Конкурсное задание номинации
«Искусственный интеллект»

Искусственный интеллект (ИИ) – это область компьютерных наук, изучающая вопросы, связанные с разработкой систем, имитирующих человеческие когнитивные способности, такие как обучение, рассуждение, решение проблем и принятие решений.

Машинное обучение – это область искусственного интеллекта, определяющая набор методов, техник и парадигм, которые позволяют системе обучаться на прецедентах и выявлять закономерности, делая прогнозы без явного программирования.

Глубокое обучение – это область машинного обучения, которая использует многослойные нейронные сети для анализа данных и принятия решений.

Обязанности учащегося:

Этап	Действия участника
До конкурса	• Скачать публичные датасеты на свой носитель: – HaGRID (рекомендуется) – ASL Alphabet – Sign Language MNIST • Подготовить ноутбук с дискретной видеокартой, веб-камерой и микрофоном; • Установить все необходимые библиотеки (PyTorch/TensorFlow, OpenCV, MediaPipe, PyQt5).
В начале конкурса	• Продемонстрировать члену жюри наличие датасетов на своём носителе; • Подготовить структуру папок train/ и test/ согласно требованиям задания.
По окончании работы	• Предоставить жюри: 1) Флеш-диск с полной структурой проекта (включая обученную модель); 2) Демонстрацию работы приложения на своём ноутбуке; 3) Дизайн-проект в формате PDF/документа.

Конкурс состоит из одного задания по разработке приложения с использованием алгоритмов машинного обучения, в т.ч. нейронных сетей.

Учащийся за отведенное время должна предоставить готовое приложение, оцениваемое по следующим критериям:

1. Дизайн-проект

Начисление баллов выполняется за корректную и качественную реализацию следующих пунктов:

общее описание работы приложения;

блок-схема алгоритма работы приложения;

схемы всех экранов приложения, в т.ч. переходы между ними;

2. Прототип

Начисление баллов выполняется за корректную и качественную реализацию следующего функционала:

реализация основных функций задания;

соответствие дизайн-проекту;

запуск на целевом устройстве;

обеспечение приемлемой скорости обработки видеопотока (либо

скорость обработки видеопотока - не менее 1 кадра в секунду (1 FPS));

оценка точности (accuracy) базовой модели нейронной сети на тестовых данных;

построить ROC-кривую.

3. Оптимизация

Начисление баллов выполняется за корректную и качественную реализацию следующих пунктов:

использование аугментации данных;

дообучение (fine-tuning) готовой модели для решения конкурсного задания;

оценка точности (accuracy) модели нейронной сети на тестовых данных;

построить ROC-кривую;

построение матрицы ошибок (confusion matrix);

оценка метрики F1.

4. Сборка и запуск проекта

Начисление баллов выполняется за корректную и качественную реализацию следующих пунктов:

корректный запуск собранного проекта на целевом устройстве (окно приложения с демонстрацией видеопотока с веб-камеры);

отсутствие «падений» во время работы приложения;

наложение результата работы алгоритма машинного обучения поверх видеопотока;

вывод оценки производительности поверх видеопотока (в FPS);

оценка точности (accuracy) модели нейронной сети на тестовых данных, представленных в папке “test” на целевой машине;

оценка “уверенности” алгоритма в решении задачи, при получении видеопотока от камеры.

Конкурсное задание

Задание 1. Распознавание жестов рук в реальном времени

1. Описание программного средства.

Разрабатываемое программное средство состоит из двух независимых частей:

Python-ноутбук (или Python-скрипт), в котором выполняется проектирование, обучение и/или дообучение модели нейронной сети для распознавания жестов рук на основе входного изображения;

Основное приложение, обеспечивающее взаимодействие с пользователем, написанное на Python с применением библиотеки PyQt5.

1.1. Структура и наполнение Python-ноутбука (или Python-скрипта).

В разрабатываемом скрипте должна быть представлена возможность указать 2 папки:

папка “train”, в которой размещаются изображения из тестового набора, на которой участник выполняет обучение и/или дообучение, а также тестирование модели нейронной сети; папка “test”, которая будет использоваться для “слепого” тестирования
членами комиссии и оценки точности на новых данных.

Используя рекомендуемые библиотеки (PyTorch/TensorFlow, OpenCV, MediaPipe), участник должен:

- Реализовать базовую модель нейронной сети (допускается использование предобученных моделей и архитектур, например, MobileNetV2 с дополнительными слоями);
- Оценить полученную точность (Accuracy) на валидационном наборе;
- Применить аугментацию данных (повороты, изменение яркости, сдвиги) для улучшения обобщающей способности модели;

- Выполнить дообучение (fine-tuning) модели на предоставленном наборе данных;
- Построить матрицу ошибок (confusion matrix) и ROC-кривую;
- Рассчитать метрику F1 для каждого класса жестов.

Лучшая версия обученной модели должна сохраняться в формате .pt или .h5. Название файла должно начинаться с фамилии участника, написанной латиницей (например, ivanov_gesture_model.pt).

Результаты работы ноутбука или скрипта:

- Вывод метрик: точность (Accuracy), F1-score для каждого класса;
- Визуализация матрицы ошибок и ROC-кривой;
- Файл с сохранённой обученной моделью.

Структура проекта:

```
gesture_recognition_project/
├── data/                    # Обучающие и тестовые данные (подготовлены участником)
│   ├── train/               #  Минимум 300 изображений на класс
│   │   ├── fist/
│   │   ├── palm/
│   │   ├── thumb_up/
│   │   ├── thumb_down/
│   │   ├── ok/
│   │   └── victory/
│   └── test/                #  Для финальной проверки (заполняется участником)
│       ├── fist/
│       ├── palm/
│       └── ...              #  (структура аналогична train/)
│
├── notebooks/
│   ├── 01_data_preparation.ipynb  # Скрипт преобразования исходного датасета → структура train/
│   ├── 02_model_training.ipynb    # Обучение + аугментация + дообучение
│   └── 03_evaluation.ipynb        # Метрики: Accuracy, F1, confusion matrix, ROC
│
├── app/
│   ├── main.py              # Точка входа приложения (PyQt5)
```

```
│   ├── camera_widget.py
│   └── gesture_recognizer.py      # Интеграция модели + MediaPipe
│
├── models/
│   └── [фамилия_латиницей]_gesture_model.pt  # Пример: ivanov_gesture_model.pt
│
├── docs/
│   ├── design_project.pdf         # Дизайн-проект: архитектура, блок-схема, макеты интерфейса
│   └── dataset_sources.txt        # Ссылки на использованные датасеты + объём данных
│
├── requirements.txt               # Список зависимостей для воспроизведения
└── README.md                      # Инструкция запуска
```

1.2. Организация работы основного приложения.

При запуске основного приложения появляется главное окно с следующим функционалом:

- Элемент для отображения видеопотока с веб-камеры;
- Выбор источника видеопотока (веб-камера, файл);
- Кнопка «Старт распознавания»;
- Панель выбора загруженной модели нейронной сети;
- Информационная панель для отображения статистики.

При нажатии кнопки «Старт распознавания» приложение должно:

- Загрузить выбранную модель нейронной сети в память GPU/CPU;
- Активировать видеопоток с камеры;
- Начать детекцию кистей рук (с использованием MediaPipe Hands или аналога);
- Выполнять классификацию жестов в реальном времени.

Поверх видеопотока отображается:

- Обведённая рамкой область с распознанной кистью руки;
- Название распознанного жеста (крупным шрифтом);
- Уровень «уверенности» модели в процентах;
- Текущая производительность обработки (в FPS).

Поддерживаемые жесты (минимум 6 классов):

1. Кулак
2. Раскрытая ладонь
3. «Палец вверх»
4. «Палец вниз»
5. «ОК» (кольцо из большого и указательного пальцев)
6. «Мир» (указательный и средний пальцы в форме V)

2. Набор данных

Рекомендуется использовать один из следующих наборов данных:

- **ASL Fingerspelling Alphabet Dataset** (американский жестовый алфавит) - адаптированный под 6 базовых жестов;
- **HaGRID (HAnd Gesture Recognition Image Dataset)** - крупный набор жестов рук;
- **Самостоятельно собранный набор** (участник может собрать до 500 изображений на 6 классов с помощью веб-камеры в течение первых 30 минут конкурса).

Формат изображений: RGB, размер 128×128 или 224×224 пикселей.\

Допускается предварительная обрезка изображений до области кисти руки.

Если в исходном датасете недостаточно изображений для класса - используйте аугментацию (повороты ±15°, изменение яркости ±20%, горизонтальное отражение) или соберите недостающие изображения с веб-камеры за первые 20 минут конкурса.

3. Технические требования к реализации

- Минимальная скорость обработки: **не менее 10 кадров в секунду (10 FPS)** при разрешении 640×480;
- Точность модели на тестовом наборе: **не менее 85%** для базовой версии, **не менее 92%** после оптимизации;
- Отсутствие критических ошибок («падений») при непрерывной работе не менее 5 минут.

Показатели и критерии оценок конкурса
«Искусственный интеллект»

Категория	Макс. баллов	Подкритерии и распределение баллов
Дизайн-проект	20	• Общее описание архитектуры приложения (5 баллов) • Блок-схема алгоритма работы (захват кадра

Категория	Макс. баллов	Подкритерии и распределение баллов
		-детекция кисти - классификация - вывод результата) (7 баллов) • Макеты всех экранов интерфейса с переходами (5 баллов) • Описание технических решений (выбор архитектуры нейросети, библиотек) (3 балла)
Прототип	20	• Базовая функциональность: видеопоток + детекция кисти + распознавание ≥3 жестов (6 баллов) • Отображение результатов поверх видеопотока (рамка + название жеста) (3 балла) • Соответствие дизайн-проекту (4 балла) • Скорость обработки ≥10 FPS при разрешении 640×480 (4 балла) • Базовая точность модели (Accuracy) ≥75% на валидации (3 балла)
Оптимизация	40	• Применение аугментации данных (повороты, яркость, сдвиги) - 6 баллов • Дообучение (fine-tuning) предобученной модели (например, MobileNetV2) - 8 баллов • Точность (Accuracy) ≥85% - 7 баллов; ≥92% - +3 балла (итого 10) • Построение матрицы ошибок (confusion matrix) - 4 балла • Расчёт метрики F1 для каждого класса - 5 баллов • Построение ROC-кривой (для мультиклассовой задачи - One-vs-Rest) - 7 баллов
Сборка и запуск	20	• Корректный запуск на устройстве участника - 5 баллов • Стабильность: отсутствие «падений» при работе ≥5 минут - 5 баллов • Отображение производительности (FPS) и «уверенности» модели (%) поверх видеопотока - 5 баллов • Точность на «слепом» тесте (папка test) ≥85% - 5 баллов

Участник, набравший наибольшее количество баллов, считается победителем.

В случае набора участниками одинакового количества баллов, победителем конкурса считается участник, набравший наибольшее количество баллов и выполнивший конкурсное задание за наименьшее время.

В случае разногласий окончательное решение оценки конкурса принимает председатель жюри.