AdaFed: Performance-based Adaptive Federated Learning

ALESSANDRO GIUSEPPI, University of Rome "La Sapienza", Italy
LUCREZIA DELLA TORRE, University of Rome "La Sapienza", Italy
DANILO MENEGATTI, University of Rome "La Sapienza", Italy
FRANCESCO DELLI PRISCOLI, University of Rome "La Sapienza", Italy
ANTONIO PIETRABISSA, University of Rome "La Sapienza", Italy
CECILIA POLI, Istituto Superiore di Sanità, Italy

Federated Learning is a distributed and privacy-preserving machine learning technique that allows local clients to learn a model without sharing their own data by coordinating with a global server. In this work, we present the Adaptive Federated Learning (AdaFed) algorithm, which aims at improving the training performance of deep neural networks in Federated Learning settings by: (i) dynamically weighting the local models in the model averaging procedure; (ii) by adapting the loss function used by the federation at every communication round. We discuss the specialisation of AdaFed for both classification and regression tasks, providing several validation examples. Due to its adaptive design, the AdaFed algorithm showed a robust behaviour against unbalanced data distributions and adversarial clients.

CCS Concepts: • **Computer systems organization → Distributed architectures**; • **Computing methodologies** → *Distributed algorithms*; **Distributed artificial intelligence**; *Multi-agent systems*; **Intelligent agents**; **Cooperation and coordination**.

Additional Key Words and Phrases: Federated Learning, Deep Neural Networks, Distributed Learning Systems

ACM Reference Format:
Alessandro Giuseppi, Lucrezia Della Torre, Danilo Menegatti, Francesco Delli Priscoli, Antonio Pietrabissa, and Cecilia Poli. 2022. AdaFed: Performance-based Adaptive Federated Learning. *ACM Trans. Graph.* 37, 4, Article 111 (August 2022), 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Federated Learning (FL) is a distributed learning solution to address Machine Learning (ML) problems without the need of collecting the available data in a single data center. FL finds application in scenarios in which the data are distributed over a multitude sources that, for privacy or communication constraints, cannot share it among them or with a centralised entity. The need of analysing data *locally* becomes of crucial importance when dealing with personal data (e.g., collected by smartphones) or sensitive information (e.g., regarding the customers of a company).

FL is then a technology enabler to analyse data in heavily regulated fields such as healthcare [1], connecting the fragmented data sources while preserving privacy [2]. Several studies in the literature underline the potential of deep learning methodologies in identifying complex models in the medical field, with numerous applications, e.g., in radiology, pathology, genomics [3]. The identification and availability of sufficiently large and diverse data sets, necessary for the training of the model, is a considerable challenge in medicine and can rarely be met by single institutions. On the other hand, multi-institutional collaborations based on centrally-shared patient data face difficulties in terms of data privacy and ownership, especially in international collaborations, and are also not suitable for cases where there are the number of institutions is large. As a result, the knowledge generated around the world remains distributed across multiple institutions, raising the need for seeking alternative approaches: due to its privacy preserving characteristics, FL represents a collaborative learning approach enabling multi-institutional collaborative learning tasks for intelligent healthcare applications. In other words, given a network of hospitals, instead of training a neural network on the clinical data of all the hospitals gathered in a single server, with FL it is sufficient to share the knowledge acquired by the single hospital model to improve the model of the entire federation. While the first solution has privacy related issues, the latter can be made GDPR compliant.

In FL, the server's model, is updated at every *communication round* by averaging the models of the federated clients, trained on their locally available data. In this work, we propose the ADAptive Federated Learning algorithm, AdaFed, a two-step procedure to improve both the model averaging and the local training processes by i) dynamically weighting the client models contributions to the federation based on their performance, and ii) adapting the federated loss function depending the global model performance at each communication round.

For the sake of presentation clarity, the proposed solution is designed to extend the formulation and results of original FL algorithm Federated Averaging (FedAvg) [4], but the concepts behind the proposed innovations are independent of the specific implementation and may be seamlessly translated to other algorithms such as FedProx [5]. The evaluation of of AdaFed in terms of versatility, performance improvements and capability of addressing new challenging scenarios is illustrated by comparison with FedAvg on different tasks involving several different data-sets.

Authors' addresses: Alessandro Giuseppi, giuseppi@diag.uniroma1.it, University of Rome "La Sapienza", Via Ariosto 25, Rome, Italy, 00185; Lucrezia Della Torre, dellatorre@diag.uniroma1.it, University of Rome "La Sapienza", Via Ariosto 25, Rome, Italy, 00185; Danilo Menegatti, menegatti@diag.uniroma1.it, University of Rome "La Sapienza", Via Ariosto 25, Rome, Italy, 00185; Francesco Delli Priscoli, dellipriscoli@diag.uniroma1.it, University of Rome "La Sapienza", Via Ariosto 25, Rome, Italy, 00185; Antonio Pietrabissa, pietrabissa@diag.uniroma1.it, University of Rome "La Sapienza", Via Ariosto 25, Rome, Italy, 00185; Cecilia Poli, cecilia.poli@iss.it, Istituto Superiore di Sanità, Viale Regina Elena 299, Rome, Italy, 00161.

0730-0301/2022/8-ART111 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The remainder of this paper is organized as follows: In Section 2 we present an overview of related work. In Section 3 we present our framework, AdaFed, specialising it for classification and regression tasks, in Section 3.1 and 3.2, and discussing its limitations in Section 3.3. Some validation experiments are presented in Section 4, while Section 5 draws the conclusions and highlights future work.

2 RELATED WORKS AND MAIN CONTRIBUTIONS

FL was introduced in 2016 by the authors of [6] and, in its original formulation [4; 6], FL was proposed specifically to address the collaboration among a group of smartphones and consisted in an iterative model-averaging procedure. According to such procedure, local model updates, independently computed by the smartphones, were gathered and averaged by a centralised server that then propagated the updated *global* model in the network, as depicted in Figure 1.

As discussed in the recent surveys [7–9], since its introduction FL was extended to different architectures and uses cases, with several works focusing on enhancing its privacy preserving and security related characteristics [10–13] to prevent direct or indirect data leakage [14], and on reducing the communication cost associated to the distributed training [15–17]. FL found application in several domains, spacing from smartphones/Internet of Things tasks such as Natural Language Processing (NLP) [18–20], image analysis [21; 22] and distributed sensing and computing [8; 23], to scenarios in which organisations and institutions cooperate to obtain better models to analyse complex and highly confidential data, as typical in the healthcare domain [24–26]. Contrary to federated database systems, in which data can be distributed freely by a central entity, the typical scenario for a FL application is characterised by the need of analysing data partitioned *as given*, implying that FL algorithms need to be able to cope with data that are:

- non-IID and imbalanced, as the geographical dislocation of the federated organisations or data sources may significantly affect the data distribution and collecting procedures;
- extremely distributed, as in scenarios in which the federated entities are smart connected devices their number is likely to be orders of magnitude more than the quantity of their individual data samples.

This work explores the concept of associating to each client a different weight in the model averaging procedure depending on their contribution to the federation. Several other works explore this research direction, as the FOCUS algorithm [27] where the authors design a procedure to determine the weighting factors based on a credibility score assigned to each client. Such credibility score aims at minimizing the sensitivity of the federation model to a possible disparity in the labeling quality of the clients' datasets, hence addressing two of the main implications of considering distributed data sources: the different collecting procedures and human factors. The FOCUS algorithm relies on the idea that the combination of the performance of the global model on the clients' datasets and the performance of the clients' models on the server dataset (which is assumed to be correctly labeled) provides an indicator of the quality of the clients' labelling. The framework proposed by AdaFed is not specifically designed for providing robustness to labelling quality disparity, and focuses more on the evaluation of the clients' model instead of inferring the quality of their data. Furthermore, FOCUS determines its credibility scores under the assumption that a correctly labeled client dataset is IID with the one available to the server (Theorem 1 in [27]), whereas, as we will show in the simulations, AdaFed does not make any assumption on the clients' data. In fact, AdaFed will be shown to be an enabler approach to allow FL algorithms to cope with scenarios in which data are distributed in extremely unbalanced ways and even in the presence of malicious/compromised clients. A similar approach is followed by [28], where a federated way to estimate the Shapey Value (SV) [29], that captures the value of the clients' data for the federation, is proposed. The federated estimation of the SV allows several useful properties, as the detection of poor/noisy labelling, malicious clients and communication minimisation. The main idea behind the algorithm in [28] is to sort the clients according to their SV (i.e., their contribution to the federation) and then let only the most contributing ones participate to the model averaging. AdaFed, on the other hand, performs a weighted model averaging, where the weights given to all the clients are dynamically adapted based on their model performance, reducing the impact of - or even excluding - the low performing/malicious ones. We mention that a federated version of SV was also explored in [30], where a similar metric is developed to allow the distribution of revenue/profits to the clients depending on their contribution to the federation. In this sense, a future work may explore the combination of federated SV estimation with the adaptive model averaging included in AdaFed by utilising the SV values as clients' weights. One of the most promising contribution in the FL field was made by the authors of [5] with the so-called FedProx algorithm. FedProx, similarly to AdaFed, builds on top of FedAvg formulation to cope with some of its limitations, and in particular proposes two improvements to deal with systems' heterogeneity (i.e., significant differences in the optimisation capabilities of the federation clients) and statistical heterogeneity (i.e., significant non-IID nature of the data distribution). FedProx archives its properties by proposing an extended loss function that includes a so-called proximal term that limits the impact of different local training settings (e.g., epochs number, optimiser settings, ...) and non-IID local data distribution on the clients' models. In principle, AdaFed may be deployed on a federation that utilises a loss function with the structure proposed by FedProx, but the impact of the adaptive features of AdaFed on the convergence properties of FedProx should be explored in detail in a future work.

Another contribution of this work is in the direction of combining FL with the recent research trend of *adaptive loss functions* [31–34], that showed promising results in problems characterised by complex loss functions. An example is multi-objective learning, where the models are required to optimize multiple loss functions at the same time seeking a Pareto-like optimality [32]. Complex loss functions are also found in computer vision [31; 34; 35] where they are commonly used in solutions such as You-Only-Look-Once (YOLO) [35] or Single-Image-Super-Resolution (SISR) [36]. The fundamental idea at the basis of adaptive loss approaches is that the loss function itself evolves during training depending on the recent intra-training performances of the model. FL, due to its iterative nature, provides

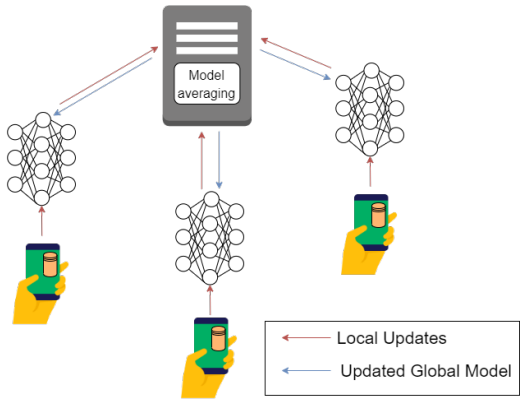

Fig. 1. Graphical representation of the architecture for "horizontal Federated Learning" [9] employed by the FedAvg algorithm presented in [4].

an ideal setting for such an updating solution, as the models' loss function can be adapted at every communication round.

A summary of the main contributions of the paper follows:

(1) This work introduces a new adaptive federated learning algorithm, AdaFed, in which two mechanisms drive and speed up the learning process
   (a) by dynamically weighting the contributions of the clients' models in the server's model based on their performance (over a representative test set), and
   (b) by modifying the local training with an adaptive update of the loss functions of the clients' models at each communication round, depending on the performance achieved by the server's model on its test set.
(2) It also illustrates the versatility of AdaFed in learning different models, including convolutional neural networks for both classification and regressions tasks, also in the transfer learning setting.
(3) It discusses through simulations the performance improvement that the proposed method is able to achieve in various scenarios with respect to one of the most common baseline FL algorithm, FedAvg [4].

3 ADAFED: ADAPTIVE FEDERATED LEARNING

The proposed solution for FL draws inspiration from the well known ensemble learning algorithm ADAptive Boosting (AdaBoost) [37]. In AdaBoost, a weak learner is trained at each epoch (commonly in the form of a *decision stump* for classification tasks) and the data samples are associated with a weighting factor proportional to their contribution to the loss attained by the learner. Consequently, at the next epoch, the new learner will focus more on the miss-evaluated data.

The AdaFed algorithm pseudo-code is written in the Algorithm 1 table. Differently from AdaBoost solutions, AdaFed does not involve ensemble learners and, instead, aims at making the model averaging procedure at the backbone of FL an adaptive process to improve the performance of the global model learned by the federation. This result is pursued by the following two step-procedure, that

Algorithm 1 AdaFed: Performance-Based Adaptive Federated Learning algorithm

1: **SERVER'S UPDATE:**
2: **for** each round $t = 1, 2, ..., R$ **do**
3: ClientsUpdate
4: **for** each client $i = 1, 2, ..., K$ **do**
5: receive the client's model w_i
6: evaluate w_i on the server test set
7: use the evaluation to determine the weight p_i
8: **end for**
9: update the server's model $w_S \leftarrow \frac{\sum_{i=1}^{nc} w_i p_i}{\sum_{i=1}^{nc} p_i}$
10: evaluate its performance p_S on the server test set
11: adapt the loss function l depending on p_S
12: propagate w_S and l to the clients
13: **end for**

14: **ClientsUpdate:**
15: **for** each client $i = 1, 2, ..., K$ **do**
16: $w_i \leftarrow w_S$
17: **for** each local epoch $j = 1, 2, ..., E$ **do**
18: **for** each mini-batch b of size B **do**
19: $w_i \leftarrow w_i - \eta \nabla l(w, b)$
20: **end for**
21: **end for**
22: return w_i to server
23: **end for**

will be specialised for classification problems in section 3.1 and for regression problems in section 3.2:

(1) **Weighted Model Average**: during the server update (lines 4-9 of Algorithm 1), the collected models are evaluated on a common test set and are weighted according to their performance for averaging. The performance metric can in principle be any quantity that captures how well the model performs for the given task (e.g., accuracy for classification tasks), or may even be set to a quantity such as the SV to evaluate the client's contribution to the federation.
(2) **Adaptive Loss**: the server propagates to the federation both the updated model and a new loss function, which was adapted to the performance of the server own model on a dedicated test set according to a use-case dependant metric (e.g., update class weights for classification tasks depending on recall) (lines 10-12 of Algorithm 1).

We formulated the Algorithm 1 similarly to FedAvg, as this allows for a clearer presentation of its innovations. We mention that privacy preserving features, communication minimization policies (e.g., random selection of clients for each averaging) and other enhancements (e.g., the heterogeneity reduction approa-ches of FedProx) may be included in the AdaFed formulation as long as they are compatible with the standard FL algorithm structure of iterative local updates and centralised model averaging.

As in [27], AdaFed aims at evaluating the contribution level of the various clients to determine their weights in the model averaging,

and to do so AdaFed assumes the availability of a representative server test set on which its is possible to evaluate the performance of the various clients' models for the given task, without any significant increase in the training complexity.

The combination of a dynamic weighted model averaging procedure with an adaptive loss function allows the federation to: i) give more weight to better performing clients, while reducing or preventing the negative influence of malicious/noisy ones (by lowering or setting to 0 their weights) and ii) give more attention to data samples needed to improve the model performance, as in the case of the most rare/harder to discern labels in classification tasks.

As mentioned, in the remainder of the paper, we will neglect aspects linked to communication efficiency and privacy preserving solution, as they are already well documented in the works [4; 7; 9; 17] and are beyond the scope of the present work, which instead focuses on the model averaging and adaptive loss aspects.

3.1 Application: AdaFed for Classification Tasks

Multi-class classification is one of the most common ML task examples. A typical choice for the loss function utilised in this setting is the categorical cross-entropy:

$$l(X,Y) = -\frac{1}{M}\sum_{c=1}^{C}\sum_{m=1}^{M} y_m^c log(\hat{y}_m^c), \tag{1}$$

where $x_m \in X$ and $y_m \in Y$ are the m-th data sample and label, respectively, in the dataset (X,Y), M and C are respectively the number of data samples and classes, y_m^c and $\hat{y}_m^c$ denote the c-th component of the vectors y_m and $\hat{y}_m$ and are respectively the true and predicted labels for the sample m regarding class c. Note that $\hat{y}_m^c$ is typically produced, for single-label problems, by a deep neural network with a *softmax* output activation function and can be interpreted as the probability of correctness for the given label.

As it is, the categorical cross-entropy does not compensate for imbalanced class distributions, that in our reference scenarios are likely to characterise the datasets available to the various clients. A typical solution to improve the learning process in this kind of scenarios is to utilise a weighted categorical cross-entropy of the form

$$l(X,Y) = -\frac{1}{M}\sum_{c=1}^{C}\sum_{m=1}^{M} \kappa^c y_m^c log(\hat{y}_m^c), \tag{2}$$

where κ^c is a class-dependent weight. In [38] the κ^c were set as proportional to the inverse of the number of data samples available for each class, while in Focal Loss [39] a complex weight is associated to each class based on its classification difficulty. To the best of the authors knowledge, the weighting approaches available in literature either introduce static, rule based, coefficients or consider the weights as hyper parameters, which is in contrast to what was recently proposed for regression in [31], where the loss is dynamically modified during training.

Inspired by [31; 38; 39], we choose the class dependent weight κ^c to be inversely proportional to the performance p_S obtained by the server model on the server test set with respect to its corresponding class, expressed by means of its F_1^c-score, e.g. $\kappa^c = 1/(F_1^c + \epsilon)$, where $\epsilon < 1$ is a design parameter to limit κ^c. Note that this choice is however arbitrary as other metrics to evaluate p_S can be chosen, as shown in Section 4. The rationale behind this *adaptive loss logic* is to encourage the clients to focus, during their training phase, on classes which are misclassified by the global model.

Referring to Algorithm 1, this choice translates in having $p_S = \{F_1^c, \text{ for } c = 1, ..., C\}$ (line 10) and updating the loss function l with the new weights κ^c derived from the F_1^c-scores (line 11).

The same idea is used in the Weighted Model Average step. Note that, in principle, the weight p_i of the i-th client can be computed via the performance of its model over the server test set with a different metric than the one used for the Adaptive Loss step, so instead of the F1 score one may utilise for example the model accuracy or the diagnostic odds ratio, depending on the specific use case

3.2 Application 2: AdaFed for Regression Tasks

The same principles discussed in section 3.1 can be directly adapted to regression tasks. For example, as it will be discussed in the simulation section, for model averaging in regression problems it may be suitable to select an application-dependent metric not directly related to the loss. In our example (simulation 3), where the objective is to estimate the number of cells in a given microscope photo, we will consider the mean absolute percentage error on the counting of cells, whereas the models' loss will be related to the mean squared error between a target image and the image generated by the deep neural network. Furthermore, it is in principle possible to consider a loss function as the one presented in [31] and adapt or tune its hyper parameters at each communication round, depending on the server's model performance on its dataset.

3.3 Limitations

The main limitation of the proposed framework is the availability of a server test set that is qualitatively and quantitatively representative of the learning task. While its numerosity is not required to be particularly high, on certain scenarios it may not be possible to assume the existence of such a set. A possible solution would be to share the local models in the network and collect their performance on the distributed test set available to the various clients. By properly averaging the collected performance it is then possible to determine the various performance weights p_i and obtain the updated global model. This procedure may be iterated again for adapting the loss function l. Note that this approach, besides adding a significant communication overhead, may be sensitive to adversarial attacks as a level of trust in the evaluations of the clients is required if no adversary detection strategy is implemented. An alternative approach could be imposing to every member of the federation to share a small portion of their data, selected randomly, so that the resulting dataset may be representative for all the data available to the federation.

4 EXPERIMENTS

In this section, AdaFed is tested and compared to FedAvg in different scenarios. The focus of the experiments is on the performance of the server's model, neglecting communication efficiency arguments for the sake of the results clarity. Nevertheless, in principle AdaFed

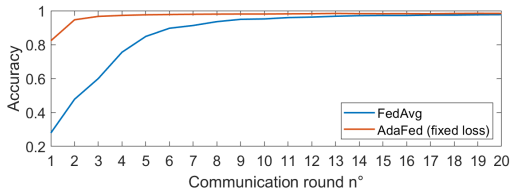

Fig. 2. Simulation 1.A (MNIST): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

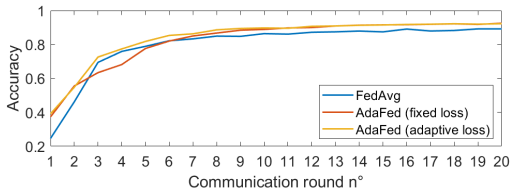

Fig. 3. Simulation 1.B (MNIST): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

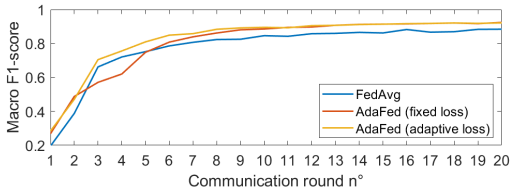

Fig. 4. Simulation 1.B (MNIST): Evolution of the server model macro F1-score over communication rounds, evaluated on the separated test set of the server.

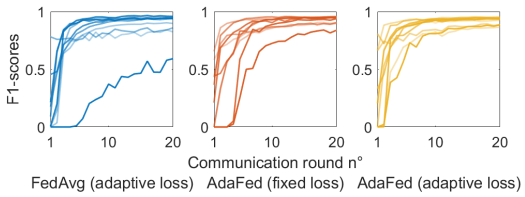

Fig. 5. Simulation 1.B (MNIST): Per-class evolution of the server model F1-scores over communication rounds, evaluated on the separated test set of the server.

can be seamlessly modified to accommodate for the differential privacy and communication efficiency improvements proposed in literature to extend FedAvg. For these reasons, in the following, we consider that all the clients send their model to the server at each communication round.

4.1 Simulation 1 - MNIST and OARF Classification Tasks

In the first simulation we train our models on the MNIST [40] dataset to better evaluate the effect of each of the proposed enhancements. The MNIST dataset consists in a set of 60k+10k labeled images of handwritten digits (from 0 to 9), and represents one of the most common baselines for classification tasks. Simulation 1.A will discuss the benefit of the Weighted Model Average step, Simulation 1.B will detail the effects of the Adaptive Loss step, and Simulation 1.C will validate the robustness of the federation against adversarial clients. For the MNIST simulations, we set the parameters of the Algorithm 1 (Table 1) as $E = 5$ (number of epochs in the clients' update), $B = 100$ (batch size) and $R = 20$ (number of communication rounds). The server test set is composed by the whole MNIST test set, whereas clients' data are distributed as described in the following sections. Simualtion 1.D considers a binary sentiment analysis classification task from the benchmark suite OARF [41], further highlighting the capabilities of AdaFed to cope with poorly distributed data. The parameters were set as $E = 3$, $B = 100$ and $R = 5$.

4.1.1 Simulation 1.A (MNIST) - Validation of Weighted Model Averaging. In this experiment, we consider a federation constituted by $K = 6$ clients. The first five clients have access to 5500 samples from two classes only, with no overlapping, except for classes 0 and 5 which have 100 and 200 samples only, while the sixth one has 500 samples from each classes. The model of each client is a simple deep convolutional neural network from the official documentation of Keras [42], composed by two 2-D convolutional layers followed by two dense layers. The performance weight p_i of the i-th client for AdaFed is computed as its accuracy times the number of its available data samples, i.e., $p_i = \text{accuracy}_i \times \#\text{training-data}_i$.

Figure 2 shows that AdaFed starts with a higher accuracy from the very first communication round, thus attaining a faster convergence rate compared to FedAvg. The reason behind this behaviour is the presence of a client whose dataset is IID with respect to the server test set; despite its limited amount of data, the sixth agent exhibits from the start a better accuracy compared to the other clients which have visibility only on two different classes, meaning that its performance weight its significantly higher than the others in the model averaging procedure. Starting the next round with a better baseline model, AdaFed achieves, from the second round, an accuracy level achieved by FedAvg only after twelve rounds.

4.1.2 Simulation 1.B (MNIST) - Validation of adaptive loss function. In this experiment, we want to evaluate the impact of adapting the clients loss function after each communication round. The scenario is similar to the one of Simulation 1.A, with data now randomly distributed in a less uniform way, as shown in Table 1.

The weighted categorical cross-entropy loss 2 of the clients is updated after each communication round according to the server model F1 scores for the various classes. In particular, the class weights κ^c are set to be equal to $1/(\text{F}_1^c + 0.1)$. The logic behind this choice is to increase the contribution to the loss for the classes that are more commonly miss-classified (i.e., $\text{F}_1^c \to 0$) by a factor of 10, whereas the weights of the better recognised classes (i.e., $\text{F}_1^c \to 1$) are slightly lowered.

Figure 3 reports the accuracy evolution of FedAvg compared with two different AdaFed federations, one implementing both the weighted averaging and the adaptive loss procedure described above, and another that only implements the weighted model averaging. Note that, in this simulation and in the following ones, for simplicity we set for AdaFed $p_i = \text{accuracy}_i$.

Both the AdaFed implementations are able to offset the FedAvg one by about 3%; moreover the adaptive loss one attains convergence faster. The reason for this different behaviour can be seen in Figure 4, which reports the Macro F_1-scores (i.e. the arithmetic mean of the F_1^c-scores of all the classes). The adaptive loss implementation of AdaFed outperforms the others, meaning that it is able to better discern classes for which limited samples are available. Figure 5 details the F_1^c-scores for all the ten classes of the datasets, and it is clear that AdaFed performs better in this sense, with the adaptive loss significantly contributing in having the scores of the classes evolve more uniformly and converge more rapidly.

4.1.3 Simulation 1.C (MNIST) - Robustness to adversarial actors. Considering the same data distribution of Simulation 1.B, we show in this experiment the resiliency of Ada-Fed to adversarial clients that try to lower the federation performances. To this end, we now add two additional clients 7 and 8 (i.e., $K = 8$) with the same data distribution of clients 3 and 4 but with incorrect labeling for respectively 50% and 100% of their samples. Furthermore, the malicious clients also discard the federated model they receive from the server and update their model only on the basis of their own data.

Figures 6 and 7 show that AdaFed (implemented with the adaptive loss and weighted averaging as in the previous simulation) is practically unaffected by the presence of the new malicious clients (the difference with respect to the previous simulation is about 0.01% in accuracy and 0.015 in the in the macro F_1 score), while on the contrary FedAvg shows a significant drop in both accuracy (about 2%) and in the macro F_1-score (about 0.1). We note that, in more complex cases, a different strategy may be followed for AdaFed to select the weights p_i (e.g. a quadratic or cubic function of the accuracy) to further decrease or annihilate the weights of bad performing (and hence potentially adversarial) clients, as will be demonstrated in the following simulation.

4.1.4 Simulation 1.D (OARF) - Robustness to unfavourable data distributions and/or adversarial actors. The results obtained for Simulation 1.D are applicable also in scenarios in which the data are distributed in an extremely unfavorable way. In this simulation, we consider the sentiment analysis dataset from the benchmark suite Open Application Repository for Federated Learning (OARF) [41], consisting in 50k entries from the Amazon Movie Review dataset proposed in [43] for a binary classification task. We assume the data to be partitioned among $K = 5$ clients, as follows: 1 client only has access to $\sim$ 5k samples from the negative class (i.e., negative review), 2 clients have access to a total of $\sim$ 15k samples from the positive class (i.e., positive review), while the remaining two clients have $\sim$ 10k and $\sim$ 6k samples equally distributed over the two classes. Each agent employs a two-layer Long-Short-Term-Memory (LSTM) [44] neural network, with the same characteristics and data preprocessing employed in [41] and made available in [45]. The particular distribution of data makes so that all the clients that only have a single class available will tend to not generalise correctly, as always predicting the same class (even without considering the input data) will minimize their loss and let them reach a 100% accuracy. This in turn implies that their contribution to the federation will be questionable, if not negative, as the model they will forward for the averaging procedure will tend to be insensible to the input. To address such scenario, we propose for this simulation a different rule to compute the model weights p_i for the averaging procedure. Namely, we determine the weights as the accuracy percentage exceeding 55%, meaning that random models that reach an accuracy level of about 50% are associated to a 0 weight. Figure 8 reports a comparison of the accuracies attained by AdaFed and FedAvg in the described scenario, evaluated on the entire test set of the Amazon database. It is possible to note that AdaFed (orange) reaches an overall accuracy of 80% from the very first iterations, in line with [41], while FedAvg (blue) fails to converge and overs around 50%, which is the accuracy level expected from a random agent.

Table 1. Simulation 1.B - Data distribution among clients

Client	Classes									
1	10	0	30	10	30	50	20	20	10	10
2	10	0	0	500	100	0	0	500	100	500
3	0	0	30	500	100	150	500	0	0	500
4	0	0	30	0	100	0	500	500	100	0
5	0	10	30	500	0	0	500	500	0	500
6	0	10	10	10	10	100	10	0	10	3000

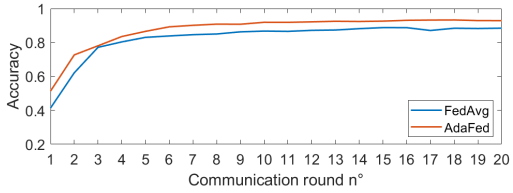

Fig. 6. Simulation 1.C (MNIST): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

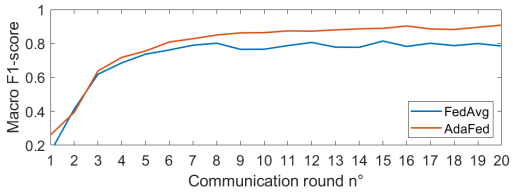

Fig. 7. Simulation 1.C (MNIST): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

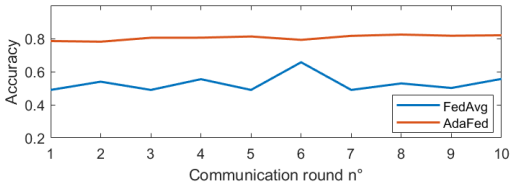

Fig. 8. Simulation 1.D (OARF Sentiment Analysis): Evolution of the server model Macro F1-score over communication rounds, evaluated on the separated test set of the server.

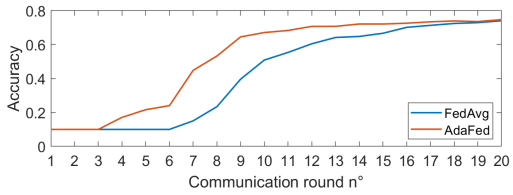

Fig. 9. Simulation 2 (CIFAR10): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

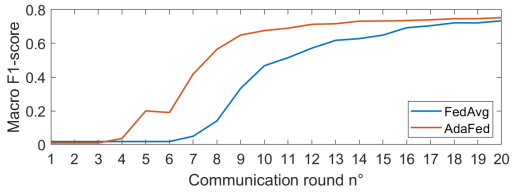

Fig. 10. Simulation 2 (CIFAR10): Evolution of the server model macro F1-score over communication rounds, evaluated on the separated test set of the server.

4.2 Simulation 2 (CIFAR10) - Transfer Learning

Having validated the effect of the two proposed modifications, we now test AdaFed on a more complex classification problem. The dataset considered for this simulation is the well-known CIFAR10 (Canadian Institute for Advanced Research) [46], and to further show the flexibility of the proposed approach we consider in this section a federated Transfer Learning [47] [48] solution. This case study is of particular importance, as transfer learning significantly reduces the number of trainable parameters by starting the training process with a neural network that was already trained to solve a similar task. The usage of such a network as the basis for the new predictor, effectively allows for a knowledge transfer between the previosuly solved probelm and the new one (e.g., a typical solution in specialised computer vision task is to employ transfer learning with a general purpose image classifier that was trained on a complex - yet general - dataset such as ImageNet [49]). The reduction of the number of the trainable parameter leads directly impacts the training complexity and communication overhead needed to sustain the federation, making the combination of FL and transfer learning an efficient and effective solution.

We also utilise this simulation to test AdaFed on a more complex task in a more general scenario that was not designed to stress any particular feature of the proposed algorithm.

Being the dataset constituted by ~ 60k 32x32 color images of 10 different classes, we consider $K = 8$ clients divided in two groups: the first four clients were given between 300 and 600 samples for each of the ten classes, while the remaining four only had data from five classes. The transfer learning model was constituted by the VGG19 [50] network trained for ImageNet [49] and attached to four dense layers with respectively 1024, 512, 256 and 128 neurons and ReLU [51] activation functions. The same adaptive loss and weighted averaging procedures as in simulation 1 were employed. We also set $E = 5$, $B = 100$, $R = 20$.

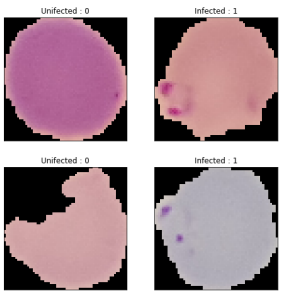

Fig. 11. Simulation 3.A (NIH malaria dataset): Example of dataset images.

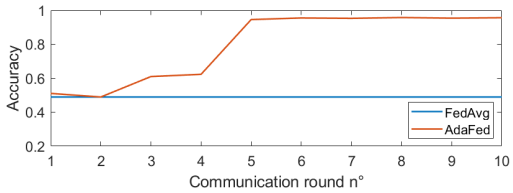

Fig. 12. Simulation 3.A (NIH malaria dataset): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

Looking at Figure 9, it is clear that AdaFed outperforms FedAvg starting from the 4th round, while the latter reaches the performance of the former as late as the 19th communication round. This behaviour can be partially explained by Figure 10, which shows that AdaFed has an overall better F_1 score performance, and by its improved weighted averaging rule, namely its ability to increase the weight of clients that have a more balanced dataset at their disposal.

4.3 Simulation 3 - Medical Imaging

As introduced, for its privacy preserving features, one of the most promising applications of FL is in the healthcare domain [52]. In fact, over the last few years several disruptive ML solutions have been developed to support medical operators and caretakers, but their development and training typically required a complex and expensive data collection campaign subject to several strict regulations such as GDPR.

The FL framework provides an opportunity to enable the collaboration among clinical institutions by abolishing any confidential data exchanges. In this section, we will utilise two different medical datasets to demonstrate some additional characteristics of ADAFED.

4.3.1 Simulation 3.A (NIH malaria). In this simulation we test our algorithm on the NIH malaria dataset [53] that consists in 27,558 cell images (of which 10% were reserved for the test set). The task associated to this dataset is a binary classification one and consists in discerning whether the depicted cell is infected. Figure 11 reports an example of images contained in the dataset.

We considered $K = 5$ clients, figuratively representing five different medical institutions, and we divided uniformly the dataset among them. The neural network we used consisted in a stack of

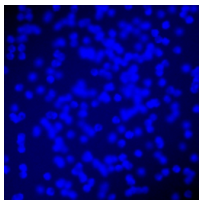

Fig. 13. Simulation 3.B (VGG-Cell): Example of dataset image

convolutional layers of 32,32,64,64,128,128 filters of size 3×3 followed by a fully connected layer of 128 neurons, all with ReLu activation functions and a dropout of 0.15. The output layer of the network consisted in a layer with a single sigmoid neuron. The loss considered was, as customary for binary classification tasks, the binary cross-entropy and it was minimized by an ADAM optimizer initialized with the standard Keras parameters. All other parameters were the same as in the first simulation.

The neural network architecture performs very well when trained on the entirety of the dataset, fast reaching a classification accuracy on the test set of over 95% in about three epochs. Conversely, due to the significant level of dropout and the relatively high number of parameters, the same network performed poorly on the individual clients, often becoming insensitive to the input and returning, as prediction, always the (slightly) more common label in the available dataset. This phenomenon compromises the performance of FedAvg, as shown in Figure 12 (blue line), where the federation does not manage to outperform a random agent. On the contrary, AdaFed is able to discard the models that show this unfavourable behaviour and, by propagating a neural network that combines only the properly trained ones, is able to reach the centralised performance (95%) after about five communication rounds.

As already shown in the test on the OARF classification task (section 4.1.4), AdaFed demonstrated robustness against adversarial/unfavourable clients, allowing the federation to solve the task at hand.

4.3.2 Simulation 3.B (VGG-Cell) - Regression. In this final experiment we consider a regression problem, consisting in counting the number of cells that appear in a microscope image. For this task, in [54] the VGG-cell dataset was proposed, consisting of 200 simulated images (e.g., see Figure 13). The considered federation is formed by $K = 4$ clients (once again representing different medical institutions) with evenly distributed data. The client's model implemented is the deep autoencoder [55] proposed in [56], that is characterised by over 3M parameters. The autoencoder is to be trained to reconstruct an image that identifies the centers of the cells, so that the image integral (pixel-wise sum) is equal to the cell count. For both AdaFed and FedAvg, for the training of the clients we implemented a data augmentation procedure that rotated and flipped each sample, obtaining eight times the original data. We set $E = 3$, $B = 50$ and $R = 10$. The evaluation of the model performance and loss is carried out on a separated test set consisting in 10% of the original data, which was also augmented as described above.

In this simulation, the model averaging procedure is conducted based on a different performance index with respect to the pixel-wise mean-squared error loss of the models, and it is set as the mean absolute percentage error (MAPE) of the cell counting.

Figures 14 and 15 show that also for this regression task AdaFed is able to outperform the FedAvg: the model loss decreases faster and, after 10 rounds, the loss value is 150.6 for AdaFed and 152.6 for FedAvg; the cell count MAPE reaches a value of about 6% at the fourth communication round while FedAvg needs 7 rounds to reach the same value. The main reason behind this behaviour is that AdaFed emphasizes the better performing clients, enabling a more efficient knowledge sharing through the federation. On the contrary, FedAvg gives the same importance to clients that are currently failing the task, consequently affecting and slowing down the convergence of the overall federated model.

5 CONCLUSIONS

In a federated learning setting, AdaFed envisages the dynamic update at every communication round of the clients' models loss functions, depending on the performance achieved by the server's model, combined with a weighted model averaging procedure that depends on the individual clients' model evaluation.

AdaFed allows to deal with non-IID, imbalanced and extremely distributed data also in the presence of malicious/bad performing federation members. The proposed solution was shown to achieve good performance in both classification and regression tasks, also in transfer learning settings.

Future research directions involve the explicit inclusion of communication efficiency and privacy-related features in the framework. More extensive tests are also required to evaluate the algorithm in heavily distributed settings with hundreds of clients.

6 ACKNOWLEDGMENTS

This work has been partially funded by the Lazio region, in the scope of the project FedMedAI, POR FESR Lazio 2014 – 2020 (Azione 1.2.1), Prot. n. A0375-2020-36491 - 23/10/2020

REFERENCES

[1] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, May 2020. doi:10.1109/msp.2020.2975749.

[2] Jie Xu, Benjamin S. Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19, November 2020. doi:10.1007/s41666-020-00082-4.

[3] Micah J. Sheller, Brandon Edwards, G. Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R. Colen, and Spyridon Bakas. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10(1), December 2020. doi:10.1038/s41598-020-69250-1.

[4] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Seth Hampson. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017. JMLR: W&CP volume 54*, 2016. URL: arXiv:1602.05629.

[5] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020. arXiv:1812.06127.

[6] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging, 2016. URL: arXiv:1602.05629.

[7] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection, 2019. URL: arXiv:1907.09693.

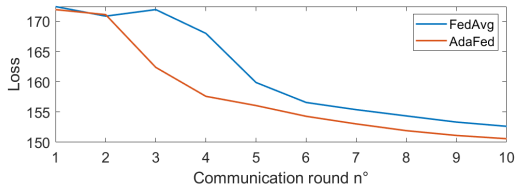

Fig. 14. Simulation 3.B (VGG cell): Evolution of the server model loss over communication rounds, evaluated on the separated test set of the server.

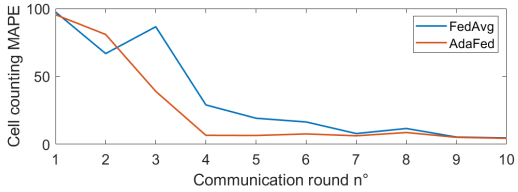

Fig. 15. Simulation 3.B (VGG cell): Evolution of the server model performance metric (cell counting mean absolute percentage error) over communication rounds, evaluated on the separated test set of the server.

[8] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, pages 1–1, 2020. doi:10.1109/comst.2020.2986024.

[9] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, February 2019. doi:10.1145/3298981.

[10] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, October 2017. doi:10.1145/3133956.3133982.

[11] Robin C. Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective, 2017. URL: arXiv:1712.07557.

[12] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks, 2018. URL: arXiv:1812.00910.

[13] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security - AISec19*. ACM Press, 2019. doi:10.1145/3338501.3357370.

[14] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. IEEE, April 2019. doi:10.1109/infocom.2019.8737416.

[15] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency, 2016. URL: arXiv:1610.05492.

[16] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training, 2017. URL: arXiv:1712.01887.

[17] Felix Sattler, Simon Wiedemann, Klaus-Robert Muller, and Wojciech Samek. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2019. doi:10.1109/tnnls.2019.2944481.

[18] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction, 2018. URL: arXiv:1811.03604.

[19] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions, 2018. URL: arXiv:1812.02903.

[20] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard, 2019. URL: arXiv:1906.04329.

[21] Jiahuan Luo, Xueyang Wu, Yun Luo, Anbu Huang, Yunfeng Huang, Yang Liu, and Qiang Yang. Real-world image datasets for federated learning, 2019. URL: arXiv:1910.11089.

[22] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. Fedvision: An online visual object detection platform powered by federated learning, 2020. URL: arXiv:2001.06202.

[23] Ahmed Imteaj and M. Hadi Amini. Distributed sensing using smart end-user devices: Pathway to federated learning for autonomous IoT. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, December 2019. doi:10.1109/csci49370.2019.00218.

[24] Theodora S. Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch. Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics*, 112:59–67, April 2018. doi:10.1016/j.ijmedinf.2018.01.007.

[25] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. FedHealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, pages 1–1, 2020. doi:10.1109/mis.2020.2988604.

[26] Micah J. Sheller, G. Anthony Reina, Brandon Edwards, Jason Martin, and Spyridon Bakas. 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*, pages 92–104. Springer International Publishing, 2019. doi:10.1007/978-3-030-11723-8_9.

[27] Yiqiang Chen, Xiaodong Yang, Xin Qin, Han Yu, Biao Chen, and Zhiqi Shen. Focus: Dealing with label quality disparity in federated learning, 2020. URL: arXiv:2001.11359.

[28] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. A principled approach to data valuation for federated learning, 2020. arXiv:2009.06192.

[29] Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[30] Tianshu Song, Yongxin Tong, and Shuyue Wei. Profit allocation for federated learning. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, December 2019. doi:10.1109/bigdata47090.2019.9006327.

[31] Jonathan T. Barron. A general and adaptive robust loss function. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019. doi:10.1109/cvpr.2019.00446.

[32] A. Ali Heydari, Craig A. Thompson, and Asif Mehmood. Softadapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions, 2019. URL: arXiv:1912.12355.

[33] Conrado Silva Miranda and Fernando José Von Zuben. Multi-objective optimization for self-adjusting weighted gradient in machine learning tasks, 2015. URL: arXiv:1506.01113.

[34] Brian Teixeira, Birgi Tamersoy, Vivek Singh, and Ankur Kapoor. Adaloss: Adaptive loss function for landmark localization, 2019. URL: arXiv:1908.01070.

[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. doi:10.1109/cvpr.2016.91.

[36] Seyed Mehdi Ayyoubzadeh and Xiaolin Wu. Adaptive loss function for super resolution neural networks using convex optimization techniques, 2020. URL: arXiv:2001.07766.

[37] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Lecture Notes in Computer Science*, pages 23–37. Springer Berlin Heidelberg, 1995. doi:10.1007/3-540-59119-2_166.

[38] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019. doi:10.1109/cvpr.2019.00949.

[39] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, October 2017. doi:10.1109/iccv.2017.324.

[40] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[41] Sixu Hu, Yuan Li, Xu Liu, Qinbin Li, Zhaomin Wu, and Bingsheng He. The oarf benchmark suite: Characterization and implications for federated learning systems, 2020. URL: arXiv:2006.07856.

[42] Official Keras.io documentation. Example of convolutional neural network for MNIST. URL: https://keras.io/examples/mnist_cnn/.

[43] Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs. In *Proceedings of the 22nd international conference on World Wide Web - WWW '13*. ACM Press, 2013. doi:10.1145/2488388.2488466.

[44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. doi:10.1162/neco.1997.9.8.1735.

[45] Sixu Hu, Yuan Li, Xu Liu, Qinbin Li, Zhaomin Wu, and Bingsheng He. Official implementation of the oarf benchmark suite: Characterization and implications for federated learning systems, 2020. URL: https://github.com/Xtra-Computing/OARF.

[46] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[47] Stevo Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44(3), September 2020. doi:10.31449/inf.v44i3.2828.

[48] S Bozinovski and A Fulgosi. The influence of pattern similarity and transfer learning upon training of a base perceptron b2. In *Proceedings of Symposium Informatica*, pages 3–121, 1976.

[49] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009. doi:10.1109/cvpr.2009.5206848.

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL: arXiv:1409.1556.

[51] Abien Fred Agarap. Deep learning using rectified linear units (relu). *ArXiv*, abs/1803.08375, 2018.

[52] Georgios A. Kaissis, Marcus R. Makowski, Daniel Rückert, and Rickmer F. Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311, June 2020. doi:10.1038/s42256-020-0186-1.

[53] Sivaramakrishnan Rajaraman, Sameer K. Antani, Mahdieh Poostchi, Kamolrat Silamut, Md. A. Hossain, Richard J. Maude, Stefan Jaeger, and George R. Thoma. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, 6:e4568, April 2018. doi:10.7717/peerj.4568.

[54] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In *Advances in Neural Information Processing Systems 23*, pages 1324–1332. Curran Associates, Inc., 2010.

[55] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, February 1991. doi:10.1002/aic.690370209.

[56] Weidi Xie, J. Alison Noble, and Andrew Zisserman. Microscopy cell counting and detection with fully convolutional regression networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 6(3):283–292, May 2016. doi:10.1080/21681163.2016.1149104.