* **Slide 1 - Title Page**
* **Slide 2-3 - Medical Dressings Production**
  + Sections of production Line [1]
    - Material section – Where material reels are placed on rollers (spindles), and are then fed into joining section
    - Joining section – Materials are joined together, and cut to final size of a dressing
    - Sealing section – Dressings are sealed in a sachet, and are passed out to the packing area
  + Dust particles, and other foreign bodies being trapped between layers of materials
  + Joining tape not being fully rejected
  + Unexpected cuts in dressings during packaging stage
* **Slide 4-8 - Current state of visual inspection** 
  + **Current Vision Systems** [2] – Production is currently using a traditional MV system, which can visually measure the dimensions of dressings, to check if they comply with specification
  + **Visual inspection** [3] – Person is observing the dressings at the production line for defects. Prone to fatigue, distraction, and subjective decisions, which can lead to error rates of 20-30% in manual inspection tasks [4]
  + **QC checks** [5] – Random quality checks performed by Quality Controllers during the production, plus quality checks at the start, and end of each batch.
  + **Customer returns**
* **Slide 8 - Methodology**
  + **Literature Review** - Conduct a thorough review of relevant research papers, case studies, and articles to understand the current state of the art in machine vision, object detection, and classification models
  + **Data Collection -** Capture images of dressings with three different defects: foreign body, red tape, and cut. Ensure that the dataset is balanced and includes a variety of instances for each defect type to improve the model’s ability to generalise
  + **Data Pre-Processing -** Pre-process the images by resizing, normalisation, and augmentation, if necessary, to prepare the dataset for model training.
  + **Model Selection -** Choose one or more suitable models based on the literature review and project requirements
  + **Model Training -** Train the selected AI model(s) using the pre-processed dataset. Monitor the training process to avoid overfitting and ensure optimal performance.
  + **Model Evaluation -** Evaluate the trained model(s) using appropriate metrics, such as accuracy, precision, recall, and F1 score
  + **Model Optimisation -** If the detection accuracy is lower than expected, gathering more data, adjusting model parameters, or using other models can be considered
  + **Hardware Testing -** Test the trained model on a Raspberry Pi or other suitable hardware to check if it meets the system requirements;
  + **System Integration -** Integrate the trained model with the AIVQC system, including the user interface and user-adjustable controls.
  + **User Testing -** Gather feedback on the user interface from potential users, such as factory workers and university students
  + **Iteration and Improvement**
* **Slide 9 - Methods and Tools used**

AIVQC is powered by following tools:

* + Roboflow
  + Darknet
  + Opencv
  + Tensorflow
* **Slide 10 - Demonstration**
* **Slide 11 - Development (Reflection)**
  + Experimentation with Different methods of Object detection and image classification
  + Linux Implementation
* **Slide 12 - Future development**
  + Adding new defect classes by transfer learning
  + Linux OS implementation
  + User Interface improvements
    - Hiding console window
    - Settings screen
      * Configuring custom dependencies location
      * Selecting different model
      * Controlling detection settings
      * Limits for detection thresholds for users with higher access levels – preventing operators from completely overriding detection
  + Support for production line sensors
  + Installation Package
* **Slide 13 – Conclusion**

**In summary, the AI Visual Quality Controller project represents a significant advancement in utilizing machine vision and deep learning for industrial quality control. By accurately identifying defects in medical dressings, it enhances the quality assurance process and reduces the risk of defective products reaching consumers. While challenges such as power consumption and dataset quality exist, they can be effectively addressed through proper management. Future enhancements include Linux implementation, improved customization, integration with production line sensors, and a learning mode for continuous improvement. Although further real-world testing is needed, this project shows great potential in driving quality, efficiency, and innovation in the manufacturing sector.**

* **Questions?**

# References

|  |  |
| --- | --- |
| [1] | Optima, "Manufacturing and Packaging Machine OPTIMA MDC150," Optima, 2023. [Online]. Available: https://www.optima-packaging.com/en/machine-solutions/manufacturing-and-packaging-machine-optima-mdc150. [Accessed 10 05 2023]. |
| [2] | NAS, "In-Line Multi-Purpose Line-Scan Vision System," Northfield Automation Systems, [Online]. Available: https://northfieldautomation.com/solutions/systems/in-line-vision/. [Accessed 17 05 2023]. |
| [3] | UFP MedTech, "Shaping solutions that rise above the ordinary," UFP MedTech, [Online]. Available: https://ufpmedtech.com/capabilities/. [Accessed 17 05 2023]. |
| [4] | E. Goffin, "Bringing AI to Visual Inspection," Quality Magazine, 01 01 2022. [Online]. Available: https://www.qualitymag.com/articles/96809-bringing-ai-to-visual-inspection. [Accessed 26 03 2023]. |
| [5] | CutSheetLabels, "CutSheetLabels," CutSheetLabels, 28 02 2019. [Online]. Available: https://cutsheetlabels.com/blog/quality-control-checks-label-production/. [Accessed 17 05 2023]. |

# Appendix

A screenshot of a computer

Description automatically generated with medium confidence

**Tuning Hyperparameters for YOLOv4**

**Batch size** refers to the number of images processed in a training iteration. Larger batch size can enhance the stability and generalisation of the model but may require increased memory and computation resources. Conversely, a smaller batch size may necessitate less memory and computation power but may reduce model stability and generalisation capabilities.

**Subdivisions** denote the number of mini-batches created from a single batch. Employing smaller subdivisions can minimise memory usage and accelerate training; however, this may also increase the variance and noise of gradient updates, adversely affecting model performance.

**Learning rate** is the step size utilised in updating the model parameters based on the gradient descent algorithm. A larger learning rate can expedite the model’s convergence, but it also risks overshooting the optimal solution or diverging. On the other hand, a lower learning rate can provide a more accurate solution, albeit with longer training times or the possibility of getting trapped in local minima.

**Momentum** is a parameter that determines the extent to which the previous gradient update influences the current update. A higher momentum value can aid in overcoming local minima and smooth out gradient fluctuations but may also render the model less receptive to new information. Conversely, smaller momentum values can make the model more sensitive to new data but may also increase noise and oscillations in gradient updates.

**Decay** is a parameter that controls the rate at which the learning rate decreases over time. A larger decay can help prevent overfitting and improve generalisation, but it can also impede convergence or cause underfitting. In contrast, a smaller decay can maintain a high learning rate and expedite training, but it may also risk overfitting or divergence.

**Maximum number of iterations** performed during the training process is called max batches. A larger max batch size can improve the model’s performance by exposing it to more data; however, it can also prolong training time and increase costs. A smaller max batch size can reduce training time and costs, but it may also restrict the data used for training, potentially affecting model performance.

**Steps** are the iterations at which the learning rate is multiplied by a factor (scale), implementing a step decay schedule that reduces the learning rate by a fixed amount at specified intervals. This strategy helps balance exploration and exploitation during the training process. Scales determine how much the learning rate drops at each step decay interval. A larger scale can induce a more drastic change in the learning rate, which may help escape local minima or plateaus but also risks missing the optimal solution or diverging. A smaller scale can produce a more gradual change in the learning rate, promoting fine-tuning and smooth convergence, albeit with potentially longer training times or the risk of getting stuck in local minima or plateaus.

**Performance Evaluation**

Precision measures the proportion of true positive predictions among all the positive predictions made by the model. It is a useful metric for assessing the performance of the model when the cost of false positives is high. However, precision alone may not provide a complete picture of the model’s performance, as it does not take into account false negatives.

Recall, also known as sensitivity or the true positive rate, is the proportion of true positive predictions among all the actual positive instances in the dataset. It is a valuable metric when the cost of false negatives is high. Like precision, recall alone may not provide a comprehensive view of the model’s performance, as it does not consider false positives.

The F1 score is the harmonic mean of precision and recall, offering a balance between these two metrics. It is particularly useful when dealing with imbalanced datasets, as it takes both false positives and false negatives into account, providing a more holistic view of the model’s performance.

Mean Average Precision (mAP) is a metric commonly used to evaluate object detection models, such as YOLOv4-tiny, taking into account both the precision and recall across all object classes and various intersection over union (IoU) thresholds. Higher mAP values indicate the better overall performance of the model in detecting objects.

A picture containing text, screenshot, plot, line

Description automatically generated