Laporan Tugas EL4233 Digit Recognition

Anggota Kelompok:

- Pravitasari Anjani / 18214032
- Nurlaili Rizki Hasanah / 18214049

Contents

1 Deskripsi Masalah 1
2 Desain Solusi 1
3 Implementasi Solusi 2
4 Pengujian & Analisis 5
5 Permasalahan 6
6 Kesimpulan 6
7 Literatur 6

1 Deskripsi Masalah

Terdapat data berupa 4.600 gambar angka yang perlu dikenali. Data tersebut berupa *file* JPG *True Color* RGB dengan resolusi 125x125. Data tersebut perlu dikenali digitnya dengan menggunakan metode *multi-class classification* pada Neural Network. Untuk memodelkan Neural Network tersebut dapat digunakan data *training set* MNIST yang telah tersedia, yaitu 60.000 data gambar *greyscale* dengan resolusi 28x28 beserta label digitnya.

2 Desain Solusi

Desain solusi yang digunakan untuk pemecahan masalah adalah dengan menggunakan Multi-Layer Perceptron. Multi-Layer Perceptron digunakan untuk *multi-class classification.* Perceptron yang dirancang yaitu tiga *layer* Perceptron yaitu *input layer, hidden layer,* dan *output layer.* Data yang akan dikenali dicocokkan dengan *training set* terlebih dahulu, yaitu diubah menjadi format yang sama berupa gambar *greyscale* dengan resolusi 28x28. Jumlah masukan pada *layer input* adalah jumlah total pixel dari masing-masing gambar yaitu sebanyak 784. Jumlah keluaran pada *output layer* yaitu 10, yang merupakan representasi angka 0-9. Pada *output layer* digunakan *Softmax regression,* yaitu suatu fungsi aktivasi yang memberikan nilai probabilitas gambar tersebut adalah digit tertentu (dari 0-9).

3 Implementasi Solusi

Implementasi solusi dilakukan dengan menggunakan *library* Keras untuk Neural Network pada Python. Selain itu juga digunakan *library* tambahan Pandas dan NumPy untuk pemrosesan data. Prosedur implementasi yang dilakukan adalah sebagai berikut.

1. Menyiapkan *training dataset* dari MNIST *database* dan mengkonversi data tersebut dalam format csv
2. Menyiapkan *test data* dengan mengkonversi gambar ke format *greyscale* dengan resolusi 28x28 dan mengkonversi data tersebut dalam format csv
3. Merancang model Neural Network dengan *library* Keras
4. Melakukan *training* pada model yang sudah dirancang dengan *dataset* yang sudah disiapkan
5. Melakukan prediksi pada data *test* dengan model yang ada

Berikut ini adalah *script* untuk memproses *training dataset.*

```
def convert(imgf, labelf, outf, n):
    f = open(imgf, "rb")
    o = open(outf, "w")
    l = open(labelf, "rb")

    f.read(16)
    l.read(8)
    images = []

    for i in range(n):
        image = [ord(l.read(1))]
        for j in range(28*28):
            image.append(ord(f.read(1)))
        images.append(image)

    for image in images:
        o.write(",".join(str(pix) for pix in image)+"\n")
    f.close()
    o.close()
    l.close()

convert("train-images.idx3-ubyte", "train-labels.idx1-ubyte",
        "train.csv", 60000)
```

Script tersebut akan mengkonversi data ke format csv. Data yang tersedia yaitu sebanyak 60.000 data.

Setelah itu data *test* diproses dengan *script* berikut. Sebelumnya, data mentah yang merupakan *file* JPG *True Color* RGB dengan resolusi 125x125 diubah menjadi *file greyscale* dengan resolusi 28x28.

```
import os
from PIL import Image

path = './'
images = []
o = open ("test.csv", "w")
l = open ("label.txt", "w")
i = 1

for filename in os.listdir(path):
        img = Image.open(path + filename)  # convert image to 8-bit grayscale
        WIDTH, HEIGHT = img.size

        data = list(img.getdata()) # convert image data to a list of integers
        # convert that to 2D list (list of lists of integers)
        data = [data[offset:offset+WIDTH] for offset in range(0, WIDTH*HEIGHT, WIDTH)]

        image = []
        for y in range(HEIGHT):
                for x in range(WIDTH):
                        image.append(255 - data[y][x])
        images.append(image)
        l.write(filename+"\n")
        img.close()
        i += 1
        if (i > 4600):
                break

for image in images:
   o.write(",".join(str(pix) for pix in image)+"\n")
o.close()
l.close()
```

Selanjutnya dilakukan perancangan *Multi Layer* Perceptron dan *training* data serta prediksi data *test* dengan *script* berikut.

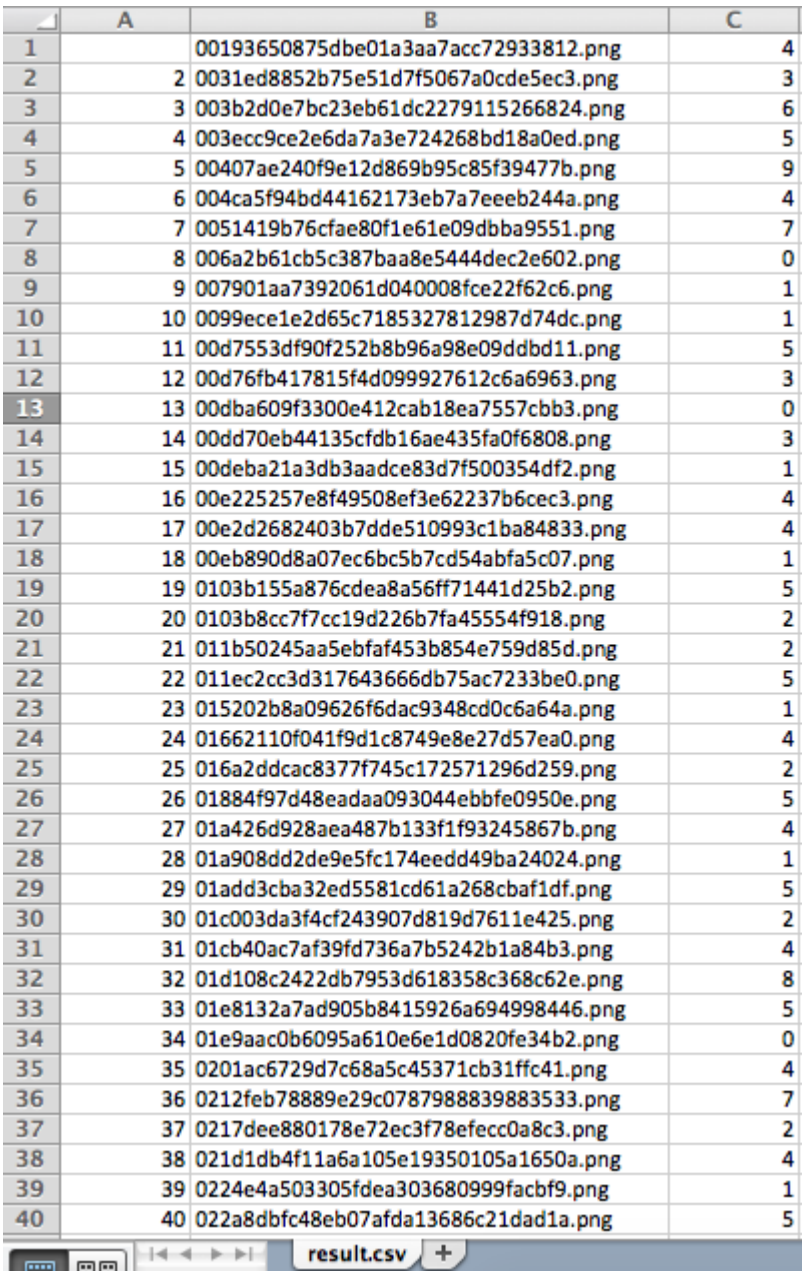
```
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers.core import Dense, Activation, Dropout

import pandas as pd
import numpy as np

# Read data
```

```
train = pd.read_csv('./train.csv')
labels = train.ix[:,0].values.astype('int32')
X_train = (train.ix[:,1:].values).astype('float32')
X_test = (pd.read_csv('./test.csv').values).astype('float32')

# convert list of labels to binary class matrix
y_train = np_utils.to_categorical(labels)

# pre-processing: divide by max and substract mean
scale = np.max(X_train)
X_train /= scale
X_test /= scale

mean = np.std(X_train)
X_train -= mean
X_test -= mean

input_dim = X_train.shape[1]
nb_classes = y_train.shape[1]

# Here's a Deep Dumb MLP (DDMLP)
model = Sequential()
model.add(Dense(512, input_dim=input_dim))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

# we'll use categorical xent for the loss, and RMSprop as the optimizer
model.compile(loss='categorical_crossentropy', optimizer='rmsprop')

print("Training...")
model.fit(X_train, y_train, nb_epoch=28, batch_size=128, validation_split=0.1, verbose=1)

print("Generating test predictions...")
preds = model.predict_classes(X_test, verbose=0)

def write_preds(preds, fname):
   pd.DataFrame({"ImageId": list(range(1,len(preds)+1)), "Label": preds}).to_csv(fname, index=False,
header=True)

write_preds(preds, "result.csv")
```

Hasil prediksi data *test* disimpan dalam format csv.

4 Pengujian & Analisis

Berikut merupakan tampilan *result.csv* yang merupakan hasil prediksi data *test.*



	A	B	C
1		00193650875dbe01a3aa7acc72933812.png	4
2	2	0031ed8852b75e51d7f5067a0cde5ec3.png	3
3	3	003b2d0e7bc23eb61dc2279115266824.png	6
4	4	003ecc9ce2e6da7a3e724268bd18a0ed.png	5
5	5	00407ae240f9e12d869b95c85f39477b.png	9
6	6	004ca5f94bd44162173eb7a7eeeb244a.png	4
7	7	0051419b76cfae80f1e61e09dbba9551.png	7
8	8	006a2b61cb5c387baa8e5444dec2e602.png	0
9	9	007901aa7392061d040008fce22f62c6.png	1
10	10	0099ece1e2d65c7185327812987d74dc.png	1
11	11	00d7553df90f252b8b96a98e09ddbd11.png	5
12	12	00d76fb417815f4d099927612c6a6963.png	3
13	13	00dba609f3300e412cab18ea7557cbb3.png	0
14	14	00dd70eb44135cfdb16ae435fa0f6808.png	3
15	15	00deba21a3db3aadce83d7f500354df2.png	1
16	16	00e225257e8f49508ef3e62237b6cec3.png	4
17	17	00e2d2682403b7dde510993c1ba84833.png	4
18	18	00eb890d8a07ec6bc5b7cd54abfa5c07.png	1
19	19	0103b155a876cdea8a56ff71441d25b2.png	5
20	20	0103b8cc7f7cc19d226b7fa45554f918.png	2
21	21	011b50245aa5ebfaf453b854e759d85d.png	2
22	22	011ec2cc3d317643666db75ac7233be0.png	5
23	23	015202b8a09626f6dac9348cd0c6a64a.png	1
24	24	01662110f041f9d1c8749e8e27d57ea0.png	4
25	25	016a2ddcac8377f745c172571296d259.png	2
26	26	01884f97d48eadaa093044ebbfe0950e.png	5
27	27	01a426d928aea487b133f1f93245867b.png	4
28	28	01a908dd2de9e5fc174eedd49ba24024.png	1
29	29	01add3cba32ed5581cd61a268cbaf1df.png	5
30	30	01c003da3f4cf243907d819d7611e425.png	2
31	31	01cb40ac7af39fd736a7b5242b1a84b3.png	4
32	32	01d108c2422db7953d618358c368c62e.png	8
33	33	01e8132a7ad905b8415926a694998446.png	5
34	34	01e9aac0b6095a610e6e1d0820fe34b2.png	0
35	35	0201ac6729d7c68a5c45371cb31ffc41.png	4
36	36	0212feb78889e29c07879888839883533.png	7
37	37	0217dee880178e72ec3f78efecc0a8c3.png	2
38	38	021d1db4f11a6a105e19350105a1650a.png	4
39	39	0224e4a503305fdea303680999facbf9.png	1
40	40	022a8dbfc48eb07afda13686c21dad1a.png	5

result.csv

Kolom B menunjukkan nama file (data *test)* dan kolom C menunjukkan hasil *digit recognition*. Hasil pengenalan digit tersebut kemudian dicocokkan dengan data gambar yang ada untuk validasi. Karena proses validasi masih dilakukan secara manual dengan mencocokkan masing-masing *row* dan melihat *file* gambar, data yang kami cocokkan hanya 1.631 dari 4.600 data.

Dari hasil validasi tersebut, beberapa hasil tidak sesuai dengan gambar. Hal tersebut dapat disebabkan karena gambar yang harus dikenali tidak ideal, misalnya ukuran gambar yang tidak proposional (terlalu kecil ataupun terlalu besar), dan posisi gambar yang tidak sesuai. Selain itu, karena angka tersebut adalah *handwritten,* bentuk angka yang ada juga berbagai macam. Hal itu menyebabkan model yang dirancang tidak dapat sepenuhnya mengenali digit yang ada dengan benar. Selain itu, model Neural Network yang dirancang kemungkinan masih belum optimal dan masih dapat diperbaiki lagi.

5 Permasalahan

Permasalahan yang ditemui selama melakukan *digit recognition* adalah menentukan *tools* yang digunakan. Awalnya, kami mencoba menggunakan Tensorflow namun kami menemukan kendala saat harus mengonversi *test data* ke dalam format IDX-Ubyte. Akhirnya kami mencoba menggunakan Keras yang merupakan API *library* untuk menjalankan Tensorflow.

6 Kesimpulan

Multi-Layer Perceptron dapat digunakan untuk meyelesaikan permasalah *digit recognition* pada gambar. Implementasi dari solusi ini dapat dilakukan dengan berbagai alternatif, salah satunya adalah library Keras pada Python. Ketelitian yang dihasilkan dari implementasi solusi permasalahan *digit recognition* ini adalah sekitar 34,76%. Nilai ini masih sangat rendah dan masih bisa lebih ditingkatkan. Solusi yang sudah ada dapat dikembangkan kembali, misalnya dengan mengubah desain Neural Network dan mencari alternatif implementasi lain.

7 Literatur

https://www.kaggle.com/c/digit-recognizer/data

https://www.kaggle.com/fchollet/simple-deep-mlp-with-keras/code/code

https://pjreddie.com/projects/mnist-in-csv/

https://keras.io/

https://www.datacamp.com/community/blog/keras-cheat-sheet#gs.A5jI34E

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

http://codegists.com/snippet/python/mnist-to-jpgpy_ischlag_python