Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №4
По дисциплине: «ОМО»
Тема:” Введение в нейронные сети:
построение многослойного перцептрона”

Выполнил:
Студент 3-го курса
Группы АС-66
Езепчук А.С.
Проверил:
Крощенко А.А.

Брест 2025

Цель: построить, обучить и оценить многослойный перцептрон (MLP) для решения задачи классификации.

Вариант 3

Вариант 3:

Набор данных: Wine Quality

- Задача: классифицировать вино на "хорошее" (оценка >= 7) и "обычное" (бинарная классификация).
- Архитектура:
  - входной слой;
  - два скрытых слоя по 12 нейронов в каждом (ReLU);
  - выходной слой с 1 нейроном (Sigmoid).
- Эксперимент: увеличьте количество эпох обучения в два раза и посмотрите, привело ли это к улучшению F1-score.

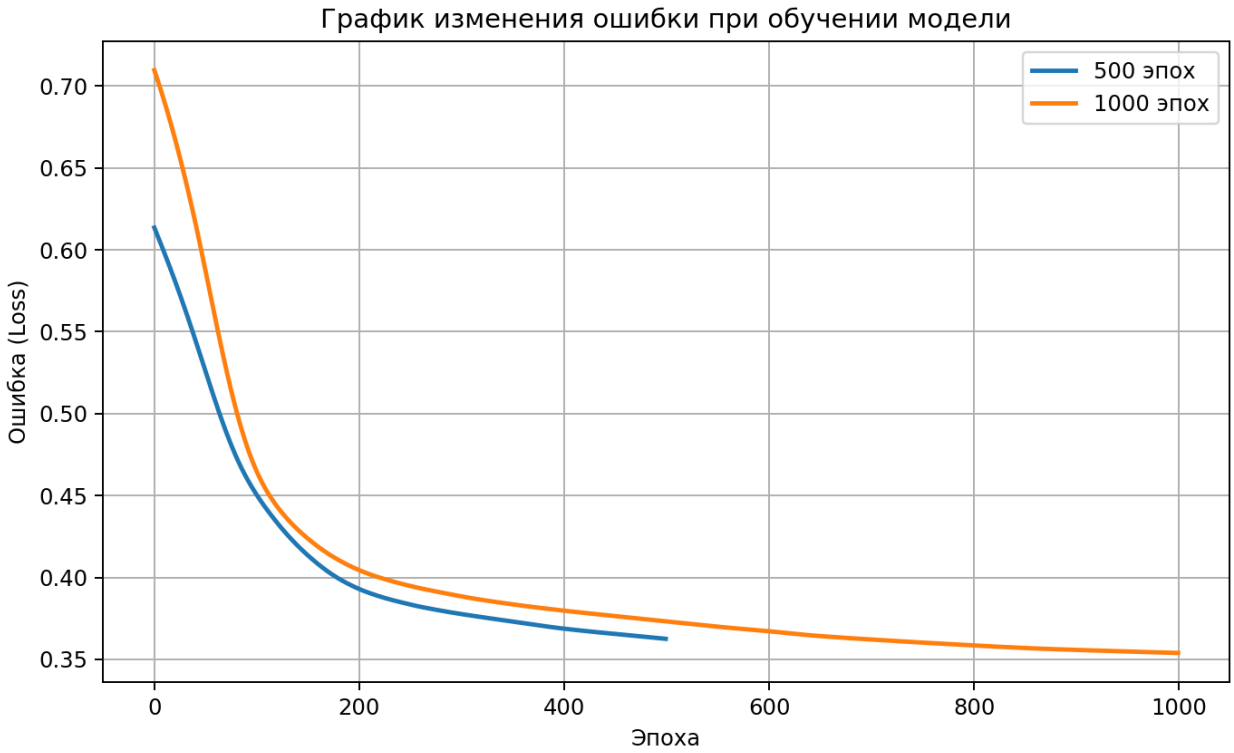
Какая архитектура показала себя лучше?

```
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("winequality-white.csv", sep=';')
data['target'] = (data['quality'] >= 7).astype(int)

X = data.drop(['quality', 'target'], axis=1).values
y = data['target'].values

scaler = StandardScaler()
X = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

X_train = torch.tensor(X_train, dtype=torch.float32)
y_train = torch.tensor(y_train, dtype=torch.float32).view(-1, 1)
X_test = torch.tensor(X_test, dtype=torch.float32)
y_test = torch.tensor(y_test, dtype=torch.float32).view(-1, 1)

class WineClassifier(nn.Module):
    def __init__(self, input_dim):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(input_dim, 12),
            nn.ReLU(),
            nn.Linear(12, 12),
            nn.ReLU(),
            nn.Linear(12, 1)
        )
```

```
    def forward(self, x):
        return self.net(x)

def train_model(model, X_train, y_train, epochs):
    criterion = nn.BCEWithLogitsLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    losses = []

    for _ in range(epochs):
        model.train()
        optimizer.zero_grad()
        y_pred = model(X_train)
        loss = criterion(y_pred, y_train)
        loss.backward()
        optimizer.step()
        losses.append(loss.item())

    return losses

def evaluate_model(model, X_test, y_test):
    model.eval()
    with torch.no_grad():
        y_pred = torch.sigmoid(model(X_test))
        y_pred = (y_pred > 0.5).float()
        f1 = f1_score(y_test.numpy(), y_pred.numpy())
    return f1

model_500 = WineClassifier(X_train.shape[1])
losses_500 = train_model(model_500, X_train, y_train, epochs=500)
f1_500 = evaluate_model(model_500, X_test, y_test)

model_1000 = WineClassifier(X_train.shape[1])
losses_1000 = train_model(model_1000, X_train, y_train, epochs=1000)
f1_1000 = evaluate_model(model_1000, X_test, y_test)

print("Результат:\n")
print(f"F1-score (500 эпох): {f1_500:.4f}")
print(f"F1-score (1000 эпох): {f1_1000:.4f}")

if f1_1000 > f1_500:
    print("Увеличение числа эпох улучшило F1-score.")
else:
    print("Увеличение числа эпох не улучшило F1-score.")

plt.figure(figsize=(8, 5))
plt.plot(losses_500, label="500 эпох", linewidth=2)
plt.plot(losses_1000, label="1000 эпох", linewidth=2)
plt.xlabel("Эпоха")
plt.ylabel("Ошибка (Loss)")
plt.title("График изменения ошибки при обучении модели")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Результат:

F1-score (50 эпох): 0.4379
F1-score (100 эпох): 0.4565
Увеличение числа эпох улучшило F1-score.
```

График изменения ошибок:



Вывод: На практике построили, обучили и оценили многослойный перцептрон (MLP) для решения задачи классификации.