Group 18 System Design Rev0
MECHTORN 4TB4 – Fall 2025

Authors:

Mohammed Fuzail

Ayush Patel

Tathagata Sikdar

Haerain Yu

Table of Contents

Introduction:

This document presents the system design for an edge-based, multi-camera computer vision platform developed to automatically assess dairy cattle lameness under real barn operating conditions. The system combines synchronized RGB and depth video acquisition, automated cow detection and identification, machine-learning–based gait analysis, and a human-in-the-loop review process to generate reliable lameness scores for individual animals. The design describes the physical setup of the camera system, the organization of the software components running on the edge device, the data flow from raw video to model inference output, and the supporting database and user interface elements. Particular emphasis is placed on the learning pipeline, which allows the model to continuously improve by incorporating human corrections made through the review dashboard. The document also specifies the expected behaviour of each subsystem during normal operation, identifies possible undesired events, and outlines the associated mitigation strategies that ensure robustness in noisy agricultural environments. Overall, the goal of this system is to create a practical, scalable tool that supports farm personnel by enabling earlier detection of lameness, reducing missed cases, and improving overall herd health and productivity.

Purpose:

The purpose of this project is to assist the Cattlytics team in deploying an edge computing system which utilizes computer vision along with machine learning to analyze real-live footage of cow gait (stride pace and posture) and categorize each cow on a scale of 0 to 5 for lameness with a confidence level. Cow lameness is a clinical sign of pain implying an issue with the locomotor system of a cow stemming from possible issues related with the environment, management or individual cow habits. Late detection of lameness is often catastrophic for both the cow and its owner; often leading to death of cow or substantial economic losses for the farmer. The objective of this system is to be in the barn and supplement the visual inspections of cows by alerting farmers of early signs of lameness prior to significant negative impacts. The system will be non-invasive; only depending on visual inspection and will be cost effective while having a high confidence level of lameness detection. The Cattlytics team will be deploying the project in phases, but due to the timeframe of the project our team may or may not be participating in the multiple stages of the project.

Scope:

The scope of this project includes a software, mechanical and electrical component with the emphasis put on the technical software data pipeline. The system will be installed around a return lane where cows will walk through one at a time in a single file. There will be 6 input video streams sourced from 4 RGB Reolink RLC-843A cameras situated above, in front, beside

and hoof level of the lane and two depth cameras, Intel RealSense D455 and LIPSedge DL both situated above the lane. The system will be deployed in two phases: phase 1 (only 4 RGB video streams) and phase 2 (4 RGB video streams + 2 RGB video streams + 2 depth video streams) Implementation of phase 2 is dependent on the performance on phase 1, inadequate performance of phase 1; when accuracy improvement plateaus at 88-92% accuracy, will trigger implementation of phase 2. The additional cameras will provide both RGB and depth video steams with the goal that the accuracy can be increased to 95%+. All video streams will be routed via cable to a central mini computer which will be responsible for processing all video streams and utilizing machine learning to analyze and log outputs. A human in the loop online learning methodology is adopted for this project with daily reviews facilitated through the python dashboard. Human reviews performed from this dashboard are then fed back into the edge device system to improve the model and increase accuracy for future lameness inference. In regards to the hardware, modifications will be added to the cameras to ensure image fidelity by keeping lens clean and a housing will be fabricated to hose the computer providing an IP-95 protection to the computer (dust and water protection), design of such components are still being worked out with the client and hence are not a big focus of this system requirement document.

Table 1: Hardware Table

Phase	Hardware	Type	Key Specification	Mounting Position	Primary use
Phase 1	Reolink RLC-843A	RGB Camera	4K @ 25FPS	Top	Body tracking, spine alignment
	Reolink RLC-843A	RGB Camera	4K @ 25FPS	Side	Gait analysis, leg movement
	Reolink RLC-843A	RGB Camera	4K @ 25FPS	Front	Head carriage, approach giat, visual tag backup
	Reolink RLC-843A	RGB Camera	4K @ 25FPS	Hoof	Hoof placement detail
	DreamQuest Mini PC Windows 11 Pro	Mini PC	32 GB RAM	Central processing unit	All video stream will be routed here and all resource management will be carried out here.
Phase 2 Additions	Intel RealSense D455	RGB + Depth Camera	Depth: 1280x720 @ 90FPS	Top	Body volume estimation, back arch,

			RGB: 1280x800 @ 30FPS		curvature depth measurement
	LIPSedge DL	RGB + Depth Camera	Depth: 320x240 @ 30FPS (20cm to 4m range)	Top	Complimentary body depth data, wider field of view.
			RGB: 1920x1080 @ 30FPS		

System Architecture:

Our system will consist of 3 main components which will interact with each other. The three systems being:

1. Edge Device.
2. Review Dashboard.
3. Learning pipeline.

The input to the system are the video stream gathered through the 6 cameras along with the initial labelled dataset for the machine learning pipeline. The initial dataset to the system will consist of 50 labelled cows with human denoted lameness which will be used to initially start the bootstrap program and allow to learn overtime. The output of the system is data condensed into table format and written to a database in the form of a CSV.

Table 2: System Monitored Inputs

Input	Type	Example	Description
M_RGB_TOP_VIDEO	numpy.ndarray	N/A	Top-down view of return lane.
M_RGB_SIDE_VIDEO	numpy.ndarray	N/A	Side view of return lane.
M_RGB_FRONT_VIDEO	numpy.ndarray	N/A	Front view of return lane.
M_RGB_HOOF_VIDEO	numpy.ndarray	N/A	Hoof level of return lane.
M_DEPTH_TOP_VIDEO_INTEL	numpy.ndarray	N/A	Top-down Depth perspective of return lane with certain dimensions.
M_DEPTH_TOP_VIDEO_LIPSEDGE	numpy.ndarray	N/A	Top-down Depth perspective of return lane with alternate dimensions.
M_ML_DATA	dataset	N/A	Data for bootstrap ML model.

Table 3: Predictions Table: System Controlled Outputs

Output	Type	Example	Description
C_ID	Integer	4	Primary key, autoincrement to keep track of total entries in database.
C_TIMESTAMP	Datetime	05-07-2026 07:04:34	When the cow was analyzing
C_COW_ID	Text	"Cow_0109"	Cow identification (number from ear tag).
C_PREDICTION	Integer (0 – 5)	3	Lameness score.
C_CONFIDENCE	Real	89.2	System confidence in lameness score.
C_NEEDS_REVIEW	Boolean (default false)	true	If system confidence low, require review.
C_REVIEW_PRIORITY	Text	"*high*", "*low*" or "*null*"	If lameness high and confidence low, then high priority.
C_CORRECTED	Boolean (default false)	true	Status of review correction.
C_CORRECTION_VALUE	Integer (0 – 5)	3	Human lameness score given upon review.
C_CORRECTION_TIMESTAMP	Datetime	05-07-2026 07:04:34	When human lameness was given.
C_MODEL_VERSION	Text	"V1.14"	Current model of system.

Table 4: Additional optional System Outputs (Dependent on Timely Project Completion)

Metric	Type	Example	Description
C_COW_ID	text	"Red_Heifer_A176"	Farmer chosen identification.
C_EAR_TAG	text	"0192"	Number on ear tag.
C_BIRTH_DATE	date	05-07-2026	Birthdate of cow.
C_LAST_SEEN	datetime	05-07-2026 07:04:34	Last known date in the system.
C_AVG_LAMENESS_SCORE	Real	3.2	Average lameness score.
C_LAMENESS_TREND	text	"*improving*", "*worsening*", or "*stable*"	Lameness trend of the cow.

Table 5: Lameness Scoring

Integer Value	Textual Representation
0	Normal – Level back, even gait.
1	Slight irregularity.
2	Uneven gait – mildly lame.
3	Obviously lame – short strides.
4	Severely lame.
5	Non-weight bearing.

File System:

To support the development of this multi-camera gait analysis system, the software has been structured into a clear and modular file hierarchy. The organization separates core processing scripts, model definitions, utilities, and data directories to streamline implementation and maintenance. Input and output data paths are explicitly defined to ensure traceability from raw video streams through labelled datasets and trained models. The output data paths depicted only include the subfolder labelled "*labels*" as this is where the output files will be generated and stored, hence all other file paths are consistent with the input file path. The following sections outline the directory structure and the Python files used in the project, along with brief descriptions of their roles and the primary libraries leveraged for computer vision, machine learning, and data management. In addition to this, we have also included the dependent core python libraries used categorized by use case and version. This organization enhances readability of the codebase and clarifies the implementation workflow.

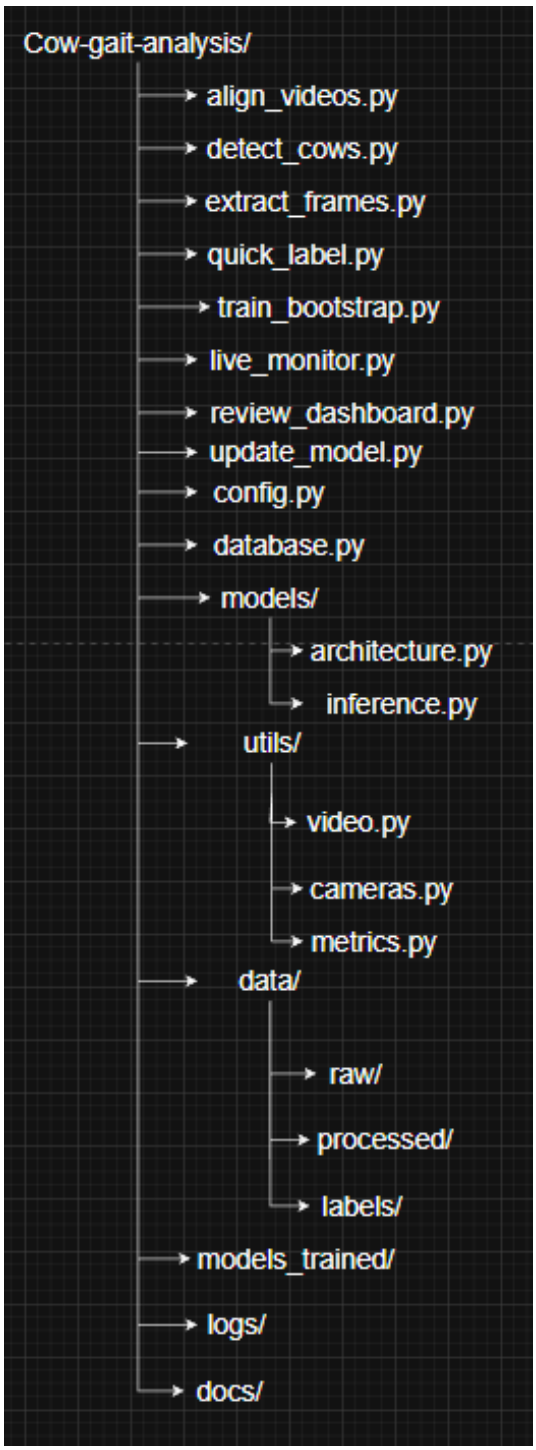

Figure 1: Input File Structure

Table 6: Component Breakdown

Component	Role in System
align_videos.py	Manual video alignment tool.
detect_cows.py	Automated cow detection.
extract_frames.py	Bulk frame extraction
quick_label.py	Streamlit labelling interface.
train_bootstrap.py	Initial model training
live_monitor.py	24/7 monitoring system.
review_dashboard.py	Streamlit review interface
update_model.py	Incremental learning pipeline
config.py	Configuration management
database.py	Database utilities
architecture.py	Model definitions
inference.py	Inference utilities
video.py	Video processing utilities
cameras.py	Camera management
metrics.py	Performance tracking
raw/	Holds raw 3-hour videos for initial offline implementation and training
processed/	Extracted cow datasets
labels/	Training labels
models_trained/	Saved model checkpoints
logs/	System logs
docs/	Documentation

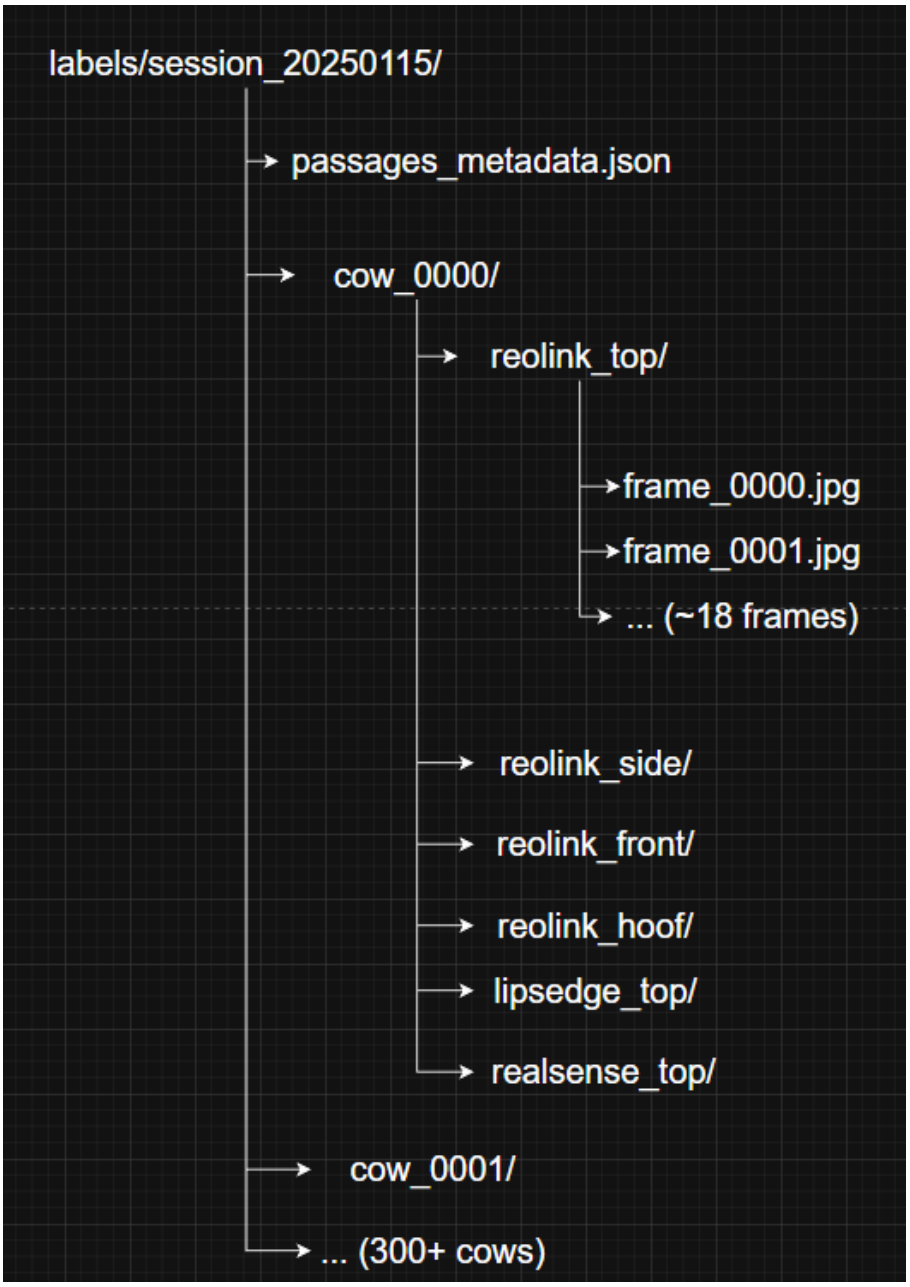

Figure 2: Output File Structure

Table 7: Used Python Packages

Use case	Libraries	Version
Computer Vision	opencv-python	4.8.0
	opencv-contrib-python	4.8.0
Deep Leaning	torch	2.0.0
	torchvision	0.15.0

Depth Cameras	pyrealsense	2.54.0
	primesense	N/A
Data processing	numpy	1.24.0
	pandas	2.0.0
	scikit-learn	1.3.0
Web Dashboard	streamlit	1.28.0
Database	sqlalchemy	2.0.0
	sqlite3 (built-in)	N/A
Visualization	matplotlib	3.7.0
	plotly	5.17.0
Progress Tracking	tqdm	4.66.0
Utilities	pillow	10.0.0

Context Diagram:

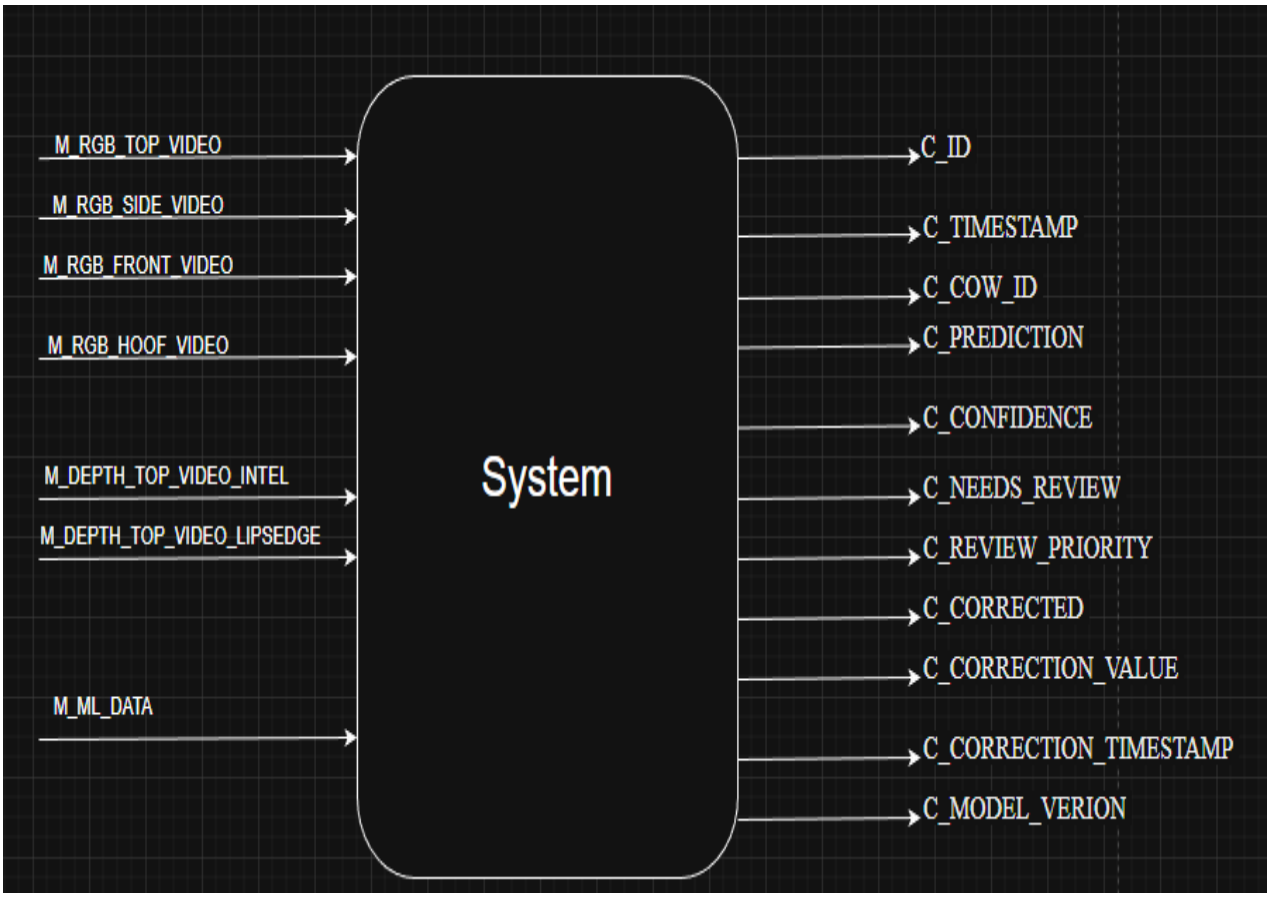

Figure 3: Context Diagram

Detailed Sub-System Diagram:

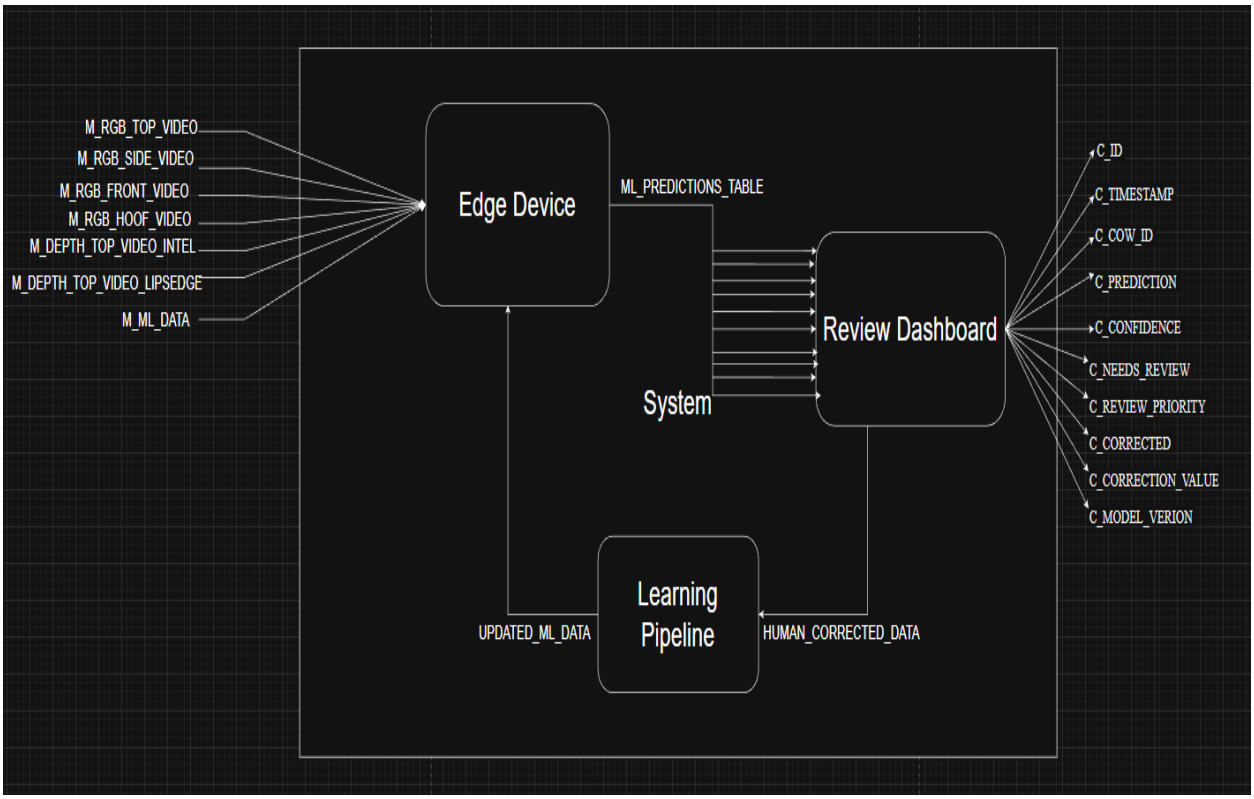

Figure 4: Module Decomposition Diagram

Subsystem 1: Edge Device

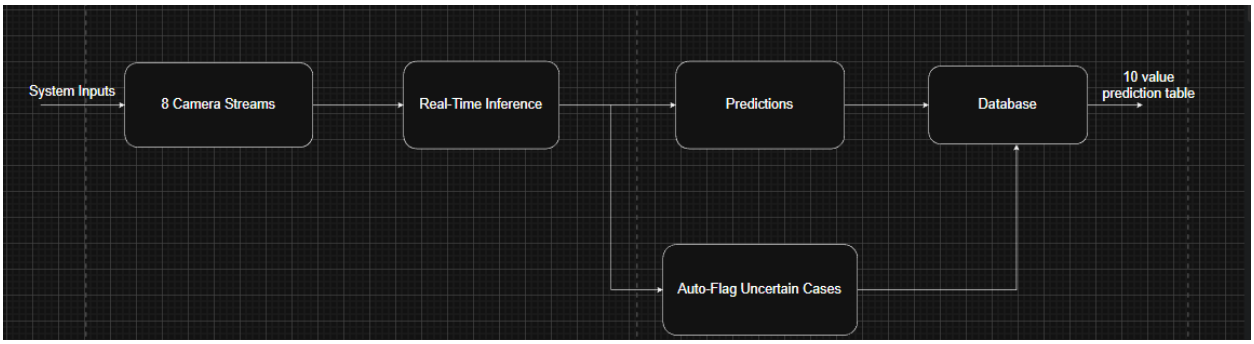

Figure 5: Edge Device Operation

Behaviour Description:

The objective of this subsystem is to analyze the environment through the use of the different cameras and capture video data. Initially the system will run on recorded video files until a certain threshold accuracy is achieved after which the system will be fast enough to work in real time. The main purpose of this sub system is to achieve the following functions:

1. Detect cow entering lane
2. Capture synchronized frames from all 6 video streams
    a. Either through manual or automatic synchronization algorithms.
3. FPS down sampling to save memory.
4. Run model inference to generate lameness scores.
    a. Calculate confidence score.
    b. Log all prediction
    c. Flag low-confidence predictions for review

Following this, the implementation pipeline will move unto the review dashboard subsystem. The input of this system will be the raw video footage of cows traversing the return lane while the output will be the 10 values aforementioned as part of the predictions table also known as the controlled outputs. At this stage all of the data from the predictions table has not yet been reviewed by humans and hence differs from the system output/review dashboard output. The output of this system will be used by the review dashboard to organize certain uncertain lameness scores which will be reviewed by humans. The above-mentioned functions are described in detail below.

Capture Synchronized Frames

Motivation

The different camera streams are used for different viewpoints and hence capture different biomechanical features of gait, such as stride length, hoof placement, back curvature, and head movement. For detailed breakdown of different biomechanical objectives please see the hardware list. To accurately assess lameness, these views must represent the same moment in time and represent the same cow; since only the front cow can identify through the ear tag. The edge subsystem therefore needs to capture synchronized frames across all six video streams so that a consistent, multi-view representation of the cow is available to the model. If frames were unsynchronized, the system could combine mismatched gait phases, which would degrade model performance and produce unreliable lameness scores.

Table 8: File Breakdown

File	Input	Input Data Type	Output	Output Data Type
align_videos.py	6 unsynchronized raw video stream footage	Array of numpy.ndarray	6 synchronized raw video stream footage	Array of numpy.ndarray

Table 9: Software Model

Class Name	Key Responsibilities	Methods
MultiCameraSynch	Synchronize the different camera streams.	add_frame(), get_synchronized_frames()

Detect Cow Entering Lane

Motivation

The edge system must be able to automatically detect when a cow enters the lane because the cameras operate continuously and record large amounts of mostly empty footage. Detecting entry allows the system to distinguish meaningful gait events from idle periods, which drastically reduces storage, computation, and human review workload. This detection step also defines the temporal window of interest for downstream processing, ensuring that only relevant segments are analyzed and logged. Without automatic cow-entry detection, the system would either waste

resources processing empty video or rely on manual trimming, which is impractical in a farm environment.

Table 10: File Breakdown

File	Input	Input Data Type	Output	Output Data Type
detect_cows.py	6 synchronized raw video stream footage	Array of numpy.ndarray	passages_metadata.json with timestamps	Json

Table 11: Software Model

Class Name	Key Responsibilities	Methods
VisualTagReader	OCR-based visual ear tag recognition utilizing the front camera. Primary identification method for the system.	preprocess_frame(), detect_tag_region(), read_tag_from_frame()
CowIdentifier	Identify cows using visual ear tag recognition and fall back to timestamp based identification if tag unreadable.	Identify_cow()

FPS Down Sampling

Motivation:

Cow gait is extremely slow often with roughly 1-2 steps every second. The reolink cameras record at 25FPS. Assuming cow is present in the FOV of camera for roughly 5-10 seconds; the cow would have taken ~20 steps yet 250 frames captured. By sampling every 3-6 frames we can achieve a target FPS of 5-10 FPS resulting in ~100 frames. The memory bandwidth is reduced from ~113GB to ~7.5GB per session. This implementation is more realistic considering the memory constraints while still preserving substantial gait information.

Table 12: File Breakdown

File	Input	Input Data Type	Output	Output Data Type
extact_frames.py	6 synchronized raw video stream footage	Array of numpy.ndarray	Creates cow_0000, cow_0001 with 6 camera subfolders	jpeg

			(reolink x4, Intel , lipsedge).	

Model Inference

Motivation

The primary objective of this function is to automatically assess lameness without requiring continuous human observation. Running model inference directly on the edge device allows real-time generation of lameness scores while video is being captured. Edge inference avoids the need to upload large video files to the cloud, which is important in farm environments with limited internet connectivity. It also enables rapid feedback for early detection, improving animal welfare and reducing productivity losses. This step operationalizes the trained machine learning model in routine daily monitoring. In the initial stages of the system a manual data set will need to be provided to the system to ensure initial bootstrap. For this purpose, manual human labelling is also provided through the use of the quick_label.py python script which will allow a human operator to rate the lameness based on short video samples.

The inference models to be used in this system are divided into three for the sake of practicality and performance. The three models being CowGaitClassifier, MultiViewClassifier and lastly TemporalGaitClassifier. A detailed description of each is provided below. Additionally, details regarding logging the prediction and confidence levels along with flagging for human review are also listed below.

CowGaitClassifier

The CowGaitClassifier represents the simplest form of inference used in the system. It operates on images from a single camera view and produces a lameness score directly from individual frames or short clips. This model treats each frame independently and does not explicitly combine information across multiple viewpoints or time steps. Its primary purpose is to provide a lightweight baseline classifier suitable for rapid inference and constrained compute environments, such as early prototyping or reduced-hardware deployments. Although it does not capture full gait dynamics, it establishes the fundamental mapping from visual features to lameness scores and serves as the foundation for the more advanced multi-view and temporal models.

MultiViewGaitClassifier

The MultiViewGaitClassifier extends the basic classifier by integrating information from several synchronized camera perspectives, such as top, side, front, and hoof views. Each view

captures different biomechanical cues—stride length, back curvature, weight shifting, and foot placement—which are complementary and, when combined, improve diagnostic accuracy. In this architecture, features are typically extracted independently from each view and then fused to form a joint representation before classification. This model is motivated by the physical reality that lameness is a three-dimensional phenomenon that cannot be fully observed from a single viewpoint. By leveraging synchronized multi-camera input, the MultiViewGaitClassifier produces more robust and reliable lameness assessments under real-world farm conditions.

TemporalGaitClassifier

The TemporalGaitClassifier incorporates the dimension of time, modeling how gait evolves over sequential frames as the cow walks through the lane. Instead of classifying frames independently, this model processes an ordered sequence of images or features using temporal architectures such as RNNs, LSTMs, or temporal convolutional networks. This allows the system to capture stride cycles, asymmetry between steps, hesitation patterns, and rhythm abnormalities that are often key indicators of lameness but may not be evident in single frames. Temporal modeling also improves stability of predictions by smoothing out frame-to-frame noise. This classifier therefore represents the most advanced inference stage, combining spatial visual information with motion dynamics to more closely approximate the way human experts evaluate gait in practice. As mentioned previously, implementation of the TemporalGaitClassifier will be dependent on the performance of the previous two implementation.

Table 13: 3 Types of Inference Models

Inference Model	Key Design Decision
Cow Gait Classifier	Fast Inference, takes less than ~50ms per cow on edge device.
	Pre-trained model acts as the backbone leading to faster initial training.
	Confidence scores for uncertainty estimation
	Model size is relatively small with only taking ~10 MB
Multi View Gait Classifier	Shared Backbone: all cameras use same feature extractor (reduced parameters from 14M to 2M).
	Learned Weights: Model automatically learns which views are most informative.
	Graceful degradation: Works with 1-7 cameras, automatically reweights.
	Side view priority: Architecture based toward primary gait view (which is the side angle).

	Early fusion: Combines features before classification (vs late fusion of predictions).	
	Relatively fast: takes less than ~100ms on edge device.	
Temporal Gait Classifier	**When to add?**	Single frame accuracy plateaus at 88-90%
		Misclassifications are "borderline" cases (score 1-2 lameness)
		After phase 1 is stable.
	Sequence based classification considers time, bases decision on multiple frames.	
	Capture 10-15 frames over 2-3 seconds.	
	Model learns stride patterns, asymmetry timing	
	Slower out of all three: ~300ms but more accurate for subtle lameness.	

Table 14: Software Model

Class Name	Key Responsibilities	Methods
CowGaitClassifier	Perform as described above	forward(), predict_with_confidence()
MultiViewGaitClassifier	Perform as described above.	extract_features(), forward(), predict_with_confidence(), get_camera_weights()
TemporalGaitClassifier	Perform as described above.	forward(), predict_with_confidence()
StrideAnalyzer	Analyze gait symmetry and stride timing, complements temporal gait classifier	detect_hoof_contacts(), calculate_stride_symmetry(),extract_temporal_features(),

Table 15: Flag for Review Conditions

Condition	Flag for Review
Confidence > 0.85	needs_reivew = false & review_priority = null
0.85 > Confidence > 0.60	needs_reivew = True & review_priority = low
Confidence < 0.60	needs_reivew = True & review_priority= high

Table 16: File Breakdown

File	Input	Input Data Type	Output	Output Data Type
quick_label.py	Short bursts of cow gait videos of the side view.	Array of numpy.ndarray	Lameness score (0 –5)	Labels.csv
train_bootstrap.py	Lameness Score (0 – 5)	Labels.csv	Models/cow_gait_v0.pth	Weights for the trained neural network model

Subsystem 2: Review Dashboard

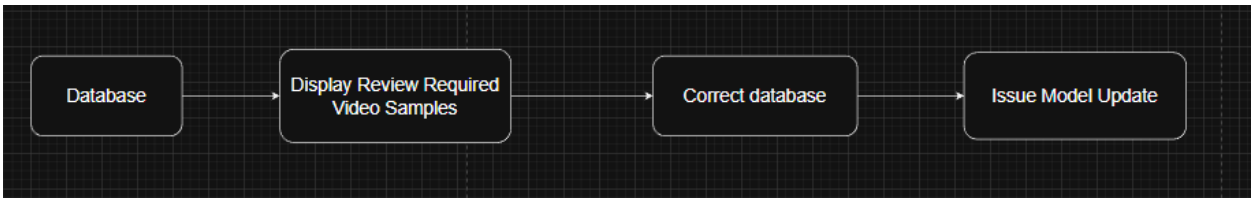

Figure 6: Review Dashboard Operation

Behaviour Description:

In the previous subsystem, the system flagged certain lameness rating for review. The objective of this system is to present these low confidence lameness ratings to a human through a simple interface and obtain the corrected lameness rating. This subsystem is implemented in python through the use of the streamlit package; which allows simple HTML based web apps to be created using only python. The subsystem will show to the user a multiview short video sample of the cow gait, the current model declared lameness rating and the confidence level. Based on this the user will be prompted to select an updated lameness rating from the same 0-5 scale. The subsystem will also prioritize lameness ratings with a "*high*" review_priority value, making sure to present these to the user primarily. The new chosen lameness ratings will then be marked in the database under the corresponding entries, the corrected flag (which denotes whether the value has been corrected or not) will be changed to true. Since the learning pipeline will take as an input the correction_value, the prior subsystem (edge device) would have already set the correction_value equal to the prediction entry if the confidence was satisfactory and no review was required. The subsystem tasks can be summarized into:

1. Display Pending Reviews (prioritized by uncertainty).
    a. Show 3-5 second video clips from multiple angles
    b. One-click correction submission.
    c. Track review progress and accuracy trends.

Table 17: Review Dashboard Features

Feature	Description
Progressive Loading	Show primary (side) view immediately (<1 second load).
	Additional views lazy-loaded in expandable section.
	Reduced perceived latency
View Prioritization	Primary: Side view (always shown, most important)
	Secondary: Top view (for body posture)

	Tertiary: Front view (head carriage, tag verification)
	Hoof/Depth views: not shown in UI (too technical and low SNR).
Possible Mobile Optimization	Responsive design works on tablet/phone
	Large touch-friendly buttons
	Single-column layout on small screens.
Keyboard Shortcuts	Power users can review in 10-15 seconds per cow.
	Number keys for scores, Enter to submit
Review Statistics	Track review time per cow
	Show accuracy trends
	Leaderboard (if multiple reviewers)
Expected Performance Targets	Dashboard load time: <2 sec
	Video playback start: <1 sec
	Review time per cow: 15-30 sec
	Daily review session: 10-15 min

Display Pending Reviews

The interface design will be implemented in the review_dashboard.py python file. This standalone python file is all that is necessary to implement the functionality described above. A table has been included below to expound on some necessary details pertaining to it. As aforementioned, the streamlit package will be heavily relied upon.

Table 18: File Breakdown

File	Function	Description
review_dashboard.py	review_dashboard()	Creates the hosting frame to host all other subcomponents
	generate_review_clips()	Generate compressed video clips for review dashboard comprising of 3 views (side, top and front)
	submit_correction()	Save correction to database, once all corrections saved. Trigger learning.

Subsystem 3: Learning Pipeline (Server/cloud)

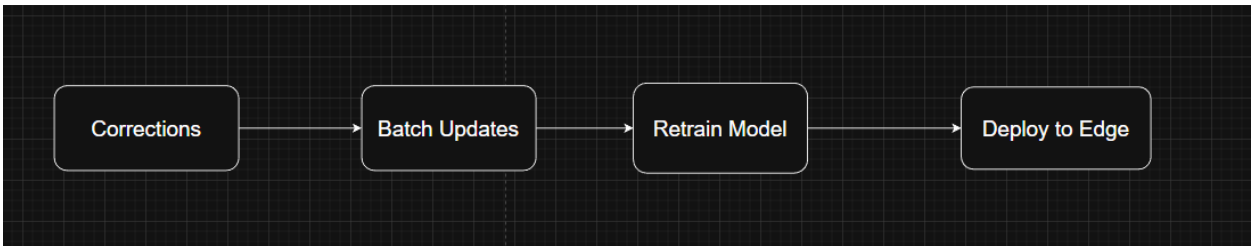

Figure 7: Learning Pipeline Operation

Behaviour Description:

The learning pipeline is responsible for continually improving the lameness-detection model as new data is reviewed. As the system runs, the edge device generates predictions, and the review dashboard allows users to confirm or correct these results. Each correction is stored as a new labeled example, along with the associated video frames and metadata. At the end of a review session, the pipeline automatically checks how many corrections were made; if 20 or more new corrections are available, the system packages these samples with a small replay buffer of past data and retrains the model to incorporate the new information. The updated model is then versioned and redeployed back to the edge device. In simple terms, the model gradually learns from its mistakes: users review predictions, the system counts corrections, and whenever enough new feedback exists, it updates itself to get a little smarter over time. The tasks of this subsystem can be summarized as such:

1. Collection corrections from dashboard.
2. Trigger model update every 20 corrections.
3. Incremental fine-tuning (low learning rate to avoid catastrophic forgetting).
4. Version models with timestamps.
5. Push updated model to edge device
6. Track accuracy metrics over time.

The complete implementation of the learning pipeline will be performed in the update_model.py python file. Due to the simplicity of the file a detailed breakdown in table format is not provided. The file will contain two functions on_correction_submitted() and trigger_model_update(). The first function will run each time a correction is performed, and it keeps track of number of corrections performed. If 20 corrections have been performed it issues trigger_model_update() which updates the model and pushes to the edge device.

Handling of Undesired Event

Under normal operating conditions, the system is expected to run continuously and autonomously with minimal human intervention. Video streams from all configured cameras are captured and synchronized, cows are detected as they pass through the lane, and lameness inference is performed on synchronized multi-view frames. Each cow is assigned an identity when possible, predictions are logged, and low-confidence cases are sent to the review dashboard for human verification. The learning pipeline then incorporates sufficient new corrections into periodic model updates. In other words, the expected behavior is a smooth end-to-end flow from data capture to prediction to incremental learning. However, real-world barn environments introduce a number of undesired events that may interrupt or degrade this process. The following table outlines these potential events and the strategies the system uses to detect, tolerate, or recover from them.

Table 19: Handling of Undesired Events

Undesired Event	System Handling / Mitigation Strategy
Camera stream drops or freezes	Automatically attempt reconnection; if unavailable, continue inference with remaining cameras and log the failure for maintenance.
Cameras become unsynchronized	Re-run synchronization routine and fall back to looser temporal alignment; discard badly misaligned frames.
Multiple cows in lane simultaneously	Reject sample as invalid, flag for human review, and prevent model update from this recording.
Cow partially occluded	Skip affected frames and continue with next valid frames; lower confidence and flag if too many frames unusable.
Ear tag unreadable or missing	Assign temporary **UNKNOWN_xxx** ID and still run inference; queue for later manual identity resolution.
Depth camera produces noisy output	Use RGB-only model fallback and suppress depth view display in UI; mark depth modality as degraded.
Poor lighting / motion blur	Reduce confidence score and forward case to review dashboard rather than auto-accepting result.
Model produces very low confidence prediction	Automatically flag for human review rather than logging as final decision.
Model crashes or returns invalid output	Fail gracefully, restart inference service, and log crash event; avoid writing corrupt results to database.

Network interruption between edge device and dashboard	Buffer results locally on edge device and push them once connection is restored.
Storage becoming full on edge device	Trigger automatic cleanup of oldest raw video while preserving labeled and training data.
Review dashboard closed mid-session	Save current progress and maintain correction counter in database so learning pipeline can resume.
Training job fails during auto-update	Roll back to previous stable model version and notify operator while logging failure.
Wrong cow direction of travel (backwards movement)	Discard gait sample and mark as invalid passage event.
Lane blockage or human entering scene	Abort collection, label as contamination, and ignore segment for inference or training.

References

Merck Veterinary Manual. (n.d.). *Overview of lameness in cattle*. Retrieved from https://www.merckvetmanual.com/musculoskeletal-system/lameness-in-cattle/overview-of-lameness-in-cattle

Green, M. J., Green, L. E., Hedges, J. J., & Blowey, R. W. (2002). A descriptive study of the prevalence and associations of lameness in dairy cattle. *Journal of Dairy Science, 85*(8), 2250–2259. https://www.journalofdairyscience.org/article/S0022-0302(06)72077-X/fulltext

Discussions with client, personal communication, 2025.