

Нижегородский государственный университет им. Н.И. Лобачевского  
Институт информационных технологий, математики и механики  
Кафедра Математического обеспечения и суперкомпьютерных технологий

**Архипелаг 20.35**  
**Трек «ИИ в здравоохранении»**

**Лабораторная работа №1**  
**Построение алгоритма классификации ЭКГ**

*При поддержке компании Intel*

*Васильев Е.П.*

Нижний Новгород  
2020

# Содержание

1	Введение.....	4
2	Методические указания .....	4
2.1	Цели и задачи работы .....	4
2.2	Структура работы.....	4
2.3	Рекомендации по проведению занятий .....	4
3	Установка Intel Distribution of OpenVINO Toolkit и его зависимостей для Python .....	4
3.1	Установка Python 3.....	4
3.2	Создание виртуальной среды Python.....	5
3.3	Установка Intel Distribution of OpenVINO Toolkit.....	5
3.4	Установка дополнительных модулей Python.....	5
4	Запуск примеров и демо-приложений OpenVINO на языке Python .....	6
4.1	Настройка окружения OpenVINO.....	6
4.2	Скачивание моделей .....	6
4.3	Конвертация моделей.....	6
4.4	Запуск примеров для классификации изображений .....	7
5	Разработка приложения для классификации изображений с использованием OpenVINO.....	7
5.1	Рабочие скрипты .....	7
5.2	Загрузка модели.....	7
5.3	Загрузка и предобработка изображения.....	7
5.4	Вывод модели .....	7
5.5	Обработка выхода модели.....	7
5.6	Создание тестирующего примера.....	7
5.6.1	Разбор параметров командной строки .....	7
5.6.2	Создание основной функции.....	7
5.7	Запуск приложения .....	7
6	Разработка приложения для классификации ЭКГ с использованием OpenVINO.....	8
6.1	Конвертация модели .....	8
6.2	Рабочие скрипты .....	8
6.3	Загрузка модели.....	9
6.4	Загрузка и предобработка данных.....	9
6.5	Вывод модели .....	9
6.6	Создание тестирующего примера.....	10
6.6.1	Разбор параметров командной строки .....	10
6.6.2	Создание основной функции.....	10
6.7	Запуск приложения .....	10
7	Дополнительные задания.....	11
8	Литература .....	11

8.1	Основная литература .....	11
8.2	Дополнительная литература.....	11
8.3	Ресурсы сети Интернет .....	11

# 1 Введение

Настоящая практическая работа является вводной и предусматривает создание инфраструктуры для выполнения последующих работ. Здесь дается описание процедуры установки Intel Distribution of OpenVINO Toolkit [3] и настройки окружения для работы с данным инструментом.

Решение задач, поставленных в ходе выполнения практических работ, осуществляется на языке Python 3. В качестве натренированных моделей глубокого обучения используются модели из множества обученных моделей Open Model Zoo [3]. В данной практической работе рассматривается задача классификации изображений и предлагается общая схема ее решения средствами Intel Distribution of OpenVINO Toolkit.

## 2 Методические указания

### 2.1 Цели и задачи работы

*Цель работы* состоит в изучении глубоких моделей для решения задачи классификации изображений с применением инструмента Intel Distribution of OpenVINO Toolkit.

Для достижения поставленной цели необходимо решить следующие задачи:

- Установить Intel Distribution of OpenVINO Toolkit.
- Настроить рабочее окружение.
- Изучить структуру и состав Intel Distribution of OpenVINO Toolkit.
- Скачать и конвертировать глубокую классификационную модель.
- Выполнить классификацию изображения.

### 2.2 Структура работы

В работе приводится руководство по установке Intel Distribution of OpenVINO Toolkit и необходимых зависимостей. Руководство включает описание последовательности действий, которую следует выполнить из командной строки для настройки рабочего окружения и проведения экспериментов по классификации при помощи глубоких моделей. Далее поэтапно разрабатывается программный код, решающий задачу классификации изображений.

### 2.3 Рекомендации по проведению занятий

При выполнении данной практической работы рекомендуется следующая последовательность действий:

- Выполнить установку Intel Distribution of OpenVINO Toolkit и зависимостей.
- Настроить рабочее окружение.
- Изучить структуру и состав Intel Distribution of OpenVINO Toolkit, опираясь на лекционный материал курса и дополнительные источники.
- Разработать программный код для решения задачи классификации с применением компонента Inference Engine, входящего в состав инструмента Intel Distribution of OpenVINO Toolkit, проверить его работоспособность.

## 3 Установка Intel Distribution of OpenVINO Toolkit и его зависимостей для Python

### 3.1 Установка Python 3

Для работы OpenVINO 2021.1 лучше всего использовать последнюю версию Python 3.8 (также можно использовать Python 3.6 или 3.7), которую можно скачать с официального сайта [5].

Информация, приведенная ниже, актуальна для пользователей операционной системы Windows. Если при установке путь до бинарных файлов Python не добавлен в переменную окружения **PATH**,

то для работы с Python из командной строки требуется выполнить следующую команду, чтобы сделать его доступным из командной строки:

```
set  
PATH="C:\Users\<USERNAME>\AppData\Local\Programs\Python\Python38;%PATH%"
```

### 3.2 Создание виртуальной среды Python

Библиотеки на языке Python устанавливаются в систему подобно тому, как устанавливаются программы в операционную систему. При этом может потребоваться несколько разных версий одной библиотеки. Для этого можно создать несколько окружений – *виртуальных сред Python*, в которых будут установлены разные версии библиотеки.

Для того, чтобы создать новую виртуальную среду Python, выполните команды, показанные ниже.

```
mkdir openvino-virtual-environments && cd openvino-virtual-environments  
python -m venv openvinoenv
```

Чтобы активировать виртуальную среду, выполните одну из следующих команд.

В Windows:

```
openvino-virtual-environments\bin\activate.bat
```

В Linux:

```
source openvino-virtual-environments/bin/activate
```

В дальнейшем при работе с Python необходимо активировать существующую виртуальную среду Python. Для удобства можно сохранить все команды в текстовый файл, чтобы в следующий раз не набирать их вручную, либо создать скрипт, активирующий Python, виртуальную среду и OpenVINO.

### 3.3 Установка Intel Distribution of OpenVINO Toolkit

Для установки OpenVINO необходимо скачать инсталлятор с официального сайта [3]. Для скачивания потребуется бесплатная регистрация. Во время скачивания вам станет доступен ключ. Сохранять ключ не обязательно, ПО работает без активации.

### 3.4 Установка дополнительных модулей Python

В лабораторной работе все пути представлены в Windows формате (обратные слэши в путях, пути с пробелами как Program Files должны быть в кавычках). При работе в Linux и Mac пути будут аналогичными до уровня папки установки OpenVINO.

Для работы с OpenVINO необходимо конвертировать модель из оригинального фреймворка модели в промежуточный формат (Intermediate representation, IR) OpenVINO. Для конвертации требуется установить актуальную версию тренировочного фреймворка. Можно установить один интересующий фреймворк или все сразу, используя одну из следующих команд.

Для всех фреймворков:

```
pip install -r "C:\Program Files  
(x86)\Intel\openvino_2021\deployment_tools\model_optimizer\requirements.tx  
t"
```

Только для Caffe:

```
pip install -r "C:\Program Files  
(x86)\Intel\openvino_2021\deployment_tools\model_optimizer\requirements_ca  
ffe.txt"
```

Для скачивания моделей из зоопарка моделей Open Model Zoo требуется установить зависимости для работы с сетью из Python.

```
pip install -r "C:\Program Files
(x86)\Intel\openvino_2021\deployment_tools\open_model_zoo\tools\downloader
\requirements.in"
```

После данных действий ваш виртуальный энвайрмент полностью готов к разработке на языке Python с использованием OpenVINO.

## 4 Запуск примеров и демо-приложений OpenVINO на языке Python

### 4.1 Настройка окружения OpenVINO

После того, как установлен OpenVINO, создана и активирована виртуальная среда, необходимо добавить Python-библиотеки OpenVINO в переменную окружения **PATH** с помощью одной из нижеперечисленных команд.

В Windows:

```
"C:\Program Files (x86)\Intel\openvino_2021\bin\setupvars.bat"
```

В Linux:

```
source ~/intel/openvino_2021/bin/setupvars.sh
```

### 4.2 Скачивание моделей

Open Model Zoo [3] – репозиторий глубоких нейросетевых моделей, содержащий большое количество обученных моделей, которые могут исполняться при помощи OpenVINO. Данный репозиторий хранит не только модели, но и параметры для конвертации моделей из разных фреймворков в промежуточный формат OpenVINO.

Для скачивания моделей из репозитория Open Model Zoo нужно воспользоваться инструментом Model Downloader, точка входа – скрипт **download.py**.

```
python "C:\Program Files
(x86)\Intel\openvino_2021\deployment_tools\tools\model_downloader\download
er.py" --name <model_name> --output_dir <destination_folder>
```

где **<model\_name>** – название скачиваемой модели, а **<destination\_folder>** – директория, в которую необходимо скачать модель.

Список доступных для скачивания моделей можно получить посредством указания ключа **--print\_all**:

```
python "C:\Program Files
(x86)\Intel\openvino_2021\deployment_tools\tools\model_downloader\download
er.py" --print_all
```

Для решения задачи классификации неплохой моделью, являющейся хорошим компромиссом между производительностью и качеством, является модель Squeezenet1.1.

### 4.3 Конвертация моделей

Для конвертации загруженных моделей нужно воспользоваться инструментом Model Optimizer и входящим в него модулем **converter.py**. Данный модуль имеет доступ к параметрам конвертации моделей из зоопарка моделей.

```
python "C:\Program Files
(x86)\Intel\openvino_2021\deployment_tools\tools\model_downloader\converte
r.py" --name <model_name> --download_dir <destination_folder>
```

Для конвертации собственных моделей необходимо узнать дополнительные параметры и использовать модуль **mo.py**, это будет рассмотрено позже.

## 4.4 Запуск примеров для классификации изображений

# 5 Разработка приложения для классификации изображений с использованием OpenVINO

## 5.1 Рабочие скрипты

## 5.2 Загрузка модели

## 5.3 Загрузка и предобработка изображения

## 5.4 Вывод модели

Следующий этап – реализация метода классификации изображения **classify**, который запускает вывод глубокой модели на устройстве, указанном в конструкторе. Логика работы метода **classify** следующая:

1. Получить данные о входе и выходе нейронной сети.

```
input_blob = next(iter(self.net.input_info))
out_blob = next(iter(self.net.outputs))
```

2. Из данных о входе нейронной сети получить требуемые нейросетью размеры для входного изображения.

```
n, c, h, w = self.net.input_info[input_blob].input_data.shape
```

3. С помощью функции **\_prepare\_image** подготовить изображение.
4. Вызвать функцию синхронного исполнения модели.

```
output = self.exec_net.infer(inputs = {input_blob: blob})
```

5. Из выхода модели получить тензор с результатом классификации.

```
output = output[out_blob]
```

## 5.5 Обработка выхода модели

## 5.6 Создание тестирующего примера

## 5.7 Запуск приложения

Запуск разработанного приложения удобнее всего произвести из командной строки. Для этого необходимо открыть командную строку. Строка запуска будет иметь следующий вид:

```
python ie_classification_sample.py -i image.jpg -m squeezenet1.1.xml \
-w squeezenet1.1.bin -c imagenet_synset_words.txt
```

Аргумент **-i** задает путь к изображению, аргумент **-m** задает путь к конфигурации модели, аргумент **-w** задает путь к весам модели, аргумент **-c** задает путь к файлу с именами классов для модели.

Результат запуска приложения должен выглядеть следующим образом. Выводится сообщение о старте приложения, затем выводится список классов и их вероятность.

```
[ INFO ] Start IE classification sample
[ INFO ] Predictions: [['n02099712 Labrador retriever',
67.87383556365967], ['n02108422 bull mastiff', 11.615122854709625],
['n02099601 golden retriever', 7.8480251133441925], ['n02109525 Saint
Bernard, St Bernard', 2.984011545777321], ['n02089973 English foxhound',
1.2024826370179653]]
```

## 6 Разработка приложения для классификации ЭКГ с использованием OpenVINO

### 6.1 Конвертация модели

Для конвертации загруженных моделей нужно воспользоваться инструментом Model Optimizer, который располагается по пути: `openvino_2021.1.110\deployment_tools\model_optimizer\mo.py`. Данный модуль анализирует архитектуру глубоких моделей. Вызовите скрипт `mo.py` с одним параметром – путем до вашей модели классификации ЭКГ.

```
python "C:\Program Files (x86)\Intel\openvino_2021\
deployment_tools\model_optimizer\mo.py" --input_model <path_to_your_model>
```

Если все слои поддерживаются в Model Optimizer, то вы увидите следующий текст в консоли:

```
Model Optimizer arguments:
Common parameters:
  - Path to the Input Model:
C:\Users\eugene\Desktop\2035\arrhythmia.onnx
  - Path for generated IR:      C:\Users\eugene\Desktop\2035\.
  - IR output name:            arrhythmia
  - Log level:                 ERROR
  - Batch:                     Not specified, inherited from the model
  - Input layers:              Not specified, inherited from the model
  - Output layers:            Not specified, inherited from the model
  - Input shapes:              Not specified, inherited from the model
  - Mean values:               Not specified
  - Scale values:              Not specified
  - Scale factor:              Not specified
  - Precision of IR:           FP32
  - Enable fusing:             True
  - Enable grouped convolutions fusing: True
  - Move mean values to preprocess section: None
  - Reverse input channels:    False
ONNX specific parameters:
Model Optimizer version:      2021.1.0-1237-bece22ac675-releases/2021/1

[ SUCCESS ] Generated IR version 10 model.
[ SUCCESS ] XML file: C:\Users\eugene\Desktop\2035\.\arrhythmia.xml
[ SUCCESS ] BIN file: C:\Users\eugene\Desktop\2035\.\arrhythmia.bin
```

В выходной папке появятся файлы с расширением `.xml` и `.bin` – модель в формате OpenVINO. Данные модели готовы к запуску в OpenVINO.

### 6.2 Рабочие скрипты

Для выполнения классификации ЭКГ необходимо создать файл: файл `ie_ecg_classifier.py`, содержащий класс классификатора изображений `ECGClassifier` с методами `_prepare_data`, `classify` и тестирующий код для запуска `ECGClassifier`.

Методы класса `ECGClassifier`:

- `_init_` – конструктор класса, инициализирует Inference Engine и загружает модель (ключевое слово `pass` добавлено чтобы код компилировался даже когда тело функции пусто);



- **\_prepare\_data** – метод, который преобразует ЭКГ в формат входа нейронной сети;
- **classify** – метод классификации ЭКГ при помощи нейронной сети;

```
class ECGClassifier:
    def __init__(self, configPath = None, weightsPath = None,
                 device = 'CPU'):
        pass

    def _prepare_data(self, data):
        pass

    def classify(self, data):
        pass
```

### 6.3 Загрузка модели

Для того, чтобы выполнить загрузку модели, в файле **ECGClassifier.py** необходимо реализовать конструктор класса **ECGClassifier**. Подробно прочитать про загрузку модели в конечное устройство можно в лекции №3 «Обзор инструмента Intel Distribution of OpenVINO Toolkit». Конструктор получает следующие обязательные и необязательные параметры:

- **configPath** – путь до xml-файла с описанием модели.
- **weightsPath** – путь до bin-файла с весами модели.

Конструктор выполняет следующие действия:

1. Создание объекта класса **IECore**.

```
self.ie = IECore()
```

2. Создание объекта **INetwork** загруженной с диска модели с помощью функции **ieCore.read\_network**, параметрами этой функции являются пути до модели.

```
self.net = self.ie.read_network(model=configPath, weights=weightsPath)
```

3. Создание объекта **ExecutableNetwork** объекта с помощью функции **ieCore.load\_network**, параметрами этой функции загруженная модель, устройство для инференса и параметры запуска.

```
self.exec_net = self.ie.load_network(network=self.net,
                                     device_name=device)
```

### 6.4 Загрузка и предобработка данных

В реальных приложениях вам может понадобиться предобработать ваши данные – заполнить неизвестные значения, привести к входному формату модели. Мы же проверим что наш входной вектор такого же размера, что и вход модели.

```
def _prepare_data(self, df, n, l):
    data = df.astype(np.float)
    if data.shape[-1] != l:
        raise RuntimeError('Input data is not acceptable to model')
    return data
```

### 6.5 Вывод модели

Следующий этап – реализация метода классификации изображения **classify**, который запускает вывод глубокой модели на устройстве, указанном в конструкторе. Логика работы метода **classify** следующая:

6. Получить данные о входе и выходе нейронной сети.

```
input_blob = next(iter(self.net.input_info))
out_blob = next(iter(self.net.outputs))
```

7. Из данных о входе нейронной сети получить требуемые нейросетью размеры для входного изображения.

```
n, l = self.net.input_info[input_blob].input_data.shape
```

8. С помощью функции `_prepare_image` подготовить изображение.
9. Вызвать функцию синхронного исполнения модели.

```
output = self.exec_net.infer(inputs = {input_blob: blob})
```

10. Из выхода модели получить тензор с результатом классификации.

```
output = output[out_blob]
```

## 6.6 Создание тестирующего примера

### 6.6.1 Разбор параметров командной строки

### 6.6.2 Создание основной функции

Создайте функцию `main`, которая выполняет следующие действия:

6. Разбор аргументов командной строки.
7. Создание объекта класса `ECGClassifier` с необходимыми параметрами.
8. Чтение ЭКГ.
9. Классификация ЭКГ.
10. Вывод результата классификации на экран.

Для вывода логов в консоль предлагается использовать пакет `logging`.

```
def main():
    log.basicConfig(format="[ %(levelname)s ] %(message)s",
                    level=log.INFO, stream=sys.stdout)
    args = build_argparser().parse_args()

    log.info("Start ECG classification sample")

    ie_classifier = ECGClassifier(configPath=args.model,
                                weightsPath=args.weights, device=args.device)

    # Read data

    data = pd.read_csv(args.input, header=None, na_values="?")
    # Start inference

    prob = ie_classifier.classify(data)
    log.info("Predictions: " + str(prob))

    return

if __name__ == '__main__':
    sys.exit(main())
```

## 6.7 Запуск приложения

Запуск разработанного приложения удобней всего произвести из командной строки. Для этого необходимо открыть командную строку. Строка запуска будет иметь следующий вид:

```
python ECGclassification.py -i test1.txt -m ECG.xml -w ECG.bin
```

Аргумент `-i` задает путь к текстовому файлу, аргумент `-m` задает путь к конфигурации модели, аргумент `-w` задает путь к весам модели, аргумент

Результат запуска приложения должен выглядеть следующим образом. Выводится сообщение о старте приложения, затем выводится список классов и их вероятность.

```
[ INFO ] Start ECG classification sample  
[ INFO ] Predictions: [[-223.05115]]
```

## 7 Дополнительные задания

Созданный пример классификации содержит минимально необходимый функционал. В качестве дополнительных заданий предлагается обеспечить поддержку следующих возможностей:

1. Поддержка классификации не только одной картинке, но и набора из нескольких изображений.
2. Поддержка выполнения вывода глубоких моделей не только на CPU, но и на Intel Processor Graphics или Neural Compute Stick (при наличии).
3. Поддержка асинхронного вывода глубоких моделей.

Данные задания предлагается выполнить самостоятельно, опираясь на документацию и примеры, входящие состав пакета OpenVINO.

## 8 Литература

### 8.1 Основная литература

1. Шолле Ф. Глубокое обучение на Python. – СПб.: Питер. – 2018. – 400с.

### 8.2 Дополнительная литература

2. Рамальо Л. Python. К вершинам мастерства / Пер. с англ. Слинкин А.А. – М.: ДМК Пресс. – 2016. – 768с.

### 8.3 Ресурсы сети Интернет

3. Страница репозитория Open Model Zoo [[https://github.com/openvinotoolkit/open\\_model\\_zoo](https://github.com/openvinotoolkit/open_model_zoo)].
4. Страница скачивания Intel Distribution of OpenVINO Toolkit [<https://software.intel.com/en-us/openvino-toolkit/choose-download>].
5. Страница скачивания Python 3.8.6 [<https://www.python.org/downloads/release/python-386/>].
6. Документация Intel Distribution of OpenVINO Toolkit [<https://docs.openvinotoolkit.org/latest/index.html>].
7. OpenVINO classification sample [[https://docs.openvinotoolkit.org/latest/\\_inference\\_engine\\_ie\\_bridges\\_python\\_sample\\_classification\\_sample\\_README.html](https://docs.openvinotoolkit.org/latest/_inference_engine_ie_bridges_python_sample_classification_sample_README.html)].