EuroCC4SEE Seminar Series – “5 Beats of Intelligence: AI Meets Diverse Domains”

AI Meets Precision Agriculture

Powered by
Arizona State University®

Tech members at AI-HPC department

PhD Tomo Popović (Team Lead, associate professor)

PhD Luka Filipović

HPC expert

MSc Stevan Šandi

AI and Brain (EEG)

MSc Stevan Čakić

GenAI/CV and HPC

MSc Ivan Jovović

EdgeAI and CV

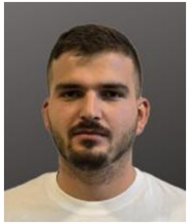
MSc Dejan Babić

AI and Medicine

MSc Zoja Šćekić

AI and Energy

MSc Elvis Taruh



EdgeAI and HPC

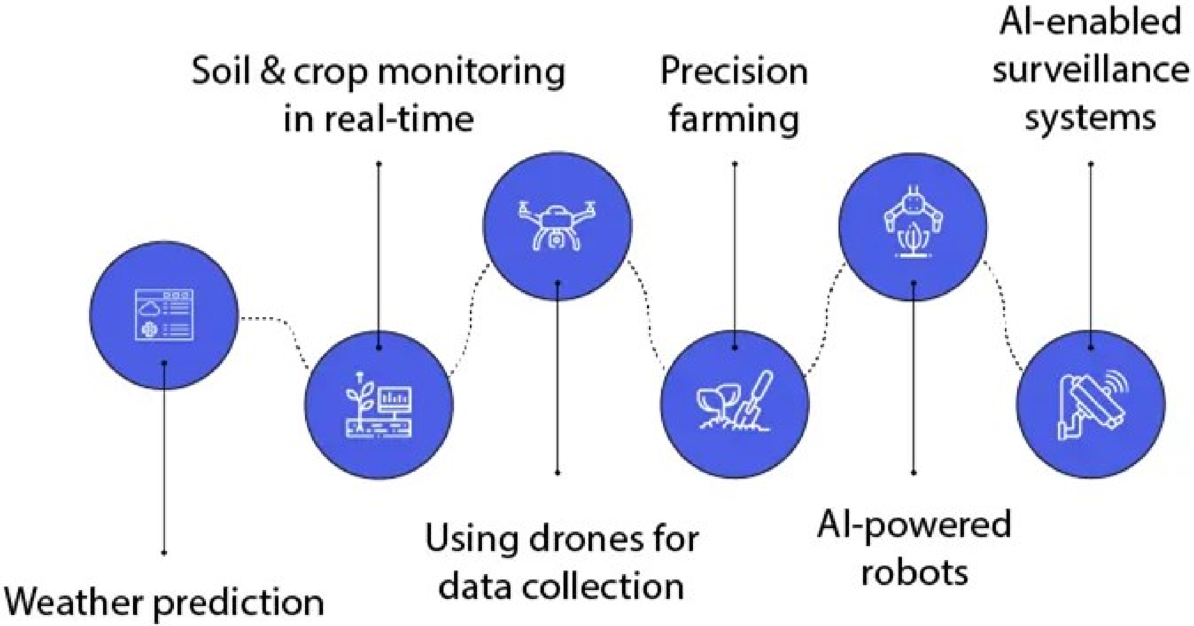
MSc Arnad Lekić

AI and Education

AIMHIGH

Developing Edge AI Computer Vision Kit for Smart Poultry Farms Using Deep Learning and High-performance Computing

Artificial intelligence and agriculture



Reference: *Implementation of AI in Agriculture, https://appinventiv.com/blog/benefit-of-ai-in-agriculture/*

The problem and questions

- How do we address the **growing demand for food** and animal protein?
- How do we **scale and optimize animal farming process** to answer the needs of the global meat market?
- Can we use AI advancement to contribute in **optimization** of poultry farming?

Main focus

- Research was focused on poultry farms and the following:
  - **Disease** outbreaks among chickens;
  - **Dead chickens** that need to be removed;
  - All the basic life needs: feed, water, lighting, air conditions;
- All of that while ensuring animal well-being and to make the life for farmers easier.

Research goals

- Create poultry farm camera **sensors** based on deep learning and IoT edge devices.
- The use of HPC to support development of **new smart IoT sensors** for poultry farms, based on Edge AI/DL computer vision and faster process of development AI models.
- Enabling deployment on the edge devices equipped with camera sensors.
- Model integration with a web application.

Research key tasks

- Object detection
  - Chicken/poultry counting (density estimation);
  - Detection of dead chickens;
- Object segmentation
  - Estimation of the weight of the chicken;
- AutoML
  - Hyperparameters optimization (HPO);
- Generative AI
  - Reduce the need for real data;

Implementation phases

1 **DESIGN:** Requirements Analysis and Experiment Design

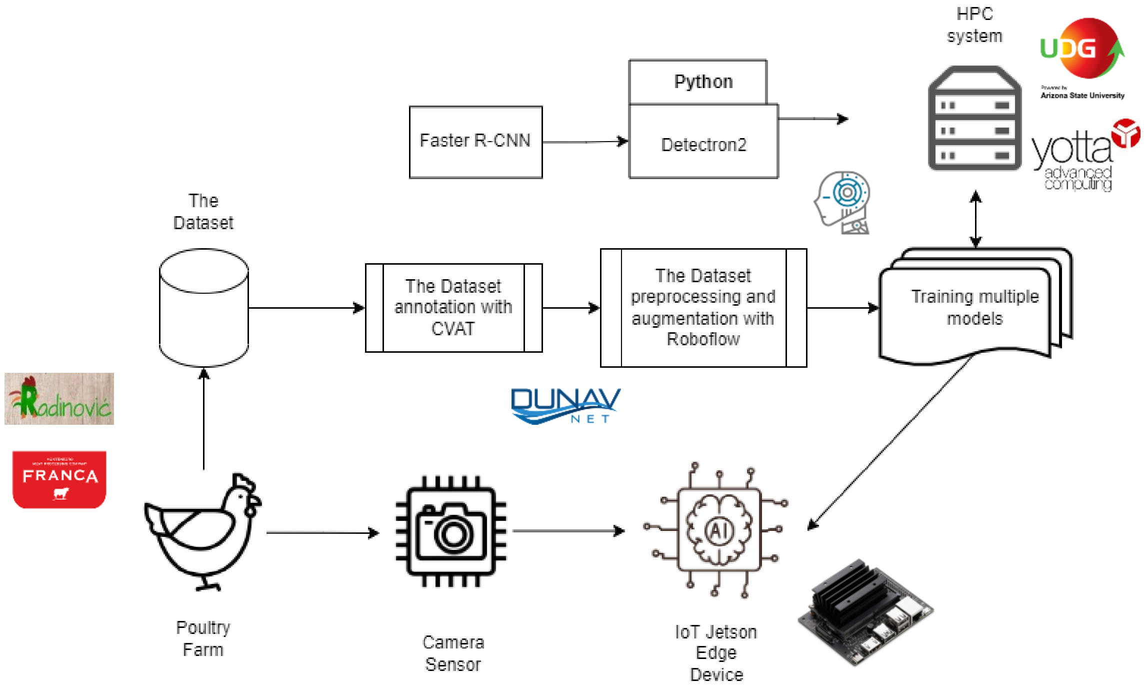
2 **LEARN:** Development of Prediction Models Using HPC

3 **INTEGRATE:** AI-Based Edge Camera Nodes and Decision Support

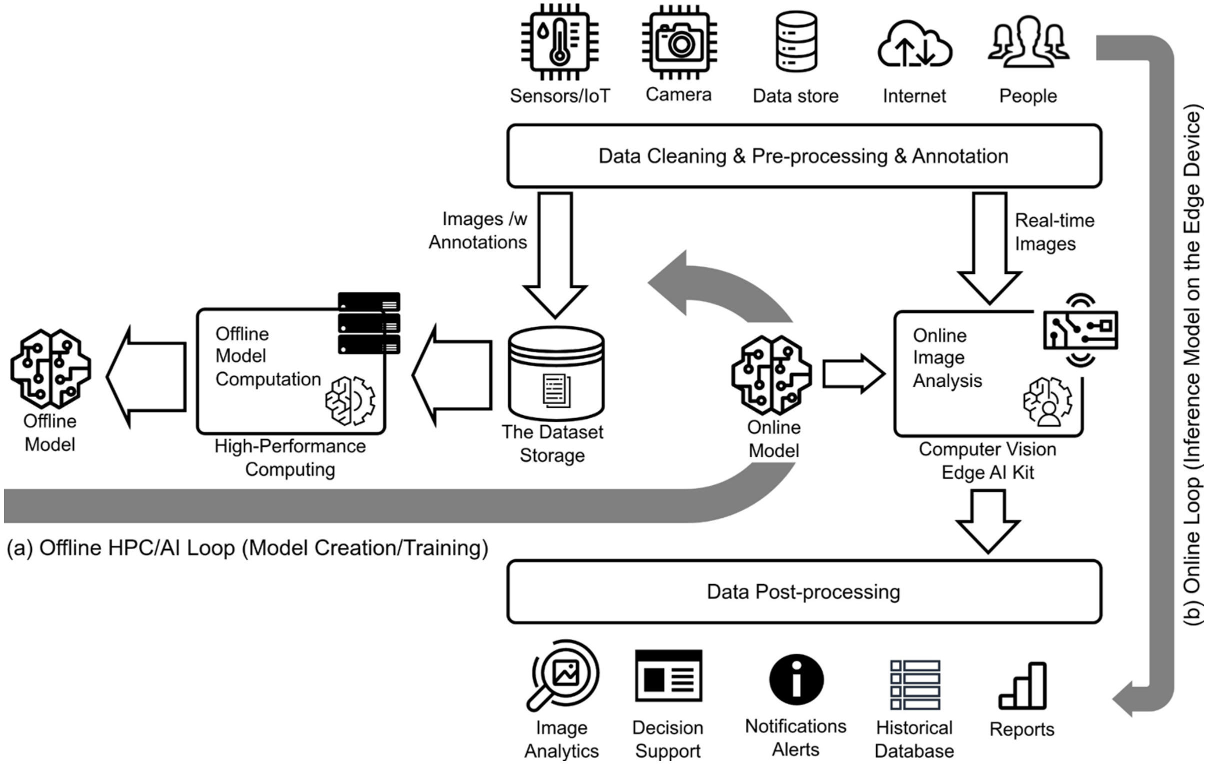
4 **VALIDATE:** Overall Verification and Validation

5 Project Management, Dissemination and Exploitation

Architecture (simplified)



Architecture (2)

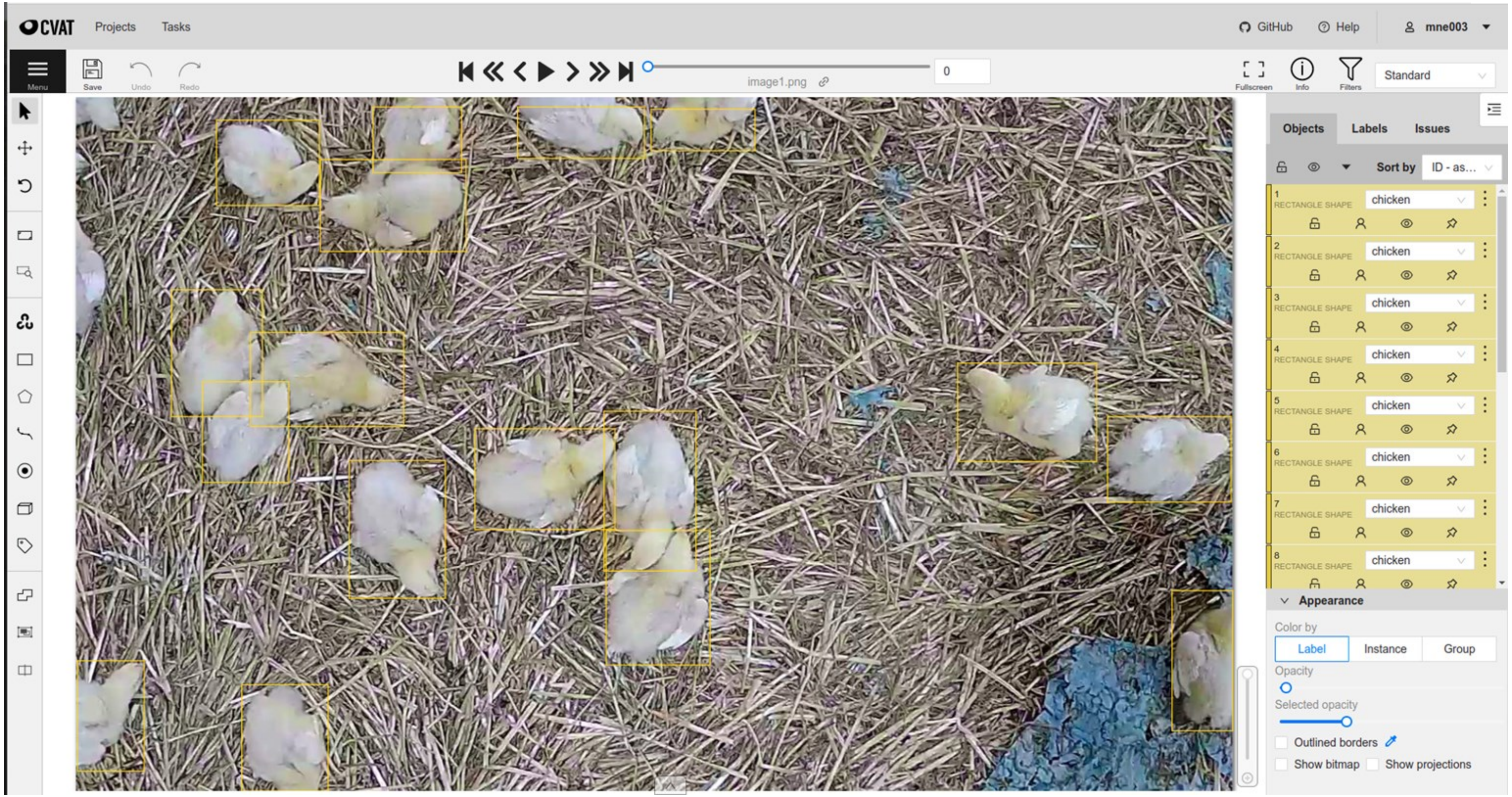


Reference: *Cakic, S.; Popovic, T.; Krco, S.; Nedic, D.; Babic, D.; Jovovic, I. Developing Edge AI Computer Vision for Smart Poultry Farms Using Deep Learning and HPC. Sensors 2023, 23*

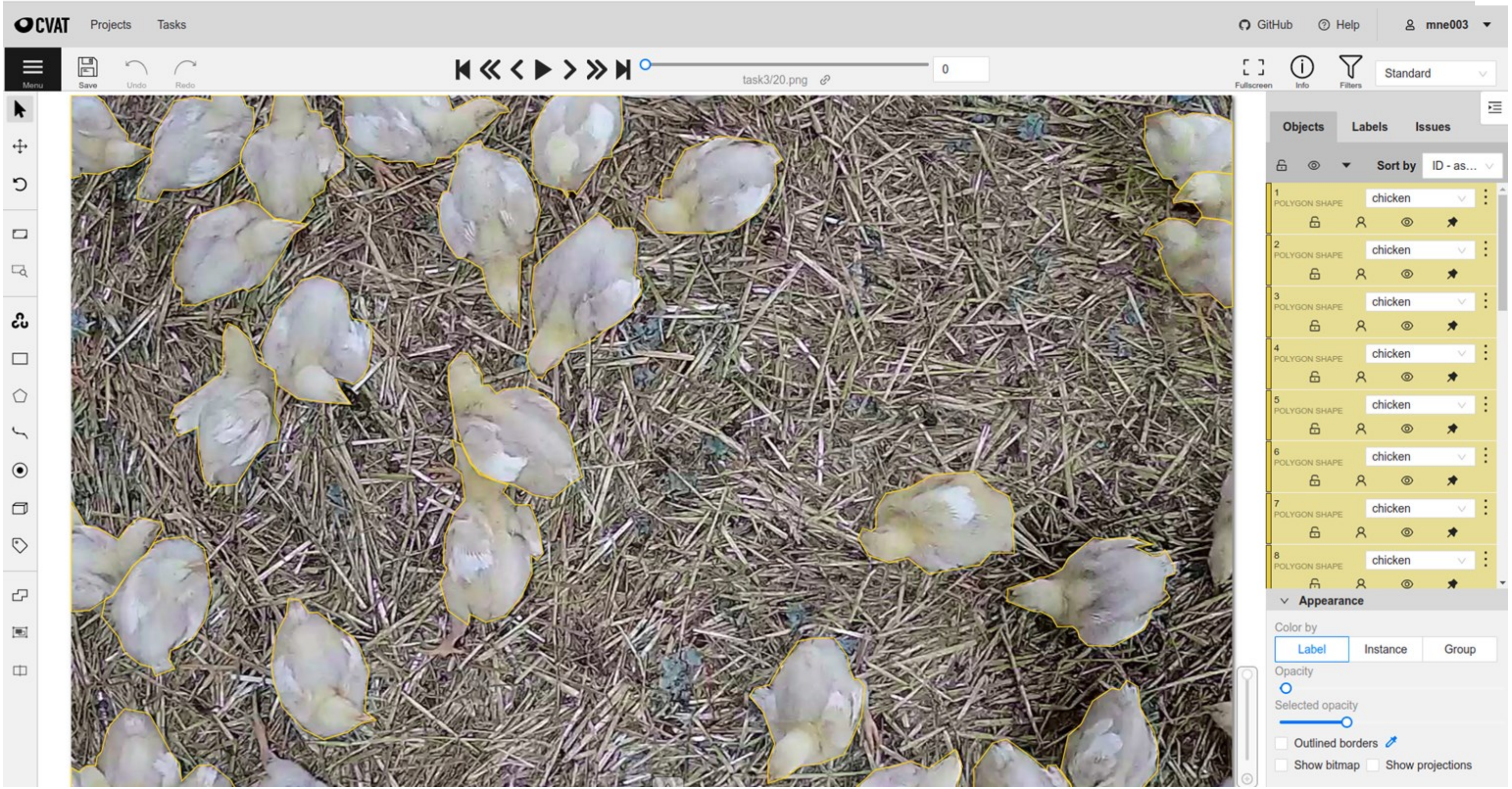
Data creation and preparation

- Dataset was extended from DunavNet dataset
- >4000 annotated images for **object detection**
  - after augmentation ~10000
- >1000 annotated images for **object segmentation**
  - after augmentation ~2500

Data labeling, object detection



Data labeling, object (instance) segmentation



Exploring different deep learning models

UDG
Powered by Arizona State University®

- Training on high-performance computing system.
- More than three thousand experiments on one or multiple GPUs.
- Fine tune hyperparameters (AutoML, HPO)

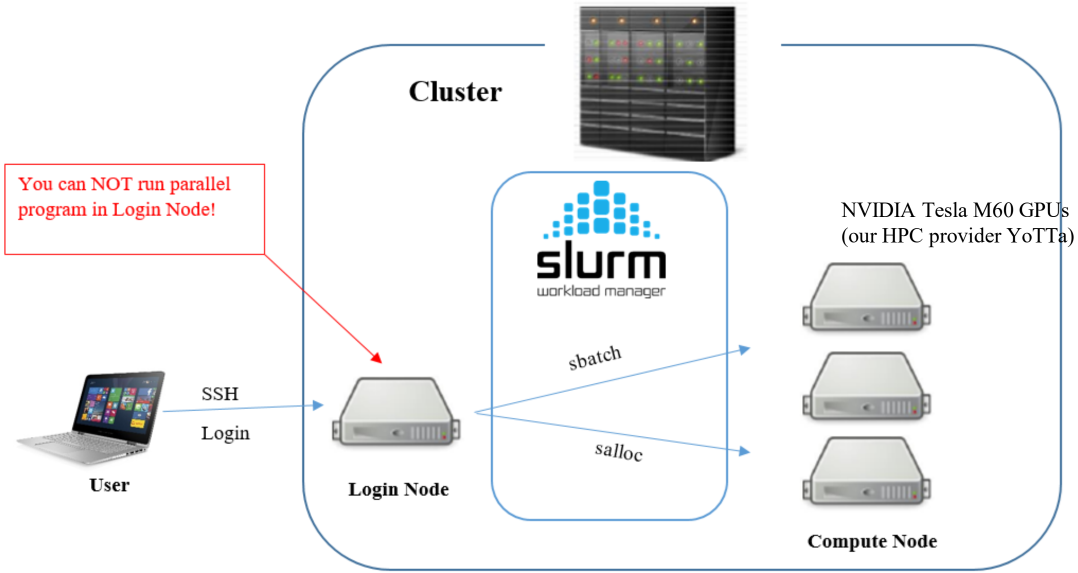
  and decrease training time.

DUNAV NET

Radinović

FRANCA

Slurm and high performance computing



Simple Linux Utility for Resource Management (SLURM) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system:

- **Allocates** access to **resources** for some duration of time.
- Provides a framework for **starting**, **executing**, and **monitoring** work on the set of allocated nodes.
- **Manage** resources by managing a queue.

Source: https://pdc-support.github.io/hpc-intro/09-scheduling/

High performance computing (slurm) job, example

```
#SBATCH --nodes=1
#SBATCH --account=DigitalSmart
#SBATCH --time=360:00:00
#SBATCH --ntasks=1
#SBATCH --partition=gpu
#SBATCH --output=pd_8.out
#SBATCH --job-name=pd_8
#SBATCH --gres=gpu:8

#učitavanje Python i CUDA softvera, pytorch, torchvision i detectron se
automatski učitaju zajedno s Pythonom
module load Python/3.9.5-GCCcore-10.3.0
module load CUDA/11.1.1-GCC-10.2.0

#promjena radnog foldera
cd /home/users/digitalsmart/aimhigh/detectron2/poultry_object_detection
srun --gres=gpu:8 python main.py --num-gpus 8
```

```
dataset ← load(object_detection_dataset)
optimal_batch ← 0
selected_nn_arch ←" "
ratio ← 0
gamma ← 0.5
steps ← 1000
base_lr ← 0.001
while num_gpus ∈ [1,2,4,8] do
    while curr_nn_arch ∈ faster_r_cnn[r_101_c4_3x, r_101_fpn_3x, r_50_c4_1x,
  r_cnn_r_50_c4_3x, r_50_FPN_1x, x_101_32x8d_fpn_3x] do
        while batch_size ∈ [2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8] do
            train(num_gpus, curr_nn_arch, gamma, base_lr, batch_size, data, steps)
            if ratio < average_precision/prediction_time then
                ratio ← average_precision/prediction_time
                optimal_batch_size ← batch_size
                selected_nn_arch ← curr_nn_arch
            end if
        end while
    end while
end while
```

```
detection_prediction_model ← none
best_average_precision ← 0
while num_gpus ∈ [1,2,4,8] do
    while batch_size ∈ [2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8] do
        while gamma ∈ [0.2, 0.5, 0.8, 1] do
            train(num_gpus, selected_nn_arch, gamma, base_lr, batch_size, data, steps)
            if best_average_precision < average_precision then
                best_average_precision ← average_precision
                update(detection_prediction_model)
            end if
        end while
    end while
end while
save(detection_prediction_model)
```

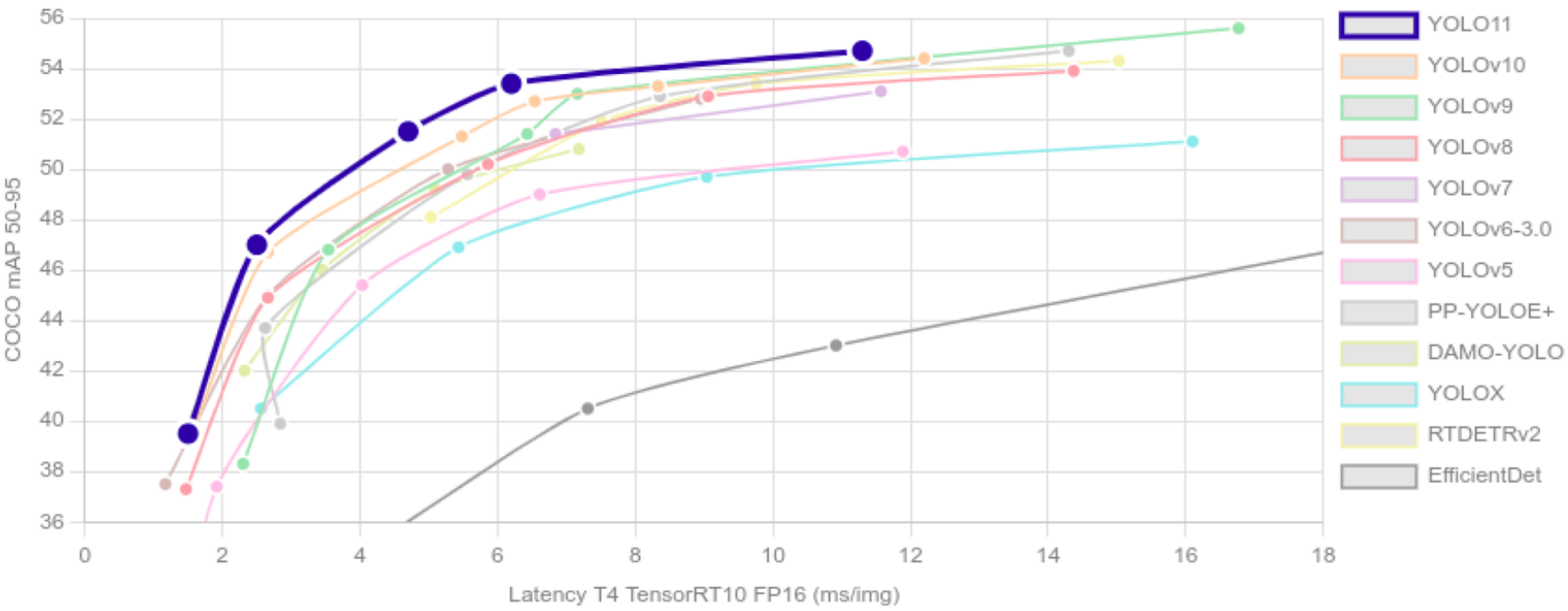
Note: for the instance segmentation we used the same methodology with different neural networks.

Number of GPUs	Neural Network Arhitecture	Training Time [min]	Batch Size	Validation Loss	AP	AP50	AP75	Prediction Time [s]
1	faster_r_cnn_R_101_C4_3x	50:50	128	0.27	83.92	97.79	94.31	0.6
	faster_r_cnn_R_101_FPN_3x	**22:44**	**256**	**0.31**	**82.96**	**97.8**	**94.3**	**0.2**
	faster_r_cnn_R_50_C4_1x	44:44	64	0.34	81.74	97.68	93.1	0.6
	faster_r_cnn_R_50_C4_3x	44:41	32	0.31	82.78	97.76	94.32	0.6
	faster_r_cnn_R_50_FPN_1x	16:43	8	0.36	80.75	97.7	93.2	0.2
	faster_r_cnn_X_101_32x8d_FPN_3x	42:05	8	0.31	83.39	97.81	94.42	0.4
2	faster_r_cnn_R_101_C4_3x	34:16	2	0.27	84.1	97.81	94.43	0.6
	faster_r_cnn_R_101_FPN_3x	15:58	4	0.3	82.9	97.82	94.4	0.2
	faster_r_cnn_R_50_C4_1x	30:03	4	0.34	82.22	97.73	93.9	0.6
	faster_r_cnn_R_50_C4_3x	30:0	256	0.31	82.73	97.78	94.2	0.6
	faster_r_cnn_R_50_FPN_1x	12:10	4	0.35	81.18	97.74	93.22	0.2
	faster_r_cnn_X_101_32x8d_FPN_3x	31:00	16	0.3	83.34	97.86	94.51	0.4
4	faster_r_cnn_R_101_C4_3x	32:19	256	0.26	84.1	97.8	94.3	0.6
	faster_r_cnn_R_101_FPN_3x	17:56	4	0.29	83.14	97.81	94.44	0.2
	faster_r_cnn_R_50_C4_1x	28:36	256	0.33	82.43	97.72	93.13	0.6
	faster_r_cnn_R_50_C4_3x	28:30	256	0.3	82.58	97.8	94.3	0.6
	faster_r_cnn_R_50_FPN_1x	14:25	128	0.33	81.34	97.74	93.22	0.2
	faster_r_cnn_X_101_32x8d_FPN_3x	31:17	64	0.31	83.47	97.82	94.56	0.4
8	faster_r_cnn_R_101_C4_3x	36:10	256	0.25	83.8	97.81	94.3	0.6
	faster_r_cnn_R_101_FPN_3x	13:57	16	0.28	83.24	97.81	94.43	0.2
	faster_r_cnn_R_50_C4_1x	32:28	256	0.32	82.43	97.72	93.21	0.6
	faster_r_cnn_R_50_C4_3x	32:31	256	0.29	82.83	97.8	94.3	0.6
	faster_r_cnn_R_50_FPN_1x	20:57	8	0.32	81.61	97.76	94.15	0.2
	faster_r_cnn_X_101_32x8d_FPN_3x	37:36	32	0.32	83.59	97.85	94.56	0.4

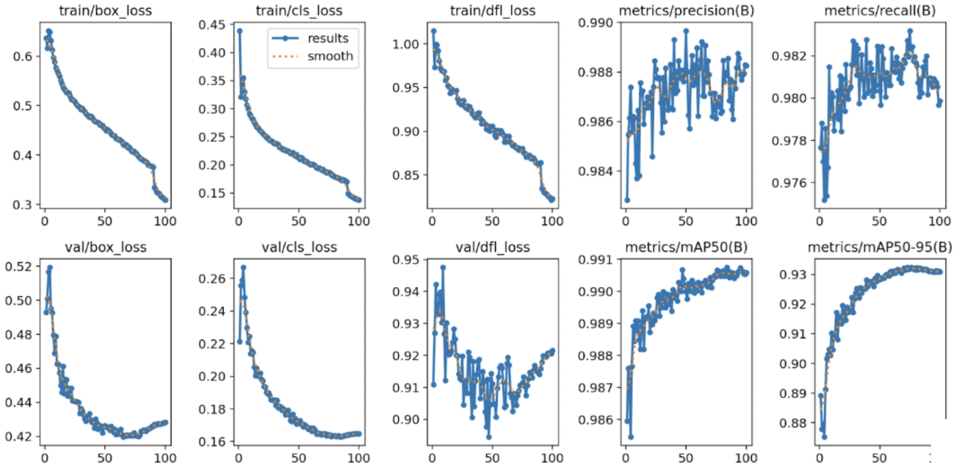
- **Object detection (Table 1)**: evaluation of different Faster R-CNN network architectures in Detectron2, gamma = 0.5, steps = 1000.
- **Table 2**: Hyperparameter optimization results for faster_rcnn_R_101_FPN_3x, 1000 epochs, for different values of gamma and batch sizes.

Number of GPUs	Training Time [min]	Gamma	Batch Size	Validation Loss	AP	AP50	AP75
1	22:20	0.8	1	0.31	83.7	97.86	94.62
2	15:45	0.8	128	0.3	84.11	97.85	95.43
4	17:53	1	256	0.27	84.56	97.88	95.59
8	24:25	1	8	0.26	84.62	97.88	95.68

Note: for the instance segmentation we used the same methodology with different neural networks.

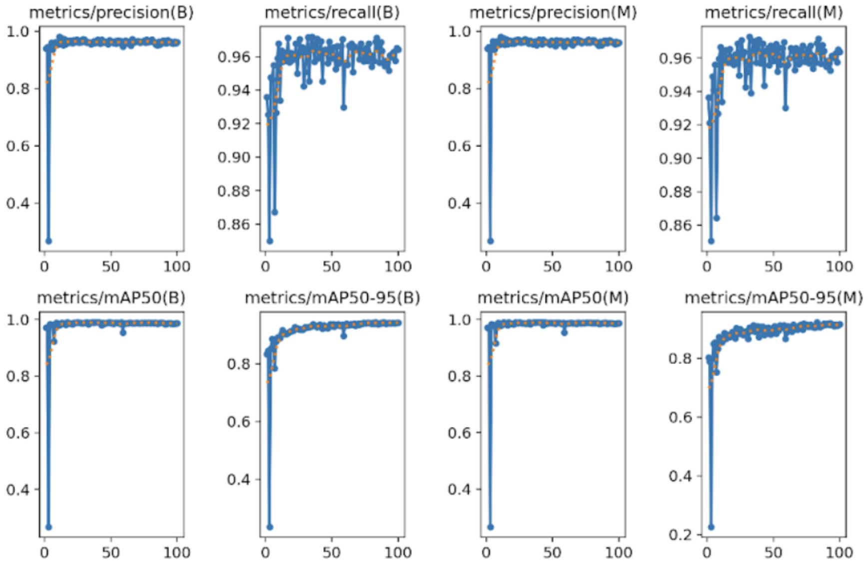


Reference: Ultralytics. YOLOv11 Overview. Available online: https://docs.ultralytics.com/models/yolo11/#overview

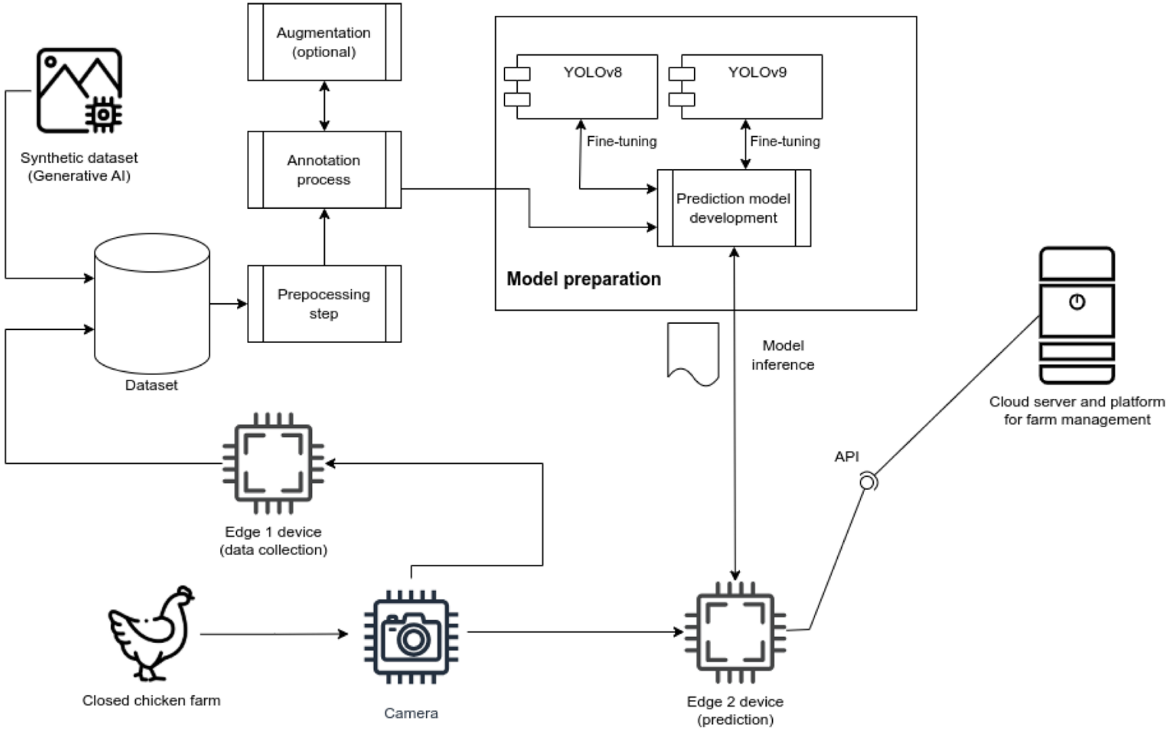


YOLOv9e object detection (chickens)
mAP more than 93%

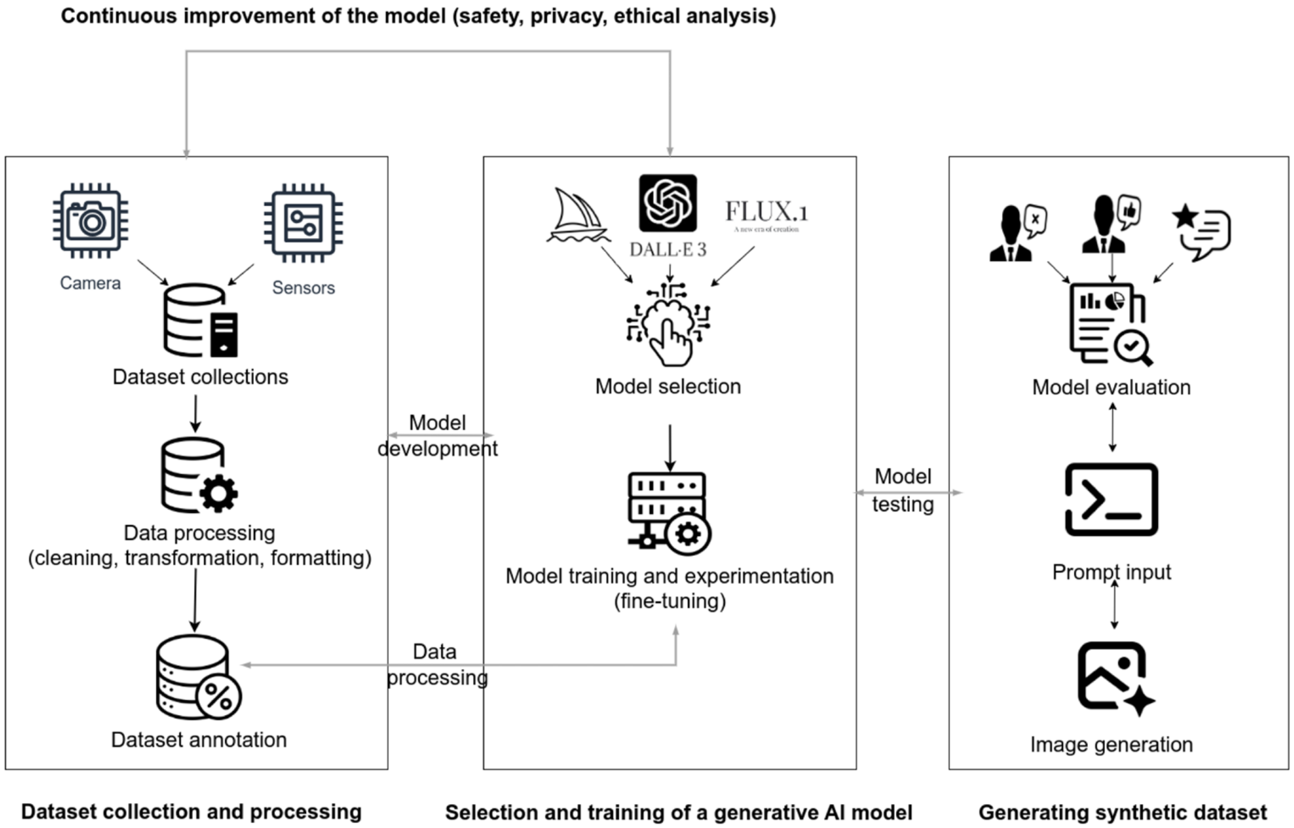
YOLOv9-seg segmentation (chickens)
mAP more than 90%



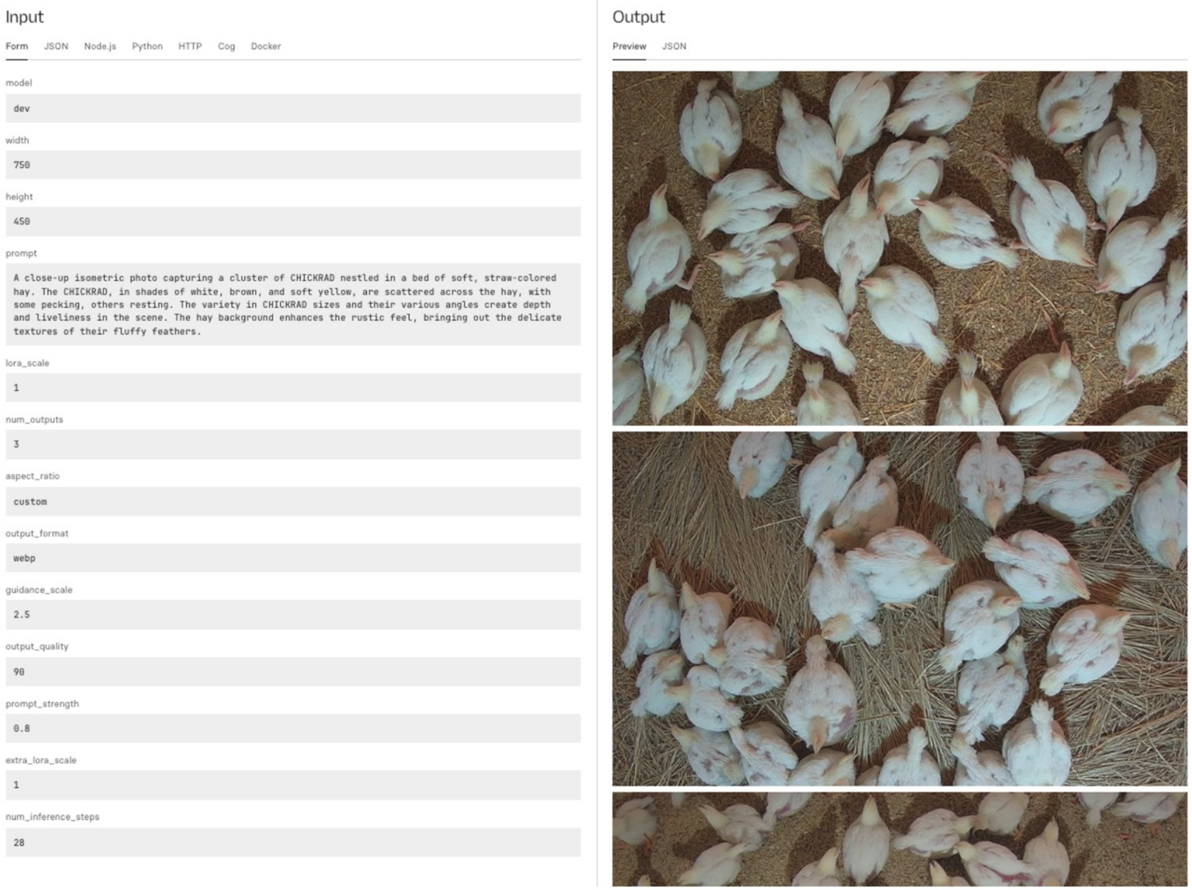
Architecture (with GenAI)



Reference: Stevan Cakic; Popovic, T.; Srdjan Krco; Jovovic, I.; Babic, D. Evaluating the FLUX.1 Synthetic Data on YOLOv9 for AI-Powered Poultry Farming. *Applied Sciences* 2025, *15*, 3663–3663, doi:https://doi.org/10.3390/app15073663



Reference: Stevan Cakic; Popovic, T.; Srdjan Krco; Jovovic, I.; Babic, D. Evaluating the FLUX.1 Synthetic Data on YOLOv9 for AI-Powered Poultry Farming. *Applied Sciences* 2025, *15*, 3663–3663, doi:https://doi.org/10.3390/app15073663



Reference: Stevan Cakic; Popovic, T.; Srdjan Krco; Jovovic, I.; Babic, D. Evaluating the FLUX.1 Synthetic Data on YOLOv9 for AI-Powered Poultry Farming. *Applied Sciences* 2025, *15*, 3663–3663, doi:https://doi.org/10.3390/app15073663

Algorithm 1 Implementation steps of generative AI for extending dataset with synthetic data.

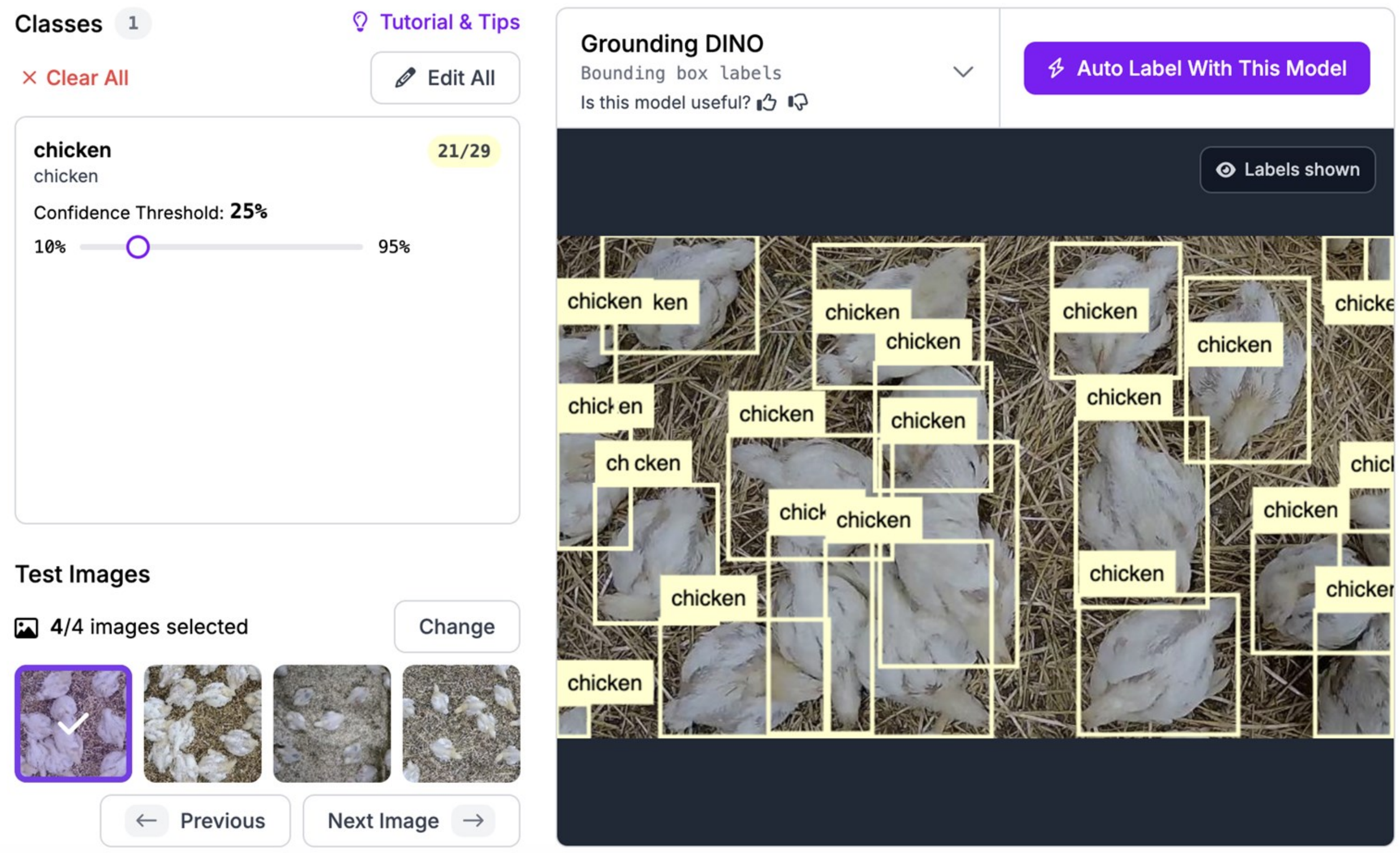
$real\_dataset \leftarrow collect\_real\_dataset$
$real\_dataset\_chunk \leftarrow select\_sample\_from\_real\_dataset$
$selected\_generative\_ai\_model \leftarrow Flux1.1[dev]$
Model training
$trigger\_word \leftarrow$ "CHICKRAD"
$lora\_rank \leftarrow 16$
$steps \leftarrow 1000$
$train(selected\_generative\_ai\_model)$
Generate synthetic data
$lora\_scale \leftarrow 1$
$guidance\_scale \leftarrow 2.5$
$prompt\_scale \leftarrow 0.8$
$prompt \leftarrow$ "A close – up isometric photo capturing a cluster of CHICKRAD nestled in a bed of soft, straw
$generate\_synhtetic\_data()$

Table 1. The impact of synthetic data on the accuracy of the YOLOv9-e model, tested on 50 real images, object detection. Number of training epochs: 100, batch size: 16.

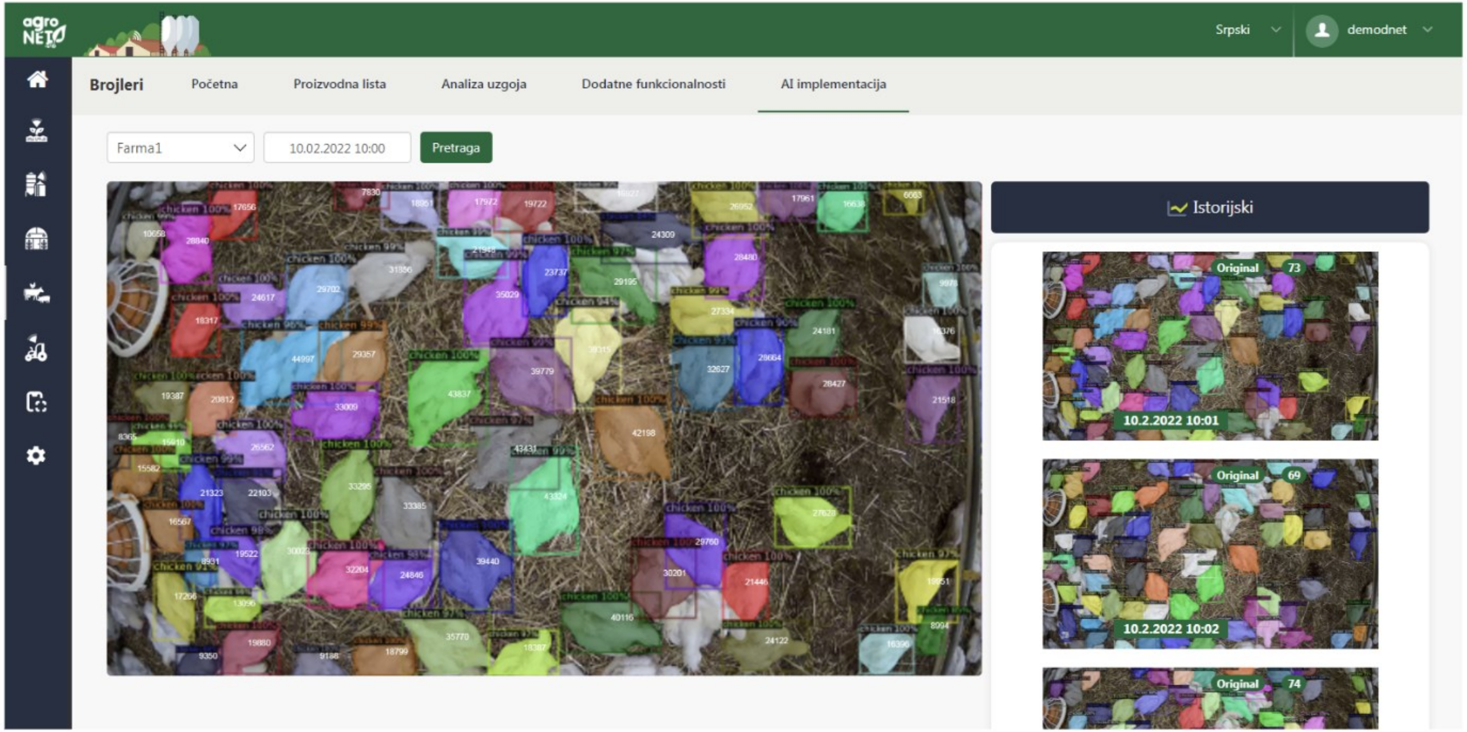
Experiment ID	Dataset Split				Metric
	Real	Synthetic	Augmented	Total	mAP
1	0	0	0	0	0.245
2	100	0	0	100	0.796
3	400	0	0	400	0.822
4	0	100	0	100	0.793
5	0	400	0	400	0.789
6	50	50	0	100	0.801
7	200	200	0	400	0.821
8	300	100	0	400	0.829
9	100	300	0	400	0.820
10	400	0	800	1200	0.820
11	0	400	800	1200	0.792
12	200	200	800	1200	0.814
13	300	100	800	1200	0.812
14	100	300	800	1200	0.808
15	400	0	800	1200	0.827

UDG
Powered by
Arizona State University®

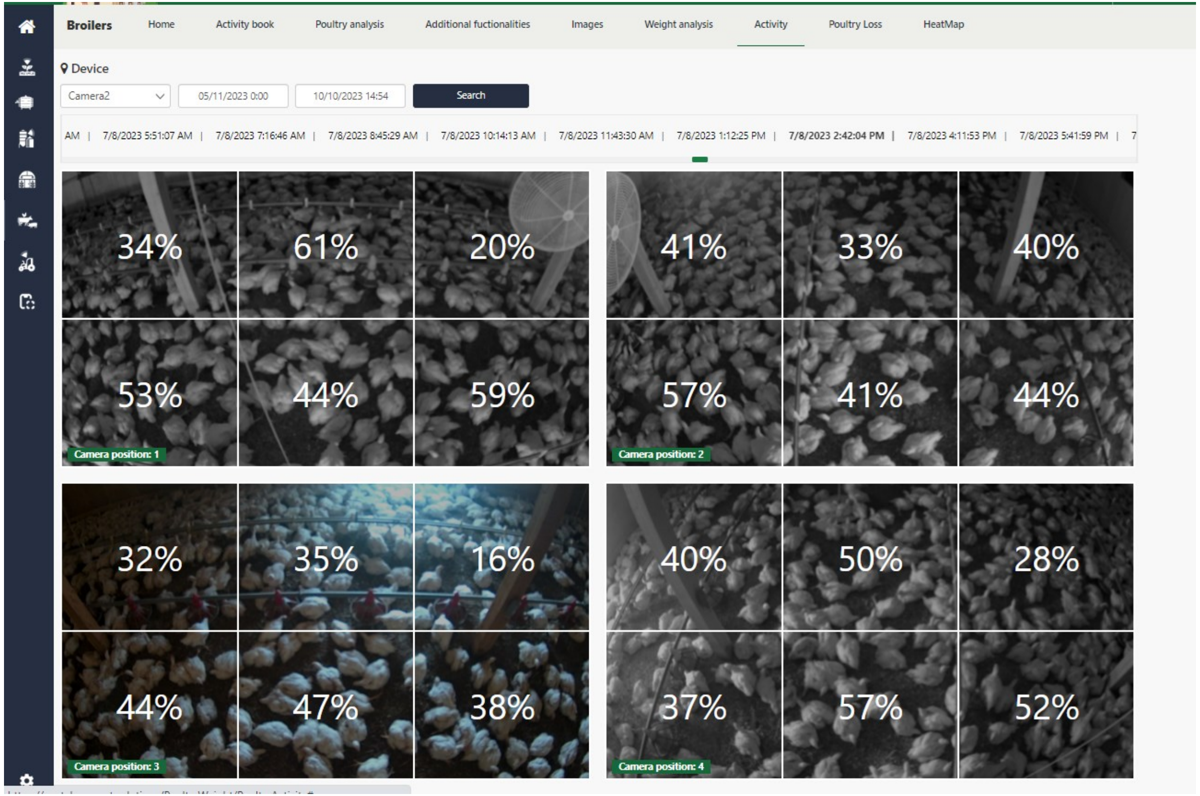
Reference: Stevan Cakic; Popovic, T.; Srdjan Krco; Jovovic, I.; Babic, D. Evaluating the FLUX.1 Synthetic Data on YOLOv9 for AI-Powered Poultry Farming. *Applied Sciences* 2025, *15*, 3663–3663, doi:https://doi.org/10.3390/app15073663


Classes 1
Tutorial & Tips
Clear All
Edit All
chicken
chicken
21/29
Confidence Threshold: 25%
10%
95%
Test Images
4/4 images selected
Change
Previous
Next Image
Grounding DINO
Bounding box labels
Is this model useful?
Auto Label With This Model
Labels shown
chicken

Platform integration



Degree of occupancy (density)



Key outcomes

Powered by Arizona State University®

- Annotated large amount of data (several thousands of images, extended DunavNet dataset);
- Choice of optimal model architecture and parameters for learning/training efficient AI model;
- Transferring the model to the IoT/AI edge platform;
- Integration with a commercial platform for agriculture agroNet (90% + accuracy);
- Validation with images, end users; Generative AI can reduce the need for real data;
- Ongoing research is related to weight estimation.