**Министерство образования Республики Беларусь**

**Учреждение образования**

**«Брестский государственный технический университет»**

**Кафедра ИИТ**

**ОТЧЕТ**

**Тема: «Разработка модели для детекции препятствий для автономного трактора»**

**Выполнили:**

Студенты 4-го курса,

ФЭИС,

Группы ИИ-22

Борейша О.С.

Гузаревич Д.А.

Копанчук Е.Р.

Леваневская Н.И.

**Проверил:**

Козик И.Д.

**Брест 2024**

**Подготовка датасета для модели детекции**

Разработка модуля компьютерного зрения для автономного трактора требует создания надежной модели детекции объектов, способной эффективно распознавать выбранные классы: cattle, human, tractor, fallen\_tree, power\_line. Для достижения этой цели был проведен полный цикл подготовки данных и экспериментального обучения моделей, включающий разметку, аугментацию и выбор архитектуры (рис. 1) (рис 2).

Разметка данных. Для обучения модели потребовался набор данных, отражающий реальные условия эксплуатации трактора. Силами команды из пяти студентов была проведена разметка изображений, содержащих целевые классы. Основные этапы работы:

1. Сбор данных: собраны 700 изображений, охватывающих разнообразные условия освещения, ракурсы и сцены.

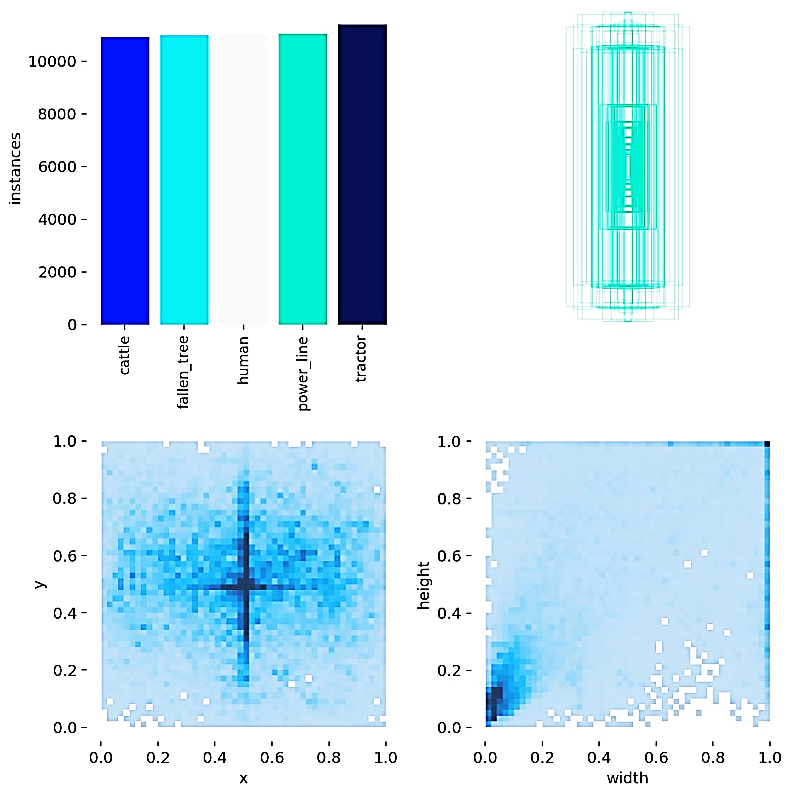
2. Ручная разметка: на изображениях были выделены объекты каждого из пяти классов. В общей сложности размечено около 1500 объектов.

Балансировка данных. В процессе анализа стало ясно, что количество объектов в классах значительно различается (например, людей или тракторов было меньше, чем линий электропередач). Чтобы исключить дисбаланс, был выполнен процесс балансировки:

1. Общее количество объектов было уравнено: около 400 объектов для каждого класса.

2. Искусственная балансировка позволила обеспечить равное представление всех классов, что снизило риск смещения модели в сторону более представленных классов.

Аугментация данных. Для повышения устойчивости модели к различным условиям эксплуатации и увеличения объема тренировочного набора была выполнена аугментация данных. Разнообразие данных увеличено с 2000 объектов до 20 000 объектов. Применены следующие методы. Изменение яркости и контрастности. Добавление шума и размытия для имитации погодных условий. Повороты, масштабирование и обрезка для учета различных углов съемки. Аугментированный набор данных обеспечил разнообразие и позволил модели лучше обобщать знания, снижая риск переобучения.

****

*Рисунок 2. Распределение классов и bboxes в получивщемся наборе данных*

**

*Рисунок 1. Пример тренировочных батчей*

**Обучение модели**

Для обучения модели детекции объектов мы использовали фреймворк **Ultralytics YOLOv8**, который предоставляет удобные инструменты для настройки и выполнения процесса обучения. Обучение проводилось в облачной среде **Google Colab**, что позволило использовать мощные графические процессоры (GPU) для ускорения вычислений.

Для проверки производительности различных конфигураций YOLOv8 (таблица 1) были выбраны следующие параметры:

* **Обучающие веса**: предобученные веса для каждой модели: yolov8n.pt, yolov8s.pt, yolov8m.pt.
* **Размер входного изображения (imgsz)**: 640px или 800px, в зависимости от эксперимента.
* **Размер батча (batch)**: 16 для большинства экспериментов, уменьшенный до 8 для модели с большими изображениями (800px и весами YOLOv8m).

Таблица 1. Конфигурация моделей

|  |  |  |  |
| --- | --- | --- | --- |
| Модель | Веса | Размер изображения (px) | Batch |
| YOLOv8n | yolov8n.pt | 800 | 16 |
| YOLOv8s | yolov8s.pt | 800 | 16 |
| YOLOv8m | yolov8m.pt | 640 | 16 |
| YOLOv8m | yolov8m.pt | 800 | 8 |

Обучение проводилось на подготовленном и аугментированном наборе данных, состоящем из **20 000 объектов** пяти классов: **cattle**, **human**, **tractor**, **fallen\_tree**, **power\_line**. Каждая модель обучалась с использованием библиотеки Ultralytics.

#### Время обработки одного изображения

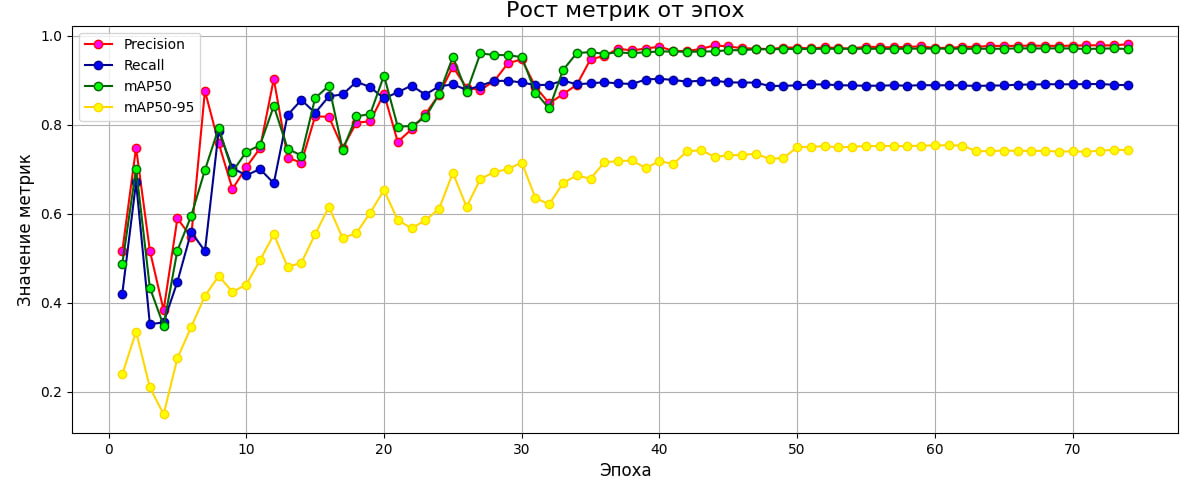
Для оценки производительности на этапе вывода была замерена скорость обработки одного изображения для каждой модели. Тестирование проводилось на процессоре AMD Ryzen 3 Cores 4 Threads 4 (таблица 2).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 3.1.2. Конфигурация моделей   |  |  |  | | --- | --- | --- | | Модель | Размер изображения (px) | Время обработки (мс) | | YOLOv8n | 800 | 156 | | YOLOv8s | 800 | 488 | | YOLOv8m | 640 | 559 | | YOLOv8m | 800 | 854 | |  |  |
|  |  |  |

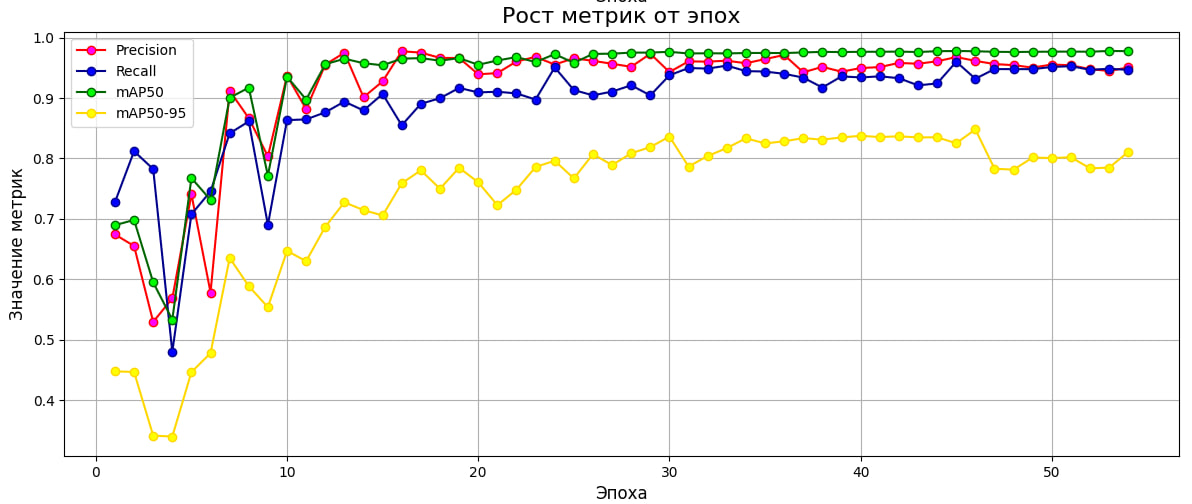
После завершения обучения (рис. 3.1.1) модели были протестированы на отдельном тестовом наборе данных, чтобы оценить их точность (mAP), производительность (FPS) и устойчивость к реальным условиям.

**Сравнение моделей**

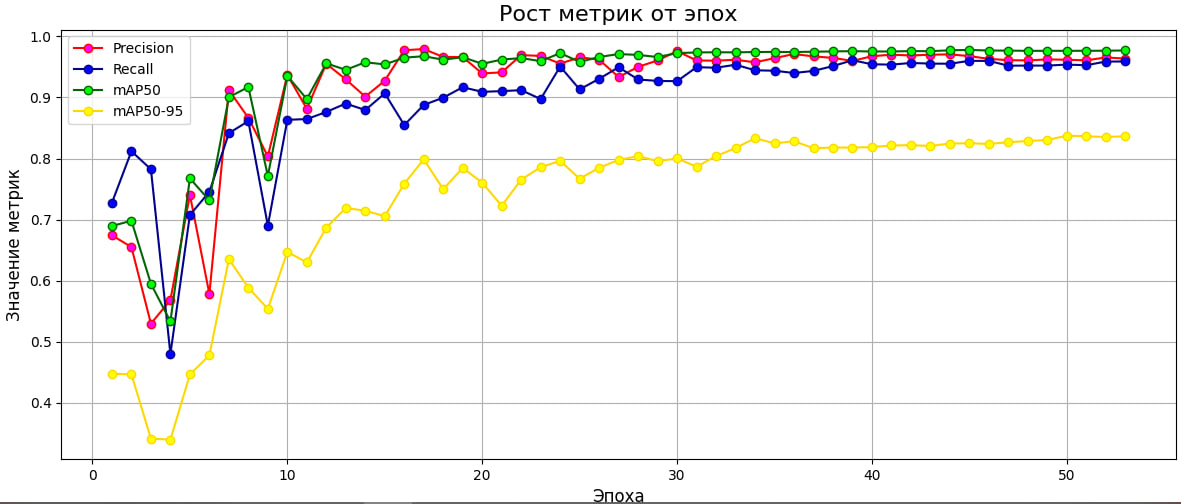
Для каждой из четырех моделей были построены графики метрик, показывающие изменение значений **precision**, **recall**, **mAP50** и **mAP50-95** в процессе обучения. Эти графики позволили визуально оценить, как модели справлялись с задачей детекции по мере увеличения числа эпох (рис 3) (рис 4) (рис 5) (рис 6).



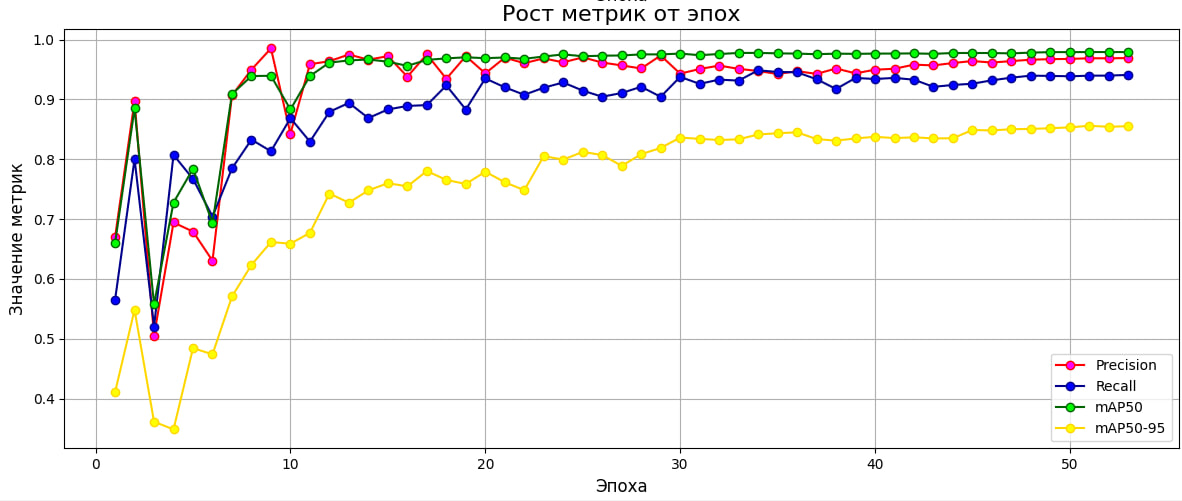
*Рисунок 3. График четырех основных метрик модели yolov8n*

**

*Рисунок 4. График четырех основных метрик модели yolov8s*

**

*Рисунок 5. График четырех основных метрик модели yolov8m(640px)*

**

*Рисунок 6. График четырех основных метрик модели yolov8m(800px)*

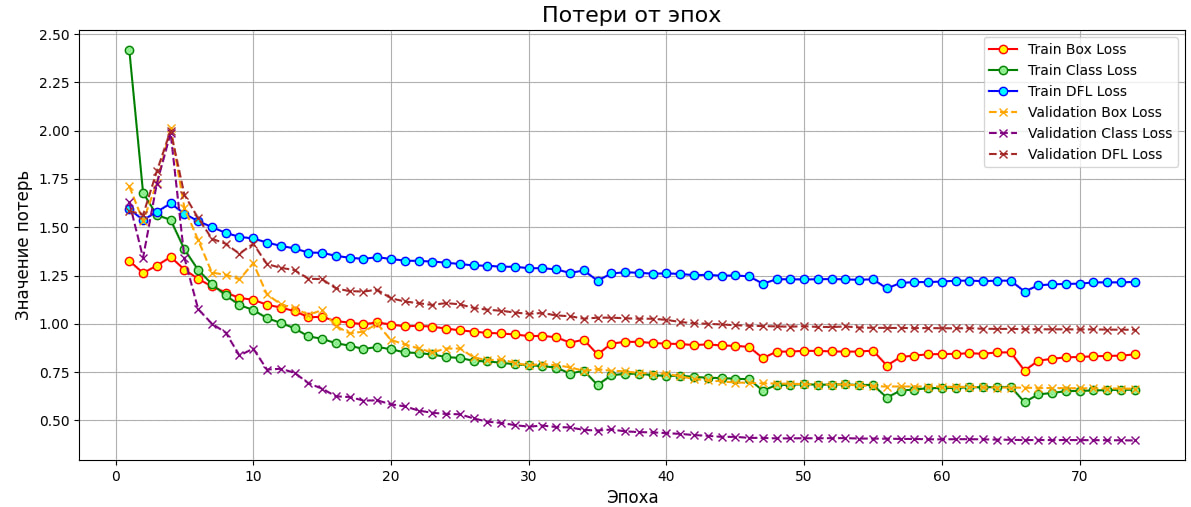
Модель yolov8n (рис 4.1.1.) с минимальной вычислительной сложностью продемонстрировала самые низкие значения метрик, но при этом отличилась высокой скоростью обучения. На последних эпохах **mAP50** составил **74%**, а **mAP50-95** — **66%**.

Модель yolov8s (рис 4.1.2.) среднего уровня сложности показала более высокие результаты, достигая **mAP50 = 97%** и **mAP50-95 = 77%** на последних эпохах.

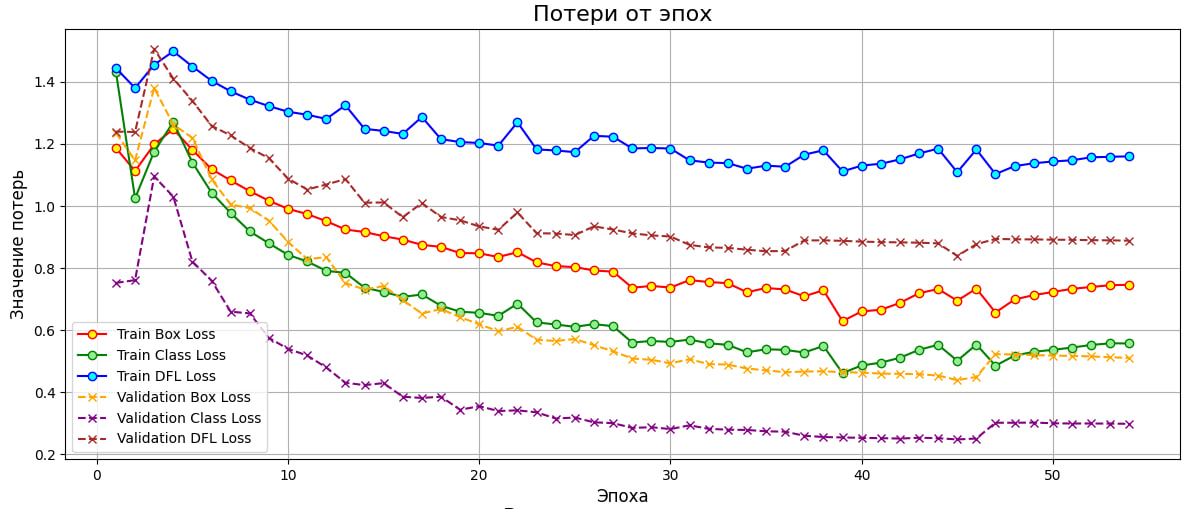
Модель yolov8m(640px) (рис 4.1.3.) модель с уменьшенным размером входных изображений показала оптимальное сочетание скорости обучения и качества. Итоговый **mAP50** достиг **98%**, а **mAP50-95** — **84%**.

Модель yolov8m(800px) (рис 4.1.4.) несмотря на использование увеличенного разрешения изображений, модель не смогла существенно превзойти результаты предыдущей модели. Итоговые значения mAP50 = 97% и mAP50-95 = 83%, при этом процесс обучения занимал больше времени.

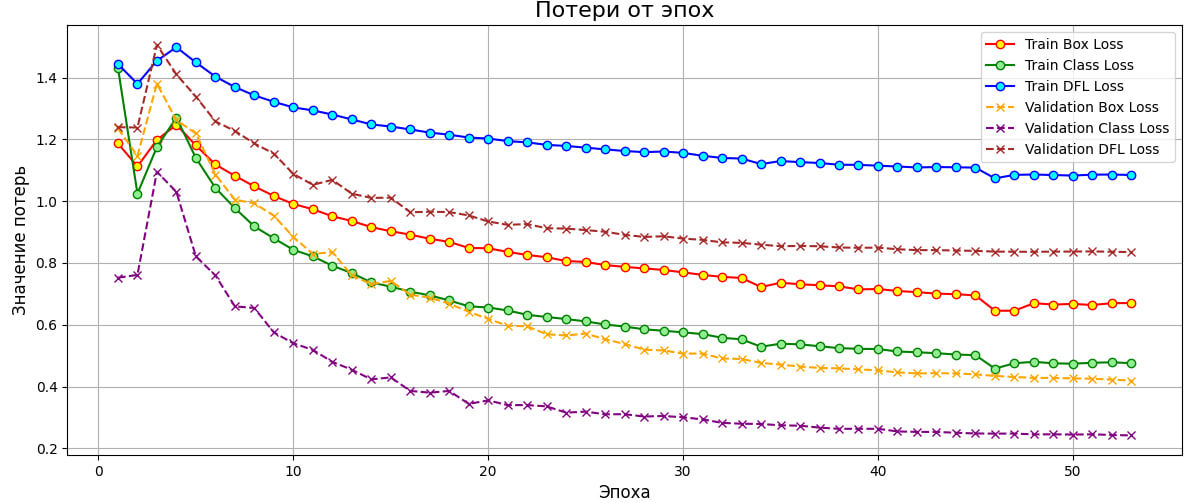
Теперь можем перейти к анализу ошибок для всё тех же четырех моделей YOLOv8. Для каждой модели были построены графики ошибок. Они позволяют оценить процесс обучения и тестирования с точки зрения снижения потерь (рис 7) (рис 8) (рис 9) (рис 10).

**

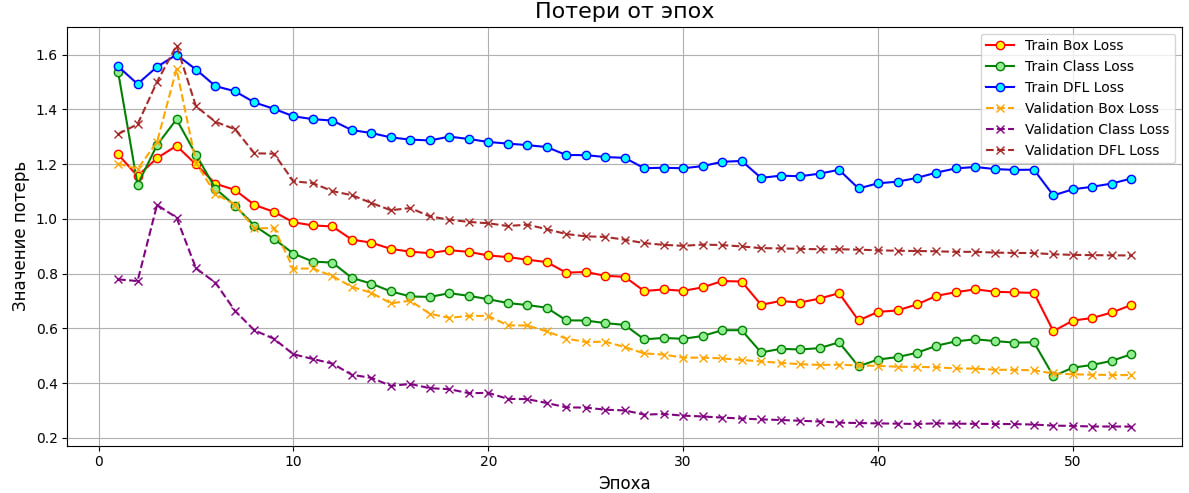
*Рисунок 7. График ошибок модели yolov8n*

**

*Рисунок 8. График ошибок модели yolov8s*

**

*Рисунок 9. График ошибок модели yolov8m(640px)*

**

*Рисунок 10. График ошибок модели yolov8m(800px)*

Модель yolov8n (рис 7) быстро достигла стабилизации ошибок в середине обучения, но итоговые значения остались высокими: **val/box\_loss = 66%**, **val/cls\_loss = 40%, val/dfl\_loss = 96%**. Это свидетельствует о недостаточной способности модели к обобщению данных.

Модель yolov8s (рис 8) близясь к концу обучения ошибки этой модели уменьшались, но после чего стабилизировались на уровнях: **val/box\_loss = 50%**, **val/cls\_loss = 30%, val/dfl\_loss = 88%.** Снижение потерь было более устойчивым, чем у YOLOv8n.

Модель yolov8m(640px) (рис 9) показала плавное снижение ошибок на протяжении всего процесса обучения. Итоговые значения составили: **val/box\_loss = 42%**, **val/cls\_loss = 24%, val/dfl\_loss = 83%,** что указывает на её высокую эффективность**.** Это подтверждает её способность к точному распознаванию объектов, включая сложные классы, такие как линии электропередач и упавшие деревья.

Модель yolov8m(800px) (рис 10) с показателями ошибок была близка к предыдущей модели, но всё же не превосходили ее. Итоговые значения составили: **val/box\_loss = 43%**, **val/cls\_loss = 24%, val/dfl\_loss = 87%.**

Проведенный анализ ошибок и метрик обучения моделей YOLOv8 позволил сделать следующие выводы из полученных результатов:

Модель yolov8n с минимальной вычислительной сложностью продемонстрировала высокую скорость обучения, однако её итоговые результаты оказались недостаточными для детекции объектов с высокой точностью. Значения ошибок быстро стабилизировались и не падали. Эта модель не смогла обеспечить необходимую точность для сложных классов, таких как линии электропередач и упавшие деревья.

Модель yolov8s среднего уровня сложности показала более устойчивое снижение ошибок. Она оказалась более эффективной, чем YOLOv8n, но не достигла уровня моделей с более крупной архитектурой.

Модель yolov8m(640px) эта модель показала оптимальное сочетание скорости обучения и качества детекции. Значения ошибок стабилизировались на минимальных уровнях. Она продемонстрировала отличные результаты в распознавании всех классов объектов, включая сложные категории.

Модель yolov8m(800px) несмотря на использование увеличенного разрешения входных изображений, результаты этой модели практически не отличались от результатов yolov8m (640px). Увеличенное разрешение замедлило процесс обучения без существенного повышения точности.

**Квантизация моделей**

Представленные модели:

1. yolo8m\_fp32 – не квантизованная модель.
2. yolo8m\_qdynamic – квантизованная динамически модель.
3. Yolo8m\_qstatic – квантизованная статически модель.

Результаты тестирования представлены в таблицах 3-5.

Таблица 3 – Метрики модели yolo8m\_fp32

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **mAP50** | **mAP50-95** | **Inference Time per Image (ms)** | **FPS** | **Model Size (MB)** |
| **all** | 0.951 | 0.914 | 0.946 | 0.793 | 1624.27 | 0.616 | 98.82 |
| **cattle** | 0.8767 | 0.8364 | 0.8344 | 0.6678 |  |  |  |
| **fallen\_tree** | 0.9632 | 1.0000 | 0.9950 | 0.9522 |  |  |  |
| **human** | 0.9844 | 0.8282 | 0.9242 | 0.7158 |  |  |  |
| **power\_line** | 0.9829 | 0.9348 | 0.9867 | 0.7341 |  |  |  |
| **tractor** | 0.9461 | 0.9697 | 0.9905 | 0.8943 |  |  |  |

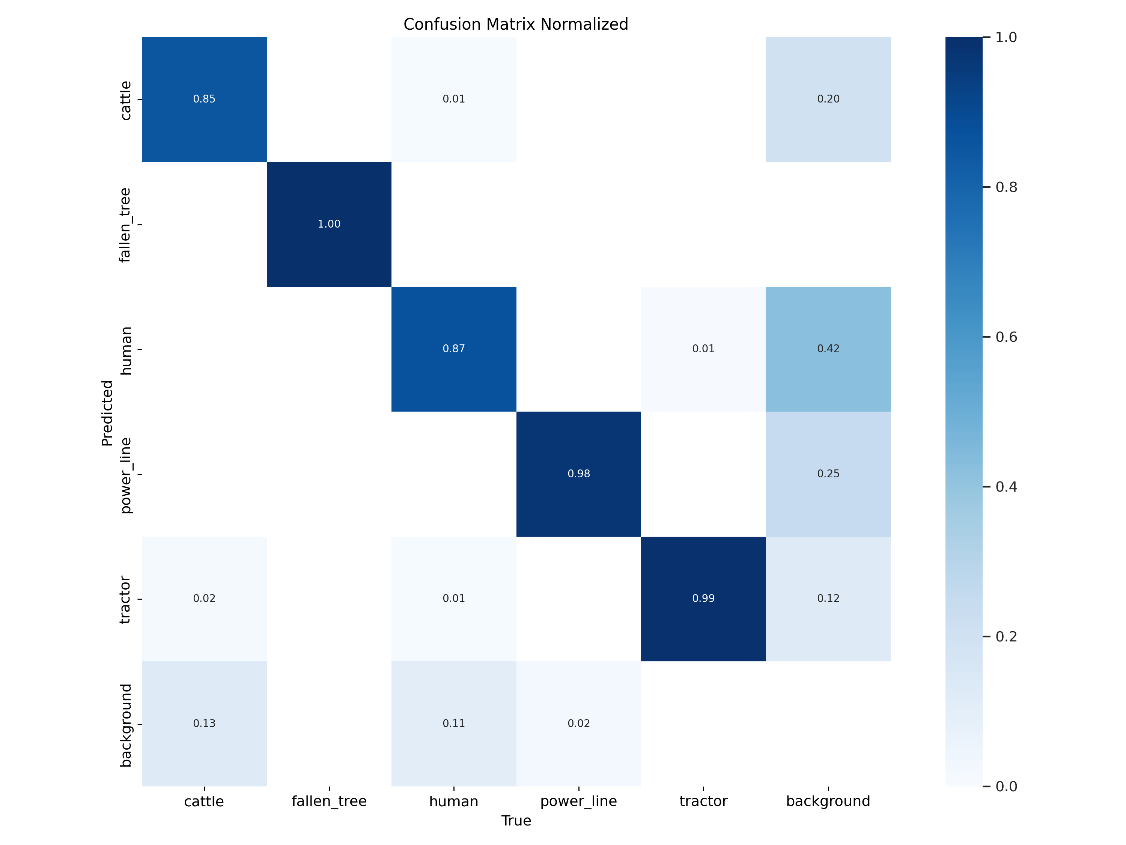
Таблица 4 – Метрики модели yolo8m\_qdynamic

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **mAP50** | **mAP50-95** | **Inference Time per Image (ms)** | **FPS** | **Model Size (MB)** |
| **all** | 0.928 | 0.899 | 0.939 | 0.782 | 1750.62 | 0.571 | 25.04 |
| **cattle** | 0.8582 | 0.8364 | 0.8171 | 0.6437 |  |  |  |
| **fallen\_tree** | 0.9492 | 1.0000 | 0.9950 | 0.9501 |  |  |  |
| **human** | 0.9486 | 0.8247 | 0.9202 | 0.7071 |  |  |  |
| **power\_line** | 0.9513 | 0.8491 | 0.9732 | 0.7248 |  |  |  |
| **tractor** | 0.9335 | 0.9848 | 0.9891 | 0.8835 |  |  |  |

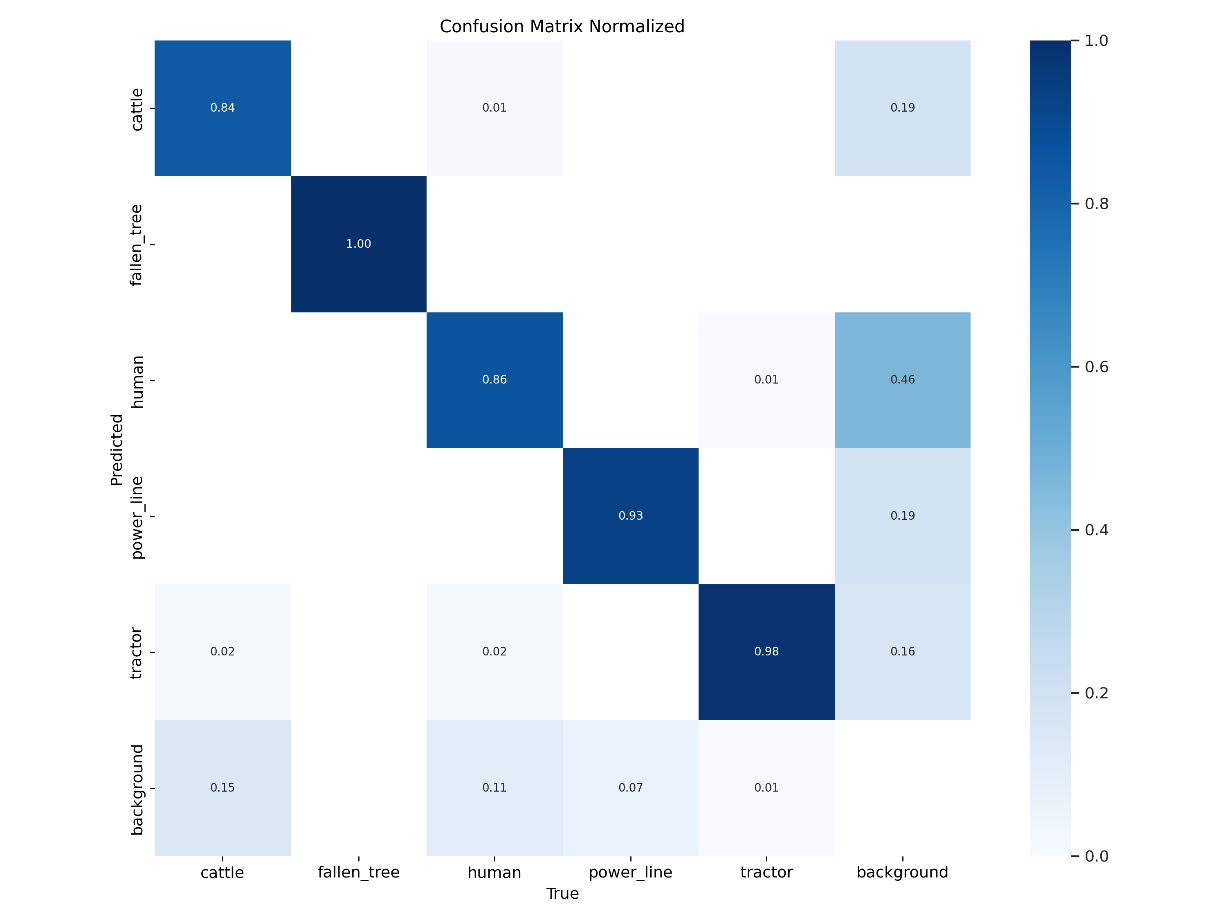
Таблица 5 – Метрики модели yolo8m\_qstatic

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **mAP50** | **mAP50-95** | **Inference Time per Image (ms)** | **FPS** | **Model Size (MB)** |
| **all** | 0.886 | 0.888 | 0.907 | 0.695 | 943.12 | 1.060 | 25.17 |
| **cattle** | 0.8199 | 0.8545 | 0.7986 | 0.5658 |  |  |  |
| **fallen\_tree** | 0.9602 | 1.0000 | 0.9950 | 0.9383 |  |  |  |
| **human** | 0.9363 | 0.8087 | 0.9000 | 0.6535 |  |  |  |
| **power\_line** | 0.7656 | 0.8043 | 0.8555 | 0.5249 |  |  |  |
| **tractor** | 0.9483 | 0.9722 | 0.9882 | 0.7938 |  |  |  |

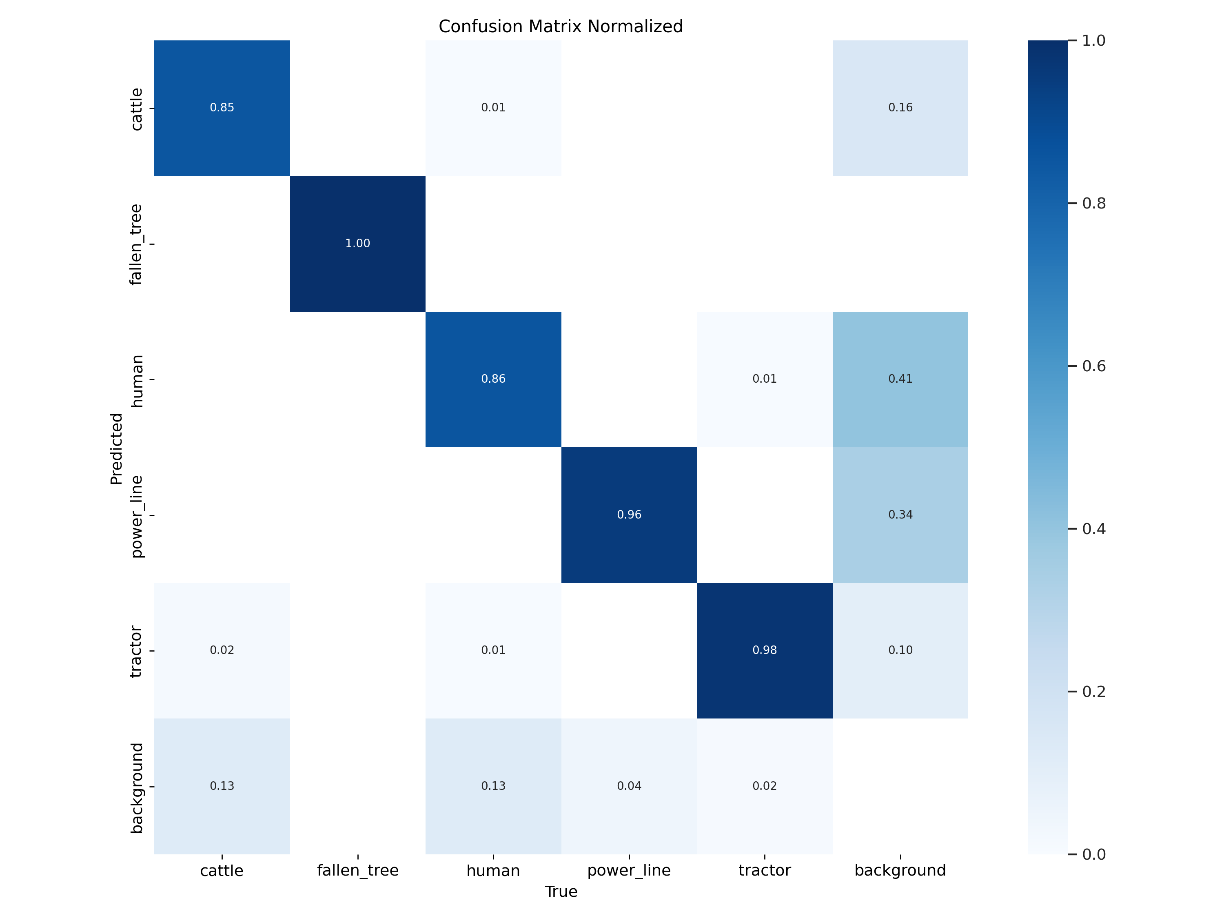
На рисунках 11 – 13 представлены матрицы ошибок по итогам тестирование полученных моделей.



*Рисунок 11. Матрица ошибок модели yolo8m\_fp32*

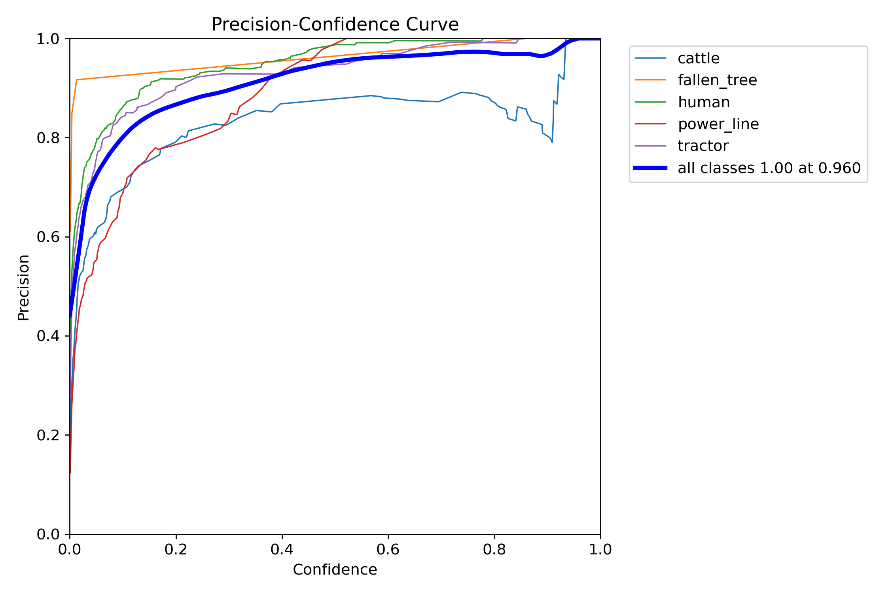
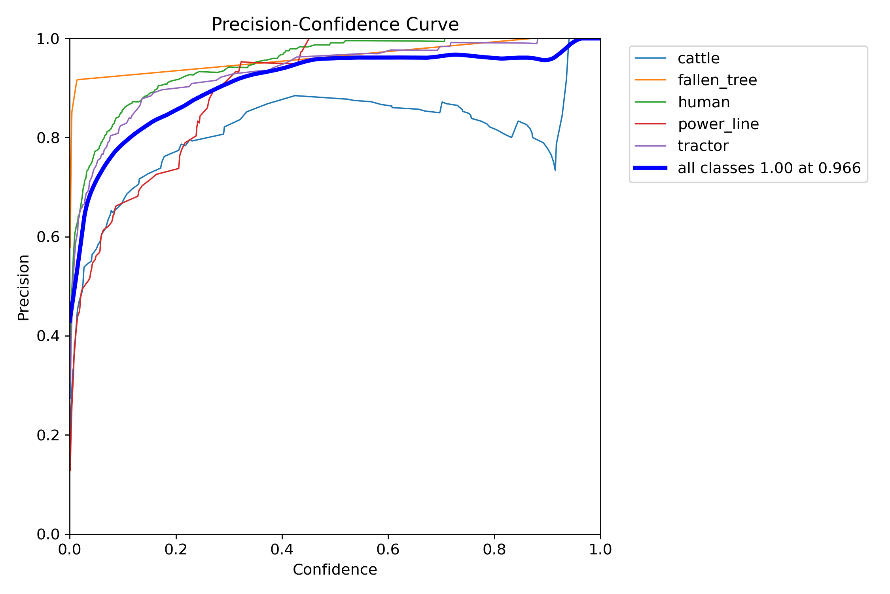
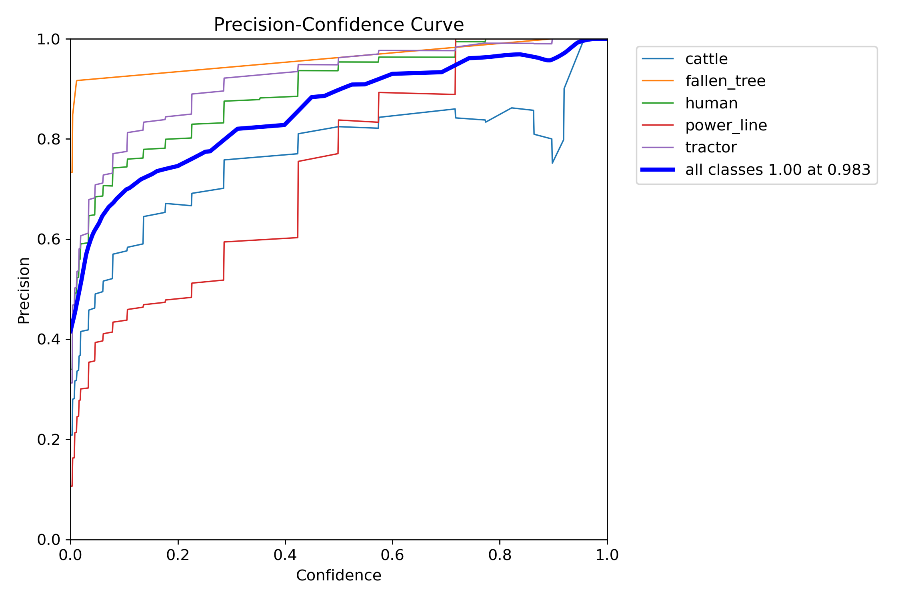


*Рисунок 12. Матрица ошибок модели yolo8m\_qdynamic*



*Рисунок 13. Матрица ошибок модели yolo8m\_qstatic*

На рисунке 14 представлены графики precision\_curve для всех моделей.



*Рисунок 14. Графики precision\_curve для всех моделей (yolo8m\_fp32, yolo8m\_qdynamic, yolo8m\_qstatic сверху вниз)*

Таблица 6. Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| **Модель** | **Метрика** | **До квантизации** | **После квантизации** |
| **YOLOv8m640** | Precision | 0.953 | 0.928 |
| Recall | 0.921 | 0.899 |
| mAP50 | 0.951 | 0.939 |
| mAP50-95 | 0.796 | 0.782 |
| **YOLOv8m800** | Precision | 0.945 | 0.901 |
| Recall | 0.924 | 0.919 |
| mAP50 | 0.953 | 0.930 |
| mAP50-95 | 0.795 | 0.770 |
| **YOLOv8s800** | Precision | 0.912 | 0.954 |
| Recall | 0.923 | 0.906 |
| mAP50 | 0.957 | 0.954 |
| mAP50-95 | 0.773 | 0.771 |
| **YOLOv8n800** | Precision | 0.861 | 0.845 |
| Recall | 0.902 | 0.891 |
| mAP50 | 0.923 | 0.915 |
| mAP50-95 | 0.687 | 0.686 |

Таблица 7. Размер файлов весов моделей

|  |  |  |
| --- | --- | --- |
| Модель | До квантизации (КБ) | После квантизации (КБ) |
| YOLOv8m640 | 101 428 | 25 645 |
| YOLOv8m800 | 101 456 | 25 737 |
| YOLOv8s800 | 43 851 | 11 302 |
| YOLOv8n800 | 12 096 | 1. 357 |

**Изменение метрик**: Квантизация привела к небольшому снижению значений метрик, таких как Precision, Recall, mAP50 и mAP50-95. Среднее снижение составило около 1-2%, что является приемлемым компромиссом для большинства приложений, где важнее компактность модели.

**Уменьшение размера модели**: Уменьшение размеров моделей на 75% делает их особенно подходящими для использования на компактных платах, содержащих на борту все компоненты полноценного компьютера.

**Преимущества onnxruntime**: Использование onnxruntime позволило добиться простоты реализации и минимальных временных затрат на процесс квантизации. Универсальность формата ONNX также облегчила последующее тестирование моделей.