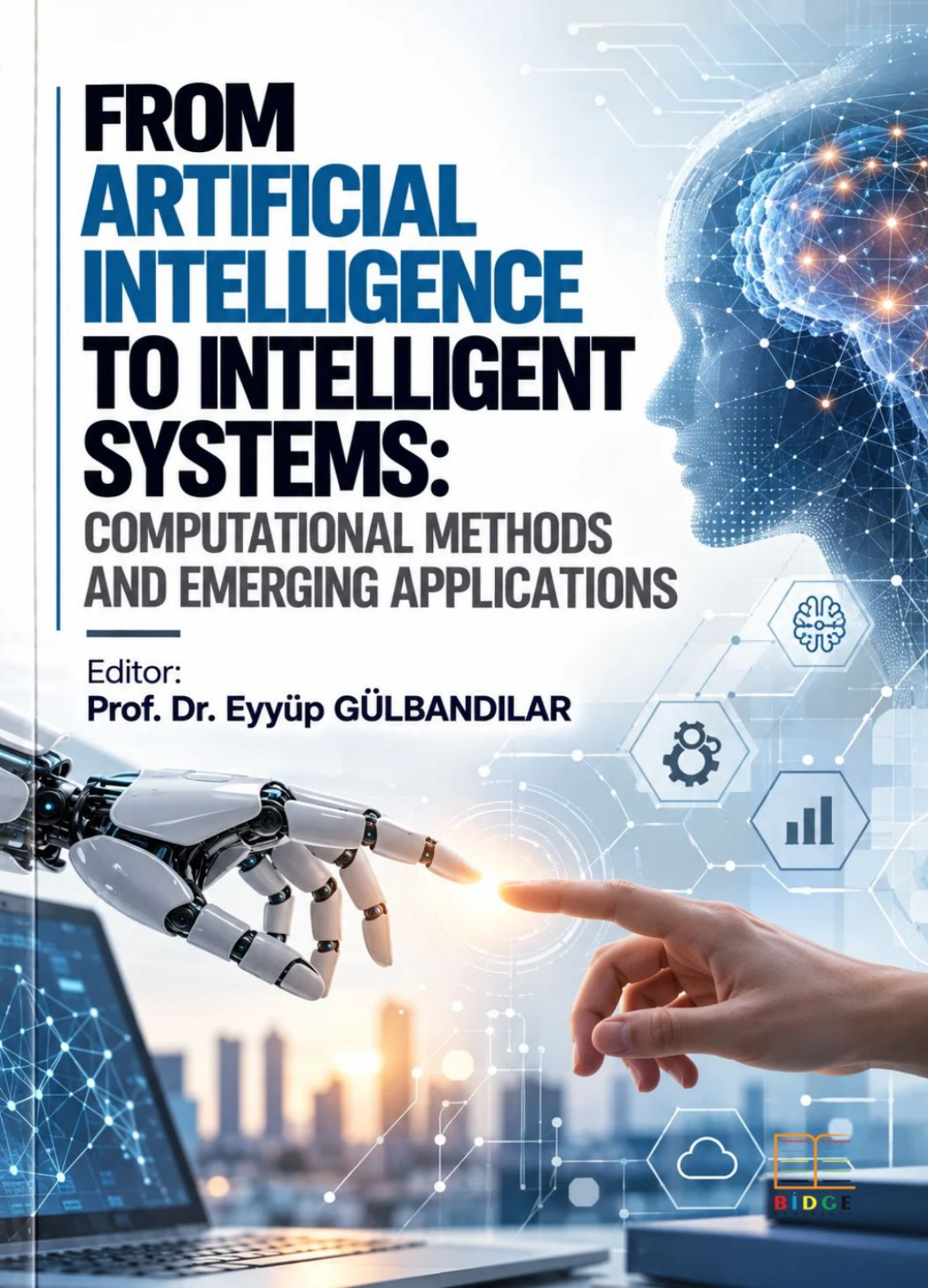
FROM ARTIFICIAL INTELLIGENCE TO INTELLIGENT SYSTEMS:

COMPUTATIONAL METHODS AND EMERGING APPLICATIONS

Editor:
Prof. Dr. Eyyüp GÜLBANDILAR



BİDGE

BİDGE Yayınları

From Artificial Intelligence To Intelligent Systems: Computational Methods and Emerging Applications

Editor: Prof Dr. Eyyüp GÜLBANDILAR

ISBN: 978-625-5057-70-9

1st Edition
Page Layout By: Gözde YÜCEL
Publication Date: 2026-10-25
BİDGE Yayınları

All rights reserved. No part of this work may be reproduced in any form or by any means, except for brief quotations for promotional purposes with proper source attribution, without the written permission of the publisher and the editor.

Certificate No: 71374
All rights reserved © BİDGE Yayınları
www.bidgeyayinlari.com.tr - bidgeyayinlari@gmail.com
Krc Bilişim Ticaret ve Organizasyon Ltd. Şti.
Güzeltepe Mahallesi Abidin Daver Sokak Sefer Apartmanı No: 7/9 Çankaya / Ankara

PREFACE

Artificial intelligence has become one of the most significant drivers of transformation in information technologies, leading to profound changes in scientific research, engineering, and many aspects of everyday life. In particular, advances in machine learning, deep learning, image processing, and data analytics have enabled the extraction of meaningful patterns from large and complex datasets, the support of prediction and decision-making processes, and the development of intelligent systems for a wide range of applications. However, the rapid evolution of artificial intelligence brings not only the need to develop more accurate models but also the challenge of ensuring that these models can be deployed reliably, efficiently, securely, and at scale.

This book has been prepared to bring together current perspectives on artificial intelligence and computational methods across diverse application domains. Although the chapters focus on different methodologies and fields of application, they share a common objective: transforming data into meaningful information, evaluating that information through analytical methods, and developing reliable and intelligent systems.

The deployment of artificial intelligence models on embedded and resource-constrained platforms has become an increasingly important area of research. In this context, the relevant chapter examines deep learning in embedded systems from the perspectives of hardware platforms, model adaptation, and performance evaluation. These aspects are particularly important for transferring artificial intelligence models from conventional computing environments to real-time systems with limited computational and energy resources.

In the field of image processing, the book addresses local binary patterns and related approaches, focusing on their theoretical foundations, variants, and recent developments. Such methods play an important role in extracting local structural characteristics from images and continue to contribute to the development of machine learning- and deep learning-based image analysis systems. In addition, the analytical foundations of regression and its application to stock market data prediction demonstrate how computational methods can be employed in financial data analysis and forecasting.

Cybersecurity constitutes another major theme of this book. As digital systems become increasingly interconnected, cyber threats are becoming more diverse, sophisticated, and difficult to detect using conventional security mechanisms alone. Consequently, artificial intelligence and computational approaches have gained increasing importance in the development of modern cybersecurity solutions. In this book, topics such as automated cyberbullying detection, graph-based cyber-attack detection, data representation, topological analysis, and computational efficiency are examined from this perspective.

Cloud security represents another contemporary area addressed in the book. Zero Trust architecture and artificial intelligence-driven identity protection constitute important components of a modern security paradigm in which users, devices, applications, and access requests are continuously evaluated rather than automatically trusted. The integration of artificial intelligence into identity verification, behavioral analysis, and threat detection processes provides opportunities for developing more dynamic, adaptive, and context-aware security mechanisms.

A central perspective throughout the book is that artificial intelligence methods should not be evaluated solely on the basis of prediction accuracy or classification performance. The practical usability of an intelligent model depends on a broader set of factors, including computational cost, data representation, model complexity, reliability, scalability, security, and the characteristics of the deployment environment. Accordingly, the chapters consider not only the theoretical and algorithmic aspects of the methods discussed but also their practical implications and performance considerations.

This book is intended for academics, researchers, graduate students, and practitioners working in computer engineering, artificial intelligence, data science, image processing, cybersecurity, and related disciplines. It is also designed for readers who wish to understand how artificial intelligence and computational methods can be adapted to different real-world problems and how seemingly distinct research areas can converge through data-driven and intelligent approaches.

The rapid pace of development in artificial intelligence means that methods considered state-of-the-art today may soon be complemented or replaced by new approaches. Therefore, this book is intended not merely to present existing techniques but also to encourage readers to formulate new research questions, explore emerging technologies, and establish connections between different disciplines.

I would like to express my sincere gratitude to all chapter authors who contributed their knowledge and expertise to this work, to the reviewers who provided valuable academic feedback, and to everyone who supported the preparation and completion of this book.

I hope that this book will contribute to the understanding and advancement of artificial intelligence and computational methods, inspire new research, and facilitate the transfer of scientific knowledge into diverse application domains.

Prof. Dr. Eyyüp GÜLBANDILAR
Editor

CONTENTS

MACHINE LEARNING IN RESOURCE CONSTRAINED EMBEDDED SYSTEMS 1
MEHMET BİLGE HAN TAŞ

EXPLAINABLE ARTIFICIAL INTELLIGENCE: INTERPRETING BLACK-BOX MODELS AND CURRENT APPROACHES 25
İSMAİL AKGÜL

IDENTIFYING SIMILARITIES BETWEEN PROPOSED AND EXISTING COURSES IN GRADUATE PROGRAMMES USING NATURAL LANGUAGE PROCESSING 50
MURAT İNCE, EBRU İNCE

DIGITAL PHENOTYPING OF MIGRAINE: PERSONALIZED MONITORING, RISK MODELING, AND DECISION SUPPORT WITH ARTIFICIAL INTELLIGENCE AND ENGINEERING APPROACHES 66
ŞULE YÜCELBAŞ, CÜNEYT YÜCELBAŞ

ZERO TRUST CLOUD SECURITY AND ARTIFICIAL INTELLIGENCE DRIVEN IDENTITY PROTECTION 86
ÖMER SEVİNÇ

CHAPTER 1

MACHINE LEARNING IN RESOURCE CONSTRAINED EMBEDDED SYSTEMS

Mehmet Bilge Han TAŞ[1]

Introduction

Advances in technology have brought more functions into individual systems, making them more complex to operate and manage. These systems are often divided into smaller units, each responsible for a specific task. Embedded systems support this structure by performing dedicated functions within a larger system. They are used to monitor and control physical processes in applications ranging from industrial machinery to portable devices. In many of these applications, sensor data are processed using rules defined in advance. Defining suitable rules becomes more difficult when operating conditions change, measurements are noisy, or several variables need to be evaluated together. Machine learning offers an alternative approach by enabling the identification of patterns within data through examples this can be applied to tasks such as fault detection, motion recognition, and voice command detection. Integrating machine learning into embedded systems

[1] Res. Asst. Dr., Erzincan Binali Yildirim University, Faculty of Engineering and Architecture, Department of Computer Engineering, Erzincan, Türkiye, ORCID: 0000-0001-6135-1849 bilgehantas@erzincan.edu.tr

requires balancing model accuracy against computational costs. In microcontroller-based devices, in particular, factors such as processing capacity, available memory, and energy consumption strictly limit the range of feasible methods. Beyond ensuring that a model fits within the available memory, it is also essential to consider the resources required for input data preparation and the storage of intermediate results generated during inference. The system must also deliver its decisions within the timeframe specified by the application. A highly accurate model is thus suitable only provided its memory usage, execution time, and energy consumption are acceptable on the target device (Lin et al., 2023).

Processing data on the device that collects it can minimize data transfers and make the system less reliant on a network link. These advantages contribute to the interest in performing machine learning on devices with limited resources. TinyML focuses on microcontroller-class hardware, with models typically trained on a more powerful computer and sent to the target device for inference. This workflow necessitates the coordinated planning of data collection, preprocessing, training, and on-device execution. Saha et al. (2022) analyze these linkages in their assessment of machine learning for microcontroller-class hardware, taking into account data, methods, and the execution environment. The available resources also determine the model family that can be employed. Kumar et al. (2017) proposed Bonsai, a tree-based technique for making predictions with minimal memory. Gupta et al. (2017) addressed resource limits with ProtoNN, which represents data with a small collection of prototypes in a low-dimensional space, reducing the storage and compute required for prediction. These papers show that embedded machine learning includes a diverse set of algorithmic families.

The choice of approach is influenced by the data format, task demands, and target hardware capabilities. Models with comparable

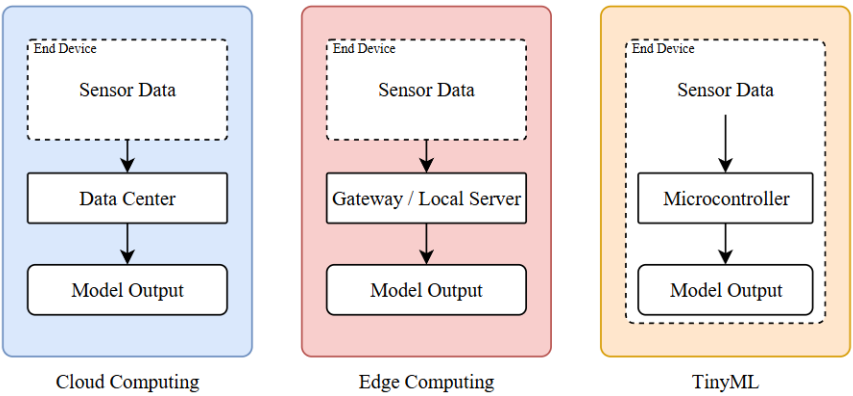
accuracy on the same job may have differing memory, latency, and energy requirements when deployed. This section investigates the architectural decisions used while running machine learning on resource-constrained embedded systems. Hardware limitations, data processing, model selection and compression, and on-device inference are all taken into account together because a decision made at one stage might have an impact on others. The review uses published studies to explain how methods can be selected for a particular application and how accuracy is balanced against resource use. Illustrative calculations of memory and energy requirements show how these choices can be assessed in practice.

The Concept and Limits of TinyML

TinyML applications need to be assessed in relation to the task and the device that will run it. An application that operates at intervals has different resource needs from one that runs continuously. Even within a processor family, devices may offer different SRAM capacities, accelerator support, and power-management options. Single limit on model size, memory, or power consumption cannot capture all these differences. The assessment must therefore consider whether the task can be carried out with the resources available on the target microcontroller. Lin et al. (2023) discuss TinyML in terms of the joint design of hardware, algorithms, and software. Edge AI covers a broader range of computing environments near the data source, including local servers, single-board computers, and industrial gateways. TinyML forms part of this area, with a particular focus on constrained hardware such as microcontrollers (Saha et al., 2022). A Linux-based single-board computer and a microcontroller should be treated as different execution environments because their memory management and software requirements differ. The RP2040, for example, is a microcontroller with two Cortex-M0+ cores, distinct from Raspberry Pi's Linux-based single-board computers (Pinheiro et al.,

2023). Figure 1 shows where computation takes place and the possible data flows between components.

Figure 1. Inference locations and data flows in cloud computing, edge computing, and TinyML deployments.



In a typical TinyML workflow, the model is trained on a computer or server, and its learned parameters are then transferred to the device (David et al., 2021). Inference uses these parameters to produce an output for a new input. Training changes the parameters using data. On-device learning is also possible, although its memory needs, update methods, and security requirements must be assessed separately. Cloud, edge, and microcontroller-based approaches can differ in how they divide the work, as well as where they run a model. A cloud service may carry out detailed image analysis while a local device checks only whether an event is present. These tasks can be combined within a single system architecture. The comparison in Table 1 should therefore be read with this division of work in mind. The extent to which connectivity requirements are reduced or data sharing is limited depends on how the application is structured.

Table 1. Qualitative comparison of cloud, edge, and TinyML deployments

Criterion	Cloud	Edge platform	TinyML
Typical execution environment	Data center	Gateway, local server, or single-board computer	Microcontroller and embedded firmware
Resource budget	Relatively ample and scalable	Varies by device	Strict SRAM, flash memory, and energy constraints
Connectivity	Data transfer is required for remote processing	Local networking and task sharing may be used	Local inference can run offline
Task scope	Large, multipurpose models	Moderate to high complexity	Narrowly scoped detection and prediction tasks
Sources of latency	Networking, queuing, and computation	Local communication and computation	Sampling, preprocessing, and inference
Data sharing	Structured around centralized processing	Data can be aggregated or summarized locally	Raw data can remain on the device

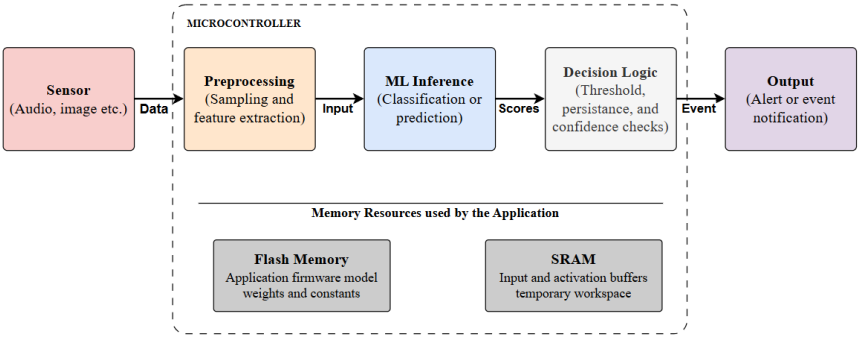
Table 1 compares cloud platforms, edge platforms, and TinyML in terms of their execution environments, resource budgets, connectivity requirements, and data-sharing characteristics. While the cloud enables the execution of large models through extensive and scalable computing resources, the transfer of data to remote servers introduces network and queuing latencies into the total processing time. In edge platforms which may operate on gateways, local servers, or single-board computers available resources vary depending on the hardware, and data can be collected and processed locally. TinyML is part of edge computing and usually targets a specific sensing or prediction task. Its implementation must fit within a microcontroller's SRAM, flash memory, and energy budget.

With a suitable application design, local inference can support offline operation and allow raw data to remain on the device (Saha et al., 2022). The combined time for sampling, preprocessing, and inference must meet the application's latency requirements. Choosing an execution environment therefore requires considering model complexity, hardware capacity, connectivity, and where the data needs to be processed.

System architecture a microcontroller

In a TinyML application, sensor acquisition, input preparation, inference, and decision logic form a single processing chain. Analog signals are read directly as they are converted into digital data. It collects measurements over a time window and converts them into appropriate input for the model. While this process can be taken as time/frequency for audio, for an image it can just be pixel numbers. The model's prediction also goes through additional checks before the app sends a notification or takes action with a physical effect, and this is very important to check. If the data that does not pass these checks is also affected by the surrounding noise, poor performance results may occur that seriously affect the success. Figure 2 shows the place of a previously trained model in the large processing chain. The diagram was prepared for this section by taking advantage of runtime and memory distinctions, as explained in the study by David et al. (2021). The arrows in the figure show the data flow and the dashed lines show the components on the microcontroller.

Figure 2. Conceptual TinyML inference pipeline and memory organization



TensorFlow Lite Micro is an early example of a system that uses a portable interpreter to run models on embedded devices with limited resources (David et al., 2021). Its scope is the set of data structures and operation kernels needed for inference, rather than the functions of a complete training library. Sensor drivers, communication code, and error handling are integrated with the rest of the firmware. Program code, model parameters, and constants are stored in non-volatile flash memory. SRAM provides space for input buffers, intermediate activations, temporary workspaces, and the software stack. Model file size alone therefore gives little indication of the SRAM needed during execution. A model with few parameters can still produce high-resolution intermediate feature maps. Peak memory use also depends on which tensors must be kept in memory at the same time as operations are executed (Lin et al., 2021).

Multiplying the number of weights by the bits used for each weight gives an initial storage estimate in bits. This calculation covers only the raw weight data. It excludes model metadata, scaling factors, alignment padding, and firmware. In the expression below, N is the number of weights and b is the bit width of each weight.

$$M_{weight} = \frac{N \times b}{8} \ [bayt] \quad (1)$$

Consider an array of 100,000 weights. Storing each weight as a 32-bit value takes 400,000 bytes, or about 390.6 KiB. Using 8-bit values reduces this to 100,000 bytes, or approximately 97.7 KiB. The fourfold reduction concerns the weight array alone. The program's total flash or SRAM use will not necessarily decrease by the same factor. Peak runtime memory must be assessed from the data and workspaces that need to be held in memory at the same time.

$$M_{peak} = max_t \left[M_{live}(t) + M_{temp}(t) + M_{stack}(t) + M_{input}(t) + M_{system}(t)\right] \quad (2)$$

The expression accounts for live intermediate tensors, temporary workspaces, the software stack, input and sensor buffers, and other system components at the same point during execution. Each memory allocation must belong to a single category so that the same storage is not counted twice. The peak value represents the maximum total usage at any given moment, rather than the sum of the individual peak values of each component. Simply aggregating the memory requirements of all layers can lead to an unnecessarily high overestimation, while considering only the largest layer might overlook buffers that coexist. MCUNetV2 addresses the high activation requirements of early layers by executing operations on smaller spatial patches (Lin et al., 2021).

Table 2 provides examples of four common microcontrollers. The values represent hardware specifications reported in the peer-reviewed articles cited in Table 2. Flash and PSRAM configurations, which are specific to the development board, are listed separately from the chip's internal SRAM capacity. Clock frequency alone is insufficient to establish a performance ranking factor such as the instruction set, memory access, processing cores, compiler options, and the numerical values employed also influence the outcome. Two cores do not automatically make a single inference run twice as fast. The benefit depends on how the software distributes the workload across the cores (Garofalo et al., 2020).

Table 2. Microcontroller specifications reported in peer-reviewed articles

Microcontroller	Core and maximum clock frequency	Internal SRAM	Flash and memory notes
nRF52840	Arm Cortex-M4F, 64 MHz	256 kB	1 MB of internal flash. Usable model memory also depends on the firmware (Verbiest et al., 2023).
RP2040	Dual-core Arm Cortex-M0+, 133 MHz	264 kB	External QSPI flash is used for program storage (Pinheiro et al., 2023).
ESP32-S3	Dual-core Xtensa LX7, 240 MHz	512 kB	The studied DevKitC-1 configuration has 8 MB of flash and 8 MB of external PSRAM (Fanariotis et al., 2024)
STM32H743VI	Arm Cortex-M7, 480 MHz	1 MB (nominal total)	2 MB of internal flash. The total RAM is not the usable model workspace (Alajlan & Ibrahim, 2022).

As shown in Table 2, data was collected from various publications on the subject and includes the memory capacity terms used. Total capacities do not indicate the amount of memory used by the model. External flash memory and PSRAM must be accounted for separately from the chip's internal SRAM. Access latencies affect operations such as memory organization and where the system stores data. Driver software on the hardware and some stack operations required for communication also affect memory consumption. Considering the total capacity of any card as the memory budget of the model may cause errors when transferring the model to the device and running it. The TinyML development process should clearly define the decision made and set out any clarity and framework for implementation. For example, detecting whether there are people in a room requires much different learning and

modeling than determining the locations of all objects in an image. Specific examples are available as part of MLPerf Tiny's Visual Wake Words task. For this comparison task, it is desired to determine whether there is a human in an image (Banbury, Reddi et al., 2021).

Data gathered must depict the target sensor along with the environment and expected use scenarios. The placement of the microphone, the way a vibration sensor is fixed, and the camera's field of view affect the statistical properties of the data. Lab environment recordings might not really replicate the reverberation, temperature changes, vibration, or lighting changes present in the field environment. It would be wrong to think that having a large number of samples would take care of these conditions. The training, validation, and test sets must be representative of the new scenarios that are expected when implementing them. If from one recording overlapping windows are randomly assigned to the sets the training and test data will contain very similar examples. In the context of human activity recognition, the impact of overlapping windows on the evaluation's accuracy is mainly determined by splitting the data in either a person-dependent or person-independent manner (Dehghani et al., 2019). Select the splits based on person, session, device, or time period to be in line with the generalization claim that you are making. If you say that the model has to work on a different device, that must be verified by the test device. Testing different time segments from the same device is not enough to convince anyone.

The first step involves testing out the chosen model on a PC or computer setup to see how it operates and what it delivers. Next, one is to adapt the model's representation such that it can be handled by the device's software capabilities and the data types it supports. Quantization samples that are used for post-training quantization (PTQ) must reflect the variety and depth of input data anticipated when the application is deployed. If one chooses to go for a training method, such as quantization-aware training (QAT), where the loss

of information due to small computational precision is taken into account, the reduced number of bits that represent the model can already be trained (Jacob et al., 2018). After doing all of the above, one should load the model onto the device and validate the consistency of both device and PC outputs, assuming that the inputs are the same. Any discrepancies could be due to input order, normalization method, data type, and/or scale and zero point. Besides, just because the model has been successfully converted, one has to be sure that the firmware preprocessing is the same as that used for the model training. Finally, one should carry out a test for each and every operation phase under conditions that mimic real-world usage, for example, a sensor coming online, a buffer that becomes full, performing calculations in the model and sending the decision. Identifying potential problem places with regard to performance of the whole system as compared to the model performance at this point in the development cycle will give the developers a head start and also save time.

Quantization

Quantization is a method that reduces the numerical representation of weights and activations to a much smaller set of numbers. 8-bit integers are by far the most widely used representation when replacing floating-point numbers. A continuous or floating-point value can be approximated to, or mapped precisely through, a scaled discrete value. When performing a rounding operation, one error could occur, but a further level of information loss can also be due to clipping the values outside the calibration range. Compressing the size of a model file won't necessarily equate to faster inference or lower power consumption. When no integer kernel is available for operation, type conversion or changing execution path can be the way to achieve a smaller degree of expected savings. In terms of numerical representation, it must be the same as the target runtime's numerical type. Jacob et al., (2018).

Also, checking operator implementations can be useful for knowing whether to proceed. David et al., (2021). Prior to moving your INT8 model to the new platform, you need to do this check again. Once quantized model is created, it is important to look at the predictions again; a model can be stable with the overall accuracy, and yet lose sensitivity at rare events. Using the data collected under the intended operating conditions is an effective way of reviewing the class results and decision thresholds. If performance drops below the level required for the application, one has to choose between quantization-aware training (QAT), a different input representation or a whole different architecture. Quantization does not fix the underlying problem of having poor data.

Pruning

By means of Pruning, some weights, channels, or computation paths are removed from the model. Unstructured pruning results in removal of individual connection weights. In structured pruning a larger unit like a channel or filter is targeted for removal (He et al., 2017). By way of Deep Compression Han et al. (2016) reduced storage requirements by way of combining pruning, quantization, and encoding of trained models. A model stored smaller doesn't guarantee faster execution on the microcontroller. With sparse representations, indexing overhead is added, while execution speed depends on support for sparse computation. In knowledge distillation, a larger teacher model guides the training of a smaller student. Class labels and similarities between classes expressed in the teacher’s outputs inform the student’s learning (Gou et al., 2021). Table 3 presents costs targeted by these methods and conditions limiting their gains.

Table 3. Resource efficiency methods and limitations regarding benefits

Method	Core modification	Intended benefit	Key limitation
Quantization	Numerical bit widths are reduced	Lower memory requirements for weights and some activations	Calibration ranges, scaling, and integer operator support affect accuracy and efficiency
Structured pruning	Channels or filters are removed	Reduced dense computational workload	Discriminative information relevant to the task may be lost
Unstructured pruning	Individual weights are pruned	Compressed storage	Indexing overhead and sparse kernel costs may limit speed gains
Knowledge distillation	A student model learns from a teacher model	Maintaining predictive quality in a smaller model	Teacher errors may be transferred to the student
Architecture search	Layer and channel configurations are selected	A model that meets the resource budget	Search can be costly and requires a model of the target hardware
Memory planning	Tensor lifetimes and execution order are planned	Lower peak SRAM usage	Buffer reuse depends on tensor lifetimes. Recomputation, if used, adds work

Author synthesis based on Jacob et al. (2018), He et al. (2017), Han et al. (2016), Gou et al. (2021), Banbury, Zhou, et al. (2021), David et al. (2021), and Lin et al. (2021) in Table 3. Only the student model needs to run on the target hardware (Gou et al., 2021). The student risks inheriting the teacher's errors and limitations arising from the training data. Therefore, repeating hardware verification after the distillation process is essential.

Joint evaluation of foundational studies

Measuring TinyML progress only by accuracy gains obscures the bottlenecks researchers aim to address. Early studies focused on low-memory inference methods and processing cores. Table 4 summarizes these contributions by highlighting the specific questions they address. The studies included in the table are not presented as results re-run within a unified experimental framework.

Table 4. Bottlenecks addressed by representative TinyML studies

Author(s)	Bottleneck	Main contribution	Considerations when interpreting results
Kumar et al. (2017)	Extremely limited memory	A compact, shallow tree-based prediction model learned in a low-dimensional space	Results depend on the input representation and the prediction task
Gupta et al. (2017)	Training-example storage and prediction cost	Joint learning of a small set of prototypes and a sparse, low-dimensional projection	The method is not a universal replacement for deep networks across all tasks
Garofalo et al. (2020)	Inefficient execution of quantized neural network operators on parallel MCUs	PULP-NN: Parallel kernels for quantized neural networks on ultra-low-power RISC-V processors	Gains depend on the ISA, parallelism, memory layout, and evaluated workload
Lin et al. (2020)	Mismatch between network architecture and inference runtime	Joint design of TinyNAS and TinyEngine	Applicability depends on the resource budget and target-specific architecture–runtime co-design
Banbury, Zhou, et al. (2021)	Architecture selection under strict resource constraints	MCU-oriented differentiable neural architecture search	Operation count is a latency proxy within the evaluated hardware and runtime settings
Lin et al. (2021)	Peak activation memory in early network layers	Patch-based inference scheduling jointly optimized with network architecture	Overlapping patches can introduce additional computation
Lin et al. (2022)	Memory requirements of on-device training	Quantization-aware scaling, sparse updates, and Tiny Training Engine	The 256 KB result applies to the evaluated models, tasks, and transfer-learning setup
Jia et al. (2024)	Resource-constrained adaptation under distribution shifts	Test-time adaptation using early-exit self-ensembles and TinyTTA Engine	Adaptation overhead and predictive quality require evaluation in the intended deployment context

Bonsai and ProtoNN demonstrate that generating learned decisions on a microcontroller is not limited to deep convolutional networks. For small inputs consisting of numerical features, a tree-based or prototype-based model may suffice. This observation suggests that, rather than starting with the largest possible network and attempting to shrink it, one should also consider the simplest model family appropriate for the given task. Using a more complex model is justified when its predictive performance and the information it retains warrant the extra resources required. In some

image-processing networks, the early layers have high activation memory requirements; MCUNetV2 addresses this problem. Peak memory usage drops when processing feature maps in patches, but overlapping can still involve extra computing work. Lin et al. (2021) manages this compromise by collaboratively designing the network layout and the schedule of inference. It is possible, nevertheless, for activations to take up more memory than what is available after weight compression.

Lin et al. (2022) presented work on On-device training under 256KB memory where they have adapted few vision tasks in 256 KB of SRAM and 1 MB of flash memories. Their methods rely mainly on sparse parameter updates and training graph optimization to achieve the desired results. They report that the updated layers, data layout and training framework have contributed to performance of the model. However, since the budget for retraining arbitrarily chosen models is different from one to another, one cannot derive general resource requirements based on the published results. The study focuses on limited adaptation on a device, a narrower scope than a complete training library. Alharthi et al. (2026) highlight continuing concerns about reliability and long-term operation in industrial IoT alongside resource limits. Performance across a product's lifecycle cannot be established from laboratory results alone. Even accuracy values from the same dataset need context, since input sizes, preprocessing, model versions, and measurement procedures influence the comparison. These conditions are overlooked when hardware or tools are ranked by a single accuracy percentage.

Measuring Performance and Energy Consumption

Accuracy, latency, peak memory use, and energy consumption should be reported together when evaluating TinyML. Comparisons between systems can use the common tasks and

measurement procedures defined by MLPerf Tiny (Banbury, Reddi, et al., 2021). The results of benchmarking have to be viewed in the context of task and reference model as well as measurement scope. A benchmarking reference model is suitable for comparison. However, real-life implementation might require a different model to satisfy its requirements. The overall accuracy still serves as a reliable measure for classification tasks but is not adequate for the occurrence of unevenly distributed events. The system that constantly reports no event could still achieve a high accuracy in a mostly quiet environment. Use sensitivity, false alarm and missed event rates to evaluate performance of event detectors. In case of non-stop input, report error rate per window and false alarms per hour as well. If the experiment lasts more than a day, the definition of an event needs to be mentioned. A model processing audio stream may need one second to get the necessary input before it can make decision even though the inference takes only couple of ms. Overlapping windows provide updates more frequently but processing has to be faster which may lead to higher energy consumption. Just knowing average inference latency does not provide a sufficient argument to declare a model real-time. To know how well the firmware does under worst-case or extreme-percentile latency scenarios, one should check how a combination of inference plus other firmware tasks behaves with such latencies.

Inference energy is calculated by integrating power over the time taken to run the model. Battery life depends on the energy consumed across the whole sensing cycle, including sensor operation, preprocessing, inference, communication, sleep, and transitions between sleep and active states. Measuring energy at the MCU during inference therefore captures only part of the energy used by the complete product over a sensing cycle. For instance, Fanariotis et al. (2024) demonstrate that the inclusion of peripherals and wireless communication must be considered alongside the

measured power. In the expression in Equation 3, E_{cycle} represents the energy of a cycle with duration T, and P_{mean} denotes the average power.

$$E_{cycle} = \int_0^T P(t)dt, \quad P_{mean} = \frac{E_{cycle}}{T} \tag{3}$$

Aspects of learning and research on the device

In the classic TinyML workflow, a model is trained in an external environment and subsequently operates with fixed parameters. While this setup is simple and controllable, it may prove insufficient when a device needs to adapt to its specific context. On-device learning aims to address this requirement by utilizing new data locally. However, training unlike inference requires additional resources for intermediate results, gradients, and the parameters to be updated consequently, the memory sufficient for inference may be inadequate for training. TinyOL is one of the research efforts exploring online learning on microcontrollers (Ren et al., 2021). The key here is not to comprehensively retrain every layer of the model for each new sample, but rather to determine how a limited adaptation can fit within the available computational and memory budget. Subsequent training systems have further reduced resource requirements by selectively updating specific layers or sub-tensors. These gains need to be assessed for the task and the adaptation method used. TinyTTA is looking into whether the use of early-exit ensembles helps memory and computation to decrease during test-time adaptation if data distributions change and the devices are low on resources (Jia et al., 2024). From these results, it cannot be reliably confirmed that a device will adapt to each change it encounters during deployment. After adaptation, the task performance and the use of resources should be measured jointly. Local learning is an effective strategy to cut down the device-to-device or device-cloud data transfers. Besides using bad data, an update could mistakenly interpret short-term conditions as

permanent patterns or may even un-learn past behaviors. So, the performance before and after an update must be compared. If labeled data are small, the evidence for a successful update might be less clear than doing the update itself. A model's adaptability does not imply a performance increase. Comparison methods should be based on a common starting model, incoming data, and amount of adaptation data and number of updates to be reported. A superficial gain could be just from having more data, and such a factor should be accounted for. Energies required to update and model's downtime should be part of the measurement when memory and latency of the model or the system are measured. Even though each step of an inference is efficient, the total energy consumption will be high because of the large number of updates. So, it is important to plan the timing of updates as efficiently as designing the model's architecture. Evaluation should include the model's behavior under the new and also some old conditions. The new setting samples assess the model's capabilities in the current context. The previously encountered setting samples show if the model's learned capability remains at performance level. Any deterioration in the rest of the areas remains hidden only if testing is done solely of the new setting. This kind of evaluation sets the boundaries for the extent of the adaptation claim. Besides, using this same framework the current model version of every devices deployed and the conditions of running should also be tracked.

If there aren't any trustworthy annotated labels or feedback and the consequences of errors are very high, using on-device learning may not be the best approach. As an option to it, employing a fixed model which is validated outside the machine may help. The product requirements should include the verification of local adaptation as well as the update resource costs and the probability of errors during the update process. Four areas still require more exploration according to us. The first direction concerns the joint

design of model structures, runtimes, and new accelerators. Then, from the sensor readout to the final outcome, one must do a complete performance analysis. The third research question is how the sharing or the storing of data costs the whole chain including the changes in the devices and operating environments. With very low resource limits, it remains very challenging even today to produce updates which do not go back in time (i.e., that a user cannot revert to) and that they actually reach the devices, the last of which will be the subject of the fourth area. To this, we can add the fact that Alharthi et al. (2026) study the related problems in the field of industrial applications under restricted hardware constraints. In the situation of laboratory prototypes being converted to industrial operation for a long-term purpose, these problems become most pertinent.

Conclusion

Developing TinyML, the first step is to clarify if the work will require a learning-based model. The decision may be based upon physical constraints and the availability of a threshold or a rule from these, in which case a model may not be the way to go and instead thresholding will suffice. It will be a case study to choose between a thresholding and a learning model and to see which one leads to the least cost and error rate. Taking the case of a learning-based approach as necessary: identify how long the observation window is for the inputs; define the number of details about past events to be kept; specify an acceptable level of error. Before actually allocating the memory to the models themselves, you should take preprocessing and firmware into consideration as well. On a device intended for running the models, measure both latency and energy consumption. Although knowing the desktop execution time and the number of parameters can give an approximate idea about the model characteristics, they cannot confirm if the model will satisfy the requirements of the application. Instead, you should perform the actual performance assessment on data that represents the target

operating environment. It is also important to assess the feasibility to do updates to the model after it has been deployed to the users. Besides these, other feasible measures would be to vary the input; reduce the problem to smaller scope; or change to another type of computing platform.

TinyML enables constrained devices to perform localized sensing tasks. Local processing supports offline operation, stores raw data on the device, or restricts the transmission of information about relevant events depending on the application. In this respect, TinyML is a great opportunity. Feasibility is impacted by model size and activation memory, but also sensing and preprocessing costs, bandwidth, mistakes while making decisions and maintenance. It is not enough to keep a running model for success. Within a resource budget, the system is to give decisions using a recognizable software version and under the target operating conditions. TinyML is an engineering solution of practically limited tasks that fit within the constraints. When you find tasks where you need either a broader sensing or intensive computation, you can distribute the computations to the other computing layers. As the reviewed studies have shown, the physical aspects, hardware, as well as the actual conditions of the operation must influence the model development decisions.

REFERENCES

Alajlan, N. N., & Ibrahim, D. M. (2022). TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications. *Micromachines, 13*(6), Article 851. https://doi.org/10.3390/mi13060851

Alharthi, S., Rashid, M., & Aljabri, M. (2026). TinyML in industrial IoT: A systematic review of applications, system components, and methodologies. *Sensors, 26*(8), Article 2550. https://doi.org/10.3390/s26082550

Banbury, C., Reddi, V. J., Torelli, P., Holleman, J., Jeffries, N., Kiraly, C., Montino, P., Kanter, D., Ahmed, S., Pau, D., Thakker, U., Torrini, A., Warden, P., Cordaro, J., Di Guglielmo, G., Duarte, J., Gibellini, S., Parekh, V., Tran, H., … Xu, X. (2021). MLPerf Tiny benchmark. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, 1*. https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/da4fb5c6e93e74d3df8527599fa62642-Abstract-round1.html

Banbury, C., Zhou, C., Fedorov, I., Matas Navarro, R., Thakker, U., Gope, D., Reddi, V. J., Mattina, M., & Whatmough, P. N. (2021). MicroNets: Neural network architectures for deploying TinyML applications on commodity microcontrollers. *Proceedings of Machine Learning and Systems, 3*, 517–532. https://proceedings.mlsys.org/paper_files/paper/2021/hash/c4d41d9619462c534b7b61d1f772385e-Abstract.html

David, R., Duke, J., Jain, A., Reddi, V. J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., & Warden, P. (2021). TensorFlow Lite Micro: Embedded machine learning on TinyML systems. *Proceedings of Machine Learning and Systems, 3*, 800–811. https://proceedings.mlsys.org/paper_files/paper/2021/file/6c44dc73014d66ba49b28d483a8f8b0d-Paper.pdf

Dehghani, A., Sarbishei, O., Glatard, T., & Shihab, E. (2019). A quantitative comparison of overlapping and non-overlapping sliding windows for human activity recognition using inertial sensors. *Sensors, 19*(22), Article 5026. https://doi.org/10.3390/s19225026

Fanariotis, A., Orphanoudakis, T., & Fotopoulos, V. (2024). Reducing the power consumption of edge devices supporting ambient intelligence applications. *Information, 15*(3), Article 161. https://doi.org/10.3390/info15030161

Garofalo, A., Rusci, M., Conti, F., Rossi, D., & Benini, L. (2020). PULP-NN: Accelerating quantized neural networks on parallel ultra-low-power RISC-V processors. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 378*(2164), Article 20190155. https://doi.org/10.1098/rsta.2019.0155

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision, 129*(6), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z

Gupta, C., Suggala, A. S., Goyal, A., Simhadri, H. V., Paranjape, B., Kumar, A., Goyal, S., Udupa, R., Varma, M., & Jain, P. (2017). ProtoNN: Compressed and accurate kNN for resource-scarce devices. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 1331–1340). PMLR. https://proceedings.mlr.press/v70/gupta17a.html

Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *4th International Conference on Learning Representations (ICLR 2016)*. https://arxiv.org/abs/1510.00149

He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 1398–1406). IEEE. https://doi.org/10.1109/ICCV.2017.155

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2704–2713). IEEE. https://doi.org/10.1109/CVPR.2018.00286

Jia, H., Kwon, Y. D., Orsino, A., Dang, T., Talia, D., & Mascolo, C. (2024). TinyTTA: Efficient test-time adaptation via early-exit ensembles on edge devices. *Advances in Neural Information Processing Systems, 37*, 43274–43299. https://doi.org/10.52202/079017-1371

Kumar, A., Goyal, S., & Varma, M. (2017). Resource-efficient machine learning in 2 KB RAM for the Internet of Things. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 1935–1944). PMLR. https://proceedings.mlr.press/v70/kumar17a.html

Lin, J., Chen, W.-M., Cai, H., Gan, C., & Han, S. (2021). MCUNetV2: Memory-efficient patch-based inference for tiny deep learning. *Advances in Neural Information Processing Systems, 34*, 2346–2358. https://proceedings.neurips.cc/paper_files/paper/2021/hash/1371bccec2447b5aa6d96d2a540fb401-Abstract.html

Lin, J., Chen, W.-M., Lin, Y., Cohn, J., Gan, C., & Han, S. (2020). MCUNet: Tiny deep learning on IoT devices. *Advances in Neural Information Processing Systems, 33*, 11711–11722. https://proceedings.neurips.cc/paper/2020/hash/86c51678350f656dcc7f490a43946ee5-Abstract.html

Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., Gan, C., & Han, S. (2022). On-device training under 256KB memory. *Advances in Neural Information Processing Systems, 35*, 22941–22954. https://doi.org/10.52202/068431-1667

Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., & Han, S. (2023). Tiny machine learning: Progress and futures [Feature]. *IEEE Circuits and Systems Magazine, 23*(3), 8–34. https://doi.org/10.1109/MCAS.2023.3302182

Pinheiro, I., Aguiar, A., Figueiredo, A., Pinho, T., Valente, A., & Santos, F. (2023). Nano aerial vehicles for tree pollination. *Applied Sciences, 13*(7), Article 4265. https://doi.org/10.3390/app13074265

Ren, H., Anicic, D., & Runkler, T. A. (2021). TinyOL: TinyML with online-learning on microcontrollers. In *2021 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). IEEE. https://doi.org/10.1109/IJCNN52387.2021.9533927

Saha, S. S., Sandha, S. S., & Srivastava, M. (2022). Machine learning for microcontroller-class hardware: A review. *IEEE Sensors Journal, 22*(22), 21362–21390. https://doi.org/10.1109/JSEN.2022.3210773

Verbiest, J.R., Bonnechère, B., Saeys, W., Van de Walle, P., Truijen, S., & Meyns, P. (2023). Gait stride length estimation using embedded machine learning. *Sensors, 23*(16), Article 7166. https://doi.org/10.3390/s23167166

CHAPTER 2

EXPLAINABLE ARTIFICIAL INTELLIGENCE: INTERPRETING BLACK-BOX MODELS AND CURRENT APPROACHES

İsmail AKGÜL[1]

1. Introduction

Artificial intelligence (AI) systems can learn patterns from large amounts of data and perform classification, prediction, and decision-support tasks. Advances in machine learning (ML) and deep learning (DL), in particular, have created significant opportunities for solving complex problems. Nevertheless, the decision-making processes of some models with high predictive performance cannot be easily traced or understood by humans. Large numbers of parameters, nonlinear relationships, and interactions among features can make it difficult to explain why a particular prediction was produced (Arrieta et al., 2020). This issue is commonly referred to in the literature as the black-box problem (Guidotti et al., 2018). Although a model's inputs and outputs may be observable, the inability to understand directly the computational processes that lead to its outputs can make the model difficult to

[1] Assoc. Prof. Dr., Erzincan Binali Yıldırım University, Faculty of Engineering and Architecture, Department of Computer Engineering, Orcid: 0000-0003-2689-8675

evaluate. For example, when a credit assessment system rejects an application, simply communicating the decision is not sufficient to establish which features influenced it. Similarly, when a clinical decision-support system indicates a high risk of disease, questions arise about how the prediction should be interpreted in a clinical context.

Explainable artificial intelligence (XAI) encompasses the methods and approaches developed to address these needs. Such methods can serve different purposes, ranging from examining the features associated with an individual prediction to evaluating a model's overall behavior. Explainability is therefore a broad field that cannot be reduced to a single algorithm or a uniform form of explanation (Ding et al., 2022; Schwalbe & Finzel, 2024; Fridkin & Bendersky, 2026). However, a model's ability to generate an explanation does not necessarily mean that it is accurate or trustworthy. Incorrect predictions can also be presented in an explainable form. Moreover, an explanation that appears convincing to users may not accurately reflect the decision patterns actually used by the model. The sensitivity tests developed by Adebayo et al. (2018) to evaluate explanation methods are important because they demonstrate that judging explanations solely by how plausible they appear is insufficient.

This chapter examines the fundamental concepts and principal methods of XAI, compares the differences between these methods, and evaluates their contributions across application domains. Its central approach is to treat explainability not merely as the production of a technical output, but as part of a broader process of understanding model behavior, identifying limitations, and evaluating results in the appropriate context.

2. Fundamental Concepts of Explainable Artificial Intelligence (XAI)

2.1. Explainability and Interpretability

Although the terms explainability and interpretability are frequently used together in the literature, they can emphasize different aspects of understanding a model. Interpretability refers to the extent to which a model's structure or decisions can be understood directly by humans. Explainability, by contrast, focuses on providing rationales and representations that help users understand a model's predictions or behavior. Nevertheless, the literature does not establish a universally accepted, precise distinction between these concepts (Lipton, 2018; Arrieta et al., 2020). Linear models and small decision trees can serve as directly interpretable models under appropriate conditions. Their structures can be examined to understand the effects of individual variables or the rules governing decisions. By contrast, it is more difficult to trace the internal computations that produce predictions in large ensemble models and complex neural networks. Post-hoc explanation methods can be used to investigate the behavior of these models.

An important distinction here is between explaining a complex model after it has been developed and designing an interpretable model from the outset. Post-hoc methods attempt to represent an original model's individual predictions or overall behavior in an understandable form. However, the explanatory model and the model being explained are not the same. Therefore, the extent to which an explanation represents the original model's behavior must be evaluated separately (Ribeiro et al., 2016; Rudin, 2019; Di Marino et al., 2025). Directly interpretable models should also be considered in high-stakes applications. Model complexity does not necessarily lead to better results in every situation. Likewise, choosing an interpretable model does not automatically

guarantee more accurate decisions. Model selection should take predictive performance, explainability, the costs of errors, and the intended application context into account.

2.2. Local and Global Explanations

Explanations can be classified as local or global according to their scope. Local explanations focus on understanding the rationale behind a prediction for a specific instance. For example, they can be used to investigate why a particular credit application was rejected or why an image was assigned to a certain class. LIME and instance-level SHAP explanations can be used for this purpose (Ribeiro et al., 2016; Lundberg & Lee, 2017; Lin et al., 2026). Global explanations, on the other hand, aim to examine a model's overall behavior, the importance of variables across a dataset, or patterns that emerge across different inputs. Feature importance analyses and certain feature relationship plots can be used in such evaluations (Friedman, 2001; Molnar, 2022).

These two approaches address different questions. A feature that is generally important does not necessarily have the same effect on every individual prediction. Similarly, an explanation generated for a single instance may not represent the model's behavior across the entire dataset. Accordingly, using local and global explanations together may be appropriate when a comprehensive assessment of a model is required.

2.3. Model-Specific and Model-Agnostic Methods

Explanation methods can also be classified according to whether they require access to a model's internal structure. Model-agnostic methods generally generate explanations using the model's inputs and outputs. This makes it possible to apply the same approach to different types of models. LIME is one of the principal methods in this category (Ribeiro et al., 2016). Model-specific methods, by contrast, rely on the internal structure, parameters, or

gradients of particular model families. For example, Gradient-weighted Class Activation Mapping (Grad-CAM) was developed to visualize image regions associated with a particular class prediction in convolutional neural networks (CNNs) (Selvaraju et al., 2017; Fayyaz et al., 2025).

Although this classification can help guide method selection, it does not establish a ranking of explanation quality. A model-agnostic method is not necessarily more reliable in every situation, just as a method that uses a model's internal structure does not automatically produce more accurate explanations. The suitability of a method should be considered in relation to the problem being explained and the criteria used to evaluate the explanation.

2.4. Evaluating Explanations

The success of an explanation depends on more than its ability to produce an understandable graph or textual description. An explanation should accurately represent model behavior, avoid unnecessary changes under similar conditions, and provide useful information to its intended users. Various technical metrics and user-centered evaluation approaches are used to assess explanation quality (Carvalho et al., 2019; Naveed et al., 2024; Fridkin & Bendersky, 2026). The principal criteria include the following:

- **Fidelity:** The extent to which an explanation accurately represents the behavior of the model being explained.
- **Stability:** Whether explanations change unnecessarily for similar inputs.
- **Understandability:** The extent to which the intended user can comprehend the explanation.
- **Actionability:** The extent to which the explanation helps identify an appropriate next step.

- **Scope:** The instances or conditions to which the explanation applies.

These criteria are not independent of one another. For example, an easy-to-understand explanation may be misleading if it does not accurately represent model behavior. Conversely, an explanation that is technically faithful may be too complex for users to understand. The technical properties of an explanation and the benefits it provides to users must therefore be evaluated together.

3. Principal Methods Used in XAI

Explanation methods differ in the types of information they provide and the questions they are designed to answer. Some assign feature contributions to predictions, whereas others construct a simple model around a particular prediction. Counterfactual methods investigate which changes may be required to obtain alternative outcomes, while gradient-based methods examine the relationship between model outputs and inputs. Method selection should therefore be guided by the specific explanatory need (Ding et al., 2022; Schwalbe & Finzel, 2024).

3.1. Shapley Additive Explanations (SHAP)

SHAP is a family of methods developed to explain the contributions of individual features to ML predictions. Lundberg and Lee (2017) introduced this approach as a unified framework that brings different additive feature attribution methods together. Its theoretical foundation lies in Shapley values from cooperative game theory. The central idea is to assess a feature's contribution by considering its marginal contributions when added to different subsets of features.

The additive structure of SHAP can generally be expressed as shown in Equation 1:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i \, z_i' \qquad (1)$$

Here, g denotes the explanation model, ϕ_0 represents the reference prediction value, ϕ_i denotes the contribution attributed to the corresponding feature, and z_i' represents the feature in the explanation space. M is the number of features considered in the explanation. This formulation represents a prediction in terms of a reference value and the contributions attributed to individual features (Lundberg & Lee, 2017).

One advantage of SHAP is that it can be used not only to examine individual predictions but also to analyze multiple predictions collectively. For example, in a risk prediction model, it is possible to investigate which variables increase or decrease the score assigned to a particular application. When explanations for multiple applications are considered together, additional insights into the model's overall behavior can be obtained.

Nevertheless, SHAP values must be interpreted carefully. Results may depend on how features are handled and which reference data are used. For correlated variables, the assumptions adopted can also influence how contributions are distributed among features. Furthermore, a feature's substantial contribution to a prediction does not prove that the feature causes the outcome in the real world. SHAP is therefore a useful tool for examining predictive behavior, but it is not a substitute for causal inference (Molnar, 2022; Salih et al., 2025).

3.2. Local Interpretable Model-agnostic Explanations (LIME)

LIME aims to explain a specific prediction made by a complex model by constructing a simpler model around the instance of interest. Ribeiro et al. (2016) proposed generating new samples in the neighborhood of the instance to be explained, evaluating these

samples using the original model, and learning an interpretable model that approximates its behavior in the local region.

The basic approach underlying LIME can be expressed as shown in Equation 2:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}}[L(f, g, \pi_x) + \Omega(g)] \quad (2)$$

Here, f denotes the original model being explained, g represents the explanation model, G is the set of candidate explanation models, L is a loss function measuring local disagreement between the two models, π_x represents the weights assigned to the neighborhood of the instance being explained, and $\Omega(g)$ represents the complexity of the explanation model. The objective is to obtain a model that adequately represents the original model's behavior around the instance of interest while remaining interpretable.

A major advantage of LIME is that it can be applied to different classifiers without being restricted to a particular model family. In text classification, local explanations can identify words that support or weaken a prediction. In image classification, explanations can be generated using specific image regions.

However, LIME's results may vary depending on the sampling strategy, how the local neighborhood is defined, and the complexity of the explanation model. Moreover, the fact that a local explanation model performs well around a particular instance does not mean that it accurately represents the original model's behavior throughout the entire input space. LIME explanations should therefore be evaluated for their stability under different settings and their agreement with the original model (Ribeiro et al., 2016; Carvalho et al., 2019; Meng et al., 2024).

3.3. Feature Importance Analysis

Feature importance analysis encompasses methods used to investigate which variables play a greater role in model predictions. Such analyses can help identify the inputs on which a model relies, detect potential data problems, and investigate model behavior (Carvalho et al., 2019). One widely used approach is permutation-based feature importance, which examines how much model performance changes after the values of a variable are shuffled. If performance declines after shuffling, this may indicate that the model relies on the corresponding variable when making predictions (Molnar, 2022).

However, importance scores do not directly indicate the causal effects of variables in the real world. Correlated features may contain similar information within a model, making their importance scores more difficult to interpret. Furthermore, the fact that a variable is important to a model does not establish that decisions based on that variable are fair or accurate (Ebrahimi et al., 2025). Rather than using feature importance analysis as a standalone decision criterion, it is therefore more appropriate to treat it as a starting point for understanding overall model behavior. In applications involving consequential decisions, such analyses should be supplemented with instance-level investigations and independent performance tests.

3.4. Counterfactual Explanations

Counterfactual explanations aim to identify the changes to input features that might be required to produce an outcome different from the current prediction. While feature attribution methods address the question, “Which features are associated with this prediction?”, counterfactual methods focus on the question, “Which changes could be associated with a different outcome?” (Wachter et al., 2018; Guidotti, 2024).

A basic counterfactual explanation problem can be formulated as shown in Equation 3:

$$x' = \underset{z}{\operatorname{argmin}}\ d(x,z) \quad \text{subject to} \quad f(z) = y_{\text{target}} \tag{3}$$

Here, x represents the current instance, z denotes an alternative instance, $d(x,z)$ measures the distance between the two instances, f is the prediction model, and y_{target} is the desired outcome. The objective is to find an alternative instance that produces the target output while remaining as close as possible to the original instance.

Counterfactual explanations can be particularly useful when users need actionable information. However, a change that appears mathematically feasible may not be practical in real life. Explanations that recommend changing immutable features, such as age or events that have already occurred, are also unsuitable in practical terms. Changeability, realism, and relevance to the user should therefore be considered when evaluating counterfactual explanations (Guidotti, 2024).

It is also important not to confuse counterfactual explanations with causal relationships in the real world. The fact that a model produces a different output for a different input does not, by itself, prove that making the same change in real life would produce the same outcome. These explanations should therefore primarily be regarded as tools for examining model behavior.

3.5. Gradient-Based and Visual Explanation Methods

Some explanation methods used with DL models exploit the sensitivity of model outputs to changes in their inputs. Gradient-based methods examine the relationship between changes in the output and components of the input to identify the features to which the model is sensitive (Wang et al., 2024). Integrated Gradients evaluates changes in a model's output along a path between an input

and a reference input by using gradients. The method provides an axiomatic approach to addressing situations in which ordinary gradients may be insufficient (Sundararajan et al., 2017). Gradient-weighted Class Activation Mapping (Grad-CAM), in turn, aims to display image regions associated with a particular class prediction in CNNs as a heatmap. This method makes it possible to visually examine which regions of an image may contribute to a classification result (Selvaraju et al., 2017).

These visualizations can make it easier to investigate the image regions to which a model is sensitive. However, highlighting a particular region in a heatmap does not prove that the model understands that region in the same way humans do or that its decision necessarily depends on that region. Whether the explanation reflects the model's learned behavior must be tested separately (Adebayo et al., 2018). The reliability of gradient-based and visual explanation methods should therefore be assessed not merely by inspecting the visualizations they produce, but by systematically testing whether the explanations are consistent with the behavior learned by the model.

3.6. Attention Mechanisms and Relationship to Explanation

Attention mechanisms, particularly in natural language processing and transformer architectures, involve computations that indicate how a model distributes weights across different input components. These weights can be used to examine which components receive greater weight during processing. However, interpreting attention weights directly as explanations of model decisions remains controversial (Jain & Wallace, 2019; Lyu et al., 2024).

Jain and Wallace (2019) presented evidence showing that attention weights cannot always be regarded as reliable explanations for model predictions. A high attention weight assigned to a word or

text segment does not, by itself, establish its influence on the final prediction.

Attention weights should therefore be treated as indicators that can help investigate a model's internal computations. Their validity as explanations must be independently established for the particular model and task.

3.7. Rule-Based and Example-Based Explanations

Rule-based explanations aim to describe the conditions under which a decision is made in an understandable form. Example-based methods, on the other hand, help investigate model decisions by relating a prediction to similar instances. These approaches can be useful, particularly when explanations need to be understood by non-technical users (Guidotti et al., 2018; Hong & Park, 2025).

The Anchors approach developed by Ribeiro et al. (2018) focuses on identifying sets of high-precision conditions under which a particular prediction remains valid. However, a rule that achieves high precision for certain instances is not necessarily valid across a broader set of examples. Similarly, two instances that a model considers mathematically similar may not be meaningfully similar from a human perspective.

Accordingly, when using rule-based and example-based explanations, it is important to consider the conditions under which an explanation remains valid, its scope, and the suitability of the similarity measure being used.

4. Comparison of Explanation Methods

Because explanation methods address different questions, it would be inappropriate to claim that a single method is the best choice for every application. SHAP and LIME help examine feature contributions, whereas counterfactual explanations focus on the changes required to obtain alternative outcomes. Methods such as

Integrated Gradients and Grad-CAM can be used to investigate the relationship between inputs and the outputs of DL models. The main characteristics of these explanation methods are summarized comparatively in Table 1 (Ribeiro et al., 2016; Lundberg & Lee, 2017; Selvaraju et al., 2017; Sundararajan et al., 2017; Adebayo et al., 2018; Guidotti et al., 2018; Ribeiro et al., 2018; Jain & Wallace, 2019; Molnar, 2022; Guidotti, 2024).

Table 1 Comparison of the main characteristics of explanation methods

Method	Main Purpose	Advantage	Limitation
SHAP	Attribute feature contributions to predictions	Can be used for individual and aggregate analyses	Depends on assumptions and computational cost
LIME	Explain a specific prediction using a local model	Applicable to different model types	Results may vary with sampling and parameter settings
Feature importance analysis	Examine variables considered important by the model	Provides insights into overall behavior	Does not establish causal effects
Counterfactual explanations	Identify changes associated with an alternative outcome	Can provide actionable information	Suggested changes may be unrealistic or impractical
Integrated Gradients	Examine input contributions using gradients	Can be used with deep neural networks	May be sensitive to the reference input
Grad-CAM	Visualize regions associated with an image prediction	Facilitates visual inspection	Does not, by itself, establish the reason for a decision
Attention weights	Examine the distribution of attention	Provides an indication of internal computations	Not always a reliable explanation
Rule-based and example-based methods	Explain decisions through conditions or examples	Can be understandable to humans	May involve limitations in scope and similarity

In practice, method selection should begin by identifying the question that needs to be explained. Feature importance analysis may be appropriate when the objective is to investigate which variables the model generally relies on. SHAP or LIME may be considered when the rationale behind a specific prediction is of interest. Counterfactual explanations may be more suitable when the aim is to inform users about how an alternative outcome might be achieved.

It is also possible to combine multiple methods. For example, general feature importance can be examined first, followed by SHAP explanations for selected instances. Nevertheless, different methods should not be expected to produce identical results, as they rely on different assumptions and explanatory objectives. Differences between explanations can therefore serve as an indication that the model and the methods themselves warrant further investigation.

5. Application Domains and Current Challenges

5.1. Healthcare and Clinical Decision Support

AI can be used in healthcare for tasks such as medical image analysis, disease risk prediction, and clinical decision support. In these applications, it is important not only for a model to produce accurate predictions but also for users to understand how those predictions should be assessed in a clinical context. Explanation methods can help healthcare professionals examine model outputs and identify potential errors (Arrieta et al., 2020; Saw et al., 2025).

For example, the image regions to which an image classification model is sensitive can be examined using methods such as Grad-CAM. However, the fact that a visualization highlights a clinically meaningful region does not, by itself, prove that the model identified the disease for the correct reasons. The model may also rely on image characteristics that are not directly related to the disease. Explanations should therefore not replace clinical validation or independent performance evaluations (Selvaraju et al., 2017; Adebayo et al., 2018).

The intended audience is also important in this domain. The technical details required by a researcher may differ from the information a clinician needs when making a decision. Explanations intended for patients may require a different language and level of detail. The target user and decision-making context should therefore be taken into account when designing explanations.

5.2. Finance and Credit Assessment Systems

In the financial sector, ML can be used for tasks such as credit risk prediction, fraud detection, and risk management. In these applications, explanations can support the examination of features associated with a particular prediction and the investigation of potential model errors. For example, SHAP or LIME can be used to examine the contributions of specific variables to a credit application assessment (Ribeiro et al., 2016; Lundberg & Lee, 2017).

However, the importance of a variable in a prediction does not establish that a decision is fair or legally compliant. If the training data reflect historical inequalities, explanation methods may help investigate these patterns, but they do not eliminate them automatically. Fairness assessments must also consider model performance across different groups, data quality, and compliance with the rules governing the decision-making process.

Counterfactual explanations can also be used to provide information to users in financial applications. For example, they can indicate the conditions under which a model might produce a different decision. Nevertheless, the suggested changes must be realistic and feasible. A mathematical scenario that changes a model's output may not translate into a recommendation that a user can implement in real life (Wachter et al., 2018; Guidotti, 2024).

5.3. Cybersecurity and Anomaly Detection

In cybersecurity, ML is used for tasks such as network traffic classification, malware detection, and the identification of unusual behavior (Mohale & Obagbuwa, 2025). Explanation methods can help investigate why an event was classified as suspicious or which features increased its risk score (Guidotti et al., 2018; Calzarossa et al., 2025). Security analysts can use these explanations to compare model outputs with independent indicators and investigate potential false positives.

However, explainability has a particular dimension in cybersecurity: detailed information about a model's decision logic can support defensive operations, but it can also help malicious actors understand the system's limitations. It is therefore important to consider who receives an explanation and the level of detail shared. Explainability should form part of security assessment rather than replace robustness testing, error analysis, or operational monitoring.

5.4. Public Services and High-Stakes Decisions

Automated decision systems used in public services, education, employment, and social assistance can affect individuals' rights and opportunities. Understanding the reasons behind decisions, investigating errors, and auditing decision-making processes are therefore important. In high-stakes applications, implementing a system that generates post-hoc explanations may not be sufficient; the use of directly interpretable models should also be considered whenever feasible (Rudin, 2019).

The European Union's Artificial Intelligence Act establishes a regulatory framework that includes risk-based obligations and transparency-related requirements for certain AI systems. However, the regulation should not be interpreted as imposing the same unrestricted explanation requirements on every AI system. The applicable requirements depend on the system's classification, intended purpose, and relevant provisions (European Parliament & Council of the European Union, 2024).

The Artificial Intelligence Risk Management Framework published by the US National Institute of Standards and Technology (NIST) also links explainability and interpretability to the management of risks associated with trustworthy AI. This approach supports the view that explainability should not be reduced to a single technical method but should instead be incorporated into the

design, development, evaluation, and use of AI systems (NIST, 2023).

5.5. Explainability, Trust, and User Evaluation

One of the major challenges in XAI is the assumption that providing more explanations will always increase trust. In practice, however, the goal should not simply be to increase trust but to ensure that it is appropriately calibrated. Excessive trust in an unreliable model, as well as the unjustified rejection of a reliable one, can lead to undesirable outcomes (Miller, 2019; NIST, 2023).

The benefits explanations provide to users should not be measured solely by whether users like them or find them easy to understand. It is also necessary to examine whether explanations improve decision quality, help users understand model limitations, and prevent misplaced confidence. The literature on XAI evaluation emphasizes that user-centered evaluation constitutes a distinct dimension that complements technical criteria (Miller, 2019; Naveed et al., 2024).

Accordingly, two fundamental questions should be considered together when evaluating explanations: Does the explanation accurately reflect the model's behavior? Can the intended user understand and use the explanation appropriately? Answering only one of these questions is insufficient to assess the overall quality of an explanation.

5.6. Current Challenges and Research Directions

One of the major challenges in XAI is establishing context-appropriate criteria for evaluating the quality of different explanation methods. Because these methods have different objectives, comparing all of them using a single metric is difficult. For example, evaluating a feature attribution method and a method that identifies the changes required to obtain alternative outcomes using the same

criterion may overlook their fundamental differences (Ding et al., 2022; Schwalbe & Finzel, 2024).

Another challenge concerns the stability and reliability of explanations. Producing substantially different explanations for similar inputs can make it difficult for users to understand model behavior. Nevertheless, consistency alone is not sufficient; an explanation must also be sensitive to the model's actual behavior. Explanation methods should therefore be tested both for stability and for their ability to represent model behavior accurately (Adebayo et al., 2018; Carvalho et al., 2019; Calzarossa et al., 2025; Fridkin & Bendersky, 2026).

The relationship between explainability and causality also requires careful consideration. Feature contributions, gradients, and changes in model predictions can provide information about how a model behaves, but they do not independently establish real-world cause-and-effect relationships. Explaining model behavior must therefore be distinguished from identifying causal mechanisms in the real world (Wachter et al., 2018; Baron, 2023).

The growing adoption of generative AI and large language models (LLMs) has introduced new questions for explainability research. The complex internal structures of these systems can make it difficult to fully explain the processes through which a particular response is generated. Although attention weights and feature attribution methods can be useful for specific investigations, they should not be assumed to provide a complete explanation of the model's internal workings or the correctness of its responses (Jain & Wallace, 2019; Fridkin & Bendersky, 2026).

Finally, explainability should not be addressed only at the final stage of model development. Different explanatory needs may arise during data preparation, model development, testing, deployment, and post-deployment monitoring. The selection of

explanation methods should therefore be planned alongside model documentation, risk management, and user evaluation (NIST, 2023).

6. Conclusion and Future Research

XAI is an important research field that aims to make the predictions of complex ML models more understandable to humans. Feature attribution methods such as SHAP and LIME help investigate individual predictions, while feature importance analyses provide insights into overall model behavior. Counterfactual explanations identify changes that may lead to alternative outcomes, whereas gradient-based and visual methods can be used to investigate the relationship between inputs and DL model outputs.

Each of these methods has its own assumptions, conditions of use, and limitations. An explanation method should therefore not be selected solely because it is widely used or produces visually understandable outputs. The question to be explained must first be identified, the suitability of the method must be tested, and the results must be evaluated in the context in which they will be used.

An understandable explanation does not establish that a model is accurate or fair. Explanation methods can account for incorrect decisions and may, in some cases, represent model behavior only approximately. The technical reliability of explanations and the benefits they provide to users should therefore be evaluated separately.

Future research will continue to focus on evaluating explanation methods more consistently, developing forms of explanation that meet user needs, and clarifying the boundaries between causal explanations and explanations of model behavior. In addition, explainability should be considered alongside fairness, robustness, accountability, and risk management.

In conclusion, XAI provides tools that contribute to understanding black-box models, but it is not sufficient on its own to ensure trustworthy AI. An effective approach requires selecting an appropriate explanation method, testing the resulting explanations, and evaluating the findings in the relevant decision-making context. Particularly in high-stakes applications, explainability should be treated as part of a broader assessment and governance process that extends throughout the entire model lifecycle.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems* (Vol. 31, pp. 9505–9515).

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion, 58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Baron, S. (2023). Explainable AI and causal understanding: Counterfactual approaches considered. *Minds and Machines, 33*, 347–377. https://doi.org/10.1007/s11023-023-09637-x

Calzarossa, M. C., Giudici, P., & Zieni, R. (2025). An assessment framework for explainable AI with applications to cybersecurity. *Artificial Intelligence Review, 58*, Article 150. https://doi.org/10.1007/s10462-025-11141-w

Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics, 8*(8), Article 832. https://doi.org/10.3390/electronics8080832

Di Marino, A., Bevilacqua, V., Ciaramella, A., De Falco, I., & Sannino, G. (2025). Ante-hoc methods for interpretable deep models: A survey. *ACM Computing Surveys, 57*(10), Article 262, 1–36. https://doi.org/10.1145/3728637

Ding, W., Abdel-Basset, M., Hawash, H., & Ali, A. M. (2022). Explainability of artificial intelligence methods, applications and challenges: A comprehensive survey. *Information Sciences, 615*, 238–292. https://doi.org/10.1016/j.ins.2022.10.013

Ebrahimi, S., Abdelhalim, E., Hassanein, K., & Head, M. (2025). Reducing the incidence of biased algorithmic decisions through feature importance transparency: An empirical study.

European Journal of Information Systems, 34(4), 636–664. https://doi.org/10.1080/0960085X.2024.2395531

European Parliament & Council of the European Union. (2024). *Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*. Official Journal of the European Union.

Fayyaz, A. M., Abdulkadir, S. J., Talpur, N., Al-Selwi, S. M., Hassan, S. U., & Sumiea, E. H. (2025). Grad-CAM (Gradient-weighted Class Activation Mapping): A systematic literature review. *Computers in Biology and Medicine, 198*, Article 111200. https://doi.org/10.1016/j.compbiomed.2025.111200

Fridkin, S., & Bendersky, M. (2026). Interpretable machine learning: A comprehensive review of foundations, methods, and the path forward. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 16*(1), Article e70075. https://doi.org/10.1002/widm.70075

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics, 29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Guidotti, R. (2024). Counterfactual explanations and how to find them: Literature review and benchmarking. *Data Mining and Knowledge Discovery, 38*, 2770–2824. https://doi.org/10.1007/s10618-022-00831-6

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys, 51*(5), Article 93. https://doi.org/10.1145/3236009

Hong, S., & Park, W. (2025). Developing user-centered system design guidelines for explainable AI: A systematic literature review. *Artificial Intelligence Review, 58*, Article 386. https://doi.org/10.1007/s10462-025-11363-y

Jain, S., & Wallace, B. C. (2019). Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 3543–3556). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1357

Lin, T., Lau, R. Y. K., & Hu, S. (2026). A critical review of state-of-the-art explainable artificial intelligence (XAI) methods and their business applications. *Artificial Intelligence Review, 59*, Article 161. https://doi.org/10.1007/s10462-026-11565-y

Lipton, Z. C. (2018). The mythos of model interpretability. *ACM Queue, 16*(3), 31–57. https://doi.org/10.1145/3236386.3241340

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 4765–4774).

Lyu, Q., Apidianaki, M., & Callison-Burch, C. (2024). Towards faithful model explanation in NLP: A survey. *Computational Linguistics, 50*(2), 657–723. https://doi.org/10.1162/coli_a_00511

Meng, H., Wagner, C., & Triguero, I. (2024). SEGAL time series classification—Stable explanations using a generative model and an adaptive weighting method for LIME. *Neural Networks, 176*, Article 106345. https://doi.org/10.1016/j.neunet.2024.106345

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence, 267*, 1–38. https://doi.org/10.1016/j.artint.2018.07.007

Mohale, V. Z., & Obagbuwa, I. C. (2025). A systematic review on the integration of explainable artificial intelligence in intrusion detection systems to enhancing transparency and interpretability in cybersecurity. *Frontiers in Artificial Intelligence, 8*, Article 1526221. https://doi.org/10.3389/frai.2025.1526221

Molnar, C. (2022). *Interpretable machine learning: A guide for making black box models explainable* (2nd ed.).

National Institute of Standards and Technology. (2023). *Artificial intelligence risk management framework (AI RMF 1.0)* (NIST AI 100-1). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.100-1

Naveed, S., Stevens, G., & Robin-Kern, D. (2024). An overview of the empirical evaluation of explainable AI (XAI): A comprehensive guideline for user-centered evaluation in XAI. *Applied Sciences, 14*(23), Article 11288. https://doi.org/10.3390/app142311288

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939778

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence, 32*(1), 1527–1535. https://doi.org/10.1609/aaai.v32i1.11491

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*, 206–215. https://doi.org/10.1038/s42256-019-0048-x

Salih, A. M., Raisi-Estabragh, Z., Galazzo, I. B., Radeva, P., Petersen, S. E., Lekadir, K., & Menegaz, G. (2025). A perspective on explainable artificial intelligence methods: SHAP and LIME. *Advanced Intelligent Systems, 7*(1), Article 2400304. https://doi.org/10.1002/aisy.202400304

Saw, S. N., Yan, Y. Y., & Ng, K. H. (2025). Current status and future directions of explainable artificial intelligence in medical imaging. *European Journal of Radiology, 183*, Article 111884. https://doi.org/10.1016/j.ejrad.2024.111884

Schwalbe, G., & Finzel, B. (2024). A comprehensive taxonomy for explainable artificial intelligence: A systematic survey of surveys on methods and concepts. *Data Mining and Knowledge Discovery, 38*, 3043–3101. https://doi.org/10.1007/s10618-022-00867-8

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 618–626). https://doi.org/10.1109/ICCV.2017.74

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 3319–3328). PMLR.

Wachter, S., Mittelstadt, B., & Russell, C. (2018). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology, 31*(2), 841–887.

Wang, Y., Zhang, T., Guo, X., & Shen, Z. (2024). Gradient based feature attribution in explainable AI: A technical review. *arXiv*. https://doi.org/10.48550/arXiv.2403.10415

CHAPTER 3

IDENTIFYING SIMILARITIES BETWEEN PROPOSED AND EXISTING COURSES IN GRADUATE PROGRAMMES USING NATURAL LANGUAGE PROCESSING

EBRU YILMAZ İNCE[1]
Murat İNCE[2]

Introduction

Universities have become heavy producers of text, not just teachers of it. Course descriptions, programme structures, syllabi, learning outcomes and a thick layer of institutional paperwork accumulate year on year. Once that material is read as data rather than stored and forgotten, the academic processes behind it can be managed more deliberately and more sustainably (Jurafsky & Martin, 2020; Romero & Ventura, 2020; Aldowah et al., 2019).

[1] Doç.Dr., Isparta University of Applied Sciences, Computer Technologies, Orcid: 0000-0001-9462-0363

[2] Doç.Dr., Isparta University of Applied Sciences, Computer Technologies, Orcid: 0000-0001-5566-5008

Graduate programmes sit at the sharp end of this. Their content has to track a moving research frontier, so academics put forward new courses each term and the relevant boards weigh them up. A proposal that repeats too much of an existing course threatens the diversity a programme is supposed to protect, which is why the overlap is held below a set ceiling. In the institutional context examined here that ceiling is 70%; a proposal whose content overlaps an existing course at or above this level is sent back for review rather than added as it stands.

In practice that check still falls to board members reading side by side. In a large graduate institute, with well over a hundred courses already on the books, the comparison is slow, tiring and open to the quirks of individual judgement. An automatic first pass would relieve all three pressures, and natural language processing is well suited to supply one.

Measuring how alike two texts are a staple of the field. The literature usually splits the methods in two; lexical measures, which count shared words and their forms, and semantic measures, which weigh the contextual meaning of words and so read content more deeply (Gomaa & Fahmy, 2013; Wang & Dong, 2020; Chandrasekaran & Mago, 2021). The present study works at the lexical and subword end of that spectrum, where methods are fast, transparent and easy to reproduce.

This chapter applies that toolkit to a concrete institutional question. We compared 132 analysed existing courses with 31 proposed in 2024 at a university graduate institute, running each proposal through three measures, Jaccard overlap, TF-IDF cosine similarity and a character n-gram cosine measure, and reading the results against the institutional 70% threshold. The aim is not to make the final call, which stays with curriculum committees, but to show that a modest multi-measure screen can give a fast, repeatable

first pass that flags proposals for expert review. Three questions guide the work;

RQ1. How similar are newly proposed graduate courses to existing courses under lexical and subword similarity measures?

RQ2. How consistently do Jaccard, TF-IDF cosine and character n-gram cosine identify potentially overlapping courses?

RQ3. Can a multi-measure similarity screen support the preliminary review of proposals against an institutional similarity threshold?

The remainder of the chapter reviews the relevant background, sets out the methods and dataset, presents and discusses the findings, and closes with the limitations and the implications for programme development.

Text Similarity and Its Role in Education

Lexical similarity grew out of the vector space model, where a document becomes a vector of weighted terms (Salton et al., 1975; Salton & McGill, 1983). TF-IDF supplies the weights, rewarding terms that are frequent in one document but rare across the collection (Spärck Jones, 1972; Ramos, 2003), and the cosine of two such vectors gives a tidy overlap score. The Jaccard index takes an even simpler view, dividing the words two texts share by the words they use between them (Arts et al., 2018; Tan et al., 2006). Character n-gram representations sit alongside these; by breaking text into overlapping character sequences, they pick up subword, orthographic and morphological overlap, which is helpful in a language where a single stem spawns many surface forms. Such measures are fast and transparent, and they remain a sensible first pass for near-duplicate detection (Manning et al., 2008).

Semantic measures try to see past the exact words. Distributional embeddings such as Word2Vec and GloVe place related terms close together in a dense space (Mikolov et al., 2013; Pennington et al., 2014), latent semantic analysis recovers meaning from co-occurrence structure (Landauer et al., 1998), and corpus- and knowledge-based measures combine several signals at once (Mihalcea et al., 2006). The Transformer, and BERT in particular, pushed this further by reading context from both directions (Vaswani et al., 2017; Devlin et al., 2019); sentence-level encoders then made whole-text comparison practical (Reimers & Gurevych, 2019; Cer et al., 2018), while a growing body of analysis has unpicked what these models learn (Rogers et al., 2020; Sitikhu et al., 2019). These contextual methods are powerful but heavier to deploy, and the present study deliberately stays with lighter lexical and subword measures, returning to contextual models among the directions for future work.

In education, text similarity already underpins academic plagiarism detection, where lexical signals are combined to catch reworded passages (Foltýnek et al., 2019; Alzahrani et al., 2012; Potthast et al., 2010). The same machinery extends naturally to comparing course content for curriculum management. What is far less common is a transparent, multi-measure screen tied to an institutional overlap threshold, and work grounded in Turkish graduate course content is rarer still. This chapter sets out to fill that gap.

Methods

Text Preprocessing

Every course was represented by the concatenation of its title, description and topic list, and this text passed through a conventional cleaning pipeline built with NLTK (Chai, 2023; Bird et al., 2009).

For the word-level measures, Jaccard and TF-IDF cosine, the text was tokenised, lower-cased and stripped of punctuation, Turkish function words were removed with a Turkish stop-word list, and the remaining words were lemmatised so that inflected forms collapse onto a single term. The character n-gram measure, by contrast, was computed on the normalised but non-lemmatised text, so that the subword and morphological cues it is meant to capture were preserved rather than removed. We note in the limitations that NLTK offers only limited morphological support for Turkish, and that a dedicated analyser such as Zemberek would be a natural upgrade.

Similarity Measures

Three measures ran in parallel. The Jaccard index, Eq. (1), treats each course as a set of words and divides the size of their intersection by the size of their union, giving a quick lexical overlap for sets A and B. Because it works on sets, it ignores how often a term repeats, which makes it sensitive to shared vocabulary but blind to emphasis.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \qquad (1)$$

TF-IDF weighting, Eq. (2), scores a term t in document d by its local frequency $tf_{t,d}$ against its rarity across the N documents, where df_t is the document frequency. We used word unigrams and bigrams with sublinear term frequency, smoothed IDF and L2 normalisation.

$$w_{t,d} = tf_{t,d} \cdot log\left(\frac{N}{df_t}\right) \qquad (2)$$

Two courses were then compared by the cosine of their TF-IDF vectors, Eq. (3), which measures the angle θ between vectors A and B and so stays robust to differences in length.

$$cos\ \theta\ =\ \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A^i} \cdot \sqrt{\sum_{i=1}^{n} B^i}} \qquad (3)$$

The third measure is lexical rather than semantic. Each course was turned into character n-gram TF-IDF vectors using within-word n-grams of length two to five (scikit-learn's char_wb analyser) with sublinear term frequency and L2 normalisation, and two courses were compared by the cosine of these vectors. Character n-grams capture subword and orthographic overlap, which matters in an agglutinative language such as Turkish, where a shared stem surfaces in many inflected forms; they do not, however, model meaning in the way a contextual encoder does, and we are careful not to present them as a semantic measure.

Decision Rule

For each proposal we took the highest of its three similarity scores against any existing course as a single screening value and compared that value with the 0.70 threshold. A proposal reaching the threshold on any measure was flagged for expert review; the rest were marked as below the threshold. Two points follow from this design and are important to state plainly. First, the rule is deliberately conservative; because any of the three measures can raise a flag, it favours recall over precision, which suits a first-pass screen whose errors are corrected by human reviewers. Second, the three measures live on different scales and distributions, so applying one threshold across all of them is a screening heuristic rather than a validated cross-metric cut-off; the value of a normalised or weighted combination, and of an empirically validated threshold, is taken up in the limitations.

Dataset

The data come from the course catalogue of a university graduate institute. The catalogue held 136 existing courses; once

four specialisation courses were set aside, 132 entered the analysis, and these 132 were the set each proposal was compared against. Thirty-one courses proposed in 2024 were screened. Each course carried about thirteen topics, for a total of 1,716. Two of the proposals, both in robotics, returned a score of zero on every measure, which indicates that their catalogue records held no extractable content at the time of analysis; these two were set aside as a data-quality exclusion, leaving 29 proposals with usable content for interpretation. Table 1 summarises the dataset.

Table 1. Characteristics of the dataset.

Characteristic	Value
Existing courses in catalogue	136
Existing courses analysed (4 specialisation excluded)	132
Proposed courses (2024)	31
Proposed courses with usable content	29
Average topics per course	13
Total number of topics	1,716

Tools and Environment

The analysis ran on a desktop with a 12th-generation Intel Core i7-12650H CPU at 3.5 GHz and 16 GB of RAM, under Python 3.9. Preprocessing used NLTK (Bird et al., 2009), the similarity computations used scikit-learn (Pedregosa et al., 2011), and the data were handled with pandas (McKinney, 2010).

Results and Discussion

Overall Similarity Distribution

Each of the 29 proposals with usable content was compared against all 132 analysed courses under the three measures, and the maximum similarity per proposal was kept as the screening value. Figure 1 plots the twelve proposals with the highest maximum similarity, and Table 2 lists the full set. One pattern stands out at once; the character n-gram measure tends to return the largest scores,

the TF-IDF cosine sits in the middle, and Jaccard is usually the most conservative, except where two courses share a great deal of literal wording.

Figure 1. Similarity scores of the twelve proposals with the highest maximum similarity, under three measures, against the 0.70 threshold.

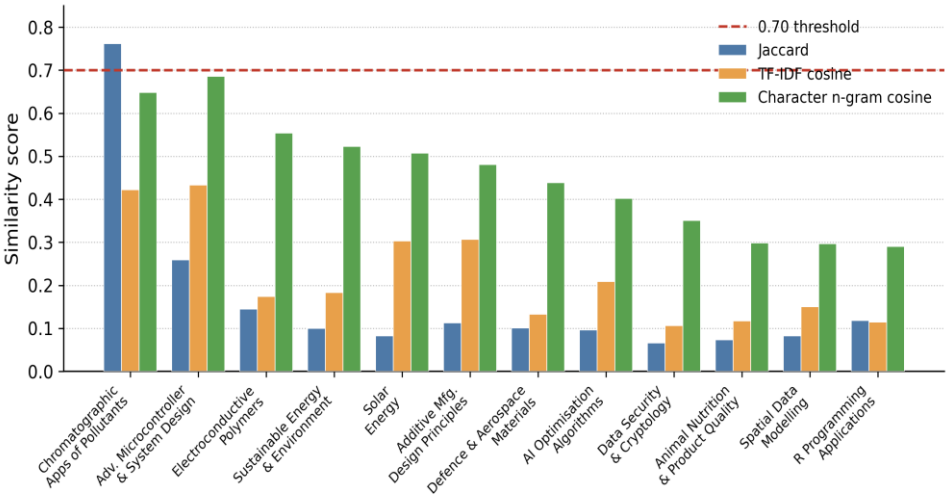


Table 2. Similarity screening of the courses proposed in 2024 (maximum similarity and screening outcome by the 0.70 threshold).

Proposed course	Jaccard	TF-IDF cos.	Char n-gram	Max.	Screening outcome
Chromatographic Applications of Pollutants	0.762	0.422	0.648	0.762	Flag for review
Advanced Microcontroller and System Design Applications	0.259	0.433	0.686	0.686	Below threshold
Electroconductive Polymers	0.145	0.174	0.554	0.554	Below threshold
Sustainable Energy and Environment	0.100	0.183	0.523	0.523	Below threshold
Solar Energy	0.083	0.303	0.507	0.507	Below threshold
Design Principles for Additive Manufacturing	0.113	0.307	0.481	0.481	Below threshold
Defence and Aerospace Materials Technologies and Applications	0.101	0.133	0.439	0.439	Below threshold

Artificial Intelligence Optimisation Algorithms	0.096	0.209	0.402	0.402	Below threshold
Data Security and Cryptology Applications	0.066	0.106	0.351	0.351	Below threshold
Animal Nutrition and Its Effects on Product Quality	0.073	0.117	0.299	0.299	Below threshold
Modelling and Mapping Techniques with Spatial Data Analysis	0.083	0.150	0.297	0.297	Below threshold
R Programming Applications	0.118	0.115	0.290	0.290	Below threshold
Artificial Intelligence Applications with R	0.083	0.114	0.287	0.287	Below threshold
Waste Reduction, Recovery and Energy from Biomass under Zero Waste	0.087	0.116	0.257	0.257	Below threshold
Biofuel Use in Sustainable Development	0.067	0.072	0.253	0.253	Below threshold
Folkloric Use of Turkey's Wild Plants	0.064	0.062	0.253	0.253	Below threshold
Sustainable Urban Area Applications and Urban Transformation	0.079	0.055	0.249	0.249	Below threshold
Spectroscopic Methods in Organic Chemistry	0.076	0.059	0.242	0.242	Below threshold
Methods for Monitoring and Researching Forest Ecosystems	0.051	0.080	0.222	0.222	Below threshold
Fundamentals of Ecological Research in Forest Ecosystems	0.061	0.050	0.198	0.198	Below threshold
Instrumental Analysis Methods	0.123	0.104	0.194	0.194	Below threshold
Advanced Deep Learning Techniques	0.052	0.071	0.192	0.192	Below threshold
Developments in Vegetable Oil Production Technology and Quality Control	0.096	0.032	0.188	0.188	Below threshold
Important Plant Areas of Turkey	0.071	0.089	0.184	0.184	Below threshold
University-Industry Cooperation and Technoparks	0.049	0.057	0.170	0.170	Below threshold
Essential Oil Technology	0.065	0.025	0.158	0.158	Below threshold
Real Estate Valuation	0.095	0.029	0.151	0.151	Below threshold
Microbiology Applications in Biogas Technologies	0.063	0.037	0.148	0.148	Below threshold
Above-ground and Below-ground Interactions	0.063	0.046	0.106	0.106	Below threshold
Robotic Motor Applications	–	–	–	–	No content (excluded)
Robot Motors and Drives Applications	–	–	–	–	No content (excluded)

Threshold Screening

Read against the 0.70 ceiling, the picture is clear. Of the 29 proposals with usable content, exactly one reached the threshold; Chromatographic Applications of Pollutants, which overlapped the existing course Analysis Methods of Pollutants in Water and Agricultural Products at a Jaccard score of 0.762. It was therefore flagged for expert review. The other 28 stayed below 0.70 on every measure. Table 3 lists the five proposals with the highest maximum similarity and the existing course each is closest to.

Table 3. Threshold screening against the 0.70 ceiling (the five proposals with the highest maximum similarity).

Proposed course	**Closest existing course**	**Max.**	**Screening outcome**
Chromatographic Applications of Pollutants	Analysis Methods of Pollutants in Water and Agricultural Products	0.762	Flag for review
Advanced Microcontroller and System Design Applications	Wireless System Design Applications	0.686	Below threshold
Electroconductive Polymers	Polymer Chemistry	0.554	Below threshold
Sustainable Energy and Environment	Alternative Energy Sources	0.523	Below threshold
Solar Energy	Plant Use in Solar Cell Production	0.507	Below threshold

Discussion

The single flagged course is instructive. Its overlap was caught by Jaccard, at 0.762, even though the TF-IDF cosine put it at only 0.422 and the character n-gram measure at 0.648, both below the threshold. The gap between Jaccard and TF-IDF here is itself informative; Jaccard rewards the raw set of shared words, so heavy reuse of the same specialised vocabulary pushes it up, whereas TF-IDF down-weights terms that recur across many course descriptions and so reads the pair as less alike. In practice this means the proposal leaned on much of the existing course's wording without being its statistical twin, and only the measure most sensitive to literal overlap

raised the flag. The practical lesson is that running several measures together guards against the blind spots of any one of them, which is consistent with the mixed-signal approach favoured in plagiarism research (Foltýnek et al., 2019).

More broadly, the 28 proposals that fell below the threshold are better described as below the textual-similarity ceiling than as original; the measures compare wording, not learning outcomes or pedagogical intent, so two courses can read as distinct yet still share substantial academic ground, and the reverse is also possible. Read this way, the screen says which proposals need a closer human look, not which are genuinely novel. The two robotics proposals that returned zero on every measure are a reminder of the same point from the data side; their records held no extractable content, so the zeros reflect missing input rather than true distinctiveness, and they were set aside rather than interpreted.

Limitations and Future Work

Several limitations frame these results, and each point to a clear next step. The study covers one graduate institute in a single proposal cycle, so it is best read as a practical case study rather than a general model. No independent expert ground truth was collected, so the screen is validated against the institutional threshold rather than against human judgements of overlap; a small annotation study, in which two or three academics rate the top candidate pairs and agreement is measured with Cohen's or Fleiss' kappa, would let precision, recall and F1 be reported and would test whether 0.70 is the right line to draw. The decision rule takes the maximum across three measures that sit on different scales, so a normalised or weighted combination, or a learned ensemble, would rest on firmer ground than a shared threshold. The character n-gram measure captures subword and orthographic overlap but not meaning; adding a genuine semantic measure, a Turkish or multilingual sentence

transformer such as Sentence-BERT or a BERTurk-based encoder (Reimers & Gurevych, 2019; Devlin et al., 2019; Schweter, 2020), is the most promising extension and would let lexical, subword and semantic signals be compared directly. Finally, the Turkish preprocessing relied on NLTK, whose morphological support for Turkish is limited; a dedicated analyser such as Zemberek would sharpen lemmatisation, and widening the corpus across institutions and cycles would test how far the approach travels.

Conclusion

This chapter set out to screen graduate course proposals for content overlap, using natural language processing in place of a purely manual first pass. Across 132 analysed existing courses and 31 proposed in 2024, three measures, Jaccard overlap, TF-IDF cosine similarity and a character n-gram cosine measure, were combined and read against the institutional 70% threshold. Of the proposals with usable content, only one reached the threshold, overlapping an existing course at a Jaccard score of 0.762 and so warranting expert review; the rest fell below it.

The wider value lies in the framing as much as the finding. A multi-measure similarity screen gives academic boards a fast, consistent and reproducible first pass that prioritises proposals for human review, rather than a verdict that replaces it. The right picture is human-in-the-loop: the screen flags candidates, and the committee decides. Strengthened with expert validation, a principled way of combining measures and a genuine semantic model, and extended beyond a single institution and language, the approach could become a standing aid in curriculum development, turning a slow manual chore into a routine, data-informed step.

References

Aldowah, H., Al-Samarraie, H., & Fauzy, W. M. (2019). Educational data mining and learning analytics for 21st century higher education: A review and synthesis. Telematics and Informatics, 37, 13–49.

Alzahrani, S. M., Salim, N., & Abraham, A. (2012). Understanding plagiarism linguistic patterns, textual features, and detection methods. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 42(2), 133–149.

Arts, S., Cassiman, B., & Gomez, J. C. (2018). Text matching to measure patent similarity. Strategic Management Journal, 39(1), 62-84.

Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media.

Cer, D., Yang, Y., Kong, S.-Y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., & Kurzweil, R. (2018). Universal sentence encoder. arXiv:1803.11175.

Chai, C. P. (2023). Comparison of text preprocessing methods. Natural Language Engineering, 29(3), 509–553.

Chandrasekaran, D., & Mago, V. (2021). Evolution of semantic similarity: A survey. ACM Computing Surveys, 54(2), 1–37.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT 2019 (pp. 4171–4186). Association for Computational Linguistics.

Foltýnek, T., Meuschke, N., & Gipp, B. (2019). Academic plagiarism detection: A systematic literature review. ACM Computing Surveys, 52(6), 1–42.

Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. International Journal of Computer Applications, 68(13), 13–18.

Jurafsky, D., & Martin, J. H. (2014). Speech and language processing (Vol. 3, pp. 2-5). London: Pearson.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. Discourse Processes, 25(2–3), 259–284.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.

McKinney, W. (2010). Data structures for statistical computing in Python. In Proceedings of the 9th Python in Science Conference (pp. 56–61).

Mihalcea, R., Corley, C., & Strapparava, C. (2006). Corpus-based and knowledge-based measures of text semantic similarity. In Proceedings of AAAI 2006 (pp. 775–780). AAAI Press.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv:1301.3781.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In Proceedings of EMNLP 2014 (pp. 1532–1543). Association for Computational Linguistics.

Potthast, M., Stein, B., Barrón-Cedeño, A., & Rosso, P. (2010). An evaluation framework for plagiarism detection. In Proceedings

of COLING 2010 (pp. 997–1005). Association for Computational Linguistics.

Ramos, J. (2003). Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning (Vol. 242, No. 1, pp. 29-48).

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of EMNLP-IJCNLP 2019 (pp. 3982–3992). Association for Computational Linguistics.

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. Transactions of the Association for Computational Linguistics, 8, 842–866.

Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. WIREs Data Mining and Knowledge Discovery, 10(3), e1355.

Salton, G., & McGill, M. J. (1983). Introduction to modern information retrieval. McGraw-Hill.

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613–620.

Schweter, S. (2020). BERTurk: BERT models for Turkish. Zenodo. https://doi.org/10.5281/zenodo.3770924

Sitikhu, P., Pahi, K., Thapa, P., & Shakya, S. (2019). A comparison of semantic similarity methods for maximum human interpretability. In 2019 Artificial Intelligence for Transforming Business and Society (AITB) (pp. 1–4). IEEE.

Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28(1), 11–21.

Tan, P.-N., Steinbach, M., & Kumar, V. (2006). Introduction to data mining. Pearson Education.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems 30 (pp. 5998–6008).

Wang, J., & Dong, Y. (2020). Measurement of text similarity: A survey. Information, 11(9), 421.

CHAPTER 4

DIGITAL PHENOTYPING OF MIGRAINE: PERSONALIZED MONITORING, RISK MODELING, AND DECISION SUPPORT WITH ARTIFICIAL INTELLIGENCE AND ENGINEERING APPROACHES

Şule YÜCELBAŞ[1]
Cüneyt YÜCELBAŞ[2]

INTRODUCTION

The Clinical and Societal Significance of Migraine

Migraine is a primary headache disorder whose symptoms vary considerably from one person to the next. Besides headache, an attack may bring nausea, sensitivity to light and sound, worsening with physical activity and, in some people, transient neurological aura symptoms. The International Classification of Headache Disorders defines migraine as migraine without aura, migraine with aura, and related subtypes. Assessment depends on the features of the pain, but also on attack frequency, duration, and accompanying

[1] Assoc.Prof.Dr., Tarsus University, Department of Computer Engineering, Orcid: 0000-0002-6758-8502
[2] Assoc.Prof.Dr., Tarsus University, Department of Electronics and Automation, Orcid: 0000-0002-4005-6557

symptoms (HCCIHS, 2018). This classification is the basic reference for defining clinical labels in AI-based analyses. Operational prediction targets, such as an attack onset derived from a diary, still have to be defined separately (HCCIHS, 2018; Ferrari et al., 2022).

The Global Burden of Disease 2023 analysis estimated that headache disorders affect billions of people worldwide, and that migraine accounts for a considerable share of the years lived with disability (GBD, 2025a). An analysis based on GBD 2021 data reported that migraine prevalence rose from roughly 733 million cases in 1990 to 1.16 billion in 2021 (Lu et al., 2025). Figures like these come from models, and "cases" should not be read as the number of people with a confirmed individual diagnosis. Even so, the findings point to a real need for monitoring methods that are accessible and sustainable.

A defining feature of migraine is that symptoms shift over days and even hours. In a clinical interview, retrospective recall can distort when an attack began, what may have triggered it, which medication was taken, and how much daily functioning was lost. A regular headache diary can help a person and their healthcare professional see patterns and treatment response more clearly (American, 2025). Digital diaries can make recording easier, but using an app is not in itself evidence of clinical benefit or diagnostic accuracy (Ferrari et al., 2022; American, 2025).

Defining the Problem From an AI and Engineering Perspective

The role of engineering and Artificial Intelligence (AI) in migraine research goes beyond training a classifier. The real problem is to bring together data from different sources, with different timestamps and degrees of reliability, under a meaningful representation; to identify clinically useful patterns; and to turn the output into a safe decision process. It therefore helps to keep the following target tasks separate:

- distinguishing symptom patterns that support a migraine diagnosis,
- predicting a person's attack probability over a defined time horizon,
- estimating attack duration or functional loss,
- tracking change after a treatment or lifestyle modification,
- producing a summary that a clinician can readily understand.

These tasks differ in their labels, time horizons, and error costs, so evaluating them against a single "success rate" would be inappropriate.

The perspective offered in this chapter is to treat migraine not as a fixed disease label but as a "digital phenotype" that changes over time within a person's context. A migraine-based digital phenotype can be thought of as a representation built by bringing together, over time, self-reported symptoms, sleep and activity patterns, environmental context, and, where appropriate, wearable measurements. It does not replace a clinical diagnosis. The aim is not to reduce a person to data but to look more carefully at temporal relationships that a clinical interview may miss. From this standpoint, the chapter examines how migraine can be monitored digitally using mobile diaries, wearables, and environmental data. It discusses what time-series analysis, machine learning, and explainable AI might contribute, along with data processing and evaluation procedures. Three cautions stay in view throughout: missing or biased data can produce misleading predictions, the relationships identified cannot be read as direct cause and effect, and digital systems cannot stand in for clinical diagnosis. No experiment with new patient data and no clinical performance result is presented

here. Instead, drawing on the literature, the chapter proposes a conceptual system architecture that accounts for person-specific change and prediction uncertainty, respects privacy, and supports clinician judgment, together with an evaluation framework for future research.

MATERIALS AND METHODS

Scope and Methodological Approach

This chapter should not be taken for a systematic review or an experimental study run on new patient data. It is a narrative work that proposes a research design, drawing on clinical classification, global burden studies (GBD, 2025a; Lu et al., 2025; GBD, 2018; GBD, 2025), and sources on digital headache diaries. The architecture, tables, and evaluation plan below are a proposed methodological framework, not experimental performance results. A systematic assessment of the evidence would call for a separate systematic review, with a predefined search strategy, inclusion and exclusion criteria, independent assessment, and proper PRISMA reporting (Page et al, 2021). The proposed reporting and evaluation approach could later be elaborated in line with the TRIPOD+AI and PROBAST+AI principles (Collins et al., 2024; Moons et al., 2025).

Proposed Research Questions

A study proposed on this topic could reasonably be built around questions such as the following:

- Does a person's own diary data calibrate individual attack risk better than a model trained only at the population level?
- Does adding context on sleep, stress, the menstrual cycle, weather, and medication use improve prediction meaningfully compared with pain records alone?

- When missing data and logging behavior are added to the model, are predictions more reliable, or does logging frequency simply become a proxy for disease severity?
- Does an interface that is explainable and reports uncertainty make it easier for clinicians to assess the output?

These questions can be expanded or revised according to the aim, scope, and intended audience of the research. It would be best to settle that together with experts in the field.

Data Sources and the Data Dictionary

The minimum dataset is a headache diary containing date and time, headache onset and end, pain intensity, accompanying symptoms, daily functional loss, and the acute medication used. Optional context fields may cover sleep duration and quality, self-reported stress level, skipped meals, fluid intake, physical activity, menstrual information, and environmental conditions. Sleep or activity data from smartwatches should be used only when the device's measurement limits are clearly documented. Physiological signals such as heart rate are not specific to migraine, and they should not be presented as a diagnostic biomarker on their own. Table 1 summarizes the main data domains that can be used in digital phenotyping of migraine, example variables, and what to watch for when interpreting them. This matters most when self-report and sensor measurements are considered together: neither should be read as a migraine-specific biomarker.

Table 1 Data sources and variables that can be used in digital phenotyping of migraine

Data domain	Example variables	Measurement format	Main risk / point of caution
Symptom diary	Onset and end, intensity, nausea, light sensitivity	Timestamped self-report	Recall error; missing entries
Functional loss	Attendance at work or school, reduced daily activity	Likert scale or standardized instrument	Comparability across individuals
Medication use	Drug class, time, dose (where appropriate)	Event log	Clinical supervision and protection of sensitive data
Sleep/activity	Sleep duration, awakenings, step and activity summary	Wearable device or self-report	Differences between devices; measurement error
Context	Stress, meals, menstruation, weather	Self-report / external data	Confounding variables; causal fallacy
Logging behavior	Skipped days, delayed entries, app usage	System metadata	Digital access and participation bias

The data dictionary should define each variable's unit, permitted values, source, temporal resolution, missingness code, and clinical meaning. For "attack onset," for example, it must be decided in advance whether the reference point is the moment the person first notices a symptom or the moment the headache begins. If information gathered after the time of prediction is used as model input, data leakage results and the model's performance is overestimated. Model inputs should therefore be limited to what is actually available when the prediction is made. Only then can performance under real-world conditions be assessed reliably and without bias.

Data Processing Pipeline

The proposed pipeline has six main stages: data collection and informed consent, quality control and time alignment, feature extraction, training/validation/test splitting, modeling and calibration, and clinical evaluation and reporting. If records from the same person are scattered at random across training and test sets, the model can pick up person-specific traits and its apparent accuracy will be inflated. To evaluate how well a model performs on other

people, each participant's data should therefore sit in only one set. When the goal is instead to predict the same person's future attacks, earlier records should be used for training and later ones for testing. The two approaches answer different questions, so their results should be evaluated separately. Where possible, validation on independent data is also advisable, to see whether the model holds up across different patient groups and devices. Splits that prevent leakage, along with transparent reporting, matter for judging how a clinical prediction model will perform in practice (Collins et al., 2024; Wolff et al., 2019; Collins et al., 2015; Stubberud et al., 2023). One small prospective study used mobile diary entries and physiological measurements to predict next-day headache. With data from 18 participants, its best model reached a test ROC-AUC of 0.62. That modest performance shows the approach cannot yet be considered a dependable clinical forecasting system, and that it needs independent validation (Stubberud et al., 2023).

Missing or faulty columns in a person's data rows should not be filled automatically with zeros. In this setting, a person reporting no pain and a person making no entry for that day are different situations. Treating days without records as pain-free puts false information into the dataset and can hurt predictive performance. Missingness indicators should be kept separate. Where appropriate, multiple imputation or imputation methods learned on the training set alone should be used, and sensitivity analyses should be run under different missingness scenarios. Scaling, feature selection, balancing, and hyperparameter tuning must likewise be done within the training stage only. Otherwise, information from the test data leaks into the model and produces misleading output.

Feature Extraction and Model Families

For tabular data, logistic regression and regularized linear models are strong, interpretable starting points. Tree-based methods

such as random forests and gradient boosting can capture nonlinear relationships. Recurrent neural networks or Transformer-based time-series models may be worth exploring when there is enough good-quality timestamped data, but complex models should not be assumed to do better on small datasets. When there is little data per person, hierarchical models or adaptation to a personal baseline deserve consideration in place of added complexity.

Features can include not only raw values but also change relative to the person. Examples are the change in sleep over the previous 24–72 hours, deviation from the person's own average, attack frequency in the past week, regularity of logging over time, and lagged values of environmental variables. This approach accepts that "little sleep" in the general population and sleeping less than usual for one particular person may not mean the same thing. Even so, such features should be reported with a clear line between those justified clinically in advance and those that came out of exploratory analysis. Table 2 compares, for different prediction tasks, example targets, the model families that can be tried first, and suitable performance measures. As the type of prediction task changes, so should the evaluation approach.

Table 2 Possible prediction tasks, model families, and evaluation measures

Task	Example target	Suitable starting models	Recommended evaluation
Attack probability	Attack onset in the next 24 hours	Logistic regression, gradient boosting	PR-AUC, Brier score, calibration
Attack duration	Duration in hours	Regression, survival analysis	MAE; survival metrics if data are censored
Functional loss	Functional impact the next day	Ordinal logistic model, tree-based model	Class-wise sensitivity, macro-F1
Personal change	Trend before and after treatment	Interrupted time series, mixed-effects model	Effect size and confidence interval
Clinical summary	Weekly symptom and medication pattern	Rule-based summary and visualization	Clinician usability and error analysis

Evaluation and Validation

When classes are imbalanced, accuracy alone misleads. In a dataset where attacks are rare, a model that always says “no attack” can reach high accuracy. Sensitivity, specificity, precision, F1, and ROC-AUC should therefore be examined, and, especially when the positive class is rare, PR-AUC and calibration too. For clinical use, what matters is whether a predicted 30% risk actually materializes about 30% of the time in similar cases. The Brier score and calibration plots help here. Decision curve analysis can ask whether a model offers net benefit at various thresholds, compared with the options of “doing nothing” or “intervening for everyone.” Calibration captures a dimension distinct from discrimination, and it needs separate examination if predicted probabilities are to be interpretable (Van Calster et al., 2019).

Confidence intervals should be computed with suitable methods such as person-level resampling; the many days belonging to one person must not be treated as independent samples. Subgroup analyses may cover age, sex, device type, aura status, and digital literacy, but unstable results in small subgroups should not harden into firm conclusions. A model failing in one subgroup is not merely a statistical problem; it is a problem of access and equity. Systematic review of model development and evaluation for bias and applicability is also recommended (Moons et al., 2025; Wolff et al., 2019).

PROPOSED SYSTEM ARCHITECTURE AND EVALUATION FRAMEWORK

End-to-End Architecture

In the proposed architecture, a mobile app or web interface forms the data-entry layer; validation and timestamp services form the data-quality layer; encrypted storage and access control form the security layer; feature generation and model serving form the

analytics layer; and visualization and clinician approval form the decision-support layer. Data processing can be done on the device as far as possible. If transfer to a central server is needed, only the fields required for the purpose should be sent, identifiers should be separated from analytic records, and a retention period should be set. Figure 1 shows the proposed system architecture for digital phenotyping of migraine. The architecture offers an integrated workflow that starts with data collection and preprocessing and extends to personalized prediction, model validation, and clinical decision support. Data security and ethical governance are handled across the system's whole life cycle, not only at the final stage.

Figure 1 Proposed conceptual system architecture for digital phenotyping of migraine and personalized clinical decision support



Table 3 presents the main architectural layers of the proposed system, each layer's responsibilities, and the engineering controls that can be applied if something fails. Security and privacy measures should not be confined to a single layer; they need to run through the whole system.

Table 3 Architectural layers and controls of the proposed digital phenotyping system

Layer	Responsibility	Engineering control	In case of failure
User interface	Recording symptoms and context	Accessible design, low logging burden	Short form / offline entry
Data validation	Unit, time, and range checks	Schema validation, consistency rules	Prompt the user to correct
Storage	Protecting records	Encryption, role-based access, audit trail	Restrict access, log the incident
Analytics	Feature generation and risk prediction	Versioning, leakage tests	Withhold the prediction and give the reason
Decision support	Summary and uncertainty display	Calibration, explanation, clinician check	Halt the automated decision
Monitoring	Model and data drift	Performance and subgroup tracking	Revalidate or roll back

Predictions the system produces should not be treated as a definitive diagnosis or as showing what caused a migraine attack. Instead of saying “lack of sleep caused the migraine attack,” for example, the system can note that attacks were more frequent in periods when the person’s records showed shorter sleep. That does not prove that lack of sleep directly causes attacks. Too many alerts can also cause needless anxiety, so their frequency and content need careful tuning. Every prediction should state which time window was assessed, how reliable the result is, and which decisions must not be made on the basis of the prediction. In short, the system’s purpose should not be to make a definitive diagnosis or steer treatment decisions. It should support clinical decision-making by giving users clear and dependable information for tracking attacks and assessing possible risks.

Reporting and Interpretation Limits for Future Research

No patient data were collected and no machine learning algorithm was trained for this chapter. Accordingly, no numerical results on accuracy, AUC, or clinical impact are given. What the chapter covers instead are the key considerations, grounded in the

existing literature, for developing a system that predicts migraine attacks: defining attacks accurately from a clinical standpoint, evaluating data with person and time characteristics in mind, handling missing records properly, examining how reliable the predictions are, and protecting personal data. Whether the proposed approach is useful in real-world conditions can be established only by testing it on independent data under a protocol prepared in advance.

In the results section of a study, the number of participants, the days over which records were collected, and the data collection process should be shown in a flow diagram. Missing-data rates and the distribution of migraine attacks should also be described. The performance of the models developed should be compared, and results should be assessed across both different participants and different time periods. It is also necessary to examine how closely the model's predicted risk levels match actual outcomes, in what situations wrong predictions occur, and how the model performs in different participant groups. Clinicians' and users' assessments of the system should be included as well. A model's success should be judged not only by overall accuracy but by its errors: attacks it fails to predict, and warnings it gives when no attack will occur. That gives a fuller picture of the system's strengths and limits. Current reporting guidelines for clinical prediction models should be followed when reporting results (Collins et al., 2024; Liu et al., 2020; Cruz Rivera et al., 2020).

DISCUSSION

The Contribution of the Digital Phenotype Approach

Digital monitoring is a valuable tool that helps clinicians assess patients' symptoms and changes in health status more fully during visits. When attack onset, accompanying symptoms, medication timing, and functional loss are recorded regularly, the

clinician is likely to see a more concrete pattern. The purpose of a headache diary is not to generate as much data as possible but to collect, in a sustainable way, reliable information that can influence treatment decisions (American, 2025). The best application may therefore be not the one that uses the most sensors but the one that gets meaningful data with the least burden on the person.

Another point in digital migraine tracking is that missing records should not be seen only as lost data; their possible causes matter too. Someone may be unable to log on days when the attack is severe. Another person may forget to open the app on symptom-free days. Missingness, in other words, may not be random at all but tied to disease severity. Filling such gaps with zeros, or evaluating only the days with complete records, can create systematic bias. Data collection should accordingly be designed with short entries that reduce user burden and an interface that does not penalize skipped entries. Users should also be told, briefly and plainly, about situations like these.

Causality, Triggers, and Personalization

Variables linked to migraine, such as sleep, stress, meals, hormones, or weather, can change together in observational records. Observing a variable before an attack does not by itself show that it caused the attack. Early prodromal symptoms, for instance, may have changed the person's sleep, appetite, or activity. In that case the relationship we see may reflect an early sign of an approaching attack more than a trigger. This distinction is critical if a prediction model is not to be read as a causal explanation.

Personalization raises some points that need care. A model built for one person can predict well from that individual's past data. Over time, though, the person's living conditions, medications, data-collection devices, and logging habits can change, and these changes can affect the model's performance. Performance should therefore

be monitored regularly, and retraining should be considered once enough reliable new data has accumulated. A retrained model should be tested on independent data before it goes into use. The cases where the model fails to predict migraine attacks, or warns although no attack will occur, should also be examined. Such errors may cause needless anxiety, create false reassurance, or affect decisions about seeking healthcare, and the likelihood of each should be assessed. Once the necessary checks are complete, changes to the model should be documented and the updated version released. Newly collected data should likewise not be assumed to be always correct; its accuracy and reliability should be checked before it is used in retraining.

Ethics, Privacy, and Security

Migraine diaries can contain personal and sensitive health information, such as patients' symptoms, medications, and reproductive health. Data privacy and security should therefore be considered from the design stage of digital monitoring systems onward. To begin with, only the data needed for the purpose of the research should be collected, participants should be informed clearly and understandably about how their data will be used, and the necessary consent should be obtained. How long data will be kept should be determined, security measures such as encryption and access control applied, and deletion made possible when needed. Anonymization may also fall short of removing the risk of re-identification entirely. The timing of records and rare data patterns in particular can lead to indirect identification. To reduce these risks, approaches such as secure analysis environments, aggregated results, and, where appropriate, federated learning can be considered when sharing data. Federated learning, however, does not guarantee data privacy on its own. Data security in digital migraine monitoring systems should therefore not be limited to technical measures; it should be addressed together with ethical principles such as

individual privacy, autonomy in decision-making, transparency, and accountability. International ethics guidance on AI in health also stresses the importance of these principles (WHO, 2021).

Developers of digital migraine monitoring systems also have to consider users' differing access to technology. It matters that people who do not use smartwatches, lack regular internet access, or find digital record-keeping difficult can still benefit from the system. Apps should therefore be designed, as far as possible, to work without extra devices and to be easy to use in their basic features. Nor should the use of AI-supported prediction systems become a precondition for receiving healthcare. These systems cannot replace clinical assessment, and that should not be forgotten. Headaches that begin suddenly, are very severe, or differ from a person's usual migraine symptoms can signal other health problems. Because digital systems cannot reliably identify the cause in such cases, users should be told clearly to seek medical assessment and emergency care when needed.

Limitations

The chapter's main limitation is that it rests neither on a systematic literature search nor on an experimental application supported by clinical data. The sources used establish the societal impact and clinical features of migraine, but they do not show how useful the proposed system would be under real-world conditions. The data that digital monitoring systems can draw on have their own limits as well. Self-reported records, for example, can be affected by misremembered past symptoms or by personal expectations. Measurements from smartwatches and similar wearables vary with the device's characteristics and the data-processing methods. Models built on data from a single healthcare institution or from volunteer participants alone cannot be expected to perform equally well across age groups, populations, and health systems. The reliability of the

proposed approach should therefore be evaluated in different participant groups and on independent data. Finally, a model reaching high predictive performance does not mean that patients' health or quality of life has improved. Showing such benefits in a meaningful way requires separate clinical evaluation.

A Roadmap for Future Research

In the first stage, a diary containing a small number of clinically meaningful variables should be developed through co-design with patients and clinicians. In the second, a prospective pilot study should examine usability, logging adherence, and data quality. In the third, simple baseline models and more complex ones should be compared under a preregistered analysis plan, with person-wise and time-wise testing. The next stage should carry out external validation and decision curve analysis at different centers. The final stage should be an impact study asking whether the system improves clinical decisions, quality of life, or the care process. Until these stages are complete, the application should not be described as one that "diagnoses migraine" or "prevents attacks." If a clinical AI intervention is later to be evaluated prospectively, the protocol could be prepared following the SPIRIT-AI recommendations and the results report following CONSORT-AI (Liu et al., 2020; Cruz Rivera et al., 2020).

CONCLUSION

AI and engineering approaches can contribute to migraine management through data integration, time-series analysis, personalized risk prediction, and clinical decision support. But a dependable solution is more than a high-performing algorithm. Clinically meaningful labels, correct handling of missing records, person-wise and time-wise independent validation, calibration, explainability, privacy, and equitable access have to be designed together. The central proposal of this chapter is to build a digital

phenotyping system that states its uncertainty openly and supports the conversation between clinician and patient, rather than an AI that collapses migraine into a single class. Such an approach uses sensor data not as an end in itself but to understand the individual's experience better and to make care decisions better informed. The clinical value of the proposed framework can be shown only through transparent, pre-planned, and independently validated research.

Table 4 brings together the key elements that a future study should report, from cohort and label definitions to validation, performance, fairness, usability, and safety, as a checklist. The list can be used alongside reporting and bias-assessment guidelines for clinical prediction models (Collins et al., 2024; Moons et al., 2025; Wolff et al., 2019; Collins et al., 2015; WHO, 2021).

Table 4 Minimum reporting checklist for the proposed research

Domain	Information to be reported
Cohort	Inclusion criteria, centers, demographic distribution, participants lost
Label	Attack definition, prediction horizon, clinical assessment method
Data quality	Missingness, timestamp consistency, sensor type and version
Modeling	Baseline model, feature selection, hyperparameter search, leakage safeguards
Validation	Person-wise, time-wise, and, where possible, external-center testing
Performance	Sensitivity, specificity, PR-AUC, calibration, confidence intervals
Fairness	Subgroup results and possible differences in data access
Usability	Logging burden, user dropout rate, clinician feedback
Safety	Consent, access control, retention period, incident response
Interpretation	Error analysis, uncertainty, clinical limits and intended use

Acknowledgements

The authors used ChatGPT (OpenAI) to assist with translation, language editing, improving the clarity and readability of the manuscript, and the generation and editing of Figure 1. All scientific content was reviewed, edited, and approved by the authors.

REFERENCES

American Migraine Foundation. (2025, April 8). Headache journals: Tracking your migraine. https://americanmigrainefoundation.org/resource-library/headache-journals/

Collins, G. S., Reitsma, J. B., Altman, D. G., & Moons, K. G. M. (2015). Transparent reporting of a multivariable prediction model for individual prognosis or diagnosis (TRIPOD): The TRIPOD statement. British Journal of Surgery, 102(3), 148–158. https://doi.org/10.1002/bjs.9736

Collins, G. S., Moons, K. G. M., Dhiman, P., Riley, R. D., Beam, A. L., Van Calster, B., et al. (2024). TRIPOD+AI statement: Updated guidance for reporting clinical prediction models that use regression or machine learning methods. BMJ, 385, Article e078378. https://doi.org/10.1136/bmj-2023-078378

Cruz Rivera, S., Liu, X., Chan, A.-W., Denniston, A. K., & Calvert, M. J. (2020). Guidelines for clinical trial protocols for interventions involving artificial intelligence: The SPIRIT-AI extension. The Lancet Digital Health, 2(10), e549–e560. https://doi.org/10.1016/S2589-7500(20)30219-3

Ferrari, M. D., Goadsby, P. J., Burstein, R., Kurth, T., Ayata, C., Charles, A., Ashina, M., van den Maagdenberg, A. M. J. M., & Dodick, D. W. (2022). Migraine. Nature Reviews Disease Primers, 8, Article 2. https://doi.org/10.1038/s41572-021-00328-4

GBD 2016 Headache Collaborators. (2018). Global, regional, and national burden of migraine and tension-type headache, 1990–2016: A systematic analysis for the Global Burden of Disease Study 2016. The Lancet Neurology, 17(11), 954–976. https://doi.org/10.1016/S1474-4422(18)30322-3

GBD 2021 Headache Collaborators. (2025). Global, regional, and national burden of headache disorders, 1990–2021, with forecasts to 2050: A Global Burden of Disease study 2021. Cell Reports Medicine, 6(10), Article 102348. https://doi.org/10.1016/j.xcrm.2025.102348

GBD 2023 Headache Collaborators. (2025a). Global, regional, and national burden of headache disorders, 1990–2023: A systematic analysis for the Global Burden of Disease Study 2023. The Lancet Neurology, 24(12), 1005–1015. https://doi.org/10.1016/S1474-4422(25)00402-8

Headache Classification Committee of the International Headache Society (HCCIHS). (2018). The International Classification of Headache Disorders, 3rd edition. Cephalalgia, 38(1), 1–211. https://doi.org/10.1177/0333102417738202

Liu, X., Cruz Rivera, S., Moher, D., Calvert, M. J., & Denniston, A. K. (2020). Reporting guidelines for clinical trial reports for interventions involving artificial intelligence: The CONSORT-AI extension. The Lancet Digital Health, 2(10), e537–e548. https://doi.org/10.1016/S2589-7500(20)30218-1

Lu, Y., Li, Q.-Y., Gan, L., You, Y., Wang, C.-D., Guo, Z.-W., Shi, J., & Liu, X.-Y. (2025). The global and regional burden and trends of migraine from 1990 to 2021: Global Burden of Disease Study 2021. Frontiers in Neurology, 16, Article 1686288. https://doi.org/10.3389/fneur.2025.1686288

Moons, K. G. M., Damen, J. A. A., Kaul, T., Hooft, L., Andaur Navarro, C., Dhiman, P., et al. (2025). PROBAST+AI: An updated quality, risk of bias, and applicability assessment tool for prediction models using regression or artificial intelligence methods. BMJ, 388, Article e082505. https://doi.org/10.1136/bmj-2024-082505

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., et al. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. BMJ, 372(71). https://doi.org/10.1136/bmj.n71

Stubberud, A., Ingvaldsen, S. H., Brenner, E., Winnberg, I., Olsen, A., Gravdahl, G. B., Matharu, M. S., Nachev, P., & Tronvik, E. (2023). Forecasting migraine with machine learning based on mobile phone diary and wearable data. Cephalalgia, 43(5). https://doi.org/10.1177/03331024231169244

Wolff, R. F., Moons, K. G. M., Riley, R. D., Whiting, P. F., Westwood, M., Collins, G. S., et al. (2019). PROBAST: A tool to assess the risk of bias and applicability of prediction model studies. Annals of Internal Medicine, 170(1), 51–58. https://doi.org/10.7326/M18-1376

World Health Organization. (2021). Ethics and governance of artificial intelligence for health: WHO guidance. https://www.who.int/publications/i/item/9789240029200

Van Calster, B., McLernon, D. J., van Smeden, M., Wynants, L., & Steyerberg, E. W. (2019). Calibration: The Achilles heel of predictive analytics. BMC Medicine, 17, Article 230. https://doi.org/10.1186/s12916-019-1466-7

CHAPTER 5

ZERO TRUST CLOUD SECURITY AND ARTIFICIAL INTELLIGENCE DRIVEN IDENTITY PROTECTION

ÖMER SEVİNÇ[1]

Abstract Cloud computing, hybrid work, software as a service, and machine identities have weakened the value of a fixed network perimeter. Attackers increasingly target credentials, session tokens, application programming interfaces, and cloud control planes rather than attempting to penetrate a traditional office network. This chapter examines zero trust as an architectural approach for protecting cloud-native and hybrid systems and analyzes how artificial intelligence can support identity threat detection, access-risk assessment, policy enforcement, and incident response. It distinguishes zero trust from a product category and explains its dependence on accurate asset inventories, strong authentication, workload identity, least privilege, microsegmentation, telemetry, and governance. Particular attention is given to token theft, non-human identities, compromised cloud administration, model limitations, and privacy. A phased implementation model is proposed for

1 Dr., Ondokuz Mayıs University, Vezirköprü Vocational School, Department of Computer Programming.

organizations seeking measurable security improvement without disrupting legitimate users or critical services.

Keywords zero trust, cloud security, identity threat detection, artificial intelligence, cloud native security, access control

Introduction

Enterprise computing no longer resides within a single controlled network. Employees, contractors, applications, devices, and automated workloads access resources distributed across private data centers, several cloud providers, and software-as-a-service platforms. A user may authenticate through one identity provider, receive a token, and reach data through multiple application programming interfaces without traversing a traditional security gateway. Network location therefore provides weak evidence of trust. A valid session may belong to a legitimate employee, an attacker using stolen credentials, or malware operating through a compromised browser.

Zero trust addresses this environment by removing implicit trust based on location, ownership, or a previous authentication event. Access decisions are made for a specific subject, resource, action, and context, and are reassessed as risk changes. NIST defines zero trust as a shift from static network perimeters toward protection of users, assets, and resources (Rose et al., 2020). Artificial intelligence can strengthen this approach by identifying unusual relationships and behavior at cloud scale, but it cannot replace identity governance, secure configuration, or accountable policy.

Cloud Identity as the Primary Attack Surface

Cloud incidents frequently begin with identity rather than exploitation of a network service. Password phishing, consent phishing, adversary-in-the-middle techniques, help-desk manipulation, leaked access keys, and stolen browser tokens can

give an attacker the same interfaces available to an authorized user. Multifactor authentication reduces risk, but some methods remain vulnerable to push fatigue, social engineering, or token replay. Phishing-resistant authentication based on cryptographic credentials provides stronger protection for administrators and other high-impact roles.

Session tokens and assertions require specific controls because they may remain useful after authentication has completed. Short lifetimes, secure storage, audience restriction, sender constraints where supported, continuous access evaluation, and rapid revocation reduce exposure. Identity providers, federation services, and cloud control planes should receive the highest level of monitoring because their compromise can create valid credentials, alter logging, or grant persistent access across many systems. Recent CISA and NIST guidance treats the protection of cloud tokens and assertions as a distinct implementation problem rather than an ordinary password issue (CISA & NIST, 2026).

Zero Trust Architecture for Hybrid and Cloud Native Systems

A zero trust architecture separates policy decisions from enforcement. A policy engine evaluates identity, device posture, resource sensitivity, requested action, threat information, and environmental context. Enforcement points then allow, restrict, or deny the transaction. This model can operate at an identity proxy, application gateway, service mesh, database, endpoint, or software-defined network. Effective placement is close to the protected resource so that internal traffic and service-to-service calls do not bypass the policy.

Cloud-native systems introduce short-lived containers, serverless functions, managed services, and microservices that communicate through application programming interfaces.

Workloads need identities that are independent of internet addresses and do not rely on embedded secrets. Mutual authentication, workload certificates, signed software artifacts, and narrowly scoped authorization can reduce lateral movement. NIST SP 800-207A describes how application-level policies and service infrastructure can apply zero trust principles across hybrid and multi-cloud environments (Chandramouli & Butcher, 2023).

Artificial Intelligence for Identity Threat Detection

AI-supported identity analytics can establish patterns for sign-in time, device, network, privilege use, resource access, transaction volume, and relationships among users and workloads. A model may identify impossible travel, unusual administrative actions, a service account acting interactively, or a user downloading data that is inconsistent with the person's role. Graph analysis is particularly useful when an attacker performs several individually plausible actions that form a suspicious path toward a sensitive resource.

Risk scoring should support proportionate controls. A low-confidence anomaly might increase logging; moderate risk may trigger step-up authentication; strong evidence of active compromise may suspend a token or isolate a workload. Organizations should preserve the reasons, evidence, and policy used for consequential actions. A model that produces an unexplained score should not have unrestricted authority to disable an executive, clinician, production service, or emergency operator. Human review and tested emergency criteria remain necessary.

Non Human Identities and Machine Access

Modern environments contain more machine identities than human users. Service accounts, application registrations, automation agents, build pipelines, managed identities, certificates, and API keys may hold broad privileges and operate continuously. These

identities are often poorly inventoried because development teams create them through infrastructure code or cloud consoles. Long-lived secrets can be copied into repositories, configuration files, tickets, or logs and remain usable after the original project changes.

A secure program assigns an owner and purpose to every non-human identity, eliminates unused credentials, prefers short-lived tokens, and restricts each identity to defined resources and actions. Workload identity federation can reduce static secrets by exchanging trusted platform evidence for temporary credentials. AI can assist with inventory, privilege analysis, and detection of abnormal use, but automated recommendations must recognize scheduled jobs, failover activity, and deployment patterns to avoid interrupting production.

Data Protection Microsegmentation and Policy Enforcement

Zero trust protects data and services through granular policy rather than assuming that internal communication is safe. Data classification allows policy to reflect sensitivity, legal duties, and business impact. Encryption protects content in transit and at rest, while authorization determines who or what may perform a particular operation. Policy should consider the requested action because reading a public record, exporting a database, changing a security rule, and deleting a backup create different risks.

Microsegmentation limits the paths available after compromise. In cloud-native systems, segmentation may be enforced through security groups, service-mesh policy, application gateways, database permissions, and workload identity. Excessive segmentation can create operational complexity, so organizations should begin with high-value resources and common attack paths. Policy-as-code can improve consistency and review, but policy

changes require testing, version control, approval, and rollback because an incorrect rule can cause a widespread outage.

Implementation and Maturity Model

Implementation should begin with visibility. The organization needs an inventory of users, privileged roles, workloads, devices, applications, data, access paths, identity providers, and enforcement points. The next stage strengthens authentication, removes standing privilege, and protects administrative interfaces. Subsequent stages introduce context-aware decisions, workload identity, microsegmentation, automated response, and continuous policy evaluation. CISA's Zero Trust Maturity Model organizes progress across identity, devices, networks, applications and workloads, and data, supported by visibility, automation, and governance (CISA, 2023).

A phased roadmap should define measurable outcomes rather than claiming that the organization has purchased zero trust. Useful measures include phishing-resistant authentication coverage, privileged-role activation time, dormant identity removal, number and age of static secrets, token revocation time, policy coverage for sensitive applications, and the proportion of service traffic using authenticated workload identities. Incident exercises should test stolen tokens, compromised administrators, malicious applications, and cloud-provider outages.

Domain	Foundational control	Advanced capability
Identity	Phishing-resistant authentication and role governance	Continuous risk evaluation and just-in-time privilege
Workloads	Unique identities and secret reduction	Short-lived credentials and service-level policy
Networks	Segmentation around critical resources	Dynamic application-aware microsegmentation
Data	Classification and access logging	Transaction-level authorization and loss detection

Privacy Governance and Operational Limitations

Continuous identity analytics can collect detailed information about work patterns, location, devices, communication, and resource use. Monitoring should have a defined security purpose, minimum necessary data, access restrictions, retention limits, and independent review. Models should be evaluated for false positives affecting remote workers, travelers, shared devices, accessibility tools, and unusual job roles. Security monitoring must not silently become a general employee-performance system.

Zero trust also has limits. It does not correct insecure software, prevent every supply-chain compromise, or guarantee that an authorized user will act safely. Policy engines and identity providers become critical dependencies that require resilience, backups, and protected administrative paths. Organizations need break-glass procedures that are strongly controlled and reviewed. Success depends on accurate information, disciplined governance, and usable controls that legitimate users can follow.

Conclusion

Zero trust is a practical response to cloud systems in which users, workloads, and data move across organizational and provider boundaries. Its central contribution is the replacement of broad, persistent trust with explicit and contextual authorization. AI improves visibility and can reveal identity abuse that fixed rules miss, but automated decisions must remain explainable, proportionate, and reversible. Organizations should focus first on high-impact identities and resources, then expand enforcement as inventories, telemetry, and operational maturity improve.

References

Chandramouli, R., & Butcher, Z. (2023). A zero trust architecture model for access control in cloud-native applications in multi-cloud environments (NIST SP 800-207A). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207A

Cybersecurity and Infrastructure Security Agency. (2023). Zero Trust Maturity Model Version 2.0. https://www.cisa.gov/resources-tools/resources/zero-trust-maturity-model

Cybersecurity and Infrastructure Security Agency. (2024). Implementing secure practices for cloud services (Binding Operational Directive 25-01). https://www.cisa.gov/news-events/news/cisa-directs-federal-agencies-secure-cloud-environments

Cybersecurity and Infrastructure Security Agency, & National Institute of Standards and Technology. (2026). Protecting tokens and assertions from forgery, theft, and misuse (NIST IR 8587). https://www.cisa.gov/news-events/news/cisa-and-nist-release-guidelines-protect-federal-cloud-identity-systems-token-theft-forgery-and

National Institute of Standards and Technology. (2025). Implementing a zero trust architecture (NIST SP 1800-35). https://doi.org/10.6028/NIST.SP.1800-35

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero trust architecture (NIST SP 800-207). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207


BİDGE