

Федеральное государственное бюджетное образовательное учреждение высшего образования
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ МЕДИЦИНСКИЙ УНИВЕРСИТЕТ»
Министерства здравоохранения Российской Федерации
Передовая медицинская инженерная школа

«СОГЛАСОВАНО»

УМО ИПО
протокол № 2 от 01 марта 2023

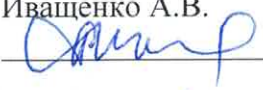
«УТВЕРЖДАЮ»

Проректор по профессиональному
образованию и межрегиональному
взаимодействию, директор ИПО,
д.м.н., МВА


С.А Палевская
« 01 » марта 2023 г.

**Дополнительная профессиональная программа профессиональной переподготовки
инженеров по направлению подготовки
09.04.01 Информатика и вычислительная техника
«ЦИФРОВИЗАЦИЯ В ЗДРАВООХРАНЕНИИ»
со сроком освоения 256 часов
(форма обучения очная)**

Программа рассмотрена и утверждена
на заседании Передовой медицинской
инженерной школы
(протокол № 1 от 15 февраля 2023 г.)
Директор ПИИШ, д.т.н., профессор
Ивашенко А.В.


« 27 » февраля 2023 г.

Разработчики:

Ивашекно Антон Владимирович - доктор технических наук, профессор, директор передовой медицинской инженерной школы

Куликовских Илона Марковна – доктор технических наук, руководитель учебного центра цифровых образовательных программ

Чертыковцева Наталья Валерьевна – кандидат технических наук, заместитель директора передовой медицинской инженерной школы

ДОПОЛНИТЕЛЬНАЯ ПРОГРАММА ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ «ЦИФРОВИЗАЦИЯ В ЗДРАВООХРАНЕНИИ»

Дополнительная профессиональная программа профессиональной переподготовки «Цифровизация в здравоохранении» (далее – программа) разработана на основе ФГОС ВО по направлению подготовки 09.04.01 Информатика и вычислительная техника (уровень магистратуры), утвержденного приказом Минобрнауки России от 19 сентября 2017 г. № 918, а также профессионального стандарта «Руководитель разработки программного обеспечения» утвержденного приказом Министерства труда и социальной защиты РФ от 22 июля 2022 г. № 423н.

I. ОБЩИЕ ПОЛОЖЕНИЯ

Цель Программы: подготовка инженерных кадров, способных решать задачи разработки новой медицинской техники, средств реабилитации и диагностики, а также телемедицинских систем на основе комбинированного владения информационными технологиями и глубокого понимания особенностей назначения и применения инженерных решений в медицине.

Направленность Программы: совершенствование, повышение профессионального уровня в рамках имеющейся квалификации, а также качественное изменение профессиональных компетенций в области информационных технологий; обучение навыкам разработки программного обеспечения для реализации проектов по цифровой трансформации отрасли здравоохранения.

Задачи Программы:

- Обновление существующих теоретических знаний, методик и изучение передового практического опыта в области применения современных информационных технологий, технологий искусственного интеллекта и робототехники в медицинских приложениях.
- Усвоение и закрепление на практике профессиональных знаний, умений и навыков, обеспечивающих приобретение новых профессиональных компетенций по вопросам разработки программного обеспечения для реализации проектов по цифровой трансформации отрасли здравоохранения.

Трудоемкость освоения: 256 академических часов.

Форма обучения: очная (с применением дистанционных образовательных технологий)

Категория обучающихся:

К обучению по программе профессиональной переподготовки допускаются лица, получившие высшее образование по очной (очно-заочной) форме, лица, освоившие основную образовательную программу бакалавриата, специалитета, а также магистратуры.

Основными компонентами Программы являются:

- цель программы;
- планируемые результаты обучения;
- учебный план;
- требования к итоговой аттестации обучающихся;
- календарный учебный график;
- организационно-педагогические условия реализации Программы;
- оценочные материалы и иные компоненты.

Содержание Программы построено в соответствии с модульным принципом, структурными единицами модуля являются разделы. Каждый раздел дисциплины подразделяется на темы, каждая тема - на элементы, каждый элемент - на подэлементы. Для удобства пользования программой в учебном процессе каждая его структурная единица кодируется. На первом месте ставится код раздела дисциплины (например, 1), на втором - код темы (например, 1.1), далее - код элемента (например, 1.1.1), затем – код подэлемента (например, 1.1.1.1). Кодировка вносит определенный порядок в перечень вопросов, содержащихся в программе, что, в свою очередь, позволяет кодировать контрольно-измерительные (тестовые) материалы в учебно-методическом комплексе (далее - УМК).

Учебный план определяет состав изучаемых дисциплин с указанием их трудоемкости, объема, последовательности и сроков изучения, устанавливает формы организации учебного процесса и их соотношение (лекции, лабораторные и практические занятия), конкретизирует формы контроля знаний и умений обучающихся.

Результаты обучения: в программу включены планируемые результаты обучения. Планируемые результаты обучения направлены на получение компетенций, необходимых для выполнения нового вида профессиональной деятельности в области применения современных информационных технологий, технологий искусственного интеллекта и робототехники в медицинских приложениях; приобретение новой квалификации «Руководитель разработки программного обеспечения».

В Программе содержатся требования к аттестации обучающихся. Итоговая аттестация по Программе осуществляется посредством проведения демонстрационного экзамена и выявляет теоретическую и практическую подготовку слушателя в соответствии с целями и содержанием программы.

Организационно-педагогические условия реализации программы.

Условия реализации Программы включают:

- а) учебно-методическую документацию и материалы по всем разделам (модулям) специальности;
- б) учебно-методическую литературу для внеаудиторной работы обучающихся;
- в) материально-технические базы, обеспечивающие организацию всех видов дисциплинарной подготовки:
 - учебные аудитории, оснащенные материалами и оборудованием для проведения учебного процесса;
 - клинические базы в медицинских и научных организациях;
- в) кадровое обеспечение реализации Программы соответствует требованиям штатного расписания Передовой медицинской инженерной школы, реализующей дополнительные профессиональные программы;
- г) законодательство Российской Федерации.

II. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ СЛУШАТЕЛЕЙ, УСПЕШНО ОСВОИВШИХ ПРОГРАММУ

Результаты обучения по Программе направлены на формирование новой профессиональной компетенции на основе Федерального государственного образовательного стандарта высшего образования по направлению подготовки 09.04.01 «Информатика и вычислительная техника»¹.

¹Приказ Министерства образования и науки РФ от 19 сентября 2017 г. № 918 «Об утверждении федерального государственного образовательного стандарта высшего образования – магистратура по направлению подготовки 09.04.01 Информатика и вычислительная техника»

Характеристика новых профессиональных компетенций формирующихся в результате освоения Программы

В результате освоения программы слушатель осваивает следующую профессиональную компетенцию:

- Способен руководить проектированием специализированного программного обеспечения в рамках цифровой трансформации сферы здравоохранения (ПК-1).

Трудовые действия (функции) руководителя разработки программного обеспечения

Трудовая функция (профессиональная компетенция)	Трудовые действия	Необходимые умения	Необходимые знания
Обобщенная трудовая функция: организация процессов разработки компьютерного программного обеспечения			
В/01.7 Управление проектированием компьютерного программного обеспечения	Анализ архитектуры компьютерного программного обеспечения и ее согласование с заинтересованными сторонами Распределение заданий на проектирование компьютерного программного обеспечения, структуры базы данных, программных интерфейсов Оценка качества проектирования компьютерного программного обеспечения, структуры базы данных, программных интерфейсов Принятие управленческих решений по результатам проектирования компьютерного программного обеспечения, структуры базы данных, программных интерфейсов	Применять принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения Применять методологии и средства проектирования компьютерного программного обеспечения Применять методы и средства проектирования баз данных Применять методы и средства проектирования программных интерфейсов Применять основные принципы и методы управления персоналом Применять методологию функциональной стандартизации для открытых систем Применять методы принятия управленческих решений Применять нормативно-технические документы (стандарты и регламенты) по процессу разработки архитектуры компьютерного программного обеспечения Осуществлять рабочие коммуникации с подразделениями организации и заинтересованными сторонами в рамках процесса проектирования компьютерного программного обеспечения, структуры базы данных, программных интерфейсов	Принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения Методологии и средства проектирования компьютерного программного обеспечения Методы и средства проектирования баз данных Методы и средства проектирования программных интерфейсов Методы принятия управленческих решений Основные принципы и методы управления персоналом Методология функциональной стандартизации для открытых систем Технологии межличностной и групповой коммуникации в деловом взаимодействии, основы конфликтологии

Перечень знаний, умений и навыков, обеспечивающих формирование профессиональных компетенций

По окончании обучения слушатель должен знать:

- принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения
- методологии и средства проектирования компьютерного программного обеспечения
- методы и средства проектирования баз данных
- методы и средства проектирования программных интерфейсов
- методология функциональной стандартизации для открытых систем
- методы принятия управленческих решений

По окончании обучения слушатель должен уметь:

- применять принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения
- применять методологии и средства проектирования компьютерного программного обеспечения
- применять методы и средства проектирования баз данных
- применять методы и средства проектирования программных интерфейсов
- применять основные принципы и методы управления персоналом
- применять методологию функциональной стандартизации для открытых систем
- применять методы принятия управленческих решений
- применять нормативно-технические документы (стандарты и регламенты) по процессу разработки архитектуры компьютерного программного обеспечения

По окончании обучения слушатель должен владеть:

- навыками анализа архитектуры компьютерного программного обеспечения;
- навыками распределения заданий на проектирование компьютерного программного обеспечения, структуры базы данных, программных интерфейсов;
- навыками оценки качества проектирования компьютерного программного обеспечения, структуры базы данных, программных интерфейсов;

III. РАБОЧАЯ ПРОГРАММА

РАЗДЕЛ 1 «Управление проектами по цифровой трансформации здравоохранения»

Код	Наименования тем, элементов и подэлементов
1.	Стратегия цифровой трансформации отрасли «Здравоохранения»
1.1.	Создание единого цифрового контура в здравоохранении.
1.2.	Медицинские платформенные решения федерального уровня.
1.3.	Персональные медицинские помощники.
1.4.	Цифровые платформы и технологии.
1.5.	Инструменты цифровой трансформации.
2.	Методы и средства цифровой трансформации проектов.
2.1.	Понятия и определения цифровой трансформации.
2.2.	Применение современных информационных технологий.
2.3.	Моделирование, управление и мониторинг проектов.
3.	Цифровые платформы и технологии
3.1.	Инструменты цифровой трансформации. Организация пользовательских интерфейсов.

3.2.	Организация баз данных и баз знаний. Интеллектуальная поддержка принятия решений.
4.	Особенности цифровой трансформации здравоохранения
4.1	Основные проблемы и фронтирные задачи.
4.2	Тенденции по цифровизации здравоохранения в России и за рубежом.

РАЗДЕЛ 2 «Инженерия искусственного интеллекта в медицине»

Код	Наименования тем, элементов и подэлементов
1.	Искусственный интеллект в медицине
1.1.	Внедрение искусственного интеллекта в медицинскую сферу.
1.2.	Методы обучения ИИ на основе разнородной информации.
2.	Обучение с учителем и без учителя.
2.1.	Модель линейной регрессии. Модель логистической регрессии. Метод опорных векторов.
2.2.	Деревья принятия решений и случайный лес.
2.3.	Методы K-Means & PCA
3	Классификация снимков КТ с помощью сверточных нейронных сетей.
3.1.	Отладка модели сверточной нейронной сети.
3.2.	Классификация медицинских изображений сверточной нейронной сетью
4.	Построение представлений из многоканальных временных рядов ЭЭГ.
4.1	Классификация умственной нагрузки
4.2	Повышение точности классификации умственной нагрузки по ЭЭГ

РАЗДЕЛ 3 «Вычислительная анатомия на базе технологий AR/VR»

Код	Наименования тем, элементов и подэлементов
1.	Сферы применения 3D моделирования в медицине:
1.1.	Путь от 2D до 3D визуализации в медицине и анатомии. Статическое и динамическое моделирование в медицине.
1.2.	Оборудование и программное обеспечение, необходимое для 3D моделирования в медицине
1.3.	Применение медицинского 3D моделирования в научных исследованиях
2	Тема 2. Методы моделирования анатомических структур
2.1.	Сложность анатомического моделирования
2.2.	3D моделирование в анатомии и физиологии
2.3.	Обзор 3D анатомического атласа «Пирогов»
3	Тема 3. Хирургические тренажеры с использованием 3D моделирования:
3.1.	3D моделирование для тренажеров VR/AR
3.2.	3D моделирование для физических тренажеров
3.3.	3D модели для сцен в тренажерах для нейрореабилитации

РАЗДЕЛ 4 «Анализ и визуализация медицинских данных»

Код	Наименования тем, элементов и подэлементов
1.	Визуализация поля измерений. Статика и динамика измерений биопотенциального поля
1.1.	Системы отведения. Методы электрокардиографического картирования, применяемые в предрейсовых осмотрах
1.2.	Система управления данными и компонентами автоматизации OLE математических пакетов
1.3.	Моделирование поверхностного моментного распределения биопотенциалов. Расчет временной модели. Пространственная форма записи модели
2	Цифровые изображения, методы преобразований и способы анализа. Про-

	блемы медицинской визуализации
2.1.	Пространственные методы
2.2.	Частотные методы
2.2.	Восстановление изображений
3	Анализ больших данных
3.1.	Проблема интеграции информационных ресурсов.
3.2.	Особенности восприятия и интерактивная визуализация. Этапы обработки информации в приложениях

РАЗДЕЛ 5 «Телемедицина и системы поддержки принятия врачебных решений (СППВР)»

Код	Наименования тем, элементов и подэлементов
1.	Телемедицина
1.1.	Пропедевтика телемедицины.
1.2.	Клиническая телемедицина.
1.3.	Пациент-центрированная телемедицина
2	Информационные технологии принятия управленческого решения
2.1.	Уровни управления и типы ИС организации. Содержание и стадии процесса принятия решения. Процесс принятия решений. Задачи принятия решения. Технология принятия решения.
2.2.	Системы поддержки принятия решений. Разработка и принятие управленческого решения на основе информационно-коммуникационных технологий
3	Технология принятия решений на базе интеллектуальных информационных систем
3.1.	Структура систем принятия решений на базе ИИС. Характеристика. Классификация.
3.2.	Принципы построения систем принятия решений на базе ИИС. Подходы к реализации систем принятия решений на базе ИИС

РАЗДЕЛ 6 «Научно-исследовательская работа»

Код	Наименования тем, элементов и подэлементов
1	Работа в группе по проведению исследования в рамках индивидуального задания.
2	Участие в проведении эксперимента и написании научной статьи.

РАЗДЕЛ 7 «Проектно-технологическая практика»

Код	Наименования тем, элементов и подэлементов
1	Работа в группе по проведению исследования в рамках индивидуального задания.
2	Участие в реальном проекте по цифровой трансформации здравоохранения.

IV. УЧЕБНЫЙ ПЛАН ПРОГРАММЫ.

Цель: подготовка инженерных кадров, способных решать задачи разработки новой медицинской техники, средств реабилитации и диагностики, а также телемедицинских систем на основе комбинированного владения информационными технологиями и глубокого понимания особенностей назначения и применения инженерных решений в медицине.

Категория обучающихся: получившие высшее образование по очной (очно-заочной) форме, лица, освоившие основную образовательную программу бакалавриата, специалитета, а также магистратуры по инженерным специальностям.

Трудоемкость обучения: 256 академических часов

Форма обучения: очная.

№ п/п	Наименование раздела (модуля)	Всего часов	В том числе			Форма контроля
			Л	ПЗ	СР	
1.	Управление проектами по цифровой трансформации здравоохранения	36	8	10	18	тестовый контроль
2.	Инженерия искусственного интеллекта в медицине	36	6	12	18	тестовый контроль
3.	Вычислительная анатомия на базе технологий AR/VR	36	8	10	18	тестовый контроль
4.	Анализ и визуализация медицинских данных	36	6	12	18	тестовый контроль
5.	Телемедицина и СППВР	36	6	12	18	тестовый контроль
6.	Научно-исследовательская работа		-	-	36	практическое задание
7.	Проектно-технологическая практика		-	-	36	практическое задание
8.	Промежуточная аттестация	2				зачет
9.	Итоговая аттестация	2				демонстрационный экзамен
	Всего:	256				

* Л-лекции, ПЗ - практические занятия, СР – самостоятельная работа,

Все занятия проводятся в дистанционном формате: вебинар, он-лайн-чат.

ДОТ и ЭО осуществляются на платформе Электронно-информационной образовательной среды СамГМУ <https://samsmu.ru/edu/>.

V. КАЛЕНДАРНЫЙ УЧЕБНЫЙ ГРАФИК

Сроки обучения: образовательный процесс по программе может осуществляться в течение всего года.

Трудоёмкость освоения: 256(ак.ч.)

Режим занятий: 16 недель / 3 дня в неделю /6 ч. в день

№ пп	Наименование раздела(модуля)	Учебные недели															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1.	Управление проектами по цифровой трансформации здравоохранения		Л(4) ПЗ(2) СР(6)	Л(4) ПЗ(2) СР(6)	ПЗ(6) СР(6)												
2.	Инженерия искусственного интеллекта в медицине					Л(2) ПЗ(4) СР(6)	Л(2) ПЗ(4) СР(6)	Л(2) ПЗ(4) СР(6)									
3.	Вычислительная анатомия на базе технологий AR/VR		Л(4) ПЗ(2) СР(6)	Л(4) ПЗ(2) СР(6)	ПЗ(6) СР(6)												
4.	Анализ и визуализация медицинских данных					Л(2) ПЗ(4) СР(6)	Л(2) ПЗ(4) СР(6)	Л(2) ПЗ(4) СР(6)									
5.	Телемедицина и СППВР								Л(2) ПЗ(4) СР(6)	Л(2) ПЗ(4) СР(6)	Л(2) ПЗ(4) СР(6)						
6.	Научно-исследовательская работа	Постановка задачи (4)			НИР(4)			НИР(4)	НИР(8)	НИР(8)	НИР(8)						
7.	Проектно-технологическая практика											ПП(8)	ПП(8)	ПП(8)	ПП(6)	ПП(6)	
	Промежуточная аттестация				ПА			ПА			ПА					ПА	
	Итоговая аттестация																ДЭ

Условные обозначения

Л Лекция

Пз Практические занятия

СР Самостоятельная работа

ПП Проектно-технологическая практика

НИР Научно-исследовательская работа

ПА Промежуточная аттестация

Итоговая аттестация: ДЭ - Демонстрационный экзамен.

VI. ОРГАНИЗАЦИОННО-ПЕДАГОГИЧЕСКИЕ УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ

Материально-техническое обеспечение, необходимое для организации всех видов дисциплинарной подготовки:

- учебно-методическую документацию и материалы по всем разделам (модулям) программы, в том числе в ЭИОС университета;
- учебно-методическую литературу для внеаудиторной работы обучающихся;
- материально-технические базы, обеспечивающие организацию всех видов дисциплинарной подготовки:
- учебные аудитории, оснащенные оборудованием для проведения учебного процесса;
- дистанционные и электронные ресурсы для самостоятельной подготовки обучающихся, в частности Электронно-информационная образовательная среда СамГМУ <https://samsmu.ru/edu/>.
- кадровое обеспечение: реализация Программы обеспечивается научно-педагогическими кадрами Университета систематически занимающихся научной и научно-методической деятельностью со стажем работы в системе высшего и/или дополнительного профессионального образования не менее 5 лет, допустимо привлечение к образовательному процессу высококвалифицированных специалистов ИТ-сферы и/или дополнительного профессионального образования в части, касающейся профессиональных компетенций в области руководства проектированием специализированного программного обеспечения в рамках цифровой трансформации сферы здравоохранения, с обязательным участием представителей профильных организаций-работодателей. Возможно привлечение региональных руководителей цифровой трансформации (отраслевых ведомственных и/или корпоративных) к проведению итоговой аттестации, привлечение работников организаций реального сектора экономики субъектов Российской Федерации.

Учебно-методическое и информационное обучение:

Основная литература:

1. Шелудько В.М. Основы программирования на языке высокого уровня Python : учебное пособие / Шелудько В.М.. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2017. — 146 с.
2. Сузи Р.А. Язык программирования Python : учебное пособие / Сузи Р.А.. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 350 с.
3. Джонс М. Т. Программирование искусственного интеллекта в приложениях / М. Т. Джонс; пер. с англ. А. И. Осипов. - М. : ДМК Пресс, 2018. – 311с.
4. Спиридонов И. Н. Методы и алгоритмы вычислительной диагностики: учебное пособие / И.Н. Спиридонов, А.В. Самородов ; Московский государственный технический университет им. Н.Э. Баумана. – Москва : МГТУ им. Н.Э. Баумана, 2006. – 50 с.
5. Немирко А. П. Математический анализ биомедицинских сигналов и данных / А.П. Немирко, Л.А. Манило, А.Н. Калиниченко. – Москва : Физматлит, 2017. – 248 с.
6. Кобринский, Б. А. Телемедицина в системе практического здравоохранения: практическое пособие / Б.А. Кобринский. – 2-е изд., стер. – Москва ; Берлин : Директ-Медиа, 2016.
7. Интеллектуальный анализ данных в системе Loginom : учеб.-метод. пособие / О. Ю. Кузнецова, В. И. Горбаченко. – Пенза : Изд-во ПГУ, 2020. – 60 с.

Дополнительная литература:

1. Трухачева Н.В. Цифровая медицина : учебное пособие / Трухачева Н.В., Пупырев Н.П.. – Москва : Ай Пи Ар Медиа, 2022. – 169 с.

2. Шелудько В.М. Язык программирования высокого уровня Python. Функции, структуры данных, дополнительные модули : учебное пособие / Шелудько В.М.. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2017. — 107 с.
3. Буйначев С.К. Основы программирования на языке Python : учебное пособие / Буйначев С.К., Боклаг Н.Ю.. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2014. — 92 с.
4. Кобринский, Б. А. Интеллектуальные и информационные системы в медицине: мониторинг и поддержка принятия решений : сборник научных трудов / Б. А. Кобринский, и др. – Москва ; Берлин : Директ-Медиа, 2016. – 529с.
5. Вычислительная анатомия - Computational anatomy. Электронный ресурс Режим доступа https://wiki5.ru/wiki/Computational_anatomy

Программное обеспечение:

При проведении занятий по дисциплине используется следующее общесистемное и прикладное программное обеспечение:

1. программные средства общего назначения: Microsoft Window, Microsoft Office;
2. программный комплекс PyCharm Community Edition;
3. мобильный 3D-атлас интерактивный стол «Пирогов»;
4. платформа разработки RT3D-приложений Unity Pro;
5. программный комплекс 3ds-MAX.
6. российская интегрированная платформа для реализации аналитических процессов Loginom Community Edition 6.3.0

Базы данных, информационно-справочные системы:

1. <http://www.4medic.ru/> - информационный портал для врачей и студентов
2. <http://www.sportmedicine.ru> - электронные медицинские книги
3. www.pubmed.com - электронная база данных медицинских и биологических публикаций.
4. <http://elibrary.ru/defaultx.asp> - научная электронная библиотека.
5. <http://www.infostat.ru/> - электронные версии статистических публикаций.
6. <http://diss.rsl.ru/> - электронная библиотека диссертаций РГБ.
7. <http://www.consultant.ru> Компьютерная справочная правовая система. «КонсультантПлюс» [Электронный ресурс].

VII.ТРЕБОВАНИЯ К ИТОГОВОЙ АТТЕСТАЦИИ

Итоговая аттестация проходит в форме демонстрационного экзамена. Вначале обучающиеся получают задание разработать проект цифрового инженерного решения для одной из актуальных задач здравоохранения (ориентировочные темы представлены ниже).

Оформление проекта для программных решений производится на языке UML или BPMN, для программно-аппаратных решений возможен выбор другой методологии.

Необходимо построить модели «Как есть» и «Как должно быть» с обоснованием преимуществ цифровой трансформации процессов. Защита проекта производится путем демонстрации моделей в формате презентации.

Прием демонстрационного экзамена производится комиссией, оценивающей владение методологией проектирования цифровых инженерных решений и умение выполнять процедуры сборки программных модулей и компонент в программный продукт и выявлять соответствие требований заказчиков с существующими продуктами в области здравоохранения.

Ориентировочные темы учебных проектов для демонстрационного экзамена:

- 1) Обработка рентгеновских изображений с использованием технологий глубокого обучения.

- 2) Интеллектуальный анализ результатов скринингового обследования.
- 3) Анализ результатов телемедицинского обследования.
- 4) Система сбора и обработки данных ежегодного профилактического медицинского осмотра.
- 5) Распознавание опухолей на МРТ изображениях.
- 6) Интеллектуальная обработка данных офтальмологического исследования.
- 7) Система сбора и обработки результатов электрокардиографии.
- 8) Разработка и реализация распределенной архитектуры телемедицинского Интернета вещей.

Лицам, успешно освоившим соответствующую ДПП ПП (в области руководства проектированием специализированного программного обеспечения в рамках цифровой трансформации сферы здравоохранения) и прошедшим итоговую аттестацию, выдается документ о квалификации: диплом о профессиональной переподготовке.

Лицам, не прошедшим итоговую аттестацию или получившим на итоговой аттестации неудовлетворительные результаты, а также лицам, освоившим часть Программы, выдается справка об обучении или о периоде обучения по образцу, самостоятельно устанавливаемому Университетом.

VIII. ОЦЕНОЧНЫЕ СРЕДСТВА

Контроль знаний, полученных слушателями при освоении разделов (модулей) Программы, осуществляется в следующих формах:

- текущий контроль успеваемости – обеспечивает оценивание хода освоения разделов Программы, проводится в форме тестового контроля и практических заданий;
- промежуточная аттестация – завершает изучение отдельного модуля Программы, проводится в форме зачета;
- итоговая аттестация – завершает изучение всей программы.

В ходе освоения Программы каждый слушатель выполняет следующие отчетные работы:

№ п/п	Наименование раздела (модуля)	Задание	Критерии оценки
1.	Управление проектами по цифровой трансформации здравоохранения	Тестовые задания	«зачтено» выставляется обучающемуся, если количество правильных ответов на тестовые вопросы – более 60 % от общего объема заданных вопросов; «незачтено» выставляется обучающемуся, если количество правильных ответов – менее 60 % от общего объема заданных вопросов
2	Инженерия искусственного интеллекта в медицине	Практические задания	«зачтено» – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов. «не зачтено» – ставится за работу, если правильно выполнено менее 2/3 всей работы.
3	Вычислительная анатомия на	Тестовые задания	« зачтено » выставляется обучающе-

	базе технологий AR/VR		муся, если количество правильных ответов на тестовые вопросы – более 60 % от общего объема заданных вопросов; «незачтено» выставляется обучающемуся, если количество правильных ответов – менее 60 % от общего объема заданных вопросов
4	Анализ и визуализация медицинских данных	Практические задания	«зачтено» – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов. «не зачтено» – ставится за работу, если правильно выполнено менее 2/3 всей работы.
5	Телемедицина и СППВР	Практические задания	«зачтено» – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов. «не зачтено» – ставится за работу, если правильно выполнено менее 2/3 всей работы.
6	Научно-исследовательская работа	Работа в группе по проведению исследования в рамках индивидуального задания. Участие в проведении эксперимента и написании научной статьи.	«зачтено» – ставится за работу, если обучающийся разработал алгоритм, реализовал его и продемонстрировал работоспособную версию программы «не зачтено» – ставится, если обучающийся не выполнил программу стажировки.
7	Проектно-технологическая практика	Работа в группе по проведению исследования в рамках индивидуального задания. Участие в реальном проекте по цифровой трансформации здравоохранения.	«зачтено» – ставится за работу, если обучающийся разработал алгоритм, реализовал его и продемонстрировал работоспособную версию программы «не зачтено» – ставится, если обучающийся не выполнил программу стажировки.
	Итоговая аттестация	Разработка учебного проекта для решения профессиональной задачи с последующей реализацией и демонстрацией	– оценка «отлично» выставляется обучающемуся, если диапазон набранных им баллов 70 – 100; – оценка «хорошо» выставляется обучающемуся, если диапазон набранных им баллов 40 – 69; – оценка «удовлетворительно» выставляется обучающемуся, если диапазон набранных им баллов 20 – 39; – оценка «неудовлетворительно» выставляется обучающемуся, если диапазон набранных им баллов 0 – 19.

Примеры оценочных средств:

Текущий контроль.

1.1. Модуль «Управление проектами по цифровой трансформации здравоохранения»

Примеры тестовых заданий

1. Задание

_____ - технология представляет собой методологию проектирования, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, производить ее анализ на всех этапах разработки и сопровождения информационных систем и разрабатывать приложения в соответствии с информационными требованиями пользователей.

Правильные варианты ответа: CASE

2. Задание

Какая модель предлагает каждую итерацию начинать с выделения целей и планирования очередной итерации, определения основных альтернатив и ограничений при ее выполнении, их оценки, а также оценки возникающих рисков и определения способов избавления от них, а заканчивать итерацию оценкой результатов проведенных в ее рамках работ

Правильные варианты ответа: спиральная

3. Задание

Какой стандарт описывает основные положения методологии SADT?

- ☒ IDEF 0
- ☐ IDEF 1
- ☐ IDEF 2
- ☐ IDEF 3

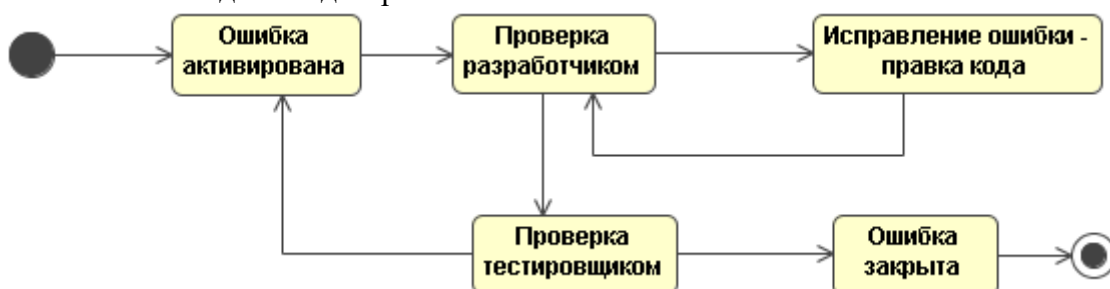
4. Задание

Какие диаграммы поведения входят в UML?

- ☐ вариантов использования
- ☐ классов
- ☐ рисков
- ☒ состояний
- ☒ деятельности
- ☒ последовательности
- ☐ цепочек процесса
- ☐ стоимостного анализа
- ☒ кооперации
- ☐ компонентов
- ☐ развертывания
- ☐ программных продуктов

5. Задание

Как называется данная диаграмма?



- ☐ классов

- ☒ состояний
- ☐ деятельности
- ☐ последовательности
- ☐ кооперации

1.2. Модуль «Инженерия искусственного интеллекта в медицине»

Примеры типовых заданий

Задание 1. Отладка модели линейной регрессии

Проанализировать параметры, загруженного набора данных diabetes:

<https://archive.ics.uci.edu/ml/datasets/diabetes>

Редактируя и запуская на исполнение ячейки в Jupiter Notebook:

<https://colab.research.google.com/drive/1IkaJ1oZFztAuIXnh1Vmxod0SkIHR15sU?usp=sharing>,

подобрать параметры линейной модели и набор исходных признаков так, что значение метрики R_2 повысилось на 0.25.

Интерпретировать полученные результаты.

Задание 2. Загрузка набора данных и построение модели логистической регрессии.

Запустить код, предложенный в Jupiter Notebook

<https://colab.research.google.com/drive/1n1QeaiGatd3Y3VWs0avxnXg9uZO68I32?usp=sharing>

на реальном наборе данных из UCI

<https://archive.ics.uci.edu/ml/index.php>,

выбрав в качестве целевого вектора - бинарный признак

Задание 3. Классификация медицинских изображений сверточной нейронной сетью

Загрузить изображения альтернативного набора данных, но идентичного формата в Jupyter Notebook:

<https://colab.research.google.com/drive/1ezLyEhg-hISuLfjFk96pB-ftOwlbNLL?usp=sharing> ;

Построить и отладить модель сверточной нейронной сети.

Интерпретировать полученный результат.

Задание 4. Повышение точности классификации умственной нагрузки по ЭЭГ

Повысить точность классификации умственной нагрузки, изменяя значения параметров модели сверточной нейронной сети:

https://colab.research.google.com/drive/1NzggkDcT1vnNYhg3_lJh0_lFkdEF7uSFp?usp=sharing

Интерпретировать полученный результат.

1.3. Модуль «Вычислительная анатомия на базе технологий AR/VR»

Примеры тестовых заданий

1) К какой группе устройств персонального компьютера относятся устройства виртуальной реальности?

- ☒ **внешние устройства**
- ☐ устройства управления
- ☐ основные устройства
- ☐ телекоммуникационные устройства

2) К устройствам виртуальной реальности относится:

- ☒ **треккер**
- ☐ тачнад
- ☐ трекбол
- ☐ дигитайзер

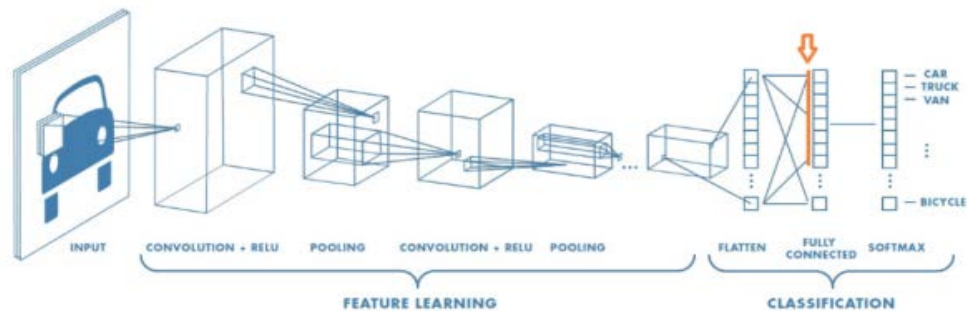
- 3) Самым распространенным устройством виртуальной реальности является:
- ☐ жилетка
 - ☒ шлем
 - ☐ графический акселератор
 - ☐ перчатки
- 4) Максимальное количество датчиков у перчаток, доступных в России:
- ☐ 11
 - ☐ 14
 - ☒ 16
 - ☐ 20
- 5) Минимальная частота пересчета изображения в графических акселераторах:
- ☒ 20 кадров
 - ☐ 15 кадров
 - ☐ 10 кадров
 - ☐ 5 кадров

1.4. Модуль «Анализ и визуализация медицинских данных»

Примеры практических заданий

Задание 1. Сформировать векторное представление изображения.

Пример выполнения:



```
!pip install
git+https://github.com/jaredwinick/img2vec-keras.git

import gdown # модуль для загрузки с гугл-диска
url =
'https://drive.google.com/uc?id=1Madin3qXe3Xexox5qOhASGu1FR2wvsFz' # адрес
output =
'101_ObjectCategories.tar.gz' # название загружаемого файла
gdown.download(url, output,
quiet=False)
!echo "Downloading
101_ObjectCategories for image notebooks"
# распаковываем
!tar -xzf
101_ObjectCategories.tar.gz
# удаляем после распаковки
!rm 101_ObjectCategories.tar.gz
!ls
root = '101_ObjectCategories'

Downloading...
From: https://drive.google.com/uc?id=1Madin3qXe3Xexox5qOhASGu1FR2wvsFz
To: /content/101_ObjectCategories.tar.gz
100%|██████████| 132M/132M [00:01<00:00, 66.1MB/s]
Downloading 101_ObjectCategories for image notebooks
```

```

101_ObjectCategories sample_data

from img2vec_keras import Img2Vec # подключаем библиотеку для векторных представлений

from IPython.display import Image

import glob # для работы с файлами

import numpy as np
from sklearn.metrics.pairwise import cosine_similarity # похожесть по косинусному рас-
СТОЯНИЮ.
from sklearn.decomposition import PCA # метод главных компонент
from sklearn.manifold import TSNE # метод TSNE
from sklearn.preprocessing import StandardScaler # масштабирование, убирает среднее,
единичный разброс.

import matplotlib.pyplot as plt
from matplotlib.offsetbox import OffsetImage, AnnotationBbox

import cv2

img2vec = Img2Vec()

Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5
102973440/102967424 [=====] - 0s 0us/step
102981632/102967424 [=====] - 0s 0us/step

image_paths = []
image_classes = ['pizza', 'schooner', 'scissors', 'panda', 'brain'] # выберем классы
для отображения, проверьте с другими

# считываем пути к изображениям этих классов
for image_class in image_classes:
    image_paths.extend(glob.glob('101_ObjectCategories/' + image_class + '/*.jpg')) #
ТОЛЬКО .jpg

len(image_paths)
291

# создаем вектора для этих изображений
image_vectors = {}
for image_path in image_paths: # для каждого файла
    vector =
    img2vec.get_vec(image_path)
    image_vectors[image_path] = vector

X =
np.stack(list(image_vectors.values()))

pca_50 = PCA(n_components=50)
pca_result_50 =
pca_50.fit_transform(X)
print('Cumulative explained variation for 50 principal components:
{}'.format(np.sum(pca_50.explained_variance_ratio_)))
print(np.shape(pca_result_50))

tsne = TSNE(n_components=2, verbose=1, n_iter=3000)
tsne_result =
tsne.fit_transform(pca_result_50)
Cumulative explained variation for 50 principal components: 0.7652437686920166
[291, 50]
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 291 samples in 0.000s...
[t-SNE] Computed neighbors for 291 samples in 0.007s...
[t-SNE] Computed conditional probabilities for sample 291 / 291
[t-SNE] Mean sigma: 12.785958
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:783: FutureWarning: The
default initialization in TSNE will change from 'random' to 'pca' in 1.2.
FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793: FutureWarning: The
default learning rate in TSNE will change from 200.0 to 'auto' in 1.2.
FutureWarning,
[t-SNE] KL divergence after 250 iterations with early exaggeration: 58.010796
[t-SNE] KL divergence after 1650 iterations: 0.356967

tsne_result_scaled =
StandardScaler().fit_transform(tsne_result)

```

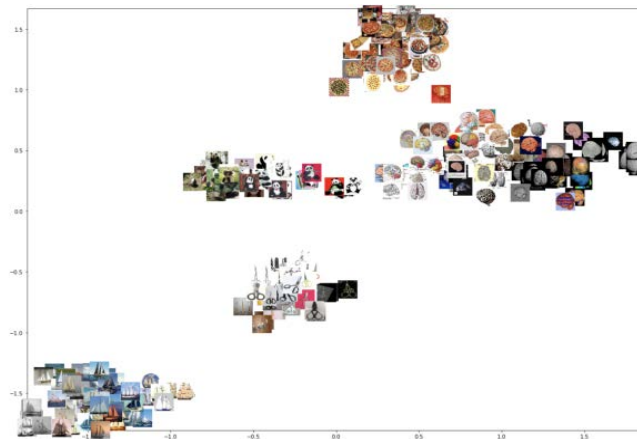
```

images = []
for image_path in image_paths:
    image = cv2.imread(image_path, 3) # загружаем изображение
    b,g,r = cv2.split(image)           # разделяем каналы b, g, r
    image = cv2.merge([r,g,b])         # собираем в r, g, b
    image = cv2.resize(image, (50,50)) # масштаб
    images.append(image)

fig, ax =
    plt.subplots(figsize=(20,15))
artists = []

for xy, i in zip(tsne_result_scaled, images): # перебираем TSNE вектора (это двумерные
координаты) и изображения
    x0, y0 = xy # разделяем горизонтальную и вертикальную координату
    img = OffsetImage(i, zoom=.7) # задаем графический объект с картинкой
    ab = AnnotationBbox(img, (x0, y0), xycoords='data', frameon=False) # вставляем эту
картинку по координатам из вектора TSNE
    artists.append(ax.add_artist(ab)) # добавляем на график
ax.update_datalim(tsne_result_scaled) # пределы отображения
ax.autoscale(enable=True, axis='both', tight=True) # автомасштаб
plt.show()

```



```

image_path1='101_ObjectCategories/Leopards/image_0001.jpg'

vector1 =
np.reshape(img2vec.get_vec(image_path1),(1,-1))

# путь ко второму изображению
image_path2='101_ObjectCategories/Motorbikes/image_0001.jpg'

vector2 =
np.reshape(img2vec.get_vec(image_path2),(1,-1))

similarity_matrix =
    cosine_similarity(vector1, vector2)

print('для разных ',
similarity_matrix[0][0])

# путь к третьему изображению
image_path3='101_ObjectCategories/Motorbikes/image_0005.jpg'

vector3 =
np.reshape(img2vec.get_vec(image_path3),(1,-1))

similarity_matrix =
    cosine_similarity(vector3, vector2)

print('для похожих ',
similarity_matrix[0][0])

для разных  0.25445843
для похожих  0.901036

```

Задание 2. Выполнить перенос стиля изображения. Обосновать необходимость решения задачи в медицинском контексте. Для скорости рекомендуется сменить среду выполнения на GPU: Меню - Среда выполнения - Сменить среду выполнения - Аппаратный ускоритель - GPU.

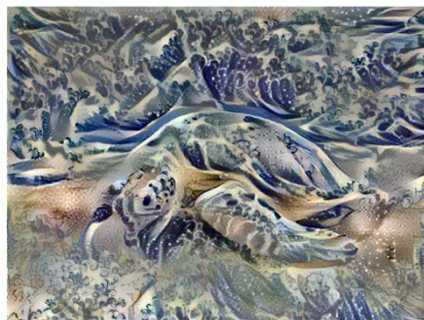
Исходное изображения:



Стиль :



Результат:



Пример выполнения:

```
import os
img_dir = './img/'
if not os.path.exists(img_dir):
    os.makedirs(img_dir)
!wget --quiet -P ./img/ https://upload.wikimedia.org/wikipedia/commons/d/d7/Green_Sea_Turtle_grazing_seagrass.jpg
!wget --quiet -P ./img/ https://upload.wikimedia.org/wikipedia/commons/0/0a/The_Great_Wave_off_Kanagawa.jpg
!wget --quiet -P ./img/ https://upload.wikimedia.org/wikipedia/commons/b/b4/Vassily_Kandinsky%2C_1913_-_Composition_7.jpg
!wget --quiet -P ./img/ https://upload.wikimedia.org/wikipedia/commons/0/00/Tuebingen_Neckarfront.jpg
!wget --quiet -P ./img/ https://upload.wikimedia.org/wikipedia/commons/6/68/Pillars_of_creation_2014_HST_WFC3-UVIS_full-res_denoised.jpg
!wget --quiet -P ./img/ https://upload.wikimedia.org/wikipedia/commons/thumb/e/ea/Van_Gogh_-_Starry_Night_-_Google_Art_Project.jpg/1024px-Van_Gogh_-_Starry_Night_-_Google_Art_Project.jpg
```

подключение библиотек

```

import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['figure.figsize'] =
    (10,10)
mpl.rcParams['axes.grid'] = False

import numpy as np
from PIL import Image
import time
import functools

import tensorflow as tf

from tensorflow.keras.preprocessing import image as kp_image
from tensorflow.keras import models
from tensorflow.keras import losses
from tensorflow.keras import layers
from tensorflow.keras import backend as K

# содержание
content_path =
    './img/Green_Sea_Turtle_grazing_seagrass.jpg'
# стиль
style_path =
    './img/The_Great_Wave_off_Kanagawa.jpg'

# чтение с диска и преобразование изображения
def load_img(path_to_img):
    max_dim = 512 # максимальная размерность
    img = Image.open(path_to_img) # читаем с диска
    long = max(img.size) # находим максимальную из размерностей
    scale = max_dim/long # масштабирующий коэффициент
    img =
img.resize((round(img.size[0]*scale), round(img.size[1]*scale)), Image.ANTIALIAS) #
масштабируем

    img = kp_image.img_to_array(img) # переводим в массив numpy

    # первая размерность - пакеты (батчи). Надо добавить ее.
    img = np.expand_dims(img, axis=0)
    return img

# функция для рисования
def imshow(img, title=None):
    # для рисования размерность батчей не нужна, уберем ее
    out = np.squeeze(img, axis=0)
    # переводим в тип uint8
    out = out.astype('uint8')
    plt.imshow(out) # рисуем
    if title is not None:
        plt.title(title) # заголовок, если есть
    plt.imshow(out)

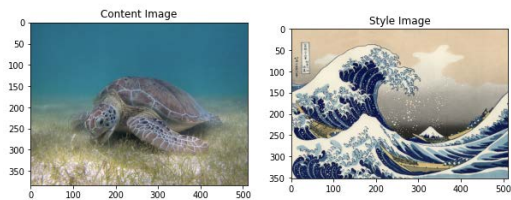
plt.figure(figsize=(10,10))

content =
load_img(content_path).astype('uint8')
style =
load_img(style_path).astype('uint8')

plt.subplot(1, 2, 1)
imshow(content, 'Content Image')

plt.subplot(1, 2, 2)
imshow(style, 'Style Image')
plt.show()

```



Воспользуемся обученной нейронной сетью для классификации изображений VGG-19. Для нее надо дополнительно предобработать изображения, например вычесть "среднее" `mean = [103.939, 116.779, 123.68]` и перевести каналы из RGB в BGR. Все это сделает функция `tf.keras.applications.vgg19.preprocess_input`

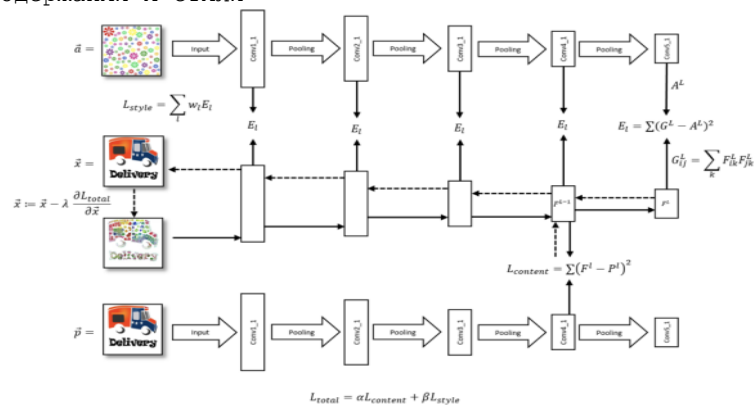
```
def load_and_process_img(path_to_img):
    img = load_img(path_to_img) # загружаем изображение
    img =
    tf.keras.applications.vgg19.preprocess_input(img) # обрабатываем его для VGG
    return img

# обратный препроцессинг
def deprocess_img(processed_img):
    x = processed_img.copy() # копия изображения
    if len(x.shape) == 4: # если 4 канала
        x = np.squeeze(x, 0) # отрезаем первое измерение (батчи)
    # если 3, то ничего не делаем
    assert len(x.shape) == 3, ("Input to deprocess image must be an image of "
                                "dimension [1, height, width, channel] or [height, width, "
                                "channel]")
    if len(x.shape) != 3:
        raise ValueError("Invalid input to deprocessing image")

    # добавляем "среднее", которое раньше вычитали
    x[:, :, 0] += 103.939
    x[:, :, 1] += 116.779
    x[:, :, 2] += 123.68
    x = x[:, :, ::-1] # меняем для каналов цвета (3 измерение) порядок обратно на RGB

    x = np.clip(x, 0, 255).
    astype('uint8') # обрезаем до диапазона 0...255 и переводим в uint8
    return x

# представление содержания и стиля
```



```
# Название слоев из нейронной сети которые используем для признаков содержания
content_layers = ['block5_conv2']
```

```
# название слоев из нейронной сети которые используем для признаков стиля
style_layers = ['block1_conv1',
                'block2_conv1',
                'block3_conv1',
                'block4_conv1',
                'block5_conv1'
                ]
```

```
num_content_layers =
    len(content_layers)
```

```

num_style_layers = len(style_layers)

# загрузка модели нейронной сети
def get_model():

    # загружаем сеть VGG, обученную на данных imagenet
    vgg =
tf.keras.applications.vgg19.VGG19(include_top=False, weights='imagenet')
    vgg.trainable = False # выключаем обучение весов этой сети, мы обучаем только входы
    в нее.
    # возвращаем выходы слоев признаков
    style_outputs =
[vgg.get_layer(name).output for name in style_layers] # стиля
    content_outputs =
[vgg.get_layer(name).output for name in content_layers] # содержания
    model_outputs = style_outputs +
content_outputs
    # построить модель
    return models.Model(vgg.input, model_outputs)

def get_content_loss(base_content, target):
    #base_content -
    признаки изображения-содержания
    #target -
    признаки генерируемого изображения
    return tf.reduce_mean
(tf.square(base_content - target))

# расчет матрицы Грамма
def gram_matrix(input_tensor):
    channels =
int(input_tensor.shape[-1]) # количество каналов = фильтров
    a = tf.reshape(input_tensor, [-1, channels]) # многомерный массив переформатируем в
двумерный, второе измерение - каналы
    n = tf.shape(a)[0] # число признаков каждого канала
    gram = tf.matmul(a, a, transpose_a=True) # матричное произведение
    return gram / tf.cast(n, tf.float32) # делим на количество признаков

# ошибка стиля
def get_style_loss(base_style, gram_target):
    """принимает изображение-стиль размером h - высота, w - ширина, c - каналы
    и матрицу Грамма для генерируемого изображения
    """

    # We scale the loss at a given layer by the size of the feature map and the number
of filters
    height, width, channels =
base_style.get_shape().as_list() # высота, ширина, количество каналов в виде списка
    gram_style =
gram_matrix(base_style) # считаем матрицу грамма изображения-стиля
    return tf.reduce_mean(tf.square(gram_style - gram_target))# делим на нормирующий ко-
эффициент / (4. * (channels ** 2) * (width * height) ** 2)

# перенос стиля
def get_feature_representations(model, content_path, style_path):
    """
    Функция загрузки и расчета признаков изображений стиля и содержания
    Аргументы:
        model: модель нейронной сети.
        content_path: путь к изображению-содержанию.
        style_path: путь к изображению-стилю

    Возвращает:
        признаки содержания и стиля.
    """
    # загружаем изображения
    content_image =
load_and_process_img(content_path) # содержание
    style_image =
load_and_process_img(style_path) # стиль

```

```

# подаем их на нейронную сеть и возвращаем признаки
style_outputs =
model(style_image)# стиль
content_outputs =
model(content_image) # содержание

# собираем признаки тех слоев, что выбрали для стиля и содержания
style_features = [style_layer[0] for style_layer in
style_outputs[:num_style_layers]]
content_features =
[content_layer[0] for content_layer in content_outputs[num_style_layers:]]
return style_features, content_features

# расчет функции ошибки и градиента
def compute_loss(model, loss_weights, init_image, gram_style_features, con-
tent_features):
    style_weight, content_weight =
    loss_weights # веса ошибки стиля и содержания

    # подаем начальное изображение в нейронную сеть, возвращаем признаки для него.
    model_outputs = model(init_image)
    # из них к стилю относятся первые num_style_layers выходов (величина определена ра-
нее)
    style_output_features =
    model_outputs[:num_style_layers]
    # остальные выходы относятся к содержанию
    content_output_features =
    model_outputs[num_style_layers:]

    style_score = 0 # ошибка стиля
    content_score = 0 # ошибка содержания
    weight_per_style_layer = 1.0 /
    float(num_style_layers) # веса для разных слоев слагаемых ошибки стиля, равные.
    # перебираем все выбранные слои для стиля
    for target_style, comb_style in zip(gram_style_features, style_output_features):
        # target_style - значения матрицы Грамма выбранных слоев признаков для изображе-
ния-стиля
        # comb_style - посчитанные выше признаки выбранных слоев обучаемого изображения
        # считаем ошибку стиля со всех выбранных слоев с весами.
        style_score +=
        weight_per_style_layer *
        get_style_loss(comb_style[0], target_style)
        weight_per_content_layer = 1.0 / float(num_content_layers) # веса для разных слоев
слагаемых ошибки содержания, равные.
        # перебираем все выбранные слои для содержания
        for target_content, comb_content in zip(content_features, content_output_features):
            # target_content - значения выбранных слоев признаков для изображения-содержания
            # comb_content - посчитанные выше признаки выбранных слоев обучаемого изображения
            # считаем ошибку содержания со всех выбранных слоев с весами.
            content_score +=
            weight_per_content_layer * get_content_loss(comb_content[0], target_content)

    style_score *= style_weight # умножаем на вес ошибку стиля
    content_score *= content_weight # умножаем на вес ошибку содержания

    # общая ошибка
    loss = style_score + content_score
    return loss, style_score, content_score

def compute_loss(model, loss_weights, init_image, gram_style_features, con-
tent_features):
    """Расчет функции ошибки.

    Аргументы:
    model: Модель нейронной сети
    loss_weights: Веса стиля и содержания в общей ошибке.

    init_image: Начальное изображение, с которого начнется обучение, по его пикселям
мы и считаем градиенты.
    gram_style_features: уже посчитанная матрица Грамма для изображения-стиля.

```

```

    content_features: уже посчитанные признаки для изображения-содержания.

Возвращает:
    Общую ошибку, ошибку стиля, ошибку содержания
"""
style_weight, content_weight = loss_weights # веса ошибки стиля и содержания

# подаем начальное изображение в нейронную сеть, возвращаем признаки для него.
model_outputs = model(init_image)
# из них к стилю относятся первые num_style_layers выходов (величина определена ранее)
style_output_features = model_outputs[:num_style_layers]
# остальные выходы относятся к содержанию
content_output_features =
model_outputs[num_style_layers:]

style_score = 0 # ошибка стиля
content_score = 0 # ошибка содержания

weight_per_style_layer = 1.0 /
float(num_style_layers) # веса для разных слоев слагаемых ошибки стиля, равные.
# перебираем все выбранные слои для стиля
for target_style, comb_style in zip(gram_style_features, style_output_features):
    # target_style - значения матрицы Грамма выбранных слоев признаков для изображения-стиля
    # comb_style - посчитанные выше признаки выбранных слоев обучаемого изображения
    # считаем ошибку стиля со всех выбранных слоев с весами.
    style_score +=
weight_per_style_layer *
get_style_loss(comb_style[0], target_style)

weight_per_content_layer = 1.0 /
float(num_content_layers) # веса для разных слоев слагаемых ошибки содержания, равные.
# перебираем все выбранные слои для содержания
for target_content, comb_content in zip(content_features, content_output_features):
    # target_content - значения выбранных слоев признаков для изображения-содержания
    # comb_content - посчитанные выше признаки выбранных слоев обучаемого изображения
    # считаем ошибку содержания со всех выбранных слоев с весами.
    content_score +=
weight_per_content_layer* get_content_loss(comb_content[0], target_content)

style_score *= style_weight # умножаем на вес ошибку стиля
content_score *= content_weight # умножаем на вес ошибку содержания

# общая ошибка
loss = style_score + content_score
return loss, style_score, content_score

# цикл обучения
import IPython.display
# цикл обучения
def run_style_transfer(content_path, # путь к изображению-содержанию
                      style_path, # путь к изображению-стилю
                      num_iterations=1000, # число итераций обучения
                      content_weight=1e3, # вес содержания
                      style_weight=1e-2): # вес стиля

    model = get_model() # получаем модель нейронной сети
    for layer in model.layers: # нам не надо обучать веса самой модели,
        layer.trainable = False # ставим для всех слоев запрет на обучение

    # получаем признаки для изображений стиля и содержания (конкретные слои мы выбрали ранее)
    style_features, content_features = get_feature_representations(model, content_path,
                                                                    style_path)
    # рассчитываем матрицы Грамма для изображения-стиля
    gram_style_features = [gram_matrix(style_feature) for style_feature in
                           style_features]

```

```

# Задаем начальное изображение.
init_image = load_and_process_img(content_path) # Здесь используем изображение-
содержание, но вы можете и другие использовать, хоть белое полотно.
init_image = tf.Variable(init_image, dtype=tf.float32) # переводим в тип Variable
для которого работает автодифференцирование

# Параметры обучения
opt=tf.optimizers.Adam(learning_rate=5, beta_1=0.99, epsilon=1e-1)

# счетчик итераций
iter_count = 1

# Место для запоминания лучшего сгенерированного изображения и ошибки на нем
best_loss, best_img = float('inf'), None

loss_weights = (style_weight, content_weight)# веса ошибок стиля и содержания как
список
# объект (словарь) содержащий необходимые тензоры и объекты
cfg = {
    'model': model, # модель нейронной сети
    'loss_weights': loss_weights, # веса ошибок
    'init_image': init_image, # начальное изображение
    'gram_style_features': gram_style_features, # матрицы Грамма изображения-стиля
    'content_features': content_features # признаки изображения-содержания
}

# Для рисования
num_rows = 2 # строки
num_cols = 5 # столбцы
display_interval = num_iterations/(num_rows*num_cols) # как часто рисовать
start_time = time.time() # запуск таймера
global_start = time.time() #

norm_means = np.array([103.939, 116.779, 123.68]) # "среднее" изображение
# диапазон чисел для изображения
min_vals = -norm_means
max_vals = 255 - norm_means

imgs = []
# Обучение
for i in range(num_iterations): # в цикле по количеству итераций
    grads, all_loss = compute_grads(cfg) # вычисляем ошибки и градиент
    loss, style_score, content_score = all_loss
    opt.apply_gradients([(grads, init_image)]) # применяем вычисленный градиент в ме-
тоде обучения, изменяем init_image
    clipped = tf.clip_by_value(init_image, min_vals, max_vals) # обрезаем до нужного
диапазона
    init_image.assign(clipped)
    end_time = time.time() # время итерации

    if loss < best_loss: # если ошибки меньше предыдущей лучшей
        # обновляем лучшие
        best_loss = loss # ошибку
        best_img = deprocess_img(init_image.numpy()) # и изображение, переведа в исход-
ный вид

    if i % display_interval== 0: # каждые несколько итераций
        start_time = time.time() # обновляем таймер

        #
        plot_img = init_image.numpy() # переводим текущее изображение из тензора Keras в
numpy
        plot_img = deprocess_img(plot_img) # возвращаем его в исходный вид
        imgs.append(plot_img) # добавляем в массив для рисования
        IPython.display.clear_output(wait=True) # очищаем выход текущей ячейки
        IPython.display.display_png(Image.fromarray(plot_img)) # рисуем текущее изобра-
жение
        # выводим информацию о процессе обучения
        print('Iteration: {}'.format(i))

```

```

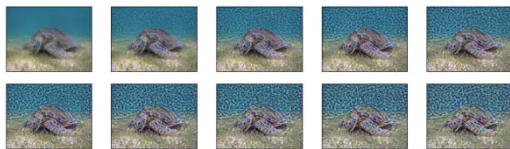
        print('Total loss: {:.4e}, '
              'style loss: {:.4e}, '
              'content loss: {:.4e}, '
              'time: {:.4f}s'.format(loss, style_score, content_score, time.time() -
start_time))

# По завершению обучения
print('Total time: {:.4f}s'.format(time.time() - global_start))
IPython.display.clear_output(wait=True)
plt.figure(figsize=(14,4))
# рисуем все сохраненные изображения
for i,img in enumerate(imgs):
    plt.subplot(num_rows,num_cols,i+1)
    plt.imshow(img)
    plt.xticks([])
    plt.yticks([])

    return best_img, best_loss # возвращаем лучшее изображение и его ошибку

проверка работы
best, best_loss =
run_style_transfer(content_path,
num_iterations=100)
style_path,

```



```

img=Image.fromarray(best)
img

```



```

from google.colab import files
img.save('Wave_turtle.png')
files.download('Wave_turtle.png')

```

```

def show_results(best_img,
content_path, style_path,
show_large_final=True):
    plt.figure(figsize=(10, 5))
    content = load_img(content_path)
    style = load_img(style_path)

    plt.subplot(1, 2, 1)
    imshow(content, 'Content Image')

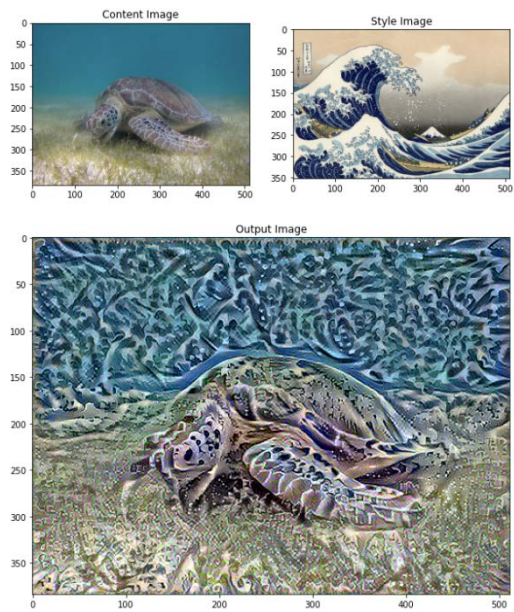
    plt.subplot(1, 2, 2)
    imshow(style, 'Style Image')

    if show_large_final:
        plt.figure(figsize=(10, 10))

        plt.imshow(best_img)
        plt.title('Output Image')
        plt.show()

show_results(best, content_path, style_path)

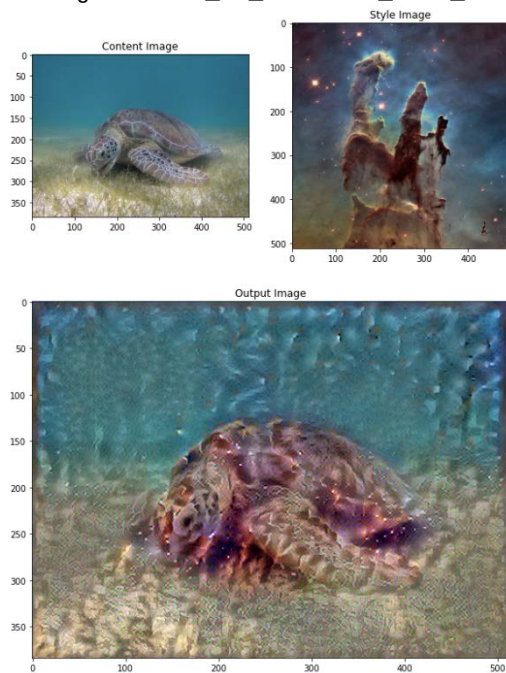
```



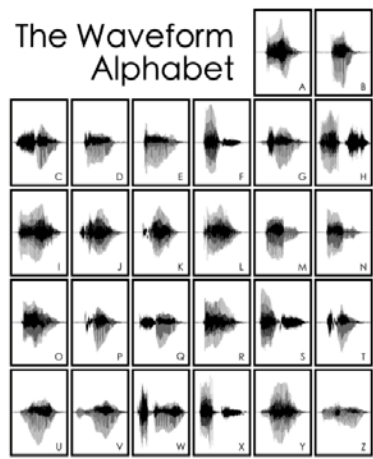
```
best_poc_turtle, best_loss =
run_style_transfer('./img/Green_Sea_Turtle_grazing_seagrass.jpg',
'./img/Pillars_of_creation_2014_HST_WFC3-UVIS_full-
res_denoised.jpg', num_iterations=100)
```



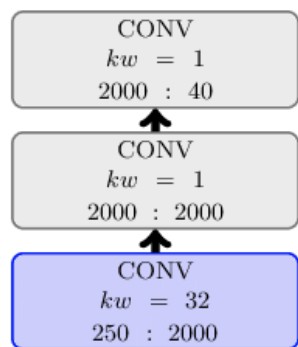
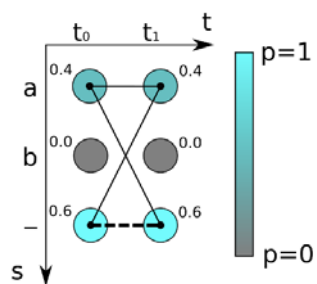
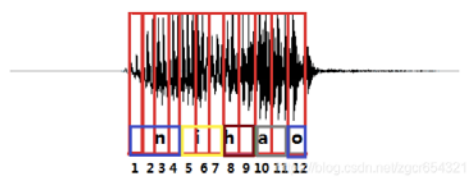
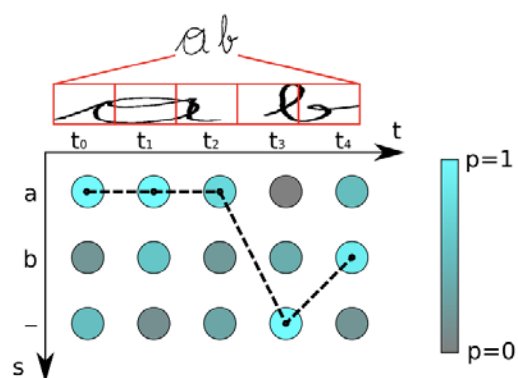
```
show_results(best_poc_turtle,
'./img/Green_Sea_Turtle_grazing_seagrass.jpg',
'./img/Pillars_of_creation_2014_HST_WFC3-UVIS_full-res_denoised.jpg')
```

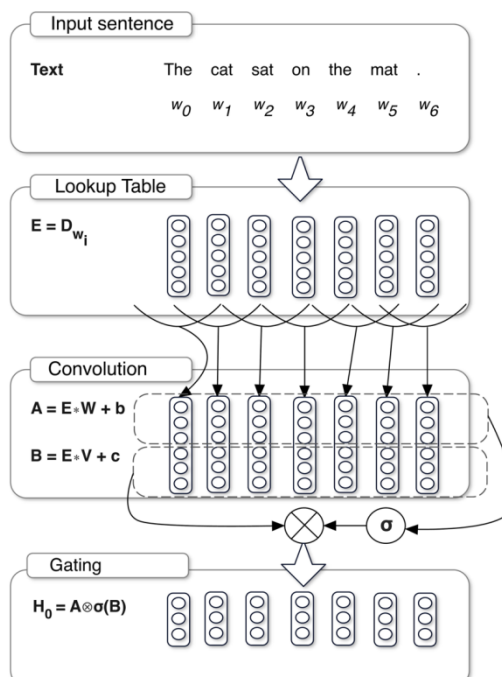
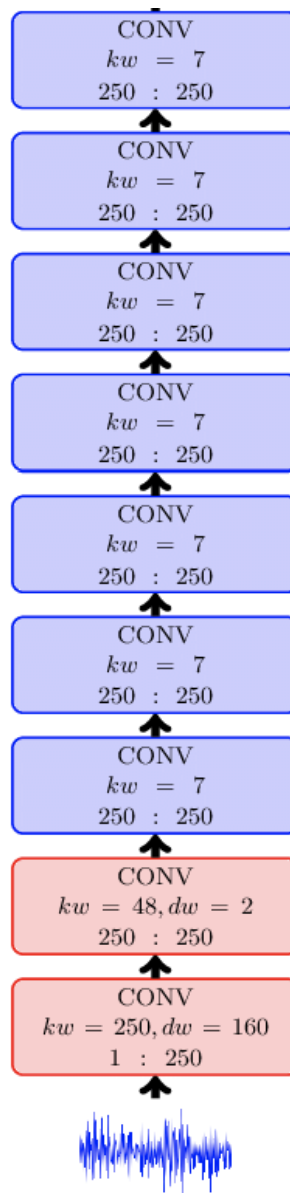


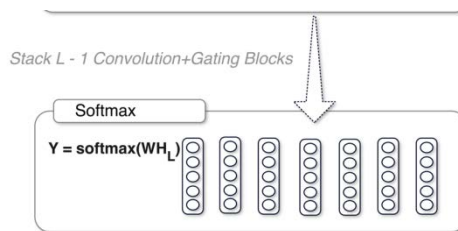
Задание 3. Распознавание звуковых изображений с помощью сверточных нейронных сетей.



Пример выполнения:







```
# эксперимент с NVIDIA/OpenSeq2Seq
import os
from os.path import exists, join, basename, splitext

git_repo_url =
'https://github.com/NVIDIA/OpenSeq2Seq.git' # адрес библиотеки
project_name =
splitext(basename(git_repo_url))[0]
if not exists(project_name):
    # клонируем себе на диск
    !git clone -q --depth 1 {git_repo_url}
    # и ставим вспомогательные библиотеки
    !pip uninstall -y -q pymc3
    !pip install --upgrade joblib
    !cd {project_name} && pip install -q -r requirements.txt
    !pip install -q youtube-dl librosa python_speech_features sentencepiece

    # создаем файлы конфигурации config
    # копируем файл по умолчанию
    !cp {project_name}/example_configs/speech2text/w2lplus_large_8gpus_mp.py {project_name}/conf.py
    # правим его под наши нужды
    !sed -i -e 's/data\//librispeech\//librivox-test-clean/test/' {project_name}/conf.py
    !sed -i -e 's/# "use_lang/"use_lang/' {project_name}/conf.py
    !echo 'backend = "librosa"' >> {project_name}/conf.py
    !cat {project_name}/conf.py
    !echo "wav_filename, wav_filesize, transcript" > {project_name}/test.csv
    !echo "test.wav, UNUSED, UNUSED" >> {project_name}/test.csv

import sys
sys.path.append(project_name)
from IPython.display import YouTubeVideo

# загрузка предобученной модели
def download_from_google_drive(file_id, file_name):
    # download a file from the Google Drive link
    !rm -f ./cookie
    !curl -c ./cookie -s -L "https://drive.google.com/uc?export=download&id={file_id}" > /dev/null
    confirm_text = !awk '/download/ {print $NF}' ./cookie
    confirm_text = confirm_text[0]
    !curl -L "https://drive.google.com/uc?export=download&confirm={confirm_text}&id={file_id}" -o {file_name}

if not exists(join(project_name, 'w2l_log_folder')):
    download_from_google_drive('10EYe040qVW6cfygSZz6HwGQDylahQNSa',
    'w2l_plus_large.tar') # вот с этого адреса
    !tar xf w2l_plus_large.tar
    !mv w2l_plus_large {project_name}/w2l_log_folder

Download...
From: https://drive.google.com/uc?id=10EYe040qVW6cfygSZz6HwGQDylahQNSa&export=download&confirm=t
To: /content/w2l_plus_large.tar
100% [1.14G/1.14G [00:30<00:00, 37.5MB/s]
'w2l_plus_large.tar'

# транскрипция для видео Youtube
# идентификатор видео ролика с Ютуб, введите свой
YOUTUBE_ID = '2AFpAATHXtc'
# посмотрим его
YouTubeVideo(YOUTUBE_ID)

Скачиваем видео, конвертируем его звук в WAV файл и распознаем.
Распознавание выполняется скриптом run.py, которому указываем аргументы:
--config_file conf.py - файл конфигурации (conf.py)
```

```
--mode=infer - показывает что сеть работает в режиме расчета, а не обучения
--infer_output_file=output.txt - файл результата
--use_horovod=False - использовать ли специальное средство для распределенных вычислений (нет, у нас его нет)
--num_gpus=1 - число графических процессоров
--batch_size_per_gpu 1 - размер пакета
```

удаляем лишние

```
!rm -rf *.wav
```

скачиваем

```
!youtube-dl --extract-audio
```

```
--audio-format wav --output
```

```
"downloaded.%(ext)s" https://www.youtube.com/watch?v={YOUTUBE_ID}
```

конвертируем

```
!ffmpeg -loglevel panic -y -i downloaded.wav -acodec pcm_s16le -ac 1 -ar 16000 {project_name}/test.wav
```

распознаем

```
!cd {project_name} && python run.py --config_file conf.py --mode=infer --infer_output_file=output.txt --use_horovod=False --num_gpus=1 --batch_size_per_gpu 1
```

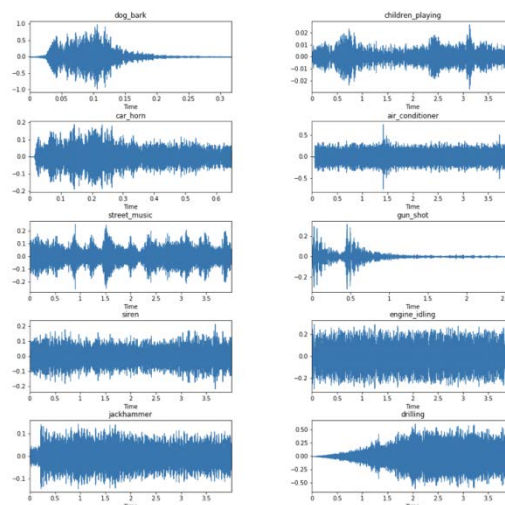
```
!tail -n1 {project_name}/output.txt
```

результаты распознавания

```
>test.wav, hieverybody in our house everybody knows that president is only the third most important job in the family so this we get a met to take a lottle extra time to say thank you to michel for the remarkable way she does the most important jout bigama am in a given extra thanks to my mother in law for the roe mal she's always been to michel in the countless selfless ways in which he help was shallin me raise moy ansuch i am incredibly luck to have these wonderful woman help me rays love and look after our cose i hope you' walts't take a moment to so thank you to the woman in your life who love you in that special way monster bylogical mobs the doctive moms and foster mots single moss grandmas godmothers atts mentors whoever you think of when you think a mother's da or take a moment like our walter remember the mobs who raised us whis big hearts sustained us and whom we mess every day no matter howld we get giving flowers is always a good idea but i hope that on this mother's day will recommit ourselves to doing more than that through deeds that match ar words let's give mothers the respect they desert yet all women the equality miser and give all parents the support they need in their most important wolls an enclude pameturnity and the turnamely sickly accommodations for workers were prignant good health dear a fordable child care flex billy at work equall pay and a decent minimum which we ask our mobs to do more than their fair shareof just about everything making sure they re treated fairly is the least we can do the idea of setting aside of sunday and made forer mothers became an official holiday with a connercional resolution a little more than a hundred years ago they did it on my eighth the same day will celebrate mother's date this ehem if commerce can make a hollow surely they could back it up with the things thare given me after all that's what my mother taught me i couldn'tjust sails no and do the right thing or say i agreed with it on principle ihve actually do it so this mother's day say thank you say i want it and what's make sure we show that gratitude ind appreciation to acts of respect throughout the year no one deserves that more than armand happy mothers day than have a right wic
```

Задание 4. Выполнить визуализацию аудио изображений.

Пример выполнения:

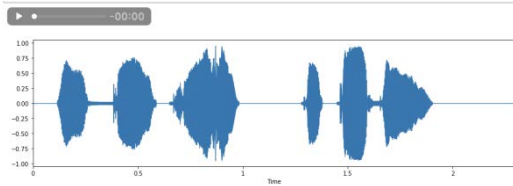


```
filename1 = 'ML_School/test.wav'
plt.figure(figsize=(15,4))
data1,sample_rate1 =
```

```

librosa.load(filename1,
mono=True,
duration=200,
librosa.display.waveplot(data1,
max_points=50000.0,
max_sr=1000)
ipd.Audio(data1,rate=sample_rate1)
sr=22050,
offset=0.0,
res_type='kaiser_best')
sr=sample_rate1,
x_axis='time', offset=0.0,

```



```

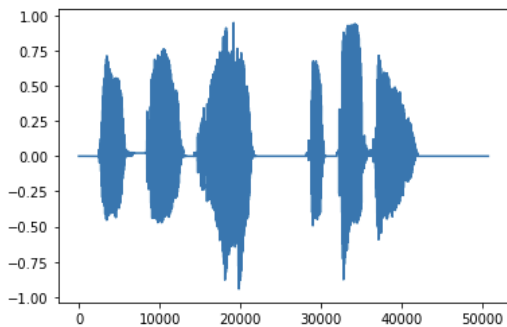
print(data1)
print(len(data1))
print(sample_rate1)

```

```

import librosa
import matplotlib.pyplot as plt
signal, sr =
librosa.load('ML_School/test.wav') # загружаем (все прочие параметры по умолчанию)
plt.plot(signal)

```



```

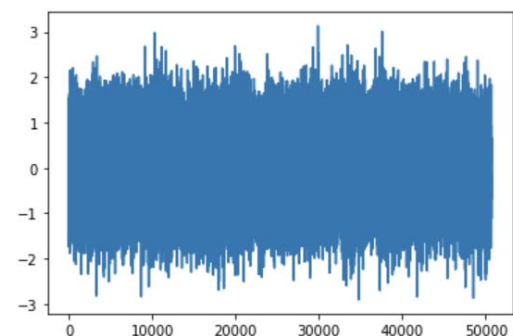
# добавление шума
import numpy as np

```

```

H=0.1 #
H=0.7 # попробуйте разный уровень шума
noise=np.random.normal(0, H, signal.shape[0]) #шум - случайная величина
signal_noise = signal+noise #
plt.plot(signal_noise) #
ipd.Audio(signal_noise,rate=sr) #

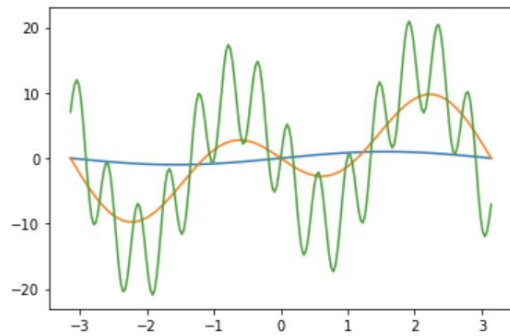
```



```

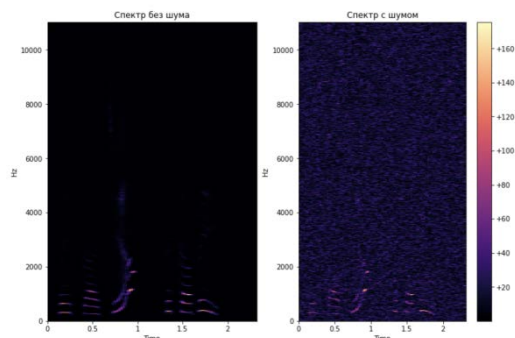
# частотное представление и спектр
x = np.linspace(-np.pi, np.pi, 201)
y1=np.sin(x)
y2=5*np.sin(x)-6*np.sin(2*x)
y3=3*np.sin(x)-12*np.sin(2.2*x)+7.6*np.sin(14*x)
plt.plot(x,y1,x,y2,x,y3)

```

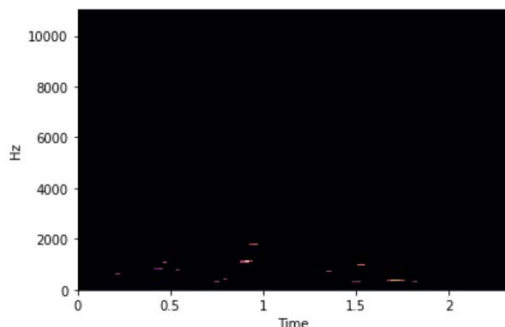


```
plt.figure(figsize=(12, 8)) # сделаем полотно
plt.subplot(1, 2, 1) # первый подграфик
D_compl=librosa.stft(signal)# считаем спектр чистого сигнала (он комплексный)
D=np.abs(D_compl) # берем только амплитуду комплексного числа
print(D_compl.shape) # посмотрим, что же такое спектр, это двумерный массив
librosa.display.specshow(D,# нарисуем спектр, массив, который отображать
                        x_axis='time', # временные отметки по горизонтальной оси
                        y_axis='linear') # вертикальная ось - линейная (а можно логарифмическую 'log')
plt.title('Спектр без шума') # название графика

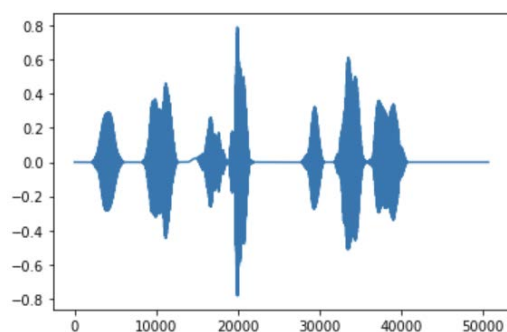
# спектрограмма для звука с шумом
plt.subplot(1, 2, 2) # второй подграфик
D_compl=librosa.stft(signal_noise)
D=np.abs(D_compl)
librosa.display.specshow(D, x_axis='time',y_axis='linear')
plt.colorbar(format='%+2.0f') # нарисуем цветовую шкалу
plt.title('Спектр с шумом')
```



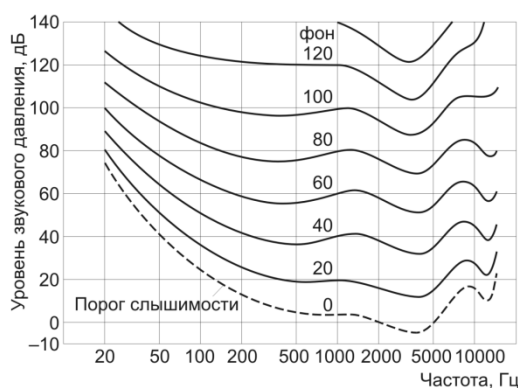
```
threshold=70 # порог для отсеивания частот
freq_max=700 # максимальная частота (в условных единицах)
D_compl1=D_compl.copy() # сделаем копию массива, чтобы не испортить
D_compl1[np.where(D<threshold)]=0 # найдем все элементы которые меньше порога и заменим их нулями
D_compl1[freq_max,:]=0
librosa.display.specshow(D_compl1, x_axis='time',y_axis='linear')
```



```
signal_reconstructed=librosa.istft(D_compl1)
plt.plot(signal_reconstructed)
ipd.Audio(signal_reconstructed,rate=sr)
```



mel-спектры



1.5. Модуль «Телемедицина и СППВР»

Примеры практических заданий

Задание 1. Трансформация данных. Преобразовать данные к определенному представлению с применением типичных средствами трансформации: преобразование временных данных, квантование, сортировка, слияние, группировка.

Задание 2. Изучение инструментов предобработки и подготовки данных к анализу. На примере аналитической платформы Loginom изучить следующие средства: очистка от шумов и сглаживание рядов данных, восстановление пропущенных значений, редактирование аномальных значений.

Промежуточная аттестация.

2.1. Модуль «Управление проектами по цифровой трансформации здравоохранения»

Перечень примерных тестовых заданий к зачету

1. Задание

Стандарт ISO/IEC 12207

- ☒ определяет структуру жизненного цикла
- ☐ определяет основные понятия и определения жизненного цикла
- ☐ определяет область применения моделей жизненного цикла

2. Задание

Упорядочите четыре квадранта спирали спиральной модели жизненного цикла:

1. планирование – определение (уточнение) целей, задач, вариантов решения и ограничений, сбор требований, в том числе на основе рекомендаций заказчика;
2. оценка предложенных решений и анализ риска (превышения сроков и стоимости проекта) на основе начальных требований и реакции заказчика;
3. конструирование и выполнение основных работ итерации;
4. оценивание заказчиком текущих результатов конструирования.

3. Задание

Как располагаются функциональные блоки в порядке доминирования?

- ☐ Сверху вниз
- ☐ Слева направо
- ☒ Наиболее доминирующая работа располагается в левом верхнем углу диаграммы, наименее – в правом нижнем
- ☐ Не имеет значения

2.2. Модуль «Инженерия искусственного интеллекта в медицине»

Перечень примерных вопросов к зачету

1. Методы машинного обучения с учителем. Основные библиотеки и инструменты Python.
2. Методы машинного обучения с учителем. Классификация и регрессия.
3. Методы машинного обучения с учителем. Алгоритмы машинного обучения с учителем.
4. Методы машинного обучения без учителя. Типы машинного обучения без учителя.
5. Методы машинного обучения без учителя. Предварительная обработка и масштабирование
6. Методы машинного обучения без учителя. Снижение размерности, выделение признаков и множественное обучение.
7. Методы машинного обучения без учителя. Анализ главных компонент.
8. Методы машинного обучения без учителя. Множественное обучение
9. Методы машинного обучения без учителя. Кластеризация.
10. Методы машинного обучения без учителя. Сравнение и оценка алгоритмов кластеризации
11. Типы данных. Объединение алгоритмов Категориальные переменные
12. Типы данных. Объединение алгоритмов. Биннинг, дискретизация, линейные модели.
13. Типы данных. Объединение алгоритмов. Применение экспертных знаний.
14. Типы данных. Объединение алгоритмов. Построение конвейеров.

2.3. Модуль «Вычислительная анатомия на базе технологий AR/VR»

Перечень примерных вопросов к зачету

1. Определение понятия «виртуальная реальность» (VR)
2. Определение понятия «дополненная реальность» (AR)
3. Основные понятия виртуальной реальности.
4. Сетевая виртуальная реальность
5. Аппаратные средства виртуальной реальности
6. Виртуальная реальность в промышленности
7. Виртуальное обучение, тренажеры и симуляторы
8. Системы виртуальной реальности в проектировании
9. Виртуальные решения в медицинской практике
10. Компании-лидеры в развитии систем виртуальной реальности
11. Виды виртуальной реальности
12. Объекты виртуальной реальности
13. Виртуальная реальность и дополненная реальность - сравнение.
14. Этапы и технологии создания систем VR, структура и компоненты.
15. Этапы и технологии создания систем AR, структура и компоненты.

2.4. Модуль «Анализ и визуализация медицинских данных»

Перечень примерных вопросов к зачету

1. Виды представления изображений. Классификация изображений с помощью сверточных нейронных сетей.
2. Векторное представление изображений. Перенос обучения.

3. Генерация изображений. Перенос стиля изображения.
4. Написание кода программы для обработки изображений, используя заданный набор конфигураций нейронной сети. Выбрать наилучшую конфигурацию. Обосновать выбор.
5. Написание кода программы для генерации изображений, используя заданный набор конфигураций нейронной сети. Выбрать наилучшую конфигурацию. Обосновать выбор.
6. Физико-технические основы ультразвукового метода исследования
7. Физико-технические основы рентгеновской компьютерной томографии.
8. Физико-технические основы магнитно-резонансной томографии.
9. Методы радионуклидной визуализации: сканирование, сцинтиграфия, ОФЭКТ, преимущества и недостатки.
10. Электрокардиография
11. Система управления данными и компонентами автоматизации OLE математических пакетов
12. Алгоритм оценивания параметров пространственной и временной модели.
13. Цифровые изображения, методы преобразований.
14. Аналитическая обработка неструктурированной информации.
15. Этапы обработки информации в приложениях

2.5. Модуль «Телемедицина и СППВР»

Перечень примерных вопросов к зачету

1. Дайте определение телемедицины.
2. Перечислите основные этапы развития телемедицины в России, их основные черты.
3. Перечислите, что необходимо оцифровывать при оказании медицинской помощи с применением телемедицинских технологий.
4. Дайте определение телемедицинской деонтологии.
5. За что несут ответственность участники проведения телемедицинских консультаций?
6. Какие типичные ошибки допускаются при формировании материалов для телеконсультаций?
7. Какие исследования обычно включаются в материалы телеконсультации больных с заболеваниями сердечно-сосудистой системы?
8. Из чего состоит комплекс оборудования для персональной телемедицины?
9. Понятия телемедицины, медицинской телематики, электронного здравоохранения, информационно-телекоммуникационных технологий.
10. Телерадиология, телепатология, телекардиология.
11. Понятие и виды телеконсультаций.
12. Алгоритм подготовки и проведения телеконсультации.
13. Информационные технологии в работе медицинского персонала.
14. Домашняя (персональная) телемедицина. Носимые комплексы. Планшетные компьютеры. Консультативные центры.
15. Медицинская, социальная, экономическая эффективность телемедицины.

IX. НОРМАТИВНЫЕ ПРАВОВЫЕ АКТЫ

Дополнительная профессиональная программа (программа профессиональной переподготовки) ИТ-профиля «Цифровизация в здравоохранении» разработана в соответствии с нормами Федерального закона РФ от 29 декабря 2012 года № 273-ФЗ «Об образовании в Российской Федерации», с учетом требований приказа Минобрнауки России от 1 июля 2013 г. № 499 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным профессиональным программам», с изменениями, внесенными приказом Минобрнауки России от 15 ноября 2013 г. № 1244 «О внесении изменений в Порядок организации и осуществления образовательной деятельности по допол-

нительным профессиональным программам, утвержденный приказом Министерства образования и науки Российской Федерации от 1 июля 2013 г. № 499», приказа Министерства образования и науки РФ от 23 августа 2017 г. N 816 «Об утверждении Порядка применения организациями, осуществляющими образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ»; ФГОС ВО по направлению подготовки 09.04.01 Информатика и вычислительная техника (уровень магистратуры), утвержденного приказом Минобрнауки России от 19 сентября 2017 г. № 918, а также профессионального стандарта «Руководитель разработки программного обеспечения» утвержденного приказом Министерства труда и социальной защиты РФ от 22 июля 2022 г. № 423н.