



# SHAPING THE FUTURE WITH EMBEDDED AI

Artificial Intelligence is revolutionizing industries by enabling devices to process data and make decisions locally, reducing reliance on cloud computing. Renesas is at the forefront of this transformation, providing comprehensive solutions that facilitate AI deployment from the cloud to the edge and endpoints with AI-optimized semiconductor solutions designed for low-power, high-performance applications.

These articles highlight our strategic approach to simplifying AI deployment across various sectors – including industrial automation, smart homes and energy management. You'll read about advancing embedded AI and machine learning along with the efficiency and value of edge processing with Reality AI Tools™, which empower engineers to develop and implement AI models optimized for signal processing applications. Additionally, the book addresses best practices for data collection in edge AI applications, emphasizing the importance of comprehensive planning to ensure the accuracy and reliability of AI models.

We hope this will be a valuable resource for engineers, developers, and business leaders seeking to navigate the complexities of AI integration at the edge, offering practical guidance on leveraging advanced technologies to create intelligent, efficient, and responsive systems.

DIGITAL EBOOK  
AN EE WORLD RESOURCE

## WHAT'S INSIDE

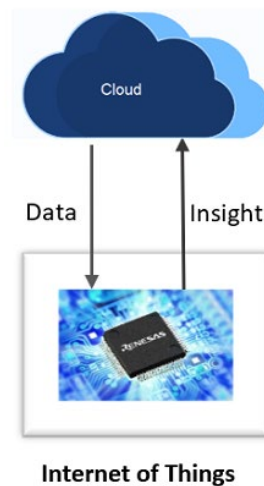
- 01 Executive Summary
- 02 Endpoint Intelligence
- 06 Simplifying AI Deployment from the Cloud to Edge and Endpoint
- 20 Ultimate Guide to Machine Learning for Embedded Systems
- 30 AI-as-a-Service for Signal Processing
- 39 Data Collection for Edge AI / Tiny ML with Sensors

and more >>

# Making the Endpoint Intelligent

The Internet of Things (IoT) has transformed the fabric of the world into a smarter and more responsive one by merging the digital and physical universes into one. Over the past few years, the IoT has exhibited exponential growth across a wide range of applications. According to a McKinsey study, the IoT will have an economic impact of \$4 - \$11 trillion by 2025. The edge continues to become more intelligent, and vendors are racing to support more connected and smart endpoint devices.

Backed by secure cloud infrastructure, smart connected devices offer many advantages which include the cost of ownership, resource efficiency, flexibility, and convenience. However, the process of transferring data back and forth from a device to the cloud results in additional latency and privacy risks during the data transfer. This is generally not an issue for non-real-time, low-latency applications, but for businesses that rely on real-time analytics with a need to quickly respond to events as they happen, this could end up in a major performance bottleneck.



**AIoT – Endpoint AI**

Traditional IoT vs Artificial Intelligence of Things

Imagine an industrial plant where the use of real-time data analytics and intelligent machine-to-sensor communication can significantly optimize the overall operations, logistics and supply chain. Data generated from such industrial sensors and control devices would be particularly beneficial for factory operators as it could enable them to overcome any challenges by pre-empting anomaly detection, prevent costly production errors and above all make the workplace safe.

This presents a real need to perform localized machine learning processing and analytics that would help reduce latency for critical applications, prevent data breaches, and effectively manage the data being generated by IoT devices. The only way to accomplish this is to bring the computation of data closer to where it is collected, namely the endpoint, rather than sending that data all the way back to a centralized cloud or datacentre for processing.



# Making the Endpoint Intelligent CONTINUED

Combining high-performance IoT devices with ML capabilities has unlocked new use cases and applications that resulted in the phenomenon of Artificial Intelligence of Things. The possibilities of AIoT — AI at the edge — are endless. For instance, visualize hearing aids that utilize algorithms to filter background noise from conversations. Likewise envision smart-home devices that rely on facial and vocal recognition to switch to a user's personalized settings. These personalized insights, decisions and predictions are a possibility because of a concept called Endpoint Intelligence or Endpoint AI.

Endpoint AI is a new frontier in the space of artificial intelligence which brings the processing power of AI to the edge. It is a revolutionary way of managing information, accumulating relevant data, and making decisions locally on a device. Endpoint AI employs intelligent functionality at the edge of the network. In other words, it transforms the IoT devices that are used to compute data into smarter tools with AI features. This equips them with real-time decision-making capabilities and functionalities. The goal is to bring machine learning based intelligent decision-making physically closer to the source of the data.

As illustrated below, pre-trained AI/ML models can now effectively be deployed at the endpoint enabling higher system efficiency vs. traditional cloud-connected IoT Systems.

Endpoint AI basically uses machine learning algorithms that run on local edge devices to make decisions without having to send information to cloud servers (or at least reduce how much information is sent).

With the vast amount of real-time data collected from IoT devices, intelligent machine learning algorithms are the most efficient way to get valuable insights from the data. However, these machine learning algorithms can be complex as they require higher compute power and a larger memory. Furthermore, the time frame required to identify patterns and make accurate decisions in huge datasets can be quite lengthy.

In the past, the ability to adopt efficient machine learning algorithms on constrained devices like a microcontroller was simply unimaginable, but this is now possible with advancements in the TinyML space. TinyML has been a game-changer for many embedded applications as it allows users to run ML algorithms directly

## VUI DEVELOPMENT KITS

Leverage a purpose-built hardware platform for Voice User Interface solutions based on the RA family of 32-bit MCUs. Quickly control your system with a simple voice command interface without extensive coding experience or in-house expertise. The reference kits enable local voice recognition without a network connection to allow designers to quickly start building an enhanced VUI in minutes that recognizes custom voice commands to trigger corresponding operations.

Select from a variety of kits based on your solution's performance requirements.



EXPLORE  
THE KITS

# Making the Endpoint Intelligent CONTINUED

on microcontrollers. This enables more efficient energy management, data protection, faster response times, and footprint-optimized AI/ML endpoint-capable algorithms.

Additionally, the new generation of multi-purpose Microcontrollers now offers sufficient compute power, intelligent power-saving peripherals, and most importantly, robust security engines that enable the mandatory privacy of data within the device. This allows for new applications in the AIoT space as well as new types of data processing, latency, and security solutions that can operate offline as well as online.

Let's look briefly into the advantages of Endpoint AI.

## Privacy and Security - A Prerequisite

At the heart of effective endpoint AI is data collection and analysis — often in environments where privacy and security are paramount concerns due to some regulations or business needs.

Endpoint AI is fundamentally more secure. Data isn't just sent to the cloud - it is being processed right in the endpoint itself. According to the F-Secure report, IoT

endpoints were the "top target of internet attacks in 2019" and another study suggests that IoT devices experience an average of 5,200 attacks per month. These attacks mostly arise due to the transfer and flow of data from IoT devices into the cloud. Being able to analyze data without moving it outside its original environment provides an added layer of protection against hackers.

## Efficient Data Transfer

Centralized processing of data entails that data be relocated from its source to a centralized location where it can be analyzed. The time spent transferring the data can be significant and poses a risk for inaccurate results especially if the underlying circumstances have changed considerably between collection and analysis.

Endpoint AI transmits data from devices, sensors, and machines to an edge data center or cloud, significantly reducing the time for decisive actions and increasing the efficiency of the transfer, processing, and results.

Some processing can be done on distributed sources (edge devices) effectively reducing network traffic, improving accuracy, and reducing costs.

## Minimal wait time

A latency of 1,500 milliseconds (1.5 seconds) is the limit for an e-commerce site to achieve a similar user experience as a brick-and-mortar store. Users will not tolerate such a delay and they will leave, resulting in lower revenues. With Endpoint AI, latency is reduced by transforming data closer to where it is collected. This enables software and hardware solutions to be deployed seamlessly, with zero downtime.

## Reliability When it Matters

Another key advantage of endpoint AI is reliability as fundamentally it is less dependent on the cloud, improving overall system performance and reducing the risk of data loss risk.

Endpoint AI ensures that your information is always available, and never leaves the edge, allowing independent and real-time decision-making. The decisions must be accurate and done in real time. The only way to achieve this is to implement AI at the edge.

# Making the Endpoint Intelligent CONTINUED

## AI DEVELOPMENT KITS

The AIK series are complete reference solutions designed for fast, simple integration of AI / ML technology for edge and endpoint systems address real-time analytics, vision and audio applications. Each kit comes with highly-tuned AI / ML models combined in a multi-modal solution with optimized CPU performance requiring very low memory resources. Each kit in the series provides a scalable approach enabling a flexible selection of processing performance based on your requirements.



EXPLORE  
THE KITS

### All in One Device

Endpoint AI offers the ability to integrate multimodal AI/ML architectures that can help enhance system performance, functionality and above all safety. For example, a voice + vision functionality combination is particularly well suited for hands-free AI-based vision systems. Voice recognition activates objects and facial recognition for critical vision-based tasks for applications like smart surveillance or hands-free video conferencing systems. Vision AI recognition may also be used to monitor operator behavior, control critical operations, or manage error or risk detection across a number of commercial or industrial applications.

### A Sustainable and viable approach

Integrating AI and ML capabilities with high-performance on-device compute has opened up a new world of possibilities for developing highly sustainable solutions. This integration has resulted in portable, smarter, energy-efficient, and more economical devices. AI can be harnessed to help manage environmental impacts across a variety of applications e.g., AI-infused clean distributed energy grids,

precision agriculture, sustainable supply chains, environmental monitoring as well as enhanced weather and disaster prediction and response.

Renesas is actively involved in providing ready-made AI/ML solutions and approaches as references within various applications and systems. Together with its partners, Renesas offers a comprehensive and highly optimized AI/ML end-point capable solution both from the hardware as well as from the software side. Fulfilling hereby all the attributes that need to be considered as a necessity from the very beginning.

The impact of AI isn't just in the cloud; it will be everywhere and in everything. Localized on-device intelligence, reduced latency, data integrity, faster action, scalability, and more are what Endpoint AI is all about, making the opportunities in this new AI frontier endless.

So now is the time for developers, product managers, and business stakeholders to take advantage of this huge opportunity by building better AIoT systems that would solve real world problems and generate new revenue streams.



# Simplifying AI Deployment from the Cloud to Edge and Endpoint

## Abstract

Artificial Intelligence (AI) is transforming every aspect of life. It is enhancing quality in industrial applications, enabling smart home systems, monitoring our safety as we work and play. Advances in technology have allowed us to run complex machine learning algorithms to tackle unique problems allowing those to be implemented also on embedded devices used in our daily life in home and industry. To enable scalable intelligence at all levels of the network, a decentralized intelligence architecture is needed. This means running cloud-independent inference engines on power-efficient or tiny computers within the edge and endpoints.

In this article, we will address the trends driving a decentralized intelligence model, the key application categories benefitting from AI, and how Renesas' approach simplifies design and deployment for AI / ML developers to help surmount commonly encountered challenges.

## Overview

Today there isn't any doubt that AI is changing the aspects of our everyday life both on and offline mostly in the background. The AIoT or Artificial Intelligence of Things is recognized as a transformational megatrend driven by four underlying market dynamics which include technology convergence, decentralized intelligence, a new design mindset, and an explosion of data.

- Technology convergence – IoT, AI, 5G maturing at roughly the same time
- Decentralized intelligence – Tremendous benefits of a distributed intelligence model
- New design mindset – AI disrupting system design approaches
- Data explosion – Endpoint data creation expected to grow 85% from 2017-2025\*



Technology  
Convergence



Decentralized  
Intelligence



New Design  
Market



Data Explosion

\*Source: IDC Research

# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

## Why Decentralize Intelligence?

Up to this point, the Internet of Things (IoT) has been built on a cloud-centric intelligence architecture. Based on overall experience and analysis over the past several years, we've learned the lesson we had to learn. Namely that widely distributed applications generate a vast amount of data. It has been clearly identified that this enormous amount of available data is unable to be handled, analyzed, and used. Furthermore, it has increased energy consumption and overall system costs. Even worse, the intelligence of AI was not intelligent enough to act/react when the cloud connection to the application was missing or interrupted. We don't need to mention the inability of any real-time response in this case. So, what we have realized is simply that not every application needs to be or should be served with cloud-centric intelligence. It doesn't mean that the data exchange to the cloud must be abandoned, instead it means we need to remove this unnecessary dependency and establish a more suitable decentralized approach that can enable the use of intelligence right where it's needed. Decentralizing intelligence, especially in the context of AI, refers to distributing

computing power and decision-making capabilities across a network of devices rather than relying solely on centralized systems and cloud architecture.

## Key benefits of a decentralized intelligence:



Supports Scalability

Allows the addition of resources to the network as demand grows and enables the system to handle increasing workloads without reaching critical bottlenecks or performance degradation.



Enables Real-time Response

Latency can be drastically reduced by distributing intelligence closer to where data is generated or consumed, enabling real-time processing and optimization of time-critical applications.



Enhances Security and Data Privacy

Enhances privacy and security by minimizing the need to transmit sensitive data to centralized servers. Instead, data can be processed locally on edge devices

tightly coupled to hardware root-of-trust, reducing the risk of unauthorized access or data breaches.



Reduces Costs and Increases Network

Agility Enables edge computing, where data is processed and analyzed closer to the source—such as IoT devices or sensors—allowing for faster insights and actions along with reduced bandwidth usage and costs associated with transferring large amounts of data to centralized servers.

## Merging AI and IoT

The Artificial Intelligence of Things represents a powerful synergy transforming industries and enabling innovative applications across various market segments and domains. IoT systems can be thought of as an interplay of one or more core technologies as shown in Figure 1 on page 8.

The vast amount of data generated from sensors, machines and connected devices can now be analyzed in real-time to extract valuable insights, detect patterns, and make predictions targeting process optimizations, equipment failure detection

# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

and proactive maintenance planning. All of which are exactly the goals of a decentralized approach which will allow us to embed AI directly into IoT or edge computing devices so systems can adapt to changing conditions, autonomously make decisions and respond to events in real-time.

Personalized experience, tailored recommendations, notifications, and services offered to users, create enhanced and more engaging and relevant experiences. Overall, AI and machine learning (ML) can be viewed as enablers bridging underlying technologies and applications, offering tremendous potential to improve efficiency, enhance user experience and drive innovation. AI / ML use can be classified broadly into three main categories – voice which deals with spoken voice and linguistics, vision which deals with camera, machine vision and robot vision and, finally, real-time analytics mainly addressing time-series data that is captured from sensors or sensor-less using the available system parameters.

Renesas offers a comprehensive solution stack for AI / ML developers, as well as a broad product portfolio spanning sensing, connectivity, computing, and actuation to

cover all layers of the IoT. In addition, software tools, solutions, and a broad partner ecosystem are purpose-built to accelerate your AIoT designs.

From the multi-modal needs of applications and overall complexity, users often have the challenge of selecting the best fitting and cost-optimized device for their system.

Here Renesas offers a scalable lineup within the MCU/MPU product families, helping to select the right one for a dedicated AI / ML application while also fulfilling the requirements from a performance vs multimodality and complexity perspective as depicted in Figure 3.

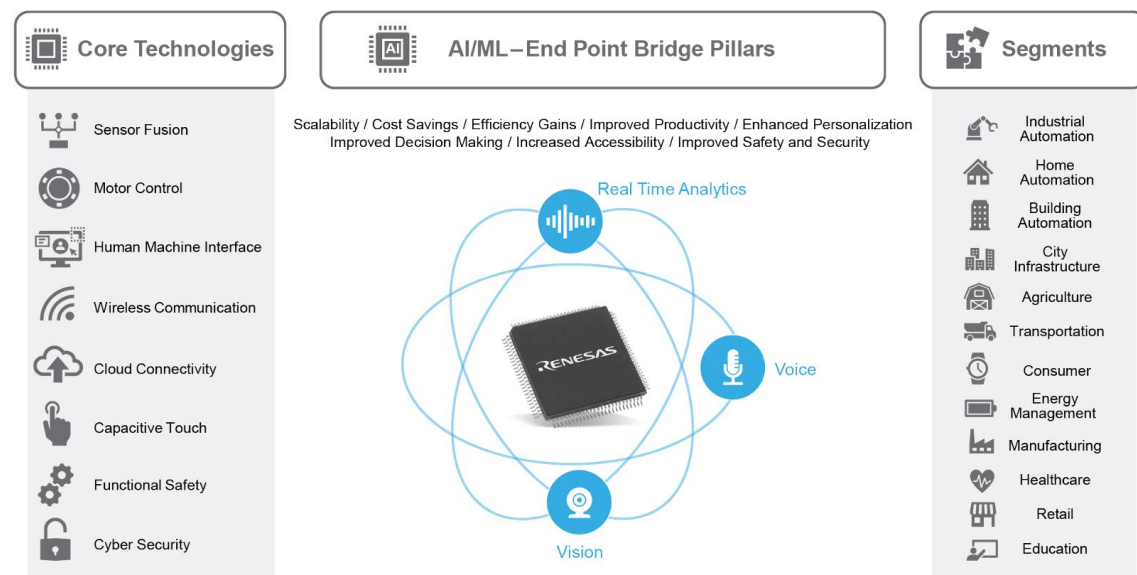


Figure 1. Core IoT technologies that can be enabled by AI and machine learning



# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

We've addressed the Renesas AI / ML view and segmentation into three main pillars, as well as a bit about the overall stack offering and the scalable device lineup. Now, let's look at the Renesas coverage for the three main AI / ML pillars and see the available resources and how they simplify AI deployment.

## AI / ML Pillars

### Voice

As speaking is a natural and intuitive way for humans to communicate, a Voice User Interface (VUI) allows us to interact with devices using our natural language. It reduces the learning curve and makes the interaction more user-friendly. VUI allows us to control various smart devices, such as thermostats, lights, appliances, making the overall experience more seamless and convenient. Despite significant advancements, VUI development challenges related to accuracy, robustness, and privacy still need to be considered. Otherwise, the user-friendly aspect can easily disappear in frustration. Support of different languages and accents is a requirement for global systems development. And that's where the journey begins with the extensive preparation

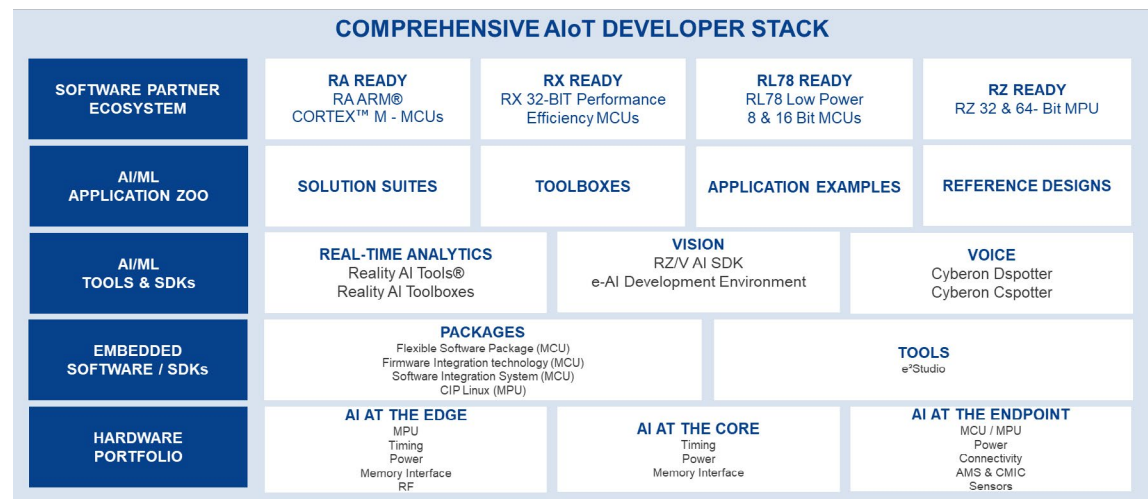


Figure 2. Renesas multi-layer developer stack  
Learn more: [www.renesas.com/key-technologies/artificial-intelligence](http://www.renesas.com/key-technologies/artificial-intelligence)

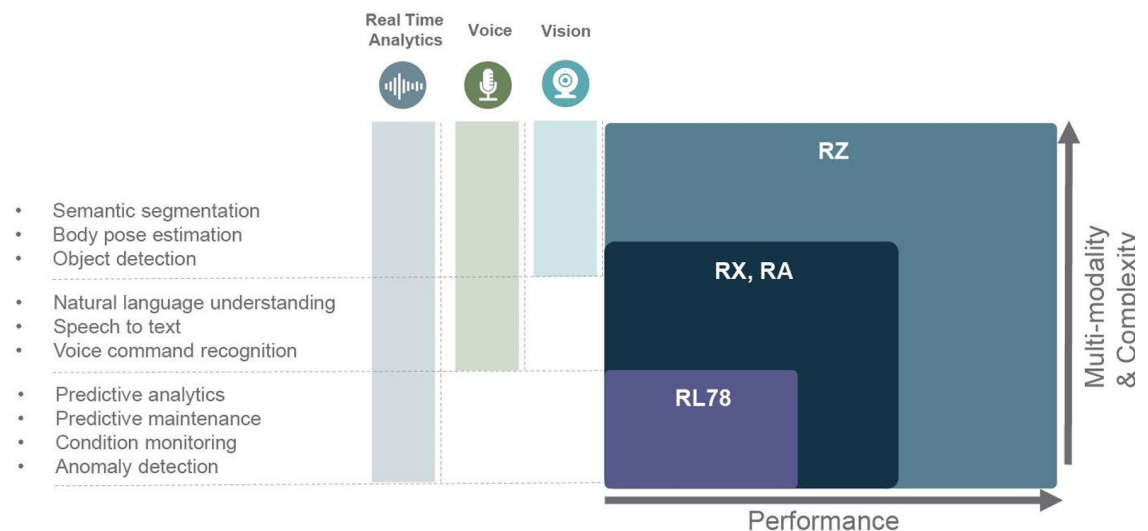


Figure 3. Renesas MCU / MPU portfolio addressing AI / ML specifications

# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

process and data collection for each language. Training, evaluation, testing and optimizing can become a prolonged process with preprogrammed headaches.

And we cannot forget to ensure critical user privacy and data security, given the sensitive nature of spoken interactions. Of course, optimal designs will strive for a cost-effective, decentralized endpoint on an embedded device.

The complete Renesas solution offers accelerated integration of your own VUI in your application. The development environment has already tackled the burdens and represents an easy-to-use embedded hardware platform for Voice User Interface solutions without extensive coding experience or in-house expertise. It is based on small, general purpose MCUs and even offers a RISC-V ASSP option as a standalone solution.

From a software perspective, the DSpotter Modeling Tool from Cyberon is provided as a GUI development environment. It enables simple, flawless development of an endpoint voice command recognition (VCR) or Natural Language Understanding (NLU) user interface. Pretrained models for 44+ languages eliminate the burden of extensive data collection and training the model accordingly. Customized commands

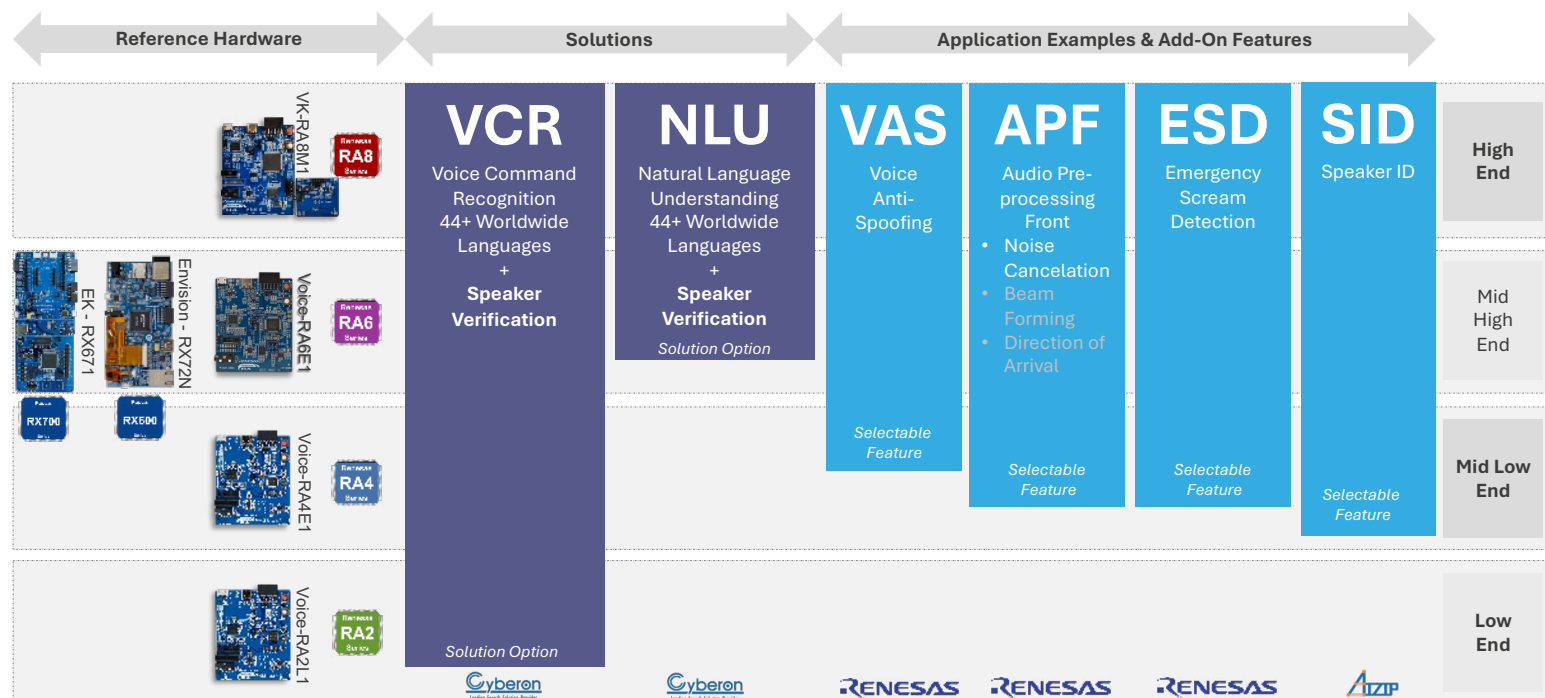


Figure 4. AI / ML Voice solutions



# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

can be easily generated by text input and the user can have offline performance tuning and testing available.

Additionally, Renesas provides enhanced features and application examples like voice anti-spoofing and audio front end, as well as other add ons from partners, as depicted in Figure 4. These feature sets can be added to the VCR or NLU to enhance the capabilities of the VUI to address safety, noise resistance and personalization along with ensuring user privacy and data security.

From the value proposition standpoint, here are some important take aways:

- Extensional application examples as add on features
- Voice anti-spoofing enables safety by distinguishing between real human and recorded voice
- Audio Front End enhances the audio / noise resistance capabilities
- Noise cancellation and beamforming included
- Speaker ID enables safety and personalization with user identification

Furthermore, material including documentation, webinars, videos, even hand-on session training is available on our web site to support you and your new VUI development. For more information, please visit the [Renesas Voice Solutions](#) page.

## Vision

Vision is enabled by camera-based systems and provides machines with the ability to interpret and understand visual information from images and videos. As a subset of AI, computer vision focuses on algorithms and techniques that enable machines to analyze, process, and extract meaningful insights from visual data.

Enabling a wide range of technological innovation, vision systems provide diverse approaches, such as image segmentation, object detection, facial recognition, edge detection, pattern detection, image classification, and feature matching. It is widely used in the manufacturing industry, healthcare, transportation, city & infrastructure, building automation and everyday life.

Key challenges are related to scalability and efficiency, data availability and ensuring quality and robustness, just

## REALITY AI TOOLS EXPLORER

The Reality AI Edge AI software development environment combines advanced signal processing, machine learning and anomaly detection on Renesas processors. The software is underpinned by a proprietary AI / ML algorithm that delivers accurate and fully explainable results supporting diverse applications and enabling these features to be added to products with minimal BOM impact. Use cases include equipment monitoring, predictive maintenance and sensing user behavior, as well as the surrounding environment. Try Reality AI Explorer for free to evaluate the evaluating this unique, powerful embedded real-time analytics development environment.



EXPLORE  
THE TOOLS

# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

to name a few. The crucial step here is addressing the question of how to tackle those from the very beginning in order to reduce burdens like the duration of evaluation and preparation of the substantial amounts of data needed for training. As if that were not enough, implementation on an embedded device must now be considered. Typical concerns for resource limitations, performance and power consumption still need to be addressed. These can all be managed, including initial cost reduction, shortening the transition from development to deployment and accelerating time to revenue. So where do we begin and how?

## Getting started

Let's begin with the selection of the hardware. For AI / ML systems, as with others, designers need hardware platforms offering scalability for factors such as processing power, memory, connectivity options, size, and power consumption. Evaluating off-the-shelf embedded systems, development boards or custom hardware solutions based on project requirements is more than given and important to address the target of our mission.

Renesas provides comprehensive offerings that includes a wide range of scalable MCU and MPU product families. An environment of reference hardware, application examples, pre-trained models, SDKs and tools such as the AI Navigator help guide developers to accelerate the evaluation and development of Visionbased AI applications.

Widely available devices feature one or more camera interfaces, dedicated graphical units (2D / 3D), highperformance image signal processor (ISP) supporting

4K/30fps and dynamic reconfigurable processor units known as DRP and DRP-AI (DRP-AI3) as AI Accelerators, in addition to advanced power management systems. Figure 3 shows the device series arranged by vision approach, complexity and multi-modality.

The RA8 series of MCUs is equipped with Arm® Cortex®-M85 (Armv8.1-M architecture) core enhanced with the Helium™ vector processor which significantly uplifts the

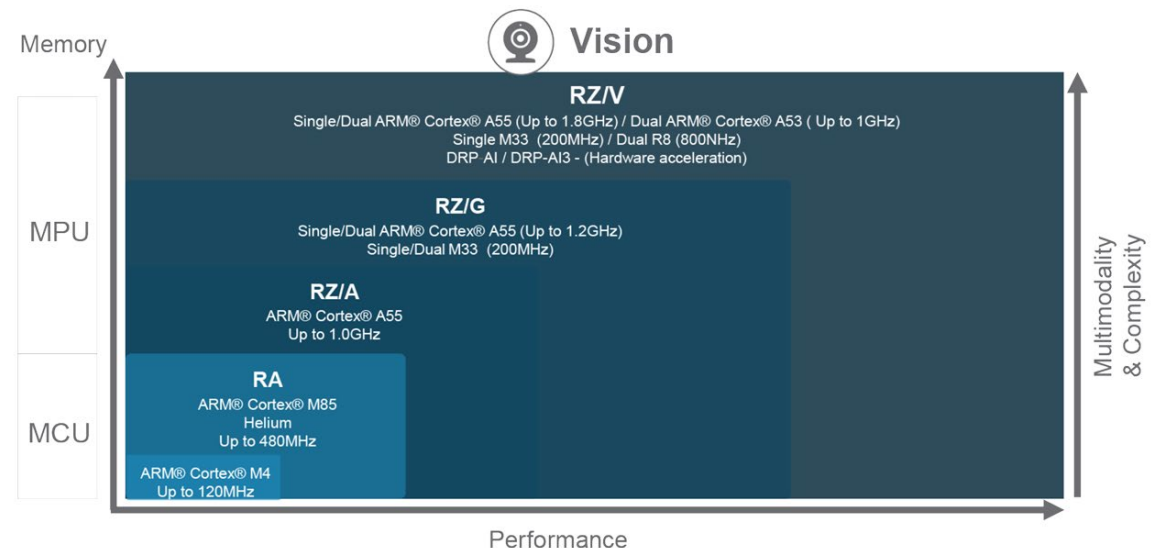



Figure 5. MCU / MPU options to address performance and complexity of vision systems



# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED



performance for ML and signal processing (DSP) applications. It delivers over 3000 CoreMark points at 480MHz operation. The RZ family of MPUs and the RZ/V2H device catapults users to the high-end of computing and computer vision. It comes with quad Cortex-A55 at 1.8GHz, dual Cortex-R8 real-time processor at 800MHz and single Cortex-M33 at 200MHz plus DRP-AI3 and DRP unit. With support for spars, pruning modeling techniques, it also features low-power consumption with best-in-class thermal performance at amazing 10TOPS/W performance/power consumption. For more information about key devices and specific AI accelerators and enhanced IP, visit the [RA8D1 page](#), [RZ/V series page](#) or [Renesas DRP-AI page](#).

On the microcontroller side, both the [RA8 AI development kit](#) and the [RA8D1 evaluation kit](#) comes with the Flexible Software Package (FSP) and the embedded development environment e<sup>2</sup>studio. Additional vision-based application examples are available from our Ecosystem partners for different use cases, such as driver condition recognition, camera-based QR code scanning, person access system with vision based anti-spoofing, just to name few. Visit the [RA AI partner page](#) for more

information and a complete list of all the available application examples, including people and object recognition, detection and access solutions.

On the microprocessor side, Renesas offers a development environment to support different levels of expertise in AI vision design with a combination of comprehensive tools and a wide range of application examples with pre-trained AI models.

The RZ/V2H evaluation kit supports standard software packages and enables implementation of low power consumption AI inferences and video streaming. Additionally, it supports optional functions like the image signal processor (ISP), 3D graphics engine (GE3D), and Trusted Secure IP. It comes with the RZ/V2H AI Software Development Kit (AI SDK) as an AI application development environment. The [Renesas RZ/V2HEVK](#) provides more information and resources including documentation, video tutorial, and others.

DRP-AI TVM (powered by the Edgecortex MERATM compiler framework) is a tool that generates runtime executables for AI from trained AI models for RZ/V series devices. It enables a specific separation of the executables to run on the CPU and the DRP-

AI3 for highly optimized inference execution.

DRP-AI Translator enables the translation of the AI model into an DRP-AI library which completely runs on the DRP-AI. The SDK package includes a complete image with a compiler and all the libraries needed for Linux cross-compilation. It enables execution of an application compiled with e2studio on the evaluation board.

AI Navigator is a plug in for e<sup>2</sup>studio and includes a transfer learning tool to retrain new classes of classifications within the available AI models. It allows integration and operation of various functions needed to develop AI and provides:

- Selection of AI applications from the Renesas AI application zoo and download of corresponding e2studio projects
- Customization of AI models for supported AI applications with their own datasets with the transfer learning feature
- Conversion of AI models to executable files. The RZ/V tool allows conversion to DRP-AI executable code using TVM

Developers can choose one of three



# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

options to get started with Renesas tools based on their level of expertise and where they are in the development journey:

**Immediate start:** accelerate the process with available, pre-trained models within the application zoo, select from among 50+ different models to generate your executable file using the AI SDK and an RZ/V evaluation kit.

**Bring your own Data (BYOD):** for a more

advanced approach where the available models need some customization based on available data set, using the transfer learning tool and retrain new classes of classifications. The DRP-AI TVM generates highly optimized runtime executable and the AI SDK generates the compilation for the RZ/V evaluation kit.

**Bring your own Model (BYOM):** if you

have one you would like use to evaluate the RZ/V capabilities, this path enables you to immediately generate the runtime executable through the DRP-AI TVM and AI SDK. The process of the integrated flow, as depicted in Figure 6, incorporates all the above tools and enables selection for the right path for your vision-based AI use case.

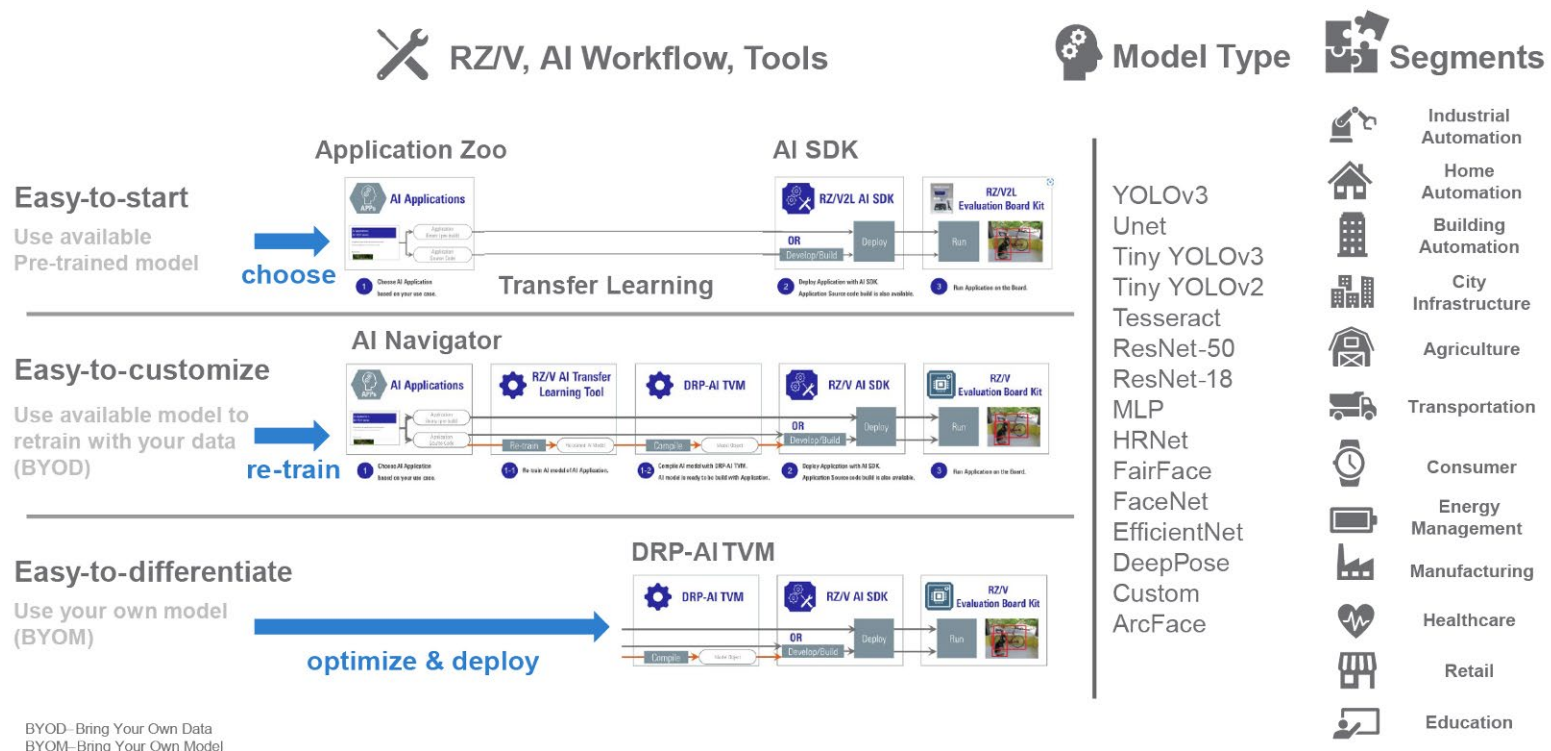


Figure 6. Selectable path for the RZ/V AI workflow, tools, and supported model types



# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

From lighting control depending on room occupancy to security area instructions or hand gesture recognition and touchless elevator control, the application zoo includes a variety of different use cases as applications with pre-trained models that are ready for evaluation, testing and further work, depending on your own application requirements. For more information and to start working with Renesas AI Vision solutions, please visit the [RZ/V AI GitHub site](#).

## Real-time Analytics (RTA)

This data driven engineering discipline enables a process of preparation, proceedings, and analyzing time series data as soon as available. Time-series data are simple measurements or tracked conditions/events, monitored, sub-sampled, and accumulated over time from different sources and sensors. It allows AI / ML models to learn from temporal dependencies, identify patterns, inconsistencies, and track changes over time. This is essential for applications where past observations influence future outcomes, such as anomaly detection and sequential decision-making. RTA empowers the opportunity for feature engineering,

which can be derived from temporal patterns and decision-making in applications where timely insights are critical and can adapt to evolving conditions and learn from new observations, ensuring robustness and accuracy in dynamic settings.

Renesas RTA solutions address non-visual time-series data and provide an extensive development environment, helping starters or longtime pros to accelerate embedded deployment and cut months, or even years

from an R&D cycle. Easy orientation within the full product lineup provides powerful tool environment, reference kits, solutions, toolboxes, and various application examples.

Reality AI Tools, the Renesas AI / ML flagship software, is a cloud-based edge and endpoint AI development environment. It combines advanced signal processing and machine learning which allow engineers to generate and build TinyML / edge AI models using powerful feature space ML path including SVM's and NN

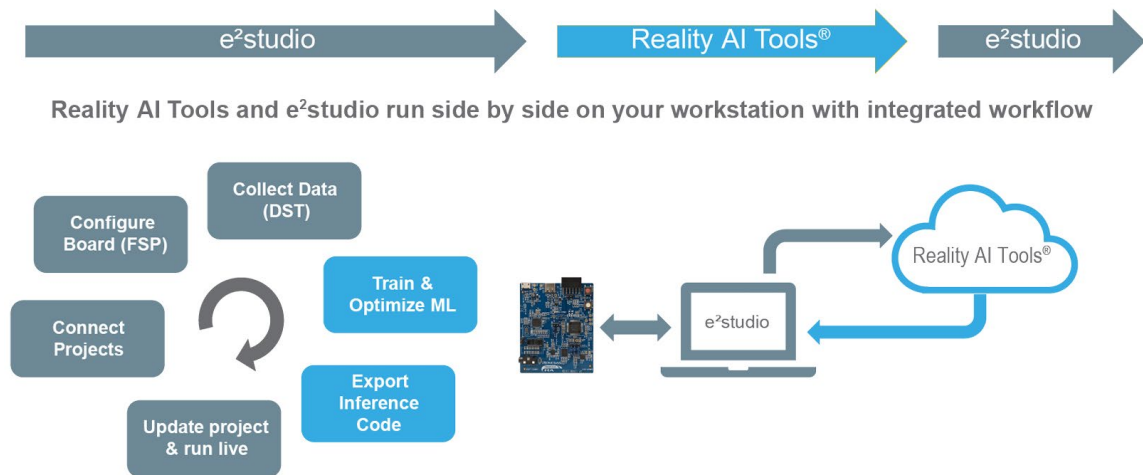


Figure 7. Integrated workflow between e²studio and Reality AI Tools

# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

techniques. The tool supports all Renesas processor cores and delivers automatic parameters tune capabilities of the machine learning algorithm based on discovery features that leads to accuracy improvements and greater computational efficiency. Users can explore sensor data and generate optimized models for small MCUs automatically and generate the embedded code targeting the best fitting device out of the full range of device portfolio.

The tool contains very helpful analytics to find the best sensor, combination of sensors or the best locations for placement and automatically generates the component specs. Includes fully explainable model functions in terms of time/frequency domain. For further information, visit the [Reality AI Tools page](#).

Furthermore, a seamless developer workflow has been established for rapid AI model prototyping with Renesas evaluation kits integrated with e<sup>2</sup>studio and Reality AI Tools. This integration enables users of e<sup>2</sup>studio to easy collect and upload data to Reality AI Tools, work in the environment to develop, train, and optimize the ML blocks and export the embedded code back into their project for live testing.

Alongside the Reality AI Tools suite, Renesas offers function-dedicated products accelerating the development to deployment phase within specific industrial application fields as depicted in Figure 7. These solution suites combined with dedicated Renesas support services assist developers from the very beginning to the end of the deployment.

- Reality AI Tools – Automatically explores sensor data and generates optimized models
- Reality AI Utilities – Plug-ins for e2 studio and popular IDEs to accelerate edge AI development
- Automotive Sound Recognition – Safety Warning System (SWS) combines hardware and software to give passengers a new level of protection
- RealityCheck™ HVAC – Complete framework to enable smart, self-diagnosing HVAC systems
- RealityCheck™ Motor – Advanced software toolbox enables predictive maintenance and anomaly detection
- RealityCheck™ AD – Anomaly detection for monitoring factory and process-industry assets

## Application examples and reference kits

Interested enthusiasts and developers who want to immediately evaluate and try out any of the RTA AI / ML use cases can leverage a suite of application examples and reference kits. From anomaly detection and presence detection to asset movement recognition, Renesas offers a variety of pre-built reference applications, enabling an easy and quick practical introduction to the subject of interest as depicted in Figure 8. A wide range of examples across different applications are available, in combination with the reference kits supported or cloud-only for testing and evaluation.

Reality AI Tools supports a variety of different evaluation and development kits with different devices for different applications.

The default kits package includes the full collateral material from schematics, BoM, documentation to software packages with dedicated BSP's, application notes and more. Technology-driven boards like the Cloud Kit (CK), Motor Control Kit (MCK) and others, are intended to support the technology they are build for and provide dedicated capabilities demonstration of the



# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

device itself within the technology requirements and serves as a proof of concept for further development as an application reference. Within the technology driven boards, Renesas has recently released scalable AI Kits (AIKs) to support AI / ML evaluation and development of vast of different use cases tackling complex tasks and challenges across various domains and also enables multimodal implementation. Figure 9 shows the AIK-RA6M3 and the AIK-RA4E1 kits with communication interfaces to enable a full flexibility in selection of sensors, adaption to different systems and enabling centralized or decentralized node.

Each kit comes with highly tuned AI / ML models combined in a multi-modal solution with optimized CPU performance requiring very low memory resources. In combination with Reality AI Tools, users have a powerful AI / ML environment to kick off the development of RTA-based applications and build a flexible system. Users are equipped with camera, Pmod acceleration sensor module and Pmod display. An out-of-the-box example for asset movement recognition is included. As an example of a multi-modal approach including vision, Renesas recommends the Person Access System (PAS) application example from our partner Aizip.

Cloud Only	Hardware	Reference Kit
Presence Detection	Voice Anti-Spoofing	<a href="#">Voice-RA6E1</a> / <a href="#">VK-RA8M1</a>
Urban Scene Classification	Asset Movement Recognition	<a href="#">AIK-RA6M3</a> / <a href="#">AIK-RA4E1</a> / <a href="#">AIK-RA8D1</a> / <a href="#">FPB-RL78</a> / <a href="#">CK-RA6M5</a>
Pump Condition Monitoring	Friction Detection	<a href="#">MCK-RA6T2</a> / <a href="#">MCK-RA8T1</a> / <a href="#">RSSK-RX66T</a>
Activity Detection Run vs Walking	Unbalanced Load Detection	<a href="#">MCK-RA6T2</a> / <a href="#">MCK-RA8T1</a> / <a href="#">RSSK-RX66T</a>
Vibration Analysis on Rotating Shaft	Floor Type Detection	<a href="#">MCK-RA6T2</a>
Fan Blower Condition Monitoring	Shaft Alignment and Unbalanced Load Detection	<a href="#">MCK-RA8T1</a>

Figure 8. Outlines the various available application examples within Real-Time Analytics AI / ML field

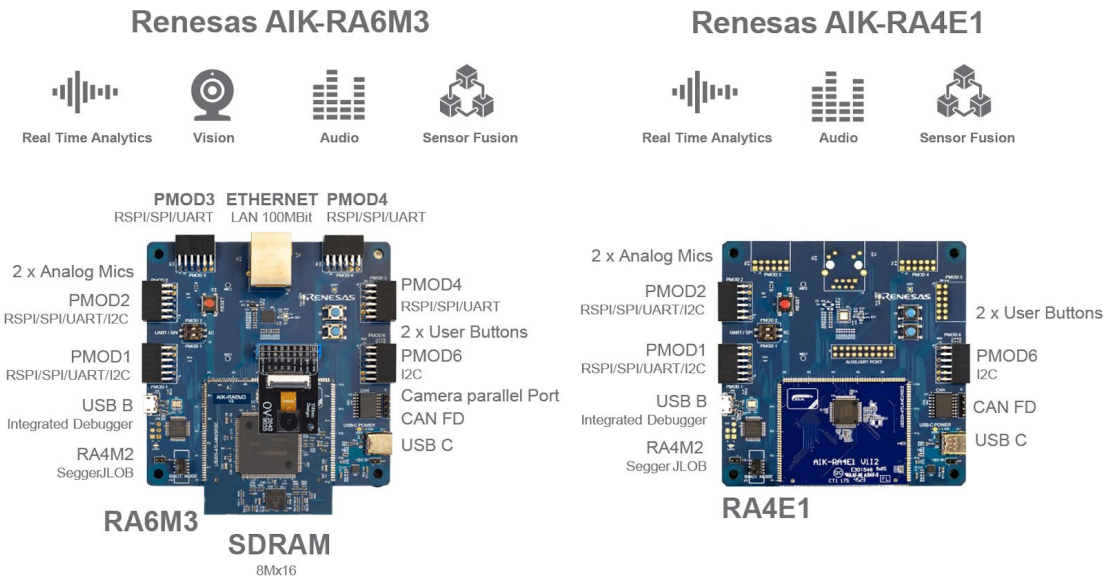


Figure 9. Outlines the characteristics of the AI Kits

# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

For more information on the AI Kits, visit the individual [AIK-RA6M3](#) and [AIK-RA4E1](#) pages.

## Broad Partner Ecosystem

The Renesas Ready Partner Network is an extensive, curated network of trusted technology partners delivering commercial-grade building blocks that work out-of-box with Renesas products. These solutions are designed to help accelerate the development of IoT capabilities around core technologies such as security, safety, artificial intelligence/ machine learning, connectivity, cloud, sensing & control and human machine interface. Whether adding voice capability to an IoT product or applying machine learning to a specific design problem, these pre-developed solutions will significantly help reduce design complexity and speed time to revenue. For more information, see the [AI ecosystem solutions](#) page.

## Example of AI Deployment in Arc Fault Application

Let's take a look at a real-world example of AI / ML technology in an arc fault circuit interrupter (AFCI) for PV/ESS systems enabled and deployed with Reality AI Tools at a solar panel manufacturer.

## Artificial Intelligence (AI)



Quick Overview: Arc fault detection is very important technology to maintain and guarantee the safety of power systems and is essential for manufacturing practical power systems for real-world applications.

Needs and Challenges: Arc-faults in PV systems may occur caused by various issues, such as faulty components, installation errors, mechanical damage or aging occurring after installation. Common examples causing arc-fault are damaged, pinched, or abraded conductors.

Furthermore, loose or separated connections or terminations can lead also to arc faults. Additionally, switching noise from a gridconnected inverter makes detecting arc fault conditions even more difficult. There are various algorithms already available and implemented to solve arc fault detection, but

the pain points remain, namely the maximum achievable accuracy of fault detection and the stability of detection under different conditions. Some system algorithms do have arc fault detection capabilities but typically at only 50% accuracy, which is clearly a disadvantage and poses a major safety threat.

Solution and Benefits: Renesas provided a comprehensive solution using AI / ML technology running on an RA6M4 MCU. With the data available in combination with additional data collected at the facility/lab, the Reality AI Tools deployment effectively generated the best fitting compact model with the lowest footprint, at  $\leq 8K$  RAM, 23K Flash and reach accuracy of  $\geq 99.8\%$ . The flexible and scalable approach for this application enables further overall CPU



# Simplifying AI Deployment from the Cloud to Edge and Endpoint

CONTINUED

performance and supports various system parameter-level scenarios up to maximum of 200A by maintaining the required effectiveness and stability.

Benefits:

- Performance optimized solution
- Compact model with low footprint
- Improved accuracy and stability
- Cost optimized and platform capable

Figure 10 shows an example of an AFCI implementation where the RA6M4 MCU combined with Reality AI Tools software enabled and integrated AI / ML approach to provide arc fault detection

## Renesas Approach

Renesas aims to accelerate your development and deployment of intelligent systems to enhance efficiency, safety, and user experience. Combining expertise in semiconductor design, embedded systems, and realtime control with AI processing capabilities, Renesas is actively investing in AI technology and solutions to address the growing demand for AI-driven applications across various industries, including industrial automation, automotive, consumer electronics, and IoT.

The Renesas approach integrates:

- Wide range of comprehensive hardware platforms with scalable MCU and MPU families
- Embedded tools and solution packages
- Comprehensive developer stack for AI / ML
- Tools and workflows suitable for multiple developer journeys – bring-your-own-model, transfer learning, bespoke consulting and more
- Rich library of easy-to-find solutions – application examples, toolboxes, solution suites, hardware reference kits
- Broad ecosystem of trusted partners offering commercial-grade building blocks

## Conclusion

Renesas is broadly committed to providing innovative semiconductor solutions for a diverse range of applications and industries. With continued investment in technology development, product innovation and customer support, we will address evolving market needs and enable the next generation of intelligent systems. Renesas edge AI solutions enable on-device AI processing for applications requiring low latency, real-time

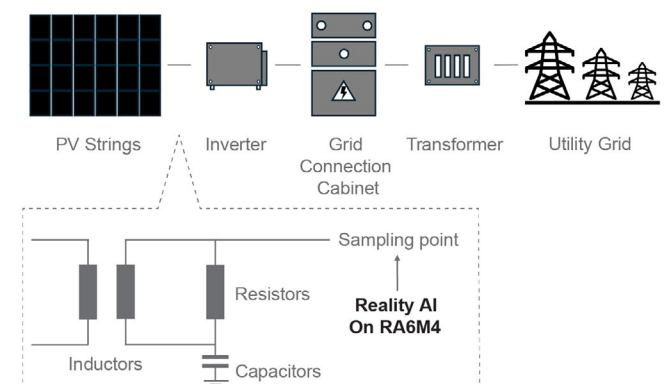


Figure 10. Outlines a typical AFCI PV/EVSS system topology example

responsiveness, and privacy protection. The decentralized approach supported by these solutions leverages our hardware platforms, AI accelerators and software frameworks to perform AI inference tasks locally on edge devices, reducing reliance on cloud-based services and enabling autonomous decisionmaking in distributed systems.

Visit [renesas.com/AI](https://renesas.com/AI) to put the Renesas approach to work on your next intelligent design.

# Ultimate Guide to Machine Learning for Embedded Systems

## What is Machine Learning for Embedded Systems?

Machine learning is a subfield of Artificial Intelligence which gives computers an ability to learn from data in an iterative manner using different techniques. Our aim here being to learn and predict from data. This is a big diversion from other fields which poses the limitation of programming instructions instead of learning from them. Machine Learning in Embedded Systems specifically target embedded systems to gather data, learn and predict for them. These systems typically consist of low memory, low Ram and minimal resources compared to our traditional computers.

So now you know a little more about what we mean by “machine learning for embedded systems”, but maybe you’re still unsure about where or how to start? That’s why we’ve created the ultimate guide to machine learning for embedded systems. Over the last few years, as sensor and MCU prices plummeted and shipped volumes have gone thru the roof, more and more companies have tried to take advantage by adding sensor-driven embedded AI to their products.

Automotive is leading the trend – the average non-autonomous vehicle now has 100 sensors, sending data to 3050 microcontrollers that run about 1m lines of code and generate 1TB of data per car per day. Luxury vehicles may have twice as many, and autonomous vehicles increase the sensor count even more dramatically. But it’s not just an automotive trend. Industrial equipment is becoming increasingly “smart” as makers of rotating, reciprocating and other types of equipment rush to add functionality for condition monitoring and predictive maintenance, and a slew of new consumer products from toothbrushes, to vacuum cleaners, to fitness monitors add instrumentation and “smarts”.

## Real-Time, at the Edge, and a Reasonable Price Point

What these applications have in common is the need to use real-time, streaming, complex sensor data – accelerometer, vibration, sound, electrical and biometric signals – to find signatures of specific events and conditions, or detect anomalies, and do it locally on the device: with code that runs in firmware on a microcontroller that fits the product’s price point.

When setting out to build a product with these kinds of sensor-driven smarts, there are three main challenges that need to be overcome.

Simultaneous challenges when using sensors with Embedded AI:

- Variation in target and background
- Real-time detection
- Constraints – size, weight, power consumption, price

## Variation

Real world data is noisy and full of variation – meaning that the things you’re looking for may look different in different circumstances. You will face variation in your targets (want to detect sit-ups in a wearable device? First thing you will hit is that people do them all slightly differently, with myriad variations). But you will also face variation in backgrounds (vibration sensors on industrial equipment will also pick up vibrations transmitted thru the structure from nearby equipment).

# Ultimate Guide to Machine Learning for Embedded Systems

CONTINUED

## WEBINAR: LIGHTWEIGHT AI FOR PREDICTIVE MAINTENANCE

Make your products smarter with sensors and TinyML. Watch this webinar and see how low-power, general purpose MCUs with TinyML can enable your products to:

- Recognize and localize sounds
- Recognize non-visual gestures (capacitive, radar, motion, etc.)
- Detect anomalies in smart equipment

Reality AI Tools® Edge AI software is a complete embedded machine learning development environment for incorporating real-time analytics in MCU-based systems. This webinar provides an introduction to Reality AI Tools and how they enable you to build smart embedded products with TinyML.



VIEW THE  
WEBINAR

Background variation can sometimes be as important as target variation, so you'll want to collect both examples and counter-examples in as many backgrounds as possible.

### Real-Time Detection in Firmware

The need to be able to accomplish detections locally that provide a user with a "real-time" experience, or to provoke a time-sensitive control response in a machine, adds complexity to the problem.

### Constraints – Physical, Power, and Economic

With infinite computing power, lots of problems would be a lot easier. But real-world products have to deliver within a combination of form factor, weight, power consumption and cost constraints.

### Traditional Engineering vs Machine Learning

To do all of this simultaneously, overcome variation to accomplish difficult detections in real-time, at the edge, within the necessary constraints is not at all easy. But with modern tools, including new options for machine learning on signals (like Renesas Reality AI) it is becoming easier.

Certainly, traditional engineering models constructed with tools like Matlab are a viable option for creating locally embeddable detection code. Matlab has a very powerful signal processing toolbox which, in the hands of an engineer who really knows what she is doing, can be used to create highly sophisticated, yet computationally compact, models for detection.

### Why Use Machine Learning?

#### Why is machine learning increasingly a tool of choice?

For starters, the more sophisticated machine learning tools (like Reality AI from Renesas) that are optimized for real-time analytics of signal data and embedded deployment can cut months, or even years, from an R&D cycle. They can get to answers quickly, generating embeddable code fast, allowing product developers to focus on their functionality rather than on the mathematics of detection.

But more importantly, they can often accomplish detections that elude traditional engineering models. They do this by making much more efficient and effective use of data to overcome variation.



# Ultimate Guide to Machine Learning for Embedded Systems

CONTINUED

Where traditional engineering approaches will typically be based on a physical model, using data to estimate parameters, machine learning approaches can learn independently of those models. They learn how to detect signatures directly from the raw data and use the mechanics of machine learning (mathematics) to separate targets from non-targets without falling back on physics.

## Different Approaches for Different Problems

It is also important to know that there are several different approaches to machine learning for this kind of complex data. The one getting most of the press is “Deep Learning”, a machine learning method that uses layers of convolutional and/or recurrent neural networks to learn how to predict accurately from large amounts of data. Deep Learning has been very successful in many use cases, but it also has drawbacks – in particular, that it requires very large data sets on which to train, and that for deployment it typically requires specialized hardware (\$\$\$). Other approaches, like the one we take at Renesas, may be more appropriate if your deployment faces cost, size or power constraints.

## Three things to keep in mind when building products with Embedded AI

If you’re thinking of using machine learning for embedded product development, there are three things you should understand:

1. Use rich data, not poor data. The best machine learning approaches work best with information-rich data. Make sure you are capturing what you need.
2. It’s all about the features. Once you have good data, the features you choose to employ as inputs to the machine learning model will be far more important than which algorithm you use.
3. Be prepared for compromises and trade-offs. The sample rate at which you collect data and the size of the decision window will drive much of the requirements for memory and clock-speed on the controller you select. But they will also affect detection accuracy. Be sure to experiment with the relationship between accuracy, sample rate, window size, and computational intensity. The best tools in the market will make it easy for you to do this.

## Machine Learning: Lab vs Real World

Not long ago, TechCrunch ran a story reporting on Carnegie Mellon research showing that an “Overclocked smartwatch sensor uses vibrations to sense gestures, objects and locations.” These folks at the CMU Human-Computer Interaction Institute had apparently modified a smartwatch OS to capture 4 kHz accelerometer waveforms (most wearable devices capture at rates up to 0.1 kHz), and discovered that with more data you could detect a lot more things. They could detect specific hand gestures, and could even tell a what kind of thing a person was touching or holding based on vibrations communicated thru the human body. (Is that an electric toothbrush, a stapler, or the steering wheel of a running automobile?) “DUH!”

To those of us working in the field, including those at Carnegie Mellon, this was no great revelation. “Duh! Of course, you can!” It was a nice-but-limited academic confirmation of what many people already know and are working on. TechCrunch, however, in typical breathless fashion, reported as if it were news. Apparently, the reporter was unaware of the many commercially available products

# Ultimate Guide to Machine Learning for Embedded Systems

CONTINUED

that perform gesture recognition (among them Myo from Thalmic Labs, using its proprietary hardware, or some 20 others offering smartwatch tools). It seems he was also completely unaware of commercially available toolkits for identifying very subtle vibrations and accelerometry to detect machines conditions in noisy, complex environments (like Reality AI solutions for industrial equipment monitoring), or to detect user activity and environment in wearables (Reality AI for consumer products). But my purpose is not to air sour grapes over lazy reporting. Rather, I'd like to use this case to illustrate some key issues about using machine learning to make products for the real world: Generalization vs Overtraining, and the difference between a laboratory trial (like that study) and a real-world deployment.

## Generalization and Overtraining

Generalization refers to the ability of a classifier or detector, built using machine learning, to correctly identify examples that were not included in the original training set. Overtraining refers to a classifier that has learned to identify with high accuracy the specific examples on which it was trained, but does poorly on similar

examples it hasn't seen before. An overtrained classifier has learned its training set "too well" – in effect memorizing the specifics of the training examples without the ability to spot similar examples again in the wild.

That's ok in the lab when you're trying to determine whether something is detectable at all, but an overtrained classifier will never be useful out in the real world. Typically, the best guard against overtraining is to use a training set that captures as much of the expected variation in target and environment as possible. If you want to detect when a type of machine is exhibiting a particular condition, for example, include in your training data many examples of that type of machine exhibiting that condition, and exhibiting it under a range of operating conditions, loads, etc. It also helps to be very skeptical of "perfect" results. Accuracy nearing 100% on small sample sets is a classic symptom of overtraining.

It's impossible to be sure without looking more closely at the underlying data, model, and validation results, but this CMU study shows classic signs of overtraining. Both the training and validation sets contain a single example of each target machine



Illustration from the CMU study using vibrations captured with an overclocked smartwatch to detect what object a person is holding.



# Ultimate Guide to Machine Learning for Embedded Systems

CONTINUED

## WEBINAR: ACCELERATE EMBEDDED VISION AI SYSTEM DEVELOPMENT

Explore how developing advanced embedded Vision AI systems at the edge or endpoints can be simpler than ever before. Watch this on-demand virtual hands-on session to learn how to accelerate your next design:

- Hear directly from engineers how to use the compact and cost-effective dual-core RZ/V2L MPU-based single board computer
- Learn how this SBC can help accelerate your system design with options to use it directly in your product

Learn how to deploy next-gen vision applications into your designs.



VIEW THE  
WEBINAR

collected under carefully controlled conditions. And to validate, they appear to use a group of 17 subjects holding the same single examples of each machine. In a nod to capturing variation, they have each subject stand in different rooms when holding the example machines, but it's a far cry from the full extent of realworld variability. Their result has most objects hitting 100% accuracy, with a couple of objects showing a little lower. Small sample sizes. Reuse of training objects for validation. Limited variation. Very high accuracy... Classic overtraining. Both the training and validation sets contain a single example of each target machine collected under carefully controlled conditions. And to validate, they appear to use a group of 17 subjects holding the same single examples of each machine. In a nod to capturing variation, they have each subject stand in different rooms when holding the example machines, but it's a far cry from the full extent of realworld variability. Their result has most objects hitting 100% accuracy, with a couple of objects showing a little lower. Small sample sizes. Reuse of training objects for validation. Limited variation. Very high accuracy... Classic overtraining.

## Detect Overtraining and Predict Generalization

It is possible to detect overtraining and estimate how well a machine learning classifier or detector will generalize. At Renesas, our go-to diagnostic is the K-fold validation, generated routinely by our tools. K-fold validation involves repeatedly; 1) holding out a randomly selected portion of the training data (say 10%), 2) training on the remainder (90%), 3) classifying the holdout data using the 90% trained model, and 4) recording the results.

Generally, hold-outs do not overlap, so, for example, 10 independent trials would be completed for a 10% holdout. Holdouts may be balanced across groups and validation may be averaged over multiple runs, but the key is that in each iteration the classifier is tested on data that was not part of its training. The accuracy will almost certainly be lower than what you compute by applying the model to its training data (a stat we refer to as "class separation", rather than accuracy), but it will be a much better predictor of how well the classifier will perform in the wild – at least to the degree that your training set resembles the real world. Counter-intuitively, classifiers with



# Ultimate Guide to Machine Learning for Embedded Systems

CONTINUED

weaker class separation often hold up better in K-fold. It is not uncommon that a near perfect accuracy on the training data drops precipitously in K-fold while a slightly weaker classifier maintains excellent generalization performance.

And isn't that what you're really after? Better performance in the real world on new observations? Getting high-class separation, but low K-fold? You have a model that has been overtrained, with poor ability to generalize. Back to the drawing board. Maybe select a less aggressive machine learning model, or revisit your feature selection. Reality AI solutions do this automatically. Be careful, though, because the converse is not true: A good K-fold does not guarantee a deployable classifier. The only way to know for sure what you've missed in the lab is to test in the wild. Not perfect? No problem: collect more training data capturing more examples of underrepresented variation. A good development tool (like ours) will make it easy to support rapid, iterative improvements of your classifiers.

## Lab Experiments vs Real World Products

Lab experiments like this CMU study don't need to care much about generalization – they are constructed to illustrate a very

specific point, prove a concept, and move on. Real-world products, on the other hand, must perform a useful function in a variety of unforeseen circumstances. For machine learning classifiers used in real-world products, the ability generalize is critical. But it's not the only thing. Deployment considerations matter too. Can it run in the cloud, or is it destined for a processor-, memory- and/or power-constrained environment? (To the CMU guys – good luck getting acceptable battery life out of an overclocked smartwatch!) How computationally intensive is the solution, and can it be run in the target environment with the memory and processing cycles available to it?

What response-time or latency is acceptable? These issues must be factored into a product design, and into the choice of machine-learning model supporting that product. Tools like Reality AI can help. R&D engineers use Reality AI Tools® to create machine learning-based signal classifiers and detectors for real-world products, including wearables and machines and can explore connections between sample rate, computational intensity, and accuracy. They can train new models and run k-fold diagnostics (among others) to guard against overtraining and predictability to

generalize. And when they're done, they can deploy to the cloud, or export code to be compiled for their specific embedded environment. R&D engineers creating real-world products don't have the luxury of controlled environments – overtraining leads to a failed product. Lab experiments don't face that reality. Neither do TechCrunch reporters.

## How To Succeed with Machine Learning

At Renesas we see a lot of machine learning projects that have failed to get results, or are on the edge of going off the rails. Often, our tools and structured approach can help, but sometimes not.

### Here are 3 ways to succeed with Machine Learning:

Number 1: Get Ground Truth Machine learning isn't a magic wand, and it doesn't work by telepathy. Algorithms need data and examples of what it is trying to detect, as well as examples of what it is not trying to detect, so that it can tell the difference. This is particularly true of "supervised learning" algorithms, where the algorithm must train on sufficient numbers of examples in order to generate results. But it also applies to "unsupervised learning"

# Ultimate Guide to Machine Learning for Embedded Systems

CONTINUED

algorithms, which attempt to discover hidden relationships in data without being told ahead of time, as well. If relationships of interest don't exist in the data, no algorithm will find them.

Number 2: Curate the Data Data should be clean and well curated. Meaning that to get the best results, it is important to have faith in the quality of the data. Misclassifications in training data can be particularly damaging in supervised learning situations -- some algorithms (like ours) can compensate for occasional misclassifications in training data, but pervasive problems can be hard to overcome.

Number 3: Don't Overtrain Overtraining is a situation where a machine learning model can predict training examples with very high accuracy but cannot generalize to new data, leading to poor performance in the field. Usually, this is a result of too little data, or data that is too homogenous (i.e. does not truly reflect natural variation and confounding factors that will be present in deployment), but it can also result from poor tuning of the model.

Overtraining can be particularly pernicious, as it can lead to false optimism and premature deployment, resulting in a

visible failure that could easily have been avoided. Renesas AI engineers oversee and check customer's model configurations to prevent this unnecessary pitfall.

## Example: AI for machine health and preventative maintenance

(Names and details have been changed to protect the inexperienced.)

For example, we recently had a client trying to build a machine health monitoring system for a refrigerant compressor. These compressors were installed in a system subject to rare leaks, and they were trying to detect in advance when refrigerant in the lines has dropped to a point that put the compressor at risk -- before it causes damage, overheats, or shuts down through some other mechanism. They were trying to do this via vibration data, using a small device containing a multi-axis accelerometer sensor mounted on the unit.

Ideally, this client would have collected a variety of data with the same accelerometer under known conditions: including many examples of the compressor running in a range of normal load conditions, and many examples of the compressor running under adverse low refrigerant conditions in a similar variety of loads. They could then use our algorithms and tools in confidence that the

data contains a broad representation of the operating states of interest, including normal variations as load and uncontrolled environmental factors change. It would also contain a range of different background noises and enough samples so that the sensor and measurement noise is also well represented.

But all they had was 10 seconds of data of a normal compressor and 10 seconds with low refrigerant collected in the lab. This might be enough for an engineer to begin to understand the differences in the two states -- and a human engineer working in the lab might use his or her domain knowledge about field conditions to begin extrapolating how to detect those differences in general. But a machine learning algorithm knows only what it sees. It would make a perfect separation between training examples, showing a 100% accuracy in classification, but that result would never generalize to the real world. In order to consider all the operational variation possible, the most reliable approach is to include examples in the data of a full range of conditions, both normal and abnormal, so that the algorithms can learn by example and tune themselves to the most robust decision criteria.

# Ultimate Guide to Machine Learning for Embedded Systems

CONTINUED

Reality AI Tools automatically do this by using a variety of methods for feature discovery and model selection. To help detect and avoid overtraining, our tools also test models with “K-fold validation,” a process that repeatedly retrains, but holds out a portion of the training data to for testing. This simulates how the model will behave in the field, when it attempts to operate on new observations it had not trained on. K-fold accuracy is almost never as high as training separation accuracy, but it’s a better indicator of likely real-world performance – at least to the degree that the training data is representative of the real world.

To understand our machine learning tools more fully and how they can be applied to your data, read our [Reality AI Tools technology page](#).

## Machine Learning for Embedded Software is Not As Hard As You Think

Machine Learning is hard and the challenges have been nearly insurmountable for most designers...until now. The modern Renesas Reality AI Tools

make it possible to use highly sophisticated machine learning models on all the Renesas processing platforms. From low-end 16-bit (RL78) to 32-bit (RA and RX) and up to 64-bit (RZ) families, end point-capable embedded solutions are available unlike ever before. But there are some considerations to keep in mind.

Has the Machine Learning Module been thoroughly tested and validated? It will go without saying to experienced embedded engineers, that the decision to commit any code module into an embedded system product must be careful and deliberate. Firmware pushed to products must be solid, and reliable. Usually, products are expected to have a long lifetime between updates, if updates are even feasible at all. And they are expected to work out of the box. In the case of a machine learning-based classifier or detector module, the usual validation concerns apply, but the caveats we’ve talked about elsewhere apply in spades: In particular, whether the system has been trained and tested on a large enough variety of REAL WORLD data to give confidence that it will hold up, unsupervised in the wild.

## VISION AI SOFTWARE DEVELOPMENT KIT

The RZ/V AI SDK is the most comprehensive solution for building end-to-end accelerated AI applications. With application examples included, it provides a quick and easy solution for starting your AI visions system development and start building smarter, faster, and more efficient AI-powered solutions today.



EXPLORE  
THE KITS



# Ultimate Guide to Machine Learning for Embedded Systems

CONTINUED

What are the risks and trade-offs of incorrect results?

If you know that the machine learning models results are useful, but not perfect, can customers be educated to work within known limitations? Is the use case safety critical, and if so, what are the fail-safe backups? Testing and validation is much better done in an easy to use, sandbox environment. Our tools, for example provide the user the ability to experiment, retrain, and test with as much data as they choose before the classifier ever leaves the safety of the cloud. We even support “live” testing through cloud APIs, so that the customer can have every confidence they have tested and characterized a classifier or detector module before ever committing the resources to push it to firmware and customer devices.

Will it translate to your Embedded Hardware?

Processing speed, memory, energy use, physical size, cost, time to deployment: all of these are critically balanced elements in an embedded processing system design. The resources required to deploy the classifier

and to execute operations - often within critical real-time latency constraints - will make or break your design. Machine learning approaches like Deep Learning are very powerful, but only if you’ve got space, power, and budget in your design to support its rather intensive requirements. Other methods can be reduced to sequences of standard signal processing or linear equation type operations at execution time, like ours, may be less general but can comfortably run in real-time on an inexpensive microcontroller. Don’t forget the math: it is easy and natural these days to develop detectors and classifiers in high-precision, floating point tools on large desktop or cloud-based systems.

Ensuring the dynamic range and precision is sufficient to reproduce similar performance on, a low-bit depth, fixpoint platform can also put notable constraints on embeddable algorithms.

It is important to know what the requirements will be, as there is no sense spending expensive R&D efforts on a project that cannot possibly be deployed in your target system.

Is there an early feedback mechanism?


Your detector looks good enough to release, but is it perfect? Iteration is the key to perfecting AI classifiers and detectors. Will the designers have access to what really happens in the wild? Can you set up a pilot with one or more friendly customers to give you both quantified validation of the system in a real setting, and, even better, access to data and signals that may have caused problems so that they can be incorporated into future training sets? Remember that most machine learning is data driven. Though you have analyzed the problem and synthesized a wide variety of test vectors back at the lab, you’ll never cover all possible variations. Hopefully, you’ll find the first-cut classifier or detector is robust and does its job. But few good plans survive first contact with the enemy, and having data from even one or two real customers will quickly introduce situations, noises, systematic errors, and edge cases for which no one fully planned.

How fast can you respond with improvements?

Most companies that ship products with embedded software have a long and (deliberately) onerous review and quality control process for getting new code into production.

# Ultimate Guide to Machine Learning for Embedded Systems

CONTINUED



You can't afford to mess up, so validation, walk-throughs, and other process steps necessarily take time and money -- to ensure that only quality goes out the door. Of course this is exactly the opposite of what you need for fast, iterative product improvements. But there are compromises available. Perhaps the classifier design can be reduced to a standard module, which is fully code validated, but operation of which is in part defined by a changeable set of coefficient data (something we support with Reality AI). In a design like this, updates to just the operational coefficients can be pushed, in many cases, without requiring a major code revision and trigger revalidation procedures. Hot updates are comparatively quick and safe, and the results can be revalidated in vivo, so to speak.

So what does it take to embed my classifier?

In the old school world of signal and machine learning R&D, the distance between a laboratory test prototype and an embedded deployment is substantial. The successful custom prototype was only the start of a long process of analysis of requirements and completely re-coding for the target device. But that's changing. If you know you are targeting and embedded platform, and you have some idea of the requirements, modern AI tools can help plan for it from the start. They can be configured to choose from among proven, highly deployable AI module designs, and provide for thorough testing and validating end to end in the cloud. With these tools generating an embedded classifier module may be as simple as linking in a small library function into your code base. An update as simple as pushing a small binary data file to the end device.

## Conclusion

Edge AI and TinyML have paved the way for enterprises to build smart product features that use machine learning running on highly constrained edge nodes. The sophisticated Reality AI machine learning tools from Renesas are optimized for real-time analytics of signal data and embedded deployment can cut months or even years from an R&D cycle. Even more importantly, they can often accomplish detections that elude traditional engineering models by making much more efficient and effective use of data to overcome variation. The complete suite of Reality AI software and tools running on Renesas processors help deliver endpoint intelligence in your products.

Learn more, review example projects and request a demo at [www.renesas.com/key-technologies/artificial-intelligence/real-time-analytics#tools\\_resources](http://www.renesas.com/key-technologies/artificial-intelligence/real-time-analytics#tools_resources).



# AI-as-a-Service for Signal Processing

## Overview

Reality AI software from Renesas provides AI tools, optimized for solving problems related to sensors and signals. Working with acceleration, vibration, sound, electrical, RF, and proprietary signal sets, Reality AI Tools™ identify signatures of real-world events or conditions in sensor and signal data, and create software that can then spot those signals in the wild, notifying applications and devices so they can take action. Reality AI software also has uses for certain use cases with image, LiDAR, radar other image based applications. This white paper will cover technical aspects of our approach to machine learning and the architecture of our solution.

## Classifiers and Detectors

Reality AI Tools are machine learning tools that create classifiers, predictors and anomaly detectors – executable code that takes incoming observations and determine what has occurred in the real world based on those observations. Making Classifiers, Predictors and Detectors Making classifiers and detectors involves determining which features of incoming signals are relevant to

the classification/detection problem, and developing the means to identify which combination of features correspond to which results. Classifiers, predictors and detectors can be created through many different means, including traditional signal processing, statistical methods, or through machine learning. Reality AI Tools use advanced machine learning techniques (described below), optimized for complex signal recognition.

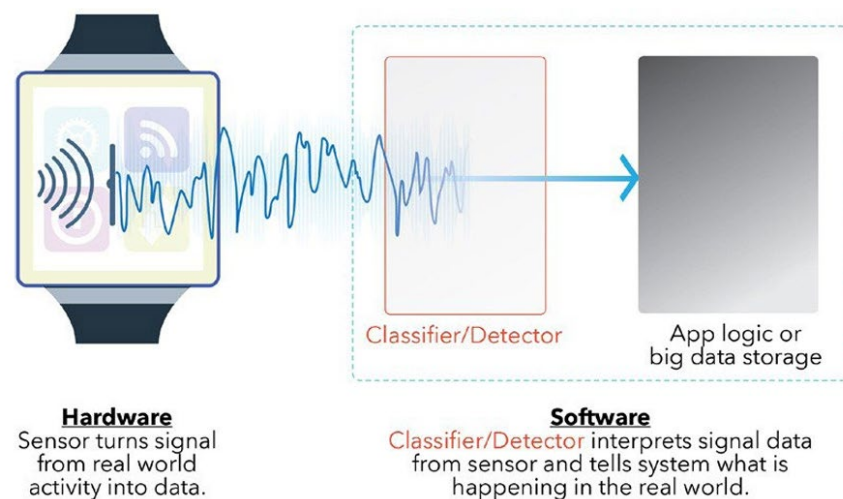


Figure 11. Classifier Example: An exercise app running on a smartwatch contains classifier/detector code that reads incoming accelerometer/gyroscope data, determines that an exercise has been completed, and determines which exercise it was.

## Using Classifiers, Predictors and Detectors

Once made, classifier, predictor or detector code can process new observations and deliver results. Results can be in the form of binary determinations, type classifications, or predictions/estimates of quantitative measures, depending on how it was constructed.



# AI-as-a-Service for Signal Processing

CONTINUED

## Signals are Different

What are Signals?

Signals are sensed-phenomena originating in the physical world, and when digitally sampled contain an umber of characteristics that make them fundamentally different from other kinds of time-series and spatial data, like machine logs, discrete event data and locations data. Signals data include sound, images, accelerometry, vibrations, electrical, RF and inputs from other sensors covering a wide range of phenomena.

What Makes Signals Different?

Signals are often characterized by:

HIGH SAMPLE RATE

High number of samples per unit time/space relative to other kinds of data (such as event logs, discrete time series, or geolocation data). This gives non-signal tools great difficulty because important distinguishing features are obscured by high-dimensionality and mathematically complex relationships.

JITTER & PHASESHIFTS

Irregular variation and time-shifting of target signatures in data due to physical phenomena, as well as inexact relationship between observed events and measured signals.



Figure 12. Sound of ocean waves crashing into rocks

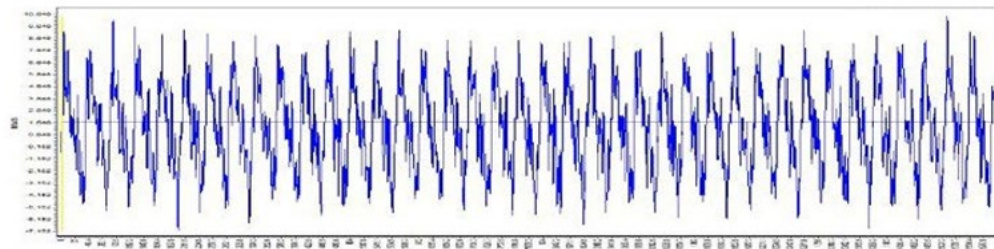


Figure 13. Vibrations of a bearing in an industrial machine

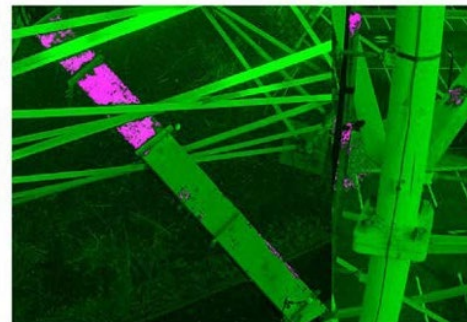


Figure 14. Images of a cell-tower collected by drone  
Reality AI has highlighted rust in purple

# AI-as-a-Service for Signal Processing

CONTINUED

## DYNAMIC NOISE & DISTORTION

Obscuring elements in collected signals that result from the physics of the sensors, and variation or other natural phenomena in the collection environment.

## TRANSIENT SIGNATURES

Brief and non-stationary targets in signals data that can be very hard to separate from noise and other aggregated signals, but which often contain the most revealing signatures. Local or transient features may be just as important as more widespread, slower, or persistent signal features; best results come when both are considered simultaneously.

These aspects and others make signals data very difficult to analyze using statistical or machine learning tools intended for discrete time-series or categorical data.

## Working with Signals the Traditional Approach

Engineers, however, have developed a number of tools for making signal data more tractable for statistical analysis. Signal processing typically begins by subjecting the incoming signal to a Fast Fourier Transform (FFT), filter banks, or a linear systems analysis to find the amount of energy in different parts of the signal's frequency and time domains.

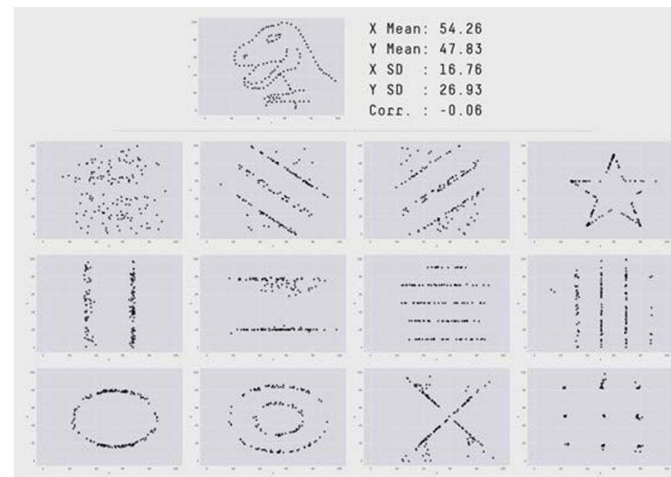


Figure 15. Why basic statistics are never enough. All of these plots have the same X and Y means, the same X and Y standard deviations, and the same X:Y correlation. Without the ability consider all of the source detail, your algorithm would never see any of these patterns in your data.

This is the way signal analysis has been taught in engineering schools since the 1970s, and there are a number of robust tools supporting this approach – perhaps the most widely used being the Matlab Signal Processing Toolbox<sup>1</sup>. The output of these processes can then be treated as features in statistical detection methods, but rely heavily on the engineer's understanding of the physics of the connection between the signal and what she is trying to detect.

## Machine Learning for Signals

### Traditional Machine Learning Doesn't Work Well on Signals

For years now machine learning has posed an obvious alternative to these traditional methods of analyzing signals data. But for all the reasons above, signals data confounds most machine learning tools. The high dimensionality of signal analysis problems and complex mathematical relationship between the sampled values and underlying features leads to long processing times and low accuracy for most real world problems.



# AI-as-a-Service for Signal Processing

CONTINUED

It's the Features!

At the heart of the matter is feature discovery. In machine learning parlance, features are the specific variables that are used as input to an algorithm. Features can be selections of raw values from input data, or can be values derived from that data. With the right features, almost any machine learning algorithm will find what you're looking for. Without good features, none will – and that's especially true for real-world problems where data comes with lots of inherent noise and variation.

When signals become complex enough so that the individual values recorded no longer work as features, most engineers turn first to the traditional RMS energy, peak-to-peak measures, or the FFT to create the features – treating the averages or FFT coefficients as the relevant features for training. But the problem is that these features discard much of the information necessary to make the judgements that you most want to make. RMS can certainly help identify some conditions, and an FFT run with relatively high frequency and time resolution can be quite valuable. But high-level descriptive statistics discard almost all the essential signature information you need to find granular events and conditions that justify the value of the sensor

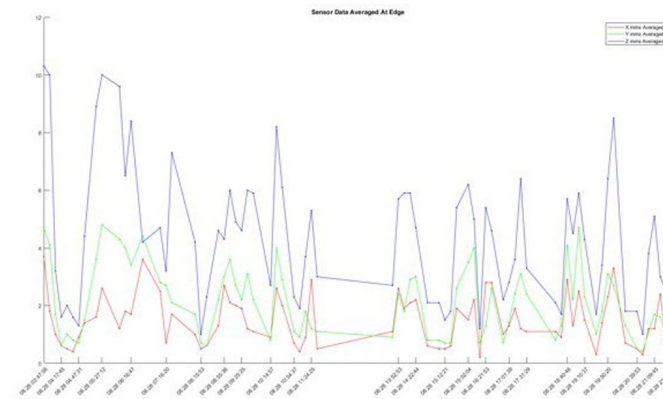


Figure 16. This example shows a time series of data from an accelerometer attached to a machine in a manufacturing facility. X, Y and Z components of the acceleration vector are averaged over one second. There is very little information in this data – in fact, just about all it can tell us is which direction is gravity. This data was provided from an actual customer implementation in a manufacturing facility, and is basically useless for anomaly detection, condition monitoring, or predictive maintenance.

implementation in the first place. And as this excellent example from another domain shows, descriptive statistics just don't let you see the most interesting things:

Instead of relying on aggregate features defined by the traditional signal processing tools, Reality AI discovers features directly from the raw sensor signal data in all of its inherent noisiness and complexity. This approach allows for consideration of information in both time and frequency domains, and allows for detection of conditions with much more subtle signatures.

## Data-Driven Feature Discovery

Our algorithms are based on patented techniques using mathematics described in the literature as sparse decomposition or sparse coding, as well as a number of other related proprietary techniques. These techniques operate unsupervised or semi-supervised to dynamically discover an optimal feature set for describing similarities between signals of the same class and differences between signals of different classes.



# AI-as-a-Service for Signal Processing

CONTINUED

When using Reality AI Tools, we will encourage you will point the algorithms at your source data in its most detailed, un-preprocessed form, and run a process that we call AI Explore™. AI Explore will then attempt to identify the features that best separate your training classes (if your data is labeled), or that best characterizes “normal” (for unlabeled anomaly detection). It will then offer a selection of machine learning models based on the best features – ranking them by accuracy and by computational complexity. Those models can then be trained on whatever data is available, and validated using holdout sets or a variety of statistical techniques.

Once the right features have been identified, the classification/detection problem becomes much simpler and even basic machine learning algorithms become much more effective– delivering more accurate results with less training data.

This approach to data-driven feature discovery is the core insight that drives Reality AI technology and its unique effectiveness with signal data

## Reality AI vs Deep Learning

One other type of machine learning approach that has recently shown good results on a number of different signal-related problems is Deep Learning. Deep Learning is a branch of

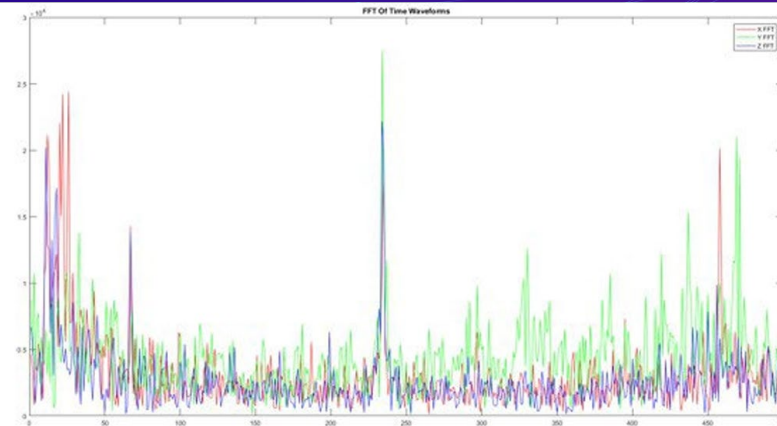


Figure 17. This example shows vibration data pre-processed thru a Fast Fourier Transform (FFT) at high frequency resolution. The X-axis is frequency and the Y-axis is intensity. This data is much more useful than Figure 6 – the spikes occurring at multiples of the equipment’s base rotation frequency give important information about what’s happening in the machine and is most useful for rotating equipment. FFT data can be good for many applications, but it discards a great deal of information from the time-domain. It only shows a single snapshot in time – this entire chart is an expansion of a single data point from Figure 16.

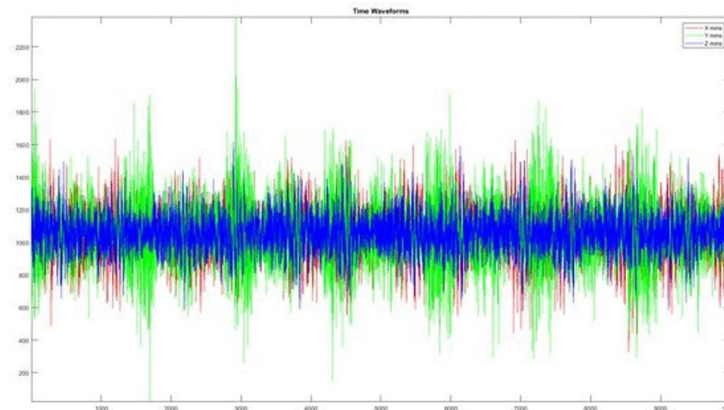


Figure 18. Raw time-waveform data as sampled directly from the accelerometer. This data is information-dense – being the raw data from which both the simple averages in Figure 16 and the FFT in Figure 17 were computed. Here we have extremely detailed frequency information, coupled with important time information such as transients and phase. We also have all of the noise, however, which makes this kind of data very difficult for human analysts to use directly. But a data-driven feature discovery algorithm like Reality AI can extract maximum value from this kind of data, creating a custom transform that zeros in on exactly those parts of the signal – and only those parts of the signal – that are relevant for your detection problem.

# AI-as-a-Service for Signal Processing

CONTINUED

machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers with complex structures – usually composed of recurrent or convolutional neural networks. Deep Learning also uses an approach for datadriven feature discovery – essentially using lower layers of convolutional neural networks or similar algorithms to find features that higher layers can then learn from. Deep Learning has been used to good effect for image sorting and tagging, image and scene interpretation, automatic speech recognition, and natural language processing.

## Good at Different Things

While there is certainly overlap between problems for which Deep Learning is well suited and problems that Reality AI can address, there are also some differences. In images, for example, Deep Learning is particularly good at identifying objects and characterizing scenes, while Reality AI is particularly good at problems related to subtlety in texture: identifying surfaces and spotting surface discontinuities or nuanced anomalies. You can see this surface classification and discontinuity identification

aspect most readily in image-related data, but that same approach is at work with signals as well, for example in identifying subtle variations in vibration or machine noise to spot maintenance conditions in complex machinery.

## Easier Setup, Less Training Data and Embedded Deployment

But even when both Deep Learning and Reality AI are capable of delivering good results, Reality AI has some important advantages: Technical Architecture

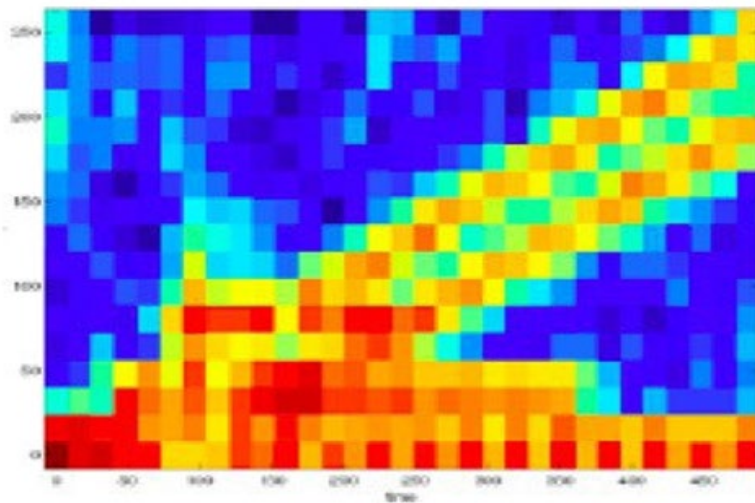


Figure 19. Using a Fourier transform to define features in complex, dynamic signals, there is loss of information due to the transform, noise and jitter.

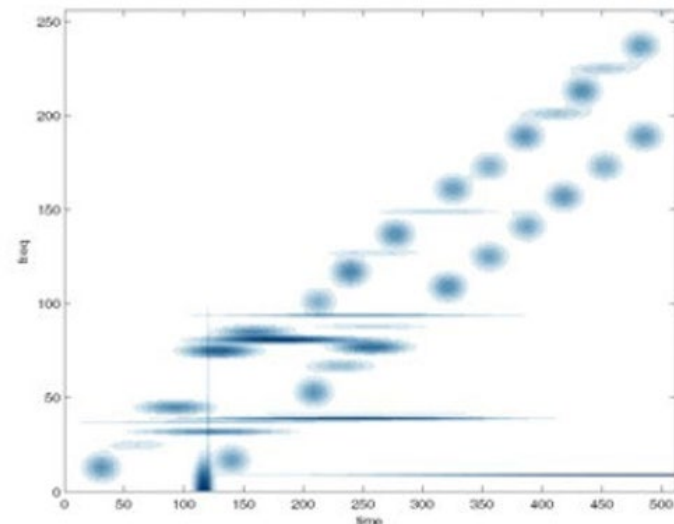


Figure 20. Our methods consolidate features and de-blurr jittered information. Features are much more precise, and more easily learned.

# AI-as-a-Service for Signal Processing

CONTINUED

The Right Tool for the Right Job Examples:

	Deep Learning	Reality AI
UAV/ Drone	Asset surveys identifying dump trucks and bulldozers	Surface classifications identifying characteristics of microterrain
Sound	Understanding speech	Understanding machine noise Complex anomaly detection
Images	Identifying objects Describing scenes	Texture and related nuance Subtle detection in noise or at limits of resolution

	Deep Learning	Reality AI
Setup Time	Each new problem requires a new model to be designed from scratch, leading to long lead-times and upfront costs	Configuration in hours
Data Requirements	Large amounts of training data required for most every problem. Multiple layers of neural networks with lots of nodes each requiring sufficient data for statistically significant training	Much less training data required - often several orders of magnitude less
Training Times	Weeks to months	Hours to days
Hardware Requirements	Often requires specialized hardware to deliver required processing power - both for training and in field deployment	No specialized hardware required
Embedded Deployment	Possible on specialized hardware optimized for deep learning and carrying significant power requirements	OK for microcontroller or DSP-driven environments.  Suitable for cycle, memory- and power- constrained deployment.  Suitable for real-time detection at the edge



# AI-as-a-Service for Signal Processing

CONTINUED

Customers interact with Reality AI thru two methods:

## Reality AI Tools

Web-based interface to tools for data curation, classifier maintenance, retraining, testing and validation, and API dashboard monitoring.

## Reality AI API

Both Reality AI Tools and the Reality AI API are front-ends to a multi-tiered server environment that handles all transactions.

Customers who are using Reality AI Tools to create embedded classifiers will also have interaction methods specific to their deployment.

## Cloud Deployment Architecture

Both Reality AI Tools and the Reality AI API are front-ends to a multi-tiered server environment that handles all transactions.

## Cloud Deployment Scalability

Reality AI relies on a multi-tiered, load-sensing, load-balancing server architecture that determines when additional capacity is needed and then provides it:

### Reality API Management

Handles API security and determines need for additional Process Servers to handle incoming volume. We employ a RESTful

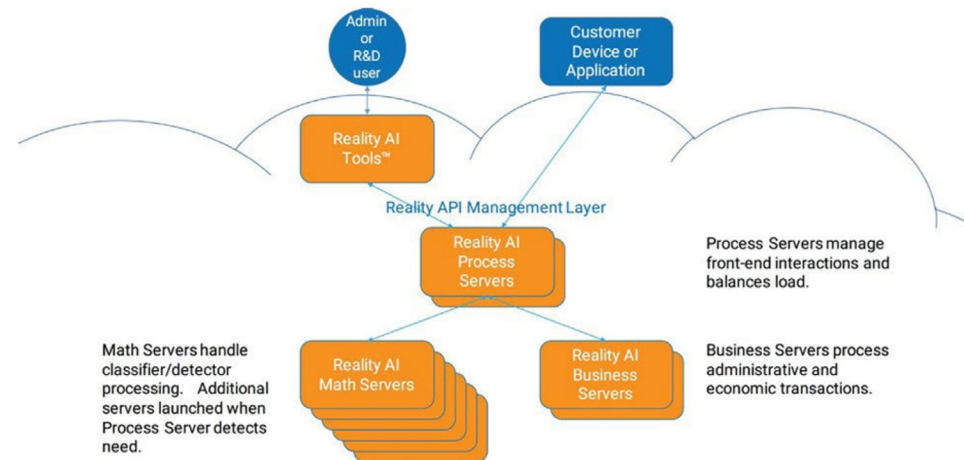


Figure 21. Reality AI Technical Architecture for Cloud Deployments Multi-tiered server architecture for scalable processing and loadbalancing

API design to enable scalable, asynchronous processing for our customers.

### Process Servers

Manages incoming API transactions from Reality AI Tools and from customer API usage. Each incoming API call, once validated, is then farmed out to Math Servers and Business Servers as appropriate. When results are returned, they are combined into a response and passed back to the requesting application. Monitors Math Servers and Business Servers and provisions additional capacity as needed.

### Math Servers

Interfaces with other systems at Reality AI software from Renesas, and handles both administrative and economic transactions (eg usage tracking).

### Business Servers

Interfaces with other systems at Reality AI software from Renesas, and handles both administrative and economic transactions (eg usage tracking).

As additional capacity is needed, each service tier is able to provision and spin up additional servers in the tier below, allowing services to scale seamlessly and without performance degradation.

# AI-as-a-Service for Signal Processing

CONTINUED

## Embedded Deployment

For customers requiring embedded deployment, code that would have run on a Math Server in the cloud is instead exported in a form suitable for adaptation to the specific embedded target. Typically, this is in the form of compiled code, though other options are also available. Contact us for more information.

## Security

Reality AI takes security seriously, and our systems are engineered for privacy and security from the ground up.

## Standard Security Measures

Our servers are secured by industry standard security measures, including SSL encryption of all IP traffic and web transactions.

Customers are provided unique administrative log-in credentials, and API transactions handled using security tokens generated on a dynamic basis.

In addition, we limit public address exposure of our machines to monitored API portals.

## Data Security

Customers can choose to securely upload and store data with us in a persistent data store. This is maintained separately for each user, and data is not shared between or aggregated across customers.

Customers may choose to link data dynamically from other data sources. For example, they may furnish us unique credentials to an Amazon AWS bucket, which we will access on a transient basis only when needed for processing. We can also support links to private servers, so that your data remains in customer control up to the moment it is processed. Unless otherwise required by the customer's use case, Reality AI discards all processed observations.

Finally, for sensor data being shown to a deployed, production classifier/detector where that data is sensitive and must be kept private, or where the data is too sensitive to transmit thru the cloud (some sound recordings, for example), it is possible to configure the system to complete pre-processing locally so that only feature vectors are transmitted for classification. In this way, features unrelated to the classification are not available for observation, and raw signals need not leave the local device.

## Anonymity and Data Sanitization

Because our algorithms are sensor-agnostic and 100% data-driven, it is entirely possible for customers to use our technology to create classifiers/detectors

without disclosing any specifics about the data at all, and using completely anonymized labeling. This functionality is particularly attractive to customers working with personally identifiable information, or those with proprietary sensors and sensor configurations who do not wish to disclose details of the data collection methods or pre-processing.

## Conclusion

Reality AI software from Renesas provides AI tools with real-time analytics optimized for solving problems related to sensors and signals. Working with non-visual sensing and proprietary signal sets, the software identifies signatures of real-world events or conditions in the sensor and signal data so users can create models to deploy on their systems to identify signals and notify applications and devices to take action. This Reality AI software running on Renesas processors helps deliver endpoint intelligence to end products and support solutions across all markets.

## Resources

[renesas.com/realityai](https://renesas.com/realityai)

# Data Collection for Edge AI / Tiny ML with Sensors

## Overview

Reality AI software from Renesas provides solution suites and tools for R&D engineers who build products and internal solutions using sensors. Working with accelerometers, vibration, sound, electrical (current/voltage/ capacitance), radar, RF, proprietary sensors, and other types of sensor data, Reality AI software identifies signatures of events and conditions, correlates changes in signatures to target variables, and detects anomalies. Since data collection and preparation is both the most costly part of any machine learning project, and also the place where most failed projects go wrong, Reality AI software contains functionality to keep data collection on track, to assist with its pre-ML processing, and to get the most out of it using synthetic augmentation techniques. This white paper covers the approach we recommend for data collection planning, execution, and post-collection processing.

## Data Collection Planning

Data collection is the most time consuming, most expensive part of any machine learning development effort — and it is also

the part where the seeds of project failure are most easily set. Planning for data collection is the stage where problems can be foreseen and avoided without extra expense. Therefore we always advise customers to create a comprehensive data collection plan before beginning.

Avoid the biggest pitfalls in data collection with good planning

- Inadequate data coverage
- Insufficient ground truth
- Unreliable or inappropriate instrumentation
- Inconsistent collection protocol

So what are the elements of a good data collection plan?

## DATA COVERAGE MATRIX

The Data Coverage Matrix lays out the breadth of data that would need to be collected if examples of every possible combination of target, environmental or background condition, and equipment-related variation were to be covered. It is rarely necessary to collect data from each possible combination, but understanding these sources of variation is key to creating an adequate plan.

## INSTRUMENTATION PLAN

The Instrumentation Plan covers hardware and communication aspects of the data collection effort. It should also layout explicitly how ground truth will be collected. For machine learning development, it is always important to have some way of knowing — as definitively and precisely as possible — what the correct prediction result should be from the data collected (the “ground truth”).

## DATA COLLECTION PROTOCOL

The Data Collection Protocol spells out the specific steps that will be followed during data collection. How the apparatus will be set up, how trials will be run, how results will be recorded, how data will be manipulated and stored. The protocol should be as detailed as possible and should contain checklists that can be followed by those doing the actual collection. Consistently following a comprehensive and carefully constructed protocol is the best way to ensure that the data is usable and that machine learning results are not “tricked” by artifacts in the data.



# Data Collection for Edge AI / Tiny ML with Sensors

CONTINUED

## DATA COVERAGE MATRIX

Data coverage refers to the distribution of data across target classes or values, as well as across other variables that are significant contributors to variation. Target classes are just the things you are looking to detect or predict. For example, if you're building a model that will detect whether a machine is behaving normally or is exhibiting a particular kind of fault, then "Normal" and "Fault" are the target classes.

But other variables may also be important. If you have good reason to expect that "Normal" and "Fault" will look different when the machine is operating in different modes, those modes are contributors to variation and should be tracked as metadata variables. Different versions of equipment or different installation types, for example, are almost always significant contributors to variation.

The purpose of coverage planning is to make sure you collect enough data to capture and overcome the variation inherent in what you're trying to measure. To do that you'll need a statistically significant number of observations taken across a reasonable number of combinations of different target classes and metadata variables (contributors to variation).

	Operating Mode 1	Operating Mode 2	Operating Mode 3
Normal	# of actual or planned trials/observations		
Failure			

Example of a simple data coverage matrix

So in our example above, we have two target classes and a couple of different modes.

This is a data coverage matrix and allows us to plan for the data we need to collect. In the beginning, since we don't yet know whether it's possible to detect fault conditions in different types of equipment with the same machine learning model, we probably want a separate matrix for each equipment type – making this more of a data cube than a 2x3 matrix.

A fully articulated data coverage matrix would cover all of the material variations in target, background environment and equipment that we believe would make for differences in the data, or that we need to test to have confidence in the outcome. Usually, these have too many dimensions to list in table form, so instead, we just list each of the dimensions and the categories

or ranges within each. These can then be used to construct a table or database for actual tracking.

If we bucket Equipment Age by five year increments, temperature by increments of 10°F, and humidity by 10%, this coverage matrix has  $6 \times 5 \times 4 \times 4 \times 15 \times 7 = 50,400$  cells. Ideally, each of these cells would be populated with at least 25 independent observations or trials — say 25 machines in each combination of model, size and age (i.e.  $25 \times 5 \times 4 \times 4 = 2,000$  machines) that could be observed a statistically significant number of times in each failure mode, in each environmental condition.

Why 25 independent observation sets per cell? It comes from the Central Limit Theorem and the Law of Large Numbers in statistics and is usually cited as a rule of thumb for achieving statistical significance. In efforts like those we are discussing here, what we really

# Data Collection for Edge AI / Tiny ML with Sensors

CONTINUED

need are enough independent observations to be able to account for noise and variation in the data from sources that we cannot control (or may not even be aware of) and that the model will need to learn to ignore. You may need more, or fewer, depending on the amount of variation in target, background, and equipment.

But the real world is not ideal, and the ideal population of the Data Coverage Matrix is not always possible.

Target: Failure Mode	Equipment: Model	Size	Age	Environment: Temp	Humidity
Normal	QV22	15	New to	-40°F to 110°F	20% to 80%
Motor	QA22	20	15yrs		
Rotor	QV23	22			
Capacitor	SA22	25			
Bearing	SA23				
Coil					
6	5	4	4 (5yr bucket)	15 (10°F bucket)	7 (10% bucket)

Example of a more complex data coverage matrix

## How Much Data Do You Really Need? Iterative Data Coverage

The purpose of the data coverage matrix is to help understand how much data you really need. How much is ideal, and how much will be the bare minimum? The real answer to “How much data do I need?” is “You need enough.”

There is no one-size-fits-all answer to the question. It very much depends on the specifics of your problem and its difficulty, and the amount of variation in targets, backgrounds, and equipment.

Therefore, we always recommend collecting data iteratively and using empirical results to guide cost-effective data collection. You typically do not need to cover every single cell in the coverage matrix, but a smart collection plan that fills

one section of it comprehensively and then tests the degree to which more data from other sections is required can get you to effective, reliable results with a minimum of cost.

It works like this:

In our example above of a “More Complex Data Coverage Matrix”, we start with a “Proof-of-Concept” test. Perhaps we pick one model and size, get 4 or 5 examples at different ages into a test facility where we can induce the target failure modes. If we can control the environmental variables in our test facility, we collect data in 4-5 different environment types, but if we can’t we just record what the environmental variables are and do our best to keep them constant through the data collection.

We collect data to fully populate the 6 x 1 x 1 x 4 x 3 x 2 = 144 cells of this matrix and use the data to build a version 1.0 of our prediction model. In evaluating the results, we would look carefully at the prediction accuracy for each failure mode, and the false positive rate on Normal, but we would also look to see how that accuracy varies across the other metadata categories. Did it do equally well across age, temperature, and humidity? If not, those might be categories that require additional coverage in later stages.

We can also do some selective holdout testing to see which categories are important. Perhaps we train our model on data from 4 of the 5 machines, holding out data from the oldest. Does our model

# Data Collection for Edge AI / Tiny ML with Sensors

CONTINUED

predict that holdout data well? If so, you have good reason to believe that you will generalize well across age and can deprioritize that dimension in the later collection. If not, the opposite.

Similarly, we could run a test of a sixth machine, either the same model in a different size or a different model in the same size. Collect data, and use our ML model to predict. How did it work? If accuracy was low, add some of the data to the training set, retrain, and test again (we recommend using K-fold validation for this kind of test). If accuracy improves, that tells you that additional data coverage will likely get you good results — you just need coverage of different models or sizes.

Steps for iterative data coverage

1. Pick a section of the Data Coverage Matrix for an initial focus
2. Collect data to cover that section as comprehensively as possible
3. Build an initial version of the model based on initial data
4. Examine variation in accuracy across dimensions of the matrix and prioritize additional collection accordingly
5. Test generalization to new sections of the matrix:

Target: Failure Mode	Equipment: Model	Size	Age	Environment: Temp	Humidity
Normal Motor Rotor Capacitor Bearing Coil	QV23	15	New to 15yrs	60 70 80	30% 60%
6	1	1	4	3	2

Example of the data coverage matrix for the “proof-of-concept”

- a. Collect data representing a new category
  - b. Test the previous version of the model against new data and evaluate accuracy.
  - c. If accuracy is high, deprioritize additional data collection.
  - d. If accuracy is disappointing, incorporate new data into the training set and test again. If accuracy improves, prioritize additional data collection for this region. If not, investigate differences in data more closely.
6. Continue until all regions of the Data Coverage Matrix have been explored and tested

So how much data will be enough? As a general rule of thumb, if you follow the Iterative Data Coverage method outlined above, you will likely wind up with full

population (data from at least 25 independent trials) in around three of cells in the Data Coverage Matrix, with enough additional data collected for testing that no row or column is completely empty

## INSTRUMENTATION PLANNING

The second main section of the Data Collection plan covers instrumentation and hardware. The instrumentation plan should cover what sensors will be used, how they will be sampled, where and how they should be mounted, what control apparatus will be required, and how data will be stored and forwarded to the machine learning environment for processing and use. If there are environmental considerations (high temperatures, hazardous environments, etc.) these should



# Data Collection for Edge AI / Tiny ML with Sensors

CONTINUED

be highlighted and special requirements they put on the equipment specified.

In early-stage Proofs-of-Concept, shortcuts make sense. Dev boards with sensors pre-mounted or pre-integrated can make a lot of sense and help you collect data quickly and inexpensively. But they have their limitations.

## RECOMMENDATIONS FOR INSTRUMENTATION PLANNING

- Over-instrumentation
- Collect rich data
- Incorporate ground truth

## Over-Instrumentation

Once past the initial PoC, we generally recommend over-instrumentation for the first serious product-development data collections, mainly to collect data that can help explore the interaction between component choices and machine learning accuracy.

During this stage, we suggest using more sensors than you actually intend to put in the final device solution. If there are different candidates for the accelerometer or mic placement, for example, place a sensor in each potential location and collect simultaneously from all of them. Similarly, use high-fidelity sensors for those early collections, sampled at the highest possible rate. Software like Reality AI Tools® can then

explore the data to select the optimum set of data channels, and can examine the relationship between solution accuracy and component sample rate, bit depth, and sensitivity — and do it much more quickly and cheaply than repeating data collection with different sensor choices.

## Rich Data

We always recommend collecting data at the most detailed level possible — particularly in the early stages of development. Many engineers working with

vibration data, for example, will collect only traditional RMS energy or peak-to-peak measures. Perhaps they'll collect FFT results for a set of frequency bands they expect to be important. But these data, if that's all that is collected and retained, discards much of the information necessary to make the judgments that you most want your machine learning algorithm to make.

RMS can certainly help identify some conditions, and an FFT run with relatively high frequency and time resolution can be

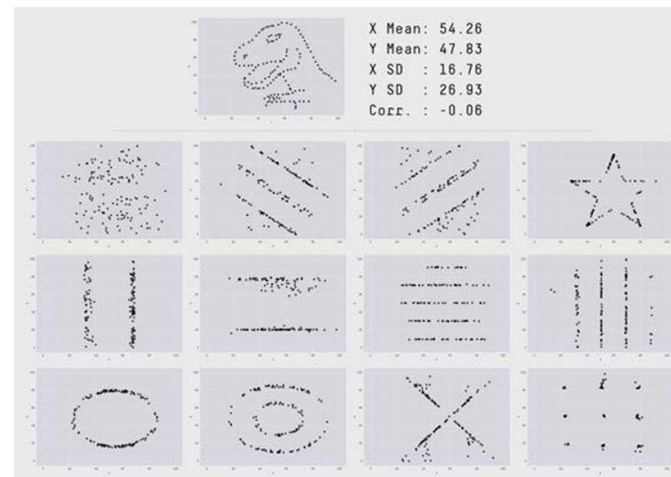


Illustration of why aggregate statistics are never enough. All of these plots have the same X and Y means, the same X and Y standard deviations, and the same X:Y correlation. Without the ability to consider all of the source detail, your algorithm would never see any of these patterns in your data.

# Data Collection for Edge AI / Tiny ML with Sensors

CONTINUED

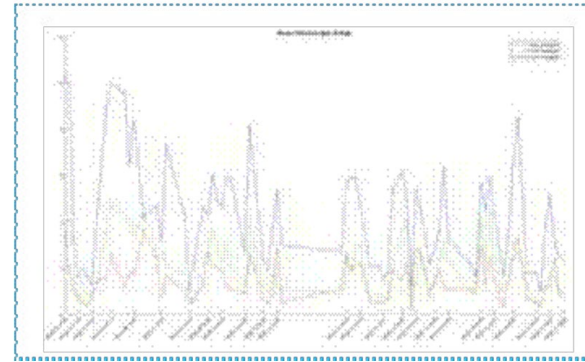
quite valuable. But high-level descriptive statistics lose almost all the essential signature information you need to find granular events and conditions. And as this excellent example from another domain shows, descriptive statistics often don't let you see the most interesting things:

Instead of relying on aggregate features defined by the traditional signal processing tools, Reality AI discovers features directly from the raw sensor signal data in all of its inherent noise and complexity. This approach allows for consideration of information in both time and frequency domains and allows for detection of conditions with much more subtle signatures.

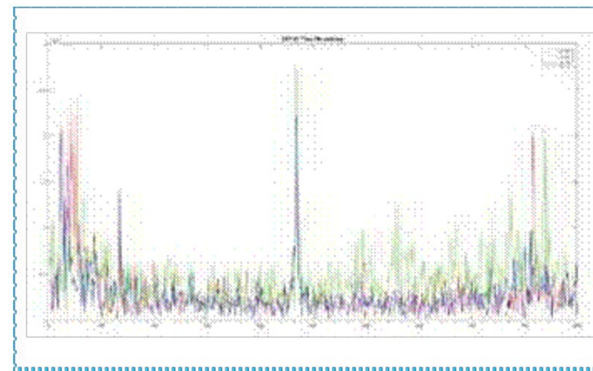
When deciding what data to collect and retain, always opt for the most information-rich data available, reducing only as a final tuning and optimizing step in the final stages of development.

## Ground Truth

Perhaps the most frequently overlooked aspect of data collection — yet also the most important — is the question of how to collect ground truth.



Example showing a time series of data from an accelerometer attached to a machine in a manufacturing facility. X, Y, and Z components of the acceleration vector are averaged over one second. There is very little information in this data. This data was provided from an actual customer implementation in a manufacturing facility and is basically useless for anomaly detection, condition monitoring, or predictive maintenance.



Example showing vibration data pre-processed using FFT. The X-axis is frequency and the Y-axis is intensity. Peaks at multiples of the equipment's base rotation frequency give important information.

FFT data is most useful for rotating equipment and can be good for many applications, but it discards the time-domain. It only shows a single snapshot in time – this entire chart is an expansion of a single data point from the previous example.



# Data Collection for Edge AI / Tiny ML with Sensors

CONTINUED

Ground truth refers to the real outcome that you are trying to predict: for example, was the machine actually normal, or was there really an error condition? Ground truth is important both as a basis for training and as a basis for judging accuracy. If you don't know what any of the data really was, how will you ever know whether your model is predicting accurately?

Your instrumentation plan should include a means of determining and collecting ground truth with as much accuracy and as much time precision as possible. In some cases, this may mean deploying a second set of sensors that enable the determination. In other cases, it may mean capturing maintenance logs, or recording observer perceptions. But whatever it is, make sure it's covered by your plan.

## Approaches for Collecting Ground Truth

### EXPERIMENTAL DESIGN

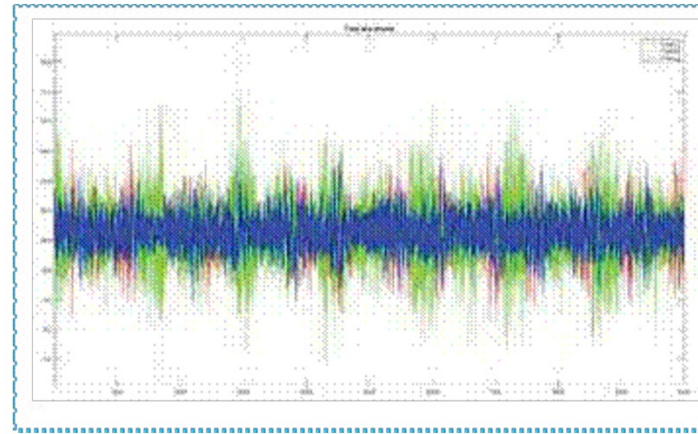
Set up conditions to create specific ground truth values in advance and collect data in blocks

### SIMULTANEOUS MEASUREMENT

Add additional sensors that allow complete determination of ground truth state so that signals can be correlated

### DIRECT OBSERVATION

Operators or data collectors record events or states



Raw time-waveform as sampled directly from the accelerometer. This is information-dense and is the raw data from which both previous examples were computed. Here we have both frequency and time information, including transients and phase. This kind of data is very difficult for human analysts to use directly, but is the best for algorithmic tools like Reality AI, who's feature discovery process zeros in on exactly those parts of the signal – and only those parts of the signal – that are relevant for the specific problem.

### SERVICE LOGS

Events determined after observation by correlating to other documentation

### AFTER-THE-FACT INVESTIGATION

Most often used in anomaly detection when other sources of ground truth are unavailable. Anomalous readings are investigated to determine what actually happened.

Even if you plan to use an anomaly detection approach, where data will not (or can not) be systematically labeled with ground truth values, you will need a plan for determining ground truth. When an anomaly

is detected, you will need some way to determine whether or not it is valid — without that kind of feedback you will never be able to determine the model's accuracy, and there will be no way for it to improve.

### Data Collection Protocol

The Data Collection Protocol is the step-by-step instructions for performing the data collection.

These steps should be laid out as specifically as possible to minimize inconsistencies in the way data is collected.



# Data Collection for Edge AI / Tiny ML with Sensors

CONTINUED

## Things to Cover in Data Collection Protocol

- Installing instrumentation
- Site preparation and recording of metadata
- Initiation of data collection
  - How to initiate or induce conditions
  - Stability or equilibrium requirements for reliable data
- Recording data and ground truth
- Ending data collection
  - Minimum length of observation or time-in-class
  - How to terminate conditions
- Data annotation - how observations should be logged and files should be named
- Data storage and transmission - how and how often data should be recorded, harvested, and loaded into staging area for further processing

## Collection Monitoring

### And Data Readiness

Another common issue we see in data collection efforts is failure to monitor data collection and spot problems early. Issues with instrumentation, file capture, and other technical snags happen often, and if left undetected until the data analysis step

once the collection is complete, risk needing to discard entire data sets. More than once, we have seen projects need to start over from scratch due to minor instrumentation or processing issues that, though easily correctable, rendered data unusable.

To address these issues in Reality AI Tools, we have added functionality for automated Data Readiness Assessment.

### Automated Data Readiness Assessment

Every time a new source data file is added to a project in Reality AI Tools, the system reads the file, parses it, and looks for issues. If any are found, they generate immediate user alerts. The system also generates statistics that can be used to identify trends and report on progress.

Reality AI Tools data readiness reports on:

#### FILE CONSISTENCY

File size, file type, number of data columns, sample rate, and other file-level demographics. Inconsistencies in these categories can indicate systemic problems in data collection or in post-collection processing.

#### DATA QUALITY

Issues that prevent a file from being read, such as corrupted files, invalid characters, blank rows, and missing data elements. It also looks for suspicious data, such as blocks of data with repeating patterns or zeros.

#### DATA COVERAGE

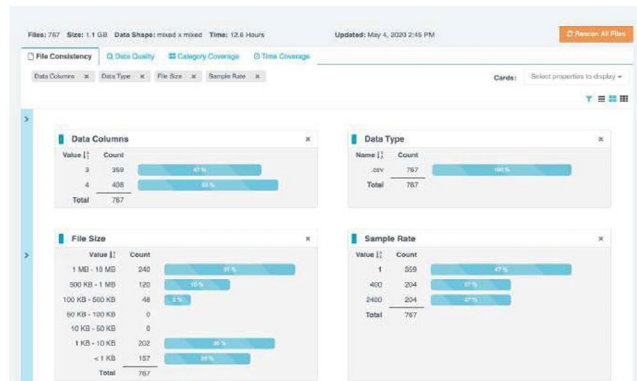
Cumulative statistics showing how much data has been collected for key target and metadata categories - can be used to track progress vs the Data Coverage Matrix in a Data Collection Plan.

#### TIME COVERAGE

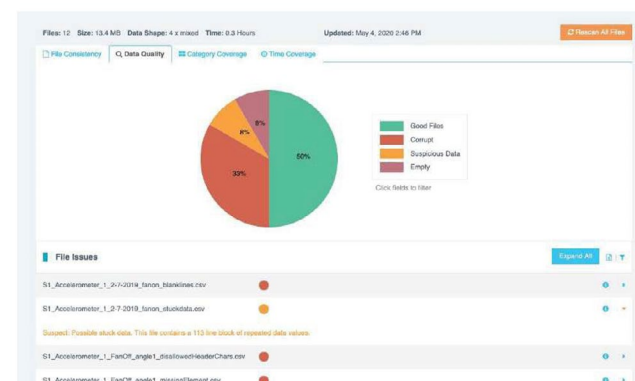
Cumulative statistics describing data in terms of time-in-class for key target and metadata categories. This is particularly useful for looking at data coverage for real-time streaming applications when the length of the decision window or smoothing methods are under evaluation.

# Data Collection for Edge AI / Tiny ML with Sensors

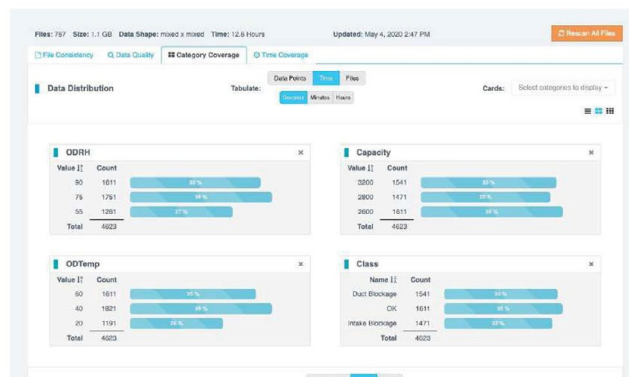
CONTINUED



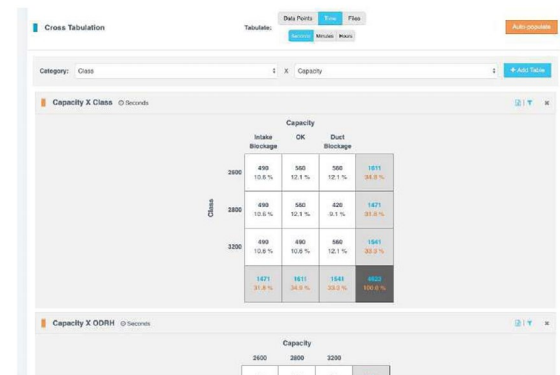
Screenshot of a File Consistency report from Reality AI Tools, run automatically whenever a file is loaded. Shows file-level demographics. Inconsistencies or unexpected values should be investigated immediately - they may indicate problems in data collection or post-collection processing.



Screenshot of a Data Quality report in Reality AI Tools, run automatically whenever a file is loaded. Shows issues spotted when the file is read that either prevent use entirely (Corrupt or Empty), or that indicate a potential problem with instrumentation (repeated or frozen data, zeros, etc.)



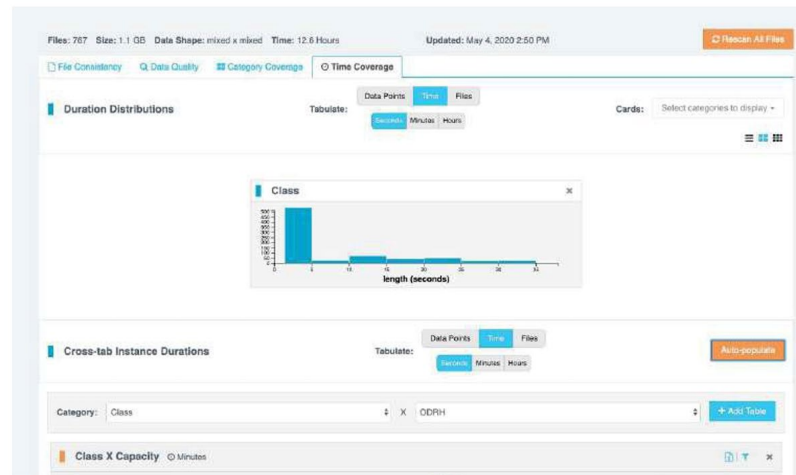
Screenshot of a Data Coverage report in Reality AI Tools, run automatically whenever a file is loaded. Shows amount of data collected for values of target and metadata variables, for tracking against a Data Coverage Matrix in a Data Collection Plan.



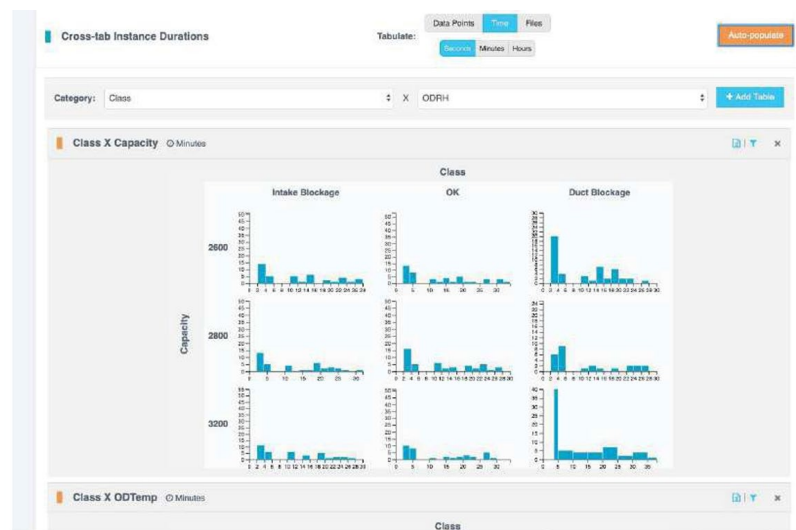
Screenshot of a Data Coverage cross-tab in Reality AI Tools to support analysis and monitoring of coverage as data is collected. Data Coverage statistics can also be downloaded for use in other reporting software as needed.

# Data Collection for Edge AI / Tiny ML with Sensors

CONTINUED



Screenshot of a Time Coverage report in Reality AI Tools, run automatically whenever a file is loaded. Shows the amount of data collected for values of target and metadata variables by time-in-class. This is particularly useful for real-time processing applications where the length of the decision window or smoothing over time is a consideration.



Screenshot of a Time Coverage cross-tab in Reality AI Tools. Time Coverage statistics can also be downloaded for use in other reporting or analysis software as needed.



# Data Collection for Edge AI / Tiny ML with Sensors

CONTINUED

## Data Collection Process

Using Reality AI Tools for automated data readiness assessment and generation of machine learning models, it is easy to implement an Iterative Data Coverage process as discussed above.

For each coverage area, you initiate data collection and regularly run Data Readiness Assessment in Reality AI Tools. Any issues uncovered can be resolved quickly, and good data can be tested on the most recent version of the ML model. If accuracy is good, you can conclude that the model is generalizing well to the current coverage area. If not, additional data from this coverage area may be required. In any event, errors made by the model should be added to the training set and the model retrained.

When doing extended field testing, you may want to run a similar process. Here the objective is also to ensure that collected data is high quality and that ML models can be continuously improved. When error rates are within acceptable limits for the use case, testing is complete.

## Conclusion

Instrumentation and data collection are more than 80% of the cost of most machine learning projects. Reality AI software from Renesas has analytics that can help reduce the cost of both. Reality AI Tools identify the most cost-effective combinations of sensor channels, find the best sensor locations, and generate minimum component specifications. It can also help you manage the cost of data collection by finding instrumentation and data processing problems as data is gathered so you can get the most out of using synthetic augmentation techniques.

## Reference Material

[www.renesas.com/realityai](http://www.renesas.com/realityai)

[Real-time Analytics technology](#)

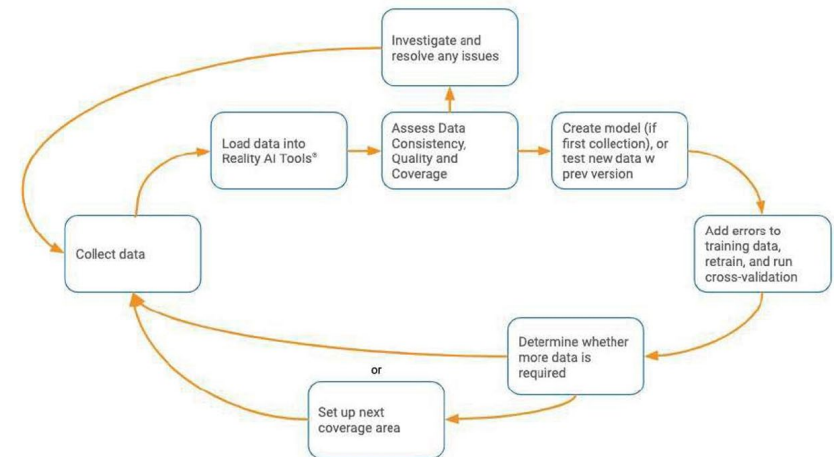


Figure 22. Diagram showing process for collection using the Iterative Data Coverage method in Reality AI Tools

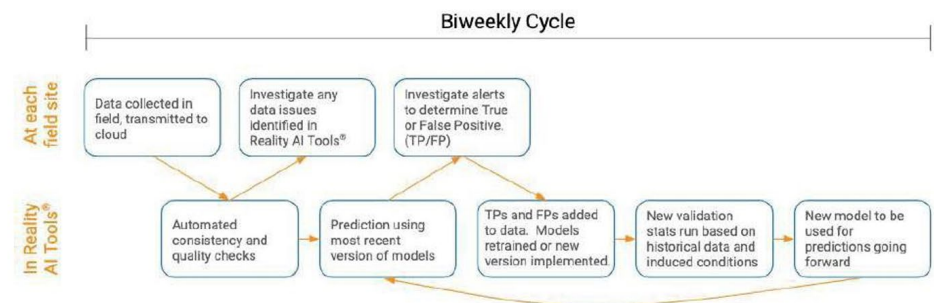


Figure 23. Diagram for a typical field testing process using Reality AI Tools



○

At Renesas we continuously strive to drive innovation with a comprehensive portfolio of microcontrollers, analog and power devices. Our mission is to develop a safer, healthier, greener, and smarter world by providing intelligence to our four focus growth segments: Automotive, Industrial, Infrastructure, and IoT that are all vital to our daily lives, meaning our products and solutions are embedded everywhere.

## STAY CONNECTED



[CLICK HERE TO VISIT OUR WEBSITE](#)