Wstęp do Sztucznej Inteligencji - rok akademicki 2022/2023

Przed rozpoczęciem pracy z notatnikiem zmień jego nazwę zgodnie z wzorem: `NrAlbumu_Nazwisko_Imie_PoprzedniaNazwa`.

Przed wysłaniem notatnika upewnij się, że rozwiązałeś wszystkie zadania/ćwiczenia.

Temat: Sztuczne Sieci Neuronowe - Lab 3 - Zadania (obowiązkowe)

Biblioteka Keras. Aspekty uczenia sieci neuronowych.

Sieci neuronowe w języku Python

Obecnie za sprawą rozwoju i popularności tzw. głębokich sieci neuronowych (Deep Neural Network) dostępnych jest bardzo dużo bibliotek/frameworków do budowy i uczenia sieci neuronowych (TensorFlow, Theano, Spark MLlib, MXNet, Microsoft Cognitive Toolkit, Caffe itp.). Z wielu z nich można korzystać w prosty sposób przy wykorzystaniu języka Python.

Biblioteka Keras (na TensorFlow)

Biblioteka Keras jest wysokopoziomową nakładką na biblioteki takie jak TensorFlow, CNTK (Microsoft Cognitive Toolkit) lub Theano napisaną w języku Python. Domyślnie wykorzystywanym backendem jest TensorFlow i z takiego będziemy korzystać. Biblioteka ta pozwala na:

- Łatwe i szybkie prototypowanie modeli (pełna modularność).
- Wspiera zarówno "klasyczne" sieci neuronowe jak i konwolucyjne czy rekurencyjne.
- Umożliwia uczenie przy wykorzystaniu CPU oraz GPU.

Keras: https://keras.io/

TensorFlow: https://www.tensorflow.org/

Szybkie wprowadzenie na przykładzie sieci dla problemu XOR

Dane:

```
import numpy as np
data_x = np.array([[-1,-1],[-1,1],[1,-1],[1,1]])  # backpropagation
nie lubi zer, bez biasu
data_y = np.array([0,1,1,0])
```

Import biblioteki Tensorflow i Keras

Biblioteke Kreas można zaimportować bezpośrednio `import keras` Jednak obecnie bublioteka Keras jest również dostępna jako podmoduł biblioteki Tensorflow.

```
import tensorflow as tf
print('Tensorflow version:', tf.__version__)
print('Keras z tensorflow version:', tf.keras.__version__)

Tensorflow version: 2.16.1
Keras z tensorflow version: 3.3.3
```

Przygotowanie architektury sieci

Stworzenie sieci MLP o dwóch neuronach ukrytych i jednym wyjściowym:

Tworzenie modelu odbywa się na zasadzie budowania modelu z klocków (warstw). Najpierw tworzymy tensor będący warstwą wejściową `Input` a następnie dodajemy do niego kojejne warstwy np. `Dense` (warstwa neuronów typu każdy z każdym). Następnie dysponując tensorem wejściowym i wyjściowym określamy Model. Model można tworzyć też z wykorzystaniem klasy `Sequential`.

```
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense

x = Input(shape=(2,))  #należy ustawić kształt tensora wejściowego
h = Dense(2, use_bias=True, activation='tanh',
kernel_initializer='random_uniform',
bias_initializer='random_uniform')(x)   #parametry patrz dokumentacja
y = Dense(1, use_bias=True, activation='sigmoid',
kernel_initializer='random_uniform',
bias_initializer='random_uniform')(h)  # parametry patrz dokumentacja
# alternatywnie activation można ustawić na None i dodać funkcje
aktywacj jako osobną warstwę
mlp = Model(inputs=x, outputs=y)

mlp.summary()

Model: "functional_1"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ input_layer (InputLayer)        │ (None, 2)              │
0 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense (Dense)                   │ (None, 2)              │
6 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense_1 (Dense)                 │ (None, 1)              │
3 │
└─────────────────────────────────┴────────────────────────┴──────────
─────┘
```

```
 Total params: 9 (36.00 B)

 Trainable params: 9 (36.00 B)

 Non-trainable params: 0 (0.00 B)
```

```
m2 = Sequential()
m2.add(Dense(2, use_bias=True, activation='tanh',
kernel_initializer='random_uniform',
bias_initializer='random_uniform', input_shape=(2,)))
m2.add(Dense(1, use_bias=True, activation='sigmoid',
kernel_initializer='random_uniform',
bias_initializer='random_uniform'))
```

```
c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass
an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
m2.summary()
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
```

```
│ dense_2 (Dense)                  │ (None, 2)                │
6 │
├──────────────────────────────────┼─────────────────────────┼───────
─────┤
│ dense_3 (Dense)                  │ (None, 1)                │
3 │
└──────────────────────────────────┴─────────────────────────┴───────
─────┘

 Total params: 9 (36.00 B)

 Trainable params: 9 (36.00 B)

 Non-trainable params: 0 (0.00 B)
```

Kompilacja modelu

Po stworzeniu modelu należy go skompilować, podczas kompilacji podajemy m.in. rodzaj funkcji używanej do liczenia błędu (`loss`) oraz algorytm wykorzystywany do uczenia (`optimizer`).

```
rms = tf.keras.optimizers.RMSprop(learning_rate=0.01)  #lr = learning
rate; parametry patrz dokumentacja
mlp.compile(loss='mse', optimizer=rms)  #mse = mean squared error

m2.compile(loss='mse', optimizer=rms)
```

Uczenie

Po kompilacji możemy przystąpić do uczenia za pomocą metody `fit`.

```
print('rozpoczecie uczenia')
#ustaw verbose=0 aby wyłączyć szczegóły
hist = mlp.fit(data_x, data_y, epochs=300, verbose=1, batch_size=4)  #
parametry patrz dokumentacja
print('koniec uczenia')
# ponowne wykonanie powoduje douczanie a nie uczenie od nowa

rozpoczecie uczenia
Epoch 1/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 538ms/step - loss: 0.2500
Epoch 2/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 3/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 4/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 5/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 6/300
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 7/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 8/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 9/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 10/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 11/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 12/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 13/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 14/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 15/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 16/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 17/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 18/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 19/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 20/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 21/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 22/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 23/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 24/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 25/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 26/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 27/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - loss: 0.2500
Epoch 28/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - loss: 0.2500
Epoch 29/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 30/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - loss: 0.2500
```

```
Epoch 31/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 32/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 33/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - loss: 0.2500
Epoch 34/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 35/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 36/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - loss: 0.2500
Epoch 37/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 38/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - loss: 0.2500
Epoch 39/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 40/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 41/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 42/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 43/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step - loss: 0.2500
Epoch 44/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 45/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 46/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - loss: 0.2500
Epoch 47/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 48/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 49/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 50/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 51/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 52/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 53/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 54/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 55/300
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 56/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 57/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 58/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 59/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 60/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 61/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 62/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 63/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 64/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 65/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - loss: 0.2500
Epoch 66/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 67/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 68/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 69/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 70/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - loss: 0.2500
Epoch 71/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 0.2500
Epoch 72/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 73/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.2500
Epoch 74/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 75/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 76/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 77/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 78/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 79/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
```

```
Epoch 80/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 81/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 82/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 83/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 84/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 85/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 86/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 87/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 88/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 89/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 90/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 91/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 92/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 93/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 94/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 95/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 96/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 97/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 98/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 99/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 100/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 101/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 102/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 103/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 104/300
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 105/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 106/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 107/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - loss: 0.2500
Epoch 108/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 0.2500
Epoch 109/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 110/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 111/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 112/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 113/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 114/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 115/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 116/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 117/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 118/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 119/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 120/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 121/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 122/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 123/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 124/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 125/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 126/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - loss: 0.2500
Epoch 127/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 128/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
```

```
Epoch 129/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 130/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 131/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 132/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 133/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 134/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.2500
Epoch 135/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 136/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 137/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 138/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 139/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 140/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 141/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step - loss: 0.2500
Epoch 142/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - loss: 0.2500
Epoch 143/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 144/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 145/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 146/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - loss: 0.2500
Epoch 147/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 148/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 149/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 150/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 151/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 0.2500
Epoch 152/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 153/300
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 154/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 155/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 156/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 157/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 158/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 159/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 160/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 161/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 162/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 163/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 164/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 165/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 166/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 167/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 168/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 169/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 87ms/step - loss: 0.2500
Epoch 170/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 171/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.2500
Epoch 172/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 173/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 174/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 175/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 176/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 177/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
```

```
Epoch 178/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 179/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 180/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 181/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - loss: 0.2500
Epoch 182/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - loss: 0.2500
Epoch 183/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 184/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 0.2500
Epoch 185/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.2500
Epoch 186/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 187/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 188/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 189/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 190/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 191/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 192/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 193/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 194/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 195/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - loss: 0.2500
Epoch 196/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 197/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 198/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 199/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 200/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 201/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 202/300
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 203/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 204/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - loss: 0.2500
Epoch 205/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 206/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 207/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step - loss: 0.2500
Epoch 208/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 209/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 210/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 211/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 212/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 213/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - loss: 0.2500
Epoch 214/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 215/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 216/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 217/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - loss: 0.2500
Epoch 218/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 219/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.2500
Epoch 220/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 221/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 222/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 223/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 224/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 225/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 226/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
```

```
Epoch 227/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 228/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.2500
Epoch 229/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 230/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 231/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 232/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 233/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 234/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 235/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 236/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 237/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - loss: 0.2500
Epoch 238/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 0.2500
Epoch 239/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 240/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step - loss: 0.2500
Epoch 241/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - loss: 0.2500
Epoch 242/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 243/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.2500
Epoch 244/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 245/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 246/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 247/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.2500
Epoch 248/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 249/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 250/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 251/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 0.2500
```

```
Epoch 252/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 253/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 254/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 255/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 256/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 257/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 258/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 259/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 260/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 261/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 262/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 263/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 264/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.2500
Epoch 265/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 266/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 267/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 268/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 269/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 270/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 271/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - loss: 0.2500
Epoch 272/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 273/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 274/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 275/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 276/300
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.2500
Epoch 277/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 278/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 279/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - loss: 0.2500
Epoch 280/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 0.2500
Epoch 281/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 282/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - loss: 0.2500
Epoch 283/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 284/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 0.2500
Epoch 285/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.2500
Epoch 286/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - loss: 0.2500
Epoch 287/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - loss: 0.2500
Epoch 288/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - loss: 0.2500
Epoch 289/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - loss: 0.2500
Epoch 290/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 291/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.2500
Epoch 292/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.2500
Epoch 293/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.2500
Epoch 294/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.2500
Epoch 295/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 0.2500
Epoch 296/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 297/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 101ms/step - loss: 0.2500
Epoch 298/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.2500
Epoch 299/300
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.2500
Epoch 300/300
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - loss: 0.2500
koniec uczenia
```

Sprawdzenie czego model się nauczył:

```
pred = mlp.predict(data_x)
print(pred)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
[[0.49891087]
 [0.4989258 ]
 [0.4989175 ]
 [0.49893245]]
```

Zagadnienie niedouczenia lub przeuczenia sieci

Przykładowy problem aproksymacji funkcji.

Zwróć uwagę

- W rzeczywistości dane często pochodzą z pomiarów, które obarczone są niepewnością. W poniższym przykładzie modelujemy to poprzez dodanie losowego błędu do wartości funkcji sinus.

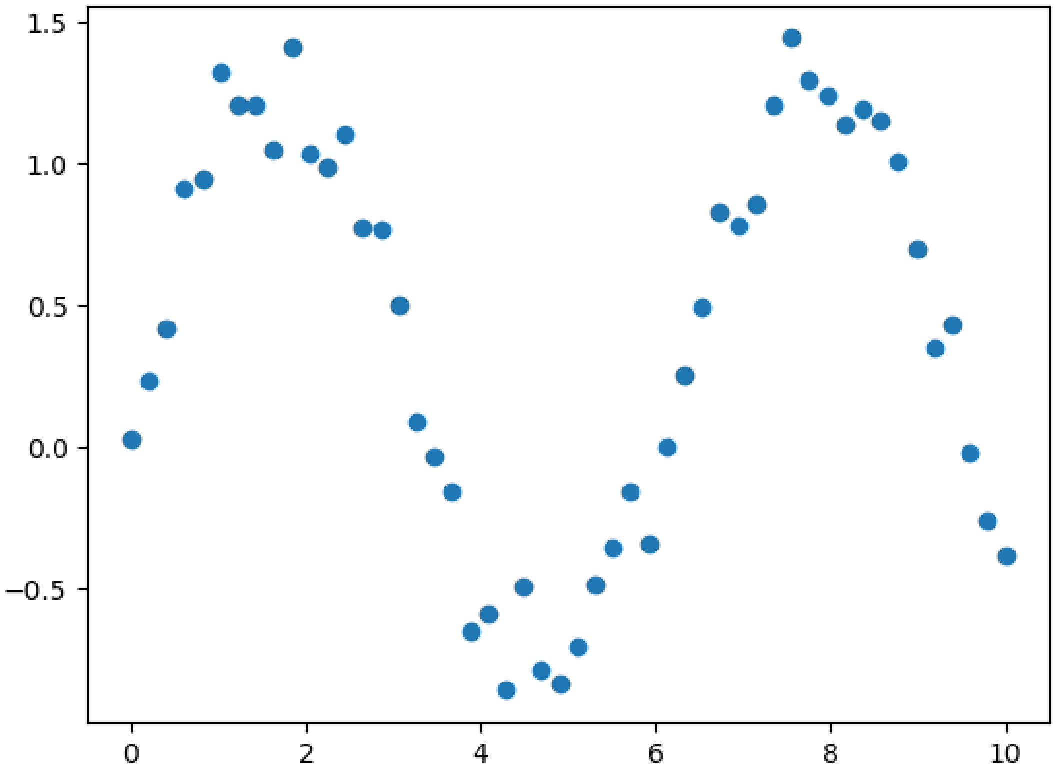
- Funkcja sinus pełni tu rolę rzeczywistego modelu, którego w praktycznych problemach tak naprawdę nie znamy. Próbujemy go odkryć/aproksymować na podstawie dostępnych nam danych.

- Celem nauki jest osiągnięcie dobrej generalizacji. Tutaj oznacza to, że sieć, na podstawie dostępnych (zaszumionych) przykładów, powinna nauczyć się prawidłowego przebiegu funkcji sinus.

- Jeśli będziemy uczyć sieć zbyt długo, może pojawić się niekorzystny efekt zwany przeuczeniem. Ma to miejsce gdy sieć po odkryciu głównych zależności/ogólnego przebiegu funkcji, zaczyna dostosowywać się do szumu istniejącego w danych. Można temu przeciwdziałać poprzez odpowiednio wczesne zatrzymanie procesu uczenia.

Dane

```
import matplotlib.pyplot as plt
data_x = np.linspace(0, 10, 50)
data_y = np.sin(data_x) + np.random.random(data_x.shape[0])*0.5
fig = plt.figure()
plt.plot(data_x, data_y, 'o')
plt.show()
```

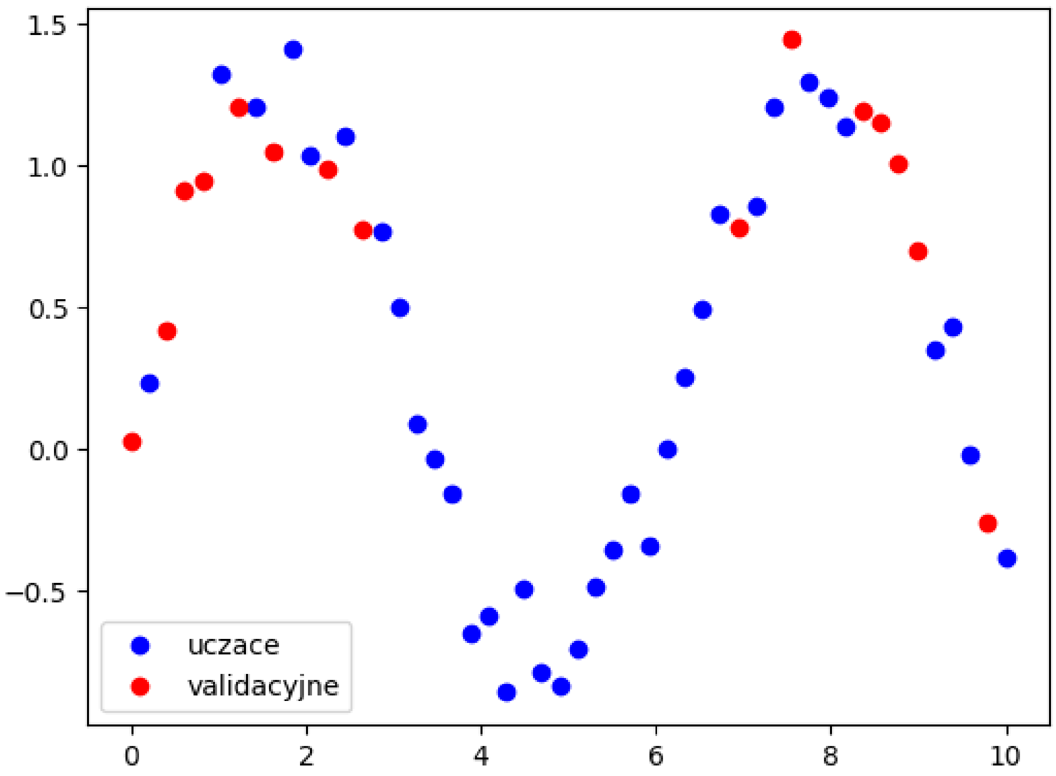


Podział na dane uczące i walidacyjne

Dane walidacyjne służą do monitorowania procesu uczenia, sprawdzania jak sieć radzi sobie z danymi, które nie są wykorzystywane do modyfikacji wag.

Jeśli błąd na danych uczących maleje, a na danych walidacyjnych już nie (lub wręcz rośnie), jest to potencjalny sygnał, że sieć jest przeczuczona.

```
temp = np.arange(50)
np.random.shuffle(temp)
val_x = data_x[temp[35:]]
val_y = data_y[temp[35:]]
data_x = data_x[temp[:35]]
data_y = data_y[temp[:35]]

fig = plt.figure()
plt.plot(data_x, data_y, 'bo', label='uczace')
plt.plot(val_x, val_y, 'ro', label='walidacyjne')
plt.legend()
plt.show()
```



Model sieci MLP

Mała sieć, 35 neuronów w warstwie ukrytej.

```
x = Input(shape=(1,))
h = Dense(35, input_dim=1, use_bias=True, activation='tanh',
kernel_initializer='random_uniform',
bias_initializer='random_uniform')(x)
y = Dense(1, use_bias=True, activation='linear',
kernel_initializer='random_uniform',
bias_initializer='random_uniform')(h)
model1 = Model(inputs=x, outputs=y)
rms = tf.keras.optimizers.RMSprop(learning_rate=0.001)
model1.compile(loss='mse', optimizer=rms)
model1.summary()
```

```
Model: "functional_5"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━
━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━
━━━━━┩
│ input_layer_2 (InputLayer)      │ (None, 1)              │
0 │
```

```
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense_4 (Dense)                 │ (None, 35)             │
70 │
├─────────────────────────────────┼────────────────────────┼──────────
─────┤
│ dense_5 (Dense)                 │ (None, 1)              │
36 │
└─────────────────────────────────┴────────────────────────┴──────────
─────┘

 Total params: 106 (424.00 B)

 Trainable params: 106 (424.00 B)

 Non-trainable params: 0 (0.00 B)
```

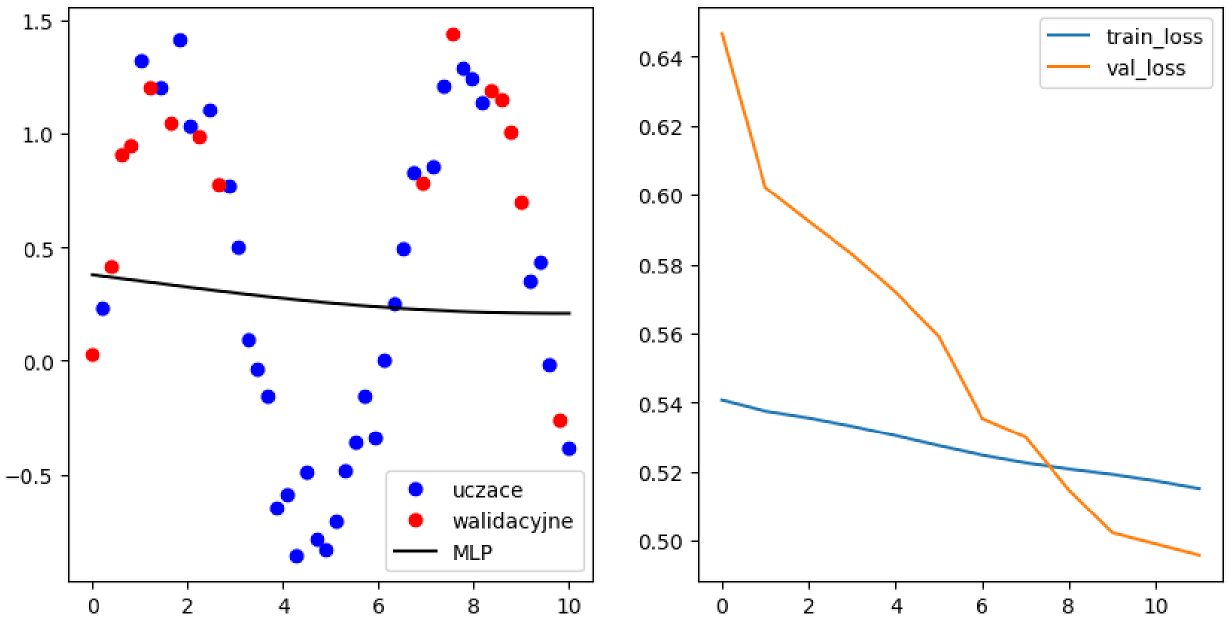
Wizualizacja uczenia

```
import io
import time
import base64
import IPython

def fig2b64(f):
  data = io.BytesIO()
  f.savefig(data, format='png')
  data.seek(0)
  return base64.b64encode(data.read()).decode()

data_xx = np.linspace(0, 10, 100)
fig = plt.figure(figsize=(10,5))
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)

train_loss = []
val_loss = []

model = model1
n_epochs = 300

hist = model.fit(data_x, data_y, epochs=10, verbose=0, batch_size=35,
validation_data=(val_x, val_y))
train_loss.append(model.evaluate(data_x, data_y, verbose=0))
val_loss.append(model.evaluate(val_x, val_y, verbose=0))
pred = model.predict(data_xx)
ax1.plot(data_x, data_y, 'bo', label='uczace')
ax1.plot(val_x, val_y, 'ro', label='walidacyjne')
ax1.plot(data_xx, pred, 'k-', label='MLP')
ax1.legend()
```

```python
ax2.plot(train_loss, label='train_loss')
ax2.plot(val_loss, label='val_loss')
ax2.legend()
data_str = fig2b64(fig)
rys = IPython.display.display_html(f'<img class="myimage"
src="data:image/png;base64,{data_str}"></img>', raw=True)

for i in range(n_epochs):
  IPython.display.clear_output(wait=True)
  #time.sleep(0.2)
  hist = model.fit(data_x, data_y, epochs=10, verbose=0,
batch_size=35, validation_data=(val_x, val_y))
  train_loss.append(model.evaluate(data_x, data_y, verbose=0))
  val_loss.append(model.evaluate(val_x, val_y, verbose=0))
  pred = model.predict(data_xx)
  ax1.clear()
  ax2.clear()
  ax1.plot(data_x, data_y, 'bo', label='uczace')
  ax1.plot(val_x, val_y, 'ro', label='walidacyjne')
  ax1.plot(data_xx, pred, 'k-', label='MLP')
  ax1.legend()
  ax2.plot(train_loss, label='train_loss')
  ax2.plot(val_loss, label='val_loss')
  ax2.legend()
  data_str = fig2b64(fig)
  rys = IPython.display.display_html(f'<img class="myimage"
src="data:image/png;base64,{data_str}"></img>', raw=True)
plt.close(1)
```

```
----------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Cell In[16], line 40
     38 IPython.display.clear_output(wait=True)
     39 #time.sleep(0.2)
---> 40 hist = model.fit(data_x, data_y, epochs=10, verbose=0,
batch_size=35, validation_data=(val_x, val_y))
     41 train_loss.append(model.evaluate(data_x, data_y, verbose=0))
     42 val_loss.append(model.evaluate(val_x, val_y, verbose=0))

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\utils\traceback_utils.py:117, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)
```

```
File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\backend\tensorflow\trainer.py:339, in
TensorFlowTrainer.fit(self, x, y, batch_size, epochs, verbose,
callbacks, validation_split, validation_data, shuffle, class_weight,
sample_weight, initial_epoch, steps_per_epoch, validation_steps,
validation_batch_size, validation_freq)
    328 if getattr(self, "_eval_epoch_iterator", None) is None:
    329     self._eval_epoch_iterator = TFEpochIterator(
    330         x=val_x,
    331         y=val_y,
   (...)
    337         shuffle=False,
    338     )
--> 339 val_logs = self.evaluate(
    340     x=val_x,
    341     y=val_y,
    342     sample_weight=val_sample_weight,
    343     batch_size=validation_batch_size or batch_size,
    344     steps=validation_steps,
    345     callbacks=callbacks,
    346     return_dict=True,
    347     _use_cached_eval_dataset=True,
    348 )
    349 val_logs = {
    350     "val_" + name: val for name, val in val_logs.items()
    351 }
    352 epoch_logs.update(val_logs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\utils\traceback_utils.py:117, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

KeyboardInterrupt:
```



Model sieci MLP

Większa sieć, dwie warstwy ukryte odpowiednio 100 i 50 neuronów.

```
x = Input(shape=(1,))
h1 = Dense(100, input_dim=1, use_bias=True, activation='tanh',
kernel_initializer='random_uniform',
bias_initializer='random_uniform')(x)
h2 = Dense(50, input_dim=1, use_bias=True, activation='tanh',
kernel_initializer='random_uniform',
bias_initializer='random_uniform')(h1)
y = Dense(1, use_bias=True, activation='linear',
kernel_initializer='random_uniform',
bias_initializer='random_uniform')(h2)
model2 = Model(inputs=x, outputs=y)
rms = tf.keras.optimizers.RMSprop(learning_rate=0.001)
model2.compile(loss='mse', optimizer=rms)
model2.summary()
```

```
Model: "functional_7"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ input_layer_3 (InputLayer)       │ (None, 1)               │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
```

```
───────┤
│ dense_6 (Dense)                  │ (None, 100)              │
200 │
├──────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ dense_7 (Dense)                  │ (None, 50)               │
5,050 │
├──────────────────────────────────┼──────────────────────────┼──────────
───────┤
│ dense_8 (Dense)                  │ (None, 1)                │
51 │
└──────────────────────────────────┴──────────────────────────┴──────────
───────┘

 Total params: 5,301 (20.71 KB)

 Trainable params: 5,301 (20.71 KB)

 Non-trainable params: 0 (0.00 B)
```

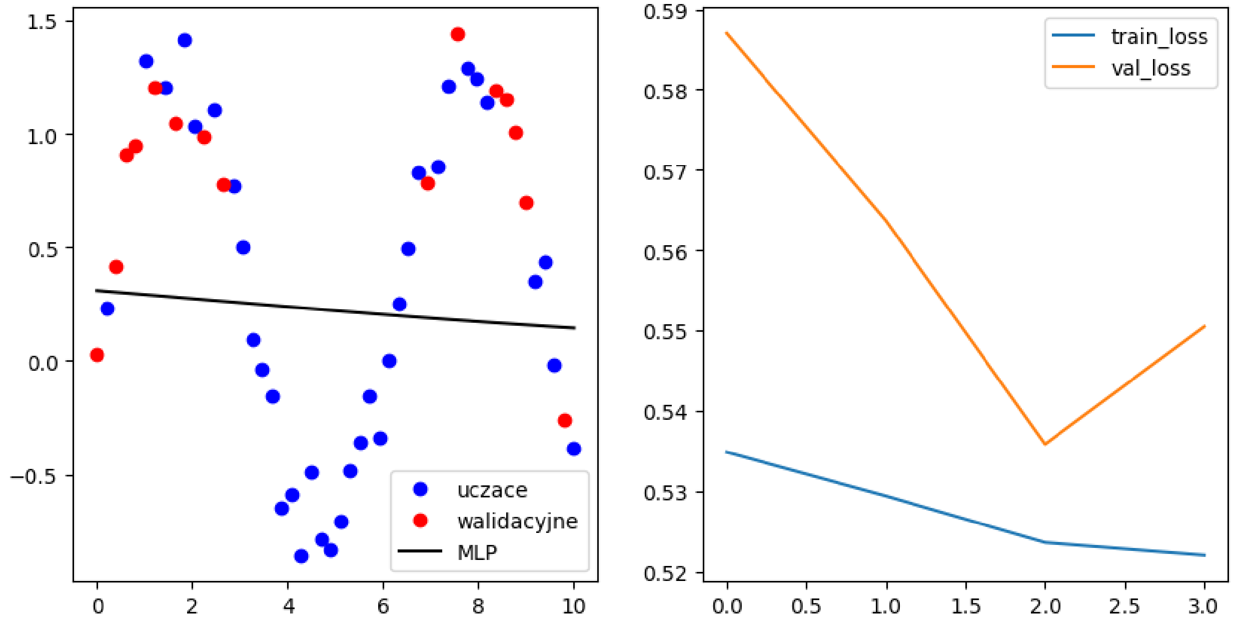
Wizualizacja uczenia

```
data_xx = np.linspace(0, 10, 100)
fig = plt.figure(figsize=(10,5))
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)

train_loss = []
val_loss = []

model = model2
n_epochs = 300

hist = model.fit(data_x, data_y, epochs=10, verbose=0, batch_size=35,
validation_data=(val_x, val_y))
train_loss.append(model.evaluate(data_x, data_y, verbose=0))
val_loss.append(model.evaluate(val_x, val_y, verbose=0))
pred = model.predict(data_xx)
ax1.plot(data_x, data_y, 'bo', label='uczace')
ax1.plot(val_x, val_y, 'ro', label='walidacyjne')
ax1.plot(data_xx, pred, 'k-', label='MLP')
ax1.legend()
ax2.plot(train_loss, label='train_loss')
ax2.plot(val_loss, label='val_loss')
ax2.legend()
data_str = fig2b64(fig)
rys = IPython.display.display_html(f'<img class="myimage"
src="data:image/png;base64,{data_str}"></img>', raw=True)

for i in range(n_epochs):
  IPython.display.clear_output(wait=True)
  #time.sleep(0.2)
  hist = model.fit(data_x, data_y, epochs=10, verbose=0,
batch_size=35, validation_data=(val_x, val_y))
  train_loss.append(model.evaluate(data_x, data_y, verbose=0))
  val_loss.append(model.evaluate(val_x, val_y, verbose=0))
  pred = model.predict(data_xx)
  ax1.clear()
  ax2.clear()
  ax1.plot(data_x, data_y, 'bo', label='uczace')
  ax1.plot(val_x, val_y, 'ro', label='walidacyjne')
  ax1.plot(data_xx, pred, 'k-', label='MLP')
  ax1.legend()
  ax2.plot(train_loss, label='train_loss')
  ax2.plot(val_loss, label='val_loss')
  ax2.legend()
  data_str = fig2b64(fig)
  rys = IPython.display.display_html(f'<img class="myimage"
src="data:image/png;base64,{data_str}"></img>', raw=True)
plt.close(1)
```

```
-----------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Cell In[18], line 29
     27 IPython.display.clear_output(wait=True)
     28 #time.sleep(0.2)
---> 29 hist = model.fit(data_x, data_y, epochs=10, verbose=0,
batch_size=35, validation_data=(val_x, val_y))
     30 train_loss.append(model.evaluate(data_x, data_y, verbose=0))
     31 val_loss.append(model.evaluate(val_x, val_y, verbose=0))

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\utils\traceback_utils.py:117, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\backend\tensorflow\trainer.py:312, in
TensorFlowTrainer.fit(self, x, y, batch_size, epochs, verbose,
callbacks, validation_split, validation_data, shuffle, class_weight,
sample_weight, initial_epoch, steps_per_epoch, validation_steps,
validation_batch_size, validation_freq)
    310 callbacks.on_epoch_begin(epoch)
    311 with epoch_iterator.catch_stop_iteration():
```

```
--> 312     for step, iterator in epoch_iterator.enumerate_epoch():
    313         callbacks.on_train_batch_begin(step)
    314         logs = self.train_function(iterator)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\backend\tensorflow\trainer.py:645, in
TFEpochIterator.enumerate_epoch(self)
    643         yield step, self._current_iterator
    644 else:
--> 645     iterator = iter(self._distributed_dataset)
    646     if self.num_batches:
    647         for step in range(
    648             0, self.num_batches, self.steps_per_execution
    649         ):

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\data\ops\dataset_ops.py:501, in
DatasetV2.__iter__(self)
    499 if context.executing_eagerly() or ops.inside_function():
    500   with ops.colocate_with(self._variant_tensor):
--> 501     return iterator_ops.OwnedIterator(self)
    502 else:
    503   raise RuntimeError("`tf.data.Dataset` only supports Python-
style "
    504                      "iteration in eager mode or within
tf.function.")

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\data\ops\iterator_ops.py:705, in
OwnedIterator.__init__(self, dataset, components, element_spec)
    701   if (components is not None or element_spec is not None):
    702     raise ValueError(
    703         "When `dataset` is provided, `element_spec` and
`components` must "
    704         "not be specified.")
--> 705   self._create_iterator(dataset)
    707 self._get_next_call_count = 0

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\data\ops\iterator_ops.py:744, in
OwnedIterator._create_iterator(self, dataset)
    741   assert len(fulltype.args[0].args[0].args) == len(
    742       self._flat_output_types)
    743   self._iterator_resource.op.experimental_set_type(fulltype)
--> 744 gen_dataset_ops.make_iterator(ds_variant,
self._iterator_resource)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\ops\gen_dataset_ops.py:3478, in
make_iterator(dataset, iterator, name)
```

```
   3476 if tld.is_eager:
   3477   try:
-> 3478     _result = pywrap_tfe.TFE_Py_FastPathExecute(
   3479       _ctx, "MakeIterator", name, dataset, iterator)
   3480     return _result
   3481   except _core._NotOkStatusException as e:

KeyboardInterrupt:
```



Kiedy zakończyć uczenie?

Jednym z kluczowych aspektów (poza doborem architektury) jest zdecydowanie kiedy zakończyć uczenie sieci neuronowej. Najpopularniejsza technika polega na obserwacji wartości błedów osiąganych na zbiorze uczącym i zbiorze walidacyjnym. Gdy błąd na zbiorze walidacyjnym przestaje maleć (zazwyczaj zaczyna rosnąć) to znaczy, że sieć zaczyna się przeuczać (traci swoje zdalonośći generalizacyjne) i wtedy należy zakończyć proces uczenia. Taka strategia nazywa się strategią wczesnego zatrzymania (*early stopping*).

Problemy klasyfikacyjne z wieloma klasami

Baza danych irysów zawiera przykłady z trzech klas. Zwróć uwagę na odpowiednie zakodowanie informacji o etykietach klas dla przykładów za pomocą funkcji `keras.utils.to_categorical`

```
from sklearn import datasets
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import SGD
```

```
iris_db = datasets.load_iris()
print(dir(iris_db))
print(type(iris_db.data)) #dane jako macierz numpy
print(iris_db.data.shape) #kazdy przyklad w wierszu
print(iris_db.feature_names) #nazwy atrybutow (sygnaly wejsciowe
sieci)
print(iris_db.data[:10,:]) #podglad
print(iris_db.target_names) #nazwy trzech klas
print(iris_db.target) #etykiety klas zakodowane numerycznie jako 0, 1,
2

#d: zakodowane etykiety klas w sposob umozliwiajacy uczenie sieci
d = tf.keras.utils.to_categorical(iris_db.target, num_classes=3)
print(type(d))
print(d.shape)
print(d[:5,:])
```

```
['DESCR', 'data', 'data_module', 'feature_names', 'filename', 'frame',
'target', 'target_names']
<class 'numpy.ndarray'>
(150, 4)
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal
width (cm)']
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
['setosa' 'versicolor' 'virginica']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2
 2 2]
<class 'numpy.ndarray'>
(150, 3)
[[1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
```

Warstwa wyjściowa softmax

W warstwie wyjściowej softmax, każdy neuron realizuje sumę ważoną dochodzących do niego sygnałów wejściowych. Następnie, odpowiedzi wszystkich neuronów wyjściowych są przetwarzane zgodnie ze wzorem

$$P(y=j|\bf{x})=\frac{e^{\bf{x}^{T}\bf{w}_j}}{\sum_{k=1}^{K}{e^{\bf{x}^{T}\bf{w}_k}}}$$

gdzie K to liczba neuronów wyjściowych (liczba klas w problemie klasyfikacyjnym), w_j to wagi j-tego neuronu wyjściowego, x to sygnały wejściowe neuronów z warstwy wyjściowej (odpowiedzi poprzedniej warstwy).

Wartości te mogą być interpretowane jako prawdopodobieństwa przynależności danego przykładu (podanego na wejście sieci) do danej klasy, którą reprezentuje j-ty neuron wyjściowy.

Dla takiej warstwy wyjściowej, funkcją straty używaną w trakcie uczenia jest zazwyczaj `categorical_entropy`, która mierzy podobieństwo dwóch rozkładów prawdopodobieństwa przynależności danych trenujących do klas: rzeczywisty (na podstawie zbioru trenującego) oraz ten realizowany przez sieć.

Zapis i odczyt modelu do/z pliku

W poniższym przykładzie zwróć uwagę na zapis modelu do pliku i jego ponowne wczytanie.

```python
model = Sequential()
model.add(Dense(30, activation='tanh', input_dim=4))
model.add(Dense(3, activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

model.fit(iris_db.data, d,
          epochs=200,
          batch_size=10)

score = model.evaluate(iris_db.data, d, batch_size=10)
print('model koncowy:',score)

#sprawdzenie czy dziala zapis/odczyt modelu z pliku
model.save('my_model.h5')
from tensorflow.keras.models import load_model
model2 = load_model('my_model.h5')
score2 = model2.evaluate(iris_db.data, d, batch_size=10)
print('model z pliku:',score2)
```

```
c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\optimizers\base_optimizer.py:33: UserWarning:
Argument `decay` is no longer supported and will be ignored.
  warnings.warn(
----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[20], line 5
      2 model.add(Dense(30, activation='tanh', input_dim=4))
      3 model.add(Dense(3, activation='softmax'))
----> 5 sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
      6 model.compile(loss='categorical_crossentropy',
      7               optimizer=sgd,
      8               metrics=['accuracy'])
     10 model.fit(iris_db.data, d,
     11           epochs=200,
     12           batch_size=10)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\optimizers\sgd.py:60, in SGD.__init__(self,
learning_rate, momentum, nesterov, weight_decay, clipnorm, clipvalue,
global_clipnorm, use_ema, ema_momentum, ema_overwrite_frequency,
loss_scale_factor, gradient_accumulation_steps, name, **kwargs)
     43 def __init__(
     44     self,
     45     learning_rate=0.01,
   (...)
     58     **kwargs,
     59 ):
---> 60     super().__init__(
     61         learning_rate=learning_rate,
     62         name=name,
     63         weight_decay=weight_decay,
     64         clipnorm=clipnorm,
     65         clipvalue=clipvalue,
     66         global_clipnorm=global_clipnorm,
     67         use_ema=use_ema,
     68         ema_momentum=ema_momentum,
     69         ema_overwrite_frequency=ema_overwrite_frequency,
     70         loss_scale_factor=loss_scale_factor,
     71
gradient_accumulation_steps=gradient_accumulation_steps,
     72         **kwargs,
     73     )
     74     if not isinstance(momentum, float) or momentum < 0 or
momentum > 1:
     75         raise ValueError("`momentum` must be a float between
[0, 1].")

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\backend\tensorflow\optimizer.py:22, in
TFOptimizer.__init__(self, *args, **kwargs)
     21 def __init__(self, *args, **kwargs):
```

```
---> 22     super().__init__(*args, **kwargs)
     23     self._distribution_strategy = tf.distribute.get_strategy()

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\optimizers\base_optimizer.py:37, in
BaseOptimizer.__init__(self, learning_rate, weight_decay, clipnorm,
clipvalue, global_clipnorm, use_ema, ema_momentum,
ema_overwrite_frequency, loss_scale_factor,
gradient_accumulation_steps, name, **kwargs)
     33     warnings.warn(
     34         "Argument `decay` is no longer supported and will be
ignored."
     35     )
     36 if kwargs:
---> 37     raise ValueError(f"Argument(s) not recognized: {kwargs}")
     39 if name is None:
     40     name = auto_name(self.__class__.__name__)

ValueError: Argument(s) not recognized: {'lr': 0.01}
```

Zadanie 1

Naucz sieć diagnozować cukrzycę.

- Wykorzytaj dane z pliku `pima-indians-diabetes.data.csv`. Dane są również dostępne w `sklearn`. Zaimportuje je jako `diab_db=datasets.load_diabetes()`

- Podziel dostępne dane losowo na dane uczące i testowe (walidacyjne) w proporcji 70% / 30%. Podział danych jest wykonywany raz i jest używany niezmieniony w dalszych obliczeniach dla wszystkich sieci.

- Dobierz jak najlepsze parametry uczenia oraz architektury sieci z jedną oraz z dwiema warstwami ukrytymi (po jednej na każdy rodzaj). Jakość działania sieci oceniamy na podstawie jej wyników na danych testowych. Postaraj się w odpowiednim momencie zatrzymać proces uczenia.

- Czy sieć z dwiema warstwami ukrytymi działa lepiej niż sieć z jedną warstwą ukrytą? Porównania i wnioski przedstaw na podstawie uśrednionych wyników dziesięciu sieci każdego rodzaju (tzn. najpierw ustal architekturę sieci, następnie przeprowadź 10 procesów trenownia, startując za każdym razem z losowych początkowych wag).

- W dostarczonym kodzie umieść proces uczenia i testowania wybranych architektur sieci.

Uwaga: Przy uczeniu większych modeli warto wykonywać obliczenia z wykorzystaniem karty graficznej. Aby uruchomić notatnik z wykorzystaniem GPU należy wejść do Edit->Notebook settings i zmienić Hardware accelerator na GPU.

TWÓJ KOD TUTAJ:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_diabetes
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import IPython
import io
import base64

def fig2b64(f):
  data = io.BytesIO()
  f.savefig(data, format='png')
  data.seek(0)
  return base64.b64encode(data.read()).decode()
data_str = fig2b64(fig)
rys = IPython.display.display_html(f'<img class="myimage"
src="data:image/png;base64,{data_str}"></img>', raw=True)

NUM_RUNS = 10

diab_db = load_diabetes()
X = diab_db.data
y = diab_db.target

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

mse_single_results = []
mse_double_results = []
for _ in range(NUM_RUNS):
    IPython.display.clear_output(wait=True)
    model_single = Sequential()
    model_single.add(Dense(10, activation='relu',
input_shape=(X_train.shape[1],)))
    model_single.add(Dense(1, activation='linear'))
    model_single.compile(optimizer='adam', loss='mse')
    history_single = model_single.fit(X_train, y_train,
validation_data=(X_test, y_test), epochs=100, batch_size=32,
verbose=0)
    y_pred_single = model_single.predict(X_test)
    mse_single = np.mean((y_pred_single - y_test) ** 2)
    mse_single_results.append(mse_single)
    plt.plot(history_single.history['loss'], label='Train')
    plt.plot(history_single.history['val_loss'], label='Test')
    plt.xlabel('Epoch')
    plt.ylabel('MSE')
    plt.title('Model z jedną warstwą ukrytą - Proces uczenia')
    plt.legend()
    plt.show()
```

```
----------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Cell In[21], line 42
     40 model_single.add(Dense(1, activation='linear'))
     41 model_single.compile(optimizer='adam', loss='mse')
---> 42 history_single = model_single.fit(X_train, y_train,
validation_data=(X_test, y_test), epochs=100, batch_size=32,
verbose=0)
     43 y_pred_single = model_single.predict(X_test)
     44 mse_single = np.mean((y_pred_single - y_test) ** 2)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\utils\traceback_utils.py:117, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\backend\tensorflow\trainer.py:314, in
TensorFlowTrainer.fit(self, x, y, batch_size, epochs, verbose,
callbacks, validation_split, validation_data, shuffle, class_weight,
sample_weight, initial_epoch, steps_per_epoch, validation_steps,
validation_batch_size, validation_freq)
    312 for step, iterator in epoch_iterator.enumerate_epoch():
    313     callbacks.on_train_batch_begin(step)
--> 314     logs = self.train_function(iterator)
    315     logs = self._pythonify_logs(logs)
    316     callbacks.on_train_batch_end(step, logs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\util\traceback_utils.py:150, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
```

```
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
polymorphic_function.py:833, in Function.__call__(self, *args, **kwds)
    830 compiler = "xla" if self._jit_compile else "nonXla"
    832 with OptionalXlaContext(self._jit_compile):
--> 833   result = self._call(*args, **kwds)
    835 new_tracing_count = self.experimental_get_tracing_count()
    836 without_tracing = (tracing_count == new_tracing_count)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
polymorphic_function.py:889, in Function._call(self, *args, **kwds)
    886 try:
    887   # This is the first call of __call__, so we have to
initialize.
    888   initializers = []
--> 889   self._initialize(args, kwds,
add_initializers_to=initializers)
    890 finally:
    891   # At this point we know that the initialization is complete
(or less
    892   # interestingly an exception was raised) so we no longer
need a lock.
    893   self._lock.release()

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
polymorphic_function.py:696, in Function._initialize(self, args, kwds,
add_initializers_to)
    691 self._variable_creation_config =
self._generate_scoped_tracing_options(
    692     variable_capturing_scope,
    693     tracing_compilation.ScopeType.VARIABLE_CREATION,
    694 )
    695 # Force the definition of the function for these arguments
--> 696 self._concrete_variable_creation_fn =
tracing_compilation.trace_function(
    697     args, kwds, self._variable_creation_config
    698 )
    700 def invalid_creator_scope(*unused_args, **unused_kwds):
    701   """Disables variable creation."""

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
tracing_compilation.py:178, in trace_function(args, kwargs,
tracing_options)
    175     args = tracing_options.input_signature
    176     kwargs = {}
--> 178   concrete_function = _maybe_define_function(
    179       args, kwargs, tracing_options
    180   )
    182 if not tracing_options.bind_graph_to_function:
    183   concrete_function._garbage_collector.release()  # pylint:
disable=protected-access

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
tracing_compilation.py:283, in _maybe_define_function(args, kwargs,
tracing_options)
    281 else:
    282   target_func_type = lookup_func_type
--> 283 concrete_function = _create_concrete_function(
    284     target_func_type, lookup_func_context, func_graph,
tracing_options
    285 )
    287 if tracing_options.function_cache is not None:
    288   tracing_options.function_cache.add(
    289       concrete_function, current_func_context
    290   )

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
tracing_compilation.py:310, in
_create_concrete_function(function_type, type_context, func_graph,
tracing_options)
    303   placeholder_bound_args =
function_type.placeholder_arguments(
    304       placeholder_context
    305   )
    307 disable_acd = tracing_options.attributes and
tracing_options.attributes.get(
    308     attributes_lib.DISABLE_ACD, False
    309 )
--> 310 traced_func_graph = func_graph_module.func_graph_from_py_func(
    311     tracing_options.name,
    312     tracing_options.python_function,
    313     placeholder_bound_args.args,
    314     placeholder_bound_args.kwargs,
    315     None,
    316     func_graph=func_graph,
    317     add_control_dependencies=not disable_acd,
    318     arg_names=function_type_utils.to_arg_names(function_type),
    319     create_placeholders=False,
    320 )
    322 transform.apply_func_graph_transforms(traced_func_graph)
    324 graph_capture_container = traced_func_graph.function_captures
```

```
File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\framework\func_graph.py:1059, in
func_graph_from_py_func(name, python_func, args, kwargs, signature,
func_graph, add_control_dependencies, arg_names, op_return_value,
collections, capture_by_value, create_placeholders)
   1056   return x
   1058 _, original_func = tf_decorator.unwrap(python_func)
-> 1059 func_outputs = python_func(*func_args, **func_kwargs)
   1061 # invariant: `func_outputs` contains only Tensors,
CompositeTensors,
   1062 # TensorArrays and `None`s.
   1063 func_outputs =
variable_utils.convert_variables_to_tensors(func_outputs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
polymorphic_function.py:599, in
Function._generate_scoped_tracing_options.<locals>.wrapped_fn(*args,
**kwds)
    595 with default_graph._variable_creator_scope(scope,
priority=50):  # pylint: disable=protected-access
    596   # __wrapped__ allows AutoGraph to swap in a converted
function. We give
    597   # the function a weak reference to itself to avoid a
reference cycle.
    598   with OptionalXlaContext(compile_with_xla):
--> 599     out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    600   return out

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
autograph_util.py:41, in
py_func_from_autograph.<locals>.autograph_handler(*args, **kwargs)
     39 """Calls a converted version of original_func."""
     40 try:
---> 41   return api.converted_call(
     42       original_func,
     43       args,
     44       kwargs,
     45       options=converter.ConversionOptions(
     46           recursive=True,
     47           optional_features=autograph_options,
     48           user_requested=True,
     49       ))
     50 except Exception as e:  # pylint:disable=broad-except
     51   if hasattr(e, "ag_error_metadata"):

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\autograph\impl\api.py:339, in
converted_call(f, args, kwargs, caller_fn_scope, options)
```

```
    337 if is_autograph_artifact(f):
    338   logging.log(2, 'Permanently allowed: %s: AutoGraph
artifact', f)
--> 339   return _call_unconverted(f, args, kwargs, options)
    341 # If this is a partial, unwrap it and redo all the checks.
    342 if isinstance(f, functools.partial):

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\autograph\impl\api.py:459, in
_call_unconverted(f, args, kwargs, options, update_cache)
    456   return f.__self__.call(args, kwargs)
    458 if kwargs is not None:
--> 459   return f(*args, **kwargs)
    460 return f(*args)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\autograph\impl\api.py:643, in
do_not_convert.<locals>.wrapper(*args, **kwargs)
    641 def wrapper(*args, **kwargs):
    642   with ag_ctx.ControlStatusCtx(status=ag_ctx.Status.DISABLED):
--> 643     return func(*args, **kwargs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\backend\tensorflow\trainer.py:117, in
TensorFlowTrainer.make_train_function.<locals>.one_step_on_iterator(it
erator)
    115 """Runs a single training step given a Dataset iterator."""
    116 data = next(iterator)
--> 117 outputs = self.distribute_strategy.run(
    118     one_step_on_data, args=(data,)
    119 )
    120 outputs = reduce_per_replica(
    121     outputs,
    122     self.distribute_strategy,
    123     reduction="auto",
    124 )
    125 return outputs

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\distribute\distribute_lib.py:1673, in
StrategyBase.run(***failed resolving arguments***)
   1668 with self.scope():
   1669   # tf.distribute supports Eager functions, so AutoGraph
should not be
   1670   # applied when the caller is also in Eager mode.
   1671   fn = autograph.tf_convert(
   1672       fn, autograph_ctx.control_status_ctx(),
convert_by_default=False)
-> 1673   return self._extended.call_for_each_replica(fn, args=args,
kwargs=kwargs)
```

```
File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\distribute\distribute_lib.py:3263, in
StrategyExtendedV1.call_for_each_replica(self, fn, args, kwargs)
   3261   kwargs = {}
   3262 with self._container_strategy().scope():
-> 3263   return self._call_for_each_replica(fn, args, kwargs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\distribute\distribute_lib.py:4061, in
_DefaultDistributionExtended._call_for_each_replica(self, fn, args,
kwargs)
   4059 def _call_for_each_replica(self, fn, args, kwargs):
   4060   with ReplicaContext(self._container_strategy(),
replica_id_in_sync_group=0):
-> 4061     return fn(*args, **kwargs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\util\traceback_utils.py:150, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
polymorphic_function.py:833, in Function.__call__(self, *args, **kwds)
    830 compiler = "xla" if self._jit_compile else "nonXla"
    832 with OptionalXlaContext(self._jit_compile):
--> 833   result = self._call(*args, **kwds)
    835 new_tracing_count = self.experimental_get_tracing_count()
    836 without_tracing = (tracing_count == new_tracing_count)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
polymorphic_function.py:906, in Function._call(self, *args, **kwds)
    902     pass  # Fall through to cond-based initialization.
    903   else:
    904     # Lifting succeeded, so variables are initialized and we
can run the
    905     # no_variable_creation function.
--> 906     return tracing_compilation.call_function(
    907         args, kwds, self._no_variable_creation_config
    908     )
    909 else:
    910   bound_args =
self._concrete_variable_creation_fn.function_type.bind(
    911       *args, **kwds
    912   )

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
tracing_compilation.py:132, in call_function(args, kwargs,
tracing_options)
    130 args = args if args else ()
    131 kwargs = kwargs if kwargs else {}
--> 132 function = trace_function(
    133     args=args, kwargs=kwargs, tracing_options=tracing_options
    134 )
    136 # Bind it ourselves to skip unnecessary canonicalization of
default call.
    137 bound_args = function.function_type.bind(*args, **kwargs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
tracing_compilation.py:178, in trace_function(args, kwargs,
tracing_options)
    175     args = tracing_options.input_signature
    176     kwargs = {}
--> 178   concrete_function = _maybe_define_function(
    179       args, kwargs, tracing_options
    180   )
    182 if not tracing_options.bind_graph_to_function:
    183   concrete_function._garbage_collector.release()  # pylint:
disable=protected-access

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
tracing_compilation.py:283, in _maybe_define_function(args, kwargs,
tracing_options)
    281 else:
    282   target_func_type = lookup_func_type
--> 283 concrete_function = _create_concrete_function(
    284     target_func_type, lookup_func_context, func_graph,
tracing_options
    285 )
    287 if tracing_options.function_cache is not None:
    288   tracing_options.function_cache.add(
    289       concrete_function, current_func_context
    290   )

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
tracing_compilation.py:310, in
_create_concrete_function(function_type, type_context, func_graph,
tracing_options)
    303   placeholder_bound_args =
function_type.placeholder_arguments(
    304       placeholder_context
    305   )
    307 disable_acd = tracing_options.attributes and
tracing_options.attributes.get(
    308     attributes_lib.DISABLE_ACD, False
    309 )
--> 310 traced_func_graph = func_graph_module.func_graph_from_py_func(
    311     tracing_options.name,
    312     tracing_options.python_function,
    313     placeholder_bound_args.args,
    314     placeholder_bound_args.kwargs,
    315     None,
    316     func_graph=func_graph,
    317     add_control_dependencies=not disable_acd,
    318     arg_names=function_type_utils.to_arg_names(function_type),
    319     create_placeholders=False,
    320 )
    322 transform.apply_func_graph_transforms(traced_func_graph)
    324 graph_capture_container = traced_func_graph.function_captures

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\framework\func_graph.py:1059, in
func_graph_from_py_func(name, python_func, args, kwargs, signature,
func_graph, add_control_dependencies, arg_names, op_return_value,
collections, capture_by_value, create_placeholders)
   1056   return x
   1058 _, original_func = tf_decorator.unwrap(python_func)
-> 1059 func_outputs = python_func(*func_args, **func_kwargs)
   1061 # invariant: `func_outputs` contains only Tensors,
CompositeTensors,
   1062 # TensorArrays and `None`s.
   1063 func_outputs =
variable_utils.convert_variables_to_tensors(func_outputs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\eager\polymorphic_function\
polymorphic_function.py:599, in
Function._generate_scoped_tracing_options.<locals>.wrapped_fn(*args,
**kwds)
    595 with default_graph._variable_creator_scope(scope,
priority=50):  # pylint: disable=protected-access
    596   # __wrapped__ allows AutoGraph to swap in a converted
function. We give
    597   # the function a weak reference to itself to avoid a
reference cycle.
    598   with OptionalXlaContext(compile_with_xla):
--> 599     out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    600   return out

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
```

```
packages\tensorflow\python\eager\polymorphic_function\
autograph_util.py:41, in
py_func_from_autograph.<locals>.autograph_handler(*args, **kwargs)
     39 """Calls a converted version of original_func."""
     40 try:
---> 41   return api.converted_call(
     42       original_func,
     43       args,
     44       kwargs,
     45       options=converter.ConversionOptions(
     46           recursive=True,
     47           optional_features=autograph_options,
     48           user_requested=True,
     49       ))
     50 except Exception as e:  # pylint:disable=broad-except
     51   if hasattr(e, "ag_error_metadata"):

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\autograph\impl\api.py:331, in
converted_call(f, args, kwargs, caller_fn_scope, options)
    329 if conversion.is_in_allowlist_cache(f, options):
    330   logging.log(2, 'Allowlisted %s: from cache', f)
--> 331   return _call_unconverted(f, args, kwargs, options, False)
    333 if ag_ctx.control_status_ctx().status ==
ag_ctx.Status.DISABLED:
    334   logging.log(2, 'Allowlisted: %s: AutoGraph is disabled in
context', f)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\autograph\impl\api.py:459, in
_call_unconverted(f, args, kwargs, options, update_cache)
    456   return f.__self__.call(args, kwargs)
    458 if kwargs is not None:
--> 459   return f(*args, **kwargs)
    460 return f(*args)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\autograph\impl\api.py:643, in
do_not_convert.<locals>.wrapper(*args, **kwargs)
    641 def wrapper(*args, **kwargs):
    642   with ag_ctx.ControlStatusCtx(status=ag_ctx.Status.DISABLED):
--> 643     return func(*args, **kwargs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\backend\tensorflow\trainer.py:104, in
TensorFlowTrainer.make_train_function.<locals>.one_step_on_data(data)
    101 @tf.autograph.experimental.do_not_convert
    102 def one_step_on_data(data):
    103     """Runs a single training step on a batch of data."""
--> 104     return self.train_step(data)
```

```
File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\backend\tensorflow\trainer.py:57, in
TensorFlowTrainer.train_step(self, data)
     53     y_pred = self(x)
     54 loss = self.compute_loss(
     55     x=x, y=y, y_pred=y_pred, sample_weight=sample_weight
     56 )
---> 57 self._loss_tracker.update_state(
     58     loss, sample_weight=tf.shape(tree.flatten(x)[0])[0]
     59 )
     60 if self.optimizer is not None:
     61     loss = self.optimizer.scale_loss(loss)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\metrics\reduction_metrics.py:141, in
Mean.update_state(self, values, sample_weight)
    137 def update_state(self, values, sample_weight=None):
    138     values, sample_weight = reduce_to_samplewise_values(
    139         values, sample_weight, reduce_fn=ops.mean,
dtype=self.dtype
    140     )
--> 141     self.total.assign(self.total + ops.sum(values))
    142     if len(values.shape) >= 1:
    143         num_samples = ops.shape(values)[0]

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\ops\numpy.py:5894, in sum(x, axis, keepdims)
   5892 if any_symbolic_tensors((x,)):
   5893     return Sum(axis=axis, keepdims=keepdims).symbolic_call(x)
-> 5894 return backend.numpy.sum(x, axis=axis, keepdims=keepdims)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\backend\tensorflow\numpy.py:2339, in sum(x, axis,
keepdims)
   2335 if isinstance(x, tf.SparseTensor):
   2336     return tf.sparse.reduce_sum(
   2337         x, axis=axis, keepdims=keepdims, output_is_sparse=True
   2338     )
-> 2339 return tf.reduce_sum(x, axis=axis, keepdims=keepdims)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\ops\weak_tensor_ops.py:88, in
weak_tensor_unary_op_wrapper.<locals>.wrapper(*args, **kwargs)
     86 def wrapper(*args, **kwargs):
     87   if not ops.is_auto_dtype_conversion_enabled():
---> 88     return op(*args, **kwargs)
     89   bound_arguments = signature.bind(*args, **kwargs)
     90   bound_arguments.apply_defaults()
```

```
File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\util\traceback_utils.py:150, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\util\dispatch.py:1260, in
add_dispatch_support.<locals>.decorator.<locals>.op_dispatch_handler(*
args, **kwargs)
   1258 # Fallback dispatch system (dispatch v1):
   1259 try:
-> 1260   return dispatch_target(*args, **kwargs)
   1261 except (TypeError, ValueError):
   1262   # Note: convert_to_eager_tensor currently raises a
ValueError, not a
   1263   # TypeError, when given unexpected types.  So we need to
catch both.
   1264   result = dispatch(op_dispatch_handler, args, kwargs)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\ops\math_ops.py:2209, in
reduce_sum(input_tensor, axis, keepdims, name)
   2146 @tf_export("math.reduce_sum", "reduce_sum", v1=[])
   2147 @dispatch.add_dispatch_support
   2148 def reduce_sum(input_tensor, axis=None, keepdims=False,
name=None):
   2149   """Computes the sum of elements across dimensions of a
tensor.
   2150 
   2151   This is the reduction operation for the elementwise
`tf.math.add` op.
   (...)
   2206   @end_compatibility
   2207   """
-> 2209   return reduce_sum_with_dims(input_tensor, axis, keepdims,
name,
   2210                               _ReductionDims(input_tensor,
axis))

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\ops\math_ops.py:2221, in
reduce_sum_with_dims(input_tensor, axis, keepdims, name, dims)
   2213 def reduce_sum_with_dims(input_tensor,
   2214                          axis=None,
   2215                          keepdims=False,
   2216                          name=None,
   2217                             dims=None):
   2218   keepdims = False if keepdims is None else bool(keepdims)
   2219   return _may_reduce_to_scalar(
   2220       keepdims, axis,
-> 2221       gen_math_ops._sum(input_tensor, dims, keepdims,
name=name))

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\ops\gen_math_ops.py:12388, in _sum(input,
axis, keep_dims, name)
  12386   keep_dims = False
  12387 keep_dims = _execute.make_bool(keep_dims, "keep_dims")
> 12388 _, _, _op, _outputs = _op_def_library._apply_op_helper(
  12389       "Sum", input=input, reduction_indices=axis,
keep_dims=keep_dims,
  12390              name=name)
  12391 _result = _outputs[:]
  12392 if _execute.must_record_gradient():

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\framework\op_def_library.py:796, in
_apply_op_helper(op_type_name, name, **keywords)
    791 must_colocate_inputs = [val for arg, val in
zip(op_def.input_arg, inputs)
    792                         if arg.is_ref]
    793 with _MaybeColocateWith(must_colocate_inputs):
    794   # Add Op to graph
    795   # pylint: disable=protected-access
--> 796   op = g._create_op_internal(op_type_name, inputs,
dtypes=None,
    797                              name=scope,
input_types=input_types,
    798                              attrs=attr_protos, op_def=op_def)
    800 # `outputs` is returned as a separate return value so that the
output
    801 # tensors can the `op` per se can be decoupled so that the
    802 # `op_callbacks` can function properly. See
framework/op_callbacks.py
    803 # for more details.
    804 outputs = op.outputs

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\framework\func_graph.py:670, in
FuncGraph._create_op_internal(self, op_type, inputs, dtypes,
input_types, name, attrs, op_def, compute_device)
    668   inp = self.capture(inp)
    669   captured_inputs.append(inp)
--> 670 return super()._create_op_internal(  # pylint:
disable=protected-access
    671     op_type, captured_inputs, dtypes, input_types, name,
attrs, op_def,
    672     compute_device)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\framework\ops.py:2682, in
Graph._create_op_internal(self, op_type, inputs, dtypes, input_types,
name, attrs, op_def, compute_device)
   2679 # _create_op_helper mutates the new Operation.
`_mutation_lock` ensures a
   2680 # Session.run call cannot occur between creating and mutating
the op.
   2681 with self._mutation_lock():
-> 2682   ret = Operation.from_node_def(
   2683       node_def,
   2684       self,
   2685       inputs=inputs,
   2686       output_types=dtypes,
   2687       control_inputs=control_inputs,
   2688       input_types=input_types,
   2689       original_op=self._default_original_op,
   2690       op_def=op_def,
   2691   )
   2692   self._create_op_helper(ret, compute_device=compute_device)
   2693 return ret

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\framework\ops.py:1177, in
Operation.from_node_def(***failed resolving arguments***)
   1174     control_input_ops.append(control_op)
   1176 # Initialize c_op from node_def and other inputs
-> 1177 c_op = _create_c_op(g, node_def, inputs, control_input_ops,
op_def=op_def)
   1178 self = Operation(c_op, SymbolicTensor)
   1179 self._init(g)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\util\traceback_utils.py:150, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\ziggs\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tensorflow\python\framework\ops.py:1034, in
_create_c_op(graph, node_def, inputs, control_inputs, op_def,
extract_traceback)
   1030   pywrap_tf_session.TF_SetAttrValueProto(op_desc,
compat.as_str(name),
   1031                                          serialized)
   1033 try:
-> 1034   c_op = pywrap_tf_session.TF_FinishOperation(op_desc)
   1035 except errors.InvalidArgumentError as e:
   1036   # Convert to ValueError for backwards compatibility.
   1037   raise ValueError(e.message)

KeyboardInterrupt: 
```

TWOJE KOMENTARZE I WNIOSKI