Raport

Projekt - Głębokie sieci neuronowe

Rozpoznawanie modeli samochodów

Aneta Bargiel 252937, Jan Zając 253997

Prowadzący:
dr hab. inż. Andrzej Rusiecki, prof. ucz.

Wydział Informatyki i Telekomunikacji
Politechniki Wrocławskiej

2 lutego 2024

1 Cel i opis projektu

Celem projektu było nauczenie sieci rozpoznawania ze zdjęć modelu samochodu. Do nauczenia zostały wybrane dwie sieci ResNet50 oraz Xception. Po nauczniu sieci zostały przetestowane i porównano ich wyniki.

1.1 Zbiór danych

W zbiorze danych znalazło się 8144 zdjęć ze zbioru Stanford Cars Dataset train, jest to pomniejszony zbiór danych, zawarta jest wnim część treningowa tylko, bez testowej z głównego zbioru. Zdjęcia samochodów są nie jednolite, auta są pod różnym kątem, zdjęcia zrobione są zarówno od tyłu samochodów jak i od przodu. Ponad to samochody są na różnych tłach, jednolitych jak na tle budynku ale także na tle parku itp.
Zbiór dzieli się na 10 klas: cabriolet, auto osobowe, pickup, suv,van, hatchback, minivan, sedan, wagon, pozostałe.
W późniejszych badaniach jako, że zbiór był mocno niezbalansowany wykorzystano bibliotekę scikit i funkcję compute_class_weight, która automatycznie przypisuje wagi klas w zbiorze treningowym na podstawie częstości ich występowania, co umożliwia modelowi lepszą adaptację do niezrównoważonych danych poprzez nadanie większej wagi mniej licznie reprezentowanym klasom. Utworzony słownik wag klas (class_weights_dict) jest używany podczas treningu, aby uwzględnić te wagi podczas obliczania funkcji straty.

1.2 Modele sieci neuronowych

Zdecydowano się na dwie sieci:

1.2.1 ResNet50

Kluczowym elementem architektury ResNet-50 jest wprowadzenie połączeń pomijanych lub połączeń skrótowych, które pozwalają sieci ominąć niektóre warstwy i zachować informacje z wcześniejszych warstw. Zdecydowano się na wykorzystanie tej sieci ze względu na jej popularność oraz jakość klasyfikacji w przykładach podobnych projektów. Zastosowano transfer learning poprzez wykorzystanie wcześniej wytrenowanych wag na zbiorze danych ImageNet.

1.2.2 Xception

Sieć Xception zbudowana jest w oparciu o głęboki stos 36 warstw splotowych z liniowymi połączeniami szczątkowymi (rysunek 5.4). Łącznie, model ma 71 warstw. Istnieją dwie ważne warstwy splotowe w tej architekturze. Pierwszą z nich jest głęboka warstwa splotowa. W niej znajduje się splot przestrzenny, niezależny w każdym kanale danych wejściowych. Następnie występuje warstwa splotowa z jądrem 1×1, która mapuje kanały wyjściowe na nową przestrzeń kanałów, za pomocą splotu głębokiego.

2 Implementacja modeli

Zbiór danych podzielono w stosunku 80:20 na foldery treningowy i testowy, a każdy z nich zawierał w zobie podfoldery o nazwie klasy, której zdjęcia w sobie zawierały.

```
import os
import shutil
from sklearn.model_selection import train_test_split
import pandas as pd
# Load the CSV file
csv_path = '/content/drive/MyDrive/stanford_cars_type.csv'
df = pd.read_csv(csv_path)
```

```
# Split data
train_data, test_data = train_test_split(df, test_size=0.2,
                                         random_state=42)

# Loop through unique car types
car_types = df['car_type'].unique()
# Create subfolders for each car type in both training
and testing folders
for car_type in car_types:
    os.makedirs(os.path.join(train_folder, car_type),
                exist_ok=True)
    os.makedirs(os.path.join(test_folder, car_type),
                exist_ok=True)

# Copy images to training folder
for i in train_data.index:
    source_path = os.path.join(
            '/content/drive/MyDrive/stanford_cars_type',
            train_data['car_type'][i],
            train_data['file_name'][i])
    destination_path = os.path.join(
            train_folder,
            train_data['car_type'][i],
            train_data['new_filename'][i])
    shutil.copy(source_path, destination_path)

# Copy images to testing folder
for i in test_data.index:
    source_path = os.path.join(
            '/content/drive/MyDrive/stanford_cars_type',
            test_data['car_type'][i],
            test_data['file_name'][i])
    destination_path = os.path.join(
            test_folder,
            test_data['car_type'][i],
            test_data['new_filename'][i])
    shutil.copy(source_path, destination_path)
```

Kod, który został użyty do balansowania zbioru danych, korzysta z funkcji compute_class_weight z modułu sklearn.utils. Proces ten ma na celu uwzględnienie niezrównoważenia liczby przykładów w poszczególnych klasach i dostosowanie wag, aby model mógł lepiej radzić sobie z tymi nieregularnościami.
Do wykorzystanego bazowego modelu ResNet50 dodano warstwę Global Average Pooling

oraz warstwę gęstą, z funkcją aktywacji softmax, z 10 neuronami, odpowiadającymi liczbie klas w zbiorze danych.

```
base_model = ResNet50(weights='imagenet', include_top=False)
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])
```

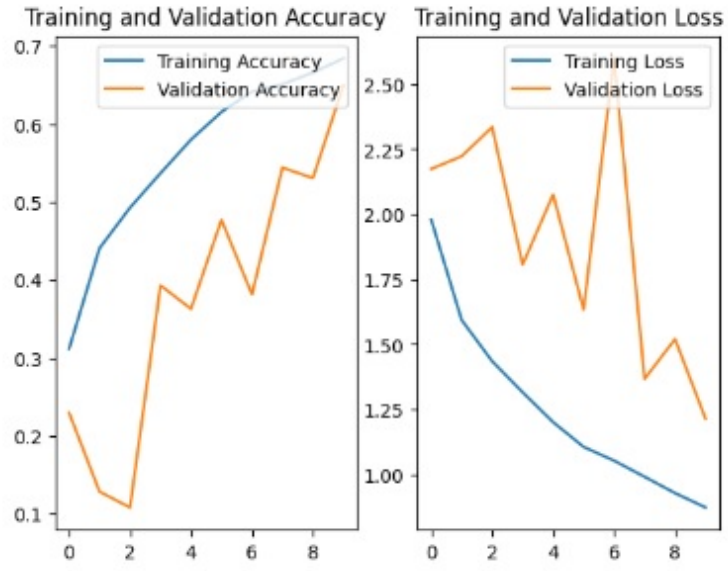
```
base_model = Xception(weights='imagenet', include_top=False)
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
            metrics=['accuracy'])
```

Po wytrenowaniu modeli, podczas weryfikacji wyników na podstawie klasyfikacji przykładowych zdjęć, dla każdego pliku wykonywane są następujące kroki preprocessingu i klasyfikacja:

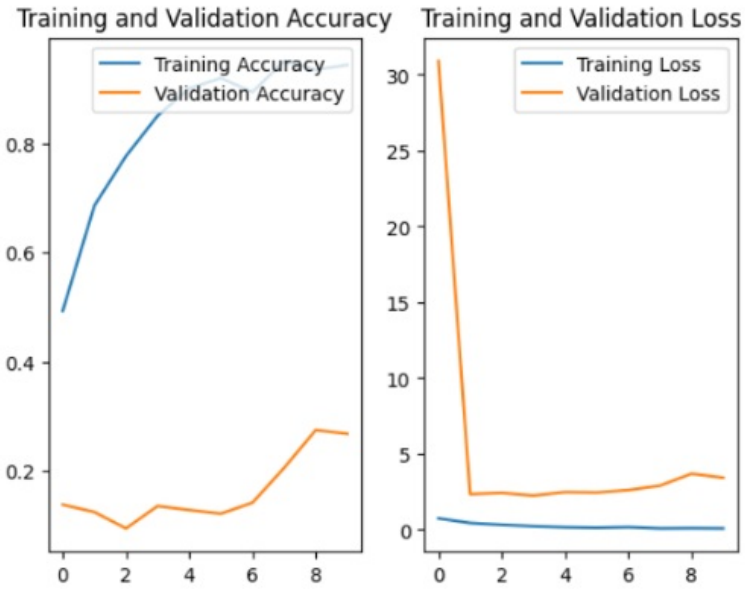
```
    image = load\_img(car, target\_size=image\_size)
image = img\_to\_array(image)
image = image.reshape((
        1,
        image.shape[0],
        image.shape[1],
        image.shape[2]))
temp = plt.imread(car)
plt.imshow(temp)
plt.show()
image = keras.applications.resnet50.preprocess\_input(image)
predictions = model.predict(image)
score = tf.nn.softmax(predictions[0])
print(
"Prediction:␣{}␣with␣a␣{:.2f}␣percent␣confidence."
.format(class\_names[np.argmax(score)], 100 * np.max(score)))
```

3 Wyniki

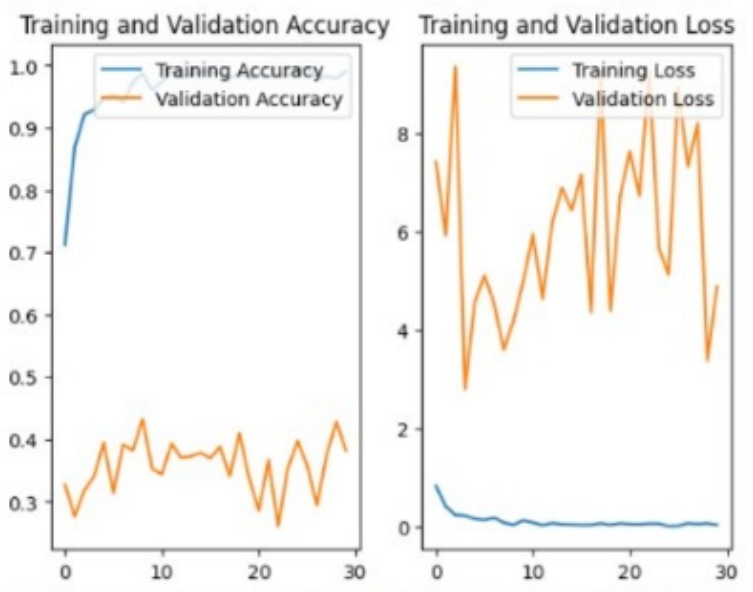
Wyniki wytrenowanego modelu ResNet50 na nie znormalizowanych danych:



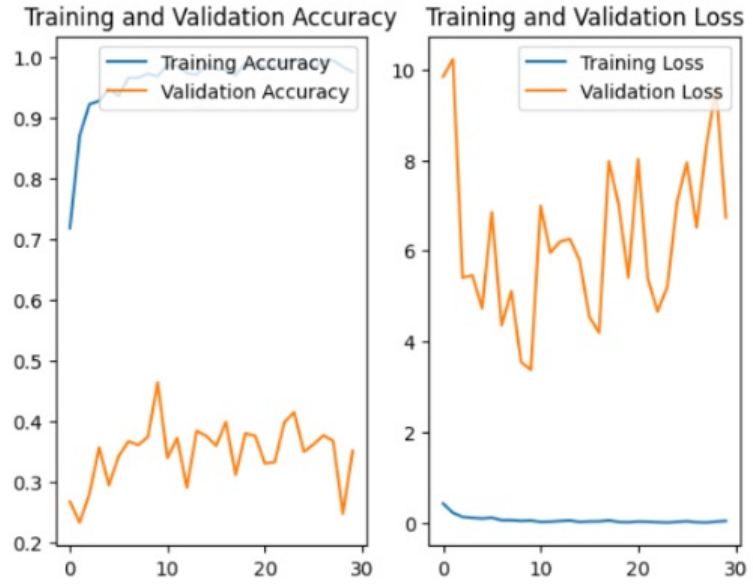
Wyniki wytrenowanego modelu ResNet50 na znormalizowanych danych:



Wyniki wytrenowanego modelu Xception na nie znormalizowanych danych:



Wyniki wytrenowanego modelu Xception na znormalizowanych danych:



Przykłady dobrze wykrytych modeli samochodów przez wytrenowaną sieć Xception, sięć trenowana przez 30 epok na znormalizowanych danych treningowych.

4 Podsumowanie i wnioski

- Na wykresach widać, że model ResNet50 uczy się lepiej na podanych danych niż model Xception,
- Głębokie sieci neuronowe uczą się stosunkowo wolno, jeśli nie wykorzystuje się GPU,
- Otrzymane wyniki nie pokrywają się z wiedzą teoretyczną, ani z wynikami podobnych klasyfikatorów o podobnej nauturze, co może świadczyć albo o złym dobraniu zbioru testowego, dla którego ekstrakcja cech okazała się niewystarczająca, albo problem jest ze wstępnym przetwarzaniem danych.
- Duża dokładność klasyfikatora podczas treningu, a niska przy validacji i testach świadczy o przeuczeniu danych.

5 Bibliografia

1. https://blog.aiensured.com/resnet50/
2. https://maelfabien.github.io/deeplearning/xception/
3. https://www.kaggle.com/datasets/mayurmahurkar/stanford-car-body-type-data/data
4. https://patricia-schutter.medium.com/car-image-recognition-with-convolutional-neural-network-applications-e791c98c9d72
5. https://bc.pollub.pl/Content/13855/diagnostyka.pdf