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

Скварнюк Д. Н.

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
		регрессия	

Код программы:

```
def build_base_model(input_dim):
    model = keras.Sequential([
        layers.Input(shape=(input_dim,)),
        layers.Dense(128, activation='relu'),
        layers.Dense(64, activation='relu'),
        layers.Dense(32, activation='relu'),
        layers.Dense(16, activation='relu'),
        layers.Dense(1)  # регрессия
    ])
    model.compile(optimizer='adam', loss='mse')
    return model

base_model = build_base_model(X_train_scaled.shape[1])

history_base = base_model.fit(
    X_train_scaled, y_train,
    validation_split=0.2,
    epochs=50,
    batch_size=64,
    verbose=1
)

# Предсказание и оценка
y_pred_base = base_model.predict(X_test_scaled)
mape_base = mean_absolute_percentage_error(y_test, y_pred_base)
```

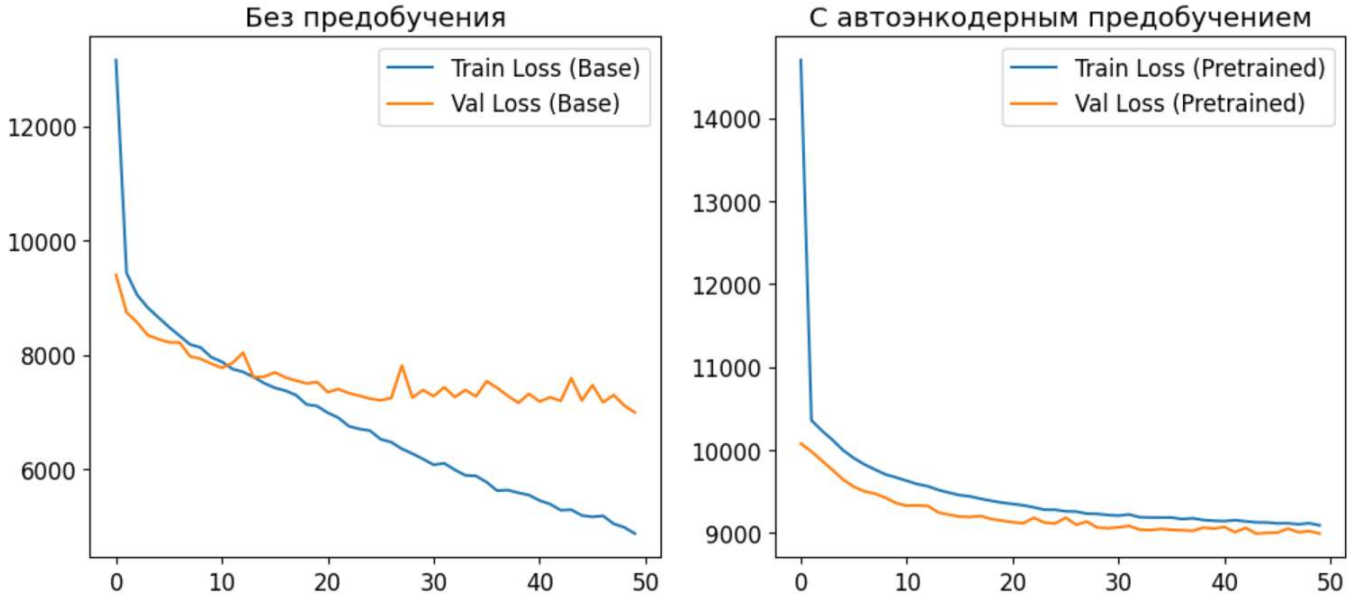
```
print(f"MAPE без предобучения: {mape_base:.4f}")


input_dim = X_train_scaled.shape[1]
encoding_dim = 16  # размер скрытого кода

# Архитектура автоэнкодера
input_layer = layers.Input(shape=(input_dim,))
encoded = layers.Dense(128, activation='relu')(input_layer)
encoded = layers.Dense(64, activation='relu')(encoded)
encoded = layers.Dense(encoding_dim, activation='relu')(encoded)

decoded = layers.Dense(64, activation='relu')(encoded)
decoded = layers.Dense(128, activation='relu')(decoded)
decoded = layers.Dense(input_dim, activation='linear')(decoded)

autoencoder = keras.Model(inputs=input_layer, outputs=decoded)
autoencoder.compile(optimizer='adam', loss='mse')

# Обучение автоэнкодера
history_ae = autoencoder.fit(
    X_train_scaled, X_train_scaled,
    validation_split=0.2,
    epochs=50,
    batch_size=64,
    verbose=1
)

# Извлечение энкодера
encoder = keras.Model(inputs=input_layer, outputs=encoded)


encoded_train = encoder.predict(X_train_scaled)
encoded_test = encoder.predict(X_test_scaled)

def build_pretrained_model(input_dim):
    model = keras.Sequential([
        layers.Input(shape=(input_dim,)),
        layers.Dense(32, activation='relu'),
        layers.Dense(16, activation='relu'),
        layers.Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')
    return model

pretrained_model = build_pretrained_model(encoded_train.shape[1])

history_pre = pretrained_model.fit(
    encoded_train, y_train,
```

```
    validation_split=0.2,
    epochs=50,
    batch_size=64,
    verbose=1
)

y_pred_pre = pretrained_model.predict(encoded_test)
mape_pre = mean_absolute_percentage_error(y_test, y_pred_pre)

# Графики обучения
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history_base.history['loss'], label='Train Loss (Base)')
plt.plot(history_base.history['val_loss'], label='Val Loss (Base)')
plt.title('Без предобучения')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history_pre.history['loss'], label='Train Loss (Pretrained)')
plt.plot(history_pre.history['val_loss'], label='Val Loss (Pretrained)')
plt.title('С автоэнкодерным предобучением')
plt.legend()

plt.show()
```

===== Appliances_Energy_Prediction Dataset =====

Without pretraining — MAPE: 0.4137

With pretraining — MAPE: 0.6642



Вывод: научился осуществлять предобучение нейронных сетей с помощью автоэнкодерного подхода