Project Report – Privacy-Preserving and Explainable Federated Learning for Robust Digital Forensics

Advanced Research Topics in IT Security (ARTIS)

Omar Abushanab (mail)
Ibrahim Selim (mail)
Malak Abdelaziz (mail)
Peter Schropp (mail)
Matthias Högel (matthias.hoegel@uni-ulm.de)

January 9, 2026

1. Introduction

Electronic devices are now involved in 85% of all criminal investigations. Each device generates and stores data, leaving behind a digital footprint. This footprint can be used as evidence in subsequent investigations and consists of GPS coordinates, text messages, wearables and more. In the age of big data, the amount of data is growing exponentially, making it much more difficult to search for important evidence and human capacity is not sufficient. Machine learning (ML) is intended to remedy this lack of capacity and the increasing complexity of data. ML models can be used to search for complex patterns and anomalies in large data sets. However, privacy and data security are also playing an increasingly important role. In 2018, the GDPR was published in Europe to protect the data of individuals. Data may not be shared between organizations without consent in order to train ML models. This leads to data islands, which are a problem for classic ML models that require a large data set for the training process. Federated learning aims to counteract these data islands by pursuing a decentralized approach to training an ML model. Each client owns a portion of the data and trains an ML model locally. After the training phase, all clients send their learned parameters to a server, which aggregates these parameters from all clients. This method allows ML models to be trained in a decentralized manner without having to share client data. However, this method also has vulnerabilities that can be exploited by attackers. Attackers can attempt to disrupt the training process or draw conclusions about training data by reading the clients' parameters.
The DF has a multimedia division that specializes in image forgery. This also includes the detection of fake faces. Training a fake face detection model requires real faces, which constitute sensitive data and cannot be shared without further ado. That is why a decentralized approach using FL is one way to connect the data islands between organizations.
The contributions of our project are as follows:

1. We want to analyze existing forensic ML models in the field of fake face detection and reimplement one of them in an FL architecture without reducing accuracy.

2. We want to develop a XAI module that provides insights into the model's decisions. This should reveal whether FL leads to different decisions being made.

3. The FL model should be protected against attackers by integrating robust defense mechanisms.

The rest of the project report is structured as follows. Section 1 provides an overview of existing central ML models in the field of fake face detection. The following section 2 introduces the basic concepts. This is followed by a description of an ML model and how it is converted into FL, as well as the presentation of a simulation to simulate the training process. Section 4 presents the results, which are then discussed and classified in section 5.

2. Related Work

In the literature, there are various types of approaches to fake face detection without FL.
The first group of literature uses CNNs as feature extractors and trains ML models with these features for the classification task. In [1], the authors use the VGG16 as a model and initialize it with the weights trained on ImageNet. The VGG16 consists of 13 convolutional layers, which extract the features of the input image. The output of the 13th layer is flattened and consists of 73984 features, which represent the high-level features of the image. The 140-real-fake-faces dataset is used as input and the features are extracted for each image. These features are then used to train various ML models such as logistic regression, K-means (KNN), decision trees, artificial neural networks (ANN) and random forests (RF). RF performs best with an accuracy of 78.6%, 76% precision, 79% recall and an F1 score of 77.4%.
In [11], the authors use the same approach, but the GoogLeNet, ResNet18, and SqueezeNet models are used to extract the features. In addition, the authors use the 140k-real-fake-faces-with-ELA dataset, in which all images have undergone an additional preprocessing step. This step is error level analysis (ELA), which is a forensic technique for examining image segments for varying compression levels that arise during digital editing of the image. The models again extract the high-level features of the images, which are used to train KNN and Support Vector Machine (SVM) models. The combination of ResNet18 and SVM performs best in fake face detection with an accuracy of 88.6%, 88.5% precision, 89% recall and 85% f1-score.
The second group of fake face detection approaches does not use CNNs as feature extractors, but trains them and uses them for classification. Almost all of the papers mentioned use the 140k real-fake-faces dataset, which makes comparison easier.
In [5], the authors developed a very lightweight fake face detection system that uses LightFDDNetv1 and v2, which contain 3 and 5 convolutional layers and one output layer, respectively. These models contain very few parameters so that they can be used on edge devices. Both models were trained using transfer learning. LightFDDNetv1 achieved a test accuracy of 69.9%, 62% precision, 85% recall and an f1-score of 71.2% after 10 epochs and LightFDDNetv2 accuracy of 71.2%, 78.2% precision, 71% recall and an f1-score of 74.5%.
In [12], the authors used the stacking ensemble learning method, in which several models are trained separately from each other on the same dataset and the predictions are combined during the prediction process. The models used were EfficientNetB0, MobileNetv1 and MobileNetv2, which were trained on the FFHQ dataset using transfer learning. This enabled the authors to achieve an accuracy of 96.4%, 97.8% precision, 97.4% recall and 97.6% f1-score.
In [6], the authors use EfficientB0 with transfer learning. They use a learning rate scheduling technique to adjust the learning rate based on the training epoch. This allows them to achieve an accuracy of 99.06% and a loss of 0.057.

3. Background

3.1. Digital Forensics and the use of Machine Learning

A perpetrator always leaves traces of evidence of their involvement at the crime scene, as described by Dr. Edmond Locard in his exchange principles, which are used in forensic science [2].
The landscape of forensic science has changed due to the rise of electronic devices, which play an increasingly important role in our daily lives and are often connected to the internet and accessible from anywhere. The field has expanded since the early 2000s with digital forensics

(DF), which specializes more in the growing number of cybercrimes. However, even crimes that are not classified as cybercrime are becoming increasingly digital in most modern crime scenes. According to the EU, digital evidence is involved in 85% of criminal investigations. This evidence consists of data generated in our daily lives through the use of digital devices, leaving behind a digital footprint. The footprint consists of data generated by wearable devices, emails, cloud service providers, online payments and other sources [3].

The field of digital forensics can be divided into seven identifiable sub-areas, namely blockchain, networks, mobile, cloud, IoT, file systems & data storage and multimedia. This project is limited to the sub-area of multimedia, which specializes in image forger [3].

A major challenge in this field is dealing with large, complex data sets and classifying them. This problem can be addressed by machine learning (ML), whose techniques have expanded and improved in recent years. ML techniques search through the data and look for anomalies and patterns in the investigation process. The largest area for ML in DF is image forensics, accounting for 62.7%. In the field of image forensics, convolutional neural networks (CNNs) are typically used to recognize such complex patterns in the data, which is why this approach was pursued in the project. [9]

3.2. CNNs for face fake detection

3.2.1 Fundamentals

In recent years, there has been intensive research into CNNs in the areas of image classification, facial recognition and facial expressions, resulting in significant improvements. However, with the emergence of increasingly sophisticated deepfakes, it is becoming more and more difficult to distinguish between real and fake faces [6]. During the training process, CNNs learn complex patterns and can provide information about whether an input image is fake or real. Among CNN architectures, EfficientNet is one of the most modern models, as it is faster, has fewer parameters and has more capabilities for extracting features than other CNN models. These parameters are also called weights and can be learned by the model during the training process. During the training process, the model is shown many input images of positive (real faces) and negative (fake faces) [14]. The model independently learns complex patterns in the form of weights over several iterations (epochs) in order to distinguish positive from negative examples in the case of a binary classifier, or to assign probabilities to classes in the case of a multiple classifier system. [7][4]. What is special about CNNs are the convolutional layers, which reduce the resolution of the images and extract the spatial local features through weighted convolutions. Such local features can be low-level features such as edges, end points and corners in the first convolutional layers and complex high-level features in the last layers. In the final layers, the high-level features (3D vector) are combined into a fully connected layer (1D) to make a classification decision. In this project, the output consists of two output nodes with probabilities indicating whether the input image is a fake or real face [14].

3.2.2 Transfer Learning

To avoid training a model from scratch for a task, a pre-trained model can be reused and retrained for a new task. This process is called transfer learning. To do this, the fully connected layers are replaced with the number of nodes required for the new task. In the case of fake face detection classification, two nodes are required for the probabilities of the image being a real or fake face. Furthermore, the weights in the first layers (close to the input) are frozen so that the parameters are not changed during the training process. These weights represent what has been learned from the original model and should remain as consistent as possible. The weights in the back layers can then be adjusted to the new task during training by fine-tuning [6].

3.2.3 Model Explainability using SHAP

TBD

3.3. Federated Learning

3.3.1 Fundamentals

In the age of big data, data privacy and security are becoming increasingly important and are receiving more attention. The in 2018 published GDPR aims to protect individuals' data, users of services should be able to have their data deleted at any time and data may not be used for training ML models without consent. This leads to a problem in training classical ML models, as large amounts of data form the basis for them. However, the data volumes are now not stored centrally in a single data set but scattered across data islands. Federated learning addresses this problem by training ML models with data from data islands without moving the data from the islands, thereby preserving privacy. [15]

To perform FL, three main steps must be carried out, as described in [8].
1 - Model selection: A global model must be selected, which is initialized with pre-trained weights to speed up the learning process. The global model is located on a central server.
2 - Model training: The parameters of the global model are distributed to all clients. Each of the clients owns a portion of the data for the training process. The clients initialize their local model with the parameters of the global model and train it for several epochs.
3 - Parameter aggregation: All clients send their local parameters to the server. The server updates the global model with an algorithm that incorporates the local updates from the clients into the global model.
The last two steps are repeated until a specified number of epochs or a desired accuracy has been achieved.

3.4. The Flower Federated Learning Framework

Flower is an open-source framework designed for FL applications. It provides a flexible and extensible architecture that enables machine learning across decentralized data while maintaing data privacy on client devices.

Key Advantages

1. **Framework Agnosticism**: Flower supports many ML frameworks such as PyTorch, TensorFlow, JAX and scikit-learn. This allowed us freedom in development without compatibility issues.

2. **Low Learning Curve and Ease of Use**: Flower is specifically designed with developer experience in mind. Its clean, minimal API allows for quick prototyping—a working FL system can be implemented with just a few lines of code. Compared to alternatives like TensorFlow Federated (which requires learning specialized abstractions) or PySyft (with its complex privacy-preserving cryptography integration), Flower offers a more intuitive approach that reduces the barrier to entry for both researchers and practitioners.

3. **Scalability**: The framework supports deployments ranging from small-scale simulations to production systems with hundreds of clients. This scalability enables both algorithm development in simulated environments and real-world deployment across distributed devices.

4. **Production Readiness**: Beyond research use cases, Flower includes features necessary

for production deployment, including fault tolerance, client management and monitoring capabilities. This makes it suitable for both experimental validation and real-world applications.

3.4.1 Attack Vectors in FL

Attack vectors on FL are divided into two large groups, which are described in [10].
On the one hand, there are model performance attacks, which focus on damaging the training process in various ways. These include data poisoning attacks, in which the training data is modified by replacing images or labels before the training. Model poisoning attacks, in which the gradients of the locally trained model are altered. Free-riding attacks, in which the model is only used without adding value to the global model.
On the other hand, there are privacy attacks that attempt to draw conclusions about training data from the local parameters of clients or the global parameters of the global model. These include model inversion and gradient inference attacks, as well as GAN reconstruction attacks, in which private training data is to be reconstructed.

4. Methodology

4.1. Experimental Setup

A fake face detection model is developed using federated learning (FL) to distinguish between real and fake faces. First, a centrally trained model without FL is implemented and used as a baseline for performance comparison. Second, a decentralized fake face detection model is trained using FL in a simulated multi-client environment.

4.1.1 Dataset

The well-known 140k real-fake faces dataset was used, which consists of 70,000 real faces and 70,000 fake faces with a image size of 256px [13]. The fake faces were generated using StyleGAN, a generative adversarial network developed by NVIDIA that is capable of producing highly photorealistic synthetic facial images.

4.1.2 Model

The base model chosen is EfficientNet-B0, a convolutional neural network (CNN), which was also used by Khudeyer et al. [6]. The authors trained a fake face detection model using transfer learning. For this purpose, the model was initialized with the pretrained weights of EfficientNetB0 on the ImageNet dataset. A lightweight head was attached to the pre-trained base model, consisting of global average pooling, a 256-dimensional fully connected layer with ReLU activation, batch normalization, and dropout, followed by a 2 dimensional softmax output layer. The output is a 2-dimensional vector with probabilities indicating whether the input image is a fake or real face. The model was optimized with binary cross-entropy loss and the Adam optimizer. All input images for training were resized to 244px.

4.2. Centralized Fake Face Detection Model

The authors of [6] developed a method for fake face detection using CNN, which achieves an accuracy of 99.06%. This approach is used as a benchmark for comparing central fake face detection with fake face detection using FL. This work was reimplemented for verification purposes in order to ensure a meaningful comparison.
The dataset was divided into 100.000 training images, 20.000 test images, and 20.000 validation images. EfficientNetB0 with the hyperparameters of section 4.1.2 is used. Training was

Epoch	**Learning rate**
$epoch \leq 2$	0.01
$2 < epoch \leq 15$	0.001
$epoch > 15$	0.0001

Table 1: Adjustment of the learning rate during training.

performed with a batch size of 32 over 30 epochs with early stopping to reduce training time. The paper presented a learning rate scheduler that adjusts the learning rate during training based on the epoch, as shown in Table 1. The learning rate scheduler ensures that significant weight adjustments are made early in training. Furthermore, in later iterations, a strong adjustment is prevented by the decreasing learning rate. This leads to faster convergence in early epochs, while weight optimizations can be performed in later itterations.
Due to time constraints, the model was only trained once, as training was very computationally intensive due to the large amount of data.

4.3. Decentralized Fake Face Detection using FL

In this section a decentralized trained fake face detection model is developed. The training process will be explained below. In addition, we will discuss how privacy attacks can be prevented during training.

4.3.1 Research Scenario

The following scenario is fictional.
Several research organizations want to work together to train a fake face detection model. Each individual organization has images of fake faces, but also images of faces that belong to their customers. An ML model should be trained together that can distinguish between real and fake images. To do this, a large data set containing all images would have to be created in order to train the model. However, all organizations are interested in protecting the privacy of their customers and therefore do not want to share the real images. The solution is to train the model using FL. This fake face detection model should be trained with all data from all organizations and should be available to everyone without the need to share data between organizations.

4.3.2 Thread Model

All participants in the fake face detection training process are trusted. This means that all model performance attacks that seek to undermine the convergence of the global model can be excluded. After training, participants send their model weights to a trusted third-party server. This ensures the secure aggregation of weights. Weights sent by participants to the trusted third party could be captured during transmission. Capturing the weights of individual organizations represents an attack vector for privacy attacks. This attack vector should be reduced by the proposed encryption.

4.3.3 FL Simulation

A simulation environment is developed to simulate a global model with any number of participants. The EfficientNetB0 with the hyperparameters of section 4.1.2 is used as the base model. A batch size of 128 is selected. The training data set was divided among five participants, with each receiving approximately 20,000 images, comprising 10,000 real faces and 10,000 fake faces. On the server, the global model is initialized with the weights from EfficientNetB0, which was pre-trained on the ImageNet dataset. The global model is trained in 10 epochs.
Each epoch consists of the following steps.

First, the weights of the global model are distributed among the 5 participants. Each participant initializes their local model with the global weights. The participants train their local models in 3 epochs. The weights of the local models are encrypted to prevent privacy attacks by intercepting the weights. The server receives the weights and decrypts them. The global model parameters are updated using weighted federated averaging (FedAvg), where each client's contribution is proportional to the number of local samples.

4.3.4 FL attack mitigations

If the local updates of the participants are captured during transmission to the server, privacy attacks on the participants' data can be carried out. To counteract this, the weight updates must be encrypted. The weights are serialized in bytes and encrypted with AES-128. Each participant has a secret key that the server knows as a trusted third party. When a server receives the encrypted data, it can decrypt it for further processing.

4.4. Explainability using SHAP

In order to analyze the centrally trained model and the decentralized trained model, the decisions made by the models should be compared for individual inputs. The aim is to clarify whether the models have developed differently as a result of the different training processes and whether they focus on different details when making decisions. The SHAP method is used for this purpose.

5. Results

This are the results of the paper.

6. Discussion

This is the discussion of the paper.

7. Conclusion

This is the conclusion of the paper.

References

[1] Abdullah Mujawib Alashjaee. "Machine Learning Approach for Fake Image Forensics". In: *University of Bisha Journal for Basic and Applied Sciences* 1.1 (2025), p. 5.

[2] Jeroen Bode. "Every Contact Leaves a Trace: A Literary Reality of Locard's Exchange Principle". In: *Outside the Box: A Multi-Lingual Forum.* 2019, p. 18.

[3] Fran Casino et al. "Research trends, challenges, and emerging topics in digital forensics: A review of reviews". In: *Ieee Access* 10 (2022), pp. 25464–25493.

[4] Bilal Hadjadji, Youcef Chibani, and Yasmine Guerbai. "Multiple one-class classifier combination for multi-class classification". In: *2014 22nd International Conference on Pattern Recognition.* IEEE. 2014, pp. 2832–2837.

[5] Günel Jabbarlı and Murat Kurt. "Lightffdnets: Lightweight convolutional neural networks for rapid facial forgery detection". In: *arXiv preprint arXiv:2411.11826* (2024).

[6] Raidah Salim Khudeyer and Noor Mohammed Almoosawi. "Fake Image Detection Using Deep Learning". In: *Informatica* 47.7 (2023).

[7] Ana Carolina Lorena, André CPLF De Carvalho, and João MP Gama. "A review on the combination of binary classifiers in multiclass problems". In: *Artificial Intelligence Review* 30 (2008), pp. 19–37.

[8] Hania Mohamed et al. "Harnessing federated learning for digital forensics in IoT: A survey and introduction to the IoT-LF framework". In: *IEEE Open Journal of the Communications Society* (2024).

[9] Tahereh Nayerifard et al. "Machine learning in digital forensics: a systematic literature review". In: *arXiv preprint arXiv:2306.04965* (2023).

[10] Helio N Cunha Neto et al. "A survey on securing federated learning: Analysis of applications, attacks, challenges, and trends". In: *IEEE Access* 11 (2023), pp. 41928–41953.

[11] Rimsha Rafique et al. "Deep fake detection and classification using error-level analysis and deep learning". In: *Scientific reports* 13.1 (2023), p. 7422.

[12] Emre Şafak and Necaattin Barışçı. "Detection of fake face images using lightweight convolutional neural networks with stacking ensemble learning method". In: *PeerJ Computer Science* 10 (2024), e2103.

[13] Bojan Tunguz. *140K Real and Fake Faces*. `https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces`. Accessed: 2025-01-05. 2020.

[14] Wei Wang et al. "Development of convolutional neural network and its application in image classification: a survey". In: *Optical Engineering* 58.4 (2019), pp. 040901–040901. DOI: `10.1117/1.OE.58.4.040901`.

[15] Chen Zhang et al. "A survey on federated learning". In: *Knowledge-Based Systems* 216 (2021), p. 106775.