CowStallNumbers: A Small Dataset for Stall Number Detection of Cow Teats Images using Object Detection Model

Shilpa Kuppili
Yeshiva University
3 PUBLICATIONS **0** CITATIONS

CowStallNumbers: A Small Dataset for Stall Number Detection of Cow Teats Images using Object Detection Model

Shilpa Kuppili
Yeshiva University
shilpakuppili@i1.org

Abstract

To further advance cow stall number detection, we present in this study a tiny cow stall number dataset called CowStallNumbers, which is extracted from cow teat recordings. There are 1042 training photos in this dataset and 261 test pictures with stall numbers between 0 and till 60.

1. Introduction

With the rising demand for milk and milk products in recent years, the dairy industry has experienced significant expansion. The management of the production and health of the herd is one of the most important parts of dairy farming. The cows must have a pleasant environment, the right nutrition, and routine health checks in order to increase milk output and maintain the cows' health. We have already looked at the classification of several teat classifications [25, 26]. These teat photos, however, were manually cropped after being snapped with a phone. We pointed a camera in the direction of the teat end areas in order to save time [27]. We suggested extracting the important teat frames from the video photos, which creates additional issue in that we are unable to identify the cows' ID until we can identify the video's stall number. As a result, a model for identifying the stall numbers needs to be created.We improve the ResNet model for cow stall detection in this research. We test our approach on a small collection of photos of cows that have annotations for the locations of their stalls based on ground truth. Our findings demonstrate that our approach has a high degree of accuracy when it comes to identifying cow stalls, and it can be a useful tool for dairy farmers looking to streamline their processes. The health and productivity of cows can be enhanced, which will raise the sustainability and profitability of cow stalls by enabling accurate and efficient identification of them. We make two contributions: first, we want to advance the field of cow stall number detection by presenting a tiny dataset called Cowstall-Number.To determine the location of the stall number and identify various stall numbers, we adjusted the pre-trained ResNet model.

2. Related Work

When used for cow tail detection, the quicker Region-based Convolutional Neural Network (R-CNN) method outperforms the R-CNN algorithm in terms of detection speed and accuracy.Still, one drawback of the challenge these object detection systems face is identifying items in specific environmental circumstances,like dim illumination or inadequate contrast.Due to the fact that cows tend to blend in with their environment or have their distinguishing characteristics hidden by shadows or other sight impediments. Researchers have looked into thermal imaging cow detection as a solution to this problem. With the use of infrared technology, thermal imaging can identify an object's heat signature with greater accuracy, especially in low-light or low-contrast settings. Thermal imaging can also identify cows even in circumstances where they are only partially visible, like in the tall grass or behind shrubs. Despite improvements in cow detection technology, obstacles still need to be addressed. Finding cows in big herds, for instance, can be challenging because cows can crowd closely together and partially block each other's views. Additionally, various breeds of cows may possess distinct physical traits that call for particular detection algorithms. In the paragraphs that follow, we will go over the relevant work on object detection. The region proposal algorithm was used by Girshick to create candidate object regions in their R-CNN model. [Positions 1 and 2 of the boxes Box position 3, Box position 4] denotes the bounding box location's [x, y, height, width]. The stall number class is the class name.When there is empty space, the stall number in the picture cannot be identified. system (CNN) for categorization [8]. Later, Girshick put forth an enhanced Fast R-CNN model edition that provides faster and more accurate detection by using a shared feature map for both region proposal and classification. Ren and colleagues created a faster R-CNN model, an additional R-CNN extension that creates a Region Proposal Network (RPN) to produce region proposals more effectively, establishing it as the first technique for real-time object detection.(YOLO-you only look once) is a real-time object detection technique that uses the entire image to produce predictions in real time, based on a individual CNN that forecasts each object's class and position at the same time. Redmon and colleagues introduced a Single The Shot MultiBox Detector (SSD) model predicts the class and location of objects by generating proposals for them one neural network only. SSD outperforms other edition of R-CNN that provides faster and more accurate detection by using a shared feature map for both region proposal and classification. Ren and colleagues created a faster R-CNN model, an additional R-CNN extension that creates a Region Proposal Network (RPN) to produce region proposals more effectively, establishing it as the first technique for real-time object detection.Yolo, you only look once is quicker R-CNN and surpasses YOLO in accuracy. Lin and RetinaNet, presents a novel focal loss function that upweights difficult examples and downweights easy examples. An expansion is Mask R-CNN of Faster R-CNN, which can segment the detected data by adding a branch to predict segmentation masks for each objects.DetNet uses convolutional layers that are aware of deformations in objects and changes in viewpoint. DenseBox uses dense regression to forecast object bounding box coordinates. CoupleNet outperforms other real-time detection systems by predicting an object's size and position in a coupled manner techniques. A Reverse Connection was suggested by Kong with Objectness Prior Networks (RON) model,which gathers object context data via a reverse connection creation of proposals. The object RefineDet has multiple stages. detection technique that gradually narrows down the area and dimensions of proposed objects. Decoder for Edge Sampling (ESD) uses deep neural networks with edge sampling to detect boundaries and object edges.Using neural architecture search, a feature pyramid network with enhanced object detection capabilities is created. There are various variations of YOLO methods as well. In 2016, YOLOv1 was released.In 2017, YOLOv2 (YOLO9000) was made available and utilized an architecture for the Darknet-19 network and integrated batch anchor boxes and normalization. Moreover, multiscale predictions were introduced, making it possible to detect objects at various scales. 2018 saw the release of YOLOv3 , which was used for residual connections, feature pyramid networks (FPNs), and a Darknet-53 network architecture. Additionally, it dynamically scaled anchor boxes, which improved detection across a range of scales. YOLOv4 was made available utilizing a CSPDarknet-53 network architecture, added a number of enhancements in 2020, including spatial pyramid pooling and made use of an intricate pipeline for data augmentation.Additionally, comprised effective training methods like as focal loss and augmentation of mosaic data.YOLO was not a fork from PyTorch, but rather an implementation of the original Darknet, with its PANET neck and CSP backbone. One of the main enhancements is auto-learning anchors in bounding boxes. In 2022, YOLOv6 became available and was a single-stage object detection framework with excellent performance, an efficient hardware-friendly design, and a focus on industrial applications. When YOLOv7 was released in 2022, the quantity of parameters was decreased by 75 percent, uses 36 percent less processing power, and yields 1.5 percent greater average precision (AP) in comparison to YOLOv4. YOLOv7-X outperforms inference speed by 21 FPS. YOLOv5–X. The YOLOv8 model operates quickly, precisely, and simple to use and suitable for a variety of object detection and segmentation tasks for images. It is instructable on large datasets and are compatible with a range of hardware platforms, from GPUs to CPUs.

3. Data Collection

The images of the stall numbers are taken from cow teat videos, which are made in order to assess the health of the cow teats standing. First, we used the unsupervised few-shot method UFSKEF model for key frame extraction to extract the coarse key frames for stall numbers. Next, we verified by hand the incorrect key frame images were eliminated from these key frames.Three sample stall numbers are displayed where 0 denotes that the stall numbers are invisible to us.Statistics of our CowStallNumber dataset: Training 1042, Test 261, Total 1303

4. Methods

The code defines a custom PyTorch Dataset class called EnhancedImageDataset for working with image data. Let's break down the key components of the code: init Method: Parameters: annotation path: Path to the CSV file containing image annotations (e.g., file names, labels). images folder: Path to the folder where the images are stored. transformation: Optional image transformation to be applied (e.g., resizing, normalization). Functionality: Initializes the dataset by reading image annotations from a CSV file using pd.read csv. Sets the path to the image folder and an optional image transformation. len Method: Functionality: Returns the total number of samples in the dataset, which is the length of the image annotations getitem Method: Parameters: index: Index indicating the position of the sample in the dataset. Functionality: Constructs and returns a single sample (image and label) based on the specified index. Constructs the file path of the image using the images folder and the file name from the annotations. Opens the image file using PIL.Image.open and converts it to RGB format. Retrieves the label from the annotations (assuming it's at index

5) and converts it to a PyTorch tensor of type long. Applies the specified transformation to the image if provided. Returns the preprocessed image and its label as a tuple. This dataset class is designed to be used with PyTorch's DataLoader to efficiently load and preprocess batches of images during training or evaluation. The transformation parameter allows for applying image transformations, such as resizing or normalization, as needed for the specific task at hand. The class assumes that image annotations are stored in a CSV file with file names and corresponding labels. Further, code defines a function calculate iou that calculates the Intersection over Union (IoU) between two bounding boxes. IoU is a metric commonly used to evaluate the performance of object detection algorithms.

The function takes two bounding boxes, box1 and box2, as input arguments.

The coordinates of the bounding boxes are assumed to be in the format x min, y min, x max, y max where (x min, y min) are the coordinates of the top-left corner, and (x max, y max) are the coordinates of the bottom-right corner.

The code calculates the intersection area (inter area) between the two bounding boxes. If there is no intersection (boxes do not overlap), the intersection area is set to 0.

The total area of each bounding box (box1 area and box2 area) is calculated.

The union area (union area) is computed as the sum of the areas of both boxes minus the intersection area.

Finally, the IoU is calculated as the ratio of the intersection area to the union area. If the union area is 0 (indicating no overlap), the IoU is set to 0.

This function is useful for evaluating the accuracy of object detection models by comparing the predicted bounding boxes to the ground truth bounding boxes. The IoU ranges from 0 (no overlap) to 1 (perfect overlap).

Datasets are prepared for training and testing by applying advanced data transformations using PyTorch's transforms module. transforms.Compose: This function is used to compose a series of transformations. It takes a list of transformation functions and applies them sequentially to the input data.

Transformation list:

transforms.Resize((256, 256)): Resizes the image to the specified size (256x256 pixels). transforms.RandomHorizontalFlip(): Randomly flips the image horizontally. transforms.RandomVerticalFlip(): Randomly flips the image vertically (new addition). transforms.RandomRotation(15): Randomly rotates the image by up to 15 degrees. transforms.ColorJitter(): Randomly adjusts brightness, contrast, saturation, and hue. transforms.RandomResizedCrop(224): Randomly crops and resizes the image to the specified size (224x224 pixels, new addition). transforms.ToTensor(): Converts the image to a PyTorch tensor. EnhancedImageDataset: This is a custom dataset class, likely designed to work with image data and corresponding annotations. It uses the specified transformations for data augmentation during training and testing. annotation path: Path to the CSV file containing image annotations. images folder: Path to the folder containing image files. transformation=advanced transform: The advanced transform is applied to the images during loading. PyTorch DataLoader objects are created for the training and testing datasets. It specifies the batch size and whether to shuffle the training data.A pre-trained ResNet-50 model is loaded and modify the final fully connected layer to match the number of classes in the dataset.The loss function (cross-entropy loss), optimizer (Adam), and a learning rate scheduler (ReduceLROnPlateau) is set up. Class weights are applied to handle imbalanced datasets. ResNet, short for Residual Network, is a type of deep neural network architecture designed to address the vanishing gradient problem that can occur in very deep neural networks. ResNet was introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their paper "Deep Residual Learning for Image Recognition" in 2015.

Here are the key components and concepts of the ResNet model:

Residual Blocks:

The core idea behind ResNet is the use of residual blocks. A residual block contains a shortcut connection (or skip connection) that bypasses one or more layers. The output of the residual block is the sum of the input to the block and the output of the block's internal layers. This allows gradients to flow directly through the shortcut connection, mitigating the vanishing gradient problem. Identity Shortcut Connection:

In the simplest form, the shortcut connection simply performs an identity mapping, meaning it passes the input directly to the output. If the dimensions of the input and output do not match (e.g., due to different numbers of filters), a linear projection is used to match the dimensions. Residual Learning:

The network is trained to learn residuals, i.e., the difference between the predicted output and the input. The shortcut connection then adds this residual to the input. Bottleneck Architecture:

To improve computational efficiency, ResNet often uses bottleneck blocks, especially in deeper layers. These blocks consist of three convolutional layers: a 1x1 convolution (dimension reduction), a 3x3 convolution, and another 1x1 convolution (dimension restoration). Skip Connections Across Stages:

In deeper ResNet architectures, skip connections are not only within the same stage but also across different stages of the network. These connections help in the flow of information across multiple scales. Pre-Trained Models:

ResNet models pre-trained on large datasets, such as

ImageNet, have shown impressive generalization capabilities and are widely used for various computer vision tasks. Variants: There are different variants of ResNet, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, which differ in the number of layers. The numbers in the names indicate the total number of layers in the network. ResNet architectures have become a cornerstone in deep learning, and their residual learning concept has been widely adopted in the design of various neural network architectures. The skip connections and residual blocks enable the training of very deep networks, leading to improved performance in image classification, object detection, and other computer vision tasks. some applications of the ResNet model:

Image Classification:

ResNet was originally designed for image classification tasks, and it has demonstrated superior performance in large-scale image classification competitions, such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

ResNet is employed in tasks related to scene understanding, including scene classification and scene parsing.

ResNet architectures can be used for tasks related to document analysis, such as document classification, text extraction, and layout analysis. Further, we define the training loop. It iterates over the specified number of epochs, loads batches of training data, computes the loss, performs backpropagation, and updates the model parameters. The learning rate scheduler is also applied to adjust the learning rate during training.

The epoch loss history list keeps track of the training loss for each epoch, which can be useful for plotting the training progress. The model is set to evaluation mode. During evaluation, the model behaves differently than in training mode. For example, layers like dropout layers work differently in training and evaluation modes. These lists are initialized to store the predicted and true labels, respectively. Within the torch.no grad context, the model is applied to the test dataset. For each batch of inputs and labels in the test loader, it moves the inputs to the device (GPU or CPU), computes the model outputs, and then extracts the predicted labels by selecting the class with the maximum probability using torch.max. The predicted and true labels are then appended to the respective lists. The accuracy is calculated by comparing the predicted labels with the true labels, and the result is printed. The accuracy score function from scikit-learn is used for this purpose. The accuracy is expressed as a percentage. Further, import the matplotlib.pyplot module, which provides a collection of functions to create various types of plots and visualizations. The figsize parameter defines the width and height of the figure in inches. The plt.plot function is used to create the line plot. The range(1, epochs + 1) generates the x-axis values representing the epochs, and epoch loss history provides the corresponding y-axis values, which are the training losses at each epoch. The marker='o' argument adds circular markers at each data point for better visibility.

5. Results

A line plot is generated showing how the training loss changes with each epoch during the training process. It helps visualize the training progress and can be useful for identifying patterns or potential issues, such as overfitting or underfitting.The accuracy is calculated as 73.65 percent .

5.1. Datasets

The cow stall number dataset called CowStallNumbers,has been extracted from cow teat recordings. There are 1042 training photos in this dataset and 261 test pictures with stall numbers between 0 and till 60.

6. Discussion

YOLOv3 [21] was released in 2018 and used a Darknet-53 network architecture, introduced feature pyramid networks (FPNs), and used residual connections. It also used anchor boxes with dynamic scaling, leading to better detection at different scales. YOLOv4 [3] was released in 2020, which used a CSPDarknet-53 network architecture, added various improvements such as spatial pyramid pooling, and used a complex data augmentation pipeline. YOLOv4 also included efficient training approaches such as mosaic data augmentation and focal loss. YOLOv5 [12] was a PyTorch implementation rather than a fork from the original Darknet, which has a CSP backbone and PANET neck. The major improvement includes auto-learning bounding box anchors. YOLOv6 [14] was released in 2022 and was a single-stage object detection framework dedicated to industrial applications, with hardware-friendly efficient design and high performance. YOLOv7 [23] was released in 2022 and reduced the number of parameters by 75 percent, requires 36 percent less computation, and achieves 1.5 percent higher AP (average precision) compared with YOLOv4. YOLOv7-X achieves a 21 FPS faster inference speed than YOLOv5-X. The YOLOv8 [1] model is fast, accurate, and easy to use, which can be applied to a wide range of object detection and image segmentation tasks. It can be trained on large datasets and run on various hardware platforms, from CPUs to GPUs.

7. Conclusion

We create a small cow stall number dataset named CowStallNumbers with the goal of advancing cow stall number detection. We fine-tuned a model and augmented the datasets.

References

[10] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. Densebox: Unifying landmark localization with end to end object detection. arXiv preprint arXiv:1509.04874, 2015. 2 [11] Xiaoping Huang, Xinru Li, and Zelin Hu. Cow tail detection method for body condition score using faster r-cnn. In 2019 IEEE International Conference on Unmanned Systems and Artificial Intelligence (ICUSAI), pages 347–351. IEEE, 2019. 1 [12] Nelson Joseph and Solawetz Jacob. Yolov5 is here: Stateof-the-art object detection at 140 fps. https://blog. roboflow.com/yolov5-is-here/. Accessed: 2023- 03-15. 3 [13] Tao Kong, Fuchun Sun, Anbang Yao, Huaping Liu, Ming Lu, and Yurong Chen. Ron: Reverse connection with objectness prior networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5936–5944, 2017. 2 [14] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6: A single-stage object detection framework for industrial applications. arXiv preprint arXiv:2209.02976, 2022. 3 [15] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet: A backbone network for object detection. arXiv preprint arXiv:1804.06215, 2018. 2 [16] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, ´ Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2117–2125, 2017. 2 [17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In ´ Proceedings of the IEEE international conference on computer