

Dairy Cow Teat Health Condition Classification using Deep Learning Techniques

Ankit Kumar Aggarwal
Yeshiva University, New York
aaggarwa@mail.yu.edu

Abstract

Cow teat health plays a vital role in maintaining the quality of the produced milk and the health of the dairy cows. One of the widely used traditional methods is to manually examine the health of the cow teat but these manual methods could be highly time-consuming and prone to human errors. Alternatively, recent advancements and research in the field of Artificial Intelligence have opened up several opportunities to use this technology in the field of disease detection providing faster results with high accuracy. In this paper, I will present how we can classify cow teat health by leveraging deep-learning neural network architectures and the impact on detection accuracy by adding different layers. First, I will present the use of a convolutional layer-based neural network. Second, I will present the impact on accuracy by the inclusion of Residual blocks along with the CNN layer. Lastly, I will present, how we can improve accuracy by including the inception layer into the neural network. Results will demonstrate the impact on the model accuracy by including different layers in the neural network for better classification of the teat health on unseen test images. Along with the mentioned three layers, there are other techniques also been tested and implemented wherever applicable like data augmentation, and batch normalization. In order, to prepare and train this model we have used a total of 1147 digital images. These images are classified into four different labels. We have used industry best practices to train a good model that can be used on test images to quickly predict accurate labels of the images. We have a set of 380 test images that can be used to test and calculate the model accuracy. The results indicate that our trained model, when tested on 380 unseen test images, has successfully achieved a classification accuracy of 63.91 percent which can be beneficial for faster and more efficient prediction.

1. Introduction

With recent advancements in the field of artificial intelligence, disease detection using artificial intelligence-based

techniques like machine learning, deep learning techniques, and generative AI has shown tremendous results. Influenced by the remarkable results of Artificial Intelligence in the health care field, I will present my research where I will try to find an AI-based solution to classify dairy cow teat health conditions, this information can be utilized to take necessary action within time that will help in maintaining the quality of the produced milk and the health of the dairy cows which in turn leads to more revenue generation and less expenses on the medical health condition of the dairy cows.

This paper can act as a foundation block for similar problems. As a part of this research, I have used deep-learning neural networks to identify and classify the health of dairy cows on a given set of training images from the SCTL paper. I have built a custom neural network using PyTorch. I initiated the research by creating a simple deep-learning neural network by using two convolutional layers to understand the feasibility of classifying the cow teat problem statement. This model provides me with the base classification result and sets a tone that I'm in the right direction and we can use deep learning-based neural networks.

As a next step, I have introduced multiple other layers and accuracy improvement techniques to further enhance the classification accuracy of the trained model. The final proposed model uses convolutional layers, residual blocks, inception layers, batch normalization, and data augmentation techniques to further improve the accuracy of the model on unseen data.

To train and prepare the final model, I have a total of 1149 labeled cow teat health images where images are stored in four different folders named Score1, Score2, Score3, and Score4. By using the industry's best practice, I have split these images into training and validation data sets where the training set consists of 80 percent of the total images which is equal to 919 images and the validation set consists of the remaining 20 percent of the total images which is equal to 230 images. The training and validation accuracy achieved from the final model are 65.50 percent and 64.50 percent respectively. After this, I passed the test dataset which contains 380 images for the prediction and I

have successfully achieved 63.91 percent accuracy.

Further, improvement can be done in the custom neural network to achieve comparable and better accuracy as achieved by VGG16 [1] and SCTL [3].

2. Related Work

Deep learning is one of the growing fields and a lot of research is going into this. Due to this, there are a lot of related work in progress. A few of the which I found beneficial are as follows:

The paper "Separable Confident Transductive Learning for Dairy Cows Teat-End Condition Classification" by Youshan Zhang, Ian R Porter, Matthias Wieland, and Parminder S Basran explains the importance of milk quality and the health of the mammary gland. In their research, they have used "Separable Confident Transductive Learning" which is based on a convolutional neural network, with the intent to increase the feature differences in the image with different classifications of hyperkeratosis. Using these techniques, they were able to successfully achieve an overall accuracy of 77.6 percent.

Another paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" by Simonyan also provided great insight which helped to understand the impact of convolutional filters. Their research explores how the depth of convolutional networks affects accuracy in large-scale image recognition. By using small (3x3) convolution filters, they achieved significant improvements by increasing the network depth to 16-19 layers. With this, they were able to represent the depth that is beneficial for classification accuracy.

3. Methods

Current research in the field of deep learning and neural networks to identify and classify diseases played a crucial role which acts as the notification to create and establish a custom neural network. The method used to create a custom neural network involves various steps starting with the data acquisition which is facilitated by the SCTL [3] followed by the data loading, data loading data pre-processing, model development, model training, and model evaluation.

3.1. Data Acquisition

First, we have explored the source of data SCTL [3] and downloaded the training and test images. Once the images were downloaded successfully, I studied the directory structure how the data is stored in different folder structures, and how many images are available in each of the folders to get a basic understanding which will help to prepare data loaders.

3.2. Data Loading and Data Processing

After understanding the data, I created two custom data loaders, one for training images and one for test images due to the different directory structures the data is stored. The data loader function is designed to handle custom images and classify them. The data loader is capable of applying transformations as well. In my case, I have applied a 30-degree rotation transformation. As a part of the data loader, the incoming images are also been transformed to a defined size and then the normalization is also been applied to bring pixel values within the range [0.1]. Similarly, another function was defined for the test.

3.3. Model Development

With the intent to resolve the problem of dairy cow teat health condition, I started to develop a deep neural network that consists of various layers. The proposed neural network architecture is built using PyTorch which can be used for image classification. The following are the main components of the proposed neural network:

Inception Layer

Inception is a convolutional neural network for assisting in image analysis and object detection, and got its start as a module for GoogLeNet. It was originally introduced during the ImageNet Recognition Challenge. The design of Inception was intended to allow deeper networks while also keeping the number of parameters from growing too large. To utilize the inception, we have created a class called inception that takes two input arguments. It consists of five convolutional layers: conv1x1, conv3x3 reduce, conv3x3, conv5x5 reduce, and conv5x5. It also contains one pooling layer. At last, the forward method applies the five convolutional layers and the pooling layer to the input and then concatenates the outputs of the layers.

Residual Layer

A Residual Neural Network is a deep learning model in which the weight layers learn residual functions about the layer inputs. This network skips connections that perform identity mappings, merged with the layer outputs by addition. It behaves like a Highway Network with gates open through strongly positive bias weights. Due to this, it becomes possible for This enables deep learning models to be built with tens or hundreds of layers to train easily and obtain better accuracy. To utilize the residual layer, we have created a class called residual block that takes three input arguments. The proposed residual block consists of two 3x3 convolutional layers. It further uses two ReLU activation functions and a shortcut connection. The shortcut connection skips over the convolution layers if the input and output dimensions match finally returning the result as the block's output.

Custom Neural Network

The custom neural network architecture consists of Inception Layers and Residual Blocks. It starts with a convolutional layer (conv1) followed by a ReLU activation function (relu1) and a max pooling layer (pool1). Then, it uses two Inception Layers (inception1 and inception2) distributed with two Residual Blocks, each followed by a max pooling layer. Finally, it applies multiple fully connected layers (fc1, fc2, fc3) with ReLU activation functions (relu3, relu4) and a dropout layer (dropout) to extract high-level features and output the final prediction.

Regularization

Dropout is a regularization technique that randomly sets a fraction of the input units to zero during the training process. Adding dropout as a regularization helps to prevent the model from overfitting to the training data and improves its generalization performance percent. Normally the dropout rate is set between 0.2 and 0.5. In our custom neural network, we have also used dropout as one of the regularization techniques to avoid overfitting. We have used a dropout rate of 0.2 to the input tensor, which means that 20 percent of the input units will be randomly set to zero during training.

Architecture Diagram

To research and resolve daily cow teat health condition, the following custom architecture has been created.

```
Input (3, 224, 224)
-- Conv2d(3, 256, kernel_size=3, stride=1, padding=1)
|-- ReLU
-- MaxPool2d(kernel_size=2, stride=2)
-- ResidualBlock(256, 256)
|-- Conv2d(256, 256, kernel_size=3, stride=1, padding=1)
|-- ReLU
|-- Conv2d(256, 256, kernel_size=3, stride=1, padding=1)
|-- ReLU
-- Shortcut (Sequential)
|-- Conv2d(256, 256, kernel_size=1, stride=1)
|-- ReLU
-- MaxPool2d(kernel_size=2, stride=2)
-- ResidualBlock(256, 128)
|-- Conv2d(256, 128, kernel_size=3, stride=1, padding=1)
|-- ReLU
|-- Conv2d(128, 128, kernel_size=3, stride=1, padding=1)
|-- ReLU
-- Shortcut (Sequential)
|-- Conv2d(256, 128, kernel_size=1, stride=1)
|-- ReLU
-- MaxPool2d(kernel_size=2, stride=2)
-- ResidualBlock(128, 64)
|-- Conv2d(128, 64, kernel_size=3, stride=1, padding=1)
|-- ReLU
|-- Conv2d(64, 64, kernel_size=3, stride=1, padding=1)
|-- ReLU
-- Shortcut (Sequential)
|-- Conv2d(128, 64, kernel_size=1, stride=1)
|-- ReLU
-- MaxPool2d(kernel_size=2, stride=2)
-- Linear(12544, 256)
|-- ReLU
-- Linear(256, 128)
|-- ReLU
-- Dropout(p=0.2)
-- Linear(128, 128)
|-- ReLU
-- Linear(128, 64)
|-- ReLU
-- Linear(64, 4)
```

3.4. Model Training

Model testing is one of the crucial steps in determining the effectiveness of the trained neural network model. As a part of this, once the model is trained on the training data,

we provide the model with the unseen data to make the predictions. Once the predictions are made on the unseen data then we try to verify the predictions against the actual value to calculate the model accuracy, precision, recall, and other evaluation metrics. It ensures that the model can easily generalize predictions to a new unknown dataset and further helps to identify potential overfitting, underfitting, and bias-related issues.

In this case, to train the model, I have used the widely used industry best practices where as a first step, I have split the whole training data which consists of 1149 images into training and validation datasets along with their corresponding labels. After splitting, training data consists of 80 percent of the total images, and the validation dataset consists of 20 percent of the total images which comes to be equal to 919 images and 230 images respectively. Additionally, we are also provided with the 380 test images, which can be used to evaluate the generalization of the model. Due to image processing and limited resources, batch processing technique has been used to train and validate the training dataset. A training loop has been established to iterate over a set number of epochs, during which the model is trained and evaluated. The training loop includes back passes, loss computation, and parameter updates using an optimizer.

3.5. Model Evaluation

Model evaluation is a crucial step to understanding the strengths and weaknesses of the model. In this case, after initializing a neural network, I have used CrossEntropy-Loss as a loss function and Adam as an optimizer with a defined value of learning rate. During the batch execution, training and validation metrics like loss and accuracy were calculated. Certain hyperparameters like the number of the epoch, batch size, kernel size, etc can be used to fine-tune the training process. The following hyperparameters have been used to train the model:

Table 1. Hyperparameter Used to train the final model

Parameter Name	Value
Learning Rate	0.001
Batch Size	0.4
Number of Epochs	0.20

Other evaluation metrics that can be used and beneficial as specified in cow paper [2]

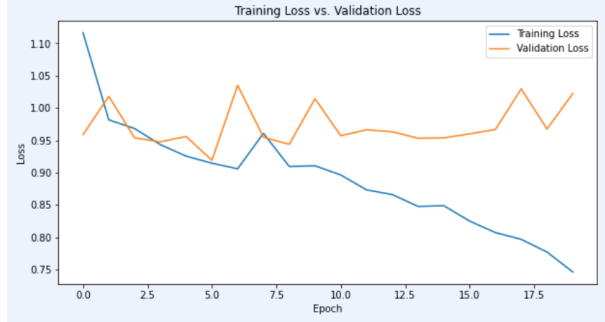
$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2, \quad (1)$$

where Y_i is the prediction value, and \hat{Y}_i is the orthogonal distance measurement.

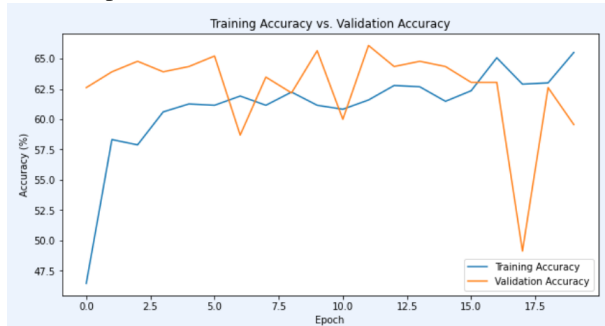
4. Results

Here are the observations and results achieved during the research performed by successfully creating a custom neural network using Pytorch.

The first graph is plotted between the Training Loss vs Validation Loss. The line graph indicates loss over the y-axis and the number of epochs across the x-axis. Overall, the graph shows a bit of inconsistent behavior between training and validation loss. As per the trends shown graph, training loss shows a gradual decrease as the number of epochs increases whereas the validation loss is inconsistent.



The second graph is plotted between the Training Accuracy vs Validation Accuracy. The line graph indicates accuracy in percent over the y-axis and the number of epochs across the x-axis. Overall, the graph shows a bit of mixed behavior between training and validation accuracy. As per the trends shown in the graph, training accuracy shows an increase in accuracy as the number of epochs increases whereas the validation accuracy is overall consistent within 58 and 65 percent.



4.1. Test Results

Here is the test result summary on training, validation, and test datasets.

Table 2. Summary

Dataset	of Images	Loss	Accuracy
Training Set	919	0.7464	65.51
Validation Set	230	1.0221	59.57
Test Set	380	NA	63.94

4.2. Model Summary

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 256, 224, 224]	7,168
ReLU-2	[-1, 256, 224, 224]	0
MaxPool2d-3	[-1, 256, 112, 112]	0
Conv2d-4	[-1, 256, 112, 112]	590,080
ReLU-5	[-1, 256, 112, 112]	0
Conv2d-6	[-1, 256, 112, 112]	590,080
ReLU-7	[-1, 256, 112, 112]	0
ResidualBlock-8	[-1, 256, 112, 112]	0
MaxPool2d-9	[-1, 256, 56, 56]	0
Conv2d-10	[-1, 128, 56, 56]	295,040
ReLU-11	[-1, 128, 56, 56]	0
Conv2d-12	[-1, 128, 56, 56]	147,584
Conv2d-13	[-1, 128, 56, 56]	32,896
ReLU-14	[-1, 128, 56, 56]	0
ResidualBlock-15	[-1, 128, 56, 56]	0
MaxPool2d-16	[-1, 128, 28, 28]	0
Conv2d-17	[-1, 64, 28, 28]	73,792
ReLU-18	[-1, 64, 28, 28]	0
Conv2d-19	[-1, 64, 28, 28]	36,928
Conv2d-20	[-1, 64, 28, 28]	8,256
ReLU-21	[-1, 64, 28, 28]	0
ResidualBlock-22	[-1, 64, 28, 28]	0
MaxPool2d-23	[-1, 64, 14, 14]	0
Linear-24	[-1, 256]	3,211,520
ReLU-25	[-1, 256]	0
Linear-26	[-1, 128]	32,896
ReLU-27	[-1, 128]	0
Dropout-28	[-1, 128]	0
Linear-29	[-1, 128]	16,512
ReLU-30	[-1, 128]	0
Linear-31	[-1, 64]	8,256
ReLU-32	[-1, 64]	0
Linear-33	[-1, 4]	260

=====

Total params: 5,051,268
Trainable params: 5,051,268
Non-trainable params: 0

=====

Input size (MB): 0.57
Forward/backward pass size (MB): 370.67
Params size (MB): 19.27
Estimated Total Size (MB): 390.51

=====

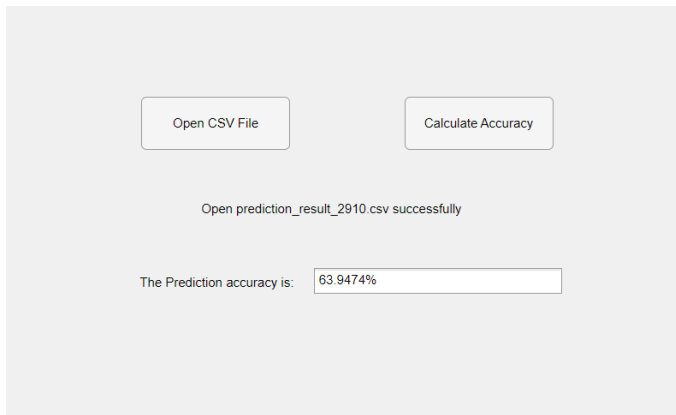
4.3. Datasets

The dataset used for this research has been referenced and used from original research performed by SCTL [3]. The available dataset contains two zip files. The first file is referenced as a training file which consists of 1149 images organized under different folders as per their respective labels. All images are categorized under four different labels 1, 2, 3, 4. In this research, we have split the training dataset into training and validation datasets into an 80:20 ratio. We are also provided with the test images which can be passed to the trained model to perform predictions. These images don't have labels. In total, we have 380 test images that can be used to verify the model's accuracy.

Table 3. Dataset Summary

Dataset Type	Percent of Images	Number of Images
Training	80	919
Validation	20	230
Test	100	380

5. Accuracy Calculated by Tool



Open CSV File Calculate Accuracy

Open prediction_result_2910.csv successfully

The Prediction accuracy is: 63.9474%

6. Discussion

The final neural network follows a Convolutional Neural Network (CNN) architecture that consists of multiple other layer and regularization techniques like residual block, dropout layers etc. At the start, the network receive input images of size (3, 224, 224).

Then the next layer is a convolutional layer with 256 filters and a kernal size of 3x3 and padding equal to 1. This layer is followed by a Rectified Linear Unit (ReLU) activation function, which helps extract important features from the input image. After the convolutional layer, a max-pooling operation layer is applied to downsample the feature maps and reducing their spatial dimensions.

The network further consists of three Residual Blocks. Each Residual Block consists of two convolutional layers with ReLU activation. These blocks facilitate the capture of hierarchical features and mitigate the vanishing gradient problem. A shortcut connection is used to add the input to the output of each block, allowing for smoother gradient flow during training.

After this, one more additional max-pooling layers is included for further downscaling of feature maps and reducing the spatial resolution. After this a series of fully connected layers has been added where each layer followed by the ReLU activation function. These layers gradually reduce the dimensionality of the data, transforming it into a form suitable for cow teat classification into one of the 4 classes. To prevent overfitting, a dropout layer with a dropout probability of 0.2 is added to the network.

In the end, the final layer is a Linear layer with four output units. It is done because the network is designed for a classification task with four different classes of dairy cow teat heanht condition, where it makes predictions based on the features extracted from the input image.

7. Conclusion

In conclusion, I have created a custom Convolutional Neural Network (CNN) architecture using PyTorch which is specifically built for the classification of dairy cow teat conditions based on image data. This architecture consists of different layers and other components, including convolutional layers, batch normalization layers, residual blocks, pooling layers, and fully connected layers. This custom neural network successfully achieved the test accuracy of 63.94 percent on completely unseen images outcome represents a promising achievement, especially given the complexity of the problem. During the research, it has been evident that accuracy can be achieved if the right layers and parameters are used properly. Hence, this research still has lots of opportunities to further improve and figure out other different layers that can play a vital role in improving the accuracy of the model. One such layer could be Inception layer which has a lot of potential to further improve the accuracy of the model and achieve the required minimum accuracy of 66.8 percent. Therefore, by further refining the model through additional layers, associated hyperparameters and comprehensive training, we can achieve an even higher level of accuracy, that has the potential to make people trust and use our product for rendering the system increasingly valuable for practical, real-world applications.

References

- [1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. [2](#)
- [2] MC Wiltbank, AHMET Gümen, and Roberto Sartori. Physiological classification of anovulatory conditions in cattle. *The-riogenology*, 57(1):21–52, 2002. [3](#)
- [3] Youshan Zhang, Ian R Porter, Matthias Wieland, and Parminder S Basran. Separable confident transductive learning for dairy cows teat-end condition classification. *Animals*, 12(7):886, 2022. [2](#), [4](#)
- [4] Maurya, Kanchan. (2023). Image classification of Cow Teat by implementing Convolution Neural Network using PyTorch and Residual Block.