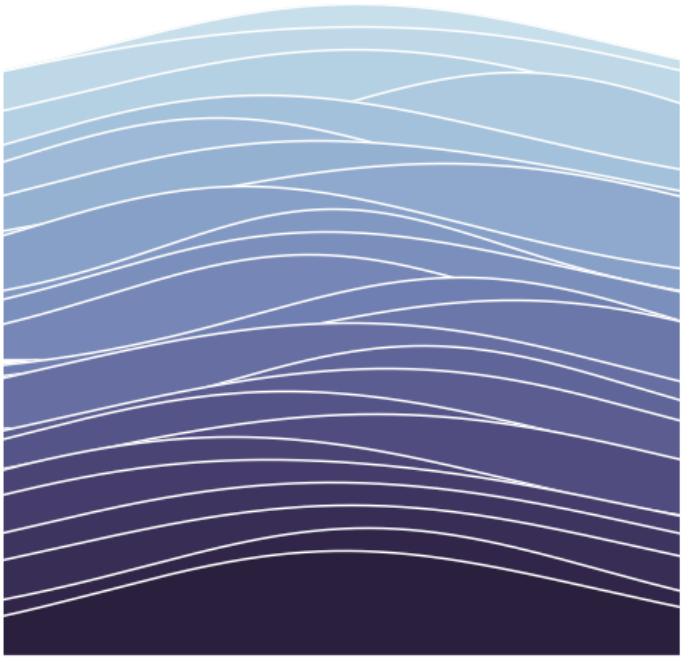


Posterior Estimation in Federated Learning

A Bayesian perspective and non-parametric approach for model aggregation and inference

OSCAR JOHANSSON & MARTIN GULLBRANDSON

Department of Electrical Engineering
Engineering Mathematics and Computational Science
Chalmers University of Technology, www.chalmers.se
Gothenburg, Sweden 2022

MASTER'S THESIS 2022

Posterior Estimation in Federated Learning

A Bayesian perspective and non-parametric approach for model aggregation and inference

Oscar Johansson
Martin Gullbrandson

CHALMERS
UNIVERSITY OF TECHNOLOGY

AI
SWEDEN

Department of Electrical Engineering
Engineering Mathematics and Computational Science
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022

Posterior Estimation in Federated Learning
A Bayesian perspective and non-parametric approach for model aggregation and inference
Oscar Johansson & Martin Gullbrandson

Supervisor:
Lennart Svensson, Professor, Electrical engineering, Chalmers University
Mattias Åkesson, Data Scientist, Scaleout Systems
Fredrik Wrede, Data Scientist, Scaleout Systems

Master's Thesis 2022
Engineering Mathematics and Computational Science
Department of Electrical Engineering
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Multiple wide kernel density estimates of random weights in an MNIST model.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Posterior Estimation in Federated Learning
A Bayesian perspective and non-parametric approach for model aggregation and inference
Oscar Johansson & Martin Gullbrandson
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Federated learning is an emerging field within machine learning that enables learning on distributed data. The concept is to train models locally on edge nodes, and globally aggregate these models across a graph of edge nodes. Doing so increases both end-user privacy and decrease communication costs, as data is not communicated. With data censored, the learning task is considerably more difficult than in the centralized case. It is further complicated as data is usually not independent and identically distributed (non-IID) among edge nodes.

This thesis explores how a Bayesian perspective may consider the inherent model variance when aggregating different locally trained models, specifically in non-IID environments. This is done by evaluating algorithms that take advantage of a probabilistic perspective, such as FedPA [1], and compare to a standard algorithm, FedAvg [2]. Further, we propose a novel lightweight approach using kernel density estimation to estimate a posterior distribution and using mean-shift for model inference. We call the algorithm federated kernel posterior (FedKP) and show that it outperforms other algorithms on bench-marking examples, while requiring no extra information and without any hyperparameter-optimization.

Keywords: federated learning, kernel density estimation, Bayesian, posterior estimation, heterogeneity, non-IID, mean-shift.

Acknowledgements

We like to acknowledge our partners for making this thesis possible. These include our supervisors and professors at Chalmers, Scaleout Systems for providing expertise within federated learning and AI Sweden for providing computational hardware and premises to conduct the thesis within. Specifically, we would like to acknowledge Mattias Åkesson and Fredrik Wrede from Scaleout, Erik Svensson and Kim Henriksson at AI Sweden, and our supervisor Lennart Svensson at Chalmers for helping us with inspiration and ideas for our thesis.

We would also like to acknowledge the help we have gotten from Google when implementing their algorithms. Specifically, Louise Anderson-Conway for putting us in contact with the right people and Zachary Charles for answering our questions.

Lastly, as this finishes up our years in school, we would also like to acknowledge all the teachers and staff who have helped us through the years. Without your teaching, this thesis would not be possible.

Oscar Johansson & Martin Gullbrandson, Gothenburg, June 8, 2022.

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AI	Artificial Intelligence
BNN	Bayesian Neural Network
EU	European Union
FedAG	Federated Average Gaussian
FedAvg	Federated Average
FedBE	Federated Bayesian Ensemble
FedKP	Federated Kernel Posterior
FedPA	Federated Posterior Averaging
FL	Federated Learning
GDPR	General Data Protection Regulation
IQR	Interquartile Range
KDE	Kernel Density Estimation
LMC	Langevin Monte Carlo
MAP	Maximum a Posteriori
MCMC	Markov Chain Monte Carlo
MISE	Mean Integration Square Error
ML	Machine Learning
MLE	Maximum Likelihood Estimate
PDF	Probability Density Function
ROT	Rule of Thumb
SGLD	Stochastic Gradient Langevin Dynamic

Nomenclature

Below is the nomenclature of indices, distributions, sets, functions, and variables that have been used throughout this thesis.

Indices

i	Indices for clients
t	Index for communication round

Distributions

$\mathcal{P}_i$	Distribution of data for client i
$\mathcal{Q}$	Distribution of clients in the federated problem

Sets

$\mathcal{D}$	Set of training data
$\mathcal{D}_i$	Set of training data for client i
$\mathcal{S}$	A sample of clients drawn from from Q

Functions

$\mathcal{H}$	Loss function
K	Kernel function
K_h	Kernel function with respect to bandwidth h

Variables

θ	Model parameters
x	Model input data
y	Model output
h	Bandwidth
N	Number of clients participating in communication round

List of Figures

List of Tables

Contents

1 | Introduction

With the rise of AI, machine learning is continuously becoming integrated into various domains and applications [3]. These applications rely on information and henceforth an increased need for data [4]. It is most likely one of the driving factors behind the saying "data is the new oil". It is not by chance that many of today's most highly valued companies, Alphabet, Apple, or Amazon to name a few, are also leading in data collection. With the focus on data collection from companies, end-user data privacy has become increasingly important both for users and institutions. As of 2018, GDPR was introduced throughout the EU [5] which has affected how data can be communicated and stored.

Perhaps as a response to the potential privacy issues of data collection, conducting machine learning without centrally collecting data is attracting attention. Training models in a distributed setting, referred to as federated learning, is a meta-algorithm for machine learning that was introduced in 2017 [6]. The main idea is to decouple the training step of machine learning from the data collection by training models locally and then aggregating these models to a single global model. This way data is kept local to the user, and only the model parameters are communicated. The difficulty in such a task is how to infer an optimal global model from only knowing the local models. This problem becomes incresingly challenging when users possess very different information and data is heterogeneously distributed among the users.

Aggregation in federated learning is essential, and guaranteeing good model convergence is difficult compared to centralized approaches. The most common solution is to average the local model parameters, commonly referred to as FedAvg [2]. FedAvg is guaranteed to converge [7, 8] and usually result in satisfactory performance. When data is distributed differently between clients, determining the optimal model is increasingly difficult, making it an active research field [6]. Recently, there has been a growing interest in aggregation algorithms that does not discard the inherent variance obtained from collecting multiple samples of locally trained model [1, 9, 10].

To account for the inherent variance it is reasonable to assume that each parameter behaves as a distribution, where each local weight is a sample of that distribution. This new *probabilistic viewpoint* of the federated learning problem potentially opens up multiple new ways of studying federated learning; how does the distributions change when data is heterogeneous, and how can an optimal model be obtained? Drawing from previous works [1, 9, 10], we introduce a new algorithm that explores how global parameter distributions can be leveraged to improve model aggregation.

1.1 Objective and Goal

The purpose of this thesis is to provide further insight into the probabilistic perspective in a federated learning setting. Primarily in the context of heterogeneous data. There is also an aspect of how such a perspective can be used to optimize aggregation and convergence in a heterogeneous setting. The purpose is as such twofold:

I We wish understand how modern federated learning algorithms that leverage a probabilistic perspective perform in a heterogeneous scenario.

II We want to explore how a probabilistic perspective may optimize model inference and convergence when aggregating local models.

1.2 Scope

There are many variants of federated learning, described in more detail in Chapter 2, but this thesis focuses on how model aggregation and convergence behave when data is heterogeneously distributed. The scope is to study the aggregation step in a federated setting where multiple edge nodes communicate with a centralized server. We hence study a simulated federated setting where a fixed number, sub-sample, of clients participate in each communication round.

Within a centralized federated setting, we will explore how a probabilistic perspective can be used for aggregation. Specifically, algorithms that have a theoretical foundation in Bayesian analysis is compared to the standard algorithm FedAvg with regards to varying degrees of client heterogeneity.

The thesis aim to compare algorithms by testing them on common bench-marking tasks for image recognition: MNIST, EMNIST-62 and CIFAR-100. Different scenarios will be simulated by introducing either a label skew between clients or adverse sampling of clients containing a feature skew.

1.3 Contributions

The main contribution of this thesis is the introduction of a novel federated learning algorithm that show promising performance on a selection of bench-marking datasets. Our algorithm is built on a Bayesian perspective of federated learning and leverage a non-parametric approach to estimate the posterior distribution. We have chosen to call the new algorithm *Federated Kernel Posterior*, or FedKP, since it uses a Kernel density for inference at the server.

The thesis further contributes by expanding the knowledge of the performance and stability of existing algorithms. Specifically, we implement the algorithm FedPA in PyTorch

and illustrate that the algorithm is unstable when data is distributed according to an extreme label skew.

1.4 Thesis Outline

In Chapter 2 we present the theoretical background of the thesis. We will as such develop a deeper understanding of the federated learning problem and different approaches to solving it. These solutions describe multiple algorithms that try to leverage a probabilistic perspective. Finally, we explore the underlying theory of our solution, being kernel density and mean shift.

Chapter 3 explores the behavior of model parameter distributions that emerge when training a federated model. We reason about how one may exploit this for our algorithm.

In Chapter 4 we define several experiments that we use to measure how the algorithms perform. Chapter 5 presents the results of previous defined experiments and, Chapter 6 analyses the results.

Finally, we conclude the thesis in chapter 7 where we present our major findings and suggest possible directions of future work.

2 | Theory

This chapter aims to present fundamentals, related work and techniques used throughout the thesis. We introduce the main concepts of machine learning and Bayesian statistics relevant to the thesis. Then, federated learning is formally defined, and the federated learning problem is defined both from a classical and probabilistic standpoint. We explore related works and algorithms that leverage a probabilistic perspective in federated learning. Latent Dirichlet allocation is introduced as a way of creating decentralized dataset. Finally, we introduce kernel density estimation and the mean shift algorithm used in the design of FedKP.

2.1 Machine Learning

Machine learning means many things depending on whom you ask. In its most basic form, it is learning an estimator of some arbitrary relation [11]. This relation is traditionally a relationships between data. A classical machine learning workflow follows a procedure of data collection, designing a model, and learning. Learning a model usually means adjusting the model to fit the data, and at some instance of this training the model is said to have converged as additional data provide marginal performance improvement to the model. Below we will elaborate on each of these steps and how to evaluate the performance of a model.

2.1.1 Data Collection

Any machine learning task requires data for the learning process. As alluded to earlier this is one of the reasons why companies such as Alphabet or Meta, that collect data are pioneers within the AI field. Most common machine learning algorithms use supervised learning, which requires each sample to be labeled with the correct class or value. Labeling, or annotation, is an expensive and repetitive task that could prove a bottleneck for any machine learning project. Other alternatives exist, such as semi-supervised or unsupervised learning that does not require as much or any annotated data. Arguably, supervised learning is the most common and therefore a large part of the typical data pipeline consists of annotation. Additionally, in this thesis, all models used are supervised.

2.1.2 Model Design

Depending on the specific task to solve multiple different machine learning architectures may be used. A common machine learning architecture is a fully connected neural network, which essentially works by stacking multiple linear layers with non-linear activation functions in between. A layer could be mathematically described as the transformation

$$\mathbf{y} = f(W\mathbf{x} + b), \tag{2.1}$$

where the weights W define a linear transformation of the input $\mathbf{x}$ and b is a constant bias term, both subject to an activation function f. To create a fully connected network, one stack multiple of these transformations described in (2.1).This fundamental block may be modified and combined to create more complex architectures such as convolutional neural networks (CNN) [12] or residual neural networks (ResNet) [13].

There are many more details and nuances to model architecture. However, for our purpose, the key insight is that neural networks, in theory, are multiple linear transformations combined with activation functions. As such, a neural network is uniquely described by the parameters, usually referred to as weights, of the W matrix for each layer and the activation function.

2.1.3 Learning

With a given model design and data, how do we set the weights W such that the model can describe the underlying relationship between $\mathbf{y}$ and $\mathbf{x}$? In abstract terms, this requires learning the model by providing it with a sensible amount of examples to map the relation. In detail one usually use *gradient descent* techniques [14] to update model weights. The weights are updated proportional to a *learning rate* times a "direction" or *gradient*.

The gradient is usually derived from an objective, or so-called *loss function*. The choice of loss function differs depending on the problem to be solved, but as this thesis will solve classification tasks the *categorical-cross-entropy loss* will be of interest. The categorical-cross-entropy loss is defined as

$$\mathcal{H} = \sum_{i \in \mathcal{C}} -y_i \log(\hat{p}_i), \tag{2.2}$$

where $\mathcal{H}$ denotes the loss, $\mathcal{C}$ the set of possible labels, y_i a Boolean whether i is the true label or not, and $\hat{p}_i$ the estimated probability of it being label i. Given the loss, the gradient can be calculated through *back-propagation*, and given this gradient one uses a gradient descent algorithm to optimize the model performance.

A common gradient descent algorithm is *stochastic gradient descent* (SGD). Instead of using all the data, SGD uses a sub-sample of the data each iteration and therefore gives the gradients calculated an element of randomness. This randomness is practical as it makes the model update randomly, exploring the loss landscape more than the traditional deepest descent.

There are many other algorithms for learning. Famously there are Adam [15], AdaGrad [16], and many other optimizers that leverage different techniques for updating the weights given loss gradients. A common technique used in many such algorithms is *momentum* [17] which adds a momentum term to the resulting gradient as

$$\Delta\theta_t = \eta\nabla\theta_t + m\Delta\theta_{t-1}, \tag{2.3}$$

with η being the learning rate, m the momentum coefficient, and $\Delta\theta_t$ being the resulting gradient. Momentum add inertia to the resulting gradient, resulting in a decaying moving average of the gradients calculated. The resulting gradients are therefore less sensitive to exceptions or "side-steps".

2.1.4 Evaluation

The evaluation of the training is usually summarized into key performance metrics. The most used metric is *accuracy* which in a binary classification setting is defined as

$$\textit{Accuracy} = \frac{TP + TN}{TP + TN + FP + TN}$$

where TP and TN are *true positive* and *true negative* being the correctly predicted classes. FP and FN are instead the *false positive* and *false negative*, in other words, the incorrectly classified samples. In a multi-class setting, one can calculate the accuracy by calculating the accuracy for each class compared to the rest and average the results.

Depending on the task, other metrics may be used. Other metrics are *recall*, being defined in the binary classification setting as

$$\textit{Recall} = \frac{TP}{TP + FN}.$$

Recall is a metric describing how many the samples of a specific class the algorithm got right, in other words, how many of the true labels it was able to recall. Another metric is *precision*, commonly defined as

$$\textit{Precision} = \frac{TP}{TP + FP}.$$

Precision reflects how many of the samples predicted to be a class actually belong to that class, or how precise the algorithm is. Recall and precision thus complement each other well in the sense that is it easy to achieve 100% recall if you predict the same class all the time, but then the precision will be poor. Similarly, it is easy to achieve good precision by rarely predicting a specific class but again the recall will be poor. This thesis uses the aforementioned metrics for evaluation but many metrics, not only recall and precision, complement each other to give richer understanding of the model performance.

2.2 Bayesian Fundamentals

Within the statistical community, there are two main opposing theories: frequentist and Bayesian statistics. According to the frequentist school of thought, one assumes an ab-

solute truth and that the true value of a random variable can become estimated by observation. In contrast, a Bayesian perspective sees the truth as asymptotic. There is no true value of a random variable, only evidence of how the variable is distributed. This slight difference lead to drastically different paradigms of statistics.

In Bayesian statistics, we can learn more about the random variable by observing samples of it. This learning procedure follows from the Bayes theorem. Given a random variable θ and an observed set of data $\mathcal{D}$, the Bayes theorem can be written as

$$\mathcal{P}(\theta|\mathcal{D}) \propto p(\theta)\mathcal{L}(\mathcal{D}|\theta). \tag{2.4}$$

Equation (2.4) states that the *posterior* distribution of θ is proportional to the *prior* distribution times the *likelihood* $\mathcal{L}$ of the observed data $\mathcal{D}$. The prior reflects the information we already know about θ, and the likelihood is a description of how likely it is to observe the data $\mathcal{D}$ given θ. Multiplying these two distributions one weigh the prior known knowledge of θ towards new evidence of its likelihood.

Due to the form of the prior and likelihood, a closed-form analytical expression of the posterior is often hard to evaluate. Therefore, a common approach is to estimate the posterior using *Markov-Chain Monte-Carlo* (MCMC) sampling [18]. Since we may not analytically solve the posterior distribution, by sampling from it we can estimate it empirically. As MCMC sampling takes time before the sampling reflects the true posterior, one often only use samples after a so-called *burn-in*. As convergence is stochastic one often uses multiple runs or *chains* to parallelize and more quickly explore the posterior. The fundamental idea is that the empirical distribution of samples in each chain will converge towards a stationary distribution, i.e., the posterior distribution [19]. Observe that the parallel use of multiple chains is potentially practical in decentralized settings such as federated learning.

2.2.1 Bayesian Neural Networks

Equipped with new knowledge about Bayesian statistics it is natural to wonder how the new tool may be used for neural networks. Instead of viewing each weight in a network as a point estimate, we may view it as a distribution given the training data, $\mathcal{P}(\theta|\mathcal{D})$, illustrated in figure 2.1. In Bayesian terms, each parameter is a posterior distribution given the data it is trained on. Using a Bayesian approach to model parameters comes with multiple advantages such as natural uncertainty of predictions. The prior can also be used as regularization to mitigate overfitting [20].

Predictions in such a network is defined by the predictive distribution $P(y|\theta, \mathbf{x})$, which is usually intractable [20]. In practice, one evaluate the model on a constant input, $\mathbf{x}$, using multiple different samples of θ as model parameters. Sampling results in an ensemble of networks and yield an empirical sample of $P(y|\theta, \mathbf{x})$. A traditional neural network could be described by using the weights corresponding to the *maximum a posteriori* (MAP) estimate given the posterior. Note, using an uniform prior, the MAP and *maximum likelihood estimate* (MLE) are equivalent. Naturally, the Bayesian neural network provide a richer understanding of the variance and uncertainty related to the problem compared to traditional neural networks.

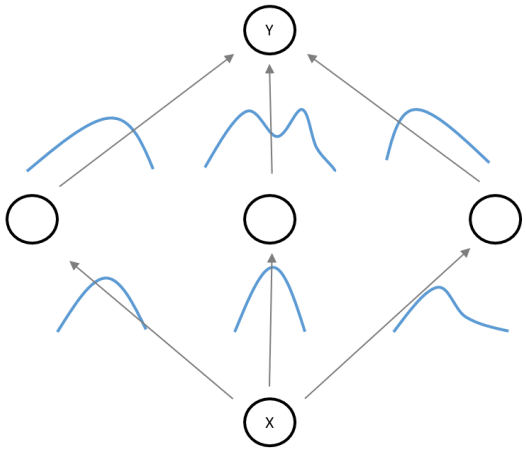


Figure 2.1: Schematic illustration of a Bayesian neural network, where weights is a posterior distribution given the data: $\mathcal{P}(\theta|\mathcal{D})$. As such each value produced is a distribution rather than a point estimate.

2.3 Federated Learning

Federated learning was introduced in 2017 by the introduction of the *Federated Average* (FedAvg) algorithm [2]. The key feature of the federated paradigm is that data is no longer collected from the user, usually referred to as edge or client. Instead, the federated approach allow clients who collect the data to train their model locally and only communicate the resulting model instead of the data. Traditionally, client models aggregate into a global model, available for all clients to use. The privacy preserving property and communication efficiency enable federated learning to be applied in many different real-world scenarios. One such example is the keyboard predictions used by Google [21]. Another area where federated learning has been applied with success, at least theoretically, is predictions of battery range and demand for electric vehicles [22, 23]. An illustrative example of federated learning is shown in figure 2.2.

Though federated learning is decentralized, two key categories within the field are centralized and decentralized, commonly referred to as peer-to-peer. The centralized approach concerns having a centralized computational node responsible for aggregating and combining client models [6]. The peer-to-peer setting, is the scenario of removing the centralized computational node. Then clients typically interact through graph structures [24]. One key advantage when working peer-to-peer is that each client control its data and can use

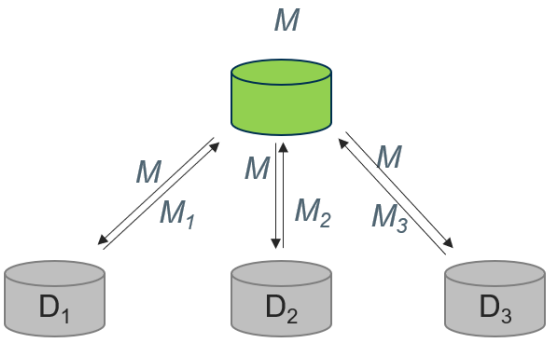


Figure 2.2: Illustration of federated learning with 3 clients, each containing local data D_i and model M_i for $i = 1, 2, 3$. The server computes a global model M.

that information when aggregating models.

Another set of classes within federated learning are cross-silo and cross-device. These categories are vaguer but generally concern the amount of nodes and their computational capacity. Cross-silo is often associated with large institutes (hospitals, banks) with a large data capacity cooperating over a federated setting [6]. In contrast, smaller computational nodes (cellphones, IoT devices) define the cross-device setting of federated learning [25]. Besides the difference in computational capacity between the paradigms, the major difference is that of connectivity. In a cross-device setting, there are large uncertainties about what and how many devices will contribute in each round of aggregation. A cross-device setting therefore usually operates with a subset of all possible devices. This comes naturally in an applied setting where only a subset of devices can participate in training due to availability and data generation.

Whether one uses federated learning in a centralized, decentralized, cross-device, or cross-silo setting the key idea remains. In the federated paradigm of machine learning one decouples the task of learning and the collection of data. This thesis concerns the aggregation of models using a subset of clients in a centralized way.

2.3.1 The Federated Learning Problem

The problem in the federated setting is how to guarantee good performance of the global model. As performance is commonly measured by the objective function, i.e., the loss, it is sensible that the goal would be to minimize the loss across all clients. Given a loss function $\mathcal{H}$, parameters θ and N clients with local data $\mathcal{D}_i$ we have the global loss, or sum of losses defined as

$$\mathcal{H}(\theta|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{H}(\theta|\mathcal{D}_i). \tag{2.5}$$

We observe that if all client data fractions are identical, optimization of the global loss is the same as optimizing the local loss, further if the data distributions are identical the expected value of optimum the loss is the same for the global and local problem. In such a case it seems intuitively reasonable to find an optimum by aggregating local models. However, when data distributions are not identical it is unclear how to guarantee good performance when aggregating local models, something this thesis explores.

2.3.2 Data Distributions

We can view the sampling of the data as a two-step process described as

$$\mathcal{D}_i \sim \mathcal{P}_i \tag{2.6}$$

$$i \sim \mathcal{Q} \tag{2.7}$$

where $\mathcal{Q}$ is a distribution over all available clients and $\mathcal{P}_i$ is a distribution of the data for any given client i. As previously mentioned, it is practical to assume that all underlying

distributions $\mathcal{P}_i$ are identical. However, this is not usually the case when algorithms are applied outside an academic setting. Given the sampling in (2.6), one has non-IID distributed data in the case where $\mathcal{P}_i \neq \mathcal{P}_j$ for $i \neq j$. This type of non-IID data is referred to as *heterogeneous data* or in this specific case as a label skew.

One should note that this is not the only way for data to be non-IID. The distributions $\mathcal{Q}$ and $\mathcal{P}_i$ can vary over time and cause the sampling to be non-IID [6]. Further, different clients may have different relations in the data which can be referred to as a feature skew. An example of a concept skew would be where one set of clients only observer black dogs while an other set observe only brown dogs. There are in other words multiple ways that data in a federated setting may be non-IID.

How to deal with heterogeneous data is a key aspect of applied federated learning and one way to separate federated learning from decentralized computation. The problem is much harder, and performance decay as the amount of non-IID behavior increases [8]. Hence, an active area of research to find a more robust technique for federated learning with respect to heterogeneous data.

2.3.3 Model Aggregation

The most common way to aggregate the local models into one centralized model is *federated averaging*, FedAvg, which was introduced in 2017 [2] and described in more detail as algorithm 1. The technique is motivated by viewing federated learning as a type of SGD optimization where multiple gradient steps are performed locally before averaging.

FedAvg is widely used due to its simplicity with relatively good results. It has been shown that FedAvg converges with a decreasing learning rate scheme and that the speed of convergence correlate to the number of locally trained epochs [2, 8]. But there are aspects to critique with FedAvg which have led to modifications and many other algorithms [26, 27]. In general, simplicity is practical but often not optimal.

With federated learning requiring multiple rounds for convergence, optimization is not only done during aggregation. Instead, one may optimize model convergence across multiple *communication rounds* between server and clients. Recent research suggests that FedOpt could present a general framework for federated algorithms [28]. FedOpt, algorithm 2, notes that the optimization performed on the client and server are separate from the general federated framework. The crucial part is that each client starts from the same point and calculates a gradient. It is therefore possible to abstract the server and client process into two different, independent optimization problems. FedAvg is thus a special case of FedOpt where the client optimization is performed with stochastic gradient descent and the server optimization is done by weighted averages. Additionally, FedOpt is often associated with using training optimization techniques such as SGD or Adam on the server-side, which will be done in this thesis.

Algorithm 1 FedAvg

input θ^0
for t=0,1,2,... **do**
  Sample subset S of clients $i \sim Q$.
  $\theta_i^t = \theta^t, \forall i \in S$
  for $i \in S$ **do**
    $\hat{\theta}_i^t \leftarrow \text{SGD}(\theta_i^t)$
  end for
  $\theta^{t+1} \leftarrow \frac{1}{\sum_{i \in S} |\mathcal{D}_i|} \sum_{i \in S} |\mathcal{D}_i| \hat{\theta}_i^t$
end for

Algorithm 2 FedOpt

input θ^0, ClientOpt, ServerOpt
for t=0,1,2,... **do**
  Sample subset S of clients $i \sim Q$.
  $\theta_i^t = \theta^t, \forall i \in S$
  for $i \in S$ **do**
    $\Delta_i^t \leftarrow \text{ClientOpt}(\theta_i^t)$
  end for
  $\theta^{t+1} \leftarrow \text{ServerOpt}(\theta^t, \{\Delta_i^t\}_{i \in S})$
end for

2.3.4 The Bayesian Federated Learning Problem

To deal with heterogeneously distributed data, it may be practical to view federated learning through a Bayesian lens. In doing so we need to expand the federated problem when having model parameters distributed as

$$\theta \sim \mathcal{P}(\theta|\mathcal{D}), \tag{2.8}$$

where $\mathcal{D} = \bigcup_i \mathcal{D}_i \ \forall i$ is the collection of data.

Using Bayes formula, we have that the posterior is proportional to the prior times the likelihood. With the data distributed among different clients, we have Bayes formula defined as a product of local likelihoods. In short, this can be written as

$$\mathcal{P}(\theta|\mathcal{D}) \propto p(\theta) \prod_{i \in S} \mathcal{L}(\mathcal{D}_i|\theta), \tag{2.9}$$

where p is the prior distribution reflecting prior information of θ and $\mathcal{L}(\mathcal{D}_i|\theta)$, $i \in S$, is the local likelihood.

The Bayesian federated problem is thereby defined as calculating a posterior distribution given the locally available data. The more evidence provided by local likelihoods, the better information we have for the most likely parameters θ, given that the model describes the relationship. The idea to reformulate the federated learning problem is explicitly referred to by Al-Shedivat et al. [29, 1] but appears central in other works [10, 9, 30]. In practice, we usually need to infer a specific θ given the posterior, and in fact, it is possible to derive that FedAvg is a MAP of the global posterior given reasonable assumptions.

Generally, the area of a probabilistic perspective in a federated setting is unexplored [6]. There have been attempts to adapt Bayesian neural networks to a federated setting with good results [31]. Other attempts have been made to use non-parametric Bayesian processes to aggregate client models [32], the works have since been further developed into algorithms such as FedMA [33] that have shown good performance compared to multiple algorithms. There is also a relevant paper by Lalitha et. al [24] using posterior distribution matching between clients to optimize convergence in a peer-to-peer setting of federated

learning. Using posterior matching Lalitha et. al [24] show convergence between clients and is an example of how taking a probabilistic perspective improve model convergence.

Even given previous research of probabilistic federated learning, leveraging this perspective in a scenario of heterogeneous distributed data is relatively unexplored. It is of interest if using the inherent variance from the local client distributions one may infer a more suitable model than by only taking an average.

2.3.5 A Bayesian Derivation of FedAvg

Given the Bayesian federated learning problem, it will now be shown that under sensible assumptions, FedAvg maximizes the global posterior likelihood, i.e., it is the MAP estimate of the global posterior. The problem of maximizing the posterior likelihood is defined as

$$\underset{\theta}{\operatorname{argmax}} \quad \mathcal{P}(\theta|\mathcal{D}) = Cp(\theta)\prod_{i\in\mathcal{S}}\mathcal{L}(\mathcal{D}_i|\theta). \tag{2.10}$$

Using the log-likelihood we have the problem

$$\begin{aligned}\underset{\theta}{\operatorname{argmax}} \quad \log(\mathcal{P}(\theta|\mathcal{D})) &= \log(C) + \log(p(\theta)) + \sum_{i\in\mathcal{S}}\ell_i(\theta) \\ \ell_i(\theta) &= \log(\mathcal{L}(\mathcal{D}_i|\theta)),\end{aligned} \tag{2.11}$$

where ℓ_i denote the local cost function of the likelihood. Assuming Gaussian local likelihood distributions $\mathcal{L}(\mathcal{D}_i|\theta) = \frac{1}{\sqrt{2}\sigma_i}\exp(-\frac{||\theta-\theta_i||^2}{2\sigma_i^2})$ with a constant variance, the local log-likelihood equals $l_i(\theta) = C_i - \frac{1}{\sqrt{2}\sigma_i}||\theta-\theta_i||^2$. In fact, it is sufficient to assume the distribution to be locally Gaussian in a small region d_i around $\theta_i : \forall i, j \in S, |\theta_i - \theta_j| < d_i$. Additionally, assuming a flat prior and that $\sigma_i = \sigma, \quad \forall i \in \mathcal{S}$ the optimization problem equals

$$\underset{\theta}{\operatorname{argmax}} \quad \log(\mathcal{P}(\theta|\mathcal{D})) = C - \frac{1}{2\sigma^2}\sum_{i\in\mathcal{S}}||\theta-\theta_i||^2. \tag{2.12}$$

Observe that this is equivalent to minimizing a sum of squares, which is solved by the arithmetic mean $\bar{\theta} = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\theta_i$. This motivates the use of FedAvg [1].

In practice, one assume that the means θ_i of the local Gaussian distributions are communicated to the server. This assumption is not necessarily true, as there is a variance associated with the local training. Additionally, there may be noise introduced by the sampling process of clients and data. The derivation FedAvg is reasonable in many situations, but it is unclear if all assumptions hold, specifically in heterogeneous environments.

2.4 Related Algorithms

The related algorithms section aims to introduce different algorithms and research within probabilistic federated learning more thoroughly. The federated algorithms establish the use of a parameter distribution on the server as how to incorporate Bayesian methods

such as MCMC sampling into a federated setting effectively. We will also touch on the parallel emergence of distributed Bayesian computing methods [34] that in recent years have moved from assuming perfect IID data to non-IID data [35], and thus entered a similar domain as federated learning.

2.4.1 FedAG

Federated Average Gaussian (FedAG) is a simple approach to incorporate probabilistic models in federated learning described in algorithm 3 [9]. This is done by estimating a Gaussian distribution from the client weights using the sample mean and variance on the server side. The expected value for each parameter distribution is broadcast and used for training in the next communication round. The authors imply using this distribution is a simple way to implement probabilistic behavior in a federated setting. They show that using the parameter distributions and performing inference similar to a Bayesian network they can produce probabilistic predictions.

Algorithm 3 FedAG, ServerOpt

Input: θ^t, $\{\Delta_i\}_{i \in \mathcal{S}}$
  $\theta_i^{t+1} \leftarrow \theta^t + \Delta_i, \forall i \in \mathcal{S}$
  $\mu^{t+1} \leftarrow \frac{1}{|S|} \sum_{i \in \mathcal{S}} \theta_i^{t+1}$
  $(\sigma^{t+1})^2 \leftarrow \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} (\theta_i^{t+1} - \mu^{t+1})^2$
  $\hat{\theta} \sim \mathcal{N}(\mu^{t+1}, \sigma^{t+1})$
  $\theta^{t+1} \leftarrow E[\hat{\theta}] = \mu^{t+1}$
Return: θ^{t+1}

Note that during the training, the algorithm is identical to FedAvg as the expected value of a Gaussian fitted from a sample is equal to the average value of that same sample. The training part is therefore not probabilistic. However, during the inference on the server the distributions are used and produce a probabilistic output. There is therefore an important difference between inference and training for FedAG where the probabilistic behavior on the server stem from the variance of the client weights. Future work is suggested to research using other distributions outside the Gaussian assumption, which is relevant for our work.

2.4.2 FedBE

Federated Bayesian Ensemble (FedBE) presented in algorithm 4, is an ensemble aggregation method built on similar ideas that guided the design of FedAG [10]. As the server receives the weights from the clients, FedBE fit them to a Gaussian distribution similar to how FedAG does. FedBE then samples M set of weights from the posterior distributions and combines them with the average weight (FedAvg). This set of weights, in conjunction with client weights, is then treated as an ensemble which is to be distilled into a single model using *stochastic weight averaging* (SWA) on an unlabeled dataset $\hat{\mathcal{D}}$ on the server.

Algorithm 4 FedBE, ServerOpt

Input: θ_t, $\{\Delta_i\}_{i\in\mathcal{S}}$, $\hat{\mathcal{D}}$
  $\theta^i \leftarrow \theta_t + \Delta_i, \forall i \in \mathcal{S}$
  $\bar{\theta} = \frac{1}{S}\sum_{i\in S}\theta^{(i)}$
  Construct posterior $p(\theta|\mathcal{D})$ from $\{\theta^i\}_{i\in S}$
  $\{\theta^{(m)} \sim p(\theta|\mathcal{D})\}_{m=1}^{M}$ ▷ Sample new models
  $\{\theta^{(m')}\}_{m'=1}^{M'} \leftarrow \{\bar{\theta}\} \cup \{\theta^{(m)}\}_{m=1}^{M} \cup \{\theta^{(i)}\}_{i\in S}$
  $\hat{\mathbf{p}}_j = \frac{1}{M'}\sum_{m'} p(\hat{\mathcal{D}}|;\theta^{(m')})$ ▷ Calculate likelihood
  $\mathcal{T} = \{\hat{\mathcal{D}}, \hat{\mathbf{p}}_j\}_{j=1}^{J}$
  Knowledge distillation: $\theta_{t+1} \leftarrow \text{SWA}(\bar{\theta}, \mathcal{T})$
Return: θ_{t+1}

This method is suitable in a context where the number of clients is low, thus one may need to bootstrap new weights to gain a better aggregation and point estimate. In general, having data available on the server is not always true, and if data is available there is a multitude of algorithms exploring similar scenarios. Due to the scope of this thesis, we will exclude algorithms that require data on the server, labeled or not. Nevertheless, FedBE establishes an idea of leveraging the posterior distribution to make an inference resulting in better performance than the average [10]. It thus expands on the idea that the inherent variance of client weights may be viewed as a distribution.

2.4.3 FedPA

Federated Posterior Averaging (FedPA) [1] is a third algorithm which take a client-side approach to incorporate a Bayesian perspective. The algorithm is derived similarly to FedAvg with the modification of not assuming constant variance between clients. M. Al-Shedivat et. al states that under uniform priors and Gaussian local posteriors one has the global posterior defined as $\mathcal{P}(\theta|\mathcal{D}) \propto \prod_{i\in S}\mathcal{P}(\theta|\mathcal{D}_i)$ which results in a Gaussian posterior with mean

$$\mu = A^{-1}\mathbf{b}. \tag{2.13}$$

Here, $A = \sum_{i\in S} q_i\Sigma_i^{-1}$ and $\mathbf{b} = \sum_{i\in S} q_i\Sigma_i^{-1}\mu_i$, where q_i is a client weight, μ_i the client mean estimate and $\Sigma_i^{-1} = X_i^T X_i$ the local feature covariance matrix of data $X_i \in \mathcal{D}_i$ [1].

Algorithm 5 FedPA, ClientOpt

Input: θ, MCMC samples: l
  $\{\theta\}_{i=1}^{l} \sim$ IASG(θ, l)
  $\mu \leftarrow \frac{1}{l}\sum_{i=1}^{l}\theta_i$
  $\Delta = \Sigma^{-1}(\theta - \mu)$
Return: Δ

Algorithm 6 IASG sampling

Input: θ, $Optimizer$, steps per sample: K, samples: l
for $s = 1, ..., l$ **do**
  $\mathcal{S} \leftarrow \emptyset$
  for $t = 1, ..., K$ **do**
    $\theta \leftarrow Optimizer(\theta)$ ▷ e.g., SGD
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\theta\}$
  end for
  $\theta_s \leftarrow Average(\mathcal{S})$
end for
Return: $\{\theta_1, ..., \theta_l\}$

To calculate this new mean the authors show that, instead of sending $\Sigma_i^{-1}, i \in \mathcal{S}$, to the central server for computation, a global gradient can be approximated from a sum of locally computed gradients as

$$\Delta \approx \sum_{i \in S} \Delta_i = \sum_{i \in S} \Sigma_i^{-1}(\theta - \mu_i). \tag{2.14}$$

The local covariance matrix Σ is estimated through shrinked covariance matrix estimation [36], which introduces a *shrinkage* hyperparameter. Further, FedPA uses local MCMC sampling through *iterate averaging stochastic gradient* (IASG) sampling, see algorithm 6, [37] where the average of multiple SGD steps can be seen as an MCMC posterior mean estimate. The sample of posterior mean estimates is then used to approximate the covariance. This way FedPA fits into the FedOpt framework as only the client optimization in algorithm 2 needs to be adjusted, the final algorithm for FedPA is described in algorithm 5.

The advantage of FedPA is that it standardizes the resulting posterior sample with respect to the inherent parameter covariance and mean. As a consequence, all posterior samples will approximately have an identity covariance, which motivates common assumptions of diagonal variance such as in FedAvg. Further, the approach is well motivated by the Bayesian federated learning problem and, like FedBE, results in better performance than FedAvg. FedPA also directly connects federated learning and Bayesian methods such as MCMC sampling and is one of the algorithms this thesis focuses on.

2.4.4 Distributed Bayesian Methods

As previously stated, MCMC algorithms is a commonly used technique to estimate the posterior. MCMC techniques usually use multiple chains run in parallel to find the posterior. This follows from that the convergence towards the posterior is independent on the starting sample of the simulation. Hence, you can run multiple chains that all converge towards the same posterior [19].

In essence, federated learning is very similar to a MCMC technique with local SGD seen as a single chain and, under IID conditions, all resulting client weights are posterior sam-

ples. In contrast, federated learning clients only communicate the final weight in the chain while a true MCMC technique would seek to find the MAP estimate, usually through a chain average.

The similarities between a federated learning setting with IID data and the distributed MCMC algorithm are striking. However, in many cases where one uses federated learning one does not expect the data to be homogeneous between clients. This results in different chains not converging to the same posterior. To adjust for different posteriors, the consensus Monte Carlo algorithm [38] shows that one can use non-IID data in a distributed Bayesian setting to learn the global posterior as

$$p(\theta|\mathcal{D}) \propto \prod_{i \in S} p(\theta)^{\frac{1}{|S|}} \mathcal{L}(\mathcal{D}_i|\theta), \tag{2.15}$$

where $p(\theta)^{\frac{1}{|S|}}$ is the weighted prior and $\mathcal{L}(\mathcal{D}_i|\theta)$ is the local likelihood. Observe that each product is a local posterior as $p(\theta|\mathcal{D}_i) \propto p(\theta)^{\frac{1}{|S|}} \mathcal{L}(\mathcal{D}_i|\theta)$. Still the consensus Monte Carlo algorithm derived from (2.15) relies on having data present on the server, which is a limitation for federated learning [38].

A way to leverage the parallel between MCMC techniques and federated learning is through *Langevin Monte Carlo* (LMC) [39], which on the server-side seeks to update weights according to

$$\theta_{t+1} = \theta_t - \eta \nabla \log[p(\theta|\mathcal{D})] + \xi, \tag{2.16}$$

where $\xi \sim \mathcal{N}(0, 2\eta)$ is Gaussian noise added to the new posterior estimate. Through adding noise a LMC chain explores the global posterior, it is therefore a Metropolis-adjusted Langevin algorithm [40].

A practical algorithm that uses LMC theory is *stochastic gradient Langevin dynamics* (SGLD). The SGLD algorithm uses FedAvg and SGD to estimate the gradient $\nabla log[p(\theta|\mathcal{D})]$. Taking the average of the chain after a burn-in period s results in a MAP estimate of the global posterior.

The similarities between (2.16) and FedAvg are striking and the possibility to incorporate LMC methods in standard federated learning is a relatively new area of research. Liu, and Simeone use the noise generated from SGLD in combination with noise generated by wireless communication to derive an optimal learning rate [41]. It has also been shown that using conducive gradients in SGLD lead to better posterior estimation in the scenario of heterogeneous data [42]. Sadly, with model parameters being large-scale posteriors, SGLD shows slow convergence and that there is a trade-off between scale and communication [43].

Even with some complications regarding convergence, federated SGLD show an interesting concept drawing a clear parallel between Bayesian MCMC sampling and federated learning. Regarding finding a MAP estimate, assuming posterior samples to be Gaussian distributed may not be optimal. Specifically in non-IID scenarios where different chains may converge towards different local posteriors. Instead, one may want to estimate the MAP through other means. One way would be through a non-parametric approach, for example using a kernel density estimation of the global posterior. An approach which is to be discussed in further detail below.

2.5 Latent Dirichlet Allocation

Even if non-IID behavior is a crucial aspect of federated learning, it can be hard to test as many bench-marking datasets are centralized. There is hence a need to divide datasets into federated versions. It has been argued that a natural way to split data into federated clients is with *latent Dirichlet allocation* (LDA) [28, 44]. Latent Dirichlet allocation introduces a parameter α that determines how uniform client label distributions are. Therefore, changing α enables an easy way to adjust the label skew, which serves our purpose of examining varying non-IID settings.

The main idea of LDA is to sample clients from the centralized dataset based on their labels. Sampling is done through a multinomial distribution, where the probabilities used are sampled from a Dirichlet distribution. The parameter selection of the Dirichlet distribution lets us select how uniform the sampling of labels is supposed to be. The algorithm used in this thesis and proposed by [28] is presented in algorithm 7. The number of clients and number of samples per client is denoted as M and N respectively, G_0 is the set of available labels, and finally, α is the Dirichlet parameter.

Algorithm 7 Latent Dirichlet Allocation

Require: α, N and M
  for $m = 1, ..., M$ **do**
    $\theta_{\text{label}} \sim \text{Dir}(\alpha, G_0)$ ▷ Sample probabilities of selecting labels.
    $D_m = \emptyset$ ▷ Create an empty set for client m.
    for $n = 1, ..., N$ **do**
      $y \sim \text{Multinomial}(\theta_{\text{label}})$
      Sample image x uniformly given label y
      $D_m = D_m \cup \{(x, y)\}$
      Remove samples x from available samples
      if label y contain no available samples **then**
        $G_0 = G_0 \setminus \{y\}$
        Normalize θ_{label}
      end if
    end for
  end for

2.6 Kernel Density Estimation

In an approach to estimate the global posterior from local posterior samples, parametric methods are not always suitable. Instead, *kernel density estimation* (KDE) is a non-parametric approach to estimating the true density of distribution from a sample [45]. Assuming samples $\{\theta_1, ..., \theta_N\}$ drawn from a distribution with probability density func-

tion (PDF) f we may create an estimate of the distribution at any point θ as

$$\hat{f}_h = \frac{1}{N}\sum_{i=1}^{N} K_h(\theta - \theta_i) = \frac{1}{Nh}\sum_{i=1}^{N} K\Big(\frac{\theta - \theta_i}{h}\Big). \tag{2.17}$$

Here K is a kernel function and h is the bandwidth. As such, there are two decision to be made, what kernel to use and what bandwidth to use. A commonly used kernel is the Gaussian, defined as

$$K_h(u) = \frac{1}{\sqrt{2\pi}h}\exp\Big(-||\frac{u}{\sqrt{2}h}||^2\Big).$$

Another common kernel is the Epanechnikov kernel defined as

$$K_h(u) = \begin{cases} \frac{3}{4h}(1 - (||\frac{u}{h}||^2), & ||\frac{u}{h}|| \leq 1 \\ 0, & ||\frac{u}{h}|| > 1, \end{cases}$$

which can be shown to be optimal with regard to the mean integrated square error towards the true distribution [46]. Observe that these kernels are strictly positive and integrate to 1, which makes the the kernel density a true probability density function. An illustrative example of a kernel density is shown in figure 2.3.

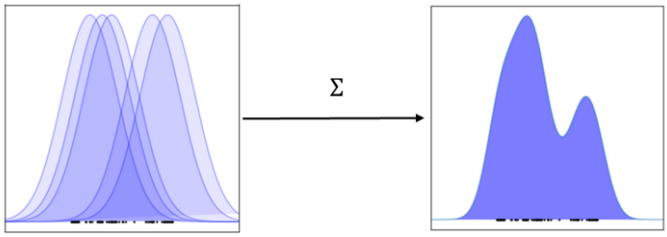


Figure 2.3: Schematic illustration of a kernel density estimation using local Gaussian kernels.

Besides choosing which kernel may be the most suitable, there is also a problem when selecting the optimum bandwidth h, commonly referred to as the "bandwidth selection problem". Too small bandwidths will create a biased estimator that is too reliant on the samples provided. Too high bandwidth will instead smooth the density and remove any multi-modalities that might have been present in the true distribution. Optimizing the bandwidth one usually seeks to minimize the *mean integrated square error* (MISE) between the true distribution f and the kernel density $\hat{f}_h$ defined as

$$MISE = \int_{-\infty}^{\infty} (f(x) - \hat{f}_h(x))^2 dx. \tag{2.18}$$

Using the weak assumption of f being twice differentiable and a Taylor expansion [47] on arrives at the approximate mean integrated square error

$$\begin{aligned} AMISE &= \frac{R(K)}{Nh} + \frac{1}{4}m_2(K)^2 h^4 R(f'') \\ R(g) &= \int_{-\infty}^{\infty} g(x)^2 dx \\ m_2(g) &= \int_{-\infty}^{\infty} x^2 g(x) dx \end{aligned} \tag{2.19}$$

Taking the derivative with respect to h to find the minima of the AMISE one arrives at the optimal bandwidth

$$h_{AMISE} = \left(\frac{R(K)}{m_2(K)^2 R(f'')} \right)^{\frac{1}{5}} N^{-\frac{1}{5}}. \tag{2.20}$$

Now observe that the minimization problem only has a single unknown quantity being $R(f'')$ which through its estimation is the source for most bandwidth selection techniques.

The classical *rule of thumb bandwidth* (ROT) is using normal approximation bandwidth to solve $R(f'')$ which in that case equals the variance of the PDF. Using *Silverman's rule* [48] one estimates the variance through the sample variance $\hat{\sigma}^2$, or the interquartile range (IQR) as

$$h = 0.9 \min \left(\hat{\sigma}, \frac{IQR}{1.34} \right) N^{-\frac{1}{5}}. \tag{2.21}$$

This bandwidth selection is commonly used due to it's simplicity. Generally it tends to be over-smoothed for non-Gaussian distributions and show weak convergence towards the true density distribution [49]. In that sense, using Silverman's bandwidth selection rule we are not using an optimal technique but rather a simple rule-of-thumb [50]. Further bandwidth selection techniques can be found in Appendix C.

In figure 2.4 an illustration of a KDE using Epanechnikov kernels and Silverman rule bandwidth is observed. As can be seen, it shows how the use of kernel densities successfully adapts to skewness and multi-modalities in the distributions, a useful property when potential parameter distributions are non-Gaussian.

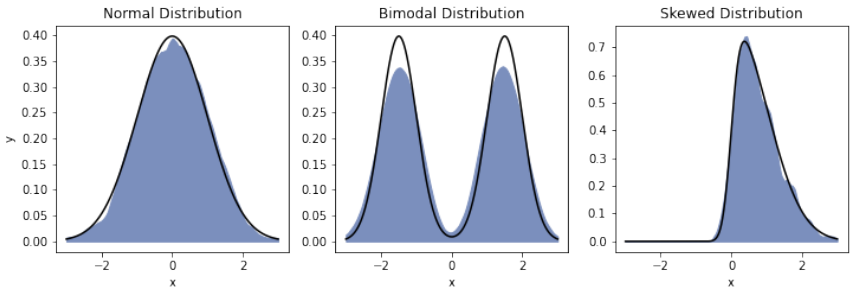


Figure 2.4: Illustration of a kernel density estimation on different type of distribution using an Epanenchikov kernel density with Silverman's rule on 100 samples.

Kernel density estimation has not seen much use in federated learning. In a paper by Christos et al. (FedSparse) [51] kernel density estimation is used to filter outliers in a federated setting. In another example, Rahif and Simeone show how kernel densities can be communicated through particle-based techniques in a federated setting [52]. These approaches do show how kernel estimation can be used in a federated setting, but to our knowledge, no previous work has been done to infer the MAP of the global KDE.

Given a probability density, a common problem is the inference of the most likely value. In classical parametric statistics, this usually has a parametric solution. In kernel density estimation it is more complicated. Due to the non-convex shape of kernel densities, defining a technique that finds the optima is not straightforward, specifically in high dimensional problems [53]. There are some techniques available [54, 55] but generally, no

technique can guarantee to find a global optimum. Instead, breadth-first algorithms are commonly used, which convergences to local maxima but rely on good initial values. In this thesis mean-shift is one such algorithm which is to be explained further below.

2.7 Mean Shift

The mean shift algorithm is a density gradient estimation algorithm commonly used for clustering and image segmentation [55]. It works by iteratively calculating

$$m(\theta) = \frac{\sum_{i \in S} K_h(\theta_i - \theta)\theta_i}{\sum_{i \in S} K_h(\theta_i - \theta)},$$

where K_h is a kernel and thus the sum $\sum_{i \in S} K_h(\theta_i - \theta)$ a kernel density estimate given the samples $\{\theta_i\}_{i \in S}$. The expression $m(\theta) - \theta$ is referred to as the mean shift. By iteratively reassigning the parameter, $\theta \leftarrow m(\theta)$, we are moving towards a mode of the kernel density. In a sense, mean-shift is a direct application of the Banach fixed-point theorem, written in 1922 [56].

The algorithm does not guarantee convergence towards any global optima [57], but due to the local complexity of kernel densities, it will converge towards the closet mode [58]. The iterative process of the mean shift algorithm is described in algorithm 8. Mean shift is commonly used for image segmentation and clustering but is relatively unused outside these areas of ML. To our knowledge, this is the first paper to study how mean-shift can be used and implemented in a federated setting to provide better model inference in non-IID scenarios.

Algorithm 8 Mean-shift

Require: samples $\{\theta_i\}, i \in S$, inital value θ_0, tolerance τ, max iterations t_{max}.
  $t = 1$
  do
    $\theta_t = \frac{\sum_{i \in S} \theta_i K_h(\theta_i - \theta_{t-1})}{\sum_{i \in S} K_h(\theta_i - \theta_{t-1})}$
    $t = t + 1$
  while $(\theta_t - \theta_{t-1} > \tau)$ and $t < t_{max}$
  return θ_{t-1}

This thesis will seek to estimate the mode of the posterior distribution obtained by kernel density estimation and then use mean-shift to infer a new optimal weight. How this is explicitly done, and the motivation behind using such a technique, will be explained further in the methodology part of the thesis, which include a short motivation and then a description of said algorithm.

3 | Methodology

The central idea in many of the previous works is the assumption of a Gaussian global posterior. This thesis presents the motivation for why such an assumption might not be suitable for non-IID environments. Based on such a motivation we present the FedKP algorithm and describe new ways to describe the global posterior and how inference on such a posterior can be made.

3.1 Motivation

A common approach in Bayesian federated learning is the assumption of a Gaussian global posterior. This is sensible in the homogeneous case as local posterior samples θ_i for $i \in \mathcal{S}$ can be assumed to be drawn from the same posterior distribution as local posteriors are identical due to sampling from the same data distribution as per equation (2.6). These posterior samples can further be assumed Gaussian around an MAP $\bar{\theta}$ since MCMC chains in deep neural networks converge towards a local Gaussian noise [59]. By the central limit theorem [60] we thus have that

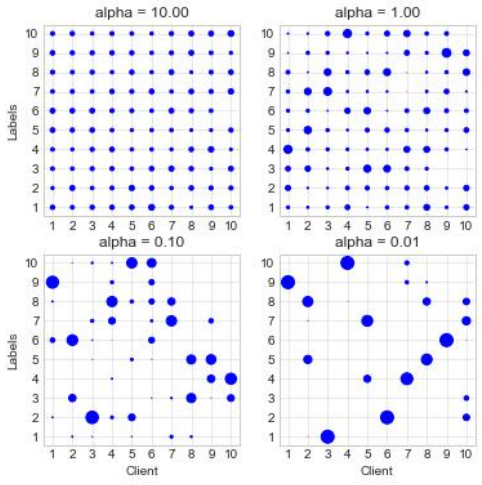
$$\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\theta_i \sim \mathcal{N}\Big(\bar{\theta}, \frac{1}{\sqrt{|\mathcal{S}|}}\Sigma\Big).$$

If data is non-IID the posterior samples are not identically distributed, and the MAP may differ between local posteriors. In such scenario the central limit theorem does not necessarily hold, and therefore the average is not necessarily the MAP estimate of the global posterior.

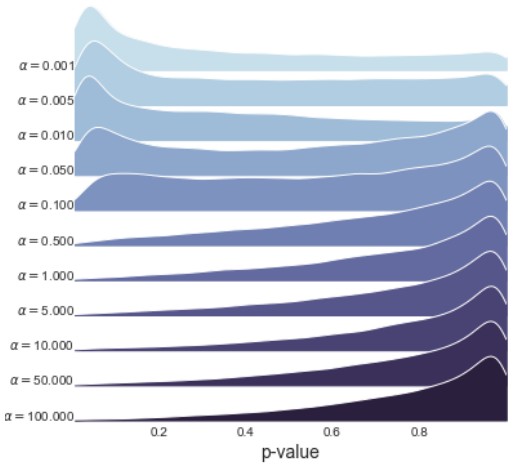
Studying how well this assumption hold for varying levels of non-IID data is the preliminary research done in this thesis. It is not unreasonable that in non-IID setting, different clients have different data and loss landscape. This would cause clients to converge towards different optima and distributions not centered around a common minimum. Instead, one may expect multi-modal posterior distributions which makes inference of the global model harder. To provide more intuition for the logic argued, an exploration of MNIST in a federated setting was done to research how Gaussian the posterior samples are under different levels of non-IID.

3.1.1 MNIST Example

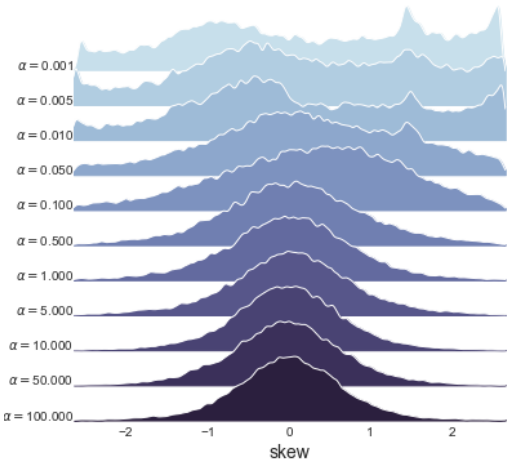
To attain an intuition for the previous logic described, an exploration of MNIST in a federated setting was done. The MNIST dataset is a lightweight dataset of handwritten digits suitable for smaller experiments. We split the dataset into a federated version by using LDA. The sampling can be done varying degrees non-IID by changing the parameter α, an example of such sampling can be seen in figure 3.1a where the amount of each label sampled correspond with the dot size. In this experiment each client has an equal amount of 5470 samples and as can be seen the label distribution become more homogeneous as α increase. For this example we let each client train a small CNN for a single epoch and study the properties of the resulting posterior samples from all models. A full description of the model used and other hyperparameters can be found in the Appendix D.



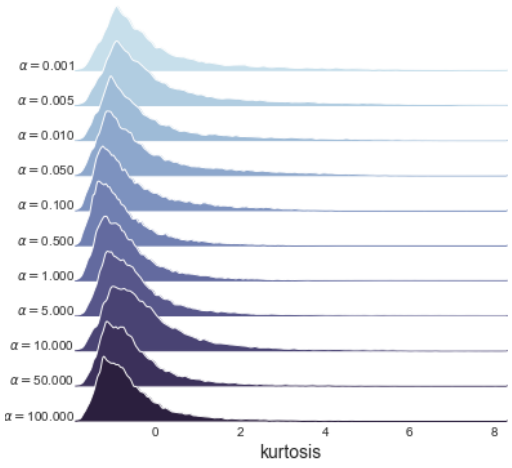
(a) Distribution of labels for different α.



(b) KS-test after one epoch for different α.



(c) Skewness after one epoch for different α.



(d) Kurtosis after one epoch for different α.

Studying how quantiles of separate parameters agree with a normal distribution, through

taking a Kolmogorov–Smirnov test on each parameter θ_i, we have the densities shown in figure 3.1b. As can be seen, there is a shift between the IID and non-IID cases where densities are more centered around 0 rather than 1. This implies that as data becomes more non-IID distributed the posterior samples are less likely to be Gaussian distributed. This is under the condition of studying each parameter separately, and covariance between factors may exaggerate non-Gaussian behavior but the results should be viewed in relation to the IID case. Observe that non-Gaussian distributions are more prominent in non-IID scenarios in comparison to IID scenarios.

Studying how the empirical skewness is distributed for the different parameters, we again see very similar behavior in figure 3.1c with less 0 centered skewness as we increase the amount of non-IID behavior in the data. In a sense, this implies that the global posterior distribution becomes less symmetrical as α decreases. In the extreme scenario of $\alpha = 0.001$ skewness is no longer 0 centered, but instead more uniform, implying non-symmetric distributed samples.

Studying the empirical kurtosis, seen in figure 3.1d, one observes more heavy tails as we increase the amount of non-IID behaviour. The difference is not as apparent compared to skewness and the ks-test, but there is a difference with the amount of non-IID. The difference may be less apparent as heavy tails largely depend on the amount of SGD steps made during training. As all clients contain the same amount of local data, these steps are mostly the same, the total size of the gradient step (the norm) in similar between clients. The subtle difference implies that the non-IID scenario have fatter tails compared to the IID case, implying non-Gaussian behavior.

To finish this motivational example, we also illustrate the performance of an average model on test-data. This can be seen in figure 3.2 where the model performance decreases as we increase the amount of non-IID data. Do observe the relative smooth transition in model performance regarding the amount of non-IID follows the previous conclusion that non-Gaussian behavior being smooth with regards to the amount of non-IID.

There are also other factors that affect the resulting posterior samples. In the example shown here, MNIST is explored for a specific model and training parameters. In Appendix A further analysis is done with respect to changing the number of local epochs, rounds of training, and the number of clients included in the federated setting.

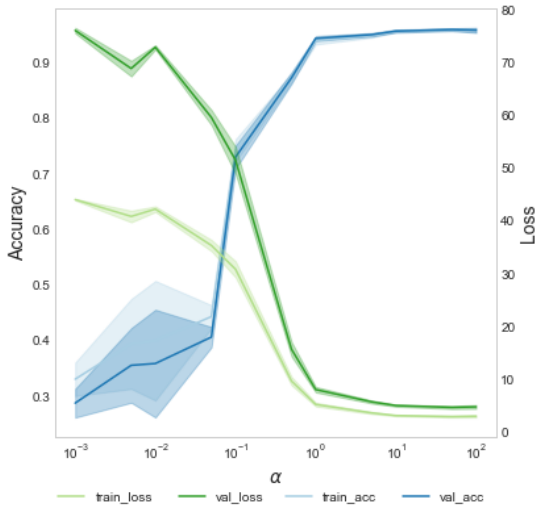


Figure 3.2: Loss and accuracy for different levels of α when training for 1 communication round using FedAvg.

In the end, it is not clear that the average model is the most suitable inference due to the non-Gaussian behavior of a heterogeneous setting. Instead, it may be sensible to study different aggregation techniques, specifically techniques suitable when the global parameter posteriors are non-Gaussian. Assuming other parametric distributions may not be an easy task, even though there is research testing such approaches. Instead, a non-parametric approach may be reasonable; which is the main idea of the proposed *federated kernel posterior* (FedKP) algorithm.

3.2 FedKP

Based on the preliminary work done, Gaussian assumptions may not be suitable in a non-IID setting of federated learning. Instead, a more plausible approach is to look beyond parametric solutions and look into techniques from the non-parametric realm. In this paper, we propose a novel algorithm, FedKP, where instead of assuming a Gaussian global posterior, one estimates the global posterior through kernel estimation

$$\log\left(\mathcal{P}(\theta|\mathcal{D})\right) = \ell(\theta|\mathcal{D}) = C + \sum_{i\in\mathcal{S}} K_h(\theta - \theta_i). \tag{3.1}$$

We denote K_h as the kernel function, θ_i as the local posterior sample from $p(\theta|\mathcal{D}_i)$, and h as the bandwidth parameter. The Kernel function used in this paper is the Epanechnikov kernel with a Silverman bandwidth selection rule. This choice of kernel is motivated by its similarities to maximizing the posterior log-likelihood described in Appendix B.

To provide intuition for kernel density estimation, one observes in figure 3.3 the difference between the kernel densities of a random bias weight of the final layer in the MNIST example. As can be seen, there is a difference in the appearance of the KDEs where the IID example seemingly is more Gaussian and the non-IID is non-Gaussian. This example uses a Silverman's bandwidth rule and an Epanechnikov kernel.

It is important to highlight that the choice of bandwidth changes the look of the posterior distribution. Larger bandwidths correlate to a more uni-modal kernel density estimate meanwhile a smaller bandwidth correlates to a more multi-modal density. It is shown in Appendix B that FedKP is a regularization of FedAvg where an infinite bandwidth corresponds to taking the average. As such FedKP shares similarities with FedSparse [51], with the difference that FedKP uses the kernel density for inference, not filtering.

3.2.1 Inference

Given the posterior kernels, the inference of an optimal parameter selection is not obvious. In subject to the Bayesian federated learning problem, we seek to estimate the MAP. Selecting the MAP with respect to the kernel estimate leads to a model selection which can be compared to taking the most common model (mode) instead of the average model. This potentially has nice properties in the non-IID scenario where the average model may

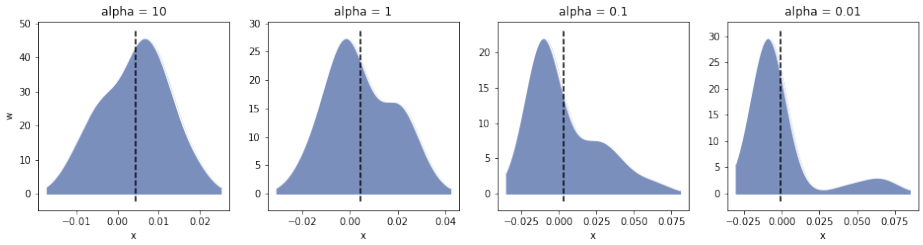


Figure 3.3: Difference in kernel estimates on final bias layer of MNIST example, dashed line denote the mean.

have worse performance than the most frequent model.

To calculate the mode of a kernel density we propose using the mean-shift algorithm. The advantages of mean shift are its simplicity and lightweight calculation. One does not need to calculate the kernel but only needs to iterate through client weights θ_i, $i \in S$. This leads to the worst-case complexity of $\mathcal{O}(t_{max}|\mathcal{S}|)$, where t_{max} is a limit on the amount of mean-shift iterations done.

The algorithm does not guarantee convergence towards any global optimum [57], but due to the local complexity of kernel densities, it will converge towards the closet mode [58]. Therefore, a sensible choice of initial value could be the mean of client weights. This way FedKP adjusts from the federated average towards the nearest mode of the posterior.

There are other approaches to estimate the MAP [54] which could be of interest, but due to the relative computational complexity and unexplored area of kernel posteriors, simplicity was chosen and more advanced estimation techniques are left for future work.

In summary, the proposed algorithm is defined in algorithm 9, as seen it is only a slight modification of FedAvg, adding the mean-shift of the mean to attain the closest mode.

Algorithm 9 FedKP

Require: intial value θ_0, tolerance τ, max iterations t_{max}.
  for $t = 0, 1, 2, \ldots$ **do**
    Sample subset S of clients $i \sim Q$.
    Communicate θ_t to all $i \in S$.
    for $i \in S$ **do** in parallel
      $\theta^i_{t+1} \leftarrow$ CLIENTUPDATE(θ_t)
    end for
    $\bar{\theta}_{t+1} \leftarrow \frac{1}{|S|} \sum_{i \in S} \theta^i_{t+1}$
    $\theta_{t+1} \leftarrow$ MeanShift$(\bar{\theta}_{t+1}, \{\theta^i_{t+1}\}_{i \in S})$
  end for

As stated, the mode is sensible as it is likely to be the best model for the majority of clients after training. In a more general purpose to find the best model for all clients, there is a risk that selecting the MAP of every parameter independently leads to poor performance for some clients. This problem arise when the posterior is multi-modal with multiple sensible models.

With h seen as a regularization parameter for deciding the amount modes, setting low bandwidths may lead to largely multi-modal densities. In an attempt to adjust for potential issues with multi-modal densities we wish to somehow smooth the parameter inference compared to strictly taking the MAP. Therefore, a version of FedKP that cluster each posterior sample into its nearest mode and then perform averaging to aggregate the shifted parameter is presented, see algorithm 10 for details. This solution does for uni-modal distributions, result in the same model as regular FedKP as all mean-shifted client weights converge to the same mode. In contrast this solution solves the problem when different clients converge to different models. In that scenario the modified version of FedKP takes the average proportional to how common each model is. Note that such a solution is closer to FedAvg, but is in some aspect refined as it clusters models which has converged to similar, or the same, models. Thus effectively removing noise from the posterior samples before averaging.

Algorithm 10 FedKP with clustering

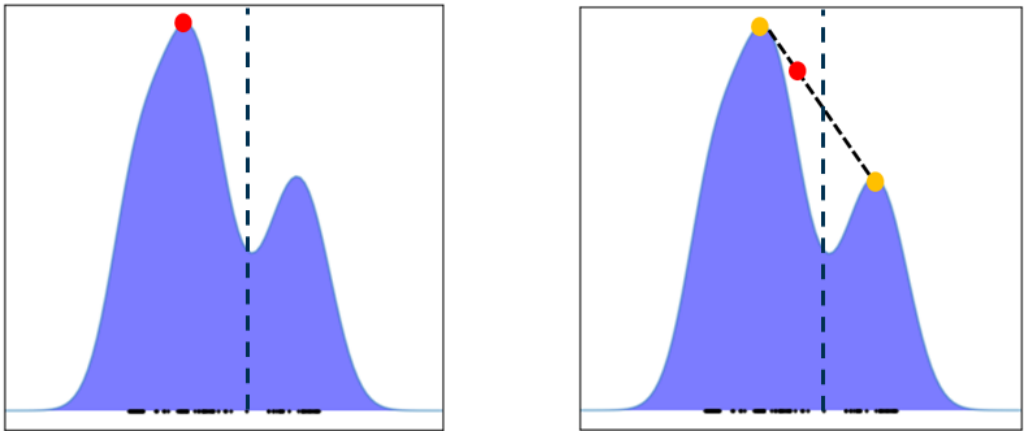
Require: intial value θ_0, tolerance τ, max iterations t_{max}.
  for $t = 0, 1, 2, ...$ **do**
    Sample subset S of clients $i \sim Q$.
    Communicate θ_t to all $i \in S$.
    for $i \in S$ **do** in parallel
      $\theta^i_{t+1} \leftarrow$ CLIENTUPDATE(θ_t)
    end for
    for $i \in S$ **do** in parallel
      $\hat{\theta}^i \leftarrow$ MeanShift$(\theta^i_{t+1}, \{\theta^i_{t+1}\}_{i \in S})$
    end for
    $\theta_{t+1} \leftarrow \frac{1}{|S|} \sum_{i \in S} \hat{\theta}^i_{t+1}$
  end for

Using a client-shifted approach adds a layered complexity and has a worst-case convergence in $\mathcal{O}(t_{max}|\mathcal{S}|^2)$ time. This does not pose any obvious problems as the mean-shift algorithm is lightweight, specifically on a server with good computational power. Both variants of FedKP are illustrated in figure 3.4.

3.2.2 The Problem With Dimensionality

A problem dismissed throughout the previous description is the curse of dimensionality associated with calculating norms of a large-dimensional problem. Since approximating the posterior through a single multivariate kernel function result is numerically unstable calculations and bad convergence rates to the true posterior [7]. Instead, FedKP assumes independent variables, and each parameter can be mean-shifted separately. This is an assumption but is motivated by the correlation being reflected in the independent densities with the highest density of a single parameter reflecting the highest density of a multivariate solution.

Solving the federated learning problem while assuming diagonal covariance is not rare [10, 9, 51], but often unmotivated. To adjust for the problem with covariance we propose



(a) Illustration of standard FedKP **(b)** Illustration of FedKP with clustering.

Figure 3.4: Illustration of FedKP with and without clustering. The vertical dashed line denote the average, red dot the resulting inference, yellow dots the modes, and black dots along the x-axis the samples.

a combined solution of using FedPA optimization [1] on the client-side. In FedPA clients standardize the local posterior sample towards the covariance as

$$\Delta_i = \sigma_i^{-1}(\theta - \mu_i). \tag{3.2}$$

Standardizing the posterior sample make the resulting covariance be approximately the identity matrix, which would motivate the assumptions made by FedKP. Therefore a final algorithm will be tested which make use of FedPA client optimization in combination with clustered FedKP on the server-side. This solution is implemented as FedKPPA and is to be compared to the other solutions in the results.

3.2.3 Hyperparameter Tuning

With the introduction of FedKP, a series of new hyperparameters are introduced. These concern how to tune the kernel density, and concerns two choices:

- Choice of kernel.
- Bandwidth selection technique.

The choice of the kernel will be an Epanechnikov kernel, primarily due to the proven optimality [46] and analog to maximizing the global log-likelihood as argued in Appendix B. However, the choice of other kernels may also be reasonable. For example, a Gaussian kernel is sensible in the context of posterior samples being means around a Gaussian local posterior. In that sense the choice of kernel is a parametric assumption of the local posterior distributions.

There has been previous work in studying bandwidth selection methods [45]. Generally, convergence is slow, and finding a good bandwidth is hard. Most approaches are different

ways to estimate $R(f'')$ and minimize the AMISE. In some sense, a large bandwidth is suitable to avoid multi-modality. It may also be sensible with a small one in much more heterogeneous data.

Throughout the experiments and results in Chapter 5, a basic set of hyperparameters are used. We use the Epanechnikov kernel with a Silverman bandwidth selection rule. A more detailed discussion of different bandwidth selection rules can be found in Appendix C.

4 | Experiment

The first goal of this thesis is to understand how FedAvg, FedPA, and SGLD perform in a non-IID federated setting. Further, we compare these algorithms with FedKP. We use three versions of FedKP: the standard defined in algorithm 9, one with clustering denoted FedKP_cluster_mean defined in algorithm 10, and a combination of algorithm 10 with gradients produced by FedPA defined as FedKPPA. We implement each algorithm and evaluate it on models using datasets with varying degrees of non-IID behavior.

The datasets used in the experiments is the federated version of **CIFAR-100** [61], **MNIST** [12], and **EMNIST-62** [62] which all are image classification problems. These datasets are common benchmarks within federated learning and suitable for comparable results [28, 1, 10]. Hyperparameters was selected to replicate the hyperparameters used by Al-Shedivat et al. [1]. The selection of hyperparameters is seen in table 4.1, full model details is found in Appendix D, and the code is available on GitHub[1].

4.1 MNIST

MNIST is a collection of hand-written digits ranging from 0 to 9 and training a CNN on MNIST is considered an easy task in machine learning. Therefore, MNIST is used as part of the exploratory work leading up to the FedKP algorithm. We also conduct experiments on the MNIST dataset to compare the different algorithms.

A CNN architecture, described in detail in Appendix D is used, and the data is split using a LDA with parameter α. This way, the data allocation will simulate scenarios with label skewed data. The data is split into 100 clients with varying degrees of non-IID through different α.

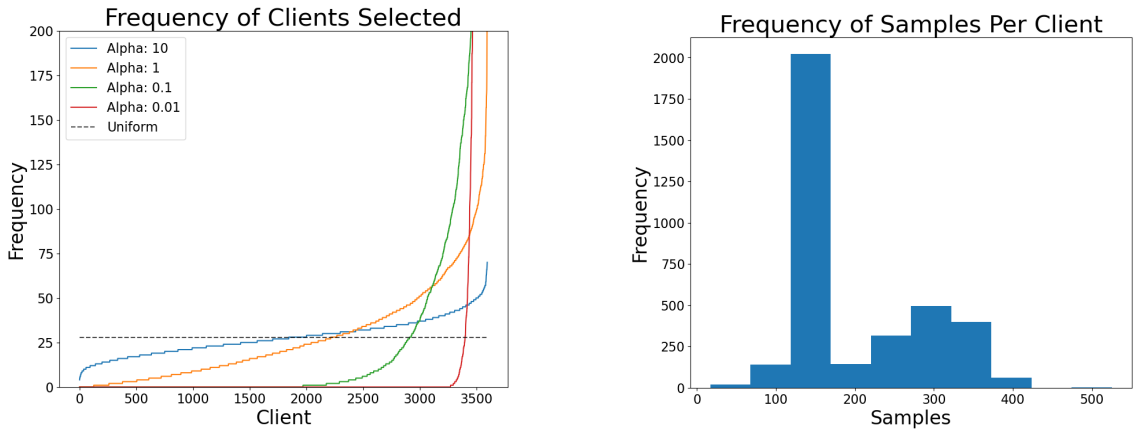
As MNIST is regarded as an easy machine learning task, evaluating it in a federated setting can be done relatively quickly. Hence, the MNIST experiments also measure how performance depend on the number of clients per communication round and the number of local epochs. In essence, reflecting the exploratory work done in Appendix A. Varying these hyperparameters, we compare performance between FedKP approaches and FedAvg.

[1] https://github.com/MartinHex/master-thesis

4.2 EMNIST-62

EMNIST-62 is the extension of the MNIST dataset, containing the digits 0-9 and letters a-z both in upper- and lower-case. The commonly used dataset is centralized, but it turns out that one can recall the original authors and thus create a naturally federated dataset split into unique writers [63]. During our experiments we use the federated EMNIST-62 dataset, often referred to as FEMNIST, produced by the LEAF[2] script written by S. Caldas et al. The options chosen can be found in table D.3.

When constructing the FEMNIST dataset a natural difference between P_i and P_j from equation (2.6) occurs when $i \neq j$, since the data is created by different authors. As authors write each letter differently a feature skew is introduced, and the dataset, therefore, contains natural heterogeneous behavior. Since the distribution P_i for any client is fixed we cannot effectively test how different data distributions of P_i impact the algorithms on the FEMNIST dataset. We instead focus on the $\mathcal{Q}$ distribution from equation (2.7), by changing the $\mathcal{Q}$ distribution we can control how uniform the selection of clients at each round is. We propose to define $\mathcal{Q} = \text{Multinomial}(\theta)$ where $\theta \sim \text{Dir}(\alpha)$, changing the parameter α would thus let us control how uniform the distribution of clients at each round is. When $\alpha \to \infty$ we regain a uniform distribution of clients at each round. The impact of different α used throughout this experiment is illustrated in figure 4.1a.



(a) Ordered client sampling frequency.

(b) Size of all client training datasets.

Figure 4.1: Illustrative examples for the EMNIST-62 sampling process.

Classification is done with a simple CNN with architecture described in detail in table D.2 in Appendix D. This is the same model as used in [28, 1][3]. When creating the model a Xavier initiation was used [64].

[2]https://github.com/TalwalkarLab/leaf/tree/master/data/femnist

[3]We have added activation functions as done in their code implementations but not described in their papers.

4.3 CIFAR-100

CIFAR-100 is an image dataset consisting of 100 different labels. The labels are hierarchically ordered where each image has both a "coarse" and "fine" label where each fine label is part of a coarse label. As such, each coarse label form groups of finer labels. For example, the CIFAR-100 dataset contain a coarse label called "aquatic mammals" and examples of fine labels that correspond to this coarse label could be "dolphine" or "otter". figure 4.2 show an illustration of how the dataset is structured.

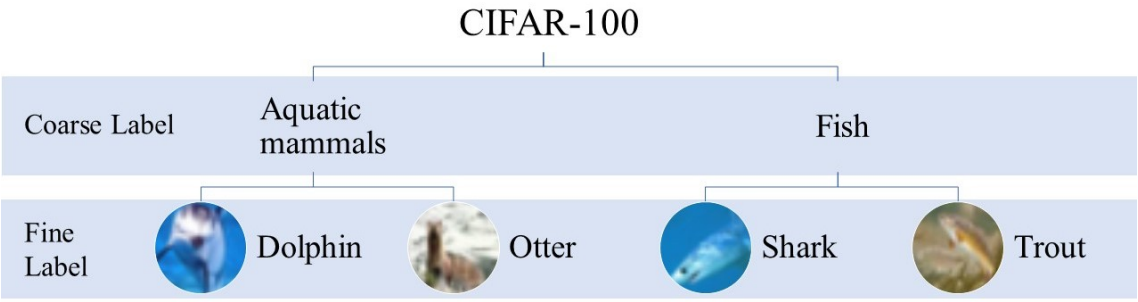


Figure 4.2: Illustration of how the CIFAR-100 dataset is structured into two levels of labels. Where each sample correspond to both a coarse and fine label and there is a relation between the level of labels.

To use the CIFAR-100 dataset in a federated setting we need to divide the data into clients. We do this by using a hierarchical LDA proposed by [28]. The hierarchical allocation uses the structure of the CIFAR-100 dataset to use LDA to sample a coarse label, and then within that coarse label again use LDA to sample a fine label. The hierarchical LDA, therefore, uses two Dirichlet parameters which we will denote α for the coarse label and β for the fine label. Hierarchical LDA produces natural clients in the sense that reducing α introduces clients that are domain-specific in that they are likely to view multiple different fine labels within a specific coarse label.

During the experiment we primarily change the behavior in terms of the coarse label sampling, meaning we vary α. Specifically, we test $\alpha \in [0.01, 0.1, 1, 10]$ while maintaining the label sampling under each course label fixed at $\beta = 10$ and use $M = 500$ and $N = 100$. We use a ResNet-18 architecture [13] to conduct the experiments and training is done by uniformly sampling 20 clients each communication round.

Table 4.1: Hyperparameters used for the experiments conducted in this thesis. The $\pm$ imply the standard deviation over amount of samples at each client.

Name	**MNIST**	**EMNIST-62**	**CIFAR-100**
Number of clients	100	3700	500
Client samples	540 ± 0	204.18 ± 79.95	100 ± 0
clients per round	20	100	20
batch-size	16	20	20
local epochs	5	5	20
server momentum	0.0	0.0	0.0
client momentum	0.9	0.9	0.9
server learning rate	0.5	0.5	0.5
client learning rate	0.001	0.01	0.01
burn-in	25	500	400
shrinkage	0.1	0.1	0.01
t_{max}	20	20	20
server optimizer	SGD	SGD	SGD
client optimizer	SGD	SGD	SGD
bandwidth	Silverman	Silverman	Silverman
kernel function	Epanechnikov	Epanechnikov	Epanechnikov
rounds	50	1000	800

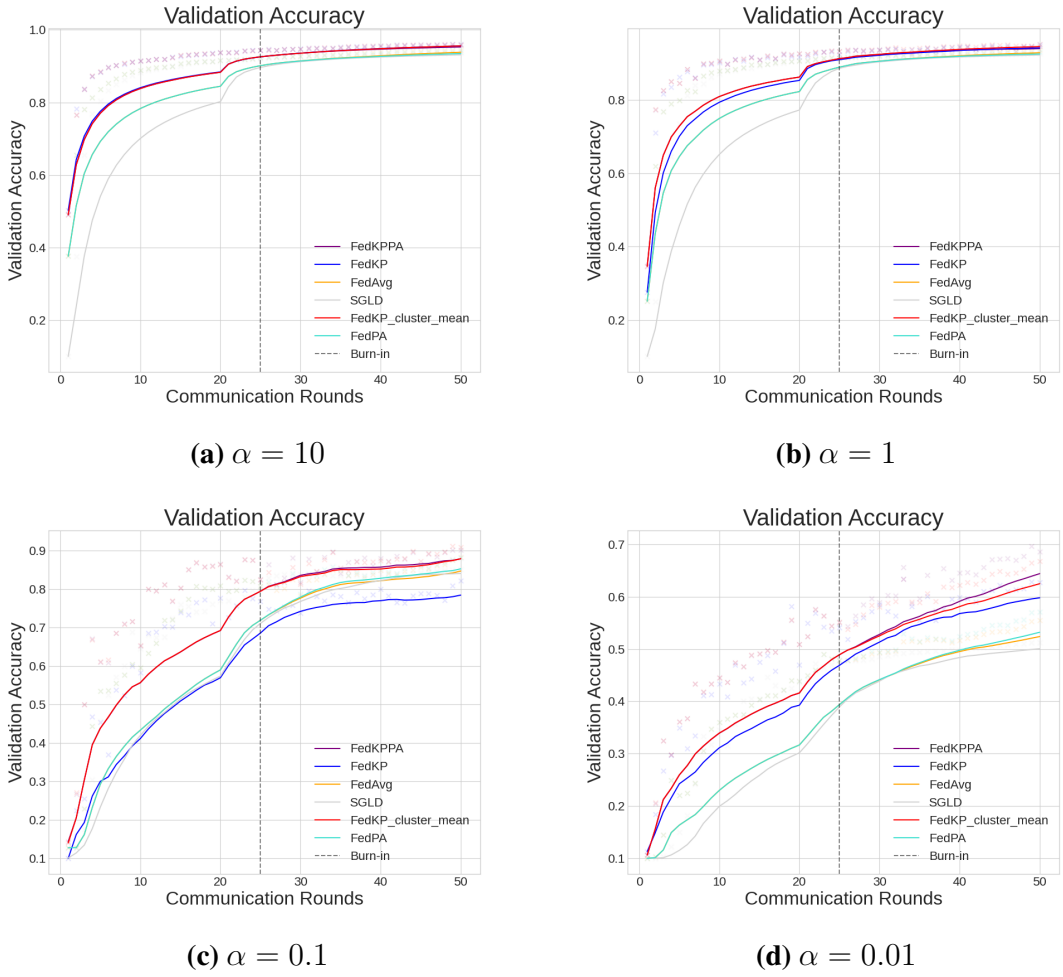
5 | Results

The main results of this thesis will be presented separately for each of the three datasets. On each dataset algorithms are evaluated on accuracy, loss, average recall and average precision on the validation data and training data at each communication round. Additional results are presented in Appendix E. Results are presented as a running average over the last ten communication rounds, with the first ten rounds being the average over previous rounds. When stating that a specific algorithm passes a performance threshold, it refers to the running average passing the threshold This is the same way results are presented by Al-Shedivat et al. [1] and give a more robust indication of the performance.

5.1 MNIST

Figure 5.1 shows the validation accuracy, and figure 5.3 shows the validation loss of all algorithms on the MNIST dataset with various degrees of label skew. The amount of label skew is determined through LDA with sampling parameter α. Results are made by sampling 20 out 100 clients each communication round, with each client training 5 epochs on their local data.

Figure 5.4 presents results when varying the amount of clients participating. Figure 5.2 describe results when varying the number of epochs trained locally during each communication round. These results compare FedKP with clustering, FedKP and FedAvg on validation and training data for the clients sampled in communication round 10. Results where calculated 5 times with the standard deviation shown in black on top of the bars.



(a) $\alpha = 10$ **(b)** $\alpha = 1$

(c) $\alpha = 0.1$ **(d)** $\alpha = 0.01$

Figure 5.1: Validation accuracy of all algorithms on the MNIST dataset with varying α.

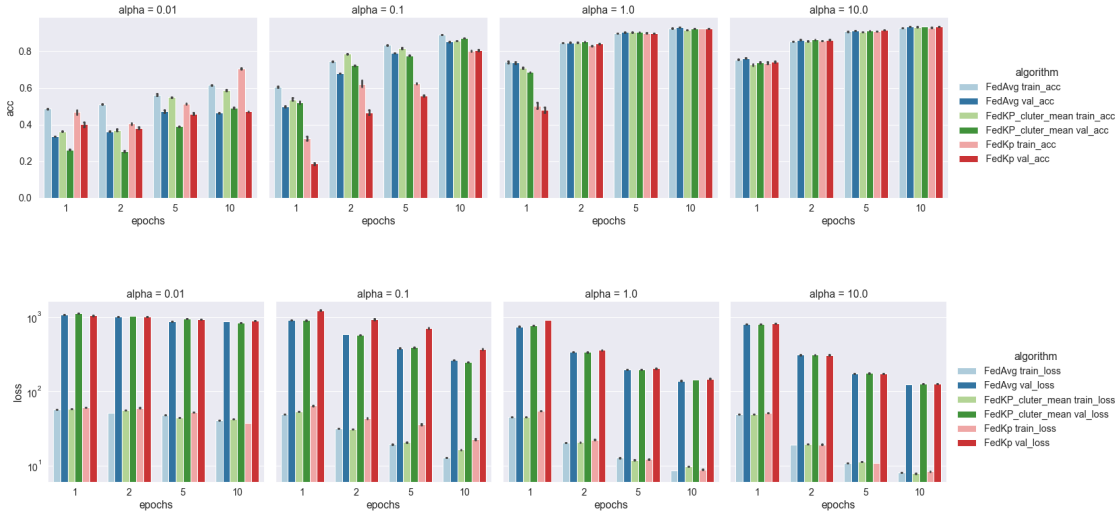
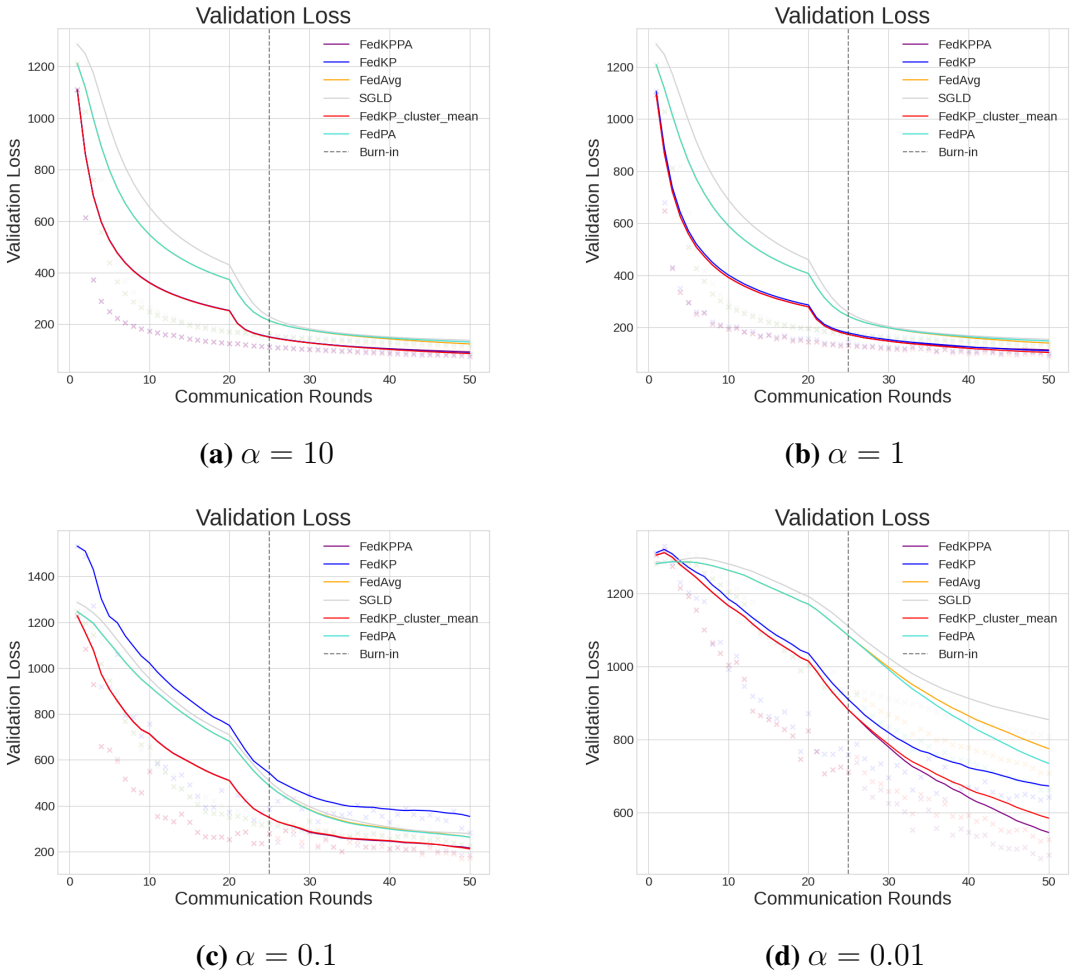


Figure 5.2: MNIST results after 10 communication rounds when varying the amount of local epochs and sampling 20 clients per round.



(a) $\alpha = 10$ **(b)** $\alpha = 1$

(c) $\alpha = 0.1$ **(d)** $\alpha = 0.01$

Figure 5.3: Validation loss of all algorithms on the MNIST dataset with varying α.

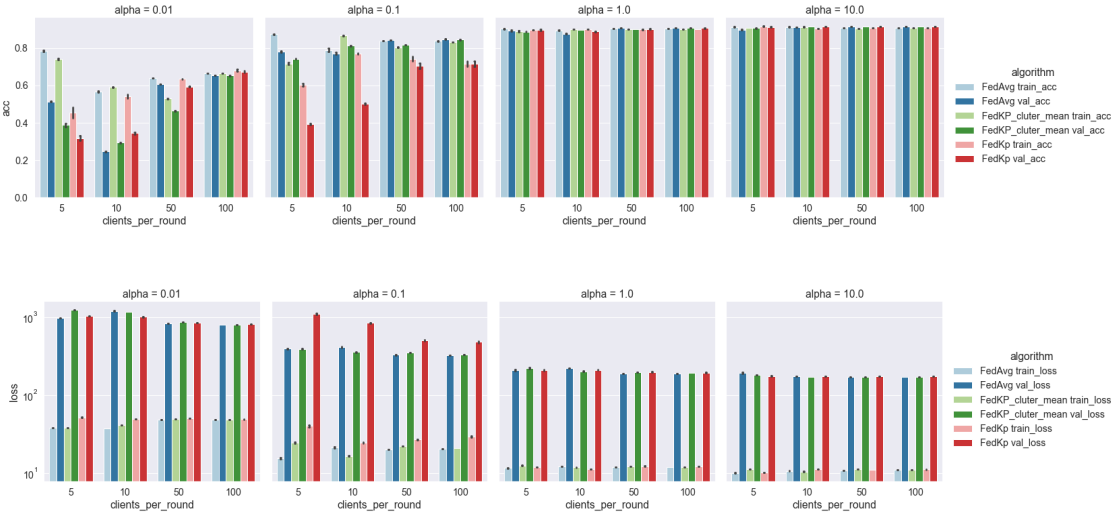


Figure 5.4: MNIST results after 10 communication rounds when varying the amount of clients sampled per round. Where each sampled client train 5 local epochs per round.

5.2 EMNIST-62

Table 5.1 shows an overview of the performance on the EMNIST-62 dataset with varying degrees of client sampling heterogeneity. The result is measured at communication rounds 600 and 1000. These rounds correspond to 100 communication rounds after the burn-in period and the final round of training. Further, we present the number of rounds for each algorithm to achieve 84% and 86% accuracy, recall, or precision. These values are taken to indicate sufficient performance, similar to what is done by Al-Shedivat et al. [1], but as can be seen recall and precision never obtain these values. Sampling was determined through Dirichlet sampling with sampling parameter α. Training is done on 100 clients and 5 local epochs per client each communication round. More detail about the hyperparameter selection is found in table 4.1.

Figure 5.5 shows the accuracy, and figure 5.6 shows the loss of all algorithms on the EMNIST-62 dataset. Observe that the left column show the aggregated server model performance on the validation data and the right column the performance on the sampled clients data. The results on training data is weighted mean proportional to the amount of data locally for each client.

	Algorithm	Round 600				Round 1000				84%				86%			
α		0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0
Accuracy	FedAvg	0.824	0.837	0.841	0.842	0.833	0.848	0.852	0.852	-	671	559	561	-	-	-	-
	FedKP	0.782	0.792	0.793	0.791	0.787	0.803	0.797	0.798	-	-	-	-	-	-	-	-
	FedKPPA	**0.838**	**0.855**	**0.861**	**0.862**	**0.840**	**0.861**	**0.863**	**0.864**	**838**	**271**	**262**	**250**	-	**890**	**542**	**524**
	FedKP_cm[a]	0.835	0.850	0.856	0.856	0.838	0.858	0.858	0.860	-	**271**	263	**250**	-	-	829	754
	FedPA	0.829	0.844	0.847	0.848	0.834	0.851	0.856	0.856	-	528	512	507	-	-	-	-
	SGLD	0.789	0.795	0.802	0.799	0.793	0.049	0.804	0.805	-	-	-	-	-	-	-	-
Training Accuracy	FedAvg	0.885	0.856	0.843	0.844	0.899	0.864	0.848	0.847	15	16	16	16	17	18	18	18
	FedKP	0.820	0.800	0.800	0.796	0.824	0.803	0.809	0.796	14	14	15	15	16	17	17	17
	FedKPPA	**0.936**	**0.886**	**0.866**	**0.870**	**0.945**	**0.891**	**0.872**	**0.869**	**10**	**10**	**10**	**10**	**12**	**12**	**12**	**12**
	FedKP_cm[a]	0.931	0.881	0.862	0.864	0.942	0.888	0.870	0.865	**10**	**10**	**10**	**10**	**12**	**12**	**12**	**12**
	FedPA	0.890	0.861	0.847	0.849	0.903	0.867	0.851	0.850	15	16	16	16	17	18	19	18
	SGLD	0.807	0.794	0.793	0.787	0.813	0.056	0.792	0.792	14	14	13	16	16	16	15	18
Precision	FedAvg	0.728	0.748	0.753	0.755	0.740	0.762	0.768	0.769	-	-	-	-	-	-	-	-
	FedKP	0.693	0.701	0.708	0.710	0.698	0.710	0.714	0.717	-	-	-	-	-	-	-	-
	FedKPPA	**0.742**	**0.779**	**0.790**	**0.794**	**0.745**	**0.788**	**0.794**	**0.805**	-	-	-	-	-	-	-	-
	FedKP_cm[a]	0.739	0.768	0.775	0.777	0.742	0.778	0.782	0.786	-	-	-	-	-	-	-	-
	FedPA	0.735	0.761	0.764	0.771	0.742	0.769	0.779	0.783	-	-	-	-	-	-	-	-
	SGLD	0.700	0.697	0.705	0.704	0.713	0.049	0.732	0.722	-	-	-	-	-	-	-	-
Recall	FedAvg	0.695	0.716	0.723	0.721	0.706	0.730	0.738	0.736	-	-	-	-	-	-	-	-
	FedKP	0.615	0.623	0.635	0.626	0.627	0.639	0.642	0.639	-	-	-	-	-	-	-	-
	FedKPPA	**0.709**	0.734	0.740	**0.743**	0.711	0.741	0.747	0.749	-	-	-	-	-	-	-	-
	FedKP_cm[a]	**0.709**	**0.735**	**0.741**	**0.743**	**0.712**	**0.744**	**0.751**	**0.751**	-	-	-	-	-	-	-	-
	FedPA	0.698	0.717	0.722	0.721	0.704	0.727	0.733	0.733	-	-	-	-	-	-	-	-
	SGLD	0.616	0.630	0.628	0.626	0.618	0.016	0.626	0.630	-	-	-	-	-	-	-	-

[a]Subscript cm denotes cluster-mean, described in algorithm 10

Table 5.1: A summary of the performance of the algorithms with respect to accuracy, average recall and average precision on the EMNIST-62 dataset subject to different selections of α. The columns indicate the performance after 600 and 1000 communication rounds as well as the number of round until the algorithms reached a specific threshold.

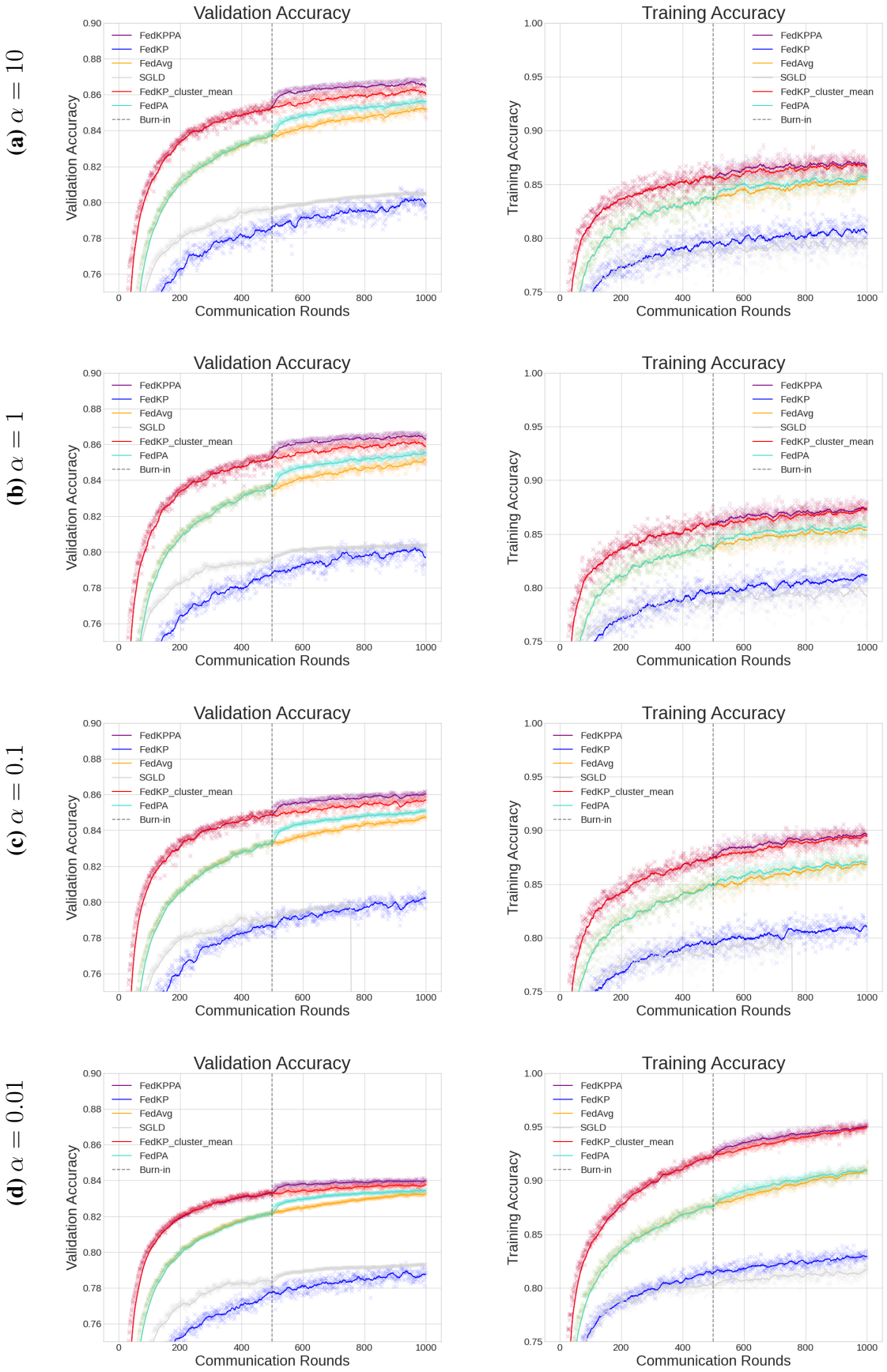


Figure 5.5: Accuracy of all algorithms on the EMNIST-62 dataset with various α.

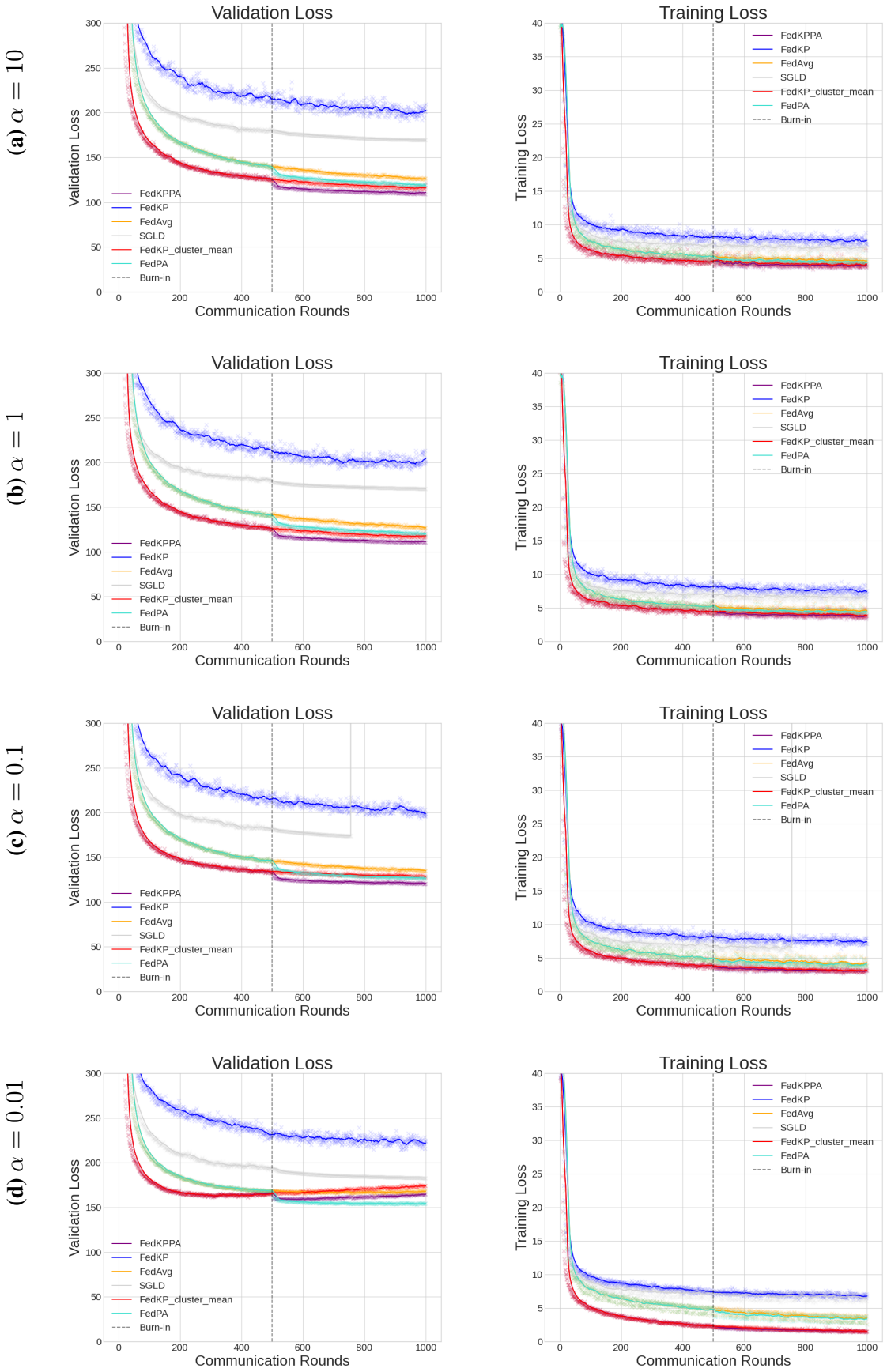


Figure 5.6: Loss of all algorithms on the EMNIST-62 dataset with various α.

5.3 CIFAR-100

Table 5.2 give an overview of the performance for varying degrees of heterogeneity. The result is measured at communication rounds 600 and 800 which correspond to 100 communication rounds after the burn-in period and the final round of training. Further, we present the number of rounds for each algorithm to achieve 30% and 40% accuracy, recall, or precision. These values are taken to indicate sufficient performance, similar to what is done by Al-Shedivat et al. [1]. The amount of label skew was determined through hierarchical LDA with sampling parameter α. Results are made by sampling 20 out of 500 clients in each communication round, with clients training 20 epochs on their local data. More detail about the hyperparameter selection is found in table 4.1.

Figure 5.7 shows the accuracy, and figure 5.8 shows the loss of all algorithms on the CIFAR-100 dataset across communication rounds with various degrees of coarse label skew. Observe that the left column show the aggregated server model performance on the validation data and the right column the performance on the sampled clients data.

	Algorithm	Round 600				Round 800				30%				40%			
α		0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0	0.01	0.1	1.0	10.0
Accuracy	FedAvg	**0.267**	0.378	0.418	0.419	**0.271**	0.389	0.433	0.434	-	182	107	104	-	-	460	460
	FedKP	0.240	0.383	0.420	0.427	0.254	0.394	0.436	0.439	-	125	85	83	-	-	444	400
	FedKPPA	0.010	0.291	0.433	0.442	0.010	0.233	0.438	0.451	-	125	74	**67**	-	-	**284**	290
	FedKP_cm[a]	0.259	**0.402**	**0.438**	**0.446**	0.265	**0.414**	**0.448**	**0.457**	-	**122**	**68**	73	-	**588**	290	**288**
	FedPA	0.213	0.340	0.410	0.413	0.010	0.335	0.412	0.423	-	170	115	102	-	-	423	430
	SGLD	0.151	0.256	0.297	0.299	0.155	0.264	0.302	0.303	-	-	641	608	-	-	-	-
Training Accuracy	FedAvg	0.433	0.572	0.620	0.613	0.459	0.619	0.669	0.664	166	76	56	53	449	199	141	145
	FedKP	0.462	0.614	0.603	0.605	0.469	0.636	0.631	0.635	76	32	29	31	275	101	79	88
	FedKPPA	0.010	0.393	0.628	0.653	0.006	0.294	0.639	0.671	76	32	**25**	**25**	212	93	**60**	**60**
	FedKP_cm[a]	**0.543**	**0.704**	**0.745**	**0.746**	**0.565**	**0.744**	**0.788**	**0.789**	**75**	**30**	**25**	**25**	**211**	**84**	62	**60**
	FedPA	0.262	0.440	0.550	0.558	0.006	0.426	0.568	0.586	146	63	56	51	-	178	144	145
	SGLD	0.145	0.259	0.307	0.307	0.162	0.262	0.311	0.309	-	-	506	493	-	-	-	-
Precision	FedAvg	0.288	0.371	0.412	0.415	0.282	0.386	0.426	0.427	-	180	120	110	-	-	478	491
	FedKP	0.295	0.387	0.416	0.426	0.300	0.400	0.434	0.439	411	104	81	81	-	765	460	388
	FedKPPA	0.010	0.305	0.434	0.440	0.010	0.243	0.437	0.448	405	106	68	**65**	-	-	**283**	**295**
	FedKP_cm[a]	**0.312**	**0.407**	**0.436**	**0.445**	**0.313**	**0.420**	**0.445**	**0.453**	**258**	**99**	**60**	70	-	**563**	290	299
	FedPA	0.234	0.353	0.407	0.408	0.010	0.337	0.408	0.417	472	168	121	108	-	-	445	463
	SGLD	0.151	0.255	0.293	0.298	0.167	0.281	0.314	0.315	-	-	687	622	-	-	-	-
Recall	FedAvg	**0.267**	0.378	0.418	0.419	**0.271**	0.389	0.433	0.434	-	182	107	104	-	-	460	460
	FedKP	0.240	0.383	0.420	0.427	0.254	0.394	0.436	0.439	-	125	85	83	-	-	444	400
	FedKPPA	0.010	0.291	0.433	0.442	0.010	0.233	0.438	0.451	-	125	74	**67**	-	-	**284**	290
	FedKP_cm[a]	0.259	**0.402**	**0.438**	**0.446**	0.265	**0.414**	**0.448**	**0.457**	-	**122**	**68**	73	-	**588**	290	**288**
	FedPA	0.213	0.340	0.410	0.413	0.010	0.335	0.412	0.423	-	170	115	102	-	-	423	430
	SGLD	0.151	0.256	0.297	0.299	0.155	0.264	0.302	0.303	-	-	641	608	-	-	-	-

[a] Subscript cm denotes cluster-mean, described in algorithm 10

Table 5.2: A summary of the performance with respect to accuracy, average recall and average precision on the CIFAR-100 dataset subject to different selections of α. The columns indicate the performance after 600 and 800 communication rounds as well as the number of round until algorithms reached the 30% and 40% thresholds.

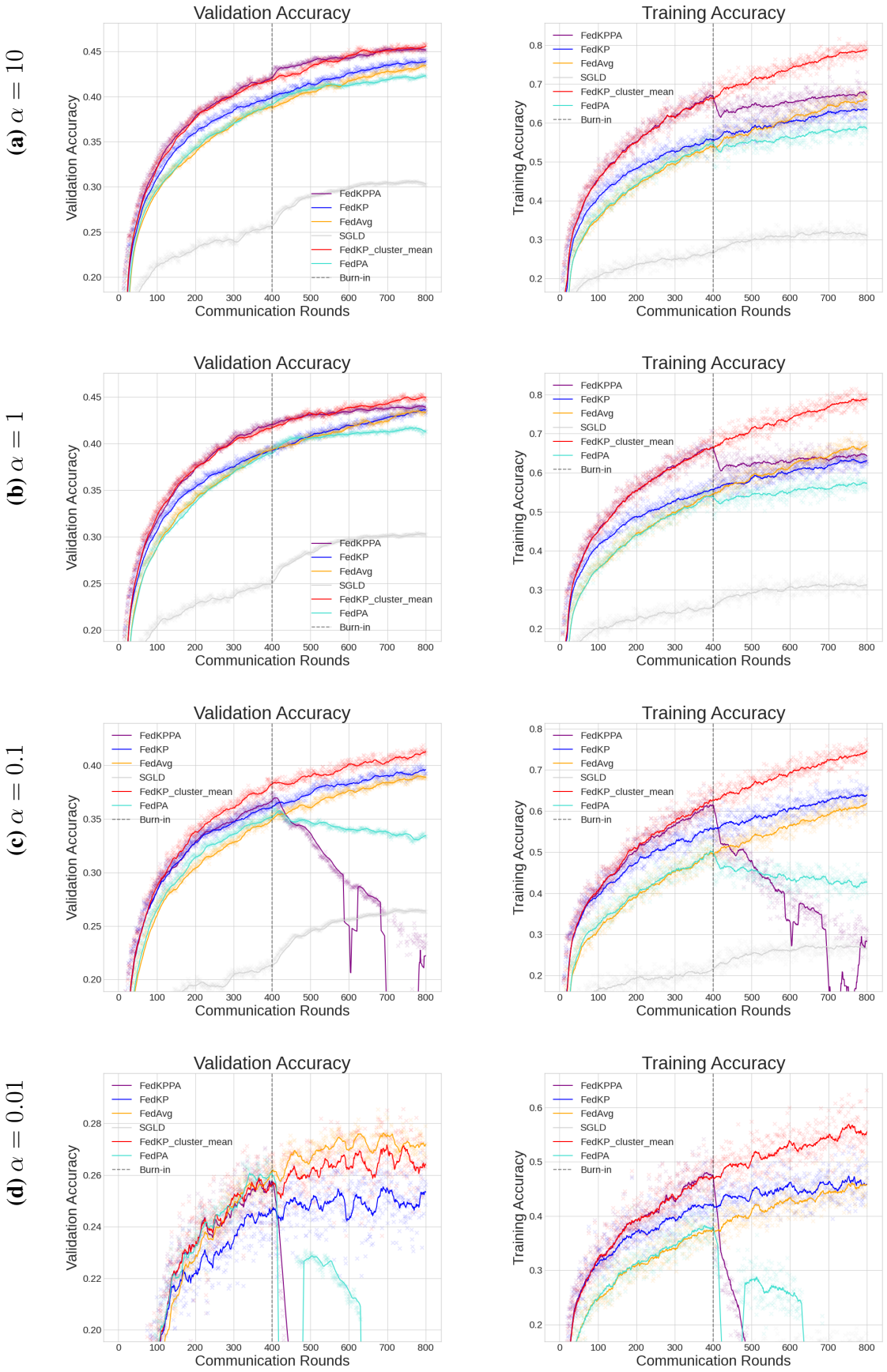


Figure 5.7: Accuracy of all algorithms on the CIFAR-100 dataset with various α.

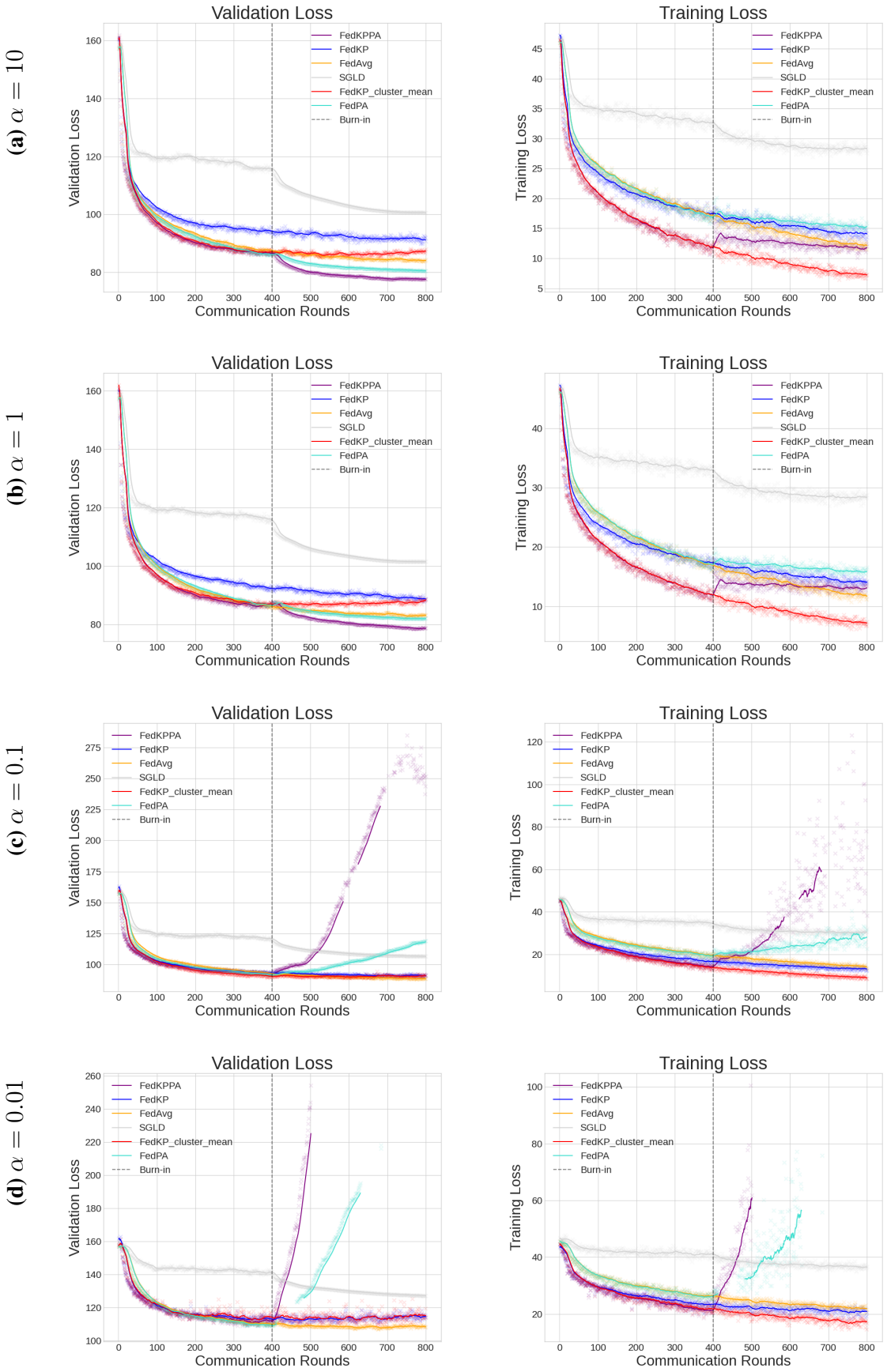


Figure 5.8: Loss of all algorithms on the CIFAR-100 dataset with various α.

6 Discussion

This chapter analyzes and discusses the results presented in chapter 5. This is done in five major subsections beginning with a general performance discussion over the different datasets, followed by hyperparameter selection, heterogeneity in data, and lastly algorithm-specific discussion concerning FedKP, FedPA, and SGLD.

6.1 General discussion

Throughout all experiments on MNIST, CIFAR-100, and EMNIST-62 one of the FedKP variants outperformed the other algorithms, both in the homogeneous and heterogeneous case. Specifically, the clustered version of FedKP showed prominent performance and more robust results with respect to heterogeneity. The FedPA gradients generally boosted performance both in regular FedPA and FedKPPA; with an exception for the heterogeneous case of CIFAR-100, where the algorithm was unstable. SGLD showed poor performance for the larger models and on par performance for the MNIST model.

Across the results, it seems that FedKP has a quicker rate of convergence compared to other algorithms. The quick convergence of FedKP, with and without clustering, is promising and given that it shows up for all α indicates that it is consistent. A caveat is that the algorithms have not converged fully, and as such, it is still possible that the asymptotic convergence of FedKP is worse than that of FedAvg or FedPA. This is a consequence of the experiment, and time limitations.

The observations made on the validation data are further apparent when evaluating the algorithms on training data. On the training data one observe that the gap between FedKP, and specifically with clustering, with other algorithms is much higher compared to the validation data. In contrast, we observe that FedPA algorithms drop in performance as it modifies the resulting gradient. The problem with asymptotic convergence not being reached is further observed in the training data, where all results have yet to converge to a steady performance.

6.1.1 MNIST

Being generally considered an easy machine learning task, the initial result of this thesis show that it also is the case in a federated setting, as seen in figure 5.1. In contrast

to other datasets studied, convergence is achieved under much smaller sets of communication rounds. For the homogeneous case, all algorithms converge within 10 rounds in contrast to the heterogeneous case where 50 is not enough for full convergence. The FedKP version with clustering appears to yield the best and most consistent results for varying heterogeneity. Lastly, the use of FedPA does not show any boost in performance for neither FedPA nor FedKPPA. This could be due to covariance adjustments not being needed for good model convergence. One could potentially have used a larger shrinkage to increase the covariance adjustment.

When varying the number of local epochs and the number of clients sampled, as seen in figures 5.2 and 5.4, one observes that FedKP is favored when many clients are involved and clients train many epochs. These results support the previously proposed theory as many clients imply a better density estimate, and longer training implies a better posterior sample. It should be noted that FedKP seems to yield a smaller gap between the training and validation accuracy and loss. This is particularly noticeable for plain FedKP and likely a consequence of the MAP being a better posterior estimate for the clients sampled. These results are made on MNIST which, compared to other federated problems, is very lightweight. Further work should thereby analyze the effect of these parameters on larger examples, such as CIFAR-100 or EMNIST-62.

6.1.2 EMNIST-62

Expanding the difficulty of MNIST through more output labels, the results on EMNIST-62 do not converge at the same rate as MNIST. FedKP performs well on EMNIST-62, but as can be seen in figure 5.5, FedKP without clustering lacks in performance. Instead, FedKP with clustering consistently outperforms FedAvg on EMNIST-62. Adding FedPA gradients, i.e., FedPA and FedKPPA, boost performance after the burn-in period.

6.1.3 CIFAR-100

The CIFAR-100 problem is not as trivial as the MNIST problem. This is also the case in a federated setting. For the more extreme case of heterogeneity with $\alpha = 0.01$, all algorithms are very volatile, with FedPA being unstable. The instability of FedPA is likely due to singularities arising when calculating the inverse local covariance matrix. With the exception to $\alpha = 0.01$, FedKP, both with and without clustering, show consistent performance above FedAvg.

6.2 Hyperparameter Selection

To compare performance between algorithms one ideally should perform a hyperparameter search to maximize the performance of each algorithm. Unfortunately, we have not had the resources to do such a hyperparameter tuning. Instead, we have tried to, as close as possible, emulate the experimental setup used by Al-Shedivat et al. for FedPA and

therefore use their selection of hyperparameters. Such a hyperparameter selection should in theory favor FedPA, as these parameters were attained through hyperparameter search with FedPA performance as the objective. Still, FedKP outperforms FedPA in most results, and with clustering, it also shows to be more stable. It should be noted that the results presented are not perfectly replicating the results by Al-Shedivat et al. We noticed at a too late of a time that the server momentum used by Al-Shedivat et al. was not correctly implemented in our code, which most likely is a cause for the generally slow convergence in our experiments. Other differences that also could play a role is that we have implemented the algorithms and datasets in PyTorch rather than Tensorflow Federated, the datasets could therefore as example slightly differ.

6.2.1 Momentum

Even if no hyperparameter search was conducted, a small observation was made during implementation. Generally, a clear increase in performance was seen when momentum was implemented on the client-side. The effects were particularly apparent for FedPA and FedKP. It is hypothesized that since the resulting client weights are a moving average during training, such an average is a better estimate of the local posterior mean [37]. In theory, this should favor FedKP and FedPA, since they assume that each client weight is a local posterior mean estimate.

6.2.2 Kernel and Bandwidth

An aspect and hyperparameter not explored in these results are the choice of kernel and bandwidth to the FedKP algorithm. In this thesis, the Epanechnikov kernel was chosen due to its relation to assuming local Gaussian posteriors as described in Appendix B. The choice of Silverman bandwidth was done out of simplicity, and is a simple rule-of-thumb bandwidth. Even with a simple exploration of these hyperparameters in Appendix C, further work may study if there is a more optimal choice of kernel and other bandwidth selection methods.

6.2.3 Burn-in

Another hyperparameter that impacted the results is the choice of burn-in. It is theorized that the boost gained by FedPA is in part a consequence of taking smaller covariance adjusted gradient steps. As such the performance gained by FedPA after the burn-in is largely independent of when the burn-in ends, and if put too early, learning will be slower than other algorithms. Under such reasoning, the end of the burn-in period is most suitable when convergence is reached. In the CIFAR-100 results, the burn-in is likely too short as convergence has not been reached, leading to FedPA plateauing earlier than other algorithms. In contrast, the EMNIST-62 results have largely converged at end of burn-in, leading to a clear edge in performance in comparison to other algorithms. It is still unclear whether such a performance boost is independent of the burn-in and if it is

sustained when running further iterations. Further research and exploration are needed to understand if this is the case.

6.3 Heterogeneity

It can be seen from the results that heterogeneity in the data distribution decreases performance across all different algorithms evaluated, as can be seen in tables 5.1 and 5.2. This is most prevalent in the CIFAR-100 results but is also noticeable for EMNIST-62. Although this seems to hold for MNIST as well, it is less clear as the results for $\alpha = 0.01$ in figure 5.1d have not converged. Decreasing the learning rate may help to improve performance for continued training, but of course, this would still not be guaranteed to reach the same performance as in the homogeneous case.

Even if asymptotic convergence is not achieved it is clear that the rate of convergence drastically decreases as one increases the amount of heterogeneity. This can be seen across most of the results presented and is not unreasonable as heterogeneity would increase the amount of noise in the posterior sample.

Solving the federated problem in heterogeneous environments is significantly harder than the homogeneous case. Although tables 5.2 and 5.1 indicate that FedKP with cluster mean is more resilient to heterogeneity, it is clear that FedKP does not provide a silver bullet solution. It is hard to find a model that reflects the whole data when models are trained on very separate local problems. It is unlikely that a model would converge towards a minimum suitable for all clients. As such, the heterogeneous case of federated learning remains difficult. Further work should seek to explore how FedKP may be optimized in heterogeneous cases.

6.4 FedKP

This thesis presents two ways to infer models using kernel posteriors, mean-shifting the mean or mean-shifting clients before taking the average, denoted as clustering. The difference between these two approaches is that the clustering adjusts for multi-modal distributions, by taking an average over multiple modes. Results for uni-modal densities should therefore still be the same across the two approaches. We observe that clustering provides more robust results and better performance compared to the non-clustering variant, with some exceptions. As such it seems that a better model is not the most frequent, but an interpolation between models.

What is unclear is if a larger bandwidth would achieve a similar effect as clustering. Since the bandwidth parameter, h, determine the amount of smoothing a large h would make distributions uni-modal. FedKP generates a model closer to the average if distributions are uni-modal , which FedKP clustering already does. Tuning the size of h could therefore likely improve the performance of FedKP, and is briefly explored for MNIST in Appendix C but requires further exploration.

Observe from the training results on CIFAR-100 that FedKP is a better fit, on average, for all the sampled clients. This aligns with the purpose of FedKP selecting the most "frequent" model among the sample of client models. Likely, FedKP overfits more towards the clients sampled. The same pattern does not seem to be true for EMNIST-62. Given that FedKP select the most frequent model the sample of clients likely is more important for FedKP than for FedAvg. As the client sampling of EMNIST-62 is non-uniform this might be a reason for the reduced performance by FedKP in this scenario. The training results indicate that FedKP might be more sensitive to the client sampling than other algorithms.

Another interesting characteristic of FedKP is the speed at which it converges. For all of the FedKP versions that out-perform FedAvg, we observe the largest performance gap early on in the learning, which is most apparent in the EMNIST-62 results. Intuitively selecting a specific mode of the posterior distribution would imply a larger step during each communication round than taking the mean of the same distribution. Therefore, it is not surprising that FedKP converges faster, albeit not necessarily to the same point in the parameters space as FedAvg. Perhaps more surprising is how well it works to increase the gradient by taking the mode instead of the mean.

A curious case is that the validation loss of FedKP for CIFAR-100, as seen in figure 5.8, is poor but the algorithm still maintain a high validation accuracy. In the case of EMNIST-62, figure 5.6, the loss of FedKP is significantly worse than for FedAvg or FedPA, although in this case, the accuracy is also worse for FedKP. It could be the case that FedKP is "polarizing" in the sense that since it chooses the most likely mode of the posterior it more clearly favors specific clients. Since the categorical-cross-entropy loss, defined as (2.2), is unbounded given the output, a model that is very sure during each prediction can generate a large loss when wrong while a more balanced model generate a smaller loss for each wrong prediction.

6.5 FedPA

The initial expectation of FedPA was that it should yield good results on all datasets. Our results indicate a more nuanced expectation of FedPA where, as discussed earlier, the burn-in hyperparameter appear to be crucial to achieving better convergence. Further, our results indicate that FedPA is unstable for extreme heterogeneous data, as shown for CIFAR-100 where $\alpha = 0.01$ or $\alpha = 0.1$ in figure 5.7d and 5.7c. Curiously no similar instabilities are present when training on the EMNIST-62 dataset.

The difference in results may be due to the different types of heterogeneous behavior injected into the CIFAR-100 and EMNIST-62 experiments. The CIFAR-100 experiment concerns a label skew through label sampling and the EMNIST-62 experiment a feature skew due to adverse client sampling. It is not unreasonable to assume that FedPA handles one type of disruption well and other types poorly.

Studying the instability of FedPA, it appears to stem from exploding gradients on the clients. Analyzing FedPA and the delta computed in Equation (2.14), we notice that we

scale the delta by the inverse of the local covariance matrix. In the case of extremely heterogeneous label-skewed data between clients, it would also mean that each client holds very similar data. Theoretically, this could be problematic as the local covariance matrix becomes singular due to non-varying data. This problem may then be further exaggerated by the instability associated with the cross-entropy loss [65].

Assuming that the covariance calculation is the root of the instability shown, one can potentially mitigate the issue by adjusting the hyperparameters. In theory, setting the shrinkage parameter to 0 would remove the impact of the inverse covariance resulting in an update like

$$\Delta_i = \theta - \mu_i.$$

However, this solution is not satisfactory since the primary idea with FedPA is to adjust for covariance, which is no longer done. Setting the shrinkage close to 0 likely leads to a more stable algorithm, but finding an optimal shrinkage near zero is hard.

Another interesting result is the performance of FedPA on the training data. As can be seen from figure 5.7, the training accuracy drops, and the loss increase for FedPA after the burn-in. Likely this is a consequence of adjusting for covariance and results in less overfitted models. In contrast to FedKP, where overfitting seems like a prominent reason for good performance, the FedPA results show that not overfitting may also lead to good performance on the validation data.

In the end, FedPA presents a good theory for adjusting for covariance for the posterior. Sadly it is hard to predict the stability and if the resulting model is better in an applied situation. Therefore FedPA should be implemented with care, using a small shrinkage and be applied at a time of convergence.

6.6 SGLD

Although the SGLD algorithm performs poorly on the larger datasets it performs similar to FedAvg on the MNIST dataset. This indicates that the SGLD implementation is correct but that it perhaps does not handle large models as well as other algorithms. It would not be surprising if the noise added in SGLD for mathematical completeness cause issues when the dimensionality increase. Most papers that study decentralized versions of SGLD work on very limited examples that are more akin to toy examples than model machine learning problems.

An idea is that as dimensionality increases, backpropagation may suffer from vanishing gradients [66], making the noise introduced disproportionate for some parameters. Adding disproportionately large noise may potentially cause instabilities as seen in figure 5.5c. Instead, noise may be better introduced if not proportional to the learning rate but towards the variance observed in the posterior sample. Such an adjustment is not studied here, but would be a direction of future work if one seeks to optimize SGLD in a federated setting.

7 | Conclusion

This thesis has explored data heterogeneity in a federated setting. This was done through a Bayesian perspective, analyzing how the distribution of resulting edge node models behave for varying degrees of heterogeneity. Different algorithms have been evaluated to measure the impact of heterogeneous data in a federated setting. Further, leveraging a Bayesian perspective, an analysis of how aggregation can be improved was done.

We propose a novel algorithm called Federated Kernel Posterior, or FedKP, that leverages kernel densities to estimate the global posterior from a sample of local posteriors of participating clients. To perform model inference with respect to these kernel densities we developed two approaches:

- Mean-shifting the average model towards the nearest mode.
- Mean-shifting client weights before taking the average.

Both approaches were evaluated and compared to other algorithms on a series of datasets with varying degrees of heterogeneity.

In the experiments, FedKP shows good, and many times superior, performance compared to a baseline of FedAvg and FedPA. Specifically, FedKP shows better performance than the baseline for data where heterogeneity is primarily prevalent in the shape of a feature skew. If heterogeneity instead is a label skew, FedKP shows worse performance than the baseline. However, the altered version of FedKP that use an average of mean-shifted clients shows greater performance and robustness than both FedAvg and FedPA when data is subject to both label skew, feature skew, or non-uniform client sampling. The improved performance by FedKP is specifically large early on in the training. Asymptotically it is not clear if FedKP converges to better performance than FedAvg.

We believe there is much work that can be done on FedKP, and potential areas include,

- Intuitively, the speed of FedKP early on likely is a result of favoring specific clients. However, what impact does this choice have asymptotically?
- In the experiments, we have used stochastic gradient descent with momentum on the server. How do this choice and the hyperparameter choices on the server impact the convergence of FedKP?

- Throughout the thesis, we have used an Epanechnikov kernel. However, during testing a Gaussian kernel also showed promising results. How does the choice of kernel impact the performance?
- A preliminary study of bandwidth size/scaling in Appendix C shows the potential impact of the hyperparameter. Further research should be done to optimize the bandwidth for federated settings.
- Going outside the scope of this thesis, how may FedKP be used in for instance a decentralized setting would also be of interest.

We would also like to mention how weighting could be implemented in FedKP. The straightforward approach would be through a weighted kernel density

$$\ell(\theta) = \frac{1}{Nh\sum_{i\in S}|\mathcal{D}_i|}\sum_{i\in S}|\mathcal{D}_i|K\big(\frac{\theta-\theta_i}{h}\big).$$

This would make the FedKP put more likelihood for parameters centered around client with a lot of data. Another approach would be by scaling the bandwidth proportional towards the amount of data observed as

$$\ell(\theta) = \frac{\sum_{i\in S}|\mathcal{D}_i|}{N^2h}\sum_{i\in S}K(\frac{\theta-\theta_i}{h_i}), \quad h_i = \frac{h}{|\mathcal{D}_i|}.$$

By weighting the bandwidth this way, "uncertain" clients trained on little data would have a larger bandwidth associated and the opposite for clients with large data. Further work may look into other weighting schemes when the local data sizes differ.

Bibliography

[1] M. Al-Shedivat, J. Gillenwater, E. Xing, and A. Rostamizadeh, "Federated learning via posterior averaging: A new perspective and practical algorithms," *arXiv e-prints*, pp. arXiv–2010, 2020.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[3] M. Chui, B. Hall, A. Singla, and A. Sukharevsky, "The state of AI in 2021," *McKinsey & Co. https://www. mckinsey. com/business-func tions/mckinsey-analytics/our-in sights/global-survey-the-state-of-ai-in-2021 (Accessed: April 22, 2022)*.

[4] J. Desjardins, "How much data is generated each day?| World Economic Forum," in *World Economic Forum, accessed September*, vol. 28, p. 2019, 2019.

[5] M. Goddard, "The eu general data protection regulation (GDPR): European regulation that has a global impact," *International Journal of Market Research*, vol. 59, no. 6, pp. 703–705, 2017.

[6] H. B. McMahan *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1, 2021.

[7] T. Duong and M. L. Hazelton, "Convergence rates for unconstrained bandwidth matrix selectors in multivariate kernel density estimation," *Journal of Multivariate Analysis*, vol. 93, no. 2, pp. 417–433, 2005.

[8] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-iid data," in *International Conference on Learning Representations*, 2019.

[9] A. T. Thorgeirsson and F. Gauterin, "Probabilistic predictions with federated learning," *Entropy*, vol. 23, no. 1, p. 41, 2020.

[10] H.-Y. Chen and W.-L. Chao, "FedBE: Making Bayesian model ensemble applicable to federated learning," *arXiv e-prints*, pp. arXiv–2009, 2020.

[11] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[14] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv e-prints*, pp. arXiv–1609, 2016.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv e-prints*, pp. arXiv–1412, 2014.

[16] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[18] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine learning*, vol. 50, no. 1, pp. 5–43, 2003.

[19] R. Christensen, W. Johnson, A. Branscum, and T. E. Hanson, *Bayesian ideas and data analysis: an introduction for scientists and statisticians*. CRC press, 2011.

[20] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*, pp. 1613–1622, PMLR, 2015.

[21] A. Hard, C. M. Kiddon, D. Ramage, F. Beaufays, H. Eichner, K. Rao, R. Mathews, and S. Augenstein, "Federated learning for mobile keyboard prediction," 2018.

[22] A. T. Thorgeirsson, S. Scheubner, S. Fünfgeld, and F. Gauterin, "Probabilistic prediction of energy demand and driving range for electric vehicles with federated learning," *IEEE Open Journal of Vehicular Technology*, vol. 2, pp. 151–161, 2021.

[23] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, and S. Srikanteswara, "Energy demand prediction with federated learning for electric vehicle networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[24] A. Lalitha, O. Cihan Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," *arXiv e-prints*, pp. arXiv–1901, 2019.

[25] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan, *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[26] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, vol. 3, p. 3, 2018.

[27] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv e-prints*, pp. arXiv–1909, 2019.

[28] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv e-prints*, pp. arXiv–2003, 2020.

[29] M. Al-Shedivat, *Principles of Learning in Multitask Settings: A Probabilistic Perspective*. PhD thesis, National Institute of Health, 2021.

[30] A. Lalitha, X. Wang, O. Kilinc, Y. Lu, T. Javidi, and F. Koushanfar, "Decentralized Bayesian learning over graphs," *arXiv e-prints*, pp. arXiv–1905, 2019.

[31] X. Zhang, F. Fang, and J. Wang, "Probabilistic solar irradiation forecasting based on variational Bayesian inference with secure federated learning," *IEEE Transactions on Industrial Informatics*, 2020.

[32] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *International Conference on Machine Learning*, pp. 7252–7261, PMLR, 2019.

[33] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *arXiv e-prints*, pp. arXiv–2002, 2020.

[34] S. Ahn, B. Shahbaba, and M. Welling, "Distributed stochastic gradient MCMC," in *International conference on machine learning*, pp. 1044–1052, PMLR, 2014.

[35] K. El Mekkaoui, D. Mesquita, P. Blomstedt, and S. Kaski, "Federated stochastic gradient Langevin dynamics," in *Uncertainty in Artificial Intelligence*, pp. 1703–1712, PMLR, 2021.

[36] O. Ledoit and M. Wolf, "A well-conditioned estimator for large-dimensional covariance matrices," *Journal of multivariate analysis*, vol. 88, no. 2, pp. 365–411, 2004.

[37] S. Mandt, M. D. Hoffman, and D. M. Blei, "Stochastic gradient descent as approximate Bayesian inference," *Journal of Machine Learning Research 18*, pp. 1–35, 2017.

[38] S. L. Scott, A. W. Blocker, F. V. Bonassi, H. A. Chipman, E. I. George, and R. E. McCulloch, "Bayes and big data: The consensus Monte Carlo algorithm," *International Journal of Management Science and Engineering Management*, vol. 11, no. 2, pp. 78–88, 2016.

[39] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, Citeseer, 2011.

[40] J. Besag, "Comments on "representations of knowledge in complex systems" by u. grenander and mi miller," *J. Roy. Statist. Soc. Ser. B*, vol. 56, no. 591-592, p. 4, 1994.

[41] D. Liu and O. Simeone, "Wireless federated Langevin Monte Carlo: Repurposing channel noise for Bayesian sampling and privacy," *arXiv e-prints*, pp. arXiv–2108, 2021.

[42] M. Vono, V. Plassier, A. Durmus, A. Dieuleveut, and E. Moulines, "QLSD: Quantised Langevin stochastic dynamics for Bayesian federated learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 6459–6500, PMLR, 2022.

[43] W. Deng, Y.-A. Ma, Z. Song, Q. Zhang, and G. Lin, "On convergence of federated averaging Langevin dynamics," *arXiv e-prints*, pp. arXiv–2112, 2021.

[44] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv e-prints*, pp. arXiv–1909, 2019.

[45] B. A. Turlach, "Bandwidth selection in kernel density estimation: A review," in *CORE and Institut de Statistique*, Citeseer, 1993.

[46] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," *Theory of Probability & Its Applications*, vol. 14, no. 1, pp. 153–158, 1969.

[47] R. A. Davis, K.-S. Lii, and D. N. Politis, "Remarks on some nonparametric estimates of a density function," in *Selected Works of Murray Rosenblatt*, pp. 95–100, Springer, 2011.

[48] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.

[49] Y.-p. Mack and B. W. Silverman, "Weak and strong uniform consistency of kernel regression estimates," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 61, no. 3, pp. 405–415, 1982.

[50] M. C. Jones, J. S. Marron, and S. J. Sheather, "A brief survey of bandwidth selection for density estimation," *Journal of the American statistical association*, vol. 91, no. 433, pp. 401–407, 1996.

[51] C. Louizos, M. Reisser, J. Soriaga, and M. Welling, "An expectation-maximization perspective on federated learning," *arXiv e-prints*, pp. arXiv–2111, 2021.

[52] R. Kassab and O. Simeone, "Federated generalized Bayesian learning via distributed stein variational gradient descent," *IEEE Transactions on Signal Processing*, 2022.

[53] H. Samawi, H. Rochani, J. Yin, D. Linder, and R. Vogel, "Notes on kernel density based mode estimation using more efficient sampling designs," *Computational Statistics*, vol. 33, no. 2, pp. 1071–1090, 2018.

[54] J. C. Lee, J. Li, C. Musco, J. M. Phillips, and W. M. Tai, "Finding the mode of a kernel density estimate," *arXiv e-prints*, pp. arXiv–1912, 2019.

[55] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[56] S. Banach, "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales," *Fund. math*, vol. 3, no. 1, pp. 133–181, 1922.

[57] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1197–1203, IEEE, 1999.

[58] M. A. Carreira-Perpinan, "Gaussian mean-shift is an EM algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 5, pp. 767–776, 2007.

[59] A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani, "Gaussian process behaviour in wide deep neural networks," *arXiv e-prints*, pp. arXiv–1804, 2018.

[60] H. Fischer, *A history of the central limit theorem: From classical to modern probability theory*. Springer, 2011.

[61] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[62] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, IEEE, 2017.

[63] S. Caldas, S. Meher Karthik Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," *arXiv e-prints*, pp. arXiv–1812, 2018.

[64] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[65] A. Ghosh, N. Manwani, and P. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, vol. 160, pp. 93–107, 2015.

[66] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[67] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 53, no. 3, pp. 683–690, 1991.

[68] J. Liao, Y. Wu, and Y. Lin, "Improving Sheather and Jones' bandwidth selector for difficult densities in kernel density estimation," *Journal of nonparametric statistics*, vol. 22, no. 1, pp. 105–114, 2010.

[69] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[70] P. D. Gerard and W. R. Schucany, "Local bandwidth selection for kernel estimation of population densities with line transect sampling," *Biometrics*, vol. 55, no. 3, pp. 769–773, 1999.

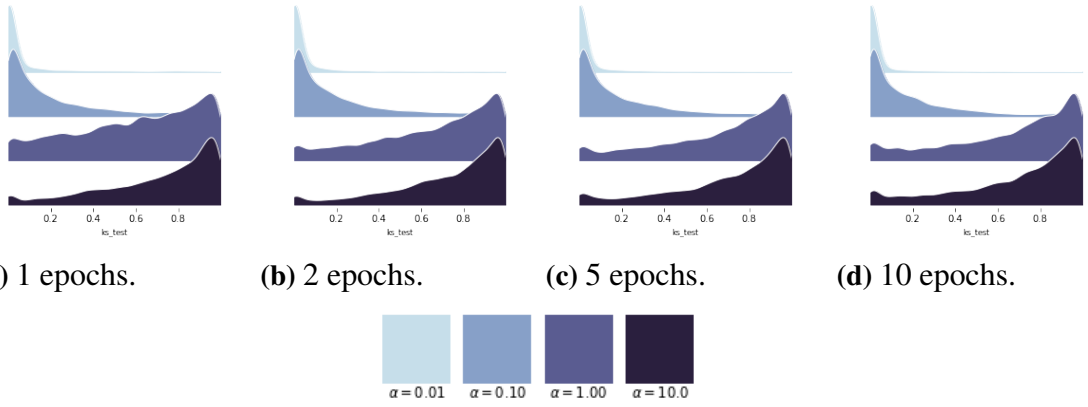
[71] J. G. Staniswalis, "Local bandwidth selection for kernel estimates," *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 284–288, 1989.

A | Normality in non-IID Settings

In this chapter, we seek to explore how heterogeneity affects the distributions of the posterior samples. To aid our understanding we performed experiments on MNIST. We use FedAvg and the same hyperparameters as in the main results, seen in table 4.1. Using these hyperparameters and varying the number of epochs, the number of clients sampled, and the number of communication rounds we analyze the normality of the posterior sample distribution through KS-tests as in figure 3.1b. In conclusion, as is explored below, no asymptotic behavior between these parameters and the amount of Gaussian distributions was found.

A.1 Number of Epochs

Studying how the amount of training impacts the normality of the posterior sample one observes, in figure A.1, that there is no relation between the number of epochs and data being less Gaussian. It may be the case that 1 epoch of training is sufficient for overfitting. But results do reflect a clear difference between the IID and non-IID scenarios.

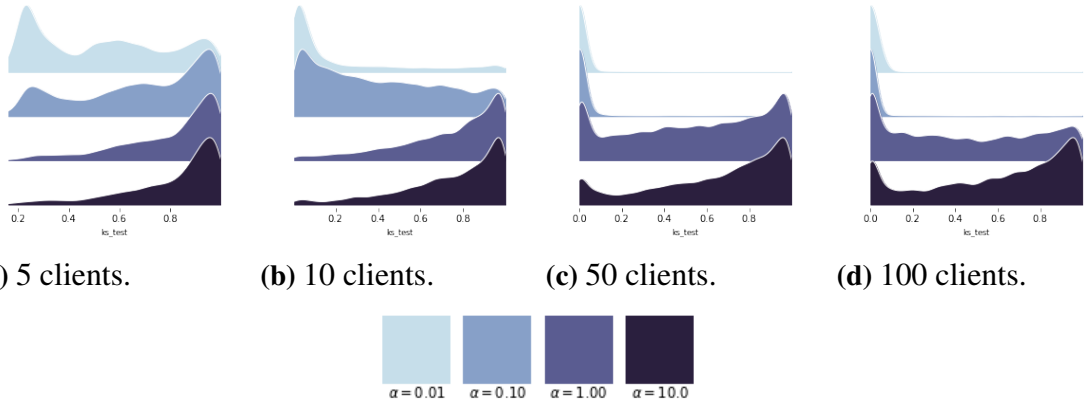


(a) 1 epochs. **(b)** 2 epochs. **(c)** 5 epochs. **(d)** 10 epochs.

Figure A.1: KS-Test distribution density for all model weights using different amount of local using 5 local epochs and 1 round of training.

A.2 Amount of Clients

Analyzing how the number of clients impacts the posterior sample normality, seen in figure A.2, one observes that increasing the number of clients leads to less Gaussian behavior. In some sense, one may expect that as we increase the number of posterior samples there is an asymptotic behavior toward normality. In this short example, that does not seem the case but an opposite relationship appears.

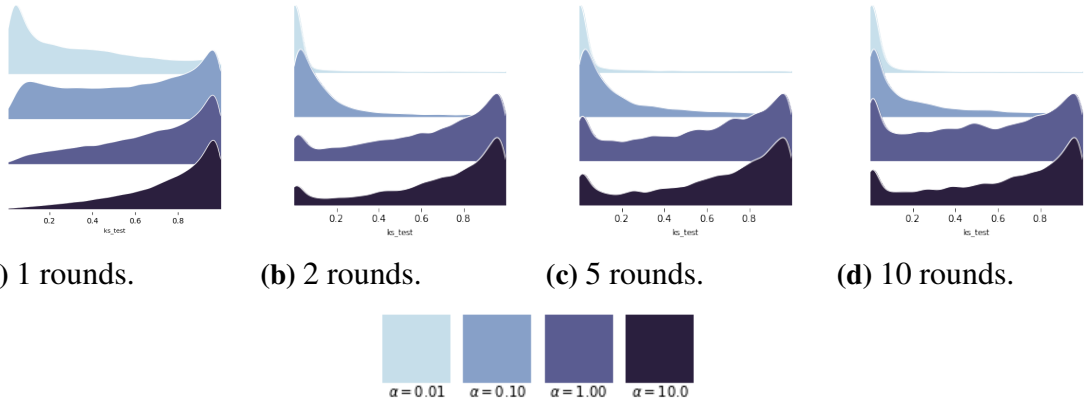


(a) 5 clients. **(b)** 10 clients. **(c)** 50 clients. **(d)** 100 clients.

Figure A.2: KS-Test density for different clients from same initial model using 5 local epochs and a single round of training.

A.3 Amount of Rounds

Lastly, one may expect that Gaussian behavior appears as the federated learning model converges. Using FedAvg, figure A.3 shows the distribution of p-values after different numbers of communication rounds. Observe that the resulting posterior distributions are less Gaussian as the number of rounds increases. Likely this is caused by clients overfitting towards local data, even when the global model has converged.



(a) 1 rounds. **(b)** 2 rounds. **(c)** 5 rounds. **(d)** 10 rounds.

Figure A.3: KS-Test density of weights after 1,2,5,10 rounds of FedAvg using 5 local epochs and 20 out 100 clients each round.

B | FedKP a Regularization of FedAvg

This chapter intends to show a connection between FedAvg and FedKP using the Bayesian federated learning problem. Given the same assumptions as in the derivation of FedAvg, we will show that FedKP can be viewed as a regularization of FedAvg. We also show that, for a specific choice of bandwidth, FedKP is the same as FedAvg and why an Epanechnikov kernel is a reasonable choice of kernel.

With the Bayesian federated problem defined in (2.11) as

$$\begin{aligned} \underset{\theta}{\operatorname{argmax}} \quad \ell(\theta|\mathcal{D}) &= log(p(\theta)) + \sum_{i \in \mathcal{S}} f_i(\theta) \\ f_i(\theta) &= C_i + log(\mathcal{L}(\mathcal{D}_i|\theta)). \end{aligned} \tag{B.1}$$

Assuming locally Gaussian likelihoods with mean θ_i and standard deviation σ we arrive at the log-likelihood

$$f_i(\theta) = \hat{C}_i - \Big(\frac{||\theta - \theta_i||}{\sqrt{2}\sigma}\Big)^2, \tag{B.2}$$

where $\hat{C}_i = C_i + log(\frac{1}{\sqrt{2}\sigma})$. We observe that this expression has a similarity to an Epanechnikov kernel [46] defined as

$$K_h(u) = \begin{cases} \frac{3}{4h}(1 - ||\frac{u}{h}||^2), & ||u|| \leq h \\ 0, & ||u|| > h \end{cases}$$

In fact, let $h \in \mathbb{R}^+$ and define $\tilde{C}_i = \frac{4}{3}2\sigma^2\frac{C_i}{h^2} - \frac{3}{4h}$ we have the local log-likelihood

$$\begin{aligned} f_i(\theta) &= \frac{4h^2}{6\sigma^2}\Big(\tilde{C}_i + \frac{3}{4}\Big(1 - \Big|\Big|\frac{\theta - \theta_i}{h}\Big|\Big|^2\Big)\Big) \\ \theta \in \mathcal{I}_i &= \{||\theta - \theta_i|| \leq h\}. \end{aligned} \tag{B.3}$$

Observe that the second term in the parenthesis of equation (B.3) equals an Epananchikov kernel with $u = ||\theta - \theta_i||$. The local log-likelihood can thus be written as

$$f_i(\theta) = \frac{4h^2}{6\sigma^2}\Big(\tilde{C}_i + K_h(\theta - \theta_i)\Big), \quad \theta \in \mathcal{I}_i. \tag{B.4}$$

We then have as a result the global log-likelihood as

$$\begin{aligned} \ell(\theta|\mathcal{D}) &= log(p(\theta)) + \sum_{i \in Q} f_i(\theta) \\ f_i(\theta) &= \frac{4h}{6\sigma^2}\Big(\tilde{C}_i + K_h(\theta - \theta_i)\Big), \end{aligned} \tag{B.5}$$

With a flat prior $p(\theta) = c$ and letting $C = \frac{6\sigma^2}{4h}c + \sum_{i \in S}(\tilde{C}_i)$ we have the global log likelihood

$$\begin{aligned} \ell(\theta|\mathcal{D}) &= \frac{4h}{6\sigma^2}\Big(C + \sum_{i \in Q} K_h(\theta - \theta_i)\Big), \\ \theta \in \mathcal{I} &= \bigcap_{i \in Q} \mathcal{I}_i. \end{aligned} \tag{B.6}$$

We thus have that maximizing the log-likelihood of the global log-likelihood is analog to maximizing the kernel density as long as $\theta \in \mathcal{I}$. As $h \to \infty$ we note that $\mathcal{I}$ spans the entire feature space, and thus, the mean $\bar{\theta}$ solves the optimization problem. Therefore, for a sufficiently large h, the maximum likelihood solution of the kernel density estimation corresponds to the average of $\{\theta_i\}_{i \in \mathcal{S}}$, i.e., the FedAvg solution. For smaller values of h FedKP is a regularization of FedAvg.

C | Hyperparameter Optimization

This chapter intends to be an initial exploration of how FedKP performance depends on the choice of bandwidth. The experiment is done on MNIST using the same setup as the main results, i.e., sampling 20 out 100 clients with 5 local epochs but only studying performance after one round in an IID scenario with $\alpha = 10$. The idea is to measure how FedKP improves performance after a single round,compared to FedAvg, and how performance depends on the bandwidth. As MNIST is an easy problem, good performance is yielded after a single round, making a single round sufficient. Accounting for sampling variance, results were cross-validated 20 times with the standard deviation plotted in a shaded color.

The experiment will cover a series of different types of bandwidth selection techniques. Further, results will be measured when *scaling* the bandwidths. Scaling is done to analyze if larger bandwidth may be sensible in a federated setting. The scaling will be presented as h along the x-axis of the plots.

We will now present different bandwidth techniques one by one. After a short description, results will be presented in terms of the server accuracy and loss using the specified bandwidth technique and by scaling it according to h.

C.1 Rule of Thumb Bandwidth

The classical rule of thumb bandwidth is using normal approximation bandwidth to solve $R(f'')$ which equals the sample variance of the distribution. Using Silverman's rule [48] and estimating the variance through the sample variance or the interquartile range (IQR) as

$$h = 0.9 \min(\hat{\sigma}, \frac{IQR}{1.34}) n^{-\frac{1}{5}}. \tag{C.1}$$

This bandwidth selection is used due to its simplicity. Generally it tends to be oversmoothed for non-Gaussian distributions and show weak convergence towards the true density distribution [49]. In figure C.1 one observes the results from using the Silverman's rule and scaling the bandwidth for different h.

What is observed from these results is that scaling the bandwidth may be sensible in a

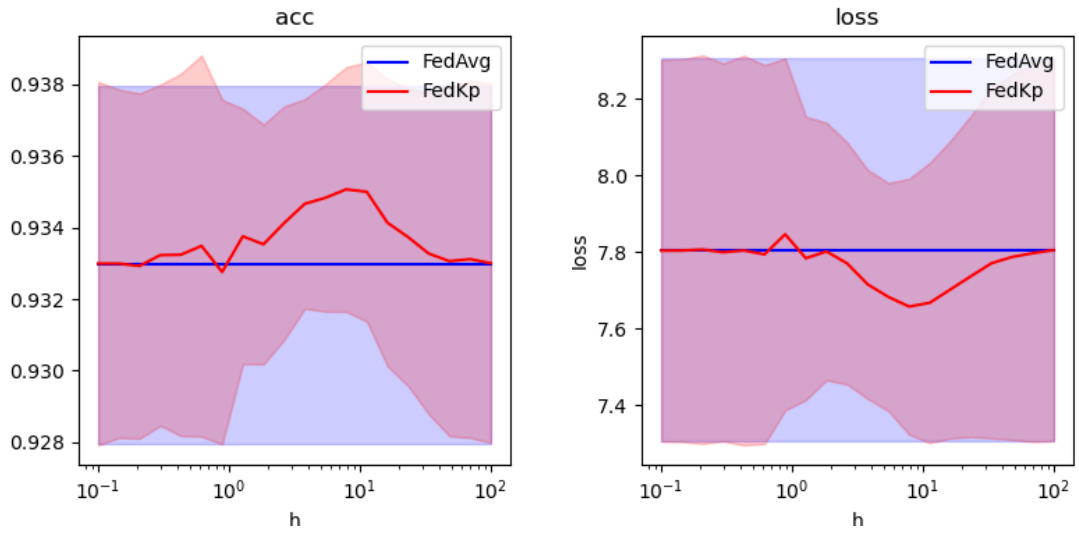


Figure C.1: Loss and accuracy's for non-IID distributed data with different bandwidth scaling h using a rule of thumb bandwidth.

federated setting. Even if the edge in performance is marginal, and not significant, there is an indication that performance may be improved by scaling the bandwidth.

C.2 Plug-in Bandwidth

Instead of assuming that f is Gaussian to calculate $R(f'')$ another approach is to take kernel density estimation one step further and estimate f'' through kernel estimation. This leads to a technique referred to as *plug-in* or *solve-the-equation* bandwidth selection techniques. In estimating f'' through kernel estimation one comes to the same problem initially but now need to estimate $R(f'''')$. This loop is referred as the *plug-in depth* which for a single specific value requires you to go with a pilot estimator for $R(f^{(2k+2)})$. Usually $k = 2$ is a sufficient depth [67]. In this work, an improved Sheater-Jones bandwidth selection method will be used [68] using $k = 2$ and a biased-adjusted pilot estimator. In figure C.1, one observes the results from using the plug-in bandwidth and scaling the bandwidth for different h.

The results provided here show marginal differences between using the plug-in and Silverman's rule. The edge provided is just as marginal and not significant. Observe that increasing the bandwidth indicates some increase in performance, but that these results are not conclusive.

C.3 Cross-val Bandwidth

A more non-analytical approach is to estimate the bandwidth through cross-validation. Using this technique the bandwidth is selected to minimize the out-of-sample likelihoods.

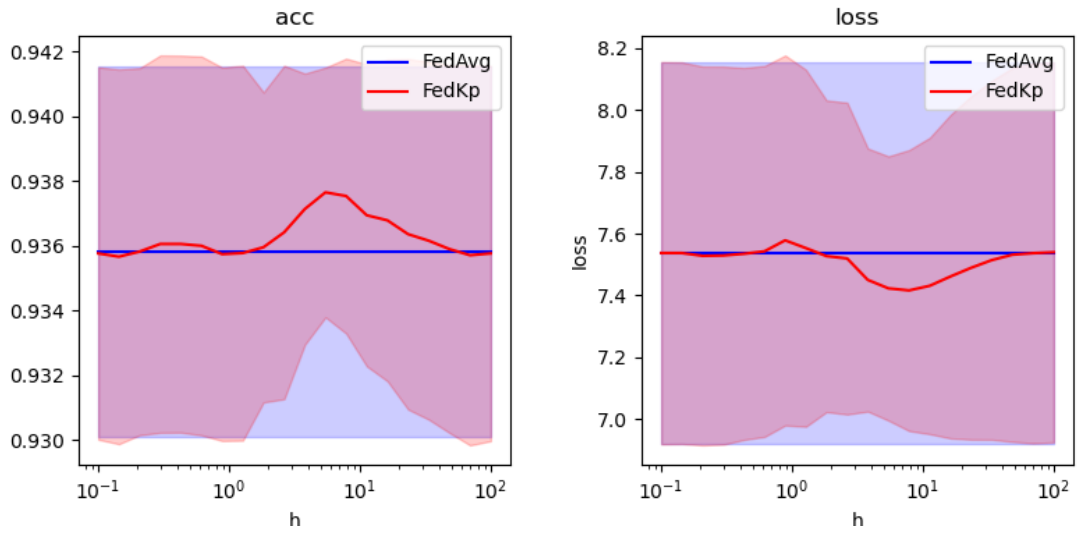


Figure C.2: Loss and accuracy's for non-IID distributed data with different bandwidth scaling h using plug-in bandwidth

This thesis will use cross-validation over 5 folds using an initial Silverman's rule to choose the $\tilde{h} \in [0.01, 0.1, 1, 10]$ minimizing the cross-validation likelihood. This method is computationally expensive which is why a small selection of scaling parameters $\tilde{h}$ was used. In figure C.3, one observes the MNIST results from running the cross-validation and scaling the bandwidth for different h.

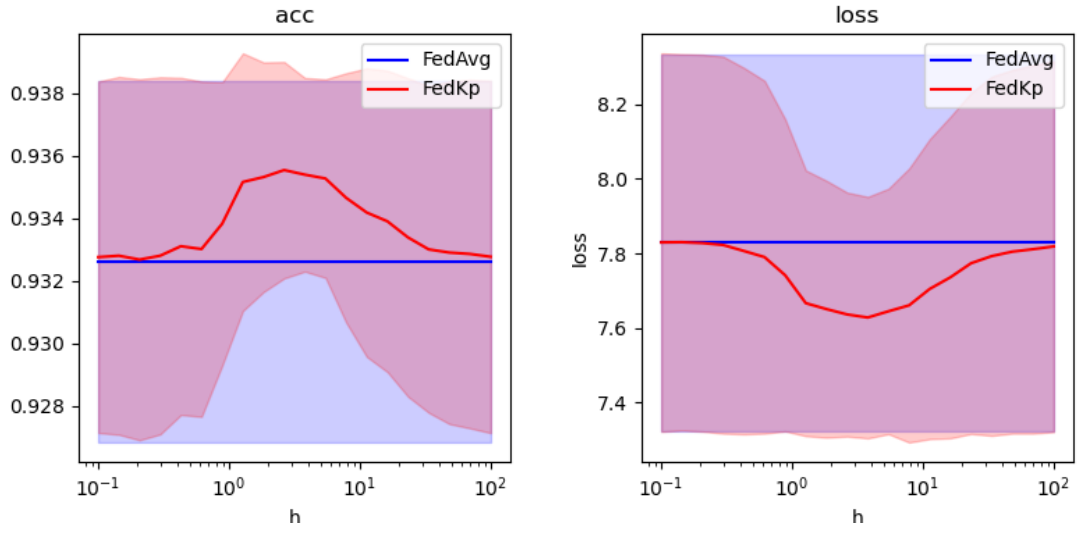


Figure C.3: Loss and accuracy's for non-IID distributed data with different bandwidth scaling h using a cross-val bandwidth selection.

Again these results are similar to the previous bandwidth-selection techniques. Likely, cross-validating with few folds makes the method largely inefficient. Sadly, using more folds would be computationally expensive.

C.4 Adaptive Plug-in Bandwidth

In the previous bandwidth selection methods, the premise is to minimize the AMISE of our global kernel estimate. During inference of FedKP, the purpose is not the best global posterior but to find the MAP estimate. It is thereby sensible to instead of minimizing the AMISE minimize the AMSE around the mode. As such a local bandwidth selection method [69] will be tested which estimates the bandwidth around the current value m_x. This will be done by estimating the variance around the nearest 5 clients using Scott's rule [69]. This technique, even if requiring calculating the bandwidth of each mean-shift iteration is not extremely computationally expensive due to the simplicity of Scott's rule and the reduction of clients calculating the bandwidth. In figure C.4, one observes the MNIST results from running the adaptive plug-in and scaling the bandwidth for different h.

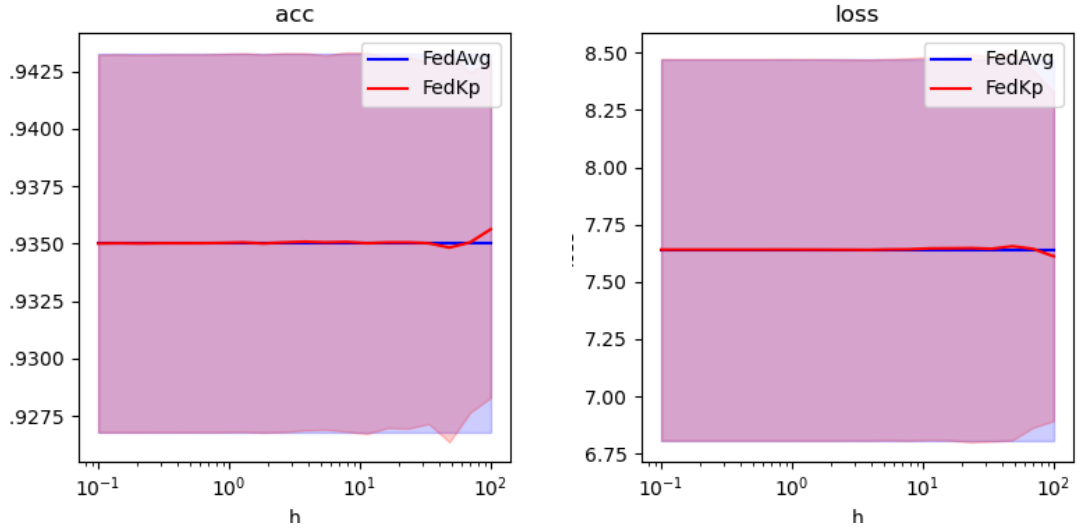


Figure C.4: Loss and accuracy's for non-IID distributed data with different bandwidth scaling h using adaptive bandwidth selection.

The results provided here are near identical with FedAvg and are not unreasonable. Only studying the closest neighbors from an initial value would decrease the estimated bandwidth. Even if these results do not provide any new insight, there is a theoretical motivation for using adaptive bandwidth, and other possible adaptive bandwidth selection methods [70, 71] may be explored in the future.

C.5 Conclusion

This short exploration of bandwidth selection techniques provides no conclusive results on how the performance of FedKP depends on the bandwidth. The results are likely inconclusive as only a single round is insufficient to show the difference between FedAvg

and FedKP. Also, the variance in performance may largely be caused by the sampling process. What this chapter shows are possible directions for further work, namely

- There are some implications that scaling the bandwidth may be suitable to improve the performance of FedKP.

  In fact, it may considered if scaling the bandwidth would result in similar performance as FedKp with clustering.

- Adaptive bandwidths may be explored further in purpose to better estimate the bandwidth around the mode of interest.

- No radical difference was seen using the plug-in or cross-validated bandwidth in comparison to Silverman's rule.

It is suggested that further work would analyze the bandwidth selection across multiple rounds of training as that may show larger difference between FedAvg and FedKP. Also, such exploration should be on larger datasets, possibly in nonIID scenarios, where the federated problem is more difficult.

D | Experiment details

This is a compliment to chapter 4, the intent is to provide enough detail of each model and dataset such that the experiment can be replicated.

Table D.1: Model architecture used for the MNIST experiments.

Name	Out Shape	Activation	Comment
Input	(28, 28, 1)		
Conv1	(28, 28, 16)	ReLU	Size = (5,5), Stride = (1,1), Padding = 2
Maxpool	(14, 14, 16)		Size = (2, 2)
Conv2	(14, 14, 32)	ReLU	Size = (5,5), Stride = (1,1), Padding = 2
Maxpool	(7 ,7 , 32)		Size = (2,2)
Flatten	1568		
Dense	10	Softmax	

Table D.2: Model architecture used for the EMNIST-62 experiments.

Name	Out Shape	Activation	Comment
Input	(28, 28, 1)		
Conv1	(26, 26, 32)	ReLU	Size = (3,3), Stride = (1,1)
Conv2	(24, 24, 64)	ReLU	Size = (3,3), Stride = (1,1)
Maxpool	(12, 12, 64)		Size = (2,2)
Dropout1	(12, 12, 64)		$p = 0.25$
Flatten	9216		
Dense1	128	ReLU	
Dropout2	128		$p = 0.5$
Dense2	62	Softmax	

Table D.3: Options when generating the federated EMNIST-62 dataset with the LEAF repository

Flag	Option
-s	non-iid
–iu	1
–sf	“default”
-k	2
-t	sample
–tf	“default”
–smplseed	1234
–spltseed	1234

E | Extra Results

E.1 CIFAR-100

In figure E.1 we observe the recall and precision on the CIFAR-100 dataset across communication rounds. Generally these results mirror the previous results, which likely follows from that the CIFAR-100 dataset is well balanced in regards to labels. Do observe that before FedPA becomes unstable, precision shortly increases and recall decreases. This is noticeable in figure E.1d but also apparent in E.1c.

E.2 EMNIST-62

While the recall and precision for various algorithms on CIFAR-100 mirror the overall performance well, this is not the case for the EMNIST-62 results. Instead, in figures E.2a through E.2d, we notice that for EMNIST-62 the recall for FedPA appears to stay consistent while the precision drastically increases. In other words, the increase in accuracy that FedPA show on EMNIST-62 appears to be mainly due to an increase in precision.

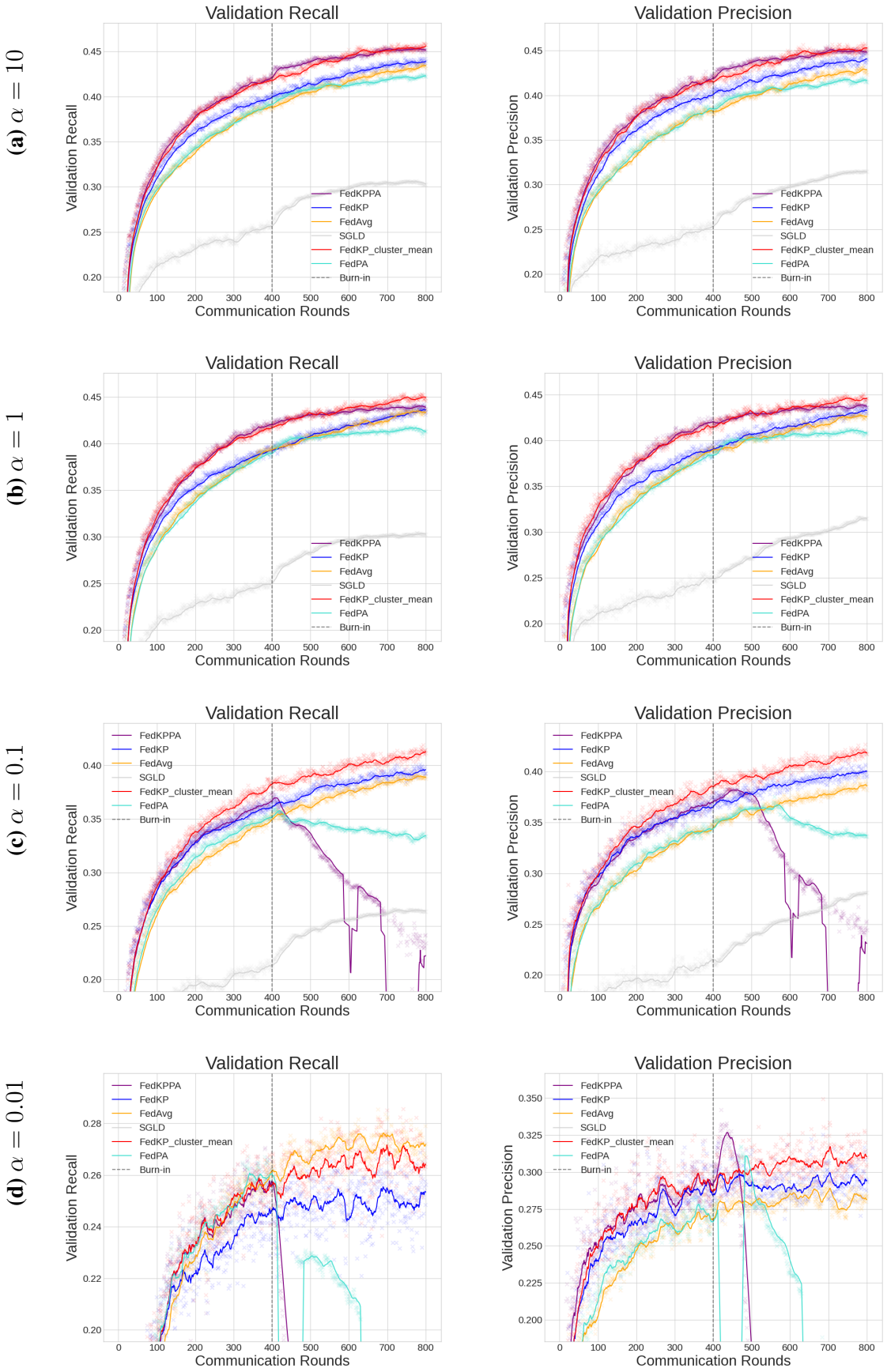


Figure E.1: Recall and precision of all algorithms on CIFAR-100 when varying α.

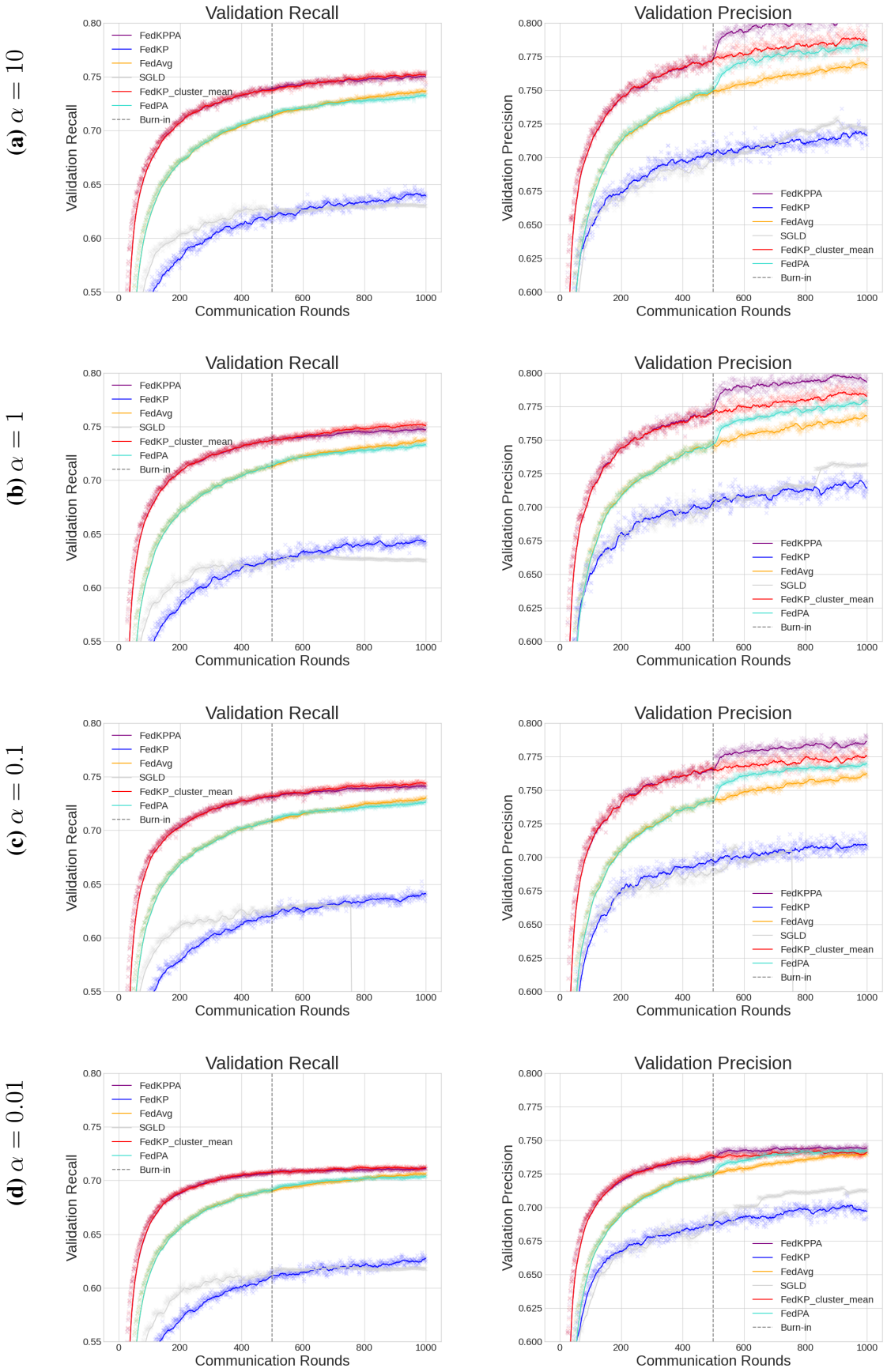


Figure E.2: Recall and precision of all algorithms on EMNIST-62 when varying α.