Заключение

Проведённое исследование позволило всесторонне рассмотреть современные подходы к прогнозированию спортивных событий и выявить наиболее эффективные методы анализа футбольных матчей. Особое внимание было уделено сравнительному изучению статистических моделей, алгоритмов машинного обучения и нейронных сетей, что позволило сформировать целостное представление о возможностях каждого класса методов и определить области их практического применения. Анализ показал, что традиционные статистические подходы сохраняют ценность благодаря своей интерпретируемости, однако в ряде случаев их предсказательная сила ограничивается линейностью используемых моделей и устойчивостью параметров.

Наиболее высокие результаты продемонстрировали алгоритмы машинного обучения – прежде всего ансамблевые методы градиентного бустинга и Random Forest, которые обладают способностью выявлять сложные зависимости и обеспечивать устойчивость при наличии шумов в данных. Применение нейронных сетей позволило дополнить картину: модели глубокого обучения продемонстрировали высокую точность при анализе нелинейных и слабо структурированных признаков, особенно в тех случаях, когда необходимо учитывать динамику формы команды и временные закономерности.

Практическим результатом исследования стало создание интеллектуальной программной системы, включающей инструменты обработки данных, построения моделей и визуализации результатов. Такая система обеспечивает удобный и интуитивно понятный механизм работы с массивами спортивной статистики и может использоваться как в учебных, так и в прикладных задачах. Возможность интеграции различных моделей, адаптивность архитектуры и реализованный механизм сравнения прогнозов делают разработанную систему гибким инструментом для анализа широкого спектра спортивных данных.

Таким образом, представленный в исследовании подход подтверждает высокую эффективность использования современных методов машинного обучения и нейросетевого моделирования при прогнозировании спортивных событий. Перспективы дальнейшего развития включают расширение набора данных, интеграцию текстовой аналитики для учёта новостного контекста, применение рекуррентных и трансформерных моделей для анализа временных последовательностей, а также создание гибридных систем, объединяющих несколько типов алгоритмов. Всё это будет способствовать повышению точности прогнозов и укреплению роли интеллектуальных технологий в области спортивной аналитики.

Список литературы

1. Андреев К. С. Компьютерное зрение и машинное обучение. СПб.: Питер, 2020.

2. Брук Т. Статистические методы в спортивной аналитике. М.: Спорт и наука, 2020.

3. Гудфеллоу Я., Бенджио И., Курвиль А. Глубокое обучение. М.: ДМК Пресс, 2018.

4. Половников Е. П. Технологии машинного обучения в автоматизированных системах. М.: Академия, 2021.

5. Bukreiev D. O. et al. Features of the use of software and hardware of the educational process in the conditions of blended learning // AET 2020-Symposium on Advances in Educational Technology. Technology (AET 2020). SCITEPRESS, 2022. №. 2. С. 236-244.

6. Bukreiev D. Neuro-network technologies as a mean for creating individualization conditions for students learning //SHS Web of Conferences. EDP Sciences, 2020. Т. 75. С. 04013.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МОДЕЛЕЙ YOLOV8–YOLO11 В ЗАДАЧАХ ОБНАРУЖЕНИЯ ОБЪЕКТОВ В РЕАЛЬНОМ ВРЕМЕНИ

Корж А. А., Олейник Н. П.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: alex.korhz13@mail.ru, nata.oleynik.2014@mail.ru,

Постановка проблемы. Отсутствие сопоставимых по методике сравнений YOLOv8–YOLO11 приводит к выбору моделей «по привычке» или по разрозненным публикациям, где различаются датасеты, размер входа и правила измерения задержки. Для практических систем это означает либо избыточные требования к вычислительным ресурсам, либо неприемлемое снижение качества детекции при переносе на целевую сцену.

Материалы и методы исследования

История семейства YOLO начинается с работы J. Redmon и соавт. (2015) [1], где предложен одноэтапный подход к детекции объектов. В последующих версиях развивались идеи ускорения и повышения точности; например, для YOLOv7 показано, что переработка backbone/neck и обучение с учётом задач реального времени позволяют сохранить высокую скорость при конкурентной точности [2].

Ultralytics в YOLOv8 перешли к anchor-free предсказаниям и унифицированному фреймворку задач, оставив NMS как обязательную часть конвейера [3].

В YOLOv9 введены GELAN и механизм PGI, рассчитанный на улучшение градиентного потока на этапе обучения [4]. Работа по YOLOv10 (Tsinghua University) предложила end-to-end детекцию без NMS через согласованное двойное назначение (one-to-many и one-to-one) и более лёгкие вычислительные блоки [5].

В YOLO11 Ultralytics сфокусировались на снижении параметрической сложности и уско-

рении CPU-инференса, заменив базовые блоки и расширив внимание (C2PSA) [6].

При этом в прикладной литературе преобладают оценки отдельных версий на COCO; сравнения по одинаковым метрикам задержки и потребления ресурсов для edge/CPU встречаются ограниченно.

Цель исследования – выполнить структурированный анализ YOLOv8–YOLO11, сопоставив архитектурные изменения и их влияние на mAP, задержку инференса и число параметров, и дать практические рекомендации выбора модели для типовых сценариев развёртывания.

Результаты исследования и их обсуждение

Все рассматриваемые версии (табл. 1) сохраняют трёхкомпонентную схему (backbone–neck–head), однако различаются тем, как организован поток признаков и как формируется обучающий сигнал. YOLOv8 использует блок C2f и SPPF, работает в anchor-free постановке и требует NMS. YOLOv9 вводит GELAN и PGI: вспомогательный путь для устойчивых градиентов используется на обучении и удаляется на инференсе, сохраняя классическую постобработку. YOLOv10 переносит часть сложности из постобработки в обучение: схема one-to-one позволяет выполнять вывод без NMS, что уменьшает задержку и упрощает конвейер. YOLO11 развивает линию повышения эффективности за счёт более компактных блоков (C3k2) и внимания (C2PSA), что важно для CPU и энергоограниченных систем.

По опубликованным бенчмаркам [4,5], различия точности между поколениями после YOLOv9 сравнительно невелики, тогда как различия по задержке и цене вывода заметнее. В частности, для малых конфигураций заявлено, что YOLOv10-S достигает латентности порядка 2,49 мс с учётом всей постобработки (без отдельного NMS), тогда как для YOLOv8-S приводится около 7,07 мс с учётом NMS. Для CPU-сценариев YOLO11 демонстрирует ускорение порядка 30% для nano-модели, что критично при работе на видеосерверах без GPU и на edge-устройствах.

В прикладных областях выбор версии (табл. 2) определяется не только mAP, но и распределением задержки, памятью и устойчивостью. В автономном вождении важны стабильные 30 FPS и поведение при тумане, дожде и ночной сцене; отсутствие NMS в YOLOv10 снижает вариативность задержки, а модели YOLO11 среднего размера целесообразны, когда требуется более высокая точность при наличии GPU. В видеонаблюдении приоритетом становится непрерывная работа 24/7 и ограничение ложных тревог; компактные модели YOLO11n подходят для CPU-развёртывания и малой памяти, а YOLOv10s оправдана при наличии GPU и жёстком ограничении на задержку. В промышленном контроле качества критичны мелкие дефекты и высокая частота кадров; здесь полезны более точные конфигурации и, при необходимости, варианты с сегментацией, при этом ускорение вывода уменьшает нагрузку на линию инспекции.

Личный вклад авторов заключается в систематизации различий между версиями по единой схеме сравнения (архитектура, обучение, постобработка, ресурсы) и в формировании матрицы выбора для типовых сценариев. Представленная структура анализа может использоваться как шаблон для проверки новых выпусков YOLO, когда скорость обновления превышает скорость появления полноценных сравнительных публикаций.

Для прикладных внедрений требуется расширять набор проверок за пределы стандартных отчётов COCO: оценивать устойчивость к шуму, сжатию и артефактам видеопотока, проверять перенос на узкие доменные датасеты (аэрофото, подводная съёмка, медицинские изображения) с фиксированным бюджетом дообучения, а также измерять влияние INT8-квантования и дистилляции на качество и задержку на конкретном CPU/NPU.

Таблица 1

Сравнение ключевых компонентов YOLOv8–YOLO11

Компонент	YOLOv8	YOLOv9	YOLOv10	YOLO11
Backbone block	C2f	RepNCSPELAN4 (GELAN)	C2f / C2fCIB (rank-guided)	C3k2
Downsampling	Conv (3×3, stride=2)	ADown (pooling-based)	SCDown (decoupled)	Conv (3×3, stride=2)
Neck attention	SPPF	SPPELAN	SPPF + PSA	SPPF + C2PSA
Необходимость NMS	Требуется	Требуется	Не требуется	Требуется (ускорена)
Эффективность параметров	Базовая	Выше (меньше параметров)	Высокая	Высокая
Особенности вывода	Стандартный конвейер	PGI только на обучении	End-to-end one-to-one	Ускорение CPU, DWConv

Таблица 2

Рекомендации по выбору версии для типовых сценариев

Сценарий	Рекомендуемая версия	Основание выбора
Edge-устройства (Raspberry Pi, Jetson Nano)	YOLO11n	Малая модель, ускоренный CPU-инференс, низкое потребление памяти.
Видеонаблюдение 24/7	YOLO11n/s или YOLOv10s	YOLO11 – экономия ресурсов на CPU; YOLOv10 – минимальная задержка при наличии GPU и отсутствии NMS.
Автономное вождение	YOLOv10m или YOLO11m	YOLOv10 – более предсказуемая задержка без NMS; YOLO11m – выше точность при наличии вычислительного бюджета.
Промышленный контроль качества	YOLO11m (и варианты с сегментацией) / YOLOv10m	Нужны высокая точность по мелким объектам и высокая частота кадров; выбор зависит от доступного ускорителя.
Облачный сервис	YOLO11x или YOLOv10x	Приоритет точности; задержка на GPU приемлема, выбирается по нагрузке и стоимости вычислений.

Заключение

Отдельный интерес представляет сравнение не только средних значений, но и распределений латентности, поскольку именно предсказуемость задержки определяет пригодность детектора для реального времени.

Сопоставление YOLOv8–YOLO11 показывает, что эволюция после 2023 года в значительной мере направлена на снижение задержки и стоимости вывода при близких уровнях точности: YOLOv9 усиливает обучение через PGI, YOLOv10 устраняет необходимость NMS и тем самым уменьшает задержку, а YOLO11 сокращает число параметров и ускоряет CPU-инференс за счёт замены базовых блоков и расширения внимания, на практике это позволяет обоснованно выбирать версию под ограничения конкретного стенда (CPU/GPU, память, требуемый FPS) и сокращает риск неоправданных затрат при развёртывании, при этом дальнейшая проверка должна включать доменную переносимость, устойчивость к артефактам видеопотока и эффект квантования на целевом оборудовании.

Список литературы

1. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). P. 779-788. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf (date accessed: 18.12.2025).

2. YOLOv7: Trainable state-of-the-art object detector. arXiv preprint arXiv:2207.02696. URL: https://arxiv.org/abs/2207.02696 (date accessed: 18.12.2025).

3. YOLOv8: A state-of-the-art real-time object detector. Ultralytics GitHub Repository. URL: https://github.com/ultralytics/ultralytics (date accessed: 15.12.2025).

4. YOLOv9: Learning what you want to learn using programmable gradient information. arXiv preprint arXiv:2402.13616. URL: https://arxiv.org/abs/2402.13616 (date accessed: 20.12.2025).

5. YOLOv10: Real-time end-to-end object detection. arXiv preprint arXiv:2405.14458. URL: https://arxiv.org/abs/2405.14458 (date accessed: 18.12.2025).

6. YOLO11: Advanced Real-time Object Detection. Ultralytics Documentation. URL: https://docs.ultralytics.com/ (date accessed: 16.12.2025).

ОБЗОР РАЗВИТИЯ И ТЕХНОЛОГИЧЕСКИХ ВЫЗОВОВ ПЛАТФОРМ ДЛЯ ПРОВЕДЕНИЯ СОРЕВНОВАНИЙ ПО СПОРТИВНОМУ ПРОГРАММИРОВАНИЮ

Креминь А. А., Мозговенко А. А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: alex.kremi@mail.ru, ya@amozgovenko.ru

Постановка проблемы. Рост массовых онлайн-соревнований по спортивному программированию, характеризующихся одновременной подачей тысяч решений, многоязычной поддержкой и разнообразием форматов задач, обнажил архитектурные ограничения существующих платформ автоматизированной проверки. Практика их эксплуатации показывает, что различия в способах изоляции исполнения кода, организации очередей компиляции и запуска, а также в механизмах масштабирования вычислительных ресурсов приводят к нестабильному времени проверки, трудновоспроизводимым результатам и рискам нарушения безопасности. Отсутствие систематизированного анализа архитектурных решений, подтверждённого сопоставлением реальных платформ и их инженерных компромиссов, затрудняет выработку обоснованных рекомендаций по проектированию устойчивых и детерминированных соревновательных систем, что формирует центральную проблему настоящего исследования.

Материалы и методы исследования

Анализ публикаций 2020–2025 годов демонстрирует, что исследовательский и инженерный фокус в области платформ для спортивного программирования сместился от описания базовой функциональности онлайн-проверки к решению прикладных задач масштабируемости, безопасной изоляции исполнения и воспроизводимости