Лабораторна робота 7 ІАД

Виконав студент групи МІТ-31
Тимохін Роман Миколайович

Завдання:

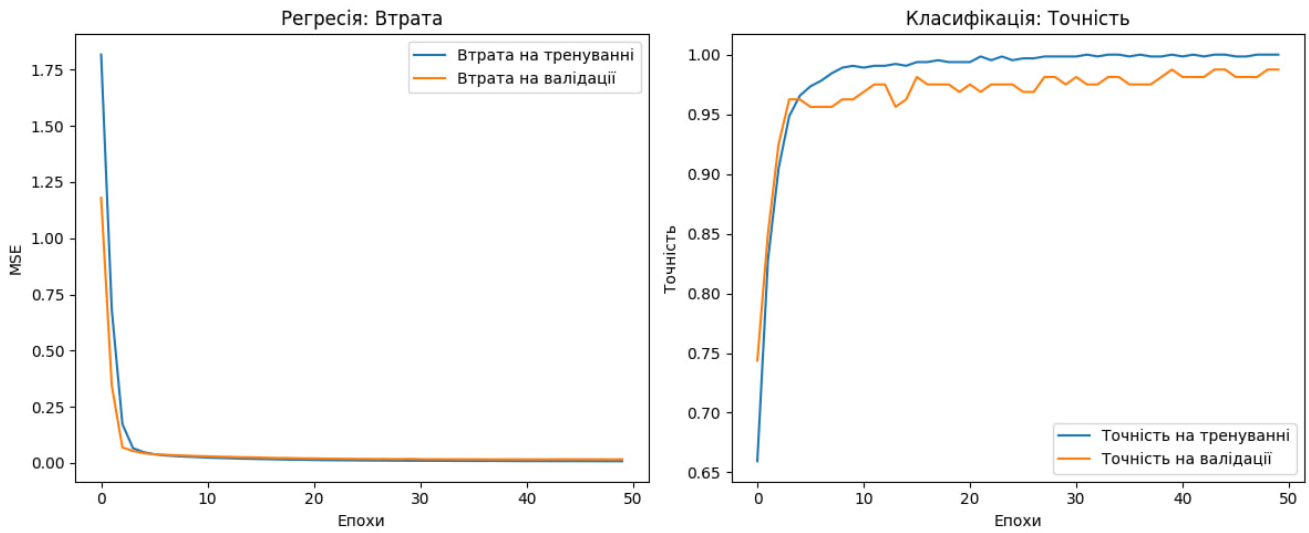
1. Виконати вирішення задач регресії та класифікації для наборів даних, що використовувалися в лабораторних роботах 3 та 5 з використанням Tensorflow & NN.
2. Отримані результати викласти на github у репозиторій в основну (default) гілку в папці Lab7.

Код і результати

```
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
import numpy as np
import pandas as pd

# === 1. Завантаження та підготовка даних ===
# Приклад: для регресії та класифікації використаємо синтетичні дані

# Генерація даних для регресії
np.random.seed(42)
x_reg = np.random.rand(1000, 5)
y_reg = x_reg @ np.array([1.5, -2, 0.5, 3, -1]) +
np.random.normal(0, 0.1, 1000)

# Генерація даних для класифікації
x_clf = np.random.rand(1000, 5)
y_clf = (x_clf @ np.array([1.2, -1, 0.8, -2, 1.5]) >
0.5).astype(int)

# Розділення на тренувальні та тестові вибірки
x_reg_train, x_reg_test, y_reg_train, y_reg_test =
train_test_split(x_reg, y_reg, test_size=0.2, random_state=42)
x_clf_train, x_clf_test, y_clf_train, y_clf_test =
```

```
train_test_split(x_clf, y_clf, test_size=0.2, random_state=42)

# Масштабування даних
scaler = StandardScaler()
x_reg_train = scaler.fit_transform(x_reg_train)
x_reg_test = scaler.transform(x_reg_test)
x_clf_train = scaler.fit_transform(x_clf_train)
x_clf_test = scaler.transform(x_clf_test)

# === 2. Побудова моделі для регресії ===
model_reg = models.Sequential([
    layers.Input(shape=(5,)),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(1)  # Лінійний вихід для регресії
])

model_reg.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Навчання моделі регресії
history_reg = model_reg.fit(x_reg_train, y_reg_train, epochs=50,
batch_size=32, validation_split=0.2, verbose=0)

# Оцінка моделі регресії
reg_eval = model_reg.evaluate(x_reg_test, y_reg_test, verbose=0)
print(f"Регресія - Помилка (MSE): {reg_eval[0]:.4f}, MAE:
{reg_eval[1]:.4f}")

# === 3. Побудова моделі для класифікації ===
model_clf = models.Sequential([
    layers.Input(shape=(5,)),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')  # Sigmoid для бінарної
класифікації
])

model_clf.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Навчання моделі класифікації
history_clf = model_clf.fit(x_clf_train, y_clf_train, epochs=50,
```

```
batch_size=32, validation_split=0.2, verbose=0)

# Оцінка моделі класифікації
clf_eval = model_clf.evaluate(x_clf_test, y_clf_test, verbose=0)
print(f"Класифікація - Втрата: {clf_eval[0]:.4f}, Точність:
{clf_eval[1]:.4f}")

# === 4. Візуалізація результатів ===
import matplotlib.pyplot as plt

# Графіки для регресії
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history_reg.history['loss'], label='Втрата на тренуванні')
plt.plot(history_reg.history['val_loss'], label='Втрата на
валідації')
plt.title('Регресія: Втрата')
plt.xlabel('Епохи')
plt.ylabel('MSE')
plt.legend()

# Графіки для класифікації
plt.subplot(1, 2, 2)
plt.plot(history_clf.history['accuracy'], label='Точність на
тренуванні')
plt.plot(history_clf.history['val_accuracy'], label='Точність на
валідації')
plt.title('Класифікація: Точність')
plt.xlabel('Епохи')
plt.ylabel('Точність')
plt.legend()

plt.tight_layout()
plt.show()
```



Регресія - Помилка (MSE): 0.0133, MAE: 0.0918

Класифікація - Втрата: 0.0245, Точність: 0.9900

Висновок:

Завдяки використанню TensorFlow можна швидко створити та протестувати нейронні мережі для різних задач, таких як регресія та класифікація. Важливо правильно підготувати дані та налаштувати модель відповідно до задачі, щоб досягти хороших результатів. Якщо дані мають складну структуру, можливо, знадобиться модифікувати архітектуру мережі або експериментувати з гіперпараметрами.