Unsupervised Representation Learning for Animal Activity Recognition

Ren, Chenhao

Award date:
2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
System Architecture and Networking Group

Unsupervised Representation Learning for Animal Activity Recognition

Master Thesis

Chenhao Ren

Supervisors:
Prof.Dr.Ir. Nirvana Meratnia
Dr. Tanir Ozcelebi

Eindhoven, August 2021

Abstract

Animal activities provides rich information for human beings to study their behaviours, their environments and possibly, recognise their abnormal pathological behaviour. Utilizing unsupervised representation learning as a feature extraction method for animal activity recognition avoids the feature engineering which relies on domain experts and the need of large amount of labeled data. Obtaining labeled data is tedious, costly and labor-intensive while obtaining raw unlabeled data is much easier.

In this thesis, we investigate how unsupervised representation learning using variational autoencoder (VAE) cope with lack of labeled data in animal activity recognition. Furthermore, we check into the effect of three types of VAE models, two types of classifiers, different amount of labeled training data. We also inquire into the transferability of our model, the visualization on how our model learns, the performance improvement by unfreezing the neural network, the robustness of the model, the performance improvement of our VAE model and the time consumption of network training. We evaluate the performance by F1 score and compare our baseline with the state-of-art.

We demonstrate: our model performs better than the state-of-art with more robustness; Softmax classifier performs better than Support Vector Machine classifier; the latent space visualization let us be able to check how good the VAE learns; With the amount of labeled training data increasing, the performance increases before reaching top; our model has good transferability for another animal; by unfreezing the neural network layer, the performance can be improved while the training time is increased; the performance of the same neural network architecture with more trainable parameters has more fluctuation than that with less trainable parameters; unsupervised representation learning helps increase the performance especially with low amount of labeled training data; LSTM-VAE training costs the longest time while CNN-VAE costs the shortest and SVM takes more training time than Softmax.

Acknowledgements

I would like to thank my supervisors, Prof.Dr.Ir. Nirvana Meratnia and Dr. Tanir Ozcelebi for the guidance, motivations and feedback to me during my work on this thesis. Also for Dr. George Exarchakos for being the committee member. Furthermore, I would like to thank my family for the financial and mental support. Finally, I would like to thank my friends for the mental support especially during the hard COVID-19 pandemic period.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Monitoring animal activities provides rich information, about them ranging from their daily routines, well-being and interaction with their environment [5, 6]. It is crucial for human beings to monitor animal activities to study their behaviours, their environments and possibly, recognise their abnormal pathological behaviour. As a result, Animal Activity Recognition (AAR) has received great attention as a relatively new research area [2]. Thanks to the development of small, light-weight and low-power sensors, it is possible to monitor animal activities by attaching sensors to them without disturbing their daily lives. These sensors are able to collect a variety of data such as image, temperature, humidity, location, motion and so on [7, 8].

Human beings can track animal activities using inertial motion sensors such as accelerometers, gyroscopes and magnetometers [3, 9, 10]. Typically, there are four steps to do the animal activity recognition: data collection, data preprocessing, feature learning or extraction, and activity recognition [11].To recognize the animal activity, the collected data needs to be mapped into the actual activity space performed by the animals. This is done through a classification task. As for the classification, feature extraction is vital. Feature is defined as an individual measurable property or characteristic of a phenomenon [12]. The features may be defined manually through feature engineering in the classical machine learning era, where the features are selected by experts using domain knowledge. The work of feature engineering is vast, and the quality of feature selection will directly affect the quality of data representation. It would be desirable to select leading features with stability and interpretability from a set of distinct features for a comprehensive data description [13]. Therefore, the feature selection will impact the performance of downstream tasks such as classification. However, in the deep learning era, the features can be extracted automatically, the process also called representation learning. The hidden data representation can be learned automatically by AI models, which does not depend on the domain experts experience. At the same time, it needs extensive training data set because instead of manually feature extraction, the model needs to learn data representation and more training data would help the model in better representation learning. Learning representations of the data makes it easier to extract useful information while building classifiers or other predictors [14].

The classification performance (accuracy or F1 score) mainly depends on two factors; one is the quality of features extracted or learned, and the other is the type and complexity of the classifier. Various statistics on the time and frequency domains could be used in terms of the handcrafted extracted features, and the Naive Bayes, Support Vector Machine, K-Nearest neighbours, Decision Trees, etc. could be used for activity recognition [3, 9]. However, a significant disadvantage of the handcrafted feature is the labour required from the domain experts, and it cannot capture the complex features. This drawback can be overcome by employing unsupervised representation learning techniques such as the Variational Autoencoder (VAE), which can automatically learn complex features [15]. Both pre-defined and learned features need labelled data to link the features to the actual situation represented by data. However, obtaining the labelled data is tedious and expensive, it costs time, labour, and even sometimes, cannot be done due to the harsh or impractical environmental conditions, for example, monitoring the wild animals or birds. At

the same time, it is much easier to obtain raw unlabelled data. In this context, unsupervised representation learning would help increase classification accuracy as plenty of unlabeled data could be used for it. As a result, how to cope with the lack of labeled data in the context of Animal Activity Recognition becomes a problem.

In some scenarios, in which tasks or subjects are similar, transfer learning may be used to save the training resource usage and improve generalization in another setting. Transfer learning is an optimization that allows rapid progress or improved performance when modeling a second related task [14]. As a result, it can be expected that the model trained with dataset of particular set of subjects should be transferable to other similar subjects performing similar tasks.

There are only a few recent papers that focused on the topic of representation learning in terms of AAR using time series inertial measurement unit (IMU) data [2, 10, 16]. The effect of VAE using time series data has been researched in multiple contexts, such as Human Activity Recognition (HAR) and Anomaly Detection (AD), and it has obtained competitive performance (F1 score) compared to other methods [17, 18]. However, to the best of our knowledge, it has not been researched for AAR so far. Furthermore, it has been shown that different classifiers would result in different performance [19], and the effectiveness of transfer learning has been investigated in HAR but not in AAR [20, 21]. It is therefore still interesting to investigate the transferability of the unsupervised representation learning for different animals.

1.1 Research Questions

The main research question to be addressed in this thesis is:

To what extent can unsupervised representation learning using variational auto-encoder (VAE) cope with lack of labeled data in AAR?

To answer the main research question, we formulate the following sub-research questions:

- RQ-1: To what extent VAE improves the performance (F1 score) of AAR?
- RQ-2: What are the impacts of different classifiers (Support Vector Machine (SVM), Softmax) on the performance?
- RQ-3: What is the impact of the amount of labeled data on the performance?
- RQ-4: How transferable is the unsupervised representation learning (VAE) for another animal?

1.2 Contributions

We summarized the main contribution of this thesis as follows:

- We proposed and implemented three types of unsupervised representation learning VAE models to cope with lack of labeled data in animal activity recognition and compared the results with the state-of-art, LSTM-VAE model is the best performed model and it outperforms the state-of-art, meanwhile, it is more robust compared with the state-of-art.
- We implemented the latent space visualization using t-SNE to check how good the unsupervised representation learning has learned and the quality of clustering in the visualization is consistent with the performance of downstream classification performance.
- We discovered how transferable is the unsupervised representation learning using VAE for another animal and our model trained with horse movement data has good performance on goat movement data.

- We looked into how unsupervised representation learning improves the performance compared with the supervised learning from scratch with lack of labeled data and it was found that the performance of supervised learning from scratch is less than that of unsupervised representation learning and with the amount of labeled data increasing, the performance difference becomes less and less.
- We measured the training time of our neural network and we found that LSTM-VAE takes the most amount of time while CNN-VAE takes the least amount of time; By unfreezing the neural network (the parameters of the whole neural network can be updated), the training time of LSTM classification neural network increases dramatically while for CNN classification neural network, the training time slightly increases; SVM classifier takes more training time than Softmax classifier.

1.3 Thesis Outline

The rest of thesis is structured as follows: Chapter 2 provides background information and related work about the topic of unsupervised representation learning. Chapter 3 presents the general structure of animal activity recognition pipeline, the baseline of this thesis, the comparison with the basis of this work and our methodology and building blocks in details. Chapter 4 presents details of the experiment setup. Chapter 5 presents the evaluation metrics, the experiment procedures, results, discussion and the experiment environment. Chapter 6 is the last chapter of this thesis which concludes this thesis and provides possible future work based on this thesis.

Chapter 2

Background and Related Work

This chapter provides background information and related work about the topic of unsupervised representation learning. The variational autoencoder, transfer learning are addressed in detail, and the related work regarding this topic is discussed as well.

2.1 Unsupervised Representation Learning

Representation learning aims at obtaining a valuable representation of data. Learning representations of the data makes it easier to extract useful information when building classifiers or other predictors [14]. It is a domain in machine learning. It can learn representations automatically, which avoids the labour of handcrafted feature extraction. Supervised learning learns a hierarchy of representations in hidden layers, and it learns from low-level representations to high-level representations towards the supervised goal [22]. However, it needs a large amount of labeled data. It is also possible to learn the representations in an unsupervised manner where it needs a large amount of unlabeled data that is much easier to obtain.

Bengio et al. [23] gave an overview of the background, motivations, applications and techniques of representation learning. Multiple unsupervised representation learning may be used while the focus in this thesis is on Variational Autoencoders (VAE). It is a relatively new technique that was introduced in 2013 [15]. Furthermore, it links the representation learning to a generative model, and it provides methods to create sound data output from scratch, for instance, after training VAE, we could input different feature vectors to its decoder and it can create sound data output no matter whether it has been seen during the training. VAE tends to encode the data input into the latent representation space, close to a given prior distribution. Therefore, VAE forces the data to be representative of this distribution. The followings are general concepts of autoencoder, VAE and the type of layers used in this thesis.

2.1.1 Autoencoders

For decades, autoencoders have appeared for dimensionality reduction and representation learning [14, 24, 25]. It is an unsupervised neural network that is trained to reconstruct its input. The structure of an autoencoder is illustrated in Figure 2.1. It consists of an encoder and a decoder, where the encoder and decoder itself are neural networks. X is input data, and X' the output, which is also called reconstructed input. The encoder takes the input X and transforms it into code z, which is the latent space, also called as representations. The code z is the features extracted from input data X. The decoder then inputs the generated code z and attempts to reconstruct the original input X. In other words, it attempts to make the output X' identical to the input X. Therefore, training such an autoencoder is based on the difference between the input X and the output X'. The goal of training an autoencoder is to learn valuable representations z of the input X, which can be used in the downstream task such as classification or prediction.

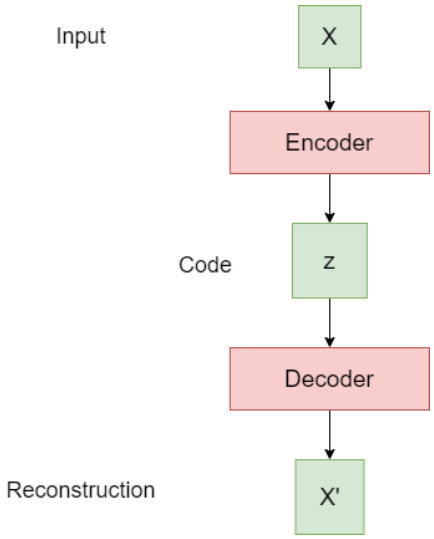

Figure 2.1: The structure of an autoencoder

However, the standard autoencoders have some limitations [15]; for instance, they can only reconstruct those data on which they are trained; in other words, they compress the data while producing the code and try to replicate the output to the input. Another limitation is that the latent space vectors are not continuous, which means new data cannot be generated from the latent space vector. Meanwhile, VAE tends to be more robust [26]. In terms of these limitations, variational autoencoder works better than standard autoencoders.

2.1.2 Variational Autoencoders (VAEs)

In terms of structure, VAE is similar to the standard autoencoder, where VAE also consists of an encoder and a decoder. However, the significant difference is that the latent vector is continuous, making VAE a generative model. In the following, the details of the standard VAE are addressed [15].

The structure of a VAE is illustrated in Figure 2.2. The problem of the standard VAE is that let us assume we have a dataset $X = \{x^{(i)}\}_{i=1}^{N}$ which consists of N independent identically distributed samples from some continuous or discrete variable x and the data are generated by a random process including an unobserved continuous random variable z, which is the latent variable. During this process, a value $z^{(i)}$ is generated from a prior distribution $p_{\theta^*}(z)$, and a value $x^{(i)}$ is generated from a conditional distribution $p_{\theta^*}(x|z)$. It is assumed that this prior $p_{\theta^*}(z)$, and likelihood $p_{\theta^*}(x|z)$ come from parametric families of distributions $p_\theta(z)$ and $p_\theta(x|z)$, respectively, and their probability density functions are differentiable almost everywhere with regard to both θ and z. However, the true parameters θ^* and the values of the latent variables $z^{(i)}$ are unknown to us. We are interested in the marginal likelihood $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$ and the true posterior density $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x)$ to describe the original data distribution and its relation to the latent variables. The integral and posterior are intractable. Thus, we would be interested in an efficient approximation for these distributions. A recognition model $q_\phi(z|x)$ approximation to the intractable true posterior $p_\theta(z|x)$ can be introduced. Then, a method to learn the recognition model parameters ϕ jointly with the generative model parameters θ is introduced.

The marginal likelihood consists of a sum over the marginal likelihoods of individual datapoints, which is,

$$\log p_\theta(x^{(1)}, ..., x^{(N)}) = \sum_{i=1}^{N} \log p_\theta(x^{(i)}) \tag{2.1}$$

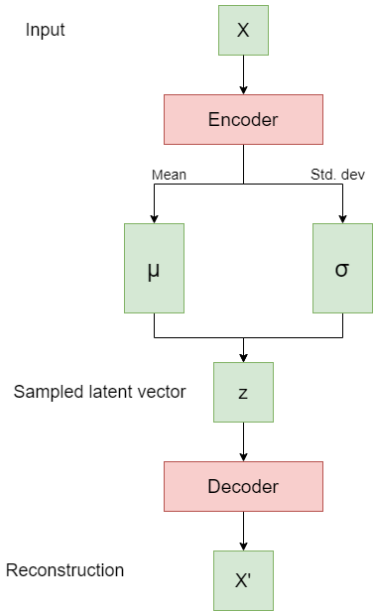

Figure 2.2: The structure of a VAE

Each of the terms can be written as

$$\log p_\theta(x^{(i)}) = D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z|x^{(i)})) + \mathcal{L}(\theta, \phi; x^{(i)}) \tag{2.2}$$

The first term is the KullbackLeibler (KL) divergence [27] of the approximate posterior from the true posterior, and the second term is the variational lower bound. The KL divergence can be regarded as a measure of "surprise", which illustrates the extent where the model must update its parameters to fit the new observations. The second term is called the variational lower bound on the marginal likelihood of data point $x^{(i)}$, which can be rewritten as

$$\mathcal{L}(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \mathbf{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)] \tag{2.3}$$

The objective is to maximize the variational lower bound by optimizing the parameters ϕ and θ of the neural network. The goal is to maximize the $\mathbf{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)]$ term which means that the decoder is getting better at reconstruction. This means that the reconstruction loss which is $\mathcal{L}_R$ need to be minimized. The reconstruction loss $\mathcal{L}_R$ is the log of the Gaussian distribution which can be regarded as Mean Square Error (MSE) loss [28] and then we need to maximize $-D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z))$ which can be rewritten as

$$-D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) = \frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma_j)^2 - (\mu_j)^2 - (\sigma_j)^2) \tag{2.4}$$

where σ_j is the standard deviation and μ_j is the mean. For maximizing $-D_{KL}$, we need to make σ_j close to 1 and μ_j close to 1. This loss is called $\mathcal{L}_{KL}$, and σ and μ are sampled from the encoder's output. So, the final VAE loss that needs to be optimized is:

$$\mathcal{L}_{VAE} = \mathcal{L}_R + \mathcal{L}_{KL} \tag{2.5}$$

In order to train the network by means of backpropagation with gradient descent, the model should be differentiable with regard to its learned parameters which means the model should be deterministic such that only the inputs are stochastic. However, if z is sampled based on a probability distribution with parameters inside the model, this is not the case. A simple parametrisation trick can be used to solve this problem where we define auxiliary random variables $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ as input; then, we can sample the latent variable z by using the following formula:

$$z = \mu + \epsilon \odot \sigma \tag{2.6}$$

2.1.3 Layer of Encoder and Decoder

The layers of the encoders and decoders can be any type of neural network. For example, Recurrent Neural Network (RNN), which is to handle the sequential data, Long Short-Term Memory (LSTM), which is an optimization algorithm based on RNN; and Convolutional Neural Network (CNN), which is to handle high dimensional data such as images [29, 30].

Recurrent Neural Network (RNN)

RNN is used to handle sequential data such as the audio content in the voice [31], the text content in an article [32], the accelerometer reading during an activity [33], etc. RNN should be able to know what happened in the last step; as a result, RNN needs memory. Figure 2.3 shows the structure of RNN; the green box indicates the sequential inputs from x_1 to x_n, the red box indicates the cell, and the blue box indicates the output. So, the inputs for each step consists of the input from the current sequential data and the hidden state from the last step. As a result, the previous inputs have an impact on the output.

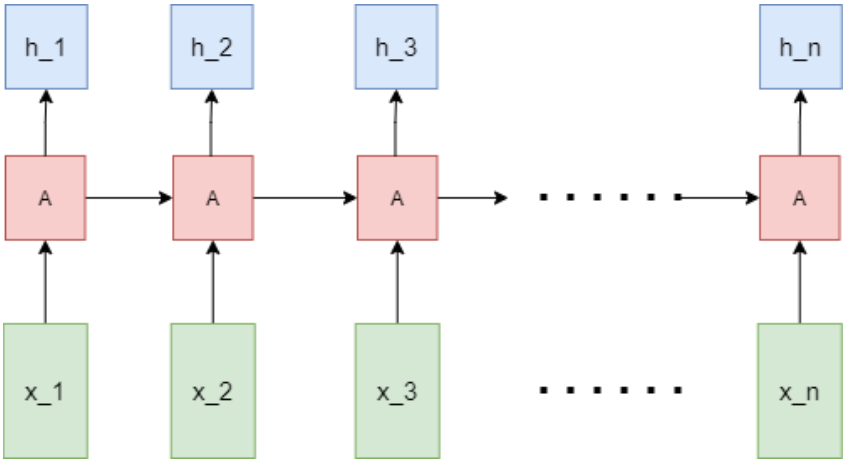

Figure 2.3: The structure of RNN

A cell of RNN illustrated by "A" in Figure 2.3 can be found in Figure 2.4. In Figure 2.4, "tanh" indicates the hyperbolic tangent activation function and "tanh" can be calculated as:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.7}$$

The RNN cell "A" takes the output from the last cell and the current sequential input as the cell inputs. After a simple structure such as a single "tanh" layer, it generates the output, and meanwhile, the output also goes to the next cell.

However, there are obvious disadvantages using RNN. Due to the characteristics of RNN, the later input has more impact on the output, and the earlier input has less impact on the output. So, it has a short-term memory problem which means it cannot handle long input sequence. This would further cause gradient vanishing problems. To solve it, the Long Short-Term Memory (LSTM) networks have been developed.

Long Short-Term Memory (LSTM)

LSTM network has tackled the problem of RNN where LSTM can handle long input sequence and the gradient vanishing problems. The overall structure is the same as RNN, which can be found in Figure 2.3. However, the cell structure of LSTM is different from that of RNN. The cell structure of LSTM can be found in Figure 2.5.

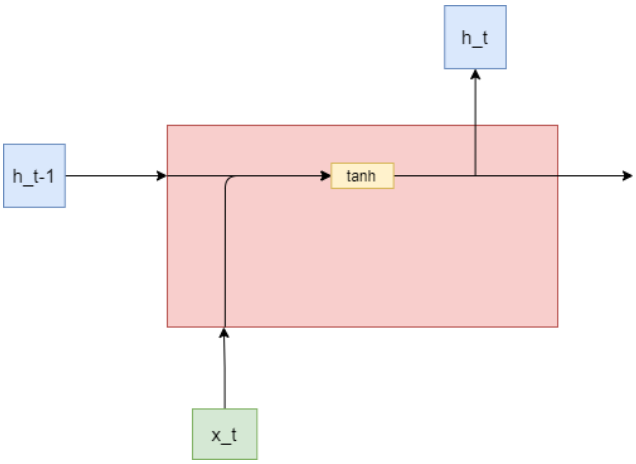

Figure 2.4: A cell of RNN

As can be seen in Figure 2.5, the number of components is more than that of the RNN cell. "σ" indicates the sigmoid function which can be calculated as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

There are four shadowed parts which are noted by "C", "I", "O" and "F" where "C" indicates the cell state, "I" indicates the input gate, "O" indicates the output gate, and "F" indicates the forget get. The followings are some explanations about these shadowed parts:

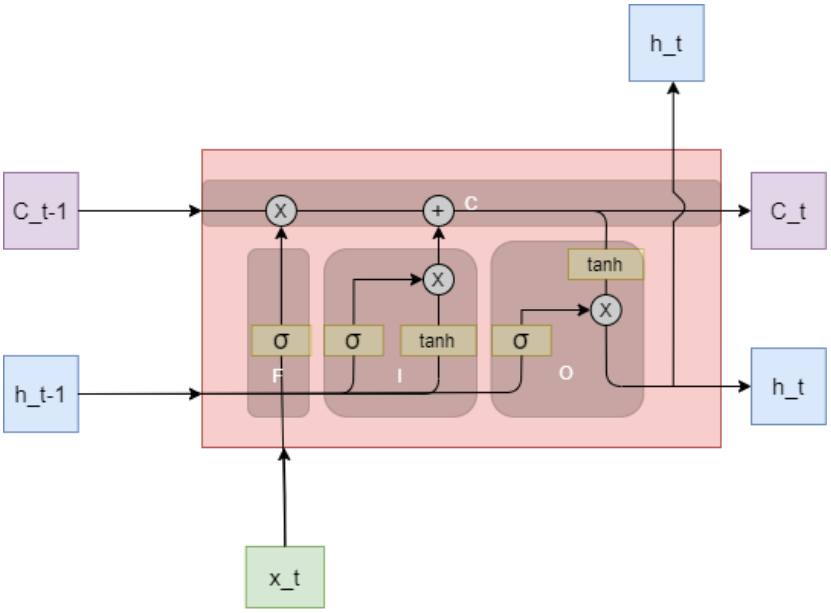

Figure 2.5: A cell of LSTM

- "C": the cell state behaves like a conveyor belt. It goes through the entire chain with some minor linear interactions in order to make the information flowing along with it unchanged.
- "I": the input gate decides which information is added to the cell state. It consists of two parts which are a "sigmoid" function and a "tanh" function. "sigmoid" function decides whether to pass the information, and the "tanh" function decides what information to pass.

- "O": similar to the input gate, the output gate decides which information is output from the cell. It also consists of two parts which are the "sigmoid" function and "tanh" function. "sigmoid" function decides whether to output the information, and the "tanh" function decides what information to output.
- "F": the forget gate decides which information can be passed from the cell state output of the last cell

With those shadowed parts, LSTM is able to handle the long sequence and the gradient vanishing problems from RNN.

Convolutional Neural Network (CNN)

CNN is able to handle high dimensional data such as images. Furthermore, it can also handle the data with local correlation such as speech [34], natural language processing (NLP) [35] and activity recognition [36]. A typical CNN consists of three parts which are convolution layer, pooling (subsampling) layer and fully-connected layer. The convolution layer is for feature extraction; the pooling layer is for dimension reduction and over-fitting avoidance; the fully-connected layer is for classification. Normally, instead of the 3-layer mentioned above, CNN would have multiple layers to gain higher performance. For example, the architecture of LeNet-5 [1] can be found in Figure 2.6, which consists of seven layers.

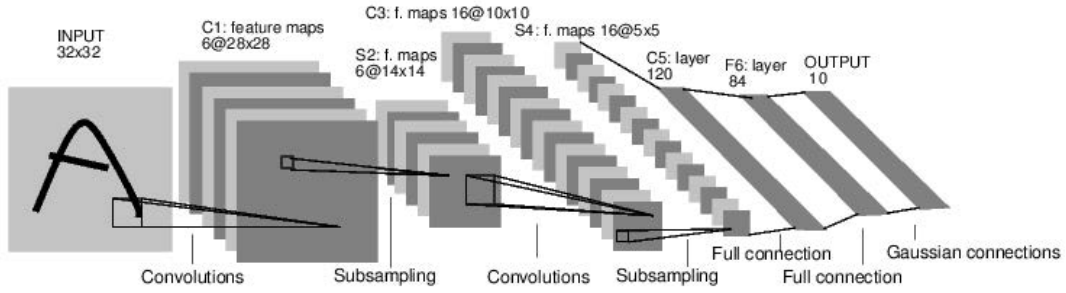

Figure 2.6: Architecture of LeNet-5 taken from [1]

2.1.4 Optimization

The loss functions of VAE and classification neural network training can be found in Section 2.1.2 and 2.2 individually. Machine learning optimization is the process of tuning hyperparameters in order to minimize the loss functions by using one of the optimizers. It is vital to minimize the loss function as it describes the discrepancy between the true value of the estimated parameter and what the model has predicted. There are various optimizers [37] that we can use for training our neural network such as Batch Gradient Descent, Stochastic Gradient Descent, Adagrad, RMSprop, Adam. Recommended by [37], we chose Adam as the optimizer training our neural networks as Adam is the best performed optimizer.

Adam [38], short for Adaptive Moment Estimation, is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squard gradients v_t, Adam also keeps an exponentially decaying average of past gradient m_t:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\, g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\, g_t^2 \end{aligned} \tag{2.9}$$

m_t and v_t are the estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively. As m_t and v_t are initialized as the zero vector, the authors of [38] observe that they are biased towards zero, especially during the initial time steps and when

the decay rates (β_1 and β_2) are small. g_t indicates the gradient. They counteract these biases by computing the bias-corrected first and second moment estimates:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{2.10}$$

this is called bias correction which is a correction for the large initial moving average deviation, so, when t becomes larger and larger both $1-\beta_1^t$ and $1-\beta_2^t$ are converged to 1. The last thing is to update the parameters using Adam update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t \tag{2.11}$$

θ indicates the parameters, η is the learning rate we set and ϵ is a small value to avoid "divide by zero error" while updating the variables. The learning rate of Adam optimizer is $\frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}$ which is adaptive, so, the learning rate of each parameter and each training epoch becomes varying. The proposed default values suggested by [38] is 0.9 for β_1, 0.999 for β_2 and 10^{-8} for ϵ. They show that Adam works empirically well in practice and compared with other adaptive learning method algorithms, Adam performs better.

The hyper-parameters such as batch size, learning rate, number of epochs should depend on the experiments. As for the batch size, authors of [39] suggests that a batch size of 32 is a good starting point while we can also try with 64, 128, and 256 which are generally good choice. As for the learning rate and the number of epochs choices, by observing the loss curve during training (for example, if the loss fluctuates a lot, then we should decrease the learning rate; if the loss decreases too slow, then, we may consider increasing the learning rate; if the trend of loss decreasing of the last epoch training is still high, then, we may consider increasing the number of epochs), a descent choice can be made. In terms of hidden size, authors of [40] suggests to start with the hidden size equal to half of the sum of the number of input neurons and the number of output neurons and [41] indicates that we should avoid using too large latent space dimension. As a result, for batch size, hidden size and latent space dimension tuning, we may use trial and error strategy [42] to choose a descent hyper-parameter. For example, we try different values of one hyper-parameter while we fix the value of all other hyper-parameters, by comparing the performance, we can choose a descent one.

2.2 Classifiers

The task of unsupervised representation learning such as VAE is to learn the representations and after that, the down stream task is classification where a classifier is used to do the classification job with labeled data. There are many different types of classifiers. The frequently used classifiers are Softmax and Support Vector Machine (SVM) [43].

2.2.1 Softmax

The Softmax classifier takes the input no matter they are positive, negative or zero and output the value between 0 and 1 which can be interpreted as probability of that class. The Softmax formula is in the following:

$$S_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.12}$$

where S_i is the output, the probability that it belongs to the class i; All the z_i values are the elements of the input vector to the Softmax classifier; e^{z_i} is the standard exponential function applied to each element of the input vector where it gives a positive value above 0 and if the input is negative, it gives small value while it gives large value when input is large; $\sum_{j=1}^{K} e^{z_j}$ is

the normalization term which ensures that all the output values are summed to 1 and each of the output value is in the range (0,1); K is the number of classes. In Neural Network, the softmax classifier can be implemented by using Cross-Entropy Loss Function [44] which is defined as:

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log\left(p_i\right) \tag{2.13}$$

where t_i is the truth label, p_i is the Softmax probability for the i^{th} class and n is the number of classes. Each predicted class probability is compared to the actual class desired output 0 or 1 and a loss is calculated that penalizes the probability based on how far it is from the actual expected value. The penalty is logarithmic in nature yielding a large score for large differences close to 1 and small score for small differences close to 0. Given the above loss functions, it can be trained using the gradient descent optimization.

2.2.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a classification model which is defined as a linear classifier with the largest margin in the feature space. The training strategy of SVM is to maximize the margin. In Neural Network, the SVM can be implemented by using hinge loss function [45]:

$$L(y) = \max(0, 1 - \hat{y}y) \tag{2.14}$$

where $\hat{y}$ is the prediction and y is the truth label. The loss measures the error due to misclassification and the error is 0 if the data point is classified correctly. Minimizing the hinge loss function is equivalent to finding the max-margin. By using the gradient descent optimization, the SVM can be trained with the hinge loss function.

2.3 Transfer Learning

Amount, type, and quality of data play an essential role in the performance of machine learning. A common assumption that the machine learning methods work well is that the training and testing data come from the same feature space and distribution [46]. When the training data and testing data are from different distribution (for example, from different animal species such as horse and goat for AAR), the performance may decrease. One of the solutions to handle it is to re-train the whole model from scratch, however, it is expensive and it can have a good result when the amount of provided labeled training data is high. When the amount of labelled training data is low, it would be a good idea to use transfer learning as transfer learning can be a good starting point to make the model produce a good performance, because it encompasses prior knowledge on similar cases. Transfer learning is formally defined as follows [46]: Given a source domain D_s and a learning task T_s, a target domain D_T and a target learning task T_T, transfer learning aims to help improve the learning of the target prediction function in D_T using the knowledge in D_S and T_S where $T_S \neq T_T$ and $D_S \neq D_T$.

Some methods are suggested by Karpathy et al [47] to do the transfer learning for CNN:

1 Use CNN as a feature extractor:

  A pre-trained CNN on large datasets is taken and all layers except the last layer are frozen, then, we train it where only the parameters of the last layer can be updated.

2 Fine-tune the CNN:

  A pre-trained CNN model is taken and we train it on the new dataset where all layers parameters can be updated by backpropagation

Although the methods suggested above [47] are for CNN, it also works for other neural network.

2.4 Related Work

Over the past decade, the interest in activity recognition employing Machine Learning (ML) techniques has been increasing. In terms of activity recognition, there are human activity recognition (HAR) and animal activity recognition (AAR). They are similar because the same type of sensors could be used, such as cameras, accelerometers, gyroscopes and microphones, which can be placed on the subject's body or the environments they live in or both of them. Furthermore, they can share the same activity recognition pipeline for time-series sensor data which involves sliding window segmentation, feature extraction and activity classification [48]. However, there are differences between them, such as the type of activities, the sensor location and the movement patterns. HAR has been more widely studied than AAR in the research fields; however, they have overlap, and the research in HAR could inspire the research in AAR. [2]

The traditional approach to activity recognition is to adopt hand-craft features such as statistical features [49, 50] and basis transform features [51] and couple them with classifiers such as Support Vector Machine (SVM) [52], decision trees [53] and graphical model [54]. However, it relies on domain expert knowledge to design laborious features and cannot learn complex features. Then, with more and more popularity in deep learning in various research areas such as natural language processing, speech recognition and computer vision [14]. It inspires the interest in adopting deep learning techniques in HAR, such as using the Convolutional Neural Networks (CNNs) [55, 56] or using a recurrent-based neural network (RNN) [16, 33, 57] to extract the high-level features. However, the feature is learned from supervised learning and by considering the difficulty of annotating a large amount of data, it motivates the need for unsupervised representation learning.

2.4.1 Human Activity Recognition (HAR)

Gu et al. [11] proposed a deep learning method consisting of stacked denoising autoencoders for locomotion activity recognition where it collected four sensor readings: accelerometer, gyroscope, magnetometer and barometer together with their timestamps, then, the learned features were fed into a Softmax classifier. It turned out that a combination of sensor data had better recognition performance than using only acceleration data, more neurons resulted in better performance but higher computational cost and fine-tuning increased the performance. Similarly, Wang et al. [58] adopted the same representation learning method using multiple binary sensor data in two formats which are binary representation and numerical representation, the feature extraction and classifier training were integrated into a unified framework to obtain a jointly optimized activity recognizer. They addressed the challenge of building a robust activity recognition model as the inherent nature of human activities is characterized by a high degree of complexity and uncertainty. Almaslukh et al. [59] adopted a stacked autoencoder with a softmax classifier on the top while input handcrafted features instead of data. It had better performance than directly inputting data. Their proposed methods provided a low cost option for smartphone-based HAR, meanwhile, the results of their model outperformed the state-of-art studies on the same dataset. Similarly, Alireza et al. [60] considered HAR as a set prediction problem which is able to recognize multiple activities in a time segment. It was the first time that a novel formulation of human activity recognition problem from body worn sensor data streams where the predictions for sensory time segments were expressed as activity sets and it avoided the potential loss of information from conventional ground-truth approximations. João et al. [18] used recurrent variational autoencoder (VAE) to learn representations for unsupervised Anomaly Detection(AD) with several classifiers and latent space-based detection using clustering and the Wasserstein distance where the fully unsupervised methods were addressed. Bai et al. [17] used VAE where the encoder and decoder are bidirectional Long Short-Term Memory (Bi-LSTM). A large wrist-band dataset of real-world activities without labelling were used to train the model. Their model enabled the movement embedding utilising the raw input from wearable sensor data, meanwhile, the effectiveness of their model was validated on both public datasets and their own collected datasets.

The taxonomy of HAR discussed above can be found in Figure 2.7

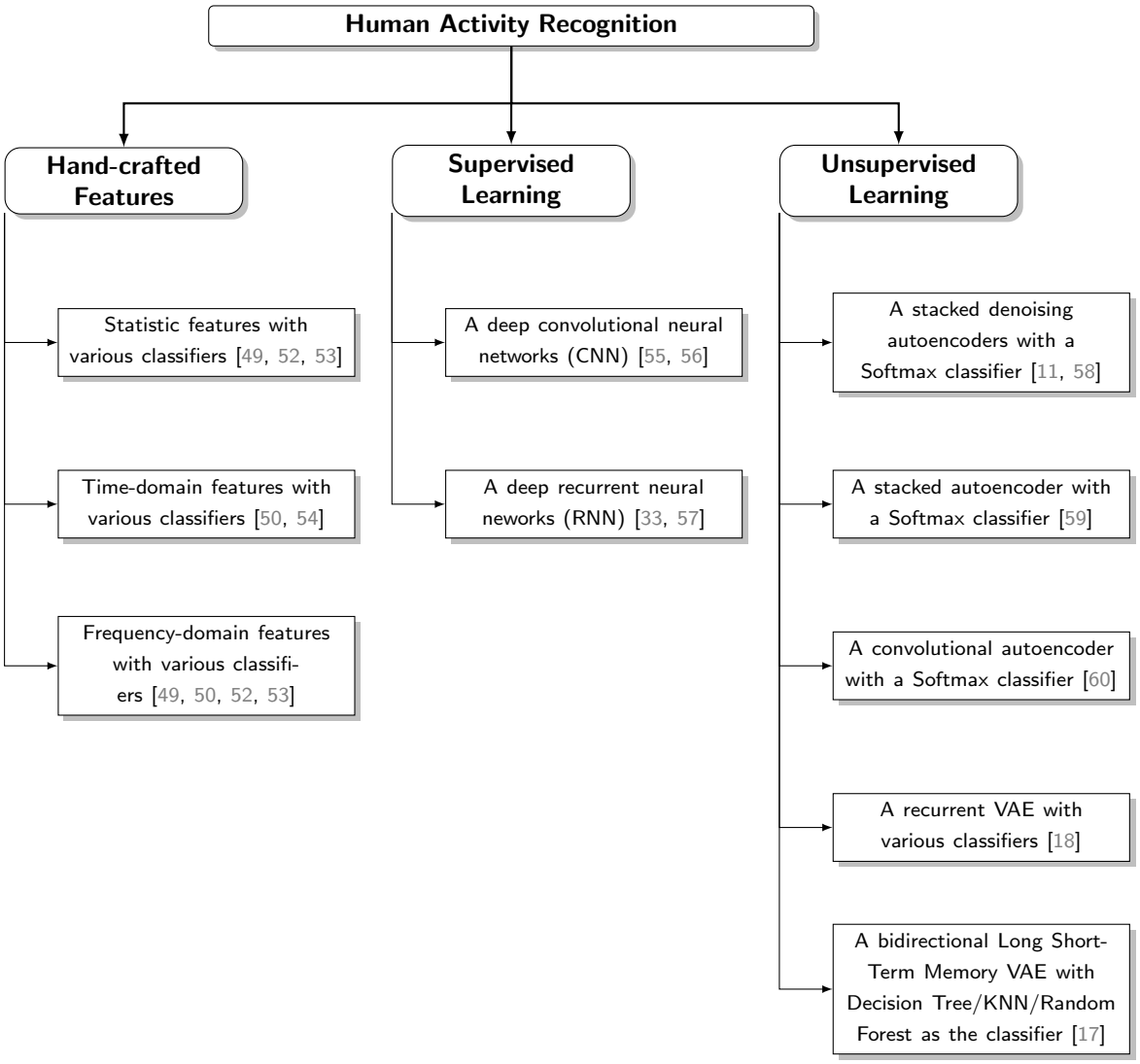

Figure 2.7: The taxonomy of HAR

2.4.2 Animal Activity Recognition (AAR)

Kamminga et al. [3] collected a large amount of horse movement data from eighteen individual horses where the sensors were attached to a collar around the neck of the horse with the orientation not fixed. The sensors contained a 3-axis accelerometer, gyroscope and magnetometer. The labeled data was then fed into a Naive Bayes (NB) classifier. Similarly, Kamminga et al. [9] collected a large number of goat movement data from five individual goats where the sensors were attached to a collar around the neck of the goats with orientation not fixed. The sensors contained a 3-axial accelerometer and a 3-axial gyroscope. The collected data was fed into multiple ML classifiers such as SVM, K-Nearest neighbours (K-NN), Neural Networks (NN), Decision Tree, NB and Linear Discriminant Analysis (LDA). It turned out that the goat activities can be accurately classified by using only the accelerometer data. Kamminga et al. [2] compared three unsupervised representation learning techniques (Convolutional Deep Belief Network (CDBN), Principle Component Analysis (PCA) and Sparse Autoencoder (SAE)) with three conventional feature extraction methods (spectral, temporal and combined) on two animal datasets (horses and goats), meanwhile investigating the effect of data size, depth of feature architecture. Then a multi-class SVM was used to classify the activities with the derived features. It turned out that

unsupervised representation learning techniques can approach and outperform the performance of engineered features for AAR and CDBN benefits the most from an increasing amount of unlabeled data especially in the more diverse and imbalanced dataset. By concatenating the 1st and 2nd layer, the representations often resulted in better classification performance. Furthermore, deep representations provided more robustness to imbalance in smaller labeled datasets. Enkeleda et al. [10] employed deep CNN for activity recognition of livestock animals such as horses and goats which outperforms the baseline [3]. They designated the advantages of late sensor fusion and showed that an increase on the number of filters on each layer did not necessarily lead to a greater classification accuracy and this was the first time that ConvNets were employed for animal activity recognition. Yingqi et al. [16] developed a recurrent neural network (RNN) with a long short-term memory (LSTM) model to monitor and classify cattle behaviour patterns using inertial measurement units (IMU), meanwhile, a convoluation neural network (CNN) was implemented for comparison. They also investigated the effect of multiple window size (64 (3.2s), 128 (6.4s), and 256 (12.8s)). It was found that the LSTM-RNN model classification performance was superior to the CNN model and the best performance was achieved with a window size of 64 (3.2s), Furthermore, a few physically similar behaviours were easily misclassified. Casella et al. [61] recognised horse gaits such as walking, trotting and canter using accelerometer embedded in smartwatches. Several ML techniques were used such as SVM, K-NN, NN and Discrimination Trees. The smartwatch was fitted in two horses' saddles and riders' wrists. All the four sensors on two horses and two riders had similar and high performance in all the cases (all the four sensors on two horses and two riders). An easy-to-use app for a commercial smartwatch was developed to perform data collection from the accelerometer sensor. A high classification accuracy was obtained for both data collected from the saddle and the riders' wrist. Ritaben et al. [62] applied several unsupervised clustering to study the natural structure of the sensor data and the outcome was then fed into supervised machine learning to classify cattle behaviour patterns using 3-axis accelerometer and magnetometer to investigate their physical behaviours. It turned out that the Tree learner model using the bagging ensemble classification reached the highest classification accuracy.

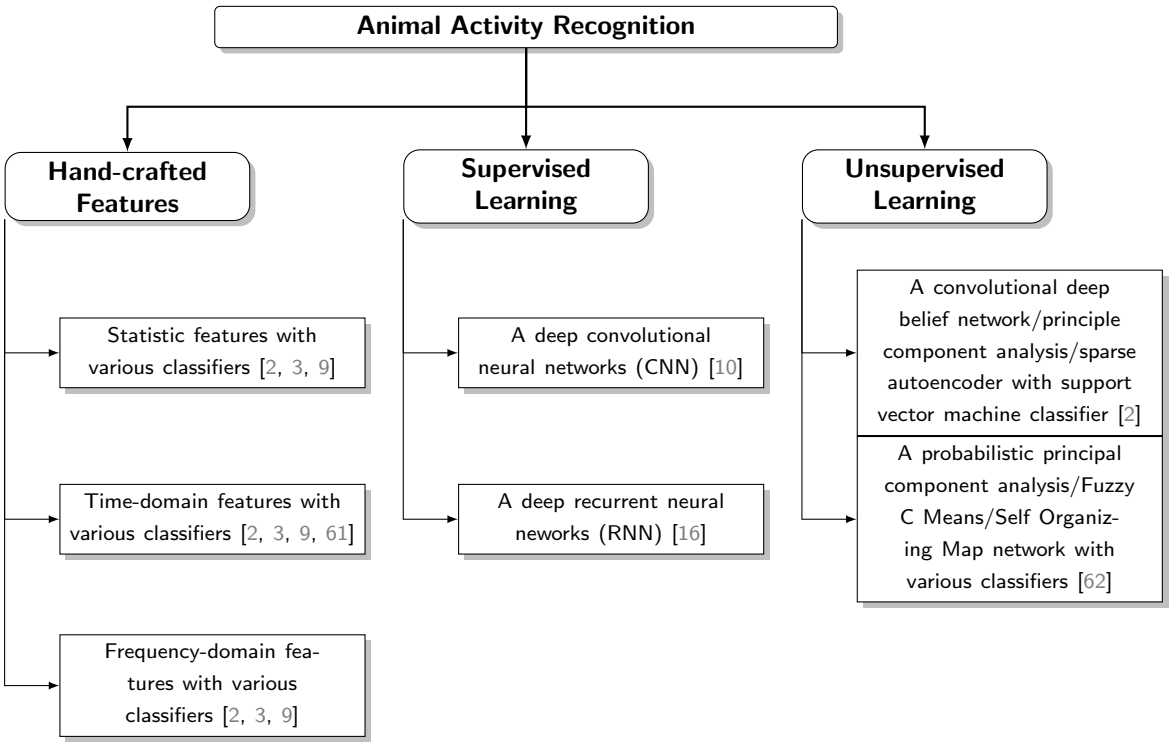

Figure 2.8: The taxonomy of AAR

The taxonomy of AAR discussed above can be found in Figure 2.8

2.4.3 Transfer Learning

Transfer learning has been applied successfully in many fields such as computer vision, NLP, HAR and so on [21]. For instance, Gopalan et al. [63] applied transfer learning to transfer models between different image data sets. They provided a comprehensive overview of domain adaptation solutions for visual recognition problems with three adaptation scenarios, namely unsupervised adaptation; semi-supervised adaptation; multi-domain heterogeneous adaptation. These techniques have shown improved performance on a variety of applications such as object recognition, face recognition, activity analysis. Francisco et al. [64] explored deep transfer learning with HAR, they considered transfer between users, application domains, sensor modalities and sensor locations. The trade-offs of transferring various convolutional laters along model size, learning speed, recognition performance and training data were characterized. A descent training time reduction without additional complexity was obtained. Kurz et al. [65] demonstrated the transfer of recognition capabilities from a fused multi-sensor ensemble to an untrained sensing device within the system in a real-world setup. Akbari et al. [66] addressed the limitation that machine learning algorithms trained on particular sensors need to be retrained upon any changes in configuration of the system, for example, adding a new sensor. They aimed to seamlessly train the machine learning algorithms for the new sensors to identify activities and observations which are detectable by the pre-existing sensors. Their proposed framework outperformed the state-of-art domain adaptation algorithm and gained one tenth improvement when training new sensors with limited unlabeled training data compared to training the model from scratch for the new sensor. However, to the best of our knowledge, transfer learning on AAR has not been addressed yet.

Chapter 3

Unsupervised Animal Activity Recognition using VAE

In the previous chapters, the motivations, relevant background information and the related work were discussed. In this chapter, firstly, we present the general structure of animal activity recognition pipeline and then we present and explain our baseline and the difference compared with the state-of-art [2] which is the basis of this work. Finally, we address our methodology and building blocks in details.

3.1 Unsupervised Animal Activity Recognition

Figure 3.1 shows a high-level overview of standard unsupervised animal activity recognition pipeline. It contains four main components which are data preprocessing, unsupervised representation learning, classification and optimization.

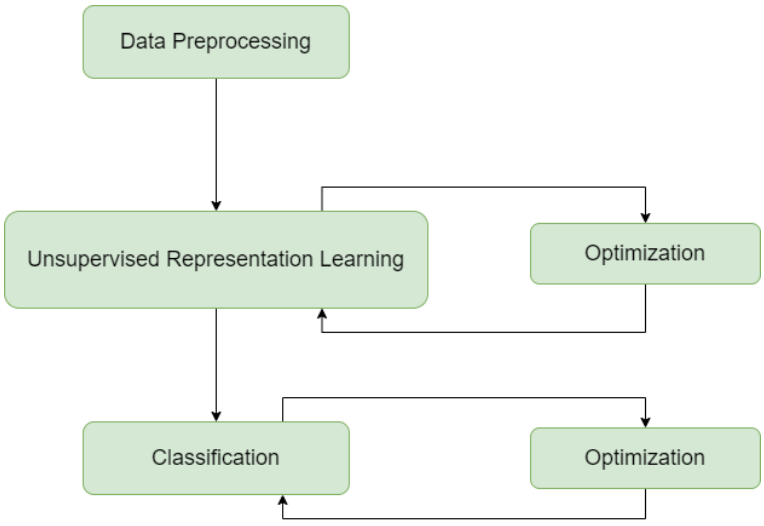

Figure 3.1: Standard pipeline of unsupervised animal recognition

- Data Preprocessing: This is to process the data so that the data can be used in the downstream task. In general, it consists of data cleaning, integration, reduction, transformation and so on [67]. In this thesis, we performed segmenting the data points, dividing dataset into unlabelled dataset and labeled dataset, adjusting data format, etc.

- Unsupervised Representation Learning: A large amount of unlabeled data is used for unsupervised representation learning. (Various unsupervised learning techniques could be used such as PCA, CDBN, autoencoder, VAE, GAN, etc. [2, 24, 68])
- Classification: The labeled training dataset is used to train a classifier and the labeled testing dataset is used for performance evaluation. (Various classifiers could be used such as Softmax, SVM, decision tree, KNN, Naive Bayes, etc. [2, 3, 62])
- Optimization: During the unsupervised representation learning model and classification model training, the hyper-parameters such as learning rate, batch size, amount of training epochs, etc. need to be tuned to minimize the loss functions, in order to make the model perform better [37].

3.2 The Inspiration for This Work

The baseline of this work was inspired by the work of Kamminga et al. [2]. The pipeline comparison between our work and their work can be found in Figure 3.2. On the left, it is the pipeline of our work baseline and on the right, it is the pipeline of their work. For both of them, horse [3] and goat [4] are utilized to evaluate the classificaiton performance (F1 score). The red color indicates the difference between our baseline and their work. The same methods of data preprocessing and classification were used while in our work, we further investigated the performance of another classifier (Softmax). The difference between our work and their work is the unsupervised representation learning methods. In their work, they used convolutional deep belief network, principle component analysis and sparse autoencoder while in our work, we used three different types of VAE as the unsupervised representation learning methods. As described in Seciton 2.1.1, VAE is a powerful generative probabilistic unsupervised representation learning model which has avoided many limitations of standard autoencoder and it tends to be more robust. As a result, the baseline of our work investigates how the unsupervised representation learning using VAE performs compared with the state-of-art [2]. The results of our work using different unsupervised representation learning methods (with SVM classifier) are to be compared with the results of [2].

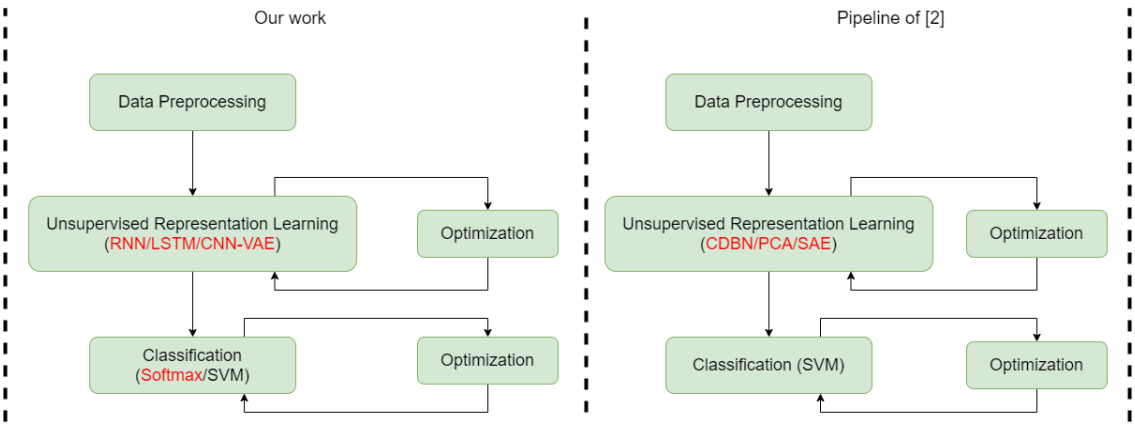

Figure 3.2: Comparison of our pipeline and pipeline of [2]

3.3 The Detailed Approach

Figure 3.3 shows a detailed overview of our approach, highlighting data flow and exact techniques used for each of main pipeline components. Each component will be explained in more details in the following sections.

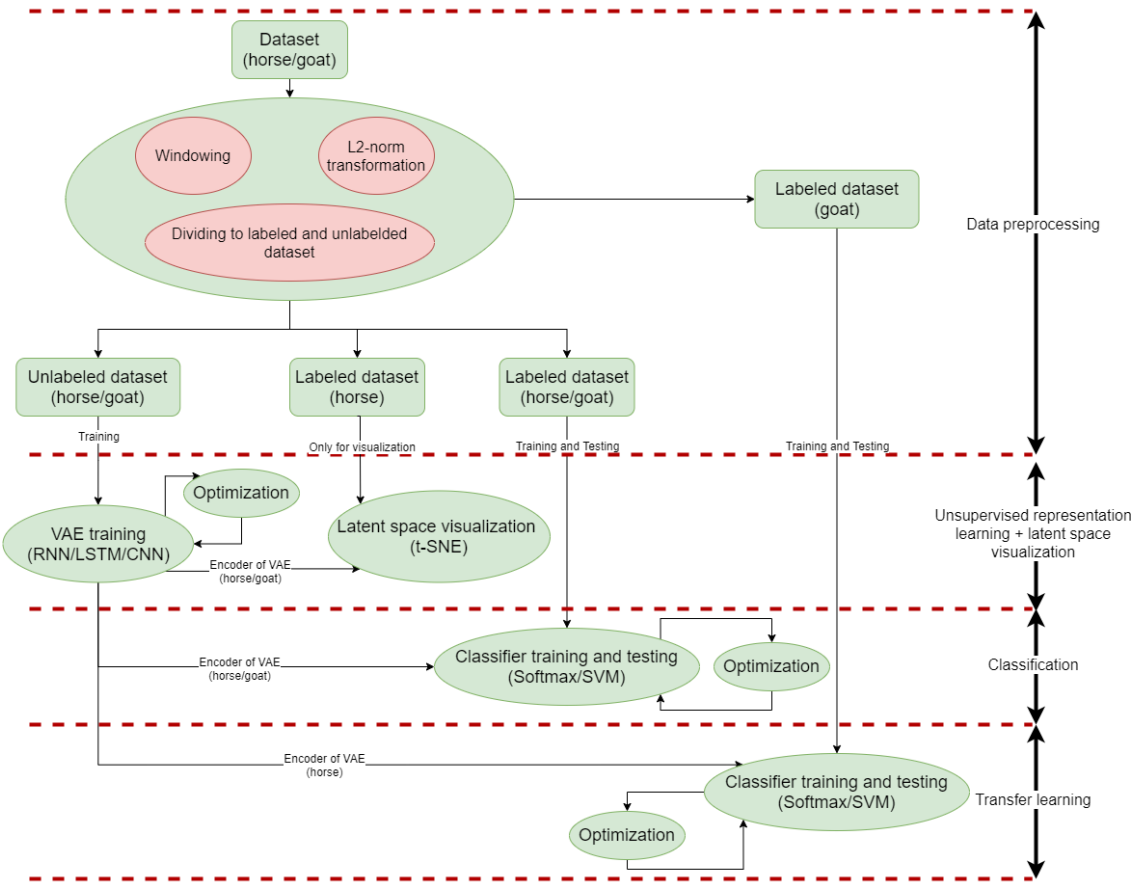

Figure 3.3: Detailed overview of unsupervised animal activity recognition pipeline of this thesis

3.3.1 Data Preprocessing

The datasets used are the horse dataset [3] and the goat dataset [4]. A more detailed description about these two datasets and how data was preprocessed are provided in Section 4.1. As the animal activities last for a time segment which contains multiple time instances (for example, multiple sensor readings comprise an animal activity), it is essential to take a segment (windowing) of multiple time instances. Besides, the datasets should be divided into unlabeled datasets and labeled datasets where unlabelled datasets are used for unsupervised representation learning and labelled datasets are used for latent space visualization and classifier training and testing.

For both datasets, the magnitude of 3D vector (l^2-norm) of the accelerometer data was used as it is theoretically orientation-independent [69] and it reduces the input dimensionality by a factor of three. The formula of obtaining the 3D vector can be found as below:

$$V(t) = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2}$$

where V(t) is the 3D vector (l^2-norm), a_x, a_y and a_z are the three axis of the accelerometer data. As for the more detailed specifications about the dataset, windowing and data division, they can be found in Section 4.1

3.3.2 Unsupervised Representation Learning

As described in the Section 2.1.2, VAE, as a generative model, is a good candidate for the unsupervised representation learning model. The VAE layers can be any type of neural network.

Considering the type of the input data which is time-series data, we consider the VAE with RNN layer, LSTM layer and CNN layer. In the rest of this thesis, "RNN-VAE" refers to the VAE with RNN layer as its encoder and decoder; "LSTM-VAE" refers to the VAE with LSTM layer as its encoder and decoder; "CNN-VAE" refers to the VAE with CNN layer as its encoder and decoder.

RNN-VAE

In an RNN-VAE, the encoder consists of one or more RNN layer, followed by one or more linear layers which connect the input layer to the μ and σ layers. The decoder consists of one or more linear layers which connect the sampled latent vectors (z) to the RNN layer and then followed by one or more linear layers connecting to the output layer. RNN-VAE is symmetric which means that the dimensions of the hidden layers of the decoder is the same as those of the encoder while the order is reversed.

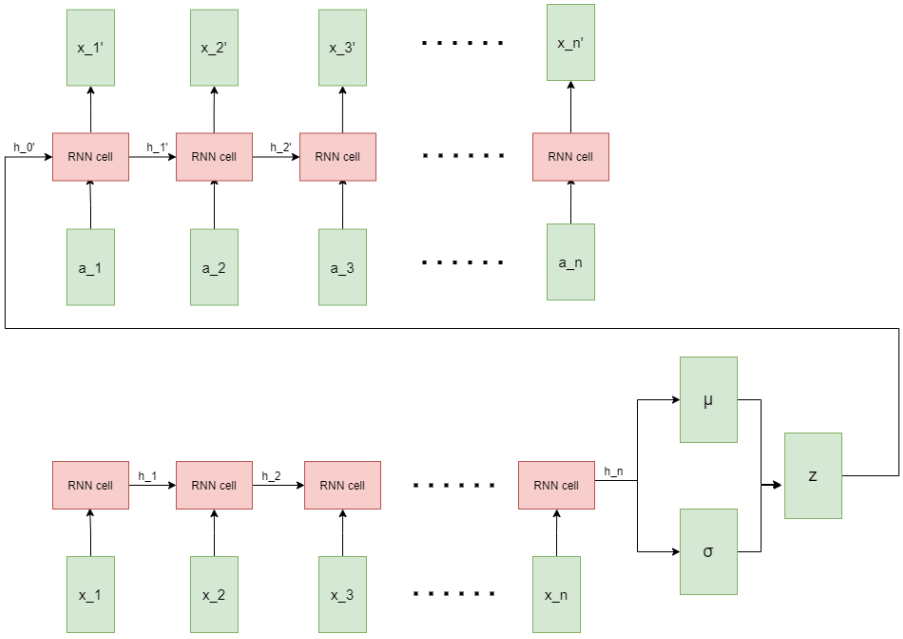

Figure 3.4: The structure of RNN-VAE

Figure 3.4 represents the RNN-VAE architecture, the RNN cells were used as the encoder and decoder. The (1 x n) input data (x_1,x_2,x_3, ..., x_n) to the VAE encoder is the time window of n data points. The (1 x n) data (a_1,a_2,a_3, ..., a_n) is the input data to the VAE decoder. The decoder inputs are initialized to zero and they are updated using backpropagation. h_i is the output of the last RNN cell (the $(i-1)^{th}$ RNN cell) of the encoder and h_i' is the output of the last RNN cell (the $(i-1)^{th}$ RNN cell) of the decoder. μ and σ are mean and standard deviation respectively. h_n is mapped to the mean (μ) and standard deviation(σ) by using a linear layer. z is the sampled latent vector and it is passed through a linear layer to obtain the initial states for h_0' which is the hidden state input of the first RNN cell of the decoder. The output of the RNN cell is mapped to the final reconstruction of the input by a linear layer. (x_1',x_2',x_3', ..., x_n') is the final reconstruction of the input vector. More details and specifications about RNN-VAE architecture are provided in Chapter 4, Section 4.2.1

LSTM-VAE

In an LSTM-VAE, similarly to the RNN-VAE, the encoder consists of one or more LSTM layers, followed by one or more linear layers which connect the input layer to the μ and σ layers. The decoder consists of one or more linear layers which connect the sampled latent vectors (z) to

the LSTM layer and then followed by one or more linear layers connecting to the output layer. LSTM-VAE is also symmetric.

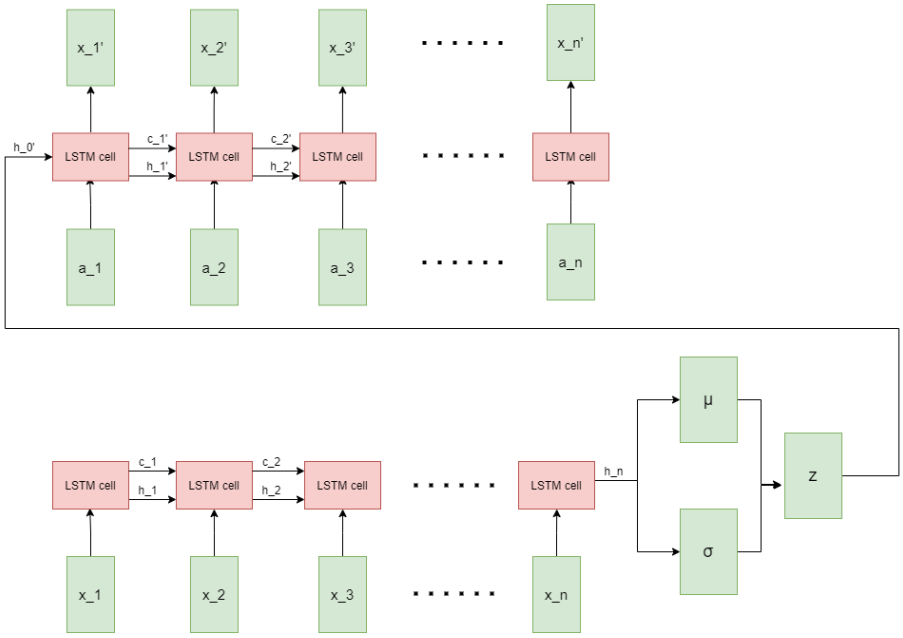

Figure 3.5: The structure of LSTM-VAE

Our architecture of LSTM-VAE can be found in Figure 3.5, it is similar to RNN in Figure 3.4 while LSTM-VAE has c_i which is the cell state output of the last LSTM cell(the $(i-1)^{th}$ LSTM cell) of the encoder and c_i' is the cell state output of the last LSTM cell (the $(i-1)^{th}$ LSTM cell) of the decoder. Figure 2.5 in Section 2.1.3 shows the details inside the LSTM cell. More details and specifications about LSTM-VAE architecture are provided in Chapter 4, Section 4.2.1

CNN-VAE

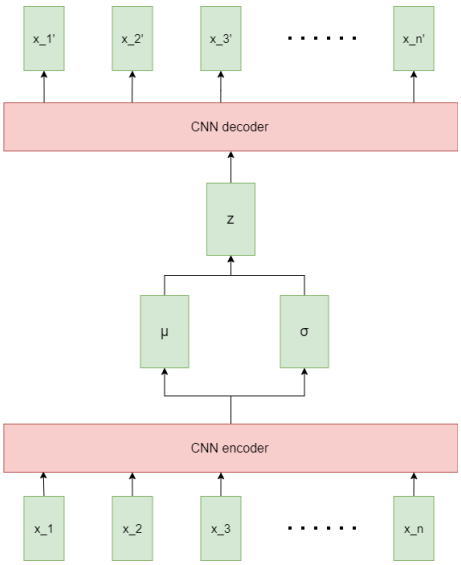

Figure 3.6: The structure of CNN-VAE

In a CNN-VAE, the encoder consists of one or more convolutional layers, followed by one or more linear layers which connect the input layer to the μ and σ layers. The decoder consists of one or more linear layers which connect the sampled latent vectors (z) to the CNN layer and then followed by one or more linear layers connecting to the output layer. CNN-VAE is also symmetric.

The architecture of CNN-VAE can be found in Figure 3.6, the notations shown in the Figure 3.6 is the same as those described above in the RNN-VAE Section. The output of the CNN encoder is mapped to the mean (μ) and standard deviation(σ) by using a linear layer. z is passed through a linear layer to obtain the input to the CNN decoder and the output of the CNN decoder is passed through a linear layer to obtain the reconstructed data input (x_1',x_2',x_3', ..., x_n'). More details and specifications about CNN-VAE architecture are provided in Chapter 4, Section 4.2.1

3.3.3 Latent Space Visualization

After the VAE was trained, all of the horse labeled data were fed into the encoder and the latent vectors were output, then t-Distributed Stochastic Neighbor Embedding (t-SNE) was implemented to reduce the dimensions of the latent space and visualize the distribution of the embedded vectors so that we can check how good the VAE has learned by visualizing how separable the tasks are.

t-SNE is a non-linear dimensionality reduction algorithm which is used for exploring the high-dimensional data. It maps the high-dimensional data to low dimensions such as 2 dimensions which is suitable for human being's visualization. The latent space vector tends to be in high dimension, with the help of t-SNE, we could decrease its dimensions and visualize it.

Generally, there are two popular dimensionality reduction algorithms. These two are Principal Component Analysis (PCA) and t-SNE [70]. PCA is a linear algorithm which will not be able to show the complex polynomial relationship between different features while t-SNE is based on probability distributions with random walk on neighbourhood to look for the structure within the data. Furthermore, linear algorithm concentrates on placing dissimilar data points far apart in a lower dimension representation, however, the similar data points should be represented close together to represent high dimension data points on low dimension, non-linear manifold. As a result, we decided to use t-SNE as dimensionality reduction algorithm for latent space visualization.

3.3.4 Classification

For the classification, we freezed the encoder of the trained VAE and a linear layer (classifier) was implemented on top of it, whose structure can be seen in the Figure 3.7. We used such a shallow classifier architecture as we do not want the classification performance rely too much on the classifier, on the contrast, the classification performance should rely on the unsupervised representation learning (for example, a complex classifier may results in better performance but it is not the contribution of the unsupervised representaiton learning). For more straightforward expression, the freezed trained VAE encoder here refers to the VAE encoder with Mean layer (μ) and Std.dev layer (σ) in Figure 2.2 which means that the output is the latent vector of input in the latent space. So, basically, we do the classification on the latent space. When it is RNN-VAE, then, the "Trained VAE encoder" is "RNN encoder". When it is LSTM-VAE, then, the "Trained VAE encoder" is "LSTM encoder". When it is CNN-VAE, then, the "Trained VAE encoder" is "CNN encoder".

Two widely-used classifiers are Softmax and Support Vector Machine (SVM). They can be implemented in the gradient descent manner where we can use backpropagation to update the parameters. They share the same structure as shown in Figure 3.7 but different loss functions. Softmax uses Cross-entropy loss function and SVM uses hinge-loss functions [45]. A more detailed description about these two classifiers is provided in Section 2.2 The number of output of the "Classifier" in Figure 3.7 is the number of output classes.

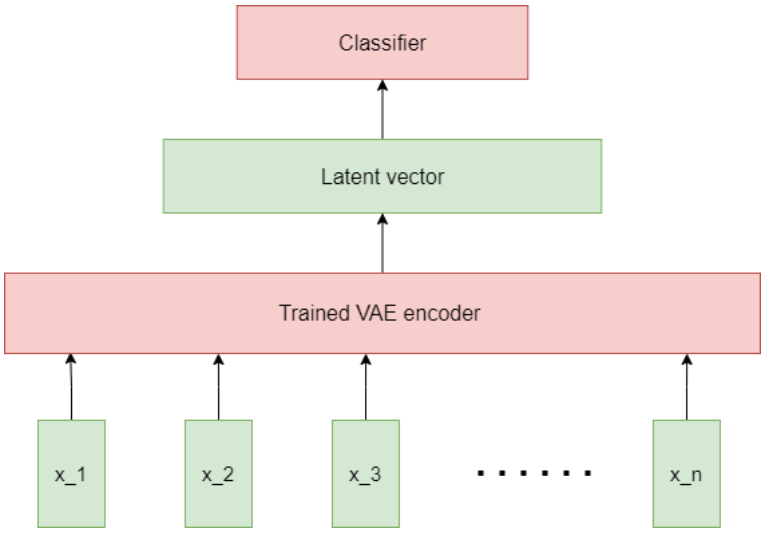

Figure 3.7: The structure of classifier

"Frozen Encoder" and "Not Frozen Encoder"

In this thesis, we also investigate the performance of "Frozen Encoder" and "Not Frozen Encoder", as can be seen in Figure 3.7, "Frozen Encoder" implies that the parameters of "Trained VAE encoder" in the red block are frozen and "Not Frozen Encoder" indicates that the parameters of "Trained VAE encoder" in the red block are not frozen where its parameters can be updated during the training.

Pure Classification

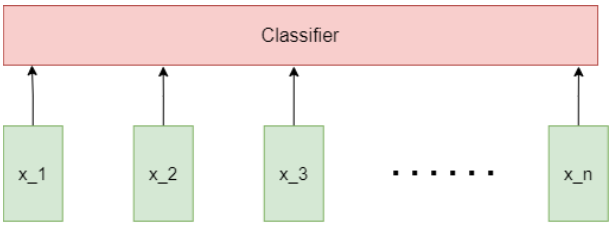

Figure 3.8: The structure of pure classifier

To inspect whether the data can be easily classified, we investigated the performance of feeding the input data directly into the classifier where the structure can be seen in Figure 3.8. As the input data is directly fed into the shallow architecture classifier, so, a term "pure classification" was used here.

3.3.5 Optimization

As indicated in Section 2.1.4, machine learning optimization is the process of adjusting hyperparameters so that the loss functions can be minimized by using one of the optimizers. Suggested by [37], we chose to use Adam [38] as the optimizer training our neural networks because Adam performs better than other optimizers [37]. Adam optimizer is addressed in details in Section 2.1.4. A learning rate that is too small results in painfully slow convergence, however, while a learning rate is too large, it can hinder convergence and lead to the loss function to fluctuate around the minimum or even to diverge. As a result, it is essential to choose a descent learning rate. The

strategy we used to choose a descent learning rate is by observing the loss curve, for example, if the loss decrease is too slow, then, we may increase the learning rate; if the loss decreases too fast or the loss fluctuates a lot or even diverges, we may decrease the learning rate. In terms of the amount of training epochs, the same strategy was used, for example, if the trend of loss decrease of the last epoch is still high, we may consider increasing the amount of epochs. Furthermore, based on [39, 40, 41], we used trial and error strategy [42] to choose a descent batch size, hidden size and the latent space dimensions.

3.3.6 Transfer Learning

To see how the model is transferable for another animal species, the encoder of trained horse VAE model is extracted and a classification layer is applied on the top of the extracted encoder. The structure is the same as 3.7 and then, we investigated performance on goat labeled data with "Frozen Encoder" and "Not Frozen Encoder" (if it is with "Not Frozen Encoder", then the parameters of the whole network can be updated and if it is "Frozen Encoder", only the parameters of the last layer(the classification layer) can be updated). Besides, we also investigated how representative the data of one individual horse to six horses.

3.3.7 Model Training Time

As different models have different computation complexities, it might be the case that the model has better performance while it takes more time to train. As a result, in this thesis, the timing of VAE training and classification training are measured so that it can give us an indication of the computation complexities of these models.

Chapter 4

Experiment Setup

In the previous chapter, the system overview and working methodology structures are discussed, in this chapter, we discuss the details of the experiment setup.

4.1 Datasets and Preprocessing

In this thesis, we used two datasets which are the horse dataset [3] and the goat dataset [4]. These two datasets are both open-access, and they comprise a considerable amount of data.

4.1.1 Horse Dataset

We use a publicly available horse movement dataset, collected by Kamminga et al. [3]. The data was collected over 7 days, and the complete horse dataset consists of 18 individual horse subjects. The 3-axial accelerometer and gyroscope sensors were sampled at 100 Hz and the 3-axis magnetometer was sampled at 12Hz. One single sensor node was attached to the neck of the horses using a collar fabricated from a hook and loop fastener. In this thesis, in order to make the results comparable with the state-of-art [2], we utilized only the data of 3-axis accelerometer and we used the same data preprocessing methodology where we exploited 6 individual horse subjects (which are "Galoway", "Happy", "Patron", "Zafir", "Driekus" and "Bacardi") and 5 major activities (which are "Standing", "Walking", "Trotting", "Galloping" and "Grazing"). The magnitude of 3D vector (l^2-norm) of the accelerometer data is taken. Then, the time-series data were segmented with a window size of 2 seconds and 50% overlap. As a result, the input is a low dimensional (1 x n) vector where n is the window size (in this case, n is 200 as the accelerometers are sampled with a sampling rate of 100HZ for horse datasets). The activity distribution of horse movement data is illustrated in Table 4.1.

Activity\Subject	1(Galoway)	2(Happy)	3(Patron)	4(Zafir)	5(Driekus)	6(Bacardi)
Galloping	1043	696	714	835	323	328
Trotting	6423	7038	3402	3559	2673	1981
Walking	11055	9642	5538	5239	4294	1677
Grazing	4331	5063	1951	1091	2496	1116
Standing	1750	1186	1244	347	341	245
Total	24602	23625	12849	11071	10127	5347

Table 4.1: Activity distribution of horse dataset [3]

4.1.2 Goat Dataset

We also use a publicly available goat movement dataset, collected by Kamminga et al. [4]. The data was collected over 5 days and the complete goat dataset consists of 5 individual goat subjects.

A collar with six sensor nodes was attached to the goat's neck with the 3-axial accelerometer, gyroscope, and magnetometer sensors sampled at 100 Hz. In this thesis, in order to make the results comparable with the state-of-art [2], we utilized the data from six sensors attached around the goats' neck and we only utilized the data of 3-axis accelerometer sensors. We exploited 5 individual goats subjects and 5 major activities (which are "Standing", "Walking", "Trotting", "Galloping" and "Grazing"). Identical to the preprocessing of the horse dataset, the magnitude of 3D vector (l^2-norm) of the accelerometer data is taken. Then, the time-series data were segmented with a window size of 2 seconds and 50% overlap. As a result, the input is a low dimensional (1 x n) vector where n is the window size (in this case, n is 200). The activity distribution of goat movement data is illustrated in Table 4.2.

Activity\Subject	1	2	3	4	5
Galloping	88	497	56	40	115
Trotting	100	526	120	154	145
Walking	4526	4737	2381	3339	1015
Grazing	12973	3811	8129	6781	152
Standing	9879	10597	7275	3415	3199
Total	27566	20168	17961	13729	4626

Table 4.2: Activity distribution of goat dataset [4]

4.1.3 Varying Amount of Data

As the main research question is to investigate to what extent can unsupervised representation learning using VAE cope with the lack of labeled data in AAR, we use different amount of labeled data for model training. Figure 4.1 illustrates how the data was divided. To make the result comparable with the state-of-art [2], the same amount of labeled data division was used. Seen

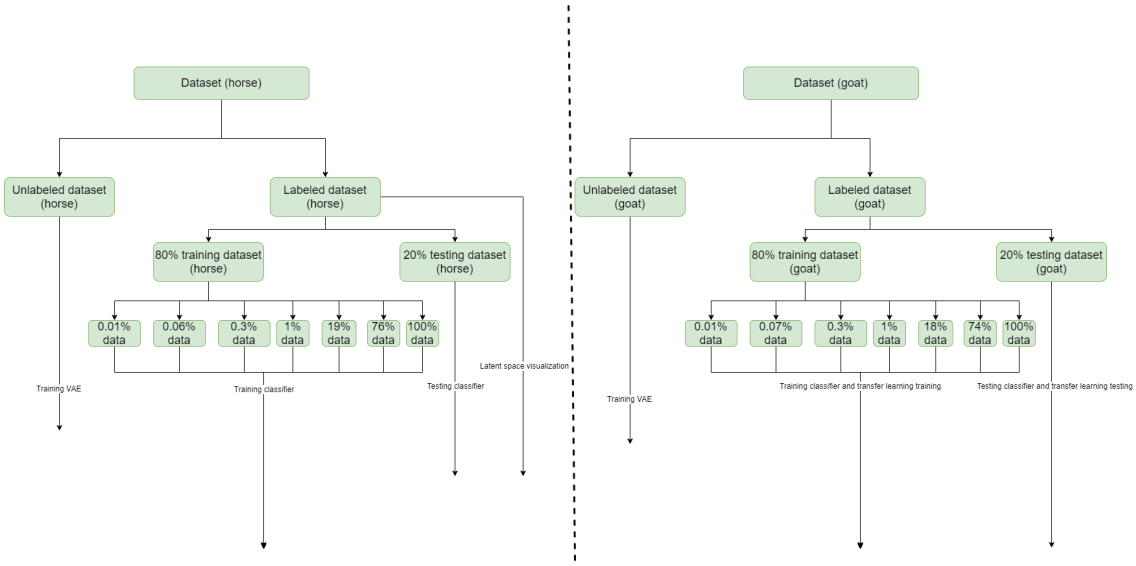

Figure 4.1: The different amount of data

from the Figure 4.1, the left part shows the horse dataset and the right part illustrates the goat dataset. In terms of the horse dataset, the unlabeled data was used to train the VAE, 100% labeled data was used for latent space visualization so that it gives us an indication of how good the VAE learned the representation. The labeled dataset was divided into training set (80%) and testing set (20%), the training set was used to train the classifier and the testing set was used

to evaluate the performance (F1 score). As for the 80% training set, it was further divided into "0.01% data", "0.06% data", "0.3% data", "1% data", "19% data", "76% data", "100% data" to investigate the performance with different amount of labeled data. Similar to the horse dataset, the goat unlabeled dataset was used to train the VAE and the goat labeled data was divided into training set (80%) and testing set(20%), the training set was used to train the classifier, besides, it was also used for transfer learning training. The testing set was used to test the performance of the classifier as well as testing the performance of transfer learning. The 80% training set was further divided into "0.01% data", "0.07% data", "0.3% data", "1% data", "18% data", "74% data", "100% data". In terms of data division, stratified random sampling was utilized to handle the imbalance of the dataset.

4.2 Unsupervised Representation Learning

The neural network training and architectures of unsupervised representation learning (VAE) are presented in this section. We used trial and error strategy [42] to choose a descent batch size, hidden size and latent space dimensions. By considering the suggestions of [39, 40, 41] which can be found in Section 2.1.4, we chose the batch size from {32, 64, 128, 256}, hidden size from {90, 110, 130} and latent space dimensions from {10, 20, 30}. The learning rate and the number of epochs were adjusted by experience and experiment. For instance, by observing the loss curve, a decent learning rate and the number of epochs can be determined.

4.2.1 VAE

For all VAE architectures, we set the batch size to 32, the hidden size to 90, the latent space dimension to 20, the learning rate for Adam optimizer to 0.0005 and the sum of KL-divergence loss and reconstruction loss (Mean Square Error) was set as the loss functions in VAE.

RNN-VAE

The architecture of RNN-VAE can be found in Figure 4.2. The "Encoder" consists of one RNN

```
==========================================================================================================================
Layer (type:depth-idx)                   Kernel Shape              Output Shape              Param #
==========================================================================================================================
VAE                                      --                        --                        --
├──Encoder: 1-1                            --                        [32, 20]                  --
│     └──RNN: 2-1                          --                        [200, 32, 90]             8,370
│     └──Linear: 2-2                       [90, 20]                  [32, 20]                  1,820
│     └──Linear: 2-3                       [90, 20]                  [32, 20]                  1,820
├──Decoder: 1-2                            --                        [200, 32, 1]              --
│     └──Linear: 2-4                       [20, 90]                  [32, 90]                  1,890
│     └──RNN: 2-5                          --                        [200, 32, 90]             8,370
│     └──Linear: 2-6                       [90, 1]                   [200, 32, 1]              91
==========================================================================================================================
Total params: 22,361
Trainable params: 22,361
Non-trainable params: 0
Total mult-adds (M): 107.33
==========================================================================================================================
```

Figure 4.2: The architecture of RNN-VAE

layer and two linear layers, the output of the RNN layer (the hidden state of the last cell) is mapped to μ and σ (which can be found in Figure 3.4 in Section 3.3.2) by the two linear layer respectively. Then, the latent vector is calculated by Equation 2.6 in Section 2.1.2. The calculated latent vector is then fed into the "Decoder" where first, it is mapped back to the vector of hidden size and this vector is the hidden state input of the decoder RNN (which is h_0' in Figure 3.4 in Section 3.3.2). The last linear layer maps the output of RNN to the final reconstruction.

LSTM-VAE

As can be seen in Figure 4.3, architecture of LSTM-VAE is similar to RNN-VAE. The only difference is that in "Encoder" and "Decoder", the LSTM layer is used instead of RNN layer. By comparing LSTM-VAE and RNN-VAE, it can be found that number of total parameters to be updated during training of LSTM-VAE is approximately 3.25 times higher than that of RNN-VAE and the total number of calculation operations (indicated by "Total mult-adds (M)" in Figure 4.3) of LSTM-VAE is around 3.99 times higher than that of RNN-VAE.

```
==========================================================================================================================
Layer (type:depth-idx)                   Kernel Shape              Output Shape              Param #
==========================================================================================================================
VAE                                      --                        --                        --
├─Encoder: 1-1                           --                        [32, 20]                  --
│    └─LSTM: 2-1                         --                        [200, 32, 90]             33,480
│    └─Linear: 2-2                       [90, 20]                  [32, 20]                  1,820
│    └─Linear: 2-3                       [90, 20]                  [32, 20]                  1,820
├─Decoder: 1-2                           --                        [200, 32, 1]              --
│    └─Linear: 2-4                       [20, 90]                  [32, 90]                  1,890
│    └─LSTM: 2-5                         --                        [200, 32, 90]             33,480
│    └─Linear: 2-6                       [90, 1]                   [200, 32, 1]              91
==========================================================================================================================
Total params: 72,581
Trainable params: 72,581
Non-trainable params: 0
Total mult-adds (M): 428.74
==========================================================================================================================
```

Figure 4.3: The architecture of LSTM-VAE

CNN-VAE

Figure 4.4 illustrates architecture of the CNN-VAE. For each convolution layer, a Rectified Linear Unit (ReLU) layer is followed to increase the non-linearity and then followed by a Max Pooling layer which reduces the parameters and the amount of computation so that the training can be faster. The kernel size of 1x9 was chosen for all of the convolution layers. "Encoder" has two convolution layers where for the first convolution layer, the input channel is 1 and output channel is 32 and for the second convolution layer, the input channel is 32 and the output channel is 64. After that, a linear layer is map the output to the vector of hidden size and then, the output is mapped to μ and σ as can be seen in Figure 3.6 in Section 2.2. Then, the latent vector is calculated by Equation 2.6 in Section 2.1.2. The "Decoder" is sort of symmetric to the "Encoder" where the calculated latent vector is fed into the "Decoder" and it is mapped back to the vector of hidden size. After two "symmetric" convolution layers, a linear layer maps the output to the final reconstruction.

```
==========================================================================================================================
Layer (type:depth-idx)                   Kernel Shape              Output Shape              Param #
==========================================================================================================================
VAE                                      --                        --                        --
├─Encoder: 1-1                           --                        [32, 20]                  --
│    └─Sequential: 2-1                   --                        [32, 32, 1, 96]           --
│    │    └─Conv2d: 3-1                  [1, 32, 1, 9]             [32, 32, 1, 192]          320
│    │    └─ReLU: 3-2                    --                        [32, 32, 1, 192]          --
│    │    └─MaxPool2d: 3-3               --                        [32, 32, 1, 96]           --
│    └─Sequential: 2-2                   --                        [32, 64, 1, 44]           --
│    │    └─Conv2d: 3-4                  [32, 64, 1, 9]            [32, 64, 1, 88]           18,496
│    │    └─ReLU: 3-5                    --                        [32, 64, 1, 88]           --
│    │    └─MaxPool2d: 3-6               --                        [32, 64, 1, 44]           --
│    └─Sequential: 2-3                   --                        [32, 90]                  --
│    │    └─Linear: 3-7                  [2816, 90]                [32, 90]                  253,530
│    │    └─ReLU: 3-8                    --                        [32, 90]                  --
│    └─Linear: 2-4                       [90, 20]                  [32, 20]                  1,820
│    └─Linear: 2-5                       [90, 20]                  [32, 20]                  1,820
├─Decoder: 1-2                           --                        [1, 200]                  --
│    └─Sequential: 2-6                   --                        [32, 32, 1, 18]           --
│    │    └─Linear: 3-9                  [20, 90]                  [32, 90]                  1,890
│    │    └─Linear: 3-10                 [90, 2816]                [32, 2816]                256,256
│    │    └─ReLU: 3-11                   --                        [32, 2816]                --
│    │    └─UnFlatten: 3-12              --                        [32, 64, 1, 44]           --
│    │    └─Conv2d: 3-13                 [64, 32, 1, 9]            [32, 32, 1, 36]           18,464
│    │    └─ReLU: 3-14                   --                        [32, 32, 1, 36]           --
│    │    └─MaxPool2d: 3-15              --                        [32, 32, 1, 18]           --
│    └─Sequential: 2-7                   --                        [32, 1, 1, 5]             --
│    │    └─Conv2d: 3-16                 [32, 1, 1, 9]             [32, 1, 1, 10]            289
│    │    └─ReLU: 3-17                   --                        [32, 1, 1, 10]            --
│    │    └─MaxPool2d: 3-18              --                        [32, 1, 1, 5]             --
│    └─Sequential: 2-8                   --                        [1, 200]                  --
│    │    └─Linear: 3-19                 [160, 200]                [1, 200]                  32,200
│    │    └─ReLU: 3-20                   --                        [1, 200]                  --
==========================================================================================================================
Total params: 585,085
Trainable params: 585,085
Non-trainable params: 0
Total mult-adds (M): 91.94
==========================================================================================================================
```

Figure 4.4: The architecture of CNN-VAE

4.3 Classification

As can be seen in Figure 3.7 in Section 3.3.4, the encoder of VAE is extracted and meanwhile, a linear classification layer was attached on the top of the encoder. We also did some extra experiments to investigate the characteristics of the model:

- We investigated the performance difference between that of frozen encoder and not-frozen encoder.
- We fed the input data directly to the single classification layer to inspect whether the data can be easily classified.
- We used the VAE encoder architecture with parameters random initialized to do the supervised learning from scratch to see how unsupervised representation learning improves the performance especially when the amount of labeled data is low.

In terms of the classification neural networks training, we set the batch size to 32 which is the same as VAE training. We found that the learning rate and the number of epochs of VAE training are not suitable for classification neural networks training, as a result, we adjusted these two hyper-parameters based on the experience and experiment. The number of epochs was chosen to 50, the learning rate of that with frozen encoder was set to 0.1, that of not-frozen encoder and supervised learning from scratch were set to 0.001 and that of pure classification was set to 0.0005. Adam optimizer was used for all of the neural network training. The loss function differs from different classifiers used where Softmax uses Cross-entropy loss and SVM uses hinge-loss.

4.3.1 VAE Encoder with Classification Layer

As described above, the "Encoder" from VAE was extracted and a linear layer was attached on the top of the "Encoder". The input size of the classification layer is always 20 as the latent space vector dimension is 20 and the output size of the classification layer is always 5 because there are 5 activity classes to be recognised. With-Frozen-Encoder, With-Not-Frozen-Encoder and supervised learning from scratch (with VAE encoder parameters random initialized) share the same architecture and the same number of total parameters while the number of trainable parameters are different. With-Frozen-Encoder has less number of trainable parameters than the other two because the encoder is frozen where the parameters of encoder cannot be updated. With-Not-Frozen-Encoder and supervised learning from scratch have the same amount of trainable parameters which is the number of total parameters. As for all of the encoders of VAE, the last two layers are always the same which are two linear layers take the input size of 90 (hidden size) and output vector size of 20 (latent space vector size). These two layers map the output from last stage to μ and σ and the latent space vector is calculated by Formula 2.6. Then, the classification is on the latent space vector.

RNN-Encoder + Classifier:

Figure 4.5 shows the architecture of RNN-Encoder with classifier where the encoder of RNN-VAE is extracted and a linear classification layer is implemented on the top of the encoder. As can be

Layer (type:depth-idx)	Kernel Shape	Output Shape	Param #
VAE	--	--	--
├─Encoder: 1-1	--	[32, 20]	--
│ └─RNN: 2-1	--	[200, 32, 90]	8,370
│ └─Linear: 2-2	[90, 20]	[32, 20]	1,820
│ └─Linear: 2-3	[90, 20]	[32, 20]	1,820
├─Classifier: 1-2	--	[32, 5]	--
│ └─Linear: 2-4	[20, 5]	[32, 5]	105

Figure 4.5: The architecture of RNN-Encoder + Classifier

seen in Figure 4.6, it indicates the number of RNN-Encoder + classifier parameters of different setup. It can be found that the number of trainable parameters With-Frozen-Encoder is much less than the other two because as for With-Frozen-Encoder, only the parameters of the last layer (the linear classification layer) are trainable and from Figure 4.5, the number of parameters is 105 which is consistent to the number of "Trainable params" indicated in the Figure 4.6.

With-Frozen-Encoder/ Supervised Learning from scratch	With-Not-Frozen-Encoder
Total params: 12,115	Total params: 12,115
Trainable params: 12,115	Trainable params: 105
Non-trainable params: 0	Non-trainable params: 12,010
Total mult-adds (M): 53.69	Total mult-adds (M): 53.69

Figure 4.6: The number of trainable parameters difference among With-Frozen-Encoder, With-Not-Frozen-Encoder and supervised learning from scratch of RNN-Encoder + classifier

LSTM-Encoder + Classifier:

Figure 4.7 shows the architecture of LSTM-Encoder with classifier where the encoder of LSTM-VAE is extracted and a linear classification layer is implemented on the top of the encoder. As shown in Figure 4.8, it indicates the number of parameters of LSTM-Encoder + classifier. The

```
==========================================================================================================
Layer (type:depth-idx)                   Kernel Shape              Output Shape              Param #
==========================================================================================================
VAE                                      --                        --                        --
├─Encoder: 1-1                             --                        [32, 20]                  --
│    └─LSTM: 2-1                           --                        [200, 32, 90]             (33,480)
│    └─Linear: 2-2                         [90, 20]                  [32, 20]                  (1,820)
│    └─Linear: 2-3                         [90, 20]                  [32, 20]                  (1,820)
├─Classifier: 1-2                          --                        [32, 5]                   --
│    └─Linear: 2-4                         [20, 5]                   [32, 5]                   105
==========================================================================================================
```

Figure 4.7: The architecture of LSTM-Encoder + Classifier

number of trainable parameters of With-Frozen-Encoder is the same as that of RNN-Encoder + classifier as the architectures of the linear classification layer are the same which both take input size of 20 and output size of 5.

```
With-Frozen-Encoder/
Supervised Learning from scratch
============================
Total params: 37,225
Trainable params: 37,225
Non-trainable params: 0
Total mult-adds (M): 214.39
============================
```

```
With-Not-Frozen-Encoder
============================
Total params: 37,225
Trainable params: 105
Non-trainable params: 37,120
Total mult-adds (M): 214.39
============================
```

Figure 4.8: The number of trainable parameters difference among With-Frozen-Encoder, With-Not-Frozen-Encoder and supervised learning from scratch of LSTM encoder + classifier

CNN-Encoder + Classifier:

Figure 4.9 shows the architecture of CNN-Encoder with classifier where the encoder of CNN-VAE is extracted and a linear classification layer is implemented on the top of the encoder. As shown in Figure 4.10, it indicates the number of parameters. The number of trainable parameters of With-Frozen-Encoder is the same as that of the other two as they share the same architectures of classification layer.

```
==========================================================================================================
Layer (type:depth-idx)                   Kernel Shape              Output Shape              Param #
==========================================================================================================
VAE                                      --                        --                        --
├─Encoder: 1-1                             --                        [32, 20]                  --
│    └─Sequential: 2-1                     --                        [32, 32, 1, 96]           --
│    │    └─Conv2d: 3-1                    [1, 32, 1, 9]             [32, 32, 1, 192]          (320)
│    │    └─ReLU: 3-2                      --                        [32, 32, 1, 192]          --
│    │    └─MaxPool2d: 3-3                 --                        [32, 32, 1, 96]           --
│    └─Sequential: 2-2                     --                        [32, 64, 1, 44]           --
│    │    └─Conv2d: 3-4                    [32, 64, 1, 9]            [32, 64, 1, 88]           (18,496)
│    │    └─ReLU: 3-5                      --                        [32, 64, 1, 88]           --
│    │    └─MaxPool2d: 3-6                 --                        [32, 64, 1, 44]           --
│    └─Sequential: 2-3                     --                        [32, 90]                  --
│    │    └─Linear: 3-7                    [2816, 90]                [32, 90]                  (253,530)
│    │    └─ReLU: 3-8                      --                        [32, 90]                  --
│    └─Linear: 2-4                         [90, 20]                  [32, 20]                  (1,820)
│    └─Linear: 2-5                         [90, 20]                  [32, 20]                  (1,820)
├─Classifier: 1-2                          --                        [32, 5]                   --
│    └─Linear: 2-6                         [20, 5]                   [32, 5]                   105
==========================================================================================================
```

Figure 4.9: The architecture of CNN-Encoder + Classifier

With-Frozen-Encoder/
Supervised Learning from scratch

```
==========================
Total params: 276,091
Trainable params: 276,091
Non-trainable params: 0
Total mult-adds (M): 62.28
==========================
```

With-Not-Frozen-Encoder

```
==============================
Total params: 276,091
Trainable params: 105
Non-trainable params: 275,986
Total mult-adds (M): 62.28
==============================
```

Figure 4.10: The number of trainable parameters difference among With-Frozen-Encoder, With-Not-Frozen-Encoder and supervised learning from scratch of CNN encoder + classifier

Pure Classification

To investigate whether the animal movement data can be easily classified, the data was directly fed into the linear classification layer. The difference among the linear classification layers described above is the input size. The input size is the number of input data point which is 200 instead of the hidden size 90 used above where the architecture can be found in Figure 4.11. The network is simple which contains only one linear layer.

```
==========================================================================================================================
Layer (type:depth-idx)                   Kernel Shape              Output Shape              Param #
==========================================================================================================================
Classifier                               --                        --                        --
├─Linear: 1-1                              [200, 5]                  [32, 5]                   1,005
==========================================================================================================================
Total params: 1,005
Trainable params: 1,005
Non-trainable params: 0
Total mult-adds (M): 0.03
==========================================================================================================================
```

Figure 4.11: The architecture of pure classifier

Chapter 5

Performance Evaluation

In this chapter, we present the evaluation metrics used in this thesis, the experiment procedures, their results, the results discussion and the experiment environments when conducting these experiments

5.1 Evaluation Metrics

5.1.1 Cross Validation

Cross-Validation is a technique to evaluate the predictive model performance by partitioning the original dataset into training set and testing set. As described above in Section 4.1, when partitioning the original labeled dataset to training set and testing set and further partitioning the training set to different amount of labeled data training set, the stratified random sampling was applied which means for each experiment of classification training and evaluating, the training set and testing set were random sampled and this increased the reliability of the model. Furthermore, to make the results more reliable, for each experiment, we repeatedly ran it for 10 times and took the mean value and the standard deviation.

5.1.2 F1 Score

We used F1 score as the evaluation of our model where the formula is:

$$\text{F1} = 2 \times \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.1}$$

Precision shows how precise the model is out of the predicted positive and it is a good measure when the costs of False Positive is high, for example, detecting the email spam. The precision can be calculated by:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{5.2}$$

Recall determines the ratio of True Positive out of the Actual Positive and it is a good measure when there is a high cost related to False Negative, for instance, in fraud detection or sick patient detection. The Recall equation is:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{5.3}$$

5.2 Experiment Environments

The experiments were implemented in PyTorch and the experiment environments can be found in Table 5.1.

Processer	AMD Ryzen 5 2600X Six-Core Processor 3.60GHz
RAM	16.0GB
OS	Windows 10 Pro 64-bit 20H2 19042.1110
GPU	NVIDIA GeForce GTX 1660
CUDA version	11.4.56
Conda version	4.10.1
Python version	3.8.8
Jupyter client version	6.1.12
PyTorch version	1.8.1
Matplotlib version	3.3.4
Numpy version	1.20.1
Scikit-learn version	0.24.1

Table 5.1: Experiment environments

5.3 Performance Evaluation of Unsupervised Representation Learning

There are several sets of experiments where the results can help us answer a series of questions.

5.3.1 Baseline

The baseline experiments are shown in Figure 5.1, the difference between top part and bottom part is that top part used all six horses data and bottom part used all five goats data. In terms of top part, RNN-VAE/LSTM-VAE/CNN-VAE was trained with six horses unlabeled data. After VAE training, the VAE encoder was taken and a linear classifier was attached on the top of the encoder. The extracted VAE encoder was frozen while training, which means the parameters of the encoder cannot be updated, only the parameters of the last classification layer can be updated and this reduced the amount of computation dramatically. The VAE encoder + classifier was trained with different amount of six horses labeled training dataset and the performance was evaluated with the six horses labeled testing dataset. The detailed data partitioning can be found in Section 4.1. The bottom part is similar to the top part while the data used was five goats data instead of six horses data. The results of baseline were to be compared with the state-of-art [2].

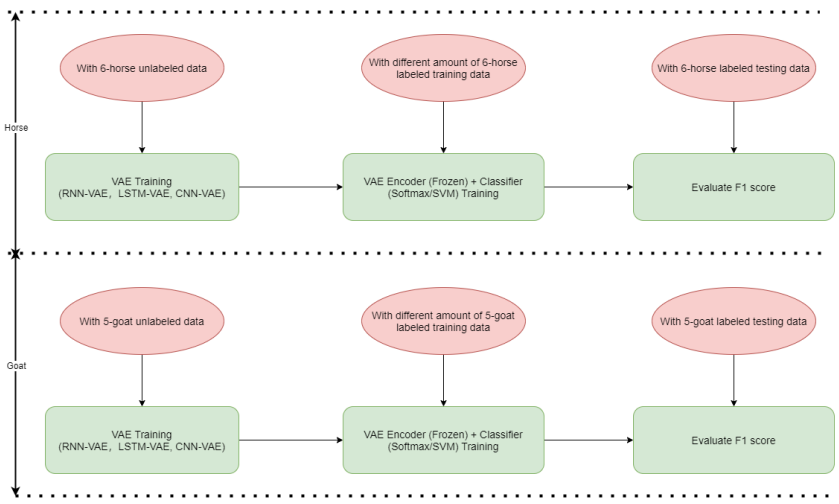

Figure 5.1: The baseline experiment

Horse Baseline

As can be seen in Figure 5.2, it shows the results of classification of horse baseline. The different colors indicate the different type of VAE model. Deeper color implies that the Softmax was used and lighter color indicates that SVM was used. In the notations, "NN type + classifier" indicates the type of VAE and classifier. For example, "LSTM+Softmax" indicates LSTM-VAE was trained and then, the classification NN consists of the trained LSTM-VAE encoder and a linear Softmax classification layer where the architecture can be found in Figure 4.7. The F1 score is the evaluation of the classification NN on the test dataset and the performance shown in the graph is the mean value of ten experiments with an error bar which is the standard deviation.

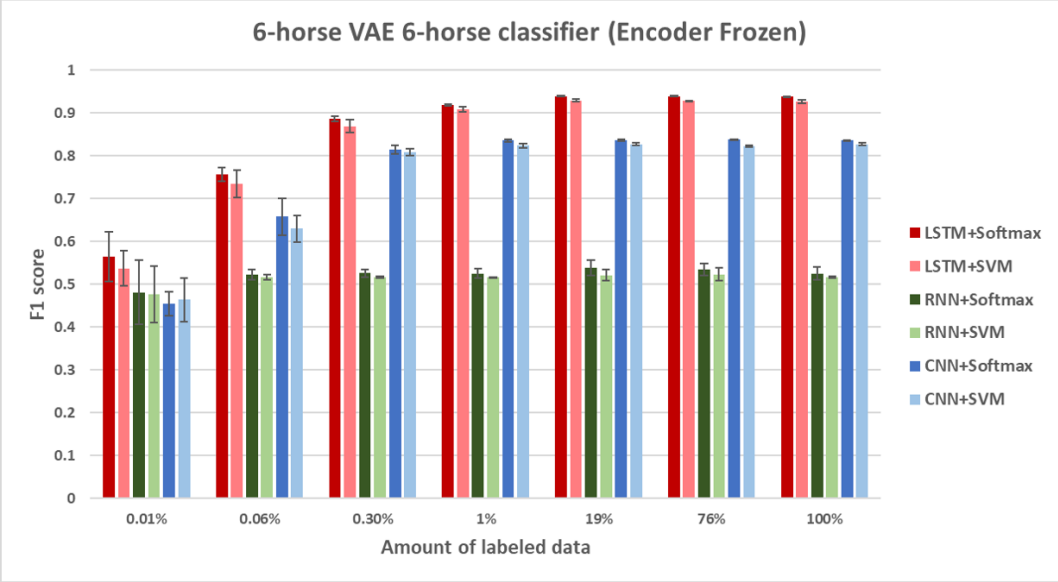

Figure 5.2: The results of horse baseline

It can be found that the LSTM-VAE learned best and the next is CNN-VAE. RNN-VAE had a bad performance because RNN-VAE suffered from short-term memory problem which means it cannot handle long input sequence and it would further cause the gradient vanishing problems. In our case, the input was 200 data samples which illustrates that only the representations of the last few data samples of the 2s window were learned and this resulted in the bad performance of RNN-VAE.

The performances of different classifiers were close, where generally, the Softmax performed slightly higher than that of SVM because SVM used hinge loss function where the margin was set to 1 so that the minimal loss was 1 and Softmax used cross entropy loss where the loss can be less than 1. Because the encoder was frozen, only the parameters of the classification layer can be updated which makes the difference tiny.

In terms of the best performed model "LSTM+Softmax", with the least amount of training data (0.01%), the F1 score reaches around 0.57. For LSTM-VAE and CNN-VAE, with the amount of labeled training data increasing, the performance increases while when it reaches 1% and higher, the performance reaches the top where for LSTM-VAE, it is approximately 0.93 and for CNN-VAE, it is around 0.84. The error bar is shrinking with the increasing amount of labeled data until the performance reaches the top (higher than 1%) for "LSTM" and "CNN" models.

Latent Space Visualization

Figure 5.3 shows the six-horse latent space visualization of LSTM-VAE. 100% horse labeled data was used for labeling the data points on the graph. The visualization let us be able to check how

good the VAE has learned by visualizing how separable the tasks are.

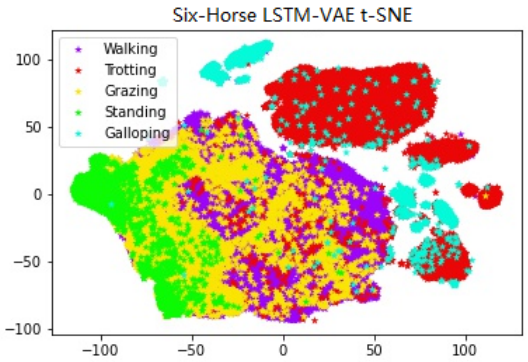

Figure 5.3: The latent space visualization of six-horse movement data

It can be found that the model has learned useful representations where we can clearly see the clustering of different activity classes in the graph. Seen from left to right, the type of activity becomes more active and there are overlapping activity distribution such as "Standing-Grazing", "Grazing-Walking", etc. The reason could be that the two activities have a similar extent of activity level, so, sometimes, it gets confused. The activity labeling order is "Walking", "Trotting", "Grazing", "Standing" and "Galloping", so, the activity labeled later covers the activities labeled earlier, for example, "Galloping" will never be covered by other activities while "Walking" can be covered by any other activities.

Comparison with the State-of-art [2]

Figure 5.4 shows the comparison of horse baseline with the state-of-art, on the left, it is our results which is the same as in Figure 5.2 and on the right, it is the result of the state-of-art. The difference is the unsupervised representation learning method used and the state-of-art also used the handcrafted feature engineering method. The state-of-art uses SVM as classifier and there was no fine-tuning (the encoder is frozen), so as for our result, we used the result of "LSTM+SVM" as LSTM-VAE model is the best performed model and we compare the results using the same classifier. As the state-of-art did not provide the exact result number, we compared them in this way. The two brown auxiliary lines were the indications of F1 score 0 and 1. The red auxiliary lines were the indications of how our "LSTM+SVM" results compared with the state-of-art.

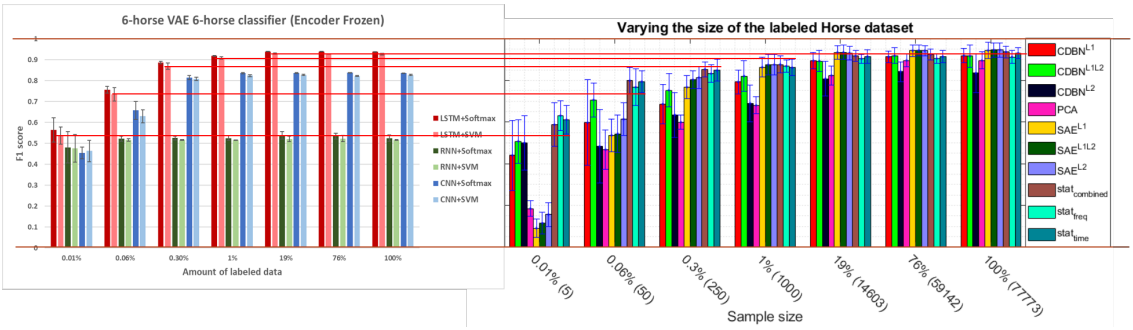

Figure 5.4: The horse baseline comparison with the state-of-art [2]

It can be found that for 0.01% and 0.06% amount of data, our model outperforms all unsupervised representation learning methods but underperforms the handcrafted feature engineering

methods of the state-of-art. As for 0.3% and 0.1% amount of data, our model beats both unsupervised representation learning and handcrafted feature engineering and for 19% and higher, they have nearly the same performance. Furthermore, the error bar of our model tends to be much less than that of the state-of-art which means our model is more robust than the state-of-art. From Formula 2.6 in Section 2.1.2, the latent vector z is sampled based on a probability distribution and this increases the robustness of the model, as a result, especially with a large amount of labeled data, the results of our model tend to have a more robust result.

Therefore, although the performance of a large amount of labeled data (19% and higher) of our model is nearly the same as the state-of-art, the performance of a low amount of labeled data (1% and less) outperforms the unsupervised representation learning methods of the state-of-art and our model increases the robustness dramatically.

Goat Baseline

The results of classification of goat baseline can be found in Figure 5.5, the colour indications are the same as that of horse baseline, where red, green and blue indicates LSTM-VAE, RNN-VAE and CNN-VAE separately and the deeper color illustrates the Softmax classifier while the lighter color illustrates the SVM classifier. Similar to the horse baseline, it can be found that LSTM-VAE has learned best and the next is CNN-VAE while for goat baseline, the performance difference between LSTM-VAE and CNN-VAE is smaller than that of horse baseline. The same as horse baseline, the RNN-VAE failed to learn useful representations due to the long sequence input problem. In terms of the same type of VAE model, the performances of two classifiers are close to each other while Softmax performs a little bit better than that of SVM.

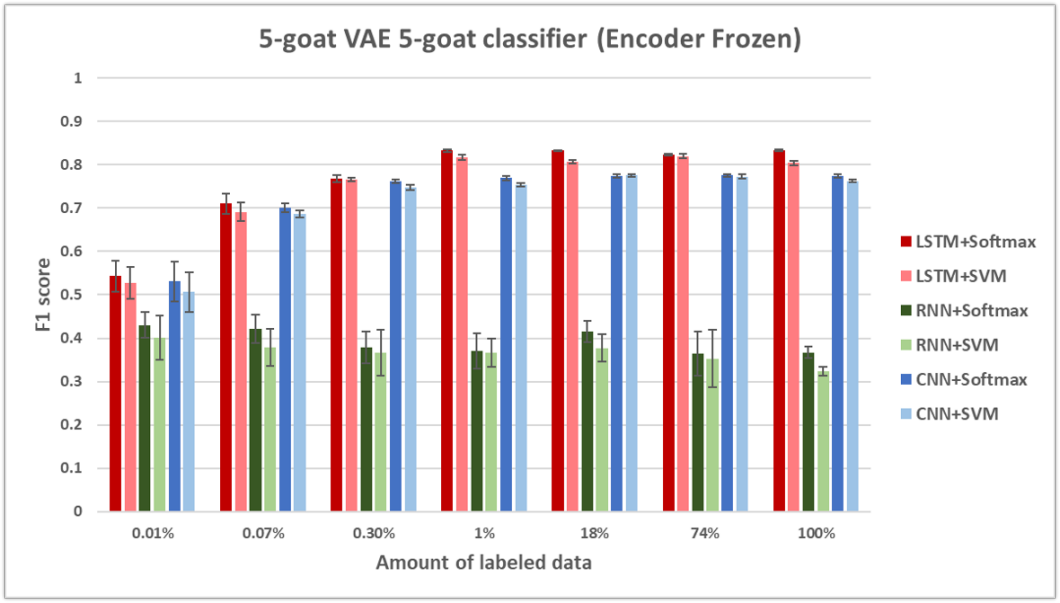

Figure 5.5: The results of goat baseline

If we look at the best performed model "LSTM+Softmax", with the least amount of training data (0.01%), the F1 score reaches around 0.54. For LSTM-VAE and CNN-VAE, with the amount of labeled training data increasing, the performance increases while when it reaches 1% and higher, the performance reaches the top where for LSTM-VAE, it is approximately 0.83 and for CNN-VAE, it is around 0.77. Meanwhile, the error bar is shrinking with the increasing amount of labeled data until the performance reaches the top (higher than 1%) for "LSTM" and "CNN" models.

It can be found that the performance of goat baseline is worse than that of horse baseline for both LSTM-VAE and CNN-VAE. There are multiple reasons that may result in it such as the

data quality [71], noise [72], imbalance dataset [73], etc.

Latent Space Visualization

Figure 5.6 shows the five-goat latent space visualization of LSTM-VAE. 100% goat labeled data was used for labeling the data points on the graph. The visualization let us be able to check how good the VAE has learned by visualizing how separable the tasks are.

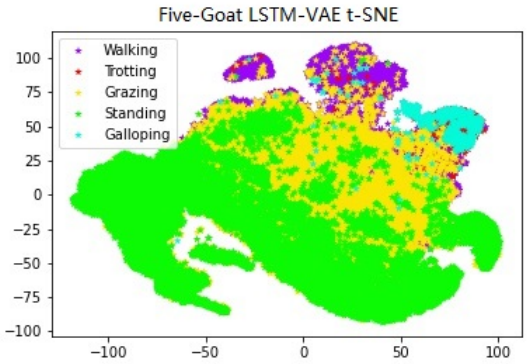

Figure 5.6: The latent space visualization of five-goat movement data

The clustering of different activities in the graph shows that the model has learned useful representations. Similar to the horse latent space visualization shown in Figure 5.3, seen from left bottom to right top, the type of activity becomes more active and there are overlapping activity classes such as "Standing-Grazing", "Grazing-Walking", etc. The reason could be that the two activities have a similar extent of activity level, so, sometimes, it gets confused. It can be found in the goat latent space visualization in Figure 5.6, the tiny amount of "Trotting" samples (red points) is because on the one hand, the amount of activity "Trotting" takes up a small proportion in the activity distribution, which can be found in Table 4.2 and on the other hand, "Trotting" samples are secondly labeled, so, in principle, it can be covered by "Grazing", "Standing" and "Galloping", as a result, by observing the graph, we can found that some "Trotting" samples are covered by "Grazing" samples (yellow points) and "Galloping" samples (blue points).

Comparison with the State-of-art [2]

Similar to horse baseline above, we compare the results of our goat baseline with the state-of-art. Figure 5.7 shows the comparison of goat baseline with the state-of-art, on the left, it is our results and on the right, it is the result of the state-of-art. The difference is the unsupervised representation learning method used while the state-of-art also used the handcrafted feature engineering method. The state-of-art uses SVM as classifier and there was no fine-tuning (the encoder is frozen), so as for our result, we used the result of "LSTM+SVM" as LSTM-VAE model is the best performed model and we compare the results using the same classifier. The same as the comparison of horse above, we compare them by drawing two brown auxiliary lines which are the indications of F1 score 0 and 1. The red auxiliary lines are the indications of how our "LSTM+SVM" results compared with the state-of-art.

It can be found that for the amount of data below 1%, our model beats both unsupervised representation learning and handcrafted feature engineering of state-of-art. As for 1% and higher amount of data, they have nearly the same performance. Furthermore, the error bar of our model is much less than that of the state-of-art which means our model is more robust than the state-of-art.

Therefore, although the performance of a large amount of labeled data (1% and higher) of our model is nearly the same as the state-of-art, the performance of a low amount of labeled data

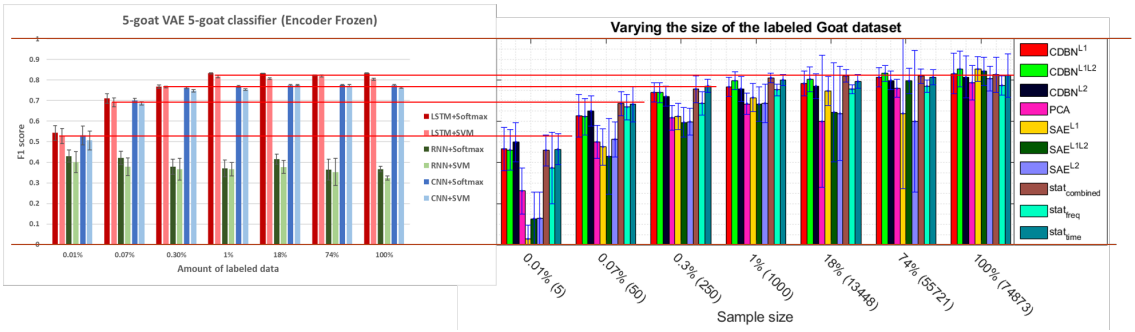

Figure 5.7: The goat baseline comparison with the state-of-art [2]

(1% and less) outperforms both unsupervised representation learning and handcrafted feature engineering methods of the state-of-art and our method increases the robustness dramatically.

Conclusions From Results of Baseline Experiments and Comparison with the state-of-art

From both horse and goat baseline results, it can be concluded that VAE model is a good candidate for the unsupervised representation learning. However, the results of RNN-VAE is poor because RNN-VAE suffers from short-term memory problem, where it cannot handle long input sequence and it would further cause the gradient vanishing problems. The best performed VAE model is "LSTM-VAE" model and the following is "CNN-VAE" model. The performance of Softmax classifier is close to that of SVM classifier while Softmax performs slightly better than that of SVM in baseline. The best performed model is "LSTM+Softmax" and the followings are "LSTM+SVM", "CNN+Softmax", "CNN+SVM". With the amount of labeled data increasing, the performance increases and the error bar shrinks. After the performance reaches top, increasing the amount of labeled training data will not increase the performance as the performance becomes stable. By comparing the results of our model with the state-of-art, our model outperforms the state-of-art with low amount of labeled training data and performs nearly the same when the amount of labeled training data is high. Furthermore, our model tends to be much more robust than the state-of-art.

5.3.2 Pure Classification

To investigate whether the animal movement data can be easily classified, the data was directly fed into a single linear classification layer. The network is simple which contains only one linear layer. Figure 5.8 shows the experiment illustration of pure classification to inspect whether the data can be distinguished easily where it only consists of a single linear classification layer.

Horse Pure Classification

As can be seen in Figure 5.9, it is the results of horse pure classification. It can be found that the horse movement data cannot be classified easily. As a result, the last layer of our classification neural network does not dominate the good performance of our horse baseline. So, the unsupervised representation learning (VAE) dominates the good performance of our model.

Goat Pure Classification

As can be seen in Figure 5.10, it is the results of goat pure classification. Similar to the horse pure classification, the goat movement data cannot be easily distinguished. As a result, the last layer of our classification neural network does not dominate the good performance of our goat

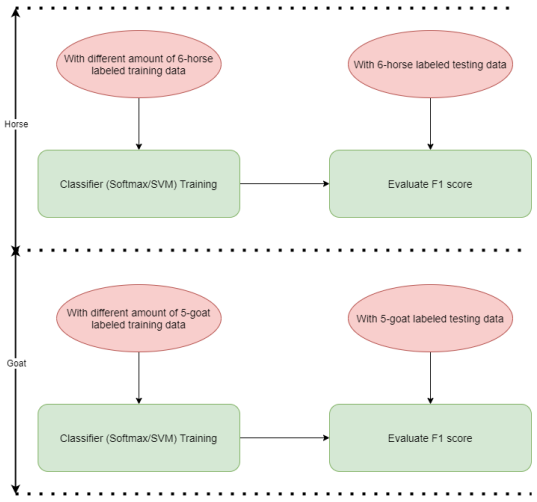

Figure 5.8: The experiment of pure classifier

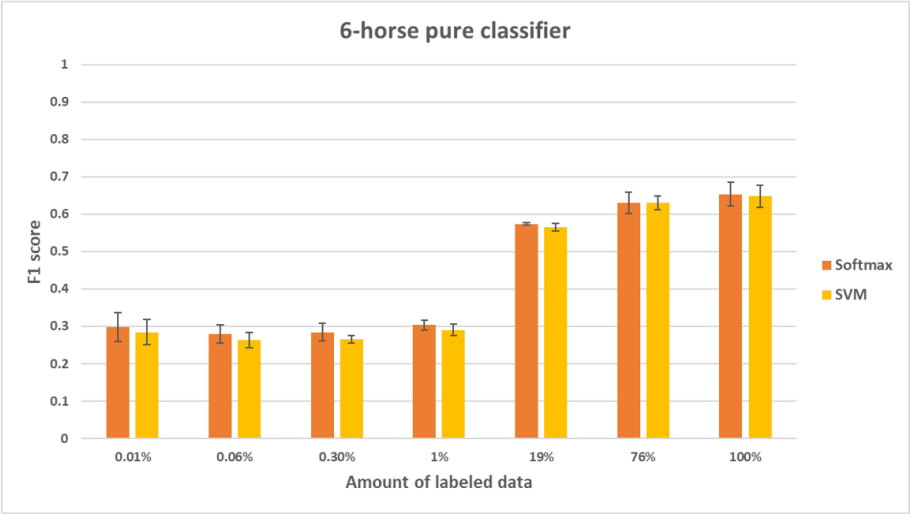

Figure 5.9: The results of horse pure classification

baseline. So, the unsupervised representation learning (VAE) dominates the good performance of our model.

Conclusions From Pure Classification

From both horse and goat pure classification results, it can be concluded that the animal data cannot be easily distinguished so that the good results of our baseline model are dominated by the unsupervised representation learning using VAE instead of the classifier.

5.3.3 Baseline VS Baseline with Encoder Not Frozen VS Supervised Learning from Scratch

We have seen that our baseline for both horse and goat movement data has gained good results and outperforms the state-of-art. Meanwhile, there are something interesting for us to further investigate such as the effect of unfreezing the encoder; how it performs if we conduct the supervised

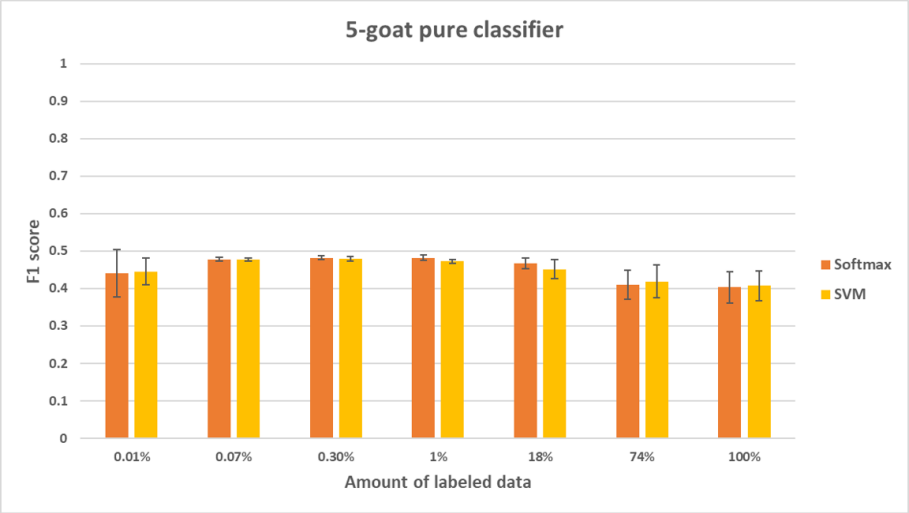

Figure 5.10: The results of goat pure classification

learning from scratch (only do the supervised learning with labeled data). In the following, we address these.

Baseline with Encoder Not Frozen

In the previous section 5.3.1, we investigated the performance of the baseline where the encoder is frozen, one way to increase the performance is to unfreeze the encoder where the parameters of encoder can be updated during the training. As a result, the difference between this section and the baseline is that during the classification neueral network training, the parameters of encoder can be updated. Figure 5.11 illustrates the experiment. The VAE model is not retained meaning that the extracted VAE encoder is the same as that of the baseline in section 5.3.1.

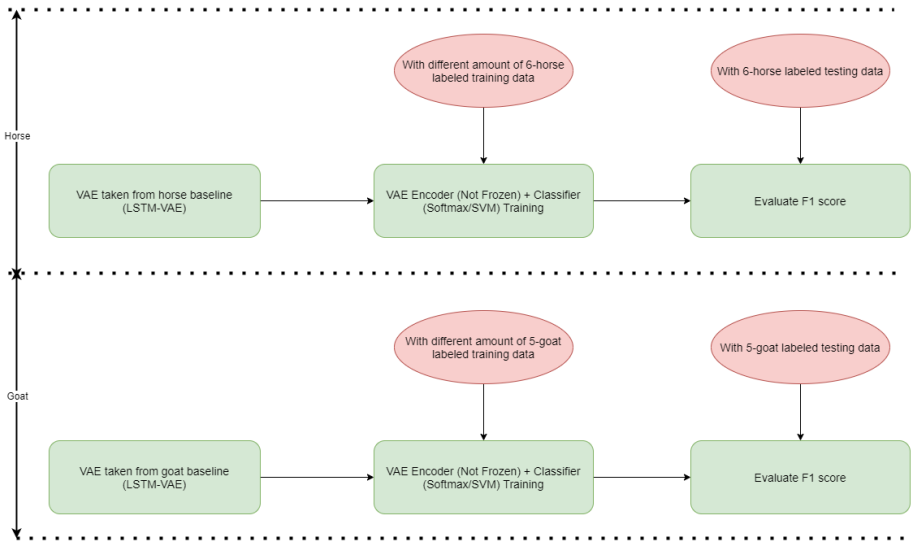

Figure 5.11: The baseline with encoder not frozen experiment

Supervised Learning from Scratch

We have seen that our baseline in Section 5.3.1 has good performance when the amount of labeled data is low. If we regard our classification neural network such as "LSTM+Softmax" as a supervised learning method and train it with labeled data from scratch. If the performance is close to the results of our baseline, then, there is no point to conduct the unsupervised representation learning. Instead, we can directly conduct the supervised learning by using the labeled data. As a result, it is interesting to investigate how the supervised learning from scratch performs.

Figure 5.12 implies the experiment illustration of supervised learning from scratch, it is aimed at discovering the performance improvement of unsupervised representation learning. The NN consists of random initialized VAE encoder and a linear classification layer. The architecture of VAE encoder is the same while the parameters have been random initialized. As a result, we consider it as the supervised learning from scratch.

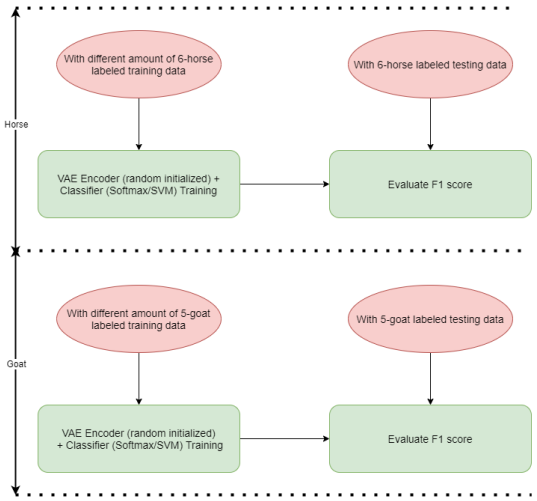

Figure 5.12: The experiment of supervised learning from scratch

Results

In the following, we illustrate our results of the LSTM-VAE model on horse and goat movement data. For the results shown in Figure 5.13 and 5.14, we mainly used two colours to indicate the different classifiers used, where "red" indicates the Softmax classifier and "blue" indicates the SVM classifier. The gradations of color indicate the different experiments, where the dark colour shows the results of baseline (noted by "Encoder Frozen"), the middle colour indicates the results of baseline with Encoder Not Frozen (noted by "Encoder Not Frozen") and the light colour illustrates the supervised learning from scratch (noted by "Learning from Scratch").

Horse

Figure 5.13 illustrates the results on horse movement data. By comparing dark red and middle red or comparing dark blue and middle blue, it can be found that generally, by unfreezing the encoder, the performance tends to be increased. For high amount of labeled data (1% and higher), the performance increase by unfreezing the encoder is stable at approximately 3% for Softmax and around 4% for SVM while for low amount of labeled data (less than 1%), the performance increase varies. By observing the error bar of "Encoder Frozen" and the other two ("Encoder Not Frozen" and "Learning from Scratch"), so, by comparing the dark red with middle and light red or comparing the dark blue with middle and light blue, it can be found that generally, the error bar is less if we freeze the encoder because by freezing the encoder, less amount of parameters are updated

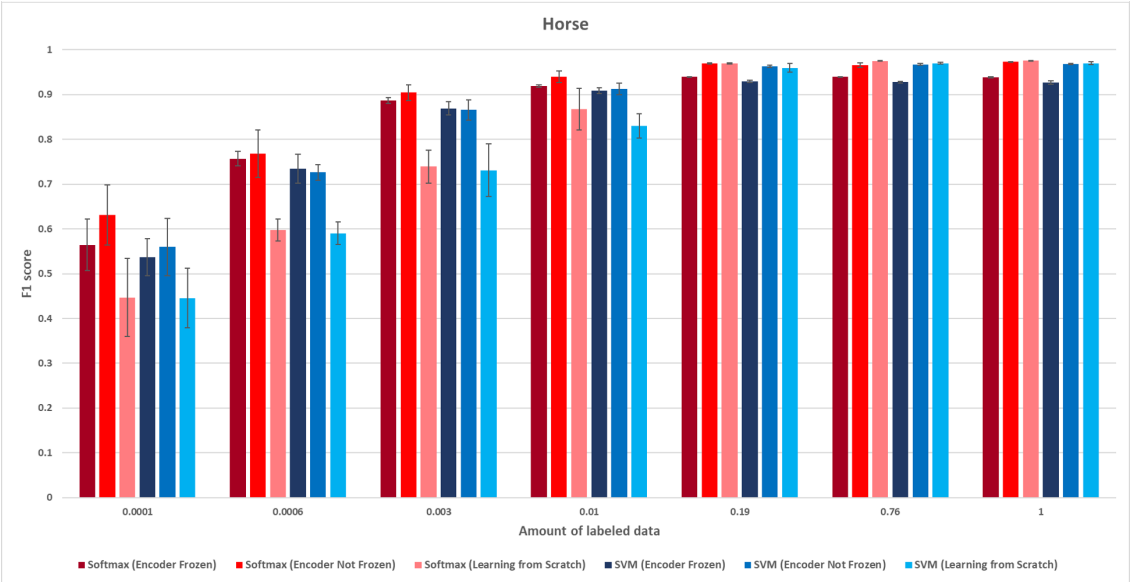

Figure 5.13: The results on horse movement data

during the training. By comparing the light red and the other two (dark red and middle red) or comparing the light blue and the other two (dark blue and middle blue), it can be found that for low amount of labeled training data (less than 19%), the supervised learning from scratch performs worse than unsupervised representation learning, however, with the amount of labeled training data increasing, the performance difference becomes less. And finally, with adequate amount of labeled training data (19% and higher), the performance of "Encoder Not Frozen" (middle red or middle blue) becomes close to the performance of supervised learning from scratch (light red or light blue). Furthermore, by comparing dark red and dark blue, middle red and middle blue or light red and light blue, it can be observed that generally the performance of Softmax classifier is slightly higher than that of SVM classifier.

Goat

Figure 5.14 shows the results on goat movement data. Similar to the results on horse movement data, we could draw the following conclusions:

- By unfreezing the encoder, the performance increases
- Generally, the error bar of "Encoder Not Frozen" and "Learning from Scratch" is larger than that of "Encoder Frozen" due to more trainable parameters
- For low amount of labeled training data (less than 19%), the performance of supervised learning from scratch is worse than that of unsupervised representation learning
- With the amount of labeled training data increasing but less 19%, the performance difference among the supervised learning from scratch and the other two becomes less
- The performance of supervised learning from scratch is close to the performance of "Encoder Not Frozen" with high amount of labeled training data (19% and higher)
- Generally, the Softmax classifier performs better than SVM classifier

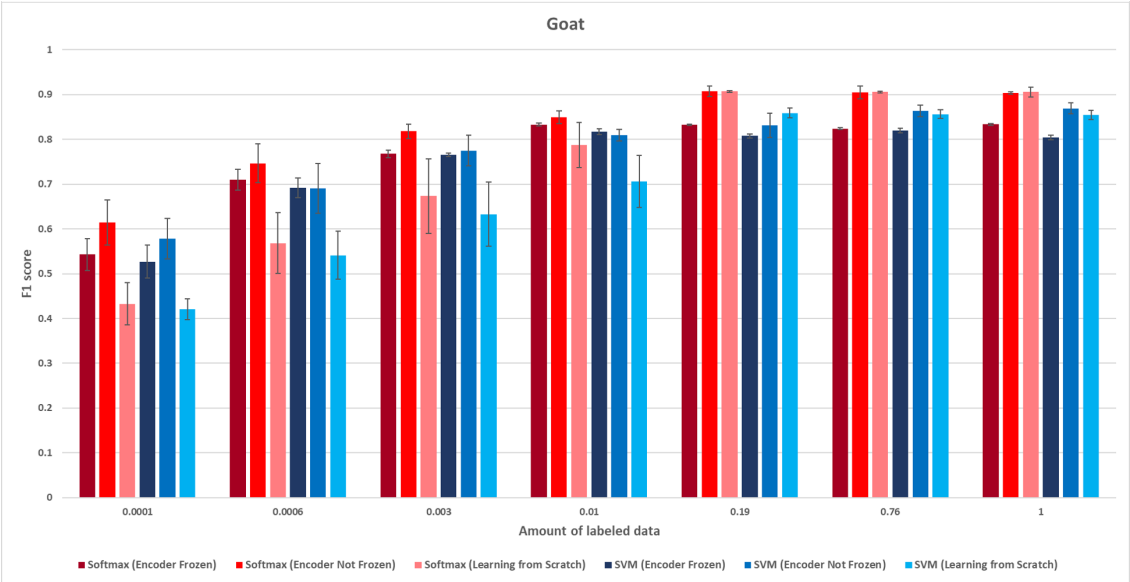

Figure 5.14: The results on goat movement data

5.4 Performance Evaluation of Transferability

To investigate how transferable our model is, we looked into how representative the individual horse data are for six horses and how our horse model is transferable for goat data.

5.4.1 How Representative Individual Horse Data are for Six Horses

As the experiment illustration shown in Figure 5.15, to see how representative one individual horse data to six horses data. We used individual horse unlabeled data to train the VAE model, from the performance evaluation of baseline, the best performed LSTM-VAE model is chosen. After training of VAE, the VAE encoder is taken and a linear classification layer is applied on the top of the encoder, frozen encoder is used here so the parameters of VAE encoder cannot be updated. We used small amount of six horses labeled training data(0.01% and 0.06%) for the classification neural network training and the six horses labeled testing data is used to evaluate the performance.

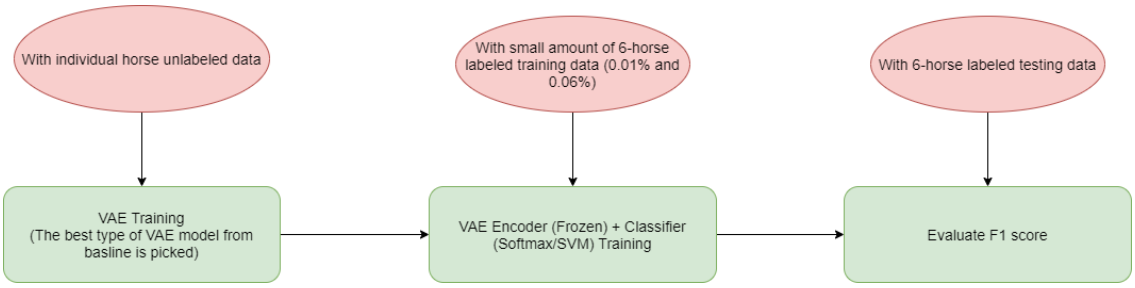

Figure 5.15: The experiment of how representative individual horse data to six horses data

Results:

The results can be found in Figure 5.16. The left graph shows the classification NN trained with 0.01% horse labeled training data and the right graph shows the classification NN trained with

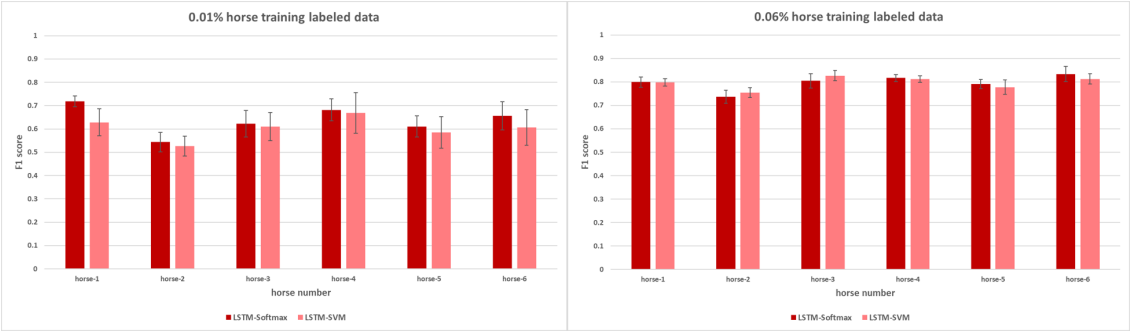

Figure 5.16: LSTM-VAE trained with individual horse and validated with six horses data

0.06% horse labeled training data. The x-axis indicates the individual horse, so from "horse-1" to "horse-6" where the index is corresponding to the index shown in Table 4.1 in Section 4.1. It can be found that the results differ among each other especially with the least amount of training data (0.01%) and some of them perform even better than 6-horse LSTM-VAE shown in Figure 5.2. We looked into the LSTM-VAE final loss after training to see whether we can obtain some information. The LSTM-VAE end loss after training can be found in Figure 5.17.

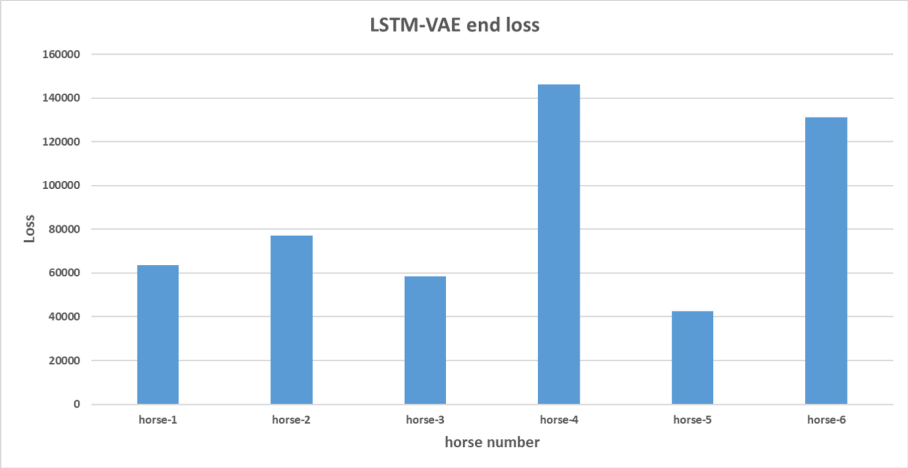

Figure 5.17: LSTM-VAE end loss after training

It was found that after training, LSTM-VAE trained with horse-4 and horse-6 data has high end loss while the performances of horse-4 and horse-6 are not the worst. We then realized that the VAE loss indicates how close the reconstructions are compared with the input. So, we cannot make relations between the VAE loss and the classification performance, for instance, a simple input into VAE and VAE can easily reconstruct it, in this case, the VAE loss is low while it does not mean that it has learned good representations; Or a complex meaningful input into VAE and it is hard for VAE to make perfect reconstruction, now, the VAE loss is high, but it does not mean that the VAE does not learn good representations. During the VAE training, the VAE can still learn good representations from the complex input while the VAE loss is high. If we inspect Figure 5.16, from both 0.01% and 0.06% training data, it can be found that horse-4 model is better than horse-2 model. Another way to corroborate it is to observe the latent space visualization of horse-2 model and horse-4 model as the latent space visualization is a direct illustration of how good the VAE learns representations. The latent space visualization of horse-2 model and horse-4 model can be seen in Figure 5.18 and 5.19 respectively.

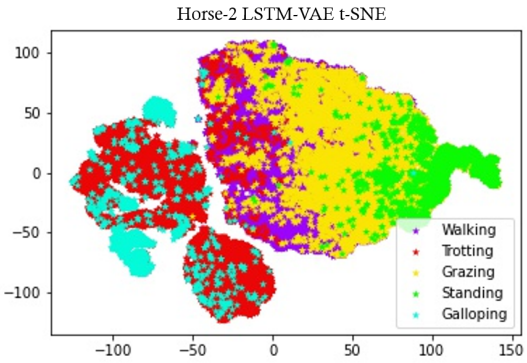

Figure 5.18: The horse-2 model t-SNE

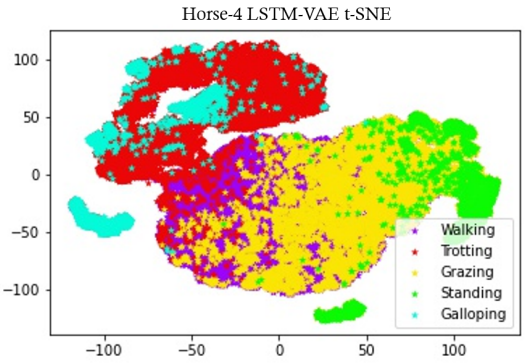

Figure 5.19: The horse-4 model t-SNE

It can be observed that horse-4 model has better clustering than horse-2 model, for instance, horse-2 model has more blue points("Galloping") mixed into the red cluster ("Trotting") and more green points ("Standing") mixed into the yellow cluster ("Grazing"). As a result, by observing the latent space visualization, it gives us the intuition that model-4 is better than model-2 and indeed, Figure 5.16 illustrates horse-4 model has better performance than horse-2 model. The reason causing it should be the horse data related, for instance, the different data distribution of each horse, the different characteristics of the sensors used, the different horse movement pattern etc. But what caused it leaves open in this thesis.

5.4.2 Transferability of Horse VAE Model for Goat Data

As shown in Figure 5.20, it is aimed at investigating how transferable our best performed LSTM-VAE model trained with horse data for another animal goat. The LSTM-VAE model trained with six horses data from baseline was taken. Then, the classification NN was trained with different amount of five goats labeled training data. Both "Encoder Frozen" and "Encoder Not Frozen" were implemented so that we can compare the results. The performance was evaluated with five goats labeled testing data.

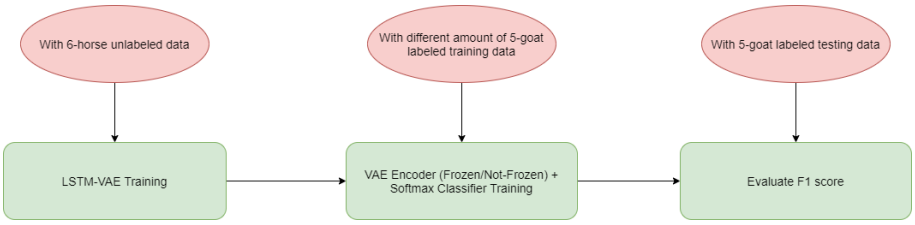

Figure 5.20: The experiment of transferability of horse VAE model for goat data

Results:

The results of the transferability experiments can be found in Figure 5.21. Similar to the previous result illustration, we used red color to indicate our baseline, so, the VAE model was trained on goat movement data and tested on goat movement data. Dark red indicates the situation of "Encoder Frozen" and light red indicates the condition of "Encoder Not Frozen". We used blue color to indicate the performance of the transferability, so, the VAE model was trained on horse movement data and tested on goat movement data. Dark blue shows the case of "Encoder Frozen" and light blue illustrates the condition of "Encoder Not Frozen".

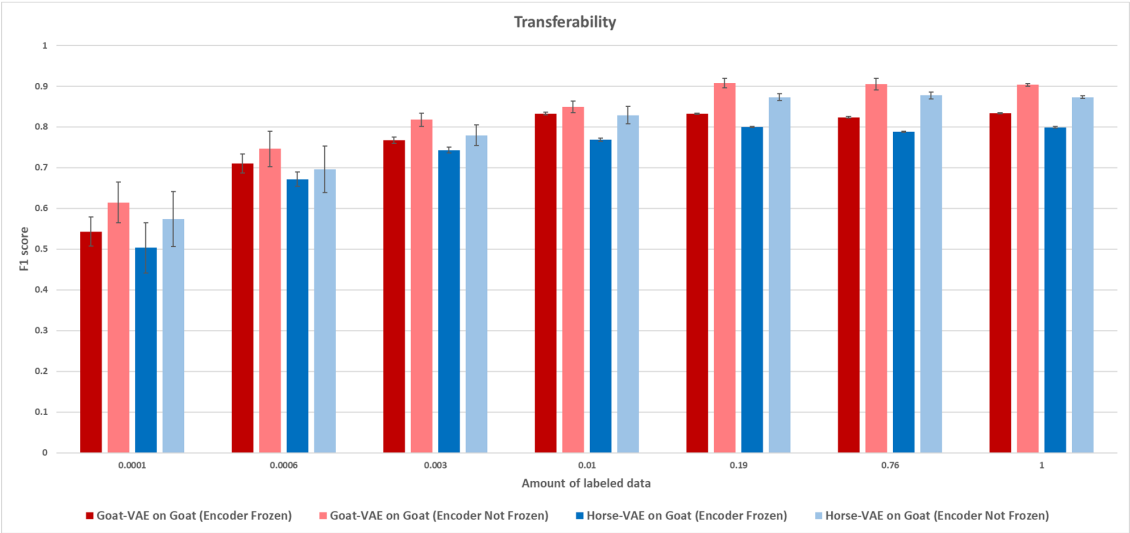

Figure 5.21: The results of transferability experiment

By comparing the dark red with dark blue or comparing light red with light blue, it can be found that "Horse-VAE on Goat" performs slightly worse than "Goat-VAE on Goat", so, the model trained on horse movement data and tested on goat movement data performs slightly worse than the model trained on goat movement data and tested on goat movement data. The performance difference for the case of "Encoder Frozen" is approximately 4% and for the case of "Encoder Not Frizen" is around 3%. The performance difference is not huge, therefore, we can conclude our model has good transferability for another animal goat.

5.5 Model Training Time

Figure 5.22 illustrates the experiments of measuring the neural network training time. We measured the time consumption of training VAE with 100% six horse unlabeled data and the time consumption of training classification neural network with 100% six horse labeled training data for both cases of "Encoder Frozen" and "Encoder Not Frozen".

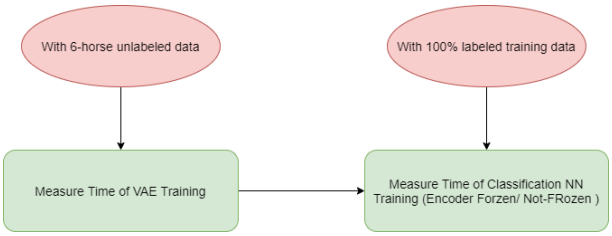

Figure 5.22: The experiment of measuring training time

5.5.1 Time Consumption of VAE Training

Figure 5.23 illustrates the time consumption of VAE training. It can be found that although LSTM-VAE is the best performed model, it takes the most amount of time for training. CNN-VAE costs the least amount of time for training. One reason why CNN-VAE performs worse than LSTM-VAE might be that we did not use deep CNN architecture (only two CNN layers are used in CNN-VAE encoder and decoder). It might be the case that the performance can be increased

by using deeper CNN architecture [74] or using different kernel size [75]. In that case, the time consumption of training CNN-VAE might increase with deeper CNN architecture, however, we might still expect that time consumption of training CNN-VAE is less than that of LSTM-VAE as the time consumption of training LSTM-VAE is approximately four times more than that of LSTM-VAE in our case.

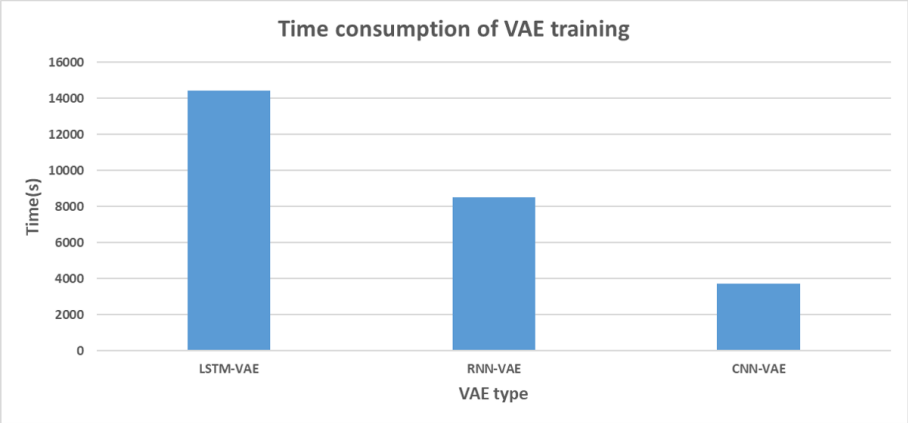

Figure 5.23: The time consumption of VAE training

5.5.2 Time Consumption of Classification Neural Network Training

Figure 5.24 illustrates the time consumption of classification neural network training. In the bracket, we used "F" indicates the situation of "Encoder Frozen" and "NF" indicates the case of "Encoder Not Frozen".

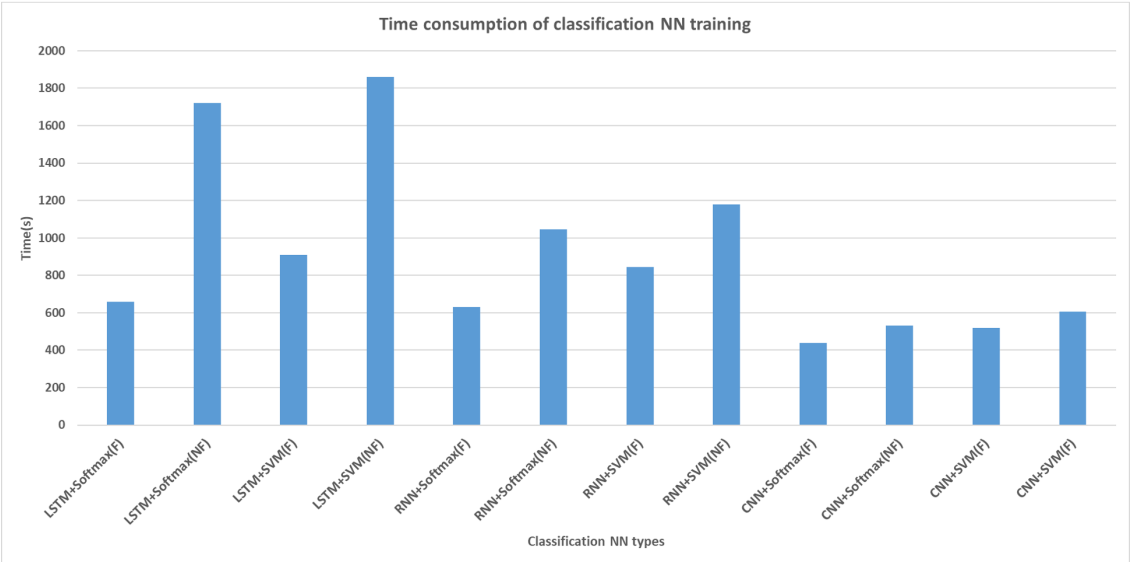

Figure 5.24: The time consumption of classificaiton neural network training

It can be found that with all other settings the same, LSTM takes the most amount of time for training while CNN costs the least amount of time for training. For LSTM networks, by unfreezing the encoder, the time consumption increases dramatically while for CNN networks, the time consumption slightly increases. Furthermore, SVM classifier training takes more time than Softmax classifier training.

Chapter 6

Conclusions

This thesis addressed the problem of how to cope with lack of labeled data in the context of animal activity recognition. We investigated to what extent can unsupervised representation learning using variational auto-encoder cope with lack of labeled data in animal activity recognition. The effectiveness of three types of VAE (RNN-VAE, LSTM-VAE and CNN-VAE) was studied. We also looked into the impacts of different classifiers (Support Vector Machine and Softmax). Besides, we discovered how different amount of labeled training impacts the classification performance and how transferable our unsupervised representation learning methods for another animal. In addition, we investigated the performance improvement by enabling the fine-tuning (unfreezing the encoder) and discovered whether the animal movement data is naturally distinguished and how our unsupervised representation learning helps increase the performance compared with directly supervised learning from scratch. In terms of the model transferability, we inspected how representative individual horse data are for six horse and how transerable the model for another animal. Finally, we measured the time consumption of model training to discover the resource usage for training.

Hereby, based on the experiment evaluation in Chapter 5, we answer the research questions that were proposed in Chapter 1 as follows:

- RQ-1: To what extent VAE improves the performance (F1 score) of AAR?

  The best performed model is LSTM-VAE model and it outperforms the unsupervised representation learning methods of the state-of-art [2] with lack of labeled data and our methods tend to be more robust than that of state-of-art [2].

- RQ-2: What are the impacts of different classifiers (Support Vector Machine (SVM), Softmax) on the performance?

  Generally speaking, the unsupervised representation learning with Softmax classifier has better performance than that with Support Vector Machine classifier.

- RQ-3: What is the impact of the amount of labeled data on the performance?

  With the amount of labeled training data increasing, the performance increases and the error bar shrinks. When it reaches the top, the performance stays stable.

- RQ-4: How transferable is the unsupervised representation learning (VAE) for another animal?

  Our model has good transferability for another animal. Specifically, the model trained with horse movement data has good performance on goat movement data.

Besides, from the experiment evaluation in Chapter 5, more conclusions can be drawn:

- LSTM-VAE model has the best performance and the following is the CNN-VAE model.

- RNN-VAE mode has bad performance as it suffers from short-term memory problem so that it cannot handle long input sequence.
- The animal movement data cannot be easily distinguished so that the good performance of our model is dominated by the unsupervised representation learning method using VAE instead of the classifier.
- The performance can be improved by unfreezing the encoder.
- For the same neural network architecture, if the neural network has more trainable parameters, the error bar tends to become larger than that with less trainable parameters.
- With lack of labeled training data, the performance of supervised learning from scratch is less than that of unsupervised representation learning and with the amount of labeled training data increasing, the performance difference becomes less.
- Latent space visualization using t-SNE is a good method to inspect how good the unsupervised representation learning has learned by visualizing how separable the tasks are.
- Data quality plays an important role in unsupervised representation learning, by observing the latent space visualization using t-SNE, we can intuitively distinguish which one learns better and that results tend to be consistent with the result of the downstream classification task.
- LSTM-VAE is the best performed model while it takes the most amount of training time, CNN-VAE costs the least amount of training time.
- By unfreezing the encoder, the training time of LSTM classification neural network increases dramatically while for CNN classification neural network, the training time slightly increases.
- SVM classifier training takes more time than Softmax classifier training.

6.1 Future Work

In this section, we provide some possible future work based on this thesis:

- The good performance of VAE motivates further investigation with generative models such as generative adversarial network (GAN) for animal activity recognition
- As CNN-VAE training time consumption is much less than that of LSTM-VAE although the performance of CNN-VAE is worse than LSTM-VAE, by using a different CNN-VAE architecture (for example, deeper architecture, different kernal size, etc.), we might obtain better performance which is close to LSTM-VAE while with less resource usage.
- As for our best performed LSTM-VAE model, we only have one hidden layer for encoder and one hidden layer for decoder. It might be the case that with deeper architecture (more hidden layers), we can obtain better results.
- As the reasons why individual horse model performs different from each other leave open in this thesis, it might be interesting to inspect the reasons, for example, by looking into the data distribution, doing controlled experiments to see whether it is the reason of horse or sensors, etc.

Bibliography

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[2] Jacob W. Kamminga, Nirvana Meratnia, Duc V. Le, and Paul J.M. Havinga. Towards deep unsupervised representation learning from accelerometer time series for animal activity recognition. August 2020. URL https://kdd-milets.github.io/milets2020/. 6th Workshop on Mining and Learning from Time Series, MiLeTS 2020, MiLeTS 2020 ; Conference date: 24-08-2020 Through 24-08-2020.

[3] Jacob Kamminga, Nirvana Meratnia, and Paul Havinga. Dataset: Horse movement data and analysis of its potential for activity recognition. 11 2019. ISBN 9781450369930. doi: 10.1145/3359427.3361908.

[4] J.W. Kamminga. Multi sensor-orientation movement data of goats. 2018. doi: 10.17026/DANS-XHN-BSFB.

[5] Linda Kalof and Georgina M. Montgomery. *Making Animal Meaning*. Michigan State University Press, 2011. ISBN 9781611860160. URL http://www.jstor.org/stable/10.14321/j.ctt7zt85w.

[6] Ran Nathan. An emerging movement ecology paradigm. *Proceedings of the National Academy of Sciences*, 105(49):19050–19051, 2008. ISSN 0027-8424. doi: 10.1073/pnas.0808918105.

[7] Antonio-Javier Garcia-Sanchez, Felipe Garcia-Sanchez, Fernando Losilla, Pawel Kulakowski, Joan Garcia-Haro, Alejandro Rodríguez, José-Vicente López-Bao, and Francisco Palomares. Wireless sensor network deployment for monitoring wildlife passages. *Sensors*, 10(8):7236–7262, 2010. ISSN 1424-8220. doi: 10.3390/s100807236.

[8] Sheng-He Sun. Development trend of modern sensor. *Journal=Journal of Electronic Measurement and Instrument, year = 2009, number = 1, pages = 1-10,.*

[9] Jacob Kamminga, Duc Le, Jan Meijers, Helena Bisby, Nirvana Meratnia, and Paul Havinga. Robust sensor-orientation-independent feature selection for animal activity recognition on collar tags. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2:1–27, 03 2018. doi: 10.1145/3191747.

[10] Enkeleda Bocaj, Dimitris Uzunidis, Panagiotis Kasnesis, and Charalampos Z. Patrikakis. On the benefits of deep convolutional neural networks on animal activity recognition. In *2020 International Conference on Smart Systems and Technologies (SST)*, pages 83–88, 2020. doi: 10.1109/SST49455.2020.9263702.

[11] Fuqiang Gu, Kourosh Khoshelham, Shahrokh Valaee, Jianga Shang, and Rui Zhang. Locomotion activity recognition using stacked denoising autoencoders. *IEEE Internet of Things Journal*, 5:2085 – 2093, 04 2018. doi: 10.1109/JIOT.2018.2823084.

[12] C.M. Bishop. *Pattern Recognition and Machine Learning: All "just the Facts 101" Material*. Information science and statistics. Springer (India) Private Limited, 2013. ISBN 9788132209065. URL https://books.google.nl/books?id=HL4HrgEACAAJ.

[13] Hong Huo Tao Fang Yiyou Guo, Jinsheng Ji and Deren Li. Sip-fs: a novel feature selection for data representation. *Journal of Image Video Processing*, (14), 2018. doi: 10.1186/s13640-018-0252-3.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN 9780262035613. http://www.deeplearningbook.org.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. 2014. URL https://arxiv.org/abs/1312.6114.

[16] Yingqi Peng, Naoshi Kondo, Tateshi Fujiura, Tetsuhito Suzuki, Wulandari, Hidetsugu Yoshioka, and Erina Itoyama. Classification of multiple cattle behavior patterns using a recurrent neural network with long short-term memory and inertial measurement units. *Computers and Electronics in Agriculture*, 157:247–253, 2019. ISSN 0168-1699. doi: 10.1016/j.compag.2018.12.023.

[17] Lu Bai, Chris Yeung, Christos Efstratiou, and Moyra Chikomo. Motion2vector: Unsupervised learning in human activity recognition using wrist-sensing data. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, UbiComp/ISWC '19 Adjunct, page 537–542, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368698. doi: 10.1145/3341162.3349335.

[18] J. Pereira and M. Silveira. Learning representations from healthcare time series data for unsupervised anomaly detection. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–7, 2019. doi: 10.1109/BIGCOMP.2019.8679157.

[19] Thomas Pellegrini. Comparing SVM, softmax, and shallow neural networks for eating condition classification. In *Proc. Interspeech 2015*, pages 899–903, 2015. doi: 10.21437/Interspeech.2015-191.

[20] Diane Cook, Kyle D Feuz, and Narayanan C Krishnan. Transfer learning for activity recognition: A survey. *Knowledge and information systems*, 36(3):537–556, 2013. doi: 10.1007/s10115-013-0665-3.

[21] Renjie Ding, Xue Li, Lanshun Nie, Jiazhen Li, Xiandong Si, Dianhui Chu, Guozhong Liu, and Dechen Zhan. Empirical study and improvement on deep transfer learning for human activity recognition. *Sensors*, 19(1), 2019. ISSN 1424-8220. doi: 10.3390/s19010057.

[22] L.M.A. Tonnaer. Active learning in VAE latent space. Master's thesis, Eindhoven University of Technology, the Netherlands, 2017.

[23] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

[24] Herve Bourlard and Y Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59:291–4, 02 1988. doi: 10.1007/BF00332918.

[25] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS)*, page 3–10, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[26] Alexander Camuto, Matthew Willetts, Stephen Roberts, Chris Holmes, and Tom Rainforth. Towards a theoretical understanding of the robustness of variational autoencoders. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3565–3573. PMLR, 13–15 Apr 2021.

[27] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. ISSN 00034851.

[28] Shimao. Why is vae reconstruction loss equal to mse loss. Blog post, 07 2019. URL https://stats.stackexchange.com/q/419784.

[29] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, Mar 2020. ISSN 0167-2789. doi: 10.1016/j.physd.2019.132306.

[30] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017. doi: 10.1109/ICEngTechnol.2017.8308186.

[31] Gerhard Hagerer, Vedhas Pandit, Florian Eyben, and Björn Schuller. Enhancing lstm rnn-based speech overlap detection by artificially mixed data. In *Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. Audio Engineering Society, 2017. URL https://www.aes.org/e-lib/browse.cfm?elib=18764.

[32] Richard Sproat and Navdeep Jaitly. RNN approaches to text normalization: A challenge. *CoRR*, abs/1611.00068, 2016. URL http://arxiv.org/abs/1611.00068.

[33] Francisco Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16:115, 01 2016. doi: 10.3390/s16010115.

[34] Dimitri Palaz, Ronan Collobert, et al. Analysis of cnn-based speech recognition system using raw speech as input. Technical report, Idiap, 2015.

[35] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[36] Sebastian Münzner, Philip Schmidt, Attila Reiss, Michael Hanselmann, Rainer Stiefelhagen, and Robert Dürichen. Cnn-based sensor fusion techniques for multimodal human activity recognition. page 158–165, 2017. doi: 10.1145/3123021.3123046.

[37] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[38] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[39] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of convolution neural network advances on the imagenet. *Computer Vision and Image Understanding*, 161:11–19, Aug 2017. ISSN 1077-3142. doi: 10.1016/j.cviu.2017.05.007.

[40] Tarik Rashid. How to derive the relationship between number of inputs and number of neurons in backpropagation neural network with one hidden layer? Blog post, 09 2015. URL https://www.researchgate.net/post/How-to-derive-the-relationship-between-number-of-inputs-and-number-of-neurons-in-Backpropagation-neural-network-with-one-hidden-layer.

[41] Xenon. How does the bottleneck z dimension affect the reconstruction loss in vaes. Blog post, 02 2019. URL https://stats.stackexchange.com/q/450385.

[42] Steven D. Whitehead and Dana H. Ballard. Learning to perceive and act by trial and error. *Mach. Learn.*, 7(1):45–83, July 1991. ISSN 0885-6125. doi: 10.1023/A:1022619109594.

[43] Xingqun Qi, Tianhui Wang, and Jiaming Liu. Comparison of support vector machine and softmax classifiers in computer vision. In *2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 151–155. IEEE, 2017. doi: 10.1109/ICMCCE.2017.49.

[44] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '19, page 75–78, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368810. doi: 10.1145/3341981.3344221.

[45] Xenon. An introduction to support vector machines (svm): Gradient descent solution. Blog post, 05 2019. URL https://nianlonggu.com/2019/05/24/tutorial-on-SVM/.

[46] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[47] Ranjay Krishna Fei-Fei Li and Danfei Xu. Transfer learning. URL https://cs231n.github.io/transfer-learning/.

[48] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3), January 2014. ISSN 0360-0300. doi: 10.1145/2499621.

[49] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence - Volume 3*, IAAI'05, page 1541–1546. AAAI Press, 2005. ISBN 157735236x. doi: 10.5555/1620092.1620107.

[50] Wei Zhuang, Yi Chen, Jian Su, Baowei Wang, and Chunming Gao. Design of human activity recognition algorithms based on a single wearable IMU sensor. *International Journal of Sensor Networks*, 30(3):193, 2019. doi: 10.1504/ijsnet.2019.100218.

[51] Tâm Huynh and Bernt Schiele. Analyzing features for activity recognition. In *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies*, sOc-EUSAI '05, page 159–163, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595933042. doi: 10.1145/1107548.1107591.

[52] Andreas Bulling, Jamie A. Ward, and Hans Gellersen. Multimodal recognition of reading activity in transit using body-worn sensors. *ACM Transactions on Applied Perception*, 9(1), March 2012. ISSN 1544-3558. doi: 10.1145/2134203.2134205.

[53] Ling Bao and Stephen S. Intille. *Activity Recognition from User-Annotated Acceleration Data*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-24646-6.

[54] Roberto Luis Shinmoto Torres, Qinfeng Shi, Anton van den Hengel, and Damith C. Ranasinghe. A hierarchical model for recognizing alarming states in a batteryless sensor alarm intervention for preventing falls in older people. *Pervasive and Mobile Computing*, 40:1–16, 2017. ISSN 1574-1192. doi: 10.1016/j.pmcj.2017.04.002.

[55] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. page 3995–4001, 2015. doi: 10.5555/2832747.2832806.

[56] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *6th International Conference on Mobile Computing, Applications and Services*, pages 197–205, 2014. doi: 10.4108/icst.mobicase.2014.257786.

[57] Tahmina Zebin, Matthew Sperrin, Niels Peek, and Alexander J Casson. Human activity recognition from inertial sensor time-series using batch normalized deep lstm recurrent networks. In *2018 40th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 1–4. IEEE, 2018. doi: 10.1109/EMBC.2018.8513115.

[58] Aiguo Wang, Guilin Chen, Cuijuan Shang, Miaofei Zhang, and Li Liu. Human activity recognition in a smart home environment with stacked denoising autoencoders. In Shaoxu Song and Yongxin Tong, editors, *Web-Age Information Management*, pages 29–40, Cham, 2016. Springer International Publishing. ISBN 978-3-319-47121-1.

[59] Abdelmonim Artoli Bandar Almaslukh, Jalal Al Muhtadi. An effective deep autoencoder approach for online smartphone- based human activity recognition. *IJCSNS International Journal of Computer Science and Network Security*, 17(4), 2017.

[60] Alireza Abedin Varamin, Ehsan Abbasnejad, Qinfeng Shi, Damith C. Ranasinghe, and Hamid Rezatofighi. Deep auto-set: A deep auto-encoder-set network for activity recognition using wearables. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '18, page 246–253, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360937. doi: 10.1145/3286978.3287024.

[61] E. Casella, A. R. Khamesi, and S. Silvestri. Smartwatch application for horse gaits activity recognition. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 409–416, Los Alamitos, CA, USA, jun 2019. IEEE Computer Society. doi: 10.1109/SMARTCOMP.2019.00080.

[62] Ritaban Dutta, Daniel Smith, Richard Rawnsley, Greg Bishop-Hurley, James Hills, Greg Timms, and Dave Henry. Dynamic cattle behavioural classification using supervised ensemble classifiers. *Computers and Electronics in Agriculture*, 111:18–28, 2015. ISSN 0168-1699. doi: 10.1016/j.compag.2014.12.002.

[63] Raghuraman Gopalan, Ruonan Li, Vishal M Patel, and Rama Chellappa. Domain adaptation for visual recognition. *Foundations and Trends® in Computer Graphics and Vision*, 8(4): 285–378, 2015. URL https://ieeexplore.ieee.org/document/8187168.

[64] Francisco Javier Ordóñez Morales and Daniel Roggen. Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, ISWC '16, page 92–99, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450344609. doi: 10.1145/2971763.2971764.

[65] Alois Ferscha Alberto Calatroni Daniel Roggen Gerhard Troster Marc Kurz, Gerold Holzl. Real-time transfer and evaluation of activity recognition capabilities in an opportunistic system. In *Proceedings of the Third International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2011. ISBN 978.

[66] Ali Akbari and Roozbeh Jafari. Transferring activity recognition models for new wearable sensors with deep generative domain adaptation. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, IPSN '19, page 85–96, New York,

NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362849. doi: 10.1145/3302506.3310391.

[67] Md. Sharif Hossen. *Data Preprocess.* John Wiley Sons, Ltd, 2020. ISBN 9781119654834. doi: 10.1002/9781119654834.ch4.

[68] Wenqian Jiang, Yang Hong, Beitong Zhou, Xin He, and Cheng Cheng. A gan-based anomaly detection approach for imbalanced industrial time series. *IEEE Access*, 7:143608–143619, 2019. doi: 10.1109/ACCESS.2019.2944689.

[69] Jacob W. Kamminga, Duc V. Le, Jan Pieter Meijers, Helena Bisby, Nirvana Meratnia, and Paul J.M. Havinga. Robust sensor-orientation-independent feature selection for animal activity recognition on collar tags. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies.*, 2(1), March 2018. doi: 10.1145/3191747.

[70] Saurabh Jaju. Guide to t-sne machine learning algorithm implemented in r python. Blog post, 01 2017. URL https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r-python/.

[71] Venkat Gudivada, Amy Apon, and Junhua Ding. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, 10(1):1–20, 2017. URL https://thinkmind.org/download.php?articleid=soft_v10_n12_2017_1.

[72] Abhinav Atla, Rahul Tada, Victor Sheng, and Naveen Singireddy. Sensitivity of different machine learning algorithms to noise. *Journal of Computing Sciences in Colleges*, 26(5): 96–103, May 2011. ISSN 1937-4771. doi: 10.5555/1961574.1961594.

[73] Fadi Thabtah, Suhel Hammoud, Firuz Kamalov, and Amanda Gonsalves. Data imbalance in classification: Experimental evaluation. *Information Sciences*, 513:429–441, 2020. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2019.11.004.

[74] Alison O'Shea, Gordon Lightbody, Geraldine Boylan, and Andriy Temko. Investigating the impact of cnn depth on neonatal seizure detection performance. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5862–5865. IEEE, 2018. doi: 10.1109/EMBC.2018.8513617.

[75] Abhinav Agrawal and Namita Mittal. Using cnn for facial expression recognition: a study of the effects of kernel size and number of filters on accuracy. *The Visual Computer*, 36(2): 405–412, 2020. doi: 10.1007/s00371-019-01630-9.