SPRAWOZDANIE

Zajęcia: Nauka o danych II

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 6 Data 11.05.2025 Temat: Projektowanie zaawansowanych architektur sieci neuronowych w TensorFlow lub PyTorch Wariant 6	Imię Nazwisko Hubert Mentel Informatyka II stopień, niestacjonarne, 2 semestr, gr.1a

1. Zadanie:

Cel:
Celem ćwiczenia jest zapoznanie się z projektowaniem i implementacją zaawansowanych architektur sieci neuronowych w środowisku TensorFlow lub PyTorch, z uwzględnieniem nowoczesnych rozwiązań oraz ich praktycznych zastosowań.

Treść:
Zadanie 6 (Autoencoder): Wykorzystaj autoencoder do wykrywania anomalii w zbiorze MNIST — MNIST.
Dane: Trenuj na klasach 0–4. Ewaluuj na klasach 0–9. Użyj progu błędu rekonstrukcji do detekcji.

Pliki dostępne są pod linkiem:
https://github.com/HubiPX/NOD/tree/master/NOD2/Zadanie%206

2. Opis programu opracowanego (kody źródłowe, zrzuty ekranu)

[2]:

```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt

# ====== Załaduj dane MNIST ======
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train = x_train.astype("float32") / 255.
x_test = x_test.astype("float32") / 255.
x_train = x_train.reshape(-1, 28 * 28)
x_test = x_test.reshape(-1, 28 * 28)

# ====== Filtruj tylko klasy 0–4 do treningu ======
train_mask = y_train <= 4
x_train_filtered = x_train[train_mask]

# ====== Budowa autoenkodera ======
input_dim = 784
encoding_dim = 64

input_img = tf.keras.Input(shape=(input_dim,))
encoded = layers.Dense(128, activation='relu')(input_img)
encoded = layers.Dense(encoding_dim, activation='relu')(encoded)

decoded = layers.Dense(128, activation='relu')(encoded)
decoded = layers.Dense(input_dim, activation='sigmoid')(decoded)

autoencoder = models.Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='mse')
autoencoder.summary()

# ====== Trening autoenkodera ======
autoencoder.fit(x_train_filtered, x_train_filtered,
                epochs=20,
                batch_size=256,
                shuffle=True,
                validation_split=0.2)

# ====== Ewaluacja na pełnym zbiorze testowym (klasy 0–9) ======
reconstructions = autoencoder.predict(x_test)
reconstruction_errors = np.mean(np.square(x_test - reconstructions), axis=1)

# ====== Ustal próg detekcji na podstawie klas 0–4 ======
test_mask_0_4 = y_test <= 4
threshold = np.mean(reconstruction_errors[test_mask_0_4]) + 2 * np.std(reconstruction_errors[test_mask_0_4])

# ====== Wykrywanie anomalii ======
y_pred_anomaly = reconstruction_errors > threshold
y_true_anomaly = y_test > 4  # klasy 5–9 to anomalie

# ====== Ocena skuteczności ======
from sklearn.metrics import classification_report, confusion_matrix

print("Raport klasyfikacji (anomalia = 1):")
print(classification_report(y_true_anomaly, y_pred_anomaly))
print("Macierz pomyłek:")
print(confusion_matrix(y_true_anomaly, y_pred_anomaly))

# ====== Przykładowe błędy rekonstrukcji ======
plt.hist(reconstruction_errors[test_mask_0_4], bins=50, alpha=0.6, label='Klasy 0–4')
plt.hist(reconstruction_errors[~test_mask_0_4], bins=50, alpha=0.6, label='Klasy 5–9 (anomalie)')
plt.axvline(threshold, color='red', linestyle='--', label='Próg detekcji')
plt.xlabel("Błąd rekonstrukcji")
plt.ylabel("Liczba próbek")
plt.legend()
plt.title("Histogram błędu rekonstrukcji")
plt.show()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 784)	0
dense (Dense)	(None, 128)	100,480
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 128)	8,320
dense_3 (Dense)	(None, 784)	101,136

Total params: 218,192 (852.31 KB)

Trainable params: 218,192 (852.31 KB)

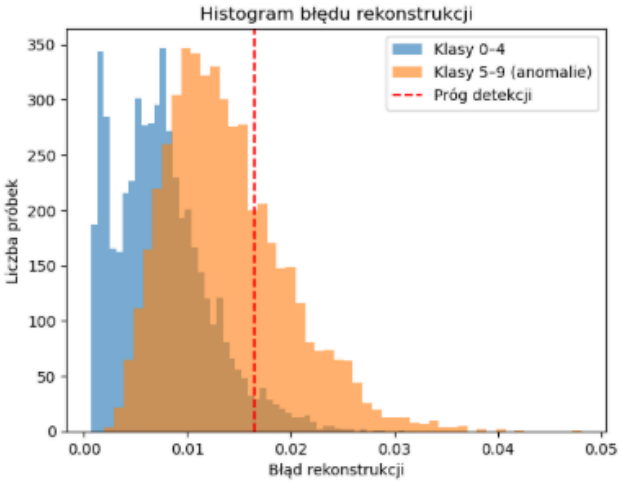
Non-trainable params: 0 (0.00 B)

```
Epoch 1/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 0.1210 - val_loss: 0.0446
Epoch 2/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0407 - val_loss: 0.0307
Epoch 3/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0288 - val_loss: 0.0237
Epoch 4/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0226 - val_loss: 0.0190
Epoch 5/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0182 - val_loss: 0.0160
Epoch 6/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0154 - val_loss: 0.0145
Epoch 7/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0139 - val_loss: 0.0131
Epoch 8/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0129 - val_loss: 0.0123
Epoch 9/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0118 - val_loss: 0.0115
Epoch 10/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0112 - val_loss: 0.0109
Epoch 11/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0106 - val_loss: 0.0103
Epoch 12/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0100 - val_loss: 0.0099
Epoch 13/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0096 - val_loss: 0.0095
Epoch 14/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0092 - val_loss: 0.0092
Epoch 15/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0089 - val_loss: 0.0089
Epoch 16/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0086 - val_loss: 0.0085
Epoch 17/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0083 - val_loss: 0.0086
Epoch 18/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0081 - val_loss: 0.0083
Epoch 19/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0079 - val_loss: 0.0079
Epoch 20/20
96/96 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0077 - val_loss: 0.0078
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Raport klasyfikacji (anomalia = 1):
              precision    recall  f1-score   support

       False       0.58      0.96      0.72      5139
        True       0.86      0.28      0.42      4861

    accuracy                           0.63     10000
   macro avg       0.72      0.62      0.57     10000
weighted avg       0.72      0.63      0.58     10000

Macierz pomyłek:
[[4913  226]
 [3503 1358]]
```



3. Wnioski

Model autoenkodera został skutecznie wytrenowany na klasach 0–4 zbioru MNIST w celu wykrywania anomalii, definiowanych jako obrazy cyfr z klas 5–9. Proces uczenia przebiegał stabilnie, o czym świadczy systematyczny spadek błędu rekonstrukcji — z początkowej wartości 0.1210 do końcowej 0.0077 przy bardzo niskim błędzie walidacyjnym (0.0078). Model wykazał się wysoką precyzją detekcji anomalii (0.86), co oznacza, że w większości przypadków poprawnie klasyfikował dane odstające. Jednocześnie uzyskano wysoką czułość (recall) dla danych normalnych (0.96), ale niższą dla anomalii (0.28), co sugeruje, że model jest bardziej zachowawczy i część anomalii może pomijać. Całkowita dokładność klasyfikacji wyniosła 63%, a dokładność dla klas normalnych była zdecydowanie wyższa niż dla anomalii. Histogram błędu rekonstrukcji pokazuje wyraźne rozróżnienie rozkładów między klasami 0–4 a 5–9 oraz odpowiednio dobrany próg detekcji, który pozwala oddzielić większość typowych próbek od odstających. Macierz pomyłek wskazuje, że głównym problemem pozostaje liczba anomalii zaklasyfikowanych jako dane normalne, co można poprawić, np. przez modyfikację architektury lub dynamiczne dopasowanie progu. Mimo tej niedoskonałości model dobrze spełnia swoją rolę jako narzędzie do wykrywania nietypowych danych obrazowych i może stanowić podstawę do bardziej zaawansowanych systemów detekcji anomalii.