Paweł BIEGUN[1], Krzysztof GRYBOŚ[2]

Opiekun naukowy: Sławomir HERMA[3]

PRÓBA OPRACOWANIA SYSTEMU INTERPRETACJI JĘZYKA MIGOWEGO W CZASIE RZECZYWISTYM

Streszczenie: Celem projektu było opracowanie systemu sieci neuronowych korzystając z bibliotek Tensorflow oraz Keras, który interpretował by obraz z kamery komputera. Wykonanie i implementacja modelu interpretującego znak pokazywany do kamery zakończyło się sukcesem. Nie udało się opracować modelu do predykcji czy za danym fragmentem tekstu powinna być spacja.

Słowa kluczowe: Tensorflow, Keras, machine learning, convoluted neural network

AN ATTEMPT ON CREATING A SYSTEM FOR INTERPRETING SIGN LANGUAGE IN REAL TIME

Summary: The goal of the project was to create a neural network using Tensorflow and Keras libraries for Python that interprets sign language shown to the computers camera. Creating and implementing the model for interpreting images was a success. An attempt to create a model for predicting whether there should be a space after a particular string failed.

Keywords: Tensorflow, Keras, machine learning, convoluted neural network

1. Cele programu

Zagadnieniem przewodnim przeprowadzonych prac było stworzenie systemu do rozpoznawania znaków amerykańskiego języka migowego w czasie rzeczywistym. W pracy przyjęto, że każdy gest w języku migowym reprezentuje jedną literę. W tym celu stworzono sieć neuronową do rozpoznawania pokazywanego znaku korzystając z rozwiązań oferowanych przez pakiety Tensorflow i Keras. Zagadnienie rozpoznawania znaków amerykańskiego języka migowego z pojedynczego zdjęcie było z sukcesem podejmowane przez wielu badaczy.

[1] V Liceum Ogólnokształcące Bielsko-Biała, biegunpawel900@gmail.com
[2] V Liceum Ogólnokształcące Bielsko-Biała, krzysiu.grybos@gmail.com
[3] dr. inż.,Katedra Inżynierii Produkcji, Wydział Budowy Maszyn i Infirmatyki, Akademia Techniczno-Humanistyczna, sherma@ath.bielsko.pl

2. Importowanie i formatowanie obrazu

Pobieranie obrazu

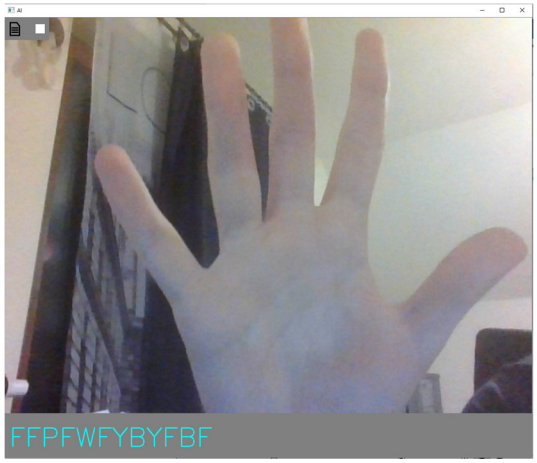
Pobieranie obrazu z kamery komputera zostało osiągnięte w języku Python za pomocą modułu OpenCV2.

Formatowanie obrazu

W każdej klatce działania programu pobierano obraz z aparatu. Zmieniano jego rozdzielczość do 1200 x 1000 pixeli, gdyż był to dogodny format do wyświetlenia obrazu w graficznym interfejsie użytkownika. Następnie obraz przekształcano w odcienie szarości. Obraz w odcieniach szarości był zmniejszony do rozdzielczości 28x28 pixeli, potrzebnej do działania sieci neuronowej.

3. Autorski graficzny interfejs użytkownika

Elementy interfejsu użytkownika



Zdjęcie 1. Screenshot działającego programu.

Graficzny interfejs użytkownika składał się z kolorowego obrazu pobranego z kamery komputera w rozdzielczości 1200 x 1000 pixeli. Na niego nakładane były dwa elementy. W prawym górnym rogu zamieszczono element wyświetlający czy pokazywane znaki są zapisywane do pliku czy też nie. Zmiana tej opcji mogła się odbyć poprzez naciśnięcie przez użytkownika klawisza ‘s’. Na dole interfejsu zamieszczono szary pasek, na którym wyświetlane są znaki rozpoznawane. Wszystkie elementy interfejsu zostały zaprojektowane za pomocą modułu OpenCV2. Symbol zapisu jest oddzielnym plikiem graficznym nakładanym na każdą klatkę. Poprawne zamknięcie programu wymaga naciśnięcia klawisza ‘q’.

4. Konsekwencje niepoprawnego zamknięcia programu kamery OpenCV2

Wykryte błędy

Zamknięcie programu w formie innej niż klawiszem ‘q’ mogło uniemożliwić użytkowanie aparatu w tym ponowne uruchomienie programu. Działo się tak gdyż, polecenie `cv2.VideoCapture(0)` rezerwowało do swojego użytku kamerę w systemie Windows. Niespodziewane wyjście z programu nie wywoływało funkcji `.release()` a więc aparat pozostawał zarezerwowany dla systemu Windows.

Rozwiązania problemów

Znane są dwa rozwiązania tego problemu. Pierwszym z nich jest wejście do Menedżera Urządzeń system operacyjnego, a następnie wyłączenie i ponowne włączenie aparatu komputera. Drugim sposobem jest wyłączenie komputera na nie mniej niż około 2 minuty tak aby pamięć systemowa RAM została wyczyszczona, a więc rezerwacja programu skasowana.

5. Tworzenie modelu do rozpoznawania pokazywanego znaku na podstawie zdjęcia

5.1 Wczytywanie i preprocessing danych treningowych

Celem modelu jest wykonanie predykcji dot. znaku jaki został przekazany do pliku przez program zbierający obraz z kamery opisany w poprzednich rozdziałach.

```
train      =      pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/sign_mnist_train.csv')
test       =      pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/sign_mnist_test.csv')
```

Dane zostają wczytane z formatu .csv jako *pandas dataframe*.

```
train_set = np.array(train, dtype = 'float32')
test_set = np.array(test, dtype='float32')
```

Forma danych zostaje zmieniona z *pandas dataframe* na *numpy array*.

```
class_names = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
'V', 'W', 'X', 'Y' ]
```

Następuje deklaracja nazw przewidywanych klas:

```
X_train = train_set[:, 1:] / 255
y_train = train_set[:, 0]
X_test = test_set[:, 1:] / 255
y_test = test_set[:,0]
```

Zmiana wartości opisujących nasycenie koloru z od 0 do 255 na od 0 do 1.

```
from sklearn.model_selection import train_test_split
X_train, X_validate, y_train, y_validate = train_test_split(X_train, y_train, test_size = 0.001, random_state = 12345)
```

Podział danych na dane do treningu modelu oraz dane do walidacji w trakcie treningu:

```
X_train = X_train.reshape(X_train.shape[0], *(28, 28, 1))
X_test = X_test.reshape(X_test.shape[0], *(28, 28, 1))
X_validate = X_validate.reshape(X_validate.shape[0], *(28, 28, 1))
```

Do danych zostaje dodany dodatkowy pusty wymiar aby mogły zostać użyte do treningu modelu.

5.2 Struktura prponowanego modelu.

```
cnn_model = Sequential()
cnn_model.add(Conv2D(32, (3, 3), input_shape = (28,28,1), activation='relu'))
cnn_model.add(Conv2D(64, (3, 3), input_shape = (28,28,1), activation='relu'))
cnn_model.add(Conv2D(128, (3, 3), input_shape = (28,28,1), activation='relu'))
cnn_model.add(Flatten())
cnn_model.add(Dense(units = 512, activation = 'relu'))
cnn_model.add(Dense(units = 25, activation = 'softmax'))
```

Warstwy konwolucyjne odnajdują charakterystyczne dla danego znaku wzorce, następne warstwy tworzą gęstą sieć neuronową interpretującą wzorce jakie zostały znalezione na obrazie.

5.3 Trening modelu

```
cnn_model.compile(loss ='sparse_categorical_crossentropy', optimizer='adam' ,metrics =['accuracy'])
```

Model jest kompilowany używając bardzo uniwersalnej funkcji *loss*, z optymalizatorem *Adam*.

```
history = cnn_model.fit(X_train, y_train epochs = 3, verbose = 1, validation_data = (X_validate, y_validate))
```

Trening odbywa się na 3 „epochach”, zastosowanie większej ilości ekspozycji modelu z tym zestawem danych skutkuje overfittingiem.

Nie zaobserwowano znaczących różnic w zastosowaniu innych funkcji *loss*, ilości „epochów”, dodania lub odejmowania warstw gęstych lub konwolucyjnych oraz warstw „dropout”. Podniesienie realnej skuteczności modelu powyżej ±1, 95% wydaje się być niemożliwe.

6. Próba stworzenia modelu do przewidywania czy za fragmentem tekstu powinna być spacja

Baza słownictwa służącego do treningu sieci neuronowej

Bazą słownictwa do treningu sieci neuronowej stanowią transkrypty wszystkich zebranych rozmów TED talk w języku angielskim. Stworzono listę słownictwa, które występuje w tych trankryptach. Stworzono 7.2 mln losowych wyrażeń o długości od 1 do 7 składających się z losowych liter i usunięto z nich pojedynczo występujące litery „a" oraz „i" ze względu na ich rolę w języku angielskim. Połączono słowa losowe oraz słowa, po których ma być spacja.

Model który został użyty

```
VOCAB_SIZE = len(vocab)
MAXLEN = 1
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(VOCAB_SIZE, 32),
    tf.keras.layers.Dense(1, activation="sigmoid")
])
model.compile(loss="binary_crossentropy",optimizer="adam",me
trics=['acc'])
history = model.fit(vocab, labels, epochs=10)
```

Błąd będący rezultatem działania modelu

```
InvalidArgumentError       Traceback (most recent call last)
InvalidArgumentError: 2 root error(s) found.
  (0) Invalid argument:  indices[1,0] = 6 is not in [0, 6)
     [[node         sequential_3/embedding_3/embedding_lookup
(defined at <ipython-input-7-fd60008e478c>:9) ]]
  (1) Invalid argument:  indices[1,0] = 6 is not in [0, 6)
     [[node         sequential_3/embedding_3/embedding_lookup
(defined at <ipython-input-7-fd60008e478c>:9) ]]
     [[sequential_3/embedding_3/embedding_lookup/_22]]
0 successful operations.
0            derived            errors            ignored.
[Op:__inference_train_function_5527]
Errors may have originated from an input operation.
Input     Source     operations     connected     to     node
sequential_3/embedding_3/embedding_lookup:
 sequential_3/embedding_3/embedding_lookup/5227  (defined  at
/usr/lib/python3.6/contextlib.py:81)
Input     Source     operations     connected     to     node
sequential_3/embedding_3/embedding_lookup:
 sequential_3/embedding_3/embedding_lookup/5227  (defined  at
/usr/lib/python3.6/contextlib.py:81)
Function call stack:
train_function -> train_function
```

7. Podsumowanie

Udało się częściowo osiągnąć cele projektu. Stworzono program pozwalający na rozpoznawanie pokazywanych znaków w czasie rzeczywistym a następnie zapisanie zinterpretowanych znaków do pliku. Dalsze prace badawcze należy nakierować na interpretację uzyskanych danych np. dodawanie przerw między słowami oraz znaków diakrytycznych. Skuteczność programu na danych z używanego „datasetu" wynosi ponad 95% i nie wydaje się możliwe znaczące podniesienie tej skuteczności. Potencjalnym kierunkiem rozwoju dla tego typu programów jest interpretacja obrazu 3D, ponieważ oferuje znacznie lepsze metody różnicowania pomiędzy niektórymi znakami i wykrycia obecności znaku. Oprogramowaniem open-source które wydaje się oferować wszystkie konieczne funkcje do odnalezienia nadgarstka osoby pokazującej znaki jest *openpose*. Należy w takim przypadku odnaleźć „dataset" danych 3D. Potencjalnym problemem są wtedy trudności w użytku i tworzeniu trójwymiarowego obrazu w życiu codziennym użytkownika.

LITERATURA

1. Kaggle *https://www.kaggle.com/datamunge/sign-language-mnist* 1.10.2020 pod licencją CCO 1.0
2. Kaggle *https://www.kaggle.com/rounakbanik/ted-talks* 2.10.2020 pod licencją CC BY-NC-SA 4.0