Klasyfikacja z wykorzystaniem sieci neuronowych

W ramach tego ćwiczenia Państwa zadaniem będzie przeprowadzenie eksperymentu mającego na celu wytrenowanie modelu do klasyfikacji odręcznie pisanych cyfr z bazy MNIST. Jako klasyfikator należy zastosować każdy z trzech typów: pojedynczy neuron, sieć neuronowa MLP oraz sieć konwolucyjna.

Na początku proszę zaimportować odpowiednie pakiety

```
import numpy as np
from keras.datasets import mnist
import keras
from keras import layers
from sklearn.metrics import classification_report, confusion_matrix
```

Wczytanie i konwersja danych

Baza danych jest zawarta w bibliotece tensorflow, która zostanie też użyta do utworzenia i wytrenowania każdego z modeli.
Aby wczytać dane należy wywołać metodę:

```
training_set, test_set = mnist.load_data()
```

Dla wygody, warto podzielić zbiór danych na przykłady oraz etykiety:

```
X_train, y_train = training_set
X_test, y_test = test_set
```

Po wypisaniu wymiaru wczytanych danych, możemy się spodziewać wartości:

```
print(X_train.shape)
(60000, 28, 28)
```

Co oznacza, że tensor ma 3 wymiary i każdy z nich jest reprezentowany przez:
- ilość przykładów,
- wysokość obrazu,
- szerokość obrazu.

Tensorflow przyjmuje dane w postaci obrazów z jeszcze jednym dodatkowym wymiarem na końcu - liczbą kanałów (w przypadku obrazów w skali szarości jest to 1 kanał, dla obrazów RGB - 3 kanału). Należy dodać wymiar kanału do zbioru uczącego i testowego:

```
X_train = np.expand_dims(X_train, -1)
X_test = np.expand_dims(X_test, -1)
```

Ponowne sprawdzenie wymiary danych powinno przynieść spodziewaną zmianę.

```
print(X_train.shape)
(60000, 28, 28, 1)
```

W ostatnim kroku, wartości pikseli warto przeskalować do zakresu 0.0-1.0. Obraz jest w skali szarości, gdzie wartość pikseli jest reprezentowana na ośmiu bitach, dlatego też wystarczy podzielić każdy z pikseli przez maksymalną możliwą wartość, tj 255:

```
X_train = X_train.astype(np.float) / 255.0
X_test = X_test.astype(np.float) / 255.0
```

Przygotowanie eksperymentu

Na początku proszę zdefiniować kilka wartości definiujących warunki eksperymentu:

```
num_classes = 10
input_shape = (28, 28, 1)
batch_size = 128
epochs = 30
```

gdzie każda z nich oznacza:
num_classes - ilość klas do klasyfikacji (w tym przypadku 10 gdyż klasyfikujemy cyfry 0-9),
input_shape - rozmiar wejścia na sieć (obraz w skali szarości z jednym kanałem),
batch_size - rozmiar paczki będącej jednorazowym wejściem na sieć (z reguły sieci nie trenuje się na wszystkich danych na raz, m. in. z powodu ograniczeń pamięci dla obszernych danych),
epochs - ilość epok uczenia. Jako jedną epokę interpretujemy “pokazanie” wszystkich przykładów modelowi.

Następnie należy przygotować model:

```
model = keras.Sequential(
        [
            keras.Input(shape=input_shape),
            keras.layers.Flatten(),
            layers.Dense(num_classes, activation="softmax")
        ]
    )
```

Wyświetlić podsumowanie:

```
model.summary()
```

I go skompilować:

```
model.compile(loss="sparse_categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])
```

W przypadku tworzenia modelu, korzystamy tu z interfejsu biblioteki Keras, która stanowi uproszczenie interfejsu biblioteki tensorflow. Powyższy kod utworzy model sekwencyjny (bez rozgałęzień) składający się z warstwy wejściowej, warstwy rozplatającej obraz z macierzy HxWxD na wektor jednowymiarowy o długości H*W*D, oraz warstwy gęstej reprezentowanej przez jeden neuron dla każdej z klas. Warstwa gęsta kończy się funkcją aktywacji softmax często stosowanej do klasyfikacji wieloklasowej. Tak przygotowaną sieć można interpretować jako regresję logistyczną dla 10 klas - jest to klasyfikator liniowy. W przypadku kompilacji modelu, należy podać funkcję straty - w tym przypadku entropię krzyżową, typ f. optymalizującej, oraz dodatkowe metryki które chcemy śledzić podczas uczenia. Categorical w przypadku entropii krzyżowej oznacza klasyfikację wieloklasową (w przypadku klasyfikacji binarnej byłoby binary), natomiast sparse - że nie podajemy na wyjściu sieci etykiet kodowanych one-hot, tylko liczbą naturalną od 0 do 9.

Uczenie modelu

Tak przygotowany model można poddać uczeniu:

```
model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs,
validation_split=0.2)
```

validation_split oznacza jaka część zbioru jest przeznaczona na walidację.

W konsoli pojawią się logi zawierające informacje o postępie uczenia, przykładowo:

```
Epoch 1/30
375/375 [==============================] - 5s 14ms/step - loss: 0.3938 -
accuracy: 0.8814 - val_loss: 0.1062 - val_accuracy: 0.9703
Epoch 2/30
375/375 [==============================] - 5s 15ms/step - loss: 0.1185 -
accuracy: 0.9632 - val_loss: 0.0740 - val_accuracy: 0.9788
Epoch 3/30
375/375 [==============================] - 5s 15ms/step - loss: 0.0898 -
accuracy: 0.9727 - val_loss: 0.0585 - val_accuracy: 0.9839
```

Spodziewamy się spadku wartości funkcji kosztu (loss) oraz wzrostu metryki accuracy z epoki na epokę na zbiorze uczącym i walidacyjnym. Ilość epok staramy się tak dobrać aby otrzymać jak najniższe wartości kosztu (loss oraz val_loss) na obu zbiorach.

Ocena modelu

Do oceny modelu klasyfikacji głównie stosuje się metryki czułości (sensitivity, recall) oraz pozytywnego przewidywania (positive predictivity, precision) oraz ich średnią harmoniczną (F1-score). Aby policzyć metryki można użyć wbudowanych w bibliotekę sciki t-learn funkcji:

```
from sklearn.metrics import classification_report, confusion_matrix
```

Jednak przed ich użyciem należy wygenerować odpowiedzi modelu na zbiorze testowym:

```
y_probab = model.predict(X_test)
```

```
y_pred = np.argmax(y_probab, axis=1)
```

```
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

Przykładowe wyniki dla powyższego modelu mogą wyglądać następująco:

```
precision    recall  f1-score   support

           0       0.96      0.98      0.97       980
           1       0.97      0.98      0.98      1135
           2       0.94      0.90      0.92      1032
           3       0.90      0.92      0.91      1010
           4       0.94      0.93      0.94       982
           5       0.91      0.87      0.89       892
           6       0.94      0.95      0.95       958
           7       0.93      0.93      0.93      1028
           8       0.87      0.91      0.89       974
           9       0.92      0.91      0.92      1009

    accuracy                           0.93     10000
   macro avg       0.93      0.93      0.93     10000
weighted avg       0.93      0.93      0.93     10000
```

```
[[ 959    0    1    2    0    6    9    2    1    0]
 [   0 1115    3    3    0    1    3    2    8    0]
 [   3   10  926   15    8    4   13    9   41    3]
 [   3    0   18  926    0   23    2   11   22    5]
 [   1    2    5    2  917    0   10    4   10   31]
 [   8    2    1   34    5  779   14    8   35    6]
 [  11    3    8    1    7   14  911    1    2    0]
 [   1    7   21    9    7    1    0  951    2   29]
 [   7    6    6   23    7   21    7    9  883    5]
 [  11    7    1   13   23    5    0   23    6  920]]
```

Uczenie modeli nieliniowych

W ramach ćwiczenia, należy powtórzyć eksperyment dla modelu MLP oraz sieci konwolucyjnej. Sieć MLP powinna składać się z dwóch ukrytych warstw Dense, natomiast sieć konwolucyjna naprzemiennie z warstw konwolucyjnych oraz maxpooling. Przykładowa realizacja powyższych sieci mogłaby wyglądać następująco:

```
model = keras.Sequential(
        [
            keras.Input(shape=input_shape),
            keras.layers.Flatten(),
            layers.Dense(64, activation="tanh"),
            layers.Dense(128, activation="tanh"),
```

```
            layers.Dense(num_classes, activation="softmax")
        ]
    )
```

```
model = keras.Sequential(
        [
            keras.Input(shape=input_shape),
            layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.Flatten(),
            layers.Dropout(0.5),
            layers.Dense(num_classes, activation="softmax"),
        ]
    )
```