Splątane sieci neuronowe CNN - architektura Inception

mgr inż. Grzegorz Kossakowski

8.10.2024

1. Opis architektury

Inception [1] jest to architektura opracowana przez badaczy Google. Obecnie jest używana wersja trzecia, która wprowadza kilka dodatkowych ulepszeń. w porównaniu do wcześniejszych wersji.

2. Pobranie potrzebnych bibliotek

Kolejnym krokiem jest wczytanie wszystkich potrzebnych bibliotek, dzięki którym będzie możliwe wykorzystanie ich w procesie klasyfikacji.

[2]:
```
TF_ENABLE_ONEDNN_OPTS=0
from astropy.io import fits
from keras.callbacks import ReduceLROnPlateau
from keras.optimizers import Adam
from keras import Sequential
from tests.test_layers import Dense, Flatten
from keras.applications import InceptionV3
import pandas as pd
```

3. Pobranie danych z pliku fits

Dlatego że wcześniej podzieliliśmy dane na odpowiednie części, teraz pobieramy dwa zbiory. Pierwszy z nich to zbiór, na którym będziemy uczyć nasz model. Drugi to zbiór walidacyjny.

[3]:
```
hdu_train = fits.open('Data/train.fits')
hdu_valid = fits.open('Data/valid.fits')
x_train = hdu_train[0].data
y_train = hdu_train[1].data
x_valid = hdu_valid[0].data
y_valid = hdu_valid[1].data
```

[4]:
```
x_train.shape, x_valid.shape, type(x_train)
```

[4]:
```
((11350, 256, 256, 3), (2838, 256, 256, 3), numpy.ndarray)
```

4. Pobranie danych

W tym kroku pobieramy dane, a następnie przygotowujemy je do klasyfikacji. Modele głębokiej sieci neuronowej [4] wymaga danych z zakresu 0..1, dlatego wszystkie wartości w danych są dzielone

przez 255. Powodem takiego zachowania jest fakt, że dane obrazów są przechowywane w zakresie liczb 0..255. Dzielenie przez 255 powoduje, że dane zostaną zapisane w zakresie od 0..1, zgodnie z wymaganiami modelu.

[5]:
```
reduceLR = ReduceLROnPlateau(monitor='accuracy', factor=.001, patience=1,␣
 ↪min_delta=0.01, mode="auto")
x_train = x_train / 255.0
x_valid = x_valid / 255.0
```

5. Ustawienie sposobu nauki

Modele, które używany są już wstępnie wyuczone, dlatego chciałem sprawdzić, jak dany model będzie się zachowywał w dwóch przypadkach. Pierwszy przypadek gdy wartość fullStudy zostanie ustawiona na false wtedy model będzie wykorzystywał wcześniej nauczony model i na ostatnich warstwach będzie douczał tylko danymi astronomicznymi. Gdy ustawimy wartość na true, model od początku będzie, wykonał naukę architektury. Wcześniejsza nauka nie będzie brana pod uwagę. Pozwoli to ocenić, który sposób jest bardziej efektywny.

[6]:
```
fullStudy = False
```

6. Budowa modelu.

Model w tym przypadku to Inception v3. Po warstwach splątanych jest warstwa flatten. Zadaniem tej warstwy jest spłaszczenie obrazu z wymiarów otrzymanych po przejściu warstw splątanych do pojedynczego ciągu. Ostatnią warstwą jest gęsto połączona warstwa wyjściowa. W naszym modelu klasyfikacja odbywa się dla 10 kategorii, dlatego ta warstwa ma 10 neuronów.

[7]:
```
base_model = InceptionV3(weights='imagenet', include_top=False,␣
 ↪input_shape=(256, 256,3))
base_model.trainable = fullStudy
model_optimizer = Adam(learning_rate=0.001)

model = Sequential()
model.add(base_model)
model.add(Flatten())
model.add(Dense(10, activation="softmax"))

model.compile(optimizer=model_optimizer, loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 6, 6, 2048)        21802784

 flatten (Flatten)           (None, 73728)             0
```

```
 dense (Dense)               (None, 10)                737290

=================================================================
Total params: 22540074 (85.98 MB)
Trainable params: 737290 (2.81 MB)
Non-trainable params: 21802784 (83.17 MB)
_________________________________________________________________
```

7. Uczenie

W tym momencie model zaczyna proces uczenia. Czyli otrzymuje dwa zbiory danych i etykiet. Pierwszy z nich to dane, na podstawie których model się uczy. Drugi mniejszy zbiór jest zbiorem walidacyjnym, który pozwala na sprawdzenie postępów w nauce, na danych, których model jeszcze nie widział. Pozwala to ocenić postępy w nauce już w czasie uczenia. Kolejny zbiór danych zostanie wykorzystany na końcu celem ostatecznego sprawdzenia poprawności działania modelu.

[8]:
```
history = model.fit(x_train, y_train, epochs=10,␣
↪callbacks=[reduceLR],validation_data=(x_valid, y_valid))
```

```
Epoch 1/10
355/355 [==============================] - 232s 642ms/step - loss: 5.7742 -
accuracy: 0.4504 - val_loss: 5.3125 - val_accuracy: 0.4951 - lr: 0.0010
Epoch 2/10
355/355 [==============================] - 222s 625ms/step - loss: 3.1561 -
accuracy: 0.6152 - val_loss: 4.0943 - val_accuracy: 0.5560 - lr: 0.0010
Epoch 3/10
355/355 [==============================] - 221s 622ms/step - loss: 2.0094 -
accuracy: 0.7128 - val_loss: 5.0332 - val_accuracy: 0.5092 - lr: 0.0010
Epoch 4/10
355/355 [==============================] - 221s 622ms/step - loss: 1.6254 -
accuracy: 0.7685 - val_loss: 4.7033 - val_accuracy: 0.5835 - lr: 0.0010
Epoch 5/10
355/355 [==============================] - 221s 623ms/step - loss: 1.3923 -
accuracy: 0.8021 - val_loss: 5.0677 - val_accuracy: 0.5743 - lr: 0.0010
Epoch 6/10
355/355 [==============================] - 221s 621ms/step - loss: 0.9373 -
accuracy: 0.8515 - val_loss: 6.1575 - val_accuracy: 0.5458 - lr: 0.0010
Epoch 7/10
355/355 [==============================] - 220s 620ms/step - loss: 0.8588 -
accuracy: 0.8647 - val_loss: 7.5009 - val_accuracy: 0.5148 - lr: 0.0010
Epoch 8/10
355/355 [==============================] - 220s 620ms/step - loss: 1.2673 -
accuracy: 0.8440 - val_loss: 7.3508 - val_accuracy: 0.5599 - lr: 0.0010
Epoch 9/10
355/355 [==============================] - 220s 620ms/step - loss: 0.7327 -
accuracy: 0.9004 - val_loss: 6.2742 - val_accuracy: 0.5736 - lr: 1.0000e-06
Epoch 10/10
```

```
355/355 [==============================] - 220s 621ms/step - loss: 0.4171 -
accuracy: 0.9293 - val_loss: 5.8870 - val_accuracy: 0.5765 - lr: 1.0000e-06
```

8. Zapis architektury

[9]:
```
if fullStudy == True:
    model.save('Models/Inception_full.keras')
else:
    model.save('Models/Inception.keras')
```

9. Zapis otrzymanych danych podczas nauki

Po zakończeniu uczenia zapisujemy dane, które otrzymaliśmy podczas uczenie do pliku CSV. Pozwoli nam to później przeanalizować dane w późniejszym czasie.

[10]:
```
historyModelLearning = pd.DataFrame()
historyModelLearning['loss'] = history.history['loss']
historyModelLearning['accuracy'] = history.history['accuracy']
historyModelLearning['val_loss'] = history.history['val_loss']
historyModelLearning['val_accuracy'] = history.history['val_accuracy']
if fullStudy == True:
    historyModelLearning.to_csv('ResultLearning/Inception_full.csv', index=True)
else:
    historyModelLearning.to_csv('ResultLearning/Inception.csv', index=True)
```

Literatura

1. https://iq.opengenus.org/inception-v3-model-architecture/#google_vignette