VU VRIJE UNIVERSITEIT AMSTERDAM

CWI

Master Thesis

The application and analysis of meta-learning in Federated Face Recognition

Author: Emiel Hess (2646644)

1st supervisor: Sandjai Bhulai
daily supervisor: Arwin Gansekoele
2nd reader: Mark Hoogendoorn

A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Business Analytics

October 14, 2023

The application and analysis of meta-learning in Federated Face Recognition

Emiel Hess

Vrije Universiteit Amsterdam

Faculteit of Science

Business Analytics

De Boelelaan 1081a

1081 HV Amsterdam

Host organisation:

Centrum Wiskunde & Informatica (CWI)

Stochastics Department

Science Park 123

1098 XG Amsterdam

Abstract

The increasing privacy concerns surrounding face image data demand new techniques that can guarantee user privacy. One such face recognition technique claiming to achieve more user privacy is Federated Face Recognition (FRR) a subfield of Federated Learning (FL). Federated face recognition is subject to data heterogeneity, because of the large number of classes that need to be handled, and to overcome this problem solutions are sought in the field of personalized FL.

This thesis introduces new data partitions based on the CelebA dataset that add new layers of heterogeneity to the data and implements Hessian-Free Model Agnostic Meta-Learning (HFMAML) in an FFR setting. This thesis shows that HFMAML scores higher on verification tests than current FFR models on three different CelebA data partitions, improving verification scores the most on the data partition with the most data heterogeneity.

In order to balance personalization and a good global model, an embedding regularization term is introduced for the loss function that can be combined with HFMAML and is shown to boost global model verification performance.

Lastly this thesis performs a fairness analysis, showing that HFMAML and its embedding regularization extension can increase fairness by reducing the standard deviation over the client evaluation scores.

Acknowledgements

This thesis is written in fulfillment of the master of Business Analytics program. I want to thank Sandjai for his help and support in the weekly meetings. I especially want to thank Arwin for his in-depth knowledge of the subject and his critical eye regarding my ideas and work.
This thesis is done as part of the fundamental research done at Centrum Wiskunde & Informatica (CWI). It is a national research institute founded in 1946 at Amsterdam Science Park. They do fundamental research and long-term innovation research in mathematics and computer science.
The research done in this thesis was commissioned by the stochastics department which has a branch doing research in AI and security.

Contents

List of Figures

List of Tables

1

Introduction

The recent decade has seen significant advances in the use of AI and machine learning. Because more and more data is being collected and stored, we can train larger, more complicated models. Acquiring data is often times very hard, especially when data is privacy sensitive like personal-, face image- or bank account data. Nevertheless, privacy-sensitive data can be very valuable when training machine learning models. A good example would be the training of deep learning models to detect brain tumors (38). The perfect solution would be for hospitals to collaborate by using MRI- and CT scans to train a global model. These MRI- and CT scans cannot be shared or collected on a single server since they belong to a user and privacy rules are in place to prevent this sensitive information from becoming public.

Federated Learning (FL) is a promising method that is used to develop algorithms to solve this problem. FL was first proposed by McMahan et al.(2016) (35). The basic idea of FL is that multiple clients can train on the same model without having to share their data. Instead of sending their data to a central server, each client receives model weights, which are updated using their own private data, sent back, and aggregated.

One of the types of data that is very privacy sensitive is face image data. There already exist many face recognition models(24) trained on datasets like LFW(19), MegaFace(39), MS-Celeb-1M(14), IJB-C(34) among others, but there are privacy issues and licensing issues surrounding these datasets(17). This makes it increasingly difficult to find face image data for training accurate models for face recognition, let alone training face recognition models for commercial use on commercial face image data. Assuming FL can achieve a sufficient amount of privacy it could easily be applied to scenarios where we have many different clients with large amounts of face images and use these images to train models that are able to distinguish faces without the need to share face images among clients. This new

sub-field of federated face recognition (FFR) has been introduced by Bai et al.(2021)(4) and Aggarwal et al.(2021)(2). Even though many techniques from FL can be applied to FFR, the fact that face recognition works with a very large amount of different identities (classes) makes it harder to overcome problems considering the aggregation of local models into a global model. On top of that, data heterogeneity is an issue in FFR. Making the assumption that all clients draw from the same distribution of face identities unrealistic.
Face image data suffers more from data heterogeneity than other classification tasks because of the large number of classes and small amounts of data per class. On top of that, making an assumption that each client has an approximately equal share of images is also rather unrealistic. Lastly, the data distribution over clients might even be more heterogeneous due to the fact that each client has different types of facial images, where for example one client has more images of old people and the other might have more images of people with a darker skin color.
Data heterogeneity in FL has been shown to influence performance negatively(61). There are a few papers that look at FFR that make different heterogeneity assumptions. Up to now, this has only been done with each client having different classes or only assigning one class per client(2)(4)(31). Aggarwal et al.(2021)(2) and Liu et al.(2022)(31) try to solve the heterogeneity problem in FFR using personalized FL (PFL)(61). This sub-field of FL consists of many algorithms that use an FL setup that is combined with personalization techniques like multi-task learning or meta-learning. These ideas can be used in FFR, where instead of creating a global model for all clients, each client can have its own local model that is better at identifying its own images in the set of all images.
Up to now the option of adding new clients after training has been explored by Jiang et al(2019)(20), but only briefly. It fits perfectly in a meta-learning paradigm since it is known to be good at quickly learning new tasks based on training old tasks(50). Finn et al.(2017)(12) introduced model agnostic-meta learning (MAML) which has been introduced into (personalized) FL by Fallah et al.(2020)(11) and Jiang et al.(2019)(20).
MAML or any other meta-learning algorithm has never been implemented in an FFR setting and has shown good results in FL settings. That is why this thesis will focus on combining meta-learning with FFR by introducing the Hessian-Free MAML (HFMAML) meta-learning algorithm in an FFR setting.
In order to measure performance this thesis uses the widely used verification evaluation technique, but also at the fairness of these verification scores per client, which is a measure of how fair all evaluation scores are distributed over all clients(28).
Lastly, since the paper by Liu et al.(2022)(31) provides the FedFR algorithm that is most

closely related to the setting of this thesis it will be used as a benchmark. The only big difference is the fact that FedFR uses a global dataset, which is not used in this thesis. Combining this all together has resulted in the following research question:

How can meta-learning be used to improve personalized performance and fairness on existing clients and newly added clients in federated face recognition?

The thesis made the following contributions to the field of FFR answering this question:

- Implementing the existing HFMAML meta-learning algorithm in an FFR setting.
- Introducing two new types of data partitions based on the CelebA dataset that can be used for training and evaluating FFR models.
- Showing that HFMAML achieves higher verification TAR@FAR scores than current FFR algorithms on data partitions with higher levels of data heterogeneity when no global dataset is available.
- Introducing embedding regularization that can be combined with HFMAML in order to control the amount of personalization, showing that this regularization can boost performance before tuning.
- Showing that keeping the last classification layer local does not negatively impact evaluation performance. Instead, it might even improve fairness.
- Showing that HFMAML decreases the standard deviation for the existing clients, suggesting that HFMAML can prevent clients from impacting the global model at the cost of other clients.

Before diving deeper into the methodology it is first necessary to cover some background knowledge about (personalized) federated learning, meta-learning, face recognition, and the FedFR and HFMAML algorithms. This is covered in Section 2.
Section 3 and Section 4 cover the methodology and experimental setup. Afterward, an overview of the results supporting the thesis contributions is given in Section 5. The thesis is finished with a conclusion in Section 6. The Appendix gives an overview of all the tables supporting claims made in the results.

2

Related Work

2.1 Federated learning

FL is a setting in which machine learning or deep learning models are trained by many clients and where the updated model weights are aggregated on a central server. The main purpose of FL is the decentralization of data to make sure it is kept private. FL was first introduced by McMahan et al.(2016)(35). They introduced a new algorithm called Federated Averaging (FedAvg). This algorithm prescribes how stochastic gradient descent can be performed on all client devices separately. The updated weights are collected and averaged on a central server. Algorithm 1 shows the FedAvg algorithm. This is used in FL as a general template for other aggregation algorithms.

Algorithm 1 shows that each client does batch stochastic gradient descent locally for a specified amount of epochs E. Afterwards all of these are collected on a central server where they are averaged. Here $\frac{n_k}{n}$ is the fraction of the total data from client k which means that the global model weights are a weighted average of the local weights.
FedAvg is the first algorithm used in FL and performs well on IID data. McMahan et al.(2016)(35) claims that it also performs well on non-IID data. However, other papers claim a decrease in performance due to non-IID data. After a brief overview of aggregation algorithms in FL, I will discuss non-IID performance issues and solutions in more detail.

2.1.1 Federated learning categories

This section gives a short overview of the different categories of FL. Yang et al.(2019)(59) gives a good explanation of the three categories of FL that are used in this field of research. The first one is horizontal Federated Learning (HFL), which is the most common category. In HFL all clients share the same features but have different samples of the data. This is

Algorithm 1 FedAVG algorithm, McMahan et al.(2016)(35)

Server
Initialize w_0
for t rounds **do**
  Sample subset of clients of size $m, 1 \leq m \leq K$ $\{S_1, ..., S_m\} \subset \{S_1, ..., S_K\}$
  Send w_t to all clients $S_k \in \{S_1, ..., S_m\}$
  for k clients $1 \leq k \leq m$ **do**
    $w_{t+1}^k = ClientUpdate(k, w_t)$
  end for
  $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for
ClientUpdate(k,w)
Split dataset into batches
for each epoch E **do**
  for each batch **do**
    $w = w - \alpha \nabla loss(w, b)$
  end for
end for

equal to the FedAvg algorithm by McMahan et al.(2016)(35) explained above. Each client has its own dataset with the same features, but they do not share any of the samples. In an ideal case, this data is IID, but usually this is not the case.

Vertical Federated Learning (VFL) is another category in FL, where instead of sharing the same features clients share the same samples, yet they do not share any of the features. Imagine different departments in a hospital that all have data on a client, but they are not allowed to exchange any of this. One department has brain-scan images, another one has blood testing results and another department tests the patient's urine. VFL is a process that aggregates this data in a privacy-preserving manner to train a model using local functions and SGD.

The last category is federated transfer learning. Here clients have a very small intersection in data when it comes to their sample space and feature space. This can be seen as a more extreme form of non-IID data. Transfer learning is used in a federated setting by first training a global model based on shared data between clients and afterwards these parameters are used as an initialization of local model training (23)(55)(7).

2.2 Non-IID data types

Before looking at non-IID data in an FL context it is good to explore different types of non-IID data that have been used in FL research.
Most papers that present non-IID robust aggregation methods only test their algorithm on a very limited amount of data partitioning strategies, whereas in reality a robust FL aggregation algorithm should be able to handle multiple types of non-IID data.
For example, McMahan et al.(2016)(35) test their FedAvg algorithm on 2-class partitions, where each client has parts of data from 2 different classes. Wang et al.(2020)(54) distribute the number of samples per class according to a Dirichlet distribution. This means both papers only look at a label-skewed non-IID case. For this reason Li et al.(2021)(25) created a benchmark to test different aggregation algorithms on different non-IID cases. They used six different image datasets and three tabular datasets and tested for the following non-IID cases based on the survey paper by Kairouz et al.(2019)(21). They give a good summary of different types of distributing samples over clients to create non-IID datasets. However, Zhu et al.(2021)(62) gives a more complete overview. The most interesting non-IID cases that are relevant to this thesis are given below.
In the non-IID case each client k has its own sample distribution $P_k(x, y)$, where x is the input data and y is the label.

2.2.1 Label distribution skew

Label distribution skew is the most commonly tested type of non-IID setting and has many applications. The idea is that $P_k(y)$ differs per client k, whereas $P_k(x|y)$ is the same for all clients.
Weather data for example is dependent on geographical location. Heavy snowfall will occur more often in Anchorage, Alaska than in Amsterdam. If you would try to predict snowfall and rainfall based on labeled cloud data you would find more data labeled as snowfall in Anchorage than in Amsterdam.
Label distribution skew is very easy to realize given a labeled dataset. The most common forms of label distribution skew are label size imbalances as used in McMahan et al.(2016)(35). They allocate a fixed amount of label classes to each client.
Another way of adding label distribution skew is by using a Dirichlet distribution, where each client is assigned a portion of the samples of a label class with probability $p_k \sim Dir(\beta)$.(1)(25)

2.2.2 Label preference skew

In label preference skew is the kind of skew that occurs in labeling tasks where the label assignment is open to interpretation. Take for example hotel review websites. Imagine you want to train an FL model that predicts user review scores of different hotels and each client trains the model on a dataset collected from a different website. Some website might give a certain hotel 4 stars, whereas some other website gives the hotel 4.5 stars. This is referred to as label preference skew.

2.2.3 Feature distribution skew

In feature distribution skew a distinction is made between different feature distributions $P_k(x)$ between clients, but $P_k(y|x)$ is similar for all clients. Animals classified as mammals differ in different geographical locations in the world. A dataset containing images of Australian animals labeled by species (mammals, reptiles, birds, etc.) is different from a dataset containing images of animals from Europe.
Li et al.(2021)(25) gives the most extensive explanation of feature distribution skew. They distinguish between three different types of feature distribution. The first one is noise-based feature imbalance. Here they add different amounts of Gaussian noise to each client dataset.
The second type of feature distribution skew they use is a synthetically created dataset. Here a feature imbalance is created by dividing a synthetic dataset based on synthetic features. Li et al.(2021)(27) created this type of feature distribution by dividing the coordinates of a cube into eight sections and giving each of four clients two of these sections.
The last type of feature distribution skew is based on the EMNIST dataset(9). This dataset contains different handwritings of characters and digits written by different people. Handwriting differs between people and when each client in an FL setting only has access to handwriting from an or a small sample of people a natural form of feature distribution skew arises.

2.2.4 Quantity skew

Quantity skew refers to different clients having different dataset sizes. Papers analyzing quantity skew (25)(43) in FL show that the FedAvg algorithm is quite robust against quantity skew in the data. What is important to note here is that the underlying distribution of data is the same for each client, so the data itself is IID distributed, but the dataset size is heterogeneous.

2.2.5 Temporal skew

Temporal skew can occur when data is temporal. A good example would be when multiple clients hold data from different time periods. For example, one client has a dataset with the local weather data in Amsterdam for the first 2 months of the year and another client has the weather data in Amsterdam for March and April.

2.3 Non-IID data issue in FL

Since FL works with many different clients who all have their personal data it is a very naive assumption to say that all data is IID. Therefore letting this assumption go would be a lot more realistic. McMahan et al.(2016)(35) already looked into non-IID data and found that the FedAvg algorithm was robust to non-IID datasets. They experimented with non-IID on the MNIST dataset, where they divided the labeled data into shards containing two classes and divided these shards over the clients. They also trained a language model on *The Complete Works of William Shakespeare*, where they divided the lines over clients by speaking role, which creates an unbalanced situation with clients with just a few lines and clients with a lot of lines. They also created an IID variant for comparison of this dataset with a more equal division of lines from the plays.
They compared performance by looking at the number of communication rounds it takes for each model to reach a certain accuracy. Although models trained on non-IID data take longer to reach this threshold they still reach it eventually given the right set of hyperparameters.
Where this paper is still quite optimistic about handling non-IID data with FedAvg, other papers (e.g. (62)(61)) explore the difficulties using FedAvg for training on non-IID data among clients both theoretically and empirically. Zhao et al.(2018)(61) show that highly skewed non-IID data reduces accuracy by up to 55%.

2.3.1 Weight divergence and client drift

To get a good sense of why the FedAvg algorithm has difficulties training on non-IID data we can compare the weights of FL in a non-IID setting with FL in a non-Federated setting with stochastic gradient descent. For this, we use the weight divergence, which Zhao et al.(2018)(61) define as:

$$weightdivergence = \|w^{fedAVG} - w^{SGD}\|/\|w^{SGD}\| \tag{2.1}$$

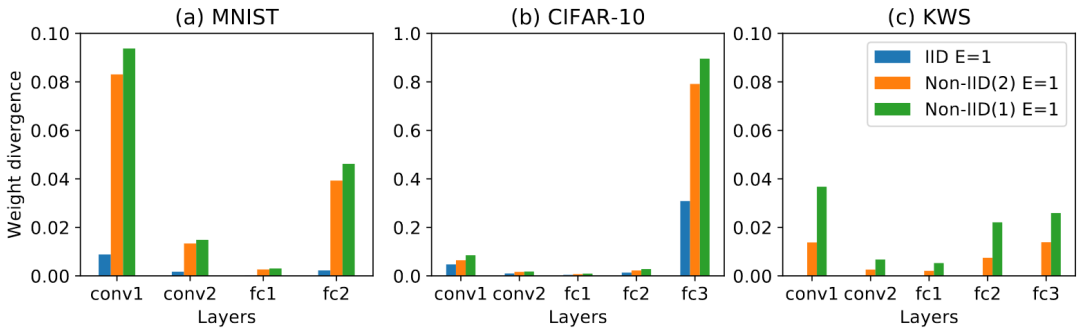

Figure 2.1: Weight divergence for different layers of the FL models on MNIST, CIFAR-10 and KWS data by Zhao et al(2018)(61)

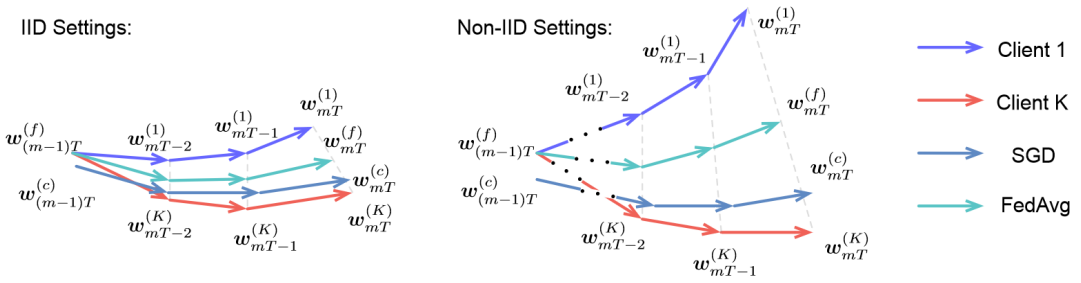

Figure 2.2: Weight divergence comparing IID setting with non-IID setting from Zhao et al.(2018)(61)

They compute the weight divergence for all layers of the neural network in their models and find a significantly higher divergence for the non-IID-trained models compared to the IID-trained models. This weight divergence difference between non-IID- and IID-trained models is clear in Figure 2.1. They have two different non-IID datasets. One with a 1-class partition, where each client only has data from one class, and a 2-class partition, where each client has data from 2 classes.

The weight divergence shows that when performing FedAvg on non-IID data the average distance between the weights of non-IID trained model and the benchmark non-FL model is larger than when performing FedAvg with IID data. This means that the non-IID trained model does not converge to the optimal weight values and thereby loses accuracy when testing the model. This phenomenon is also called client drift (25)(49). Zhao et al.(2018)(61) also gives a good illustration of weight divergence shown in Figure 2.2. Here we can see that weights diverge a lot more in the non-IID setting compared to the IID

setting and this divergence keeps accumulating per round.

2.3.2 FedAvg based solutions addressing drift

Li et al.(2021)(25) compare several different solutions trying to overcome this drift to make FL models perform better on non-IID data. The benchmark created by Li et al.(2021)(25) is a good starting point to see what is possible by using FL to train a global model on non-IID data. The paper compares $FedProx$(27), $FedNova$(54) and $SCAFFOLD$(22) on different types of non-IID data which are among the ones discussed in Section 2.2.
$FedProx$ algorithm adds a regularization term to the local loss function in the FedAvg algorithm to limit the distance between the global- and local model.
$FedNova$ looks at the aggregation function from FedAvg and assumes different amounts of local steps for different clients, due to computation power differences between them. Clients that perform more steps usually have greater updates, which has more impact on the average in the aggregation function. Normalizing for the number of steps taken per communication round removes the bias.
Lastly, $SCAFFOLD$ takes into account the variance between clients and performs variance reduction techniques. The updated direction of the local clients and the global model are estimated. These are then used to compute an estimate of the drift, which is used in the local objective function to correct local updates and avoid drift. Figure 2.3.2 shows a decision tree created by Li et al.(2021)(25). It shows which algorithm gives the best result for every non-IID category adopted in the benchmark they created. They did this by counting per non-IID category the number of times each algorithm achieves the highest testing accuracy. $SCAFFOLD$ performs very well on images with label distribution skew based on a Dirichlet distribution and feature distribution skew but performs very poorly on quantity skew. $FedProx$ performs well on quantity-skewed data and quantity-based label imbalance, where each client receives a fixed amount of label classes (either 1, 2, or 3 labels per client). It performs well on tabular datasets with a label distribution skew.

2.4 Personalized models

Tan et al.(2022)(49) gives two reasons for using personalized FL (PFL) models instead of regular FL models. The first one is a lack of convergence due to client drift as talked about in Section 2.3.1. The second reason is a lack of personalization from global models. In many situations, clients or groups of clients have personal preferences which deviate from the global average. In these cases, client drift should not be seen as a problem that needs

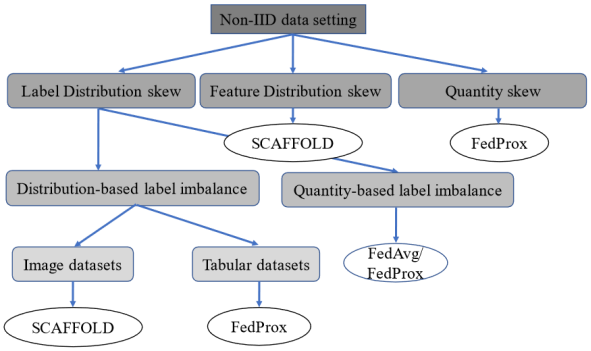

Figure 2.3: Best FedAvg-based algorithm for different non-IID settings from Li et al.(2021)(25)

to be controlled, but it can offer new opportunities in the field of FL.

PFL can be a solution for clients that lack sufficiently large datasets yet cannot share their data with other clients because of privacy reasons and/or personalization needs. Shared features can be exploited by collaboration between clients through FL. Many different methods of personalizing FL models have been studied.

In multi-task learning multiple tasks are solved simultaneously to exploit shared features between tasks(23). This form of learning is easily translated to the field of FL, where each client is seen as a separate task. Smith et. al.(2018)(46) introduced MOCHA.

Another technique is by using personalization layers. Each local client has a model containing multiple layers. A fraction of these layers is sent to the central server for aggregation. These are the base layers. The rest of the layers are never shared. These layers add personalization to the model. Arivazhagan et al.(2019)(3) introduced *FedPer*.

Hanzely et al.(2021)(16) proposed a mixture of local and global model training. Here they train multiple local models using local gradient descent with a penalty term for deviating too much from the global average model.

2.4.1 Meta-learning

Recently Meta-learning has introduced the idea of learning-to-learn to the field of FL. The idea is to expose the model to different data distributions (tasks) to make it better at learning new tasks very quickly. This is called few-shot learning. The model is trained to incorporate new classes with very few extra gradient descent steps and needing very few extra data points. The trained meta-learning model can be seen as an initialization from which new models can be learned. This is perfect for a PFL model, because each client

has its own data distribution which can be seen as different tasks(20). Training a global model on these different tasks results in a model that can learn new tasks very quickly.

$$\min_{w} \sum_{k} f_k(w - \alpha \nabla f_k(w)) \tag{2.2}$$

Two meta-learning approaches that are often used in a PFL setting are Model Agnostic Meta-Learning (MAML) by Finn et al.(2017)(12) and Reptile by Nichol et al.(2018)(40). MAML is model agnostic, which means that it applies to any model that uses stochastic gradient descent. This makes it easily transferable to a FL setting. The downside of MAML is that you need to calculate second derivatives for each gradient step, which makes it computationally demanding. That is why Finn et al.(2017)(12) also introduced First-order MAML (FOMAML), where the second-derivate terms are ignored. Reptile is very closely related to FOMAML. A MAML model is trained on a set of tasks in such a way that the parameters are very sensitive to small changes. This means only a few samples are necessary to personalize the model for a certain client. Equation 2.2 gives the MAML training objective.

Here f_i can be any loss function. This unfortunately means that to perform gradient descent on f_k you have to compute second-order derivatives. FOMAML and Reptile solve this problem by omitting these second derivatives. Jiang et al.(2019)(20) and Fallah et al.(2020)(11) used MAML to improve FL for personalized settings. Fallah et al.(2020)(11) created the $Per-FedAVG$ algorithm which is a combination of FedAvg and MAML. Jiang et al(2019)(20) shows the connection between the FedAvg algorithm and meta-learning. They claim that FedAvg already is a form of meta-learning. They see FL global model training as a form of meta-training and the subsequent personalization as the meta testing phase.

Based on both MAML and Meta-SGD (30) Chen et al.(2019)(6) proposes $FedMeta$, which is an algorithm that works with both MAML and Meta-SGD. Meta-SGD is an extension to MAML, where they do not only train the model weights but also a learning rate parameter α, one for each parameter. Overall, Meta-SGD seems to outperform MAML.

2.4.2 Multi-task learning

Multi-task learning tries to train models for multiple similar tasks simultaneously. What the relationship is between these tasks depends on the assumptions made. This is either known or unknown beforehand. This idea of learning multiple tasks simultaneously can be perfectly transferred to the field of FL. Similarly to meta-learning, the idea behind

multi-task learning is to treat each client as a task and train a different model for each client instead of a global model for all clients together.
Smith et al.(2018)(46) have introduced the algorithm MOCHA that implemented multi-task learning in a federated setting.

2.5 Federated facial recognition

Facial recognition is a field that has been extensively studied in the field of artificial intelligence, but it is still relatively new in the field of FL. Face images are very privacy sensitive and the way face recognition models are trained makes it very hard to ensure this privacy.

2.5.1 Face recognition

A lot of progress has been made in Face recognition. Li et al. (2020)(24) gives an overview of work already done in the field of facial recognition. Many algorithms achieve scores higher than 99% on the LWF public benchmark(19).
Important in face recognition is choosing an appropriate loss function. The paper by Shang et al.(2022)(45) studies different face recognition loss functions in a federated setting. Different from other classification tasks, face recognition has a very large amount of different classes for each individual. These classes are also called identities in face recognition
Different from closed-set classification tasks, face recognition is an open-set task(8). This means that face recognition models should be able to not only make a distinction between existing classes but also between unseen new classes. This is why face recognition models are usually tested on different data with different classes than it was trained on(8).
In order to work with an open-set problem an embedding is often used instead of the last classification layer to evaluate the model. This embedding is often the model output before it goes into the classification layer(8). The embedding is a unique vector for each input image that can be compared to embeddings of other images.
Schroff et al.(2015)(44) introduced $FaceNet$, which uses triplet loss. Here each face sample is compared to a positive sample from the same person and a negative sample, which represents a face from a different person to minimize the distance to the positive sample and maximize the distance to the negative sample. The problem with Triplet Loss and other pair-based loss functions is that it requires a lot of computational power to find good positive and negative samples.
Another type of loss function is the the classification-based loss function. The most widely

used loss function for classification in general is the softmax loss function, which is introduced to face recognition by Sun et al.(2014)(48) and later used for NormFace by Wang et al.(2017)(52).

2.5.2 Arcface Loss function

Since Arcface loss is the loss function that will be used in this thesis it is important to explain it a little further.

The softmax loss function is good at classifying based on closed-set classification tasks, but it is deemed not good enough for open-set face recognition tasks(8). Triplet loss performs better on open-set face recognition problems, but the combinatorial explosion problem makes it hard to use on datasets with many different identities.

Different loss functions have been introduced to avoid triplet loss and improve upon the softmax loss to make it better on open-set problems by increasing the margins between different identities. There is SphereFace(32), which introduces an angular margin, and CosFace(53) which uses a large margin cosine loss (LMCL) function. This loss function ensures that the margins between different classes are increased to improve the classification task of faces.

Arcface introduces an Additive angular margin to improve to improve the discriminative power of the model. It uses the Arc-cosine to compute distances between embeddings of images and the target embedding. An extra additive margin is added to improve the open-set classification task.

$$Loss_{Softmax} = -log\frac{e^{W_i x+b}}{e^{W_i x+b} + \sum_{j\neq i} e^{W_j x+b}} \tag{2.3}$$

$$Loss_{Arcface} = -log\frac{e^{scos(\theta_i+m)}}{e^{scos(\theta_i+m)} + \sum_{j\neq i}^{N} e^{scos(\theta_j+m)}} \tag{2.4}$$

Equation 2.3 gives the equation of a softmax loss function and Equation 2.4 gives the Arcface loss function. The structure of both functions is similar, but Arcface uses $scos(\theta_i + m)$ instead of $W_i x + b$ as logits, which are the weights W_i including the bias b. x is the feature from the input image after a few layers in the deep learning network. The logits are transformed in the following way:

$$W_i x + b = \|W_i\|\|x\|cos(\theta_i). \tag{2.5}$$

If $\|W_i\| = 1$ and $\|x\| = s$ by normalization we can transform the Softmax loss function into the Arcface loss function. If an extra margin m is added to the discriminative power

of the model the softmax loss from Equation 2.3 is transformed into the Arcface loss in Equation 2.4.

2.5.3 Federated face recognition

The field of Federated Face Recognition is still relatively new, but face image datasets have been used in federated learning before, like CelebA in the LEAF benchmark by Caldas et al.(2019)(5).

The explicit introduction of face recognition loss functions like $CosFace$ is done in two prominent papers in the field of federated face recognition by Aggarwal et al.(2021)(2), Niu et al.(2021)(41) and Liu et al.(2022)(31).

Aggarwal et al.(2021)(2) assumes that each client only has faces belonging to one class. They proposed FedFace which solves two problems related to facial recognition. The first problem is the fact that the classification matrix W_t contains sensitive information about each client, assuming that each client has images of only one user. The second problem is also related to each client having images of one user. Each client only has positive images and no negative images. Positive images are images from the same class and negative images are images from another class. In facial recognition classes and input images are embedded onto a high dimensional Euclidian space, whereby training you want to be as close as possible to the positive class and as far as possible from the negative incorrect classes. Without the images from negative classes it is not possible to train a model that tries to get as close as possible to the positive embedding by looking at the distances in the Euclidian space.

They solve this problem by pre-training the model on publicly available datasets. Afterward, to make sure that classes are separated in the embedding space without the need for a negative loss function they use a spread-out regularizer to aggregate the embeddings from all clients.

Liu et al.(2022)(31) does not assume one class per client and uses multi-task learning to train both a personalized and global face recognition model called FedFR. For personalization they used a technique called Decoupled Feature Customization.

What both FedFR and FedFace have in common is that they use pre-trained models that they subsequently try to improve with new client data.

Finally, Niu et al.(2021)(41) use gradient correction in their aggregation function to correctly aggregate the local class embeddings and create cross-client separability where all client class combined are correctly separated in the class space. They mention non-IID data but do not go into much depth. They separate the CASIO-WebFace(60)

A more recent work by Shang et al.(2022)(45) compares different loss functions for face recognition in FL. It also looks at the effect of clients having only small samples with very few classes. They find that the selection of the loss function depends on the data and has a great impact on the final model accuracy.

2.5.4 Classification in federated face recognition

In federated face recognition the number of different identities is large and one must carefully consider how to handle the classification layer and its weights because the total amount of identities is large and each client has a different set of identities.
Having a different classification layer for each client makes aggregation on the server harder, since separations between classes on the class embedding space are different for each client. Niu et al.(2021)(41) concatenates all local classification weights into a large weight vector: $W = [W_1, ..., W_C]$. To fix this class separation problem they apply a softmax regularizer and an extra step of stochastic gradient descent on the server. Unfortunately, this would require a globally accessible dataset on the server which may stop being realistic in the future.
Liu et al.(2022)(31) combines a local and global classification layer by sending each client a global classification layer with classes belonging to a global dataset. This is locally concatenated to a local classification layer belonging to the local data.

2.5.5 Face recognition datasets & privacy issues

There are many large prominent face recognition datasets like MegaFace(36), MS-Celeb-1M(14) and VGG face(42), but there exists some controversy around them.(37)(18). Nevertheless, these datasets are used in many papers related to face recognition.
Another widely used dataset in face recognition is Labeled Faces in the Wild (LFW) containing just 13,000 faces of 5749 people with 1680 people having two images or more. Since it is a relatively small dataset it is mostly used for evaluation. LFW was first proposed by Huang. et al.(2007)(19).
A dataset that has received less attention in the face recognition world is CelebFaces Attributes Dataset (CelebA)(33). The dataset was released in 2015 and consists of 202,599 images of 10,177 different identities. The dataset has 40 different attributes, which is its main selling point. An attribute is a visible feature on a person's face. This can be for example what color hair the person in the image has, whether the person wears glasses or has a beard, is either male or woman, or wears lipstick etcetera. The attributes are binary,

meaning that an image either does or does not have said attribute. Different images of the same identity can have different attributes. One person might have blond hair in one image, but gray hair in another.
CelebA has been part of the LEAF benchmark created by Caldas et al. (2019)(5). The LEAF benchmark is an FL benchmark that is created for creating and testing FL algorithms. They provide one client partition where each client has images belonging to one identity.

2.5.6 Federated Face recognition data partitions

Current literature on FFR gives three general different types of data partitions.
The first one is a random partition of data over the clients as used by Niu et al.(2021)(41). Since the partition is random this dataset is IID distributed and since we are only interested in heterogeneous data distributions this data partition is not used.
The second one is to assign one identity to each client. This way a scenario is replicated where each client can be seen as a person that has a set of images on its device. A paper that has used this data partition is by Caldas et al.(2019)(5). They created the LEAF benchmark, an FL benchmark that included the CelebA dataset, and assigned one identity to each client. Another paper that uses this partition in its experimental setup is by Aggarwal et al.(2021)(2). This type of data partitioning will not be used in this thesis, since it requires different techniques to deal with the small datasets per client and to solve the one-class-per-client problem.
The third type of partition is to assign a fixed equal set of identities to each client as is done by Liu et al.(2022)(31) and Shang et al.(2022)(45). This is the first type of data partitioning that will be used in this thesis. This gives some form of data heterogeneity since each client has a dataset sampled from a different distribution of identities. This is however still very limited.

2.6 FEDFR

In face recognition one can easily assume that before performing FL you can pre-train the model on publicly available face classification data. This approach is used in federated face recognition algorithms like FedFR(31). FedFR splits a total of 10,000 face identity classes into a pre-training dataset containing 6,000 classes and a federated training dataset, where 40 different clients each receive 100 identities. Since performance distinguishing only local

identities is not a very interesting or useful task it sends a subset of the global dataset to each client to help each client train a personalized model. Hard negative sampling is applied to select only the identities that are hardest to distinguish from the identities per client. Performance is measured in how well each personalized model can distinguish its own identities from other random identities. This in itself is a very interesting problem and Liu et al(2022)(31) show that their FedFR algorithm outperforms the regular FL algorithm in this task by using a form of multi-task learning. Each client has its own binary classification task that tries to classify its own clients. This is done next to performing cosface loss and contrastive regularization.
The biggest assumption that the paper makes is that there exists a global dataset. This is at first glance a very reasonable assumption, but in an increasingly privacy-aware world it is interesting to explore a situation where no such dataset exists. Many face recognition datasets are getting retracted narrowing the possibilities of large enough datasets that can be used for pretraining and supporting PFL (37)(18). It might be interesting to see how well FEDFR holds up when it does not have access to a global dataset.

2.6.1 FEDFR algorithm

The FEDFR algorithm consists of four different parts, and three of these parts are used in the loss function. Algorithm 2 shows the structure of the FEDFR algorithm

Hard Negative Sampling To avoid the model from overfitting on local data resulting in weight divergence Liu et al.(2020)(31) decided to sample a subset from a global dataset that that can be used for local training. This subset can be compiled randomly or you could simply send the entire global dataset to the client, but it is also possible to sample a subset of images that are the hardest to distinguish from the images on the client, making more effective use of the global dataset.
This hard negative sampling is done by computing an embedding vector of the all the global and local data using the local model without the Arcface head. Next, cosine similarity is computed between the local data and the global data. Then n images with the highest average similarity score with all the local data are chosen and used for local training in the FL process.

Contrastive regularization

$$Loss_{con} = -log\frac{exp(\frac{cossim(f,f_{glob})}{\tau})}{exp(\frac{cossim(f,f_{glob}}{\tau}) + exp(\frac{cossim(f,f_{prev})}{\tau})} \tag{2.6}$$

Algorithm 2 FEDFR algorithm

Server:
Initialize global dataset x_{glob}
Initialize global model $f_{glob}(x_{glob}, \theta)$
Train model $f_{glob}(x_{glob}, \theta)$
Initialize n clients
Send model $f_{glob}(x_{glob}, \theta)$ to all n clients
Send global dataset x_{glob} to all clients
Send local dataset x_i to each client $i \in n$
for r rounds **do**
    Train all clients and receive the model weights
    Perform FedAvg with weights received from clients
end for
Client training process:
Perform Hard negative sampling $x_{HNS} = HNS(x_{local}, x_{glob})$
for e epochs **do**
    Combine datasets $x = \{x_{HNS}, x_{local}$
    Compute feature embedding $x_{emb} = f_{emb}(x)$
    Compute $loss_{con} = f_{con}(x_{emb}, f_{glob}, f_{loc})$
    Compute $loss_{bce} = f_{bce}(x_{emb})$
    Compute $loss_{arc} = f_{arc}(x_{emb})$
    Perform SGD on $loss_{con} + loss_{bce} + loss_{arc}$
end for
Send model weights back to the server from f_{emb}, and the global weights from f_{arc}

Contrastive regularization is used to make sure that the local model does not deviate too much from the global model. This was introduced by Li et al.(2021)(26). Next to decreasing the distance between the global and local model Contrastive regularization also increases the distance between the previous and current model. The equation of contrastive regularization that needs to be minimized during the training process is defined in Equation 2.6. This function penalizes similarity between f and f_{prev} and rewards similarity between f and f_{glob}. Adding this to the loss function results in the wanted regularization for the federated model.

Decoupled feature customization The biggest addition that FEDFR adds to personalized learning is decoupled feature customization, which is inspired by the binary classification from Wen et al.(2022)(56). Here a transformation function is used to create a client-specific feature space that stimulates personalization. This is done using binary classification, where each identity on the local client refers to one class in the binary classification layer. For each image, it is determined if the image belongs to any of the local identities. This part of the model is different for each client and the weights of this part are not sent to the server. This means that the local- and global models are trained simultaneously, making tuning obsolete.

$$Loss_{bce} = \frac{\lambda}{s} log(1+exp(-s\cdot(g\cdot cos(\theta_i)-m)-b)+\frac{1-\lambda}{s}\sum_{j\neq i} log(1+exp(s\cdot(g\cdot cos(\theta_j)+m)+b)) \tag{2.7}$$

Equation 2.7 shows the loss function for the decoupled feature customization. θ_i is the cosine similarity between a trained weight vector for identity i and the embedding for input image with identity i. All other values are hyperparameters that need tuning. The minus in the first part of the equation shows that the larger the cosine similarity with the correct weight vector the smaller the loss and the smaller the cosine similarity with the incorrect weight vector the smaller the loss function. Computing the loss function for an image from the global training set means that the first part of the equation is 0 because the identity is not part of the local identities set.

Combining loss functions

$$Loss_{total} = \alpha_1 Loss_{arc} + \alpha_2 Loss_{con} + \alpha_3 Loss_{bce} \tag{2.8}$$

After a feature embedding has been computed from the backbone model it is sent to three different parts that compute three different loss functions, which are shown in Equation 2.8.

The first part is the Arcface loss using a classification layer with the local client identities and all identities from the global dataset. The second part is the contrastive regularization loss described in Section 2.6.1. The third part is the Binary classification loss shown in Section 2.6.1. The entire Algorithm is shown in Algorithm 2.

2.7 Fairness in FL

$$\min_{w} f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w) \tag{2.9}$$

Working with non-IID data distributions will probably result in an uneven influence over the global FL model. Li et al.(2020b)(28) have proposed q-fair Federated Learning (q-FFL), which encourages a more fair distribution of accuracy for all clients. They do this by adding a q-term to the FL optimization problem as shown in Equation 2.9.
The higher this q, the more emphasis is put on clients k with a higher loss. This means that clients with higher loss weigh heavier in the total aggregated weight. This means that to minimize this aggregated global weight more emphasis needs to be put on clients with higher local loss functions, which increases fairness.
Under the normal FL learning paradigm using FedAvg no guarantees can be given for local client model accuracy. One client might achieve higher accuracy because it has more data. Another reason is related to client drift. Clients that have weights that are very close to the global model weight, might achieve higher accuracy than clients with much more deviating weights. Li et al.(2020b)(28) and Li et al.(2021)(29) give a good definition of fairness:
Model m_1 is more fair than model w_2 if the test accuracy of model w_1 is more uniformly distributed over the clients than model w_2. This can be measured by taking the standard deviation over the test loss $F_k(w_1)$ and $F_k(w_2)$ and see which one is the lowest.
Li et al.(2021)(29) proposes *Ditto* a FL framework for PFL based on a multi-task learning approach. In the paper, the authors introduce *Ditto* a framework for PFL that outperforms other FL frameworks in fairness, robustness, and test accuracy. Fairness was measured by taking the standard deviation over the test accuracy, robustness was measured by test accuracy over different poisoning attacks.

2.7.1 q-FEDAVG algorithm

To solve Equation 2.9 Li et al.(2020b)(28) proposes the q-FEDAVG algorithm, which is an adaption of the original FEDAVG algorithm. This algorithm is described in Alogirhtm 3.

Algorithm 3 q-FEDAVG

Server:
Initialize global model f_{glob} with weights w_0
Initialize n clients
Initialize local models f_{loc}^n
for t rounds **do**
  Select k clients from all n clients
  Sent w_t to all k selected clients
  for k clients **do**
    Compute **Client** returning Δ_t^k and h_t^k
  end for
  Compute $w_{t+1} = w_t - \frac{\sum_k \Delta_t^k}{\sum_k h_t^k}$
end for
Client:
for E epochs **do**
  Perform stochastic gradient descent creating w_{t+1}
end for
$\Delta w_t = L(w_t - w_{t+1})$
$\Delta_t = f(w_t)^q \Delta w_t$
$h_t = q f(w_t)^{q-1} \|\Delta w_t\| + L f(w_t)^q$

This means that instead of sending back weights w_{t+1}^k for k clients we send back Δ_t^k and h_t^k, which is an adapted form of sending back gradients instead of weights. Δ_t^k are the client's gradients. These are subsequently weighted by their loss reflected in h_t^k. The higher the loss the more the client's gradient will influence the global weight. h_t^k is an upper bound to the Lipschitz constant which is used to dynamically tune the step size, which ensures we do not have to tune both q and the step size in the final gradient descent step on the server.
According to Li et al.(2020b)(28) q-FFL can decrease the variance in evaluation performance between clients by 45% on average.

2.8 Model Agnostic Meta-learning (MAML)

The model applied in this thesis is model agnostic meta-learning (MAML). This algorithm was first introduced by Finn et al.(2017)(12). Fallah et al.(2020b)(10) gives convergence guarantees of this approach and also gives a more in-depth explanation of the MAML algorithm and the different approaches of implementation.
The priority of the MAML algorithm is not to optimize the model, but to optimize the weights in such a way that they are easily adaptable in such a way that the model only needs very little unseen data to get accurate results. In other words, meta-learning and MAML specifically try to find a model that can easily be fine-tuned on new unseen tasks. Most machine learning algorithms try to find an approximation to an optimal solution by minimizing a loss function. In deep learning this is mostly done by updating weights w with stochastic gradient descent. In meta-learning one tries to find an initialization of w such that only a few steps of stochastic gradient descent are needed to find a good approximation of the optimal solution. This is done by training and testing a set of tasks $\{T_i\}_{i\in n}$ during training. Updated weights w are used for a second round of stochastic gradient descent to optimize these weights to find a solution in as few steps as possible.
The idea of training on a set $\{T_i\}_{i\in n}$ is very easily portable to a FL setting in multiple ways. The first way to implement meta-learning is shown by Fallah et al.(2020a)(11) and Jiang et al(2019)(20). Jiang et al.(2019) shows that the FedAvg algorithm can be interpreted as a meta-learning algorithm. Clients can be seen as different tasks, where an initial global model is trained on these clients. This initial model can subsequently be used to train personalized models for each client. This is similar to the fine-tuning step in meta-learning. To implement this in a face recognition setting and compare this to FedFR we could first pre-train this model similar to the FedFR model and even implement the

entire FedFR algorithm in the inner loop of the MAML algorithm.
In order to see the true power of meta-learning it could be interesting to see what happens when after training in a federated setting new clients are added with unseen data and perform few-shot learning on their face images.

2.8.1 MAML algorithm

The MAML algorithm as proposed by Finn et al.(2017)(12) is shown in Algorithm 4. This algorithm is implemented as closely as possible for experiments in this thesis.

Algorithm 4 MAML algorithm

Server
Initialize w_0
for t rounds **do**
  Sample sets n of images per identity $\tau_i \in p(\tau)$
  for all n tasks τ_i **do**
    Sample set k images D_i from τ_i
    $\tilde{w} = w - \alpha \nabla f(w, D)$
    Sample set l images D_i' from τ_i different from D_i
  end for
  $w = w - \beta \nabla_w \sum_i f(\tilde{w}, D_i')$
end for

$$\nabla_w f(w - \alpha \nabla_w f(w)) = \nabla_{\tilde{w}} f(\tilde{w})[I - \alpha \nabla_w^2 f(w)] \tag{2.10}$$

What happens in Algorithm 4 is the following: To promote the adaptability to new tasks property of meta-learning, training is done on small sets of tasks. In face recognition, each task can be seen as a different identity. Every iteration a small sample of faces is fed into the model after which stochastic gradient descent is performed. This first sample is considered the support data. Afterwards, new data, known as the query set is fed to the updated model. The loss in this step can be seen as how effective the model is in learning to recognize faces of the same identity after seeing just a small sample of them. Using stochastic gradient descent is done again in the so-called meta update w.r.t. the original weights w instead of new weights $\tilde{w}$. This means that a second derivative needs to be calculated. This is shown in more detail in Equation 2.10.

$$\nabla_w f(w - \alpha \nabla_w f(w)) = \nabla_{\tilde{w}} f(\tilde{w}) \tag{2.11}$$

where $\tilde{w} = w - \alpha\nabla_w f(w)$. Computing a second derivative can be computationally expensive. This is why Finn et al.(2017) (12) proposes a first-order approximation of the second derivative, which is almost as accurate. This updating rule is shown in Equation 2.11. Here $[I - \alpha\nabla_w^2 f(w)]$ is assumed to be 0.

2.8.2 Hessian-free MAML

Fallah et al.(2020a) (10) proposes a Hessian-free second derivative approximation instead of computing the second derivative. Equation 2.12 can be used to solve the second derivative problem. Fallah et al.(2020a) (10) claims that this can be computed in O(d) time.

$$\nabla_w^2 f(w)\nabla_{\tilde{w}} f(\tilde{w}) = \left[\frac{\nabla_w f(w + \delta\nabla_{\tilde{w}} f(\tilde{w})) - \nabla_w f(w - \delta\nabla_{\tilde{w}} f(\tilde{w}))}{2\delta}\right] \tag{2.12}$$

$$\nabla_w f(w - \alpha\nabla_w f(w)) = \nabla_{\tilde{w}} f(\tilde{w})[I - \alpha\nabla_w^2 f(w)] = \nabla_{\tilde{w}} f(\tilde{w}) - \alpha\nabla_w^2 f(w)\nabla_{\tilde{w}} f(\tilde{w}) \tag{2.13}$$

Here Equation 2.12 can be substituted in Equation 2.13 to create a Hessian-free approximation, which can be used in the MAML algorithm.

2.8.3 Federated MAML algorithm

To get better results with fine-tuning FL and meta-learning the MAML algorithm can be implemented more explicitly into the federated setting. The approach in this thesis is mostly adopted from Fallah et al.(2020a)(11) and Fallah et al.(2020b)(10)

Algorithm 5 Federated MAML algorithm

Initialize w_0 on the server
for t rounds **do**
  Sample n clients $k, 1 \leq k \leq n$ with dataset $\tau_k \in p(\tau)$
  for all n clients τ_k **do**
    Sample set of images D_k from τ_k
    $\tilde{w_{t+1}}^k = w_t - \alpha\nabla f(w_t^k, D_k)$
    Sample sets of images D_k' and D_k'' from τ_k different from D_k
    $w_{t+1}^k = w^k - \beta\nabla_{\tilde{w_t^k}} f(\tilde{w_t^k}, D_k')[I - \alpha\nabla^2_{w_t^k} f(w_t^k, D_k'')]$
  end for
  $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for

$$\nabla_w^2 f(w, D_k'')\nabla_{\tilde{w}} f(\tilde{w}, D_k') = \left[\frac{\nabla_w f(w + \delta\nabla_{\tilde{w}} f(\tilde{w}, D_k'), D_k'') - \nabla_w f(w - \delta\nabla_{\tilde{w}} f(\tilde{w}, D_k'), D_k'')}{2\delta}\right] \tag{2.14}$$

As is shown in Algorithm 5, the federated variant of MAML is very similar to the normal MAML algorithm. Instead of tasks, you sample from different clients and instead of doing the second gradient descent step with the sum of all meta-losses per task, the step is performed on each client with their own loss function separately. Afterward, these meta-updated weights are sent back to the server and federated averaging is performed.
In order to avoid computing the second gradient $\nabla_{\tilde{w_t^k}} f(\tilde{w_t^k}, D_k')[I - \alpha\nabla_{w_t^k}^2 f(w_t^k, D_k'')]$ is computed as shown in Equations 2.12 and Equation 2.13. This approximation requires a third round of data sampling D_k''. This third dataset is necessary to do the second-order approximation. Otherwise, the same dataset is used twice. Adapting Equation 2.12 to show what dataset is used where results in Equation 2.14.

3

Methodology

To solve the question of how meta-learning can be used to improve personalized performance and fairness on existing clients and newly added clients in federated face recognition the following problems need to be tackled:

- Data partitions with differing degrees of data heterogeneity are needed to evaluate how the different algorithms perform.

- A meta-learning algorithm implemented in an FFR scenario. Meta-learning has been applied in FL, but not in FFR. Implementing a meta-learning algorithm in FFR will come with implementation problems that need to be overcome.

- A definition of fairness. To properly evaluate the fairness a definition is needed. Fairness can be defined in multiple ways(28). For example, it might be unfair to value each client equally in the determination of fairness. Some clients might require a model that has to be more accurate in its prediction than other clients. In this case, a weighted fairness might be more appropriate. This is why it is important to give good definitions before starting the experiment.

- Algorithms that can help increase fairness. When a model implemented in an FFR setting scores very low on fairness it is convenient to have algorithm extensions that are specifically created to help increase the fairness of this model.

After these problems are solved an experiment can be set up to test different models on different clients containing separate train- and test datasets.

3.1 Fairness

Definition 1 *Model A is considered to have higher fairness than Model B iff* $Var(X_a) < Var(X_b)$*, where* X_m *is defined as the evaluation score of model* m*.*

The same definition of fairness of Li et al.(2020b)(28) is used in this thesis too. This definition is described in Definition 1. Fairness looks at the impact a model has on the variance in client evaluation scores. The lower the variance in evaluation scores for clients, the more fair this model is deemed to be. One can use both the variance and the standard deviation.

3.2 Data partitions

In this thesis three different types of data partitions have been created to evaluate the models. FFR data partitions currently used in literature are explained in Section 2.5.6. This thesis assumes that there can be multiple identities per client and is interested in heterogeneous data partitions. This means that there is only one data partition left that has already been explored in current literature. This will be called the equal class partition in this thesis. To dive deeper into researching the effect of data heterogeneity on different algorithms this thesis introduces two more data partitions that have not been used in this context before.

3.2.1 Lognormal class partition

To extend this data partition this paper introduces a class-based partition where the amount of classes assigned to each client is based on a lognormal distribution. The idea for using a lognormal distribution comes from the paper by Niu et al.(2021)(41), but they do not partition based on classes, but on the data, creating a quantity skewness in the data. This is also very similar to label-skew described in Section 2.2.1, but previous works have used a Dirichlet distribution(1)(25). This means that each client gets a portion of the samples belonging to a class. This works in settings with few classes, like MNIST where the model has to distinguish handwritten digits. This means that there are ten classes. Assigning fractions of samples of each class to clients in different proportions creates data heterogeneity. For example, based on the Dirichlet distribution one client gets assigned 5 images of digit 0, 100 images of digit 1, 2000 images of digit 2, 5 images of digit 3, and no images of the other digits. It is shown that this is a very hard partition to train on(1)(25),

but this does not translate well to face recognition data.
Face recognition datasets have large amounts of classes with very small sample sizes. Assigning data randomly will create almost the same effect as partitioning based on a Dirichlet distribution since clients will not get data from all identities anyway. It therefore makes more sense to assign entire classes (identities) to clients instead of fractions of samples of classes.

$$C\frac{S_i}{\sum_i S_i}, S_i \sim lognormal(\mu, \sigma), 1 \leq i \leq n \tag{3.1}$$

To use the lognormal distribution to create a partitioning a random number is generated based on a lognormal distribution for each client. To give each client a fraction of the identities based on this random number each number is divided by the sum of all random numbers. This creates a value between 0 and 1 for each client. This value is multiplied by the total amount of identities to create the number of identities this client will have and rounded such that the sum of all these values equals the total amount of identities. Lastly, each client is randomly assigned identities based on the random value. This process is shown in Equation 3.1. Here, C is the total amount of identities.

3.2.2 Attribute-based partition

The last newly added data partition type is an attribute-based partition. The images in the CelebA dataset have not only been labeled based on identity but on more features like whether the person has a beard or not or if the person is wearing make-up. These image features are called attributes and these attributes can be used for classification or for improving face recognition based on dependence of attributes(15). Using the attributes for creating a new data partition to test heterogeneity in FFR is as far as I have seen not been done before. Different attributes can be combined to create more specific partitions. For example, one client has a dataset containing only images of men with beards and glasses. This data partition is new, but based on the idea of the FEMNIST dataset that is included in the LEAF benchmark. Each client gets assigned handwritten letters based on the writer resulting in each client having data in different handwriting. This is a form of feature-based skewness.
Further explanation of these data partitions is provided in Section 4.2.

3.3 Algorithms

The meta-learning algorithm used in this thesis is HFMAML. The reason is twofold. First of all, MAML is model agnostic, meaning that it is compatible with any model that uses gradient descent(12). Second, HFMAML uses an approximation of the Hessian matrix instead of computing it, making it more time-efficient compared to MAML. MAML (and HFMAML) has been shown to work well with classification models(20)(11)(12), but implementation in Face Recognition is more complicated since you have to assume knowing the total amount of identities of all client combined beforehand. This also includes This thesis proposes two ways of dealing with this problem.

3.3.1 Classification layer

The classification layer of the model refers to the part of the model after the backbone. This part of the model can be a classification loss function, like Softmax, CosFace, or ArcFace.
Whether you would use a Softmax or an Arcface loss function in the classification layer, if all weights are sent to the server for aggregation the total amount of identities need to be known before training. This includes clients that are only involved in the testing phase if you want to tune for these clients.
As discussed in Section 2.5.4 the amount of different identities is large in FFR and a decision must be made whether you would only consider the identities per client or all identities of all clients combined when creating the classification layer of the model. Section 2.5.4 shows two different solutions by Niu et al.(2021)(41) and Liu et al.(2022)(31), but since both solutions require the existence of a global dataset these cannot be used in this thesis. Instead, two different options are proposed. The first one is to use a global classification layer that is similar for each client. The second one would be to use local classification layers for each client, which are not sent back to the server for aggregation.

Global classification Using a global classification layer means that each client has the same classification model part and thus the same classification weights. This means that weights can be sent back and forth from the server to the clients without any issue.
Even though this is protocol-wise the easiest solution there are two problems. The first one each related to privacy. Using a global classification layer means that each client will

receive all classification weights of all clients increasing privacy-related risks. Even though this thesis does not go in-depth on these privacy-related risks of FFR it is still important to keep in mind that weights can be inverted to recover images of other clients(13).
The second problem occurs during training when most classes in the classification layer remain unused. A solution to this is to set the logits of the classes that are not on the client to $-\infty$. This requires keeping track of which logits belong to which identity on which client. For example, if you have a 100 classes in total and you have 2 clients the most obvious thing to do is relabeling the classes of client 1 as classes 1 up and until 50 and relabel classes of client 2 as 51 up and until 100.

$$\lim_{x \to -\infty} \frac{e^y}{e^x + e^y} = \frac{e^y}{e^y + 0} = 1 \tag{3.2}$$

$$\lim_{x \to -\infty} \frac{e^x}{e^x + e^y} = \frac{0}{e^y + 0} = 0 \tag{3.3}$$

Equation 3.2 and Equation 3.3 show what happens when you set a logit in the softmax function to $-\infty$. Setting a logit for an image to $-\infty$ means that the probability of this image belonging to that identity is zero. This restricts the options available to each client making training local models faster.

Local classification layer Using local classification layers per client means initializing a local classification part for each client separately and keeping the weights local while sending the weights of the backbone model to the server for aggregation.
Each training round the aggregated backbone model weights are sent to each client and combined with the local classification weights for training.

3.3.2 Embedding regularization

$$Loss = f(x) + C(1 - cos(f_{emb}^{global}(x), f_{emb}^{local}(x))) \tag{3.4}$$

The idea of adding an extra term to the local loss function in FL is not new. Li et al.(2020a)(27) introduced FedProx, where they added a regularization term $\frac{\mu}{2}\|w - w_t\|$, where w are the global weights and w_t the local weights. Adding this to the loss function results in the local model being punished for diverging from the global model. Liu et al.(2022)(31) also did some form of regularization by adding contrastive regularization as explained in Section 2.6.1. They do not use all the weights in their regularization term, but only the embedding layer. Combining the simplicity of FedProx and the idea of using cosine similarity and the embedding vector of FedFR results in Equation 3.4. Here the

embedding vectors from the local and global models are used for each input for computing the cosine similarity and adding this to the loss function multiplied by a predetermined constant. The loss function is shown in Equation 3.4. Important to note that $f(x)$ is the criterion loss value of the model and $f_{emb}(x)$ is the embedding vector.
The reason for using embedding regularization in this thesis is two-fold:

- Keeping local models closer to the global model might avoid overfitting on local client models with smaller datasets. If a local model is overfitting its weights might diverge more from the global averaged weights. This might improve evaluation performance for smaller clients using the global model since the optimum found for the client is closer to the global optimum. This might subsequently decrease the variance between client evaluation scores, resulting in more fairness.

- Since the local model is closer to the global model there is less room for personalization. The hyperparameter C can be tuned to balance personalization and a good global model to its preference.

3.4 Combined algorithm

Algorithm 3.4 shows all implementation details that are added to the HFMAML model as described Section 3.3.1 and Section 3.3.2. Parts marked dark blue are algorithm additions for the local classification layer implementation, parts marked green refer to using a global classification layer, the part marked red is Embedding Regularization and the parts with the cyan color are general algorithm additions to make HFMAML work in an FFR setting.

Algorithm 6 Federated HFMAML local regularized

Server
Initialize w_0^{server} on the server
if Local classification layer **then**
  Initialize a_0^k as the classification weights on each $client_k$ $1 \leq k \leq K$
end if
if Global classification layer **then**
  Initialize a_0^{server} as the classification weights on the server
end if
for t rounds **do**
  for $client_k$ $1 \leq k \leq K$ **do**
    Sample set D_k from $client_k$ training dataset
    $w_t^k = w_t^{server}$
    if Global classification layer **then**
      $a_t^k = a_t^{server}$
      Determine indexes of logits that are cast to $-\infty$
    end if
    Define $F(w, a, D) = f(w, a) + C(1 - cos(f_{emb}^{global}(w, D), f_{emb}^{local}(w, D)))$
    $(\tilde{w}_{t+1}^k, \tilde{a}_{t+1}^k) = (w_t^k, a_t^k) - \alpha \nabla F(w_t^k, a_t^k, D_k)$
    Sample set of images D_k' and D_k'' from $client_k$ training dataset different from D_k
    $(w_{t+1}^k, a_{t+1}^k) = (w_t^k, a_t^k) - \beta \nabla_{\tilde{w}_t^k, \tilde{a}_t^k} f(\tilde{w}_t^k, \tilde{a}_t^k, D_k')[I - \alpha \nabla^2_{w_t^k, a_t^k} f(w_t^k, a_t^k, D_k'')]$
    Send w_{t+1}^k back to the server
    if Global classification layer **then**
      Send a_{t+1}^k back to the server
    end if
  end for
  $w_{t+1} = \sum_{k=1}^K \frac{n_k}{\sum_{i=1}^K n_i} w_{t+1}^k$
  if Global classification layer **then**
    $a_{t+1} = \sum_{k=1}^K \frac{n_k}{\sum_{i=1}^K n_i} a_{t+1}^k$
  end if
end for

4

Experimental Setup

4.1 Experimental Assumptions and limitations

Before diving deeper into the experimental setup it is necessary to go over some assumptions and limitations in this experimental setup.

- There is no global dataset available on the server. Liu et al.(2022)(31) assume that there is a global dataset available that can be accessed by all clients in their proposed FedFR algorithm. This means no optimal scenario for the FedFR algorithm. The reason FedFR is still used as a benchmark in this thesis is that it is still the only personalized FL algorithm in FFR.

- There is no pretraining before the start of the FL training process. Liu et al.(2022)(31) uses the global dataset to perform pretraining. The same argumentation applies as described in the previous point.

- Multiple classes are allowed per client. Aggarwal et al.(2021)(2) and Caldas et al.(2019)(5) make the assumption for their FedFace algorithm and LEAF benchmark respectively that there can only be images belonging to one identity per client. In other words, only one class per client. In this setup, one class per client is allowed, but the total amount of classes will be bigger than the total amount of clients resulting in at least one client having more than one class.

- With a global classification layer the amount of identities of all clients combined must be known beforehand, which includes clients that are not involved in the training process. This means that in this situation the problem is reduced to a closed-set classification problem since all classes are included in the classification layer

- With a local classification layer the total amount of identities does not need to be known beforehand. This makes it possible to add new clients after the training process without specifying, making it an open-set problem.

- Most FFR experiments use multiple datasets for training and evaluating their models(31)(2)(41). This thesis only uses part of the CelebA dataset for both training and testing. The reason is that other datasets do not split their images based on attributes and are mainly used for global model evaluation. This thesis does not aim at achieving state-of-the-art evaluation performance. It rather focuses on the relative results between different models.

4.2 Data partitioning

This section gives an overview of how the three different data partitions are created. The dataset that has been used to create these partitions is the CelebA dataset.
In total there are 20 clients. 15 of these clients are used for training and 5 are not involved in the training process and are only used for evaluation together with the first 15 clients. This idea of adding clients after training is introduced by Jiang et al.(2019)(20). A total of 1,500 classes were used for the experiment. For an equal class partition this means that each client will have 75 classes. For the lognormal class partition this means that each client has 75 classes on average. The data for each client is divided into a train, validation, and test set, where for each class 70% of data goes to training, 10% is used as validation dataset and 20% is used as test set. The reason that only a fraction of the dataset is that it seemed unnecessary to include more data. The goal of this thesis is not to achieve high verification scores, but to compare different algorithms in different scenarios.
Finally it is important to note that to make results reproducible a random seed is used.

4.2.1 Equal class partition

In the equal class partition the dataset of each client consists of images belonging to 75 randomly assigned classes, such that no two clients share the same classes.
Since each class has a different sample size it means that the total amount of images per client can differ.

4.2.2 Lognormal class partition

The lognormal partition is explained in Section 3.2.1. For the lognormal distribution, the following weights are used: $\mu = 3$ and $\sigma = 3$. Random numbers are generated using the same seed for all experiments and all runs and the assignment of identities to each client is fixed beforehand as well. This is done because the other two partitions have fixed partitions too for each run.
The amount of test data and training data per client is shown in Table 4.3.

4.2.3 Attribute-based partition

The CelebA dataset contains a file that keeps track of 40 binary attributes per image. These attributes range from hair color to whether or not the person is wearing glasses. Since some attributes are more common than others different combinations of attributes are made to create subsets of data that can be assigned to the clients. Table 4.1 shows how the attributes are combined for each client. For example, client 5 has images with both male and female images. It is the only client with people with no glasses. They wear hats, they can be both old and young and the client has images of people with all types of hair color. In short, the focus of client 5 is on people with hats. People with hats are only located on client 5, since there are very few images of people with hats.
Client 12 only contains images of males that are not wearing either glasses or a hat. They are old people with no blond hair, meaning they can have any hair color except blond. The people in the images have no goatees, which means that all men with goatees are filtered out. Lastly, they all have bushy eyebrows.
When an attribute is not mentioned it means that the client is agnostic towards this attribute; it has both people with or without this attribute in its dataset.
Of course an identity may have images with different attribute distributions.
As can be seen in Table 4.1 dataset sizes are kept as close as possible to reduce the effect of quantity skew as much as possible.

4.3 Haar Cascades

Even though the images from the CelebA dataset are focused on the faces themselves they still contain a lot of background. Therefore, since this thesis only focuses on face recognition and not the detection part some cropping of the images is necessary in the preprocessing

Table 4.1: Different combinations of attributes that are assigned to each client

Cl.	Gender	Glasses	Hat	Old/Yng	Hair	Attr 6	Attr 7	Attr 8	Size
1	Male	Yes	No	Young	All Types				1066
2	Male	Yes	No	Old	Grey				1007
3	Male	Yes	No	Old	All Types	Chubby			1001
4	Female	Yes	No	Both	All Types				925
5	Both	No	Yes	Both	All Types				1010
6	Female	No	No	Both	Blond	Oval Face	Rosy Cheeks		1109
7	Male	No	No	Both	All Types	Goatee	Not bald	Smiling	1044
8	Male	No	No	Both	All Types	No Goatee	Bald		1019
9	Male	No	No	Both	Gray	No Goatee			1040
10	Male	No	No	Old	All Types	No Goatee	Bushy Eyeb.		1062
11	Male	No	No	Old	No black	No Goatee	Bushy Eyeb.		1007
12	Male	No	No	Old	No blond	No Goatee	Bushy Eyeb.		1062
13	Male	No	No	Young	Brown	No Goatee	Bushy Eyeb.		1077
14	Female	No	No	Old	All Types	Oval Face	Lipstick	No R. Chks	1076
15	Female	No	No	Young	All Types	No Oval Face	Lipstick	Rosy Chks	1248
16	Male	No	No	Both	All Types	No Bushy Eyeb.	Mustache	No Bushy Eyeb.	1077
17	Male	No	No	Young	All Types	Beard	No Mustache	Bushy Eyeb.	1147
18	Male	No	No	Young	All Types	Beard	No Mustache	No Bushy Eyeb.	1108
19	Male	No	No	Young	Black	Bags u. Eyes			1039
20	Female	No	No	Old	All Types	Bags u. Eyes			1029

phase. Here a technique called Haar Cascade by P.Viola and M.Jones (2001) (51) is used. This technique is very fast; even fast enough for real-time detection and also very accurate. Haar Cascade works with so-called Haar features. These features are used to find features in the data. These Haar features traverse the entire image, searching for their particular shapes.

The biggest downside when using Haar Cascades is the large fraction of false negatives. From a total of 202,599 images from the CelebA dataset, Haar Cascade managed to detect 180,977 faces, which resulted in a successful detection in 89% of the images. A false negative of 11% is not ideal, but 180,977 cropped images are still large enough to work with. Figure 4.1 and Figure 4.2 show an example of an image pre- and post-cropping. Figure 4.3 shows an example of an image where Haar Cascade did not manage to detect a face. This has probably to do with the fact that we only see the side of the face and not the entire face.

Figure 4.1: uncropped version of a CelebA image

Figure 4.2: the same image cropped

Figure 4.3: An example of an image where Haar Cascade did not detect a face

Table 4.2: Train and test sizes for each client for the lognormal client partition

client	train data	test data	client	train data	test data
1	134	32	11	264	65
2	1488	376	12	389	99
3	748	184	13	2203	554
4	95	23	14	222	54
5	259	66	15	287	69
6	2585	650	16	276	71
7	27	7	17	4657	1163
8	346	86	18	4307	1076
9	1955	495	19	1326	334
10	220	58	20	773	196

4.4 Scenarios

For this experiment 23 different types of model scenarios are trained, based on the FL algorithm and FL using either HFMAML or FedFR.
The different types of experiments that are being run are shown in Table 4.4. FL refers to Federated Learning using only FedAvg.
Generally, there are three types of algorithms being tested: Federated Learning described in Section 2.1, federated HFMAML described in Section 2.8 and FedFR from Section 2.6. Four different types of algorithm extensions are introduced in Section 3.3.1 (local or global classification layer), Section 3.3.2 (embedding regularization), and Section 2.7.1 (q-FedAvg by Li et al.(2020b)(28)).
All algorithm extensions are combined with all algorithms on all data partitions except for the following exceptions:
Embedding regularization and q-FedAvg are not tested on this partition, because of the lower amount of data heterogeneity compared to the other two partitions resulting in more initial fairness. This makes these two algorithms less interesting to test on the equal class partition.
There is only one FedFR scenario per dataset partition. FedFR is combined with a local classification layer, because after removing the global dataset identities from the classification layer there is only a local classification part left and this is not sent to the server in the paper by Liu et al.(2022)(31). Furthermore, embedding regularization is not used in combination with FedFR, since FedFR already uses a different regularization term in the loss function. Lastly, FedFR is not combined with q-FedAvg, since q-FedAvg is used as a benchmark for embedding regularization, and since embedding regularization is not combined with FedFR it made no sense to combine q-FedAvg with FedFR.

4.5 Model architecture and implementation

The model architecture used to train both the HFMAML and FL model is shown in Table 4.5. The Arcface loss function is used as described in Section 2.5.2 with $s = 8$ and $m = 0.5$ that have been chosen after some tuning. Important to note is that Table 4.5 portrays the global classification layer scenario with 1500 identities. For the local classification layer, the output of the Arcface part depends on the number of local identities. The linear layer between the ResNet and Arcface is used to cast the output vector of ResNet of size 1000 to the size required for the embedding vector. In this thesis, the embedding

Table 4.3: The different scenarios that will be tested in this thesis. If local is not marked it means that the classification layer is global

Algorithm	Data partition	Local	Regularization	q-FedAvg
FL	Equal			
FL	Equal	x		
HFMAML	Equal			
HFMAML	Equal			
FedFR	Equal	x		
FL	Lognorm			
FL	Lognorm	x		
FL	Lognorm		x	
FL	Lognorm			x
HFMAML	Lognorm			
HFMAML	Lognorm	x		
HFMAML	Lognorm		x	
HFMAML	Lognorm			x
FedFR	Lognorm	x		
FL	Attribute			
FL	Attribute	x		
FL	Attribute		x	
FL	Attribute			x
HFMAML	Attribute			
HFMAML	Attribute	x		
HFMAML	Attribute		x	
HFMAML	Attribute			x
FedFR	Attribute	x		

Table 4.4: The model architecture

Layer type	In	Out
ResNet 18 (pre-trained)	3x128x128	1000
Linear	1000	512
Arcface (s=8 m=0.5)	512	1500 (total amount of identities)

Table 4.5: The chosen hyperparameters

Hyperparameters SGD optimizer		Hyperparameters q-FedAvg	
Learning rate	0.01	q	0.001
Momentum	0.9	L	100
Hyperparameters HFMAML		Hyperparameters FedFR	
α	0.01	α_1	1
β	0.1	α_2	5
δ	0.001	α_3	10

vector will have a length of 512. This embedding vector is used for computing the cosine similarity. This cosine similarity is used for model evaluation and used for embedding regularization as described in Section 3.3.2.

For doing both training and tuning a I use a stochastic gradient descent optimizer with learning rate 0.01 and momentum 0.9. Table 4.5 shows the hyperparameters used for the HFMAML algorithm. These parameters correspond to parameters in Equation 2.12 and Algorithm 5.

Important to note on these hyperparameters is that some tuning has been done, yet not very extensively since the importance is not to achieve state-of-the-art performance, but to compare performance between algorithms.

4.6 FedFR setup

$$Loss_{total} = \alpha_1 Loss_{arc} + \alpha_2 Loss_{con} + \alpha_3 Loss_{BCE} \tag{4.1}$$

The FedFR model that is being implemented is trying to follow the original model as described in the paper by Liu et al.(2022)(31) and Section 2.6.1. The same hyperparameters have been used for combining the different loss functions as shown in Equation 4.1. Here we have the Arcface loss, the contrastive regularization loss, and the binary cross-entropy loss. Table 4.5 shows these hyperparameters.

In order to conform to the assumptions as described in Section 4.1 FedFR does not have

access to a global dataset or pretraining as in the original paper. This means that hard negative sampling and a global class embedding containing all of these global classes are not implemented in this version of FedFR.
Since the original paper does not perform tuning it will not be used in evaluation in this thesis as well. Tuning results are shown, but will usually be around the same evaluation score.

4.7 HFMAML setup

The HFMAML algorithm as described in Section 3.4 is implemented as closely as possible to its form in Algorithm 3.4. Because the basic structure is taken from the personalized algorithm as described by Fallah et al.(2020)(11). For the regularization embedding term in the loss function, the value $C = 10$ is used in Equation 3.4.

4.8 q-FedAvg algorithm

The q-FedAvg algorithm is implemented as closely as possible to the algorithm described by Li et al.(2020b)(28). The hyperparameters used are shown in Table 4.5.

4.9 Evaluation metrics

For evaluation face recognition models there are two options: identification and verification(57). Verification is the process of comparing two images and determining whether these two images are the same person or not. Identification is the task of correctly matching an image to the correct class in a gallery of different classes. Both evaluation metrics are in many face recognition benchmarks like the IARPA Janus Benchmark(34)(57) and the LFW dataset(19). These face matchings and comparisons are done using the embedding vector. These embedding vectors are usually the outputs of the backbone model before they go into the classification layers. Embeddings are compared using cosine similarity.
Whether verification, identification, or both are used depends on the benchmark dataset used. LFW provides tools for doing verification and not identification and uses accuracy to score evaluation performance(19). The IARPA Janus benchmark on the other hand provides tools for doing both verification and identification. They use the TAR@FAR scoring system to evaluate model performance (57).
Current FFR research uses these public benchmarks to evaluate their model performance (2)(31)(45)(41). The problem is that this can only be used to evaluate the global model

because these benchmarks do not include FL client splits. Since this thesis focuses on performance per client these benchmarks are not relevant to this thesis. Instead, a new evaluation metric needs to be set up that can evaluate performance per client. Liu et al.(2022)(31) came across the same problem and their client verification protocol is used in this thesis on the CelebA dataset. For each client data is split in a train and test set as described in Section 4.2 such that each identity is in both the test set and the train set. After training is done random pairs are sampled from the test data in the following way: half of these pairs have the same identity and the other half have different identities. When two identities are different at least one of these identities must belong to the client that is being evaluated. When the images have similar identities they are both from the client that is being evaluated. Of all these pairs the cosine similarity is computed.
After all cosine similarity scores are computed the scores from the dissimilar images are ordered and the cosine similarity score from the 90th percentile is taken as a threshold. After establishing this threshold the fraction of similarity scores above this threshold from the similar images is computed. This fraction is known as the true acceptance rate at a fixed false acceptance rate (TAR@FAR). With the 90th percentile threshold, we talk of a TAR@FAR 0.1, which is the fraction of true positives with a 10 percent false positive rate.

4.10 Training & Evaluation

Training is done for 30 rounds where local training is performed once on each client for 50 epochs. One epoch does not equal an entire iteration over the client training dataset, but it can mean one of two things. For the FedAvg and FedFR algorithm it means 50 batches of size 64. The HFMAML algorithm however uses three sampling rounds as shown in Section 3.4 and Section 2.8. This means that in this case each epoch the algorithm samples three separate batches of size 64. Both FL and HFMAML have been subject to tuning of the amount of epochs used. 50 epochs seemed best in both cases.
The same training process of 30 rounds is done 10 times for each scenario to make performance estimates as accurate as possible. After training is done, performance is evaluated using verification testing as described in Section **??**. The evaluation procedure is described in Algorithm 4.10. This is done for all 20 clients. Each scenario has been subject to 10 training runs with different seeds.
During evaluation each training run is tested 5 times resulting in 50 TAR@FAR 0.1 evaluations using the global model and 50 TAR@FAR 0.1 evaluations using a global model

tuned on local client data. These two models will be called the global model and the tuned model from now on.
In total there are 100 TAR@FAR 0.1 scores computed for the first 15 clients and 100 scores for the last 5 clients per scenario.
In scenarios with a local classification layer the local classification weights are combined with the global backbone weights to tune the model. This is shown in Algorithm 4.10.
A more detailed description of the TAR@FAR 0.1 verification implementation is shown in Algorithm 4.10.
After evaluation is done TAR@FAR 0.1 scores are averaged per client for each scenario. Averaging over all average client TAR@FAR 0.1 scores results in an average score per scenario.
The standard deviation over all client scores is computed per run and subsequently averaged to find an average standard deviation per scenario. This standard deviation is used to measure fairness as defined in Section 3.1. To test for significance Student's t-test(47) and the Wilcoxon test(58) are used. These two tests require a sample of results. There are 50 scores computed per scenario. If we average over all clients 50 times per scenario a sample of 50 is created which can be used to perform these two tests.

Performing this test multiple times creates the opportunity to test whether tuning the model results in significant improvement and for a significant difference between TAR@FAR 0.1 results between different models. The tests that will be performed are the Student's t-test and the Wilcoxon test.

Algorithm 7 TAR@FAR test

Initialize test dataset D_{client}
Initialize dataset $D_{global} = \{D_1, ..., D_{20}\}$
Initialize model $f_{emb}(w_{client})$
for 1000 rounds **do**
  Sample $d_c \in D_{client}$
  Sample $e \in D^{class}_{client} \subseteq D_{client}, e \neq d_c$. D^{c}_{client} is a subset containing only images of the same class c as image d_c
  Compute cosine similarity $c^{true}_{client} = cos(f_{emb}(d_{class}, e)$
end for
$c^{true} = \{c^{true}_1, ..., c^{true}_{1000}\}$
for 1000 rounds **do**
  Sample $d_c \in D_{client}$
  Sample $e \in D_{client}$ for local testing or $e \in D_{global}$ for global testing, again $e \neq d_c$
  Compute cosine similarity $c^{false}_{client} = cos(f_{emb}(d_{class}, e)$
end for
$c^{false} = \{c^{false}_1, ..., c^{false}_{1000}\}$
Sort c^{false} from lowest to highest and take the 10th percentile highest cosine similarity, threshold $t_{0.1}$.
$c^{true}_{0.1} = \{c | c \in c^{true} \wedge c > t_{0.1}\}$
Return $\frac{c^{true}_{0.1}}{c^{true}}$

Algorithm 8 Evaluation process

Initialize model weights w^{i}_{FedAvg} for $model_i$ $1 \leq i \leq 10$
for each w^{k}_{FedAvg} $1 \leq k \leq 10$ **do**
  for each $path$ $1 \leq path \leq 5$ **do**
    Compute TAR@FAR 0.1 for w^{k}_{FedAvg}
    Tune w^{k}_{FedAvg} using 10 batches of size 64 of the training set resulting in w^{k}_{Tuned}
    Compute TAR@FAR 0.1 for w^{k}_{Tuned}
  end for
end for

Algorithm 9 Evaluation process

Initialize w^k_{FedAvg} for $1 \leq k \leq 10$
Initialize local Arcface weights a^k_{FedAvg} for $1 \leq k \leq 10$
Initialize $f(w^k_{FedAvg}, a^k_{FedAvg})$
for each w^k_{FedAvg} $1 \leq k \leq 10$ **do**
    for each $path$ $1 \leq path \leq 5$ **do**
        Compute TAR@FAR 0.1 for $f(w^k_{FedAvg}, a^k_{FedAvg})$
        Tune $f(w^k_{FedAvg}, a^k_{FedAvg})$ using 10 batches of size 64 of the training set resulting in tuned model weights w^k_{Tuned} and a^k_{Tuned}
        Compute TAR@FAR 0.1 for $f(w^k_{Tuned}, a^k_{Tuned})$
    end for
end for

5

Results

This section will give an overview of the results in support of the claims made in this thesis. This section shows plots in support of these claims. Tables with the exact numbers are given in Appendix A. The statistical tests for comparing TAR@FAR scores between scenarios is given in Appendix B. The statistical tests that compare standard deviations is given in Appendix C.

5.1 Convergence analysis

Figure 5.1 shows the average validation score per round for the first 15 clients. The blue line follows the score per client on the global model and the yellow line is the score on the tuned model. Figure 5.2 shows the same validation score per round for the last 5 clients that have not been involved in the training process but are still validated each round. Both figures show converged validation scores.
The difference is that the first 15 clients converge more smoothly than the last 5 clients. This might be a result of a sample size of 15 compared to 5 and the fact that the last 5 clients have not been used in the training process making the validation performance less predictable.
These two figures are representative of the training curves of all other scenarios. All scenarios have been checked to see whether the validation score is converged before round 30 to make the comparison between scenarios as fair as possible.

5.2 Comparing performance between FL and HFMAML

One thing that becomes immediately clear when looking at Figure 5.3, Figure 5.5 and Figure 5.7 is that HFMAML outperforms FL and FedFR models consistently within all

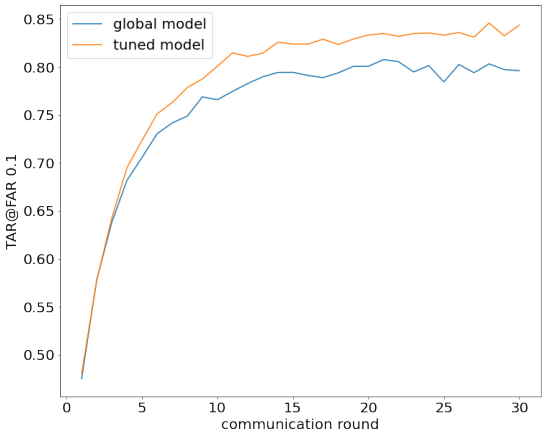

Figure 5.1: FL with equal class partition. TAR@FAR 0.1 results after each communication round with the clients 1-15; before and after tuning with 5 batches.

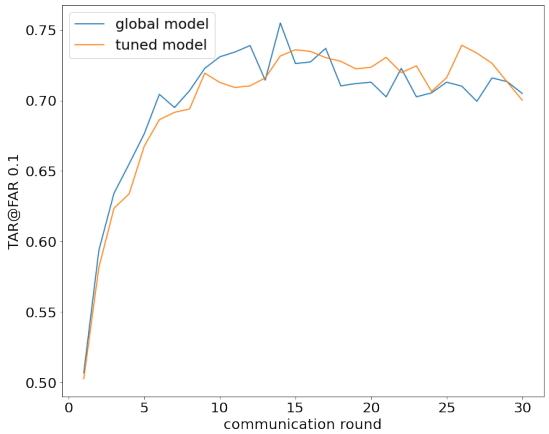

Figure 5.2: FL model with equal class partition. TAR@FAR 0.1 results after each communication round with the clients 16-20; before and after tuning with 5 batches.

scenarios; the global classification, local classification, q-FedAvg or regularization models. These differences are in almost all cases significant. This section focuses most on the difference in performance between FL and HFMAML because even though FedFR does perform comparable to FL and HFMAML before tuning it is outperformed by both FL and HFMAML models on its TAR@FAR 0.1 score.

Figure 5.4, Figure 5.6 and Figure 5.8 show that the effect of tuning is comparable for FL and HFMAML scenarios. Interestingly, the effect of tuning is equal for the first 15 and last 5 clients on the equal class partition as shown in Figure 5.4. For the lognormal class partition shown in Figure 5.6 tuning sees the highest increase on the first 15 clients and for the attribute-based partition the highest increase due to tuning is on the last 5 clients as shown in Figure 5.8. This can be explained by the structure of these partitions. Table 4.3 shows that the last 5 clients have relatively large datasets compared to the first 15 clients. This happened by accident, but it still affects the tuning performance of the last 5 clients. Large datasets that have not been trained on require more iterations to get the same TAR@FAR 0.1 score as the first 15 clients.

The bigger TAR@FAR 0.1 score increase on the last 5 clients for the attribute-based partition is probably related to the fact that there is more data heterogeneity between clients. The models have been trained on data heterogeneous clients or tasks making especially the HFMAML model better at learning new tasks quickly.

The margin by which HFMAML outperforms FL scenarios is shown in Figure 5.9. It

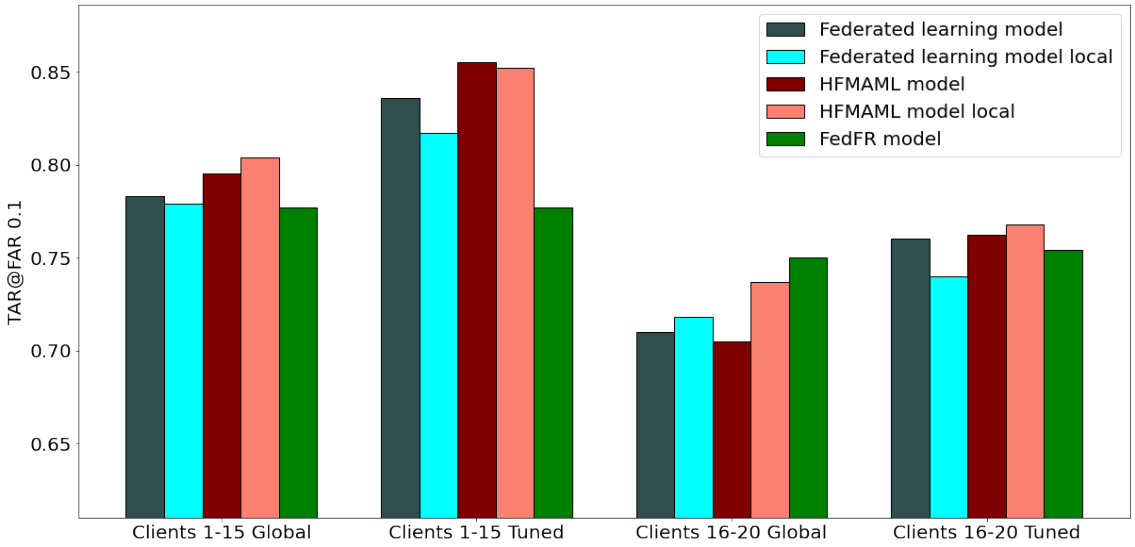

Figure 5.3: Results for the equal class partition dataset.

shows that the largest difference in performance in terms of TAR@FAR 0.1 is on the attribute-based dataset. This shows that HFMAML performs best in situations with feature-skewness in the data. This fits the idea of what the strong points of meta-learning and MAML are: quickly adapting to new tasks. The attribute-based dataset has the clearest form of task separation between clients because it adds an extra layer of data heterogeneity on top of the separation in classes.

5.3 Comparison of local- and global classification

Looking at Figure 5.3, Figure 5.5 and Figure 5.7 we see that the TAR@FAR 0.1 scores of local- and global classification scenario's are close. Using a local classification layer achieves higher TAR@FAR 0.1 scores before tuning and the global classification layer models perform better after tuning. This is most clearly visible in Figure 5.7 for both the FL and HFMAML scenarios

The most likely cause is that tuning a local classification layer model requires the combination of a global backbone and a local classification layer. These two model parts probably need a few extra epochs to synchronize.

Overall, this closeness in the TAR@FAR 0.1 score means that classification weights can be kept local and new clients can be added after model initialization without a large evaluation performance drop.

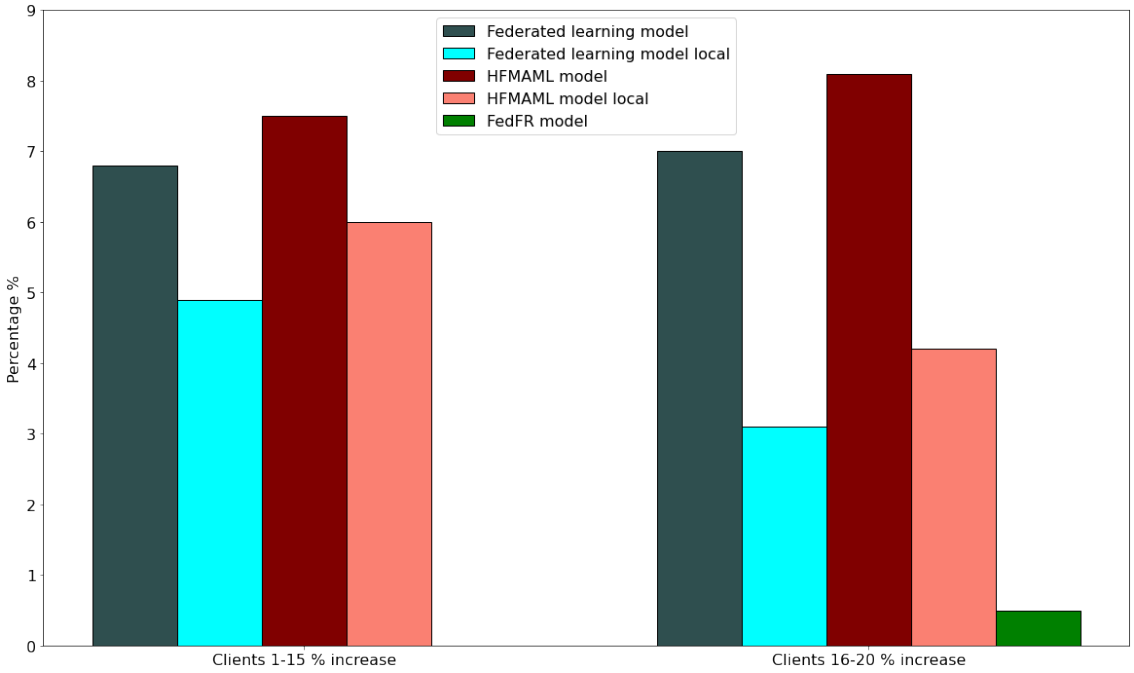

Figure 5.4: Percentage increase in TAR@FAR 0.1 score for the equal class partition dataset due to tuning.

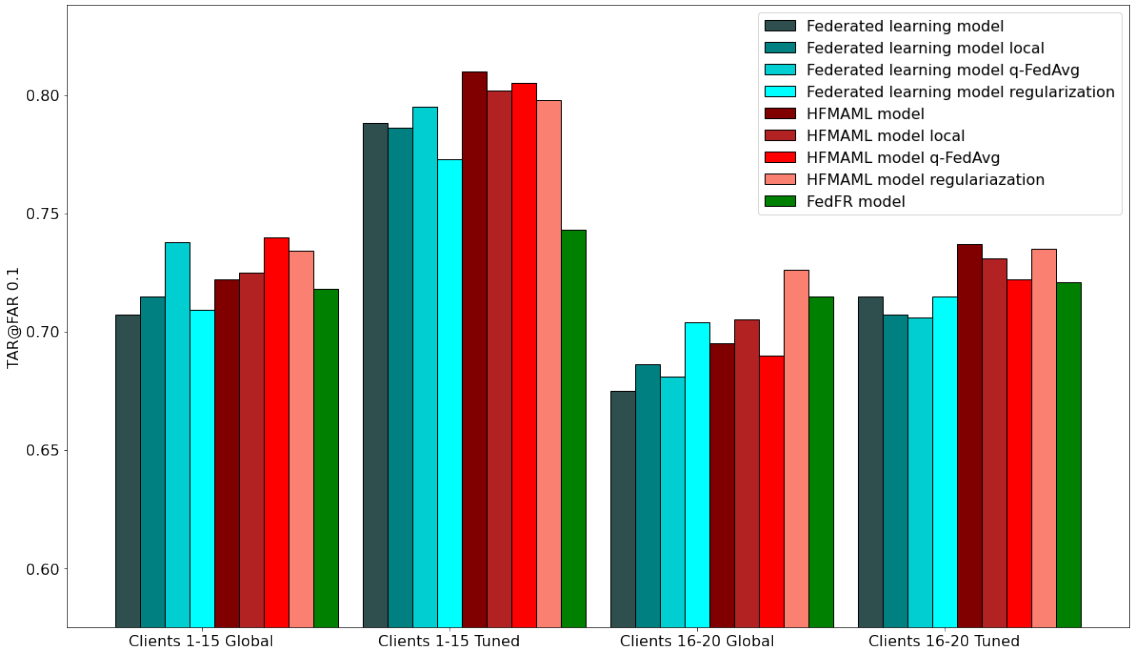

Figure 5.5: Results for the lognormal class partition dataset.

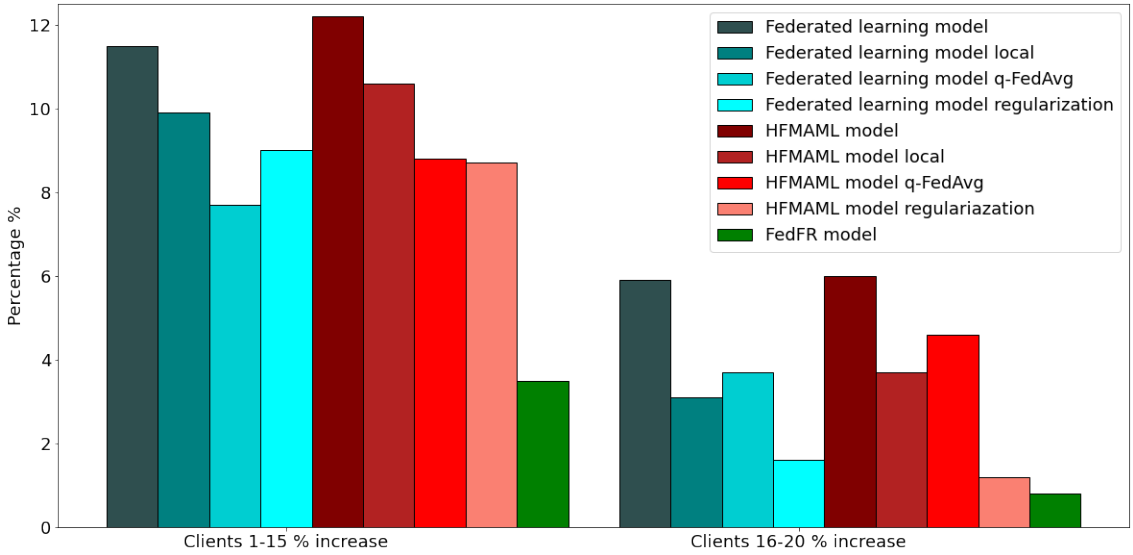

Figure 5.6: Percentage increase in TAR@FAR 0.1 score for the lognormal class partition due to tuning

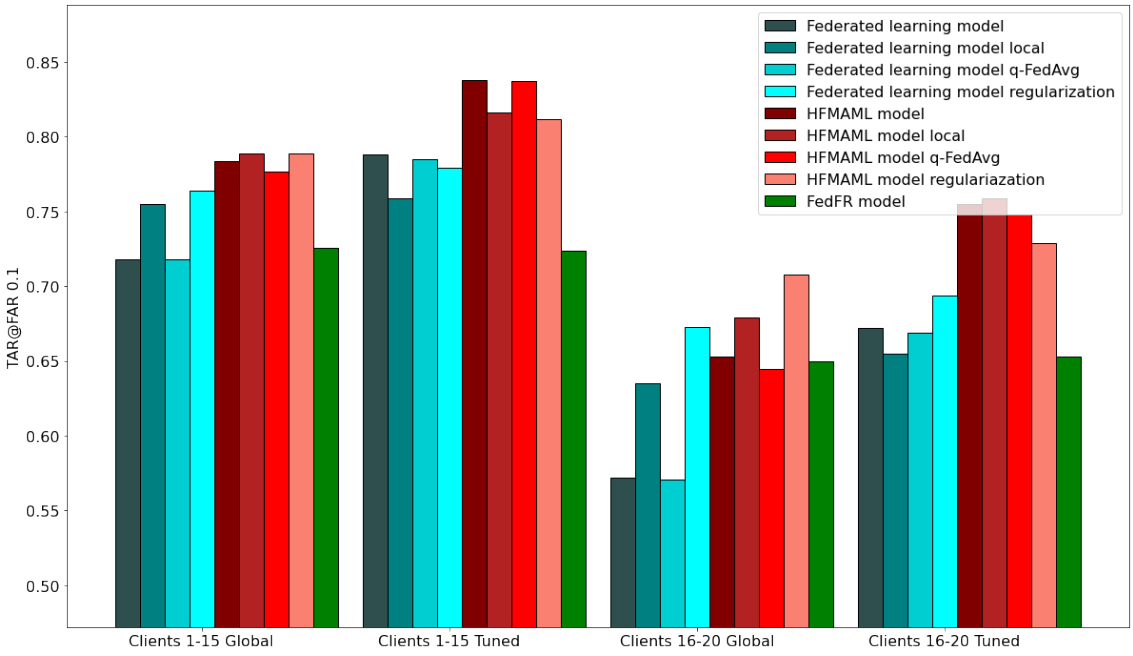

Figure 5.7: Results for the attributes-based partition dataset.

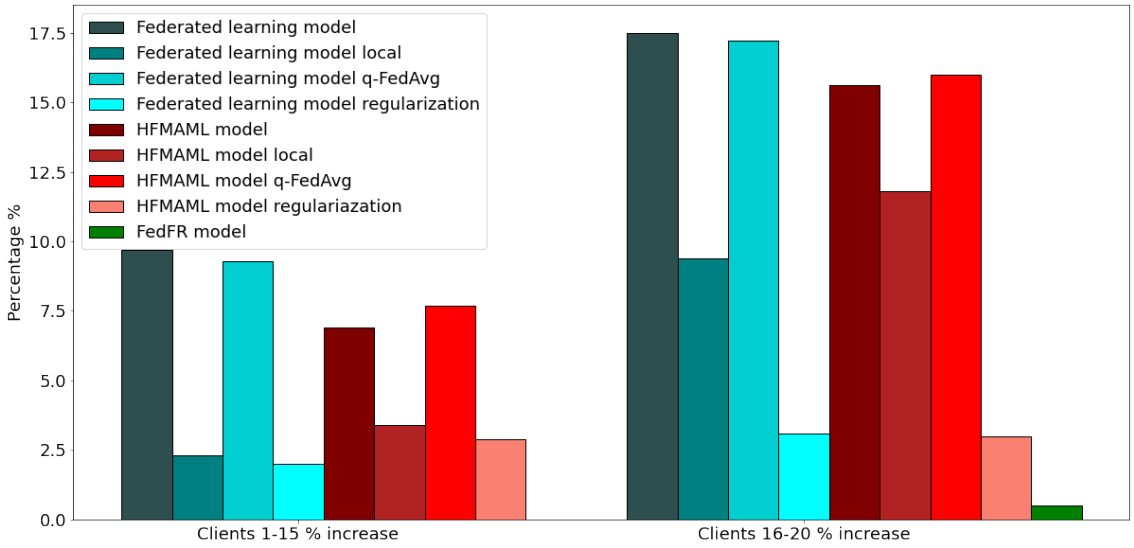

Figure 5.8: Percentage increase in score for the attribute-based partition due to tuning

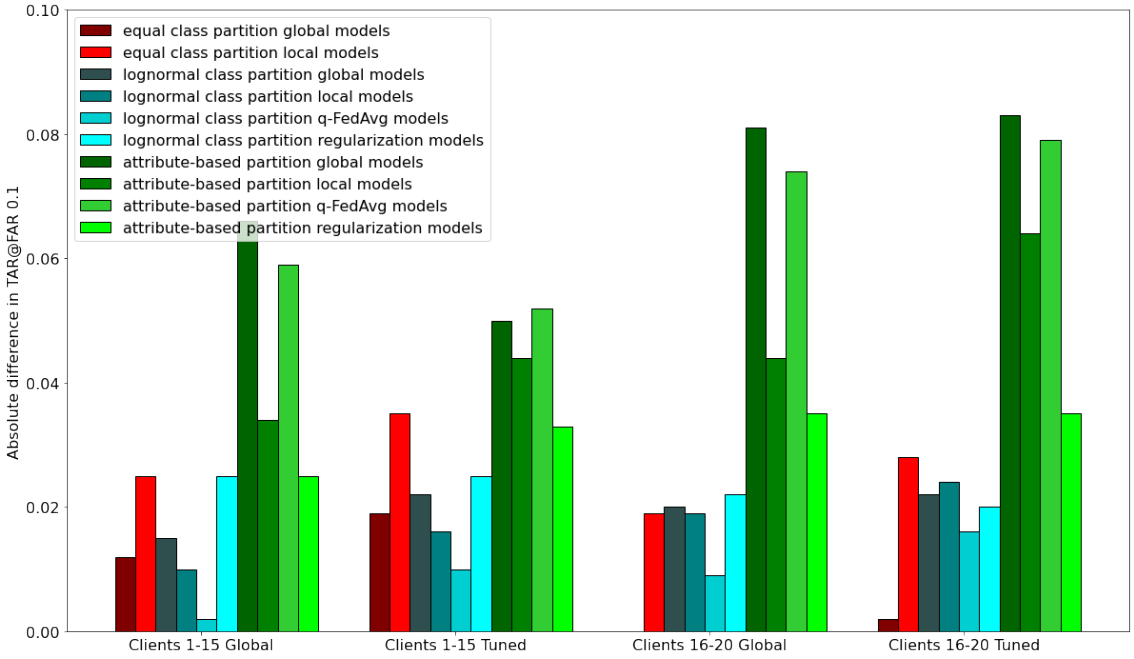

Figure 5.9: The absolute difference in TAR@FAR 0.1 score between FL scenario's and HFMAML scenario's on different data partitions.

5.4 Embedding regularization analysis

Figure 5.5 and Figure 5.7 show embedding regularization models outperforming the other models for both HFMAML and FL on the global model evaluation. This shows that embedding regularization results in the local client models staying close to the global model, resulting in a higher TAR@FAR 0.1 score before tuning compared to the non-regularization models. This higher evaluation performance using embedding regularization is most pronounced in the untrained clients. Figure 5.5 even shows the untrained clients achieving a TAR@FAR 0.1 almost as high as the first 15 clients using HFMAML.
Figure 5.6 and Figure 5.8 show that the increase in TAR@FAR 0.1 due to tuning is lowest for embedding regularization scenarios. Finding a good global model seems to come at the cost of finding good personalized models.

5.5 Fairness analysis

The last part of the analysis will look at the experimental results concerning fairness. Figure 5.10, Figure 5.11 and 5.12 show the average standard deviation per scenario for all tree data partitions.
There are five general trends visible that are important to explore further. The first of these trends is the effect on the standard deviation of embedding regularization. The regularization term seems to show evidence in Figure 5.11 of a lower standard deviation compared to its global classification layer model counterpart. This improvement only holds for the HFMAML model, where the difference is most accentuated on the tuned model of the last 5 clients. On the attribute-based partition in Figure 5.12 the regularization model does not show any significant decrease in the average standard deviation, only an increase.
The second trend is that there is no evidence found of q-FedAvg having a lower standard deviation than its global classification layer counterpart model. This means that there is no evidence found in this thesis that q-FedAvg can increase the fairness, where the embedding regularization might improve fairness in certain situations on the lognormal partition.
The third trend is how local classification layer models show evidence of decreasing the average standard deviation combined with HFMAML on the attribute-based dataset in Figure 5.12, except for the tuned model evaluation performance on the first 15 clients. Since this trend is not measured on the equal class partition in Figure 5.10 and to a lesser degree on the lognormal class partition in Figure 5.11 there might be a connection between improved fairness and using a local classification layer on the attribute-based dataset using

HFMAML.

The fourth trend is that there is evidence of HFMAML achieving a lower average standard deviation than FL. This decrease is the largest and most consistent on the attribute-based partition in Figure 5.12 and on the equal class partition with a global classification layer shown in Figure 5.10. On the lognormal class partition in Figure 5.11 FL achieves a lower average standard deviation on the untuned evaluation for the first 15 clients. HFMAML scenarios have a lower standard deviation after tuning. The difference for the last 5 clients is small, lacking evidence of a difference in standard deviation. The biggest difference in standard deviation is observed on the attribute-based partition as can be seen in Figure 5.12, specifically on the first 15 clients after tuning. There is also evidence of a lower standard deviation of the HFMAML scenarios before tuning. This difference is less pronounced on the last 5 clients. On both the lognormal class partition and the attribute-based partition the average standard deviation is lower on the last 5 clients compared to the first 15 clients. This might be a result of the last 5 clients not being involved in the training process. During training some clients might have influenced the global model more than others, resulting in a larger difference in the TAR@FAR 0.1 scores. The new clients have not had this opportunity and the global model is equally unfamiliar with the data of all the new clients. Figure 5.12 shows that HFMAML is better at avoiding clients to influence the global model to perform better on its own data, resulting in the standard deviation being closer to that of the unseen clients 16-20. HFMAML influences fairness.

Lastly, comparing HFMAML and FedFR shows FEDFR performing well on the first 15 clients on the lognormal class partition, suggesting it is better at keeping the TAR@FAR 0.1 scores closer together than both HFMAML and FL. The reason for this better performance might be the evaluation performance on the outlier client 7 as shown in Figure 5.13 and Figure 5.14. The FedFR model scenario has the highest TAR@FAR 0.1 score of all scenarios. This might explain the lower standard deviation.

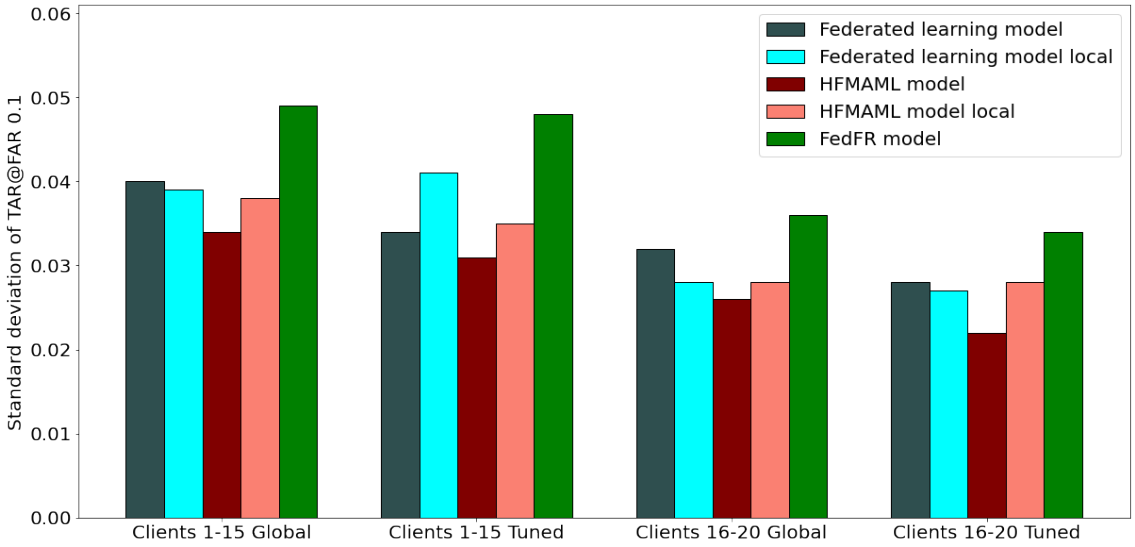

Figure 5.10: Standard deviation of client's global evaluation scores for the equal class partition dataset.

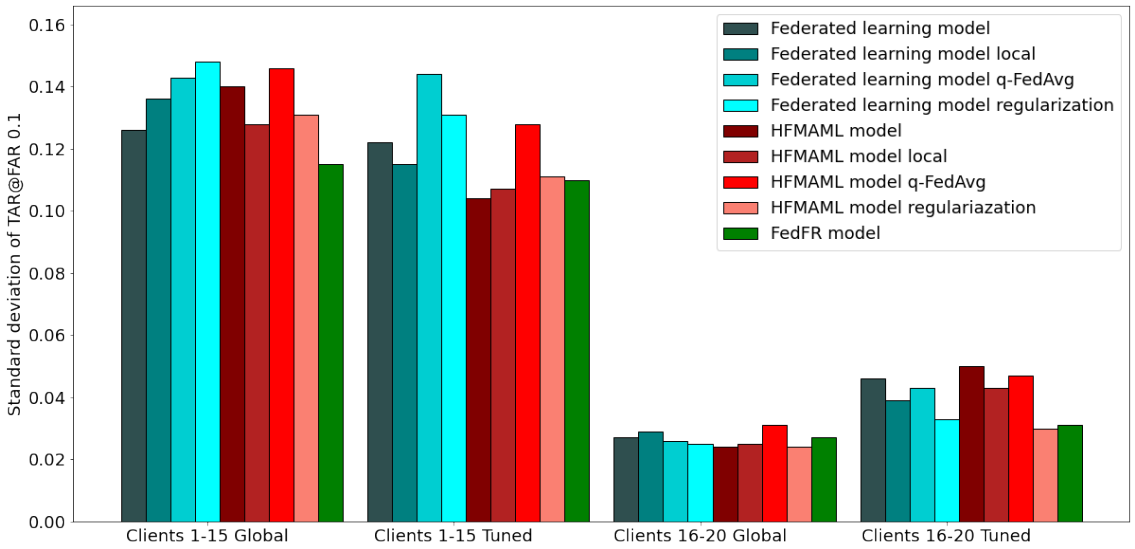

Figure 5.11: Standard deviation of client's global evaluation scores for the lognormal class partition dataset.

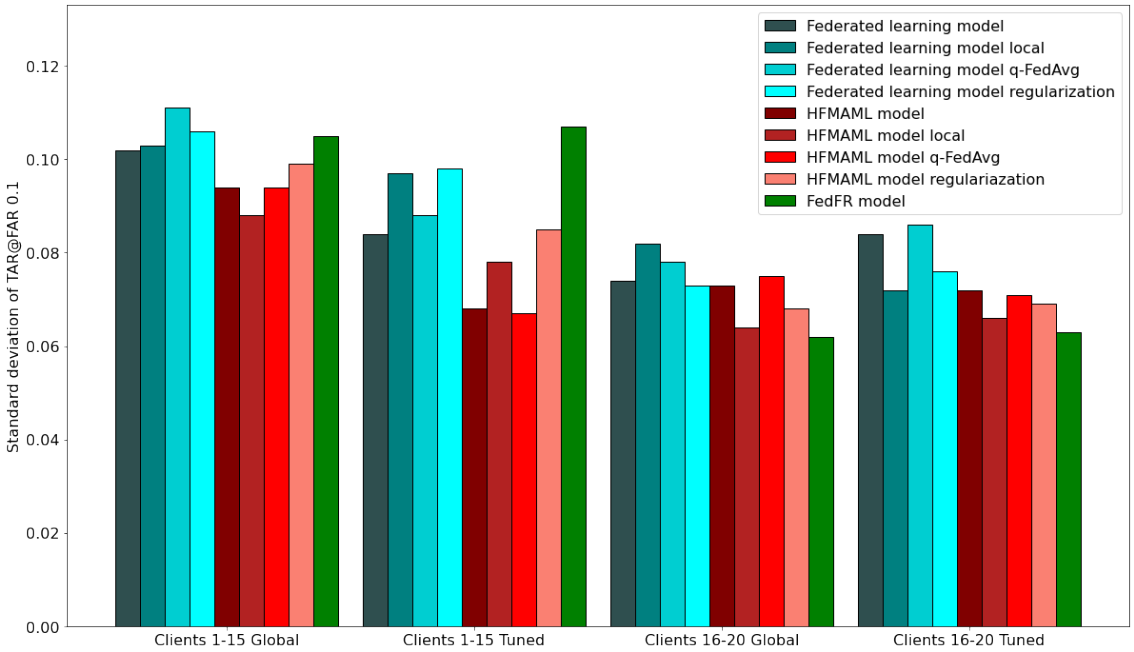

Figure 5.12: Standard deviation of client's evaluation scores for the attribute-based partition dataset.

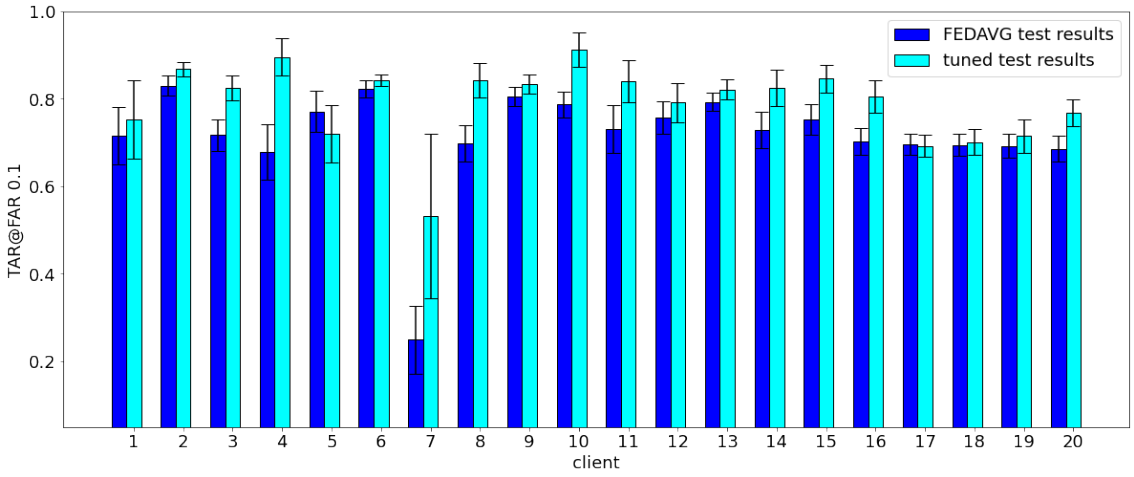

Figure 5.13: HFMAML model with lognormal class partition evaluation results. Client 7 has TAR@FAR 0.1 0.250 before tuning

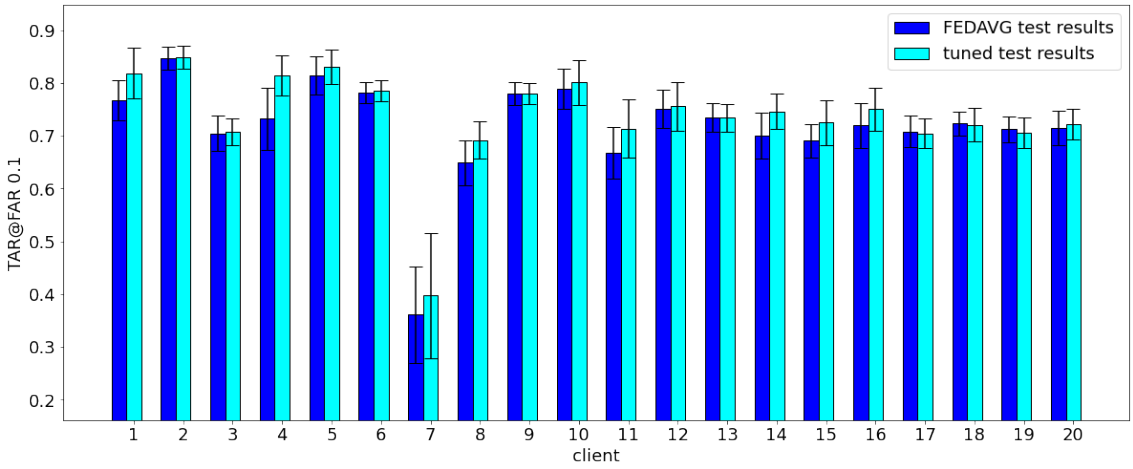

Figure 5.14: FedFR model with lognormal class partition evaluation results. Client 7 has TAR@FAR 0.1 0.361 before tuning

6

Conclusion

This thesis has focused on answering the question of how meta-learning can be used to improve personalized performance and fairness on existing clients and newly added clients in federated face recognition.

Implementing HFMAML has indeed improved verification performance under different levels of data heterogeneity. The results even show that HFMAML performs better when there is more data heterogeneity compared to the FL and FedFR benchmarks. This holds for both the existing- and the newly added clients. On top of that, HFMAML has been shown to decrease the standard deviation, which is most clearly visible on the attribute-based partition, suggesting that HFMAML limits clients from having too much influence compared to other clients on the global model and subsequently their local model too.

The results and methodology support the contributions made in this thesis that are stated in Section 1. Section 3 gives an overview of the HFMAML implementation in FFR and results show that this implementation works in an FFR setting.

The new types of data partitions based on CelebA gave new perspectives of how meta-learning performs on verification evaluation and fairness in an FFR environment. These two data partitions might be interesting to include in future research and to consider as benchmarks for developing future FFR models.

The embedding regularization shows how easily HFMAML can be combined with extensions to the loss function. Embedding regularization achieves higher verification scores before tuning, showing that a balance is possible between a good global model and a personalized local model. Future research might dive deeper into balancing between a good global- and personalized evaluation performance or looking at other model extensions that can improve HFMAML.

While implementing HFMAML, a decision needed to be made between keeping the classification layer local or sharing it with other clients which might increase privacy risks. This paper shows that keeping the classification layer local does not significantly impact the evaluation performance. Only tuning the model results in slightly lower verification scores, but this is compensated by slightly higher verification scores before tuning. All in all, it is not possible to say that there is a trade-off between verification performance and keeping the classification weights local.

Lastly, when looking at fairness the intended effect of q-FedAVg and embedding regularization does not seem visible in the data. Where embedding regularization achieves to lower the standard deviation under some circumstances there is no evidence of q-FedAVg to lower the standard deviation. Instead, HFMAML seemed to have the greatest effect on the standard deviation, especially on the attribute-based dataset.

All in all, the combination between FFR and HFMAML seems very promising, and hopefully future research will continue work on HFMAML and the new data partitions in FFR or use them as benchmarks for developing new and better algorithms.

References

[1] Acar, D.A.E., Zhao, Y., Navarro, R.M., Mattina, M., Whatmough, P.N. & Saligrama, V. (2021). Federated learning based on dynamic regularization. 6, 28

[2] Aggarwal, D., Zhou, J. & Jain, A.K. (2021). FedFace: Collaborative learning of face recognition model. 2, 15, 17, 34, 35, 42

[3] Arivazhagan, M.G., Aggarwal, V., Singh, A.K. & Choudhary, S. (2019). Federated learning with personalization layers. 11

[4] Bai, F., Wu, J., Shen, P., Li, S. & Zhou, S. (2021). Federated face recognition. 2

[5] Caldas, S., Duddu, S.M.K., Wu, P., Li, T., Konečný, J., McMahan, H.B., Smith, V. & Talwalkar, A. (2019). LEAF: A benchmark for federated settings. 15, 17, 34

[6] Chen, F., Luo, M., Dong, Z., Li, Z. & He, X. (2019). Federated meta-learning with fast convergence and efficient communication. 12

[7] Chen, Y., Wang, J., Yu, C., Gao, W. & Qin, X. (2021). FedHealth: A federated transfer learning framework for wearable healthcare. 5

[8] Deng, J., Guo, J., Yang, J., Xue, N., Kotsia, I. & Zafeiriou, S. (2021). ArcFace: Additive angular margin loss for deep face recognition. 1–1. 13, 14

[9] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, **29**, 141–142. 7

[10] Fallah, A., Mokhtari, A. & Ozdaglar, A. (2020). On the convergence theory of gradient-based model-agnostic meta-learning algorithms. 23, 25

[11] Fallah, A., Mokhtari, A. & Ozdaglar, A. (2020). Personalized federated learning: A meta-learning approach. 2, 12, 23, 25, 30, 42

[12] Finn, C., Abbeel, P. & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. 2, 12, 23, 24, 25, 30

[13] Geiping, J., Bauermeister, H., Dröge, H. & Moeller, M. (2020). Inverting gradients - how easy is it to break privacy in federated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan & H. Lin, eds., *Advances in Neural Information Processing Systems*, vol. 33, 16937–16947, Curran Associates, Inc. 31

[14] Guo, Y., Zhang, L., Hu, Y., He, X. & Gao, J. (2016). Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. 1, 16

[15] Hand, E.M. & Chellappa, R. (2016). Attributes for improved attributes: A multi-task network for attribute classification. 29

[16] Hanzely, F. & Richtárik, P. (2021). Federated learning of a mixture of global and local models. 11

[17] Harvey, J., Adam. LaPlace (2021). Exposing.ai. 1

[18] Harvey, J., Adam. LaPlace (2021). Exposing.ai. 16, 18

[19] Huang, G.B., Ramesh, M., Berg, T. & Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Tech. Rep. 07-49, University of Massachusetts, Amherst. 1, 13, 16, 42

[20] Jiang, Y., Konečný, J., Rush, K. & Kannan, S. (2019). Improving federated learning personalization via model agnostic meta learning. 2, 12, 23, 30, 35

[21] Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R.G.L., Eichner, H., Rouayheb, S.E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P.B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S.U., Sun, Z., Suresh, A.T., Tramèr, F., Vepakomma, P., Wang, J., Xiong,

L., Xu, Z., Yang, Q., Yu, F.X., Yu, H. & Zhao, S. (2021). Advances and open problems in federated learning. 6

[22] Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S.J., Stich, S.U. & Suresh, A.T. (2021). SCAFFOLD: Stochastic controlled averaging for federated learning. 10

[23] Kulkarni, V., Kulkarni, M. & Pant, A. (2020). Survey of personalization techniques for federated learning. Version: 1. 5, 11

[24] Li, L., Mu, X., Li, S. & Peng, H. (2020). A review of face recognition technology. **8**, 139110–139120, conference Name: IEEE Access. 1, 13

[25] Li, Q., Diao, Y., Chen, Q. & He, B. (2021). Federated learning on non-IID data silos: An experimental study. v, 6, 7, 9, 10, 11, 28

[26] Li, Q., He, B. & Song, D. (2021). Model-contrastive federated learning. 20

[27] Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A. & Smith, V. (2020). Federated optimization in heterogeneous networks. 7, 10, 31

[28] Li, T., Sanjabi, M., Beirami, A. & Smith, V. (2020). Fair resource allocation in federated learning. 2, 21, 23, 27, 28, 39, 42

[29] Li, T., Hu, S., Beirami, A. & Smith, V. (2021). Ditto: Fair and robust federated learning through personalization. 21

[30] Li, Z., Zhou, F., Chen, F. & Li, H. (2017). Meta-SGD: Learning to learn quickly for few-shot learning. 12

[31] Liu, C.T., Wang, C.Y., Chien, S.Y. & Lai, S.H. (2022). FedFR: Joint optimization federated framework for generic and personalized face recognition. **36**, 1656–1664, number: 2. 2, 15, 16, 17, 18, 30, 31, 34, 35, 39, 41, 42, 43

[32] Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B. & Song, L. (2018). Sphereface: Deep hypersphere embedding for face recognition. 14

[33] Liu, Z., Luo, P., Wang, X. & Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*. 16

[34] Maze, B., Adams, J., Duncan, J.A., Kalka, N., Miller, T., Otto, C., Jain, A.K., Niggel, W.T., Anderson, J., Cheney, J. & Grother, P. (2018). IARPA janus benchmark - c: Face dataset and protocol. In *2018 International Conference on Biometrics (ICB)*, 158–165, IEEE. 1, 42

[35] McMahan, H.B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B.A. (2016). Communication-efficient learning of deep networks from decentralized data. 1, 4, 5, 6, 8

[36] Miller, D., Brossard, E., Seitz, S. & Kemelmacher-Shlizerman, I. (2015). Megaface: A million faces for recognition at scale. 16

[37] Murgia, M. (2019). Microsoft quietly deletes largest public face recognition data set. 16, 18

[38] Nazir, M., Shakil, S. & Khurshid, K. (2021). Role of deep learning in brain tumor detection and classification (2015 to 2020): A review. **91**, 101940. 1

[39] Nech, A. & Kemelmacher-Shlizerman, I. (2017). Level playing field for million scale face recognition. 1

[40] Nichol, A., Achiam, J. & Schulman, J. (2018). On first-order meta-learning algorithms. 12

[41] Niu, Y. & Deng, W. (2021). Federated learning for face recognition with gradient correction. 15, 16, 17, 28, 30, 35, 42

[42] Parkhi, O.M., Vedaldi, A. & Zisserman, A. (2015). Deep face recognition. In *Procedings of the British Machine Vision Conference 2015*, 41.1–41.12, British Machine Vision Association. 16

[43] Qu, L., Balachandar, N. & Rubin, D.L. (2021). An experimental study of data heterogeneity in federated learning methods for medical imaging. 7

[44] Schroff, F., Kalenichenko, D. & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815–823. 13

[45] Shang, E., Yang, Z., Liu, H., Du, J. & Wang, X. (2022). FedFR: Evaluation and selection of loss functions for federated face recognition. In H. Gao, X. Wang,

W. Wei & T. Dagiuklas, eds., *Collaborative Computing: Networking, Applications and Worksharing*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 95–114, Springer Nature Switzerland. 13, 16, 17, 42

[46] Smith, V., Chiang, C.K., Sanjabi, M. & Talwalkar, A.S. (2017). Federated multi-task learning. 11, 13

[47] Student (1908). The probable error of a mean. *Biometrika*, 1–25. 44

[48] Sun, Y., Wang, X. & Tang, X. (2014). Deep learning face representation by joint identification-verification. 14

[49] Tan, A.Z., Yu, H., Cui, L. & Yang, Q. (2022). Towards personalized federated learning. 1–17, conference Name: IEEE Transactions on Neural Networks and Learning Systems. 9, 10

[50] Vanschoren, J. (2018). Meta-learning: A survey. 2

[51] Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, I–I. 37

[52] Wang, F., Xiang, X., Cheng, J. & Yuille, A.L. (2017). NormFace: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, 1041–1049. 14

[53] Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z. & Liu, W. (2018). CosFace: Large margin cosine loss for deep face recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5265–5274, IEEE. 14

[54] Wang, J., Liu, Q., Liang, H., Joshi, G. & Poor, H.V. (2020). Tackling the objective inconsistency problem in heterogeneous federated optimization. 6, 10

[55] Wang, K., Mathews, R., Kiddon, C., Eichner, H., Beaufays, F. & Ramage, D. (2019). Federated evaluation of on-device personalization. 5

[56] Wen, Y., Liu, W., Weller, A., Raj, B. & Singh, R. (2022). Sphereface2: Binary classification is all you need for deep face recognition. 20

[57] Whitelam, C., Taborsky, E., Blanton, A., Maze, B., Adams, J., Miller, T., Kalka, N., Jain, A.K., Duncan, J.A., Allen, K., Cheney, J. & Grother, P. (2017). IARPA janus benchmark-b face dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 592–600, IEEE. 42

[58] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, **1**, 80–83. 44

[59] Yang, Q., Liu, Y., Chen, T. & Tong, Y. (2019). Federated machine learning: Concept and applications. 4

[60] Yi, D., Lei, Z., Liao, S. & Li, S.Z. (2014). Learning face representation from scratch. *CoRR*, **abs/1411.7923**. 15

[61] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D. & Chandra, V. (2018). Federated learning with non-iid data. v, 2, 8, 9

[62] Zhu, H., Xu, J., Liu, S. & Jin, Y. (2021). Federated learning on non-iid data: A survey. 6, 8

Appendix A

Tables with results

Table A.1: Results per scenario as shown in Table 4.4.

Algorithm	Data dist.		Client 1-15			Client 16-20		
			FEDAVG	Tuned	Increase	FEDAVG	Tuned	Increase
FL	Equal		0.783	0.836	6.8%	0.71	0.76	7.0%
FL	Equal	Local	0.779	0.817	4.9%	0.718	0.74	3.1%
HFMAML	Equal		0.795	0.855	7.5%	0.705	0.762	8.1%
HFMAML	Equal	Local	0.804	0.852	6.0%	0.737	0.768	4.2%
FedFR	Equal		0.777	0.777	0.0%	0.75	0.754	0.5%
FL	Lognorm		0.707	0.788	11.5%	0.675	0.715	5.9%
FL	Lognorm	Local	0.715	0.786	9.9%	0.686	0.707	3.1%
FL	Lognorm	Fair	0.738	0.795	7.7%	0.681	0.706	3.7%
FL	Lognorm	Penalty	0.709	0.773	9.0%	0.704	0.715	1.6%
HFMAML	Lognorm		0.722	0.81	12.2%	0.695	0.737	6.0%
HFMAML	Lognorm	Local	0.725	0.802	10.6%	0.705	0.731	3.7%
HFMAML	Lognorm	Fair	0.74	0.805	8.8%	0.69	0.722	4.6%
HFMAML	Lognorm	Penalty	0.734	0.798	8.7%	0.726	0.735	1.2%
FedFR	Lognorm		0.718	0.743	3.5%	0.715	0.721	0.8%
FL	Attributes		0.718	0.788	9.7%	0.572	0.672	17.5%
FL	Attributes	Local	0.755	0.772	2.3%	0.635	0.695	9.4%
FL	Attributes	Fair	0.718	0.785	9.3%	0.571	0.669	17.2%
FL	Attributes	Penalty	0.764	0.779	2.0%	0.673	0.694	3.1%
HFMAML	Attributes		0.784	0.838	6.9%	0.653	0.755	15.6%
HFMAML	Attributes	Local	0.789	0.816	3.4%	0.679	0.759	11.8%
HFMAML	Attributes	Fair	0.777	0.837	7.7%	0.645	0.748	16.0%
HFMAML	Attributes	Penalty	0.789	0.812	2.9%	0.708	0.729	3.0%
FedFR	Attributes		0.726	0.724	-0.3%	0.65	0.653	0.5%

Table A.2: Results per scenario as shown in Table 4.4 for the standard deviation.

Algorithm	Data dist.		Client 1-15		Client 16-20	
			Global	Tuned	Global	Tuned
FL	Equal		0.040	0.034	0.032	0.028
FL	Equal	Local	0.039	0.041	0.028	0.027
HFMAML	Equal		0.034	0.031	0.026	0.022
HFMAML	Equal	Local	0.038	0.035	0.028	0.028
FedFR	Equal		0.049	0.048	0.036	0.034
FL	Lognorm		0.126	0.122	0.027	0.046
FL	Lognorm	Local	0.136	0.115	0.029	0.039
FL	Lognorm	Fair	0.143	0.144	0.026	0.043
FL	Lognorm	Penalty	0.148	0.131	0.025	0.033
HFMAML	Lognorm		0.140	0.104	0.024	0.050
HFMAML	Lognorm	Local	0.128	0.107	0.025	0.043
HFMAML	Lognorm	Fair	0.146	0.128	0.031	0.047
HFMAML	Lognorm	Penalty	0.131	0.111	0.024	0.030
FedFR	Lognorm		0.115	0.110	0.027	0.031
FL	Attributes		0.089	0.102	0.078	0.074
FL	Attributes	Local	0.101	0.103	0.070	0.082
FL	Attributes	Fair	0.091	0.111	0.085	0.078
FL	Attributes	Penalty	0.097	0.106	0.075	0.073
HFMAML	Attributes		0.094	0.068	0.073	0.072
HFMAML	Attributes	Local	0.088	0.078	0.064	0.066
HFMAML	Attributes	Fair	0.094	0.067	0.075	0.071
HFMAML	Attributes	Penalty	0.099	0.085	0.068	0.069
FedFR	Attributes		0.105	0.107	0.062	0.063

Appendix B

Statistical tests for comparing TAR@FAR scores

The values that are marked in bold in the tables with the statistical tests correspond to the hypotheses that have not been rejected with $\alpha = 0.05$

Table B.1: statistical p-values for t-test and Wilcoxon test. FL and HFMAML global classification layer model with equal class partition scenarios are compared

		T-test p-value	Wilcoxon test p-value
clients 1-15	Global	1.08e-15	3.48e-9
	Tuned	2.79e-32	7.42e-10
clients 16-20	Global	**1.60e-1**	**1.04e-1**
	Tuned	**5.80e-1**	**7.23e-1**

Table B.2: statistical p-values for t-test and Wilcoxon test. Local and global classification layer scenarios are compared with equal class partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	3.70e-3	3.69e-2	3.48e-8	3.03e-6
	Tuned	1.54e-27	1.10e-9	4.11e-2	**5.70e-2**
clients 16-20	Global	5.60e-3	5.00e-3	3.05e-17	2.09e-9
	Tuned	5.27e-12	1.12e-8	2.58e-2	3.45e-2

Table B.3: statistical p-values for t-test and Wilcoxon test. Within the HFMAML with lognormal class partition scenarios regularization is compared to non-regularization

		HFMAML	
		T-test p-value	Wilcoxon test p-value
clients 1-15	Global	5.14E-06	5.82E-06
	Tuned	1.89E-05	5.36E-04
clients 16-20	Global	8.48E-25	7.53E-10
	Tuned	**5.33E-01**	**6.83E-01**

Table B.4: statistical p-values for t-test and Wilcoxon test. FL is compared to HFMAML in the lognormal class partition scenarios

		Global Arcface weights		Local Arcface weights	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	5.45E-08	3.30E-05	3.21E-04	2.62E-04
	Tuned	7.68E-13	5.79E-09	7.68E-13	5.79E-09
clients 16-20	Global	7.76E-12	1.35E-07	1.86E-09	7.13E-07
	Tuned	4.99E-11	1.39E-07	4.90E-11	1.39E-07

Table B.5: statistical p-values for t-test and Wilcoxon test. Global classification layer and local classification layer scenarios are compared

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	5.34E-03	6.60E-03	**3.04E-01**	**4.04E-01**
	Tuned	**4.40E-01**	**5.74E-01**	4.75E-03	1.07E-02
clients 16-20	Global	6.85E-05	2.10E-04	2.33E-04	1.13E-04
	Tuned	9.94E-03	5.67E-03	3.13E-02	4.54E-02

Table B.6: statistical p-values for t-test and Wilcoxon test. FL and HFMAML algorithm scenarios are compared for the attribute-based partition.

		Global Arcface weights		Local Arcface weights	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	3.62E-65	7.46E-10	3.67E-39	7.47E-10
	Tuned	2.79E-50	7.49E-10	3.92E-53	7.50E-10
clients 16-20	Global	6.89E-48	7.54E-10	3.39E-32	7.53E-10
	Tuned	3.32E-44	7.53E-10	1.45E-45	7.52E-10

Table B.7: statistical p-values for t-test and Wilcoxon test. Global- and local classification layer scenarios are compared for the attribute-based partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	7.46E-43	7.47E-10	6.14E-04	9.75E-05
	Tuned	3.07E-14	1.62E-07	1.39E-18	8.23E-10
clients 16-20	Global	3.68E-42	7.53E-10	2.75E-16	9.77E-16
	Tuned	1.43E-11	3.08E-08	**2.28E-01**	**3.84E-01**

Table B.8: statistical p-values for t-test and Wilcoxon test. Embedding regularization and non-embedding regularization scenarios are compared for the attribute-based partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	4.70E-48	7.52E-10	1.53E-03	8.10E-03
	Tuned	3.16E-06	2.50E-04	1.32E-03	8.00E-10
clients 16-20	Global	1.50E-56	7.53E-10	5.30E-34	7.50E-10
	Tuned	4.33E-10	3.71E-07	5.84E-15	3.43E-09

Appendix C

Statistical tests for comparing standard deviations

The values that are marked in bold in the tables with the statistical tests correspond to the hypotheses that have not been rejected with $\alpha = 0.05$

Table C.1: statistical p-values for t-test and Wilcoxon test. This table tests significance for the difference in standard deviation between FL and HFMAML algorithms for the equal class partition.

		Global classification		Local classification	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	5.74E-07	1.77E-05	**3.20E-01**	**3.89E-01**
	Tuned	1.51E-03	7.96E-04	5.24E-06	1.18E-04
clients 16-20	Global	1.85E-03	2.49E-03	**8.29E-01**	**7.43E-01**
	Tuned	8.49E-04	2.49E-03	**6.35E-01**	**5.02E-01**

Table C.2: statistical p-values for t-test and Wilcoxon test. This table tests significance for the difference in standard deviation between global and local classification layer algorithms on the equal class partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	**7.93E-01**	**8.61E-01**	5.36E-04	1.13E-03
	Tuned	5.15E-09	9.07E-07	9.06E-05	3.32E-05
clients 16-20	Global	2.46E-02	4.56E-02	**5.27E-01**	**6.91E-01**
	Tuned	**6.39E-01**	**6.16E-01**	1.41E-04	3.61E-04

Table C.3: statistical p-values for t-test and Wilcoxon test. This table tests significance for the difference in standard deviation of the FL and HFMAML algorithms for the lognormal class partition.

		Global classification		Local classification	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	6.83E-04	2.00E-03	3.54E-02	**5.12E-02**
	Tuned	1.61E-03	1.69E-03	**1.31E-01**	**9.11E-02**
clients 16-20	Global	**6.04E-02**	**6.48E-02**	2.53E-02	4.94E-02
	Tuned	**1.05E-01**	**8.84E-02**	**1.04E-01**	**1.05E-01**

Table C.4: statistical p-values for t-test and Wilcoxon test. This table tests significance for the difference in standard deviation of the global and local classification layer algorithms on the lognormal class partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	9.01E-03	2.09E-03	3.37E-03	2.18E-03
	Tuned	**1.63E-01**	**4.79E-01**	**5.29E-01**	**4.06E-01**
clients 16-20	Global	**2.82E-01**	**2.11E-01**	**5.08E-01**	**3.84E-01**
	Tuned	9.57E-03	4.89E-02	1.05E-02	8.38E-03

Table C.5: statistical p-values for t-test and Wilcoxon test. This table shows whether the addition of embedding regularization can significantly decrease the standard deviation in the lognormal class partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	1.18E-06	2.98E-05	8.14E-03	9.69E-03
	Tuned	**9.03E-02**	**5.00E-02**	**1.99E-01**	**2.53E-01**
clients 16-20	Global	**2.71E-01**	**2.06E-01**	**9.90E-01**	**7.14E-01**
	Tuned	1.94E-06	1.66E-05	1.95E-11	7.32E-07

Table C.6: Statistical p-values for t-test and Wilcoxon test. This table shows whether the addition of q-FedAvg can significantly decrease the standard deviation in the lognormal class partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	1.66E-05	9.80E-05	**7.93E-02**	**7.10E-02**
	Tuned	1.33E-05	1.98E-05	7.99E-06	2.53E-04
clients 16-20	Global	**4.18E-01**	**4.88E-01**	6.23E-04	1.79E-03
	Tuned	**5.09E-01**	**6.92E-01**	**2.96E-01**	**2.53E-01**

Table C.7: statistical p-values for t-test and Wilcoxon test. This table tests significance of the difference in standard deviation of the FedFR and HFMAML algorithms for the lognormal class partition.

		Glocal classification		Local classification	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	7.50E-10	5.59E-06	2.78E-03	2.99E-03
	Tuned	**2.82E-01**	**2.20E-01**	**5.63E-01**	**4.32E-01**
clients 16-20	Global	**1.03E-01**	**1.96E-01**	**3.13E-01**	**6.37E-01**
	Tuned	4.89E-10	8.72E-07	1.96E-06	7.48E-06

Table C.8: statistical p-values for t-test and Wilcoxon test. This table tests significance of the difference in standard deviation of the FL and HFMAML algorithms for the attribute-based partition.

		Global classification		Local classification	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	2.10E-04	1.00E-04	2.92E-11	3.61E-07
	Tuned	9.85E-12	4.10E-09	9.50E-23	1.29E-09
clients 16-20	Global	**8.29E-01**	**6.58E-01**	5.61E-11	3.34E-07
	Tuned	2.92E-04	9.44E-04	3.49E-06	4.10E-04

Table C.9: statistical p-values for t-test and Wilcoxon test. This table tests significance of the difference in standard deviation of the global classification layer and local classification layer algorithms for the attribute-based partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	**7.39E-01**	**6.36E-01**	7.78E-03	9.39E-03
	Tuned	5.11E-17	2.42E-09	8.62E-12	1.06E-08
clients 16-20	Global	2.83E-02	1.51E-02	2.12E-04	1.99E-04
	Tuned	**6.57E-02**	2.93E-02	2.90E-02	4.94E-02

Table C.10: statistical p-values for t-test and Wilcoxon test. This table shows whether the addition of embedding regularization can significantly decrease the standard deviation in the attribute-based partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	2.14E-02	3.80E-02	1.09E-02	**5.87E-02**
	Tuned	1.74E-13	1.89E-08	1.06E-12	2.46E-08
clients 16-20	Global	**9.15E-01**	**8.97E-01**	**5.58E-02**	**5.38E-02**
	Tuned	6.06E-04	6.54E-04	**1.57E-01**	**1.86E-01**

Table C.11: statistical p-values for t-test and Wilcoxon test. This table shows whether the addition of q-FedAvg can significantly decrease the standard deviation in the attribute-based partition.

		FL		HFMAML	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	2.17E-06	1.59E-04	**7.56E-01**	**7.71E-01**
	Tuned	1.76E-02	1.11E-02	**4.76E-01**	**4.78E-01**
clients 16-20	Global	**2.20E-01**	**1.80E-01**	**4.19E-01**	**3.32E-01**
	Tuned	**3.70E-01**	**4.23E-01**	**6.93E-01**	**9.44E-01**

Table C.12: statistical p-values for t-test and Wilcoxon test. This table tests significance of the difference in standard deviation of the FedFR and HFMAML algorithms for the attribute-based partition.

		Global classification		Local classification	
		T-test p-value	Wilcoxon test p-value	T-test p-value	Wilcoxon test p-value
clients 1-15	Global	3.80E-07	2.92E-06	1.15E-17	8.45E-09
	Tuned	3.69E-37	7.51E-10	1.84E-36	7.51E-10
clients 16-20	Global	1.25E-05	8.95E-06	**5.02E-01**	**2.88E-01**
	Tuned	5.00E-03	8.02E-03	**3.97E-01**	**5.38E-01**