Minimal Neural Network Untuk Pengenalan Bendera Negara

Ahmad Sabri dan Feni Andriani
{sabri, feni.andriani}@staff.gunadarma.ac.id
Teknik Informatika, Universitas Gunadarma
Jl. Margonda Raya 100, Depok, Indonesia

ABSTRAK

Penelitian ini menerapkan TensorFlow pada minimal neural network untuk mengenali citra bendera negara berdasarkan negara asalnya.. Dataset dihimpun dari internet berupa citra yang didominasi bendera sebagai latar depannya, dan berbagai latar seperti langit biru/berawan, latar pepohonan, menghadap kiri atau kanan, berkibar ataupun kuncup. Citra bendera yang mendominasi diperlukan untuk meniadakan prosedur lokalisasi. Sebagai contoh penerapaan, dataset dibatasi pada bendera negara-negara ASEAN. Untuk pengembangannya, cakupan dapat diperluas untuk seluruh negara di dunia. Hasil yang diperoleh memiliki ketepatan 74% terhadap testing image yang digunakan.

Kata Kunci: *Neural Network, Tensorflow, Dataset, Klasifikasi Objek, Pengenalan Objek*.

PENDAHULUAN

Perkembangan neural network dalam kurun dekade terakhir mengalami peningkatan yang signifikan seiring dengan meningkatnya kemampuan komputasi komputer. Penerapan neural network dalam *computer vision* antara lain untuk pengklasifikasian dan pengenalan objek, seperti wajah [1,2], rambu lalu lintas [3,4]; dalam bidang *natural language processing* seperti *speech recognition* [5] dan *text classification* [6].

Dimotivasi oleh [4] yang menerapkan minimal neural network untuk pengenalan rambu lalu lintas, paper ini menggunakan model serupa dengan dataset yang berbeda, dan memodifikasi *script* sehingga diperoleh model yang dapat mengenali (mengklasifikasikan) bendera nasional negara-negara di dunia berdasarkan negara asalnya, dengan file image sebagai inputnya. Pemilihan bendera negara sebagai objek klasifikasi disebabkan sejauh penelusuran penulis, belum terdapat penelitian serupa dengan pendekatan neural network. Selain itu, pengembangan dapat diarahkan untuk pendidikan pengenalan negara-negara di dunia, termasuk di antaranya mengenali bendera negara.

Pada penelitian ini penggunaan model minimal neural network dimaksudkan untuk melakukan pengenalan bendera negara ASEAN.

METODE PENELITIAN

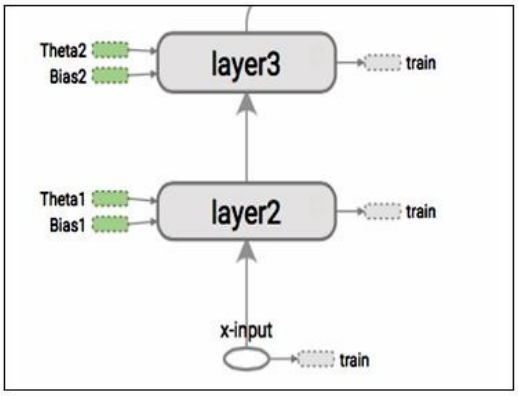
Penggunaan model minimal neural network dimaksudkan agar program dapat berjalan pada komputer dengan spesifikasi minimal, dengan waktu training yang relatif singkat, dengan hanya mengandalkan komputasi CPU. Jaringan pada model minimal neural network ini terdiri dari sebuah *fully connected layer*, di mana setiap neuron terhubung ke setiap nilai input. Model dalam penelitian ini menggunakan ReLU *activation function*:

$$f(x) = \max(0,x) \quad (1)$$

TensorFlow [7] merupakan struktur data array multidimensi yang mengalir di dalam graf *neural network*. Operasi terhadap TensorFlow dilakukan pada neuron. Operasi yang dimaksud terdiri dari operasi penjumlahan, perkalian, mengubah matriks menjadi vektor atau sebaliknya, dan sebagainya. Output akhir dari neural network adalah sebuah vektor berukuran 1 X *n* (di mana *n* adalah banyak kelas klasifikasi) yang setiap elemennya merupakan skor dari masing-masing kelas. Sebagai contoh, jika inputnya berupa bendera Indonesia, maka diharapkan skor elemen ke 1 memiliki skor tertinggi (lihat penataan folder di bagian Pembahasan).

Gambar 1 menampilkan visualisasi alur TensorFlow pada neural network multilayer, dan Gambar 2 menampilkan model *single fully-connected layer* yang digunakan pada

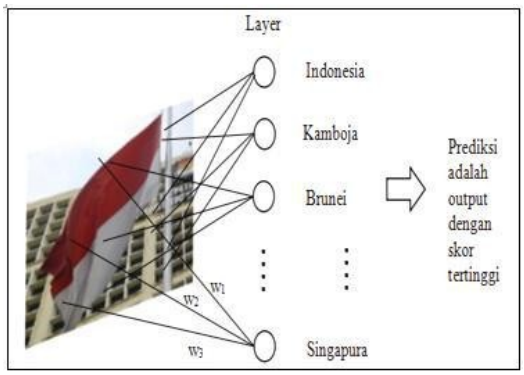
penelitian ini, di mana inputnya berupa citra bendera.



Gambar 1. *Visualisasi TensorFlow* [4]

Dalam [4] diberikan Python *script* untuk pengklasifikasian citra rambu lalu lintas dengan neural network dan TensorFlow sebagai proses yang berjalan di atasnya.

Penelitian ini mengadaptasi dan memodifikasi *script* dan mengganti dataset pada [4] tersebut untuk pengenalan bendera negara-negara dunia. Modifikasi yang dilakukan berupa pemisahan antara *script* untuk *training* dan *testing*/*implementing* menjadi dua file yang berbeda. Hal ini memerlukan penambahan fitur *save graph* pada fase training untuk kemudian di-*restore* pada fase testing. Pemisahan ini dilakukan agar implementasi dapat dilakukan untuk seterusnya tanpa harus melakukan kembali proses training, kecuali jika terjadi penambahan dataset.



Gambar 2. Ilustrasi model *single fully connected layer*

Cakupan dataset dibatasi pada bendera negara-negara anggota ASEAN sebanyak 10 negara (10 kelas), yaitu Indonesia, Malaysia, Singapura, Filipina, Thailand, Brunei Darussalam, Vietnam, Laos, Kamboja, dan Myanmar. Setiap folder berisi citra berformat jpg dari bendera-bendera negara yang sama, dengan berbagai keadaan seperti: berkibar/kuncup, latar belakang langit (cerah/berawan) ataupun gedung maupun pohon. Dataset terdiri dari dataset untuk training (15-17 citra per kelas) dan testing (5 citra per kelas) Citra diperoleh dari internet dengan meng-*crop* sedemikian rupa sehingga bendera menjadi objek dominan. Hal ini dilakukan karena *script* hanya untuk mengklasifikasikan citra tanpa melakukan lokalisasi objek pada citra.

Pengujian model dilakukan pada fase testing terhadap 50 *testing image* (citra uji). Akurasi dinyatakan sebagai rasio antara jumlah klasifikasi yang tepat terhadap total *testing image*.

Script ditulis dalam Python 3.5 dan TensorFlow versi 1.6.0, dan dijalankan pada platform Jupyter Notebook.

Perangkat yang digunakan dalam penelitian ini adalah sebuah laptop dengan spesifikasi:
CPU AMD A4-5000 APU.
Memori 2 GB.
Sistem operasi: Windows 8.1.

PEMBAHASAN

Dataset

Penataan folder dataset bendera negara negara ASEAN ditunjukkan dalam Tabel 1 berikut:

Tabel 1. *Penataan folder dataset bendera negara*

Folder	Bendera	Negara
000		Indonesia
001		Kamboja
002		Brunei D.
003		Filipina
004		Laos
005		Malaysia

006		Vietnam
007		Thailand
008		Myanmar
009		Singapura

Fase Training

Pada fase training (untuk *script* lihat Lampiran 1), citra pada training dataset dimuat ke dalam memori dan di-*resize* menjadi berukuran 64 x 64, dengan 3 *channel* RGB.

Model neural network menggunakan minimal neural network, berupa satu *fully connected layer* dengan 10 neuron, sesuai dengan banyak kelas. Citra dan label yang bersesuaian masing-masing diterima oleh tensor placeholder image_ph dan label_ph.

Setiap neuron menerima 64 x 64 x 3 = 12288 input dari tensor image_flat, yang merupakan *flattened image* dari image_ph. Output dari *layer* ini adalah tensor logits dengan 10 kolom. Prediksi (klasifikasi) citra dihasilkan oleh tensor predicted_labes, berupa indeks dengan nilai logits terbesar. Nilai *loss* diperoleh dari tensor loss dengan menggunakan fungsi *cross-entropy*, yang merupakan fungsi yang paling cocok untuk mengklaasifikasikan objek. Tensor train menerapkan ADAM optimizer untuk memperoleh nilai *loss* minimal.

Training session dilakukan sebanyak 201 iterasi (*epoch*). Jika iterasi menghasilkan nilai *loss* yang semakin kecil, model semakin akurat. Graf beserta *checkpoint* yang dihasilkan kemudian disimpan pada folder yang telah diterntukan. Gambar 1 menampilkan output dari fase training berupa *loss value* pada setiap sepuluh iterasi. Fase training dengan perangkat yang digunakan membutuhkan waktu maksimal 5 menit.

Fase Testing dan implementing

Pada fase testing (*script* lihat Lampiran 2), citra pada testing dataset dimuat ke dalam memori, dan di-*resize* menjadi berukuran 64 x 64 x 3, sebagaimana pada tahap training.

Kemudian graf dan *checkpoint* di-*restore*, dan *testing session* dijalankan berdasar graf dan *checkpoint* yang di-*restore* tersebut. Output dari testing ini adalah persentase akurasi dari model terhadap testing dataset, ditunjukkan pada Gambar 2. Fase testing membutuhkan waktu tidak lebih dari 2 menit. Lihat Lampiran 3 untuk output final dari fase testing.

```
Loss:  4.21112
Loss:  3.16355
Loss:  1.47721
Loss:  0.703151
Loss:  0.416746
Loss:  0.282524
Loss:  0.213053
Loss:  0.166212
Loss:  0.136547
Loss:  0.114552
Loss:  0.0983067
Loss:  0.0856873
Loss:  0.0755615
Loss:  0.067287
Loss:  0.0603968
Loss:  0.0545894
Loss:  0.0496423
Loss:  0.0453887
Loss:  0.0417012
Loss:  0.0384808
Loss:  0.0356494
training done
```

Gambar 1. Output dari fase training

Untuk keperluan implementasi, *testing script* ini dimodifikasi sehingga user dapat memasukkan input citra, dan outputnya berupa prediksi dari klasifikasi image tersebut.

HASIL

Berdasarkan hasil testing, diperoleh akurasi model adalah 74%, sebagaimana output yang ditunjukkan pada Gambar 2. Tampilan output berikutnya berupa visualisasi *testing image* dan prediksi yang diberikan dapat dilihat pada Lampiran 3. Persentase akurasi dapat ditingkatkan jika fase training dilakukan dengan dataset dengan jumlah yang masif, dan *testing image* tidak jauh berbeda daripada *training image* Dalam beberapa percobaan training, konvergensi

nilai *loss* berjalan sangat lambat atau bahkan tidak berjalan sama sekali (nilai *loss* tidak berubah setelah sejumlah iterasi). Hal ini diduga akibat adanya kemiripan beberapa bendera negara, dalam hal ini Indonesia, Singapura dan Thailand, serta Laos dan Kamboja. Hipotesis untuk mengatasinya adalah bendera yang memiliki kemiripan diberi label yang berbeda jauh (tidak berurutan). Oleh karena itu, pada dataset bendera Indonesia berlabel 000, Thailand berlabel 007, dan Singapura belabel 009.

Untuk bendera lainnya yang memiliki kemiripan, Kamboja berlabel 001, dan Laos berlabel 004. Berdasarkan observasi pada fase training, dataset yang disusun demikian menunjukkan bahwa pada setiap iterasi, lebih banyak didapati nilai loss semakin kecil dan konvergen.

```
INFO:tensorflow:Restoring parameters
from ./saved graph/flag_detector
citra uji: 50
Prediksi tepat: 37
Akurasi: 74.0000 %
```

Gambar 2. Output awal dari fase testing

KESIMPULAN DAN SARAN

Model dapat memprediksi bendera dengan ketepatan 74% pada testing dataset. Eksekusi model pada fase training dengan dataset yang tersedia memerlukan waktu tidak lebih dari 5 menit, dan untuk fase testing membutuhkan waktu tidak lebih dari 2 menit. Akurasi model dapat ditingkatkan dengan menggunakan training dataset yang lebih banyak dan bervariasi, serta labeling yang tidak berurutan bagi citra yang memiliki kemiripan. Sebagai saran, selain file citra, input image dapat diberikan melalui video ataupun webcam secara real time.

DAFTAR PUSTAKA

Taylor, W.K. "Machine learning and recognition of faces", *Electronics Letters*, 3:436-437, 1967

Kasar, M.M., Bhattacharyya, D., dan Kim, T.-h, "Face Recognition Using NeuralNetwork:AReview", *International Journal of Security and Its Applications,* 10(3):81-100, 2016.

Boujemaa, K.S., Berrada, I., Bouhoute, A., dan Boubouh, K., "Traffic sign recognition using convolutional neural networks," WINCOM, Rabat, hal. 1-6, 2017.

Abdulla, W. https://github.com/waleedka/traffic-signs-tensorflow. , 2017.

Lim, C.P., Woo, S.C., Loh. A.S. dan Osman, R., "Speech recognition using artificialneuralnetworks," *Proceedings of the First International Conference on Web Information Systems Engineering*, Hong Kong, hal. 419-423 vol.1, 2000.

Li,C.H.,Park,S.C.,"Text Categorization Based on Artificial Neural Networks". Dalam: Neural Information Processing. ICONIP 2006. Lecture Notes in Computer Science, vol 4234. Springer, Berlin, Heidelberg.

Google, Inc., TensorFlow™, https://www.tensorflow.org/

LAMPIRAN 1: Script *training.py*

```
import os
import random
import skimage.data
import skimage.transform
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

def load_data(data_dir):
    directories = [d for d in os.listdir(data_dir)
                   if os.path.isdir(os.path.join(data_dir, d))]
    labels = []
    images = []
    for d in directories:
        label_dir = os.path.join(data_dir, d)
        file_names = [os.path.join(label_dir, f)
                      for f in os.listdir(label_dir) if
        f.endswith(".jpg")] for f in file_names:
            images.append(skimage.data.imread(f))
            labels.append(int(d))
    return images, labels

ROOT_PATH = "YOUR_ROOT_PATH"
train_data_dir = os.path.join(ROOT_PATH, "TRAINING_DATASET_DIRECTORY")
images, labels = load_data(train_data_dir)
siz=64
images32 = [skimage.transform.resize(image, (siz, siz))
            for image in images]
labels_a = np.array(labels)
images_a = np.array(images32)

graph = tf.Graph()
with graph.as_default():
    images_ph = tf.placeholder(tf.float32, [None, siz, siz, 3],name="images_ph")
    labels_ph = tf.placeholder(tf.int32, [None],name="labels_ph")
    images_flat = tf.contrib.layers.flatten(images_ph)
    logits = tf.contrib.layers.fully_connected(images_flat, 62, tf.nn.relu)
    predicted_labels = tf.argmax(logits, 1,name="predicted_labels")
    loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(labels=labels_ph,
logits=logits))
    train = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)
    init = tf.initialize_all_variables()
    session = tf.Session()
    _ = session.run([init])
    saver = tf.train.Saver()
    for i in range(201):
        _, loss_value = session.run([train, loss],
                                    feed_dict={images_ph: images_a, labels_ph: labels_a})
        if i % 10 == 0:
            print("Loss: ", loss_value)
    saver.save(session, './saved graph/flag_detector')
```

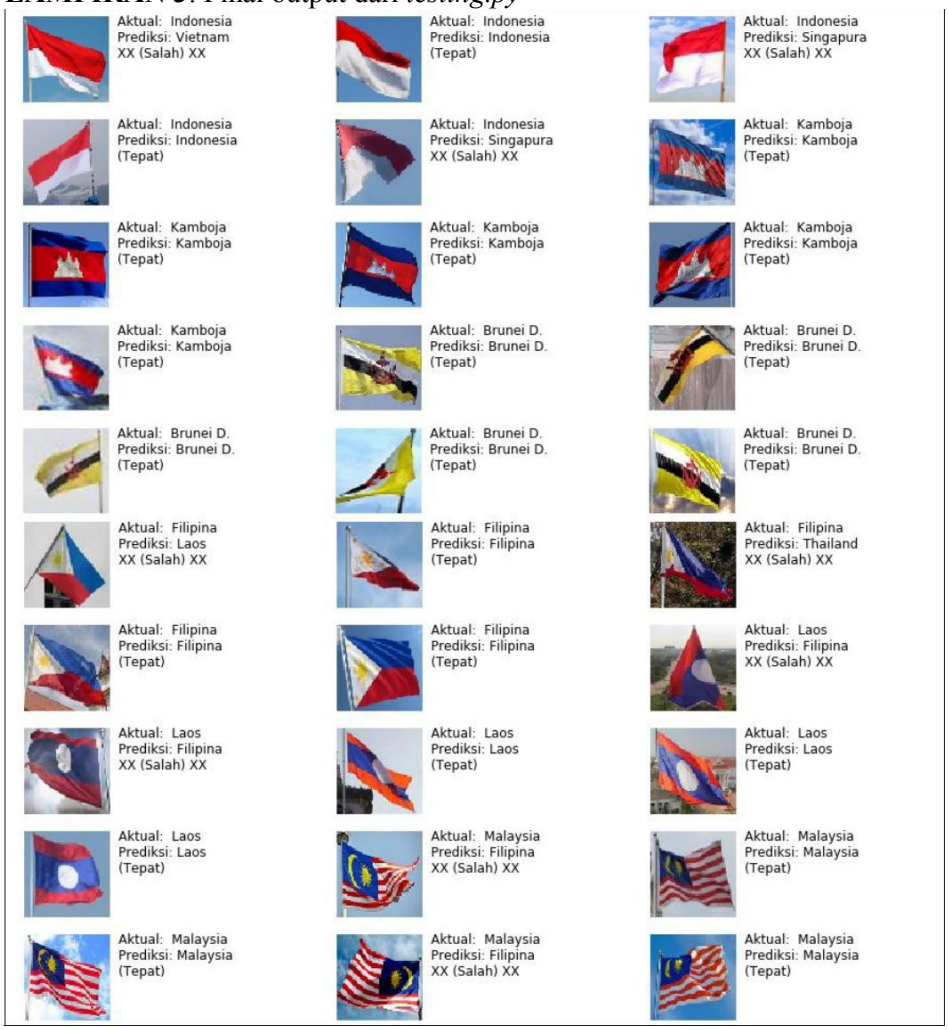
```
print("training done")
session.close()
```

LAMPIRAN 2: Script *testing.py*

```
import os
import random
import skimage.data
import skimage.transform
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

def load_data(data_dir):
    directories = [d for d in os.listdir(data_dir)
                   if os.path.isdir(os.path.join(data_dir, d))]
    labels = []
    images = []
    for d in directories:
        label_dir = os.path.join(data_dir, d)
        file_names = [os.path.join(label_dir, f)
                      for f in os.listdir(label_dir) if f.endswith(".jpg")]
        for f in file_names:
            images.append(skimage.data.imread(f))
            labels.append(int(d))
    return images, labels

ROOT_PATH = "YOUR_ROOT_PATH"
test_data_dir = os.path.join(ROOT_PATH, "TESTING_DATASET_DIRECTORY")

siz=64
test_images32 = [skimage.transform.resize(image, (siz, siz))
                 for image in images]
test_labels = [labels[i] for i in range(len(test_images32))] countries=['Indonesia','Kamboja','Brunei
D.', 'Filipina', 'Laos', 'Malaysia', 'Vietnam', 'Thailand',
  'Myanmar', 'Singapura']

with tf.Session() as sess:
    new_saver = tf.train.import_meta_graph('./saved graph/flag_detector.meta')
    graph = tf.get_default_graph()
    new_saver.restore(sess, tf.train.latest_checkpoint('./saved graph/'))
    images_ph=graph.get_tensor_by_name('images_ph:0')
    labels_ph=graph.get_tensor_by_name('labels_ph:0')
    predicted_labels = graph.get_tensor_by_name('predicted_labels:0')
    predicted = sess.run([predicted_labels],
                 feed_dict={images_ph:test_images32})[0]
    match_count = sum([int(y == y_) for y, y_ in zip(test_labels, predicted)])
    accuracy = match_count / len(test_labels)
    print('# citra uji:',len(test_labels))
    print ('Prediksi tepat:',match_count)
    print("Akurasi: {0:.4f} %".format(accuracy*100))
    print(test_labels)
```

```
    print(predicted)
sess.close()

fig = plt.figure(figsize=(15, 30))
for i in range(len(test_images32)):
    truth = test_labels[i]
    prediction = predicted[i]
    plt.subplot(17, 3,1+i)
    plt.axis('off')
    if truth == prediction:
        plt.text(70, 30, "Aktual: {0}nPrediksi: {1}\n(Tepat)".format(countries[truth],
countries[prediction]),
            fontsize=12)
    else:
        plt.text(70, 30, "Aktual: {0}nPrediksi: {1}\nXX (Salah) XX ".format(countries[truth],
countries[prediction]),
            fontsize=12)
    plt.imshow(test_images32[i])
```

LAMPIRAN 3: Final output dari *testing.py*




Aktual: Vietnam
Prediksi: Vietnam
(Tepat)
Aktual: Vietnam
Prediksi: Vietnam
(Tepat)
Aktual: Vietnam
Prediksi: Vietnam
(Tepat)
Aktual: Vietnam
Prediksi: Vietnam
(Tepat)
Aktual: Vietnam
Prediksi: Vietnam
(Tepat)
Aktual: Thailand
Prediksi: Laos
XX (Salah) XX
Aktual: Thailand
Prediksi: Laos
XX (Salah) XX
Aktual: Thailand
Prediksi: Thailand
(Tepat)
Aktual: Thailand
Prediksi: Thailand
(Tepat)
Aktual: Thailand
Prediksi: Thailand
(Tepat)
Aktual: Myanmar
Prediksi: Myanmar
(Tepat)
Aktual: Myanmar
Prediksi: Myanmar
(Tepat)
Aktual: Myanmar
Prediksi: Myanmar
(Tepat)
Aktual: Myanmar
Prediksi: Myanmar
(Tepat)
Aktual: Myanmar
Prediksi: Myanmar
(Tepat)
Aktual: Singapura
Prediksi: Singapura
(Tepat)
Aktual: Singapura
Prediksi: Singapura
(Tepat)
Aktual: Singapura
Prediksi: Vietnam
XX (Salah) XX
Aktual: Singapura
Prediksi: Indonesia
XX (Salah) XX
Aktual: Singapura
Prediksi: Singapura
(Tepat)