# Agricultural Machinery Operator Monitoring System (Ag-OMS): A Machine Learning Approach for Real-Time Operator Safety Assessment

Terence Irumva[1,*], Herve Mwunguzi[1], Santosh Kumar Pitla[1], Bethany Lowndes[2], Aaron M. Yoder[3], Ka-Chun Siu[4]

[1] Biological Systems Engineering, University of Nebraska, Lincoln, Nebraska, USA.
[2] Neurological Sciences, University of Nebraska Medical Center, Omaha, Nebraska, USA.
[3] Environmental, Agricultural, & Occupational Health, University of Nebraska Medical Center, Omaha, Nebraska, USA.
[4] Department of Health & Rehabilitation Sciences, University of Nebraska Medical Center, Omaha, Nebraska, USA.
[*] Correspondence: irumvater@huskers.unl.edu

## HIGHLIGHTS

- A machine learning-based real-time monitoring system for agricultural machinery operators was developed.
- Categorization of tractor operators' behaviors in real-time into low, medium, and high-risk safety behaviors.
- Visual and sound feedback alert system of Ag-OMS triggered when operators engaged in unsafe operating behaviors.

**ABSTRACT.** *The 2015 CS-CASH (Central States Center for Agricultural Safety and Health, 2015) Injury Surveillance Surveys showed that around 19% of injuries to agricultural producers are related to tractors or large agricultural machinery, yet only a limited number of studies are found that address tools and methods for monitoring safety behaviors of agricultural machinery operators in real-time. The current safety behavior monitoring approaches require an in-person presence, which can be both time- and cost-inefficient, and the other available methods lack a feedback element to alert operators in real-time. As a result, the research presented in this study aimed to develop an automated approach to monitoring tractor operators' safety behaviors through the use of a trained machine learning (ML) model and a feedback system to alert operators when they engage in unsafe practices. For the ML model development, a skeleton-detecting algorithm called OpenPose was used to detect real-time human postures in a livestreaming video feed from a camera installed in the tractor cab. The model was then trained on three separate categories of tractor operators' safety operating behaviors, and this trained classifier was used to label operators' safety behaviors in real time based on the three safety classes. A feedback mechanism controlled by an onboard microcontroller was then used to alert the operators when unsafe operating behavior was detected to facilitate safe practices. This monitoring system, named Ag-OMS (Agricultural Machinery Operators Monitoring System),*

Submitted for review on 16 September 2022 as manuscript number JASH 15357; approved for publication as a Research Article by Associate Editor Dr. John Shutske and Community Editor Dr. Michael Pate of the Ergonomics, Safety, & Health Community of ASABE on 23 January 2023.

*monitored the ingress/egress operators' behaviors in real-time entering and exiting the tractor cab. The Ag-OMS successfully identified the ingress/egress operators' behaviors with an accuracy of 97% on the testing datasets for all safety risk categories.*

***Keywords.*** *Ag-OMS, Machine learning (ML), Safety behaviors, OpenPose, Tractor operator.*

Agricultural machinery and tractor operators' injuries account for 19% of all agriculture producers' injuries (CS-CASH, 2015), and for older farmers (over 65 years), the risk and severity of these injuries increase (McLaughlin and Sprufera, 2011). To improve agricultural machinery operators' safety, machinery-related injuries should get much more attention, especially in the Midwest, where row crops dominate agriculture and large equipment is commonly used by most farmers. Tractor operators' safety should be of paramount concern, as tractors are one of the most used pieces of agricultural equipment and power multiple implements like planters, tillage equipment, and fertilizer applicators. Further, tractor-related injuries account for 37% of injuries in the agriculture industry (Murphy et al., 2010). Ideally, a viable approach to addressing the tractor operators' safety issue would be to track safety practices that could potentially lead to injuries and alert operators when such practices are engaged in. By monitoring tractor operators' safety behaviors, it is possible to obtain real-time feedback on the safety risk level of the operator's behaviors. This important information can then be used to encourage the practice of safe behaviors in real-time using ML and onboard computers. To manually achieve such a monitoring task, the physical presence of a third party to observe operators' safety behaviors would be required, which is time-consuming and very challenging to implement. Consequently, the presented research explores an ML-based real-time operator monitoring system that can be field deployed to enable time-efficient autonomous monitoring of operators' ingress/egress behaviors.

The concept of monitoring machinery operators' safety practices using ML and computer vision has been researched in high-risk occupations like construction (Zhang et al., 2020). In agriculture, a similar approach has been used to monitor tractor drivers' safety behaviors while driving (Zhang, 2019); however, many research gaps still exist. For example, research work related to monitoring operators' safety practices at the most preliminary operating stages (i.e., entering a tractor) is missing. A thorough and comprehensive monitoring system would require that the operators' safety practices be monitored starting from their first interaction with the tractor. Subsequently, the objective of this study was to develop and test an automated system called the Agricultural Operator Monitoring System (Ag-OMS) for monitoring tractor operators' ingress/egress safety behaviors in and out of the tractor cab. The AG-OMS will have the ability to alert operators when they are engaging in high-risk operating behaviors. The AG-OMS was designed with a Deep Neural Network (DNN) classifier ML model, trained to identify the operator's safety practice in an image or a video frame and categorize the detected behavior as a low, medium, or high-risk safety behavior.

# Materials and Methods

### Model Development Software Platforms

To design the ML model for Ag-OMS, two main open-source software frameworks were used to code scripts for training and testing the model. The first framework used was the TensorFlow open-source library, which is commonly used in ML and artificial

intelligence for the computation of multiple complex analytical and mathematical tasks (Abadi et al., 2016). TensorFlow has multiple functions and modules that enable ML model developers to design training and testing scripts for neural networks (NNs) and calculate metrics to measure the models' efficiencies. The second software framework used to design the AG-OMS ML model was OpenPose. OpenPose is also an open-source Python library that uses a "two branch multi-stage" convolutional neural network (CNN) to detect joints and important features of the human body in an image or a video frame. The set of detected joints and body features is commonly referred to as a skeleton, and OpenPose version 1.7.0 used for this project can detect up to 18 different joints and features. Figure 1 shows body joints and features that are detected by OpenPose to form a full skeleton.

**ML Model Design**

The ML Model of AG-OMS was developed in such a way that it takes in an image/video frame of a tractor operator entering or exiting a tractor as an input and then identifies the risk level of the safety behavior being practiced by the operator detected in the input image/video frame. To assign the safety behaviors risk level categories, we used the Infrastructure Health and Safety Association's (IHSA) suggested three points of contact mechanism when climbing, which recommends having two hands and one foot, or two feet and one hand, in contact with the equipment (including ladders) that you are climbing (IHSA, 2019). Using this three points of contact model, we divided the operators' ingress/egress behaviors into three distinct categories: (1) entering/exiting the tractor while facing away from the cab (high-risk), (2) entering/exiting the tractor while facing into the cab but without all the three points of contact (medium-risk), and (3) entering/exiting the tractor while facing into the cab and with all the three points of contact (low-risk). Safety practices classified as medium and high-risk are considered unsafe, while low-risk safety practices are considered safe. So, the ML model takes in an input (image/video frame), detects the operator in the input, identifies the safety behavior being practiced by the operator in the input, and then classifies the detected safety behavior as a low, medium, or high-risk safety practice (fig. 2). This was done in two main stages: (1) obtaining and processing skeleton data from the input using OpenPose, and then (2) using the processed skeleton data as the input to a Multi-Layer Perceptron (MLP) classifier, which uses the skeleton data to identify

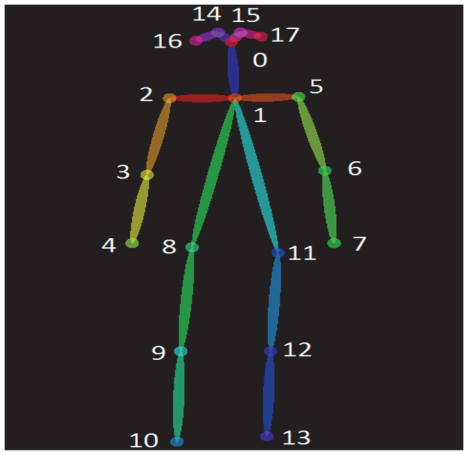


**Figure 1. OpenPose skeleton Image (Openpose, 2017).**

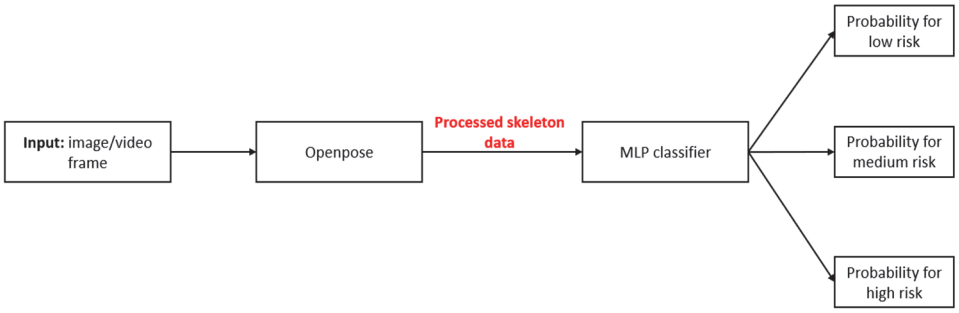


**Figure 2. Structure of ML model design. MLP classifier output (probabilities for each class that input belongs to it), and class with highest probability is picked as label for input.**

the safety risk category of the behavior being practiced by the operator. An appropriate label is then assigned to the input skeleton data (fig. 2). These two stages are discussed in detail below.

To design our customized ML model, we used Felix Chenfy's real-time action recognition model format (Chenfy, 2019), which uses OpenPose to detect everyday actions like jumping and running.

### *Obtaining and Processing Skeleton Data Using Openpose*

Using its aforementioned two-branch CNN, OpenPose identified and saved the skeleton data of operators detected in the input image or video frame. These stored skeleton data represent the relative positions of the 18 body joints and features detected by OpenPose. Obtaining processed skeleton data from an image or a video frame was done in three steps: (1) getting skeletons from the input (S1), (2) preprocessing skeleton data before feature extraction (S2), and (3) feature extraction (S3).

In the first step (S1), skeletons were extracted from the input image/video frame, and the number of people (operators) in the input was identified. The input images or video frames were of size 1920×1080, and the input images/video frames with no people/operators in them were deemed to have no skeleton data. This first step was achieved using two OpenPose classes called *Tracker* and *Skeleton_detector*. *Skeleton_detector* is an OpenPose class that detects a person's skeleton in an image or a video frame. The OpenPose module uses the CMU (Carnegie Mellon University) panoptic dataset, which contains around 1.5 million skeletons (Cao et al., 2017), and it is the dataset that the *Skeleton_detector* class uses to detect skeletons. On the other hand, *Tracker* detects the number of people in each video frame and tracks the skeleton of each detected person throughout the whole video (if video frames are used as inputs). The *Tracker* accomplishes this by saving each detected skeleton in the first video frame as Sk1[i], and if it detects another skeleton, Sk2[j], in the next frame, it decides whether Sk1[i] and Sk2[j] are the skeletons of the same person by comparing the distance between them (eq. 1). To understand how this tracking works, let's say that function *dist()* calculates the distance between skeletons detected in different video frames (which we can be calculated because all video frames have similar dimensions), and the constant *dist_thresh* is the threshold distance to determine whether Sk1[i] and Sk2[j] are skeletons of the same person. Then, if

$$dist\left(Sk1[i],\ Sk2[j]\right) \le dist\_thresh \tag{1}$$

S1[i] and S2[j] are skeletons of the same person, and if

$$dist\left(Sk1[i],\ Sk2[j]\right) > dist\_thresh \tag{2}$$

Sk1[i] and Sk2[j] are skeletons of different people (eq. 2)eq. This analysis is very important because we need to keep track of each individual's actions when we use the model on a livestreaming video. This is also achievable because, from frame to frame, there is not much difference in a person's posture since the model creates video frames from a recorded or livestreaming video at a speed of 10 frames per second (fps).

In the second step (S2), the raw skeleton data obtained in step one (S1) were preprocessed and reorganized to get it ready for feature extraction. Some of the modifications implemented on the skeleton data as part of data preprocessing before feature extraction included scaling the coordinates of joints' positions in the skeleton data. The scaling of coordinates was done because joints' positions given by OpenPose have different units for x and y coordinates, so they were rescaled to have the same unit for both coordinates. Also, during this preprocessing, all skeleton data missing a neck or thighs were discarded because they were most likely other objects in an image misidentified as operators' skeletons. However, if the skeleton data was missing joints other than a neck or thigh, they were filled in during this step to make feature extraction possible. These missing joints were filled in using their relative positions in the previous frames. For example, if the joint position for the left hand is missing in the skeleton data for the current video frame but present in the previous video frame, the joint position for the left hand in the current video frame will be filled in at the same coordinates as the left-hand joint position in the previous video frame. Again, this is a viable approach since frames are being extracted from the input video at a speed of 10 fps, so there are no big differences in joints' positions in consecutive video frames.

In the third step (S3), features were extracted from the preprocessed skeleton data. Features in this sense are the important and easily identifiable attributes of the skeleton data that can help us differentiate between different skeletons and predict/identify the safety practice risk level they depict. The extracted features were normalized joint positions, which, as the name suggests, are features scaled using the normalization feature scaling mechanism, which scales down features between zero and one (for simplicity) before feeding them to the classifier algorithm (MLP in our case). For image classification, feature normalization is recommended and very useful, as it allows "comparable effects in distance computations" (Xie et al., 2013). After extracting the features, a defined feature processing function converted them from raw features of individual images to time-series features (i.e., features at regular time intervals), and then they were saved in a .csv file.

***The Multi-Layer Perceptron (MLP) Classifier***

After obtaining and processing skeleton data from input images/video frames, a classification algorithm to classify skeletons as representing a specific safety risk among the three safety risk level classes was developed. The classifier was designed as a Multi-Layer Perceptron (MLP) with a Deep Neural Network (DNN) of three layers of 50×50×50, using the scikit-learn machine learning Python library (Pedregosa et al., 2011). MLP is one of the commonly used types of Artificial Neural Networks (ANNs), which consists of multiple layers of neurons/perceptrons with each layer fully connected to the previous and next layer, and the number of layers is selected and adjusted to improve the classifier's accuracy. The first layer of neurons is the input layer, and it takes in the input data (processed skeleton data in this case) and feeds it forward to a given number of hidden layers (the number is

chosen and adjusted by the model developer with the goal of improving the classifier's accuracy). Each neuron in the hidden layers is assigned a specific weight and bias (weights and biases are assigned and continuously updated when training the classifier), and these parameters are applied to the input data before the data is directed through an activation function and fed to an output layer of neurons to compute the respective probabilities that the input data sample belongs to each of the pre-defined classes (Noriega, 2005). When designing the MLP classifier for Ag-OMS, a Rectified Linear Unit (ReLU) activation function was used. ReLU activates neurons by the scaling the neurons' inputs to zero if they are negative, or keeping their actual value if they are positive (Agarap, 2018).

Another important technique that was implemented when designing the MLP classifier is regularization. Regularization refers to methods used to prevent a machine learning model from underfitting or overfitting the training dataset. For the MLP classifier, the L2 regularization method was used, and it prevented overfitting by driving the MLP parameters that are likely to cause overfitting close to zero. By shrinking these parameters, their impact on the model tunes out, which reduces the possibility of overfitting (Li and Bilmes, 2006). After designing the classifier architecture, the MLP was finalized in two main steps: training (S4) and testing (S5), which are discussed in detail in the following sections.

***Training (S4)***

Before training the MLP classifier, a comprehensive training dataset that represents all the training classes (i.e., the three safety risk level classes) was collected. To develop a good training dataset, it must contain training samples for a big enough part of the feature space to make accurate predictions, regardless of the part of the feature space that was used for predictions (Gu and Easwaran, 2019). The training dataset was created by recording three different videos of people/operators entering and exiting the tractor cab, with each video representing one of the three safety risk level classes (i.e., one video of operators practicing low-risk safety practices, one video in which they are practicing medium-risk safety practices, and another where operators are engaged in high-risk safety practices). To record these videos, a webcam (Model: Logitech) was installed in the tractor cab in such a way that its field of view always included the whole body of the operator as they entered or exited the tractor cab, and this was done to facilitate skeleton data extraction from the video frames by OpenPose. Four different students were used as operators while collecting the training dataset, and each student was recorded engaging in safety practices for each of the three safety risk level classes.

After recording the videos, they were transformed into images of 1920 × 1080 dimensions using a Python script to convert video frames into separate images at a speed of 10 fps. So, the resulting training dataset had three different labeled folders of images, with each folder representing a distinct class among the three safety risk level classes (and labeled as such), and each image was stored in the appropriate folder. The initial size of the training dataset was 3027 images. After the images for the training dataset were collected, they were processed using the aforementioned S1 and S2 steps to get them ready for training the MLP classifier. These first two steps made sure that all images in the training dataset contained skeleton data (i.e., there was an operator/person detected in the image) and that they all had labels consistent with the three defined safety risk level classes. After this preprocessing was done, 2370 images were remaining in the training dataset, and figure 3


**Figure 3. Analysis of accepted and rejected training images by S1 and S2 for each class.**

shows how many images were remaining in each of the three classes compared to the initial number of images.

After preprocessing the training dataset through the S1 and S2 steps, the third step, S3 (feature extraction), was applied to extract features of normalized joint positions (as discussed above) from the training dataset skeleton data, and these features were saved in a.csv file to be used for training the MLP classifier. After the training dataset was ready, the S4 step, which is training the MLP classifier, was implemented. The classifier was trained using the Adam optimization algorithm (Kingma and Ba, 2014) for 20 epochs with a learning rate of 0.001, and as mentioned above, the L2 regularization method was used while training to prevent the classifier from overfitting to the training dataset, and then the trained classifier was saved. During training, depending on the label (class) of a given skeleton in the training dataset, the classifier was trained such that when a similar (or somewhat similar) skeleton is detected in a different image or video frame, it will be labeled like that specific processed skeleton feature in the training dataset.csv file. This means that the classifier was trained to label each skeleton it encounters as one of the three safety risk level classes based on the skeleton features used to train it, which makes a good training dataset imperative in order to design an accurate classifier.

***Testing (S5)***

For the fifth step (S5), which is testing, the trained and saved classifier was used on real-world data (see fig. 4, which summarizes the relationships between the inputs and outputs of the five main model development steps). By using the *OpenCV* python module (which is an open-source library used for multiple computer vision and ML tasks) (Culjak et al., 2012), we get an output of a window with an image or a video frame containing prediction probabilities that the detected skeleton belongs to each of the three safety risk level classes (fig. 5). The numbers on the left side of the image in figure 5 show the classifier's prediction confidence level. The trained classifier can be used on images, pre-recorded videos, or livestreamed videos as input.

***Classifier Evaluation***

Before training, the training dataset was split into a training set and a testing set (known as the train-test split) (Tan et al., 2021), so that the testing set could be used later to evaluate the classifier's performance on a dataset other than the training set. After the train-test split, out of 2370 images in the training dataset, 1659 images were designated for training

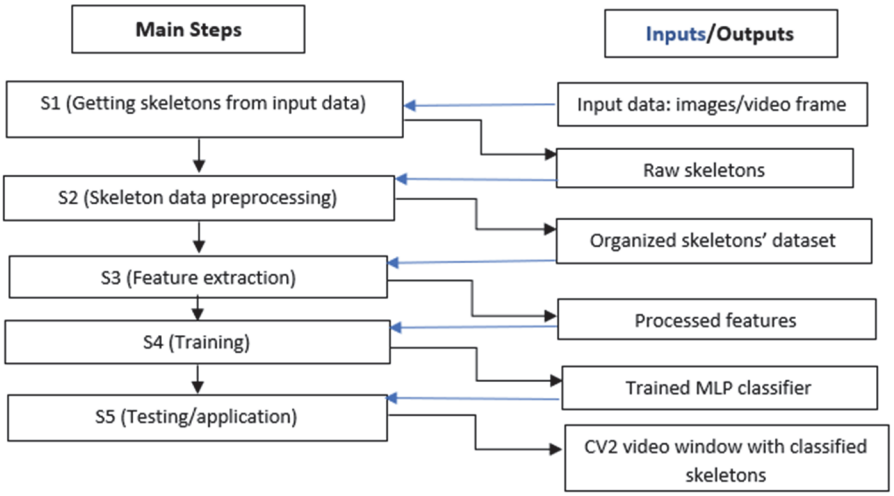


**Figure 4. Relationships between model development steps and their inputs and outputs.**


**Figure 5. Sample image output when testing MLP classifier (S5).**

(elements of the training set), and 711 images were designated for testing (elements of the testing set). The classifier was evaluated on the testing set using the precision, recall, and f1-score metrics (Raschka, 2014). These metrics give a good overview of how accurately the classifier performed on each individual class among the three classes it was trained on. Precision for a specific class is the ratio of the correctly predicted images in that class to the total number of images predicted by the classifier as part of that class. A high precision rate (eq. 3) implies a low rate of incorrect predictions. The recall metric (eq. 4) for each class is the ratio of the correctly predicted images in that class to the actual number of images that belong in that class. Lastly, the f1-score is weighted as the average of recall and precision, as shown in equation 5.

$$Precision(P) = \frac{C_p}{I} \quad (3)$$

where

$C_P$ = Correct predictions in each class

$I$ = The number of images predicted by the classifier to be in the same class

average of recall and precision, as shown in equation 5.

The values of the Precision, recall, and f1-score metrics for each class are given in table 1, and they evaluate the classifier to be adequately accurate.

$$Recall(R) = \frac{C_p}{T} \quad (4)$$

where

$C_P$ = correct predictions in each class

$T$ = the number of images belonging to the same class in the testing set.

However, the classifier also scored high on these evaluation metrics because it was evaluated on a testing set that was derived from a processed dataset that contained only labeled images with detectable skeletons, so, the classifier might not score as high when being used on unprocessed data. More details on the MLP classifier accuracy evaluation are discussed in the results and discussion section.

$$F1-score = \frac{2(P*R)}{P+R} \quad (5)$$

where

$P$ = Precision

$R$ = Recall.

**The Alert/Feedback System**

The alert system was designed to complement the ML model when it is being used to monitor operators' safety practices in real-time (i.e., on a livestreaming video), and to provide real-time feedback to the operators on the safety risk level of the safety practice that they are engaged in. So, if the ML model is being used in real-time and it detects and identifies a tractor operator's safety behavior from a livestreaming video, the alert/feedback system notifies the operator whether their safety practice is classified as medium or high-risk by the model. The alert system was created using an embedded microcontroller (Model: Arduino Uno), an LED, and a 2KHz 5V buzzer (Model: Adafruit). To establish communication between the ML model and Arduino, the serial Python module was used to allow serial communication between the Python integrated development environment (IDE) and Arduino. When the ML model identifies the detected tractor operator's safety practice risk level, it sends a specified character to Arduino through serial communication at a baud rate of 9600 (i.e., 9600 bits per second), and the transmitted character is different for each class. Then, the Arduino's serial monitor receives the transmitted character, and

**Table 1. Values of the precision, recall, and f1-score metrics for the classifier's accuracy evaluation.**

| Classes | Precision | Recall | F1-score |
|---|---|---|---|
| Low-risk | 0.99 | 0.98 | 0.99 |
| Medium-risk | 0.97 | 0.91 | 0.94 |
| High-risk | 0.97 | 0.99 | 0.98 |

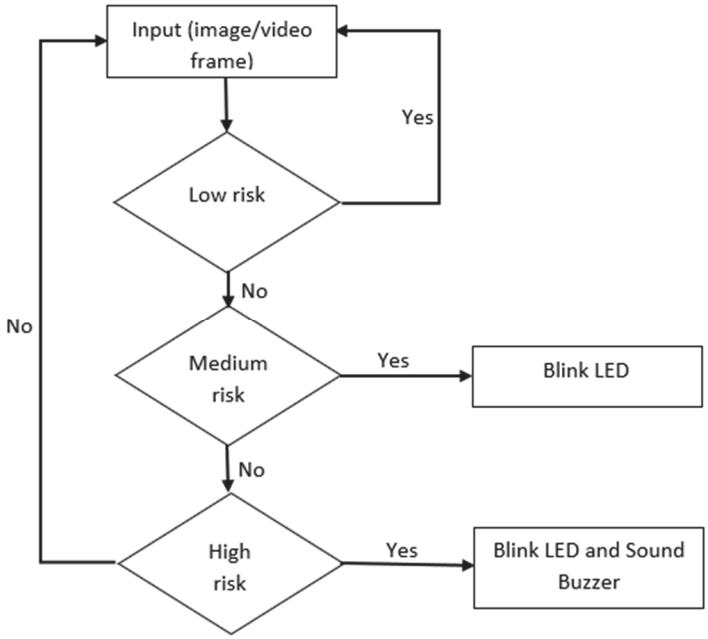


**Figure 6. Ag-OMS alert system logic flowchart.**

by using an Arduino script uploaded to the microcontroller, alerting instructions are issued accordingly.

Arduino gives out instructions to either continuously blink the LED, sound the buzzer, or do both based on the received character (fig. 6). When the received character indicates that a medium-risk safety practice was detected, the Arduino will give instructions to continuously blink the LED, and when the received character indicates that a high-risk safety practice was detected, the Arduino will give instructions to both blink the LED and sound the buzzer (fig. 6). When the received character indicates that a low-risk safety practice was detected, the alert system does nothing. The alert system was tested by running the ML model on a desktop computer, connecting the Arduino microcontroller and breadboard with an LED and an Adafruit buzzer to the desktop, and feeding a recorded video of operators entering and exiting the tractor cab into the model (the video is processed at a speed of 10 fps). However, to create a complete stand-alone system bundle, a *Jetson Nano 2GB* embedded computer can be used in the future to run the ML model and the alert system because of its small and convenient size, and good processing ability, which is adequate to run the system (more discussion about this in the future work section).

# Results and Discussion

As mentioned in the classifier evaluation section, the collected training dataset of 2370 images was split into a training set of 1659 images and a testing set of 711 images. This means that the classifier was trained only on the training data remaining after the train-test split (i.e., the 1659 images in the training set), because the remaining data (i.e., the 711 images in the testing set derived from the train-test split) was used to evaluate the

**Table 2. Confusion matrix to visualize the classifier's prediction results.**

| Confusion Matrix | | | | |
|---|---|---|---|---|
| True label | High risk | 1 | 1 | 298 |
| | Medium risk | 2 | 89 | 7 |
| | Low-risk | 308 | 2 | 3 |
| | | Low-risk | Medium-risk | High-risk |
| | | Predicted label | | |

classifier's accuracy. The classifier accuracy on the testing set was 97% (or 0.97), and this is the ratio of the number of accurate predictions in the testing set to the size of the testing set (Yin et al., 2019). This is a very good accuracy rating, but it might be a bit higher than the actual accuracy because, in real-time applications, the classifier will not only be tested on processed datasets with detectable skeletons (Yin et al., 2019; Sarker, 2021). In some real-world situations, skeletons may not be easily detectable in images, and differences in lighting conditions might also affect the model's ability to detect skeletons in images (Mitchell et al., 2019; Zhou et al., 2018). However, this accuracy rating is a good estimate of the model's prediction ability, because the training dataset was adequately diverse, as it accounted for multiple situations (ex: outdoors and indoors), and different camera placements.

The results obtained from evaluating the model's accuracy using the testing set are presented in table 2 using a confusion matrix, which is a method of visualizing a classifier's prediction results by showing the number of both correct and incorrect predictions in each class (Susmaga, 2004). The values in the blue-colored cells (table 2) are the number of images in the testing set that the classifier correctly predicted the tractor operator's safety practice, and the values in the non-highlighted cells are the number of images that the classifier labeled incorrectly in each class.

# Conclusion and Future Work

Regardless of a few adjustments needed to make the Ag-OMS more robust, it functions sufficiently well to be viewed as a promising approach to monitoring agricultural machinery operators' safety practices. The ML model has very good accuracy, and by training it using a more diverse dataset that addresses different lighting conditions and different operators' attire like hats and glasses, the model can be made more suitable for real-world operating environments. Apart from the ML model, the alert system is also very effective since it considers both the sight and sound senses. For future work, the system can be designed such that it is not very dependent on visualizing the whole body of the operator to identify the safety behaviors that they are practicing. Also, when collecting and processing the training dataset in the future, the FSPT (Feature Space Partitioning Tree) approach can be used to optimize the "training space" (i.e., a feature space with sufficient training samples) even more (Gu and Easwaran, 2019), by creating the dataset such that it represents larger portions of the feature space.

Another way of improving the training dataset is by identifying ideal positions to place the camera in the tractor cab when collecting the dataset. For example, as shown in figure 3, there was a high rejection rate for the medium-risk level class training images (after the first two steps (S1 and S2) of processing the training dataset), which might be a result of poor camera placement during the training dataset collection. This is because, as mentioned in the previous sections, when collecting the training dataset, the camera was mounted in

a different position before recording training videos for each class, to later evaluate how each camera position affected the accuracy of the model. Thus, the camera placement when collecting the medium-risk safety class training video was the least ideal. So, in the future, better camera positions can be identified by finding the camera placements that lead to capturing videos with smaller rejection rates (this can be achieved by making sure that all the operator's limbs are in the camera's frame of view).

Also, to use Ag-OMS in a real-world setting, a complete stand-alone system bundle can be created using an embedded computer (e.g., a Jetson Nano 2 GB) to run both the ML model and the alert system. The system bundle (including a camera) can be installed in the tractor cab, and the alerting system would give real-time feedback to tractor operators on their safety practices so that they could use that feedback to adjust their practices accordingly. Ag-OMS can also be used by other agriculture safety researchers to collect and analyze data on agricultural machinery safety behaviors by retraining the ML model to fit the specific behaviors being studied.

**Acknowledgments**

Thanks to CheeTown Liew, Machine Automation and Agricultural Robotics (MAARS) Lab, and the Cole Culver Biological Systems Engineering – Product Innovation (BSE-PI) service center for helping with data collection for the training dataset. This project is supported by the Central States - Center for Agricultural Safety and Health (CS-CASH), the NIOSH Agriculture, Forestry, and Fishing Grant (U54 OH010162), and the CS-CASH Pilot Grant.

# References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J.,... Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. *Proc. 12th USENIX symposium on operating systems design and implementation (OSDI 16)*, (pp. 265-283). Retrieved from https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU). *arXiv preprint arXiv:1803.08375v2*. https://doi.org/10.48550/arXiv.1803.08375

Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. *Proc. 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, (pp. 1302-1310). https://doi.org/10.1109/CVPR.2017.143

Central States Center for Agricultural Safety and Health. (2015). Agricultural Injuries in the Central States Region. Retrieved from https://www.unmc.edu/publichealth/cscash/_documents/_landing/landing-farm-survey-results.pdf

Chenfy, F. (2019). Multi-person real time action recognition based-on human skeleton [machine learning]. Retrieved from https://github.com/felixchenfy/Realtime-Action-Recognition

Culjak, I., Abram, D., Pribanic, T., Dzapo, H., & Cifrek, M. (2012). A brief introduction to OpenCV. *Proc. of the 35th Int. Convention MIPRO*, (pp. 1725-1730). Retrieved from https://ieeexplore.ieee.org/abstract/document/6240859

Gu, X., & Easwaran, A. (2019). Towards safe machine learning for CPS: Infer uncertainty from training data. *Proc. of the 10th ACM/IEEE Int. Conf. on Cyber-Physical Systems* (pp. 249–258). Association for Computing Machinery. https://doi.org/10.1145/3302509.3311038

Infrastructure Health and Safety Association (IHSA). (2019). 3-point contact - Vehicles and equipment. Retrieved from https://www.ihsa.ca/pdfs/safety_talks/3-point_contact_vehicles_and_equipment.pdf

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. https://doi.org/10.48550/arXiv.1412.6980

Li, X., & Bilmes, J. (2006). Regularized adaptation of discriminative classifiers. *Proc. 2006 IEEE Int. Conf. on Acoustics Speech and Signal Processing.* https://doi.org/10.1109/ICASSP.2006.1660001
McLaughlin, A. C., & Sprufera, J. F. (2011). Aging farmers are at high risk for injuries and fatalities: How human-factors research and application can help. *N. C. Med. J., 72*(6), 481-483. https://doi.org/10.18043/ncm.72.6.481
Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B.,... Gebru, T. (2019). Model cards for model reporting. *Proc. Conf. on Fairness, Accountability, and Transparency* (pp. 220–229). Association for Computing Machinery. https://doi.org/10.1145/3287560.3287596
Murphy, D. J., Myers, J., McKenzie Jr., E. A., Cavaletto, R., May, J., & Sorensen, J. (2010). Tractors and rollover protection in the United States. *J. Agromed., 15*(3), 249-263. https://doi.org/10.1080/1059924X.2010.484309
Noriega, L. (2005). Multilayer perceptron tutorial. School of Computing, Staffordshire University. Retrieved from http://www.amno.moph.go.th/research/uploadfile/1365058846mlp.pdf
OpenPose. (2017). Pose Output Format (COCO). CMU-Perpetual-Computing-Lab. Retrieved from https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_02_output.html
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O.,... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res., 12*, 2825-2830. Retrieved from https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
Raschka, S. (2014). An overview of general performance metrics of binary classifier systems. *arXiv preprint arXiv:1410.5330*. https://doi.org/10.48550/arXiv.1410.533
Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Comp. Sci., 2*(3), 160. https://doi.org/10.1007/s42979-021-00592-x
Susmaga, R. (2004). Confusion matrix visualization. In M. A. Kłopotek, S. T. Wierzchoń, & K. Trojanowski (Eds.), *Intelligent information processing and web mining. Advances in soft computing* (Vol. 25, pp. 107-116). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-39985-8_12
Tan, J., Yang, J., Wu, S., Chen, G., & Zhao, J. (2021). A critical look at the current train/test split in machine learning. *arXiv preprint arXiv:2106.04525*. https://doi.org/10.48550/arXiv.2106.04525
Xie, L., Tian, Q., & Zhang, B. (2013). Feature normalization for part-based image classification. *Proc. 2013 IEEE Int. Conf. on Image Processing*, (pp. 2607-2611). https://doi.org/10.1109/ICIP.2013.6738537
Yin, M., Wortman Vaughan, J., & Wallach, H. (2019). Understanding the effect of accuracy on trust in machine learning models. *Proc. 2019 CHI Conf. on Human Factors in Computing Systems.* Association for Computing Machinery. https://doi.org/10.1145/3290605.3300509
Zhang, M., Shi, R., & Yang, Z. (2020). A critical review of vision-based occupational health and safety monitoring of construction site workers. *Saf. Sci., 126*, 104658. https://doi.org/10.1016/j.ssci.2020.104658
Zhang, Y. (2019). Development of a machine learning system for safety of operators in agricultural farm environments using machine vision. Diss. Graduate School of Life and Environmental Sciences, The University of Tsukuba. Retrieved from https://tsukuba.repo.nii.ac.jp/record/50931/files/DA09065_abstract.pdf
Zhou, M., Lin, H., Young, S. S., & Yu, J. (2018). Hybrid sensing face detection and registration for low-light and unconstrained conditions. *Appl. Opt., 57*(1), 69-78. https://doi.org/10.1364/AO.57.000069