Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Studie zur

fachlichen Einschätzung und Prüfung des Potenzials von Federated-Learning-Algorithmen in der amtlichen Statistik

Abschlussbericht

Joshua Stock, Tom Petersen, David Zadim, Hannes Federrath

Universität Hamburg

Fachbereich Informatik

Arbeitsbereich SVS

erarbeitet für das

Statistische Bundesamt (Destatis)

14. Februar 2023

Zusammenfassung

Das statistische Bundesamt hat den Arbeitsbereich Sicherheit in verteilten Systemen damit beauftragt, eine Studie zur fachlichen Einschätzung und Prüfung des Potenzials von Federated Learning (FL) Algorithmen mit Fokus auf die amtliche Statistik durchzuführen. Der vorliegende Abschlussbericht fasst die gewonnenen Erkenntnisse zusammen, die während der Durchführung von August bis Dezember 2023 gewonnen wurden.

Der Begriff FL wurde das erste Mal von McMahan u. a. in [McM+17] verwendet und beschreibt eine spezielle Form des verteilten Trainierens von Modellen im Bereich Maschinelles Lernen (ML). Seit dem erstmaligem Erscheinen wird FL oft auch als Synonym zu verteiltem ML verwendet, wobei in diesem Feld bereits vor Begriffseinführung umfangreiche Forschungen durchgeführt wurden.

Einerseits wurde im Rahmen der Studie eine umfassende Literaturrecherche betrieben, die von verschiedenen FL-Arten und verwandten Konzepten, unter anderem über die Umsetzung von ML-Algorithmen in FL-Szenarien, bestehenden Frameworks und praktischen Fallstricken bis hin zu Privatsphäreaspekten des FL reicht. Dabei wurde stets ein Augenmerkt auf die konkreten Implikationen für die amtliche Statistik gelegt.

Andererseits enthält der Bericht eine ausführliche Zusammenfassung von 13 Anwendungsbeispielen in der Literatur, in denen FL in Szenarien eingesetzt wird, die der amtlichen Statistik nahestehen. Auf zwei Anwendungsbeispiele wird dabei genauer eingegangen, inklusive ihrer Übertragbarkeit auf die amtliche Statistik.

Außerdem wurden im Rahmen der Studie Simulationen mit drei verschiedenen Datensätzen vorgenommen, wodurch Rückschlüsse auf den praktischen Einsatz von FL gezogen werden können. Während die Beschreibung der Simulationen in eine eigene Veröffentlichung ausgelagert wurde (siehe Anhang B), fließen die dabei gemachten Erfahrungen maßgeblich in das Fazit dieses Abschlussberichts mit ein.

Insgesamt ist die Anwendbarkeit von FL für die amtliche Statistik durchaus gegeben. Während diese Studie in den Bereichen der praktischen Umsetzung und Datenschutzfragen lediglich erste Ansatzpunkte schafft, lässt insbesondere die umfassende Literaturrecherche erste Schlüsse zu. Die systematische Fokussierung auf den *Cross-Silo*-Ansatz des FL und der vorübergehende Ausschluss von Modellpersonalisierungen für Anwendungen des statistischen Bundesamtes lassen viele Möglichkeiten der praktischen Umsetzung offen. Während im beschränkten Umfang der Simulationen keine Experimente zu heterogenen Datensätzen und der Anwendbarkeit verschiedener Frameworks durchgeführt werden konnten (auch hierzu wurden jedoch Informationen aus der Literatur zusammengetragen), konnte gezeigt werden, dass unter bestimmten Voraussetzungen FL-Algorithmen mit der Performance von zentralisiert trainierten ML-Modellen mithalten können. Die Simulationen haben außerdem gezeigt, dass vor allem das Vorhandensein von Trainingsdaten in ausreichendem Maße, sowie sorgfältig ausgewählte Hyperparameter eine große Rolle im Gelingen eines FL-Trainings spielen.

Inhaltsverzeichnis

Akronyme 6

1 Einleitung 8

2 Federated Learning 9

3 Arten des Federated Learnings und verwandte Konzepte 11
- 3.1 Cross-Device Federated Learning . . . 11
- 3.2 Cross-Silo Federated Learning . . . 11
- 3.3 Fully Decentralized Learning . . . 12
- 3.4 (Cyclic) Institutional Incremental Learning . . . 13
- 3.5 Split Learning . . . 13
- 3.6 Verteiltes Lernen aus Performancegründen . . . 14
- 3.7 Auswirkungen auf die Studie . . . 14

4 Verschiedene Methoden für Federated Learning 16
- 4.1 Baumbasierte Methoden . . . 16
- 4.2 Methoden für Deep Learning . . . 17
- 4.3 Einfache Methoden . . . 18
- 4.4 Support Vector Machines . . . 18
- 4.5 Auswirkungen auf die Studie . . . 18

5 Bestehende Frameworks 20

6 Probleme durch Heterogenität der lokalen Datensätze 22
- 6.1 Mögliche Ursachen . . . 22
- 6.2 Auswirkungen auf die Studie . . . 23

7 Personalisierung 24
- 7.1 Global Model Personalization . . . 24
- 7.2 Solution Personalization . . . 26
- 7.3 Auswirkungen auf die Studie . . . 26

8 Einfluss von Netzwerk-Topologien und Hyperparametern auf FL-Ergebnisse 27

9 Privatsphäre-Angriffe im Kontext von Federated Learning 29
- 9.1 Angreifermodelle . . . 29
- 9.2 Privatsphäreangriffe . . . 30
- 9.3 Auswirkungen auf die Studie . . . 30

10 Privacy-Preserving Machine Learning 32
- 10.1 Techniken zum Schutz der Privatsphäre . . . 32
- 10.2 Auswirkungen auf die Studie . . . 33

11 Bewertung der Ergebnisse der Literaturrecherche bezogen auf die amtl. Statistik 34

12 Anwendungsfälle aus der Literatur 35
12.1 Anwendungsbeispiel 1: Lungenkrebs-Klassifizierung 35
12.2 Anwendungsbeispiel 2: Feinstaubvorhersage 37

13 Schlussbemerkungen 39

Literatur 40

A Anwendungsfälle von Federated Learning (Literaturübersicht) 49

B Veröffentlichung der Simulationsergebnisse 55

Akronyme

ADMM Alternating Direction Method of Multipliers 35, 49

AUC Area under the ROC Curve 36, 49–52

CNN Convolutional Neural Network 51, 52, 54

CT Computertomographie 51, 52

DAN Deep Averaging Network 53

DNN Deep Neural Network 14, 16, 54

DP Differential Privacy 10, 32, 49

DSC Dice Similarity Coefficient 52

FedAvg Federated Averaging 12, 17–19, 22, 49, 51–54

FedDyn Federated Dynamic Regularizer 17

FedSGD Federated Stochastic Gradient Descent 17, 18

FL Federated Learning 2, 8–18, 20, 23–30, 32–37, 39, 52, 53

HE Homomorphic Encryption 17, 32, 51

LSTM Long Short-Term Memory Network 37, 53

ML Maschinelles Lernen 2, 9, 14, 18, 20, 35, 38, 39, 49–54

MRT Magnetresonanztomographie 49, 52

NLP Natural Language Processing 53

PCA Principal component analysis 54

PPML Privacy-Preserving Machine Learning 32–34

RMSE Root Mean Square Error 36, 49, 54

SGD Stochastic Gradient Descent 10, 12, 16, 18

SMPC Secure Multi-Party Computation 30, 32, 34

Sub-FedAvg Subnetwork Federated Averaging 17

SVM Support Vector Machine 18, 50, 53

TEE Trusted Execution Environment 30

1 Einleitung

Diese Studie wurde im Auftrag des statistischen Bundesamt als Expertise zur fachlichen Einschätzung und Prüfung des Potenzials von Federated Learning (FL) Algorithmen in der amtlichen Statistik durchgeführt, insbesondere im Hinblick auf den gesetzlichen Auftrag des Statistischen Bundesamtes, der Bundesregierung und Öffentlichkeit neue Informationen bereitzustellen. Grundsätzlich wird FL im Rahmen der Studie nicht als Methode für bessere Ergebnisse gegenüber einer zentralen Sammlung und Auswertung von Daten betrachtet und bewertet. Vielmehr kann FL als Technologie zur Ermöglichung der Gewinnung von statistischen Zusammenhängen in Kontexten gesehen werden, in denen kein Zugriff auf Rohdaten möglich ist – sei es aus rechtlichen oder wirtschaftlichen Erwägungen. Hierbei kann das Statistische Bundesamt als zentrale Partei in einem verteilten Lernverfahren mit den jeweils teilnehmenden datenhaltenden Parteien ein Modell trainieren. Ein Zugriff auf die Rohdaten der Parteien ist dabei weder möglich noch nötig.

Dieser Abschlussbericht ist wie folgt gegliedert: In den Kapiteln 2 bis 10 erfolgt die Darstellung der Ergebnisse einer umfassenden Literaturrecherche zu FL. Hierbei werden sowohl die in der Ausschreibung des Statistischen Bundesamtes geforderten Schwerpunkte als auch aus unserer Sicht sinnvolle weitere Vertiefungen adressiert. Jeder Schwerpunkt wird dabei auch unter den Kriterien der amtlichen Statistik dargestellt, um die Bedeutung des jeweiligen Themas einzuordnen. In Kapitel 11 werden die Erkenntnisse der Literaturrecherche im Hinblick auf die Nutzung in der amtlichen Statistik zusammengefasst und bewertet. Anschließend stellt Kapitel 12 eine Übersicht über Anwendungsfälle von FL in der Literatur bereit, wobei sich eine ausführliche Tabelle in Anhang A befindet. Hierbei wurden Fälle aus verschiedenen Einsatzkontexten gewählt, die jeweils der amtlichen Statistik ähneln. Eigens im Rahmen dieser Studie durchgeführte Simulationen werden in einer separaten Publikation beschrieben und bewertet, die diesem Dokument in Anhang B beigefügt ist. Kapitel 13 rundet den Bericht mit einer abschließenden Bewertung ab.

2 Federated Learning

Eine grundlegende Definition des Begriffs FL lässt sich in [Kai+19] finden:

> *Federated Learning* beschreibt ein Maschinelles Lernen (ML)-Szenario, in dem mehrere Entitäten (*Clients*) kooperativ und unter Koordinierung durch eine zentrale Instanz (*Server* oder *Service Provider*) ein maschinelles Lernproblem lösen. Die Daten der Clients werden lokal gespeichert und nicht ausgetauscht oder übertragen, stattdessen wird das Lernziel durch gezielte Modellupdates und ihre unmittelbar erfolgende Aggregation erreicht.

In anderen Veröffentlichungen wird die zentrale Instanz auch als *Parameter-Server*, *Aggregations-Server* oder *Koordinator* bezeichnet [Yan+19b].

Formaler betrachtet beschreibt FL ein globales Optimierungsproblem auf lokalen Datensätzen[1] [Xu+21]: Für K datenhaltende Instanzen (Clients) mit jeweils n_k Datenpunkten aus der clientabhängigen Verteilung $\mathscr{D}_k$ – also einem Gesamtdatensatz der Größe $n = \sum_{k=1}^{K} n_k$ – sind Modellparameter w zu lernen, die folgendes Problem minimieren:

$$\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) := \sum_{k=1}^{K} \frac{n_k}{n} F_k(\mathbf{w}) \text{ mit } F_k(\mathbf{w}) := \frac{1}{n_k} \sum_{x_i \in \mathscr{D}_k} f_i(\mathbf{w})$$

Die Funktion f_i beschreibt hier eine von einem gelabelten Datum (x_i, y_i) abhängige und verfahrensspezifische Loss-Funktion. Beispiele für gängige einfache Loss-Funktionen lassen sich in [Xu+21] finden.

Ein gängiger FL-Trainingsprozess[2] kann folgendermaßen beschrieben werden [Kai+19]. Ein Server koordiniert den Trainingsprozess, indem die folgenden Schritte iterativ[3] bis zur Erreichung des Trainingsziels durchgeführt werden:

0. **Initialisierung** (nur in der ersten Runde): Je nach Modellart können Modellparameter mit Nullwerten, Zufallszahlen oder anderen geeigneten Verteilungen initialisiert werden. Sofern dem Server eigene Daten zur Verfügung stehen, kann auch ein sogenanntes *Pre-Training* vorgenommen werden. Diese Möglichkeit wird in [Che+22] umfassend beleuchtet.
1. **Auswahl von Clients:** Der Server wählt eine geeignete Untermenge aller Clients aus (etwa abhängig von Erreichbarkeit, Netzanbindung, freien Rechenressourcen, ...).

1. Ein Hinweis zur verwendeten Terminologie: Der Begriff *Datensatz* wird für eine Sammlung von Datenpunkten verwendet. *Datenpunkte* beschreiben Beobachtungen, die jeweils aus einer Menge von Attributwerten bestehen.
2. Genauer wird hier ein *Cross-Device*-FL-Prozess beschrieben. Unterschiede zu anderen Arten des FL werden in Kapitel 3 dargestellt.
3. [Kai+19] nennt hier 500 - 10.000 Runden als erwartete Anzahl bis zur Konvergenz des Modells.

2. **Broadcast:** Den Clients werden aktuelle Modellgewichte und ein Trainingsprogramm zur Verfügung gestellt.
3. **Berechnungen auf den Clients:** Jeder Client berechnet lokal ein Modellupdate durch Ausführung des Trainingsprogramms (z.B. mithilfe von Stochastic Gradient Descent (SGD) auf den lokalen Daten).
4. **Aggregation:** Der Server berechnet eine Aggregation der Modellupdates der Clients. In diesem Schritt können auch weitergehende Maßnahmen etwa in Form von Differential Privacy (DP) (siehe Kapitel 10) integriert werden.
5. **Modellupdate:** Der Server aktualisiert das globale Modell basierend auf der zuvor erhaltenen Aggregation.

Der Ablauf einer Trainingsrunde wird in Abbildung 2.1 dargestellt.

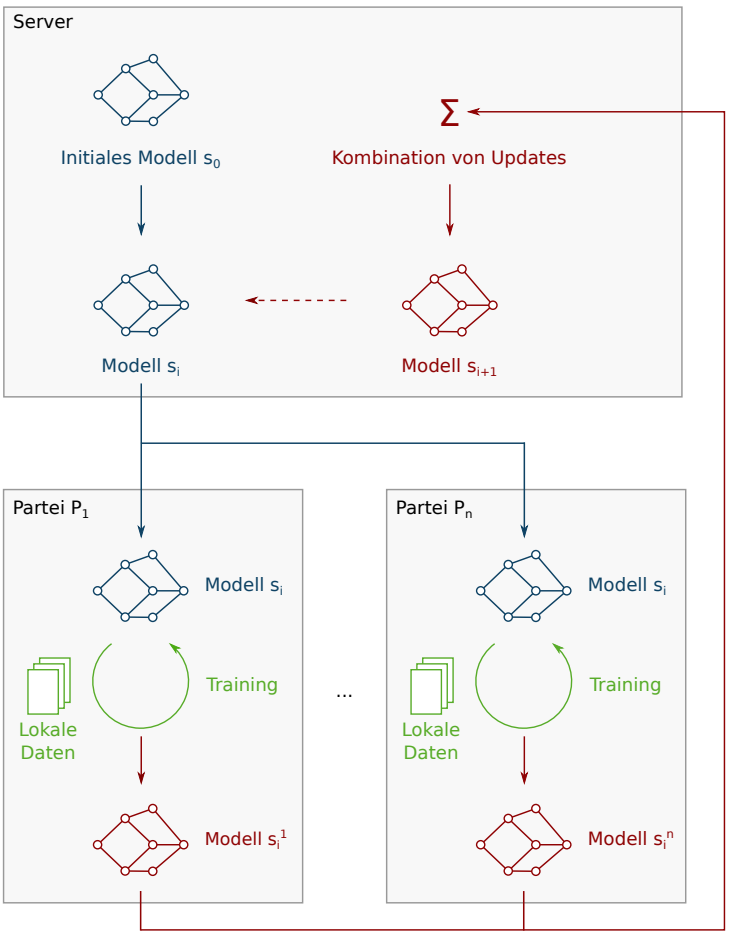

Abbildung 2.1: Eine vereinfachte Darstellung des FL-Prozesses für eine Runde i mit einem zentralen Server und n teilnehmenden Parteien. Entnommen aus [Sto+22].

3 Arten des Federated Learnings und verwandte Konzepte

FL kann einerseits in Unterarten unterteilt werden, andererseits gibt es zahlreiche verwandte Konzepte, wobei eine klare Abgrenzung nicht immer möglich ist. In [Kai+19] werden im Rahmen von FL insbesondere zwei Unterarten betrachtet: *Cross-Device* und *Cross-Silo* FL. Weiterhin werden in diesem Abschnitt Verfahren für verteiltes Lernen ohne zentrale Instanz, Split Learning und aus Performancegründen verteilte Lernverfahren dargestellt. Eine Übersicht der Konzepte bietet Abbildung 3.1.

3.1 Cross-Device Federated Learning

Cross-Device FL beschreibt eine Form des FL, bei der die Daten auf eine Vielzahl von Geräten (etwa Smartphones) verteilt sind und das Lernen dezentral auf den Geräten geschieht. Für die Koordination des Lernprozesses ist ein zentraler Server zuständig, der jedoch keinen Zugriff auf Rohdaten der Clients erhält. Im Vergleich zum *Cross-Silo FL* sind die Clients seltener erreichbar, potentiell schlechter an das Netz angebunden und unzuverlässig – es muss davon ausgegangen werden, dass Clients aus dem Lernprozesses aussteigen. Aus diesem Grund wird insbesondere im Cross-Device-Szenario auch an Möglichkeiten geforscht, die notwendige Kommunikation zwischen Parteien zu reduzieren oder auf kleinere Datenmengen zu beschränken. In jeder Trainingsrunde wird nur ein Bruchteil aller verfügbaren Clients am Lernprozess beteiligt, so dass der Trainingsprozess auf Clients keinen Zustand benötigt, sondern in jeder Runde von unbekannten, neuen Clients durchgeführt werden kann.

3.2 Cross-Silo Federated Learning

Im *Cross-Silo-FL*-Szenario sind die Daten auf eine begrenzte Anzahl von Datensilos (etwa Datenbanken in verschiedenen Organisationen wie Krankenhäusern) verteilt und das Lernen geschieht dezentral in den Silos. Für die Koordination des Lernprozesses ist ein zentraler Server zuständig, der wiederum keinen Zugriff auf Rohdaten der Clients erlangt. Im Vergleich zum *Cross-Device FL* sind die Silos grundsätzlich besser verfügbar, gut angebunden und zuverlässig. Hierdurch können alle Clients an jeder Runde des Lernprozesses beteiligt werden und, falls notwendig, auch Zustände über Runden hinweg aufrechterhalten werden. Im Rahmen von Cross-Silo FL kann – je nachdem wie die Daten in den Silos verteilt vorliegen – zwischen horizontalem (auch *Sample-Partitioned* oder *Example-Partitioned FL*) und vertikalem (auch *Feature-Partitioned FL*) Lernen sowie *Federated Transfer Learning* unterschieden werden [Yan+19c].

Im horizontalen Szenario liegen in allen Silos verschiedene Datensätze mit den jeweils gleichen erfassten Merkmalen vor (bspw. Kundendatenbanken in verschiedenen Banken).

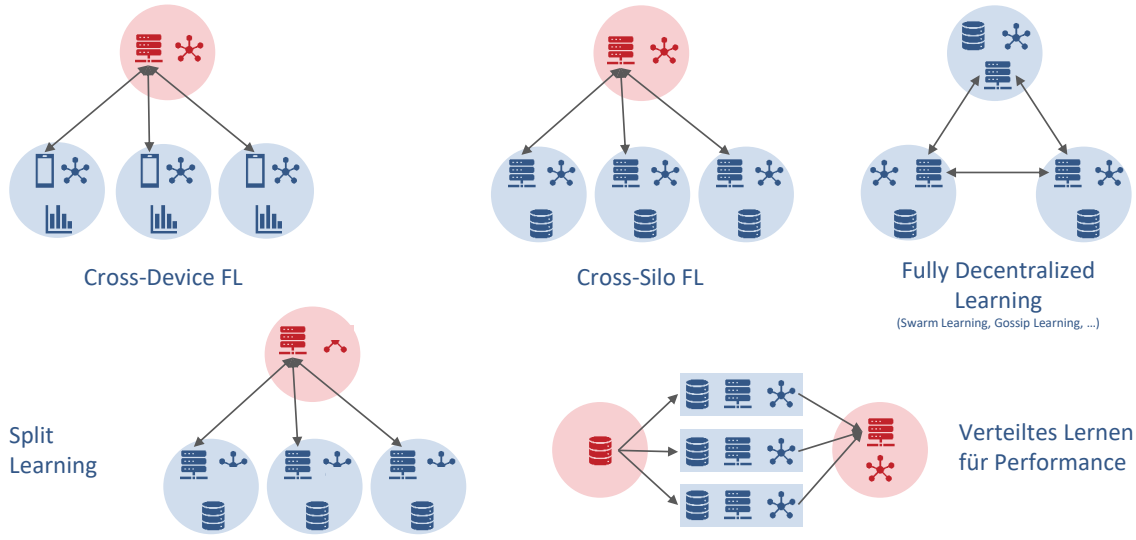

Abbildung 3.1: High-Level-Darstellung von FL-Arten und verwandten Konzepten.

Hierbei handelt es sich um das gängigste Szenario und Verfahren wie etwa Federated Averaging (FedAvg) (siehe Kapitel 4) kommen zum Einsatz.

Im vertikalen Szenario liegen zusammengehörige Datensätze über mehrere Silos verteilt vor (bspw. Patientendaten in einem Krankenhaus und Abrechnungsdaten zu den Patienten in einer Versicherung). In diesem Szenario müssen sowohl die Zusammenführung von Daten für das gleiche Individuum (etwa anhand einer in allen Datensätzen vorliegenden ID) als auch das anschließende verteilte Training bedacht werden. Ein Beispiel für Lineare Regression in einem vertikalen Szenario stellen Yang u. a. vor [Yan+19c].

Federated Transfer Learning kann zum Einsatz kommen, wenn heterogene Datensätze (etwa in ihrer Verteilung, im Labeling, in der Anzahl von in allen Datensätzen vertretenen Individuen) mit wenigen Überlappungen in verschiedenen Silos vorliegen. Ein auf dem überlappenden Datensatz trainiertes Modell kann innerhalb der einzelnen Silos als Grundlage für bessere Modelle verwendet werden (Wissenstransfer). Eine Technik für *Federated Transfer Learning* stellen Liu u. a. vor [Liu+20].

3.3 Fully Decentralized Learning

Fully Decentralized Learning (auch als *Peer-to-Peer Distributed Learning* bezeichnet) beschreibt einen verteilten Lernprozess ohne zentralen koordinierenden Server, bei dem die Clients untereinander (im Normalfall nur mit einigen benachbarten Clients) Nachrichten austauschen. In dieser Art des Lernens gibt es kein global gültiges Modell, stattdessen nähern sich die lokalen Modelle der Clients schrittweise einer globalen Lösung an. Zwei Beispiele für Ansätze aus diesem Bereich sind *Gossip Learning* und *Swarm Learning*.

Gossip learning [OHJ13] beschreibt ein verteiltes Lernverfahren ohne zentralen Server. Bei diesem Verfahren kommunizieren die Clients direkt untereinander und tauschen auf diese Weise Modellupdates aus. In [HDJ19] stellen Hegedűs, Danner und Jelasity ein lineares Lernverfahren unter Nutzung von SGD vor. Die Evaluation zeigt vergleichbare

Performance im Vergleich zu einem zentral koordinierten Federated-Learning-Ansatz. Giaretta und Girdzijauskas [GG19] beschreiben Limitierungen von Gossip Learning in Szenarien, in denen Annahmen der Originalveröffentlichung bezogen auf Kommunikationsperformance und Datenverteilungen nicht erfüllt sind.

Eine aktuelle Variante im Bereich des *Fully Decentralized Learning* ist das sogenannte *Swarm Learning* [War+21]. Zusätzlich zu dem verteilten Lernen ohne zentrale Instanz wird hier auf einen Blockchain-basierten Registrierungsprozess für neue Knoten zurückgegriffen, um das Vertrauen in die teilnehmenden Parteien zu vergrößern. Die Autoren evaluieren ihr Framework anhand von verschiedenen Datensätzen aus dem medizinischen Bereich. Der Quellcode für das Framework ist öffentlich verfügbar[1].

3.4 (Cyclic) Institutional Incremental Learning

Eine weitere mit FL verwandte Möglichkeit des verteilten Lernens ist das von Chang u. a. vorgestellte Lernen mittels *(cyclical) weight transfer* [Cha+18]. Sheller u. a. verwenden hierfür den Begriff *(Cyclic) Institutional Incremental Learning* [She+20]. Im Unterschied zu FL, bei dem alle Parteien parallel ein Modell trainieren, wird bei dem hier vorgestellten Verfahren seriell vorgegangen: Eine Partei trainiert das Modell solange, bis das Optimierungsziel erreicht ist, und gibt das Modell dann an die nächste Partei weiter, die das erhaltene Modell mithilfe ihrer lokalen Daten weitertrainiert. Bei der einfachen Variante trainiert jede Partei das Modell einmalig, bei der zyklischen Variante wird der gesamte Vorgang mehrfach durchlaufen und die letzte Partei sendet das Modell wieder an die erste Partei. Sheller u. a. vergleichen das Institutional Incremental Learning mit FL und zeigen eine Überlegenheit von FL in Bezug auf die Konvergenzzeit und Stabilität des Trainingsprozesses [She+20]. Als mögliche Ursache wird hier eine schwächere Art des *Catastrophic Forgetting*[2] vermutet, wodurch das Modell Daten stärker gewichtet, auf denen es später trainiert wurde.

3.5 Split Learning

Split Learning [GR18] beschreibt ein verteiltes Lernverfahren, das die Trainingsphase eines neuronalen Netzes an einer bestimmten Schicht teilt (die sogenannte *Cut Layer*). Ein datenhaltender Client berechnet für die lokal vorliegenden Daten den ersten Teil des Netzes bis zum Cut Layer und teilt das Ergebnis einer zentralen Partei mit. Diese Partei berechnet dann den Rest des Netzes bis zur Ausgabeschicht und beginnt die Backpropagation im ihr vorliegenden Teil des Netzes bis zum Cut Layer. Das Resultat wird nun zurück an den Client übermittelt, der die Backpropagation in den verbleibenden Schichten beendet. Auf diese Weise werden lediglich Gewichte des Netzes in einer bestimmten Schicht übermittelt, jedoch keinerlei lokale Daten. Auch diese Gewichte lassen jedoch Rückschlüsse auf die Daten zu – ein Problem, für dessen Lösung Vepakomma u. a. [Vep+20] einen Ansatz vorstellen.

1. https://github.com/HewlettPackard/swarm-learning (abgerufen am 29.09.2022)
2. *Catastrophic Forgetting* beschreibt die Tendenz eines neuronalen Netzes beim sequentiellen Lernen in der Vergangenheit genutzte Trainingsdaten im Vergleich zu neuen Trainingsdaten stark unterzugewichten (ehemals gelernte Zusammenhänge zu „vergessen“ [MC89].

3.6 Verteiltes Lernen aus Performancegründen

ML-Verfahren, die den Lernprozess aus Performancegründen auf mehrere Knoten verteilen, können auch als *Datacenter Distributed Learning* [Kai+19] oder *Scalability-Motivated Distributed ML* [Yan+19b] bezeichnet werden. Im Gegensatz zum FL wird hierbei angenommen, dass sämtliche Knoten potentiellen Zugriff auf alle Daten besitzen bzw. die Daten beliebig auf die Knoten verteilt werden. Diese Verfahren werden eingesetzt, um Beschränkungen in dem zur Verfügung stehenden Speicher zu umgehen oder die für das Training notwendige Rechenzeit zu reduzieren [Yan+19b]. Hierbei kann sowohl ein dem FL ähnliches Vorgehen genutzt werden, bei dem Trainingsdaten auf mehrere Knoten verteilt werden, die ein vollständiges Modell trainieren und Modellupdates kommunizieren (*Data Parallelism*), als auch ein Vorgehen, bei dem das Modell selbst auf verschiedene Knoten verteilt wird (*Modell Parallelism*) [Yan+19b]. Im Folgenden werden einige Ergebnisse in diesem Bereich kurz dargestellt.

Dean u. a. [Dea+12] stellen mit *DistBelief* ein Framework vor, das es erlaubt, ein Deep Neural Network (DNN) und dessen Training auf mehrere Maschinen zu verteilen. Diese Verteilung erlaubt es, Netze in Größen zu erstellen, die auf einzelnen Maschinen nicht erreicht werden können. Li u. a. [Li+14] stellen mit der *Parameter Server Architecture* ein Framework vor, dass auch für Lernprobleme mit großen Datenmengen im Terabyte-Bereich verwendet werden kann. Eine Übersicht über verteilte Verfahren im Bereich von DNNs bieten Ben-Nun und Hoefler [BH18]. Gängige Bibliotheken, die verteiltes Lernen zu Performancezwecken anbieten, sind *Apache SystemDS*[3], *PyTorch Distributed*[4] und *TensorFlow Distributed*[5].

3.7 Auswirkungen auf die Studie

Anwendungsfälle, die im Bereich der amtlichen Statistik auftreten, sind unserer Einschätzung nach durch folgende Aspekte charakterisierbar:

- Wenige Parteien besitzen größere Datensätze, die nicht zusammengeführt werden dürfen oder sollen. Vorstellbare Parteien sind hier etwa *Other National Authorities (ONAs)* im Sinne des Europäischen Statistischen Systems[6], dazu zählen datenhaltende Behörden (wie die Bundenetzagentur, das Umweltbundesamt, ...) oder auch Forschungseinrichtungen wie das Robert-Koch-Institut. Weitere datenhaltende Parteien könnten Unternehmen oder Konzerne sein.
- Die vorhandene technische Infrastruktur bietet gute Grundvoraussetzungen, um zukünftig mit hoher Verfügbarkeit und Geschwindigkeit mit Partnern zu kommunizieren zu können. In der Praxis können unterschiedliche technische Voraussetzungen und Möglichkeiten bei den Parteien für mehr Komplexität sorgen.

3. https://systemds.apache.org/ (abgerufen am 19.09.2022)
4. https://pytorch.org/tutorials/intermediate/dist_tuto.html (abgerufen am 19.09.2022)
5. https://www.tensorflow.org/guide/distributed_training (abgerufen am 19.09.2022)
6. Für einen Überblick über das Europäische Statistische System siehe https://ec.europa.eu/eurostat/de/web/european-statistical-system, aufgerufen am 09.02.2023

- Weiterhin ist mit dem Statistischen Bundesamt eine zentrale Partei vorhanden, die die Koordination der Parteien in einem verteilten Lernszenario übernehmen kann. Diese Anwendungsfälle sind daher am ehesten durch das *Cross-Silo FL* (in Abgrenzung zum *Cross-Device*-Szenario [Bon+19]) zu beschreiben.
- Zusätzlich sind die durch das Statistische Bundesamt vorgelegten Beispielanwendungsfälle fast ausschließlich in dem horizontalen Szenario (siehe Abschnitt 3.2 zu verorten. Verbleibende Abschnitte der Recherche und auch die in Anhang B beschriebene Simulation von Anwendungsfällen werden daher die *Cross-Silo*-Anwendungsfälle mit einem horizontalen Szenario fokussieren.

4 Verschiedene Methoden für Federated Learning

FL wird vor allem als Erweiterung zu etablierten zentralisierten Ansätzen genutzt, beispielsweise für neuronale Netze, die per SGD trainiert werden [McM+17]. In den folgenden Abschnitten werden die jeweiligen Spezifika von verschiedenen Ansätzen für FL dargestellt.

4.1 Baumbasierte Methoden

In baumbasierten Methoden werden Entscheidungen getroffen, indem rekursiv binäre Fragen gestellt werden und so eine Baumstruktur aufgebaut wird. Die Blätter repräsentieren dann die Ergebnisse des Modells, welche zur *Klassifizierung*, *Regression* oder zum Bewerten (*Ranking*) genutzt werden können. Baumbasierte Methoden haben gegenüber komplexeren Methoden wie etwa DNN den inhärenten Vorteil, dass ihre Entscheidungen einfacher nachzuvollziehen sind und dennoch oft gute Ergebnisse erzielen. Je nach Einsatzgebiet kann dies eine wichtige Eigenschaft sein. Außerdem können baumbasierte Modelle in der Regel besser mit unterschiedlichen Featurearten (*numerische* oder *kategorische*) und allgemein weniger vorverarbeiteten Daten umgehen. Dies kann vor allem in verteilten, nicht einheitlichen Datenbanken relevant sein, wie es in FL-Szenarien der Fall ist [OBZ22].

Da das Trainieren von Baum-Klassifierungsalgorithmen sehr datennah geschieht, ist eine Übertragung auf FL-Szenarien mitunter schwierig [LB22]. Die Festlegung einzelner Knoten in einem Entscheidungsbaum geschieht in der Regel basierend auf Metriken wie *Information Gain*, die auf eine bestmögliche Aufteilung der Daten pro Entscheidungsknoten („*split*“) abzielen. Beim zentralisierten Lernen ist dies trivial, da alle Trainingsdaten lokal vorliegen.

In einem FL-Szenario müssen die benötigten Informationen für die Berechnung des jeweiligen *Information Gain* für mögliche splits allerdings erst zusammengetragen werden. Dafür kann die zentrale Instanz bspw. bei allen Teilnehmern die Anzahl von Daten mit gewissen Merkmalen abfragen (etwa „Wie viele Daten von Frauen unter 40 Jahren liegen vor?“). Mit den erhaltenen Antworten kann die zentrale Instanz alle *Information Gains* berechnen und im Anschluss jenen *split*, der den höchsten *Information Gain* mit sich bringt, auswählen. Für die nächste Trainingsrunde muss dann lediglich kommuniziert werden, welcher *split* gewählt wurde, damit die Teilnehmer ihre lokalen Entscheidungsbäume aktualisieren können [LB22].

Für moderne Klassifizierungsalgorithmen werden häufig mehrere Entscheidungsbäume kombiniert. Während des Trainings werden dann entweder mehrere, zusammen arbeitende, Bäume erstellt (*Bagging*), oder ein einzelner Baum wird aus mehreren weniger genauen Bäumen zusammengestellt bzw. durch diese verbessert (*Boosting*) [OBZ22].

Random Forests sind eine Art von baumbasierten Modellen, welche zu der Kategorie Bagging gehören. Bei klassischen *Random Forests* werden nur Teilmengen der Daten und Features pro erstelltem Baum ausgewählt, was leicht auf FL übertragen werden kann. Dadurch sind sie zusätzlich zu den genannten Vorteilen auch robust gegen Overfitting. Giacomelli u. a. haben einen *Random Forest* entwickelt, in dem die lokal erstellten Bäume nach dem Training dezentral bei den lernenden FL-Parteien verbleiben. Eine Vorhersage muss dann von allen Teilnehmern separat beantwortet werden. Die Informationsübertragung wird dabei zusätzlich durch Homomorphic Encryption (HE) geschützt [Gia+18]. Ein ähnlicher Ansatz, der ebenfalls HE für die dezentrale Beantwortung von Anfragen nutzt, allerdings die Teilnehmer auch während des Trainings miteinander vernetzt, stammt von Vaidya u. a. [Vai+13].

Eine weit verbreiteter *Boosting*-Algorithmus, welcher auch in FL Anwendung findet, ist *XGBoost*, da dieser effizient ist und gute Ergebnisse liefert [CG16]. Cheng u. a. stellen einen „*lossless*“ Ansatz (sprich: er erreicht dieselbe Performance wie klassische XGBoost-Algorithmen) für vertikal verteilte Daten vor [Che+19]. Cheng u. a. stellen ebenfalls einen verteilten XGBoost-Ansatz vor, der versendete Daten verschlüsselt [Che+19]. Yang u. a. nutzen eine Kombination aus verschiedenen Techniken für ihren verteiten XGBoost-Algorithmus, unter anderem k-Anonmyität [Yan+19a].

4.2 Methoden für Deep Learning

Federated-Deep-Learning-Modelle sind weit verbreitet. Die originale Veröffentlichung von McMahan u. a. stellt zunächst den Algorithmus Federated Stochastic Gradient Descent (FedSGD) vor [McM+17]: Hier schickt eine zentrale Instanz ein initialisiertes Modell an eine zufällige Menge von Teilnehmern. Diese trainieren dann das empfangene Model mit ihren jeweils lokalen Daten weiter – im FedSGD pro Runde eine Iteration (oder *epoch*), im erweiterten Algorithmus FedAvg sind mehrere lokale Iterationen pro Trainingsrunde erlaubt. Das so weiterentwickelte Modell wird zurück an die zentrale Instanz gesendet, wo wiederum die nächste Iteration des zentralen Modells aus dem Durchschnitt aller lokalen Modelle berechnet wird. Dieser Vorgang wird wiederholt, bis das Modell konvergiert (siehe auch Abbildung 2.1). Durch die Erweiterung FedAvg wird der Kommunikationsaufwand gesenkt, da weniger Zwischenergebnisse der lokalen Berechnungen ausgetauscht werden müssen. Zusammengefasst ist der wesentliche Unterschied zwischen FedSGD und FedAvg, dass die Teilnehmer bei FedAvg pro Runde mehrere Iterationen des lokalen Trainings vornehmen (können), bei FedSGD werden lokale Modelle pro Runde stets nur einmal lokal iteriert.

Um die Kommunikation weiter zu reduzieren, haben Acar u. a. FL um eine dynamische Regulierung namens Federated Dynamic Regularizer (FedDyn) erweitert [Aca+21]. FedDyn baut grundsätzlich auf FedAvg auf. Zusätzlich kommen Regulierungsterme zum Einsatz, die während der lokalen Berechnungen auf dynamische Weise das Training unterstützen und die Konvergenz zu einem globalen Optimum beschleunigen.

Die Teildatensätze der Teilnehmer sind häufig nicht unabhängig und identisch verteilt (vgl. Kapitel 6). Konkrete Ansätze für Personalisierung wie das Subnetwork Federated Averaging (Sub-FedAvg) [VML21] (siehe auch Kapitel 7) können das Training auf solchen Daten unterstützen. Sub-FedAvg ermittelt für jeden Teilnehmer ein Subnetzwerk

innerhalb des globalen Netwerks, für das ein jeweils eigenes „globales“ Modell trainiert wird [VML21].

4.3 Einfache Methoden

Klassische Methoden des ML wie lineare Regressions- oder Klassifizierungsalgorithmen können auf ähnliche Weise per FL trainiert werden. Alle Methoden, deren Ergebnis (Regressions-)Koeffizienten sind, können durch gewichtete Mittelwerte gleichartiger Koeffizienten zu einem Gesamtmodell aggregiert werden.

4.4 Support Vector Machines

Für das föderierte Training einer Support Vector Machine (SVM) scheint ebenso das einfache Mitteln dezentral gelernter Gewichte möglich zu sein [Nai+22; LB22], wobei beide Quellen keine technischen Details nennen.

4.5 Auswirkungen auf die Studie

Die Tatsache, dass für viele ML-Algorithmen ein einfaches Mitteln der gelernten Parameter (etwa bei Regressionskoeffizienten) genügt, um dezentrale Modellupdates zu einem globalen Modellupdate zu kombinieren, spricht einerseits dafür, FL auch mit einfachen Mitteln, etwa der manuellen Kombination von Modellupdates der Teilnehmer umsetzen zu können. Andererseits gibt es mehrere Gründe, die dagegen sprechen:

1. Auch triviale Algorithmen können in der Umsetzung fehleranfällig sein. Bei viel verwendeten, öffentlichen Bibliotheken sind fehlerhafte Implemenentierungen hingegen fast ausgeschlossen.
2. Viele Verfahren müssen mehrere Iterationen durchlaufen, um zu einem globalen Optimum zu konvergieren: Beispielsweise werden bei dezentralem SGD jeweils jene Gradienten berechnet, die auf den Teilnehmerdatensätzen zu einem lokalen Optimum auf den jeweils lokalen Daten führen. Werden diese Gradienten gemittelt, müssen sie keineswegs die Richtung zu einem globalen Optimum zeigen. Mehrere Iterationen, in denen iterativ kleine Schritte gegangen werden (wie in FedSGD und FedAvg) können hingegen zum Ziel (also zu einem globalen Optimum) führen. Wird über viele Runden hinweg trainiert, wäre eine manuelle Koordination wiederum aufwändig; hier kann der Einsatz etablierter Frameworks Abhilfe verschaffen.
3. Wird eine Standardimplementierung bzw. werden etablierte Frameworks verwendet, können bereits definierte Datenschnittstellen einen weiteren Vorteil gegenüber der manuellen Kombination von Updates darstellen. Sind im FL-Prozess mehrere Parteien involviert, kann dies inkompatiblen Datenformaten und der Notwendigkeit, Daten zu konvertieren, von Anfang an vorbeugen.

Zusammengefasst: FedAvg und FedAvg können unter Umständen auch durch einfache Selbstimplementierungen ersetzt werden. Je nach konkretem Anwendungsfall muss individuell überprüft werden, ob (bspw. bei einfachen Regressionsmodellen) eine einzige Trainingsrunde, in deren Anschluss die erhaltenen Koeffizienten gemittelt werden, genügt, um die jeweiligen Qualitätsanforderungen zu erfüllen. Um die Fehleranfälligkeit zu senken und insbesondere Kooperationspartnern in standardisierte Trainingsworkflows einzubinden, liegt es dennoch nahe, auf bestehende, möglichst etablierte, Frameworks zurückzugreifen.

5 Bestehende Frameworks

Es gibt mittlerweile ein weites Angebot an Frameworks für FL. Dazu zählen Open-Source-Frameworks wie kommerzielle Lösungen, die jeweils auf unterschiedliche Nutzung ausgerichtet sind. Dabei kann grob in Frameworks unterteilt werden, die den produktiven Einsatz fokussieren, und solche, die vorwiegend zur Forschung und Simulation von Federated-Learning-Szenarien eingesetzt werden. Produktorientierte Frameworks sind zwar oft weniger flexibel einsetzbar, adressieren dabei jedoch Probleme, die in einer Simulation vernachlässigt werden können. Dazu zählen insbesondere Kommunikationsprotokolle, Verschlüsselung und Authentifizierung.

Im Folgenden werden einige dieser Frameworks kurz vorgestellt:

1. *TensorFlow Federated* ist ein von Google entwickeltes Open-Source-Framework für Python. Es ist sehr umfangreich und legt einen Fokus auf Forschung und Simulation von Federated-Learning-Szenarien. Aus diesem Grund werden neben Funktionen und Modellen für FL auch weitreichende Analysewerkzeuge bereitgestellt. Da *TensorFlow Federated* von Google auch für tatsächliche Produkte eingesetzt wird, ist es das umfassendste, hier vorgestellte Framework[1] [Kai+19].

2. *FedML* ist ein Open-Source-Framework, das einen Fokus auf Forschung legt. Es unterstützt mobile Geräte, einfache Rechner und Cluster sowie eine breite Palette an FL-Algorithmen. *FedML* ist für Python entwickelt[2] [He+20].

3. *OpenFL* ist ein quelloffenes, pythonbasiertes Framework, das von Intel entwickelt wird. Es ist darauf ausgelegt mit PyTorch, TensorFlow oder einem anderen ML-Framework genutzt zu werden und wurde ursprünglich für medizinische Anwendungen entwickelt[3] [Fol+22].

4. *Flower* ist ein quelloffenes Framework, das insbesondere darauf abzielt, Forschung und Simulationen realistischer zu machen. Dies wird erzielt durch Funktionen und Komponenten, die die diverse Verteilung von Teilnehmern und Daten simulieren. Weiterhin wird auch ein Fokus auf die Skalierbarkeit gelegt – auch diesen Aspekt sehen die Autoren als stark unterschiedlich ausgeprägt zwischen Forschung und realistischem Einsatz an. *Flower* kann dazu auch mit anderen Frameworks kombiniert werden[4] [Beu+20].

5. *FATE* ist ein Framework, welches ebenfalls quelloffen ist und von der chinesischen Webank entwickelt wird. *FATE* ist produktorientiert und dient vor allem als sicheres Framework für das Federated-AI-(FedAI-)Ökosystem. Es unterstützt die Verteilung von aufwendigen Berechnungen auf Clustern[5] [Kai+19].

1. https://www.tensorflow.org/federated (abgerufen am 11.10.2022)
2. https://fedml.ai/ (abgerufen am 12.10.2022)
3. https://github.com/intel/openfl (abgerufen am 12.10.2022)
4. https://flower.dev/ (abgerufen am 12.10.2022)
5. https://fate.fedai.org/ (abgerufen am 11.10.2022)

6. *NVIDIA Clara* wurde für medizinische Anwendungen entwickelt. Die Einsatzbereiche sind spezifischer und stärker auf *Cross-Silo*-Situationen ausgelegt. Diese Einschränkungen, genau wie die feste Server-Client-Architektur, vereinfachen allerdings die Entwicklung[6] [Kai+19].

7. *IBM Federated Learning* ist ebenfalls für Python entwickelt und nutzbar als alleinstehende Lösung[7]. Als produktorientiertes Framework unterstützt es viele Basisfunktionen und bekannte Algorithmen und legt seinen Fokus auf schnelle Ergebnisse [Lud+20].

6. https://www.nvidia.com/de-de/clara/ (abgerufen am 12.10.2022)
7. https://www.ibm.com/docs/en/cloud-paks/cp-data/4.6.x?topic=models-federated-learning (abgerufen am 12.02.2023)

6 Probleme durch Heterogenität der lokalen Datensätze

In der Literatur werden vielfach unabhängige und identisch verteilte Zufallsvariablen (engl. *independent and identically distributed random variables* (*IID*)) als optimale Ausgangssituation für verteilte Lernverfahren genannt. Dies bedeutet, dass es nur eine mathematische Verteilung der (Trainings-)Daten gibt, die über die am Lernfahren teilnehmenden Parteien hinweg in allen Teildatensätzen identisch ist und lokale Verteilungen nicht voneinander abhängen. In der Praxis ist dies kaum möglich, da die Teildatensätze der Parteien in der Regel Unterschiede aufweisen: Dies kann durch geografische Unterschiede begründet werden, etwa wenn die Teildatensätze in verschiedenen Bundesländern erhoben wurden. Es sind aber auch viele weitere Gründe möglich, bspw. Unterschiede in der Datenerhebung, Erfassung verschiedener sozialer Milieus, etc.

6.1 Mögliche Ursachen

Ursachen für die Abweichung vom IID-Szenario lassen sich in folgende Kategorien einteilen [Kai+19; Pra+21]:

1. *Anders verteilte Features* bei den jeweiligen Parteien (engl. *covariate shift*) über alle Klassen hinweg: Beispielsweise ist dies bei einer föderiert trainierten Handschriftserkennung der Fall, da die Handschrift jeder Person individuelle Abweichungen aufweist, in der Größe der Buchstaben, Dicke der Linien, etc.

2. *Anders verteilte Klassen* in den Trainingsdaten (engl. *prior probability shift*): Durch geografische oder demografische Unterschiede in den jeweiligen Teildatensätzen können bestimmte Klassen häufiger oder seltener auftreten: So gibt es Kängurus fast ausschließlich in Australien und in Zoos, bestimmte Wörter werden nur regional verwendet. Diese Problematik wurde im Design des vielfach verwendeten FedAvg-Algorithmus berücksichtigt [McM+17].

3. *Andere Features für dieselbe Klasse* (engl. *concept drift*) können auftreten, wenn Daten bspw. aufgrund von kulturellen Verschiedenheiten zwar anders ausgeprägt, aber über verschiedene Parteien mit denselben Labels versehen werden. Bei einer Bilderkennungssoftware könnte etwa die Klasse „Auto“ in den USA mit Bildern von tendenziell größeren Autos befüllt werden als in Italien. Während die erste Kategorie (*covariate shift*) Verschiebungen in den Gesamtverteilungen der Trainingsdaten über **alle** Klassen hinweg beschreiben, beschreibt diese Kategorie konkrete Abweichungen in den Features bzw. Beschreibungen **einzelner** Klassen.

4. Wenn Daten mit *denselben Features* nicht immer denselben Klassen zugeordnet werden, handelt es sich um *concept shift*. Dies kann eintreten, wenn die Klassenlabel bspw. manuell von Fachexperten hinzugefügt werden und persönliche Präferenzen und Einschätzungen der Experten variieren.

5. *Unbalanciertheit* (engl. *quantity skew*) der Daten kann häufig ein Problem darstellen, wenn teilnehmende Parteien über verschieden viele Datenpunkte verfügen und damit auch unterschiedlich viel Einfluss auf das Resultat des Lernverfahrens haben.

Eine Extrem- bzw. Mischform aus der 1. und 3. Kategorie stellen Prayitno u. a. als *Feature distribution skew* vor: Dies ist der Fall, wenn einzelne Features bei einer oder bei mehreren teilnehmenden Parteien nicht existieren, weil sie bspw. nicht erfasst wurden [Pra+21].

6.2 Auswirkungen auf die Studie

Wenn dieselben Features von unterschiedlichen Parteien mit jeweils anderen Klassen versehen werden (*concept shift*), kann eine Personalisierung (siehe nächster Abschnitt) essenziell sein, um brauchbare Ergebnisse zu erzielen [Kai+19].

Weitere Ansätze, um das Problem heterogener Datensätze zu lösen, beinhalten etwa das Teilen eines öffentlichen Teildatensatzes, den alle Parteien ihren lokalen Trainingsdaten hinzufügen [Zha+18]. Alternativ kann dieser öffentlicher Teildatensatz (sofern vorhanden) auch dafür genutzt werden, das globale Modell vor dem eigentlichen FL-Trainingsprozesses *vor*zutrainieren, was insbesondere bei heterogenen lokalen Datensätzen für eine bessere Performance sorgen kann [Che+22]. Ein weiterer Vorschlag nutzt einen Regularisierungsterm in den lokalen Trainingsprozessen, um große Abweichungen vom globalen Modell zu vermeiden [Li+20b].

Weitere experimentelle Ansätze wie [Hsi+20; Wan+20b] versuchen das Problem tiefgreifender zu lösen, indem die übliche *Batch-Normalisierung* durch eine andere Formen der Normalisierung ersetzt wird; in [Hsi+20] wird zusätzlich die Kommunikation zwischen den Parteien adaptiv erhöht, indem häufiger Zwischenergebnisse der trainierten Modelle ausgetauscht werden.

7 Personalisierung

Im vorhergehenden Abschnitt wurde dargestellt, welche Probleme durch die Heterogenität lokaler Daten entstehen können. Verteilte Lernprozesse auf heterogen verteilten Datensätzen können zu ungenaueren globalen Modellen führen, die in Einzelfällen auch Falschaussagen treffen können. Zusätzlich ist die Genauigkeit eines einzelnen globalen Modells zwangsläufig geringer, wenn sich die Verteilungen innerhalb der Datensätze zwischen den Clients stark unterscheiden, da lediglich ein Modell für den Durchschnittsclient gelernt wird [Tan+22].

Eine Möglichkeit um diesen Effekten zu begegnen stellt die Personalisierung von Modellen dar. Hierbei wird der Lernprozess individualisiert, indem Client-spezifische Modelle mit besonderem Fokus auf den lokalen Daten der Clients erzeugt werden. Eine Übersicht über aktuelle Ergebnisse in der Personalisierung von FL wurden von Kulkarni, Kulkarni und Pant [KKP20] und Tan u. a. [Tan+22] erarbeitet. Im Folgenden werden einige Ansätze zur Personalisierung kurz vorgestellt. Dabei folgen wir der von Tan u. a. vorgestellten Taxonomie (dargestellt in Abbildung 7.1). Grundsätzlich wird hier zwischen *Global Model Personalization* und *Solution Personalization* unterschieden.

7.1 Global Model Personalization

Global Model Personalization beschreibt Verfahren, die darauf abzielen, das globale Modell auf den heterogen verteilten Daten derart zu trainieren, dass es in einem anschließenden Personalisierungsschritt unter Nutzung lokaler Daten bestmöglich personalisiert werden kann. Das entsprechende Vorgehen ist in Abbildung 7.2 dargestellt. Die Unterklasse *datenbasierter* Verfahren versucht die statistische Heterogenität der Clientdaten zu reduzieren. Hierbei kann entweder auf *Data-Augmentation*-Techniken, die mithilfe von Beispieldaten oder Datenverteilungen homogenere Daten erzeugen, oder auf *Client-Selection*-Techniken, die auf einer geschickten Auswahl von am Training beteiligten Clients beruhen, zurückgegriffen werden. Ein Beispiel für ein solches Verfahren ist das von Wang u. a. vorgestellte Framework *Favor* [Wan+20a]. Eine weitere Möglichkeit basierend auf einer unter allen Clients geteilten, kleinen Untermenge aller Daten präsentieren Zhao u. a. [Zha+18]. Die Unterklasse *modellbasierter* Verfahren versucht, das globale Modell mit einem Fokus auf Adaptivität zu trainieren oder die Anpassung der lokalen Modelle zu verbessern. Hier kann zwischen auf Regulierung, Meta-Lernen oder Transfer-Lernen basierenden Verfahren unterschieden werden. Ein Beispiel für ein Regulierungsverfahren ist das von Li u. a. vorgestellte *FedProx* [Li+20c].

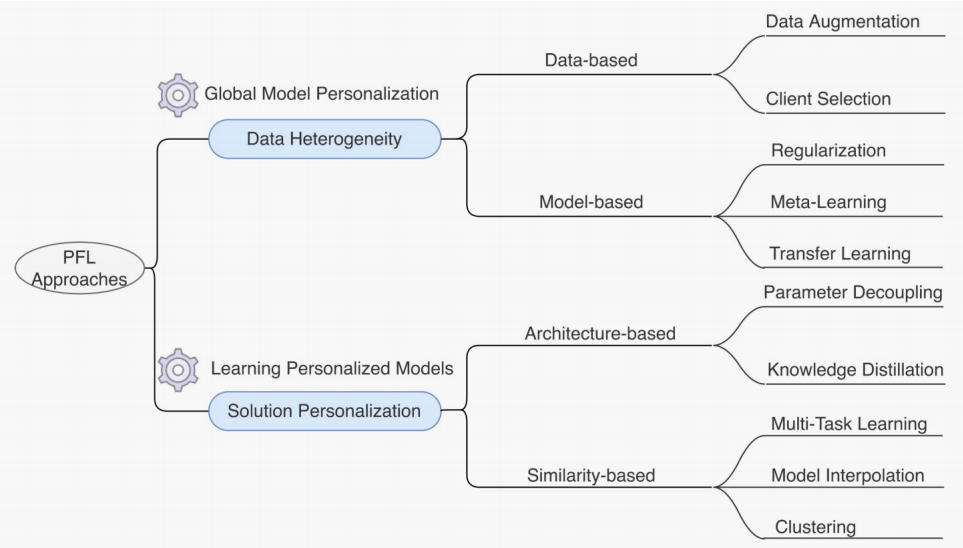

Abbildung 7.1: Taxonomie zu Ansätzen für die Personalisierung von FL-Modellen. Entnommen aus [Tan+22].

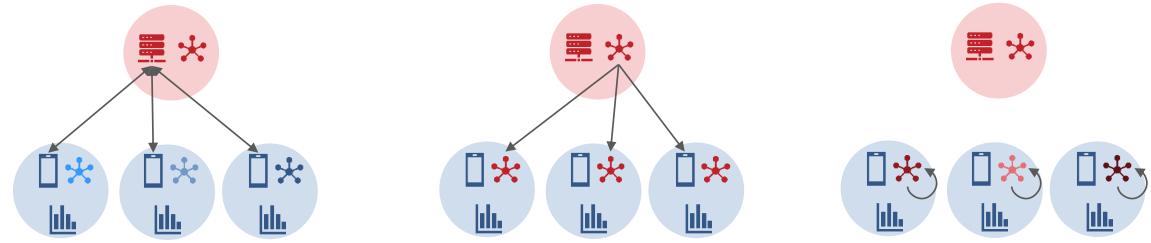

Abbildung 7.2: Schematische Darstellung der einzelnen Schritte für die *Global Model Personalization*.

7.2 Solution Personalization

Im Vergleich zur *Global Model Personalization* zielen Verfahren der *Solution Personalization* darauf ab, mehrere personalisierte Modelle zu trainieren ohne auf ein globales Modell zurückzugreifen. Die Unterklasse *architekturbasierter* Verfahren stellt clientspezifische Modellarchitekturen bereit. Ein Beispiel findet sich in dem Framework *FedPer*, das von Arivazhagan u. a. entworfen wurde [Ari+19]. Hier werden bestimmte Schichten eines neuronalen Netzes für die Personalisierung genutzt. Eine weitere Unterklasse wird durch die *ähnlichkeitsbasierten* Verfahren gebildet, die anhand der Ähnlichkeit von Clientdaten ähnliche Modelle erstellen. Ein generischer Ansatz wird hier etwa von Hanzely und Richtárik vorgestellt, die einen parameterbasierten Ansatz für die Balance zwischen Allgemeingültigkeit und Personalisierung der Modelle wählen [HR20]. Die Verfahren der dritten Unterklasse basieren auf *Clustering* und können in Szenarien genutzt werden, in denen die Clienten (bzw. ihre lokalen Datenverteilungen) natürliche Partitionen bilden. In diesen Szenarien können ähnliche Clients zusammengefasst und für die Gruppen spezifische Modelle trainiert werden. Ein Beispiel für dieses Vorgehen beschreiben Sattler, Müller und Samek [SMS19].

7.3 Auswirkungen auf die Studie

Der Fokus des Statistischen Bundesamtes liegt weniger auf einer Verbesserung des Modells in den einzelnen Datensilos, sondern primär auf der Erzeugung und Weiterverwendung eines globalen Modelles aus nicht zusammenführbaren Daten. Dies spricht tendenziell gegen den Einsatz von FL-Personalisierungstechniken in der amtlichen Statistik. Trotzdem könnte die Personalisierung unter bestimmten Voraussetzungen gewinnbringend eingesetzt werden, etwa wenn starke regionale Unterschiede in den Datenverteilungen der verschiedenen Datensilos vorliegen.

Allgemein sind Personalisierungsverfahren in Szenarien sinnvoll, in denen Clientdaten stark heterogen verteilt sind. Ein gutes Beispiel ist etwa die Nutzung von FL für Textvorschläge und Textkorrektur auf mobilen Geräten (etwa GBoard von Google [Har+18; MR17]). Hier unterscheidet sich das Verhalten zwischen einzelnen Nutzern in der Regel erheblich und insbesondere regionale Unterschiede durch verschiedene Sprachen und Dialekte können einen starken Einfluss besitzen. In *Cross-Silo* Szenarien mit homogen verteilten Daten kann die Personalisierung hingegen als weniger relevant eingestuft werden (für heterogen verteilte Datensätze siehe Kapitel 6).

8 Einfluss von Netzwerk-Topologien und Hyperparametern auf FL-Ergebnisse

Modelle des maschinellen Lernens verfügen über viele sogenannte Hyperparameter. Anders als die *Parameter* eines neuronalen Netzes, die während des Trainingsvorgangs schrittweise angepasst werden und essentiell für die Nützlichkeit eines trainierten Modells sind, sind *Hyperparameter* zunächst unabhängig von der eigentlichen Aufgabe des Modells. Dennoch beeinflussen sie die Qualität und die Geschwindigkeit des Lernprozesses maßgeblich, indem sie die verwendete Modellarchitektur und den Trainingsprozess beschreiben – bei neuronalen Netzen etwa die Anzahl und Anordnung der einzelnen Neuronen, die Lernrate und die Batchgröße der Trainingsdaten.

Die idealen Werte der Hyperparameter unterscheiden sich grundlegend von Anwendungsfall zu Anwendungsfall – je nach Domäne, eingesetztem Algorithmus, Zahl der teilnehmenden Parteien, Datensatzqualität und -größe. Abgesehen von sehr allgemeinen Empfehlungen (Modellkapazität so niedrig wie möglich wählen, um Overfitting zu vermeiden etc.) existieren daher keine allgemeingültigen Ratschläge zur Wahl von FL-Hyperparametern.

Neben der manuellen Suche durch Versuch und Irrtum existieren für das klassische, nicht föderierte, Szenario des maschinellen Lernens viele, mittlerweile etablierte Techniken für die Hyperparametersuche (auch: Hyperparametertuning, -optimierung oder *neural architecture search*, NAS) [YS20]. Diese fallen in der Regel in eine der folgenden Kategorien [Kai+19; YS20]:

1. *Evolutionäre Algorithmen*: Die Idee von evolutionären Algorithmen ist es, neue Generationen durch leichte Veränderungen in der „Population“ (in diesem Fall verschiedene Sätze von Hyperparametern für ein Modell) entstehen zu lassen. Jene Mitglieder einer Population, die am besten abschneiden (hier: die größten Verbesserungen des Modells hervorbringen), werden erneut modifiziert, um eine weitere Generation zu erstellen. Der Prozess kann dann beendet werden, wenn die Leistungen konvergieren. Hierzu existieren fertige Implementierungen wie `sklearn-genetic-opt` in der Bibliothek sklearn[1].

2. *Reinforcement Learning* kann ebenfalls für die Suche nach optimalen Hyperparametern genutzt werden. Hierbei können auch lokale Optima zugunsten eines global(er)en Optimums überwunden werden, da Reinforcement Learning auf langfristige Strategien abzielt – in diesem Fall auf die geeignetsten Hyperparameter für ein Modell [JGS19; Nea18].

3. *Gradient Descent* kann dann für die Hyperparameteroptimierung verwendet werden, wenn die Wahl der Hyperparameter in die Trainingspipeline derartig eingeflochten werden, dass ein Backtracking bis zu den Hyperparametern möglich ist. Hierfür existieren modellspezifische Algorithmen, bspw. [MDA15].

1. https://sklearn-genetic-opt.readthedocs.io/ (abgerufen am 10.10.2022)

Außerhalb dieser Kategorien existieren weitere Ansätze wie *Successive Halving* [JT16], dessen Weiterentwicklung *Hyperband* [Li+16], Bayes'sche Optimierung, *Surrogate Model Based Optimization* (SMBO) [BLP05], *Sequential Parameter Optimization Toolbox* (SPOT) [Bar+21] und *multinomial distribution learning* [ZZJ21]. Eine Kombination aus Bayes'scher Optimierung (BO) und Hyperband (HB) ist BOHB von Falkner, Klein und Hutter [FKH18]. Eine Übersicht über verschiedene Methoden der Hyperparameteroptimierung bietet [Bis+21].

Für FL ist das Problem der Hyperparametersuche noch komplexer. Einerseits ist die übliche Praxis, unzählige Trainingsvorgänge mit jeweils verschiedenen Hyperparametern anzustoßen, aufgrund der Dezentralität oft nicht ohne Weiteres möglich [Zho+21]. Andererseits können im FL-Szenario neben der Genauigkeit des fertig trainierten Modells noch andere Zielgrößen für die Wahl der optimalen Hyperparameter entscheidend sein – je nach Einsatzszenario bspw. verschiedene Kenngrößen des Kommunikations- oder Rechenaufwands [Zha+21][2]. Hinzu kommt das Problem der oftmals verteilten Validierungsdaten, falls die zentrale Instanz über keine Daten verfügt. Dadurch kann der Vergleich von resultierenden Modellen einer Hyperparameter-Optimierungsrunde ggf. erschwert werden [Kho+21].

Obwohl in den letzten Jahren einige Ansätze zur FL-Hyperparameteroptimierung publiziert wurden [Mos19; Kho+21; DLJ20; Zho+21; KH18], haben sich noch keine praktikablen Werkzeuge etabliert [Kai+19].

2. Diese Probleme betreffen zwar eher das *Cross-Device*-Szenario (vgl. Abschnitt 3.1), die Literatur der FL-Hyperparametersuche scheint dies aber eher zu berücksichtigen als das *Cross-Silo*-Szenario.

9 Privatsphäre-Angriffe im Kontext von Federated Learning

Um einen generellen Überblick über die Angriffsmöglichkeiten in FL-Szenarien zu bekommen, müssen zunächst mögliche Angreifermodelle in Betracht gezogen werden. In der besonderen Situation der amtlichen Statistik sind manche Angreifermodelle weniger wahrscheinlich als andere, worauf im folgenden Abschnitt 9.3 eingegangen wird.

9.1 Angreifermodelle

Baracaldo und Xu [BX22] unterscheiden zwischen den drei Rollen Aggregator, Teilnehmer und Außenseiter, die sich jeweils aktiv und passiv bösartig verhalten können: Während passive Angreifer (oft auch als „honest-but-curious“ bezeichnet) stets dem vorgeschriebenen FL-Protokoll folgen und lediglich aus den ihnen zur Verfügung stehenden Daten versuchen, möglichst viele Informationen zu gewinnen, können aktive Angreifer (auch als „malicious“ bezeichnet) auch bösartig in die FL-Prozesse eingreifen, indem sie beispielsweise manipulierende Updates beisteuern. Somit existieren grundsätzlich sieben verschiedene Angriffsmöglichkeiten [BX22]:

1. Ein *passiver Aggregator* kann aus den individuellen Modellupdates der Teilnehmer zusätzliche Informationen zu ihren Trainingsdaten gewinnen.
2. Ein *aktiver Aggregator* kann die Modellupdates für die Teilnehmer manipulieren, um an weitere Informationen der Teilnehmer zu gelangen.
3. Ein *passiver Teilnehmer* kann aufgrund seines Wissens über die zu lernende Aufgabe und den erhaltenen Modellupdates Informationen gewinnen.
4. Ein *aktiver Teilnehmer* kann zusätzlich sein eigenes Update derartig manipulieren, dass er aus den folgenden aggregierten Modellupdates des Servers Informationen gewinnen kann.
5. *Gemeinsam agierende Teilnehmer* können sich untereinander austauschen und ihren Angriff (passiv oder aktiv) derartig orchestrieren, dass sie weitere Informationen gewinnen können.
6. Ein *Aggregator* kann *mit einem oder mehreren bösartigen Teilnehmern* gemeinsam einzelne Zielteilnehmer noch besser kompromittieren.
7. *Außenseiter* haben keinen Einfluss auf den Trainingsvorgang, können aber mit dem fertigen Modell interagieren und daraus Informationen extrahieren.

9.2 Privatsphäreangriffe

Mögliche Angriffe lassen sich in folgende Kategorien einordnen [BX22]:

- Geschickt konstruierte Modellupdates, die von bösartigen Teilnehmern selbst im Trainingsprozess beigesteuert werden, können die *Extraktion einzelner Trainingsdaten* aus Modellupdates ermöglichen (bspw. in [ZLH19]).
- *Membership Inference* Angriffe zielen darauf ab, herauszufinden, ob einzelne Datenpunkte Teil des Trainingsdatensatzes waren [NSH19].
- Bei einem *Model-Inversion*-Angriff ist das Ziel, repräsentative Datenpunkte für eine Klasse zu erstellen. Dieser Angriff ist vor allem dann relevant, wenn die Daten innerhalb einer Klasse sehr ähnlich sind (zum Beispiel Gesichtserkennung) [HAP17].
- *Property-Inference*-Angriffe zielen nicht auf die Trainingsdaten selbst ab, sondern auf statistische Verteilungen im Datensatz, bspw. das Verhältnis von Männern zu Frauen in einem Datensatz [Mel+19].

Für ausführliche Beschreibungen der einzelnen Angriffe sei auf ein kürzlich erschienenes Survey von Baracaldo und Xu verwiesen [BX22].

9.3 Auswirkungen auf die Studie

Die Rolle des Aggregators wird im Kontext dieser Studie stets vom statistischen Bundesamt eingenommen, das den Ruf eines zuverlässigen Statistikproduzenten hat und gesetzlich dazu verpflichtet ist, Daten mit hohen Datenschutz- und IT-Sicherheitsstandards zu bearbeiten. Von Privatsphäreangriffen, bei denen der Aggregator aus ausgetauschten Daten mehr Informationen gewinnt als vereinbart (als passiver Angreifer) oder gar den Lernprozess derartig beeinflusst, dass darüber hinaus weitere Informationen gewonnen werden können (als aktiver Angreifer), ist somit nicht auszugehen. Dennoch sind Informationen über *mögliche* Angriffe ein wichtiger Transparenzaspekt, etwa in der Kommunikation mit möglichen Teilnehmern eines FL-Trainingsprozesses. Um das Vertrauen zusätzlich zu stärken, können in kritischen Anwendungen auch zusätzliche Sicherheitsmaßnahmen (bspw. durch ein Trusted Execution Environment (TEE) oder Secure Multi-Party Computation (SMPC), siehe folgender Kapitel 10) getroffen werden.

Dies gilt auch für den Fall, dass FL-Teilnehmer selbst versuchen, Informationen zu den Trainingsdaten anderer Parteien zu erlangen: Da Angriffe per se nicht ausgeschlossen werden können, sollte darüber mindestens transparent kommuniziert werden. Wenn technisch möglich, können darüber hinaus Gegenmaßnahmen getroffen werden, um Angriffe aktiv zu verhindern.

Aus Sicht des statistischen Bundesamtes stellen *Property-Inference*-Angriffe die relevanteste Angriffsfläche dar – einerseits, weil die Natur der Anwendungsfälle oftmals weniger in personenbezogenen Daten zu finden ist, die die Grundlage der meisten anderen Privatsphäreangriffe darstellt. Andererseits handelt es sich bei potenziellen FL-Teilnehmern oftmals um Unternehmen, die ein Interesse daran haben, auch die Verteilung ihrer Daten

geheimzuhalten (bspw. um nicht die Demographie ihres Kundenstamms an Marktkonkurrenten preiszugeben). Dem folgenden Kapitel 10 sei vorweggenommen, dass derzeit noch keine pauschale Vorsorge-/Verteidigungsmaßnahme für den Property-Inference-Angriff entdeckt wurde [RG20].

Bei passiven (honest-but-curious) Angreifern ist eine Beobachtbarkeit des Angriffs – sprich: die Erkennung, dass ein Angriff stattfindet bzw. stattgefunden hat – selbst ausgeschlossen: Von außen ist die teilweise anderweitige Nutzung der ausgetauschten Daten nicht kontrollier- bzw. erkennbar. Auch bei aktiven Angreifern hängt die Feststellbarkeit des Angriffs stark vom jeweiligen Szenario und der konkreten Angriffsstrategie ab. Aufgrund der Vielzahl möglicher Angriffe und deren Ausgestaltungen ist das Erkennen manipulierter Modellupdates mindestens schwierig, in Anbetracht des Ziels der Unauffälligkeit der Angriffe in vielen Fällen sicherlich auch unmöglich.

Darüber hinaus dürfte eine rechtliche Belangbarkeit unproblematisch sein, wenn Teilnehmer klar von den im Vertrag der Auftragsverarbeitung definierten Aufgaben abweichen, indem sie die übertragenen Modellparameter zu anderen Zwecken (etwa Privatsphäreangriffen) verarbeiten.

10 Privacy-Preserving Machine Learning

Grundsätzlich bedeutet ein Schutz der Privatsphäre im Kontext von FL-Trainingsprozessen, sowohl die intermediären Modellupdates als auch das fertig trainierte Modell gegen die oben genannten Angriffe zu schützen [BX22]. Mit diesem Schutz beschäftigt sich das Forschungsfeld des Privacy-Preserving Machine Learning (PPML). Die hierbei fokussierten Ziele bestehen etwa darin, konkrete Informationen über die Trainingsdaten schwerer zugänglich zu machen oder die Modellupdates der Teilnehmer zu kombinieren, ohne dass der Aggregator Zugriff auf die individuellen Updates erhält (*Secure Aggregation* [Bon+17]).

10.1 Techniken zum Schutz der Privatsphäre

Zu diesen Zwecken können verschiedene Techniken zum Einsatz kommen, u. a.:

- Secure Multi-Party Computation (SMPC) [Lin20] bezeichnet Protokolle, die es mehreren Parteien ermöglichen, gemeinsam eine Funktion auf privaten Eingaben zu berechnen, ohne dass Parteien im Zuge des Protokolls die privaten Eingaben anderer Parteien lernen. Ein Beispiel für den Einsatz von SMPC im Kontext von FL ist in [Tru+19] zu finden.
- Homomorphic Encryption (HE) [Gen09] beschreibt Verschlüsselungsverfahren, die Berechnungen auf verschlüsselten Daten erlauben, ohne eine vorhergehende Entschlüsselung vornehmen zu müssen. In [Zha+20] wird HE im *Cross-Silo*-FL-Szenario (siehe Kapitel 3) eingesetzt.
- Unter Differential Privacy (DP) [DR+14] wird eine Klasse von Methoden zusammengefasst, die es durch Randomisierung erlauben, den Einfluss einzelner Datenpunkte in Berechnungen zu reduzieren. Ein Angreifer erhält so kaum Informationen über einzelne Individuen in den Daten. Beispielhaft wird DP in [GKN17] für FL genutzt.
- Es kann eine syntaktische Anonymisierung von Eingabedaten, bspw. durch k-Anonymisierung [Swe02], erfolgen. Auf diese Weise erfolgt eine Reduzierung des Personenbezugs bereits vor der Trainingsphase. Ein Beispiel für diesen Ansatz ist in [Cho+20] zu finden.
- In bestimmten Szenarien können *Trusted Execution Environments* [Law+20] zusätzlichen Schutz bieten, indem bspw. die Hardware des Aggregators vor Zugriffen von außen geschützt wird oder per *Remote Code Attestation* sichergestellt werden kann, dass auf einem Server nur bestimmter Code ausgeführt wird.

10.2 Auswirkungen auf die Studie

Der häufigste Einsatzzweck von PPML liegt in dem Schutz sensibler Daten von Individuen, wie es etwa bei der Verwendung von Gesundheitsdaten notwendig sein kann. Insbesondere auch in *Cross-Device*-FL-Szenarien, bei denen jede Partei (jedes Gerät) Daten genau eines Individuums bereitstellt, sind entsprechende Methoden von hoher Relevanz. Im Rahmen der Arbeit des Statistischen Bundesamtes in Kombination mit FL sind diese Techniken aus unserer Sicht weniger entscheidend, da wie in Kapitel 3 dargestellt, insbesondere das *Cross-Silo* FL im Fokus steht. In diesem Szenario stellen Modellupdates bereits eine Aggregation über allen Datenpunkten (die potentiell Individuen zuzuordnen sind) einer Partei dar. Weiterhin sind viele der in diesem Abschnitt dargestellten Techniken noch eher dem Stand der Forschung als der Technik zuzurechnen und stellen für praktische Anwendungen zu hohe Anforderungen in Bezug auf notwendige Expertise der Entwickler und insbesondere auch in Bezug auf Performanceaspekte. In der Zukunft könnten sich jedoch Szenarien ergeben, in denen entsprechende Methoden in technischen Lösungen gewinnbringend für Anwendungen des Statistischen Bundesamtes einsetzbar sein werden – sei es im Kontext von FL oder unabhängig davon.

11 Bewertung der Ergebnisse der Literaturrecherche bezogen auf die amtl. Statistik

Zusammenfassend kann FL aus unserer Sicht gewinnbringend für die amtliche Statistik eingesetzt werden. Hierbei geht es um den Einsatz in Szenarien, in denen Daten verteilt auf mehrere Parteien vorliegen und ein direkter Zugriff durch das Statistische Bundesamt nicht möglich ist. FL kann in diesen Szenarien als „Enabler-Technologie“ dienen. Es geht dabei explizit nicht darum, Performancegewinne oder bessere Modelle im Vergleich zu zentralisierten Verfahren zu erhalten, sondern in diesen Szenarien überhaupt statistische Zusammenhänge modellieren zu können.

Als relevant hat sich das *Cross-Silo*-FL-Szenario mit horizontaler Datenverteilung herausgestellt (siehe Kapitel 3). Hierbei sei noch einmal darauf hingewiesen, dass der Begriff FL inzwischen teilweise synonym zu verteilten Lernverfahren benutzt wird. In diesem Sinne ist auch unsere Einschätzung zu verstehen: Auch verwandte Konzepte, die inzwischen unter dem Schirm von FL aufkommen, können hier sinnvoll einsatzbar sein. Erkenntnisse aus anderen Arten von verteiltem Lernen können evtl. in aufbauenden Forschungsarbeiten auf dieses Szenario transferiert werden.

Auch wenn das Problem der Heterogenität lokaler Datensätze (siehe Kapitel 6) im *Cross-Silo*-Szenario weniger kritisch als etwa im *Cross-Device*-Szenario zu bewerten ist, so können z.B. unterschiedliche Arten der Datenerfassung bei den verschiedenen Parteien trotzdem zu Problemen führen. Da der Einsatz von FL für die amtliche Statistik jedoch darauf abzielt, ein einzelnes globales Modell zu trainieren, sind Personalisierungsmethoden, wie sie in Kapitel 7 vorgestellt werden, keine geeigneten Lösungsansätze. Möglicherweise kann hier auf andere Verfahren, etwa geeignete Regularisierungs- oder Normalisierungsansätze, zurückgegriffen werden.

Die Hyperparameteroptimierung im Kontext von FL (siehe Kapitel 8) scheint ein noch offenes Problem zu sein. Abhängig vom Einsatzkontext könnten hier möglicherweise erst einmal auf Teildatensätzen lokal geeignete Hyperparameter gesucht werden und anschließend das verteilte Training mit diesen Parameter durchgeführt werden.

PPML, wie in Kapitel 10 dargestellt, fokussiert insbesondere den Schutz von Individuen in den Daten. Dieses Ziel ist unserer Einschätzung nach weniger relevant für die amtliche Statistik. Techniken wie etwa SMPC, die im Rahmen von PPML zum Einsatz kommen, könnten jedoch eventuell sinnvoll in anderen Kontexten für die Arbeit des Statistischen Bundesamtes nutzbar sein.

12 Anwendungsfälle aus der Literatur

Während Anhang A einen detaillierten Überblick über Anwendungsfälle für FL in der Literatur enthält, werden nachfolgend zwei Anwendungsbeispiele genauer vorgestellt. Veröffentlichungen, die eine vergleichbare Übersicht abbilden, wurden unter anderem von Crowson u. a. für den Bereich der Biomedizin [Cro+22], von Antunes u. a. für FL auf elektronischen Gesundheitsakten [Ant+22], von Prayitno u. a. für den gesamten Gesundheitsbereich [Pra+21] sowie von Li u. a. für verschiedene Anwendungsbereiche des Ingenieurwesens [Li+20a] verfasst.

Die folgenden zwei Anwendungsbeispiele widmen sich der Klassifizierung von Patienten mit Lungenkrebs und der Feinstaubvorhersage in einer Großstadt.

12.1 Anwendungsbeispiel 1: Lungenkrebs-Klassifizierung

Im Rahmen der Arbeit mit dem Titel „Distributed learning on 20 000+ lung cancer patients - The Personal Health Train“ von Deist u. a. [Dei+20] wurde ein Klassifikator entwickelt, der vorhersagt, ob ein Lungenkrebspatient in den zwei Jahren nach der ersten Diagnose verstirbt. Der Klassifikator wurde auf Gesundheitsdaten von 23 203 Patienten aus 8 Instituten trainiert. Folgende Attribute liegen in den Trainingsdaten vor: Diagnose, Datum der Diagnose, Status bei letztem Kontrolltermin (lebendig oder tot), Datum des letzten Kontrolltermins und vier standardisierte Attribute, die den Status der Krankheit beschreiben (Tumor, Lymphknoten, Metastasen und Gesamtzustand). Während die Trainingsdaten aus den Jahren 1978 bis 2012 stammen, sind die Validierungsdaten aus den Jahren 2012 bis 2015.

Die Datensätze der teilnehmenden Organisationen sind stark heterogen. Einerseits betrifft dies die Größe: Zwei Teilnehmer haben insgesamt mehr als die Hälfte (mit 43% bzw. 17%) der Daten beigesteuert, während insgesamt weniger als 6% der Daten von drei Krankenhäusern stammen. Andererseits trifft die Heterogenität auch auf die Zielvariable (Überleben nach zwei Jahren) zu: Während bei einem der teilnehmenden Krankenhäuser zwei Jahre nach der Diagnose noch über 89% der Patienten lebten, trifft dies in einem anderen Krankenhaus nur auf knapp 19% der Patienten zu.

Als ML-Methode wird verteilte Logistic Regression ohne Strafterme genutzt, die auf der Alternating Direction Method of Multipliers (ADMM) [Boy+11] basiert und in MATLAB implementiert wurde. Der verwendete Code ist öffentlich einsehbar[1]. Da ein realer Anwendungsfall beschrieben wird, in dem Patientendaten nicht den Ort der Datenerhebung verlassen, konnte kein Benchmark des zentralisierten MLs errechnet werden, für den die Daten alle zentral gesammelt vorliegen müssten.

1. https://github.com/RadiationOncologyOntology/20kChallenge, abgerufen am 12. November 2022

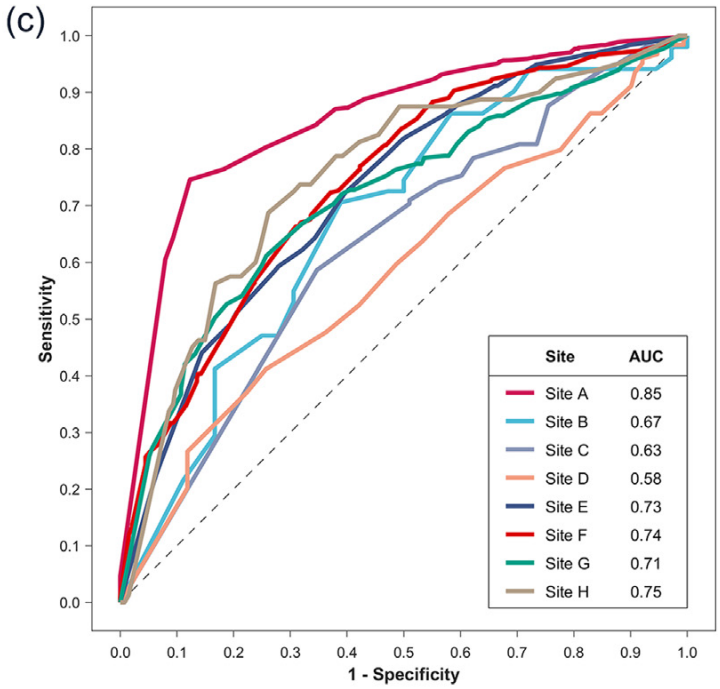

Abbildung 12.1: AUC der teilnehmenden Krankenhäuser in Anwendungsbeispiel 1 (Lungenkrebs Klassifizierung). Grafik entnommen aus [Dei+20].

Das Endergebnis des FL-Lernprozesses ist ein globales Gesamtmodell. Da es jedoch keinen global verfügbaren Testdatensatz gibt, findet auch die Validierung dieses Modells dezentral, jeweils in den Datensilos der FL-Teilnehmer mit eigenen Testdaten statt. Entsprechend unterschiedlich sind auch die Ergebnisse der Validierung: Das tainierte Gesamtmodel erreicht je nach Standort der Validierung jeweils eine Area under the ROC Curve (AUC) zwischen 0,58 und 0,85 (siehe Abbildung 12.1).

Die beiden Institutionen mit den kleinsten AUC Werten gehören zu jenen dreien, die am wenigsten Trainingsdaten beigesteuert haben. Insgesamt wird ein Root Mean Square Error (RMSE) von 0,42 erreicht. Das Modell konvergierte nach 81 Iterationen (in der Veröffentlichung 25 Minuten). Weitere detaillierte Angaben zur Datenverteilung und den Ergebnissen können [Dei+20] entnommen werden.

Übertragbarkeit auf die amtliche Statistik

Die Relevanz des beschriebenen Anwendungsbeispiels aus [Dei+20] für die amtliche Statistik sehen wir vor allem im realistischen Kollaborationsszenario und dem medizinischen Kontext. Dabei wurde eine kommerzielle Lösung namens *Varian Learning Portal* genutzt, die mit einen *learning connector*, der lokal bei den Clients installiert wurde, verbunden wurde. Die zentrale Instanz hatte während des gesamten Vorgangs keinen Zugang zu den Klardaten der Teilnehmer. Um dennoch sicherzustellen, dass die Daten unter den Teilnehmern kompatibel sind, wurde ein detailliertes Datenmodell namens *FAIR* entworfen und veröffentlicht. Das Vorgehen der Autoren kann wertvolle Hinweise darauf geben, wie auch in der amtlichen Statistik erfolgreich FL-Modelle trainiert werden können.

Abbildung 12.2: Aufteilung der 13 meteorologischen Messstationen in Beijing. Die strichlierten Linien kennzeichnen die Regionen, die dem „Region-Learning“ in [Hu+18] zugrunde liegen. Abbildung entnommen aus [Hu+18].

12.2 Anwendungsbeispiel 2: Feinstaubvorhersage

Im Paper „Federated region-learning: An edge computing based framework for urban environment sensing“ [Hu+18] stellen Hu u. a. einen Ansatz zur Schätzung der Feinstaubbelastung in einer Großstadt vor. Die Datengrundlage ist ein öffentlich verfügbarer Datensatz mit „mehr als 100 000“ Datenpunkten, für den allerdings keine Quelle genannt wird[2]. Die Daten stammen von 13 meteorologischen Messstationen in Peking (China) und enthalten verschiedene Wetter- und Luftverschmutzungsattribute wie Temperatur, Luftdruck, Windstärke, SO_3-Belastung, CO-Belastung und Feinstaubbelastung ($PM_{2.5}$). Die modellierte Aufgabe besteht darin, die Höhe der $PM_{2.5}$-Feinstaubbelastung anhand der Wetterdaten der vorhergehenden 48 Stunden zu schätzen. Dies wird mittels eines Long Short-Term Memory Network (LSTM) in Form von Klassifikation umgesetzt. Es werden somit keine konkreten Werte geschätzt, sondern lediglich Belastungsbereiche (bspw. *niedrig*, *mittel* und *hoch*).

Der von den Autoren vorgestellte Ansatz des „Region-Learnings“ ist dabei eine Form von FL-Personalisierung (siehe Kapitel 7), wobei pro Region der Stadt ein separates Modell gelernt wird. Der Zuteilung der Messstationen zu Regionen wird anhand der Ähnlichkeit ihrer Datenpunkte, kombiniert mit der geographischen Nähe der Stationen zueinander, vorgenommen. Das Resultat jener Zuteilung ist in Abbildung 12.2 ersichtlich.

Neben einem Benchmark des zentralisierten Lernens (für den alle Datenpunkte in einen globalen Trainingsdatensatz zusammengeführt wurden) wird Region-Learning auch mit Standard-FL verglichen. Letzteres schneidet mit 76% Testgenauigkeit insgesamt etwas schlechter ab als zentralisiertes Lernen (80%) und Federated Region-Learning (81%). Die Ansätze werden weiterhin bezüglich ihres Kommunikationsaufwands (in Kommunikationsrunden) verglichen und mit verschiedenen Parametern für die Regionszuteilung, sowie für die Batchgröße der Trainingsdaten (96 und 196) evaluiert. Dabei

2. In einer der im Anhang B des Berichts beschriebenen Simulationen wird ein ähnlicher Datensatz verwendet, der mit über 420 000 Datenpunkten von 12 Messstationen etwas umfangreicher ist.

fällt auf, dass der Kommunikationsaufwand für das zentralisierte Lernen als am höchsten angegeben wird – wobei der Begriff der „Kommunikationsrunden“ nicht definiert wird. Zudem scheint die kleinere Batchgröße 96 meist bessere Ergebnisse zu erzielen, wenn die anderen Hyperparameter optimal gewählt sind.

Die Reproduzierbarkeit der Ergebnisse in [Hu+18] ist aufgrund zahlreicher Informationslücken leider nicht gegeben: Es fehlt eine konkrete Quelle zum verwendeten Datensatz sowie Informationen über die zum Training verwendeten Features, die Aufteilung in Trainings- und Testdaten, die Anzahl der für die Klassifizierung verwendeten Klassen und die jeweils verwendete Modellarchitektur. Die im Anhang B beschriebene Simulation 2, die diesem Anwendungsbeispiel nachempfunden ist, lässt sich somit nur sehr eingeschränkt mit den Ergebnissen in [Hu+18] vergleichen.

Übertragbarkeit auf die amtliche Statistik

Wenngleich die Daten in diesem Anwendungsfall öffentlich vorliegen und somit auch ein zentralisiertes ML-Training möglich ist, ist der Kontext des Anwendungsfalls (Umweltbelastung durch Feinstaub) der amtlichen Statistik ähnlich. Zudem ermöglicht die FL-Simulation in diesem Fall einen Vergleich mit zentralisiertem Lernen, der im ersten Anwendungsfall nicht möglich war. Dabei zeigt sich, dass die Performance von FL um einige Prozentpunkte schlechter abschneidet als die des zentralisierten Lernens. Dennoch könnte in ähnlichen Anwendungsfällen, in denen Umweltbelastungsdaten nicht veröffentlicht wurden, durch FL Vorhersagemodelle trainiert werden, die der Öffentlichkeit einen Mehrwert bieten könnten.

13 Schlussbemerkungen

Insgesamt lässt sich festhalten, dass FL (in seiner Standardausführung ohne Personalisierung) in der Regel mit leichten Performanceschwächen im Vergleich zu zentralisierten Lernverfahren einhergehen dürfte. In Fällen, in denen das zentrale Zusammenführen von Daten nicht in Frage kommt, etwa aus wirtschaftlichen oder rechtlichen Gründen, kann FL aber als „Enabler-Technologie“ fungieren und das dezentrale Training von ML-Modellen ermöglichen. Die so trainierten Modelle können der Öffentlichkeit zur Verfügung gestellt werden und somit einen Mehrwert generieren, etwa im Kontext von öffentlicher Gesundheit oder konkreten Fragestellungen einzelner Wirtschaftsbranchen wie dem Tourismus.

Voraussetzung hierfür ist die Gewinnung von (Wirtschafts- oder Verwaltungs-)Partnern, die dazu bereit sind, sich an FL-Trainingsprozessen zu beteiligen. Wenngleich FL die Übertragung der Trainingsdaten selbst unnötig macht, können dabei weiterhin Risiken für die Privatsphäre und Geschäftsgeheimnisse der Teilnehmer bestehen. Wie in Kapitel 9 geschildert, sind hier keine pauschalen Einschätzungen möglich, da etwaige Risiken stets individuell für einzelne Anwendungsfälle untersucht werden müssen. In vielen denkbaren Anwendungen des statistischen Bundesamtes dürften mangels der Verarbeitung individueller personenbezogener Daten allerdings nur wenige der möglichen Risiken zutreffen.

Im Rahmen dieser Studie wurden drei Anwendungsfälle simuliert, die der amtlichen Statistik nahestehen. Hierbei wurde derselbe Datensatz jeweils für das Training von herkömmlichen ML-Modellen genutzt, deren Performance als Vergleichswerte dienen, und im Anschluss FL genutzt. Detaillierte Beschreibungen der Datensätze und Anwendungsfälle sowie eine ausführliche Evaluation sind in einer separaten Veröffentlichung mit dem Arbeitstitel „On the Applicability of Federated Learning for Official Statistics“ beschrieben (siehe Anhang B).

In den vorgenommen Simulationen hat sich gezeigt, dass insbesondere die Hyperparameteroptimierung ein notwendiger und oftmals aufwendiger Schritt in der Vorbereitung eines FL-Trainingsvorgangs ist. Ähnlich wie der Abstimmungsaufwand zu den Trainingsdaten selbst, inklusive geeigneter Vorverarbeitungsschritte, dürfte die Komplexität mit der Zahl an teilnehmenden Parteien stark ansteigen. Wenngleich die Teilnehmerzahl im angestrebten *Cross-Silo*-Szenario begrenzt ist, ist hier eine gute Vorbereitung zu empfehlen – etwa durch das Erstellen eines detaillierten Datenmodells, das Anfertigen eines geeigneten Testdatensatzes und einen simulierten Testlauf vor dem eigentlichen Training.

Literatur

[Aca+21] Durmus Alp Emre Acar u. a. *Federated Learning Based on Dynamic Regularization*. In: *CoRR* abs/2111.04263 (2021). arXiv: 2111.04263. URL: https://arxiv.org/abs/2111.04263.

[Ant+22] Rodolfo Stoffel Antunes u. a. *Federated Learning for Healthcare: Systematic Review and Architecture Proposal*. In: *ACM Trans. Intell. Syst. Technol.* 13.4 (Mai 2022). ISSN: 2157-6904. DOI: 10.1145/3501813. URL: https://doi.org/10.1145/3501813.

[Ari+19] Manoj Ghuhan Arivazhagan u. a. *Federated Learning with Personalization Layers*. 2019. DOI: 10.48550/ARXIV.1912.00818. URL: https://arxiv.org/abs/1912.00818.

[Bai+21] Xiang Bai u. a. *Advancing COVID-19 diagnosis with privacy-preserving collaboration in artificial intelligence*. In: *Nature Machine Intelligence* 3.12 (2021), S. 1081–1089. ISSN: 2522-5839. DOI: 10.1038/s42256-021-00421-z. URL: https://doi.org/10.1038/s42256-021-00421-z.

[Bar+21] Thomas Bartz-Beielstein u. a. *Surrogate Model Based Hyperparameter Tuning for Deep Learning with SPOT*. In: *arXiv preprint arXiv:2105.14625* (2021).

[Beu+20] Daniel J. Beutel u. a. *Flower: A Friendly Federated Learning Research Framework*. 2020. DOI: 10.48550/ARXIV.2007.14390. URL: https://arxiv.org/abs/2007.14390.

[BH18] Tal Ben-Nun und Torsten Hoefler. *Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis*. 2018. DOI: 10.48550/ARXIV.1802.09941. URL: https://arxiv.org/abs/1802.09941.

[Bis+21] Bernd Bischl u. a. *Hyperparameter optimization: Foundations, algorithms, best practices and open challenges*. In: *arXiv preprint arXiv:2107.05847* (2021).

[BLP05] Thomas Bartz-Beielstein, Christian WG Lasarczyk und Mike Preuß. *Sequential parameter optimization*. In: *2005 IEEE congress on evolutionary computation*. Bd. 1. IEEE. 2005, S. 773–780.

[Bon+17] Keith Bonawitz u. a. *Practical Secure Aggregation for Privacy Preserving Machine Learning*. Cryptology ePrint Archive, Paper 2017/281. https://eprint.iacr.org/2017/281. 2017. URL: https://eprint.iacr.org/2017/281.

[Bon+19] Keith Bonawitz u. a. *Towards federated learning at scale: System design*. In: *Proceedings of Machine Learning and Systems* 1 (2019), S. 374–388.

[Boy+11] Stephen Boyd u. a. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. In: *Foundations and Trends® in Machine learning* 3.1 (2011), S. 1–122.

[Bri+18] Theodora S. Brisimi u. a. *Federated learning of predictive models from federated Electronic Health Records*. In: *International Journal of Medical Informatics* 112 (2018), S. 59–67. ISSN: 1386-5056. DOI: https://doi.org/10.1016/j.ijmedinf.2018.01.007. URL: https://www.sciencedirect.com/science/article/pii/S138650561830008X.

[BX22] Nathalie Baracaldo und Runhua Xu. *Protecting Against Data Leakage in Federated Learning: What Approach Should You Choose?* In: *Federated Learning: A Comprehensive Overview of Methods and Applications*. Hrsg. von Heiko Ludwig und Nathalie Baracaldo. Cham: Springer International Publishing, 2022. ISBN: 978-3-030-96896-0. DOI: 10.1007/978-3-030-96896-0_13. URL: https://doi.org/10.1007/978-3-030-96896-0_13.

[CG16] Tianqi Chen und Carlos Guestrin. *XGBoost: A Scalable Tree Boosting System*. In: *CoRR* abs/1603.02754 (2016). arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754.

[Cha+18] Ken Chang u. a. *Distributed deep learning networks among institutions for medical imaging*. In: *Journal of the American Medical Informatics Association* 25.8 (März 2018), S. 945–954. ISSN: 1527-974X. DOI: 10.1093/jamia/ocy017. eprint: https://academic.oup.com/jamia/article-pdf/25/8/945/34150259/ocy017.pdf. URL: https://doi.org/10.1093/jamia/ocy017.

[Che+19] Kewei Cheng u. a. *SecureBoost: A Lossless Federated Learning Framework*. In: *CoRR* abs/1901.08755 (2019). arXiv: 1901.08755. URL: http://arxiv.org/abs/1901.08755.

[Che+22] Hong-You Chen u. a. *On pre-training for federated learning*. In: *arXiv preprint arXiv:2206.11488* (2022).

[Cho+20] Olivia Choudhury u. a. *A syntactic approach for privacy-preserving federated learning*. In: *ECAI 2020*. IOS Press, 2020, S. 1762–1769.

[Cro+22] Matthew G. Crowson u. a. *A systematic review of federated learning applications for biomedical data*. In: *PLOS Digital Health* 1.5 (Mai 2022). DOI: 10.1371/journal.pdig.0000033. URL: https://doi.org/10.1371/journal.pdig.0000033.

[Day+21] Ittai Dayan u. a. *Federated learning for predicting clinical outcomes in patients with COVID-19*. In: *Nature medicine* 27.10 (2021), S. 1735–1743.

[Dea+12] Jeffrey Dean u. a. *Large Scale Distributed Deep Networks*. In: *Advances in Neural Information Processing Systems*. Hrsg. von F. Pereira u. a. Bd. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/6aca97005c68f1206823815f66102863-Paper.pdf.

[Dei+20] Timo M. Deist u. a. *Distributed learning on 20 000+ lung cancer patients - The Personal Health Train*. In: *Radiotherapy and Oncology* 144 (März 2020), S. 189–200. ISSN: 0167-8140. DOI: 10.1016/j.radonc.2019.11.019. URL: https://doi.org/10.1016/j.radonc.2019.11.019.

[DLJ20] Zhongxiang Dai, Bryan Kian Hsiang Low und Patrick Jaillet. *Federated Bayesian optimization via Thompson sampling*. In: *Advances in Neural Information Processing Systems* 33 (2020), S. 9687–9699.

[Dou+21] Qi Dou u. a. *Federated deep learning for detecting COVID-19 lung abnormalities in CT: a privacy-preserving multinational validation study*. In: *npj Digital Medicine* 4.1 (2021), S. 60. ISSN: 2398-6352. DOI: 10.1038/s41746-021-00431-6. URL: https://doi.org/10.1038/s41746-021-00431-6.

[DR+14] Cynthia Dwork, Aaron Roth u. a. *The algorithmic foundations of differential privacy*. In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014), S. 211–407.

[FKH18] Stefan Falkner, Aaron Klein und Frank Hutter. *BOHB: Robust and efficient hyperparameter optimization at scale*. In: *International Conference on Machine Learning*. PMLR. 2018, S. 1437–1446.

[Fol+22] Patrick Foley u. a. *OpenFL: the open federated learning library*. In: *Physics in Medicine & Biology* (2022). DOI: 10.1088/1361-6560/ac97d9. URL: http://iopscience.iop.org/article/10.1088/1361-6560/ac97d9.

[Gen09] Craig Gentry. *Fully homomorphic encryption using ideal lattices*. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, S. 169–178.

[GG19] Lodovico Giaretta und Šarūnas Girdzijauskas. *Gossip Learning: Off the Beaten Path*. In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, S. 1117–1124. DOI: 10.1109/BigData47090.2019.9006216.

[Gia+18] Irene Giacomelli u. a. *Privacy-Preserving Collaborative Prediction using Random Forests*. In: *CoRR* abs/1811.08695 (2018). arXiv: 1811.08695. URL: http://arxiv.org/abs/1811.08695.

[GKN17] Robin C Geyer, Tassilo Klein und Moin Nabi. *Differentially private federated learning: A client level perspective*. In: *arXiv preprint arXiv:1712.07557* (2017).

[GR18] Otkrist Gupta und Ramesh Raskar. *Distributed learning of deep neural network over multiple agents*. In: *Journal of Network and Computer Applications* 116 (2018), S. 1–8. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2018.05.003. URL: https://www.sciencedirect.com/science/article/pii/S1084804518301590.

[HAP17] Briland Hitaj, Giuseppe Ateniese und Fernando Perez-Cruz. *Deep models under the GAN: information leakage from collaborative deep learning*. In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017, S. 603–618.

[Har+18] Andrew Hard u. a. *Federated learning for mobile keyboard prediction*. In: *arXiv preprint arXiv:1811.03604* (2018).

[HDJ19] István Hegedűs, Gábor Danner und Márk Jelasity. *Gossip Learning as a Decentralized Alternative to Federated Learning*. In: *Distributed Applications and Interoperable Systems*. Hrsg. von José Pereira und Laura Ricci. Cham: Springer International Publishing, 2019, S. 74–90.

[He+20] Chaoyang He u. a. *FedML: A Research Library and Benchmark for Federated Machine Learning*. 2020. DOI: 10.48550/ARXIV.2007.13518. URL: https://arxiv.org/abs/2007.13518.

[HR20] Filip Hanzely und Peter Richtárik. *Federated Learning of a Mixture of Global and Local Models*. 2020. DOI: 10.48550/ARXIV.2002.05516. URL: https://arxiv.org/abs/2002.05516.

[Hsi+20] Kevin Hsieh u. a. *The non-iid data quagmire of decentralized machine learning*. In: *International Conference on Machine Learning*. PMLR. 2020, S. 4387–4398.

[Hu+18] Binxuan Hu u. a. *Federated region-learning: An edge computing based framework for urban environment sensing*. In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, S. 1–7.

[Iyy+15] Mohit Iyyer u. a. *Deep unordered composition rivals syntactic methods for text classification*. In: *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. 2015, S. 1681–1691.

[JGS19] Hadi S Jomaa, Josif Grabocka und Lars Schmidt-Thieme. *Hyp-rl: Hyperparameter optimization by reinforcement learning*. In: *arXiv preprint arXiv:1906.11527* (2019).

[Joc+16] Arthur Jochems u. a. *Distributed learning: Developing a predictive model based on data from multiple hospitals without data leaving the hospital - A real life proof of concept*. In: *Radiotherapy and Oncology* 121.3 (Dez. 2016), S. 459–467. ISSN: 0167-8140. DOI: 10.1016/j.radonc.2016.10.002. URL: https://doi.org/10.1016/j.radonc.2016.10.002.

[Joc+17] Arthur Jochems u. a. *Developing and Validating a Survival Prediction Model for NSCLC Patients Through Distributed Learning Across 3 Countries*. In: *International Journal of Radiation Oncology, Biology, Physics* 99.2 (Okt. 2017), S. 344–352. ISSN: 0360-3016. DOI: 10.1016/j.ijrobp.2017.04.021. URL: https://doi.org/10.1016/j.ijrobp.2017.04.021.

[JT16] Kevin Jamieson und Ameet Talwalkar. *Non-stochastic best arm identification and hyperparameter optimization*. In: *Artificial intelligence and statistics*. PMLR. 2016, S. 240–248.

[Kai+19] Peter Kairouz u. a. *Advances and Open Problems in Federated Learning*. 2019. DOI: 10.48550/ARXIV.1912.04977. URL: https://arxiv.org/abs/1912.04977.

[KB14] Diederik P. Kingma und Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[KH18] Antti Koskela und Antti Honkela. *Learning rate adaptation for federated and differentially private learning*. In: *arXiv preprint arXiv:1809.03832* (2018).

[Kho+21] Mikhail Khodak u. a. *Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing*. In: *Advances in Neural Information Processing Systems* 34 (2021), S. 19184–19197.

[KKP20] Viraj Kulkarni, Milind Kulkarni und Aniruddha Pant. *Survey of Personalization Techniques for Federated Learning*. In: *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*. 2020, S. 794–797. DOI: 10.1109/WorldS450073.2020.9210355.

[Law+20] Andrew Law u. a. *Secure collaborative training and inference for xgboost*. In: *Proceedings of the 2020 workshop on privacy-preserving machine learning in practice*. 2020, S. 21–26.

[LB22] Heiko Ludwig und Nathalie Baracaldo. *Introduction to Federated Learning*. In: *Federated Learning*. Springer, 2022, S. 1–23.

[LDM19] Dianbo Liu, Dmitriy Dligach und Timothy Miller. *Two-stage federated phenotyping and patient representation learning*. In: *Proceedings of the conference. Association for Computational Linguistics. Meeting*. Bd. 2019. NIH Public Access. 2019, S. 283.

[Li+14] Mu Li u. a. *Communication Efficient Distributed Machine Learning with the Parameter Server*. In: *Advances in Neural Information Processing Systems*. Hrsg. von Z. Ghahramani u. a. Bd. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/1ff1de774005f8da13f42943881c655f-Paper.pdf.

[Li+16] L Li u. a. *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. arXiv 2018*. In: *arXiv preprint arXiv:1603.06560* (2016).

[Li+20a] Li Li u. a. *A review of applications in federated learning*. In: *Computers & Industrial Engineering* 149 (2020), S. 106854.

[Li+20b] Tian Li u. a. *Federated Learning: Challenges, Methods, and Future Directions*. In: *IEEE Signal Processing Magazine* 37.3 (2020), S. 50–60. DOI: 10.1109/MSP.2020.2975749.

[Li+20c] Tian Li u. a. *Federated Optimization in Heterogeneous Networks*. In: *Proceedings of Machine Learning and Systems*. Hrsg. von I. Dhillon, D. Papailiopoulos und V. Sze. Bd. 2. 2020, S. 429–450. URL: https://proceedings.mlsys.org/paper/2020/file/38af86134b65d0f10fe33d30dd76442e-Paper.pdf.

[Li+20d] Xiaoxiao Li u. a. *Multi-site fMRI analysis using privacy-preserving federated learning and domain adaptation: ABIDE results*. In: *Medical Image Analysis* 65 (2020). ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2020.101765. URL: https://www.sciencedirect.com/science/article/pii/S1361841520301298.

[Lin20] Yehuda Lindell. *Secure multiparty computation*. In: *Communications of the ACM* 64.1 (2020), S. 86–96.

[Liu+20] Yang Liu u. a. *A Secure Federated Transfer Learning Framework*. In: *IEEE Intelligent Systems* 35.4 (2020), S. 70–82. DOI: 10.1109/MIS.2020.2988525.

[Lud+20] Heiko Ludwig u. a. *IBM Federated Learning: an Enterprise Framework White Paper V0.1*. 2020. DOI: 10.48550/ARXIV.2007.10987. URL: https://arxiv.org/abs/2007.10987.

[MC89] Michael McCloskey und Neal J. Cohen. *Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem*. In: Hrsg. von Gordon H. Bower. Bd. 24. Psychology of Learning and Motivation. Academic Press, 1989, S. 109–165. DOI: https://doi.org/10.1016/S0079-7421(08)60536-8. URL: https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[McM+17] Brendan McMahan u. a. *Communication-efficient learning of deep networks from decentralized data*. In: *Artificial intelligence and statistics*. PMLR. 2017, S. 1273–1282.

[MDA15] Dougal Maclaurin, David Duvenaud und Ryan Adams. *Gradient-based hyperparameter optimization through reversible learning*. In: *International conference on machine learning*. PMLR. 2015, S. 2113–2122.

[Mel+19] Luca Melis u. a. *Exploiting unintended feature leakage in collaborative learning*. In: *2019 IEEE symposium on security and privacy (SP)*. IEEE. 2019, S. 691–706.

[Men+15] Bjoern H. Menze u. a. *The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)*. In: *IEEE Transactions on Medical Imaging* 34.10 (2015), S. 1993–2024. DOI: 10.1109/TMI.2014.2377694.

[Mos19] Hesham Mostafa. *Robust federated learning through representation matching and adaptive hyper-parameters*. In: *arXiv preprint arXiv:1912.13075* (2019).

[MR17] Brendan McMahan und Daniel Ramage. *Federated Learning: Collaborative Machine Learning without Centralized Training Data*. 2017. URL: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (besucht am 23. 10. 2023).

[Nai+22] Divya G Nair u. a. *Exploring SVM for Federated Machine Learning Applications*. In: *Advances in Distributed Computing and Machine Learning*. Springer, 2022, S. 295–305.

[Nea18] Patrick Neary. *Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning*. In: *2018 IEEE international conference on cognitive computing (ICCC)*. IEEE. 2018, S. 73–77.

[NSH19] Milad Nasr, Reza Shokri und Amir Houmansadr. *Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning*. In: *2019 IEEE symposium on security and privacy (SP)*. IEEE. 2019, S. 739–753.

[OBZ22] Yuya Jeremy Ong, Nathalie Baracaldo und Yi Zhou. *Tree-Based Models for Federated Learning Systems*. In: *Federated Learning: A Comprehensive Overview of Methods and Applications*. Hrsg. von Heiko Ludwig und Nathalie Baracaldo. Cham: Springer International Publishing, 2022, S. 27–52. ISBN: 978-3-030-96896-0. DOI: 10.1007/978-3-030-96896-0_2. URL: https://doi.org/10.1007/978-3-030-96896-0_2.

[OHJ13] Róbert Ormándi, István Hegedűs und Márk Jelasity. *Gossip learning with linear models on fully distributed data*. In: *Concurrency and Computation: Practice and Experience* 25.4 (2013), S. 556–571. DOI: https://doi.org/10.1002/cpe.2858. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.2858. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.2858.

[Pra+21] Prayitno u. a. *A Systematic Review of Federated Learning in the Healthcare Area: From the Perspective of Data Properties and Applications*. In: *Applied Sciences* 11.23 (2021). ISSN: 2076-3417. DOI: 10.3390/app112311191.

[Rem+20] Samuel W. Remedios u. a. *Distributed deep learning across multisite datasets for generalized CT hemorrhage segmentation*. In: *Medical Physics* 47.1 (2020), S. 89–98. DOI: https://doi.org/10.1002/mp.13880.

[RFB15] Olaf Ronneberger, Philipp Fischer und Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Hrsg. von Nassir Navab u. a. Cham: Springer International Publishing, 2015, S. 234–241. ISBN: 978-3-319-24574-4.

[RG20] Maria Rigaki und Sebastian Garcia. *A survey of privacy attacks in machine learning*. In: *arXiv preprint arXiv:2007.07646* (2020).

[Sap+19] Yuris Mulya Saputra u. a. *Energy demand prediction with federated learning for electric vehicle networks*. In: *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2019, S. 1–6.

[She+19] Micah J. Sheller u. a. *Multi-institutional Deep Learning Modeling Without Sharing Patient Data: A Feasibility Study on Brain Tumor Segmentation*. In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Hrsg. von Alessandro Crimi u. a. Cham: Springer International Publishing, 2019, S. 92–104. ISBN: 978-3-030-11723-8.

[She+20] Micah J Sheller u. a. *Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data*. In: *Scientific reports* 10.1 (2020), S. 1–12.

[SMS19] Felix Sattler, Klaus-Robert Müller und Wojciech Samek. *Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization under Privacy Constraints*. 2019. DOI: 10.48550/ARXIV.1910.01991. URL: https://arxiv.org/abs/1910.01991.

[Sto+22] Joshua Stock u. a. *Privatsphärefreundliches maschinelles Lernen*. In: *Informatik Spektrum* 45.2 (2022), S. 70–79.

[Swe02] Latanya Sweeney. *k-anonymity: A model for protecting privacy*. In: *International journal of uncertainty, fuzziness and knowledge-based systems* 10.05 (2002), S. 557–570.

[Tan+22] Alysa Ziying Tan u. a. *Towards Personalized Federated Learning*. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022), S. 1–17. DOI: 10.1109/TNNLS.2022.3160699.

[Tru+19] Stacey Truex u. a. *A hybrid approach to privacy-preserving federated learning*. In: *Proceedings of the 12th ACM workshop on artificial intelligence and security*. 2019, S. 1–11.

[Vai+13] Jaideep Vaidya u. a. *A random decision tree framework for privacy-preserving data mining*. In: *IEEE transactions on dependable and secure computing* 11.5 (2013), S. 399–411.

[Vep+20] Praneeth Vepakomma u. a. *NoPeek: Information leakage reduction to share activations in distributed deep learning*. 2020. DOI: 10.48550/ARXIV.2008.09161. URL: https://arxiv.org/abs/2008.09161.

[VML21] Saeed Vahidian, Mahdi Morafah und Bill Lin. *Personalized Federated Learning by Structured and Unstructured Pruning under Data Heterogeneity*. In: (2021). URL: https://arxiv.org/abs/2105.00562.

[Wan+17] Ruoxi Wang u. a. *Deep & Cross Network for Ad Click Predictions*. In: *Proceedings of the ADKDD'17*. ADKDD'17. Halifax, NS, Canada: Association for Computing Machinery, 2017. ISBN: 9781450351942. DOI: 10.1145/3124749.3124754. URL: https://doi.org/10.1145/3124749.3124754.

[Wan+20a] Hao Wang u. a. *Optimizing Federated Learning on Non-IID Data with Reinforcement Learning*. In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 2020, S. 1698–1707. DOI: 10.1109/INFOCOM41043.2020.9155494.

[Wan+20b] Jianyu Wang u. a. *Tackling the objective inconsistency problem in heterogeneous federated optimization*. In: *Advances in neural information processing systems* 33 (2020), S. 7611–7623.

[War+21] Stefanie Warnat-Herresthal u. a. *Swarm learning for decentralized and confidential clinical machine learning*. In: *Nature* 594.7862 (2021), S. 265–270.

[Xu+20] Yongchao Xu u. a. *A collaborative online AI engine for CT-based COVID-19 diagnosis*. In: *medRxiv* (2020). DOI: 10.1101/2020.05.10.20096073. eprint: https://www.medrxiv.org/content/early/2020/05/19/2020.05.10.20096073.full.pdf. URL: https://www.medrxiv.org/content/early/2020/05/19/2020.05.10.20096073.

[Xu+21] Jie Xu u. a. *Federated Learning for Healthcare Informatics*. In: *Journal of Healthcare Informatics Research* 5.1 (2021), S. 1–19. ISSN: 2509-498X. DOI: 10.1007/s41666-020-00082-4. URL: https://doi.org/10.1007/s41666-020-00082-4.

[Yan+19a] Mengwei Yang u. a. *The Tradeoff Between Privacy and Accuracy in Anomaly Detection Using Federated XGBoost*. In: *CoRR* abs/1907.07157 (2019). arXiv: 1907.07157. URL: http://arxiv.org/abs/1907.07157.

[Yan+19b] Qiang Yang u. a. *Federated learning*. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13.3 (2019), S. 1–207.

[Yan+19c] Qiang Yang u. a. *Federated Machine Learning: Concept and Applications*. In: (2019). DOI: 10.48550/ARXIV.1902.04885. URL: https://arxiv.org/abs/1902.04885.

[Yan+19d] Wensi Yang u. a. *Ffd: A federated learning based method for credit card fraud detection*. In: *International conference on big data*. Springer. 2019, S. 18–32.

[YS20] Li Yang und Abdallah Shami. *On hyperparameter optimization of machine learning algorithms: Theory and practice*. In: *Neurocomputing* 415 (2020), S. 295–316.

[Zha+18] Yue Zhao u. a. *Federated Learning with Non-IID Data*. In: (2018). DOI: 10.48550/ARXIV.1806.00582. URL: https://arxiv.org/abs/1806.00582.

[Zha+20] Chengliang Zhang u. a. *{BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning*. In: *2020 USENIX annual technical conference (USENIX ATC 20)*. 2020, S. 493–506.

[Zha+21] Huanle Zhang u. a. *Automatic Tuning of Federated Learning Hyper-Parameters from System Perspective*. In: *arXiv preprint arXiv:2110.03061* (2021).

[Zho+21] Yi Zhou u. a. *FLoRA: Single-shot Hyper-parameter Optimization for Federated Learning*. In: *arXiv preprint arXiv:2112.08524* (2021).

[ZLH19] Ligeng Zhu, Zhijian Liu und Song Han. *Deep leakage from gradients*. In: *Advances in neural information processing systems* 32 (2019).

[ZZJ21] Hangyu Zhu, Haoyu Zhang und Yaochu Jin. *From federated learning to federated neural architecture search: a survey*. In: *Complex & Intelligent Systems* 7.2 (2021), S. 639–657.

A | Anwendungsfälle von Federated Learning (Literaturübersicht)

Publikation	Titel	Bereich	ML-Problemklasse	Beschreibung	Methoden	Software	Quellcode	Ergebnisse
[Dei+20]	Distributed learning on 20 000+ lung cancer patients - The Personal Health Train	Medizin	Binäre Klassifizierung	Basierend auf Gesundheitsdaten zu 23.203 Patienten aus 8 Instituten wurde ein Klassifikator für das Überleben von Lungenkrebspatienten für einen Zeitraum von 2 Jahren nach der Behandlung trainiert.	Es wird verteilte Logistic Regression ohne Strafterme genutzt, die auf ADMM [Boy+11] basiert.	MATLAB	https://github.com/RadiationOncologyOntology/20kChallenge	Das trainierte Modell erreicht eine AUC zwischen 0.58 und 0.85 (in der Veröffentlichung angegeben nur für die Modellperformance in den jeweiligen Krankenhäusern, aber nicht global). Es erreicht einen RMSE von 0.42. Das Modell konvergierte nach 81 Iterationen (in der Veröffentlichung 25 Minuten).
[Li+20d]	Multi-site fMRI analysis using privacy-preserving federated learning and domain adaptation: ABIDE results	Medizin	Binäre Klassifizierung	Ziel der Studie war die Klassifizierung von Autismus-Spektrum-Störungen basierend auf funktionellen Magnetresonanztomographie (MRT)-Bildern von 370 Patienten aus 4 Kliniken, die in dem *Autism-Brain-Imaging-Data-Exchange*-Datensatz vorliegen.	Es kommt eine Art von neuronalem Netz (Multilayer Perceptron) mit 6105 Input- 16 Hidden- und 2 Output-neuronen und Adam-Optimierer [KB14] zum Einsatz. Das globale Modell wird durch den Durchschnitt der lokalen Gewichte gebildet (FedAvg).	Pytorch	https://github.com/xxlya/Fed_ABIDE	Für das Modell ergaben sich Accuracy-Werte zwischen 0.647 und 0.849 (in der Veröffentlichung angegeben nur für die Modellperformance in den jeweiligen Krankenhäusern, aber nicht global). Zusätzlich wurden verschiedene Experimente mit einer Randomisierung der Gewichte (DP mit Laplace- und Gaussian-Noise) durchgeführt, die die Accuracy abhängig von gewählten Parametern unterschiedlich stark beeinflussten.

Publikation	Titel	Bereich	ML-Problemklasse	Beschreibung	Methoden	Software	Quellcode	Ergebnisse
[Bri+18]	Federated learning of predictive models from federated Electronic Health Records	Medizin	Binäre Klassifizierung	Die Studie zielt darauf ab, notwendige Einlieferungen in ein Krankenhaus innerhalb eines Jahres basierend auf elektronischen Gesundheitsakten vorherzusagen. Das Training fand mit einem Datensatz von 45.579 Patienten aus dem Zeitraum von 2001 bis 2012 statt, der zufällig auf 5 – 10 Parteien verteilt wurde.	Die Autoren entwickeln einen iterativen *Cluster-Primal-Dual-Splitting*-Algorithmus um verteilt ein Sparse-SVM-Modell zu trainieren.	*Unbekannt*	*Nicht verfügbar*	Die Autoren geben für das verteilt trainierte Modell eine AUC von 0.7806 an, die leicht über der AUC von in der Veröffentlichung genutzten verwandten Ansätzen liegt. Das Training benötigte 100 Iterationen und 544 Sekunden.
[Joc+17]	Developing and Validating a Survival Prediction Model for NSCLC Patients Through Distributed Learning Across 3 Countries	Medizin	Binäre Klassifizierung	Basierend auf Gesundheitsdaten zu 698 Patienten aus 2 Instituten wurde ein Klassifikator für das Überleben von Lungenkrebspatienten für einen Zeitraum von 2 Jahren nach der Behandlung trainiert. Anschließend erfolgte eine Validierung auf einem Datensatz mit 196 Patienten eines dritten Instituts.	Es wird ein Bayesian Network mit einer manuell definierten Struktur verwendet. Das Zusammenführen geschieht durch weighted Averaging verschiedener Werte (Details sind in [Joc+16]) zu finden.	JSMILE-Bibliothek, R	*Nicht verfügbar*	Für das Modell ergab sich eine AUC von 0.62 in einem Kreuzvalidierungsschritt und 0.66 in einem externen Validierungsschritt.

Publikation	Titel	Bereich	ML-Problemklasse	Beschreibung	Methoden	Software	Quellcode	Ergebnisse
[Day+21]	Federated learning for predicting clinical outcomes in patients with COVID-19	Medizin	Klassifizierung	Es wird ein Risikoscore für das 24- und 72-Stunden-Risiko eines COVID-19-Patienten für den zukünftigen Bedarf nach zusätzlichem Sauerstoff erstellt.	Es wird ein ResNet für das Extrahieren von Features aus Röntgenaufnahmen genutzt. Anschließend erfolgt eine Kombination mit Features aus der Patientenakte und die Nutzung eines Deep & Cross Networks [Wan+17]. Das verteilte Lernen geschieht durch Federated Averaging mit 20 teilnehmenden medizinischen Instituten und insgesamt 16.148 betrachteten Fällen.	Tensorflow, NVIDIA Clara Train SDK	https://catalog.ngc.nvidia.com/orgs/nvidia/teams/med/models/clara_train_covid19_exam_ehr_xray	Das globale Modell verbessert die AUC um 16% (24h) bzw. 18% (72h) auf 0.920 ggü. dem Durchschnitt ausschließlich lokal trainierter Modelle. Beispielhaft (bei Validierung auf neuem Datensatz) ergeben sich eine Sensitivität von 0.950 und eine Spezifität von 0.882. Es wird ein Privacy-Preserving-Ansatz durch Weitergabe partieller Updates (nur die Gewichte mit größtem Beitrag) verfolgt.
[Xu+20; Bai+21]	A collaborative online AI engine for CT-based COVID-19 diagnosis	Medizin	Klassifizierung	Ziel der Studie war die Klassifikation von Lungenentzündungen durch COVID-19, andere Viren oder Bakterien basierend auf Computertomographie (CT)-Scans des Brustkorbs. Hierbei wurden 1276 Datensätze aus dem Tongji Hospital (1034 Datensätze für Training, 240 für Test) und 507 Datensätze aus dem Wuhan Union Hospital verwendet.	Für die Klassifizierung wurde ein Convolutional Neural Network (CNN) trainiert. Das verteilte Training wurde mithilfe von FedAvg umgesetzt.	PyTorch	https://github.com/HUST-EIC-AI-LAB/UCADI	Die Autoren geben für das trainierte Modell eine Sensitivität von 0.98 an. Additive HE (durch *Learning with Errors*) schützt die geteilten Gewichte auf Kosten höherer Rechenzeit und Datenübertragung.

Publikation	Titel	Bereich	ML-Problemklasse	Beschreibung	Methoden	Software	Quellcode	Ergebnisse
[Rem+20]	Distributed deep learning across multisite datasets for generalized CT hemorrhage segmentation	Medizin	Bildsegmentierung	Ziel der Studie war die Bildsegmentierung von CT-Scans zur Erkennung von Hirnblutunn. Hierbei kamen Daten aus 5 Instituten (69 Scans für das Training, 94 Scans für die Validierung) zum Einsatz.	Es wurde eine besondere Art CNN (U-Net [RFB15]) unter Nutzung des Adam-Optimierers trainiert. Für das verteilte Lernen kam *Cyclic Weight Transfer* zum Einsatz (siehe Abschnitt 3.4).	keras, Tensorflow	https://github.com/MASILab/tbi_ct_lesion_segmentation	In der Studie wurde eine Verbesserung des Dice Similarity Coefficient (DSC) von 7% ggü. dem Durschnitt der auf den einzelnen Institutsdaten trainierten Modelle auf 0.690 erreicht.
[She+20; She+19]	Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data	Medizin	Bildsegmentierung	Basierend auf dem öffentlichen Brain-Tumor-Segmentation-Datensatz (BraTS [Men+15]) wurde ein Modell zur Segmentierung von MRT-Aufnahmen zur Identifikation von Hirntumoren trainiert. Hierbei wurden 210 MRT-Scans aus 10 Krankenhäusern für das Training und 196 weitere für Test und Validierung genutzt.	Ein U-Net und der Adam-Optimierer werden verwendet. Der Zustand der Optimierung wird per gewichtetem Durchschnitt aggregiert.	Tensorflow	https://github.com/IntelAI/unet/tree/master/2D	Für die Validierung der Segmentierungsergebnisse wird auf den DSC zurückgegriffen und es konnte gezeigt werden, dass der FL-basierte Ansatz (0.852) nur minimal schlechter abschneidet als ein zentralisierter Ansatz (0.862).
[Dou+21]	Federated deep learning for detecting COVID-19 lung abnormalities in CT: a privacy-preserving multinational validation study	Medizin	Objekterkennung	Ziel der Studie war die Erkennung von Läsionen in CT-Scans von an COVID-19 erkrankten Patienten. Zu diesem Zweck wurden 75 Scans aus drei Krankenhäusern in Hongkong für das Training verwendet und 55 weitere Scans aus drei verschiedenen Quellen für die Validierung verwendet.	In der Studie wird ein CNN mittels FedAvg und dem Adam-Optimierer trainiert. Weiterhin nutzen die Autoren Transfer Learning mit einem in einer vorhergehenden Studie trainierten Modell.	Tensorflow	https://github.com/med-air/FL-COVID	Die Autoren geben für die drei externen Testdatensätze die Sensitivitäten 90.35, 73.31 und 85.06, die Spezifitäten 88.56, 91.93 und 87.94 sowie die AUC 95.66, 88.15 und 91.99 an. Mit diesen Werten konnte das verteilte Training die Werte lokal trainierter Modelle fast immer (teilweise erheblich) verbessern.

Publikation	Titel	Bereich	ML-Problemklasse	Beschreibung	Methoden	Software	Quellcode	Ergebnisse
[LDM19]	Two-stage federated phenotyping and patient representation learning	Medizin	Representation learning und Natural Language Processing (NLP)	Ziel des Papers ist föderiertes NLP, um mithilfe dezentral verfügbarer klinischer Notizen eine Kohortenzuordnung von Patienten vorzunehmen. Die erste Stufe bildet ein Representation Learning, in der zweiten Stufe werden die daraus gewonnen Resultate für ein Phenotyping (Einschätzung von Adipositas und 15 Komorbiditäten) genutzt.	In der ersten Stufe wird ein Deep Averaging Network (DAN) [Iyy+15] trainiert. Das Modell wird ohne output layer in der zweiten Stufe als Eingabe für eine SVM genutzt, um die Phänotypen zu bestimmen. Für das föderierte DAN-Training kommt FedAvg zum Einsatz. Ob und wie das SVM-Modell föderiert trainiert wird, wird nicht beschrieben.	*Unbekannt*	Verweis auf leeres Repository[1]	Die Autoren geben Precision, Recall und F1-Score für verschiedene Szenarien an. In der komplett föderierten Lösung ist der Wert mit 0,72 deutlich höher als in der zentralisierten (0,63).
[Hu+18]	Federated region-learning: An edge computing based framework for urban environment sensing	Luftverschmutzung in Großstädten	Klassifizierung	Auf der Basis von 48h Wetter- und Luftverschmutzungsdaten wird die aktuelle Luftverschmutzung geschätzt. Zur Verbesserung der Ergebnisse wird aufbauend auf normalem FL „Federated Region-Learning“ eingeführt, das die Daten von geographisch nah beieinander gelegenen Wetterstationen mit ähnlichen Daten zusammen gruppiert. Insgesamt sind über 100.000 Datenpunkte von 13 Wetterstationen in die Analyse eingeflossen.	Die Autoren trainieren ein LSTM in drei Szenarien: 1. zentralisiert (als Benchmark), 2. per Standard FL (mit FedAvg)	*Unbekannt*	*Nicht verfügbar*, öff. Datensatz (ähnlich)[2]	Klassisches FL schneidet mit 76% Testgenauigkeit insgesamt etwas schlechter ab als zentralisiertes Lernen (80%) und Federated Region-Learning (81%)

1. https://github.com/kaiyuanmifen/FederatedNLP, (abgerufen am 12.10.2022)
2. https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data (abgerufen am 10.10.2022)

Publikation	Titel	Bereich	ML-Problemklasse	Beschreibung	Methoden	Software	Quellcode	Ergebnisse
[Sap+19]	Energy Demand Prediction with Federated Learning for Electric Vehicle Networks	Energieverbrauchvorhersage	Regression	Öffentliche Daten über die Nutzung von 58 Ladesäulen für elektrische Fahrzeuge in einer englischen Stadt wurden dafür genutzt, um die jeweils abgegebene Energiemenge einer Transaktion in kWh zu schätzen. Insgesamt wurden über 65.000 Datenpunkte verwendet. Es wurden zwei auf das Problem zugeschnittene Lernalgorithmen (EDL und die föderierte Version FEDL) evaluiert. Zusätzlich kam ein Clustering von Ladesäulen mit ähnlichen Daten zum Einsatz.	Ein DNN wird mit Adam-Optimierer und per FedAvg trainiert.	TensorFlow	*Nicht verfügbar*, Datensatz: https://data.dundeecity.gov.uk/dataset/ev-charging-data	RMSE-Auswertungen für die jeweiligen Algorithmen zeigen, dass EDL und FEDL deutlich besser als die klassischen zentralisierten Lernalgorithmen abschneiden (etwa 5,8 statt 7,7–6,5); Clustering verbessert die Ergebnisse von EDL und FEDL weiterhin minimal.
[Yan+19d]	FFD: A Federated Learning Based Method for Credit Card Fraud Detection	Finanzwesen	Klassifizierung	Der verwendete Datensatz enthält 284.807 Kreditkartentransaktionen, die per Principal component analysis (PCA) komprimiert wurden. Ziel des Algorithmus ist die Erkennung von auffälligen Transaktionen. Die Daten werden zufällig auf 100 fiktive Banken aufgeteilt. Durch Oversampling wird der Datensatz ausgeglichen, da ursprünglich nur 0,17% auffällige Transaktionen im Datensatz vorhanden sind.	Ein CNN wird per FedAvg trainiert.	*Unbekannt*	*Nicht verfügbar*, Datensatz: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud	Mit Oversampling wird ein F1-Wert von bis zu 1 erreicht wird (vs. 0,7 ohne Ausgleich). Zentralisierte Methoden kommen auf bis zu 0,89 (siehe bspw. https://www.kaggle.com/code/niallmcnulty/fraud-a-comparison-of-models-and-sampling-methods).

Tabelle A.1: Übersicht über Anwendungsfälle

B Veröffentlichung der Simulationsergebnisse

Im Zuge der Studie wurden Simulationen durchgeführt, die Anwendungsfällen der amtlichen Statistik ähneln und in Form einer wissenschaftlichen Publikation evaluiert wurden. Da die Veröffentlichung jenes Dokuments noch aussteht, ist es auf den nachfolgenden Seiten angehängt. Dabei wurde das Layout nicht verändert und auch die angegebenen Seitenzahlen entsprechen den Seiten innerhalb der Publikation.

On the Applicability of Federated Learning for Official Statistics

Joshua Stock, Tom Petersen, Hannes Federrath
Universität Hamburg
{joshua.stock, tom.petersen, hannes.federrath}@uni-hamburg.de

February 13, 2023

Abstract

This work investigates the potential of Federated Learning for official statistics. By simulating three different use cases, two of which are based on large, real-world data sets, important insights on the applicability of the technology are gained. The use cases are based on a medical insurance data set, a mobile radio coverage data set and a fine dust pollution data set – all of which are from domains close to official statistics. We provide a detailed analysis of the results, including an in-depth comparison of centralized and federated learning algorithm performances for each simulation. As a main contribution, key observations and their implications for transferring the simulations into practice are summarized. We arrive at the conclusion that Federated Learning can become a key technology in future use cases of official statistics, although some work still needs to be done to facilitate the Federated Learning process and ensure the privacy of data holders.

1 Introduction

In the context of official statistics, there are standard tasks such as the preparation of census data and computing the GDP and inflation rates with established data sources. But like in many other domains, the work of official statistical agencies such as the Federal Statistical Office of Germany has been evolving in the past years, as more and more complex use cases are arising. Some of these new use cases involve machine learning (ML) models, which need to be trained with large amounts of data. Often, data from other official agencies or private companies is needed for these tasks. But, among other reasons, legal restrictions and competitive markets often make it difficult to share data with others.

To overcome these issues, we investigate the application of Federated Learning [McM+17], as it enables training ML models cooperatively without the need to transmit private data. In this paper, the results of a study regarding the applicability of Federated Learning for official statistics are presented. After providing a brief introduction to machine learning and Federated Learning in Section 2, we present a detailed description of three simulations which we have conducted in Section 3 and provide key observations from these simulations in Section 4. The simulations are based on use cases close to the domain of official statistics, such that we can draw conclusions on the topic of Federated Learning and official statistics in Section 5, which are summarized in Section 6.

2 Background

In classical machine learning algorithms, a model is designed and then fitted (or "trained") on a training data set. The idea is to iteratively adapt the trainable parameters of a model to perform well on a given task. Standard tasks for such algorithms are classification, regression or clustering. The automated training process allows machine learning models to capture complex patterns in the data which might not be visible even to human experts. This has led to the widespread use of machine learning models in both the every day lives of most people (for example next-word-prediction in smartphone keyboards, anti-bot CAPTCHAs on websites, or automatically

generated subtitles of videos) and expert domains such as automated medical image annotations or more accurate weather and climate models.

Usually, the unit which trains a model is also the owner of the training data. However, there are many scenarios in which potential training data is spread across multiple data holders. In such cases, the easiest way to train a machine learning model is often to first combine all available data on a central server and then train the model. But if data contains sensitive information, this solution is unfavorable since all information will be made available on the central server.

2.1 Federated Learning

Instead, McMahan et al. have presented Federated Learning (FL) in [McM+17]. It describes a special form of decentralized machine learning, in which participating data holders do not need to share their private training data during the training process. Instead, a central server (the aggregator) coordinates the FL training process, initializing a global model and forwarding it to the participants. In each training round, each participant trains the model with their private data and sends the resulting model back to the aggregator. The aggregator combines the updates of the participants into the next iteration of the global model and starts the next round by distributing the updated model to the participants. This can be repeated until the model converges to the desired performance, just as in a regular ML training process.

In standard FL, the primary goal is to train *one global model* over the data of all participating parties, such that the model can be provided to the public afterwards in order for individuals and organizations to profit from their individual queries. An example for such a model is the personalized inflation calculator on the website of the Federal Statistical Office[1]. This is in contrast to model personalization, where the goal is to train multiple models, which derive from the global model to reach a better performance on individual client data sets (e.g. [VML21; Hu+18]).

In contrast to the *cross-device* FL setting, where thousands or even millions of participants jointly train a model, we focus on the *cross-silo* setting in this paper. This is due to the fact that as explained above, most use cases in official statistics are focused on scenarios in which data is split across a few data holders (such as companies) and not on thousands of, e.g., smartphones of individual people.

2.2 Hyperparameters

Most machine learning techniques such as random forests [Bre01] and neural networks need to be parametrized with *hyperparameters*. For decision trees (of which random forests build an ensemble), an example of a hyperparameter is the maximal depth which a tree is allowed to reach. For neural networks, the most important hyperparameters capture the architecture of a network, such as the number of layers or the number of neurons per layer. But also other parameters need to be set, like the kind of activation function used in each neuron and the batch size of the data during training.

While there exist many frameworks which find optimal hyperparameters (semi-)automatically for centralized learning algorithms [YS20], hyperparameter optimization (HPO) is a more difficult issue for FL-algorithms. On the one hand, most HPO techniques work by testing many different training configurations. Since the whole training process needs to be repeated multiple times, usually a lot of time and computational resources are needed, which is often not available in FL scenarios. On the other hand, the central server (the party conducting the HPO) needs to have a consistent test set during the HPO process. If the central party has no test data available (as it is the case in many FL scenarios), clients who drop out during the HPO process might cloud the comparability of different configurations. Although many approaches have been published in this area (e.g. [Mos19; Kho+21; DLJ20; Zho+21; KH18]), there seem to be no established standard techniques [Kai+19].

[1] Personal Inflation Calculator at https://service.destatis.de/inflationsrechner/EN/, accessed on January 20, 2023.

Attribute	Value range
age	18–64
gender	“male” or “female”
bmi	16–53.1
children	0–5
smoker	“yes” or “no”
region	“southwest”, “southeast”, “northwest” or “northeast”
charges	1121.8739 and 63 770.428 01

Table 1: Attributes and value ranges of the medical insurance data set.

2.3 Frameworks

For the centralized benchmark models, we use the frameworks TensorFlow[2] (for neural networks) and scikit-learn[3] (for linear regression and random forests). The latter frameworks is also used to preprocess the data sets, such as scaling the data via a *MinMaxScaler*[4]. For the FL training in our simulations, we use the framework TensorFlow Federated[5].

3 Simulations

To analyze the potential of FL for official statistics, we run simulations with three different data sets. The use cases of the simulations are in domains which are close to official statistics. For each use case, we first compute benchmarks by training and testing centralized ML models, e.g., models which are trained on the whole data set. Afterwards, we split the data set and assign the parts to (simulated) FL clients for the FL simulation. This way, we have a basis for interpreting the performance of the model resulting from the FL training simulation. The performance metrics of the trained ML models (coefficient of determination R^2 and accuracy) are computed on the test set of each data set.

The code we have written for our experiments is available on GitHub[6].

3.1 Medical insurance data

Data set The first data set, in which medical insurance costs are linked to individuals, is publicly available on the internet[7]. The data set consists of 1338 records. Each record has eight attributes, which are listed in Table 1. The data is complete, i.e. there are no missing attributes for any record. Also, the data set is very balanced: The numbers in the *age* attribute are very evenly dispersed, just as the distribution of male and female records is about 50/50 (attribute *gender*) and each *region* is represented almost equally often. The origin of the data is unknown, however its homogeneity and integrity suggest that it has been created artificially.

Data preprocessing We encodee the binary attributes *sex* and *smoker* into a numeric form (0 or 1). We also scale the attributes *age*, *bmi* and *children* to a range from 0 to 1. We use a one-hot-encoding for the *region* attribute for the centralized benchmarks. For FL, we separate the data set into 4 client data sets based on the *region* attribute (and remove the attribute from the client data sets). In both scenarios, 20% of the data is excluded from the training process as a test data set.

Setup As the ML task, we model the following regression problem: Given the six other attributes of the data set, how high are the insurance *charges* of an individual?

[2] https://www.tensorflow.org/, accessed on January 20, 2022
[3] https://scikit-learn.org/, accessed on January 20, 2023
[4] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html, accessed on January 20, 2023
[5] https://www.tensorflow.org/federated/, accessed on January 12, 2022
[6] https://github.com/joshua-stock/fl-official-statistics
[7] https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset, accessed on January 12, 2022

Parameter	Value
bootstrap	True
max_depth	4
max_features	0.8
min_samples_leaf	5
min_samples_split	9
n_estimators	1050

Table 2: Hyperparameters for the random forest regressor of the medical insurance data simulation.

The two centralized learning approaches random forest and neural network are used as benchmarks for the FL scenario. A hyperparameter search yields the hyperparameters in Table 2. For the second benchmark (neural network), a hyperparameter search yields the following architecture: 16 dense-neurons in the first layer and 2 dense-neurons in the second layer, followed by a single dense-neuron in the output layer. The neural network is trained for 100 epochs, using the stochastic gradient descent (SGD) implementation by TensorFlow.

For the FL scenario, we use a slightly larger neural network with 16 dense-neurons in the first layer and 6 neurons in the second layer, again followed by a single neuron in the output layer. We run the FL training process for 150 rounds, with SGD and a learning rate of 0.8 for the clients and 3.0 for the server.

Results As benchmarks, the random forest regressor achieves an R^2 value of 0.877 on the test data set, while the neural network achieves an R^2 value of 0.85. Federated Learning performs significantly worse, at an R^2 value of -0.075.

Discussion The strong performance of the centralized benchmark models (R^2 of 0.877 and 0.85) may be due to the ML-friendly and possibly artificial nature of the data set. Regarding the significantly worse performance of the FL neural network (R^2 of -0.075), we test the hypothesis that this is rooted in the fact that the FL training data is lacking the *region* attribute. Recall that we construct the FL client data sets by dividing the data set by its region. Without removing the attribute from the data, the attribute would hence be the same in each client data set – thus it has no use in the local training process. To test the influence of the *region* attribute on the model performance, we retrain the benchmark random forest regressor on a copy of the original data set *without* the one-hot encoded *region* attribute. This does not influence the performance of the benchmark, as it still achieves an R^2 of 0.877. We run a second test in the FL scenario: Instead of dividing the data set by the region of the data records, we randomly split the records into 4 evenly sized client data sets and keep the *region* attribute. If the hypothesis above (that the *region* attribute contains useful information for the training process, explaining the strong performance of the benchmark models) is true, we would expect a big improvement of the FL model by keeping the *region* attribute in the (randomly dispersed) client data sets. However, the original result of -0.075 R^2 (with one client per region) has been worsened in this test scenario (with random client data sets) to an R^2 value of -0.106. Thus, we conclude that we can find no evidence for the hypothesis stated above.

Instead, we conjecture that the data set is too small for a FL training scenario – with about 350 data records per client, minus 20% for the test data. Although data augmentation (adding more artificial data with a similar distribution) could be a possible solution to this problem, we want to stress that this is not trivial. In a quick test, in which we have used a generative adversarial network (GAN), we were not able to improve the resulting FL regressor's performance by data augmentation, reaching an R^2 value of -1.02.

3.2 Mobile radio (LTE)

The second use case is based on a private data set by the company umlaut SE[8]. On the one hand, it contains mobile communication network coverage data, including latency and speed tests, each

[8] https://www.umlaut.com/, accessed on January 12, 2022

Feature	Example value
User ID	360a02ef8afd197e2a6fe...
radius of action in meters*	2300
calendar week	20
day of the week	3
month	5
weekend	True
number of data records*	30
variance LTE-RSRQ*	0.84
mean LTE-RSRQ*	-11.3
variance LTE-RSRP*	12.3
mean LTE-RSRP*	-90.8
variance LTE-RSSNR*	7.54
mean LTE-RSSNR*	6.25
variance LTE-RSSI*	12.4
mean LTE-RSSI*	-84.5
share of records with WiFi connection*	0.1

Table 3: Aggregated features of the LTE data set by the company umlaut SE. The *user* attribute is not used as training data, it is merely used to aggregate the features per user. Features marked with * are computed over the aggregated data for one particular day.

linked to the mobile LTE devices of individual users and a specific timestamp. On the other hand, the data records are associated with GPS-coordinates, such that a daily "radius of action" can be computed for each user. This radius describes how far a user has moved from their home base within one day. The user home bases have also been computed on the available data – a home base is defined as the place where most data records have been recorded. The ML task we model in this use case is to estimate the daily radius of action for a user, given different LTE metrics of one particular day (see below).

Data set The whole data set originally contains 286 329 137 data records. The aggregated features of the data set which we have used are depicted in Table 3.

Data preprocessing We set a specific time frame of six months and a geofence around the German state of North Rhine-Westphalia, all other records are excluded – leaving 2 718 416 records in the data set. Additionally, we apply a filtering strategy to clean our data: each user in the data base needs to have data for at least 20 different days (within the time span of six months) and 10 records on each of these days. Otherwise, all records of this user are discarded. After the second filtering step, there are 1 508 102 data records in the data set, which we scale using a standard scaler implementation and then use for training, validating and testing our models.

60% of the data are used as training data, 20% are used as validation data and 20% as test data. For FL, we have divided the data set according to the mobile network operators (MNOs) of the users. Since more than 99.6% of the data records are associated with three major providers, the other 0.4% of the data records (associated with 29 other MNOs) are eliminated from the data set.

Setup As in the previous use case, we use two centralized learning benchmarks: a random forest regressor and a neural network. A hyperparameter search for random forest yields the parameters as shown in Table 4. The network architecture for both the centralized benchmark neural network and the FL training process is the same: The first layer consists of 28 dense-neurons and the second layer consists of 14 dense-neurons, which lead to the single-neuron output layer with a ReLu activation function. For FL, we use a server learning rate of 3.0, a client learning rate of 0.8 and a batch size of 2^9. Additionally, we train a linear regression model as a centralized

[9]Due to time constraints, we were not able to spend many resources on hyperparameter tuning. Hence, these might not be optimal hyperparameters.

Parameter	Value
bootstrap	True
max_depth	20
max_features	0.75
min_samples_leaf	0.001
min_samples_split	0.005
n_estimators	150

Table 4: Hyperparameters for the random forest regressor of the mobile radio (LTE) data simulation.

benchmark.

Results The benchmarks of the centralized learning regressors are R^2 values of 0.158 (random forest), 0.13 (neural network) and 0.13 (linear regression). For the neural network trained in the FL scenario, we achieve a slightly lower R^2 value of 0.114.

Runtime On a desktop machine with 16GB RAM and 8 CPU-cores, training the neural network takes 22 seconds per epoch in the centralized learning scenario and 6 seconds per epoch in the FL scenario. No more than 2GB RAM were used during the training. While 10 epochs suffice for the model to converge during centralized learning, FL needs at least 25 epochs.

Discussion The reasons behind the weak performance of the benchmark models (R^2 of 0.158 and 0.114) are not clear. Since we have conducted a thorough hyperparameter search, we conjecture that the problem is more likely to be rooted in the data set than in architectures of the two models. Either the use case we have modeled (estimating the radius of action based on LTE connection data) is hard to learn, or there are other issues with the data set which we were not able to identify. Nonetheless, we were able to reproduce almost the same performance in the FL training scenario.

3.3 Fine dust pollution

In the third simulation, we model a classification task in which the current fine dust pollution is inferred based on meteorological input data. More precisely, 48 consecutive hourly measurements are used to make a prediction for the current $PM_{2.5}$ pollution (the total weight of particles smaller than 2.5 µm in one m^3). The output of the predictor is one of the three classes *low*, *medium* or *high*. The threshold for each class are chosen in a way such that the samples of the whole data set are distributed evenly among the three classes.

Data set The data set we use is a multi-feature air quality and weather data set [Zha+17] which is publicly available online[10]. It consists of hourly measurements of 12 meteorological stations in Beijing, recorded over a time span of 4 years (2013–2017). Figure 1 depicts the locations of the 12 stations in Beijing. In total, more than 420 000 data records are included in the data set. Although some attributes are missing for some data records, most records have data for all of the 17 attributes listed in Table 5. An example plot for the two attributes PM2.5 and temperature is shown in Figure 2.

Data preprocessing To complete the missing data records, we use linear interpolation. We encode the wind direction by parsing the *wd* attribute into four binary attributes (one for each cardinal direction). All other features are scaled using a standard scaler implementation. We exclude the following pollution features from training, since we expect a high correlation with the target attribute PM2.5: PM10, SO2, NO2, CO and O3.

[10] https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data, accessed on January 17, 2023

Attribute	Description	Example value
year	year of recording	2018
month	month of recording	10
day	day of recording	9
hour	hour of recording	4
PM2.5	$PM_{2.5}$ concentration ($\mu g/m^3$)	24
PM10	PM_{10} concentration ($\mu g/m^3$)	43
SO2	SO_2 concentration ($\mu g/m^3$)	16
NO2	NO_2 concentration ($\mu g/m^3$)	57
CO	CO concentration ($\mu g/m^3$)	600
O3	O_3 concentration ($\mu g/m^3$)	112
TEMP	Temperature (degree Celcius)	13.3
PRES	Pressure (hPa)	1013.5
DEWP	dew point temperature (degree Celsius)	-12.1
RAIN	Precipitation (mm)	0.1
wd	Wind direction	NNW
WSPM	Wind speed (m/s)	3.5
station	Name of the air-quality monitoring site	Shunyi

Table 5: Attributes of the fine dust pollution data set [Zha+17].

Figure 1: Location of meteorological stations for simulation 3 (fine dust pollution) on a map of Bejing, China. 12 of the 13 stations are included in the public data set which we have used for our simulations. The dashed lines mark *regions* of the "Region-Learning" approach in [Hu+18]. Image source: [Hu+18].

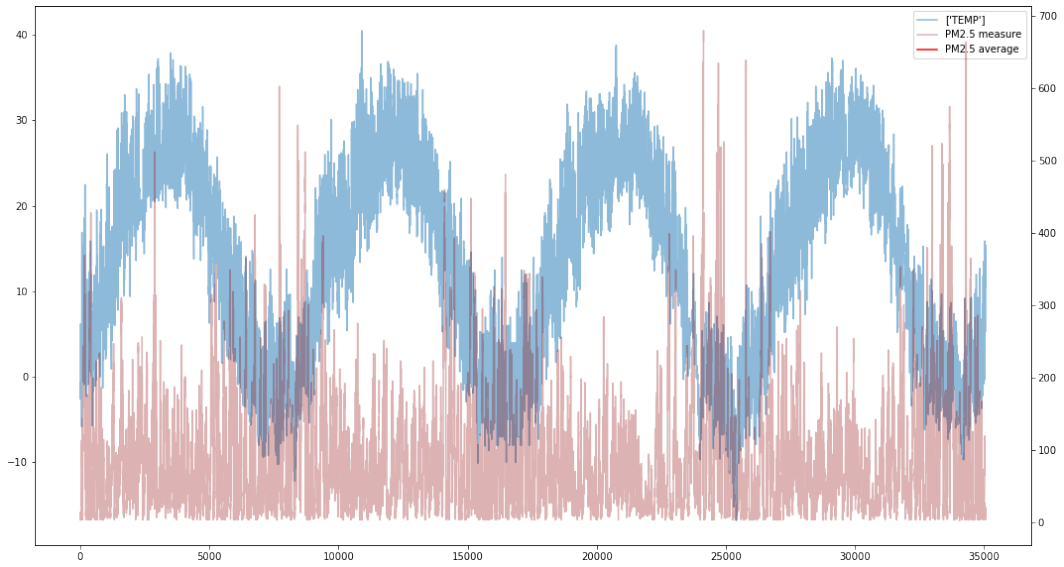

Figure 2: Example plot for the data of one station and the two features PM2.5 and TEMP ($PM_{2.5}$ pollution in red and temperature in blue). The four year time span is clearly visible by the temperature wave, due to hot summers and cold winters.

80% of the data are used as training data, the rest is used as test data. We consistently apply 5-fold cross validation, meaning that we repeat the training process five times and use a different part of the the data set as the test set each time.

Setup We simulate three different scenarios:

1. *Centralized learning (one model per station):* In the first scenario, one model is trained on the data set of each station separately.

2. *Centralized learning (global model over all stations):* The data sets of all stations are merged into a single data base which is used for training the global model.

3. *Federated learning:* In this scenario, every station is one FL client with the corresponding data measured by the station.

We use the same model architecture for all three scenarios, a neural network with five layers: A 10-neuron LSTM (long-short term memory) layer, a dropout layer with a dropout rate of 25%, a 5-neuron LSTM layer, another dropout layer with a dropout rate of 35% and a 3-neuron dense layer for the classification output.

We train for 20 epochs in the first scenario, 10 epochs in the second scenario and 160 epochs in third scenario (FL). In all scenarios, we use *CategoricalCrossEntropy* as the loss function. While we use the Adam optimizer with an automatic learning rate in both of the centralized learning scenarios, we employ the Stochastic Gradient Descent (SGD) optimizer in the FL scenario. On the server we use a learning rate of 1 for SGD, on the client we start with a learning rate of 0.1. The latter is divided by 10 every 64 rounds, such that at the end of the 160 epochs in the FL scenario, the client learning rate is at 0.001.

Results In the first scenario, the test accuracy of each station model is in the range of 69%–73%, with an average of 70.05% and a standard deviation of 0.015. The precision, recall and f1-score for all models are listed in Table 6. Similar to the accuracy, these metrics are also close to 70% for all models. As the exemplary confusion matrix in Table 7 shows, the most misclassified examples belong to the *medium* class.

For the second scenario, we apply 5-fold cross validation, yielding an average accuracy of 72.4% with a standard deviation of 0.005. The precision, recall and F1-score for each class are listed in Table 8. The confusion matrix in Table 9 shows that as in the previous scenario, the samples

Station	Precision	Recall	F1-score	Accuracy
Aotizhongxin	0.6999	0.7017	0.7004	0.7
Changping	0.715	0.7103	0.7119	0.7176
Dingling	0.7212	0.7089	0.7098	0.7255
Dongsi	0.701	0.7002	0.7005	0.7021
Guanyuan	0.6908	0.7017	0.6945	0.6909
Gucheng	0.6693	0.6781	0.6699	0.6702
Huairou	0.6833	0.6814	0.682	0.6859
Nongzhanguan	0.7122	0.7088	0.7102	0.7114
Shunyi	0.7183	0.7158	0.7167	0.7217
Tiantan	0.6976	0.6976	0.692	0.6965
Wanliu	0.696	0.6992	0.6959	0.6932
Wanshouxigong	0.689	0.6979	0.6878	0.692

Table 6: Different metrics of the resulting single station models (scenario 1).

	Prediction $PM_{2.5}$ low	**Prediction $PM_{2.5}$ medium**	**Prediction $PM_{2.5}$ high**
$PM_{2.5}$ actually low	10 048	2176	217
$PM_{2.5}$ actually medium	1872	6344	2353
$PM_{2.5}$ actually high	425	2701	8880

Table 7: Exemplary confusion matrix (in absolute values) for single station model Shunyi.

Class	Precision	Recall	F1-score
$PM_{2.5}$ low	0.82	0.78	0.80
$PM_{2.5}$ medium	0.61	0.61	0.61
$PM_{2.5}$ high	0.75	0.78	0.67

Table 8: Precision, recall and F1-score of an exemplary scenario 2 model: a centralized model over all available data.

	Pred. $PM_{2.5}$ low	**Pred. $PM_{2.5}$ medium**	**Pred. $PM_{2.5}$ high**
$PM_{2.5}$ actually low	114 450	23 712	5769
$PM_{2.5}$ actually medium	23 635	80 718	33 754
$PM_{2.5}$ actually high	1944	24 629	111 581

Table 9: Exemplary confusion matrix for one of the five models in the cross-validation training of the global model (scenario 2).

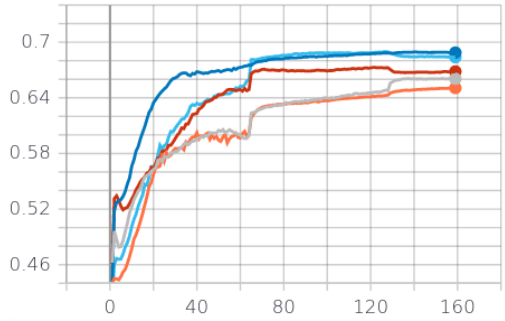

Figure 3: The test accuracy history during training of the 5-fold cross validation process in scenario 3 of the fine dust pollution simulation (FL).

labeled with *medium* are misclassified more often than the others. Both tables are exemplary and calculated on one of the five models of the 5-fold training process.

In the third scenario, the FL training, we achieve an average test accuracy of 67.0% with a standard deviation of 0.014 via cross validation. Precision, recall and F1-score of the resulting model are at 0.671, 0.670 and 0.669, each with a standard deviation of 0.013. The training history of the five cross validation models in this scenario is depicted in Figure 3. In a first attempt, we have *not* used the encoded wind direction and the time features (year, month, day) during the training process, yielding a significantly lower accuracy of 63.5% with a standard deviation of 0.01.

Runtime We run this simulation on a CPU server with 56 Intel® Xeon® E5-2660 CPUs with a clock speed of 2.0 GHz and 14 cores each. These are the runtimes for our three scenarios:

1. *Centralized learning (one model per station)*: 10 seconds per epoch, 3 minutes for 20 epochs for each of the 12 station models.
2. *Centralized learning (global model over all stations)*: 104 seconds per epoch, 17 minutes for 10 epochs.
3. *Federated learning*: 22 seconds per epoch, 58 minutes for 160 epochs.

Discussion Since the homo- or heterogeneity of data sets can have a major influence on the performance of FL [Kai+19; Pra+21], we want to point out the homogeneity of the data set in this case: The data collection process has been the same for all meteorological stations and they are geographically close to each other. The similar numbers in Table 6 reflect this homogeneity.

Furthermore, we were able to observe the importance of choosing the right test set in this simulation. Initially, we chose the last 20% of each station data set as the test set, which has resulted in a significantly higher test than training accuracy. We mitigate this bias by applying 5-fold cross validation.

Apart from preprocessing the data, another time consuming process was tuning the hyperparameters of the FL training (scenario 3). Different from local training with TensorFlow, TensorFlow Federated requires the specification of a client and a server learning rate. While the server learning rate can be set to a value around 1 and has less influence on the success of the training process, the client learning rate and batch size have a large impact and need to be optimized for good results. Since we were not able to find practical tools for FL hyperparameter optimization (see Section 2.2), it was necessary to manually trigger different trial runs with varying hyperparameters.

Comparison with literature Hu et al. have also worked with this data set, introducing "Federated Region-Learning" [Hu+18]. The authors compare the results of their novel strategy to a baseline of centralized learning (our scenario 2) and standard FL (scenario 3). Although according to the authors, Region-Learning outperforms the other two approaches (by 5 percentage points compared to standard FL), it is important to note that Region-Learning has another goal than the FL we analyze in this paper. Different from standard FL, Region-Learning is a form

of model personalization (see Section 2.1), where the result consists of multiple models – in this case, one model per predefined region of the city of Beijing, capturing the area of multiple meteorological stations. For the grouping of regions, see Figure 1.

Furthermore, Hu et al. have not provided sufficient information to properly retrace their experiments. Especially the number of classes for $PM_{2.5}$ classification and information on the features used for training the classifiers are missing, so that their results are hard to compare to ours. For example, setting the number of classes to 2 and using all features of the data set (including the other pollution attributes PM10, SO2 etc.) would significantly ease the estimation task. Also, we have no information on the choice of test sets or whether cross validation was applied in the work of Hu et al. Two more hints in the paper by Hu et al. suggest that they have used a slightly different data set than we have: The data set they describe includes "more than 100 000" data records from 13 meteorological stations in Beijing, while our data set contains more than 420 000 records from 12 stations.

Accuracy comparison	**Centralized (S1)**	**Centralized (S2)**	**FL (S3)**	**Region-Learning**
Hu et al. [Hu+18]	–	80%	76%	81%
Our study	70%	72%	67%	–

Table 10: Comparison of results of the Hu et al. paper [Hu+18] and our study. The performance differences might be explained by significant differences in the experimental setup, on which Hu et al. have not provided enough information. S1, S2, S3 = scenario 1, 2 and 3.

With these major limitations of comparability stated, we present the summarized results of Hu et al. in Table 10. One consistency across both their and our work is the accuracy drop from centralized learning (scenario 2) to FL, with 4 percentage points in [Hu+18] and 5 percentage points in our work.

4 Key Observations and Future Work

Our simulations lead us to the following key observations:

- Training data must be available in sufficient quantities, otherwise the training process cannot be successful (see simulation 1).
- While the random forest regressor has outperformed the other centralized learning benchmarks in simulations 1 and 2, we were not able to find a practical federated random forest implementation. This could be an interesting research direction.
- Choosing the right hyperparameters is crucial for any machine learning model. Since automatic HPO is still an open problem for FL algorithms, (manually) finding the right settings can be a time consuming process. Developing a suitable framework for automated HPO for FL would be important future work.
- In our second simulation (mobile radio data), we did not have access to the data, just like in a real-world scenario. This means both HPO and technical debugging needed to be performed remotely, without access to the data. We believe that in scenarios with multiple data holders and possibly heterogeneous data sets, these tasks will be significantly harder.
- All FL simulations were performed on the machine which also had access to the complete data set (simulation 2 was performed on a machine of the company umlaut). In a real-world application, where each client runs on a distinct machine, other settings and other frameworks might be more practical than TensorFlow Federated.
- FL is outperformed by centralized learning on the same data set in all cases. While this is not surprising (since the FL model has been exposed to the complete data set only indirectly), we want to stress that without FL, many machine learning scenarios might not be possible due to privacy concerns, trade secrets, or similar reasons.

- Last but not least, we want to emphasize that FL is no one-size-fits-all solution for machine learning privacy issues. Although no private data needs to be transmitted, sensitive information might be leaked by the model updates which are exchanged by the parties. This leakage can be leveraged by privacy attacks like training data extraction [ZLH19], model inversion [HAP17], membership inference [NSH19], and property inference [Mel+19]. The applicability of these attacks highly depends on the use case in which a model is trained, the type of model, the attacker model and other factors. Hence, analyzing these risks is a crucial step before an application is rolled out in practice. Possible mitigation strategies might be found in privacy-preserving machine learning (PPML) technologies.

5 Implications for Official Statistics

In the simulations of this study, we have demonstrated how FL can be used in use cases similar to official statistics. In this section, we continue the discussion by elaborating on the concrete implications of the simulations on the applicability of FL in official statistics.

In general, if a task can be successfully implemented in a centralized ML, we expect FL models to reach a similar performance to centralized models – as long as there is enough training data available at the participants and the appropriate hyperparameters are chosen for the training. Choosing the right hyperparameters will take time and trial runs, both with and without an automated framework. Hence, it is important for the FL clients to be reachable even prior to the actual model training.

Also, it is crucial to note that the FL training process itself takes more training rounds to converge compared to centralized ML training. This is the case in both simulations 2 and 3. We suspect that this is due to the averaging process on the server: Since some information is lost by averaging, the training data's influence on the model during training is less direct than during centralized ML training and thus, training takes more time.

As elaborated in the previous Section 4, FL does not prevent all kinds of privacy attacks – in contrast, more and more privacy attacks specifically tailored to FL training scenarios are developed [Mot+21]. Since it is impossible to prevent *all* attacks in a specific setting, we believe that transparency is the key to get potential partners on board to participate in an FL training. By explaining the technology, listing security measures in place and elaborating on risks which cannot be mitigated, an honest partnership can be established. As mentioned before, an individual assessment of security and privacy risks of each use case is important, since they may vary significantly (depending on the nature of the data, the technology used, the model architecture and other factors).

FL is evolving rapidly and both industry and research will continue to improve the field in the coming years. The performance and efficiency of practical FL frameworks is expected to be further optimized. Similarly, we expect the development of more usable PPML algorithms including the ones based on Secure Multi-Party Computation (SMPC) and Homomorphic Encryption (HE) – allowing for provably secure collaborative machine learning. Although such PPML methods have been proposed and frameworks exist, their performance today is often far from acceptable for many practical applications. With more standardization and simpler, respectively more efficient, applications, FL will become even more beneficial to official statistics. Thus, it should be useful to track the coming progress of industry and resarch in this area.

6 Conclusion

In conclusion, FL has a lot of potential for official statistics. When data holders have doubts to share their data but are willing to contribute to a decentralized training process, FL can help to overcome these issues. As we have shown in our simulations, FL can reach a similar performance to centralized learning algorithms, provided there is enough training data available at each party. Before FL can be transferred into the practice of official statistics, we see the following topics which should be investigated in more detail: developing a practical frameworks for federated hyperparameter optimization, finding the right framework for a real-world FL process, and the analysis of privacy risks and mitigation strategies for the concrete scenario in question. Especially the latter is an important stepping stone in ensuring the privacy protection of involved parties, even if FL itself is often already categorized as a privacy-preserving technology. Just as in

countless other domains, we expect FL to become a relevant technology for official statistics in the near future.

References

[Bre01] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[DLJ20] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. "Federated Bayesian optimization via Thompson sampling". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9687–9699.

[HAP17] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. "Deep models under the GAN: information leakage from collaborative deep learning". In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017, pp. 603–618.

[Hu+18] Binxuan Hu et al. "Federated region-learning: An edge computing based framework for urban environment sensing". In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, pp. 1–7.

[Kai+19] Peter Kairouz et al. *Advances and Open Problems in Federated Learning*. 2019. DOI: 10.48550/ARXIV.1912.04977. URL: https://arxiv.org/abs/1912.04977.

[KH18] Antti Koskela and Antti Honkela. "Learning rate adaptation for federated and differentially private learning". In: *arXiv preprint arXiv:1809.03832* (2018).

[Kho+21] Mikhail Khodak et al. "Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 19184–19197.

[McM+17] Brendan McMahan et al. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[Mel+19] Luca Melis et al. "Exploiting unintended feature leakage in collaborative learning". In: *2019 IEEE symposium on security and privacy (SP)*. IEEE. 2019, pp. 691–706.

[Mos19] Hesham Mostafa. "Robust federated learning through representation matching and adaptive hyper-parameters". In: *arXiv preprint arXiv:1912.13075* (2019).

[Mot+21] Viraaji Mothukuri et al. "A survey on security and privacy of federated learning". In: *Future Generation Computer Systems* 115 (2021), pp. 619–640.

[NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning". In: *2019 IEEE symposium on security and privacy (SP)*. IEEE. 2019, pp. 739–753.

[Pra+21] Prayitno et al. "A Systematic Review of Federated Learning in the Healthcare Area: From the Perspective of Data Properties and Applications". In: *Applied Sciences* 11.23 (2021). ISSN: 2076-3417. DOI: 10.3390/app112311191.

[VML21] Saeed Vahidian, Mahdi Morafah, and Bill Lin. "Personalized Federated Learning by Structured and Unstructured Pruning under Data Heterogeneity". In: (2021). URL: https://arxiv.org/abs/2105.00562.

[YS20] Li Yang and Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice". In: *Neurocomputing* 415 (2020), pp. 295–316.

[Zha+17] Shuyi Zhang et al. "Cautionary tales on air-quality improvement in Beijing". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2205 (2017), p. 20170457.

[Zho+21] Yi Zhou et al. "FLoRA: Single-shot Hyper-parameter Optimization for Federated Learning". In: *arXiv preprint arXiv:2112.08524* (2021).

[ZLH19] Ligeng Zhu, Zhijian Liu, and Song Han. "Deep leakage from gradients". In: *Advances in neural information processing systems* 32 (2019).