Sieci neuronowe i sztuczna inteligencja – laboratorium 8

Monika Błyszcz, 236623

Kod:

```python
# import os utilities
import os

# import numpy
import numpy as np

# images
import skimage
from skimage import transform
from skimage.color import rgb2gray

# Import the `pyplot` module
import matplotlib.pyplot as plt


# function to load data
def load_data(data_directory):
    """Loads sign images data from their folder.

    Returns:
        images: list of images, i.e., signs
        labels: list of labels, i.e., signs IDs
    """
    # We need back labels and the row images
    images = []
    labels = []

    # We have one folder per sign type
    directories = []
    for d in os.listdir(data_directory):
        if os.path.isdir(os.path.join(data_directory, d)):
            directories.append(d)

    # In each foder there are not only images but also csv description
    # files
    for d in directories:
        label_directory = os.path.join(data_directory, d)
        file_names = [
            os.path.join(label_directory, f)
            for f in os.listdir(label_directory)
            if f.endswith(".ppm")
        ]

        for f in file_names:
            images.append(skimage.io.imread(f))
```

```python
                labels.append(int(d))

    return images, labels


ROOT_PATH = os.getcwd()

# Download training data

train_data_directory = os.path.join(ROOT_PATH, "Training")
test_data_directory = os.path.join(ROOT_PATH, "Testing")

images, labels = load_data(train_data_directory)
test_images, test_labels = load_data(test_data_directory)

print(labels)

## The following commented lines were reported in the DataCamp materials
## but they does not work here
print(images.ndim)
print(images.size)
images[0]
print(len(images))
print(len(labels))

# this should be a bar plot but an histogram with the same  number of
# bins that that unique levels of the labels list should be fine :-)
unique_labels = set(labels)
n_labels = max(unique_labels) + 1

# Make a histogram with 62 bins of the `labels` data
plt.hist(labels, n_labels)

# Show the plot
plt.show()

# Determine the (random) indexes of the images that you want to see
traffic_signs = [300, 2250, 3650, 4000]

# Fill out the subplots with the random images that you defined
for i in range(len(traffic_signs)):
```

```
    plt.subplot(1, 4, i + 1)
    plt.axis('off')
    plt.imshow(images[traffic_signs[i]])
    plt.subplots_adjust(wspace=0.5)
    plt.show()
    print(
        "shape: {0}, min value: {1}, max value: {2}".format(
            images[traffic_signs[i]].shape,
            images[traffic_signs[i]].min(),
            images[traffic_signs[i]].max()
        )
    )

# Plot a grid with a sample of all the signs
plt.figure(figsize=(15, 15))

i = 1

for label in unique_labels:
    # pick the first image for the label.
    #
    # The index() method searches an element in the list and returns its
    # index. In simple terms, index() method finds the given element in
    # a list and returns its position. However, if the same element is
    # present more than once, index() method returns its smallest/first
    # position.
    image = images[labels.index(label)]

    # We have 62 images. Hence, define a 64 grid sub-plots
    plt.subplot(8, 8, i)

    # Don't include axes
    plt.axis('off')

    # Add a title to each subplot
    #
    # The count() method returns the number of elements with the
    # specified value.
    plt.title("Label {0} ({1})".format(label, labels.
```

```
count(label)))

    # Add 1 to the counter
    i += 1

    # Plot this first image
    plt.imshow(image)

plt.show()

# To tackle the differing image sizes, you’re going to rescale the images
images_28 = [
    transform.resize(image, (28, 28))
    for image in images
]

# Convert `images28` to an array
images_28 = np.array(images_28)

# Convert `images28` to grayscale
images_28 = rgb2gray(images_28)

for i in range(len(traffic_signs)):
    plt.subplot(1, 4, i + 1)
    plt.axis('off')
    plt.imshow(images_28[traffic_signs[i]], cmap="gray")
    plt.subplots_adjust(wspace=0.5)

plt.show()

# Test set
# Transform the images to 28 by 28 pixels
test_images_28 = [
    transform.resize(image, (28, 28))
    for image in test_images
]
# Convert to grayscale
test_images_28 = rgb2gray(np.array(test_images_28))

```

Komentarz:

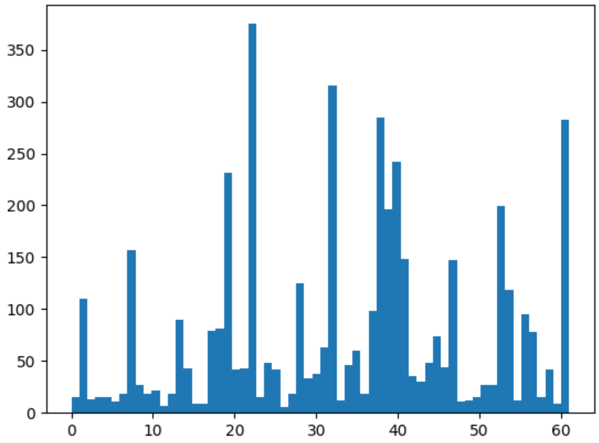
W trakcie wykonywania zadania, w pierwszym kroku wczytano dane testowe i dane treningowe. Uzyskano dzięki temu zbiór obrazów i ich indeksów (etykiet). Następnie utworzono histogram obrazujący ilość znaków danej kategorii. Potem wyszukano 4 losowe kategorie obrazów. Po czym wykorzystując cały zbiór pokazano po 1 przykładzie z każdej kategorii. Na końcu 4 obrazy z klas wybranych wcześniej ukazano w odcieniach szarości

Wyniki:

```
C:\Users\Mo\AppData\Local\Microsoft\WindowsApps\python3.10.exe C:\Users\Mo\Documents\Studia\SN_lab\SN\PWR_2023_SN\lab8\tensor.py
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
4575
4575
shape: (62, 61, 3), min value: 3, max value: 160
shape: (110, 96, 3), min value: 3, max value: 255
shape: (379, 153, 3), min value: 0, max value: 255
shape: (100, 68, 3), min value: 17, max value: 255

Process finished with exit code 0
```

Histogram:



Znaki:

Skala szarości: