Master Degree Project

UNIVERSITY OF SKÖVDE

QUALITY INSPECTION OF MULTIPLE PRODUCT VARIANTS USING NEURAL NETWORK MODULES

Master Degree Project in Virtual Product Realization
Two year Level 30 ECTS
Spring term 2022

Fredrik Vuoluterä

Supervisor: Stefan Ericson

Examiner: Sunith Bandaru

Abstract

Maintaining quality outcomes is an essential task for any manufacturing organization. Visual inspections have long been an avenue to detect defects in manufactured products, and recent advances within the field of deep learning has led to a surge of research in how technologies like convolutional neural networks can be used to perform these quality inspections automatically. An alternative to these often large and deep network structures is the modular neural network, which can instead divide a classification task into several sub-tasks to decrease the overall complexity of a problem. To investigate how these two approaches to image classification compare in a quality inspection task, a case study was performed at AR Packaging, a manufacturer of food containers. The many different colors, prints and geometries present in the AR Packaging product family served as a natural occurrence of complexity for the quality classification task. A modular network was designed, being formed by one routing module to classify variant type which is subsequently used to delegate the quality classification to an expert module trained for that specific variant. An image dataset was manually generated from within the production environment portraying a range of product variants in both defective and non-defective form. An image processing algorithm was developed to minimize image background and align the products in the pictures. To evaluate the adaptability of the two approaches, the networks were initially trained on same data from five variants, and then retrained with added data from a sixth variant. The modular networks were found to be overall less accurate and slower in their classification than the conventional single networks were. However, the modular networks were more than six times smaller and required less time to train initially, though the retraining times were roughly equivalent in both approaches. The retraining of the single network did also cause some fluctuation in the predictive accuracy, something which was not noted in the modular network.

Keywords: quality inspection, defect detection, variants, modular neural network, convolutional neural network, case study

Acknowledgements

I’d like to thank Andreas and Therese for acting as my company supervisors, your support made the project possible. I’d also like to thank AR Packaging as a whole, but specifically the personnel at the Tibro production facility. The interest in the study, help in finding defective products and welcoming attitude made my visits a pleasant experience.

Furthermore, Richard and Bernard deserve an acknowledgement for helping me shape the early stages of the project and offering support throughout. A very special thanks goes to Ainhoa, who helped me find the perfect company to partner with for the study. Additionally, I’d like to acknowledge the patience and understanding from friends and family for enduring my ramblings about the thesis. Your support was essential to keep my spirits up during the study.

Lastly, I’d like to thank Sunith for taking on the role of examiner in a project which was at the time barely coherent and Stefan for relentlessly supporting me as the project supervisor, whether it pertained to acquisition of hardware, choice of algorithms, strategy for data collection, organizing the report and so many other things. I extend my most heartfelt gratitude to you.

Skövde, May 2022

Fredrik Vuoluterä

Certificate of Authenticity

Submitted by Fredrik Vuoluterä to the University of Skövde as a master's degree thesis at the School of Engineering.

I certify that all material in this master thesis project which is not my own work has been properly referenced.

Fredrik Vuoluterä

Table of Contents

Table of Figures

Index of Tables

1 Introduction

1.1 Background

Quality management has become a central concern for manufacturing organizations, as achieving good quality is necessary to remain competitive in the market (Harik & Wuest, 2020). In addition to the establishment of international standards (European Committee for Standardization, 2015a, 2015b), different frameworks have been developed to aid organizations in establishing well-functioning quality management systems, such as Total Quality Management and Six Sigma (Oakland, 2014). A prominent element within these larger quality management frameworks is quality control, which is implemented for the expressed purpose of ensuring manufactured products comply with quality requirements (Anand & Priya, 2020). A common approach to identifying products which do not conform to requirements is visual inspections (Harik & Wuest, 2020). Due to the relative cost and variability in using human operators to perform these inspections, automated machine vision systems are often desirable (Anand & Priya, 2020).

During the last decade, deep learning techniques have been increasingly used when constructing automated inspection systems, often utilizing convolutional neural networks (CNNs) (Czimmermann et al., 2020; Naranjo-Torres et al., 2020; Qi et al., 2020; J. Wang et al., 2018; J. Yang et al., 2020). While CNNs can solve a range of other tasks, they have proved to be very effective for visual classification and object recognition applications (Alom et al., 2018). As such, several recent studies (Chen et al., 2022; Riedel et al., 2022; Singh & Desai, 2022; Stephen et al., 2022) have used CNN-based vision systems to automatically detect surface defects in products.

Despite showing promise as a technique for defect detection (Czimmermann et al., 2020; Ferguson et al., 2018; Qi et al., 2020; J. Yang et al., 2020), many studies note issues and challenges which can occur when utilizing CNNs for quality inspection tasks. These include misidentifying product variants or defect types (Block et al., 2021; Huang & Ren, 2020; Nasiri et al., 2020; Y. Yang, Yang, et al., 2020), time-consuming training or retraining of a network (Czimmermann et al., 2020; Ferguson et al., 2018; Y. Yang, Pan, et al., 2020), need for large amounts of data (Naranjo-Torres et al., 2020; Qi et al., 2020; Tout et al., 2021; J. Yang et al., 2020) and the network size or computational cost being larger than desired (Naranjo-Torres et al., 2020; Y. Yang, Yang, et al., 2020).

It is also accepted that visual classification tasks with higher complexity require more data, larger networks and more resource-intensive training of the neural network (Amer & Maul, 2019). For quality inspection applications, this complexity may increase when more product variants are introduced to the network, resulting in a wider range of, for example, colors, materials, defect types and geometry.

Modular neural networks (MNNs) have been developed to mitigate the issue of growing complexity and do this by dividing the classification into several sub-tasks. Each module in a MNN consists of an independent neural network which is trained to deal with only a subsection of the total complexity. While multiple approaches exist to create MNNs, the connecting tissue between them is this division of complexity among a collection of modules, in contrast to conventional 'monolithic' neural networks which instead attempt to handle all complexity in one network (Amer & Maul, 2019).

1.2 Problem Description

AR Packaging is a manufacturer of different kinds of trays and containers used in the food industry, such as for fast food or microwave meals. The main material used is carton board, but some product variants make use of plastic and aluminum to meet customer requirements. As the products are intended for use in the food industry, strict quality and hygiene standards are enforced in the manufacturing process. A defective tray may cause food leakage, vacuum seals breaking, contamination or issues when the food producer fills the tray. Today the company mainly relies on experienced human operators to inspect products for defects and identify those that do not meet quality specification. Due to the variability, cost and speed of manual inspections there is an interest in developing an automated inspection system.

Given the diverse materials and features within the AR Packaging product family, along with the need for automatic inspection solutions, the company offers an interesting opportunity to investigate the usefulness of modular CNN solutions for quality inspection. As the company continuously develops new designs based on customer demand, changes existing materials for more sustainable alternatives and experiences shifts in which products are manufactured at the site, it is imperative that the inspection solution not only performs well in the current product family, but is further able to easily be updated when changes in the product family occur.

1.3 Aim and Objectives

The intersection of visual quality inspections, CNNs and modularity form the central theme of the thesis, which seeks to explore how neural network modules perform compared to conventional neural network implementations. The aim of the thesis is thus to answer the research question:

- How do conventional monolithic neural networks compare to modular neural networks in quality inspection applications with high product variety and frequent changes to the product family?

For this question to be answered, the thesis has been divided into a series of objectives to be achieved:

1. Review the contemporary literature of how CNNs are applied in quality inspection tasks.
2. Plan the study and define the design of both the monolithic and modular approach.
3. Perform an initial study using publicly accessible datasets to verify the designs.
4. Collect image data of several product variants in the production facilities of AR Packaging.
5. Use data from a selection of products to train both the monolithic and modular network designs.
6. Compare performance and training process using established metrics in the field.
7. Add data from one or more variants and re-train the monolithic and modular network designs.
8. Evaluate and compare both the new performance and the re-training process.
9. Document and summarize the project in a report.

By applying the two different approaches to the same task, using the same training data, it will be possible to evaluate how fit each approach is with regard to not only the accuracy of the quality inspection, but also factors concerning training or re-training, input data required, size of the networks, response time and how data handling is affected.

1.4 Methodology

As the thesis is contained to a single use case, it will be performed within the methodological framework of a case study, drawing heavily upon guidelines for the method by Runeson & Höst (2008). They define a case study as an investigation of a contemporary phenomenon in its context. They further describe it as an empirical method and note similarities between case studies and quasi-experiments conducted in industrial settings. Case studies also often lack strong experimental controls. These properties are easily found in the thesis, making a case study approach a good fit.

The thesis will be exploratory in its nature, seeking to establish how modular neural networks perform in comparison to the dominant monolithic approach to quality inspections. In addition, it will act within the positivist paradigm, gathering empirical data to explain the interactions of the two approaches and reinforce the conclusions of the study.

Runeson & Höst (2008) outline five major steps which are necessary for a successful case study. The first step, case study design, involves planning the study and defining objectives to be achieved. Next is preparation for data collection, where the approach and methods for collecting data is defined. The third step is evidence collection, where data collection for the case study is performed. Then step four, analysis of collected data, seeks to generate some result. The fifth and final step is reporting, in which the findings of the case study are presented.

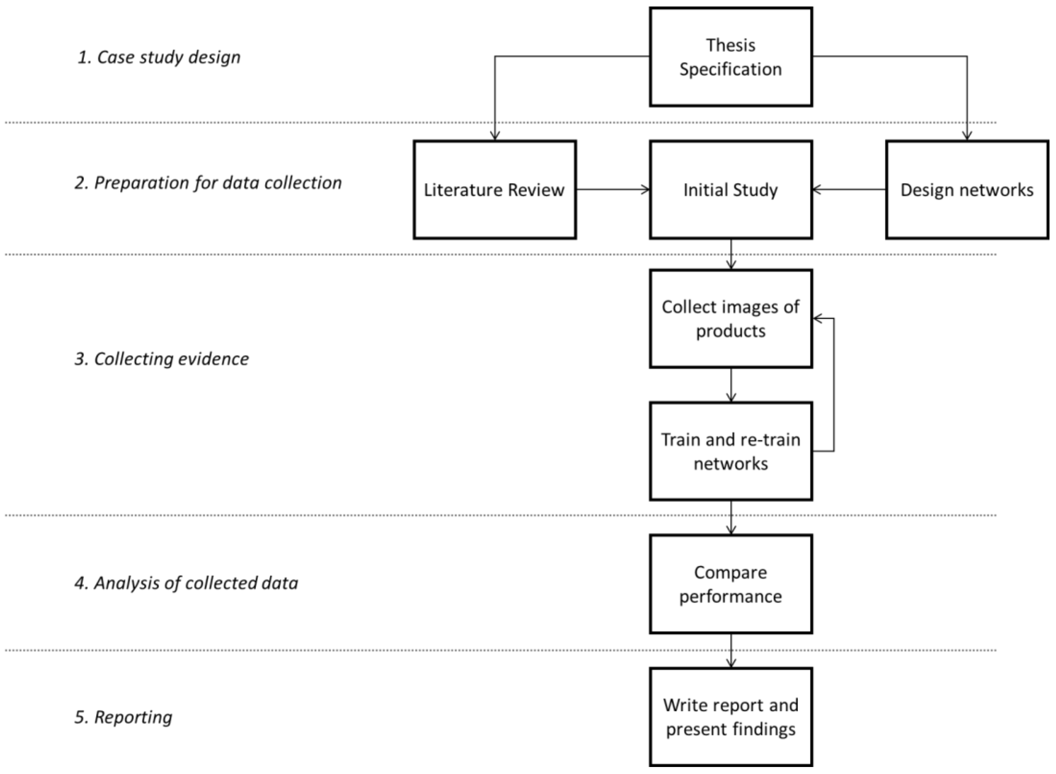

1. Case study design
Thesis Specification
2. Preparation for data collection
Literature Review
Initial Study
Design networks
3. Collecting evidence
Collect images of products
Train and re-train networks
4. Analysis of collected data
Compare performance
5. Reporting
Write report and present findings

Figure 1: Five steps of a case study (Runeson & Höst, 2008) mapped onto project activities.

Data collection and analysis for the thesis will however be performed with two distinct objectives in mind. The first one is to gather image data for the neural networks to train on, which implicitly requires some degree of analysis of the collected images. The second will seek to extract data for the final analysis from the networks regarding training, accuracy and other relevant measurements of their performance. As such it diverges slightly from the idealized framework presented by Runeson & Höst (2008), but they note that case studies are by nature flexible and data collection can be conducted incrementally. Thus, these steps can be mapped onto project activities, see Figure 1.

1.5 Scope

The study will limit itself to investigating surface defects in the products, such as scratches, holes, cracks, tearing or deformations. Other quality properties, like product dimensions being within tolerances, will not be included in the study.

Only a selection of product variants and defect types found at AR Packaging will be included in the study. This is both due to the number of different variants exceeding what is feasible to examine given the resources of the thesis and some defect types occurring infrequently, making the generation of training data hard.

2 Frame of Reference

2.1 Quality Management

Quality management systems are the means by which an organization seeks to ensure any products and services they provide conform to set requirements (European Committee for Standardization, 2015a). Within manufacturing organizations there is a range of properties that can be considered when evaluating the quality of a product, such as adherence to tolerances or surface quality (Harik & Wuest, 2020). Oakland (2014) notes the danger of delivering unreliable, poor-quality products or services, as it damages customer trust, generates a poor reputation and lowers the organizations competitiveness. Furthermore, there may be other implications of poor quality. Complying with regulations, such as food safety guidelines or ensuring brake functionality in vehicles, is required not only to protect the end user but also to prevent costly recalls, bad press coverage and legal consequences like fines.

Several different approaches to supporting the goals of achieving quality have emerged, such as Total Quality Management, Six Sigma and Lean production. These approaches often overlap or can be used in conjunction with each other but share the common goal of satisfying customer expectations with regard to quality, reliability, price and delivery. These considerations are not a strictly internal matter, as suppliers may have significant impact on the ability to meet them and customer requirements may shift over time (Oakland, 2014).

Harik & Wuest (2020) distinguish between process quality and product quality, noting that the former affects the latter. As each stage in the manufacturing process impacts the quality of the final product, a holistic understanding and awareness of the process is important for quality management. Another distinction present within quality management is between quality control and quality assurance. According to the ISO 9000 standard (European Committee for Standardization, 2015a) quality assurance seeks to increase confidence in the achievement of quality requirements while quality control is directly concerned with fulfilling said requirements. This difference is echoed by Poornima (2017) and Anand & Priya (2019), who describe quality control as being product-oriented and focusing on identifying defects whereas quality assurance is process-oriented and seeks the prevention of product defects.

2.1.1 Quality Inspections

One of the main tools of quality control is visual quality inspections, used to verify the quality of a part or product (Harik & Wuest, 2020). The inspections can be done at any stage of production, either on incoming materials, products or components within the manufacturing process or finished products. Inspected subjects which do not meet set quality requirements are rejected while those that conform progress to the next stage of production or are delivered to the customer (Anand & Priya, 2020). The traditional form of inspection is done by human operators, inspecting either all or a sampling of the finished products at the end of the production process (Harik & Wuest, 2020). However, in addition to finished goods inspection, it is common practice to inspect parts and products during their manufacturing, called in-process or in situ inspection (Goetsch & Davis, 2013). Visual inspection methods are typically limited to detecting surface defects, such as scratches and deformations, while internal quality issues generally require other techniques to detect (Harik & Wuest, 2020).

Manual inspections do suffer from a number of downsides, such as operators getting bored (Goetsch & Davis, 2013), experiencing fatigue, being unable to keep up with production rates or varying in their judgement (Anand & Priya, 2020). These factors make inspections less reliable, even when 100% of all products are inspected, risking either defective products being approved or non-defective products being discarded. In addition, the process of manual inspection is expensive, time-consuming and is in some cases either infeasible or impossible to perform (Harik & Wuest, 2020).

Automatic inspection systems utilizing machine vision technology is an alternative to using human operators. According to Anand & Priya (2019), there are some key benefits of machine vision inspection compared to manual inspections. Firstly, machine vision systems can achieve higher accuracy and precision compared to humans. Speed is another factor, as inspections can be carried out at a faster rate than conventional quality assessments. The measurements are also repeatable and can be done continuously with the same degree of accuracy, as the system does not experience tiredness or boredom. Machine vision systems can thus be more feasibly tasked with inspecting 100% of all products and are more cost effective in their operation. Anand & Priya (2019) further point out that machine vision performs noncontact inspections, reducing the risk of the inspection damaging the product, and enables inspections to be performed in hazardous environments where human operators cannot be present.

While there exists a variety of different approaches and techniques to implement machine vision algorithms, increasing interest has been garnered for applications using artificial intelligence (AI) and machine learning (ML). Combined with industrial-grade equipment, neural networks in particular have shown the ability to successfully perform high-speed inspections tasks (Harik & Wuest, 2020).

2.2 Deep Learning

Deep learning is a subset of the greater study of AI and ML. Artificial Neural Networks (ANNs), which are simplified mathematical models of how neurons in a brain interact (Khishe & Parvizi, 2020), are central to any deep learning application. ANNs are constructed in layer of neurons, as shown in Figure 2, with each neuron being connected to all the neurons in both the preceding and succeeding layers. Each connection has a weight, usually a number between 0 and 1, which is multiplied with the value passed on from the previous neuron. Each neuron adds all the incoming values and uses the sum as input for an activation function, mimicking the process of how real neurons process information. Values get propagated through the network, from the input layer to the output layer, and generate a result in the final node or nodes (Di et al., 2018).

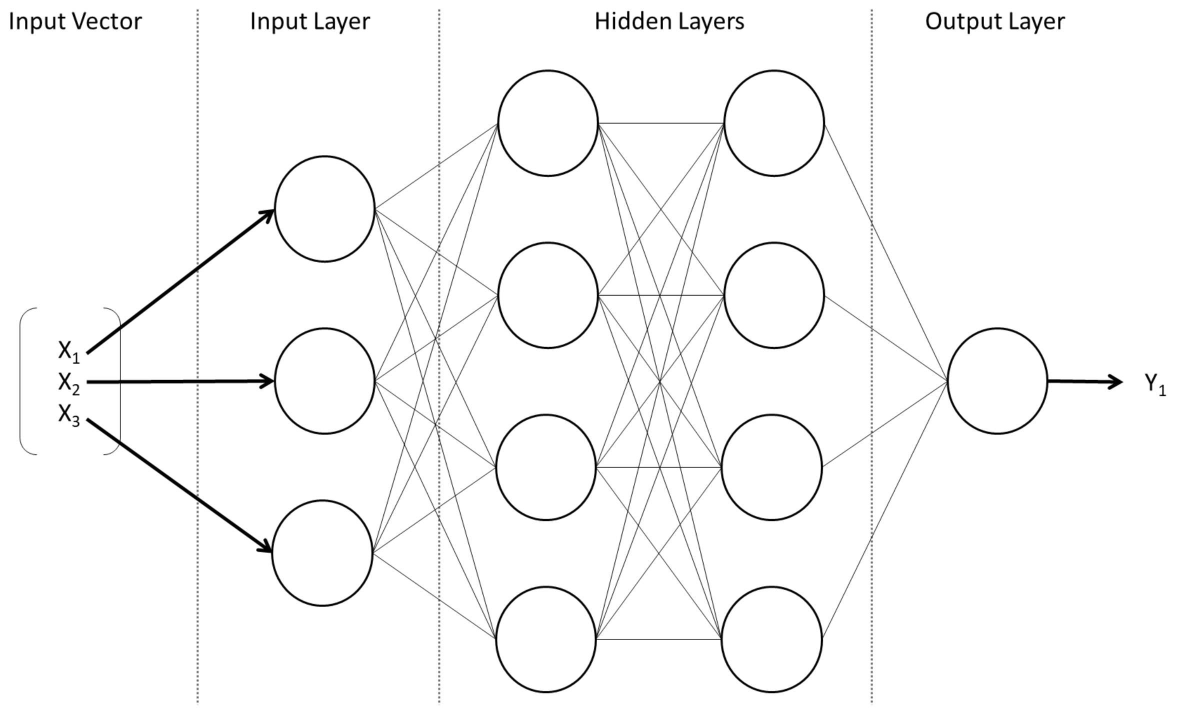


Figure 2: Illustration showing the structure of a simple neural network.

The idea of constructing ANNs are far from a new phenomenon, with the first mathematical model being suggested in 1943. Over the years interest and funding for the field fluctuated, with the most dire periods being dubbed "AI winters". Despite these setbacks the research progressed, with deep neural networks emerging during the 1980s. These new ANNs utilized more layers in their architecture, hence the description as "deep", enabling more complex task to be solved (Di et al., 2018).

A recent surge of interest in deep learning started a decade ago, with the breakthrough of AlexNet, a CNN. AlexNet was entered into the 2012 ImageNet Large Scale Visual Recognition Challenge, a contest where different image recognition algorithms are tasked with classifying images belonging to 1000 different classes. No other entry managed to come close to the accuracy of AlexNet, which achieved a top five error rate of just 15,3% compared to the second-place finishers 26,2% (Di et al., 2018). The manner in which AlexNet outperformed all contemporary visual recognition approaches started a rush to iterate and improve upon the architecture, spawning a host of new CNN architectures in the following years which managed to achieve error rates below that of a human (Alom et al., 2018).

2.2.1 Neural Network Training

As ANNs are not inherently able to solve tasks, the networks need to be trained to “learn” the features of a problem. This is accomplished by tuning the weights within the network to generate outputs which correctly, or close to correctly, correspond to the inputs (Rumelhart et al., 1986). One approach to training networks which has gained popularity within deep learning is backpropagation. It is a supervised learning technique relying on labelled data (Alom et al., 2018) and was used by Rumelhart et al. (1986) together with the deep neural networks being developed in the 1980s to demonstrate how data features could be created and represented. LeCun et al. (1998) further show the applicability of backpropagation in deep neural networks for practical purposes, such as recognizing handwritten letters and digits, and highlight CNNs as a particularly effective technique for visual recognition.

When a neural network classifies input data, this data is propagated forward in the network until it reaches the last layer and an output can be generated. This is known as forward propagation (Goodfellow et al., 2016). However, unless the network is completely accurate, some error between the actual classes and predicted classes will occur. By using a set of known input-output pairs, the performance and total error of the network can be known. This is where backpropagation is used to propagate the error backwards in the network. The error for each individual weight is calculated using the partial derivative of the error function, allowing each weight to be tuned to minimize the total error of the network (Rumelhart et al., 1986). This process repeats to iteratively reduce the total error until a stop condition is met, such as a threshold of accuracy or number of iterations.

Contemporary methods for training neural networks usually divide data into two datasets, one used for training and the other for validation. An issue that plagues deep learning networks is overfitting to the training data, thus seeming accurate while training but failing to generalize to real world applications. To mitigate this issue, the error rate between the training and validation datasets can be compared and evaluated for divergence, as the validation data is not used in backpropagation and can act as a stand-in for real world data. If the network achieves progressively better results for the training set while the accuracy for the validation set changes at a different rate, see Figure 3, it is reasonable to assume overfitting is occurring as the network is less attuned to the general features of the population compared to the specific features of the training sample (Goodfellow et al., 2016).

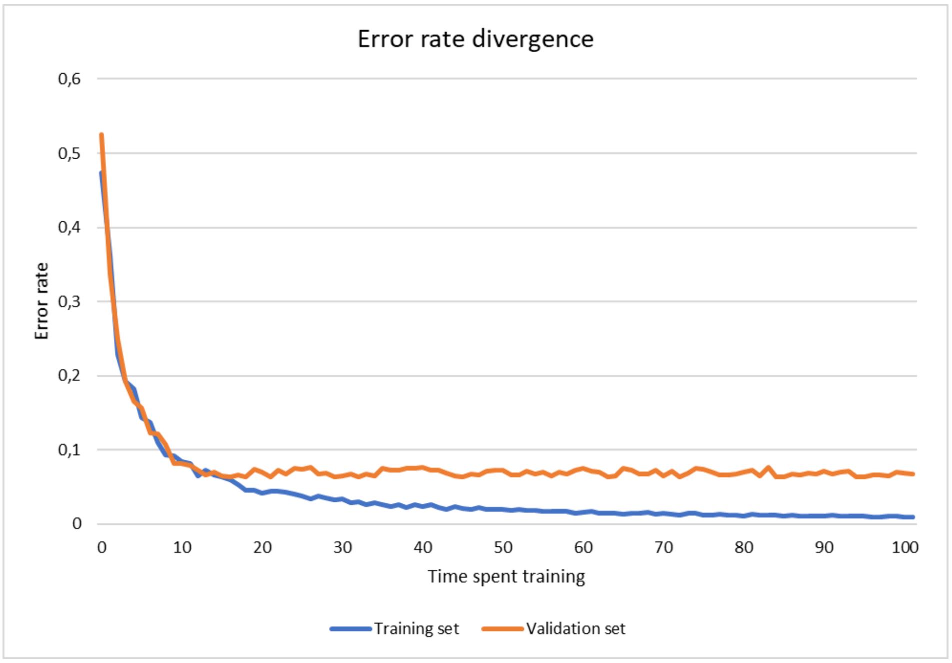


Figure 3: Example of divergence in the error rate between training and validation data.

Despite showing promise at the time, and being viewed as state-of-the-art techniques to this day, some factors held back CNNs and backpropagation from widespread use until recently. Goodfellow et al. (2016) note that the availability of more data and advances in both hardware and software infrastructure contributed to massive improvements in the performance of neural networks, as the networks could better generalize with more data and larger networks with more layers could be feasibly trained.

2.2.1.1 Dropout

Dropout is a method to further mitigate overfitting when training neural networks, described extensively in a paper by Srivastava et al. (2014). It works by “thinning” the network, temporarily disabling random neurons during training from receiving or passing on values. This method is highly effective as it prevents the training algorithm from relying on certain sections of the network, and instead generalizes the learned features to more neurons. Like ANNs themselves, the function of dropout has a parallel in nature. Specifically, genes which can work well with a random assortment of other genes create more robust systems, rather than relying on individual fitness or a large set of partners. While Srivastava et al. (2014) do find increased performance for multiple applications when using dropout, they also conclude that using the technique requires training to be performed for 2-3 times longer than conventional training methods without dropout.

2.2.1.2 Labelling

Labelling is the process of creating labels for unlabeled data. Within image classification tasks, labels are necessary not only as a precondition for training using backpropagation, but also to determine the output classes (Alom et al., 2018; Goodfellow et al., 2016). The process of manually labelling large image datasets can however be immensely labor-intensive, time-consuming and costly as well as requiring experts to perform the labelling (Alencastre-Miranda et al., 2021; Nasiri et al., 2020; Zheng et al., 2021). Sources of already labelled data and methods which decrease the need for labelled data, such as transfer learning, are therefore highly desired in classification tasks.

2.2.1.3 Data Augmentation

Another approach used to reduce overfitting for image datasets is data augmentation. The existing dataset is extended by, for example, flipping images horizontally or vertically, dividing images in patches, modifying color properties in the images or rotating images. These transformations also have the benefit of maintaining the label of the original image, lessening the burden of labelling. Thus, a single image can yield multiple other images that preserve the features of the object while at the same time increasing the ability of the network to handle variation in how objects are presented (Krizhevsky et al., 2012).

2.2.1.4 Transfer Learning

The weights in a neural network are commonly randomly initialized before training, but an alternative is to use transfer learning. This methods uses weights from a previously trained network to “transfer” knowledge of features instead of training from a randomized set of parameters (Alom et al., 2018). Transfer learning can be conceptualized as “fine-tuning” the weights of a network to a new task (Di et al., 2018; Shin et al., 2016). Pre-trained neural network models are preferred when there is a lack of training data, as they require less data to achieve good performance. Transfer learning is also less computationally expensive compared to training new networks, aids the generalizability of the network and attains accuracy quicker (Alom et al., 2018).

2.3 Modular Neural Networks

Modularity is an important concept within many different fields, but it is understood slightly differently depending on how it is applied. For example, in the context of Industry 4.0, Saxena & Vijaivargia (2020) define modularity with regard to modular systems, emphasizing the ability to adapt to emerging circumstances. Modularizing a production system essentially allows more flexibility and rapid reconfigurations, with each module being able to collaborate in different ways. A more general understanding is offered by Kamrani & Salhieh (2002) in discussing modular product design. They state the necessity of modular product being able to be coupled to form a complex whole. Each module can perform some function, but it is the connection between them that truly enables the full use of these functions. One example the authors give is a computer, which can freely exchange or add a range of different components depending on which functions are desired.

However, Kamrani & Salhieh (2002) further make clear that a modular design is not a web of complex interactions. Rather, interactions between modules should be kept to a practical minimum, while module design and production should be independent from each other. To achieve this the full task which the modular design should attempt to solve should be thoroughly decomposed to properly understand which sub-functions are required and how they can be independently achieved by a module. This definition lies closer to the ideas underpinning the development of MNNs.

MNNs are built on the concept of modularity, with each module containing a neural network. In discussions of MNNs, conventional neural networks utilizing a single, interconnected structure are often called monolithic neural networks to distinguish them from the modular approach (Amer & Maul, 2019; Castillo-Bolado et al., 2021). Proponents of MNNs claim several advantages over monolithic techniques, such as the ability to break down complexity into smaller pieces (Amer & Maul, 2019; Hu, 2020), exchange or reuse modules (Castillo-Bolado et al., 2021), be scalable (Goel et al., 2021; Valdez et al., 2019), reduce the amount of parameters (Goel et al., 2021; Intisar & Zhao, 2019) and use less computational power (Castillo-Bolado et al., 2021; Goel et al., 2021).

While monolithic approaches are still dominant in the deep learning field, modularization has shown promise in problems with large scale and complexity (Amer & Maul, 2019), including recent studies within speech recognition (Ansari & Seyyedsalehi, 2017), image classification (Goel et al., 2021) and pattern recognition (Valdez et al., 2019).

However, development of MNNs necessitate decomposition of the problem space into sub-tasks, and a process for generating a modular structure dealing with those sub-tasks (Castillo-Bolado et al., 2021; Goel et al., 2021; Meng et al., 2020). While modularization techniques exist, they lack extensive research which have made them more difficult to employ than more established monolithic approaches (Amer & Maul, 2019).

Just as there is variety within monolithic neural networks approaches, modularization processes for MNNs can differ greatly. In a literature review by Amer & Maul (2019) on modularization techniques four classes of these techniques are described, those being Domain, Topology, Formation and Integration. These can be understood as methods of manipulating the properties of the MNN depending on the task the network is being deployed to solve.

2.3.1 Domain

Domain modularization deals with how input data used by the MNN is partitioned. One justification for modularizing the data is that modules can act on sub-domains of the data and not have to learn the features of the entire dataset. How data is partitioned depends heavily on the application of the MNN and can be based on any number of data features. This step is directly related to problem decomposition, and good domain modularization can result in vastly lower problem complexity. However, modularizing the data is not mandatory for MNN applications (Amer & Maul, 2019).

Domain modularization can be performed in a number of ways, but generally fall into one of two categories, those being Manual or Learned. Learned domain modularization relies on learning algorithms to partition the data into appropriate sub-domains (Amer & Maul, 2019). One example of this approach can be found in a paper by Goel et al. (2021), where a MNN is developed for image classification. Their approach relies on a visual similarity metric to group object classes with other visually similar objects. To demonstrate this, they note how vehicles like trucks and cars would be grouped separately from animals like cats and dogs, at which point the module handling animals would not need to train on the features of a vehicle and vice versa. The groupings can be generated dynamically, allowing the number of modules to fit the needs to the data.

Manual domain modularization on the other hand requires prior knowledge of the data and can be hard to perform on large datasets or datasets with unclear boundaries. It does however allow very fine control over the modularization and lacks the computational expense of an algorithmic approach (Amer & Maul, 2019). In addition, as it requires knowledge of the problem space, known best practices can emerge to further ease manual domain modularization.

2.3.2 Topology

Topological modularity describes how the units within a MNN interact with each other and the number of modules, thus generating the network structure. While a modular topology is a precondition for an ANN to be considered a MNN, it will only achieve true modularity if the functionality of the network is distributed across the topological modules as well. Attaining functional modularity enables more opportunities for debugging the network as predictive functions are concentrated to parts of the MNN, in contrast to monolithic networks which distribute their representational abilities across the entire network resulting in a logical black box (Amer & Maul, 2019).

Several modular topologies exist, each with their own advantages and disadvantages. One of these is the repeated block class of topologies. Repeated block MNNs are, as the name implies, a repetition of building blocks in a specified configuration. These blocks should share some common characteristics with each other, although they do not need to be exact copies. Repeated block topologies are usually easy to describe analytically and can be extended using simple rules. These are not traits found in all modular topologies as some, like the Highly-clustered non-regular topology, can't be described as a repeating pattern due to its non-regularity (Amer & Maul, 2019).

Depending on how a repeated block MNN is constructed, it can be placed into several sub-topologies, see Figure 4. The multipath topology is constituted by semi-independent sub-networks which only converge at the network input and output. This structure enables parallelization of the network. Using a multipath MNN does however add new parameters to the construction of the network, such as the size and number of modules. To determine proper values for these parameters optimization may be required during the construction of the network (Amer & Maul, 2019).

Another sub-topology of repeated block MNNs is the sequential approach. The structure is similar to multipath, but instead of input data passing through the modules in parallel a sequential MNN passes data through all modules in series. This topology shares origins with deep learning, increasing depth to enable more complex representations. By dividing the depth into modules, higher and lower features are learned in different modules and thus concentrating representational knowledge in separate parts of the network structure (Amer & Maul, 2019). Sequential MNNs are however unable to parallelize their operation and suffer from some of the same downsides as monolithic networks.

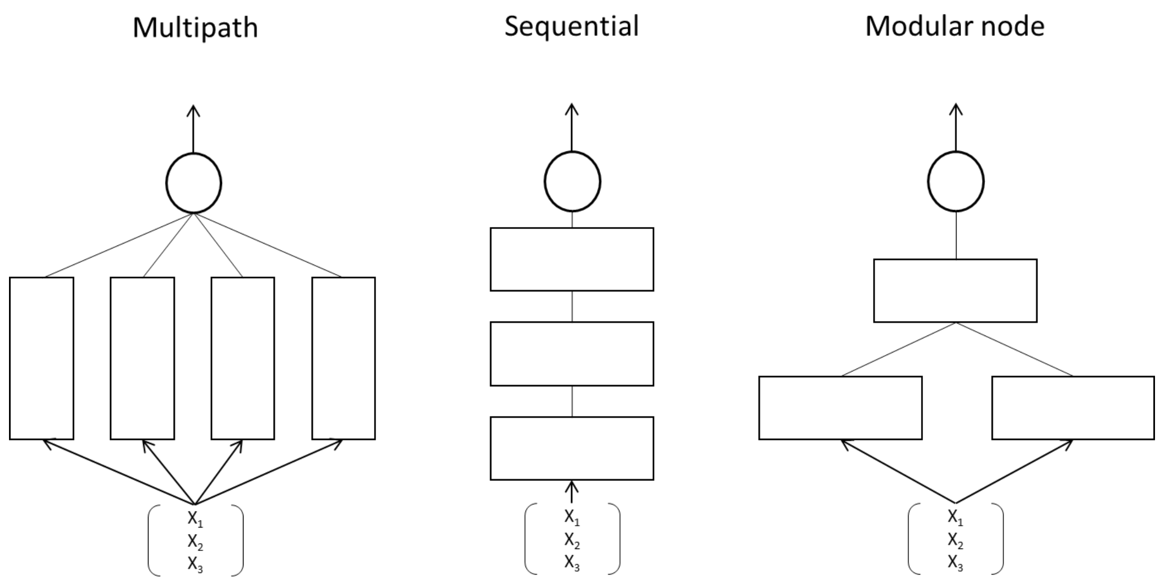


Figure 4: Diagram of different repeated block sub-topologies, inspired by Amer & Maul (2019)

The modular node sub-topology combines the parallel and serial nature of multipath and sequential MNNs and structures its modules analogously to neurons in a conventional monolithic neural network. While the repeatable structure makes it easy to scale and studies have shown it can reduce the number of model parameters while maintaining performance, it also introduces more parameters to be considered in network construction (Amer & Maul, 2019).

Other classes of topology exist as well, such as the multi-architectural topology. It can share the structure of repeated block topologies but carries the distinction of being a combination of fully independent network architectures which are connected through some overriding algorithm. As the modules within a multi-architectural MNN can differ in their architecture, these networks are usually tailored to the specific needs of an implementation rather than being repeatable blocks with similar properties. As such, this topology can be more specialized, but the training of multiple different architectures may require more time and computational resources than a more homogenous approach (Amer & Maul, 2019).

2.3.3 Formation

Formation is the process by which the topology of the MNN is constructed. Amer & Maul (2019) divide these methods into three classes, Manual, Evolutionary and Learned. The manual technique, just as the manual approach to domain modularization, relies on a human designer to form the network using previous knowledge, best practices and intuition. While this approach offers the opportunity to integrate knowledge of the problem into the very structure of the MNN, it can be difficult to perform when there is low understanding of problem space.

Evolutionary and Learned formation both offer automated alternatives when constructing MNNs. Evolutionary algorithms are currently considered state of the art in formation of MNNs, as the fitness functions they deploy have proven to excel at finding suitable parameters, such as number of modules and how to connect them to each other. Learned formation relies on machine learning algorithms to dynamically construct the network, choosing how to apply the topology and the associated parameters. Both of these algorithmic solutions to network formation have the downside of being computationally expensive and lack the expert knowledge which can be introduced in manual formation (Amer & Maul, 2019).

2.3.4 Integration

Integration is the feature of MNNs seeking to specify how different modules collaborate to generate network outputs. Amer & Maul (2019) claim this process can be handled cooperatively or competitively, as the modules either try to collectively contribute to the final output or compete to be the only module to have its output passed forward. This negotiation of outputs can be handled either through a logic-based approach, stating rules based on the information known to the network, or through a learning algorithm, which can combine outputs from different modules to achieve the best performance. Such learning algorithms are useful for problems characterized by complex interactions and relationships which are hard to solve through predefined rules.

3 Literature Review

3.1 Quality Inspection using Convolutional Neural Networks

Studies on the application of CNNs in quality inspections tasks have steadily increased over the past few years. This is clearly visualized in Figure 5, which shows the number of papers on Scopus containing the keywords “defect detection” or “quality inspection” alongside either “convolutional neural network” or “CNN”. As with any rapidly expanding research area, there are many different methods and approaches being employed, as well as their application in different contexts. Despite the existing differences within the field, it is possible to find common themes among many of the published papers, providing valuable lessons when applying CNNs in a quality control role.

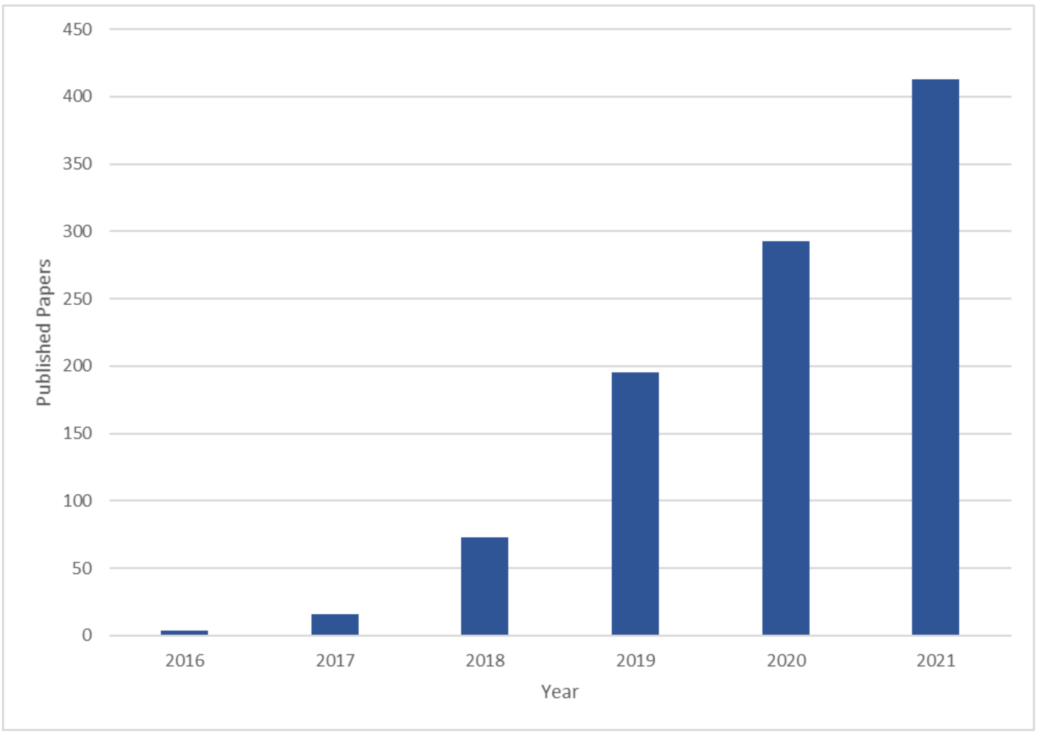


Figure 5: Number of published papers found on Scopus concerning both quality inspection and CNNs.

Studies often highlight the comparative advantages of a deep learning approach in quality inspection over not only manual visual inspection, but also other automated methods. In a study by Weimer et al. (2016) it is noted that manual feature engineering for automated vision systems often struggle with noisy input and classification of complex surfaces. This idea is echoed in a paper by T. Wang et al. (2018), where the authors claim that hand-crafted features often require specific conditions to work as intended. Both studies then go on to highlight CNN models as alternative due to their inherent ability to automatically extract features in their training process and handle noise in the input data.

A frequent challenge in these studies is the acquisition of sufficient data to train their neural network models. This is a common issue for the wider deep learning field but is exacerbated in quality inspection tasks as defects may rarely occur, thus limiting the amount of data which can be feasibly collected. Several approaches exist in the literature to mitigate this lack of data. For example, in a study by Ferguson et al. (2018) which attempts to classify defects in casting products using X-ray imaging, transfer learning is utilized to increase the accuracy of the system. The authors first use two large public image datasets to attain the general features and aspects of casting defects, and then finetune the network with a smaller dataset. This leveraging of pretrained networks showed an accuracy of 95,7% compared to only 65,1% in a network which used randomly initialized weights before being trained only on the small casting dataset.

Another approach to solving the issue of imbalanced data is shown in a paper by Yun et al. (2020) seeking to classify defects in metal surfaces. The authors identify six different defect types in the metal surface, but the data is highly imbalanced, with the most common type occurring five times more often than the rarest one. To address both the imbalance and the amount of training data available they develop a conditional convolutional variational autoencoder, or CCVAE for short, which can generate new images based on the ones already collected. By training a CNN with both the original data and the dataset of generated images the network achieves higher accuracy and faster convergence than a network model using only the real-world data. Such data augmentation techniques to extend smaller datasets can be invaluable for cases with heavily imbalanced or small datasets.

There are also a vast array of studies showing CNNs being successfully applied in quality inspection tasks for a variety of products and materials, using different kinds of input data, across several sectors. Agricultural research have adopted CNNs in several roles, among them being defect detection or quality inspection of fruits (Naranjo-Torres et al., 2020), eggs (Nasiri et al., 2020), potatoes (Casaño et al., 2020) and coffee (Pradana-López et al., 2021). Studies within construction and property maintenance have also applied CNNs in inspection tasks, such as crack detection in concrete (Chordia et al., 2021) and ceramic tiles (Stephen et al., 2022). The manufacturing sector is however seeing a comparably massive number of similar quality inspection studies within various industrial processes and fields, such as additive manufacturing (B. Zhang et al., 2019), welding (Z. Zhang et al., 2019), metal production (Tao et al., 2018), wood-based manufacturing (Chen et al., 2022), electronics (Shu et al., 2021) and battery production (Badmos et al., 2020).

3.2 Modular Neural Network Studies

The research on MNNs is significantly thinner than the comparably popular application of CNNs, but the literature still covers a variety of different approaches of how to either develop or use a MNN. Chowdhury et al. (2021) demonstrate how a modular structure for classification tasks utilizing router and expert networks can achieve fast computation and small size while still being scalable and perform parallel computation. As their suggestion uses routing networks to activate only parts of the total structure, for example three expert modules which then integrate their output collaboratively, it avoids performing a lot of redundant computation in the parts of the network which lack knowledge of the relevant data features.

Goel et al. (2021) developed a similar system, but with a stronger focus on low energy consumption and ability to run on embedded devices. Their system uses a tree structure with branches across the problem space, grouping object classes based on a visual similarity metric. This creates super-groups of similar objects, such as animals or vehicles, which can then be further broken down into smaller groups until a 'leaf' module is found which can classify the object. The system shows that the avoidance of redundant operations can significantly reduce computation, memory requirements, response time and energy consumption compared to monolithic approaches.

Dedicated quality control applications for MNNs are somewhat rare in the contemporary literature. While many of the developed architectures or formation methods could be leveraged in a quality control task, the studies are not inherently about quality inspection or similar tasks. However, examples do exist. A paper by Kauer-Bonin et al. (2022) describes the development of a MNN quality inspection system within the healthcare sector. Retinal optical coherence tomography scans are used to diagnose various disorders or diseases, often neurological in nature, but high-quality scans are essential for proper analysis. While quality criteria exist for what is deemed acceptable, these evaluations are performed manually by experts in the field, resulting in the same shortcomings of manual inspections often noted in similar studies. To automate this process, the authors construct a modular structure containing three neural networks, each with separate functionality applied sequentially to the input data, such as confirming the scan is properly centered and contains complete information.

While the authors do not conceptualize their network as such, according to the definitions provided by Amer & Maul (2019), it follows a sequential multi-architectural topology using manual formation, collaborative integration and no domain modularization. The MNN developed by Kauer-Bonin et al. (2022) manages to achieve an accuracy of 97% as well as classification speeds of 0.301s in their largest model, making the system suitable for real-time use. Additionally, the network module checking completeness and signal strength for the scans manages to drastically reduce the computing time in both the CPU and GPU while using a fraction of parameters compared to monolithic CNNs found in other studies. The high speed, low computational cost and small size provide MNNs with a distinct advantage over monolithic alternatives, even if they add some complexity to the network development process.

Another application of MNNs which lies closer to traditional manufacturing defects is shown in a study by Rathinavel & Kannaianl (2018), where they propose a defect detection system for fabric which incorporates a MNN classifier. Problem decomposition is accomplished by partitioning the data using clustering, and the MNN processes the input in all modules. Then a gating network chooses which module produces an output based on what data it was trained on. The MNN achieves an accuracy of 96.7%, which the authors conclude demonstrates the strength of the technique.

4 Network Design

4.1 General Network Design

The monolithic and modular neural network approaches share a general workflow structure within this study, as can be seen in Figure 6. Upon receiving an input image, a pre-processing procedure is performed to enable more accurate classification as well as ensuring the image resolution corresponds to the expected size of input images in the network. See *6 Image Processing* for a more detailed description of the implemented image processing algorithm. Once processed, the image is transferred to a classification network which attempts to determine if the product is defective or not.

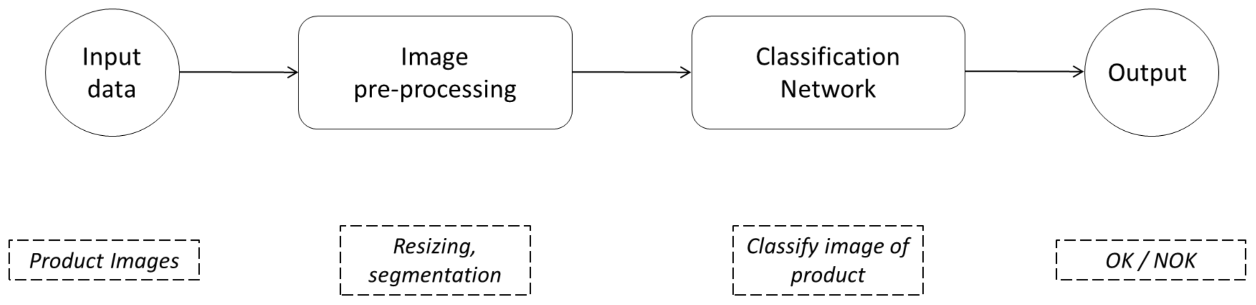


Figure 6: Workflow of the proposed classification networks.

The framework allows for easy exchange of the classification network to test how different approaches perform in the same conditions. Monolithic approaches, of which there are many architectures available for use, do not need any further design to function and can be implemented as soon as they have been trained. However, the implementation of a modular network requires additional steps, especially given how the very structure of modular networks can be tailored to specific use cases and specific sets of data.

4.2 Modular Design

As outlined in the previous section *2.3 Modular Neural Networks*, Amer & Maul (2019) propose the fundamental features of MNNs as Domain, Topology, Formation and Integration. These concepts heavily informed the construction of the MNN. The initial question of domain modularization is simple, if not trivial, in many quality inspection applications as the division of defective and non-defective products often form the central motivation for such implementations. However, there exists two further levels of possible data modularization which could be relevant depending on the specifics of a case.

The first one is defect type. As seen in other studies, such as Yun et al. (2020), defects may present in different ways and therefore could form an interesting basis for data partition. The second level is product variant. As an inspection system may be expected to encounter different kinds of products, modularizing data based on which product is present is a further possibility. With a central component of the case study being the classification of multiple product variants, the variant type alongside the quality distinction were chosen as the factors for data modularization. Thus, images are first clustered based on which variant they depict and secondly if they show defective products or not, see Figure 7.

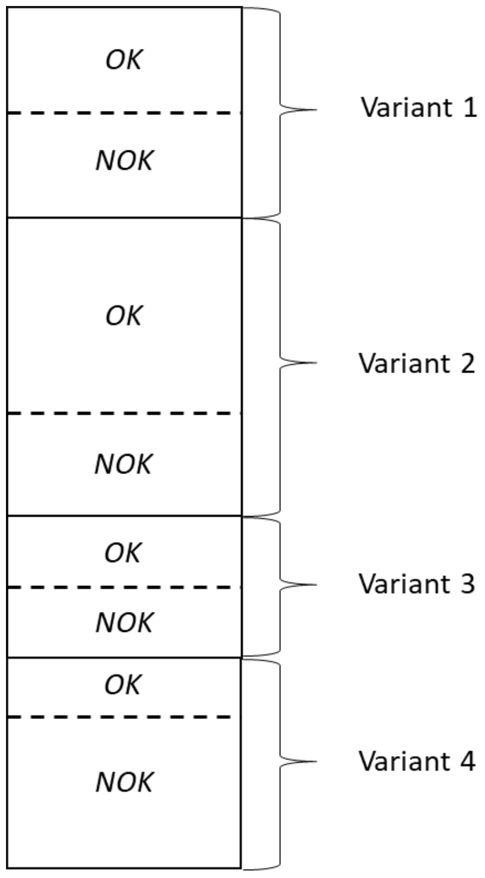


Figure 7: Illustration of how image data is clustered based on product variant and quality.

The possibility of further partitioning into defect types was considered, but ultimately the idea was discarded. In addition to some defect types occurring rarely, which causes lacking training data, there was difficulty in categorizing images with multiple types of defect present. As such, all defective data is pooled to impart a general idea of defective features, with the downside of the network being unable to distinguish different types from each other.

The breakdown of input data also informs the choice of network topology. Using the MNNs described by Goel et al. (2021) and Chowdhury et al. (2021) as inspiration, the modular structure is composed of a routing module and a set of expert modules. Each expert module is trained on the data of a product variant, foregoing the complexity of the entire dataset. The routing module serves to identify which variant is present in an image and can then active only the relevant expert module, see Figure 8. To facilitate easy exchange of modules, each contains a fully independent CNN architecture. As such the topology fits the designation of multi-architectural modular node as outlined by Amer & Maul (2019).

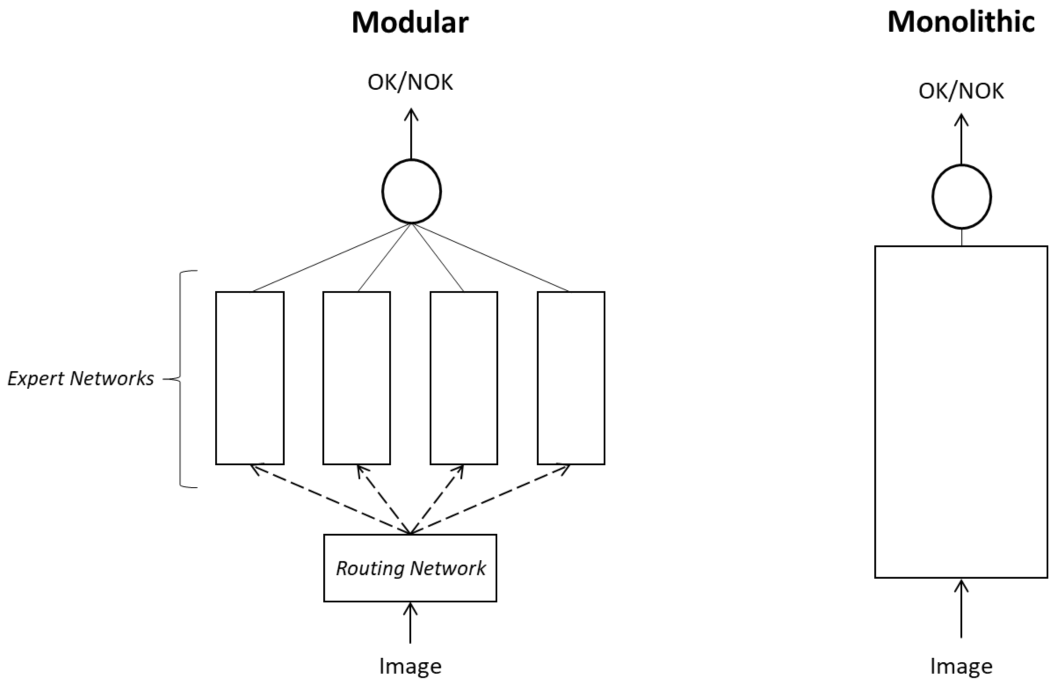


Figure 8: Visualization of the designed modular structure and a monolithic network.

As the domain is well known and the number of modules is easy to expand, no dynamic generation of the MNN is required. Instead, the formation of all connections and modules is performed manually to fit the product variants present in the dataset. As an additional consequence of the division of knowledge in the modules, the integration will be competitive by necessity, with the routing network determining which modules gets to generate an output.

This design allows not only modules to be trained separately, but further eases the addition of new modules if the product family changes. If new variants are introduced, additional modules can be trained, and the routing network retrained to recognize the new product. All other modules remain preserved, requiring no further validation of their function. In contrast, retraining a monolithic network may compromise the detection accuracy of previous variants as the weights of the network are shared.

4.3 Initial Tests

To verify the functionality of the two approaches, a public dataset of fruits and vegetables, made available by Mureșan & Oltean (2018), was used. The images in the dataset are preprocessed to exclude any background, thus isolating only the pictured object, see Figure 9. Fruits and vegetables were selected as stand-ins for different product variants and similar varieties of an object, such as two image sets of apples, were used to mimic either okay or defective products. By using pretrained networks, either standalone in the monolithic approach or as modules in the MNN, a high degree of accuracy was attained after only a short training horizon, thus verifying the functionality of both networks.

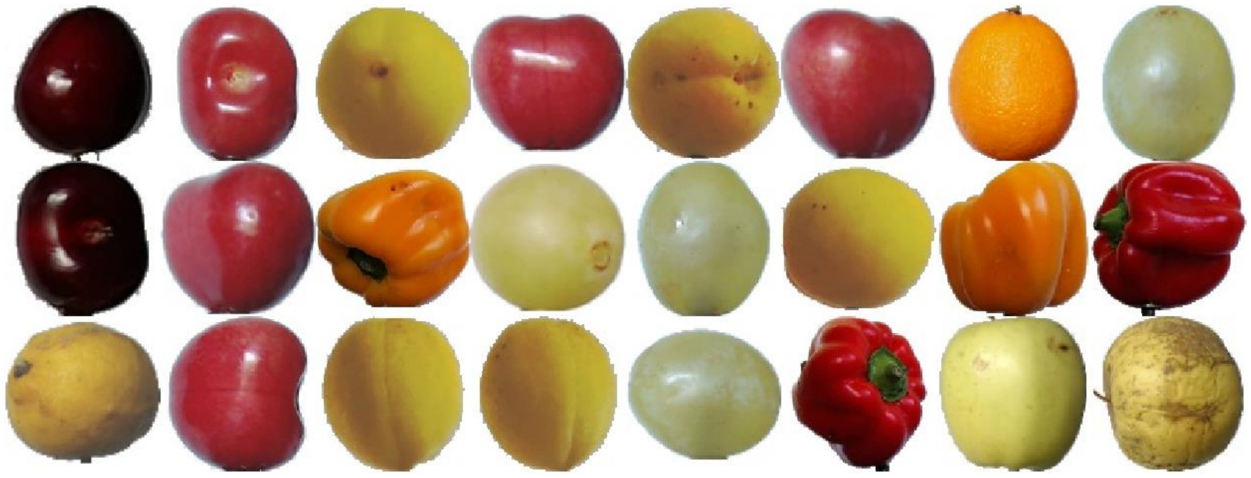


Figure 9: Example of images used to test both network approaches (Mureșan & Oltean, 2018).

5 Data Handling

5.1 On-site Data Collection & Labelling

To collect images of products, an area of AR Packaging's production facility was outfitted with a camera stand and a Logitech Brio 4K webcam capable of capturing high resolution images. To keep a consistent background in the images, a cardboard platform was assembled to place products onto. The camera was positioned with a top-down perspective of the cardboard sheet and products placed upon it. As the data collection occurred within the production environment, the camera was exposed to the same amount of natural light that would be expected in a real implementation. To generate images, products would be manually placed on the platform and a picture would be manually taken using the webcam. The product would then be rotated approximately 45º and have another picture taken, with this pattern repeating until the product had made a full rotation. Each product would thus generate eight images showing it from different angles and slightly different positions on the platform, see Figure 10, integrating a level of natural variation in how products would be presented.

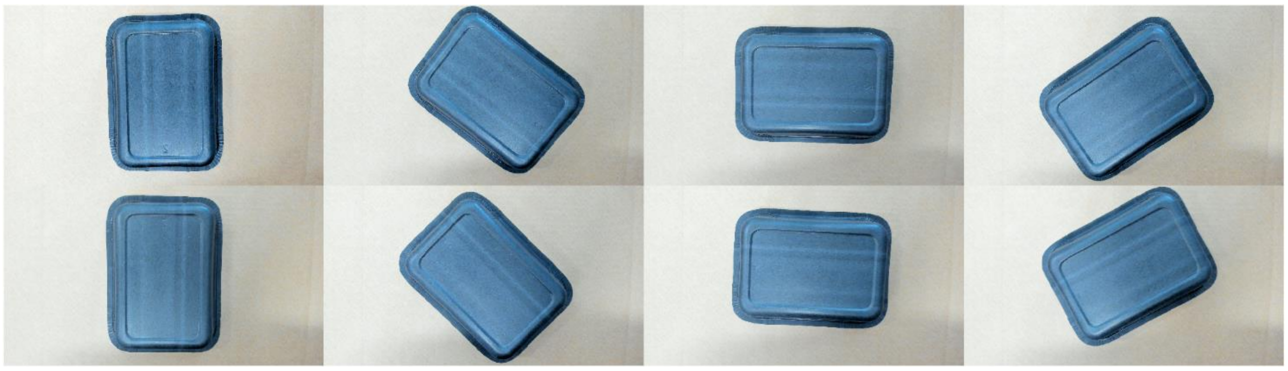


Figure 10: A set of eight images collected of the same product in different rotations.

Early in the study, automatic collection of images was considered as an alternative to this manual method. Using a camera mounted in a machine and a connected sensor which could indicate the presence of a product, product images could be captured automatically. Such an approach could drastically increase the number of images which would be feasible to collect as well as more closely resembling the operation environment of a real-world inspection system.

However, automatic image collection would introduce significant problems when labelling products as it would require each image to be inspected manually to identify defects. Not only is this a highly time-consuming process, but it also introduces the risk of the data handler mislabeling products due to inexperience, misunderstanding of quality requirements, fatigue or variability in judgement. These issues were exacerbated by the high likelihood of massive imbalance in the data occurring, as non-defective products represent a vast majority of all production.

In contrast, the manual approach to image collection which was used could exploit already existing quality inspection procedures to pre-label image samples. By gathering batches of products which have been inspected by production personnel, the quality status could be known and thus preserved as labels. The knowledge of the labels of gathered products also allowed for the number of images of each class to be roughly balanced. The increased labor intensity of image collection and lower image yield was thus justified by a significantly easier labelling process and more control over the balance of data.

5.2 Image Dataset & Filtering

At the completion of the main data collection in the production environment, a dataset containing 14577 images covering more than ten product variants had been generated. The images were captured in a resolution of 3840x2160 and have an average size of 0,69 MB. The data collection focus was on a subsection of variants, initially a set of eight (V1-V8) as seen in Figure 11.

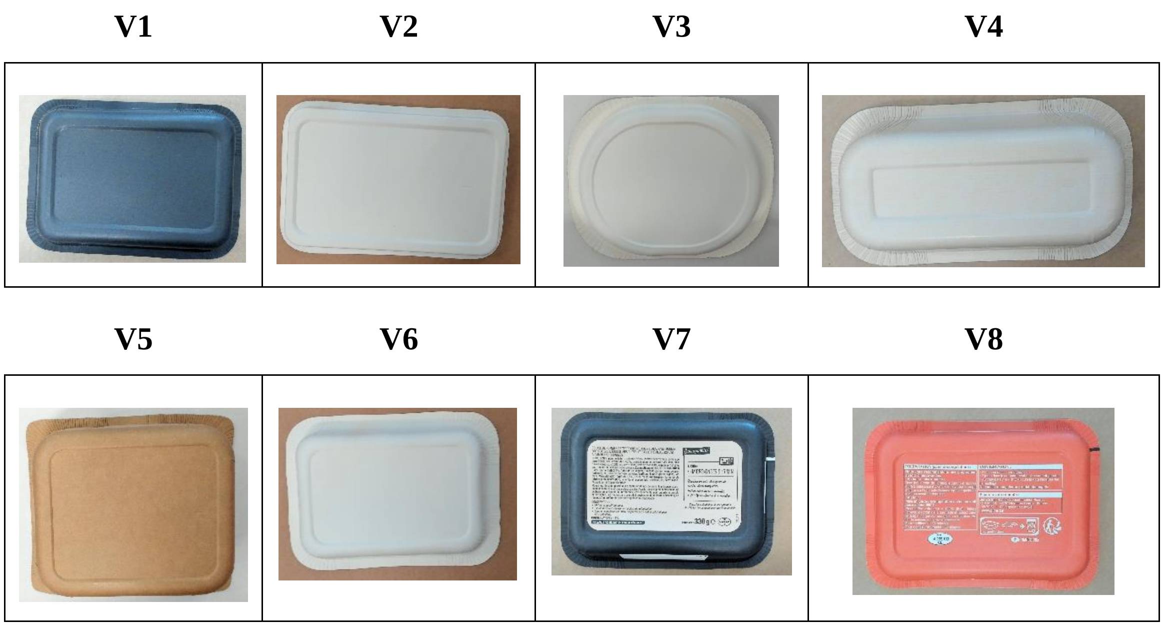


Figure 11: All eight variants which were initially included in the study.

Two variants, V3 and V5, were later excluded from the study, reducing the set to only six types. This is due to circumstances such as production planning or variant volume limiting the number of images which could feasibly be collected in a timely manner. Other issues were variant specific, such as the high profile of V5 obscuring most of the product surface when seen from a top-down perspective, see Figure 12. This makes single camera inspection unreliable as other angles would be necessary to properly evaluate the quality. Another contributing factor to reducing the number of variants in the study was the increased time and resources it would have taken to collect further images and train networks on additional image datasets.

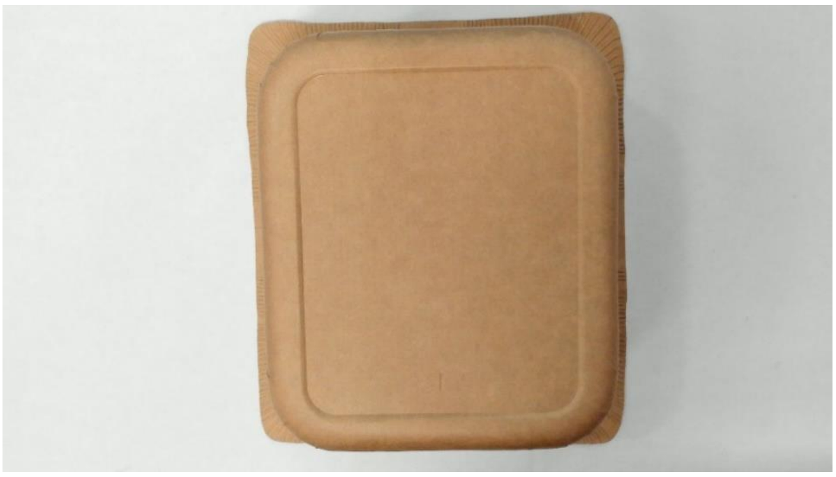


Figure 12: The high profile of V5 obscures the sides of the product.

The dataset also had to undergo filtering to verify the quality of images before being used in network training. A series of properties, or lack thereof, were used as grounds for excluding an image from the filtered dataset, such as hands showing in the picture, bad focus, the product being substantially outside of frame or defective images where the defect was not visible. The largest set of images filtered occurred for V7, where it was found post facto that a large chunk of OK samples showed a similar, but different, product variant. As such, almost 500 images had to be removed from the dataset. No filtering was done for V3 and V5 as they were excluded from the study at this point. The distribution of images across variant and class, as well as the results of the filtering process, can be seen in Table 1.

Table 1: Breakdown of the number of images for each product variant before and after filtering.

	Before filtering			After filtering		
Variant	**OK**	**NOK**	**Total**	**OK**	**NOK**	**Total**
V1	304	911	1215	304	882	1186
V2	1173	1039	2212	1173	1003	2176
V3	n/a	n/a	526	n/a	n/a	n/a
V4	634	601	1235	634	592	1226
V5	65	324	389	n/a	n/a	n/a
V6	1624	1775	3399	1622	1764	3386
V7	1355	1484	2839	804	1387	2191
V8	579	1205	1784	575	993	1568
Sum	5734	7339	13 599	5112	6621	11 733

While a roughly equal image balance was achieved for V2, V4 and V6, issues arose in the balance of V1, V7 and V8. The above-mentioned exclusion of a significant number of V7 images is to blame for the bad balance in that specific case. Regarding the lack of OK samples for V1 and V8, insufficient communication was the culprit, as the planned production batches of these variants were finished before enough images were collected. As such, additional products were unavailable during the allotted time for data collection.

However, unlike typical issues of imbalance in quality inspection applications, defective samples are overrepresented in these three variants. This is much more desirable than OK samples making up a disproportionate part of the dataset, as these tend to be more uniform in their presentation and thus represent a set of features which are easier to learn. Defective products on the other hand tend to show a much wider range of possible variation and type, making those features more challenging to represent. As such, even with almost a 3:1 imbalance in the worst case, these variants were still included in the study.

5.3 Defect Types

Within the collected data a multitude of different defect types are present. Even defects which could be classified as the same type differ in their presentation within and between product variants. As the data partition did not concern identifying different kinds of defects, a full cataloging of the difference was not attempted, but some discussion of the subject is essential to understand both the complexity and limitations of the dataset.

One important point to understand is that the dataset represents a snapshot of selected defects over a limited time period. As such, even if some defect type could occur in more variants, there may only be data for the defect occurring in one variant. This is the case for misaligned print, which only occurred in V7. A similar case is the presence of so-called “washlines” only in images of V1 and V8. Over a long enough horizon of data collection there would thus be a much better representation of defect types across the variants included in the study.

This issue is exacerbated by the fact the defect often come in batches, with several products suffering from the same issue but in different degrees, see Figure 13. The occurrence of a series of defects during the data collection period may thus provide a good representation of a defect type for one variant but be totally lacking in any of the other. A selection of defect types is discussed in the rest of the section.

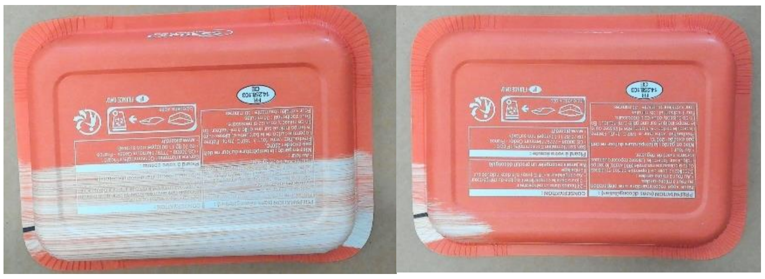


Figure 13: Two products produced close in time may suffer the same defect in varying degrees.

Deformation: In all likelihood the most commonly occurring defect, being well represented in the data of each product variant. Deformations of a product geometry range from quite extreme to more subtle damage. A comparison between two deformed products can be seen in Figure 14, where the left one is severely damaged and the right one only suffering from a dent on one side. However, due to the visually striking character of most deformation and the mechanical issues which often coincide with their occurrence, these defects are often very easily identified by inspection personnel.

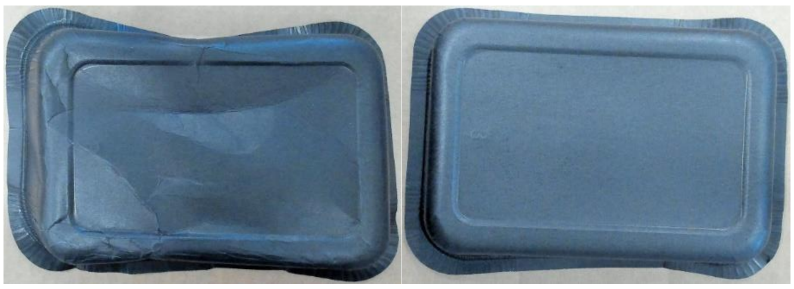


Figure 14: Two varying degrees of deformation.

Double-pressing: Double-pressing occurs when twice as much material is fed to the pressing machine, producing a product with two layers. The features of the product geometry are often less pronounced as there is more resistance when pressing it, see Figure 15. Much more uncommon than simple deformation, but occurrences are noted in all six product variants.

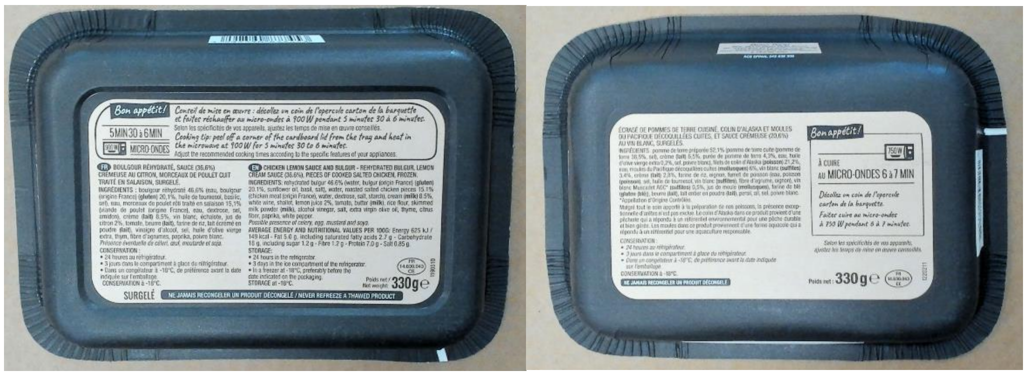


Figure 15: Comparison of a single-pressed product on the left and a double-pressed product on the right.

Misaligned Print: As the name indicates, this defect takes place when printed material is misaligned when fed into the pressing machine. The geometry of the product is not necessarily compromised, but printed features such as barcodes or cooking instruction may be compromised, see Figure 16. Data for this defect was only collected for V7 but could occur for any variant with a printed surface.

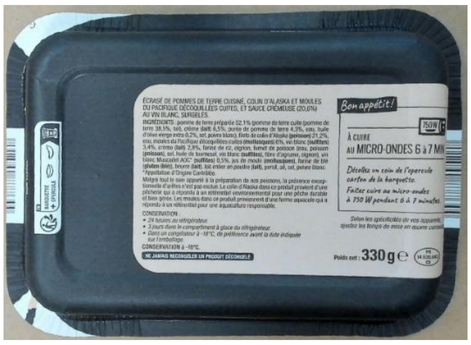


Figure 16: Example of product with misaligned print.

Tearing: Tears are often found in the corners of products or occur when there is significant deformation but are rarer than deformation itself, see Figure 17. In some variants, like V4, tears can occur on the side of the product making it hard to show from a top-down perspective.

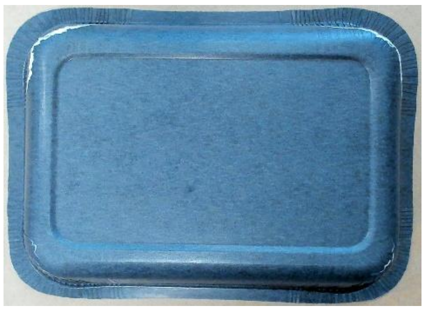


Figure 17: Example of a product with tearing in the corners.

Surface Damage: Surface damage is here used as a catch-all term for any defect which is visible on the surface of the product but does not necessarily deform the geometry or occur as the result of misaligned print or tearing. Appearance varies depending on source of surface damage, see Figure 18.

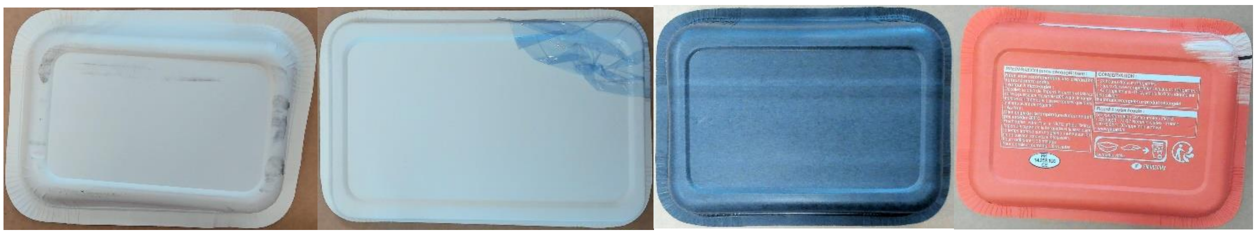


Figure 18: Different kinds of surface damage showing dirt, tape, "washlines" and paint on the products.

6 Image Processing

6.1 Segmentation

As mentioned in the section *5.2 Image Dataset & Filtering*, the images in the dataset have a resolution of 3840x2160 which is much larger than the typical input size of CNNs. For example, Krizhevsky et al. (2012) rescaled images used in AlexNet to 224x224. Resizing images before classification thus becomes a required pre-processing step, but simply reducing the size of image itself risks removing significant information. This is especially important in networks performing quality inspection, as subtle defect may only occupy a small patch of the image which could become impossible to differentiate with the loss of resolution.

To address this loss of detail, a more advanced image pre-processing procedure was developed to minimize the resolution of the input image while keeping the entire product in frame before the image is resized to fit the classification network. Images are first subjected to segmentation to identify the location of the product in an image, see Figure 19. This is accomplished by using *k*-means clustering to automatically differentiate the product from the cardboard background. Any gaps within the cluster are also filled to create a solid shape which can then be represented in a binary mask, where the location of the product is marked by 1s, and the background is marked by 0s. The code used was generated through the Image Segmenter App in Matlab, a part of the Image Processing Toolbox.

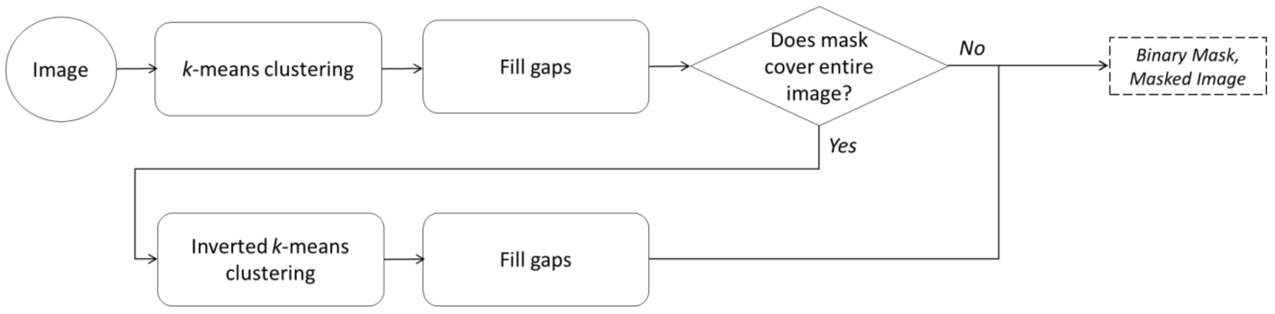


Image
k-means clustering
Fill gaps
Does mask cover entire image?
No
Binary Mask, Masked Image
Yes
Inverted k-means clustering
Fill gaps

Figure 19: Segmentation algorithm workflow.

If the algorithm misidentifies the background as the product, the function which fills any gaps will cause the entire image to be covered by the binary mask leaving no information about the location of the product. To prevent this from happening, the procedure is repeated if the binary mask covers the entire image but inverts the mask before the gaps are filled. Upon the completion of the procedure a binary mask corresponding to the product within the image is attained, see Figure 20.

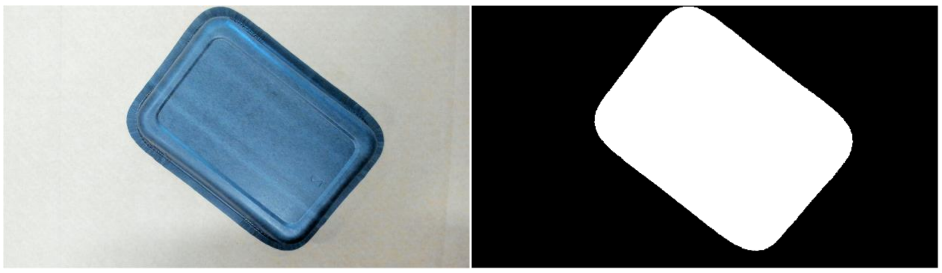


Figure 20: Input image and the binary mask generated from the image segmentation.

6.2 Orientation

By localising the highest, lowest, right-most and left-most points of the binary mask created by the segmentation algorithm, the original image can be cropped to remove as much background from the image as possible. However, in some cases this approach would still include large sections of the background as seen Figure 21. To further preserve the features of the product, the second major image pre-processing algorithm seeks to maximize the portion of the cropped image containing the binary mask by orienting the product horizontally or vertically.

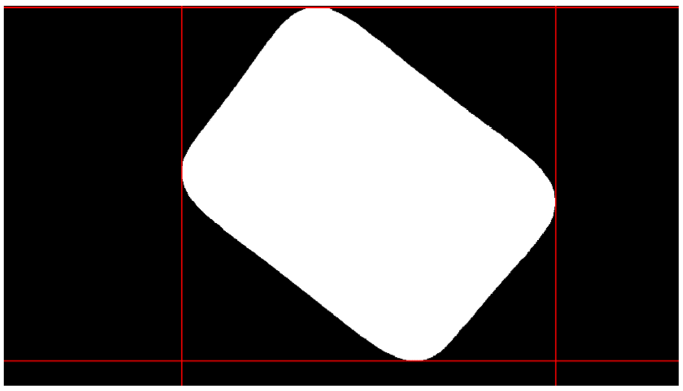


Figure 21: Binary mask allows areas around the product to be cropped.

To accomplish this, the binary mask is iteratively rotated while calculating the resolution between the edge points, see Figure 22 and Figure 23. If the first rotation is found to increase the resolution, rotation is attempted in the opposite direction. As long as these points define the resolution the entire product will be within the new image, so by iterating until the smallest resolution is found the portion of the image containing the product will be maximized. Intuitively for the rectangular product variants included in the study, the minimum resolution is obtained when they are either horizontally or vertically aligned.

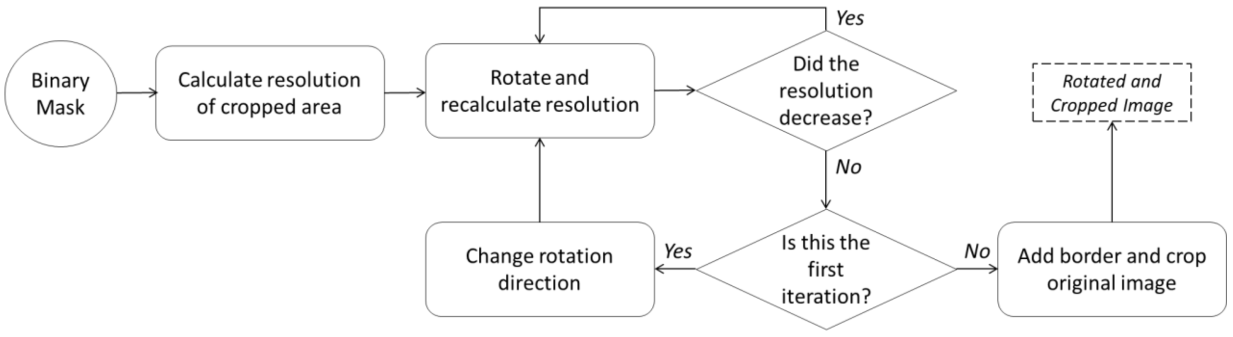

Binary Mask
Calculate resolution of cropped area
Rotate and recalculate resolution
Yes
Did the resolution decrease?
No
Rotated and Cropped Image
Change rotation direction
Yes
Is this the first iteration?
No
Add border and crop original image

Figure 22: Orientation algorithm workflow

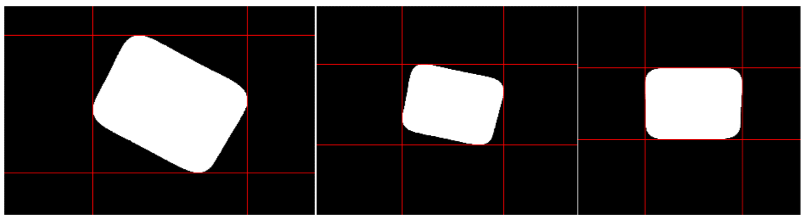


Figure 23: Binary mask at 10, 26 and 39 degrees of rotation from initial position.

Once further iterations are unable to minimize the resolution, the final area around the product is calculated. The rotation is applied to the original image, as the binary mask was simply used to represent location. Before cropping the image, a 10-pixel border is added in every direction to ensure the edges of the product are properly visible. If the new image includes area which were not present in the original image, the new space will be black. Given that the process was completed successfully, resulting images will be aligned either vertically or horizontally. In contrast to the segmentation part of the image processing procedure, this code was manually written specifically for this task and was not generated using any app or toolbox.

By using this procedure, the resolution of the example shown in this chapter was reduced by almost 67% while not losing any product details, see Figure 24. Similar results are true for all other successfully processed images.

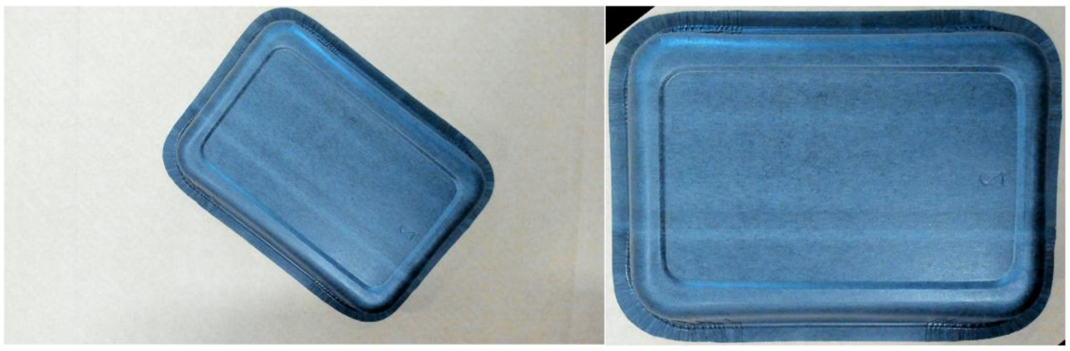


Figure 24: Comparison of the original image and the processed image.

6.3 Image Processing Performance

While the image processing algorithm manages to isolate products within an image for the most part, there are cases where it fails to achieve this task. This can result in processed images failing to align the product or failing to crop one side of the image, but more severe issues where the processing completely fails were also recorded. However, as the algorithm is currently constituted it is impossible for the product to be excluded from an image, making it possible to classify images even if they fail to be properly aligned and cropped.

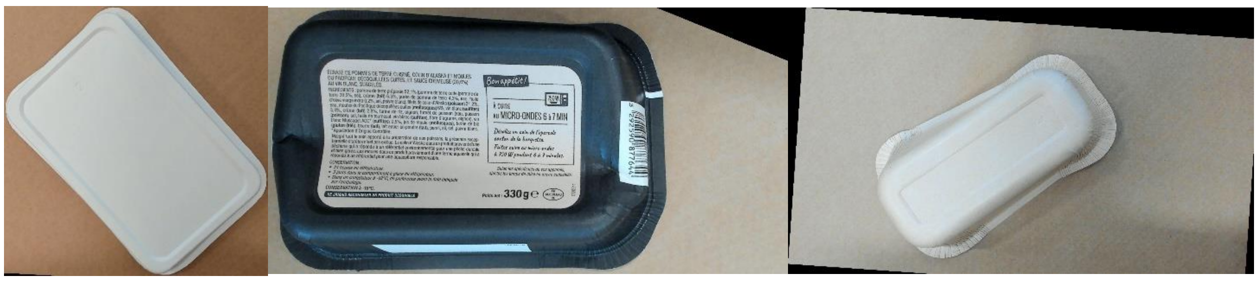


Figure 25: Examples of images which either partially or entirely failed to be processed.

To gain an overview of how well the processing worked, the occurrence of major processing failures for each variant was investigated. This evaluation was conducted manually and relied on subjective judgements of what was deemed a failure. As a reference, partially successful processing like the two left-most images in Figure 25 were not counted as failures, while the right-most image would have been. Given these rough definitions, the processing success rate was estimated as seen in Figure 26. Generally, individual images which failed to process may have been overexposed to light which then compromised the segmentation of the image and thus subsequent steps.

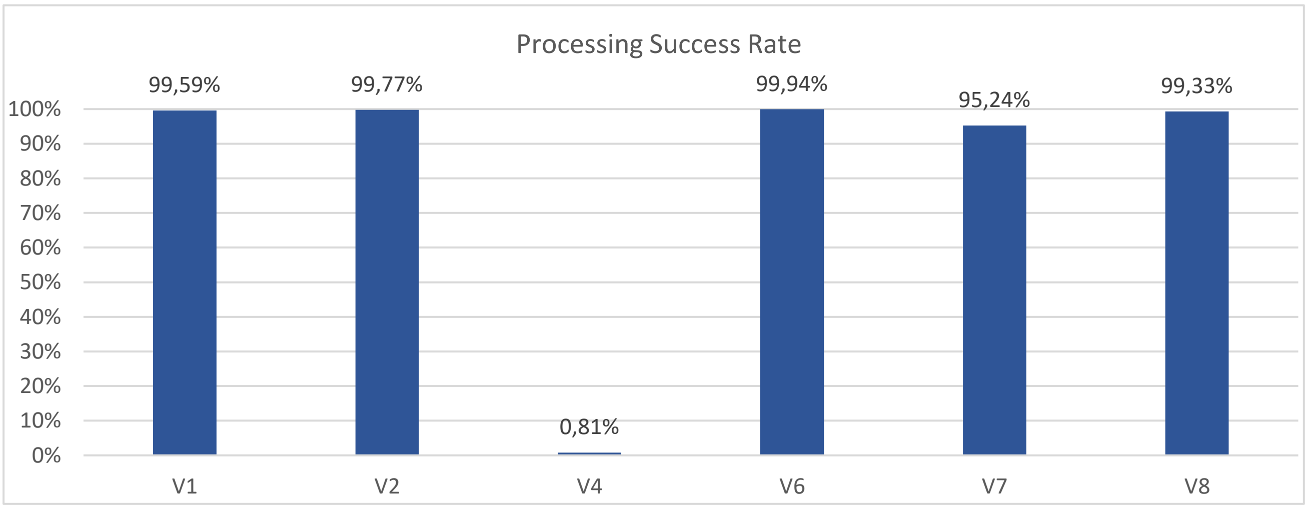


Figure 26: Rate of successful processing of images per product variant.

However, it is also obvious that a large part of the failures of the algorithm are unrelated to temporary environmental noise and are rather variant specific. Notable here is V7 with a success rate notably below most other variants, and V2 for which the algorithm totally fails to consistently process images. The working theory for this fact is that the internal print of V7 and the entire surface of V2 resemble the cardboard background to such an extent that clustering routinely fails, though at much higher rates for V2.

Some attempts were made to correct this fact, but due to time constraints the current performance was deemed good enough to produce training data. As mentioned before, even in the most severe failures of image processing the image clearly portrays the products, though at a lower resolution than properly processed images. Since both the monolithic and modular networks approaches will be trained and tested on the same data, any lingering issues caused by these shortcomings will not compromise the overall comparison between the techniques.

As for the speed of the algorithm, it achieves an average processing time of 2,17 seconds when tested on 792 samples from each of the six studied variants. It should be noted that the time can vary depending on the rotation of the product or if an inverted segmentation must be performed. The spread of the processing time is visualized in Figure 27 below.

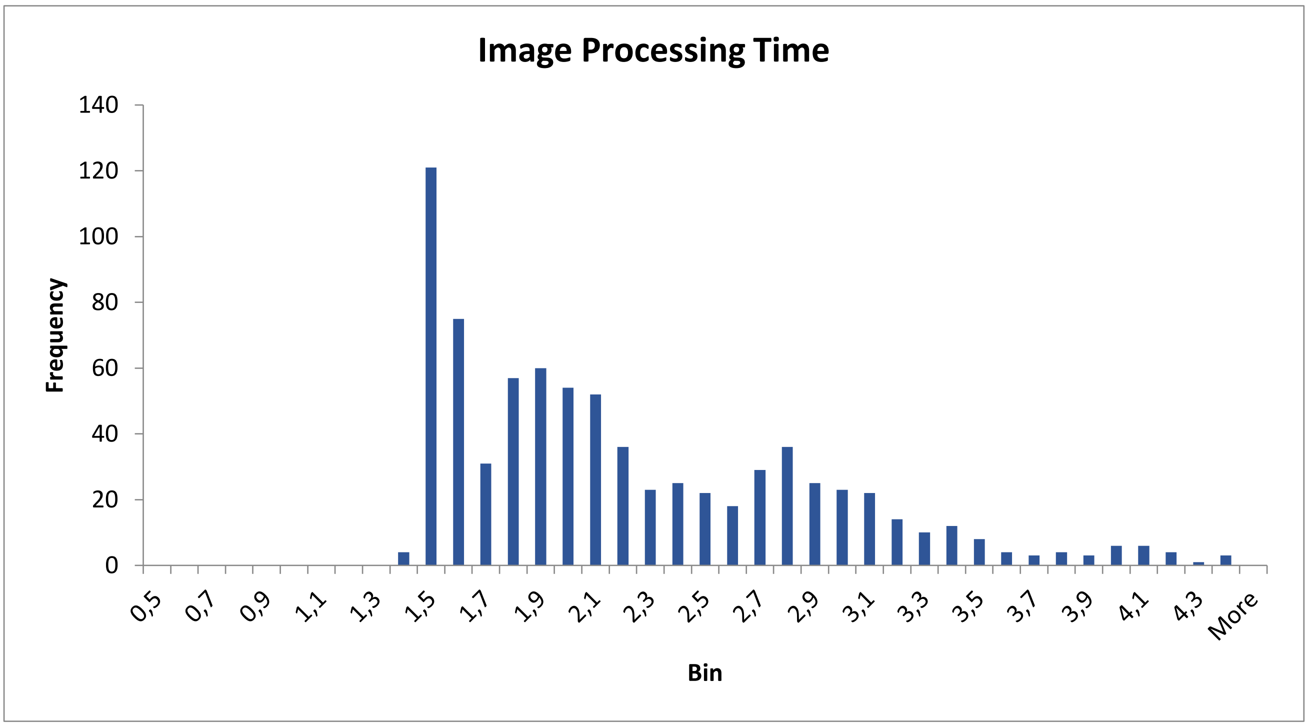


Figure 27: Histogram showing the image processing time for 792 samples.

7 Network Training

7.1 Data Preparation

Properly training the networks in the study is essential to be able to perform a useful and valid comparison of the two competing approaches. As can be seen in Figure 28, the process of training a network can be divided into several stages. The first two strictly deal with acquiring and preparing input data for the training. Using the image dataset generated in chapter *5 Data Handling*, images can be extracted based on the needs of a specific network. Before being used in training however, these were all run through the image processing algorithm discussed in chapter *6 Image Processing*. While it is possible to integrate the algorithm into the training process to avoid having to store all the processed images, the time lost for processing was deemed a far greater cost than additional storage.

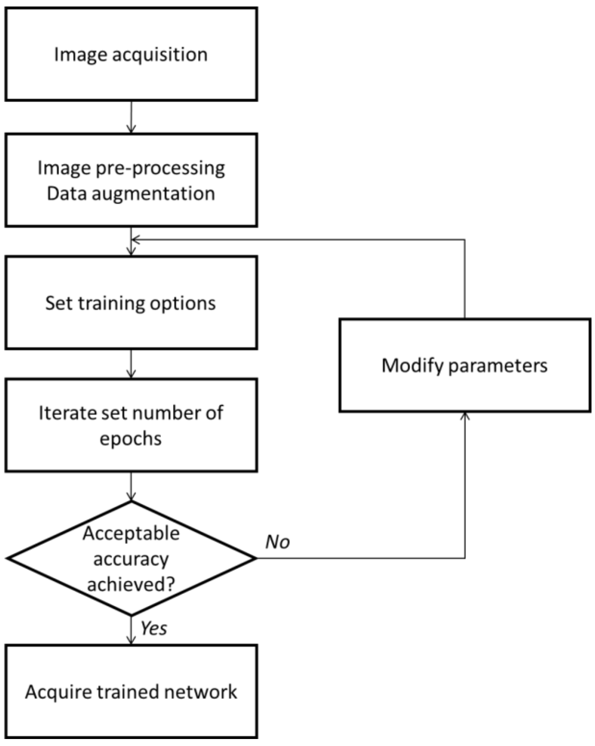


Figure 28: Stages in the training process, adapted for the project from Zheng et al. (2021).

The processed images are then subjected to data augmentation before being used to train the networks. To ensure a fair comparison between networks the parameters for data augmentation were kept static in every training sequence. The augmented options which were found to yield the best results were randomly reflecting an image vertically and/or horizontally, randomly rescaling an image by 10% and randomly translating an image by a maximum of 30 pixels in any direction. These added data features not only help the networks deal with variation in the input data, but also prevents overfitting. Training was performed within the framework provided by the Deep Learning Toolbox in Matlab, establishing easy control over training options and augmentation settings. Subsequent pretrained CNNs used in the study were also acquired as add-ons to the Deep Learning Toolbox.

7.2 Modular Network Training

As outlined in the section *4.2 Modular Design*, the MNN is made up of two constituent parts, those being a routing module and a series of expert modules. The CNN architecture ShuffleNet, first introduced by X. Zhang et al. (2018), was selected as a good candidate for the modular structure envisioned. It uses an input resolution of 224x224 and was designed to be computationally efficient, managing to be roughly 13 times faster than AlexNet while still achieve similar classification accuracy. The small size of the network also lends itself well to a modular structure, with a single instance of the network taking no more than a few MB of storage compared to the hundreds necessary to store larges architectures. Following the example of other studies, such as Ferguson et al. (2018), transfer learning of a version of ShuffleNet pretrained on the ImageNet classification dataset was used instead of randomly initializing the weights.

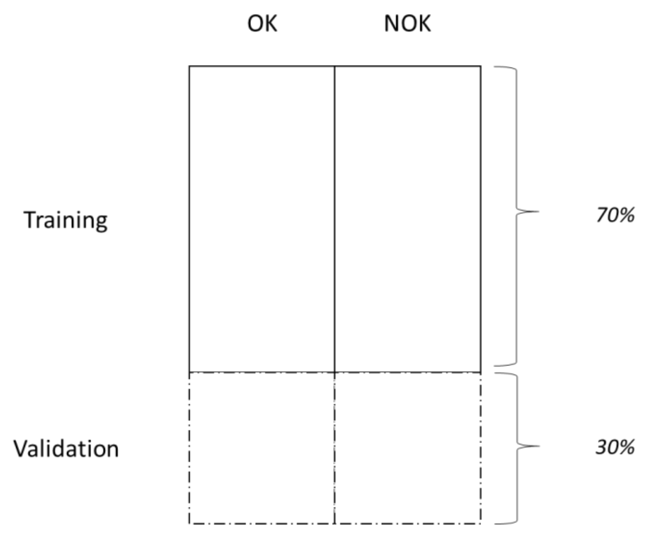


Figure 29: Division of image data for training and validation.

The data for each variant was divided into a training set, containing 70% of OK and NOK samples, and a validation set, containing the last 30% of each class, see Figure 29. After several tests, it was found that 50 epochs, that is iterating the training process over the entire training set 50 times, was roughly the length of training required for the expert modules to achieve acceptable accuracy while avoiding the risk of overfitting. Training was conducted in batches of 10, and the convergence of the validation set checked every epoch.

In the first iteration of the MNN, an expert module was trained for V1, V2, V4, V6 and V7, resulting in a set of five networks learning the feature of just one variant each. The last variant, V8, was left unused as it would be introduced in the retraining stage to simulate an addition to the product family.

A custom dataset was built for the routing module, being trained on each of the five selected variants instead of a division of defective and non-defective. This dataset thus contained a mixture of OK and NOK samples within each class, but a similar division of 70% training and 30% validation was used. Since the complexity of differentiating product variants was deemed much easier than detecting defects, and due to the much larger dataset the routing network had to work with, a larger batch size of 32 along with a shorter training horizon of 30 epochs.

7.3 Monolithic Network Training

For the monolithic network ResNet-101 was chosen as the CNN architecture. The ResNet series of CNNs, presented by He et al. in 2016, have proven to be very popular among quality inspection applications. For this study, the 101-layer variant was chosen as it represents a much deeper and larger network than ShuffleNet, containing about 30 times more parameters and taking approximately 50 times more space to store. This contrast between the networks fit the narrative previously discussed regarding modular and monolithic structures. Another upside of using ResNet-101 is that it shares an input resolution of 224x224 with ShuffleNet, making it slightly easier to interchange processed data between the approaches. Like the network modules previously trained, the ResNet-101 model in use was pretrained on the ImageNet classification dataset allowing for the use of transfer learning.

To ensure a valid comparison of the different approaches, the training parameters used to train the expert modules were also used by the monolithic network. Likewise, the datasets used to train and validate the first five expert modules were extracted and combined to be used in the training of the monolithic network. Thus, the exact same images are used for both approaches. However, since the output of the network is only concerned with detecting defects, all NOK and OK samples are pooled to form two classes.

7.4 Retraining Network Approaches

To simulate the introduction of a new product to the production line, both approaches were retrained with the data for V8. Updating the MNN requires only another expert module to be trained on the new data and the routing module to be retrained with another variant category. This was accomplished using the exact same procedure as described previously, except for the routing module transfer learning using the network previously trained on the first five variants instead of the ShuffleNet model trained on the ImageNet classification dataset.

The retraining of the monolithic approach proved a bit more troublesome. Using the same method as previously described, the data used to train the additional module was added to the dataset of the first five variants, once again ensuring that both approaches had access to the same images. Repeating the method used for retraining the routing module, the monolithic network previously trained on five variants was used as the starting point of training the network with the added data.

To shorten the time required to retrain the network, the number of epochs was reduced to 20 while all other settings remained the same. Achieving unsatisfactory results, especially for V1, another attempt was made with the full 50 epochs. Strangely this further reduced the accuracy for some variants, with a massive reduction for V1. Finally, a training using only 10 epochs was attempted. This iteration seemed to result in the best overall accuracy and avoided the huge drop-off in accuracy for V1. The difference in accuracy can be seen in Figure 30, where divergence from the 20-epoch retraining results in the subsequent tests is visualized. The cause for this phenomenon is currently unknown but it highlights the importance of properly validating retrained networks.

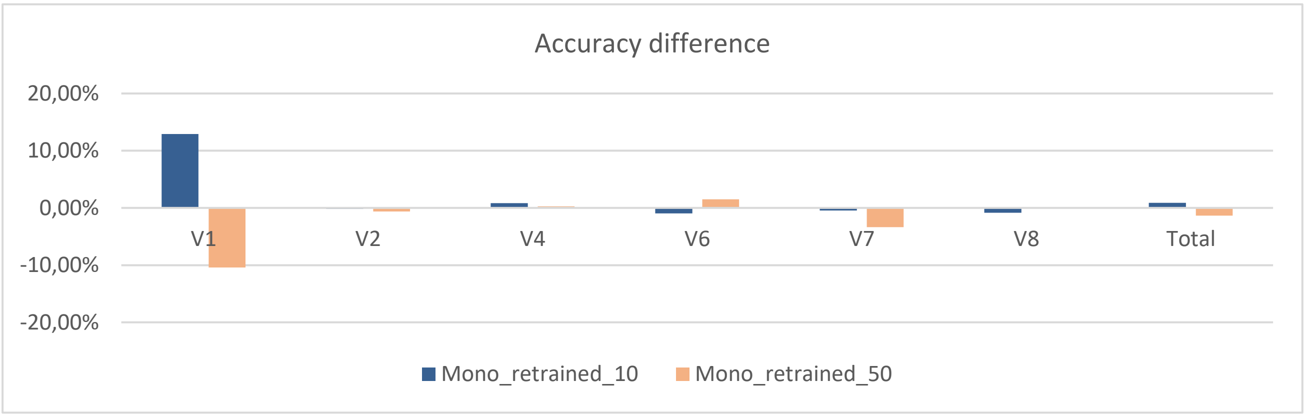


Figure 30: Accuracy divergence from 20-epoch retraining when using 10 or 50 epochs instead.

8 Results

8.1 Classification Accuracy

To test the classification accuracy of each of the trained networks, all the base validation images were gathered and classified. Since these images had not undergone any augmentation, the generated results deviated somewhat from the validation accuracy found in during the training process. The results from these tests are shown in Table 2 below, with both variant specific performance and the overall accuracy. Confusion matrices can be seen in Appendix A:.

Table 2: Classification accuracy for each of the trained networks.

	MNN	*MNN (retrained)*	*Mono*	*Mono (retrained)*
V1	93,54%	93,54%	98,03%	98,88%
V2	97,55%	97,55%	99,23%	98,62%
V4	98,64%	98,64%	99,46%	98,91%
V6	96,95%	96,95%	98,33%	95,28%
V7	98,78%	98,78%	99,39%	98,63%
V8	n/a	95,11%	n/a	98,72%
Overall	97,28%	96,99%	98,85%	97,73%

The best overall performance is found in the two monolithic networks, which clearly outperform the two MNNs. Notable is the fact that since even the retrained routing network can differentiate the variants at practically 100% accuracy, the accuracy of the first five modules remains unchanged in the retrained MNN. This is not true for the monolithic networks however, as the retraining of the network affected the accuracy of each variant in some way. Laying this unreliability aside, there is no question that a greater accuracy is attained when pooling the images in the dataset compared to when they are partitioned in smaller modules. The idea of reducing complexity of a task may be a sound one, as discussed previously, but there is demonstrably much to gain by utilizing more general features in this case. This idea was further cemented by training a monolithic network using ShuffleNet on the first five variants. Even when using only a fifth of the parameters this network managed outperform the modular approach in overall accuracy, achieving 97,51%.

8.2 Classification Speed

During the tests which recorded the classification accuracy of the networks, the speed of classification was logged as well. These measurements do not include any image pre-processing, but instead measure the time it takes for the image to pass through either the modular or monolithic networks. The MNN activates two modules, first the routing module to detect which variant is present in the image and then the corresponding expert module to classify the quality. The monolithic network needs only pass the image through itself and generate an output. As can be seen in Table 3, the monolithic networks are somewhat faster than the modular approach. This is contrary to the literature on the subject previously cited in this study but can be the result of inefficiencies in how the two modules are called and executed. However, both approaches achieve response times which are well below the production rates expected, making the classification speed suitable for real-time application.

Table 3: Average classification speed for each of the trained networks.

	MNN	*MNN (retrained)*	*Mono*	*Mono (retrained)*
Average Speed (s)	0,0273	0,0256	0,0209	0,0210

8.3 Network Size

The overall size of the modular networks is substantially smaller, taking up roughly 20 MB of storage compared to the over 150 MB of the monolithic networks, see Table 4. These results were expected given the choice of network architectures but are necessary to note as some applications may have limited storage capacities making the MNN more suitable. However, as more modules are added, the MNNs will grow while the monolithic network will stay approximately the same size.

Table 4: Size table for each of the trained networks.

	MNN	*MNN (retrained)*	*Mono*	*Mono (retrained)*
MBs	19.839	23.149	155.145	155.147

8.4 Network Training and Retraining Time

The time spent training networks is of course determined by the user in some ways, with training parameters such as number of epochs and image batch size affecting how quickly the process is concluded. Consequently, the results visible in Figure 31 should primarily be understood strictly as the outcome of this specific case study, even more so than the other measurements of network performance.

As expected, even when using the same training data and parameters, individual modules train much faster than the monolithic networks. The most computationally expensive module to train in the study was the routing network, owing to the large size of the dataset it was trained on. It is possible that even fewer epochs could be used for the routing module and still maintain accuracy, though no test was conducted to confirm this idea.

Of more interest is that the retraining time for the monolithic network was surprisingly low. As discussed in *7.4 Retraining Network Approaches*, fewer epochs of retraining yielded better results than a longer training horizon. As the underlying causes of this remain unexplored, on can only state that this study does not find a modular approach easier or quicker to retrain upon the addition of more product variants.

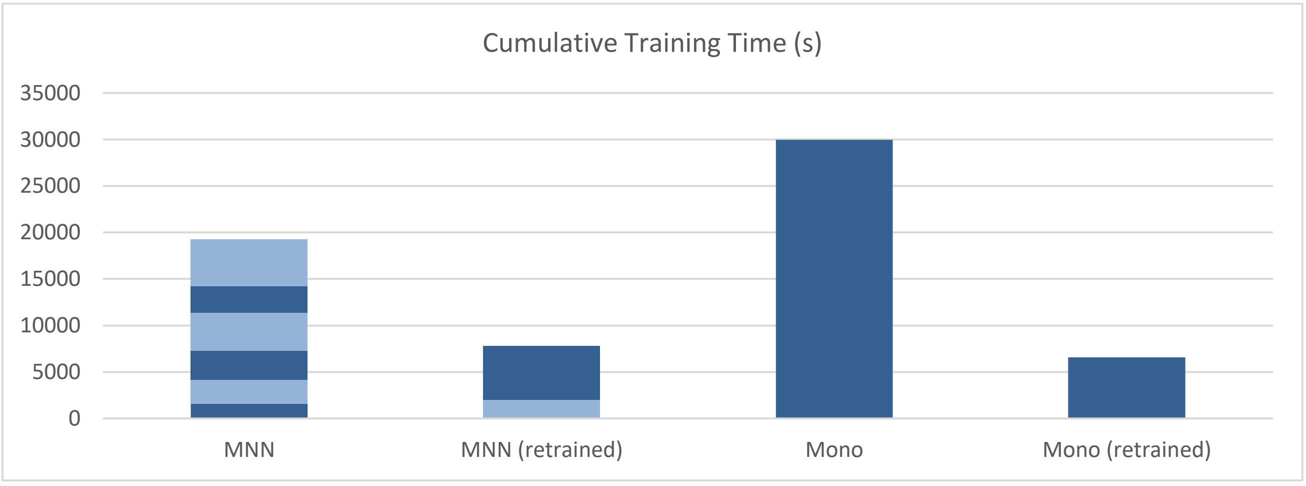


Figure 31: Cumulative training times for each network, with segments representing network modules.

9 Discussion

9.1 Complexity and Generalizability

The genesis of interest in studying a modular approach of deep learning quality inspection was the supposition that product families which introduce increased complexity to a classification problem through the presence of various colors, prints, geometries and so on, could be more easily handled by dividing the inspection task into smaller sub-tasks. More specifically, it was believed that this reduction of complexity may overshadow any general understanding of the problem attained from pooling the image data. The most immediate manifestation of this would have been a higher classification accuracy in individual modules than a large network could achieve.

This study found the opposite. Before retraining the networks using the data from another variant, the monolithic network outperformed the modular alternative both in overall predictive accuracy and for each individual product variant. While this study does not invalidate a modular structure for quality inspection networks, it should serve to advance critical examination of actual problem complexity. As this case shows, any gains made through reducing the overall difficulty of classifying a single variant can be outweighed by understanding the features of defective products more generally. Restating this result from another perspective, the modular approach within this case requires more image data from each variant to reach comparable precision with the monolithic approach. Given that acquisition, labelling and processing image data forms a large part of the labor necessary to produce a trained neural network, increasing that burden is a bad proposition.

9.2 Reusability of Modules

One point where the modular approach showed a clearly advantageous feature was the fact that the retrained MNN did not suffer any change in accuracy, in contrast to the retrained monolithic networks which were significantly affected in all categories of accuracy. This will hold true as long as the routing module is able to maintain accuracy, thus activating the correct expert module. Additionally, in a final implementation of the system, a neural network may not be needed to perform the routing. The part of the structure routing images could just as easily be replaced by a system which reads a barcode, senses an RFID tag, receives the variant type electronically from a PLC or relies on a human operator to manually enter the current batch of variants. While this ability wasn't utilized in the study, that level of control and reliability when changing modules could be highly sought after in other applications.

9.3 Implications for Sustainability

Throughout the study, questions of sustainability have informed the work in several ways. AR Packaging is itself in an ongoing process of lowering or eliminating plastic content in their products where possible in favor of renewable alternatives like carton board. Despite making significant progress in this area, the company predicts that the process of rethinking product designs to decrease environmental impact will be a continuous project over the foreseeable future. This fact makes the adaptability of quality inspection systems a vital aspect, as inspection software must keep pace with changing product designs and materials. Thus, by studying how to retrain deep learning applications quickly and reliably within quality control, the transformation to more sustainable products can be supported.

Despite being outperformed in many metrics within this specific case, modular designs of neural networks may offer more energy efficient and less computationally expensive alternatives to monolithic approaches. The idea of minimizing redundant or irrelevant computation for classification tasks is a sound one, and the potential of being able to extract specific modules and reuse in other applications may remove the need to even train a network in some cases. As the research within the field progresses, these questions of energy efficiency and reusability will be important to discuss.

There are also implications for more economic and social factors pertaining to sustainability. Technologies such as automatic inspection systems are often developed to replace, or at least minimize the need for, human labor. Positive aspects to this process certainly exist, as performing visual inspections in production facilities is usually monotonous and uninteresting work, and by reducing the amount of time spent doing such tasks frees us to engage in more fulfilling and rewarding activities while still maintaining the same levels of quality and production as before. A cheaper, less labor intensive and more productive manufacturing process is certainly a good outcome for economic sustainability.

However, as history has shown from the mechanization of agriculture to automation in the automotive industry, workers who are displaced by technological progress often lack appreciation for such macro-perspectives when their own standard of living is directly connected to labor which is now in less demand. Monotonous as the work may be, it is often performed by a knowledgeable unionized workforce which not only form a social base of their respective communities, but also an economic one. Therefore, to avoid economic shocks, efforts to counterbalance reduced labor demand should be considered to safeguard economic and social stability of affected groups.

10 Conclusions

This case study sought to compare a modular and monolithic network approach to quality inspection using data from a real production environment where changes to the product family may occur. Using guidelines found in contemporary scientific literature, a modular network was designed to correspond to the more widely applied monolithic structure. An image dataset was gathered from the partner company, from which data on six product variants was chosen to be processed and used to train and test the two competing network structures. To simulate an addition to the product family, both networks were first trained on only five variants, and then retrained using data from the sixth.

The monolithic networks outperformed the modular networks in overall accuracy, with the initial monolithic network achieving higher precision in all five trained variants compared to both modular networks. The retrained monolithic network experienced some fluctuation in its accuracy compared to the initial results, which was not observed in the retrained modular network. The classification speed of the modular networks was also measured to be slower than the two monolithic networks, which both achieved equivalent speeds. The modular networks were smaller than their monolithic counterparts, but the retrained network grew in size due to an added module while the retrained monolithic network experienced no significant change in size. The cumulative time spent training the initial modular network was roughly 29% shorter than the initial monolithic network, but the retraining time for both networks was approximately equivalent.

10.1 Future Work

Conducting a follow-up study in a more controlled setting would be highly interesting. Several compromises had to be made to fit the specific conditions of the case, such as data collection being subjected to both production schedules and stochastic quality outcomes. If one tired to procedurally generate image data instead of relying on manual collection and labelling, much of the limitations and noise of a real-world manufacturing plant could be eliminated. Thus, a more general and in-depth understanding of the interaction between complexity and generalizability could be attained than what is possible from an isolated case study.

An aspect of the study which needs drastic improvement is the image processing algorithm. While the speed and partial failure to process images did not undermine the comparison between the two studied approaches, these factors would need to be more thoroughly addressed in an actual implementation of the system.

11 References

Alencastre-Miranda, M., Johnson, R. M., & Krebs, H. I. (2021). Convolutional Neural Networks and Transfer Learning for Quality Inspection of Different Sugarcane Varieties. *IEEE Transactions on Industrial Informatics*, *17*(2), 787–794. https://doi.org/10.1109/TII.2020.2992229

Alom, Md. Z., Taha, T., Yakopcic, C., Westberg, S., Hasan, M., Esesn, B., Awwal, A., & Asari, V. (2018). *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*. https://arxiv.org/abs/1803.01164v2

Amer, M., & Maul, T. (2019). A review of modularization techniques in artificial neural networks. *Artificial Intelligence Review*, *52*(1), 527–561. https://doi.org/10.1007/s10462-019-09706-7

Anand, S., & Priya, L. (2020). *A Guide for Machine Vision in Quality Control*. CRC Press.

Ansari, Z., & Seyyedsalehi, S. A. (2017). Toward growing modular deep neural networks for continuous speech recognition. *Neural Computing and Applications*, *28*(1), 1177–1196. https://doi.org/10.1007/s00521-016-2438-x

Badmos, O., Kopp, A., Bernthaler, T., & Schneider, G. (2020). Image-based defect detection in lithium-ion battery electrode using convolutional neural networks. *Journal of Intelligent Manufacturing*, *31*(4), 885–897. https://doi.org/10.1007/s10845-019-01484-x

Block, S. B., Silva, R. D. da, Dorini, L. B., & Minetto, R. (2021). Inspection of Imprint Defects in Stamped Metal Surfaces Using Deep Learning and Tracking. *IEEE Transactions on Industrial Electronics*, *68*(5), 4498–4507. https://doi.org/10.1109/TIE.2020.2984453

Casaño, C. D. L. C., Sánchez, M. C., Chavez, F. R., & Ramos, W. V. (2020). Defect Detection on Andean Potatoes using Deep Learning and Adaptive Learning. *2020 IEEE Engineering International Research Conference (EIRCON)*, 1–4. https://doi.org/10.1109/EIRCON51178.2020.9254023

Castillo-Bolado, D., Guerra-Artal, C., & Hernández-Tejera, M. (2021). Design and independent training of composable and reusable neural modules. *Neural Networks*, *139*, 294–304. https://doi.org/10.1016/j.neunet.2021.03.034

Chen, L. C., Pardeshi, M. S., Lo, W. T., Sheu, R. K., Pai, K. C., Chen, C.-Y., Tsai, P.-Y., & Tsai, Y.-T. (2022). Edge-glued wooden panel defect detection using deep learning. *Wood Science and Technology*. https://doi.org/10.1007/s00226-021-01316-3

Chordia, A., Sarah, S., Gourisaria, M. K., Agrawal, R., & Adhikary, P. (2021). *Surface Crack Detection Using Data Mining and Feature Engineering Techniques*. 2021 IEEE 4th International Conference on Computing, Power and Communication Technologies, GUCON 2021. Scopus. https://doi.org/10.1109/GUCON50781.2021.9574002

Chowdhury, I. M., Su, K., & Zhao, Q. (2021). MS-NET: modular selective network. *International Journal of Machine Learning and Cybernetics*, *12*(3), 763–781. https://doi.org/10.1007/s13042-020-01201-8

Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C. M., & Dario, P. (2020). Visual-Based Defect Detection and Classification Approaches for Industrial Applications—A SURVEY. *Sensors*, *20*(5). https://doi.org/10.3390/s20051459

Di, W., Bhardwaj, A., & Wei, J. (2018). *Deep Learning Essentials: Your Hands-on Guide to the Fundamentals of Deep Learning and Neural Network Modeling*. Packt Publishing.

European Committee for Standardization. (2015a). *Quality management systems – Fundamentals and vocabulary (ISO 9000:2015)*.

European Committee for Standardization. (2015b). *Quality management systems—Requirements (ISO 9001:2015)*.

Ferguson, M., Ak, R., Lee, Y.-T. T., & Law, K. H. (2018). Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning. *Smart and Sustainable Manufacturing Systems*, *2*(1), 137–164. https://doi.org/10.1520/SSMS20180033

Goel, A., Aghajanzadeh, S., Tung, C., Chen, S. H., Thiruvathukal, G. K., & Lu, Y. H. (2021). Modular Neural Networks for Low-Power Image Classification on Embedded Devices. *ACM Transactions on Design Automation of Electronic Systems*, *26*(1). https://doi.org/10.1145/3408062

Goetsch, D. L., & Davis, S. (2013). *Quality Management for Organizational Excellence: Introduction to Total Quality* (7th ed.). Pearson.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Harik, R., & Wuest, T. (2020). *Introduction to Advanced Manufacturing*. SAE International.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Hu, P. (2020). A classifier of matrix modular neural network to simplify complex classification tasks. *Neural Computing and Applications*, *32*(5), 1367–1377. https://doi.org/10.1007/s00521-018-3631-x

Huang, X., & Ren, J. (2020). Quality Control on Manufacturing Computer Keyboards Using Multilevel Deep Neural Networks. *2020 IEEE 6th International Conference on Control Science and Systems Engineering (ICCSSE)*, 184–188. https://doi.org/10.1109/ICCSSE50399.2020.9171988

Intisar, C. M., & Zhao, Q. (2019). A Selective Modular Neural Network Framework. *2019 IEEE 10th International Conference on Awareness Science and Technology (ICAST)*, 1–6. https://doi.org/10.1109/ICAwST.2019.8923334

Kamrani, A., & Salhieh, S. (2002). *Product Design for Modularity* (2nd ed.). Springer Science + Business Media. https://doi.org/10.1007/978-1-4757-3581-9

Kauer-Bonin, J., Yadav, S. K., Beckers, I., Gawlik, K., Motamedi, S., Zimmermann, H. G., Kadas, E. M., Haußer, F., Paul, F., & Brandt, A. U. (2022). Modular deep neural networks for automatic quality control of retinal optical coherence tomography scans. *Computers in Biology and Medicine*, *141*. https://doi.org/10.1016/j.compbiomed.2021.104822

Khishe, M., & Parvizi, Gh. R. (2020). Artificial Neural Networks, Concept, Application and Types. In D. Alexander (Ed.), *Neural Networks: History and Applications*. Nova.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, *25*. https://doi.org/10.1145/3065386

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Meng, X., Quan, L., & Qiao, J. (2020). A Self-Organizing Modular Neural Network for Nonlinear System Modeling. *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207263

Mureșan, H., & Oltean, M. (2018). Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, *10*, 26–42. https://doi.org/10.2478/ausi-2018-0002

Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C., & Valenzuela, A. (2020). A Review of Convolutional Neural Network Applied to Fruit Image Processing. *Applied Sciences*, *10*(10). https://doi.org/10.3390/app10103443

Nasiri, A., Omid, M., & Taheri-Garavand, A. (2020). An automatic sorting system for unwashed eggs using deep learning. *Journal of Food Engineering*, *283*. https://doi.org/10.1016/j.jfoodeng.2020.110036

Oakland, J. S. (2014). *Total quality management and operational excellence: Text with cases* (4th ed.). Routledge.

Poornima, M. C. (2017). *Total Quality Management*. Pearson India.

Pradana-López, S., Pérez-Calabuig, A. M., Cancilla, J. C., Lozano, M. Á., Rodrigo, C., Mena, M. L., & Torrecilla, J. S. (2021). Deep transfer learning to verify quality and safety of ground coffee. *Food Control*, *122*, 107801. https://doi.org/10.1016/j.foodcont.2020.107801

Qi, S., Yang, J., & Zhong, Z. (2020). A Review on Industrial Surface Defect Detection Based on Deep Learning Technology. *2020 The 3rd International Conference on Machine Learning and Machine Intelligence*, 24–30. https://doi.org/10.1145/3426826.3426832

Rathinavel, S., & Kannaianl, T. (2018). An Efficient Fabric Defect Prediction Based on Modular Neural Network Classifier with Alternative Hard C-Means Clustering. *International Journal of Engineering and Technology*, *7*, 277–284. https://doi.org/10.14419/ijet.v7i3.27.17892

Riedel, H., Mokdad, S., Schulz, I., Kocer, C., Rosendahl, P. L., Schneider, J., Kraus, M. A., & Drass, M. (2022). Automated quality control of vacuum insulated glazing by convolutional neural network image classification. *Automation in Construction*, *135*. https://doi.org/10.1016/j.autcon.2022.104144

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Runeson, P., & Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131. https://doi.org/10.1007/s10664-008-9102-8

Saxena, P., & Vijaivargia, A. (2020). Characteristics and Design Principles of Industry 4.0. In K. Jayakrishna, K. E. K. Vimal, S. Aravind, A. Kulatunga, M. T. H. Sultan, & J. Davim (Eds.), *Sustainable Manufacturing for Industry 4.0—An Augmented Approach*. CRC Press.

Shin, H.-C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., & Summers, R. M. (2016). Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, *35*(5), 1285–1298. https://doi.org/10.1109/TMI.2016.2528162

Shu, Y., Li, B., & Lin, H. (2021). Quality safety monitoring of LED chips using deep learning-based vision inspection methods. *Measurement*, *168*. https://doi.org/10.1016/j.measurement.2020.108123

Singh, S. A., & Desai, K. A. (2022). Automated surface defect detection framework using machine vision and convolutional neural networks. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-021-01878-w

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958.

Stephen, O., Maduh, U. J., & Sain, M. (2022). A machine learning method for detection of surface defects on ceramic tiles using convolutional neural networks. *Electronics*, *11*(1). https://doi.org/10.3390/electronics11010055

Tao, X., Zhang, D., Ma, W., Liu, X., & De Xu. (2018). Automatic metallic surface defect detection and recognition with convolutional neural networks. *Applied Sciences (Switzerland)*, *8*(9). https://doi.org/10.3390/app8091575

Tout, K., Meguenani, A., Urban, J.-P., & Cudel, C. (2021). Automated vision system for magnetic particle inspection of crankshafts using convolutional neural networks. *The International Journal of Advanced Manufacturing Technology*, *112*(11), 3307–3326. https://doi.org/10.1007/s00170-020-06467-4

Valdez, F., Melin, P., & Castillo, O. (2019). Optimization of Modular Neural Networks for Pattern Recognition with Parallel Genetic Algorithms. In L. Martínez-Villaseñor, I. Batyrshin, & A. Marín-Hernández (Eds.), *Advances in Soft Computing* (pp. 223–235). Springer International Publishing.

Wang, J., Ma, Y., Zhang, L., Gao, R. X., & Wu, D. (2018). Deep learning for smart manufacturing: Methods and applications. *Special Issue on Smart Manufacturing*, *48*, 144–156. https://doi.org/10.1016/j.jmsy.2018.01.003

Wang, T., Chen, Y., Qiao, M., & Snoussi, H. (2018). A fast and robust convolutional neural network-based defect detection model in product quality control. *International Journal of Advanced Manufacturing Technology*, *94*, 3465–3471. https://doi.org/10.1007/s00170-017-0882-0

Weimer, D., Scholz-Reiter, B., & Shpitalni, M. (2016). Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals - Manufacturing Technology*, *65*(1), 417–420. https://doi.org/10.1016/j.cirp.2016.04.072

Yang, J., Li, S., Wang, Z., Dong, H., Wang, J., & Tang, S. (2020). Using deep learning to detect defects in manufacturing: A comprehensive survey and current challenges. *Materials*, *13*(24), 1–23. https://doi.org/10.3390/ma13245755

Yang, Y., Pan, L., Ma, J., Yang, R., Zhu, Y., Yang, Y., & Zhang, L. (2020). A High-Performance Deep Learning Algorithm for the Automated Optical Inspection of Laser Welding. *Applied Sciences*, *10*(3). https://doi.org/10.3390/app10030933

Yang, Y., Yang, R., Pan, L., Ma, J., Zhu, Y., Diao, T., & Zhang, L. (2020). A lightweight deep learning algorithm for inspection of laser welding defects on safety vent of power battery. *Computers in Industry*, *123*. https://doi.org/10.1016/j.compind.2020.103306

Yun, J. P., Shin, W. C., Koo, G., Kim, M. S., Lee, C., & Lee, S. J. (2020). Automated defect inspection system for metal surfaces based on deep learning and data augmentation. *Journal of Manufacturing Systems*, *55*, 317–324. https://doi.org/10.1016/j.jmsy.2020.03.009

Zhang, B., Liu, S., & Shin, Y. C. (2019). In-Process monitoring of porosity during laser additive manufacturing process. *Additive Manufacturing*, *28*, 497–505. https://doi.org/10.1016/j.addma.2019.05.030

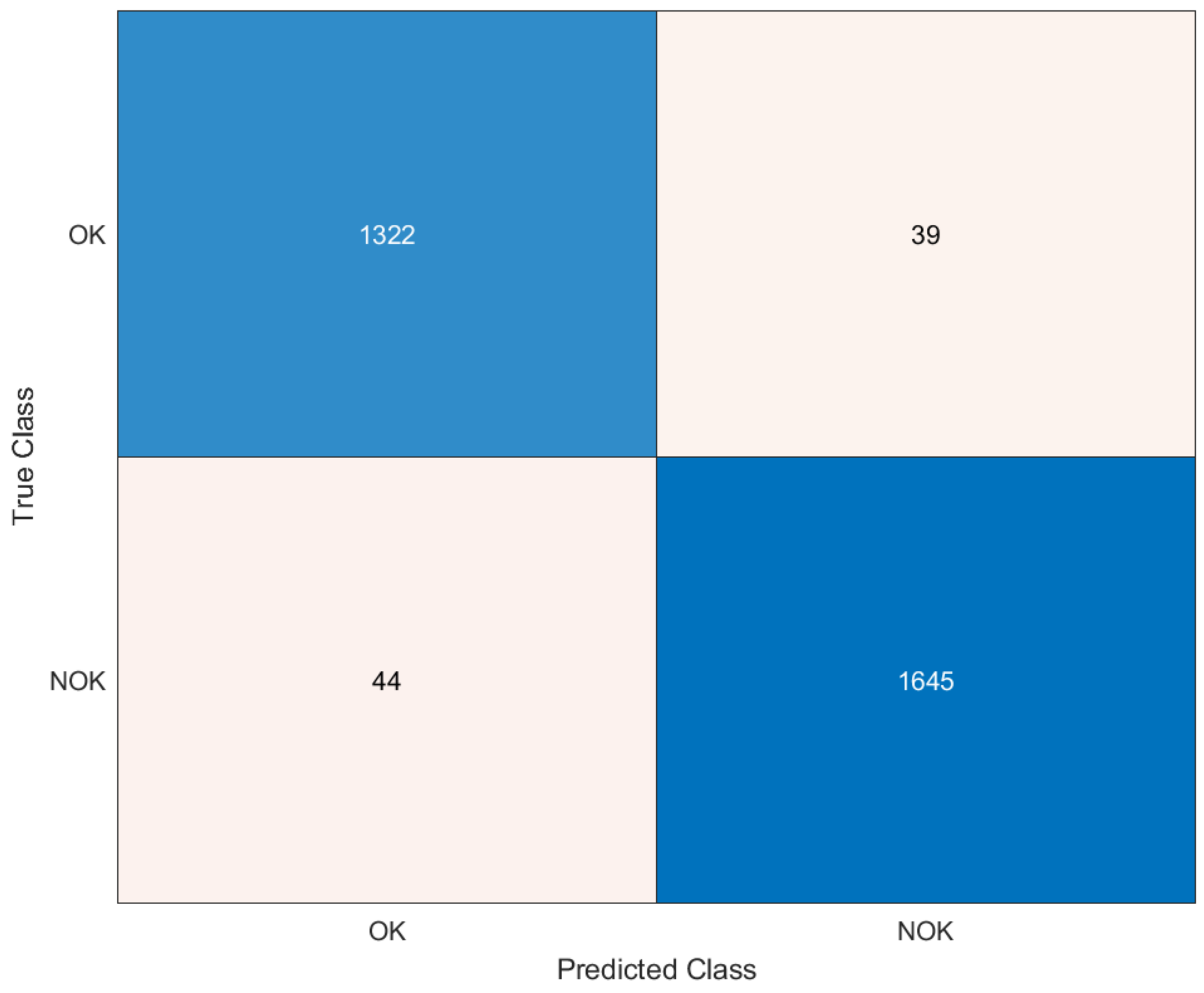
Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6848–6856. https://doi.org/10.1109/CVPR.2018.00716

Zhang, Z., Wen, G., & Chen, S. (2019). Weld image deep learning-based on-line defects detection using convolutional neural networks for Al alloy in robotic arc welding. *Journal of Manufacturing Processes*, *45*, 208–216. Scopus. https://doi.org/10.1016/j.jmapro.2019.06.023

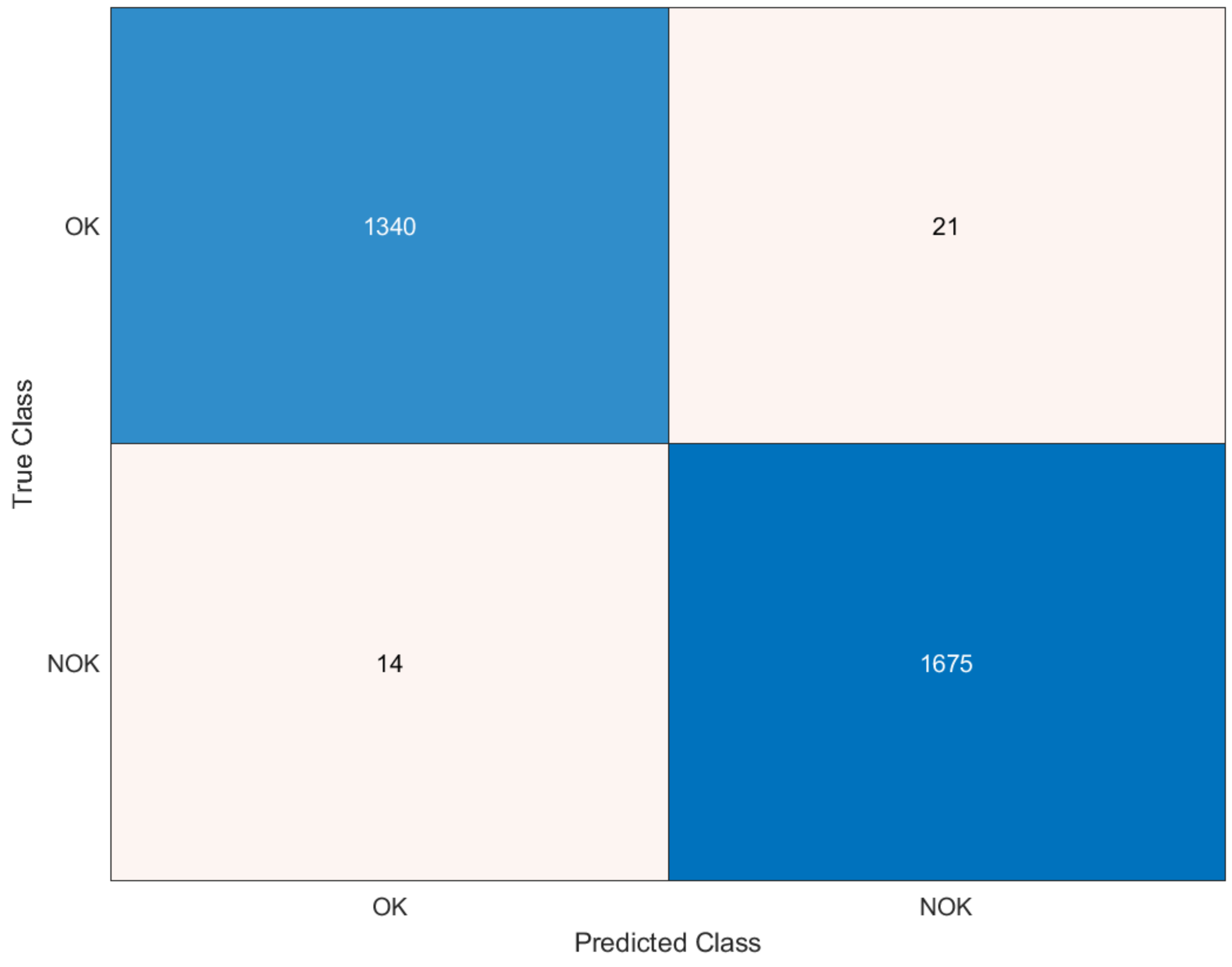
Zheng, X., Chen, J., Wang, H., Zheng, S., & Kong, Y. (2021). A deep learning-based approach for the automated surface inspection of copper clad laminate images. *Applied Intelligence*, *51*(3), 1262–1279. https://doi.org/10.1007/s10489-020-01877-z

Appendix A: Confusion Matrices

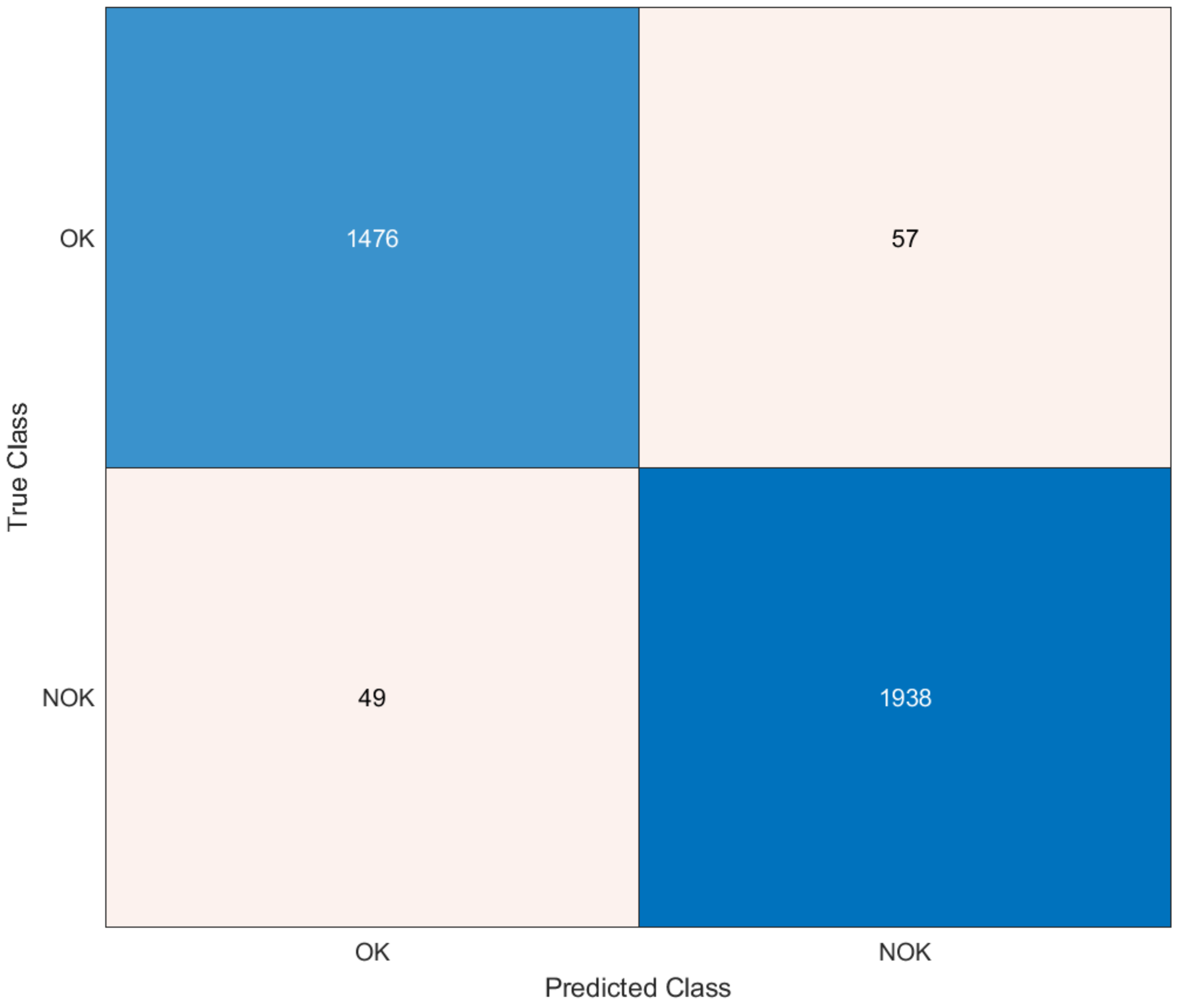
Confusion Matrix for initial modular network:



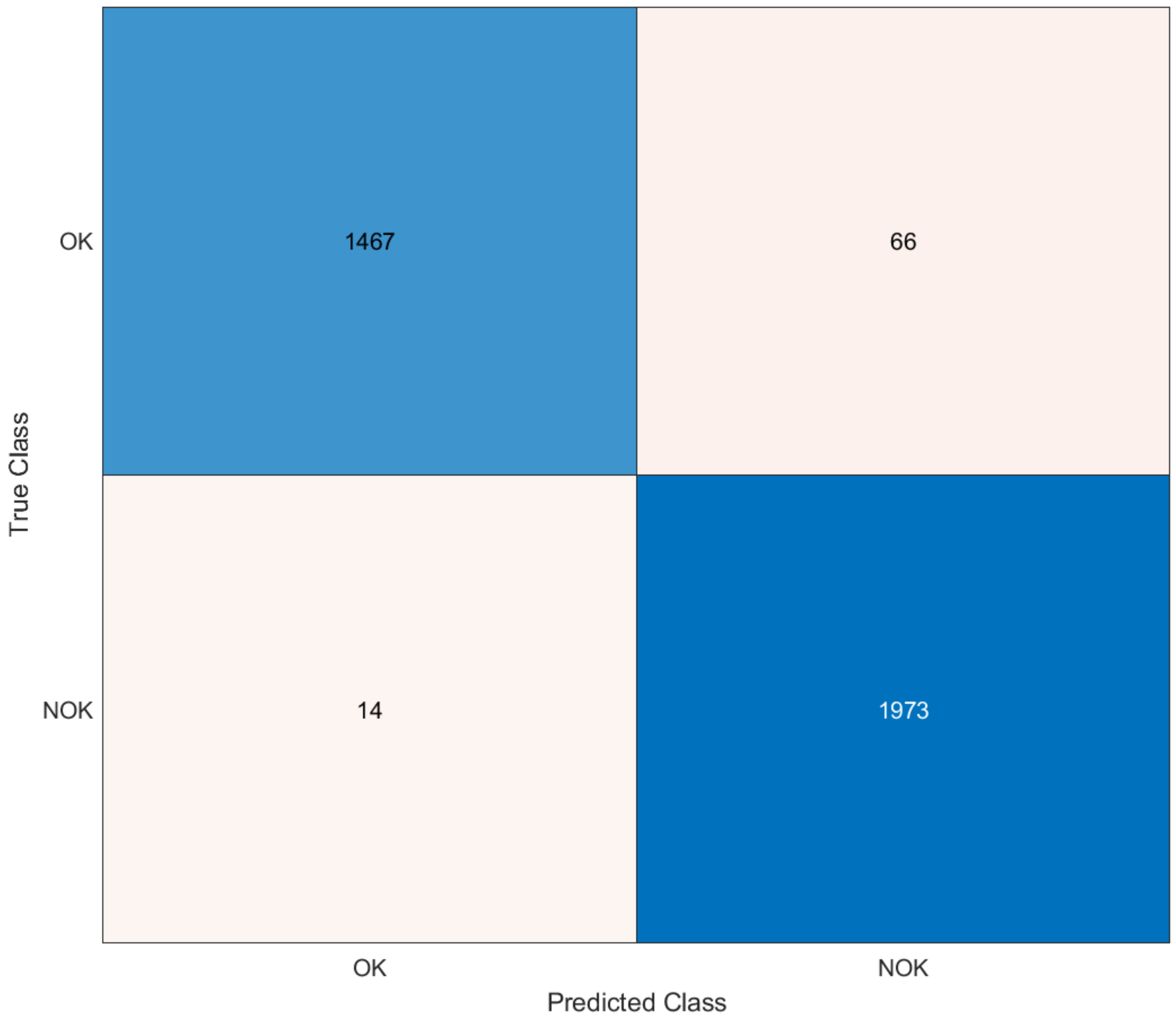
Confusion Matrix for initial monolithic network:



Confusion Matrix for retrained modular network:



Confusion Matrix for retrained monolithic network:


True Class
OK
NOK
OK
NOK
Predicted Class