Feed and Bunk Analysis using Computer Vision

John Vu

UNL- Department of Mechanical and Materials Engineering

Jean Niwenshuti, Biological Systems Engineering

UNL- Department of Biological Systems Engineering

David Fernandez Exposito,

UNL- School of Computing

University of Nebraska-Lincoln, Lincoln, NE

1 INTRODUCTION

Estimating the amount of feed in the bunk is a critical task in feedlot management. It allows the feedlot to provide the animal the right rations at the right time in the right amount [1]. Currently, feed bunk management relies on bunk scoring techniques that are not accurate. These bunk scoring techniques may cause the excess feed to spoil or bunks to be empty for hours. This affects the animal's daily gain and ultimately the cost of production increases

Alternative methods are needed to ensure accurate feed estimation and minimize errors caused by current feed scoring techniques. Computer vision provides a lot of techniques that can be used to monitor animal feed consumption in real-time. Feedlot managers will be able to monitor their feed bunks using cameras attached to mobile systems such as drones or robots, and estimate the amount of feed remaining in the bunk.

Computer vision is a growing area of research and discovery. Several techniques, such as texture segmentation and edge detection, made computer vision a powerful tool to use in object detection. The goal of this study was to explore ways to detect the feed and identify the type of feed and its position in the bunk. This is the first step in developing a robust computer vision method to quantify the amount of residual feed in the bunk.

Therefore, in order to contribute to this final goal, this study used a combination of machine learning and texture segmentation techniques to identify the feed in the image.

2 OBJECTIVES

The objectives are:

- Detect whether there is feed in the bunk and evaluate the model accuracy.
- Identify the region with feed and evaluate its accuracy.
- Identify the type of feed in the bunk and evaluate the accuracy of the model.

3 RELATED WORK

The livestock industry has been exploring machine vision systems to predict the amount of feed in the bunk. In dairy farms, studies have been done in using machine learning techniques to predict the daily feed intake.

Saar et al used depth images of feed to train a convolutional neural network to detect changes in the feed [2]. The study was able to predict change in feed with an average mean absolute error of 0.14kg and a random mean square error of 0.19kg.

In 2016, ResNet50 was proposed as a new deep learning model that was more accurate and was able to achieve a prediction error of 3.57% [3]. It is made up of more than 158 layers and performs well at object detection and classification

4 PROBLEM DEFINITION

The current problem is that there is no automation system to estimate the amount of feed in the bunk. More specifically, we are addressing the automation of interpreting feed bunk information, such as the amount of feed, where the feed is, and the type of feed.

The input for the system would be feed bunk images containing a bunk and feed, and the output would be the segmentation and identification of the feed.

Our results could later be used to quantify the residual feed in the bunk and, in combination with a depth map, to determine volume of the residual feed. .

5 IMAGES

Image data were collected at the Animal Science Complex (UNL) lab. The images were acquired using an azure kinect sdk depth camera that acquires both the RGB images and the depth images. A custom C++ algorithm was developed to acquire images of 3072 x 4096 pixels in visual studio.

Figure 1: RGB image of a concrete feed bunk with feed used in the study.

6 APPROACH

Developing Image Data Sets

Our study consisted of 2 main datasets.

- Detection of feed in the bunk: The dataset was made of 2 classes. The first class had 42 images of empty feed bunk with no feed and the second class had 300 images of feed bunk with different types of feed (Hay and grains).
- Identification of the type of feed: This dataset was made of 3 classes. Each class contained 50 images of the common diets fed in Nebraska feedlots. The first class had 3 diets (*CB Stalks : 15 MDGS Finisher, CB Stalks : 0 MDGS Finisher, and CB CS: 0 MDGS Finisher*), second class had 2 diets (*Merck High Systems and Merck Low*), and the third class had 2 diets (*CB CS: 15 MDGS Finisher, and 500/600 Finisher*). These classes were made according to the diet's texture similarities.

Objective 1 used the first dataset and the last objective used the second dataset.

Detecting if There is Feed Using Deep Learning

A ResNet50 model was trained to classify between 2 classes. Before training, the input images were resized to 224x224x3. The dataset was augmented by 30% by rescaling and rotation. After, the dataset was split into 70% training and 30% to validate.

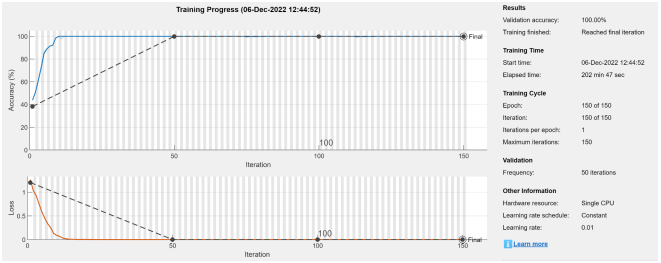

Figure 2: ResNet50 training and validation process graph with 30 iterations, validation accuracy of 91.3% and a validation loss of 0.12.

Based on the certainty, we can determine if there is feed in the bunk or not. It was chosen that if the model predicted that there is feed with 95% certainty, then there is feed. Else there is none. Then running the test will give us our Confusion Matrix, F1 Score, Accuracy, Recall, and Precision.

Locate the Feed Using Texture Segmentation

After completing the two previous objectives, we wanted to locate the feed in the image as our final objective. We decided to do it without deep learning, directly applying texture segmentation. Texture segmentation is the process of partitioning the image into regions with different textures. We decided to use this method because in our specific case, the dataset we are using verifies that the feed has a very distinguishable texture, so it can be easily detected.

Proposed method:

Figure 3. Texture segmentation algorithm.

The schema of the algorithm can be found in figure 1. The first step of our algorithm is to convert the image to grayscale. This is because the functions we will use work with two dimensional arrays, and the texture (which is what we care about) is not affected by this.

The next step is to get the texture image. For this, we calculate the entropy of the image (one value for each pixel), which is a statistical measurement of randomness. We do this using the matlab function *entropyfilt(I)*. Similar results can be obtained using different matlab functions, such as *stdfilt(I)* or *rangefilt(I)*, but we decided to go with *entropyfilt(I)* because it is the most widely used in all examples of texture segmentation we have found online.

After this, we rescale the result obtained for the entropy using the function *rescale(I)*, so we have all the values between 0 and 1.

After this, we create the mask. First of all, we threshold the rescaled image to segment the textures. We used the function *imbinarize(I, 0.85)*. We chose this value after trying various values and deciding this is the value that suits our case better.

After completing this, we eliminate the unwanted points. For that, we just do a simple convolution with a 105x105 matrix with all ones. This convolution just counts the number of points around each pixel. This way, we just keep those points with a value greater than 4500 in the result of the convolution. We chose this value after trying with different values and determining that this was the one that worked the best.

We also used the functions *imclose*, to smooth the edges and close possible open holes, and *imfill*, to fill holes. Once we have done this the mask is finished, although we add 1 to the values obtained so instead of 0 and 1, we have 1 and 2 and we can superimpose the mask and the image, as shown in Image 2.

For displaying the result, we use the function *labeloverlay(img, L)*.

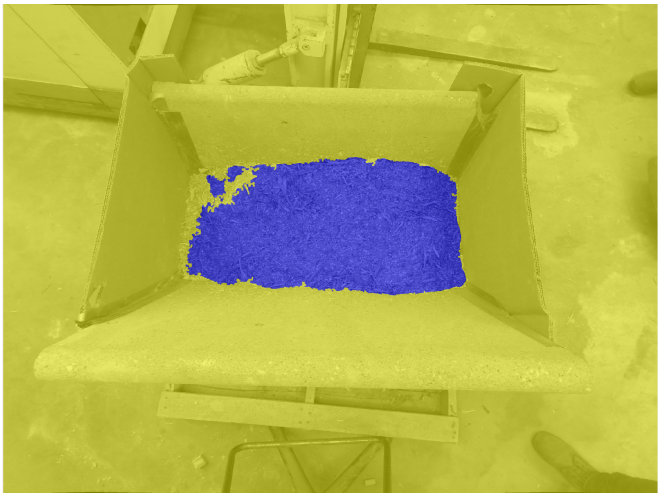

Figure 4. Result of our texture segmentation method

APP:

We built a MatLab app that shows the result of our texture segmentation method. With this app, the user can select an image and the result of the texture segmentation will be shown along with the original image. This app basically just calls the texture segmentation and shows the result alongside the image selected.

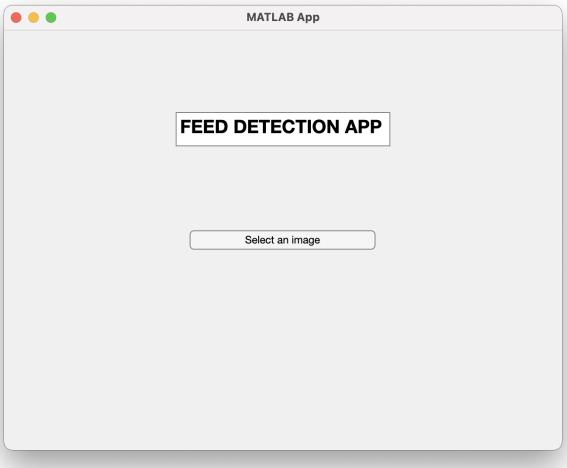

Figure 5. App

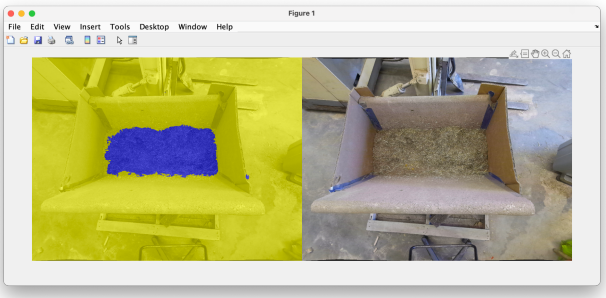

Figure 6. Result shown by the app.

Detect the Type of Feed Using Deep Learning

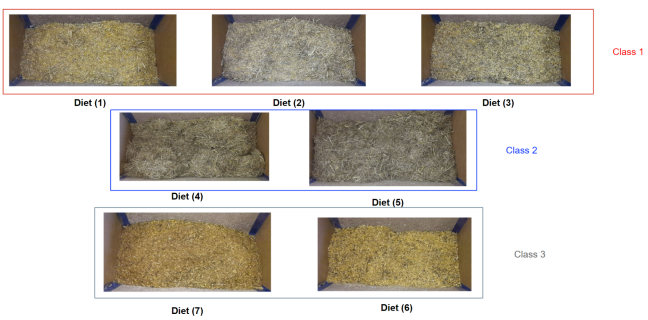

Figure 7. Different classes used to identify the type of feed

For the last objective, the same training process was followed as in the first objective. Our initial approach was to train our model to work with the 7 diets. However, we got extremely low validation accuracy (66.67%). We think this happened because there are diets that are almost identical in terms of color and texture, making it almost impossible for the model to distinguish them. For this reason, we decided to group the diets in 3 classes, as

shown in Figure 7. To increase the accuracy more options were added in to augment the dataset and more iteration were added in the training process. Blurring and pixel translation were added in the augmentation process.

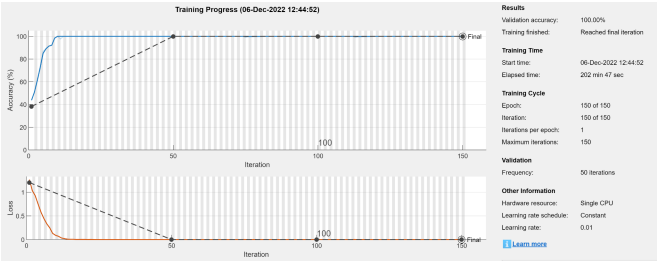

Figure 8: ResNet50 training and validation process graph with 150 iterations, validation accuracy of 100% and a validation loss of 5e-5.

7 EVALUATION

Deep Learning for Detecting if There is Feed

We used the confusion matrix and the accuracy, F1 score, precision, and recall, to assess our Deep Learning model. Here is the resulting confusion matrix and the equations used:

**Note: We are testing for feed and if the prediction score > 95%, then it is feed.*

Confusion Matrix (#sample = 72) (#Test image =72)		**Real Label**	
		+	-
Predicted Label	+	True Positive(TP) **52**	False Positive(FP) **0**
	-	False Negative(FN) **11**	True Negative(TN) **9**

Figure 9: Confusion Matrix for Feed or No Feed

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

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

$$F1\,score = 2 * (precision^{-1} + recall^{-1})^{-1}$$

$$Accuracy = \frac{\sum TP + TN}{\#Samples}$$

Metric	**Score**
Accuracy	0.847
F1 Score	0.904
Precision	1.000
Recall	0.825

Table 1: Metrics used for Deep learning Feed or no Feed

Observation

The deep learning model scored well with an accuracy and F1 score of .847 and .904, respectively. Since the recall score is .825 and precision score is 1, it is concluded that the model performs slightly worse in determining if the bunk is actually empty. This is due to the stone pattern at the bottom of the bunk, which causes the model to confuse some feed images as the bottom of the bunk (False Negatives). This can be mitigated by changing the bunk material to have more contrast against the feed. The empty bunk is shown in Figure 10.

Figure 10: Image of Empty Trough.

Texture Segmentation Evaluation:

First of all, we got qualitative results. For these, we ran the algorithm with 30 images and checked that the result obtained was good. A summary of qualitative results is shown in Figure 11.

We also got quantitative results. For these, we decided to use the intersection over union (IoU) metric as a measure. We chose this measure because we consider it's the one that suits our case and is widely used in similar situations in a large amount of papers. This metric basically takes the union of the result and the groundtruth and checks the proportion in which both intersect. The higher, the better. Quantitative results are shown in Table 2.

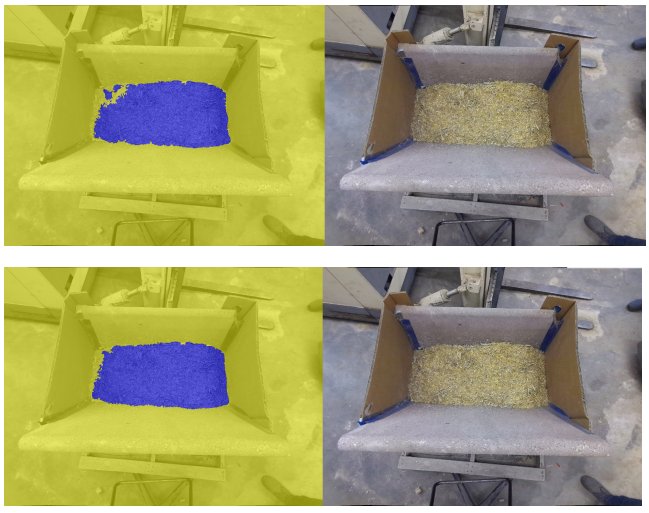

Figure 11: Qualitative result.

Here are the quantitative results:

# Samples	34
Mean IoU	.8635

Table 2: Quantitative results for texture filtering

Observations

Most of the poorly performed IoU (<.90) were images of low bunk. This makes sense due to less texture features from corn, grains, hay, etc., as the bunk becomes more empty. This can be seen in the following figure where the blue is the groundtruth and the red is the result from the algorithm:

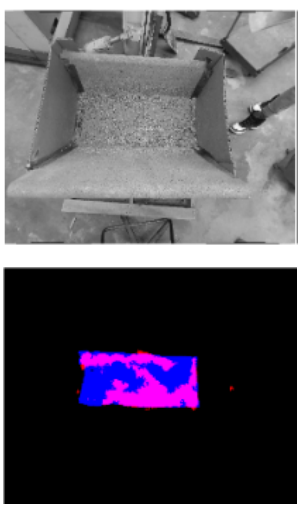

Figure 12: IoU.

Deep Learning for Detecting Types of Feed

We used the confusion matrix and the accuracy, F1 score, precision, and recall, to assess our Deep Learning model. Here is the resulting confusion matrix and the equations used:

**Note: We are testing for the current class and if the prediction score is highest compared to the other classes, then it is that class.*

Confusion Matrix (#sample = 261) (#Test image =87)		**Real Label**	
		+	-
Predicted Label	+	True Positive(TP) **87**	False Positive(FP) **0**
	-	False Negative(FN) **0**	True Negative(TN) **174**

Figure 13: Confusion matrix for type of feed.

Metric	**Score**
Accuracy	1
F1 Score	1
Precision	1
Recall	1

Table 3: Metric for Deep learning type of feed

Observation

The deep learning model performed perfectly, concluding that the identification of the 3 classes are trivial and the error is within the classes. Further testing in different environments is needed to validate our results.

8 IMPLEMENTATION TIMELINE

Overall we had 6 in person meetings. The three of us attended all of them. Here is a summary of the implementation timeline of this project:

- The first meeting was on October 30, and we just scratched a timeline for the project and assigned some tasks to get started
- In the second meeting, on November 12, we discussed all the problems we were encountering with our initial objectives. We concluded that the dimensions objectives (one of the initial objectives) were not possible because we don't have enough information about the camera parameters or stereo vision. In the same way, segmenting the through was not viable because we could not find a way of doing it independently of the image used.

Therefore, we shifted to our final objectives: deciding if there is feed or not using deep learning, locating the feed using texture segmentation and detecting the type of feed using deep learning.

- The following meeting was on November 19. In this meeting we discussed the progress made with the deep learning models and the texture segmentation. We agreed that all the objectives were possible, although we needed to work hard to achieve the feed location and the model for deciding the type of feed.
- The next meeting was on November 26. In this meeting David showed the final code for the texture segmentation that was finished. We decided that John would get the quantitative results of this, while David would work on a basic app for a better user experience.
- The next meeting was on December 1. In this meeting we shared all the final codes for the deep learning models and we got the results for the testing. We started preparing the final report and the slides for the presentation.
- The last meeting was on December 5, where we finished the slides and made progress with the final report. We decided that the report would be finished conjunctly but we were not going to meet in person again.

9 FUTURE WORK

- Improve Deep learning model by
    - Increasing layers and interactions
    - Using a wider variety of training and testing data sets.
    - Increase Data set
    - Include more practical data set such as in a farm setting with cows.
- Further investigation of classification of type of feed
    - Make a robust deep learning model for the 3 classes to detect similar looking feed.
- Estimate volume of feed for practical use in feed lot automation.

REFERENCES

[1] K. K. Bolsen, B. E. Brent, M. K. Siefers, and M. E. Uriarte, "Silage management: Important practices often overlooked," *Kansas Agricultural Experiment Station Research Reports*, no. 2, pp. 9–12, 2000.

[2] M. Saar, Y. Edan, A. Godo, J. Lepar, Y. Parmet, and I. Halachmi, "A machine vision system to predict individual cow feed intake of different feeds in a cowshed," *Animal*, vol. 16, no. 1, p. 100432, 2022.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.