Clinical Training Simulation for Suturing: A Feedback System Incorporating Machine Learning to Assess Adequate Knot Tightness

BME 402

Authors: Michels M, Hiller K, Rowe S, Hockerman L, Hansen P

Abstract

Proper suture tension is critical for achieving wound closure integrity and preventing complications such as wound dehiscence. However, current veterinary training methods rely on subjective evaluation and repeated practice, with no qualitative system to teach students how to achieve optimal suture strength. To address this gap, a visual knot characteristic system that incorporates a machine learning model was developed to assess and identify adequate knot tightness. A dataset of 200 images of square knots (tight and loose) was collected and used to train a ResNet50 convolutional neural network. Multiple preprocessing techniques, including binarization and contrast enhancement, were evaluated using K-fold Cross-Validation (K=6). The unedited image set yielded the best performance and was used to train our final model that achieved 95% accuracy. The trained model was implemented onto a Raspberry Pi-based system including a camera and LED feedback. To improve the full training system's identification of loose knots, a 85% confidence threshold was implemented, resulting in overall performance of 82.5% accuracy. The system achieved a mean latency of 3.93 seconds. This work demonstrates the feasibility of implementing a machine learning model into a Raspberry Pi to create an accessible, low-cost training tool that provides objective, real-time feedback on suture technique. The training system has the potential to improve training efficacy and ultimately contribute to better clinical outcomes.

Table of Contents

Abstract....2
Introduction....4
Methods....5
    Dataset Assembly....5
    Model Training....6
    Workflow Implementation....8
    System Testing....10
Results....10
Discussion....13
    Deployment and System Reliability....13
    Impact of Image Preprocessing and Cross-Validation....13
    Dataset and Imaging Constraints....13
    Embedded System Feasibility....14
Conclusion....14
    Key Insights....14
    Project Impact....14
References....16

Introduction

A properly secured knot is critical for safely closing a wound and preventing healing complications. The security of a knot depends on achieving appropriate tension during the final throw, which ensures sufficient plastic deformation of the suture material. However, students often struggle with either tying the knot too tightly and causing material failure, or too loosely, allowing the knot to unravel after they release the suture material, as shown in Figure 1.

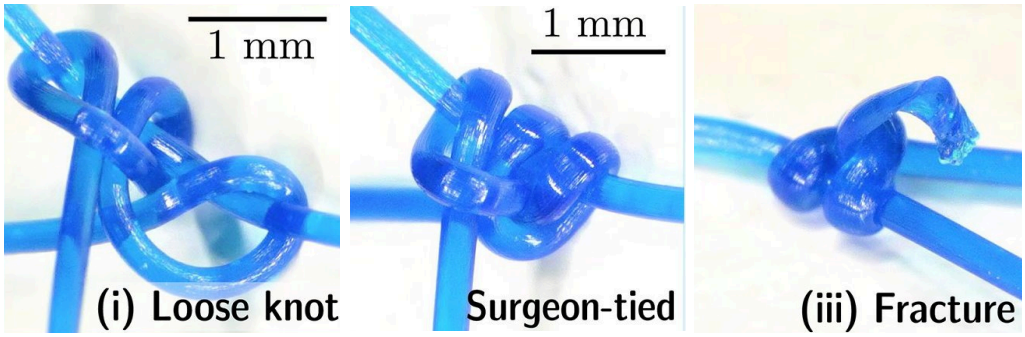

Figure 1: Examples of suture knots tied under different tension conditions, showing a loose knot (left), a correctly tied surgeon's knot (middle), and a fractured knot (right) resulting from excessive tension [1].

Improper suture tension can have significant clinical consequences. Insufficient tension can lead to knot slippage or wound dehiscence [2]. Wound dehiscence can lead to an average of $40,323 in excess hospitalization charges per year [3].

There is currently no device on the market or in clinical skills laboratories that allows students to learn and assess proper suture knot tension. Existing training tools either lack real-time feedback, are not commercially available, or only measure tension in the first knot throw, leaving a gap in effective suture training devices. The current method involves students learning the proper knot tension through "feel" and expensive practice. Instructors typically provide feedback through visual observation of the knot after it is tied. This approach lacks objectivity and consistency, which can potentially lead to prolonged training and increases the likelihood for failed sutures in clinical practices.

To address this gap, a simulation-based training tool was developed that incorporates a machine learning model. This training system has the potential to ultimately increase both training efficacy and surgical outcomes by providing real-time feedback of an adequate versus inadequate classification of the final throw of a square knot. The design aims to classify a commonly used black 2-0 nylon monofilament suture with an accuracy, precision, and recall all above 80%, as well as a F1-score above 0.8. The fully integrated training system should be able to withstand repetitive use in a clinical training setting, minimize disruption to the natural suturing process and provide feedback to the user within 5 seconds.

Methods

Dataset Assembly

Prior to training a ResNet50 convolutional neural network (CNN), a dataset of suture knots was created. The dataset consisted of two classes: tight (adequate) and loose (inadequate) knots. To ensure consistency across knots, black 2-0 monofilament sutures were tied on a light-colored silicone skin pad. Each knot consisted of a square knot with four throws, in which the final throw was either tightened or left loose to meet class criteria. Interrupted sutures were placed at 0.5-inch intervals, with 100 samples generated per class.

Images of each individual suture were captured under diffuse daylight from a consistent viewpoint, approximately 45° above the horizontal plane, with fixed-focus using an Arducam IMX477 Pi HQ Camera. This camera was subsequently integrated into the full training workflow to maintain consistency between data collection and deployment. Images were saved as jpegs and transferred from the Raspberry Pi to a computer for processing prior to training.

Because the original images included large portions of the surrounding skin pad, center-cropping was performed to remove irrelevant background information, including skin pad edges and adjacent knots. Each image was cropped to 500 x 500 pixels, with the top throw of the knot positioned at the center of each frame. Consistent framing across all images increased the likelihood that relevant knot features were the focus during model training, validation, and testing. Following image acquisition, each image was officially classified as either tight or loose and then validated.

In an attempt to maximize the utility of the relatively limited 200-image dataset, multiple preprocessing techniques were applied to copies of the full dataset to compare their impact on model performance using K-fold Cross-Validation, prior to selection of the final dataset used for training the CNN.

Notably, images were binarized, as shown in Figure 2A. Binarization converts a colored image into a black-and-white representation using a defined intensity threshold. The goal of this preprocessing step was to isolate the knot structure and reduce irrelevant background information, including shadows, skin pad texture, and reflective surfaces. This approach aimed to shift model attention toward edge detection and structural differences between tight and loose knots, where loose knots exhibit a visible gap in the top throw.

In a second preprocessing approach, image contrast was scaled by a factor of 1.5 to amplify differences between light and dark regions of the images, particularly between the suture and the lighter skin pad. Examples of both tight and loose knots processed using this technique are shown in Figure 2B. The intent of this method was similarly to reduce background noise and partially suppress texture; however, shadows remained prominent.

These two processed datasets, along with a dataset of unedited images (Figure 4C), were evaluated using K-fold Cross-Validation. K-fold is a method used to assess model performance more reliably by partitioning the dataset into multiple subsets and iteratively training and testing the model across different train-test splits. In this study, K = 6 was used to balance computational

efficiency with reliable model evaluation. This approach was used to compare preprocessing techniques by estimating their cross-validated performance prior to final model training.

Based on the K-fold results presented in Table 1, the unedited image dataset achieved the highest performance across all metrics and was selected for training a ResNet50 convolutional neural network within the full workflow.

Figure 2: Preprocessed images of both classes (loose, left; tight, right). (A) Images are made binary to enhance edge detection. (B) Contrast is increased by x1.5 to reduce background information. (C) Unedited images with no preprocessing are used for comparison.

Dataset Type	Average Accuracy	Average Recall	Average Precision	Average F1 Score
Binary Images	53.00	0.5300	0.3386	0.3877
Contrasted Images	61.58	0.6158	0.5216	0.5166
Unedited Images	73.93	0.7393	0.7651	0.6930

Table 1: K-fold cross-validation results. *Unedited images yielded the highest performance across all evaluated metrics, including average accuracy, recall, precision, and F1-score.*

Model Training

A ResNet50 convolutional neural network was trained using the unedited images within the PyTorch Python framework. PyTorch was selected due to its flexibility, strong community support, and efficient deployment capabilities on embedded systems, making it well-suited for real-time inference applications. Additionally, its dynamic computation graph and ease of debugging enabled more streamlined experimentation during model development.

Given the relatively small dataset, a batch size of four was chosen to allow for more frequent weight updates during training, which can help produce a smoother optimization trajectory and reduce the risk of overfitting. The original dataset, consisting of 200 images, was partitioned into training (80%), validation (10%), and testing (10%) subsets, maintaining the

class balance between the “Tight” and “Loose” categories. Model training and hyperparameter tuning were conducted using the training and validation sets, respectively. Upon completion, the model was saved both as learned weights and as a full serialized model definition to ensure reproducibility and ease of deployment.

The held-out testing set was then used to evaluate model performance on unseen data. Predictions generated from this set were used to construct the confusion matrix shown in Figure 3, summarizing the key performance metrics including precision, recall, accuracy, and F1 score. The model has a strong and well-balanced performance, achieving an overall accuracy of 95%. For the “Tight” class, the model attained a precision of 1.00, indicating that all instances predicted as “Tight” were correctly classified. However, a slightly lower recall suggests that a small number of “Loose” knots were predicted as “Tight”, contributing to false positives. Despite these small discrepancies, the model achieves an overall F1 score of 0.95, reflecting a good balance between the two classifications.

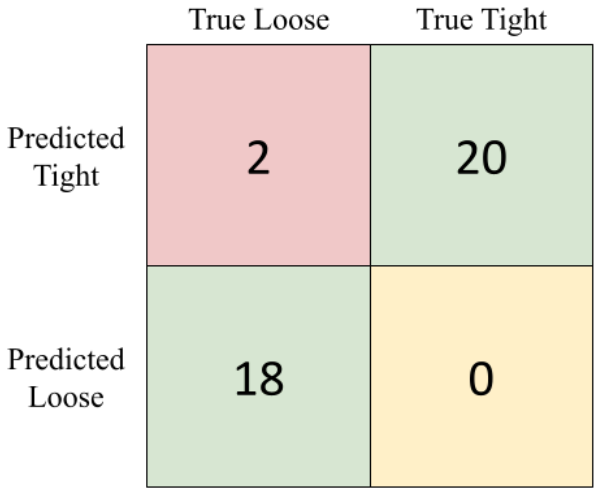

Figure 3: Confusion matrix for ResNet model testing. The model achieved an overall accuracy of 95% with a slightly better performance on tight knots, likely due to their more consistent structure.

To further interpret model behavior, a visualization technique based on gradient-weighted class activation mapping (Grad-CAM) was employed to generate a heatmap highlighting the regions of each image most influential in the model’s decision-making process. As illustrated in Figure 4, the heatmap confirms that the model primarily focuses on the knot region when making classifications, while also incorporating some peripheral features, including shadowing and background edges near the knot’s profile. This alignment between model attention and expected physical features provides additional confidence in the model’s validity and interpretability.

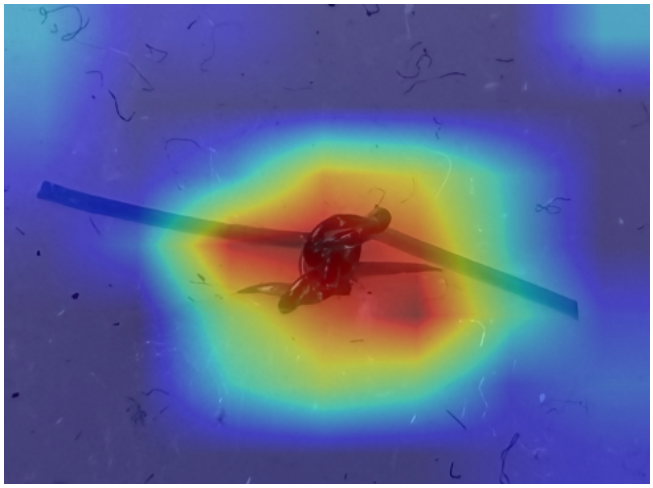

Figure 4: Heatmap for the ResNet model classification tasks. The gradient shows high focus points in red and low focus areas in blue. Visualization confirms the model is identifying and referencing the knots in the images submitted for classification. Ideally, only the top through of the knot would be highlighted.

Workflow Implementation

To develop the physical training module, the ResNet model was then deployed on a Raspberry Pi 5. This platform was selected due to its compact form factor, and sufficient computational capability to support on-device inference in a postable setting. The same Arducam IMX477 Pi HQ Camera used during dataset creation was integrated with the system, along with a simple circuit board containing three LED outputs and a user input button shown in Figure 5.

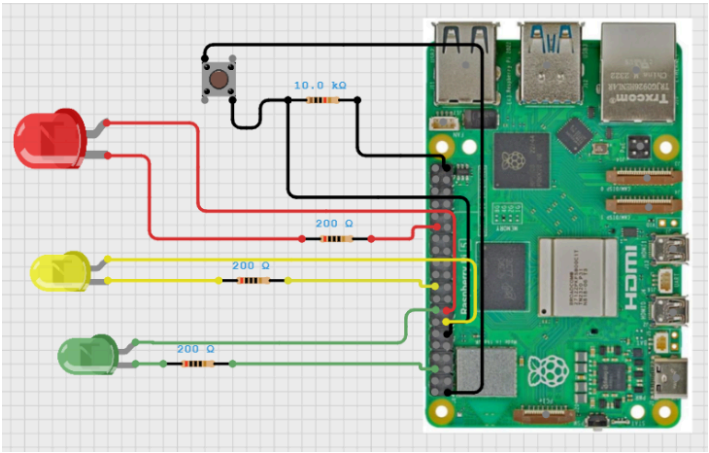

Figure 5: Raspberry Pi pin-out diagram with LED outputs and user input button. The button is connected to pin 17 (pull-up, active LOW). LEDs: yellow (GPIO 27), green (GPIO 22), red (GPIO 23).

To maintain consistency with the training dataset and eliminate variation in image capture angle, a custom 3D-printed mount was designed to fix the position and orientation of the camera 45° relative to a skin-pad. The final prototype in Figure 6 consists of a Raspberry Pi 5 connected to the Pi HQ camera and the LED/button circuit. The system can be wirelessly connected to an external display to provide a live preview of the camera feed prior to image capture.

The implemented workflow (Figure 7) is initiated by a button press, which activates the live camera preview. A central reference marker on the display screen is used to guide consistent positioning of the knot within the center of the field of view. Once aligned, a second button press captures the image and initiates the model pipeline, including cropping and classification. A yellow LED indicates that processing was in progress. Upon completion, the classification result is communicated to the user via a green LED (tight, adequate knot) or red LED (loose, inadequate knot). An additional indicator for an overly tight knot was not included, as excessive tightening typically results in premature suture failure before a stable knot state can be achieved.

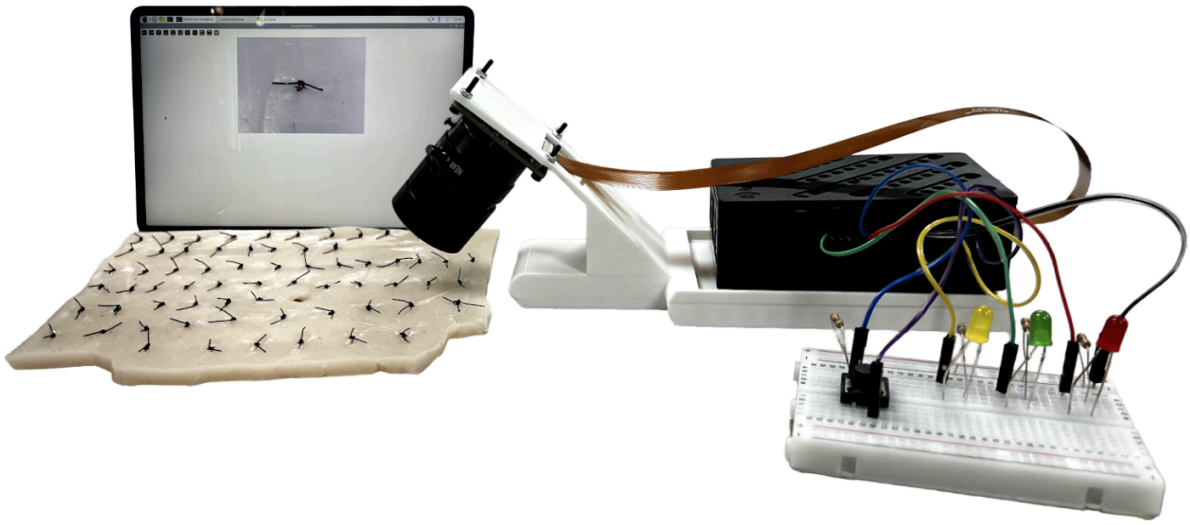

Figure 6: Overall system setup showing HQ camera (center), Raspberry Pi (rear right), and breadboard hardware (front right).

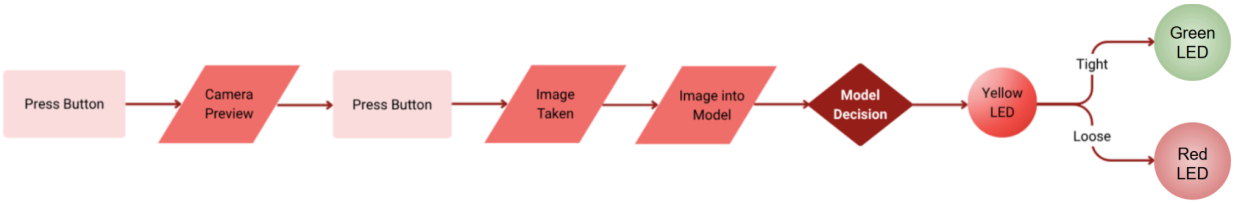

Figure 7: Workflow diagram of the user interaction and classification pipeline. A button press initiates camera preview and image capture, after which the image is processed by the model. The model's decision triggers LED feedback: yellow indicates processing, green indicates "tight," and red indicates "loose."

System Testing

Following deployment of the machine learning model onto the Raspberry Pi, preliminary manual testing revealed poor performance in classifying loose knots compared to pre-integration testing conducted in a Jupyter Notebook environment. To improve prediction reliability, an 85% confidence threshold was implemented, requiring classifications labeled as "tight" to exceed this threshold.

System performance testing was conducted using a new dataset of $n = 40$ tied knots, consisting of 20 tight and 20 loose samples. Images were captured manually on the Raspberry Pi system and processed by the deployed model, which generated a classification output and confidence score for each input. Results were compiled into a confusion matrix to evaluate model performance and allow comparison with pre-integration performance metrics. Performance was assessed using standard classification metrics, with "tight" class as the positive case. Model performance was evaluated using accuracy, precision, recall, and F1 score, defined as follows: Accuracy was calculated as the proportion of total correct predictions. Precision (positive predictive value) quantified the proportion of predicted "tight" knots that were actually tight. Recall (sensitivity) measured the proportion of the actual "tight" knots correctly identified. The F1 score was calculated to provide a balanced measure of precision and recall.

System latency testing was conducted to ensure timely user feedback. Latency was defined as the total time from the second button press, which is the image capture trigger, to the final LED output response. Testing was performed on a sample of $n=20$ images, consisting of 10 tight and 10 loose samples. Timing measurements were recorded and logged directly within the Python code and reported alongside the model's knot classification and confidence score.

Results

The integrated machine learning model was evaluated on the Raspberry Pi using a dataset of $n = 40$ knot images. Preliminary testing revealed that the deployed model exhibited bias toward the "tight" class, as all input images were initially classified as "tight". As a result, no samples were initially predicted as "loose", indicating poor class separation after deployment. To address this limitation, an 85% confidence threshold was implemented. Under this threshold, a knot was classified as "tight" only if the model prediction was "tight" and the associated confidence score was ≥85%. The distribution of confidence scores used to validate this threshold is shown in Figure 7.

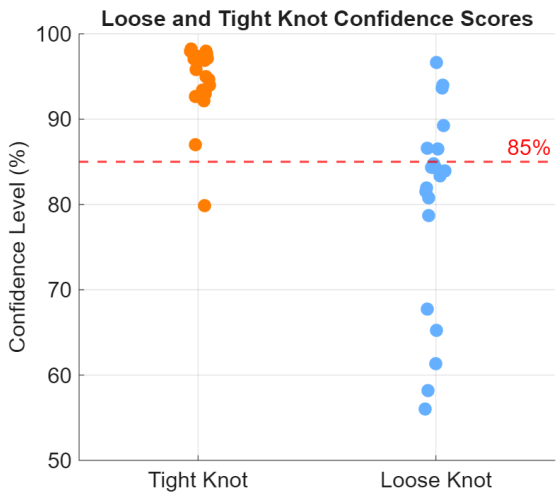

Figure 7: Distribution of model confidence scores measured during testing. Tight knot inputs exhibited consistently higher confidence scores, whereas loose knot inputs produced lower and more variable confidence values. An 85% confidence decision threshold is shown and was applied to classify predictions as "tight".

As displayed in Figure 7, confidence scores for true tight knots were consistently high, whereas loose knot samples had generally lower and more variable confidence values. This separation supported the use of a confidence threshold to improve classification reliability and reduce false positive predictions.

Following threshold implementation, model performance was evaluated using a confusion matrix (Figure 8), with "tight" defined as the positive class. The system achieved an accuracy of 82.5% and a recall of 85%, both exceeding the design requirement of ≥ 80%. The F1 score was 0.84, meeting the target value of 0.80. However, precision was measured at 76%, which falls below the ≥ 80% requirement, indicating that false positives remained despite the implementation of the confidence threshold.

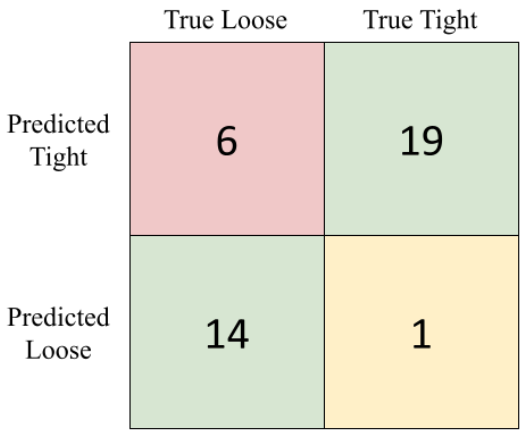

Figure 8: Confusion matrix for model performance with the 85% confidence threshold implemented. A true positive (TP) is defined as a tight knot correctly classified as tight. The model meets the overall accuracy requirement of ≥ 80%, but exhibits false positives, as reflected in the precision metric.

Latency testing results are displayed in Figure 9. Loose knots exhibited slightly greater variability in latency compared to tight knots, although the differences were minimal and remained within tenths of a second. The mean latency for tight knots inputs was 3.94 seconds, while loose knots averaged 3.93 seconds, resulting in an overall mean latency of 3.93 seconds. These results satisfy the latency design requirement of less than 5 seconds and demonstrate the system's ability to produce real-time feedback.

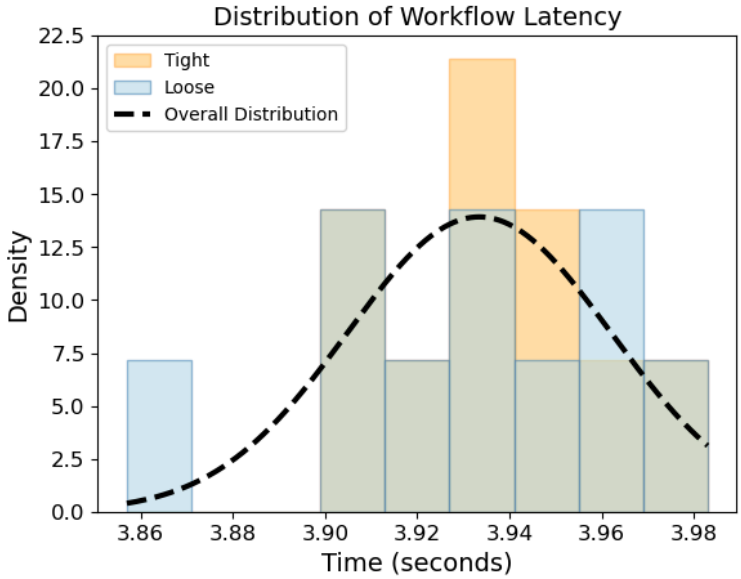

Figure 9: Distribution of latency (seconds) measured from the second button press to LED feedback for both tight and loose knot classifications. The mean latency was 3.94s for tight inputs and 3.93s for loose inputs, indicating a negligible difference (0.01) between classes. Overall mean latency was 3.93 seconds.

Discussion

This work describes the design, development, and validation of a machine learning system for automated assessment of suture knot classification, specifically as adequately tight or too loose. The project focused on the end-to-end development of a real-time suturing training system that combines machine learning, embedded hardware, and user feedback.

Deployment and System Reliability

During early model development, the ResNet-based model achieved strong performance metrics in a desktop computing environment. However, integration onto the Raspberry Pi revealed reduced accuracy during manual testing, particularly when identifying loose knots. This behavior may indicate model overfitting, as performance was strong on the training dataset but decreased when evaluated on new, previously unseen data. This discrepancy highlighted an important challenge in embedded machine learning: models that perform well in controlled environments may experience reduced reliability when deployed in real-world settings, where variability in factors such as lighting, suture materials, and user technique can differ significantly from the controlled training environment. This challenge can be addressed through incorporation of more diverse and representative training data that represents real-world conditions.

Impact of Image Preprocessing and Cross-Validation

Multiple preprocessing strategies were explored to improve model performance and compensate for the limited dataset. Binarization effectively reduced lighting variability and clearly defined knot edges but removed depth cues and oversimplified knot structure. Contrast enhancement reduced background noise while preserving more structure, but it exaggerated shadows and occasionally overexposed the suture material.

K-fold Cross-Validation ultimately showed that the model performed best using unedited images. These findings informed the final workflow and highlight the importance of evaluating preprocessing choices early in the design of machine learning–based imaging systems.

Dataset and Imaging Constraints

The dataset size was limited due to time constraints and available materials during the project. As a result, data collection was conducted under consistent and controlled conditions to support initial model development. While this approach helped ensure repeatability and simplified early training, it reduced the variability present in the dataset. The model was therefore trained using a single suture material, fixed lighting conditions, and a fixed imaging setup, which may have contributed to differences in performance observed during manual testing. Similarly, the fixed camera stand improved repeatability but constrained suture

placement and reduced flexibility. These constraints are typical of early-stage prototypes and provide clear direction for future system refinement.

Embedded System Feasibility

The final integrated prototype successfully demonstrated real-time image capture, processing, and feedback using a low-cost embedded platform. The system achieved a mean latency of 3.93 seconds from button press to LED feedback, meeting the project requirement of providing results within 5 seconds. This confirms the feasibility of deploying convolutional neural networks (CNNs) on small-scale computing hardware for hands-on clinical training applications.

Conclusion

Key Insights

Several key insights emerged from the design, development, and testing of the suture knot classification system. Implementing a confidence threshold improved system reliability by reducing the influence of low-confidence predictions that were more prone to error, leading to more consistent classification results in practice. Image preprocessing was found to significantly influence model performance. Techniques such as binarization and contrast enhancement improved certain visual features but also introduced trade-offs, including loss of depth information and exaggerated image distortions. This highlighted the importance of carefully balancing preprocessing choices when designing a system.

Differences between controlled testing performance and manual validation revealed limitations in model generalization. The model initially showed strong performance metrics, specifically with an accuracy of 95%, but manual testing after the confidence threshold implementation revealed a much lower accuracy of 82.5%, with issues mainly arising from incorrect loose knot classification. These discrepancies were likely influenced by the small dataset size or additional noise from the Raspberry Pi. In addition, the imaging setup used a non-adjustable camera stand which limited where sutures could be placed on the skinpad.

Successful integration of the model onto a Raspberry Pi demonstrated that real-time classification of suture knot tightness is feasible using low-cost hardware, supporting the potential for practical deployment in training environments.

Project Impact

This project resulted in the successful development of a standalone clinical training simulator capable of providing real-time feedback on suture knot tightness. The system addresses a key gap in current training methods, which rely heavily on subjective instructor feedback and student intuition. By providing immediate and objective feedback, the system has the potential to

improve training efficiency, support independent practice, and promote more consistent suturing technique. The low cost and portability of the prototype also make it suitable for classroom, laboratory, and simulation-based training environments. More broadly, this project demonstrates how machine learning and embedded systems can be combined to create accessible, intelligent tools for clinical skills training.

References

[1] “The ideal surgical knot (according to science).” Accessed: Oct. 07, 2025. [Online]. Available: https://healthcare-in-europe.com/en/news/ideal-surgical-knot.html

[2] “What Is Wound Dehiscence?,” Cleveland Clinic. Accessed: Oct. 07, 2025. [Online]. Available: https://my.clevelandclinic.org/health/diseases/wound-dehiscence

[3] V. K. Shanmugam *et al.*, “Postoperative wound dehiscence: Predictors and associations,” *Wound Repair Regeneration*, vol. 23, no. 2, pp. 184–190, Mar. 2015, doi: 10.1111/wrr.12268.