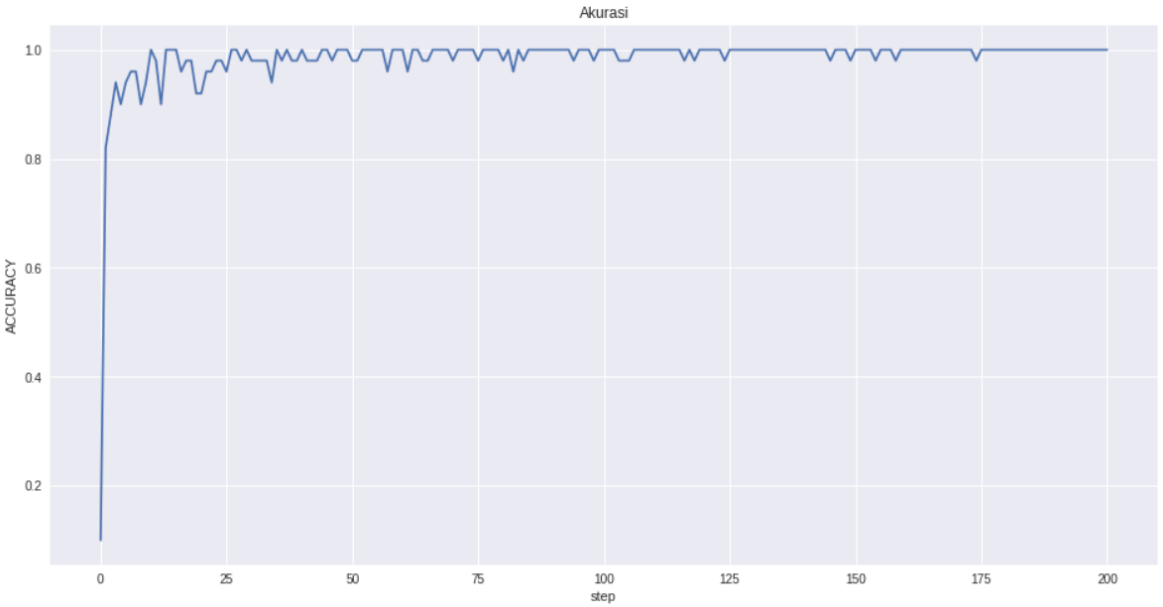
Menampilkan grafik hasil akurasi serta loss function untuk setiap step

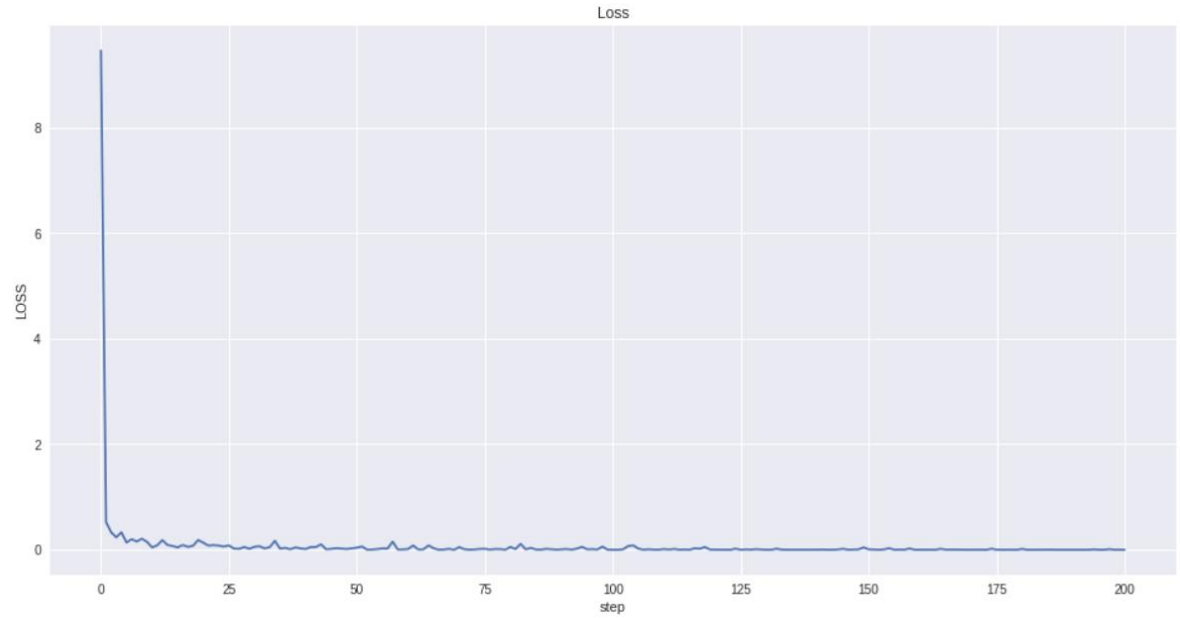
```
step = np.arange(n)

plt.figure(figsize=(16,8))
plt.plot(step, ACCURACY)
plt.xlabel("step")
plt.ylabel("ACCURACY")
plt.title("Akurasi")
plt.show()
```

```
plt.figure(figsize=(16,8))
plt.plot(step, LOSS)
plt.xlabel("step")
plt.ylabel("LOSS")
plt.title("Loss")
plt.show()
```

Berikut adalah grafik hasil akurasi serta loss function saat program menjalankan proses training.





Referensi:

https://www.tensorflow.org/versions/r1.0/get_started/mnist/pros

https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94

https://deeplearning4j.org/convolutionalnetwork

https://en.wikipedia.org/wiki/Convolutional_neural_network