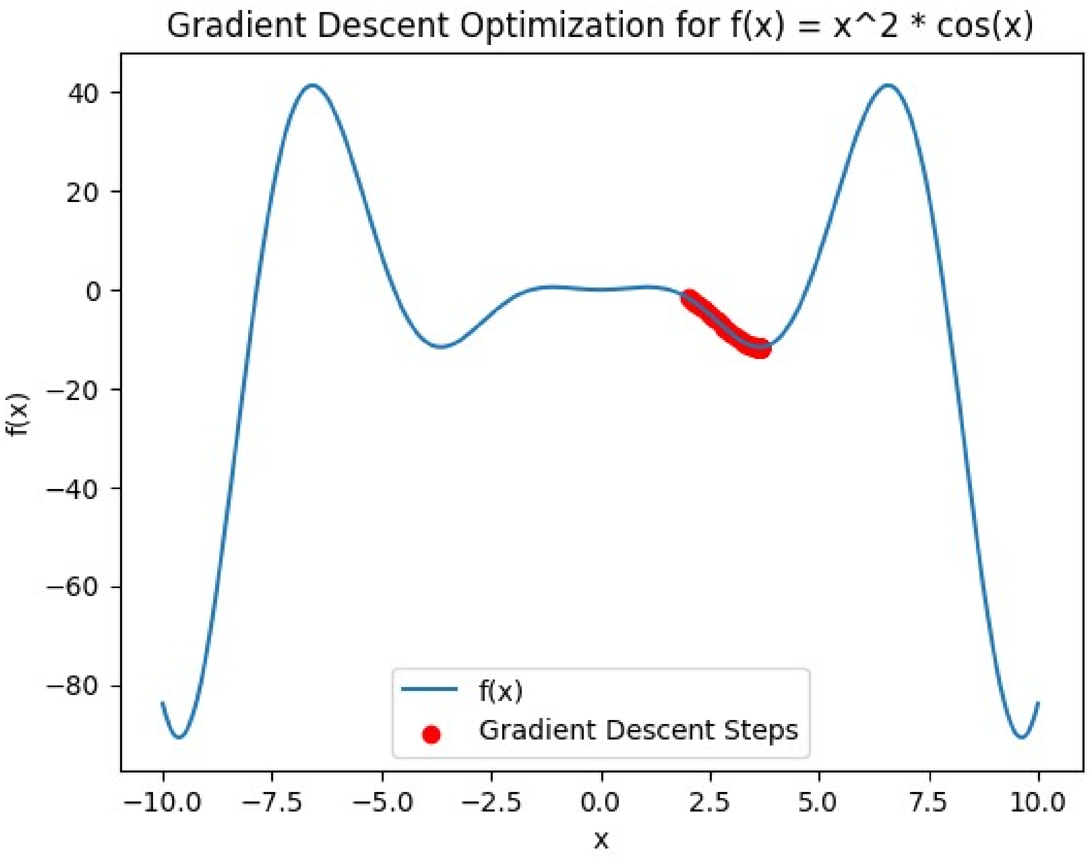
```
import numpy as np
import matplotlib.pyplot as plt

def gradient_descent(f, grad_f, theta_init, learning_rate,
iterations):
    theta = theta_init
    history = [theta]
    for i in range(iterations):
        theta -= learning_rate * grad_f(theta)
        history.append(theta)
    return theta, history

# Definicja funkcji i jej pochodnej
f = lambda x: x**2 * np.cos(x)
grad_f = lambda x: 2*x*np.cos(x) - x**2*np.sin(x)

# Parametry
theta_init = 2.0
learning_rate = 0.01
iterations = 100

# Optymalizacja
optimal_theta, history = gradient_descent(f, grad_f, theta_init,
learning_rate, iterations)

# Wizualizacja
x = np.linspace(-10, 10, 400)
y = f(x)

plt.plot(x, y, label='f(x)')
plt.scatter(history, f(np.array(history)), color='red',
label='Gradient Descent Steps')
plt.legend()
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Gradient Descent Optimization for f(x) = x^2 * cos(x)')
plt.show()

print("Optymalne theta:", optimal_theta)
```



```
Optymalne theta: 3.6435970380919547
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Wygenerowanie danych make_moons
X, y = make_moons(n_samples=1000, noise=0.1, random_state=42)

# Standaryzacja danych
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Podział na dane treningowe i testowe
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Budowa modelu
model = Sequential([
```

```python
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')  # Warstwa wyjściowa do
klasyfikacji binarnej
])

# Kompilacja
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Trening
history = model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_split=0.2)

# Ocena modelu
loss, accuracy = model.evaluate(X_test, y_test)
print("Strata na danych testowych:", loss)
print("Dokładność na danych testowych:", accuracy)

# Wizualizacja
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```

```
Epoch 1/50
```

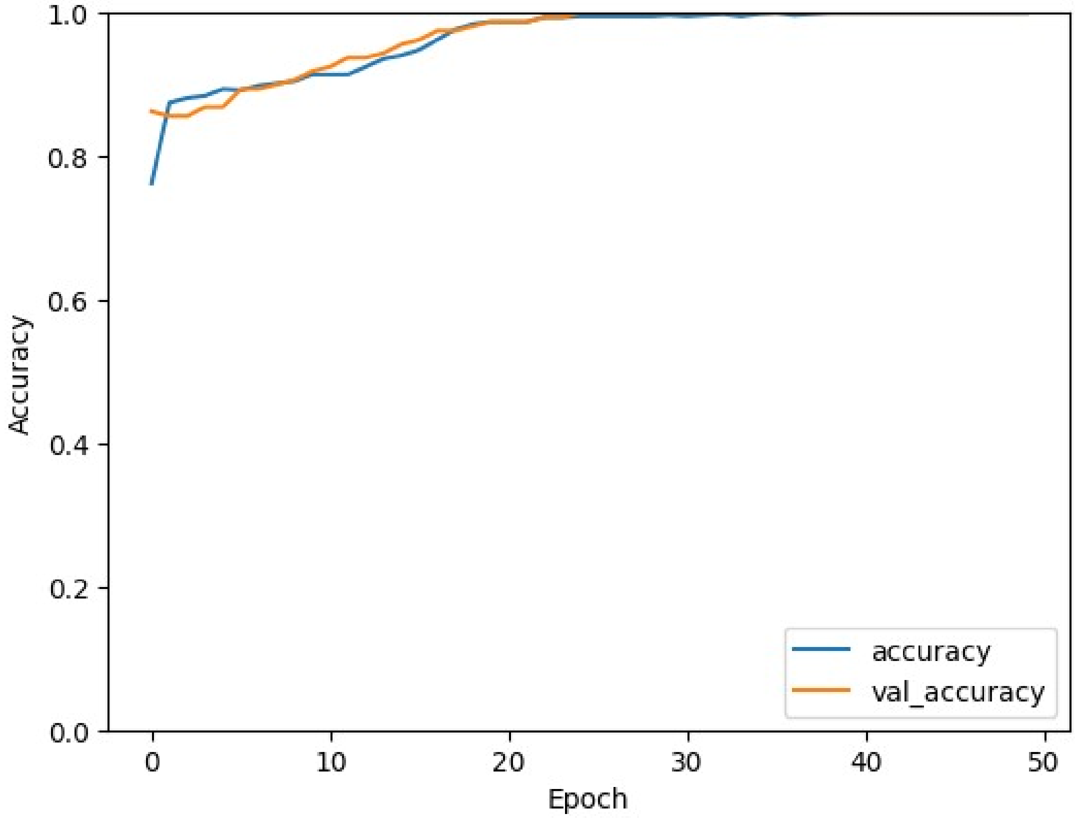
```
c:\Users\szymo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass
an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - accuracy: 0.6322 - loss:
0.6372 - val_accuracy: 0.8625 - val_loss: 0.4851
Epoch 2/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8549 - loss:
0.4458 - val_accuracy: 0.8562 - val_loss: 0.3400
Epoch 3/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9015 - loss:
0.2973 - val_accuracy: 0.8562 - val_loss: 0.2732
Epoch 4/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8902 - loss:
0.2603 - val_accuracy: 0.8687 - val_loss: 0.2511
Epoch 5/50
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9045 - loss:
0.2328 - val_accuracy: 0.8687 - val_loss: 0.2404
Epoch 6/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9091 - loss:
0.2215 - val_accuracy: 0.8938 - val_loss: 0.2308
Epoch 7/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8837 - loss:
0.2418 - val_accuracy: 0.8938 - val_loss: 0.2218
Epoch 8/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8859 - loss:
0.2356 - val_accuracy: 0.9000 - val_loss: 0.2112
Epoch 9/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9052 - loss:
0.1916 - val_accuracy: 0.9062 - val_loss: 0.1995
Epoch 10/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9101 - loss:
0.2046 - val_accuracy: 0.9187 - val_loss: 0.1872
Epoch 11/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9204 - loss:
0.1809 - val_accuracy: 0.9250 - val_loss: 0.1734
Epoch 12/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9168 - loss:
0.1699 - val_accuracy: 0.9375 - val_loss: 0.1593
Epoch 13/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9146 - loss:
0.1728 - val_accuracy: 0.9375 - val_loss: 0.1435
Epoch 14/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9367 - loss:
0.1387 - val_accuracy: 0.9438 - val_loss: 0.1295
Epoch 15/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9463 - loss:
0.1210 - val_accuracy: 0.9563 - val_loss: 0.1143
Epoch 16/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9442 - loss:
0.1175 - val_accuracy: 0.9625 - val_loss: 0.0997
Epoch 17/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9476 - loss:
0.1062 - val_accuracy: 0.9750 - val_loss: 0.0841
Epoch 18/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9652 - loss:
0.0955 - val_accuracy: 0.9750 - val_loss: 0.0722
Epoch 19/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9920 - loss:
0.0608 - val_accuracy: 0.9812 - val_loss: 0.0605
Epoch 20/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9861 - loss:
0.0699 - val_accuracy: 0.9875 - val_loss: 0.0506
Epoch 21/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9876 - loss:
```

```
0.0568 - val_accuracy: 0.9875 - val_loss: 0.0419
Epoch 22/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9827 - loss:
0.0502 - val_accuracy: 0.9875 - val_loss: 0.0366
Epoch 23/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9903 - loss:
0.0466 - val_accuracy: 0.9937 - val_loss: 0.0303
Epoch 24/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9974 - loss:
0.0317 - val_accuracy: 0.9937 - val_loss: 0.0269
Epoch 25/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9959 - loss:
0.0273 - val_accuracy: 1.0000 - val_loss: 0.0227
Epoch 26/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9958 - loss:
0.0283 - val_accuracy: 1.0000 - val_loss: 0.0196
Epoch 27/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9985 - loss:
0.0183 - val_accuracy: 1.0000 - val_loss: 0.0181
Epoch 28/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9965 - loss:
0.0204 - val_accuracy: 1.0000 - val_loss: 0.0151
Epoch 29/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9974 - loss:
0.0142 - val_accuracy: 1.0000 - val_loss: 0.0135
Epoch 30/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9928 - loss:
0.0203 - val_accuracy: 1.0000 - val_loss: 0.0121
Epoch 31/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9946 - loss:
0.0178 - val_accuracy: 1.0000 - val_loss: 0.0107
Epoch 32/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9990 - loss:
0.0117 - val_accuracy: 1.0000 - val_loss: 0.0098
Epoch 33/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9980 - loss:
0.0145 - val_accuracy: 1.0000 - val_loss: 0.0088
Epoch 34/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9984 - loss:
0.0112 - val_accuracy: 1.0000 - val_loss: 0.0079
Epoch 35/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9986 - loss:
0.0124 - val_accuracy: 1.0000 - val_loss: 0.0070
Epoch 36/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss:
0.0137 - val_accuracy: 1.0000 - val_loss: 0.0064
Epoch 37/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9973 - loss:
0.0102 - val_accuracy: 1.0000 - val_loss: 0.0061
```

```
Epoch 38/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9996 - loss:
0.0104 - val_accuracy: 1.0000 - val_loss: 0.0056
Epoch 39/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss:
0.0067 - val_accuracy: 1.0000 - val_loss: 0.0050
Epoch 40/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 1.0000 - loss:
0.0086 - val_accuracy: 1.0000 - val_loss: 0.0048
Epoch 41/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 1.0000 - loss:
0.0061 - val_accuracy: 1.0000 - val_loss: 0.0044
Epoch 42/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss:
0.0084 - val_accuracy: 1.0000 - val_loss: 0.0040
Epoch 43/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss:
0.0069 - val_accuracy: 1.0000 - val_loss: 0.0040
Epoch 44/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss:
0.0056 - val_accuracy: 1.0000 - val_loss: 0.0035
Epoch 45/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss:
0.0050 - val_accuracy: 1.0000 - val_loss: 0.0033
Epoch 46/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss:
0.0049 - val_accuracy: 1.0000 - val_loss: 0.0031
Epoch 47/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss:
0.0058 - val_accuracy: 1.0000 - val_loss: 0.0029
Epoch 48/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss:
0.0054 - val_accuracy: 1.0000 - val_loss: 0.0027
Epoch 49/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss:
0.0052 - val_accuracy: 1.0000 - val_loss: 0.0026
Epoch 50/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss:
0.0066 - val_accuracy: 1.0000 - val_loss: 0.0023
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0022

Strata na danych testowych: 0.0028550871647894382
Dokładność na danych testowych: 1.0
```

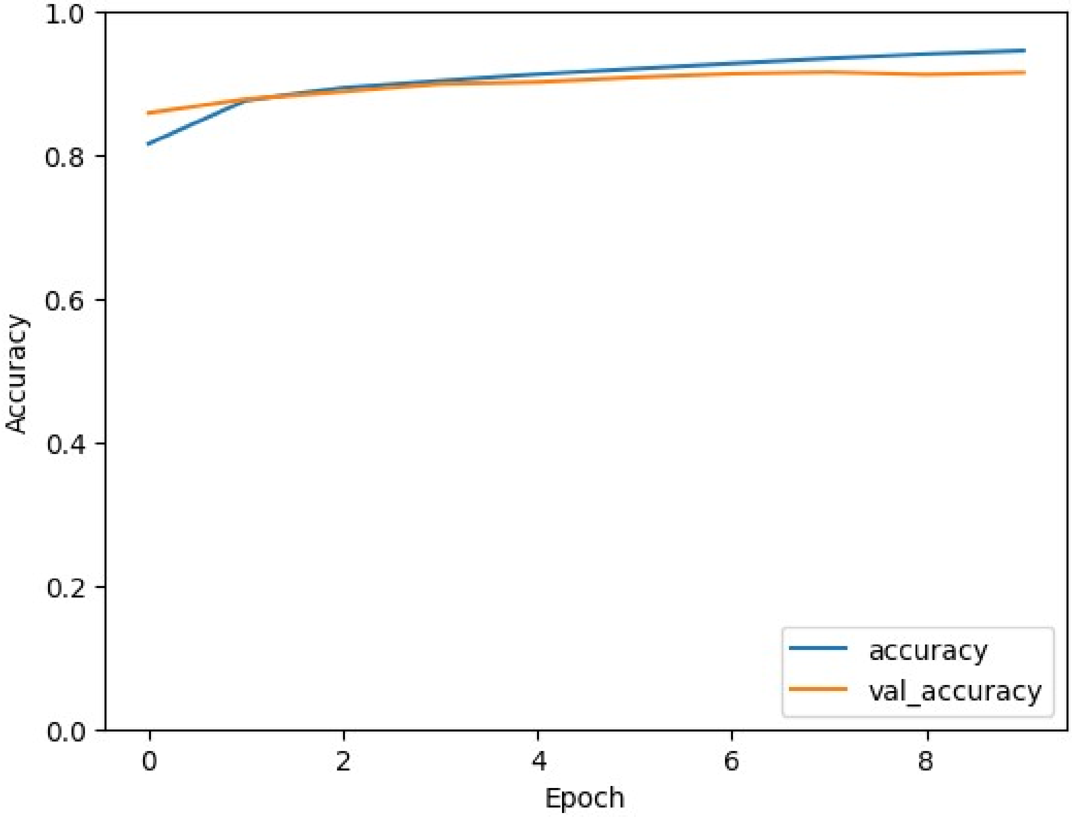


```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense,
MaxPooling2D
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.utils import to_categorical

# Wczytanie danych Fashion MNIST
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

# Normalizacja danych
X_train = X_train / 255.0
X_test = X_test / 255.0

# Dodanie wymiaru dla kanałów (konieczne dla Conv2D)
X_train = X_train[..., np.newaxis]
X_test = X_test[..., np.newaxis]

# One-hot encoding etykiet
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Budowa modelu
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')  # Warstwa wyjściowa do
klasyfikacji 10 klas
])

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

# Trening
history = model.fit(X_train, y_train, epochs=10, batch_size=64,
validation_split=0.2)

# Ocena modelu
loss, accuracy = model.evaluate(X_test, y_test)
print("Strata na danych testowych:", loss)
print("Dokładność na danych testowych:", accuracy)

# Wizualizacja
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 3us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

```
c:\Users\szymo\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: 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)

Epoch 1/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 16s 19ms/step - accuracy: 0.7457 - loss:
0.7229 - val_accuracy: 0.8590 - val_loss: 0.3910
Epoch 2/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 14ms/step - accuracy: 0.8696 - loss:
0.3544 - val_accuracy: 0.8783 - val_loss: 0.3341
Epoch 3/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 15ms/step - accuracy: 0.8909 - loss:
0.3001 - val_accuracy: 0.8887 - val_loss: 0.3034
Epoch 4/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 12s 16ms/step - accuracy: 0.9055 - loss:
0.2582 - val_accuracy: 0.8995 - val_loss: 0.2785
Epoch 5/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 11s 15ms/step - accuracy: 0.9129 - loss:
0.2384 - val_accuracy: 0.9022 - val_loss: 0.2740
Epoch 6/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - accuracy: 0.9196 - loss:
0.2176 - val_accuracy: 0.9085 - val_loss: 0.2540
Epoch 7/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - accuracy: 0.9284 - loss:
0.1940 - val_accuracy: 0.9137 - val_loss: 0.2464
Epoch 8/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 12s 16ms/step - accuracy: 0.9343 - loss:
0.1754 - val_accuracy: 0.9159 - val_loss: 0.2417
Epoch 9/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 13s 17ms/step - accuracy: 0.9417 - loss:
0.1599 - val_accuracy: 0.9127 - val_loss: 0.2442
Epoch 10/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 12s 16ms/step - accuracy: 0.9474 - loss:
0.1405 - val_accuracy: 0.9153 - val_loss: 0.2464
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9114 - loss:
0.2615
Strata na danych testowych: 0.2572506368160248
Dokładność na danych testowych: 0.9115999937057495
```


1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
0
2
4
6
8
Epoch
accuracy
val_accuracy