Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Markus Rupp

born in Karlsruhe (Germany)

2023

Early Stopping in Artificial Neural Networks

This Bachelor Thesis has been carried out by Computational Biology and
Biological Physics at Lunds Universitet under the supervision of
Prof. Dr. Jürgen Hesser and
Dr. Patrik Edén

Contents

1 Abstract in English

Early stopping with a validation set is a widely used method for regularization in the training of artificial neural networks, among other concepts. Two other alternative methods have been introduced which are able to use the entire dataset for the training instead of holding out a validation set. They are based on correlations between the gradient components of the updated weights. The robustness of two novel criteria, the Evidence based criterion and the Gradient correlation criterion are tested for different batch sizes from stochastic gradient descent and learning rate as well as a validation-based criterion. Six different datasets from different domains are used. The novel criteria stop mostly earlier than the validation-based one. Based on the specifications, the differences between the averages are less than 1 %. After that, optimal values for the learning rate and the batch size are obtained using cross-validation and the Tree Parzen Optimizer. In most of the cases, the performances can be increased, still the performance averages between the different stopping criteria remain close. Subsequently, the effectiveness of incorporating L2-weight regularization and dropout layers is evaluated, resulting in slightly improved performances. However, their outcomes are influenced by various factors, making it challenging to precisely determine their exact effects.

2 Abstract in German

Early stopping mit einem Validierungdatensatz ist eine weit verbreitete Methode in der Regularisierung im Training von neuronalen Netzen neben weiteren Konzepten. Zwei alternative Methoden wurden vorgestellt, die dafür sorgen, dass kein Validierungsdatensatz beiseite gehalten werden muss. Die Robustheit gegenüber Änderungen der Päckchengröße aus dem Gradientenabstieg und Lernrate werden überprüft mit den beiden Stoppalgorithmen, inklusive einem Stoppalgorithmus basierend auf dem Validierungsdatensatz. Sechs Datensätze aus verschiedenen Bereichen werden zum Training ausgewählt. Die neuartigen Kriterien führen meist zu einem früheren Stopp; wenn man die Durchschnitte der Leistungen vergleicht, kommt es zu Unterschienden von weniger als einem Prozent. Anschließend werden die optimalen Werte der Lernrate und Päckchengröße durch einen Tree Parzen Optimizer berechnet, die Durchschnitte der Leistungen steigen, aber bleiben trotzdem nah bei einander. Anschließend wird die Wirksamkeit der Integration von L2-Gewichtsregularisierung und Dropout-Schichten bewertet, was zu leicht verbesserten Leistungen führt. Ihre Ergebnisse werden jedoch von verschiedenen Faktoren beeinflusst, wodurch es schwierig ist, ihre genauen Auswirkungen genau zu bestimmen.

3 Introduction

In recent years, artificial neural networks have become increasingly popular in machine learning, with their history dating back to the 19th century when Gauss and Legendre implemented the first shallow networks for regression [20]. However, significant progress in the field was made during the second half of the 20th century. The emergence of Graphical Processing Units (GPUs) after 2012 further accelerated their adoption.

Artificial neural networks can be scaled up to handle complex classification and regression problems. Nevertheless, they may suffer from overfitting, wherein the trained model becomes too adapted to the

specific training data and performs poorly on unseen data. To address this issue, various approaches have been introduced [17].

One common strategy is the use of regularization techniques such as Ride regression (L-2) and Ridge regression (L-1), which help prevent overfitting by adding a penalty term to the loss function. Additionally, dropout layers or data augmentation by injecting noise into the data are employed to improve generalization. Moreover, the size of the neural network can be adjusted to suit the problem at hand, offering more flexibility and potential for better performance.

A straightforward approach to assess model performance is by dividing the dataset into a training set and a validation set. The model is trained on the training set and evaluated on the validation set. Intuitively, the training loss should be lower than the validation loss, as the model becomes adapted to the training data while the validation set estimates its performance on new data [19]. During the training process, the validation loss may initially decrease along with the training loss. However, after some time, the validation loss might start to increase while the training loss continues to decrease. To achieve the best performance on unseen data, it is recommended to stop the training when this occurs. One drawback of this method is the need to find an appropriate split between the validation and training sets. Larger validation sets provide more precise insights into the model's generalization ability, but they result in smaller training sets, making it challenging to showcase the entire complexity of the model [17].

In order to find this point without having to use a held-out validation set, Mahsereci, Lassner, Balles and Hennig introduced a stopping criterion based on fast-to-compute statistics of the computed gradient in their paper "Early stopping without validation set"[17]. Furthermore, the undergraduate student Lizotte presented a gradient correlation criterion which depends on the correlations of the computed gradients in his thesis [16]. While in the beginning, the gradients calculated from the individual datapoints are highly correlated they become anticorrelated when getting closer to the global minimum in weight space. This correlation is quantified by an estimated which has to reach a specific threshold in order to stop the training.

In this study, the criteria are empirically tested on classification datasets from various domains, each having unique specifications. They are compared with a validation-based criterion, which involves selecting the stopping point through an independently training loop with a held-out validation set. The performance of the stopping epochs is then compared to the optimal stopping epoch in which the model achieves the best generalized fitting on a separate held-out test set. To ensure robustness, these criteria are tested using different batch sizes and learning rates similar to grid search. Afterward, the Tree Parzen estimator is employed to identify the best hyperparameters for each stopping criterion. Subsequently, the effects of incorporating L-2 regularization and dropout layers are evaluated with further optimizations.

The focus of this thesis is more on the model performance and less on saving computation time.

4 Theory

4.1 Learning tasks of ANN

Assuming we have a metrics space $\mathcal{X}$ and a subset of the real numbers $\mathcal{Y} \subset \mathbb{R}$, we define $\mathcal{Z}$ as the product of both sets $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Then $f : \mathcal{X} \to \mathcal{Y}$ is a predictor and $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ a convex loss function which measures the loss $\mathcal{L}(y, f(x))$ for a given $(x, y) \in \mathcal{Z}$. We define ρ as a Borel-probabilistic measure on $\mathcal{Z}$ and our objective is to minimize the generalization error $\epsilon(f) = \int_{\mathcal{Z}} \mathcal{L}(y, f(x))d\rho$ in order to find the best predictor f. We assume that ρ is unknown and we just take a finite sample set $\mathcal{D} = (x_i, y_i)_{i=1}^n \subset \mathcal{Z}$. Sometimes we call (x_i, y_i) also a datapoint, x_i an input or feature vector and y_i a target value or a label in case of classification. We estimate $f_\rho^{\mathcal{L}}$ according to $\mathcal{D}$ to find our best guess based on the information we have [21].

4.1.1 Regression

Regression is the most general problem task. Given some $\mathcal{D}$ a certain loss function is minimized. Very common is the mean squared error as the loss function:

$$\mathcal{L} = MSE = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2 \tag{1}$$

In many cases, it is assumed that the ground truth exhibits a linear relationship, and our objective is to estimate the slope and intercept of a linear function $f_\rho^{\mathcal{L}}$ for one or multiple independent variables.

4.1.2 Binary classification

A special case of regression where we restrict our y_i values just to take discrete natural numbers is called classification. We call the sets of different y_i-values classes denoted by C_j:

$$C_j := \{(x_i, y_i)|y_i = j\} \subset \mathcal{D} \tag{2}$$

and our whole dataset $\mathcal{D}$ can be divided in different classes C_j. In case $j \in \{0, 1\}$ it is called binary classification and $f(x_i) \in [0, 1] \forall\ (x_i, y_i) \in \mathcal{D}$. The function $f_\rho^{\mathcal{L}}$ is called a (binary) classifier which we will just denote by f for now.

The loss function which we want to minimize is typically the binary cross entropy:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[(y_i \cdot \log(f(x_i)) + (1 - y_i) \cdot \log(1 - f(x_i)))\right] \tag{3}$$

If values $y_i = 0$ the first term disappears and only the logarithmic value of $1 - f(x_i)$ is taken whereas for $y_i = 1$ the second term disappears and we just take the logarithm of the output $f(x_i)$. Usually $f(x_i)$ takes only values between 0 and 1 and it can be interpreted as a probability of x_i to be in class 1 whereas $1 - f(x_i)$ represents the probability to be in class 0. That is why we multiply our sum with minus 1 to keep the loss positive. $C_{j,pred}$ are our predicted classes:

$$C_{1,\gamma} = \{(x_i, y_i) \in \mathcal{D}|f(x) \geq \gamma\} \tag{4}$$

$$C_{0,\gamma} = \{(x_i, y_i) \in \mathcal{D}|f(x) < \gamma\} \tag{5}$$

$\gamma \in [0,1]$ is a threshold value, most commonly 0.5.

Another value of great importance is the accuracy of our classification. It gives the fraction of correctly predicted values. It is defined in the following way with a dataset $(x_i, y_i)_{i=1}^n$ and a classifier f:

$$\text{acc} = \frac{|C_0 \cap C_{0,0.5}| + |C_1 \cap C_{1,0.5}|}{n} \tag{6}$$

Both the loss and accuracy determine how good the function f performs and both are independent on the sample size. In case of unbalanced $\mathcal{D}$, e.g. with $|C_0| \gg |C_1|$, another measure is favoured, the so called AUC (area under curve). Let *prec* be the precision, also known as sensitivity

$$\text{prec}(\gamma) = \frac{|C_{1,\gamma}|}{C_1} \tag{7}$$

Then the area under the curve is

$$\text{AUC} = \int_0^1 \text{prec}(\gamma) d\gamma. \tag{8}$$

The higher the value the better the performance according to the evaluated set $\mathcal{D}$.

4.1.3 Multiple target classification

A multiple target classification is a classification with more than two classes C_j. Often, we want to avoid any correlation between the index i and the classes C_i. That is why we introduce the principle one-hot encoding. We define y_{ij} in the following way:

$$y_{ij} = \begin{cases} 1, & \text{if } y_i \in C_j \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Instead of using just one predictor f we estimate one probability estimator f_j for each class assuming c different classes C_j at the same time. This is done because we consider each $f_j(x_i)$ the probability of x_i to be in class C_j. We also have the restriction that

$$f_j(x_i) \in [0,1] \quad \forall\ (x_i, y_i) \in \mathcal{D} \quad \forall f_j \tag{10}$$

is true and

$$\sum_{j=1}^{c} f_j(x_i) = 1 \quad \forall (x_i, y_i) \in \mathcal{D}. \tag{11}$$

Then we take following loss function called the categorical cross entropy:

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} y_{ij} \log(f_j(x_i)) \tag{12}$$

And the predicted classes $C_{i,\gamma}$ ($\gamma \in [0,1]$) with the accuracy acc:

$$C_{j,\gamma} = \{(x_i, y_i) \in \mathcal{D} | f_j(x_i) > f_k(x_i) \forall k \neq j\} \tag{13}$$

$$\text{acc} = \frac{\sum_{j=1}^{c} |C_j \cap C_{j,0.5}|}{n} \tag{14}$$

It is the percentage of correctly classified datapoints.

4.2 Artificial neural networks

In the recent years, artificial neural networks have been become very popular as they can solve complex classification problems and their computations are inspired by the human brain [20].

4.2.1 Perceptron

The easiest neural network is the perceptron consisting only of 1 neuron. Important is the so called activation function. The most common ones are the following:

$$\text{Linear} \quad a(x) = x \tag{15}$$

$$\text{Step function} \quad a(x) = \theta(x) \tag{16}$$

$$\text{Logistic sigmoid} \quad a(x) = \frac{1}{1+e^{-x}} \tag{17}$$

$$\text{Hyperbolic function} \quad a(x) = \tanh(x) \tag{18}$$

$$\text{Rectifier function} \quad a(x) = \max(0, x) \tag{19}$$

The perceptron calculates the prediction in the following

$$f(x) = a\left(\sum_{i=1}^{d} w_i x_i + b\right), \tag{20}$$

where ${x_i}_{i=1}^{d}$ is the d-dimensional feature vector for each element $(x, y) \in \mathcal{D}$, w_i are the weights and b is the so called bias. During the training, process the best weights and the best bias are found. The perceptron can be used for regression problems where the linear function is preferred as activation function or for binary classification purposes. In this case logistic sigmoid is mainly used since the output is between 0 and 1 which can be plugged in in to the cross entropy loss function. The rectifier function is more used for hidden layers in feed-forward-networks due to its fast calculation. Regarding classifications, the purpose of activation functions is mostly to introduce some nonlinear dependency [18].

4.2.2 Gradient descent

In order to find the best perceptron to fit our model, we minimize the loss function by changing our weights w_i and our bias b. That process is called training and is performed over training steps so called epochs. In the first epoch an initial weight vector w^0 and a bias b^0 are taken, the output (20) is calculated and plugged in into the loss function (3,12) for every datapoint (x_i, y_i). After that, the derivative of the loss function with respect to the weights w_i is calculated and denoted as the gradient $\nabla\mathcal{L}$. For the next

steps the new weights and biases are calculated elementwise in the following way,

$$w_i^{j+1} = w_i^j - \alpha \cdot \nabla \mathcal{L}_i \tag{21}$$
$$b_i^{j+1} = b_i^j - \alpha \cdot \nabla \mathcal{L}_i \tag{22}$$

where α is the so called learning rate. It is a factor usually in the range between 0.0001 and 0.1 and it has to be specified before the training process. How long the training process takes and when to stop has also to be decided. Usually a certain number of epochs is taken or until the loss function reaches a certain threshold. There are also more methods which are explained in 2.3 Regularization.

An important improvement used to escape local minima and to save computational time is the so called stochastic gradient descent. There, the loss function is not calculated over all the samples from $\mathcal{D}$ at the same time. Instead $\mathcal{D}$ is divided equally and randomly into so called batches $\mathcal{B}$. Now, in each epoch, the loss is calculated for the batches one after another, and the weights are updated directly. Like the learning rate the batch size has also to be decided in beforehand.

Another common improvement is to use a dynamical learning rate instead of a constant learning rate. The Adam algorithm is designed to reduce the learning rate when a minimum is almost reached. It is based on a running average. The elementwise weight update works in the following way:

$$g_k \equiv \nabla \mathcal{L}_k \tag{23}$$
$$w_k^{i+1} = w_k^i - \alpha \cdot \frac{\hat{m}_k^i}{\sqrt{\nu_k^i} + \epsilon} \tag{24}$$
$$\hat{m}_k^{i+1} = \frac{m_k^{i+1}}{1 - \beta_1} \tag{25}$$
$$\hat{\nu}_k^{i+1} = \frac{\nu_k^{i+1}}{1 - \beta_2} \tag{26}$$
$$m_k^{i+1} = \beta_1 m_k^i + (1 - \beta_1) g_k^i \tag{27}$$
$$\nu_k^{i+1} = \beta_2 \nu_k^i + (1 - \beta_2)(g_k^i)^2 \tag{28}$$

Typical values for β_1, β_2 and ϵ are $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$

4.2.3 Multilayer Perceptron

A common type of neural network is the so called Multiplayer perceptron which is capable to solve difficult classification problems. It consists of different layers and each layer of a certain number of neurons. The number of layers is denoted with m and the number of neurons of each layer with N_l. The output of each

single node is denoted with o_i^j for layer j and node i. The output f_j is computed in the following way:

$$f_i(x) = o_i^m \tag{29}$$

$$o_i^j = a_j \left(\sum_{l=0}^{N_j - 1} w_{lj}^j o_l^{j-1} \right) \text{ for m} \geq j > 1 \text{ and } i \in \{1, ..., N_i\} \tag{30}$$

$$o_i^1 = a_1 \left(\sum_{l=0}^{d} w_{li}^j x_l \right) \text{ for } i \in \{1, ..., N_1\} \tag{31}$$

$$o_0^j = 1 \text{ for all j} \tag{32}$$

The upper indices are not exponents and w_{0j} are the biases. Instead they indicate the layer of the weights and outputs. For every layer one activation function can be chosen. For a_m the same activation function can be chosen which are also possible for the single perceptron. This last layer is also called the output layer. The other layers (hidden layers) can also take ReLu as activation function which has the advantage of low cost computation.

In case of multi-classification we have 1 output layer per class. In order to make sure that the constraints (10) and (11) are fulfilled, following activation function is used, the soft max function which takes all the o_i as input:

$$a_j(\boldsymbol{x}) = \frac{e^{x_j}}{\sum_{k=1}^{c} e^{x_k}} \tag{33}$$

This activation function makes sure that $\sum_{j=1}^{c} a_j(\boldsymbol{x}) = 1$. Together with the learning rate and batch size the number of layers, number of neurons for each layer and activation functions are hyperparameters which have to be selected before the training.

4.2.4 Backpropagation

The calculation of the output is called the forward propagation. After each forward propagation the backpropagation is performed in order to calculate the weight updates for the epoch or the batch in case of stochastic gradient descent. In order to do the gradient descent $\Delta\mathcal{L}$ must be calculated:

$$\nabla\mathcal{L}_{ij} = \frac{\partial\mathcal{L}}{\partial w_{ij}^k} \tag{34}$$

Because the loss functions are linear in the datapoints we can denote the loss for each single datapoint k with l_k. ∇l each weight w_{ij} and in the end the end $\nabla\mathcal{L}_{ij} = \frac{1}{c}\sum_{k=1}^{c} \nabla L_k$ can be calculated. In order to

avoid too many indices, we just write L for one single datapoint. Based on the chain rule:

$$\frac{\partial L}{\partial w_{jk}^l} = \frac{\partial z_j}{\partial w_{jk}^l} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial L}{\partial a_j^l} \tag{35}$$

$$\frac{\partial L}{\partial a_k^{l-1}} = \sum_{j=0}^{N_{l-1}} \frac{\partial z_j^l}{\partial a_k^{l-1}} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial L}{\partial a_j^l} \tag{36}$$

$$\text{with } z_j^l = \sum_{k=0}^{N_l} w_{jk}^l a_k^{N_{l-1}} \tag{37}$$

$$\text{and } a_k^0 = x_k \tag{38}$$

4.3 Regularization

Up until now, our objective has been to find the best estimate, denoted as $f_\rho^{\mathcal{L}}$, based on our training dataset $\mathcal{D}$. However, it is important to recognize that $\mathcal{D}$ is only a finite subset sampled from the underlying distribution ρ, and therefore it may not fully represent the entire $\mathcal{Z}$-space. Instead of minimizing only $\mathcal{L}_\mathcal{D}$ we look for a low generalization error ϵ. Having a low $\mathcal{L}_\mathcal{D}$ with a worse ϵ is called overfitting. The model becomes excessively complex and overly adapted to the specific set $\mathcal{D}$ used. This can lead to poor performance when applied to new, unseen data $\mathcal{T}$ with $\mathcal{T} \cap \mathcal{D} = \emptyset$ [17].

Regularization techniques aim to address this problem by reducing the complexity of the model. The goal is to strike a balance where the model is complex enough to capture the underlying truth in unseen data, while avoiding excessive complexity. One common approach is to introduce regularization criteria, such as the correlation criterion and the evidence-based criterion, in addition to other well-known regularization methods.

In regression statistical tests such as the chi-squared test can be used in order to determine how good the model generalizes. In classification other methods are used:

In order to determine how good the model performs on new untrained data in classification a specific amount of the data is held out during the training process. In the end it is used to test the performance of the model on new unseen data. Typically the test set contains ca. 20% and the train set 80% of the entire dataset. The model is trained using the train set and evaluated using the test set. In the evaluation process only the accuracy is calculated. An advantage of taken more data as the test set is that the test is more representative. On the other hand, less available information is used for the training process which leads to worse performance.

4.3.1 L2 regularization

The most common approach to overcome overfitting is the so called L2 regularization, also known as Ride decay or weight decay. Our loss function is modified in the following way:

$$\hat{\mathcal{L}} = \mathcal{L} + \frac{\beta}{2} \sum_{j=1}^{m} \sum_{l=1}^{N_j} \sum_{i=0}^{N_{j-1}} (w_{il}^j)^2 \tag{39}$$

A regularization term is added, the sum of all the squared weights multiplied with the regularization rate $\frac{\beta}{2}$.

The update rule works in the following way, the upper indices are the steps:

$$w_i^{j+1} = w_i^j - \beta(\alpha w_i^j + \nabla\mathcal{L}(w_i^j) \tag{40}$$

$$w_i^{j+1} = (1 - \alpha\beta)w_i^j - \alpha\nabla\mathcal{L}(w_i^j) \tag{41}$$

The weights are suppressed a bit in each step of the gradient descent process. The value β is an additional hyperparameter value which has to be selected in the beginning.

4.3.2 L1 regularization

The L1 regularization, also known as Tikhonov regularization or Lasso regularization, is similar to the L1-regularization. The loss function is modified by adding following:

$$\hat{\mathcal{L}} = \mathcal{L} + \frac{\beta}{2}\sum_{j=1}^{m}\sum_{l=1}^{N_j}\sum_{i=0}^{N_{j-1}}|w_{il}^j| \tag{42}$$

Therefore, the gradient is the following:

$$\nabla\hat{\mathcal{L}}_i = \beta\ \text{sign}(w_i) + \nabla\mathcal{L}_i \tag{43}$$

Also the value β has to be chosen before the training [18].

4.3.3 Drop out

With the drop out regularization method for each neuron there is a probability that is is deactivated in the way, that all weights of the next layers which depend on the neuron get the value zero. For each training epoch another set of neurons is deactivated whereas all of the neurons are activated when evaluating. Using drop out leads to less dependencies of the network performance of a few neurons which enhances generalization [18].

4.3.4 Early stopping with a validation-based criterion

Until now, determining when to stop the training process has been a predefined decision. However, during the evaluation on the test set, it is often observed that the test loss starts increasing while the train loss continues to decrease. To address this, the idea is to identify the optimal epoch to stop the training process.

One common approach involves dividing the train set into two subsets: the train set itself and a validation set. The training is initially performed on the train set, and the validation loss is calculated after each step. The epoch at which the validation loss reaches its minimum is tried to determine. Either the training process can directly stop when the validation loss starts to increase again or other methods such as a patience criterion or threshold value can be used in order to make sure that the training does not stop in a local minimum or when having small fluctuations. Prechelt compared the different methods [19].

Typically the size of the validation set is chosen to be approximately equal to the size of the test set. This ensures that the validation set provides a reliable estimate of the model's performance on unseen data. In the paper of the Max-Planck-Institute in Tübingen they name the criterion "arguable the gold-standard

of early stopping" since there the training process has less data to train on. "The trade-off is not easily resolved"[17].

4.3.5 Early stopping with Gradient correlation criterion

The undergraduate student Daniel Lizotte presented in his thesis another approach: using a so called gradient correlation coefficient. For the next the loss function is denoted as dependent on the input set: $\mathcal{L}_{\mathcal{D}}$. Because of the linearity of the loss function with respect to the datapoints and the gradient can be expressed as a sum of the losses of each datapoint:

$$\nabla\mathcal{L}_{\mathcal{D}} = \sum_{(x,y)\in\mathcal{D}} \nabla\mathcal{L}_{\{(x,y)\}} \tag{44}$$

In case of neural networks all the weights and biases can just be aligned in one vector.

$$g_i = \nabla\mathcal{L}_{\{(x_i,y_i)\}} \text{ for } i \in \{1,2,3,...,n\} \tag{45}$$

with $n = |\mathcal{D}|$ the number of datapoints. When taking the square of the absolute value of the sum

$$|\sum_{i=1}^{n} g_i|^2 = \sum_{i=1}^{n} |g_i|^2 + \sum_{i,j\in\{1,2,3,...,n\, i\neq j\}} g_i \cdot g_j \tag{46}$$

with $g_i \cdot g_j$ as a dot product.

During our training process we assume that if our loss is far away from our global minimum most of the datapoints yield a gradient which is looking in the similar direction in weight space. Thus, the correlation is high and $|\sum_{i=1}^{n} g_i|^2 \gg \sum_{i=1}^{n} |g_i|^2$. When coming closer to the minimum the gradients have different directions and the components weight out each other and $|\sum_{i=1}^{n} |g_i|^2 > |\sum_{i=1}^{n} g_i|^2$. We can set it together by defining the gradient correlation criterion G:

$$G = \frac{|\sum_{i=1}^{n} g_i|^2}{|\sum_{i=1}^{n} |g_i|^2|} \tag{47}$$

According to the gradient correlation criterion (GC-criterion) the training stops when $G < 1$. Lizotte tested the GC-criterion for classification of Gaussian clouds. He used neural networks with one layer, a varying number of nodes, a varying learning rate, batch size, number of data points and distance between cloud centers changing the problem difficulty. He looked at the train loss and the convergence process. The threshold of $G = 1$ yielded stable results when changing the problem difficulty, similar stopping epochs when varying the number of nodes, problematic behaviour for learning rates larger than 0.1, different stopping epochs for changing batch sizes but similar loss and similar epochs when changing the number of datapoints. Except the case with a learning rate of 1 it always stopped at reasonable times.

4.3.6 Early stopping with an evidence based criterion

The evidence based criterion is a stopping criterion presented by researchers of the Max Planck institute for intelligent systems in Tübingen (Germany) in 2017 [17]. First the generalization error of 2.1 is taken: $\epsilon(f) = \int_{\mathcal{Z}} \mathcal{L}(y, f(x))d\rho$. When taken a randomly selected $\mathcal{D} \subset \mathcal{Z}$ $\mathcal{L}_{\mathcal{D}}$ is an estimator of $\epsilon(f)$ according some f. Because $\mathcal{L}_{\mathcal{D}} = \sum_{(x,y)\in\mathcal{D}} \mathcal{L}(x, f(x))$ and likewise for the gradient $\nabla\mathcal{L}$ we can interpret the

probability distribution for $\mathcal{L}_\mathcal{D}$ as normal distributed around $\epsilon(f)$

$$\mathcal{L}_\mathcal{D} \sim \mathcal{N}\left(\epsilon(f), \frac{\Lambda(f)}{|\mathcal{D}|}\right) \tag{48}$$

with the variance $\frac{\Lambda(f)}{|\mathcal{D}|}$ and $\Lambda(f) = var_{(x,y)\sim\rho}\mathcal{L}(f(x), y) \in \mathbf{R}$. Likewise with $\nabla\mathcal{L}_\mathcal{D}$

$$\nabla\mathcal{L}_\mathcal{D} \sim \mathcal{N}\left(\nabla\epsilon(f), \frac{\Sigma(f)}{|\mathcal{D}|}\right) \tag{49}$$

with $\Sigma(f) = cov_{(x,y)\sim\rho}\nabla\mathcal{L}(f(x), y) \in \mathbf{R}^{nxn}$. Since ρ is unknown in general $\Sigma(f)$ can be estimated with

$$\hat{\Sigma}(f) = \frac{1}{|\mathcal{D}| - 1}\sum_{(x,y)\in\mathcal{D}}(\mathcal{L}(f(x), y) - \nabla\mathcal{L}_\mathcal{D}(f))^{*2} \tag{50}$$

^{*2} stands for elementwise squared. Because $\mathcal{L}$ and ϵ have different behaviour in general they do not reach the minimum for the same f but are close close to the minima. The function f which minimizes $\mathcal{L}$ can be written as $f^* = argmin_f(\mathcal{L}(f))$ and $\Sigma^*(w) := \Sigma(w^*)$. Then the probability distribution for $\nabla\mathcal{L}_\mathcal{D}$ when ϵ is minimized:

$$p(\nabla\mathcal{L}_\mathcal{D}|\nabla\epsilon = 0) = \mathcal{N}\left(\nabla\mathcal{L}_\mathcal{D}; 0, \frac{\hat{\Sigma}^*}{|\mathcal{D}|}\right) \tag{51}$$

Because Σ is also unknown we can estimate it with $\hat{\hat{\Sigma}}$. When also treating the gradient dimensions independent of each other our probability distribution is written as

$$p(\nabla\mathcal{L}_\mathcal{D}(f)) \approx \prod_{k=1}^{m}\mathcal{N}\left(\nabla\mathcal{L}^k_\mathcal{D}(f); 0, \frac{\hat{\Sigma}_k(f)}{|\mathcal{D}|}\right). \tag{52}$$

m is the number of all the weights and biases, the dimension of the weight space. Now we use techniques from hypothesis testing, we stop when:

$$\log p(\nabla\mathcal{L}_\mathcal{D}) - \mathbb{E}_{\nabla\mathcal{L}\sim p}[\log p(\nabla\mathcal{L}_\mathcal{D})] > 0 \tag{53}$$

Combining it with (52) leads to

$$\frac{2}{m}[\log p(\nabla\mathcal{L}_\mathcal{D}) - {}_{\nabla\mathcal{L}_\mathcal{D}\sim p}\left[\log p(\nabla\mathcal{L}_\mathcal{D})\right]] = 1 - \frac{|\mathcal{D}|}{m}\sum_{k=1}^{m}\left[\frac{(\nabla\mathcal{L}^k_\mathcal{D})^2}{\hat{\Sigma}_k}\right] > 0. \tag{54}$$

The stopping point is reached when the gradient is one standard deviation:

$$\sum_{k=1}^{m}\frac{(\Delta\mathcal{L}^k_\mathcal{D})^2}{\hat{\Sigma}_k} = \sum_{k=1}^{m}\frac{\hat{\Sigma}_k}{|\mathcal{D}|\cdot\hat{\Sigma}_k} = \frac{m}{|\mathcal{D}|} \tag{55}$$

In order to get the same threshold value of 1 as for the gradient correlation criterion, the Evidence based criterion is written as:

$$\text{EB} = \frac{|\mathcal{D}|}{m}\sum_{k=1}^{m}\left[\frac{(\nabla\mathcal{L}^k_\mathcal{D})^2}{\hat{\Sigma}_k}\right] \tag{56}$$

After each training step this rule is checked. Whenever this is true the training process is immediately stopped and the best guess is reached. Lassner, Mahsereci, Hennig and Bassner [17] tested it emperically for a regression problem with artificially generated data, a logistic regression method on the Wisconsin Breast cancer dataset [22], a multilayer perceptron on the MNIST dataset and a logistic regression and shallow net method on another dataset called SECTOR. In the toy problem they also generated a scenario where no overfitting occured and the training would have stopped to early according to the evidence based criterion. Besides this case the evidence based criterion seemed to be a viable alternative to the common validation-based early stopping criterion. The evidence based criterion is also noted as EB-criterion in the following.

4.4 Hyperparameter optimization

Hyperparameter optimization describes techniques which are used in order to find the best hyperparameter which leads to the best performance on a specific data set or for general datasets.

4.4.1 Cross-validation

After dividing the data into a training and a test set for different hyperparameters, models could be trained and tested afterwards and the test performance could be compared, either using the test loss or test accuracy. On the other hand, the performance depends strongly on the test data in case and can thus be biased. That is why another method can be used: cross-validation. The entire dataset is divided in k parts and for each part k the other k-1 parts are trained and evaluated on the k'th part. The losses or accuracies can be saved in the end and the mean gives a more reliable measure how good the model performs on new data. When trying different hyperparameter search methods, for each set of hyperparameters the cross-validation can be used [18].

4.4.2 Grid search

In grid search possible discrete values for hyperparameters are selected in advance and written in different discrete sets $A, B, C, ...$ with discrete values with $A = \{a_1, a_2, ...\}, B = \{b_1, b_2, ...\}, ...$ Then the cross product is taken over all of them $Z = AxBxCx....$ The full dataset is divided into a train set, test set and validation set. The training is performed over the test set for each of the z and is evaluated on the validation set by calculating the validation loss. Then the hyperparameters with the lowest validation loss are taken and used in order to train the model again using both the train and the validation set for training. After the training, the chosen model is evaluated on the test set which measures the performance [18].

4.4.3 Random search

When using random search instead of taking the cross product, we select random distributions for all of the possible values $a \in A$ and select Z by randomness. One advantage of this method is that the selection is not restricted by certain values [18].

4.4.4 Tree Parzen Estimator

The Tree Parzen Optimizer (TPE) algorithm was presented in 2011 by Bergstra, Bardenet, Yoshua Bengio and Balázs Kégl [15]. It has the advantage that it is an iterative finding algorithm that takes into account all the prior trials when determining the best set of hyperparameters. Instead of modelling

the best hyperparameter according to the calculated losses $p(a|\mathcal{L})$, it models a function of the losses in dependence of a hyperparameter a $p(\mathcal{L}|a)$ for each hyperparameter a independently. First, all possible ranges of hyperparameters or possible ranges have to be defined. Then at least two random choices of hyperparameters are taken and the training processes are done using these. Let be $\gamma \in [0, 1]$, most commonly is $\gamma = 0.5$. After that, the different configurations are sorted by their loss

$$((a_0, \mathcal{L}_0), (a_1, \mathcal{L}_1), (a_2, \mathcal{L}_2), ...) \tag{57}$$

with $\mathcal{L}_0 \leq \mathcal{L}_1 \leq \mathcal{L}_2 \leq ...$ and divided into one set with approximately γ times the number of total trials with the lower $\mathcal{L}$ losses and one with approximately $(1 - \gamma)$ times the number of the higher losses. Then, kernel density estimation is used in order to create two likelihood functions $p_l(\mathcal{L}|a)$ and $p_h(\mathcal{L}|a)$. For each prior selected hyperparameter value a , a Gaussian is taken with the value as its mean and the distance to the neighbor which is farther away as standard deviation and truncated Gaussians are used for the constraints regarding the possible range. Then, the Gaussians corresponding to the hyperparameters a with the lower losses are summed up and normalized, that is the function $p_l(\mathcal{L})$ and the Gaussians corresponding to the hyperparameters a with the higher values are summed and normalized to get $p_h(\mathcal{L})$. We assume that a next good guess for a hyperparameter has a high value according to $p_h(\mathcal{L}|a)$ but a low value in $p_l(\mathcal{L})$. That is why a function called promisingness is calculated by taking the quotient of both distributions:

$$\text{promisingness} = \frac{p_h(\mathcal{L}|a)}{p_l(\mathcal{L}|a)} \tag{58}$$

The hyperparameter which maximizes the promisingness is selected for the next trial. For certain hyperparameters such as learning rate, the likelihood functions can also created in log-space and for categorical values categorical distributions can be applied.

5 Data and Methods

5.1 Datasets and Preprocessing

Following datasets were tested:

- Iris Dataset
- Jet tagging Dataset
- Credit Card Fraud Dataset
- Titanic Dataset
- Obesity Dataset
- Pima Indians Diabetes Database

5.1.1 Iris dataset

The first dataset which was tested is the Iris dataset. It was invented by the British statistician and biologist Ronald Fisher in 1936 and it contains information about three different Iris species: Iris Setosa,

Iris Versicolor and Iris Virginica. The size is 150 observations, 50 belonging to each class. Each sample consists of 4 numerical features: sepal length, sepal width, petal length, petal width in cm [14]. The input data was standardized: the mean of all feature vectors was subtracted and divided by the standard deviation. This dataset was divided in three subsets: the training set with 60% of the entire set and the test and validation set with 20% each.

5.1.2 Credit card fraud dataset

The Credit card fraud dataset comprises data on 284,807 transactions conducted by European cardholders in 2023, out of which 492 transactions are classified as fraudulent, accounting for only 0.172% of the total. In order to ensure data security, the dataset excludes specific transaction details, retaining only the time elapsed since the first transaction (measured in seconds) and the transaction amount. Additionally, 28 other features are included in the dataset, derived from a Principal Component Analysis (PCA) transformation [**?**]. For our purposes in order to speed up calculations, only 2000 samples were randomly chosen with 400 fraud ones. Subsequently, the input data was also standardized and divided into training, and test set with a 60-20 ratio. The fraud samples and the non-fraud samples were divided separately because of the imbalance of the dataset. After that, the datapoints with the label 1 was upsampled, twice duplicated in order to get a better balance between fraudulent and non fraudulent samples.

5.1.3 Titanic dataset

The Titanic dataset contains information about the passenger of the Titanic and whether they survived or not. The features are passenger ID, class, name, sex, age, number of siblings/spouses aboard the Titanic, number of parents or children aboard the Titanic, how much they paid for the journey, the ticket number and the port of embarkation. In total, there are 891 training datapoints with known survived label and 418 without for test purposes [10]. The features Passenger ID, name and ticket were just removed and in the sex features male was replaced with 1 and female with 0. Additionally the places of embarkation were replaced in the following way: Cherbourg $\rightarrow$ -1, Southhampton $\rightarrow$ 0, Queenstown $\rightarrow$ 1. Missing values were replaced with the mean of each feature vector. After that, the data was also standardized and divided in a training, validation and test set with the same ratios as before.

5.1.4 Obesity dataset

The Obesity dataset is used in order to estimate the Obesity levels of people based on different attributes such as weight, height, sex, if they smoke, how much they drink, which transportation they use, if they eat frequently high caloric food, if they eat vegetables, how many meals per day, how much water they drink, if they monitor their calories daily, how often they do physical activity and how much they use technological devices such as cell phones, video games, computer and others. 485 people were asked through an online page survey. With a tool called Weka and a filter called SMOTE, 1626 synthetic datapoints were generated in order to obtain a more balanced dataset [12]. The categorical features were changed to numerical ones in the following way:

- Gender: Female $\rightarrow$ 0, Male $\rightarrow$ 1
- family history with overweight: no $\rightarrow$ 0, yes $\rightarrow$ 1
- if they eat caloric food frequently: no $\rightarrow$ 0, yes $\rightarrow$ 1

- how often they have food between meals: no → 0, sometimes → 1, frequently → 2, always → 3
- if they smoke: no → 0, yes → 1
- if they monitor their calories: no → 0, yes → 1
- how often they drink alcohol: no → 0, sometimes → 1, frequently → 2, always → 3
- their main way of transportation: bike → 0, walking → 1, public transportation → 2, motorbike → 3, automobile → 4

Additionally, the feature height was dropped, otherwise the labels could be easily just calculated using the weight and height because the labels originate from the weight and the height. The dataset was standardized and divided into training, validation and test set with the ratio 60-20-20.

5.1.5 Diabetes dataset

The Pima Indians Diabetes Database deals with the disease Diabetes. It consists of data from 768 women of the Pima Indian tribe and contains following information:
how many times they have been pregnant, glucose concentration according to a 2 hours oral glucose tolerance test, their diastolic blood pressure, their Triceps skin fold thickness, 2-Hour serum insulin, their BMI, their value of the Diabetes pedigree function, their age and if they are diabetes positive or negative. The data stems from the National Institute of Diabetes and Digestive and Kidney Diseases in the US [6]. The data was also standardized and divided into training, validation and test set with the ratios 60-20-20.

5.1.6 Wisconsin Breast cancer dataset

The Breast cancer Wisconsin dataset deals with the prediction of Breast cancer according to the measured features of each cell nucleus: radius (mean of distances from center to points on the perimeter, texture (standard deviation of gray-scale values), perimeter, area, smoothness (local variation in radius lengths), compactness (perimeter^2 / area - 1.0), concavity (severity of concave portions of the contour), concave points (number of concave portions of the contour), symmetry, fractal dimension ("coastline approximation" - 1). The database is available at the University of California Irvine (UCI) Machine Learning Repository [22]. The feature sample ID was removed and the data was standardized and divided into training, validation and test set with the ratio 60-20-20.

5.2 Neural network tools

All the neural networks were used in python [7] with Jupyter notebook jupyternotebook and Colab [1]. For preprocessing purposes, the libaries Pandas [5] and Numpy [3] were used. For the calculations of the neural network, Pytorch [8] was used and for the analysis and the plots Numpy, Matplotlib.pyplot [2] and Seaborn [9]. Additionally, for the hyperparameter optimization with TPE the library Optuna [4] was applied. For saving the progress of the optimizations because they took long computational time, the library Pickle was used.

5.3 Hyperparameter test

The different stopping algorithms gradient correlation criterion (GC-criterion), evidence based criterion (EB-criterion) and validation criterion (val-criterion) were tested on the 7 datasets presented in 3.1. For

Table 1: Details of dataset and fixed hyperparameters networks

Dataset	Iris	Credit Card Fraud	Titanic
Dimension	4	30	8
Training size	90	2100	534
Test size	30	400	179
Validation size	30	700	178
No. of layers	3	3	3
No. of nodes first layer	50	100	50
No. of nodes second layer	30	60	30
No. of nodes third layer	20	40	20
Activation function hidden layers	tanh	tanh	tanh
Activation function output layer	soft max	sigmoid	sigmoid
No. of of classes	3	2	2

Table 2: Details of datasets and fixed hyperparameters of networks

Dataset	Obesity	Diabetes	Breast Cancer
Dimension	15	8	9
Training size	1266	460	418
Test size	423	154	140
Validation size	422	154	140
No. of layers	3	3	3
No. of nodes first layer	100	50	50
No. of nodes second layer	50	30	30
No. of nodes third layer	40	20	20
Activation function hidden layers	tanh	tanh	tanh
Activation function output layer	soft max	sigmoid	sigmoid
No. of classes	7	2	2

every dataset, the Adam algorithm was used with the common Adam-parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Multilayer perceptrons with following specifications were used: For each dataset 3 hidden layers and one output layer. For the activation functions for the hidden layers the tangens hyperbolicus was chosen. In case of binary classification (Titanic, Diabetes, Credit Card Fraud, Wisconsin Breast Cancer), a sigmoid output activation function with a binary cross entropy loss function was used and in the case of multilayer classification a softmax activation function with the cross entropy loss as loss function. Table 1 and 2 display the detailed properties of the datasets and the netowrks, table 3 the ranges of the batch sizes in the test.

The batch sizes and learning rates were changed. In all of the configurations the learning rates took the values: 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.05, 0.1. Following different batch sizes were taken: The training took up to 6400 epochs. In the first step, the construction set

Table 3: Tested batch sizes according to the dataset

Dataset	**batch sizes**
Iris	5, 10, 15, 30
Credit card fraud	50, 100, 200, 400
Titanic	12, 23, 45, 89, 178
Obesity	50, 100, 200, 400
Diabetes	20, 39, 77, 154, 307
Breast Cancer	11, 22, 44, 88, 175, 349

(training set and validation set together) was used and after the training of all the batches in each step the EB-criterion and the GC-criterion was calculated and saved. Additionally, the model was evaluated on the test set after each epoch and its test accuracy was calculated and saved.

After that, the training was repeated with the training set and evaluated with the validation set after each epoch. No criteria were calculated, instead only the epoch with the maximum of the validation accuracy was saved. This version of validation-based stopping was used in order to compare the different stopping epochs using the similar training process.

Subsequently, the epochs with corresponding test accuracy were recorded in which the GC correlation criterion and the evidence based criterion became lower than 1 for the first time, also the test accuracy was recorded for the corresponding epochs and the maximum test accuracy with corresponding epoch was saved. In case of the Credit card fraud dataset, instead of the accuracy, the area under the curve (AUC) of the test set was recorded because of the imbalance in the dataset.

5.4 Hyperparameter optimization with Tree Parzen Optimizer

In order to find the best hyperparameter, a Tree Parzen Estimation was conducted for the datasets Obesity, Diabetes, Titanic and Credit card fraud. In order to obtain meaningful results that are less dependent on the random selection of a test set, 10-k cross-validation was used in each trial. Up to 160 trials were used, the learning rate in the log space and the batch size were optimized. The used maxima and minima of the continous ranges of the hyperparameter space were close the highest and lowest values used earlier in the hyperparameter test. One optimization was performed for each of the three stopping criteria and each of the four datasets: validation-based stopping, gradient correlation criterion and evidence based criterion. The four were chosen because they are the more challenging classification problems. In the end, the best combinations are saved.

With the optimal choices for batch size and learning rate, all hidden layers become dropout layers and L_2 regularization is added in order to test if the results can be enhanced using these regularization methods. The dropout probabilities in the range of $p \in [0, 0.5]$ and L_2-weights in the log space with range $w \in [0.0001, 0.01]$ are optimized.

6 Results

6.1 Hyperparameter test

The figures 7-18 show the results of the hyperparameter test. First, it is compared when the different stopping algorithms would have stopped, then the different performances are compared.

The figures 7, 9, 11, 13, 15 and 17 shows four tables for each dataset. Firs,t the indices with a maximum test accuracy are displayed which is also known as maximum achievable performance from no on; then, the differences to the maximum test accuracy indices are compared for each of the three stopping criteria. Comparing the epochs where the optimal accuracy is reached, all datasets except the Diabetes dataset have in common that the training process would have taken longer for learning rates lower than 0.001 as the learning rate decreases and the batch size increases. For the Diabetes dataset, no tendency

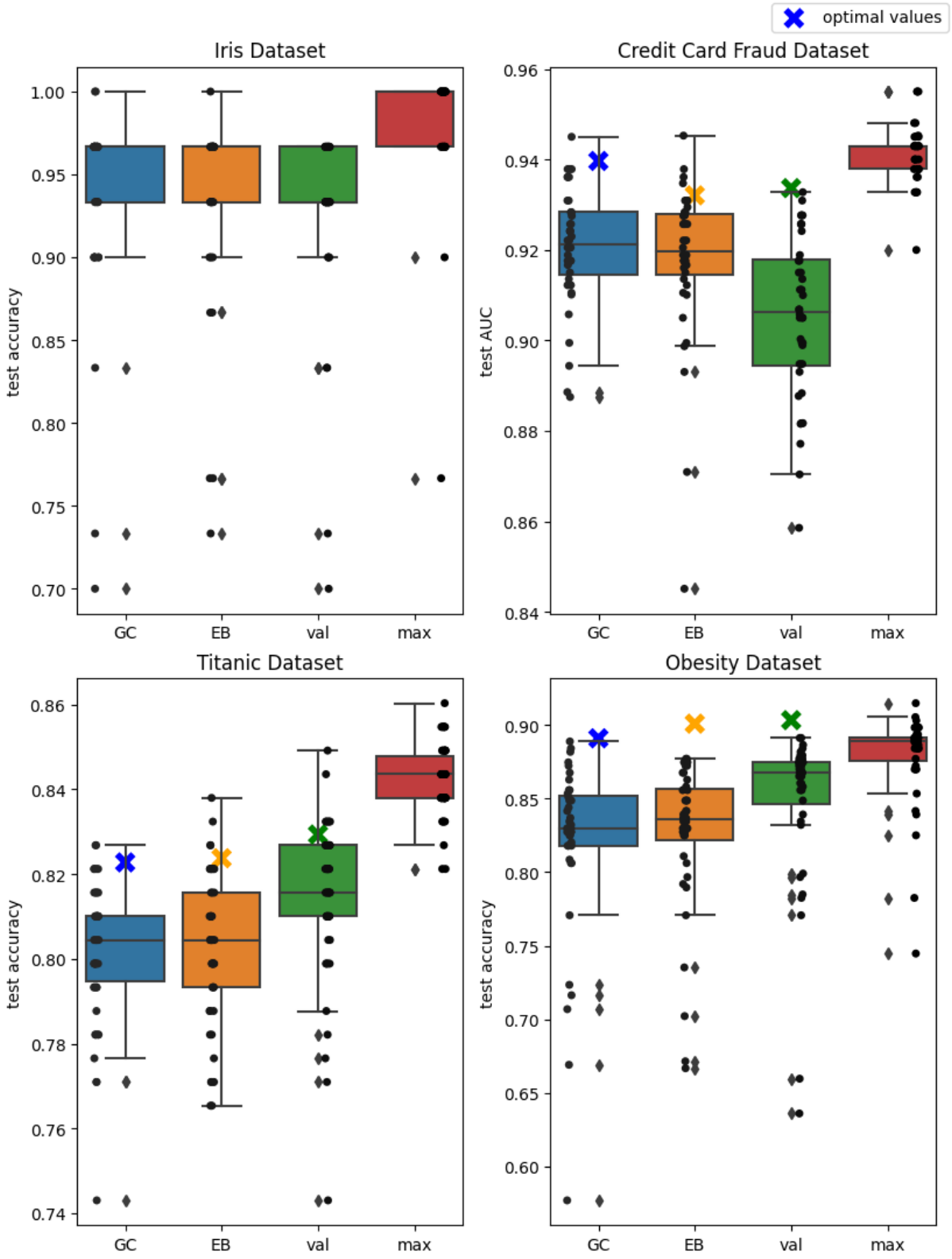

Figure 1: Comparisons of the performance of the three different stopping algorithms with the maximum achievable performances. The third column stands for the validation-based criterion, the fourth for the maximum achievable performance. The crosses indicate the performance with the optimized parameters.

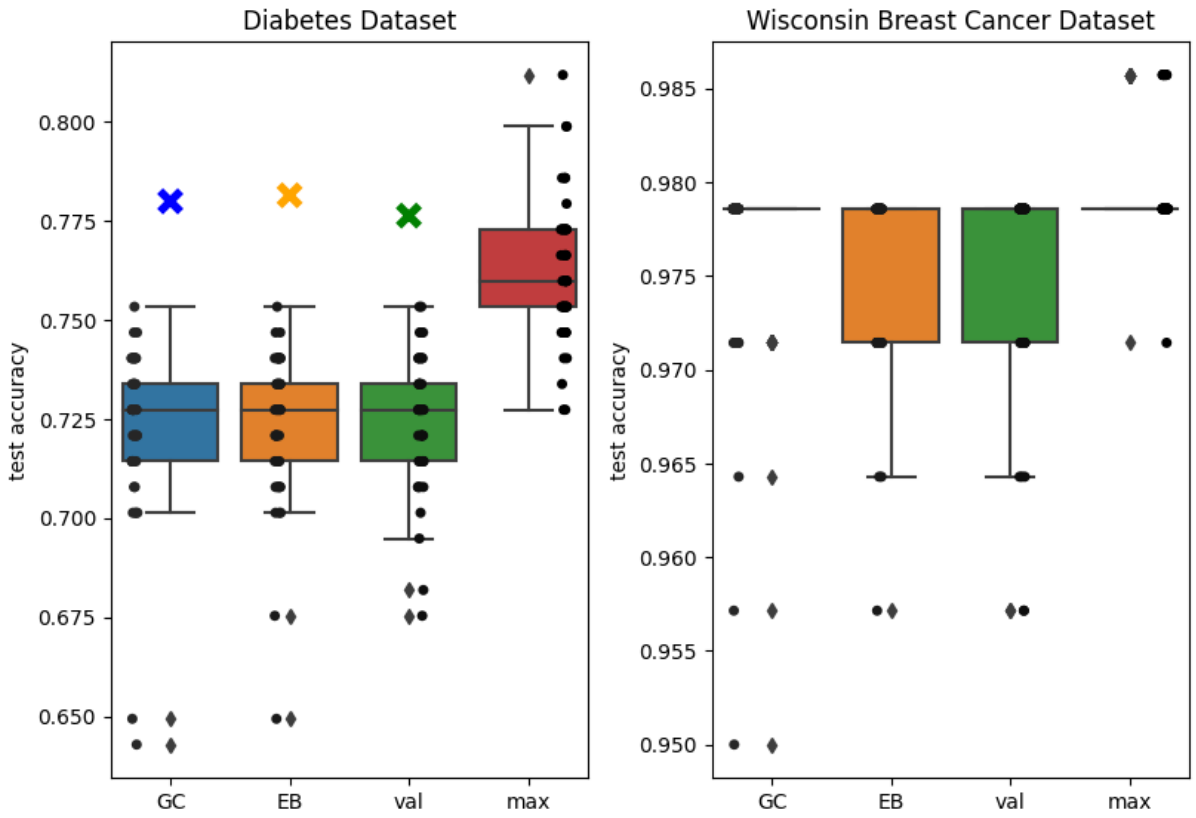

Figure 2: Comparisons of the performance of the three different stopping algorithms with the maximum achievable performances. The third column stands for the validation-based criterion, the fourth for the maximum achievable performance. The crosses indicate the performance with the optimized parameters.

can be found because it is characterized by many fluctuations for both higher and lower learning rates (Fig. 15 a). The Credit card fraud, Titanic, Obesity have fluctuations more for higher learning rates. On the other hand, the Breast cancer dataset and the Iris dataset and the Diabetes dataset show also fluctuations for other learning rate ranges. In general, higher batch sizes and lower learning rates lead to more epochs. There are more fluctuations for higher learning rates, which is intuitively expected because higher learning rates lead to stronger weight updates that differ more from each other.

In most of the datasets, the training would have stopped according to the GC-criterion and the EB-criterion at similar epochs. However, in the case of the Iris dataset, training would have stopped almost 12 times later on average according to the EB-criterion than according to the GC-criterion (Fig. 7). Other exceptions are using the Breast cancer for learning rates 0.005 or lower (Fig. 17) and using Credit card fraud dataset for learning rates 0.05 and 0.1 (Fig. 9). Mostly, the training would have stopped later according to the validation criterion than according to the EB- or GC-criterion. Exceptions are fluctuations or higher learning rates in the Credit card fraud dataset and the batch sizes 200 and 400 with learning rates 0.005-0.1 in the Obesity dataset. In most of the cases, the training would have stopped before the maximum was reached according to the EB- and GC-criterion and later according to the validation criterion. In the few cases where the training would have stopped later according to the EB-criterion in the Obesity dataset it would have stopped later than the epoch where the maximum accuracy was reached.

The figures 8, 10, 12, 14, 16 and 18 show the different accuracy results for the different datasets. Additionally, Fig. **??** compares the accuracy (or AUC) results for the different stopping criteria with the maximum achievable accuracy.

The in the following calculated averages provide insights into the determined values of the chosen configurations; however, they are highly dependent on the chosen hyperparameters and the random selection of the test set. Therefore, they should not be overinterpreted.

If the model had stopped at the epoch with the best accuracy it would have yielded values close to 1 for the Iris dataset (Fig. 8 a). Using all of the three criteria, there would have been more fluctuations for higher learning rates (Fig. 8 b-d). In general, the GC- and the EB-criterion would have led to better stoppings whereas the validation criterion would have slightly overfitted. The average accuracy of the GC-criterion is 0.9383, of the EB-criterion 0.9350 and of the validation-based criterion 0.9258 with a maximum accuracy of 0.9825. Fig. 1 shows that the maximum accuracy are higher than the achieved ones when stopping with a criterion. Fig. 3 shows a case where the training would have stopped slightly too early according to the EB- and GC-criterion but it would not have stopped according to the validation-based criterion.

The best results for the Credit card fraud dataset are rather constant in general (Fig. 10 a). As the training would have stopped latest according to the validation-based stopping criterion and it has still the worst results with an average AUC of 0.9050 it tends to overfit (Fig. 10 d). The GC- and EB-criterion would have led to better results with average AUC of 0.9204 and 0.9178 respectively (Fig. 10 b-c). The maximum average AUC is 0.9415. In the case of a batch size of 200 and learning rate of 0.0001 it might have stopped even after 3200 epochs if more epochs had been used in order to find the optimal stopping epoch. This achieved value of 0.926 is close to the values of the two other criteria for the same hyperparameters. Fig. 4 shows this case. All of the three plots have high fluctuations which are

caused by the relatively high learning rate of 0.05. Whilst the GC- and the val-criterion stop earlier the threshold of 1 is never reached for the EB-criterion. On the other hand the accuracy does not decrease in the end, no significant overfitting occurs.

In the Diabetes dataset higher learning rates would have led to better results if the training would have stopped in the epoch with the best result (Fig. 16 a). All of the stopping criteria would have yielded similar results, the results of the GC- and the EB- criterion would have been closer to each other (Fig. 16 b-d). Except for low learning rates and high batch sizes all of the three criteria would have led to an underfitting. The average accuracy according to the EB-criterion would have been 0.7229, according to the GC-criterion 0.7242 and according to the validation-based criterion 0.7254 with a maximum achievable average accuracy of 0.7614. Fig. 3 shows an example where the training would have stopped the earliest according to the evidence based criterion.

In the process of training on the Titanic dataset, the GC- and the EB- criterion would have led to an underfitting whereas the validation criterion would result an slightly overfitting with a few percent difference to the maximum achievable accuracy (Fig. 12). The GC- and EB-criteria would have led to similar results as they stop at similar epochs. The only obvious difference is that the EB-criterion has a few more fluctuations. In most of the cases, the validation-based criterion would have led to better results than the GC- or the EB-criterion. The average accuracy with the validation-based criterion would have been 0.8143, the one with the GC-criterion 0.8022 and the one with the EB-criterion 0.8021 while having an average maximum achievable accuracy of 0.8420.

When training on the Obesity dataset, the EB- and GC criterion would have led to an overfitting and would have brought about worse results than the validation-based criterion (Fig. 14). On the other hand, for learning rates 0.005 and above with batch sizes 200 and 400 the training would have led to better results according to the EB- and GC- criterion than to the validation criterion. The average accuracy according to the validation-based criterion would have been 0.8470 while according to the GC- and EB-criterion it would have been 0.8186 and 0.8249 respectively with a maximum achievable accuracy of 0.8778.

With a few exceptions all of the three criteria would have led to results close to the maximal possible results for the Breast cancer dataset, in most cases 0.979 accuracy (Fig. 18). The taken averages also show 0.9758 for the EB-criterion, 0.9763 for the GC-criterion and 0.9757 for the validation-based criterion. Fig. 6 shows the behaviour of the training process. As soon as the plateau of 0.979 is reached the training is stopped. A better accuracy is 1 which is probably hard to achieve. All of the three criteria are also able to save training time.

The averages of all the accuracies/AUC when stopping according to the GC-criterion is 0.8631, according to the EB-criterion 0.8631 and according to the validation-based criterion 0.8656. Thus, the validation-based criterion still brings about a slightly better accuracy/AUC than the EB- and GC-criterion.

6.2 Hyperparameter optimization with the Tree Parzen Optimizer

The different optimal parameters batch size and learning rate stemming from the Tree Parzen Optimize are shown in table 4 and their approximate region in Fig. 10, 12, 14 and 16. Even though the results of the GC- and EB-criterion are rather close except for the Diabetes criterion in the hyperparameter test,

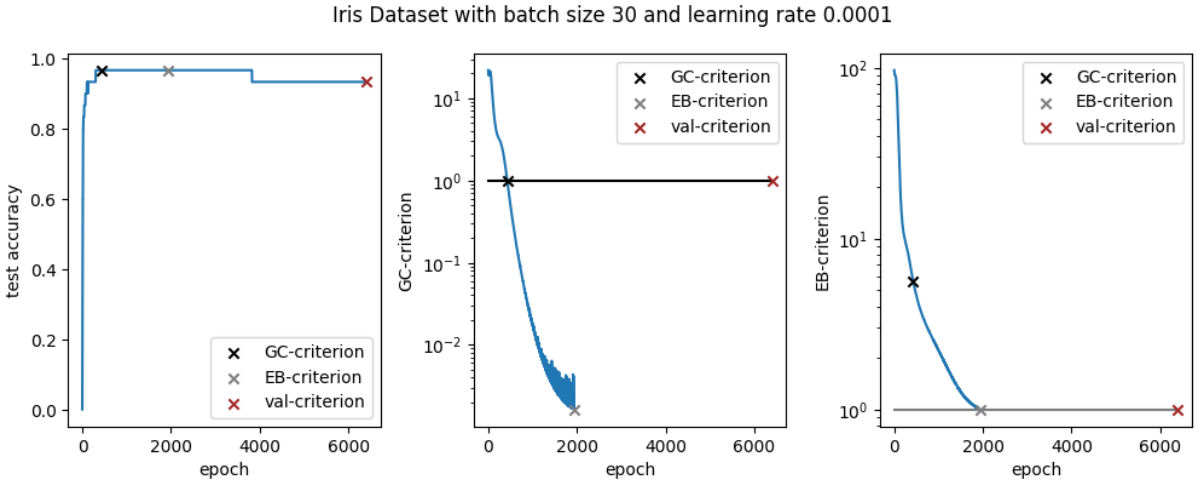

Figure 3: Training behaviour of the Iris Dataset with a batch size of 30 and a learning rate of 0.0001. The black and the gray line indicate the threshold after which the training would stop according to the respective criterion. The GC- and EB-criteria are stopped to calculate after the threshold is reached for faster computation reasons.

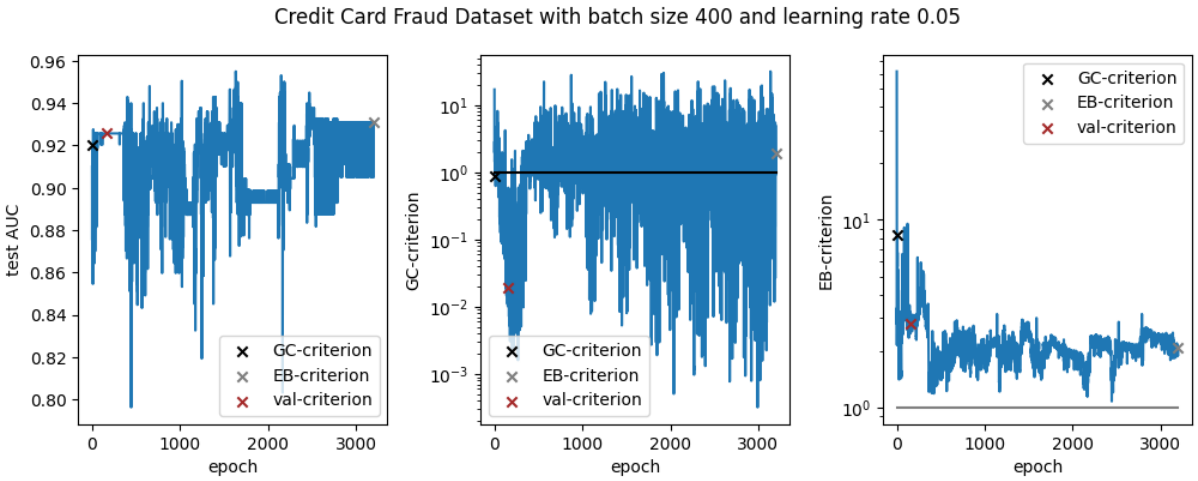

Figure 4: Training behaviour of the Credit Card Fraud Dataset with a batch size of 44 and a learning rate of 0.001. The black and the gray line indicate the threshold after which the training would stop according to the respective criterion.

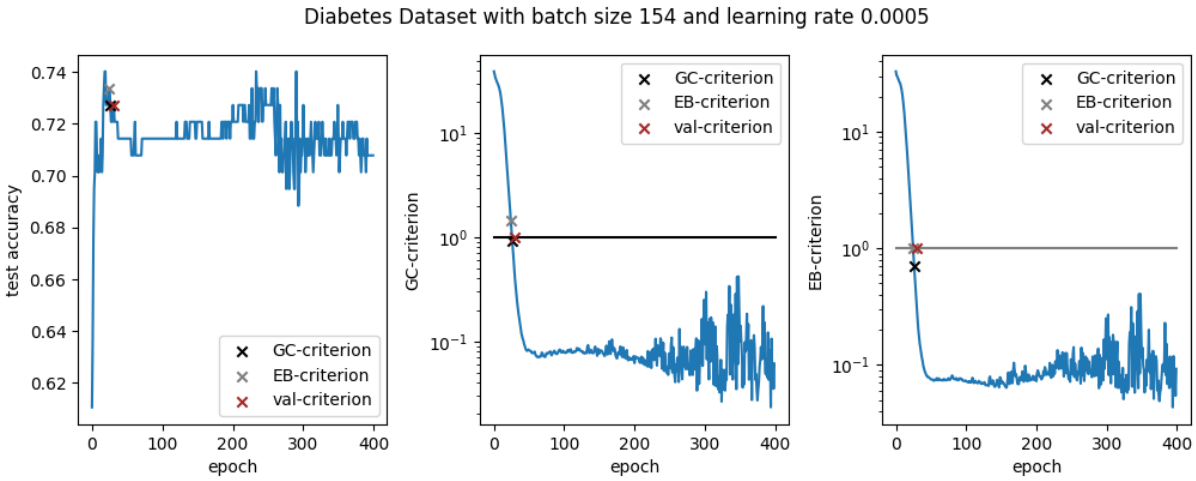

Figure 5: Training behaviour of the Diabetes Dataset with a batch size of 54 and a learning rate of 0.0005. The black and the gray line indicate the threshold after which the training would stop according to the respective criterion.

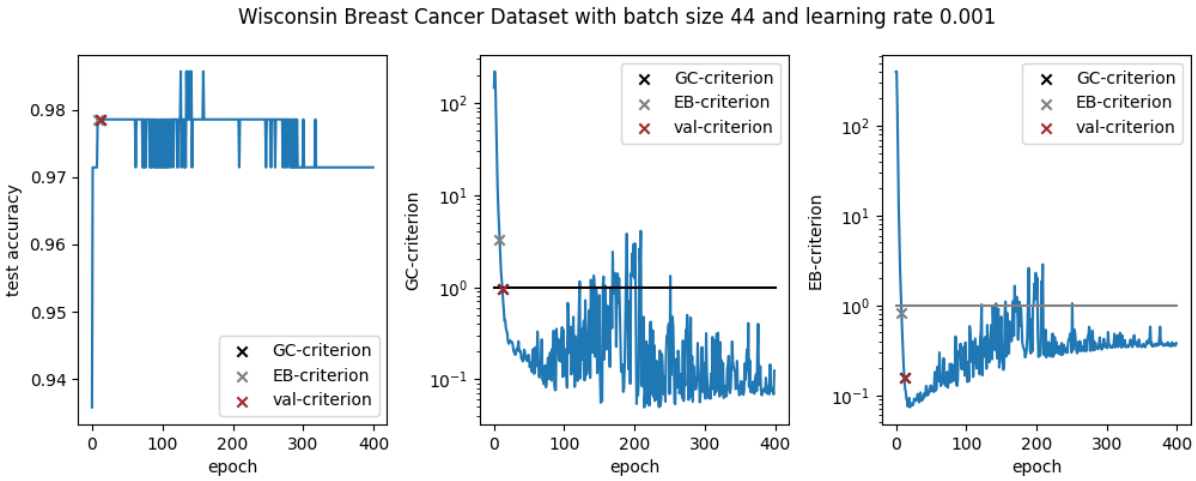

Figure 6: Training behaviour of the Wisconsin Breast Cancer Dataset with a batch size of 44 and a learning rate of 0.001. The black and the gray line indicate the threshold after which the training would stop according to the respective criterion.

the optimized hyperparameters are not pretty close. The best accuracies or AUC in the case of the Credit card fraud dataset with the previously achieved values are compared in the Fig. 1 and 2. Whereas for the Obesity and the Diabetes dataset lead to best accuracies greater than any of the obtained before the Credit card fraud dataset and the Titanic dataset yield some best AUC and accuracies which are lower than the best accuracy/AUC obtained before. As the ones calculated in the hyperparameter optimization are evaluated using cross-validation they are more reliable. The other values from the test are dependent on the choice of the test set and depend on the random choices of the batches. Two of the cases in which the optimized accuracies/AUC were lower than the maximum accuracies/AUC from the hyperparameter test:

- Credit card fraud with the EB-criterion, learning rate 0.05, batch size 50
- Titanic with the validation-based criterion, learning rate 0.001, batch size 12

The two cases were trained and evaluated again using cross-validation which leads to following results: AUC of 0.9295 for the Credit card fraud dataset and test accuracy of 0.7678 for the Titanic dataset. Both values are low compared to the values obtained in the optimization (table 4). Combining the TPE with cross-validation leads to more reliable results for two reasons: Cross-validation uses the entire dataset for training and validation and during the optimization algorithm hyperparameters are tested which have a high promisingness, slightly different hyperparameters also lead to good results.

The mean of the performances of the hyperparameter test of the four datasets used for optimizations with the validation-based criterion is 0.8230, with the GC-criterion 0.8160 and with the EB-criterion 0.8173. When taking the performances of the optimizations, the GC-criterion yielded the best value of 0.8607, the EB-criterion 0.8600 and the validation-based criterion 0.8583. Therefore, using customed hyperparameters can significantly increase the effectiveness of the GC and Eb-criterion.

Following test result was also obtained in another project for the Diabetes dataset. With k-nearest-neighbours, a classifier which classifies test datapoints based on the majority labels of the k nearest neighbours, a test accuracy of 0.7721 was obtained [13] which is lower than the three of the obtained results with the optimization (table 4). k=25 was used which was obtained by grid search.

In another project for the Credit card fraud dataset, a Random Forest Classifier was used and via 9-cross-validation an average AUC of 0.9277 was obtained [11] which is also lower than the obtained ones by TPE for all of the three stopping criteria (table 4). The Random Forest Classifier is a machine learning algorithm belonging to the ensemble-based methods category. It combines multiple decision trees, referred to as a "Random Forest," to make accurate predictions. Each tree is trained on a random subset of the training data and uses a random subset of available features to increase model diversity and robustness. During classification, the Random Forest aggregates the decisions of its individual trees to achieve reliable predictions.

In most of the cases, the accuray/AUC including some adapted Dropout probability and L-2-weight is slightly better than without. Unfortunately, the standard deviation of the performances from cross-validation are not calculated, that is why it is not clear how significant these differences are. In the Titanic dataset, the performance is even lower with some dopout probability and L-2-weight. Probably the optimizer do not find the better value with a dropout probability of 0 and a L-2-weight of 0. Regarding the hyperparameters, noticeble dependencies are only that the optimized dropout probability with

Table 4: The different optimized hyperparameters from the TPE and with the obtained accuracies.

Stopping criterion	**Values**	**Credit Card Fraud**	**Titanic**	**Obesity**	**Diabetes**
GC-criterion	Best accuracy/AUC	0.9397	0.8228	0.8910	0.7798
	Optimized learning rate	0.00045	0.01907	0.00751	0.00439
	Optimized batch size	51	158	345	198
EB-criterion	Best accuracy/AUC	0.9321	0.8238	0.9010	0.7813
	Optimized learning rate	0.00011	0.03796	0.00910	0.00027
	Optimized batch size	206	155	340	327
validation-based criterion	Best accuracy/AUC	0.9337	0.8294	0.9034	0.7762
	Optimized learning rate	0.00827	0.00010	0.00012	0.00012
	Optimized batch size	35	156	312	312

Table 5: The different optimized hyperparameters from the TPE and with the obtained accuracies.

Criterion	**Values**	**Credit Card Fraud**	**Titanic**	**Obesity**	**Diabetes**
GC-criterion	Best accuracy/AUC	0.9413	0.8126	0.8996	0.7761
	Optimized L-2 weight	0.008507	0.000116	0.001761	0.001012
	Optimized dropout probability	0.03165	0.45679	0.48874	0.45571
EB-criterion	Best accuracy/AUC	0.9402	0.8182	0.9011	0.7826
	Optimized L-2 weight	0.000046	0.000176	0.000364	0.000062
	Optimized dropout probability	0.4218	0.0152	0.4996	0.0797
Val criterion	Best accuracy/AUC	0.9322	0.8316	0.9057	0.7826
	Optimized weight	0.000183	0.000571	0.000707	0.006173
	Optimized dropout probability	0.3421	0.2266	0.117635	0.4996

the GC-criterion is close to 45-49 % except the probability for the Credit card fraud dataset. However, this dropout probability lowered probably the performance on the Titanic dataset (table 5).

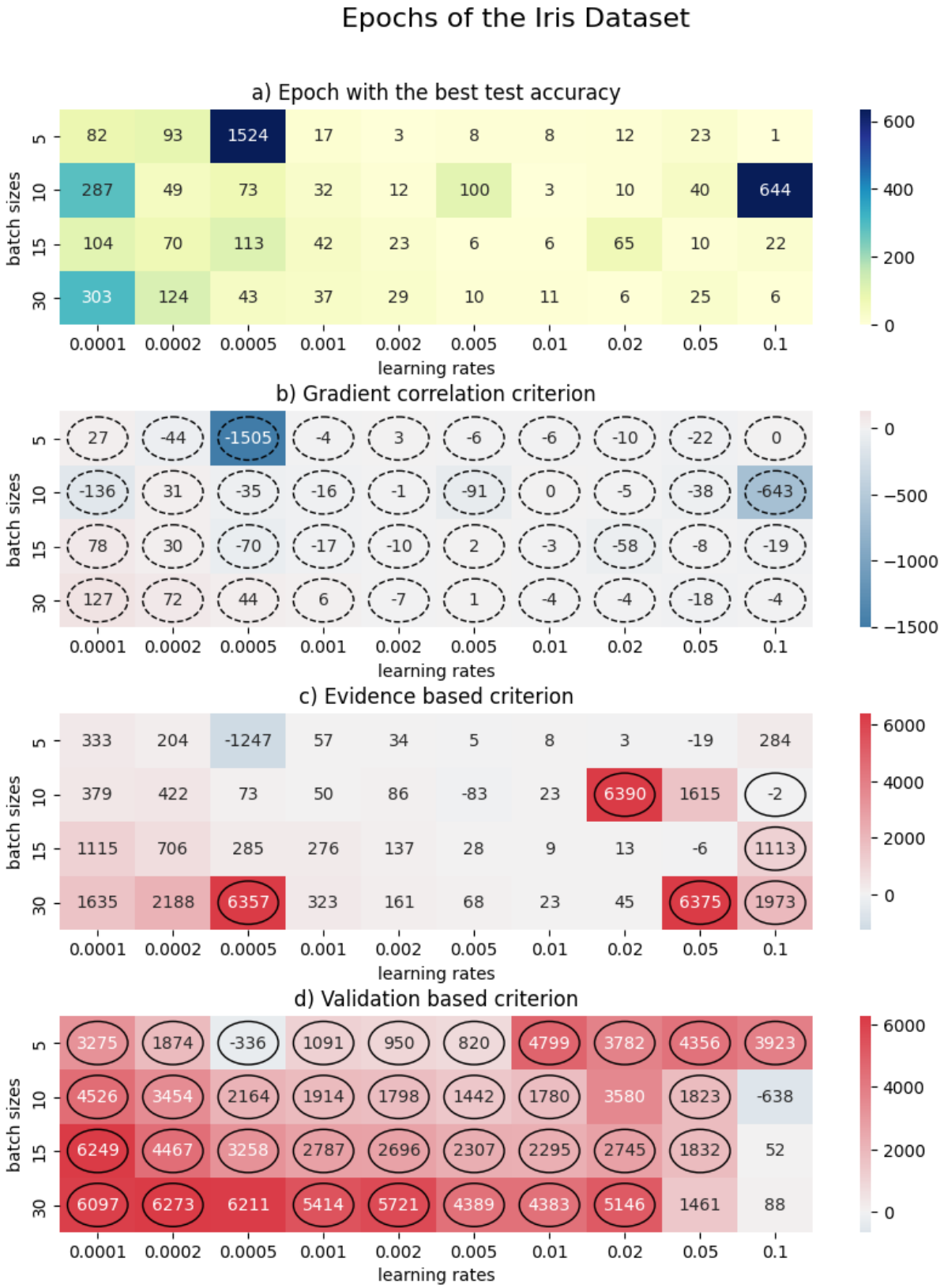

Figure 7: From the training on the Iris Dataset: a) Epochs after which the best test accuracy is reached. b) - d) Differences between the epochs where the training would have stopped according to the criterion and the index with the maximal achievable accuracy. Solid black encirclement indicate that the training would have as the first according to the criterion compared to the others, dashed black encirclements the last, solid gray encirclements one of the first and dashed gray one of the last.

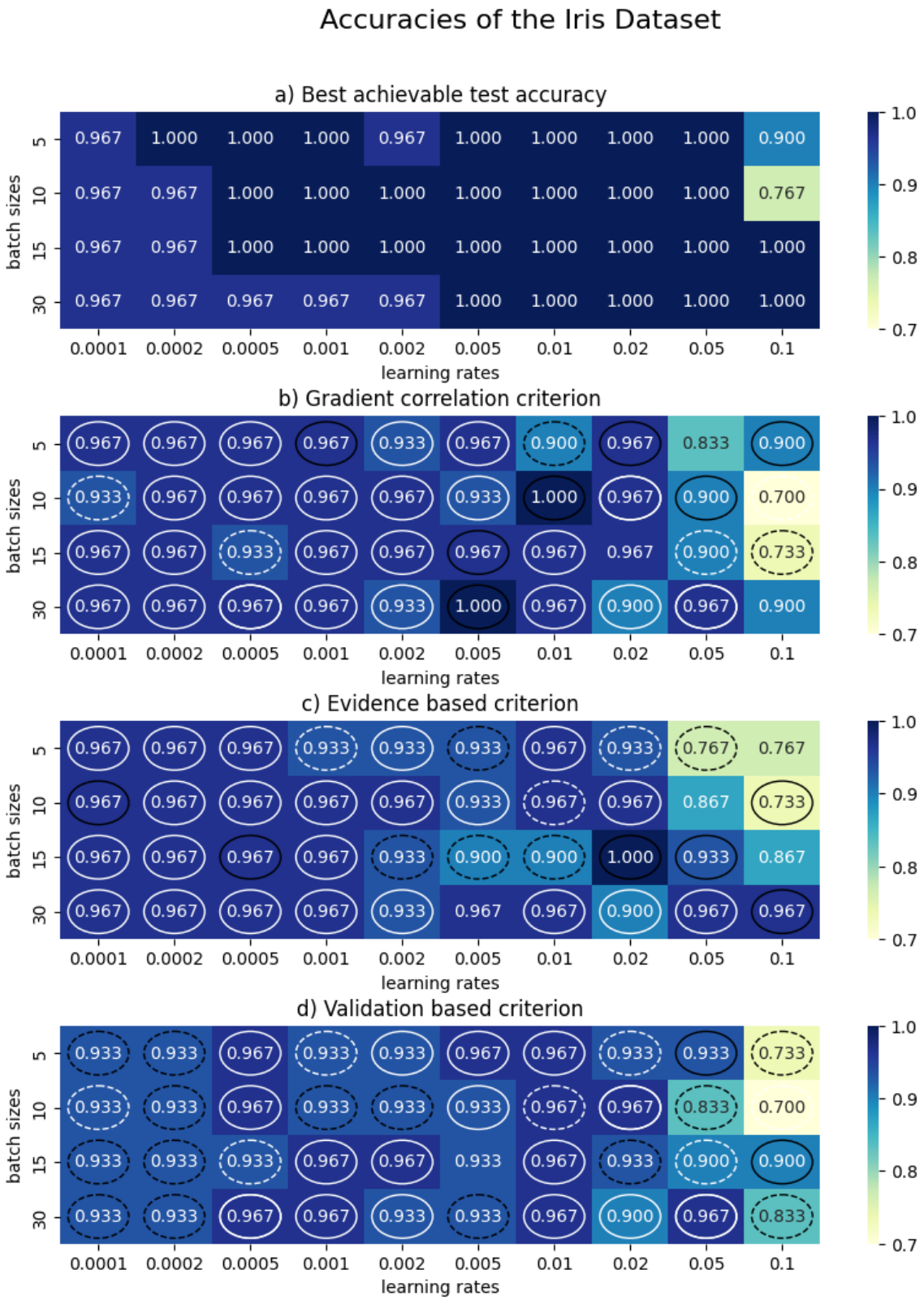

Figure 8: From the training on the Iris Dataset: a) Best achievable accuracies. b)-d) Achieved accuracies according to the respective criterion. Solid black encirclements indicate that the training would have led to the highest accuracy according to the criterion compared to the other 2, the lowest accuracy, solid white encirclements one of the highest und dashed white encirclements one of the lowest.

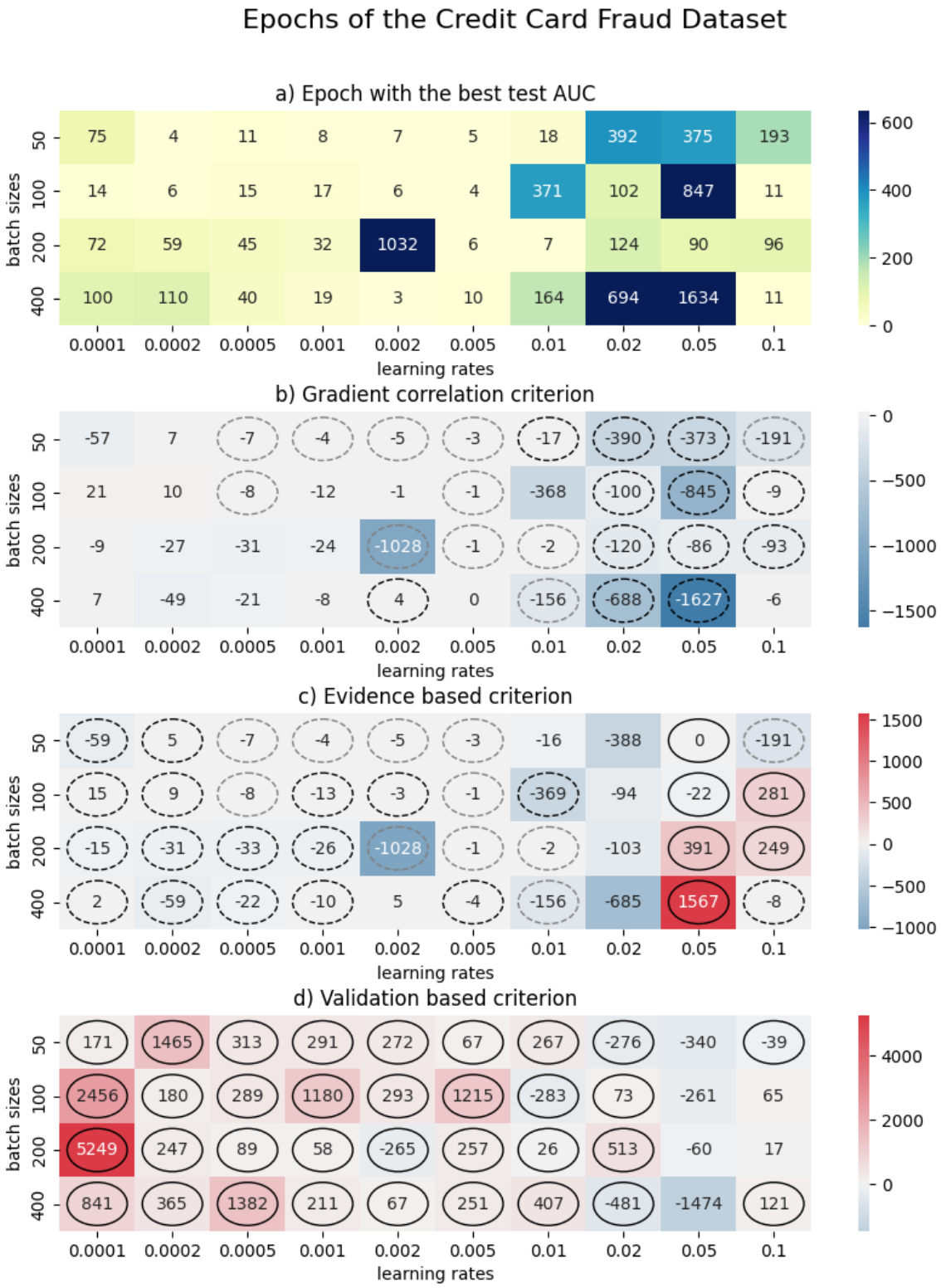

Figure 9: From the training on the Credit Card Fraud Dataset: a) Epoch after which the best test accuracy is reached. b) - d) Difference between the epochs where the training would have stopped according to the criterion and the index with the maximal achievable AUC. Solid black encirclement indicate that the training would have as the first according to the criterion compared to the others, dashed black encirclements the last, solid gray encirclements one of the first and dashed gray one of the last.

Figure 10: From the training on the Credit Card Fraud Dataset: a) Best achievable accuracies. b)-d) Achieved accuracies according to the respective criterion. Solid black encirclements indicate that the training would have led to the best accuracy according to the criterion compared to the other 2, the lowest accuracy, solid white encirclements one of the highest and dashed white encirclements one of the lowest. The crosses indicate between which values the optimal parameters are located.

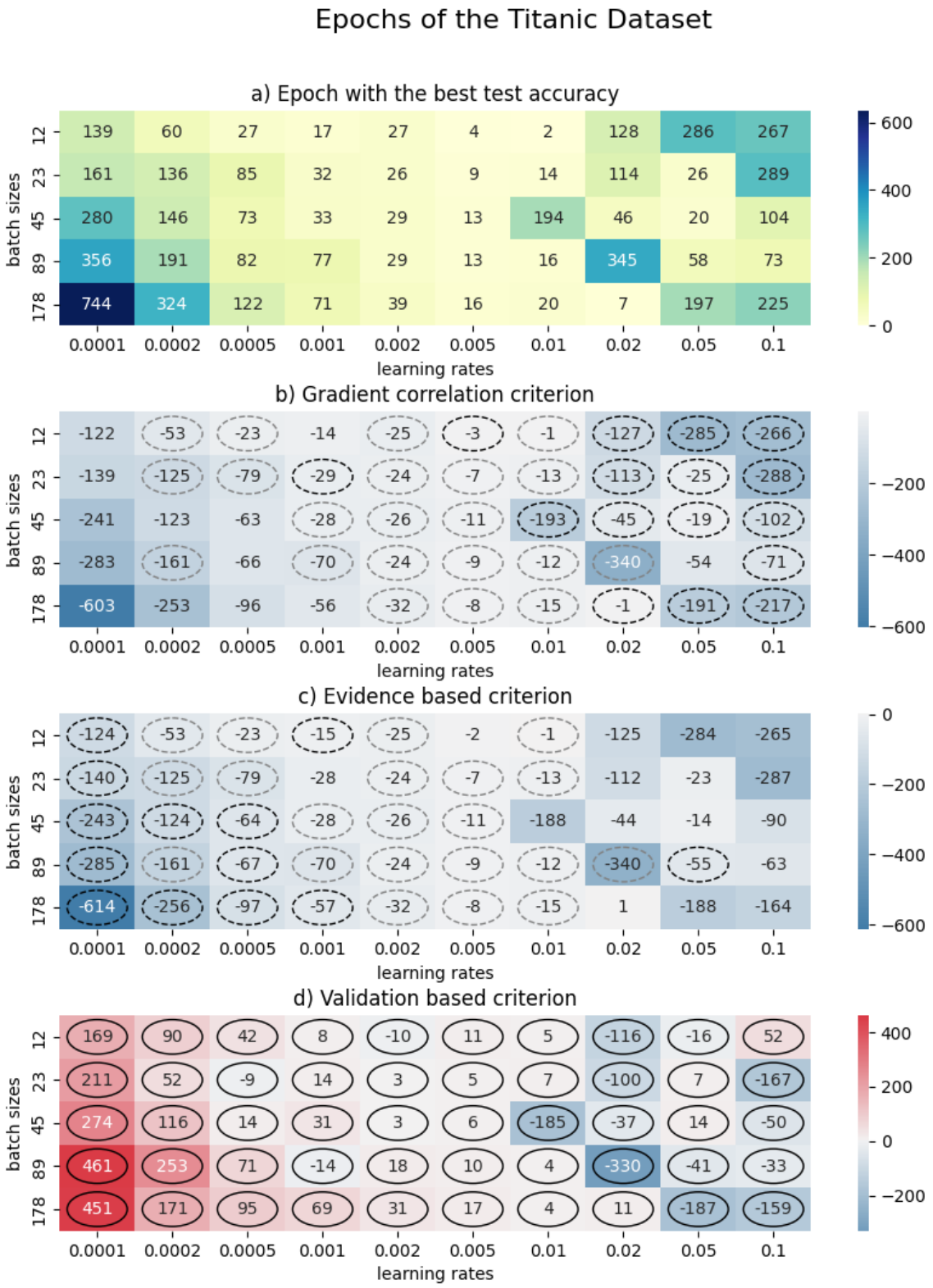

Figure 11: From the training on the Titanic Dataset: a) Epoch after which the best test accuracy is reached. b) - d) Difference between the epochs where the training would have stopped according to the criterion and the index with the maximal achievable accuracy. Solid black encirclement indicate that the training would have as the first according to the criterion compared to the others, dashed black encirclements the last, solid gray encirclements one of the first and dashed gray one of the last.

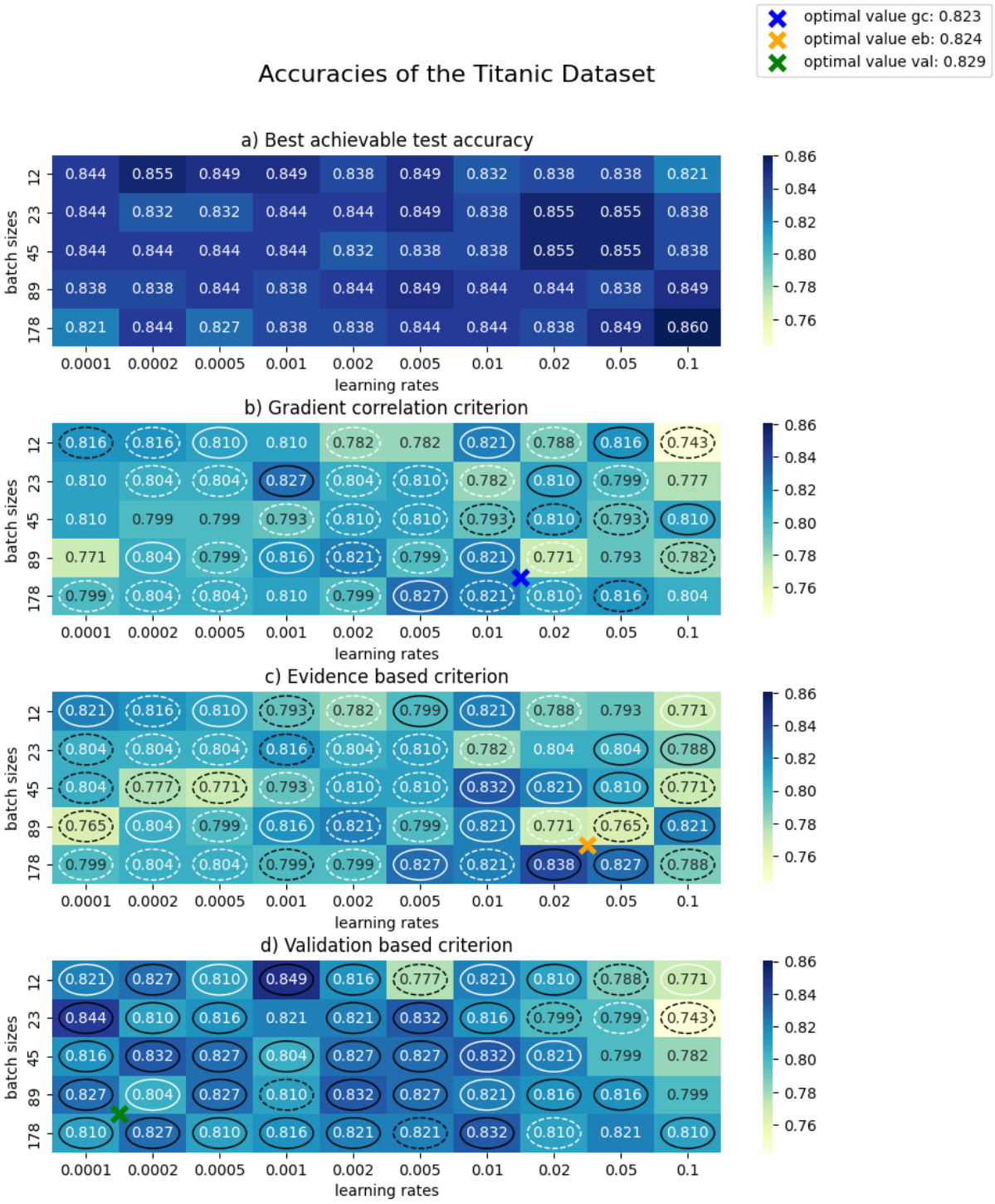

Figure 12: From the training on the Titanic Dataset: a) Best achievable accuracies. b)-d) Achieved accuracies according to the respective criterion. Solid black encirclements indicate that the training would have led to the best accuracy according to the criterion compared to the other 2, the lowest accuracy, solid white encirclements one of the highest and dashed white encirclements one of the lowest. The crosses indicate between which values the optimal parameters are located.

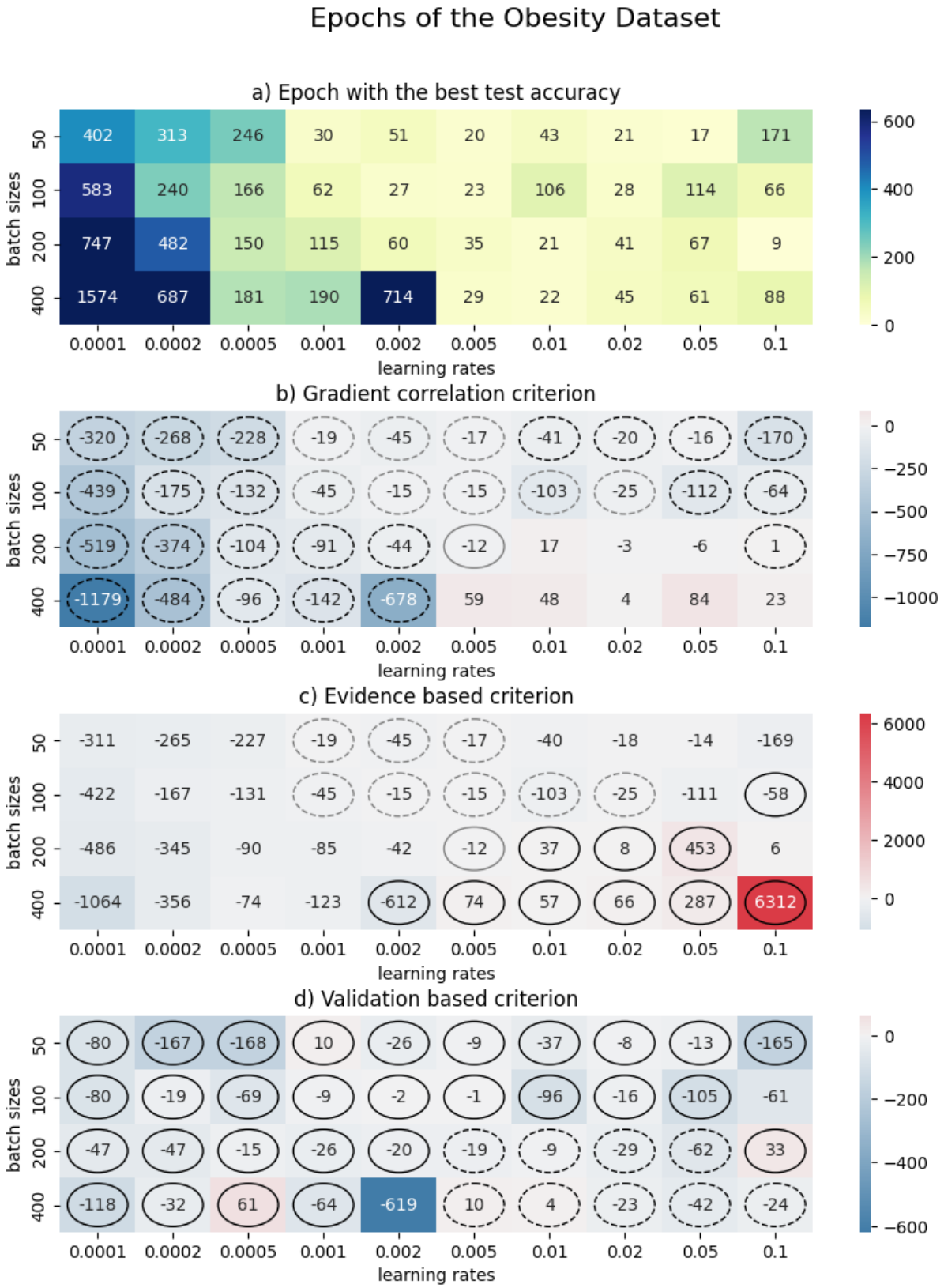

Figure 13: From the training on the Obesity Dataset: a) Epoch after which the best test accuracy is reached. b) - d) Difference between the epochs where the training would have stopped according to the criterion and the index with the maximal achievable accuracy. Solid black encirclement indicate that the training would have as the first according to the criterion compared to the others, dashed black encirclements the last, solid gray encirclements one of the first and dashed gray one of the last.

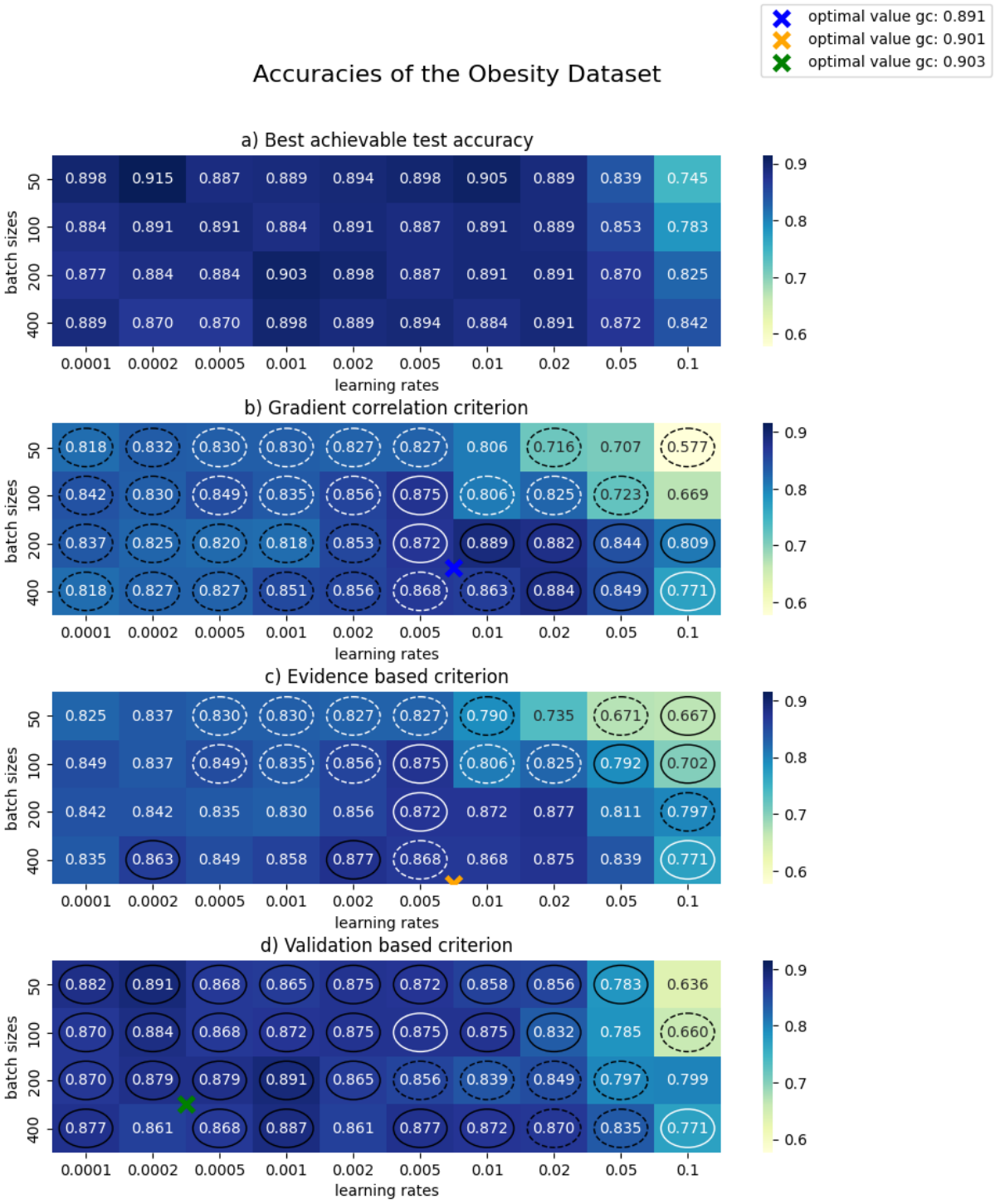

Figure 14: From the training on the Obesity Dataset: a) Best achievable accuracies. b)-d) Achieved accuracies according to the respective criterion. Solid black encirclements indicate that the training would have led to the best accuracy according to the criterion compared to the other 2, the lowest accuracy, solid white encirclements one of the highest and dashed white encirclements one of the lowest. The crosses indicate between which values the optimal parameters are located.

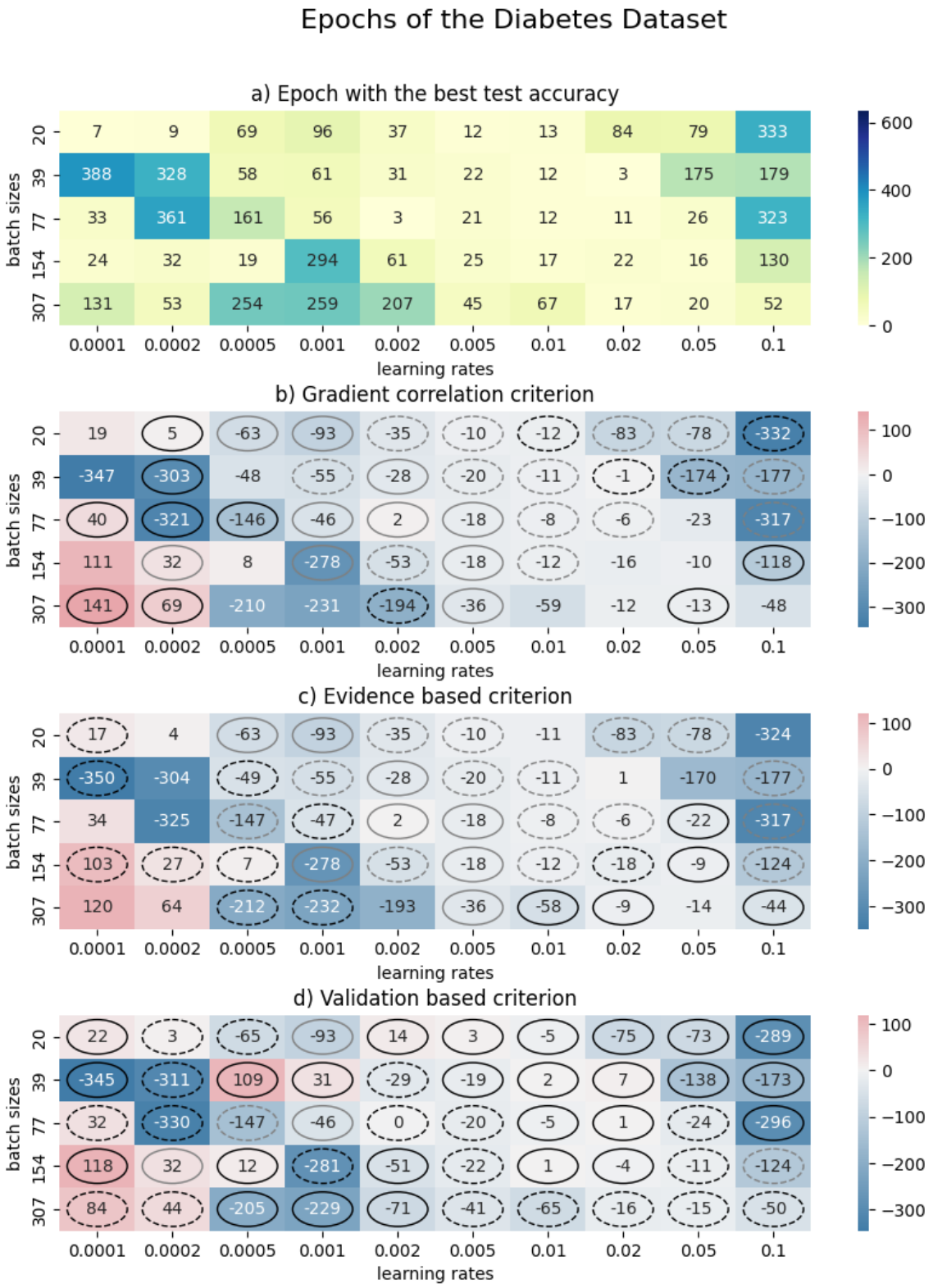

Figure 15: From the training on the Diabetes Dataset: a) Epoch after which the best test accuracy is reached. b) - d) Difference between the epochs where the training would have stopped according to the criterion and the index with the maximal achievable accuracy. Solid black encirclement indicate that the training would have as the first according to the criterion compared to the others, dashed black encirclements the last, solid gray encirclements one of the first and dashed gray one of the last.

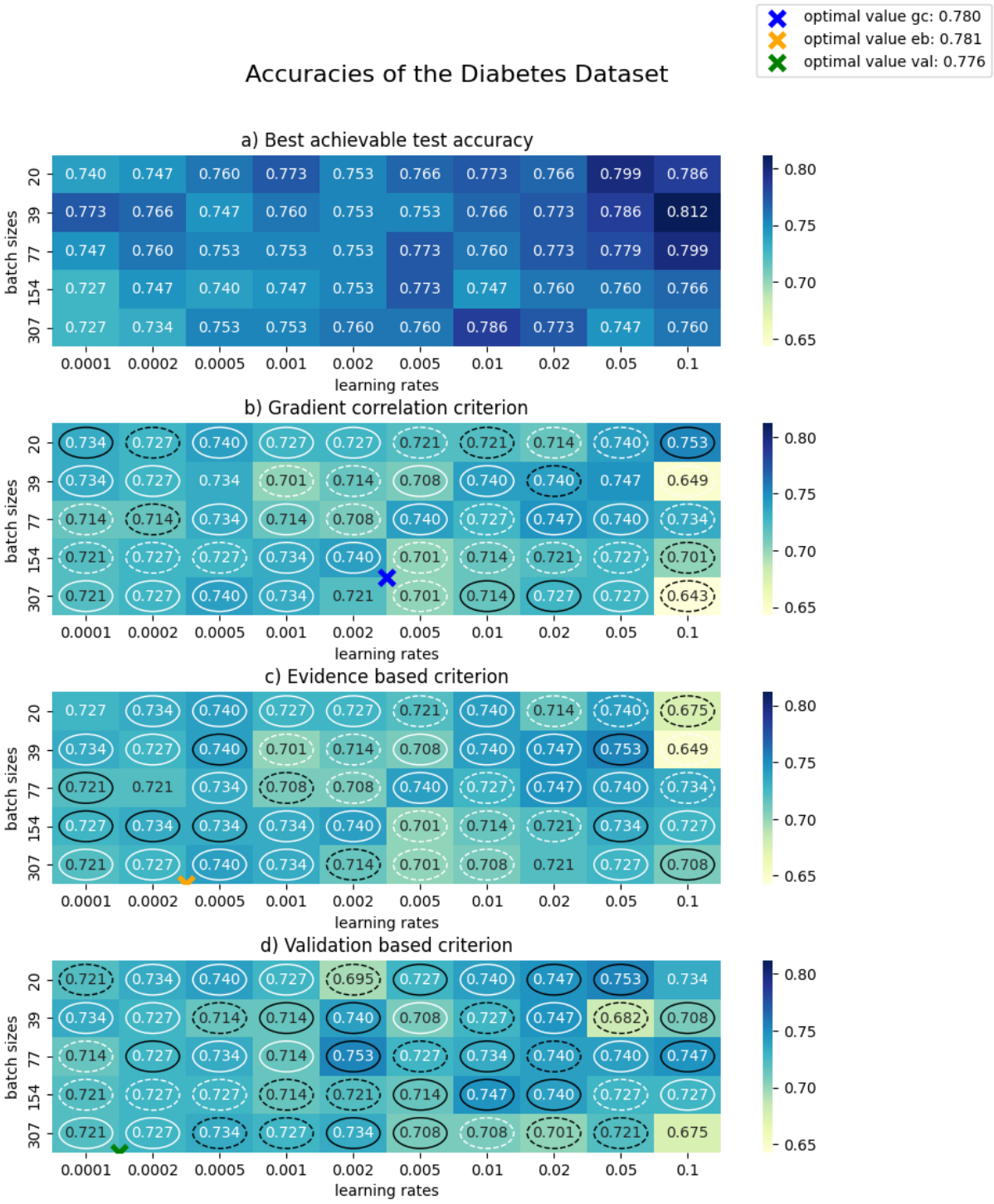

Figure 16: From the training on the Diabetes Dataset: a) Best achievable accuracies. b)-d) Achieved accuracies according to the respective criterion. Solid black encirclements indicate that the training would have led to the best accuracy according to the criterion compared to the other 2, the lowest accuracy, solid white encirclements one of the highest and dashed white encirclements one of the lowest. The crosses indicate between which values the optimal parameters are located.

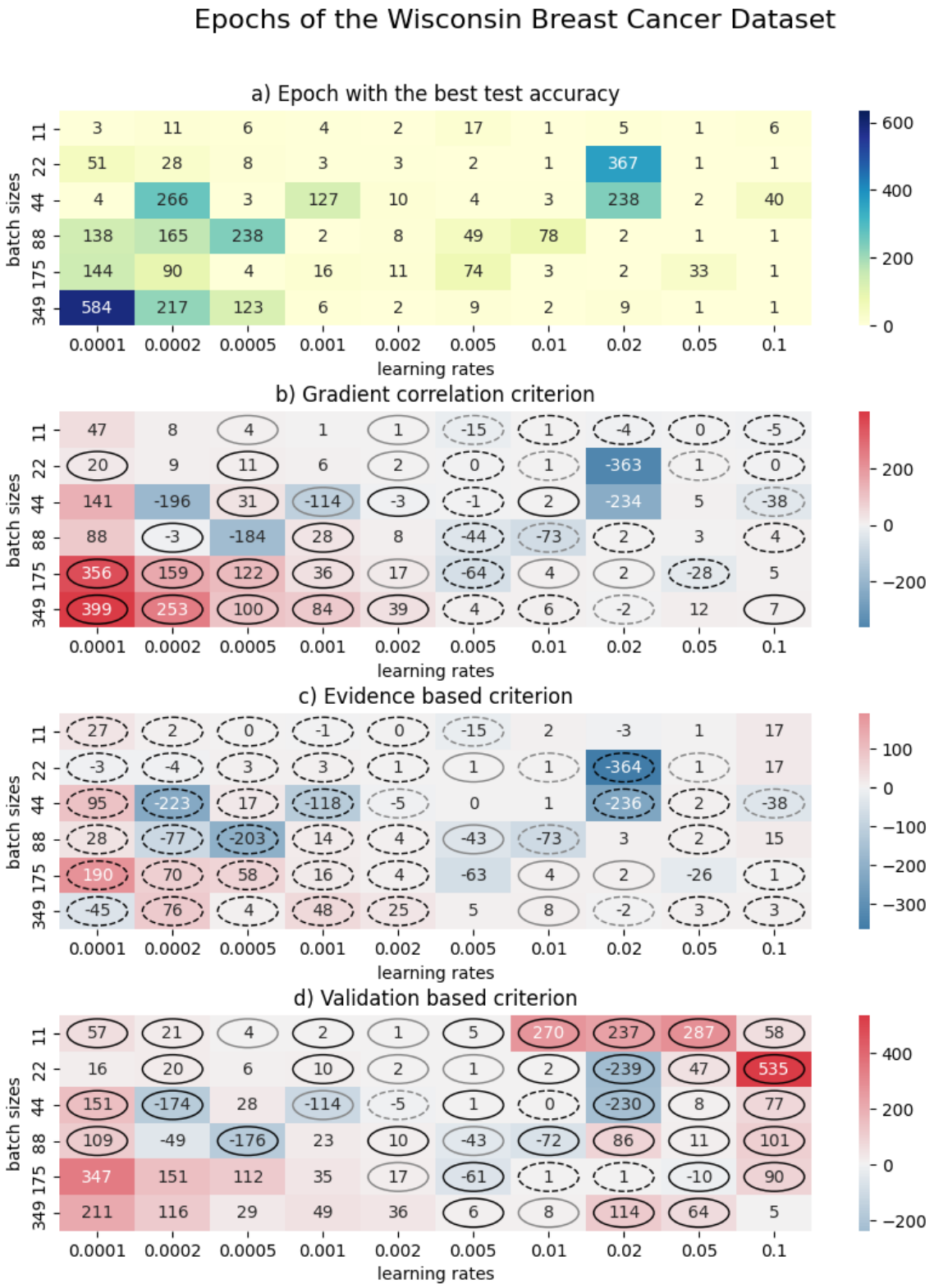

Figure 17: From the training on the Wisconsin Breast Cancer Dataset: a) Epoch after which the best test accuracy is reached. b) - d) Difference between the epochs where the training would have stopped according to the criterion and the index with the maximal achievable accuracy. Solid black encirclement indicate that the training would have as the first according to the criterion compared to the others, dashed black encirclements the last, solid gray encirclements one of the first and dashed gray one of the last.

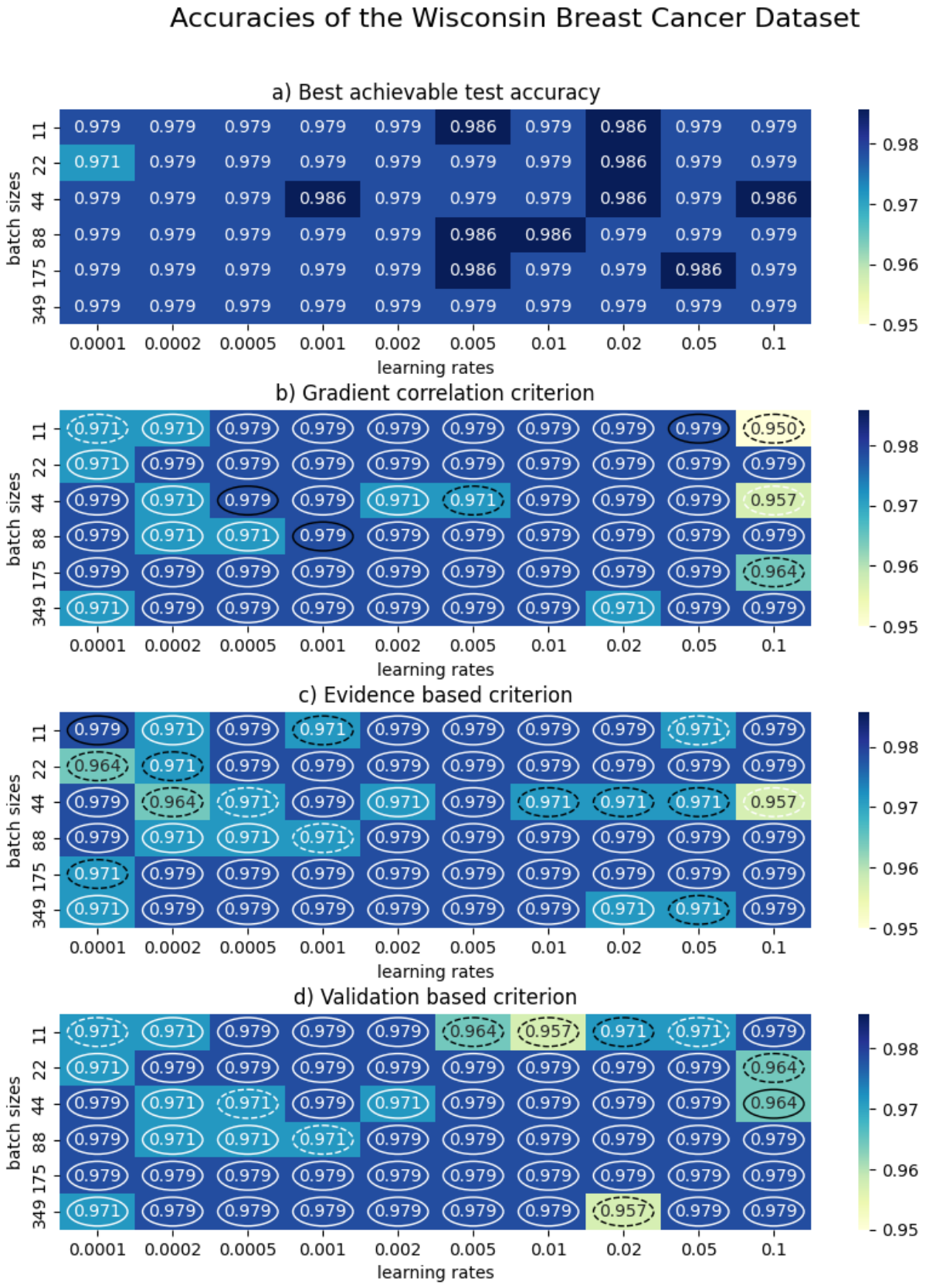

Figure 18: From the training on the Breast Cancer Dataset: a) Best achievable accuracies. b)-d) Achieved accuracies according to the respective criterion. Solid black encirclements indicate that the training would have led to the best accuracy according to the criterion compared to the other 2, the lowest accuracy, solid white encirclements one of the highest and dashed white encirclements one of the lowest.

7 Discussion

In this thesis, the performances of novel stopping criteria are compared to a stopping criterion according to a performance on a validation set. The robustness of the hyperparameters batch size and learning rates are tested, and the training is performed until a specific epoch. T gradient-based criteria are directly calculated together with the test loss, in a separate train loop the validation stopping point is determined.

With a few exceptions, the algorithm would have stopped according to the gradients-based criteria before it would have stopped according to the validation-based criteria. Based on the maximum achievable values, the gradient-based criteria stop early enough or too early whereas the validation-based criterion tends to slightly overfit. For the specific chosen hyperparameters and hyperparameter ranges, the average of performance of the validation-based criterion is with 0.8656 slightly higher than the one of the EB- and GC- criteria with both approximately 0.8631. For all datasets, except the Diabetes dataset, some dependencies on the batch size and learning rate can be seen and statements about the preference of stopping criteria can be made. Higher batch sizes and lower learning rates lead to more training epochs and higher learning rates to more fluctuations. However, the diabetes dataset, which is the one with the lowest performance in general, has more fluctuations in both the stopping moments and performances. For the Credit card dataset, both the EB- and GC-criteria would have led to better performances whereas using the validation would have been better for the Titanic and Obesity dataset. For the training of the Credit card fraud dataset, an upsampling of the training data was used. The training set is equally duplicated twice. Both the EB- and the GC- criterion cross the threshold of 1 at the same point even though the training set is duplicated. On the other hand, the randomly chosen validation set can have more imbalances now because specific datapoint can appear different times. That is probably the reason why the validation-based criterion performs worse on the Credit card fraud dataset. It would be interesting to see, how the performance would change if the validation set is chosen before the duplication. Taking into account the issue with the Credit card dataset the gradient-based criteria performs worse than the validation-based criterion. Because the stopping epoch for the validation-based criterion is not determined in the main training process it may differ the optimum epoch differs from the epoch a held-out validation set in the main process would have as it is common according to [19]. The difference would be that the validation stopping criterion stops more precisely according to the specific training process while the overall would be lower. It would be also interesting to gain more insights about both of the two different stopping algorithms.

In the second part, the hyperparameters batch size and learning rate are optimized with the Tree Parzen Estimator. While in the Obesity and in the Diabetes dataset the optimized performances exceed most of the maximum achievable performances in the hyperparameter test they are only partially reached in the Titanic and the Credit card dataset. Maybe they are not stable enough and just stem from peaks in fluctuations during the training process. In both of the training sets the optimized performances did not reach the very best performances from the hyperparameter test and when checking the very best performances from the hyperparameter test with cross-validation it turned out that they are not stable. It would also be interested to take the standard deviations of the cross-validations in order to get more information about the fluctuations of the specific datasets. Comparing the overall performances of the optimizations, the GC- criterion leads to a an average accuracy/AUC of 0.8607, the EB-criterion an average accuracy/AUC of 0.8600 and the validation-based criterion an average accuracy/AUC of 0.8583. The issue with the upsampling with the Credit card dataset also has to be taken into account. The influence

of adding L-2 and dropout regularization depends on the dataset. There is no clear structure, too many different factors influence the performance at the same time. Longer optimization processes with more hyperparameters optimized at the same time would gain more information about the usefulness. Also L-1 regularization could be added. Since this led to node deaths (weights become zero) the calculation of the EB-criterion would have to be adapted to avoid dividing by zero.

[17] also proposes to calculate the EB-criterion for each single layer and let stop training the weights after the particular EB-criterion reaches the threshold of 1. The performance of such an algorithm could also be studied as it might lead to more adaptive results. The backpropagation algorithm would not be changed, only some of the weight updates would not be conducted if the layer has already stopped.

The different datasets bring about individual results. There was the hope that different stopping criteria would be preferred according to the specifications of the input. Analyzing by taking fewer dimensions or adding noise would give more insight into that. But the data itself influences the performances of the hyperparameters including the stopping criteria. At least, this thesis shows that the EB- and GC-criteria can be applicated to different classification problems. If there is not enough computational time available to optimize hyperparameters both of them can just be chosen. According to [17] the characteristics of the EB-criterion are fast to compute and each epoch takes around 1.25 times longer. With the library Pytorch with python on CPU, it takes longer.

8 Summary

First, the hyperparameter test is conducted, robustness of different stopping criteria with changing batch size and learning rate are compared and also their stopping behaviour. Both the novel criteria EB-criterion and GC-criterion stop earlier than the validation-based method in most of the cases. Both of them tend to stop before the maximum accuracy could be reached whereas the validation-based method stops later in the majority of the cases. The three criteria bring about similar accuracies/AUC on average, also after conducting a hyperparameter optimization.

References

[1] Google colaboratory jupyter notebook service. https://colab.google/.

[2] Matplotlib visualization in python. https://matplotlib.org/.

[3] Numpy scientific computing package. https://numpy.org/.

[4] Optuna hyperparameter optimization framework. https://optuna.org/.

[5] Pandas data analysis library. https://pandas.pydata.org/.

[6] Pima indians diabetes database on uci irving machine learning repository. https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

[7] Python programming language. https://www.python.org/.

[8] Pytorch for deep learning. https://pytorch.org/.

[9] Seaborn statistical data visualization. https://seaborn.pydata.org/.

[10] Titanic dataset on kaggle. https://www.kaggle.com/datasets/brendan45774/test-file.

[11] Cv'd tuning with credit card fraud data. https://www.kaggle.com/code/joshred/cv-d-tuning-with-credit-card-fraud-data, 2019.

[12] Estimation of obesity levels based on eating habits and physical condition on uci irving machine learning repository. https://archive.ics.uci.edu/dataset/544/es timation+of+obesity+levels+based+on+eating+habits+and+physical+condition, 2019. DOI: https://doi.org/10.24432/C5H31Z.

[13] Step by step diabetes classification-knn-detailed. https://www.kaggle.com/code/shrutimechlearn/step-by-step-diabetes-classification-knn-detailed, 2020.

[14] Fisher, R. A. Iris dataset on uci irving machine learning repository. https://archive.ics.uci.edu/dataset/53/iris.

[15] James Bergstra, Rémi Bardenet, Y. B. B. K. Algorithms for hyper-parameter optimization. https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf, 2011.

[16] Lizotte, D. Dynamic stopping for artificial neural networks. https://lup.lub.lu.se/student-papers/search/publication/9078057, 2022.

[17] Maren Mahsereci, Lukas Balles, C. L. P. h. Early stopping without a validation set. https://arxiv.org/abs/1703.09580: :text=Earlygradient

[18] Mattias Ohlsson, P. E. Introduction to artificial neural networks and deep learning, 2020.

[19] Prechelt, L. Early stopping - but when? https://page.mi.fu-berlin.de/prechelt/Biblio/stop$_t$$ricks$1997.$pdf$, 2011.

[20] Schmidhuber, J. The road to modern ai: artificial neural networks up to 1979—from shallow learning to deep learning. https://people.idsia.ch/ juergen/deep-learning-history.html, 2022.

[21] Ting Hu, Y. L. Early stopping for iterative regularization with general loss functions. https://www.jmlr.org/papers/volume23/21-0983/21-0983.pdf, 2022.

[22] William Wolberg, Olvi Mangasarian, N. S. W. S. Breast cancer wisconsin (diagnostic) on uci irving machine learning repository. https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic, 1995.