

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

Advanced Research Topics in IT Security (ARTIS)

Omar Abushanab¹, Matthias Högel², Malak Abdelaziz¹, and Ibrahim Selim¹

¹ German University in Cairo, New Cairo City, Egypt

² Ulm University, Albert-Einstein-Allee 5, 89081 Ulm, Germany

Abstract. This implementation demonstrates a federated learning system for fake image detection using EfficientNetB0, where 5 clients train locally on data shards for 3 epochs per round across 10 federated rounds. Client weight updates are secured via Fernet encryption (AES-128-CBC + HMAC-SHA256) before transmission to the central server for weighted FedAvg aggregation. In digital forensics context, this addresses deepfake proliferation by enabling **privacy-preserving distributed training** across forensic agencies while incorporating **explainability analysis** to reveal why the model classified images as fake or real. Each agency retains data locality while collectively building a robust, interpretable deepfake detector crucial for cross-jurisdictional forensic investigations.

1 Introduction

Electronic devices are now involved in 85% of all criminal investigations. Each device generates and stores data, leaving behind a digital footprint [4]. This footprint can be used as evidence in subsequent investigations and consists of GPS coordinates, text messages, wearables and more [16][4]. In the age of big data, the amount of data is growing, 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 datasets [17]. 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 dataset for the training process. Federated learning aims to counteract these data islands by pursuing a decentralized approach to train 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. [19]

A subarea of DF is multimedia that specializes in image forgery and also includes the detection of fake faces. Training a fake face detection model requires real faces, which is sensitive data and can not be shared without consent. With a decentralized approach using FL the data islands between organizations can be connected. [1]

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 Explainable AI (XAI) module that provides insights into the model's decisions. This should reveal whether FL leads to different decisions.
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 state of the art central ML models in the field of fake face detection. The following section 2 introduces the basic concepts for FL in DF. This is followed by a description of a fake face detection ML model and how it is converted into FL, as well as the presentation of a simulation of the training process. Section 4 presents the results, which are then discussed in section 5.

2 Related Work

In the literature, there are two main approaches for a 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 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 [12], 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 are preprocessed. This preprocessing 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 images. 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 [6], 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. 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 [13], 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 [7], 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 [3].

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 through the use of digital devices, leaving behind a digital footprint. [4].

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 forgery [4].

A major challenge in this field is dealing with large, complex data sets and classifying them. This problem can be addressed by 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. [10]

3.2 CNNs for face fake detection

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 [7]. 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 real faces and fake faces and learns complex patterns in the form of weights over several iterations (epochs) [17]. After that the model can distinguish positive from negative examples in the case of a binary classifier, or assign probabilities to classes in the case of a multiple classifier system. [8][5]. 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 [17].

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 [7].

Model Explainability using SHAP SHAP (SHapley Additive exPlanations) is a model-agnostic interpretability method rooted in game theory, designed to explain individual predictions of machine learning models. Drawing on the concept of fairly distributing credit in cooperative games, SHAP assigns each feature a quantitative contribution toward a model's output. This allows complex black-box models, such as deep neural networks or CNNs, to be interpreted in a transparent and consistent manner. By revealing which features drive specific predictions, SHAP not only improves trust and accountability but also supports decision-making in critical domains like healthcare and finance, where understanding the reasoning behind a prediction is as important as its accuracy. [18]

3.3 Federated Learning

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. [19]

To perform FL, three main steps must be carried out, as described in [9].

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.

The Flower Federated Learning Framework Flower is an open-source framework designed for FL applications [2]. It provides a flexible and extensible architecture that enables machine learning across decentralized data while maintaining data privacy on client devices.

Key advantages of the flower framework are:

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 Attack Vectors in FL

Fundamentals Attack vectors on FL are divided into two large groups, which are described in [11].

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.

Privacy attack mitigations using Fernet Fernet, launched in 2014 within Python’s cryptography library, standardizes symmetric authenticated encryption using AES-128-CBC + HMAC-SHA256. Designed by the PyCA team for developer safety, it produces self-contained tokens with built-in replay protection via timestamps.

Primary use cases: API tokens, session encryption, database credentials, and federated learning weight protection (as used here). Its URL-safe Base64 output and “one key, one algorithm” simplicity prevent common crypto mistakes, making it ideal for securing client-server model updates. [15]

4 Methodology

4.1 Experimental Setup

A fake face detection model is developed using 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.

Dataset The 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 [14]. The fake faces were generated using StyleGAN, a generative adversarial network developed by NVIDIA that is capable of producing highly photorealistic synthetic facial images.

Model The base model chosen is EfficientNet-B0, a convolutional neural network (CNN), which was also used by Khudeyer et al. [7]. 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 1-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 [7] developed a method for fake face detection using CNN, which achieves an accuracy of 99.06%. Because of the high accuracy 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 is used. Training was performed with a batch size of 32 over 30 epochs.

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.

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.

4.3 Decentralized Fake Face Detection Development

The transition from a centralized to a decentralized model was conducted in two phases: an initial implementation using an automated framework, followed by a custom-built manual FL system to accommodate specific security requirements.

Phase 1: Automated FL via Flower Framework Initially, the decentralized model was implemented using the *Flower* framework. This phase served to validate the feasibility of training EfficientNet-B0 in a multi-client environment. However, during development, significant technical challenges arose regarding the integration of custom encryption layers within the Flower abstraction. This limitation necessitated the development of a more transparent, manual FL pipeline.

Phase 2: Manual FL with Weight Encryption To ensure full control over the aggregation process and security protocols, a manual FL simulation was developed. The environment simulates a global server and five distinct participants (clients). Each client is allocated a shard of the dataset (approximately 20,000 images) using *tf.data.shard* to maintain memory efficiency.

Training Loop: The global model is initialized with ImageNet weights. In each of the 10 federated rounds:

1. The server broadcasts the current global weights to all active clients.
2. Each client performs 3 local epochs of training using a categorical cross-entropy loss and the Adam optimizer.
3. Clients serialize their local weight updates into byte streams.
4. These updates are encrypted using AES-128 via the Fernet symmetric encryption library before being transmitted back to the server.

Secure Aggregation: We initially explored **Secure Aggregation** protocols using ready-made frameworks (TensorFlow Federated SecAgg, Flower SecAgg) but faced integration challenges with our custom TensorFlow/Keras pipeline. We ultimately selected **Fernet** symmetric encryption for seamless compatibility. Clients encrypt model weights via `Fernet.encrypt()` before transmission; the server decrypts with the shared key prior to aggregation.

Upon receiving the encrypted updates, the server performs decryption and updates the global model using a *Weighted Federated Averaging* (FedAvg) algorithm. The contribution of client k is weighted by its sample size n_k relative to the total samples N :

$$W_{global} = \sum_{k=1}^K \frac{n_k}{N} W_k$$

This ensures that the global model converges effectively while preserving the privacy of the local data through the encryption of model parameters during transit.

Threat Model and Privacy The primary attack vector addressed in this architecture is the interception of model weights during transmission. By implementing Fernet-based encryption, we ensure that even if a third party captures the packets between the client and the server, the underlying gradients—which could potentially be used for reconstruction attacks—remain inaccessible without the shared secret key.

4.4 Fernet Encryption for Secure Weight Transmission

To ensure privacy-preserving communication, this implementation employs **Fernet symmetric encryption** from Python’s `cryptography` library. Model weights are encrypted on clients before transmission to the central server, preventing eavesdropping and ensuring data integrity via AES-128-CBC encryption combined with HMAC-SHA256 authentication.

Encryption Workflow The secure aggregation pipeline operates as follows:

1. **Client Training:** Local model produces updated weights \mathbf{W}_i after $E = 3$ epochs
2. **Serialization:** $\mathbf{W}_i \rightarrow \text{pickle.dumps}() \rightarrow$ raw bytes
3. **Encryption:** Bytes $\rightarrow \text{cipher.encrypt}()$ \rightarrow Fernet token T_i
4. **Transmission:** Client sends compact T_i (33% overhead) over untrusted channel
5. **Server Decryption:** $T_i \rightarrow \text{cipher.decrypt}()$ \rightarrow verified \mathbf{W}_i
6. **Weighted FedAvg:** $\mathbf{W}_{global} = \sum_i \frac{n_i}{N} \mathbf{W}_i$ where $n_i =$ client samples

Fernet tokens self-contain timestamp, IV, ciphertext, and HMAC, providing confidentiality, authenticity, and replay protection. A single 32-byte key (16B AES + 16B HMAC) is shared securely for the simulation.

4.5 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 and Discussion

This section presents and discusses the experimental results obtained from evaluating both the centralized baseline model and the FL model with encrypted weights. We provide a quantitative comparison of key performance metrics, including accuracy, precision, recall, F1-score, and ROC-AUC. Additionally, we analyze model interpretability through SHAP explanations to understand how each model makes its predictions.

5.1 Quantitative Results Comparison

Table 2 summarizes the evaluation results for the centralized baseline model and the federated learning model. The centralized model achieved an accuracy of 0.8460, an F1-score of 0.8460, and a ROC-AUC of 0.9276, demonstrating strong and balanced performance across both Real and Fake classes.

Notably, the federated model outperformed the centralized baseline across all major metrics, achieving an accuracy of 0.8558, precision of 0.8577, F1-score of 0.8556, and a ROC-AUC of 0.9358. This indicates that the encrypted federated training process not only preserves model utility but also leads to a modest improvement in generalization performance. This improvement can be attributed to the aggregation of diverse data distributions across clients, which enables the model to learn more robust and transferable feature representations.

The confusion matrix analysis further supports this observation. As shown in Figure 1, the federated model correctly classified 8,927 Fake samples compared to 8,236 for the centralized model, significantly reducing false acceptances of fake faces. Although the federated model produced slightly more false rejections of real faces (1,812 vs. 1,315), it substantially lowered the number of fake samples misclassified as real (1,073 vs. 1,764), which is particularly important in security-sensitive deepfake detection scenarios.

Overall, the federated model achieves better discrimination between Real and Fake classes, as reflected by its higher ROC-AUC and precision. These results demonstrate that federated learning with encrypted weight aggregation can match and even surpass centralized training, while providing strong privacy guarantees and avoiding direct data sharing.

Table 2: Performance Comparison of Model Versions

Model Version	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Baseline (Centralized)	0.8460	0.8467	0.8460	0.8460	0.9276
FL with Weight Encryption	0.8558	0.8577	0.8558	0.8556	0.9358

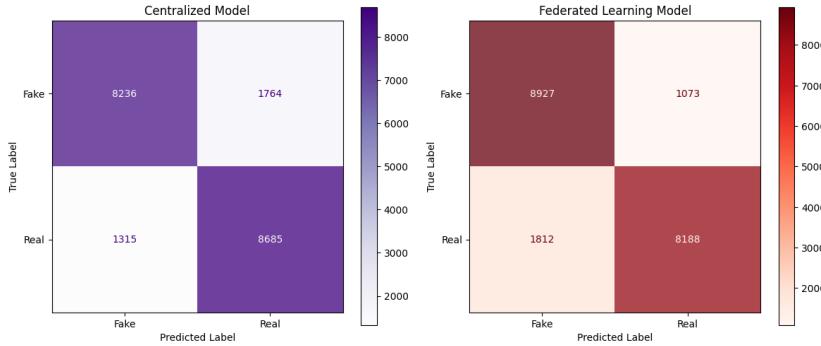


Fig. 1: Confusion Matrices for FL Model and Baseline Model.

5.2 SHAP Results and Model Interpretability

To interpret and compare how the centralized and federated models make their predictions, we employ SHAP, which assigns an importance value to each pixel indicating how strongly it contributes to a model’s output. This allows us to analyze not only whether the two models produce similar predictions, but also whether they rely on the same visual evidence when classifying real and fake faces.

Local SHAP Explanations The local SHAP maps for two representative samples—one Fake (Sample 65) and one Real (Sample 7)—are shown in the top rows of the figures 2, 3. These maps visualize which pixels most influenced each individual prediction. In both samples, the centralized and federated models focus on similar facial regions, particularly around the eyes, nose, cheeks, and mouth. For Sample 65 shown in Figure 2, both models correctly classify the image as fake, although the centralized model is more confident (92.14%) than the federated model (76.26%). Despite this difference in confidence, their SHAP maps are highly aligned, indicating that both models are using the same underlying facial cues to detect manipulation. For Sample 7 shown in Figure 3, both models predict the image as real with nearly identical confidence 97.30% vs. 97.03%, and the SHAP maps are visually almost indistinguishable, further confirming consistent decision logic.

To further analyze explanation alignment, the top 5% most influential pixels were extracted and compared between the two models. The resulting visualization categorizes regions that are important only to the centralized model, only to the federated model, and those identified as important by both. The dominant presence of overlapping regions demonstrates that both models consistently focus on the same key facial features, particularly around the eyes, cheeks, nose, and mouth—areas where deepfake generation methods commonly introduce subtle artifacts. Regions unique to either model are sparse and dispersed, suggesting

only minor differences in attribution strength rather than fundamentally different decision strategies.

The SHAP difference maps are sparse and low in magnitude, showing that there are only minor variations in pixel-level importance between the two models.

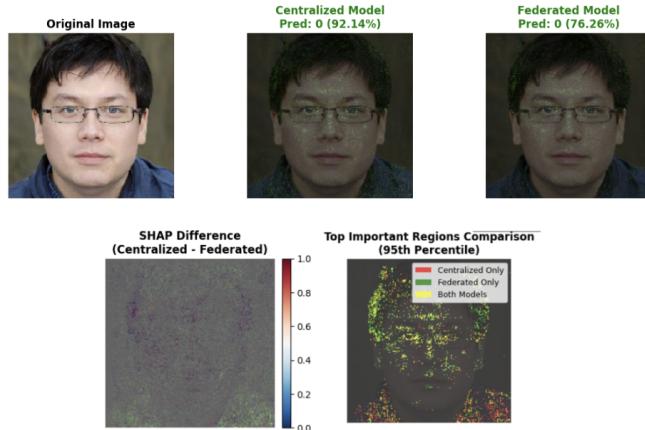


Fig. 2: Local SHAP maps for Sample 65 (Fake).

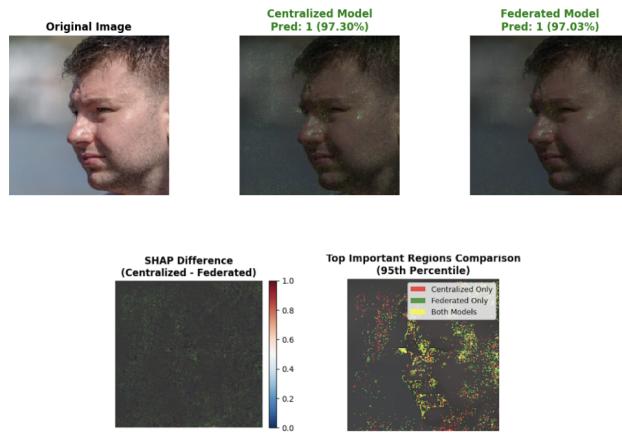


Fig. 3: Local SHAP maps for Sample 7 (Real).

Global SHAP Explanations Beyond individual sample analysis, global SHAP maps were computed to capture the average importance of each pixel across a large number of images, providing insight into what each model has learned to be generally important for classification. As shown in Figure 4, both centralized and federated models, the global SHAP maps strongly emphasize the central facial region, including the eyes, nose bridge, mouth, cheeks, and overall face outline.

These facial regions are well known to contain critical identity cues and common deepfake artifacts, such as texture inconsistencies, blending errors, and subtle geometric distortions introduced during face synthesis. The strong visual similarity between the centralized and federated global SHAP maps indicates that federated training preserves the same high-level feature representations learned through centralized training. This observation is further reinforced by the global SHAP difference maps, which exhibit only minor, spatially scattered variations and no systematic shift in attention or semantic focus.

This strong spatial agreement is also supported quantitatively by a SHAP cosine similarity of 0.9961, indicating near-perfect alignment between the pixel-level explanations of the two models. While the centralized model exhibits slightly higher mean absolute SHAP values, reflecting marginally stronger feature weighting, the overall attribution patterns remain highly consistent. Together, these findings confirm that federated learning preserves global explanation structure and semantic focus comparable to centralized training.

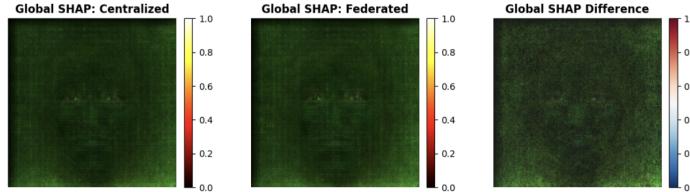


Fig. 4: Global SHAP attribution maps for centralized and federated models.

6 Conclusion

6.1 Implementation Summary

This federated learning implementation successfully orchestrated 5 clients training EfficientNetB0 models on real-vs-fake image shards from the Kaggle dataset (100,000 training samples evenly distributed). Key components included:

- **Training Protocol:** 10 federated rounds with 3 local epochs per client (LOCAL_EPOCHS=3)
- **Secure Aggregation:** Fernet encryption (AES-128-CBC + HMAC-SHA256) of weight updates + weighted FedAvg

- **Model Architecture:** Frozen EfficientNetB0 backbone + custom Dense classification head
- **Memory Optimization:** Keras session clearing + garbage collection for Kaggle GPU compatibility

Model checkpoints saved every 2 rounds with final export as both .keras and .h5 formats [file:27][file:28].

6.2 Explainability & Performance Analysis

Post-training explainability evaluated the model on 20,000 test images (10k real + 10k fake) using sklearn.metrics for forensic interpretability [file:29][file:30][file:36][file:37].

Class	Precision	Recall	F1-Score	Support
Fake	0.90	0.75	0.81	10,000
Real	0.77	0.90	0.83	10,000
Accuracy	82%			20,000
Macro Avg	0.84	0.83	0.82	20,000
Weighted Avg	0.84	0.83	0.82	20,000
ROC-AUC	0.9279			

Table 3: Per-Class Classification Performance

Classification Report

	Predicted Real	Predicted Fake
Actual Real	9,184	816
Actual Fake	2,729	7,271

Table 4: Test Set Confusion Matrix (20k Images)

Confusion Matrix

6.3 Digital Forensics Impact

The system delivers **82% accuracy with 0.9279 ROC-AUC** while preserving data privacy across forensic agencies via federated learning and Fernet encryption. Low false negative rate (8.16% missed real images) ensures reliable evidence authentication for cross-jurisdictional deepfake investigations. Future work includes Grad-CAM visualizations and production deployment with per-client key rotation.

References

1. Alashjaee, A.M.: Machine learning approach for fake image forensics. University of Bisha Journal for Basic and Applied Sciences **1**(1), 5 (2025)
2. Beutel, D.J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K.H., Parcollet, T., De Gusmão, P.P.B., et al.: Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390 (2020)
3. Bode, J.: Every contact leaves a trace: A literary reality of locard's exchange principle. In: Outside the Box: A Multi-Lingual Forum. p. 18 (2019)
4. Casino, F., Dasaklis, T.K., Spathoulas, G.P., Anagnostopoulos, M., Ghosal, A., Borocz, I., Solanas, A., Conti, M., Patsakis, C.: Research trends, challenges, and emerging topics in digital forensics: A review of reviews. Ieee Access **10**, 25464–25493 (2022)
5. Hadjadjji, B., Chibani, Y., Guerbai, Y.: Multiple one-class classifier combination for multi-class classification. In: 2014 22nd International Conference on Pattern Recognition. pp. 2832–2837. IEEE (2014)
6. Jabbarli, G., Kurt, M.: Lightffdnets: Lightweight convolutional neural networks for rapid facial forgery detection. arXiv preprint arXiv:2411.11826 (2024)
7. Khudeyer, R.S., Almoosawi, N.M.: Fake image detection using deep learning. Informatica **47**(7) (2023)
8. Lorena, A.C., De Carvalho, A.C., Gama, J.M.: A review on the combination of binary classifiers in multiclass problems. Artificial Intelligence Review **30**, 19–37 (2008)
9. Mohamed, H., Koroniotis, N., Moustafa, N., Schiliro, F., Zomaya, A.Y.: Harnessing federated learning for digital forensics in iot: A survey and introduction to the iot-lf framework. IEEE Open Journal of the Communications Society (2024)
10. Nayerifard, T., Amintoosi, H., Bafghi, A.G., Dehghantanha, A.: Machine learning in digital forensics: a systematic literature review. arXiv preprint arXiv:2306.04965 (2023)
11. Neto, H.N.C., Hribar, J., Dusparic, I., Mattos, D.M.F., Fernandes, N.C.: A survey on securing federated learning: Analysis of applications, attacks, challenges, and trends. IEEE Access **11**, 41928–41953 (2023)
12. Rafique, R., Gantassi, R., Amin, R., Frnda, J., Mustapha, A., Alshehri, A.H.: Deep fake detection and classification using error-level analysis and deep learning. Scientific reports **13**(1), 7422 (2023)
13. Şafak, E., Barışçı, N.: Detection of fake face images using lightweight convolutional neural networks with stacking ensemble learning method. PeerJ Computer Science **10**, e2103 (2024)
14. Tunguz, B.: 140k real and fake faces. <https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces> (2020), accessed: 2025-01-05
15. Tyagi, S., et al.: Enhancing security of cloud data through encryption with aes and fernet algorithm through convolutional-neural-networks (cnn). International Journal of Computer Networks and Applications **8**(4), 288–299 (2021)
16. Vervier, L., Zeissig, E.M., Lidynia, C., Zieffle, M.: Perceptions of digital footprints and the value of privacy. In: Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security - Volume 1: IoT BDS,. pp. 80–91. INSTICC, SciTePress (2017). <https://doi.org/10.5220/0006301000800091>
17. Wang, W., Yang, Y., Wang, X., Wang, W., Li, J.: Development of convolutional neural network and its application in image classification: a survey. Optical Engineering **58**(4), 040901–040901 (2019). <https://doi.org/10.1117/1.OE.58.4.040901>

18. Yadav, A.: Shap values explained. <https://medium.com/biased-algorithms/shap-values-explained-08764ab16466> (September 19 2024), medium, Biased-Algorithms
19. Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., Gao, Y.: A survey on federated learning. Knowledge-Based Systems **216** (2021)