Sprawozdanie z laboratorium - sieci neuronowe

Temat: Plant Seedlings Classification

Skład grupy: Julia Blicharska, Weronika Deleżuch, Aleksandra Zalewska

Opis zadania

Celem jest stworzenie klasyfikatora, który będzie rozpoznawał gatunek sadzonki na podstawie zdjęcia i przypisywał mu odpowiednią nazwę.

Dane

Danymi, nad którymi pracujemy jest zestaw treningowy i zestaw testowy zdjęć sadzonek roślin na różnych etapach rozwoju.

Każdy obraz ma nazwę pliku, która jest jego unikalnym identyfikatorem.

Zestaw danych obejmuje 12 gatunków roślin. Lista gatunków jest następująca:

- Black-grass (Konwalnik płaskopędowy)
- Charlock (Gorczyca polna)
- Cleavers (Przytulia czepna)
- Common Chickweed (Gwiazdnica pospolita)
- Common wheat (Pszenica zwyczajna)
- Fat Hen (Komosa biała)
- Loose Silky-bent (Miotła zbożowa)
- Maize (Kukurydza zwyczajna)
- Scentless Mayweed (Maruna nadmorska)
- Shepherds Purse (Tasznik pospolity)
- Small-flowered Cranesbill (Bodziszek drobny)
- Sugar beet (Burak cukrowy)

Zobaczmy kilka przykładowych zdjęć roślin.

Charlock

Cleavers

Common wheat

Maize

Niestety nie wszystkie zdjęcia są tak dobrej jakości. Zobaczmy po 2 zdjęcia – jedno dobrej i jedno złej jakości roślin z gatunków Scentless Mayweed oraz Shepherds Purse.



Jak widać na powyższych zdjęciach, niektóre gatunki sadzonek mogą być ciężkie do rozróżnienia dla człowieka, ponieważ obrazy często bywają słabej jakości lub kilka gatunków może być do siebie bardzo podobnych i ciężko byłoby nam je rozróżnić i odpowiednio sklasyfikować.

Źródła danych

Dane pochodzą z platformy Kaggle. Są ogólnodostępne dla każdego po utworzeniu konta na Kaggle. Link do strony z danymi: https://www.kaggle.com/c/plant-seedlings-classification/data .

Analiza danych

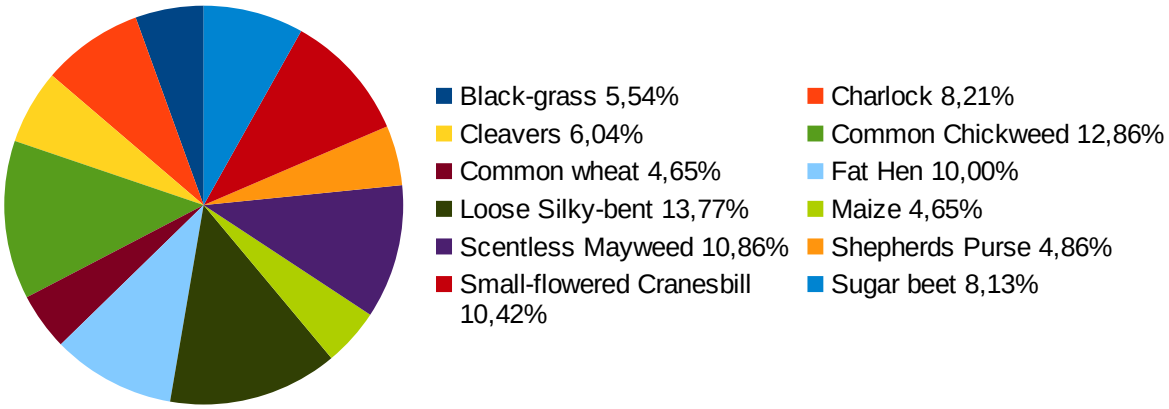
Dane treningowe

Mamy 4750 zdjęć sadzonek, które są podzielone na foldery pod względem gatunków.

Ilość zdjęć w każdym z folderów:

- Black-grass (Konwalnik płaskopędowy) - 263
- Charlock (Gorczyca polna) - 390
- Cleavers (Przytulia czepna) - 287
- Common Chickweed (Gwiazdnica pospolita) - 611
- Common wheat (Pszenica zwyczajna) - 221
- Fat Hen (Komosa biała) - 475
- Loose Silky-bent (Miotła zbożowa) - 654
- Maize (Kukurydza zwyczajna) - 221
- Scentless Mayweed (Maruna nadmorska) - 516
- Shepherds Purse (Tasznik pospolity) - 231
- Small-flowered Cranesbill (Bodziszek drobny) - 495
- Sugar beet (Burak cukrowy) – 386

Zobaczmy udział procentowy każdego z gatunków w naszych danych treningowych:



Dane testowe

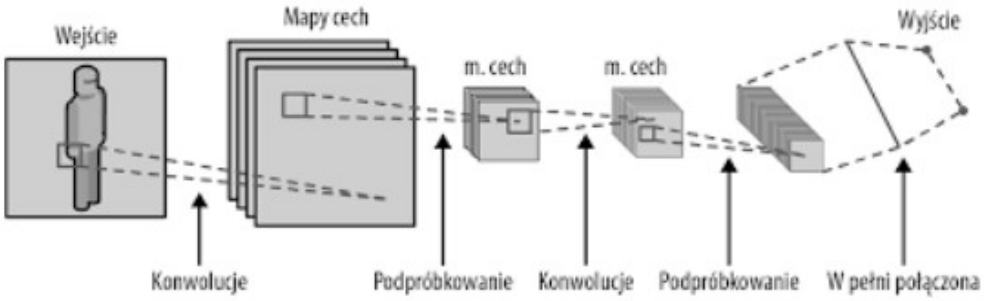
W tej grupie znajduje się 794 zdjęć. Będziemy na nich sprawdzać czy nasz model został dobrze wytrenowany i czy sieć działa poprawnie.

Przykładowe zdjęcia ze zbioru danych testowych:

Model

Do naszego zadania wybrałyśmy sieć konwolucyjną - CNN. Jest to struktura, która bardzo dobrze sprawdza się w zadaniach dotyczących rozpoznawania i klasyfikacji obrazów. Dzieje się tak, ponieważ głębokie sieci konwolucyjne (CNN) potrafią stopniowo filtrować różne części danych uczących i wyostrzać ważne cechy w procesie dyskryminacji wykorzystanym do rozpoznawania lub klasyfikacji wzorców. Pozwala to na znajdowanie najważniejszych elementów na danym zdjęciu sadzonki i dobrym dopasowaniu gatunku rośliny.

Model przedstawia się następująco:



Opis naszego kodu:

```
# Na tym etapie importujemy wykorzystywane później biblioteki.
# Są to m.in. elementy sieci neuronowej (Klasy warstw, optimizerów, funkcje uczące itd), a także
# funkcje transformujące obrazki (array_to_img, img_to_array, load_img)
# oraz biblioteki dostarczające operacji numerycznych (numpy)

import numpy as np
import pandas as pd


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers.core import Activation, Flatten, Dense
from keras.layers import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
import numpy as np
import random
import os
import sys
import cv2
from keras.utils import to_categorical
import matplotlib.pyplot as plt


from subprocess import check_output
```

```python
# Tutaj przyporządkowywujemy nazwy klas do wartości liczbowych.
# Jako, że sieć neuronowa jako klasy przyjmuje liczby, a człowiek preferuje operowanie na nazwach
# to dzięki powyższemu przyporządkowaniu obie strony są w stanie zrozumieć dane testowe oraz wyniki.

def classes_to_int(label):
    label = label.strip()
    if label == "Black-grass":  return 0
    if label == "Charlock":  return 1
    if label == "Cleavers":  return 2
    if label == "Common Chickweed":  return 3
    if label == "Common wheat":  return 4
    if label == "Fat Hen":  return 5
    if label == "Loose Silky-bent": return 6
    if label == "Maize":  return 7
    if label == "Scentless Mayweed": return 8
    if label == "Shepherds Purse": return 9
    if label == "Small-flowered Cranesbill": return 10
    if label == "Sugar beet": return 11
    print("Invalid Label", label)
    return 12
```

```python
def int_to_classes(i):
    if i == 0: return "Black-grass"
    elif i == 1: return "Charlock"
    elif i == 2: return "Cleavers"
    elif i == 3: return "Common Chickweed"
    elif i == 4: return "Common wheat"
    elif i == 5: return "Fat Hen"
    elif i == 6: return "Loose Silky-bent"
    elif i == 7: return "Maize"
    elif i == 8: return "Scentless Mayweed"
    elif i == 9: return "Shepherds Purse"
    elif i == 10: return "Small-flowered Cranesbill"
    elif i == 11: return "Sugar beet"
    print("Invalid class ", i)
    return "Invalid Class"
```

```python
# Nadajemy stałe wartości

TEST_DIR = "../input/plant-seedlings-classification/test/"
NUM_CLASSES = 12   # Ilość klas
WIDTH = 128   # Wymiary obrazku
HEIGHT = 128   # Wymiary obrazku
DEPTH = 3   # Głębia
INPUT_SHAPE = (WIDTH, HEIGHT, DEPTH)   # Forma danych wejściowych
EPOCHS = 25    # Liczba epok uczenia
INIT_LR = 1e-3   # Initial learnig rate - parametr wejściowy optimizera Adam
BATCH_SIZE = 32   # Liczba danych testowych wykorzystywna przy pojedynczym uczeniu sieci
```

```
# Poniższa funkcja odczytuje dane testowe z plików we wskazanym katalogu (trainDir).
# TrainDir zawiera foldery, gdzie każdy folder to pojedyncza klasa (odpowiednia nazwa folderu wskazująca na klasę).
# Następnie dane są przetwarzane do macierzy w odpowiednim kształcie i zmieniane są wymiary tak by pasowały do tej
# ustalonej w stałych.
# Funkcja zwraca liste macierzy (tablic wielowymiarowych) reprezentujących dany obrazek oraz drugą liste,
# gdzie znajdziemy odpowiadające im klasy

def readTrainData(trainDir):
    data = []
    labels = []
    dirs = os.listdir(trainDir)
    for directory in dirs:
        absDirPath = os.path.join(trainDir, directory)
        images = os.listdir(absDirPath)
        for imageFileName in images:
            imageFullPath = os.path.join(trainDir, directory, imageFileName)
            img = load_img(imageFullPath)
            arr = img_to_array(img)  # Numpy array with shape (233,233,3)
            arr = cv2.resize(arr, (HEIGHT,WIDTH)) #Numpy array with shape (HEIGHT, WIDTH,3)
            #print(arr.shape)
            data.append(arr)
            label = classes_to_int(directory)
            labels.append(label)
    return data, labels
```

```
# Poniższa funkcja tworzy strukture modelu, który będzie wykorzystywany w sieci neruonowej

def createModel():
    model = Sequential()
    model.add(Conv2D(32, (3,3), padding="same", input_shape=INPUT_SHAPE))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Conv2D(64, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Conv2D(128, (3,3), padding="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Flatten())
    model.add(Dense(units=500))
    model.add(Activation("relu"))
    model.add(Dense(units=12))
    model.add(Activation("softmax"))
    opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
    model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=["accuracy"])
    return model
```

```
# Jako optimizer przyjęto algorytm Adam, a jako metrykę przyjęto dokładność predykcji.
# Powyższy kod odczytuje dane trenujące z użyciem wcześniej zdefiniowanej funkcji
# Z danych tworzone są tablice wielowymiarowe
# Następnie wykorzystuje się funkcje train_test_split dostępną w bibliotece Keras, która
# w losowy sposób rozdziela dane uczące tak, aby wydzielić z nich część danych, które będą wykorzystywane do
# testowania postępów uczenia.

X, Y = readTrainData("../input/plant-seedlings-classification/train/")
X = np.array(X, dtype="float") / 255.0
Y = np.array(Y)
Y =  to_categorical(Y, num_classes=12)

print("Parttition data into 75:25...")
(X_train, X_val, Y_train, Y_val) = train_test_split(X,Y,test_size=0.25, random_state=25)
```

```
Parttition data into 75:25...
```

```
# Z wykorzystaniem wcześniej zdefiniowanej funkcji tworzymy model, który zostanie następnie wykorzystany

model = createModel()
```

```
# Tworzony jest obiekt klasy ImageDataGenerator, który pozwala na tworzenie wiele transformacji obrazków
# takich jak rotacja, przybliżenie, odbicie horyzontlane. Sieć wyuczona takimi danymi będzie bardziej elastyczna.

aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1, \
    height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,\
    horizontal_flip=True, fill_mode="nearest")
```

```
# Sieć neuronowa jest uczona danymi wygenerowanymi z użyciem wcześniej opisanej klasy. Liczba epok i wielkość
# partii danych
# ustalona jest w stałych

history = model.fit_generator(aug.flow(X_train, Y_train, batch_size=BATCH_SIZE), \
    validation_data=(X_val, Y_val), \
    steps_per_epoch=len(X_train) // BATCH_SIZE, epochs=EPOCHS, verbose=1)
```

```
Epoch 23/25
111/111 [==============================] - 97s 877ms/step - loss: 0.4297 - accuracy: 0.8482 - val_loss: 0.3790 - val_
accuracy: 0.8603
Epoch 24/25
111/111 [==============================] - 97s 873ms/step - loss: 0.4118 - accuracy: 0.8518 - val_loss: 0.4125 - val_
accuracy: 0.8561
Epoch 25/25
111/111 [==============================] - 97s 878ms/step - loss: 0.3991 - accuracy: 0.8530 - val_loss: 0.4007 - val_
accuracy: 0.8527
```

```
 #Jak widać wyżej, nasz model jest dokładny w około 85%.
```

```
%matplotlib inline
```

```
# Tworzony jest wykres parametrów jakościowych (takich jak dokładność predykcji) względem wykonywanej epoki
# Pozwoli to nam na zbadanie jaka liczba epok jest optymalna i jak wpływa zwiększenie liczby epok uczenia na
# dokładność sieci

import matplotlib
matplotlib.use("Agg")
plt.style.use("ggplot")
plt.figure(figsize=(20,12)),
plt.plot(np.arange(0, EPOCHS), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, EPOCHS), history.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, EPOCHS), history.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, EPOCHS), history.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy on  crop classification")
plt.xlabel("Epoch Number")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("foo.png")
plt.show()
```

```python
# Ta funkcja nie jest nigdzie wykorzystywana,
# ale pozwala na podstawie nazwy pliku wczytać dany plik obrazka (do odpowiedniej formy obsługiwanej przez sieć),
# następnie wykorzystać sieć neuronową do predykcji klasy danego obrazka.

def predict_image_name(image_name):
    image_path = os.path.join(TEST_DIR, image_name)
    img = load_img(image_path)
    img_arr = img_to_array(img)
    img_arr = cv2.resize(img_a, (HEIGHT, WIDTH))
    if len(x.shape) == 3:
        img_arr = np.expand_dims(img_a, axis=0)
    prediction = model.predict(img_a)
    idx = np.argmax(prediction)
    return int_to_classes(idx)
```

```python
# Funkcja ta przewiduje klasę danego obrazka (najbardziej prawdopodobna klasa)
# i zwraca ją w formie czytelnej dla człowieka (tekstowej)
# Zwróćmy uwagę, że obrazek musi być wcześniej przetworzony do formy macierzy, która jest
# ustalona jako wejściowa dla sieci.

def predict_image_array(x):
    if len(x.shape) == 3:
        x = np.expand_dims(x, axis=0)
    prediction = model.predict(x)
    idx = np.argmax(prediction)
    return int_to_classes(idx)
```

```python
# Wywołuje wcześniej wywołaną funkcje dla obrazka o indeksie 12 z listy X_train,
# a następnie wyświetla przewidywaną klasę.

prediction = predict_image_array(X_train[12])
prediction
```

```
'Loose Silky-bent'
```

```python
# Działa to też dla obrazków o innych indeksach.
prediction = predict_image_array(X_train[77])
prediction
```

```
'Fat Hen'
```

```python
prediction = predict_image_array(X_train[5])
prediction
```

```
'Black-grass'
```

```python
prediction = predict_image_array(X_train[100])
prediction
```

```
'Common Chickweed'
```

Podsumowanie

Udało nam się stworzyć model, który rozpoznaje gatunek sadzonki widocznej na zdjęciu i podaje jej nazwę. Jest nauczony rozpoznawać aż 12 gatunków roślin na różnych etapach rozwoju. Po przetestowaniu modelu mamy pewność, że jest on dokładny i nie powinien robić wielu błędów przy klasyfikacji roślin. Rozpozna on odpowiedni gatunek rośliny z prawdopodobieństwem 85%.