Universiteit Leiden
The Netherlands

Opleiding Informatica

Impact of Hyperparameters on Model Performance in Split Learning

Amr Adwan

Supervisors:
Dr. J.N. van Rijn & Dr. O. Gadyatskaya & Dr. A.N. van der Meulen

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

07/12/2023

Abstract

This thesis investigates how hyperparameters influence model performance in Split Learning, a decentralized machine learning method. The study conducts thorough testing on numerous datasets to assess the effects of various hyperparameter configurations. This study points out which hyperparameters are significant for model accuracy, and to which degree. Graphical representations provide additional insight on these complex interactions, underlining the significance of fine-tuning hyperparameters in Split Learning. Finally, given this privacy-focused learning framework, our study specifically pinpoints the importance of optimizing the hyperparameters: cut layer, batch size, partition alpha, and total number of clients. Experiments performed over 11 datasets revealed that, in most cases, the partition alpha hyperparameter emerged as the most significant. This offers actionable insights for effective Split Learning model tuning.

Acknowledgements

I am grateful for Amin's contributions to this study on Federated and Split Learning. His guidance and support have been invaluable. Thank you.

Contents

1 Introduction

The amount of data produced in the twenty-first century has substantially increased, altering how we use and analyze information. The Internet enables quick data transfer throughout the world because to its variety of data formats, including text, graphics, and audio. The sharing of information has facilitated developments in a number of fields, including telemedicine, which allows for real-time patient consultations regardless of distance; e-commerce, which has revolutionized online shopping; and online education, which has opened up high-quality learning resources to people all over the world. Despite the potential research benefits, sharing private information like medical records raises privacy issues. Strict data protection laws, such as the General Data Protection Regulation (GDPR), must be followed when sharing this type of data [1].

Machine learning algorithms have evolved in response to growing worries about data privacy. This undergraduate research explores the split learning method see section 2.1.2, which divides a neural network model into two or more parts. In 2016, Google researchers proposed federated learning, which enables numerous clients, such as mobile devices or organizations, to cooperatively train a model under the direction of a central server without sharing the raw data [2]. Split Learning, on the other hand, enhances data privacy by dividing the model training process across several entities [3].

However, knowledge about how variables like hyperparameters affect the effectiveness of Split Learning models is still limited. Hyperparameters are predefined settings for machine learning models that are selected before training begins. In the area of neural networks, for example, hyperparameters may control things such as the learning rate, which determines how rapidly the model responds to training data, or the number of hidden layers, which can influence the model's complexity and ability for recognizing patterns. Choosing the best hyperparameter settings is critical since they directly affect the training process's efficiency and accuracy. It's critical to distinguish between globally controllable hyperparameters like "batch size" and those impacted by external limitations like "cut layer," which are frequently motivated by privacy concerns. Finding the appropriate hyperparameter settings is a major emphasis in machine learning research because of its influence on model performance.

This study aims to investigate the impact of hyperparameters on Split Learning model performance across multiple datasets. We employ Functional Analysis of Variance (fANOVA) [4] for this. The fANOVA technique evaluates the impact of individual hyperparameters or their combinations on the variation in model output. It explains how adjusting a hyperparameter might change the performance that can be expected from a model.

This research lies at the crossroads of privacy-focused machine learning approaches. Utilizing Python and fANOVA, it investigates the effects of different hyperparameters accross 11 datasets on the performance of Split Learning models. Although there are many theoretical insights about how hyperparameters affect privacy- centric machine learning, there is less discussion of the actual effects in various real- world scenarios. Our goal is to identify which hyperparameters have a significant impact on the model and which ones are still unimportant in order to give guidance on the practical consequences of changing the hyperparameters.

2 Related Work

2.1 Federated & Split Learning

2.1.1 Federated Learning

Federated Learning has risen as an imperative technique to cater to the escalating emphasis on privacy in machine learning. Instead of centralizing data, it empowers distinct entities, such as hospitals or smartphones, to collaboratively enhance a shared model without disclosing their raw data. This paradigm serves as a bridge between realms of privacy and collaborative learning. Take hospitals as a case in point: they can collectively refine a predictive model for patient diagnoses without revealing individual patient data, thereby striking a balance between improved patient care and stringent data privacy. Such a framework not only champions privacy but also mitigates risks associated with data breaches [5]. Its versatile applicability spans sectors like healthcare [6], Internet of Things (IoT) [7], and advanced natural language processing tasks [8].

A tangible manifestation of Federated Learning is its integration by Google for enhancing mobile keyboard predictions. This is how it operates: during idle hours when a user's phone is connected and charging, Google harnesses the local computational prowess to refine its model. Instead of transmitting raw data to central servers, it merely updates the model, thereby ensuring user data remains protected [9].

Delving deeper into the mechanics of Federated Learning, it facilitates decentralized machine learning by leveraging data across multiple local devices without sensitive data transfer [2]. The workflow can be broken down as follows:

1. An initial global model is synthesized using a randomly selected subset of global data.
2. This model is dispatched to all participating local entities.
3. Each entity, equipped with its unique data, fine-tunes this global model.
4. Post local training, models are sent back to a central server, culminating in an aggregated, enhanced global model.

The cyclical nature of this process persists until desired model efficacy is attained. The pivotal advantage? All raw data remains sequestered in its original location, fortifying data privacy. For a graphical representation of Federated Learning, one can refer to the figure 1.

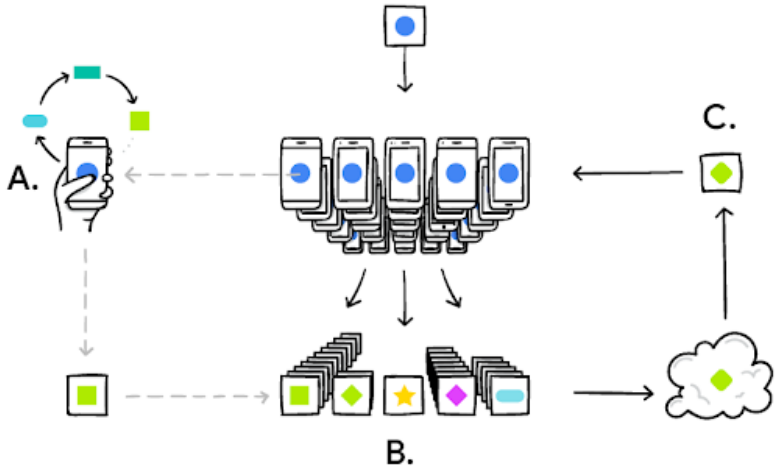


Figure 1: Sequential illustration of Federated Learning: local device personalizes the model (A), numerous user updates are compiled (B), and a consensus model update is achieved (C), initiating the next cycle. [9]

2.1.2 Split Learning

Split Learning, on the other hand, employs an enhancing yet different technique as compared to existing privacy-preserving machine learning methods. The neural network model is split between entities in Split Learning, guaranteeing that only specific layers of the model are developed at each client, while the remainder is learned at a central server.
The general process for Split Learning is as follows:

1. **Model Splitting:** Deep learning models are divided into parts. One part is stored on the client side, while the other is retained on the central server. The split is usually done so that the first layers, which extract features from raw data, are retained on the client side, while the subsequent layers, which generate predictions based on those characteristics, are kept on the server.

2. **Local Forward Propagation:** Each client executes a forward pass on its portion of the model using its local data. This yields intermediate representations or features.

3. **Feature Transmission:** Instead of sending raw data, clients send these intermediate features to the central server.

4. **Server-side Forward and Backward Propagation:** The central server completes the rest of the forward pass using its segment of the model and then calculates the loss. It then performs a backward pass to calculate the gradients.

5. **Update Transmission:** Only necessary gradients or updates are transmitted back to the clients.

6. **Local Backward Propagation and Update:** Each client then uses these updates to adjust its model segment, executing the backward pass and updating its weights.

This configuration keeps raw data on the client side, preserving data privacy. Only the intermediate features are supplied, which are less informative than the raw data. As a consequence, while the central server never sees the raw data, it can still direct the model's training [10].

2.1.2.1 Vertical Split Learning

In the context of Split Learning, the configuration for vertically partitioned data presents a unique approach where multiple clients collaborate, each possessing a distinct set of features for the same data samples.

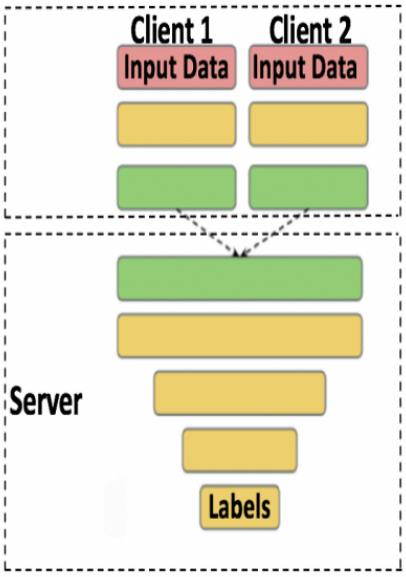


Figure 2: Split learning configurations showing the approach for vertically partitioned data. Each layer represents a layer in a neural network [10]

.

As depicted in Figure 2:

- Client 1 and Client 2 have distinct parts of the input data. This segmentation is not based on discrete samples, but rather on various types or attributes of the same data.
- Each client processes its own data subset through localized neural network layers, creating intermediate features.
- These intermediate features are then sent to the server. The server, which is provided with the subsequent layers of the neural network, analyses these combined features to provide final predictions or outcomes.

2.1.3 Moradi's Benchmark in Federated Learning

Moradi investigated privacy-focused machine learning approaches, with an emphasis on Federated Learning and Split Learning [11]. He created a benchmarking application utilizing PyTorch and PySyft. This application focuses flexibility to different datasets, modular training and testing methodologies, and GPU processing capability.

2.1.3.1 Benchmarking and Hyperparameter Importance

Moradi's benchmarking tool uses a special dataset class that supports picture, text, and tabular datasets; a subset of the OpenML CC18. This enables clients to employ individual network designs or collaborate on a shared model. The OpenML python library [12] is integrated into the program for quick data loading. Moradi [11] proposed a way to mitigate potential biases in data distribution in federated learning. Using a Dirichlet distribution technique, the system experiments with different data distributions across clients.

2.1.3.2 Functional ANOVA and Hyperparameter Analysis

Moradi also utilized the Functional ANOVA framework see section 2.2.1 in order to assess hyperparameter contributions. This approach examines how hyperparameters and their relationships affect outcomes. It assesses hyperparameter performance across continuous and discrete variables using tree-based surrogate models, assisting algorithm designers in setup and performance prediction [11].

2.1.3.3 Comparison with Current Research

Building on Moradi's [11] findings, this research aims to refine and enhance the original methodology. Specific improvements include rectifying bugs in the original code that resulted in repetitive configurations during each run. The modified approach now allows the program to read a new configuration for each subsequent run. Furthermore, the split learning benchmark of Moradi has been executed, with the addition of integrating the fANOVA to visualize the results, a feature that was absent in Moradi's original code.

2.2 Hyperparameter Importance Frameworks

2.2.1 Overview of fANOVA

fANOVA is a widely known method for determining which hyperparameters are most influential when training machine learning models, especially in high-dimensional hyperparameterspaces [4]. This is especially important in scenarios with advanced models or when working with complex datasets when the correlations between hyperparameters and model performance are not clear. Furthermore, these approaches frequently fail to identify which hyperparameters are critical for optimal performance. fANOVA, on the other hand, tackles these constraints by being specifically designed for the complexities of machine learning. Key aspects of fANOVA include:

1. **Using a Surrogate Model:** Typically, a surrogate model such as a random forest is trained. This captures the relationship between various hyperparameter configurations and the performance of the primary model.

2. **Studying Performance Differences:** fANOVA analyzes the sources of substantial performance changes, assessing whether they are caused by individual hyperparameters or their interactions [13].

3. **Ranking of variance:** It quantifies the effect of each hyperparameter, showing which values are critical to the model's success.

4. **Visualizing Results:** Visualization approaches are used to show how different hyperparameters (or their combinations) affect model outcomes.

Last but not least, fANOVA provides a detailed view of hyperparameter significance, easing the tuning process and assuring more effective model optimization [14].

2.2.2 Understanding Ablation Analysis

On the other hand, **Ablation Analysis** is a methodological approach that has received notable attention in the field of deep learning [15]. Its primary goal is to discern the contribution of individual components, layers, or features of a model towards its overall performance. This is achieved by either removing (or "ablating"), altering, or replacing these elements sequentially and observing the ensuing impact on model outcomes. Goodfellow et al. underline the importance of understanding each component's contribution, especially in complex architectures like deep neural networks, to enhance the model's effectiveness and interpretability [15]. Take, for instance, a deep neural network comprised of several layers. To clarify how each layer contributes to the overall performance, the following approach—which is similar to Zeiler and Fergus'—might be used:

1. Train the network in its entirety and document its performance.

2. Isolate and remove (or modify) a particular layer, subsequently retraining the network, and noting down the performance.

3. Iterate this process, each time reintroducing the previously altered component and selecting another for alteration.

4. Compare the performance of the original network against that of its updated variants.

5. Use visualization techniques to show what each layer has learnt and how it contributes to the overall prediction [16].

If there is a significant drop in performance following the modification, it emphasizes the important role that the component in issue plays. As Zeiler and Fergus point out, if particular layers or features can be changed without significantly affecting performance, they may not be capturing essential information and may indicate that the layer is unnecessary or of decreasing value [16].

2.2.3 Choice of fANOVA Over Ablation Analysis

Therefore, **fANOVA** was chosen over Ablation Analysis because of its ability to consider combinations of hyperparameters. While Ablation Analysis mainly focuses on individual components, fANOVA explores how different hyperparameters work together. This capability of fANOVA provides a comprehensive view of hyperparameter influences, making it more suitable for our complex modeling needs.

2.3 fANOVA

Optimizing hyperparameters is critical in machine learning since they have a major impact on model performance. The **fANOVA** technique, introduced for machine learning by Hutter et al. [4], offers a method to understand these hyperparameters' effects on model outcomes. Notable applications of fANOVA by researchers such as Sharma et al. and Eggensperger et al. have demonstrated its efficacy in pinpointing impactful hyperparameters across various contexts [17, 18].

van Rijn and Hutter [19] investigated hyperparameter optimization in order to find essential hyperparameters and their optimum values. They discovered the significance of particular hyperparameters across different datasets by using functional ANOVA on 100 datasets. For SVMs, the gamma and complexity were critical; for Adaboost, it was the maximum depth and learning rate; and for random forests, it was the minimum number of samples per leaf and maximum features for a split. They ran an optimization experiment, altering one hyperparameter at a time for each classifier. The findings agreed with the functional ANOVA findings and the overall consensus in the field. They also emphasized the minimal influence of data approximation procedures on results. They utilized kernel density estimators on previously successful values to obtain effective hyperparameter values, highlighting the benefit of data-driven priors in hyperparameter optimization compared to uniform prior sampling.

Moradi [11] utilized the **fANOVA** framework to investigate the effect of hyperparameters on machine learning data privacy strategies. The impact of hyperparameters to function performance is measured using functional ANOVA. It assesses the performance of a hyperparameter across many variables using tree-based surrogate models. Moradi's study emphasizes on the larger influence of hyperparameters on overall performance rather than focusing on specialized settings. [11].

This research will further explore the hyperparameters emphasized by Moradi, focusing on their application in data privacy techniques using the **fANOVA** method. The aim is to enhance the understanding and application of privacy-centric machine learning techniques, building on Moradi's work [11, 20].

3 Methods

3.1 fANOVA in Details

Functional ANOVA is a method for determining the significance of an algorithm's hyperparameters. It takes as input performance data collected with various hyperparameter settings of the algorithm, fits a random forest to capture the relationship between hyperparameters and performance, and then applies functional ANOVA to determine how important each hyperparameter and each low-order interaction of hyperparameters is to performance [21].

3.1.1 Decomposing Variance with fANOVA

The basic idea behind fANOVA is to describe the model prediction function $\hat{f}$ in terms of summative functional components for each hyperparameter and their interactions. This is expressed as:

$$\hat{f}(x) = \sum_{S \subseteq \{1,\ldots,p\}} \hat{f}_S(x_S) \tag{1}$$

In Equation 1, the variables represent the following:

- $\hat{f}(x)$: The estimated prediction function of the model, where x is a vector of hyperparameters.
- S: A subset of hyperparameters from the set $\{1, \ldots, p\}$, where p is the total number of hyperparameters.
- $\hat{f}_S(x_S)$: A component of the estimated prediction function $\hat{f}(x)$ that is associated with the subset of hyperparameters S. Here, x_S denotes the values of hyperparameters in subset S.
- $\sum$: The summation symbol denotes the sum of all the functional components $\hat{f}_S(x_S)$ across all possible subsets of hyperparameters S.

Hooker (2004) [22] defines each component with the following formula:

$$\hat{f}_S(x) = \int_{X_{-S}} \left(\hat{f}(x) - \sum_{V \subset S} \hat{f}_V(x) \right) dX_{-S} \tag{2}$$

In Equation 2, the variables represent the following:

- $\hat{f}_S(x)$: As earlier, this represents a component of the estimated prediction function associated with a subset of hyperparameters S.
- X_{-S}: The values of all hyperparameters not in subset S.
- $\int$: The integration symbol, representing integration over the space of hyperparameters X_{-S} not in subset S.
- V: A subset of hyperparameters within the larger subset S.
- $\sum_{V \subset S} \hat{f}_V(x)$: The sum of all functional components $\hat{f}_V(x)$ associated with all subsets V of hyperparameters within the larger subset S.
- dX_{-S}: The differential element for integration over the space of hyperparameters X_{-S} not in subset S.

These equations describe a decomposition of the prediction function $\hat{f}(x)$ into components associated with different subsets of hyperparameters, elucidating how each of these components can be computed via integration over the space of other hyperparameters not in the considered subset. This decomposition facilitates understanding and analyzing the influence of individual hyperparameters and their interactions on the model's predictions. For more details the reader is referred to this website.

3.1.2 Surrogate Models for Computation

Due to computational constraints, it's common to employ surrogate models, such as Random Forests, to estimate $\hat{f}$ [4]. These models are trained on the primary dataset and are leveraged to compute the effects of hyperparameter variations without the extensive retraining of the main model.

3.2 Hyperparameters

Hyperparameters are critical in machine learning because they influence the model's learning trajectory, generalization ability, and overall performance. A solid understanding of these characteristics is required for improving the training process and achieving the desired results. In the following discussion, hyperparameters are divided into two types: tunable hyperparameters, which practitioners may alter, and contextual hyperparameters, whose values are determined by the issue setting or external restrictions.

3.2.1 Tunable Hyperparameters

These hyperparameters are adjustable and play a direct role in the learning process.

- **Batch size:** Determines the number of training samples used in a single gradient descent iteration. Smaller batch sizes offer more frequent updates and potentially faster convergence, albeit at the risk of a less stable training process. Conversely, larger batch sizes provide more accurate gradient estimates, better generalization but demand more memory [23].
- **Learning rate:** Controls the step size during gradient descent optimization, impacting the convergence speed and accuracy [24].
- **Weight decay:** A regularization term reducing overfitting by penalizing large weights [25].
- **Epochs:** Specifies the total passes through the entire training dataset [15].
- **Number of local updates:** Dictates the local update frequency before communication with the server in Federated Learning [2].
- **Aggregation type:** Defines the aggregation method for combining model updates from different clients in Federated Learning [2, 26].
- **Random seed:** Ensures reproducibility by initializing a fixed sequence of random numbers [27].

3.2.2 Contextual Hyperparameters

These parameters are often predetermined by the problem setting or external constraints, hence not freely tunable.

- **Partition alpha:** Controls data partitioning degree in Federated Learning, affecting the data distribution among clients [28].

- **Total number of clients:** Represents the total client number in Federated Learning, affecting model diversity and generalization [5].
- **Cut layer:** In Split Learning, denotes the division point in the neural network between the client and server, balancing privacy preservation against model performance [3].

4 Experimental Setup and Implementation

Building upon the methodology described in the previous section, this part of the study focused on identifying the most influential hyperparameters and their impact on test set accuracy. An extensive series of experiments were conducted, examining all available hyperparameters in the context of the Split Learning approach.

Prior to applying the fANOVA technique, a series of preliminary experiments were conducted to gain an initial understanding of the hyperparameters' individual and interactive effects on the model performance. In these preliminary experiments, one hyperparameter was varied while the others were held constant in each run, creating a controlled environment to assess the individual impact of each parameter. This step aimed to provide a foundational understanding of the hyperparameters' behavior which would later be further examined and validated using the fANOVA technique.

4.1 Configuration Space

This study was conducted with an emphasis on four key hyperparameters: **batch size**, **total number of clients**, **cut layer**, and **partition alpha**. These hyperparameters were chosen because they are crucial for the workings of Split Learning models. By examining these specific hyperparameters, the study aimed to uncover how they individually and collectively affect the model's performance. Additionally, the choice of these hyperparameters was partly dictated by the availability of varied values in the configuration files, as other hyperparameters held constant values across all configurations, limiting the scope for their analysis. some hyperparameters are used to simulate an experimental setting (number of clients and partition alpha), as these are in practical settings not under the control of the data scientist. Other hyperparameters (e.g. batch size) are actual hyperparameters that are under the control of the data scientist. They were defined as the feature set 'X' for the analysis, with the corresponding model accuracy set as the target 'y'.

The data was split into training and testing sets, maintaining a 70-30 ratio, to enable a decent evaluation of model performance. A critical aspect of this process was ensuring that the distribution of data in the training set accurately represented the complete dataset. Additionally, a fixed seed (42) was set for the random number generator used in the split to ensure reproducibility of results.

In this analysis, a structured representation of the hyperparameters was established using a Configuration Space. This space defines the boundaries within which each hyperparameter can vary. The minimum and maximum values for each hyperparameter were identified from the original data, ensuring a comprehensive exploration within the actual range of values each hyperparameter can

take.

The hyperparameters were then defined as UniformHyperparameters within this Configuration Space. This reflects the assumption that any value within the given range of each hyperparameter is equally likely to be selected.

Table 1: The hyperparameters used and their ranges

Hyperparameter	Range	Description
`partition alpha`	0.1 - 20.0	The value used for partitioning the dataset during training.
`batch size`	8 - 128	The number of samples in each mini-batch during training.
`cut layer`	2.0 - 9.0	The depth at which the neural network model is cut for transfer learning.
`total number of clients`	1 - 19	The total number of clients participating in the split learning process.

4.2 Implementation of fANOVA for Datasets

To apply fANOVA on our datasets, several preparatory steps were taken. Initially, a dataset was generated by running the benchmark for various configurations of the Split Learning model. In each run, the model's accuracy was evaluated and logged along with the corresponding hyperparameter settings. This data was saved to a separate file, which was then loaded for the fANOVA analysis. The dataset containing accuracy values corresponding to different configurations was used to assess the impact of hyperparameters on model performance. Subsequently, the data was split into training and testing subsets to ensure an unbiased evaluation of the model. Notably, the target variable, in this case, the accuracy, was normalized to ensure the stability of the surrogate model.

The Configuration Space, which defines the range and type of each hyperparameter, was set up based on the minimum and maximum values from the dataset containing the accuracy values and hyperparameter configurations. This space serves as the domain over which the surrogate model makes its predictions.

Upon setting up the Configuration Space, the fANOVA method was invoked to derive the importance of each hyperparameter. A dictionary was constructed to store the computed importance for every hyperparameter. In addition to these numerical results, visualizations were created to offer an intuitive understanding of hyperparameter importances and were saved for further analysis.
With this methodology, we not only gained insights into which hyperparameters had a significant impact on the model's performance but also visualized their effects, aiding in the informed decision-making process for future optimization tasks.

4.3 Data Sets

The experiments were conducted using a total of 11 distinct databases, which are detailed below:

Mice Protein: The dataset comprises expression levels of 77 proteins or protein modifications detectable in the nuclear fraction of the cortex. It includes data from 72 mice, with 38 control mice and 34 trisomic mice (Down syndrome). In total, the dataset contains 1080 measurements per protein, with 15 measurements recorded for each protein per sample/mouse. This results in 570 measurements for control mice and 510 measurements for trisomic mice, with each measurement treated as an independent sample/mouse. The mice are classified into eight groups based on characteristics such as genotype, behavior, and treatment. Genotypically, the mice are categorized as control or trisomic. Behaviorally, some mice undergo learning stimulation (context-shock), while others do not (shock-context). To evaluate the effect of memantine, a drug aimed at restoring learning ability in trisomic mice, some mice receive the drug, and others serve as a control group without the treatment. [29]

Segment: The Segment dataset is a valuable resource for image segmentation research, comprising outdoor image data. The instances within the dataset were drawn randomly from a database of 7 outdoor images, which were hand- segmented to create a classification for every pixel. Each instance represents a 3x3 region. The dataset contains 2310 instances, aiming to classify regions into seven predefined classes, such as brickface or sky. Instances are described by 19 continuous attributes capturing color, texture, and shape features. However, major changes have been made with respect to version 2 of the dataset: the first two variables should be ignored, as they do not fit the classification task and merely reflect the location of the sample in the original image. Additionally, the third variable is constant and should also be disregarded.

The Segment dataset serves as a useful resource for researchers and practitioners in computer vision, machine learning, and image processing who seek to develop, refine, and evaluate segmentation algorithms tailored to outdoor scenes. [30]

Bioresponse: The Bioresponse dataset focuses on predicting the biological response of molecules using their chemical properties. Each row in the dataset represents an individual molecule. The first column contains experimental data, indicating whether the molecule elicited a specific biological response (1) or not (0). The remaining columns represent molecular descriptors (d1 through d1776), which are calculated properties that capture certain characteristics of the molecule, such as size, shape, or elemental composition. The descriptor matrix has been normalized to facilitate analysis.

The original training and test sets have been merged to form a comprehensive dataset for the study. This dataset provides valuable insights for researchers and practitioners in the fields of chemistry, biology, and data science who are interested in predicting the biological response of molecules based on their inherent chemical properties. [31]

Internet-Advertisements: The Internet-Advertisements dataset contains information on various features of images found on the internet. These features include the image's geometry (if available), phrases present in the URL, the image's URL and alt text, the anchor text, and words occurring in close proximity to the anchor text. The primary objective of this dataset

is to predict whether an image serves as an advertisement ("ad") or not ("nonad").
The dataset comprises a total of three continuous attributes, while the remaining attributes are binary in nature. This dataset offers valuable insights for researchers and practitioners in the fields of advertising, computer vision, and data science, who are interested in developing algorithms to distinguish between advertisement and non-advertisement images. [32]

Isolet: The ISOLET (Isolated Letter Speech Recognition) dataset was created by collecting speech samples from 150 participants, each of whom pronounced the name of every letter in the alphabet twice. As a result, each speaker contributed 52 training examples. The dataset is organized into groups of 30 speakers, with four groups serving as the training set and the last group as the test set. Due to recording difficulties, three examples are missing and were excluded from the dataset.
The ISOLET dataset represents a suitable domain for studying noisy, perceptual tasks and examining the scalability of algorithms in such contexts. Interestingly, on this domain, the C4.5 algorithm performs slower than backpropagation, demonstrating the unique challenges posed by the dataset in the realm of speech recognition and machine learning. [33]

HAR: The Human Activity Recognition (HAR) dataset is derived from 30 subjects performing daily activities while wearing a waist-mounted smartphone with embedded inertial sensors. The participants, aged 19- 48 years, engaged in six activities, such as walking and sitting. Data was captured using the smartphone's accelerometer and gyroscope at a rate of 50Hz. The dataset was divided into training (70%) and testing (30%) subsets.
Sensor signals were pre-processed using noise filters and sampled in fixed-width sliding windows of 2.56 seconds with 50% overlap. Acceleration signals were separated into body acceleration and gravity components using a Butterworth low- pass filter. From each window, a 561-feature vector was obtained by calculating time and frequency domain variables.
The dataset provides records with triaxial acceleration, triaxial angular velocity, a 561-feature vector, and the corresponding activity label, allowing for the development and testing of human activity recognition algorithms. [34]

DNA: The Primate Splice-Junction Gene Sequences (DNA) dataset, originally from the StatLog project, comprises 3,186 data points representing splice junctions. The objective is to classify these junctions into three classes (ei, ie, neither), which signify the boundaries between exons (DNA sequence parts retained after splicing) and introns (spliced-out DNA sequence parts). The dataset underwent processing to transform the original 60 symbolic attributes into 180 binary attributes by converting nucleotides (A, G, T, C) into three binary indicator variables. Examples with ambiguities were removed, and the StatLog version was created by Ross King at Strathclyde University. The nucleotides were assigned indicator values as follows: A $\rightarrow$ 1 0 0, C $\rightarrow$ 0 1 0, G $\rightarrow$ 0 0 1, and T $\rightarrow$ 0 0 0. This dataset allows researchers to develop algorithms for splice-junction gene sequence recognition and classification. [35]

Nomao: The Nomao dataset, developed by Nomao Labs, contains 34,465 instances with a mix of continuous and nominal attributes, labeled by human experts. This dataset was enriched during the Nomao Challenge, organized alongside the ALRA workshop (Active Learning in Real-world Applications) held at the ECML- PKDD 2012 conference.
The dataset consists of 120 attributes, including 89 continuous and 31 nominal attributes,

featuring 'label' and 'id' as well. The instances are divided into various sections based on the labeling methods used. The first 29,104 instances were labeled with "human prior," while the subsequent instances were labeled using different active learning methods such as "marg," "wmarg," "wmarg5," and "rand" (random selection). [36]

Ozone-level-8hr: The ozone-level-8hr dataset contains attributes related to temperature (prefix "T") and wind speed (prefix "WS") measurements taken at various times throughout the day. The dataset also includes measurements at different atmospheric pressure levels (850 hpa, 700 hpa, and 500 hpa) for temperature, relative humidity, wind, and geopotential height. Key attributes involve local ozone peak prediction, upwind ozone background level, precursor emissions-related factor, maximum temperature, base temperature where net ozone production begins (50 F), solar radiation total for the day, and wind speed near sunrise and mid-day. These attributes are highly valued by the Texas Commission on Environmental Quality (TCEQ). Further details can be found in the relevant papers associated with this dataset. [37]

Optdigits: The Optical Recognition of Handwritten Digits dataset comprises preprocessed and normalized bitmaps of handwritten digits extracted from preprinted forms. With contributions from 43 individuals, 30 provided data for the training set, while the remaining 13 contributed to the test set. The 32x32 bitmaps were partitioned into non-overlapping 4x4 blocks, with the number of on pixels counted in each block. This process resulted in an 8x8 input matrix, with each element being an integer ranging from 0 to 16. By reducing dimensionality and providing invariance to minor distortions, this dataset allows for the analysis and recognition of handwritten digits. [38]

Splice: The Primate splice-junction gene sequences (DNA) dataset is accompanied by an associated imperfect domain theory. Splice junctions represent locations on a DNA sequence where extraneous DNA is eliminated during protein synthesis in higher organisms. The challenge presented by this dataset involves identifying the boundaries between exons (DNA segments retained after splicing) and introns (DNA segments removed during splicing) within a given DNA sequence. This task consists of two subtasks: detecting exon/intron boundaries (known as EI sites) and identifying intron/exon boundaries (referred to as IE sites). In biological research, IE borders are commonly termed "acceptors," while EI borders are called "donors." [39]

5 Results

In this research, Moradi's Benchmark was utilized as a foundation. However, some results varied slightly from the original findings. This variation can be attributed to modifications implemented in the benchmark's code to address certain anomalies and bugs in the original code that resulted in repetitive configurations during each run. The modified approach now allows the program to read a new configuration for each subsequent run.

In this section, the results obtained from applying the fANOVA method to each individual dataset are presented in the form of graphs. For each dataset, 800 distinct instances of hyperparameters were evaluated to gain a comprehensive understanding of their impact on performance. The application of fANOVA allowed for an analysis and quantification of the significance of each hyperparameter,

providing valuable insights into their contributions to the overall performance across different datasets.

The following graphs illustrate the specific implications of hyperparameter adjustments and their respective importance in the context of each dataset. By examining these visual representations, a deeper understanding of the relationships between hyperparameters and their influence on the performance of the model can be gained.

5.1 Performance Results

Although the primary objective of this research is not to achieve state-of-the-art performance results, it's important to ensure that the performance outcomes align with the high-quality results documented in the literature to validate the findings. Figure 3 represents the predictive accuracy and the run-time of all the various hyperparameter configurations per dataset in a box-plot format.

5.2 Marginals per Dataset

The detailed results for each dataset are moved to an appendix for brevity. This section now provides a summary of those results, particularly focusing on the general trends and behaviors of the hyperparameters and their interactions.

Please refer to Appendix A for the comprehensive results.

The following provides an overview of the general trends observed for each hyperparameter and their interactions:

Cut Layer: For the majority of the datasets, a descending trend in model performance was observed as the cut layer value increased. A performance dip was particularly noticeable when the cut layer value reached approximately 7 or 8.

Batch Size: Batch size displayed a generally decreasing trend in most datasets. However, in certain cases, it exhibited an initial increase followed by a decrease.

Partition Alpha: The trend for partition alpha was found to fluctuate across many of the datasets. However, a clear downward trend was observed after the value reached 10.

Total Number of Clients: The results for this parameter were more complex, with three distinct trends observed:

- General fluctuation
- A decrease after reaching certain values, such as 14 or 15
- A general increase or a sharp spike at the value 14

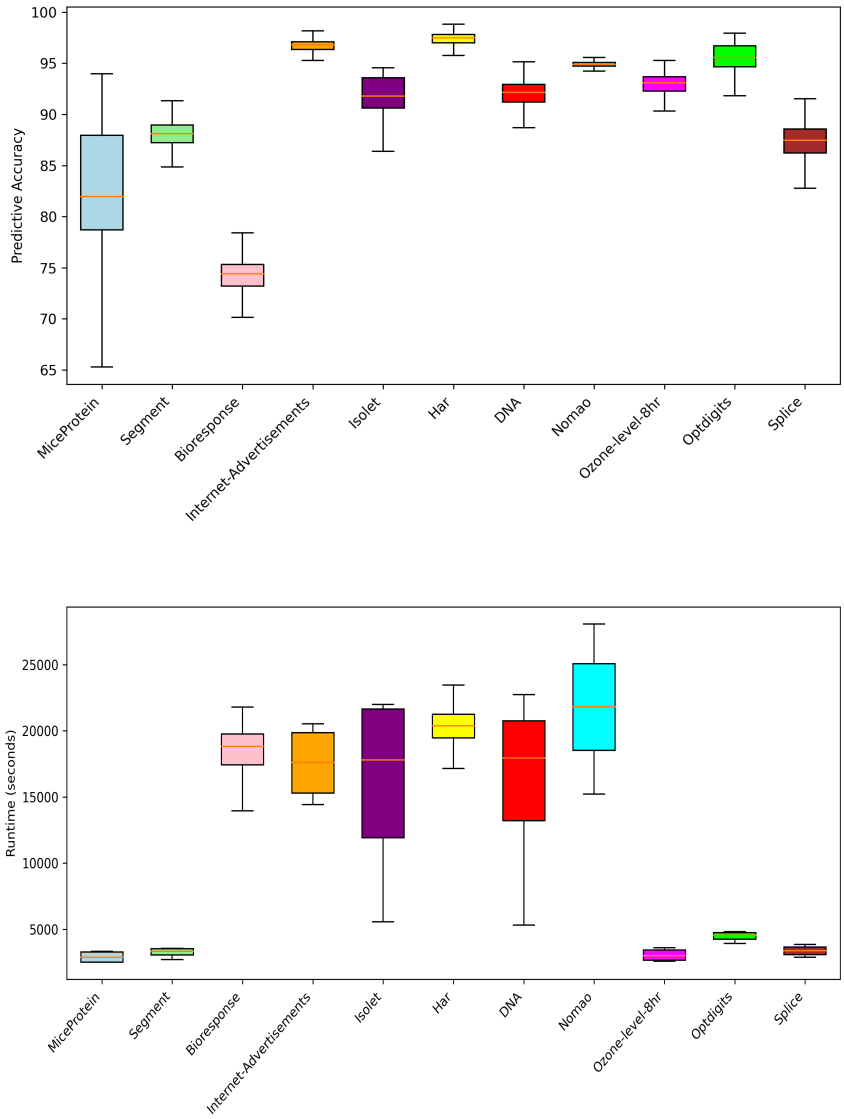


Figure 3: The performance results, including predictive accuracy and run time in seconds, are reported for each dataset and the different configurations. Run time was measured for each configuration during the benchmarking phase, with the values being noted from the runs stored on the Weights and Biases platform.

Batch Size & Cut Layer Interaction: Across all datasets, optimal performance was observed when the batch size was 20 and the cut layer was 2. Beyond these values, a downward trend was seen.

Batch Size & Total Number of Clients Interaction: In most instances, optimal performance was achieved when both parameters had smaller values, with performance declining as these values increased.

Batch Size & Partition Alpha Interaction: For all datasets, the best performance was achieved when both hyperparameters were at their minimum values.

Total Number of Clients & Cut Layer Interaction: In the majority of cases, smaller cut layer values led to optimal performance.

Total Number of Clients & Partition Alpha Interaction: Performance was generally optimal when the number of clients was low.

Cut Layer & Partition Alpha Interaction: While performance values varied, in most instances, performance was best when the partition alpha value was low.

5.3 Importance Across Datasets

Based on the insights derived from the fANOVA, as shown in Figure 4a, partition alpha appears to have the most significant influence on model performance, whereas batch size seems to have the least impact. The total number of clients and the cut layer have a similar level of impact on performance.

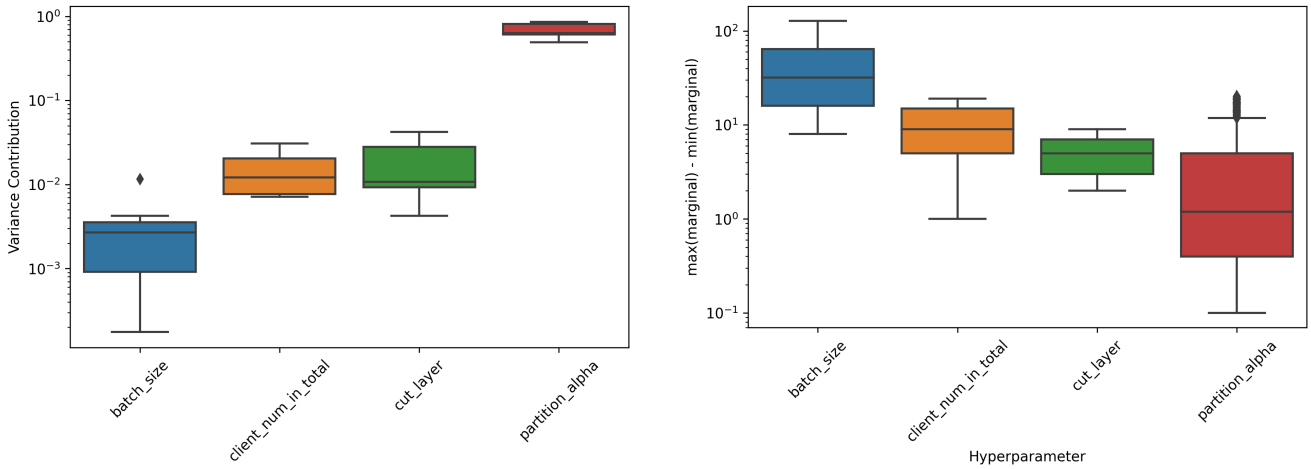
Figure 4b presents a depiction of the marginal maximum and minimum values used in the fANOVA for each hyperparameter. The chart helps illustrate the range of influence for each hyperparameter. The partition alpha appears to have the widest range of impact, aligning with its status as the most influential hyperparameter. Conversely, the batch size has the smallest range, aligning with its lower influence on performance.

In conclusion, these results highlight the importance of each hyperparameter and the varying degrees of impact they have on model performance. These insights can inform future optimization and tuning efforts in Federated Learning and Split Learning applications.

5.4 Discussion

5.4.1 Weaknesses

Reflecting on the methods and design utilized in this study, a number of limitations and areas for potential improvement become evident. The observed trends in hyperparameters and their interactions, though insightful, are largely dependent on the specific datasets used. Thus, the generality of these findings across other data contexts remains an open question.



(a) Functional ANOVA

(b) Marginal max-min

Figure 4: The importance of each combination of hyperparameters across datasets is assessed.

One notable weakness was the simplicity of the model's design. This simple design helped to save computing time and made the analysis easier to understand. However, it may not provide an entire overview of how hyperparameters work in more complex models. Our simple model may not learn as well from the data, perform as well in real-world situations, or show all the effects and interactions of hyperparameters compared to more complex models. Choosing a simple model was a trade-off. It made our study easier and faster, but might not give a full understanding of hyperparameter dynamics, especially when compared to more complex models.

The choice of specific hyperparameter ranges was also a potential limitation. While the range values were chosen based on the initial data analysis and were designed to explore a variety of model behaviors, they may not encompass the entire space of potential performance outcomes.

Based on preliminary data analysis, the selected hyperparameter ranges may not completely represent the large spectrum of potential model behaviors, necessitating a broader or more dynamic range decision in the future.

In order to sum up, while the approach of this work provides important insights into hyperparameters and their interactions, there are evident areas for refinement and further research. Recognizing these flaws provides direction for future study and improves comprehension of model optimization complexities.

6 Conclusion

The purpose of this research is to investigate the influence of hyperparameters in the Split Learning machine learning technique. We concentrated on four major hyperparameters: cut layer, batch size, partition alpha, and total number of clients. We used the fANOVA tool to gain an improved understanding of the influence of these hyperparameters on model performance.

It is important to emphasize the contrast between hyperparameters. While "batch size" is a variable that may be changed at any time, "cut layer," "partition alpha," and "total number of clients" frequently draw their values from external circumstances, such as privacy limitations for "cut layer."

Our findings revealed complexities in hyperparameter behavior. Specifically, when increasing the cut layer and batch size values, model performance often declined, although there were exceptions. The behavior of partition alpha varied based on the dataset, especially when its value exceeded 10. The total number of clients presented varied outcomes, with some configurations enhancing and others reducing performance.

When comparing our results to Moradi's study [11], differences were evident, emphasizing the importance of precise hyperparameter adjustments. This research provides insights for those in the machine learning field, suggesting that for optimal Split Learning outcomes, careful hyperparameter tuning based on methodical testing is crucial.

6.1 Future work

Looking ahead, numerous exciting options for additional investigation and extension of the current work open themselves. To begin, a logical next step would be to investigate the influence of hyperparameters and their interactions in increasingly complicated model designs. While the current study employed a simple model for interpretability and computational efficiency, more advanced models might give alternative insights on the roles of hyperparameters.

Second, the space of hyperparameters might be expanded. The current study only looked at four hyperparameters and their relationships, however there are many more that may be studied. This expanding would provide a deeper knowledge of the hyperparameter environment and its impact on model performance.

Additionally, a more dynamic determination of hyperparameter ranges could be implemented. While the ranges in this study were set based on initial data analysis, a method that dynamically adjusts these ranges based on observed model performance could be more effective.

Lastly, the application of these findings to other domains or datasets would be an interesting direction. While the current study was limited to specific datasets, the methods and insights could be applied to other areas to evaluate their generalizability.

To summarize, while the current study offers valuable insights into the impact of hyperparameters on model performance, there are numerous opportunities for extension and further exploration. The study of hyperparameters and their interactions remains a rich and important area of research in the field of machine learning.

References

[1] European Parliament and Council, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)." Official Journal of the European Union, 2016.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and X. J. Zhu, eds.), pp. 1273–1282, PMLR, 2017.

[3] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *CoRR*, vol. abs/1812.00564, 2018.

[4] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, vol. 32 of *JMLR Workshop and Conference Proceedings*, pp. 754–762, JMLR.org, 2014.

[5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019.

[6] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Medical Informatics*, vol. 112, pp. 59–67, 2018.

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 12:1–12:19, 2019.

[8] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *CoRR*, vol. abs/1811.03604, 2018.

[9] Google, "Federated learning: Collaborative machine learning without centralized training data," 2017.

[10] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Reducing leakage in distributed deep learning for sensitive health data," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1040–1044, IEEE, 2019.

[11] A. Moradi, "Federated learning: Literature review and hyperparameter importance," master's thesis, Leiden University, 2021.

[12] M. Feurer, J. N. van Rijn, A. Kadra, P. Gijsbers, N. Mallik, S. Ravi, A. Müller, J. Vanschoren, and F. Hutter, "Openml-python: an extensible python API for openml," *J. Mach. Learn. Res.*, vol. 22, pp. 100:1–100:5, 2021.

[13] P. Probst, M. N. Wright, and A. Boulesteix, "Hyperparameters and tuning strategies for random forest," *WIREs Data Mining Knowl. Discov.*, vol. 9, no. 3, 2019.

[14] P. Probst, A. Boulesteix, and B. Bischl, "Tunability: Importance of hyperparameters of machine learning algorithms," *J. Mach. Learn. Res.*, vol. 20, pp. 53:1–53:32, 2019.

[15] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*. Adaptive computation and machine learning, MIT Press, 2016.

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I* (D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8689 of *Lecture Notes in Computer Science*, pp. 818–833, Springer, 2014.

[17] A. Sharma, J. N. van Rijn, F. Hutter, and A. Müller, "Hyperparameter importance for residual neural networks," in *Discovery Science - 22nd International Conference, DS 2019, Split, Croatia, October 28-30, 2019, Proceedings*, vol. 11828 of *Lecture Notes in Computer Science*, pp. 112–126, Springer, 2019.

[18] K. Eggensperger, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Efficient benchmarking of hyperparameter optimizers via surrogates," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (B. Bonet and S. Koenig, eds.), pp. 1114–1120, 2015.

[19] J. N. van Rijn and F. Hutter, "Hyperparameter importance across datasets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018* (Y. Guo and F. Farooq, eds.), pp. 2367–2376, ACM, 2018.

[20] A. Moradi, "Federated learning benchmarks." https://github.com/maminio/fed-benchmarks, 2021.

[21] AutoML, "fanova: Functional anova: an efficient tool for assessing feature importance." https://www.automl.org/automl/fanova/, 2023. Accessed: October 19, 2023.

[22] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. URL: https://christophm.github.io/interpretable-ml-book, 2 ed., 2022.

[23] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *19th International Conference on Computational Statistics* (Y. Lechevallier and G. Saporta, eds.), pp. 177–186, Physica-Verlag, 2010.

[24] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[25] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems 4* (J. E. Moody, S. J. Hanson, and R. Lippmann, eds.), pp. 950–957, Morgan Kaufmann, 1991.

[26] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *CoRR*, vol. abs/1907.02189, 2019.

[27] N. Reimers and I. Gurevych, "Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging," *CoRR*, vol. abs/1707.09861, 2017.

[28] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

[29] C. Higuera, K. J. Gardiner, and K. J. Cios, "Miceprotein dataset," 2015. Available at: https://www.openml.org/searchtype=data&sort=runs&id=4550&status=active.

[30] U. of Massachusetts Vision Group, "Segment data set." https://archive.ics.uci.edu/ml/datasets/segment, 1990. Contact person: Carla Brodley.

[31] B. Ingelheim, "Bioresponse data set." https://archive.ics.uci.edu/ml/datasets/Bioresponse, 2014.

[32] N. Kushmerick, "Internet advertisements data set." https://archive.ics.uci.edu/ml/datasets/Internet+Advertisements, 1999.

[33] R. Cole and M. Fanty, "Isolet data set." https://archive.ics.uci.edu/ml/datasets/Isolet, 1997. Donor: Tom Dietterich.

[34] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition using smartphones data set." https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones, 2012. Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition Using Smartphones. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.

[35] R. King, "Dna data set." https://archive.ics.uci.edu/ml/datasets/DNA, 2014. Based on data from Genbank 64.1.

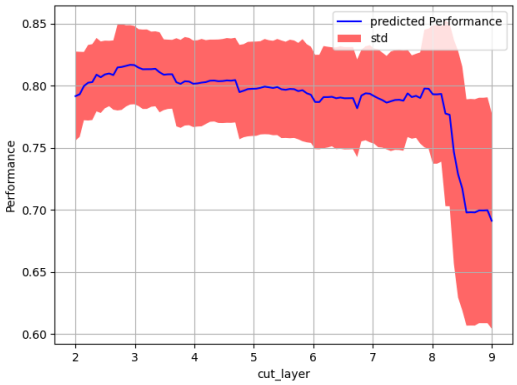
[36] N. Labs, "Nomao data set." https://archive.ics.uci.edu/ml/datasets/Nomao, 2010. Laurent Candillier and Vincent Lemaire. Design and Analysis of the Nomao Challenge - Active Learning in the Real-World. In: Proceedings of the ALRA : Active Learning in Real-world Applications, Workshop ECML-PKDD 2012, Friday, September 28, 2012, Bristol, UK.

[37] K. Zhang, W. Fan, and X. Yuan, "Ozone level detection data set." https://archive.ics.uci.edu/ml/datasets/Ozone+Level+Detection, 2008. Kun Zhang, Wei Fan, XiaoJing Yuan. Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond, Knowledge and Information Systems, Vol. 14, No. 3, 2008.

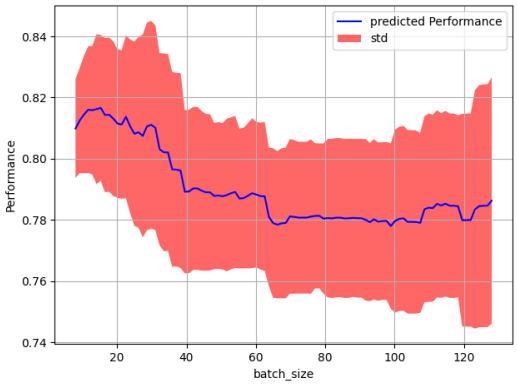
[38] E. Alpaydin and C. Kaynak, "Optical recognition of handwritten digits data set." https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits, 1998. Dua, Dheeru and Graff, Casey. UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2017. http://archive.ics.uci.edu/ml.

[39] Genbank, "Splice junction gene sequences data set." https://archive.ics.uci.edu/ml/datasets/Splice+Junction+Gene+Sequences, 1992. Donated by G. Towell, M. Noordewier, and J. Shavlik.

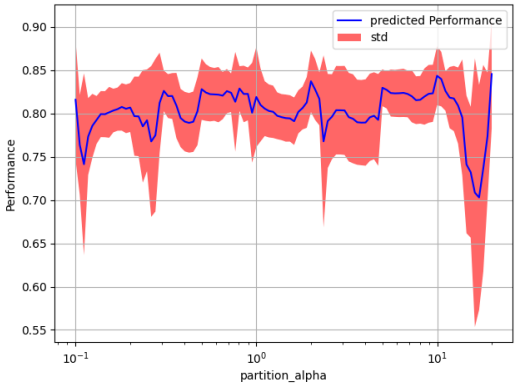
A Appendix



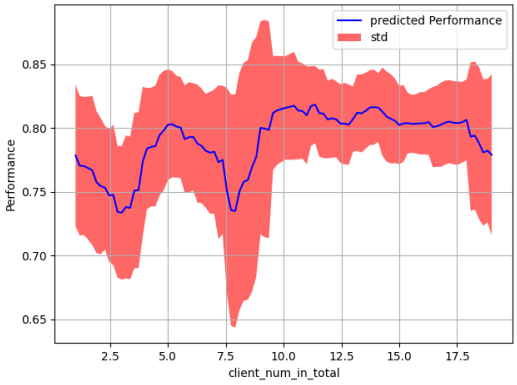
(a) Cut layer



(b) Batch size

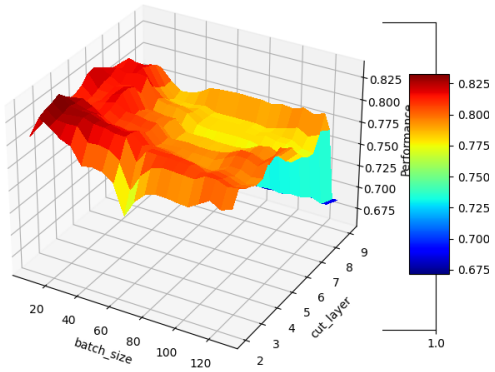


(c) Partition alpha

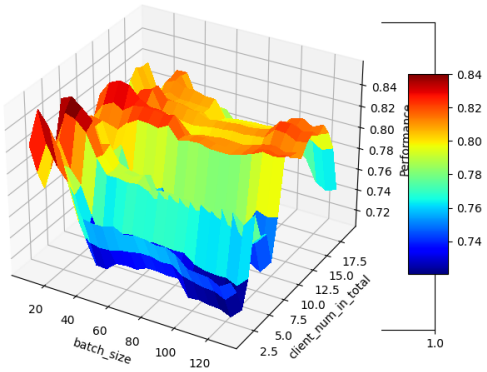


(d) Total number of clients

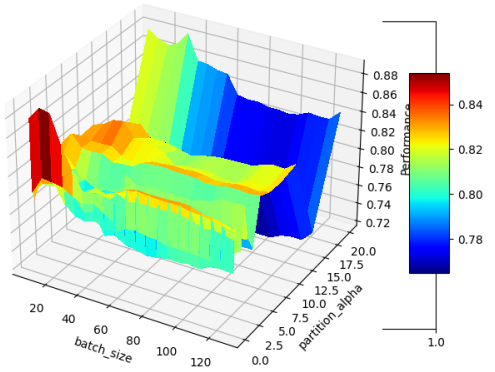
Figure 5: **Mice Protein** Importance of each hyperparameter



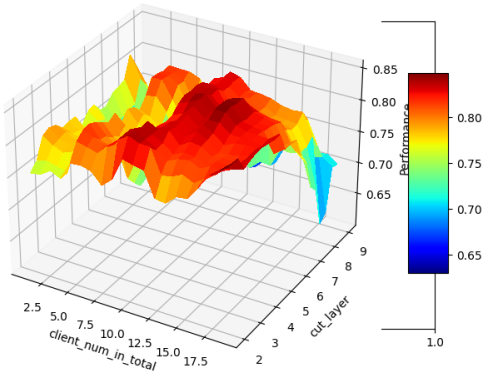
(a) Batch size & Cut layer



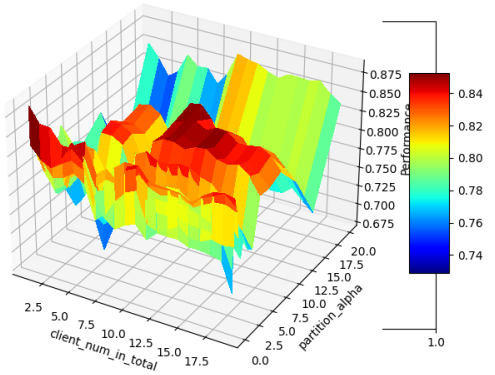
(b) Batch size & Total number of clients



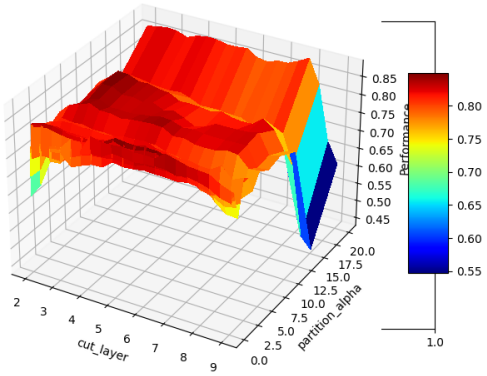
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

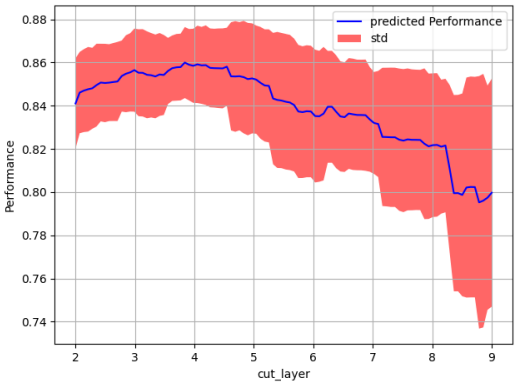


(e) Total number of clients & Partition alpha

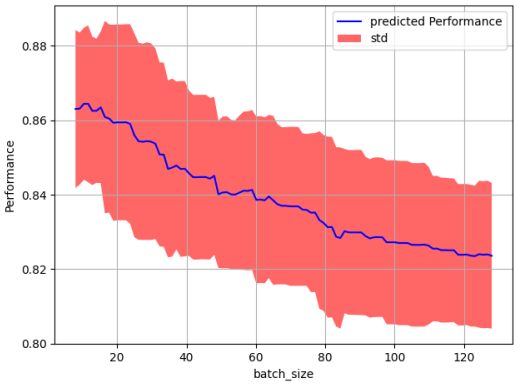


(f) Cut layer & Partition alpha

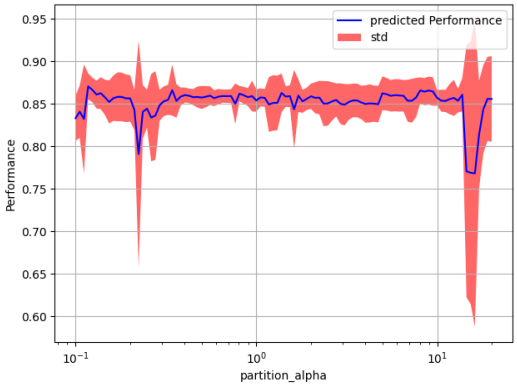
Figure 6: **Mice Protein** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



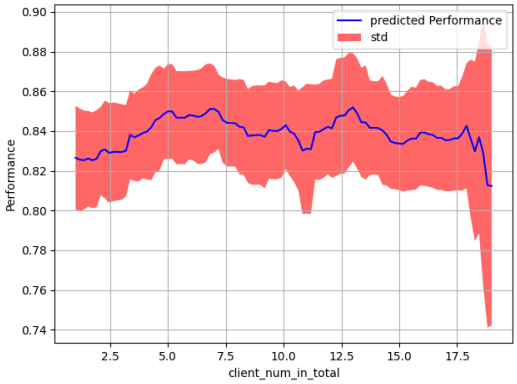
(a) Cut layer



(b) Batch size

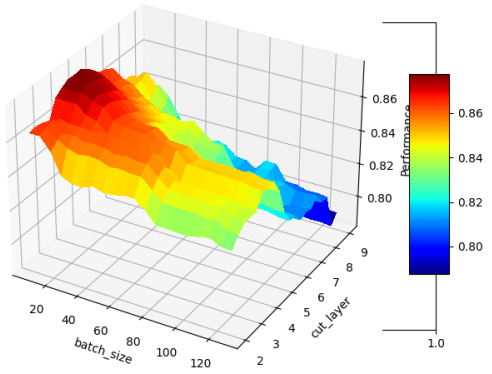


(c) Partition alpha

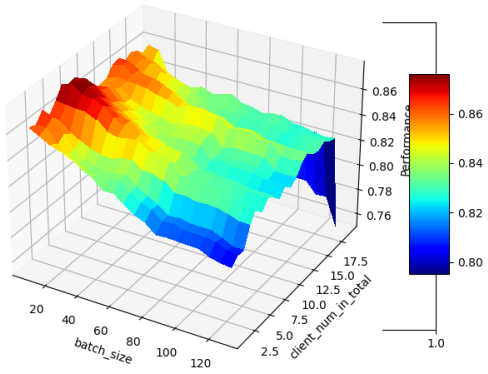


(d) Total number of clients

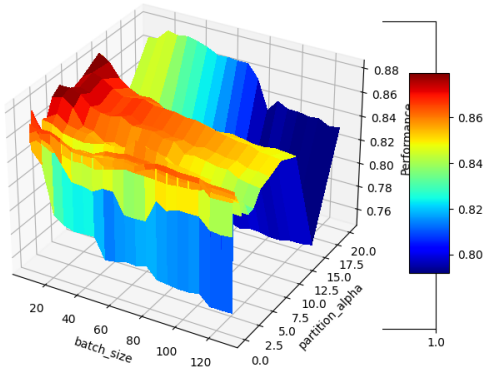
Figure 7: **Segment** Importance of each hyperparameter



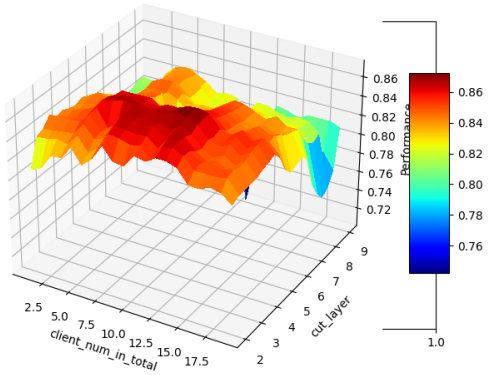
(a) Batch size & Cut layer



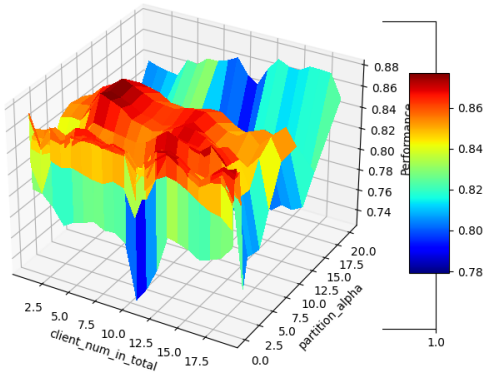
(b) Batch size & Total number of clients



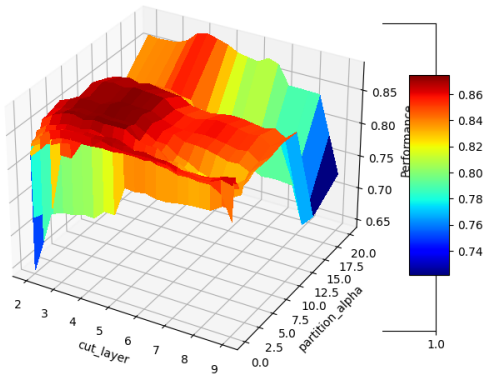
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

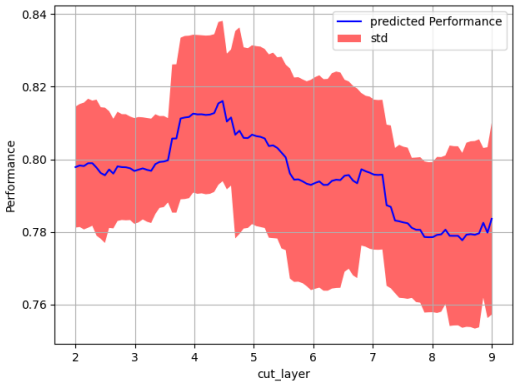


(e) Total number of clients & Partition alpha

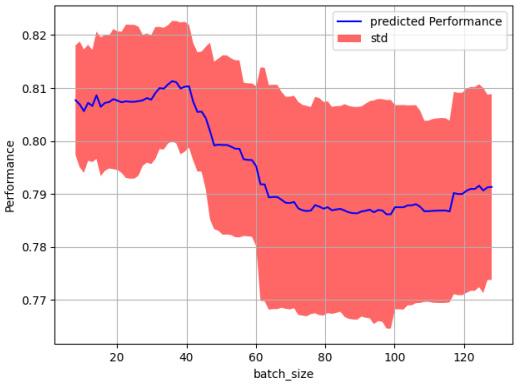


(f) Cut layer & Partition alpha

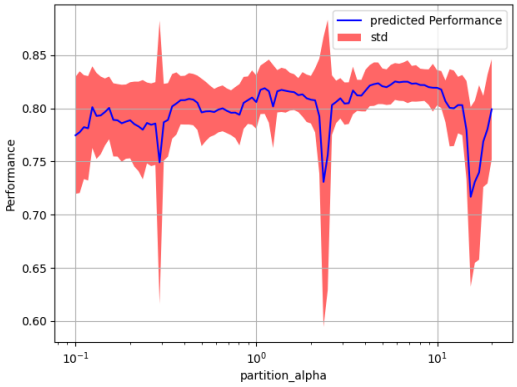
Figure 8: **Segment** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



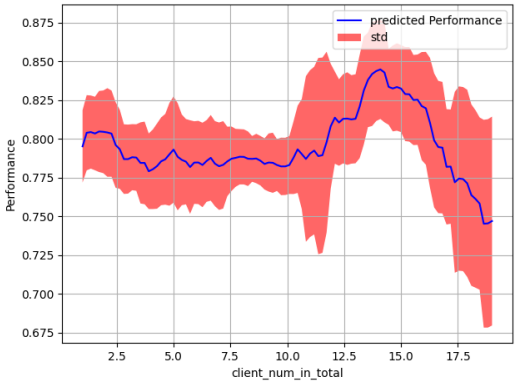
(a) Cut layer



(b) Batch size

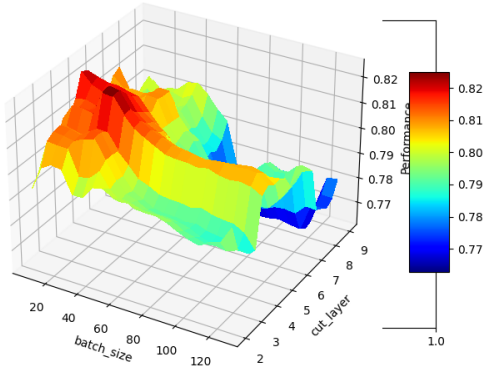


(c) Partition alpha

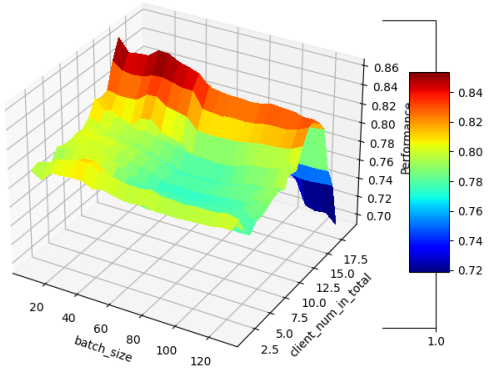


(d) Total number of clients

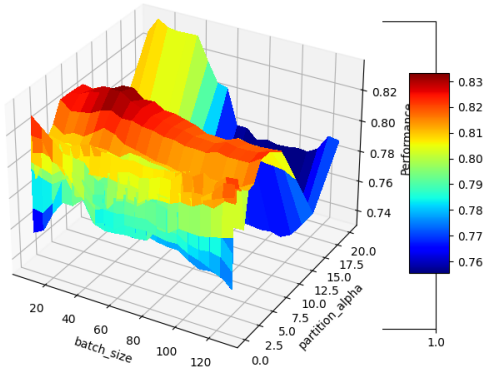
Figure 9: **Bioresponse** Importance of each hyperparameter



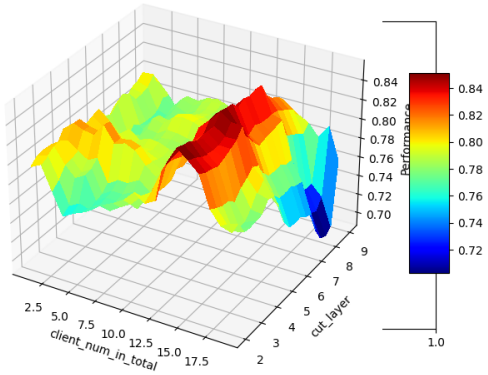
(a) Batch size & Cut layer



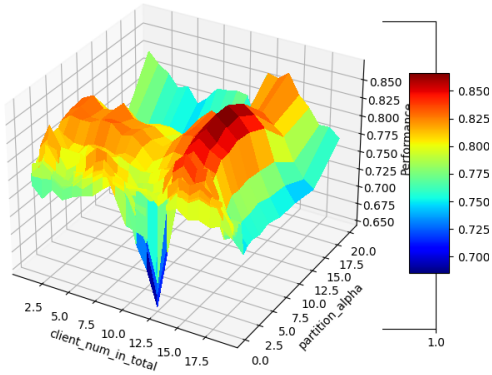
(b) Batch size & Total number of clients



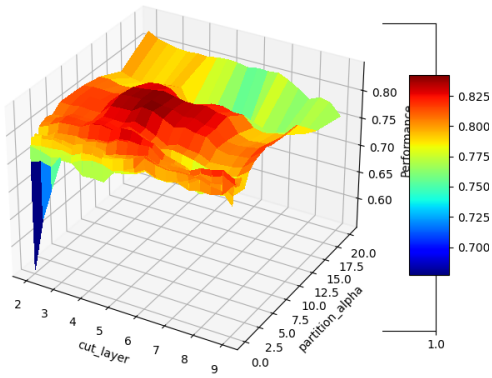
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

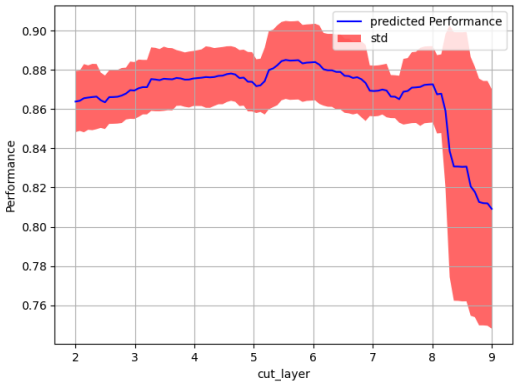


(e) Total number of clients & Partition alpha

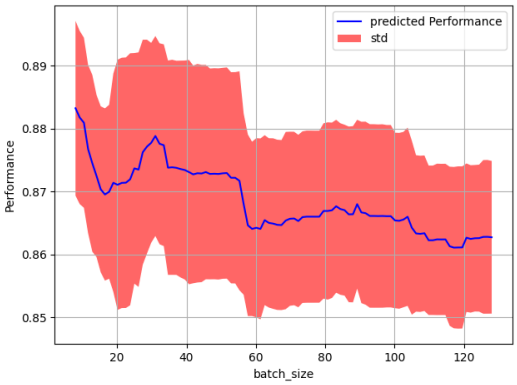


(f) Cut layer & Partition alpha

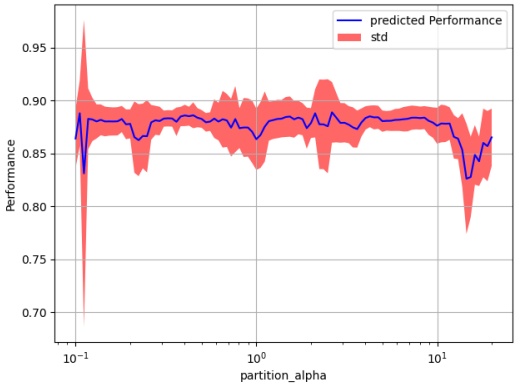
Figure 10: **Bioresponse** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



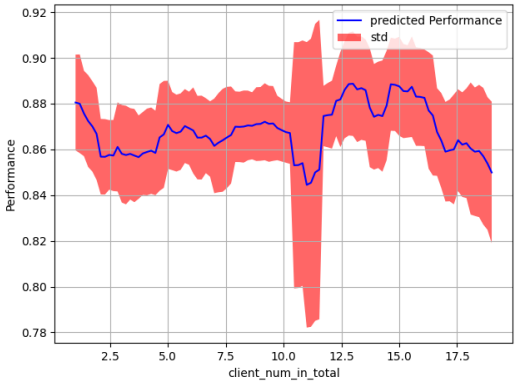
(a) Cut layer



(b) Batch size

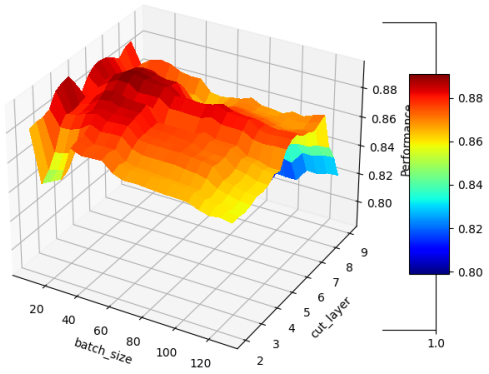


(c) Partition alpha

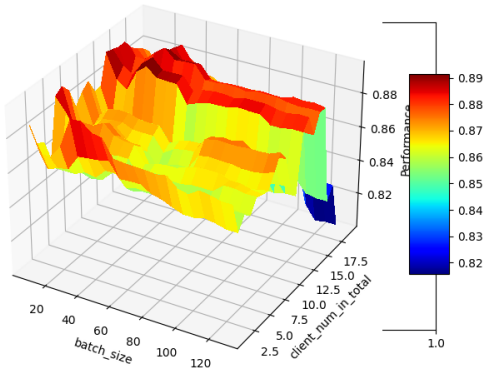


(d) Total number of clients

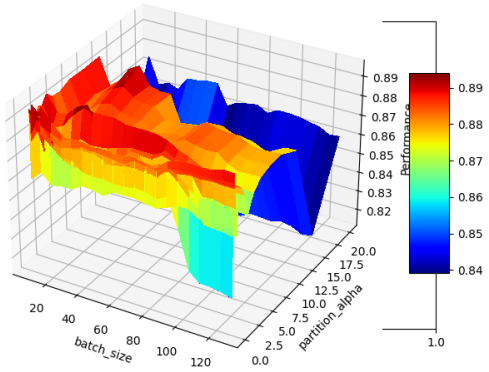
Figure 11: **Internet-Advertisements** Importance of each hyperparameter



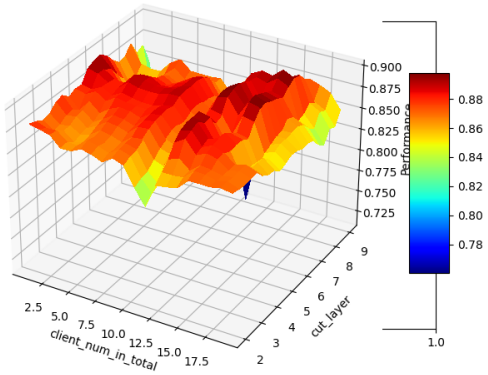
(a) Batch size & Cut layer



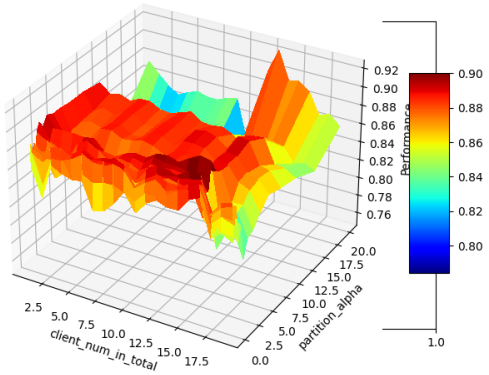
(b) Batch size & Total number of clients



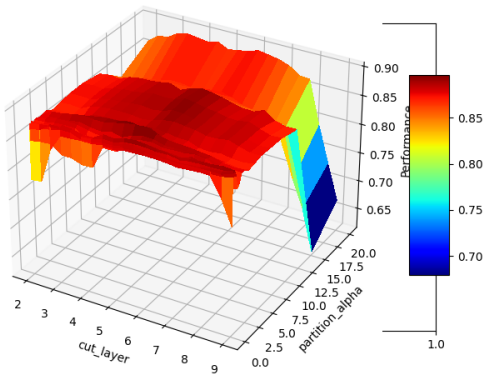
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

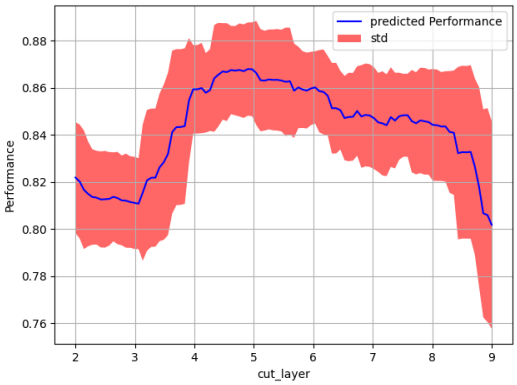


(e) Total number of clients & Partition alpha

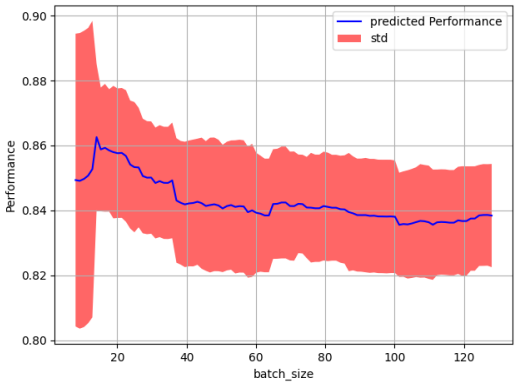


(f) Cut layer & Partition alpha

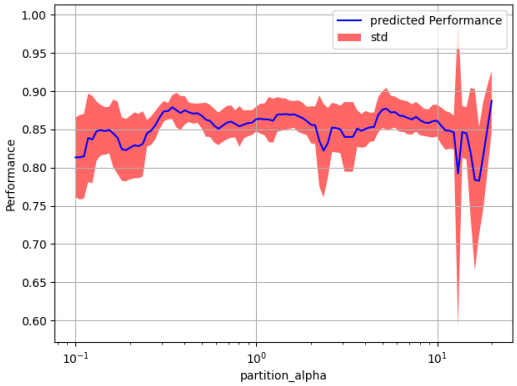
Figure 12: **Internet-Advertisements** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



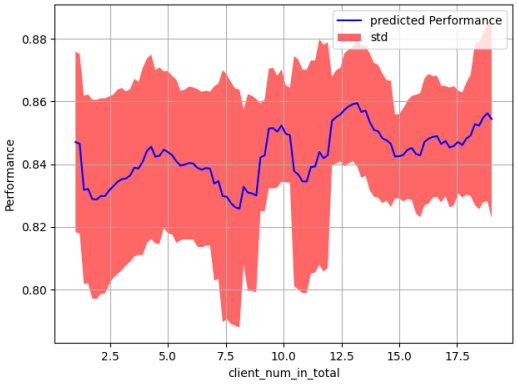
(a) Cut layer



(b) Batch size

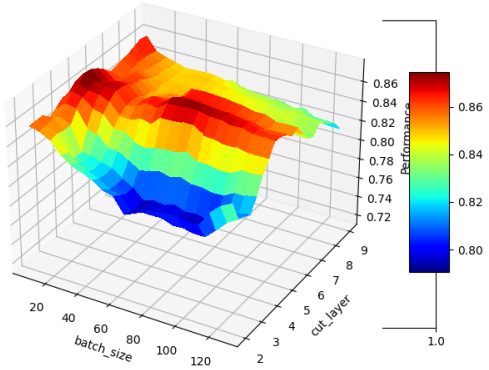


(c) Partition alpha

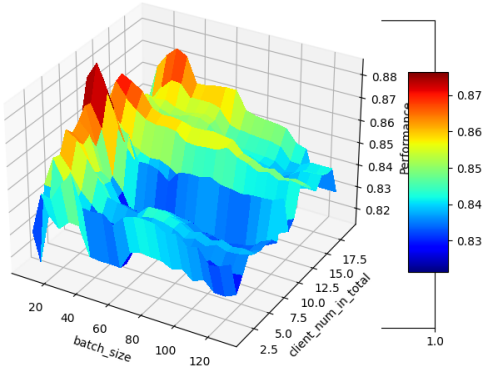


(d) Total number of clients

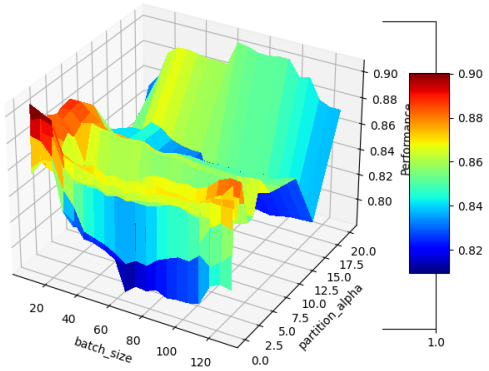
Figure 13: **Isolet** Importance of each hyperparameter



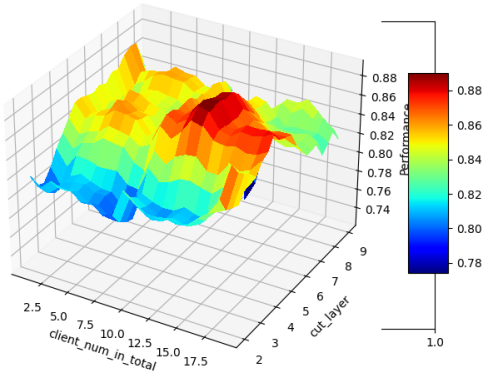
(a) Batch size & Cut layer



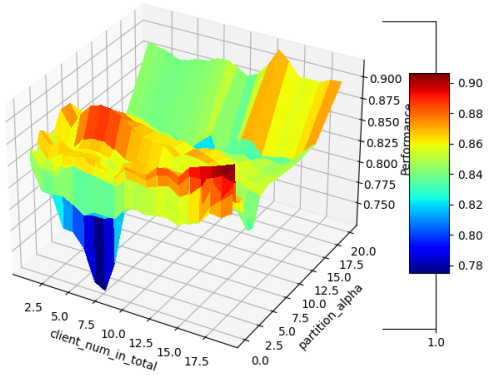
(b) Batch size & Total number of clients



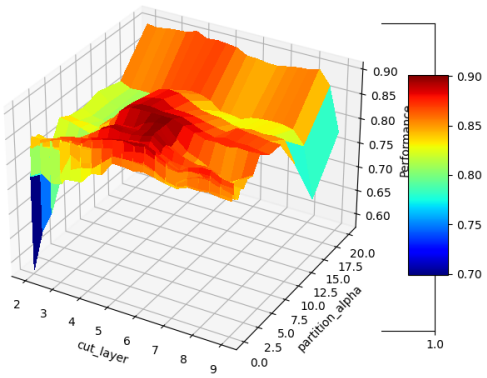
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

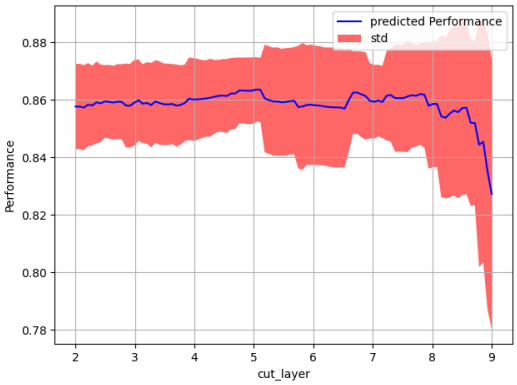


(e) Total number of clients & Partition alpha

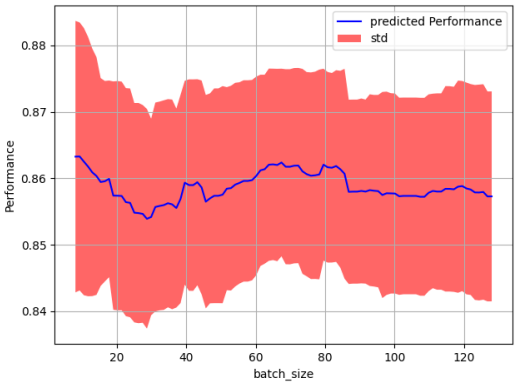


(f) Cut layer & Partition alpha

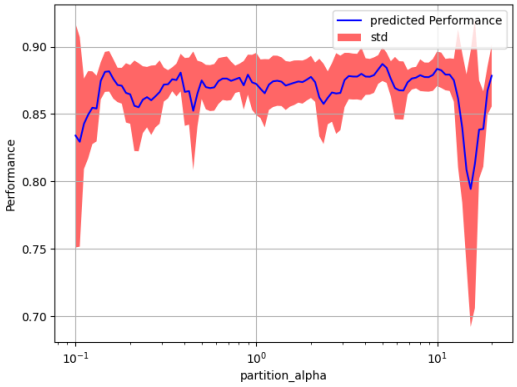
Figure 14: **Isolet** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



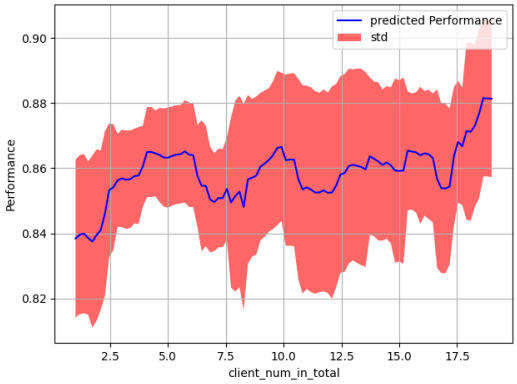
(a) Cut layer



(b) Batch size

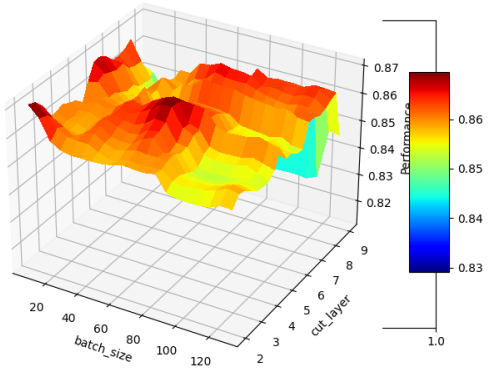


(c) Partition alpha

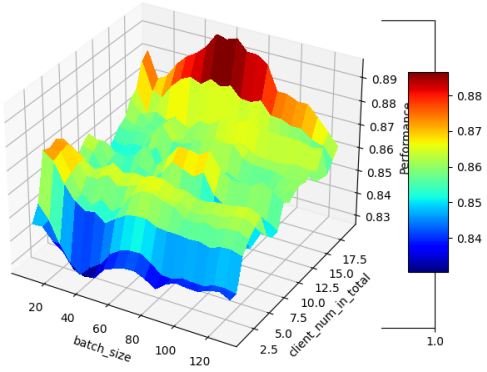


(d) Total number of clients

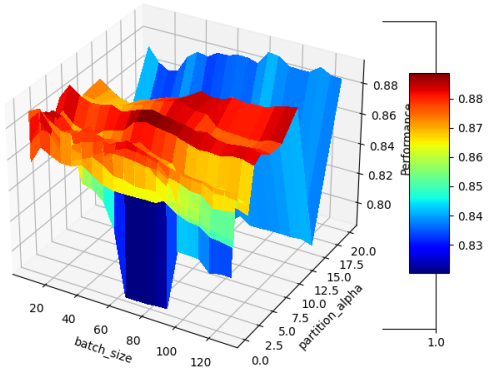
Figure 15: **HAR** Importance of each hyperparameter



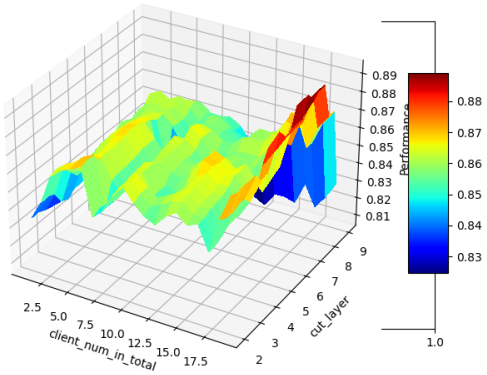
(a) Batch size & Cut layer



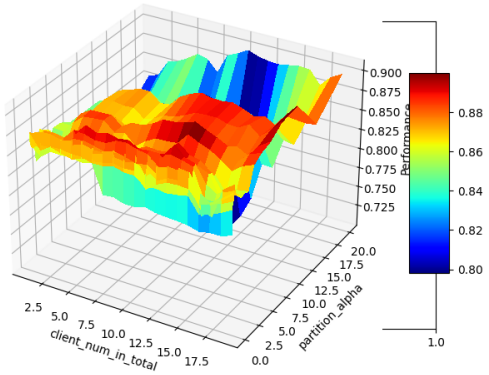
(b) Batch size & Total number of clients



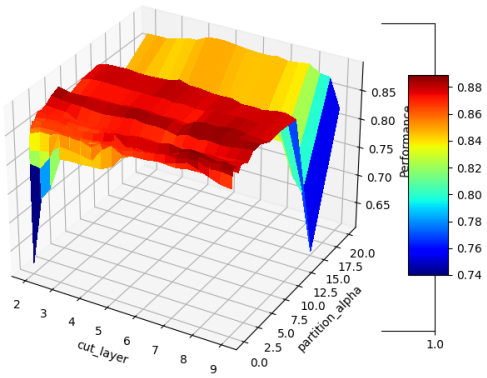
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

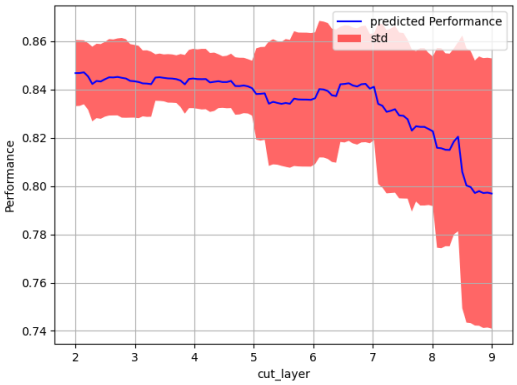


(e) Total number of clients & Partition alpha

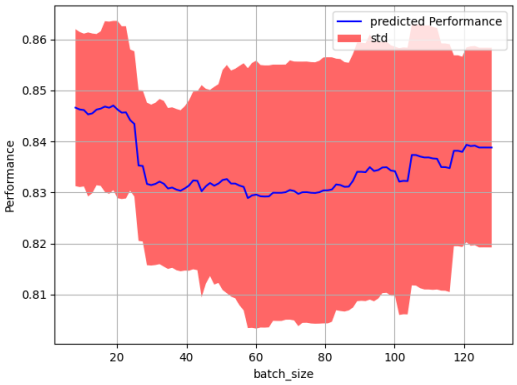


(f) Cut layer & Partition alpha

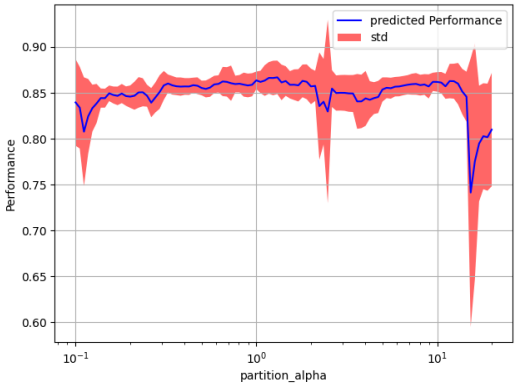
Figure 16: **HAR** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



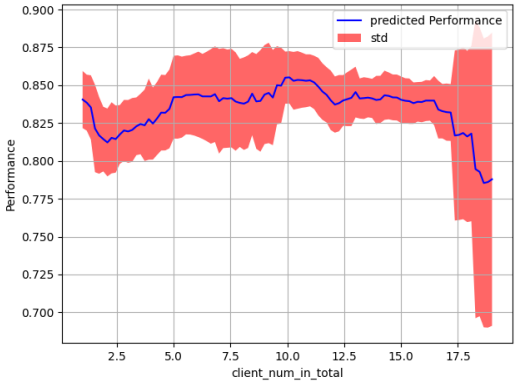
(a) Cut layer



(b) Batch size

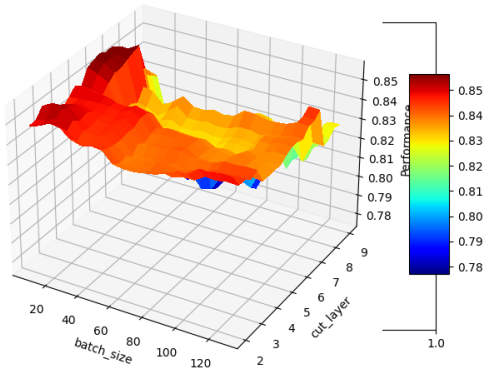


(c) Partition alpha

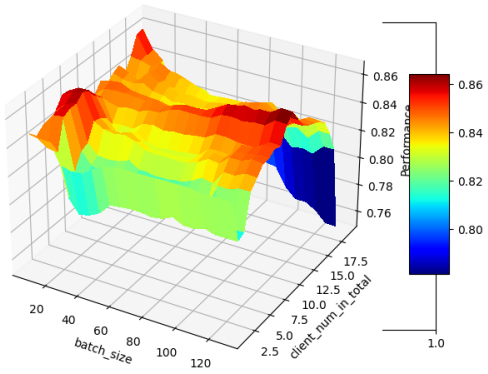


(d) Total number of clients

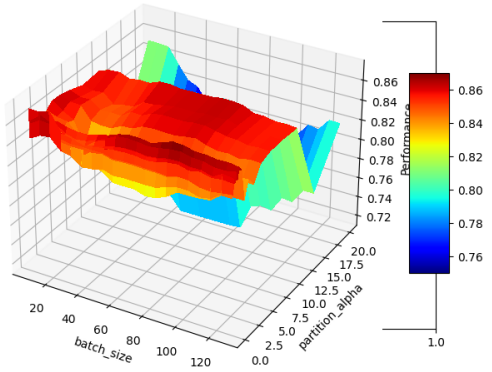
Figure 17: **DNA** Importance of each hyperparameter



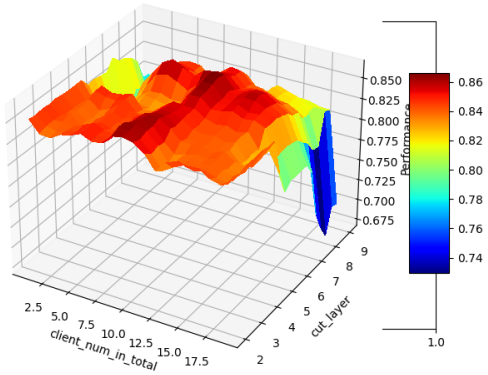
(a) Batch size & Cut layer



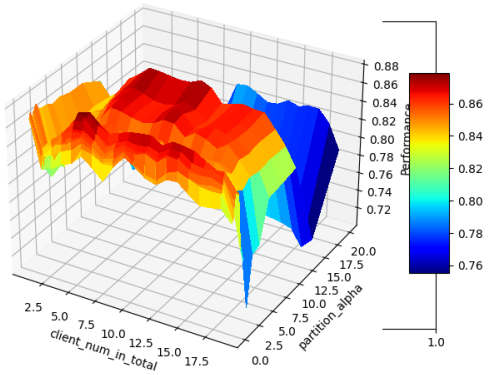
(b) Batch size & Total number of clients



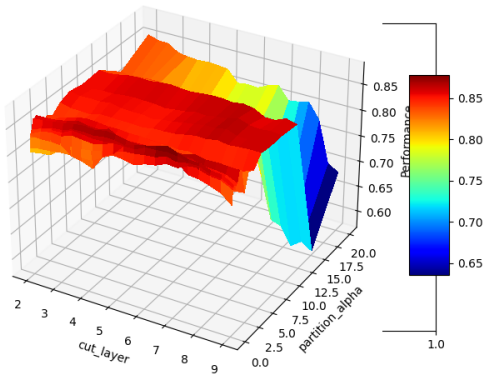
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

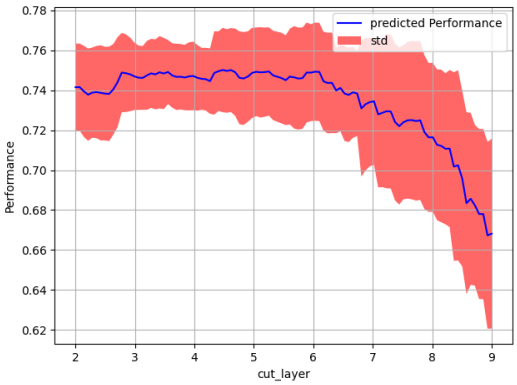


(e) Total number of clients & Partition alpha

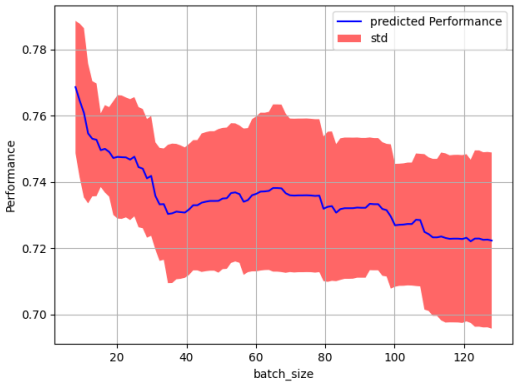


(f) Cut layer & Partition alpha

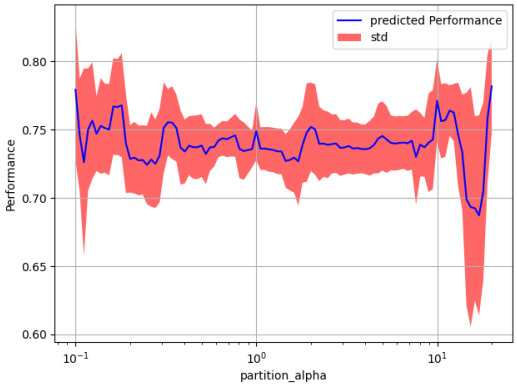
Figure 18: **DNA** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



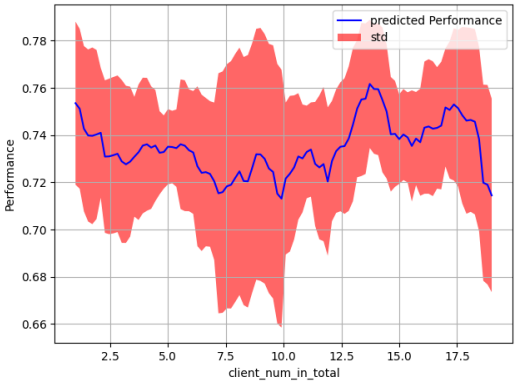
(a) Cut layer



(b) Batch size

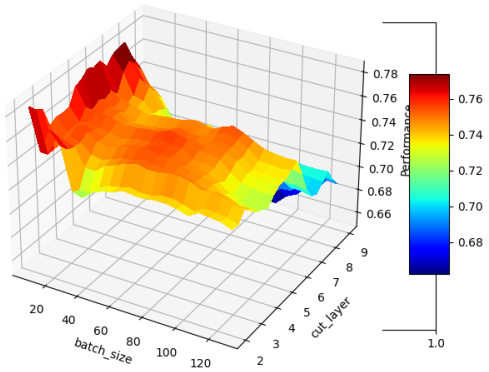


(c) Partition alpha

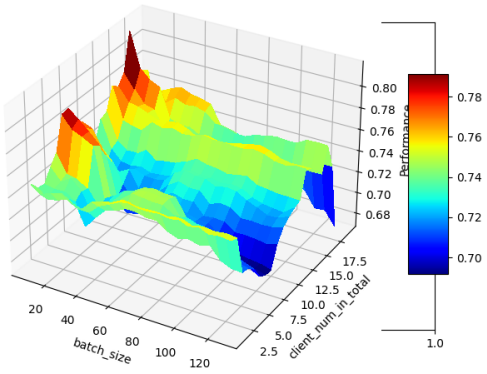


(d) Total number of clients

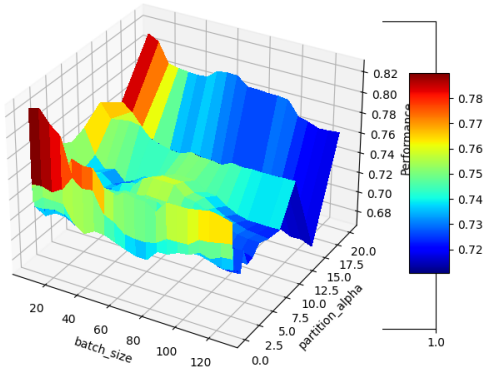
Figure 19: **Nomao** Importance of each hyperparameter



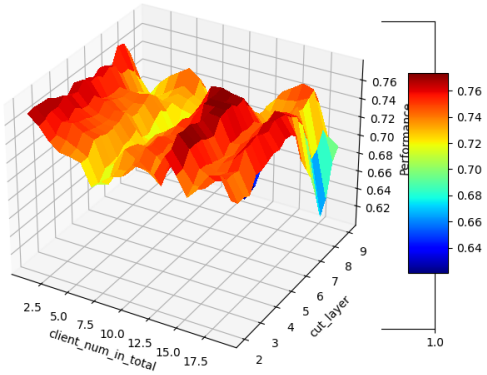
(a) Batch size & Cut layer



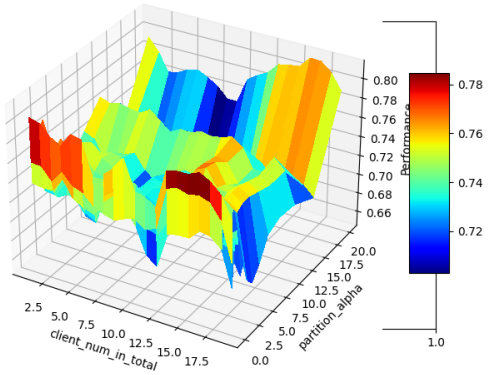
(b) Batch size & Total number of clients



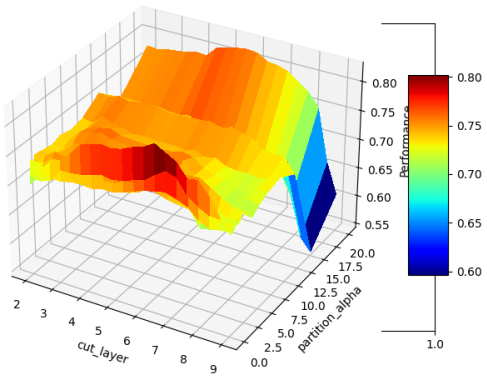
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

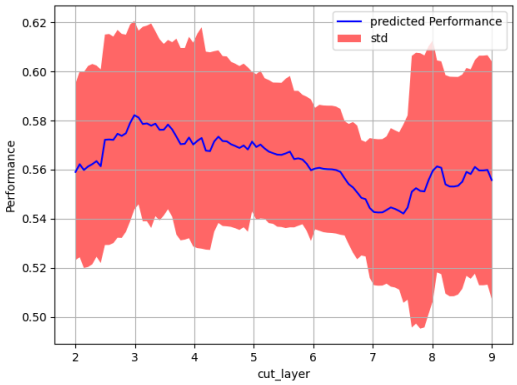


(e) Total number of clients & Partition alpha

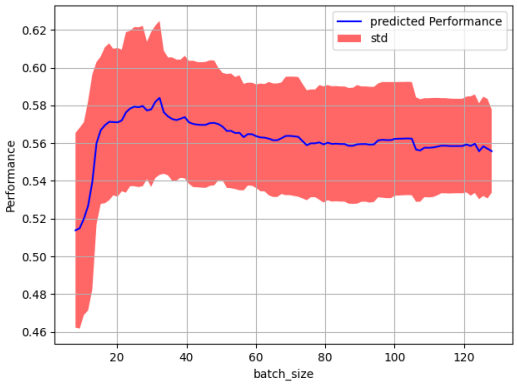


(f) Cut layer & Partition alpha

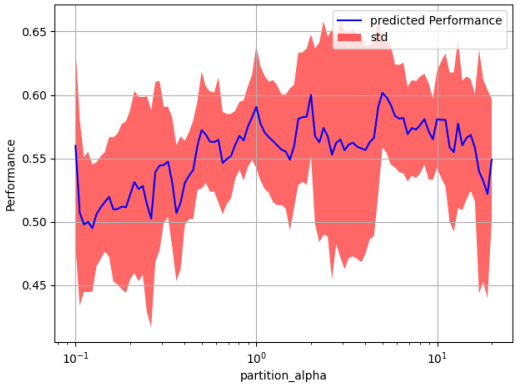
Figure 20: **Nomao** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



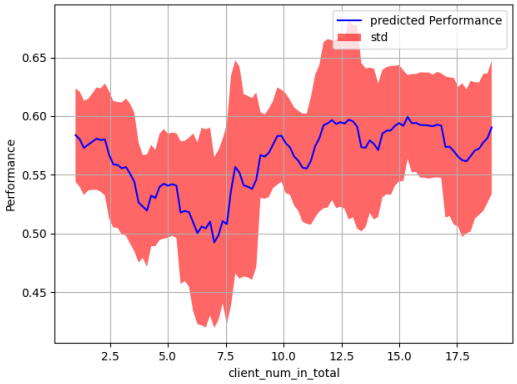
(a) Cut layer



(b) Batch size

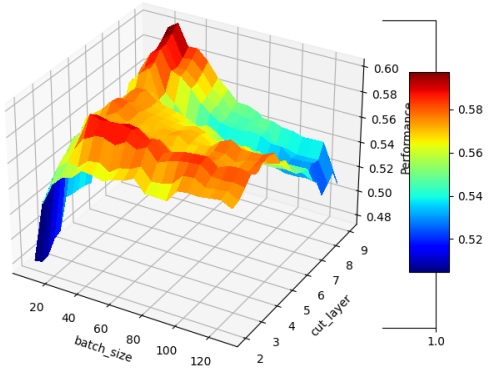


(c) Partition alpha

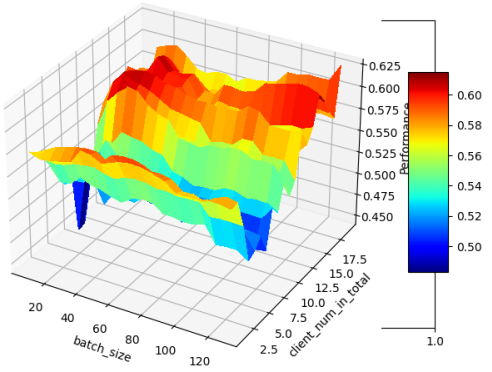


(d) Total number of clients

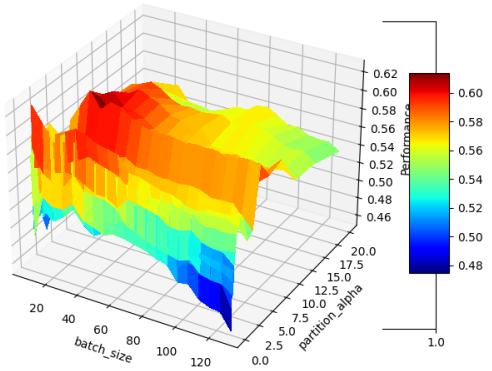
Figure 21: **Ozone-level-8hr** Importance of each hyperparameter



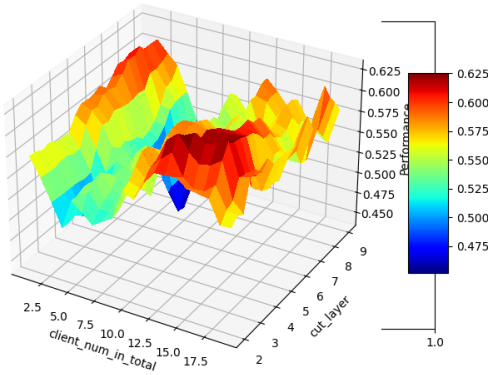
(a) Batch size & Cut layer



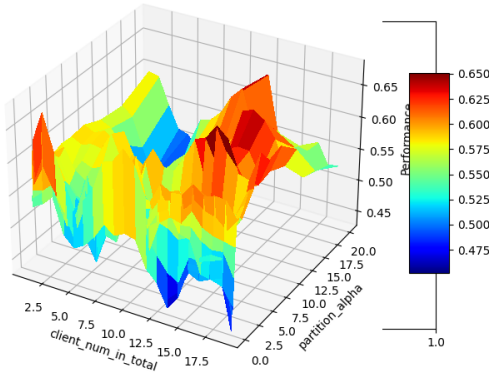
(b) Batch size & Total number of clients



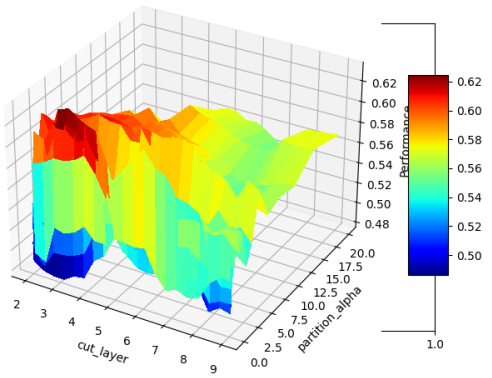
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

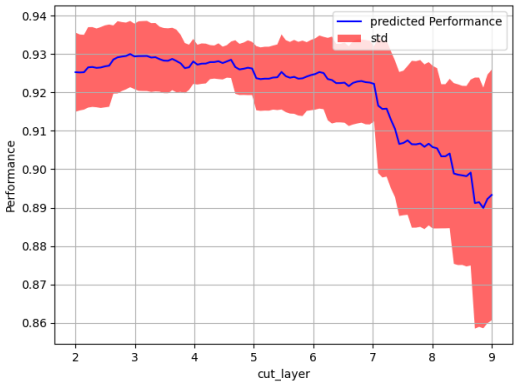


(e) Total number of clients & Partition alpha

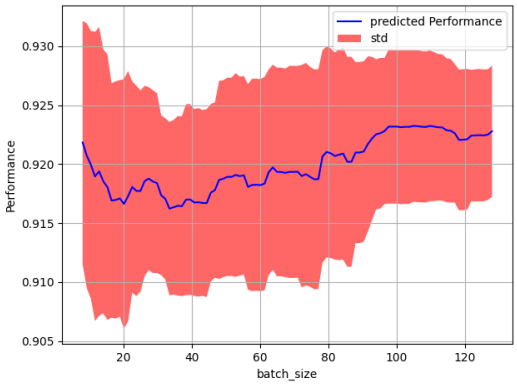


(f) Cut layer & Partition alpha

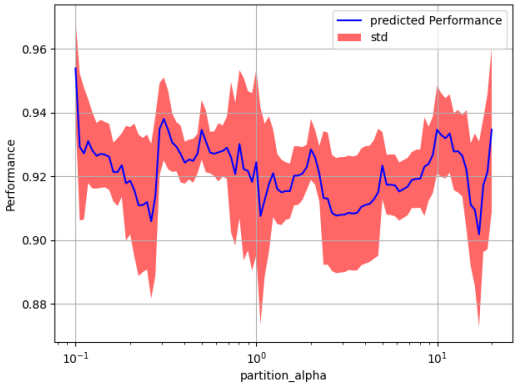
Figure 22: **Ozone-level-8hr** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



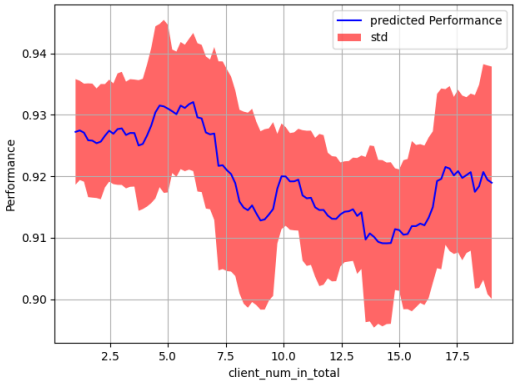
(a) Cut layer



(b) Batch size

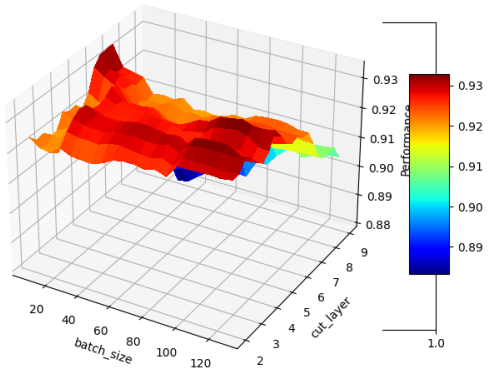


(c) Partition alpha

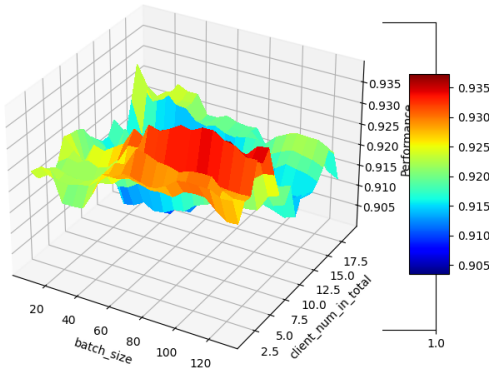


(d) Total number of clients

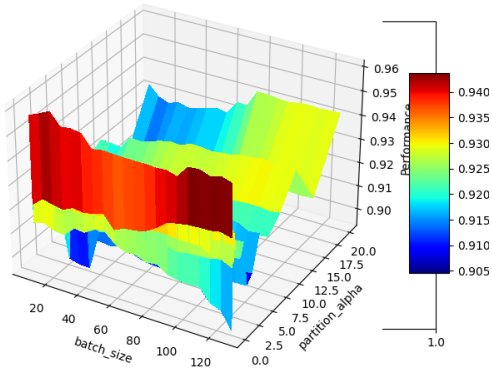
Figure 23: **Optdigits** Importance of each hyperparameter



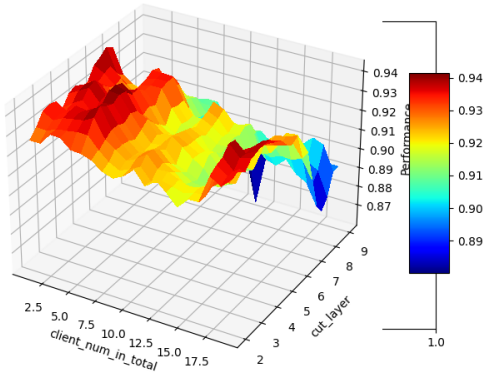
(a) Batch size & Cut layer



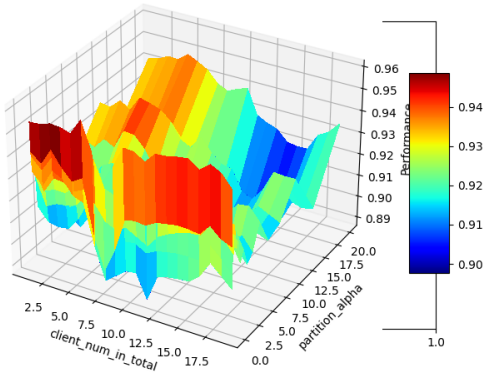
(b) Batch size & Total number of clients



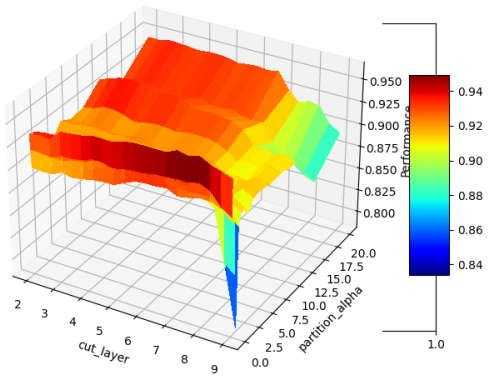
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer

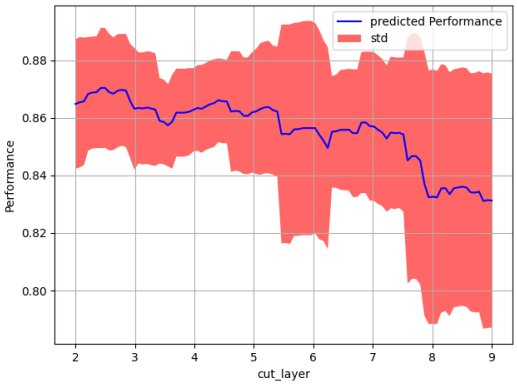


(e) Total number of clients & Partition alpha

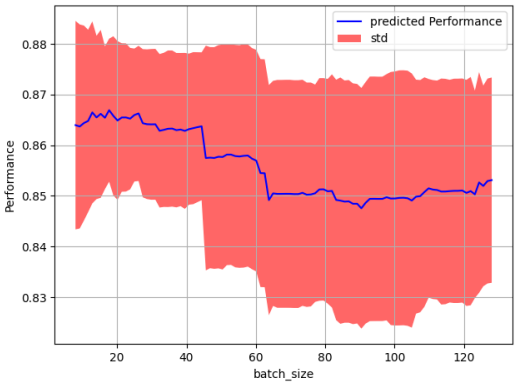


(f) Cut layer & Partition alpha

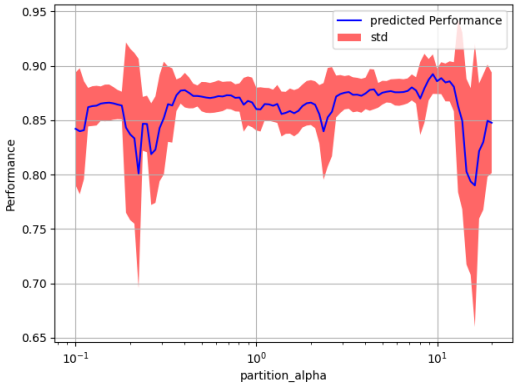
Figure 24: **Optdigits** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization



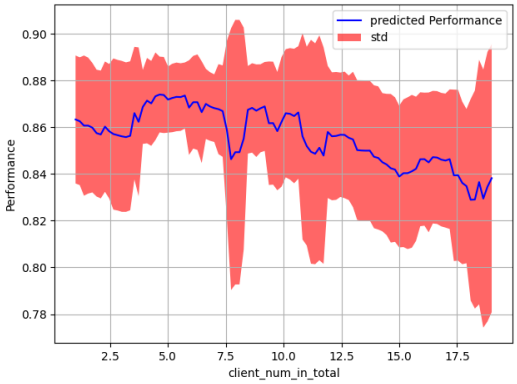
(a) Cut layer



(b) Batch size

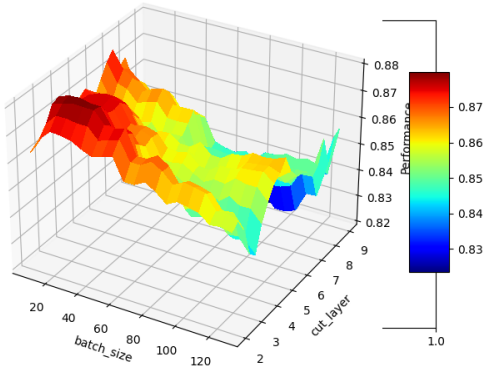


(c) Partition alpha

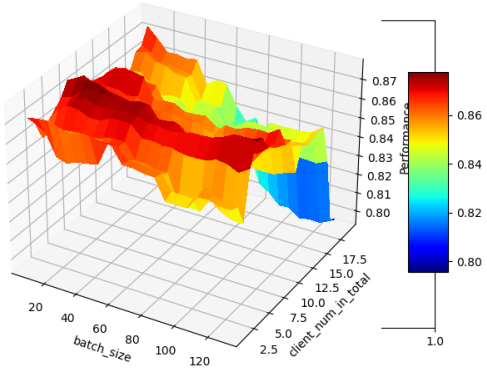


(d) Total number of clients

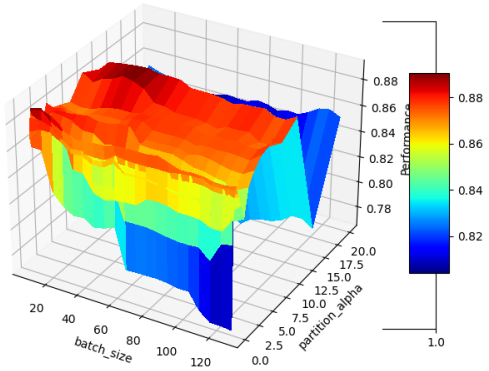
Figure 25: **Splice** Importance of each hyperparameter



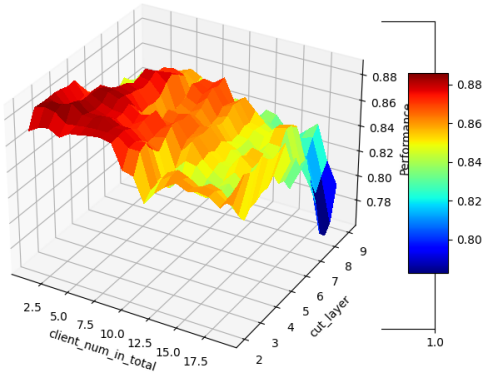
(a) Batch size & Cut layer



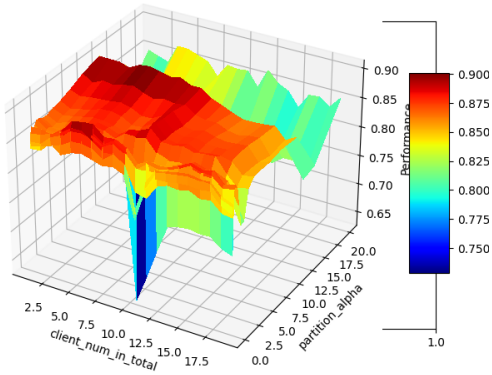
(b) Batch size & Total number of clients



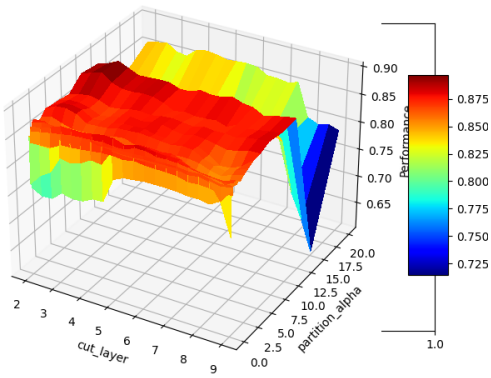
(c) Batch size & Partition alpha



(d) Total number of clients & Cut layer



(e) Total number of clients & Partition alpha



(f) Cut layer & Partition alpha

Figure 26: **Splice** Hyperparameter interaction effects on performance: Visualizing combined influence and trade-offs in model optimization

B Appendix

Link of the benchmark code on Github: https://github.com/AmrAdwan/Split-learning

B.1 Usage of ChatGPT

ChatGPT was used in the writing of this thesis in the following ways:

- ChatGPT was used to obtain a better understanding of specific subjects by seeking explanations. Alternative sources were explored when the replies were not adequate or believable.
- The tool was also used to find flaws in the created text or concepts. This includes grammatical errors, and coherence in topic discussions.