Systemy Obliczeniowe

Laboratorium 5 – Lista A (12 grudnia 2024)

dr inż. Paweł Trajdos

1 Uwagi

1. Po zakończeniu realizacji zadań należy załadować na Eportal skrypy pythona. Format nazwy pliku `ZA.py`; (`A` to numer zadania). Źle nazwane pliki nie są oceniane.
2. W przypadku niepewności lub zauważenia jakichkolwiek błędów w instrukcji należy niezwłocznie powiadomić prowadzącego laboratorium w celu wyjaśnienia sprawy. Reklamacje po zakończeniu zajęć nie będą uwzględniane.

2 Zadania

Zadanie 5.0(Pkt. 6.0):
Zadanie dotyczyć będzie danych sygnałowych oraz sieci konwolucyjnych. Badania będą przeprowadzane z wykrozystaniem zbioru fashion mnist, który można pobrać w następujący sposób:

```
import ssl
ssl._create_default_https_context = ssl._create_unverified_context # na problemy z ssl
from tensorflow.keras.datasets.fashion_mnist import load_data
(X_train, y_train), (X_test, y_test) = load_data()
```

Następnie przygotujemy dane do eksperymentu poprzez przekodowanie wektorów klas do postaci kategorycznej oraz znormalizowanie danych:

```
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], X_train.shape[2], 1))
X_test = X_test.reshape((X_test.shape[0], X_train.shape[1], X_train.shape[2], 1))

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

Przyszedł czas na zdefiniowanie struktury sieci, powinna ona zawierać dowolnie wybraną przez Państwa kombinację warstw konwolucji oraz poolingu, prosty przykład poniżej:

```
model = Sequential([
  layers.Conv2D(16, 3, padding='same', activation='relu',
                input_shape=(img_height, img_width, 1)),
  layers.MaxPooling2D(2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes, activation="softmax")
])
```

Kompilacji dokonujemy z wykorzystaniem optymalizatora **adam** oraz funkcji straty **categorical crossentropy** pod względem metryki dokładności (accuracy):

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Dzięki temu możemy wyświetlić strukturę modelu za pomocą metody **summary()**.

Teraz sprawdźmy, jak zdefiniowana przez nas sieć radzi sobie z klasyfikacją obiektów. W tym celu należy wywołać metodę **fit()**, która zwraca nam całą historię treningu modelu. Poniżej przykładowo tylko trzy epoki dla skrócenia czasu uczenia:

```
history = model.fit(X_train, y_train, epochs=3,
                    validation_data=(X_test, y_test), batch_size=32)
```

Wiedząc, że **history** jest obiektem zawierającym informacje o dokładności na zbiorze treningowym (accuracy) oraz na zbiorze walidacyjnym (vam accuracy w notacji wężowej) proszę o wyrysowanie krzywych uczenia sieci. Dodatkowo proszę o podanie wartości dokładności klasyfikacji sieci na zbiorze testowym (metoda **evaluate()**).

Pamiętaj, że częścią zadania jest zaimportowanie odpowiednich modułów.

Zadanie 5.1(Pkt. 6.0):

Wykorzystaj wytrenowany model jako ekstraktor cech dla klasyfikatora **Random Forest** (scikit-learn). W tym celu należy m.in pozbyć się warstw typu **Dense**.

```
    extractor = Sequential( model.layers[:-2])#Nowy model bez warstw Dense

    X_train_extracted = extractor.predict(X_train)#Ekstrakcja cech za pomoca nowego modelu


```

Wytrenuj klasyfikator **Random Forest** na atrybutach uzyskanych z ekstraktora. Pamiętaj, że RandomForest przyjmuje etykiety w innym formacie niż keras. Dokonaj przekodowania etykiet wykorzystując np. **numpy.argmax**.

Na zbiorze testowym porównaj jakość klasyfikacji oryginalnego modelu z modelem Random Forest wytrenowanym na wyekstrachowanych atrybutach.