Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: «ИАД»

Тема: «**Предобучение нейронных сетей с использованием автоэнкодерного подхода**»

Выполнил:

Студент 4 курса

Группы ИИ-23

Ежевский Е.Р.

Проверила:

Андренко К.В.

Брест 2025

Цель: научиться осуществлять предобучение нейронных сетей с помощью автоэнкодерного подхода

Общее задание

1. Взять за основу любую сверточную или полносвязную архитектуру с количеством слоев более 3. Осуществить ее обучение (без предобучения) в соответствии с вариантом задания. Получить оценку эффективности модели, используя метрики, специфичные для решаемой задачи (например, MAPE – для регрессионной задачи или F1/Confusion matrix для классификационной).
2. Выполнить обучение с предобучением, используя автоэнкодерный подход, алгоритм которого изложен в лекции. Условие останова (например, по количеству эпох) при обучении отдельных слоев с использованием автоэнкодера выбрать самостоятельно.
3. Сравнить результаты, полученные при обучении с/без предобучения, сделать выводы.
4. Оформить отчет по выполненной работе, загрузить исходный код и отчет в соответствующий репозиторий на github.

Задание по вариантам

№	Выборка	Тип задачи	Целевая переменная
1	https://archive.ics.uci.edu/dataset/27/credit+approval	классификация	+/-

Код программы:

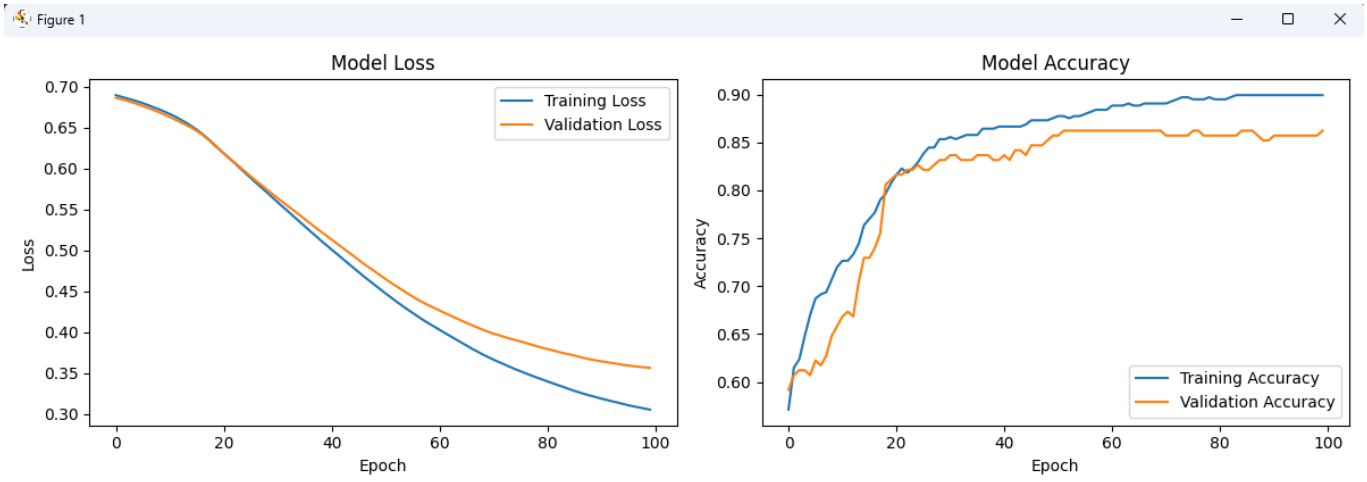
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import matplotlib.pyplot as plt
import numpy as np

columns = [
   'A1','A2','A3','A4','A5','A6','A7','A8','A9','A10','A11',
   'A12','A13','A14','A15','A16'
]

data = pd.read_csv('crx.data', names=columns, na_values='?')

data = data.dropna()

for col in data.columns:
   if data[col].dtype == 'object' and col != 'A16':
      data[col] = LabelEncoder().fit_transform(data[col])

label_encoder = LabelEncoder()
data['A16'] = label_encoder.fit_transform(data['A16'])

X = data.drop('A16', axis=1)
y = data['A16']
```

```
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.3, random_state=42
)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam

model = Sequential([
   Dense(64, input_shape=(X_train.shape[1],), activation='relu'),
   Dense(32, activation='relu'),
   Dense(16, activation='relu'),
   Dense(8, activation='relu'),
   Dense(len(np.unique(y)), activation='softmax')
])

model.compile(optimizer=Adam(learning_rate=0.0001),
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])

history = model.fit(
   X_train, y_train, epochs=100,
   validation_data=(X_test, y_test),
   batch_size=64
)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

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

```
from sklearn.metrics import classification_report, confusion_matrix

y_pred = np.argmax(model.predict(X_test), axis=-1)

print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

from keras.layers import Input
from keras.models import Model

def build_autoencoder(input_dim, encoding_dim):
    input_layer = Input(shape=(input_dim,))
    encoded = Dense(encoding_dim, activation='relu')(input_layer)
    decoded = Dense(input_dim)(encoded)
    autoencoder = Model(input_layer, decoded)
    encoder = Model(input_layer, encoded)
    autoencoder.compile(optimizer='adam', loss='mse')
    return autoencoder, encoder

autoencoder1, encoder1 = build_autoencoder(X_train.shape[1], 64)
autoencoder1.fit(X_train, X_train, epochs=100, batch_size=64, shuffle=True)
X_train_encoded = encoder1.predict(X_train)

autoencoder2, encoder2 = build_autoencoder(64, 32)
autoencoder2.fit(X_train_encoded, X_train_encoded, epochs=100, batch_size=64, shuffle=True)
X_train_encoded = encoder2.predict(X_train_encoded)

autoencoder3, encoder3 = build_autoencoder(32, 16)
autoencoder3.fit(X_train_encoded, X_train_encoded, epochs=100, batch_size=64, shuffle=True)
X_train_encoded = encoder3.predict(X_train_encoded)

autoencoder4, encoder4 = build_autoencoder(16, 8)
autoencoder4.fit(X_train_encoded, X_train_encoded, epochs=100, batch_size=64, shuffle=True)

model.layers[0].set_weights(autoencoder1.get_weights()[:2])
model.layers[1].set_weights(autoencoder2.get_weights()[:2])
model.layers[2].set_weights(autoencoder3.get_weights()[:2])
model.layers[3].set_weights(autoencoder4.get_weights()[:2])

history = model.fit(
    X_train, y_train, epochs=100,
    validation_data=(X_test, y_test),
    batch_size=64
)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
```

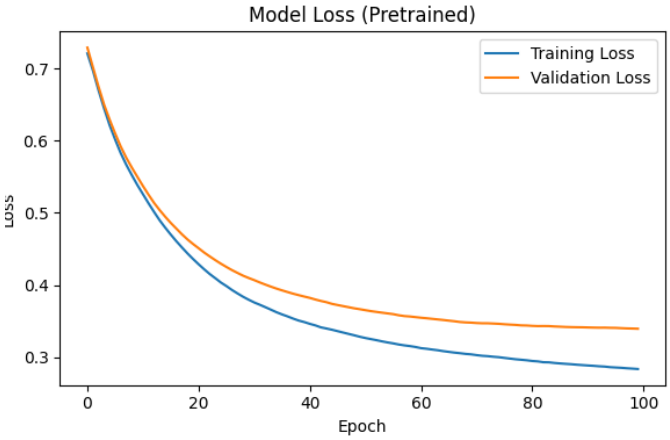
```
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss (Pretrained)')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy (Pretrained)')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()

y_pred_pretrained = np.argmax(model.predict(X_test), axis=-1)

print("Classification Report (Pretrained):\n",
    classification_report(y_test, y_pred_pretrained))
print("Confusion Matrix (Pretrained):\n",
    confusion_matrix(y_test, y_pred_pretrained))
```

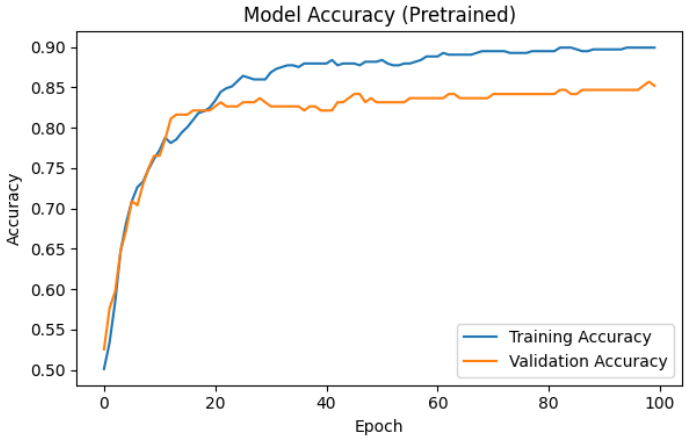
С нуля:




```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.85      0.84        86
           1       0.88      0.87      0.88       110

    accuracy                           0.86       196
   macro avg       0.86      0.86      0.86       196
weighted avg       0.86      0.86      0.86       196

Confusion Matrix:
 [[73 13]
 [14 96]]
```

Предтренированная:

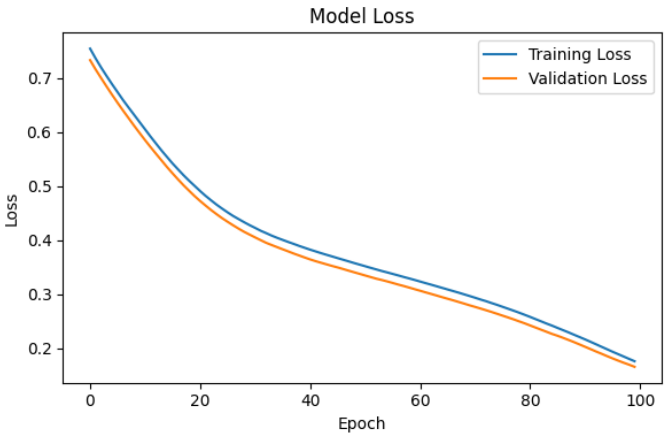


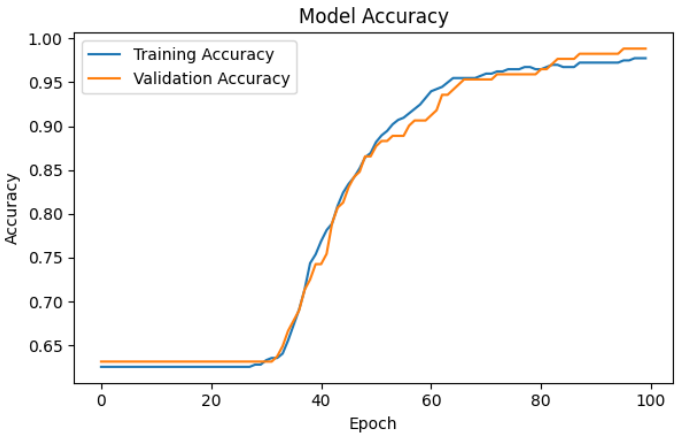



```
Classification Report (Pretrained):
              precision    recall  f1-score   support

           0       0.82      0.85      0.83        86
           1       0.88      0.85      0.87       110

    accuracy                           0.85       196
   macro avg       0.85      0.85      0.85       196
weighted avg       0.85      0.85      0.85       196

Confusion Matrix (Pretrained):
 [[73 13]
 [16 94]]
```

Данные для датасета [Wisconsin Diagnostic Breast Cancer (WDBC)]:

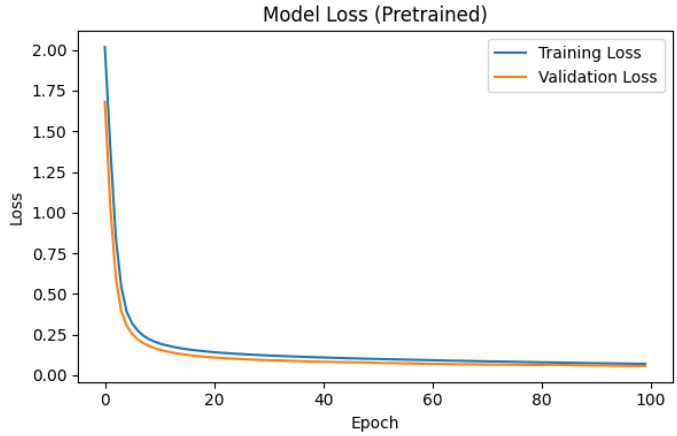
С нуля:

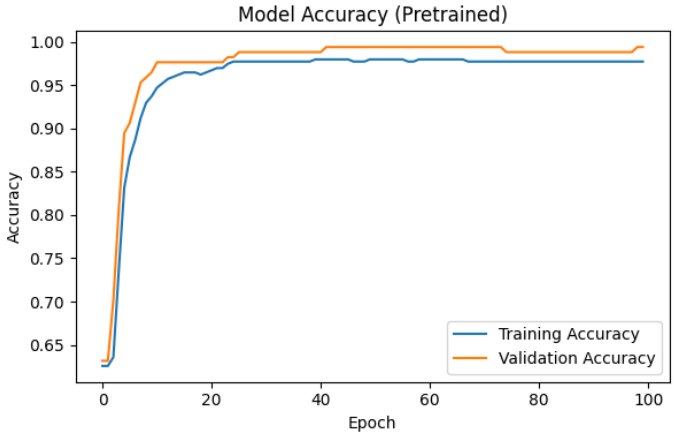




```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       108
           1       1.00      0.97      0.98        63

    accuracy                           0.99       171
   macro avg       0.99      0.98      0.99       171
weighted avg       0.99      0.99      0.99       171

Confusion Matrix:
 [[108   0]
 [  2  61]]
```

Предтренированная:






```
Classification Report (Pretrained):
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       108
           1       1.00      0.98      0.99        63

    accuracy                           0.99       171
   macro avg       1.00      0.99      0.99       171
weighted avg       0.99      0.99      0.99       171

Confusion Matrix (Pretrained):
 [[108   0]
 [  1  62]]
```

Вывод: научился осуществлять предобучение нейронных сетей с помощью автоэнкодерного подхода