

ESTIMATING INFRARED-TO-VISIBLE LIGHT RATIO IN IMAGES USING GRADIENT-BOOSTED DECISION TREES AND MULTILAYER PERCEPTRONS

AXEL AHLIN

Master's thesis
2025:E10



LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Estimating Infrared-to-Visible Light Ratio in Images Using Gradient-Boosted Decision Trees and Multilayer Perceptrons

Axel Ahlin



LUND
UNIVERSITY

Department of Mathematics

Abstract

This thesis investigates using machine learning for estimating the infrared-to-visible light ratio (q_{IR}) in images, which is used for day/night synchronization in surveillance cameras. Three different models were developed, namely a regression multilayer perceptron (RMLP), a regression-by-classification MLP (RbC MLP), as well as a gradient-boosted decision tree (GBDT). Using an internal dataset collected in both laboratory and real environments, the MLP models had the best performance, with both the RMLP (RMSE = 0.071, $R^2 = 0.883$) and the RbC MLP (81.2% accuracy) showing good results. The GBDT model showed limited capabilities of generalizing to unseen data (RMSE = 0.137, $R^2 = 0.592$). An ablation study underscored the importance of using both data augmentation and regularization, which improved the overall accuracy of the RbC MLP by approximately 48 percentage points. Overall, the results demonstrate the potential of using multilayer perceptrons for spectral estimation in surveillance cameras.

Acknowledgements

First, I would like to thank Axis Communications for providing me with the opportunity of writing my thesis there. I want to thank Robin Eriksson for his technical knowledge, good vibes, and fika. I also want to thank people around the Fixed Camera department for their helpfulness and technical knowledge. I also want to thank Ludvig and Viktor for their guidance, expertise and well-structured feedback throughout the entirety of this thesis.

Contents

1. Introduction	11
1.1 Background	11
1.2 Problem	11
1.3 Aim of research	11
1.4 Related work	12
1.5 Delimitations	12
1.6 Report outline	12
2. Theory	13
2.1 Light and the electromagnetic spectrum	13
2.2 Digital imaging	14
2.3 Color theory	15
2.4 Foundations of machine learning	15
3. Methodology	20
3.1 Data collection	20
3.2 Data preprocessing	22
3.3 Data augmentation	24
3.4 Model architectures	24
3.5 Model evaluation	26
3.6 Ablation study	28
4. Results & Analysis	29
4.1 Model results	29
4.2 Comparing regression-by-classification and regression models	35
4.3 Ablation study	36
4.4 Challenges and limitations	37
5. Conclusions & Future Work	39
5.1 Conclusions	39
5.2 Future work	39
References	41

1

Introduction

1.1 Background

Surveillance cameras have become an important part of modern society. One of the companies manufacturing such cameras is Axis Communications. These devices are deployed in various environments, such as public spaces, traffic monitoring, and other parts of infrastructure. A surveillance camera is expected to handle different lighting conditions and automatically adapt its camera settings to its environment. Standard industry practice for 24-hour outdoor imaging involves using a camera paired with near-infrared (NIR) illumination at night [1]. In extension, this means that a surveillance camera needs to have the ability to infer whether it is daytime or nighttime from the characteristics of the light reaching the image sensor. One possible approach is to exploit the fact that cameras can register both infrared and visible light [2]. One appropriate measure in this context is the ratio of infrared light to the total amount of light, $q_{IR} = \frac{\text{NIR light}}{\text{NIR light} + \text{visible light}}$, $q_{IR} \in [0, 1]$. The reasoning is that if a camera scene is dominated by infrared light, $q_{IR} \rightarrow 1$, it means that there is hardly any visible light present and it can be assumed that it is nighttime in the scene. The camera should then be in night mode. When the amount of visible light increases (e.g during sunrise), q_{IR} will generally decrease until it reaches a threshold for switching to day mode.

In addition to visible light, surveillance cameras register NIR light, which the human eye cannot see. To deal with this difference, the cameras use an IR-cut filter to separate visible light from the invisible NIR light, which otherwise corrupts the colors. This type of spectral information can be acquired through the use of expensive equipment such as spectrometers, though the vast majority of consumer-grade surveillance cameras are not equipped with such instruments as it would likely make them more expensive. Using a visible light sensor with the camera is another possibility, but this approach has its own drawbacks. The key issue with light sensors is that they generally have a significantly more limited field of view than the camera, which is problematic because they might miss sources of light visible to the camera. A trivial way to solve the problem in the thesis would of course be to turn the IR-cut filter on and off multiple times per night and calculate q_{IR} . However, this is unreasonable, as it would more or less render the camera visionless for a total of a few minutes every night. Another factor to consider would be the additional mechanical wear on the IR-cut filter by switching it on and off many more times than intended. In addition, simply using the time of day for day/night synchronization is unreliable as factors such as weather, artificial lighting, and geographic location can cause significant variations in the lighting of the camera scene. Switching to day mode and turning on the IR-cut filter should only happen when there is enough visible light, regardless of the current time. Due to the difficulty of accurately modeling the problem with traditional methods and heuristics, this thesis will investigate different machine learning models for predicting the q_{IR} value for a given camera scene.

1.2 Problem

The purpose of this thesis is to explore the possibilities of leveraging various machine learning architectures for estimating the infrared-to-visible ratio, q_{IR} for light registered by a camera's image sensor. Additionally, the models should be lightweight since they will be deployed on the cameras.

1.3 Aim of research

The research questions of this thesis can be stated as follows:

- Is it possible to estimate how much of the incoming light lies within the infrared spectrum using machine learning architectures?
- Which architectures are suitable for the task?

1.4 Related work

After conducting a literature study, there seems to be a research gap in the literature for estimating the infrared-to-visible ratio of an image with machine learning techniques for day/night-synchronization. In a broader context, there has been some related work on the intersection of spectral imaging and machine learning in the field of agricultural science, where it has been used for measuring vegetation health. NDVI (normalized difference vegetation index) [3] is a metric used for quantifying vegetation health by measuring the difference between near-infrared light (NIR) (which is strongly reflected by vegetation) and visible red light (VIS) (which the vegetation reflects weakly or absorbs), where $NDVI = \frac{NIR - VIS}{NIR + VIS}$. Acquiring near-infrared (NIR) and red spectral data for calculating NDVI usually relies on specialized equipment like multispectral or hyperspectral cameras [4]. However, these cameras are expensive and require technical knowledge for proper use and data interpretation. Wang et al. [4] collected reflectance data from corn plants with an IR-sensitive Raspberry Pi camera sensor and a Raspberry Pi. The data was then used for training different machine learning models, such as support vector regression, for estimating the NDVI of the plants. The models showed strong performance when evaluated against hyperspectral cameras. This suggests that it is possible to use machine learning for estimating spectral characteristics of image sensor data and by extension q_{IR} .

1.5 Delimitations

We will only consider one camera with its specific configuration of image sensor and optics. The configurations of these components will differ between cameras and as such the machine learning models developed in this thesis would likely need to be retrained or recalibrated to work as intended on other cameras.

1.6 Report outline

- **Chapter 2** provides the theory needed for understanding the work in this thesis.
- **Chapter 3** describes the methodology used concerning collection of data, preprocessing, model architectures, and evaluation metrics.
- **Chapter 4** presents the model results, along with analysis and discussion of the methodology, results, and sources of error.
- **Chapter 5** concludes the thesis and suggests directions for future work.

2

Theory

To predict q_{IR} , the proportion of infrared light in a scene, it is important to first understand the physics of light and how cameras work. First, we start with the basics of the electromagnetic spectrum, focusing on visible and near-infrared light. Second, we explain how modern camera sensors capture and process light, including the role of color filters and why specific features like the red and blue color ratios are particularly useful for this task. Finally, we present some of the fundamentals of machine learning, with a closer look at two models used in this thesis: multilayer perceptrons (MLPs) and gradient-boosted decision trees (GBDTs).

2.1 Light and the electromagnetic spectrum

Electromagnetic radiation makes up what is called the electromagnetic spectrum [2]. It is usually divided into different segments such as gamma rays, ultraviolet radiation, visible light, and infrared light [2]. Visible light constitutes a specific segment of this spectrum, ranging approximately from 400 to 700 nanometers in wavelength and corresponds to the wavelengths of light that the human eye's photoreceptor cells (*cones*) can detect and interpret [2]. When visible light enters the eye and strikes the retina, it triggers chemical reactions in the cones, which in turn leads us to perceive different colors [2]. In this thesis, we will also concern ourselves with the near-infrared region (1000 nm (upper limit for the camera) $> \lambda > 700$ nm) in addition to the region of visible light.

Light in the real world is composed of combinations of illuminants, both natural and man-made. An illuminant in this context represents a certain *spectral power distribution* [5]. A spectral power distribution represents the physical wavelength composition of an illuminant, often expressed as radiance given as a function of wavelength [6]. CIE (*The International Commission on Illumination*) has defined a wide range of *standard illuminants* [5]. In Figure 2.1, the spectral power distributions of the noon daylight standard illuminant D65 and the fluorescent standard illuminant F4 are given as examples. Note the difference in the infrared region between D65 and F4, with D65 exhibiting significant spectral intensity while F4 has almost no intensity. Considering the case of D65, this makes sense, as a large portion of the noon sun light which reaches Earth's surface and heats it is infrared [7].

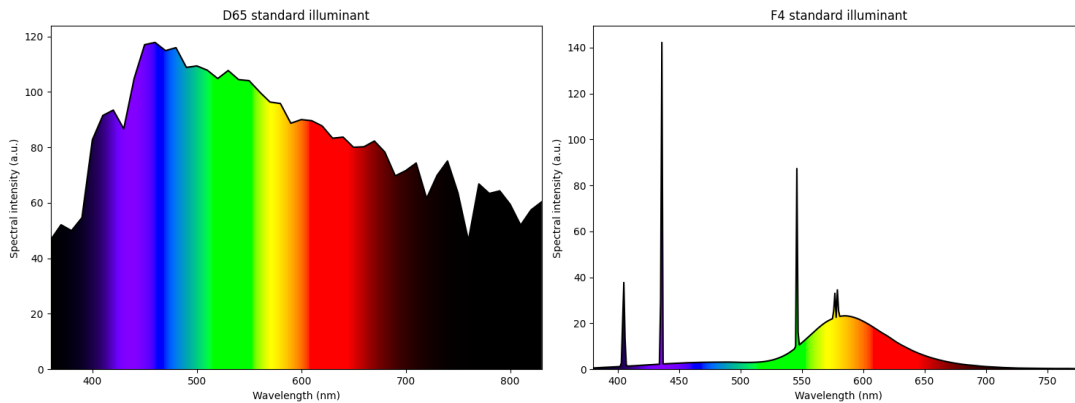


Figure 2.1 The spectral power distributions of the standard illuminants D65 and F4, generated with LuxPy [8].

2.2 Digital imaging

Digital image sensors are the heart of modern digital camera systems. They are constructed by placing a large number of photosensitive elements in an array, which convert light into electrical signals [5]. These signals can then be processed into digital images [5]. We will use a complementary metal-oxide-semiconductor (CMOS) sensor in this thesis. When the sensor has captured the light from the scene, the captured image signal must then be reconstructed back to a color image. This is done by placing a *color filter array* on top of the image sensor, which allows each pixel to be sensitive to specific wavelengths of light [5]. A common choice for filter pattern is the *Bayer mask*. As shown in Figure 2.2, the pattern is a 2×2 mosaic of one blue, one red and two green filters. The reason for this design is that the cones of the human eye are more sensitive to green light and this will allow the image sensor to give a scene representation that would be more accurate to how a human observer would interpret it [5].

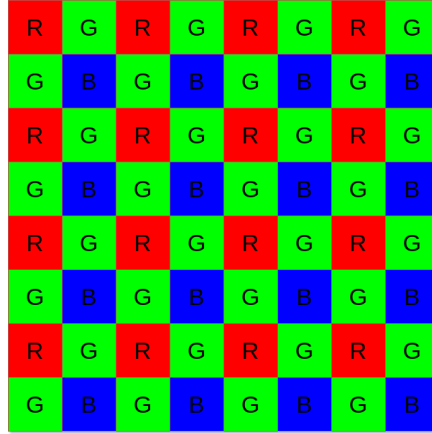


Figure 2.2 Example of a Bayer filter pattern arrangement.

Furthermore, the quantum efficiency of a sensor reflects how efficiently the sensor converts incident photons of different wavelengths into electrical charges [9], as illustrated in Figure 2.3.

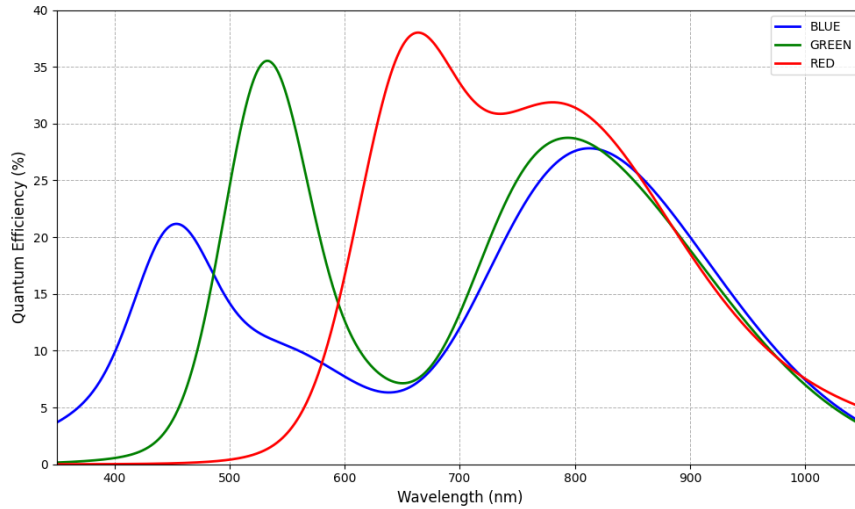


Figure 2.3 Example quantum efficiency of a CMOS Sensor. For the purposes of this thesis, the quantum efficiency of the CMOS sensor we use can be regarded as roughly equivalent to the quantum efficiency illustrated in this figure.

In the case of infrared light striking the camera sensor, the resulting image can look greyed out. For this reason, some cameras implement an infrared cut filter or IR-cut filter for short [5]. This filter is a mechanical component often used in digital cameras to improve daytime color accuracy by blocking infrared light while allowing visible light to pass through. By filtering out unwanted infrared light, the

IR-cut filter helps ensure that the camera sensor receives only the visible light spectrum, which results in more accurate color reproduction in captured images. During nighttime however, infrared light is often leveraged for surveillance purposes in low-light environments [1]. This is illustrated in Figure 2.4. A surveillance camera equipped with an NIR LED can emit infrared light invisible to the human eye and infrared light reflected by surfaces in the camera scene can be detected by the camera sensor. This ensures that outdoor scenes remain observable even during nighttime.

2.3 Color theory

Color does not exist and is merely a psychological phenomenon, which can vary greatly from observer to observer [5]. However, it is still possible to reason about it. The most common color model is RGB (Red, Green, Blue), which is a color model where colors are generated by combining varying intensities of red, green, and blue light and is widely used to create and display color in electronic displays, such as computer monitors and digital cameras [2]. By mixing these primary colors at different levels of intensity, one can produce a wide range of colors [2]. In this thesis, we will work exclusively with linear RGB values and ratios. In camera imaging, it is advantageous to perform calculations in a linear color space that adheres to the linear characteristics of physical light. Modern camera sensors behave linearly in the sense that they produce signals with amplitudes proportional to the intensity of the incoming light [10]. For example, this means that multiplying an intensity value by two would correspond to doubling the intensity of the incident light on the sensor. However, though easy to work in, linear RGB does not reflect how human observers interpret changes in light very well. To mimic how human vision would respond, imaging systems apply a non-linear transfer function called *gamma correction*, which essentially is a power function

$$\mathcal{T}(l) = l_{\max} \left(\frac{l}{l_{\max}} \right)^\gamma \quad (2.1)$$

where l is the intensity level, $l_{\max}$ is the maximum intensity level of the image and γ is the correction factor [11]. $\gamma > 1$ increases the darkness of the pixels and $\gamma < 1$ increases the brightness [11]. Fortunately, we can disregard any processing such as gamma correction as those will be applied further down the imaging pipeline and not take that into account when preprocessing the data. In this thesis, the image sensor uses a normalized $\frac{X}{R+G+B}$ color space, $X \in \{R, G, B\}$, for its data. However, the two ratios of $\frac{R}{R+G+B}$ and $\frac{B}{R+G+B}$ are often more interesting from an imaging perspective, such as white balance tuning. The reason for this is analogous to how the human vision system works, as discussed in section 2.2: the green component will almost always dominate in the light signal. As such, one rarely has to apply gain to the green component, which is why the image sensor only presents values for $\frac{R}{R+G+B}$ and $\frac{B}{R+G+B}$.

2.4 Foundations of machine learning

In this section we present an overview of different machine learning concepts that were used in this thesis.

2.4.1 Multilayer perceptron

A multilayer perceptron (MLP), as described in [12], is a basic feed-forward neural network which consists of at least three layers: one input layer, one (or more) hidden layers and one output layer. An MLP has the goal of approximating a target function f^* . For instance, in classification, $f^*(x)$ maps an input x to its corresponding category. The network establishes a function $f(x; \theta)$, where θ represents the parameters. It then optimizes the mapping between inputs and outputs by adjusting θ through training to approximate f^* as closely as possible [12]. The layers consist of neurons, with each neuron fully connected to the neurons in the preceding and succeeding layers [13]. A neuron receives input from many other neurons and computes its own activation output value y [12]. This can be expressed as

$$y = \phi \left(\sum_{i=1}^n w_i x_i + b \right) \quad (2.2)$$

where ϕ is an activation function, $\mathbf{w} = (w_1, w_2, \dots, w_n)$ is a weight vector, $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is an input vector of activations to be weighted, and b is a bias term. The purpose of an activation function is to



(a) IR-cut filter on.



(b) IR-cut filter off.

Figure 2.4 This montage shows the impact of an IR-cut filter on the color reproduction of the camera's image sensor. With the filter on (a), the colors of the image appear more natural to the human eye. With the filter off (b), the colors of the scene are washed out and magenta-tinted due to interference in the Bayer filters from the NIR light.

introduce nonlinearity in the layers of the model, which otherwise would be just a combination of linear transformations and as such unable to learn more complex functions [13]. A popular activation function is the rectified linear unit (ReLU), $\phi(x) = \max(0, x)$ [14]. As activation function for the output layer, a multiclass classification MLP often uses softmax (equation 2.3) for getting the predicted probabilities from an output vector o [13].

$$\hat{y} = \text{softmax}(o) \quad \text{where} \quad \hat{y}_i = \frac{\exp(o_i)}{\sum_j \exp(o_j)} \quad (2.3)$$

Depending on the task to be performed, different loss functions are used, which will be further explained in Section 2.4.3.

2.4.2 One-hot encoding

A one-hot encoding is a vector where each component corresponds to a specific category, with a value of 1 for the relevant category and 0 for all others [13]. For example, given three different labels, "cat", "chicken", and "dog", the resulting one-hot encoding for these would be as presented below [13].

- (1, 0, 0) for "cat"
- (0, 1, 0) for "chicken"
- (0, 0, 1) for "dog"

2.4.3 Loss functions

A loss function is used to optimize the parameters of a model by calculating the error between the ground truth labels and the predicted labels and minimizing this error [15]. Different loss functions are employed depending on whether the model should perform classification (predicting a class) or regression (predicting a numeric value) [15]. This subsection provides an overview of loss functions used in this thesis, both for classification and regression.

Classification

Categorical cross-entropy Categorical cross-entropy loss is a widely used metric for multiclass classification which measures the dissimilarity between the actual label distribution y and the predicted probability distribution p [15]. It can be expressed as

$$\mathcal{L}_{CE}(y, p) = -\frac{1}{n} \sum_{i=0}^{n-1} \sum_{k=0}^{m-1} y_{ik} \ln p_{ik} \quad (2.4)$$

where n is the number of samples, m is the number of classes, y_{ik} is the one-hot encoded true label for the k -th class of the i -th sample, and p_{ik} is the predicted probability for the same class [15].

Regression-by-classification *Regression-by-classification* [16] is a technique where a continuous variable is predicted by discretizing it into bins and training a classifier to predict the bin labels. It can be performed using a loss function which combines the cross-entropy loss (equation 2.4.3) with the one-dimensional Wasserstein loss [16]. The Wasserstein loss quantifies how much effort is required to shift the predicted distribution p to match the true distribution y [17]. For the case of one-dimensional discrete distributions, it can be expressed as

$$\mathcal{L}_{W_1}(y, p) = \frac{1}{n} \sum_{i=0}^{n-1} \sum_{k=0}^{m-1} |Y_{ik} - P_{ik}| \quad (2.5)$$

, where Y_{ik} , P_{ik} represent the cumulative distributions functions of y_{ik} , p_{ik} such that $Y_{ik} = \sum_{l=0}^k y_{il}$ and $P_{ik} = \sum_{l=0}^k p_{il}$ [16]. The loss function for *regression-by-classification* is then defined as

$$\mathcal{L}_{RbC}(y, p) = \alpha \mathcal{L}_{CE}(y, p) + (1 - \alpha) \mathcal{L}_{W_1}(y, p) \quad (2.6)$$

, where $\alpha \in [0, 1]$ and $\alpha = 0.5$ unless specifically stated [16]. The reasoning is that $\mathcal{L}_{W_1}$ will contribute toward the loss function $\mathcal{L}_{RbC}(y, p)$ treating near mispredictions of labels as less severe and penalize mispredictions which are far away from the true labels harder [16].

Regression

Mean Absolute Error (MAE) MAE calculates the average of the absolute differences between predicted and actual values [15].

$$\mathcal{L}_{\text{MAE}}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} |y_i - \hat{y}_i| \quad (2.7)$$

Mean Squared Error (MSE) MSE calculates the average of the squared differences between the predicted values and the actual values. [15].

$$\mathcal{L}_{\text{MSE}}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 \quad (2.8)$$

Larger errors gets amplified because of the squaring operation, which in turn makes MSE (equation 2.8) more sensitive to outliers than MAE (equation 2.7) [15].

2.4.4 Optimization and hyperparameters

The purpose of an optimization algorithm is to provide an efficient way to minimize the loss function used in the model [13]. One of the most widely used optimization algorithms is stochastic gradient descent (SGD), which computes gradients with respect to the loss function and adjusts the model parameters accordingly. The training process is often controlled with hyperparameters, such as learning rate and batch size, which are externally defined parameters and not updated during training. Learning rate controls the step size used by the optimizer for updating the model weights, while batch size controls how many samples from the training data are processed before the model updates its weights [13]. ADAM [18] is an optimization algorithm, based on SGD, which will be used to train the MLP models in this thesis.

2.4.5 Overfitting and underfitting

Two important concepts to consider when creating machine learning models is *overfitting* and *underfitting*. Overfitting occurs when a model performs well on the training dataset but fails to find and learn the true underlying patterns in the data distribution that would allow the model to perform well on new, unseen samples, but rather memorizes the data [19]. Underfitting occurs when a model performs poorly by failing to find any underlying patterns in the training dataset and therefore cannot generalize either [19]. Fortunately, a number of measures can be taken to mitigate this. On data level, dataset augmentation is a widely used technique [20], and is further discussed in Section 2.4.6. On model level, one common technique to prevent overfitting is using random dropout layers [21], where on each training sample a hidden neuron is randomly excluded from the model with a probability p . Moreover, Ioffe et al. [22] introduced batch normalization, which reduces the internal covariate shift of a model by normalizing the inputs of each layer in a neural network to have the same mean and standard deviation. It has been shown to improve training stability and speed as well as acting as a model regularizer [22].

2.4.6 Data augmentation

Data augmentation is a technique used to reduce the generalization error of a machine learning model [12]. In computer vision applications such as image classification or object detection, data augmentation is useful as it can potentially expand the training dataset greatly by applying transformations such as flipping, cropping, shearing and zooming to the images [23]. Data augmentations is usually successful in domains where these image transformations maintain the semantic structure of the data, i.e., predicting on an image of an upside-down cat is still meaningful because the underlying content (the cat) remains the same despite the transformation. In contrast, other domains of data, such as tabular data, lack such clear invariant structures, which makes these transformations less suitable for dataset expansion [24].

2.4.7 Gradient boosted decision trees

Gradient boosting is a concept introduced by Friedman et al. [25]. It can be formulated as a numerical optimization problem which aims to minimize the loss function by adding new corrective models rather than changing model weights for misclassified events [26]. Following the outline in [26], gradient boosting can be described as a model F being constructed iteratively, where F_k denotes the model at step

k . At this stage, F_k serves as an approximation, and errors remain such that $F_k(x) \neq y$. To improve the model, an additional component h_k is introduced, satisfying:

$$F_{k+1}(x) = F_k(x) + h_k(x) = y,$$

which can be rewritten as:

$$h_k(x) = y - F_k(x).$$

Rather than training a completely new model, the process focuses on fitting the residual $y - F_k(x)$, which represents the part of the data not yet captured by F_k . If the updated model F_{k+1} remains insufficient, this procedure is repeated in subsequent iterations until satisfactory performance is achieved [26]. A *decision tree* is a supervised machine learning approach used for both classification as well as regression tasks, where a tree is built based on the training data with conditional statements as branches, decision nodes for said statements and gives its final decision according to the leaf node it ends up in [19]. For regression, the output of the tree model is the mean (or median if using the mean absolute error) value of the training events in the corresponding leaf node [26].

3

Methodology

This chapter aims to explain how the data for this thesis was acquired and preprocessed as well as the designs of the machine learning models. As stated at the start of this thesis (in Section 1.2), the overall goal is to build a machine learning model that, given image sensor data when the IR-cut filter is disengaged, can predict the infrared to visible light ratio, q_{IR} . The algorithm currently used by Axis is proprietary software and thus cannot be disclosed in this thesis.

3.1 Data collection

The collected dataset consists of frame pairs, each containing two frames of the same scene: one captured with the IR-cut filter off (F_{off}) and the other with the filter on (F_{on}). Each pair also includes the two exposure settings captured for the scene. A frame in this context consists of 34×60 blocks extracted from the sensor raw image, where each block is an averaged patch of the neighborhood pixels. The data of a block is a 2-tuple presented in the normalized $\frac{X}{R+G+B}$ coordinate space, $X \in \{R, B\}$, of the red color ratio and blue color ratio. The blocks has not been subjected to any gamma correction and is linear. These ratios $\frac{R}{R+G+B}$ and $\frac{B}{R+G+B}$ will throughout this thesis be referred to as r and b , for the sake of brevity. A frame is numerically represented as a NumPy array of shape $(34, 60, 2)$, where the third axis represents the red color ratio r and blue color ratio b , respectively. As highlighted in Section 2.3, an aspect to consider is that the green color ratio $\frac{G}{R+G+B}$ is not taken into account, due to the fact that r and b are often more interesting from an imaging perspective and is therefore not readily available in the block statistics. Of course, this ratio could be calculated using both r and b to possibly provide the models with some extra spectral information, but we decided to only use the available data to avoid the extra calculations the camera would have to do in a real installation.

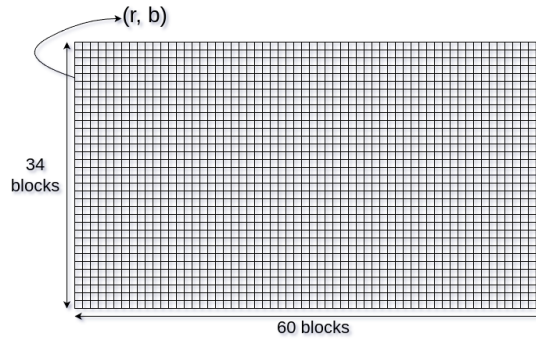


Figure 3.1 Illustration of the block frame layout.

Other potentially important data is the *exposure* value of the camera. The exposure value for the camera used in this thesis can be calculated as

$$exposure = \tau \cdot \beta \quad (3.1)$$

where τ is the shutter speed in milliseconds and β is a linear gain term. Slower shutter speeds result in more light reaching the sensor, allowing more photons to reach the surface of the image sensor. Another concept related to exposure is the *lens aperture*. A lens aperture is an adjustable opening through which

light reaches a camera. It is measured by f-stops, e.g. f/2.8 and f/16. Smaller numbers correspond to wider apertures which admit more light into the camera and vice versa. We will keep the aperture locked at open (f/16) when collecting data, as seen in Table 3.1. We capture the exposure values of both frames F_{off} and F_{on} , as they are used to calculate the ground truth q_{IR} , further explained in Section 3.2.1.

There are some datasets of visible-NIR paired images, such as [27] and [28], which unfortunately is incompatible with the data used in this thesis. One reason is that our camera has a different combination of image sensor and optics. Another reason is that most likely, processing such as gamma correction has been applied to the images in those datasets, where the correction factor γ is not specified. As such, the gamma correction could not be undone from those images. In addition, we do not know the exposure value. Although it would be interesting to explore these datasets, this was not done due to a lack of time.

Data was first and foremost acquired from various imaging labs at Axis, where it is possible to run scripts which systematically illuminates a camera scene with different combinations of illuminants and light intensities. These illuminants range from fluorescent lights with very low amounts of infrared light in their spectral power distributions to lamps which produce large amounts of infrared light, especially when dimmed down to low light levels. Some data was also acquired from natural scenes from around the offices by letting a script collect data from a camera looking out a window from dusk to dawn, as illustrated in Figure 3.2. Comparing the pair of (3.2a, 3.2b) with (3.2c, 3.2d), the latter has a more pronounced difference between its frames, which indicates a larger impact from NIR light. The black spots in frames 3.2a and 3.2c can be attributed to pixel saturation, which we will come back to in Section 4.4. All data was collected with the settings presented in Table 3.1. A total of 1972 samples from 11 capturing sessions were collected for training. In Figure 3.3, this data collection process is summarized in the red module.

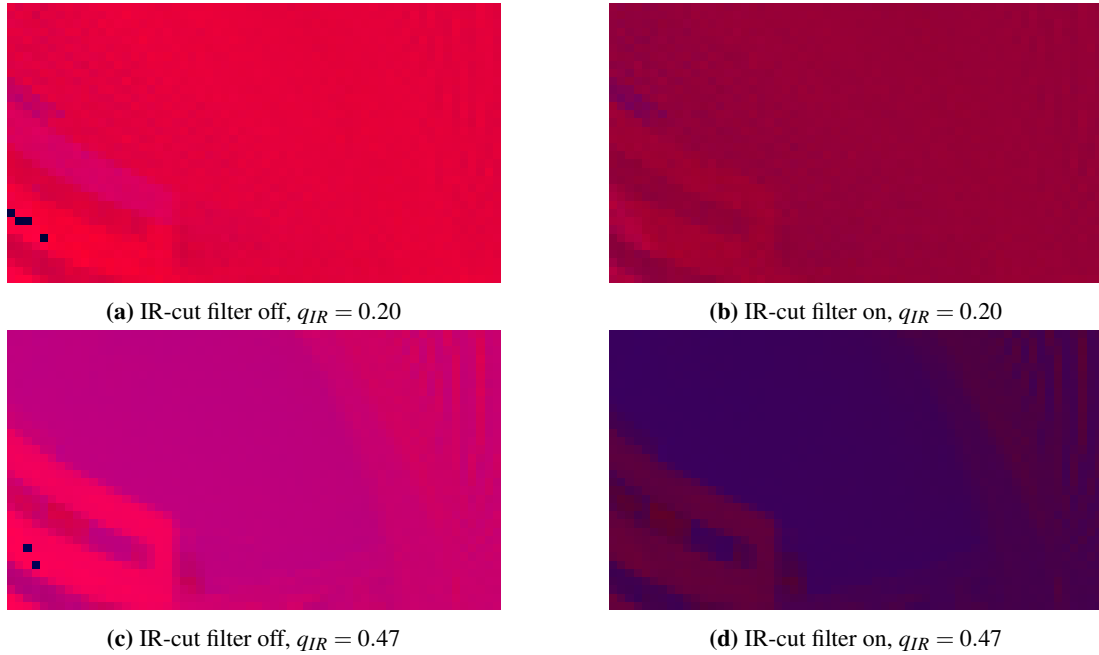


Figure 3.2 Two horizontally paired block frames from the same scene

Camera settings	
Max shutter speed (s)	1/50
Max gain (dB)	75
Aperture locked at	f/16 (open)

Table 3.1 Camera settings used during data collection

3.2 Data preprocessing

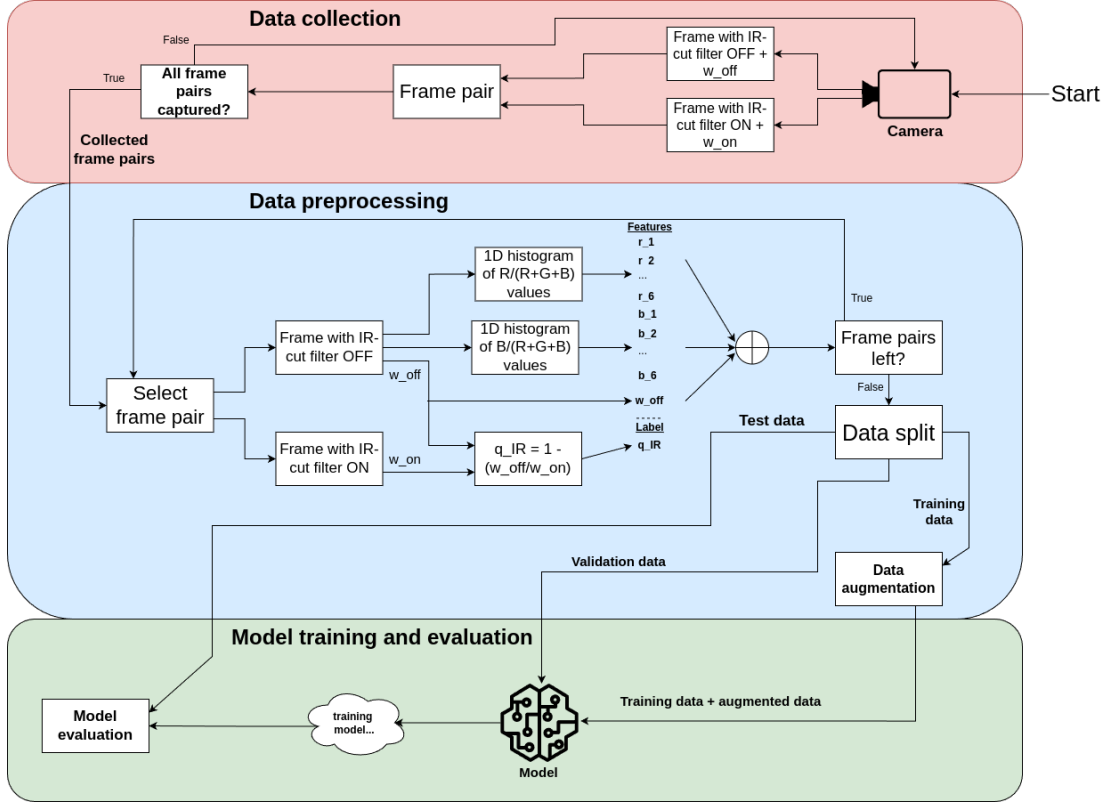


Figure 3.3 Illustration of the data and model pipeline.

Data preprocessing plays an important role in preparing the collected dataset for use in our models. The captured block frames need to be transformed into a format that can be easily understood and processed by machine learning algorithms. Each component of the preprocessing part of the pipeline (the blue module in Figure 3.3) will be described in detail in this section. First, we explain how q_{IR} as a ground truth label (or numerical value in the regression case) is calculated. Second, we describe how features are extracted from the dataset. Finally, we present the feature representation after preprocessing along with a motivation of our approach.

3.2.1 Calculating q_{IR}

q_{IR} is an estimation of how much of the light which strikes the camera sensor lies within the near-infrared region on the electromagnetic spectrum and can be expressed as $q_{IR} = \frac{\text{NIR light}}{\text{NIR light} + \text{visible light}}$, $q_{IR} \in [0, 1]$. The output voltage of the CMOS sensor will generally maintain a linear relationship with the incident photons [5]. This means that given two light source components L_1 and L_2 , the output voltage will be proportional to the sum $L_1 + L_2$. In other words, the sensor responds additively. However, if light levels are very high, the pixels become oversaturated and they will not contain any useful information. On the other end, if light levels are very low, the output signal will be noisy due to sensor gain. It is also important to note that q_{IR} is an estimation that is specific to the camera used in this thesis. Below we explain how the ground truth value for q_{IR} is calculated.

Due to the additivity of the image sensor response, the total scene luminance L_{scene} can be expressed as the sum of the infrared luminance component L_{IR} and visible luminance component $L_{visible}$

$$L_{scene} = L_{IR} + L_{visible} \quad (3.2)$$

We can then write the ratio of infrared luminance to all luminance, q_{IR} , as

$$q_{IR} = \frac{\text{NIR light}}{\text{NIR light} + \text{visible light}} = \frac{L_{IR}}{L_{scene}} \quad (3.3)$$

For the image sensor, the exposure value w is inversely proportional to sensor luminance L

$$w = \frac{1}{L} \quad (3.4)$$

Using equation 3.4, we can now write the exposure value for the frame where the IR-cut filter is engaged, w_{on} , as

$$w_{on} = \frac{1}{L_{visible}} \quad (3.5)$$

and the exposure value for the frame where the IR-cut filter is disengaged, w_{off} , as

$$w_{off} = \frac{1}{L_{visible} + L_{IR}} \quad (3.6)$$

Now, the ratio of the infrared luminance component to the sum of all luminance components, q_{IR} , can be written as

$$q_{IR} = \frac{L_{IR}}{L_{scene}} = \frac{L_{IR}}{L_{IR} + L_{visible}} = 1 - \frac{L_{visible}}{L_{IR} + L_{visible}} = 1 - \frac{w_{off}}{w_{on}} \quad (3.7)$$

For classification, the q_{IR} values are divided into four label intervals: [0.0–0.5], [0.5–0.7], [0.7–0.9], and [0.9–1.0]. We chose these intervals as they approximately correspond to the quartiles of the total dataset distribution of q_{IR} values. Each label value is then one-hot encoded depending on the interval it falls into. For regression, the labels are kept as their original numerical values.

3.2.2 Feature extraction

Each frame F_{off} is transformed into two separate one-dimensional *color histograms*, one for the red ratios r and one for the blue ratios b . The histograms consist of six equally wide bins, with a value range of [150, 300] for r and [50, 200] for b . The widths of the histogram bins were empirically chosen based on observations of the dataset distributions as well as making sure bins/features that would always set to zero were not included. The choice of color histograms as the primary representation of input features has several advantages. They are computationally efficient and interpretable. A potential disadvantage when compared to other alternatives, such as using the full image as input, is that a color histogram essentially discards the spatial information from the image. However, since we are more concerned with the color intensity ratios than the spatial information, this is considered an acceptable compromise. Another aspect is the dimensionality of the color histograms. Some efforts were also made with two-dimensional histograms of r and b , an $m \times n$ matrix where the bin counts correspond to the occurrence of combinations of intensities. While two-dimensional histograms could perhaps capture more complex relationships between r and b , our initial experiments showed no advantage in model performance when compared to their one-dimensional counterparts. Therefore, we chose to use one-dimensional histograms rather than two-dimensional histograms.

Another potentially important feature is the exposure value w_{off} for a given frame F_{off} . As shown in Section 3.2.1, it is a variable that is directly related to q_{IR} and will also be available at inference time for the models. For this reason, we motivate its inclusion as an input feature. Thus, the preprocessed feature representation of the data becomes $[r_1, r_2, \dots, r_6], [b_1, b_2, \dots, b_6], w_{off}$ with q_{IR} as label. After pre-processing is completed, the dataset was further split into training, validation, and test sets, represented with CSV (comma-separated value) files. The splits are approximately 70/20/10. However, due to the fact that the amounts of captured samples and labels varied between each data collection session, it was difficult to ensure similar class balances between the splits. Obviously, we do not want to leak data from the same scene between splits. The class distribution of the entire dataset is shown in Figure 3.4.

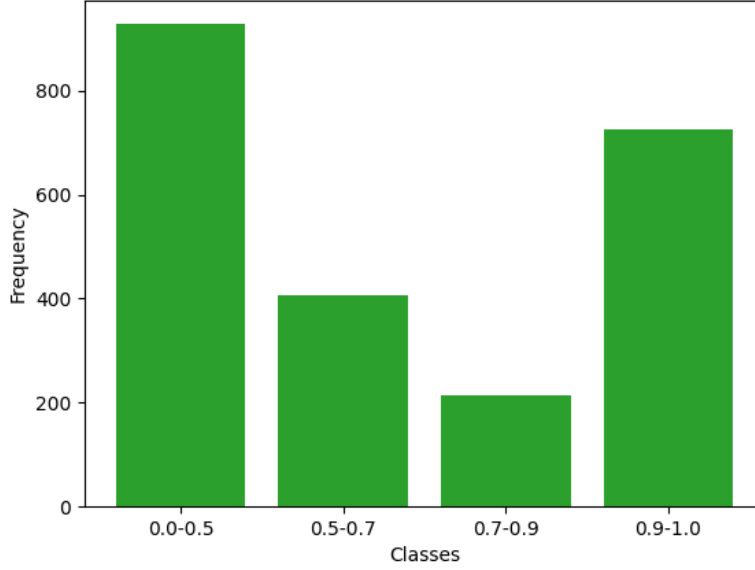


Figure 3.4 Class distributions of q_{IR} values for the entire dataset.

3.3 Data augmentation

As mentioned in Section 2.4.6, data augmentation allows us to extend a limited dataset by applying various transformations and generating new samples for training for the purpose of reducing the generalization error of a machine learning model. For this reason, we believe it is motivated to use some kind of data augmentation for our dataset, which is quite limited in terms of size. Our first approach is to apply random Gaussian noise to both histograms. This is done by sampling a vector $\mathbf{X} \sim \mathcal{N}(0, 30)$ and adding it to the histograms, clipping negative bins to zero. The standard deviation $\sigma = 30$ was chosen after experiments on a subset of the training data. Our second approach is to identify peaks in the histograms, and then randomly redistribute some of the bin frequency values to the neighboring bins. From each data augmentation approach, we generate 300 random new samples. To ensure that our results are reproducible, we also set the same random seed for the pseudorandom generator. We will also train the models with these data augmentations applied both independently and stacked.

3.4 Model architectures

This section presents the machine learning models used in this thesis as well as their designs. A key pointer in the architectural decision making of the models was simplicity. The cameras are essentially embedded systems with limited amounts of memory as well as computational power and many concurrent processes sharing these resources. If we also consider the delimitation that the models in this thesis likely have to be retrained for use on other cameras, it is a positive if the models are small and cheap to retrain. As a result, simpler models with smaller memory footprints are often preferred for these devices. Another advantage of keeping models simple is reducing the risk of potential overfitting.

In its nature, the task of predicting q_{IR} is a regression task. When faced with the choice between reformulating the problem as a classification problem or keeping it as a regression problem, there are several factors to consider. Classification comes at the cost of losing precision, so the choice depends on whether the loss of continuous information in q_{IR} is acceptable. For the task of predicting q_{IR} , this loss can be justified, provided that the predictions are correct and mispredictions not far away from the true label. If the model is overestimating q_{IR} for a scene, this will lead to the camera remaining in night mode for longer than needed. On the other hand, an underestimation would cause the camera to oscillate between day and night mode. Obviously, both scenarios are unwanted but from an end user perspective, the latter one is probably seen as more disruptive than the former. On the other hand,

regression models offer more precision than classifiers as they output a continuous value, which also can be used directly with the existing algorithms. However, they might be more difficult to train. Therefore, we choose to investigate a total of three different models which may be suitable for the task: a regression-by-classification MLP, a pure regression MLP, and a gradient boosted decision tree (GBDT) model also performing regression. One advantage of a GBDT model is that they do not easily overfit [29], which can be beneficial considering our limited dataset. Furthermore, both [30] and [31] found MLPs to perform well for other prediction tasks involving near-infrared imaging, which suggests that they may be suitable for our task as well.

3.4.1 Regression-by-classification MLP

The regression-by-classification model consists of an input layer, a batch normalization layer, a dense layer with 64 units and ReLU activation, a dropout layer with a dropout rate of 0.1, and finally a output layer with four units and softmax activation. The model uses ADAM for parameter optimization and is built with the Keras [32] framework. Due to the task being a regression problem at its core, we also want the loss function to consider the inherent order of the labels, i.e. making wrong predictions which are far from the true label should be penalized more than wrong predictions that are closer to the true label. For this reason, we believe it is motivated to use the loss function for regression-by-classification $\mathcal{L}_{\text{RbC}}(y, p)$ to achieve this. Another aspect to consider is the imbalance between classes in the dataset, shown in figure 3.4. One way to mitigate this on model level is using class weights. Imbalanced data can bias models toward the majority class, leading to poor minority class performance. We try to counteract this by assigning higher weights to minority samples during training. These weights are calculated for each class i as $\text{weight}_i = \frac{\text{Total samples}}{\text{Number of classes} \cdot \text{Frequency of class}_i}$, which in turn will be assigned to the loss function. In other words, we tell the model to pay extra attention to underrepresented classes during training. Hopefully, this will encourage the model to learn minority class patterns more efficiently and in turn improve overall model performance.

Hyperparameters The hyperparameters used for training the regression-by-classification model are presented in Table 3.2. These values were chosen by iteratively adjusting the parameters manually and evaluating the model’s performance on a subset of the validation set.

Parameter	Value
num. epochs	60
batch size	17
learning rate	0.001
α	0.25

Table 3.2 This table presents the hyperparameters used for training the regression-by-classification MLP.

The number of epochs and learning rate was chosen on the basis that the network should given enough time with a steady learning rate to converge. A batch size of 17 was also found to have a positive impact on validation accuracy. The parameter α controls the weighting between the cross-entropy and Wasserstein loss of the loss function $\mathcal{L}_{\text{RbC}}(y, p)$ (equation 2.6), where experiments showed $\alpha = 0.25$ to be a good trade-off between the two terms.

3.4.2 Regression MLP

The regression MLP consists of an input layer, a batch normalization layer, a dense layer with 16 units and ReLU activation, a dropout layer with a dropout rate of 0.15, and a output layer with a single unit and an activation function that clips output values to [0,1]. Our reasoning behind the choice of the final activation function is that since the range of q_{IR} is between 0 and 1, it can be beneficial to constrain the model outputs within this range. This may help the model training to converge faster, as the outputs are already within the desired range. For training the model, we use MSE (equation 2.8) as loss function. The model also uses ADAM for parameter optimization and is built with the Keras [32] framework.

Hyperparameters The hyperparameters used to train the regression model are however different and are presented in Table 3.3.

Parameter	Value
num. epochs	100
batch size	32
learning rate	0.001

Table 3.3 This table presents the hyperparameters used for training the regression MLP.

Similar to the regression-by-classification MLP, the hyperparameters were chosen through iterative manual tuning on a subset of the validation set.

3.4.3 Gradient boosted decision trees

For constructing the GBDT model we use LightGBM [33], which is a tree-based ensemble framework for efficiently constructing gradient boosted decision trees. These parameters were selected using the hyperparameter optimization framework Optuna [34]. The parameters used for training the model are shown in Table 3.4, along with brief descriptions.

Parameter	Value
Learning Rate	0.143
Number of Leaves	102
Max Depth	2
Min Child Samples	21
Min Child Weight	6.33×10^{-5}
Min Data in Leaf	5
Subsample	0.853
Subsample Frequency	1
Regularization Alpha	5.25×10^{-4}
Regularization Lambda	9.82×10^{-5}
Min Split Gain	0.091

Table 3.4 Hyperparameters used for the LightGBM Model, selected using Optuna [34].

We can see that the learning rate was selected to be quite high, which allows for large updates with each boosting iteration but might make the model a bit more susceptible to overfitting. The main parameter for controlling model complexity is the number of leaves, which is set to 102, along with a low max tree depth of 2. The minimum child samples ensures that a node can only be split if it contains more than 21 data points. The minimum data in leaf parameter is similar to the former, but controls the minimum number of data points *after* a split. The subsample frequency of 1 means that subsampling of the dataset is used for every boosting iteration. The subsample parameter tells the model to subsample 85.3% of the training data for each boosting iteration. The regularization alpha and lambda values are used for penalizing model complexity. The minimum split gain of 0.091 is a threshold for potential tree splits, so that only informative splits are added to the model. Lastly, the minimum child weight parameter defines the sum of the weights for all samples in a leaf. It can be seen as a secondary threshold to prevent overly small splits.

3.5 Model evaluation

To better evaluate the models, an additional test set was collected from a real life scene. The collection took place outdoors during a winter morning in December 2024 and captured the transition from night to day, which resulted in 116 samples. However, we discovered a lack of samples with q_{IR} values within the range of $[0, 0.5]$ after finishing the data collection. As such, we also use an additional test set containing 17 such samples. This aggregated dataset of 133 samples will be used for the evaluating the overall predictive performance of the models. Below are a selection of both classification and regression metrics that will be used to evaluate the models. In Figure 3.3, the model training process along with the model evaluation is summarized in the green module. Furthermore, the acceptance criteria from Axis for the model predictions to be useful in a real camera environment is stated in equation 3.8.

For all q_{IR} values, the following condition must hold:

$$|q_{IR} - q_{IR}^*| \leq \varepsilon \quad (3.8)$$

where q_{IR} is the ground truth value, q_{IR}^* is the predicted value, and ε is the permissible error margin, based on the range of the ground truth q_{IR} values (Table 3.5)

q_{IR}	ε
$[0.9 - 1.0]$	stay inside range
$[0.7 - 0.9]$	0.10
$[0.5 - 0.7]$	0.15
$[0.0 - 0.5]$	0.20

Table 3.5 Error margins for different q_{IR} ranges.

For $[0.9 - 1.0]$, it suffices to know that the scene is dominated by infrared light. The most critical regions are $[0.7 - 0.9]$ and $[0.5 - 0.7]$, hence the stricter values for ε . Note that this criteria cannot be directly applied to the regression-by-classification MLP.

Classification

The predictive ability of a classifier can be evaluated using a confusion matrix, which is a square matrix table that visualizes the classifier's predictions [35].

- TP: True Positives - where the model correctly identifies positive instances,
- TN: True Negatives - where the model correctly identifies negative instances,
- FP: False Positives - where the model incorrectly predicts the positive class,
- FN: False Negatives - where the model incorrectly predicts the negative class.

Using these definitions, the following classification metrics can be calculated [35].

- **Accuracy and Precision:**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{Precision} = \frac{TP}{TP + FP}$$

- **Recall and F1-Score:**

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

In addition, we will also look at macro averaged metrics to gain a more comprehensive understanding of the overall predictive abilities of the models. Macro averaging calculates the metric separately for each class and then just takes the unweighted mean.

Regression

The regression models will be evaluated using metrics such as RMSE (equation 3.9), MAE (equation 2.7), and R^2 -score (equation 3.10).

Root mean squared error Root mean squared error, or RMSE, is a metric that calculates the average magnitude of the errors in a set of predictions. It is defined as

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2} \quad (3.9)$$

where y_i are the true values and $\hat{y}_i$ are the predictions.

Coefficient of determination The coefficient of determination, or R^2 -score, tells how much of the variation in the target variable can be explained by a model. An R^2 of 1 means the model perfectly predicts the data, while 0 means it's no better than guessing the mean. If R^2 is negative, the model performs worse than just using the mean. The formula is:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (3.10)$$

where y_i are the true values, $\hat{y}_i$ are the predictions, and $\bar{y}$ is the mean of y_i . The closer R^2 is to 1, the better the model explains the data.

3.6 Ablation study

The purpose of the ablation study is to gain an understanding of the contributions from different model components. In this thesis, we will take the best performing MLP and retrain it without data augmentation, batch normalization, and dropout layers. We hope that the results of the ablation study will motivate the inclusion of aforementioned components in the model.

4

Results & Analysis

This chapter presents the results of our models. First, we present the results of the models evaluated on the test set. Second, we present an comparison of the best configuration of each model. Third, we conduct an ablation study on the overall best performing model. Finally, we discuss challenges and limitations encountered during the thesis.

4.1 Model results

As a reminder, the task is to predict q_{IR} for a given camera frame where the IR-cut filter is off. To achieve this, we have three different machine learning models, one regression-by-classification model and two regression models. In this section, we test six different augmentation configurations (no augmentation, Gaussian noise, peak shifting, noise + peak shifting, and stacking the augmentations both ways) and discuss the model results. Common to all models is that neither structures nor hyperparameters change throughout the experiments but remain as described in Section 3.4. However, since these were optimized for the model using no data augmentation, there is an inherent bias for this in the rest of the models.

Regression-by-classification The results for the regression-by-classification model trained with the different data augmentations are shown in Table 4.1.

Augmentation	Accuracy $\uparrow$	Macro Prec. $\uparrow$	Macro Recall $\uparrow$	Macro F1-Score $\uparrow$
No augmentation	0.42	0.45	0.35	0.31
Gaussian noise	0.81	0.79	0.83	0.81
Peak shifting	0.62	0.68	0.63	0.61
GN + PS	0.37	0.59	0.46	0.41
Stacked augm. (GN $\rightarrow$ PS)	0.79	0.77	0.80	0.78
Stacked augm. (PS $\rightarrow$ GN)	0.78	0.78	0.77	0.76

Table 4.1 Comparison of data augmentations for the regression-by-classification model.

We can see that the regression-by-classification model performance without any data augmentations is poor, where the highest score is the macro precision = 0.45. Fortunately, the model performance sees a drastic increase together with the Gaussian noise augmentation, achieving the highest score in accuracy and macro precision/recall, resulting in a high macro F1-score. Furthermore, we can see that the GN $\rightarrow$ PS stacked augmentation achieves the second-highest score across all metrics. Both of our proposed augmentations of the training histograms seems to have a positive impact on model performance. On the other hand, applying both augmentations independently resulted in an even further decline in performance except in precision, as illustrated in Figure 4.1.

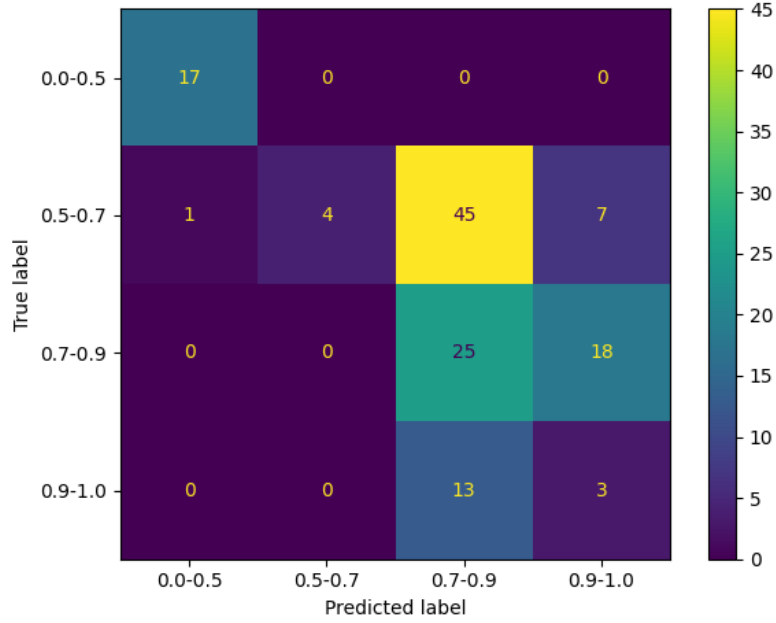


Figure 4.1 Confusion matrix of the worst-performing regression-by-classification MLP (Gaussian noise + peak shifting).

It is not entirely clear why applying both augmentations independently causes the model to experience such a drastic decline in performance. One possible reason could be that a too large portion of the already limited training data now consists of augmented samples for the model to learn useful patterns in the training data. Since we also use the same learning rate and epochs to train all models, it is probable that the model overfits by rather fitting patterns of noise in the data.

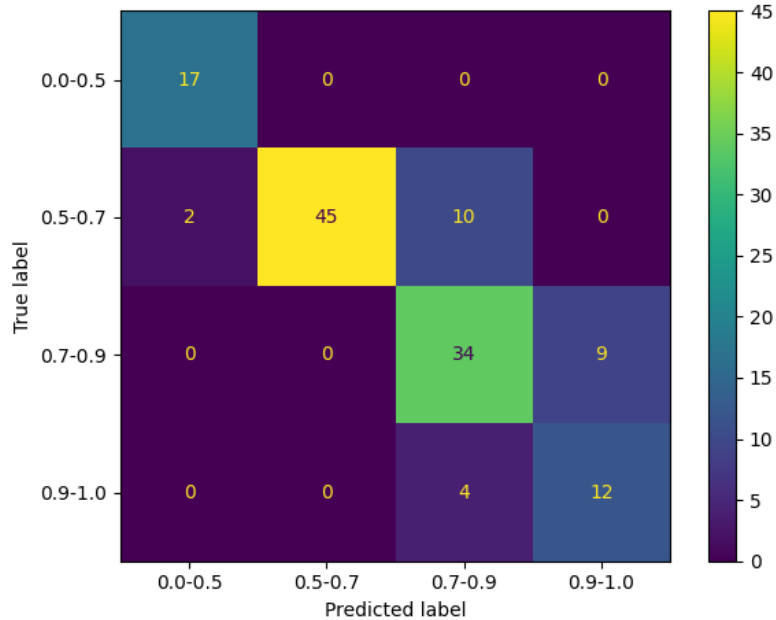


Figure 4.2 Confusion matrix of the best-performing regression-by-classification MLP (Gaussian noise).

Positively, using the loss function for regression-by-classification seems to have been successful, as there are no severe mispredictions (predictions more than one label off), as shown in Figure 4.2. Adding Gaussian noise to the histograms of r and b seems to have a regularizing effect. However, when comparing the test accuracy of this model to training and validation accuracies, it showed a decline in accuracy

from approx. 90% to 81.2%, which indicates that this model is also slightly overfitting on the training data. In Table 4.2, further performance metrics for the aforementioned model.

Metric	0.0–0.5	0.5–0.7	0.7–0.9	0.9–1.0
Accuracy	1.0	0.79	0.79	0.75
Precision	0.91	1.0	0.71	0.57
Recall	1.0	0.79	0.79	0.75
F1-Score	0.95	0.88	0.75	0.65

Table 4.2 Performance metrics for the best-performing regression-by-classification MLP (Gaussian noise).

As demonstrated by the relatively low scores across the performance metrics, [0.9-1.0] seems to be the hardest label to make correct predictions for, even though it has the second largest amount of samples. One probable reason is the lack of training samples with the label [0.7-0.9] leading to the model not finding patterns necessary to distinguish between samples with these two labels. It can be also be that some numerical ground truth values of q_{IR} were just below the upper interval limits of [0.7-0.9]. As such, they had similar feature representations to samples with label [0.9-1.0] but were labeled differently by the preprocessing pipeline. These cases are hard to distinguish between for the regression-by-classification MLP, indicating that there is room for improvement regarding the preprocessing of the data. Another explanation can be found in the loss function. The Wasserstein loss term in $\mathcal{L}_{\text{RbC}}(y, p)$ aims to minimize the total distance that probability mass needs to be moved. As such, when this loss is uncertain, it shifts the mass towards the center as this generally minimizes the average distance to other points within the distribution. In other words, when $\mathcal{L}_{\text{RbC}}(y, p)$ is uncertain, it will favor middle labels such as [0.5-0.7] and [0.7-0.9]. This indicates that the parameter $\alpha = 0.25$ in $\mathcal{L}_{\text{RbC}}(y, p)$ might weigh the Wasserstein loss term too highly and that $\mathcal{L}_{\text{RbC}}(y, p)$ is not confident enough to shift mass towards the labels at the extremes, such as [0.9-1.0].

Regression models In Table 4.3, a metric comparison for the two regression models is presented.

Augmentation	Model	RMSE ↓	MAE ↓	R^2 ↑
No augmentation	LightGBM	0.144	0.127	0.524
	Regression MLP	0.071	0.057	0.883
Gaussian noise	LightGBM	0.137	0.117	0.592
	Regression MLP	0.072	0.056	0.883
Peak shifting	LightGBM	0.138	0.116	0.560
	Regression MLP	0.074	0.057	0.876
Gaussian noise + peak shifting	LightGBM	0.137	0.118	0.569
	Regression MLP	0.076	0.063	0.869
Stacked augm. (GN→PS)	LightGBM	0.148	0.123	0.502
	Regression MLP	0.071	0.059	0.882
Stacked augm. (PS→GN)	LightGBM	0.140	0.123	0.567
	Regression MLP	0.075	0.062	0.872

Table 4.3 Comparison of data augmentations for the regression models.

For all six augmentation configurations, we can see that the regression MLP performs better than the LGBM model across all three metrics. When comparing the models without augmentations, the regression MLP achieved both a lower RMSE (0.049) and MAE (0.057) than the LGBM model (RMSE = 0.144, MAE = 0.127). Furthermore, the R^2 score for the regression MLP was higher ($R^2 = 0.883$ compared to $R^2 = 0.524$). With $R^2 = 0.592$, the LGBM model accounts for about 59.2% of the variance, which indicates that the model is missing some important patterns in the data by leaving 40.8% of the variance unexplained. On the other hand, the regression MLP achieved $R^2 = 0.883$, which means that 88.3% of the variance is explained, leaving just 11.7% unexplained. Once again, combining or stacking both augmentations seems to have a detrimental effect on model performance, though not as prominent as in the case of the regression-by-classification MLP. The most surprising result is perhaps the

small impact that data augmentation has on the models. When comparing model performance metrics for different augmentations, it is clear that only the LGBM model of the two models experiences some increase in performance by training on augmented data. This indicates that our suggested approaches for data augmentation may not be effective for the regression models. One reason for this may be that analogous to the classification case, we do not modify the q_{IR} . It is not very clear how to modify this value either. It is hard to pinpoint an exact cause for the performance drop but it is likely related to model confusion stemming from these erroneous samples. In conclusion, the regression MLP shows significantly better results at predicting q_{IR} as a continuous value. In Table 4.3, we can also see that this conclusion holds for model comparisons across all augmentations.

However, these metrics do not necessarily offer a comprehensive depiction of the situation. For example, $MAE = 0.057$ means that, on average, the model's predictions deviate from the ground truth q_{IR} by 0.057 on a scale from 0 to 1. $RMSE = 0.071$ means that the average magnitude of the errors is 0.071, which is slightly larger than the MAE, partly due to the squaring effect. To put these metrics further into perspective, we provide scatter plots of the predictions of both the LGBM model (Figure 4.3) and the regression MLP (Figure 4.4).

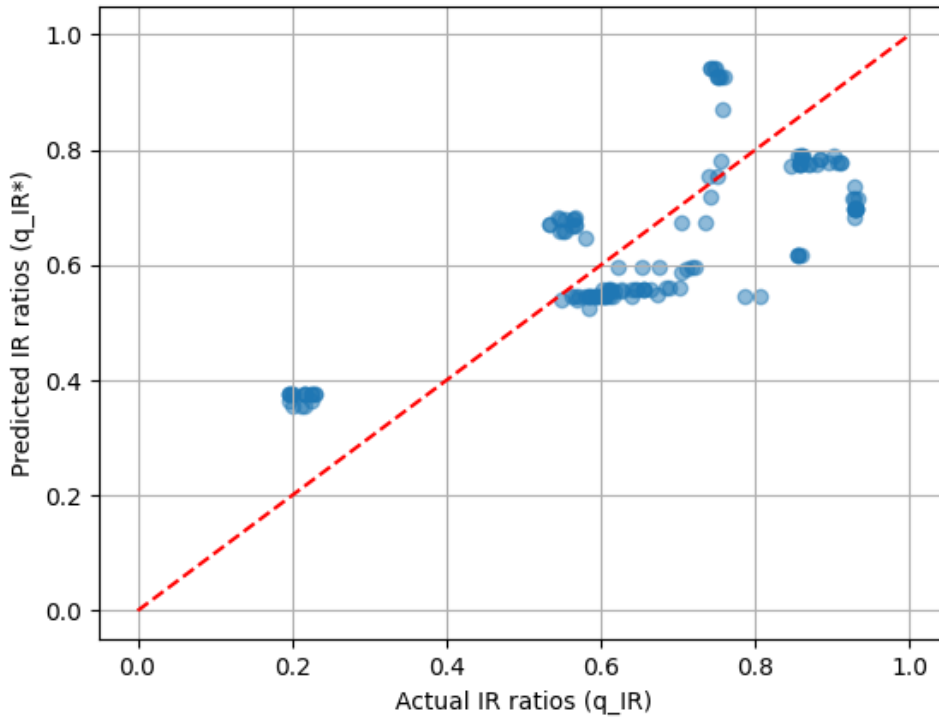


Figure 4.3 Scatter plot of the predictions (blue) of the LGBM model (Gaussian noise as augmentation). The red dashed line serves as a visual reference point, representing the ideal predictions where the model's outputs would perfectly match the target values.

Observing Figure 4.3, we can see that the LGBM model seems to cluster predictions. One possible reason for this is simply overfitting. It seems like the model has failed to create a decision tree with splits from the training data which would allow it to generalize well to the test set. The LGBM model showed good performance on both training and validation set, but evidently not on the collected test set. This points to either the distributions of the train and the test set being too different for the model or that the model is simply overfitted. As for the scatter plot in Figure 4.4, the regression MLP exhibits a better overall fit to the data than the LGBM model. Still, the error patterns reveal some notable systematic bias in that the MLP has a tendency to overestimate q_{IR} . Additionally, there is more variance in the predictions at the higher values, right below 0.8 on the horizontal axis, where some samples are predicted as > 0.9 . Overall, the regression MLP appears to have learned some reasonable, but not optimal, patterns for generalization.

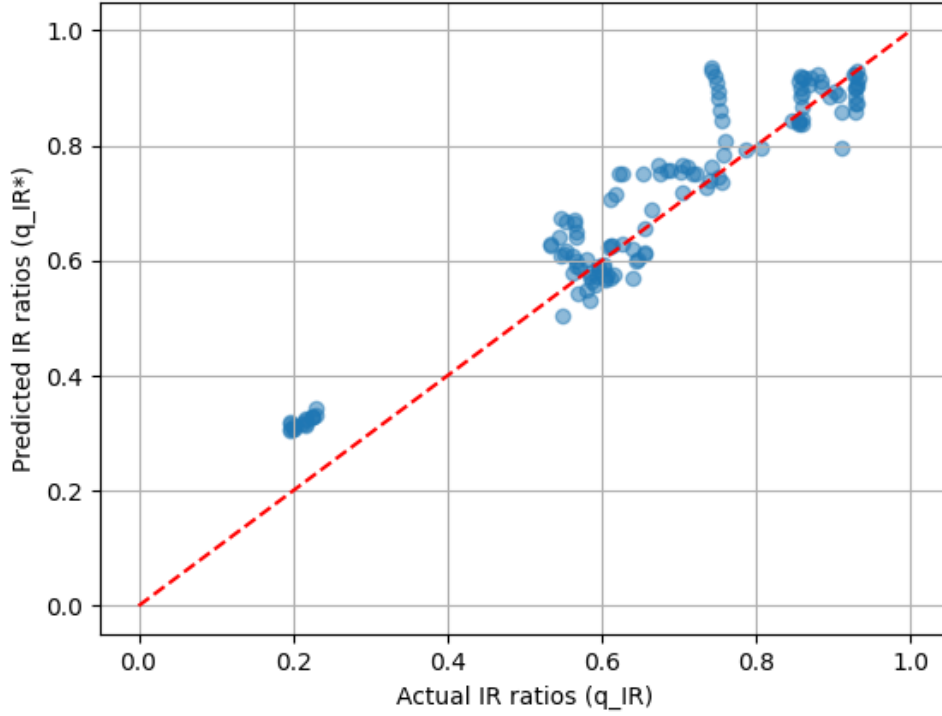
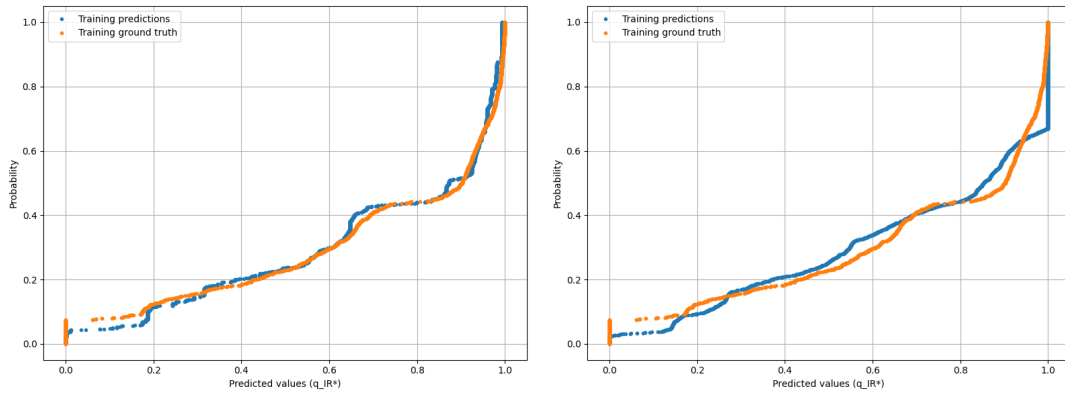


Figure 4.4 Scatter plot of the predictions (blue) of the regression MLP (Gaussian noise as augmentation). The red dashed line serves as a visual reference point, representing the ideal predictions where the model's outputs would perfectly match the target values.

As is evident from both scatter plots (Figure 4.3 and 4.4), there is a lack of samples with q_{IR} values in the range of $[0,0.5]$. To provide more information about the predictive abilities of the models, we present plots of the cumulative distribution functions from the q_{IR}^* predictions during training in Figure 4.5.



(a) LGBM training q_{IR}^* predictions.

(b) RMLP training q_{IR}^* predictions.

Figure 4.5 Cumulative distribution functions (CDF) of the training q_{IR}^* predictions.

Both models perform well during training, with the LGBM model (Figure 4.5a) perhaps showing an even better fit with its predictions. Conversely, this points even further to that this model is overfitted and cannot generalize to unseen data. In Figure 4.5b, the RMLP clips a large portion of high value q_{IR} samples to 1 during training.

We can understand more about the model predictions by examining the histogram of model residuals, or the differences between predictions and actual values.

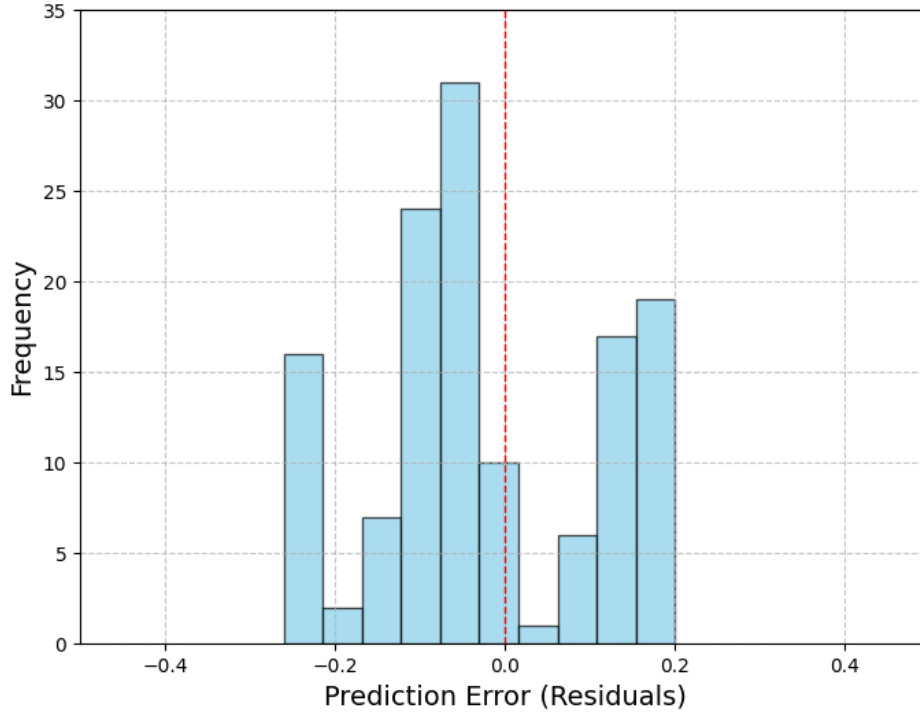


Figure 4.6 Error histogram of the LightGBM regression model.

As illustrated in Figure 4.6, the shape of the residual distribution appears multimodal. Moreover, the spread of the prediction errors is quite wide, with large bins across a range of below -0.2 and up to 0.2. Note also the difference in scale on the vertical axis for both histograms, where the magnitude of the residual bins is much larger for the LGBM model than for the regression MLP. All in all, this affirms that the LGBM model struggles to learn useful patterns from the training data.

For the regression MLP, Figure 4.7 shows that the residuals are centered closer to zero than for the LGBM model, with a clear peak near zero. The distribution also appears not overly asymmetric. There is however a slight negative skewness, which indicates a bias in the model to overpredict q_{IR} . More positively, the error residuals fall within a range (around -0.15 to right below 0.2).

As stated in Section 3.5, the acceptance criteria was that for all q_{IR} values, the condition $|q_{IR} - q_{IR}^*| \leq \varepsilon$ must hold, where ε is the permissible error margin, based on the range of the ground truth q_{IR} values. The results from the acceptance test with the best performing model configuration (Gaussian noise) are presented in Table 4.4.

q_{IR}	ε	RMLP	LGBM	Nbr. of samples
[0.9 – 1.0]	stay inside range	43.75%	0.0%	16
[0.7 – 0.9]	0.10	83.72%	25.58%	43
[0.5 – 0.7]	0.15	100%	100%	57
[0.0 – 0.5]	0.20	100%	0.0%	17

Table 4.4 Correct prediction percentages of the acceptance test.

As depicted in Table 4.4, both the RMLP and LGBM fail to meet the error level for the acceptance criteria with their predictions on the test set, although the former comes closer. The RMLP aces the predictions for both [0.0 – 0.5] and [0.5 – 0.7]. However, making correct predictions for [0.9 – 1.0]

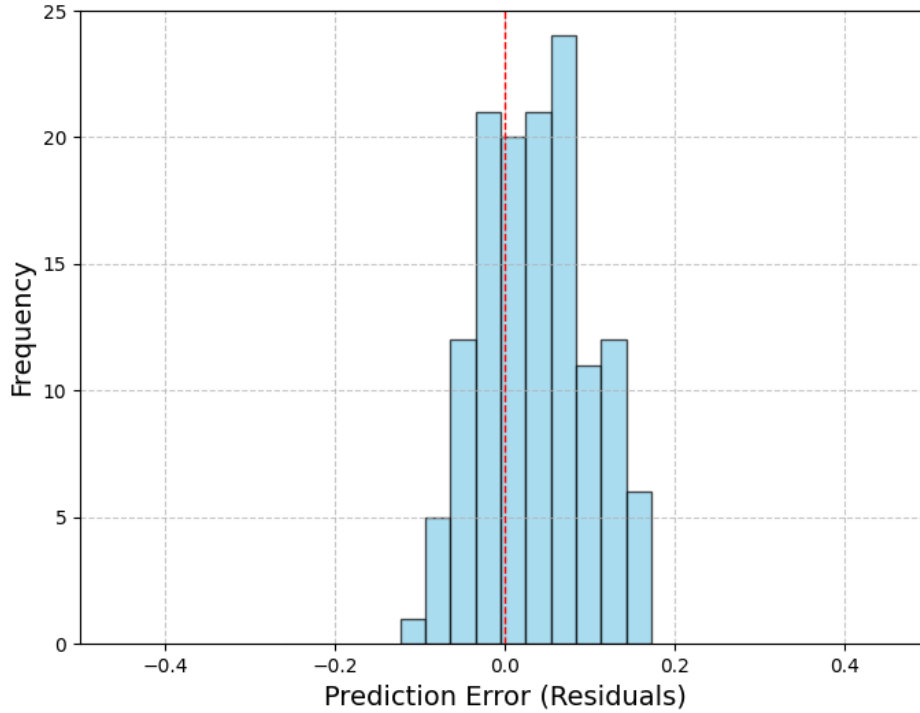


Figure 4.7 Error histogram of the regression MLP.

seems to be tougher with no correct prediction for LGBM and only 43.75% correct predictions for the RMLP. In the case of the latter, the activation function clipping high q_{IR} values during training (as seen in Figure 4.5b) may cause the model to not learn enough information to predict $[0.9 - 1.0]$ samples correctly, even though they comprise the second largest class in the dataset.

In summary, we find that the regression MLP outperformed LGBM in all metrics, with little benefit from augmentations for either model. LGBM showed signs of overfitting and struggled to generalize, as showcased by its clustered predictions and wide residual distribution. This points to that the LGBM may not be a suitable architecture for the task, at least not given the current set of training data. While more accurate overall, the MLP also displayed a bias toward overestimating q_{IR} . These results showcase the limitations of current augmentation methods and the regression MLP's superior capacity to model the data despite some biases.

4.2 Comparing regression-by-classification and regression models

Preferably, our models should be benchmarked on a real camera installation for seeing how they control the IR-cut filter but due to a lack of time and technical knowledge for creating an implementation, this could not unfortunately not be done. Regardless, it is of interest to compare the model performances, but as the models differ in their problem approaches (classification and regression), some minor adjustments have to be made. Therefore, we bin the predictions of the regression models into the corresponding bin labels used for regression-by-classification. In this way, we can compare the predictive performances of all three models with the best augmentation (Gaussian noise). The results of this comparison are presented in Table 4.5.

ξ_k	RbC	RMLP	LGBM
ξ_1	18.8%	24.1%	22.6%
ξ_2	0.0%	0.0%	6.0%
ξ_3	0.0%	0.0%	0.0%

Table 4.5 Comparison of the three best performing models of each architecture. ξ_k indicates the fraction of the predictions that are k labels off the true label. The smallest fraction for each ξ_k is typed in boldface font.

We can see that the regression-by-classification MLP has the lowest fraction of mispredictions and performs best, with only 18.8% of predictions one label off (ξ_1) and no larger errors ($\xi_2 = 0\%$, $\xi_3 = 0\%$). The regression MLP has slightly more one-label errors at 24.1%. The LGBM model has 22.6% one-label-off predictions and 6% two-label-off predictions, which makes it the least accurate of the three. Going by this comparison, the regression-by-classification MLP is the most consistent performer. The severity of mispredicting certain labels also differs. For example, a model predicting $q_{IR} = 0.51$ as $q_{IR}^* = 0.49$ will be put in ξ_1 . On the other hand, a model predicting $q_{IR} = 0.71$ as $q_{IR}^* = 0.95$ will also be put in bin ξ_1 , even though this is a much worse prediction when applied to a real context of predicting q_{IR} .

Furthermore, we also compare training time, inference time, and memory usage of the models in Table 4.6

Metric	RbC	RMLP	LGBM
Mean training time	36.16s	36.31s	1.07s
Mean inference time	2.0ms	2.0ms	4.2ms
Memory usage	4.72KB	1.14KB	352KB

Table 4.6 Comparison of training time, inference time, and memory usage between the models. The lowest value for each metric is typed in boldface font.

We can see that all three models are low in training time, inference time, and size. The LGBM model has the fastest mean training time by a margin at 1.07 seconds, while the regression-by-classification MLP and RMLP are similar at over 36 seconds. For inference time, RbC and RMLP are faster at 2.0 milliseconds compared to 4.2 milliseconds for LGBM. In practice, all of these times are acceptable for the task. Finally, the RbC and RMLP are the most efficient with regard to memory usage, even though all three models are relatively low in size.

4.3 Ablation study

In the previous section, a head-to-head comparison showed that the model with the highest prediction accuracy was the regression-by-classification MLP, with Gaussian noise as data augmentation. In the ablation study, this model is retrained from scratch without data augmentation, as well as without batch normalization and dropout layers. The results from the ablation study are presented in Table 4.7.

Configuration	Data Augm.	Batch Norm.	Dropout	Accuracy $\uparrow$
W/o Data Augm. & Dropout		X		33.8
W/o Data Augm., Batch Norm. & Dropout				34.6
W/o Batch Norm. & Dropout	X			34.6
W/o Data Augmentation		X	X	42.1
W/o Batch Normalization	X		X	45.1
W/o Data Augm. & Batch Norm.			X	46.6
W/o Dropout	X	X		66.9
Best Model (Full Configuration)	X	X	X	81.2

Table 4.7 Ablation study of various configurations of model components.

The most unexpected result from the ablation study is perhaps the poor performance of the model where only batch normalization was used. It seems that both of the ablated components techniques

play an important role in stabilizing training and preventing overfitting. The model trained without dropout, but with both data augmentation and batch normalization, showed a significant improvement in performance and achieved an accuracy of 66.9%. This points to that data augmentation and batch normalization working in tandem is what enables the model to find some meaningful patterns in the training data. The same pattern is also seen with the best performing model, which included all three components and achieved an accuracy of 81.2%. This demonstrates that all of these components had a positive impact on model performance when combined.

4.4 Challenges and limitations

This section aims to highlight various challenges and limitations encountered during the research process.

4.4.1 Data collection

Dataset Due to a lack of an appropriate existing dataset for the task, this also had to be established. The vast majority of the data collection was handled automatically by scripts without need for human intervention, which overall worked well. The biggest problem was the limited dataset. As previously stated, the majority of the data was acquired from imaging labs at Axis and while these labs allowed for large control of scene illumination, there was a lack of diversity with regard to scene dynamics. The static nature of the captured camera scenes meant that the scenes reflected light relatively uniformly, which resulted in the histograms of some samples to be quite similar to one another. As such, it would be desirable to acquire more data from a variety of camera installations to expand the dataset. Although there were some plans for this at the start, these never came to fruition. In addition to collecting more data, alternative approaches for data augmentation e.g. using variational autoencoders to generate training samples should also be explored for the purpose of expanding the dataset.

Noise We also noticed that some frames captured in low-light conditions had noise issues. This was caused by the camera's autoexposure algorithm responding to low light levels by increasing gain and slowing the shutter speed. The resulting frames often had large variations in pixel block values, sometimes deviating by as much as ± 40 depending on the millisecond that the frame was captured. Fortunately, this was only noticed in a few frames, which did not seem to impact overall performance. Thus, no further action was taken to remove them. On the other end, frames captured in high light level environments sometimes suffered from pixel overflow, due to the saturation of CMOS sensor elements. This results in the corresponding blocks in the frame being set to 0 and containing no information. The reason for why this happened is that the autoexposure algorithm failed to set an exposure value such that a reasonable amount of light reaches the sensor and at the same time avoids saturation. Once again, this was not deemed to be a big problem and no further action was taken to remove them either.

Test set The test set used for final evaluation also has some technical limitations due to several reasons. First, the camera was mounted on a tripod standing on the ground, which is likely an uncommon way to position a surveillance camera. However, there were no other real alternatives for nearby environments in which this data could be collected undisturbed and without other people being recorded. Of course, it would be desirable to have the data collected with the camera more realistically mounted. Second, the camera was tilting slightly upward from its ground position, capturing large portions of the sky and thus high ratios of blue color. This likely had some impact on the distributions of r and b in the test set as it represented a scene which could not be produced in the lab and therefore not included in the training data. In other words, the test set was a tough selection of samples for the models. Positively, the results can be an indicator of some model ability to generalize to previously unseen environments but comparisons with other test sets are needed to verify this. Seemingly, there are also not a lot of samples with q_{IR} values in the range of $[0.0-0.5]$ in the collected test set. This points to the possibility of its label equivalent being a bit redundant and raises questions about its relevance in a real installation when the camera needs to perform day/night-synchronization.

4.4.2 Data preprocessing

The features representations of the block frames were two histograms and one exposure value. Histograms provided a simple and interpretable way to represent the frames but likely missed finer details that could help in predicting q_{IR} . The exposure value most likely added valuable context about the

lighting conditions of the scene, though its exact impact on performance is unclear. While this feature set worked for our models, more statistical features describing the scene could be used, such as mean, standard deviation, and skewness of the distributions of r and b , could improve results. Perhaps a distance metric between these distributions, such as the Wasserstein distance, might provide the models with additional important information about the spectral characteristics of the light which reaches the sensor. Therefore, we suggest that future work should also explore whether these alternative feature representations can enable the models to capture even more patterns and use them for better predictions.

In addition, a more systematic approach than just eyeballing the distributions for choosing the number of bins as well as their widths. It is likely that we did not choose the optimal histogram representation for the block frames. Alternative approaches such as the Freedman-Diaconis rule [36] could be used for better bin width selection. This rule seeks to minimize the integral of the squared error between the estimated probability density function (which is derived from the histogram) and the true probability density function of the underlying r and b distributions. From this, the authors inferred that a good representation in terms of bin width is $2 \frac{IQR(x)}{\sqrt[3]{n}}$, where $IQR(x)$ denotes the interquartile range of the dataset and n is the number of observations in the sample x . In summary, more careful preprocessing of the histograms may very well improve the results.

4.4.3 Models

The models presented in this thesis were developed with the scarcity of the dataset in mind. The MLPs were designed with different regularization mechanisms such as batch normalization and dropout layers. The parameters of the LGBM model were tuned on the validation set, but still failed to fully translate its learned knowledge to the test set, something which the MLPs did better. This points to that a GBDT-based model might not be the most suitable architecture for the task. Moreover, there are not many immediate modifications to make for improving model performance, but rather improve the data first before modifying the model architectures further. However, the MLPs could likely benefit from a more systematic approach of hyperparameter tuning, such as performing a hyperparameter search with Optuna [34]. Future work should look into this as well as exploring other machine learning models for making more comprehensive performance comparisons.

5

Conclusions & Future Work

In this final chapter we present the conclusions from this thesis as well as directions for future work.

5.1 Conclusions

In this thesis, we presented three different machine learning models for estimating q_{IR} , i.e. how much of the incoming light lies within the near-infrared spectrum. The research questions of the thesis are restated below:

- Is it possible to estimate how much of the incoming light lies within the infrared spectrum using machine learning architectures?
- Which architectures are suitable for the task?

The most promising model architecture for predicting q_{IR} is the multilayer perceptron, where the regression-by-classification MLP reached an overall accuracy of 81.2% and the pure regression MLP achieved good scores across all metrics ($RMSE = 0.072$, $MAE = 0.056$, $R^2 = 0.883$). The LGBM model showed more lackluster results ($RMSE = 0.137$, $MAE = 0.117$, $R^2 = 0.592$). As a result, it might not be a suitable architecture. Furthermore, the two regression models failed to meet the acceptance criteria, although the RMLP made only correct predictions for two of four q_{IR} intervals as well as reaching 83% accuracy for predictions on a third interval. It should be noted that the test set was likely not entirely representative of a real camera installation and an extra hard case for the models, which all showed varying degrees of bias and overfitting. With more training data and further tuning of hyperparameters, these models could possibly perform better. In conclusion, the overall results suggest that there is potential for a machine learning approach for predicting q_{IR} . The models developed in this thesis are low in both training and inference time as well as leaving a small memory footprint. They are not yet ready for production and their results should be seen as indicative rather than definitive.

5.2 Future work

The two most critical next steps is acquiring more training data as well as further optimizing the data preprocessing. This includes more data collection, crafting better feature representations and exploring more sophisticated augmentation techniques.

An approach that may prove to be better suited for the task at hand is online learning. It is an approach in which data becomes available sequentially. The cameras could be deployed with baseline models trained on data from the imaging labs (like the models developed in this thesis) and for each day/night-switch that occurs, a new data sample becomes available for the models to retrain on. This approach is integrity- and privacy preserving by keeping the data locally on the cameras and just adjusting the local model parameters according to the intensity distributions of its surveilled scene. We already have small models with low inference times, which could be utilized for local training on the devices. For multi-camera installations, the dataset could possibly be even further expanded by leveraging distributed learning, in which a machine learning model is collectively trained by weights sharing between the cameras.

Another potential approach is to draw inspiration from the field of optimal transport and find a mapping from the r and b distributions of F_{off} to the corresponding distributions F_{on} , as well as between

the exposure values w_{off} and w_{on} . This would be similar to modeling the IR-cut filter and its impact on the light captured by the sensor.

References

- [1] F. Lv, Y. Zheng, Y. Li, and F. Lu. “An integrated enhancement solution for 24-hour colorful imaging”. *Proceedings of the AAAI Conference on Artificial Intelligence* **34**:07 (2020), pp. 11725–11732. DOI: [10.1609/aaai.v34i07.6843](https://doi.org/10.1609/aaai.v34i07.6843).
- [2] S. S. Andrews. *Electromagnetic Waves*. Light and Waves: A Conceptual Exploration of Physics. Springer International Publishing, Cham, 2023, p. 305. ISBN: 978-3-031-24096-6. DOI: [10.1007/978-3-031-24097-3_11](https://doi.org/10.1007/978-3-031-24097-3_11).
- [3] V. Dutta Gupta, G. Areendran, K. Raj, S. Ghosh, S. Dutta, and M. Sahana. “Assessing habitat suitability of leopards (*panthera pardus*) in unprotected scrublands of bera, rajasthan, india”. In: P. Kumar Shit et al. (Eds.). *Forest Resources Resilience and Conflicts*. Elsevier, 2021, pp. 329–342. ISBN: 978-0-12-822931-6. DOI: [10.1016/B978-0-12-822931-6.00026-5](https://doi.org/10.1016/B978-0-12-822931-6.00026-5). URL: <https://www.sciencedirect.com/science/article/pii/B9780128229316000265>.
- [4] L. Wang, Y. Duan, L. Zhang, T. U. Rehman, D. Ma, and J. Jin. “Precise estimation of ndvi with a simple nir sensitive rgb camera and machine learning methods for corn plants”. en. *Sensors* **20**:1111 (2020), p. 3208. ISSN: 1424-8220. DOI: [10.3390/s20113208](https://doi.org/10.3390/s20113208).
- [5] E. Reinhard, E. Khan, A. Akyuz, and G. Johnson. *Color Imaging: Fundamentals and Applications*. CRC Press, 2008. ISBN: 9781439865200.
- [6] C. Poynton. “25 - the cie system of colorimetry”. In: *Digital Video and HD (Second Edition)*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, Boston, 2012, pp. 265–286. ISBN: 978-0-12-391926-7. DOI: [10.1016/B978-0-12-391926-7.50025-4](https://doi.org/10.1016/B978-0-12-391926-7.50025-4).
- [7] D. Barolet, F. Christiaens, and M. R. Hamblin. “Infrared and skin: friend or foe”. eng. *Journal of Photochemistry and Photobiology. B, Biology* **155** (2016), pp. 78–85. ISSN: 1873-2682. DOI: [10.1016/j.jphotobiol.2015.12.014](https://doi.org/10.1016/j.jphotobiol.2015.12.014).
- [8] K. A. G. Smet. “Tutorial: the luxpy python toolbox for lighting and color science”. *LEUKOS* **16**:3 (2020), pp. 179–201. ISSN: 1550-2724. DOI: [10.1080/15502724.2018.1518717](https://doi.org/10.1080/15502724.2018.1518717).
- [9] Y. Kim, M. Ikeda, and K. Asada. “Sensitivity of cmos image sensor and scaling”. *IEICE Technical Committee on VLSI Design Technologies* (2006), pp. 2–6.
- [10] C. Poynton. “3 - linear-light and perceptual uniformity”. In: *Digital Video and HD (Second Edition)*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, Boston, 2012, pp. 27–36. ISBN: 978-0-12-391926-7. DOI: [10.1016/B978-0-12-391926-7.50003-5](https://doi.org/10.1016/B978-0-12-391926-7.50003-5).
- [11] S.-O. Shim, M. H. Alkinani, L. Hussain, and W. Aziz. “Feature ranking importance from multimodal radiomic texture features using machine learning paradigm: a biomarker to predict the lung cancer”. *Big Data Research* **29** (2022), p. 100331. ISSN: 2214-5796. DOI: <https://doi.org/10.1016/j.bdr.2022.100331>.
- [12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [13] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into deep learning*. 2023. arXiv: [2106.11342](https://arxiv.org/abs/2106.11342) [cs.LG]. URL: <https://arxiv.org/abs/2106.11342>.
- [14] V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Omnipress, Haifa, Israel, 2010, pp. 807–814. ISBN: 9781605589077.

- [15] J. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza, E. A. Chavez-Urbiola, and J. A. Romero-Gonzalez. *Loss functions and metrics in deep learning*. 2024. arXiv: 2307.02694 [cs.LG]. URL: <https://arxiv.org/abs/2307.02694>.
- [16] L. Dillén. *Feature-aware point transformer for point cloud alignment classification: pose your pose to fact*. 2023.
- [17] Z. Xiong, A. Jonnarth, A. Eldesokey, J. Johnander, B. Wandt, and P.-E. Forssén. “Hingewasserstein: estimating multimodal aleatoric uncertainty in regression tasks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 3471–3480.
- [18] D. P. Kingma. “Adam: a method for stochastic optimization”. *arXiv preprint arXiv:1412.6980* (2014).
- [19] J. Faouzi and O. Colliot. “Classic machine learning methods”. en. In: O. Colliot (Ed.). *Machine Learning for Brain Disorders*. Springer US, New York, NY, 2023, pp. 25–75. ISBN: 978-1-07-163195-9. DOI: 10.1007/978-1-0716-3195-9_2. URL: https://doi.org/10.1007/978-1-0716-3195-9_2.
- [20] X. Ying. “An overview of overfitting and its solutions”. In: *Journal of physics: Conference series*. Vol. 1168. IOP Publishing. 2019, p. 022022.
- [21] G. Hinton. “Improving neural networks by preventing co-adaptation of feature detectors”. *arXiv preprint arXiv:1207.0580* (2012).
- [22] S. Ioffe. “Batch normalization: accelerating deep network training by reducing internal covariate shift”. *arXiv preprint arXiv:1502.03167* (2015).
- [23] C. Shorten and T. M. Khoshgoftaar. “A survey on image data augmentation for deep learning”. *Journal of big data* 6:1 (2019), pp. 1–48.
- [24] J. Fang, C. Tang, Q. Cui, F. Zhu, L. Li, J. Zhou, and W. Zhu. “Semi-supervised learning with data augmentation for tabular data”. In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. CIKM ’22. Association for Computing Machinery, Atlanta, GA, USA, 2022, pp. 3928–3932. ISBN: 9781450392365. DOI: 10.1145/3511808.3557699. URL: <https://doi.org/10.1145/3511808.3557699>.
- [25] J. H. Friedman. “Greedy function approximation: a gradient boosting machine”. *The Annals of Statistics* 29:5 (2001), pp. 1189–1232. ISSN: 00905364, 21688966.
- [26] Y. Coadou. “Boosted decision trees”. In: *Artificial Intelligence for High Energy Physics*. WORLD SCIENTIFIC, 2022, pp. 9–58. ISBN: 9789811234033. DOI: 10.1142/9789811234033_0002. URL: http://dx.doi.org/10.1142/9789811234033_0002.
- [27] X. Jia, C. Zhu, M. Li, W. Tang, and W. Zhou. “Llvp: a visible-infrared paired dataset for low-light vision”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3496–3504.
- [28] H. Xu, J. Ma, Z. Le, J. Jiang, and X. Guo. “Fusiondn: a unified densely connected network for image fusion”. In: *proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*. 2020.
- [29] M. Alizamir, S. Kim, O. Kisi, and M. Zounemat-Kermani. “A comparative study of several machine learning based non-linear regression methods in estimating solar radiation: case studies of the usa and turkey regions”. *Energy* 197 (2020), p. 117239. ISSN: 0360-5442. DOI: 10.1016/j.energy.2020.117239.
- [30] E. Kavuncuoğlu, N. Çetin, B. Yildirim, M. Nadimi, and J. Paliwal. “Exploration of machine learning algorithms for ph and moisture estimation in apples using vis-nir imaging”. en. *Applied Sciences* 13:1414 (2023), p. 8391. ISSN: 2076-3417. DOI: 10.3390/app13148391.
- [31] Y. Xiong, C. McCarthy, J. Humpal, and C. Percy. “Non-visual common root rot disease detection using nir spectrum and machine learning methods”. In: *2023 11th International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*. 2023, pp. 1–6. DOI: 10.1109/Agro-Geoinformatics59224.2023.10233631.
- [32] F. Chollet et al. *Keras*. <https://keras.io>. 2015.
- [33] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. “Lightgbm: a highly efficient gradient boosting decision tree”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

- [34] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. “Optuna: a next-generation hyperparameter optimization framework”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.
- [35] S. Uddin, A. Khan, M. E. Hossain, and M. A. Moni. “Comparing different supervised machine learning algorithms for disease prediction”. *BMC Medical Informatics and Decision Making* **19**:1 (2019), p. 281. ISSN: 1472-6947. DOI: [10.1186/s12911-019-1004-8](https://doi.org/10.1186/s12911-019-1004-8).
- [36] D. Freedman and P. Diaconis. “On the histogram as a density estimator: I2 theory”. en. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* **57**:4 (1981), pp. 453–476. ISSN: 1432-2064. DOI: [10.1007/BF01025868](https://doi.org/10.1007/BF01025868).

Master's Theses in Mathematical Sciences 2025:E10

ISSN 1404-6342

LUTFMA-3569-2025

Mathematics

Centre for Mathematical Sciences

Lund University

Box 118, SE-221 00 Lund, Sweden

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