Analiza Obrazów
Odróżnianie określonych gatunków pająków z wykorzystaniem sieci neuronowej – projekt.

Grudzień 2022

1. Opis projektu.
   Projekt dotyczy wykorzystania sieci neuronowej z podbiblioteki Keras, zaczerpniętej z biblioteki Tensorflow. Po wytrenowaniu sieci jest ona wykorzystywana do odróżniania trzech gatunków pajęczaków: Deinopis sp. (pająk gladiator), Brachypelma Hamorii (ptasznik czerwono-kolanowy) oraz Maratus sp. (Peacock spider).
2. Założenia wstępne.
   Do realizacji projektu wykorzystano wersję Python'a 3.10 oraz biblioteki:

```
matplotlib==3.5.2,
numpy==1.22.4,
opencv_python==4.6.0.66,
PySimpleGUI==4.60.4,
tensorflow==2.10.1
```

   Językiem wykorzystywanym w projekcie był python, ponieważ obecnie jest to język najczęściej asocjowany z sieciami neuronowymi.

3. Analiza projektu

Specyfikacja danych wejściowych:

Program pracuje na obrazach z rozszerzeniem jpg o dowolnym rozmiarze.
Zaleca się by obraz przedstawiał osobnika, z któregoś z powyżej wymienionych gatunków/rodzin.

Opis oczekiwanych danych wyjściowych:

W wyniku zuploadowania obrazu zostanie zwrócony przykładowy obraz przedstawiający pajęczaka z podpisem, który najbardziej oddaje(przypomina) to co znajduje się na wysłanym obrazie.

Definiowanie struktur danych:

W trakcie uczenia neuronu wykorzystywane są zdjęcia skalowane do tego samego rozmiaru, które następnie są zmieniane do postaci, na której neuron jest w stanie określić najwięcej wspólnych cech wykorzystywanych do rozpoznania pajęczaka na obrazie.

Specyfikacja interfejsu użytkownika:

Po uruchomieniu programu, pojawia się okienko, w którym widnieją różne opcje dostępne dla użytkownika.

Decyzja o wyborze narzędzi programistycznych:

Do napisania programu został wykorzystany ogólnodostępny edytor tekstowy Visual Studio Code, a językiem stosowanym był python. Korzystano z wielu bibliotek wymienionych w punkcie (2) w tym Tensorflow, Keras, SimplePyGui.

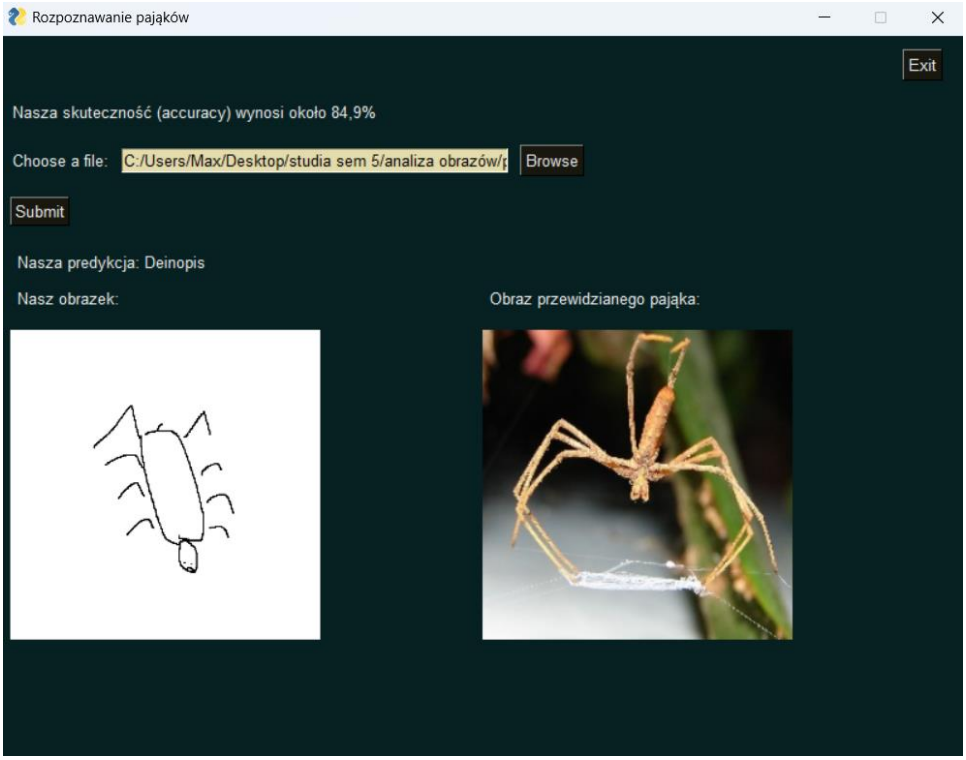
4. Podział pracy i analiza czasowa.
   Rozpoczęcie pracy wiązało się z wyborem tematu projektu w dniu 27.11.2022
   i skonfigurowaniem repozytorium na platformie GitHub.
   Przez następny tydzień autorzy projektu wspólnie budowali kolejne elementy programu.
   Kolejnym krokiem było wytrenowanie gotowego neuronu na bazie zdjęć pozyskanej z Internetu.
   Baza zdjęć wykorzystanych do trenowania została szczegółowo przeanalizowana przez Wojciecha Gwiazdę.
   Maksymilian Gibas stworzył UI.
   Michał Sienkiewicz napisał dokumentację, a Wojciech Gwiazda dodał do niej szatę graficzną.

5. Działanie programu.
   Po uruchomieniu wprowadzamy do programu zdjęcie pająka z jednej z trzech podanych kategorii (Deinopis (Ogre spider), Brachypelma hamorii (Red Knee tarantula), Maratus volans (Peacock spider)). Wprowadzony plik musi mieć rozszerzenie .jpg lub maksymalnie trzy warstwy, gdyż sieć nie obsługuje plików z kanałem alfa.

6. Testowanie
   a. Wgraliśmy zdjęcie o rozszerzeniu .png i program wyrzucił błąd, przez posiadanie 4 warstwy, ponieważ nasz program jej nie obsługuje.

```
ValueError: could not broadcast input array from shape (256,256,4) into shape (256,256,3)
```

   b. Sprawdziliśmy działanie rozpoznawania przez wgranie różnych obrazków (nie naszych pająków) i sprawdziliśmy uzyskane wyniki. Program zwraca coś co wg. naszej sieci jest najbardziej podobne do jednego z naszych trzech pająków. Np. narysowany w paint'cie "pająk" rozpoznaje jako Deinopis, przez jego cienkie nóżki.



Rys. 1: Predykcja naszej sieci na pająka Deinpois, poprzez największe podobieństwo według sieci

Rys. 2: Poprawnie rozpoznany przypadek Deinopisa

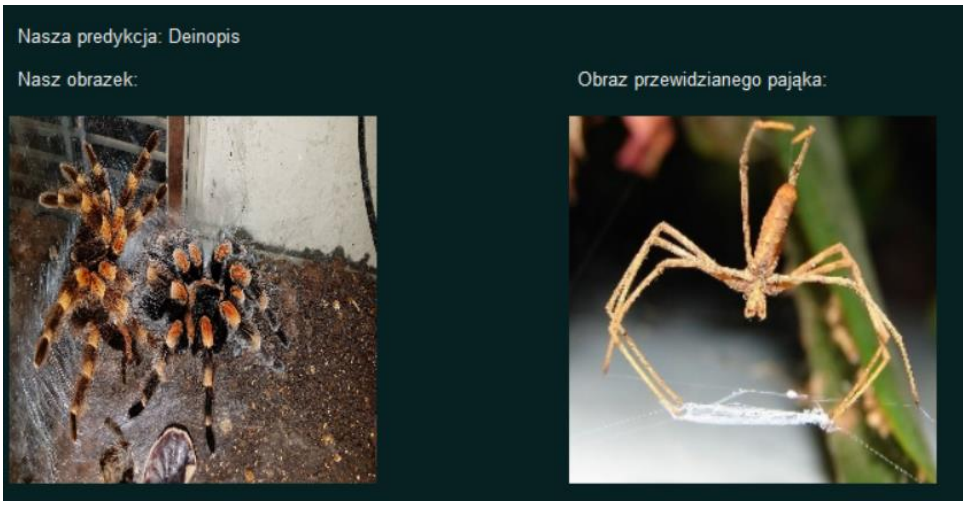
c. Czasami program jest w stanie poprawnie rozpoznać typ pająka, mimo że nie wygląda jak ten z obrazka, np. tutaj program rozpoznał poprawnie ptasznika, mimo że nie jest tą sama rodzina, ponieważ wgraliśmy ptasznika gatunku Typhoclaena seladonia, a otrzymaliśmy ptasznika gatunku Brachypelma hamorii.

W drugim przykładzie program poprawnie rozpoznał skakuna, mimo że tak jak wcześniej to nie są nawet spokrewnione ze sobą gatunki.

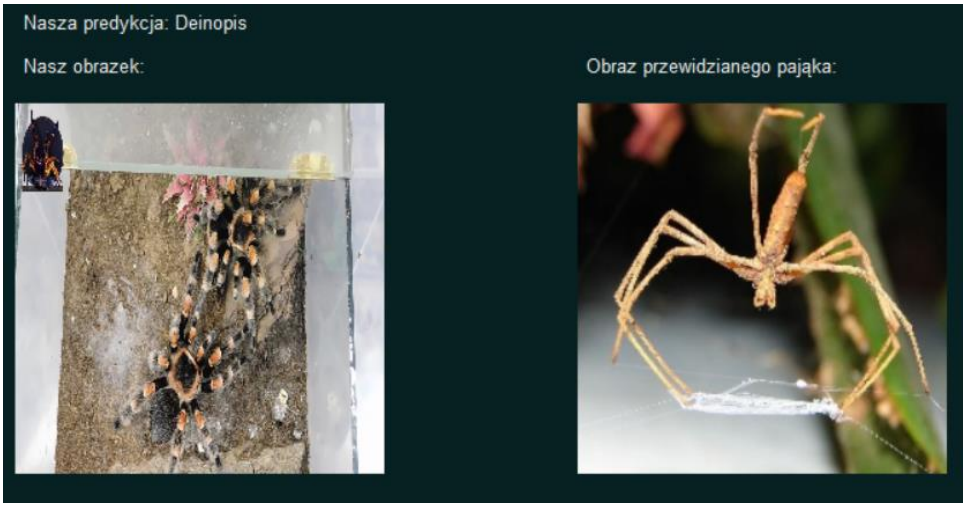
Rys. 1: Dwa ptaszniki różnych gatunków - na lewo T. seladonia, na prawo B. hamorii

Rys. 2: Dwa skakuny różnych gatunków - na lewo P. regius, na prawo Maratus unicup

d. Nasza sieć nie była trenowana na grupach, dlatego też jej zachowanie po wklejeniu zdjęcia zawierającego więcej niż jednego pająka nie jest zdefiniowane, chociaż w niektórych przypadkach (np. jak kolory oraz kształt pająków zgadzały się idealnie) sieć odgadywała poprawnie.



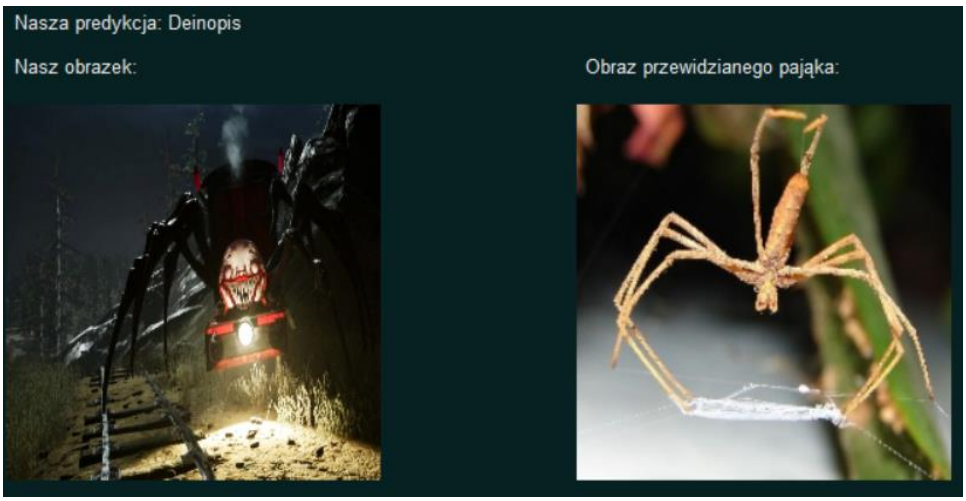
Rys. 1: Źle rozpoznane pająki w grupie (parze)

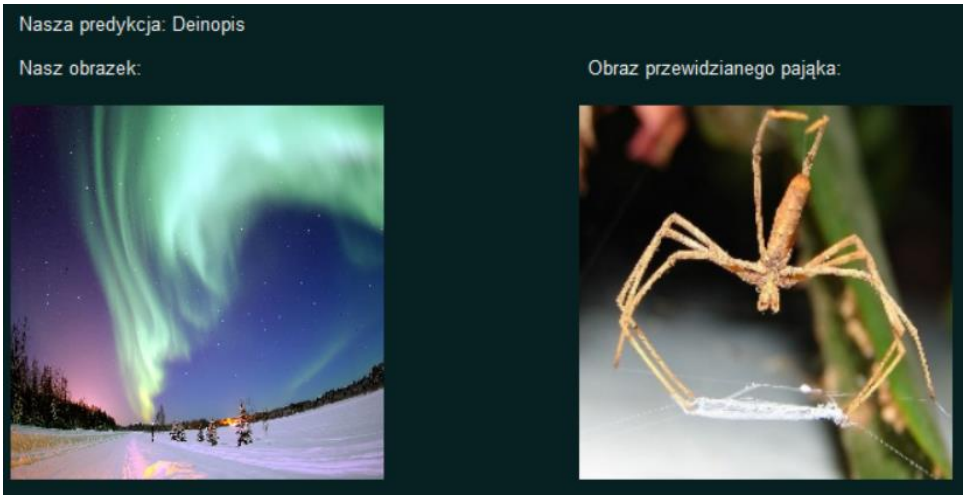


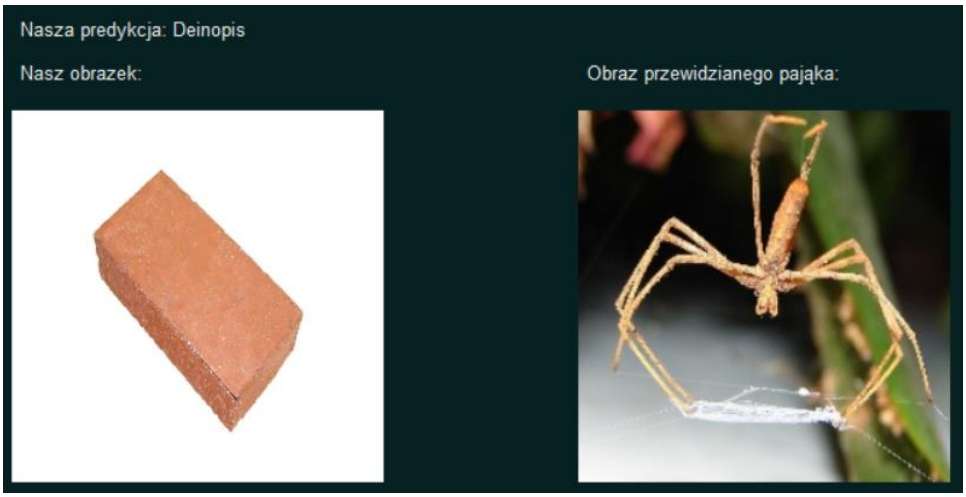
Rys. 2: Źle rozpoznane pająki w grupie (parze)

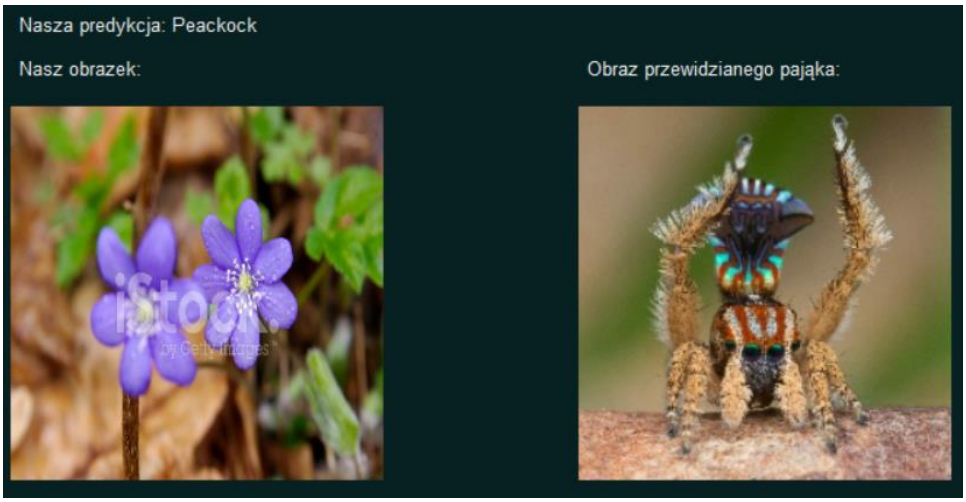
Rys. 3: Dobrze rozpoznane pająki w grupie (parze)

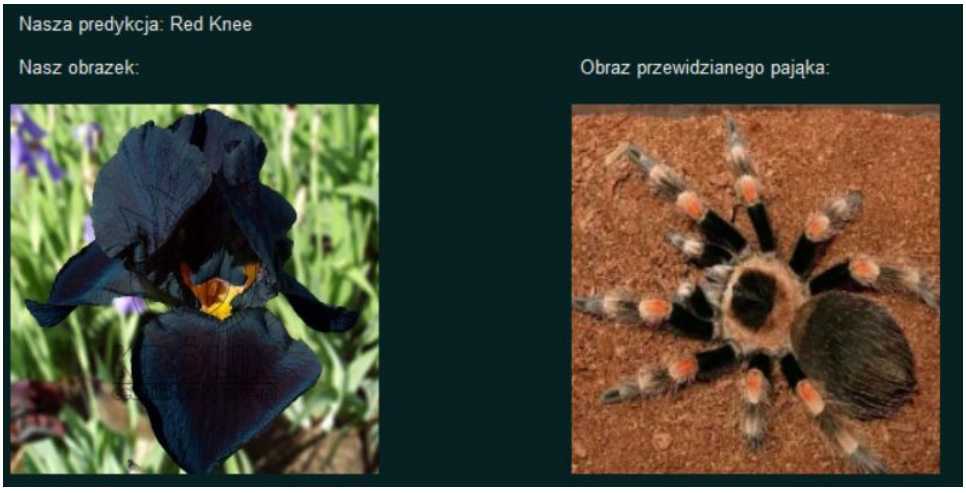
e. Sprawdziliśmy różne losowe zdjęcia i doszliśmy do wniosku, że najczęściej nasz program rozpoznaje je jako Deinopis'a, jednak gdy kolory bardziej pasują do Red Knee lub Peacock'a to wypisuje, że to one.











7. Problemy i co nie działa - zostało opisane w punkcie powyższym.

8. Źródła inspiracji wykorzystane w projekcie

Baza danych:

https://www.kaggle.com/datasets/gpiosenka/yikes-spiders-15-species

Początkowa inspiracja działania projektu:

https://thedatafrog.com/en/articles/dogs-vs-cats/

GUI:
https://www.pysimplegui.org/en/latest/cookbook/

Warstwy (augmentation):

https://machinelearningmastery.com/image-augmentation-with-keras-preprocessing-layers-and-tf-image/

Ogólne informacje o keras'ie:

https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_directory

Tworzenie pliku requirements:

https://stackoverflow.com/questions/57907655/how-to-use-pipreqs-to-create-requirements-txt-file

Wczytywanie pliku requirements:

https://stackoverflow.com/questions/7225900/how-can-i-install-packages-using-pip-according-to-the-requirements-txt-file-from

Testowanie działania na czystym środowisku (instalacja pliku requirements):

https://docs.python.org/3/library/venv.html

+ Stackoverflow oraz oficjalne dokumentacje dla poszczególnych problemów,

oraz losowe zdjęcia z google (np. ze sklepów).

9. Instrukcja

- `pip install -r requirements.txt (terminal musi być otwarty w naszym głównym folderze, gdzie znajduje się plik requirements.txt oraz gui.py)`
- Po zainstalowaniu bibliotek uruchamiamy nasz program przez wpisanie komendy *py gui.py* (jeśli znajdujemy się na windowsie oraz mamy wersję pythona 3.10)
- W naszym GUI widnieje parę przycisków:

- Możemy zakończyć nasz program wciskając *Exit*
- Wybieramy zdjęcie do rozpoznania przez kliknięcie *Browse* i zatwierdzamy
- Klikamy *Submit* aby otrzymać wynik, wraz z wczytanym oraz przewidzianym obrazkiem