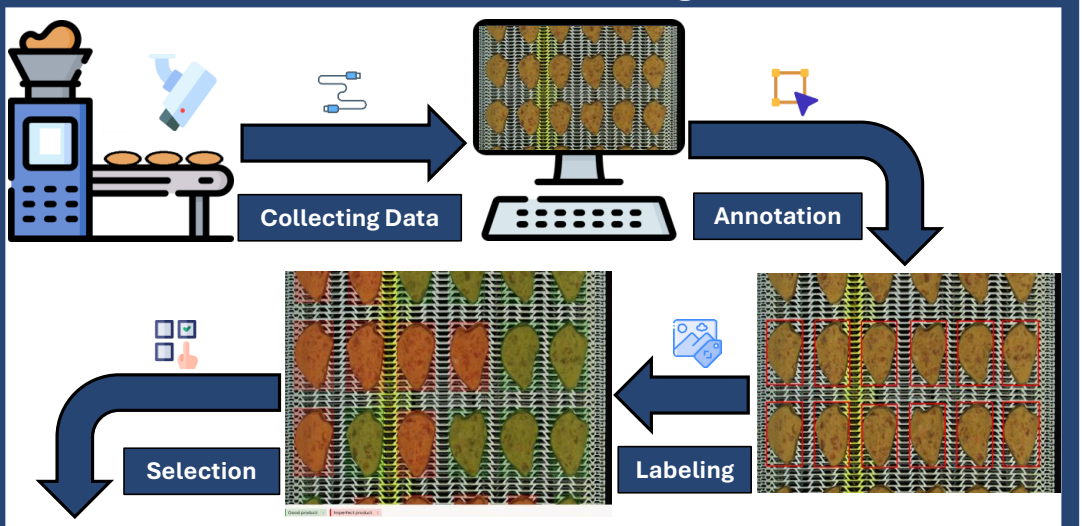
Graphical Abstract

Poultry Perfection - Comparison of Computer Vision Models to Detect and Classify Poultry Products in a Production Setting

Daniel Einsiedel, Marco Vita, Dana Jox, Bertus Dunnewind, Johan Meulendijks, Christian Krupitzer

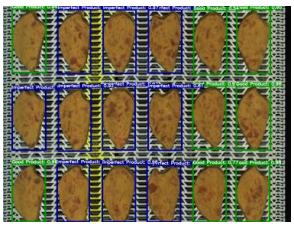
Poultry Perfection - Comparison of Computer Vision Models to Detect and Classify Poultry Products in a Production Setting



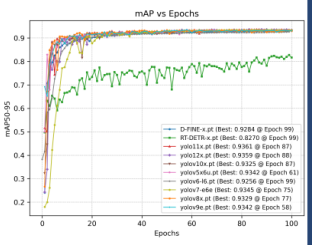
Models

- YOLO (v5, v6, v7, v8, v9, v10, 11, 12)
- RT-DETR
- D-Fine

Training



Evaluation



Conclusion:
The study shows that modern object detectors can successfully be used for food quality analysis without extensive modifications, with a diverse and well-labeled dataset being crucial.

UNIVERSITY OF HOHENHEIM

Graphical abstract icons: Icons made by Freepik, Smashicons, Eucalyp, Vitaly Gorbachev, Good Ware, surang, Icon Hubs, and Uniconlabs from www.flaticon.com

Highlights

Poultry Perfection - Comparison of Computer Vision Models to Detect and Classify Poultry Products in a Production Setting

Daniel Einsiedel, Marco Vita, Dana Jox, Bertus Dunnewind, Johan Meulendijks, Christian Krupitzer

- Computer vision was applied to assess quality of ready-to-eat meat products.
- Models reliably distinguished between "good" and "imperfect" products based on visual quality deviations.
- High detection accuracy (mAP50-95 > 0.9) demonstrates the feasibility of automated visual inspection in food processing.
- Balanced and consistently labeled datasets are critical for reliable food quality control in practice.

Poultry Perfection - Comparison of Computer Vision Models to Detect and Classify Poultry Products in a Production Setting

Daniel Einsiedel[a,b,c], Marco Vita[a], Dana Jox[a,b], Bertus Dunnewind[c], Johan Meulendijks[c], Christian Krupitzer[a,b,*]

[a] *Food Informatics, University of Hohenheim, Fruwirthstraße 21, Stuttgart, 70599, Germany*
[b] *Computational Science Hub, University of Hohenheim, Schloss Hohenheim 1, Stuttgart, 70593, Germany*
[c] *Marel Further Processing B.V., Handelstraat 3, Boxmeer, 5831, Netherlands*

Abstract

This study explores the use of computer vision, specifically object detection, for quality control in ready-to-eat meat products. We focused on a single process step, labeling products as "good" or "imperfect". An "imperfect product" constitutes a product that deviates from the norm regarding shape, size, or color (having a hole, missing edges, dark particles, etc.). Imperfect does not mean the product is inedible or a risk to food safety, but it affects the overall product quality. Various object detectors, such as YOLO, including YOLO12, were compared using the mAP50-95 metric. Most models achieved mAP scores over 0.9, with YOLO12 reaching a peak score of 0.9359. The precision and recall curves indicated that the model learned the "imperfect product" class better, most likely due to its higher representation. This underscores the importance of a balanced dataset, which is challenging to achieve in real-world settings. The confusion matrix revealed false positives, suggesting that

*Corresponding author
Email address: Christian.Krupitzer@uni-hohenheim.de (Christian Krupitzer)
URL: https://foodinformatics.uni-hohenheim.de/krupitzer (Christian Krupitzer)

increasing dataset volume or hyperparameter tuning could help. However, increasing the dataset volume is usually the more difficult path since data acquisition and especially labeling are by far the most time-consuming steps of the whole process.

Overall, current models can be applied to quality control tasks with some margin of error. Our experiments show that high-quality, consistently labeled datasets are potentially more important than the choice of the model for achieving good results. The applied hyperparameter tuning on the YOLO12 model did not outperform the default model in this case. Future work could involve training models on a multi-class dataset with hyperparameter optimization. A multi-class dataset could contain more specific classes than just "good" and "imperfect," making trained models capable of actually predicting specific quality deviations.

Keywords: Food quality, Computer vision, Object detection, Neural networks

PACS: 0000, 1111

2000 MSC: 0000, 1111

---

1. Introduction

With an ever-increasing world population and various crises affecting humanity, advancements across the entire food supply chain are necessary to secure food production and maintain food safety and quality in the future (Krupitzer and Stein, 2024). In recent years, the world has seen significant progress in the areas of artificial intelligence leveraged by machine learning methods (Kakani et al., 2020). This leads to the question of how these developments can be used to enhance food production.

One important aspect of food production is food quality control, which still often relies on humans doing the evaluation. This is costly on one side and also prone to errors due to the influence of physiological factors, especially when it comes to tasks such as visual inspection (Du and Sun, 2006). Computer vision in itself is not a

new concept and has been employed in various areas such as medicine (Esteva et al., 2021), autonomous driving (Dong and Cappuccio, 2024), security monitoring (Johnson, 2023) but also to some degree in agriculture and food production (Ma et al., 2024; Zhu et al., 2021). While earlier computer vision tasks were primarily focused on basic image processing, today's computer vision models are significantly more sophisticated (Laad et al., 2024). This advancement is largely due to the increase in computational power and the availability of large datasets, which enable the training of complex models capable of handling intricate visual data (Laad et al., 2024).

Most modern computer vision models are currently based on neural networks, more specifically, the established convolutional neural networks (CNNs) but also the more recent vision transformers (ViTs) (Li and Zhang, 2024). The main difference between these two architectures is how they treat the input images. CNNs use, as the name indicates, convolutional layers to process images in a spatially localized manner, while ViTs treat images as a sequence of tokens, also leveraging a self-attention mechanism to keep long-range dependencies (Haruna et al., 2025). Both ViTs and CNNs have already been used for various tasks related to food quality, for example for the detection of diseases in tomatoes (Simonyan and Zisserman, 2015), classification and identification of olives (Salvucci et al., 2022), classification of red meat based on its freshness (Bajpai et al., 2024), or identification of poultry diseases (Vrindavanam et al., 2024). However, most existing studies in this field are limited in two key aspects: they are typically conducted in controlled laboratory environments and often focus on evaluating a single model architecture. This limits their generalizability and practical applicability in real-world production settings. In contrast, our study addresses this gap by applying and comparing several cutting-edge object detection models, on data collected directly from a running poultry production process.

The goal of this work was to implement and compare different computer vision models to find out which ones perform best in the task of distinguishing "good" products from "imperfect" products. Another important aspect of this work was how simple the implementation of such models is for someone without specific domain knowledge in computer vision.

While this work does not introduce a novel model architecture or algorithmic advancement, its contribution lies in the application of state-of-the-art object detection models to a real-world industrial setting. Unlike many studies that rely on standardized or laboratory-collected datasets, our data was gathered during live production at a poultry processing facility, reflecting realistic visual variability and operational conditions. This practical grounding, combined with a comparative evaluation of several recent models, including YOLOv10 through YOLOv12 and DFine, addresses a gap in applied computer vision literature, particularly within the underexplored domain of food processing environments. To the best of our knowledge, this is the first study to apply some of these state-of-the-art models in a real-world food production environment.

First, the images were collected on-site using cameras mounted on a production belt. The images were then preprocessed, meaning bounding boxes and labels were added to the images, which is necessary to train a model. Different models were chosen, which was mainly based on how recent, popular, and easy to implement the models were. The models were trained and evaluated using our dataset and subsequently compared to each other using the mAP50-95 metric. Determined by this comparison, we examined the "best" model by using additional performance metrics for this purpose. The paper is structured as follows: In Section 2, our overall approach is explained; this includes an explanation of the production process where the images came from, as well as a more detailed explanation of how a machine

learning task like this is tackled and how the evaluation was done. Section 3 presents the results of the experiments and discusses them. Section 3.1 and 3.2 focus on some preliminary experiments that were conducted as a set-up for the main comparison. The comparison of multiple models is described in Section 3.3, these results directly lead to Section 3.4 where the most promising (or recent) model was examined more closely. In Section 4, potential threats to validity are brought forward. Finally, in Section 5, the ndings are summarized, drawing a conclusion and giving an outlook on potential future aspects of the work.

2. Materials and Methods

This section describes our approach and how we implemented the various steps. The owchart shown in Figure 1 provides an overview of how a machine learning task is usually tackled. We also followed this approach and will explain it in the following in detail.

[Figure 1 about here.]

The rst step, as simple as it may sound, is to understand the underlying problem to be solved. In this study, the problem can be described as an object detection task. Objects, namely chicken products on a belt, should be individually localized and then a label should be assigned, sorting them into the correct class. This already narrows down which models can be used and also means that the necessary data comes in the form of images. The next step is data collection and for machine learning it is essential to collect as much relevant data as somehow possible. After enough data is collected, the data needs to be cleaned. This step varies depending on what kind of data is used. In the case of image data, it means selecting relevant images and building a training dataset for the machine learning task. Afterwards, the images are

checked (le type, duplicates, etc.) and then brought into the format expected by the algorithm. This often happens with the next step already in mind, which is the model selection. Most models have certain parameters that can be adjusted, which might be necessary depending on the input. Afterwards, the model is trained and the results are evaluated using the various metrics mentioned in Section 2.1. Usually, multiple models are rst tested, and the best one is selected before switching from the O ine-phase to the Online-phase, in which a model is deployed and used. This work only includes the rst phase, which will be explained thoroughly on the presented use case, including the implementation.

Data collection: To collect enough data, camera systems were installed after various steps over the belt of a running process line as displayed in Figure 2. Each camera system featured two cameras. In total, there are four camera systems in the process, all equipped with two cameras and a powerful light source to ensure consistent lighting. Due to this setup, the camera's angle and distance to the belt were xed, so the cameras were used as intended for customer use. Additionally, since the data collection happened in an active process environment during regular production, no changes could be made regarding the environment or test setup. While this setup does not allow extensive experimentation, it does produce real-world data from an actual production process with no alteration and consistent data even over multiple production days. The cameras were run on three di erent production days for approximately. 30 minutes to 1 hour each, collecting between 2,000 and 10,000 images per production step. The camera symbols) (in Figure 2 show the places where images were continuously taken.

The analyzed process includes the following steps: First, the raw meat mass is lled into a pump, which applies it into forming moulds of the forming machine. Second, the formed product is ejected on the production belt, and liquid batter is

applied, which acts as an adhesive for the following coating. Then, the product gets par-fried and cooked in an oven, which nally results in a ready-to-eat product. For this work, we only focused on the forming production step.

[Figure 2 about here.]

Data Analysis and Preprocessing: First, we used GroundingDINO (Liu et al., 2025), which essentially generates bounding boxes automatically using a text-prompt as input. Bounding boxes are rectangular boxes that de ne the size and position of an object within a picture (see Figure 3, the red boxes). Bounding boxes are needed for the models to understand where the objects are located so that they know what the regions of interest are, and to learn the objects within them.

[Figure 3 about here.]

An example of an image processed with GroundingDINO is depicted in Figure 3. The image processing works quite well and reduces the work for manual annotation considerably. The gure also shows that this process is not always perfect, e.g., the top row of chicken products does not have bounding boxes, so manual intervention is still needed. The bounding box information generated by GroundingDINO is saved to JSONles, which can then be imported to a labeling software. We used the open source tool Label Studio as labeling software. Using Label Studio, the images were labeled manually and sorted into classes (see Figure A1 in Einsiedel et al. (2025)).

An example of what is considered a good product and what would be considered imperfect is shown in Figure 4. It is important to note that we chose the class name imperfect product since sorting out or throwing away those products is not necessarily required. They just deviate in some visible way from what might be expected of that product regarding shape or color, among other aspects.

[Figure 4 about here.]

After all images are labeled, they can be exported in various formats, such as the YOLO format, the COCO format, and Pascal VOC, to support different machine learning frameworks for further analysis. For our models, only the first two are relevant. The YOLO format has a corresponding .txt file for each image, meaning for a image1.png, a corresponding image1.txt file containing a class ID representing the class (e.g., 0 = good product, 1 = imperfect product). In comparison to that, the COCO format is usually one singular .JSON file containing the image names as well as the bounding box information for the entire dataset (see Listing 1 and Listing 2 in Einsiedel et al. (2025)). Using the label files, the succeeding steps of the machine learning analysis could be performed. However, it is recommended that the different classes be balanced in size, i.e., the number of pictures should be similar. In cases with an imbalanced size of items, there is the optional step of augmenting the images.

Data augmentation is a method used not only in computer vision but in machine learning analysis in general. It inates the data set by applying various transformations to the input data. In the case of images, it can be operations like ipping the image, changing its brightness, and shuing the color channels, among other operations. The method is useful to tackle common problems such as tting while also making the model more robust (Shorten and Khoshgoftaar, 2019). For image augmentation, there are libraries such as umentations , which provide various augmentation techniques for images (see Listing 3 in Einsiedel et al. (2025). In our study, we experimented with custom augmentation, which included horizontal ipping, random brightness and contrast adjustment, grayscale conversion, and channel shuing. These augmentations were selected to mimic plausible variations that

may occur in practice due to changes in product orientation, lighting fluctuations, or sensor-related artifacts. They also include some very extreme changes, such as channel shuffling, even though the latter is probably not something that realistically occurs. The results of the augmentations are depicted in Figure 5. It is essential to remember that corresponding label files need to be created for the augmented images as well.

[Figure 5 about here.]

Once the images and label files are in the correct format, the dataset is split into a training and validation subset. In this study, a 0.8 split was chosen, meaning 80 % of the images were used for training, while 20 % were used to validate the model. The 80:20 split has been widely used, and in our case (with 2040 images), it would mean the model gets trained on 1632 images, while 408 are reserved for validation. This split ratio is also recommended by TensorFlow in their tutorial for image classification (tensorflow, 2019). However, it depends on the size of the dataset. For larger datasets, for example, a 0.9 split might also be appropriate.

Model Selection: The model selection in this study was even done before the preprocessing, since it determined how to format the data. For this work, we chose different versions of YOLO, most of them by Ultralytics, due to their easy implementation and fine-tuning capabilities, from YOLOv5 (Jocher, 2020) up to the most recent YOLO12 (Jocher et al., 2023). YOLO has been used for applications in food quality control to some degree. For example, YOLOv3 has been implemented for a mobile food grading system to detect defects on the peel of bananas (Zhu and Spachos, 2021). YOLOv5 has been used to detect impurities among walnut kernels on a belt, and YOLOv8 has been proposed for the detection of spoilage with a focus on fruits (Dhelia et al., 2024; Yu et al., 2023). Most of the research using YOLO

in the context of food seems to be on agricultural and farm-level applications rather than processed foods. We did not include all YOLO models since there have been many new developments over the last couple of years, and we wanted to focus more on the state-of-the-art. Also, older repositories are often no longer maintained and sometimes create issues with, e.g., newer software and dependencies. Since ViTs are also becoming more popular, we also trained two of those architectures, namely RT-DETR (Zhao et al., 2024) and D-FINE (Peng et al., 2025). ViTs have been examined in the food context as well, for example, for classifying mushrooms (to prevent people from consuming poisonous mushrooms) or for disease detection in strawberries (Aghamohammadesmaeilketabforoosh et al., 2024; Wang, 2022). RT-DETR has been used for ripeness detection of tomatoes (Wang et al., 2024b). However, the research for ViTs in the food sector is currently rather limited. We chose RT-DETR from Ultralytics because we could train it with very few modi cations using our training set-up from the YOLO models. Furthermore, this makes it easy to compare it to the other models provided by Ultralytics. D-FINE was chosen because it is one of the most recent ViT detectors, having been released in October 2024.

Model Con guration : Most object detection models have a couple of con gurations that can be adjusted. So, in the following, the most common ones that are usually speci ed in the training command will be explained:

- Image size: The image size determines the dimensions of the input. All images will usually be resized to this size to get a uniform input. A higher resolution is often desirable, but training on higher resolutions becomes computationally intensive very fast.

- Epoch: An epoch refers to one pass of the entire dataset through the model. For example, 100 epochs mean that the model sees the entire dataset 100

times during training. This allows the model to re ne and adjust its weights incrementally. If the number of epochs is set too low, the model might not perform ideally, while too many epochs can lead to over-training, which impacts the model's ability to generalize.

- Batch size: The batch size determines the number of images that are processed together as a group during one forward and backward pass. This means if you have 1,000 images and a batch size of 8, then 8 images are always fed at once to the model, and after 125 batches, one epoch is completed. Larger batch sizes allow faster training. However, it is also computationally intensive, so batch sizes are usually in the range of 8-32.

Besides that, most models also have a hyperparameter le containing additional parameters (such as learning rate, weight-decay, built-in augmentation operations, etc.). They can be manually tweaked or adjusted via hyperparameter tuning to further improve performance. For our experiments, we mainly applied the suggested default parameters of each model and only adjusted epochs and batch sizes when necessary. However, this will be discussed in more detail in Section 2.2.

Model training : After the appropriate con gurations have been chosen, the model can be trained. Usually, the training script is called, and optional pretrained weights are passed, along with the desired con gurations and the path to the dataset (see Listing 4 in Einsiedel et al. (2025)).

Model evaluation: The model evaluation takes place after and sometimes even during training between epochs. For evaluation, the validation dataset is used, which consists of labeled images the model has not seen during training. The model tries to predict the bounding boxes and their corresponding labels and then compares them to the ground truth. Based on that, various performance metrics are calculated,

which are further discussed in Section 2.1. Everything after the evaluation is part of the Online Phase and will not be covered in this work.

2.1. Performance metrics

Different metrics exist to evaluate machine learning models and give insight into their performance. Often, it is best to consider multiple metrics since no metric covers every aspect. The following presents a couple of the most popular metrics in computer vision, some of which are also used in this work. Table 1 summarizes the definition of true positives (TP), false negatives (FN), true negatives (TN), and false positives (FP), which are used in various metrics.

[Table 1 about here.]

Precision: Precision measures the number of correctly retrieved instances divided by the total number of retrieved instances (Dalianis, 2018). So, as an example, how many imperfect products were correctly classified as such. The equation for precision is shown in Equation 1.

$$\text{Precision: } P = \frac{TP}{TP + FP} \tag{1}$$

Recall: Recall measures the number of correctly retrieved instances divided by the total number of correct instances (Dalianis, 2018). Equation 2 is used for the calculation of recall.

$$\text{Recall: } R = \frac{TP}{TP + FN} \tag{2}$$

F -score: The F -score calculates the harmonic mean of precision and recall (Dalianis, 2018). Equation 3 shows how the Fscore is defined. The parameter deter-

mines the importance of precision or recall (Jiang et al., 2022). In this work, is set to 1, indicating that precision and recall are considered equally important.

$$F = (1 + ^2) \frac{\text{Precision} \quad \text{Recall}}{(^2 \quad \text{Precision}) + \text{Recall}} \tag{3}$$

Intersection over union (IoU): Intersection over union is the ratio of the area of overlap to the area of union between the predicted bounding box and the ground truth bounding box (Terven et al., 2023). IoU is used as a metric for assessing the quality of predicted bounding boxes, particularly in the eld of object detection, which is the focus of this work, making it a crucial metric (Stodt et al., 2023). The IoU can be calculated using Equation 4 (Hua, 2018). IoU primarily evaluates the detection of predicted bounding boxes rather than the classi cation of products. Hence, it allows us to evaluate how accurately the model detects and localizes the chicken products within the image.

$$IoU = \frac{A \setminus B}{A [B} \tag{4}$$

Mean average precision (mAP): The average precision (AP) represents the area under the precision-recall curve, evaluated at a speci c IoU threshold (Li et al., 2023b) and is calculated for each class individually, whereas the mAP provides the average of the computed AP values across all classes (Li et al., 2023b). Often, the distinction between mAP and AP is no longer made, leading to the frequent use of the term mAP. Frequently used are mAP50, mAP75, and mAP50-95, the rst two are the mAP evaluated at an IoU threshold of 0.5 and 0.75 respectively, while the last one is evaluated over a range of IoUs starting from 0.5 to 0.95 in steps of 0.05 (Reis et al., 2024). The mAP is calculated according to Equation 5.

$$\text{mAP} = \frac{\sum_{i=1}^{k} \text{AP}_i}{k} \quad \text{for k classes} \tag{5}$$

Confusion matrix: The confusion matrix is used to obtain more detailed information about the performance of the model (Gajjar and Jethva, 2022). It allows us to infer how many instances of a speci c class were correctly classi ed or misclassi ed as other classes (Gajjar and Jethva, 2022).

2.2. Experimental set-up

In the following, the overall set-up for the experiment will be explained. This will include a short explanation of the dataset and list the various models used and how they were compared. The models were trained using an NVIDIA TeslaV100 graphics card with 32 GB of VRAM on a virtual machine with an AMD EPYC 7452 Dotriaconta-core (32 Core) CPU, 16 GB of RAM using Ubuntu 22.04.5 LTS as OS and Python 3.10.12.

Dataset: As already mentioned, the images used for training the models were captured after the forming processing step on a production line for ready-to-eat chicken products, as depicted in Figure 2. The used data is a combination of images recorded on the production days, August 2, 2023, November 8, 2023, and January 31, 2024. The images were collected during the live production of corn akes-coated chicken escalopes. From the captured images, 2,040 were labeled and used for the subsequent training of the models. Additionally, 40 images containing only background were added. We also applied augmentation based on probability (see Listing 3 in Einsiedel et al. (2025)) to the images to further increase the dataset size. This was done according to Listing 3 in Einsiedel et al. (2025), where augmentations are applied with a certain probability. We chose 50 % for horizontal ip since that doesn't change the images dramatically while still providing some variation. The

other probabilities for the other augmentations were set to 20 % to provide some edge case examples. In Figure 6, the structure of the dataset is shown.

[Figure 6 about here.]

Preliminary experiments: When this study was started, YOLOv8 was the most recent version. In this study, we used it as a baseline model to start the experiments and explore how di erent approaches in uence the training. In the rst experiment, we trained a YOLOv8 model from scratch and one with pretrained weights. We used YOLOv8x as the largest YOLOv8 model with the default settings recommended by Ultralytics, which are an image size of 640, a batch size of 16, and 300 epochs. Using the results from that experiment, we also compared how our own augmented images in uence the training process by training a YOLOv8x model with and without them. Be aware that most models do apply their own augmentations during training. We, however, wanted to see the e ect of what happens when we include our own augmented images. To compare the models, we used the mAP50-95, which is commonly used for that. Table 2 shows the setup. From the results of these experiments, we derive the settings for further analyses. Section 3.1 and Section 3.2 present the results of the preliminary experiments.

[Table 2 about here.]

Comparison of di erent models: We trained several models and compared their mAPs after training to examine di erences between them. Based on the results from the preliminary experiments, we chose to use pretrained models when available, and if possible, we used the largest model size. We also tried to use similar settings for all models while adhering to the repositories' recommendations. Some of the models were more di cult to implement and computationally more intensive, which

means that in those cases, we adjusted them so that we could still run them with the available resources. Table 3 lists the complete list of trained models, including their settings. Section 3.3 shows and discusses the results of the models' comparison.

[Table 3 about here.]

We also explored the best model(s) using additional performance metrics besides the mAP, such as precision, recall, the F_1-score, and the confusion matrix. The selection of the best model(s) was based on their achieved mAP50-95 as well as factors such as the model's recentness and user-friendliness.

Additionally, Tukey's HSD was performed to find out which of the models differ significantly from each other (Tukey, 1949). The level of significance was set to $= 0.05$. To calculate the pairwise differences, the `statsmodels` library in Python was used. Section 3.4 elaborates on the details of the best-performing model. To calculate the difference all mAPs over the training epoch were considered.

Hyperparameter tuning of the best model: We conducted hyperparameter tuning of our top model, which was done with Ultralytics' integrated `tuner` class. To see whether this further improves the model compared to what it achieves with the default setting. The aim is to see whether this further improves the model compared to the results achieved with the default setting. This automated process trains the model over multiple iterations, applying a mutation of the hyperparameters after each training. As stated on their website, a genetic algorithm is used for this purpose. [1]. We implemented it according to their documentation; the structure of our `tune` command can be found in Listing 5 in Einsiedel et al. (2025). The epochs were set

[1] (https://docs.ultralytics.com/de/guides/hyperparameter-tuning/#what-are-hyperparameters)

to 100. While Ultralytics used 300 iterations in their example, we reduced this to 100, as otherwise the runtime would increase signi cantly.

3. Results and Discussion

This section explains and discusses the results of the experiments. First, we describe the results of the experiments that were conducted at the beginning to understand the in uence of pretraining (see Section 3.1) and data augmentation (see Section 3.2) using the YOLOv8x models. Second, Section 3.3 compares di erent models using the mAP50-95 as key metric. Third, Section 3.4 evaluates the best model by including additional performance metrics.

3.1. Preliminary Experiments: Fine-tuned Model vs. Model Trained from Scratch

As shown in Table 2, we started by training YOLOv8x on our dataset with the default settings. One model was initialized with pretrained weights; the other was trained from scratch. Figure 7a shows that the pretrained model surpasses the one trained from scratch very fast, which should not be surprising since it has already been trained on a dataset with other objects. The curve also shows that it converges faster than the model trained from scratch; however, it also runs into over tting visibly earlier, ultimately falling beneath the curve of the model trained from scratch at later epochs.

To train an ML model, a function is needed that provides a measure of the algorithm's performance (Ciampiconi et al., 2024). Loss functions are functions in ML that we want to minimize in order to optimize the model (Goodfellow et al., 2016, p. 80). The validation loss is calculated on the validation dataset and helps to detect over tting. Di erent loss functions exist depending on the type of task or application (Wang et al., 2022). The over tting can also be observed in Figure 7b,

where the validation loss curve increases towards the end for both models. The increase happens around the same epoch as the mAP starts decreasing. Important to mention is that while the epochs for the training were set to 300, the training was interrupted early in both instances. This is due to the so-called *patience* parameter, which was set to 100, i.e., if the validation loss does not improve for 100 consecutive epochs, training is halted, serving as an early stopping mechanism.

[Figure 7 about here.]

As shown in Table 4, the differences between the two models are not statistically significant. We did limit the epochs to 100 to prevent a drop in the model's performance and to reduce overall training time. This approach could be compared to early-stopping, which is widely used in practice to mitigate overfitting in deep learning (Hussein and Shareef, 2024).

[Table 4 about here.]

These results are also in line with other findings, which show that pretrained models converge faster, but they also show that, while training from scratch might take a bit longer to converge, the overall results are often similar (He et al., 2019). Additionally, a study observed that while pretraining may not always improve the model on the classic performance metrics, it may still improve the robustness and uncertainty estimates (Hendrycks et al., 2019). Considering those observations and taking our results into account, we decided to use the pretrained models for model comparison.

3.2. Preliminary Experiments: Influence of Data Augmentation

The next experiment was conducted to determine the influence of data augmentations on the mAP. So, we used the yolov8x.pt model and fine-tuned it without any

augmentations and once with our own augmentations as specified in Listing 3 in Einsiedel et al. (2025), comparing it to the default model (yolov8x.pt).

[Figure 8 about here.]

As we see in Figure 8, the model trained with custom augmentations reached a peak mAP of 0.9243, and the one trained without any augmentations reached a peak mAP of 0.9262, while the default model reached a peak mAP of 0.9320. However, while the default model converges more slowly at the start, it also does seem to remain rather stable over more training epochs, whereas the other two models, especially the one without any augmentations, drop off after reaching their peak. This can, as previously determined, be a sign of overfitting. With the mosaic augmentation, the default model applies one of the more sophisticated augmentation techniques to the data, which might be a reason for overfitting to occur later. For the custom augmentations, it happens earlier, and the model without any augmentations overfits after around 35-40 epochs. Table 5 provides a summary of the achieved peak mAPs of each model and the respective epoch in which it was achieved. It also shows that there are significant differences between the default yolov8x.pt model and the one trained without augmentations.

This experiment indicates that augmentations indeed influence the training process, and the type of augmentation seems to play a role. This is also confirmed by (Kumar and Muhammad, 2023) and (Modak and Stein, 2025), where different augmentation combinations were used, resulting in different mAP scores. Additionally, it also shows that augmentation is an effective and rather easy-to-implement regularization technique to prevent overfitting (Santos and Papa, 2022). Using own augmentations (or changing the augmentation settings in the model) might potentially yield better results than using the default settings predefined by the model.

Even though the model without augmentations also reached very good mAP values, it might still be a good choice to include some augmentations to increase the model's generalization abilities (Shorten and Khoshgoftaar, 2019). A model that has been trained on augmented images might be superior if the environment changes even slightly (e.g., lighting).

These results show that object detectors such as YOLO can work quite well straight out of the box. So, tweaking the model or doing hyperparameter tuning might not always be necessary unless it is absolutely critical to reach the highest possible mAP.

[Table 5 about here.]

3.3. Comparison of Different Models

The following section shows the results of comparing various iterations of YOLO among two other detectors (D-FINE and RT-DETR). In order to compare them, the biggest model size was used for all of them. We used pretrained weights and, to allow a baseline comparison, we also tried to train them (as far as possible) on their default settings, as shown in Table 3. The results are shown in Figure 9. The curve for the RT-DETR is visible below the other curves over the proposed training epochs and it also reached the lowest peak mAP with 0.8270. Compared to D-FINE, which is based on RT-DETR, D-FINE performs significantly better, reaching a peak mAP of 0.9284. The highest mAP overall was achieved by YOLO11x, closely followed by YOLO12x, with 0.9361 and 0.9359, respectively. However, what becomes quite evident when looking at these curves is that all of these models reach very similar mAPs for the given task.

[Figure 9 about here.]

Table 6 shows all the peak mAP50-95, and Tukey's HSD test reveals that, for the most part, there are no significant differences between the models, except RT-DETR, which was significantly different from all other models. These results reveal that with respect to pure accuracy, no model surpasses all the others by some visible margin.

[Table 6 about here.]

In recent years, the field of deep learning has matured substantially due to significant developments; however, within computer vision, the task of object detection remains particularly challenging (Dogra et al., 2024). While modern approaches perform well on generic scenarios, more specialized and niche applications still face numerous challenges, including background interference, occlusion, limited generalization, dataset bias, computational efficiency, and real-time performance constraints (Li et al., 2024). In recent years, there has been an increase in the AP from 30 % to 59.5 % on the COCO val dataset, but this remains largely unchanged despite quite a lot of new detectors and new iterations of YOLO being released (Du et al., 2021; Zong et al., 2023). Recent analyses suggest that this stagnation may stem from inherent dataset limitations and annotation inconsistencies (Tschirschwitz and Rodehorst, 2025). YOLOv8x, which we used as the baseline model, performs just as well as the newer iterations, and similarly, older iterations also still hold up. It seems like a lot of current advancement focuses mainly on the deployment of such models since the key metrics that are often compared when a new model is released are the latency versus the mAP. That means, how fast is the prediction while still keeping a certain level of accuracy. Also, the FLOPs versus the mAP is compared, meaning the trade-off between accuracy and computational cost. This implies that newer models often use fewer parameters, hence, are more lightweight and, thus,

also have less computational costs. But in the end, overall raw accuracy does not improve by a large margin (Tian et al., 2025a). For someone interested mainly in the application of such models and their overall accuracy of predictions, this also indicates that the selection of a model might be less important and should perhaps be decided based on how easy it is to implement. Therefore, it is more important to focus on having a large representative dataset and optimizing the model with regard to hyperparameters since this will have the most signi cant impact (Deepak and Bhat, 2025; Sun et al., 2017).

3.4. Detailed Analysis of one Model

[Figure 10 about here.]

As mentioned, the used models showed comparable accuracy on the use case; hence, we focused on the most recent iteration in the YOLO family since that could be considered state-of-the-art. Figure 10 displays, in addition to the mAP, the metrics recall, precision, and F_1-score, plotted over their con dence threshold. The con dence threshold represents how sure the model is about the presence of an object.

For the recall curve in Figure 10a, at low con dence levels, most predictions are accepted by the mode. At a threshold of 0 %, for example, all predictions are accepted, meaning the recall is at 100 %. However, this does not mean that the predictions are correct. As con dence increases, the recall slowly drops, reaching 0 % at 100 % con dence, which is logical since that would mean the model needs to be 100 % sure about this prediction. The curve for all classes remains over 80 % recall up to a con dence of around 70 %. But then dropping sharply after the 80 % con dence threshold is crossed. What also becomes evident is that the recall for the imperfect product class remains pretty high even at con dence thresholds beyond

80 %. On the other hand, the good product class lies below that. This could be explained simply by our dataset (see Figure 6), where we have less instances of the good product category.

The precision curve, which can be seen in Figure 10b, shows the precision of the model at various con dence thresholds. So, for example, if con dence is at 0 %, the model will output every possible detection, even the ones with low certainty. Hence, as the con dence increases, the precision is bound to increase as well. At 100 % con dence, predictions are only made when the model is absolutely certain. However, it is important to note here that at that con dence level, the model will probably just not return any predictions at all because it will never reach 100 % certainty for any of them. We can see in the graph that even for low con dence scores close to 0, the model reaches precision scores of around 70 % for the imperfect product category and around 50 % for the good product category. Since the good product category is less represented in our dataset, it makes sense that we reach lower precision at a low con dence threshold when compared to the imperfect product category. Hence, when running inference on images, it would be sensible to increase the con dence threshold to lter out potentially wrong predictions.

To select an appropriate threshold, both the recall and the precision should be considered, and one option is the F_1-score depicted in Figure 10c, which forms the harmonic mean between precision and recall and considers them equally. So, it balances the two main objectives of making correct detections while also nding the most objects. The highest F_1-score of 0.88 for both classes is achieved at a con dence level of 0.356. The imperfect products category achieved higher F_1-scores which remained rather constant over all con dence levels, only dropping after crossing the 80 % con dence threshold. Compared to that, the curve for the good product category starts slowly decreasing after the 40 % con dence threshold. Depending on

the application of a model, recall or precision might be more important. In those cases, the F_1-score can be replaced by calculating an F_β-score (see Equation 3) that ts the desired balance.

Lastly, the mAP50-95 is shown in Figure 10d over the trained epochs. Since the mAP is a single metric for all classes, only one curve is plotted. The curve increases sharply at the beginning and converges after approximately 10 epochs, reaching its peak mAP of 0.9359 at epoch 88, as listed in Table 6. This re ects the model's overall ability to detect objects across a range of IoU thresholds.

Figure 11 shows the confusion matrices obtained after validation, one depicting the absolute values and the other the normalized values. The absolute confusion matrix shows the raw counts of predictions for each class, while the normalized confusion matrix shows a percentage relative to the total number of samples in each actual class. The Background class is not a separate class but gets added automatically if, during the validation process, the model detects an object for which there is no bounding box or misses an object, essentially representing false positives or false negatives". For the good product category, we have a total of 2,768 instances, 2,092 were correctly predicted as such, which corresponds to 71 %. Then 667 of the same class were wrongly placed into the imperfect product class. Regarding the imperfect product class, the total amount of instances was 5,726, of which 5,425 were correctly predicted, which corresponds to 95 %. As it was already visible from the other metrics, this class was the one most reliably predicted. Some misclassi cations occurred, namely, 547 objects were detected and placed in the good product class, and 531 were predicted as imperfect product even though these belonged to no category at all. For our use case, this error is not that detrimental since it accounts for 11.2 % of all predictions. Still, it would be desirable to reduce this issue. We already included 1 % background images containing no objects at all, as it is

recommended by Ultralytics (Ultralytics, 2023). However, this did not improve the issue. Another potential solution to reduce this problem could be the addition of more images. However, this is a very time-intensive endeavor. Another possibility is hyperparameter tuning, which has been shown to improve models without needing more data and labeling (Deepak and Bhat, 2025).

[Figure 11 about here.]

[Figure 12 about here.]

We also ran inference (conf. threshold = 0.5) on a couple of images from the validation dataset to visualize detections, including some of our own augmented images (which were not used for the training of this model). This is depicted in Figure 12. The model works quite well on the rst three images 12a - 12c. Here, the model detected every object, and the predictions also had high con dence scores of approximately. 0.7-0.9. But on a gray image, it began to struggle, not detecting all instances and it completely failed on images where channel shu e was applied. Those results show how important the training data is, and that if a model is not trained on a certain edge case, it can fail even if that edge case isn't that much di erent than the normal images by human standards.

3.5. Hyperparameter tuning with YOLO12x

In this subsection, we present the results from conducting hyperparameter tuning using YOLO12x. The total runtime of the tuning was7 days, 15 hours, 31 minutes, and 49 seconds. After completion of all tuning iterations a yaml le with the best parameters is generated which was used to train the tuned model. In

Table 7, the hyperparameters that are used in the default model are compared to the ones used for the tuned model.

[Table 7 about here.]

In the following, we present the performance metrics of the tuned model, following a similar format to Section 3.4. The results are shown in Figure 13 and Figure 14. When compared to the default models, it is evident that tuning did not lead to any signi cant improvement in performance. We also evaluated the tness score, which in the Ultralytics implementation is calculated as a weighted sum of the mAP50 and the mAP50-95 as follows:

$$\text{tness score} = 0{:}1 \quad \text{mAP } 50 + 0{:}9 \quad \text{mAP } 50 \quad 95 \tag{6}$$

This analysis aimed to determine whether there was any gradual improvement over the course of hyperparameter tuning, or if the model performance had plateaued. As shown in Figure 15, aside from three outliers, the model consistently achieved very similar tness scores across all hyperparameter combinations. From this, we conclude that additional tuning iterations are unlikely to yield improvements beyond the current model's capabilities. We hypothesize that further performance gains may only be achievable with a larger or more diverse dataset.

[Figure 13 about here.]

[Figure 14 about here.]

[Figure 15 about here.]

4. Threats to Validity

In this section, the limitations of this work will be explained. For most machine learning tasks, the primary issues typically arise from the data used for training. In

this work, 2,040 images with multiple products were labeled and used for training and validation. This is not a huge amount of images, but since one person did it, labeling even more images would have been too time-consuming (especially since images for the other processing steps that were not part of this work had to be labeled as well). However, more images might have already solved the misclassification issue that occurred in this work. The relatively small dataset size limits the model's ability to generalize to unseen cases, especially rare edge cases or shifts in the production environment. Furthermore, the dataset may lack sufficient diversity to reflect all relevant real-world variations. Also, with 2,040 images, it is not unreasonable to assume that the model might not have seen all possible variations and edge cases that could occur in a longer running production. We did try to counteract this by collecting and using data from different production days. Thus, we accounted for changes in the overall production setup, different operators, and slight changes in the raw material, among other possible variations. Additionally, for one of the experiments, we applied data augmentation techniques and were able to increase the dataset to 3,714 pictures.

Another issue is the consistency in labeling. Since one person did it, there might be a bias toward what is considered a "good product" and what is considered an "imperfect product". Although two other people reviewed a sample of the labels and confirmed their correctness, no quantitative assessment of inter-rater reliability was conducted. In future work, including formal inter-rater reliability metrics would enhance the credibility of the dataset. Clear labeling guidelines and multiple annotators can reduce bias and enhance label quality, while semi-automated or active learning methods may lower manual effort and improve consistency. However, as we focused on binary classification, potential bias is less critical. Another issue is that there might be slight environmental changes during production, which can influence

a model's final performance. The dataset contained images from different production days, which might account for some variation. However, if, for example, the lighting changes significantly or even the product itself changes, the models would probably reach their limits. The same is true if, for example, the camera position changes or even the cameras themselves are replaced by another type of camera. Despite efforts to vary image conditions, this still poses a risk to result validity. A significant constraint is that we did not conduct systematic hyperparameter tuning on all models. We fine-tuned YOLOv12x (the model that performed best overall), but the other models were assessed using their default settings. This has a natural effect on how fair the comparison is. The main reason for this decision was the amount of time and computational resources required for thorough tuning, which exceeded the scope of this work.

5. Conclusions

In this work, multiple object detectors were trained for the task of food quality analysis and distinguishing good products from imperfect products. First, the results show that including augmentation in some way can be considered good practice since it reduces overfitting and increases the model's ability to generalize. For the binary detection task presented in this work, most of the recent models that can be considered state-of-the-art yield comparable results. The work also shows that out of the box detectors can achieve very good results without the need for any modifications to the models' architecture. So training and deployment of such models do not necessarily require years of expertise in the field of Computer Science anymore, unless everything should be built from scratch and tailored to the use case. So this work demonstrates that object detection tasks can be solved with relatively low

technical overhead, using open-source models and tools. This highlights a realistic path for industrial adoption without the need for deep model customization.

Our hyperparameter tuning with YOLOv12x did not yield better results than the default model. So the most essential part is acquiring a diverse dataset with consistent labeling, the latter being the most time-consuming aspect of it all. In general, it can be said that the larger and more diverse the dataset is, the better the model will perform. The work also shows why it is necessary to look at other metrics as well, since the misclassi cations issue that becomes evident in Figure 11 would not be visible if we only looked at metrics like mAP, precision, recall, or F_1-score. A potential solution for the issue could be an increased dataset size, which would be very time-intensive.

A key contribution of this study is the use of real-world data collected during live poultry production, o ering a rare look at model performance under operational conditions rather than in a controlled lab environment. This adds practical relevance and re ects challenges such as noise, motion blur, and product variation. To the best of our knowledge, this study presents the rst applied comparison of YOLOv11, YOLOv12, and DFine models within the food quality inspection domain, providing a valuable foundation for future research in real-world food processing environments

Furthermore, future work could include the training of models to not only discern between good and imperfect products , but also classify the imperfect product into more speci c categories as it was also shown in Figure 4 (e.g., this product has a dark particle, this product has a hole, etc.). It would also be interesting to see how easily the results of this work can be transferred to the other processing steps, since, as shown in Figure 2 for this work, only the images of the forming step were used. Another promising direction for future work lies in leveraging the temporal nature of the image data. Since the camera is mounted on a moving production

belt, each product is typically captured multiple times from slightly different perspectives. This temporal redundancy could be exploited by aggregating predictions across consecutive frames, potentially increasing the robustness and accuracy of the quality assessment. Integrating the results into digital food twins might be beneficial for tracing the status of the products throughout the production (Henrichs et al., 2022; Krupitzer et al., 2022). Further, we plan to correlate the state of the products, analyzed through computer vision, with state-of-the-art machine learning techniques (Jox et al., 2025) for root cause analysis.

CRediT authorship contribution statement

- Daniel Einsiedel: Writing – Original Draft, Visualization, Conceptualization, Formal Analysis, Methodology, Software, Data Curation, Investigation
- Marco Vita: Writing – Original Draft, Formal Analysis, Methodology, Review & Editing, Software, Data Curation, Investigation
- Dana Jox: Review & Editing
- Bertus Dunnewind: Conceptualization, Supervision, Resources, Review & Editing
- Johan Meulendijks: Conceptualization, Supervision, Resources, Project administration
- Christian Krupitzer: Conceptualization, Supervision, Review & Editing, Methodology, Funding acquisition, Project administration

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

- Daniel Einsiedel reports financial support and equipment, drugs, or supplies were provided by Marel Stork Poultry Processing BV. Daniel Einsiedel reports a relationship with Marel Stork Poultry Processing BV that includes: travel reimbursement.
- Bertus Dunnewind reports a relationship with Marel Stork Poultry Processing BV that includes: employment.
- Johan Meulendijks reports a relationship with Marel Stork Poultry Processing BV that includes: employment.

If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Code availability

The used code can be shared upon reasonable request

Declaration of generative AI and AI-assisted technologies in the writing process.

Statement: During the preparation of this work the author(s) used ChatGPT4o, Microsoft Copilot, and Grammarly in order to enhance the language and improve readability. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

References

Kimia Aghamohammadesmaeilketabforoosh, Soodeh Nikan, Giorgio Antonini, and Joshua M. Pearce. Optimizing strawberry disease and quality detection with vision transformers and attention-based convolutional neural networks. *Foods*, 13(12), 2024. ISSN 2304-8158. doi: https://doi.org/10.3390/foods13121869.

Abhishek Bajpai, Harshvardhan Rai, and Naveen Tiwari. An efficient cnn-based method for classification of red meat based on its freshness. In Siba K. Udgata, Srinivas Sethi, and Xiao-Zhi Gao, editors, *Intelligent Systems*, pages 393–405, Singapore, 2024. Springer Nature Singapore. ISBN 978-981-99-3932-9. doi: https://doi.org/10.1007/978-981-99-3932-9_34.

Lorenzo Ciampiconi, Adam Elwood, Marco Leonardi, Ashraf Mohamed, and Alessandro Rozza. A survey and taxonomy of loss functions in machine learning, 2024. Preprint, doi: https://doi.org/10.48550/arXiv.2301.05579.

Hercules Dalianis. Evaluation Metrics and Evaluation, chapter Evaluation Metrics and Evaluation, pages 45–53. Springer International Publishing, Cham, 2018. ISBN 978-3-319-78503-5. doi: https://doi.org/10.1007/978-3-319-78503-5.

G. Divya Deepak and Subraya Krishna Bhat. Optimizing yolov4 hyperparameters for enhanced vehicle detection in intelligent transportation systems. *International Journal of Intelligent Transportation Systems Research*, July 2025. ISSN 1868-8659. doi: https://doi.org/10.1007/s13177-025-00519-3.

Ayushi Dhelia, Srishti Chordia, and Kanisha B. Yolo-based food damage detection: An automated approach for quality control in food industry. In 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)

(I-SMAC) , pages 1444 1449, 2024. doi: https://doi.org/10.1109/I-SMAC61858.2024.10714664.

Ashish Kumar Dogra, Vipal Sharma, and Harsh Sohal. A survey of deep learning techniques for detecting and recognizing objects in complex environments. Computer Science Review, 54:100686, 2024. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2024.100686.

Xingshuai Dong and Massimiliano L. Cappuccio. Applications of computer vision in autonomous vehicles: Methods, challenges and future directions, 2024. Preprint, doi: https://doi.org/10.48550/arXiv.2311.09093.

Cheng-Jin Du and Da-Wen Sun. Learning techniques used in computer vision for food quality evaluation: a review. Journal of Food Engineering, 72(1):39 55, 2006. ISSN 0260-8774. doi: https://doi.org/10.1016/j.jfoodeng.2004.11.017.

Xianzhi Du, Barret Zoph, Wei-Chih Hung, and Tsung-Yi Lin. Simple training strategies and model scaling for object detection, 2021. Preprint, doi: https://doi.org/10.48550/arXiv.2107.00057.

Daniel Einsiedel, Marco Vita, Dana Jox, Bertus Dunnewind, Johan Meulendijks, and Christian Krupitzer. Supplementary material for: Poultry perfection comparison of computer vision models to detect and classify poultry products in a production setting. Supplementary Appendix, 2025. Associated with the article published in Applied Food Research.

Andre Esteva, Katherine Chou, Serena Yeung, Nikhil Naik, Ali Madani, Ali Mottaghi, Yun Liu, Eric Topol, Je Dean, and Richard Socher. Deep learning-enabled

medical computer vision.npj Digital Medicine, 4(1):5, 2021. ISSN 2398-6352. doi: https://doi.org/10.1038/s41746-020-00376-2.

Amitkumar N Gajjar and Jigneshkumar Jethva. Intersection over union based analysis of image detection/segmentation using cnn model. In 2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T), pages 1 6, 2022. doi: https://doi.org/10.1109/ICPC2T53885.2022.9776896.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org .

Yunusa Haruna, Shiyin Qin, Abdulrahman Hamman Adama Chukkol, Abdulganiyu Abdu Yusuf, Isah Bello, and Adamu Lawan. Exploring the synergies of hybrid convolutional neural network and vision transformer architectures for computer vision: A survey. Engineering Applications of Arti cial Intelligence, 144: 110057, March 2025. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2025.110057.

Kaiming He, Ross Girshick, and Piotr Dollar. Rethinking imagenet pre-training. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 4917 4926, 2019. doi: https://doi.org/10.1109/ICCV.2019.00502.

Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In International conference on machine learning, pages 2712 2721. PMLR, 2019. doi: https://doi.org/10.48550/arXiv.1901.09960.

Elia Henrichs, Tanja Noack, Ana María Pinzon Piedrahita, María Alejandra Salem, Johnathan Stolz, and Christian Krupitzer. Can a byte improve our bite? an

analysis of digital twins in the food industry. Sensors, 22(1), 2022. ISSN 1424-8220. doi: https://doi.org/10.3390/s22010115.

Shuai Hua. Vehicle tracking based on historical intersection over union. Master's thesis, San José State University, 2018.

Bootan Hussein and Shareef Shareef. An empirical study on the correlation between early stopping patience and epochs in deep learning. ITM Web of Conferences, 64, 07 2024. doi: https://doi.org/10.1051/itmconf/20246401003.

Tingyao Jiang, Cheng Li, Ming Yang, and Zilong Wang. An improved yolov5s algorithm for object detection with an attention mechanism. Electronics, 11(16):2494, 2022. doi: https://doi.org/10.3390/electronics11162494.

Glenn Jocher. YOLOv5 by Ultralytics, May 2020. Last accessed: 20 March 2025.

Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, January 2023. URL https://github.com/ultralytics/ultralytics . Last accessed: 20 March 2025.

Emily Johnson. Deep learning and computer vision for visual security monitoring in devops environments. J. of Art. Int. Research, 3(2):173 179, November 2023. URL https://thesciencebrigade.com/JAIR/article/view/403 .

Dana Jox, Christine Borsum, Darius Hummel, Jörg Hinrichs, and Christian Krupitzer. Enhancing dairy processing with machine learning and domain knowledge: A combined analysis of o ine and time series data. Journal of Food Engineering, 391:112423, 2025. ISSN 0260-8774. doi: https://doi.org/10.1016/j.jfoodeng.2024.112423.

Vijay Kakani, Van Huan Nguyen, Basivi Praveen Kumar, Hakil Kim, and Visweswara Rao Pasupuleti. A critical review on computer vision and artificial intelligence in food industry. Journal of Agriculture and Food Research, 2:100033, 2020. ISSN 2666-1543. doi: https://doi.org/10.1016/j.jafr.2020.100033.

Christian Krupitzer and Anthony Stein. Unleashing the potential of digitalization in the agri-food chain for integrated food systems. Annual Review of Food Science and Technology 15(Volume 15, 2024):307 328, 2024. ISSN 1941-1421. doi: https://doi.org/10.1146/annurev-food-012422-024649.

Christian Krupitzer, Tanja Noack, and Christine Borsum. Digital food twins combining data science and food science: System model, applications, and challenges. Processes, 10(9), 2022. ISSN 2227-9717. doi: https://doi.org/10.3390/pr10091781.

Debasis Kumar and Naveed Muhammad. Object detection in adverse weather for autonomous driving through data merging and yolov8. Sensors, 23(20), 2023. ISSN 1424-8220. doi: https://doi.org/10.3390/s23208471.

Meena Laad, Ratan Maurya, and Najeeb Saiyed, editors. Unveiling the Vision: A Comprehensive Review of Computer Vision in AI and ML, 2024. IEEE. doi: https://doi.org/10.1109/ADICS58448.2024.10533631.

Chenghao Li and Chaoning Zhang. Toward a deeper understanding: Retnet viewed through convolution. Pattern Recognition, 155:110625, 2024. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2024.110625.

Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu, and Xiangxiang Chu. Yolov6 v3.0: A full-scale reloading, 2023a. Preprint, doi: https://doi.org/10.48550/arXiv.2301.05586.

Yong Li, Naipeng Miao, Liangdi Ma, Feng Shuang, and Xingwen Huang. Transformer for object detection: Review and benchmark. Engineering Applications of Artificial Intelligence, 126:107021, 2023b. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2023.107021.

Zonghui Li, Yongsheng Dong, Longchao Shen, Yafeng Liu, Yuanhua Pei, Haotian Yang, Lintao Zheng, and Jinwen Ma. Development and challenges of object detection: A survey. Neurocomputing, 598:128102, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2024.128102.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, Computer Vision – ECCV 2024, pages 38–55, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-72970-6. doi: https://doi.org/10.1007/978-3-031-72970-6_3.

Bohan Ma, Fuzhu Li, and Yi Ren. Review of research progress in computer vision. Advances in Engineering Technology Research, 11:558–558, 07 2024. doi: https://doi.org/10.56028/aetr.11.1.558.2024.

Sourav Modak and Anthony Stein. Enhancing weed detection performance by means of genai-based image augmentation. In Alessio Del Bue, Cristian Canton, Jordi Pont-Tuset, and Tatiana Tommasi, editors, Computer Vision – ECCV 2024 Workshops, pages 252–266, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-91835-3. doi: https://doi.org/10.1007/978-3-031-91835-3_17.

Yansong Peng, Hebei Li, Peixi Wu, Yueyi Zhang, Xiaoyan Sun, and Feng Wu. D-FINE: Rede ne regression task of DETRs as ne-grained distribution re nement. In The Thirteenth International Conference on Learning Representations, 2025. doi: https://doi.org/10.48550/arXiv.2410.13842.

Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. Real-time ying object detection with yolov8, 2024. Preprint, doi: https://doi.org/10.48550/arXiv.2305.09972.

Giorgia Salvucci, Federico Pallottino, Leonardo De Laurentiis, Fabio Del Frate, Rossella Manganiello, Francesco Tocci, Simone Vasta, Simone Figorilli, Beatrice Bassotti, Simona Violino, Luciano Ortenzi, and Francesca Antonucci. Fast olive quality assessment through rgb images and advanced convolutional neural network modeling. European Food Research and Technology, 248(5):1395 1405, May 2022. ISSN 1438-2385. doi: https://doi.org/10.1007/s00217-022-03971-7.

Claudio Filipi Gonçalves Dos Santos and João Paulo Papa. Avoiding over tting: A survey on regularization methods for convolutional neural networks. ACM Comput. Surv., 54(10s), September 2022. ISSN 0360-0300. doi: https://doi.org/10.1145/3510413.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of Big Data, 6:60, 2019. doi: https://doi.org/10.1186/s4053701901970.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego,

CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. doi: https://doi.org/10.48550/arXiv.1409.1556.

Jan Stodt, Christoph Reich, and Nathan Clarke. Unified intersection over union for explainable artificial intelligence. In Proceedings of SAI Intelligent Systems Conference, pages 758–770. Springer, 2023. doi: https://doi.org/10.1007/978-3-031-47724-9_50.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 843–852, Los Alamitos, CA, USA, October 2017. IEEE Computer Society. doi: https://doi.org/10.1109/ICCV.2017.97.

tensorflow. Image classification, 08 2019. URL: https://github.com/tensorflow/docs/blob/master/site/en/tutorials/images/classification.ipynb. Last accessed: 20 March 2025.

Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. Machine Learning and Knowledge Extraction, 5 (4):1680–1716, November 2023. ISSN 2504-4990. doi: https://doi.org/10.3390/make5040083.

Yunjie Tian, Qixiang Ye, and David Doermann. Yolov12: Attention-centric real-time object detectors, 2025a. Preprint, To appear in: The Thirty-Ninth Annual Conference on Neural Information Processing Systems (NeurIPS 2025). doi: https://doi.org/10.48550/arXiv.2502.12524.

Yunjie Tian, Qixiang Ye, and David Doermann. Yolov12: Attention-centric real-time object detectors, 2025b. URL https://github.com/sunsmarterjie/yolov12 .

David Tschirschwitz and Volker Rodehorst. Label convergence: De ning an upper performance bound in object recognition through contradictory annotations. In 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) , pages 6848 6857, 2025. doi: https://doi.org/10.1109/WACV61041.2025.00666.

John W. Tukey. Comparing individual means in the analysis of variance. Biometrics, 5(2):99 114, 1949. ISSN 0006341X, 15410420. doi: https://doi.org/10.2307/3001913.

Ultralytics. Tips for best training results, 2023. URL https://docs.ultralytics.com/de/yolov5/tutorials/tips_for_best_training_results/ .

Jayavrinda Vrindavanam, Pradeep Kumar, and Govind Patil. Poultry disease identi cation in fecal images using vision transformer. Medicon Agriculture & Environmental Sciences, 6(1):05 15, January 2024. ISSN 2972-2691. doi: https://doi.org/10.55162/MCAES.06.150.

Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, Advances in Neural Information Processing Systems, volume 37, pages 107984 108011. Curran Associates, Inc., 2024a. doi: https://doi.org/10.48550/arXiv.2405.14458.

Boyuan Wang. Automatic mushroom species classi cation model for foodborne dis-

ease prevention based on vision transformer. Journal of Food Quality, 2022(1): 1173102, 2022. doi: https://doi.org/10.1155/2022/1173102.

Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7464–7475, 2023. doi: https://doi.org/10.1109/CVPR52729.2023.00721.

Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, Computer Vision – ECCV 2024, pages 1–21, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-72751-1.

Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. Annals of Data Science, 9(2):187–212, 2022. ISSN 2198-5812. doi: https://doi.org/10.1007/s40745-020-00253-5.

Sen Wang, Huiping Jiang, Jixiang Yang, Xuan Ma, Jiamin Chen, Zhongjie Li, and Xingqun Tang. Lightweight tomato ripeness detection algorithm based on the improved rt-detr. Frontiers in Plant Science, 15, 2024b. ISSN 1664-462X. doi: https://doi.org/10.3389/fpls.2024.1415297.

Lang Yu, Mengbo Qian, Qiang Chen, Fuxing Sun, and Jiaxuan Pan. An improved yolov5 model: Application to mixed impurities detection for walnut kernels. Foods, 12(3), 2023. ISSN 2304-8158. doi: https://doi.org/10.3390/foods12030624.

Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection. In 2024

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pages 16965 16974, 2024. doi: https://doi.org/10.1109/CVPR52733.2024.01605.

Lili Zhu and Petros Spachos. Support vector machine and yolo for a mobile food grading system. Internet of Things, 13:100359, 2021. ISSN 2542-6605. doi: https://doi.org/10.1016/j.iot.2021.100359.

Lili Zhu, Petros Spachos, Erica Pensini, and Konstantinos N. Plataniotis. Deep learning and machine vision for food processing: A survey. Current Research in Food Science, 4:233 249, 2021. ISSN 2665-9271. doi: https://doi.org/10.1016/j.crfs.2021.03.009.

Zhuofan Zong, Guanglu Song, and Yu Liu. Detrs with collaborative hybrid assignments training. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV) , pages 6725 6735, 2023. doi: https://doi.org/10.1109/ICCV51070.2023.00621.

List of Figures

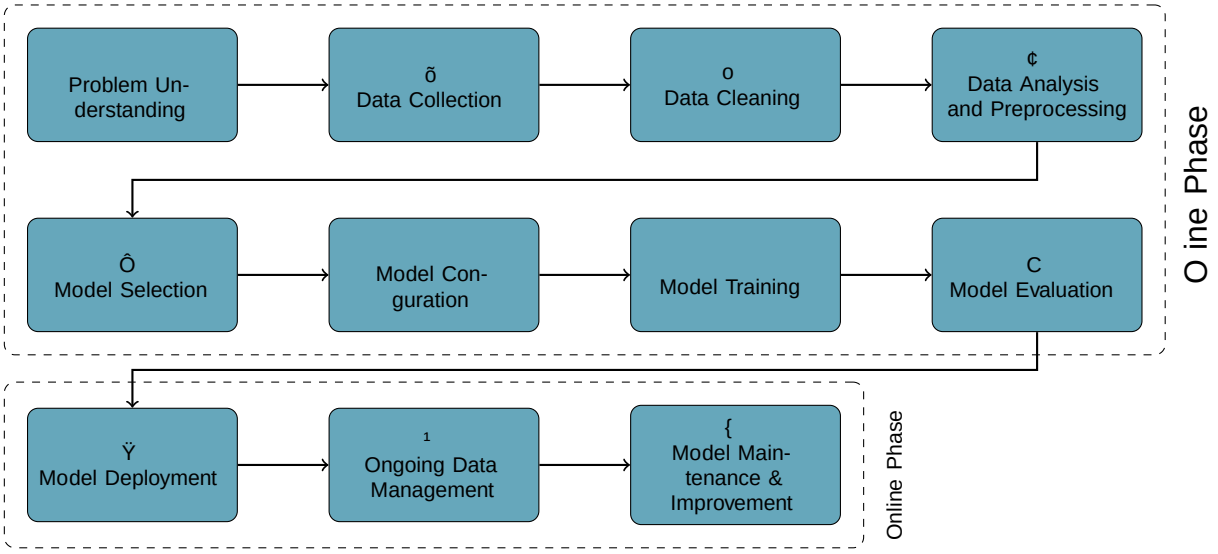


Figure 1: Applied machine learning work ow for the experiments.

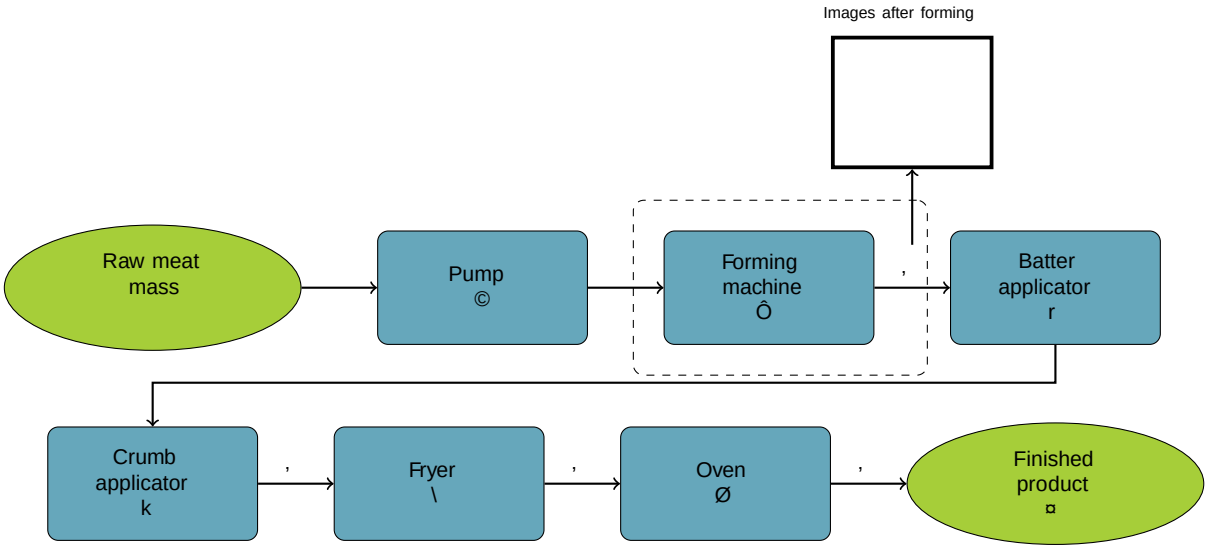


Figure 2: Process line of fried chicken products. The camera symbols () show where images were taken. The dashed box shows that the focus was foremost on the forming production step for this work.

Figure 3: Bounding boxes (red frame around poulty products) generated by GroundingDINO.

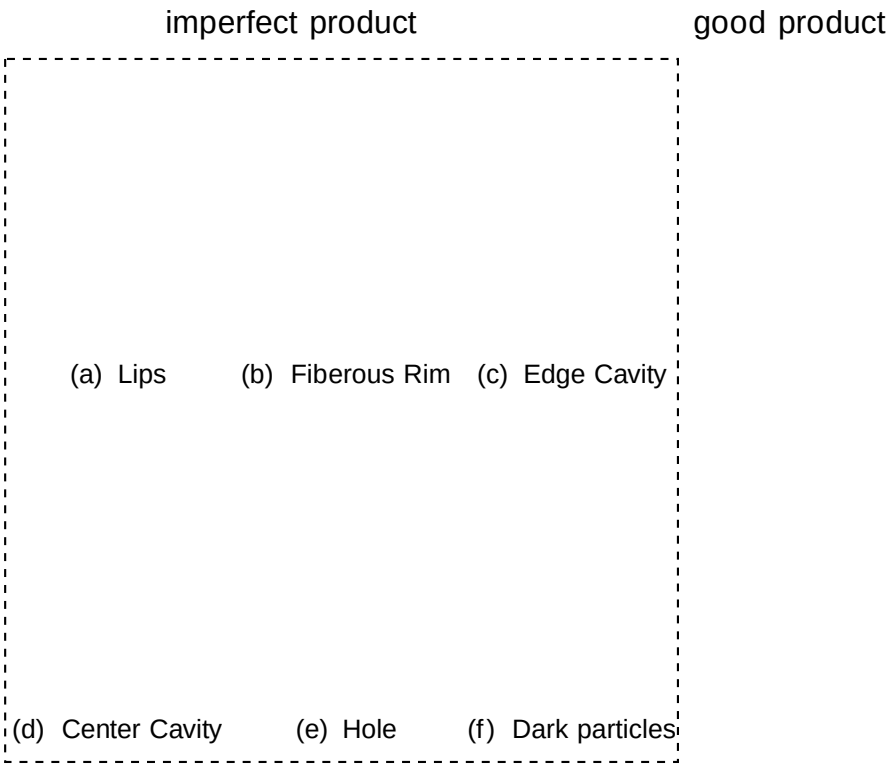
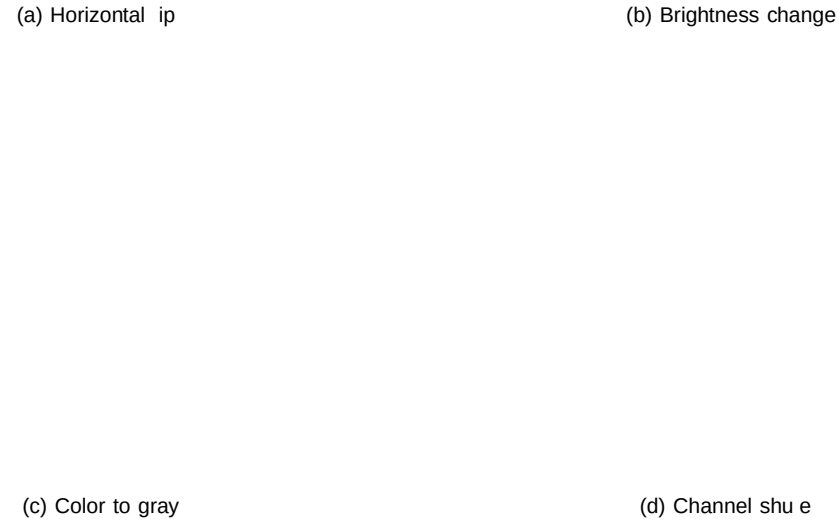


Figure 4: Depiction of a couple of product defects.(a) Lips, which are an unwanted change of the shape,(b) berous rims which are bers extending from the product, (c) edge cavities which are missing lling at the side of the product, (d) center cavities are missing lling in the middle of the product, (e) holes which go through the entire product, and(f) dark particles. On the right, a good product can be seen.



(a) Horizontal ip

(b) Brightness change

(c) Color to gray

(d) Channel shu e

Figure 5: This gure displays four exemplary augmentations (based on originals, which are not shown) that can be applied to increase the size of a dataset, based on Listing 3 in Einsiedel et al. (2025). The rst augmentation in Sub gure 5a is a simple horizontal ip, basically a mirror image of the original. The augmentation in Sub gure 5b changes the brightness/contrast of the original, resulting in a lighter or darker image. Sub gure 5c shows a simple conversion of the image to a grayscale, while Sub gure 5d depicts the random shu ing of color channels.

Figure 6: Structure of the dataset before and after augmentation. A total of 2040 original images was built, containing a total of 43,137 products. Of those products, 14,026 belonged to the good product class and 29,111 to the imperfect product class. Augmentation helped to almost double those numbers.

(a) mAP of YOLOv8x trained from scratch and using pretrained weights

(b) Validation loss of YOLOv8x trained from scratch and using pretrained weights

Figure 7: Comparison of YOLOv8x models trained from scratch (red) and ne-tuned (orange) using pretrained weights. Figure (a) shows the mAP across the training epochs, while Figure (b) shows the validation loss.

Figure 8: YOLOv8x models trained using the included augmentations (orange), custom augmentations (blue), and no augmentations (green).

Figure 9: Comparison of all trained models across a training period of 100 epochs. The model size was set to the biggest available with pretrained weights.

(a) Recall-Con dence curve

(b) Precision-Con dence curve

(c) F_1-score curve

(d) mAP50-95 curve

Figure 10: This gure displays the di erent performance metrics for the ne-tuned YOLO12x model and the respective classes. In 10a the recall of the model is plotted against the con dence, in 10b the same was done with precision. In 10c, the F_1-score is plotted. In 10d, the overall mAP50-95 is plotted over the epochs.

(a) Confusion matrix absolute values

(b) Confusion matrix normalized

Figure 11: This gure displays the confusion matrices obtained after validation. The left gure shows the confusion matrix with absolute values, while the right displays the normalized confusion matrix.