Spring 2022 Applied Analytics Practicum

Final Report

Object Detection with Semantic Segmentation

Video Team 1 – “Chick-Tok”

April 20, 2022

Daniel Muehring (dmuehring3), Jacob Rachoene (jrachoene3), Thet Hein Tun (ttun3)

Contents

Introduction

Motivation

Poultry farming is one of the primary global sources of protein and plays an integral role in eradicating world hunger [1]. Sustainable poultry farming is important not only for the world's food supply chain, but also for animal welfare so that chickens are treated humanely.

Our literature review suggests that only a small portion of mainstream approaches to poultry welfare management leverages the recent development in big data and computer vision technology. Taking advantage of large audio and video feed datasets for monitoring purposes is still limited even among the most innovative researchers and practitioners [2].

Problem Statement

The purpose of our project is to use computer vision, deep-learning approaches, and traditional statistical tools to analyze video clips to better predict chicken behavior. Specifically, can we reliably predict not only the number of chickens around a feeder at any given point, but also the number of chickens which are actively using the feeder?

AudioT Background

AudioT specializes in using acoustic monitoring of broiler houses to automate and better predict chicken welfare outcomes [3]. By applying machine learning techniques to continuously monitored audio feeds, AudioT detects anomalies in health and behavior within a flock in real time, complementing experts' observations. Through these techniques, AudioT has been able to predict outbreaks of respiratory diseases within flocks; this is a valuable service that can provide peace of mind to clients while also offering them more efficient and effective poultry monitoring and management services.

AudioT has recently ventured into using videos to expand their analytics toolkit to detect and address wider aspects of chicken behavior and welfare. By incorporating video feeds into their workflow, AudioT could potentially identify positive and negative behaviors for and between chickens such as sparring, playing, excessive movement, gait impairment, fighting, feeding, and drinking—all of which can be used to predict future health outcomes.

Literature Review

Computer Vision and Video Analytics

The field of Computer Vision experienced a seismic shift in 2012 when deep convolutional neural networks were deployed for ImageNet classifications of 1.2 million images and surpassed previous state-of-the-art approaches [4]. Since then, deep learning has become an integral and increasingly dominant part of Computer Vision.

One of the commonly used optimization algorithms for deep learning is stochastic gradient descent (SGD) [5]. To allow for sustained and deep model training, various techniques (such as Xavier initialization and ReLU activations [6]) were developed to avoid vanishing and/or exploding gradients during the back propagation phase [7]. Later, instead of using fully connected neural layers to brute-force the entire architecture (with billions of parameters to train even a single image), more structured and efficient convolution layers were introduced [4].

Learned filters of convolutional neutral networks (CNN) can extract low- and high- level edges and features, as seen in Figure 1 [8]. This means moving away from traditional (pre-2012) hand-designed features in Computer Vision and towards self-learning and extracting simple and complex features. As a result, recent deep-learning-based image processing and classification tasks all incorporate CNN as part of their overall architecture. The general direction of the field is leaning toward producing useful architectures rather than finetuning smaller components [9].

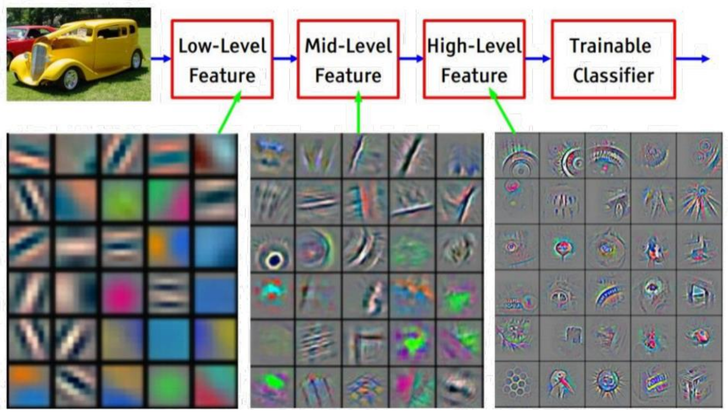


Figure 1. Learning low-, medium- and high- level features with CNN. Source: [8]

Video analytics, which can be considered a sub-field of Computer Vision, is also evolving along with deep learning. Object detection in videoframes can be achieved with great accuracy using bounding box techniques. Semantic segmentation can also be used to identify objects from the background in each videoframe (see Figure 2 that illustrates the difference). Some popular architectures include You Only Look Once (YOLO) [10] and faster R-CNN [11] for bounding-box-based object detection, and U-Net [12], DeepLab [13] and ResNet [14] for semantic segmentation.

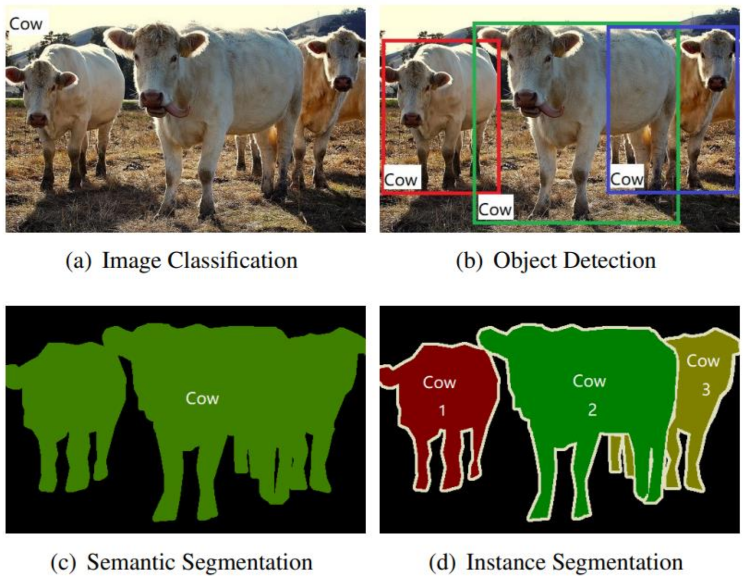


Figure 2. Comparison of different visual recognition tasks in computer vision. Source: [15]

Object tracking is different from real-time detection or segmentation in that the former "remembers" a particular object across multiple videoframes, whereas the latter detects objects repeatedly for each frame without necessarily having the "memory" of identified objects from previous frames. Object tracking is helpful when we want to avoid double counting the people or objects within a specified area over time. Example networks include Generic Object Tracking Using Regression Networks (GOTURN) [16], and DeepSORT, an extension of Simple Online Real-time Tracker (SORT) [17].

Computer Vision in Poultry

The monitoring and evaluation of how birds utilize feed resources is critical in understanding the health and well-being of the birds in broiler houses. If ignored, this can have major economic implications for poultry farmers. Traditionally, such evaluation was only possible through manual observation. One can imagine the potential benefits to an automated approach, especially at scale.

Previous methods to study poultry behavior on a large scale included weighing systems, tagging birds with radio frequency identification (RFID) [18, 19] and less invasive image-based technologies utilizing image and video feeds from cameras in poultry houses to estimate bird weights, evaluate lameness and monitor flock activities [20]. Further computer vision and image analysis approaches include real-time bird distribution indices, determining hens' preferences for certain temperature and lighting conditions, and detecting abnormal feeding behaviors [21]. Recently, deep neutral networks have slowly emerged as the tool of analysis among poultry researchers as seen in Table 1.

Table 1. Select literature on chicken monitoring using deep-learning-based computer vision

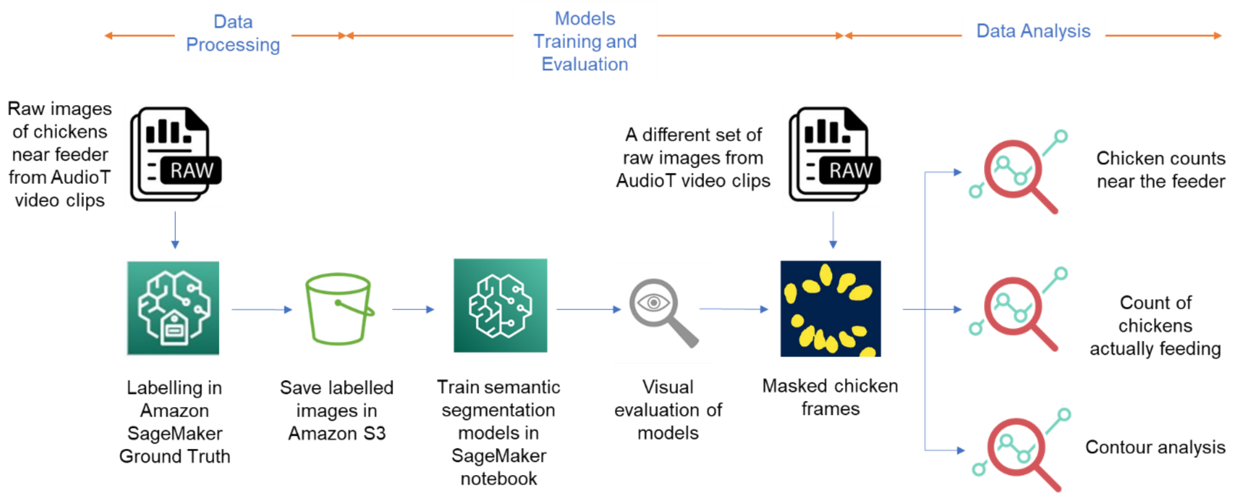
Topic	n	Model	Software	Reference
Behavior classification	350	YOLO v3	Darknet framework	[21]
Tracking of individual birds	10	Deep regression	Python	[22]
Monitoring broiler chicken floor distribution	126	Back propagation neural network	MATLAB	[23]
Heat stress monitoring	-	Faster R-CNN	Caffe	[24]
Detection of sick broilers	400,000	Improved Feature Fusion Single Shot MultiBox Detector (IFSSD)	OpenCV	[25]
Automatic detection of sick chickens	-	Residual neural network	-	[26]
Classification of broiler droppings for intestinal disease detection	10,000	Faster R-CNN, YOLO v3	Tensorflow framework; Darknet framework	[27]
Broiler chickens' weight prediction using 3D computer vision	48,000	Bayesian Artificial Neural Network	IDRISI 32	[28]

Source: Adapted from [2].

Methodology

Workflow

The overall flow for data processing, modeling and analysis is summarized in Figure 3. For the first two phases, we used cloud-based Amazon SageMaker with data stored in S3 buckets to ensure that labelling and model training were done in a single self-contained pipeline. For the data analysis phase, the team set up respective local environments.



Data Processing
Models Training and Evaluation
Data Analysis
Raw images of chickens near feeder from AudioT video clips
RAW
A different set of raw images from AudioT video clips
RAW
Labelling in Amazon SageMaker Ground Truth
Save labelled images in Amazon S3
Train semantic segmentation models in SageMaker notebook
Visual evaluation of models
Masked chicken frames
Chicken counts near the feeder
Count of chickens actually feeding
Contour analysis

Figure 3. Workflow of the Project

Data Processing: Labeling Process

Labeling was done through Amazon SageMaker's Ground Truth, a platform to flexibly label and manage datasets to be used in machine learning models [29]. It offers the ability to work with image-based datasets and offers multiple kinds of labeling including image classification, bounding box, and semantic segmentation.

We worked with semantic segmentation, which labels each pixel of an image into a specific category. For our purpose of identifying chickens, we labelled each raw image into two categories: "background" and "chicken" (Figure 4). This categorized image is also known as a "mask". These hand labelled masks were used as the training set for our semantic segmentation models. Overall, 57 images were labelled.

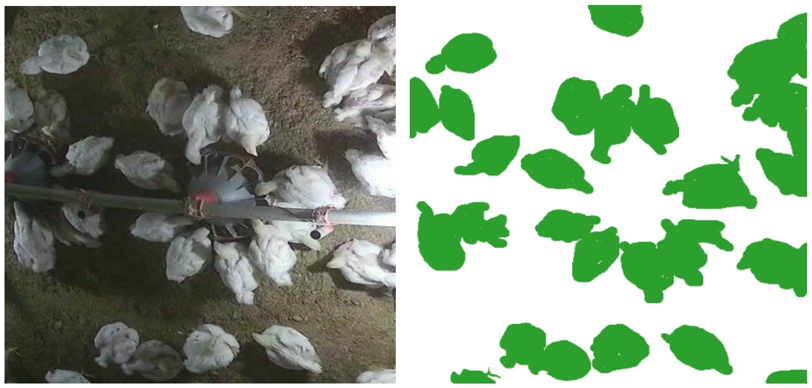


Figure 4. *left*: raw sample image; *right*: image with “mask” after labelling in SageMaker

The labelled images consisted of birds over the course of six weeks, or in other words, for most of their time in the broiler houses. These images were cropped to three locations along the feeder line which runs horizontally through the broiler house. Feeder locations were selected because of our interest in feeder behavior, and the broad age range was selected in hopes of being able to create a more generalizable model that would be applicable to birds of all ages.

Labelling a training set for the model was a tedious process; semantic segmentation labelling necessitates accounting for each pixel by hand. Ground Truth does provide some tools to help manual labelling like being able to autofill closed polygons set by the user. While SageMaker has the option to crowdsource labelling tasks to third party entities (known as “Mechanical Turk”), this was not a feasible option due to the sensitive nature of the video files from a privacy perspective. SageMaker also has the option to automatically label images for the user once a small subsection has been labelled but requires a minimum of 1,250 objects with a recommended minimum of 5,000 objects [29].

It should be noted that semantic segmentation does not distinguish between instances of a particular class; for example, labeling one group of pixels under the chicken category does not create a separate chicken object which can be distinguished from the other chicken labeled pixels.

Models Training

To train semantic segmentation models using SageMaker, we performed the following steps (Figure 5):

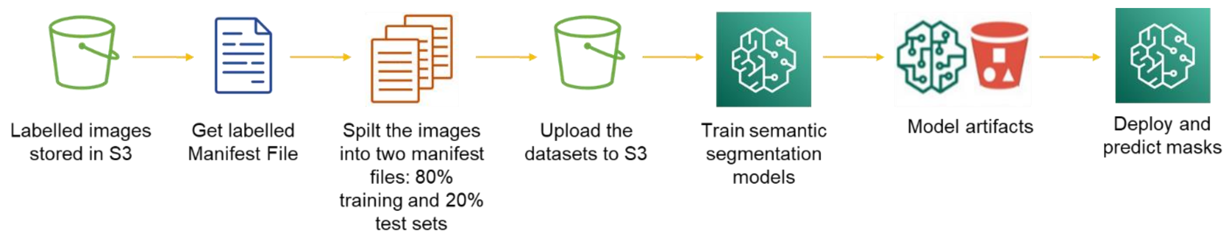

Labelled images stored in S3
Get labelled Manifest File
Spilt the images into two manifest files: 80% training and 20% test sets
Upload the datasets to S3
Train semantic segmentation models
Model artifacts
Deploy and predict masks

Figure 5. Workflow for the Semantic Segmentation Model

- We first accessed the labelled images stored in S3. Instead of retrieving the images directly, the conventional way in SageMaker is to use a *manifest file*, an entry containing dictionary references to the labelled files.
    - Location: `s3://v1-sp22-team-bucket/utk-feeder-cropped-images-all-times/utk-sem-crop-dm-v1sp22/manifests/intermediate/1/output.manifest to data/output_dm.manifest`
- We split the dataset into 80% training samples (46 images) and 20% validation samples (11 images). Although the number of images is limited, SageMaker build-in models are, by default, pre-trained on much larger datasets (such as ImageNet). As such, many low-level features are already included in SageMaker models through transfer learning, and training duration is much reduced. The main task of our model training is then to specifically recognize chickens.
- The split datasets were then saved as manifest files and uploaded to S3.
    - Locations: `s3://v1-sp22-team-bucket/utk-feeder-cropped-images-all-times/training/train.manifest; s3://v1-sp22-team-bucket/utk-feeder-cropped-images-all-times/training/validation.manifest`
- We used these spilt manifests to train semantic segmentation models (see below for parameters used). In addition, we also specified input and output data configurations.
    - Input mode: `Pipe`
    - Record wrapper type: `RecordIO`
    - Data source: `S3`
    - S3 data type: `AugmentedManifestFile`
    - S3 data distribution type: `FullyReplicated`
- The names of the models trained must be unique, and often we attached present timestamp to our initials to automate the script. Some of the successfully trained models include:
    - `seg-4-copy-03-172`
    - `seg-5-4`
    - `semantic-segmentation-2022-03-21-20-03-11-901`
    - `ttun3-2022-03-22-18-20-1647973248`
    - `ttun3-2022-03-22-19-06-1647976015`
    - `jr-seg-2022-03-23-03`
- Outputs from the trained models are then stored as *model artifacts*, which can later be deployed via endpoint configurations for prediction and classification tasks.

Parameter tuning can be done either through codes or through SageMaker console interface. Although we have conducted many experiments during model training, those in Table 2 represent our most structured and rational models, and thus were also used for model validation and downstream data analysis (See Findings and Discussions).

Table 2. Trained models with respective parameters

	Model 1	**Model 2**	**Model 3**
Trained Model Name	`seg-5-4`	`ttun3-2022-03-22-18-20-1647973248`	`ttun3-2022-03-22-19-06-1647976015`
Training time (seconds)	218	313	318
Instance type	ml.p3.2xlarge	ml.p3.16xlarge	ml.p3.16xlarge
Additional volume size (GB)	50	50	50
Maximum runtime (s)	360,000	360,000	360,000
Output Model Artifact	`semantic-segmentation-2022-03-31-16-36-55-677`	`semantic-segmentation-2022-04-01-14-46-49-383`	`semantic-segmentation-2022-03-31-16-19-39-650`
Hyperparameters			
Algorithm	fcn	fcn	deeplab
Backbone	resnet-50	resnet-101	resnet-101
Crop size	240	240	240
Early stopping	True	True	True
Epochs	10	10	10
Learning rate	0.001	0.001	0.001
Learner scheduler	Poly	Poly	Poly
Learner scheduler factor	0.1	0.1	0.1
Minimum batch size	10	10	10
Momentum	0.9	0.9	0.9
Number of classes	2	2	2
Number of training samples	46	46	46
Optimizer	RMSProp	Adam	Adam
Use pretrained model	True	True	True
Validation minimum batch size	3	10	10
Weight Decay	0.005	0.0005	0.0005

Since training on CPU can take hours, SageMaker only allows GPU instances (such as p2 or p3). There are three built-in algorithms available with SageMaker—fully convolutional network (FCN), DeepLabV3, and Pyramid Scene Parsing (PSP)—and we deployed the first two. The two backbone choices in SageMaker include residual nets of resnet-50 and resnet-101, which help prevent vanishing gradient during training. While the default SDG-based optimizer is Root Mean Squared Propagation (RMSProp), we also tried Adam since this is a more widely accepted technique in the industry.

Methods for Chicken Feeding Analysis

To examine chickens around feeders using the trained semantic models, we first chose a five-minute video of 3-week-old birds to analyze. Note that this video clip is different from the images that were used to train the models. We divided the video into individual frames and cropped those frames around two central feeders (designated as Feeders A and B with 292 frames each). Each image measured 310 by 325 pixels. These areas were chosen because they had the clearest view from the camera in terms of both the angle at which the feeders were seen, and the lighting surrounding these areas. Other feeders were either viewed at an oblique angle or had extreme lighting issues that could potentially interfere with analysis. We also chose to look at 3-week birds because of their smaller size, which we hoped would lead to fewer problems in our analysis. Three approaches were then explored.

(1) Counting Chickens in Feeder Areas

In the first approach, we aimed to produce an easily comprehended method to determine the number of chickens in the general area around a feeder. We calculated the proportion of chicken pixels detected within a mask generated from the semantic segmentation model. We regressed this against a manual "ground truth" count of how many birds were in each raw image and derived a linear regression equation for the number of birds around a feeder at any given time during the video.

To implement this, we fed the cropped images around the two feeders to our semantic segmentation model to produce corresponding masks (Figure 6). We then calculated the proportion of chicken pixels in these masks compared to the overall area of the image using Python's OpenCV library. For a random sample of 86 frames, we determined the true count of chickens within the frame. Through a simple linear regression, we can see that the proportion of chicken pixels has a linear relationship with our "ground truth" count as seen in the scatterplot (Figure 7).

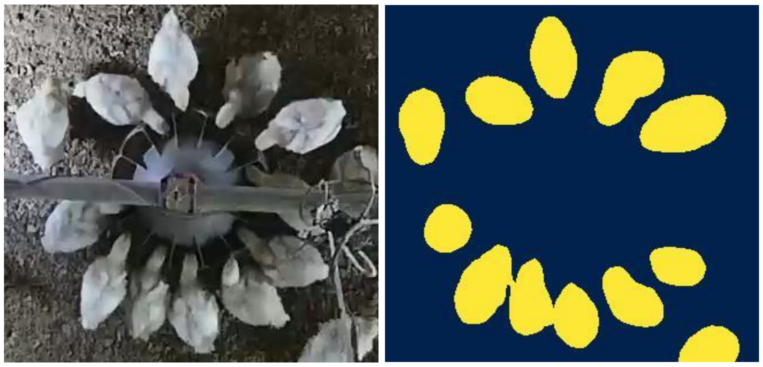


Figure 6. *left*: Raw image of Feeder A; *right*: Mask using our best trained segmentation model

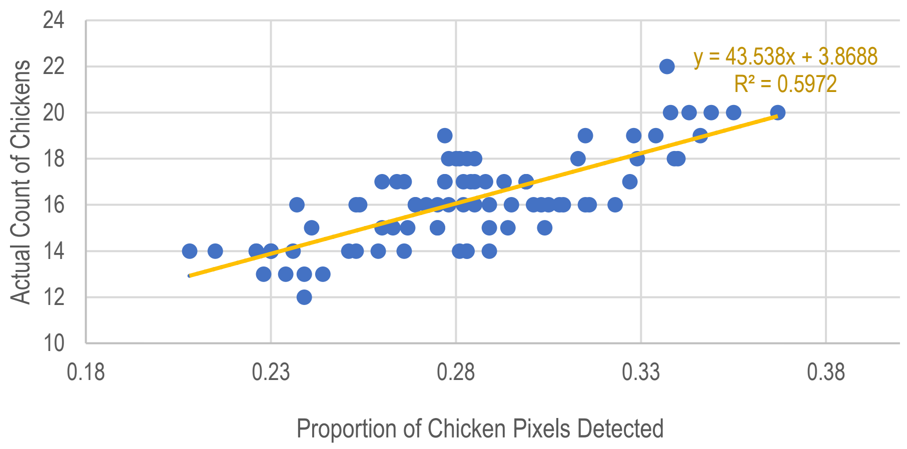


Figure 7. Pixel proportion vs. (manual) chicken count

The regression equation below was used to determine the bird count in other frames:

$$Chicken\ Count = 43.54(Chicken\ Pixel\ Proportion) + 3.87$$

We tested the performance of this equation on a test sample of an additional 65 images, and then applied it to the full set of frames for the full five-minute video.

(2) Feeding Detection

In the second approach, we attempted to identify and count the number of chickens that were actively feeding from the feeder, i.e., chickens with their heads pointing to the feeder. We focused on a smaller circular area around the feeder to exclude non-feeding chickens and used multinomial logistic regression to predict the number of feeding chickens. We used a radius of 145 pixels from the frame center at (160, 160), as seen in the right most mask in Figure 8. We then counted the number of chicken pixels with this smaller area.

This approach has limitations: for instance, in image C of Figure 8, we can see a bird withdrawing from the feeder (circled in orange) whose extra pixels at the boundary are still included. However, the general trend in Figure 9 shows that the pixel counts of feeding birds within the narrower circular area correlates well with a manual count of the actual number of feeding birds.

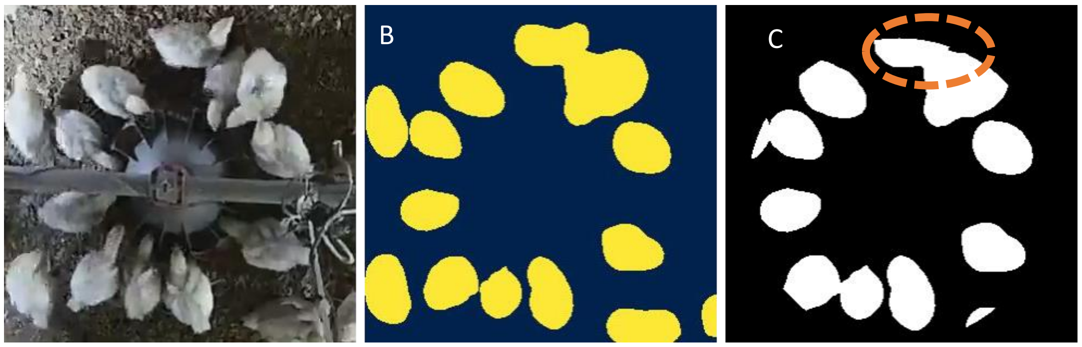


Figure 8. *left*: Raw image; *middle*: Masked image; *right*: Tighter circular mask around feeder

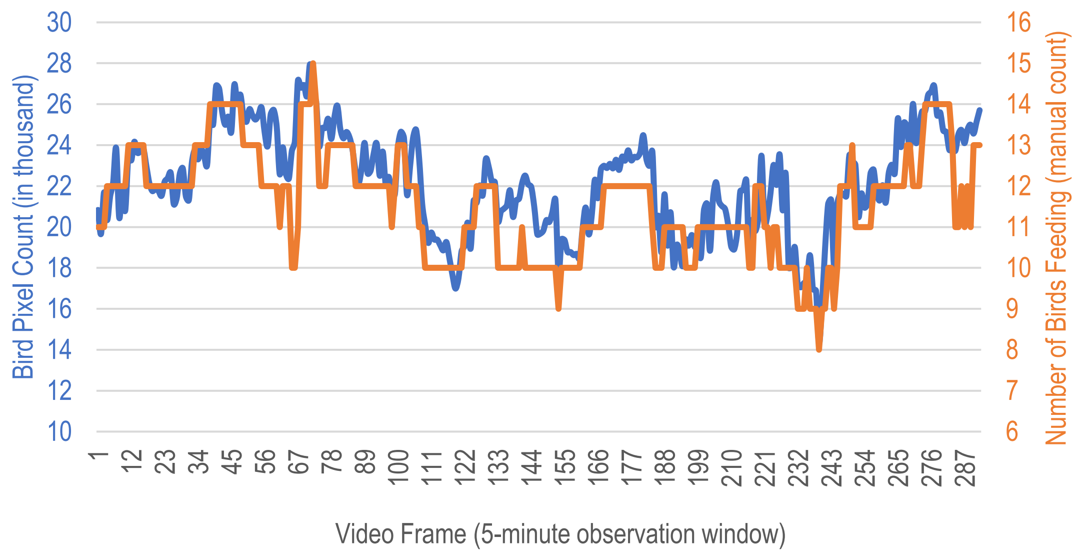


Figure 9. Distribution of bird pixels and the manual bird count over time

A multinomial logistic regression model was fitted because the total number of chickens in each frame falls within six categories, as seen in Figure 10. The category for 'number of birds = 10' was chosen as a benchmark and separate equations for each of the other categories were fitted in a 1 vs. all fashion. For example, the equation fitted for 'number of birds = 11' is below:

$$\ln\left(\frac{P(no.feeding = 11)}{P(no.feeding = 10)}\right) = b_{10} + b_{11} * (birds\_area)$$

The model used half of the frames from the five-minute video as a training set and the other half was used as a test set. The performance of multinomial model is discussed in the Findings section.

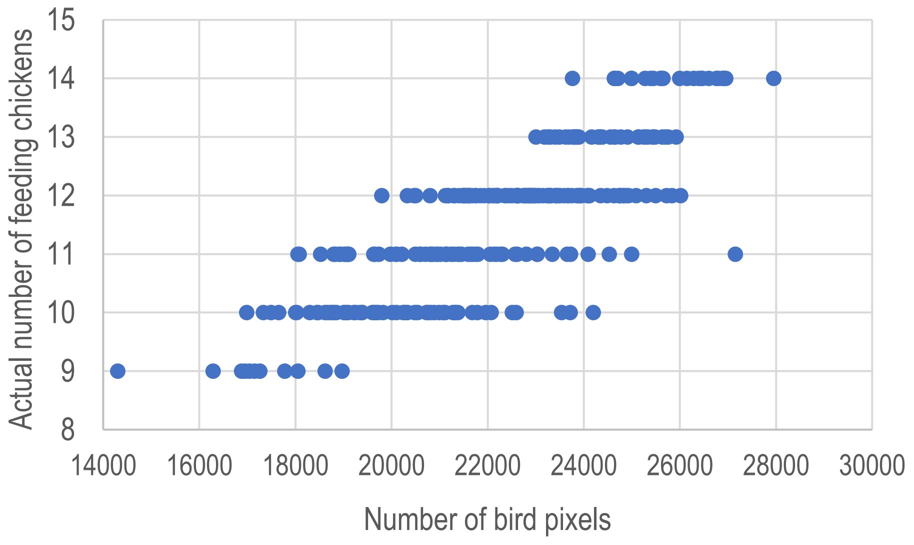


Figure 10. Number of feeding bird pixels (X) vs. actual number of feeding chickens (Y)

(3) Feeding Counts Using Contour Analysis

The third approach attempted to improve upon the second method to detect actively feeding birds by trying to determine whether chickens are facing the feeder. This involved two steps: (a) identify each chicken's head, and (b) determine whether the chicken is feeding based on the head's position to the feeder.

(a) To identify bird heads, the following steps were implemented:

- First, we calculated the centroid of each contour in the mask.
- We then detected the cardinal extreme points of each contour. These "extreme points" are identified by creating rectangular boxes around each contour. Those points that are tangent to the boxes are designated as cardinal extreme points (See Figure 11).
- Next, we determined the direction the chicken is facing, using the distance between the centroid of each contour and the extreme points. Given its protrusion, we predicted that the chicken head would be the point that is *farthest* from the contour centroid. In Contour 7 of Figure 11 (circled in orange), for example, the green point would be classified as the bird's head.

(b) To determine whether the chicken is feeding, the following steps are taken:

- For each contour, determine which of the four extreme points is *closest* to the center of the feeder.
- If this point is the same as the point for the determined head, we conclude that that the bird is feeding. Otherwise, the bird is not feeding.

Given the semantic (instead of instance segmentation) masks, the algorithm can run into issues with overlapping chickens as seen in Contour 3 in Figure 11 (circled in red).

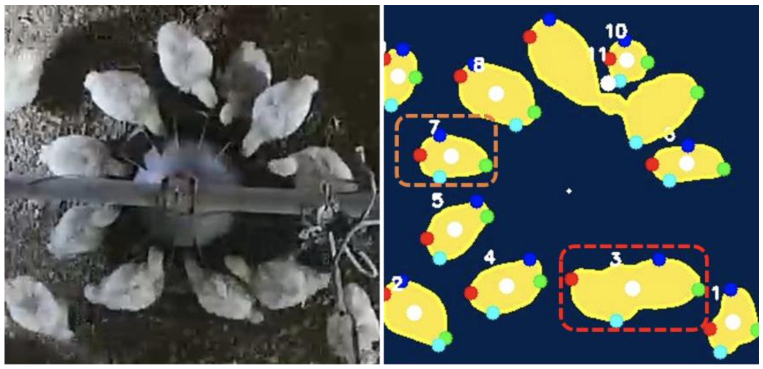


Figure 11. l*eft*: Raw Image; *r*ight: Contoured Image

Findings and Discussions

Semantic Segmentation Models Comparison

We evaluated our semantic segmentation models primarily through visual means. Tables 3 and 4 below provide the summary assessment using both full and cropped images.

Table 3. Semantic segmentation model outputs on a full image

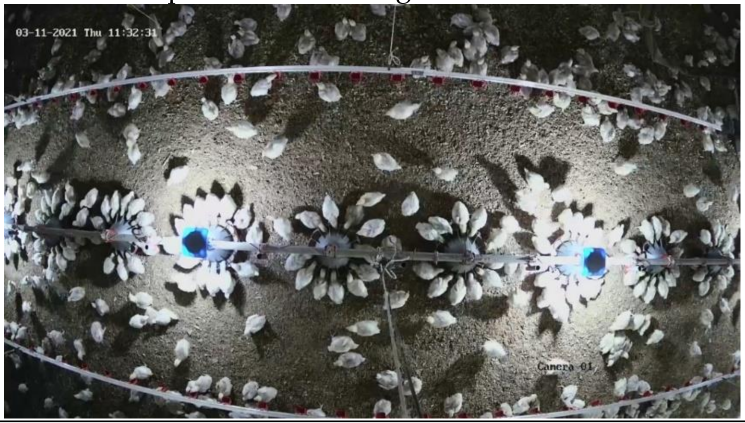
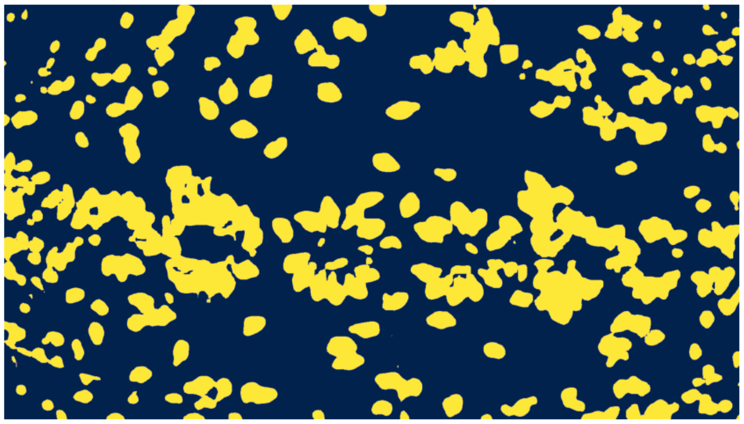
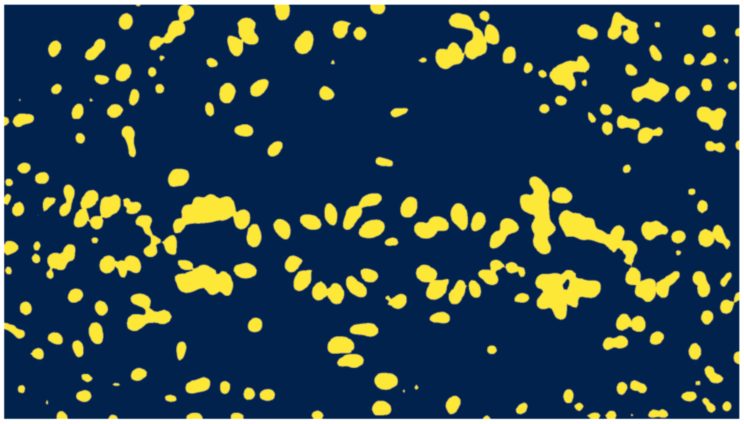
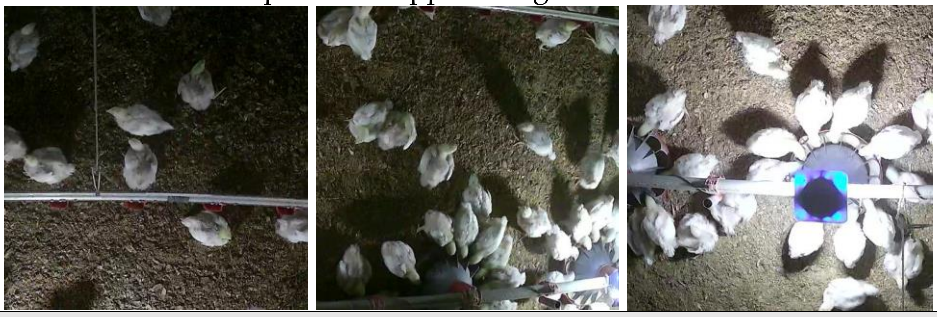
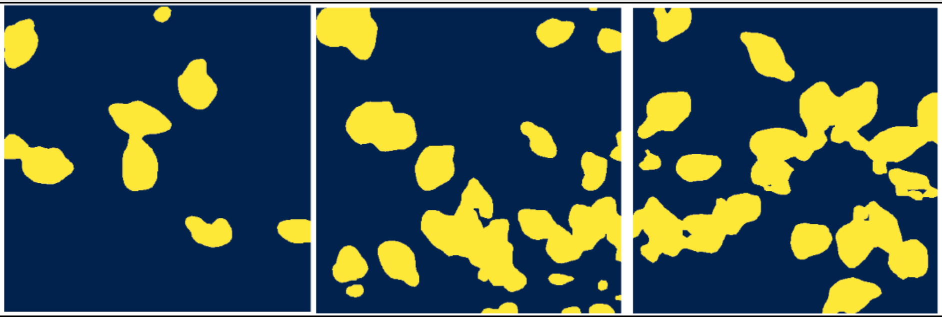
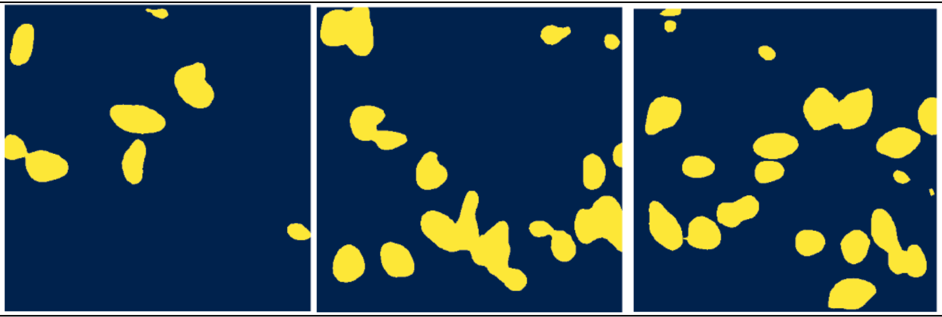
Original	
Models	**Mask results**
Model 1: `seg-5-4`	
Model 2: `ttun3-2022-03-22-18-20-1647973248`	
Model 3: `ttun3-2022-03-22-19-06-1647976015`	The Deeplab model took a long time to deploy, and eventually ran out of memory so the model did not produce any results.

Table 4. Semantic segmentation model outputs on cropped images

Original	
Models	**Mask results**
Model 1: `seg-5-4`	
Model 2: `ttun3-2022-03-22-18-20-164797324`	
Model 3: `ttun3-2022-03-22-19-06-1647976015`	The Deeplab model took a long time to deploy, and we eventually ran out of memories without any results.

From visual inspections, Model 2 performed well especially for the purpose of downstream analysis on chicken feeding behavior. While both Models 1 and 2 are based on FCN, the latter uses resnet-101, and thus has twice the number of residual feedbacks embedded in the architecture. As expected, Adam optimizer in Model 2 seems to play a role in its performance. Unfortunately, we had difficulties deploying our trained Deeplab model.

Both Models 1 and 2 generate better masks on the large full image, when the chicken pixels are smaller, whereas the models perform less well on zoom-in cropped images. In addition, our best semantic segmentation model, Model 2, has some occlusion issues, as birds that were directly underneath the feeder line which ran horizontally throughout each image (Figure 12). We used Model 2 as an input for further downstream analyses.

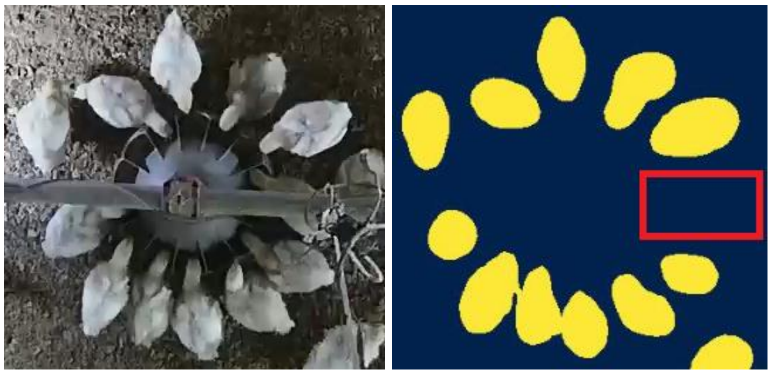


Figure 12. Occlusion issue where a bird under the feeder line is not captured in the model

Results from Chicken Feeding Behaviors Analysis

(1) Counting Chickens Near Feeders: Linear Regression Results

For the original regression equation, we found a standard error of 1.28 and an R^2 of 0.59. We can interpret this as meaning we would expect to see an error of about 1.28 birds for any given predicted count for a frame. From testing the derived linear regression equation on a test set of a randomly selected sample of 60 frames, we found a standard error of 1.36 and an R^2 of 0.64 which is in line with the results from the training set and gives hope that the equation would work in more generalized settings.

By looping each frame of the five-minute video through our regression equation, we can see a graph of the predicted count of chickens over time around a specified feeder for the five-minute video (Figure 13).

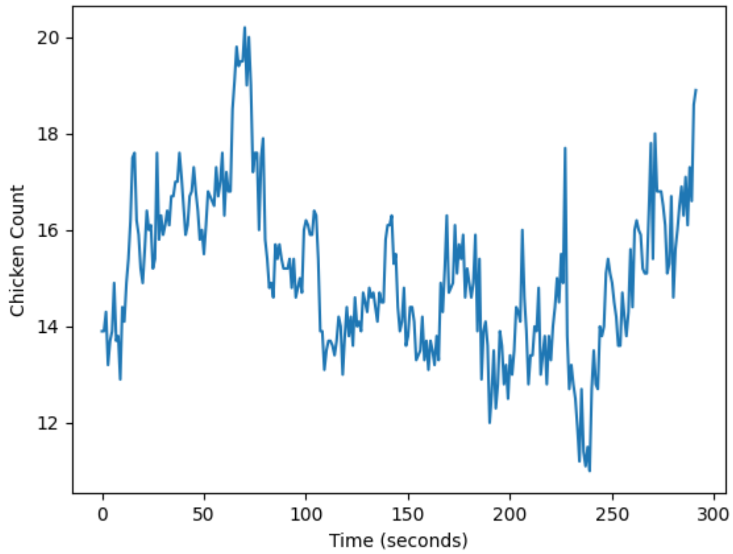


Figure 13. Count of chickens around Feeder "A" over time

Future applications of the linear regression approach include covering longer periods of time to create historical averages of chicken counts around each feeder which could then be used to determine anomalous periods of either low or high feeder activity. Instead of manually searching through video files for anomalies, these graphs could highlight and narrow down the timeframe to be inspected. However, further use of this technique does come with some caveats; the regression equation we derived will not be suited for birds of different ages and sizes since it is based on the proportion of chicken pixels within the frame which will change as the chickens grow over time for the same number of birds. Further steps with this approach could be taken to make it more generalizable.

(2) Feeding Detection: Multinomial Logistic Regression Results

The fitted multinomial logistic regression model (with the benchmark of 10 birds feeding prediction yielded the following coefficients):

```
Call:
nnet::multinom(formula = NumFeeding ~ birds_area, data = train.data)

Coefficients:
         (Intercept)     birds_area
Num_9      27.52377    -0.0015615216
Num_11    -12.62432     0.0006111543
Num_12    -28.79571     0.0013732927
Num_13    -63.12033     0.0027843809
Num_14   -114.39180     0.0047892933
```

The overall accuracy of the multinomial regression model was 55.14% (by calculating the percentage of correctly predicted cases from the total cases). The model accuracy on the training and test sets were 58% and 52%, respectively. The confusion matrix in Table 5 shows that model's prediction is only off by one bird in most cases.

Truth /Predicted	9	10	11	12	13	14	*Total*
9	**7 (64%)**	4 (36%)					11 (100%)
10	3 (5%)	**37 (61%)**	8 (13%)	13 (21%)			61 (100%)
11		22 (37%)	10 (17%)	**24 (41%)**	2 (3%)	1 (2%)	59 (100%)
12		4 (4%)	4 (4%)	**74 (76%)**	13 (13%)	3 (3%)	98 (100%)
13				**17 (46%)**	**16 (43%)**	4 (11%)	37 (100%)
14				1 (4%)	8 (31%)	**17 (65%)**	26 (100%)

Table 5. Confusion matrix of predicted number of birds vs actual number of birds

Previous works on similar approach have achieved overall accuracy metrics nearing 90%. These studies also experienced challenges such as counting birds with non-feeding behaviors (e.g., birds walking past or withdrawing from the feeder) [30].

One way to improve our model is to try to account for the dependency among adjacent frames. Since the images were extracted from a continuous five-minute video, there naturally is a correlation of bird counts between adjacent frames. In addition, birds that were under the feeder line were not consistently captured, thus leading to underestimations in the downstream multinomial model. The work by [20] addresses the challenge of undercounting hidden birds by compartmentalizing the feeder and predicting for individual parts of the feeder.

(3) Feeding Counts with Contour Analysis

Using contour analysis resulted in mixed results. Figure 14 shows an example of detected contours with 'x-y' labels where x represents the distinct contour number and y is either 1 (feeding) or 0 (not feeding). Here, only seven out of the eleven chickens that are feeding are detected correctly. Overlapping contours are also difficult to separate (circled in white in Figure 14) and instance segmentation models might be better suited to pair with this contour analysis.

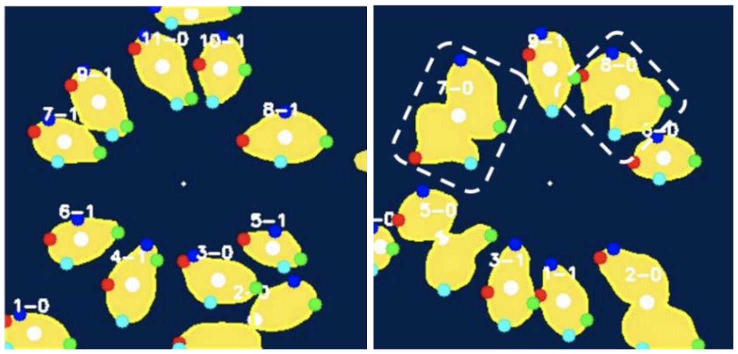


Figure 14. *left*: Contour analysis without overlapping chickens; *right*; Contour analysis with overlapping chickens

Although overlapping contour groups are a challenge in obtaining accurate bird counts, we believe it could be alleviated in combination with the pixel proportion approaches discussed earlier. Figure 15 shows pixel area count for all the contours detected on the frames for Feeder A. We can observe peaks for single chicken contours (between 900 and 1800 pixels), two chicken contours and so forth. By comparing contour size to typical areas for a single bird, we could determine a better count of chickens.

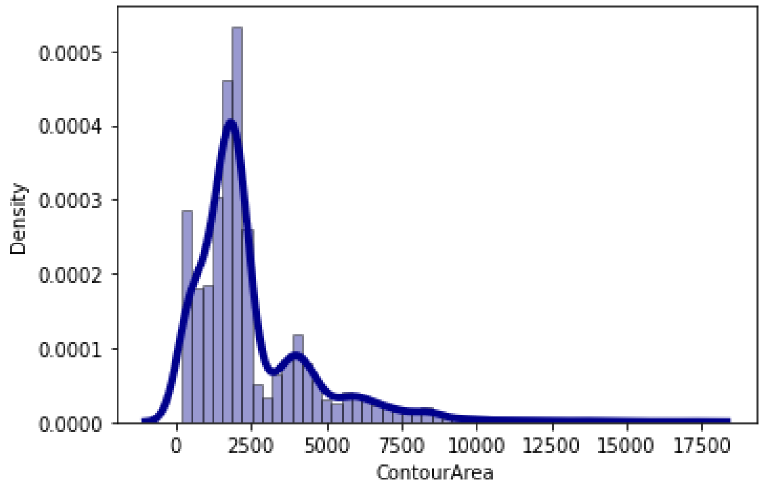


Figure 15. Density distribution of contour area size

Recommendations: Challenges and Opportunities

As discussed previously in this report and by past cohorts, labelling is a tedious process. We recommend that future cohorts familiarize themselves with SageMaker's Ground Truth infrastructure as soon as possible and try to use past cohorts labelling efforts so they do not have to start from nothing. Semantic segmentation labelling is especially tedious compared to other labelling approaches due to its task of accounting for every pixel within an image, so future cohorts should be prepared for this effort if semantic segmentation models are pursued.

We also found SageMaker machine learning pipeline to be inflexible. One cannot deviate much from the available built-in models and code templates. SageMaker does not seem to accommodate labelling jobs done outside of its own Ground Truth infrastructure.

We recommend future interested cohorts to use instance segmentation approaches which may provide more fruitful evidence for chicken behavior since individual birds can be tracked over time. One potential application could be to see how long each bird typically spends in an area.

Bibliography

[1] Pius, L. O., Strausz, P., & Kusza, S. (2021). Overview of Poultry Management as a Key Factor for Solving Food and Nutritional Security with a Special Focus on Chicken Breeding in East African Countries. *Biology*, *10*(8), 810. https://doi.org/10.3390/biology10080810

[2] Okinda, C., Nyalala, I., Korohou, T., Okinda, C., Wang, J., Achieng, T., Wamalwa, P., Mang, T., & Shen, M. (2020, September 9). *A review on Computer Vision Systems in monitoring of Poultry: A Welfare Perspective*. Artificial Intelligence in Agriculture. Retrieved April 20, 2022, from https://www.sciencedirect.com/science/article/pii/S2589721720300258.

[3] *AudioT - Crunchbase Company Profile & Funding*. Crunchbase. (n.d.). Retrieved April 20, 2022, from https://www.crunchbase.com/organization/audiot

[4] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[5] Le, Q.V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. (2011). On optimization methods for deep learning. *ICML*.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[7] Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J. (2001). "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". In Kremer, S. C.; Kolen, J. F. (eds.). *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

[8] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In *ECCV*, 2014.

[9] Alzubaidi, L., Zhang, J., Humaidi, A.J. *et al.* Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J Big Data* 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[11] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[12] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[15] Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J, et al. Recent advances in convolutional neural networks. *Pattern Recogn*. 2018;77:354–77.

[16] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765, 2016.

[17] Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: *ICIP* (2016).

[18] Aydin, A., Berckmans, D., 2016. Using sound technology to automatically detect the short-term feeding behaviors of broiler chickens. *Computers and Electronics in Agriculture*. 121, 25–31.

[19] Li, G., Zhao, Y., Hailey, R., Zhang, N., Liang, Y., Purswell, J.L., 2018a. Radio-frequency Identification (RFID) System for Monitoring Specific Behaviors of Group Housed Broilers,

10th International Livestock Environment Symposium (ILES X). *American Society of Agricultural and Biological Engineers*, p. 1.

[20] Guoming Li, Yang Zhao, Joseph L. Purswell, Qian Du, Gray D. Chesser Jr., John W. Lowe, 2020. Analysis of feeding and drinking behaviors of group-reared broilers via image processing. *Computers and Electronics in Agriculture*. 175 (2020) 105596.

[21] Kashiha, M., Pluk, A., Bahr, C., Vranken, E., Berckmans, D., 2013. Development of an early warning system for a broiler house using computer vision. *Biosys. Eng*. 116, 36–45.

[22] Fang, C., Huang, J., Cuan, K., Zhuang, X., Zhang, T., 2020. Comparative study on poultry target tracking algorithms based on a deep regression network. *Biosyst. Eng*. 190, 176–183. https://doi.org/10.1016/j.biosystemseng.2019.12.002.

[23] Guo, Y., Chai, L., Aggrey, S.E., Oladeinde, A., Johnson, J., Zock, G., 2020. A machine vision-based method for monitoring broiler chicken floor distribution. *Sensors* 20, 3179. https://doi.org/10.3390/s20113179.

[24] Lin, C.-Y., Hsieh, K.-W., Tsai, Y.-C., Kuo, Y.-F., 2018. Monitoring chicken heat stress using deep convolutional neural networks. 2018 ASABE Annual International Meeting. *American Society of Agricultural and Biological Engineers*, p. 1 https://doi.org/ 10.13031/aim.201800314.

[25] Zhuang, X., Zhang, T., 2019. Detection of sick broilers by digital image processing and deep learning. *Biosyst. Eng*. 179, 106–116. https://doi.org/10.1016/j.biosystemseng.2019.01.003.

[26] Zhang, H., Chen, C., 2020. Design of sick chicken automatic detection system based on improved residual network. 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). *IEEE*, pp. 2480–2485 https:// doi.org/10.1007/978-3-319-50835-1_14.

[27] Wang, J., Shen, M., Liu, L., Xu, Y., Okinda, C., 2019. Recognition and classification of broiler droppings based on deep convolutional neural network. J. *Sensors*, 2019 https://doi.org/10.1155/2019/3823515.

[28] Mortensen, A.K., Lisouski, P., Ahrendt, P., 2016. Weight prediction of broiler chickens using 3D computer vision. *Comput. Electron. Agric*. 123, 319–326. https://doi.org/10.1016/j.compag.2016.03.011.

[29] Mishra, A. (2019). *Machine learning in the AWS cloud*. Amazon. Retrieved from https://docs.aws.amazon.com/sagemaker/latest/dg/sms.html.

[30] Mehdizadeh, S.A., Neves, D., Tscharke, M., Nääs, I., Banhazi, T.M., 2015. Image analysis method to evaluate beak and head motion of broiler chickens during feeding. *Comput. Electron*. Agric. 114, 88–95.