Практическая работа № 4

**Анализ временных рядов**

**1. Постановка задачи**

**Цель**: Построить модель для классификации аномальных сердечных сокращений (аритмий) по временным рядам ЭКГ.  
**Датасет**: [ECG Heartbeat Categorization Dataset](https://www.kaggle.com/datasets/shayanfazeli/heartbeat) (ЭКГ-сигналы для классификации аритмий)

* **Данные**:
  + 5 классов сердечных сокращений:
    - 0: Нормальное (N),
    - 1: Наджелудочковая аритмия (S),
    - 2: Желудочковая аритмия (V),
    - 3: Фибрилляция (F),
    - 4: Шум (Q).
  + Каждый временной ряд содержит 187 отсчетов (≈1 сердечный цикл).
* **Размер**: 109,446 примеров (сбалансированные классы).

**Задача**:

* Классифицировать тип аритмии по временному ряду ЭКГ.
* Учесть временные зависимости и шумы в сигналах.

**2. Подготовка данных**

**Шаги**:

1. **Загрузка данных**:

import pandas as pd

import numpy as np

# Пример загрузки данных (CSV-файлы с сигналами и метками)

df = pd.read\_csv("mitbih\_train.csv", header=None)

X = df.iloc[:, :-1].values # Временные ряды (187 точек)

y = df.iloc[:, -1].values # Метки классов (0-4)

1. **Нормализация**:

from sklearn.preprocessing import StandardScaler

# Нормализация каждого сигнала отдельно

X\_normalized = np.zeros\_like(X)

for i in range(X.shape[0]):

X\_normalized[i] = (X[i] - X[i].mean()) / X[i].std()

1. **Разделение на выборки**:

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_normalized, y, test\_size=0.2, stratify=y, random\_state=42

)

1. **Преобразование в 3D-тензор** (для моделей временных рядов):

X\_train = X\_train.reshape(X\_train.shape[0], X\_train.shape[1], 1)

X\_test = X\_test.reshape(X\_test.shape[0], X\_test.shape[1], 1)

**3. Построение модели**

**Архитектура**: **LSTM** + **CNN** (гибридная модель для временных рядов).

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Conv1D, MaxPooling1D, Dense, Dropout, Flatten

model = Sequential()

# Слой Conv1D для извлечения локальных паттернов

model.add(Conv1D(filters=64, kernel\_size=5, activation='relu', input\_shape=(187, 1)))

model.add(MaxPooling1D(pool\_size=2))

# Слой LSTM для учета временных зависимостей

model.add(LSTM(100, return\_sequences=True))

model.add(Dropout(0.5))

model.add(LSTM(50))

model.add(Dense(64, activation='relu'))

model.add(Dense(5, activation='softmax')) # 5 классов

model.compile(

optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy']

)

model.summary()

**4. Обучение модели**

**Параметры**:

* Батч: 64,
* Эпохи: 20,
* Callbacks: EarlyStopping, ModelCheckpoint.

**Код**:

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

callbacks = [

EarlyStopping(patience=5, restore\_best\_weights=True),

ModelCheckpoint("best\_model.h5", save\_best\_only=True)

]

history = model.fit(

X\_train, y\_train,

epochs=20,

batch\_size=64,

validation\_split=0.1,

callbacks=callbacks

)

**5. Оценка модели**

**Метрики**:

* **Accuracy**, **F1-Score** (мультиклассовый),
* **ROC-AUC** (One-vs-Rest для каждого класса).

**Код**:

from sklearn.metrics import classification\_report, roc\_auc\_score

y\_pred = model.predict(X\_test)

y\_pred\_classes = np.argmax(y\_pred, axis=1)

print(classification\_report(y\_test, y\_pred\_classes))

# ROC-AUC для каждого класса

for class\_id in range(5):

auc = roc\_auc\_score((y\_test == class\_id).astype(int), y\_pred[:, class\_id])

print(f"Class {class\_id}: AUC = {auc:.3f}")

**Пример вывода**:

precision recall f1-score support

0 0.99 0.99 0.99 18118

1 0.93 0.93 0.93 556

2 0.97 0.97 0.97 1448

3 0.96 0.96 0.96 162

4 0.99 0.99 0.99 1606

accuracy 0.99 21890

macro avg 0.97 0.97 0.97 21890

weighted avg 0.99 0.99 0.99 21890

Class 0: AUC = 0.999

Class 1: AUC = 0.985

...

**Визуализация**:

* График обучения:

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.legend()

plt.show()

* Пример сигнала и предсказания:

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

plt.plot(X\_test[0], label='ЭКГ-сигнал')

plt.title(f"Истинный класс: {y\_test[0]}, Предсказанный: {y\_pred\_classes[0]}")

plt.legend()

plt.show()

**6. Анализ ошибок**

* **Класс 1 (S) и 2 (V)**: Частая путаница из-за схожих паттернов.
* **Класс 3 (F)**: Редкие случаи фибрилляции, требующие увеличения данных.  
  **Рекомендации**:

1. Добавить аугментацию временных рядов (добавление шума, сдвиг во времени).
2. Использовать **Attention-механизмы** для выделения ключевых участков сигнала.
3. Применить **Transfer Learning** с предобученными на ЭКГ моделями.

**7. Пример улучшения модели**

**Добавление Attention-слоя**:

from tensorflow.keras.layers import Layer, Dense

class Attention(Layer):

def \_\_init\_\_(self, \*\*kwargs):

super(Attention, self).\_\_init\_\_(\*\*kwargs)

def build(self, input\_shape):

self.W = self.add\_weight(name='attention\_weight', shape=(input\_shape[-1], 1), initializer='glorot\_uniform')

super(Attention, self).build(input\_shape)

def call(self, x):

e = tf.tanh(tf.matmul(x, self.W))

a = tf.nn.softmax(e, axis=1)

output = x \* a

return tf.reduce\_sum(output, axis=1)

# Обновленная модель

model = Sequential()

model.add(Conv1D(64, 5, activation='relu', input\_shape=(187, 1)))

model.add(MaxPooling1D(2))

model.add(LSTM(100, return\_sequences=True))

model.add(Attention()) # Attention вместо GlobalAveragePooling

model.add(Dense(5, activation='softmax'))

**8. Выводы**

1. Гибридные модели (CNN + LSTM) эффективны для анализа временных рядов ЭКГ.
2. Attention-механизмы улучшают интерпретируемость и точность.
3. Аугментация данных критична для редких классов (например, фибрилляция).
4. Мультиклассовый ROC-AUC помогает оценить качество для каждого типа аритмии.