UNIVERSITÄT BERN

BACHELOR'S THESIS

Gradient boosting with explainable wavelength selection for learned spectral decoloring in optoacoustic imaging

Fabio Matti

supervised by
Prof. Martin Frenz
Dr. Thomas Kirchner

June 7, 2021

ABSTRACT

Learned spectral decoloring (LSD) encompasses the use of supervised learning (SL) to determine the blood oxygen saturation (sO_2) from multispectral optoacoustic (OA) signals originating from blood vessels. Hardware limitations and large sets of training data demand for efficient and accurate SL regressors. We investigate histogram-based gradient boosting (HGB) regressors for use in LSD, and propose a new approach to selecting ideal illumination wavelengths in an explainable way. In the course of this thesis, we are able to demonstrate that HGB regressors make predictions with an accuracy comparable to regressors previously used in LSD, while reducing the use of computational resources by an order of magnitude. Furthermore, we present a method for explainable wavelength selection which allows the number of illumination wavelengths to be decreased from 16 to 8 or even to 6 without significant consequences on the prediction accuracy of the regressor.

ZUSAMMENFASSUNG

Gelerntes spektrales Entfärben umfasst die Anwendung von überwachtem Lernen zur Bestimmung der Sauerstoffsättigung im Blut aus multispektralen optoakustischen Signalen. Aufgrund gerätetechnischer Eischränkungen und um die Trainingsdatenmenge zu erhöhen ist es von grossem Interesse, effiziente und präzise Regressoren zur Verfügung zu haben. Wir unteruchen die Tauglichkeit von histogrammbasierten "Gradient Boosting" Regressoren, und stellen eine neue Methode zur erklärbaren Auswahl von optimalen Beleuchtungswellenlängen vor. Im Laufe dieser Arbeit demonstrieren wir, dass die histogrammbasierten "Gradient Booster" in der Präzision ihrer Vorhersagen zu herkömmlichen Regressoren vergleichbar sind, jedoch viel effizienter mit den verfügbaren Resourcen umgehen. Ausserdem zeigen wir, dass es unsere erklärbare Methode um Beleuchtungswellenlängen auszuwählen erlaubt, die Anzahl solcher Wellenlängen ohne merkbaren Einfluss auf die Präzision der Vorhersagen von 16 auf 8 oder sogar auf 6 zu reduzieren.

Contents

ACRONYMS

ALE accumulated local effects

API application programming interface

bvf blood volume fraction

Hb deoxygenated hemoglobin

HbO_2 oxygenated hemoglobin

HGB histogram-based gradient boosting

LED light emitting diode

LSD learned spectral decoloring

MI-LSD multiple Illumination learned spectral decoloring

MPD mean prediction difference

MRI magnetic resonance imaging

NIR near-infrared

NN neural network

OA optoacoustic

PD partial dependence

qOAI quantitative optoacoustic imaging

rCu relative copper quantity

RF random forest

s_aO_2 blood oxygen saturation in arterial blood

SL supervised learning

sO_2 blood oxygen saturation

SOA state of the art

svf sulfate volume fraction

TV total variation

US ultrasound

1 INTRODUCTION

Non-invasive techniques of determining the blood oxygenation saturation (sO_2) in human tissue are of great importance to many clinical applications. For example oxygen deficiency in tumors was identified to be one of the leading negative impacts on the effectiveness of radiotherapy, as it enhances the tumors resistance to radiation [1]. Tracking the time evolution of the sO_2 in tumors and personalizing the treatment accordingly would therefore be of high interest [2]. A second very promising field of application may be the detection of cerebral ischemia (restricted blood-flow to the brain) in preterm infants [3]. Their low compliance combined with the risk of separating the infants from the incubator for magnetic resonance imaging (MRI), and the low spacial resolution of other imaging technologies demand for different approaches.

An emerging method to quantitatively estimate the sO_2 is optoacoustic (OA) imaging, which stands out due to its utilization of the strong optical absorption contrast in blood and the high spatial resolution it provides [4]. OA is a light in – sound out approach, which makes use of the OA effect. There, light absorption gives rise to a pressure gradient and consequently the release of acoustic waves from the affected absorber [5]. The intensity of the ultrasound (US) signal is directly proportional to the absorbed light energy, which depends on both material properties of the absorber, predominantly the absorption coefficient, as well as the light fluence through the absorber. Ideally, knowing both the light fluence through an absorber and the resulting US signal would then allow us to draw conclusions about the composition of blood. Differing absorption spectra of oxygenated (HbO_2) and deoxygenated (Hb) hemoglobin would allow us to estimate the sO_2 with high accuracy [4]. In reality, however, light is subject to the effects of scattering and absorption in the surrounding tissue, which is even further complicated by the inhomogeneous nature of human tissue. An analytical solution to the problem is therefore not feasible.

Light absorption in human tissue is at a minimum for wavelengths in the near-infrared (NIR) region, allowing light to reach deep layers of the tissue. A prerequisite for US waves to be emitted by an absorber is a very short-timed illumination with duration of just a few nanoseconds. As a consequence, only pulsed lasers and special types of fast light emitting diodes (LED) are suited for use in OA imaging [5, 6].

Improved laser technologies and US detection systems allow for the acquisition of large portions of data in short time intervals. Due to noise afflicted measurements as well as the inhomogeneous and inconsistent layout of human tissue, a data-driven approach to tackle the problem of estimating the sO_2 seems very appropriate. Previous approaches made use of linear spectral unmixing [7]. This, however, ignores fluence effects, making it inherently inaccurate. Recently, network- [8, 9] and tree-based [10, 11] supervised learning models were shown to be more suitable for this task. The application of the latter algorithms to OA imaging is termed learned spectral decoloring (LSD).

In order to validate new measurement techniques, it is not advisable to directly do so *in vivo*, because a reference measurement with a proven and accurate technol-

ogy would be needed, which, if it existed, would defeat the purpose of optoacoustic imaging. Phantom models which simulate blood flow through a vessel may be used for this purpose, but are both hard to tune and limited in accuracy [11]. A way of bypassing these issues is by mimicking sO_2 with a relative copper sulfate model, where the combination of copper sulfate ($CuSO_4$) and nickel sulfate ($NiSO_4$) water solutions shows similar absorption properties as HbO_2 and Hb do.

In our work, we first concerned ourselves with a promising class of tree-based regressors, called histogram-based gradient boosting (HGB) regressors, with the goal of evaluating their suitability for the use in OA imaging. These regressors are widely used and have a reputation for being both accurate and fast [12, 13, 14]. In a second part, we then develop an objective and explainable method for selecting ideal wavelengths for illumination. This method is based on the concept of accumulated local effects (ALE), which helps determining the importance an illumination wavelength holds in the prediction process of a regressor [15]. We will test the efficacy of our method by tracking the progression of the absolute prediction errors of a regressor when provided with ever smaller subsets of the initially available wavelength information. One specifically picked according to our method of evaluating the "importance" of a wavelength, and one picked uniformally, as is the current state of the art (SOA) [16].

This thesis starts with giving a theoretical overview of the OA effect and OA imaging, followed by the basics of supervised learning and LSD in section 2. It then moves on to introduce the data sets and software we used in section 3, and details the evaluation process of the four HGB regressors we analyzed as well as explaining our explainable method for wavelength selection in section 4. The results (section 5) and a thorough discussion of them (section 6) follow, before we conclude this thesis in section 7.

2 THEORY

The ultimate goal of our research is to determine the blood oxygen saturation (sO_2) in deep blood vessels of the human tissue.

The sO_2 is given by the relative concentrations of oxygenated $c(HbO_2)$ and deoxygenated $c(Hb)$ hemoglobin [11]:

$$sO_2 = \frac{c(HbO_2)}{c(HbO_2) + c(Hb)} \tag{2.1}$$

Pulse oximeters can, with an expected error of down to 2% under ideal circumstances[1], measure the average oxygen saturation in the arterial blood (s_aO_2) in fingertips and earlobes by comparing the light absorption for two different wavelengths when shone through these body parts [18]. They make use of the fact that both fingertips and earlobes are rather thin and optically translucent. To see this, one can simply use their smartphone flashlight and try covering it with one of their fingers; A small portion of the light will always manage to pass through the finger, particularly at higher wavelengths (red), where the transmittance seems to be comparatively large (we will make use of this observation in section 2.2).

However, the average s_aO_2 in a fingertip or an earlobe is not representative for other parts of the body, most noticeably the brain and tumors (which, as discussed in section 1, would be of particular interest to know the sO_2 for). Furthermore, the aforementioned parts are usually not as optically translucent as fingertips and earlobes are: Already when covering your smartphone's flashlight with your palm, no visible light will be able to reach the other side. This, paired with the fact that neither the human brain nor a tumor is sufficiently homogeneous to be described by an average s_aO_2, forces us to find a different approach to estimating the sO_2.

2.1 THE OPTOACOUSTIC EFFECT

In 1880, Alexander Graham Bell discovered that upon pulsed illumination with light, certain materials start emitting sounds [19]. This corresponds to the first known observation of the optoacoustic (OA) effect.

The effect is caused by a temperature rise and a consequent thermoelastic expansion following the absorption of light. The resulting mechanical stress will dissipate with the speed of sound into the surrounding medium. Very short-pulsed light (nanoseconds) is able to deposit all its energy before the stress starts dissipating, therefore enabling stress-confinement [20].

2.2 OPTOACOUSTIC IMAGING

The OA effect can be used to image human tissue.

Quantitative optoacoustic imaging (qOAI) makes use of the varying absorption properties of different components in the blood, most notably oxygenated (HbO_2) and deoxygenated (Hb) hemoglobin [21]. Of particular interest is light in the near-infrared (NIR) regime, which is also referred to as the diagnostic window. Light at

[1] This error is significantly higher in patients with high concentrations of melanin (darker skin tones) [17].

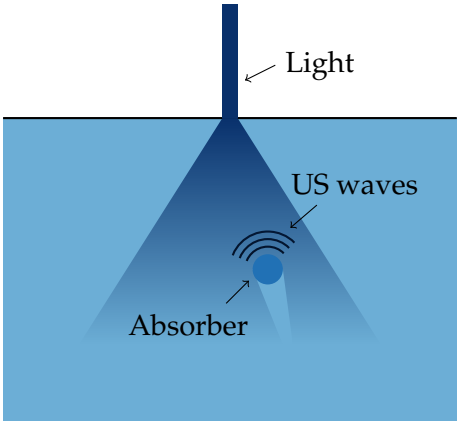


FIGURE 2.1 – When short-pulsed light is absorbed in certain materials, a thermoelastic expansion under stress-confinement causes ultrasound (US) waves to be emitted. This effect is called the optoacoustic effect.

these wavelengths can penetrate much deep into tissue [5]. For instance light from a smartphone flashlight covered by a finger appears red, meaning predominantly the red component in white light was transmitted. The absorption behavior of both Hb and HbO_2 are characterized by the absorption coefficient μ_a. In the NIR region, these coefficients wavelength-dependent coefficients form distinct curves (see figure 2.2).

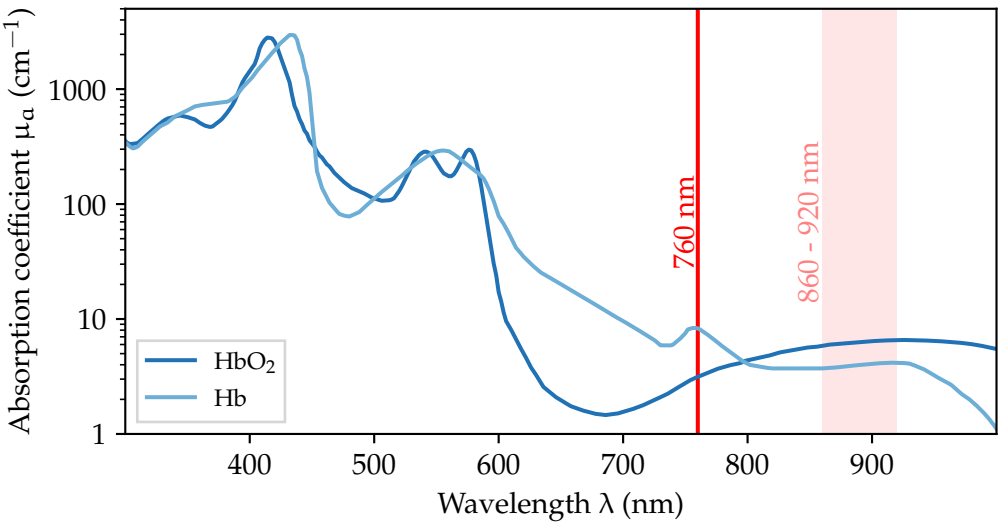


FIGURE 2.2 – The wavelength-dependent absorption coefficients μ_a for both HbO_2 and Hb. Some interesting wavelengths are highlighted in red, and will be discussed in section 6. A whole blood total Hb concentration of 150 g/l was assumed. Data provided by https://omlc.org/spectra/hemoglobin/summary.html.

Theoretically, the increase in pressure $\Delta p(\mathbf{z})$ at a position $\mathbf{z}$ in the tissue, upon

incidence of light is given by [22]

$$\Delta p(\mathbf{z}) = \Gamma \cdot H(\mathbf{z}) \tag{2.2}$$

Here, $H(\mathbf{z})$ is the absorbed energy density and Γ a proportionality factor; the Grüneisen parameter. This parameter is determined by the coefficient of thermal expansion β, the speed of sound c_s in the medium, and the heat capacity C_p:

$$\Gamma = \frac{\beta c_s^2}{C_p} \tag{2.3}$$

The US signal S resulting from the pressure gradient is characterized by the following proportionality as a consequence of (2.2) [5]:

$$S \propto \Delta p(\mathbf{z}) \propto H(\mathbf{z}) \tag{2.4}$$

$H(\mathbf{z})$, in turn, depends on the fluence $\phi(\mathbf{z})$ through the position $\mathbf{z}$:

$$H(\mathbf{z}) = \mu_a(\mathbf{z}) \cdot \phi(\mathbf{z}) \tag{2.5}$$

All building blocks for successfully determining the absorption coefficient $\mu_a(\mathbf{z})$ at a position $\mathbf{z}$ are now assembled. In practice, however, things are a lot more complicated. The fluence $\phi(\mathbf{z})$ and the US signal S depend on macroscopic structures in tissue, which, in general, prove to be difficult to account for, particularly when the exact dimensions and absorption properties of these structures are only vaguely known. What remains is a chaining of two ill-posed inverse problems: The optical inverse problem, which encompasses determining the absorption properties of the tissue from the fluence $\phi(\mathbf{z})$, and the acoustic inverse problem which aims to reconstruct the pressure-rise distribution $\Delta p(\mathbf{z})$ from the measured US signals S. The two problems are illustrated in figure 2.3.

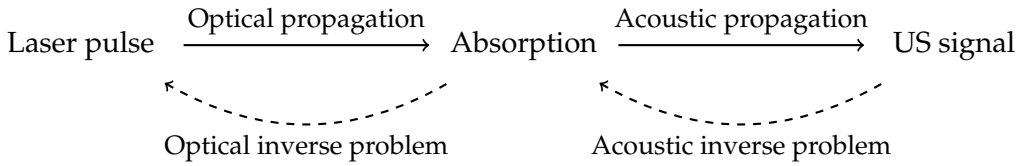


FIGURE 2.3 – The goal of optoacoustic (OA) imaging is to reconstruct the absorption for a blood vessel, and thus, obtaining information about the sO_2. Based on an US signal, the propagation of US waves in tissue must be reversed, posing the acoustic inverse problem. The absorption itself, however, depends on the flux of light through the absorber, and therefore the propagation of light in the tissue. Reverting this optical propagation is called the optical inverse problem.

Now, we can make use of the wavelength dependent absorption properties of both Hb and HbO_2 (see figure 2.2). If we successfully measure the absorbed energy density $H(\lambda, \mathbf{z})$ and the fluence $\phi(\lambda, \mathbf{z})$ for multiple wavelengths λ at a fixed position $\mathbf{z}$ we could draw conclusions about the composition of blood at $\mathbf{z}$. Information about $H(\lambda, \mathbf{z})$ can be obtained with an US detector from the wavelength

dependent signal $S(\lambda, \mathbf{z})$ (as shown in equation (2.4)). A non-invasive measurement of the fluence $\phi(\lambda, \mathbf{z})$ is not trivial to obtain. However, the propagation of light in representative models of human tissue may be simulated *in silico* using optical forward Monte Carlo simulations [23]. In fact, these simulations can then directly be used to train a supervised learning SL regressor for predicting the sO_2 based on the absorbed energy density $H(\lambda, \mathbf{z})$ for multiple wavelengths λ.

2.3 SUPERVISED LEARNING

Given a sample $\mathbf{x}$ (an example, an experiment, or an observation), which is characterized by M features $x_1, x_2, \ldots, x_M$ (measurements, traits, or properties), the goal of supervised learning (SL) is to predict a label y (a "target" feature) for this sample:

$$\mathbf{x} = (x_1, x_2, \ldots, x_M) \to y \tag{2.6}$$

This can be done by training a prediction function $p : \mathbb{R}^M \to \mathbb{R}$, which maps M features to a label. This prediction function is trained using a training set $\{\mathbf{x}, y\}_{train}$, which consists of multiple samples with their corresponding labels. For a fresh sample $\mathbf{x}$, for which we would like to determine its label y (which we might not know yet), our prediction function p can then make a prediction $\overline{y}$ for this unknown label y:

$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_M) = \overline{y} \tag{2.7}$$

In her career, a car dealer may see hundreds of used cars (samples) being bought and sold at prices y. This car dealer probably realizes, that for similar types of cars, particularly the mileage x_1 and horse power x_2 (features) seem to be indicative, of how much a car is worth. The higher the mileage x_1, the more worn down and prone to defects a car usually is, and therefore, the price y will typically decrease with rising mileage. On the contrary, a higher horse power x_2 usually results in a higher selling price. Throughout her career, her prediction function p (her brain) usually gets better at predicting a car's price y, based on these two and probably other less quantitative factors. It is continuously trained on training data $\{\mathbf{x}, y\}_{train}$ of cars which she sees being bought or sold.

One of the most fundamental way of training such a prediction function p computationally is done by constructing a so-called decision tree [24]. A reasonable prediction to make, based on a given training set $\{\mathbf{x}, y\}_{train}$ would be the arithmetic mean of the training labels y_{train}. Like this, the mean absolute deviation of the training labels y_{train} from this prediction would be rather small. We can improve upon this rough estimate, by splitting the set into two parts, such that the mean absolute deviations of the training labels y_{train} from their corresponding predictions, now given by the two arithmetic means for the labels y_{train} in the respective subsets, is as small as possible. To find the best split (i.e. the split for which the mean absolute deviation of the training labels y_{train} from the mean label in their group is smallest), we would have to search through all the ways of assigning samples

into two groups. However, there are too many possible combinations[2] we would have to consider, and later on we will only be given features to make a prediction for without labels anyway. Nevertheless, we can already find a "good" split by only considering assignments of samples to the two groups according to a threshold value for a feature. The feature for which the threshold value best separates the training set into two group, one consisting of the samples for which this feature is larger than the threshold value, and the other of the ones with smaller values for this feature. We can then employ the exact same procedure on these two groups, to get four groups and even better predictions, and repeat this procedure until a stopping criterion is met.

In the end, our prediction function p (which can be visually thought of as a tree, compare figure 2.4) is just a set of feature indices and threshold values, which tell us to whether a sample is to be moved left or right in the tree. Recall that the prediction is still just the mean of the training samples, which happened to remain in a branch ending when the stopping criterion was met.

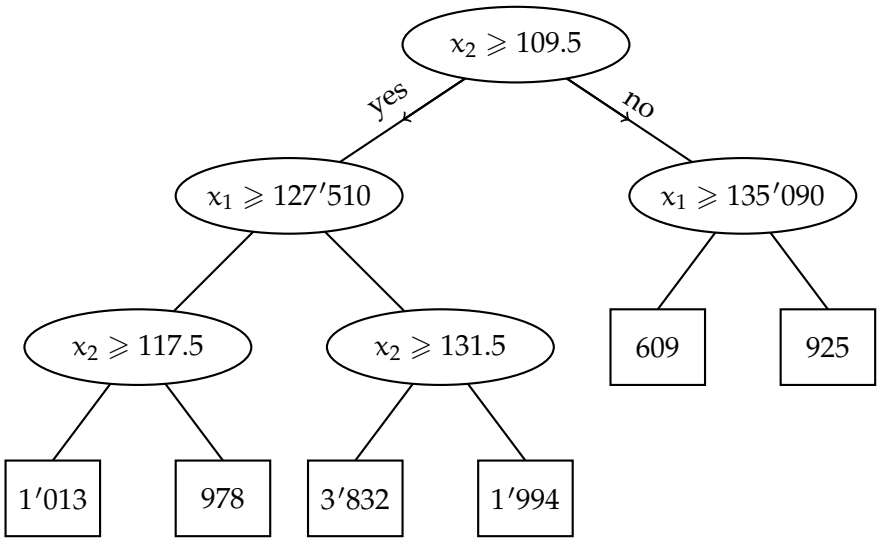


FIGURE 2.4 – Example of a small decision tree (although decision root would be more appropriate for this visual representation), which predicts prices y of similar cars, based on mileage x_1 and power x_2. For a car with a mileage of $x_1 = 110'000$ km and power $x_2 = 110$ hp, a prediction of 1'994.- would result for the price.

Decision trees by themselves are not accurate enough for most purposes. However, it is possible to "pool" multiple decision trees together, such that we do not have to rely on the predictions of a single decision tree. One such method is called random forest RF. Building multiple decision trees with the exact same data set would lead to these trees being identical copies of each other, and would automatically lead to all these trees making the exact same predictions. By reducing the amount of data we provide each of the decision trees with, for instance by pick-

[2]In fact, there are $2^{N-1} - 1$ ways of assigning N samples to two non-empty groups, if order does not matter. Even assuming we have a tiny training set of $N = 100$ samples, we would have to search through a vast 10^{30} combinations to find the best split.

ing a random subset of the features or the samples, the trees will all be built in a unique way, based on what subset of the data they received. Every tree will then make its own prediction, allowing us to take the average of the predictions made by each tree. Another approach to combining multiple decision trees to get a more powerful regressor is boosting. In boosting, trees are sequentially built, based on the predictions the previous trees made. For instance, each tree could be trained to learn the errors the previous tree makes, and try to correct them. We will come back to this technique, when we have a look at histogram-based gradient boosting (HGB) regressors in section 4.1.

2.4 LEARNED SPECTRAL DECOLORING

Learned spectral decoloring (LSD) is a data-driven approach to recover the sO_2, the label, from the wavelength-dependent absorbed energy density $H(\lambda, \mathbf{z})$ at a position $\mathbf{z}$. The features which we use for supervised learning are the absorbed energy density $H(\lambda, \mathbf{z})$ for multiple wavelengths λ (see figure 2.5). LSD involves training a supervised learning (SL) regressor on simulated training data, usually generated with Monte Carlo optical forward simulations. It was shown to be highly effective on *in silico* (i.e. computationally simulated) data, as well as on *in vitro* (i.e. phantom models) [5, 8, 9, 11].

US signals from different depths in the tissue are, in general, going to vary in strength. However, we do not really care about the absolute strength of the signals, because we are only interested in the relative changes in the absorbed energy density $H(\lambda, \mathbf{z})$, when different wavelengths λ are used for illumination. Therefore, the proportionality in (2.4) allows us to restrict ourselves to work with the L1 normalized quantities $\hat{S}(\lambda, \mathbf{z})$ and $\hat{H}(\lambda, \mathbf{z})$. For wavelengths $\lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_M\}$, the L1 normalized quantities are given by

$$\hat{H}(\lambda_i, \mathbf{z}) = \frac{H(\lambda_i, \mathbf{z})}{\sum_{j=1}^{M} H(\lambda_j, \mathbf{z})} \quad \text{and} \quad \hat{S}(\lambda_i, \mathbf{z}) = \frac{S(\lambda_i, \mathbf{z})}{\sum_{j=1}^{M} S(\lambda_j, \mathbf{z})} \tag{2.8}$$

Due to their proportionality, they satisfy the following relation

$$S(\lambda, \mathbf{z}) \propto H(\lambda, \mathbf{z}) \implies \hat{S}(\lambda, \mathbf{z}) = \hat{H}(\lambda, \mathbf{z}) \tag{2.9}$$

Recently, experiments which explicitly measure the absorbed energy density $H(i, \lambda, \mathbf{z})$ for multiple different illumination positions i have been shown to be more effective than conventional LSD [11]. Now, the regressor is trained with the L1 normalized absorbed energy density $\hat{H}(i, \lambda, \mathbf{z})$ (figure 2.6). The normalization is calculated seperately for each of the illumination positions. This approach is referred to as multiple illumination learned spectral decoloring MI-LSD.

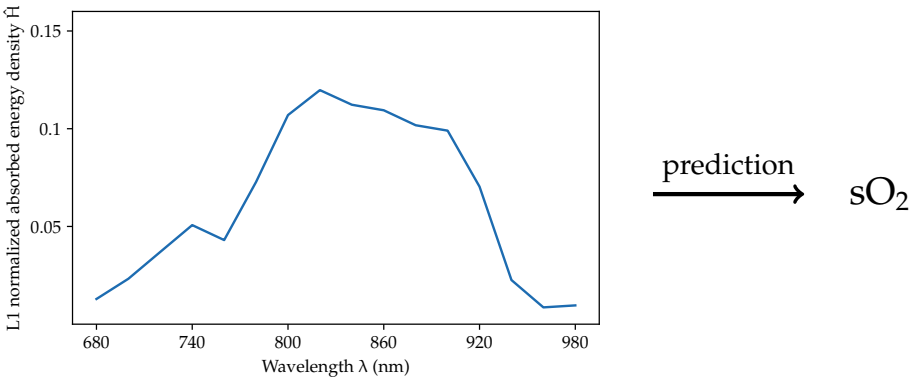


FIGURE 2.5 – The goal of learned spectral decoloring (LSD) is to train a machine learning regressor to predict the blood oxygen saturation (sO_2) based on the L1 normalized absorbed energy density $\hat{H}$.

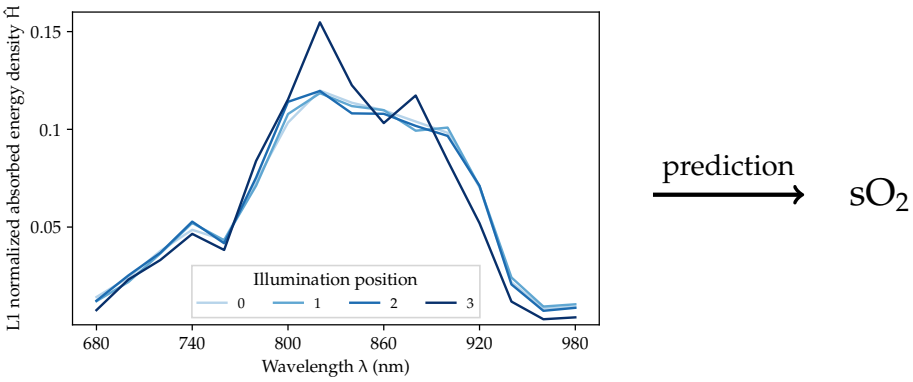


FIGURE 2.6 – In multiple illumination learned spectral decoloring MI-LSD, the regressor is trained to predict the sO_2 based on the L1 normalized absorbed energy densities $\hat{H}$ for multiple illumination positions. Here, four illumination positions were used.

3 MATERIALS

3.1 DATA SETS

Both the relative copper sulfate model (*in silico* and *in vitro*) and the sO_2 (only *in silico*) are examined.

The relative copper sulfate model is used for its capability to mimic the absorption properties of HbO_2 and Hb. The relative copper model consists of a nickel sulfate ($NiSO_4$) and copper sulfate ($CuSO_4$) water solution solution. In analogy to sO_2 (2.1), the relative copper (rCu) in this model is given by

$$\mathrm{rCu} = \frac{c_r(CuSO_4)}{c_r(CuSO_4) + c_r(NiSO_4)} \tag{3.1}$$

with relative concentrations $c_r(\cdot) = c(\cdot)/c_{wb}(\cdot)$ of the concentrations $c(\cdot)$ to their blood mimicking base solutions [11].

All our experiments for the rCu are based on identical data sets to the ones used in [11]. To summarize: They all document the absorbed energy densities $H(i, \lambda, \mathbf{z})$ for 16 uniformly spaced illumination wavelengths λ, separated by 20 nm from 680 nm to 980 nm, and for four different illumination positions i. To convert the multiple illumination data sets for LSD, the averaged absorbed energy density over all illumination positions is taken to get $H(\lambda, \mathbf{z})$. Before passing the absorbed energy density ($H(\lambda, \mathbf{z})$ in LSD or $H(i, \lambda, \mathbf{z})$ in MI-LSD) to the regressor, they are L1 normalized as described in section 2.4.

The data sets can be categorized into three different types of tube-layouts (see figure 3.1): *In silico*, Phantom B, and Phantom C. The *in silico* data sets are derived from Monte Carlo simulations of light propagation in randomized volumes. Each volume consists of two sets of between 3 and 9 tubes and each set with an rCu level randomly drawn from a uniformal distribution $U([0, 1])$. The background sulfate volume fraction svf = $c_r(CuSO_4) + c_r(NiSO_4)$ (in analogy to the blood volume fraction bvf) is randomly picked from $U([0, 3])$. For each layout 10^8 photons were used for simulation. For the training sets, a total of 4000 different volumes were simulated, and for the *in silico* validation set, 1000.

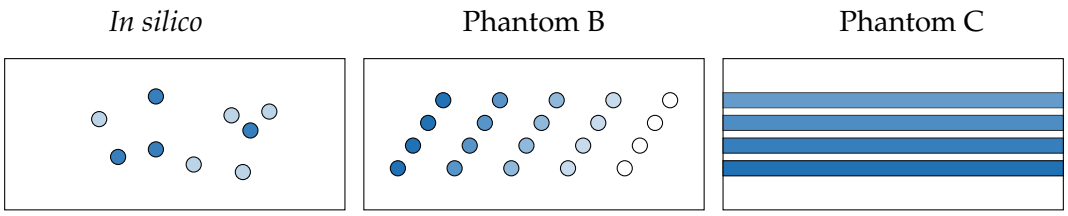


FIGURE 3.1 – Schematic representations for the three different volume configurations of the test data sets. Darker colors correspond to higher rCu values.

For the *in silico* sO_2 data sets, a similar procedure was employed. Once again 4000 and 1000 volumes were simulated for the training set and the *in silico* test set

respectively. The general layout of the volumes were chosen identically for both rCu and sO_2 models, with sO_2 = rCu and bvf = svf.

3.2 HARDWARE AND SOFTWARE

All the computations, which directly led to the results below, were performed on a single 8th generation i7 CPU (Intel, Santa Clara, USA), supplied with 16 GB of RAM. The hyperparameter tuning described in section 4.1 was performed on the UBELIX HPC cluster of the University of Bern.

For the regressions, we used the HistGradientBoostingRegressor from the scikit-learn[3] (v0.24.1), the XGBRegressor from the xgboost[4] (v1.3.3), the LGBMRegressor from the lightgbm[5] (v3.1.1), and the CatBoostRegressor from the CatBoost[6] (v0.24.4) package for Python[7] (v3.8.8). All custom algorithms were implemented using the NumPy[8] (v1.19.2) release, and the visualizations are done with the pyplot module included in the Matplotlib[9] (v3.3.4) package.

[3]https://scikit-learn.org
[4]https://xgboost.readthedocs.io
[5]https://lightgbm.readthedocs.io
[6]https://catboost.ai
[7]https://python.org
[8]https://numpy.org
[9]https://matplotlib.org

4 METHODS

4.1 HISTOGRAM-BASED GRADIENT BOOSTING REGRESSORS

We investigate the suitability of histogram-based gradient boosting (HGB) regressors for learned spectral decoloring (LSD) and multiple illumination learned spectral decoloring (MI-LSD). To do so, we compare the performance of four different implementations of HGB regressors in contrast to previous implementations based on random forests (RFs) and neural networks (NNs) [11].

Boosting refers to the practice of combining multiple weak learners (i.e. basic and not necessarily accurate regressors, usually decision trees, see section 2.3) in such a way that the weak learners are sequentially trained based on the predictions produced by the previous learner makes [25]. In gradient boosting, each learner is trained on the negative gradient of the loss function for the previous learner [12]. When training on large data sets, weak learners usually have to go through computationally taxing operations such as sorting. First assigning the continuous feature values to a number of bins (usually 256), allows the weak learner to operate on the bins instead of individual data values, and constructing feature histograms, hence 'histogram-based' [14]. A possible additional benefit of binning the data is a reduction in the small-scale noise component in the feature values.

In recent times, gradient boosting regressors routinely win competitions[10], of which particularly the inverse problem of event reconstruction in experimental high energy physics[11] shows many similarities to the inverse problems we try to solve. Furthermore, histogram-based versions of this algorithm are orders of magnitude faster, when it comes to large scale data sets [14].

The four widely used implementations we considered are specified in section 3.2. Each algorithm may be controlled by parameters which define the circumstances under which the regressor is trained, without changing the general nature of the algorithm. These parameters are referred to as hyperparameters. We tuned the hyperparameters on the *in silico* rCu validation set. For the sake of comparability, we kept the hyperparameters as similar as possible for all the evaluated regressors. Where possible, logistic regression was preferable over mean squared and mean absolute error loss functions.

A summary of the tuned hyperparameters for the experiments, can be found in table 4.1. A visualization of how these hyperparameters influence the training of the regressors can be seen in figure 4.1.

To compare the performance of the four regressors, we use the specified hyperparameters in table 4.1 and train each regressor on the same *in silico* training data set (L1 normalized absorbed energy densities $\hat{H}(\lambda, \mathbf{z})$ in LSD or $\hat{H}(i, \lambda, \mathbf{z})$ in MI-LSD, and ground truth $\mathrm{rCu}_{\mathtt{tube}}$ or $\mathrm{sO}_{2\mathtt{tube}}$). Then we supply the trained regressor with $\hat{H}(\lambda, \mathbf{z})$ or $\hat{H}(i, \lambda, \mathbf{z})$ measured for the test volumes. The regressor then provides us with its predictions $\mathrm{rCu}_{\mathtt{pred}}$ or $\mathrm{sO}_{2\mathtt{pred}}$ for the ground truth. As a measure for the quality of these predictions, we look at the absolute prediction errors $\Delta_{\mathtt{pred}} = |\mathrm{rCu}_{\mathtt{pred}} - \mathrm{rCu}_{\mathtt{tube}}|$ (and analogously $\Delta_{\mathtt{pred}} = |\mathrm{sO}_{2\mathtt{pred}} - \mathrm{sO}_{2\mathtt{tube}}|$). We

[10] https://github.com/dmlc/xgboost/tree/master/demo
[11] https://higgsml.lal.in2p3.fr

TABLE 4.1 – Overview of the hyperparameters we tuned for each of the four regressors, as well as their default values. Missing entries are either set automatically or have no impact on the training process with this configuration.

Regressor	Hyperparameters		
	variable	default	tuned
sklearn	loss	'least_squares'	'least_absolute_deviation'
	learning_rate	0.1	0.1
	max_iter	100	300
	max_leaf_nodes	31	100
	max_depth	None	None
XGBoost	objective	'reg:squarederror'	'reg:logistic'
	eta	0.3	0.1
	n_estimators	100	300
	max_leaves	-	-
	max_depth	6	12
	tree_method	'auto'	'hist'
LightGBM	objective	'regression'	'cross_entropy'
	learning_rate	0.1	0.1
	n_estimators	100	300
	num_leaves	31	100
	max_depth	-1	-1
CatBoost	loss_function	'RMSE'	'MAE'
	learning_rate	-	0.1
	iterations	1000	300
	max_leaves	31	100
	depth	6	16
	grow_policy	'SymmetricTree'	'Lossguide'

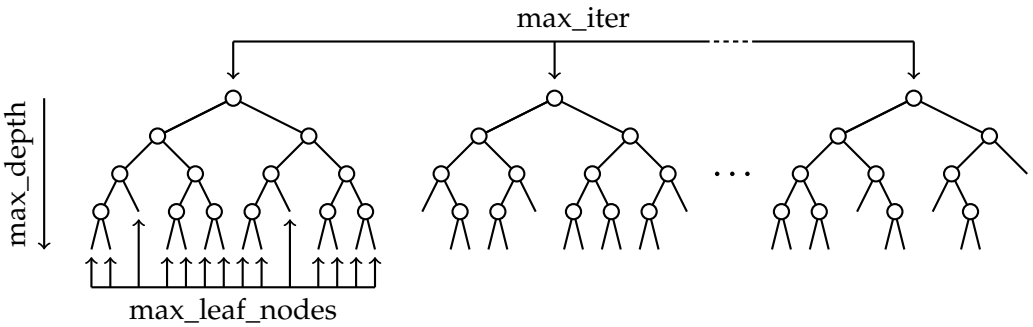


FIGURE 4.1 – A visual explanation of the what aspect of the gradient boosting regressors some of the hyperparameters in table 4.1 control. The sklearn variable convention is used (see table 4.1 for the other regressors).

then list the median Q_2 of the $\Delta_{\mathtt{pred}}$ as a robust approximation for the deviation of the predictions from the ground truth, and the 90th percentile P_{90} to quantify outliers.

We will judge the suitability of each of these implementations for LSD according to the following criteria, of which the accuracy is weighted the most:

- Accuracy – By looking at the Q_2 and P_{90} of the absolute prediction errors for the test sets specified in section 3.1.
- Agility – By the time it takes the regressor to train and make predictions, and by the memory footprint during the training stage.
- Adaptability – In the sense of both the range of applications for which the regressor can be used, and the option to taylor the regressor according to one's own needs.
- Accessibility – Regarding the installation process, user friendliness of the interface, and requirements to the system architecture.

4.2 EXPLAINABLE WAVELENGTH SELECTION

The second goal of our work is to quantitatively determine the relative importance an illumination wavelength holds in the decision process of a supervised learning (SL) algorithm while performing learned spectral decoloring (LSD) or multiple illumination learned spectral decoloring (MI-LSD). This ultimately would provide us with an explainable approach to selecting illumination wavelengths. A selection of an optimal subset of wavelengths is desirable, as it reduces data acquisition time. It can also aid in selecting fixed wavelength LEDs or laser diodes for integrated systems [26].

4.2.1 PARTIAL DEPENDENCIES

A common approach to visualizing the effect a feature holds in a supervised learning model are partial dependencies (PD), proposed in [12]. Briefly, the partial dependence function for the i-feature $PD_i(x)$ is the expected values our predictor function would give for a sample $\mathbf{x}$ for which the i-th feature takes the value x:

$$PD_i(x) = \mathbb{E}\left[p(\mathbf{x}|_{x_i=x})\right] \tag{4.1}$$

It is apparent that the partial dependence (PD) function requires the features to not be correlated. Otherwise, fixing the i-th feature at x while allowing any other (potentially highly correlated) feature to take on arbitrary values, which in reality would never be observed in combination with x as the i-th features value, will cause the predictions to be distorted.

Due to the rather smooth spectra of the absorption coefficient μ_a for both oxy-HbO_2 and deoxyhemoglobin Hb (figure 2.2), as well as in the relative copper sulfate (rCu) model [11], the absorbed energy densities $H(\lambda, \mathbf{z})$ for two neighbouring illumination wavelengths are by design going to be very similar, and thus, highly correlated. Therefore, the partial dependencies are not suitable for our problem.

4.2.2 ACCUMULATED LOCAL EFFECTS

The accumulated local effects (ALE) [15] of a predictor variable (feature) is an adaptation of the partial dependence (PD), which restricts itself to the examination of local variations in a feature's value. In practice, we need to work with a computational approximation of the theoretical ALE function. For this purpose, we first define the mean prediction difference (MPD_i) between two values $a < b$ of the i-th feature.

$$\mathrm{MPD}_i(a, b) = \underset{\{\mathbf{x}:\, x_i \in [a,b]\}}{\mathrm{mean}} [p(\mathbf{x}|_{x_i=b}) - p(\mathbf{x}|_{x_i=a})] \tag{4.2}$$

Here, $\{\mathbf{x}\}$ now refers to a set of training samples. The process of calculating this value is sketched in figure 4.2.

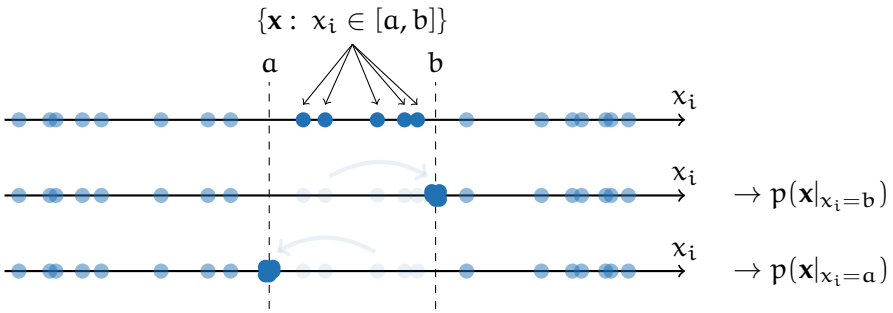


FIGURE 4.2 – The mean prediction difference (MPD) for the i-th feature on the interval $[a, b]$ is calculated as follows: First, find all the samples $\mathbf{x}$ for which the i-th feature x_i assumes values within the interval $(a, b]$. Then, replace these values with the upper interval border b, and determine the predictions $p(\mathbf{x}|_{x_i=b})$ that would result from these altered samples. Repeat this process for the lower interval border a to get $p(\mathbf{x}|_{x_i=a})$. In the end, take the mean over all samples for the differences between these two predictions.

For an approximation of the accumulated local effects ALE, we first need to define a sufficiently fine partition $\{z_k\}_{k=1}^K$ of the i-th feature's domain, where z_1 is chosen just below the lowest, and z_K is the highest attained value in this feature. The approximate function value $\mathrm{ALE}_i(x)$ of the i-th feature at x is then given by the sum of all the MPDs between consecutive points in this partition which are smaller than x.

$$\mathrm{ALE}_i(x) \approx \sum_{k:\, z_k \leqslant x} \mathrm{MPD}_i(z_k, z_{k+1}) \tag{4.3}$$

From now on (as suggested in [15]) we always take quantiles $\{\hat{q}_{(k-1)/(K-1)}\}_{k=1}^K$ of the empirical distribution function for the values of this feature as our partition, such that for every MPD the same amount of samples is considered (see figure 4.3).

For the purpose of visualizing and comparing the ALE functions of multiple features, the approximate ALE function may also be centralized in the following way:

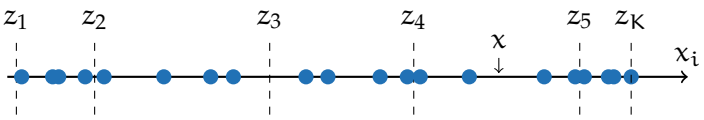


FIGURE 4.3 – In order to calculate the accumulated local effects ALE function for the i-th feature is partitioned via evenly spaced quantiles of the empirical distribution function. The accumulated local effects function for the i-th feature at a point x, $\mathrm{ALE}_i(x)$, is then given by the sum (accumulation) of the mean prediction differences MPD between all quantiles below x.

$$\overline{\mathrm{ALE}}_i(x) \approx \mathrm{ALE}_i(x) - \frac{1}{K} \cdot \sum_{k=1}^{K} \mathrm{ALE}_i(z_k) \tag{4.4}$$

To get a feeling for how the ALE function is related to the impact a feature has on the predictions, consider a feature which does not impact the predictions at all. In other words, if we change a sample's current value for this feature to any other value, the prediction our regressor would make for this sample would remain unchanged. The MPD (4.2) between any two values would be identically zero, and therefore, the ALE function (4.3) would be identically zero too. On the other hand, for features whose values have a high impact on the outcome of the predictions (i.e. small changes in their values lead to vastly different predictions) the MPD would take on rather large values (at least for some of the partitions), and the ALE function would vary a lot.

4.2.3 TOTAL VARIATION

In order to quantify the "amount of variation" the accumulated local effects ALE function exhibits, we considered the notion of the total variation (TV) of a function [27]: Let $P = \{x_l\}_{l=1}^{L}$ be a partition of the interval $[a, b]$, such that $a = x_1 < x_2 < \ldots < x_L = b$. The total variation of a function $f : [a, b] \to \mathbb{R}$ is then defined to be the supremum of the sum of differences of this function between consecutive elements in the partition P:

$$\mathrm{TV}(f) = \sup_{P} \sum_{l=1}^{L-1} |f(x_{l+1}) - f(x_l)| \tag{4.5}$$

In practice, an approximation is needed to compute this value. Because tree based regressors are, by nature, unlikely to make entirely different predictions for similar samples, particularly for data afflicted with noise, we assume that the ALE functions for each feature will be sufficiently smooth (i.e. Lipschitz continuous with a reasonably small Lipschitz constant). Therefore, we can approximate it in the following way: Let $\{x_l\}_{l=1}^{L}$ be a set of uniformly spaced points between $z_1 = x_1$ and $z_K = x_L$ (recall, that we chose z_1 to be just below the smallest and z_K to be the largest value attained by the feature we are interested in). The approximate TV of the i-th feature's ALE function is then given by

$$\mathrm{TV}(\mathrm{ALE}_i) \approx \sum_{k=1}^{L-1} |\mathrm{ALE}_i(x_{k+1}) - \mathrm{ALE}_i(x_k)| \tag{4.6}$$

A visual explanation of the calculation of this value is displayed in figure 4.4. Generally, the smaller $\mathrm{TV}(\mathrm{ALE}_i)$ the less impact this feature has on the predictions.

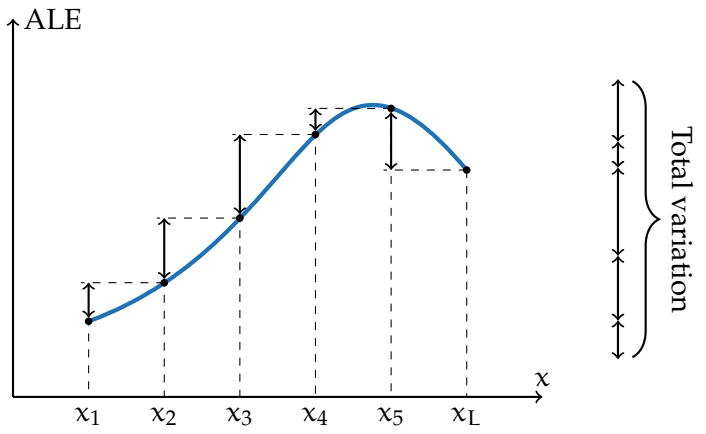


FIGURE 4.4 – The amount of fluctuation the accumulated local effects (ALE) function for a feature exhibits is indicative of how much this feature impacts the predictions. To quantify the fluctuations, an approximation for the total variation TV is calculated by taking the absolute ALE function differences between L consecutive, uniformly spaced values x_l.

4.2.4 FEATURE CLIPPING

In an attempt to prove the validity of our approach, we will perform what we call a feature clipping process with our data sets (see figure 4.5): First, use all available features to train the regressor as well as to do the predictions. Afterwards, sequentially remove one of the features, and redo training and prediction with this reduced data set. With every feature that is being removed, the regressor will have less information to base its predictions on, and we therefore expect the absolute prediction errors to increase. However, depending on the order in which the features are removed, the progression of the absolute prediction errors will vary. Henceforth we will refer to the order in which the features are removed as the clipping order.

In LSD, a feature is given by the L1 normalized absorbed energy density $H(\lambda, \mathbf{z})$ for an illumination wavelength λ. In MI-LSD, a feature is $H(i, \lambda, \mathbf{z})$ for an illumination wavelength λ and an illumination position i. For every step in the feature clipping process one illumination wavelength is removed (for MI-LSD, this means we remove four features in every step; one for each illumination position). It is important to re-normalize the absorbed energy densities in every step of the process.

We will compare three clipping orders:

- 'State of the art' (SOA) [16] - Remove wavelengths uniformly, while keeping as large of a span of the wavelengths as possible. To be precise, the order is

FIGURE 4.5 – In a feature clipping process, one of the features $\{x_1, x_2, \ldots\}$ is sequentially removed from the training and prediction stage of the regressor. The rule, which decides what feature is removed in the next step, is called the clipping order.

{980, 680, 940, 720, 900, 760, 860, 800, 960, 700, 880, 780, 920, 740, 840, 820} nm where 820 nm is removed first, while 980 nm remains until the last step.

- 'Min ALE' - Rank all 16 wavelengths according to the total variation of their accumulated local effects function TV(ALE) in ascending order, such that the feature with the lowest TV(ALE) is removed first.
- 'Updated min ALE' - In every step of the feature clipping process, for all remaining wavelengths the total variation of the accumulated local effects function TV(ALE) is determined, and the wavelength with the lowest TV(ALE) is eliminated.

For MI-LSD each illumination position is treated separately, and the resulting total variation (TV) is given by the sum of the individual TVs.

To compute the ALE function, we use every tenth sample in the *in silico* training set. For the partition, which is needed to approximate the ALE function (see equation (4.3)) we use the $K = 201$ uniformly spaced quantiles $\{\hat{q}_{(k-1)/200}\}_{k=1}^{201}$ of the empirical distribution function, such that all MPDs (4.2) are calculated with the same number of samples, and the ALE function is sufficiently smooth. For the approximation of the TV (see equation (4.6)) we use $L = 50$, because increasing this value further does not have any noticeable effects.

5 RESULTS

5.1 HISTOGRAM BASED GRADIENT BOOSTING REGRESSORS

As described in section 4.1, we test the regressors on the *in silico*, phantom B, and phantom C test data sets. Furthermore, we compare the results to the random forest and neural network investigated in [11], and present them in table 5.1 and table 5.2.

TABLE 5.1 – LSD test data sets (section 3): The median Q_2 and 90th percentile P_{90} of the absolute prediction errors Δ_{pred}, both in units of percentage points (pp), for each of the regressors (see table 4.1). The results for a random forest (RF) regressor and neural networks (NN) [11] are added for comparison.

Regressor	rCu						sO_2	
	In silico		Phantom B		Phantom C		*In silico*	
	Q_2	P_{90}	Q_2	P_{90}	Q_2	P_{90}	Q_2	P_{90}
sklearn	2.4	8.4	4.3	14.2	5.0	16.1	2.4	8.6
XGBoost	2.2	7.9	4.2	12.8	4.5	14.4	2.2	7.9
LightGBM	2.4	8.1	3.2	11.8	4.0	13.9	2.3	8.1
CatBoost	2.4	8.4	3.9	13.8	4.2	15.6	2.4	8.5
Random forest	2.5	8.6	3.3	10.7	3.9	13.7		
Neural network	2.2	7.9	3.4	16.7	5.8	32.6		

TABLE 5.2 – MI-LSD test data sets (section 3): The median Q_2 and 90th percentile P_{90} of the absolute prediction errors Δ_{pred}, both in units of percentage points (pp), for each of the regressors (see table 4.1). The results for a random forest (RF) regressor and neural networks (NN) [11] are added for comparison.

Regressor	rCu						sO_2	
	In silico		Phantom B		Phantom C		*In silico*	
	Q_2	P_{90}	Q_2	P_{90}	Q_2	P_{90}	Q_2	P_{90}
sklearn	2.5	8.8	3.6	9.7	4.6	11.4	2.5	8.6
XGBoost	2.2	7.9	3.3	8.7	3.9	10.4	2.1	7.6
LightGBM	2.5	8.2	2.8	8.3	3.8	11.5	2.4	8.1
CatBoost	2.5	8.9	3.8	10.6	4.7	12.2	2.5	8.8
Random forest	2.9	10.4	2.9	8.8	4.5	12.4		
Neural network	1.9	6.9	5.3	36.2	12.0	58.1		

5.2 ACCUMULATED LOCAL EFFECTS

From now on, we will only use LightGBM's HGB regressor.

5.2.1 LSD

We will start with presenting our results for learned spectral decoloring (LSD).

To visually confirm our algorithm for calculating the accumulated local effects (ALE) function, we first plot this function for each of the wavelengths, both for the rCu (figure 5.1) and the sO_2 (figure 5.2) *in silico* training set.

To estimate the viability of our approaches to selecting illumination wavelengths, we perform the feature clipping process as detailed in section 4.2.4. The resulting distributions of the absolute prediction errors $\Delta_{\mathtt{pred}}$ for each of the test data sets is shown in figure 5.3. A quantitative summary of the results is captured in table 5.3.

The clipping order 'min ALE' (descending in total variation (TV) of the accumulated local effects (ALE) function, meaning the most important wavelengths come first) for rCu is found to be {680, 880, 720, 740, 980, 900, 820, 920, 840, 800, 860, 760, 960, 940, 700, 780} nm, and the clipping order 'updated min ALE' (sequentially eliminating the wavelength which currently exhibits the least TV of its ALE function) is {880, 840, 680, 720, 740, 900, 820, 980, 920, 860, 800, 760, 960, 940, 700, 780} nm.

For sO_2, the 'min ALE' clipping order is {980, 940, 720, 760, 680, 960, 700, 920, 740, 900, 780, 800, 880, 820, 860, 840} nm, and for 'updated min ALE' it is {980, 960, 940, 720, 880, 700, 760, 920, 800, 680, 740, 900, 820, 860, 780, 840} nm.

TABLE 5.3 – Summary for the progression of the median absolute error Q_2 and 90th percentile P_{90} of the absolute prediction errors $\Delta_{\mathtt{pred}}$, both in units of percentage points (pp) for every LSD data set. The clipping order 'min ALE' was used, in other words, each row corresponds to the prediction series based on the n_λ wavelengths with the highest total variation (TV) of the accumulated local effect (ALE) function (see section 4.2.2 for more details).

n_λ	Clipping order	rCu						sO_2	
		In silico		Phantom B		Phantom C		*In silico*	
		Q_2	P_{90}	Q_2	P_{90}	Q_2	P_{90}	Q_2	P_{90}
8	state of the art	2.8	9.1	3.2	12.7	4.2	13.2	2.6	9.0
	min ALE	2.5	8.7	4.0	20.2	6.2	21.9	2.8	9.9
	updated min ALE	2.5	8.7	2.8	12.4	3.9	13.6	2.8	10.2
6	state of the art	3.5	11.4	3.6	15.1	3.1	14.1	2.9	10.6
	min ALE	4.9	16.5	3.5	11.4	4.4	12.2	3.3	11.7
	updated min ALE	4.2	14.6	4.5	15.9	5.4	20.0	3.8	13.1
4	state of the art	6.3	21.9	10.4	25.4	4.7	19.2	11.9	34.0
	min ALE	5.5	19.1	4.7	17.2	4.7	19.3	11.2	32.2
	updated min ALE	5.4	17.4	2.7	9.0	2.7	13.6	6.2	21.1

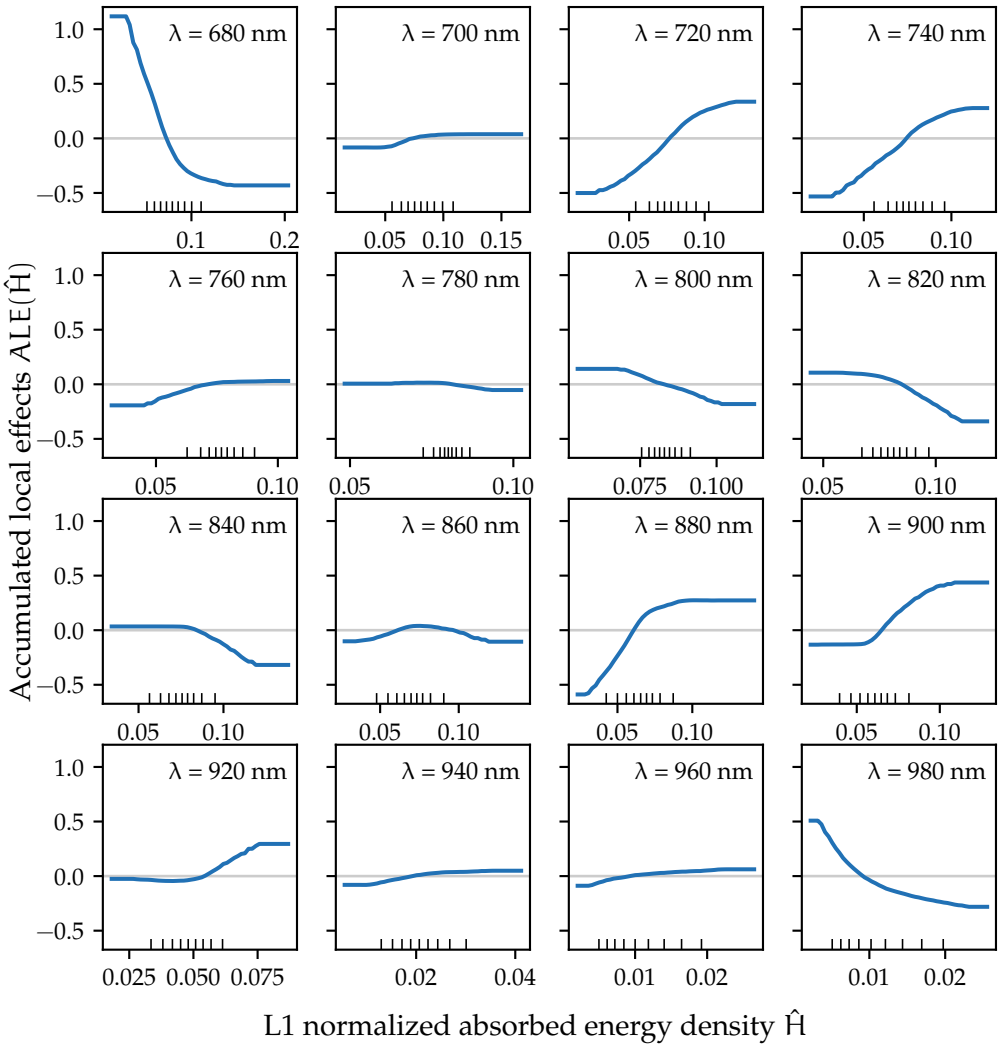


FIGURE 5.1 – The centralized accumulated local effects (ALE) functions for all illumination wavelengths with LSD for rCu. They serve as an indicator of how much different values for this feature impact the predictions. The upwards facing ticks on the horizontal axis reflect the percentiles $P_{10}, P_{20}, \ldots P_{90}$ of the L1 normalized absorbed energy densities $\hat{\mathrm{H}}$.

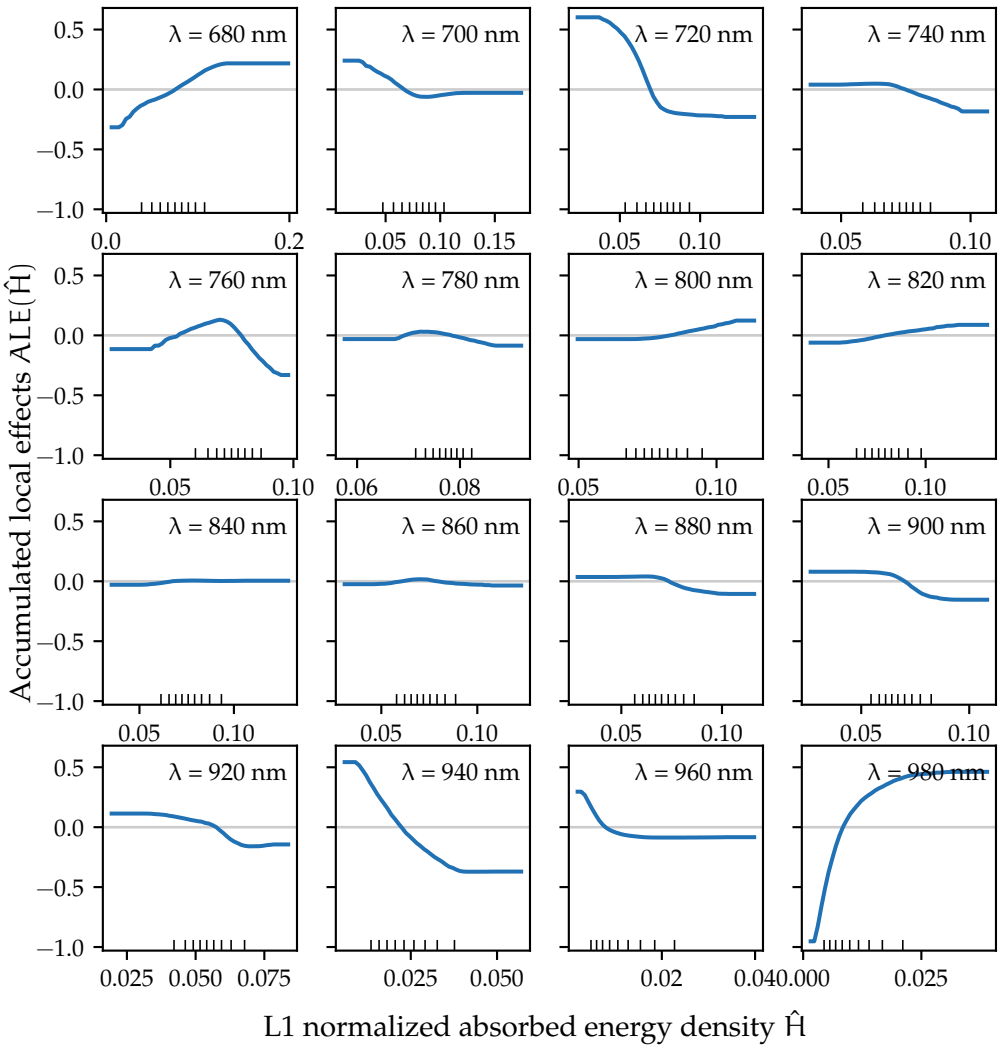
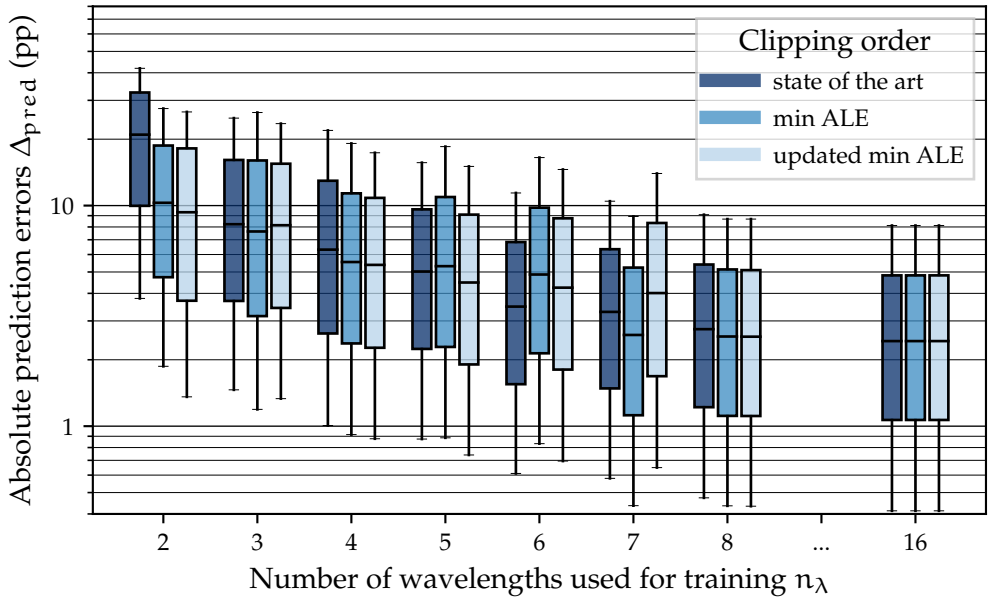
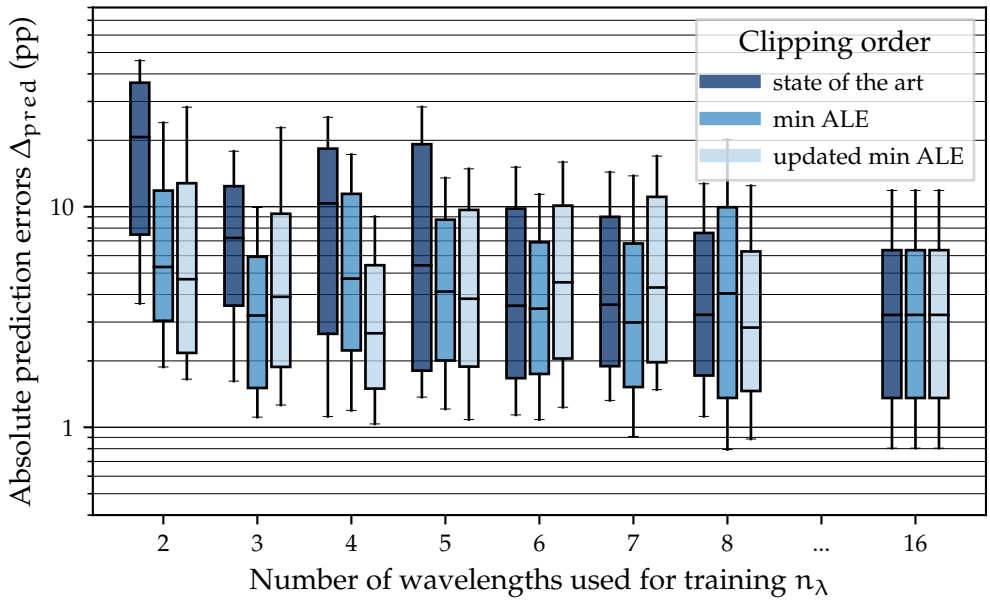


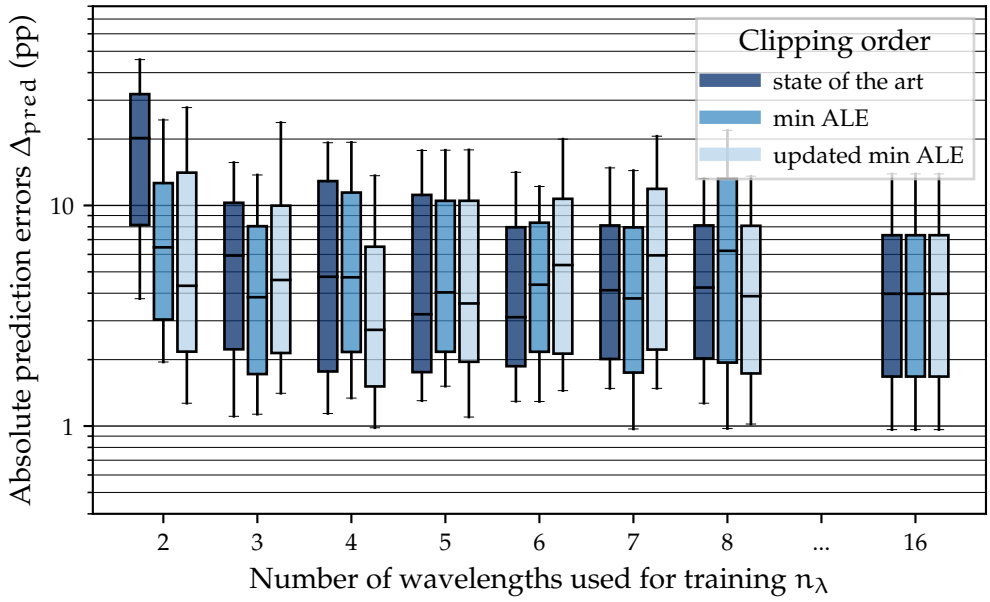
FIGURE 5.2 – The centralized accumulated local effects ALE functions for all illumination wavelengths with LSD for sO_2. The upwards facing ticks on the horizontal axis reflect the percentiles $P_{10}, P_{20}, \ldots P_{90}$ of the L1 normalized absorbed energy densities $\hat{H}$.



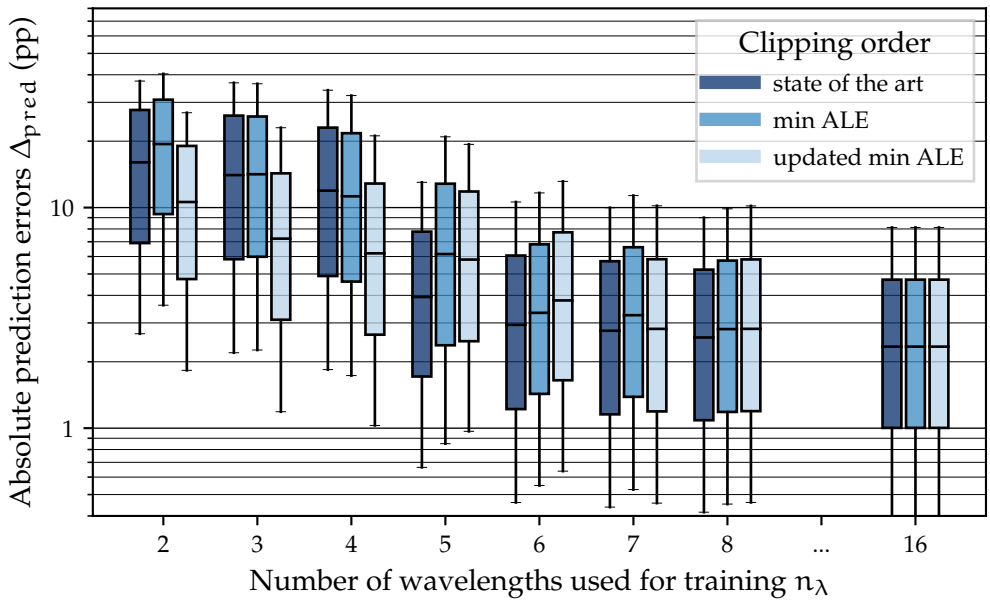
(A) LSD on the *in silico* validation set with rCu.



(B) LSD on the phantom test set B with rCu.



(C) LSD on the phantom test set C with rCu.



(D) LSD on the *in silico* test set with sO_2.

FIGURE 5.3 – LSD on each of the test data sets. The progression of the absolute prediction errors (Δ_{pred}) is shown, when only a subset (of size n_λ) of the 16 available wavelengths is used for training the regressor and making predictions. Three clipping orders (defined in section 4) determine what wavelength will be removed, when the number of available wavelengths n_λ is reduced by one. The boxes indicate the quartiles, while the whiskers show the 10th and 90th percentiles. $n_\lambda \in \{9, \ldots, 15\}$ were omitted because the absolute prediction errors for them remain almost unchanged.

5.2.2 MI-LSD

Following, the results for multiple illumination learned spectral decoloring (MI-LSD) are shown in analogy to the results for learned spectral decoloring (LSD). Both figure 5.4 and figure 5.4 aim to confirm the functionality of our algorithm, while figure 5.6 and table 5.4 show the results for the feature clipping process (see section 4.2.4).

The clipping order 'min ALE' (descending in total variation (TV) of the accumulated local effects (ALE) function, meaning the most important wavelengths come first) for rCu is found to be {680, 980, 720, 880, 740, 900, 820, 800, 840, 860, 920, 940, 760, 960, 780, 700} nm, and the clipping order 'updated min ALE' (sequentially eliminating the wavelength which currently exhibits the least TV of its ALE function) is {940, 980, 840, 880, 800, 720, 680, 780, 740, 820, 760, 860, 920, 900, 960, 700} nm.

For sO_2, the 'min ALE' clipping order is {980, 940, 720, 680, 760, 960, 700, 920, 740, 900, 820, 780, 880, 800, 840, 860} nm, and for 'updated min ALE' it is {980, 720, 740, 940, 820, 800, 780, 840, 680, 760, 880, 960, 700, 920, 900, 860} nm.

TABLE 5.4 – Summary for the progression of the median absolute error Q_2 and 90th percentile P_{90} of the absolute prediction errors $\Delta_{\mathtt{pred}}$, both in units of percentage points (pp) for every MI-LSD data set.

n_λ	Clipping order	rCu						sO_2	
		In silico		Phantom B		Phantom C		*In silico*	
		Q_2	P_{90}	Q_2	P_{90}	Q_2	P_{90}	Q_2	P_{90}
8	state of the art	2.8	9.1	2.8	9.3	3.8	11.3	2.7	8.8
	min ALE	2.5	8.5	2.7	8.7	3.2	11.4	3.1	10.1
	updated min ALE	2.7	8.2	2.6	10.7	2.6	15.0	3.4	11.5
6	state of the art	3.3	11.0	2.7	10.0	3.2	11.9	3.1	10.2
	min ALE	4.0	13.7	3.5	11.1	4.4	12.2	3.7	12.1
	updated min ALE	4.3	15.3	4.9	17.7	8.6	23.6	3.7	12.6
4	state of the art	5.1	17.2	6.9	20.4	7.6	19.3	7.4	24.1
	min ALE	4.5	16.4	4.2	13.5	5.1	18.0	7.4	24.1
	updated min ALE	5.8	18.2	11.0	38.2	15.5	51.4	6.9	24.6

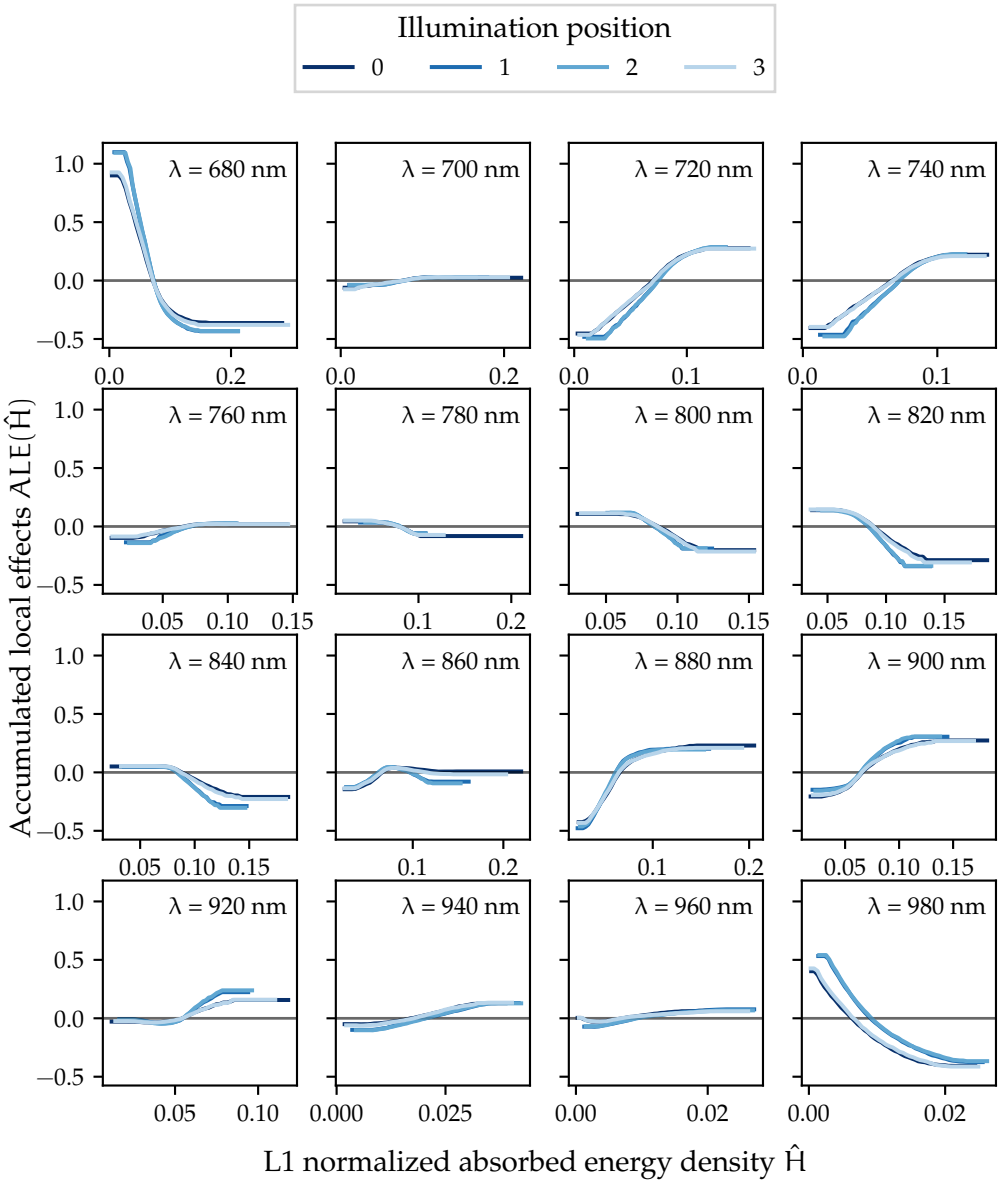


FIGURE 5.4 – The centralized accumulated local effects (ALE) functions for all illumination wavelengths and illumination positions with MI-LSD for rCu.

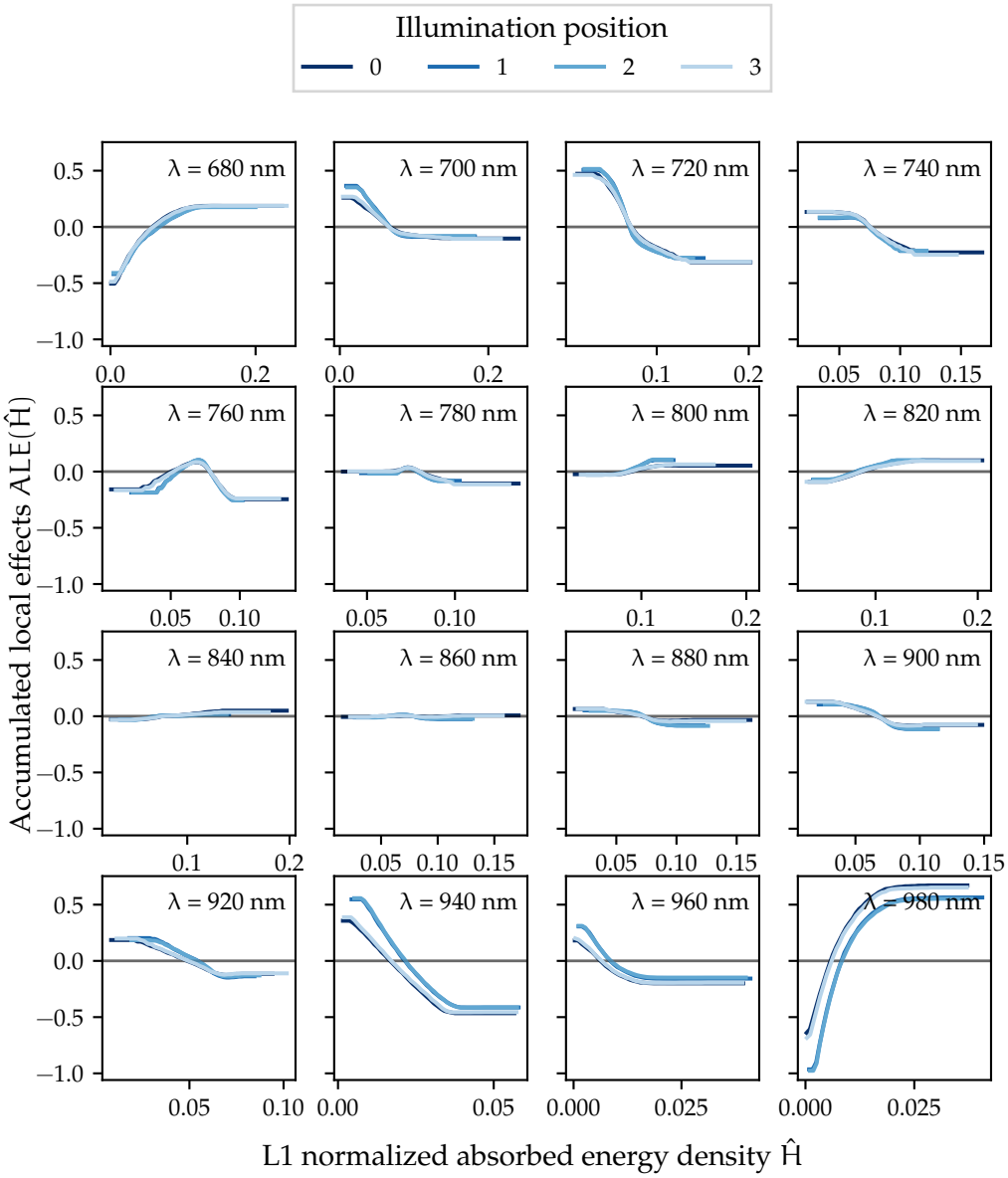
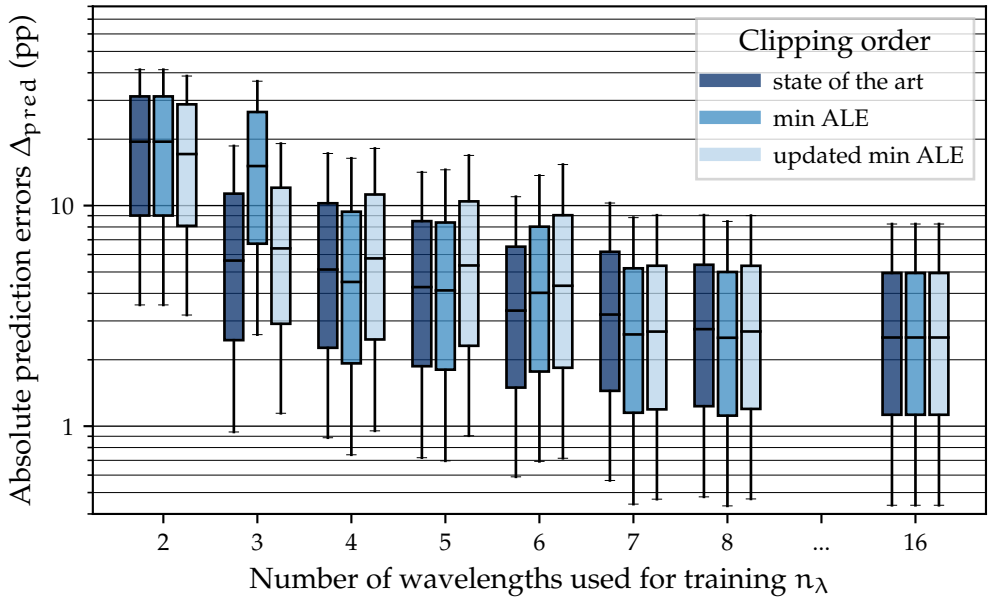
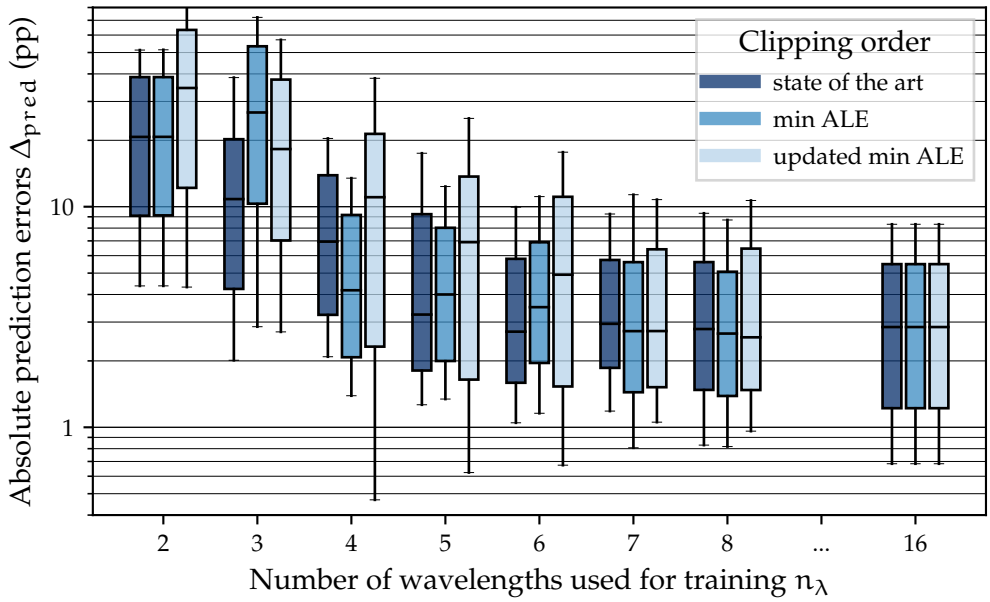


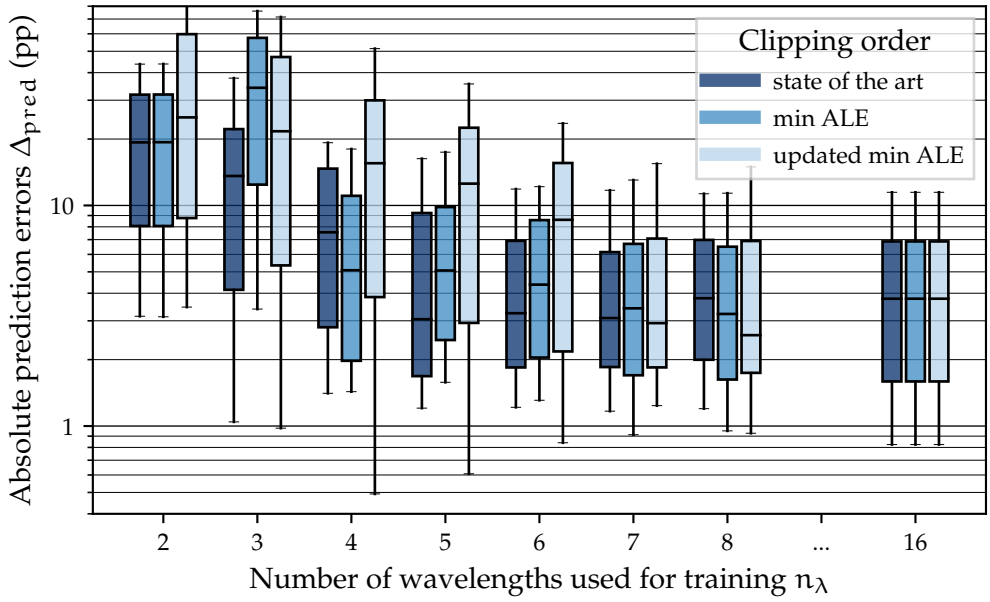
FIGURE 5.5 – The centralized accumulated local effects (ALE) functions for all illumination wavelengths and illumination positions with MI-LSD for sO_2.



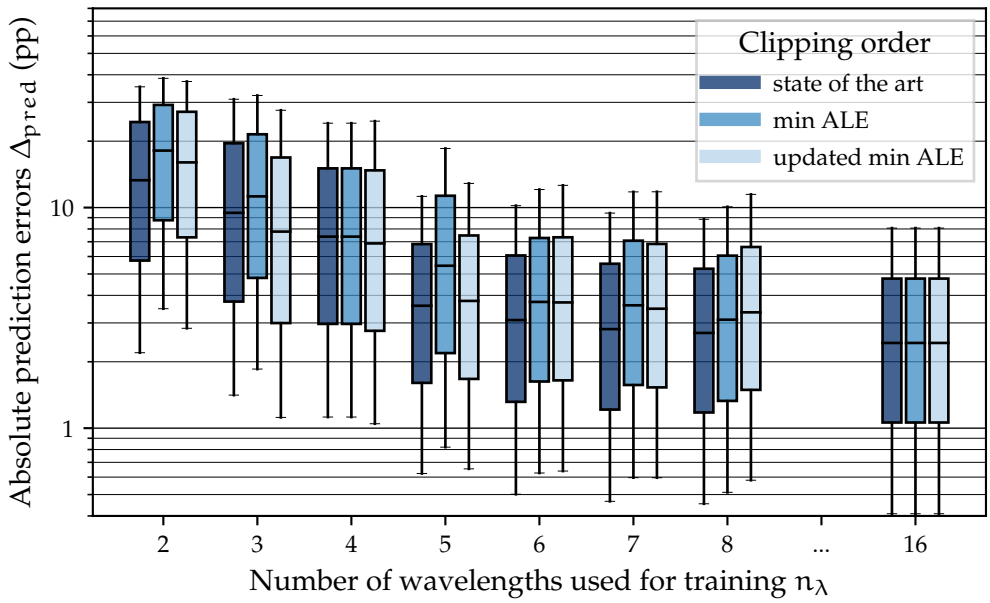
(A) MI-LSD on the *in silico* validation set with rCu.



(B) MI-LSD on the phantom test set B with rCu.



(C) MI-LSD on the phantom test set C with rCu.



(D) MI-LSD on the *in silico* test set with sO_2.

FIGURE 5.6 – For each of the MI-LSD test data sets, the progression of the absolute prediction errors (Δ_{pred}) is shown, when only a subset (of size n_λ) of the 16 available wavelengths is used for training the regressor and making predictions. Three clipping orders (defined in section 4) determine what wavelength will be removed, when the number of available wavelengths n_λ is reduced by one. The boxes indicate the quartiles, while the whiskers show the 10th and 90th percentiles. $n_\lambda \in \{9, \ldots, 15\}$ were omitted because the absolute prediction errors for them remain almost unchanged.

6 DISCUSSION

6.1 HISTOGRAM-BASED GRADIENT BOOSTING REGRESSORS

In the following, we pair table 5.1 and table 5.2 with our experience, to evaluate the suitability of four implementations of a histogram-based gradient boosting (HGB) algorithm for learned spectral decoloring (LSD) and multiple illumination learned spectral decoloring (MI-LSD). We do this according to the four criteria listed in section 4.1.

6.1.1 SKLEARN HISTGRADIENTBOOSTINGREGRESSOR

Included in the scikit-learn (short: sklearn) Python package starting from version v0.21.0, the HistGradientBoostingRegressor is still in an experimental stage, meaning the algorithm and application programming interface (API) may be altered without previous warnings. Therefore, the following statements are likely to change for later versions.

- Accuracy – With median absolute errors Q_2 ranging from 2.4 to 5.0 pp in LSD (table 5.1) and 2.5 to 4.6 pp in MI-LSD (table 5.2), this regressor is one of the least accurate regressors, particularly with LSD on the phantom test sets. Although performing worse than a random forests (RF), the HistGradientBoostingRegressor seems to work particularly well in MI-LSD, where it is able to keep the 90th percentile of the absolute errors P_{90} to a minimum.
- Agility – On the setup specified in section 3.2, the HistGradientBoostingRegressor took just below 2 and 3 minutes to train on the LSD and MI-LSD training set respectively, while never exceeding a memory usage of 3 GB. Prediction time is approximately the same as for RFs [11].
- Adaptability – The HistGradientBoostingRegressor currently only supports three types of loss functions: least squares, least absolute deviation, and poisson. Furthermore, from what we know, scikit-learn offers no support for passing a custom loss function to the HistGradientBoostingRegressor. Quantile regression (and therefore prediction intervals) and more advanced loss functions are not available for this regressor.
- Accessibility – The API from the scikit-learn package acts as a role model for most other tree-based machine learning applications. It is straight forward and consistent over all regressors offered in this package. It is by default installed with the most popular package manager Anaconda[12].

6.1.2 XGBOOST XGBREGRESSOR

- Accuracy – Q_2 (2.2 to 4.5 pp in LSD and 2.2 to 3.9 pp in MI-LSD) and P_{90} of the XGBRegressor were comparatively low for all test data sets. The difference in performance on phantom test set B to the out of distribution phantom test set C is remarkably low when compared to the other regressors.

[12]https://anaconda.com

- Agility – With less than 2 and 5 minutes training time for LSD and MI-LSD respectively, and memory usage of up to 5 GB, the XGBRegressor is computationally demanding, particularly for MI-LSD. Prediction time is approximately the same as for RFs.
- Adaptability – Besides the few built-in loss functions, XGBRegressor also allows the user to define a custom loss function to suit their needs. However, the algorithm requires the loss function to be twice differentiable. Therefore, excluding the possibility to do a quantile regression[13] and of using the mean absolute error as a loss function.
- Accessibility – Installing the xgboost Python package is straight forward, and the API is very similar to the one in sklearn.

6.1.3 LIGHTGBM LGBMREGRESSOR

- Accuracy – The LGBMRegressor almost always outperformed the other regressors, with a Q_2 ranging from 2.4 to 4.0 pp in LSD and 2.5 to 3.8 pp in MI-LSD. Particularly the performance on phantom test set B is impressive, where LGBMRegressor produces median absolute errors significantly lower than every other gradient boosting regressor, and even the RF regressor.
- Agility – The LGBMRegressor required less than 2.2 GB of memory for both LSD and MI-LSD, while training for only approximately 20 and 50 s respectively, making it the least computationally taxing regressor to be examined. Prediction time once again compares to RFs.
- Adaptability – By default, it supports a large number loss functions, including quantile loss for prediction intervals, as well as the capability of using custom ones.
- Accessibility – Installing the lightgbm Python package is straight forward, and the API is once again very similar to the one in scikit-learn.

6.1.4 CATBOOST CATBOOSTREGRESSOR

- Accuracy – With a Q_2 from 2.4 to 4.2 pp in LSD and 2.5 to 4.7 pp in MI-LSD, the CatBoostRegressor sticks out because of its remarkable performance in LSD. While all other regressors produced lower median absolute errors for MI-LSD, the CatBoostRegressor showed the exact opposite behavior. However, the rather high P_{90} for this regressor means that there are quite a lot of outlying predictions.
- Agility – Low memory usage (less than 2.5 GB) but long training times (8 and 18 minutes) characterize this regressor. Prediction times are lower than for RFs.

[13] Many online sources suggest using the $|x| \approx \ln\cosh x$ approximation to then construct an approximate quantile loss function. However, this is very much not advisable, because the second derivative of $\ln\cosh x$ does not vanish at $x = 0$, and therefore a discontinuity in the second derivative of the quantile loss function unveils itself. For the hyperparameters in table 4.1, it turns out that a quantile loss function based on x^4 delivers much better results.

- Adaptability – A large collection of loss functions (including quantile loss) is supported by default, and it is possible to define custom loss functions.
- Accessibility – Installing the catboost Python package is straight forward.

6.1.5 General remarks

In general, for both LSD and MI-LSD the examined HGB regressors are comparable to previous implementations of RFs and better than previously reported NNs on phantom data. They require comparatively much less time and resources to train than the latter two regressors. Binning the continuous feature values also has the useful side-effect of leaving the time for training almost unchanged if the amount of training data is increased.

As can be seen in table 5.1 and table 5.2, for the majority of the histogram-based gradient boosting regressors the absolute prediction errors increase significantly when switching from the *in silico* validation set to the phantom test set B, but only increase subtly when switching from the phantom test set B to the phantom test set C. The exact opposite is the case for random forests: Absolute prediction errors change only subtly between the *in silico* validation set and the phantom test set B, while the increase is relatively big when switching from the phantom test set B to the phantom test set C. These are obvious signs of overfitting. Because the hyperparameters were tuned on the *in silico* validation set, our regressors are specialized on these data sets. For their predictions the regressors likely rely on information and patterns which are characteristic for this particular *in silico* training set, which might not be as pronounced in the phantom test data sets due to higher noise or weaker signal strengths. Highly specialized regressors on one type of data will generally not be as accurate on other types of data sets.

A big disadvantage of tree-based regressors, and especially HGB regressors, is the fact that such regressors will realistically never predict an rCu or sO_2 of 0 or 1 (of which the latter is probably of greater importance). In tree-based regressors, every prediction is based on averages of the labels of multiple samples[14] in the same leaf. Even a single label with a subtly different value from otherwise equal labels is sufficient to "contamine" the average, which will then no longer exactly agree with the other labels. This, combined with binning the continuous features, might have lead to more contaminations by smaller labels, and the tendentially increasing "uncertainty" of the regressor when it comes to predicting high rCu and sO_2 values leads to highly inaccurate predictions in the high rCu and sO_2 regime.

This effect is particularly potent in phantom test sets, as can be seen in figure 6.1, because predictions for the *in silico* validation set are a lot more accurate for high rCu. Therefore, the cause of this deviation may be traced back to a shortcoming in the simulations.

[14]For most regressors, the lower bound for the minimum amount of samples that must remain in a leaf can be specified via the parameter 'min_data_in_leaf' (LightGBM, CatBoost) or 'min_samples_leaf' (sklearn), and its default value is usually set to 20. However, to avoid overfitting it is not advisable to lower this parameter by too much.

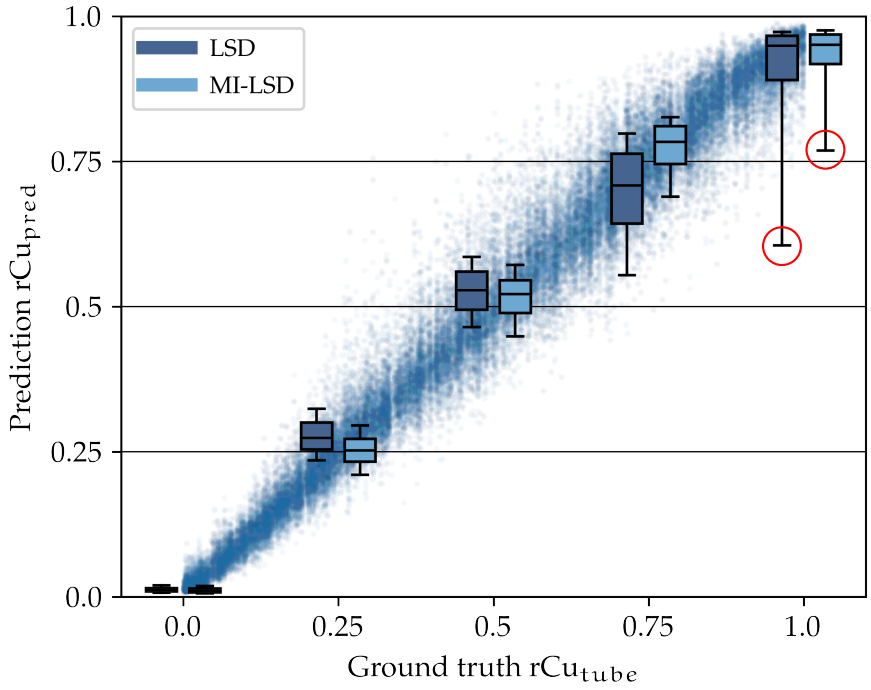


FIGURE 6.1 – The predictions made by the LGBMRegressor with hyperparameters as specified in table 4.1 plotted against the actual rCu levels $\{0, 0.25, 0.5, 0.75, 1\}$ in the tubes in phantom test set B for both LSD and MI-LSD. The boxes indicate the quartiles, while the whiskers show the 10th and 90th percentiles. For reference, the predictions for the *in silico* validation set are shown in the background. The problematic prediction behavior is circled in red.

6.2 EXPLAINABLE WAVELENGTH SELECTION

All in all, our two method of selecting wavelengths to train the regressor with compare quite well to the current state of the art (SOA) (as can be observed from the feature clipping processes, plotted in figure 5.3 and figure 5.6). Particularly for LSD our methods seem to work better. Generally, it can be seen that with our methods of selecting wavelengths, the amount of wavelengths n_λ to train the regressor with can, without much impact on the phantom test sets, be reduced from 16 to 8, and sometimes even to 6. In LSD (figure 5.3) both of our proposed methods seem to give similar results, while in MI-LSD (figure 5.6) the clipping order 'min ALE' is more reliable.

However, our methods exhibits three flaws, which shall be listed and discussed in the following.

- One thing the total variation (TV) metric does not account for when "measuring" the amount of fluctuation in the (ALE) function, is the fact that the distribution of typical values for this feature is not accounted for. To see this, consider an ALE function such as the one seen in figure 5.2 for $\lambda = 720nm$: For approximately half of the samples (as seen by the quantiles drawn as ticks on the x-axis), the ALE function has quite a steep slope, while for the other half of the samples, it is nearly constant. This means that theoretically, only half of all samples would be affected by this steep slope in the ALE function, while the other half would not. It is then obvious to say that if even an smaller portion of the samples would be located in a steep part in the ALE function, the overall impact of this feature on the predictions would be quite small. On the other hand, if almost all samples are located in a steep part in the ALE function, the impact would be much larger.

- Selecting evenly spaced wavelengths while maintaining the largest possible wavelength span is only the SOA for the sO_2 imaging [16]. Since this might not directly translate to rCu imaging. This might be the reason why our method seems to perform much better with rCu than with sO_2, when compared to the SOA.

- Uniformly selecting elements from a set of 16 is extremely biased and can only effectively be achieved in four cases (for $n_\lambda \in \{16, 6, 4, 2\}$). By our definition of the feature clipping process, it is impossible to have both $n_\lambda = 6$ and $n_\lambda = 4$ be uniformly spaced[15]. In fact, except for $n_\lambda = 2$ and $n_\lambda = 16$, not a single n_λ in the feature clipping plots is truly uniformly spaced, but rather a compromise to have as many n_λ be as close as possible to a truly uniformal spacing. The fact that similar interpretations of uniformly removing elements from a set of 16 lead to vastly different outcomes (compare figure 6.2 to figure 6.3), even further shows the problem of this subjective decision.

[15] Because for $n_\lambda = 6$, the wavelengths $\lambda \in \{680, 740, 800, 860, 920, 980\}$ and for $n_\lambda = 4$, the wavelengths $\lambda \in \{680, 780, 880, 980\}$ would need to be used, but these sets obviously are not subsets of each other.

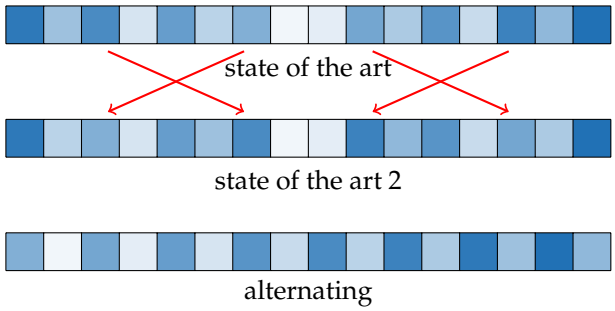


FIGURE 6.2 – Similar uniformal clipping orders, visualized by the color brightness, lead to vastly different results (compare figure 6.3). Each square corresponds to one wavelength, starting with 680 nm on the left, and ending with 980 nm on the right. The darker the color, the longer the corresponding wavelength will be kept. Notice, that from 'state of the art' to 'state of the art 2', the order of only four wavelengths was altered (red arrows).

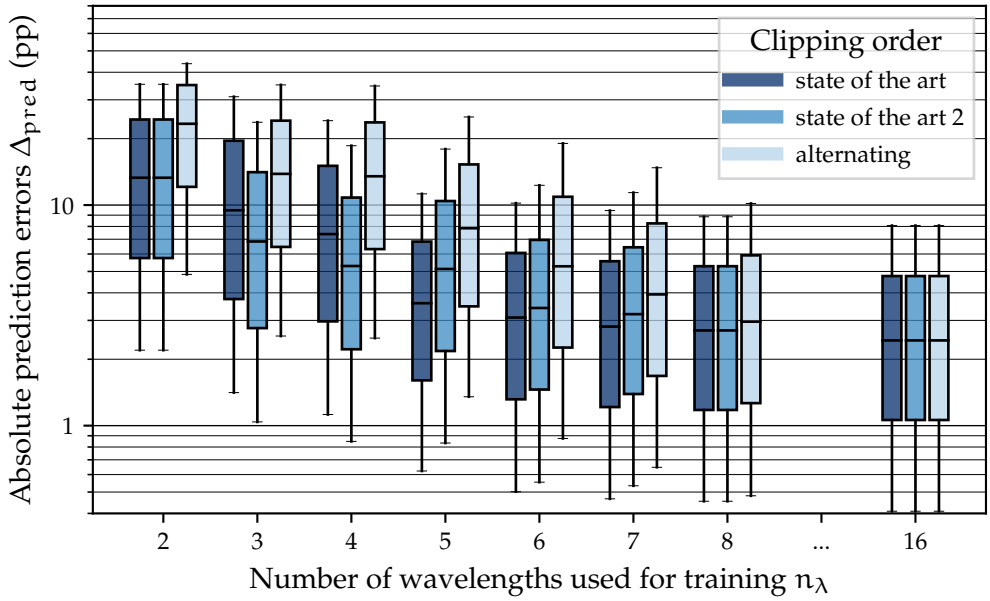


FIGURE 6.3 – Similar clipping orders lead to vastly different results (compare figure 6.2). Here the progression of the absolute prediction errors is shown for MI-LSD with the sO_2 *in silico* test set. The boxes indicate the quartiles while the whiskers show the 10th and 90th percentiles.

The accumulated local effects plots in section 5.2 provide us with three interesting insights. First, the only ALE function which is far from being monotone (see figure 5.2 at $\lambda = 760$ nm, marked with a red line) is the one for sO_2 which corresponds to the spike in wavelength dependent absorption coefficient $\mu_a(\lambda)$ of Hb (figure 2.2). Second, the ALE functions for sO_2 at wavelengths between $\lambda = 860$ and 920 nm have very similar shapes (see red area in figure 5.2), once again corresponding to what one would expect from looking at the graphs in figure 2.2 for this wavelength region. Finally, all ALE functions in figure 5.4 and figure 5.5 only really vary in scale and not in shape for different illumination positions, so it should be sufficient to only consider one illumination position to determine the "importance" of an illumination wavelength.

7 Conclusion

In our work, we identified an efficient class of supervised learning (SL) models, the histogram-based gradient boosting (HGB) regressors. With median absolute prediction errors Q_2 of 3.2 to 5.0 pp with learned spectral decoloring (LSD) and 2.8 to 4.7 pp with multiple illumination learned spectral decoloring (MI-LSD) on rCu phantom models, their accuracy is comparable to and often slightly better than previously reported regressors [11]. However, in terms of training time, memory consumption, and scalability the HGB regressors outperform the aforementioned regressors by orders of magnitude.

We were then able to make use of the high agility of HGB regressors to develop a more explainable and intuitive alternative to the state of the art of selecting illumination wavelengths [16]. With our method of selecting wavelengths for illumination via the total variation (TV) of the accumulated local effects (ALE) function, we could show that cutting the amount of wavelengths used to train a regressor in half (16 to 8 or even to 6) does not have a significant impact on the prediction accuracy with LSD, while tending to even increase said accuracy with MI-LSD. This reduction in wavelengths allows us to decrease the time it takes to record an imaging sequence, and significantly reduces hardware requirements.

REFERENCES

[1] B. S. Sørensen and M. R. Horsman. "Tumor Hypoxia: Impact on Radiation Therapy and Molecular Pathways". In: *Frontiers in Onconology* (2020), p. 562. DOI: 10.3389/fonc.2020.00562.

[2] F. Colliez, B. Gallez, and B. F. Jordan. "Assessing Tumor Oxygenation for Predicting Outcome in Radiation Oncology: A Review of Studies Correlating Tumor Hypoxic Status and Outcome in the Preclinical and Clinical Settings". In: *Frontiers in Onconology* 7 (2017), p. 10. DOI: 10.3389/fonc.2017.00010.

[3] L. Ulrich et al. "Spectral correction for handheld optoacoustic imaging by means of near-infrared optical tomography in reflection mode". In: *Journal of Biophotonics* 12 (1) (2019). DOI: 10.1002/jbio.201800112.

[4] J. Laufer. *Photoacoustic Imaging: Principles and Applications*. Springer, Cham, 2018, pp. 303–324. DOI: 10.1007/978-3-319-65924-4_13.

[5] T. Kirchner. *Real-time blood oxygenation tomography with multispectral photoacoustics*. 2019. DOI: 10.11588/heidok.00026696.

[6] B. E. A. Saleh and M. C. Teich. *Fundamentals of Photonics*. 3rd ed. John Wiley & Sons, Inc, 2019. DOI: 10.1002/0471213748.

[7] S. Tzoumas and V. Ntziachristos. "Spectral unmixing techniques for optoacoustic imaging of tissue pathophysiology". In: *Philos Trans A Math Phys Eng Sci* 375 (2017), p. 2107. DOI: 10.1098/rsta.2017.0262.

[8] H. Deng et al. "Deep learning in photoacoustic imaging: a review". In: *Journal of Biomedical Optics* 26 (3) (2021). DOI: 10.1117/1.JBO.26.4.040901.

[9] J. Gröhl et al. "Deep learning for biomedical photoacoustic imaging: A review". In: *Photoacoustics* 22 (2021), p. 100241. DOI: 10.1016/j.pacs.2021.100241.

[10] T. Kirchner, J. Gröhl, and L. Maier-Hein. "Context encoding enables machine learningbased quantitative photoacoustics". In: *Journal of Biomedical Optics* 23 (5) (2018), pp. 1–9. DOI: 10.1117/1.JBO.23.5.056008.

[11] T. Kirchner and M. Frenz. *Quantitative photoacoustic oximetry imaging by multiple illumination learned spectral decoloring*. 2021. arXiv: 2102.11201v1.

[12] J. H. Friedman. "Greedy function approximation: A gradient boosting machine." In: *The Annals of Statistics* 29 (5) (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451.

[13] T. Chen and C. Guestrin. "XGBoost: A Scalable Tree Boosting System". In: KDD '16 (2016), pp. 785–794. DOI: 10.1145/2939672.2939785.

[14] G. Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: NIPS'17. Curran Associates Inc., 2017, pp. 3149–3157.

[15] D. W. Apley and J. Zhu. *Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models*. 2019. arXiv: 1612.08468v2.

[16] G. P. Luke, S. Y. Nam, and S. Y. Emelianov. "Optical wavelength selection for improved spectroscopic photoacoustic imaging". In: *Photoacoustics* 1 (2) (2013), pp. 36–42. DOI: 10.1016/j.pacs.2013.08.001.

[17] M. W. Sjoding et al. “Racial Bias in Pulse Oximetry Measurement”. In: *New England Journal of Medicine* 383 (25) (2020), pp. 2477–2478. DOI: 10.1056/NEJMc2029240.

[18] M. Nitzan, A. Romem, and R. Koppel. “Pulse oximetry: fundamentals and technology update”. In: *Medical Devices: Evidence and Research* 7 (2014), pp. 231–239. DOI: 10.2147/MDER.S47319.

[19] A. G. Bell. “On the production and reproduction of sound by light”. In: *American journal of science* s3-20 (1880), pp. 305–324. DOI: 10.2475/ajs.s3-20.118.305.

[20] S. L. Jacques. “Role of tissue optics and pulse duration on tissue effects during high-power laser irradiation”. In: *Applied Optics* 32 (13) (1993), pp. 2447–2454. DOI: 10.1364/AO.32.002447.

[21] B. T. Cox et al. “Quantitative spectroscopic photoacoustic imaging: a review”. In: *Journal of Biomedical Optics* 17 (6) (2012), pp. 1–23. DOI: 10.1117/1.JBO.17.6.061202.

[22] A. J. Welch and M. J. C. van Gemert. *Optical-Thermal Response of Laser-Irradiated Tissue*. 2nd ed. Springer, Netherlands, 2011. DOI: 10.1007/978-90-481-8831-4.

[23] S. L. Jacques. “Coupling 3D Monte Carlo light transport in optically heterogeneous tissues to photoacoustic signal generation”. In: *Photoacoustics* 2 (4) (2014). DOI: 10.1016/j.pacs.2014.09.001.

[24] L. Breiman et al. *Classification and regression trees*. Chapman and Hall/CRC, 1984, pp. 216–232. DOI: 10.1201/9781315139470.

[25] T. Chen, C. Guestrin, and J. H. Friedman. *The Elements of Statistical Learning*. 2nd ed. Springer, New York, 2016. DOI: 10.1007/978-0-387-84858-7.

[26] Y. Zhu et al. “Light Emitting Diodes based Photoacoustic Imaging and Potential Clinical Applications”. In: *Scientific Reports* 8 (9885) (2018). DOI: 10.1038/s41598-018-28131-4.

[27] B. I. Golubov and A. G. Vitushkin. “Variation of a function”. In: *Encyclopedia of Mathematics* (2001). URL: http://encyclopediaofmath.org/index.php?title=Variation_of_a_function&oldid=18738.

RESOURCES

The rCu data sets[16] and a Python module[17] for reproducing all plots featured in section 5.2 are available as open data and open source.

[16] doi:10.5281/zenodo.4549631
[17] https://github.com/FMatti/ALE-LSD

LIST OF FIGURES

LIST OF TABLES

ERKLÄRUNG

Ich erkläre hiermit, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe r des Gesetzes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.

Für die Zwecke der Begutachtung und der Überprüfung der Einhaltung der Selbständigkeitserklärung bzw. der Reglemente betreffend Plagiate erteile ich der Universität Bern das Recht, die dazu erforderlichen Personendaten zu bearbeiten und Nutzungshandlungen vorzunehmen, insbesondere die schriftliche Arbeit zu vervielfältigen und dauerhaft in einer Datenbank zu speichern sowie diese zur Überprüfung von Arbeiten Dritter zu verwenden oder hierzu zur Verfügung zu stellen.[18]

Bern, 07.06.2021	
Location, Date	Matti, Fabio (18-102-087)

[18] Gemäss "Reglement über das Studium und die Leistungskontrollen an der Philosophisch-naturwissenschaftlichen Fakultät" (Stand 1. August 2019).