A red shield with white letters on it

Description automatically generated

**PDE4433 – Machine Learning for Robotics**

**Coursework 2 (CW2)**

**Topic: Nut and Bolt Detection using Machine Learning**

Report Submitted by:

|  |  |
| --- | --- |
| Name: Vignesh Lakshmanasamy | MISIS No: M01026685 |
| Email: vl223@live.mdx.ac.uk | Course: M.Sc., Robotics |

Module taught by:

|  |
| --- |
| Professor Dr. Maha Saadeh |

**Table of Content**

1. **Introduction**
   1. **Problem statement**
   2. **Relevance to Robotics**
   3. **Objective**
2. **Dataset Description**
   1. **Dataset collection**
   2. **Data Samples**
   3. **Preprocessing**
3. **Machine learning model**
   1. **Using CNN model**
   2. **Using Pretrained model -Mobilenet2**
   3. **Model Training Process**
4. **Model Evaluations**
   1. **Model Accuracy**
   2. **Model Testing Scenario**
5. **Limitation**
6. **Conclusion**
7. **Future Enhancements**
8. **Link to Demonstration Video and Code**
9. **Reference**

**Abstract**

In industrial automation, the accurate identification and classification of components such as nuts and bolts is essential for ensuring product quality and efficiency. This project presents a machine learning-based approach to detect and classify nuts and bolts using image processing techniques and convolutional neural networks (CNNs). By leveraging a dataset of labelled images, the system is trained to distinguish between different types of fasteners with high accuracy. The developed model demonstrates the potential to enhance automated inspection processes in manufacturing lines, reducing human error and increasing throughput.

1. **Introduction**

The rapid growth of Industry 4.0 has increased the demand for intelligent systems capable of automating routine inspection tasks in manufacturing environments. Fasteners, such as nuts and bolts, are fundamental components used in assembling machinery and structures. Traditional manual inspection is time-consuming and prone to human error. This project explores the application of machine learning to automate the detection and classification of nuts and bolts, aiming to improve accuracy and efficiency in quality control processes.

* 1. **Problem statement**

Manual identification of nuts and bolts is inefficient and error-prone, especially in high-volume manufacturing environments. An automated, reliable detection system is required to:

* Differentiate between nuts and bolts.
* Classify them accurately under varying lighting and positional conditions.
* Minimize inspection time and human error.
  1. **Relevance to Robotics**

Object identification and classification are fundamental requirements in robotic automation systems. Robots in assembly lines, especially pick-and-place robots, depend on accurate object recognition for tasks such as sorting, inventory management, and quality inspection. Automating the identification of nuts and bolts can increase production efficiency, reduce human error, and ensure consistent quality control.

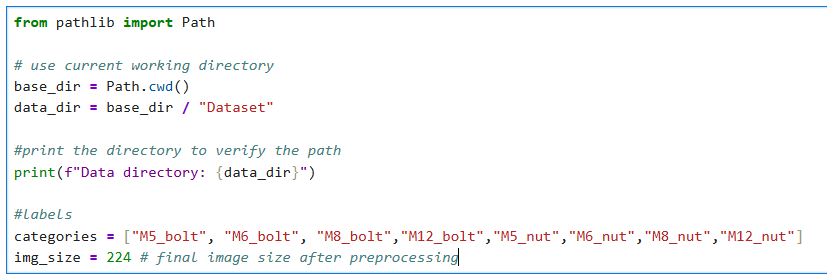
* 1. **Objective**

The main objectives of this project are:

1. Develop a machine learning model capable of detecting nuts and bolts.
2. Train the model using a labelled dataset of images.
3. Evaluate the performance of the model.
4. Deploy the model for real-time or batch image inspection scenarios.
5. **Dataset Description**

The dataset used for this coursework has been created by me specifically for the purpose of this project. To ensure consistency and minimize external variables, I prepared a custom stand where both the camera and the light source were mounted with fixed constraints. This setup helped maintain uniform lighting conditions throughout the data collection process. All images of bolts and nuts were captured against a white background, with the objects placed in various positions and angles to introduce variability and enrich the dataset for better model training.

* 1. **Dataset collection**
* **Camera Used:** Trands USB Manual Focus Webcam 1080P
* **Lighting:** Mobile flashlight used as light source
* **Background:** Maintained white background for clarity
* **Resolution:** Images captured at 640 x 480 pixels
* **Dataset Labels:** Each image of bolts and nuts was categorized and stored in separate folders according to its class, with the folder names serving as the labels for machine learning.

****

* **Stand size:** 235 (W) x 305 (L) x 240 (H) mm
* **Camera to ground distance:** 200 mm
* **Number of Samples:** 264images (8 categories x 33 images/categories)

|  |
| --- |
| **Camera set up for the data collection and real time testing** |
|  |

* 1. **Data samples:**

**Bolt image samples**

|  |  |
| --- | --- |
| **M5 Bolt** |  |
| **M6 Bolt** |  |
| **M8 Bolt** |  |
| **M12 Bolt** |  |

**Nut image samples**

|  |  |
| --- | --- |
| **M5 nut** |  |
| **M6 nut** |  |
| **M8 nut** |  |
| **M12 nut** |  |

* 1. **Preprocessing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Step No.** | **Preprocessing** | **Bolt** | **Nut** |
|  | Original image captured at 640 × 480 pixels resolution. |  |  |
|  | Converted to grayscale. |  |  |
|  | Applied Gaussian blur to reduce noise and smooth edges. |  |  |
|  | Performed thresholding to separate the bolt and nut from the background. |  |  |
|  | Applied Canny edge detection on the binary image to detect object edges. |  |  |
|  | Used dilation to enhance the detected edges. |  |  |
|  | Applied erosion to refine edges and eliminate discontinuities. |  |  |
|  | Contours extracted from the processed image. |  |  |
|  | Objects were cropped along the contours and placed on the centre of 224 × 224 pixel white background (corresponding to 59 × 59 mm). |  |  |

1. **Machine learning model**

In this project, two types of models were implemented and compared:

* 1. **Custom CNN model**

A custom CNN was developed from scratch to classify images of nuts and bolts. The architecture consists of:

* **Input Layer:** Accepts images of shape *(224, 224, 3)*.
* **Convolutional Layers:**
* First Conv2D layer with 32 filters of size (3×3), ReLU activation, and 'same' padding.
* Second Conv2D layer with 64 filters of size (3×3), ReLU activation, and 'same' padding.
* **Max Pooling Layers:** Applied after each convolutional block to down sample spatial dimensions.
* **Flatten Layer:** Converts 2D feature maps to a 1D feature vector.
* **Fully Connected Layer:** Dense layer with 256 neurons and ReLU activation.
* **Dropout Layer:** 50% dropout to prevent overfitting.
* **Output Layer:** Dense layer with 8 neurons (for 8 classes), using softmax activation for multi-class classification.

|  |
| --- |
| **Screenshot of the program used in the coursework** |
|  |

* 1. **Pretrained model -Mobilenet2**

MobileNetV2, a lightweight and efficient convolutional neural network, was utilized through transfer learning as a fixed feature extractor. The base model, pre-trained on ImageNet, was frozen to retain learned features, and custom fully connected layers were added for nut and bolt classification.

**Model Architecture:**

* Base Model: MobileNetV2 (pre-trained on ImageNet, frozen weights)
* Flatten Layer: Converts extracted features into a 1D vector
* Dense Layer: 128 units with ReLU activation
* Dropout Layer: 50% dropout to reduce overfitting
* Output Layer: Softmax activation for multi-class classification

|  |
| --- |
| **Screenshot of the program used in the coursework** |
|  |

**Advantages of using MobileNetV2:**

* Faster training time
* Higher accuracy with fewer computational resources
* Effective performance on small datasets
  1. **Model Training Process**
* **Data split:** 80% training, 20% validation
* **Loss function:** Categorical Crossentropy
* **Optimizer:** Adam
* **Metrics:** Accuracy
* **Training epochs:** Maximum of 100, but early stopping is applied (patience of 5 epochs) to prevent overfitting, typically resulting in 10–20 epochs based on validation performance.
* Both models were trained using TensorFlow and Keras, leveraging GPU acceleration for faster computation.

|  |
| --- |
| **Screenshot of model training program used in the coursework** |
|  |
|  |

1. **Model Evaluations**
   1. **Model Accuracy**

|  |  |
| --- | --- |
| **Model Accuracy for Custom CNN model** | |
|  | |
|  | |
| **Model Accuracy for Pre-trained model -Mobilenetv2** | |
|  | |
| **Performance Graph** | | |
| **Custom CNN** | | **MobileNetV2 Model** |
| **A graph of a line  AI-generated content may be incorrect.** | |  |

The accuracy of both the custom-designed CNN model and the pretrained MobileNetV2 model is evaluated. Both models demonstrated excellent performance on the testing dataset, indicating effective feature extraction and robust classification capabilities.

Despite being a manually built architecture, the custom CNN achieved an impressive **94% accuracy**, closely competing with the MobileNetV2's **98% accuracy**. This shows that the dataset quality and preprocessing steps played a crucial role in achieving high performance.

The minimal difference in accuracy highlights the effectiveness of both approaches, with MobileNetV2 slightly leading due to its advanced transfer learning architecture and optimization for real-world scenarios.

* 1. **Model Testing Scenario**

The testing phase was conducted to assess model generalization and robustness in diverse real-world scenarios. Images outside the training dataset and live stream inputs were used, featuring variations such as:

* Lighting conditions: Images were tested under artificial light and partial shadows.
* Object orientation: Fasteners were placed at different angles and rotations.

Both the custom CNN and MobileNetV2 models consistently delivered accurate classifications in these testing scenarios. Despite environmental changes, the models maintained high confidence levels in predictions, accurately differentiating between various sizes and types of nuts and bolts.

This confirms the readiness of the models for practical application in real-world automated systems, such as robotic sorting and quality inspection processes on manufacturing lines.

|  |
| --- |
| **Test result of Custom CNN with image which is not used in Training** |
|  |

|  |
| --- |
| **Test result of Custom CNN with artificial ceiling light and partial shadow** |
|  |

|  |
| --- |
| **Test result of Custom CNN in Live detection using webcam** |
|  |

1. **Limitation**

Despite the high accuracy achieved by both models, certain limitations were identified during the project,

* **Overfitting in Custom CNN:**

The custom-built CNN, while performing well on the training data, showed signs of overfitting. The model's high accuracy on the training set did not perfectly translate to unseen data, indicating its sensitivity to the training environment.

* **Lighting Conditions:**

The models performed best under controlled lighting conditions, as used during data collection. In environments with inconsistent or poor lighting, such as excessive shadows or glare, the accuracy can potentially decrease.

* **Limited Dataset Size:**

Although data augmentation was applied, the total dataset remained relatively small. A larger and more diverse dataset could improve model robustness.

1. **Conclusion**

This project successfully demonstrates the application of machine learning techniques for the automatic detection and classification of nuts and bolts in manufacturing environments. Both the custom CNN and MobileNetV2 models achieved high accuracy, with the custom CNN reaching 94% and the MobileNetV2 achieving 98%.

The close performance between both models illustrates that a carefully designed architecture can perform competitively with advanced pretrained models, especially with well-prepared datasets and preprocessing techniques. The MobileNetV2, with its transfer learning advantage, offered faster convergence and slightly better generalization, making it more suitable for practical deployment.

Overall, this solution has the potential to enhance industrial inspection processes by reducing reliance on manual inspection, minimizing errors, and increasing throughput in production lines.

1. **Future Enhancements**

Following enhancements can make the model more robust and industrially ready,

* By expanding dataset diversity
* Improving lighting robustness
* Integrating with robotic systems
* Training with complex backgrounds

1. **Link to Demonstration Video and Code**

Demonstration video link - <https://youtu.be/l8Y6bqxDLcg>

Github Code link - <https://github.com/Vignesh-Lakshmanasamy-mdx/CW2_ML_PDE4433.git>

1. **Reference**

* TensorFlow , MobileNetv2- [tf.keras.applications.MobileNetV2  |  TensorFlow v2.16.1](https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV2)
* Conv2d - [Conv2D layer](https://keras.io/api/layers/convolution_layers/convolution2d/)
* Maxpooling - [MaxPooling2D layer](https://keras.io/api/layers/pooling_layers/max_pooling2d/)