Federated Learning, Mobile Emotion Recognition, and Client-Side Data Quality: A Survey and Research Agenda

Tarun Sakthivel [*], Sabari Krishnan P. [*], Saran V. V. [*]

Posted Date: 26 March 2026

doi: 10.20944/preprints202603.2175.v1

Keywords: federated learning; facial emotion recognition; mobile edge computing; label noise; data quality; crowdsourcing; client-side validation; affective computing

 doi:10.20944/preprints202603.2175.v1

Article

Federated Learning, Mobile Emotion Recognition, and Client-Side Data Quality: A Survey and Research Agenda

Tarun Sakthivel, Sabari Krishnan P. and Saran V. V.

Department of Computer Science and Engineering, Vellore Institute Of Technology, Amaravathi, India; tarun.22bce8896@vitapstudent.ac.in (T.S.); krishnan.22bce8879@vitapstudent.ac.in (S.K.P.); saran.22bce9264@vitapstudent.ac.in (S.V.)

Abstract

The combination of federated learning (FL), mobile edge computing, and facial emotion recognition (FER) promises privacy-preserving affective computing on personal devices. Instead of uploading raw images to the cloud, models are trained collaboratively across distributed clients while inference increasingly happens on-device. However, when systems move from carefully curated research datasets to user-generated mobile data, issues such as label noise, inconsistent annotations, and heterogeneous client data quality become central bottlenecks. These factors affect FL convergence, generalization, and downstream trust in emotion-aware applications. This survey consolidates literature from four major strands: (i) FER from classical handcrafted approaches to deep and mobile models, (ii) FL foundations and its use in vision and affective computing, (iii) robust FL under label noise and unreliable clients, and (iv) crowdsourcing and AI-assisted data-labeling quality assurance. Building on these strands, we argue that client-side data validation pipelines on mobile devices are a promising but underexplored direction. We outline an architectural blueprint for such pipelines and highlight open challenges around human–AI interaction, multimodal context, privacy, and fairness in FL-based FER systems, providing a comparative analysis of past, present, and future directions in this domain.

Keywords: federated learning; facial emotion recognition; mobile edge computing; label noise; data quality; crowdsourcing; client-side validation; affective computing

1. Introduction

Advances in deep learning have made automatic facial emotion recognition (FER) a practical component of human–computer interaction, enabling applications in education, healthcare, entertainment, and customer analytics.[1,2] Rather than processing facial images on centralized servers, recent work increasingly pushes both inference and learning to mobile and edge devices, in part to address privacy, latency, and personalization requirements for emotion-aware services.[3,4]

Federated learning (FL) offers a natural training paradigm for such settings. In FL, a central coordinator iteratively aggregates model updates from many clients instead of collecting raw data in a single repository.[5] This architecture has been successfully demonstrated in mobile keyboard prediction and is now being explored for medical imaging, speech, and affective tasks.[6,7] While FL mitigates some privacy risks, it also makes training dynamics highly dependent on the quality and distribution of client-side data, which are often noisy, unbalanced, and non-independent.[12,13]

Label noise—incorrect or ambiguous annotations—is especially harmful in FER because emotions are inherently subjective and expressions can be subtle or culturally contingent.[14,15] In FL, noisy labels are further complicated by heterogeneous clients: some devices may generate mostly clean annotations, while others act as high-noise or even adversarial sources. Recent robust FL methods address these issues at training or aggregation time, but typically assume that data have already been collected locally.[12,13]

 doi:10.20944/preprints202603.2175.v1

In parallel, the crowdsourcing and data-labeling community has developed rich mechanisms for quality control, including pre-screening workers, just-in-time feedback, and AI-assisted monitoring of labeling sessions.[16–18] These mechanisms, however, are mostly designed for centralized pipelines and rarely consider FL-specific constraints such as limited visibility into raw data or strict privacy budgets.

This survey integrates these literatures with a focus on mobile FER and proposes a research agenda around interactive client-side validation pipelines. We review FER methods and datasets, summarize key FL algorithms and applications in vision and affective computing, discuss robust FL under noisy labels, and examine crowdsourced labeling quality frameworks. We then provide a cross-cutting comparison of past, present, and emerging approaches, before outlining open research challenges.

2. Facial Emotion Recognition: From Classic to Deep and Mobile Models

2.1. Early FER Using Handcrafted Features

The first generations of FER systems relied on manually engineered descriptors and conventional classifiers. Popular feature extractors included Gabor filters, Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), and geometric measurements derived from facial landmarks. These features were typically fed to classifiers such as support vector machines (SVM), k-nearest neighbours (k-NN), or hidden Markov models (HMM) trained on relatively small and controlled datasets like JAFFE and CK+.[1]

Such datasets contained frontal faces captured under constrained lighting and focused on Ekman's prototypical emotions (anger, disgust, fear, happiness, sadness, surprise, and neutral). Despite achieving reasonable performance in lab conditions, handcrafted approaches struggled with unconstrained real-world images.[1,2] Performance deteriorated under variations in pose, illumination, occlusions (e.g., glasses, masks, hands), demographic diversity, and spontaneous rather than posed expressions. Moreover, designing robust handcrafted features required significant domain expertise, and scaling to large heterogeneous datasets was difficult, limiting sustained progress once deep learning methods emerged.

2.2. Deep Learning for FER

The advent of convolutional neural networks (CNNs) transformed FER by allowing models to learn hierarchical representations directly from raw pixels. Large facial-expression datasets such as FER2013, RAF-DB, AffectNet, and in-the-wild collections enabled training of deeper models and systematic comparisons against handcrafted baselines.[1,2] Surveys and experimental studies consistently report that architectures such as VGG, ResNet, Inception, and MobileNet outperform classical pipelines on standard FER benchmarks, often by significant margins in accuracy and robustness.[2]

Deep FER research has explored a range of techniques: data augmentation to mimic pose and illumination changes, transfer learning from large-scale face or object datasets, and multi-task learning where expression recognition is combined with auxiliary tasks such as identity or facial action unit prediction.[1] More recent work introduces attention mechanisms, graph-based reasoning over facial landmarks, generative models to handle data scarcity, and temporal modelling for video-based FER.[2] At the same time, multiple review papers highlight persistent challenges: subtle and overlapping expressions, inter-subject variability, class imbalance, cross-dataset domain shift, and noisy or ambiguous labels.[1,14]

2.3. Mobile and Real-Time FER Systems

As mobile devices and embedded hardware have become more capable, there is growing interest in running FER locally for interactive applications such as emotion-aware messaging, gaming, tutoring, and telehealth.[3,4] Many prototypes deploy lightweight CNN backbones like MobileNet or EfficientNet variants, often quantized or pruned, to achieve real-time inference on smartphones or

 doi:10.20944/preprints202603.2175.v1

edge devices. These systems demonstrate that, with careful model selection and optimization, it is feasible to process camera streams and estimate emotions at interactive frame rates.[3]

Li et al. present a complete pipeline for real-time facial affective computing on mobile devices, integrating efficient CNN models with optimized image preprocessing and user-interface design to track emotional responses continuously.[4] Sajjad et al. explicitly examine FER models for edge vision platforms, including smartphones and low-power boards, and analyse trade-offs between accuracy, latency, and energy consumption.[2] Such work confirms that practical mobile FER is achievable but also exposes deployment constraints regarding energy consumption, model size, and responsiveness.

To ground our survey in practical data, Figure 1 shows example images from the facial-expression dataset used in our project. The samples illustrate the variety of expressions, viewpoints, and lighting conditions that a real-world FER system must handle.

Figure 1. Representative samples from the Young AffectNet HQ dataset used in this work, showing variation in expressions, subjects, and capture conditions.[28]

Most existing mobile FER systems, however, treat training as an offline, centralized process. The typical workflow involves collecting a dataset, training models on servers, and then exporting compressed models to devices. Continuous learning, on-device personalization, and user-driven data collection workflows are comparatively less explored. Furthermore, few works systematically address how user-generated labels—for instance, self-report annotations in an app—should be validated or corrected before being used to update FER models.

3. Federated Learning: Foundations and Vision Applications

3.1. Basic Federated Learning Paradigm

Federated learning is a distributed machine learning framework in which a central server coordinates model updates from a population of clients while keeping raw data local to each client.[5] The canonical algorithm, often referred to as Federated Averaging (FedAvg), alternates between broadcasting a current global model and aggregating locally computed updates. In each communication round, the server sends the latest parameters to a sampled subset of clients; each selected client performs several epochs of local training and sends either gradients or updated parameters back; the server aggregates these updates, typically via weighted averaging, to form the next global model.

This protocol was originally popularized in mobile settings, such as next-word prediction for virtual keyboards, where user data are sensitive and highly personalized. Since then, FL has been extended to a variety of domains including computer vision, speech recognition, healthcare, recommendation, and finance.[5,6] Key challenges in FL include non-independent and non-identically distributed (non-IID) data across clients, variable local dataset sizes, communication constraints, straggling or intermittently available clients, and threats posed by malicious or low-quality participants.[12,13]

A rich ecosystem of FL frameworks and platforms has emerged to support experimentation. Surveys in medical image analysis summarise common design patterns, including cross-silo vs. cross-device FL, secure aggregation, and personalization strategies, and they discuss system-level trade-offs between privacy, performance, and deployment complexity.[5] These overviews provide a useful foundation when designing FL architectures for FER on mobile devices.

 doi:10.20944/preprints202603.2175.v1

3.2. FL in Computer Vision and Medical Imaging

In computer vision, FL has been applied to tasks such as image classification, object detection, and segmentation, with notable emphasis on medical imaging and other privacy-sensitive domains.[5] Multi-institutional FL studies in radiology and pathology have shown that collaboratively trained models can approximate or match performance of centrally trained baselines when communication and aggregation are carefully designed. These works illustrate the feasibility of FL for learning from data siloed across hospitals or organizations that cannot easily share raw patient data.

However, many FL experiments in vision still rely on centrally available datasets partitioned synthetically to simulate clients. Labels are usually assumed to be high-quality, and label noise is either absent or artificially injected according to simple noise models.[14] As a result, issues that arise when labels are collected organically from end users—for example, through mobile applications—are not fully captured. Moreover, annotation workflows and the human factors surrounding data collection are rarely integrated into the FL problem formulation.

3.3. FL for Affective Computing and Emotion Recognition

A smaller but growing body of work studies FL in the context of affective computing. Latif et al. propose one of the earliest frameworks for federated speech emotion recognition (SER), where audio features extracted on devices are used to train a global model without sharing raw speech recordings.[6] Their experiments on SER benchmarks demonstrate that FL can preserve competitive performance while providing stronger privacy guarantees than centralized training.

In visual and multimodal affective computing, several recent studies explore FL-based architectures for emotion recognition. Salman and Busso investigate privacy-preserving personalization for video FER using FL, where a shared global model is adapted locally to individuals under FL constraints.[8] An IEEE study on the development of emotion recognition via federated learning reports results for FER models collaboratively trained across edge devices in realistic network settings.[9] Other work examines micro-expression detection and various multimodal emotion-recognition settings under FL, including combinations of visual and physiological signals.[7,10,11]

These works demonstrate that FL can support distributed emotion-aware services, but they generally adopt pre-labeled datasets or assume that annotation occurs offline, separate from deployed systems. Label noise is sometimes considered as a robustness factor in evaluation, yet there is little explicit modelling of user-generated labels or interactive annotation in deployed FER applications. Consequently, the connection between user-facing data collection and robust FL training remains weak in current affective computing literature.

4. Label Noise and Robust Federated Learning

4.1. Impact of Label Noise in FL

Label noise has long been recognized as a significant challenge in centralized machine learning, and its effects are magnified in FL due to distributed aggregation over heterogeneous clients. Classical surveys review how different types of noise—symmetric, class-dependent, instance-dependent—affect classification performance and how robust losses, regularization, and sample-selection strategies can help.[14,15] As the proportion of incorrect labels increases, models typically exhibit lower accuracy, slower convergence, and a tendency to overfit spurious patterns.

In FL, noise is often not evenly distributed: some clients may maintain mostly clean datasets, whereas others may contain many misannotations, whether due to low engagement, misunderstanding of tasks, or adversarial behaviour.[12,13] This heterogeneity complicates client selection and aggregation strategies. Standard FedAvg treats each client's contribution according to dataset size, which can give disproportionate weight to high-volume but low-quality clients. Moreover, when data distributions and noise patterns differ across clients, naive aggregation may smear out useful signals from high-quality participants.

 doi:10.20944/preprints202603.2175.v1

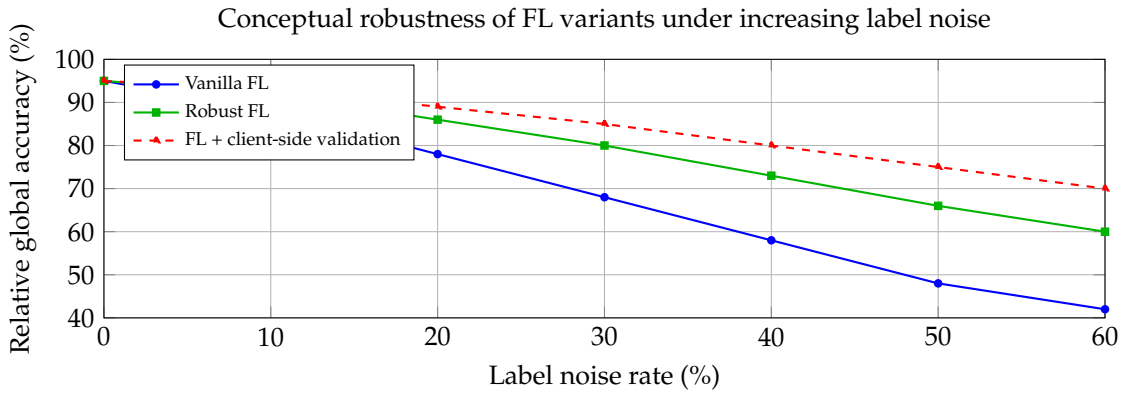

Figure 2. Illustrative effect of increasing label noise on FL accuracy for vanilla FL, robust FL methods, and a hypothetical future system combining robust FL with client-side data validation. Values are conceptual, not empirical.

4.2. Robust FL Frameworks for Noisy Labels

A substantial line of work proposes FL algorithms explicitly designed to cope with noisy labels and unreliable clients. These methods typically operate at one or more of three levels: sample-level reweighting, client-level weighting or pruning, and modified server-side aggregation.[12,13]

Fang and Ye introduce a robust heterogeneous FL framework (RHFL) that combines sample-level and client-level strategies to mitigate the impact of both noisy and statistically heterogeneous clients.[12] Their approach analyses local loss statistics to identify unreliable clients and reweights their contributions accordingly, achieving notable gains across several vision benchmarks. Wu et al. propose FedNoRo, which addresses the joint challenges of class imbalance and label-noise heterogeneity. It uses a two-stage training scheme that first identifies trustworthy clients and classes, then adapts aggregation rules to emphasize cleaner, more informative updates.[13]

Other robust FL methods in the literature employ pruning of noisy clients, confidence-based sample selection, or hybrid centralized/federated schemes to refine global models.[15] Collectively, these works show that algorithmic defences at training and aggregation time can significantly improve robustness to noisy labels, but they do not change how labels are initially collected on devices.

4.3. Client-Side Data Quality Signals for FL

Beyond algorithmic defences during training and aggregation, recent work in related FL domains has begun exploring client-side data-quality signals. In these settings, auxiliary models or metrics evaluate the consistency or alignment of local data and labels, and the resulting quality scores are used to prioritize high-quality data or clients in the global training process.[15] Such approaches suggest that quality estimation can and should happen near the data source, rather than only through aggregate losses at the server.

So far, these methods mostly focus on text or tabular data and provide limited interaction with end users. There is little integration of human-in-the-loop correction or interactive feedback at the client, particularly in FER applications. Nevertheless, the concept of client-side quality metrics illustrates how additional signals beyond gradients can inform federated scheduling and aggregation policies, pointing towards richer quality-aware FL architectures.

5. Crowdsourcing, Data Labeling, and Quality Assurance

5.1. Crowdsourced Annotation Pipelines

Modern machine learning workflows frequently rely on crowdsourcing platforms to create labeled datasets at scale. Crowdsourcing offers access to diverse annotator pools and rapid turnaround, but also introduces substantial variance in annotator expertise and attention.[16] Classical surveys on crowd labeling analyse statistical models that infer latent true labels from multiple noisy annotations,

 doi:10.20944/preprints202603.2175.v1

exploring trade-offs between redundancy, cost, and accuracy. These models have inspired practical systems in which labels from many workers are aggregated according to inferred worker reliability.

Industrial practice also emphasizes engineering controls around the crowdsourcing process. Typical mechanisms include qualification tests, gold-standard questions, consensus checks, hierarchical review by expert annotators, and continuous monitoring of worker performance.[16,17] Such controls are essential for maintaining data quality at scale, particularly in high-stakes domains like healthcare or autonomous driving where mislabels can have serious consequences.

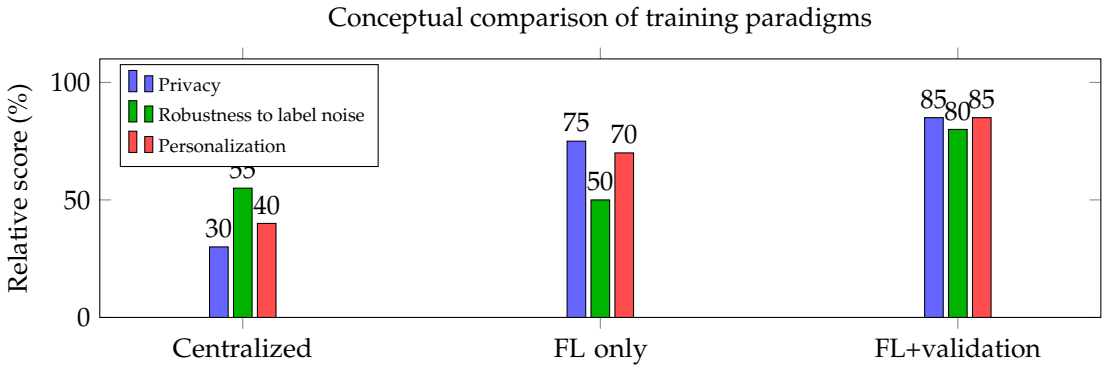

Figure 3. Conceptual comparison of centralized training, federated learning without explicit validation, and FL with client-side validation along key qualitative dimensions. Values are illustrative.

5.2. Multi-Stage Quality Control Frameworks

Recent crowdsourcing research proposes multi-stage quality-control frameworks that insert interventions before, during, and after data collection.[16,17] For example, systems may combine detailed instructions, training tasks, real-time monitoring of responses, and post-hoc auditing of difficult or inconsistent items. Design patterns include dynamically routing ambiguous samples to more experienced annotators, allocating additional redundancy when disagreement is high, and using automatic heuristics or models to flag likely errors.

Zhang et al. propose noise-correction methods that explicitly model annotator error patterns and adjust labels accordingly, improving downstream model performance without requiring perfect annotators.[17] These frameworks highlight a key lesson: effective quality control is not merely a post-processing step but an integral component of the data-collection pipeline. By identifying and correcting issues as early as possible, they reduce wasted annotation effort and improve downstream performance.

5.3. AI-Assisted Labeling and Just-in-Time Interventions

AI models are increasingly used to assist human annotators, not just to consume their labels. In AI-assisted labeling systems, models may pre-label data, suggest candidate annotations, highlight potential inconsistencies, or provide explanations to guide workers.[16] Empirical studies show that such assistance can increase accuracy and consistency, particularly for complex tasks or long labeling sessions, provided that user interfaces are designed carefully.

Zhu et al. present LabelAId, a system that implements just-in-time AI interventions during crowdsourced labeling tasks.[18] It monitors ongoing behavior in real time and triggers interventions—for example, hints, warnings, or additional feedback—when the model detects patterns indicative of confusion or error. These results underscore the importance of timing and user experience design in quality-control mechanisms.

To date, most AI-assisted labeling systems operate in centralized or client–server settings where raw data can be inspected by platform operators. Translating these ideas to FL scenarios requires rethinking what signals can be computed and shared under strict privacy and communication constraints.

 doi:10.20944/preprints202603.2175.v1

6. Mobile and Edge AI: On-Device Inference and Learning

6.1. On-Device Inference Frameworks

On-device machine learning has become mainstream thanks to frameworks such as TensorFlow Lite and related edge libraries.[19] These tools allow developers to retrain models like MobileNet on custom datasets and export quantized models optimized for smartphones and embedded hardware. For FER and related tasks, developers can integrate these models into camera-based applications that process frames locally, avoiding raw image uploads to remote servers.[4]

On-device inference lowers latency and strengthens privacy guarantees, but by itself does not address data collection and training. Most tutorials and examples target single-device inference scenarios, with model updates still performed offline. Integrating on-device inference with FL and client-side validation would enable closed-loop systems in which devices not only infer emotions but also participate in improving models over time using user feedback.

6.2. Edge FL Architectures

Edge FL architectures bring together resource-constrained devices and possibly intermediate edge servers to distribute both computation and communication.[5] Devices may perform limited local training and periodically synchronize with a central server or an edge aggregator. To cope with restricted compute, memory, and energy budgets, strategies such as compressed updates, reduced local epochs, and selective participation are widely studied.

In practice, many FL deployments in industry simulate clients in data centers or on powerful servers rather than running full FL loops on consumer devices. Nonetheless, pilot systems for speech and emotion recognition show that real on-device FL is achievable when models and protocols are carefully designed.[6,7] As these deployments mature, they create an opportunity to experiment with richer client-side logic, including data-validation pipelines and interactive labeling flows.

6.3. Security, Privacy, and Leakage Risks

Although FL keeps raw data local, it is not immune to privacy and security threats. Gradient inversion and related attacks have demonstrated that adversaries with access to model updates can sometimes reconstruct sensitive input features or infer private attributes of clients.[5] This motivates additional techniques such as secure aggregation, differential privacy, and cryptographic methods to limit what can be learned from updates.

Any proposal to transmit client-side data-quality signals must therefore consider their potential leakage. Quality scores, disagreement rates, or other statistics may correlate with sensitive aspects of local data or user behaviour. Designing quality-reporting mechanisms that are informative for training but safe under realistic threat models is a non-trivial challenge. Moreover, validation models themselves may encode biases that disadvantage certain demographic groups, raising fairness concerns in both FER and broader affective computing applications.[1,14]

7. Comparative Analysis: Past, Present, and Future

7.1. Methodological Evolution in FER

From a methodological perspective, FER has evolved from shallow, handcrafted feature pipelines to deep, end-to-end architectures, and is now moving towards multimodal and context-aware models suitable for mobile and federated environments.[1,2] Early systems relied on carefully engineered features (e.g., LBP, Gabor, HOG) and traditional classifiers tuned on small, controlled datasets, with limited robustness and generalization capacity.[1] These pipelines typically separated feature extraction, dimensionality reduction, and classification into distinct stages.

Contemporary FER methods predominantly use deep CNNs and, increasingly, hybrid architectures (CNNs + attention, graph networks, transformers) that jointly learn representations and decision boundaries from large-scale, in-the-wild data.[2] Surveys by Li and Deng and by Sajjad et al. show that modern FER models routinely handle more diverse conditions and more nuanced emotion categories

 doi:10.20944/preprints202603.2175.v1

than classic methods.[1,2] However, they also reveal continuing struggles with label noise, domain shift, and fairness, especially when moving to edge platforms.

Future FER pipelines in FL settings are likely to add a further layer of complexity: models must not only be accurate and efficient but also designed to operate in privacy-constrained, continually evolving environments. This suggests tighter integration of uncertainty estimation, active learning, and user feedback into FER architectures so that client-side validators can intelligently query or correct labels before federated updates are computed.

7.2. Deployment and System Architecture Evolution

Historically, FER deployment followed a centralized architecture: raw images or videos were uploaded to servers where models were trained and evaluated, and sometimes also executed for inference. Such setups offered maximal flexibility but exposed users to privacy risks and required reliable, high-bandwidth connectivity.[1] System design rarely accounted for on-device constraints or intermittent connectivity.

The present landscape is more heterogeneous. On-device inference with mobile-optimized models is now common, and FL has emerged as a viable alternative to centralized training in privacy-sensitive domains.[4–6] For example, real-time mobile FER systems run inference entirely on smartphones, while FL-based speech and multimodal emotion recognition frameworks distribute training across edge devices and servers.[3,6,7,10]

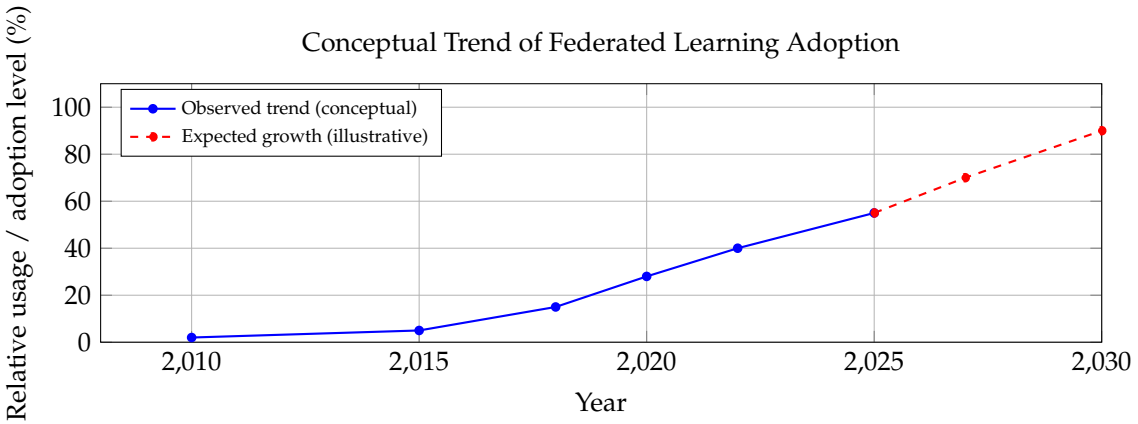

Figure 4. Illustrative evolution of federated learning adoption from early research to projected mainstream usage across domains.

Future architectures for FER in FL settings are likely to be hierarchical and hybrid. Lightweight FER and validation models may run on-device; more resource-intensive adaptation or personalization steps may run on nearby edge servers; and global aggregation and monitoring may occur in the cloud.[5,7] Client-side validation pipelines, discussed later, fit naturally into this multi-layer design by filtering and annotating data at the device layer before it is used to generate FL updates.

7.3. Data Quality and Label-Noise Handling

In early FER and machine-learning pipelines, label quality was often assumed to be high because datasets were curated by experts in controlled settings. Label noise was acknowledged but typically addressed informally (e.g., via manual cleaning) or through generic regularization.[1,14] In FL, initial work also made simplifying assumptions about clean labels, focusing more on non-IID data distributions and communication constraints.[5]

The present state-of-the-art treats label noise as a first-class concern. Comprehensive surveys on label noise analyse its effects and propose robust losses, sample-selection strategies, and reweighting schemes for centralized learning.[14,15] In FL, RHFL and FedNoRo represent the current frontier: they incorporate loss-based client weighting, class-aware aggregation, and multi-stage training to mitigate

 doi:10.20944/preprints202603.2175.v1

noisy and heterogeneous clients.[12,13] These methods significantly improve robustness compared to vanilla FedAvg under synthetic noise scenarios.

Looking ahead, robust FL must be complemented by proactive data-quality management at the client. Rather than only down-weighting noisy clients or samples after the fact, future systems will need mechanisms that prevent low-quality data from entering the training pipeline in the first place. This is where client-side validation, inspired by crowdsourcing quality-control and AI-assisted labeling, becomes critical: on-device models and user interactions can be used to detect and correct mislabels before they contaminate local training data.[16–18]

7.4. Human-in-the-Loop and User Experience

Historically, most machine-learning systems treated labeling as a one-time, offline process, with limited attention to the user experience of annotators. FER datasets were often prepared by trained coders or small groups of workers, and end users of emotion-aware applications had little or no control over how their data were labeled or used.[1,16]

Today, crowdsourcing research emphasizes human factors in labeling workflows, including instructions, incentives, cognitive load, and feedback mechanisms.[16,18] Systems like LabelAId show that just-in-time AI interventions can improve label quality and worker learning by offering targeted guidance when confusion is detected.[18] At the same time, affective computing applications increasingly need to respect user autonomy and privacy, especially on personal devices such as smartphones and wearables.

Future FER + FL systems will likely blur the lines between annotators and end users: users may be asked to confirm or adjust labels for their own data during normal app usage. This requires carefully designed interfaces that minimize burden, avoid over-questioning, and explain disagreements between model predictions and user labels in an understandable way.[2,18] Human-in-the-loop loops must be designed with not only accuracy but also user trust, transparency, and fairness in mind.

7.5. Cross-Domain Comparison

Table 1 summarizes the evolution across key axes relevant to your project domain: FER methodology, deployment architecture, data-quality handling, and human-in-the-loop involvement.

Table 1. Evolution of FER + FL Systems Across Past, Present, and Future.

Axis	Past (pre-deep / centralized)	Present (deep FER + early FL)	Future (client-side validation in FL)
FER methodology	Handcrafted features + SVM/HMM on small, controlled datasets; limited robustness.[1]	Deep CNNs, attention and hybrid models on large in-the-wild datasets; edge-optimized backbones.[1,2,4]	Multimodal, uncertainty-aware models tightly coupled with on-device validators and active learning.
Deployment architecture	Centralized training and inference on servers; raw images often uploaded.[1]	On-device inference with mobile CNNs; FL for training in select domains (medical imaging, SER, FER prototypes).[3,5,6,9]	Hierarchical edge/cloud FL with local validation pipelines and selective, privacy-preserving reporting of quality signals.
Data-quality handling	Assumed expert-labeled, mostly clean datasets; ad-hoc manual cleaning.[1]	Robust centralized training; robust FL algorithms (RHFL, FedNoRo) mitigate noisy clients during training.[12–14]	Client-side validation filters and corrects labels pre-training, combined with robust FL and secure aggregation of quality metrics.[15,16,18]
Human-in-the-loop	Limited to offline dataset creation by experts or crowd workers; end users rarely involved.[16]	Crowdsourcing platforms with multi-stage quality-control and occasional AI assistance.[17,18]	End users participate in ongoing labeling/validation via unobtrusive prompts; UX and fairness central to system design.

 doi:10.20944/preprints202603.2175.v1

8. Future Directions and Open Challenges

8.1. Formal Models of Client-Side Validation in FL

A first line of future work is to develop formal models and theoretical analyses of client-side validation policies in FL. Such models would link validation accuracy, confidence thresholds, user response patterns, and resulting effective label-noise rates to standard FL convergence guarantees.[12, 14] They could draw on existing theory for noisy labels and FL convergence, extended to scenarios where data distributions are actively shaped by on-device filtering and user interactions.

Understanding these dynamics would clarify when simple heuristics, such as confidence thresholding, yield meaningful gains and when more sophisticated strategies like active learning or uncertainty sampling are required. Analytical results could then inform the design of practical validation policies and provide guidance on trade-offs among validation frequency, user burden, and final model performance.

8.2. Human–AI Interaction and UX for FL Data Collection

Another critical direction involves human–AI interaction and user experience (UX) in FL-based FER applications. When end users serve as annotators for their own data, the design of prompts, feedback, and explanations becomes central. Studies of crowdsourcing and AI-assisted labeling indicate that task framing, perceived burden, incentives, and feedback modalities strongly influence both engagement and data quality.[16,18]

Designing FER applications that request clarification only when necessary, provide clear explanations for model disagreement, and respect user attention budgets is an open challenge. Mixed-methods research that combines quantitative FL metrics with qualitative user studies will be essential to ensure that client-side validation pipelines are usable and trustworthy rather than intrusive or confusing.

8.3. Multimodal and Context-Aware Validation

FER increasingly moves beyond static facial images to multi-modal and context-rich settings that incorporate audio, text, physiological signals, and environmental context.[6,7,10] Client-side validation could similarly leverage multiple modalities to judge whether a particular label is plausible, for example by cross-checking visual expressions with speech prosody or self-reported mood.

In federated settings, multimodal data may reside on different devices or sensors, or be subject to different privacy constraints. Exploring how to perform coherent validation across modalities, perhaps using lightweight fusion models or cross-modal consistency checks, is an important and relatively unexplored topic.

8.4. Privacy, Security, and Fairness Considerations

Designing privacy-preserving quality-reporting mechanisms is a major open challenge. Quality scores, disagreement rates, or user interaction logs might themselves leak information about sensitive attributes or behaviour if transmitted naively to a server.[5] Integrating secure aggregation, differential privacy, or local obfuscation into the reporting of quality signals is therefore an important research problem.

Fairness must also be considered. Validation models might exhibit demographic biases, for instance by misjudging expressions for certain groups, which could lead to systematically discarding or questioning their labels more often.[1,14] Emotion recognition already faces criticism regarding cross-cultural validity and potential misuse; any additional validation mechanisms must be audited and corrected to avoid reinforcing unfair treatment of particular populations.

8.5. From Simulation to Real-World Deployments

Many FL and robust-FL studies evaluate algorithms on centrally available datasets partitioned into synthetic clients under simple non-IID schemes.[12,14] While such experiments are useful for benchmarking, they do not capture the complexities of real user populations, device heterogeneity,

 doi:10.20944/preprints202603.2175.v1

and long-term behaviour. The same limitation applies to simulated label noise, which rarely reflects the structure and causes of noise in real applications.

End-to-end pilots that integrate mobile FER applications, on-device validation logic, and FL backends are needed to test whether proposed methods hold up in practice. Such deployments would provide valuable insights into communication overhead, user engagement, edge-case failures, and the stability of learning dynamics over extended periods.[6,9] They could also reveal unexpected interactions between UX decisions and FL performance, guiding iterative refinement of both algorithms and interfaces.

9. Federated Learning Frameworks and Deployment Choices

In addition to algorithms, the practical success of FL-based FER systems depends heavily on the choice of framework and deployment stack.[19–21,23,24] This section compares representative FL frameworks and on-device ML runtimes, focusing on their suitability for mobile emotion-recognition scenarios and for implementing client-side validation pipelines.

9.1. Categories of Federated Learning Tooling

Broadly, FL tooling can be grouped into two layers:

- **Server–orchestrated FL frameworks**, which provide infrastructure for coordinating training across many clients (e.g., Flower, TensorFlow Federated, FedML, PySyft).[20,21,23,24]
- **On-device ML runtimes**, which execute models (and sometimes lightweight updates) locally on mobile or embedded hardware (e.g., TensorFlow Lite and companion libraries).[19]

In a realistic FER-on-mobile deployment, both layers are needed: a server-side FL framework to orchestrate federated training, and an on-device runtime such as TFLite to run FER and validation models efficiently on user devices.

9.2. Flower: Framework-Agnostic Orchestration

Flower is an open-source FL framework designed to be framework-agnostic and highly configurable.[20] It supports PyTorch, TensorFlow, JAX, and other backends, exposing simple client and server APIs so that existing training loops can be federated with minimal changes. The core abstractions are the `Client` (which implements local training and evaluation) and the `Server` (which implements the FL strategy and aggregation rules).

For FER research, Flower is attractive because:

- It is easy to integrate with common deep-learning stacks (e.g., PyTorch-based FER models), allowing rapid prototyping of FL variants without rewriting models.[20]
- It scales from small simulations on a single machine to large-scale, distributed experiments with thousands of clients, which is useful when emulating mobile populations.[20]
- It exposes strategy hooks where researchers can plug in noise-aware aggregation, client selection, or custom validation logic, aligning well with robust FL and client-side validation experiments.

However, Flower itself does not provide on-device runtimes; for real smartphone deployments, it is typically combined with mobile libraries such as TFLite or platform-specific SDKs.

9.3. TensorFlow Federated: TensorFlow-Native FL

TensorFlow Federated (TFF) is Google's FL framework built around the TensorFlow ecosystem.[21,22] It provides two layers: a high-level federated learning API (`tff.learning`) for plugging Keras/TensorFlow models into ready-made FL algorithms, and a low-level federated core for expressing custom federated computations.[21]

TFF is a strong choice when:

- The FER models and preprocessing are already implemented in TensorFlow, and one wants to minimize stack fragmentation.[22]

 doi:10.20944/preprints202603.2175.v1

- The focus is on algorithm research and simulation rather than immediate deployment, since TFF currently targets simulation and specialized runtimes more than full end-user mobile apps.
- Formal reasoning about federated computations is important, as TFF's strongly-typed federated core makes communication patterns explicit.[21]

Compared to Flower, TFF offers deeper integration with TensorFlow and a rich functional language for federated computations, but it is less flexible for PyTorch-based workflows and less focused on heterogeneous production environments.

9.4. FedML and PySyft: Research Ecosystems and Privacy Enhancements

FedML is an open research library and benchmark suite that supports a broad ecosystem of FL tasks and settings, including cross-device, cross-silo, and single-machine simulation.[23] It offers modules for computer vision, NLP, graph learning, and IoT, along with built-in baselines and datasets. For FER and affective computing, FedML is particularly useful when:

- One needs standardized benchmarks and datasets to compare novel robust-FL or validation algorithms against published baselines.[23]
- Experiments span multiple domains (e.g., combining image-based FER with speech or physiological signals) and require consistent tooling.
- There is interest in moving from small-scale prototypes to more realistic, distributed evaluations with varied network and device conditions.[23]

PySyft, developed by OpenMined, emphasizes privacy-preserving computation, including FL, differential privacy, and secure aggregation.[24,25] It extends PyTorch and TensorFlow with primitives for remote execution, encrypted tensors, and privacy-preserving operations. PySyft is most appropriate when:

- The primary research question concerns strong privacy guarantees (e.g., exploring combinations of FL, differential privacy, and secure aggregation for sensitive emotion data).[24]
- One wants to prototype secure computation patterns (such as additive secret sharing) on top of FL, not just standard FedAvg-like training.[25]

For a student project, FedML and PySyft can be seen as complementary to Flower and TFF: FedML provides breadth of research tasks and benchmarks, while PySyft focuses on advanced privacy mechanisms that might be layered on top of simpler FL strategies.

9.5. TensorFlow Lite: On-Device Inference and Lightweight Training

TensorFlow Lite (TFLite) is a production-ready framework for running ML models on mobile and embedded devices, optimized for low latency, small footprint, and offline operation.[19,26] It supports Android, iOS, embedded Linux, and microcontrollers, and is widely used for tasks such as image classification, object detection, gesture recognition, and speech commands.[19,27]

In the context of FL-based FER, TFLite plays a different role from Flower or TFF:

- It is primarily an *on-device runtime* for inference (and limited on-device training), not an orchestration framework.[19]
- It is well suited for deploying FER and validation models inside a mobile app, supporting real-time analysis of camera frames and just-in-time feedback to the user.[26]
- Experimental on-device training support makes it possible to perform small local adaptation steps (e.g., personalization of FER models) before sending updates back to an FL server.

In a complete system, a TFLite-based FER model would run on the phone, while Flower, TFF, or FedML would coordinate periodic federated training rounds on a backend server.

9.6. Qualitative Comparison

Table 2 summarizes key qualitative differences between these tools for the purposes of FER and client-side validation.

 doi:10.20944/preprints202603.2175.v1

Table 2. Qualitative Comparison of FL Frameworks and On-Device Runtimes for FER.

Tool	Primary role	Framework support	Strengths for FER/validation	Typical usage scenario
Flower[20]	Server-side FL orchestration	PyTorch, TensorFlow, JAX (framework-agnostic)	Easy to wrap existing training loops; flexible strategy hooks for robust FL and client selection	Rapid FL prototyping and research with existing Python FER models; small to large-scale simulations.
TensorFlow Federated[21,22]	Server-side FL orchestration and simulation	TensorFlow/Keras-native	Strong integration with TF; expressive federated core for custom algorithms	Algorithm research when models are in TensorFlow and experiments run in simulated environments.
FedML[23]	Research library and benchmarks for FL	Multiple backends; focus on CV, NLP, graphs, IoT	Standardized datasets and baselines; support for cross-device and cross-silo FL	Comparative studies of FL algorithms across tasks, including FER-related vision workloads.
PySyft[24,25]	Privacy-preserving FL and secure computation	Extends PyTorch/TF with privacy primitives	Differential privacy, secure aggregation, and remote execution for sensitive data	Experiments where strong privacy guarantees for emotion data outweigh ease of deployment.
TensorFlow Lite[19,26]	On-device inference (and light training) runtime	Runs converted TensorFlow models on mobile/edge	Low-latency FER inference and validation on phones; offline operation; supports UI integration	Production mobile apps performing FER and client-side validation; combined with a server-side FL framework.

9.7. Which Framework for Which Situation?

For a FER-on-mobile project that includes federated training and client-side validation, different tools are preferable in different situations:

- **Course projects and fast experimentation:** Flower is often the most convenient choice when FER models are implemented in PyTorch or mixed frameworks, because its APIs are simple and it does not impose a fixed model representation.[20] It is therefore ideal for demonstrating robust FL under label noise or for prototyping validation strategies using simulated clients.
- **TensorFlow-centric research:** When the end-to-end pipeline (training, preprocessing, and deployment) is already in TensorFlow, TFF provides a coherent way to express and analyze federated computations without leaving the ecosystem.[21,22] This is attractive for groups that already use TensorFlow for FER and plan to deploy with TFLite.
- **Large-scale algorithm benchmarks:** FedML is well suited for systematic comparisons across many FL algorithms, datasets, and tasks.[23] It is a good fit when the goal is to situate a new robust-FL or validation method in a broader research landscape rather than to build a single integrated application.
- **Privacy-critical deployments:** PySyft is the right starting point when strict privacy guarantees are central, for example in clinical or mental-health emotion-recognition studies where combining FL with differential privacy or secure multi-party computation is essential.[24,25]

 doi:10.20944/preprints202603.2175.v1

- **Production mobile apps:** TFLite is the standard choice for running FER and validation models on-device in Android/iOS applications.[19,26] In this case, the FL orchestration layer (Flower, TFF, or FedML) runs on the server side, while TFLite handles local inference and any lightweight adaptation steps.

In summary, there is no single "best" FL framework for all FER scenarios. For the system envisioned in this paper—mobile FER with client-side validation and federated training—a pragmatic stack is to use *Flower or TFF* on the server for orchestration and *TFLite* on devices for efficient on-device FER and validation. Researchers focused on benchmarking or privacy enhancements can layer FedML or PySyft on top of this basic stack as their experimental needs evolve.

10. Conclusions

This survey has connected research on facial emotion recognition, federated learning, robust training under label noise, and crowdsourcing-based data quality assurance, with a particular emphasis on FER applications on mobile and edge devices. Deep CNNs and efficient architectures have made FER accurate and fast enough for deployment on smartphones and embedded hardware,[1,2,4] while FL provides a means to update models without centralizing raw user data.[5,6] At the same time, user-generated labels—especially for subjective phenomena like emotion—are often noisy and heterogeneous across clients, posing serious challenges for FL convergence and robustness.[12–14]

Existing robust FL algorithms provide powerful tools to mitigate noisy labels and unreliable clients, yet they predominantly operate after data have been collected and processed locally.[12,13] Crowdsourcing research, in contrast, demonstrates the effectiveness of just-in-time interventions and multi-stage quality-control frameworks that intervene during data collection.[16–18] By bringing these ideas together, we argue for the development of client-side validation pipelines that run on users' devices, combining lightweight on-device models, human feedback, and quality scoring to improve the data that feed federated training.

Realizing this vision will require interdisciplinary work across machine learning, human–computer interaction, privacy and security, and ethics. If successful, client-side validation in FL-based FER systems could enable emotion-aware services that are not only accurate and personalized, but also more transparent, fair, and respectful of user privacy.

The proposed federated learning-enhanced face recognition attendance system aligns seamlessly with prior centralized frameworks in educational applications. It can be applied to automated classroom attendance logging systems [29] by enabling local feature extraction and matching on edge devices through collaborative model updates, enhancing privacy and scalability across institutions. Similarly, our federated approach extends facial recognition systems for education [30] by supporting real-time expression tracking without centralized data storage. Furthermore, integrating federated aggregation with student emotion recognition for e-learning [31] improves concentration metric accuracy via diverse, privacy-preserving datasets from multiple learners.

References

1. S. Li and W. Deng, "Deep facial expression recognition: A survey," *IEEE Transactions on Affective Computing*, 2020.
2. M. Sajjad *et al.*, "A comprehensive survey on deep facial expression recognition: challenges, applications, and future guidelines," *Alexandria Engineering Journal*, vol. 64, 2023.
3. M. Suk and B. Prabhakaran, "Real-time mobile facial expression recognition system," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2014.
4. H. Li *et al.*, "Real-time facial affective computing on mobile devices," *Sensors*, vol. 20, no. 3, 2020.
5. H. Guan *et al.*, "Federated learning for medical image analysis: A survey," *Medical Image Analysis*, 2024.
6. S. Latif *et al.*, "Federated learning for speech emotion recognition," in *Proc. ACM/IEEE CPS-IoT Week Workshop*, 2020.
7. Y. Authors, "AFLEMP: Attention-based federated learning for emotion recognition using multi-modal physiological data," *Biomedical Signal Processing and Control*, 2024.
8. A. N. Salman and C. Busso, "Privacy preserving personalization for video facial expression recognition using federated learning," in *Proc. ACM Conf. on Multimodal Interaction*, 2022.

 doi:10.20944/preprints202603.2175.v1

9. K. Authors, "Development of emotion recognition via federated learning," in *Proc. IEEE Conf.*, 2024, IEEE Xplore Doc. 10829216.
10. N. Simic *et al.*, "Enhancing emotion recognition through federated learning: A multimodal approach with convolutional neural networks," 2024.
11. A. Authors, "Micro expression detection using federated learning," *International Journal of Computer Science Trends and Technology*, vol. 12, no. 2, 2024.
12. X. Fang and M. Ye, "Robust federated learning with noisy and heterogeneous clients," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.
13. N. Wu *et al.*, "FedNoRo: Towards noise-robust federated learning by addressing class imbalance and label noise heterogeneity," in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2023.
14. B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2014.
15. M. Authors, "Imbalanced classification with label noise: A systematic review and comparative analysis," *ICT Express*, 2025.
16. Y. Zhang *et al.*, "Learning from crowdsourced labeled data: A survey," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 543–576, 2016.
17. J. Zhang *et al.*, "Improving label quality in crowdsourcing using noise correction," in *Proc. ACM Int. Conf. on Information and Knowledge Management (CIKM)*, 2015.
18. Z. Zhu *et al.*, "LabelAId: Just-in-time AI interventions for improving human labeling quality and domain knowledge in crowdsourcing systems," in *Proc. CHI Conf. on Human Factors in Computing Systems*, ACM, 2024.
19. Google AI Edge, "Image classification with TensorFlow Lite Model Maker," 2024. [Online]. Available: https://ai.google.dev/edge/lite/libraries/modify/image_classification
20. Flower, "What is Federated Learning? Flower framework tutorial series," 2024. [Online]. Available: https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html
21. TensorFlow, "TensorFlow Federated: an open-source framework for machine learning and other computations on decentralized data," 2021. [Online]. Available: https://www.tensorflow.org/federated
22. TensorFlow, "Federated learning with TensorFlow Federated (TFF)," tutorial, 2024. [Online]. Available: https://www.geeksforgeeks.org/deep-learning/federated-learning-with-tensorflow-federated/
23. C. He *et al.*, "FedML: A research library and benchmark for federated learning," arXiv preprint arXiv:2007.13518, 2020.
24. OpenMined, "Federated learning with PySyft: privacy-preserving AI models," 2026. [Online]. Available: https://www.nivalabs.ai/blogs/federated-learning-with-pysyft-privacy-preserving-ai-models
25. A. Rieke, "Federated learning in 10 lines of code with PySyft," OpenMined blog, 2025. [Online]. Available: https://openmined.org/blog/fl-in-10-lines-of-code-with-pysyft/
26. Tops Infosolutions, "TensorFlow Lite use cases: on-device machine learning," 2024. [Online]. Available: https://www.topsinfosolutions.com/blog/tensorflow-lite-on-device-ml-use-cases/
27. GeeksforGeeks, "Introduction to TensorFlow Lite," 2021. [Online]. Available: https://www.geeksforgeeks.org/deep-learning/introduction-to-tensorflow-lite/
28. V. Fomenko, "Young AffectNet HQ," Kaggle, 2022. [Online]. Available: https://www.kaggle.com/datasets/vfomenko/young-affectnet-hq
29. L.B. Krithika, S. Kshitish, and M.R. Kishore, "Automated facial attendance logger for students," *IOP Conference Series: Materials Science and Engineering*, vol. 263, no. 4, p. 042022, 2017. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/263/4/042022
30. L.B. Krithika, K. Venkatesh, S. Rathore, and M. Harish Kumar, "Facial recognition in education system," *IOP Conference Series: Materials Science and Engineering*, vol. 263, no. 4, p. 042021, 2017. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/263/4/042021
31. L.B. Krithika and G.G. Lakshmi Priya, "Student emotion recognition system (SERS) for e-learning improvement based on learner concentration metric," *Procedia Computer Science*, vol. 85, pp. 767–776, 2016. [Online]. Available: https://doi.org/10.1016/j.procs.2016.05.264