Abstract

Loss of focus in surveillance cameras can occur due to factors such as environmental aspects, sabotage or system errors. The focus loss leads to degraded footage that is of no use for either post-investigation or automatic video analysis. It is therefore of the utmost importance to provide an automatic detection system in order to resolve the issue as fast as possible.

The most popular approaches for out-of-focus detection in image analysis can be categorized into traditional, deep learning-based and a combination of the two. The traditional methods are usually based on a comparison between consecutive frames, where the edge content is in some way quantified and compared. Recent related works focus heavily on the learning and combination-based approaches, promising a higher blur detection accuracy. We apply this to the field of surveillance footage and evaluate the performance of a popular traditional method based on Discrete Fourier Transform (DFT) and three different Convolutional Neural Networks (CNNs); Pix2Pix, MANN and MFF.

There are several important requirements for surveillance application considered in this project: flexibility in regards to different camera resolutions and robustness to different surroundings, as well as weather- and lighting conditions. Considering the computational weight and memory requirements is also of importance for on-edge camera application. To train the networks on an objective amount of blur, a method for artificial blurring is utilized. A large dataset is acquired using Axis cameras, and a Gaussian Kernel is applied for obtaining various blur levels. The models are finally tested on optical blur, on images created by manually changing the focus level on the used cameras.

Of the three implemented CNN networks, the Pix2Pix model shows the most promising results, with the disadvantage of being the largest. However, the size is dramatically decreased with the use of unstructured pruning and quantization. Unexpectedly high accuracy is also achieved by the computationally heavy DFT approach, with the exception of bad generalization properties. In the future, a combination method could be of interest, as well as expanding the model for detection of additional anomalies or weather conditions such as fog, rain, tampering etc.

Acknowledgements

We would like the offer a special thanks to our Axis supervisors Calle Janlén Stenberg, Marcus Dittmer Wennermark and Syed Asad Naveed. We feel very grateful for your continuous support, invaluable feedback and generously provided knowledge and expertise. Thanks should also go to LTH supervisor Anders Heyden for always offering valuable input and inspiration.

Notations and Abbreviations

ACAP – AXIS Camera Application Platform
ADAM – Optimizer with name derived from adaptive moment estimation
BRISQUE – Blind/Referenceless Image Spatial Quality Evaluator
CNN – Convolutional Neural Network
CPU – Central Processing Unit
DBD – Defocus Blur Detection
DCT – Discrete Cosine Transform
DFT – Discrete Fourier Transform
FAR – False Alarm Rate
FNN – Feed-forward NeuralNetworks
FPN – Feature Pyramid Network
GAN – Generative Adversarial Network
GPU – Graphics Processing Unit
LSTM – Long Short-Term Memory Networks
PSF – Point Spread Function
PTZ – Pan Tilt Zoom
ReLU – Rectified Linear Unit, type of activation function
MP – Mega Pixel
SVM – Support Vector Machine
TPU – Tensor Processing Unit
VSAM – Video Surveillance and Monitoring

Model names

MANN
MFF
Pix2Pix

Deep Learning Based Out-of-focus Detection on Surveillance Footage

Hanna Bengtsson and Teodora Kerac

13th June 2023

Contents

1 Introduction

Image degradation can occur due to factors such as sabotage, environmental aspects, or software and hardware failure. Degraded footage is of no use for neither post-investigation or automatic video analysis. It is therefore of the utmost importance to provide an automatic detection system in order to resolve the issue as fast as possible. Loss of focus is one such issue that makes the footage blurry and therefore useless. There are a number of circumstances that can lead to blurry images, such as motion, fog or dirt. In this report, we address specifically the matter of detecting out-of-focus blur.

1.1 Task

In surveillance cameras, the focus is set at a particular point, and then due to a change in temperature, weather, or scenery, the focal point might shift. The change of focus can also occur due to an error in the camera, e.g. software or hardware error. This requires a focus scan re-run in order to reestablish the focus point. By being able to automatically detect when the camera loses focus, this could be done more efficiently.

The goal of the project is to investigate methods to detect out-of-focus images. The model should be adaptable to both varifocal and fixed-focal cameras, as well as cameras with different resolutions. Varifocal cameras contain a lens with an adjustable focal length and zoom. This means that the focus changes as the focal length and magnification change. Fixed-focal cameras as the name implies have a fixed focal length and do not allow for adjustable zoom. The method should also be compatible with different scenes, weather and lighting conditions. A reasonable computational complexity should also be taken into consideration.

Finally, the goal is to validate the concept against a number of cameras in different surroundings.

1.2 Related Work

Detection of blurriness in an image can be compared to the task of evaluating image sharpness, for which methods of the past decade have been described, compared and evaluated in Zhu et al. [1]. The article gives an overview of the methods available, classifies them into categories and makes a comparison of their results. Their advantages and disadvantages are discussed and a final distribution of the different categories is mapped out for different parts of the past decade. The three categories are: Traditional, Learning-based and Combination-based, consisting of a combination of methods belonging to the other two categories.

It is clear that deep learning-based methods constitute a larger part of methods developed by scholars in recent years, as they climb from making up 19% of methods published between the years 2013-2015, to 38% between 2019-2021. It should also be noted that the number of combination methods between the two mentioned periods rose from 9% to 24%, considering that these often include a combination of a traditional image processing method and deep learning. This shift is easily explained by the results of a large scale testing performed by Zhu et al. [1], concluding the superiority of the learning and combination categories in comparison to the traditional methods.

However, for the specific task of differentiating between a blurry and a sharp image, there are a number of methods released in recent years, based on both deep learning and traditional image processing methods, or a combination of the two.

Traditional methods

Traditional methods entail spatial domain methods such as edge detection, and spectral domain methods like Fourier or Wavelet transformations [1].

The main disadvantage of the background subtraction approach is the computational burden, according to Hosseini and Taherinia [2]. Additionally, the commonly used GMM has major drawbacks such as requirements of large memory space and difficulty with non-static backgrounds, different weather conditions and camera noise [3].

Tsesmelis et al. [4] applies a method of detecting a certain number of key points in an image. It relies on the fact that the number of identified SURF points will decrease when the image is blurred, as lines and corners then tend to attenuate. As consecutive frames are analyzed, a large decrease in the number of key points indicates that the camera is out-of-focus. This method also relies on a background image database that needs to be kept and continuously updated.

Pagaduan et al. [5] is comparing traditional methods such as Tenengrad, Laplace, HaarWavelet and DFT for the detection of out-of-focus images. The dataset on which each of the methods is tested contains a mix of sharp images and images with motion blur, out-of-focus blur, and synthetic blur. HaarWavlet showed the best accuracy, while DFT presented the best precision score [5].

Learning based

Learning-based methods learn the features from the training dataset input and use the knowledge to predict unseen data. Such methods can be machine or deep learning based [1].

Khajuria et al. [6] applies a method that classifies images as sharp or blurry, solely based on the decision of a Convolutional Neural Network (CNN). The network is empirically selected and consists of only 4 layers. It is compared to the BRISQUE method using SVM, which it outperforms. The image data on which it is tested has a variety of sharp, natural blur and synthetic blur images. A downside to such a simple model has been pointed out by S et al. [7], according to which the model cannot be used for large batch sizes as it leads to the degradation of the model.

Pan et al. [8] has developed a state-of-the-art method that is meant to detect a number of different camera anomalies, such as blurring, occlusion of the camera, spray paint sabotage, etc. The method deploys a self-supervised learning (SSL) method, in order to make use of the abundance of "regular" surveillance footage (no anomalies) and limit the need for a large amount of labeled data on each detectable anomaly in the model. The method is a large model consisting of two assembled ResNet-based networks, with an added tweak to the basic ResNet blocks that build the network. This is done in order to achieve lower model complexity, and implemented by introducing a compression factor and a scaling factor to the number of channels used for feature extraction in the convolutional layers. Other additions are also made to the model but with the purpose of improving the detection of other anomalies (not blur). Finally, testing of the model in comparison to other methods like e.g. ResNet34, suggests only a minor improvement, but with a significantly smaller model size. This is, therefore, the only benefit of implementing the SSL method [8].

Senapati et al. [9] developed a model that first detects objects in the image, and then determines whether the object is blurry or not. The proposed CNN network that is responsible for the blur detection consists of only 2 convolutional layers and gives a poor detection accuracy of 73%. It is, however, trained on a very small dataset of around 400 images, which could suggest that the size of the model isn't way too small for their task.

Combination

Szandała [10] makes a comparison between the traditional Laplace method of detecting an amount of edges (with variance) and a simple CNN to detect blur. Based on the results of testing on the same dataset, the author draws the conclusion that they have very similar performance. However, the author sees a vast improvement potential for the CNN, as well as large benefits of combining it with other methods. The same cannot be said about the rigid and strict Laplace-based method that is entirely dependant on the set variance thresholds for each class. This itself makes the model hard to generalize and use on different cameras. It should also be added that the images used for their study are non-surveillance photographs, imposing higher demands on the classification network. This is due to the possibility of an extremely blurry background in contrast to a very sharp object that is in focus in the image. This scenario is harder to classify and not relevant for classifying specifically surveillance footage with a large depth of field (often infinity focus).

Almustofa et al. [11] explores using focus measure together with either thresholding or SVM-classifiers, and compares the two methods with CNNs on blurred images. The final results show that the CNN outperforms the other methods, possibly due to edge sharpness analysis being ineffective on complex images. However, the focus measure thresholding is easy to prepare and not as dependant on the dataset distribution. The paper proposes that a combination of the methods, or focus measure methods as a preprocessing technique, could be a way forward.

Related applications

Defocus blur detection (DBD) is a field that aims to differentiate between in-focus and

out-of-focus pixels in a single image, with the goal of segmenting the blurry regions of the image. Recently, a multi-scale approach has been common in DBD methods [12] [13] [14]. Wang et al. [15] proposed a deep neural network to find both motion and defocus blur. Several M-shaped, multi-input, multi-loss, encoder-decoder networks are combined in a pyramid ensemble model in an effort to resolve scale ambiguity problems. Park et al. [16] uses different handcrafted features to create a defocus map with the help of a multi-scale patch extraction strategy. This thesis focuses on global out-of-focus blur. DBD applications are therefore not the primary source of inspiration, while still being an interesting area.

1.3 Limitations

As described in the previous section, there is a large number of possible methods for solving the classification problem at hand. According to related works, deep learning methods and combination methods that include a deep learning component, have shown superior results in comparison to traditional methods. This thesis will therefore put focus on training and evaluating CNNs in order to solve the classification problem. A DFT-based traditional method is also implemented for the purpose of comparison.

When training a convolutional neural network, heavy performance limitations are imposed by the selection of images in the training dataset. As the purpose of the task is to classify specifically surveillance footage, additional requirements are imposed on the datasets used in training. The proposed method should be robust to different weather and light conditions, and be applicable to different types of surveillance cameras. The dataset should therefore consist of relevant and realistic surveillance scenes and contain a variety of weather and light conditions as well as sites. It should also contain images with different resolutions and sizes and preferably originate from several different and typically used surveillance cameras. These conditions imposed on the dataset used for training are crucial for a robust and widely applicable model. A lack of a certain type of scene, such as e.g. snowy landscapes, will lead to poor performance on the specific case and is, therefore, the main limitation of the implemented models.

1.4 Statement of Contribution

Throughout the project, Hanna and Teodora have been working very closely together. The work has been divided so that Teodora has mainly focused on the development of the MANN model and DFT approach. While Hanna mainly focused on developing the Pix2Pix and the MFF model. Both of the authors contributed equally to the discussion and conclusions of the report.

2 Theory

2.1 Neural Networks

Feed-forward neural networks (FNN) are of great importance in machine learning applications. The term neural has taken inspiration from neuroscience, and network comes from the chain of functions corresponding to different layers, composed of a number of interconnected neurons or units. The number of layers gives us the depth of the model, hence the term deep learning. The network consists of neurons in a layered structure and the connections between them. The neurons or units work in parallel, each receiving input from other units in a previous layer and computing an activation value. An example of this structure can be seen in "Trained model" in Figure 2.2. The goal of a feed-forward network is to approximate some function f by defining a mapping $y = f(x;\theta)$ and optimizing the weights θ by training. The architecture of a FNN has the structure of several unit layers where each unit in one layer is connected to all units in the next. This kind of layer is called a fully connected layer [17].

A CNN is a type of FNN, with main applications in image processing and computer vision. The inherited property from the FNN group is that information in a CNN only moves in one direction. The information is forwarded from the input layer, through the hidden layers, and finally to the output layer. One thing that distinguishes the CNN is that its neurons are arranged in three dimensions: height, width and depth. Another difference is that neurons in a certain layer are connected to only a small number of neurons in the previous layer and next layer. These two differences give CNNs the advantage of being capable of considering locality of features. [17].

The most basic layer in a Convolutional Neural Network is a **convolution layer**, giving the network its name. Each neuron in a convolutional network is connected to only a small region in the previous layer. This region is called a receptive field. In the convolutional layer, a filter or kernel is applied to the input. A filter is a set of weights applied in strides, which can be described as a small matrix of weights that slides across the input (e.g. image matrix). The stride represents the number of steps the matrix slides in the input matrix, before each application. When the filter has been applied to the entire input, the result of each application is saved in a feature map, which builds the output of a convolutional layer, see Figure 2.1. For each filter that is applied, a feature map is created. The point of the entire operation is to extract features from the input image or the previous feature map. To begin with, low-level features such as edges and colors are extracted. Further layers can combine these features to identify more complex features, such as entire objects [17].

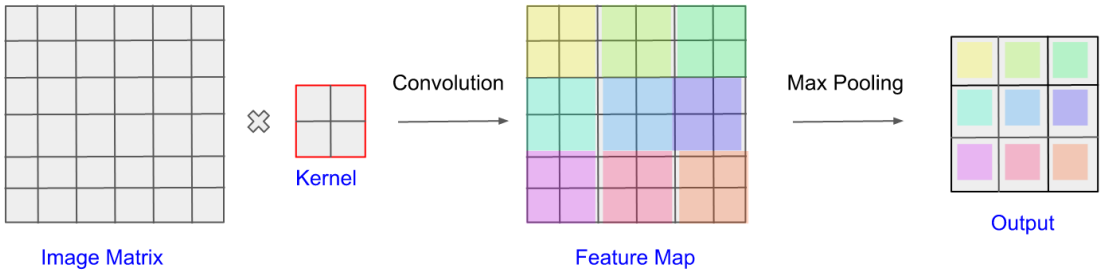


Figure 2.1: Convolution and Max pooling. The largest element of each color box in the feature map is chosen, resulting in the output.

For a single convolutional unit, the computations are shown in Equation 2.1 where x_i is the input that is multiplied by the weight $w_{i,j}$ and added together with the bias b_j. N refers to the number of neurons in the receptive field, or pixels in the input layer. Lastly, it passes through an activation function g to produce the output y_j, further used as input to the units in the next layer [17].

$$y_j = g(\sum_{i=1}^{N} x_i * w_{i,j} + b_j) \tag{2.1}$$

The output y of an entire convolutional layer produces the feature maps. Each element in the maps is the activation value of a neuron in the layer. The feature map carries information on the identified features of the processed neuron inputs.

Other commonly used layers in a CNN architecture are Max pooling, Batch normalization and Dense layers. **Pooling layers** reduce the dimensions of the feature maps by keeping the important features and reducing spatial invariance. Max pooling is the most common type of pooling, performed by taking the highest value in each submatrix. In this way the learnable features are reduced while preserving key features [18]. A simple illustration is shown in Figure 2.1.

Batch Normalization is a technique used to normalize the activation values within the intermediate layers of a deep neural network. This is done by calculating the mean value and standard deviation of the activation values from an entire hidden layer, and the given input data batch. The activation values are then normalized against the calculated mean and standard deviation, so that each neurons output follows the same distribution. The result of this method is improved generalization accuracy and accelerated training. The normalization of the activation functions also help bypass local minima [19].

Dense or Fully connected layers are at the end of a network and are where the decisions are made. Usually, the feature maps are flattened to an array before this step. The reasoning is difficult to follow due to the hidden layers and variable weight for each output from a unit [18].

The most straightforward way to gain performance in a convolutional neural network is to increase the number of layers, making it deeper. This, however, has some drawbacks.

Firstly, it is a trade-off between complexity and processing speed. A larger network produces better accuracy but will not be nearly as fast as a smaller one. Secondly, increasing the complexity with a limited training dataset will lead to over-fitting. Lastly, the gradients will dissipate with a deeper network, making it hard to optimize [20]. These difficulties will be discussed further down, in this section.

When the model has been trained, it can be used to predict on unseen data. In the field of machine learning, *inference* is described as the process of using a trained model and applying it to make decisions and predictions on new data. This could include making recommendations based on learned patterns of behavior, or classifying an image [21] into predefined classes that the model has been trained on. The process of training a neural network and using it for prediction is illustrated in a simple way in Figure 2.2.

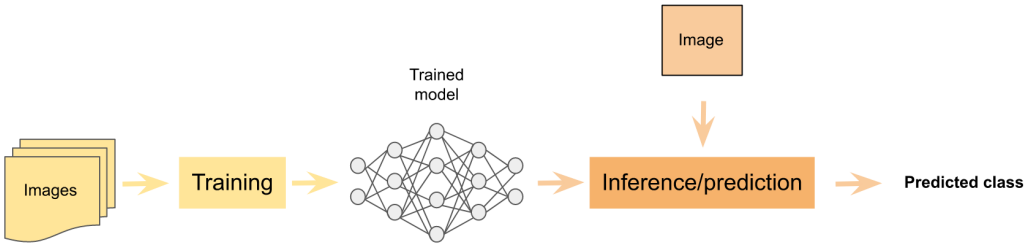


Figure 2.2: Training and inference

2.1.1 Activation Functions

In modern neural networks the most common activation function, g, is ReLU or rectified linear unit, shown in Equation 2.2. Since ReLU is almost linear, it makes it easy to optimize using gradient-based methods [17].

$$g(z) = \max\{0, z\} \tag{2.2}$$

Some other activation functions include Sigmoid and its extension Softmax, shown in Equation 2.3 and 2.4. Sigmoid is used in binary classification while Softmax can be used in multi-class problems [17].

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.4}$$

2.1.2 Weight Initialization

A usual problem with deep neural networks is vanishing or exploding gradients, which can be solved by initialization of weights. Most commonly, a normalized initialization is applied [22]. For normalized initialization a standard Gaussian distribution is utilized to select the weight parameters W, shown in Equation 2.5 where n_i is the i-th layer's dimension [23].

$$W \sim N(0, \frac{2}{n_i}) \tag{2.5}$$

Another initializer option is Kaiming initialization, which is well suited for non-linear ReLU activation and increases the ability of deeper models to converge [24], [23].

2.1.3 Optimizers

Optimizers perform the task of finding the weights θ that reduce a cost-function $J(\theta)$. One popular type of optimizers are those with an adaptive learning rate. The learning rate has a large impact on the model performance and is also one of the more difficult parameters to set.

AdaGrad adapts the rate of learning by scaling the model parameters individually, inversely proportional to the square root of the sum of all old values of the gradient square. The RMSProp algorithm is a modification of AdaGrad, made to perform better in non-convex cases by using an exponentially moving average, instead of all old values [17].

ADAM is another optimizer with an adaptive learning rate and is a method related to RMSProp and AdaGrad. ADAM is an efficient stochastic optimization method that uses first-order gradients from the loss function, and their estimates of first and second moments, to individually compute adaptive learning rates for different parameters. The name ADAM is derived from adaptive moment estimation [25].

$$\theta_t = \theta_{t-1} + -\epsilon * \frac{\hat{s}_t}{\sqrt{\hat{r}_t} + \delta} \tag{2.6}$$

Equation 2.6 describes the updating of the weights θ in ADAM where $\hat{s}_t$ and $\hat{r}_t$ represents the first and second order moments, ϵ is the step size and δ is a small constant used for numerical stabilization [25]. The gradients are calculated through back-propagation, where the results from the cost-function flow backward through the network, the cost-function is the average of the loss functions [17].

2.1.4 Loss Function

Since the last layers in convolutional neural networks usually use Sigmoid or Softmax activation, traditional losses like mean squared error would cause a decrease in effi-

ciency. The most common loss function for classification tasks with CNNs, is instead cross-entropy loss, also called log loss. It is computed using Equation 2.7, where g_j is the label of class j, and p_j is the output of the Softmax or Sigmoid activation layer [26].

$$CE(g, p) = -\sum_{j} g_j log(p_j) \tag{2.7}$$

2.1.5 Over-fitting

Over-fitting is a very common phenomenon in neural networks. The network seems to increase in performance and the training error decreases. However, at some point, the error on unseen examples, i.e. the validation data, gets worse as shown in Figure 2.3, this is over-fitting.

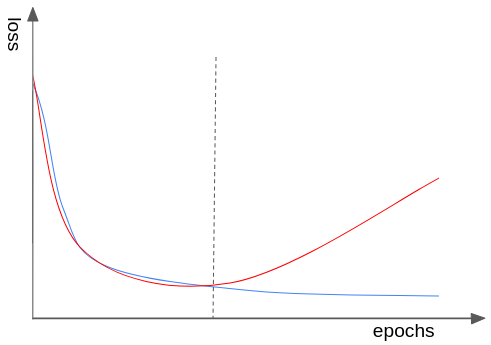


Figure 2.3: Training error (blue) and validation error (red) over time, where the dashed line shows where the over-fitting begins.

Some of the common methods to reduce over-fitting are early stopping and dropout [27].

Early stopping uses the validation data to detect over-fitting and stops the training before convergence. The exact point of stopping is decided by a pre-defined stopping-criterion [27].

Dropout is a commonly used regularization technique. The technique removes units from the network, as well as their input and output connections. The choice of which units to remove is random, but the number of units that are to be removed is pre-defined. The result of the technique is a thinned network with fewer neurons. For each iteration of training, a new thinned network is resampled. A neuron is available with a probability p during training and during the test phase the weights are multiplied with p to ensure that the expected output is the same as the actual [28].

2.1.6 Reducing Model Size

Pruning

Pruning is a method used to reduce computational costs and memory footprint. In the typical case it consists of three parts; training a large model, pruning, and lastly fine-tuning to regain performance. Pruning is usually chosen for models that achieves high performance and that are large and over-parameterized. The pruning can then safely reduce the parameters without hurting the performance significantly. Pruning is therefore usually reported to be better than training a small network from scratch. There are two main types of pruning; structured and unstructured. Unstructured pruning removes individual weights, while structured pruning removes units or even layers [29]. The increase in sparsity in the networks has also been proven to increase robustness to noise [30].

Quantization

To achieve good performance of a deep learning model on a low-power device, quantization is a commonly used method. Deep learning models are usually trained on 32-bit floating point precision numbers, which is supported by CPUs and GPUs. Inference can, however, be done with less precision at a low cost of accuracy decrease.

For a model to be deployed and run on Edge-TPU equipped cameras and in order to further reduce the model size, a TensorFlow Lite (TF Lite) format is necessary. During the conversion, a dynamic range quantization is applied, where the weights are quantized from floating point (float32) to integer, providing 8 bits of precision (int8). Furthermore, this format improves latency, reduces power consumption and processing speed, while enabling hardware acceleration [31].

2.2 Network Structures

2.2.1 Residual Networks

With initialization, deeper networks are able to converge, which has led to a degradation problem. When the depth of a network increases, the accuracy gets saturated to then rapidly degrade, Figure 2.4. This degradation is not caused by over-fitting and adding more layers leads to a higher error when training [22].

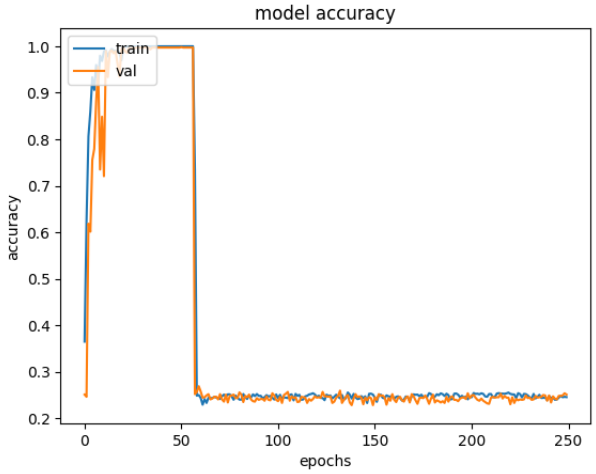


Figure 2.4: Example of rapid degradation problem

He et al. [22] have suggested identity mapping in residual blocks to optimize deeper networks. The residual building blocks use a feed-forward structure with shortcut connections to perform an identity mapping added on after one or more convolutional layers, as seen in Figure 2.5.

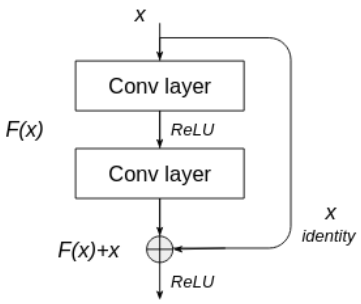


Figure 2.5: Residual block with identity mapping.

In the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), various network structures were evaluated on the large-scale ImageNet image database, with over 14 million images and more than 21 thousand groups or classes. The ResNet architecture won the challenge in 2015 [23].

2.2.2 Multi-scale Feature Fusion Structure

Other problem areas within convolutional neural networks are spatial-information loss and lacking feature representation [20]. The multi-scale structure has been shown to be effective in image classification. Because the structure is designed to use features on various scales, the network can capture more structural information [32]. Especially in the later layers of the network containing more complex semantic features, multi-scale can be crucial and enhance deep-model performance. Multi-scale has the ability to increase performance for image classification without increasing the depth of the model,

which reduces the drawbacks discussed earlier. Furthermore, multi-scale structures have the ability to be combined with other advanced networks such as LSTM, GAN, etc. Multi-scale can be divided into two approaches; multi-scale feature learning and multi-scale feature fusion.

The idea of Multi-scale feature learning is to combine the output of several parallel CNNs into a dense layer, where each CNN model has different contextual input.

Multi-scale feature fusion provides high-quality features and can be divided further into image-level fusion and feature-level fusion. Image-level fusion merges information from multiple images into a single image to produce more comprehensive representations. Feature-level fusion is usually done with a feature pyramid network (FPN) [20], see Figure 2.6. FPN combines features of high- and low-resolution, using a top-down pathway to increase feature information at all levels. An upsampling operation is used to combine the semantic information of the high levels with the spatial information of the lower levels, enhancing the feature characteristics [20].

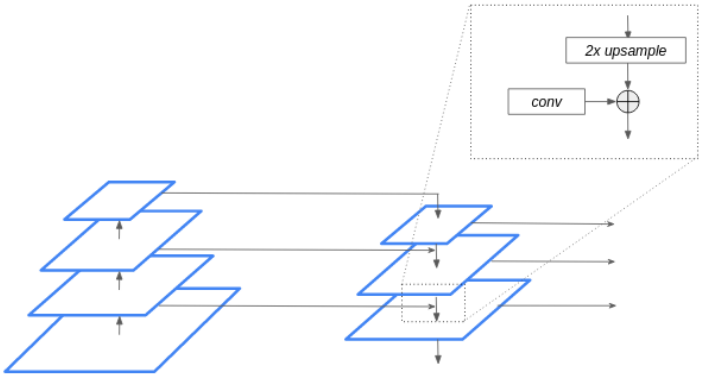


Figure 2.6: Feature Pyramid Network. The blue squares represent feature maps from different layers.

Chen et al. [23] uses a FPN structure together with residual blocks to classify camera failure in video surveillance systems. They argue that multi-scale is important to retain the feature information for blur-anomaly detection.

2.3 Transfer Learning

A limitation of Machine Learning is the large amounts of data needed for training a model. A way to deal with this is the Transfer Learning approach, allowing the knowledge from a previous classification problem to be reused on a different task. The difference between Transfer Learning and traditional Machine Learning is shown in Figure 2.7.

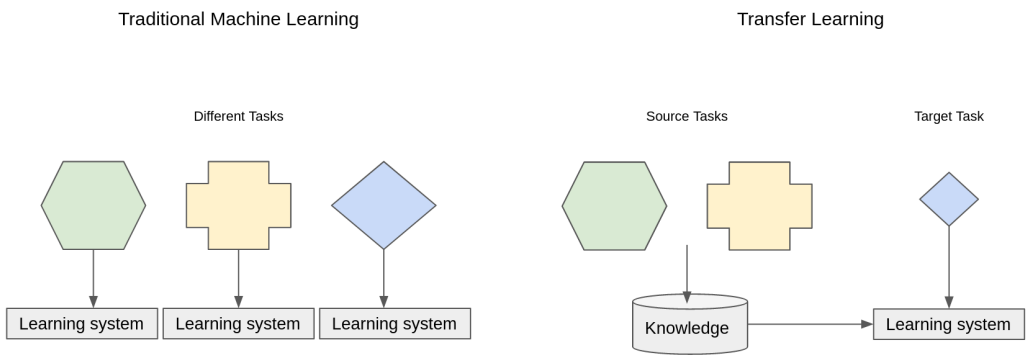


Figure 2.7: Difference between the learning process in traditional Machine Learning and Transfer Learning.

One common type of Transfer Learning is inductive Transfer Learning, where a model is trained on a large labeled dataset (like ImageNet), and then fine-tuned to recognize different classes. When fine-tuning, it is useful to freeze the weights of certain layers in the network. It is possible to choose which layers to fine-tune or freeze, and upon deciding this, the distribution of the data should be considered. The first layers in a network are more general, representing information about low level structures such as edges and corners. As this is highly relevant for the new classification as well, the first layers are often frozen when transfer learning. The dense or fully connected layers at the end of a neural network contain the classification and are very specific to a particular task. Because of this, the last layers are replaced when training on a new dataset and on different classes. There are many publicly available deep neural networks that have achieved state-of-the-art performance in different classification tasks, like the VGG, ResNet, Inception, MobileNet etc. Transfer Learning enables reuse of the knowledge in these large networks [33].

2.4 Data Augmentation

Another way to combat a smaller dataset and reduce over-fitting is by data augmentation. Many augmentation techniques are simple transformations such as horizontal flipping of the image, color space augmentation, rotation or cropping. It is important that the augmentation is label-preserving, for example, a horizontal flip of text data is not a useful transformation [34]. Data augmentation can either be used to increase the number of images in the dataset used for training, or to just add variation to it.

2.5 Gaussian

To add blur to an image, a Gaussian kernel can be applied. The 2D kernel is derived from Equation 2.8.

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.8}$$

Where σ controls the spread of the blur while x and y describe the height and width of the kernel [6].

It should, however, be noted that Gaussian blur in an image differs from optically blurred images. Blur produced by an optical system is a result of the Point Spread Function (PSF) of the lens and depends on a large number of variables. One important trait of PSF-blurring in an image is that it is not evenly distributed across the entire image. The blurring characteristics on different points in the image depend on the image depth, as well as the configuration and shape of the lens [35]. Gaussian blur, on the other hand, is applied to a 2D image and all points are smeared equally.

2.6 Discrete Fourier Transform

The Discrete Fourier Transform (DFT) has been used for many purposes in the field of image processing. The DFT is able to determine the frequency contents of an image, by converting it from spatial into the frequency domain, which allows for alteration or separation of specific frequencies [36].

DFT calculates the frequency contents in a spectrum by using Equation 2.9 [37].

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k n/N}, \quad 0 \leq k \leq N-1 \tag{2.9}$$

In a blurry image, the high-frequency content is lower than that of a sharp image of the same scene [3]. This makes DFT relevant for out-of-focus detection applications. The method can be applied by comparing the high frequency content of two consecutive frames. If the content decreases in the current frame compared to the previous one, an out-of-focus image is detected.

However, there are a few known disadvantages of DFT, as a method for detecting out-of-focus images. Large objects that obstruct the view and people moving in the video frame can cause false alarms due to a significant change to the image characteristics, and therefore also the high frequency content of a frame. Approaches that have tried to combat this, have to increase the number of computations, making the operations hard to do in real time. Furthermore, sudden light changes are another problem that is harder to combat [38].

2.7 Evaluation Metrics

In order to evaluate and compare results from different classification methods, there are several evaluation metrics that have been put to use throughout this project. These will be explained in the following section.

2.7.1 Confusion Matrix

A confusion matrix is a matrix that visualizes how the classification is done on items belonging to specific classes. More specifically, it shows how many predictions were correct and incorrect for each class, and what other classes they are often mistaken for by the model. It is a simple way to display the variation of the classification over different classes and a useful tool for gaining an understanding of tendencies that might be present. The confusion matrix is a very common performance measurement for machine learning classification and is applicable for both binary and multi-class problems.

Binary classification

When binary classification of an image is done, there is a definition of a positive and negative prediction, as well as a positive or negative actual class label [39]. The different combinations of these are as follows: True Positive, True Negative, False Positive and False Negative. True Positive (TP) can be described as an observation (in this thesis, an image) that is predicted as positive, and actually has a positive label. True Negative (TN) is when an observation is predicted as negative, and actually has a negative label. False Positive (FP) means an observation is predicted as positive, but is actually negative. And lastly, a False Negative (FN) is when an observation is predicted as negative, but is actually positive. The different outcomes can be presented in a confusion matrix [39], shown in Table 2.1.

Table 2.1: Binary confusion matrix.

		Prediction	
		Negative	Positive
Actual label	Negative	TN	FP
	Positive	FN	TP

In this report, the negative label represents the sharp class and the positive label represents the blurry class.

Multi-class classification

In the case of multi-class classification, all the mentioned combination metrics (TP, TN, FP, FN) can be produced as well, but instead they need to be calculated for each class individually [40]. A multi-class confusion matrix for N-number of classes is shown in Table 2.2 and the different colors represent the combination metrics for class 2 (C_2). For a classification problem with N classes, the confusion matrix should have a dimension NxN. Each column in the matrix shows the predicted instances and each row represents the instances of an actual class. A result at row i and column i gives the number of correctly classified instances, which can be interpreted as TP. For i=2, this is marked with green in the Table. A sum of all other elements on the same column i, is then interpreted as FP, marked with yellow in the Table. Similarly, a sum of all other elements on row i, is interpreted as FN, marked with blue in the Table. A sum of the rest represents TN, marked with red. This type of representation of the result brings insight into what sort of errors the model is making, and more specifically, what classes are being mixed up.

Table 2.2: Multi-class confusion matrix.

		Prediction			
		C_1	C_2	...	C_N
Actual label	C_1	$C_{1,1}$	FP	...	$C_{1,N}$
	C_2	FN	TP	...	FN
	...	...	...	...	...
	C_N	$C_{N,1}$	FP	...	$C_{N,N}$

2.7.2 Other Metrics

A very common metric when performing classification problems is accuracy. For binary classification and based on the metrics described above in the confusion matrix, it is calculated according to Equation 2.10 [40]:

$$accuracy(binary) = \frac{TN + TP}{TN + FP + FN + TP} \tag{2.10}$$

For multi-class classification total accuracy is calculated according to Equation 2.11 [40]:

$$accuracy(multi) = \frac{\sum_{i=1}^{N} TP(C_i)}{\sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij}} \tag{2.11}$$

In the case of unbalanced classes, i.e. different number of images in each class, balanced accuracy is used. Balanced accuracy is an average of recalls, Equation 2.12, and gives every class the same weight [41].

$$\textit{balanced accuracy (binary)} = \frac{1}{2}(\frac{TN}{TN+FP} + \frac{TP}{FN+TP}) \tag{2.12}$$

Further, we describe the metrics precision and recall according to [40]. Similar to the combination metrics, precision and recall are calculated the same for both types of classification, but in the case of multi-class, they are calculated for each class individually. The i index in Equations 2.13 and 2.14 below is only applicable for multi-class classification and represents the specific class that the metric refers to.

Precision is the fraction of positive values out of the total predicted positive instances. It is calculated according to Equation 2.13.

$$\textit{precision} = \frac{TP_i}{TP_i + FP_i} \tag{2.13}$$

Recall is the fraction of True Positive elements out of the total number actually positive instances. This is calculated according to Equation 2.14.

$$\textit{recall} = \frac{TP_i}{TP_i + FN_i} \tag{2.14}$$

Macro-recall and macro-precision are metrics used in multi-class classification and are calculated as an average of recall/precision for each class.

False Alarm Rate (FAR) is a metric for comparing the number of false alarms. It is only applied to binary classification and is calculated as the fraction of False Negatives out of the total number of actually positive instances [42], as seen in Equation 2.15:

$$FAR = \frac{FP}{TN + FP} \tag{2.15}$$

As blur is something that is to be detected, a false detection of blur (FP), i.e. a sharp image that is wrongly classified as blur, will be referred to as a false alarm in this report.

3 Method

The work process consists of three major parts that will be described in this section; the dataset acquisition phase, the implementation of models and the testing phase.

3.1 Dataset

The dataset that is used for training and testing of implemented models consists of self-acquired images and images from an Axis database of surveillance images. The latter is made up of carefully chosen images that complement the self-acquired dataset so that training is done on the necessary variety of scenes and cameras. This includes images taken in dim light or darkness, brighter images and footage containing motion blur.

3.1.1 Data Acquisition

The self-acquired images were taken on Axis premises in Lund, using 10 different Axis cameras with varying resolutions and image sizes, shown in Table 3.1. All available cameras have a large depth of field (infinity focus), which means that the possibility of an extremely blurry background in contrast to a very sharp object (in-focus) at the front, are very unlikely. The different cameras include different image processing chips and pipelines, which also impacts the image quality.

Table 3.1: Cameras and their respective resolution and image size.

Camera	Mega pixels	Image size
m3086	4 MP	(2688, 1512)
m3088	8 MP	(3840, 2160)
m4215	2 MP	(1920, 1080)
m4216	4 MP	(2304, 1728)
m4218	8 MP	(3840, 2160)
p3245	2 MP	(1920, 1080)
p3248	8 MP	(3840, 2160)
p3265	2 MP	(1920, 1080)
p3268	8 MP	(3840, 2160)
q3538	8 MP	(3840, 2160)

The cameras were set up on a rig, shown in Figure 3.1, and connected in a manner that allows for manual zoom and focus change. In this way all the cameras could take a snapshot at the same time, making 10 images of the same scene, with a different focus level in different images.

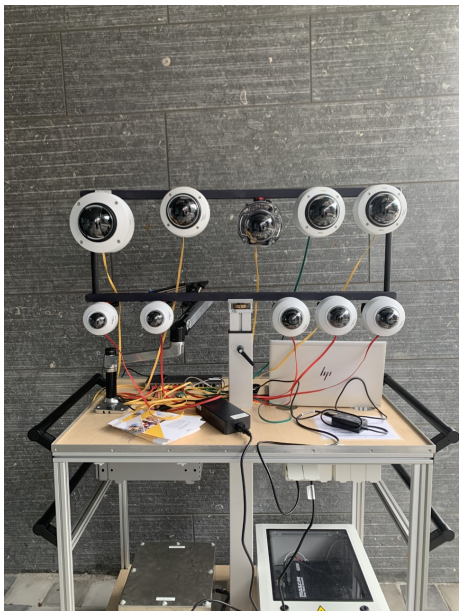


Figure 3.1: Camera Setup

Images were acquired both indoors and outdoors, as well as in different light and weather conditions. This included e.g. sunshine, rain, dim light, etc. Different zoom levels were also used to get a wider range of images. The focus levels are grouped into three classes based on visual inspection: "sharp", "some blur" and "a lot of blur", example images are shown in Figure 3.2. They are referred to as sharp, optical blur degree 1, and optical blur degree 2. The manual focus on each camera was also changed on numerous occasions, in order to acquire images belonging to each of the mentioned classes, from each camera used. For example, when collecting images on day 1, if three cameras are used, they might be configurated as following: camera A is in focus, camera B has an optical blur degree 1 and camera C has an optical blur degree 2. When images are collected the next day, camera A might instead be set on an optical blur degree 2, camera B in focus and camera C on degree 1.

Figure 3.2: Example of acquired images, where the top two are sharp, the middle two have some blur (degree 1) and the bottom two have a lot of blur (degree 2). The images are taken by different cameras, resulting in slightly different fields of view.

The final dataset consists of 1850 sharp images and 2478 images with different degrees of optical blur.

3.1.2 Artificial blur

A problem with using images with optical blur in training is the lack of control of the "blurriness". As described in Section 3.1.1, the amount of applied blur is based on visual assessment and is therefore not in any way objective between images from different cameras, taken on different occasions. If one were to create models and train on the acquired data, there are only two possibilities. The first one is a 3-class model that classifies sharp, blur degree 1 and blur degree 2, just as the data is classified in the data acquisition. The second option is a binary model, where sharp is one class and both degrees of blur make up the second, blurry class.

Now, since the goal is to identify blurry images, having a simple binary or 3-class model is not an issue per se. However, there are a few reasons why a model with more classes,

trained on quantifiable degrees of blur, can provide a more stable classification. Firstly, if the model is trained on distinguishing between a number of gradually increasing degrees of blur, it ought to be more confident in identifying a very low blur in an image, which is a crucial case to our task. Second, having a number of classes with gradually increasing blur makes it possible to set a sensitivity parameter controlling what should be classified as a sharp image. This is significant for the purpose of including a larger variety of different image qualities, where some sharp images coming from low resolution cameras might be considered blurry when compared to sharp images from other cameras. Setting a sensitivity parameter/threshold might also help the classifier reduce the effect of motion blur, in order to correctly classify an image as sharp.

Gaussian blur

In order to create classes with an objectively equal amount of blur, a Gaussian kernel is applied to all the sharp images to create a larger number of classes. Gaussian blur is described in Section 2.5. Since the smallest image size available in the dataset is $(1920, 1080)$, all images were scaled down to these measurements before filtration. The Gaussian kernel is applied and finally, the images are downsampled to the model input size. The process of preprocessing is shown in Figure 3.3, the augmentation cropping step will be further explained in Section 3.1.5.

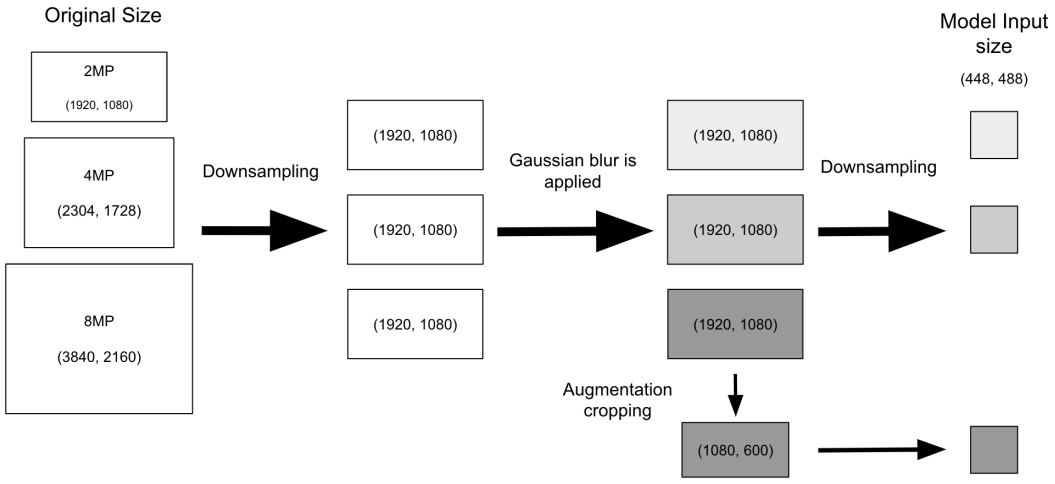


Figure 3.3: Preprocessing of images in the training dataset, showing downsampling and Gaussian blur application. The dimensions are written (width, height).

In Table 3.2, all the possible classes and the applied Gaussian blur in them are described. The table states the standard deviation, Sigma σ, and the Kernel size (x, y), described in Section 2.5. The first column will be explained in further sections. Some example images showing the outcome of the filtration are shown in Figure 3.4.

Table 3.2: Gaussian kernels applied to sharp images to create different blur classes

Binary Class with threshold	Class	Kernel size (x, y)	Sigma σ
Sharp	sharp*	-	-
	b_0.5	(10,10)	0.5
Blurry	b_0.75	(10,10)	0.75
	b_1	(10,10)	1
	b_2	(10,10)	2
	b_3	(10,10)	3
	b_4	(10,10)	4
	b_5	(10,10)	5
	b_7	(10,10)	7

3.1.3 Number of Classes and Threshold

Multi-class classification

The number of classes used in our models is a result of iterative testing. Different combinations of the amount of Gaussian blur and the number of classes were tested.

A class containing images with no blur is denoted as sharp*. Initially, two classes were added that visually match optical blur degree 1 and degree 2, described in Section 3.1.1 and shown in Figure 3.2. These were b_3 and b_7, see Table 3.2 and Figure 3.4. As a gradual increase in degrees of blur is desirable for reasons described in the previous section, a number of classes were added in between the three existing ones: b_1, b_2, b_4 and b_5, see Table 3.2. Another two classes with very low artificial blur were added for reasons described in the next section. These were b_0.5 and b_0.75, making a final total of 9 classes, see Table 3.2.

A comparison was made between a number of scenarios. One option was training on two classes: only sharp images and a mix of different amounts of Gaussian blur. Another option was a multi-class model with a number of classes from Table 3.2. Based on accuracy results, and how confident the model was in predicting the correct outcome, the choice was made to keep all the classes in the table and utilize a multi-class approach.

Binary classification and threshold

The purpose of the models in this thesis is to detect blurry images. The final classification is therefore meant to be binary, identifying images as simply sharp or blurry. This means that the 9-class models should be transformed into binary ones, by defining what classes should belong to each of the sharp and blurry categories. Note that the sharp category in binary testing is denoted as simply sharp, and the sharp class in multi class classification is denoted as sharp*, also shown in Table 3.1.2. When per-

Figure 3.4: Example of the same image before and after applying different Gaussian kernels. In order from the top; sharp, $\sigma = 3$ and $\sigma = 7$.

forming binary classification with a 9-class model, the approach is as follows. Firstly, a prediction is made by the model and an image is classified as one of the 9 classes. Secondly, we place the predictions into either sharp or blurry depending on which of the two categories the class belongs to according to our definition. Thirdly, the true labels ("sharp" or "blurry") are checked against the defined binary prediction and the classification is evaluated with a metric.

As mentioned previously in Section 3.1.2, being able to set a sensitivity threshold for what degree of blur is to be considered as sharp, is important for making classification over different camera resolutions stable and reducing the effect of motion blur. This threshold has been empirically decided, by gradually increasing it until a satisfactory

result is achieved. The process resulted in setting the binary prediction threshold for a sharp image between class b_0.5 and b_0.75 in Table 3.2. Images predicted as sharp* and b_0.5, according to the table, are therefore considered sharp in binary testing of models. All other classes in the table belong to the blur category in binary classification.

3.1.4 Training and Testing Datasets

Training, Initial and Purpose datasets

The final amount of sharp images in the dataset is as mentioned 1850. A number of randomly selected images from this dataset are used for two separate testing datasets, the Initial testing dataset and the Purpose testing dataset. The conducting and purpose of these are described in section 3.4.

The Initial testing dataset is meant to test the multi-class model on the same amount of classes that it is trained on, i.e. 9 classes. This test evaluates how well the model differentiates between the 9 artificially blurred classes. All the images are randomly selected from each class in the training dataset.

The purpose-testing dataset is meant to test the models on optical blur, in a binary classification. The testing dataset consists of one sharp and one blurry class. The blurry class consists of a number of optically blurred images, acquired as described in Section 3.1.1. This is a mix of images with optical blur degree 1 and degree 2.

The number of images in each class of the training dataset and the Initial testing dataset are described in Table 3.3. The same information about the Purpose testing dataset is found in Table 3.4.

Table 3.3: Training and Initial testing dataset

Class	Total number of images	Training dataset	Initial testing dataset
sharp*	1733	1563	170
b_0.5	1733	1563	170
b_0.75	1733	1563	170
b_1	1733	1563	170
b_2	1733	1563	170
b_3	1733	1563	170
b_4	1733	1563	170
b_5	1733	1563	170
b_7	1733	1563	170

Table 3.4: Dataset for Purpose testing

Class	Purpose testing dataset
Sharp	117
Blur	220 (Blur degree 1: 120 + Blur degree 2: 100)

Video testing dataset

For further testing, a number of videos from different cameras with different characteristics are used.

The test sequences *Q-Realistic* and *M-Realistic*, presented in Table 3.5, each consists of four separate realistic surveillance scene videos that are put together. These are recorded by us. The common factor for the four videos is that they are all taken with the same camera. This is either camera q3538 or m4215, hence the name Q or M. The q-camera has a higher resolution compared to the m-camera, which is the reason for the two separate datasets. The four videos in each camera group are shot the same way with both cameras, i.e. images are taken of the same scene, at the same time and with varying optical blur. One frame per second (1 fps) is saved from the two large sequences. The resulting number of frames in each sequence is shown in Table 3.5. The optical blur is documented by annotating each frame as "sharp" or "blurry". The four videos that each of the sequences are made of are described as follows:

- *Indoors*: The video is shot indoors in an office environment, with a lot of movement and people passing by. It is taken from a high standpoint, which gives the images plenty of depth with a distant focal point.
- *Small window*: The video is shot from a window. Cars are visible in the distance, as well as a number of wavering flags in intense windy conditions.
- *Large window*: This video is shot from a different window and the majority of the the frame is covered by the Axis headquarter building. The video is taken from a high standing point, giving a large depth of field. There is limited amount of movement, with only a few passing people.
- *Road*: This video is shot next to a busy road, with plenty of passing cars.

The same protocol is used when recording the video of each individual scene. Before the recording starts, autofocusing is done with zero zoom, meaning the cameras maximum field of view is used. The first part of the video is therefore in focus, upon which we manually change the focus level to achieve optical blur degree 1, and then take it back to in focus. Next, another change of the focus is made, but this time to optical blur degree 2, and then back to focus again. Lastly, we zoom in on the camera, autofocus, and then repeat the described process of changing the focus level to optical blur degree 1, and finally to optical blur degree 2. Focus is restored and the video ends in focus.

Another two combined videos are constructed from the axis surveillance database. Since these are existing videos, they are in focus the whole time and are not expected to contain any blurry frames at all. The first one is called *Daylight* and consists of videos taken in broad daylight, but on a more gray and slightly rainy day. The videos contain footage of a road with passing cars. The videos are taken with an unknown camera, with a seemingly lower image quality than the m and q camera.

The second combined video is called *Darkness* and is, as the name suggests, taken in low-light conditions in outdoor settings. The containing scenes are a mix of roads, city streets and fields. The videos are taken with several unknown camera types and with varying resolutions.

Just as previously, 1 fps is saved from each of the two videos, *Daylight Darkness*, see Table 3.5. All frames from are annotated as "sharp".

Table 3.5: Video testing

Video sequence	Camera type	Total number of images	Number of sharp/blurry
M-Realistic	m4215 with 2 MP	1006	871/135
Q-Realistic	q3538 with 8 MP	1084	835/249
Daylight	unknown	184	all sharp
Darkness	unknown	423	all sharp

Additional datasets for corner case testing

Some images were not part of the scope of this thesis, but were still regarded to be of interest in seeing the limitations of the model. These are small datasets that contain fisheye images, IR images and frames showing almost only walls. The Fisheye images were cropped so that none of the black background surrounding the round image is included. The videos, including the number of frames and camera type, are all stated in Table 3.6 below.

Table 3.6: Corner case testing

Video sequence	Camera type	Total number of images	Number of sharp/blurry
Fisheye images	unknown	12	all sharp
IR images	unknown	134	all sharp
M-Wall	m4215 with 2 MP	60	all sharp
Q-Wall	q3538 with 8 MP	44	all sharp

3.1.5 Data Preprocessing

Preprocessing consists of standardizing the data and two special cases - grayscale and data augmentation.

Input size and normalization

The input shape of the models implemented in this thesis is (width, height, channels), where the last number refers to the number of color channels. In an RGB image, the three channels are red, green, and blue.

To normalize the images and make the input size smaller, all images were sized down. A lot of common image classification models (AlexNet, Resnet, VGG-16, etc) use an input size of $(224, 224)$, referring to (width, height). The benefit of using a smaller size is a lower computational complexity and a smaller model, resulting in decreased inference time. However, the surveillance images that are primarily used for training in this report are originally a lot larger than this, commonly $(1920, 1080)$ or larger. If these images were to be resized to the mentioned $(224, 224)$, the quality of the images would be significantly degraded. This could make it harder to detect an actual blur anomaly, since downsampling causes an image to appear sharper [43]. For this reason, a larger input size of $(448, 448)$ has been chosen for the implemented models. Inspiration for this is taken from Chen et al. [23]. All images that are used for training are therefore bilinearly resized to $(448, 448)$, which distorts the aspect ratio.

Finally, all images were normalized to between 0 and 1, because the training of a neural network relies on gradient calculations, which is improved by normalizing pixel values.

Grayscale

When training a model on grayscale images, the color model of the images is changed from RGB to grayscale, meaning the input size is changed from (width, height, 3) to (width, height, 1). The single channel represents the intensity of light in each pixel, instead of different colors as in RGB images.

Data augmentation

Since all the classes only vary in the amount of blur, introducing other kinds of variation was seen as favorable as well as a possibility to further reduce over-fitting. Data augmentation is a good option for achieving these effects, described in 2.4. When choosing the type of augmentation, the risk of making the frame unrealistic was taken into account. The augmentation was applied before normalization of the images and prior to training. Several augmentation methods were attempted, but only *random crop*, shown in Figure 3.5, had a positive influence on the accuracy. The "random" in the name is referring to the cropping position in the image being arbitrarily chosen. The final size of the cropped image is set to $(1080, 600)$, which changes the aspect ratio slightly. The cropping was also thought to simulate zooming making the dataset more diverse. In order to also keep originally sized images in the dataset, only 50% of randomly chosen images were augmented. This application method does not increase the size of the final dataset.

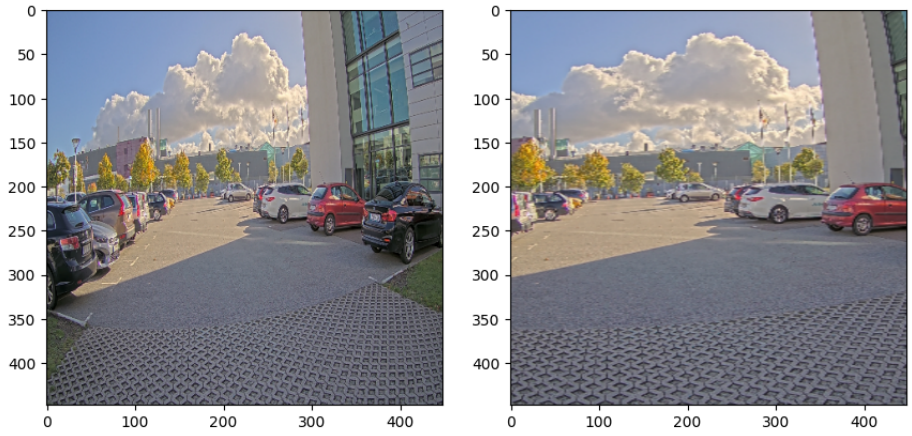


Figure 3.5: Fullsize and cropped image, both sized down to $(448, 448)$

The entire preprocessing pipeline including data augmentation is shown in Figure 3.3 in Section 3.1.2.

3.2 Deep Learning Models

3.2.1 Multi-scale Feature Fusion Residual Network – MFF

The architecture from article [23] was replicated without any changes, and is illustrated in Figure 3.6. The structure can be divided into two parts. The first part consists of two convolutional layers and four residual blocks. The second part represents a commonly used multi-scale feature fusion structure, the feature pyramid network (FPN).

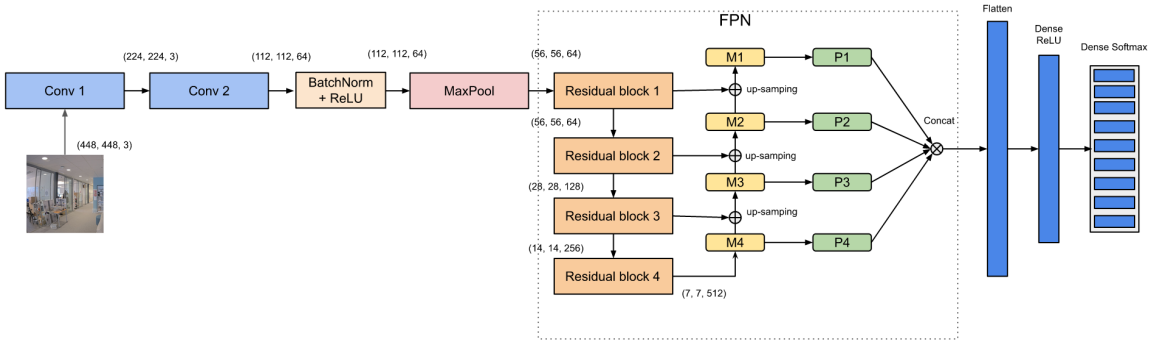


Figure 3.6: MFF architecture

The first part of the network is shown in Table 3.7. The input is set to $(448, 448, 3)$ and passed through the first 2 convolutional layers with kernel size $(7, 7)$ and stride 2, resulting in 64 feature maps with the size $(112, 112)$. All convolutional and dense layers in the network use Kaiming initialization, and padding is used in all instances of the network. The second convolutional layer is followed by batch normalization and ReLU activation, to avoid over-fitting and speed up the training. Further, a

MaxPooling layer was added, with a $(3, 3)$ window and a stride of 2, making the feature maps $(56, 56)$. Following that, 4 Residual blocks are constructed. All the blocks contain 2 convolutional layers with a ReLU activation in between. A shortcut connection is added from the beginning of the block, to after the second convolutional layer. This is used for identity mapping as described in Section 2.2.1, Figure 2.5, and results in downsampling. The block ends with a second ReLU activation. In the four residual blocks, all the convolutional layers use a kernel size of $(3, 3)$. The stride was set to 1 for most layers, except the second weighted layer in the last 3 residual blocks, where the stride is set to 2. Residual blocks are used because they reduce vanishing and exploding gradient problems.

Table 3.7: First part of the MFF architecture, not containing FPN structure

Layer	Layer size
Input Layer	(448, 448, 3)
Conv layer 1	(224, 224, 3)
Conv layer 2	(112, 112, 64)
Batch Normalization	(112, 112, 64)
ReLU Activation	(112, 112, 64)
Max pooling layer	(56, 56, 64)
Residual block 1	(56, 56, 64)
Residual block 2	(28, 28, 128)
Residual block 3	(14, 14, 256)
Residual block 4	(7, 7, 512)

The second part of the network, the FPN structure, is created by fusing the different scale feature maps together, where the lateral connections are convoluted with upsampled feature maps, and finally concatenated. This is illustrated in Figure 3.6 and explained earlier in Section 2.2.2. The fusion of different scale feature maps results in receptive fields at different scales, which can benefit image classification [23].

In the FPN part of the network, the outputs of the residual blocks are upsampled and added together to create the feature maps marked with a M in the figure (3.6). Average pooling is applied to smooth the feature maps and reduce the aliasing that comes from the upsampling, making the output $(7, 7, 64)$. The outputs are concatenated together and flattened to (12544). Finally, 2 dense layers are added, the first one with a ReLU activation, reducing it to (64), and then a Sigmoid activation, making the multi-class prediction into 9 classes.

3.2.2 Pix2Pix Network

The Pix2Pix Discriminator, as it is described in the paper by Attar et al. [44], is replicated with an added change. The paper's primary goal is to deblur artificially blurred images, but their method also has a component intended specifically to classify images as blurry or sharp. This classifier is a CNN, embedded in a Generative Adversarial Network (GAN) model. The classifier is called the discriminator and is the only part

implemented for this report. The change introduced in this thesis is a different input size. The original paper uses $(256, 256, 3)$, which we argue is too small for large surveillance images, and will lead to a heavy loss of information when resizing. We use the input size of $(448, 448, 3)$, which consequently leads to a change in the size of all feature maps in the network. The network is still implemented according to the exact description from the paper. Additional optimization of the network, as well as adding layers was attempted but did not improve the model for our task. The architecture is described in Table 3.8.

Table 3.8: Pix2Pix Architecture

Layer	Layer size
Input Layer	(448, 448, 3)
Conv block 1	(224, 224, 64)
Conv block 2	(112, 112, 128)
Conv block 3	(56, 56, 256)
Conv block 4	(28, 28, 512)
Conv block 5	(14, 14, 512)
Conv block 6	(7, 7, 512)
Flatten	(25088)
Dense ReLU	(64)
Dense Softmax	(9)

The network consists of 6 convolutional blocks followed by a flattening layer and two dense layers. Each Conv block consists of a convolutional layer followed by a batch normalization and a ReLU activation. All convolutional layers use padding, a kernel size of $(4, 4)$ and a stride of 2. A normal initialization is added to all layers. The architecture is illustrated in Figure 3.7.

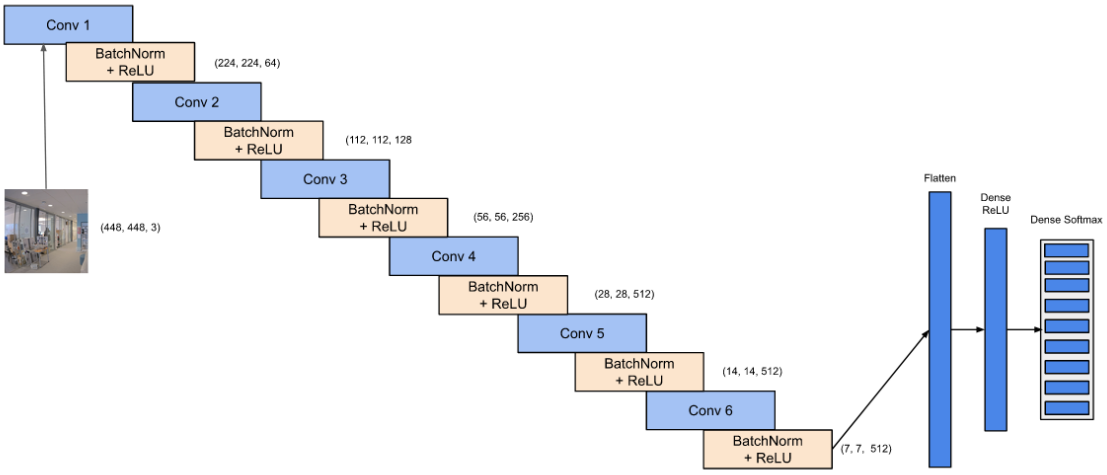


Figure 3.7: Pix2Pix architecture

3.2.3 MANN Network

The architecture of the MANN model, proposed by Dong et al. [45], was implemented and altered to suit our task. The paper uses a mix of morphological detection and deep learning to detect different camera anomalies. Only the deep learning part is explored in this report, as the other element aims at improving the detection of anomalies other than blur.

The structure of the model is alternating between convolutional layers, followed by a ReLU activation, and Max pooling layers. All convolutional layers use padding, a kernel size of $(3, 3)$ and a stride of 2, while all Max pooling layers use a window size of $(2, 2)$ and a stride of 2, and do not apply padding. The kernels were initialized by normal initialization.

Table 3.9: MANN Architecture

Layer	Layer size
Input Layer	(448, 448, 3)
Conv layer + ReLU 1	(442, 442, 16)
Max pooling layer 1	(221, 221, 16)
Conv layer + ReLU 2	(219, 219, 32)
Max pooling layer 2	(110, 110, 32)
Conv layer + ReLU 3	(108, 108, 48)
Max pooling layer 3	(54, 54, 48)
Conv layer + ReLU 4	(52, 52, 96)
Max pooling layer 4	(26, 26, 96)
Conv layer + ReLU 5	(24, 24, 192)
Max pooling layer 5	(12, 12, 192)
Conv layer + ReLU 6	(10, 10, 384)
Flatten	(38400)
Dense ReLU	(64)
Dense Softmax	(9)

The network from the paper contains 4 convolutional blocks and 3 pooling layers. A number of layers were added to increase the complexity of the network, in pursuit of increasing the accuracy without over-fitting. For this purpose, the same building blocks were used as described above. The final architecture is described in Table 3.9 and is made up of 6 convolutional layers, 5 Max pooling layers, a flattening layer and 2 dense layers, as in the original network structure. The input size from the original network is unknown. We used $(448, 448, 3)$, for reasons described in the beginning of this Section 3.2. The architecture is illustrated in Figure 4.5.

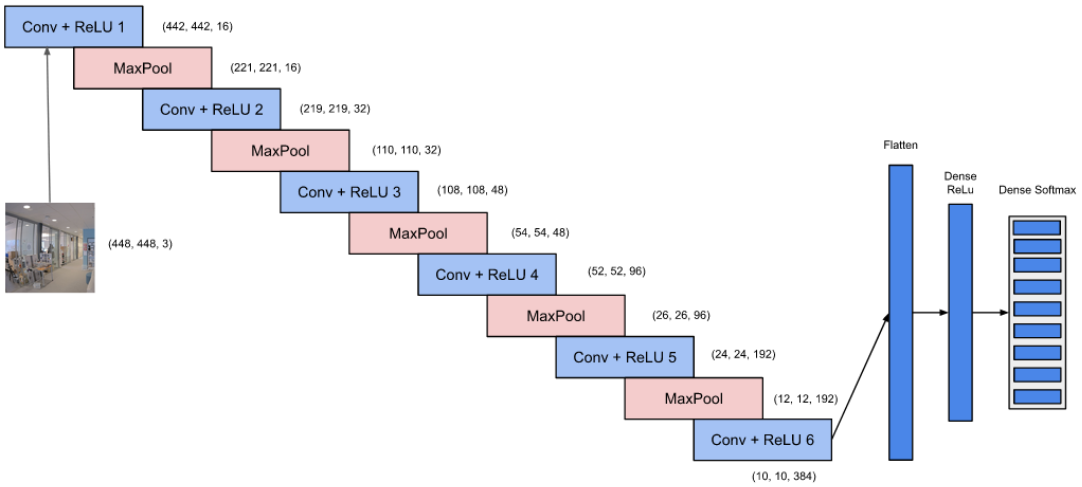


Figure 3.8: MANN architecture

3.2.4 Training of Models

The models are trained on the training dataset with artificial Gaussian blur in 9 classes, described in Table 3.3. All the models have the same input size of $(448, 448, 3)$. The three models used categorical cross entropy as the loss function, described in Section 2.1.4. The optimizer function used in all three models is ADAM, with an initial learning rate of 0.001, described in Section 2.1.3. All weights are saved during training and as a final step, the best weights are applied to the models when the training is completed.

Different augmentation methods were tested to optimize the results on the models with the best accuracy. The augmentation methods are described in section 3.1.5. Changing the input from RGB to grayscale images was also tested on the model with the best accuracy.

The final training dataset consists of a total of 14607 images, shown in Table 3.3. During training, the dataset is divided into images for training and images for validation. The dataset is then split by 80/20 – 80% for training and 20% for validation. The batch size was set to 128 for MANN and MFF, and 64 for Pix2Pix. The training batches are shuffled between each epoch.

We used the TensorFlow library [31] in Python 3.9. Training is done on a graphics card with double NVIDIA RTX2080Ti with 12GB of memory(GPU).

3.3 DFT Analysis

The DFT method is inspired by Saglam and Temizel [38], as well as similar methods [3]–[46] described in Section 1.2. Our method is implemented according to the description below.

As briefly described in Section 2.6, an out-of-focus image can be detected by comparing the high frequency content of two consecutive frames. The first step is therefore to eliminate the low frequency content from the image. The filtration is done by first running a low-pass Gaussian filter on the image, with $\sigma = 1.5$ along both axes. Secondly, the value is subtracted from the image, leaving only the high frequency content. The described approach can be considered a self constructed high-pass filter. A DFT is then applied to the newly filtered image, which yields the high frequency values. The DFT method compares the sum of all these high frequency values of the current frame, $E_{HF}(C)$, and the previous frame, $E_{HF}(P)$. P denotes the previous frame and C denotes the current. The image is considered blurry and out-of-focus when the following Equation 3.1 is true [38]:

$$E_{HF}(C) \leq Th * E_{HF}(P) \tag{3.1}$$

Th is a threshold that describes the detection sensitivity. The higher Th is set to, the higher the sensitivity. It takes into account the amount of detail in the compared images and is adaptable to the specific scene. E.g. if the observed scene has large homogeneous surfaces, the amount of high frequency content will not change much when the picture is put out-of-focus. The sensitivity therefore needs to be increased in the described case, in order to become "more sensitive" to the small changes. The threshold is calculated as follows in Equation 3.2 [38]:

$$Th_1 = \begin{cases} L_{Bound} & \text{if } 1 - \frac{E_{HF}(P)}{Max_{HF}} \leq L_{Bound} \\ 1 - \frac{E_{HF}(P)}{Max_{HF}} & \text{if } U_{Bound} \geq 1 - \frac{E_{HF}(P)}{Max_{HF}} \geq L_{Bound} \\ U_{Bound} & \text{if } 1 - \frac{E_{HF}(P)}{Max_{HF}} \geq U_{Bound} \end{cases} \tag{3.2}$$

The lower boundary and the upper boundary are set to values chosen in article [38]: $L_{Bound} = 0.2$ and $U_{Bound} = 0.8$. The Max_{HF} denotes the maximum possible value of $E_{HF}(P)$, and is equal to the highest sum of high frequency content, detected in the training dataset [38]. The value was not stated in the original paper, and was therefore defined by us and based on our own training dataset.

It should be added that this method, although inspired by article [38], it is not applied the exact same way. As mentioned in section 2.6, the DFT approach can have the disadvantage of being sensitive to a rapidly changing scene. In the original paper, [38], this is partially handled by excluding $(8, 8)$ pixel blocks with moving pixels from the equation calculating the high frequency content. Our method is only made for a basic comparison with CNN methods, and will therefore not include the mentioned solution.

Additionally, the DFT method, with an experimentally set threshold, was tested on the same video sequences as the CNN models, described in Sections 3.4.3 and 3.1.4. Although adapting the threshold to the type of data it is tested on would improve the testing results, this was not done. Our task requires a generalized method that is applicable to different settings. There would also not be a way to change it depending on the processed data if the method were to be applied in a camera. It would not be realistic, and therefore, the same threshold is used in all tests.

Implementation of the DFT method was also done in Python 3.9.

3.4 Testing

The original size of the images used in purpose testing and video sequence testing, was between $(1920, 1080)$ and $(3840, 2160)$. Images having a height larger than 2000 were cropped to 75% of the original height, maintaining the aspect ratio. The cropping removes the outer part of the image while retaining the central region in both dimensions. By making the images smaller, the effect of later downsampling is reduced. The process of preprocessing before inference is shown in Figure 3.9, where the original images are sharp or contain optical blur.

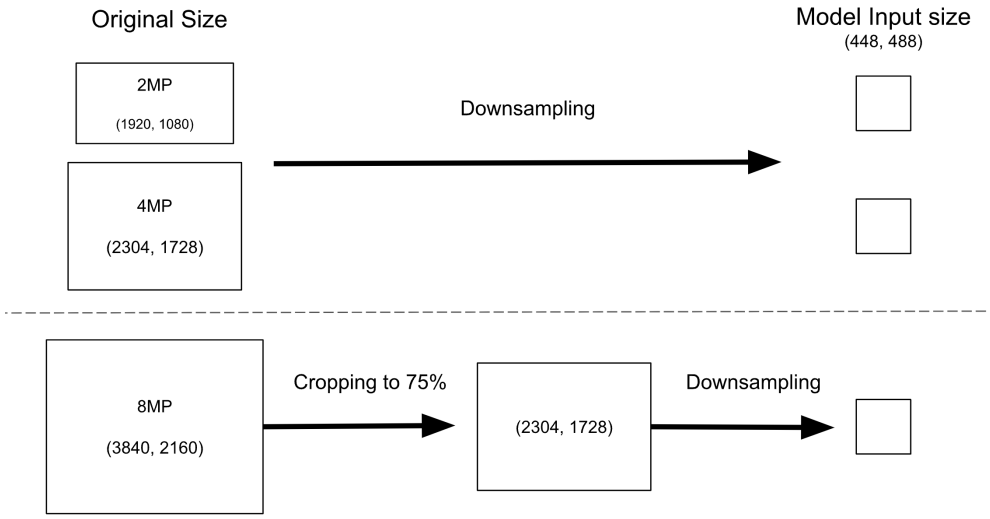


Figure 3.9: The preprocessing of the images before testing. The dimensions are written (width, height)

All images used for testing of both the CNNs and the DFT method, were downsized to the input size of $(448, 448)$. When testing the CNNs, the images were also normalized to between 0 and 1 before being predicted upon by the models. The normalization was not done when testing the DFT approach.

3.4.1 Initial Testing

In order to evaluate all deep learning models equally on the same test set, Initial testing is done by testing the models on the specific task they are trained to do. When the models are trained to classify images into 9 classes of blur, this is done on images with applied artificial Gaussian blur. For Initial testing in that case, there is a designated test dataset containing unseen images with a mix of sharp images and images with an applied Gaussian blur. This dataset is called the Initial testing dataset, described in Section 3.1.4, and details are found in Table 3.3.

Although the Initial test predicts on 9 classes, one could speculate on how well a model will perform in later binary classification, based on the resulting confusion matrix. In Figure 3.10, a "crucial division" is defined along the red circles added to the matrix. The encircled elements show how well the model differentiates between classes closest to the threshold, which is where most misclassifications occur in the binary case. The figure also illustrates the false negatives and the false positives as squares.

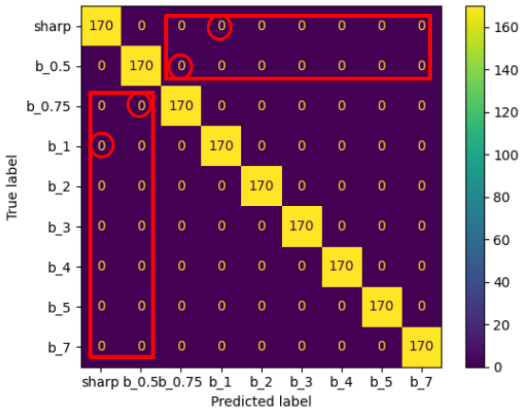


Figure 3.10: Confusion matrix showing the crucial division (circles) and false negatives and false positives (squares) in the binary case. The horizontal square is false positives, and the vertical square is false negatives.

3.4.2 Purpose Testing

The dataset used to evaluate the models on binary classification is called the Purpose testing dataset, described in Section 3.1.4, with details on the dataset found in Table 3.4. The point of purpose testing is to evaluate how well the models perform on classifying images with optical blur, unlike the Initial test that predicted images with artificial Gaussian blur and into 9 classes. Purpose testing is also meant to validate that training on artificial Gaussian blur can be used for the detection of optical blur.

3.4.3 Video Sequence Testing

To compare the DFT approach to the CNN models, consecutive frames are needed in order to compare the previous frame with the current. For this purpose, the videos described in 3.1.4 are used for evaluation and comparison. The videos are meant to show if the methods could be used on different types of cameras, and how well they handle manual blur, zoom and lighting changes. The Darkness and Daylight sequences contain only sharp frames and are therefore used as a test for false alarms.

As the videos contain scenes that are not included in the training dataset, video testing is a good way to evaluate how well the models generalize and further validate the concept.

3.4.4 Corner Case testing

The models are finally tested on the datasets containing Fisheye images, IR images and images containing very little detail. The used datasets are described in Section 3.1.4. As mentioned earlier, Fisheye and IR images were not part of the scope of the project and the models are not trained on such images. The Q-wall and M-Wall sequences, however, are images that are likely to appear in surveillance footage but are an interesting case due to less detail than most regular surveillance images.

3.5 Pruning and Quantization

To decrease the model size and make it more suitable to be run on edge, the best performing model was pruned. Unstructured pruning was performed by removing certain weights during a fine tuning process of the model. During each step of the fine tuning, the sparsity is updated using polynomial decay. The sparsity starts at 0% and ends at 40%. Additionally, the pruned model was reduced even further by applying post-training quantization in the conversion to TF Lite format. The type conversion from float32 to int8 makes it suitable to run on edge in the cameras and compresses it further, as described in Section 2.1.6. For size comparison, the models need to be in a zipped format since a non-compressed model still contains the same size weight matrices before and after pruning even though weights are set to zero.

3.6 Inference Time

Inference time was tested on a graphics card with double GPUs, NVIDIA RTX2080T with 12GB of memory. An average prediction time on 300 randomly generated image size arrays was acquired. A warm-up cycle was run before testing to initialize the GPUs and prevent them from entering power-saving mode.

4 Results

4.1 Initial Testing

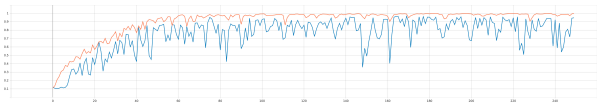
Three different CNN models; MFF, Pix2Pix and MANN, are implemented according to Section 3.2 and trained with an RGB input, according to Section 3.2.4. Initial testing, as described in Section 3.4.1, is the first method used to evaluate the models. The result of the Initial testing is presented in Table 4.1. Another two versions of the Pix2Pix model are created, due to the most promising result upon Initial testing. These are Pix2Pix with a grayscale input, and Pix2Pix with the data augmentation method "crop", both described in Section 3.1.5.

In Table 4.1, the different models and their varieties can be compared by size and result from the Initial testing. The size is shown in the number of parameters and in bytes. The results are shown as accuracy, recall and precision.

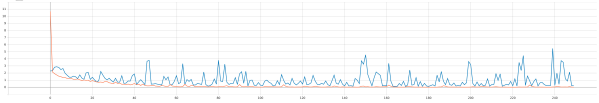
Table 4.1: Metrics results of Initial testing on the different model varieties

Model	Number of parameters	Model size (bytes)	Accuracy (%)	Recall (%)	Precision (%)
MFF	5,768,773	21,459,580	91.89	91.89	92.17
Pix2Pix	12,760,393	47,396,011	**98.04**	**98.04**	**98.05**
+gray	12,758,345	47,421,658	97.19	97.19	97.19
+aug	12,760,393	47,408,628	97.97	97.97	97.98
MANN	3,350,713	12,479,513	91.30	91.30	91.29

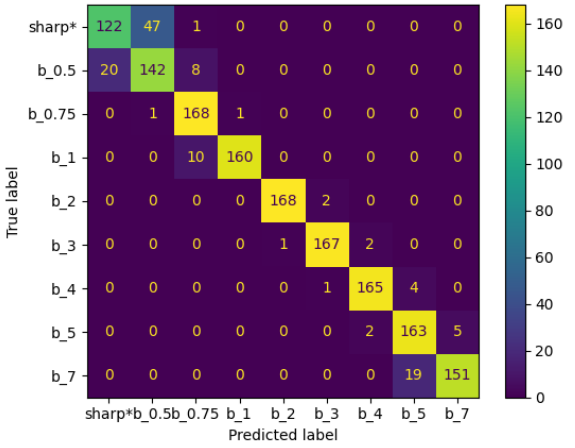
Furthermore, the training and validation accuracy per epoch during training of the models, according to Section 3.2.4, are shown below in Figures 4.1–4.5. Each illustrated training process in the figures is coupled with a confusion matrix, obtained from testing on the Initial test dataset, using the best weights from the respective training process.



(a) Training and validation accuracy per epoch.

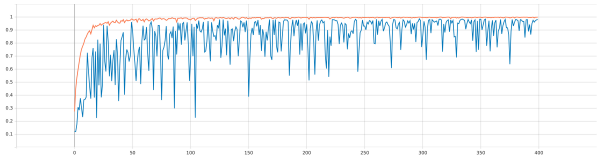


(b) Training and validation loss per epoch.

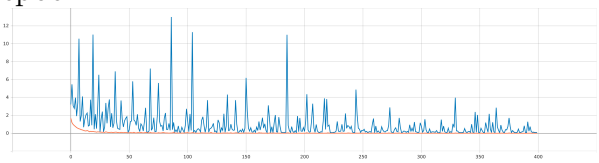


(c) Confusion matrix on test dataset using best weight from epoch 170.

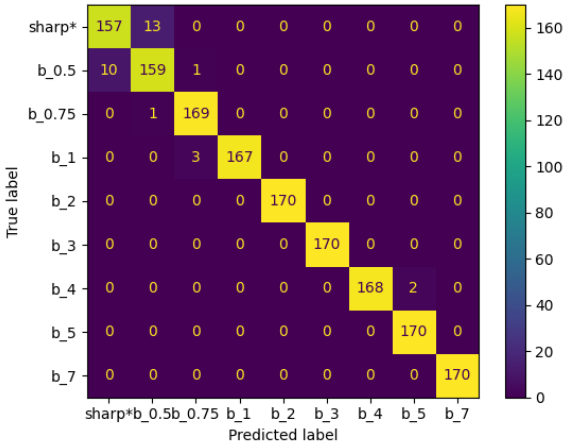
Figure 4.1: Training graphs and confusion matrix on **MFF**. In (a) and (b): The red line is the training data and the blue line shows validation data.



(a) Training and validation accuracy per epoch.

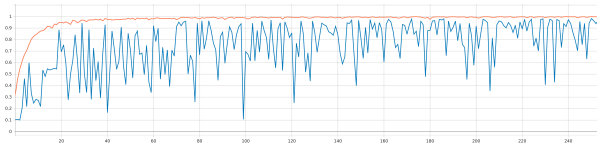


(b) Training and validation loss per epoch.

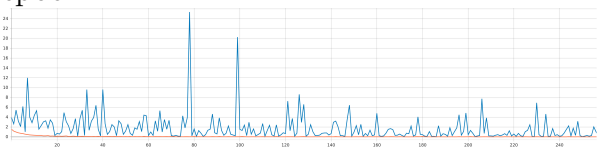


(c) Confusion matrix on test dataset using best weight from epoch 298.

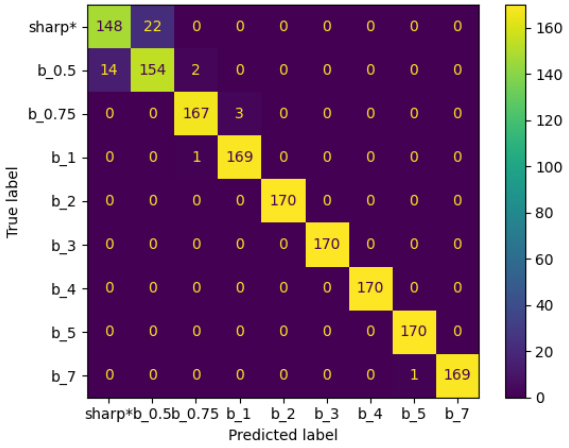
Figure 4.2: Training graphs and confusion matrix on **Pix2Pix RGB**. In (a) and (b): The red line is the training data and the blue line shows validation data.



(a) Training and validation accuracy per epoch.

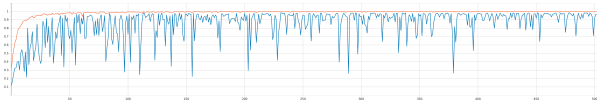


(b) Training and validation loss per epoch.

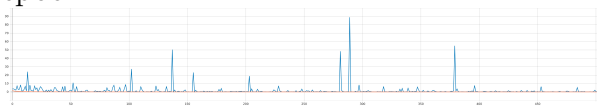


(c) Confusion matrix on test dataset using best weight from epoch 251.

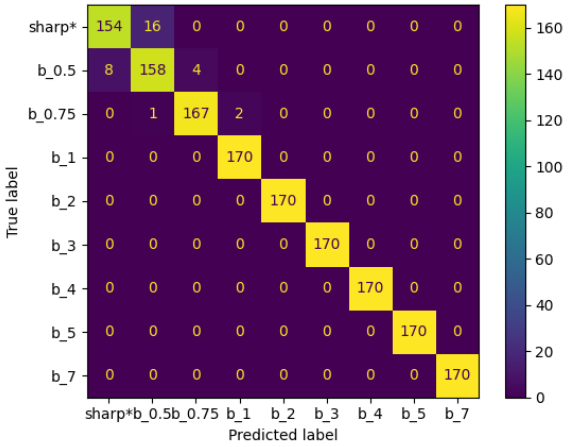
Figure 4.3: Training graphs and confusion matrix on **Pix2Pix Grayscale**. In (a) and (b): The red line is the training data and the blue line shows validation data.



(a) Training and validation accuracy per epoch.

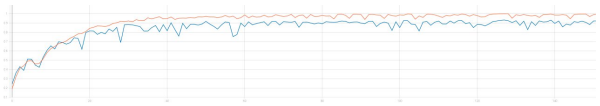


(b) Training and validation loss per epoch.

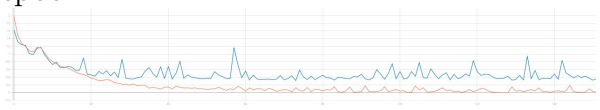


(c) Confusion matrix on test dataset using best weight from epoch 479.

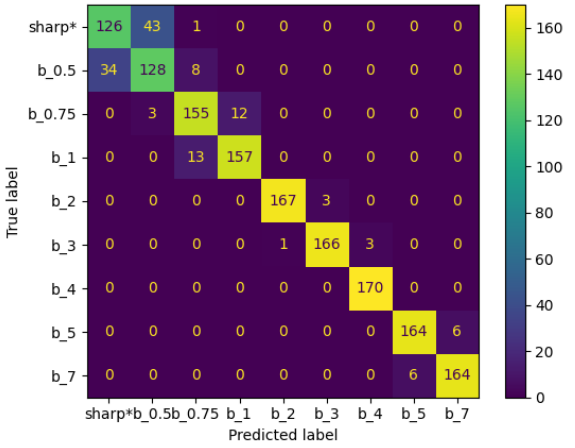
Figure 4.4: Training graphs and confusion matrix on **augmented Pix2Pix**. In (a) and (b): The red line is the training data and the blue line shows validation data.



(a) Training and validation accuracy per epoch.



(b) Training and validation loss per epoch.



(c) Confusion matrix on test dataset using best weight from epoch 140.

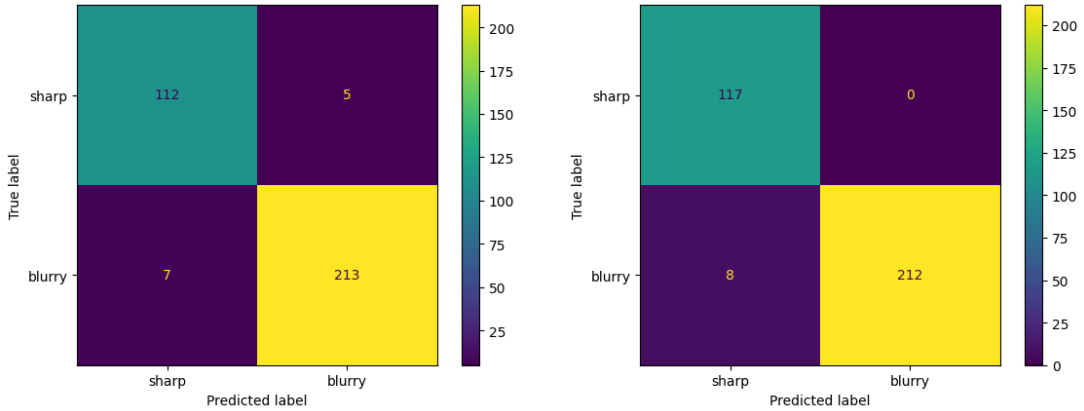
Figure 4.5: Training graphs and confusion matrix on **MANN**. In (a) and (b): The red line is the training data and the blue line shows validation data.

4.2 Purpose Testing

Purpose testing is conducted as described in Section 3.4.2. By using the dataset for Purpose testing and making predictions using the different model varieties, the following results were obtained, see Table 4.2. The resulting confusion matrices for the same varieties of models are found in Figure 4.6.

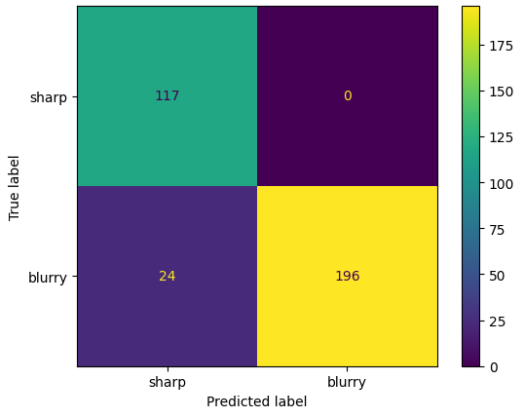
Table 4.2: Metrics results of Purpose testing on the different model varieties.

Model	Accuracy (%)	Recall (%)	Precision (%)	FAR (%)
MFF	96.27	96.82	97.71	4.27
Pix2Pix	98.18	96.36	100.0	0
+grayscale	94.55	89.09	100.0	0
+augmentation	98.41	96.82	100.0	0
MANN	96.93	97.27	98.17	3.42

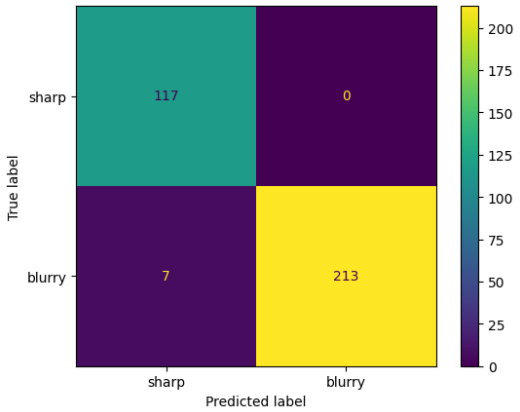


(a) MFF

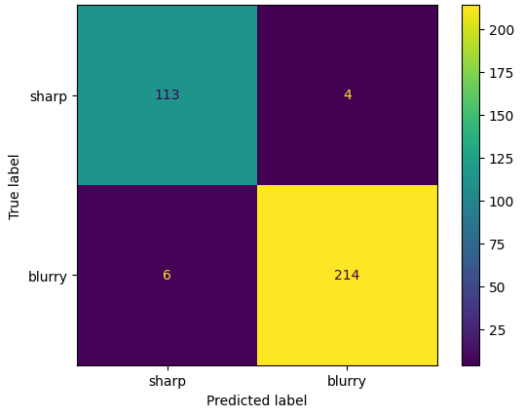
(b) Pix2Pix RGB



(c) Pix2Pix Grayscale



(d) Augmented Pix2Pix



(e) MANN

Figure 4.6: Confusion matrices of Purpose test on all models and their varieties.

4.3 Video Sequence Testing

The four video sequences were tested on the best versions of the three models and the DFT method, according to Section 3.4.3. The resulting accuracies are shown below in Table 4.3. The last two sequences (Daylight and Darkness) only contain sharp images and therefore only test for false positives (false alarms).

Table 4.3: Accuracy (%) for the different video sequences for the different models.

Accuracy	Q-Realistic	M-Realistic	Daylight	Darkness
MFF	94.78	98.52	83.69	86.99
Pix2Pix	94.58	91.11	100	99.53
(+) aug	94.78	97.78	100	99.29
MANN	94.78	97.41	55.43	80.14
DFT	98.59	99.63	100	83.92

Furthermore, precision and recall are shown in Table 4.4. Since the last two videos, Daylight and Darkness, only contain sharp frames precision and recall will result in 0 and are therefore excluded from the tables.

Table 4.4: Recall and Precision for the different video sequences for the different models

(a) Recall (%)

Recall	Q-Realistic	M-Realistic
MFF	89.56	97.04
Pix2Pix	89.16	82.22
(+) aug	89.56	95.56
MANN	89.56	94.81
DFT	97.19	99.26

(b) Precision (%)

Precision	Q-Realistic	M-Realistic
MFF	100	100
Pix2Pix	100	100
(+) aug	100	100
MANN	100	100
DFT	100	100

The False Alarm Rate (FAR) for each model, upon testing on the video sequences, is shown in Table 4.5.

Table 4.5: FAR, false alarms rate (%), for the different video sequences for the different models.

FAR	Q-Realistic	M-Realistic	Daylight	Darkness
MFF	0	0	16.3	13
Pix2Pix	0	0	0	0.47
(+) aug	0	0	0	0.71
MANN	0	0	44.56	19.85
DFT	0	0	0	16.27

4.3.1 Prediction Process illustration

Below is an illustration of the video sequence prediction process. The images are processed and predicted by the model in their true order from the video sequences, and the graphs in Figure 4.7 depict a prediction with an orange dot and a true label with a blue x. The positive axis represents the Blurry class and the negative represents the Sharp class. The prediction dots are placed on 1 if predicted as sharp* or b_0.5 and on -1 for everything above b_0.75, according to Table 3.1.4.

The reason why there are "islands" of blurriness and sharpness along the x-axis is because the optical focus of the images was manipulated during recording, described

in Section 3.1.4. As described in the same section, several videos were also put together into one, which explains the large number of "islands". The presented figures also only show the cases which are of special interest, i.e. we do not present the video sequence prediction process for each video sequence for each of the 5 models. Note that the x-axis is cropped to show the areas of interest only.

The graphs in Figure 4.7 show the illustrated prediction process of the Pix2Pix RGB and the augmented Pix2Pix on the M-Realistic video. The x-axis is divided into four parts, referring to each of the videos placed in this video sequence, see Section 3.1.4. According to the protocol described in the same section, each individual video in this sequence is subject to focus changes. The first two islands in a part are optical blur degree 1 and optical blur degree 2, at maximum field of view (no zoom), and the second two are optical blur degree 1 and optical blur degree 2, in a zoomed in frame.

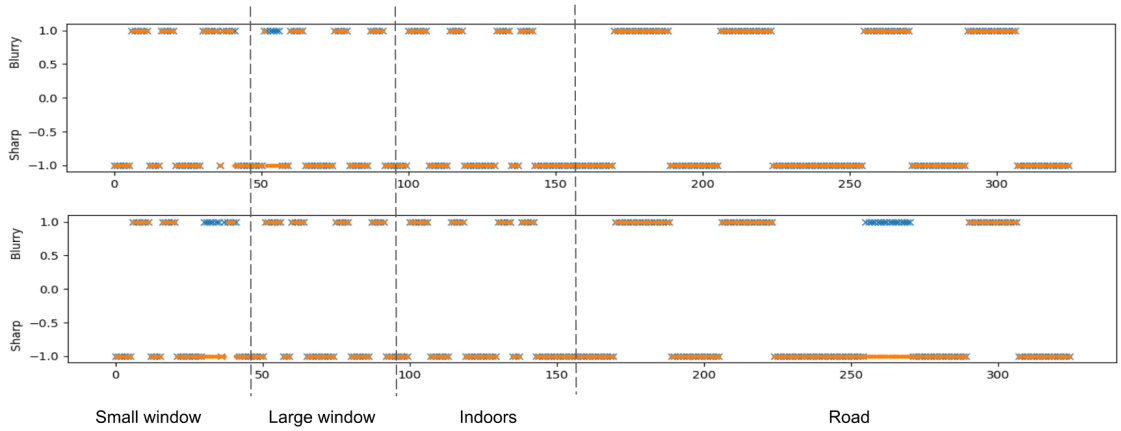


Figure 4.7: Prediction process on M-Realistic by augmented Pix2Pix on top and by Pix2Pix RGB on the bottom. Prediction is marked with an orange dot and the true label with a blue x.

4.4 Corner Case Testing

The corner cases were tested on the models to get a better picture of the limitations. Every model got equal performance results, which are shown in Table 4.6. Fisheye images were not available in the training dataset.

Table 4.6: Balanced accuracy (%) from testing of corner cases.

Accuracy	Fisheye	M-Wall	Q-Wall
All models	100.0	100.0	100.0

On the IR images, all models except for Pix2Pix grayscale got 20.9% accuracy. The grayscale input resulted in an accuracy of 0%.

4.5 Pruning and Quantization of Augmented Pix2Pix

As seen in Table 4.1, the size of the different models varies between 12MB and 47MB, where the Pix2Pix model is the largest among them. In an effort to reduce this gap, pruning was applied to the augmented Pix2Pix model, increasing the sparsity to 40%. By applying the pruned model on the Purpose test dataset, the confusion matrix in Figure 4.8 is obtained. Finally, the model is converted into a TF Lite format, quantizing it. The new post-pruning and post-quantization sizes and accuracy on the Initial test are shown in Table 4.7. The size of the final TF Lite model is reduced to 20% of the original model.

Table 4.7: Model size and test accuracy on the Initial test dataset, before and after pruning and quantization.

Model	Model size (bytes)	Accuracy (%)
Pix2Pix aug	47,408,628	97.97
Pruned	41,475,429	97.84
Quantized TFlite	9,775,431	97.78

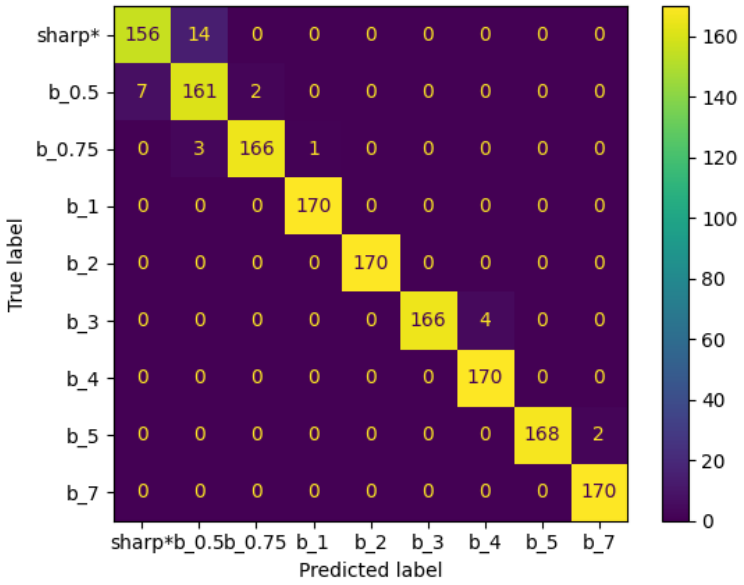


Figure 4.8: Confusion matrix showing Purpose test on Pix2Pix with augmentation after pruning

4.6 Inference Time

An average inference time was calculated for each of the models, including the pruned augmented Pix2Pix, see Table 4.8.

Table 4.8: Inference time of the different models

Model	Inference time (ms)
MFF	2.0
Pix2Pix	2.8
+aug	2.8
+aug +prune	2.8
MANN	2.0
DFT	26.4

Since the format of the quantized TF Lite model is changed for being able to run on TPU and not GPU, its inference time can not be compared to the other models in the original format.

5 Discussion

5.1 Gaussian Filter and Number of Classes

To produce artificial blur, a Gaussian kernel was initially chosen due to its simple application method. Khajuria et al. [6] used a number of different blurring methods to expand the training dataset and make a more general model. This was not attempted on our models, as the performance in early testing showed sufficient results. The models were successful at identifying optical blur, upon being trained on artificial Gaussian blur. Applying different kinds of blurring filters could, however, be a case for further development of the implemented learning-based models.

Exploring the different blurring techniques mentioned in [6] could be especially useful if the model is expanded to detect fog, rain or dirt on the lens. In those cases, it would be important to produce images with synthetic blur that is more alike optical blur than Gaussian blur is. There are a number of different blurring filters that are commonly used and available, such as box blur, shape blur, radial blur, surface blur, motion blur, and lens blur [47]. One of these might be better suited for the task of differentiation between a specific anomaly or weather condition, and optical blur.

The reason for choosing 9 classes for our multi-class model is explained in Section 3.1.3. While fewer classes demand fewer parameters and are therefore less computationally costly, the performance improved when introducing a gradually increasing range of blur. As an example, images including point light sources, motion blur, occlusion and dark images, were initially wrongly classified as blur in all approaches. All of these limitations were improved when introducing the multi-class approach with a threshold above the b_0.5 class. Increasing the dataset with more representative surveillance images also helped to improve the performance further.

5.2 Classifier Performances

In this section, the learning-based models are compared and discussed, and are therefore referred to as "the models". The DFT approach is discussed separately in the upcoming Section 5.3.

5.2.1 Model Resolution Span

All models, except Pix2Pix, achieved a lower detection accuracy on the 8MP Q-Realistic video sequence, than on the 2MP M-Realistic, see Table 4.3. A possible explanation could be found in the preprocessing steps before training, where all images are scaled down to $(1920, 1080)$, before Gaussian blur is applied, explained in

Section 3.1.2. After applying the Gaussian blur, all the images are once again resized to the model input size of $(448, 448)$. The reason that all blur was applied to the same sized images was to create equally blurred classes. However, this means that the 8MP training images are downsampled once before applying Gaussian, from $(3840, 2160)$ to $(1920, 1080)$, and then another time before being used in the training, from $(1920, 1080)$ to $(448, 448)$. The 8MP test images containing optical blur from the beginning, are downsampled directly from $(3840, 2160)$ to $(448, 448)$. This means that the blur in 8MP test images is downsampled more than the blur in the training images, since downsampling an image causes it to appear sharper, according to Trentacoste et al. [43]. Low optical blur in the 8MP test images seemingly becomes to low to be detected by the model. The result of this is that the low blur in high resolution images is mistaken for sharpness in comparison to low resolution images.

To combat this in the testing phase, the images in the test datasets coming from the 8MP cameras were cropped to 75% of their original size, as described in Section 3.2.4. Nonetheless, after cropping, they are still larger than the 2MP images. Heavier cropping of the 8MP test images was attempted in order to reduce the blur loss in the testing stage, but did not prove to increase the accuracy further. Possibly because cropping introduces a zoom effect and leads to a loss of detail in the image instead.

The simplest solution to this problem, is to train on 8MP and 2MP images separately, or reduce the span of resolutions present in the training dataset. The drawback of this, is producing models that are less suitable for application on a wide variety of cameras with different resolutions. Alternatively, a larger model input size could be used. This too has its downside, as training the model would require heavier computation to process larger images and the models would also increase in size.

5.2.2 Performance of MFF

With 5.7 million parameters, the MFF model is a medium-sized model in comparison to the other models. In the Initial testing, the MFF model achieves a relatively low accuracy of 91.89%, compared to the other models investigated in this thesis. Looking at the confusion matrix, it is clear that the model is struggling with differentiating between sharp* and b_0.5, but also shows more than a few misclassifications between b_0.75 and b_1, as well as b_5 and b_7. It also has a total of 9 misclassified images in the crucial division, between b_0.5 and b_0.75.

Furthermore, in the Purpose testing, MFF was outperformed by all the other models except Pix2Pix grayscale. The reason for the lower accuracy score is mostly due to a lower precision score than the other models. The lower precision score originates from generated false alarms, which in a detection system is more severe than missing a few minor events.

Even so, during the Video sequence testing, MFF renders the highest accuracy on the M-Realistic video. Interestingly, the MFF model, as well as most other models, performed worse on the 8MP Q-Realistic video than on the 2MP M-Realistic, failing to detect the lowest blur. The possible reason for this is discussed above, in Section 5.2.1.

The results of the MFF model on the sharp video sequences, Daylight and Darkness, are much lower than the Pix2Pix variants, leading to false alarms. The images that are missclassified in Daylight, compared to the other images in the sequence, all exhibit more of the gray and slightly rainy sky. The gray sky, compared to a blue one on a sunny day, could be mistaken for blurriness by the model, when being a large constituent of an image. This could have been an indication that the model has a problem with large homogeneous surfaces, but then a lower accuracy would be expected on the Wall videos, in Table 4.6, which is not the case. The assumption is therefore limited to surfaces perceived as slightly blurry, such as a rainy and cloudy sky. If this indeed is the problem, the model has a serious flaw of not being flexible for blur detection in a wider range of weather conditions.

As mentioned previously, multi-scale feature fusion has been applied to many different network approaches within the task of out-of-focus blur detection. The multi-scale structure helps the network see features on various scales in the image and capture more structural information, see Section 2.2.2. However, these results show that other components are more important in this classification task, for achieving a generalized and robust classifier. If blur in surveillance images was a more local feature, the multi scale structure might have been a more important part of the network.

Overall, the generalization ability of MFF is not as good as hoped for and the model does not have as robust performance as some of the other models.

5.2.3 Performance of Pix2Pix

Three different variants of the Pix2Pix were tested in the Initial test; Pix2Pix with a regular 3-channel input (henceforth Pix2Pix RGB), the same input shape with an augmentation pre-processing step (henceforth augmented Pix2Pix) and a model with grayscale input (henceforth grayscale Pix2Pix). The augmentation was added to increase the variety in the data, as this should improve the generalization abilities of the model. The grayscale format was tested in order to investigate whether this would make a more robust model as a result of a more defined focus on low level structure (edges) in the training.

All three variants demonstrate a certain robustness in the Initial testing, as all missclassifications are limited to only nearby classes, as seen in the confusion matrices in Figures 4.2–4.4. In the Initial test, the Pix2Pix RGB got the overall highest accuracy among the models. Pix2Pix RGB achieved an accuracy of 98.04%, augmented Pix2Pix got 97.97% and the model with grayscale input got a 97.19% accuracy. When comparing the different results from the crucial division, grayscale Pix2Pix has the least false negatives (0) and Pix2Pix RGB has the least false positives (1). With the augmented data, more epochs were needed in the training until the model converged. The best weights for augmented Pix2Pix were taken from epoch 479 in comparison to 251 and 298 for the other two variants. This could be explained by the larger variety in the data.

The Purpose testing shows that the augmented model gets the best results among the Pix2Pix variants and the other models, see confusion matrices in Figure 4.6. Only

in the resulting recall is it outperformed by MANN which otherwise shows a lower overall performance. All Pix2Pix variants gave a precision of 100%, meaning no false positives (false alarms).

For the video sequence testing, the grayscale Pix2Pix variant was excluded due to lesser results in the previous tests, compared to the other two variants. All models performed pretty equally on the Q-Realistic video sequence, including Pix2Pix RGB and augmented Pix2Pix. As explained above, testing on the Q-Realistic most probably gives a lower accuracy due to the higher image resolution. On M-Realistic, the augmented Pix2Pix gets a much higher accuracy. The images that Pix2Pix RGB fails to detect are mostly low blur images taken outside, both the zoomed-in and the full scene.

Augmented Pix2Pix succeeds at detecting the low blur sequences that Pix2Pix RGB misses in the "Small window" and "Road" parts of Figure 4.7. Pix2Pix RGB on the other hand, manages to detect one particular low-blur sequence in the "Large window" section, that augmented Pix2Pix fails on. Based on these events it is hard to draw a conclusion that any of the models are better at a specific blur level or scenery. As described multiple times in this report, the resulting classification capability of a model is entirely dependant on the amount and variety of data that is used for training. These two models are trained on exactly the same training dataset, with the exception of augmented Pix2Pix using cropped images in arbitrarily chosen batches. This would make it accustomed to zoomed in images with a smaller amount of included detail, as the edges of the image are cut off. Now, as described in Section 3.1.4, the four islands at 1 on the y-axis in Figure 4.7, are different degrees of optical blur, at different zoom settings. It is notable that the low-blur sequences that augmented Pix2Pix succeeded to detect and Pix2Pix failed to, are all low blur images in zoomed in mode. This result suggests that the augmented training did in fact make the augmented Pix2Pix model better at classifying zoomed in scenes from varifocal cameras.

Both of the variants have a high performance on the Daylight video, with a 100% accuracy. This was expected since no false positives were exhibited in any of the previous tests. Darkness is the only test made where the Pix2Pix variants show false alarms. Still, the false alarm rate is less than that of the other videos, wrongly detecting a total of 2 and 3 images out of 423.

The Pix2Pix network contains several batch normalization layers. The normalization of the activation values within the network structure might have helped the network not to overfit, and therefore increased the general accuracy. An attempt was made to include batch normalization layers in the MANN and MFF models as well, but this only had a negative effect on their performance.

5.2.4 Performance of MANN

The MANN network, as described in the paper [45], was originally too small for the training dataset and was therefore altered to fit our application, described in Section 3.2.3. More layers were added to increase the learning capacity of the model, which raised the accuracy in the Initial test to 91.30% before stagnation. Further layer

addition after this point did increase the validation accuracy slightly, but at the cost of overfitting. This leaves the MANN model at the lowest Initial test performance out of all of the models. In the original paper [45], it is unclear which image resolution is used and how low blur the algorithm is expected to detect. Their dataset is not publicly available and could therefore not be compared to our results. Based on visual assessment only, the example images in the article suggest a lower resolution. It is however not specified what the input size of the model is or whether the example images have been resized before presentation, which would decrease the image quality. The assumption of lower image quality, in comparison to the ones acquired by us using Axis cameras, is strengthened by the not-so-recent release date of this article in 2016. The model presented in the article is, however, tested on surveillance footage from 30 days, achieving detection accuracies between 95-100% per day. The model is also trained on a number of different anomalies, which should indicate a high learning capacity. Despite this, the model as suggested in the article is not fit for detecting the low amount of blur that our task requires. The insufficient model size might also be due to our strategy of building a 9-class model with small differences between the classes, as opposed to a simple binary classification of blur/sharp.

One key part of the MANN architecture that makes it stand out from the other models is the use of Max pooling layers. The idea of Max pooling, described in Section 2.1, is to reduce spatial invariance, i.e. indifference to rotation, translation, enlargement and reflection. This property, although useful, might not be entirely relevant to our task, given the nature of the images. Blur in surveillance images is usually a global feature, due to a large depth of field, as mentioned in Section 3.1.1. This means that there are rarely objects in complete focus, in contrast to a blurry background, or vice versa. When training a CNN on this type of images, the spatial invariance might be a given feature due to the obvious and relatively even spread of blurriness. Another factor is a large variance in the images, resulting in no pattern whatsoever regarding the placement of objects. In this context, the spatial invariance earned by applying Max pooling might be redundant to our process. On the other hand, an effect of Max pooling in the model is a size reduction of the feature maps. This reduction might be what is causing the models tendency to overfit, making it easier to memorize the specific training images instead of saving the features in undersized feature maps.

Overall, MANN exhibits a lower performance than the other models in both Initial testing, Purpose testing and Video Sequence testing. Looking at the video sequences Daylight and Darkness, it is clear that the generalization abilities of the model are not as good as the Pix2Pix variants and that it is prone to false alarms.

5.2.5 Further discussion of the models

Corner cases

All models displayed equal performance on the corner cases, see Table 4.6. The performance on the Fisheye images, as well as both of the Wall video sequences, showcased perfect accuracy of 100%. Since fisheye images were not included in the training dataset, the fact that the model achieved high accuracy on the fisheye test, is a sign of

good generalization abilities. The concern in the M and Q-Wall video testing was that the lack of detail would lead to poor results. However, this was not the case, meaning that very few details are often enough to make a correct prediction. The final corner case with IR images was not as successful, leading to a large number of false alarms. This was expected, as IR images have very different properties than RGB images used in the training. An idea was originally that grayscale input could make a better transition into IR images. This did however not prove successful, as this model had an accuracy of 0% on the IR Video sequence test, see Section 4.4, as well as lower accuracy on the Initial and Purpose testing, see Tables 4.1 and 4.2.

Comparison with related works

Exact implementation of a CNN according to instructions from articles [23] [44] [45], does not guarantee the same testing success. The dataset that a model is trained and tested on is crucial and absolutely decisive for the model performance and accuracy result. As the used datasets in the articles aren't public or particularly specified in some cases, a direct comparison cannot be made. Another major difference, making our models hard to compare to the originals in the paper, is the change of approach from binary to multi-class classification. The classes in our models have small differences in blur and the training is done on large and high resolution images. In most related works, there is also no mention of a threshold between the binary classes, or description of the amount of different scenes and cameras, which has proven essential for the success of our models. All these factors combined reduce the possibility of a fair comparison of model performances.

Training process and accuracy fluctuations

Every model has a high fluctuation in the accuracy and loss of the training and validation data throughout the training process, as seen in Figures 4.1–4.5. To decrease these fluctuations, a few available techniques have been attempted. Initially, the learning rate of the optimizer was thought to be a part of the problem, even though an optimizer with an adaptive learning rate was applied(ADAM). However, lowering the initial learning rate did not reduce the fluctuations. Furthermore, batch normalization and different regularization techniques like dropout and early stopping were tested, but did not result in increased stability during training. One potential method that should have a high impact on the issue is increasing the batch size in the training dataset. However, increasing the batch size also increases the weight of the computations, which did not prove possible on our current system setup.

The issue of high validation accuracy fluctuations is handled by saving all models before training and applying the best weights after. This is done instead of the traditional process of saving the model after training, upon which the last weights and supposedly the best are saved along with the model.

Panorama images

As previously discussed, larger-sized images can be a problem in our approach due to the downsampling. Sufficient cropping is needed while also taking the details in the image into account. An extreme case of this is images in panorama format, where multiple cameras are stitched into one image. For this format, we therefore recommend splitting the panorama frame into several images, similar in size to the training images. One option is to then run each of those images through the network.

5.2.6 Conclusion of the Models

To summarize, the strength of the Pix2Pix model in comparison to the other models, is its high precision rates. This means that overall, very few false alarms are triggered. The augmented Pix2Pix variant showed superior results to the other variants and models, which can be traced back to the added variety in the training data. The downside to the augmented Pix2Pix network is the large size, unsuitable to run on edge. This is explored further with pruning and quantization of the augmented Pix2Pix, which will be discussed later.

5.3 Performance of DFT

The DFT approach performs well on all Video sequences tests except for the one taken during the night. The method does in fact get the best results on Daylight, Q- and M-Realistic. Particularly high accuracy is achieved on the Q-Realistic video in comparison to the deep learning models. The reasoning behind this is that the DFT is based on a comparison between two consecutive images. This means that the relative high frequency change between two images is not impacted by the 8MP Q images having a larger image size and higher resolution, which is the problem for the learning-based models. However, the DFT approach does not perform as well on the video sequence Darkness, taken in low light. The poor performance on dark images can be explained by previously known limitations of the traditional method. At night, the headlights of the cars are brighter, and these point sources of direct light are a known problem [38]. Furthermore, during darkness, the high frequency content in an image is overall reduced, which makes the difference between two frames smaller and increases the influence of other small changes in the scene. If the DFT method was to be applied only to dark and dim-lit videos, the value of Max_{HF} could be lowered thus increasing the threshold and the sensitivity to the amount of detail in the image. However, changing the threshold dynamically to suit the conditions is a difficult task and there are no examples of this for the purpose of blur detection found in research.

An overall advantage of the DFT method is that the data acquisition stage, that is so important for learning-based methods, is completely unnecessary in this traditional method. However, an important disadvantage that this imposes on the DFT method in comparison to CNN methods is the lack of improvement possibilities. The threshold and standard deviation in the filtering can be tweaked for specific surroundings, but

the sensitivity to sudden light changes, direct light, darkness and occlusion still remain. The surroundings can also be changed between consecutive frames by the camera itself, in case of a varifocal lens zooming in or a movement by a PTZ camera (pan/tilt).

Another disadvantage of the DFT is that it is computationally heavy, involving both a high-pass filtration and a Fourier transformation of the image. It also uses more memory than the other method since the previous image values have to be saved in order to be compared to the current frame. The average calculation time of our DFT method, i.e. the time it takes to process one image and make a comparison between two frequency contents, is 9-13 times larger than the inference time of the learning-based models when running on GPU. The method of measuring this time is, however, highly dependant on the specific implementation of the DFT approach that we have done, and could therefore possibly be improved with more efficient coding. Nonetheless, it is notable that the average time is dramatically larger than those of the other models, and worth considering in further development and application of a method for blur-detection.

Additional steps in the method, such as the removal of moving pixel blocks, applied in the original paper [38], would make the consecutive frames more alike. This could possible reduce the effect of some of the known problem areas, such as occlusion and motion blur, at the cost of increasing the calculation time and computation requirements. In the testing, however, the results showed very little indication of the mentioned problem areas having a negative effect on the performance. This could be due to the images being taken so close in time to each other (1 fps) and objects in the frame being far away from the camera lens.

It might be an idea to make use of both methods, traditional and learning, in order to benefit from both strengths. The DFT, which is computationally heavy could e.g. be used to validate that the camera is indeed out of focus, in the event of detection by a learning-based model. This would, however, require saving previous frames or the frequency information about them, in order to have something to compare to. Similarly, the DFT model weaknesses would have to be taken into account, by perhaps not applying it in low light etc. This would also condition the running of the entire combination model, as the light conditions would have to be actively tracked for successful implementation.

5.3.1 Conclusion of the DFT

The DFT method shows good performance on most of our tests, except for the video in low light. Furthermore, the sensitivity of the DFT model is good, showing high accuracy even on low blur. However, the major drawback of the DFT is the dependence on a fixed threshold, which makes it unpredictable to known weaknesses, such as dim-light, direct light sources and occlusion. Additionally, the dependence on the threshold reduces the improvement possibilities of the method, making it inflexible for broad use. Its computation weight should also be considered in the future, if the method was to be applied.

5.4 Model Size

The models have very different sizes and number of parameters due to the amount, size of and type of layers in the networks. We can see how this also influences the results. In general, the larger the model, the better the result – given that the over-fitting can be controlled. Model size also influences inference time. In Table 4.8 the two smaller models, MANN and MFF, have a lower inference time than the Pix2Pix variants. Interestingly, the pruned augmented Pix2Pix gets the same average inference time as the non-pruned variant. However, with further pruning, the inference time would be reduced. Worth noting is also that the times will change when run on edge, both because of the TPU performance and as a result of the quantization. The relative order of inference between the methods, should however, remain unchanged.

5.4.1 Pruning

As the largest and most accomplished model in regard to high accuracy rates and low FAR, the augmented Pix2Pix was chosen to be further processed by pruning. The pruning reduces the model size in order to make it more suitable for running on edge, in terms of energy cost requirements and inference time.

By applying pruning and quantization, the model was reduced to 20% of the original size, while only reducing the accuracy on the Initial test dataset from 97.97% to 97.78%. When comparing the confusion matrix of the pruned augmented Pix2Pix to the unpruned, the pruned model has an increased number of false negatives in the crucial division. However, the pruned model also gets less confident in the upper classes, misclassifying images from b_3 into b_4, and b_5 into b_7. In between sharp* and b_0.5, the prediction does however get a bit better, classifying 3 more images correctly. This means that the pruning does not reduce the accuracy on a specific class, but on multiple places in the network output. It also shows that the predictions still follow the pattern of being misclassified into neighboring classes, and not too far off from the true label. This is a good sign, as it means that a certain robustness of the model is preserved throughout the pruning.

Compared to the MANN model, which is the smallest model tested, the pruned Pix2Pix has both a higher accuracy and a smaller size. This confirms that reducing parameters by pruning is better than training a smaller model from scratch. If future application should require the model to be even smaller in order to meet the requirements of energy cost and inference time, it is recommended to increase the pruning rather than train a smaller network. The risk of substantially lowering the accuracy upon excessive pruning should, however, be considered in such a case.

One possible future improvement is to evaluate different pruning methods. The method of pruning has since the development of CNNs risen in popularity and increased the variety of implementation techniques. There are many new suggestions [48] on how to reduce the extreme number of parameters in large networks, which could be of great significance to the large augmented Pix2Pix network, evaluated in this thesis.

5.5 Limitations

An primary expected limitation is unseen weather and possibly climates. Such conditions that have been identified are fog, snow, images containing water surfaces and heavy rain. Snow and water surfaces have a tendency to reflect the light in a way that is not common elsewhere. Fog and heavy rain introduce a natural haze to the image, which the models haven't been trained on. Training the models on these conditions would impose higher requirements on the artificial blur used to simulate focus loss, which might have to be more similar to the PSF blurring of the lens.

Another problem could possibly be dirt or spray paint on the lens, triggering false alarms. Depending on the application this might not be terrible since it can be seen as a tampering of the image. In the future, these and possibly other camera tampering/problems could be added into the model making it able to differentiate between them.

Generally, the more variety in the training scenes for the model, the better generalization will be obtained. Gathering more data increases the models' performance, and is needed for future development.

5.6 Future Work

A future development of our work could be to include IR images in the training dataset. As mentioned in Section 5.2.5, the models cannot currently handle IR input. Nonetheless, it would be of interest to expand the model to include IR images in the future, since a lot of modern surveillance cameras have IR capabilities, for use in low light conditions. One possible way to do this could be with an ensemble model. As before, the implementation would require a large dataset with a variety of scenes and weather conditions. It would also be important to keep the dark and low-light images in the training dataset in order to combat the transition into IR, and assure compatibility with cameras without IR technology.

This thesis has focused solely on the implementation and evaluation of blur detection models, and not their application and integration in Axis cameras. It is, however, still interesting to discuss some aspects of a possible future application. For the model to be deployed into a camera, it should be packaged into a more lightweight application, such as an AXIS Camera Application Platform (ACAP). How frequently a prediction is made, among other things, is decided in the ACAP. One thing to take into consideration might be how a temporary blur detection should be handled. To reduce the risk of false alarms, we recommend that a careful approach is deployed in such cases. A single or very temporary blur classification can be caused by a sudden light change or temporary occlusion. If frames are no longer classified as blurry after a short period of time, the cause of the classification might be removed, and the event should be disregarded. Furthermore, in the future, the model could be integrated into the camera system, so that an alarm could trigger an automatic refocus.

In order to deploy the model into Axis cameras, there are a number of requirements and concerns that are yet to be addressed and considered. One step towards that goal has already been made, since the quantization of the models makes a type conversion from float32 to int8, which is required to run on Edge TPU equipped cameras. Further consideration should also be taken in regards to the size of the model, depending on the result of an initial camera application. Additionally, solving the described limitations is recommended, as these are related to fairly common situations, and therefore essential in a camera application.

Finally, additional testing should be conducted until a very high and satisfactory robustness is achieved. Skipping this step can be costly, due to the risk of many false alarms that could appear in untested conditions. This is especially valid if the detection system was to be utilized for other purposes as well. For example, we know that there are certain factors that lead to the issue of focus loss, but there might be unknown reasons that could be identified upon frequent alarms in similar situations. Frequent alarms could also indicate an error in the camera, which would allow for a quick discovery. The detection system could also be put to use for camera optimization, in conditions that often lead to a loss of focus. One such case is heat, in which the amount of focus loss detected by a model could help optimize the camera configuration when a higher temperature is detected in the future. This way, future focus loss could possible be prevented in diagnosed cases.

6 Conclusion

We have shown that optical out-of-focus blur can be accurately detected with a CNN approach, trained on artificially blurred surveillance images. The CNN successfully translates artificially blurred images to real optical blur and is also more robust and efficient than traditional image processing methods. Introducing a multi-class approach, with gradually increasing blur, strongly improves the classification by decreasing ambiguity and making the model highly generalized across different types of cameras and sceneries. The multi-class approach also introduces the ability to adjust the sensitivity of out-of-focus detection in a CNN model. The best performing model shows a minor weakness regarding the detection of low blur in high resolution images. A few suggestions are proposed in order to reduce this vulnerability.

Out of the compared models, it has been shown that augmented Pix2Pix achieves the highest accuracy throughout the testing. Its larger size in comparison to the other models is combated with pruning, making it sufficiently small for on edge application. The final model is more prone to missing events of low-blur, rather than producing false alarms. The high accuracy, combined with the very low false alarm rate, makes the model a solid blur detection algorithm, with good prospects for on edge camera application.

The DFT method displays good sensitivity to changing focus and performs well in our conducted tests. However, we cannot dismiss its major drawback of depending on a fixed threshold, making it unreliable when a low false detection rate is of the greatest importance. The inherent problem with the DFT approach is a sensitivity to cases such as dim light and direct light sources, which remains even with an adequate threshold.

Utilizing the power of CNNs for this application area has major potential and a lot more can be explored in the future.

Bibliography

[1] Mengqiu Zhu, Lingjie Yu, Zongbiao Wang, Zhenxia Ke and Chao Zhi. *Review: A Survey on Objective Evaluation of Image Sharpness*. 2023. DOI: 10.3390/app13042652. URL: https://doi.org/10.3390/app13042652.

[2] Sayyed Mohammad Hosseini and Amir Hossein Taherinia. "Anomaly and tampering detection of cameras by providing details". In: *2016 6th International Conference on Computer and Knowledge Engineering (ICCKE)*. 2016, pp. 165–170. DOI: 10.1109/ICCKE.2016.7802134.

[3] Puren Guler, Deniz Emeksiz, Alptekin Temizel, Mustafa Teke and Tugba Taskaya Temizel. "Real-time multi-camera video analytics system on GPU". In: *Journal of Real-Time Image Processing* 11.3 (2016), pp. 457–472. ISSN: 1861-8219. DOI: 10.1007/s11554-013-0337-2. URL: https://doi.org/10.1007/s11554-013-0337-2.

[4] Theodore Tsesmelis, Lars Christensen, Preben Fihl and Thomas B. Moeslund. "Tamper detection for active surveillance systems". In: *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*. 2013, pp. 57–62. DOI: 10.1109/AVSS.2013.6636616.

[5] Roxanne A Pagaduan, Ma Christina R Aragon and Ruji P Medina. "iblurdetect: Image blur detection techniques assessment and evaluation study". In: *Proceedings of the International Conference on Culture Heritage, Education, Sustainable Tourism, and Innovation Technologies-CESIT*. 2021, pp. 286–291.

[6] Karan Khajuria, Kapil Mehrotra and Manish Kumar Gupta. "Blur Detection in Identity Images Using Convolutional Neural Network". In: *2019 Fifth International Conference on Image Information Processing (ICIIP)*. 2019, pp. 332–337. DOI: 10.1109/ICIIP47207.2019.8985888.

[7] Vijaya Shetty S, Madhumitha R, Mekala Meghana Reddy, Shreya Shettar and Tejashree Krishna Murthy. "Automated Identity Document Recognition and Classification (AIDRAC)-A Review". In: *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*. 2022, pp. 674–681. DOI: 10.1109/ICAISS55157.2022.10011056.

[8] Yikun Pan, Sik-Ho Tsang, Yui-Lam Chan and Daniel P.K. Lun. "Blur Detection for Surveillance Camera System". In: *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 2022, pp. 1879–1884. DOI: 10.23919/APSIPAASC55919.2022.9980343.

[9] Ayush Senapati, Atul Kumar Karn, Yash Bhardwaj, Jenicka. S, Padma Priya R, Ankit Shukla and Shivam Arora. "Identification of blurred objects in real time Video using deep learning neural networks". In: *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. 2022, pp. 1–4. DOI: 10.1109/ICCCNT54827.2022.9984429.

[10] Tomasz Szandała. "Convolutional Neural Network for Blur Images Detection as an Alternative for Laplacian Method". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020, pp. 2901–2904. DOI: 10.1109/SSCI47803.2020.9308594.

[11] Anas Nafis Almustofa, Yudhistira Nugraha, Andi Sulasikin, Irfan Dwiki Bhaswara and Juan Intan Kanggrawan. "Exploration of Image Blur Detection Methods on Globally Blur Images". In: *2022 10th International Conference on Information and Communication Technology (ICoICT)*. 2022, pp. 275–280. DOI: 10.1109/ICoICT55009.2022.9914850.

[12] Jinxing Li, Beicheng Liang, Xiangwei Lu, Mu Li, Guangming Lu and Yong Xu. "From Global to Local: Multi-Patch and Multi-Scale Contrastive Similarity Learning for Unsupervised Defocus Blur Detection". In: *IEEE Transactions on Image Processing* 32 (2023), pp. 1158–1169. DOI: 10.1109/TIP.2023.3240856.

[13] Chang Tang, Xinwang Liu, Xiao Zheng, Wanqing Li, Jian Xiong, Lizhe Wang, Albert Y. Zomaya and Antonella Longo. "DeFusionNET: Defocus Blur Detection via Recurrently Fusing and Refining Discriminative Multi-Scale Deep Features". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.2 (2022), pp. 955–968. DOI: 10.1109/TPAMI.2020.3014629.

[14] Z. Jiang, X. Xu, L. Zhang, C. Zhang, C.S. Foo and C. Zhu. "MA-GANet: A Multi-Attention Generative Adversarial Network for Defocus Blur Detection." In: *IEEE Transactions on Image Processing, Image Processing, IEEE Transactions on, IEEE Trans. on Image Process* 31 (2022), pp. 3494 –3508. ISSN: 1057-7149. URL: https://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edseee&AN=edseee.9771138&site=eds-live&scope=site.

[15] Xuewei Wang, Shulin Zhang, Xiao Liang, Hongjun Zhou, Jinjin Zheng and Mingzhai Sun. "Accurate and Fast Blur Detection Using a Pyramid M-Shaped Deep Neural Network". In: *IEEE Access* 7 (2019), pp. 86611–86624. DOI: 10.1109/ACCESS.2019.2926747.

[16] Jinsun Park, Yu-Wing Tai, Donghyeon Cho and In So Kweon. "A Unified Approach of Multi-scale Deep and Hand-crafted Features for Defocus Estimation". In: *CoRR* abs/1704.08992 (2017). arXiv: 1704.08992. URL: http://arxiv.org/abs/1704.08992.

[17] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[18] Arohan Ajit, Koustav Acharya and Abhishek Samanta. "A review of convolutional neural networks". In: *2020 international conference on emerging trends in information technology and engineering (ic-ETITE)*. IEEE. 2020, pp. 1–5.

[19] Nils Bjorck, Carla P Gomes, Bart Selman and Kilian Q Weinberger. "Understanding batch normalization". In: *Advances in neural information processing systems* 31 (2018).

[20] Elizar Elizar, Mohd Asyraf Zulkifley, Rusdha Muharar, Mohd Hairi Mohd Zaman and Seri Mastura Mustaza. "A Review on Multiscale-Deep-Learning Applications". In: *Sensors* 22.19 (2022). ISSN: 1424-8220. DOI: 10.3390/s22197384. URL: https://www.mdpi.com/1424-8220/22/19/7384.

[21] M. Sadegh Riazi, Bita Darvish Rouani and Farinaz Koushanfar. "Deep Learning on Private Data". In: *IEEE Security Privacy* 17.6 (2019), pp. 54–63. DOI: 10.1109/MSEC.2019.2935666.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[23] Fanglin Chen, Weihang Wang, Huiyuan Yang, Wenjie Pei and Guangming Lu. "Multiscale feature fusion for surveillance video diagnosis". In: *Knowledge-Based Systems* 240 (2022), p. 108103. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.108103. URL: https://www.sciencedirect.com/science/article/pii/S0950705121011655.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV].

[25] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[26] Yingjie Tian, Duo Su, Stanislao Lauria and Xiaohui Liu. "Recent advances on loss functions in deep learning for computer vision". In: *Neurocomputing* (2022).

[27] Lutz Prechelt. "Early Stopping — But When?" In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_5. URL: https://doi.org/10.1007/978-3-642-35289-8_5.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from Metrics for Multi-Class Classification". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[29] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang and Trevor Darrell. "Rethinking the value of network pruning". In: *arXiv preprint arXiv:1810.05270* (2018).

[30] Subutai Ahmad and Luiz Scheinkman. "How Can We Be So Dense? The Benefits of Using Highly Sparse Representations". In: *CoRR* abs/1903.11257 (2019). arXiv: 1903.11257. URL: http://arxiv.org/abs/1903.11257.

[31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[32] Xinxia Fan, Yanhua Yang, Cheng Deng, Jie Xu and Xinbo Gao. "Compressed multi-scale feature fusion network for single image super-resolution". In: *Signal Processing* 146 (2018), pp. 50–60. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2017.12.017. URL: https://www.sciencedirect.com/science/article/pii/S0165168417304309.

[33] Ricardo Ribani and Mauricio Marengoni. "A Survey of Transfer Learning for Convolutional Neural Networks". In: *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*. 2019, pp. 47–57. DOI: 10.1109/SIBGRAPI-T.2019.00010.

[34] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1 (2019), pp. 1–48.

[35] Chao Wang, Juan Chen, Hongguang Jia, Baosong Shi, Ruifei Zhu, Qun Wei, Linyao Yu and Mingda Ge. "Parameterized Modeling of Spatially Varying PSF for Lens Aberration and Defocus." In: *Journal of the Optical Society of Korea* 19.2 (2015), pp. 136 –143. ISSN: 12264776. URL: https://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edo&AN=ejs36083405&site=eds-live&scope=site.

[36] Mohammed H. Rasheed, Omar M. Salih, Mohammed M. Siddeq and Marcos A. Rodrigues. "Image compression based on 2D Discrete Fourier Transform and matrix minimization algorithm". In: *Array* 6 (2020), p. 100024. ISSN: 2590-0056. DOI: https://doi.org/10.1016/j.array.2020.100024. URL: https://www.sciencedirect.com/science/article/pii/S2590005620300096.

[37] Eric Jacobsen and Richard Lyons. "The sliding DFT". In: *IEEE Signal Processing Magazine* 20.2 (2003), pp. 74–80.

[38] Ali Saglam and Alptekin Temizel. "Real-Time Adaptive Camera Tamper Detection for Video Surveillance". In: *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*. 2009, pp. 430–435. DOI: 10.1109/AVSS.2009.29.

[39] Ajay Kulkarni, Deri Chong and Feras A. Batarseh. "5 - Foundations of data imbalance and solutions for a data democracy". In: *Data Democracy*. Ed. by Feras A. Batarseh and Ruixin Yang. Academic Press, 2020, pp. 83–106. ISBN: 978-0-12-818366-3. DOI: https://doi.org/10.1016/B978-0-12-818366-3.00005-8. URL: https://www.sciencedirect.com/science/article/pii/B9780128183663000058.

[40] Ioannis Markoulidakis, Ioannis Rallis, Ioannis Georgoulas, George Kopsiaftis, Anastasios Doulamis and Nikolaos Doulamis. "Multiclass confusion matrix reduction method and its application on NET promoter score classification problem". In: *Technologies* 9.4 (2021), p. 81. DOI: 10.3390/technologies9040081.

[41] Margherita Grandini, Enrico Bagli and Giorgio Visani. *Metrics for Multi-Class Classification: an Overview*. 2020. arXiv: 2008.05756 [stat.ML].

[42] Ayodele Lasisi, Rozaida Ghazali and Tutut Herawan. "Chapter 11 - Application of Real-Valued Negative Selection Algorithm to Improve Medical Diagnosis". In: *Applied Computing in Medicine and Health*. Ed. by Dhiya Al-Jumeily, Abir Hussain, Conor Mallucci and Carol Oliver. Emerging Topics in Computer Science and Applied Computing. Boston: Morgan Kaufmann, 2016, pp. 231–243. ISBN: 978-0-12-803468-2. DOI: https://doi.org/10.1016/B978-0-12-803468-2.00011-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128034682000114.

[43] Matthew Trentacoste, Rafal Mantiuk and Wolfgang Heidrich. "Blur-aware image downsampling". In: *Computer graphics forum*. Vol. 30. 2. Wiley Online Library. 2011, pp. 573–582.

[44] Vahida Attar, Rucha Sathe, Yash Shah and Dhruv Kudale. "Single Image Blind Deblurring". In: *2021 6th International Conference for Convergence in Technology (I2CT)*. 2021, pp. 1–9. DOI: 10.1109/I2CT51068.2021.9417948.

[45] Lingping Dong, Yongliang Zhang, Conglin Wen and Hongtao Wu. "Camera anomaly detection based on morphological analysis and deep learning". In: *2016 IEEE International Conference on Digital Signal Processing (DSP)*. 2016, pp. 266–270. DOI: 10.1109/ICDSP.2016.7868559.

[46] Gil-beom Lee, Myeong-jin Lee and Jongtae Lim. *Unified Camera Tamper Detection Based on Edge and Object Information*. 2015. DOI: 10.3390/s150510315. URL: https://doi.org/10.3390/s150510315.

[47] *Blur filters*. URL: https://helpx.adobe.com/photoshop-elements/using/blur-filters.html.

[48] Hao Zhou, Jose M Alvarez and Fatih Porikli. "Less is more: Towards compact cnns". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer. 2016, pp. 662–677.