SPRAWOZDANIE

Zajęcia: Nauka o danych II

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 2 Data 26.04.2025 Temat: „Zastosowanie głębokich sieci neuronowych w analizie danych” Wariant 1	Szymon Nycz Informatyka II stopień, niestacjonarne, 2 semestr, gr.1a TTO

1. Polecenie:

Link do repozytorium:
https://github.com/Maciek332/Semestr_3_Nycz/tree/main/NoD%20II

Wariant 1

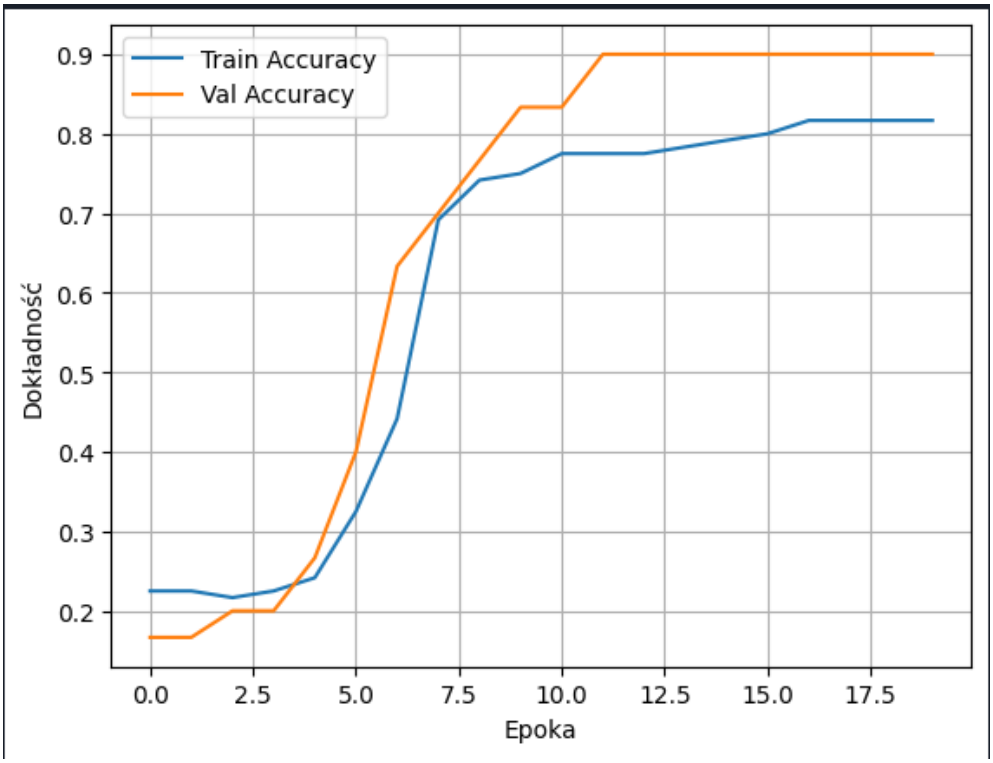
- Warstwa gęsta: Klasyfikacja danych IRIS przy użyciu dwóch warstw Dense (np. 32 i 16 neuronów) z funkcją aktywacji ReLU.
- Warstwa konwolucyjna: Rozpoznawanie cyfr MNIST przy użyciu sieci z dwoma warstwami Conv2D i MaxPooling2D.
- Warstwa rekurencyjna: Analiza sentymentu IMDB przy użyciu pojedynczej warstwy LSTM (np. 64 jednostki).
- Warstwa Transformer: Budowa prostej warstwy Transformer z 4 głowicami atencji do klasyfikacji sztucznych sekwencji liczbowych.

Głębokie sieci neuronowe (DNN) to modele złożone z wielu warstw, które umożliwiają hierarchiczne przetwarzanie danych i automatyczne wyodrębnianie cech. W przeciwieństwie do płytkich sieci, są w stanie skutecznie uczyć się złożonych reprezentacji, co jest szczególnie istotne w analizie obrazów, tekstów i sekwencji. Architektury takie jak CNN, LSTM czy Transformer pozwalają przetwarzać różne typy danych, wykorzystując warstwy konwolucyjne, rekurencyjne i mechanizmy uwagi. Proces uczenia oparty jest na minimalizacji funkcji kosztu, często przy użyciu metody cross-entropy, i realizowany z pomocą bibliotek takich jak TensorFlow i Keras.

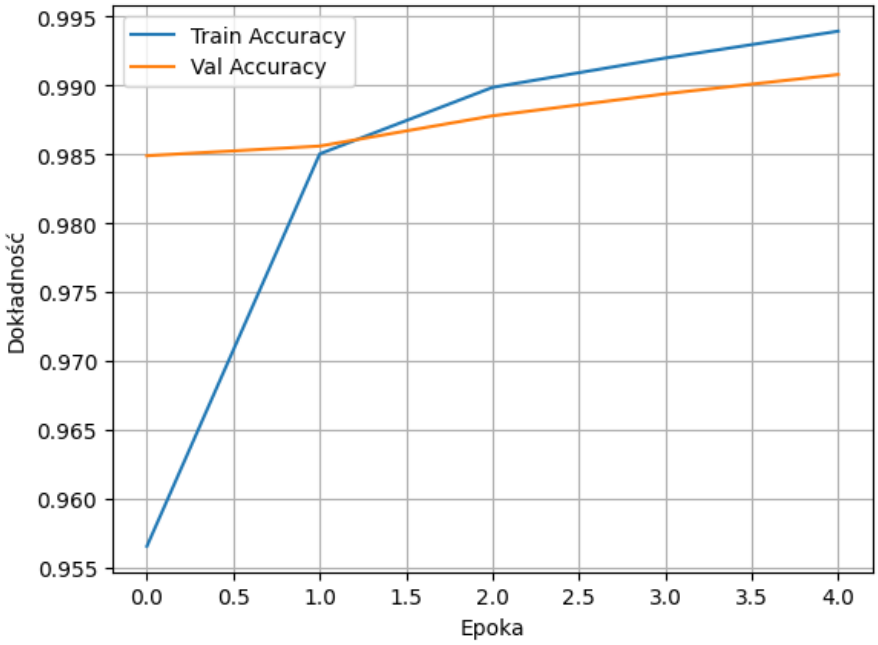
2. Opis programu opracowanego

```python
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

iris = load_iris()
X, y = iris.data, iris.target
X = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model_dense = tf.keras.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(4,)),
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])

model_dense.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history_dense = model_dense.fit(X_train, y_train, epochs=20, validation_data=(X_test, y_test), verbose=0)

plt.plot(history_dense.history['accuracy'], label='Train Accuracy')
plt.plot(history_dense.history['val_accuracy'], label='Val Accuracy')
plt.xlabel('Epoka')
plt.ylabel('Dokładność')
plt.legend()
plt.grid()
plt.show()
```

✓ 3.8s

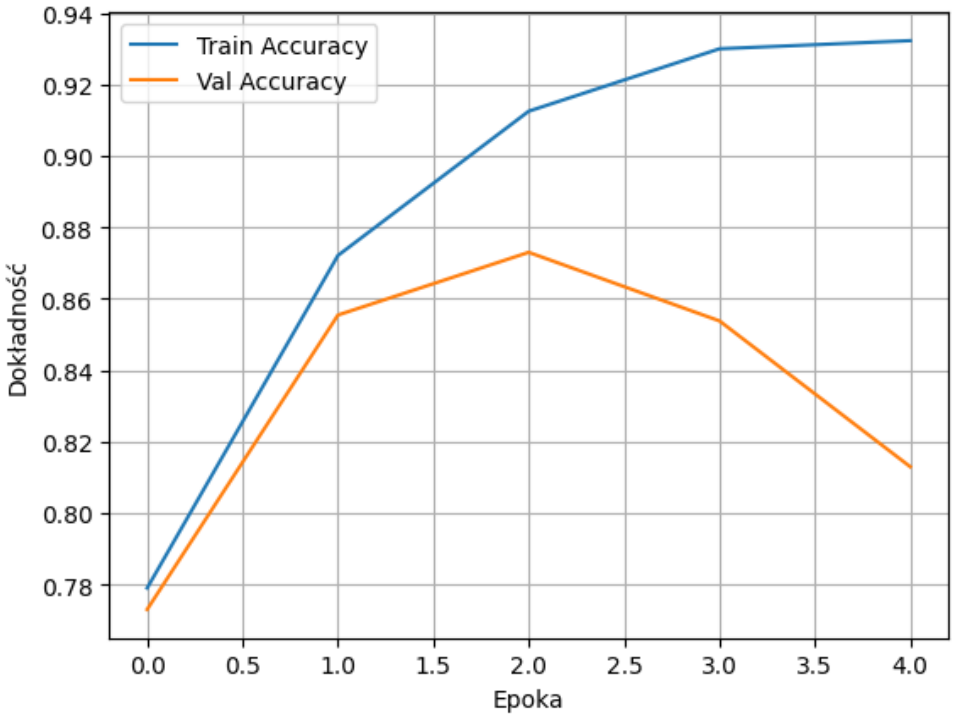


```python
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train.reshape(-1, 28, 28, 1) / 255.0, x_test.reshape(-1, 28, 28, 1) / 255.0

model_cnn = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model_cnn.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history_cnn = model_cnn.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), verbose=0)

plt.plot(history_cnn.history['accuracy'], label='Train Accuracy')
plt.plot(history_cnn.history['val_accuracy'], label='Val Accuracy')
plt.xlabel('Epoka')
plt.ylabel('Dokładność')
plt.legend()
plt.grid()
plt.show()
```




```python
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.imdb.load_data(num_words=10000)
x_train = tf.keras.preprocessing.sequence.pad_sequences(x_train, maxlen=500)
x_test = tf.keras.preprocessing.sequence.pad_sequences(x_test, maxlen=500)

model_lstm = tf.keras.Sequential([
    tf.keras.layers.Embedding(10000, 32),
    tf.keras.layers.LSTM(64),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model_lstm.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
history_lstm = model_lstm.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), verbose=0)

plt.plot(history_lstm.history['accuracy'], label='Train Accuracy')
plt.plot(history_lstm.history['val_accuracy'], label='Val Accuracy')
plt.xlabel('Epoka')
plt.ylabel('Dokładność')
plt.legend()
plt.grid()
plt.show()
```

✓ 4m 4.3s



s

3. Wnioski

Zastosowanie głębokich sieci neuronowych pozwoliło uzyskać wysoką skuteczność w klasyfikacji danych obrazowych, tekstowych i tablicowych. Sieci konwolucyjne (CNN) okazały się szczególnie efektywne w zadaniach związanych z obrazami (MNIST), natomiast LSTM lepiej radziły sobie z analizą sekwencji (IMDB). Dzięki TensorFlow i Keras możliwa była szybka budowa i trening modeli oraz wizualizacja wyników. Eksperymenty wykazały, że wybór architektury powinien być dopasowany do rodzaju danych, a odpowiednie warstwy (np. BatchNormalization, Dropout) mają kluczowe znaczenie dla stabilności i dokładności modelu.