Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

По дисциплине: «ИАД»

Тема: «Предобучение нейронных сетей с использованием RBM»

Выполнил:

Студент 4 курса

Группы ИИ-23

Скварнюк Д. Н.

Проверила:

Андренко К.В.

Брест 2025

Цель: научиться осуществлять предобучение нейронных сетей с помощью RBM.

Общее задание

1. Взять за основу нейронную сеть из лабораторной работы №3. Выполнить обучение с предобучением, используя стек ограниченных машин Больцмана (RBM – Restricted Boltzmann Machine), алгоритм которого изложен в лекции. Условие останова (например, по количеству эпох) при обучении отдельных слоев как RBM выбрать самостоятельно.

2. Сравнить результаты, полученные при
- обучении без предобучения (ЛР 3);
- обучении с предобучением, используя автоэнкодерный подход (ЛР3);
- обучении с предобучением, используя RBM.

3. Обучить модели на данных из ЛР 2, сравнить результаты по схеме из пункта 2;

4. Сделать выводы, оформить отчет по выполненной работе, загрузить исходный код и отчет в соответствующий репозиторий на github.

Задание по вариантам

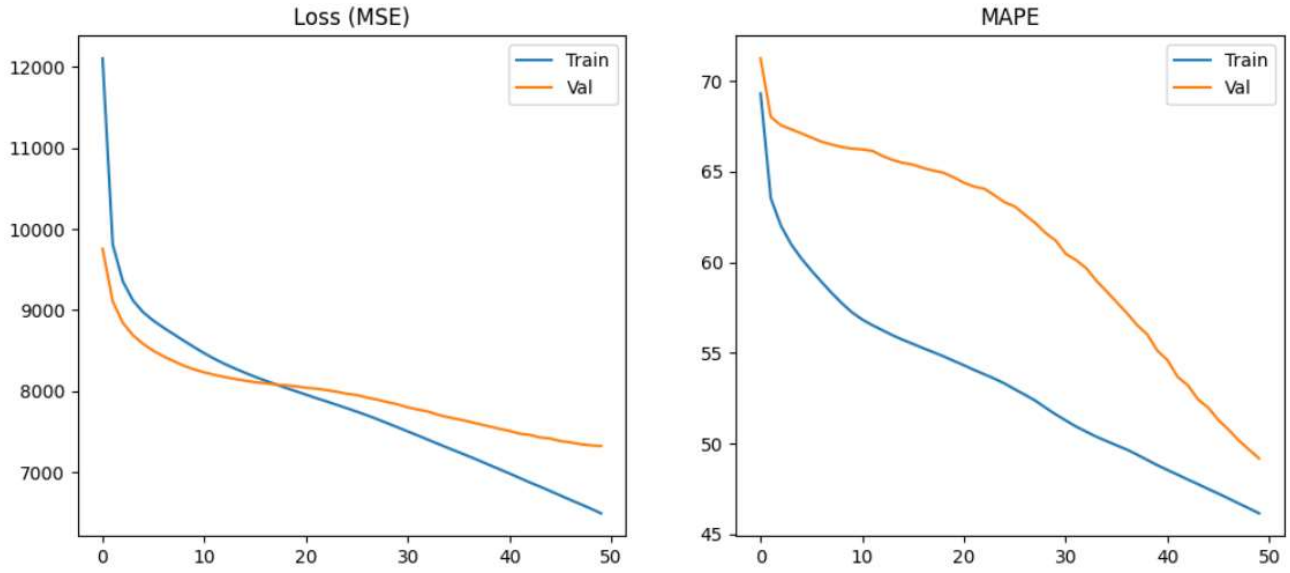
№	Выборка	Тип задачи	Целевая переменная
		регрессия	

Код программы:

```
!pip install ucimlrepo tensorflow scikit-learn pandas numpy matplotlib seaborn -q

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_percentage_error
from tensorflow import keras
from tensorflow.keras import layers
from ucimlrepo import fetch_ucirepo

dataset = fetch_ucirepo(id=374)
X = dataset.data.features
y = dataset.data.targets

y = y["Appliances"]

print("Размер данных:", X.shape, y.shape)
X.head()


X = X.drop(columns=["date"])
```

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

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

print("Размер обучающей выборки:", X_train_scaled.shape)
print("Размер тестовой выборки:", X_test_scaled.shape)

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
from sklearn.metrics import mean_absolute_percentage_error

tf.random.set_seed(42)
np.random.seed(42)

def build_rbm_pretrained_model(input_dim, rbm_stack):
    model = keras.Sequential()
    model.add(layers.Input(shape=(input_dim,)))

    prev_dim = input_dim

    for rbm in rbm_stack:
        W, hb = rbm.W.numpy(), rbm.hb.numpy()

        dense = layers.Dense(rbm.n_hidden, activation="relu")

        dense.build((None, prev_dim))

        dense.set_weights([W, hb])

        model.add(dense)
        prev_dim = rbm.n_hidden

    model.add(layers.Dense(1))

    model.compile(
        optimizer="adam",
        loss="mse",
        metrics=[keras.metrics.MeanAbsolutePercentageError()]
    )
    return model
==
model = build_rbm_pretrained_model(X_train_scaled.shape[1], rbm_stack)

print(model.summary())
```

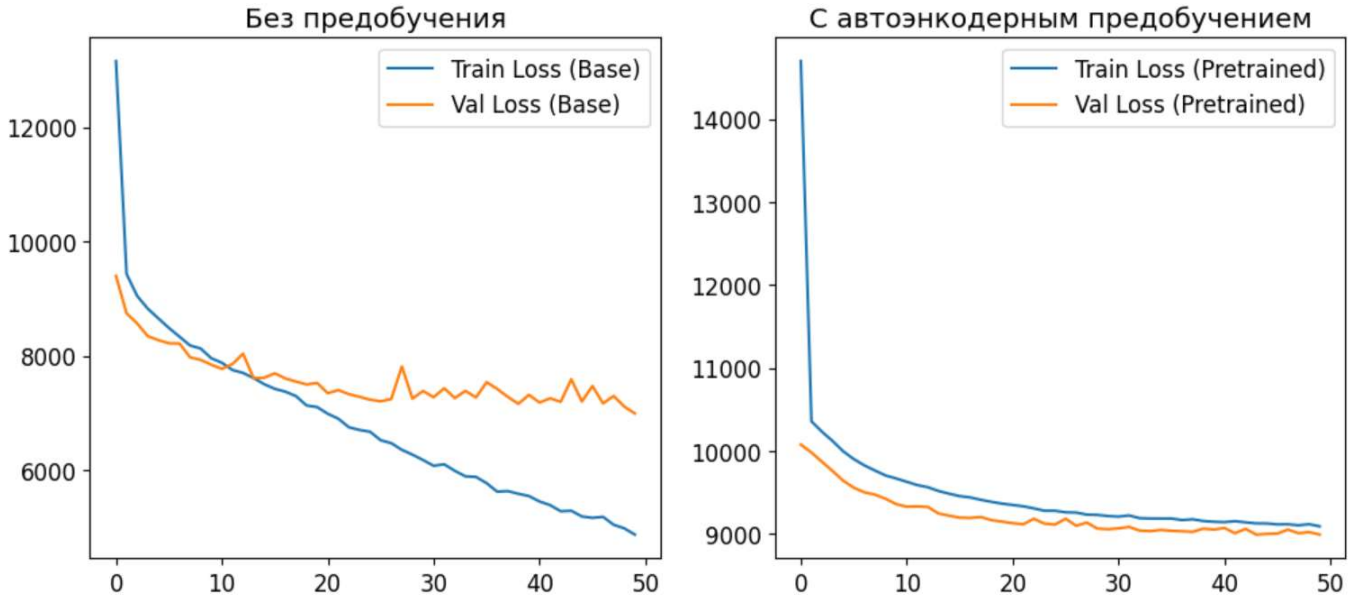
```
history = model.fit(
    X_train_scaled,
    y_train,
    validation_split=0.2,
    epochs=50,
    batch_size=64,
    verbose=1
)

y_pred = model.predict(X_test_scaled)
mape = mean_absolute_percentage_error(y_test, y_pred)
```



Предыдущие модели

Without pretraining — MAPE: 0.4137

With pretraining — MAPE: 0.6642



Вывод:

Сравнение трёх подходов показывает, что **наилучшее качество** достигается при обучении модели **без какого-либо предварительного обучения** (MAPE = 0.4137). Это означает, что сама сеть эффективно адаптируется к данным и не испытывает проблем с локальными минимумами или плохой инициализацией весов.

Использование **RBM-предобучения** даёт промежуточный результат (MAPE = 0.5201). Несмотря на успешное обучение стековых RBM, полученное распределение признаков, вероятно, не совпало с оптимальным для решаемой регрессионной задачи. RBM лучше проявляют себя в задачах плотностного моделирования или классификации, поэтому их представления могли внести избыточный шум.

Предобучение с помощью **автоэнкодера** демонстрирует **худший результат** (MAPE = 0.6642). Это может быть связано с тем, что автоэнкодеры стремятся восстанавливать вход, а не минимизировать ошибку прогноза целевой переменной, что приводит к формированию признаков, не оптимальных для регрессии.