Evaluation of Convolutional Neural Networks for Image Quality Classification based on Synthetic Data

Joar Karlgren Gustavsson, Vendela Nigård

Master's thesis
2022:E28

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

CENTRUM SCIENTIARUM MATHEMATICARUM

Evaluation of Convolutional Neural Networks for Image Quality Classification based on Synthetic Data

Vendela Nigård
Joar Karlgren Gustavsson

Spring, 2022

LUND
UNIVERSITY

Supervisors:
Karl Åström, LTH
Anton Holm, Axis Communications
Rikard Lindahl, Axis Communications

Examiner:
Alexandros Sopasakis, LTH

Abstract

In camera production the image quality is of utter importance. Several tests during the production ensure this high quality. In this thesis the possibility of creating a final test, that classifies the image quality with the help of machine learning, specifically convolutional neural networks, was investigated. The data used was made up of synthetic, simulated images with commonly observed quality defects. Eight different network architectures were evaluated on four different types of data sets; two data sets for binary classification, one for multi-class classification, and one data set containing real, non-simulated images. The results were promising with a test accuracy of 0.999 for the binary case with a two-stream network with a DenseNet base. For the multi-class classification the best test accuracy was 0.989 with the same network. The results showed that there is a high potential for the use of convolutional neural networks for classifying image quality. For a large enough data set a simple convolutional network would be sufficient, achieving similar results as the best network. The networks could handle most of the investigated defects, but seemed to have a problem with blemishes - dust/sensor defects, which is why it would be recommended to have another test for this defect. It was also concluded that there is a need for gathering a larger amount of real images for training since the results on the real data set were at best 0.594 for binary classification. This showed that the networks trained on the simulated images did not translate well to real images.

Acknowledgements

We would like to thank our manager, Erik Fors, at Axis Communications for the opportunity and the support. We would also like to thank our supervisors Anton Holm and Rikard Lindahl for their constant support, help with debugging and general assistance throughout the whole work. Additionally we would like to thank the rest of the staff at Axis Communications that we have been in contact with; our team - PSTP, the PET-imaging team, and Willie Betschart, for being forthcoming, helpful, for providing tips along the way, and last but not least for making us feel welcome. Finally we would like to show our gratitude to our supervisor at LTH, Karl Åström, for the feedback and help you have provided.

The computations and data handling were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE) partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

Popular Science Summary

Is machine learning the next big thing in camera production?

As the use of AI is increasing in all aspects of today's society, camera production companies are starting to take an interest in it as well. These companies go to great lengths and apply multiple tests to ensure that the cameras shipped to their customers are of the highest quality. We have looked into replacing some of these tests with machine learning to make them more objective and standardized. Machine learning is a subsection of AI where a computer algorithm can learn to make decisions based on experience. Can this be used in production without compromising the quality? Early tests performed on synthetically created data show promise as 99 out of 100 cameras were correctly categorized as approved or disapproved, even though some quality defects were harder to detect than others.

After several tests have been performed, the last test in camera production is to examine the general image quality by taking a picture and evaluating it. However, imagine that you have looked at images all day, at one point you are bound to get tired and miss something. Additionally, two different people can choose to evaluate images differently, resulting in subjective results. This is a problem that could be solved by using machine learning, making the test objective and standardized. It will also ensure that the cameras that do go out to the customers are of the desired quality.

This is where neural networks come in. A neural network is a machine learning algorithm that aims to mimic the human brain. Using several connections via "neurons" the network learns from experience by backtracking based on the results of its decisions. It can learn to recognize patterns which then can be used to make decisions, for example, classification. Classification is when the network can decide if an image belongs to one class or another depending on the contents of the image. In our case that would mean that the network classifies the image as "passed" if the image is of good quality or "failed" if it contains image quality defects. Moreover, the network could be trained to say what type of defect the image contains.

We were able to train several different networks of various complexity, with the help of annotated examples, to be able to recognize defects and classify whether an image was of good or bad quality i.e. passed or failed. The results were indeed very promising for synthetic data, with accuracies of up to 99.9%. The synthetic data was made up of good quality images from other tests that are used in production, on these images the varying defects were simulated. The simulation algorithms were created from scratch and the results were compared to a handful of real images. However, for the small amount of real images available the results were not as good, reaching only an accuracy of 54%, indicating that the synthetic data was not representative of the real data. It is clear that more real, authentic images are needed before these networks can be put to practical use.

Is machine learning to be used in camera production then? Yes, there is a high potential for the use, however, there is still work to be done. There are several possibilities of extending and further developing the algorithms that we have created. Last but not least it is a matter of implementing a functioning program for production.

Contents

Abbreviations

- **ANN**: Artificial Neural Network
- **CNN**: Convolutional Neural Network
- **DNN**: Deep Neural Network
- **IQA**: Image Quality Assessment
- **FR-IQA**: Full-reference Image Quality Assessment
- **RR-IQA**: Reduced-reference Image Quality Asessment
- **NR-IQA**: No-reference Image Quality Assessment
- **PSF**: Point Spread Function
- **ReLu**: Rectified Linear Unit
- **SGD**: Stochastic Gradient Descent
- **RMSprop**: Root Mean Squared Propagation
- **Adam**: Adaptive Moment Estimation
- **Nadam**: Nesterov-accelerated Adaptive Moment Estimation
- **TP**: True positives
- **TN**: True negatives
- **FP**: False Positives
- **FN**: False Negatives
- **ROC**: Receiver Operating Characteristics
- **SCN**: Small Convolutional Network
- **IV3**: Inception V3 Network
- **RN**: ResNet Network
- **DN**: DenseNet Network
- **EN**: EfficientNet Network
- **DCN**: Dual Convolutional Network
- **DDN**: Dual DenseNet Network
- **DEN**: Dual EfficientNet Network
- **GAN**: Generative Adversarial Network

1 Introduction

For many companies that produce cameras or other visual systems, the image quality of the products is of great importance. Tests that run during production secure high image quality before the products are shipped out to the customers. In order to have a high reliability in these tests they need to be standardised, repeatable and objective.

One of these companies currently employs a ”general image quality test” which aims to test the final image quality of different products. They however wish to move towards a more objective test method which does not solely rely on an operator and is also more consistent and repeatable. One way to achieve this could be to take images of the same target/scene each time and make use of a machine learning algorithm to decide whether or not an image, and therefore a camera, should be classified as passed.

1.1 Related Work

With an increase in all things digital the inspection of digital image quality and detection of defects have become large research areas to ensure the quality of image systems and products. There exists several methods for detecting specific defects like blemishes (dust/sensor-defects), blur, dead pixels and noise [31], [30], [7], [16]. However, for detection of more general defects and multiple defects at a time, more complex models need to be considered.

The research field of more general errors is often called Image Quality Assessment (IQA). The aim of this is usually to get a quantitative score, consistent with human visual assessment, which describes the quality of the image. These methods are often used to find defects which occur when images are converted and compressed in different ways, for example when converting an image to the jpg format. The loss of information can be characterized by defects such as noise, blur and dead pixels. These defects can be similar to the defects that occur when producing cameras although these defects found in production are more complex. The field of IQA can be divided into three different sub-fields: full-reference (FR-IQA), reduced-reference (RR-IQA) and no-reference (NR-IQA).

FR-IQA refers to methods that have a reference image with good quality as well as the same image with bad quality. The score is then decided based on a comparison between the reference image and the bad quality image with the help of features extracted from, for example, pixel- and structure information. Methods based on the mean square error, signal-to-noise ratio [5], structural similarity [40] and others can then be applied. The RR-IQA methods use only partial information from a reference image, for example the wavelet transform to get a natural image statistic model which can be compared between the reference image and the distorted image [39].

NR-IQA has no reference image at all, but only derives information from the distorted image. This is more similar to the problem presented in this report as no reference images exist for the images taken during camera production. Since there is no reference to compare to, the research is largely focused on different deep learning methods which can learn what features to look for themselves. This approach allows the extraction of more complex image structures [42]. Since the problem is contained to images, convolutional neural networks are especially useful. However, a big problem is that there is often a lack of annotated images to use for training. To create more training data several papers use the patch-method which means that they divide each image into a number of patches which is fed into the network after which an average of the patch score is calculated to obtain the quality score [19]. A problem could then be that there are no labels for the individual patches, and if a serious defect is local, averaging may not give a representative score.

Since a blurry photo can be recognized with the gradient, one approach that is used is a two-stream convolutional neural network that takes the image and its gradient as inputs [19], [41]. The outputs of the two parallel networks are then concatenated and a quality score decided. Another common approach is to make use of pre-trained networks, like AlexNet or Resnet50. These are networks that have been pre-trained on the data set ImageNet which is used for object recognition. The output from these can then be fine-tuned with a few additional layers [4], [20]. Other methods include using a GAN (Generative Adversarial Network) to generate a reference image and then make use of FR-IQA [32].

1.2 Aim

In an attempt to solve the issues regarding objectivity and standardization in the production, the aim of this thesis was to examine the possibility of replacing the human operator with a classification algorithm based on machine learning. The focus was on implementing this software using artificial neural networks (ANNs) and since the data examined solely consists of images, specifically convolutional neural networks (CNNs) were used.

Though the above mentioned methods in section 1.1 work well for some defects, more defects are present in the data for this thesis. This means that a new evaluation needs to be done for this data. In the production of cameras any defect being present means that the camera needs to be taken apart and remade which also means that a quality score is irrelevant but rather a ”passed” or ”failed” label would be more suitable. Additionally, it could be of interest to classify the type of defect.

In order to do this, synthetic data consisting of images with defects in them were simulated and several different neural networks were evaluated on this data. The questions investigated were if there is a potential to use convolutional neural networks for both binary and multi-class classification of image quality, if there are specific networks that work particularly well for the data and if there are defects that are easier or harder to classify. Moreover, the trained networks were evaluated on real data to see if synthetic data can accurately represent the authentic data.

1.3 Delimitations

This thesis is delimited to training networks on only synthetic data with an image test target as background, for binary and multi-class classification. Only one defect per image is considered. The thesis does not explore the use of semantic segmentation.

1.4 Contribution

The workload has been evenly distributed, as both parts have been actively participating in all aspects of the thesis.

2 Background

2.1 Camera Defects

Several different defects with different causes can occur in images. Some defects are due to the conditions of the environment the image has been taken in (relative motion, weather, lights etc.) while others are due to errors in the manufacturing of the camera. Some of the more common defects found when producing cameras are blemishes (dust/sensor defects), blurring, noise, dead pixels, and pixel column or row errors. These mentioned defects are explained in the following sections.

Blurring

Blurring is a defect where the focus of the image is off, see figure 1. It can be viewed as a bandwidth reduction in an image which is the result of a non-ideal image process. The defect can be caused by an optical system that is out of focus (out-of-focus blur), when the relative motion of the camera and its environment is faster than the speed of the shutter (motion blur) or by atmospheric turbulence [22]. The optical system can be out of focus due to its settings or even in the most perfect conditions blurring can be caused by light diffraction, sensor resolution or lens aberration. A model that describes the blur and resolution is the so called Point Spread Function (PSF). The PSF can be viewed as a function that describes the distribution of light in the camera focal plane for a point light source [9]. The blurring of an image can then be modeled as a convolution with this PSF. In a camera with circular aperture the local PSF will be a small disk with an intensity distribution within that disk. If there is no blurring, i.e. an ideal image, the intensity distribution can be modeled as a Dirac delta function. If there is a considerable defocus then the intensity distribution can be modeled uniformly. Motion blur can be modeled with a translation, rotation, change of scale or combinations of the three. Atmospheric turbulence can however be described well by a Gaussian intensity distribution [22]. Blurring can be identified by applying a laplacian filter, i.e. second derivative, and examining the variance of the result [31].

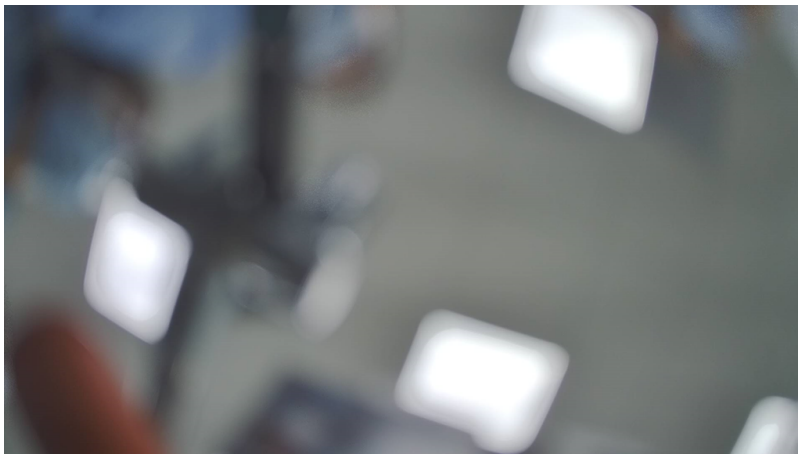


Figure 1: Example of a blurred image.

Noise

Noise is a random variation in pixel intensities in an image which always, on different scales, corrupts an image. An example of a noisy image can be seen in figure 2. It can be due to different reasons such as random absorption, scatter, sensor noise, quantization of data or by other limitations in the system [22]. There are different types of noise that can appear in an image; examples are gaussian, poisson and salt and pepper noise. Gaussian noise is evenly distributed over the image as the sum of the real pixel value and a gaussian distributed noise. Salt and pepper noise appears as white or black pixels randomly distributed over the image [28]. In contrast, Poisson noise is a multiplicative noise which depends on the true pixel value. Each pixel value is a Poisson random variable with the mean value equal to the true pixel value [29]. An example of a method used to detect noise is to calculate the differences in intensity between a pixel and its neighbours, then taking the arithmetic mean and comparing it to a predetermined threshold [30]. This method may not give good results if there are a lot of details in the image.

Figure 2: Example of an image with gaussian noise [2].

Dead Pixels

The ”dead pixels” defect refers to a local region of error in the image where the intensity of one or several of the color channels have been severely reduced or set to zero, see figure 3. The sizes and shapes of these regions can vary a lot and they can often be irregular. Dead pixels are caused by specific pixels not receiving power to one or several color channels. The reason behind why there is no power to that/those pixels can differ, it can be due to a software issue, it can be a failed power connection or it can even be due to physical damage to the sensor. An example of detection of isolated dead pixels is similar to that of noise; looking at the mean of the differences between one pixel and its neighbours [7]. For regions of dead pixels this method can be insufficient.

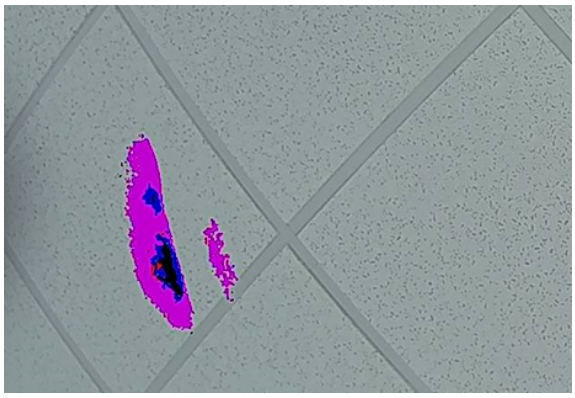


Figure 3: Example of dead pixel regions.

Faulty Pixel Row or Column Errors

Similar to the dead pixel regions discussed above, faulty pixel row or column errors are strips of pixels where some error has occurred. This results in a horizontal or vertical line of pixels with either dead color channels or erroneous channel intensities. An example can be seen in figure 4.

Figure 4: Example of a line of faulty pixels.

Blemishes

A blemish is some kind of dirt which causes a blurred spot in an image which is typically darker than its surroundings. In contrast to dead pixels, the edges of a blemish are typically smooth and therefore harder to detect. The size of the blemish can vary, depending on what it is. They can be artifacts on the lens or sensor caused by dust, water spots or fingerprints. The blurred spot due to blemishes can be more apparent in the center of the image as opposed to in the corners where they may be hard to spot [23]. When examining a camera for blemishes, a photo of a light, homogeneous background can be taken on which the blemishes can be seen, see figure 5. Algorithms to automatically detect blemishes typically make use of a band pass filter which suppresses high and low frequencies, i.e. enhances edges while reducing noise simultaneously. A threshold can then be used to identify the blemishes [16]. This method is however not suitable for non-homogeneous images with a lot of details since the blemishes can not be separated from the background.

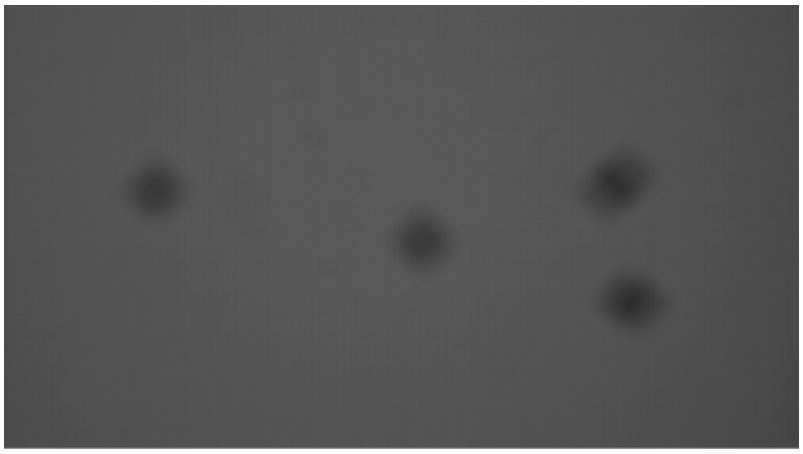


Figure 5: Example of blemishes.

2.2 Machine Learning and Convolutional Neural Networks

Machine learning is a subsection of artificial intelligence where an algorithm can learn for itself from data and typically also improve when exposed to more data. It is based on the fact that computers can learn pattern recognition from data with minimal interference from people. Based on this learning the algorithms can make decisions. First, the algorithm needs to have some data to learn from e.g. numbers or images. Generalising, there exists two branches of machine learning - supervised and unsupervised. Supervised learning is machine learning with human input, e.g. classification or regression, where the algorithm is told what the ground truth is. Unsupervised learning is performed without human input e.g. clustering, here there is no ”right” or ”wrong” answer, the algorithm tries to find patterns or underlying structure in the data. For supervised machine learning methods the data thus has a ground truth to compare to, while an unsupervised algorithm has no ground truth but has to learn how to interpret the data on its own. In machine learning there is most often two phases: the training phase and the prediction phase. During training the algorithm is learning and adapting to given data, and in the prediction phase it uses the model from the training phase to make predictions about new data.

2.2.1 Classification

Classification is a supervised machine learning problem where the goal of the algorithm is to assign the data into different pre-defined classes. For example, imagine a set of images containing cats and dogs, and you want to be able to classify the images as containing a cat or a dog. The algorithm would then train on images which are annotated and the aim would be to correctly classify an image as one of the two classes. Classification algorithms can have binary outputs like in the case with cats and dogs, or they can have multiple outputs as in the case of images containing single digits between 0-9, as for the well known MNIST data set. Either way the output is discrete, most often yielding ones and zeroes for the binary case and integers corresponding to the number of classes in the multi-class case.

2.2.2 Artificial Neural Networks

An artificial neural network (ANN) is a machine learning algorithm which is an attempt to mimic the nervous system of the brain. A very simple one layer network would consist of a single ”neuron”, see figure 6. The neuron consists of a few basic elements. The first element is the input signals. These are usually multiple and often normalized in order to enhance the computational efficiency. The inputs are then weighted one by one to describe the individual input's relevance before they are summarized. Usually a bias is then added before the output is passed through an activation function which will limit the output to a range of certain values. Finally the output is passed on. The output can be described by

$$y = g\left(\sum_{i=1}^{n} w_i x_i - \theta\right) = g\left(\mathbf{wx} - \theta\right), \tag{1}$$

where $g(\cdot)$ is the activation function, w_i the weights, x_i the inputs and θ the bias [8].

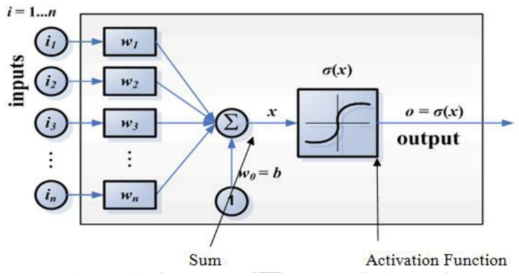


Figure 6: A single artificial neuron [10].

A larger network would usually consist of several connected layers of multiple neurons, changing eq. 1 into

$$\mathbf{y} = g\left(\mathbf{W}\mathbf{x} - \Theta\right) = g\left(\sum_{j=1}^{m}\sum_{i=1}^{n} w_{ij}x_i - \theta_j\right) \tag{2}$$

for each respective layer. If there are multiple layers between the input layer and the output layer - usually denoted as hidden layers, then the network is called a deep neural network(DNN), see figure 7. Classical machine learning requires manual pre-processing and feature selection, while deep learning has the advantage of being able to learn for itself what features are relevant to the problem [3]. Each layer will learn and extract different features from the input data.

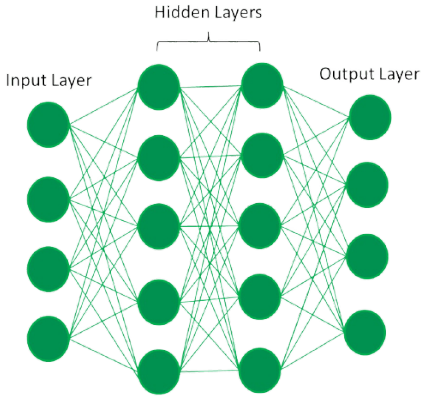


Figure 7: A deep neural network [12].

In the training phase all the weights and biases are initialized to a starting value and the output of each neuron calculated one layer at a time resulting in a final prediction. This process is called forward propagation. The prediction is then compared to the true value, and a loss value based on a loss function is calculated (see section 2.2.4). A small loss indicates a good prediction and a large loss a bad prediction. Based on the loss, backpropagation is initiated by propagating from the output to the input, layer by layer, and adjusting the weights and biases with the help of the loss feedback i.e. the gradient of the loss function (see section 2.2.5). This means that the weights and biases are all trainable parameters which will be adjusted to the specific problem as the network learns what is important. This is repeated for all the training data. When the training is completed the weights and biases are all fixed while doing predictions on new data [24].

2.2.3 Activation Functions

The activation function is an important part of the neural network as it allows the network to become non-linear and therefore learn more complex patterns. It is also used to keep the output bounded in order to prevent computational issues. The selection of activation function has an important impact on how the system behaves.

One common activation function is the sigmoid function, see figure 8. The sigmoid function is a continuous function which maps all values to the range between 0 and 1. The sigmoid function is defined as

$$g(x) = \frac{1}{1 + e^{-x}}. \tag{3}$$

This function is useful when the output should be in the range between 0 and 1, as in the final layer of a binary classification algorithm. However a disadvantage of using sigmoid as an activation function in other layers is that it suffers from the vanishing gradient problem. The vanishing gradient problem arises during backpropagation and the calculation of the loss function's gradient. The loss function's gradient is connected to the gradients of the activation functions at each layer.

If the gradients of the activation functions are too small, the weights of the network will only be subject to small if any changes. This will cause the weights of the last layers (counting backwards from the output layer) to not be able to learn properly. The sigmoid function is also not zero-centered which is undesirable when training the network, due to efficiency issues [26].

Another function that can be used as the final layer in a classification task is the softmax layer. It is usually used when there are several classes and is defined as

$$g(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{c} e^{x_j}} \tag{4}$$

where there are as many neurons in this layer as the number of classes. The softmax maps the output to values between 0 and 1, and can be interpreted as a probability [26].

Another activation function is the tangent function defined as

$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{5}$$

see figure 8. This function also suffers from the vanishing gradient problem since the output is constricted to -1 and 1, however it is zero-centered.

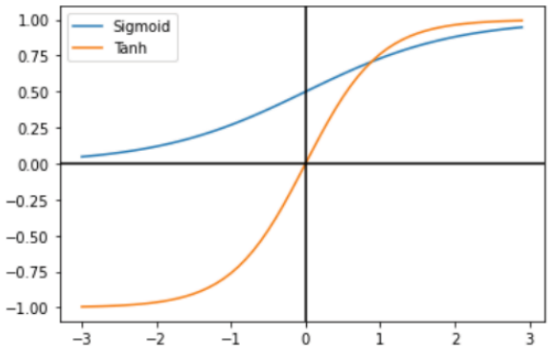


Figure 8: The sigmoid and tangent functions [26]. Copyright © 2020, IEEE

In order to combat the vanishing gradient problem one can use the ReLu (Rectified Linear Unit) function, see figure 9. It is defined as

$$g(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0. \end{cases} \tag{6}$$

It has an easy gradient and does not have a vanishing gradient when performing backpropagation. Some nodes can however ”die” since the output of negative values is zero. To avoid this problem Leaky ReLu can be used which creates a small gradient for negative values [26]:

$$g(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0, \end{cases} \tag{7}$$

see figure 9.

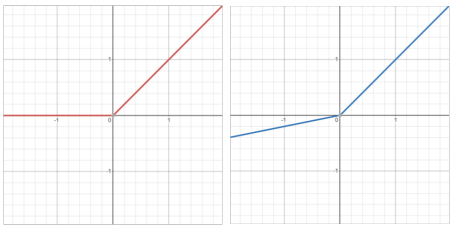


Figure 9: ReLu activation function to the left and LeakyReLu to the right.

Note that the ReLU activation function is not zero-centered and thus this will also lead to inefficient training of the network, however it is still often opted for due to it being simple, fast to compute and handles the vanishing gradient issue as stated above. The Leaky ReLU function does not have as severe zero-centering issues.

2.2.4 Loss Function

In order to define the loss so that the weights can be updated, a loss function is needed. There are different loss functions for different kinds of problems. When using neural networks these can be divided into loss functions for regression problems and loss functions for classification problems. In this thesis only loss functions suitable for classification problems were considered. One common loss function for classification is the binary cross-entropy loss function which is defined by

$$l(y, p) = -y\log(p) - (1 - y)\log(1 - p), \tag{8}$$

where y is the true label and p is the probability for that label, which has been calculated by the last layer. This means that predictions that are confident but wrong are highly penalized. The binary cross-entropy should only be used for binary classification. With several classes categorical cross-entropy could be used instead. Categorical cross-entropy is defined as

$$l(y, p) = -\sum_{i=1}^{c} y_i \log(p_i), \tag{9}$$

where c is the number of classes and $y_i = 1$ if it is the true label and 0 otherwise [25]. The categorical cross entropy loss is the most common one when dealing with multiple classes, variants of categorical cross-entropy exist but are not further discussed here.

2.2.5 Optimization

In order to properly minimize the loss function, an optimization algorithm is used. Just like loss functions, there are many different optimization algorithms that work well in different situations. One of the simplest methods is based on gradient descent. The idea is that the weights and biases get updated iteratively in the direction of the negative gradient of the loss function. The learning rate decides the step size for each update. Since the gradient descent uses all of the data to calculate the gradient, the computational complexity is high. This makes way for stochastic gradient descent (SGD). The idea of SGD is to estimate the gradient based on one sample for each iteration instead of using all of the data. However, because of the random sample selection the variance of the gradients will be large and the optimization may not always converge towards the optimal value [35].

In order to improve upon SGD the concept of momentum can be used. Momentum means that the direction is not only influenced by the current gradient, but by all the previous gradients as well which can speed up the convergence. This can help to escape local minima since even though the current gradient may be zero, the previous ones still have an impact. However, if the momentum is too large the algorithm may completely miss the global minimum [35].

It could also be of interest to change the learning rate as the search gets closer to the global minimum. This can be done by using a decay factor that decreases the learning rate with time. A method which uses this is RMSprop. RMSprop updates the learning rate by dividing it with the square root of the exponential moving average of the squared gradients [35]. A combination of the adaptive learning rate and momentum can be found in the optimizer Adaptive moment estimation (Adam). This method is also one of the more popular methods [35]. To expand on Adam, "Nadam" can be used. Nadam uses the Nesterov momentum which looks at the next gradient as well.

2.2.6 Hyperparameters

While the weights and biases are updated during the training phase, there are some parameters that cannot be trained by the network itself; these are called hyperparameters. The hyperparameters need to be optimized using a validation set which the trained network can be tested on. Depending on the result on the validation data the hyperparameters can be changed manually [13]. Examples of hyperparameters are the optimizer, the learning rate, the dropout constant (see section 2.2.9), the amount of steps between each decay in learning rate, training epochs (the number of times the training data is traversed) and the number of layers in the model.

One of the more important hyperparameters is the learning rate. The learning rate controls how big the steps are during optimization. Too small steps will result in the learning being slow, the optimization may be unable to escape a local minimum and the chosen algorithm may fail to converge. Too big of a learning rate and the optimization might completely miss or jump out of the global minimum. The learning rate will therefore impact how the loss will develop [18], see figure 10. In order to speed up the learning but still increase the chance to converge to the global minimum one can use a decaying learning rate. That way the learning rate will decrease further into the training. This is most commonly applied when using SGD, but it can be combined with the "built-in" decaying learning rates of RMSprop or Adam. One way of implementing a decaying learning rate is to use an exponential decay that reduces the learning rate exponentially over time. Another way is to use a piecewise constant decay of the learning rate at given intervals.

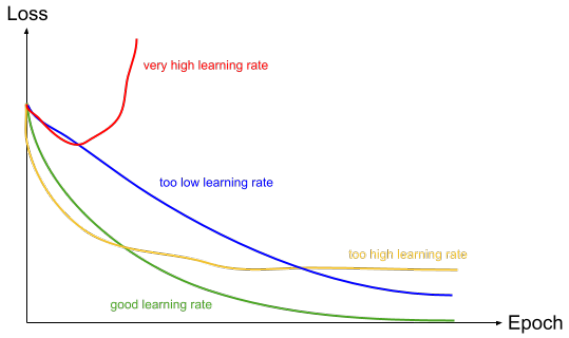


Figure 10: How loss is affected by learning rate, adapted from [27].

2.2.7 Overfitting and Underfitting

When it comes to training a machine learning algorithm, two major challenges are overfitting and underfitting [13]. Underfitting occurs when an oversimplified model, or neural network in this case, is used to describe the problem or perform classification. In the underfitting case, the neural network fails to capture the underlying structure of the data and it fails to minimize the loss function sufficiently. Underfitting leads to poor results for all input data. Overfitting on the other hand is when the model gets too specific. The neural network excels at classifying the training data samples but fails to achieve good results for the validation and test data - the network fails to generalize. In the overfitting case the loss of the training data is very small, however for the validation and test data it is big. This gap between loss values means that the network has learnt properties of the training data that do not benefit the classification of new unseen data [13]. In figure 11 the two cases of under- and overfitting are shown using a binary classification problem. In the rightmost image the "balanced" case is displayed, here the classification algorithm has derived a complex enough decision boundary while still maintaining a desired level of generalization.

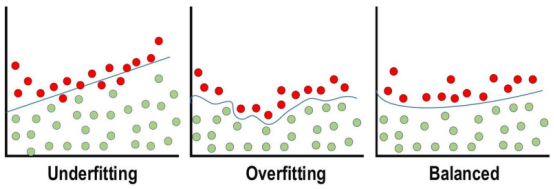


Figure 11: Demonstration of over- and underfitting [3].

2.2.8 Convolutional Neural Networks and Convolutional Layers

Convolutional neural networks (CNN) are a type of neural networks that work especially well for images. It uses the mathematical operation convolution to process, in the case of images, a 2D-grid of pixels. The convolution simply replaces the matrix multiplication (eq. 1) in the specified layers. Discrete 2D-convolution is defined as

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n), \tag{10}$$

where $I(i,j)$ is the input and $K(i,j)$ is a 2D-kernel [13]. The output $S(i,j)$ of the convolution can be referred to as a feature map. The features vary depending on the kernel. Examples of these features can be edges, vertical or horziontal lines, or something entirely different. In a convolutional layer the kernel, which could be smaller than the size of the image, is sliding along the image and the convolution is calculated for each pixel, see figure 12. The size of the kernel and the steps that are taken while sliding, known as "stride", is decided beforehand [13]. A stride of one would mean moving the kernel one pixel for each calculation, as seen in figure 12.

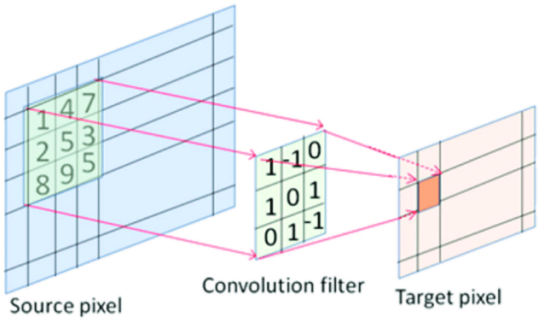


Figure 12: Example of a convolutional operation on pixels [21].

The parameters of the kernel are trained by the network to learn relevant features. The kernel is then shared among all pixels thus reducing the amount of parameters compared to a layer that uses matrix multiplication where each pixel has its own weight and bias [13]. A convolutional layer is usually made up of several filters that work in parallel, each one with its own kernel, allowing it to learn several features at a time.

2.2.9 Other Layers and Operations

Fully Connected Layers

Fully connected layers or ”dense” layers are relatively simple. Each input contributes to all outputs, meaning that every input neuron will be connected to every output neuron, resulting in a vast amount of connections/parameters to be trained. Neural networks that are used for classification will always have a fully connected layer as its final layer with a number of outputs equal to the number of classes used for the classification task at hand. As opposed to convolutional layers discussed above, there is no parameter sharing in a fully connected layer [13]. An image depicting 3 fully connected layers is shown in figure 13 below.

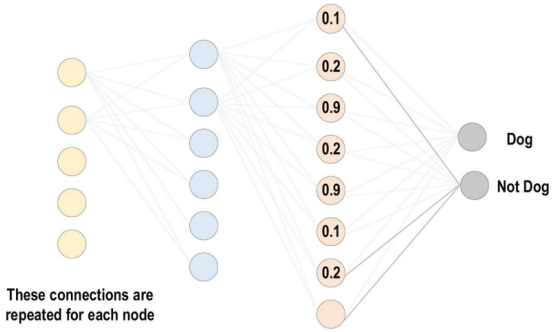


Figure 13: Fully-connected layers [3].

Pooling Layers

In a local pooling operation, a subregion of the inputs to the layer is examined and depending on the nature of the pooling, different operations are performed. In a similar fashion as for convolutional layers, the whole input is traversed using a filter. A max pooling layer looks at a subregion, eg. using a 2x2 or 3x3 filter, and outputs the maximum value of that region, see figure 14. An average pooling layer also looks at a subregion of some given size but outputs the average value instead of the maximum [3]. When using pooling layers it is most common to not access the same inputs more than one time, thus taking a large enough stride each time. There are also global pooling operations where the whole input is evaluated and a single output is produced, global average pooling is exemplified in figure 14.

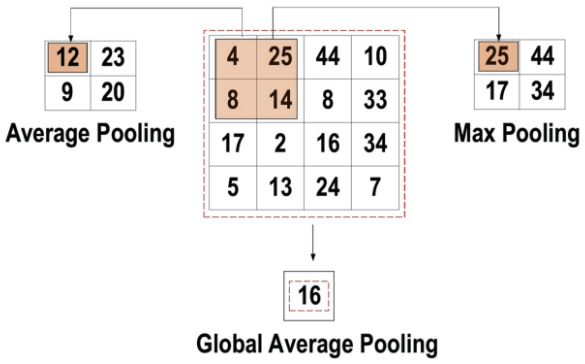


Figure 14: Example of different pooling operations [3].

Dropout

As more and more layers are added to a neural network the number of trainable parameters increase and so does the complexity of the network. As discussed in section 2.2.7 the increase in complexity tends to cause overfitting issues. In order to combat this, a ”dropout” is introduced. Dropout, which is done in every epoch, randomly selects a number of neurons based on a dropout rate in a specific layer and hides/disables them during training. The dropout rate decides the probability p that a neuron will be dropped and the probability $1 - p$ that it will be kept [34]. Since the neurons that are active during training changes between epochs, this forces the layer to generalize better and become more robust. After the training has been completed and the network is used for predictions, all neurons are active and there is no dropout.

Normalization

Normalization or more specifically batch normalization is a method often used when constructing neural networks. Batch normalization scales the values to the range $[0-1]$, stabilizing the computations performed when calculating the gradients of the network during backpropagation. Batch normalization can reduce the training time, allow a higher learning rate and has regularizing effects which can reduce the need for other regularizing operations, such as dropout discussed above [17].

2.2.10 Models

There exists a lot of different models for CNNs. A typical example of a simple CNN is seen in figure 15. Here, a convolutional layer is followed by a ReLu-layer (note that a ReLu-layer is the same as having ReLu as an activation function in the convolutional layer), a pooling layer and finally a fully connected layer before a classification is made. Most CNNs are made up of different blocks with a similar structure as this network.

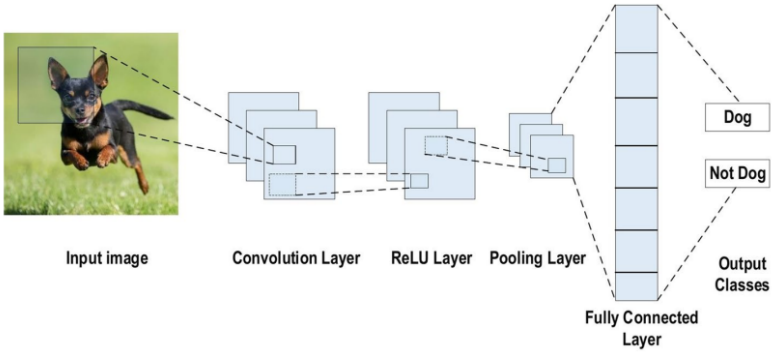


Figure 15: Example of a convolutional neural network for classification [3].

Transfer Learning

Transfer learning is an approach where a model which is originally trained for another purpose is re-used in a new machine learning problem. For example, a CNN trained for object recognition can first be used as a feature extractor in a new classification problem. By only replacing the last fully connected layers in the model it can be tuned to a new problem. This is especially useful when the available data is sparse, since the pre-trained network already has been trained on a large data set. For object classification it is common that the networks are trained on the data set "ImageNet". ImageNet contains millions of images for the purpose of improving object recognition tasks.

Inception V3

Inception V3 is a CNN which is pre-trained on the ImageNet data set. It is constructed to be computationally efficient. It extracts features on multiple levels and uses several blocks of convolutions. The architecture can be seen in figure 16. It includes convolutional layers, max- and average pooling, dropout layers, concatenations and finally fully connected layers [6].

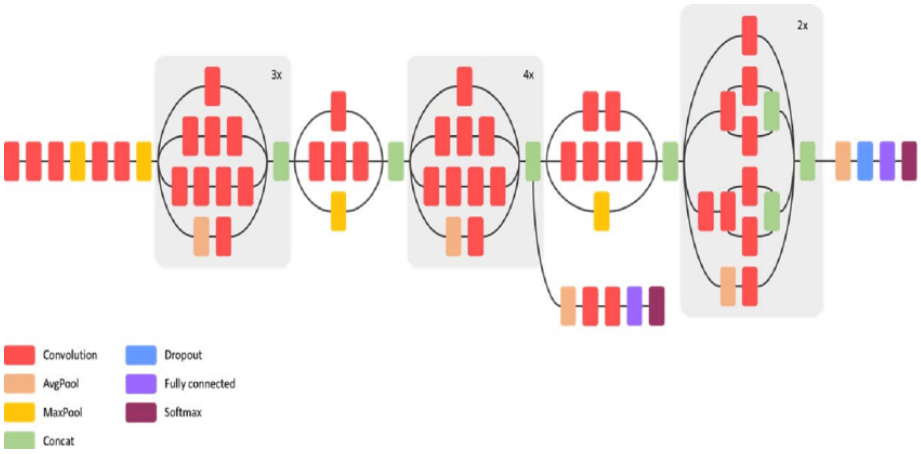


Figure 16: The Inception V3 architecture [6]. Copyright © 2019, IEEE

ResNet

One way to make neural networks better is to make them deeper, thus adding more parameters. When doing so there is however the risk of running into the vanishing gradient problem as the gradient may get smaller the deeper the network is. The vanishing gradient negatively affects the performance of the network. To surpass this problem the Residual Neural Networks was introduced. They make use of "skip-connections" where an input can skip several layers before being added to the output of another layer, see figure 17. A network can then fit a residual mapping instead of the desired underlying mapping. By repeatedly using skip-connections the network can be built deeper, gaining accuracy. This method has shown great results, outperforming many shallower networks [14].

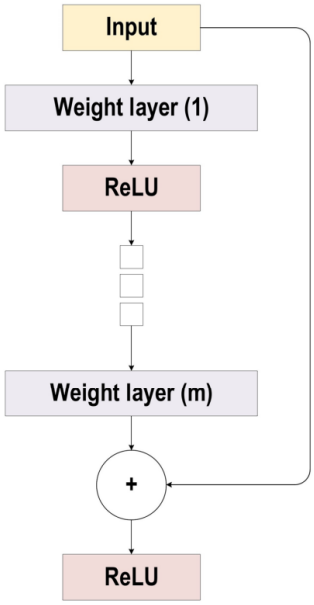


Figure 17: A ResNet block [3].

DenseNet

DenseNet, or Densely Connected Neural Network, builds on the idea from ResNet. The idea is to maximize the flow of information between each layer by connecting all convolutional blocks with each other with skip-connections, see figure 18. In contrast to ResNet, the features are not summarized, but concatenated. This results in blocks containing all features from the previous blocks. The large flow of information and gradients makes the network easy to train. It has also been observed that the network architecture has a regularizing effect, reducing overfitting [15].

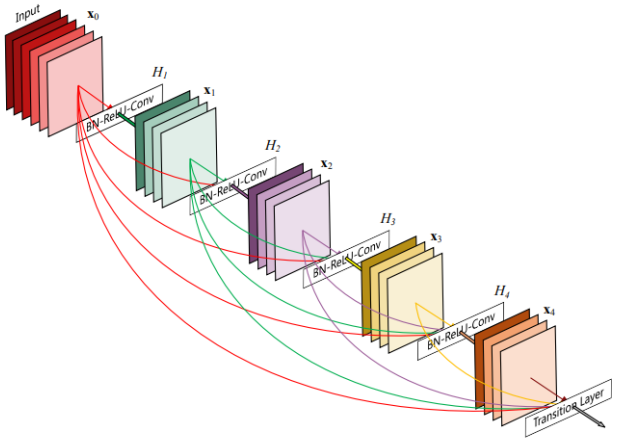


Figure 18: The architecture of DenseNet [15]. Copyright © 2017, IEEE

EfficientNet

The two previous network architectures, DenseNet and ResNet, have achieved good results but at the cost of having a large amount of parameters. Due to the large amount of parameters the networks are costly to train and require a significant amount of computer memory. EfficientNet aims to be more efficient with the amount of parameters while still achieving good results. The architecture is modular and scalable and uses a baseline model - EfficientNetB0, which can then be scaled up to and including EfficientNetB7 with higher model numbers signifying a larger network [36]. This baseline model can be seen in figure 19 below.

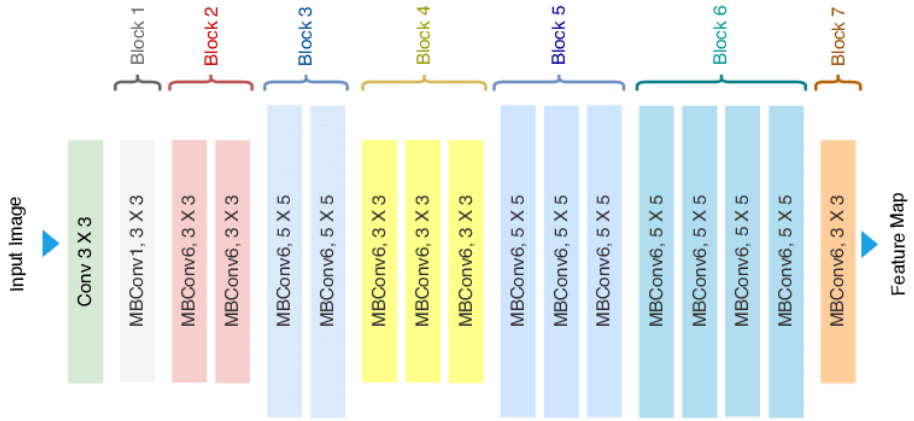


Figure 19: The baseline model for EfficentNet - EfficientNetB0 [1].

Two-stream Network

A two-stream network is a network which takes two inputs and then runs them parallel through one channel each before the respective outputs are concatenated. An example of a two-stream network which is used for image quality assessment is seen in figure 20 [41].

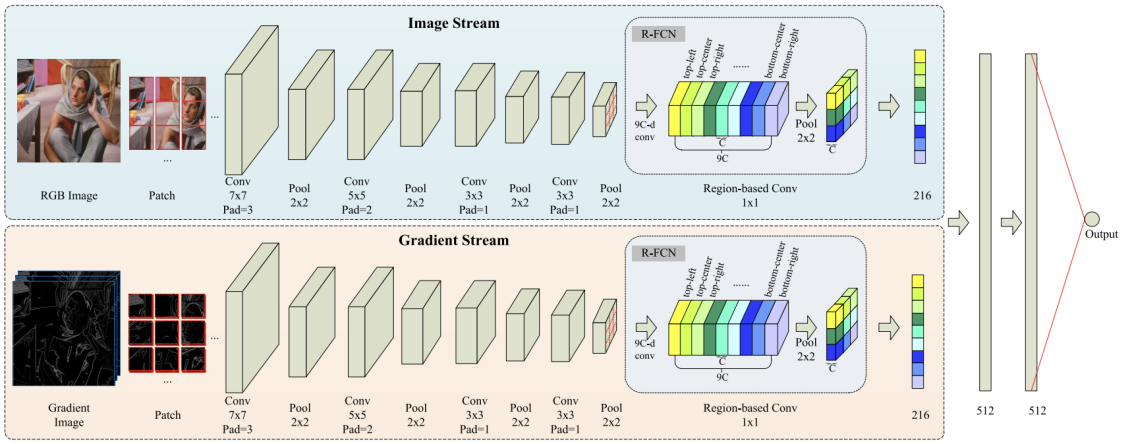


Figure 20: Example of a two-stream network for image quality classification. The inputs are the image and its gradient, divided into patches [41]. Copyright © 2019, IEEE

2.2.11 Evaluation

There are several metrics one can look at when evaluating a machine learning algorithm. When doing so the following definitions are commonly used: true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). The true positives describe how many of the positive samples that were correctly classified as positive, and false positives how many of the negative samples that were falsely classified as positive. True negatives and false negatives are the corresponding definitions for negative samples. With these four definitions it is possible to construct several measures to evaluate the performance.

Confusion Matrix

The confusion matrix shows what predictions where made for the different classes in terms of number of TP, TN, FP and TN. If the rows represent the predicted class and the columns the true class, the perfect algorithm would give a diagonal matrix, see figure 21. Sometimes the transpose of this confusion matrix is used, but the interpretation is the same. If there are incorrect classifications then the pattern will be ”confused” and the matrix is no longer diagonal.

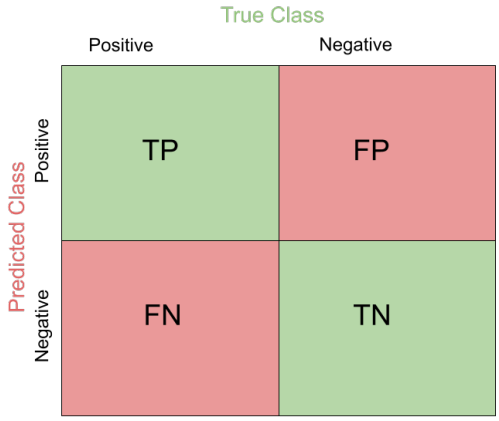


Figure 21: The confusion matrix for binary classification.

Accuracy

The accuracy describes how much of the data that was correctly classified i.e.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}. \tag{11}$$

However, imagine a data set where 90% of the images are of one class and 10% of the other. Classifying all images as the first class would then give an accuracy of 90%, but the algorithm is clearly not good. Therefore the accuracy often needs to be considered together with additional metrics [13].

Sensitivity and Specificity

The sensitivity (or true positive rate) measures the ability to correctly classify a positive sample as positive [3] i.e.

$$Sensitivity = \frac{TP}{TP + FN}. \tag{12}$$

The specificity (or true negative rate) measures the ability to correctly classify a negative sample as negative [3] i.e.

$$Specificity = \frac{TN}{TN + FP}. \tag{13}$$

Depending on the requirements on the algorithm, different values of sensitivity and specificity can be accepted. To view how the sensitivity and specificity are connected, and evaluate what hyperparameters to use one can look at the ROC-curve (Receiver Operating Charactaristics). The ROC-curve has the sensitivity on the y-axis and the false positive rate (1-specificity) on the x-axis. The principle of the ROC-curve can be seen in figure 22. This approach is straightforward on binary classification, but the interpretation gets more complex when there are more than two classes.

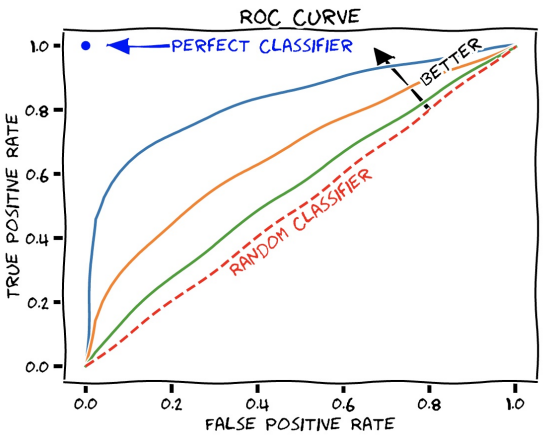


Figure 22: The principle of a ROC-curve. Adapted from [37]

Cross-validation

There is always a risk that the network could be overfitting to the data, and to properly test the network it has to be evaluated on new, unseen data. One way to get a better understanding of how well the network generalizes is to use cross-validation. Cross-validation is a technique where the training data is split into K pieces. During K iterations one piece is used as validation data and the rest as training data, see figure 23. The performance is then evaluated based on the mean of the results of each validation set. The method is a good evaluation technique, however computationally expensive [18].

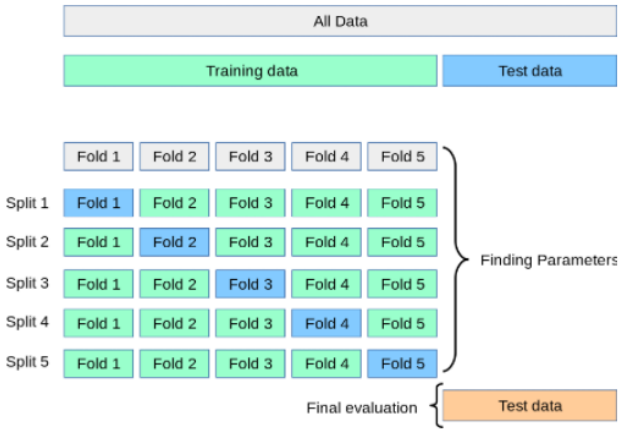


Figure 23: The concept of cross-validation [33].

2.2.12 Network Performance Improvement Techniques

Augmentation

When there is not enough data, augmentation is an easy way to synthetically increase the amount of data and also avoid overfitting. Augmentation is when more data is created by modifying copies of the existing data. For images this could be in the form of random rotation, flipping of the image or change of illumination. That way the network will see more varying data and often be able to perform better [18].

Grid Search

The choice of hyperparameters can drastically impact the performance of the model. Choosing the hyperparameters can take a lot of time and effort. One way to do this is with a so called grid search. Grid search is a method which goes through a list of different parameters and evaluates the results of the model in order to find the best ones. It is possible to go through several different hyperparameters at a time and it can also be combined with cross-validation [11].

3 Method

3.1 Data

As neural networks require a lot of data to train on in order to achieve good results, the amount of data available is crucial. The already existing data did not have enough examples containing the defects that were to be examined. In order to acquire a large enough data set to work with the decision was made to create synthetic data.

Due to the lack of pre-existing defect images, ”passed” images of image test targets that originated from other tests where used, see figure 24, and defects were simulated onto these images. These images consisted of samples from two types of cameras, 4000 from one and 14 000 from the other. The only apparent difference between the cameras being a slight difference in white balance. In total around 18 000 passed images were available. By simulating the defects in these images it was possible to create a much more standardized data set. In order to mimic images taken during production, and not use any reference images as discussed in section 1.1, all images used were unique.

Based on the expertise of our supervisors the defects, appearing in the image samples from the general image quality test, could be divided into 5 different categories. The simulations were then coded by hand and visually evaluated for all the different defects. This is all described in more detail in the following sections.

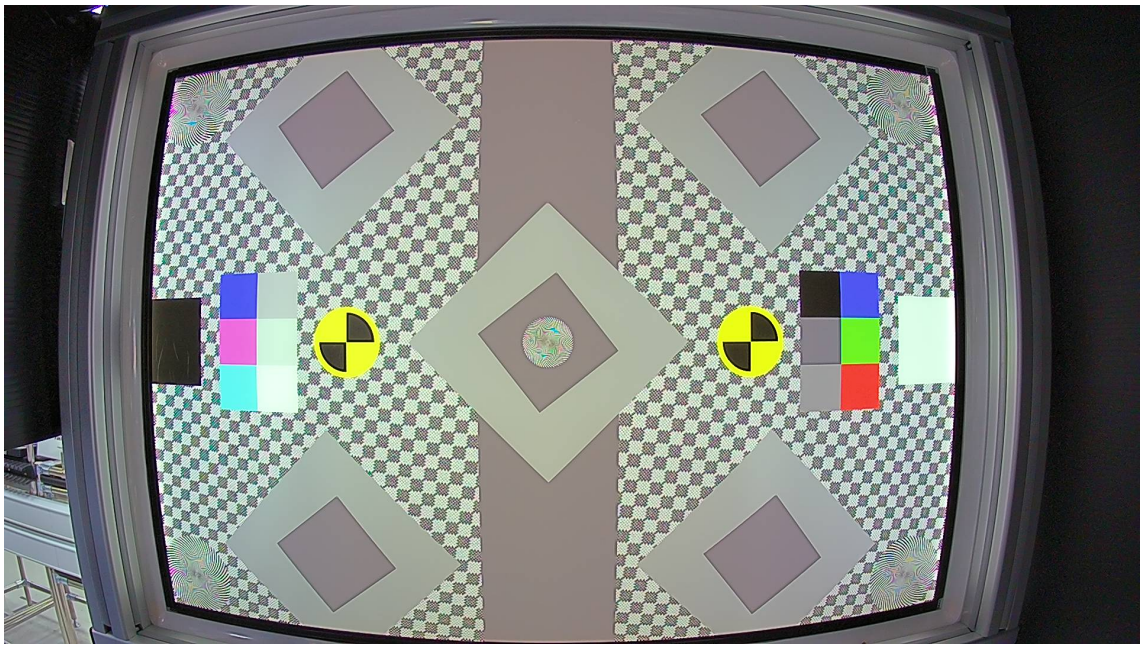


Figure 24: Example of a "passed" image which the data sets were based on.

Blurring

As discussed in section 2.1, the blurring of an image can be represented as a convolution with a PSF. If there is considerable blurring then the PSF can be described as a small disk with a uniform intensity distribution. Such a filter was created and applied to an image. However, when comparing the filtered image to authentic blurred images this approach did not give a similar result. Therefore the simulation of blurring was done with three different types of built-in blurring methods with different PSFs, as these visually gave a more similar result.

The three filters are "blur", "box blur" and "gaussian blur" that are all part of the Pillow module for Python [38]. Blur creates a blurring effect with a kernel that ignores the middle pixels and takes the average of the surrounding pixels. Box blur uses a kernel that takes the average of all the pixels in the kernel. The gaussian filter uses a gaussian kernel to create the blurring. Box blur and gaussian blur can vary the amount of blurring by taking an input that decides the size of the kernel. Gaussian blurring is more like blurring due to atmospheric turbulence according to the theory. When simulating the images, the type of filter was randomly selected, as was the amount of blurring.

Noise

Noise was generated using three different algorithms. The first one generates noise in every pixel for each color channel, the value of the noise is drawn from a uniform distribution of only positive numbers. All values were then scaled using a random constant which range was decided through visual evaluation, and the noise was added to the image. The second algorithm is very much like the first one except that it draws its values from a normal distribution instead. The third one also draws its values from a normal distribution but they are added to the image in a multiplicative way. This results in that the brighter part of the image recieve more noise than the darker parts. Which algorithm used for an image was selected randomly.

Dead Pixels

The generation of dead pixel regions was based on a random walk algorithm with a few regularizing parameters. First a random starting pixel in the image was selected, then one of three different shapes where chosen as a boundary for the algorithm: a circle, a rectangle, or an ellipse. The size of the shape was also set randomly from a predefined range of values depending on the shape selected. The amount of steps for the random walk was made proportional to the size of the bounding shape, the proportions were decided through trial and error.

In order to allow some irregularities similar to those spotted in the original data, the random walk could sometimes escape the bounding shape. The choices made and the parameters chosen for the simulation algorithm of the dead pixel regions were based solely on visual evaluation and through discussion with our supervisors.

Faulty Pixel Row or Column Defects

To begin with, the location, width and orientation of the row or column defects were randomly selected. The color palettes used to create the lines were also randomly chosen from a selection of palettes based on the colors that had been observed in the real images. There were four palettes: an orange, a grey, a pink, and a yellow and green one. The number of palettes used varied between 1 and 4. Each color palette contained several different color shades. When these different variables had been chosen, the pixels within the chosen area were traversed. When a pixel was accessed the probability of keeping the same color as the last pixel was the highest, then there was a lower probability to change color within the palette, and an even lower probability of changing the color palette. This led to a line containing several randomly distributed colors, similar to the real images. Since some of the real images contained homogeneous, thin lines in red, some of the images were simulated with only a thin red line with a randomly selected location and orientation.

Blemishes

The simulation of blemishes was accomplished by creating a mask consisting of a random number of 1-10 small ellipses. The mask was created by having a black image of the same size as the image to perform the simulation on and then drawing white ellipses in that black image. The number of ellipses, the size of the ellipses as well as their position in the image were randomly chosen. The mask was then filtered using a gaussian filter to create soft edges around the ellipses. Finally the mask was used to put a grey image on top of the test target image. The mask defines where to use the test target image and where to use the grey image. This resulted in small grey areas similar to blemishes appearing in the image.

The Complete Data Sets

Four different types of data sets were created. The first data set, the small data set, was made up of 1200 images. These images consisted of 600 passed images from the available data, 300 from each camera, and 600 simulated failed images with defects, also with 300 images from each camera. The different defects described in section 2.1 were equally represented among the failed images. This data set was created to perform binary classification, passed or failed. It was also created to investigate network models.

The next data set, the big data set, was also made for binary classification but contained many more images; a total of 17859 images. 8929 of these were passed and 8930 were failed, with all defects equally distributed. However, since fewer images were accessible from one of the cameras there were in total 4239 from that one and 13620 from the other one.

The third data set, the multi-class data set, consisted of 17859 images: 2976 each of passed, noise, and dead pixel images and 2977 each of lines, blur, and blemish images. This data set contained the same split between the cameras as the big data set. The multi-class data set was constructed for multi-class classification.

The data sets were randomly split into three subsets, one training set, one validation set used for evaluating the hyperparameters, and one test set for the final evaluation of the models. The training set consisted of 60% of the data, the validation set of 20% and the test set of 20%.

The fourth data set, the real data set, consisted of 134 real blurred images, 46 real images with other defects, and 31 real passed images, making a total of 211 images. The real passed images were images that had been passed but were not of the image test target. The real images with other defects than blurring consisted of images with dead pixels, lines, noise and other defects that were previously unseen. The latter consisted of distorted images, a black image and different types of faulty pixels.

3.2 Networks

Starting off, the small data set was as previously mentioned used to try different types of networks. From the many networks that were tested, the ones with the best accuracy were chosen to continue with. These networks were then evaluated on the small data set and the big data set. For the multi-class data set the same networks were used but with the final dense layer adjusted to the number of classes. The networks that were ultimately used are described in the following sections. The networks are described in table format with the different layers, the number of filters used for the layers, the kernel size for the layer, the size of the stride, the activation function, the output shape and the number of trainable parameters.

3.2.1 Single-input Networks

Single-input networks are straightforward. The images in the data sets were used as input and the architecture for the different networks can be seen in the following sections.

The Small Convolutional Network (SCN)

The small convolutional network was constructed from scratch and consisted of four convolutional layers, each followed by a ReLu activation function and a max pooling layer, and finally two fully connected layers, the last one with a softmax activation function. The full architecture is seen in table 1.

Table 1: The architecture of the small convolutional network.

Layer type	**# filters**	**Kernel size**	**Stride**	**Activation function**	**Output shape**	**# parameters**
Input images	-	-	-	-	270 x 480 x 3	-
Rescaling 1/255	-	-	-	-	–||–	-
Convolutional	32	3 x 3	1	ReLu	268 x 487 x 32	896
Max Pooling	-	2 x 2	1	-	134 x 239 x 32	-
Convolutional	68	3 x 3	1	ReLu	132 x 237 x 64	18496
Max Pooling	-	2 x 2	1	-	66 x 118 x 62	-
Convolutional	128	3 x 3	1	ReLu	64 x 116 x 128	73856
Max Pooling	-	2 x 2	1	-	32 x 58 x 128	-
Convolutional	128	3 x 3	1	ReLu	30 x 56 x 128	147584
Max Pooling	-	2 x 2	1	-	15 x 28 x 128	-
Flatten	-	-	-	-	1 x 53760	-
Fully Connected	-	-	-	ReLU	1 x 512	27525632
Fully Connected (Binary/Multi-class)	-	-	-	Softmax	1 x 2 / 6	1026 / 3078

The InceptionV3 Network (IV3)

The network consisted of the InceptionV3 base, see section 2.2.10, two fully connected layers and a dropout layer. See the full architecture in table 2.

Table 2: The architecture of the inceptionV3-based network.

Layer type	# filters	Kernel size	Stride	Activation function	Output shape	# parameters
Input images	-	-	-	-	270 x 480 x 3	-
InceptionV3 base	-	-	-	-	7 x 13 x 2048	-
Flatten	-	-	-	-	1 x 186368	-
Fully Connected	-	-	-	ReLU	1 x 256	47710464
Dropout	-	-	-	-	–||–	-
Fully Connected (Binary/Multi-class)	-	-	-	Softmax	1 x 2 / 6	514 / 1542

The ResNet Network (RN)

The network consisted of the ResNet50 base, see section 2.2.10, two fully connected layers and a dropout layer. See the full architecture in table 3.

Table 3: The architecture of the ResNet50-based network.

Layer type	# filters	Kernel size	Stride	Activation function	Output shape	# parameters
Input images	-	-	-	-	270 x 480 x 3	-
ResNet50 base	-	-	-	-	9 x 15 x 2048	-
Flatten	-	-	-	-	1 x 276480	-
Fully Connected	-	-	-	ReLU	1 x 256	70779136
Dropout	-	-	-	-	–||–	-
Fully Connected (Binary/Multi-class)	-	-	-	Softmax	1 x 2 / 6	514 / 1542

The DenseNet Network (DN)

The DenseNet network consisted of a DenseNet201 base, see section 2.2.10, two fully connected layers and a dropout layer. See the full architecture in table 4.

Table 4: The achitecture of the DenseNet201-based network.

Layer type	# filters	Kernel size	Stride	Activation Function	Output shape	# parameters
Input images	-	-	-	-	270 x 480 x 3	-
DenseNet201 base	-	-	-	-	8 x 15 x 1920	-
Flatten	-	-	-	-	1 x 230400	-
Fully Connected	-	-	-	ReLU	1 x 256	58982656
Dropout	-	-	-	-	–||–	-
Fully Connected (Binary/Multi-class)	-	-	-	Softmax	1 x 2 / 6	514 / 1542

The EfficientNet Network (EN)

The EfficientNet network consisted of an EfficientNetB2 base, see section 2.2.10, two fully connected layers and a dropout layer. See the full architecture in table 5.

Table 5: The architecture of the EfficientNetB2-based network.

Layer type	# filters	Kernel size	Stride	Activation Function	Output shape	# parameters
Input images	-	-	-	-	270 x 480 x 3	-
EfficientNetB2 base	-	-	-	-	9 x 15 x 1408	-
Flatten	-	-	-	-	1 x 190080	-
Fully Connected	-	-	-	ReLU	1 x 256	48660736
Dropout	-	-	-	-	–||–	-
Fully Connected (Binary/Multi-class)	-	-	-	Softmax	1 x 2 / 6	514 / 1542

3.2.2 Dual-input Networks

As mentioned in section 1.1 and 2.2.10, networks can benefit from using a two-stream architecture where one of the inputs has other properties than the original image; one such input is the Laplacian - the second derivative. Using two-stream networks allows the two channels to train their parameters independently and the channels may thus learn different features that when combined can improve the results. The Laplacian of the input image was chosen as the second input due to it having visual properties that were believed to benefit the networks.

The Dual Convolutional Network (DCN)

In the dual convolutional network a two-stream network was used where the two parallel channels were identical and the input was the same for both of the channels. As such it used a single input but with a parallel channel structure. See the architecture of the channels that run in parallel in table 6. The output of the two channels were then concatenated and followed by a few fully connected layers, dropout and batch normalization, see table 7.

Table 6: The architecture of the two channels that run parallel in the self-made dual convolutional network.

Layer type	**# filters**	**Kernel size**	**Stride**	**Activation function**	**Output shape**	**# parameters**
Input images	-	-	-	-	270 x 480 x 3	-
Rescaling 1/255	-	-	-	-	–||–	-
Convolutional	8	3 x 3	1	ReLU	268 x 478 x 8	224
Dropout	-	-	-	-	–||–	-
Batch normalization	-	-	-	-	–||–	32
Convolutional	16	3 x 3	1	ReLU	266 x 476 x 16	1168
Dropout	-	-	-	-	–||–	-
Batch normalization	-	-	-	-	–||–	64
Convolutional	16	5 x 5	1	ReLU	262 x 472 x 16	6416
Dropout	-	-	-	-	–||–	-
Batch normalization	-	-	-	-	–||–	64
Convolutional	32	5 x 5	1	ReLU	258 x 468 x 32	12832
Dropout	-	-	-	-	–||–	-
Batch normalization	-	-	-	-	–||–	128
Convolutional	64	5 x 5	1	ReLU	254 x 464 x 64	51264
Batch normalization	-	-	-	-	–||–	256
Max Pooling	-	2 x 2	1	-	127 x 232 x 64	-

Table 7: The two parallel channels are concatenated and then run through these final layers.

Layer type	**# filters**	**Kernel size**	**Stride**	**Activation function**	**Output shape**	**# parameters**
Concatenate channel 1+2	-	-	-	-	127 x 232 x 128	-
Flatten	-	-	-	-	1 x 3771392	-
Fully Connected	-	-	-	ReLU	1 x 32	120684576
Dropout	-	-	-	-	–||–	-
Batch normalization	-	-	-	-	–||–	128
Fully Connected	-	-	-	ReLU	1x32	1056
Dropout	-	-	-	-	–||–	-
Fully Connected (Binary/Multi-class)	-	-	-	Sigmoid	1 x 2 / 6	66 / 198

The Dual DenseNet Network (DDN)

The dual DenseNet network also made use of a two-stream architecture. The two parallel channels were again identical but made up of a DenseNet base, see section 2.2.10, and can be seen in table 8. The inputs were, however, not the same. The input to one of the parallel channels was the original image and the input to the other was the Laplacian of the image. The output of the two channels were then concatenated and followed by fully connected layers, dropout and batch normalization, see table 9.

Table 8: Architecture of the two channels before concatenating in the dual DenseNet network.

Layer type	# filters	Kernel size	Stride	Activation function	Output shape	# parameters
Input image or Laplacian	-	-	-	-	270 x 480 x 3	-
DenseNet201 base	-	-	-	-	8 x 15 x 1920	-
Batch normalization	-	-	-	-	–||–	7680

Table 9: Architecture of the dual DenseNet network after concatenating channel 1 and 2.

Layer type	# filters	Kernel size	Stride	Activation function	Output shape	# parameters
Concatenate channel 1+2	-	-	-	-	8 x 15 x 3840	-
Flatten	-	-	-	-	1 x 460800	-
Fully Connected	-	-	-	ReLU	1 x 128	58982528
Batch normalization	-	-	-	-	–||–	512
Dropout	-	-	-	-	–||–	-
Fully Connected (Binary/Multi-class)	-	-	-	Sigmoid	1 x 2 / 6	258 / 774

The Dual EfficientNet Network (DEN)

The dual EfficientNet had an identical architecture as the dual DenseNet, but with EfficientNet, see section 2.2.10, as a base instead. This network also used the image and its Laplacian as inputs, see table 10 and table 11.

Table 10: Architecture of the two channels before concatenating in the dual EfficientNet network.

Layer type	# filters	Kernel size	Stride	Activation function	Output shape	# parameters
Input image or Laplacian	-	-	-	-	270 x 480 x 3	-
EfficientNetB2 base	-	-	-	-	9 x 15 x 1408	-
Batch normalization	-	-	-	-	–||–	5632

Table 11: Architecture of the dual EfficientNet network after concatenating channel 1 and 2.

Layer type	# filters	Kernel size	Stride	Activation function	Output shape	# parameters
Concatenate channel 1+2	-	-	-	-	9 x 15 x 2816	-
Flatten	-	-	-	-	1 x 380160	-
Fully Connected	-	-	-	ReLU	1 x 128	48660608
Batch normalization	-	-	-	-	–||–	512
Dropout	-	-	-	-	–||–	-
Fully Connected (Binary/Multi-class)	-	-	-	Sigmoid	1 x 2 / 6	258 / 774

3.3 Training and Testing the Networks

During the training phase a grid search of hyperparameters was performed for all of the models and data sets. The grid search was combined with a 5-fold cross-validation. Each different predetermined value of a hyperparameter was evaluated. The mean of the results was used for finding the best choice. The hyperparameters that were examined were the optimizer and the learning rate as well as the decay steps, the dropout and the momentum where these were relevant. The decay steps decides the number of updates made for the weights for a specific learning rate when a decaying learning rate is used in training. In the case of a decaying learning rate the given learning rate is instead the initial learning rate. This could be used with all optimizers but Nadam. The dropout rate was evaluated only for the models containing a dropout layer, and the momentum could only be applied to the SGD and RMSprop optimizer. The grid search was done one parameter at a time, starting with optimizer, then learning rate, decay steps, momentum and finally dropout. Once the grid search for one parameter was completed, the next grid search was adjusted such that the result of the last was used. The values which were evaluated for the different parameters are presented in table 12.

Table 12: The hyperparameters and their values used in the grid search.

Optimizer	SGD	RMSprop	Adam	Nadam	
Learning rate	0.1	0.01	0.001	0.0001	
Decay steps	100	200	500	1000	
Momentum	0	0.1	0.4	0.8	
Dropout	0	0.1	0.3	0.5	0.8

When the grid search was complete the best values were chosen and the networks were re-trained on the full training set and finally evaluated on the test set.

The different networks were trained on the three data sets that contained simulated images. The fourth data set containing the real images was never used for training but only used for evaluating the performance of the networks trained on the large synthetic data sets. The real data set was primarily evaluated on the networks for binary classification, though the images with blurring and the passed images were also evaluated on the networks for multi-class classification.

3.4 Evaluation

The evaluation of the results was done in different ways in different parts of the thesis. The evaluation of the simulated images was done by human visual assessment by comparing the simulations with the few real images with defects that were accessible. When they were visually similar the simulation was considered complete.

When training the networks and performing the grid searches to select suitable hyperparameters the accuracy and loss of the validation data were evaluated. The validation loss was typically the metric evaluated when doing parameter selections, though the accuracy was still taken into account when the difference in loss between two choices was small but the difference in accuracy was more considerable.

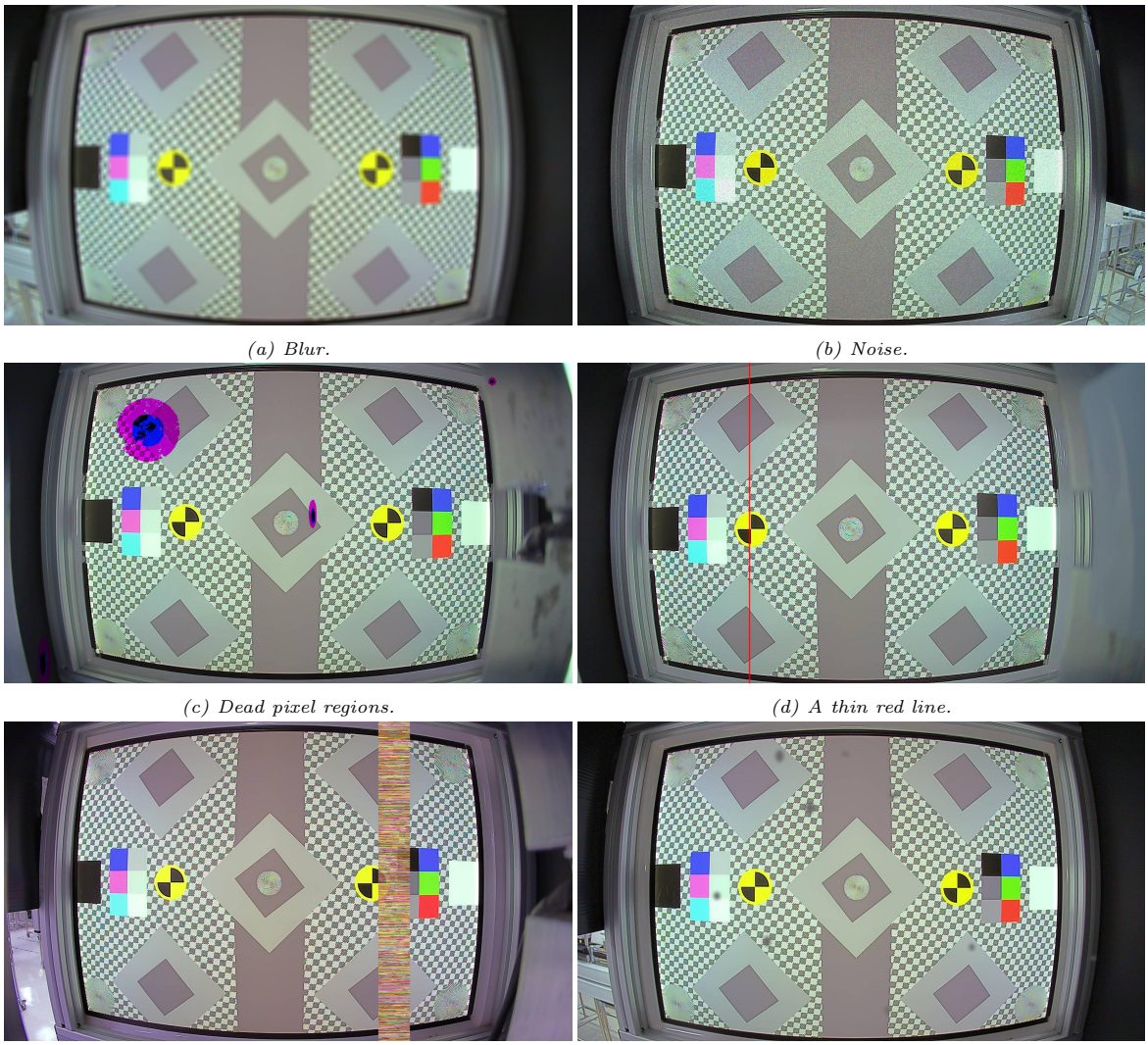
When training a network for binary classification, the model, with the values of the hyperparameters obtained during the grid search, that gave the lowest validation loss was saved. That model was evaluated on the test data based on the accuracy, sensitivity and specificity. Sensitivity in this case corresponds to how well the networks could classify failed images as failed i.e. how it responded to something that was out of the ordinary. Specificity describes how well the network could classify passed images as passed.

When training a network for multi-class classification the final model - the model obtained from the grid search, was evaluated based on the accuracy as well as the confusion matrix, since sensitivity and specificity are not well defined for more than two classes.

4 Results

4.1 Simulated Images

Examples of target images with simulated defects are seen in figure 25. The thin line and the bold line, seen in figures 25d-e belong to the same defect class, but they are not constructed in the same way, see section 3.1.



(a) Blur. *(b) Noise.*

(c) Dead pixel regions. *(d) A thin red line.*

(e) A bold line. *(f) Blemishes.*

Figure 25: Examples of images with simulated defects in.

4.2 The Small Data Set

The results of the grid searches on the small data set for the different networks can be seen in table 34 in Appendix A. The results from training and evaluating these networks using the hyperparameters from the grid searches can be seen in table 13.

Table 13: Results of the different networks when trained on the small data set for binary classification. The best results are highlighted.

Network:	Training acc.:	Validation acc.:	Test acc.:	Specificity:	Sensitivity:
SCN	0.944	0.771	0.838	0.803	0.870
IV3	0.929	0.792	0.842	0.959	0.718
RN	0.996	0.846	0.904	0.911	0.897
DN	1.000	0.913	0.946	0.967	0.923
EN	0.989	0.871	0.904	0.935	0.872
DCN	0.554	0.525	0.488	0.431	0.547
DDN	1.000	0.954	**0.992**	**0.992**	**0.991**
DEN	0.996	0.883	0.933	0.927	0.940

As seen in table 13, DDN had the best performance overall.

4.3 The Big Data Set

The results of the grid searches on the big data set for the different networks can be seen in table 35 in Appendix A. The results from training and evaluating these networks using the hyperparameters from the grid searches can be seen in table 14.

Table 14: Results of the different networks when trained on the big data set for binary classification. The best results are highlighted.

Network:	Training acc.:	Validation acc.:	Test acc.:	Specificity:	Sensitivity:
SCN	0.999	0.993	0.991	0.992	0.990
IV3	0.998	0.958	0.957	0.971	0.943
RN	1.000	0.989	0.985	0.985	0.985
DN	1.000	0.996	0.993	0.993	0.994
EN	0.996	0.980	0.981	0.983	0.979
DCN	0.999	0.998	0.997	0.994	**1.000**
DDN	1.000	0.999	**0.999**	**0.999**	**1.000**
DEN	1.000	0.996	0.995	0.997	0.994

As seen in table 13, DDN again had the best overall performance but all networks still performed very well.

4.4 The Multi-class Data Set

The results of the grid searches on the multi-class data set for the different networks can be seen in table 36 in Appendix A. The resulting accuracies from training and evaluating these networks using the hyperparameters from the grid searches can be seen in table 15. The confusion matrices for the different networks can be seen in tables 16-23.

Table 15: Results of the different networks when trained on the multi-class data set for multi-class classification. The best results are highlighted.

Network:	Training acc.:	Validation acc.:	Test acc.:
SCN	0.995	0.966	0.961
IV3	0.999	0.944	0.942
RN	1.000	0.974	0.974
DN	0.999	0.985	0.984
EN	0.996	0.977	0.981
DCN	0.966	0.931	0.939
DDN	0.999	0.987	**0.989**
DEN	0.998	0.990	**0.989**

Table 16: The confusion matrix for the SCN network on multi-class classification.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blemish	599	1	0	4	15	30
Blur	0	612	1	1	0	0
Dead Pixels	1	0	566	4	0	0
Lines	21	0	1	585	2	2
Noise	23	0	0	0	564	2
Passed	21	1	0	4	6	546

Table 17: The confusion matrix for the IV3 network on multi-class classification.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blemish	521	0	5	28	3	52
Blur	0	614	0	0	0	0
Dead Pixels	7	0	550	11	1	2
Lines	51	0	3	547	0	10
Noise	3	0	0	2	584	0
Passed	21	1	1	6	0	549

Table 18: The confusion matrix for the RN network on multi-class classification.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blemish	584	0	2	8	2	13
Blur	0	614	0	0	0	0
Dead Pixels	6	0	558	6	0	1
Lines	31	0	0	579	0	1
Noise	2	0	0	0	587	0
Passed	15	2	0	4	0	557

Table 19: The confusion matrix for the DN network on multi-class classification.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blemish	594	0	0	1	1	13
Blur	0	614	0	0	0	0
Dead Pixels	7	0	561	2	0	1
Lines	17	0	2	589	0	3
Noise	0	0	0	1	588	0
Passed	7	2	0	2	0	567

Table 20: The confusion matrix for the EN network on multi-class classification.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blemish	588	0	0	6	1	14
Blur	0	614	0	0	0	0
Dead Pixels	5	0	561	4	1	0
Lines	22	0	0	586	0	3
Noise	2	0	0	0	587	0
Passed	7	2	0	2	0	567

Table 21: The confusion matrix for the DCN network on multi-class classification.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blemish	596	5	0	0	0	8
Blur	0	614	0	0	0	0
Dead Pixels	37	0	518	0	0	16
Lines	58	0	0	543	0	10
Noise	44	0	0	0	543	2
Passed	37	2	0	0	0	539

Table 22: The confusion matrix for the DDN network on multi-class classification.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blemish	603	0	0	2	0	4
Blur	0	614	0	0	0	0
Dead Pixels	9	0	561	1	0	0
Lines	9	0	0	601	0	1
Noise	7	0	0	1	581	0
Passed	2	3	0	1	0	572

Table 23: The confusion matrix for the DEN network on multi-class classification.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blemish	601	0	0	3	0	5
Blur	0	613	0	0	0	1
Dead Pixels	3	0	566	2	0	0
Lines	15	0	0	595	0	1
Noise	2	0	0	0	587	0
Passed	2	3	0	3	0	570

In summary, the networks all performed well achieving test accuracies of 0.939 and higher, yet again DDN but also DEN had the best performance - both reaching a test accuracy of 0.989. A closer look at the confusion matrices in tables 16-23 show that all networks displayed similar difficulties , to varying degrees, as they predicted several defects as blemishes.

4.5 The Real Data Set

The results when testing the networks for binary classification on the real data set can be seen in table 24. The resulting accuracies for testing on the real passed images and the real blurred images can be seen in table 25 and the respective confusion matrices in tables 26-33.

Table 24: Results of the different networks for binary classification when tested on the real data set. The best results are highlighted.

Network:	Accuracy:	Specificity:	Sensitivity:
SCN	**0.540**	0.220	0.590
IV3	0.493	0	0.578
RN	0.507	0	**0.594**
DN	0.502	0	0.580
EN	0.474	0	0.555
DCN	0.209	0.740	0.116
DDN	0.370	**1.000**	0.267
DEN	0.483	0.419	0.494

Table 25: Results of the different networks for multi-class classification when tested only on the real passed images and real blurred images. The best result is highlighted.

Network:	Accuracy:
SCN	0.406
IV3	0.339
RN	0.309
DN	0.364
EN	0.364
DCN	**0.412**
DDN	0.358
DEN	0.394

Table 26: The confusion matrix for the SCN network on multi-class classification on the real passed images and real real blurred images.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blur	25	67	0	7	0	35
Passed	0	2	2	27	0	0

Table 27: The confusion matrix for the IV3 network on multi-class classification on the real passed images and real real blurred images.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blur	18	56	3	17	0	40
Passed	0	0	0	31	0	0

Table 28: The confusion matrix for the RN network on multi-class classification on the real passed images and real real blurred images.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blur	26	51	12	5	0	40
Passed	0	0	11	20	0	0

Table 29: The confusion matrix for the DN network on multi-class classification on the real passed images and real real blurred images.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blur	25	60	3	3	0	43
Passed	0	0	0	31	0	0

Table 30: The confusion matrix for the EN network on multi-class classification on the real passed images and real real blurred images.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blur	22	60	4	3	0	45
Passed	3	0	10	18	0	0

Table 31: The confusion matrix for the DCN network on multi-class classification on the real passed images and real real blurred images.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blur	41	67	1	1	1	23
Passed	0	1	4	20	5	1

Table 32: The confusion matrix for the DDN network on multi-class classification on the real passed images and real real blurred images.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blur	7	55	0	6	0	66
Passed	1	27	1	2	0	0

Table 33: The confusion matrix for the DEN network on multi-class classification on the real passed images and real real blurred images.

True \ Predicted	Blemish	Blur	Dead Pixels	Lines	Noise	Passed
Blur	2	65	1	1	0	65
Passed	1	12	0	18	0	0

As seen by examining tables 24-33 the performance of the networks on the real data was poor. There was no obvious choice when deciding upon the best performing network.

5 Discussion

5.1 Data Generation

Using synthetic data instead of or alongside real/authentic data is an easy and frequently used method to increase the number of samples in your data. There is however one drawback that is always present when using synthetic data, and that is how well the data represents reality. This is especially true if the data you are trying to simulate is seemingly random, while the code for generating the synthetic data is only pseudorandom. This was the case in this thesis. The pseudorandomness could create underlying structures which are not present in the real images.

The visual evaluation of the synthetic images initially gave good indications that the simulated images corresponded well to the real images. Since the benefits of using synthetic data, such as being able to acquire large amounts of data and easily perform correct annotations, far outweighed the shortcomings of the simulated images not being a entirely equal to the real images. A possible improvement could be to investigate if there is a way to mathematically evaluate the simulated images, if the visual assessment is not considered enough. This could however be difficult since the defects are different from one image to another.

As the amount of available real images deemed as failed with clear quality defects was rather small, it was clearly a possibility that the assumptions made on the character and distribution of the defects might not hold in all real cases. One example of this is the noise defect; in the simulated images the noise was added to the whole image whereas in the authentic case this might not always hold true as only parts of the image can become noisy.

For the sake of easily acquiring lots of image samples with standardized backgrounds/targets the data was gathered from the passed images of another production test. The reason for using standardized targets in the images was first and foremost for future prospects of implementing neural networks in production, where standardized tests are preferred. A second reason was that having the same targets/background in all image samples was believed to improve the results of the neural networks. This was done with the drawback that the networks will not generalize as well to images with other backgrounds.

Yet another possible issue, one that was not initially considered, is the fact that the images were all stored in the jpg format and when simulating and adding the defects the images were once again saved as jpgs. This could be problematic as editing and saving jpgs multiple times can reduce the overall image quality. However since the images only had simulations performed on them once before being saved and stored, the jpg compression should not noticeably affect the quality of the images.

When it comes to creating data sets, one should always aim to have a large amount of samples as training neural networks are heavily reliant on the amount of data available. This was not adhered when creating the small data set as it only contained 1200 images. This was first and foremost due to hardware constraints on the computers available. It was also a quick and easy way to get started with examining different networks and network architectures, and it was additionally believed that networks performing well when trained on this small data set would perform even better when trained on a larger one. Thus the small data set was seen as a ”trial” data set.

When the hardware issues had been resolved the big data set that used all available images of good enough quality was generated. It was decided that no image would be used for more than one simulated defect or as a passed image with no defect. This means that all images were unique in order to prevent possible overfitting. The decision to only use unique images limited the amount of samples in the data set but this was still preferred to having duplicate images. The third data set, the multi-class data set, was created with the same reasoning in mind, but for the sake of having all classes equally represented there were less passed images than in the big data set. However, there were still only unique images in the set.

5.2 Network Selection

The final networks were chosen based on the theory presented in section 2.2 and by testing on the small data set. Eight different networks were evaluated throughout the rest of the thesis. Two of the networks, the SCN and DCN network, were fairly simple and made from scratch. All others were based on transfer learning, i.e. they had a pretrained network as a base.

The advantage of using transfer learning is that the network has already seen many images and therefore one does not need as large of a data set. The last added layers simply tunes the network to the specific data set. A negative aspect of the pretrained networks that were used is that they were all trained on ImageNet which is made for object recognition. They are therefore made to detect physical objects in an image and since this thesis was not only looking for objects but rather the state of the image itself it could mean that the networks are not learning optimal features. The networks used are also large and memory-demanding.

In the results on the small data set, see table 13, a clear overfitting of most networks can be seen as the training accuracy was high but the test accuracy was much lower. For example the SCN network had a training accuracy of 0.944 and a test accuracy of 0.838. The best overall performance on the small data set was given by the DDN network with a test accuracy of 0.992, a specificity of 0.992 and a sensitivity of 0.992. The worst performance was given by the DCN network with an accuracy of only 0.488, a specificity of 0.431 and a sensitivity of 0.547. Here the advantages of transfer learning is clear since the networks with a pretrained base performed better than those without. A conclusion is therefore that for a smaller data set it is important to have a deep or complex network to be able to learn properly.

Contrary to the small data set, the results on the big data set, see table 14, show less signs of overfitting as the test accuracy and training accuracy were much more similar. All of the networks performed well with accuracies above 0.9. The best performance was again given by the DDN network with a test accuracy of 0.999, a specificity of 0.999 and a sensitivity of 1. The worst performing network was the IV3 network with a test accuracy of 0.957, a specificity of 0.971 and a sensitivity of 0.943. The SCN network had a test accuracy of 0.991, a specificity of 0.992 and a sensitivity of 0.990. There is not a big difference between the networks using transfer learning and those that do not, proving that if the data set is large enough then there is no need for a deep or complex network. Even though the best result was given by the DDN network it was only marginally better than the SCN network and therefore the SCN network might be preferable since it is less complex and requires less memory.

As for the multi-class data set, see table 15, the results were also very promising, though the accuracies somewhat lower than for the big data set. The best result was given by the DDN network as well as the DEN network, both with test accuracies of 0.989. The worst performance was again by the DCN network with a test accuracy of 0.939. In the confusion matrices for all of the networks, see tables 16-23, most of the confusion seemed to be regarding the blemishes; blemishes were classified as other defects as well as other defects were classified as blemishes.

In general there does seems to be a high potential for the use of neural networks for classifying image quality defects. The best performing network was the DDN network for all of the synthetic data sets. It is also clear that when training on a small data set a complex network is required for good results, but when training on a large data set it is enough with a simple network. Most of the defects' characteristics were learned properly, but there was some confusion with the blemishes. This was not surprising since the blemishes were small and blended in well with the detailed background. It would be easier to detect blemishes if the background was homogeneous. It is therefore advisable that such a test is used for the blemishes instead of combining the blemishes with the other defects in the general image quality test.

The performance of the binary classification was marginally better than for the multi-class classification. The binary classification is good for a more general use when other defects than the ones that the networks have been trained on can occur. With the multi-class classification the networks will only be able to detect the specific defects that the networks have been trained on. Other new defects have to fit in with one of these which might raise some confusion. It is possible that for the binary classification the networks can learn that all traits that are not similar to the passed target are classified as failed. For the multi-class networks a new defect still has to find a category to fit in with, which could be problematic.

5.3 Augmentation

As augmentation is a commonly used technique for network improvement when the amount of data is limited, it was used in the early stages of training different networks on the small data set. The augmentations used were randomly flipping, rotating or changing the white balance of the images. This did however not improve the accuracy on the validation data, but rather seemed to worsen the accuracy. Therefore augmentation was not further used.

The loss in accuracy could be explained by the fact that the images were of a very specific and detailed target, flipping and rotating of these targets made it harder for the network to recognize which traits should be present and which should not. Moreover, augmentation of the images was not relevant when training on the big data set since there was enough data to give satisfying results. Last but not least, one possible defect that can appear but was not included in this thesis is that an image can be turned upside down due to firmware issues which is yet another reason to not use augmentation in this particular case.

5.4 Evaluation on The Real Data

The main goal of evaluating the networks on the real data was to see if they could correctly classify defects that had not been simulated but also to see how they would react to new previously unseen images or defects. Since the passed images of image test targets were used for training there was no use in testing these in this phase, but rather images of other scenes that had been passed in production were of interest. There was a large database of blurred images, in fact most of the images used during training originated from a focus test, and therefore many more blurred images could be tested, compared to the other defects. For the other defects such as dead pixels, lines etc only a handful of images could be found. Due to this the real data set was skewed with varying amounts of images with different defects, which is why the accuracy might not be a good measure on the performance of the networks. Additionally, there were not that many images in total to test on which means that these result are not completely reliable.

Still, looking at the resulting accuracies in tables 24 and 25, these are not very promising. For the binary classification the best result was given by the SCN network with an accuracy of 0.54, while the worst was the DCN network with an accuracy of 0.209. For the multi-class networks the best was the DCN network with an accuracy of 0.412 and the worst was the RN network with an accuracy of 0.309.

A better approach would be to look at the sensitivity and specificity for binary classification and the confusion matrix for multi-class classification, see table 24 and tables 26-33. Which network is better depends on whether the specificity or sensitivity is of higher importance. In this case the sensitivity was more important, since it is worse to send out faulty cameras to customers than remaking good cameras. In the long run however, it is important that both sensitivity and specificity are high in order to reduce unnecessary costs of remaking cameras. For binary classification the highest sensitivity was 0.594, given by the RN network, but this network had a specificity of 0. The highest specificity was 1, given by the DDN-network. This network had a sensitivity of 0.267. The middle ground between sensitivity and specificity was given by the DEN network with a specificity of 0.419 and a sensitivity of 0.494. For multi-class classification only the DCN network was able to classify a passed image as passed, but only one out of 31. The best performance on the blurred images was by the DCN and SCN networks with 67 out of 134 correctly classified. In general the networks were able to correctly classify between 55-67 of the blurred images.

In conclusion, the networks trained on synthetic data did not perform well on the real data. A possible reason for this is that, as mentioned before, the synthetic data was not similar enough to the real data and the networks had not seen all of the defects present in the real data. It did seem like the networks for binary classification had learned that the passed images should have a specific structure and anything that was not similar to this was simply classified as failed. This applies to all networks but the DDN network for binary classification were it was able to classify the passed images correctly.

In general the networks were able to identify dead pixels and lines well and could classify all these images as failed. However, they did show a tendency to find similar structures such as straight edges and color spots in the passed images and therefore classify these as failed. This indicates that due to the networks being trained on the specific image test target, all images being tested should also be of the same target as the networks are specialized on that specific background. The blurred images were mostly classified as blemishes or passed images. The confusion with the blemishes was also seen in the synthetic data and therefore did not come as a surprise. A reason for a blurred image to be classified as passed could be that there was a very mild blurring in the image or that only some parts were blurry. The conclusion to be drawn is that a large data set of real images should be used to train the networks in order for them to be used in practice.

5.5 Ethical Considerations

There are no ethical considerations in this thesis as the images did not contain any ethically compromising information nor did they compromise the integrity of any concerned parties. Moreover there are no ethical dilemmas in the purpose of the thesis.

6 Conclusion

There is a clear potential for the use of machine learning and specifically a neural network based algorithm to classify image quality in camera production. The images should be restricted to the image test target, as this is what the network has been trained on. Networks for both binary and multi-class classification could be used. Binary classification networks ought to be better at handling new unseen defects - defects that would make an image fail the test, with the downside of not classifying the nature of the defect while multi-class classification networks can produce misleading results when exposed to a new type/class of defect. There is a need for a large data set of real images since the networks trained on synthetic data did not perform well on real data. If the data set is large enough a small convolutional network is sufficient, but with a limited amount of data and with enough hardware resources available, networks using transfer learning perform well. The networks evaluated were able to classify most defects, although having problems with blemishes specifically. The recommendation would therefore be to have other tests more suited to find blemishes while focusing more on other defects such as dead pixels or noise in a final general image quality test.

7 Future Work

To build on the work done in this thesis, the first thing to do would be to gather more real data and train the networks on those images. It would also be of interest to investigate how one would handle images with multiple defects in them. Expanding with more defects than were examined in this thesis project could also be an additional possibility. Moreover, it would be interesting to see if semantic segmentation could be used to find where the defects are located in the image, and display them to an onlooking operator.

Since the general image quality test used in production not only looks at images but also at video streams it would be of interest to investigate if the classification and/or segmentation could be done on videos and how well it could perform there. Other networks and architectures, more suited for video streams, might then be needed.

In the future it is also a matter of designing one or several test tools/programs that can actually be used in production.

During testing of the networks in this thesis there was a behaviour that we were unable to explain: there was some discrepancy between the results on one network when tested on different data sets. This was first encountered when testing a network, trained on the big data set, on the small data set. The reason for this weird behaviour likely lies in the simulation of the defects in the images but could be further investigated.

References

[1] T. Ahmed and N. H. N. Sabab. Classification and understanding of cloud structures via satellite images with efficientunet, licensed under creative commons attribution-noncommercial-noderivatives 4.0 international. *Earth and Space Science Open Archive*, page 7, 2021.

[2] H. A. Alshamarti. Removal of gaussian noise on the image edges using the prewitt operator and threshold function technical, licensed under creative commons cc by- nc 4.0. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 15:81–85, 11 2013.

[3] L. Alzubaidi, J. Zhang, and A. Humaidi. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions, licensed under a creative commons attribution 4.0 international. In *J Big Data*, volume 8, 2021.

[4] S. Bianco, L. Celona, P. Napoletano, and R. Schettini. On the use of deep learning for blind image quality assessment. *CoRR*, abs/1602.05531, 2016.

[5] D. M. Chandler and S. S. Hemami. Vsnr: A wavelet-based visual signal-to-noise ratio for natural images. *IEEE Transactions on Image Processing*, 16(9):2284–2298, 2007.

[6] W.-J. Chang, L.-B. Chen, C.-H. Hsu, C.-P. Lin, and T.-C. Yang. A deep learning-based intelligent medicine recognition system for chronic patients. *IEEE Access*, 7:44441–44458, 2019.

[7] C.-Y. Cho, T.-M. Chen, W.-S. Wang, and C.-N. Liu. Real-time photo sensor dead pixel detection for embedded devices. In *2011 International Conference on Digital Image Computing: Techniques and Applications*, pages 164–169, 2011.

[8] I. da Silva. et al. artificial neural networks. a practical course. Springer International Publishing, 2017.

[9] M. Delbracio. Two problems of digital image formation: Recovering the camera point spread function and boosting stochastic renderes by auto-similarity filtering. *Signal and Image processing*, 2013.

[10] S. Ele and W. Adesola. Artificial neuron network implementation of boolean logic gates by perceptron and threshold element as neuron output function, licensed under creative commons attribution cc-by. *International Journal of Science and Research (IJSR)*, 4:4–438, 10 2013.

[11] I. Fadil, M. A. Helmiawan, and Y. Sofiyan. Optimization parameters support vector regression using grid search method. *2021 9th International Conference on Cyber and IT Service Management (CITSM), Cyber and IT Service Management (CITSM), 2021 9th International Conference on*, pages 1 – 5, 2021.

[12] J. Gala. Difference between a neural network and a deep learning system,. available under a Creative Commons Attribution Licence CC BY-SA at https://www.geeksforgeeks.org/difference-between-a-neural-network-and-a-deep-learning-system/, accessed 2022-02-15, updated 07 Feb, 2022.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[16] Imatest. Imatest LLC, using blemish detect. https://www.imatest.com/docs/blemish/, accessed 2022-02-07, updated 2021.

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[18] A. Jung and s. SpringerLink (Online. *Machine Learning. [Elektronisk resurs] : The Basics*. Machine Learning: Foundations, Methodologies, and Applications. Springer Singapore, 2022.

[19] L. Kang, P. Ye, Y. Li, and D. Doermann. Convolutional neural networks for no-reference image quality assessment. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1733–1740, 2014.

[20] J. Kim, A.-D. Nguyen, S. Ahn, C. Luo, and S. Lee. Multiple level feature-based universal blind image quality assessment model. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 291–295, 2018.

[21] N. Kimura, I. Yoshinaga, K. Sekijima, I. Azechi, and D. Baba. Convolutional neural network coupled with a transfer-learning approach for time-series flood predictions, licensed under a creative commons attribution(cc by). *Water*, 12:96, 12 2019.

[22] R. L. Lagendijk and J. Biemond. Chapter 14 - basic methods for image restoration and identification. In A. Bovik, editor, *The Essential Guide to Image Processing*, pages 323–348. Academic Press, Boston, 2009.

[23] L. Lepistö, J. Nikkanen, and M. Suksi. Blemish detection in camera production testing using fast difference filtering. *Journal of Electronic Imaging*, 18(2):1 – 3, 2009.

[24] J. Li, J.-h. Cheng, J.-y. Shi, and F. Huang. Brief introduction of back propagation (bp) neural network algorithm and its improvement. In D. Jin and S. Lin, editors, *Advances in Computer Science and Information Engineering*, pages 553–558. Springer Berlin Heidelberg, 2012.

[25] K. Mahendru. Analytics Vidhya, a detailed guide to 7 loss functions for machine learning algorithms with python code. https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/, accessed 2022-02-16, updated 2019.

[26] M. A. Mercioni and S. Holban. The most used activation functions: Classic versus current. *2020 International Conference on Development and Application Systems (DAS)*, pages 141–145, 2020.

[27] A. Mohanty. The subtle art of fixing and modifying learning rate. https://towardsdatascience.com/the-subtle-art-of-fixing-and-modifying-learning-rate-f1e22b537303, accessed 2022-02-15, updated 2019.

[28] A. Nath. Image denoising algorithms: A comparative study of different filtration approaches used in image restoration. In *2013 International Conference on Communication Systems and Network Technologies*, pages 157–163, 2013.

[29] M. Niknejad and M. A. T. Figueiredo. Poisson image denoising using best linear prediction: A post-processing framework. *CoRR*, abs/1803.00389, 2018.

[30] S. M. Patil. Advanced method of detection and removal of noise from digital image. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 2, 2013.

[31] J. Pech-Pacheco, G. Cristobal, J. Chamorro-Martinez, and J. Fernandez-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 314–317 vol.3, 2000.

[32] H. Ren, D. Chen, and Y. Wang. RAN4IQA: restorative adversarial nets for no-reference image quality assessment. *CoRR*, abs/1712.05444, 2017.

[33] scikit-learn developers. Scikit-learn developers, 3.1. cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html, accessed 2022-02-15, updated 2021.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[35] S. Sun, Z. Cao, H. Zhu, and J. Zhao. A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics*, 50(8):3668–3681, 2020.

[36] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[37] M. Thoma. Receiver operating characteristic (roc) curve with false positive rate and true positive rate. https://upload.wikimedia.org/wikipedia/commons/3/36/Roc-draft-xkcd-style.svg, accessed 2022-03-05, updated 2018.

[38] Tutorialspoint. Python pillow - blur an image. https://www.tutorialspoint.com/python_pillow/python_pillow_blur_an_image.htm, accessed 2022-04-12, updated 2022.

[39] T.-S. Wang, X.-B. Gao, W. Lu, and G.-D. Li. A new method for reduced-reference image quality assessment. 35:101–104+109, 02 2008.

[40] Z. Wang, E. Simoncelli, and A. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003.

[41] Q. Yan, D. Gong, and Y. Zhang. Two-stream convolutional networks for blind image quality assessment. *IEEE Transactions on Image Processing*, 28(5):2200–2211, 2019.

[42] X. Yang, F. Li, and H. Liu. A survey of dnn methods for blind image quality assessment. *IEEE Access*, 7:123788–123806, 2019.

Appendix A Grid Search Results

Below in tables 34-36 the results from the grid searches for the different networks and data sets are presented.

Table 34: Results of grid search for the different networks when trained on the small data set for binary classification.

Network:	Optimizer:	Learning rate:	Decay steps:	Momentum:	Dropout:
SCN	RMSprop	0.001	200	0	-
IV3	RMSprop	0.001	200	0.8	0.3
RN	Adam	0.0001	100	-	0.3
DN	Adam	0.0001	200	-	0.3
EN	Adam	0.0001	200	-	0.3
DCN	RMSprop	0.01	500	0.4	0.5
DDN	RMSprop	0.0001	200	0	0
DEN	Adam	0.01	200	-	0

Table 35: Results of grid search for the different networks when trained on the big data set for binary classification.

Network:	Optimizer:	Learning rate:	Decay steps:	Momentum:	Dropout:
SCN	RMSprop	0.01	1000	0.1	-
IV3	RMSprop	0.0001	1000	0.1	0
RN	Nadam	0.0001	Not decaying	-	0.1
DN	Adam	0.0001	1000	-	0.1
EN	Adam	0.001	1000	-	0.1
DCN	RMSprop	0.001	Not decaying	0	0.6
DDN	RMSprop	0.0001	1000	0	0.1
DEN	SGD	0.1	1000	0.4	0.1

Table 36: Results of grid search for the different networks when trained on the multi-class data set for multi-class classification.

Network:	Optimizer:	Learning rate:	Decay steps:	Momentum:	Dropout:
SCN	Adam	0.001	1000	-	-
IV3	Adam	0.0001	1000	-	0.1
RN	Adam	0.0001	1000	-	0.1
DN	Adam	0.0001	1000	-	0.3
EN	Adam	0.0001	1000	-	0.1
DCN	RMSprop	0.001	1000	0	0.1
DDN	SGD	0.1	500	0.1	0.5
DEN	SGD	0.1	1000	0.1	0.5

Master's Theses in Mathematical Sciences 2022:E28
ISSN 1404-6342

LUTFMA-3477-2022

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/