2024 International Conference on Intelligent and Innovative Computing Applications (ICONIC)
7th-9th November 2024, Pearle Beach Resort & Spa, Flic en Flac, Mauritius
https://doi.org/10.59200/ICONIC.2024.015

CNN Defect Detection Using Image Classification Falls Short in Assembly

Janré Smit, Leon E. Burger, Corne S. L. Schutte
Stellenbosch University
Stellenbosch, South Africa
janresmit21@gmail.com, eldonburger@sun.ac.za, corne@sun.ac.za

Abstract
The use of Convolutional Neural Networks (CNNs) for image classification is well-established in defect detection within manufacturing environments. However, their application in assembly processes remains underexplored. This study investigates the effectiveness of using relatively smaller CNNs for detecting assembly defects by testing multiple existing pre-trained CNNs and a custom CNN on a dataset of model train seat assembly images. The dataset includes common assembly defects such as missing, rotated, and swapped parts. Despite implementing various data augmentation and image processing techniques, such as cropping, normalization, and image transforms, and making several modifications to the models, the existing techniques struggled to accurately predict defects while providing the correct rationale for their predictions. The Grad-CAM analysis revealed that the models often focused on irrelevant features, highlighting the challenges of defect detection in complex assembly environments. These findings indicate the need for more robust machine learning approaches capable of handling high levels of noise and variations typical of real-world assembly conditions. This study underscores the limitations of current CNN-based defect detection methods in uncontrolled assembly settings and the necessity for further research to develop more reliable solutions.

Keywords: defect detection, convolutional neural networks, assembly, image classification

1. Introduction

Quality control is a crucial part of the manufacturing sector, as defects can lead to inferior products, customer dissatisfaction, and damage to a company's reputation. One of the primary methods used to perform quality control in the manufacturing sector is visual inspection. Visual inspection is a non-destructive quality control method which involves the visual assessment of a product against specific criteria such as surface finish, dimensions, and colour consistency.

Traditionally, visual inspection is performed manually, requiring human inspectors to examine products for defects or irregularities using the naked eye or sometimes with the aid of magnifying tools or other equipment (See, 2012). Manual visual quality inspection is however expensive, subjective, prone to mistakes, and can cause occupational health issues such as eye strain and labour fatigue, leading to financial losses and wasted man-hours (Boby et al., 2011). Thus, the development and deployment of automated visual defect detection systems are highly advantageous for improving efficiency, reducing costs, enhancing accuracy, and promoting worker safety in manufacturing environments.

The development of artificial intelligence algorithms and the emergence of affordable devices capable of running these algorithms offer the opportunity to deploy automated visual inspection in the manufacturing environment at scale (Würschinger et al., 2020). These systems have however mostly been developed for individual parts or straightforward use cases such as surface inspection. In surface inspection, the environment can typically be controlled to avoid or minimise any image variations not attributed to defects.

In contrast to surface inspection, when visual inspection is performed on assembly lines multiple interconnected parts must be inspected on a large scale which often means that images captured contain non-defect related variations such as background noise. Such noise can introduce unwanted features into the dataset,

leading to false positives in defect detection. Assemblies flagged for defects need reworking, increasing overall manufacturing time. Conversely, noise in the dataset can also cause defects to be missed entirely, leading to problems down the assembly line or undetected defective final products. Therefore, implementing a reliable and trustworthy machine learning approach to identify defects in complex assembly environments remains a high priority.

Though previous studies on defect detection on more simple parts have shown promising results, there remains a gap for them to be tested in the assembly environment. This study aims to establish whether existing machine learning technologies used to develop automated visual defect detection systems for parts or simple tasks can be used for assembly defect detection. Specifically, the research seeks to determine if these algorithms can distinguish between defect and non-defect assemblies in a complex environment with high levels of noise and variations.

Towards this extent, a dataset of ensembled model trains is collected which includes three types of common assembly defects: missing parts, wrongly orientated parts and swapped parts. To establish whether machine learning algorithms learn to truly distinguish between defect and non-defect trains, multiple different convolutional neural networks is trained using a range of commonly used practises such as data augmentation and pre-training. Even though extensive approaches are evaluated, none of the approaches differentiate between defect and non-defect trains entirely based on the defects. Our findings highlight the need for further research into the development of improved algorithms.

In this study, we collected a dataset of top-down photos of a model train using a webcam. The seating arrangement of the model train was altered to simulate various assembly defects. A model train was chosen for its simplicity and cost effectiveness in generating defects. This assembly example is representative of real-world scenarios, as it contains many false signals and considers various types of defects simultaneously. Due to the wide variety of assembly defects that can occur, not all types were tested in this study, but the most common types of assembly defects were used as examples in the dataset, including missing parts, wrongly orientated parts (rotated 90° and 180°), and swapped parts. Subsequently this dataset was used to train a custom CNN for defect detection. Various strategies, including data augmentation, machine learning model adjustments, and image processing, were employed in attempt to refine the model's predictions, ensuring it focuses on the appropriate features of each train. Additionally, pre-trained models, including ResNet-50 (Koonce, 2021), EfficientNetV2_Small_Weights (Tan & Le, 2021), and DenseNet-161 (Dhillon & Verma, 2022) were tested and compared to the performance of our custom CNN model. We wanted to see whether our custom CNN model could outperform these pre-trained models and better distinguish between false signals and true defects.

This paper proceeds as follows. In the next section, the related works are covered. Section 3 describes the methodology that was followed while section 4 describes the experimental setup. The results and discussion sections are combined to form one section 4 due to the steps of the study being conducted iteratively. Section 5 concludes the paper.

2. Related Works

Convolutional Neural Networks (CNNs) have been widely used for defect detection on products. They play a crucial role in automated feature extraction by identifying boundaries, learning hierarchical features, and enhancing accuracy. CNNs detect edges within images, essential for object detection, segmentation, and feature extraction (Patel, 2024). Deep CNN architectures (e.g. VGG (Varshney, 2020), ResNet (He et al., 2016), and Inception (Alom et al., 2021)) automatically learn spatially relevant features from large datasets such as ImageNet (Deng et al., 2009). Pre-trained CNNs facilitate transfer learning, remaining effective even when applied to different domains(Nazare et al., 2018). Feature extraction reduces image data dimensionality, preserving essential information and improving computational efficiency (Kumar & Bhatia, 2014). Additionally, CNNs enhance image processing tasks by isolating crucial features and reducing noise (Kumar & Bhatia, 2014; Patel, 2024).

While CNNs are effective, they can sometimes learn false signals through a phenomenon known as Shortcut Learning (Geirhos et al., 2020), which may introduce more problems in assembly defect

detection due to the complexity of the environment. For instance, a model might rely on background clues for classification if objects are consistently placed against specific backgrounds, rather than focusing on the objects themselves. Similarly, if particular colours or textures are consistently present in the training images of a certain class, the model might incorrectly associate these superficial features with that class. Additionally, metadata artifacts such as image size or compression artifacts, if correlated with the labels, can also serve as shortcuts for the model. Mitigation strategies include data augmentation, adversarial training, and better data collection, all of which will be addressed in subsequent sections.

Cumbajin et al. (2023) conducted a systematic review on Deep Learning with CNNs applied to surface defect detection. They reviewed 62 papers and determined that the most common approach that was used in these papers was image classification. Of these studies, 32% used unmodified CNNs, while the rest created custom CNNs based on other networks. Data augmentation and transfer learning were also common techniques used in these studies. Notably, 66% of the datasets were private and most studies employed industrial cameras requiring specific lighting conditions and specialized lenses (Cumbajin et al., 2023). These studies also suggest that image classification is the most common approach for defect detection, hence its effectiveness in assembly environments being explored in our study. Additionally, data augmentation and transfer learning are also considered, since many papers make use of these techniques. All the papers reviewed by Cumbajin et al. (2023) focus on surface defect detection rather than assembly defect detection, and among the few studies on assembly defect detection using image classification, such as those by Burresi et al. (2021), Arjun & Mirnalinee (2016), Guo et al. (2020), and Weiss et al. (2024), most either consider only one type of defect or do not address significant false signals that could lead to false positives.

Other studies by Frustaci et al. (2020), Birari et al. (2023), Pierleoni et al. (2020), Rusli & Luscher (2018), and Mazzetto et al. (2019) focused on experiments conducted in specifically defined and controlled environments, with minimal noise or variation and meticulously controlled lighting conditions. The extensive systematic literature review by Panzer & Bender (2022) found that all the studies they examined were conducted under similarly controlled conditions with little to no variation or noise. They emphasized that assembly tasks benefit from confined and segregated environments, which limit hurdles and mitigate the risks of external factors influencing operations. Herakovic (2010) suggests that backlighting can reduce the impact of lighting conditions on machine vision tasks. However, this approach is not always viable due to the diverse conditions of assembly tasks, as in our study, where the objects of focus do not protrude from the main assembly.

A gap in the literature was identified regarding the application of CNNs in assembly environments with high levels of noise and variations. Previous studies typically focused on detecting defects on surfaces of similar parts within the assembly in controlled environments, where false signals do not influence the models' decision-making. These controlled settings are considered too straightforward for the application we aim to test. Our study builds on this previous work by investigating the implementation of existing image classification methods in uncontrolled assembly environments, where external factors cannot always be mitigated, and the data may contain false signals.

3. Methodology

In practice, defect detection in assembly should ideally not significantly disrupt the production line. Thus, the goal is to explore the possibility of developing a machine learning model that can later be implemented on a small, fast, stand-alone device capable of operating on an Internet-of-Things (IoT) infrastructure, which often leads to it being a more cost-effective solution. For this reason, a standard webcam (shown in Figure 2) is used to capture images for the dataset, and a custom CNN is developed to keep the model as small as possible.

The proposed system involves several sequential steps: data acquisition/image capturing, data augmentation and machine learning model adjustments, image pre-processing, Grad-CAM accuracy measurement, and defect detection. These processes are summarised in Figure 1 and discussed in more detail in the subsequent sections.

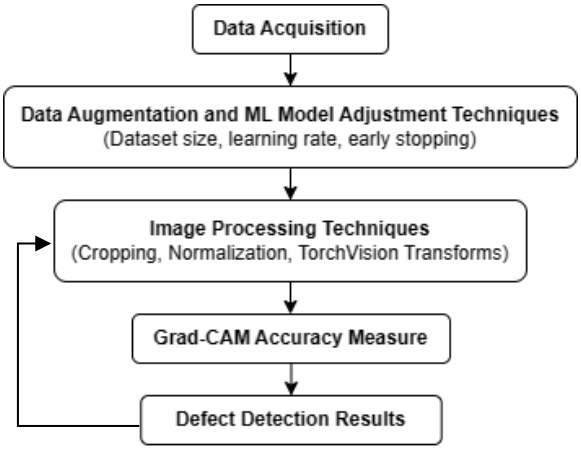

Figure 1. Iterative process flow diagram of proposed system

Data Augmentation

The dataset of images was captured using the webcam setup shown in Figure 2. This dataset is made available to the academic community. The images were captured under varying lighting conditions, predominantly using natural ambient light as the primary light source. This approach introduced some noise and blurriness into some of the images, imitating the real-world environment.

Figure 2. Camera setup

The model train, as seen from above, has a variety of different seats in a specific order. Figure 3 shows examples of the photos that are included in the dataset. The seats were rearranged to simulate defects on the model trains in three different classes: missing seats, rotated seats (rotated 90° and 180°), and swapped seats. Originally, we captured a dataset containing 454 images with the following amounts in each class: 40 photos of "Correct", 210 photos of "Missing", 44 photos of "Rotation", and 160 photos of "Swap". Given the imbalance in the dataset where the "Correct" and "Rotation" classes had fewer instances compared to the other classes, a Python script was used to address this by augmenting these classes. The script generated additional variations of each photo within these classes, incorporating changes in blurriness, sharpness, contrast, and brightness.

While it is understood that defects are naturally anomalous and typically represent a smaller fraction of real-world data, this intentional imbalance was necessary for the training phase. By exposing the model to a larger number of defective instances, we aimed to enhance its ability to detect and learn from various defect scenarios, thereby improving its overall performance and robustness in identifying defects. Following this augmentation process, the dataset was expanded to include 240 "Correct" images and 264 "Rotation" images, mitigating the initial imbalance and ensuring that all classes contain a relatively equal number of photos. The subsequent dataset contained 866 photos.

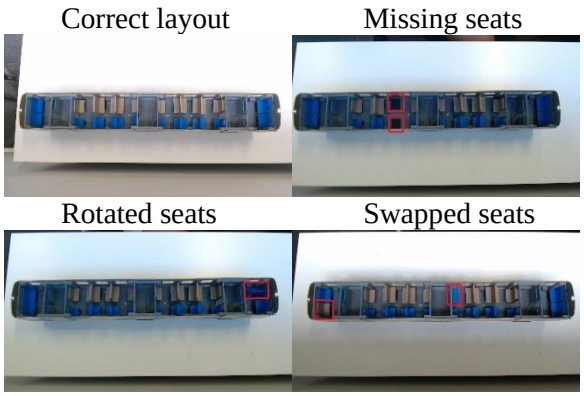

Figure 3. Example of images from the dataset (bounding boxes added afterwards for visual reference)

The dataset was divided into subsets according to the following percentages:

Training : Validation : Testing
70% : 15% : 15%

Custom CNN

The model architecture of the custom CNN is shown in Figure 4. The network was defined and subsequently called in a feedforward pass.

```
class Net(nn.Module):
  def __init__(self, use_dropout):
    super().__init__()
    self.use_dropout = use_dropout
    self.conv1 = nn.Conv2d(3, 16,
                           kernel_size=3,
                           padding=1)
    self.act1 = nn.Tanh()
    self.pool1 = nn.MaxPool2d(5)
    self.conv2 = nn.Conv2d(16, 8,
                           kernel_size=3,
                           padding=1)
    self.act2 = nn.Tanh()
    self.pool2 = nn.MaxPool2d(2)
    self.dropout = nn.Dropout(0.2)
    self.fc1 = nn.Linear(8 * 39 * 80, 32)
    self.act3 = nn.Tanh()
    self.fc2 = nn.Linear(32, 4)

def forward(self, x):
    out = self.pool1(self.act1(self.conv1(x)))
    out = self.pool2(self.act2(self.conv2(out)))
    batch_size = out.size(0)
    out = out.view(batch_size, -1)
    if self.use_dropout == True:
      out = self.dropout(out)
    out = self.act3(self.fc1(out))
    out = self.fc2(out)
    return out
```

Figure 4. CNN model architecture

The model has 800492 parameters, and the maximum number of epochs was 100. The training dataset was then loaded into the model and the accuracy of the model was evaluated resulting in 100% accuracy on the training dataset, 100% on the validation dataset, and 100% on the testing dataset. This indicated that the model has correctly classified all instances in each of these datasets. While this might initially seem ideal, it raised concern due to its following potential implications:

- Overfitting: There is a high chance that the model has memorized the training data rather than generalizing from it, especially if the training dataset is not very large or complex.
- Data leakage: There might be an overlap between the training, validation, and testing data, or the validation and testing data might not be truly representative of new, unseen data.
- Overly simple dataset: The validation and testing datasets might be too simple or similar to the training data.

Additional investigation was required to confirm whether the model accurately identified image features for predictions, as elaborated upon in Section 4. The original dataset was correctly split into subsets, eliminating data leakage as a potential issue. The Grad-CAM package from (Gildenblat & Contributors, 2021) was employed on the testing dataset's results to generate a heat map highlighting significant regions in input images influencing the network's decisions. Figure 5 illustrates that the model predominantly classified images based on their borders, which should not be a relevant feature in the model's decision making. This rules out the concerns, indicating that the model does not overfit, nor was the dataset too simple. Instead, it suggests that the CNN learned false signals, resulting in the task being trivially solved.

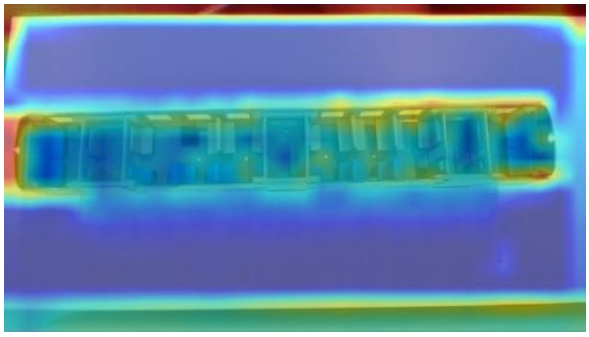

Figure 5. Grad-CAM image showing what the model focuses on

Pre-Trained CNNs

To test whether the incorrect focus of the model on irrelevant features was due to the architecture of the custom CNN, available pre-trained CNNs were also tested.

Table 1 shows the models that were used with the number of parameters. These models were chosen as they had relatively small numbers of parameters and are thus considered as suitable for implementation on a small-scale device with less computational power.

Table 1. Comparison of pre-trained models

Model	**Parameters**
Custom CNN	800 492
ResNet-50 (Koonce, 2021)	23516228
EfficientNetV2_Small_Weights (Tan & Le, 2021)	20182612
DenseNet-161 (Dhillon & Verma, 2022)	26480836

These models delivered the same results as the custom CNN with accuracies of 100%. Grad-CAM revealed that these models predominantly focused on the image background rather than the relevant seats, despite being trained with optimal

parameters validated for performance. This proved that they were not able to better distinguish between false signals and true defects. Therefore, using pre-trained models does not solve the problem, and the custom CNN was further iterated over to determine whether image processing techniques, like cropping, normalisation, and transforms will help to eliminate the bias that is formed due to irrelevant features. This is discussed in further detail in Section 4.

Grad-CAM

Various Grad-CAM methods (Gildenblat & Contributors, 2021) were used to analyse the attention patterns of the CNNs and to see if any of these techniques reveal information that might lead to a different conclusion. These methods include:

- **Standard Grad-CAM:** Uses gradients to generate visual explanations for model predictions.
- **Grad-CAM++:** An improved version of Grad-CAM with better localization.
- **XGrad-CAM:** Scales the gradients by the normalized activations.
- **AblationCAM:** Zeroes out activations and measures how the output changes. It helps identify crucial areas by removing parts of the image.
- **ScoreCAM:** Instead of relying on gradients, it alters the input image by scaling the activations of certain neurons and measures how the output drops.
- **EigenCAM:** Utilizes principal components for class activation maps.
- **LayerCAM:** Explores hierarchical class activation maps for better localization.

Surprisingly, all these techniques consistently revealed that the model focused on irrelevant areas rather than the expected features. Given this consistent outcome, the decision was made to streamline the approach and rely solely on standard Grad-CAM for future investigations. By doing so, simplicity is maintained without sacrificing accuracy, as all the methods yielded similar results.

Figure 6 shows an example of the Grad-CAM outputs compared to the original photos.

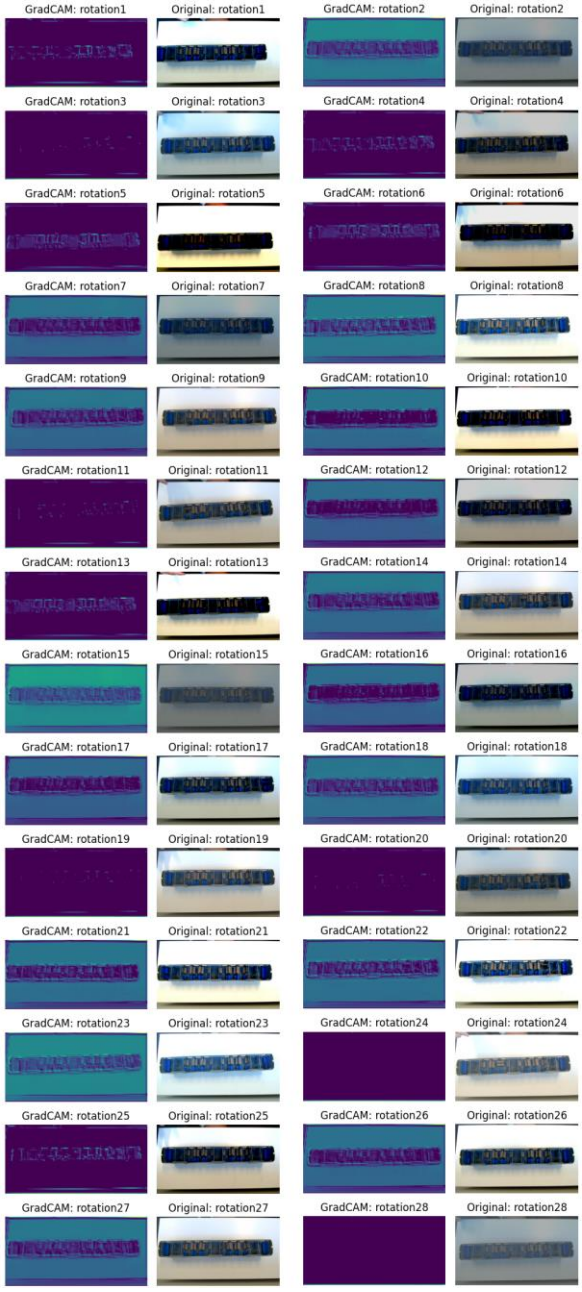

Figure 6. Example of Grad-CAM images

4. Results and Discussion

CNN Model Adjustments

Each training run of the CNN model highlighted an area of potential improvement. Initially, the models' performance was evaluated by their effectiveness in correctly classifying the images to their corresponding classes. The model's accuracy on the training, validation, and test datasets were calculated with each run and the ablations were set up to understand the contribution that each specific component and feature had to the model's performance, which is shown in Table 2.

Table 2. Ablations of CNN model adjustments

Training run #	Dataset size	Learning rate	Early stopping	Epoch duration (sec)	Training accuracy	Validation accuracy	Testing accuracy
1	454	1e-3		20	82%	82%	82%
2	866	1e-3		40	100%	100%	100%
3	866	2e-3		24	100%	100%	100%
4	866	2e-3	√	39	99%	98%	99%

In the first training run, the original dataset of 454 photos was used and the model struggle to correctly predict the classes. In the second run, the expanded dataset of 866 photos was utilised to address the imbalance discussed in section 3. The duration of the training run increased considerably and consequently, the learning rate was increased to 2e-3 in the next run to accelerate training and avoid the model becoming trapped in local minima without converging to an optimal solution. The training and validation losses did not decrease significantly after around 20 epochs, as shown in Figure 7.

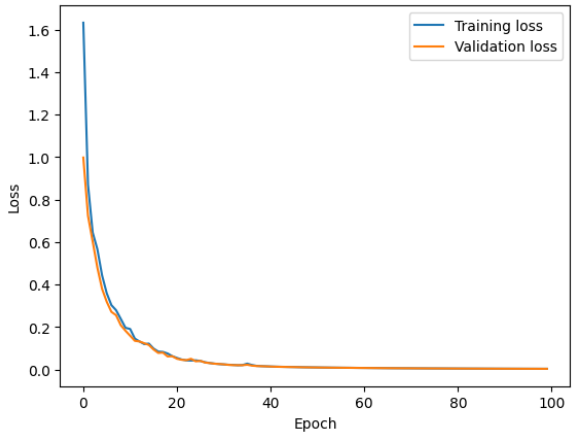

Figure 7. Training and validation loss of run 2

The flattening of the loss curves after approximately 20 epochs indicated that the model had effectively learned the patterns present in the training data. Therefore, early stopping was introduced to terminate training if either the training or the validation loss increased or if they fail to decrease by more than 0.005 across more than five consecutive epochs. Implementing a patience value of 5 epochs provided the model with sufficient leeway to move past local minima during training. After these adjustments were applied, Grad-CAM indicated that the model's decision-making was still largely influenced by the background of the images. Therefore, image processing techniques were applied.

Image Processing

A range of image processing techniques were applied to enhance the model's robustness against unwanted features. These techniques include cropping, normalisation, and some of TorchVision's transforms (TorchVision Maintainers & Contributors, 2016), which are shown in Table 3.

Table 3. Ablations of image processing techniques

#	Improvements	Epoch duration (sec)	Train Acc. (%)	Vali Vacc. (%)	Test Acc. (%)
4	Previous run	39	99	98	99
5	+ Cropping	39	97	96	95
6	+ Normalisation	33	97	96	95
7	+ Random Horizontal Flip Transform	45	95	96	93
8	+ Random Vertical Flip Transform	39	90	94	89
9	+ Random Rotation Transform	66	90	93	90
10	+ Random Perspective Transform	58	87	91	86

This performance metric evaluates how well the model's predictions align with the actual image labels. After investigating the Grad-CAM outputs, it became clear that these accuracies presented in Table 2 and Table 3 do not fully capture the essence of the model's performance as there is more complexity behind the scenes concerning what aspects of the images the model is focusing on for making these predictions.

Subsequently, the Grad-CAM outputs where further used to investigate whether the model focused on the regions in the images where the defects occurred. It became apparent that the model focused more on features external to the train, struggling to even simply focus on the train itself. Thus, before moving on to verifying whether the model could point out the correct seat

where the defect occurred, a more conservative accuracy measure was used that involves counting how often the Grad-CAM outputs show the model focusing on the train itself and not on the background. This focus accuracy metric is presented in Table 4 and visualised in Figure 8, and it will be used for further analysis throughout the study.

Table 4. Percentage of images correctly focused on in each class

Custom CNN's Focus Accuracy (%)						
#	Improvements	Correct	Missing	Rotation	Swap	Overall
1	Original dataset	0	0	0	0	0
2	+ Bigger dataset	1	7	25	52	19
3	+ New learning rate	19	0	36	17	19
4	+ Early stopping	22	0	29	31	21
5	+ Cropping	57	22	29	25	34
6	+ Normalisation	36	57	20	29	35
7	+ Random Horizontal Flip Transform	72	62	57	73	65
8	+ Random Vertical Flip Transform	7	50	38	31	31
9	+ Random Rotation Transform	74	43	49	40	53
10	+ Random Perspective Transform	25	25	30	19	25

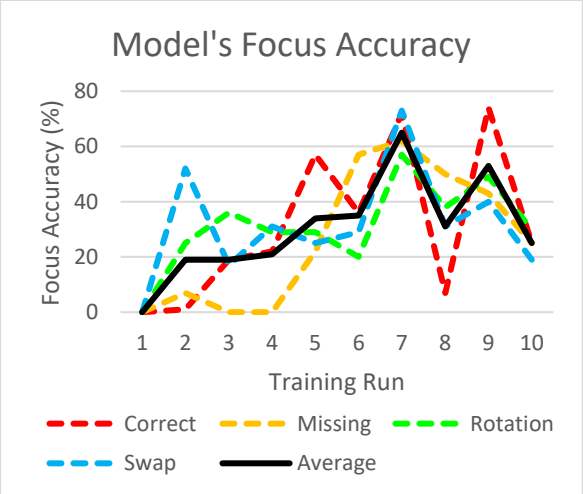

Figure 8. Visualisation of model's focus accuracy

Over the first four training runs, the focus accuracy hovers around 20%, indicating that although the model achieved nearly perfect accuracy in predicting the image classes (as shown in Table 2), it based these predictions on incorrectly learned features. In subsequent runs, image processing techniques were applied in attempt to aid the model in training on the correct features.

Cropping: The Grad-CAM images (as shown in Figure 6) revealed that the model primarily focused on the image background to make predictions. To mitigate this problem, a cropping transform was applied to the images, reducing the 1920 x 1080 resolution to 1600 x 780 by removing the border areas where the model tended to focus. This increased the CNN's focus accuracy significantly and tended to put more focus on the train itself (as shown in Figure 9). Despite this improvement, the background was still the focus in some cases.

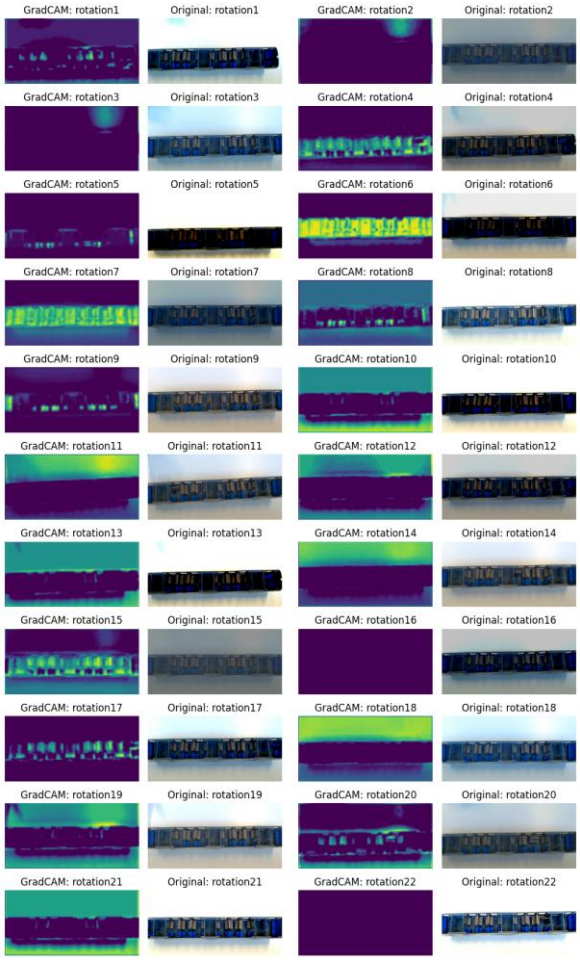

Figure 9. Grad-CAM output after cropping

The false signals learned upon may have been caused by the following factors that might have varied slightly between each class of photos:

- Lighting Changes: Differences in ambient light temperature, shadows, or bright spots due to photos being captured during a different time of day or due to passing clouds.
- Positioning Shifts: Minor adjustments in how the train was positioned under the camera due to rearranging the seats between shots.
- Photo Blur: Blurring in the photos caused by the webcam's auto focus settings.

Normalisation: With the added challenges posed by environmental conditions and camera settings in computer vision tasks, a Normalisation transform was applied to the images in attempt to address these issues. However, this did not improve the overall focus accuracy of the CNN.

Transforms: TorchVision's transforms (TorchVision Maintainers & Contributors, 2016) were applied cumulatively in attempt to mitigate the false signals that could have formed due to the train's position and/or varying brightness levels.

- Random Horizontal Flip: applied with a probability of 0.5 (approximately half of the images would be horizontally flipped). It aims to reduce the CCN's tendency to focus on variations in the background surrounding the train, which may differ to the right and the left.
- Random Vertical Flip: Similarly, this transformation aims to diminish the CNN's focus on the varying spaces at the top and the bottom of the train.
- Random Rotation: randomly rotates the image by up to 5°, aiming to further reduce the tendency to focus on the image border.
- Random Perspective: distorts the perspective of the images with the same aim. A distortion scale of 0.2 and a probability of 0.5 was used.

Despite applying various image processing techniques, the model still did not consistently focus on the train and was influenced by the background in its predictions. To address this, the train's outside borders were identified using a Jupyter Notebook developed by Roboflow & James (2023). This script processes the original images to locate the coordinates of the train's borders. These images with the located coordinates of the trains' borders were then put through another Python script that crops the images to the minimum and maximum height and width values. These final cropped images are then put through the same iterations of training runs, with consistent image processing techniques. This entire process is shown in Figure 10. However, even with these adjustments, the CNN occasionally still focused on the background of the image.

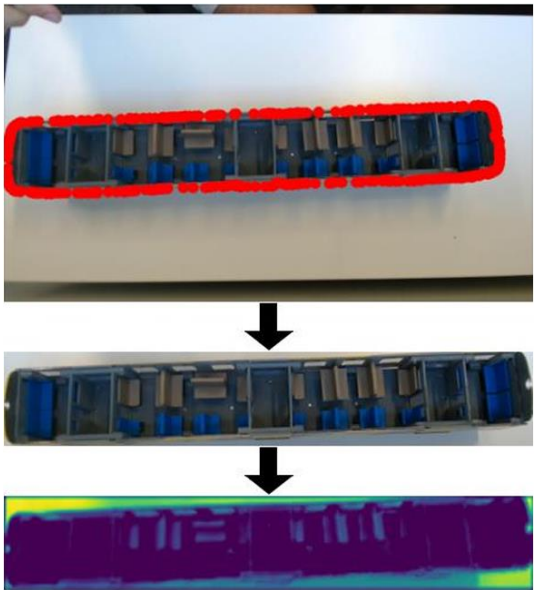

Figure 10. Outputs of cropping process

Rotated Cropping

Consequently, the goal shifted to eliminating the background entirely and rotating the trains to be horizontally aligned. This was achieved by sending the coordinates of the model train's border on each image through a Python script that aimed at optimizing the orientation of the train images based on the provided coordinates. It reads the coordinates and determines an optimal rotation angle to align the trains horizontally by using a loss function to minimize the average height of the coordinates, as the train would be perfectly horizontal at the smallest height. The images are rotated and once again cropped to the minimum and maximum height and width values. An example of a final rotated and cropped image is shown in Figure 11.

Figure 11. Example of final rotated crop image

These images were subsequently put through the same iterations of training runs. With the backgrounds completely removed or cropped out, the performance metric was adjusted to evaluate the CNN's ability to concentrate on the relevant defect. The model's accuracy is determined not only by its ability to predict the correct class but also by providing the correct rationale for that

prediction. This ensures that the model is both accurate in its classifications and in its identification of the key features leading to those classifications.

Given that correctly identifying the relevant defect is now the primary focus, the CNN's ability to classify the 'Correct' class is no longer evaluated. Instead, only the defect classes are considered, as they demonstrate whether the model can accurately classify the specific defects. This shift ensures that the model's performance is assessed based on its ability to identify and focus on the relevant defects.

When identifying missing seats, the CNN based its predictions on the presence of other seats rather than pinpointing the missing seat's location, as shown in Figure 12. If the measure had accounted for the total number of seats, Grad-CAM images might have shown the CNN's ability to detect a missing seat. However, since this measure was not used, the model's focus accuracy for 'Missing' defects is 0%.

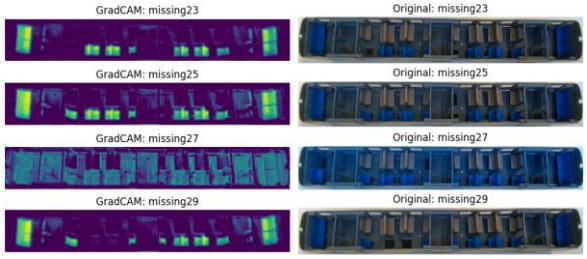

Figure 12. Grad-CAM example on 'Missing' class

For the 'Rotation' class, the CNN did not indicate the location of rotated seats (shown in Figure 13), giving a 'Rotation' class focus accuracy of 0%.

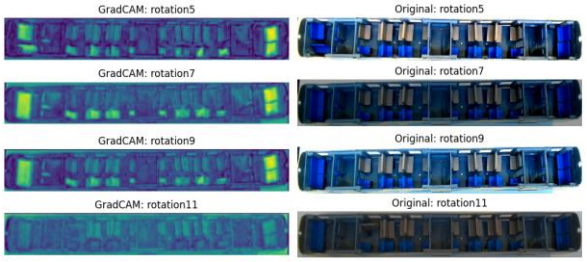

Figure 13. Grad-CAM example on 'Rotation' class

Interestingly, in predictions of the 'Swap' class, the CNN successfully identified swapped blue seats alongside others (as depicted in Figure 14) but did not do the same for swapped brown seats. This trend is evident across nearly all Grad-CAM outputs from the training runs on this dataset. The CNN's focus on blue seats may stem from challenges in distinguishing between the grey train base and brown seats, leading to a preference for more distinct colours. Despite this, the 'Swap' class also exhibited a focus accuracy of 0%.

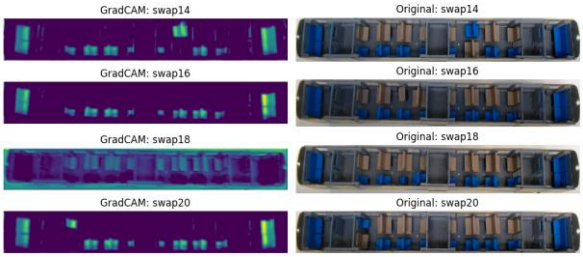

Figure 14. Grad-CAM example on 'Swap' class

Using this performance metric of looking at the specific seats, the CNN failed to provide the correct rationale for any of its predictions. This establishes the need for further research on the use of CNNs for image classification in assembly defect detection.

Assumptions and Limitations

In real-world scenarios, defects are rare and difficult to collect, unlike our dataset where defects are abundant to enhance the CNN's learning. This intentional imbalance helps detection but limits real-world transferability, as actual production environments have fewer defects, affecting model performance and generalization. We leverage this limitation to enhance our model’s robustness.

Another limitation is the use of smaller CNNs, chosen to develop a lightweight, stand-alone solution for IoT devices that minimally disrupts production. While this approach supports our goal, it may have limited the model's accuracy and robustness compared to larger networks.

Not all types of assembly defects were tested due to their wide variety. The dataset focused on common defects, such as missing parts, incorrectly oriented parts (rotated 90° and 180°), and swapped parts. Defects like gaps, lateral misalignment, and hanging offsets were not considered. Thus, while the findings may not cover all assembly defects encountered in real-world scenarios, they provide a reasonable representation of what can be expected.

Future research should improve model robustness and generalization using advanced data augmentation, semi-supervised learning, and

synthetic datasets. Balancing datasets to reflect defect rarity and evaluating trade-offs between model complexity and IoT deployment constraints are also crucial for practical applications.

5. Conclusions

This study aimed to determine whether CNNs used for automated visual defect detection in parts or simple tasks could be effectively applied to assembly defect detection based on image classification. To achieve this, a dataset of top-down images of model trains was collected, featuring three common types of assembly defects: missing parts, wrongly oriented parts, and swapped parts. The image backgrounds introduced significant noise, leading to false signals being learned by the model due to shortcut learning. Despite extensive experimentation with various CNNs and applying mitigation strategies, including data augmentation and image processing techniques, none of the models could entirely differentiate between defect and non-defect trains based solely on the defects. Grad-CAM analysis revealed that the models often focused on irrelevant features, such as image borders, rather than the actual defects. These findings highlight the need for improved algorithms that can accurately identify assembly defects without being misled by irrelevant patterns resulting from shortcut learning.

While a perfectly controlled production environment with minimal external influences is preferred, it may not be achievable or practical due to associated costs and the need for perfectionism. Therefore, the goal of future research should be to develop a CNN that is resilient against external factors and capable of mitigating unwanted influences. This approach would make the model suitable for various industrial applications without requiring extensive refinement of the production environment itself.

6. References

Alom, M. Z., Hasan, M., Yakopcic, C., Taha, T. M., & Asari, V. K. (2021). Inception recurrent convolutional neural network for object recognition. Machine Vision and Applications, 32(1), 28. https://doi.org/10.1007/s00138-020-01157-3

Arjun, P., & Mirnalinee, T. (2016). Machine parts recognition and defect detection in automated assembly systems using computer vision techniques. Rev. Téc. Ing. Univ. Zulia, 39(1), 71–80.

Birari, H. P., lohar, G. V., & Joshi, S. L. (2023). Advancements in Machine Vision for Automated Inspection of Assembly Parts: A Comprehensive Review. International Research Journal on Advanced Science Hub, 5(10), 365–371. https://doi.org/10.47392/IRJASH.2023.065

Boby, R. A., Sonakar, P. S., Singaperumal, M., & Ramamoorthy, B. (2011). Identification of defects on highly reflective ring components and analysis using machine vision. The International Journal of Advanced Manufacturing Technology, 52(1–4), 217–233. https://doi.org/10.1007/s00170-010-2730-3

Burresi, G., Lorusso, M., Graziani, L., Comacchio, A., Trotta, F., & Rizzo, A. (2021). Image-Based Defect Detection in Assembly Line with Machine Learning. 2021 10th Mediterranean Conference on Embedded Computing (MECO), 1–5. https://doi.org/10.1109/MECO52532.2021.9460291

Cumbajin, E., Rodrigues, N., Costa, P., Miragaia, R., Frazão, L., Costa, N., Fernández-Caballero, A., Carneiro, J., Buruberri, L. H., & Pereira, A. (2023). A Systematic Review on Deep Learning with CNNs Applied to Surface Defect Detection. Journal of Imaging, 9(10), 193. https://doi.org/10.3390/jimaging9100193

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, & Li Fei-Fei. (2009). ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Dhillon, A., & Verma, G. K. (2022). A Multiple Object Recognition Approach via DenseNet-161 Model. In Smart Electrical and Mechanical Systems (pp. 39–64). Elsevier. https://doi.org/10.1016/B978-0-323-90789-7.00009-9

Frustaci, F., Perri, S., Cocorullo, G., & Corsonello, P. (2020). An embedded machine vision system for an in-line quality check of assembly processes. Procedia Manufacturing, 42, 211–218. https://doi.org/10.1016/j.promfg.2020.02.072

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. Nature Machine Intelligence, 2(11), 665–673. https://doi.org/10.1038/s42256-020-00257-z

Gildenblat, J., & Contributors. (2021). PyTorch library for CAM methods. GitHub.

Guo, Z., Wu, Z., Liu, S., Ma, X., Wang, C., Yan, D., & Niu, F. (2020). Defect detection of nuclear fuel assembly based on deep neural network. Annals of Nuclear Energy, 137, 107078. https://doi.org/10.1016/j.anucene.2019.107078

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.

Herakovic, N. (2010). Robot Vision in Industrial Assembly and Quality Control Processes. INTECH Open Access Publisher London, UK.

Koonce, B. (2021). ResNet 50. In Convolutional Neural Networks with Swift for Tensorflow (pp. 63–72). Apress. https://doi.org/10.1007/978-1-4842-6168-2_6

Kumar, G., & Bhatia, P. K. (2014). A Detailed Review of Feature Extraction in Image Processing Systems. 2014 Fourth International Conference on Advanced Computing & Communication Technologies, 5–12. https://doi.org/10.1109/ACCT.2014.74

Mazzetto, M., Southier, L. F., Teixeira, M., & Casanova, D. (2019). Automatic classification of multiple objects in automotive assembly line. 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 363–369.

Nazare, T. S., de Mello, R. F., & Ponti, M. A. (2018). Are pre-trained CNNs good feature extractors for anomaly detection in surveillance videos?

Panzer, M., & Bender, B. (2022). Deep reinforcement learning in production systems: a systematic literature review. International Journal of Production Research, 60(13), 4316–4341. https://doi.org/10.1080/00207543.2021.1973138

Patel, S. (2024). Feature Extraction in Image Processing: Techniques and Applications. Geeksforgeeks.

Pierleoni, P., Belli, A., Palma, L., Palmucci, M., & Sabbatini, L. (2020). A machine vision system for manual assembly line monitoring. International Conference on Intelligent Engineering and Management (ICIEM), 33–38.

Roboflow, & James, G. (2023). How to Auto Train YOLOv8 Model with Autodistill.

Rusli, L., & Luscher, A. (2018). Fastener identification and assembly verification via machine vision. Assembly Automation, 38(1), 1–9. https://doi.org/10.1108/AA-08-2016-093

See, J. E. (2012). Visual Inspection: A Review of the Literature.

Tan, M., & Le, Q. (2021). EfficientNetV2: Smaller Models and Faster Training. In M. Meila & T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning (pp. 10096–10106). PMLR.

TorchVision Maintainers, & Contributors. (2016). TorchVision: PyTorch's computer vision library. BSD-3-Clause.

Varshney, P. (2020). VGGNet-16 Architecture: A Complete Guide. Kaggle.

Weiss, E., Caplan, S., Horn, K., & Sharabi, M. (2024). Real-Time Defect Detection in Electronic Components during Assembly through Deep Learning. Electronics, 13(8), 1551. https://doi.org/10.3390/electronics13081551

Würschinger, H., Mühlbauer, M., Winter, M., Engelbrecht, M., & Hanenkamp, N. (2020). Implementation and potentials of a machine vision system in a series production using deep learning and low-cost hardware. Procedia CIRP, 90, 611–616. https://doi.org/10.1016/j.procir.2020.01.121