Formal Modeling and Verification of a Federated Learning Protocol for Skin Lesion Segmentation

Antônio Vinicius de M. Rodrigues

Department of Computer Science - University of Brasília (UnB)
Brasilia, Brazil

Abstract. *The accurate and early detection of malignant skin cancers, such as melanoma, is a critical factor in improving patient survival rates. Computer-Aided Diagnosis (CAD) systems, powered by Convolutional Neural Networks (CNNs), have demonstrated success in the task of lesion segmentation, which is essential for diagnosis. However, the development of robust CNNs requires vast and diverse datasets, which are fundamentally constrained by patient data privacy regulations. Federated Learning (FL) has emerged as a transformative distributed learning paradigm that enables collaborative model training without sharing sensitive data. Despite its privacy advantages, FL introduces significant distributed systems challenges, including managing statistical heterogeneity, communication overhead, and client failures. This report presents a comprehensive formal verification of a synchronous FL orchestration protocol designed for medical image segmentation. The complete FL orchestration protocol, encompassing a central server, multiple clients, local training timers, communication timeouts, and failure mechanisms is modeled as a network of Transition System. Using the UPPAAL model checker, a comprehensive set of temporal properties is specified and verified using Computation Tree Logic. The verification results formally prove that the proposed protocol is free from deadlocks (Safety) and guarantees eventual model aggregation (Liveness) even in the presence of client timeouts and failures.*

1. Introduction

Skin cancers represent the most commonly diagnosed malignancies worldwide, posing a significant and growing public health challenge. According to the International Agency for Research on Cancer (IARC), over 1.5 million new cases of skin cancer were reported in 2022 alone [for Research on Cancer IARC 2023]. Within this figure, melanoma, the most aggressive form of skin cancer, accounted for approximately $330,000$ cases and was responsible for $60,000$ fatalities. While the survival rate for localized melanoma is approximately 99%, this figure drops significantly once the disease metastasizes, underscoring the vital importance of early detection [Mirikharaji et al. 2023]. Consequently, rapid and precise diagnosis is the primary determinant of patient outcomes, driving the demand for advanced technologies capable of supporting early detection.

Dermoscopy, a non-invasive imaging technique that uses magnification and specialized illumination, has become a standard tool in clinical practice. It enhances a dermatologist's diagnostic accuracy beyond what is possible with naked-eye inspection by revealing subsurface skin structures [Celebi et al. 2015]. Despite its utility, the interpretation of dermoscopic images remains a subjective task, fraught with considerable inter-observer variability even among experienced practitioners [Holmes et al. 2018]. This subjectivity, compounded by a global shortage of dermatological expertise and long wait times for specialist consultations, creates a compelling need for objective, reliable, and scalable diagnostic aids. As a result, research efforts have increasingly focused on the development of Computer-Aided Diagnosis (CAD) systems. These systems are not intended to replace clinicians but to function as decision-support tools, providing rapid, reproducible, and objective analysis of lesion characteristics to augment the clinician's judgment [Mahbod et al. 2018].

Deep Learning (DL), a subfield of machine learning that employs artificial neural networks to learn complex representations from data [Hesamian et al. 2019], has become a key approach for enhancing CAD systems in dermoscopic image analysis. DL methods, particularly Convolutional Neural Networks (CNNs), can automatically extract discriminative features and capture subtle variations in color, texture, and structure that characterize skin lesions [Jahanifar et al. 2019, Hesamian et al. 2019]. These models also improve robustness to common imaging artifacts such as hair, reflections, and uneven illumination, thereby increasing diagnostic reliability [López-Labraca et al. 2022]. Although CNNs were originally designed for image classification, their architecture can be adapted for pixel-wise prediction tasks such as segmentation, which is essential for accurately isolating lesions or specific attributes within dermoscopic images [Mirikharaji et al. 2023]. This capability is critical for enabling precise and objective analysis in modern CAD pipelines.

Despite the proven potential of CNN, a fundamental data-privacy paradox has severely limited its real-world clinical adoption. CNN models are "data-hungry," requiring large, diverse, and well-annotated datasets to learn complex patterns and achieve a high degree of generalizability [Foahom Gouabou et al. 2022]. However, the medical images required to build these models are highly sensitive Protected Health Information (PHI). Their collection, storage, and use are governed by stringent legal and ethical frameworks, such as the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR) [Conduah et al. 2025].

Federated Learning (FL) has emerged as a transformative distributed systems paradigm engineered to resolve this impasse [McMahan et al. 2023]. First proposed by Google, the core idea of FL is to "move the model, not the data.". In an FL architecture, multiple distributed clients (e.g., hospitals) collaboratively train a global model without ever exchanging or sharing their local, private data [Yurdem et al. 2024]. Each client trains the model locally, and only the updated model parameters are sent to a central server for aggregation. The server computes an improved global model (e.g., via federated averaging) and broadcasts it back to the clients for the next training round. This approach, particularly "horizontal FL" where clients share the same task but different patient samples, is ideally suited for multi-institutional medical image analysis [Yurdem et al. 2024].

While solving the privacy challenge, FL introduces a significant distributed systems reliability problem. An FL system is a complex, time-sensitive, and stateful protocol, subject to network delays, client dropouts, and clients with slower network connections (stragglers) [Yurdem et al. 2024]. To build trustworthy medical AI, mathematical guarantees of correctness must be provided through formal verification. Formal verification exhaustively explores the entire state space of a system model to mathematically prove or disprove its adherence to critical properties [Kim et al. 2023].

This article integrates the applied domain of CNNs with the rigorous domain of formal methods, fulfilling the requirements for a graduate-level software engineering assignment. The formal verification of a robust, round-based FL orchestration protocol designed for skin lesion segmentation models is proposed using Transition Systems (TS).

2. System Description and Requirements

The system under analysis is a centralized, synchronous FL orchestration protocol implementing the Federated Averaging (FedAvg) algorithm [Sun et al. 2021]. FedAvg proceeds in discrete communication rounds, each consisting of: (i) initialization and broadcast of the global model by the server; (ii) local model training on private client datasets; (iii) transmission of the resulting updates to the server; and (iv) aggregation of the received updates to generate the next global model [Sun et al. 2021]. For the purposes of formal modeling, the system comprises a single *Server* and a fixed set of three *Clients*. All clients participate in every round and are subject to real-time constraints, enabling verification through TS.

2.1. Protocol flow

1. **Round Start & Broadcast:** The *Server* initializes a round, resets counters, and starts a global round-timer. It then sends a *broadcast* signal to all clients.
2. **Local Training:** Upon receiving the signal, *Client* i enters a *Training* state, which is time-bounded by *MAX_TRAIN_TIME*.
3. **Failure Handling (Local):** If *Client* i's training time exceeds the limit, it transitions to a *Failed* state.
4. **Update and Retransmission:** Upon successful local training, the client attempts to send its update. This communication is time-bounded by *SEND_TIMEOUT* and includes a maximum number of *MAX_RETRIES* before the client transitions to the *Failed* state.
5. **Aggregation:** The *Server* waits for updates in a *Waiting* state. Aggregation is triggered by the earliest of two conditions: a) **Full Success**: All clients successfully submit updates, or b) **Global Timeout**: The server's global round-timer (*COMM_TIMEOUT*) expires, guaranteeing progress.

2.2. Formal Requirements

- **Deadlock Absence:** The system must never reach a state from which no further action or time progression is possible.
- **Eventual Progress:** The *Server* must be guaranteed to eventually reach the *Aggregating* state, even if client failures occur.
- **Bounded Operation:** Local training time (*MAX_TRAIN_TIME*) and individual send attempts (*SEND_TIMEOUT*) must be strictly time-bounded.
- **Bounded Retries:** Retransmission attempts must be bounded by *MAX_RETRIES* to prevent clients from consuming resources indefinitely.
- **State Consistency:** A client must not simultaneously be in two conflicting states (e.g., *Training* and *Sending*).

3. UPPAAL Model

3.1. Global Declarations

Global declarations establish the constants, variables, and synchronized communication channels.

Table 1. Global Declarations

Name	Type	Initial value	Description
N_CLIENTS	*const int*	3	Total number of clients.
MAX_TRAIN_TIME	*const int*	20	Maximum time for local training.
COMM_TIMEOUT	*const int*	10	Max server waiting time.
SEND_TIMEOUT	*const int*	5	Max time for single send attempt.
MAX_RETRIES	*const int*	2	Max retransmissions allowed.
models_received	*int*	0	Counter for successful client updates.
broadcast_start	*broadcast chan*	-	Channel for server to start client i.
send_update	*chan*	-	Channel for client i to send updates.

3.2. The Server

The Server TS models the behavior of the central aggregation node, acting as the federation's controller. Its behavior is defined by the following states and transitions, as illustrated in Figure 1:

1. **Idle:** The Server waits to begin a round. This state is marked *committed*, ensuring immediate transition to initialization.
2. **Broadcasting:**. The Server iterates through the list of clients, sending the *broadcast_start* signal to each. This phase is modeled as a *committed* state, ensuring atomicity: no other transitions or time delays can occur until all broadcasts are completed.
3. **Waiting:**
   - *Invariant:* `x_round_timer <= COMM_TIMEOUT`.
   - *Description:* The critical state where the Server awaits client updates. The invariant enforces that the system cannot remain in this state beyond the defined timeout.
   - *Transitions:*
     - *Receive Update:* On channel `send_update?`, increment `models_received`.
     - *Early Success:* If `models_received == N_CLIENTS`, transition to `Aggregating`.
     - *Timeout:* If `x_round_timer == COMM_TIMEOUT`, transition to `Aggregating`.
4. **Aggregating:** The state representing the computational aggregation of weights.

3.3. The Client

The Client TS models the dynamics of a hospital node, capturing the non-deterministic nature of computation and network transmission. Its behavior is defined by the following states and transitions, as shown in Figure 2:

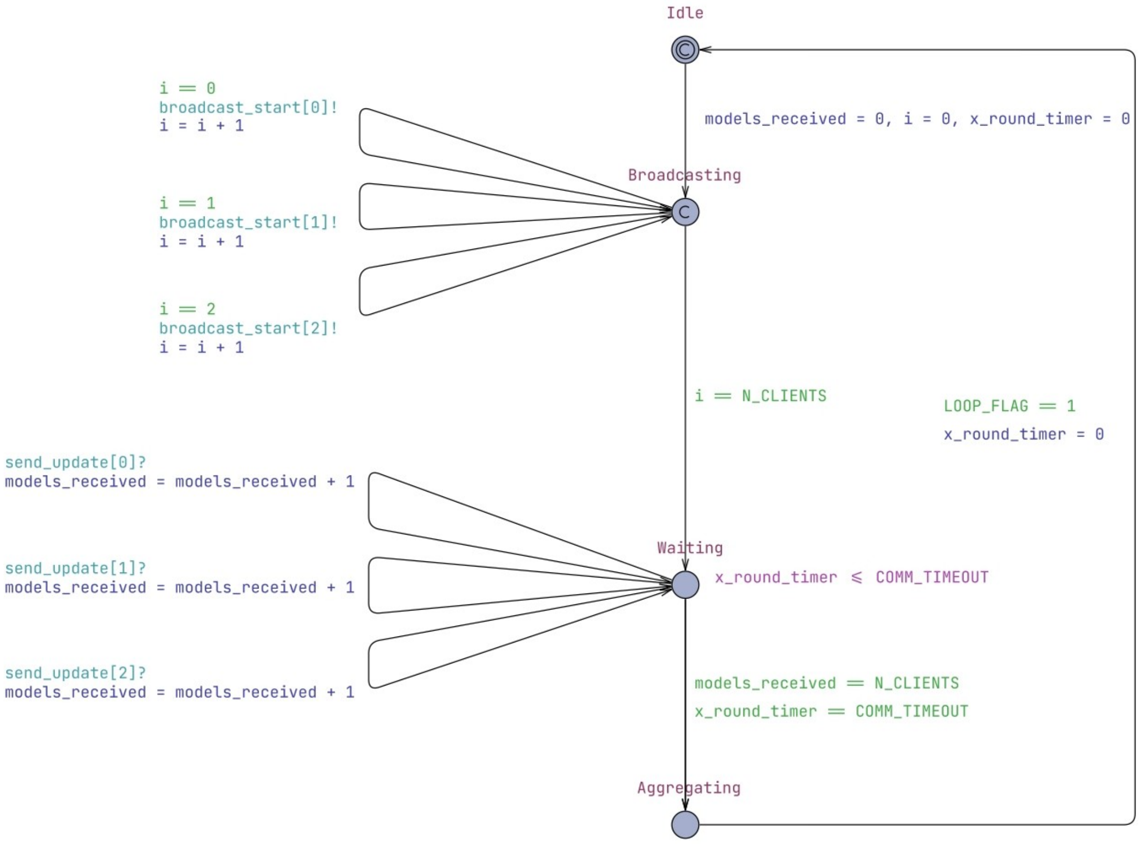

Idle
models_received = 0, i = 0, x_round_timer = 0
i == 0
broadcast_start[0]!
i = i + 1
Broadcasting
i == 1
broadcast_start[1]!
i = i + 1
i == 2
broadcast_start[2]!
i = i + 1
i == N_CLIENTS
LOOP_FLAG == 1
x_round_timer = 0
send_update[0]?
models_received = models_received + 1
Waiting
x_round_timer ≤ COMM_TIMEOUT
send_update[1]?
models_received = models_received + 1
send_update[2]?
models_received = models_received + 1
models_received == N_CLIENTS
x_round_timer == COMM_TIMEOUT
Aggregating

Figure 1. TS representing the Server logic. Source: Author's own production.

1. **Idle:** Waiting for the broadcast signal.
2. **Training:**.
   - *Invariant:* `x_train <= MAX_TRAIN_TIME`.
   - *Description:* Represents the local training process. The invariant simply sets a time limit for this task, ensuring the client doesn't stay in the training phase indefinitely.
   - *Transitions:*
     - *Completion:* If `x_train <= MAX_TRAIN_TIME`, transition to `Sending`.
     - *Failure:* If `x_train == MAX_TRAIN_TIME`, transition to `Failed`.
3. **Sending:**
   - *Invariant:* `x_send <= SEND_TIMEOUT`.
   - *Description:* Represents the data transmission. The invariant limits how long the client tries to send the update before considering it a timeout.
4. **Retrying:**
   - *Description:* An *urgent* state entered if the `Sending` state times out. It checks if `retry_count < MAX_RETRIES`. If true, the client returns to `Sending`; otherwise, it transitions to `Failed`.
5. **Failed:** A state signaling that a client has dropped out for the current round..

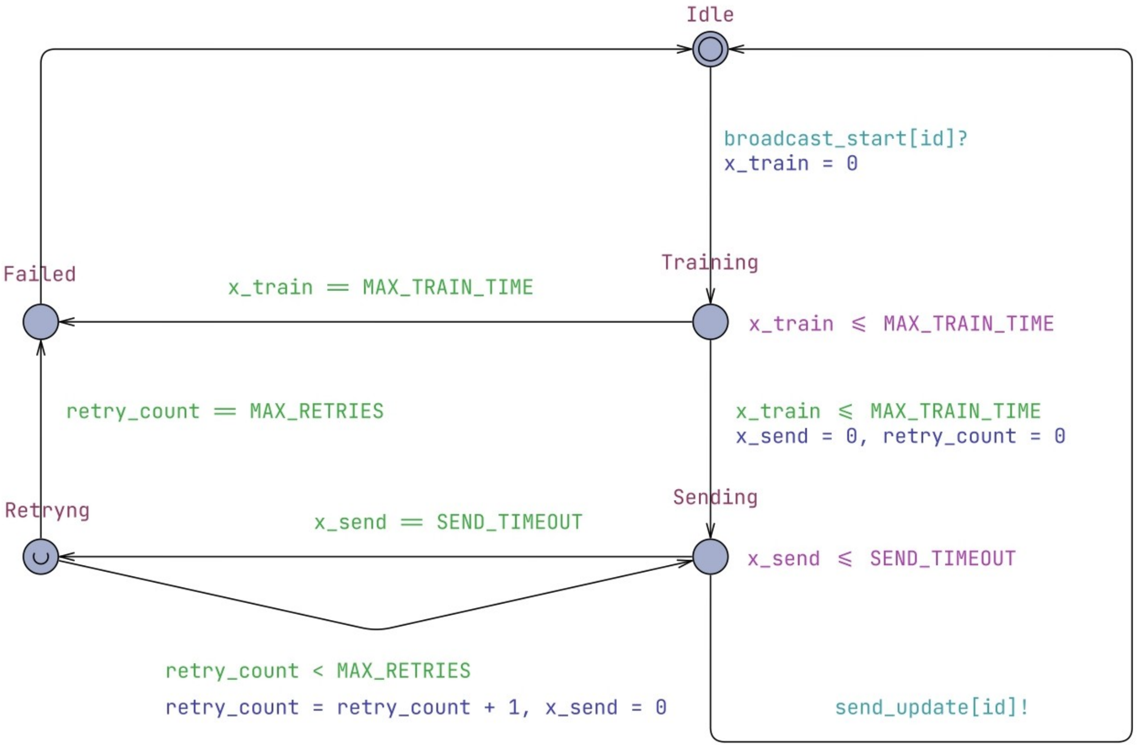


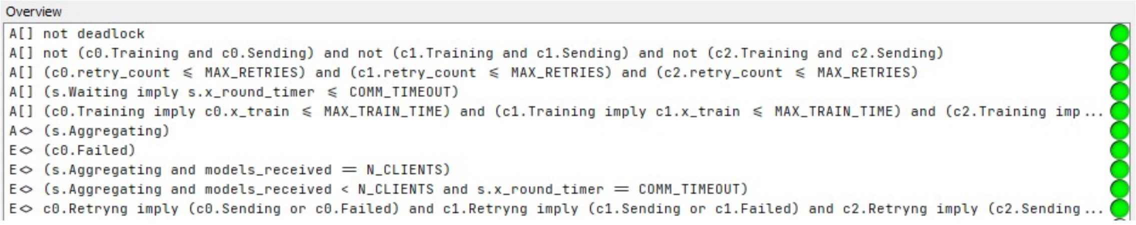
Figure 2. TS representing the Client logic. Source: Author's own production.

4. CTL Specification and Verification

To verify the correctness of the TS, properties were specified using CTL [Li et al. 2014]. The queries cover safety, liveness, and reachability requirements. A total of 10 key properties were formulated and verified, as detailed in Table 2.

5. Verification Results and Interpretation

The verification was conducted using the UPPAAL verification engine (version 5.0) [Bengtsson et al. 1996], and the complete model is available on GitHub[1]. All specified properties (P1–P10) were satisfied, as shown in Figure 3.



```
A[] not deadlock
A[] not (c0.Training and c0.Sending) and not (c1.Training and c1.Sending) and not (c2.Training and c2.Sending)
A[] (c0.retry_count ⩽ MAX_RETRIES) and (c1.retry_count ⩽ MAX_RETRIES) and (c2.retry_count ⩽ MAX_RETRIES)
A[] (s.Waiting imply s.x_round_timer ⩽ COMM_TIMEOUT)
A[] (c0.Training imply c0.x_train ⩽ MAX_TRAIN_TIME) and (c1.Training imply c1.x_train ⩽ MAX_TRAIN_TIME) and (c2.Training imp...
A<> (s.Aggregating)
E<> (c0.Failed)
E<> (s.Aggregating and models_received == N_CLIENTS)
E<> (s.Aggregating and models_received < N_CLIENTS and s.x_round_timer == COMM_TIMEOUT)
E<> c0.Retryng imply (c0.Sending or c0.Failed) and c1.Retryng imply (c1.Sending or c1.Failed) and c2.Retryng imply (c2.Sending...
```

Figure 3. Verification results for the proposed TS. Source: Author's own production.

[1] https://github.com/antonio-vmoura/es_projects

Table 2. Formal Verification Properties (CTL)

ID	CTL Formula	Description
P1	`A[] not deadlock`	The system never enters a state where no transitions are possible.
P2	`A[] not (ci.Training and ci.Sending)`	Mutual exclusion of training and transmission phases.
P3	`A[] (ci.retry_count <= MAX_RETRIES)`	Protocol adherence to retry limits.
P4	`A[] (s.Waiting imply s.x_round_timer <= COMM_TIMEOUT)`	Server strictly respects the global communication timeout.
P5	`A[] (ci.Training imply ci.x_train <= MAX_TRAIN_TIME)`	Local training never exceeds the physical time bound.
P6	`A<> s.Aggregating`	The Server eventually reaches the Aggregation state on all paths.
P7	`E<> ci.Failed`	Failure states are reachable (confirms model captures error scenarios).
P8	`E<> (s.Aggregating and models_received == N_CLIENTS)`	Ideal execution (all clients succeed) is possible.
P9	`E<> (s.Aggregating and models_received < N_CLIENTS) and (s.x_round_timer == COMM_TIMEOUT`	Partial aggregation is possible.
P10	`E<> ci.Retrying --> (ci.Sending or ci.Failed)`	The retry logic eventually resolves to a send attempt or a failure.

5.1. Analysis of Safety Properties

The satisfaction of P1 (Deadlock Freedom) confirms that the integration of time-constrained invariants with non-deterministic transitions yields a robust system. Specifically, the use of the `Waiting` state invariant in the Server TS ensures that time cannot progress indefinitely without forcing a transition. This mathematically proves that the server will never indefinitely block while waiting for a client that has disconnected. P3 and P4 validate the correctness of the parameter guards. The system strictly adheres to the retry policies and timeout thresholds defined in the global declarations.

5.2. Analysis of Liveness and Robustness

P6 (Eventual Aggregation) is the most critical liveness property for the utility of the FL system. It guarantees that for every possible execution trace, whether all clients succeed, all fail, or any combination thereof, the system terminates the round and proceeds to aggregation. P9 (Robustness to Partial Failures) verifies the ”Partial Aggregation” capability. The existence of a trace where `s.Aggregating` is reached with `models_received < N_CLIENTS` confirms that the protocol effectively handles stragglers and dropouts. Crucially, this mechanism ensures that the system is fault-tolerant; network instability or computational failure in a subset of nodes does not invalidate the training progress of functioning clients, thereby maximizing the utilization of distributed clinical data.

6. Conclusions and Future Work

This research has presented a rigorous formal verification of a FL orchestration protocol designed for privacy-preserving skin lesion segmentation. By modeling the distributed actors as a Network of TS in UPPAAL [Bengtsson et al. 1996], it was formally proven that the protocol adheres to strict safety and liveness constraints. The verification results confirm that the system successfully coordinates training across distributed clients, handling non-determinism, network latency, and failures without deadlock.

However, the verification process elucidates a fundamental trade-off in FL orchestration: the balance between system reliability and data inclusivity. The strict enforcement of `COMM_TIMEOUT`, validated by Property P4, guarantees system liveness but potentially excludes valid updates from stragglers. If the timeout is too aggressive, valuable data from under-resourced institutions may be systematically excluded, introducing bias into the global skin lesion model. The formal model provides a mechanism to simulate this trade-off; by adjusting the constants in the TS, architects can determine the optimal window that maximizes inclusion while satisfying real-time constraints.

While the current analysis utilizes conventional TS to provide binary guarantees (possible vs. impossible), real-world network failures and training durations follow probabilistic distributions [de Alfaro 1998]. Consequently, future work should extend this model to Stochastic TS to allow for quantitative queries (e.g., ”What is the probability of 80% aggregation within 10 minutes?”). Beyond stochastic modeling, future research will focus on:

1. **Secure Aggregation:** Extending the model to include states for cryptographic overhead to verify that security measures do not violate timing constraints.
2. **Asynchronous Protocols:** Modeling asynchronous FL to evaluate its potential for better handling extreme heterogeneity in client computational resources compared to the synchronous approach verified here.
3. **Empirical Validation:** Validating the formal model parameters against real-world training metrics from a deployed prototype to bridge the gap between theoretical models and physical implementation.

References

Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., and Yi, W. (1996). Uppaal—a tool suite for automatic verification of real-time systems. In *Proceedings of the DIMACS/SYCON Workshop on Hybrid Systems III: Verification and Control: Verification and Control*, page 232–243, Berlin, Heidelberg. Springer-Verlag.

Celebi, M. E., Mendonca, T., and Marques, J. S. (2015). *Dermoscopy Image Analysis*. CRC Press.

Conduah, A. K., Ofoe, S., and Siaw-Marfo, D. (2025). Data privacy in healthcare: Global challenges and solutions. *Digital Health*, 11:20552076251343959.

de Alfaro, L. (1998). Stochastic transition systems. In Sangiorgi, D. and de Simone, R., editors, *CONCUR'98 Concurrency Theory*, pages 423–438, Berlin, Heidelberg. Springer Berlin Heidelberg.

Foahom Gouabou, A. C., Collenne, J., Monnier, J., Iguernaissi, R., Damoiseaux, J.-L., Moudafi, A., and Merad, D. (2022). Computer Aided Diagnosis of Melanoma Using Deep Neural Networks and Game Theory: Application on Dermoscopic Images of Skin Lesions. *International Journal of Molecular Sciences*, 23(22):13838.

for Research on Cancer IARC, I. A. (2023). Skin cancer. Accessed on: July 14, 2024.

Hesamian, M. H., Jia, W., He, X., and Kennedy, P. (2019). Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges. *Journal of Digital Imaging*, 32:582–596.

Holmes, G. A., Vassantachart, J. M., Limone, B. A., Zumwalt, M., Hirokane, J., and Jacob, S. E. (2018). Using Dermoscopy to Identify Melanoma and Improve Diagnostic Discrimination. *Fed Pract*, 35:S39–S45. Accessed on: July 14, 2024.

Jahanifar, M., Tajeddin, N. Z., Koohbanani, N. A., Gooya, A., and Rajpoot, N. (2019). Segmentation of Skin Lesions and their Attributes Using Multi-Scale Convolutional Neural Networks and Domain Specific Augmentations. *CoRR*.

Kim, M., Sohn, H., Choi, S., and Kim, S. (2023). Requirements for trustworthy artificial intelligence and its application in healthcare. *Healthcare Informatics Research*, 29(4):315–322.

Li, Y., Li, Y., and Ma, Z. (2014). Computation tree logic model checking based on possibility measures.

López-Labraca, J., González-Díaz, I., de María, F. D., and Fueyo-Casado, A. (2022). An interpretable CNN-based CAD system for skin lesion diagnosis. *Artificial Intelligence in Medicine*, 132:102370.

Mahbod, A., Schaefer, G., Ellinger, I., Ecker, R., Pitiot, A., and Wang, C. (2018). Fusing Fine-tuned Deep Features for Skin Lesion Classification. *Computerized Medical Imaging and Graphics*, 71.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2023). Communication-efficient learning of deep networks from decentralized data.

Mirikharaji, Z., Abhishek, K., Bissoto, A., Barata, C., Avila, S., Valle, E., Celebi, M. E., and Hamarneh, G. (2023). A survey on deep learning for skin lesion segmentation. *Medical Image Analysis*, 88:102863.

Sun, T., Li, D., and Wang, B. (2021). Decentralized federated averaging.

Yurdem, B., Kuzlu, M., Gullu, M. K., Catak, F. O., and Tabassum, M. (2024). Federated learning: Overview, strategies, applications, tools and future directions. *Heliyon*, 10(19):e38137.