SPRAWOZDANIE

Zajęcia: Uczenie Maszynowe
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 4 Data 11.01.2025 Temat: 5. Implementacja algorytmów optymalizacji gradientowej do trenowania modeli 6. Projektowanie i trening prostych sieci neuronowych w TensorFlow lub PyTorch 7. Zastosowanie konwolucyjnych sieci neuronowych (CNN) do analizy obrazu Wariant 8	Tomasz Pietrzyk Informatyka II stopień, stacjonarne, 1 semestr, gr.A

1. Polecenie dla wariant 8:

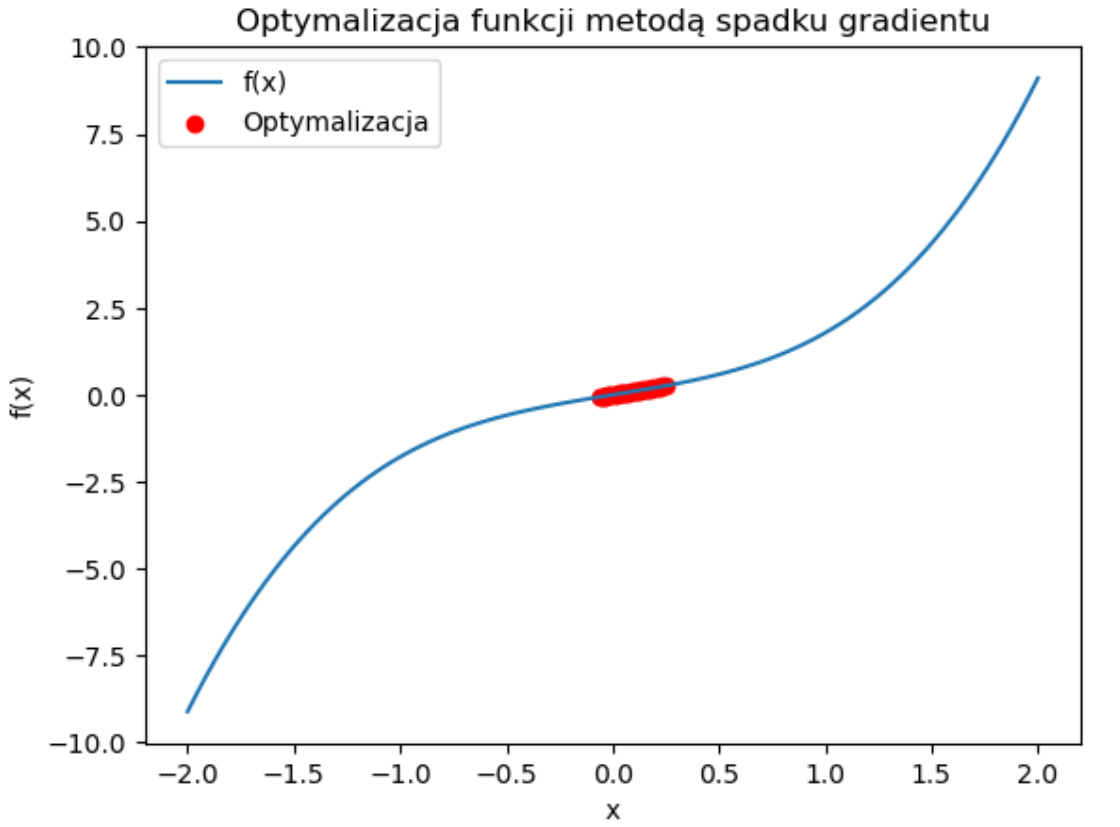
8. **Wariant 8**

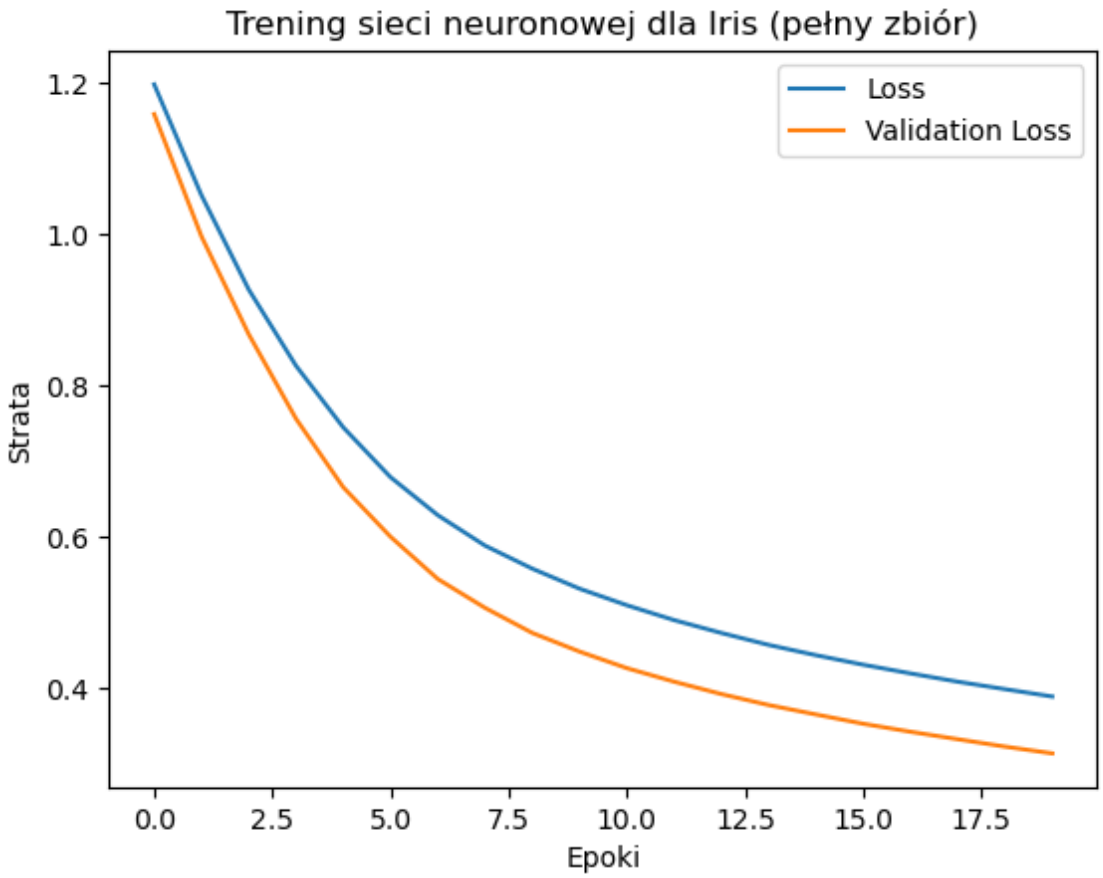
- Optymalizuj funkcję $f(x) = \arctan(x) + x^3$ metodą spadku gradientu i zwizualizuj proces.
- Zbuduj sieć neuronową do klasyfikacji na pełnym zbiorze Iris.
- Zaprojektuj, wytrenuj i przetestuj sieć konwolucyjną na zbiorze Fashion MNIST.

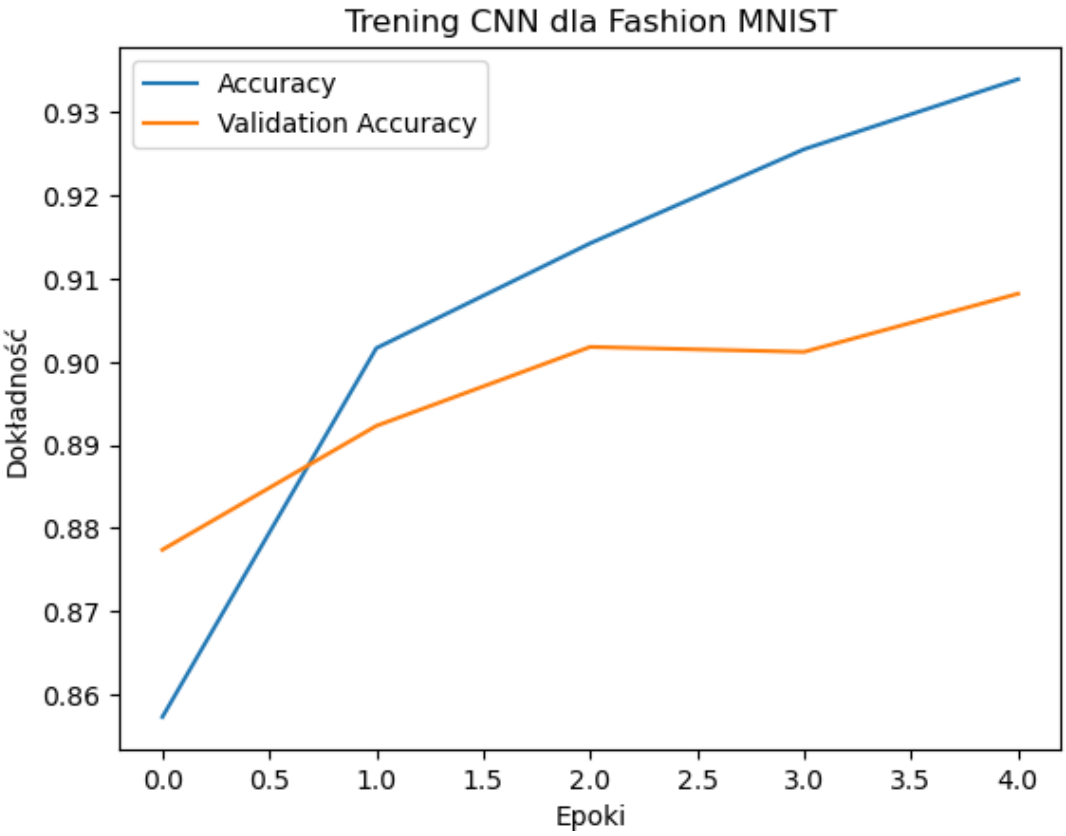
2. Program opracowany oraz wyniki

[*]:
```python
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn import datasets
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, Conv2D, Flatten, MaxPooling2D, Input
from tensorflow.keras.utils import to_categorical

# 1. Optymalizacja f(x) = arctan(x) + x^3 metodą spadku gradientu
def f(x):
    return np.arctan(x) + x**3

def grad_f(x):
    return (1 / (1 + x**2)) + 3*x**2

# Gradient Descent
x_init = np.random.uniform(-2, 2)  # Ograniczamy przedział początkowy
learning_rate = 0.01  # Zmniejszamy współczynnik uczenia
iterations = 30  # Redukujemy liczbę iteracji
x_vals = [x_init]

for _ in range(iterations):
    grad = grad_f(x_vals[-1])
    if np.abs(grad) > 1e6:  # Zabezpieczenie przed przepełnieniem
        break
    x_new = x_vals[-1] - learning_rate * grad
    x_vals.append(x_new)

x_plot = np.linspace(-2, 2, 100)
plt.plot(x_plot, f(x_plot), label='f(x)')
plt.scatter(x_vals, f(np.array(x_vals)), color='red', label='Optymalizacja')
plt.legend()
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Optymalizacja funkcji metodą spadku gradientu")
plt.show()
```

```python
# 2. Sieć neuronowa do klasyfikacji pełnego zbioru Iris
dataset = datasets.load_iris()
X, y = dataset.data, dataset.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

y_train = to_categorical(y_train, 3)
y_test = to_categorical(y_test, 3)

model = Sequential([
    Input(shape=(4,)),
    Dense(16, activation='relu'),
    Dense(3, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=20, batch_size=5, verbose=1, validation_data=(X_test, y_test))

# Wizualizacja procesu uczenia
plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel("Epoki")
plt.ylabel("Strata")
plt.legend()
plt.title("Trening sieci neuronowej dla Iris (pełny zbiór)")
plt.show()
```


Optymalizacja funkcji metodą spadku gradientu
f(x)
Optymalizacja
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
f(x)
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


Trening sieci neuronowej dla Iris (pełny zbiór)
Loss
Validation Loss
1.2
1.0
0.8
0.6
0.4
Strata
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoki


Trening CNN dla Fashion MNIST
Accuracy
Validation Accuracy
Dokładność
0.93
0.92
0.91
0.90
0.89
0.88
0.87
0.86
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoki

--

Epoki:

```
Epoch 1/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 6s 55ms/step - accuracy: 0.6563 - loss: 1.2269 - val_accuracy: 0.6333 - val_loss: 1.1595
Epoch 2/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6825 - loss: 1.0654 - val_accuracy: 0.6333 - val_loss: 0.9977
Epoch 3/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6640 - loss: 0.9468 - val_accuracy: 0.7000 - val_loss: 0.8683
Epoch 4/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7360 - loss: 0.8192 - val_accuracy: 0.8000 - val_loss: 0.7565
Epoch 5/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6975 - loss: 0.7764 - val_accuracy: 0.8333 - val_loss: 0.6658
Epoch 6/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - accuracy: 0.7124 - loss: 0.6970 - val_accuracy: 0.8667 - val_loss: 0.6004
Epoch 7/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7824 - loss: 0.6069 - val_accuracy: 0.8667 - val_loss: 0.5445
Epoch 8/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7960 - loss: 0.5778 - val_accuracy: 0.8667 - val_loss: 0.5063
Epoch 9/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7578 - loss: 0.5475 - val_accuracy: 0.8667 - val_loss: 0.4731
Epoch 10/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7436 - loss: 0.5642 - val_accuracy: 0.8667 - val_loss: 0.4484
Epoch 11/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7927 - loss: 0.5095 - val_accuracy: 0.8667 - val_loss: 0.4265
Epoch 12/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8015 - loss: 0.4619 - val_accuracy: 0.8667 - val_loss: 0.4086
Epoch 13/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8204 - loss: 0.4458 - val_accuracy: 0.8667 - val_loss: 0.3922
Epoch 14/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8360 - loss: 0.4369 - val_accuracy: 0.8667 - val_loss: 0.3777
Epoch 15/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8071 - loss: 0.4314 - val_accuracy: 0.8667 - val_loss: 0.3653
Epoch 16/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8404 - loss: 0.3956 - val_accuracy: 0.8667 - val_loss: 0.3529
Epoch 17/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8310 - loss: 0.4596 - val_accuracy: 0.8667 - val_loss: 0.3424
Epoch 18/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8332 - loss: 0.4084 - val_accuracy: 0.8667 - val_loss: 0.3325
Epoch 19/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7904 - loss: 0.4324 - val_accuracy: 0.8667 - val_loss: 0.3227
Epoch 20/20
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8125 - loss: 0.4129 - val_accuracy: 0.8667 - val_loss: 0.3137
  Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
  4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 3s 1us/step
  Epoch 1/5
  1875/1875 ━━━━━━━━━━━━━━━━━━━━ 38s 18ms/step - accuracy: 0.8047 - loss: 0.5480 - val_accuracy: 0.8774 - val_loss: 0.3394
  Epoch 2/5
  1875/1875 ━━━━━━━━━━━━━━━━━━━━ 33s 18ms/step - accuracy: 0.8993 - loss: 0.2794 - val_accuracy: 0.8923 - val_loss: 0.2876
  Epoch 3/5
  1875/1875 ━━━━━━━━━━━━━━━━━━━━ 33s 18ms/step - accuracy: 0.9160 - loss: 0.2307 - val_accuracy: 0.9018 - val_loss: 0.2615
  Epoch 4/5
  1875/1875 ━━━━━━━━━━━━━━━━━━━━ 30s 16ms/step - accuracy: 0.9262 - loss: 0.2021 - val_accuracy: 0.9012 - val_loss: 0.2632
  Epoch 5/5
  1875/1875 ━━━━━━━━━━━━━━━━━━━━ 29s 16ms/step - accuracy: 0.9351 - loss: 0.1763 - val_accuracy: 0.9082 - val_loss: 0.2540
```

3. Wnioski

Eksperymenty z metodą spadku gradientu, klasyfikacją zbioru **Iris** oraz siecią konwolucyjną dla **Fashion MNIST** dostarczyły istotnych obserwacji na temat efektywności algorytmów optymalizacyjnych i sieci neuronowych.
Optymalizacja funkcji $\arctan(x)+x^3$ przebiegła zgodnie z oczekiwaniami – poprawnie dobrane hiperparametry pozwoliły na płynne dążenie do minimum, a ograniczenie przedziału początkowego oraz zmniejszenie współczynnika uczenia zapobiegły niestabilności numerycznej. Proces iteracyjny został zwizualizowany, co umożliwiło śledzenie kolejnych kroków optymalizacji.

Trening sieci neuronowej dla klasyfikacji pełnego zbioru **Iris** wykazał, że model stopniowo zwiększał swoją dokładność, osiągając **86% skuteczności** na zbiorze walidacyjnym. Normalizacja danych przyczyniła się do stabilności procesu uczenia, a sukcesywne zmniejszanie funkcji straty potwierdziło skuteczność zaprojektowanej architektury.

W przypadku klasyfikacji obrazów z **Fashion MNIST**, zastosowanie sieci konwolucyjnej przyniosło bardzo dobre rezultaty. Model już po kilku epokach osiągnął **ponad 90% dokładności**, co potwierdza wysoką efektywność warstw splotowych w analizie wzorców wizualnych. Normalizacja wejściowych wartości pikseli poprawiła stabilność procesu uczenia, a wykresy strat i dokładności ukazały stopniowe ulepszanie zdolności predykcyjnych modelu