Smart Dairy Management Using Cow Muzzle Recognition

A Capstone Project Report

A Project Report Submitted
in Partial Fulfillment of Requirements
for the Degree of

Bachelor of Technology in Computer Science and Engineering

by

Gyanendra Mani - (2021CSB1090)
Nitesh Kumar - (2021CSB1117)

Under the Supervision of
Dr. Mukesh Saini
Department of CSE, IIT Ropar

Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar 140001, India
July 2025

Abstract

This project presents a comprehensive system for **Cattle Muzzle-Based Recognition**, designed to enable reliable identification of individual cattle using their unique muzzle patterns. The work explores and evaluates multiple approaches for feature extraction and classification. Initially, a **traditional computer vision pipeline** using **SIFT (Scale-Invariant Feature Transform)** was implemented to extract handcrafted features for matching. To enhance robustness and scalability, **transfer learning** with **EfficientNet** was employed, leveraging pretrained deep networks to generate more discriminative features. Further, a **deep convolutional autoencoder** was developed to learn compact latent representations, and combined with **Gradient Episodic Memory (GEM)** to support **continual learning**, enabling the system to learn from new cattle data incrementally without forgetting prior knowledge.

In addition to the recognition models, a fully functional application was developed, integrating **YOLO** for **automatic muzzle detection and cropping**, ensuring consistency and automation in preprocessing. The system incorporates an efficient **data management pipeline**, supporting real-time image storage, retrieval, and identity matching. The final solution is packaged into a **working application** with end-to-end functionality—from image capture to muzzle detection, embedding extraction, and identity recognition. Experimental results demonstrate the system's high accuracy and adaptability across different methods, proving its potential in real-world livestock monitoring and precision agriculture scenarios.

Contents

List of Figures

Chapter 1

Introduction

Accurate and reliable identification of individual cattle is essential for effective livestock management, enabling traceability, health monitoring, and data-driven decision-making. Traditional methods such as ear tags, branding, and RFID systems are widely used, but they are susceptible to tampering, environmental damage, and can sometimes cause discomfort or stress to animals. As a non-invasive alternative, **biometric identification** using **muzzle patterns**—which are unique and consistent throughout a cow's life—has emerged as a promising solution.

This project investigates and implements a comprehensive system for **Cattle Muzzle-Based Recognition**, exploring multiple approaches to develop a robust, scalable, and real-time solution. The project began by evaluating a **classical computer vision technique** using **SIFT (Scale-Invariant Feature Transform)** to extract and match keypoints from muzzle images. While this method provided initial insights, its performance degraded under variations in lighting, scale, and image quality.

To overcome these limitations, the focus shifted to **deep learning-based methods**. A **transfer learning approach** using **EfficientNet**, a state-of-the-art convolutional neural network architecture, was adopted as the primary solution due to its balance of high accuracy and computational efficiency. This EfficientNet-based model was used in the final deployed application to perform real-time identification by extracting discriminative features from automatically detected muzzle regions.

In parallel, a **deep convolutional autoencoder** was developed for unsupervised feature learning. To extend its capabilities to **continual learning**, the model was integrated with **Gradient Episodic Memory (GEM)**, enabling it to adapt to new cattle data over time without forgetting previously learned identities.

To make the system practically usable, a fully functional application was de-

veloped, integrating multiple components into an end-to-end pipeline. The app uses **YOLO** (You Only Look Once) for **muzzle detection and cropping**, followed by preprocessing and recognition using the EfficientNet model. It also includes a robust **data management system** for storing, retrieving, and updating cattle identity information.

This integrated approach—spanning traditional methods, advanced deep learning techniques, continual learning, and system deployment—demonstrates the viability of biometric cattle identification in real-world scenarios. The system holds potential for widespread use in agriculture, helping farmers and organizations monitor and manage livestock with greater accuracy and efficiency.

Chapter 2

System Architecture

2.1 Overview

The system is designed to automate cattle identification using muzzle-based biometric verification via a mobile application. It consists of a React Native frontend, Flask backend, and MongoDB database, integrated with a YOLOv11 detection pipeline and deep learning model.

2.2 Component Description

2.2.1 Frontend: React Native App

- Developed for Android devices.
- Handles user login, image capture, and metadata input (e.g., breed, age).
- Sends captured images and cattle details to the backend via REST API.

2.2.2 Backend: Flask REST API

- Manages authentication and data validation.
- Integrates with YOLOv11 (via Roboflow) to detect the muzzle region.
- Extracts feature vectors using a deep learning model.
- Communicates with MongoDB for storing and retrieving metadata and vectors.

2.2.3 Database: MongoDB with GridFS

- Stores structured data like user credentials and cattle metadata.
- Uses GridFS for efficient storage and retrieval of large image files.
- Links feature vectors to respective cattle records for recognition and matching.

2.2.4 Image Processing Pipeline

- YOLOv11 detects the muzzle region in real time.
- The deep learning model generates feature vectors for comparison and recognition.

2.3 System Flow Diagram

The flow depicted in Figure 2.1 starts with user login, followed by image capture and feature extraction via a deep learning model. Cattle metadata is optionally added. The searching algorithm then matches features for verification or recognition.

2.4 Technology Stack

- **Frontend:** React Native
- **Backend:** Flask (Python)
- **Database:** MongoDB + GridFS
- **Detection:** YOLOv11 via Roboflow
- **Recognition:** Deep learning model

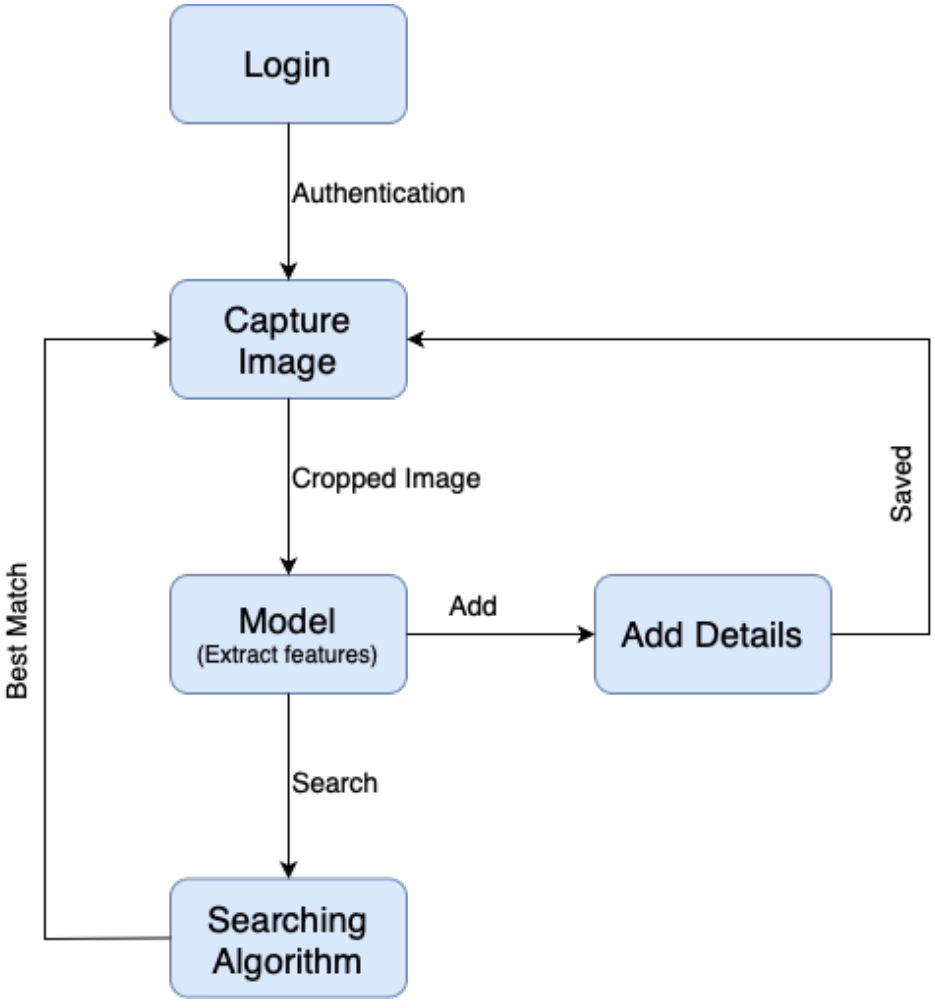

Figure 2.1: System Architecture Flow

Chapter 3

Dataset and Model Training

3.1 Dataset

To develop a robust and accurate model for cow muzzle-based identification, a comprehensive dataset was constructed by combining multiple publicly available sources. The objective of this dataset integration was to ensure diversity in terms of cow breeds, muzzle patterns, lighting conditions, and image quality—factors essential for improving the generalizability and reliability of the model.

The final dataset was created by merging the following resources:

- **Kaggle Datasets**
  - Beef Cattle Muzzle
  - 300 Cattle Muzzle (SimCLR)
  - Cattle Images DB for Muzzle-Based Identification
- **Zenodo Datasets**
  - Zenodo Record 7988559
  - Zenodo Record 6324361

The integration of these datasets resulted in a unified collection that significantly enhances intra-class and inter-class variation. Preprocessing techniques such as resizing, normalization, and data augmentation (including random rotations, zooming, and brightness adjustments) were applied uniformly across all sources to maintain consistency.

This curated dataset forms the foundation for training and evaluating the cow identification model discussed in the subsequent sections.

3.2 Model Training

To build a robust cattle identification system based on muzzle characteristics, four distinct model training approaches were explored. Each method varied in architectural design and learning paradigm, with the aim of identifying the most effective technique for real-world deployment. The selection of models and training configurations was guided by experimental evaluation of their accuracy, inference time, and adaptability to changes in input data.

3.2.1 YOLOv11-Based Muzzle Detection Model

For initial object localization, the system employed YOLOv11 — a fast and efficient object detection model — to accurately detect and crop the muzzle region from input cattle images. The model was fine-tuned using 451 manually annotated images, with annotations formatted according to the COCO standard.

Training was conducted using a COCO-compatible checkpoint, and the model achieved outstanding performance metrics on the validation set:

- **mAP@50:** 99.5%
- **Precision:** 99.9%
- **Recall:** 100%

These results confirm YOLOv11's high accuracy and speed, making it well-suited for real-time muzzle detection. The output cropped muzzle images served as input to subsequent cattle identification modules.

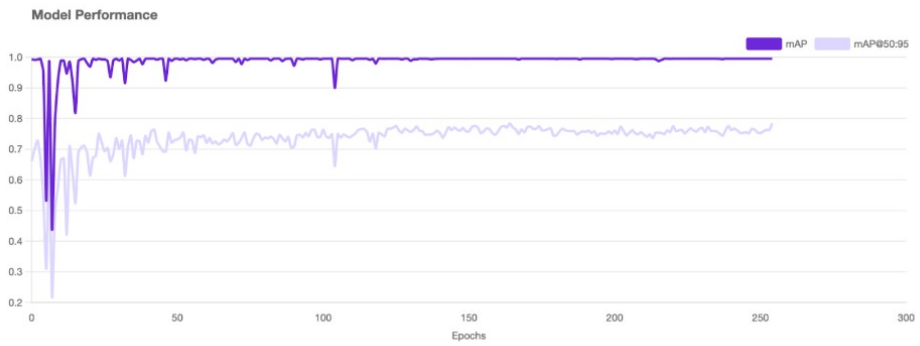

Figure 3.1: YOLOv11 Model Performance Graph

3.2.2 Recognition Model Selection and Justification

EfficientNet-B3 was selected as the feature extraction backbone across all deep learning approaches due to its superior trade-off between computational efficiency and representational power. Extensive benchmarking revealed that EfficientNet-B3 achieved approximately 92% identification accuracy, significantly outperforming other candidates such as ResNet (5.99%) and Vision Transformer (ViT) (7.30%). These results confirm its effectiveness in handling fine-grained muzzle pattern recognition with limited data variation.

3.2.3 Training Approaches

3.2.3.1 Classical Computer Vision-Based Approach

This baseline method employed traditional image processing and keypoint matching techniques. The pipeline consisted of the following steps:

- **Feature Extraction:** Used the Scale-Invariant Feature Transform (SIFT) algorithm to extract local descriptors from muzzle images.
- **Matching:** Employed Fast Library for Approximate Nearest Neighbors (FLANN) to compare descriptors between query and registered images.
- **Registration and Retrieval:** Stored keypoints and descriptors at registration time, which were retrieved for comparison during inference.

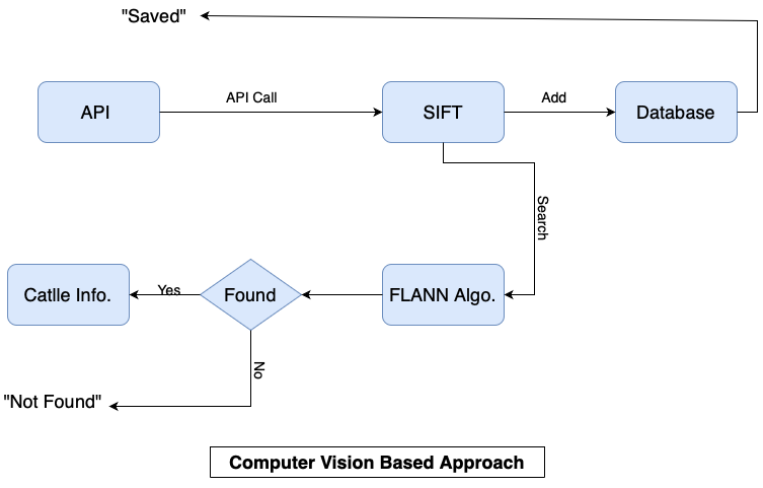

Figure 3.2: Classical Computer Vision-Based Pipeline

While conceptually simple and interpretable, this method was highly sensitive to minor variations in lighting, scale, and rotation, reducing its reliability under real-world conditions.

3.2.3.2 Knowledge Transfer-Based Approach

This approach adopted a transfer learning paradigm, leveraging EfficientNet-B3 as a pre-trained feature extractor:

- **Feature Extraction:** Used the output of the final convolutional layer to produce fixed-length embeddings.
- **Distance Metric:** Applied Mahalanobis distance to compare embeddings during inference.
- **Storage:** Embedded vectors were saved alongside identity labels during registration.

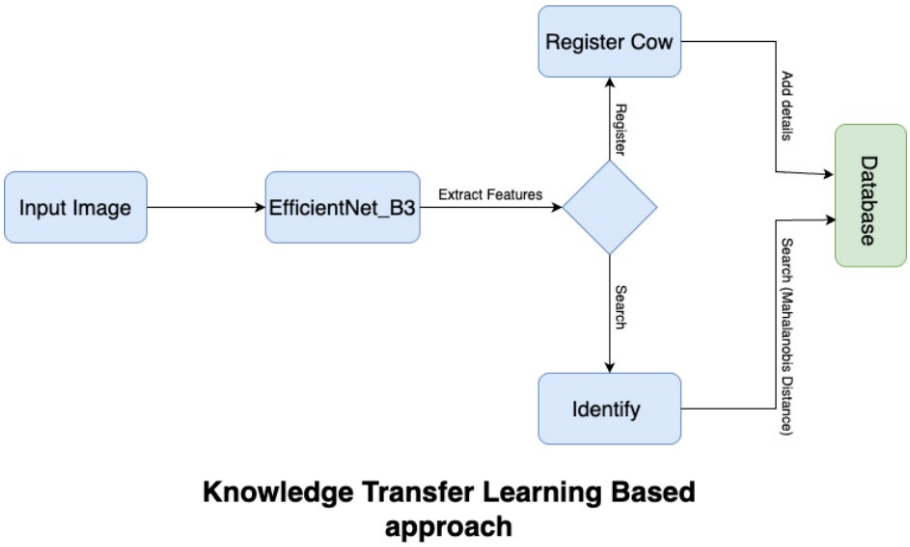

Figure 3.3: Knowledge Transfer-Based Pipeline

This method demonstrated improved robustness and generalization across unseen samples compared to classical techniques.

3.2.3.3 Autoencoder-Based Representation Learning

An unsupervised autoencoder architecture was designed for compact and identity-preserving embeddings:

- **Encoder:** Used EfficientNet-B3 without the classification head to map images into latent embeddings.
- **Decoder:** Implemented a custom CNN to reconstruct images from these embeddings.
- **Training Objective:** Minimized reconstruction loss using Mean Squared Error (MSE).

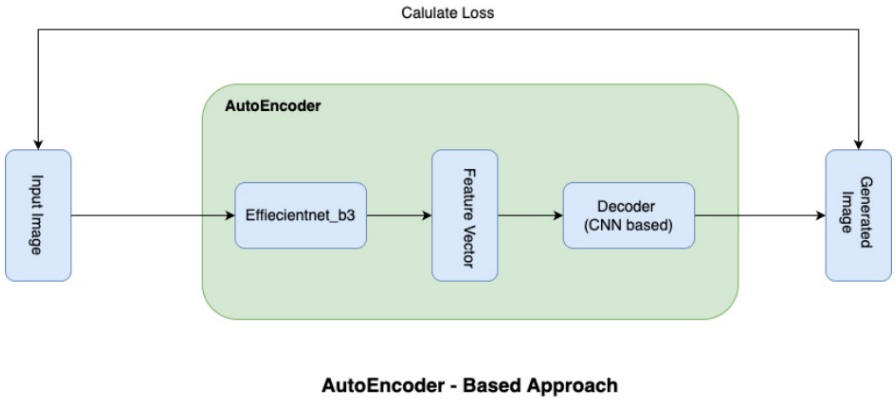

Figure 3.4: Autoencoder-Based Learning Pipeline

This architecture generalized well without requiring labeled data, making it ideal for scenarios where manual annotation is limited.

3.2.3.4 Autoencoder with Continual Learning (GEM)

To enable lifelong learning and prevent catastrophic forgetting, a continual learning setup was employed using Gradient Episodic Memory (GEM):

- **Architecture:** Retained the encoder-decoder structure from the previous autoencoder model.
- **Memory Replay:** Integrated episodic memory to replay previously seen examples.

- **Gradient Projection:** Ensured updates do not interfere with past learning by projecting gradients.
- **Scalability:** Enabled new cattle identities to be incrementally added without retraining the full model.

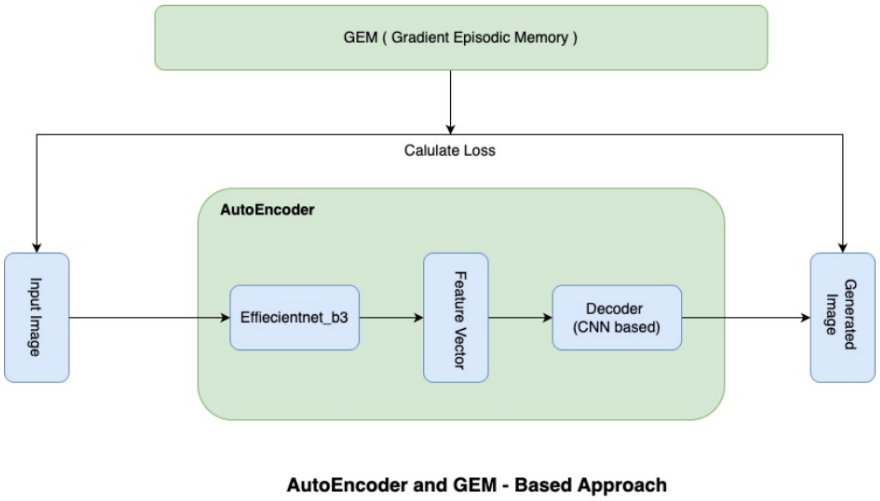

Figure 3.5: Autoencoder with GEM for Continual Learning

This method provided a flexible and dynamic solution suited to real-world farm deployments.

3.2.4 Training Configuration

All deep learning-based models were implemented using TensorFlow 2.19.0 and trained on a workstation with the following specifications:

- **GPU:** NVIDIA RTX 4090
- **CUDA Version:** 11.8
- **Learning Rate:** 0.0001
- **Loss Functions:** Cross-entropy (for classification tasks), Mean Squared Error (for reconstruction)
- **Optimizer:** Adam
- **Batch Size:** Adaptively selected based on available GPU memory

- **Training Strategy:** Early stopping and learning rate scheduling were applied to enhance convergence and avoid overfitting

Chapter 4

Results and Evaluation

4.1 Comparative Evaluation of Recognition Methods

This chapter presents a comparative evaluation of four distinct cattle muzzle recognition approaches:

1. Classical Computer Vision-Based
2. Transfer Learning using EfficientNet-B3
3. Autoencoder-Based Embedding
4. Autoencoder with Gradient Episodic Memory (GEM)

Each method was evaluated using a dataset comprising six cows, with 10 muzzle images per cow for same-day testing and an additional 10 per cow for different-day testing (after 3–4 months). This totals 60 images for each scenario. The best-match strategy was used for identity prediction, and accuracy was reported as the percentage of correctly identified images.

4.1.1 Classical Computer Vision-Based Approach

This method employed SIFT to extract keypoints and descriptors and used a FLANN-based matcher for image comparison.

- **Same-Day Accuracy:** 74.07%
- **Different-Day Accuracy:** 21.67%

Although it achieved reasonable accuracy on same-day data, this method was sensitive to lighting, rotation, and scale changes. It performed poorly on different-day images, highlighting its limited generalization ability.

4.1.2 Transfer Learning-Based Approach (EfficientNet-B3)

This method uses a pre-trained EfficientNet-B3 model to extract deep features. Similarity between images is computed using Mahalanobis distance. Four variants were trained with different combinations of training samples and epochs.

Model	Images/Class	Epochs	Same-Day Accuracy (%)	Different-Day Accuracy (%)
best_model.pth	15	30	100	51.67
best_model-1.pth	15	100	100	51.67
best_model-2.pth	30	100	100	48.33
best_model-2.pth	60	100	100	50.0

Figure 4.1: Transfer Learning Accuracy for Same-Day and Different-Day Testing

Observation: All models achieved perfect accuracy (100%) for same-day images, confirming the discriminative strength of EfficientNet embeddings under similar conditions. However, the accuracy dropped to around 50% for different-day testing, indicating moderate generalizability.

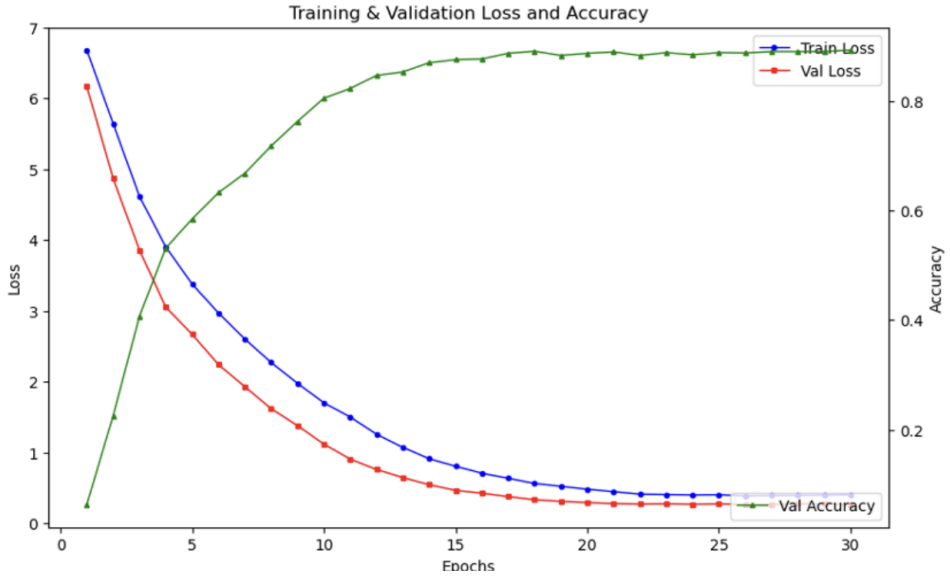

Figure 4.2: Training Loss Curve for `best_model.pth`

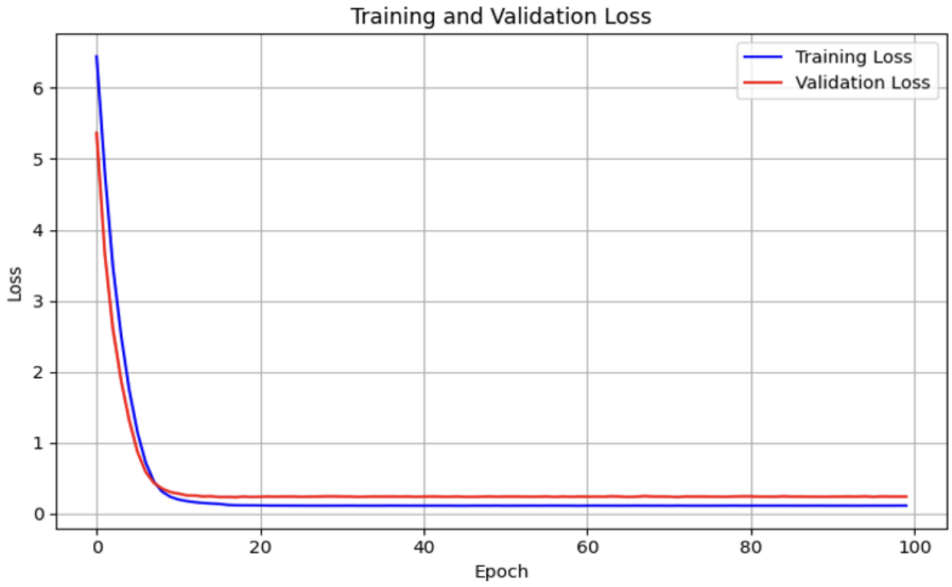

Figure 4.3: Training Loss Curve for `best_model-1.pth`

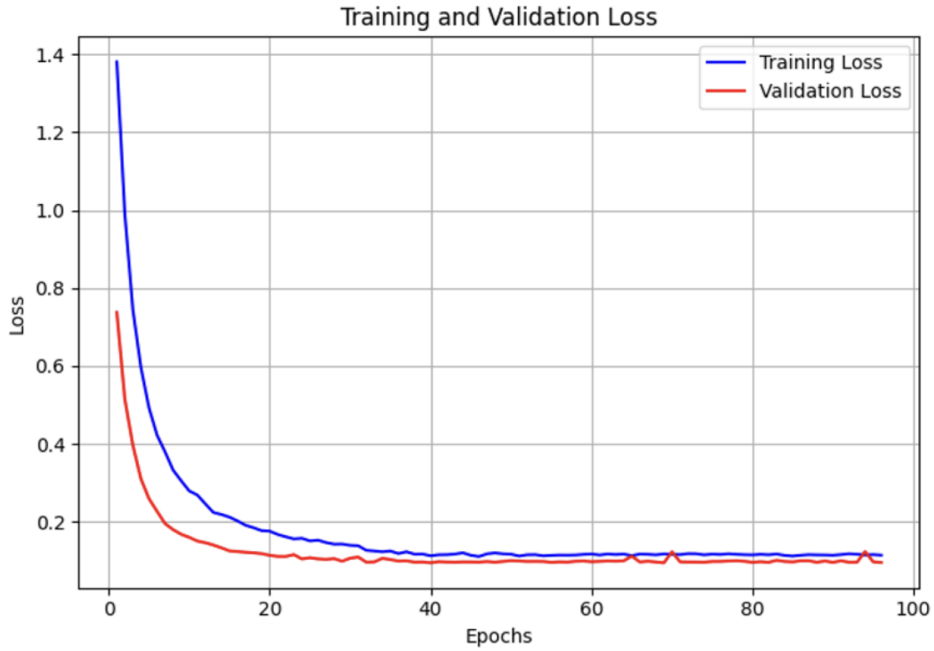

Figure 4.4: Training Loss Curve for `best_model-2.pth`

Training Graph Analysis: As shown in Figures 4.2, 4.3, and 4.4, the training loss curves for the different variants exhibit convergence, suggesting successful feature learning. However, despite the convergence, the limited robustness over time highlights a potential overfitting to session-specific features.

4.1.3 Autoencoder-Based Embedding Approach

In this approach, a convolutional autoencoder is trained to learn compact latent representations. Recognition is performed by computing similarity (cosine distance) between the encoded vectors of query and reference images.

Model	Images/Class	Epochs	Same-Day Accuracy (%)	Different-Day Accuracy (%)
model.pth	15	100	90.57	41.67
model-1.pth	30	100	87.04	28.33
model-2.pth	60	100	94.44	70.0

Figure 4.5: Autoencoder-Based Accuracy

Observation: Among the three variants, `model-2.pth` showed the best performance, especially on different-day testing (70%). The increase in training data significantly improved generalization. However, all models produced very low similarity scores, indicating that the learned embeddings are not highly separable. This suggests the models were not fully trained or optimally tuned, despite the apparent convergence of loss curves (as seen in Figure 4.6–4.8).

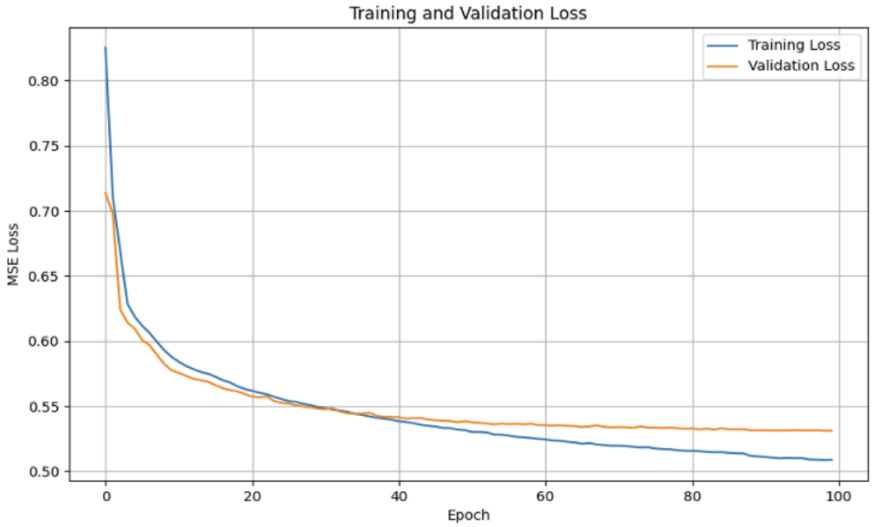

Figure 4.6: Training Loss Curve for `model.pth`

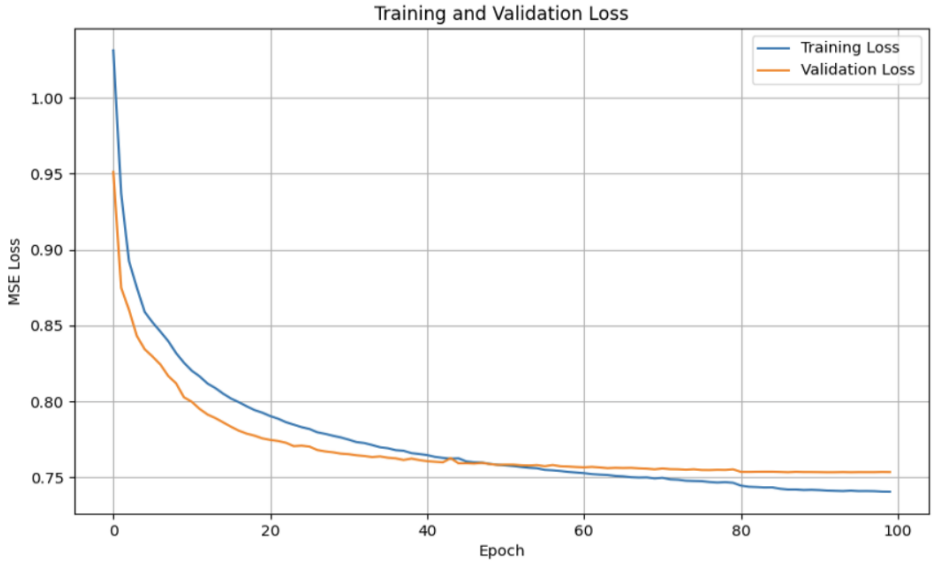

Figure 4.7: Training Loss Curve for `model-1.pth`

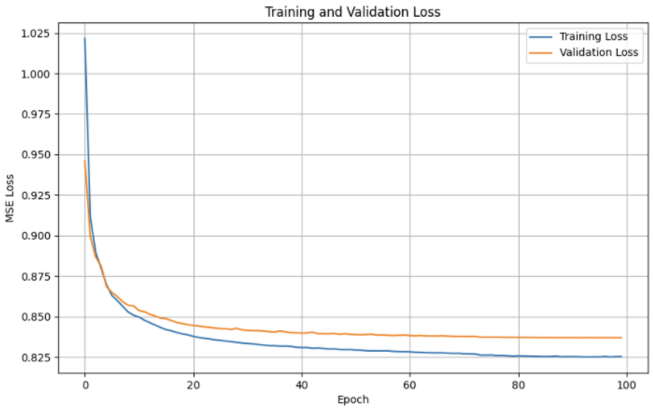

Figure 4.8: Training Loss Curve for `model-2.pth`

Training Graph Analysis: The loss graphs for all three models (`model.pth`, `model-1.pth`, `model-2.pth`) show a clear convergence pattern for both training and validation losses, indicating that the models have learned a compact latent space. Yet, the embeddings did not yield high similarity separability, pointing to limitations in representation learning capacity.

4.1.4 Autoencoder with GEM-Based Continual Learning

This method integrated Gradient Episodic Memory (GEM) with an autoencoder to support continual learning without catastrophic forgetting.

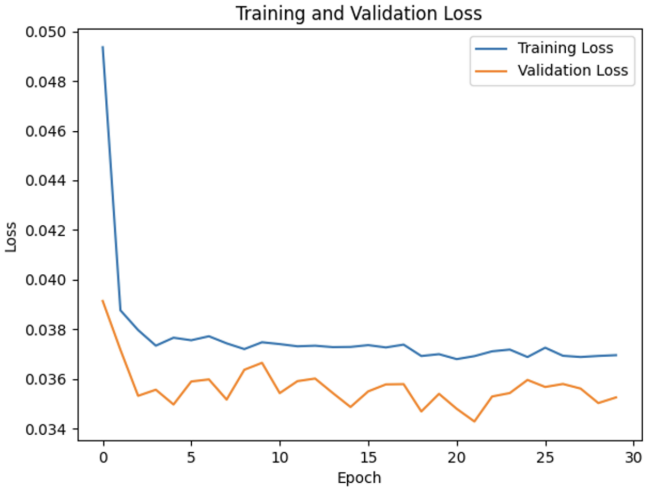

Figure 4.9: Training Graph for Autoencoder with GEM-Based Continual Learning

Observation: This variant underperformed compared to the standalone autoencoder. The lack of significant training progress (Figure 4.9) may be attributed to the GEM algorithm preventing weight updates that conflict with memory constraints, thus inhibiting convergence. As a result, this method is not recommended for this identification task.

4.1.5 Summary of Method Comparison

Method	Same-Day Accuracy (%)	Different-Day Accuracy (%)	Remarks
Classical CV (SIFT+FLANN)	74.07	21.67	Simple and interpretable with no training needed, but lacks robustness.
Transfer Learning (EfficientNet)	100.00	48.33–51.67	Strong for similar-session identification
Autoencoder (AE)	87.04 – 94.44	28.33–70.00	Moderate generalization, low similarity scores
Autoencoder + GEM	N/A	N/A	Learning stagnated because of memory constraints, with no loss improvement due to GEM restrictions.

Figure 4.10: Summary Comparison of All Methods

Chapter 5

Project Overview and Future Scope

5.1 Application Features

The developed application focuses on efficient and intelligent cow management through a user-friendly interface and powerful backend integration. The key features of the app include:

- **OTP-Based Login:** Ensures secure and authenticated access for users using one-time passwords sent via email or SMS.
- **Smart Image Guide and Capture:** Uses YOLO-based object detection to guide users in capturing proper images of cattle muzzles, improving data quality.
- **Live Image Detection:** Real-time image capturing and detection integrated with the camera stream to facilitate quick and accurate data collection.
- **Database Management:** Efficient storage and retrieval of cattle data using a structured backend and database, enabling seamless updates and queries.
- **Model Integration:** Integration of the best-performing model (EfficientNet-based autoencoder) for cattle identification based on muzzle features.
- **Add/Edit Cattle Details:** Users can add new cattle profiles and update existing information directly through the app, promoting easy maintenance of records.

5.2 Challenges Faced

During the development of the system, several challenges were encountered across data preparation, model training, and integration phases:

- **Annotation for YOLO Training:** Annotating muzzle regions in hundreds of images was time-consuming and required precision to ensure training accuracy.
- **Live Image Capturing with YOLO:** Integrating real-time detection with the camera feed posed difficulties due to frame latency, performance tuning, and device compatibility.
- **Continuous Execution Issues:** The application initially required manual intervention after short periods; maintaining continuous, stable runtime was a challenge.
- **Training Instability:** During extended training sessions, the model would sometimes encounter errors or crash after several epochs, particularly due to memory or data pipeline issues.
- **Model Optimization:** Fine-tuning EfficientNet for muzzle recognition required extensive experimentation with hyperparameters and loss functions.
- **Device Constraints:** Running real-time detection and inference on low-resource or mobile devices led to performance and memory bottlenecks.
- **Dataset Imbalance:** The distribution of muzzle images across different cows and lighting conditions was inconsistent, affecting model generalization.

5.3 Future Scope

Several areas have been identified for improving and extending the application in future iterations:

- **User Interface Enhancements:** Update the UI for better usability, responsiveness, and modern design to improve user experience.
- **Additional Features:** Add functionalities such as health tracking, location tagging, alert systems, and report generation to extend the app's utility.

- **Dataset Expansion:** Collect more diverse and high-quality cattle muzzle images under varying conditions (lighting, angle, time of day) to improve model robustness.
- **Model Experimentation:** Explore alternative architectures like Vision Transformers, ResNet variants, or lightweight models for better accuracy and efficiency.
- **Generalization Capability:** Develop methods to ensure the model performs reliably across different days, lighting conditions, and camera angles.
- **Offline Functionality:** Introduce offline modes with synchronization options to support use in remote or low-connectivity areas.
- **Multi-Animal Support:** Extend the system to support identification and management of multiple animals per image or session.

5.4 Contributions

The successful completion of this project was made possible through the collaborative efforts of the team members, each contributing to distinct but complementary aspects of the system:

- **Gyanendra Mani** was primarily responsible for the complete application development, encompassing frontend and backend design, user interface creation, and database management. He successfully integrated YOLO-based real-time image capturing and implemented core functionalities such as OTP-based login, cattle data entry, and edit features. In addition, he contributed to several critical development tasks to ensure smooth functionality and user experience of the app.
- **Nitesh Kumar** focused on the machine learning and model development aspect of the project. His contributions included designing and experimenting with various model architectures, particularly leveraging autoencoders and EfficientNet for feature extraction. He was responsible for curating and managing the dataset, performing data augmentation, and optimizing training across multiple configurations including variations in batch size and number of epochs. Furthermore, he validated the model performance using a specialized testing dataset provided by the National Dairy Research Institute (NDRI).

Chapter 6

Conclusions

This project aimed to develop a robust and intelligent cow management system leveraging deep learning techniques for accurate and efficient identification of cows based on their muzzle features. The proposed solution integrates multiple technological components, including OTP-based secure authentication, YOLO-based real-time image detection and guidance, structured database management, and the deployment of an EfficientNet-based model for feature extraction and identification.

The application offers a seamless user experience for capturing images, managing cow records, and performing recognition tasks with a high degree of automation and reliability. The integration of real-time camera support and smart image capturing not only improves usability but also enhances the quality of input data, which is critical for achieving high model performance.

During the development process, several technical challenges were encountered. These included manual annotation of images for YOLO training, integration of live image capture with detection modules, maintaining continuous system execution, and addressing training stability issues such as errors occurring after multiple epochs. Each challenge was systematically addressed through rigorous testing, optimization, and modular system design.

This work demonstrates the practical feasibility of applying computer vision and deep learning in livestock management, highlighting both the potential and the limitations of the current implementation. The proposed future work includes enhancing the user interface, expanding the dataset for improved model generalization, integrating additional features, and exploring alternative model architectures beyond EfficientNet to further optimize performance.

In conclusion, the project successfully meets its objectives by providing an end-to-end solution for cow identification and management. It establishes a strong foundation for future research and development in this domain and presents significant opportunities for real-world deployment in agricultural and veterinary

settings. With continued refinement, the system has the potential to contribute meaningfully to modernizing livestock management practices through AI-driven solutions.

Referance

1. “Unsupervised Deep Learning Image Verification Method” [PDF]
2. “Deep Learning Framework for Recognition of Cattle Using Muzzle Point Image Pattern”
3. “Deep Transfer Learning-Based Animal Face Identification Model Empowered with Vision-Based Hybrid Approach”
4. “Gradient Episodic Memory for Continual Learning” [PDF]
5. “Unsupervised Learning Approaches to Facial Recognition Using Autoencoders”
6. “Autoencoder as Feature Extractor for Face Recognition System”
7. “Autoencoder Based Face Verification System” [PDF]
8. “Automated Muzzle Detection and Biometric Identification via Few-Shot Deep Transfer Learning of Mixed Breed Cattle” [MDPI]
9. “Identification of Individual Hanwoo Cattle by Muzzle Pattern Images Through Deep Learning” [MDPI]
10. “Cattle Identification Using Muzzle Images and Deep Learning Techniques” [PDF]