AI-Driven Machine Vision for Automated Defect Classification

Using machine vision to classify defects in real time using a limited dataset

JAKOB PEDERSEN AUGSBURG, KRISTOFER RYDÉN

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

AI-Driven Machine Vision for Automated Defect Classification

Using machine vision to classify defects in real time using a limited dataset

JAKOB PEDERSEN AUGSBURG, KRISTOFER RYDÉN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
EENX30
Chalmers University of Technology
Gothenburg, Sweden 2026

AI-Driven Machine Vision for Automated Defect Classification
Using machine vision to classify defects in real time using a limited dataset
JAKOB PEDERSEN AUGSBURG, KRISTOFER RYDÉN

Supervisor: Sofia Rösberg, Kameleon Robotics
Examiner: Petter Falkman, Department of Electrical Engineering

Master's Thesis 2026
Department of Electrical Engineering
Division of Systems and control
EENX30
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Image of the projects final prototype.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

AI-Driven Machine Vision for Automated Defect Classification
Using machine vision to classify defects in real time using a limited dataset
JAKOB PEDERSEN AUGSBURG, KRISTOFER RYDÉN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Visual inspection is a critical part of modern food production, where product quality and safety must be ensured at high speed and with high reliability. Manual inspection is time-consuming and prone to human error, which motivates the use of automated machine vision systems. The objective for this project was to investigate whether a camera-based system combined with deep learning could be a suitable solution for inspecting vacuum-sealed sausage packages. This was done by designing and implementing a prototype inspection system.

The final system consists of multiple industrial cameras connected to an embedded computing platform, where images of the packages are captured and processed. With these multiple images create a overview of the product . A convolutional neural network is used to analyze the images in order to assess if the package is defect. The network is trained on a dataset collected during the project and is designed to perform the inspection automatically in real time.

The results show promising performance for the use of convolutional neural networks in this type of industrial inspection task, as the system is able to detect relevant visual features and handle variations in lighting and positioning. However, there is still room for improvement, particularly in terms of dataset size, camera setup, and further optimization of the network architecture.

This project concludes that a camera-based inspection system using deep learning is a viable solution for automated quality control of packaged food products in an industrial environment.

Keywords: machine vision, industrial inspection, deep learning, convolutional neural network, cameras, quality control, real-time processing, dataset.

Sammandrag

Visuell inspektion är en viktig del av modern livsmedelsproduktion, där produktkvalitet och säkerhet måste säkerställas i hög hastighet och med hög tillförlitlighet. Manuell inspektion är tidskrävande och känslig för mänskliga fel, vilket motiverar användningen av automatiserade inspektionssystem. Syftet med detta projekt var att undersöka huruvida ett kamerabaserat system kombinerat med djupinlärning kan vara en lämplig lösning för inspektion av vakuumförpackade korvpaket. Detta gjordes genom att utforma och implementera ett prototypsystem för inspektion.

Det slutliga systemet består av flera industriella kameror anslutna, där bilder av paketen tas och bearbetas. Med hjälp av dessa flera bilder skapas en överblick av produkten. Ett konvolutionellt neuralt nätverk används för att analysera bilderna och klassificera om det är defekt eller ej. Nätverket tränas på ett dataset som samlades in under projektet och är utformat för att utföra inspektionen automatiskt i realtid.

Resultaten visar lovande prestanda för användningen av konvolutionella neurala nätverk i denna typ av industriell inspektionsuppgift, då systemet kan identifiera relevanta visuella egenskaper och hantera variationer i belysning och placering. Det finns dock fortfarande utrymme för förbättring, särskilt vad gäller datamängdens storlek, ljuskonfiguration och ytterligare optimering av nätverksarkitekturen.

Detta projekt drar slutsatsen att ett kamerabaserat inspektionssystem med djupinlärning är en genomförbar lösning för automatiserad kvalitetskontroll av förpackade livsmedelsprodukter i en industriell miljö.

Nyckelord: maskinseende, realtid inspektion, konvolutionellt neutralt nätverk, kvalitetskontroll, automatiserad inspektion, defektdetektion, dataset.

Acknowledgements

We would like to extend our sincere gratitude to our supervisor, Sofia Rösberg, and our colleagues at Kameleon Robotics for their valuable insights, support, and for providing an excellent working environment. We also wish to thank our examiner, Petter Falkman, for his guidance and encouragement, without which this project would not have been possible.

Furthermore, we would like to thank Rybergs for allowing us to use their sausage packaging, which contributed to creating a realistic dataset. TR Electronics played a crucial role in the project by lending us cameras and lighting equipment. We are also grateful to Carryline for providing the conveyor belt and sharing their expertise in conveyor systems.

Nomenclature

Abbreviations

Abbreviation	Description
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
FOV	Field of View
FPS	Frames per second
GigE	Gigabit Ethernet
GPIO	General-Purpose Input/Output
GPU	Graphics Processing Unit
KNN	k-Nearest Neighbors
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
NOK	Not OK (defective package)
OK	Conforming package
PoE	Power over Ethernet
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue (colour camera)
SGD	Stochastic Gradient Descent
SGDM	Stochastic Gradient Descent with Momentum
SVM	Support Vector Machine
WA	Working Area
WD	Working Distance

Mathematical Symbols

Symbol	Description
α_v	Trainable parameters/view
c_{FN}	Cost for predicting False Negative
c_{FP}	Cost for predicting False Positive
c_v	Confidence
C	Number of classes
f	Focal length
f_v	Feature vector/view
k	Convolution kernel
$\mathcal{L}$	Cross-entropy loss
Ω	Pooling region / Ohm
p	Probability
$S_{x,y}$	Sensor size (horizontal/vertical)
T	Temperature parameter
V	View vector
w	Weight
x	Input feature map
y	Output feature map
y_c	True label for class c
$\hat{y}_c$	Predicted probability for class c
z	Raw logit

Contents

List of Figures

List of Tables

1
Introduction

Automated quality control is an increasingly important component of modern food-processing production lines. High throughput requirements, strict hygiene regulations, and demands for consistent product quality place significant pressure on inspection processes. In many existing facilities, visual inspection of packaged products is still performed manually by human operators. Although this approach provides flexibility, it is prone to human error, and results in inconsistent inspection accuracy over time.

1.1 Background

This thesis is conducted in collaboration with **Kameleon Robotics AB** and focuses on the analysis, design, and development of an automated system for defect detection in a food-processing production line. The project is motivated by the current dependence on manual visual inspection, in which operators continuously monitor products on a conveyor belt and remove defective items. This procedure limits throughput, increases operational costs, and introduces subjective decision-making.

The primary product investigated in this work is vacuum-sealed sausage packages, although the developed solution targets this specific product, the underlying methodology is designed to be modular and scalable, enabling future adaptation to other packaged food products with similar geometric or visual characteristics.

1.2 Purpose

The purpose of this thesis is to develop and evaluate an automated vision-based system capable of detecting defects in vacuum-sealed sausage packages. The system aims to replace or complement manual inspection by providing consistent, objective, and repeatable quality assessments.

At the end of the thesis, a validated proof-of-concept system is expected to be demonstrated. The system will be evaluated with respect to classification accuracy, robustness to variations in operating conditions such as lighting and product placement, and feasibility for industrial deployment.

As the project involves both a scientific research perspective and commercial considerations relevant to the company Kameleon, both aspects must be taken into account throughout the development and evaluation of the proposed solution.

Another objective of this project is to determine if a small dataset can achieve satisfactory precision in the classification of sausage packages. Since the available dataset is limited, the main focus lies in identifying and optimizing the most suitable model for the task.

1.3 Delimitations

All testing and validation of the proposed system will be conducted in a controlled laboratory environment at Kameleon Robotics AB. Consequently, the system will not be evaluated under full-scale industrial production conditions, such as continuous long-term operation, varying factory environments, or operator-induced variability.

The scope of the project is limited to vision-based inspection of vacuum-sealed sausage packages and focuses on alignment-related defects but also defects on the label on the packages. Other types of defects and alternative inspection modalities, including X-ray, ultrasound, or weight-based methods, are not considered. The project is based on previously published learning-based inspection methods [1], [2]. The work is limited to adapting and evaluating these approaches for the specific industrial application rather than developing new machine learning algorithms from first principles.

1.4 Problem Specification

The system to be developed shall be capable of inspecting vacuum-sealed sausage packages on a conveyor belt and classifying each package as approved or defective based on visual appearance.

Inspection process must operate under conditions representative of an industrial production line, including variations in product placement and minor changes in lighting. Preferably, the system perform in real time at a speed compatible with conveyor-based production without interrupting the production flow.

The output of the inspection system shall be a discrete classification decision. The system shall provide consistent and repeatable classification results and be robust to common variations in packaging appearance. Furthermore, the solution shall be designed with modularity in mind to allow adaptation to similar packaged food products in the future, but also investigating different techniques and controlled environment to handle a small dataset.

1.5 Research Questions

Based on the defined purpose and the specification of the problem, the following research questions are addressed:

- Is convolutional neural networks suitable for detecting alignment-related defects in vacuum-sealed food packages?
- How can the proposed inspection system be designed to be modular and scalable for adaptation to other packaged food products?
- Which defects is possible to detect with the system?
- What techniques can handle a limited dataset?

1.6 Previous Work

Automated defect detection using machine vision and learning-based methods has been widely studied as an alternative to manual inspection in high-throughput industrial environments. Manual visual inspection is typically subjective, labor-intensive, and prone to fatigue-related errors, which motivates the development of automated systems capable of delivering objective, repeatable, and scalable quality assessment.

Deep learning, and in particular convolutional neural networks (CNNs), has become a common approach for image-based defect detection. CNNs are designed to learn hierarchical feature representations directly from raw image data, capturing low-level features such as edges and textures in early layers, and progressively learning

more complex patterns such as shapes and object structures in deeper layers [3]. This ability to automatically learn relevant features reduces the need for manual feature engineering, which has traditionally been a major limitation of classical image processing approaches.

Arshaghi et al.[1] investigate the use of convolutional neural networks (CNNs) to detect and classify surface defects in food products. Using a dataset of approximately 5 000 images, their results demonstrate that CNN-based approaches can achieve high classification accuracy by automatically learning relevant shape and texture features, without the need for handcrafted feature extraction. The study highlights the suitability of deep learning methods for identifying subtle visual irregularities in food products. However, the work focuses primarily on isolated products under controlled imaging conditions and does not address real-time deployment, integration with production lines and sorting of defective items.

In addition to individual case studies, broader survey papers further strengthen the motivation for using CNNs in defect detection. Ren et al. [4] and Yang et al. [5] reviewed a wide range of industrial inspection applications and found that CNN-based approaches consistently provide improved robustness to variations in lighting conditions, viewpoints, and surface appearance. These factors are particularly important in real-world industrial environments, where conditions are rarely controlled and consistent.

These studies emphasize the importance of data quality, segmentation strategies, and training procedures to achieve robust performance under varying conditions. These insights are relevant for automated inspection systems that operate in industrial environments, where variations in lighting, positioning, and product appearance are unavoidable.

This thesis builds on established learning-based inspection methods while focusing on the practical challenges of detecting alignment deviations and other defects in packaged food products and integrated within a industry setting system with a relatively small dataset.

2

Theory

This chapter presents the theoretical background required to understand the methods and design choices proposed in this thesis. The focus is on concepts related to automated visual inspection, image-based defect detection, and learning-based classification methods, with particular emphasis on their applicability in industrial food-processing environments.

2.1 Theoretical Background of CNN-based Inspection

CNNs perform well in this context because they extract local spatial correlations using shared convolutional kernels, reducing the number of learnable parameters compared to fully connected networks. The convolutional layers learn translation-equivariant feature representations, which are important for product positioning variability. Pooling layers further improve invariance to spatial shifts while reducing computational complexity.

Deep hierarchical feature learning enables detection of progressively abstract information:

- Edges and color gradients
- Shapes and contours
- Characters and printed text
- Complete package structures

This hierarchical learning is particularly suitable for detecting packaging defects where both local defects (e.g., misprints) and global defects (e.g., misalignment) shall be recognized[6].

2.1.1 Fundamental Operations in Convolutional Neural Networks

A CNN consists of several types of layers, mainly convolutional layers, activation functions, pooling layers, and fully connected layers. The core operation is the convolution, which applies a small filter (kernel) to an input image or feature map to extract local patterns.

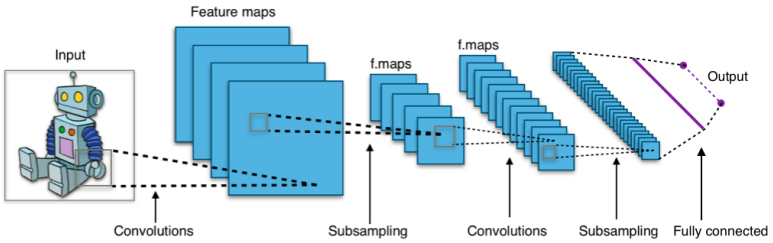


Figure 2.1: CNN architecture. The network extracts hierarchical features from low-level edges to high-level package structures.[7]

The convolution operation can be written as:

$$y(i,j) = \sum_m \sum_n x(i+m, j+n) \cdot k(m,n) \tag{2.1}$$

where x is the input feature map, k is the convolution kernel, and y is the resulting output feature map. This operation allows the network to detect local structures such as edges, corners, and textures, which are essential for visual inspection tasks [8].

After convolution, a nonlinear activation function is applied. A commonly used activation function is the Rectified Linear Unit (ReLU), defined as:

$$f(x) = \max(0, x) \tag{2.2}$$

ReLU introduces nonlinearity into the network, enabling it to model complex relationships and improving training efficiency [5].

To reduce the spatial size of feature maps and the computational cost, pooling layers are used. A typical choice is max pooling, which selects the maximum value within a local region:

$$y(i,j) = \max_{(m,n)\in\Omega} x(i+m, j+n) \tag{2.3}$$

where Ω defines the pooling window. Pooling improves robustness to small spatial shifts and helps reduce overfitting by limiting the amount of learned parameters [4].

Padding, used to generate zero valued pixels around an images border, is often applied before convolution to ensure that border pixels are treated fairly and that important information near the image edges is not lost. This is particularly important in industrial inspection images, where defects may appear close to the image boundaries [8].

2.2 Base Networks

Several convolutional neural network (CNN) architectures have been proposed to improve classification performance while balancing computational efficiency. Modern CNN architectures differ primarily in depth, connectivity patterns, and computational optimizations. This section briefly describes commonly used architectures including VGG, ResNet, MobileNet, and EfficientNet as well as a smaller basic CNN.

2.2.1 Custom Convolutional Neural Networks

In addition to established architectures, custom convolutional neural networks (CNN) are commonly used when designing models tailored to specific tasks and datasets. A custom CNN typically consists of a sequence of convolutional layers, activation functions, pooling layers, and fully connected layers. These components enable the network to extract hierarchical features from input images, ranging from low-level edges and textures to higher-level object representations [9].

Custom CNN architectures offer flexibility in terms of network depth, filter sizes, and number of channels. This flexibility allows the architecture to be adapted to specific constraints such as limited datasets, computational efficiency, or real-time requirements. Compared to large pretrained architectures, smaller custom CNNs may reduce the risk of overfitting when training data is limited and can also improve inference speed.

Furthermore, lightweight CNN architectures have been shown to perform competitively in simpler datasets such as MNIST while requiring fewer parameters and lower computational cost. [10].

2.2.2 VGG Networks

Deeper VGG variants, such as VGG13 and VGG16, increase representational capacity by stacking additional convolutional layers [11]. Batch normalization variants, such as VGG13-BN and VGG16-BN, improve training stability and convergence speed by normalizing intermediate feature activations during training [12]. Despite their relatively high computational cost, VGG networks remain widely used as baseline architectures in image classification and industrial defect detection due to their strong performance and simple architecture [5], [8].

2.2.3 Residual Networks (ResNet)

Residual Networks (ResNet) were proposed by He et al. [13] to address the degradation problem observed in deep neural networks. As network depth increases, training becomes more difficult due to vanishing gradients and optimization challenges. ResNet introduces residual connections, also known as skip connections, which allow gradients to propagate directly through the network.

These residual connections enable the training of significantly deeper networks while improving performance and training stability. Common ResNet variants include $ResNet_{18}$, $ResNet_{34}$, and $ResNet_{50}$, which differ in depth and complexity. Deeper variants generally provide improved feature extraction capabilities at the cost of increased computational requirements [5], [13].

2.2.4 MobileNet

MobileNet architectures were developed for resource-constrained environments such as mobile and embedded systems [14]. These networks use depthwise separable convolutions, which factorize standard convolutions into depthwise and point-wise operations. This significantly reduces computational cost and model size while maintaining competitive accuracy.

MobileNet architectures are designed to reduce computational cost while maintaining competitive classification performance through the use of depthwise separable convolutions and other architectural optimizations [14]. Variants such as $MobileNet_{V2}$ and $MobileNet_{V3}$ further improve efficiency and accuracy, making them particularly suitable for real-time applications where low latency and limited computational resources are important[5], [14].

2.2.5 EfficientNet

EfficientNet architectures were introduced by Tan and Le [15] and use a compound scaling method that balances network depth, width, and input resolution. This approach enables EfficientNet models to achieve improved accuracy while maintaining computational efficiency.

EfficientNet variants, including $EfficientNet_{B0}$, $EfficientNet_{B1}$, and $EfficientNet_{B2}$, provide different trade-offs between performance and computational cost. These models are designed to scale efficiently across various hardware constraints while maintaining strong classification performance[15].

2.3 Fusion Models

The fusion model combines the outputs from multiple CNN-based view models, which in this project consist of the above, side, and under models. The fusion process is based on the raw logit outputs from each CNN, where the logits represent the non-normalized decision signals of the networks and indicate both the predicted class direction (OK or NOK) and the confidence strength before conversion into probabilities.

$$z_v \tag{2.4}$$

where $v \in \{view_1, ..., view_n\}$. A temperature scaling is introduced to calibrate the raw logit values produced by the view-specific convolutional neural networks before fusion. Since independently trained models may exhibit different confidence characteristics, directly using the raw logits could result in certain views producing overconfident predictions that disproportionately influence the final package-level classification. To address this, a temperature parameter T_v is applied to each logit according to

$$z'_v = \frac{z_v}{T_v} \tag{2.5}$$

before converting the scaled logits into probabilities using the sigmoid function.

The temperature values are optimized on the validation set while keeping the original convolutional neural network weights fixed. The optimal temperature parameter is obtained by minimizing the binary cross-entropy loss between the predicted probabilities and the ground-truth labels:

$$T_v^* = \arg\min_{T_v} \mathcal{L}(\sigma(z_v/T_v), y) \tag{2.6}$$

This calibration process improves the reliability of the predicted probabilities and confidence estimates by ensuring that the confidence levels more accurately reflect the true likelihood of correct classification. Consequently, the fusion model can combine predictions from different viewpoints in a more balanced and robust manner.

The calibrated logits are then converted into probabilities using the sigmoid function:

$$p_v = \sigma(z'_v) = \frac{1}{1 + e^{-z'_v}} \tag{2.7}$$

where p_v represents the calibrated probability that the package belongs to the NOK class.

A confidence measure is additionally computed for each view according to

$$c_v = 2 \cdot |p_v - 0.5| \tag{2.8}$$

where values close to 1 indicate high confidence and values near 0 indicate uncertainty. The confidence measure therefore represents the distance from the decision boundary.

To account for differences in reliability between views, the fusion model learns adaptive weights for each camera perspective. These weights are defined through a softmax normalization:

$$w_v = \frac{e^{\alpha_v}}{\sum_{u \in V} e^{\alpha_u}} \tag{2.9}$$

where α_v are trainable parameters and $V = \{view_1, ..., view_n\}$. The weighting mechanism enables the fusion model to automatically emphasize views that provide more informative or reliable predictions.

The scalar values from each view are then subsequently concatenated and processed by a lightweight feedforward neural network consisting of fully connected layers, nonlinear activation functions, and dropout regularization. Unlike convolutional neural networks, which are designed for spatial feature extraction from image data, the feedforward network operates on the already extracted prediction-level information and learns how to combine the different view-specific outputs into a final package-level classification. This enables the fusion stage to model relationships between the different camera perspectives while maintaining low computational complexity.

Compared with early fusion approaches, where raw image data are combined prior to feature extraction, late fusion offers greater architectural flexibility by allowing each model to operate at its optimal resolution and configuration. This is especially advantageous in industrial inspection environments where different viewpoints may exhibit varying lighting conditions, textures, and feature relevance. In addition, late fusion is computationally efficient and well suited for real-time applications, as view-specific predictions can be processed independently and combined through a lightweight classification stage [16], [17], [18].

2.4 Challenges of Limited Datasets

When working with deep learning models, performance generally improves with larger datasets, as models are able to learn more representative and generalized features. However, in industrial inspection tasks, collecting large labeled datasets is often challenging due to limited availability of defective samples, labeling costs, and production constraints. Models trained on small datasets are more prone to overfitting and unstable convergence.

To mitigate these challenges, careful model selection and hyperparameter tuning become increasingly important. Factors such as network architecture, optimizer choice, data augmentation, and regularization strategies can significantly influence performance when data is limited. Therefore, systematic evaluation of different configurations is commonly employed to identify the most suitable model for the task.

2.4.1 Training and Optimization

Training deep neural networks involves iteratively updating the weight parameters to minimize a loss function. Classic stochastic gradient descent (SGD) applies a fixed learning rate to all parameters and can converge slowly or require careful tuning. Adam (Adaptive Moment Estimation) is an adaptive method that overcomes these limitations by maintaining two exponentially weighted moving averages: one of the gradients (the first moment) and one of the squared gradients (the second moment). These moment estimates are corrected for their initial bias and used to compute an individual step size for each parameter. Adam combines strengths of AdaGrad and RMSProp, which work well with sparse gradients and non-stationary data. Because it adjusts the learning rate based on both mean and variance of past gradients, the optimizer requires little tuning, is memory-efficient, and robust to noisy or sparse updates.

However, Adam can suffer from suboptimal generalization performance due to the way weight decay is implemented as L2 regularization within the gradient-based update. AdamW addresses this limitation by decoupling weight decay from the optimization step. Instead of incorporating weight decay into the loss gradient, AdamW applies weight decay directly to the weights during parameter updates. This decoupled formulation prevents the adaptive learning rate mechanism from interfering with regularization, leading to improved generalization and more stable training. Additionally, AdamW makes the choice of learning rate and weight decay more independent, simplifying hyperparameter tuning. Empirical results demonstrate that AdamW consistently outperforms standard Adam across various datasets and training configurations, achieving significantly better generalization performance [19].

The network is trained by minimizing a loss function that measures the difference between the predicted output and the true label. For multi-class classification prob-

lems, the cross-entropy loss is commonly used:

$$\mathcal{L} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c) \tag{2.10}$$

where y_c is the true label and $\hat{y}_c$ is the predicted probability for class c.

During this project, the Rectified Linear Unit (ReLU) activation function was used in all evaluated CNN architectures. ReLU is one of the most commonly used activation functions in image-based inspection and defect detection applications due to its computational efficiency and ability to mitigate the vanishing gradient problem during training. Previous surveys of deep learning methods for industrial defect detection have shown ReLU-based architectures to be widely adopted and effective across a range of inspection tasks [5].

For imbalanced datasets, weighted cross-entropy loss can be used to assign different importance to different classes. This helps the model focus more on underrepresented classes by increasing the penalty for incorrect predictions:

$$\mathcal{L} = -\sum_{c=1}^{C} w_c, y_c \log(\hat{y}_c) \tag{2.11}$$

where w_c is the weight assigned to class c. Higher weights increase the contribution of that class to the total loss, which can improve classification performance for minority classes.

The parameters of the network are optimized using gradient-based methods and backpropagation. Optimizers such as Adam/AdamW are widely used because they provide stable convergence and adaptive learning rates for each parameter [3].

2.4.2 Data Augmentation and Generalization

In industrial applications, collecting large amounts of labeled defect data is often difficult and time-consuming. Survey studies and practical applications both emphasize the importance of data augmentation to address this problem [4], [8]. Data augmentation artificially increases the size of the training dataset by applying transformations such as rotation, flipping, scaling, and brightness changes.

By training the network on augmented data, the model becomes more robust to variations in object position, orientation, and lighting conditions. This improves generalization performance and reduces the risk of overfitting, which is especially important when the number of available defect samples is limited [5].

2.5 Performance Metrics

For a binary classification problem, the outcomes of predictions can be summarized using a confusion matrix:

$$\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix} \tag{2.12}$$

where each entry describes the number of samples falling into a specific category:

- **True Positive (TP):** a positive sample correctly classified as positive,
- **True Negative (TN):** a negative sample correctly classified as negative,
- **False Positive (FP):** a negative sample incorrectly classified as positive,
- **False Negative (FN):** a positive sample incorrectly classified as negative.

Precision is defined as the proportion of predicted positive samples that are actually positive:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.13}$$

Recall (also called sensitivity) is defined as the proportion of actual positive samples that are correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.14}$$

Both metrics take values in the range $[0, 1]$ and are particularly useful for imbalanced datasets, where the majority class may otherwise dominate the evaluation. The F1-score combines precision and recall into a single metric using their harmonic mean:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.15}$$

The analogous metrics can be defined for the **negative class**. The Negative Predictive Value (also called Negative Precision) is defined as the proportion of predicted negative samples that are actually negative:

$$\text{Precision}_{\text{neg}} = \frac{TN}{TN + FN} \tag{2.16}$$

The Specificity (also called True Negative Rate or Negative Recall) is defined as the proportion of actual negative samples that are correctly identified:

$$\text{Recall}_{\text{neg}} = \frac{TN}{TN + FP} \tag{2.17}$$

The corresponding F1-score for the negative class combines these two metrics via their harmonic mean:

$$F1_{\text{neg}} = 2 \cdot \frac{\text{Precision}_{\text{neg}} \cdot \text{Recall}_{\text{neg}}}{\text{Precision}_{\text{neg}} + \text{Recall}_{\text{neg}}} \tag{2.18}$$

2.6 Effects of Camera Sensor Properties on Vision Performance

Machine vision systems rely on camera sensors, whose characteristics significantly influence the performance and suitability of the system for a given application. Important sensor properties include the **shutter type**, **sensor size**, and **color capability**.

2.6.1 Sensor Size and Working Distance

Sensor size describes the physical size of the camera sensor's active area, which is responsible for capturing incoming light and forming the image. Which is also correlated to the cameras field of view, a larger sensor size yields a greater field of view.

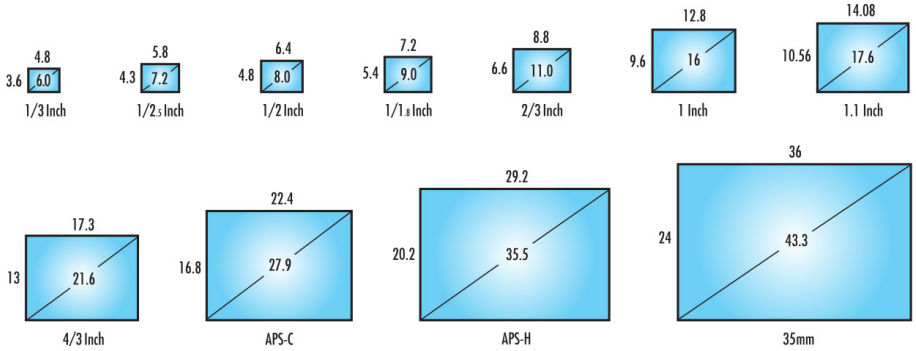


Figure 2.2: Typical sizes for machine vision cameras sensors. In millimeters. Picture taken from Edmund Optics.[20]

The camera sensor size is an important parameter for selecting the appropriate lens focal length, as it determines the relationship between the working distance and the camera's field of view. The working area (WA) refers to the physical region captured by the camera at a given working distance (WD), which is the distance between the camera and the object of interest. The focal length f can be calculated using a pinhole camera model and fixed focal length lens:

$$f = \frac{S_{x,y} \cdot \mathrm{WD}}{\mathrm{WA}_{x,y}} \tag{2.19}$$

where $S_{x,y}$ is the sensor size along the horizontal or vertical axis, $\mathrm{WA}_{x,y}$ is the corresponding width or height of the working area [21].

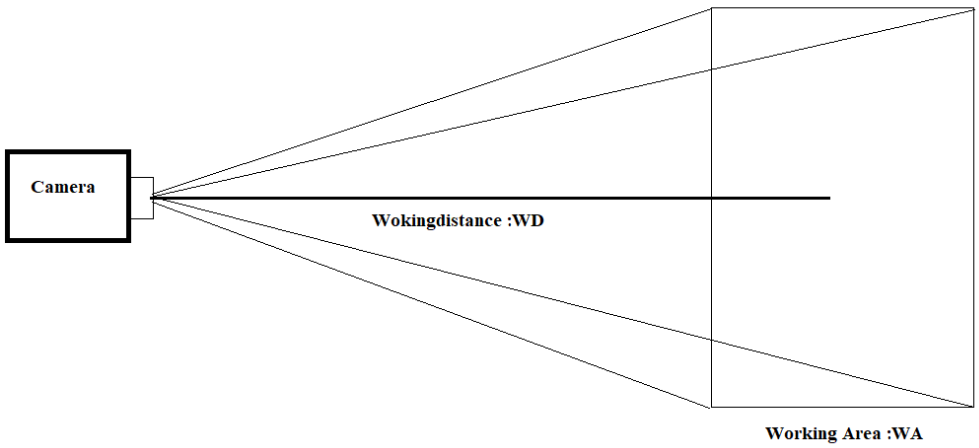


Figure 2.3: Simple picture of working distance and working area.

2.6.2 Shutter Type

The cameras considered for this application use electronic shutters. The two main shutter technologies are **global shutter** and **rolling shutter** [22]. The difference between them lies in how the sensor pixels are read out. With a global shutter, all sensor pixels are exposed and read out simultaneously, resulting in an image that represents a single instant in time. In contrast, a rolling shutter reads out the sensor row by row, causing each row to be captured at a slightly different time.

This behavior can lead to what are known as *rolling shutter artifacts*, where fast-moving objects appear distorted because the sensor is still being read while the object is moving. An example of this effect is shown in Figure 2.4.

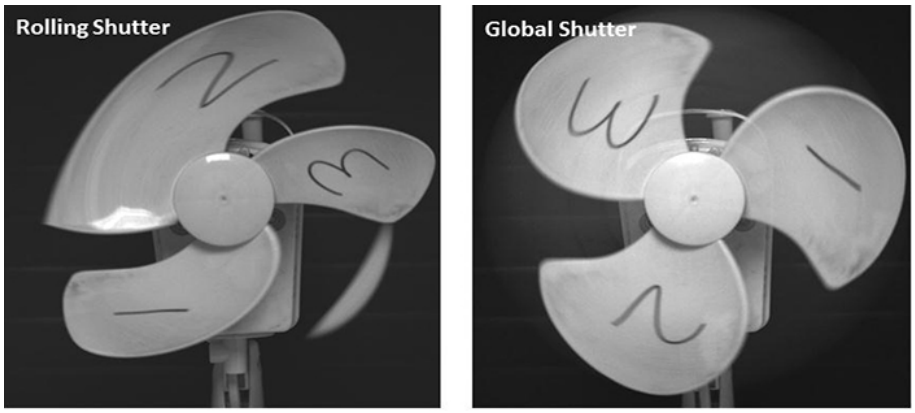


Figure 2.4: The image on the left illustrates rolling shutter artifacts, while the image on the right demonstrates the advantage of a global shutter [23].

However, global shutters also have certain drawbacks. Since they typically rely on a limited number of analog-to-digital converters, the sensor readout speed can be restricted, resulting in lower frame rates and longer duty cycles [22]. Consequently, a global shutter may be limiting in applications that require capturing a large number of images in rapid succession. Nevertheless, for applications involving fast-moving objects, the elimination of rolling shutter artifacts often makes a global shutter the preferred choice.

2.6.3 Difference Between Color and Monochromatic Cameras

When designing a machine vision system, a choice must be made between using a color camera and a monochrome camera. The most common approach for one-shot digital color imaging is to place a color filter array in front of the image sensor. This filter assigns each pixel to one of three color channels in an alternating pattern, most commonly the *Bayer* pattern [24], as shown in Figure 2.5.

The final color image is reconstructed from the sampled color information through an interpolation process known as *demosaicing*. Compared to monochrome cameras, this approach has two main disadvantages. First, the color filters block a significant portion of the incident light, reducing the sensor's sensitivity and resulting in lower image quality and resolution. Second, the demosaicing process introduces additional computational overhead, which can limit the achievable frame rate [24].

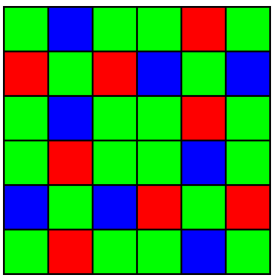


Figure 2.5: Example of a Bayer color filter array [25].

2.7 Effects of Illumination on Machine Vision Performance

A crucial part of a machine vision system is proper use of illumination, where it is needed to take in consideration of both the product being inspected but also the environment the system will be placed in. As both ambient light contribution and also the areas physical requirements, sometimes there simply are not space for the intended solution or that pre-existing support systems as a robot picker will be in the way. While designing a system, important considerations are **working distance**, **glare**, **different wavelengths of light**. These concepts will be explained with a

Led light as the source, as some of these concepts change with other sources of light.

2.7.1 Working Distance

The working distance, here it is the distance from the light source to the object of interest , and the intensity distribution at that working distance are important, as they directly affect the system's capability to handle ambient light sources. With good illuminance and intensity distribution, it is possible to use high-power strobing to wash out ambient light sources while by being able to reduce the exposure time the camera requires, thereby canceling out ambient light such as sunlight or overhead factory lighting [26].

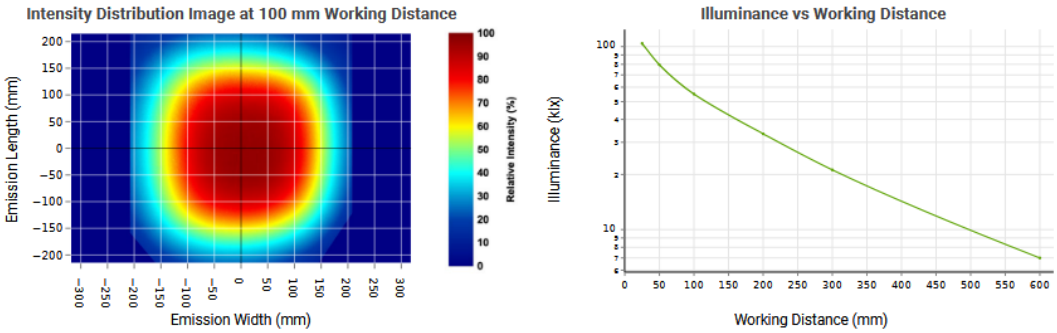


Figure 2.6: The intensity characteristics for 12x12 inch *FD2 High Intensity Backlit Flat Diffuse Light*. [27]

Using a powerful light source allows for shorter exposure times because the sensor can collect more photons in a shorter period. Exposure time is particularly important when capturing moving objects, as a long exposure can cause motion blur but also make it so the sensor is able to collect unwanted ambient lighting.

2.7.2 Different Wavelengths

The different illumination wavelengths affect vision systems performance massively, as each wavelength spectral characteristics work with different surfaces differently. Considering basic color theory and the color wheel, complementary (opposite) colors provide the highest contrast, resulting in more well-defined edges. This is important for certain computer vision applications.

Figure 2.7: Color wheel.

Therefore, illuminating a blue package with red light can improve contrast, where as using red light on a red object would be a poor choice, as illustrated in Figure 2.8.

Figure 2.8: Different wavelengths of illumination used on a paper-stamp with red-text [28].

Inspecting the image further, it becomes clear, as stated before, that green illumination creates the strongest contrast, since it is the complementary color to red. Under red light, the red text becomes nearly invisible.

Additionally, white light is effective as well and perform in a similar way for all the colors [28].

2.7.3 Glare

Complementary color contrast is not the only factor to consider when choosing the appropriate wavelength, as different wavelengths interact with materials differently. For reflective surfaces such as plastic, metal, or glass, controlling glare depends both on lighting geometry, diffusion and on wavelength choice. Longer wavelengths can also reduce glare, as shorter wavelengths are more susceptible to scattering, which can contribute to glare and reduced contrast on reflective surfaces such as vacuum-sealed packaging [29].

Glare is in general an unwanted artifact where the light of the illumination reflects into the camera, to prevent this different techniques can be applied. Commonly used is diffused light, and works by scattering the light rays evenly across the object avoiding hot spots. To further avoid glare different geometry of the illumination can be used, depending of the geometry of the object different geometry of the light should be used, see Figure 2.9 [28].

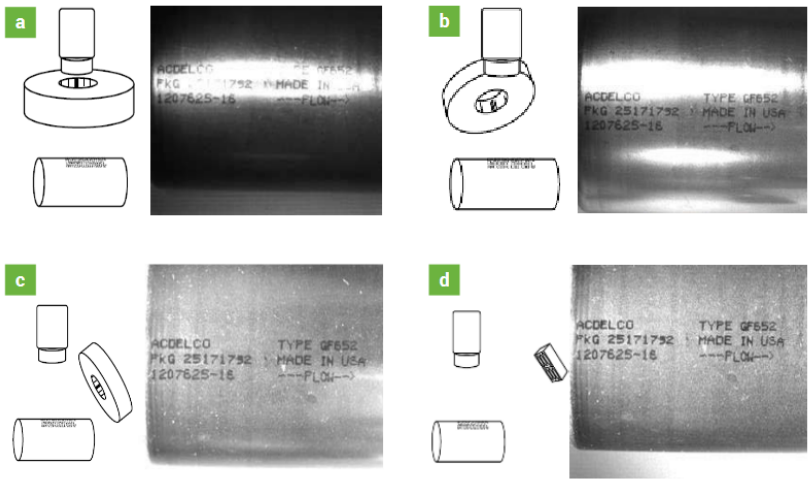


Figure 2.9: Different lighting techniques used, resulting in different glare [28].

Other ways to reduce glare include shortening the exposure time of the sensor, which limits the amount of light captured. Additionally, polarizing filters and different techniques can be applied to both the light source and lens, reducing specular reflections from smooth surfaces. Decreasing the lens aperture reduces the overall light intake of the sensor, which can help prevent overexposed regions caused by intense specular reflections.

3

Experimental Setup

This chapter describes the experimental setup and data acquisition procedure used throughout the project. It presents the different datasets, the industrial vision setup, and the hardware configuration used for image capture, illumination, and synchronized control of the inspection system.

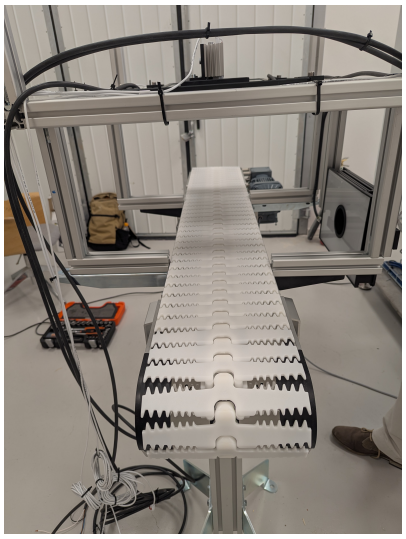
3.1 Data Acquisition Setup

Three primary datasets were collected for this project. The first was acquired using a Logitech C270 Webcam [30] under ambient lighting conditions (Figure 3.1a), with each image captured manually and the package positioned to occupy the majority of the frame.

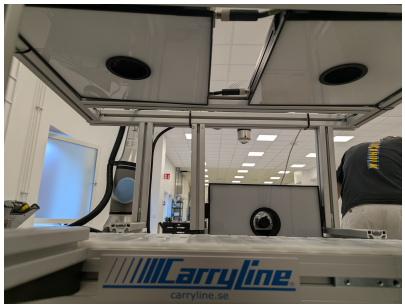
The remaining two datasets, color and monochromatic, were collected using an industrial conveyor belt setup equipped with Basler cameras and flat diffuse illumination. The setup was enclosed within a wooden structure to eliminate ambient light and the whole setup is described in further detail in Section 3.2. Three cameras were mounted approximately 300 mm from the object: two monochrome cameras (one overhead, one lateral) and one RGB camera mounted overhead (Figure 3.1c).

Together, these three datasets allow for different results: the performance of the Basler cameras, the performance of the Webcam under ambient light and the comparison between color and monochromatic camera.

(a) Webcam capture setup (Logitech C270).



(b) Front view of the vision setup.



(c) Side view showing camera and illumination placement. The camera on the left is the color camera.

(d) Test box for ambient light cover.

Figure 3.1: Overview of the data acquisition setups and configurations.

Each dataset contains images captured from three views; above, side, and under as shown in Figure 3.2, and is divided into three subsets accordingly. Images were collected across three distinct imaging environments:

1. A consumer Webcam under ambient lighting.

2. Industrial illumination with ambient light excluded, using monochrome images.

3. Industrial illumination with ambient light excluded, using color images.

Figure 3.2: The three captured views used across all datasets, photographed using a Logitech C270 Webcam.

3.2 Vision Setup

This section describes the industrial vision setup used for image acquisition and synchronized package inspection. The setup includes the conveyor system, camera and lens configuration, illumination design, and the control hardware required to capture images under controlled industrial conditions.

3.2.1 Conveyor Belt

The conveyor belt used to test the vision system under industrial conditions travels at a velocity of 20 m/min. Given that each package occupies a total spacing of 360 mm (comprising the 180 mm package length and an equal gap before the next package), the resulting throughput frequency is:

$$f = \frac{v}{d} = \frac{20\,\text{m/min}}{0.36\,\text{m}} \approx 55.6\,\text{packages/min} \tag{3.1}$$

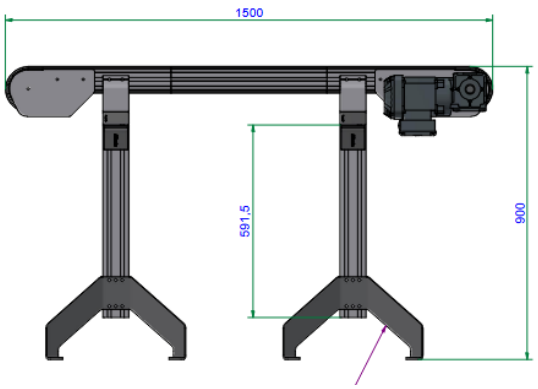


Figure 3.3: The conveyor belt used, made by Carryline.

3.2.2 Camera and Lens

The cameras used in this project are Basler ace 2 a2A1920-51BAS cameras [31], used in both monochrome and color configurations, with the specifications listed in below.

Interface	GigE (RJ45), Power over Ethernet (PoE)
Sensor	Sony IMX392 CMOS Global Shutter
Sensor Format	1/2.3”
Resolution	1920 × 1200 (2.3 MP)
Frame Rate	51 fps
Pixel Size	3.45 μm × 3.45 μm
Sensor Size	6.6 mm × 4.1 mm
Sensor Diagonal	7.77 mm
Shutter Type	Global Shutter

Table 3.1: Specifications of the Basler ace 2 a2A1920-51BAS camera.

The cameras are connected via a GigE interface and powered through Power over Ethernet (PoE), simplifying power delivery and cabling. At 51 fps, the cameras operate well above the throughput frequency of the conveyor belt, see Section 3.2.1. The working distance is set to 300 mm, and the working area is 250 mm × 250 mm, which is larger than the package, see Section 4.2. Using these values, the appropriate focal length is calculated as:

$$f = \frac{S \cdot \text{WD}}{\text{WA}} = \frac{6.6 \text{ mm} \times 300 \text{ mm}}{250 \text{ mm}} \approx 7.9 \text{ mm} \approx 8 \text{ mm} \tag{3.2}$$

where S is the sensor width along the horizontal axis, WD is the working distance, and WA is the corresponding width of the working area of the camera. Based on this calculation, the chosen lens is the TCL 0814 5 MP [32], designed for 5 MP sensors, with a focal length of 8 mm and an adjustable aperture ranging from $f/1.6$ to $f/16$, providing flexibility between light sensitivity and depth of field. The exposure time for all the cameras were set to 550μs.

3.2.3 Illumination

The system employs three light sources of the same model, *MFDOME PLUS* [33], which differ in size and wavelength. One light source has dimensions of 300 mm × 200 mm and operates at a wavelength of 625 nm (red). The remaining two light sources both have dimensions of 300 mm × 200 mm; one operates at 625 nm (red), while the other emits white light.

The choice of red light is motivated by its longer wavelength, which is less prone to scattering and therefore reduces reflections on shiny surfaces such as plastic packaging, while white light is better suited for capturing images in color. This setup

allows a comparison between red illumination with a monochrome camera and white illumination with an RGB camera, in order to evaluate differences in performance across the different views for this application. All light sources use a diffuse lighting technique, producing even illumination and minimizing glare on the plastic packaging. The lights were mounted perpendicular to their assigned camera view, approximately 300 mm from the package, while the cameras are mounted directly behind the lights on mounting plates.

3.2.4 Control

To control the illumination, the GPIO pins of the Jetson AGX Orin are used to strobe the lights in synchronization with the conveyor belt. Since the GPIO pins operate at 3.3 V and are rated for a maximum current of 1 mA, they cannot directly drive the light sources, which require 24 V and draw several amperes. To bridge this gap, an N-channel logic-level MOSFET is used as a switch, triggered by the GPIO signal.

A gate resistor and a pull-down resistor are included to prevent inrush current and floating gate voltage, respectively. Each light source is connected to a dedicated MOSFET. The timing of the strobe is determined by laser sensors mounted along the conveyor belt, which detect passing packages and trigger the GPIO pins via an optocoupler, providing galvanic isolation between the sensor circuit and the Jetson board. The full circuit schematics and component calculations are provided in Appendix B.

4

Method

This chapter presents the methodology used throughout the project, including the data handling, network architectures, training strategies, and fusion-based classification framework. It also describes the automated multi-stage model selection process and the evaluation procedure used to identify robust package classification models for the industrial inspection system

4.1 Main Foci

To address both the scientific and commercial perspectives of the project, two separate code structures have been developed. The first script evaluates a broad range of neural network architectures and performs automated tuning using reasonable parameter configurations, enabling Kameleon to easily apply the framework to other datasets in the future. The second script provides a more detailed evaluation of the individual steps involved in identifying the best-performing model, allowing for clearer insight into which optimization steps contribute positively to performance and which may be unnecessary.

4.2 The Product

The packages used in this project are vacuum-sealed sausage of two variants, Bohusgrillare and Uddevallare, both manufactured by Rydbergs. The packages are identical in dimensions, measuring 180 mm in width, 120 mm in height, and 30 mm in depth. The only visual distinction between them is found on the front face of the packaging, as illustrated in Figure 4.1.

Figure 4.1: The two different packages used in the project.

4.2.1 Fault Representation

Through communication with production staff, a number of different defects were present, see the Table below.

Defect type	Description	Visually observable
Misalignment	Sausages not properly aligned within the package	Yes
Broken sausages	One or more sausages visibly damaged or broken	Yes
Incorrect number of sausages	Too few or too many sausages in the package	Yes
Hole in vacuum-sealed package	Loss of vacuum due to a hole in the packaging	Yes
Plastic stuck on sausage	Packaging film adhering to the sausage surface	Yes
Faulty label	Unreadable date on the package	Yes

Table 4.1: Identified defect types in sausage packages. See Appendix A for images of the defects.

The problem could be framed either as a multi-class classification between all defect types, or as a simpler binary classification into correct or defective packages. For this project, a binary OK/NOK classification is used to simplify the problem and the data collection process. Each package has an OK/NOK class while each image corresponding to that package also has an OK/NOK class. Where the OK class is set to the label of 1 while the NOK class is labeled as 0. So NOK packages are labeled to the negative class while OK is labeled as the positive class, seen in Section 2.5.

4.3 Hardware and Software Platform

The whole system is based upon the *Jetson Orin AGX*[34].

The implementation platform is based on Python using PyTorch, chosen for its GPU support with the embedded hardware but also because it is well documented and has a lot of support. Hardware acceleration, which PyTorch supports using CUDA, supports efficient training and real-time inference on embedded GPUs.

4.4 Data

This chapter presents the datasets, preprocessing methods, and augmentation strategies used throughout the project. It also describes the approaches employed to improve model robustness and reliability when working with limited and imbalanced data.

4.4.1 Data Acquisition

Three datasets were collected during the project using different camera setups and imaging environments, where the details regarding how the datasets were created are explained further in the following sections.

For the Webcam dataset ($\text{Webcam}_{\mathbf{SMALL}}$), a smaller subset was captured in a controlled environment with a white background, while a larger subset ($\text{Webcam}_{\mathbf{BIG}}$), was created by adding images from a conveyor belt setting with a fixed camera position. This allows for an evaluation of the environmental settings needed to receive good output.

For the Basler monochrome datasets, was collected to assess the effect of dataset size. The smaller dataset was created by removing random packages from the training and validation of the bigger dataset. Resulting in a smaller dataset containing approximately 70% of the amount of images as the larger dataset.

The Basler RGB dataset includes only top and bottom views, as the color camera was fixed from above. These RGB images are compared against the larger monochrome dataset to examine the influence of camera type and lighting on model performance.

A common test set was created by collecting, by hand, a collection of packages that represented the different variations available in the data, then the image of those packages were captured by each camera setup to create a fair test set between setups.

4.4.2 Preprocessing

Before training and inference, all images were resized to an image size of $Y = [214, 320, 512]$, Y × Y pixels using bilinear interpolation. Grayscale images were replicated across three channels to match the expected input format of the architecture. Pixel values were scaled to $[0, 1]$ and normalized using ImageNet statistics (mean = $[0.485, 0.456, 0.406]$, std = $[0.229, 0.224, 0.225]$) when a pretrained weights were employed, and using mean = 0.5 and std = 0.5 otherwise.

4.4.3 K-Folds and Seeds

When working with large datasets, train/validation/test splits naturally capture the full variation in the data. With small datasets, however, an unlucky split can result in rare defect variations being underrepresented in the training set. For example, if most examples of a particular defect type end up in the validation split, the model never learns to recognize it, and performance on the test set suffers as a result.

To obtain more reliable results, it is beneficial to train the model across many different data splits. This ensures that any observed performance improvement stems from an actual change in the model, rather than a fortunate or unfortunate data split. To mitigate this risk, a technique called K-Folds cross-validation is employed, which systematically rotates the training and validation sets, allowing the average performance of a model to be computed across multiple folds. In a similar way, an additional approach was also used: the data was split using different random seeds to produce varied train/validation partitions, and this process was repeated to obtain an average performance across multiple seeds.

4.4.4 Data Augmentation

To mitigate the limitations imposed by the small dataset size, four distinct augmentation strategies were evaluated. Augmentation is only done on the training data and not on the validation or test datasets. The first strategy focused exclusively on photometric transformations targeting varied lighting conditions, as detailed in Table 4.2. The second strategy applied geometric transformations, including flips and rotations, as described in Table 4.3. The third strategy combined both photometric and geometric augmentations, as summarized in Table 4.4. Finally, a fourth strategy applied the same combined augmentations at a higher frequency. The augmentation strategy was not applied to the automated model filtering process , shown later in Method, as time with the project became a constraint.

All strategies were configured to allow multiple augmentation operations to affect each image simultaneously. Each augmentation probability is sampled independently per operation, meaning multiple transformations may be applied to a single

image in any given pass. In the first three strategies, each OK sample yielded three augmented copies per original image, while each NOK sample yielded one, partially compensating for class imbalance. The fourth strategy doubled these multipliers, producing six augmented copies per OK sample and two per NOK sample.

Augmentation	**Probability**	**Description**
Brightness	0.5	Random brightness scaling
Contrast	0.5	Random contrast scaling
Gaussian Noise	0.3	Adds random sensor noise
Specular Highlight	0.5	Simulates specular reflection on plastic surfaces

Table 4.2: Photometric augmentation operations (Strategy 1). OK samples receive three augmented copies per original image; NOK samples receive one. Each probability is sampled independently per operation.

Augmentation	**Probability**	**Description**
Horizontal Flip	0.5	Mirrors the image left-to-right
Vertical Flip	0.5	Mirrors the image top-to-bottom
Rotation	0.7	Random rotation within $[-8°, 8°]$

Table 4.3: Geometric augmentation operations (Strategy 2). OK samples receive three augmented copies per original image; NOK samples receive one, partially compensating for class imbalance.

Augmentation	**Probability**	**Description**
Horizontal Flip	0.5	Mirrors the image left-to-right
Vertical Flip	0.5	Mirrors the image top-to-bottom
Rotation	0.7	Random rotation within $[-8°, 8°]$
Brightness	0.5	Random brightness scaling
Contrast	0.5	Random contrast scaling
Gaussian Noise	0.3	Adds random Gaussian noise
Specular Highlight	0.5	Simulates specular reflection on plastic packaging

Table 4.4: Combined photometric and geometric augmentation operations (Strategies 3 and 4). In Strategy 3, OK samples receive three augmented copies per original image and NOK samples receive one. Strategy 4 doubles these multipliers to six and two, respectively.

The implementation of the augmentations are shown in Appendix C.

4.5 Network Architecture

During this project, convolutional neural networks (CNNs) were chosen as the core machine learning method, as they previously have shown effective performance in image-based inspection tasks. The CNNs were used to classify the different views of the package, where the system relies on visual information from cameras to determine whether a package is defective or correct based on its visual characteristics.

To identify the most suitable models for each camera view, an automated model selection framework was developed. The process begins with view-specific image classification, where multiple CNN architectures and hyperparameter configurations are evaluated independently for the above, side, and under views in order to identify the best-performing models for each perspective. These selected view-specific models are then further refined and later combined in a late-fusion framework, where the outputs from the individual CNN classifiers are merged to perform the final package-level classification.

Alternative approaches such as rule-based image processing, traditional machine learning classifiers, and Vision Transformer architectures were considered. Rule-based methods and handcrafted feature approaches were deemed unsuitable due to sensitivity to lighting variations and limited scalability [35].

4.5.1 Base Networks

In this project these networks will be explored:

Table 4.5: Evaluated Network Architectures

Architecture Family	Models
Custom CNN	Custom CNN architectures with convolutional layers; 6, 8 and 10
ResNet	ResNet18, ResNet34, ResNet50
VGG	VGG13-BN, VGG16, VGG16-BN
MobileNet	MobileNetV2, MobileNetV3-Small, MobileNetV3-Large
EfficientNet	EfficientNet-B0, EfficientNet-B1, EfficientNet-B2

Table 4.6: Models evaluated, discussed in Performance Evaluation. Where the basic convolutional neural network will be tested with different amount of neurons and layers.

4.5.2 Pretraining and Freezing Strategize

The models used will be tested with and without pretrained weights, here are the details on how they are frozen and which layers should be trained.

Freezing Strategy	Frozen Layers	Trainable Layers
`classifier_only`	All backbone feature extraction layers	Only the final classification layer (fully connected classifier head) is trainable
`layer4_fc`	Early and intermediate feature extraction layers	Final feature extraction block and classifier head are trainable (e.g. ResNet `layer4` + fully connected layer)
`layer3_layer4_fc`	Early feature extraction layers	The last two major feature extraction blocks together with the classifier head are trainable (e.g. ResNet `layer3`, `layer4`, and fully connected layer)
`none`	No layers frozen	All network layers are trainable

Table 4.7: Evaluated freezing strategies used for the pretrained CNN architectures.

4.5.3 Image Classification

Given that the project focuses on limited data and the identification of an effective ML based classification solution, the primary emphasis is placed on model development and selection. To this end, an automated filtering and tuning framework was designed, evaluating 315 model configurations for each view to ensure the selection of an optimal model.

This procedure corresponds to the initial stages illustrated in Figure 4.2. In the first stage, each configuration is trained for up to 15 epochs, with early stopping applied if the validation accuracy does not improve over three consecutive epochs. During this stage, different base network architectures are compared alongside key hyper-parameters, including image size, learning rate, and freezing strategies for pretrained networks. For the baseline CNN setup, the evaluation instead of freezing strategies focuses on determining the optimal number of convolutional layers; 6, 8 or 10 layers but also the right learning rate and image size as the pretrained networks.

In the second stage, the two best performing models for each view are selected and further trained for an additional 20 epochs, with early stopping applied after seven epochs without improvement to mitigate overfitting. This stage involves more refined hyperparameter tuning, including the choice of loss function, learning rate, dropout rate, scheduler type, and threshold optimization.

In both stages, the top five configurations are reevaluated using three different random seeds. This strategy serves as a simplified alternative to k-fold cross validation and improves robustness by reducing sensitivity to variations in data partitioning.

As a result, the final selection of the two best performing models for each view is less influenced by favorable or unrepresentative data splits.

4.5.4 Fusion-based Package Classification

In Stage 3, the best-performing models from the "above", "side", and "under" camera views were combined using a late fusion architecture to perform the final package-level classification. Prior to fusion, temperature scaling was applied to the output logits of the independently trained view-specific models in order to improve probability calibration between the different viewpoints.

Two fusion configurations were evaluated and compared. The first configuration, referred to as *scalar-only fusion*, used only prediction-level information from each view, including the calibrated logits, probabilities, and confidence estimates. These scalar values were weighted and concatenated before being processed by a lightweight feedforward neural network consisting of fully connected layers, ReLU activation functions, and dropout regularization.

The second configuration, referred to as *scalar-plus-deep fusion*, extended the scalar representation by additionally incorporating deep feature representations extracted from the final feature layers of the view-specific convolutional neural networks. Since the feature dimensions differed between architectures and camera perspectives, the extracted features were projected into a shared latent representation before concatenation with the scalar information. The combined feature representation was then processed using the same feedforward network architecture as the scalar-only configuration.

The purpose of comparing the two fusion configurations was to evaluate whether incorporating deep latent feature representations could improve the package-level classification performance compared with relying solely on prediction-level information. The different strategies are explained in detail in Section 2.3.

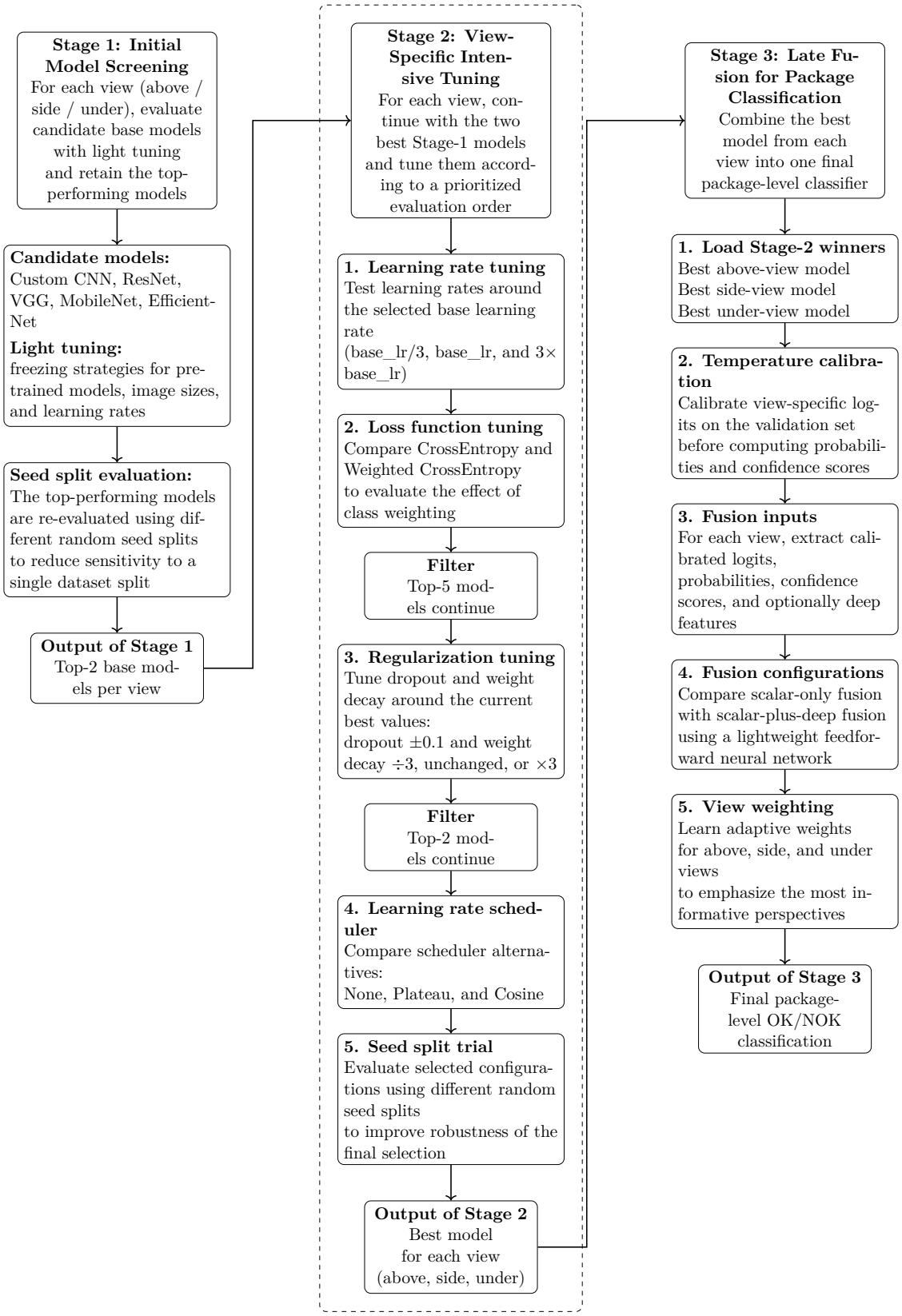


Figure 4.2: Three-stage model selection and fusion framework. Stage 1 performs initial model screening, Stage 2 performs progressive intensive tuning with filtering between tuning stages, and Stage 3 performs calibrated late fusion for final package-level classification

4.6 Training Configuration

In consideration of the real-time inspection system running on the NVIDIA Jetson AGX Orin, AdamW's properties align well with the system constraints and was therefore selected as the optimizer throughout the project. Since the training data is limited and expanded using data augmentation, adaptive optimizers with per-parameter learning rates are beneficial[36]. These methods reduce sensitivity to manual hyperparameter tuning and improve convergence stability during training. AdamW's computational efficiency and relatively low memory requirements are advantageous for training on the device, while its robustness to noisy gradients is beneficial when dealing with the increased variability introduced by data augmentation[19].

The Rectified Linear Unit (ReLU) was used as the activation function throughout the custom network architectures due to its computational efficiency, simplicity, and proven effectiveness in image classification and defect detection applications. ReLU also helps mitigate the vanishing gradient problem, enabling stable training of deeper neural networks [5].

Both standard binary cross-entropy loss and weighted binary cross-entropy loss were evaluated as part of the automated model selection process. The weighted variant was included to address potential class imbalance by assigning greater importance to underrepresented classes. Treating the loss function as a tunable hyperparameter allowed the framework to automatically determine which approach produced the best performance for a given model and dataset.

4.7 Performance Evaluation

The evaluated models are compared across environments to determine the optimal model for the task, using the metrics defined in Section 2.5.

In this application, a False Positive is considerably more costly than a False Negative, as it corresponds to a faulty product reaching the consumer. Precision is therefore a more important evaluation metric than recall, although recall remains important for ensuring that defective products are detected. The F1-score is included to provide a balanced summary that accounts for the class imbalance present in the dataset. The confusion matrix is used throughout to inspect the general behavior of each model and to identify systematic misclassification patterns.

4.7.1 Ranking the Models

Due to the limited size of the dataset, evaluating model performance based on a single metric can be misleading. To obtain a more robust and reliable assessment, a structured evaluation and ranking procedure was adopted.

The dataset was divided into three disjoint subsets: training, validation, and test.

Each model configuration was trained exclusively on the training set. During training, model performance was monitored after each epoch using the validation set, where a range of metrics was computed, including validation loss, accuracy, F1-score, sensitivity for the 'not okay" (NOK) and recall 'okay" (OK) classes, as well as the number of False Positives (FP) and False Negatives (FN).

Rather than selecting the optimal epoch based solely on validation accuracy, a cost-sensitive objective was used to better reflect the practical cost of classification errors. This was achieved by defining a selection cost:

$$\text{selection\_cost} = c_{\text{FP}} \cdot \text{val}_{\text{FP}} + c_{\text{FN}} \cdot \text{val}_{\text{FN}} \tag{4.1}$$

where the relative importance of classification errors was explicitly modeled. In this work, False Positives were penalized more heavily than False Negatives because a defective package incorrectly classified as acceptable may reach the customer, whereas a False Negative only results in an unnecessary rejection of a valid package.

$$c_{\text{FP}} = 3.0, \quad c_{\text{FN}} = 1.0 \tag{4.2}$$

This weighting reflects the higher practical cost associated with incorrectly classifying defective packages as acceptable.

To reduce sensitivity to noise in individual epochs, the final validation performance of a model was computed as the mean performance across the three highest-ranked epochs.

Epochs were ranked according to a hierarchical multi-criteria evaluation procedure. The primary objective was to minimize the selection cost. If two epochs obtained the same selection cost, additional metrics were used as tie-breaking criteria. The ranking priority was:

1. Minimize selection cost
2. Maximize validation macro F1-score
3. Maximize validation specificity for the NOK class
4. Minimize validation loss
5. Maximize validation accuracy
6. Minimize the training accuracy smoothness penalty

The training accuracy smoothness penalty was included to discourage unstable train-

ing behavior by penalizing drops in training accuracy between consecutive epochs.

The ranking procedure can be expressed as the ordered tuple

$$\Big(\text{selection\_cost}, -F1_{\text{macro}}, -\text{Specificity}, \text{Loss}_{\text{val}}, -\text{Accuracy}_{\text{val}}, \text{SmoothnessPenalty}\Big) \tag{4.3}$$

where lower tuple values correspond to better-performing epochs. After ranking all epochs, the three highest-ranked epochs were selected and their validation metrics averaged to produce a more stable estimate of model performance.

The same ranking procedure was subsequently applied when comparing different model configurations. Consequently, model selection was consistently driven by minimizing costly classification errors while still considering overall classification quality and training stability.

Early stopping was also based on this multi-criteria validation objective, ensuring consistency between training dynamics, epoch selection, and final model ranking.

The test set was strictly reserved for the final evaluation and was not used during any stage of model selection, including early stopping, epoch selection, or hyperparameter tuning. It was evaluated only after the best-performing models had been selected using the training and validation data.

Overall, this evaluation procedure provides a more robust framework than approaches based solely on validation accuracy. By explicitly accounting for both False Positives and False Negatives, incorporating F1-score to address class imbalance, and aligning model ranking with real-world error costs, the method yields a more reliable assessment of model performance.

5

Results

This chapter presents the experimental results obtained from the developed automated package classification system, including dataset analysis, hardware evaluations, single-view classification performance, and the proposed multi-stage model selection and fusion pipeline.

5.1 Dataset

In this project multiple datasets were created and used, gather as described in the Section 4.4.1. In total, 84 defect and 24 correct packages were collected, for a total of 106 packages, however, a package could be defect but a certain view of that package could be correct while another view could be defect, see Figure 5.1. Creating different splits between the views.

Figure 5.1: The same package from different views where the picture from above is correct, but the view from below is clearly defect.

All the images in the monochromatic and color Basler dataset were taken when the conveyor belt was moving, while the Webcam dataset was taken when the conveyor belt was standing still, or on a white background.

5.1.1 Monochromatic Dataset

The monochromatic dataset example split, Basler$_{\mathbf{BIG_BW}}$. The data split looked like this,

Camera Position	Dataset	Total Samples	OK	NOK
Above	Train/Validation	525	219	306
Above	Test	108	60	48
Side	Train/Validation	1074	462	612
Side	Test	216	96	120
Under	Train/Validation	537	171	366
Under	Test	113	32	81

Table 5.1: Dataset distribution for the three views (above, side, and under) across training, validation, and test datasets. Taken with the monochromatic camera

Example of these images:

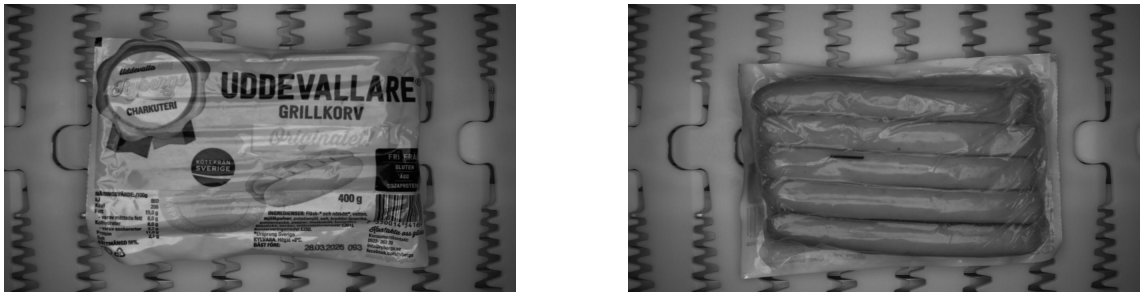


Figure 5.2: Above and under, photographed using a monochromatic camera.

From the side, here two images different images where taken as the packaged passed through the data gathering system both right side up and upside down:

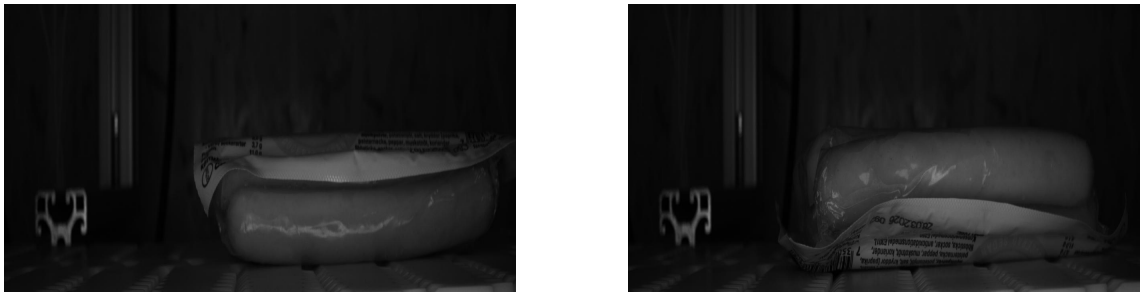


Figure 5.3: Two different views from the side.

The side illumination was to weak for the side view which created a suboptimal picture, see in Discussion 6.6.4 for fixes of this .

5.1.2 Color Dataset

The color dataset, $\text{Basler}_{\text{BIG\_C}}$. The data split looked like this,

Camera Position	Dataset	Total Samples	OK	NOK
Above	Train/Validation	525	219	306
Above	Test	108	60	48
Under	Train/Validation	537	171	366
Under	Test	113	32	81

Table 5.2: Dataset distribution of the three views (above, side, and under) across training, validation, and test datasets. Taken with the monochromatic camera

The amount of images in the split:

Figure 5.4: Above and under, photographed using a color Basler camera.

5.1.3 Webcam

The dataset collected with the Webcam, $\text{Webcam}_{\text{BIG}}$.

Camera Position	Dataset	Total Samples	OK	NOK
Above	Train/Validation	528	192	336
Above	Test	105	48	57
Side	Train/Validation	1062	423	639
Side	Test	214	102	112
Under	Train/Validation	531	139	392
Under	Test	108	30	78

Table 5.3: Dataset distribution of the three views (above, side, and under) across training, validation, and test datasets. Taken with the LG Webcam

5.1.4 Imbalanced Data

Seen in Table 5.3, particularly for the dataset containing views from beneath the package, there is a clear class imbalance, with defect images outnumbering non-defect images by roughly three to one. This imbalance is also clearly reflected in the model training results.

As an example of model behavior without any strategy for handling imbalanced data, i.e., using no pretraining and binary cross-entropy loss, the model tends toward majority-class predictions. For instance, when training a simple five-layer CNN without pretraining:

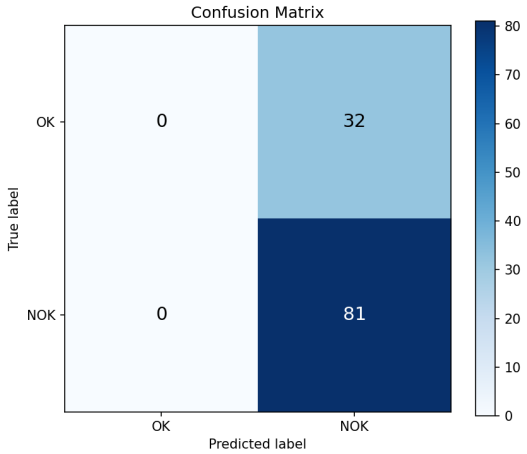


Figure 5.5: Confusion matrix of a simple 5 layer network, trained on the under dataset.

5.2 Ambient Light

Images taken with the industrial cameras and illumination setup show that ambient light has no notable effect. Comparing both packages in Figure 5.6, there is no discernible difference in the lighting of the images, aside from the glare, which is a consequence of the package surface itself and not of ambient light.

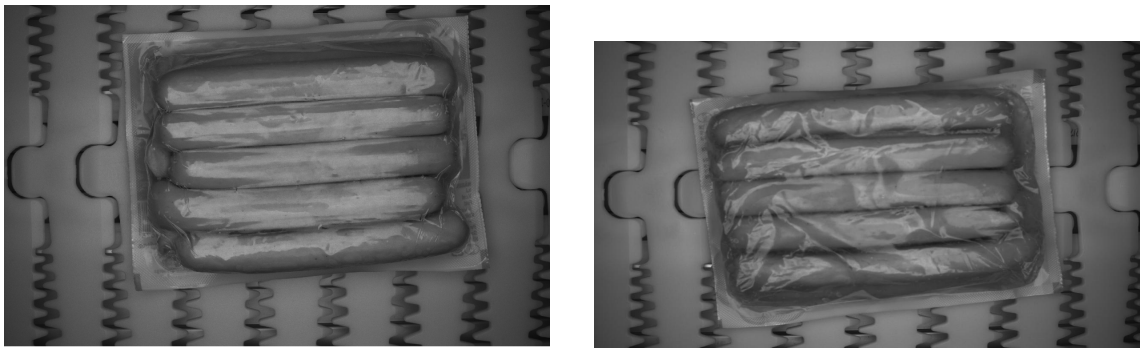


Figure 5.6: Comparison between with and without ambient light. The left image is taken with ambient light present, while the right is taken with the light cover on.

The background can also be largely ignored when using a monochromatic camera depending on the exposure time used. As seen in Figure 5.7, the background is not visible even without the cover in place. It is also worth noting that this image was taken in a normally lit room.

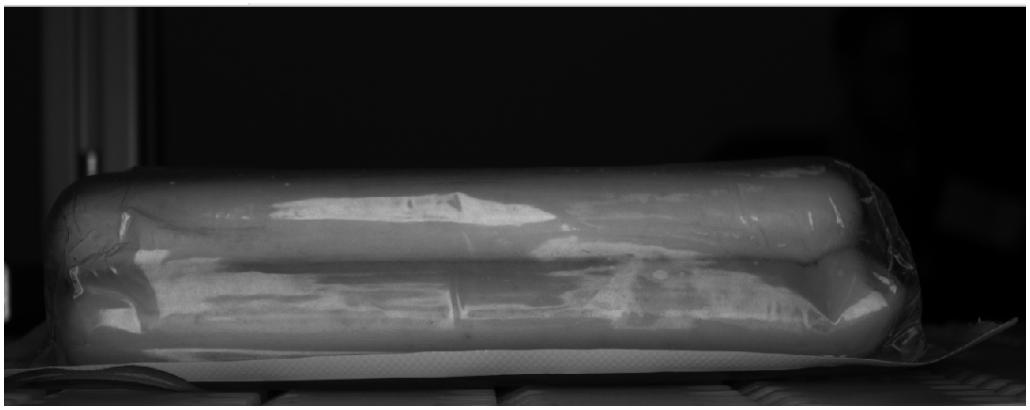


Figure 5.7: Side-view image captured without the ambient light cover.

But if the background is close enough it becomes visible, so there are some benefits with the cover of keeping the background consistent.

Figure 5.8: Side-view image captured with the ambient light cover.

5.3 General Results and Hardware Tests

The following results are created using the same model, seen below, if nothing else is specified.

Parameter	Default / Example Value	Description
Identity		
model_type	`Efficientnet_B0`	-
Model Settings		
dropout	0.5	Dropout probability for regularization
use_pretrained	True	Use pretrained weights
Dataset Settings		
view	`under`	Camera/view angle
grayscale	True	Use grayscale (monochrome) images
Training Settings		
epochs	30	-
batch_size	8	Images processed per batch
optimizer	`Adamw`	Optimization algorithm
lr	1×10^{-4}	Learning rate
weight_decay	1×10^{-4}	Weight regularization factor
Loss Function Settings		
loss_type	`wbce`	Weighted Binary Cross Entropy Loss
Decision Settings		
decision_threshold	0.5	Classification decision threshold

Table 5.4: Model settings used for results. Using ReLu as activation function and padding.

5.3.1 Fold Validation and Split Variation

The effect of split variation in this project is clearly visible in Table 5.6, where accuracy varies considerably across folds, the standard deviation is 5.6% and the gap between the best and worst fold is 16%. Despite all folds sharing the same test dataset, the models perform very differently, suggesting that fold number 4 training split lacked sufficient coverage of certain defect variations present in the test dataset.

	Train			**Validation**			**Test**		
Fold	Total	OK	NOK	Total	OK	NOK	Total	OK	NOK
Fold 1	682	281	401	132	36	96	108	30	78
Fold 2	712	290	422	132	42	90	108	30	78
Fold 3	701	301	400	132	36	96	108	30	78
Fold 4	716	291	425	129	45	84	108	30	78

Table 5.5: Dataset distribution across folds of train, validation, and test splits, using augmented data in train.

Fold	Accuracy	F1 (NOK)	F1 (OK)
Fold 1	0.991	0.994	0.984
Fold 2	0.907	0.935	0.839
Fold 3	0.907	0.939	0.808
Fold 4	0.833	0.875	0.750
Mean ± Std	$\mathbf{0.910 \pm 0.056}$	$\mathbf{0.936 \pm 0.042}$	$\mathbf{0.845 \pm 0.087}$

Table 5.6: Cross-validation results per fold over the same test dataset (mean ± standard deviation).

A related consequence of this split sensitivity is the difficulty of selecting the optimal training epoch, which is typically determined using validation performance. However, as shown in Figure 5.10, the validation set does not reliably represent the test dataset in this case fold 4 achieved higher validation accuracy (∼0.92) yet produced worse test performance than fold 2, which had a lower validation accuracy (∼0.81).

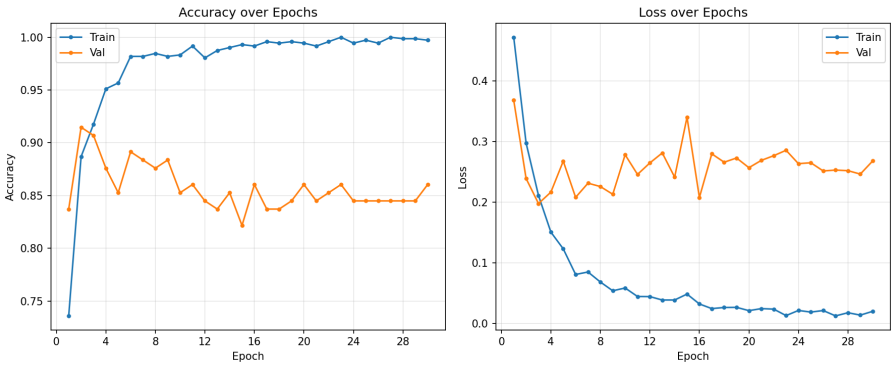


Figure 5.9: Training curves for fold 4. Highest validation accuracy approximately 0.92.

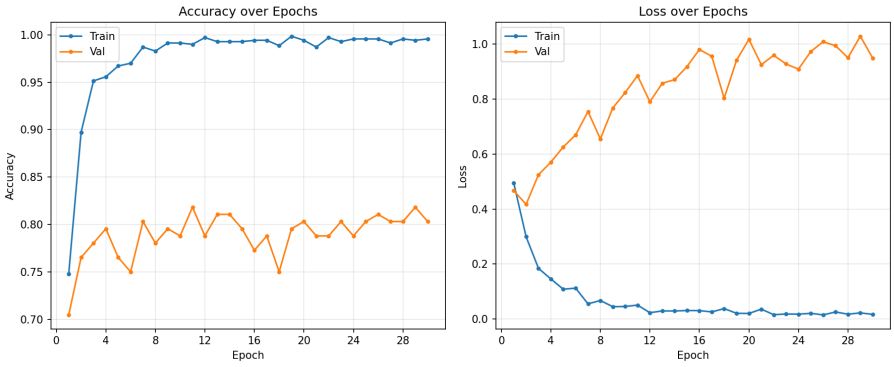


Figure 5.10: Training curves for Fold 2. Highest validation accuracy approximately 0.81.

5.3.2 Augmentation

The effect of augmentation, presented in Section 4.4.4, was evaluated by comparing all four strategies against a no-augmentation baseline across three identical cross-validation folds. As shown in Figure 5.11, the combined augmentation strategy achieved both the highest mean accuracy and the lowest inter-fold variance, while the no-augmentation baseline performed worst on both measures.

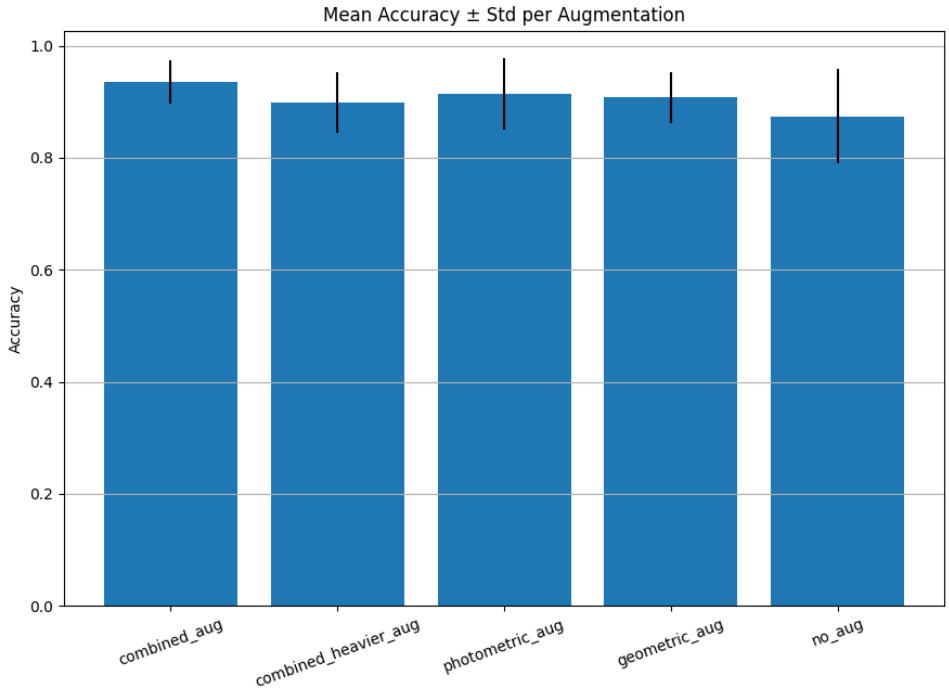


Figure 5.11: Mean accuracy with standard deviation per augmentation strategy across three folds. The combined augmentation strategy achieves the highest mean accuracy and the lowest variance, whereas the no-augmentation baseline yields the lowest mean accuracy and the greatest instability.

Notably, the no-augmentation model achieved the highest accuracy on a single fold (Figure 5.12), yet this result is misleading: the same model exhibited substantially greater variance across folds, indicating sensitivity to the composition of the training split rather than robust generalization. The combined augmentation strategy, by contrast, produced consistently high accuracy across all folds, suggesting that augmentation improves model stability in addition to average performance.

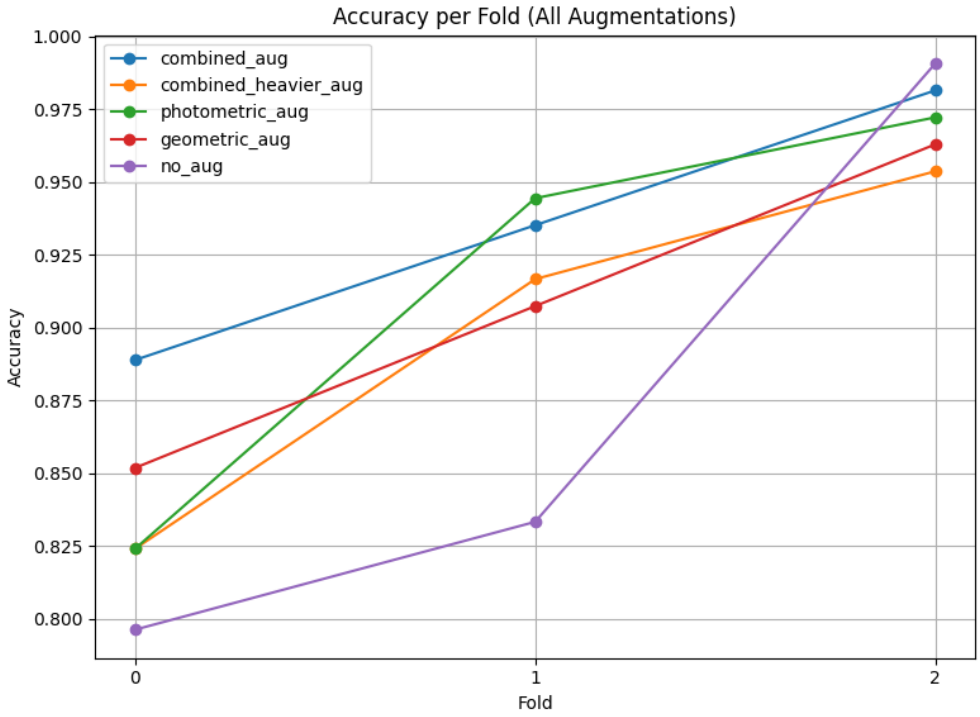
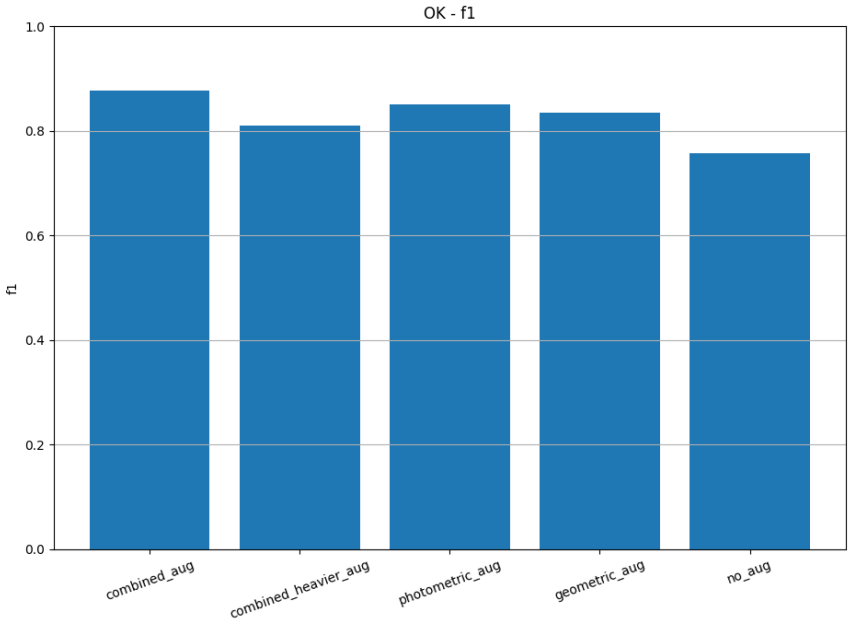


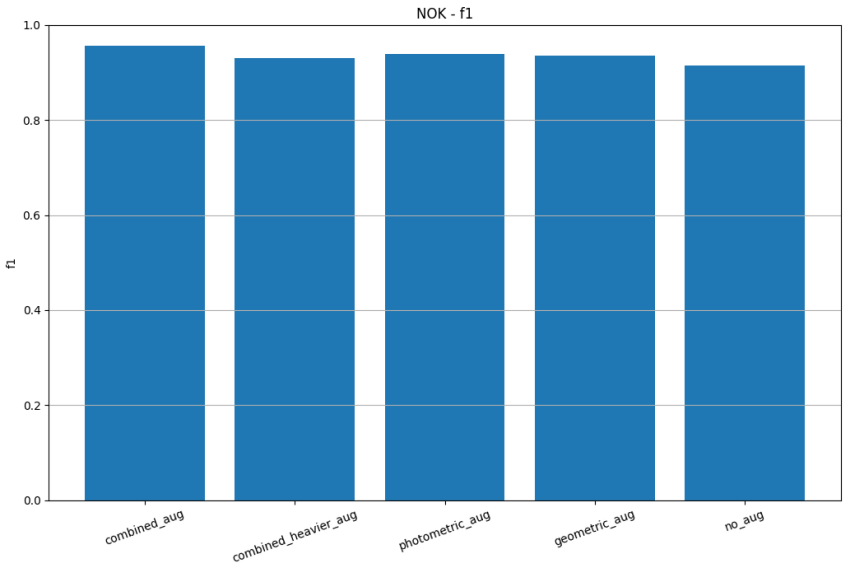
Figure 5.12: Per-fold accuracy for all augmentation strategies. While the no-augmentation baseline achieves the highest accuracy on fold 2, it shows the steepest improvement trajectory, suggesting high sensitivity to training split composition. Augmented strategies exhibit more stable performance across folds.

These findings were consistent across different model architectures, and across multiple random seeds, supporting the robustness of the observed augmentation benefit. Although to varying degrees of improvement, but it has consitently been improvments using augmentations.

Examining per-class F1-scores reveals an asymmetry in how augmentation affects each class on average. As shown in Figure 5.13, the NOK F1-scores are consistently high across all strategies, ranging from approximately 0.91 for the no-augmentation baseline to 0.95 for the combined augmentation strategy, while the OK class shows a more pronounced mean improvement of approximately 11 percentage points between the no-augmentation baseline and the combined strategy.



F1-score for the OK class.



F1-score for the NOK class.

Figure 5.13: Per-class F1-scores across augmentation strategies. On average, augmentation yields a more pronounced improvement on the OK class than on the NOK class, though this pattern varies across folds.

However, this pattern was not consistent across all cross-validation folds or data splits. With some splits, augmentation yielded a greater relative improvement on NOK F1 than on OK F1, suggesting that the observed asymmetry is partly an

artifact of the specific fold composition rather than a stable property of the augmentation strategies. This instability is likely attributable to the small dataset size, where the class distribution within any given fold can vary considerably, causing per-class metrics to fluctuate between runs.

These findings highlight the difficulty of drawing strong conclusions from per-class F1-scores when the dataset is small and class-imbalanced. The mean results favor augmentation for both classes, but the magnitude and direction of the per-class benefit should be interpreted with caution.

5.3.3 Pretrained Networks

Using pretrained networks for the weights and only training parts of the network, as shown in Section 4.5.2, gives a massive performance boost compared to using fresh weights. Looking over the result it is a boost of around 15-20% for the accuracy over three different seeds.

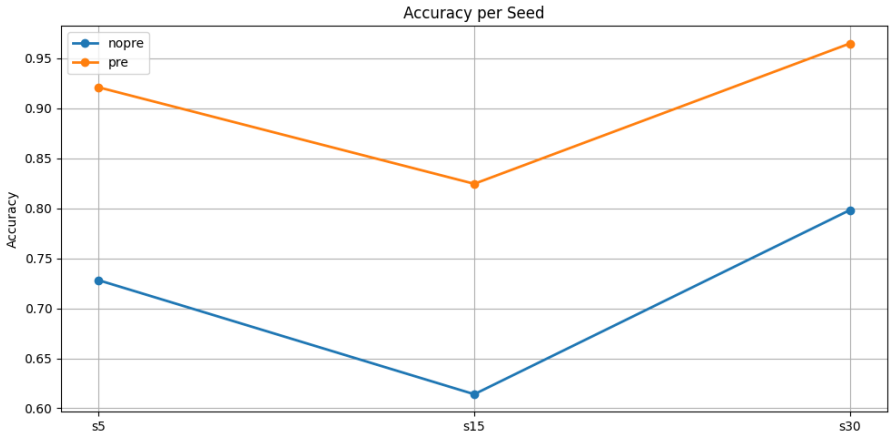


Figure 5.14: Comparison between pretrained models and fresh models.

5.3.4 Characteristics of Different Camera Views

All the previous results have been on a single view of the package, namely the view from underneath of the package. This is because those images contain more information about the package and it is in general easier to determine if a package is correct or not. When comparing the effectiveness of the models it is also clear that this is true, as the under view achieved accuracy as high as 98%, while the above and side views performed considerably worse, as seen below. The side view achieves around 76% accuracy while the above view hovers around 60%, with a considerably higher variation between folds (see Section 5.3.1).

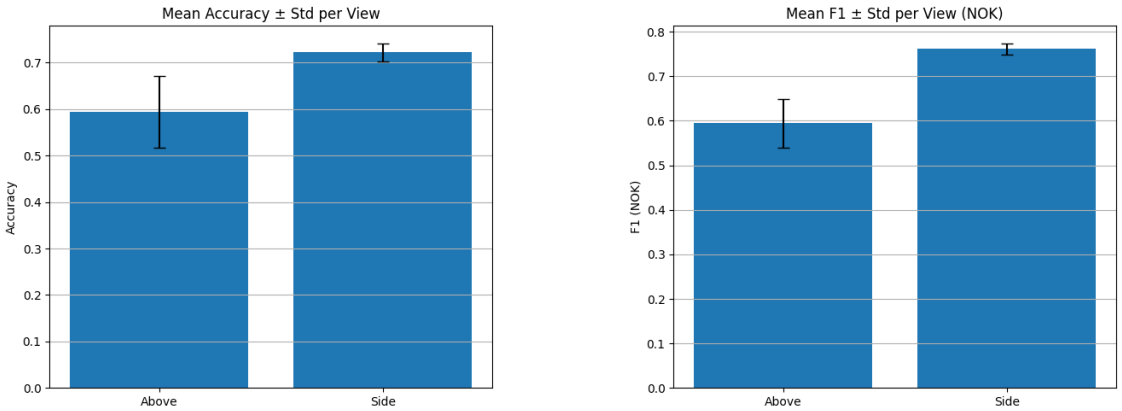


Figure 5.15: Above and side accuracy and F1-score, photographed using a monochrome Basler camera.

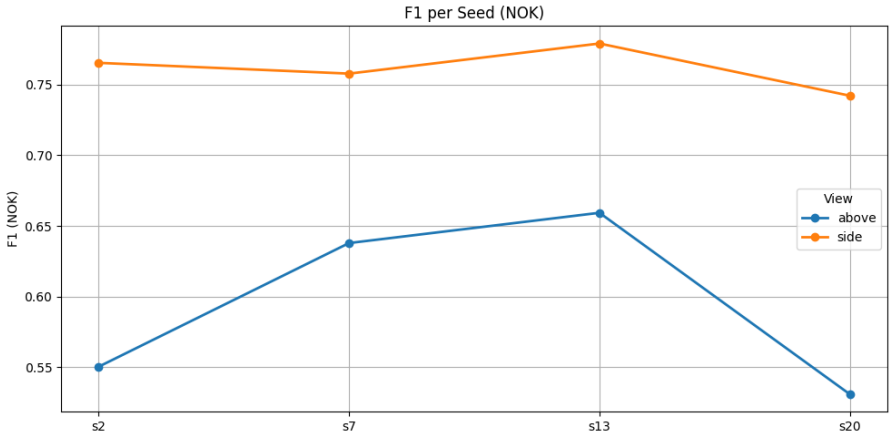


Figure 5.16: F1-score for different models over different splits.

It is clear that some views are more difficult to classify then other.

5.3.5 Color and Monochrome Cameras

Comparing monochrome and color cameras, overall performance is similar, with the color camera achieving slightly higher accuracy and F1-score across five random seeds. Seeds s10 and s12 were trained without data augmentation, while the remaining seeds used the best-performing augmentation strategy identified in Section 5.3.2.

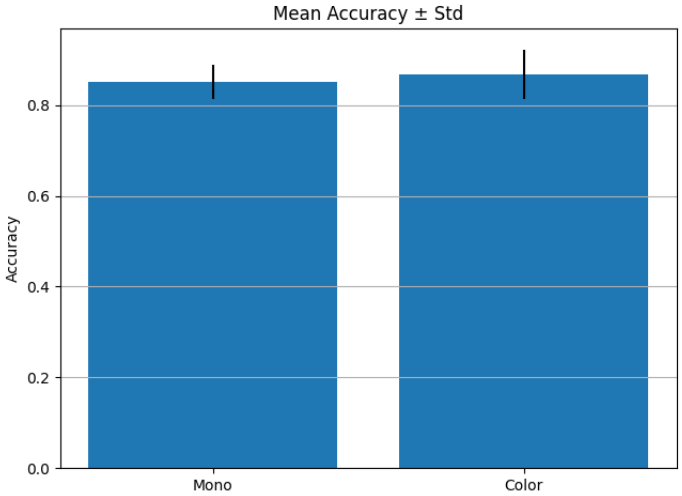


Figure 5.17: Mean accuracy: color camera slightly outperforms monochrome, but also exhibits higher standard deviation.

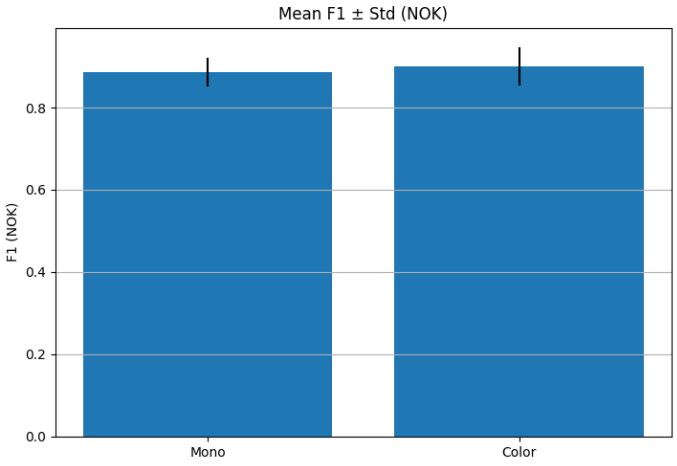


Figure 5.18: Mean F1-score shows similar trends to accuracy, with color marginally ahead.

Looking at individual seeds, Figure 5.19,it is clear that performance varies considerably across splits for both camera types, consistent with the split sensitivity observed in earlier experiments. Notably, the two models respond differently to the same splits despite being trained on similar data partitions.

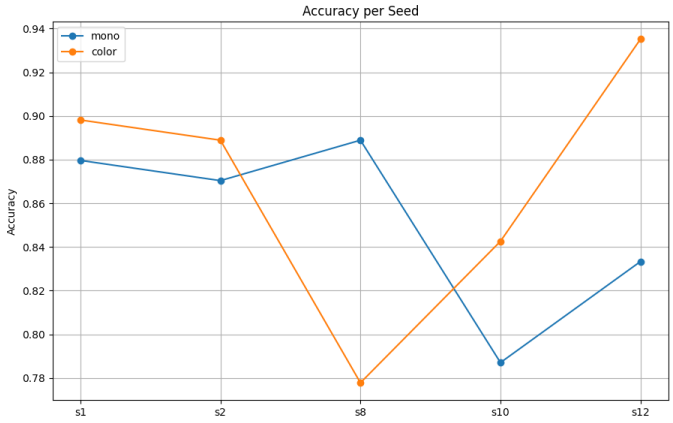


Figure 5.19: Per-seed accuracy for monochrome and color cameras across all five seeds. The models peak at s12 for color and s8 for monochrome, with the largest divergence at s10 and s12.

5.3.5.1 Webcam

Using a Webcam, it is still possible to achieve competitive results compared to the Basler cameras. This is not entirely surprising, as images are resized to 512 × 512 pixels before training, meaning factors such as resolution have little to no impact. As shown in Figure 5.20, the Webcam even outperforms the Basler cameras on the above view. However, this is less attributable to the camera itself and more to the labeling process: since the Webcam images were collected at the beginning of the dataset collection, there was less labeling error due to fatigue. Nevertheless, the results demonstrate that the above view, when provided with accurately labeled data, can achieve stronger performance then seen in Section5.3.4.

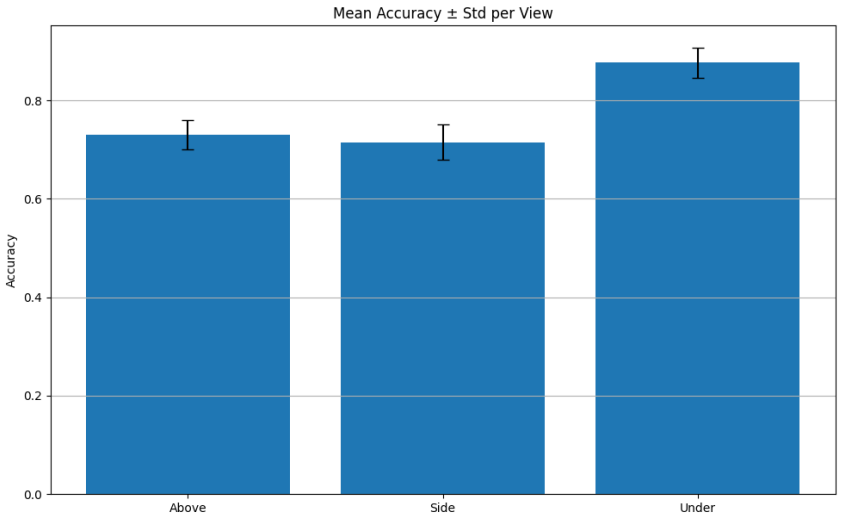


Figure 5.20: Accuracy across different views for the Webcam dataset.

These results were collected across three different splits, as shown in Figure 5.21.

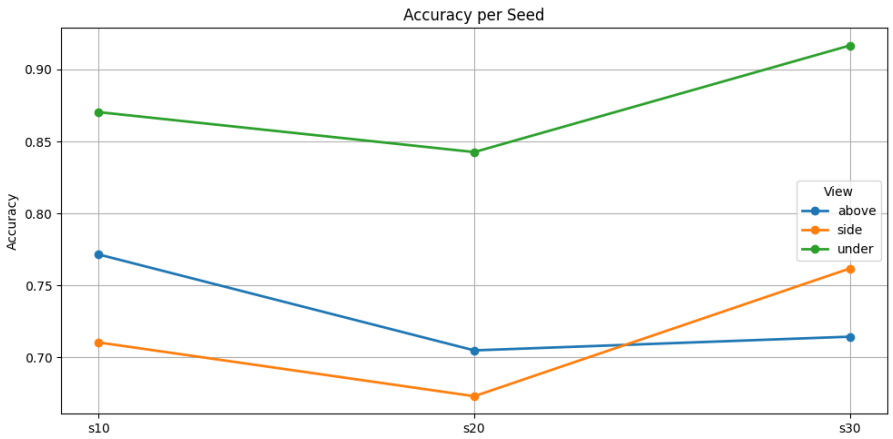


Figure 5.21: Accuracy for different views across different splits for the Webcam dataset.

As mentioned in Section 5.1, all images captured with the Webcam were taken while the product was stationary. This was necessary because the camera suffers from motion blur at its default exposure time, as illustrated in Figure 5.22.

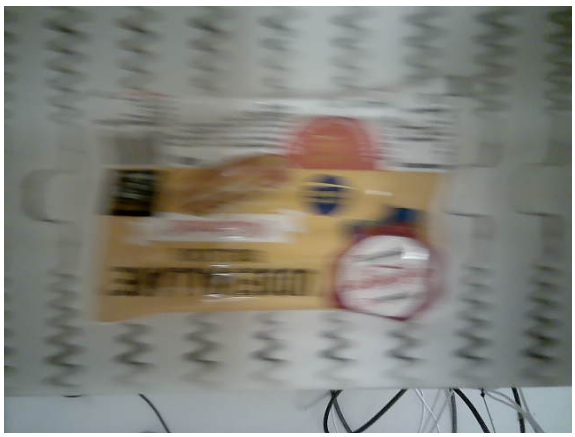


Figure 5.22: Example of motion blur from the LG Webcam.

5.4 Automated Package Classification System

The automated package classification pipeline consists of three sequential stages designed to identify, refine, and combine high-performing view-specific models for sausage package defect classification.

Stage 1 performs a broad automated search across multiple neural network architectures and hyperparameter configurations. Stage 2 continues training and fine-tuning the highest-ranked models from Stage 1 using alternative dataset splits to evaluate model robustness and generalization. Finally, Stage 3 combines the selected view-specific models into a late-fusion package-level classification system.

The following sections present the experimental results obtained throughout the different stages of the automated model-selection pipeline. Section 4.5.3 and Section 4.5.4 gives a more detailed explanation of the structure.

5.4.1 Stage 1

Stage 1 evaluates a large number of network architectures and hyperparameter configurations in order to identify robust view-specific models for the package classification task. A total of 15 different base architectures were evaluated, where each architecture was trained using up to 22 different combinations of the hyper-parameters. The evaluation was performed across four datasets:

$$\text{Basler}_{\mathbf{BIG}},\ \text{Basler}_{\mathbf{SMALL}},\ \text{Webcam}_{\mathbf{BIG}},\ \text{Webcam}_{\mathbf{SMALL}}$$

The results presented in this section include:

- evaluation of the proposed ranking strategy,
- learning-curve analysis,
- and final test-set performance of the highest-ranked models.

5.4.1.1 Evaluation of the Ranking Strategy

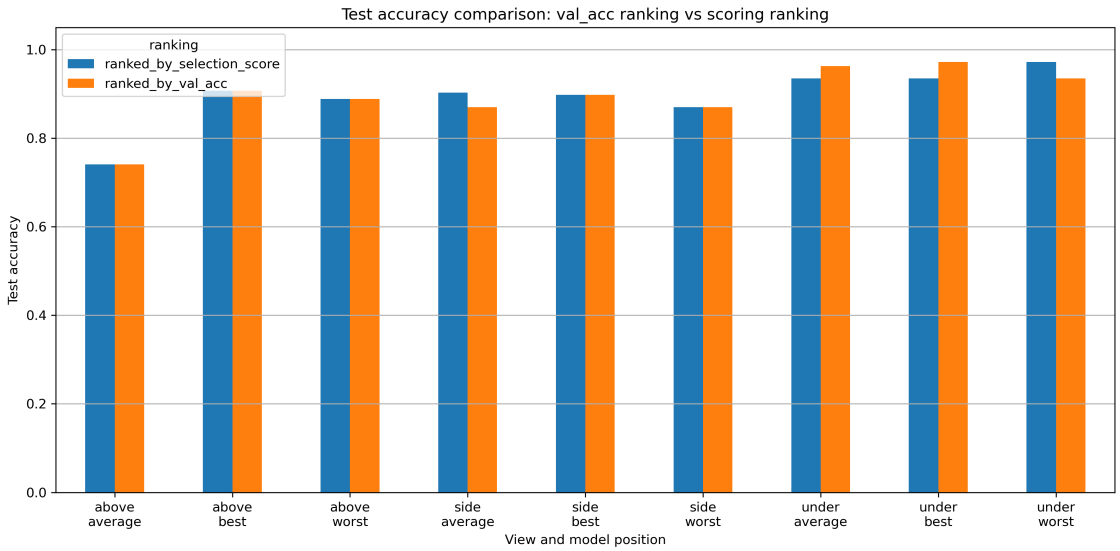
The proposed ranking strategy was compared against a standard validation accuracy based ranking approach in order to evaluate whether the scoring method improved model selection robustness.

The comparison between the two ranking approaches is presented in Figure 5.23. The evaluation was performed by comparing the ranking obtained during validation with the final performance on the independent test dataset.

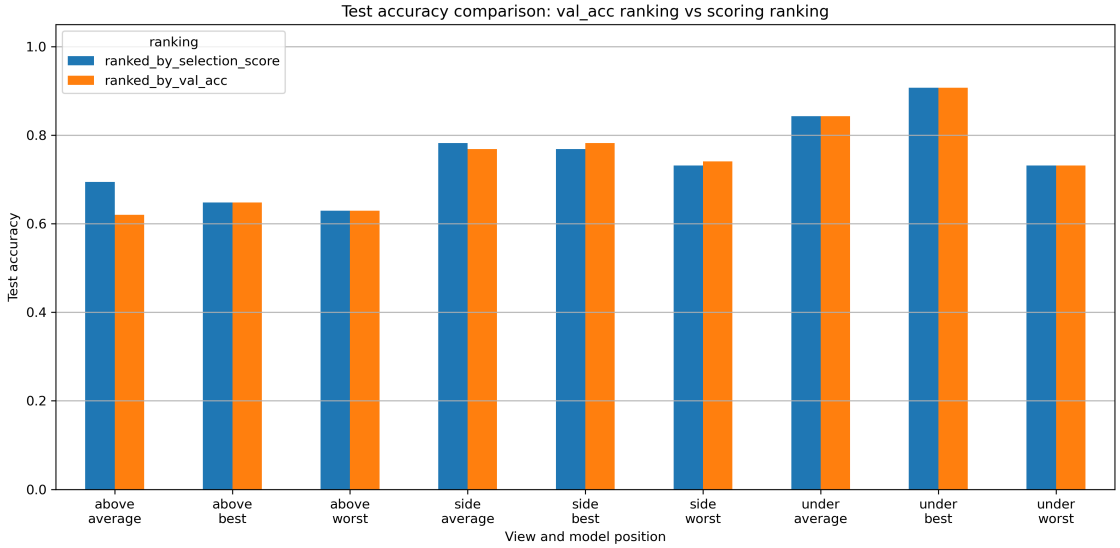
For the Basler$_{\mathbf{BIG}}$ dataset shown in Figure 5.23a, the validation accuracy based ranking achieved better agreement with the final test performance for the under view. In this case, the proposed scoring strategy incorrectly ranked a lower performing model higher than the model that achieved the best test accuracy.

In contrast, the results for the Basler$_{\mathbf{SMALL}}$ dataset in Figure 5.23b demonstrate that the proposed ranking strategy achieved better alignment with the final test dataset performance. The under view-model selected using the scoring approach showed a clear performance advantage compared to the remaining candidates.

These results indicate that the effectiveness of the ranking strategy depends strongly on dataset size and validation stability. For smaller datasets, where validation accuracy may fluctuate significantly between epochs, the proposed scoring method can provide more stable model selection compared to relying solely on peak validation accuracy.



(a) Basler$_{\mathbf{BIG}}$ dataset



(b) Basler$_{\mathbf{SMALL}}$ dataset

Figure 5.23: Comparison of scoring approach towards using standard validation accuracy across the four datasets in Stage 1.

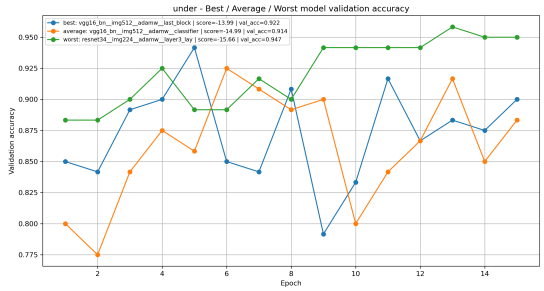
5.4.1.2 Learning Curves

An important challenge when training deep learning models on small datasets is the instability of the validation metrics during training. To analyze the training dynamics, the learning curves for the best, average, and worst performing models according to the proposed scoring strategy are presented in Figure 5.25.

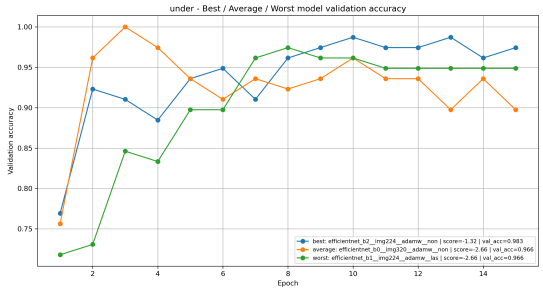
The results for the Basler$_{\mathbf{BIG}}$ dataset shown in Figure 5.24a, demonstrate relatively unstable validation behavior throughout training. However, despite the fluctuations,

the difference between the best and worst performing models remained below approximately 18 percentage points during the 15 training epochs. In comparison, the Basler$_{\mathbf{SMALL}}$ dataset in Figure 5.24b, exhibited more stable convergence behavior. The separation between the best and worst performing models was larger, exceeding 25 percentage points during training, which indicates a clearer distinction in model performance.

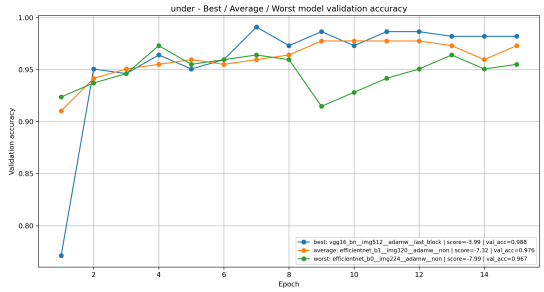
The two Webcam datasets shown in Figure 5.24c and Figure 5.24d, reached high validation accuracy significantly faster than the Basler datasets. Most models achieved near peak validation accuracy within the first four training epochs, suggesting that the Webcam$_{\mathbf{SMALL}}$ datasets may contain less visual complexity or lower internal class variation compared to the Basler datasets.



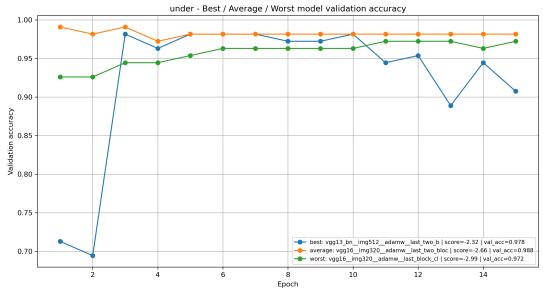
(a) Basler$_{\mathbf{BIG}}$ dataset under view



(b) Basler$_{\mathbf{SMALL}}$ dataset under view



(c) Webcam$_{\mathbf{BIG}}$ dataset under view



(d) Webcam$_{\mathbf{SMALL}}$ dataset under view

Figure 5.24: Comparison of the validation accuracy over 15 epochs for the four datasets in Stage 1 for the view from underneath

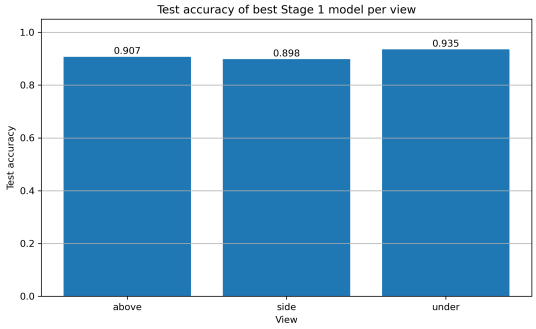
5.4.1.3 Best Performing Models from Stage 1

The final test dataset performance of the highest ranked models from Stage 1 is presented in Figure 5.25.

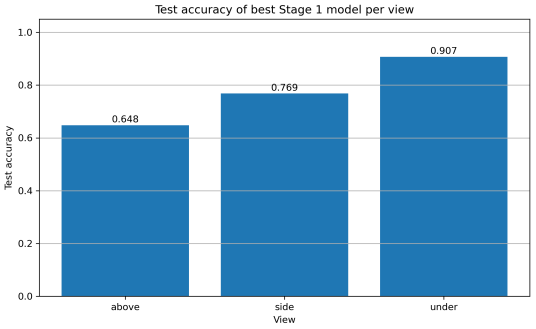
Among all evaluated datasets, the Basler$_{\mathbf{BIG}}$ dataset shown in Figure 5.25a achieved the strongest overall performance, where all three views obtained test accuracies close to or above 90%. This indicates that the larger dataset provided more robust feature representations and improved generalization across all views.

For the smaller datasets shown in Figure 5.25b and Figure 5.25d, the under view achieved performance comparable to, or in some cases better than, the corresponding larger datasets. Similar behavior was observed for the side view.

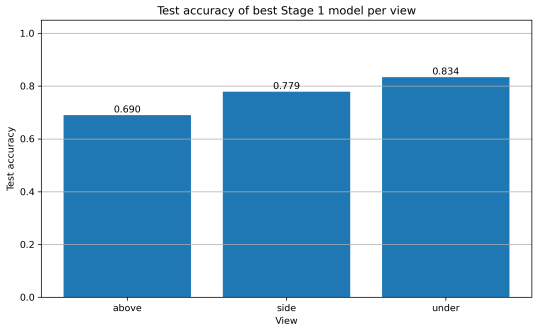
In contrast, the above view demonstrated a substantially larger performance reduction for the smaller datasets. This suggests that the above view classification task is more sensitive to reduced dataset size and may require a larger amount of training data in order to generalize reliably.



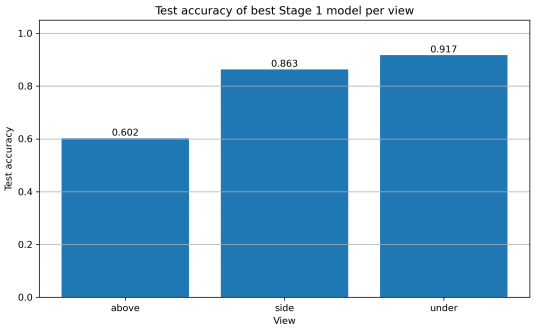
(a) Basler$_{\mathbf{BIG}}$ dataset



(b) Basler$_{\mathbf{SMALL}}$ dataset



(c) Webcam$_{\mathbf{BIG}}$ dataset



(d) Webcam$_{\mathbf{SMALL}}$ dataset

Figure 5.25: Comparison of test accuracy for the best-performing models across the four datasets in Stage 1.

5.4.1.4 Stage 1 Summary

The Stage 1 results demonstrated that automated model selection can identify robust view specific models across multiple datasets and network architectures. The proposed ranking strategy showed improved robustness for smaller datasets where validation accuracy fluctuated significantly between epochs, while standard validation accuracy ranking performed better for certain larger dataset configurations.

The learning curve analysis highlighted clear differences in convergence behavior between the datasets. The smaller dataset generally produced the most stable and consistent performance, while Basler$_{\mathbf{BIG}}$ datasets exhibited larger validation fluctuations and increased sensitivity to stochastic training behavior. Even though the

$\text{Basler}_{\textbf{BIG}}$ performed the best on the test dataset.

Among the evaluated views, the under view consistently achieved the most stable and robust performance across all datasets. In contrast, the above view showed greater sensitivity to reduced dataset size and demonstrated lower generalization capability for the smaller datasets.

Overall, the Stage 1 results indicate that both dataset size and view selection strongly influence the stability and generalization capability of the trained models.

5.4.2 Stage 2

Stage 2 continues training the two highest ranked models from Stage 1 for an additional 25 epochs using the Stage 1 checkpoints as initialization.

Stage 2 uses different random seed splits than those used in Stage 1 in order to evaluate the robustness and generalization capability of the selected models. This approach reduces the risk of selecting models that perform well only for a specific train, validation, and test split.

The datasets included in Stage 2 were:

$$\text{Basler}_{\textbf{BIG}},\ \text{Basler}_{\textbf{SMALL}},\ \text{Webcam}_{\textbf{SMALL}}$$

The $\text{Webcam}_{\textbf{BIG}}$ dataset was excluded from Stage 2 since the Webcam system was considered unsuitable for industrial motion based acquisition due to motion blur effects, as illustrated in Figure 5.22.

The $\text{Webcam}_{\textbf{SMALL}}$ dataset was nevertheless included since it required relatively low computational resources and gives an alternative for gathering the dataset at the costumer.

5.4.2.1 Learning Curves Based on the Validation Dataset

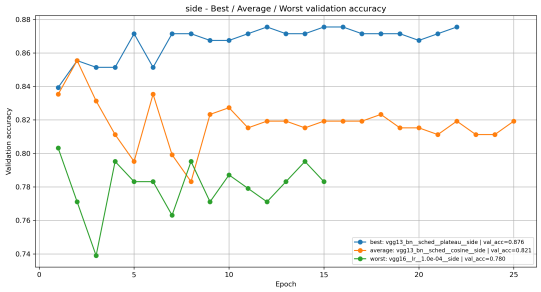
The Stage 2 learning curves for the side view-models are presented in Figure 5.26. The left column shows the validation accuracy while the right column presents the training loss.

For the $\text{Basler}_{\textbf{BIG}}$ dataset shown in Figure 5.26a and Figure 5.26b, both the best and worst performing models appear to have reached convergence relatively early during training. This behavior is visible both in the stabilization of the validation accuracy and in the plateauing of the training loss. Early stopping was triggered after 15 respectively 22 epochs without improvement in the scoring metric for the best and worst model.

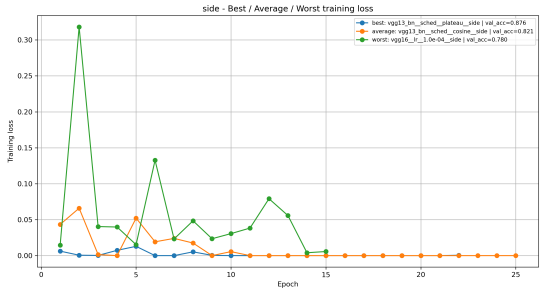
The $\text{Basler}_{\textbf{SMALL}}$ dataset shown in Figure 5.26c and Figure 5.26d, exhibited more

fluctuating learning behavior. Temporary increases in training loss were observed during training, particularly around epoch 15. However, despite the fluctuations, no clear divergence between training and validation performance was observed, suggesting that severe overfitting did not occur.

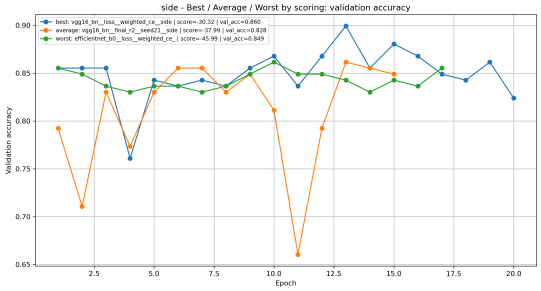
A similar trend was observed for the Webcam$_{\mathbf{SMALL}}$ dataset. Even the best performing models exhibited validation accuracy fluctuations around $90 \pm 5\%$, despite the scoring function favoring stable training behavior. This highlights the sensitivity of small datasets to stochastic training variations and dataset split differences.



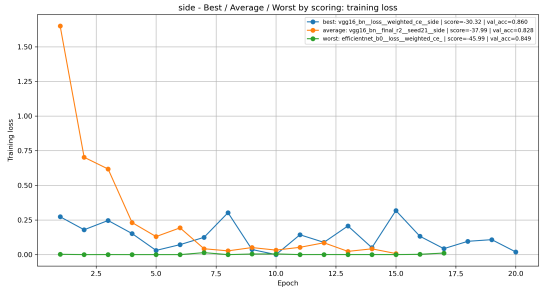
(a) Basler$_{\mathbf{BIG}}$ dataset validation accuracy side view



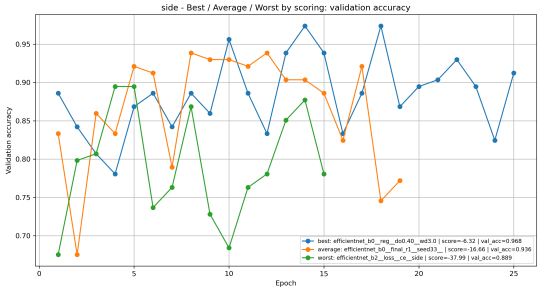
(b) Basler$_{\mathbf{BIG}}$ dataset training loss side view



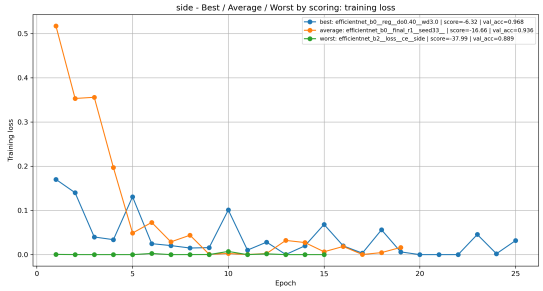
(c) Basler$_{\mathbf{SMALL}}$ dataset validation accuracy side view



(d) Basler$_{\mathbf{SMALL}}$ dataset training loss side view



(e) Webcam$_{\mathbf{SMALL}}$ dataset validation accuracy side view



(f) Webcam$_{\mathbf{SMALL}}$ training loss side view

Figure 5.26: Best, mean and worst model performance comparison for the three datasets during Stage 2 training from the side view. Left column shows validation accuracy and right column shows training loss.

5.4.2.2 Best Performing Models from Stage 2

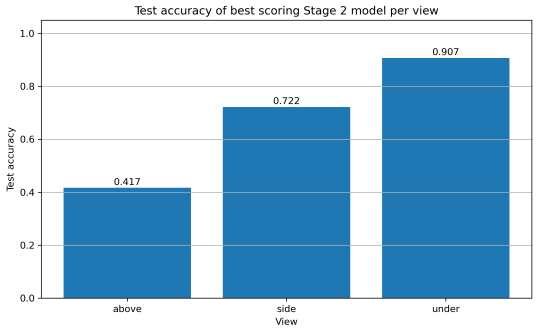
The final Stage 2 test dataset performance is presented in Figure 5.27. Compared to Stage 1, most Stage 2 models achieved lower test accuracy across the evaluated datasets. However, since Stage 2 used alternative dataset splits, the results provide important insight into the robustness and split sensitivity of the selected models rather than only absolute performance.

For the Basler$_{\mathbf{BIG}}$ dataset shown in Figure 5.27a, all three views experienced reduced test accuracy compared to Stage 1. The largest reduction occurred for the above view, where the accuracy decreased from 90.7% in Stage 1 to 41.7% in Stage 2. This corresponds to a reduction of approximately 49 percentage points, indicating strong sensitivity to dataset split variations for this view.

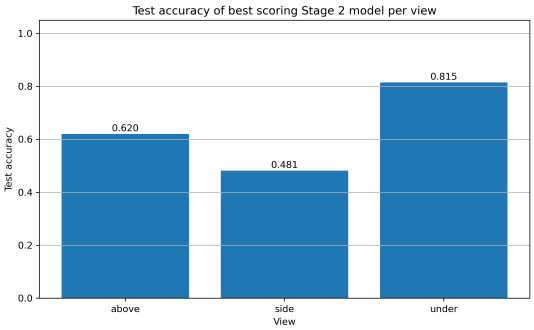
The under view remained the most stable view across both stages, maintaining a test accuracy above 90% despite the reduction compared to Stage 1. This suggests that the under view features are more robust and easier to generalize across different dataset splits.

A similar trend was observed for the Basler$_{\mathbf{SMALL}}$ dataset shown in Figure 5.27b. The largest reduction occurred for the side view, where the test accuracy decreased from 76.9% to 48.1%. In contrast, the above view remained relatively stable, with only a minor reduction of approximately 2.8 percentage points.

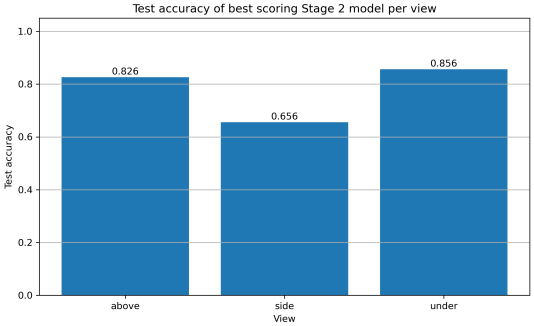
The Webcam$_{\mathbf{SMALL}}$ dataset shown in Figure 5.27c, was the only dataset where one of the views improved during Stage 2. The above view increased by approximately 22.4 percentage points compared to Stage 1. Although the remaining views did not improve, they maintained relatively high performance compared to the other evaluated datasets.



(a) Basler$_{\mathbf{BIG}}$ dataset test



(b) Basler$_{\mathbf{SMALL}}$ dataset test



(c) Webcam$_{\mathbf{SMALL}}$ dataset test

Figure 5.27: Comparison of test accuracy for the best-performing models across the four datasets in stage 2.

5.4.2.3 Stage 2 Summary

The Stage 2 results demonstrated that the selected models were highly sensitive to dataset split variations, especially for the smaller datasets. The Basler$_{\mathbf{BIG}}$ dataset generally showed the most stable learning behavior and convergence characteristics, while the Basler$_{\mathbf{SMALL}}$ and Webcam$_{\mathbf{SMALL}}$ datasets exhibited larger fluctuations in both validation accuracy and training loss.

The test dataset evaluation further highlighted that several models achieved substantially lower performance compared to Stage 1 when trained on alternative random splits. This indicates that some of the Stage 1 results were strongly dependent on the selected dataset split rather than purely on the model architecture itself.

Among the evaluated views, the under view consistently showed the most robust performance across different dataset splits, while the above and side views demonstrated larger sensitivity to variations in the training data. The Webcam$_{\mathbf{SMALL}}$ dataset was the only dataset where one of the views improved during Stage 2, suggesting that additional fine tuning and alternative dataset splits may improve generalization for certain configurations.

5.4.3 Stage 3

Stage 3 presents the results of the late-fusion classification framework, where the best-performing view-specific models from Stage 2 are combined to produce package-level predictions. The section evaluates both the scalar only and scalar plus deep feature fusion approaches across the different datasets using training history, confidence analysis, confusion matrices, and threshold behavior

5.4.3.1 Training History

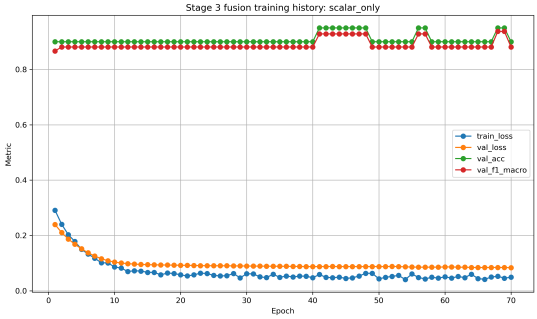
Training history for the fusion model includes the three different datasets but with the two different approaches, scalar only and scalar plus deep feature training which is explained in further detail in Section 4.5.4.

For the Basler$_{\mathbf{BIG}}$ dataset Figure 5.28a and Figure 5.28b, the different models are very similar where both reaches 88% in validation accuracy and both training losses converges towards 0.05. The differences between the plots are that the scalar plus deep method skips between 85% and 87% more frequently than the scalar only method. The scalar plus deep feature method reaches the stable training loss already after 9 epochs while the scalar only reaches the 0.05 training loss first after 15 epochs of training.

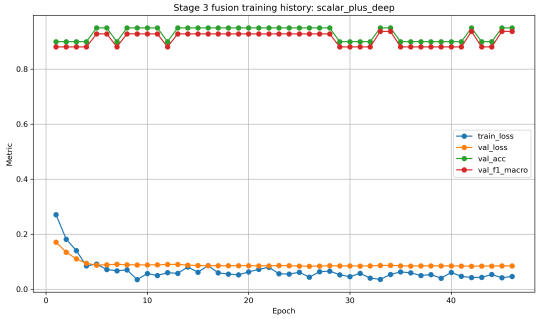
For the Basler$_{\mathbf{SMALL}}$ dataset Figure 5.28c and Figure 5.28d, where both strategies reaches 100% in both training and validation accuracy very fast. Regarding the loss, it takes longer time for the scalar only to reach 0, after 50 epochs while the scalar plus deep reaches 0 after 20 epochs.

Regarding the Webcam$_{\mathbf{SMALL}}$ dataset Figure 5.28e and Figure 5.28f, there are some more concrete differences between the different methods.

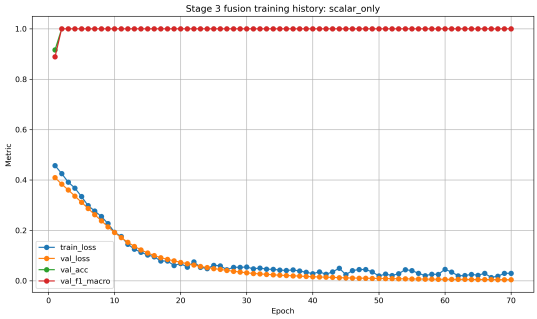
The scalar only method differs a lot more between 80 percent and 100 percent than what the scalar plus deep method does. The scalar plus deep method reaches an early stoppage after 60 epochs while the val accuracy has reached 100% and the loss are down to 0.



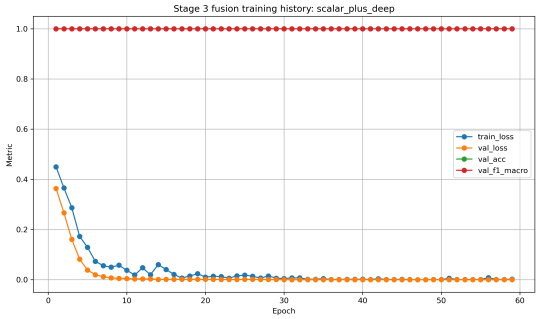
(a) Basler$_{\mathbf{BIG}}$ scalar only training history



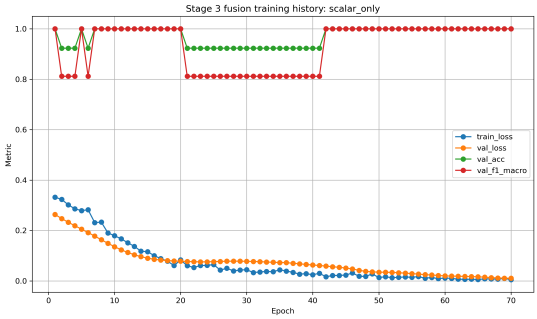
(b) Basler$_{\mathbf{BIG}}$ scalar plus deep training history



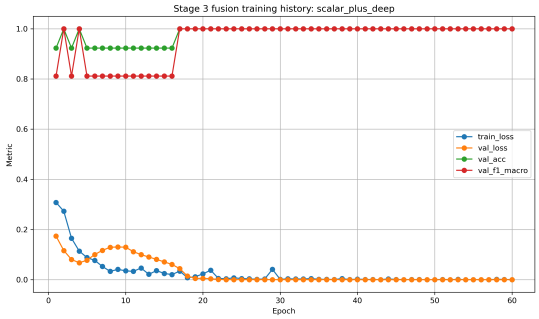
(c) Basler$_{\mathbf{SMALL}}$ scalar only training history



(d) Basler$_{\mathbf{SMALL}}$ scalar plus deep training history



(e) Webcam$_{\mathbf{SMALL}}$ scalar only training history



(f) Webcam$_{\mathbf{SMALL}}$ scalar plus deep training history

Figure 5.28: Training history for the fusion models across the difference datasets, which views validation accuracy and loss and also training accuracy and loss

5.4.3.2 Fusion Confidence and Confusion Matrix Analysis on the Test Dataset

This section analyzes the confidence behavior of the Stage 3 fusion models across the evaluated datasets. The results compare the scalar only fusion model with the scalar plus deep feature fusion model.

The presented figures illustrate how the fusion model combines the predictions from the three view specific models and how the final confidence distributions influence the classification performance. The confusion matrices provide further insight into whether the fusion models are more sensitive to false positive or false negative predictions.

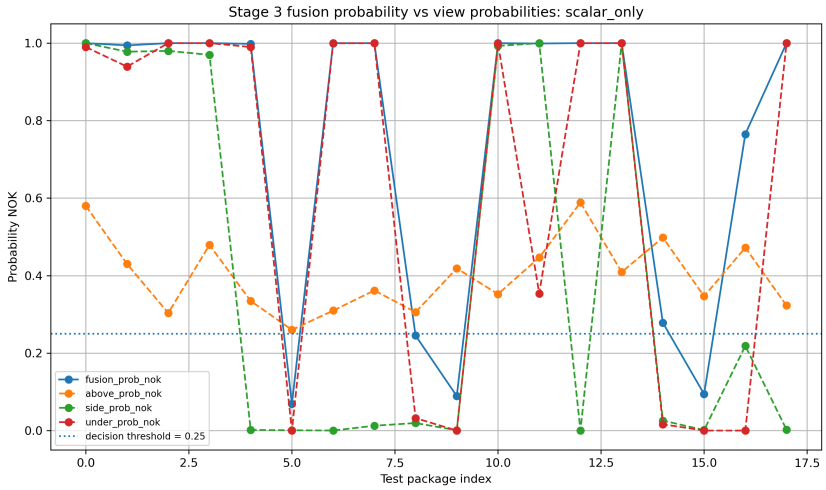
The selected decision thresholds are also analyzed in order to evaluate how the confidence calibration of the fusion models influences the final classification boundary. For the Basler$_{\mathbf{BIG}}$ dataset shown in Figure 5.29a and Figure 5.29b, the fusion model relied primarily on the predictions from the under view-model. This behavior is visible in both fusion models, where the final fusion probabilities closely follow the probability distribution from the under view throughout most test samples.

The side view-model produced consistently high confidence predictions and contributed strongly to the final fusion output. In cases where the side view predictions were correct, this improved the stability of the fusion model. However, incorrect high confidence predictions from the side view also influenced the fusion prediction negatively.

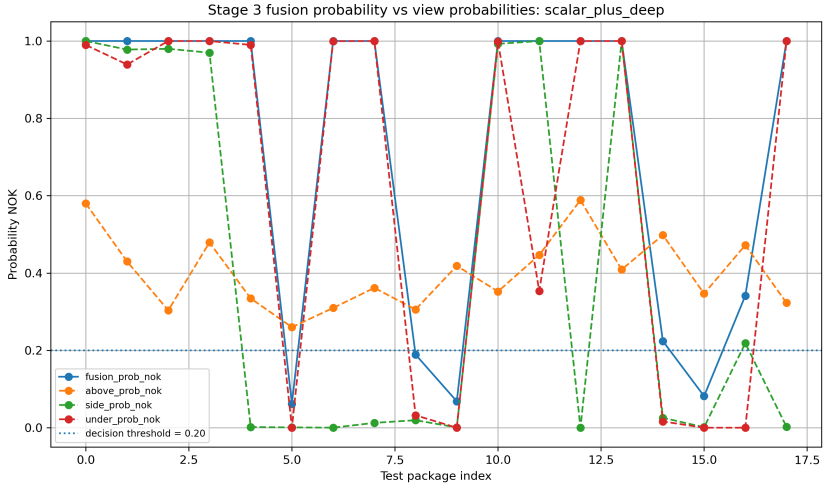
The above view-model generally produced lower confidence predictions compared to the remaining views. Despite this, the above view still contributed important information during ambiguous cases where the remaining views produced conflicting predictions. One example can be observed around package index 16, where the scalar only fusion model assigned greater importance to the above and side view probabilities, resulting in a more stable final prediction compared to the scalar plus deep feature model.

The confusion matrices presented in Figure 5.29c and Figure 5.29d demonstrate that both fusion approaches achieved identical classification performance on the test dataset. Both models produced one false positive and one false negative prediction.

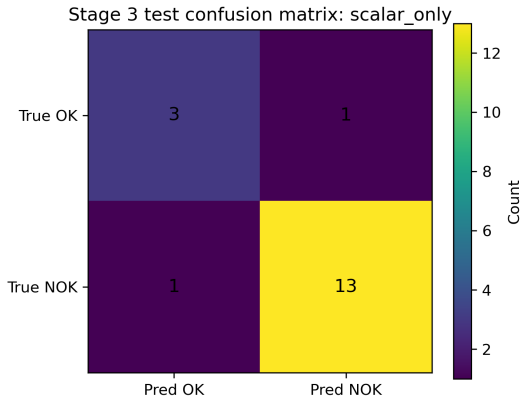
The selected decision threshold for this dataset was 0.2. This means that packages were classified as defective if the predicted defect probability exceeded 20%. The relatively low threshold indicates that the fusion model produced strong confidence separation between acceptable and defective packages. The low threshold also prioritizes sensitivity toward defective packages, reducing the probability of false negative predictions.



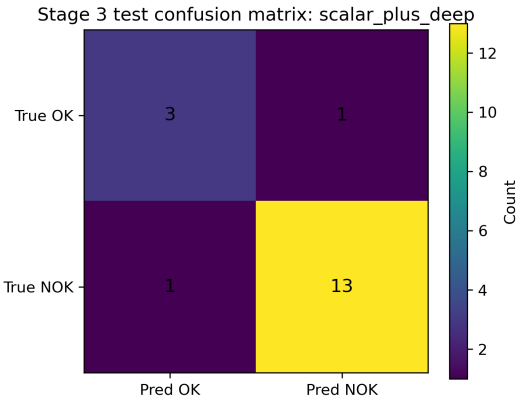
(a) Fusion and Per-View Probability Comparison for Basler$_{\mathbf{BIG}}$ with scalar only model



(b) Fusion and Per-View Probability Comparison for Basler$_{\mathbf{BIG}}$ with scalar plus deep model



(c) CM for Basler$_{\mathbf{BIG}}$ with scalar only model



(d) CM for Basler$_{\mathbf{BIG}}$ with scalar plus deep model

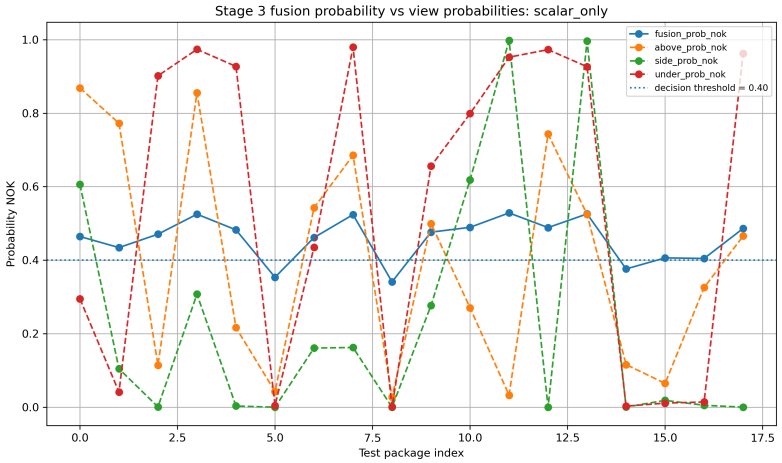
Figure 5.29: The results for the big Basler dataset when tested on the test dataset

For the Basler$_{\mathbf{SMALL}}$ dataset shown in Figure 5.30a and Figure 5.30b, the fusion model exhibited lower overall confidence compared to the Basler$_{\mathbf{BIG}}$ dataset.

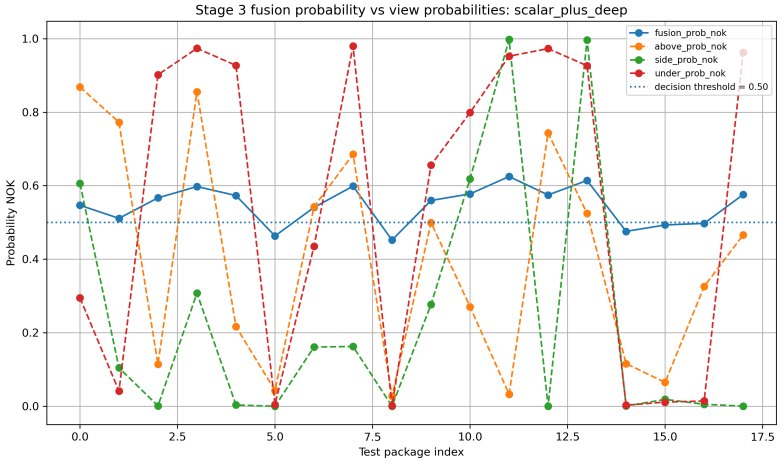
The reduced confidence mainly originated from the under and side view-models, which produced less stable probability estimates across the test samples. In contrast, the above view-model followed the fusion prediction more closely and contributed more actively to the final classification decisions.

Compared to the Basler$_{\mathbf{BIG}}$ dataset Figure 5.29a, and Figure 5.29b, the probability distributions were more compressed around the classification boundary. This indicates that the fusion model experienced greater uncertainty when separating acceptable and defective packages. The confusion matrices shown in Figure 5.30c and Figure 5.30d reveal different error characteristics for the two fusion approaches. The scalar only fusion model produced one false negative prediction by classifying an acceptable package as defective. In comparison, the scalar plus deep feature model produced one false positive prediction by classifying a defective package as acceptable.

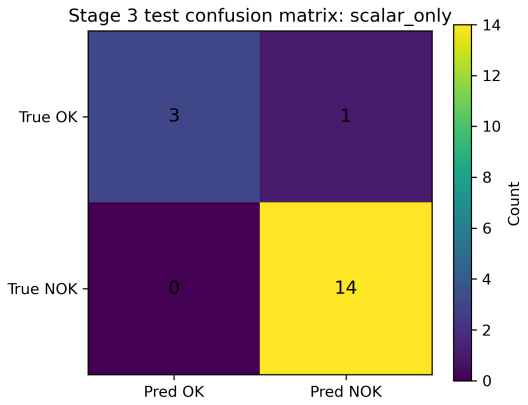
The selected threshold for the scalar only fusion model was 0.4, while the scalar plus deep feature model selected a threshold of 0.5. Compared to the Basler$_{\mathbf{BIG}}$ dataset, these higher threshold values indicate increased overlap between the predicted probabilities of the two classes. This suggests that the fusion model produced lower confidence separation between acceptable and defective packages for the smaller dataset.



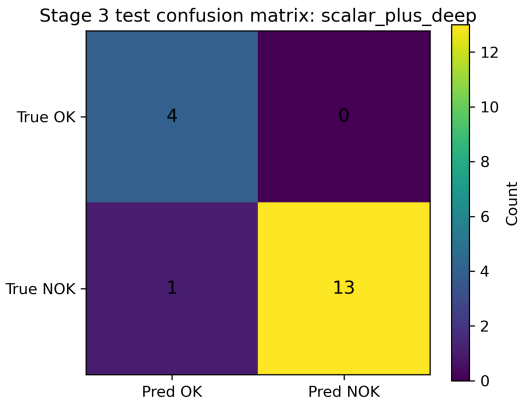
(a) Fusion and Per-View Probability Comparison for Basler$_{\mathbf{SMALL}}$ with scalar only model



(b) Fusion and Per-View Probability Comparison for Basler$_{\mathbf{SMALL}}$ with scalar plus deep model



(c) CM for Basler$_{\mathbf{SMALL}}$ with scalar only model



(d) CM for Basler$_{\mathbf{SMALL}}$ with scalar plus deep model

Figure 5.30: The results for the small Basler dataset when tested on the test dataset

The Webcam$_{\mathbf{SMALL}}$ dataset presented substantially different fusion behavior compared to the Basler datasets.

For the scalar only fusion model shown in Figure 5.31a, the predicted probabilities remained highly compressed around the classification boundary. The fusion probabilities varied only slightly between samples, even when the individual view-models produced highly confident predictions.

One example can be observed at package index 5, where all three view-models predicted defect probabilities above 90%, while the scalar fusion model produced a final probability close to 50%. This indicates that the scalar only fusion model struggled to separate acceptable and defective packages with high confidence.

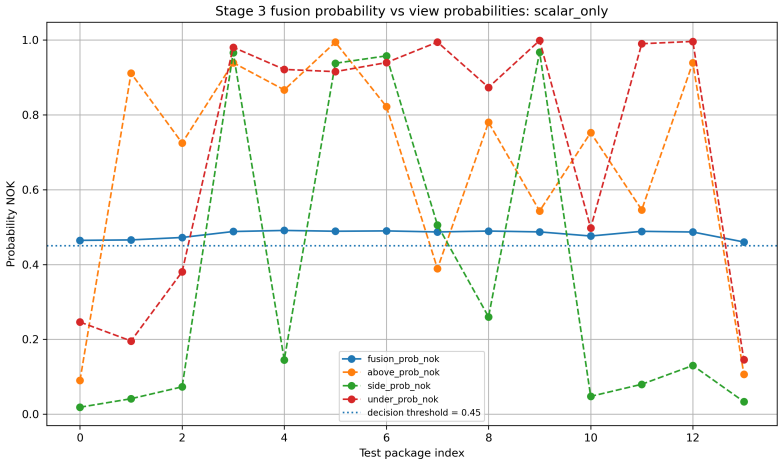
In contrast, the scalar plus deep feature fusion model shown in Figure 5.31b produced a more separated probability distribution between the two classes. Most defective packages received substantially higher probabilities, while several acceptable packages received probabilities below 50%. This indicates that the addition of deep feature representations improved the discriminative capability of the fusion model.

The prediction probability distribution shown in Figure 5.32a further illustrates this behavior. The three samples with the lowest predicted defect probabilities corresponded to acceptable packages, while the remaining defective packages received substantially higher confidence values.

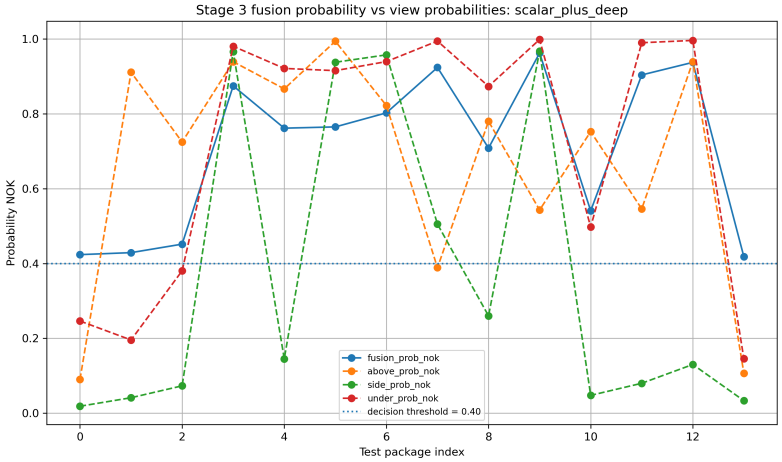
This result demonstrates the importance of threshold selection for the final classification performance. With a default threshold of 50%, several acceptable packages were incorrectly classified as defective. However, adjusting the threshold to approximately 45% would separate the acceptable packages from the defective packages more effectively and would likely result in perfect classification performance on the test dataset.

The confusion matrix for the scalar only fusion model shown in Figure 5.31c demonstrates poor class separation capability. All acceptable packages were incorrectly classified as defective, which is consistent with the highly compressed probability distribution observed in the fusion probability plots.

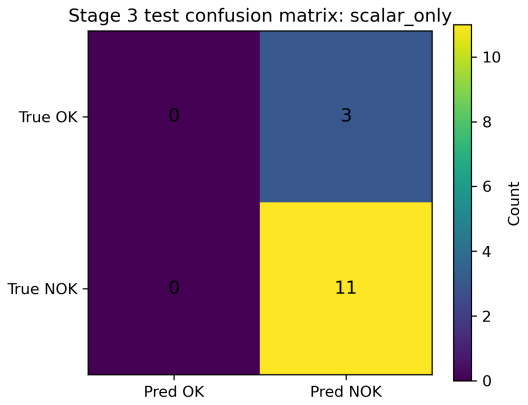
The threshold analysis for the Webcam$_{\mathbf{SMALL}}$ dataset indicates that the scalar only fusion model produced weak confidence calibration, while the scalar plus deep feature model achieved substantially better separation between the two classes. This demonstrates that the inclusion of deep feature representations improved not only the classification accuracy but also the confidence stability of the fusion model.



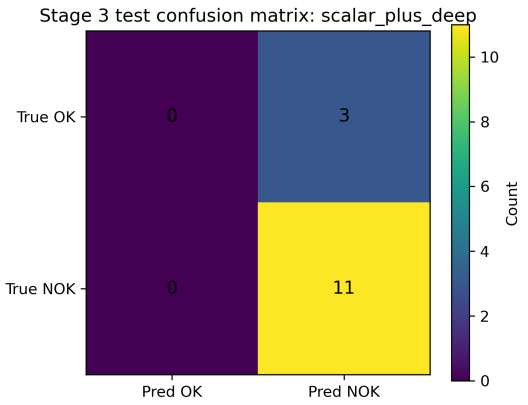
(a) Fusion and Per-View Probability Comparison for Webcam$_{\mathbf{SMALL}}$ with scalar only model



(b) Fusion and Per-View Probability Comparison for Webcam$_{\mathbf{SMALL}}$ with scalar plus deep model

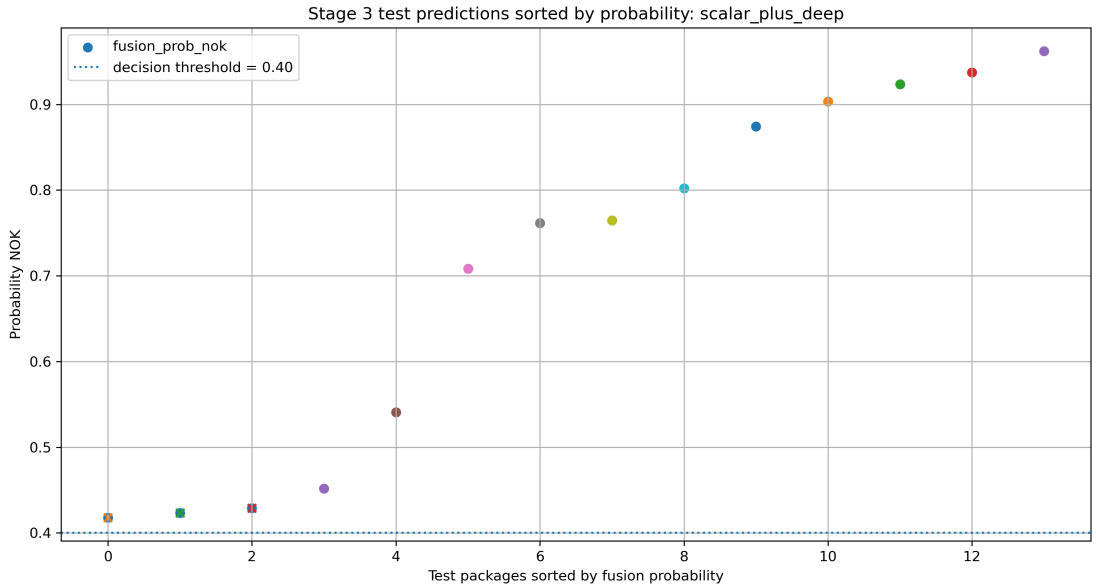


(c) CM for Webcam$_{\mathbf{SMALL}}$ with scalar only model



(d) CM for Webcam$_{\mathbf{SMALL}}$ with scalar plus deep model

Figure 5.31: The results for the small Webcam$_{\mathbf{SMALL}}$ dataset when tested on the test dataset



(a) Views the predictions that scalar plus deep feature model made

5.4.3.3 Stage 3 Summary

Overall, the fusion results demonstrated that the Basler$_{\mathbf{BIG}}$ dataset produced the clearest probability separation between acceptable and defective packages, resulting in the most stable and confident classification behavior. In contrast, the Basler$_{\mathbf{SMALL}}$ dataset showed more compressed probability distributions and increased uncertainty around the classification boundary. The Webcams$_{\mathbf{SMALL}}$ dataset exhibited the highest sensitivity to threshold selection, particularly for the scalar only fusion model, where overlapping probability distributions reduced classification reliability. However, the scalar plus deep feature fusion approach achieved noticeably better class separation and more consistent predictions.

The Stage 3 training results showed that both fusion approaches converged successfully across the evaluated datasets. The Basler$_{\mathbf{BIG}}$ dataset achieved the smoothest training behavior with consistently low loss values and stable validation accuracy, while the Webcams$_{\mathbf{SMALL}}$ dataset showed larger validation fluctuations during training, especially for the scalar only approach. Overall, the scalar plus deep feature fusion model generally converged faster and reached lower loss values earlier, indicating improved learning stability and classification capability compared to the scalar only fusion approach.

6

Discussion

This chapter discusses the obtained results, the limitations of the proposed system, and the practical challenges encountered throughout the project. It further evaluates the societal, ethical, and ecological implications of the system, while also presenting potential improvements and directions for future work.

6.1 Societal Considerations

In a study on automation and its influence on sustainable development [37], Almusharraf writes that automation has a positive impact on GDP growth and that automation contributes to increased productivity and overall economic growth. Although this economic growth is not equally distributed between sectors. Almusharraf [37] continues that high-tech industries thrive, while low-skilled, labor-intensive work struggles to adapt leading to job loss in more traditional industries. To mitigate these concerns, governments and companies should invest in different up-skilling programs and transitional support for affected workers. While these jobs are lost, automation often replaces monotonous and sometimes dangerous tasks, leading to improved workplace safety.

Even though automation contributes to economic growth, it is not evenly distributed between different income groups and without the before mentioned efforts can lead to an increase in long-term unemployment among low-skilled workers and deepen existing inequalities, which is also supported by Hémous and Olsen [38]. Almusharraf [37] also means that policy makers must prioritize strategies to ensure that the benefit of automation becomes evenly distributed.

6.2 Ethical Considerations

To assess whether the project is morally responsible, multiple ethical perspectives, including those of the worker, the business, society at large, and both local and global impacts, are needed. Ethical evaluation involves examining whether the project is fair, transparent, and socially responsible, as well as how it addresses issues of right versus wrong in its design and implementation.

From a business and societal perspective, ethical considerations include ensuring that the system does not introduce hidden biases in its design, operation, or results. The design of an ethical system requires that potential biases are identified and mitigated and that responsibility for decisions made by automated systems remains clearly assigned to human actors [39].

In addition, ethical responsibility involves considering the potential misuse or unintended consequences of the system. Even if automation is designed for efficiency and quality improvement, it could be repurposed for excessive surveillance or unfair performance evaluation if ethical safeguards are not in place. Transparency in system functionality and limitations is therefore essential to maintain trust and accountability.

Finally, ethical evaluation must also consider the broader societal and global implications. While local efficiency gains may benefit the business, the widespread adoption of automation can contribute to structural changes in the labor market. Addressing these impacts ethically requires acknowledging long-term societal effects and ensuring that technological progress aligns with social well-being and human dignity. Ethical frameworks for automated systems emphasize that legality alone is insufficient and that respect for autonomy, fairness, and harm prevention must guide technological development [39].

6.3 Ecological Considerations

The introduction of automation in this part of the production line will lead to increased electricity consumption compared to a manually operated process. However, this increase should be evaluated in relation to the overall environmental performance of the system over its entire life cycle. Life Cycle Assessment (LCA) is a commonly used methodology for assessing environmental impacts across all stages of a system's life, including manufacturing, operation and end-of-life. Previous LCA studies of automated systems indicate that an increased operational energy demand can be offset by long-term efficiency gains and optimized resource use [40].

The automated system enables higher precision, faster cycle times, and improved process stability. These improvements reduce the number of defective or discarded products, thereby reducing material waste and the environmental burden associated with the extraction, processing, and disposal of raw materials. From a life cycle

perspective, reduced scrap rates and rework contribute to a lower environmental impact per produced unit, even if electricity consumption during operation increases.

Automation also improves production flow by reducing bottlenecks and idle time within the production line. This leads to more efficient utilization of equipment and supporting infrastructure, such as lighting and auxiliary systems, which further reduces energy consumption at the system level. In addition, the automated system will be permanently integrated at the production site, eliminating transportation associated with external or temporary labor solutions. This reduction in transportation contributes to lower fossil fuel consumption and reduced emissions.

When evaluated during its operational lifetime, the automated system is therefore expected to have a net positive ecologic impact. Although the manufacturing and installation of the system introduce an initial environmental cost, life cycle-based research suggests that long-term reductions in waste generation, improved efficiency, and lower emissions per produced unit can outweigh these initial impacts [40].

6.4 The Challenges of a Small Dataset

The split sensitivity observed in Section 5.3.1 reflects a fundamental limitation of training on small datasets. With approximately 100 packages, the diversity of defect types cannot be guaranteed to be evenly distributed across splits, meaning that any single fold may fail to expose the model to the full range of variation present in the test set.

This also undermines standard validation practice. Normally, validation performance is a reliable proxy for test performance and is used to guide decisions such as epoch selection. As shown in Section 5.3.1, this assumption is not as robust, a fold with higher validation accuracy produced worse test results than one with lower validation accuracy. This makes hyperparameter tuning and model selection inherently unreliable without a larger or more carefully stratified dataset.

Increasing the number of folds reduces the variance of the mean estimate, though it does not address the underlying problem. The most direct solution would be to collect more data, particularly for rare defect types. In this dataset, some defect variations appeared in only one or two packages out of approximately 100, making it impossible to ensure representation across all three splits simultaneously.

6.5 Importance of Good Data and Labeling Strategies

As the results in Section 5.3.4 suggest, good data is fundamental to building a reliable classification system, yet it is not always straightforward to collect. One way to improve data quality is to rethink the labeling strategy itself, by planning which defects are visible from which view and which view is best suited to detect each defect type.

The current labeling strategy treats each image independently. Given a single image, the labeler determines whether the package should be classified as acceptable or defective based solely on the information visible in that image. This approach works well for viewpoints where defects are clearly distinguishable. However, for the top view, some defects are only marginally visible, introducing ambiguity in the labeling process. This uncertainty likely contributes to the behavior observed in Section 5.3.1.

Fore example, when labeling the images there was considerable uncertainty on the labeler's side, naturally leading to noisy data, as some packages would be labeled differently from another viewpoint or simply at another time. When examining two packages side by side, it is not always clear how they should be labeled under the current strategy.

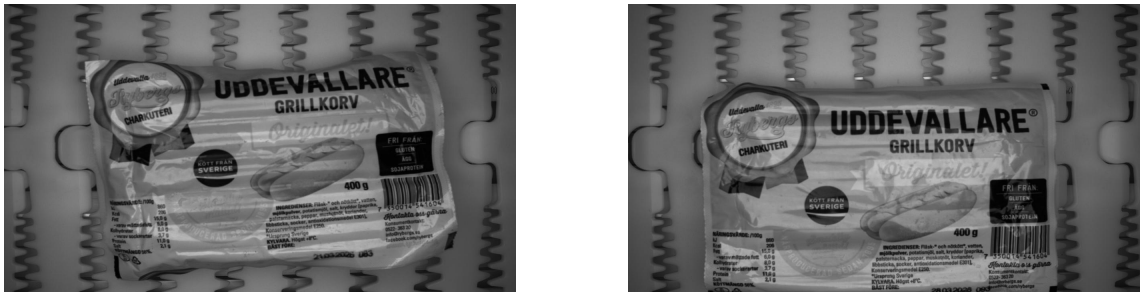


Figure 6.1: Two different packages; the left package is labeled OK and the right one is labeled NOK.

It may be clear which of these packages is okay and which is not when seen side by side, but taken in isolation it is not always obvious. This ambiguity from a human labeling perspective suggests that a performance ceiling exists for any model trained on this data that is if a humans cannot label it reliably, the model cannot learn a consistent decision boundary.

An alternative approach would be to assign each view a focused responsibility, only labeling the defects that are most reliably visible from that angle and ignoring defects better caught by another view. For example, the above view is particularly suited to detecting labeling errors and date misprints, as these are defects that may only be apparent from above. Gas leaks, on the other hand, are more clearly visible from

the under view and need not factor into the labeling of above-view images at all. This narrowing of scope per view would reduce labeling ambiguity, likely leading to cleaner labels and better model performance on each individual view.

6.6 Cameras and Illumination

In this section the effect of illumination will be discussed, requirements, improvements and alternatives.

6.6.1 Exposure Time and Illumination Requirements

While illumination represents one of the more significant cost factors in the system, adequate lighting is essential for reliable inspection of moving objects. As demonstrated in this work, sufficient illumination enables the reduction of camera exposure time to levels at which motion blur is effectively mitigated, as illustrated in Figure 5.22. It should be noted, however, that this constraint applies specifically to objects in motion, in applications where the system permits the object to be still during capture, the requirements on both illumination intensity and camera specification are reduced, a conclusion supported by the observed performance of the Webcam setup, both camera and ambient light.

With regard to resolution of camera, most cameras are not the limiting factor. As the minimum detectable defect size in this case is sufficiently large to not be a concern. Furthermore, storing images at unnecessarily high resolutions leads to increased storage requirements and training without resizing leads to substantially longer model training times, with minimal gain in model accuracy. Therefore these images are down-scaled before use. More importantly is to match the camera and lens so the working distance matches the requirements of the installation.

Beyond these considerations, industrial cameras offer practical advantages in system integration. They expose a broader range of configurable parameters — such as exposure time, gain, and trigger behavior, and are supported by dedicated software libraries that facilitate deterministic hardware control, making them considerably more suitable for deployment in automated inspection pipelines than consumer-grade alternatives.

6.6.2 Wavelength Selection

In this project, two illumination wavelengths were employed: white light and red light. White light provides broad-spectrum illumination and produces adequate contrast across a range of surface types, though it is not optimized for any particular material or defect class. The red illumination, however, presents a more fundamental compatibility issue with the object under inspection. As the sausage casings have brown tones with reddish hues, they reflect red wavelengths strongly, reducing the contrast between the object surface and potential defect regions. For optimal contrast, illumination wavelengths that are complementary to the dominant surface

color are preferred, as complementary colors are maximally absorbed rather than reflected, as illustrated in Figure 2.7. This principle is supported by the observed behavior of the plastic film defect, which is considerably more visible under red illumination, as the film itself exhibits greenish tones that are near-complementary to red, compared to white light, under which the same defect is substantially harder to discern.

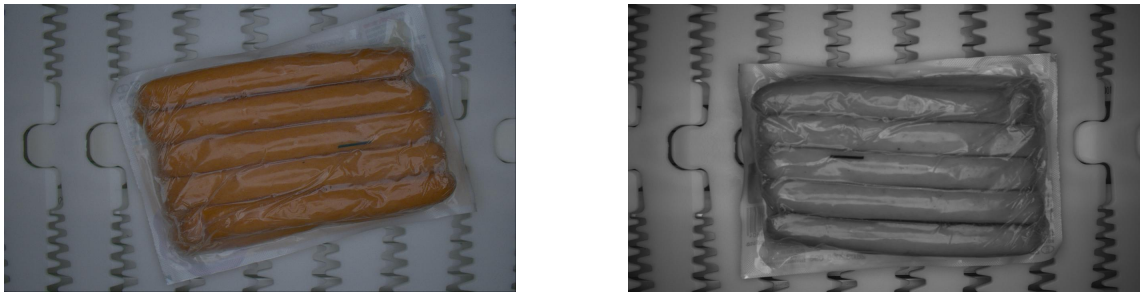


Figure 6.2: Same package, where the colored image uses a white light and the monochromatic image uses a red light.

With this in mind, blue or blue-green illumination would be a more theoretically appropriate alternative, as these wavelengths lie opposite to the reddish-brown tones of the sausage surface on the color wheel, enhancing surface contrast. It should be noted, however, that shorter wavelengths are more susceptible to scattering upon interaction with translucent or semi-transparent materials such as plastic packaging, potentially increasing unwanted glare and surface reflections. This trade-off would need to be evaluated empirically to determine whether the contrast gain outweighs the increased scattering effects.

6.6.3 Monochrome vs. Color Camera

Although with wavelength discussed it became apparent when testing monochrome and colorcamera that the diffrence in performance were minor. The choice between a color and a monochrome camera should be determined primarily by whether multiple colors is a important feature for the classification task, if there is only two colors of interest it is better to a use monochrome camera and perhaps a filter. If the defects or object properties of interest are dependent on multiple colors, a color camera is the appropriate choice. If color carries no diagnostic relevance, a monochrome camera is preferable, as it discards chromatic information that would otherwise introduce unnecessary complexity without contributing to classification performance.

The addition of the small blur associated with Bayer filter demosaicing is, in this application, of limited consequence as the defects present on these packages are sufficiently large that fine detail is not a determining factor. More importantly is the reduction in data volume afforded by monochrome imaging: single-channel images occupy considerably less storage than their three-channel color equivalents, which accelerates data handling, reduces storage requirements, and shortens neural network training times.

6.6.4 Illumination Geometry and Placement

The current setup employs a straightforward configuration, in which the camera is positioned directly behind the light source, seen in Figure 3.1. While this arrangement is simple to implement, it is prone to specular glare, as some light that hits the object surface is reflected directly back toward the camera. Glare is present , seen in Figure 6.2, in the acquired images even under diffuse illumination conditions.

Two potential strategies could be employed to mitigate this effect. First, the introduction of a polarizing filter would suppress specular reflected light while preserving the diffusely scattered light carrying surface detail. Second, angling the light source obliquely relative to the surface would transition the system toward a dark-field configuration, directing specular reflections away from the camera aperture and improving the visibility of surface irregularities.

With this project there was problems with the side view getting to little light, producing dark images where the background and the product of interest becomes hard to discern, this could be easily remedied by introducing different lighting techniques seen below. Although this was not explored due to the time-frame if the project. Example of these illumination changes can be seen with the current setup,

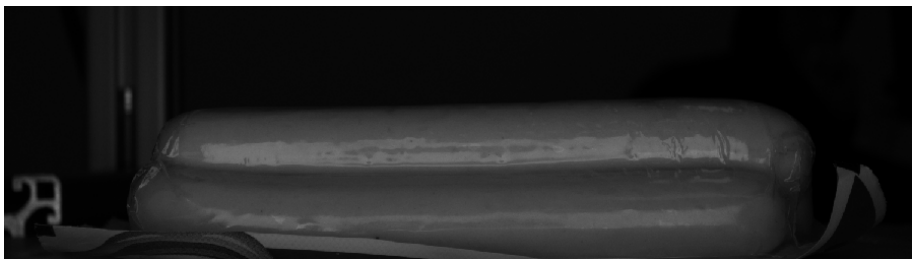


Figure 6.3: Image from the side with only the side light lit.

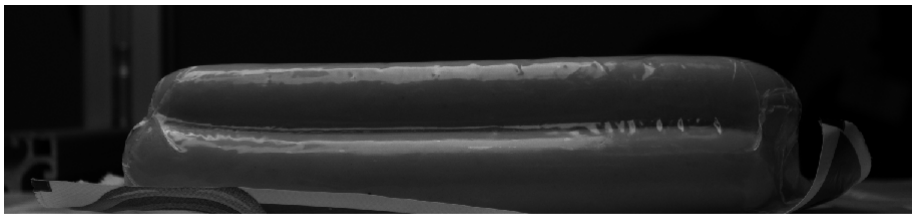


Figure 6.4: Image from the side with only the red above light lit.

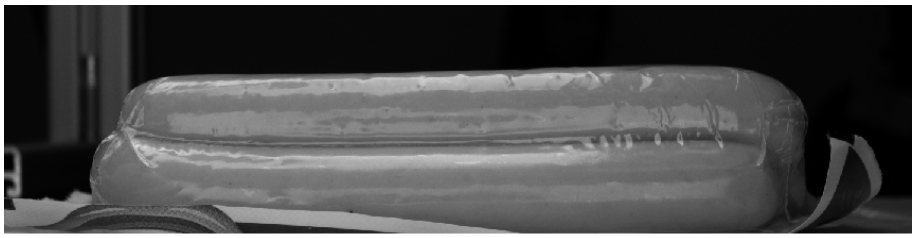


Figure 6.5: Image from the side with both the side- and above light lit

Notice that different lights highlight different things, that is necessary to take into consideration before implementation. Nevertheless, considerable scope remains for further exploration of illumination geometry in this application. Some configurations are better suited to geometric defects, such as deformations or edge irregularities, and no single geometry is likely to be universally optimal across the full range of defect classes encountered. Further reading of proper geometry and illumination techniques are recommended [26].

6.7 Automated Package Classification System Performance

Although the automated package classification pipeline demonstrated promising results, several limitations and areas for future improvement were identified throughout the experiments. The most significant limitation throughout all stages of the pipeline was the relatively small dataset size, which strongly influenced both the stability and the generalization capability of the trained models.

6.7.1 Augmentation Implementation in the Automated Model Selection Process

The augmentation methods, Section5.3.2, could be incorporated as additional tuning parameters within the automated filtering framework, Section 5.4, where different augmentation strategies, probabilities, and augmentation intensities are evaluated automatically together with the remaining hyperparameters. This would allow the system to identify which augmentation configurations are most beneficial for each camera view and dataset type.

The results also indicated that moderate augmentation generally improved model performance, while excessively aggressive augmentation could reduce performance by introducing unrealistic image variations. Future implementations should therefore focus on physically meaningful augmentations that realistically represent variations expected in industrial environments, such as illumination changes, small rotations, sensor noise, and minor geometric shifts.

Integrating augmentation into the automated process would likely improve the ro-

bustness of the selected models against variations in lighting conditions, package positioning, reflections, and small environmental changes. This is particularly important for industrial deployment, where the production environment may vary over time. Improved augmentation strategies could therefore reduce overfitting, improve generalization to unseen packages, and increase the long-term stability of the inspection system.

In addition, augmentation may become increasingly important when extending the system to new package types or when only small datasets are available during the initial deployment phase. By artificially increasing dataset diversity, the system can learn more generalized feature representations without requiring a substantially larger amount of manually collected training data.

6.7.2 Stage 1 and Stage 2

The Stage 1 (Section 5.4.1), and Stage 2 (Section 5.4.2), results demonstrated that dataset composition and dataset split selection had a major influence on the final model performance. As shown in Figure 5.27, several models experienced large reductions in test accuracy when trained and evaluated using alternative random dataset splits during Stage 2. Since Stage 2 continued training from already selected Stage 1 checkpoints, similar or improved performance would intuitively be expected if the selected models generalized robustly across different splits.

The observed performance reductions therefore suggest that the dataset itself had a larger impact on the final classification performance than the exact choice of network architecture or hyperparameter configuration. While the automated ranking strategy remains important for selecting suitable models, the results indicate that the available training data was the dominant factor limiting the robustness of the system.

A possible improvement for future work would therefore be to integrate k fold cross validation into the automated model selection process, although that would come at cost of a lengthier training time. Instead of only evaluating the top ranked models using only alternative random seed splits, the models could be evaluated across several validation folds before selecting the final architectures for the later stages. Such an approach would likely produce more statistically robust model selection and reduce the sensitivity to specific dataset splits.

Another important observation throughout Stage 1 and Stage 2 was that the highest performing models generally favored the largest image resolution used during training, which in this work corresponded to an image size of 512. Larger image resolutions allowed the networks to preserve smaller visual details and subtle defect patterns that may otherwise disappear during downsampling.

However, increasing the image resolution also introduced several practical limitations. Larger images significantly increased the training time. This creates a trade

off between classification performance and computational efficiency, which becomes particularly important when evaluating a large number of architectures and hyperparameter configurations.

The results further demonstrated clear differences between the three camera views. The under view consistently achieved the most stable and robust classification performance across the evaluated datasets and dataset splits. In contrast, the above view showed greater sensitivity to reduced dataset size and dataset split variations. Further supporting the suggestion that certain views contain more stable defect related information than others and that future work could further investigate adaptive view weighting or view selection strategies.

6.7.3 Fusion Classification

The Stage 3 fusion classification results continues to highlight the challenges associated with small datasets. Since the fusion model operates on package level predictions rather than individual images, the effective dataset size became substantially smaller. The largest fusion dataset contained approximately 80 training packages, 17 validation packages, and 17 test packages, which is extremely limited for training deep learning based fusion models.

The small test dataset was particularly limiting for evaluating the final classification performance. For example, the largest test dataset only contained four acceptable packages, meaning that a single incorrect classification had a large impact on the final evaluation metrics. A substantially larger and more balanced package level dataset would therefore be required in order to fully evaluate the robustness and practical applicability of the fusion classification strategy.

Despite these limitations, the fusion models demonstrated promising behavior, especially for the large Basler dataset shown in Figure 5.29c. The fusion approach successfully classified twelve out of fourteen defective packages correctly while also maintaining relatively stable confidence separation between acceptable and defective samples.

The threshold analysis further demonstrated that the confidence calibration of the fusion model had a major influence on the final classification performance. This behavior was especially visible for the WebCam$_{\textbf{SMALL}}$ dataset, where relatively small threshold adjustments produced large changes in the final accuracy. Such behavior indicates that the fusion model confidence distributions for acceptable and defective packages partially overlapped, making the final classification highly sensitive to threshold selection.

The scalar plus deep feature fusion approach generally produced more stable probability separation and more reliable confidence distributions compared to the scalar only fusion strategy. This suggests that the inclusion of deep feature representations improved not only the classification capability of the fusion model but also the

confidence calibration and class separation stability.

Future work could further investigate alternative fusion strategies, adaptive threshold selection methods, confidence calibration techniques, and more advanced weighting approaches between the different camera views. In addition, evaluating the fusion system on larger industrial datasets collected under varying lighting conditions, package orientations, and conveyor speeds would provide a more representative evaluation of the practical robustness of the proposed approach.

6.8 Future work

This section presents possible improvements and future research directions for the proposed inspection system. The discussion includes both technical enhancements and practical considerations related to installation, model evaluation, anomaly detection, and dataset development for improved robustness and industrial applicability.

6.8.1 Installation and Handling Plan

One of the most important areas for further work in this project, whether applied to sausage packaging or other types of packages, is the development of a proper installation and handling plan. Such a plan makes it possible to minimize errors both during system design (e.g. selecting unsuitable illumination or incorrect cameras) and during system operation (e.g. mislabelling or misuse). A proposed draft of such a plan is presented below.

1. **Assess the need for vision.** Is machine vision actually required? Could simpler, more robust sensors create a more intuitive and reliable system?

2. **Define what needs to be detected.** What are the product characteristics (size, color, texture)? What are the defect characteristics? Should detection occur while the product is in motion?

3. **Determine views and number of cameras.** Which defects are visible from each view? Can the same defect be seen from multiple views? If so, should effort be concentrated on the most informative view? See Section 6.5.

4. **Configure the camera and illumination setup.** For each necessary view, define camera distances and angles, exposure time, illumination technique, wavelength, and light-source distance. The setup should be tailored to the specific detection problem.

5. **Establish a labeling protocol.** Define clearly what constitutes a defect and how strictly it should be interpreted. Labeling must be consistent. Decide who performs labeling and whether defects visible from only one view should be labeled differently or excluded entirely.

6. **Define data ownership and storage.** Clarify who owns the data. Decide on a storage strategy: cloud, local hardware, or both. Define a backup policy and access controls.

7. **Select a data augmentation strategy.** Determine which augmentations are physically meaningful. For example, horizontal flipping may be appropriate for symmetric products but misleading for oriented ones.

6.8.2 Scoring System for Ranking the Models

The approach that were taken were that the models should have false negative predictions rather then false positives which was included in the scoring equation as 3 times the punishment for false positives than false negatives. The models were rewarded when a stable learning curve appeared since a small dataset in general has very unstable learning curves. The approach to this was a mean value over the three best validation accuracy epochs. This system was created to generate models that were stable for all types of test datasets and be used as pretrained models for further training when mounted in the factory. This scoring system needs to be evaluated when the datasets has reached a bigger size to move over to just validation accuracy and learning loss.

6.8.3 Anomaly Detection

An alternative worth exploring is anomaly detection, where the model is trained exclusively on OK packages and defects are identified as deviations from the learned normal distribution. Avoiding completely having to supervise the learning by labeling each package but rather only feed OK packages in a collection phase. This approach is well suited to scenarios where defects are scarce, as it removes the dependency on labeled NOK data entirely. Methods such as one-class SVM or autoencoders are natural candidates. With a dataset more representative of real production conditions, where defects are rare, anomaly detection could be a more appropriate than binary classification.

6.8.4 Test Product

This project has focused on classifying vacuum-sealed sausage packages, which present a more challenging surface compared to a simple box. The uneven geometry makes illumination and information gathering from the necessary regions more difficult, while the packages also contain a larger variation of both defects and acceptable configurations, meaning that each package is more or less unique. This is beneficial, as the complexity of the package gives makes the classification non-trivial with other methods.

However, one challenge with sausage packages is that they are fresh products and change over time, meaning that the dataset gradually degrades. As a result, data collected from the same packages at a later stage is not directly comparable to data collected earlier. For future work, it could therefore be beneficial to change the

product type, since the core machine learning concepts remain largely the same, although the illumination and camera setup may need to be adjusted.

If the primary focus is machine learning rather than the full industrial setup, it would also be beneficial to use a larger public dataset while limaiting the amount of training data but maintaining large validation and test sets. This would simplify the evaluation of different approaches, since it is difficult to reliably assess strategies when working with very small datasets.

6.8.5 Integration of Augmentation into the Automated Classification System

Future work should focus on extending the automated classification pipeline to include augmentation as an optimization parameter. Rather than evaluating a fixed set of augmentation strategies, the framework could automatically search for suitable augmentation types, probabilities, and intensities together with the remaining hyperparameters.

This would allow the system to identify augmentation configurations that are specific to each camera view and dataset. Furthermore, augmentation policies could be adapted based on dataset size, class imbalance, and observed model performance. Such an approach would increase the degree of automation within the model selection process and reduce the need for manual augmentation tuning when deploying the system to new products or inspection tasks.

7

Conclusions

The goal of this project was to develop and evaluate an automated multi-view package classification system capable of detecting defects in vacuum-sealed sausage packages using industrial vision systems and deep learning-based classification models.

This project shows promising results that CNNs are well suited for detecting both alignment-related defects and other visual defects in vacuum-sealed packages, but needs more data to be able with certainty come to that conclusion. The models achieved high accuracy and F1-scores, while also demonstrating the importance of larger datasets, even though the smaller datasets still achieved promising performance results. However, the results also showed that defects which are visually difficult to identify, such as alignment defects from the top-view perspective, resulted in lower classification performance. To achieve more robust and reliable inspection results, combining CNN-based vision systems with additional sensing methods, such as weight measurements, could be beneficial. Although CNNs provide strong performance, they also introduce challenges related to hyperparameter tuning, as several parameters must be optimized to achieve satisfactory results.

The fusion model showed promising results when combining information from multiple camera views, improving the robustness of the classification system, especially in situations where individual views showed uncertainty.

An additional advantage of CNN-based systems is their modularity and scalability. Adapting the inspection system to a new product mainly requires collecting and training on new data rather than redesigning the entire system. However, certain hardware aspects, such as camera placement and illumination, must still be adjusted depending on the product properties, including color, size, and material characteristics.

The adaptability of the system is further demonstrated through the benefits of transfer learning using pretrained networks. Pretrained models reduced both the required training time and the amount of training data needed to achieve promising performance, while also opening up the possibility of reusing already trained networks for similar package types. This makes transfer learning a particularly suitable technique when working with small datasets. Data augmentation techniques also showed im-

provements in model performance under limited dataset conditions, provided that the augmentations were not excessively applied.

The project also highlighted the importance of robust evaluation methods when working with limited data. Using techniques such as k-fold cross-validation or multiple dataset splits provides a more representative estimate of model performance and reduces the risk of misleading evaluation results. The evaluation of a suitable ranking system for the models also proved important in order to achieve the requirements of the process, which in this project focused on minimizing the number of false positive classifications.

Overall, the project demonstrates that deep learning-based multi-view inspection systems have potential for automated defect classification in industrial food packaging applications, provided that sufficient and representative training data is available.

Bibliography

[1] A. Arshaghi, M. Ashourian, and L. Ghabeli, "Potato diseases detection and classification using deep learning methods," *Multimedia Tools and Applications*, vol. 82, no. 4, 5725 – 5742, 2023, Cited by: 113. DOI: 10.1007/s11042-022-13390-1. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85135260046&doi=10.1007%2fs11042-022-13390-1&partnerID=40&md5=8aebcc5b55eab07a4afd9bf05396dd05.

[2] S. Meivel, K. I. Devi, A. S. Subramanian, and G. Kalaiarasi, "Remote sensing analysis of the lidar drone mapping system for detecting damages to buildings, roads, and bridges using the faster cnn method," *Journal of the Indian Society of Remote Sensing*, vol. 53, no. 2, 327 – 343, 2025, Cited by: 4. DOI: 10.1007/s12524-024-01963-6. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85202465128&doi=10.1007%2fs12524-024-01963-6&partnerID=40&md5=14b641ade59577262a24c10bd05ebd27.

[3] X. Gong, Y. Bai, Y. Bai, Y. Liu, and H. Mu, "Application of deep learning in defect detection," *Journal of Physics: Conference Series*, vol. 1684, p. 012 030, 2020.

[4] Z. Ren, F. Fang, N. Yan, and Y. Wu, "State of the art in defect detection based on machine vision," *Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 9, pp. 661–691, 2023.

[5] J. Yang, S. Li, Z. Wang, H. Dong, J. Wang, and S. Tang, "Using deep learning to detect defects in manufacturing: A comprehensive survey and current challenges," *Materials*, vol. 13, 2020.

[6] Author(s), "Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection," *CIRP Annals*, vol. 65, pp. 417–420, 2016. DOI: 10.1016/j.cirp.2016.04.072.

[7] Aphex34, *Typical CNN architecture image*, https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png, Accessed: 2026-05-26, 2024.

[8] R. Khanam, M. Hussain, R. Hill, and P. Allen, "A comprehensive review of convolutional neural networks for defect detection in industrial applications," *IEEE Access*, vol. 12, pp. 94 250–94 295, 2024. DOI: 10.1109/ACCESS.2024.3425166.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[10] Y. Kumar, "Understanding generalization, robustness, and interpretability in low-capacity neural networks," *arXiv preprint arXiv:2507.16278*, 2025. DOI: 10.48550/arXiv.2507.16278. [Online]. Available: https://arxiv.org/abs/2507.16278.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, 2015.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[15] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[16] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[17] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. CRC Press, 2012.

[18] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[19] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[20] Edmund Optics, *Imaging electronics 101: Understanding camera sensors for machine vision applications*, https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-camera-sensors-for-machine-vision-applications/, [Online; accessed: 2026-02-03], 2021.

[21] G. Hollows and N. James. "Understanding focal length and field of view," Edmund Optics, Accessed: Jun. 2, 2026. [Online]. Available: https://www.edmundoptics.com/knowledge-center/application-notes/imaging/understanding-focal-length-and-field-of-view/.

[22] Teledyne Vision Solutions. "Rolling vs global shutter." Accessed: 2026-02-02, Teledyne Vision Solutions. [Online]. Available: https://www.teledynevisionsolutions.com/learn/learning-center/imaging-fundamentals/rolling-vs-global-shutter/.

[23] C. Coates and I. Juvan-Beaulieu. "Rolling shutter vs global shutter scmos camera mode." Andor Technology (Oxford Instruments). Accessed: 2026-04-28. [Online]. Available: https://andor.oxinst.com/learning/view/article/rolling-and-global-shutter.

[24] A. Chakrabarti, W. T. Freeman, and T. Zickler, "Rethinking color cameras," in *Proceedings of the IEEE International Conference on Computational Photography (ICCP)*, IEEE, 2014, pp. 1–8. DOI: 10.1109/ICCPHOT.2014.6831801.

[25] LiamUK, *Bayer filter*, https://commons.wikimedia.org/w/index.php?curid=28250, Wikimedia Commons, licensed under CC BY-SA 3.0, Accessed: 2026-04-28, n.d.

[26] "Chapter 12 - modeling illumination variation with spherical harmonics," in *Face Processing*, W. Zhao and R. Chellappa, Eds., Burlington: Academic Press, 2006, pp. 385–424, ISBN: 978-0-12-088452-0. DOI: https://doi.org/10.1016/B978-012088452-0/50013-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780120884520500133.

[27] Advanced Illumination, *Fd2 series high intensity back-lit flat diffuse lights: Product datasheet*, Revision date: 2025-04-21. Accessed: 2026-04-28, Advanced Illumination, Apr. 2025. [Online]. Available: https://advancedillumination.com/wp-content/uploads/2025/04/FD2_Series_Datasheet.pdf.

[28] Advanced Illumination. "A practical guide to machine vision lighting." Accessed: 2026-04-28. [Online]. Available: https://advancedillumination.com/a-practical-guide-to-machine-vision-lighting/.

[29] Intelgic. "Illumination color in machine vision: Choose the right spectrum," Accessed: Jun. 2, 2026. [Online]. Available: https://intelgic.com/Illumination-color-in-machine-vision-choose-right-spectrum.

[30] Logitech, *Logitech hd webcam c270 for education - one pager*, https://www.logitech.com/content/dam/logitech/en/support/qsg/education-center/logitech-c270-for-education-one-pager.pdf, Accessed: 2026-03-12, 2022.

[31] Basler AG, *Ace 2 a2a1920-51gmbas – gige area scan camera*, Accessed: 2025, 2025. [Online]. Available: https://www.baslerweb.com/en/shop/a2a1920-51gmbas/.

[32] T. I. Source, *Tcl 0814 5mp lens technical data*, Technical data sheet for TCL 0814 5MP lens (industrial optics), published by The Imaging Source, 2019. [Online]. Available: https://s1-dl.theimagingsource.com/api/2.5/packages/documentation/factsheet-optic/tcl08145mp/f0bd545a-eb7c-5542-bd13-674963bbce71/tcl08145mp.en_US.pdf.

[33] TPL Vision, *Medium flat mfdome+ technical datasheet*, https://www.tpl-vision.com/documents/techsheets/TPL_MFDOME_PLUS_TechSheet_EN.pdf, Accessed: 2026-02-11, 2024.

[34] NVIDIA Corporation, *Nvidia jetson agx orin 64gb developer kit technical datasheet*, https://www.electrokit.com/upload/product/41020/41020588/jetson-agx-orin-64GB-developer-kit-datasheet-web-us.pdf, Accessed: 2026-02-11, 2025.

[35] E. Cumbajin et al., "A systematic review on deep learning with cnns applied to surface defect detection," *Journal of Imaging*, vol. 9, no. 10, p. 193, 2023. DOI: 10.3390/jimaging9100193.

[36] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, *The marginal value of adaptive gradient methods in machine learning*, 2018. arXiv: 1705.08292 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1705.08292.

[37] A. I. Almusharraf, "Automation and its influence on sustainable development: Economic, social, and environmental dimensions," *Sustainability*, vol. 17, no. 4,

2025, ISSN: 2071-1050. DOI: 10.3390/su17041754. [Online]. Available: https://www.mdpi.com/2071-1050/17/4/1754.

[38] B. Schmidpeter and R. Winter-Ebmer, "Automation, unemployment, and the role of labor market training," *European Economic Review*, vol. 137, p. 103 808, 2021, ISSN: 0014-2921. DOI: https://doi.org/10.1016/j.euroecorev.2021.103808. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0014292121001549.

[39] B. D. Mittelstadt, P. Allo, M. Taddeo, S. Wachter, and L. Floridi, "The ethics of algorithms: Mapping the debate," *Big Data & Society*, Jul. 2016. DOI: 10.1177/2053951716679679.

[40] Author(s), "Life cycle assessment of an automated multi-cellular case-picking system," *Journal of Manufacturing Systems*, vol. 79, pp. 419–434, 2025. DOI: 10.1016/j.jmsy.2025.01.017.

[41] Infineon Technologies (formerly International Rectifier), *Irlz44npbf power mosfet datasheet*, https://www.electrokit.com/upload/product/41013/41013937/140906.pdf, Accessed: 2026-02-11, 2003.

A

Sausage Defects

Images of the different defects observed in the packages.

Figure A.1: Missing sausage

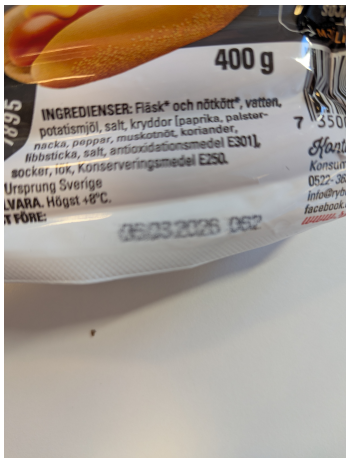


Figure A.2: Unreadable date

Figure A.3: Hole in vacuum seal

Figure A.4: Broken sausage

Figure A.5: An accepted package from above

Figure A.6: Miss-Aligned sausages

Figure A.7: A missing layer of sausage

Figure A.8: Plastic film stuck on sausage

Figure A.9: An accepted package from below

B

Electrical Design

This section will explain the different electrical choices made for the project. The electrical design can be divided into three parts, the control and power for the led lights, the control and power for the cameras and lastly the power for the controller, in this project a System-on-Module more specifically a *Jetson AGX Orin*[34] board is used, which uses laser detection sensor for timing the pictures and lighting conditions. The divide of design parts is done due to the the hardware have different source voltage needs.

B.1 Electrical Design for Lights and Sensor.

To be able to design a electrical circuit the current and source voltage is needed. As mentioned in the section Illumination, three separate light sources are used of varying sizes and wavelengths, all needing a source voltage of 24 V specified by manufacturer. The red $300 \times 300\,\text{mm}$ light will be shortened as L_{300} which is the same as for the white light of the same size as the power characteristics are identical, where the $200 \times 200\,\text{mm}$ light will be called L_{200}. The area is needed for calculating the current draw:

$$A_{L_{300}} = 300\,\text{mm} \times 300\,\text{mm} = 90\,000\,\text{mm}^2 = 900\,\text{cm}^2,$$
$$A_{L_{200}} = 300\,\text{mm} \times 200\,\text{mm} = 60\,000\,\text{mm}^2 = 600\,\text{cm}^2.$$

MFDOME PLUS has a power consumption of $1.32\,\text{W}/25\text{cm}^2$ which gives:

$$\begin{aligned} P_{L_{300}} &= 0.0528\,\text{W}\,\text{cm}^{-2} \times 900\,\text{cm}^2 \\ &= 47.52\,\text{W} \\ P_{L_{200}} &= 0.0528\,\text{W}\,\text{cm}^{-2} \times 600\,\text{cm}^2 \\ &= 31.68\,\text{W} \end{aligned}$$

and with a source voltage of 24 V the current can be calculated:

$$I_{L_{300}} = \frac{47.52\,\text{W}}{24\,\text{V}} = 1.98\,\text{A}, \tag{B.1}$$

$$I_{L_{200}} = \frac{31.68\,\text{W}}{24\,\text{V}} = 1.32\,\text{A}. \tag{B.2}$$

While the laser sensors use a effect of 1.8 W which gives at the source voltage a current draw of $\frac{1.8\,\mathrm{W}}{24\,\mathrm{V}} = 0.075\,\mathrm{A} = I_{Laser}$. This will all be powered by a bench power supply that can output 24 V and up to 10 A. However, to control the lights and sensor, GPIO pins will be used. These pins operate at 3.3 V and will be fried if directly connected into the main circuit, this is the reason behind the design seen in the next section.

B.2 Control and power for illumination

To power the jetson and keep the power supply healthy, the manufacture power brick will be used. As mentioned to control the lights, the GPIO pins will be used to strobe the lights. This will be done by pulling the NPN trigger line down to ground, seen in figure B.1

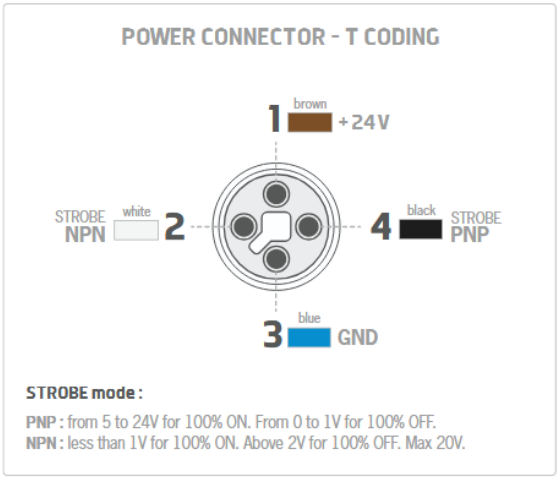


Figure B.1: Pinout for the light, picture taken from TPL electronics manual.[33]

To be able to do this with the GPIO pin a n-channel MOSFET is introduced to work as a switch. It is setup in a configuration seen below:

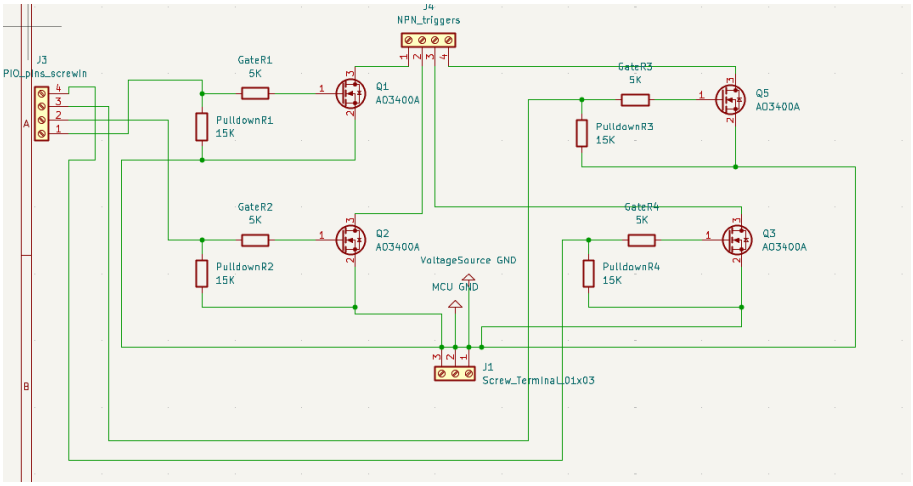


Figure B.2: Circuit designed for up to four lights.

The chosen MOSFET was *IRLz44NPbF*[41] later changed into *AO3400A SOT-23 N-ch* but the calculations follow the same pattern. The *IRLz44NPbF* can handle drain voltages, V_{DS}, up to 55 V an drain current, I_D of 47 A, which is well with in margin. What also needs to take in to consideration is if it reacts to low gate voltage, V_{GS} and for the jetson it needs to react at a the GPIO signal of 3.3 V, which it does seen in the figure below.

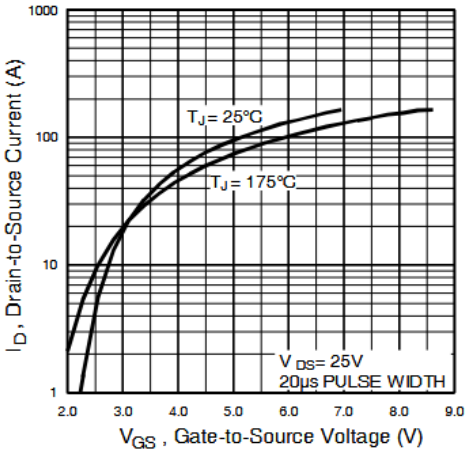


Figure B.3: Typical transfer characteristics, at 3.3 V is not ideal but will be able to switch on and off. Figure taken form the manufactures datasheet.[41]

Although due to a small V_{GS} the static drain-to-source on-resistance, $R_{DS(on)}$ will be quite high, at $V_{GS} = 4V$ and $I_D = 21A$ it is at 0.035 Ω and with smaller V_{GS} there is a higher $R_{DS(on)}$. So there is risk of thermal problems. Assumed $R_{DS(on)}$ at $V_{GS} = 3.3V$ is around 0.1 Ω at ambient temperature of 25°C. This can almost double, with rising temperature, seen according to the manufacture seen below:

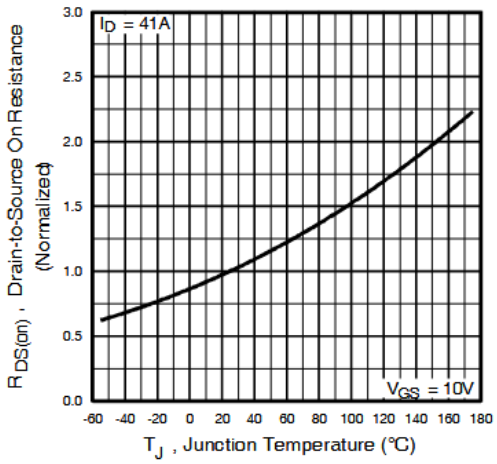


Figure B.4: $R_{DS(on)}$ Vs. Temperature. Figure taken form the manufactures datasheet.[41]

Given parameter:

$$\begin{aligned} I_D &= 1.98 \text{ A} \\ R_{DS(on)} &\approx 0.2 \ \Omega \\ R_{\theta JA} &= 62 \ ^\circ\text{C/W} \\ T_{\text{ambient}} &= 23 \ ^\circ\text{C} \end{aligned}$$

The conduction power loss is:

$$P = I_D^2 \cdot R_{DS(on)}$$

stitute the values:

$$P = (1.98)^2 \cdot 0.2 \approx 3.9204 \cdot 0.2 \approx 0.784 \text{ W}$$

The junction temperature rise above ambient is:

$$\Delta T = P \cdot R_{\theta JA}$$

$$\Delta T = 0.784 \cdot 62 \approx 48.6 \ ^\circ\text{C}$$

The junction temperature is:

$$T_J = T_{\text{ambient}} + \Delta T$$

$$T_J = 23 + 48.6 \approx 51.6\ ^\circ\text{C}$$

Even with the higher $R_{DS(on)}$ at $V_{GS} = 3.3$ V and under worst case scenario of temperature, is well below thermal operating range of 175 °C, so the device is safe thermally at this current. Although it will not be safe if multiple lights is connected to one MOSFET, so they need to be connected to three separate MOSFETs.

Only using a MOSFET would lead to two additional problems, inrush current and floating values. To avoid damaging the board from inrush current when the MOSFET is discharging, a gate resistor is needed to control the current. The chosen GPIO pins is rated for a max current of 1 mA, with a maximum voltage over it being V_{GS} gives the necessity that the resistor is over 3300 kΩ, $R_g \geq \frac{V_{GS}}{1\,\text{mA}}$. Additionally, to avoid floating gate voltage—which can lead to unpredictable switching a pulldown resistor should be connected between the MOSFET gate and source (ground). Ensuring that when the GPIO is not actively driving the gate, the voltage is pulled to ground. Which in turn keeps the MOSFET reliably turned off. A typical value for the pulldown resistor is in the range of 10 kΩ.

To determine when to strobe the LED, laser sensors, *WTB11-2P2431*, are used as triggers. These sensors operate with a PNP output that sources current when activated. In combination with an optocoupler, as shown below, the circuit functions as a trigger that drives a GPIO pin HIGH while protecting the Jetson board from harmful currents and voltages through galvanic isolation.

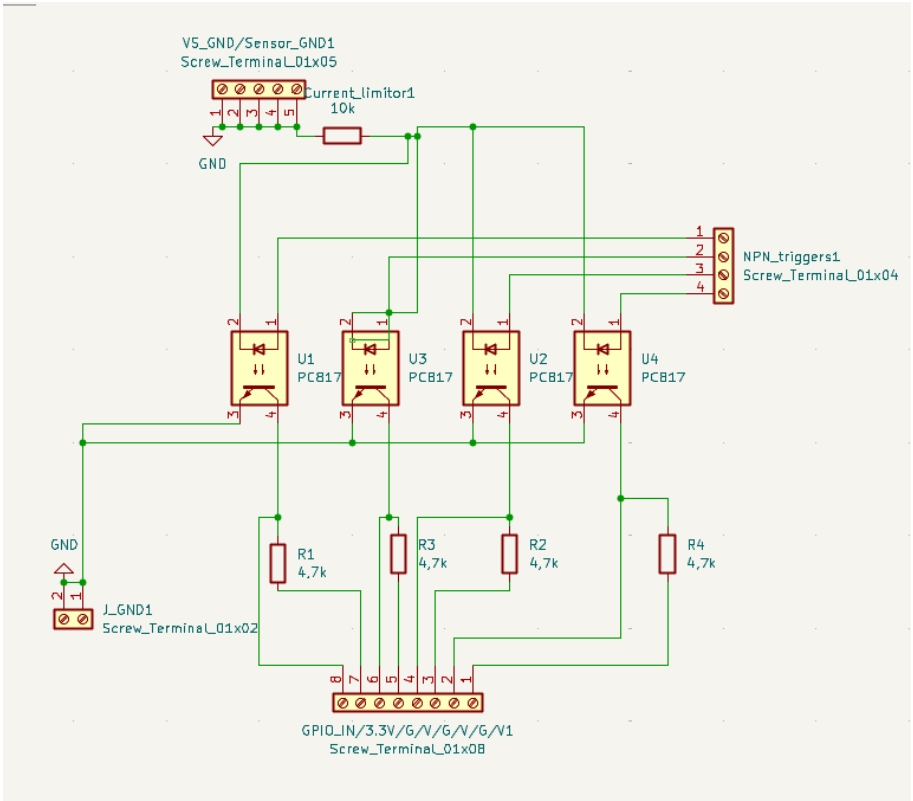


Figure B.5: Optocoupler module, uses a PCB817 optocoupler.

With the voltage source from the sensor being 21.5 V and the resistance of 4700Ω the current is $\frac{21.5}{4700} = 4.5\,\text{mA}$, which is safe for the PC817 opto coupler.

C

Augmentation Implementation

The augmentation pipeline was developed with the assistance of a large language model (LLM). The generated code was reviewed and validated to ensure correctness before integration into the training pipeline.

C.1 Photometric Augmentations

Standard photometric transformations, including brightness scaling, contrast adjustment, horizontal and vertical flips, and rotation, were implemented using the `Torchvision` and `Pillow` libraries and are not reproduced here. Two domain-specific augmentations were generated by AI and are described below.

C.2 Specular Highlight

The specular highlight augmentation simulates the reflective appearance of plastic packaging surfaces under direct lighting. A Gaussian intensity blob is placed at a randomly sampled position within the central region of the image, as industrial camera setups typically produce highlights near the centre of the field of view. The blob intensity is drawn uniformly from the range $[20, 50]$ and added to all color channels, producing a localized brightening consistent with specular reflection.

Listing C.1: Specular highlight augmentation.

```
def add_specular_blob(img: Image.Image) -> Image.Image:
    arr = np.array(img).astype(np.float32)
    h, w = arr.shape[:2]
    cx = np.random.randint(w // 4, 3 * w // 4)
    cy = np.random.randint(h // 4, 3 * h // 4)
    Y, X = np.ogrid[:h, :w]
    mask = np.exp(-((X - cx) ** 2 + (Y - cy) ** 2) / (2 * (w // 8) ** 2))
    arr += mask[:, :, None] * np.random.uniform(20, 50)
    return Image.fromarray(np.clip(arr, 0, 255).astype(np.uint8))
```

C.3 Gaussian Noise

The Gaussian noise augmentation simulates sensor noise. Per-pixel noise is sampled from a zero-mean Gaussian distribution $\mathcal{N}(0, \sigma^2)$, where the standard deviation σ is drawn uniformly from the range $[2, 6]$ for each image. This range was chosen to produce noise levels that are perceptible but do not obscure defect-relevant features. The resulting values are clipped to the valid pixel range $[0, 255]$.

Listing C.2: Gaussian noise augmentation.

```
def add_gaussian_noise(img: Image.Image) -> Image.Image:
    arr = np.array(img).astype(np.float32)
    std = np.random.uniform(2, 6)
    noise = np.random.normal(0, std, arr.shape)
    return Image.fromarray(np.clip(arr + noise, 0, 255).astype(np.uint8))
```

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY