Laporan praktikum kecerdasan buatan

Simple Neural Network

Disusun untuk memenuhi tugas
praktikum kecerdasan buatan

Oleh:

Rizka Nuzulia
(2008107010012)

PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
DARUSSALAM, BANDA
ACEH 2022

LAPORAN HASIL SIMPLE NEURAL

NETWORK DENGAN MENGGUNAKAN PYTHON DAN KERAS

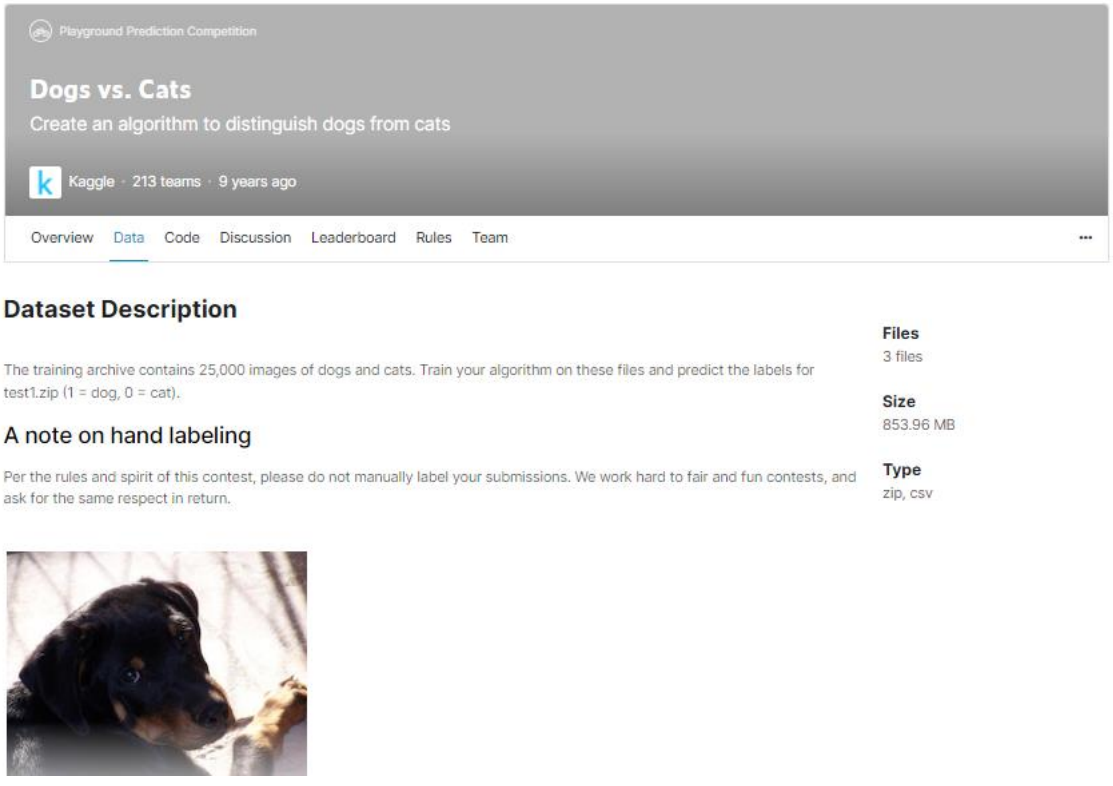
Pada percobaan simple neural network dengan menggunakan python dan keras, metode yang digunakan pada percobaan ini adalah feedforward neural networks. Feedforward Neural Network adalah jaringan saraf tiruan yang koneksi antar node-nya tidak membentuk sebuah siklus. Jaringan saraf Feedforward hanya memungkinkan sinyal untuk melakukan perjalanan melalui satu jalur saja, yakni dari input ke output. Tidak ada koneksi umpan balik dari output ke dirinya sendiri (loop).

Dataset yang digunakan yaitu Dogs vs. Cats yang diambil dari Kaggle yaitu : https://www.kaggle.com/c/dogs-vs-cats/data. Adapun tujuan dari percobaan ini adalah untuk mengklasifikasikan dengan benar apakah gambar yang diberikan berisi anjing atau kucing atau gambar yang di input dapat ditebak dengan tepat sebagai anjing dan kucing.

Proses yang dilakukan dalam percobaan ini :

1. Download dataset

   Dataset yang didownload disini yaitu data train.zip dan test.zip nya pada link Kaggle sebelumnya.



Pada data train terdapat 25.000 gambar yang berupa anjing dan kucing dan pada data test nya juga terdapat sebanyak 12.500 gambar yang berupa anjing dan kucing.

2. Menginstall library yang dibutuhkan

Adapun library yang diinstall pada percobaan ini yaitu tenserflow dan keras. Dengan menggunakan perintah `pip install keras` dan `pip install tenserflow` pada cmd.

3. Proses Training Dataset

Pada tahap selanjutnya setelah menginstall library yang dibutuhkan, maka akan dibuat dua file script python yaitu `simple_neural_network.py` dan `test_network.py`. Pada file simple_neural_network berisi code yaitu :

```
# import the necessary packages
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Activation
from keras.optimizers import SGD
from keras.layers import Dense
from keras.utils import np_utils
from imutils import paths
import numpy as np
import argparse
import cv2
import os
```

Dapat dilihat dari gambar diatas diimport library package python yang diperlukan, digunakan juga sejumlah implementasi dari scikit-learn dengan keras layers dan fungsi aktivasi nya.

```
def image_to_feature_vector(image, size=(32, 32)):
    # resize the image to a fixed size, then flatten the image into
    # a list of raw pixel intensities
    return cv2.resize(image, size).flatten()
```

Disini akan diubah ukuran gambar kedalam bentuk yang tetap, untuk memastikan setiap gambar dalam kumpulan data input memiliki ukuran “feature vector” yang sama. Karena saat menggunakan neural network gambar harus direpresentasikan dalam bentuk vector. Untuk ukuran resize yaitu 32 x 32 piksel.

```
22 # construct the argument parse and parse the arguments
23 ap = argparse.ArgumentParser()
24 ap.add_argument("-d", "--dataset", required=True,
25                 help="path to input dataset")
26 ap.add_argument("-m", "--model", required=True,
27                 help="path to output model file")
28 args = vars(ap.parse_args())
29
30 # grab the list of images that we'll be describing
31 print("[INFO] describing images...")
32 imagePaths = list(paths.list_images(args["dataset"]))
33
34 # initialize the data matrix and labels list
35 data = []
36 labels = []
```

Selanjutnya hanya diperlukan satu argument yaitu --dataset untuk direktori input yang berisi gambar anjing dan kucing. Pada baris 32 akan menginisialisasi daftar nama dan label masing-masing kemudian di baris 35 dan 36 akan mengonversi gambar satu per satu menjadi vector dan memperbarui daftar data dan labelnya.

```
38 # loop over the input images
39 for (i, imagePath) in enumerate(imagePaths):
40     # load the image and extract the class label (assuming that our
41     # path as the format: /path/to/dataset/{class}.{image_num}.jpg
42     image = cv2.imread(imagePath)
43     label = imagePath.split(os.path.sep)[-1].split(".")[0]
44
45     # construct a feature vector raw pixel intensities, then update
46     # the data matrix and labels list
47     features = image_to_feature_vector(image)
48     data.append(features)
49     labels.append(label)
50
51     # show an update every 1,000 images
52     if i > 0 and i % 1000 == 0:
53         print("[INFO] processed {}/{}".format(i, len(imagePaths)))
```

Akan dilakukan terus perulangan dari gambar input dan di konversi kedalam bentuk vector, kemudian akan di tampilakan/print update dari prosesnya setiap 1000 gambar.

```
55 # encode the labels, converting them from strings to integers
56 le = LabelEncoder()
57 labels = le.fit_transform(labels)
58
59 # scale the input image pixels to the range [0, 1], then transform
60 # the labels into vectors in the range [0, num_classes] -- this
61 # generates a vector for each label where the index of the label
62 # is set to `1` and all other entries to `0`
63 data = np.array(data) / 255.0
64 labels = np_utils.to_categorical(labels, 2)
65
66 # partition the data into training and testing splits, using 75%
67 # of the data for training and the remaining 25% for testing
68 print("[INFO] constructing training/testing split...")
69 (trainData, testData, trainLabels, testLabels) = train_test_split(
70     data, labels, test_size=0.25, random_state=42)
```

Pada baris 56 dan 57 setiap gambar yang sudah memiliki label akan di convert ke dalam bentuk angka (int). Gambar akan di bagi menjadi 2 kategori label yaitu 0 dan 1, 0 untuk kucing dan 1 untuk anjing. Kemudian pada baris 69 dan 70 akan di split untuk memisahkan antara data training dan testing. Pada percobaan ini akan digunakan 75% data untuk training dan 25% nya lagi untuk proses data testing nya.

```
72 # define the architecture of the network
73 model = Sequential()
74 model.add(Dense(768, input_dim=3072, init="uniform", activation="relu"))
75 model.add(Dense(768, input_dim=3072, activation="relu"))
76 model.add(Dense(384, activation="relu", kernel_initializer="uniform"))
77 model.add(Dense(2))
78 model.add(Activation("softmax"))
```

Dari baris 73-78 akan dibangun model neural network yaitu algoritma feedforward neural network 3072-768-384-2. Dimana input layer nya sebanyak 3072 node, 2 hidden layer dengan masing-masing sebanyak 768 dan 384 node dan output layer memiliki 2 node yaitu satu untuk setiap anjing dan kucing sebagai label. Adapun fungsi aktivasi sofmax untuk memberi nilai probabilitas pada label output nantinya.

```
80 # train the model using SGD
81 print("[INFO] compiling model...")
82 sgd = SGD(lr=0.01)
83 model.compile(loss="binary_crossentropy", optimizer=sgd, metrics=
   ["accuracy"])
84 model.fit(trainData, trainLabels, epochs=50, batch_size=128, verbose=1)
```

Untuk melatih model yang akan dibuat maka akan disetel nilai parameter laju pembelajaran SGD ke 0.01 serta digunakan binary_crossentropy loss function untuk meminimalkan loss, yaitu semakin kecil loss semakin baik modelnya. Model yang

sempurna memiliki cross-entropy loss 0. Metode ini biasanya berfungsi untuk klasifikasi multi-kelas dan multi-label, tetapi karena pada percobaan ini hanya memiliki 2 label class maka akan menggunakan binary cross-entropy. Untuk nilai epoch = 50, batch_size = 128 dan verbose =1.

```
86 # show the accuracy on the testing set
87 print("[INFO] evaluating on testing set...")
88 (loss, accuracy) = model.evaluate(testData, testLabels, batch_size=128,
   verbose=1)
89 print("[INFO] loss={:.4f}, accuracy: {:.4f}%".format(loss, accuracy * 100))
90
91 # dump the network architecture and weights to file
92 print("[INFO] dumping architecture and weights to file...")
93 model.save(args["model"])
```

Maka akan di print untuk nilai loss dan accuracy dari setiap gambar.

```
[INFO] processed 1000/25000
[INFO] processed 2000/25000
[INFO] processed 3000/25000
[INFO] processed 4000/25000
[INFO] processed 5000/25000
[INFO] processed 6000/25000
[INFO] processed 7000/25000
[INFO] processed 8000/25000
[INFO] processed 9000/25000
[INFO] processed 10000/25000
[INFO] processed 11000/25000
[INFO] processed 12000/25000
[INFO] processed 13000/25000
[INFO] processed 14000/25000
[INFO] processed 15000/25000
[INFO] processed 16000/25000
[INFO] processed 17000/25000
[INFO] processed 18000/25000
[INFO] processed 19000/25000
[INFO] processed 20000/25000
[INFO] processed 21000/25000
[INFO] processed 22000/25000
[INFO] processed 23000/25000
[INFO] processed 24000/25000
[INFO] constructing training/testing split...
```

4. Proses Testing Dataset

```
# import the necessary packages
from __future__ import print_function
from keras.models import load_model
from imutils import paths
import numpy as np
import argparse
import imutils
import cv2


def image_to_feature_vector(image, size=(32, 32)):
    # resize the image to a fixed size, then flatten the image into
    # a list of raw pixel intensities
    return cv2.resize(image, size).flatten()


# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-m", "--model", required=True,
                help="path to output model file")
ap.add_argument("-t", "--test-images", required=True,
                help="path to the directory of testing images")
ap.add_argument("-b", "--batch-size", type=int, default=32,
                help="size of mini-batches passed to network")
args = vars(ap.parse_args())
```

Untuk script python test_network memiliki 3 argumen baris perintah yang diberikan pada saat runtime. --model yaitu path ke file model serial. --test-images yaitu untuk direktori gambar test/uji. --batch-size yaitu ukuran default minimum batch yaitu 32.

```
# initialize the class labels for the Kaggle dogs vs cats dataset
CLASSES = ["cat", "dog"]

# load the network
print("[INFO] loading network architecture and weights...")
model = load_model(args["model"])
print("[INFO] testing on images in {}".format(args["test_images"]))
```

Pada baris 28 akan dibuat 2 nama daftar kelas yaitu cat dan dog. Baris 32 variabel model akan memuat model ke dalam memori untuk memudahkan dalam mengklasifikasikan gambar nantinya.

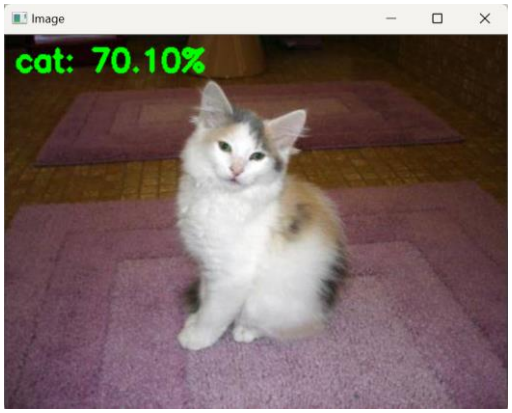
```
35 # loop over our testing images
36 for imagePath in paths.list_images(args["test_images"]):
37     # load the image, resize it to a fixed 32 x 32 pixels (ignoring
38     # aspect ratio), and then extract features from it
39     print("[INFO] classifying {}".format(
40         imagePath[imagePath.rfind("/") + 1:]))
41     image = cv2.imread(imagePath)
42     features = image_to_feature_vector(image) / 255.0
43     features = np.array([features])
```

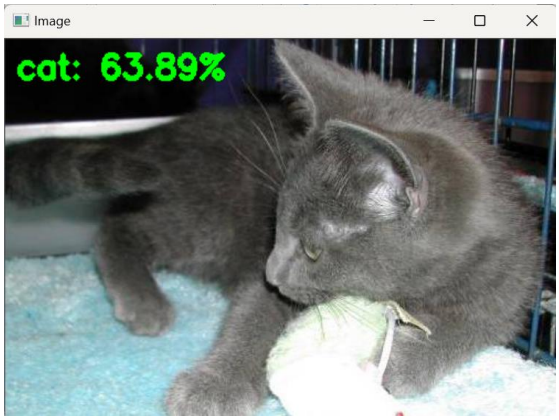
```
45 # classify the image using our extracted features and pre-trained
46     # neural network
47     probs = model.predict(features)[0]
48     prediction = probs.argmax(axis=0)
49     # draw the class and probability on the test image and display it
50     # to our screen
51     label = "{}: {:.2f}%".format(CLASSES[prediction],
52                                  probs[prediction] * 100)
53     cv2.putText(image, label, (10, 35), cv2.FONT_HERSHEY_SIMPLEX,
54                 1.0, (0, 255, 0), 3)
55     cv2.imshow("Image", image)
56     cv2.waitKey(0)
```

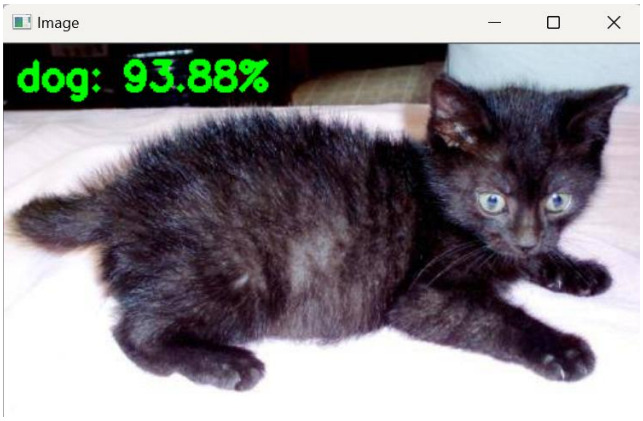
Pada baris 36 akan dilakukan perulangan pada gambar yang akan ditesting. Baris 47 dan 48 yaitu untuk memprediksi gambar yang di test. Baris 51-56 akan menghasilkan table tampilan yang berisi nama class dan skor probabilitasnya.

```
 10/147 [=>............................] - ETA: 3s - loss: 0.6952 - accuracy: 0.5039
 13/147 [=>............................] - ETA: 2s - loss: 0.6946 - accuracy: 0.5078
 16/147 [==>...........................] - ETA: 2s - loss: 0.6941 - accuracy: 0.5156
 19/147 [==>...........................] - ETA: 2s - loss: 0.6941 - accuracy: 0.5164
 22/147 [===>..........................] - ETA: 2s - loss: 0.6937 - accuracy: 0.5160
 25/147 [====>.........................] - ETA: 2s - loss: 0.6930 - accuracy: 0.5206
 28/147 [====>.........................] - ETA: 2s - loss: 0.6935 - accuracy: 0.5181
 31/147 [=====>........................] - ETA: 2s - loss: 0.6927 - accuracy: 0.5224
 34/147 [=====>........................] - ETA: 2s - loss: 0.6921 - accuracy: 0.5237
 37/147 [======>.......................] - ETA: 2s - loss: 0.6918 - accuracy: 0.5262
 40/147 [=======>......................] - ETA: 2s - loss: 0.6918 - accuracy: 0.5271
```

5. Hasil Testing







Dari gambar diatas dapat dilihat bahwa model yang telah dibangun dapat menebak gambar tersebut dengan benar bahwa mana kucing dan mana anjing dengan nilai probabilitasnya masing-masing. Akan tetapi, masih ada beberapa gambar lainnya yang salah ditebak/belum tepat, dimana gambar anjing salah ditebak sebagai kucing dan kucing salah ditebak sebagai anjing. Oleh karena itu, dari model yang telah dibangun ini masih harus diperbaiki agar model yang dibangun menjadi lebih baik.