https://doi.org/10.2352/EI.2023.35.7.IMAGE-279

Eidetic recognition of cattle using keypoint alignment

Manu Ramesh[1], Amy R. Reibman[1], Jacquelyn P. Boerman[2]
[1]School of Electrical and Computer Engineering, [2]Department of Animal Sciences,
Purdue University, West Lafayette, Indiana, USA

Abstract

We present a Holstein cattle ReID system that identifies individual cows in the top view FullHD IP camera videos given a catalog (cattlog) of those cows - a set of bit vectors. This system is designed to uniquely identify cows with a single training example per cow, with zero training time for adding new cows for recognition. Taking inspiration from face landmark detection, alignment, and morphing techniques, we build a keypoint detector that detects keypoints on the cow bodies in the top view, which are then used to extract and align cow body pixels to a predetermined template. The aligned images are then binarized and stored as bit vectors in the cattlog dictionary. Queried cow images are passed through the same keypoint detection and template alignment mechanism to get the corresponding bit vectors; these are then searched for in the cattlog dictionary to find the cows that are the closest match to the ones being queried. We also describe a mechanism we used to curtail cow images with keypoints detected in the wrong places from reaching the cow ID prediction stage and another that interpolates missing keypoints to make more instances eligible for cow ID prediction. We measure the recognition accuracy using Top-K prediction accuracy on a video dataset with 148 cows, with one cow per video. We find the Top-1 accuracy to be 61.5% and Top-4 to be about 83%.

Introduction

The ability to uniquely recognize cows helps in monitoring their health, milk production, behavior patterns on an individual basis. It is also useful in tracking their ownership. It would be an added advantage if this recognition can be achieved instantly by a machine as all the tasks listed above could be automated and that could save time money or effort. Existing approaches for cattle recognition include physical methods such as branding, tattooing, ear notching and ear tags [2]. These traditional methods are intrusive and not machine readable. Although electrical methods that use Collar RFIDs, RFID ear tags or injectable RFIDs provide a faster, machine readable alternative, they too are intrusive because they need devices to be physically mounted on the cow and besides, they are expensive. Other methods that use animal biometrics such as iris or muzzle patterns require significant time to sample the patterns and recognize cows. Lately, many computer vision techniques are being developed for individual cattle recognition that identify cows by their coat patterns [5] or their faces [1]. Computer vision techniques are non-intrusive, fast, and not as expensive as RFID based systems. They could work with even a single camera to track the positions of multiple animals simultaneously. Despite the many advantages of these computer vision approaches, there is one disadvantage common to most of them. The fact that most of them use deep learning makes them extensively slow to train.

A deep learning model identifies individual cows as separate output classes. In an ordinary dairy setting, new, previously unseen cows are added to and older cows get removed from the farm very often. When a deep learning model is employed in such a setting, the model needs to be retrained to achieve the same level of accuracy each time a new cow is added to the farm. This time-hungry retraining task is onerous. Yet another negative is the amount of training data needed for deep learning based individual cow identification to work. For these models to work effectively across variation in lighting, camera angle, size of cows etc., many training image samples of each individual cow are necessary, which are inconvenient to collect, and store. Hence, we seek a computer vision based cow ID predictor that can learn to recognize new cows with minimal training data and with zero retraining time. Such a model could be easily extended towards an open world cow predictor - a system that automatically scans and adds new cows to its cattlog, that is, its catalog of known cows. Additionally, we would like the method to be robust to variation in lighting, camera angle, size of cows, the degree to which their backs are bent, etc.

Existing computer vision methods for cattle recognition divide the task of recognition into individual localization and identity prediction. The methods differ in how both these stages are implemented. Bounding box detection is predominantly used for cow localization. Use of rotated bounding boxes forms the basis of cattle localization in [5]. This exact approach is also used in [6] as it is an extension of their previous work. The approach required them to build a custom rotated bounding box detector as no alternative existed at the time. As a result, the detector requires custom annotations that are not compatible with popular annotation formats such as COCO, YOLO or PascalVOC, making it difficult for others to replicate or extend the work. Further, despite rotated boxes eliminating more background than the edge aligned bounding boxes, the cows inside them could still be bent to varying degrees. An ideal recognition system should be able to identify these cows not only when they are straight but also when they are bent. Since the work uses deep learning for recognition, a neural network trying to memorize how a cow looks in a bounding box must also learn to identify it when it is bent. This directly translates to augmenting the training dataset with images of cows taken at different degrees of bend or devising a system to synthetically generate these images during training time. Both of these are not trivial. Thus, in these two-stage recognition systems, the cow localizers learn to localize cows to bounding boxes and the cow ID predictors learn to recognize them in any arbitrary orientation within the bounding boxes. So, the second stage in essence performs two tasks - one of localizing a cow within the bounding box and the other of finding its identity, despite it not being trained to do these separately.

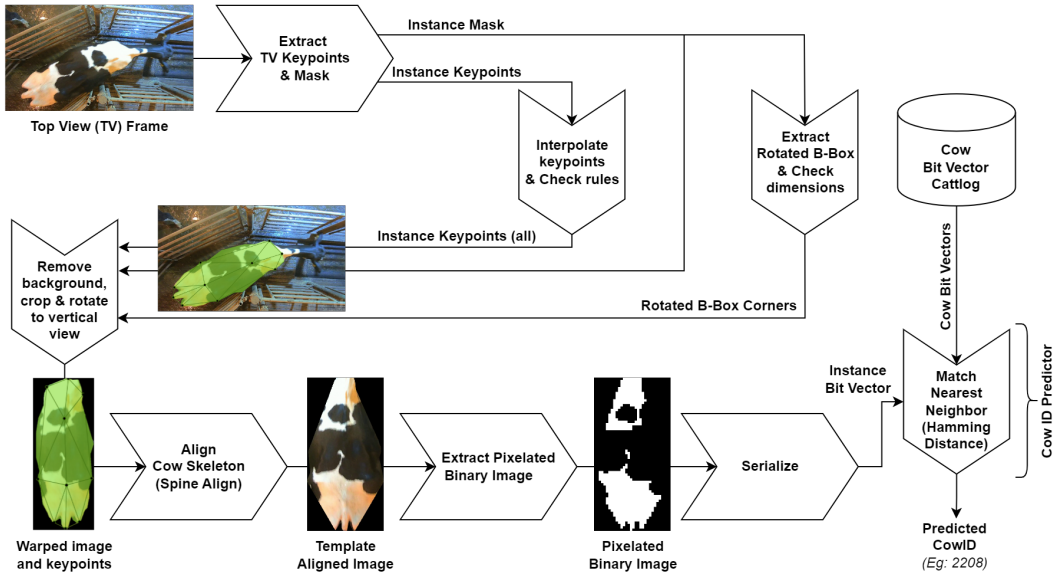

Figure 1: Block diagram of prediction at the frame level.

In this work, we follow a simple philosophy - train the stochastic, learning-based models to replicate tasks which humans are intuitively good at - tasks such as identifying parts of a cow's anatomy - and let the deterministic algorithms that are not learning-based perform tasks which computers are historically good at - tasks such as memorizing the appearances of hundreds of cows and finding the best match. Essentially we move the part of the cow ID predictor that localizes a cow within the bounding box to the first stage, and hence move the associated complexity with it. This allow us to train a complete localization system that is agnostic to individual cow identities. This training is expensive and can be done just once. The cow ID prediction system, which is now bloat free, can be adapted to learn new cow IDs with zero cost.

Face alignment techniques using landmark points [8, 4], have influenced face morphing strategies [3]. Keypoints on cows in the side view were used to extract their structure information in [11]. Taking cues from these works, we build the first ever cattle recognition system that detects and aligns keypoints in top view to align cows within the rectangular bounding boxes, and then converts the aligned images into bit patterns similar to QR codes for predicting the cow IDs. The following sections provide more detail.

Purdue Dairy Datasets and Cattlog

Data for experiments was collected at the Purdue Dairy during Summer 2021, both in the barn and in the holding area, using Ubiquiti Unifi UVC-G3 FullHD (1080p 30FPS) top view cameras. The barn, which has highly variable lighting conditions, is the area where cows walk around without much restriction and more than one cow can enter the view simultaneously. The holding area is the area in which cows walk one at a time, along a fixed path, so that they are seen in full view without obstruction. Data was also collected in the holding area on two separate days in Summer 2022 using the same cameras.

To create the training and testing sets for the cow recognition system, we use the multiple, hour-long, holding area videos of Summer 2022. We use the 153 cow videos from the first day for training and the 170 cow videos from the later date for testing. These videos are cut into segments to have just one cow in each of them. These 'cut videos' are used to form our Cow Videos Dataset, which is used for testing the recognition system at the video level. To form the corresponding Cow Images Dataset, one frame from each cut video was selected such that it contains an entire cow. Of the 170 cows in the testing set, 148 cows are also in the training set and the remaining 22 are exclusively in the testing set. We refer to these subsets as Test-InSet and Test-OutOfSet respectively.

We create yet another dataset, the Cow Keypoints Dataset, for the purpose of training our keypoint detector. 893 sampled images from Summer 2021 videos and holding area images from Day 1 of Summer 2022 collectively form its training set, and 170 images from Day 2 of Summer 2022 constitute its testing set. The 170 testing images are the same as those in the testing set of our Cow Images Dataset. These images are augmented with instance mask and keypoint annotations in COCO format [10] using COCO-Annotator. Bounding boxes, which are lower level annotations, come as a byproduct. It is important to note that there are 11 cows which the keypoint detector has not seen during training. We call this subset the Cow Keypoint Test-OutOfSet.

Implementation

Our goal is to convert each cow image into a 2D binary matrix so that it lends itself to a lightweight and immediate recognition. This approach was motivated by the idea that images of Holstein cows, in the top view with their black and white coat patterns, would look like QR codes when pixelated.

Figure 1 presents the algorithm used to predict the cow ID

given a video frame that contains a cow instance. In the first block, the keypoint and mask extractor scans the given image and returns a set of keypoints and a semantic mask for each cow instance it detects. For each instance, a rotated rectangle is fit to its semantic mask using OpenCV contour functions. Rotated rectangles, with edges parallel to the length of the cow, are a tighter fit and help eliminate more of the background compared to the standard bounding boxes that have edges parallel to those of the image, which are also detected by the mask extractor for free. To increase the chance of the entire body of the cow being visible in an instance considered for recognition, we filter out all instances whose rotated bounding boxes have areas and aspect ratios that do not fall within set limits.

Non-instance pixels in the image are eliminated using the semantic mask. In each cow instance, if not all keypoints are detected, the keypoint interpolator seeks to fill in the missing keypoints wherever possible. This set of instance keypoints (both detected and interpolated) is inspected for anomalies by testing against a set of rules. If the set breaks even a single rule, the instance is disqualified for prediction. Using a single warping step, the image is cropped to this rotated rectangle and the result is rotated to make the cow vertical with its tail down and neck up. The keypoints, also warped using the same mapping, are used to morph the resulting image into a standard template with preset image size and preset positions for each keypoint. This template-aligned image is then binarized, pixelated and serialized to obtain the instance bit vector. This bit vector is then searched for in the bit vectors dataset, or cattlog, and is matched to the one closest in terms of Hamming distance. The cow ID of the matched bit vector is the predicted cow ID.

The bit vector cattlog is created by accumulating the bit vectors generated in the exact way as described above, using just one annotated image per cow from the training set of Cow Images Dataset. For video level prediction, predictions from multiple video frames are collected and the one occurring most frequently is selected as the final prediction.

The following subsections elaborate on the procedures.

Mask and keypoint extraction

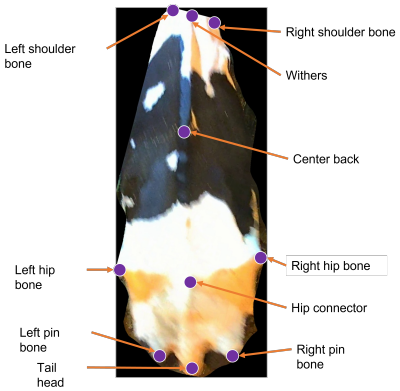

Figure 2: Example of cow top view keypoints

For implementing the keypoint and mask extractors, we use Detectron2 framework as it allows for seamlessly switching between models and adjusting hyperparameters through simple YAML files, provided the dataset is in the COCO format [10]. We trained two Keypoint R-CNN [7] models namely, keypoint_RCNN_R_50_FPN and keypoint_RCNN_R_101_FPN with batch sizes of 8 and 6 respectively, for 50,000 iterations on the training set of Cow Keypoints Dataset. Batch sizes for these models were limited by available GPU memory. Instance masks are detected by a *mask head* connected to the same backbone. Therefore, a single model predicts both the masks and the keypoints. These detectors are trained together as well. Unlike [5, 6] which have their own annotation format for extracting the rotated bounding box which is incompatible with COCO annotations, we extract rotated bounding boxes using the COCO compatible instance masks. We have empirically chosen ten keypoints on the cow's body. Figure 2 exemplifies these keypoints, namely: the shoulders, withers and center-back towards the front of the cow, and the hip bones, hip connector, pin bones, and tail head towards the rear of the cow.

Handling cow misalignment

The masked cows in cropped and rotated images have inconsistent orientation. This results in different bit vectors being generated for different instances of the same cow, leading to wrong cow ID predictions. To handle this problem, we use keypoints to align each cow in the rotated bounding box to a standard template. The template is derived heuristically and is defined by a fixed image size (of 256×512 pixels) and fixed keypoint locations. All detected cow instances, irrespective of their size, orientation, and spine curvature will be warped such that they conform to the template. To do so, the ten visible keypoints are obtained and added to the warping set. Then, the six additional points corresponding to the four corners and the left and right edge centers of the instance bounding box are added to the same set to improve warping quality. The target locations for all these points are defined by the template. The 16 points are used to partition the bounding box into triangular regions, which are then affine-warped into the corresponding triangular regions of the template. An example of keypoint based alignment to template is shown in Figure 3. To further reduce background from creeping in and affecting prediction, we exclude the four corner triangles in the final aligned image. This is illustrated by the sharp edges of the cow body towards the four corners in the rightmost image in the same figure. Note that all cows in the cattlog are also aligned to the same template.

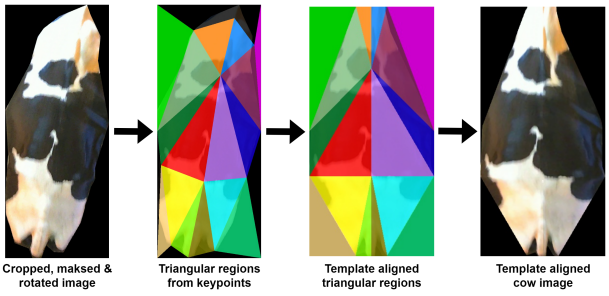

Figure 3: Example cow image alignment to keypoint template

Handling misplaced keypoints

It is often the case that the keypoint detector we use predicts one or more keypoints in the wrong locations. These wrong locations vary from being slightly off from the desired locations to lo-

cations that are very far from the cow instances. There could also be cases where a keypoint is detected in an adjacent cow and is meshed with the other keypoints of the cow of interest. These kind of errors would result in wrong triangular regions being affine-warped into the template, thereby hindering recognition.

To prevent such errors, we build a comprehensive list of rules to determine whether the predicted keypoints effectively represent a cow or not. We currently use 21 carefully hand crafted rules that check if values such as angular deviation between two sets of three keypoints, length deviation between two pairs of keypoints, minimum value of ratio of distance between keypoints to size of the instance are within the set limits or not. These limits are set by hand to reject most non-conforming cows in the training set of the Cow Images Dataset. Schemes to get these limits automatically from the training set annotations have not yet resulted in better performance than using these handcrafted limits. We allow a cow instance to proceed for cow ID prediction only if all 21 rules are passed, and we drop a cow instance if it breaks even one of these rules. Note that only if all ten of its keypoints are detected visible, a cow instance is subjected to rule checks. Otherwise, zero rules are passed. Thus, the keypoint rule checks help to reduce the number of bad cow instances reaching the cow ID prediction stage thereby reducing false matches.

Interpolated Keypoint	Required Keypoints	Interpolation policies
Hip connector	Both hip bones and any other keypoint	Point at a set distance from mid point of line segment connecting hip bones, closer to tail or away from head.
Pin/hip bone	The other pin/hip bone, tail head, hip connector	Reflect visible pin/hip bone across line through tail head and hip connector.
Tail head	Both pin bones	Mid point of line segment connecting the two pin bones.
Withers/center back	Both shoulders, tail head, hip connector, center back/withers	Point on weighted second order poly curve through visible spine points, at set distance ratio from center back/withers to hip connector and furthest from center back/hip connector.
Shoulder	All spine points, the other shoulder	Only if the spine points form a near straight line, reflect the other shoulder across line through tail head and withers. *Otherwise, do not interpolate.*

Table 1: Keypoint interpolation policies

Handling missing keypoints

The keypoint detector we use, Keypoint R-CNN, provides not only the location of a predicted keypoint but also a visibility score, which is a floating point value that could be interpreted as a confidence measure. Any keypoint with a visibility value above a set threshold (the Detectron2 default value of 0.5) is considered visible. Invisible keypoints are dropped; they are not used to form triangular regions in the source image for affine-warping. Ideally, invisible keypoints indicate that the corresponding parts of the cow are not visible in the image. However, we observed many instances where, due to poor predicted visibility scores, keypoints were not detected on parts of the cow that were actually visible. For these instances, even if all other keypoints were detected perfectly, the cows were not eligible for rule checks due to missing keypoints. This caused the system to lose many instances which could have led to perfect cow ID predictions. Reducing the visibility threshold below 0.5 was an option, but that could have led to many falsely detected keypoints. Instead, we exploit the redundancy in keypoint locations. Specifically, we create the keypoint interpolation policies as described in Table 1, to estimate the locations of missed keypoints using those of the detected keypoints.

We observe that the hind side of the cow – the region from the hip to the pin bones – stays mostly rigid, and, the frontal region – the region from the center back to the withers and the shoulders – deforms highly when the cow walks. So, the keypoints towards the rear of the cow can be interpolated with better accuracy than the ones towards the front. This is also reflected in the policy prerequisites for interpolating the shoulders in Table 1.

Thus, we now have a mechanism to enable more, valuable cow instances to be considered at the cow ID prediction stage.

Pixelation, binarization, and cow ID prediction

To convert each template-aligned cow into a 2D data matrix, we partition the template-aligned image (of size 256×512 pixels) into blocks of 16×16 pixels, where each block is intended to be stored as a single bit. We proceed to threshold the mean intensity value of each block with a threshold value of 127. All pixels above this threshold map to 255 (the maximum intensity) and those equal to or below it map to 0 (the minimum intensity). We found that this simple pixelation-binarization scheme performs better than a variety of thresholding options including adaptive and Otsu's thresholding.

The primary purpose of pixelation and binarization is data reduction. As the size of these blocks increases, though greater data reduction is achieved, the ability to distinguish individual cows decreases. This is because multiple cow-coat patterns would result in the same block pattern. On the template-aligned training set images, starting from block size of 128×128 pixels and going down to smaller block sizes, we counted the number of block patterns that mapped to more than one cow at each level. We chose block size of 16×16 as we found that it was the largest block size that had no two cows mapping to the same block pattern. A byproduct of pixelation is that it relaxes the accuracy requirement of the keypoint detector by a margin that scales with the block size used. Keypoints detected less than a block size away from their ideal positions might still result in the ideal block pattern. Binarization helps to negate the effects of lighting variation between the queried and cattlog images.

The obtained 2D data matrix is then serialized to form a bit vector, where the ones correspond to the white blocks with pixel values of 255 and the zeros correspond to black blocks with pixel values of 0. With this, we effectively reduce the cow's identity to a single bit vector of size $(256 \times 512)/(16 \times 16) = 512$, which could effectively be stored in 64 bytes. We follow this procedure to create bit vectors for each cow in the training set of Cow Images Dataset and then create a python dictionary (a hash map) of bit vectors to cow IDs. We call this the *bit vector cattlog*. In the current system, all the keypoints necessary for creating the bit vector cattlog come from manually annotated cow images. We need *exactly one* annotated instance of each cow in the training

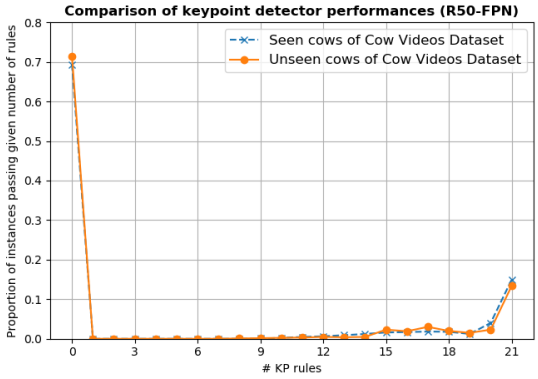

Figure 4: KeypointRCNN R50-FPN performance on seen and unseen cows

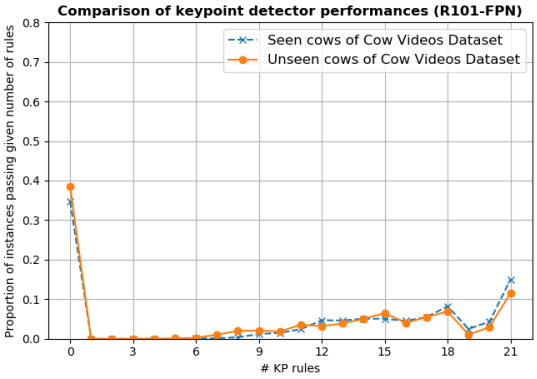

Figure 5: KeypointRCNN R101-FPN performance on seen and unseen cows

set for this purpose.

For predicting the cow ID, any queried image is converted to a bit vector as described, and this bit vector is used to query the cattlog dictionary for a matching cow ID. This basically operates as a matched filter. In practice, we cannot expect the keypoints to be detected in exactly the same locations in the queried image as in the cattlog images. This may result in different bit vectors being generated for different instances of the same cow. The margin advantage from pixelation proves to be ineffective at block sizes as small as 16. Despite the large binarization margin, significant lighting differences between the cattlog and queried images could also contribute to this error. So, in practice, the recognition system could fail to match a queried image even though the cow is in the cattlog.

To counter this issue, instead of hunting for cattlog bit vectors that match perfectly with the queried ones, we search for the cow that is the closest match, namely, the cow with cattlog bit vector that is closest to the queried bit vector in Hamming distance. The Hamming distance measures the number of bits that are different between the two given bit vectors, which here translates directly to the number of blocks in the pixelated binary image by which the queried cow differs from the cows in the cattlog. Euclidean distance can also be used here as it has a one-to-one mapping with Hamming distance.

One requirement of our cow recognition system is that it can add new cows to its catalog instantly, without the need for retraining and our system satisfies the requirement decisively. Adding a new cow to the recognition set is as simple as adding its *(bit vector, cow ID)* pair to the bit vector cattlog. This can be done in almost zero time when compared to training a deep learning model.

Results

We consider two different keypoint detection models while evaluating our system. These models differ in their backbone network architectures. One uses a Resnet50-Feature Pyramid Network (R50-FPN) backbone while the other uses Resnet101-Feature Pyramid Network (R101-FPN) [9] backbone.

The first question we consider before evaluating the recognition system as a whole is - should the keypoint detector have seen the cows while training in order for it to work? In other words, should it be retrained every time a new cow is added to the set? To explore this, we plot the proportion of cow instances in the Cow Videos Dataset that pass a given number of rules for cows both seen and unseen by the keypoint detector in Figure 4 and Figure 5. The peaks at zero in the two plots are because no rules are passed if all keypoints are not detected. This happens when all parts of the cow are not visible. Since we ignore such frames for recognition, peaks at zero have no significance. From the figures, we see that the plots of unseen cows nearly trace those of the seen cows indicating similar performance. With this we find that the keypoint detector need not be retrained on addition of new cows to the cattlog.

Next, to test the cow recognition system as a whole, testing on the entire testing set would not give an accurate picture of its performance. This is because the testing set has a few cows that are not in the training set, making it impossible for the system to recognize them. So, we use the Test-InSet subsets for evaluating the recognition system. Initially, we evaluated the recognition system on Cow Images Dataset, and observed the Top-1 accuracy to be merely 30% and Top-16 accuracy not much above 50%. These are shown by the blue plots in Figure 6 and Figure 7. This poor result is the output of the entire recognition system and we wanted to know whether to attribute this to the cow localizer or the cow ID predictor. For this reason, we evaluated just the prediction stage by using ground truth keypoint annotations of the Cow Images Dataset's Test-InSet, bypassing the keypoint detector. This provides an upper bound for the performance of the entire recognition system. These numbers, plotted in orange, are common to both figures as they are independent of the keypoint detector architecture. Observe how the Top-1 accuracy is very high (above 90%) and the Top-16 accuracy is at 100% (If we run the same experiment on the training set, we get a Top-1 accuracy of 100%. Recall that this is exactly why we settled on block size of 16.). With this we infer that the poor performance of the system is due to non-ideal performance of the keypoint detector.

With the hypothesis that providing the keypoint detector with more images of cows in different orientations could improve the result, we ran the recognition system on frames of videos of cows walking by in Cow Videos Dataset's Test-InSet. For each video, predictions for each of its frames were collected and a list with all predicted cow IDs was created. This list was sorted in decreasing order of number of predictions per cow ID and this sorted list represented the Top-K predictions at the video level. Essentially,

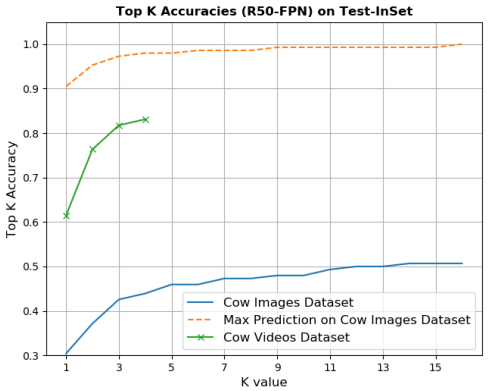

Figure 6: Top K Accuracy - R50-FPN

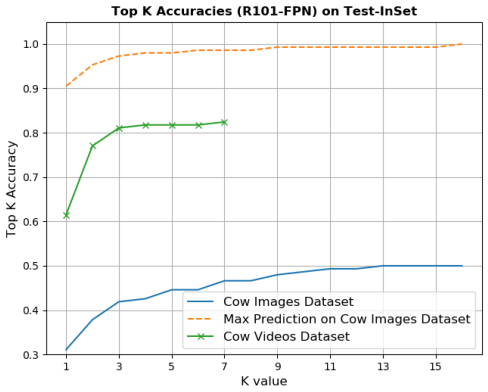

Figure 7: Top K Accuracy - R101-FPN

the Top-1 prediction, which is the final prediction, was the cow ID that was predicted the most. The K^{th} most frequently predicted cow ID for a given cow video is used as the K^{th} prediction to compute Top-K accuracy. The Top-K accuracies at the video level are shown by the green plots with crosses, in the same two figures as above. These results are also listed in Table 2.

From the plots, we see that the Top-1 accuracy has doubled and Top-K accuracy is also higher, thereby validating our supposition. Note that at the video level, we are sorting the Top-1 image level predictions and hence, video level Top-K accuracy can only be measured up to the number of different Top-1 predictions at the image level. There are only four points on the plot of R50-FPN indicating that the prediction system does not get confused beyond four different cow IDs for any given cow. This number is 7 for the R101-FPN model. This proves that the model with R50-FPN backbone, despite being smaller in size, performs better than the one with R101-FPN backbone.

From Table 2, we see that the Top-K accuracy values we obtained are similar to those from [5]. While these results are not directly comparable because they are not evaluated on the same dataset, we emphasize the fact that we achieve this performance on a dataset with roughly the same number of cows, using just one image per cow for training and near zero retraining time to add new cows to the cattlog.

Top-K	1	2	4	7
Accuracy (R50-FPN)	0.6149	0.7635	0.8311	-
Accuracy (R101-FPN)	0.6149	0.7703	0.8176	0.8243

Table 2: Top-K Accuracy on Cow Videos Dataset

Conclusion and Future scope

We design and implement a computer vision system to localize and identify cows by their individual cow IDs. This system performs similar to systems based on deep learning, without the need for many training images. We achieve this by decomposing the recognition system into a cattle ID agnostic keypoint detection system that comes pretrained and a stand-alone cow ID prediction system that requires just one image per cow to be trained (hence Eidetic). Future work could consider supplementing the bit vectors with features such as length and width of the cow for improving recognition accuracy.

References

[1] William Andrew, Colin Greatwood, and Tilo Burghardt. Visual Localisation and Individual Identification of Holstein Friesian Cattle via Deep Learning. *IEEE International Conference on Computer Vision Workshops*, 2018-Janua:2850–2859, 2017.

[2] Ali Ismail Awad. From classical methods to animal biometrics: A review on cattle identification and tracking. *Computers and Electronics in Agriculture*, 123:423–435, 2016.

[3] Sergio Canu. Face swapping (explained in 8 steps) – Opencv with Python. *https://pysource.com/2019/05/28/face-swapping-explained-in-8-steps-opencv-with-python/ Date Accessed: 2022-02-20*.

[4] Xudong Cao, Yichen Wei, Fang Wen, Jian Sun, X Cao, Y Wei, F Wen, and J Sun. Face Alignment by Explicit Shape Regression. *Int J Comput Vis*, 107:177–190, 2014.

[5] Jing Gao, Tilo Burghardt, William Andrew, Andrew W. Dowsey, and Neill W. Campbell. Towards Self-Supervision for Video Identification of Individual Holstein-Friesian Cattle: The Cows2021 Dataset. *ArXiv ID: 2105.01938*, 5 2021.

[6] Jing Gao, Tilo Burghardt, and Neill W. Campbell. Label a Herd in Minutes: Individual Holstein-Friesian Cattle Identification. In *International Conference on Image Analysis and Processing Workshops*, pages 384–396, 2022.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):386–397, 2020.

[8] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 1867–1874. IEEE Computer Society, 9 2014.

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C Lawrence Zitnick, and Piotr Dolí. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[11] He Liu, Amy R. Reibman, and Jacquelyn P. Boerman. Video analytic system for detecting cow structure. *Computers and Electronics in Agriculture*, 178:105761, 11 2020.