SAND2023-02687C

INSPECTA 1.0: IMPLEMENTATION CHALLENGES FOR ON-DEVICE SPEECH AND VISION TASKS

PROCEEDINGS OF THE INMM/ESARDA JOINT ANNUAL MEETING
MAY 22-26, 2023

Nathan Shoman*
Sandia National Laboratories†
nshoman@sandia.gov

Philip Honnold
Sandia National Laboratories†
phonnol@sandia.gov

Heidi Smartt
Sandia National Laboratories†
hasmart@sandia.gov

David Hannasch
Sandia National Laboratories†
dahanna@sandia.gov

ABSTRACT

Artificial intelligence (AI) and Machine Learning (ML) has become ubiquitous in our day-to-day lives, powering common conveniences such as smart home controls and entertainment suggestions. However, AI has not yet seen wide deployment for safeguards related tasks. Inspecta, the International Nuclear Safeguards Personal Examination and Containment Tracking Assistant, is being developed to reduce the burden of common safeguards tasks encountered during inspections. A key focus of Inspecta is to leverage existing AI/ML techniques to improve the inspection experience rather than developing new algorithms. The initial 1.0 version of the Inspecta Android application has the capability to perform on-device machine learning for several tasks including speech recognition, speech synthesis, and optical character recognition (OCR). This paper documents challenges and solutions for these tasks and how they are applied to improve seal verification.

1 Introduction

The IAEA Department of Safeguards is responsible for verifying international safeguards agreements. The mission is "to deter the spread of nuclear weapons by early detection of the misuse of nuclear material or technology. This provides credible assurances that States are honouring their legal obligations that nuclear material is being used only for peaceful purposes." [1] The implementation of safeguards is unique for individual states, as they are based on sovereign agreements between a State and the IAEA, as well as on a facility-to-facility basis as determined through a safeguards agreement's facility attachment. Safeguards activities are also based on state factor's and the IAEA's technical objectives defined in the Annual Implementation Plan.

Despite these variations, there are many common and repetitive activities performed by international nuclear safeguards (INS) inspectors regardless of the type of facility. This includes reviewing facility logs, physically inspecting and maintaining safeguards equipment, taking samples, verifying design information, item counting, and general observation of a site to identify anomalies. These inspection activities are often tedious and demanding. In addition, inspectors are commonly in hazardous environments clad in full personal protective equipment (PPE) with limited time for activities. These high-demand, high-stress environments create opportunities for human error.

There is also an upward trend in the number of responsibilities facing INS inspectors. This is a direct result of 1) an increase in the quantity and variety of nuclear facilities under safeguards (due to the development of novel fuel cycles);

*Corresponding author

†Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2023-XXXXX

2) an increase in the global number of significant quantities of special nuclear materials due to waste products and spent fuel; and 3) a push for inspectors to operate more as investigators rather than "auditors." These increased demands could make the INS inspector's job near-untenable.

Artificial Intelligence (AI) and its underlying algorithms are ubiquitous in everyday life, impacting the human experience both directly (automated driver-assistance, voice-assisted smart home controls, smart digital assistances (SDAs) like Amazon's Alexa[1], etc.) and indirectly (targeted ads from online vendors, search suggestions, etc.). Integrating these advanced capabilities with INS inspectors could increase their effectiveness and efficiency, particularly for tasks that are repetitive and prone to human error.

We are developing an AI-enabled SDA for use in the field. The International Nuclear Safeguards Personal Examination and Containment Tracking Assistant (nicknamed "Inspecta") is similar in function to Alexa or Siri[2] in performing tasks such as note-taking or timers, but also has safeguards-specific capabilities like using optical character recognition (OCR) to read seal numbers. An Inspecta prototype was completed and demonstrated in December 2022, illustrating how AI could aid inspectors. Inspecta is intended to support inspectors, rather than replace humans in the field.

In this paper, we share the high-impact task chosen for demonstration of the Inspecta 1.0 prototype, the architecture and technical capabilities ("skills") required to perform this task, and describe details, challenges, and results of the demonstration performed in December 2022.

2 Inspecta Scope Definition

In 2021, we started by defining the goals and constraints Inspecta might have, using insight from former IAEA inspectors, IAEA publications [2–5], and comprehensive lists of IAEA safeguards inspection tasks to identify high-impact tasks that commonly face current INS inspectors. The most common and high-value tasks were (1) surveillance review, (2) transcription, (3) information integration, (4) seals examination and verification, and (5) spent fuel verification. Transcription and information integration are often linked to other tasks, so we focused on surveillance review, seals, and spent fuel verification. Spent fuel verification and surveillance review are currently being explored by other R&D programs. Therefore, the Inspecta 1.0 prototype was focused on seal examination and general SDA functionality.

We reviewed how seal examination is currently performed to determine specifically how Inspecta might aid an inspector. This task is important but tedious for inspectors. The IAEA currently employs metal cup seals (though new types of seals are being developed/deployed) with a numeric identifier attached to labeled containers after the contents have been measured or verified, as shown by Figure 1. An inspector will be escorted by facility personnel to the material holding location, find seals to examine, compare the seal number and the container label against a paper list, and mark that the seal has been examined and confirmed. The inspector also inspects the seal and seal wire for signs of tampering, pulling to ensure proper connection to the container. A small set of seals may be replaced and returned to IAEA headquarters for additional verification.

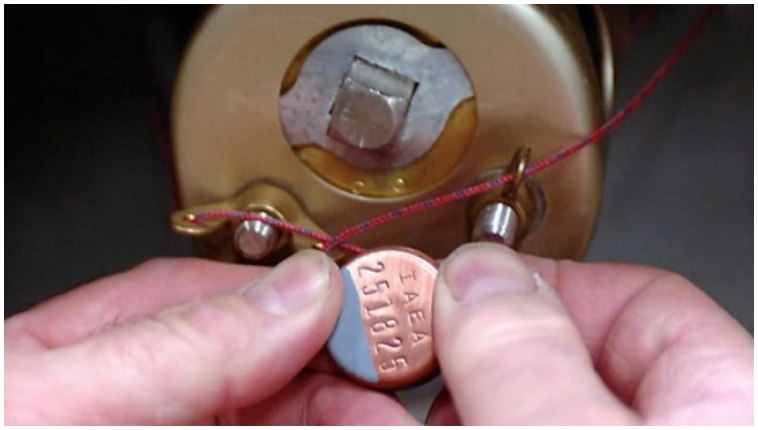

Figure 1: Cap Seals used by IAEA (https://www.iaea.org/newscenter/news/safeguarding-the-future-iaea-looks-for-improved-solutions-for-passive-loop-seals-for-nuclear-verification)

[1] https://developer.amazon.com/en-US/alexa
[2] https://developer.apple.com/siri/

We considered how Inspecta might support an inspector in this task. At one extreme, Inspecta could conceptually fully automate the seal examination task. This fully-automated concept of Inspecta could use robotics and indoor navigation to locate seals in an area, use a robotic manipulator arm and object detection to find and grasp a seal, apply optical character recognition (OCR) to acquire a seal number, employ anomaly detection to identify signs of tamper, and communicate with the inspector using speech synthesis/recognition. A full list of capabilities required is shown in Figure 2.

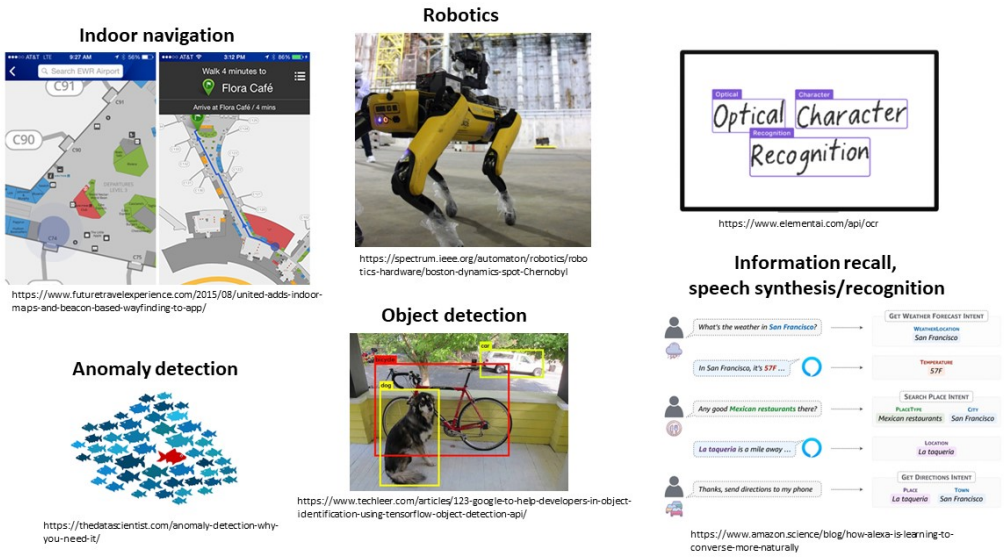

Figure 2: List of necessary capabilities for fully-automated seal examination

The Inspecta 1.0 scope was significantly narrowed from this fully automated case, focusing on skills that could directly aid an inspector while still requiring their full involvement. Specifically, we honed in on OCR to semi-automate the review of individual seals, and speech synthesis and speech recognition for note-taking and communicating with Inspecta. Speech recognition/synthesis was also applicable to the general SDA functionality of Inspecta.

Constraints and requirements were then identified, based on the projected end-use of Inspecta. These were mainly centered on security, usability, and deployment of Inspecta. In terms of security, we posited that all data must remain on-device due to safeguards agreements, which meant all machine learning should be performed on-device. In terms of usability, Inspecta needed to be approachable for new users, have multiple methods of interaction (e.g. by voice and through a screen), and be carryable/usable for a full day. Finally, Inspecta was designed with maximum flexibility in mind, being deployable to multiple platforms (e.g. Android, iOS, Windows, etc.). The actual architecture decisions are discussed through the rest of this paper.

Machine learning (ML) models were developed in Python using PyTorch[3] frameworks. However, ML training and use can be computationally demanding; cloud services can off-load the computational burden, but this option was quickly discarded given the security and data privacy constraints of the project. Instead, the Open Neural Network Exchange (ONNX) application program interface (API) was used to convert ML models built in PyTorch. This provided both a general framework for ML use for cross-platform functionality as well as helped quantize [6] and optimize models for mobile devices.

3 Inspecta 1.0 Architecture

The Inspecta 1.0 prototype was developed on the Xamarin platform. Xamarin is an open-source cross-platform code based on .NET and C#. This code environment supports deployment to multiple operating systems (Android, iOS, Windows, etc.). While there might be platform-specific code that would be required, Xamarin enables abstraction of the UI and other components.

[3] `https://pytorch.org/`

Ultimately, Inspecta 1.0 was installed on an Android Pixel 6 for testing and demonstration. The Android platform was selected due to the ease of deployment and widespread adoption of the system; there are roughly 2.3 billion Android devices in operation today. The Pixel 6 was selected as the prototype device based on the high-resolution display, powerful processor, on-device microphone and speakers, a small form factor, and long battery life. Though Android has been the specific target platform, the work done in Xamarin is transferable to other platforms.

The development of the Inspecta application itself was split into multiple modular pieces. The UI development was split off from the skills development (e.g. speech recognition, OCR) to facility quicker testing of the ML-based portions. The base Inspecta application was formed by the Grial UI kit[4], which provided a library of common elements for the construction of the app itself. This simplified the UI development, resulting in a professional-looking application without needing to hand-craft elements in Xamarin.

Three skills were necessary for the Inspecta 1.0 prototype: speech synthesis, speech recognition, and OCR. Speech synthesis uses the offline Android services. The native Android speech synthesis is robust and had easy implementation. This module could be improved in the future, if needed, by utilizing ML solutions such as Tacotron [7] and WaveGlow [8].

Speech recognition had two primary applications for Inspecta: note-taking and understanding commands. Both applications involve speech-to-text, which was performed using a PyTorch model of Wav2Vec2 [9]. This model was quantized and exported to ONNX to optimize performance. Speech-to-text requires multiple steps within Inspecta.

First, Inspecta connects to the on-device microphone to record an audio clip. This waveform is converted to short int (the datatype required for the ML algorithm) and stored to the device. The recorded audio clip is then processed using the ONNX version of Wav2Vec2 to produce a predicted text string. For note-taking, this text string is then saved to the device, for the user to review, edit, or read at a later date.

For commands, the text string is compared to the list of possible commands using the Levenshtein distance [10]. Based on the similarity between the pre-loaded commands and the predicted text, Inspecta will execute a command, suggest a command, or return a message stating that it doesn't understand. Inspecta can successfully perform the commands "Start seal examination task," "Pause seal examination," "Read my notes," "Take a note," and appropriately respond to the query "What time is it?"

The OCR module utilizes the on-device camera and ML models to read the unique alphanumeric identifiers on a metal cap seal. OCR is split into two stages: text detection and text recognition. Text detection is processing of an input image to determine the exact location of characters and create a bounding box. Text recognition involves using a prediction algorithm to correctly identify and classify the text within the bounding box. Some ML models perform both steps in a single pipeline, but the Inspecta team chose to utilize a two-stage process. This was to feed results from the text detection module directly back to the user in real-time by overlaying bounding boxes on the phone display.

Different models are used for each stage. Text detection was performed using a PyTorch Character-Region Awareness for Text Detection (CRAFT)[5] model. CRAFT is especially effective for identifying individual characters, outputting two scores: the region score and affinity score. The region score corresponds to the probability of each pixel being the center of a character, which helps localize individual characters. The affinity score relates to the likelihood that individual characters should be grouped together. Both are useful for cap seals, where individual characters are linked together on a row to form a unique identifier. A pre-trained PyTorch model was quantized and converted to ONNX.

Text recognition was accomplished using a pre-trained model based on Resnet from JaidedAI[6]. From the bounding boxes, features of the individual characters are used to generate a predicted text string. This can require some image processing, which was performed using the OpenCV[7] library. The predicted text string is compared against a seal list that was pre-loaded into Inspecta, mirroring how an inspector might have a list of seals.

OCR is performed in real-time. The on-device camera streams video to the display, such that the user is able to direct it at seals while Inspecta identifies and reads the characters. Bounding boxes, the predicted text, and confidence score (from the text recognition algorithm) are shown on-screen to the user.

Performing OCR in real-time results in a slowed framerate of the on-device camera. The accuracy of the algorithm and resolution of the input image can both negatively impact the framerate shown the user, while higher resolution means the OCR is able to correctly function at a variable distance from the seal. High accuracy is a requirement for Inspecta, as misreading seals could have adverse consequences in safeguards. Simultaneously, a too-slow framerate could make

[4] https://grialkit.com/
[5] https://github.com/clovaai/CRAFT-pytorch
[6] https://github.com/JaidedAI/EasyOCR
[7] https://opencv.org/

directing the Inspecta device frustrating for a user. The camera resolution and framerate were optimized to 10cm of distance between the camera and the seal; the user is subtly prompted to use this distance by a narrowed field-of-vision in the camera preview.

Multiple events are triggered when the predicted text matches the pre-loaded seal list. Inspecta first captures and stores an image of the seal for later review. Inspecta then provides visual feedback to the user, flashing green and displaying a message that an item was checked off the list. This is shown by Figures 5 and 6.

4 Demonstration

A live demonstration of the Inspecta 1.0 prototype was performed in December 2022. This demonstration involved recording a note, reading a note, and performing a simplified seal examination while interacting primarily through voice. This list of tasks tested the speech recognition, speech synthesis, and OCR algorithms as well as testing the general usability of the app.

Inspecta was installed on the Android Pixel 6 device, and the test user was given approximately 30 minutes of training to gain familiarity with the app. The user launched into note-taking, where some notes were pre-loaded into the app for illustrative purposes. Notes can be entered manually (e.g. using the on-screen keyboard) or by speech-to-text. After a note is created, the user can attach tags (e.g. topic, facility name) to support easy filtering and review. Timestamps are automatically assigned to saved notes.

The test user initiated a new note by using the voice command “Take a note.” When Inspecta is actively listening (either for a command or to take a note), a drop-down appears on the phone display as shown in Figure 3. For the demonstration, the user employed the speech-to-text to author the new note, as shown by the “2022 December 09 11:45 AM” note in Figure 4.

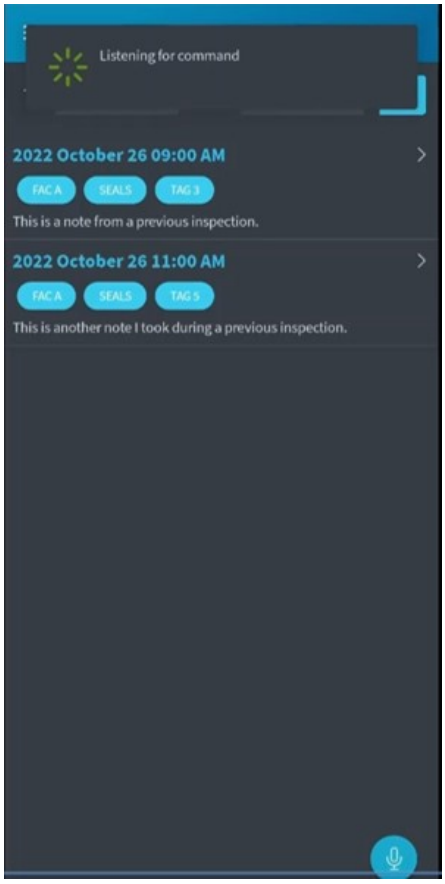

Figure 3: Inspecta Listening

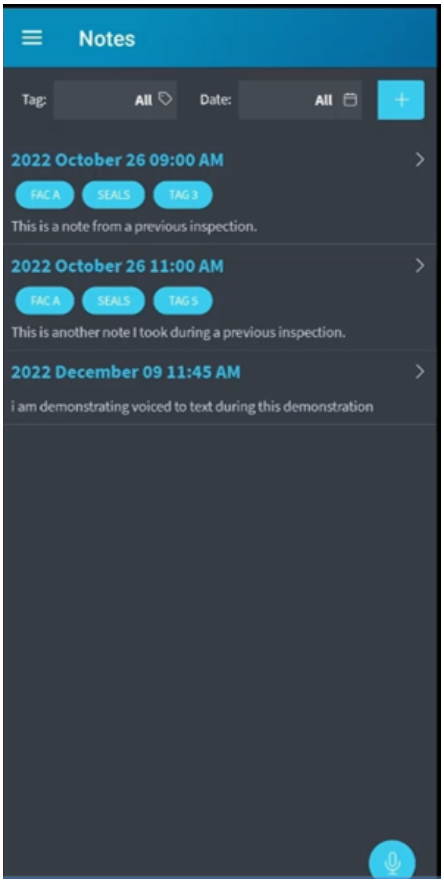

Figure 4: New Note Taken

The speech-to-text performed as intended. Though a small transcription error occurred (“voiced” instead of “voice” in Figure 4), Inspecta directly launched the note-taking skill, recorded the note, and saved it to the Notes list. The user could then select that note to perform edits or attach tags for future reference.

The user then tested the speech synthesis by the command “Read my notes.” Inspecta promptly read all notes, starting with the oldest and including the timestamps.

Finally, the user performed the seal examination task. A list of seal numbers had been previously loaded into Inspecta to emulate what might be reasonable in the field. The user initiated this by the verbal command “Inspecta, start seal examination.” Inspecta shifted to portrait orientation and loaded the camera preview. The narrowed field-of-vision guided the user to a consistent and optimal distance between the device and the target seal.

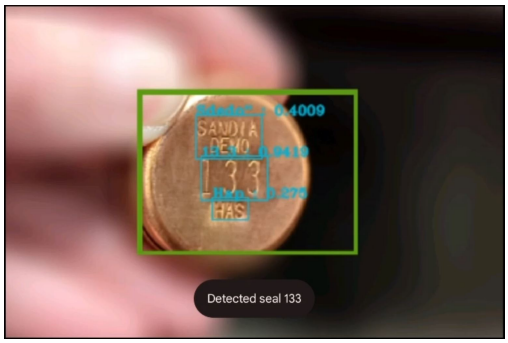

Figure 5: Seal 133 Confirmed by Inspecta

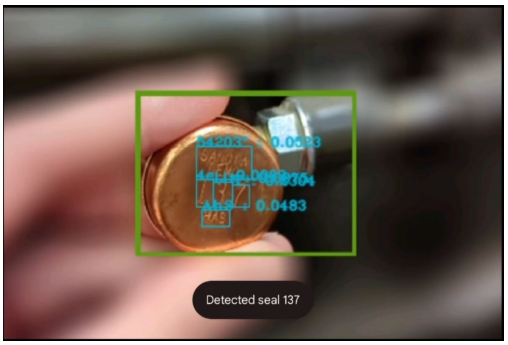

Figure 6: Seal 137 Confirmed by Inspecta

The user used one hand to hold the seal, tugging gently to confirm proper wire attachment to the sealed item. The other hand held the device with Inspecta. As described previously, bounding boxes generated by the CRAFT model are shown with predicted text and the confidence interval. This active feedback helped guide the user in adjusting the orientation and distance between the device and seal when comparing the seal identification against the pre-loaded list. Positive matches are shown in Figures 5 and 6.

Some challenges arose in OCR for Inspecta during the demonstration. First, the OCR pipeline is set to only process images in a single orientation (that is, the OCR algorithm will not identify rotated characters). This forced the user to ensure the device and seal were at the same orientation. For seals that are connected by an inflexible wire, this can be an inconvenience. Second, the identifiers on the cap seals are stamped metal, meaning there is little contrast between characters and background. This, combined with glare from room lighting, caused Inspecta some pause in correctly identifying characters. However, these challenges only slightly slowed the user from completing their task; all seals in the area were examined in less than 30 seconds.

Inspecta successfully identified all seals that were in the room, and did not give any false positives. After performing the seal examination task, the progress menu in Inspecta shows 4 seals as "Examined," 2 as "Flagged for Follow-up." The remaining seal (142) was removed from the final list. This was intentional (only 4 seals were in the area) to show progress tracking for an inspector in the field. The before and after images are shown by Figures 7 and 8.

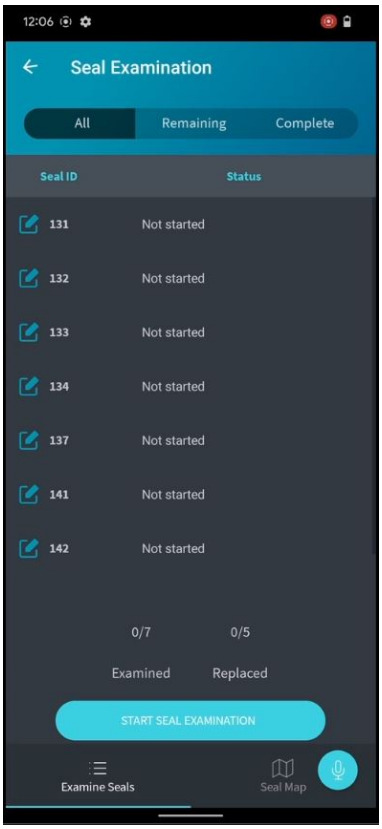

Figure 7: Before seal examination

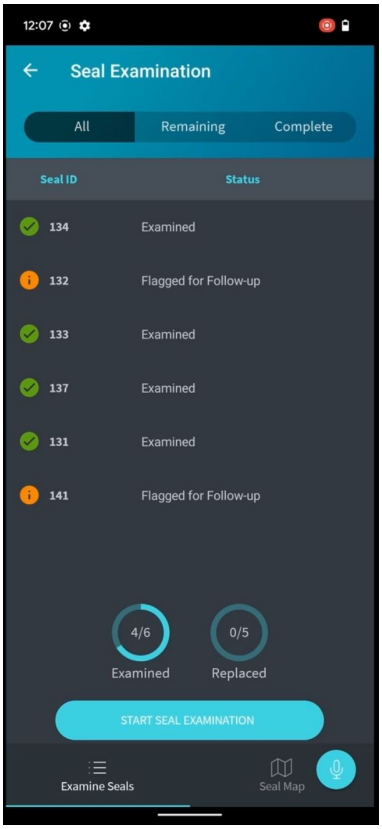

Figure 8: After seal examination

5 Conclusion and Future Work

Future work for Inspecta takes two forms: robotic support and the continued development of Inspecta skills. In September 2022, the Inspecta team procured a Boston Dynamics *Spot*[8], to explore a physical implementation of Inspecta. Spot (and other robotic support systems) can support inspectors by more actively performing tedious or hazardous tasks, reducing radiation exposure to inspectors. Spot is especially well-suited, being equipped with multiple cameras, a robotic manipulator arm, and the ability to mount an NVIDIA Jetson[9] for ML-focused tasks. Work with Spot is currently focused on object detection, training Spot to identify and navigate to specific items. Spot could become a semi-autonomous auditing "partner," enabling the inspector to focus on more investigative roles.

The Inspecta 1.0 prototype demonstrated how an SDA could enhance the effectiveness of an inspector in seal examination. That being said, there are many opportunities for expansion. Two capabilities being actively explored are wake word and information recall.

Currently, when the user wants to interact with Inspecta, they use a "press-and-hold" functionality. This results in a semi-hands-free interaction between Inspecta and the user. Using a wake word to trigger active listening would enable a fully hands-free interaction between Inspecta and the user. A wake word would be a unique command for triggering active listening (e.g. speech-to-text). This is challenging; configuring Inspecta to be passively listening and respond only to a specific command is an added layer of complexity beyond simple speech recognition. However, this would greatly increase the usability of Inspecta for an end-user.

Inspectors often face a challenge of insufficient information in the field. Site-specific technical specifications or IAEA documentation may be available, but finding specific information within these large documents can be challenging. Information recall would enable Inspecta to ingest these large documents and respond to specific queries from the user regarding the contents. This skill will require some finesse to complete, to ensure only useful information is provided to the user.

[8] https://www.bostondynamics.com/products/spot
[9] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/

6 Acknowledgements

The authors would like to acknowledge and thank the U.S. National Nuclear Security Administration (NNSA) Office of Defense Nuclear Nonproliferation R&D Safeguards portfolio for funding and supporting this research.

References

[1] International Atomic Energy Agency, "Basics of IAEA Safeguards," Jan 2021.

[2] IAEA Safeguards, "Development and implementation support programme for nuclear verification 2020-2021," Jan 2020.

[3] IAEA Safeguards, "Research and development plan: Enhancing capabilities for nuclear verification," Jan 2018.

[4] IAEA Safeguards, "Emerging technologies workshop: Trends and implications for safeguards workshop report," Feb 2017.

[5] IAEA Safeguards, "Emerging technologies workshop: Insights and actionable ideas for key safeguards challenges workshop report," Jan 2020.

[6] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," 2021.

[7] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," 2017.

[8] Ryan Prenger, Rafael Valle, and Bryan Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," 2018.

[9] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[10] Levenshtein, Vladamir I., "Binary codes capable of correcting deletions, insertions, and reversals.," *Soviet Physics Doklady*, Feb 1996.