University of Novi Sad
Faculty of Sciences
Department of Mathematics and Informatics

Student
Katarina Petranović

Classification of different cattle behaviors using advanced machine learning algorithms

Master Thesis

Mentor:
Dr Oskar Marko

Novi Sad, September 2022

I would like to express my sincere gratitude to PhD student Dejan Pavlović for his advice, help, support and suggestions during the research and writing of this thesis. I would also like to thank my mentor Dr Oskar Marko and all professors at the Faculty of Natural Sciences who inspired me to pursue a career in data science. Finally, I would also like to thank my friends and family for incredible support, I would not be who I am today without them.

Contents

List of Figures and Tables

1. Introduction

Population growth causes an increased need for food that must be produced on the same or even smaller agricultural land, with limited resources and under effects of climate changes. Moreover, the crop yield must be increased by 70% by 2050 according to the Food and Agriculture Organization of the United Nations (FAO) in order to be able to feed the world's growing population, which will reach 9.1 billion by that time [1].

The increase in productivity can be secured by expansion of cultivated areas, increasing the amount of used pesticides and fertilizers, genetic modifications and applying precision agriculture technologies. However, there are debates about safety of GMO food for human health, and additional research is needed, while increasing cultivated areas is not an efficient solution. Also, utilization of pesticides and fertilizers can have a negative impact because of their property to remain in soil and environment for a long time. Further, they have negative effects on soil, micro flora, other organisms, environment, and human health [2]. Therefore, application of precision agriculture technologies, that implies usage of Machine Learning (ML) algorithms, Internet of Things (IoT), Big Data etc., has an advantage over all the above proposed solutions.

ML is a subfield of artificial intelligence (AI), devoted to understanding and building methods that exploit data to improve performance on some set of tasks. Deep learning (DL) is considered to be a subset of machine learning that imitates the way human brains think and learn (Figure 1).

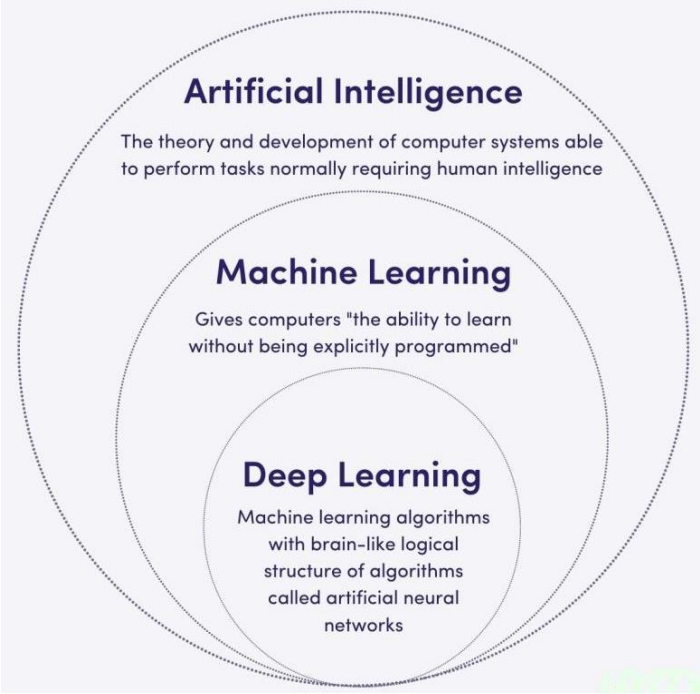


Figure 1: AI, ML and DL definition

ML algorithms build a model based on sample data, known as training data, in order to make predictions without being explicitly programmed to do so. Machine learning can mimic certain human brain functions like pattern detection, cognition, learning and even decision making while being able to process large quantities of data very quickly. In agriculture, ML has the potential to be applied to numerous areas with outstanding results, from detecting weeds and diseases, predicting yield and quality of crops, to gathering data, providing insights and offering predictions regarding livestock production [3].

For example, ML can be used for automated feeding, automated milk and genotype analysis, body weight predictions and other labor-intensive and time-consuming tasks usually manually performed by farmers [4]. Moreover, computer vision powered livestock management systems are developed so that farmers can monitor animals in real-time via mobile phones and ensure that the appropriate amount of food is supplied whenever necessary. Also, there are systems that provide remote control of milking and cleaning of the stall.

Figure 2: Collar sensors for tracking steer behavior; Source [5]

Furthermore, different agritech sensing systems are used for automatic detection of animal activity, such as eating, standing, laying, rumination etc. (Figure 2). Monitoring of these activities are important for detecting an unusual pattern in animal behavior that can be an early sign of health problems or welfare issues. Hence, if the disease or some other problem is caught in time and accordingly addressed it has a higher probability of being cured. For example, monitoring of lying and standing behavior can help in detection of lameness in dairy cows, which is a major health problem. Lameness negatively affects fertility of dairy cows and milk production and if detected late can cause significant reduction of the farmers' income [6]. Further, changes in time spend eating and ruminating can indicate health issues, but also assist in detection of oestrus leading to better conception rates [7].

Moreover, automatic monitoring is more efficient than manual one, as the latter requires high concentrations during the whole working day, which is a highly intensive

task thus being prone to errors due to human exhaustion. Further, as the average number of cows on farms increases every year, automatic monitoring is becoming a great asset to the farmers. The increasing trend in average number of cows on farms in England between 2011 and 2019 is highlighted on Figure 3 [8].

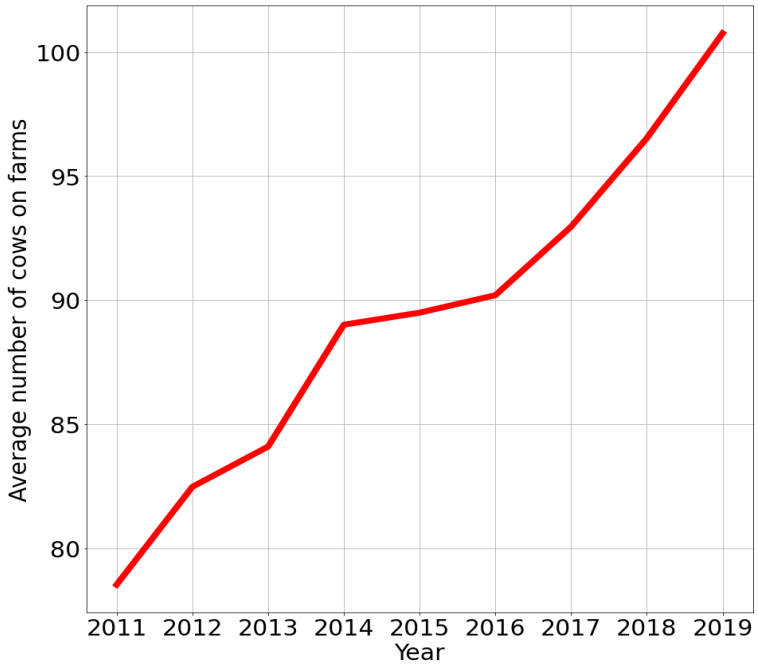


Figure 3: Average number of cows on farms in England from 2011 to 2019

The main goal of this thesis is to expand the possibilities of the ML model proposed in [5] in terms of the number of monitored activities performed through transfer learning, with the aim of producing a more efficient and effective solution. Section 2 gives a comparison between our work and relevant research from the literature. In Section 3, materials and methods used in this paper are described in detail. Furthermore, a short description of data is given, as well as the details on DL algorithms used in this paper. Additionally, performance metrics used for model evaluation are also defined within the same section. In Section 4 the model training and data preprocessing techniques are thoroughly described, followed by the presentation of classification results and the relevant discussion. Furthermore, certain techniques, used in order to additionally improve the model performance are presented as well. Finally, in Section 5 we recapitulate all the results, give final thoughts and define future work.

2. Related work

Different machine learning and deep learning algorithms can be more or less successfully trained to classify various cattle behaviors. For example, classic machine learning algorithm Random Forest (RF) was used in [9], Decision Trees (DT) in [10] whereas Support Vector Machine (SVM) algorithm was used in [11]. SVM [12] represents a popular machine learning method successfully applied to many classification and regression tasks in different domains, with a good mathematical background, high generalization ability, and ability to find global and nonlinear classification solutions. However, it needs multiple models for solving multiclass classification problems and performs better with low amounts of training data and large number of features. On the other hand, the advantage of DT [13] is that data preprocessing is not needed, and it can handle collinear data, nevertheless chances of overfitting are high, and it may grow to be very complex for complicated datasets. Finally, RF [14] is a very accurate and powerful model that efficiently handles overfitting and implicitly performs feature selection. On the other hand, it is computationally complex and can be slow if the forest or dataset is large. Additionally, certain research articles reported multiple algorithms used for cattle activity classification such as SVM and DT in [15] or k-Nearest Neighbors (kNN), Naïve Bayes (NB) and SVM in [16]. kNN [17] represents simple classification algorithm that does not require a definition of large number of hyper parameters, yet requires large real time computations compared to other ML algorithms, while NB [18] supports both binary and multiclass classification problems and handles irrelevant features successfully, however it assumes mutual independence of dataset and cannot handle large amounts of data.

Even though these approaches can be successful while discriminating samples from different cattle behaviors, they still require considerable effort to extract features from sensor signals for that purpose, which can be often a time-consuming and highly complex process. Furthermore, new classes demand the engineering of new features and the extension to additional activities requires significant effort.

On the other hand, implementation of DL algorithms such as Convolutional Neural Networks (CNN) is reported in [5, 19], or Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM) in [20]. One of the advantages of DL algorithms is that they automatically extract important features, and it is easy to expand the capabilities of the model to additional classes. Moreover, CNN is known for their ability to identify important features and ability to minimize computation in comparison with a regular network. However, a large amount of data which is not always available is required for training an effective algorithm and this process can prolong for a long time if classification problem or network architecture is complex. Further, LSTM has the ability to process sequential data and learn complex dependencies of data across time dimension. However, they can require long training time and a lot of resources. Transfer learning

represents the procedure that assumes the reuse of knowledge learned from a prior assignment which is further used to upgrade prediction power and improve generalization in a new task while reducing training time of the ML model [21].

Despite the fact that these related papers present solutions to the similar classification problems, they used not only different methodologies but also non-identical datasets collected with different devices, with different number of animals and cattle behaviors, which is highlighted in Table 1.

Table 1: Comparison of related work

	Algorithm	Number of animals	Monitored behaviors
Pavlovic, [5]	CNN	18	Eating, Rumination, Other
Rahman, [9]	RF	22	Gazing, Ruminating, Walking
Gonzales, [10]	DT	58	Foraging, Resting, Rumination, Traveling, Other
Hamilton, [11]	SVM	3	Rumination and Non-Rumination
Benaissa, [15]	SVM, DT	10	Feeding, Rumination, Other
Benaissa, [16]	kNN, NB, SVM	16	Feeding, Lying, Standing
Kasfi, [19]	CNN	22	Gazing, Other
Peng, [20]	RNN/LSTM	6	Feeding, Head butt, Licking salt, Lying, Moving, Ruminating-Lying, Ruminating-Standing & Social licking
Current study	CNN, CNN-LSTM	4	Eating, Rumination, Other, Standing, Lying

For example, in [5] they used CNN in order to predict three cattle behaviors eating, rumination and other, which are core to the early detection of health and welfare issues. Furthermore, they had data about 18 cows collected with two devices (collars and halters) and their model had F1 score of 82%. In [9] they, trained RF algorithm to classify three behaviors, gazing, rumination and walking in order to be able to detect cows with lameness disease. Further, they collected data with four different devices (collar, halter, ear tag and accelerometer) from 22 cows and their model F1 score ranged between 89% and 93%. In [10] they used DT algorithm to classify five steer activities, foraging, resting, rumination, traveling and other in order to be able to detect abnormalities in animal behaviors that indicate illness. Data was collected from 58 animals with tree devices (collar, accelerometer and magnetometer GPS) and model recall was estimated to be 90%. In [11] they tried to detect rumination, which is a key indicator of cattle health, using SVM. Further, data was collected from three cows with two devices (bolus and accelerometer) and their model's F1 score was estimated to be 86%. In [15] they compared performances of DT and SVM algorithms trained on data collected with

accelerometer and collar device. Further, data was collected from ten cows while algorithms were trained to classify three behaviors, feeding, rumination and other where DT algorithm had 93%, while SVM 91% accuracy. In [16] kNN, NB and SVM algorithms were trained to classify three dairy cows behaviors, feeding, lying and standing. Algorithms were trained on data collected from sixteen cows with three devices (collar, pedometer and accelerometer) and models accuracies were 99%. In [19] CNN was trained to detect gazing on data collected from twenty two cows with two devices (collar and accelerometer). This model F1 score was estimated to be 84%. In [20] LSTM network was trained to classify six different behaviors, feeding, head butt, licking salt, lying, moving, ruminating-lying, ruminating-standing and social licking on data collected with collar and accelerometer devices from six cows. The reported model F1 score and accuracy was 88%.

As previously stated, within this analysis, we used the same data as reported in [5], with an additional data source that enabled expanding the multiclass into multiclass/multilabel classification task. Multiclass classification indicates that only one label is correct, while in the multilabel classification any combination of labels is allowed [22]. For the classification purpose we used pre-trained CNN models used for identification of three cattle activity states reported in [5] and through transfer learning procedure managed to classify a larger number of monitored activities while not increasing model complexity of and time required for model training.

3. Materials and Methods

3.1. Data

Dataset used in this thesis consists of data collected from three farm trials in the Easter Howgate Farm, Edinburgh, UK, in the period from June 2015 to October 2016. It contains information about 18 animals collected with special sensors on two different devices: Afimilk Silent Herdsman Collar [23] and Rumiwatch Halter device [24].

Neck placed collars provide information about animal behaviors by tracing raw acceleration traces from three axes (x, y and z) and measuring the overall animal movement and contractions of neck muscles (Figure 4). Muzzle placed halter measures changes in pressure induced by the jaw, providing information about animal activity. Both devices operate at a sampling frequency of 10 Hz [5]. Animal behaviors detected with halter device are:

a) Rumination – the process in which the animal is re-chewing and re-swallowing partially digested food in order to improve nutrient absorption
b) Eating – the animal is taking food from feeding source
c) Other – the animal is not ruminating nor eating

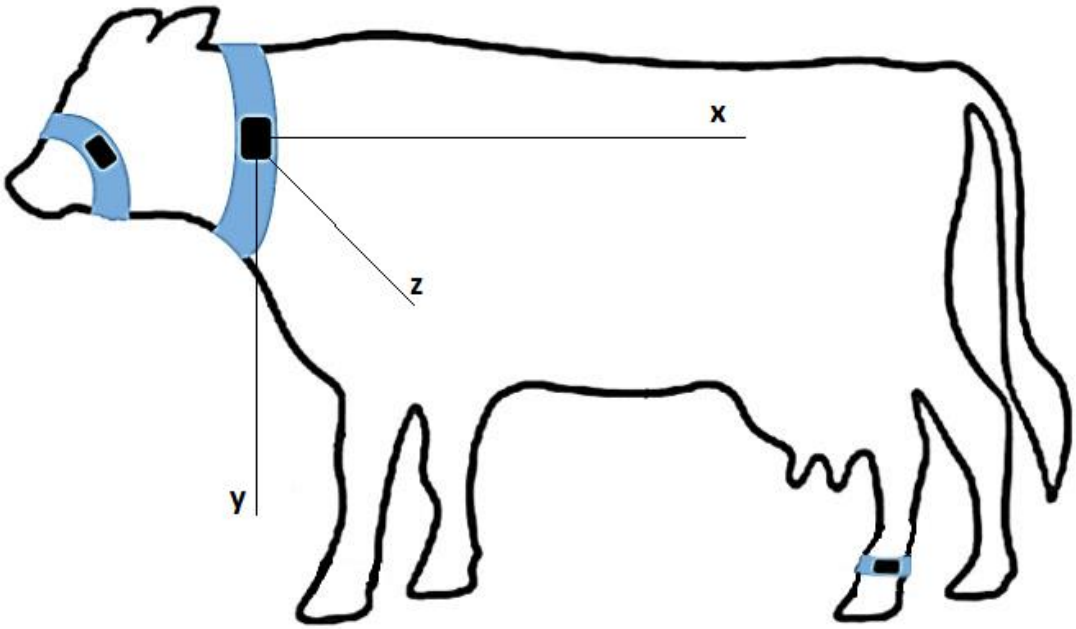


Figure 4: Position of halter, collar and pedometer device

In addition, within this analysis we used data obtained from one more device. Pedometers were placed on the rear leg of 4 cows collecting data across three axes, like collar and halter, multiple times per second. The sampling frequency of this device is 4Hz, so we needed to do resampling to adjust it to data collected with other two devices. Pedometers are most commonly used for the detection of oestrus (or "heat") to optimize herd reproductive efficiency. The oestrus detection is commonly based on standing, lying and walking time, number of steps an animal takes during a day or information about animal motion in general. All these activities can be detected by a pedometer device with a high precision. Within this analysis we used information about standing and lying activity.

3.2. Data Preprocessing

Despite the attempt to fix all collars and halters in the identical positions, this was not possible due to variations in steer anatomy. Further, constant movements of cattle were constantly causing shifting and rotation of devices. In order to overcome this problem and retain constant collar position, discrete difference along every axis is computed so that only the accelerations due to animal motions are recorded [5]:

$$\Delta s[t] = s[t] - s[t-1], \ \forall s \in \{x, y, z\} \qquad (7)$$

where $s[t]$ are the raw acceleration signals for all axes, *x*, *y* and *z*, $\Delta s[t]$ is the resultant signal for a given axis at time step t.

Every data point in this dataset is a one-dimensional time series with three signals, corresponding to the following axes: x, y and z. The device is oriented in such way that x-axis is parallel to the ground and animal body, y-axis is orthogonal to the ground and z-axis is orthogonal to the animal body and parallel to the ground, as shown on Figure 4.

In order to capture animal behavior, we made blocks of 90 seconds length without overlaps for every signal, where exactly one activity was assigned to every block. If there were several different activities in the blocks, one activity was chosen according to the majority vote.

3.3. Algorithms

In this paper, we used advanced machine learning algorithms for classification, Artificial Neural Networks (ANN). Moreover, algorithms we used in this thesis are Convolutional Neural Networks (CNN) and Long Short Term Memory Recurrent Neural Networks (LSTM-RNN).

3.3.1. Artificial Neural Networks

Artificial Neural Networks are machine learning models that were inspired by Biological Neural Networks of human brains. Their purpose is to learn values of parameter θ, minimizing some predefined loss function, and find function $f(X;\theta)$ that approximates function f^*, which maps input data X into an output $y = f^*(X)$. Here, loss function measures how similar functions f and f^* are. In these models, many different functions are composed together and that is why these algorithms are called 'networks' [25]. These functions are called layers and classical network representation is most commonly done with the following layer types (Figure 5):

- Input layer – the first layer in the network consisted of data samples. Inputs are independent variables and are called features.
- Output layer – final layer of the network; Approximation of the output data *y*.
- Hidden layer – everything placed between input and output layer.

Number of hidden layers can vary, and it indicates the depth of the model. This is why these algorithms are usually categorized as Deep learning methods.

Each layer is represented with a vector of values interpreted as neurons, whereas each neuron is connected with all other neurons in the network. Neurons or units are the reason why these algorithms are called 'neural'.

Artificial Neural Networks training consists of two phases, namely, forward propagation and backward propagation.

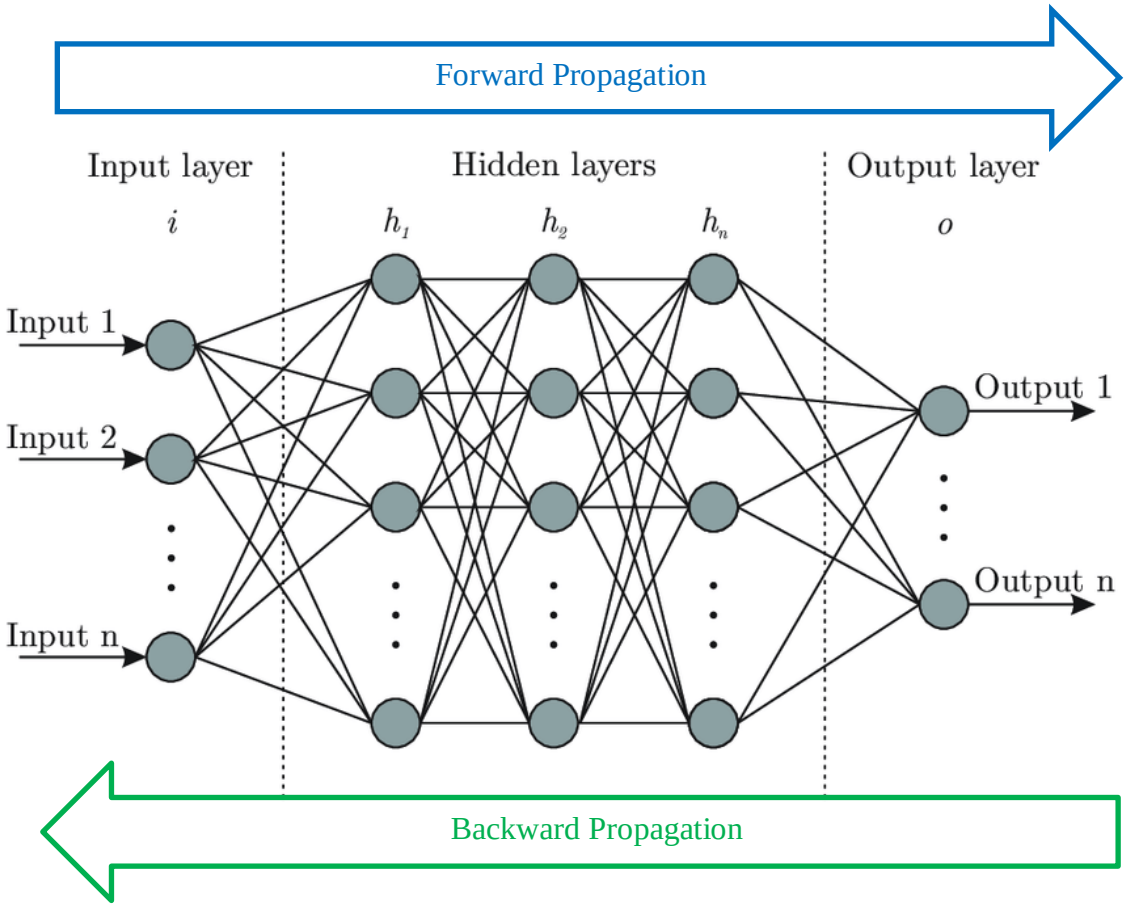


Figure 5: Architecture of Artificial Neural Network

In the forward propagation phase, the network takes input data (X) where input values are multiplied with weights (w) assigned to each neuron followed by addition of biases b (1).

$$Y = wX + b \quad (1)$$

Initially, corresponding weights are most commonly represented with small values close to zero. Further, resulting values are forwarded to hidden layers where activation functions are applied in order to introduce non-linearity to the network, so that complex, underlying patterns can be detected. The process is repeated until the values of the output are obtained. Obtained output values are called predictions and in order to determine model performance, we need to compare predicted and actual values i.e. calculate an error which is called loss function.

In the backward propagation phase, loss function is fed back to the network to optimize the network parameters (weights and biases) by minimizing loss function.

This represents an iterative process and is performed until the model learns data specifics. One pass of the whole dataset through a neural network is called an epoch.

3.3.2. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are traditionally used in the field of computer vision where input data is in the form of digital images, videos and other visual inputs, and have been proven as successful tool for image classification, object detection, instance segmentation and similar tasks. Convolution is mathematical operation which takes two functions (*f* and *g*) and gives as result third function ($f * g$) that expresses how the shape of the function f is modified by the function g [25]. It is calculated as the following integral:

$$[f * g](t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \quad (2)$$

Convolutions are applied using filters i.e., kernels; small vectors or matrices to extract features from input data valuable for various applications. Moreover, filters at the beginning detect some rough details, whereas filters from deeper layers extract more specific information present within the input signal.

CNN can also be used for one dimensional data such as text and time series signals such as audio signals, stock prices, temperature oscillations, heart rate etc. In that case, 1-dimensional kernels are used as they are able to extract features along the time dimension, taking into account the temporal behavior of the signal.

In the convolutional network architecture, few different types of layers are present:

- *Convolutional layer* – performs dot product between data and kernel to obtain useful information and extract features
- *Nonlinear layer* – consists of an activation function that takes output of the convolutional layer and creates the activation map; activation function is an element wise operation, so the dimensions of the input and the output are identical
- *Pooling layer* – allows downsizing of the data by looking at the neighborhood of the element of sequence (in case of time series data) or the neighborhood of the pixel (in case of image data) and performs some statistical operation such as finding maximum (Max pooling) or average (Average Pooling) value
- *Dropout* layer – randomly chooses neurons that are ignored during the training of the network; dropout rate p is defined and represents percentage of neurons that are masked and not used during training
- *BatchNormalization* layer – applies a transformation of the input data so mean data value and its standard deviation is close to 0 and 1, respectively.

3.3.3. Recurrent Neural Networks

Recurrent Neural Networks (RNN) are neural networks that have internal memory or, in other words outputs from the previous steps of a recurrent network are forwarded to the next step and they have effect on the next output of the network. Consequently, RNNs are suitable for sequential data and time series that are time dependent.

However, RNNs suffer from vanishing gradients. Through constant gradient calculations, network parameter values become smaller and smaller with every derivation. This is why we say that RNNs suffer from short memory, i.e. after some time they are 'forgetting' information. One way to solve this issue is to use modified version of recurrent neural networks, namely, Long Short-Term Memory (LSTM) networks. LSTMs have gates that are 'deciding' what information is irrelevant and can be forgotten and what information is important and needs to be forwarded further.

More precisely, LSTMs have feedback connection which enables them to process entire sequences of data, such as time series, while retaining useful information about previous data in the sequence. Every output of LSTM at every time point depends on three things:

- Cell state - current long-term memory of the network (*c*)
- Hidden state – output of the previous point in time (*h*)
- Input data of the current time step (*x*)

Additionally, LSTMs have three types of gates: forget gate, input gate and output gate, which filter data going through the network. All gate types are highlighted in Figure 6.

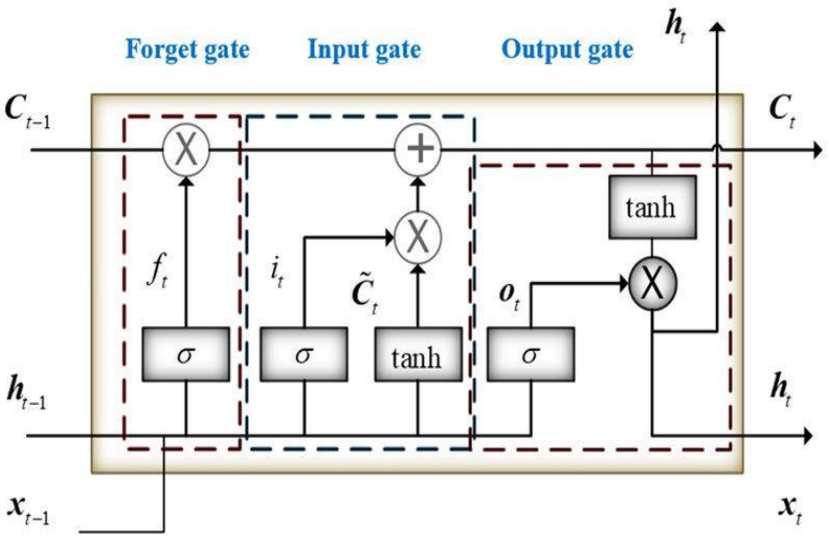


Figure 6: LSTM cell

As illustrated in Figure 5, LSTM cell starts with forget gate, which function is to determine what is irrelevant and should be forgotten according to the previous cell state (C_{t-1}), hidden state (h_{t-1}) and new input data (x_{t-1}). At this gate, there is a neural network trained in a way it takes new input data (x_{t-1}) and previous hidden state (h_{t-1}) as input and returns vector of values from the interval [0, 1], where vector values close to 0 are categorized as irrelevant and those close to 1 as highly relevant. Further, this vector is multiplied with the previous cell state (C_{t-1}); hence irrelevant values of cell state will be multiplied with small values close to 0 thus having small influence in the following steps.

Afterwards, the input gate makes the decision what new information should be added to the cell state, given the previous hidden (h_{t-1}) state and new input data (x_{t-1}). Within this gate, two neural networks are present. The first one is *tanh* activated neural network which learns the way to combine previous hidden state (h_{t-1}) and new input data (h_{t-1}) in order to obtain an updated vector for cell state ($\tilde{C}_t$), whereas *sigmoid* is activated, input gate network, detects relevant components of the updated vector (i_t). The latter is similar to the forget gate, and likewise obtains a vector with values from interval [0, 1]. At the end, outputs of these two networks are multiplied point wise to update the cell state, which is afterwards added to previous cell state (C_{t-1}) to obtain the new cell state (C_t).

Finally, the output gate returns a new hidden h_t state considering new cell state (C_t), previous hidden state (h_{t-1}) and new input data (x_{t-1}). Initially, *tanh* function is applied point wise to the current cell state, to obtain a vector with values from the interval [-1, 1]. The previous hidden state (h_{t-1}) and the current input data x_{t-1} are passed through the neural network with *sigmoid* activation function to obtain filter vector (o_t) similarly as in forget gate. Finally, this filter vector is multiplied point wise with the vector obtained after *tanh* activation and the new hidden state (h_t) is obtained.

3.4. Additional analysis

3.4.1. Signal overlaps

As mentioned before, we trained our models on a dataset which consists of time series divided into blocks of 90 seconds without any overlap between subsequent blocks. Even though the selected time window covers the whole period of rumination (main activity of interest) contractions, which are regular and typically occur at 40–60 second intervals, whereas the whole activity last considerably longer, certain ‘instantaneous’ activities such as sudden head movements can disrupt homogeneity of the 90 seconds blocks. Furthermore, instances during the transition periods between different activities can indicate more than one behavior which additionally disrupts the model training.

Therefore, we also tried to train our models on data where we took blocks with overlaps of 25%, 50% and 75% (Figure 7).

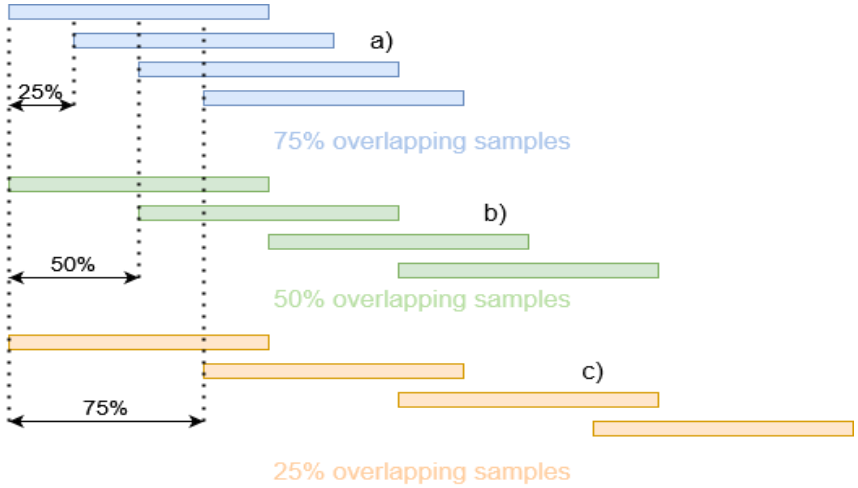


Figure 7: Overlaps of 90s blocks for signals of a) 25%, b) 50% and c) 75%

3.4.2. Data Augmentation

Data augmentation is a way to increase the amount of data used for model training by adding slightly modified copies of original data to the original dataset. Moreover, it behaves as a regularizer and prevents model overfitting to the underlying patterns in training data [26].

In this paper, we used *tsaug* library [27] that provides various functions for data augmentation in case when data is in the form of time series. Functions that we used are:

- *AddNoise* – this function adds random noise to the time series, where the noise added to every time point of time series is independent and identically distributed (Figure 8).

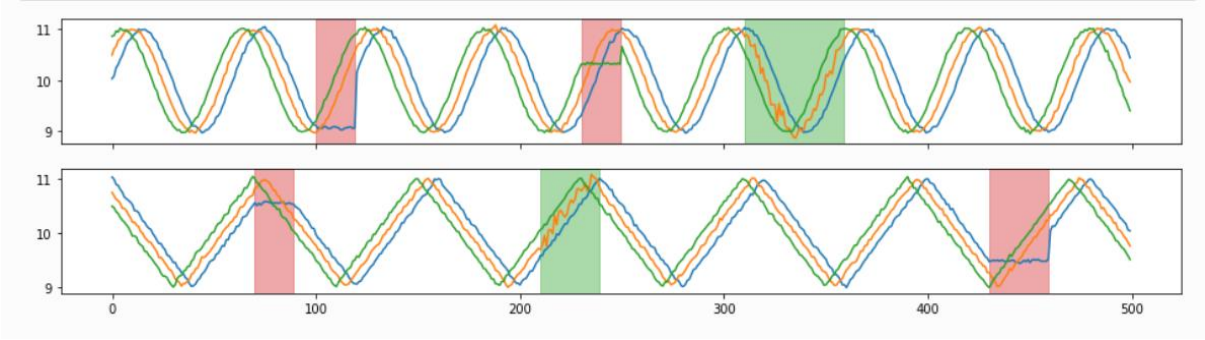


Figure 8: Example of signal before and after applying AddNoise

- *Dropout* – drops values of random time points in time series, where wither single time points or sub-sequences could be dropped out (Figure 9).

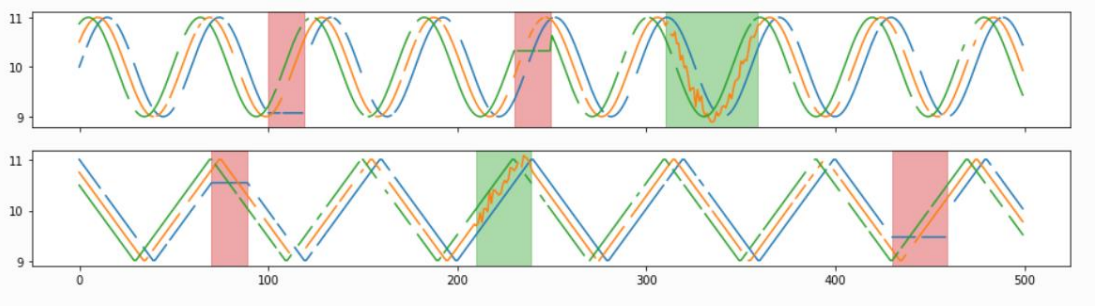


Figure 9: Example of signal before and after applying Dropout function

- *Quantize* – this function is quantizing time series to a level set, where values in a time series are rounded to the nearest level in the level set (Figure 10).

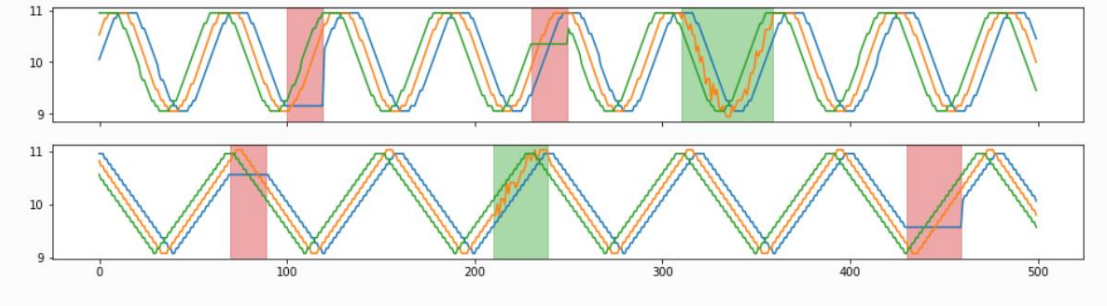


Figure 10: Example of signal before and after applying Quantize function

- *TimeWarp* – randomly changes the speed of the timeline (Figure 11). It is controlled by the number of speed changes and the maximum ration of maximum and minimum speed.

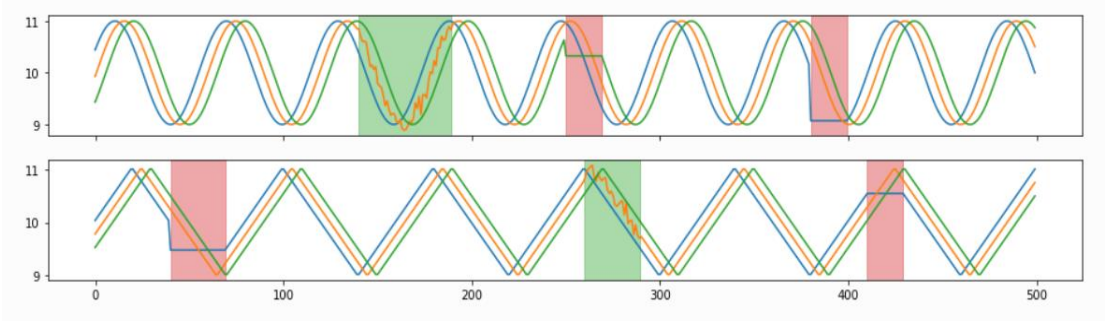


Figure 11: Example of signal before and after applying TimeWarp function

Also, we manually implemented two transformations, *MoveSig* which randomly selects the signal axis (x, y or z) and randomly changes the signal bias from a predefined range of values and *ScaleSig* that scales each signal axis with a given probability for a predefined range of values.

3.5. Performance metrics and validation procedure

In order to see capability of our model to accurately classify steer activity, the following performance metrics are used: accuracy, precision, recall and F1 score. One way to calculate these metrics is the confusion matrix, which gives us a comparison between actual and predicted values. Confusion matrix for binary classification problems can be seen in Figure 12.

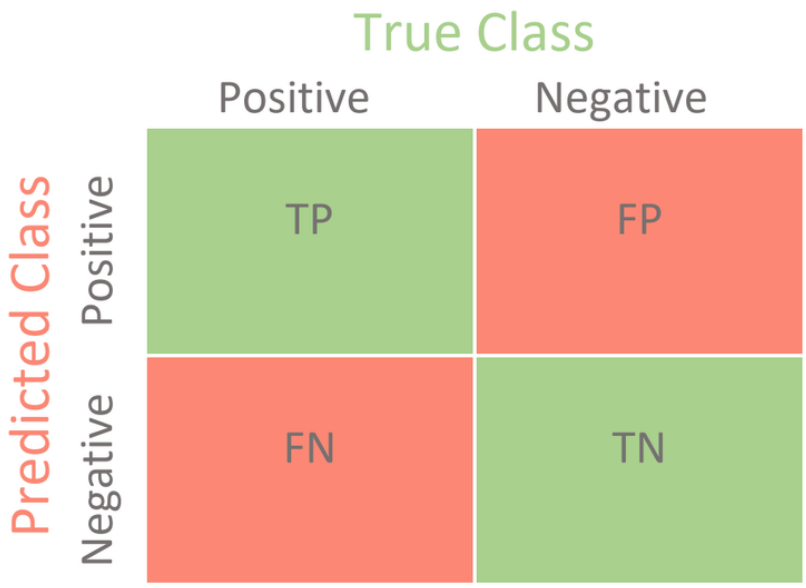


Figure 12: Confusion matrix for binary classification problem

There are four important terms to be considered linked to confusion matrices (Figure 12):

- True Positives (TP) – the label belongs to the class and it is correctly predicted
- False Positive (FP) – the label does not belong to the class but classifier predicted as positive
- True Negative (TN) – the label does not belong to the class and it is correctly predicted
- False Negative (FN) – the label does not belong to the class but is predicted as negative

Note that for the 3-class classification problem, we will have 3x3 confusion matrix, for 5-class classification problem we will have 5x5 confusion matrix etc.

Based on the concepts that we have just introduced, the metrics can be calculated in the following way:

- Accuracy: Describes how well the model predicts and how well it excludes class. It is calculated as the ratio between the number of correctly predicted samples and the total number of predictions.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

- Precision (also referred to as positive predicted value): Measures how well models classify samples as positive.

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

- Recall: Measures the fraction of positive predictions that are correctly classified.

$$Precision = \frac{TP}{TP+TN} \quad (5)$$

- F1 score: This metric represents the harmonic mean between recall and precision. As precision grows, recall usually declines and vice versa, so F1 score actually represent the balance between these two metrics.

$$F1\ score = \frac{2 \cdot Precision \cdot Recall}{Precision+Recall} \quad (6)$$

Note that given definitions for precision, recall and F1 score are for the case of binary classification and therefore, in the case of multiclass classification some kind of mean needs to be applied. We used *weighted-average* where scores are calculated by taking the mean of all per-class scores while considering weight of every class i.e. the number of actual occurrences of the class in the dataset.

For model validation purposes, we performed cross-validation. Cross-validation provides us the ability to estimate performances of the model on unseen data.

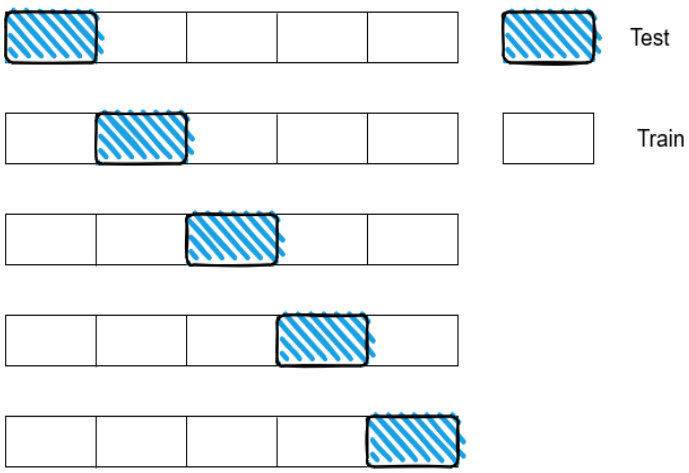


Figure 13: k-fold cross-validation

To perform cross-validation, we first need to split data into subsets called folds which need to be of similar sizes. Next step is to choose one fold for the holdout set which will be used as a test set, and use all other folds for training of the model (Figure 13). If *k* is the number of folds on which our data is divided, we will train our model *k* times.

3.6. Multiclass cattle behavior recognition

As already mentioned, we had information about 18 steers, where all cows are numbered from 01 to 18. For the purposes of model performance evaluation, three steers, 04, 10 and 11, each from distinct trial, were selected randomly for the test set as it was done in [5], while remaining 15 steer were selected to form a training and validation set. The same test set was chosen as in [5] in order for us to be able to compare performances of our models with those in [5].

In the previous research, multiclass CNN model was developed while only collar and halter data were used. They preprocessed data the same way we described in the previous section. The classifier includes two blocks, feature extractor and head (Figure 14). The first block consists of four blocks of convolutional *Dropout*, *BatchNorm* and *ReLU* layers followed by *Adaptive Average Pooling* layer. The second block i.e. head consists of a single fully connected layer followed by *Softmax* layer which generates the classified output [5].

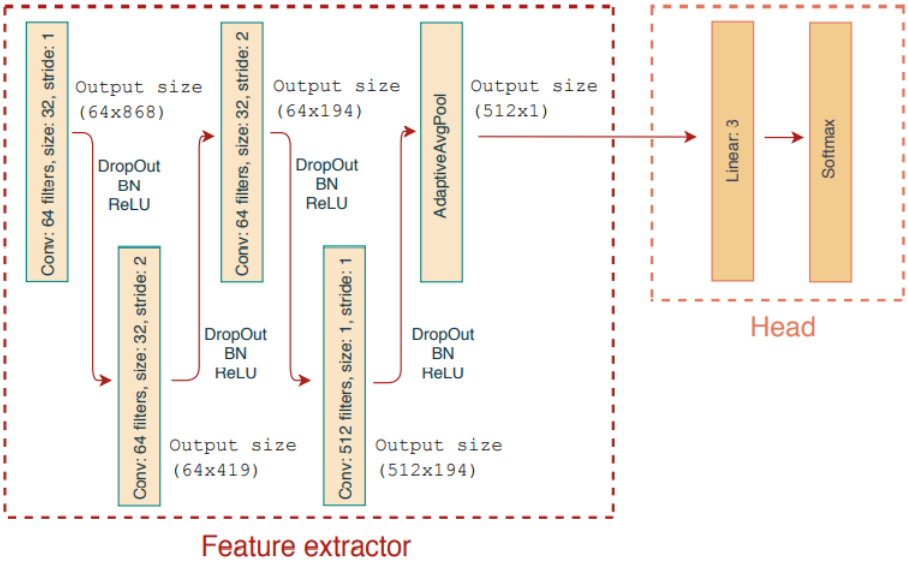


Figure 14: CNN architecture; Source [5]

Further, data was divided into 90 seconds blocks, as mentioned in section 3.2 and model was trained with *AdamW* optimizer with learning rate of 1×10^{-4} using one-cycle

training policy and cosine learning rate annealing. The optimal learning rate was found with Fastai's built in function where interval was set to be from 10^{-10} to 10^{-1}. Maximum number of epochs was set to 50 but early stopping was used with patience of 15 epochs. Moreover, delta change was set to 0.01 while batch size was set to 256 and weight decay of 0.01 was used. After parameter tuning, kernel size was set to 16 and *Dropout* probability to 0.25 [5].

In addition, they tried different lengths of the classification window, and analyzed its effect on the classification performance. This is crucial for classification of animal behavior because certain animal activities occur for a few seconds while some others typically last for several minutes. Models were trained on 60s, 90s and 120s window lengths, where the first two gave almost identical results, while 120s windows gave lower performance. Therefore, the 90s window length was chosen to match the specifications of the commercial collar system [5]. Their best model had in total 171.563 parameters and an F1 score of 82.1%.

3.7. Model training

All models were trained using Python programming language, using both Jupyter Notebook [28] software and Google Colaboratory [29]. Models were created using PyTorch [30] and Fastai [31].

As mentioned before, we used data collected with a halter and collar device like in [5]. Naturally, individual steers do not spend the same amount of time performing different activities. Therefore, data was stratified in order to obtain a balanced dataset for analysis. For every steer, the activity with the least amount of appearances was chosen and then a random subset of data points was selected for all other activities. However, dataset stratification was used only for CNN models. For LSTM models, time dependences of data are important for decision making, and balancing would shuffle samples.

We did balancing between cows in order to obtain the same number of 90 seconds blocks of signals per cow. This way, every steer would have the same impact on decision making while achieving better generalization of the model.

In order to try to expand the possibility that offers a classification of only three steer activities, we also utilized data about 4 cows collected with pedometers and using transfer learning offered classification of five activities: eating, rumination, other, standing and lying. The sample distribution is highlighted in Figure 15.

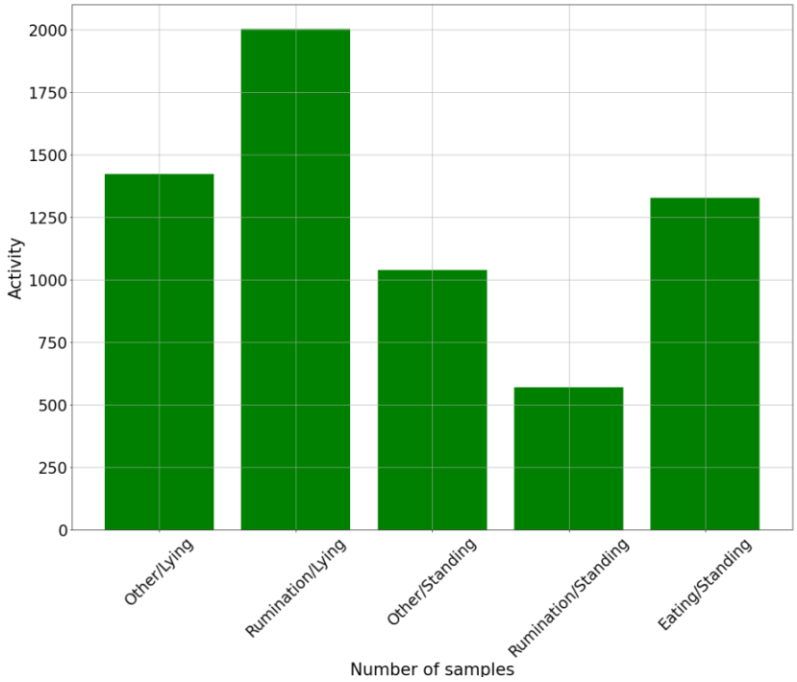


Figure 15: Histogram of sample distribution

Neural networks usually require a great amount of data for training which is not always available. By using transfer learning, the model has already been pre-trained so a good model can be generated even with a smaller amount of data. Also, some neural networks take days or even weeks to train, and by using transfer learning training time is significantly reduced.

In our case, we pre-trained the model on data collected from halter and collar, which contains three classes and used those models with new data from movement sensors which were collected for steers with id's 07, 08, 09 and 10. In [5], cow with id 10 was used for testing, and the first three were used for training. Moreover, cow 10 was used for testing and not for training in order to avoid data leakage. If we had used some other cow that was already used for training also for testing, our model would have been able to use knowledge gained from that steer and we would not have a realistic evaluation of the model performance. Instead, we used all pre-trained models on those folds which contain one of cows 07, 08 or 09 for training of the new model. However, this is no longer multiclass but multiclass/multilabel classification problem which means that we do not have mutually exclusive classes as before and therefore our classification problem is more complex. Further, note that all cows eat in a specific place on the farm from feeders which physically enables them to eat while lying. Consequently, the combination of behaviors lying and eating does not exist in dataset.

4. Results and Discussion

4.1. Multiclass/multilabel model

Architectures of multiclass and multiclass/multilabel models are highlighted in Figure 16. In multiclass/multilabel model we first have four convolutional layers followed with *Dropout*, *BatchNorm* and *ReLU* layers, which represent our feature extractor. After last convolutional block, we have *AdaptiveAveragePool*, which is followed with *Linear* layer that gives us output value i.e. tells us to which of five classes our sample belongs to.

As mentioned before, multiclass model was trained to classify three different classes and it had 171 563 trainable parameters. Even though our multiclass/multilabel model had similar number of total parameters as multiclass one, it was trained to classify 5 classes and number of trainable parameters was only 3 973. Note that trainable layers are present with green and non-trainable with blue color on Figure 16. Consequently, as we have a less number of trainable parameters, it takes less time to train the model, in our case training time is shorter for 35%.

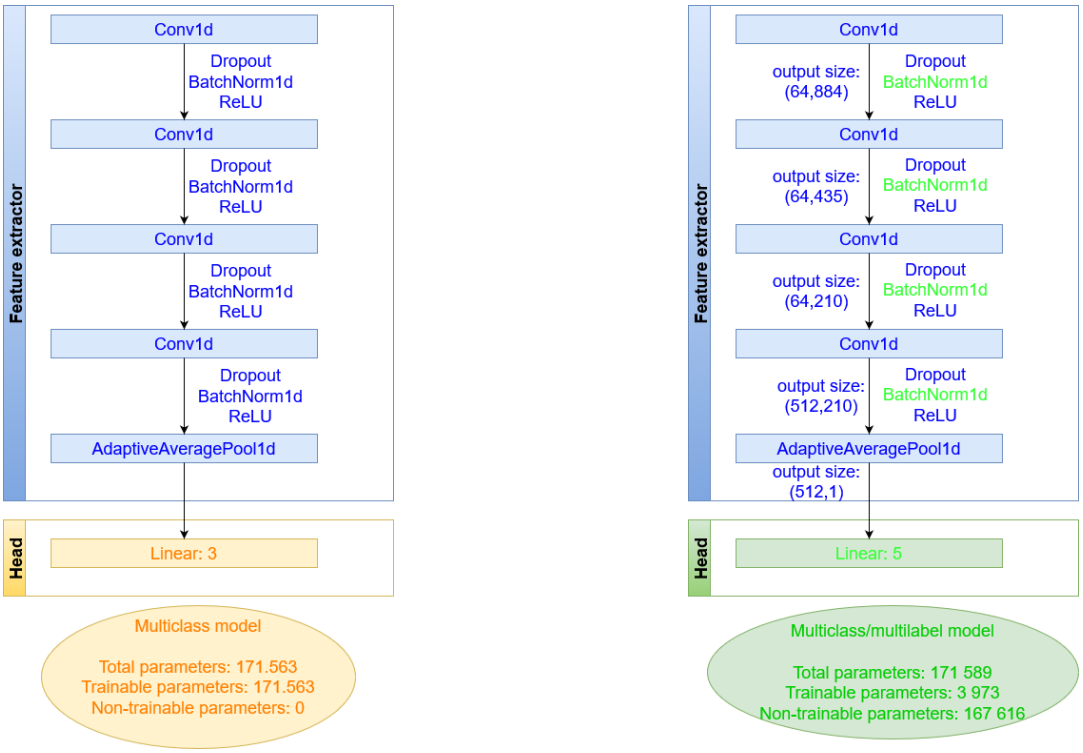


Figure 16: Architecture of multiclass CNN model from [5] (left) and multiclass/multilabel CNN model (right)

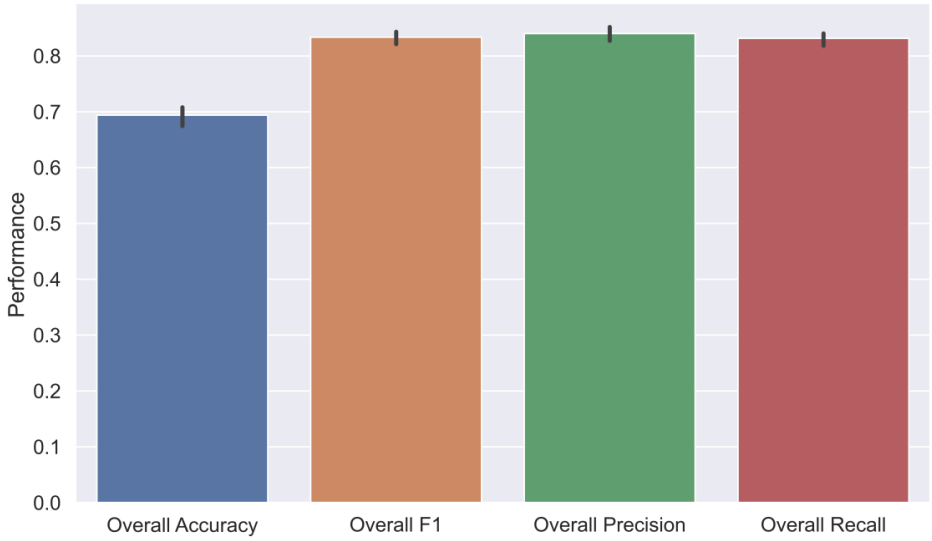


Figure 17: Overall performance of transfer learning model

On Table 2, the overall performance metrics for every of 15 iterations are shown. We can see the overall performance of the transfer learning model on Figure 17. Overall F1 score is 0.82 ± 0.028, overall precision is 0.83 ± 0.02 and overall recall of the model is 0.82 ± 0.018 while overall accuracy is 0.68 ± 0.028.

Moreover, average overall accuracy is 67.59%, overall F1 score is 82.27%, average precision score is 83.26% while average overall recall is 82.05% (Table 2).

Table 2: Overall model performance

	Overall Accuracy	Overall F1	Overall Precision	Overall Recall
Cow 1	69.18%	83.49%	84.69%	83.12%
Cow 2	65.22%	80.76%	81.49%	80.59%
Cow 3	69.57%	83.31%	84.36%	83.28%
Average score	67.59%	82.27%	83.26%	82.05%

As this problem is more demanding, it was to be expected that the overall performance metrics of this model would be lower than for the multiclass problem. For that reason, we additionally wanted to compare model performances by computing individual performances by class.

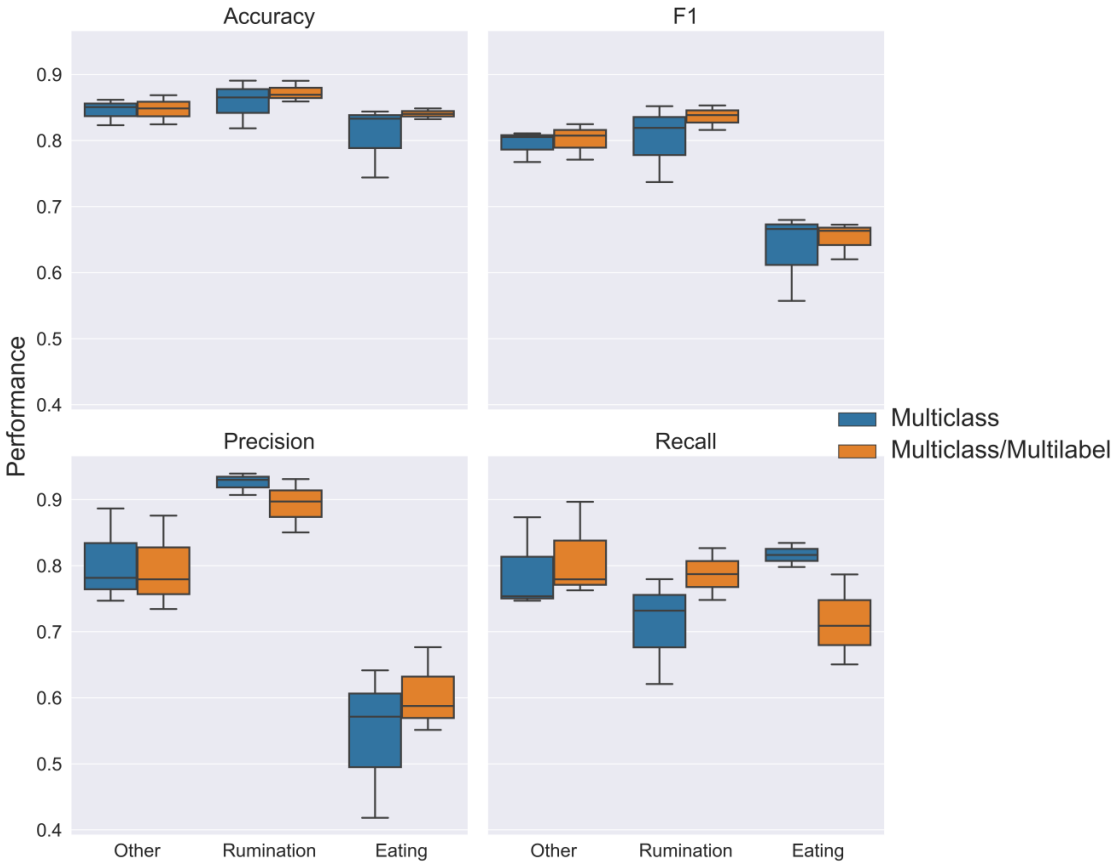


Figure 18: Overall performance comparison; original model and transfer learning model

In Figure 18, we can see the statistical distributions of accuracy, F1 score, precision and recall for every of three classes for both, original CNN (multiclass) model and transfer learning (multiclass/multilabel i.e. multilabel) model per class.

Distribution of accuracy and F1 score is less dispersed for multiclass/multilabel than for multiclass model. In other words, there are fewer oscillations in model performance for our transfer learning model for every class.

Further, if we look at precision, we can see that for class 'other', distributions are very similar. For the 'rumination' we have more oscillations in multilabel than multiclass problem performance, while variation in precision metric is smaller for 'eating' in multilabel model.

In case of recall, we again have very similar distributions for class 'other'. For the 'rumination' there are fever oscillations in multilabel model performance, while dispersion of recall for 'eating' is smaller in multiclass models.

In addition, we evaluated not only overall performance but also the ability of the model to correctly classify each steer activity separately.

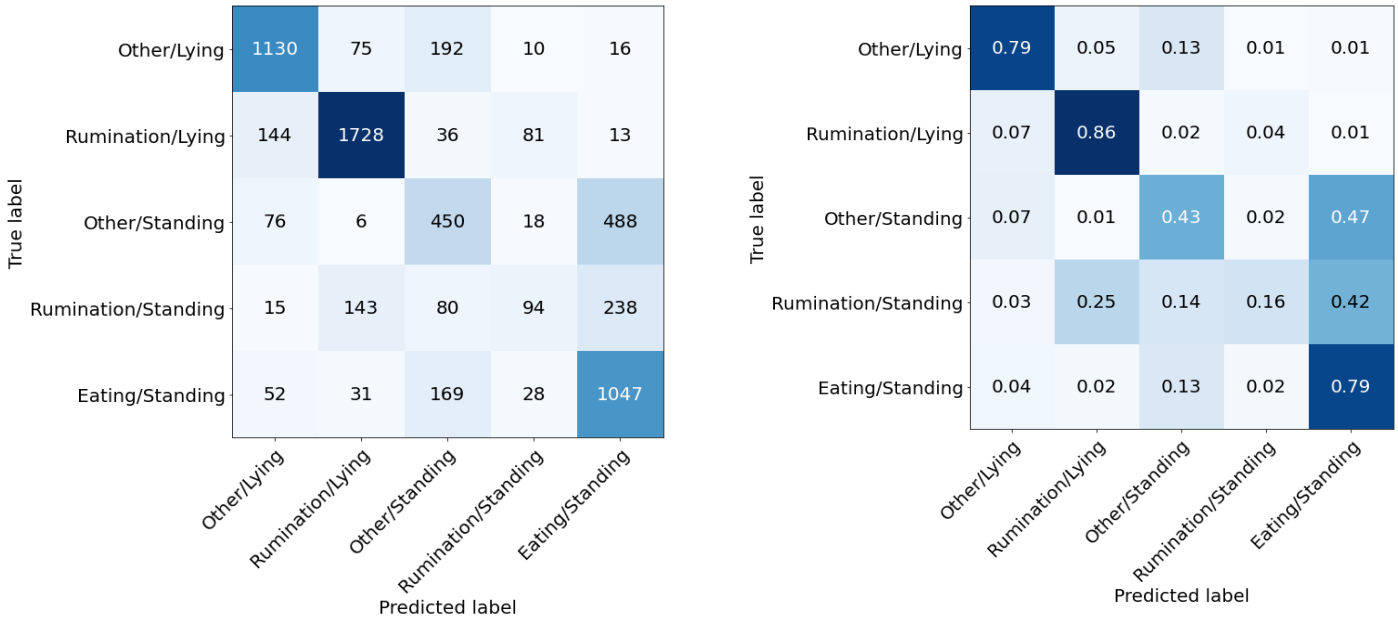


Figure 19: Confusion matrix corresponding to the best multiclass/multilabel model score (left) and its relative representation (right)

Main diagonal of confusion matrices, both highlighted in Figure 19 indicates that the model very well classifies most of the steer activities. We can see that the model best predicts rumination/lying with 86% and other/lying with 79% accuracy. However, the model does not predict very well the rumination/standing combination of activities, but this is because cows almost never ruminate while standing (Figure 10). For that reason we have a small number of samples for rumination/standing and as a consequence classification model is not able to extract enough knowledge to discriminate this particular activity from eating/standing. It misclassified rumination/standing as eating/standing with 42%. As those activities are similar in motion it is not unlikely that the classification model was not able to discriminate between them. However, a higher number of samples for rumination/standing could provide more useful information and thus possibility for DL model to increase its performance for that particular activity. For that reason, more samples from minor classes would be provided in future work.

Similarly, the model had successfully classified just 43% of samples as a combination of other/standing behavior. This can also be a consequence of less number of samples corresponding to this combination of steer activities on which model could learn to distinguish them from other combinations.

Table 3: Multiclass/multilabel model overall performance on validation and test data

	Overall Accuracy	Overall F1	Overall Precision	Overall Recall
Validation set	67.59%	82.27%	83.26%	82.05%
Test set	72.96%	85.96%	87.73%	86.05%

In Table 3, we can see that overall metrics that illustrate performance of the model on unseen data are similar or even better than those on validation data. Consequently, we can conclude that our model does not overfit. Further, we can see that model classifies unseen samples very well, as highlighted in Figure 20.

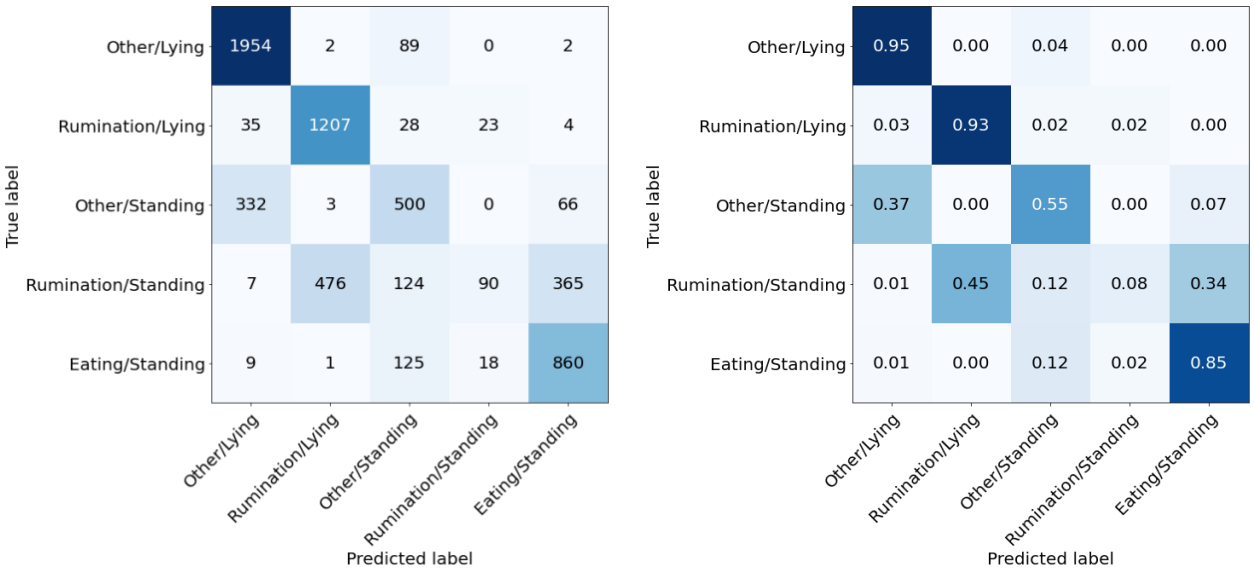


Figure 20: Confusion matrices illustrating performance of the model on test set; score (left) and its relative representation (right)

The model best predicts class other/lying with 95% and rumination/lying with 93% accuracy, which is higher than prediction accuracy on validation dataset for 16% and 7% respectively (Figure 19). Moreover, this model correctly classified 85% of samples to belong to class eating/standing, while only 8% of samples were correctly classified to belong to the least present class other/standing.

4.2. Overlaps

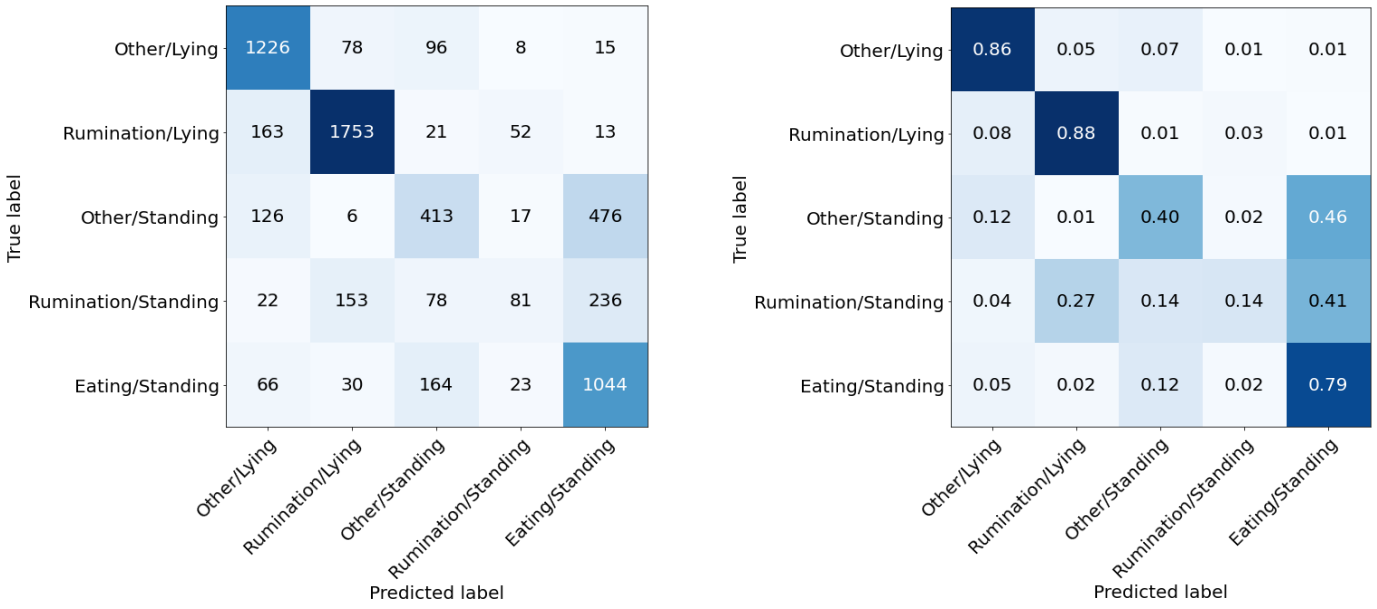
Performances of multiclass/multilabel models trained on datasets with overlapping signals are similar to the performance of that model trained on dataset without overlaps.

In Table 4, we can see that a model trained on datasets with 75% overlaps has slightly better performance. Moreover, by training the model on a dataset with overlaps of 75%, model accuracy increases for almost 1%, while F1 score, precision and recall increases for approximately 0.5%. However, this is not significant model performance improvement.

Table 4: Comparison of model performances with and without overlapping signals

	Accuracy	F1 Score	Precision	Recall
No overlaps	67.59%	82.27%	83.26%	82.05%
25% overlaps	67.04%	81.78%	82.97%	81.67%
50% overlaps	62.13%	78.03%	79.58%	78.60%
75% overlaps	68.58%	82.79%	83.61%	82.60%

Further, we can see that a model trained on a dataset with 25% overlaps has worse performance than one trained on the original dataset. Moreover, performances for all metrics are smaller by 0.5% than for the original model. Finally, model trained on dataset with 50% overlaps has the worst performances. Accuracy of this model is smaller for 5%, while F1 score, precision and recall all smaller for around 4% comparing to the original multiclass/multilabel model.



a) b)

Figure 21: Confusion matrices: a) 75% overlaps, b) 75% overlaps normalized results

In Figure 21, we can see confusion matrices of the model trained on dataset with 75% overlapping data. Looking at the main diagonal of the confusion matrices, we can see that this model showed similar classification performance as the original one for cattle behavior classification (Figure 19), were 75% overlapping model outperformed original model when classifying Other/Lying by 7% and Ruminating/Lying by 2%.

4.3. Augmentation

In order to evaluate the individual influence on the classification performance, we applied augmentation functions one by one and the corresponding model performance in terms of accuracy, F1 score, precision and recall can be seen in Table 5. Furthermore, the model performance when applying *DropOut* and *ScaleSig*, transformations with the highest influence, is also provided within the table, as well as the model performance when all transformations are provided during model training and applied with certain probability. *Dropout* transformation provided slightly higher performance in comparison to the model trained on the original dataset with improvement of around 0.2%, while the precision of this model is rather worse. Performance of the model trained on dataset obtained by applying *ScaleSig* transformation is also better in sense of accuracy with 0.3% improvement, both F1 score and recall of 0.09%, while precision of this model is worse than for multiclass/multilabel model trained on the original dataset. Nevertheless, the improvement in classification performance is not significant, as the original dataset is already diverse enough, so newly added signals did not make any difference.

Table 5: Performances of models trained on datasets obtained by different data augmentation techniques

	Accuracy	F1 score	Precision	Recall
No augmentation	67.59%	82.27%	83.26%	82.05%
MoveSig	66.27%	81.48%	82.12%	81.34%
Dropout	67.76%	82.34%	83.24%	82.14%
AddNoise	65.73%	80.67%	81.70%	80.44%
Quantize	67.50%	82.16%	82.94%	81.76%
ScaleSig	67.90%	82.36%	83.22%	82.16%
TimeWarp	65.64%	80.98%	81.62%	80.88%
Dropout&ScaleSig	67.71%	82.20%	83.16%	81.99%
All functions	61.35%	77.40%	78.06%	77.38%

4.4. CNN – LSTM

In order to try and use the best from both models, we tried to train a hybrid CNN-LSTM model to predict cattle behaviors. Convolutional Neural Networks are known for their ability to extract important data features, while Recurrent Neural Networks can very successfully deal with spatial dimension of data. Because of this, we build a model which consists of two parts, Convolutional Neural Network followed by Recurrent Neural Network, or more precisely Long-Short Term Memory Network. Architecture of CNN-LSTM model is highlighted in Figure 22.

First, we have four convolutional blocks where every block consists of *Conv1d* layer followed by *Dropout*, *BatchNorm* and *ReLU* layera, as in convolutional multiclass/multilabel model. These blocks represent our feature extractor. After this, we have *LSTM* block of 4 layers each consisted of 50 neurons, whose output is then forwarded to *Linear* layer that gives us our prediction i.e. classifies sample in one of five classes.

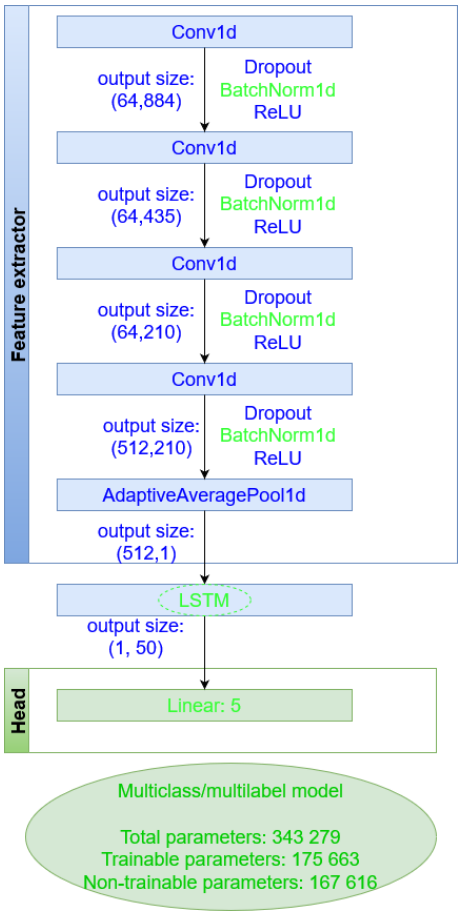


Figure 22: Architecture of CNN-LSTM multiclass/multilabel model

Total number of parameters is 343 279, which is much more than the total number of parameters for the original multiclass/multilabel model. The number of trainable parameters of CNN-LSTM model is 175 663 which is even more than total number of parameters of original multiclass/multilabel model and training time of this model is longer for 50% than for multiclass/multilabel one.

The performances of CNN-LSTM model are highlighted in Table 6. This model performance is lower than the performance of the original multiclass/multilabel problem probably because multiclass/multilabel model has already learned everything from the available dataset, and LSTM layer was not able to contribute and improve classification power.

Table 6: Performance of CNN-LSTM multiclass/multilabel model

	Accuracy	F1 score	Precision	Recall
CNN	67.59%	82.27%	83.26%	82.05%
CNN-LSTM	66.85%	81.35%	82.34%	81.33%

5. Conclusion

In this thesis, we used the accelerometer data from neck-mounted collar obtained from four animals in order to classify five different cattle activity states. The main contribution was the extension of the possibilities of the ML model proposed in [5] in terms of the number of monitored activities, where data was classified in three cattle activities (rumination, eating and other) with ground truth data provided by muzzle-mounted halter. Here, ground truth data obtained from pedometer, placed on rear leg of the animal, was additionally used in order to add the information when the animal was standing or lying. As the classes obtained from the halter and those obtained from the pedometer are not mutually exclusive the classification problem became more complex and expanded from multiclass into multiclass/multilabel classification task. For the classification purpose we used pre-trained CNN models for identification of three cattle activity states and through transfer learning procedure managed to classify five cattle activities while not increasing model complexity and time required for model training. The proposed solution can classify target behaviors with an overall accuracy, F1 score, recall and precision of 72.96%, 85.96%, 87.73%, and 86.05%, respectively. Individual behaviors are classified with an average F1 score of 90.41%, 91.11%, 93.66%, 84.30% and 84.30%, for rumination, eating, other, standing and lying. Considering the performance, it can be noted that the proposed solution not only expand the number of classified activity states, but also outperformed the multiclass solution for rumination and eating for 9% and 4% respectively, without significant increase in model complexity. Additionally, we have tried to improve results with data augmentation techniques, overlapping data samples and hybrid CNN-LSTM neural networks. Model trained on data with overlaps of 75% between subsequent samples achieved the highest classification performance with an overall F1 score of 82.79%, while *Dropout* and *ScaleSig* transformations shown a slight improvement in performance and should be further investigated. In the future, one should consider monitoring more cows and generate a bigger dataset which contains more samples of those classes that are the least present in our dataset. Also, one could try to train a model on a dataset with both overlaps and augmentation. Further, one could constrain the number of different augmentation transformations that can be applied on the same signal. Finally, considered behaviors should be linked with physical illnesses in order to bring research closer to practical application.

Bibliography

[1] U. FAO, *How to feed the world in 2050* in Rome: High-Level Expert Forum, 2009.

[2] Baweja P., Kumar S., Kumar G., *Fertilizers and Pesticides: Their Impact on Soil Health and Environment*, Soil Health, 2020.

[3] DTN Team, *How Is Machine Learning used in Agriculture?*, Agriculture, 2021.

[4] Thekkoot, D., *Big Data and Machine Learning in Animal Breeding*, Genesus, 2020.

[5] Pavlovic, D., Davidson H., Hamilton, A., Marko, O., Atkinson, R., Davison, C., Michie, C., Crnojevic, V., & Andonovic, I., Bellekens X., Tachtatzis, C. *Classification of cattle behavior using Neck-Mounted Accelerometer-Equipped Collars and Convolutional Neural Networks*. sensors, 2021.

[6] Thorup V. M., Munksgaard L., Robert P.-E., Erhard H. W., Thomsen P. T., Friggens N.C., *Lameness Detection via leg-mounted accelerometers on dairy cows on four commercial farms,* Animal, 2015

[7] Automatig Dairy Farms. Available online: https://afimilk.com

[8] AHDB Dairy. AHDB Dairy Statistics. Available online: https://ahdb.org.uk/dairy

[9] Rahman, A., Smith, D.V., Little, B., Ingham, A.B., Greenwood, P.L., Bishop-Hurley, G.J. *Cattle behavior classification from collar, halter, and ear tag sensors*, Inf. Process. Agric., 2018.

[10] González, L.A., Bishop-Hurley, G.J., Handcock, R.N., Crossman, C. *Behavioral classification of data from collars containing motion sensors in grazing cattle,*. Comput. Electron. Agric., 2015.

[11] Hamilton, A.W., Davison, C., Tachtatzis, C., Andonovic, I., Michie, C., Ferguson, H.J., Somerville, L., Jonsson, N.N. *Identification of the rumination in cattle using support vector machines with motion-sensitive bolus sensors*, Sensors, 2019

[12] Pradhan A., *Support Vector Machine-A Survey*, International Journal of Emerging Technology and Advanced Engineering, 2012

[13] Jijo B. T., Abdulazeez A. M., *Classification Based on Decision Tree Algorithm for Machine Learning*, Journal of Applied Science and Technology Trends, 2021

[14] Biau G., Scornet E., *A random forest guided tour*, TEST, 2016

[15] Benaissa, S., Tuyttens, F.A., Plets, D.; Cattrysse, H., Martens, L., Vandaele, L., Joseph, W., Sonck, B. *Classification of ingestive related cow behaviors using RumiWatch halter and neck-mounted accelerometers*, Appl. Anim. Behav. Sci., 2019

[16] Benaissa, S., Tuyttens, F.A.M., Plets, D., de Pessemier, T., Trogh, J., Tanghe, E., Martens, L., Vandaele, L., Van Nuffel, A., Joseph, W., *On the use of on-cow accelerometers for the classification of behaviors in dairy barns*, Res. Vet. Sci. 2019

[17] Zhang S., Cheng D., Deng Z., Zong M., Deng X., *A novel knn algorithm with data-driven k parameter computation*, PDMSD, 2018

[18] Chen S., Webb G. I., Liu L., Ma X., *A novel selective naïve Bayes algorithm*, Knowledge-Based System, 2020

[19] Kasfi, K.T., Hellicar, A., Rahman, A. *Convolutional Neural Network for Time Series Cattle Behavior Classification*, In Proceedings of the Workshop on Time Series Analytics and Applications—TSAA '16, Hobart, Tasmania, 5 December 2016; ACM Press: New York, NY, USA, 2016

[20] Peng, Y., Kondo, N., Fujiura, T., Suzuki, T., Wulandari., Yoshioka, H., Itoyama, E., *Classification of multiple cattle behavior patterns using a recurrent neural network with long short-term memory and inertial measurement units*, Comput. Electron. Agric. 2019

[21] Weiss K., Khoshogoftaar T.M., Wang D., *A survey of transfer learning*, Journal of Big Data, 2016

[22] Yen I. E. H., Huang X., Zhong K., Ravikumar P., Dhillon I. S., PD-Sparse: A Primal and Dual Sparse Approach to Extreme Multiclacc and Multilabel Classification, 2015

[23] Afimilk/NMR. *Silent Herdsman/Better performing cows*., 2012.

[24] ITIN+HOCH. RumiWatchSystem: *Measurement system for automatic health monitoring in ruminants*, 2014.

[25] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[26] Um T. T., Pfister F., Pitchler D., Endo S., Lang M., Hirche S, Fietzek U., Kulic D., *Data Augmentasion of Wearable Sensors Data for Parkinson's Disease Monitoring using Convolutional Neural Networks*, ICMI, 2017

[27] Tsaug library: https://tsaug.readthedocs.io/en/stable/references.html

[28] Jupyter Notebook: https://jupyter.org/

[29] Google Colaboratory: https://colab.research.google.com

[30] PyTorch: https://pytorch.org/

[31] Fastai: https://www.fast.ai/

[32] Yin X., Wu Dihua, Shang Y., Jiang B., Song H., *Using an EfficientNet-LSTM for recognition of single Cow's motion behaviors in a complicated environment*, ScienceDirect, 2020

[33] da Santos A.S., de Medeiros W.C., Goncalves G.E., *Monitoring and classification of cattle behavior: a survey*, ScienceDirect, 2022

[34] Smith L.N., *A Disciplined Approach to Neural Network Hyper-Parameters: Part 1 – Learning Rate, Batch Size, Momentum and Weight Decay*, US Naval Research Laboratory Technical Report, 2018

[35] O'Driscoll K., Boyle L., Hanlon A., *A brief note on the validation of a system for recording lying behavior in dairy cows*, ScienceDirect, 2007

[36] Blackie N., Bleach E., Amory J., Scaife J., *Impact of lameness on gait characteristics and lying behavior of zero gazed dairy cattle in early lactation*, SciecneDirect, 2011

[37] Chapinal N., de Passille A. M., Pasrell M., Hännigen L., Munksgaard L., Rushen J., *Measurement of acceleration while walking as an automated method for gain assessment in dairy cattle*, American Dairy Science Association, 2011

[38] Alsaaod M., Römer C., Kleinmanns J., Hendriksen K., Rose-Meierhöfer S., Plümer L., Büscher W., *Electronic detection of lameness in dairy cows through measuring pedometric activity and lying behavior*, ScienceDirect, 2012

[39] Taneja M., Byabazaire J., Jalodia N., Davy A., Olariu C., Malone P., *Machine learning based fog computing assisted data-driven approach for early lameness detection in dairy cows*, ScienceDirect, 2020.

[40] Yunta C., Guasch I., Bach A., *Short communication: Lying behavior of lactating dairy cows is influenced by lameness especially around feeding time*, American Dairy Science Association, 2012

Biography

Katarina Petranović was born on 5^{th} of May 1998 in Novi Sad, Serbia. She attended elementary school "Svetozar Markovic Toza" and grammar school "Jovan Jovanović Zmaj" in Novi Sad. In 2017 she started her bachelor degree studies in Financial Mathematics at Faculty of Natural Sciences, University of Novi Sad, which she finished in September, 2020 with GPA 9.15. The same year, she continued her education at the same university, enrolling in a master degree program in Applied Mathematics – Data Science. She finished the master program with a GPA of 9.56.