Aware-ables

Chester Glenn, Kevin Xie, Jong Woo Ha
Carnegie Mellon University, CIT, ECE

Introduction & Abstract

Historically, braille literacy in the United States has been on a sharp decline, and fewer than 10 percent of the legally blind Americans are braille literate [1]. Therefore, the vast majority of the visually impaired individuals can not fluently read braille text, embossed on paper or other surfaces, meant to provide pivotal guidance and assistance. Braille is an exclusively tactile system of embossed or raised dots that allow individuals with impaired vision to have similar access to inscribed forms of information. However, given how over 90 percent of legally blind Americans are braille illiterate, a means to mitigate this situation is imperative for guaranteeing adequate education to support a strong career, as well as navigational assistance. Aware-ables is a solution that aims to provide those with impaired vision with the ability to read braille regardless of their education. The proposed solution will use a bird's eye view camera on top of the wooden apparatus to capture an A4-sized braille document, then translate and read the contents back to the user. It is our hope that this solution will alleviate some of the current restrictions experienced by legally blind individuals.

Use Case Requirements

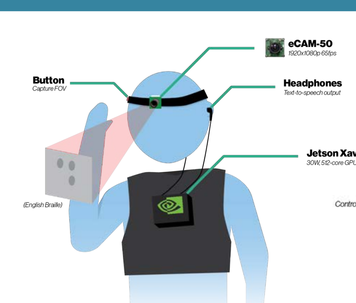


Figure 1. Initial Design

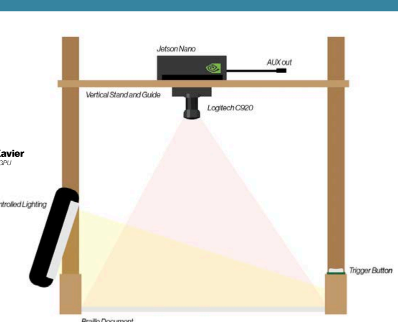


Figure 2. Final Design

Requirement	Target (units)	Justification
Text-to-Speech latency	2s	Common usability standard for loading wait times
Words per Frame	>10	~300wpm to match braille reading speed *NOTE: 150wpm is a comfortable speaking speed
Character Error Rate	10%	Matches error rate of traditional OCR
Word Error Rate	<10%	Errors should be corrected from spellcheck

Figure 3. Use Case Requirements

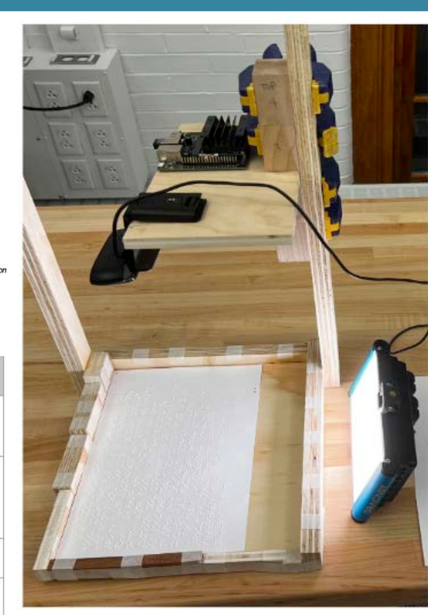


Figure 4. Final Product

System Architecture & Block Diagram

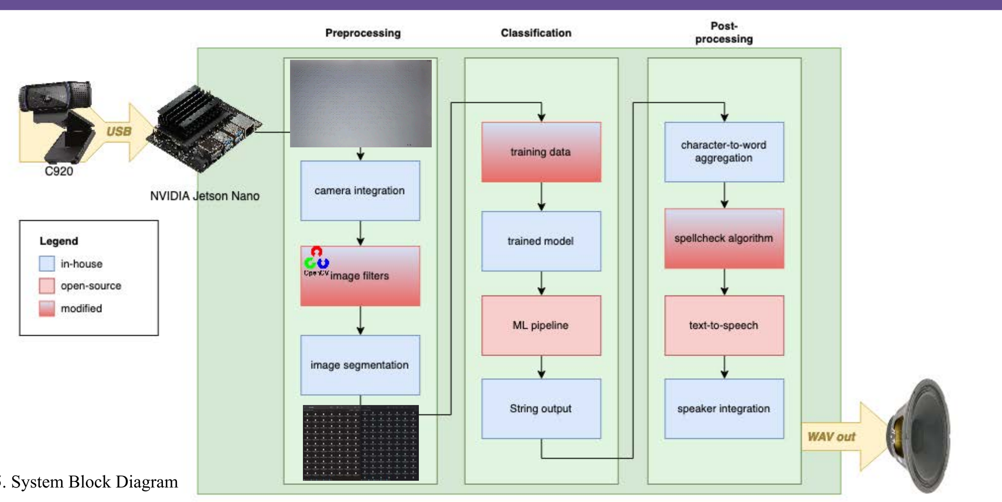


Figure 5. System Block Diagram

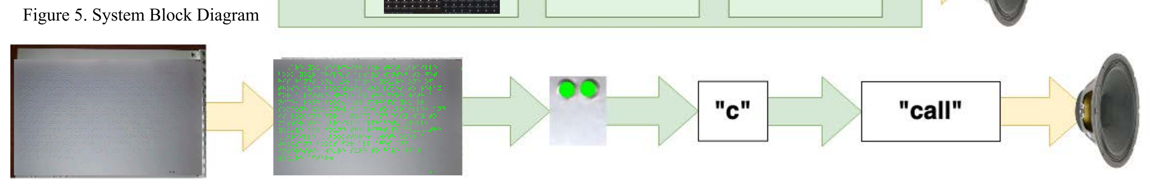


Figure 6. Data path for our proposed solution

Fig. 5 presents a high-level block diagram for our intended implementation. Our software stack is split into three successive subsystems: pre-processing, classification, and post-processing. Later sections will dive into more detail about implementation specifics, however it is important to note the color coding of the blocks indicating which software components will be sourced off-the-shelf and which will be developed in-house.

Below the block diagram, in Fig. 6, we have provided a high-level visualization of modifications being made to the input image at significant points in our data path, however, here again later figures will provide more detail. From the high-level diagram, it is clear that our software will expect an (1) uncropped, well lit image of a braille document, which will then be (2) cropped, filtered, and segmented into single braille characters, then (3) classified, and (4) concatenated into an English word, which can then be (5) read out via the speaker.

System Description

Software sub-system 1. Pre - processing

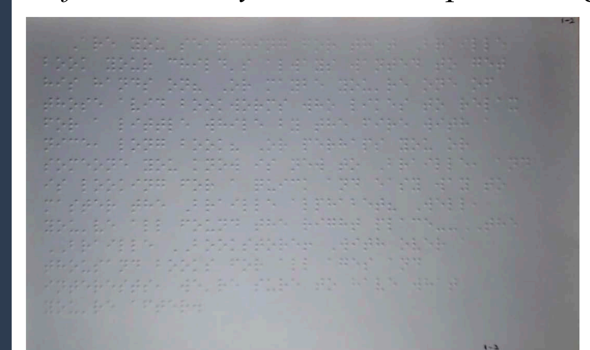


Figure 7. Original braille image capture upon button activation

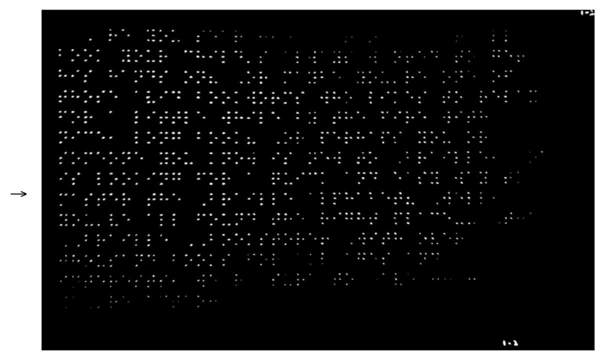


Figure 8. Image after gray scaling, adaptive thresholding

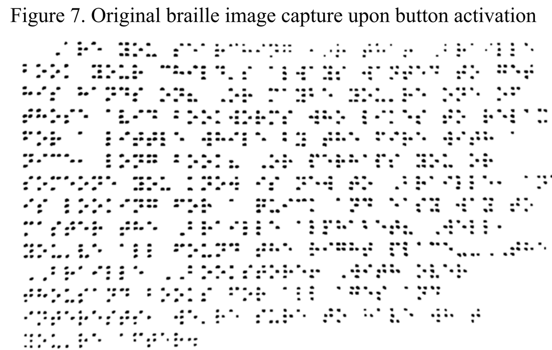


Figure 10. Image after erosion and dilation.

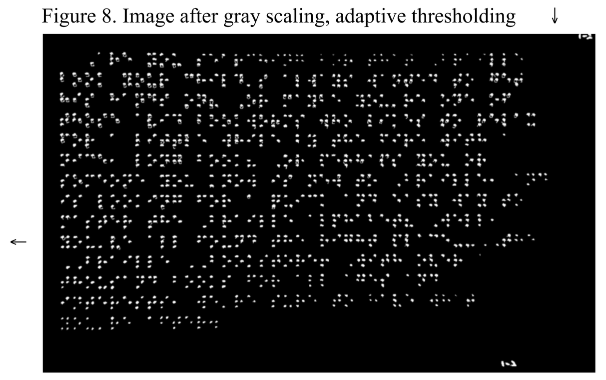


Figure 9. Image after application of MedianBlur, GaussianBlur, canny edge detection filter merged with blurred threshold

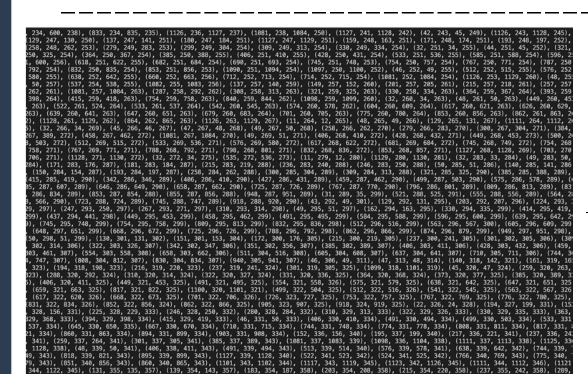


Figure 11. Filtered stats (left, top, width, height) matrix from the openCV's ConnectedComponentsWithStats() function.

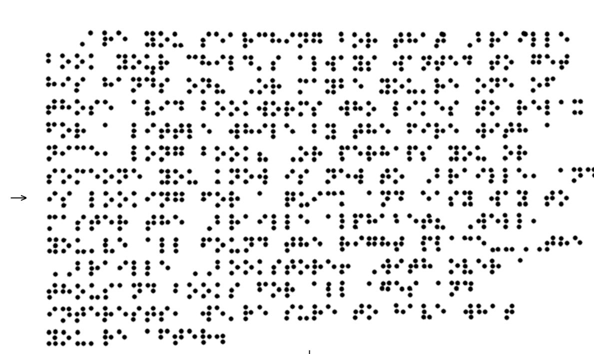


Figure 12. Image after drawing circles using filtered stats on top of the pre-processed image

Figure 14. Resulting folder containing individually cropped braille

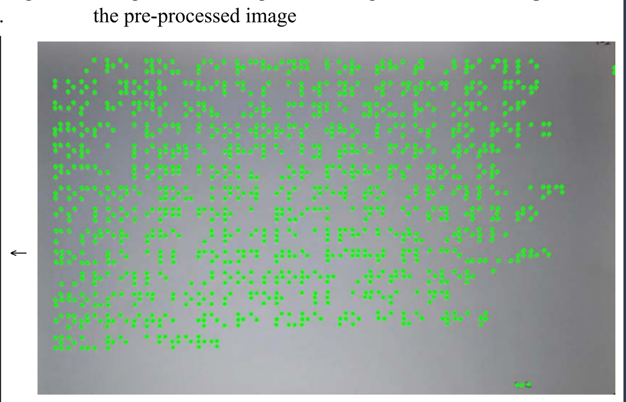


Figure 13. Color contrast for the higher classification accuracy

Software sub-system 2. Classification

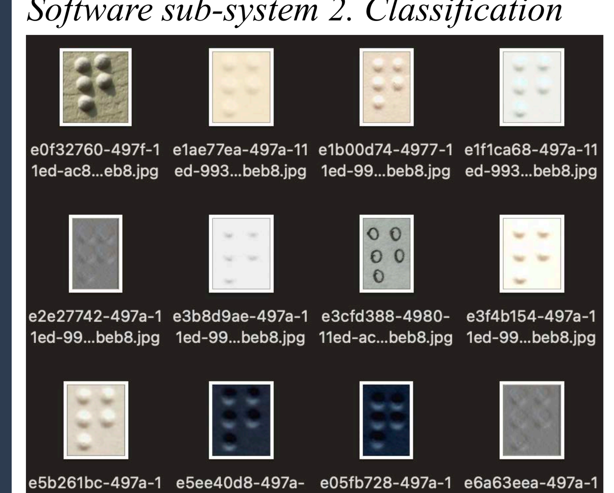


Figure 15. 20,000 image dataset
(Source : https://github.com/HelenGezahegn/aeye-alliance/tree/master/Labelled%20Data)

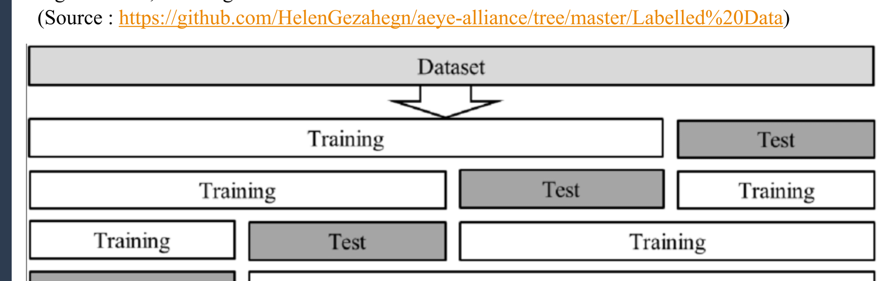


Figure 17. 4-fold cross-validation technique
(Source : https://www.researchgate.net/publication/328461242_Empirische_Evaluationsmethoden_Band_22)

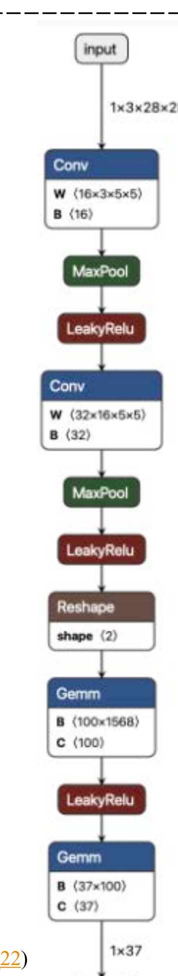


Figure 18.

System Description

Software sub-system 2. Classification, continued

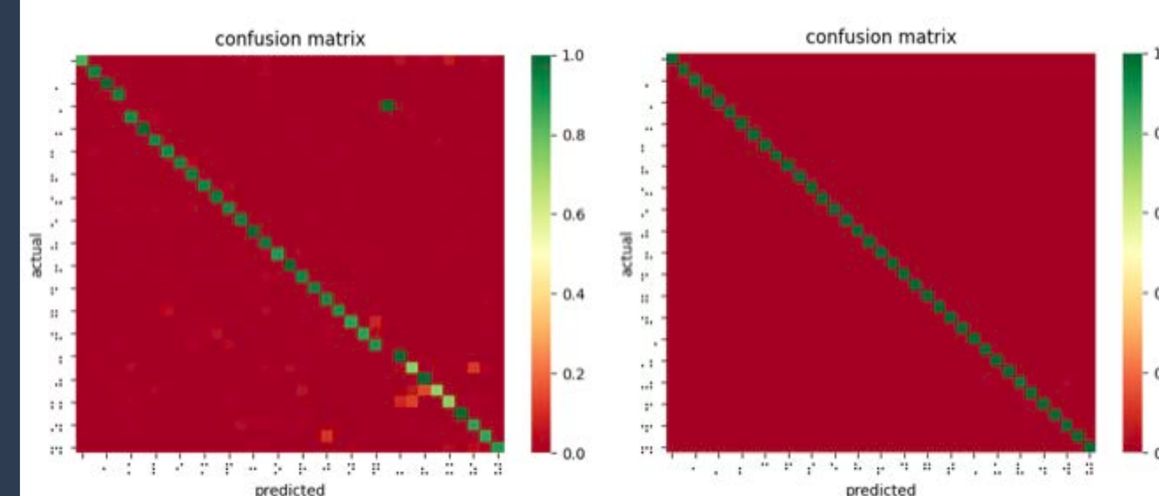
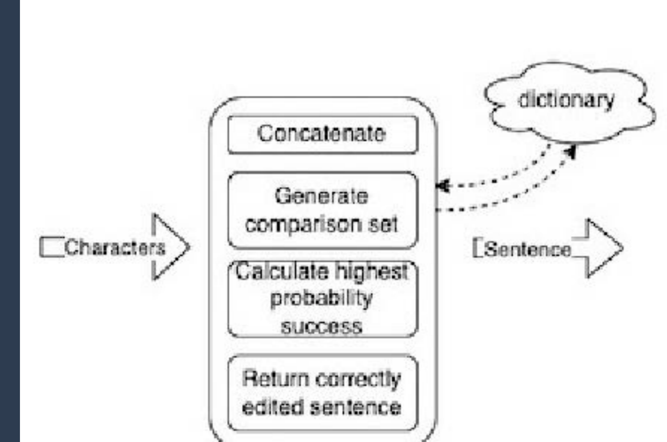


Figure 19. Abbreviated confusion matrix for a pre-trained classification model (on the left) and In-house ResNet-18 model (on the right). Green diagonal line indicates high Confidence tested against training dataset. Pre-trained model exhibits around 89% classification Accuracy whereas in-house ResNet-18 model exhibits around 99% classification accuracy

Software Sub-System 3. Post - processing



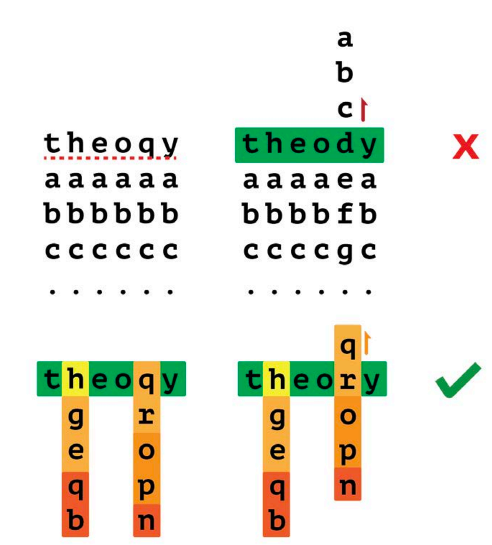


Figure 20. Subsystem C error checking flow chart (on the left)
In order to create a working spellcheck model, several key factors have to be considered. The first Indication is whether or not a word is in need of error checking after pre-processing and Classification. As stated in the classification deliverable, a confidence matrix is provided to the Post-processing section in the form of a dictionary that provides the indices of characters below the 90% confidence threshold we determined, and their 5 character array of possible options.

Figure 21. Normal Bayesian model (on the right top) vs Added confidence matrix (on the right bottom)
Right bottom - confidence matrix being applied to correct characters 'h' and 'q'
The main algorithmic approach to the spellchecking subsection involves generating the set of all Possible single character different "words" as specified by the confidence matrix, that are also Contained in the static dictionary checker provided. The second key factor in development is the "Dictionary" checker of choice. After each word has been processed and the text is completely Synthesized, the data will be sent as a request to the Google REST API, and a respective .WAV file Will be processed and sent through the speaker or headphones connected as stereo output.

Experimental Software Features: Dynamic ML Crop, Cursor Reading

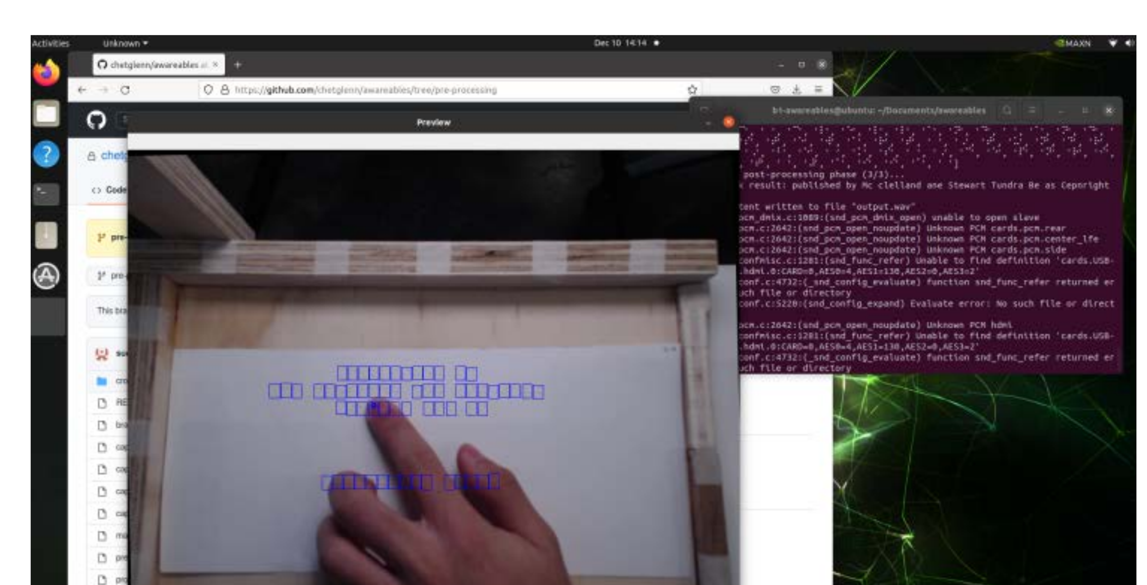


Figure 21. Combining the bounding boxes that can be extracted from AngelinaReader and Google's MediaPipe Hand pose estimation model, users can further extend their braille learning experience as moving Their fingers over individual braille characters will give an audio feedback of which character it is.

System Evaluation

1. Pre - processing

Model	Performance (Character recognition accuracy)	Latency
original capture w/o ml crop	89%	< 0.5 s
nms w/o ml cropp	98 %	< 0.5 s
nms w/ ml crop	98%	~ 5s

Figure 22. Performance and latency table for various design choices models

2. Classification

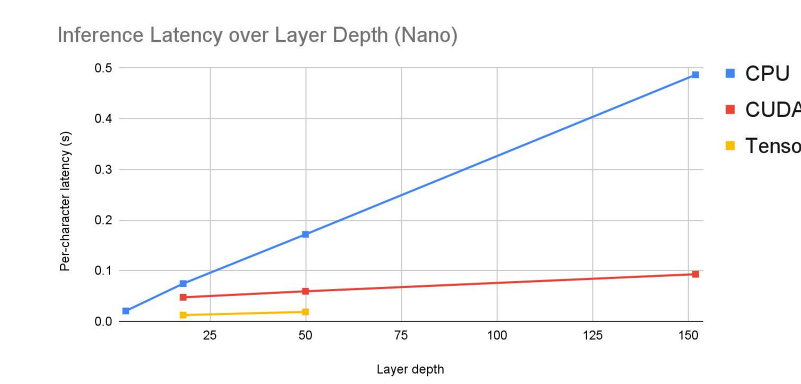


Figure 23. Inference Latency over Latency Depth tested under Jetson Nano

3. Post - processing

Model	Word Error Rate	Latency
Static Dictionary	15%	1s
Bayesian Model	8%	0.02s
Bayesian + Dictionary	5%	1s
Confidence Matrix	3%	0.5s

Figure 24. (On the left), Word Error Rate and Latency For various post-processing models

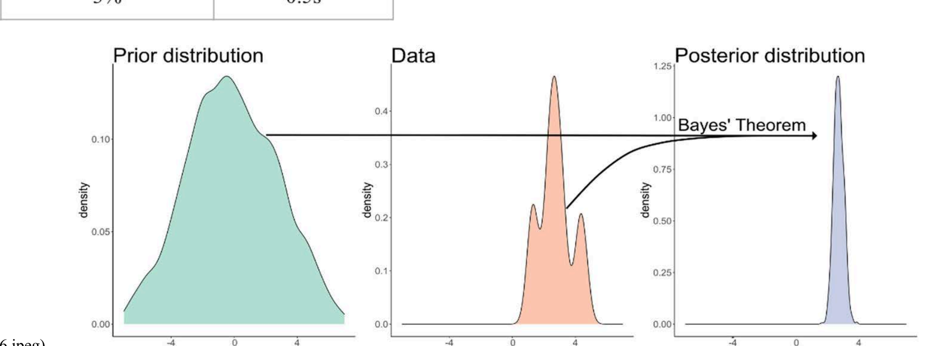
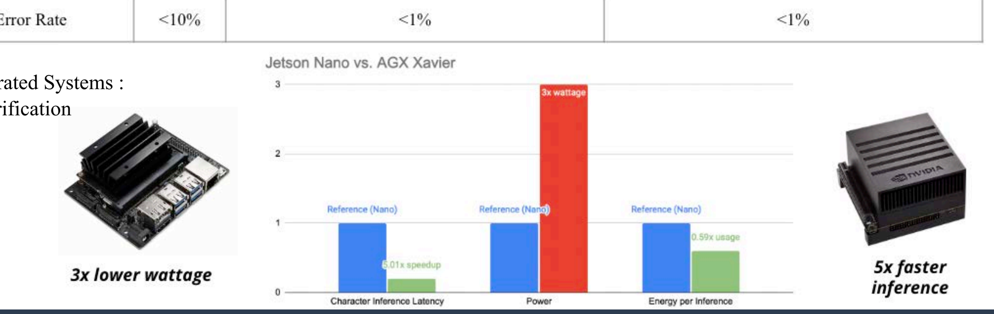


Figure 24. (On the right), (Source : https://pubs.acs.org/cms/10.1021/acs.jproteome.1c00859/asset/images/large/pr1c00859_0006.jpeg)

4. Integrated System

Requirement	Target	Actual (Nano)	Actual (Xavier)
Text-to-Speech latency	2s	**	~2.3s **
Words per Frame	>10	~8 (based on 0.5 second classification latency ceiling)	~40
Character Error Rate	10%	0.14%	0.14%
Word Error Rate	<10%	<1%	<1%

Figure 25. Integrated Systems : Validation & verification



Conclusion & Acknowledgements

We are excited to present Aware-ables, an educational and functional appliance for reading and learning braille via text-to-speech. The completed solution uses OpenCV image preprocessing, machine learning image classification, and a custom spell check to deliver reliable Braille reading to its users. Looking toward the future, we hope this system can be adapted as a wearable device to not only personalize education but enable vision impaired users to learn braille while navigating the world. We would like to acknowledge Aeye Alliance, AngelinaReader, Google, AWS, LAMP, and Capstone staffs for their resources in guiding us toward a working solution.

[1]. Jernigan Institute. "The Braille Literacy Crisis in America." In: *A Report to the Nation by the National Federation of the Blind* (Mar.2009).