university of groningen / faculty of mathematics and natural sciences / artificial intelligence

University of Groningen

The Netherlands

Master Thesis

Artificial Intelligence

On the application of deep learning methods in the real world: Image-classification of shipping container X-ray scans

Author: Jasper Krebbers (RUG, s2585529)

Internal Supervisor: Prof. Dr. Lambert R. B. Schomaker (RUG)

External Supervisor: Jetze T. Baumfalk (Douane)

Abstract

The port of Rotterdam is the largest port in Europe and millions of containers pass through annually. The herculean task of regulating, securing and inspecting this flow of goods falls on the shoulders of Dutch customs. Dutch customs is the among the biggest customs agencies of Europe. In recent decades they adapted to the increasing globalization by introducing and applying new techniques such as X-ray scanning, however due to manpower shortages only a fraction of shipping containers can be inspected. With Brexit around the corner and yearly increases in shipping volume Dutch customs will need to work both harder and smarter. In this thesis we focus on the application of deep learning algorithms in the customs domain using a real-world dataset of X-ray scanned shipping containers. We examine several aspects of the deep learning design process and evaluate which approaches can improve the reliability, extendability and explainability of deep learning systems. We created a dataset containing 14 common classes in order to compare several state-of-the-art pretrained deep learning neural networks. We found that all of them were able to achieve excellent performance with recall, precision and F1 scores. In order to verify the importance of pretraining we trained several networks with different types of initialization and sections frozen. Even though the X-ray domain looks very different from the ImageNet domain a network, which weights were completely frozen except for the final fully connected layer, was able to achieve impressive performance. The nomenclature that Dutch customs uses is 5 levels deep with a maximum of 100 classes per level, which results in 10 billion possible unique classes. We compared several types of output/label encoding in terms of size, performance and the possibility of extension using our created dataset of 14 classes. There appeared to be very little difference in performance when comparing one-hot, label and sparse-label encoding which allows for significant output size reduction when using the latter two options. Our final point of focus was the explainability aspect of classification algorithms by comparing several visualization methods such as Class Activation Maps and Pixel Voting. This was a qualitative experiment which yielded interesting insights, but yielded no clear superior approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In the recent decades international shipping has grown to surpass 10 billion tons in 2015 according to the United Nations Conference on Trade and Development (UNCTAD) [51]. Around 40% of this amount is so-called "dry cargo other than bulk", which is cargo mostly transported using container ships. It is up to Customs agencies to regulate, secure and inspect international trade. The port of Rotterdam for example services around 14.5 million of Twenty-Foot Equivalent Unit (TEU) shipping containers annually [29]. Slightly less than half is exported and the rest is imported, which means that around 4.5 million containers (7.5 TEU) at least would have to be inspected by the Douane (Dutch customs) annually. This amount increased by a quarter of a million containers year over year and it is expected that this trend will continue in the future.

Dutch customs needs to do its job reliably and efficiently, but ideally without much disruption to the flow of trade. Due to these requirements in addition to budgetary and logistical constrains it is not possible for them to inspect every container and ship. According to Dutch customs officials at the port of Rotterdam around 7.000 containers are opened up for inspection annually. This might seem relatively low, but it is important to consider that Dutch customs has only around 150-200 customs officers on site as of 2017 [8]. Only a small percentage of containers can be opened for inspection, therefore officers generally perform screening of arriving containers in addition to a random selection. The focus of this screening is to track containers with distinct manifests or from particular origins, among other factors, that are suspected to have higher chances of containing illicit goods. This approach, while somewhat effective, can be circumvented quite easily by shipping from a different country or falsifying the manifest. On top of that the approach is still limited by the amount of customs officers that can inspect those containers. If inspections could be performed faster it would allow customs officers to inspect more containers. This will increas the chances of finding illicit goods, such as weapons, drugs, people and valuable materials. International smuggling of these illicit goods is estimated to be valued

higher than $400 billion annually, thus the stakes are high [24].

One major innovation has been the application of X-ray technology for scanning containers. This allows customs officers to take a digital peek inside a container without having to open it, thus allowing officers to process more containers. X-rays (or Röntgen rays) are a type of electromagnetic radiation, which can pass through different materials at different rates [34]. Some materials such as metals and bone are more difficult to pass through for the radiation, these differences can be visualized in gray-scale. This visualization can give customs officers insights to the contents of a container. It requires extensive training to interpret the translucent and cluttered gray-scale images, this problem becomes even more challenging considering the enormous variety of objects that are transported via shipping container. Furthermore, online shopping has both fragmented and diversified the contents of shipping containers due to shipping collections of packages/orders from the same online store compared to bulk transport of a single item. Nonetheless, the technique has greatly increased the amount of containers that Dutch customs can inspect. Dutch customs scans up to 50.000 containers a year, which is a considerable number but still nowhere near the total amount that needs to be inspected.

It is apparent that the main bottleneck of Dutch customs container inspection operations is (analyst) manpower, while increasing globalization, recent talks about Brexit and emerging trade wars have presented Dutch customs with increasing responsibilities and new challenges that put extra strain on the organisation's capacities. Advances within artificial intelligence and more specifically Machine learning/Deep learning (ML/DL) could be a solution to Dutch customs manpower bottleneck, by using an automated or computer aided system based on ML methods for analysis of container X-ray scans. Previous research has shown that ML and more recently DL is capable of classification performance on images equal to or surpassing that of a human in some cases [12]. These methods could be used to automatically analyse containers that would otherwise not be scanned due to manpower shortages or as an additional tool for analysts. As such increasing both the speed of analysis and the organisation's total analysis capacity.

1.2 Machine and Deep Learning

While the convolutional neural network (CNN) was invented in the 80's its application for image analysis only became viable due to innovations like new regularization techniques, GPU acceleration, the backpropagation algorithm among many others [30, 14, 39]. The subsequent wins of image competitions by CNN based artificial neural networks (ANN), especially the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2012 [21], solidified them as state-of-the-art methods. A

great feature of CNN's is the ability to use previously learned information for other (similar) tasks, a process called transfer learning [4]. The lower layers of a CNN learn various simple shapes and features that are present in most train images regardless of the required task, making them similar to universal feature extractors [54]. Using the weights associated with these learned features, instead of randomly initializing the weights of a network, reduces the time required to converge and can save a significant amount of time. It is so effective that state-of-the-art networks such as ResNet, MobileNet, Inception etc. are often provided with the option to initialize them with weight obtained through training on the ImageNet dataset [15, 41]. Since this training is done primarily to learn low-level universal features up front instead of a main goal, like learning to classify objects, this process is referred to as "pretraining" [41]. While the performance of networks pretrained on ImageNet are generally acceptable on a wide variety of tasks, previous research found that transfer learning could harm performance if the domains are too different [37]. This leaves the question, how universal are these learned features and how well do they transfer to domains that do not consist of natural images such as ImageNet? This reduction in overall training time when using pretraining made it viable to create complex multi-purpose networks that could be extended. In order to increase performance researchers started to use an increasing amount of layers in ANN's, which resulted in the multi-layer networks being called deep neural networks (DNN) due to their increased complexity. This increased complexity often results in better performance, such as the 100+ layer network that won the 2015 ILSVRC [13, 41], but also made the model more difficult to understand. This has given rise to the call for explainable artificial intelligence (XAI), which aims to make AI more transparent [11]. Increasing transparency in AI and more specifically DNN's will not only help researchers better understand the behaviour of their models, but increase the validity and trust in the decisions that these models make. An important step required for many real-world applications where the stakes and consequences are high. In addition to explainability there are often scalability and flexibility requirements when applying deep learning algorithms in the real world. Will the approach still work if we have massive amounts of output classes or if we switch to a different type of classification? Using different encoding techniques can compress information in a smaller footprint, possibly reducing a models complexity while retaining performance and even allowing for extendability.

1.3 Research Questions

The focus of this thesis will be the application of deep learning methods on a real-world problem and possible practical problems or limitations that can be encountered. Dutch customs is the largest customs agency in the EU and is always at the forefront

of innovation. For this reason the University of Groningen (RUG) was given the opportunity to work together with Dutch customs on exploring the possible applications of ML/DL in the customs domain. Dutch customs provided a real-world dataset consisting of X-ray scans of trucks carrying shipping containers with corresponding shipping and administrative information, without localization data.

The aim of this research is to find out if it is possible to perform reliable classifications of goods with corresponding localization's or explanations using a real-world dataset. Among the other regions of interest were practical constraints, such as the benefits of dataset specific pretraining and the impact of output layer shape and size when working on large-scale data classification. The European customs classification system (nomenclature) is a method of classifying different types of goods and objects within the customs domain. It is a hierarchical system consisting of five levels with 100 possible options per level resulting in a total of 10 billion possible unique codes for an object depending on its functions, materials, construction and other attributes. This system is also used by Dutch customs and is an excellent testbed due to the many different possible classifications it facilitates [53]. Its hierarchical nature also means that there are several different encoding methods available. This thesis is based upon the following research question:

Which approaches improve the flexibility, reliability and explainability of deep learning systems while retaining performance when applied to real-world problems?

We have split this main question in four subquestions which we try to answer by using the real-world problem presented to us by Dutch customs as a reference. The subquestions are as follows:

1. Are there significant performance differences between different types of architecture and networks when using X-ray images? (Section 4.2)

2. Does pretraining provide any significant performance benefit? (Section 4.3)

3. What is the impact of different types of encoding and sizes of outputs on overall performance? (Section 4.4)

4. Is it possible to sufficiently explain the classification decisions of a system in a way that enhances an analyst's analysis? (Section 4.5)

1.4 Outline of the thesis

The next chapter of the thesis will focus on broad background information regarding the fundamentals of the used techniques. The third chapter is centered around the dataset, its origins and the steps taken for preprocessing and selection. The fourth chapter is concerned with the methods used for the experiments and the specifics such as the models, parameters, setups and techniques tested. There are four experiments which correspond to the four research questions. A standard format is used for describing each experiment, where first a small introduction to the experiment is presented. Followed by additional background information regarding the models, model-parameters and techniques. The format ends with a description of how the experiment is conducted, including any relevant parameters. The fifth chapter will report on the found results of the experiments laid out in chapter four. The thesis is concluded in chapter six where the results and the answers to the research questions are analysed, discussed and summarized.

Chapter 2
Background information

2.1 Classification

The goal of Dutch customs is to detect transported goods in containers using X-ray imagery. In order for a machine learning system to be useful it is required that the goods are classified and possibly also quantified. Classification is important to find what type of goods are transported and whether these are legal, illegal or if they require additional taxes. The latter is often dependent on the origin and the quantity or total worth of the specific good that is transported [1].

2.1.1 Image classification

Image classification is often done by searching for specific patterns or features within an image that are known to signify the presence of a specific class. Classification of images in essence is a specific form of automated pattern recognition which combines observed (sets of) patterns of features within images and their corresponding labels/classes. Pattern recognition is not only something people do (constantly) every day, but it is also widely used within computing.

2.1.2 Pattern matching

One of the basic forms of pattern recognition is pattern matching [50]. It recognizes patterns by comparing input data to a template, which is a pattern that consists of a combination of specific detectable features. In order to successfully use pattern matching it is required to know exactly which features and what specific template will match with the pattern you wish to detect. This takes a lot of experience and prior knowledge about the importance of specific features or patterns. While pattern matching can be very effective, it has some limitations. Important features and patterns used for recognition within a domain have to be created beforehand. However, in many cases not all these features and patterns are known beforehand. Therefore, it requires domain-experts to discover relevant features and patterns. This is often an expensive, labour-intensive and time-consuming process without a guarantee that

the complete domain is covered. Hand-crafted features are also quite rigid and therefore hard to adapt for other use-cases, which requires redoing a significant amount of the process for each domain. This does not lend itself well to pattern recognition in images, since patterns and features in images are often complex and have a lot of variation. This last argument is a significant factor in real-world image pattern recognition, where almost no image, pattern or feature is exactly the same.

2.1.3 Expert systems

Expert systems use logic-based inference engines and a knowledge base instead of handmade templates for recognizing patterns [18]. This was a step up from pattern matching, which was significantly more rigid and dependent on human experts. The knowledge base contained the knowledge and experience of domain-experts as highly structured rules from which the inference engine could extract new rules or make classifications. These systems could be made to work adequately, but their potential was still limited due heavy reliance on pre-existing human knowledge of a domain. That last point could be problematic, since it generally meant that it was not possible to develop an automated decision system such as an expert system without previous in depth human research, the latter often being expensive and time-consuming.

2.2 Machine Learning

The field of machine learning (ML) is focused on creating models based upon observed patterns and inference [3]. Algorithms used in ML try to create a mathematical model that learns to make predictions or decisions based on experience [20]. The ML approach tries to reduce the reliance on domain experts by learning instead of depending on pre-existing knowledge. Domain-experts have learned to detect important features and patterns through years of experience. ML tries to emulate this way of learning to detect relevant features and patterns by looking at large amounts of collected and packaged data in a dataset. The algorithm uses the dataset to learn which specific features and patterns in a data instance are important for a correct prediction [20]. Since the algorithm is software based, it is able to train itself to become an expert in this way much quicker than any human can. This approach has several advantages over the previously used expert systems. First off it does not require human defined rules, knowledge bases or solutions. Secondly, it is not dependent on features and patterns created by humans and can therefore also find patterns that domain-experts might have missed. In essence ML aims to accurately model data interactions by learning via a flexible trial and error approach instead of being fully hand-engineered by or dependent on domain-experts, thus trading in

some expert knowledge regarding features and patterns for learned experience. The data including its collection and subsequent preparation is however very important for ML algorithms and often still requires the skills of domain-experts in order to be useful.

The process of learning in ML is often separated in distinct stages, such as training, validation and testing in order to verify the performance [28]. During the training stage the algorithm learns to make correct predictions using some form of feedback. This is followed by the validation stage, where the performance of the system is evaluated without the algorithm receiving direct feedback. This evaluation can be done using labelled examples or metrics analysis and is used to monitor the progress and adjust parameters of an algorithm. These first two stages are often repeated multiple times until the performance plateaus or the algorithm is deemed to be finished. The final development stage is testing, where the ability of the algorithms to generalize is tested and its performance is evaluated [28]. This is similar to the validation stage, but it uses a set of previously unseen data instead. After development is done the next step will be putting the system in production, which often comes with a host of new challenges. Although we will touch on a few of these problems like system scalability and explainability, most fall outside the scope of this thesis. There are multiple ways to apply this way of learning to an ML algorithm, but a common method is supervised learning.

2.2.1 Supervised Learning

With supervised learning an algorithms is expected to learn patterns in the data that can correctly predict the presence of a specific class or feature [42, 20]. The algorithm is presented with a dataset that contains combinations of inputs and labels. The input contains all information while the label indicates the corresponding correct answer. Since the algorithm can only use information that is present in the dataset, it is important that all relevant information is included. The knowledge of domain-experts is therefore essential when trying to select relevant information for a dataset, especially for the labeling of data.

While all ML is based on the idea of learning through experience, the supervised approach focuses on a style similar to tutoring. The algorithm is provided with feedback on the accuracy of its prediction. During the training stage the algorithm will make a prediction based on the input combined with its previous knowledge. ML algorithms are generally randomly initialized which makes their first predictions no better than guessing. After the algorithm gives its prediction it is provided with the actual answer often referred to as the label. The algorithm can then evaluate its own answer with the actual answer by calculating the differences between them. If

the difference was small then its prediction was not far off and it should only slightly tweak its prediction. If on the other hand the difference was large it should reconsider its reasoning. The difference between a predicted and an actual answer is generally referred to as the loss, and the function used for this calculation is referred to as a loss function, which we will explain further in Section 2.3.3. The algorithm will try to find patterns or features in the input that can be used as an indicator for specific labels. It learns by cross referencing the presence of features or patterns with labels in large amounts of data and by looking at which features or patterns either reduce or increase the loss for a specific label [42].

During the validation stage the performance is generally evaluated using metrics such as the ratio of correct predictions or the average difference between the predictions and the labels (average loss). There are also other metrics that can be used as an indicator of performance, many of which focus on specific aspects of a models performance. In general the mentioned ratios give an indication of the operational accuracy of the model and whether any (hyper)parameters should be adjusted in order to improve learning performance. The set of data used for validation can be held out during the dataset selection procedure or it can be a subset split off from the original training data just before training.

The testing stage generally only differs from the validation stage in that it is only performed once when the training is finished and that the dataset used is always a holdout set that does not contain any overlap with the training- or validation set.

2.3 Artificial Neural Networks

There are many different types of models and architectures used in ML, but we chose to focus on the ANN. This family of models is loosely based on mathematical abstractions of biological neurons and is used in many state-of-the-art models [40]. It consists of many different implementations which each have their own uses. Some are focused on modelling biological neurons or networks, which allows researchers to perform digital versions of experiments. However, these are often limiting when used for machine learning or computational purposes. Most ANN's used for machine learning use the perceptron, because it is a simpler model without any timing or activation leak constraints present in (modelled) biological neurons.

2.3.1 Perceptron

The perceptron is the basic form of an artificial neural network and was designed by Rosenblatt in 1958, it was based on the original McCulloch-Pitts (MCP) model of the neuron [35, 25] which can be seen in figure 2.1. It is the simplified version of a

single biological neuron consisting of a node, a bias and weights. Biological sensory neurons register inputs and pass on electric potentials, the frequency and size of these potentials determine if the signal is passed on in the network. In the perceptron these potentials are called activation and they are only passed on once per set of inputs. Both the inputs and activation are represented as digital values.

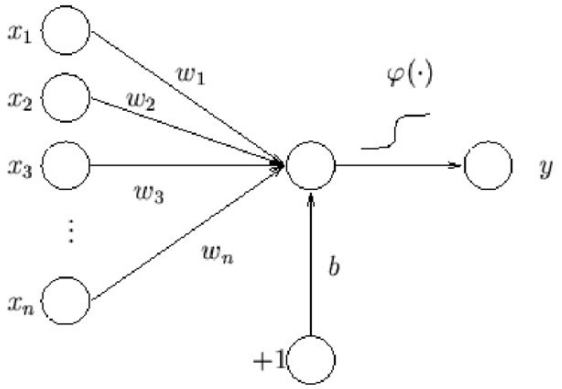

Figure 2.1: A basic schematic of the perceptron architecture [55].

The input activation is modified by the inhibitory or excitatory connections between the input and the perceptron's node, which are represented as the weights in the perceptron model [40]. The weighted activation is gathered in the perceptron's node and is calculated by taking the sum of all inputs each multiplied by the weight of its connection to the node as seen in Formula 2.1. Where a is the weighted sum of all activations, x_k is the input activation for an input k connected with the perceptron's node and w_k is the weight value of the connection between the input k and the perceptron's node. Finally, a trainable bias weight parameter b is added to the function as well.

$$a = (\sum_{k=1} x_k \, w_k) + b \tag{2.1}$$

A so-called activation function ($\varphi(a)$ in Figure 2.1) is applied to the weighted activation (a) and it determines whether the perceptron has enough activation to "fire" and with what intensity (this is explained further in section 2.3.2). Since a perceptron either does or does not fire for a specific input, it can be used as a binary classifier where a specific hyperplane in the input space is the decision boundary. The bias parameter can be used to offset this decision boundary from the origin, which can increase the activation required for firing and therefore introduces a threshold

option. Weights are updated during training using the perceptron training rule [35]. If there exists a possible hyperplane that separates the positive and negative samples of the data, the perceptron will converge on it by iteratively updating its weights. When there exists no such possible separating hyperplane, the perceptron will not converge and the training will fail.

2.3.2 Activation functions

The basic perceptron often uses the Heaviside step function which converts every positive number to 1 and every negative number to 0, similar to the all-or-nothing response of the synaptic cleft. Because the activation function outputs either 0 or 1 it makes the perceptron overall a linear (binary) classifier, which is able to divide the input-space so that (most) positive inputs are on one side and (most) negative inputs on the other, when the weights are set correctly and there is a pattern present in the data.

However, the value of the weighted activation that is calculated in a node is not limited to any range however. This can have significant impact on the performance of a neural network. To combat this an activation function is applied to the node's activation which generally restricts the output activation of the node to a specific range, often {0,1} or {-1,1}. There are other activation functions beside the Heaviside step function mentioned before, such as sigmoidal or radial basis functions. These are non-linear, an attribute that is needed to create non-linear models.

The rectified linear unit (ReLU) is an activation function that is often used in convolutional layers. Its upper range is unlimited while its lower range is restricted to 0, which means that only nodes with positive weighted activation pass their activation on to the next layer. It removes any negative activation from the networks activation maps, therefore only the presence and not the absence of features is taken into account for classification.

2.3.3 Loss functions

Loss functions are applied in many fields within AI and serve to guide learning algorithms in the right direction. They do this by giving algorithms a performance indication. Whether the algorithm needs to learn actions or predictions there is generally an optimal goal or correct prediction. The difference between the optimal goal or prediction and the action or prediction of the algorithm is referred to as the error. The loss function provides an indication of the algorithms performance by making a calculation based on this error.

A prediction algorithm will aim for a probability of 1 (100%) for the correct prediction since that is the optimal prediction. However, it might never converge

on this optimal prediction probability. The "loss" in loss function refers to the loss of efficiency of the picked action or prediction probability compared to the optimal action or prediction probability. Optimization problems always aim for minimizing the loss and therefore the error, therefore maximizing or optimizing performance.

A loss function often used for regression problems is the mean squared error (MSE) loss. Here the loss is calculated with function 2.2, where n is the sample amount and $(Y_i - \hat{Y}_i)$ is the error of a given sample i. The function makes the loss independent of the direction of the error by squaring the error, this makes it impossible for positive and negative errors to cancel each other out when summing them. The second reason that the error is squared is to penalize predictions that are far off the desired outcome more compared to predictions that are relatively close.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.2}$$

There are different types of classification problems that focus on different aspects of classification besides a simple binary classifier. For more complex problems such as multi-label and multi-class classification, different loss functions are often needed. Generally the cross-entropy loss function is used for a multi-class problem and the binary cross-entropy loss function combined with a logits function for a multi-label problem.

Cross-entropy originates from information theory and is used to calculate the difference between probability distributions. The output vector of a model or algorithm can be interpreted as a probability distribution over the different predictions. The encoded label has the same form and can be seen as a probability distribution with the probability of the label being equal to 1. Cross-entropy uses equation 2.3 to calculate the difference between these two distributions for some (random) set of inputs. Where H is the cross-entropy, p and q are the two probability distributions with i being the input. This can be re-written where y is the probability of the output for distribution p and $\hat{y}$ is the probability of the output for distribution q. Like the loss mentioned before, the goal is to minimize the dissimilarity between the prediction (probability distribution) and the label (probability distribution) for a given input.

$$H(p,q) = -\sum_{i} p_i \, log(q_i) = -y \, log(\hat{y}) - (1-y) \, log(1-\hat{y})2 \tag{2.3}$$

Using cross-entropy for multi-class classification is straightforward, since the (one-hot encoded) output vector can just be interpreted as a probability distribution. This is not the case for label-encoding, since more than one label can be present. Therefore, cross-entropy for label encoding is performed as binary cross-entropy for

every label separately. Here the distribution is not over the labels but rather a binary cross-entropy calculation regarding the probability of each label individually.

2.3.4 Optimizers

Optimization algorithms or optimizers are used to update the model. Their goal is to aid the model in minimizing the loss function, by using the feedback that the loss function provides, to determine which model parameters it should update and how it should update them. Generally it does so by making small alterations to the model weights, which results in different losses for each alteration. The backprop algorithm can compute these losses and the gradients of these losses efficiently using the chain-rule [38, 36]. In turn optimizers can use these first-order derivatives to find out where the gradient is the steepest. In theory the alteration which results in the steepest gradient should result in the highest loss decrease and the most performance gain. By only using these optimal weight updates the optimizer tries to reduce training time by increasing convergence speed.

A state-of-the-art optimizer that is very popular is Adam, which is an optimization algorithm with an adaptive learning rate that is very effective in combination with neural networks. When the gradients are computed using backprop Adam does two things, first is squares the gradients and uses that to scale the learning rate for each weight. Then it uses the moving average of each gradient instead of only the newly computed gradient. Adam also uses a global learning rate parameter for its optimization. In order to tune this parameter a learning rate finder was applied that used a cyclical learning rate approach [46]. This changes the learning rate used by the network for each mini batch in a cyclical pattern in order to estimate the corresponding losses and plots the results in a graph. The optimal learning rate will be at the point where the steepest declining slope is.

2.3.5 Multilayer Perceptrons

While a single perceptron is a fully functioning classifier, its limited linear binary nature means that it is not used as a complete network often in ML. The multilayer perceptron (MLP) is generally used instead, which is a class of neural networks that uses multiple perceptrons (often called nodes in this context) in layers [33]. An example of its basic architecture can be seen in figure 2.2. The MLP consists of three types of perceptron layers that are stacked. The first layer is referred to as the input-layer of the network, here every input is fed into the network. The final layer is referred to as the output-layer, it contains all output nodes of the network. All other layers in between the input- and output-layer are referred to as hidden layers. These layers allow the network to construct its own arbitrary features based on the input

and report on their activation. Hidden refers to the unknown and hard to interpret meaning of these features by humans and the fact that its values are never observed which is in contrast to the input and output values.

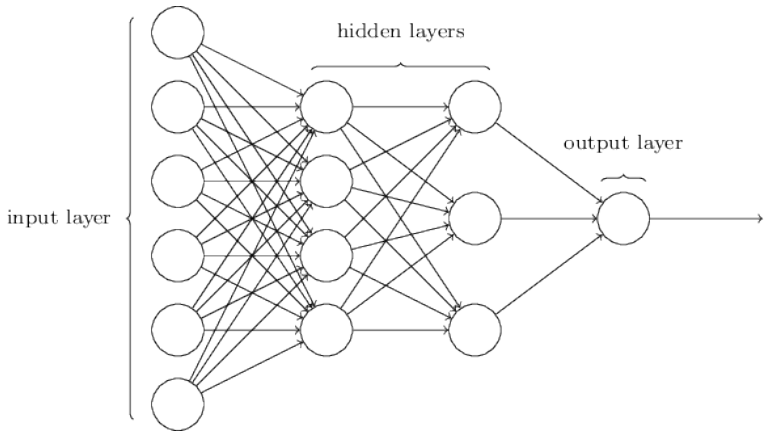

Figure 2.2: A schematic of an MLP architecture with two hidden layers and a single output node [23].

An MLP is a type neural network that is designed to be feed-forward. This means that output of every node is aligned the same way and is only connected to inputs of nodes in the next layer. There are no loops or directed cycles within the network and as such the activation is only allowed the flow from the input layer to the output layer. It is a standard format for MLP's that each node of a layer is connected to every node in the next layer, this connection is often called linear, dense or fully connected. Additionally, an MLP often uses the backpropagation algorithm to update its weights rather than the perceptron training rule.

The main reason that an MLP is used instead of the single perceptron is due to its ability to model a non-linear decision boundaries. Many problems that ML tries to solve are non-linear in nature and therefore exceed the modelling ability of the single perceptron. The MLP introduces non-linearity in the network by applying non-linear activation functions to the activations of the nodes in the hidden layers. Therefore the hidden layers map the linear input space to a non-linear space in the hidden layers. This introduces the possibility that the inputs can be linearly separated after their transformation to this non-linear space [5]. The MLP tries to learn a mapping in the hidden layers from linear to non-linear space that allows for the best linear separation in the final layer.

2.3.6 Convolutional neural network

One type of neural network that is frequently used when images are concerned is the convolutional neural network (CNN). Just like MLP's it is a feed-forward multi-layer neural network consisting of input, output and hidden layers. This class of deep neural network uses at least one convolutional layer in contrast to the fully connected layers of a MLP. A convolutional layer is different from a fully connected layer in several ways. A major difference is that the convolutional layer has three dimensions being width, height and depth instead of being linear like the MLP's fully connected layers [9]. Furthermore, the nodes in a convolutional layer are only connected to nodes in a preceding layer that fall within their receptive field. This receptive field is many times smaller than the size of the input layer of the CNN and uses convolutions to convolve its input with a set of filters or kernels [9]. Computer vision has traditionally used filters (or kernels) that were engineered by experts, but the filters used by a CNN are trainable and can learn relevant features and patterns. Each filter is an optimized detector for a specific feature that is learned by the system over time. CNNs use a hierarchical approach where each layer has its own set of (learned) features, layers at the start of a the CNN often have simple (geometric) features such as lines and curves. Layers higher in the network combine these simple features into more complex and specific features and patterns. The receptive field slides over the original input and maps the result of the filter convolutions to nodes in the next layer, which in essence becomes a mapping of features that are present at a specific location in the preceding layer. An example of the convolution operation can be seen in Figure 2.3.

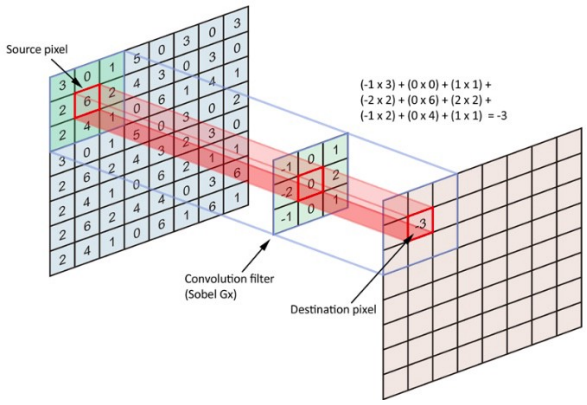

Figure 2.3: The convolution operation performed by a filter on the input [27].

This approach allows a CNN to retain information regarding to the location of

features in the input, which can be important context for classification or object detection. Another benefit is that the trainable parameters are located in the receptive field section of each layer which is relatively small, thus decreasing the amount of connections between layers compared to similar sized MLP-like networks. When used for classification, the prefinal layer of a CNN is a list of features. This list is used by the final fully connected layer for classification. In essence convolutions are a form of dimensionality reduction.

Pooling

Another type of layer that is often used in CNN's is the pooling layer. It is used to down sample a feature activation map using a function [9]. There are several different pooling operations such as average or L2 pooling, but mostly max pooling is used as it has been found to work best in practice. Pooling layers use non-overlapping patches of a feature activation map and for each patch it uses a pooling function to pass on a single value. For images usually a 2x2 patch is used, which results in a four times reduction of parameters in the next layer. While pooling reduces the total trainable parameters it also introduces some information loss regarding the exact location of feature activation.

Batch normalization

As the name implies a batch normalization layer applies a normalization operation per batch. Normalization is done by subtracting the batch mean from the output of the previous layer and dividing that by the batch standard deviation [16]. The batch mean and batch standard deviation are also added to the network as trainable parameters, this has a regularization effect by adding some noise to hidden layer outputs. While ideally normalization would use the complete dataset, this is often impractical and the reason that normalization is applied per batch. The goal of the batch normalization is to improve performance, stability and the speed of ANN's. Since normalization allows for higher learning rates which speeds up the convergence of an ANN.

2.3.7 Encoding

For the sake of uniformity Dutch customs encodes transported goods using the EU customs nomenclature. This nomenclature has five levels with a maximum of 100 options per level resulting into 10 billion possible unique combinations. The final code is a composition of a five numbers between 1 and 99 for a total length of 10 digits.

The digital implementations of ANN's using packages such as Tensorflow and PyTorch also make use of encoding, but instead rely heavily on a basic building block called the tensor. While the term originates from mathematics, the tensor can be seen as a multidimensional array which can be used by GPU's to perform large-scale matrix multiplication operations which speeds up training. In order to be used for training a network any inputs will have to be encoded as a tensor. Sometimes a simple encoding is sufficient, RGB images are often encoded as a 3D-array and can be easily converted to a 3D tensor by simply casting it. The output of a network is also encoded as a tensor and can defined by the user, although in general the shape is determined by the learning method and the format of the label or desired output. For supervised learning the output of the input-output pair are encoded differently.

One-hot encoding

One-hot encoding is the most used format for multi-class classification. The format is rather simple, but still very effective. The encoding consists of a vector with a length equal to the amount of classes and each class gets assigned an index within that vector. The vector is completely filled with zeroes except for the index of the class you wish to encode, which is filled with a one. An example can be seen in figure 2.1.

Class	**Index**	**Encoding**
DOG	0	0
CAT	1	1
BIRD	2	0
SNAKE	3	0

Table 2.1: One-hot encoding example of CAT .

One-hot refers to the restriction that only one index of the vector can be set to one. While this approach is proven to work well with a limited amount of classes, it has become clear that it does not scale well. In order to add a new animal class to our example we will have to extend our encoding with one extra digit for every class we want to add. When applied to ANN's with larger amounts of classes it starts to suffer from the curse of dimensionality, since every new class represents an extra node in the final layer. This layer is fully connected and will result an increase of connections equal to the amount of nodes present in the pre-final layer, which can be substantial in state-of-the-art deep learning networks.

Label encoding

There are different ways to implement label encoding depending on the application. The most basic form is similar to one-hot encoding but with a vector equal to the length of the amount of unique labels present in the dataset. While this vector is also filled with zeroes for not present labels, it is different from one-hot encoding by allowing more than one index (label) to be set to active. This is required for multi-label classification since instances can have multiple different labels. Because the label encoder does not have the restriction of only a single active index in the vector, it is better scalable. In table 2.2 an example of label encoding is shown, all the labels that are present in the table are also present in the dataset. There are also other types of encoding, some of which we will explain further in section 4.4.

Label	**Index**	**Encoding**
ANIMAL	0	1
MAMMAL	1	1
REPTILE	2	0
PET	3	1

Table 2.2: Label encoding example of CAT which is an ANIMAL, MAMMAL and a PET, but not a REPTILE .

2.3.8 Deep Learning

Deep Learning (DL) is a subset of machine learning, the exact boundary between the two has some ambiguity but is often referred to as using an ANN with more than one hidden layer. When using hidden layers an ANN creates its own features which is at the core of deep learning [22]. Most state-of-the-art DL systems have a significant amount of hidden layers and different types of connections between them compared to models that are generally seen as just ML. The additional layers and connections allow DL systems to extract more complex features. In the beginning these features are more geometric in nature such as corners or edges, but through the layers these get combined into more complex features. For example, for face recognition these complex features might include things like eyes, mouths, ears and noses for example.

This ability of combining and recognizing complex patterns results in improved performance by DL systems on difficult and complex tasks. The downside of DL compared to ML is that these layers contain (significantly) more nodes and connections which in turn require (significant) extra computation. Another downside is that it becomes hard to transfer any pre-existing expert knowledge to such a system since it

would need to be encoded in the weights. While (higher-order) hand-crafted features are out of the question both by DL design philosophy and the difficult encoding, there is another option for using pre-existing knowledge.

2.3.9 Transfer Learning and Pretraining

The process of transfer learning is focused on using stored knowledge learned from a previous problem to solve another similar problem [49]. The idea is that the learned information regarding the recognition of patterns within data can be used to solve multiple (different) problems. When using image data to classify different types of vehicles the system learns relevant features of these vehicles in order to make a distinction between them. These features and patterns are stored in the weights prior to the output layer. The output layer then uses these features to classify an instance, thus determining the use of the network. However, the learned features and patterns can be useful for other tasks as well, such as object detection or a different classification task regarding specific brands of certain vehicles. The features needed for these other tasks are not likely to be (very) different from the features that are already known to the system, thus reusing them will allow for learning these new tasks faster. This is especially the case concerning images, since objects in images at a base-level consist of simple geometric shapes. These simple shapes are often learned at the beginning of the network. Therefore, even if the task is significantly different from classifying vehicles using the learned features might still provide a significant boost in initial performance.

Transfer learning is often applied when prototyping, reducing the time needed to find out which approach works best for a specific problem or task. This practice is very popular and has given rise to the phenomenon of pretraining. The difference between the two is not strict but in general transfer learning is focused on transferring knowledge learned from a specific task to another similar task, while pretraining is focused on trying to fill a network with useful knowledge for many tasks [7]. In essence the intent of transfer learning is to perform well on one task and find out if the knowledge is useful for any other similar tasks, while the intent of pretraining is to learn many general features (using a general task) that are known to be useful for other tasks. Many current state-of-the-art model implementations come with optional weights pretrained on the ImageNet dataset [31, 7].

2.4 Explainable AI

The term explainable artificial intelligence (XAI) was born due to the lack of transparency present in many AI systems [11]. The good old fashioned artificial intelligence

(GOFAI) from the 80's and before was often crafted by experts and based on intelligent systems and logical based reasoning [17]. These systems and the decisions they made were explainable by their creators and the experts. The recent advances in AI with machine- and more specifically deep-learning are by contrast more like "black boxes", where even their own creators have difficulty to explain the exact reasoning behind a systems decisions [43].

The knowledge and reasoning behind the decisions of an ANN or similar types of learning algorithms are stored in the weights, which values are quite abstract to us since they are just floats that were reached after training convergence. The performance of ANN based machine- and deep-learning systems has been excellent in a wide variety of fields according to the specified metrics such as loss, accuracy and precision. This excellent performance is however not a guarantee that the system is performing as expected, but merely that it is able to solve the problem given to it.

A significant number of studies have experienced that the developed machine learning systems learned to distinguish classes not based on distinguishing features, but due to specific markers such as the presence of snow for classifying wolves or a watermark only present horse images of the dataset [43]. While the performance of these systems was great according to the metrics, they failed spectacularly in their mission. This not only undermines the trust in such systems, but without the reasoning behind a systems decision these decisions also become less valuable. Both factors harm the adoption of this promising technology in real-world applications, such as the healthcare system and risk assessment for law enforcement. Especially the latter has had trouble with automated decision systems that were supposed be fair to all. Instead these systems learned the underlying biases in the training data, which resulted in higher predictions for recidivism for specific groups [10, 24]. These decisions have a severe impact on people's lives which is hard to justify or even contest without access to the reasoning or evidence on which it is based.

The XAI movement strives to increase transparency and explainablity of AI systems not only for the reasons stated above, but also the benefits that it brings to the development of AI systems. When researchers understand a system it becomes easier to assess the systems actual performance, which allows for easier correction of unwanted behaviour and bugfixing.

Chapter 3

Data

3.1 Provenance

Dutch customs supplied an original dataset of 48.825 shipping container scans made in 2014, 24.672 of which were side-view scans and 24.153 of which were top-view scans. An example of a side-view scan can be seen in Fig 3.1. Such an image generally had a resolution of approximately 5.000 by 1.500 pixels. In addition to the scans Dutch customs supplied corresponding information regarding identifiers and contents of the containers in a csv datafile. There was no data provided regarding the objects locations within the containers. A selection of these images were processed using multiple steps to obtain the final dataset.

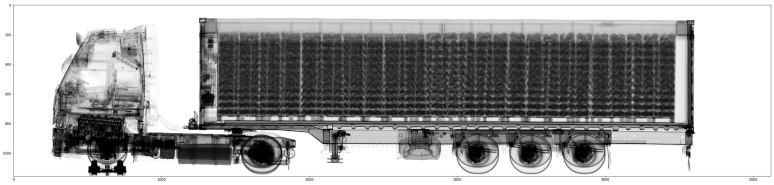

Figure 3.1: A side view scan of a truck transporting a container from the dataset supplied by Dutch customs.

3.2 Preprocessing

All the X-ray scans used were originally encoded in grayscale. These scans were transformed to 3-Dimensional RGB images through duplication and subsequently stacking, so that every dimension of the RGB format was the same. This was necessary to make use of the (pretrained) architectures which expect a 3x224x224 dimensional image/tensor as input.

3.2.1 Container extraction

The label provided for each image only takes the content of the container into account and disregards the rest of the original image. A dataset consisting of the location of the top-left and bottom-right corners of over 800 containers was created manually. A ResNet34 classifier was then trained on this dataset to predict the locations of a bounding box around the containers. This reduced the total image size significantly to resolutions in the ranges of 3.000 by 800 pixels. The classifier was able to extract most of the containers correctly. In order to automate the process several constraints and checks about minimum width and height added to the process, which could recognize any unusual or blatantly incorrect bounding boxes. The final performance of the classifier and the additional constraints was more than satisfactory for our use-case. The labelling and extraction of containers was considerably easier with the side-view images, thus the classifier was only trained with and can only extract bounding boxes for containers from side-view images. The extracted container from the example of Figure 3.1 can be seen in Figure 3.2.

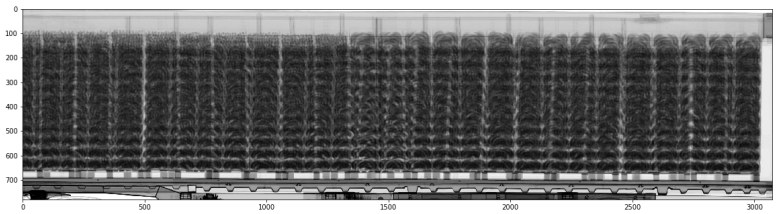

Figure 3.2: The container extracted by the ResNet34 classifier from the image in Figure 3.1.

3.2.2 Data selection

Exploratory analysis of the datafile gave a rough overview about the frequency of shipped goods within the dataset. The decision was made to only select containers that contained a single type of good in order to streamline the process. Since this research is aimed at supporting a pilot of automated classification within Dutch customs, a (semi-) broad selection of goods was agreed upon. A selection of 14 classes was made. An overview of the classes and their composite nomenclature codes of the first three levels can be found in Table 3.1. The first three levels of the nomenclature are the most relevant and distinguishing categories for classification, while the last two levels are often used for a distinction between fresh and frozen. Therefore, we decided to only use the first three levels.

There were several requirements used in making the selection. The first requirement was that the container needed to contain goods that only differed slightly in shape, form and size. This requirement was met by selecting 6 different types of fruits. The second requirement was that there needed to be classes that differed greatly in shape, material and size. Because a substantial amount of containers pass through empty and some containers can be partially filled, it was practical to include an EMPTY class. Additionally an OTHER class was included to allow the network to classify any unknown objects.

Class	**level 1**	**level 2**	**level 3**	**Nomenclature code**
AVOCADOS	08	04	40	080440
BANANAS	08	03	10	080310
CAR	87	03	00	870300
EMPTY	00	00	00	000000
GASGENERATOR	85	02	00	850200
GRAPES	08	06	10	080610
MANGOES	08	04	50	080450
OTHER	99	00	00	990000
PHOTOVOLTAIC	85	41	00	854100
PINEAPPLES	08	04	30	080430
SHOES	64	00	00	640000
TIRES	40	11	00	401100
WATERMELONS	08	07	11	080711
WINE	22	04	00	220400

Table 3.1: The selection of different classes with the corresponding categories at each of the three levels in the nomenclature and the resulting composite codes.

3.2.3 Extraction

Since the resolution can be quite important for locating or classifying certain types of goods, it was decided that resizing the container to fit the input of state-of-the-art models was not an option. The decision was made to use a sliding window to extract patches of 224x224 pixels from the containers, because most containers were filled completely and all containers needed to be completely scanned regardless. The patches overlapped 50% in both the horizontal and vertical directions, Figure 3.3 shows how and where these patches were extracted from complete images and Figure 3.4 shows an individual patch. A selection of these patches was manually verified

and divided into two categories, being either "empty" or "filled" with something. This selection was used to train a ResNet18 classifier to try and reduce the amount of manual verification needed. Performance was not satisfactory and extending the training selection was deemed to be more work than manually verifying the actual dataset at that point.

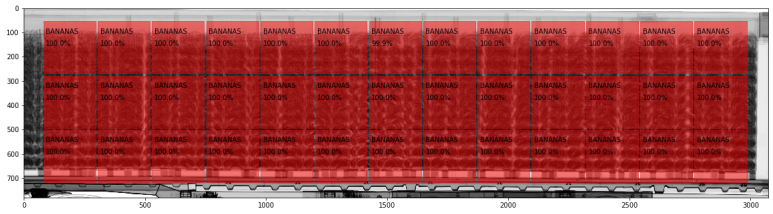

Figure 3.3: An example of 224x224 patches that were extracted from a single image, for illustration purposes these patches are non-overlapping and provided with classifications.

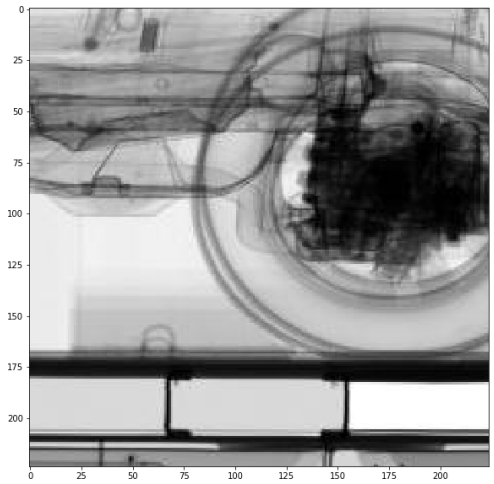

Figure 3.4: An example of a single 224x224 patch extracted from a complete image which was used for either training or testing the algorithm.

3.2.4 Dataset extension

An initial dataset with the 14 classes of Table 3.1 was selected using keyword matching on the supplied contents of the containers. This selection was manually verified

and subsequently patches were extracted for training an initial ResNet18 classifier. The patches were stratified and split between a train- and validation set, with the trained classifier obtaining satisfactory performance with accuracy $> 90\%$ on validation set. This initial classifier was then used on the complete dataset, to find more occurrences of 12 classes (EMPTY and OTHER were excluded). Due to the large variation of goods present in the original dataset the classifier suffered from a significant amount of false positives. All newly found instances, which numbered around 5.200, were manually verified using the datafile and sometimes additional verification of the images was performed. After verification some 700 new instances of our 12 classes remained. Although this seems a small percentage it increased the size of our initial dataset by 2.5 times. The main reason is that many of the found instances belonged to underrepresented classes, which had proven to be problematic for stratification of the dataset.

3.3 Datasets

3.3.1 D14 dataset

After adding the found instances to our previous selection we stratified and split our dataset again in a test set (95%) and a test set (5%). The split was done at this point to ensure a complete separation between the train- and test set when training new classifiers. The split ratio is quite uncommon, but we felt that a larger more diverse test set was more important since the 5-fold cross-validation also uses sections of it as validation sets. Using the approach outlines in the preprocessing section we extracted patches from all instances in both sets. Since not all containers are equally large, the amount of patches obtained from each instance is inconsistent. Therefore, the patches are stratified again to ensure an equal amount of patches for each class in both the train- and test set. The final stratified train set contains 55.244 unique patches and the final stratified test set contains 4.844 unique patches of 14 different classes. This final dataset will be referred to as the "D14" dataset.

3.3.2 D13 dataset

The D13 dataset is a subset of the D14 dataset with exactly the same instances for the different classes. The difference between the two datasets is that all instances of the AVOCADO class have been removed in the D13 dataset so that it only contains 13 unique classes. This dataset is primarily used to test the possible extendability of a network. The stratified train set contains 51.298 unique patches and the stratified test set contains 4.498 unique patches.

Chapter 4

Methods

The first section of this chapter will focus on the metrics used for evaluation. Following sections are concerned with the experiments each applied to one of the research sub-questions. They consist of a small introduction, some relevant background info and conclude with the practical setup of the experiment.

4.1 Evaluation metrics

The metrics used for reporting on the results are the normalized recall, precision and F1 scores. All values are initially calculated for the results on the test set per class for each fold using equations 4.1, 4.2 and 4.3 respectively. The results for the different classes are then averaged per fold to obtain representative averages of fold performance. These averages are used to calculate the overall mean recall, precision and F1 scores with their corresponding standard deviations. The final results are reported as percentages.

$$recall_{class} = \frac{true_{positives}}{total_{positives}} \tag{4.1}$$

$$precision_{class} = \frac{true_{positives}}{true_{positives} + false_{positives}} \tag{4.2}$$

$$F1_{class} = 2 \cdot \frac{precision_{class} \cdot recall_{class}}{(precision_{class} + recall_{class})} \tag{4.3}$$

4.2 State-of-the-art architecture comparison experiments

Convolutional architectures can be quite similar, but slight differences might give one architecture an edge over another when X-ray image data is concerned. This section compares the performance of several different state-of-the-art architectures and aims to answer the following research sub-question:
Are there significant performance differences between different types of architecture and networks when using X-ray images?

4.2.1 Residual network architecture

The residual network (ResNet) architecture was first proposed as a solution to the difficulty of training deep neural networks [13]. Previous research found that the depth of a neural network was connected to its image recognition performance (on the ImageNet dataset specifically). Deeper networks are however more susceptible to the problem of exploding/vanishing gradients and therefore more difficult to train. Each layer is expected to learn some function that maps its inputs to the correct output. In the paper the term unreferenced function is used for this mapping, which is denoted as $H(x)$ [13]. They propose the addition of residuals to the function to create a residual mapping function, denoted as $F(x) = H(x) - x$ and rewritten to $H(x) = F(x) + x$. Their hypothesis being that it is easier for the network to learn this residual mapping compared to the mapping of the unreferenced function alone. Should the unreferenced mapping be easy to learn the residual would simply become zero and therefore not be a problem. They realized the residual mapping function by adding "short-cut" or skip connections between layers which allows outputs to completely skip layers. A schematic of this architecture's basic block can be seen in Figure 4.1.

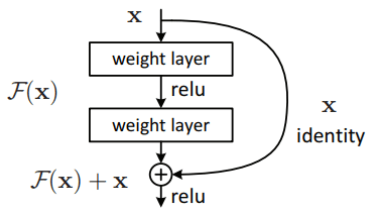

Figure 4.1: A schematic for the ResNet architectures basic block [52, 13].

4.2.2 MobileNet V2 architecture

The original MobileNet was designed to be an efficient light-weight architecture that could be used on mobile devices [44]. The main innovation is the introduction of depthwise separable convolutions. By using this type of convolution a kernel is not mapped to a single value in the next layer, but is instead mapped to three values in the next layer, where each value corresponds to a color channel.

The second version of MobileNet introduced additional improvements such as the residual connections and bottleneck layers that can be found in ResNets [44].

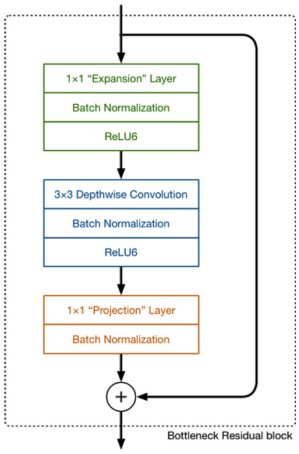

Figure 4.2: A schematic of the MobileNet V2 architectures basic block [26].

4.2.3 Inception V3 architecture

The family of deep learning networks referred to as versions of Inception are based on the original GoogleNet architecture [47, 48]. They are named after the inception module which is a key element in their architecture. Where normal CNN's simply stack more layers to create a deeper network, the Inception architecture uses Inception modules instead. The layers in a normal CNN have a single kernel size per layer, which fixes the resolution and detail that can be detected. The inception module uses multiple convolutional kernels with different sizes in each layer allowing it to detect shapes of different complexity at each level. A schematic of an inception module can be found in figure 4.3. In contrast to the other two architectures the Inception V3 network expects inputs of shape 3x299x299. However, during the construction of our dataset we divided the larger image in 3x224x224 patches. The build-in Pytorch upscale transformation operation for images was used when opening an image in order

to obtain inputs with the correct dimensions. It should be noted that operation adds about an hour of runtime to each training epoch and might have an impact on the network's overall performance.

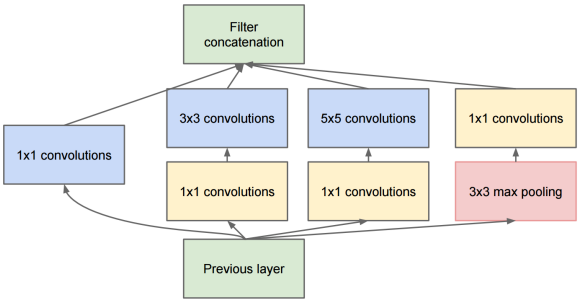

Figure 4.3: A schematic of the Inception architectures basic block [19].

4.2.4 Setup

Exploratory testing is done fore every entwork to determine suitable parameters and the approximate region where a network's train- and test loss converge. When the approximate region is found a stratified 5-fold cross-validation is performed with the maximum epochs slightly above the approximated region to ensure the absence of underfitting. During training the networks is validated and saved after every epoch. The final evaluation is done by assessing performance of the fully trained network on the test set for each fold.

4.3 Pretraining experiments

The variety of goods that are transported via shipping containers provides a wide range of different patterns that an image classification system will need to learn. With relevant labelled data being limited and unlabelled data in abundance pretraining could be an effective way to jumpstart the classification process. This experiment is focused on the performance differences between untrained and different pretrained networks and aims to answer the following research sub-question:
Does pretraining provide any significant performance benefit?

4.3.1 ImageNet

For evaluating classification and object detection networks their performance on the ImageNet has become the standard measure. The reason that ImageNet has become

the standard is its easy accessibility, its quality and the sheer size. With more than 14 million manually annotated images, 1 million of which also contain bounding boxes, in over 20.000 categories there is a lot of variety and options for researchers [6]. Additionally, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), focused on classification and object detection on ImageNet data, is held annually which promotes the use of ImageNet [41]. The contest uses a subset of 1000 non-overlapping classes instead of the total ImageNet dataset. Naturally this means that many state-of-the-art networks are trained on ImageNet even if it is not their primary purpose. Furthermore, the variety in the ImageNet dataset has proven to be a relatively good base for pretraining [7].

4.3.2 Setup

The first experiment focuses on the generalization performance of the pretrained layers which will be frozen, thus only the classifier section will be trained. The second experiment will evaluate the applicability and optimal performance, here all layers are trained and finetuned from the start. Both experiments will be performed using stratified 5-fold cross-validation and final performance will be evaluated using the average recall, precision and F1 metrics.

4.4 Encoding experiments

For administrative, legal and tax purposes all goods that are imported and exported have to be categorized. As explained before, the nomenclature is a hierarchical system that categorizes every type of good and assigns it with a corresponding code. The nomenclature uses five levels with a maximum of 100 options per level for over 10 billion possible unique combinations. The final code is a composition of a five numbers between 1 and 99 for a total length of 10 digits. By using this encoding it is easier to (partially) categorize items based on their features and properties. This is however not generally the way how multi-class classification works in deep learning. This experiment aims to answer the following research sub-question by comparing three types of encoding:
What is the impact of different types of encoding and sizes of outputs on overall performance?

4.4.1 Multiple-attribute encoding

The multiple-attribute encoder is different from the normal label encoder due to it using a given format of labels that does not need to be based on the labels present in

the dataset. The size of the encoding vector for the label encoder is determined by the amount of labels present in the dataset, while the multiple-attribute encoder is given a specific size or encoding up front. The vector given to the multiple-attribute encoder might contain labels that are not present in the dataset. This means that sections of the encoding vector are initially unused as some of the labels are not present in the initial dataset. However it also means that the encoding vector does not need to change shape when new labels are introduced by extending the dataset if they were included in the original given vector. In theory this would allow updating a trained ANN with new instances instead of requiring full retraining of the network due to the shape mismatch between the original and the new output vector. This can be useful when all possible labels are known, but the dataset does not include all possible labels.

4.4.2 Setup

For every network exploratory testing is done to determine suitable parameters and the approximate region where a networks train and test loss converge. When the approximate region is found a (stratified) 5-fold cross-validation is performed with the maximum epochs slightly above the approximated region to ensure the absence of underfitting. During training the networks are validated and saved after every epoch. The final evaluation is done by assessing performance of the fully trained network on the test set for each fold. There is no weighting done for the different levels of labels, meaning that the higher level is just as important as the lower levels.

4.5 Explainability experiments

As mentioned in section 3.1 the supplied dataset does not contain any labelled object locations or segmentation, this is a recurring theme in many real-world datasets. In real-world applications of image classification systems a form of explainability is required, especially when it is used in combination with human operators.

Similar to the assumption that only the content of a container is supposed to contribute to the classification, we can assume that the intrinsic feature used for classification will be part of the object that is has to be classified. Since features are learned by the system instead of created by hand, it cannot be assumed that these learned features are intrinsic to specific goods. This is a problem because the system might use unrelated features that are part of an underlying pattern for classification of specific goods. Using only the loss and accuracy metrics will not be able to guarantee that this assumption holds.

Since the location of these features is coupled with the location of the classified

object, an assessment that evaluates the overlap of the object location and the location of features used for classification can provide an extra validation method to the researchers. Object detection and segmentation use the same assumptions and make the system predict the location in addition to classification. It would seem a superior method, but unfortunately a method that requires labelled object locations or segmentation maps which our dataset is lacking.

Therefore different visualization/localization methods will be needed in this use-case. The experiments will be an exploratory qualitative analysis of different visualization techniques, which will be explained in the sections below, in order to try and answer the following research sub-question:
Is it possible to sufficiently explain the classification decisions of a system in a way that enhances an analyst's analysis?

4.5.1 Non-overlapping blocks approach

With this approach the container image is divided into equally sized patches of 224x224 pixels without overlap. Each patch is used as a single input and all pixels of that patch are given the corresponding classification. This is similar to classifying individual images, but in this instance they just happen to be regions extracted from a larger image.

4.5.2 Pixel-voting approach

This is a fairly old technique which can be used on images larger than the networks input size (although it is also possible to use on images equal to the input size by using padding). It works similarly to the sliding window approach of a CNN, by sliding the input window over the complete image using a specific stride. For every input a classification is made and this classification is given to every pixel of the input, which it will use to tally the amount of classifications for every class. After all classifications are completed most pixels in the image will have more than one classification tallied. The class with the highest tally for a pixel is seen as the optimal classification. When every pixel is assigned a class, the corresponding class colors are projected on the original image.

4.5.3 Class activation map approach

ANNs are a type of black box which reasoning is hard to grasp. Class activation maps aim to visualize the reasoning of ANNs. It does so by tracing the gradients of a specific class node in the output layer back through the nodes of the network [45]. Using this backwards pass it will arrive at the input layer of the network,

where it can assess which input (pixels) are part of the features that contributed to a specific classification. By visualizing the activation of these pixels using a heat map, it becomes clear which parts of the image the system uses as a basis for its classification.

4.5.4 Setup

Since there is no localization data present in the dataset it is not possible to perform an empirical analysis of each methods performance using objective measures. However, the goal of these methods is to aid the interpretation of classifications that a neural network makes by humans. Therefore, we feel that a subjective approach is sufficient for this kind of initial exploratory research. Nonetheless there will be several visual criteria such as continuity of pixels or curves and the approximate ratios of correct, incorrect or missed pixels that can be used to subjectively assess the performance of the different approaches. The qualitative analysis will use containers that are filled in different degrees with a variety of goods. Every method will be applied to each container and the resulting visualizations will be assessed in the next chapter.

Chapter 5

Results

In this chapter the results of the experiments are presented using the same order as the previous chapter. Section 5.1 will report on the performances of different architectures using the D14 X-ray dataset. Section 5.2 focuses on the usefulness of dataset specific pretraining, while section 5.3 is concerned with the effects of encoding. Finally, section 5.4 analyzes different explanation providing methods.

5.1 State-of-the-art architecture comparison experiments

In this experiment the performance of three different architectures on the D14 X-ray dataset were evaluated. The final results of the experiment can be seen in Table 5.1. All three networks seem to be capable of learning the relevant patterns present in the D14 dataset. The inception V3 architectures performance should be superior according to the ImageNet performance comparison of these architectures [2]. The other two architectures are both smaller than the inception V3 architecture in terms of trainable parameters. The ResNet 18 and Mobilenet V2 architectures are around the same size, however according to the ImageNet comparison Mobilenet V2 should perform slightly better [2].

Architecture	**Epochs**	**recall in %**	**precision in %**	**F1 in %**
ResNet 18	15	99.38% +- 0.15	99.38% +- 0.14	99.38% +- 0.15
MobileNet V2	25	98.93% +- 0.77	98.97% +- 0.71	98.91% +- 0.80
Inception V3	10	99.15% +- 0.30	99.17% +- 0.27	99.15% +- 0.30

Table 5.1: Architecture results on the test set reported as percentages +- one standard deviation.

Table 5.1 shows the recall, precision and F1 results of the different architectures on the test set of the D14 dataset. When we look at the overall results is does not appear that the Inception V3 architecture's performance is superior to the other two

architectures tested. The overall best performing architecture is the ResNet18, which also shows the lowest variance over its averaged results. The worst performing architecture with the highest variance in its averaged results was the MobileNet V2 which we expected to perform worse than the Inception V3 architecture and better than the ResNet18 architecture. According to the confusion matrix, it appears that the MobileNet V2 architecture had more trouble distinguishing between the PINEAPPLES and MANGOES class compared to the other two architectures. Using the T-test we found that the differences in performance between all the different architectures was not significant ($p>0.05$). For every architecture the classification performance on the OTHER class was around 5% less than their classification performance on any of the other classes.

5.2 Pretraining experiments

In this experiment the effect of pretraining on final performance is tested using the D14 dataset. The final results are shown in table 5.2 while figure 5.1 show the averaged accuracy and loss results per epoch.

Condition	**recall in %**	**precision in %**	**F1 in %**
Random initialization	99.00% +- 0.14	99.01% +- 0.15	99.00% +- 0.15
ImageNet + finetuned	99.38% +- 0.15	99.38% +- 0.14	99.38% +- 0.15
ImageNet + frozen	95.41% +- 0.26	95.43% +- 0.29	95.33% +- 0.29

Table 5.2: Results comparing the performance on the test set of different types of network initializations after training for 15 epochs, reported as percentages +- one standard deviation.

The results in table 5.2 show that after training for 15 epochs all configuration surpass 99% performance in recall, precision and the resulting F1 score. Looking at the initial difference between the untrained and pretrained networks shows that the ImageNet finetuned network does perform significantly better than the untrained baseline ($p < 0.05$). This is in line with our expectations, but it also shows that pretraining results in only a minor (although significant) performance advantage compared to training a network from scratch.

Figure 5.1 shows that the untrained network converges around the 7th epoch while the ImageNet pretrained network converges in the 5th epoch, suggesting that the ImageNet pretraining reduces training time by 28.6% in this particular case. It also shows that the training and validation loss of the frozen variant of the ImageNet pretrained network never converges and never reaching optimal performance.

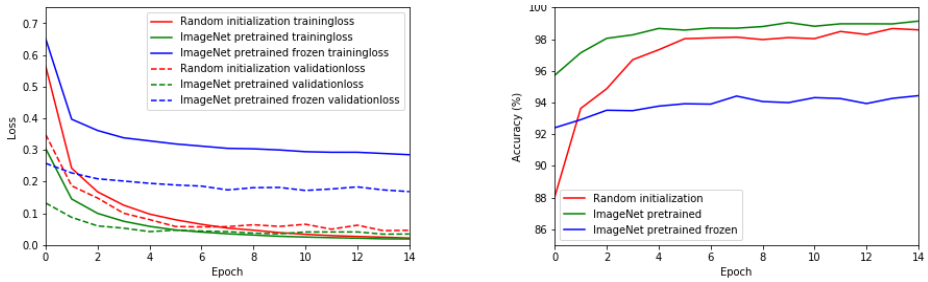

Figure 5.1: The training and validation loss per epoch for all three configurations (left) and the corresponding accuracy's per epoch (right).

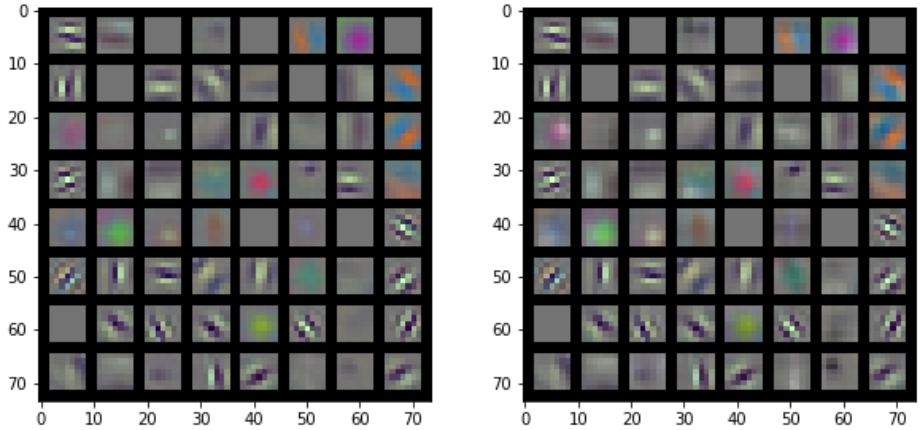

Figure 5.2: The kernels of an ImageNet pretrained ResNet18's first convolutional layer before (left) and after (right) training for 15 epochs on the D14 dataset.

The frozen version of the pretrained ResNet18 network manages to reach test accuracy performance upwards of 95%, which is significantly better than expected. When we look at Figure 5.2 it becomes clear why it did so well. The kernels (filters) show only minor differences, meaning that the features learned during ImageNet pretraining were good generalizations to start with. Although the performance of the frozen version with 95% averages is above expectation, it is significantly worse compared to both the finetuned variant and the untrained baseline. This indicates that the finetuning procedure is required to get optimal performance when using an ImageNet pretrained network.

5.3 Encoding experiments

In this experiment the effect of different types of encoding on network performance was tested. We expect that the output size and amount of active nodes required for a correct classification to be a significant influence on the performance of a network. The final results of the experiments can be seen in Table 5.3.

Encoding	**recall in %**	**precision in %**	**F1 in %**
One-hot	99.38% +- 0.15	99.38% +- 0.14	99.38% +- 0.15
Label	99.31% +- 0.16	99.39% +- 0.17	99.34% +- 0.17
Multiple-attribute	99.35% +- 0.17	99.43% +- 0.15	99.38% +- 0.16
Multiple-attribute 13 + single	99.14% +- 0.40	99.21% +- 0.35	99.10% +- 0.42
Multiple-attribute 13 + full	99.37% +- 0.07	99.43% +- 0.07	99.33% +- 0.07

Table 5.3: Results comparing the performance on the test set of different types of output/label encodings after training for 15 epochs, reported as percentages +- one standard deviation.

The output sizes differ significantly between the three different types of encoding, but there is no difference in the amount of active encodings. Only 14 different encoded values are used which represent the 14 classes of the D14 dataset. This could explain the minor differences in recall, precision and F1 results between the different encoding formats. It is possible that the network that uses sparse-label encoder might see the unused encoding space as irrelevant. Since the idea of sparse-label encoding is extendability it is important to find out whether this is the case. The last two rows in Table 5.3 show the results of a ResNet18 network using sparse-label encoding that was initially trained for 15 epochs on the first fold of the D13 set. The 14th class was added and additional training was performed in exactly the same manner as the other experiments. The row multiple-attribute 13 + single reports the performance on the test set after training one additional epoch on the D14 training set. This is equivalent to training for 16 epochs on 13 of the classes and training for 1 epoch on the 14th class. The row multiple-attribute 13 + full reports the performance on the test set after training for 15 additional epochs on the D14 training set. This is equivalent to training for 30 epochs on 13 of the classes and 15 epochs on the 14th class. Using a comparison of means T-test we determined that there were no significant differences between any of the different encoding methods.

5.4 Explainability experiments

For illustration purposes the methods are applied to a complete container image. Meaning that only the patches obtained from a single container image are used by the different visualization methods. The results are laid over the original container image to give an indication of what an analyst might see. For each sample there will be at least four images. The first image will display the original container image as extracted by the container extraction network. The second image will show the basic non-overlapping blocks and their classification. Each block contains the classification and probability of that classification in text. If the probability is higher than 95% the block is filled in with the color of the corresponding class. The third image displays the per pixel voting method where each pixel is colored according to its classification. A stride 28 pixels in both the horizontal and vertical direction was used to obtain the results. The final image(s) will show non-overlapping blocks with GradCAM visualizations of one specific class.

5.4.1 Single type of good completely filled

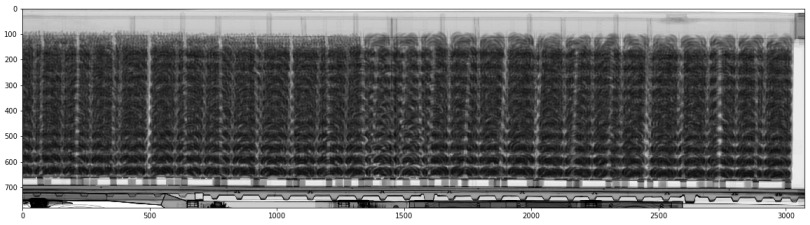

Figure 5.3: Container extracted from a full image which is completely filled with bananas.

The first example can be seen in the above Figure (5.3), it is a standard forty-foot equivalent unit (FEU) container that is loaded with boxes of bananas stacked on pallets. There are only small dead spaces between the supporting blocks of the pallets and the area between the top of the containers and the transported goods.

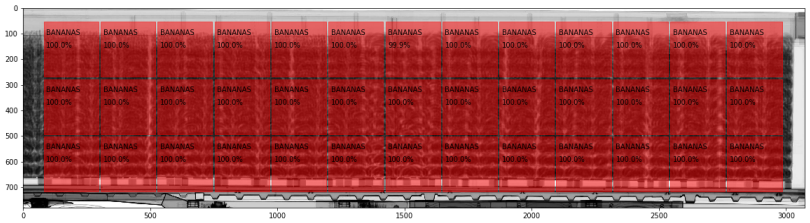

Figure 5.4: The extracted container from Figure 5.3 with the classification results of the non-overlapping blocks approach laid on top.

Figure 5.4 shows classification using the non-overlapping blocks approach, it is clear by the color and the text that all blocks got the same classification of BANANAS. Since the container was completely filled with bananas, all classifications were correct and with a probability > 95 %. Every pixel is thus classified as banana, but not every pixel contains bananas. This means that there is a substantial amount of pixels that should have been classified as being EMPTY (for deadspace between bananas and pallets) or OTHER (for the pallets). Since this is one single block there are no curves to follow, but there is continuity in the pixels.

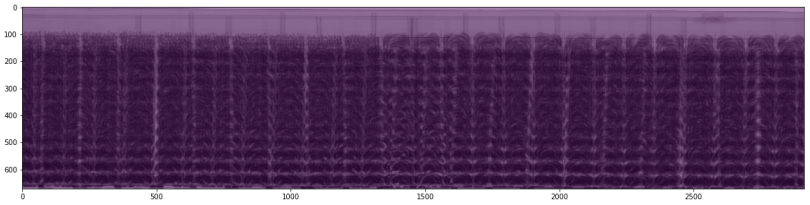

Figure 5.5: The extracted container from Figure 5.3 with the classification results of the pixel-vote approach laid on top.

Figure 5.5 show the pixel vote approach to classification. The image is filled in completely with one color, which indicates that every pixel has voted for the same classification. While confirming the classification of the non-overlapping blocks it does little more to improve interpretability for an analyst. The evaluation remarks of the non-overlapping blocks hold here as well.

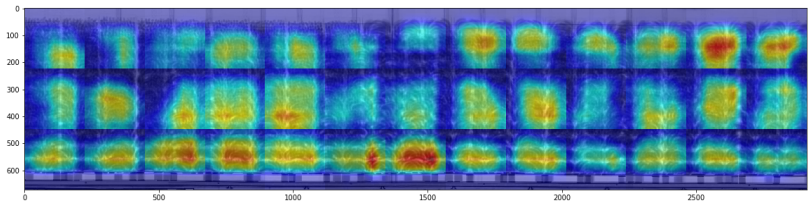

Figure 5.6: The extracted container from Figure 5.3 with a heatmap, created using the GradCAM approach, showing the activation of the BANANAS class laid on top.

Finally Figure 5.6 using the GradCAM approach gives a more dynamic interpretation. The heatmap shows clearly that the activation for the BANANAS class is concentrated around the bananas itself, which light up red and yellow. The pallets and top of the container do not contribute to the classification and are blue. Each block shows some curves and edges, although this does not hold for the container as a whole. The continuity of pixels, curves and edges is contained within blocks and does not extend to the whole container. While the dead space and the pallets are not classified as banana, with this approach there is a significant amount of pixels that does not light up with high activation while being bananas. Thus the precision is better compared to the other two methods, while the recall is considerably worse. It is however subjectively easier for an analyst to interpret why the algorithm chose to classify a particular block using the heatmap.

5.4.2 Containing several types of goods

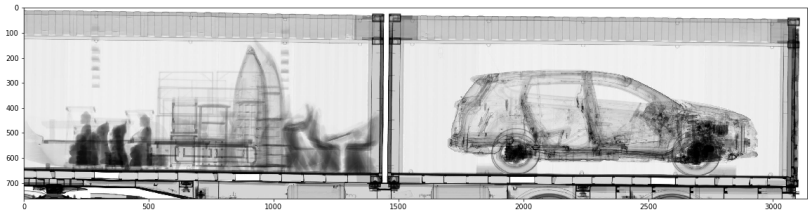

Figure 5.7: A combination of two containers extracted from a full image, one containing a CAR and the other containing several OTHER goods.

This example a combination of two twenty foot equivalent unit (TEU) containers containing a car and several goods of the OTHER class, which can be seen in the above Figure (5.7). The goods in the left most container show a large variety of types.

It appears that a large portion is wooden furniture such as chairs, tables, shelves and chests. Other notable goods are what appears to be a small boat or canoe split in two and several (Buddha) statues.

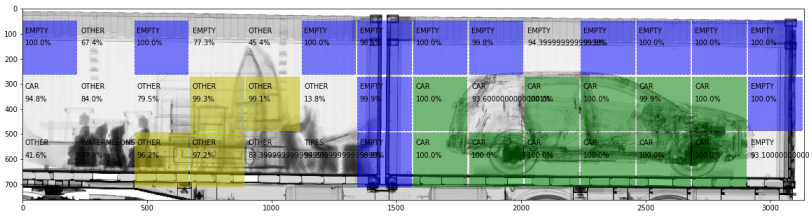

Figure 5.8: The extracted container from Figure 5.7 with the classification results of the non-overlapping blocks approach laid on top.

It is clear from Figure 5.8 that the non-overlapping blocks approach has significantly more trouble with confidently classifying containers with several types of mixed goods. The classifications and probabilities of the blocks for the right container are significantly better than those of the left container. All blocks of the right container are classified correctly even the three blocks that did not reach the confidence threshold of 95%. The left container has significantly more non colored blocks indicating that the classification probabilities are lower than the threshold. The classifications of the blocks in the left container that exceed the 95% probability are however all arguably correct. There are three notable blocks in the left container, the second block in third row with a classification of WATERMELONS. Two other blocks almost exceed the 95% threshold, namely the first block in the second row for the class CAR and finally the sixth block in the third row with class TIRES. These three blocks in addition to the second block in the first row that classifies it as class OTHER are incorrectly classified. The remaining blocks, while not reaching the 95% threshold, are arguably classified correctly.

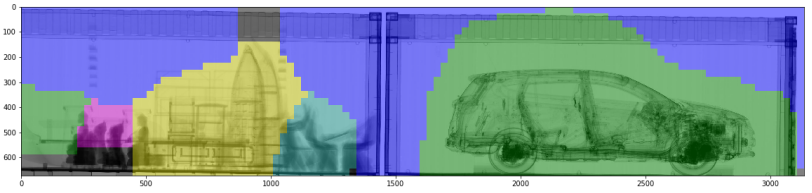

Figure 5.9: The extracted container from Figure 5.7 with the classification results of the pixel-vote approach laid on top.

Using pixel voting many of the previously uncertain regions can be classified with

probabilities exceeding the 95% threshold as seen in Figure 5.9. The magenta color is used for classifications of the WATERMELON class while cyan is used for the TIRES class. Only the second block of the first row which was classified incorrectly in Figure 5.8 has been changed to empty and is now correct. The regions where the other incorrectly classified blocks of Figure 5.8 were present are still classified incorrectly.

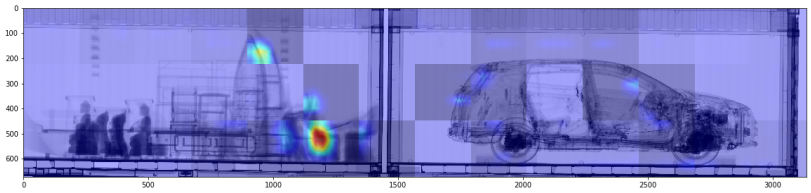

Figure 5.10: The extracted container from Figure 5.7 with a heatmap, created using the GradCAM approach, showing the activation of the TIRES class laid on top.

Since there are no tires present in either of the containers with the exception of the tires on the car in the right container we can conclude that the activation for the TIRES class present in Figure 5.10 is incorrect. In the right container there is some slight activation focused around the wheels and tires of the car, while in the left container the focus is largely on the stacks of chairs at the right side of the container.

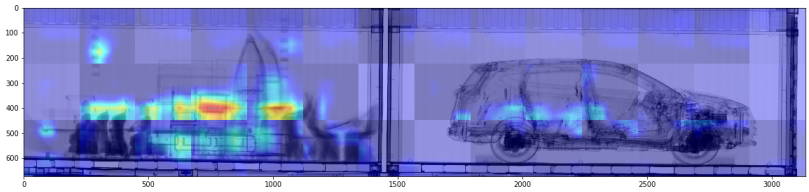

Figure 5.11: The extracted container from Figure 5.7 with a heatmap, created using the GradCAM approach, showing the activation of the OTHER class laid on top.

According to Figure 5.11 the activation for the OTHER class is mostly focused on the interior of the car which includes the seats and trunk in the right container. In the left container it is spread out amongst the different objects, although there are some cold spots around the statues and chairs located on both sides of the containers centre at the bottom. The heatmap also lacks any OTHER class activation around the top halves of the split canoe.

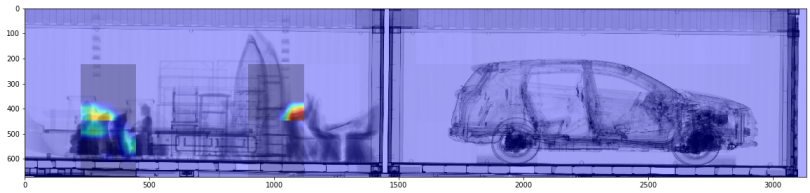

Figure 5.12: The extracted container from Figure 5.7 with a heatmap, created using the GradCAM approach, showing the activation of the WATERMELONS class laid on top.

The classification of WATERMELONS is quite interesting and Figure 5.12 shows that only two small regions contributing to all of the activation for this class. When we look closer at that region which was actually classified as watermelons in Figure 5.8 and 5.9, we can see it is mostly centered around the heads of the statues in the left container. The similarity between the shape and size apparently produced enough activation for a confident classification of the WATERMELON class in that region. While this classification is incorrect, the reasoning is clearly explainable and easy to interpret for humans.

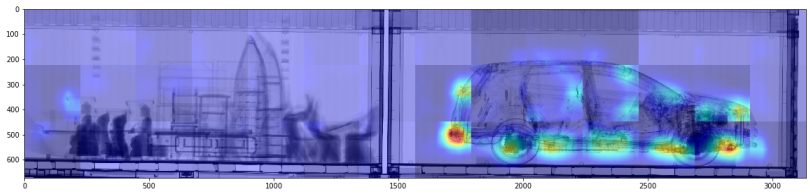

Figure 5.13: The extracted container from Figure 5.7 with a heatmap, created using the GradCAM approach, showing the activation of the CAR class laid on top.

The activation for the CAR class if focused primarily on the car in the right container as shown in Figure 5.13, with some slight spurious activation just left of the statues in the left container. The strong activation around the car is promising, it seems that most of it is focused on the chassis and wheels.

Chapter 6

Discussion

In this thesis the we investigated several aspects of applying deep learning algorithms in a real-world setting. The dataset that Dutch customs provided and the corresponding challenge to develop a pilot for an explainable classification system were the ideal testbed. The problems that one encounters when a dataset is not curated such as missing, incorrect or non-uniform labels make applying deep learning difficult. Another problem often faced is the scalability aspect when there is a large amount of possible labels, which certainly is present when using the customs nomenclature. Even if all these practical problems are solved there are the reliability and explainability aspects that need to be satisfied when people are expected to use and trust a deep learning system. In the following section we will answer the research questions presented in the introduction. Finally we will end with our conclusions and recommendations for further research.

6.1 Research questions

6.1.1 Architectures

Are there significant performance differences between different types of architecture and networks when using X-ray images?
While X-ray scans are in the image domain they are significantly different from photographic images. We tested three different state-of-the-art architectures with the D14 dataset. The ResNet18, MobileNet V2 and Inception V3 architectures were all able to converge on excellent performance around the 99% range in terms of recall, precision and the combined F1 score. Using a T-test we determined that there were no significant differences in performance between the different networks.

6.1.2 Pretraining

Does pretraining provide any performance benefit?
Using an ImageNet pretrained ResNet18 network does significantly improve performance in terms of both convergence speed and final performance compared to training

such a network from scratch. Using a T-test we determined that the difference of 0.38% in performance was significant. Additionally, the training and validation loss of the ImageNet pretrained ResNet18 network would on average converge two epochs before the initially untrained variant, which reduces the required amount of epochs with 28.57% from 7 to 5 epochs.

When freezing all layers except the final classifier in an ImageNet pretrained ResNet18 network the time required for a single epoch was reduced significantly with around 60%, which is expected due to the reduction in trainable parameters. This is also expected to have a significant impact on final performance since the convolutional layers due to being frozen cannot fine tune on new features and patterns. The results indeed showed a significant reduction of performance averaging around 95.4% compared to the 99%+ of the unfrozen models. While this reduction is significant, it is not as much as we had expected and it is still rather respectable.

6.1.3 Encoding

What is the impact of different types of encoding and sizes of outputs on overall performance?
To answer this question we compared performance of three different types of encoding on the D14 dataset. The one-hot encoding, which is the standard for multi-class classification, that had an output size of 14 for the D14 dataset. This type of encoding has low information density and therefore scalability, but it is very easy to apply. Unfortunately it does not allow for extendability.

The second type of encoding was label encoding, which is the standard type of encoding for multi-label classification, which had an output size of 26 for the D14 dataset. It information density is higher and its scalability is significantly better than one-hot encoding, but it requires multiple labels per instance and can only extend with new classes if the corresponding labels are already present.

Finally, the multi-attribute encoding format uses the standard label encoding, but extends it to include all possible labels at initialization even if the selected dataset does not contain these labels. This approach had an output size of 300 for the D14 dataset, but is in theory extendable. When we evaluated the performance of the different types of encoding on the D14 dataset the differences were minor and insignificant according to the T-test. This is a great result for the multi-attribute encoder and the extendability aspect of it. Using a network with multi-attribute encoding trained for 15 epochs on the D13 dataset as a base and resuming training with the D14 dataset allowed us to extend our network with an extra class and resulted in performance similar to a network trained for 15 epochs on D14 with only one additional epoch of training.

6.1.4 Explainability

Is it possible to sufficiently explain classification decisions of a system in a way that enhances an analyst's analysis?
We analyzed three approaches with various degrees of complexity and different methods of visualization.

As expected the non-overlapping blocks approach, which was the simplest, did not aid significantly in explaining classifications. A reduced probability per block is the only indication that an analyst might need to look more closely at a specific region of the container. The only other benefit of this approach is that it provides a rough estimate of the size and volume of the cargo.

The pixel voting approach is a variation on the first approach with blocks overlapping significantly. Most pixels, except for pixels around the edges, receive multiple classification and use majority voting for a final classification. This approach allows for better defined regions and localization of classifications within a container. This makes it easier for an analyst to narrow down specific objects or regions of interest, but it does not contribute significantly more to the interpretability of a networks classification. Additionally this approach provides a more accurate estimate of the size and volume of the cargo.

Finally, the GradCAM method showed a different approach to the problem by using a heatmap to visualize the regions that directly contributed to a specific classification. While this approach really visualizes the reasoning of the classification system and consequently increased the interpretability, by highlighting important features such as engine, fuel tanks, body and wheels for the CAR class in Figure 5.13. There was activation from incorrect classes, however the intensity and location of this activation provided by the heatmap makes it significantly easier for humans to assess compared to the non-overlapping blocks and pixel-voting approaches.

6.2 Conclusion

There is much to learn from the practical challenges that are encountered when trying to apply deep learning methods to real-world problems. In this thesis we have only looked at a small subset and found many interesting alternative approaches that could be useful both for deep learning theory and in practice. While there might be large performance differences between architectures on academic datasets this does not always apply to other datasets. Pretraining is an effective method of jumpstarting your classification algorithms and ImageNet provides excellent initial learned features, but manually finetuning is required for optimal performance. If the dataset allows it, there are significant benefits to using alternative ways of encoding your labels for

both scalability and extendability. Finally, we found that post-hoc methods can be used effectively to improve explainability or provide limited segmentation of images.

6.3 Recommendations

While many of the different approaches have shown interesting results, these were obtained using a rather small dataset. In order to validate the findings our primary recommendation is to test these methods on larger datasets with significantly more classes. This can be done manually with the approach used to create the D14 dataset, but it is also possible to use K-means clustering on the OTHER class.

We suggest to include contraband classes in these larger datasets, which as of now has been a considerable hurdle and finding contraband is one of the main goals of Dutch customs. Obviously the problem is that there is a rather limited amount of scanned examples that contain contraband, which can be solved by propping containers with contraband and scanning them. Another option for Dutch customs is to extend the dataset with new examples of found contraband, but the drawback is that this can take a significant amount of time. Additionally limited augmentation of the dataset by digitally inserting models of contraband has shown interesting results [32].

To enhance interpretability of classifications in images we suggest mapping the hierarchical nomenclature of the customs domain to a color space. This will allow analysts to see the boundaries between two distinct classes more clearly, while reducing the difference between classes that are not that different such as frozen versus fresh bananas. While object detection networks offer more precise localization compared to classification approaches it requires additional labelling which is often not present. The explainability approaches shown in this thesis, with some additional processing, might provide a way to automatically label or segment image examples for object detection networks.

Furthermore, the frequency of block classifications and the area provide by the pixel-voting approach can be used as a base for statistical analysis. Outlier detection could be as simple as looking for significant differences in area or the blocks classification frequency between shipments of the same good or by the same company.

Bibliography

[1] Belastingdienst. Costsimport. https://www.belastingdienst.nl/wps/wcm/connect/bldcontentnl/belastingdienst/prive/douane/goederen_ontvangen_uit_het_buitenland/van_organisaties_en_bedrijven/moet_ik_belastingen_bij_invoer_betalen, 2020. [Online; accessed].

[2] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] R Caruana, DL Silver, J Baxter, TM Mitchell, LY Pratt, and S Thrun. Learning to learn: knowledge consolidation and transfer in inductive systems. In *Workshop held at NIPS-95, Vail, CO, see http://www. cs. cmu. edu/afs/user/caruana/pub/transfer. html*, 1995.

[5] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 5(4):455–455, 1992.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[8] Belastindienst Douane. staff. `https://download.belastingdienst.nl/douane/docs/annual_report_customs_administration_netherlands_do3761z71fdeng.pdf`, 2020. [Online; accessed].

[9] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[10] Megan Garcia. Racist in the machine: The disturbing implications of algorithmic bias. *World Policy Journal*, 33(4):111–117, 2016.

[11] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2, 2017.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-aduaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[15] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[17] Peter Jackson. *Introduction to expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[18] Peter Jackson. Introduction to expert systems, 1998; cornelius t. leondes, fuzzy logic and expert systems applications (neural network systems techniques and applications), 1998; george f luger. *Artificial Intelligence Structure and Strategies for Complex Problem Saving, Addison Wesley*, 2005.

[19] jyotishp. Perceptron schematic. `https://i.stack.imgur.com/zTinD.png`, 2020. [Online; accessed].

[20] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[23] Erin Ledell. Multi-layer perceptron schematic. `https://github.com/ledell/sldm4-h2o/blob/master/sldm4-deeplearning-h2o.Rmd`, 2020. [Online; accessed].

[24] Simon Mackenzie. Organised crime and common transit networks. *Trends & issues in crime and criminal justice*, (233):1, 2002.

[25] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[26] Khan Muhammad. MobileNet V2 architecture. `https://www.researchgate.net/profile/Khan_Muhammad7/publication/330893810/figure/fig2/AS:836308662484992@1576402931299/Main-building-block-of-MobileNet-V2-architecture.jpg`, 2020. [Online; accessed].

[27] Omar M'Haimdat. convolution. `https://miro.medium.com/max/1356/1*yqrHxzA9nwL3rTQE_Ts_3w@2x.jpeg`, 2020. [Online; accessed].

[28] P Russel Norvig and S Artificial Intelligence. *A modern approach*. Prentice Hall, 2015.

[29] Port of Rotterdam. throughput. `https://www.portofrotterdam.com/en/our-port/facts-and-figures/facts-figures-about-the-port/throughput`, 2020. [Online; accessed].

[30] Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

[31] PyTorch. Pretrained state-of-the-art networks. `https://pytorch.org/docs/stable/torchvision/models.html`, 2020. [Online; accessed].

[32] Thomas W Rogers, Nicolas Jaccard, Emmanouil D Protonotarios, James Ollier, Edward J Morton, and Lewis D Griffin. Threat image projection (tip) into x-ray images of cargo containers for training humans and machines. In *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, pages 1–7. IEEE, 2016.

[33] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[34] Wilhelm Conrad Röntgen. On a new kind of rays. *Science*, 3(59):227–231, 1896.

[35] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[36] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[37] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, pages 1–4, 2005.

[38] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[40] David E Rumelhart, Bernard Widrow, and Michael A Lehr. The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–93, 1994.

[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[42] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[43] Wojciech Samek and Klaus-Robert Müller. Towards explainable artificial intelligence. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 5–22. Springer, 2019.

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[45] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[46] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[47] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[48] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[49] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.

[50] Godfried T Toussaint. The use of context in pattern recognition. *Pattern Recognition*, 10(3):189–204, 1978.

[51] SEABORNE TRADE. Developments in international. *REVIEW OF MARITIME TRANSPORT*, page 2, 2014.

[52] Lazaros Tsochatzidis, Lena Costaridou, and Ioannis Pratikakis. Resnet architecture. `https://www.researchgate.net/publication/331738899/figure/fig2/AS:736349447536642@1552570797128/Building-block-of-ResNet-19.ppm`, 2020. [Online; accessed].

[53] Carsten Weerth. Hs 2007: Notes of the tariff nomenclature and the additional notes of the ec. *World Customs Journal*, 2(1):111–115, 2008.

[54] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[55] Amin Ghazi Zahedi. Perceptron schematic. `https://www.researchgate.net/profile/Amin_Ghazi_Zahedi/publication/266493320/figure/fig2/AS:667619657003015@1536184338567/Signal-flow-graph-of-the-perceptron-A-single-perceptron-is-not\-very-useful-because-of-its_W640.jpg`, 2020. [Online; accessed].

Chapter 7

Appendix A

7.1 Large objects with dead space

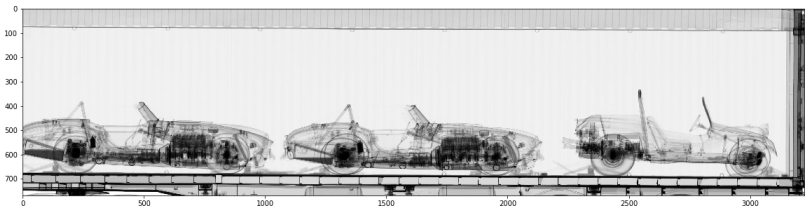

Figure 7.1: Container extracted from a full image which contains three convertible cars.

This example is a container filled with large objects, which can be seen in the above Figure (7.1). It is a standard FEU container with three large objects that are convertible (sports) cars, the rest of the container is empty resulting in a significant amount of dead space both in between the three large objects and between the objects and the container itself.

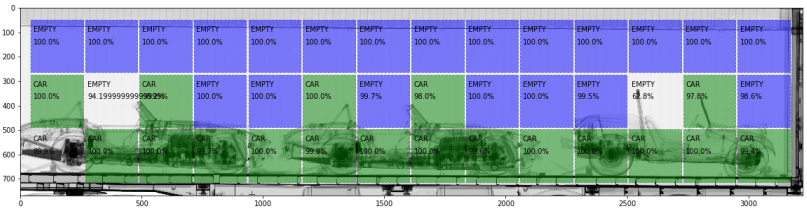

Figure 7.2: The extracted container from Figure 7.1 with the classification results of the non-overlapping blocks approach laid on top.

In contrast to the previous example the results of the non-overlapping blocks approach shown in Figure 7.2 give a slightly more diverse range of classifications and

probabilities. All blocks are either classified as CAR or EMPTY, there are three blocks that have a probability $< 95\%$ for their preferred class and are thus not filled in. The two blocks in the second row would have classified as EMPTY, but due to containing some small parts of the cars the probability did not reach the threshold of 95%. The first block in the last row would have classified as CAR, but for some reason did not. There mostly is continuity between the blocks, although arguably to much since there is not a clear distinction between the different cars when only looking at the blocks classification. There are edges to follow, but being rather large blocks the edges are quite rough.

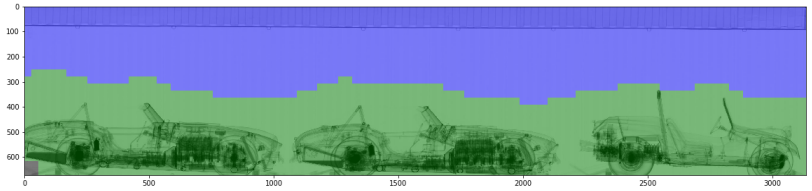

Figure 7.3: The extracted container from Figure 7.1 with the classification results of the pixel-vote approach laid on top.

The third image of Figure 7.3 displaying the per pixel voting approach softens up these edges somewhat. It is clear that the division between the top classification for EMPTY and the bottom classification for CAR follows the contours of the cars. It also filled in much of the sections that could not be classified confidently that were present in the non-overlapping blocks approach and improved the continuity as a whole. The recall is excellent since every pixel containing a car is classified as such, while the precision is similar to the non-overlapping blocks approach. Still a significant amount of pixels that should be classified as EMPTY are in fact classified as CAR.

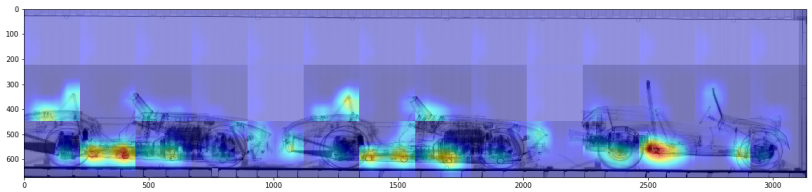

Figure 7.4: The extracted container from Figure 7.1 with a heatmap, created using the GradCAM approach, showing the activation of the CAR class laid on top.

The GradCAM approach which can be seen in Figure 7.4 shows an interesting

phenomenon. Many of the blocks classified as empty still show slight amounts of spurious activation for the CAR class while all of these blocks have a probability > 95% for the EMPTY class. The reason for these blocks showing activation for the CAR class might be that containers and cars are both made of similar types of metal, thus providing similar features. The blocks that were classified as CAR show activation on relatively small sections that contain dense features such as engines, wheels and fuel tanks. In this example the continuity is not exceptional. The activation in blocks that are classified as CAR do however show the reasoning quite clearly by being focused on important distinguishing features.

7.2 Single type of good with deadspace

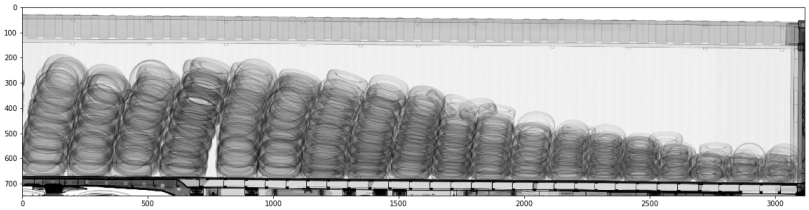

Figure 7.5: Container extracted from a full image which is half filled with tires.

This example is a FEU container half filled with tires, which can be seen in the above Figure (7.5). The tires are stacked at different heights, starting with the highest stacks on the left side and ending with the lowest stacks on the right side of the container. The stacks of tires are not uniform and bend slightly to the right.

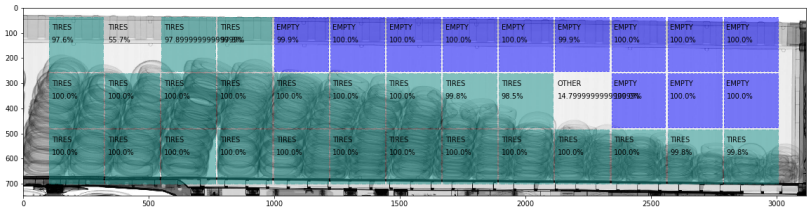

Figure 7.6: The extracted container from Figure 7.5 with the classification results of the non-overlapping blocks approach laid on top.

The non-overlapping blocks approach shown in Figure 7.6 already gives quite a good classification of the containers contents. There are many similarities to the large objects example, since there are two classes and a significant amount of deadspace.

Just like that example there are only two classes and all blocks are classified correctly. There are two blocks which probabilities did not reached the required threshold, but are still arguably classified correctly. There is a clear, albeit rough, boundary between the two different classes.

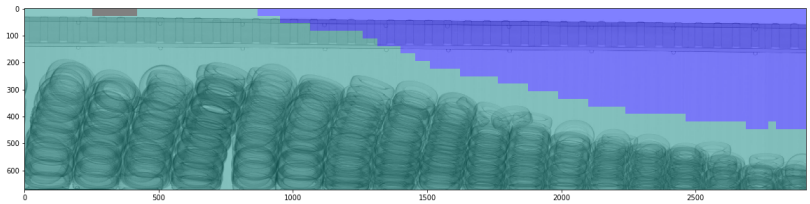

Figure 7.7: The extracted container from Figure 7.5 with the classification results of the pixel-vote approach laid on top.

The pixel-vote approach shown in Figure 7.7 exhibits the same improvements over the non-overlapping blocks for this example compared to the large objects example. The areas that the non-overlapping blocks approach was not able to confidently classify because the probabilities did not reach the threshold of 95% are classified and the boundary between the two classes is less rough. Additionally this boundary is offset quite significantly from the non-empty class present.

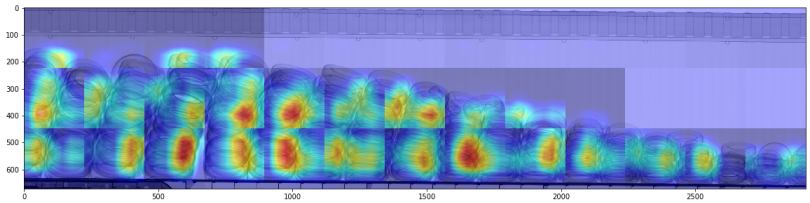

Figure 7.8: The extracted container from Figure 7.5 with a heatmap, created using the GradCAM approach, showing the activation of the TIRES class laid on top.

In Figure 7.8 the GradCAM approach shows that almost all of the activation for the TIRES class is focused on the tires themselves. Even though there are sections that contain tires, but have little activation the coverage is very good.