CATTLE SKIN DISEASE CLASSIFICATION USING MOBILENET V2

ABSTRACT – Lumpy skin disease (LSD) is a kind of infection caused by Neethling virus that affects cattle, causing them to develop nodules on their skin, have fever and show various related symptoms. Currently, there is no vaccination available for eradicating the LSD virus; instead, existing vaccines can only control its spread. Therefore, early diagnosis of the disease is crucial to prevent its transmission to other cows. To predict the disease in cattle, we utilized one of the transfer learning methods called MobileNet V2 which analyzes images of cattle and predicts if they have the disease or not.

Keywords: LSD, MobileNet V2

I. INTRODUCTION

Lumpy Skin Disease is categorized as cattle disease, which is characterized by mucous membranes, nodules on skin, fever, oedema of the skin and even death. This disease is communicative and is transmitted by insects, such as, mosquitoes, flies and ticks. LSD is caused by a virus, known as the Neethling virus, that is associated with the family of Poxviridae. Two different approaches have already been done to immunization against this disease. Though the development of sheepox and goatpox vaccines provided immunity in cattle, these vaccines were restricted to use since the level of attenuation required was not sufficient. In this study, we are incorporating MobileNetV2 model for classifying the disease affected skin region and normal skin, for the early detection of diseases in cattle. We further compare the results of this model with two customized models by incorporating the features of SVM classifier with MobileNetV2 along with the usage of Adam optimizer.

II. LITERATURE REVIEW

To develop the project further, we have referred a few research articles about the classification of the cattle skin diseases by incorporating different machine learning techniques. The first research paper discusses about, “Assessing machine learning techniques in forecasting lumpy skin disease occurrence based on meteorological and geospatial features”. Predictive models were developed using algorithms such as the ExtraTreesClassifier, which was used to select significant features to forecast the occurrence of disease in test data. The research further revealed that, ANN (Artificial Neural Networks), predicts the incidence of LSDV with a high degree of accuracy precision using the meteorological and geospatial parameters.

The article on, “Detecting High-risk Area for Lumpy Skin Disease in Cattle Using Deep Learning Feature”, discusses about the segmentation and classification of the disease based on deep learning. They have pictorially represented the disease affected area using a color histogram. The deeply pre-trained CNN extracts the features from the segmented area of affected skin color.

The classification is done by the Extreme learning machine classifier. This research achieved an accuracy of about 90.12% on the collected dataset for the lumpy skin disease of cattle.

The research about the, "Development of a Model for the Prediction of Lumpy Skin Diseases using Machine Learning Techniques", developed an advanced model to meliorate the performance of the existing model for predicting the LSDV. The development of a stacked ensemble of single classifiers, that includes, decision tree, KNN, random forest and SVM, along with an optimized ANN, exceled the results of the already existing model, by providing an accuracy of 98% approximately.

The research article, "A Survey on Deep Learning Method to Identify Lumpy Skin Disease in Cows", developed a ML architecture for detecting illness, by classifying the input images into two categories, namely, LSDV and non-LSDV. The research focussed on evaluating the ML algorithms for the accurate prediction of Lumpy Skin Diseases based on images, such as the Densenet-121, to handle the issues of disappearing gradients and minimizing the consumption of parameters.

The paper, "Cow Disease (LSD) Classification System for predicting different severity levels", proposed a system consisting of different pre-trained convolutional neural network models that has been customized by the addition of new layers. The model is trained which in turn predicts the level of severity of the Lumpy Skin Disease. After comparing their model with other pre-existing training models, such as the, VGG19, Xception, Inception V3, DenseNet121, and ResNet50, the customized model achieved an accuracy rate of 91.82%, thereby outperforming all the other models. This makes it ideal for the diagnosis of Lumpy Skin Disease in cattle.

The research article on, "Classification of Skin Disease Using Deep Learning Neural Networks with MobileNetV2 and LSTM", proposes a model which utilizes a grey-level co-occurrence matrix, that evaluates the development of the disease, by incorporating the features of both MobileNetV2 and Long Short Term Memory (LSTM). This model has exceled the other models such as the VGG, CNN, Fine-Tuned Neural Networks (FTNN), with an accuracy of more than 85%. The computational cost is two times lesser than that of the conventional MobileNet model and has a faster rate for the identification of the region that is affected by the disease.

III. METHODOLOGY

(1) Data Splitting: The given dataset for lumpy and normal skin diseases is divided in the ratio of 80:10:10 for training, validation and testing respectively. Initially, there were a total of 1024 images out of which 700 were normal and 324 were lumpy. From this dataset, after the elimination of animated and irrelevant images, we finally acquired a dataset consisting of 304 lumpy and 634 normal, which constitutes a total of 938 images. After splitting the dataset corresponding to the ratio, we obtained a total of 657 images, 213 lumpy and 444 normal for training dataset; 94 images, 31 lumpy and 63 normal for validation dataset, and 93 images in total each for test dataset 1 and test dataset 2, 30 lumpy and 63 normal. After performing data augmentation, the total number of images, including 1826 images of lumpy class and 3802 of normal class, is 5628. For the training dataset, 3939 total images, consisting of 1278 lumpy and 2661 normal; validation dataset, 184 lumpy and 380 normal from a total of 564 images; testing dataset, a total of 562 images, 182 lumpy and 380 normal class of images.

(2) Data pre-processing: Initially, there are certain pre-processing steps that need to be followed before feeding the input data (images) into the MobileNetV2 model. These involve: preprocess_input function, which performs normalization of the pixel values, of the input images, where, the pixel values are aligned to a certain range that is expected by the model. Data augmentation is a technique, which is used to modify the input data by performing various transformations, such as, rotation, width shift, height shift, shear, zoom and horizontal flip, in order to increase the size of the training dataset and thereby enhance the performance of the training model. By performing data augmentation, we also make the model sturdy invariable to different type of modifications in the input images. The next step to be done, is the resizing of the images, such that, all the images are of the target size of (256x256) pixels, and this step is ensures whether all the images in input dataset are of a consistent size for processing. The ImageDataGenerator object is utilized to load and process the images batch wise, with a batch size set to 30 for training data and 20 for validation and test data.

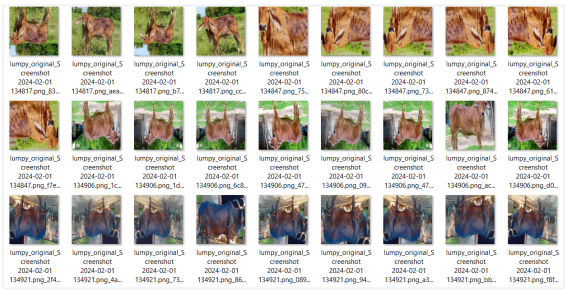


Fig 1: Images after augmentation process

IV. ARCHITECTURE

The MobileNetV2 architecture consists of the hyperparameters width and resolution multipliers. The width multiplier is represented by the symbol for alpha. It either decreases the number of channels for values of alpha less than 1 and increases the number of channels for values of alpha greater than 1. When adjusting the width multiplier, we can manipulate the capacity and computational cost of the model. The resolution multiplier is represented by the symbol for rho. For values less than 1, the resolution of the input images reduces, conversely, for values greater than 1, the resolution increases.

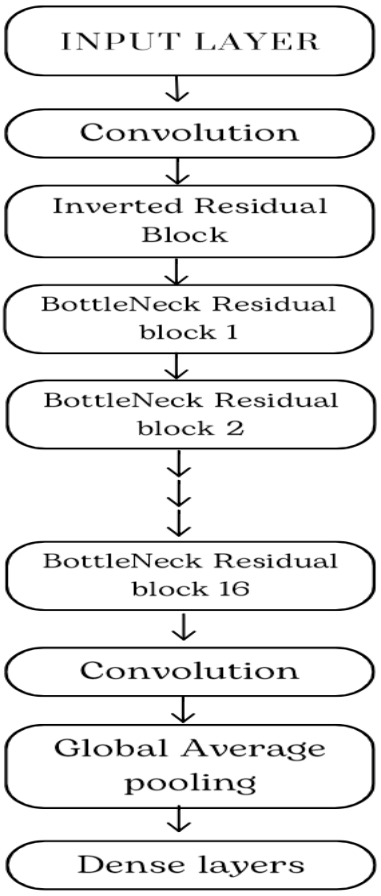


Fig 2: MobileNetV2 Architecture

The initial layer of the MobileNetV2 architecture, is the input layer, where input data or the images that need to processed are loaded. The second layer is the convolution layer, that extracts low-level features, such as, texture, lines and edges from the input images, which in turn produces a set of feature maps as the output. The inverted residual block is designed in

such a way that, it captures the complex features from the feature maps by performing a series of transformations, that include, 3x3 depthwise convolution, 1x1 convolution with ReLu activation function, and a 3x3 depthwise convolution. The bottleneck residual block is designed to maintain the efficiency of the model and in further extracting the significant features. It applies certain transformations, that includes, 1x1 convolution with ReLu for expansion, 3x3 depthwise convolution with ReLu and a 1x1 convolution without any linearity, for projection. It consists of a shortcut connection, also known as the skip connection, which helps in preserving the original information, mitigating the issues of disappearing gradients. These two are followed by a set of inverted and bottleneck residual layers, which further enhances the model to extract more complex features. A convolution layer, helps in improving the quality of the input images by applying filters and refining the set of feature maps produced by the previous blocks. This layer is followed by a global average pooling layer, which reduces the dimensionality of the feature maps, producing a fixed size output, by calculating the average of each feature map. The last block of this architecture are dense layers or the fully-connected layers, transforms the high-level features learned by the previous blocks into predictions or class scores.

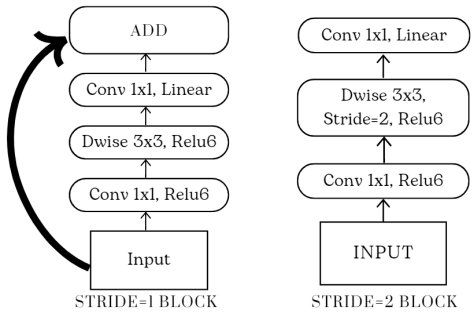


Fig 3: Stride Blocks

In the MobileNetV2 architecture, the filters convolve or slide over the input images while performing convolutional operations. The filter movements are directed by the stride parameters. When the stride parameter has a value of 1, the filter slides over the input image one pixel at a time. The feature maps with the same spatial dimensions, same as the input are produced. With a stride of 2, when the filter moves across the input images, it skips every other pixel. The output feature maps are down sampled by a factor of 2 along each spatial dimension.

V. MODEL PARAMETERS

The parameters of the model include, (a) base model parameters, (b) custom dense layer parameters, and (c) model compilation parameters. The base model parameters consist of, 'imagenet' which represents 'weights', indicating that the model is initialized with pre-trained weights from the ImageNet dataset; 'include_top', a Boolean function, is set to false, where the fully-connected top layers of the model are excluded and the base layers are used for extracting the features. Global average pooling layer is incorporated to reduce the spatial dimensions of the feature maps, thereby tackling the issues of disappearing gradients. The average of each feature map is calculated, producing a fixed-size output irrespective of the input size. There are a total of three dense layers that are added on top of the MobileNetV2 to perform classification, and these include, an initial dense layer with 512 neurons, followed by, a dense layer with 256 neurons and another dense layer with 128 neurons, each activated by a Rectified Linear Unit (ReLu). These layers are used to learn more abstract features from the output feature maps that have been produced by the previous blocks. Finally, there is a dense layer consisting of one neuron unit and a

sigmoid activation function, which compresses the output between the binary values, 0 and 1, making it ideal for the binary classification tasks. The biases and weights in these dense layers are updated during the training process to minimize the defined loss function by setting the trainable attribute to true. The model compilation parameters consist of, a loss function, 'binary_crossentropy' that needs to minimized; Adam optimizer, updates the weights of the neural network, by adapting the learning rates individually; More stable convergence is achieved for smaller learning rates and learning rate used for this model is 0.0001, which determines the step size at each iteration; 'accuracy' is an evaluation metric, which evaluates the performance of the model during training and testing processes. These parameters all together, define the architecture of the model, which is used for training the model effectively and evaluating its performance on unseen data. The parameters can be modified, in order to, influence the convergence and generalization ability of the model.

VI. EVALUATION METRICS

A confusion matrix is a table that illustrates the effectiveness of a model by comparing the predicted labels against the actual values. It encompasses four key terms: true positives (TP), accurately forecasted as positive; true negatives (TN), accurately forecasted as negative; false positives (FP), inaccurately forecasted as positive; false negatives (FN), inaccurately forecasted as negative.

Confusion Matrix		Actual	
		Positive	Negative
Predicted	Positive	**TP**	**FP**
	Negative	**FN**	**TN**

The performance of the MobileNetV2 model is assessed using two key metrics, namely, accuracy and loss. Accuracy is the fraction of samples that the model correctly classified out of the total sample count. A higher accuracy value signifies superior performance. The accuracy is calculated using the formula:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Loss is a metric that quantifies the error in the model's predictions. The binary cross-entropy loss is computed as the negative logarithm of the predicted probability for the actual class. A lower loss value signifies superior performance. A classification report provides details about precision, recall, F1-score, and support. Precision is a measure of the correctness of positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

The formula for recall is given by,

$$Recall = \frac{TP}{TP + FN}$$

The F1-score is given by the formula,

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

VII. RESULTS

The results acquired for the training of MobileNetV2 with CNN classifier are as follows:

Training Accuracy and Loss:

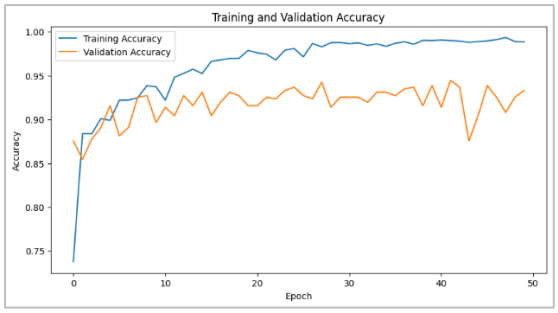


Fig 4: Training and Validation accuracy curve

Fig 5: Training and Validation loss curve

Validation Accuracy, Loss and Classification Report:

```
18/18 [==============================] - 40s 2s/step - loss: 0.4127 - accuracy: 0.9080
Validation Accuracy: 0.9080459475517273
18/18 [==============================] - 42s 2s/step

Classification Report for Validation Set:
              Precision    Recall  F1-Score   Support
     Class 0       0.28      0.28      0.28     144.0
     Class 1       0.72      0.72      0.72     378.0
Weighted Avg.      0.60      0.60      0.60     522.0
Macro Avg.         0.50      0.50      0.50     522.0
```

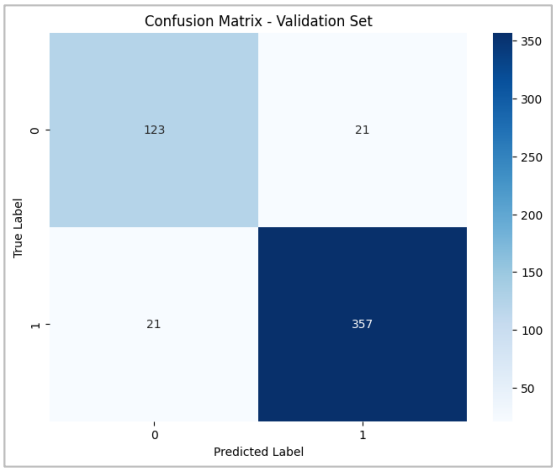
Test set1 Accuracy, Loss and Classification Report:

```
Test Set 1 Accuracy: 0.8697317838668823
17/17 [==============================] - 26s 947ms/step

Classification Report for Test Set 1:
              Precision    Recall  F1-Score   Support
     Class 0       0.28      0.31      0.29     144.0
     Class 1       0.72      0.69      0.70     378.0
Weighted Avg.      0.60      0.58      0.59     522.0
Macro Avg.         0.50      0.50      0.50     522.0
```

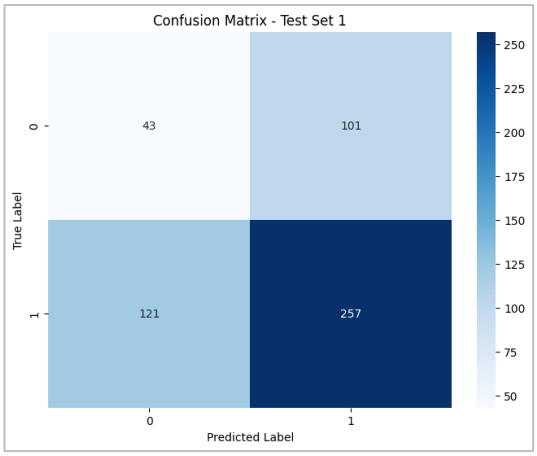
Test set2 Accuracy, Loss and Classification Report:

```
Test Set 2 Accuracy: 0.8898916840553284
28/28 [==============================] - 29s 1s/step

Classification Report for Test Set 2:
              Precision    Recall  F1-Score   Support
     Class 0       0.32      0.38      0.35     178.0
     Class 1       0.68      0.62      0.65     376.0
Weighted Avg.      0.56      0.54      0.55     554.0
Macro Avg.         0.50      0.50      0.50     554.0
```

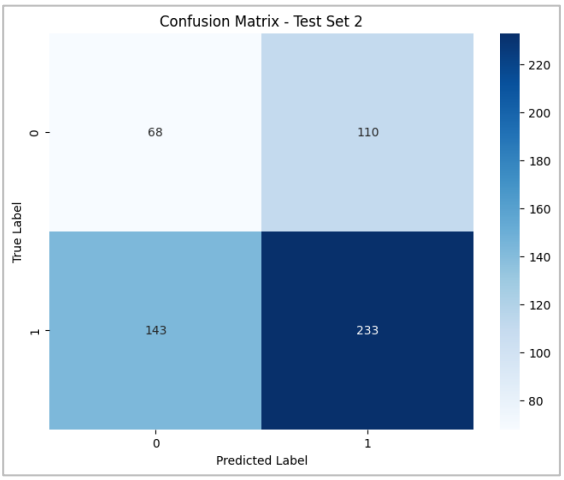
```
Confusion Matrix for Validation Set:
[[123  21]
 [ 21 357]]
```




```
Confusion Matrix for Test Set 1:
[[ 43 101]
 [121 257]]
```




```
Confusion Matrix for Test Set 2:
[[ 68 110]
 [143 233]]
```

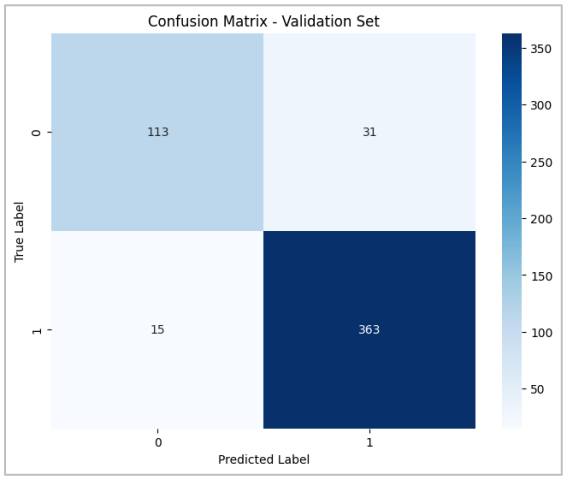


The results achieved for MobileNetV2 incorporated with **SVM classifier**:

For Validation:

Classification Report for Validation Set:

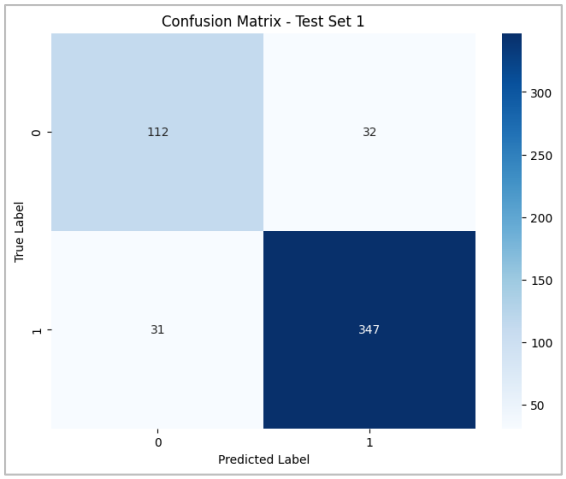
	precision	recall	f1-score	support
0.0	0.88	0.78	0.83	144
1.0	0.92	0.96	0.94	378
accuracy			0.91	522
macro avg	0.90	0.87	0.89	522
weighted avg	0.91	0.91	0.91	522



For Test set 1:

Classification Report for Test Set 1:

	precision	recall	f1-score	support
0.0	0.78	0.78	0.78	144
1.0	0.92	0.92	0.92	378
accuracy			0.88	522
macro avg	0.85	0.85	0.85	522
weighted avg	0.88	0.88	0.88	522



For Test set2:

Classification Report for Test Set 2:

	precision	recall	f1-score	support
0.0	0.80	0.80	0.80	178
1.0	0.90	0.91	0.91	376
accuracy			0.87	554
macro avg	0.85	0.85	0.85	554
weighted avg	0.87	0.87	0.87	554