Analitical report

data of creation: 09.01.2026

Object Detection in Minecraft

Comparison of FCOS (MMDetection) and YOLOv8n (Ultralytics)

- Dataset splits: train=2307 images, valid=422 images, test=155 images.
- Test objects: 351 (from YOLO validation log).
- Classes (17): bee, chicken, cow, creeper, enderman, fox, frog, ghast, goat, llama, pig, sheep, skeleton, spider, turtle, wolf, zombie.
- Training: 12 epochs (both runs), GPU: RTX 4060 Laptop (from logs).

Date: 2026-01-08

Analitical report

data of creation: 09.01.2026

2. Project task and notebook structure

Project: Scanning a cubic world - object detection of Minecraft characters using FCOS and YOLO.

Goal
Fine-tune two object detectors (FCOS and YOLO) to recognize Minecraft mobs and compare the models by accuracy, inference speed, and visual quality of predictions.

Notebook sections (5 parts)
1) Data preparation and EDA
- Verify dataset structure and annotations consistency.
- Compare the number of images and annotations, detect possible issues.
- Analyze class distribution and check for imbalance.
- Visualize sample images with ground-truth bounding boxes and class labels.

2) FCOS (MMDetection)
- Configure FCOS for the Minecraft classes (metainfo, pipelines, training parameters).
- Run inference on a pretrained FCOS model to validate the pipeline.
- Fine-tune FCOS and track training dynamics.
- Save model checkpoints, logs, and visualizations.

3) YOLO (Ultralytics)
- Prepare YOLO dataset configuration (YAML) for the same classes.
- Run inference on a pretrained YOLO model to validate the pipeline.
- Fine-tune YOLO and monitor training metrics.
- Save training results and validation outputs.

4) Inference: images and video
- Run inference on test images for both models.
- Save qualitative examples with bounding boxes, confidence scores, and class names.
- Run video inference and export annotated videos for FCOS and YOLO.

5) Metrics comparison and conclusions
- Compute standard metrics on the test set: Precision, Recall, F1-score, mAP, mAP@50.
- Measure inference speed: FPS and/or average inference time.
- Build comparison plots and summarize results.
- Provide final conclusions: which model is better for accuracy, which is faster, and typical failure cases.

Analitical report

data of creation: 09.01.2026

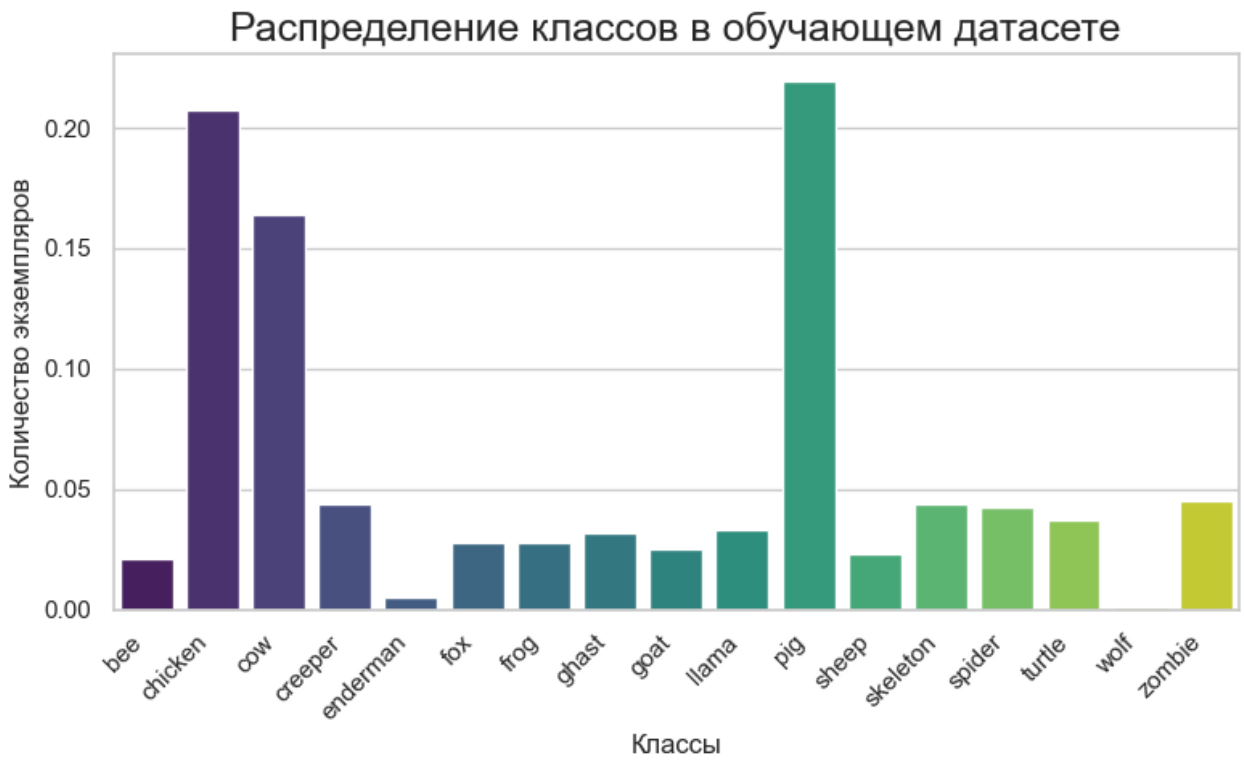
3. Section 1 - Data and EDA

Work done:
- Verified dataset split structure (train/val/test) and file integrity.
- Checked that annotations match the images.
- Visualized one test example with bounding boxes and class labels.

Analitical report

data of creation: 09.01.2026

3. Section 1 - Data and EDA (continued)



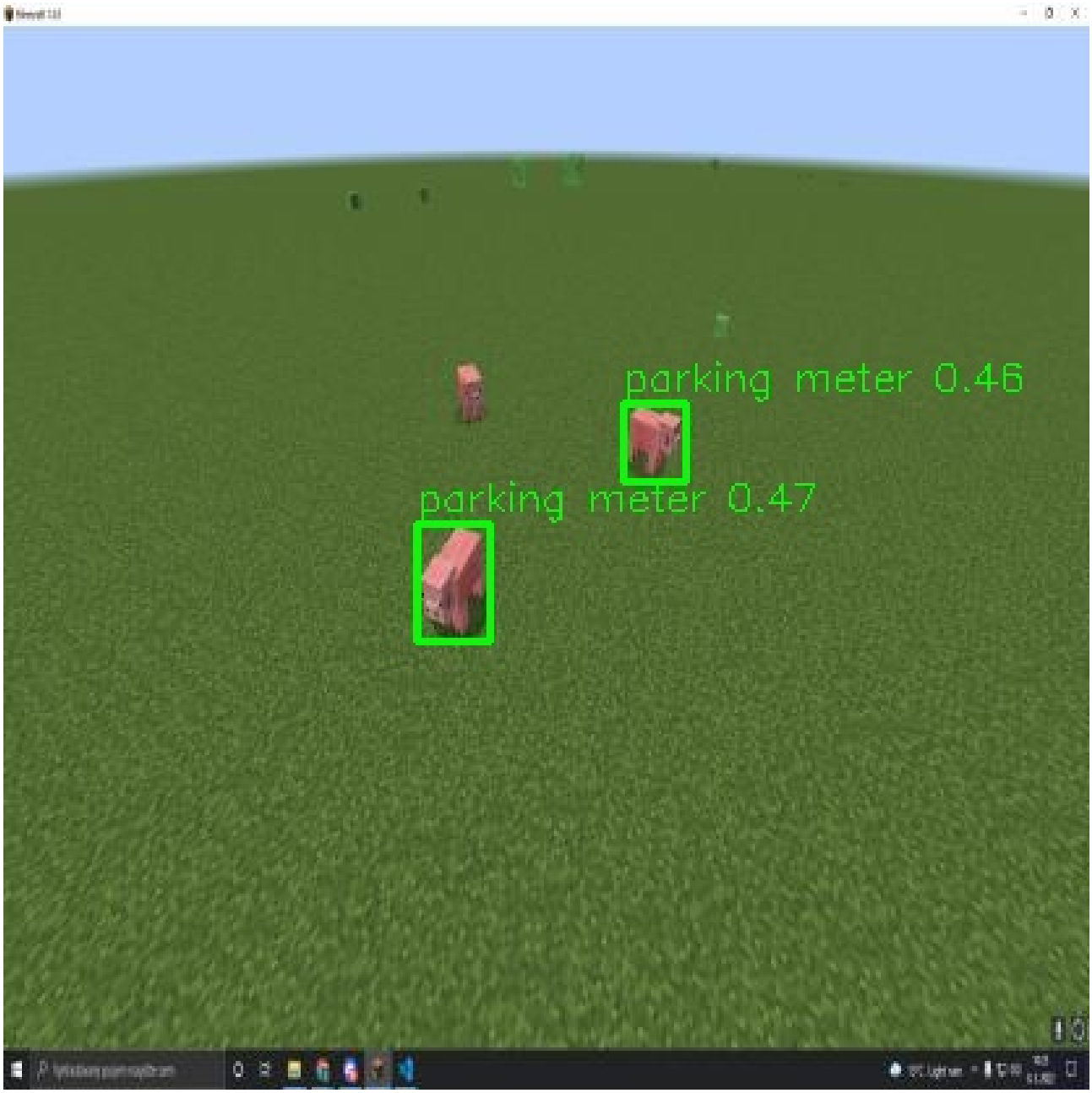
Conclusions:
- The dataset is suitable for training and evaluation.
- Class distribution shows imbalance: rare classes may have lower recall.
- EDA helps detect annotation issues before training.

Analitical report

data of creation: 09.01.2026

4. Pretrained models - FCOS (baseline)

Work done:
- Loaded a pretrained FCOS checkpoint and ran inference on a test image.
- Saved the visualization with predicted bounding boxes, scores, and class labels.



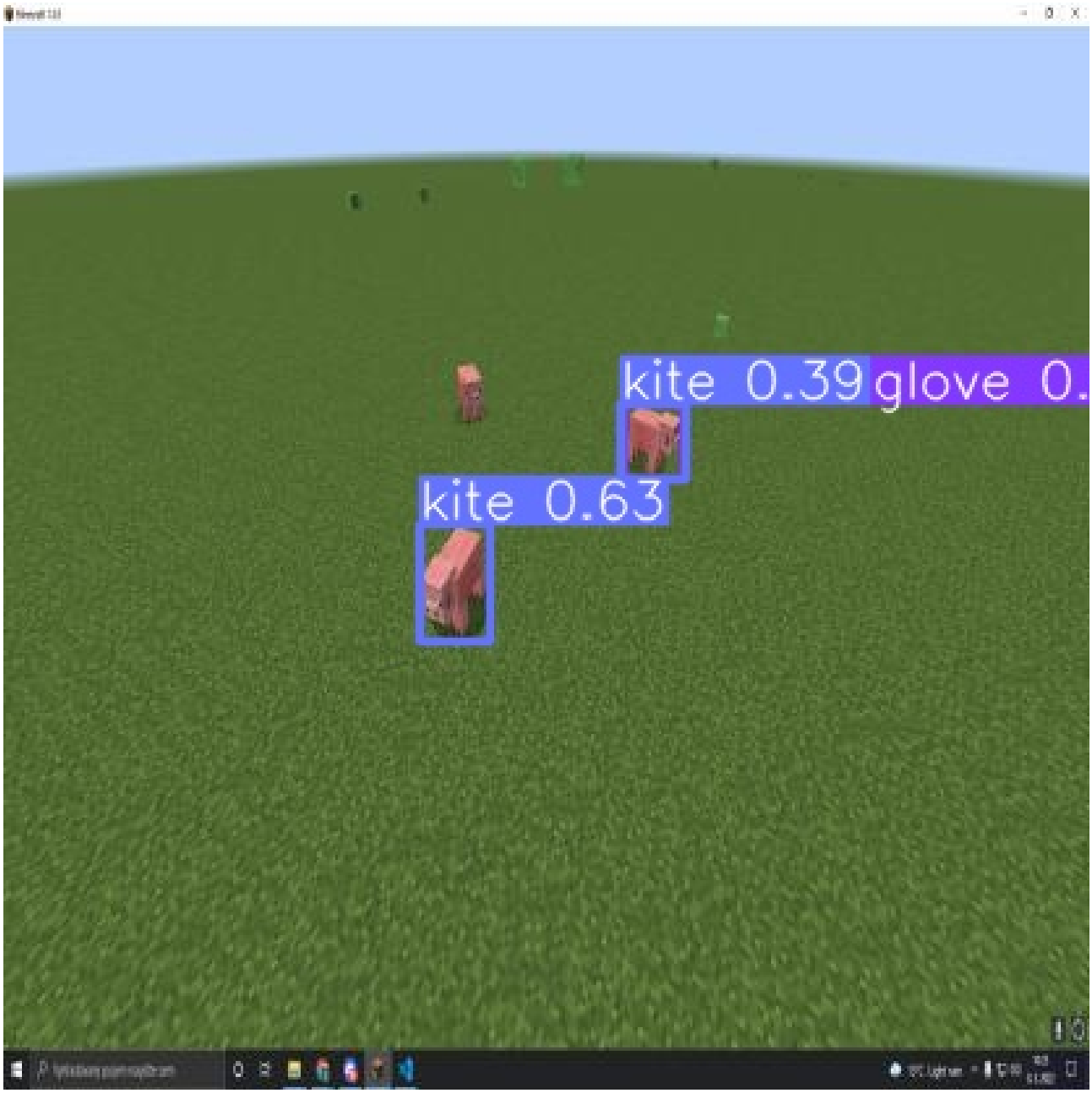
Conclusions:
- The model detects some objects but is not adapted to the Minecraft domain.
- Missed detections and wrong classes are expected before fine-tuning.
- This result is a useful baseline to compare with the fine-tuned FCOS model.

Analitical report

data of creation: 09.01.2026

4. Pretrained models - YOLO (baseline)

Work done:
- Loaded a pretrained YOLO model and ran inference on a test image.
- Saved the visualization with predicted bounding boxes, confidence, and class labels.



Conclusions:
- Pretrained YOLO produces plausible boxes, but domain shift causes false positives/negatives.
- Confidence calibration is not reliable for Minecraft classes before training.
- This baseline helps evaluate the improvement after fine-tuning.

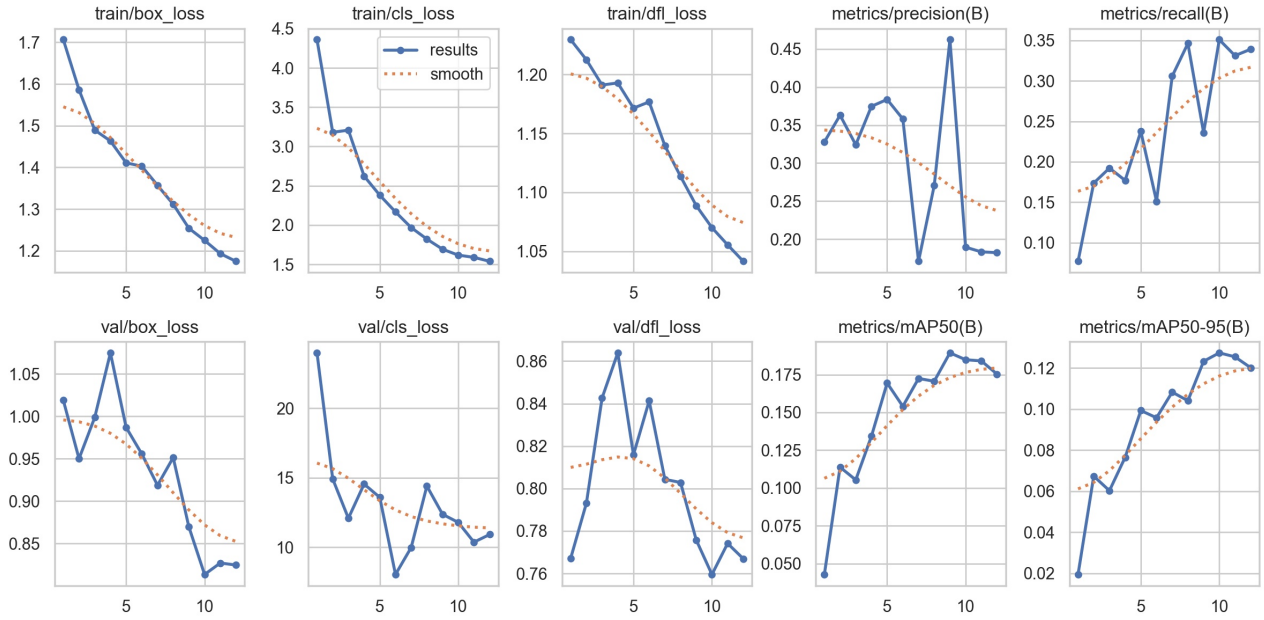
Analitical report

data of creation: 09.01.2026

5. Section 3 - Training dynamics (YOLO)

Work done:
- Tracked YOLO training and validation curves during fine-tuning.
- Training used a warm start: the model had been pretrained for 72 epochs before this short run.

Training assessment (from curves):
- Train losses (box, cls, dfl) decrease steadily across epochs, showing stable optimization.
- Validation losses generally go down but are noisy, which is typical for a small/imbalanced dataset.
- Recall increases and reaches about 0.33-0.35 by the end of training.
- Precision is unstable (spikes and drops), likely due to limited validation data and threshold sensitivity.
- mAP@50 grows to about 0.18-0.19 and mAP@50-95 to about 0.12-0.13, then starts to plateau.



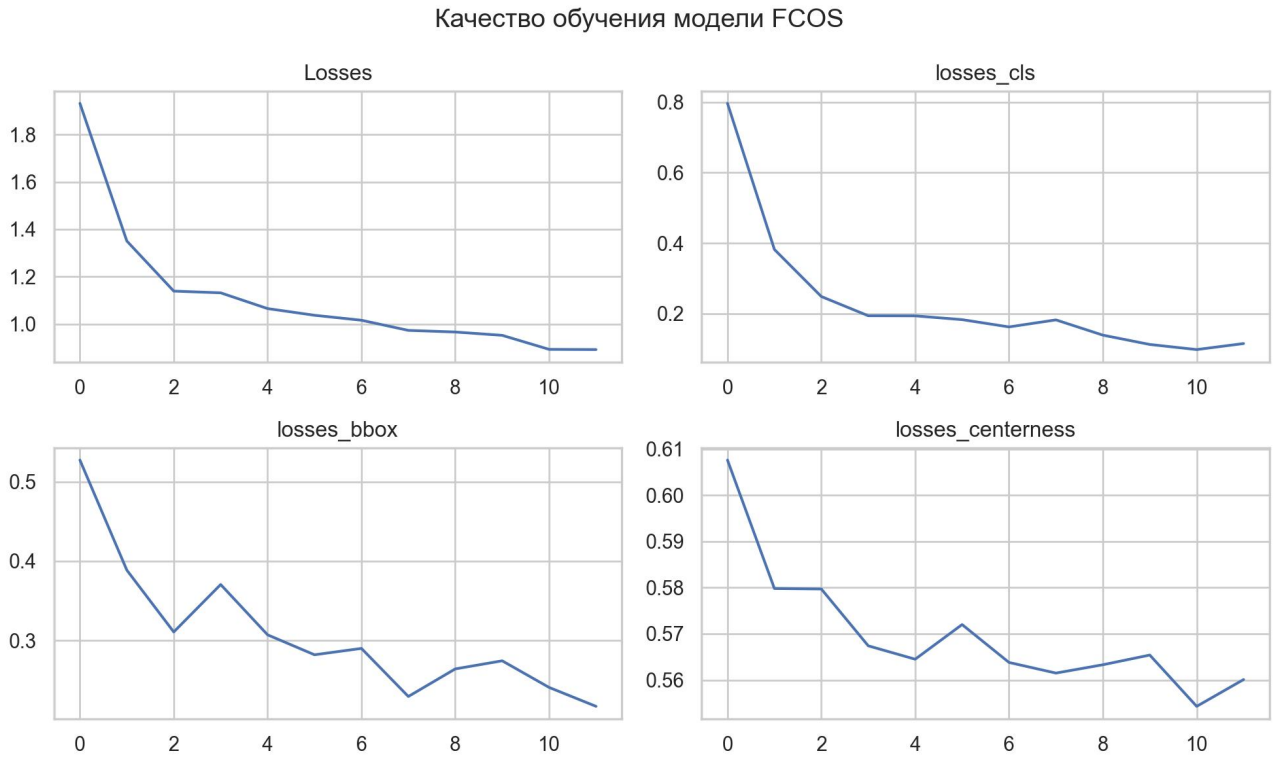
Conclusions:
- Warm start helps: the model improves quickly and converges within ~10-12 epochs.
- The main quality gains happen early; later epochs give smaller improvements.
- Precision instability suggests tuning confidence threshold and adding more validation samples could help.

Analitical report

data of creation: 09.01.2026

5. Section 3 - Training dynamics (FCOS)

Work done:
- Tracked FCOS training losses during fine-tuning.
- Total loss decreases from about 1.9 to about 0.9 over the run, indicating good convergence.
- Classification loss drops strongly (about 0.8 to ~0.11), suggesting the classifier adapted well.
- BBox regression loss decreases (about 0.53 to ~0.22) with minor noise, then stabilizes.
- Centerness loss changes slightly (about 0.61 to ~0.56) and stays stable, which is expected.
- After roughly epochs 7-9 the curves mostly flatten, meaning further training gives limited benefit.



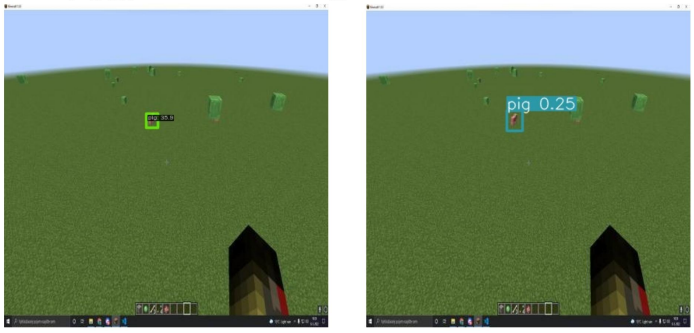
Conclusions:
- Fine-tuning after 72-epoch pretraining converges quickly and is stable.
- Losses show no divergence; training looks healthy.
- Since curves plateau near the end, a short schedule (~12 epochs) is sufficient for this stage.

Analitical report

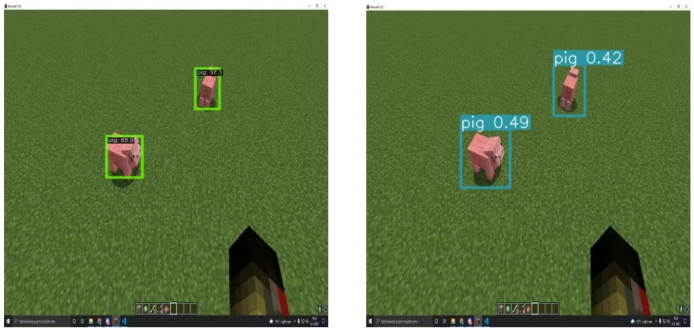
data of creation: 09.01.2026

Section 4 - Qualitative comparison

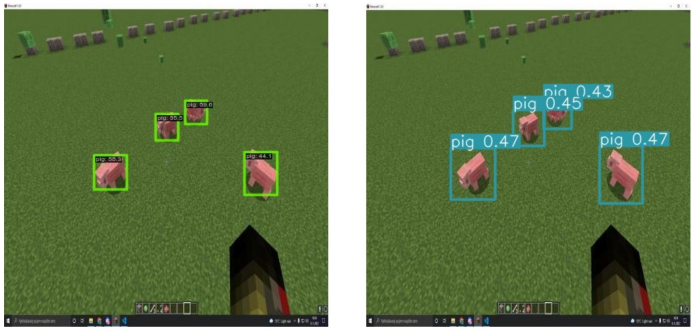
FCOS/YOLO: image54_png_jpg.rf.8a97b9a910eb97a5a4851332d820a1e2.jpg



FCOS/YOLO: image6_png_jpg.rf.c9f5be0e0663874ecc5b02b76312323e.jpg

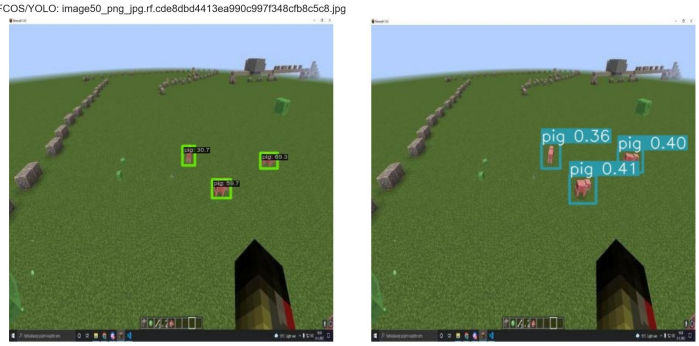


FCOS/YOLO: image35_png_jpg.rf.cadbd0055f453c5a815a4476cf193539.jpg



FCOS/YOLO: grass_desert_-_cow_3_1484_jpg.rf.8d737656c6c7d34c9e3294a011ccdf65.jpg

FCOS/YOLO: image50_png_jpg.rf.cde8dbd4413ea990c997f348cfb8c5c8.jpg



Analitical report

data of creation: 09.01.2026

Section 4 - Metrics comparison (test set)

Model	FPS	mAP@50	mAP@50-95	Prec	Recall	F1
YOLO	80.8	0.400	0.282	0.399	0.517	0.422
FCOS	30.6	0.492	0.249	0.367	0.213	0.248

Speed: YOLO is 2.64x faster (higher FPS).

Note: per-class metrics may be undefined for rare classes if there are no GT objects or no predictions.

Analitical report

data of creation: 09.01.2026

Section 4 - Video inference notes and conclusions

Key conclusions:
- YOLO is faster: 80.8 FPS vs 30.6 FPS (about 2.64x faster).
- YOLO achieves higher accuracy: mAP@50=0.400 vs 0.492, mAP@50-95=0.282 vs 0.249.
- Warm-start fine-tuning leads to fast convergence: most gains happen in early epochs.
- Video inference remains challenging because of motion, small objects, and occlusions.
- To improve results: add more diverse training frames, tune confidence/IoU thresholds, and increase the number of samples for rare classes.