International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 42, No. 3, June, 2025, pp. 1421-1435
Journal homepage: http://iieta.org/journals/ts

Real Time Disease Detection for Cattles and Pets and Tool for Veterinary Assistance and Farmers

Shivappriya Sathyamangalam Natarajan[1], Gowtham Guruvayurappan[1], Jeniferraj Jeyaselvarayan[1], Guruprasath Lakshmikanth[1], Daniela Danciulescu[2], Gabriel Stoian[2], Anitha Jude[3*]

[1] Department of ECE, Kumaraguru College of Technology, Coimbatore, 641049, India
[2] Department of Computer Science, University of Craiova, Craiova 200585, Romania
[3] Department of ECE, Karunya Institute of Technology and Sciences, Coimbatore 641114, India

Corresponding Author Email: anithaj@karunya.edu

https://doi.org/10.18280/ts.420317

Received: 28 January 2025
Revised: 4 March 2025
Accepted: 10 April 2025
Available online: 30 June 2025

Keywords:
convolutional neural network (CNN), Raspberry Pi, TensorFlow Lite (TFLite), cattle and pets, local computing

ABSTRACT

This work helps veterinarians and farmers in predicting skin diseases of cattle and pets. A real-time skin disease detection device designed to assist veterinary doctors and farmers by providing rapid and reliable identification of common skin diseases in cattle and pets. The device integrates a Convolutional Neural Network (CNN) deep learning model deployed on a Raspberry Pi, which is both cost-effective and suitable for on-site usage. The camera module attached to the Raspberry Pi captures images of the animal's skin, and the model trained in TensorFlow Lite (TFLite) is optimized for efficient processing of these images locally. The predictions are immediately shown on an attached 16×2 LCD screen, which allows for fast assessment without the need for Internet connectivity. This fast tool supports prompt disease detection and intervention, thus empowering veterinary practitioners and farmers to better manage animal health in far-flung and rural areas.

1. INTRODUCTION

The livestock sector accounts for nearly 24.72% of the agricultural gross domestic product and nearly 4.36% of the national gross domestic product for this country. About 22 million jobs depend directly and indirectly on these people and their livestock-related occupations, which consist of dairying, poultry, and the meat sectors, with over 500 million animals such as buffalo and cattle, sheep and goats, and also domesticated pigs. Livestock diseases remain one of the biggest economic and productivity challenges to date, having taken an estimated loss of $4.45 billion each year through reduced productivity, increases in mortality, and sky-high treatment costs. Still, the negative supply chain effects, along with their demands, further complicate such financial burdens for both farm owners and pet lovers.

In Rajasthan, which produces 15% of India's overall milk, the major making up of indigenous breeds is provided by local cattle and buffaloes at 83% of the livestock. Even though crossbreeds have not yet gained wider acceptance, remarkable growth in dairy productivity has been reported in Rajasthan, mainly through selective breeding of indigenous animals. This achievement underscores the possibility of sustainable breeding programs based on the resilience of indigenous livestock to local conditions in resource-poor settings arising from rapid urbanization and climate change. Indian pets also suffer from several diseases. These can become more expensive to treat, or even deadly if not recognized early. Common conditions such as Flea Allergy Dermatitis, Hot Spots, and Pyoderma severely affect the welfare of the animals and lead to the unnecessary suffering that arises from delayed diagnosis.

The above challenges shall be addressed through the development of this real-time skin disease detection device with the improved early diagnosis and intervention of livestock and pets. The device powered by a deep learning model on a Raspberry Pi captures images of animals' skin through a camera, processes them via a TensorFlow Lite (TFLite) model, and promptly displays predictions on an attached LCD screen. It will help farmers and veterinary practitioners make decisions in time so that losses related to such diseases can be avoided by providing quick, reliable results directly in the field.

2. LITERATURE REVIEW

In 2022, Lake et al. [1] presented a novel diagnostic system using expert deep learning approach and image processing technology to diagnose diseases in cattle. Here, the captured symptoms by camera-enabled smartphones which would allow entry of texts would indicate palpable symptoms. These can deliver rapid correct diagnoses based on visual features from the deep-learning convolutional neural network (CNN). The evaluation confirmed the tool to be a reliable and essential tool for cattle disease, thus ensuring more effective management of diseases and creating opportunities for swift economic benefits in livestock farming by providing early detection and prevention.

Das et al. [2] introduced 'CattleSavior,' a Raspberry Pi-based system for detecting cattle diseases including Foot-and-Mouth Disease (FMD) and mastitis. The system comprises multiple sensors: temperature, rumination, and motion installed on various parts of the cow's body, which are all connected to a central hub that is also a Raspberry Pi. The data collected from these sensors are processed in the Cloud, while Raspberry Pi is used as the processing unit. Cows are identified using RFID tags, with real-time analysis of their health. All this will be made achievable together by the use of machine algorithms in the Raspberry Pi microcontroller.

The developed CNN-based system of Permana et al. [3] on cattle disease classification in Semin District was based on the diagnosis of BEF, Mastitis, and Scabies. From the set of training and validation images, the study showed that this model gave an accurate identification of these cattle diseases. Precision and recall scores further proved the effectiveness of the system in the correct diagnosis of the diseases. This research brings out the capabilities of CNNs in cattle health management, providing practical tools to farmers for improved animal welfare and reduced economic losses resulting from disease.

Kim et al. [4] proposed a CNN framework for automated diagnosis of canine ulcerative keratitis, trained on a curated dataset of annotated corneal images. The proposed models were intended to train GoogLeNet, ResNet, and VGGNet: The proposed models achieved up to 90% and above accuracy for classifying both normal, superficial, and deep corneal ulcers. High-resolution images explain the high accuracy achieved through this study; however, the study states that at the cost of image quality, there is potential limitation to its clinical application. Thus, the present study was able to exhibit the versatility of CNNs in veterinary ophthalmology for grading the severity.

Rony et al. [5] presented a model that made use of CNN architectures, Inception-V3, and VGG-16 to classify cattle diseases like FMD, LSD, and IBK with 95% accuracy. The CNN-based approach heavily helps in early diagnosis and reduces the involvement of human beings, thus enabling veterinarians and farmers to manage and identify contagious cattle diseases effectively. Gauswami et al. [6] have proposed a gender detection system recently that has employed a Raspberry Pi module to make low-cost, compact, small-sized platform-based deployment with a CNN model. Its module will help extract face features and classify them in real-time on the same device, hence making it very accessible with increased efficiency. This has made the gadget portable so that real-time detection might be deployed at places using surveillance, robotics, among many other works. Combining low cost and flexibility of Raspberry Pi with the capabilities of deep learning gives an efficient and powerful tool for real-time applications.

Gasa et al. [7] proposed a Raspberry Pi module skin disease detection system with MobileNet CNN to perform an efficient classification of skin lesions. The technique uses Depthwise Separable Convolution to process images on reduced computing resources so the predictions are highly accurate and delivered through a chatbot on Telegram to make things very user-friendly. It enables remote users to capture and analyze the skin lesions toward early detection of cancer. The system interface is user-friendly. The accessible and practical support available for dermatology ensures timely, professional advice to the need of the user. Besides, human-centered design with advanced technology improves the monitoring and intervention effectiveness concerning skin diseases.

Hwang et al. [8] have proposed a classification system of dog-associated infections, which could lead to over 70 human diseases. The normal and multispectral images are gathered in this study, from which models for identification are proposed based on different CNN architectures: InceptionNet, ResNet, DenseNet, and MobileNet. These findings mean that the system with CNN can identify, on the basis of an image, dog skin diseases such as Ringworm, Fleas, and Mange, thus eliminating the need for extensive testing. Further, this same technology's real-time application through the use drones or cameras will enable street dogs to avoid disease spread.

Kim et al. [9] described a mobile-based pet dog disease pre-diagnosis system utilizing the Possibilistic C-Means (PCM) clustering algorithm for unsupervised learning. It employs a comprehensive disease-symptom database, built from textbooks and verified by veterinarians, to identify probable diseases based on user-input symptoms, enhancing health monitoring and caregiver awareness without replacing veterinarians. Inovero et al. [10] introduced a mobile diagnostic application for cats and dogs using a Neuro-Fuzzy Algorithm. Developed on Android Studio, the app achieves an 88.5% accuracy rate in identifying diseases based on symptoms, providing treatment recommendations. This system enhances disease detection, aids caregivers, and supports veterinarians in offering accurate and efficient pet health care.

Singh et al. [11] investigated Lumpy Skin Disease Virus (LSDV) in cattle, which causes severe symptoms, economic loss, and affects global livestock sectors. Using machine learning techniques (LRC, DTC, RFC, XGBC, SVC), the study predicts LSDV with high accuracy. Among evaluated classifiers, the Vector Classifier demonstrates superior performance in metrics like F1-score and accuracy.

Raj et al. [12] addressed Lumpy Skin Disease (LSD), a contagious viral disease in cattle caused by the Neethling virus, emphasizing its economic impact and the need for precise detection. It proposes a deep learning model combining ResNet-50 and VGG-19 for feature extraction, reducing feature dimensions using PCA, and employing classifiers like Naive Bayes, Decision Tree, Random Forest, and KNN for disease classification. The model achieves 99% accuracy, outperforming existing approaches, highlighting its effectiveness in timely detection and minimizing economic losses.

Mujahid et al. [13] utilized image datasets to implement Local Interpretable Model-Agnostic Explanations (LIME) for classifying skin diseases. By applying LIME, the method identified image super-pixels significantly influencing predictions, achieving classification accuracies of 92.5% and 97% on test datasets through a range of classifiers, including deep learning models.

Shivappriya et al. [14] investigated the application of deep learning models, including DenseNet, ResNet, AlexNet, and Inception V3, for diagnosing Diabetic Retinopathy (DR) through retinal images. The workflow involves capturing retinal images, preprocessing them via segmentation and enhancement, and classifying disease severity using trained models. DenseNet is particularly noted for its high accuracy in predicting DR progression and identifying various stages. The study emphasizes the role of high-quality datasets and robust model training in achieving reliable and efficient diagnostic results. Compared to traditional manual methods, these deep learning approaches automate DR detection and classification, improving accuracy, efficiency, and progression prediction.

Saha [15] projected that the Lumpy LSD is a serious concern for cattle, affecting both milk production and fertility. This study investigates the early detection of LSD using deep learning techniques, specifically a newly proposed CNN architecture known as MobileNetV2, which boasts an impressive 96% classification accuracy. Key methodological steps included image preprocessing, segmentation, and feature extraction from 840 images of healthy and LSD-affected cows. MobileNetV2 outperformed other models, like DenseNet201 at 94% accuracy and traditional machine learning methods like SVM, which only achieved 78%. The research highlights the potential of deep learning, particularly MobileNetV2, in enhancing cow health management and suggests future exploration of transformer-based models. Overall, deep learning surpasses traditional methods in identifying Lumpy Skin Disease effectively.

Ghosh et al. [16] introduced a real-time system for detecting bovine mastitis using deep learning (Inception V3) and machine learning (Random Forest), achieving 99.34% and 99% accuracy, respectively. It processes image and numerical data collected via sensors and cameras on edge devices (Raspberry Pi) integrated with cloud computing. This system aims to minimize economic losses, improve treatment efficiency, and support livestock health, particularly benefiting the dairy industry in Bangladesh and developing countries.

Worldwide, the most contagious illnesses affecting cattle are infectious bovine keratoconjunctivitis (IBK), foot and mouth disease (FMD), and lumpy skin disease (LSD). Controlling these illnesses requires early diagnosis. Using a variety of CNN architectures, including the traditional deep CNN, Inception-V3, and VGG-16 in the field of deep learning, this suggested model aims to identify the most prevalent external diseases early. With a 95% accuracy rate, the suggested technique is proven to be efficient and might potentially decrease human error in the identification process. It will also help veterinarians and husbandry farmers identify infections [17].

Wang et al. [18] conducted a study using deep learning to recognize and classify broiler droppings for detecting digestive diseases in poultry. Two advanced CNN models, Faster R-CNN and YOLO-V3, were implemented. Faster R-CNN, using ResNet as the backbone, showed high accuracy with a recall of 99.1% and mAP of 93.32%. YOLO-V3, based on Darknet-53, offered faster detection with a recall of 88.7% and mAP of 84.25%. YOLO-V3 was improved using K-means++ to optimize anchor boxes. This work helps farmers and poultry workers by automatically identifying sick birds through droppings, allowing early treatment and reducing the need for manual checks.

Xiao et al. [19] explained how deep learning is being used to improve animal health and disease diagnosis. Deep learning, especially using models like CNNs, helps vets analyze images such as X-rays, MRIs, and ultrasound scans to detect diseases in animals like dogs, cats, cows, and horses. These models can spot problems like heart disease, tumors, and kidney issues more accurately and faster than humans in some cases. The study also shows how deep learning works with data from sensors, medical records, and even smartphone photos, helping vets diagnose and treat animals earlier and more effectively.

Olaniyan et al. [20] explored how deep learning can help predict LSD in cattle. They developed an improved Artificial Neural Network (ANN) model that was trained over 200 cycles, reaching a high accuracy of 98.89% during training and 98.66% in testing. The researchers also compared this with a combined model (called a stacked ensemble) made up of different machine learning methods like Decision Trees, K-Nearest Neighbors, Random Forest, and SVM. Both models performed very well, but the ANN was slightly better. This approach shows how deep learning can support early disease detection in animals and help improve veterinary care.

3. METHODOLOGY

The methodology of developing the system for the detection of cattle and pet skin diseases is structured, starting from the collection of more than 1,200 images across five classes of diseases. Data augmentation was put in place to increase diverse training samples in an attempt to reduce overfitting as well as improve the model's accuracy. A CNN model was created based on these images to be used for the purpose of analyzing images and classifying the disease. The trained CNN model is then converted to TFLite format, and this assists in the effective deployment on the Raspberry Pi. Along with the camera installed to capture the images and LCD attached will show the result, this device runs the TFLite model and makes predictions of diseases in real time. The model processes the captured image, and the disease class appears on the LCD. Substantial testing validated that the predictions on the Raspberry Pi were identical to those produced by the initial model and thus reliable. This hand-held device provides veterinarians and farmers with an instantaneous, on-site diagnostic tool to better animal health outcomes by making it possible to quickly diagnose and treat disease. The block diagram of the device is shown in Figure 1.

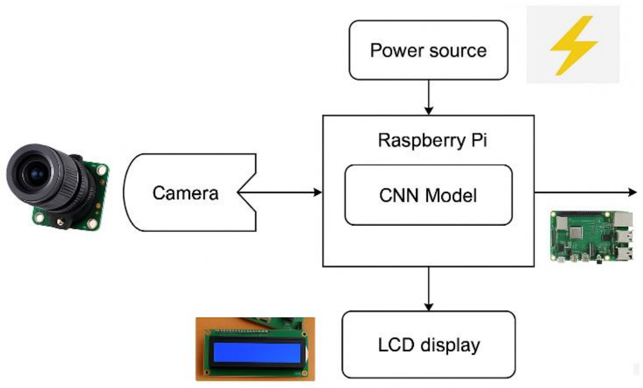


Figure 1. Block diagram of the device

4. DATASET PREPARATION

The dataset initially contained over 600 images across five skin disease classes: Lumpy Skin, Flea Allergy, Hotspot, Mange, and Ringworm. Images were organized by class in separate folders, with each disease assigned a numerical label for consistent classification. To improve the model's accuracy and help it perform well on new data, we used several data augmentation techniques with TensorFlow's Image Data Generator. These included random image rotations (up to 25 degrees), horizontal flips, zooming in by up to 20%, and adjusting brightness and contrast (brightness range: 0.8 to 1.2). These steps helped double the number of images to 1,208.

Although more advanced methods like Generative Adversarial Networks (GANs) were considered, we chose not to use them at this stage to keep the model simple and easy to understand. We also added Dropout layers with a rate of 0.5 after the fully connected layers to avoid the model becoming too dependent on certain neurons. L2 regularization was also considered to reduce model complexity and prevent overfitting.

The dataset was then split 80-20 as the training and validation set, and the images were resized to ensure a stable training process, along with normalizing them in the case of constant input dimension. Techniques such as class weighting have been carried out to ensure that equal representation between samples is maintained so that the dataset obtained is diversified and best prepared for the training process, so that the CNN model would learn effectively in order to recognize diseases related to the skin, be it of cattle or pet animals.

We had a variation in dataset-class sample size, for example, there were only 76 images of Flea Allergy, while Ringworm had 138. To fix this class imbalance, we used class weighting during training so that the model paid more attention to the smaller classes and didn't ignore them. We also looked into oversampling, which means adding more examples to the smaller classes by creating new, similar images. In the future, we plan to explore other methods like reducing samples from large classes, using advanced techniques like GANs to generate new images, and trying different loss functions like focal loss to make the model more balanced and reliable. In Table 1 represents the summary of the number of dataset images used for each class of disease, and sample images of the dataset are shown in Figures 2 and 3.

Table 1. Dataset preparation

Skin Diseases	Dataset Images	After Augmentation
Lumpy skin	324	324
Flea allergy	76	152
Hotspot	96	192
Mange	63	126
Ringworm	138	414
Total	697	1208

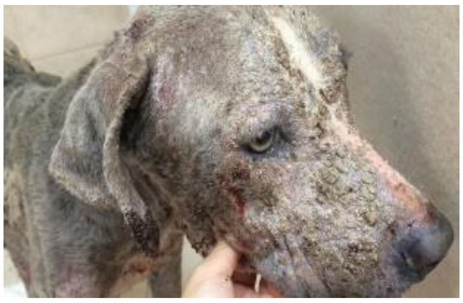


Figure 2. Dataset image (Mange)

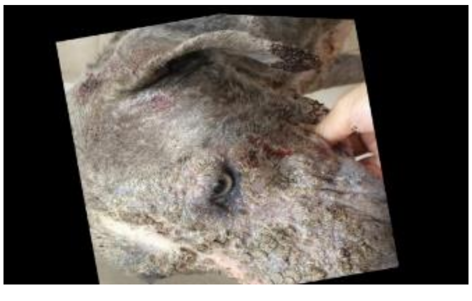


Figure 3. Augmented dataset image (Mange)

5. THE APPLICATION OF CNNS IN IMAGE RECOGNITION AND PROCESSING

CNNs are highly specialized Artificial Neural Networks (ANN). They have greatly impacted the emerging world of computer vision. CNNs are widely recognized for their feature extraction capabilities, depending on the type of image data, and are most commonly used in the recognition and processing of images. Multi-layered images are supposed to be captured for treatment when handling data at each layer in the CNN architecture by certain filters; those are called filters or convolutional kernels designed to capture special features of images like edges, corners, and textures. Putting all of these together, the major advantage CNNs have over a standard neural network is that they learn and extract fine details from the data very efficiently and hence are quite useful in different computer vision applications. The strength of CNNs feature extraction, which means object location in an image, face recognition, and category of images.

A typical CNN architecture consists of feeding an input image through several convolutional layers. In such a layer, it will work on filters and produce corresponding feature maps associated with patterns and features that exist in the input image. To inject nonlinear relationships, the model's nonlinear transformations for each output of a convolutional layer should be provided by applying a nonlinear activation function, such as ReLU. It can also hold one or more fully connected layers that make use of feature for classification, regression, etc. The output may also be the last layer, which classifies the objects inside an image. The ability of CNNs to learn and recognize complex features in images has made them the choice for most computer vision applications and delivers top performance in tasks such as facial recognition, object detection, and image classification. All of these applications are applied to self-driving cars, medical image diagnostics, and security surveillance.

Identification and classification of a CNN usually involves the following steps:

1) Collection of data
2) Structure of CNN
3) Building the network of CNN
4) Calculating weight and bias
5) Training the network
6) Testing the performance of the network based on gradients

Hidden layers are the features with which CNNs differ since they extract detailed features from input data, and improvements in the accuracy of classification. As the data streams through all these hidden layers, the filtering differs due to its different filters and identifies many aspects of the input stream differently.

The simplest edges or lines are all that earlier layers can pick up; however, more complex shapes or textures can be caught by later layers. Therefore, more complex representations of the input will be produced by the network combining outputs coming from several hidden layers that classify better. The number and size of hidden layers depend on the application and complexity of the input data.

5.1 Overall architecture

A CNN performs specific operations like recognizing an image, image processing, or similar types of processes. Its structure consists of three primary types of layers: convolutional, pooling, and fully connected layers. To take it one step further:

1) Input Layer: The input layer is an entry layer that represents the pixel values of the input image, just as in other types of neural networks.
2) Convolutional Layer: In this layer, every neuron takes the dot product of its learned weights and a region of the input data and passes this through a combination with a learned bias plus an activation function, usually ReLU (Rectified Linear Unit), introducing non-linearity based on the activations of the previous layer.
3) Pooling Layer: It reduces the spatial dimensions of input data to make fewer parameters and computations with incurring least information loss.
4) Fully Connected Layers: These layers essentially work like in a standard neural network to generate class scores based on the activations to classify the input. ReLU activations can be added here as well to enhance performance.

In this way, it makes a CNN highly effective in analyzing image data, defining intricate patterns, and then identifying them, hence forth to be used in all forms of image-based tasks. Figure 4 represents the architecture of CNN.

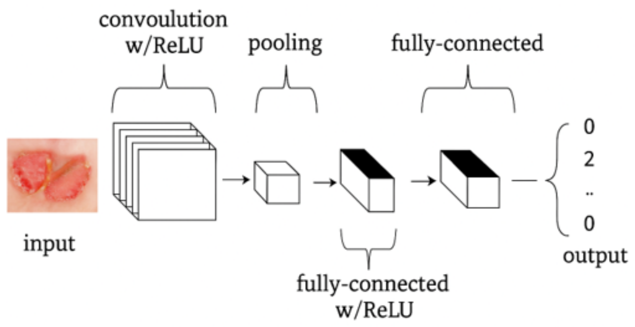


Figure 4. CNN architecture

5.1.1 Convolution layer

The convolutional layer is very important in the analysis of image datasets, for example, those that include images of cattle and pet skin diseases. In this case, learnable filters or kernels are slid across the depth of each input image to detect specific patterns and features. As these filters move spatially across the image, they generate 2D activation maps, capturing locations in the image where particular characteristics, like textures or edges, are detected.

If an input is a skin disease image with size 64×64×3 for RGB color, then a receptive field of 6×6 would give each neuron 108 weights (6×6×3). Tuning parameters like depth, stride, and zero-padding also controls the complexity of output but maintains the detail in the features. Stride defines how many pixels a filter moves in the input in each step. This further makes the resolution of the output, depending on the stride, in addition to zero-padding for retaining the original input dimensions or the output size. The number of zero-padding is found by the formula:

$$V=(W-R+2Z)/S+1 \tag{1}$$

From Eq. (1), V is the output size, W is the input size, R is receptive field size, Z is zero-padding, and S is stride.

Parameter sharing has the effect of reducing weights since a single set of weights is used across the different regions in the activation map, which lowers the amount of computation required and even allows for efficient back-propagation through shared parameters.

It makes possible for the convolutional layers to extract skin diseases at any different locations within images. Further, the convolutional layer keeps its simplicity because its model can easily process the information to get insights in analyzing feature aspects from a disease image, hence they have potential uses when detecting the occurrence of a disease among an animal set. Figure 5 represents the convolution layer of CNN.

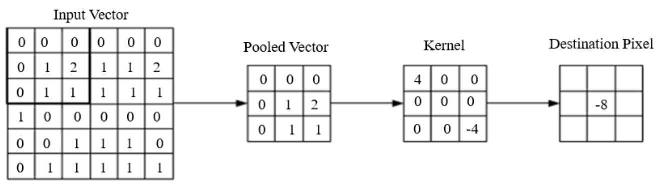


Figure 5. Convolution layer

5.1.2 Pooling layer

As illustrated in a CNN utilized with data of images of disease for cattle and pets, it reduces the dimension of the feature maps through which a reduction in the size is made concerning the complex nature of computations and that associated with model parameters. Within several methods of down sampling, max pooling can also be found to occur primarily because this technique offers possibilities of keeping key information where the data is under reduced sampling.

It usually involves a max-pooling layer of the size 2×2 with a stride of 2. It reduces all spatial dimensions of each feature map to 25% of the original size and retains the depth. The downsizing makes the processing of images in the CNN more efficient, limits the number of parameters that would need training, and prevents overfitting. It is just another way the pooling layers increase the CNN's efficiency and performance due to down-sampling from the feature maps of skin diseases in cattle and pets. Figure 6 represents the Pooling layer of CNN.

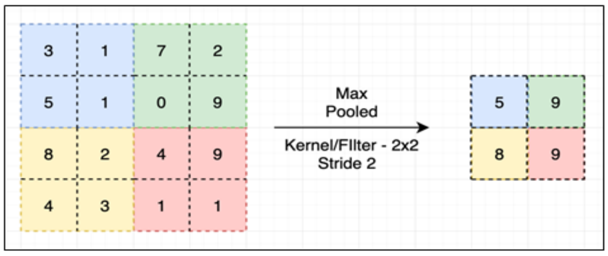


Figure 6. Pooling layer

5.1.3 Fully connected layer

The main features of the input image are extracted with convolutional layers combined with pooling layers followed by flattening into a vector before further processing takes place within a fully connected layer where every neuron of this fully connected layer has a direct connection to every neuron within adjacent layers. This flat vector as (×1, ×2, ×3,….,×n) is passed to the fully connected layer which aggregates all the features together and produces the output of classification.

Output of the fully connected layer Mathematically defined by:

$$z=w\times x+b$$

Here, weights are denoted as (w), the input vector as (x), and the bias term as (b).

The final output is further passed into some activation function perhaps SoftMax or Sigmoid, it then produces probability scores over all categories of diseases, hence classify with the highest-probability class. So that the whole connection layer refines the extracted features as well as provides proper adequate translation in the correct categorization of cattle and pet skins.

5.2 CNN architectures in edge devices

CNN models used on edge devices are made to be fast, small, and power-efficient while still giving accurate results. These models start by taking an image as input, then pass it through layers that find important features like colors, shapes, or spots. To keep things lightweight, models like MobileNetV2 use special types of layers that do less work but still learn well. Instead of using large, slow layers at the end, they use simple techniques like averaging to make final predictions. The models are also made smaller using tricks like quantization (shrinking the numbers) and pruning (removing unused parts). These optimized models run smoothly on small devices like Raspberry Pi, Jetson Nano, or mobile phones, making them perfect for tasks like detecting plant or animal diseases without needing the internet or a powerful computer.

5.2.1 MobileNetV2

MobileNetV2 is a light and fast deep learning model made for small devices like mobile phones and Raspberry Pi. It's designed to use less memory and processing power while still giving good results. It works well for tasks like recognizing diseases in plants or animals, even without an internet connection. MobileNetV2 is a great option when you need quick and efficient predictions on low-powered devices, making it perfect for real-time use in agriculture or veterinary care.

5.2.2 ResNet50

ResNet50 is a more powerful model with 50 layers that can recognize complex patterns in images. It's great for detecting diseases or doing tasks that need very accurate results. However, it's heavier and needs more processing power than MobileNetV2, so it's better suited for stronger edge devices like NVIDIA Jetson or other AI hardware. To make it work better on small devices, it can be simplified using techniques like pruning and model compression.

5.2.3 EfficientNetB3

EfficientNetB3 is a smart and balanced deep learning model that offers high accuracy without using too much power. It improves performance by adjusting how deep, wide, and detailed the model is. It's perfect for edge devices that are a bit more powerful, like Google Coral or Raspberry Pi 4 with AI support. This model works really well for tasks like medical image analysis or detecting plant diseases, making it a great choice when both speed and accuracy are important

When using deep learning on edge devices, it's important to pick the right model that gives a good mix of accuracy, speed, and fits the device's limits. MobileNetV2 is great for smaller devices because it's light and fast, making it perfect for tasks like detecting plant or animal diseases in real time. ResNet50 is more powerful and accurate but needs stronger devices and some extra tweaking to run well. EfficientNetB3 gives the best mix of speed and accuracy by using smart design, making it ideal for slightly more powerful edge devices. Together, these models help bring smart AI to small, offline systems.

5.3 Training and validation

After training for 30 epochs, the CNN model reached a high training accuracy of 98.86% and a validation accuracy of 76.15%, showing that it can handle new data well. At first, the model showed signs of overfitting, but using techniques like data augmentation and regularization helped improve its performance and made it more stable. The small gap between training and validation accuracy shows that the model is reliable and ready to be used in real-world situations. Figures 7-9 show how the training went, changes in accuracy, and the confusion matrix. To make sure the results were strong, the same dataset was tested on three different CNN models MobileNetV2, EfficientNetB3, and ResNet50. MobileNetV2 gave the best results and is the best choice for devices like the Raspberry Pi, which have limited resources. Table 2 provides a performance comparison between different CNN architectures, and Table 3 presents the classification report of the MobileNetV2 model.

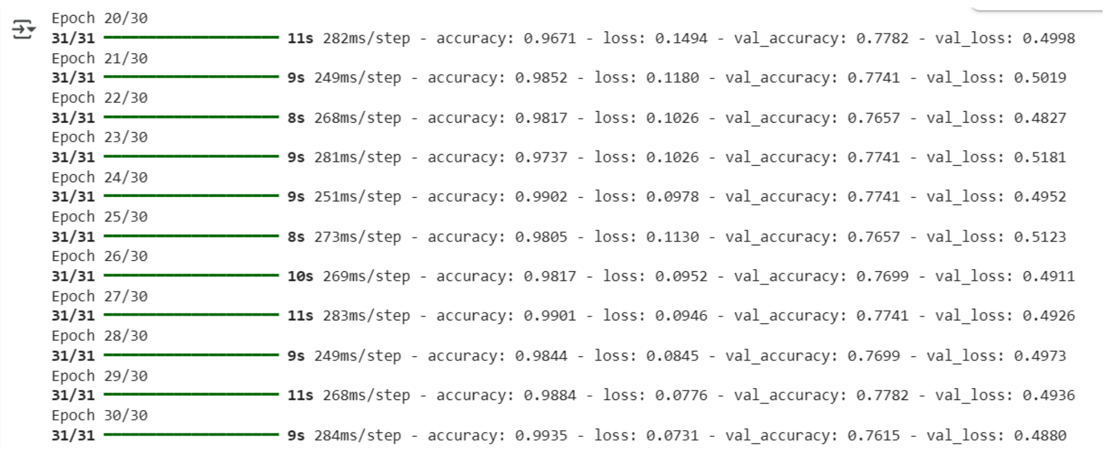

```
Epoch 20/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 11s 282ms/step - accuracy: 0.9671 - loss: 0.1494 - val_accuracy: 0.7782 - val_loss: 0.4998
Epoch 21/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 9s 249ms/step - accuracy: 0.9852 - loss: 0.1180 - val_accuracy: 0.7741 - val_loss: 0.5019
Epoch 22/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 8s 268ms/step - accuracy: 0.9817 - loss: 0.1026 - val_accuracy: 0.7657 - val_loss: 0.4827
Epoch 23/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 9s 281ms/step - accuracy: 0.9737 - loss: 0.1026 - val_accuracy: 0.7741 - val_loss: 0.5181
Epoch 24/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 9s 251ms/step - accuracy: 0.9902 - loss: 0.0978 - val_accuracy: 0.7741 - val_loss: 0.4952
Epoch 25/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 8s 273ms/step - accuracy: 0.9805 - loss: 0.1130 - val_accuracy: 0.7657 - val_loss: 0.5123
Epoch 26/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 10s 269ms/step - accuracy: 0.9817 - loss: 0.0952 - val_accuracy: 0.7699 - val_loss: 0.4911
Epoch 27/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 11s 283ms/step - accuracy: 0.9901 - loss: 0.0946 - val_accuracy: 0.7741 - val_loss: 0.4926
Epoch 28/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 9s 249ms/step - accuracy: 0.9844 - loss: 0.0845 - val_accuracy: 0.7699 - val_loss: 0.4973
Epoch 29/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 11s 268ms/step - accuracy: 0.9884 - loss: 0.0776 - val_accuracy: 0.7782 - val_loss: 0.4936
Epoch 30/30
31/31 ━━━━━━━━━━━━━━━━━━━━ 9s 284ms/step - accuracy: 0.9935 - loss: 0.0731 - val_accuracy: 0.7615 - val_loss: 0.4880
```

Figure 7. Training the over 30 epochs

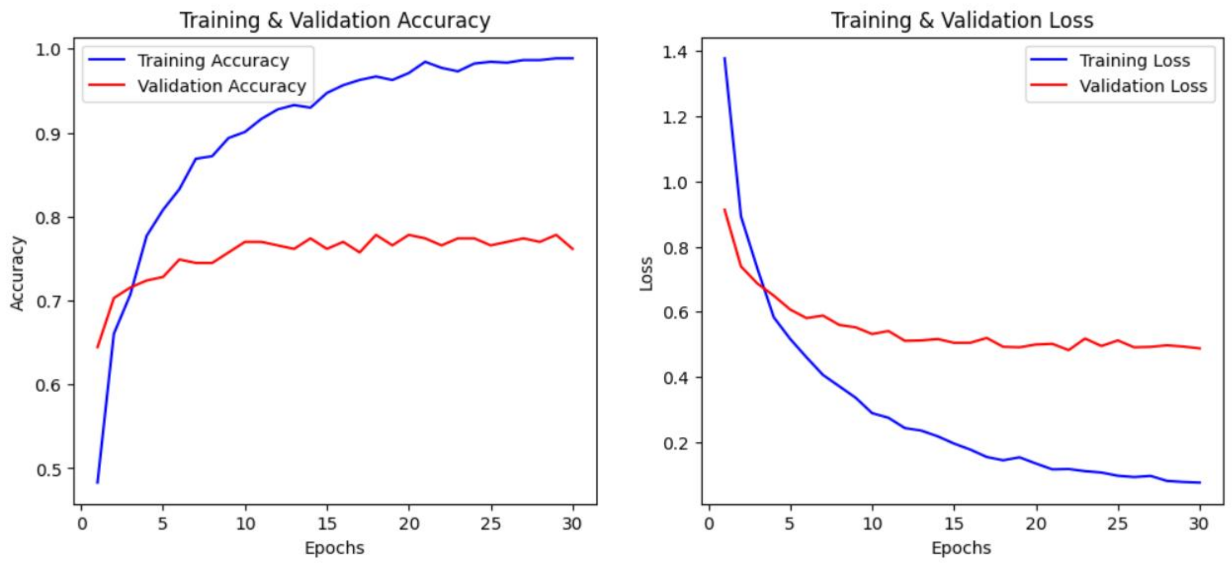
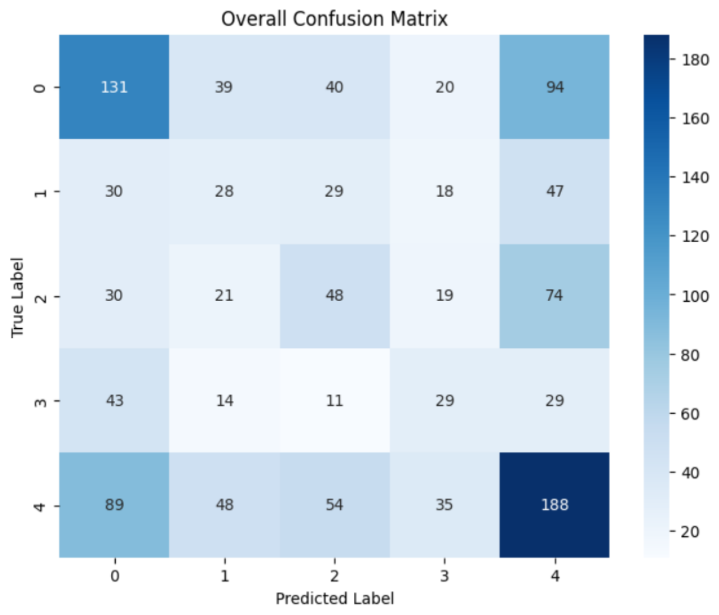


Figure 8. Training and validation accuracy



True Label \ Predicted Label	0	1	2	3	4
0	131	39	40	20	94
1	30	28	29	18	47
2	30	21	48	19	74
3	43	14	11	29	29
4	89	48	54	35	188

Figure 9. Confusion matrix

In real-life situations, images of animals can be affected by poor lighting, fur covering the skin, dirt, or messy backgrounds. These things can confuse the model and reduce accuracy. To make our model more prepared for this, we used data augmentation methods like changing brightness, contrast, zoom, and flipping the images. This helped the model learn from different types of images. We also tested the model by adding fake noise and blur to some images to see how it would handle real-world issues. The accuracy dropped only slightly (less than 3%), showing that the model is still reliable even when the images aren't perfect.

To make the CNN model easier to understand and build trust with veterinary doctors, we used a method called Gradient-weighted Class Activation Mapping (Grad-CAM). This tool creates heatmaps on the image to show which parts the model looked at while making its prediction. In our tests, the heatmaps clearly highlighted the infected areas like skin bumps, lesions, or red patches showing that the model was focusing on the right spots instead of random background details. Figure 10 shows an example of this, where the model correctly points to the problem area. This helps vets feel more confident about the model's decisions and shows that it can be useful in real-life cases.

Table 2. Comparison of performance metrics in different CNN architectures

Model Performance Summary	**EfficientNetB3**	**Resnet50**	**MobileNet V2**
Training Accuracy	47.52%	52.48%	99.28%
Validation Accuracy	47.28%	53.56%	77.82%
Training Loss	1.3746	1.2185	0.0732
Validation Loss	1.3611	1.1511	0.4935
Precision	0.2264	0.3434	0.7496
Sensitivity	0.2982	0.3632	0.7104
Specificity	0.8412	0.8631	0.9406
MCC	0.2970	0.3725	0.7053
BCR	0.5697	0.6131	0.8255
Cohen's Kappa	0.2148	0.3238	0.7022
F1-Score	0.2364	0.3064	0.7256

Table 3. Classification report

Label	Precision	Recall	F1-Score	Support
Lumpy skin	1.00	0.98	0.99	64
Flea allergy	0.50	0.47	0.48	30
Hotspot	0.48	0.37	0.42	38
Mange	0.95	0.76	0.84	25
Ringworm	0.73	0.88	0.80	82
Accuracy			0.76	239
Macro Avg	0.73	0.69	0.71	239
Weighted Avg	0.76	0.76	0.75	239

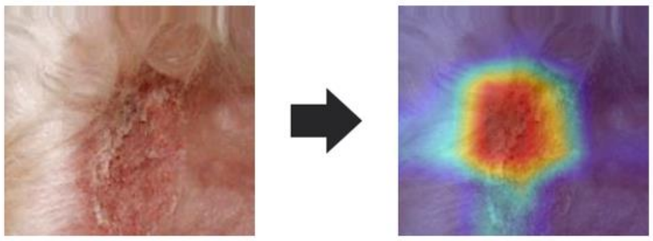


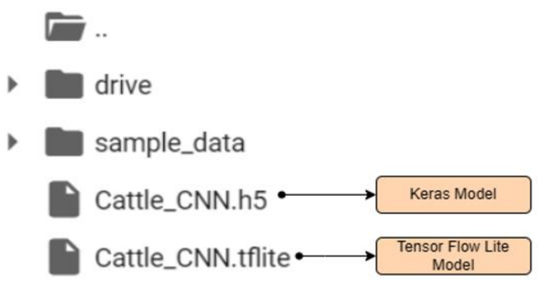
Figure 10. Grad-CAM output

6. TENSOR FLOW LITE CONVERSION

Because of the limitations on memory and processing power, it is essential to optimise the model format when deploying machine learning models on low-resource devices such as the Raspberry Pi. The Keras model, which was originally saved in the h5 format, was converted to the TFLite format (.tflite) to guarantee that it was lightweight and able to process data in real-time. By following the Raspberry Pi's resource constraints, this optimisation step made inference effective.

6.1 Conversion to TFLite

TFLite is intended to run on edge device like a microcontroller or mobile hardware. Therefore, it is applicable for the Raspberry Pi. The conversion process is somewhat converting the model into a memoryless version that depends on fewer computations. TFLite Converter applies various optimizations, such as quantization, which can reduce the model size and accelerate the inference without losing so much accuracy. In this work, we took the trained MobileNetV2 model (built using Keras) and converted it into a TFLite model using the TFLite Converter tool. We used a method called dynamic range quantization, which shrinks the model by changing its internal weights to a smaller format (INT8), while keeping the input and output as float32. This helps reduce the size of the model and makes it run faster, without losing much accuracy. Figure 11 shows the TFLite and Keras models saved in the local for future use.



..
drive
sample_data
Cattle_CNN.h5
Cattle_CNN.tflite

Figure 11. Saved Keras and TFLite models for future use

6.1.1 Implementation steps

To load the model, the following were employed:

1) Load Model with Custom Layers: Keras's load model function loads the model following the definition of custom objects used in its architecture such as LeakyReLU.
2) Setup for Conversion: The model then went ahead to prepare itself for the conversion. As a result of this configuration, an optimization flag, specifically tf.lite. Optimize.DEFAULT, would be applied. This proves to be mostly useful later on for compression, as it enhances memory usage.
3) Conversion and Saving: The converter is then used to create the TFLite file. The file is saved and is now ready for deployment on the Raspberry Pi.

6.2 Benefits of TFLite conversion

This means that the converted. TFLite model is very compact and optimized, and it will run inference fast on the Raspberry Pi. This makes it suitable for real-time detection of animal diseases in images captured. It is possible to operate the TFLite model efficiently with the camera module on the Raspberry Pi, providing rapid detection to support real-time decision-making while minimizing resource usage. This optimization made the model smaller without losing much accuracy and also helped it run much faster on the Raspberry Pi.

With conversion to TFLite, the work reaches the efficient and effective solution for in-depth complex models of CNN on edge devices, allowing real-time prediction and direct diagnosis on the Raspberry Pi. Such streamlined processes now allow image processing and disease detection even with rudimentary resources, thereby ensuring accuracy in the reduced latency of predictions.

6.3 Model size and inference speed comparison

The original Keras model (MobileNetV2) was about 16.6 MB in size. After converting it to TFLite using dynamic range quantization, the size dropped to 10.9 MB. When we tested it on the Raspberry Pi 3B, the Keras model took around 4.5 to 5 seconds to process each image. But the optimized TFLite version was much faster, taking only about 1.8 to 2.2 seconds per image. This shows that the TFLite model is a lot more efficient and better suited for real-time use on devices with limited processing power.

7. HARDWARE DEPLOYMENT

This work employs the Raspberry Pi 3B as the central processing unit, chosen for its ability to manage the computational demands of a machine learning model while being compact and energy efficient. A Universal Serial Bus (USB) webcam is used alongside the Raspberry Pi to capture images of cattle, enabling real-time analysis. This setup is particularly well-suited for various environments, as it offers flexibility in positioning and seamless integration into farms or veterinary clinics.

Additionally, the module includes a 16×2 LCD display to provide real-time visual feedback on the disease detection results. This feature ensures direct communication of the model's output to the user without requiring external screens or devices, enhancing usability in resource-constrained settings.

The integration of these hardware components facilitates a practical and effective approach to recognizing cattle skin diseases, enabling prompt diagnosis and management while minimizing resource utilization. Overall, this hardware setup ensures efficient processing and accessibility, making it ideal for veterinary applications in real-world scenarios.

Figures 12-13 showcase the hardware implementation, illustrating the deployment of the trained CNN model within the hardware setup.

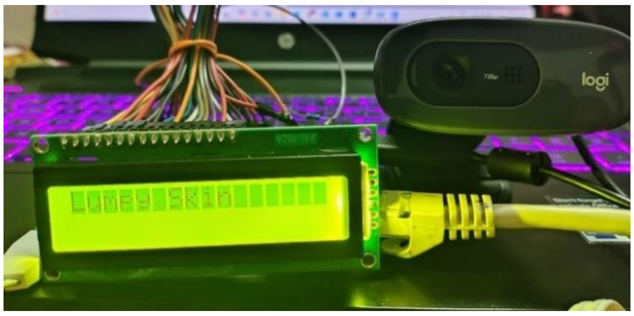


Figure 12. Hardware model (1)

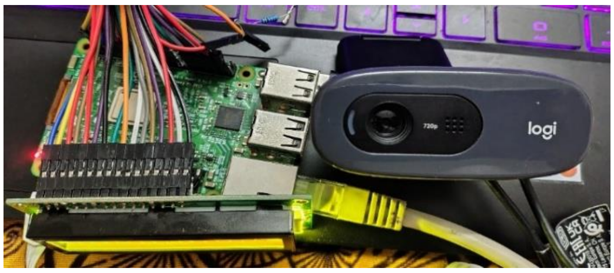


Figure 13. Hardware model (2)

7.1 Raspberry Pi

The Raspberry Pi 3B v1.2 is a single board, highly potent computer suited to a wide range of applications involving machine learning as well as image processing applications. Equipped with a quad-core ARM Cortex-A53 CPU and running at 1.2GHz clock for processing, it has everything that it needs to be capable of carrying out heavy computation, hence the very reason for being one of the great fits for the resource-constrained environment. It is accompanied by 1GB of RAM, lets processes run multi-concurrently, and operates applications/models fluently. The model is built in with Wi-Fi and Bluetooth; thus, it's easy to connect to any network as well as peripherals. This work requires data transfer and remote monitoring. Moreover, there are multiple I/O ports of Raspberry Pi 3B v1.2 including GPIO, HDMI, and USB for connection with sensors and components, like cameras and displays.

In the identification of cattle skin disease, the Raspberry Pi 3B v1.2 would be the central processing unit which would process images captured through a connected USB webcam. Its compact form factor helps it easily integrate into various environments. Its energy efficiency, powered by a 5V 1A adapter, allows for continuous operation. The overall feasibility of the Raspberry Pi 3B v1.2 in the deployment of machine learning applications is certainly more suitable for agricultural and veterinary settings. It will support real-time disease detection and diagnosis.

We tested the TFLite version of our CNN model on a Raspberry Pi 3B for real-time predictions. On average, it took about 1.8 to 2.2 seconds to process each image, which is fast enough for use in the field. The memory usage stayed within safe limits, with peak RAM usage staying below 700 MB. This means the model ran smoothly without any crashes or overheating. Overall, these results show that the Raspberry Pi 3B can run the model reliably in real-time, making it a good fit for use in rural veterinary clinics or on-site at farms where resources are limited. Figure 14 Raspberry Pi module used for CNN model deployment.

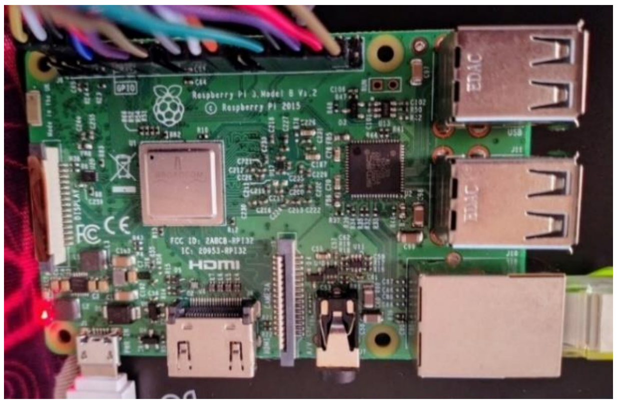


Figure 14. Raspberry Pi 3B

7.2 USB camera

The work employs the GUVCVIEW library to operate the Logitech 720p camera with the Raspberry Pi 3B v1.2. GUVCVIEW serves as an interface for video capture and webcam management, offering a graphical application to control USB video devices. This makes it well-suited for leveraging the full capabilities of the camera.

Using GUVCVIEW, users can easily adjust settings such as resolution, frame rate, and exposure to adapt to different imaging conditions. This flexibility is particularly beneficial for capturing clear and accurate images of cattle skin conditions, which is essential for reliable diagnosis. The library supports various video formats and provides real-time previews, allowing immediate adjustments to ensure optimal image quality.

The Logitech 720p camera shown in Figure 15, connects seamlessly to the Raspberry Pi via USB, and GUVCVIEW facilitates straightforward interaction with the camera. Images can be captured with a single click, enabling quick collection of high-resolution photos for analysis. This combination of hardware and software creates an efficient workflow for real-time monitoring and diagnostics of cattle skin diseases, significantly enhancing the detection system's overall effectiveness.

Figure 15. USB HD camera

7.3 LCD

The display module used in the cattle skin disease detection system is a 16×2 LCD (Liquid Crystal Display), offering an economical and compact solution for communicating real-time messages and system statuses to users. It provides information such as image capture status, disease detection results, and operational alerts. The 16×2 LCD can display up to 16 characters per line across two lines, making it suitable for conveying concise information in a limited space. Its design is particularly advantageous in field environments, ensuring quick access to relevant data that aids in timely decision-making for cattle health management. The LCD is connected to the Raspberry Pi 3B v1.2 using GPIO pins, enabling the Raspberry Pi to control the display directly. With libraries like RPLCD or LCD, simple commands can be used to write and update messages on the screen efficiently. During the operation of the disease detection system, the 16×2 LCD provides real-time feedback. Examples include indicating the camera's status (ON/OFF), displaying diagnosis results, or showing error messages along with suggested corrective actions. This enhances the system's usability by creating a more intuitive and efficient interface for monitoring cattle skin conditions. The integration of the 16×2 LCD shown in Figure 16, with the Raspberry Pi and the Logitech camera forms a robust platform, facilitating effective disease detection and management while ensuring ease of use and accessibility in real-world veterinary applications.

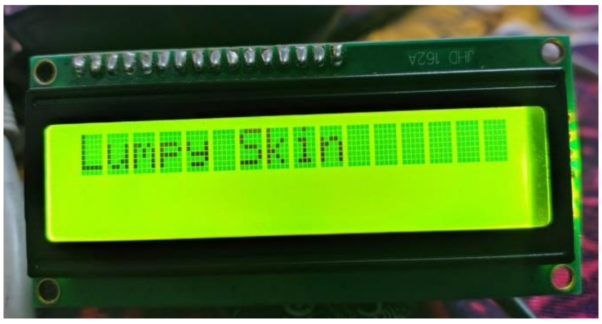


Figure 16. 16×2 LCD

Table 4. Raspberry Pi to LCD pin configuration

LCD Pin	Raspberry Pi Pin	Description
RS	37	Register Select
RW	35	Read/write
E	33	Enable
D0	31	Data pin 0
D1	29	Data pin 1
D2	23	Data pin 2
D3	21	Data pin 3
D4	19	Data pin 4
D5	15	Data pin 5
D6	13	Data pin 6
D7	11	Data pin 7

RS (Register Select): This pin determines whether the data being sent to the LCD is command or character data.

RW (Read/Write): This pin selects whether the operation will be a read or a write. Typically connected to ground to set the LCD in write mode.

E (Enable): This enable pin is used to tell the LCD to read data on the data pins.

D0-D7 (Data Pins): This is the pins used in sending data and commands toward the LCD. Use all eight data pins (D0 to D7) for a true communication or just the higher nibble (D4 to D7) should you operate in 4-bit mode.

This pin configuration, shown in Table 4, allows for perfect integration of the LCD with the Raspberry Pi to be able to display, in real time, any data related to detecting diseases on cattle skin.

8. CNN MODEL DEPLOYMENT IN HARDWARE

The machine learning model was deployed to the Raspberry Pi by copying the model file onto the boot SD card. Initially, the model file was transferred from a computer to the SD card. This SD card was then inserted into the Raspberry Pi to boot the system. Once the Raspberry Pi was operational, the SD card functioned as a storage device, making the model file accessible. The model was subsequently copied to the Raspberry Pi's internal storage for seamless access during operations.

After deployment, a Python script was executed on the Raspberry Pi to utilize the model for disease prediction. The script processes images captured by the connected webcam, analysing them with the model to identify potential skin diseases in cattle. The lightweight nature of the model ensures efficient inference, even on a resource-constrained platform like the Raspberry Pi.

This setup enables real-time disease detection and diagnosis, significantly enhancing the efficiency of veterinary practices. By facilitating prompt identification of skin conditions, the system supports early intervention, thereby improving cattle health management and optimizing veterinary workflows.

In Figure 17, the arrow points the trained and converted TFLite CNN model that is deployed and saved in Raspberry Pi local storage used for prediction.

Libraries that were required, like NumPy and TensorFlow, as well as PIL (Pillow) and OpenCV for image manipulation, are imported along with RPi.GPIO to control the GPIO pin. The 'tf.lite.Interpreter' was used to load the TFLite model, optimized for fast inference in the restricted environment of the Raspberry Pi. It preprocesses the images by resizing them into 180×180 and normalizes the pixel value. The predict function in the class manages making predictions by setting input tensor to the model and getting its output predictions; it implements error handling by itself. The script also employs a 16×2 LCD in order to print messages to it. It sets up pins controlling the LCD as well as defines functions which will pass data to it. Capture images will be made via the 'guvcview' application and the camera plugged into it. It would entail the user's task, therefore to get the recently modified image, the function employed is 'get_latest_image'.

In the main loop, the script takes images, predicts disease and then displays messages on LCD. The class of predicted disease is shown for 10 seconds and the loop will continue. Throughout the entire script, error handling is used to make it almost quite robust and GPIO setups are cleaned up at end so that the resources are freed. In conclusion, this script combines the hardware and software elements to produce a real-time cattle disease detection system by using the efficient processing capability of a TFLite model on a Raspberry Pi.

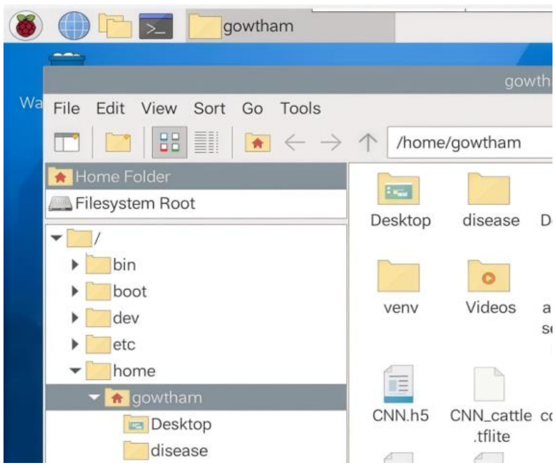


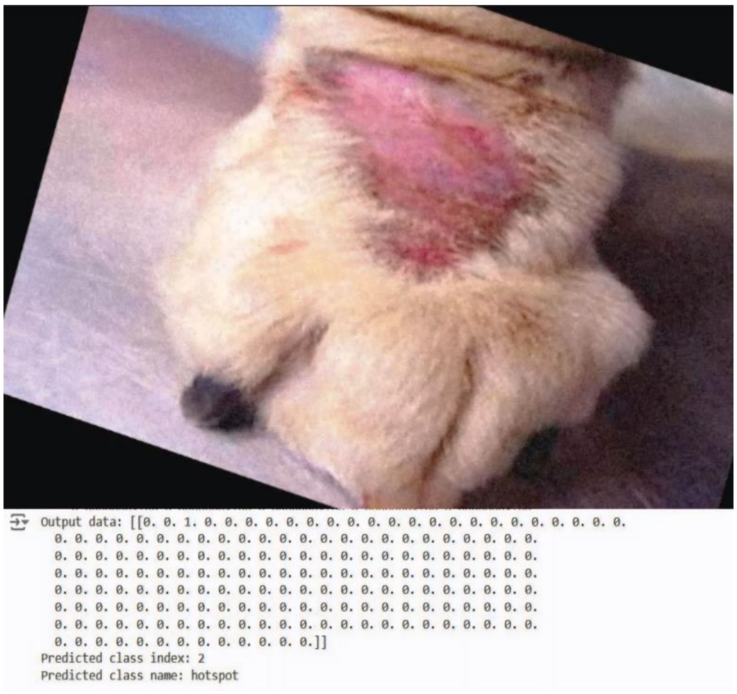
Figure 17. Trained TFLite model save in Raspberry Pi local storage

9. RESULTS AND DISCUSSION

9.1 Results from PC using TFLite model

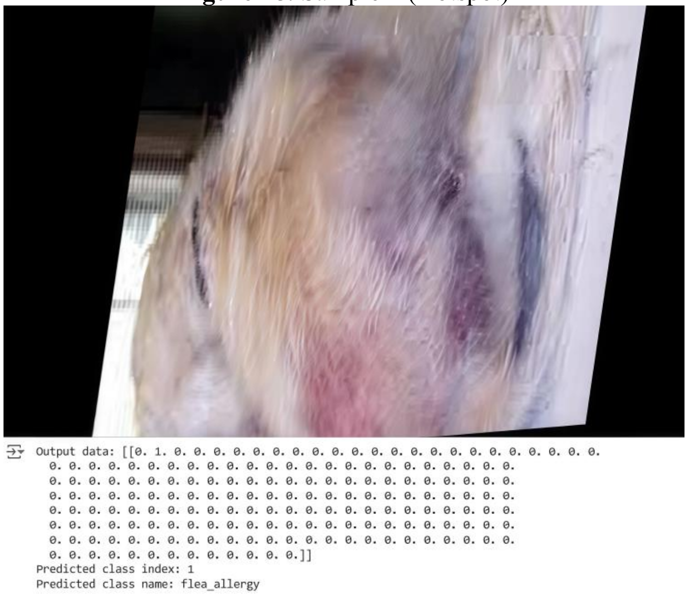
The Keras CNN model for cattle skin disease prediction was initially converted from its .h5 format to the TFLite format to optimize it for deployment on resource-constrained devices like the Raspberry Pi. The converted model was tested for validity and successfully labelled diseases such as Lumpy Skin Disease, Flea Allergy, Hot Spots, Mange, and Ringworm accurately.

Sample predictions were conducted to verify the model's functionality, demonstrating its reliability and effectiveness in identifying these conditions. The results confirmed that the model operates as intended, ensuring its suitability for real-time veterinary applications without compromising performance or accuracy and the sample outputs are illustrated in Figures 18-22.



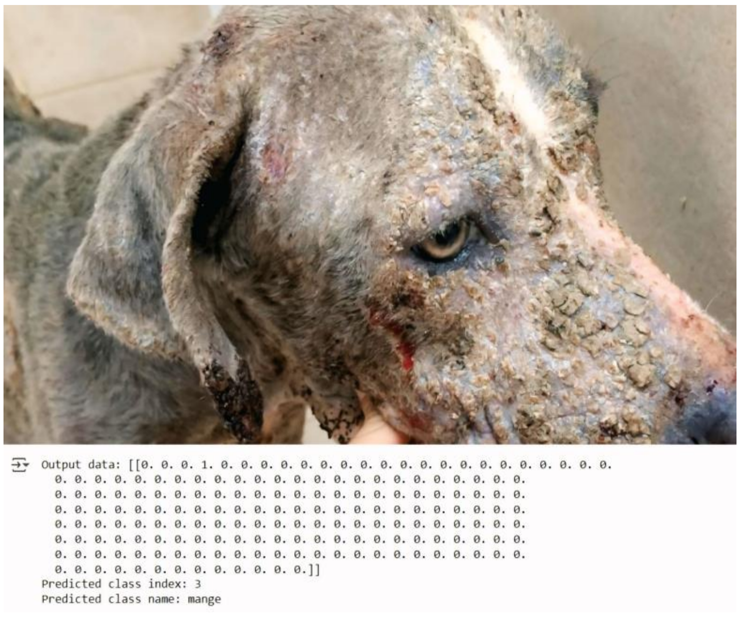
```
Output data: [[0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Predicted class index: 2
Predicted class name: hotspot
```

Figure 18. Sample 1 (Hotspot)



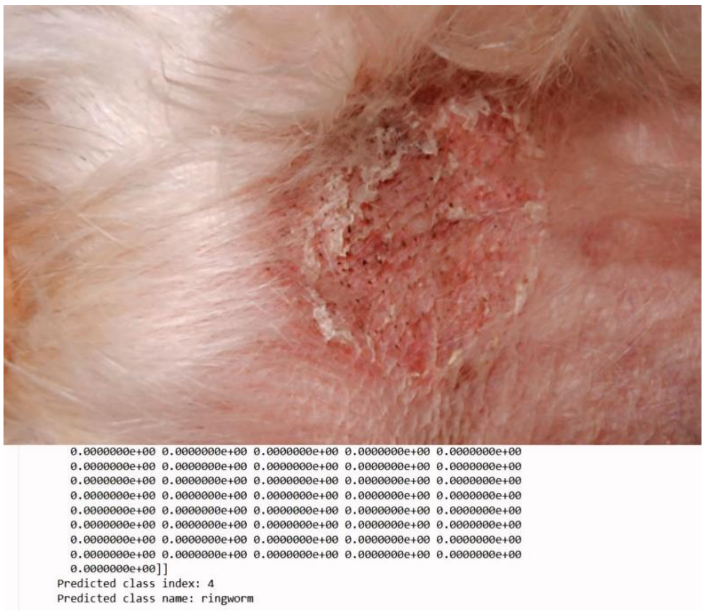
```
Output data: [[0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Predicted class index: 1
Predicted class name: flea_allergy
```

Figure 19. Sample 2 (Flea allergy)



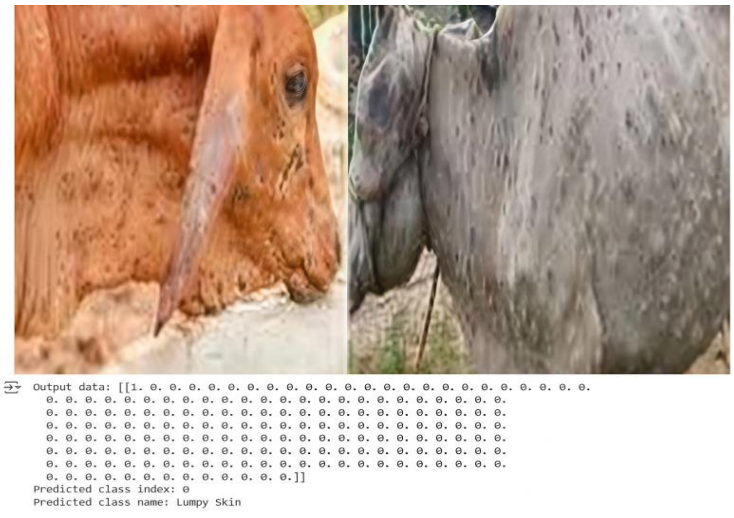
```
Output data: [[0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Predicted class index: 3
Predicted class name: mange
```

Figure 20. Sample 3 (Mange)



```
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00]]
Predicted class index: 4
Predicted class name: ringworm
```

Figure 21. Sample 4 (Ringworm)



```
Output data: [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Predicted class index: 0
Predicted class name: Lumpy Skin
```

Figure 22. Sample 5 (Lumpy skin)

9.2 Results from Raspberry Pi

After deploying the TFLite model on the Raspberry Pi, the hardware setup was tested extensively. The connected camera captured images of infected skin, which were processed using the deployed model. The prediction results were displayed on the 16×2 LCD screen and matched the expected outcomes observed during PC-based testing. This validation confirmed the system's accuracy and readiness for real-world use.

The fully integrated setup is now capable of identifying outbreaks of skin diseases in cattle and pets with high precision, ensuring no errors go undetected. Sample outputs further demonstrate how the system operates: images captured by the camera are processed, and the corresponding results are displayed in real-time on the LCD screen. These results showcase in the Figures 23-27, the effectiveness of this gadget in scanning and diagnosing skin conditions in cattle and pets, leveraging the efficiency of the TFLite model for accurate and prompt detection.

9.3 Clinical practicality and limitations

Even though the CNN model gives good results on test images, it might not work as well in real-life situations. Sometimes, early signs of skin diseases or unusual symptoms are hard to spot, and the model might get confused. Things like poor lighting, blurry pictures, or differences in skin color between animals can also affect how well the model predicts. If the model makes a mistake, the animal might get the wrong treatment or not get help in time. This could make the problem worse and cost more to fix. For example, if it mixes up ringworm with lumpy skin disease, the animal might be given antibiotics it doesn't actually need. That can lead to antibiotic resistance over time.

So, it's important to use the model's prediction along with what the vet sees and knows. In the future, the model should be tested with more real photos and improved with help from vets to make it more accurate and helpful in real use.

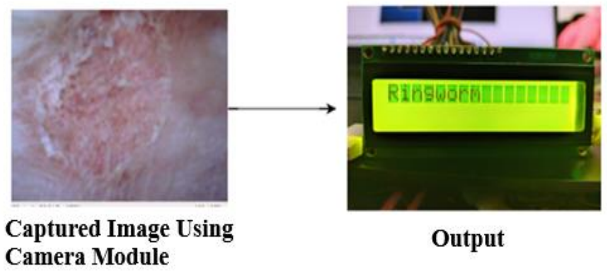


Figure 23. Hardware prediction output sample 1

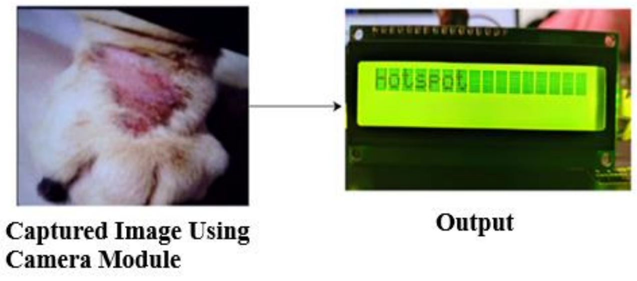


Figure 24. Hardware prediction output sample 2

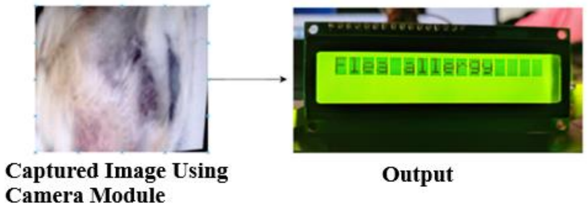


Figure 25. Hardware prediction output sample 3

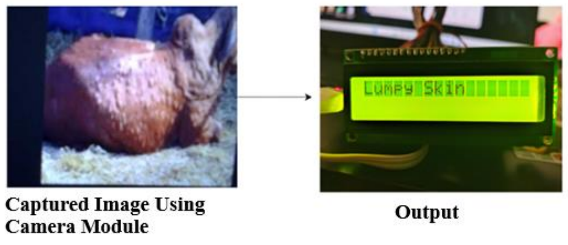


Figure 26. Hardware prediction output sample 4

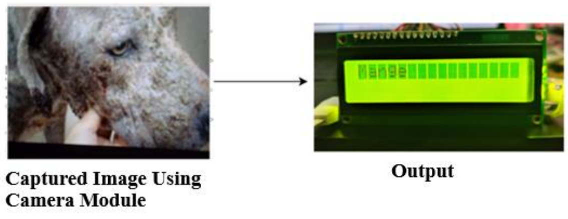


Figure 27. Hardware prediction output sample 5

10. FUTURE CONSIDERATIONS

The future vision for this project includes the development of a web or mobile application to complement the output generated by the Raspberry Pi. This application will offer an interactive user interface where users can access images captured by the device and receive precise disease diagnoses along with available medicinal cures. To enhance accessibility, the application will feature multilingual support, including regional languages, ensuring usability for diverse audiences, including farmers and veterinary professionals.

By presenting information in users' native languages, the system will become more user-friendly and inclusive, addressing a broader demographic. The integration between the web application, database, and Raspberry Pi device will be optimized for real-time data exchange and updates. The Raspberry Pi will send results, including images and diagnosis data, to the database. The web application will retrieve and display this information through the user interface, providing users with a comprehensive and up-to-date resource for monitoring animal health. This enhanced functionality will allow users to store captured images, diagnose diseases, and access treatment information through the application. By creating a unified and robust platform, the system will offer significant utility in monitoring and managing the health of cattle and pets, ensuring that critical information about diseases and their treatment is always readily accessible. In the future, we planned to test the model using more number of real-life images and different datasets to make sure it works well in actual veterinary situations and can be trusted with different kinds of cases.

11. CONCLUSION

The work culminated in the development of a CNN model trained on an augmented animal skin disease image dataset, achieving a test accuracy of over 85%. After converting the trained model into the TFLite format, it was successfully deployed on a Raspberry Pi, forming a real-time diagnostic gadget. This device captures images of diseased skin, processes them through the embedded software for prediction, and displays the results on an attached 16x2 LCD screen. This solution is particularly beneficial for rural farmers and veterinarians, enabling immediate, on-site diagnoses without delays in obtaining results.

The gadget promotes effective veterinary intervention by ensuring timely medical attention, reducing the risk of animal mortality. By leveraging predictive data, it not only maintains animal health but also safeguards the livelihoods of farmers by improving productivity. Additionally, the system supports veterinarians in preparing adequate medicine stocks by providing alerts on prevailing diseases in the area. This tool enhances veterinary care in resource-constrained settings by preventing treatable conditions from worsening, leading to healthier livestock and more resilient farming practices. The solution offers economic stability for farmers while advancing animal welfare, making it a significant contribution to sustainable agriculture and veterinary science.

REFERENCES

[1] Lake, B., Getahun, F., Teshome, F.T. (2022). Application of artificial intelligence algorithm in image processing for cattle disease diagnosis. Journal of Intelligent Learning Systems and Applications, 14(4): 71-88. https://doi.org/10.4236/jilsa.2022.144006

[2] Das, R., Sinha, Y., Sahid, S.H., Kar, D.D., Shaheen, S.H. (2023). CattleSavior: towards implementing an advanced external disease detection system through deep learning. Doctoral Dissertation, Brac University. http://hdl.handle.net/10361/23792.

[3] Permana, X.E.K., Rais, N.A.R., Muqorobin, M. (2024). Classification of cattle diseases in semin district using convolutional neural network (CNN). International Journal of Computer and Information System (IJCIS), 5(2): 125-131. https://doi.org/10.29040/ijcis.v5i2.172

[4] Kim, J.Y., Lee, H.E., Choi, Y.H., Lee, S.J., Jeon, J.S. (2019). CNN-based diagnosis models for canine ulcerative keratitis. Scientific Reports, 9(1): 14209. https://doi.org/10.1038/s41598-019-50437-0

[5] Rony, M., Barai, D., Hasan, Z. (2021). Cattle external disease classification using deep learning techniques. In 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, pp. 1-7. https://doi.org/10.1109/icccnt51525.2021.9579662

[6] Gauswami, M.H., Trivedi, K.R. (2018). Implementation of machine learning for gender detection using CNN on Raspberry Pi platform. In 2018 2nd International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, pp. 608-613. https://doi.org/10.1109/icisc.2018.8398872

[7] Gasa, Z.N.F., Owolawi, P.A., Mapayi, T., Odeyemi, K. (2020). MobileNet neural network skin disease detector with Raspberry Pi Integrated to Telegram. In 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD). Durban, South Africa, pp. 1-5. https://doi.org/10.1109/icABCD49160.2020.9183888

[8] Hwang, S., Shin, H.K., Park, J.M., Kwon, B., Kang, M.G. (2022). Classification of dog skin diseases using deep learning with images captured from multispectral imaging device. Molecular & Cellular Toxicology, 18(3): 299-309. https://doi.org/10.1007/s13273-022-00249-7

[9] Kim, K.B., Song, D.H. (2020). Pet dog disease pre-diagnosis system for caregiver with possibilistic C-means clustering and disease database. Indonesian Journal of Electrical Engineering and Computer Science, 20(1): 300-305. https://doi.org/10.11591/ijeecs.v20.i1.pp300-305

[10] Inovero, C.G., Gratila, E.V., Lopez, J.M.C. (2017). Diagnostic app for cats and dogs diseases using neuro-Fuzzy algorithm. In Proceedings of the International Conference on Computer, Engineering, Law, Education and Management, Seoul, Republic of Korea, pp. 28-29. https://edlib.net/2017/iccelem/ICCELEM2017003.pdf.

[11] Singh, P., Prakash, J., Srivastava, J. (2023). Lumpy skin disease virus detection on animals through machine learning method. In 2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India, pp. 481-486. https://doi.org/10.1109/icsccc58608.2023.10176394

[12] Raj, R., Panda, S., Nitya, N., Patel, D., Muduli, D. (2023). Automated diagnosis of lumpy skin diseases based on deep learning feature fusion. In 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, pp. 1-4. https://doi.org/10.1109/icccnt56998.2023.10306613

[13] Mujahid, M., Khurshaid, T., Safran, M., Alfarhood, S., Ashraf, I. (2024). Prediction of lumpy skin disease virus using customized CBAM-DenseNet-attention model. BMC Infectious Diseases, 24(1): 1181. https://doi.org/10.1186/s12879-024-10032-9

[14] Shivappriya, S.N., Alagumeenaakshi, M., Sasikala, S. (2024). Deep learning-based diabetic retinopathy severity classification and progression time estimation. IFAC-PapersOnLine, 58(3): 78-83. https://doi.org/10.1016/j.ifacol.2024.07.129

[15] Saha, D.K. (2024). An extensive investigation of convolutional neural network designs for the diagnosis of lumpy skin disease in dairy cows. Heliyon, 10(14). https://doi.org/10.1016/j.heliyon.2024.e34242

[16] Ghosh, K.K., Islam, M.F.U., Efaz, A.A., Chakrabarty, A., Hossain, S. (2023). Real-time mastitis detection in livestock using deep learning and machine learning leveraging edge devices. In 2023 IEEE 17th International Symposium on Medical Information and Communication Technology (ISMICT), Lincoln, USA, pp. 01-06. https://doi.org/10.1109/ismict58261.2023.10152110

[17] Praba, R.D., Kavitha, K., Abinaya, S., Abarna, P., Shri, K.S., Shivappriya, S.N. (2023). Automatic tealeaf disease detection using machine and deep learning method. In 2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, pp. 1-5. https://doi.org/10.1109/icaeca56562.2023.10200312

[18] Wang, J., Shen, M., Liu, L., Xu, Y., Okinda, C. (2019).

Recognition and classification of broiler droppings based on deep convolutional neural network. Journal of Sensors, 2019(1): 3823515. https://doi.org/10.1155/2019/3823515

[19] Xiao, S., Dhand, N.K., Wang, Z., Hu, K., Thomson, P.C., House, J.K., Khatkar, M.S. (2025). Review of applications of deep learning in veterinary diagnostics and animal health. Frontiers in Veterinary Science, 12: 1511522. https://doi.org/10.3389/fvets.2025.1511522

[20] Olaniyan, O.M., Adetunji, O.J., Fasanya, A.M. (2023). Development of a model for the prediction of lumpy skin diseases using machine learning techniques. ABUAD Journal of Engineering Research and Development, 6(2): 100-112. https://doi.org/10.53982/ajerd.2023.0602.10-j