UTRECHT UNIVERSITY
Faculty of Science
Department of Information and Computing Sciences
MSc Artificial Intelligence

AUTOMATED BEHAVIOR ESTIMATION FOR PAIN DETECTION IN DOGS WITH COMPUTER VISION

A THESIS BY
Lisanne Koetsier
6522289

Project supervisor Prof. dr. Albert Ali Salah
Second examiner Prof. dr. Remco Veltkamp

Utrecht University

Abstract

Studying dog behavior is a crucial aspect in determining their welfare since their behavior cues reflect their mental state. Studying behaviors in dogs can aid veterinarians in estimating their pain levels. Especially dogs experiencing chronic pain are masters at masking their pain signals. Deep learning models might be able to pick up the pain signals that even experts do not immediately notice. Our starting point is a two-stream network encompassing a ConvLSTM stream for RGB frames and an LSTM stream for processing keypoints to classify pain. We propose adding a third stream that encapsulates an ethogram, representing the dogs' behaviors in the corresponding input frames to improve the baseline two-stream pain estimation model. We established a method that uses the Multi-modal Semantic Query Network (MSQNet) to generate behavior labels for unlabeled data to deal with the problem of annotating an abundance of videos, which is a labor-intensive and time-consuming process. We processed the ethograms using two different methods: a simple linear layer and a random forest classifier, respectively. Analyzing the performance of these methods showed us that the former performed better than the latter, possibly due to the limited amount of training samples available. Additionally, explainability is a crucial aspect of pain detection in dogs, since we want to know why the dogs appear to be in pain. We tested several methods to aid explainability, including saliency heatmaps, Shapley values, Naive Bayes probabilities, confidence scores, and a decision tree. From the explainability measures, we can conclude that the models tend to classify dogs with higher activity into the “no pain" class, and dogs with low activity into the “pain" class.

Acknowledgements

First of all, I would like to express my gratitude to Prof. Dr. Albert Ali Salah. Throughout the year, he provided me with great insights and knowledge for the thesis and guided me in the process.
I would like to thank Prof. Dr. Yasemin Salgırlı Demirbaş for providing us with the dog pain dataset and providing an additional in-the-wild dog pain dataset for this thesis. Her insights into dog pain behavior have been of great value to this thesis.
Additionally, I would like to thank my friends and family for always being there for me.

I acknowledge using large language models to aid me in checking my spelling and grammar and providing suggestions for improving my writing style.

Table of Contents

1. Introduction

Animals play a central part in our lives and their welfare is of great value to us. Studying animal behavior is a crucial aspect in determining their welfare since their behavior cues reflect their mental state. Studying and annotating animal behavior is however a labor-intensive and time-consuming process. Advances in posture estimation and motion tracking in recent years have helped a great deal in in studying animal behavior. However, it is difficult to effectively determine the internal states of animals, such as pain levels and emotions, with only posture estimation and motion tracking. This is due to the subjective nature of these states and animals lack verbal communication and the skills to clarify what emotions and pain levels they are experiencing. It is however, possible to capture patterns and behaviors of animals associated with pain and emotional states. In other words, there is a lack of ground-truth for animal pain and emotions.

In recent years more works are released that tackle the problem of automated pain and emotion recognition in animals with the aim of improving animal welfare (Broomé et al., 2019, 2022; Zhu et al., 2022; Boneh-Shitrit et al., 2022a; Ferres et al., 2022; Feighelstein et al., 2022, 2023b,a). However, there is still little research for dogs in this field in particular. This thesis will be focusing on the automated pain estimation in dogs and the automatic estimation of behaviors that might imply that a dog is experiencing pain to a certain extent, building on existing literature in pain and behavior estimation in animals. Additionally, we devoted our time to creating explainable and interpretable results.

1.1 Relevance

Effectively capturing pain in dogs is of utmost importance for a variety of reasons. Since dogs are not able to communicate their discomfort and needs verbally, and veterinarians are not always able to accurately capture all pain signals in dogs, additional measures in detecting pain may improve dog welfare. Capturing pain signals in dogs early on can help prevent serious medical conditions from manifesting and prevent dog owners from needing to pay for high medical costs. Therefore, accurate pain detection can improve the overall well-being of dogs and detect unnoticed pain signals early on.

1.2 Baseline model

This thesis will build forth on the previous work by Zhu et al. (2022), where a two-stream model was proposed to automate pain detection in dogs. This two-stream model consists of a ConvLSTM stream that processes RGB frames of a dog video clip, and an LSTM

stream that processes the keypoints of that same clip. Ultimately, these streams are fused by concatenation and a fully-connected network produces a confidence score of how likely it deems the dog to be in pain. During the replication of the results of Zhu et al. (2022), we obtained lower accuracies than reported in the paper. This is most likely due to an unfair train, validation, and test set being used in Zhu et al. (2019). We will elaborate on this in Section 4.4.1.

1.3 Problems

There are several challenges in pain estimation in dogs. One major issue is that there is a tremendous scarcity in dog datasets, and especially in datasets of dogs in pain. Consequently, models that should handle the rather complex problem of detection subtle behavioral cues that indicate pain, must train on small datasets. This often is a difficult that since complex learning problems usually require complex models which in turn are data hungry. Simple models usually have difficulties to capture the necessary patterns to accurately predict

Deep learning models have the known problem to have a black-box nature. This is particularly undesired in models that are supposed to classify pain. Pain detection models should aid the veterinarian in their pain assessment, and not necessarily completely take over the process. The veterinarian should be able to substantiate why a dog is in pain. Thus, explainability in pain detection models is crucial.

Additionally, traditional pain assessments methods for animals suffer from the fact that they are generally subjective in nature, since observers usually assigns a degree of presence of pain or behavior which is subject to their own impression and experience (Hernandez-Avalos et al., 2019). This can lead to underestimation or overestimation in pain essessment and thus lead to inaccurate medical treatment in animals. For example, behaviors associated to pain in dogs are often overlooked by veterniarions if the particular breed is prone to exhibiting these behaviors on a regular basis (Mills et al., 2020; Rohdin et al., 2018). Traditional pain assessment methods fail here to accurately classify pain.

1.4 Proposed solution

To combat the mentioned problems, we will be using the earlier mentioned pain detection method by Zhu et al. (2022) as a baseline. To this existing model, we added a third stream that processes the behaviors occurring in the same clip that the other streams analyze. The behaviors are processed in the form of an ethogram, which is obtained by an action recognition model called Multi-model Semantic Query Network (MSQNet) (Mondal et al.,

2023). The three streams are fused by concatenation after which the feature vectors are fed to a feed-forward network that produces a pain score.

To aid explainability of the model, we will compare two different ways of processing the ethogram. Namely, we fed the ethogram to a random forest classifier. This classifier provides us with decision trees that provide an intuitive way to interpret the pain classification. On the other hand, we processed the ethogram with a simple linear layer and analyzed the output with Shapley values and Naive Bayes probabilities. Additionally, saliency heatmaps are used to determine where on the dog the model attends to the most.

1.5 Research questions

The main research question of this thesis is whether we can create **a system to automatically quantify pain expression from visual behavioral cues in dogs**. This is a challenging problem because pain is subjective, and a dog cannot communicate its pain levels clearly to a human. Consequently, the question of ground truth is challenging in creating automatic methods for animal pain detection. Another challenge is explainability in well-performing deep learning methods, which is crucial if a veterinarian wants to explain why a dog is experiencing pain or why not.

- Produce detailed annotations for pain indicators in existing datasets on dogs, and develop methods to facilitate such annotations.
- Search for new sources of data to use in the problem of pain estimation for dogs.
- Implement state-of-the-art deep neural network pipelines to process body images of dogs for pain estimation.
- Implement automatic detectors for individual indicators of pain for dogs. These indicators are typically evaluated by a veterinarian or an ethologist, on an “ethogram," which we detail later in the thesis. We will aim to create specialized detectors for each item of the ethogram.
- Perform quantitative and qualitative analyses to comprehend what the current models are capable of achieving and provide intuitive methods to aid explainability in pain detection in dogs.

1.5.1 Contributions

The contributions of this thesis are summarized as follows:

- We provide detailed behavioral annotations on the dog dataset used by Zhu et al. (2022) and we facilitate a method based on MSQNet to generate behavior annotations on unseen videos to reduce the time-consuming and labor-intensive annotation

process.

- We implement an automated behavior estimation method with MSQNet to detect individual indicators of pain in dogs.
- We propose a three-stream architecture that incorporates behavioral cues from dogs from different modalities.
- We provide detailed explainability measures for our three-stream model that aids in explaining what factors have classified a dog as pain or non-pain.
- We provide a detailed evaluation of the two different methods to handle the ethogram stream and list their pros and cons in terms of explainability and accuracy.

1.6 Structure of the thesis

This thesis will be structured as follows. First, further in this section, the goal and the research questions of the thesis will be proposed. Subsequently, in the related work section, we will discuss important posture, behavior, and pain analysis methods together with other relevant state-of-the-art neural networks. After this, we will discuss our methodology and experimental evaluation. Finally, we will explore some less successful pilot studies and our discussion and conclusion.

2. Related Work

The difficulty of pain detection in animals, and in our case dogs, can be related to the challenges of pain assessment in human infants (Zamzmi et al., 2017). Similarly to infants, dogs are not able to verbally express whether they are experiencing pain or not (Broome et al., 2023). This makes it difficult for veterinary experts to assess whether a dog is in pain, which in turn makes adequate treatment of chronic pain in dogs more difficult (Bell et al., 2014).

As far to our knowledge the only paper that has discussed automatic detection of dog pain with computer vision and pose estimation is by Zhu et al. (2022), which this thesis is extending upon. Although there is little work on dog pain detection, pain detection is closely related to emotion detection. There is some work on automatic dog emotion and pain detection using facial analysis and pose analysis which will be discussed in this section. The faces of the dogs in this thesis are not sufficiently visible in the video frames, which is why we do not use facial analysis for this thesis. However, facial analysis is an important subject in emotion and pain detection in animals, which is why we will discuss it here. Since this thesis proposes to improve the model by Zhu et al. (2022) with behavior estimation in dogs, we will be expanding upon dog pain behavior as well.

The related work section is structured as follows. First, the problem of pose estimation will be highlighted. After this several state-of-the-art action recognition methods and models. Emotions and pain are closely linked but research developed mainly separately from each other (Hale et al., 1997), which is why we choose to discuss these separately. First, emotion detection in animals will be addressed, after which the related work specific to pain detection will be scrutinized. After this, the challenges with animal pain datasets will be elaborated on. Subsequently, aided and automated annotation of animal video data will be discussed and several methods are discussed which could be helpful to reduce annotation time. Explainability in AI is discussed after this, focusing on explainability in automatic emotion and pain detection in the animal domain. Last but not least, good measures to increase the performance of deep learning methods will be addressed.

2.1 Pose estimation

The movement of animals can reflect their well-being, which can be extracted from estimating their pose. The pose of an animal is obtained by capturing the positions of their joints or other keypoints (Jiang et al., 2022). Intrusive methods such as motion capture can

be used to get this, but can negatively impact the animal's behavior and thus introduce bias in the observations (Broome et al., 2023). Additionally, for certain animals, such as wild animals, this method is not feasible. Therefore, automated pose estimation of animals is desirable. This thesis will focus on the extraction of 2D poses from dogs. Thus, there will be a focus on 2D pose estimation methods in this section.

There are two backbones that are frequently used for animal pose estimation, namely ResNet (He et al., 2016a) and HRNet (Wang et al., 2020). ResNet is a neural network, which will be further elaborated on in Section 2.2, and HRNet is a popular convolutional neural network often used for human pose estimation as well. HRNet is used in this thesis and a detailed explanation can be found in Section 3.3.

There are two main branches for pose estimation, namely multi-pose estimation and single-pose estimation. Single-pose estimation is a method where a pose is estimated for one individual in an image. In contrast, for multi-pose estimation, multiple individuals are taken into account simultaneously. There are two main ways to tackle multi-pose estimation: either a top-down or a bottom-up approach. A top-down approach in multi-pose estimation divides individuals into separate single-pose estimations. This becomes challenging when animals are occluded by each other, which makes it difficult to separate individuals from each other (Jiang et al., 2022). The bottom-up approach first detects the keypoints of all the individuals in an image, and subsequently links all the keypoints to an individual and constructs a pose. The main challenge here is the part where the keypoints must be linked to an individual (Jiang et al., 2022).

One of the challenges that animals pose estimation algorithms encounter, like many other animal-based algorithms, is a data scarcity problem. Especially in contrast to the human domain, which has access to a vast amount of data. This challenge demands animal pose estimation algorithms to be able to produce decent results with a small-scale dataset. The current most used solution to tackle this problem is transfer learning. Transfer learning is a technique where a neural network is first pre-trained on a large dataset, which, in this case, should contain a variety of animal classes. Examples of such datasets are ImageNet (He et al., 2016b) and MS-COCO (Lin et al., 2014), which contain a dog class amongst others. This leaves a pre-trained network already sensitive to detecting animals and aids in further training with smaller and more task-specific datasets, such as for animal pose estimation. One excellent example of an animal pose estimation toolbox that is used by numerous researchers is DeepLabCut (Mathis et al., 2018). DeepLabCut uses a pre-trained ResNet network, which was trained on ImageNet and achieves excellent performance on pose estimation on various animal species with only a few hundred training instances.

Another commonly used pose estimation algorithm that does not use transfer learning is DeepPoseKit (Graving et al., 2019). DeepPoseKit uses a stacked-hourglass architecture named Stacked DenseNet (Jégou et al., 2017). A stacked-hourglass network is a convolutional neural network often used in pose estimation algorithms and thanks its name due to its shape. The shape emerges due to successive pooling and upsampling, which constantly reduces and increases the dimensionality of the network. Stacked DenseNet combines two stacked hourglass networks with an encoding-decoding structure. DeepPoseKit also performs well on a small training dataset with only a few hundred training images, while it does not use transfer learning.

LEAP (Pereira et al., 2019) is another pose estimation method which is interesting. LEAP is designed to make quick predictions of an animal's pose and try to reduce the complexity of its model. This makes LEAP interesting to applications where real-time pose estimation is desirable. LEAP is based on SegNet (Badrinarayanan et al., 2017) which encompasses an encoder-decoder structure where the encoding operation is a convolutional network and the decoding operation a deconvolutional network. The trade-off for LEAP's inference speed is that it does not reach high accuracy in comparison to method such as DeepLabCut and is not able to generalize as well. For example, it has difficulties with varying light intensities in images and rotations. Recently SLEAP (Pereira et al., 2022) has been released which is the successor to LEAP. SLEAP is able to deal with multi-pose estimation in contrast to SLEAP and enables both a top-down and bottom-up approach for this. Table 1 shows an overview of the discussed pose estimation methods in this section.

Name	Pre-trained	Single/multi-pose	Top-down/bottom-up
DeepLabCut	ResNet	Both	Bottom-up
DeepPoseKit	From scratch	Single	Top-down
HR-Net	Multiple models available	Both	Both
LEAP	SegNet	Single	Top-down
SLEAP	Multiple models available	Multi	Both

Table 1. An overview table of animal pose estimation algorithms with categories whether the model is pre-trained, single or multi-pose and whether it is a top-down or bottom-up approach in case of a multi-pose classifier.

2.2 Action recognition in animals

Temporal information is an important factor in automatically detecting pain in animals, which generally requires video data input since an image only does not provide temporal information (Broomé et al., 2019). Therefore, for a model to be able to understand what is

going on in a video frame and in between video frames is crucial. In other words, models should be able to extract spatial information from single video frames and simultaneously be able to recognize temporal cues across numerous consecutive frames over a period of time. The act of detecting the boundaries of actions is called *action spotting*. The concept of classifying actions within pre-defined boundaries is called *action recognition*. We will be focusing on action recognition during this thesis.

This section will discuss a selection of popular and state-of-the-art models and methods for dealing with temporal and spatial information. The main focus will be on works applied in the field of animal emotion and pain detection. First general neural networks will be discussed such as CNNs, recurrent neural networks such as LSTMs, and Transformer-based networks. Subsequently, the popular two-stream model method will be addressed together with approaches to fuse the two streams in a two-stream network. Lastly, the rise of prompts-based learning in action recognition will be touched upon.

2.2.1 Convolutional Neural Networks (CNN)

Convolutional neural networks are great at extracting features from images in a reasonable time. This is due to CNNs being efficient at not losing image quality while reducing the vast number of parameters simultaneously. This is desirable since images have high dimensionality where every single pixel is considered as a feature. The dimensionality is reduced by using the core building block of the CNN, the convolutional layer. This is a layer where a filter slides over the input matrix and produces a smaller feature map, which reduces the dimensionality. Due to its efficiency in dealing with high dimensionality data CNNs are often used for action recognition tasks as well. There are several popular pre-trained CNNs which are often used for image recognition tasks, such as ResNet-50 He et al. (2016a), AlexNet (Krizhevsky et al., 2012) and InceptionV3 (Szegedy et al., 2016). The ResNet-50 network is for example used for automated cat pain detection by Feighelstein et al. (2022) and canine emotion detection by Boneh-Shitrit et al. (2022b), AlexNet for estimation emotions of dogs with automated facial analysis (Franzoni et al., 2019) and InceptionV3 for facial analysis of mice to estimate their well-being (Andresen et al., 2020).

2.2.2 Long-Term Short Memory Network (LSTM)

Long-Term Short Memory Networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are networks which are recurrent neural networks (RNN). These can deal with long-term dependencies, in contrast to the standard RNN which fails to remember events that occurred too long ago. Standard RNNs have only a hidden state, which forgets long-term information quickly due to the vanishing gradient problem (Hochreiter, 1998). The LSTM model tackles this by adding an input gate, forget gate, and output gate (DiPietro and Hager, 2020). These gates respectively decide which information to remember, forget, and output. The downside is that the LSTM cannot capture which timestep contains essential information. Therefore,

capturing long-term dependencies from long videos would be challenging nonetheless. To tackle this problem, an attention mechanism can be added to deal with longer-term dependencies from long videos (Wang et al., 2016). The attention mechanism consists of a context vector representing a sequence of, in the case of action recognition, video frames. The context vector assigns higher weights to specific points in the sequence, which should gain more attention than others and help remember the most important parts of a video frame.

Convolutional LSTM

Most standard LSTMs have layers that are fully connected to each other and are useful for making non-linear connections between learned features and thus being able to deal with non-linearity. This makes them efficient at learning temporal information from a sequence, for example, in Natural Language Processing.
However, the fully connected layers of LSTMs are usually inefficient at processing spatial information and extracting visual features. To solve the problem of LSTMs not being able to extract spatial information sufficiently, convolutional LSTMs (ConvLSTM) are often used. The ConvLSTM was intially proposed by Shi et al. (2015), and it replaces the matrix multiplication in the LSTM model with the convolutional operator. This allows the model to process spatial information more efficiently instead of only being able to process sequential 1D vectors. This will enable it to process temporal and spatial information simultaneously. The ConvLSTM is, for example, used by Broomé et al. (2019, 2022); Zhu et al. (2022) to extract spatial information from the video frames.

2.2.3 Transformer-based networks

Transformer-based networks grow increasingly popular for computer vision tasks due to their ability to capture long-term dependencies over a sequential input. Vision Transformers (ViT) (Dosovitskiy et al., 2020) have proven to be effective alternatives to CNN networks.

Vision Transformers are networks that entirely use attention mechanisms instead of convolutional layers such as in CNNs and ConvLSTMs. Using solely attention mechanisms instead of convolutional layers gives ViT models the ability to capture long-range dependencies effectively and pay close attention to relevant details in the image. In addition, ViT models capture global image properties, making it easier to spot subtle cues that might indicate pain in the entire image (Feighelstein et al., 2023a). Furthermore, since it has a deeper architecture with more parameters than LSTMs and CNNs, it can capture more delicate details which might be indicators of pain.

The vision transformer operates as follows and is inspired by the original transformer structure proposed in Vaswani et al. (2017). Figure 1 gives an overview of the model. The input image is first split into a sequence of patches with a pre-defined size. Input

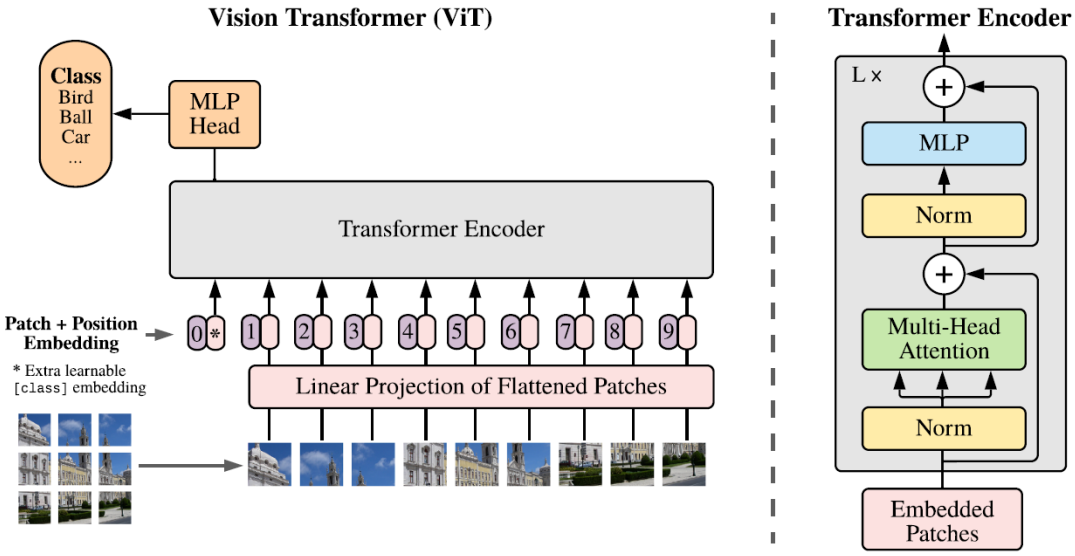


Figure 1. Figure from Dosovitskiy et al. (2020) which shows a model overview of the Vision Transformer. The image is split up into several patches with a fixed size and linearly projected, and positional embeddings are added to retain positional information in the image. These patches are subsequently fed into the transformer encoder.

$x \in \mathbb{R}^{H \times W \times C}$ is split up into a sequence of patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$. The resolution of the input image is (H, W), and it outputs patches with a resolution of a pre-defined size (P, P). C depicts the number of channels and $N = HW/P^2$ is the number of image patches in the patch sequence x_p. To all the patches in x_p 1D positional embeddings are added to be able to extract the position of the patches in the image. The patches are subsequently fed into the Transformer Encoder as shown in Figure 1.

The transformer encoder part is derived from Vaswani et al. (2017) and operates as follows. The encoder alternates between applying Multi-Head Attention (MLA) blocks and Multi-Layer Perceptron (MLP) blocks. Before each block, layer normalization (Ba et al., 2016) is applied (LN), and after each block, residual connections (He et al., 2016a) are applied.

MLA is a block that contains multiple self-attention heads. MLA combines several self-attention heads by running *k* self-attention heads in parallel and ultimately concatenate the outputs and linearly project these to obtain the final results from the MLA.

Self-attention is a mechanism that scrutinizes the relationship between different components in the input. In the case of Natural Language Processing, self-attention would look for the relationships within a single sentence, and for vision tasks, this would include looking for relationships of different regions in the same image. Self-attention works with a so-called query, key, and values structure. It compares all the queries and keys with each

other (e.g., image regions) with a similarity function; the most similar key's content would turn out to be the value. Attention is calculated as shown in Equation 2.1 (Dosovitskiy et al., 2020). q and k are the query and key values, respectively, and D_k represents the dimension of the key k. Subsequently, the self-attention as presented in Equation 2.2, the self-attention of an input sequence z is represented by multiplying the weights in the attention layer from Equation 2.1 with all the values v.

$$A = softmax\ (qk^{\intercal}/\sqrt{D_k}) \tag{2.1}$$

$$SA(z) = Av \tag{2.2}$$

In Boneh-Shitrit et al. (2022a), a ResNet-50 CNN and a Vision Transformer were compared against each other, where it appeared that the ViT performed better on the task of emotion recognition in canines. In Feighelstein et al. (2023a) a ResNet50 CNN was compared to a CLIP-based ViT model, where the latter showed a slightly better performance as well. Boneh-Shitrit et al. (2022a) suggest that the ViT model performs better than the CNN model because emotion recognition requires a good understanding of features at an object parts level, which the ViT model is more sensitive to than CNNs (Amir et al., 2021).

2.2.4 Hidden two-stream models

Before two-stream models for action recognition existed, researchers relied on classical optical flow computation to capture temporal information between consecutive video frames. This is done before feeding the input into a CNN, making it a serial process. This two-stage pipeline required a lot of storage and was computationally expensive (Zhu et al., 2019).

In Zhu et al. (2019), a hidden feed-forward two-stream network was introduced. In this network one stream focuses on capturing temporal information in videos, and the other stream focuses on extracting spatial features from the video frames, making it a process that can be executed in parallel. This enables the model to estimate optical flow in real time while the model is running. In the end, these streams are fused to produce a single output. The method is called a *hidden* two-stream network, because it implicitly estimates the optical flow, in contrast to traditional two-stage optical flow estimators. This method saves a lot of time and outperforms the serial optical flow approaches.

Hidden two-stream models are nowadays a common approach to tackle the challenge

of action recognition. The paper by Zhu et al. (2022), which this thesis is extending, uses a recurrent two-stream model as well, where one stream captures spatial-temporal information based on frames, and the other stream temporal information based on the keypoints from a pose. Broomé et al. (2019) also uses a recurrent two-stream network, where one stream captures motion and the other spatial information. The motion stream captures the optical flow by determining the horizontal and vertical direction gradient for each pixel in the frame with a ConvLSTM. The spatial stream extracts features from RGB images by using a ConvLSTM.

2.2.5 Fusion methods

The commonly used two-stream model networks typically use a fusion to combine the two separate streams into a single predictor which could improve performance significantly (Ebrahimi Kahou et al., 2015). There are different ways to fuse multiple streams, of which a few will be discussed in this section.

There are at least two levels at which fusion can take place—namely *feature-level fusion* and *decision-level fusion*. Feature-level fusion is a fusion strategy that takes place before classification. This method is used in Broomé et al. (2019), where the motion stream is added to the RGB stream element-wise with *additive fusion*. In Ebrahimi Kahou et al. (2015) the proposed emotion recognition model seems to perform better with feature-level fusion than with *decision-level fusion*, where the latter is a fusion technique where fusion takes place by combining the classifiers of separate streams instead of before classification.

In feature-level fusion, the input dimensionality is increased, which may make some approaches much more complex in case the model complexity depends non-linearly on input dimensionality. However, the learner can learn the relationship between features from different streams, which can be beneficial for context understanding and, thus, the performance of the model. In decision level fusion, different types of features are used in different sub-systems. This is typically more frequently used when the features are very different regarding range, size, or sampling frequency.

2.2.6 Prompt-based learning

With the rise of the new state-of-the-art pre-trained prompt-based network called CLIP (Radford et al., 2021), new methods for prompt-based action recognition have been developed. CLIP is an open source neural network model pre-trained on image and text pairs where the text describes the image. In this way the model is not restricted to only pre-defined labels to learn from but has access to a greater variety of text to describe images. CLIP is commonly used in recent approaches due to its accessibility of being open-source.

Lately, CLIP-based action recognition and image recognition models for animal behavior analysis, such as the CLIP-based ViT model in (Feighelstein et al., 2023a; Mondal et al., 2023) are rising. Since CLIP is prompt-based, it allows for prompt-based approaches in action recognition more easily. This gives a free path toward exploring the usage of semantic information in action labels. Using the semantic information of action labels and matching prompts can provide models with the necessary contextual information they need next to spatial-temporal features, in order to accurately recognize actions.

In Wang et al. (2021), a new zero-shot/few-shot model is introduced named ActionCLIP. In this thesis, we ran some tests with a pre-trained ActionCLIP model, this can be viewed in pilot studies in the Appendix. The authors argue that the traditional unimodal frameworks for deep learning are designed to predict a predetermined set of categories without considering the semantic value encapsulated in the categories. This disadvantages the model's generalizability since it cannot quickly learn new categories without new training data. They suggest a way broader approach to supervision where the supervision comes from processing raw text in contrast to comparing labels. This is reminiscent of how humans process new visual information together with its semantics and link it to their current knowledge of natural language. The authors aim to be able to perform zero-shot learning without requiring extra labeled training data to fine-tune the model. They realize this by comparing textual and visual features to each other with a similarity calculation. A video will be classified as the label with the highest similarity score. Figure 2 shows an overview of the pipeline.

$$\hat{y} = \arg\max_{y \in Y} P(f(x, y)|\theta) \tag{2.3}$$

$$s(x, y) = \frac{v \times w^{\intercal}}{\|v\|\|w\|}, s(y, x) = \frac{w \times v^{\intercal}}{\|w\|\|v\|} \tag{2.4}$$

$$\begin{aligned} p_i^{x2y}(x) &= \frac{\exp(s(x, y_i)/\tau)}{\sum_{j=1}^{N} \exp(s(x, y_i)/\tau)} \\ p_i^{y2x}(y) &= \frac{\exp(s(y, x_i)/\tau)}{\sum_{j=1}^{N} \exp(s(y, x_i)/\tau)} \end{aligned} \tag{2.5}$$

The similarity calculation operates as follows. Instead of training a model to solely predict the conditional probability of a label y for an input video x, Wang et al. (2021) suggest

another approach by calculating the probability $P(f(x,y)|\theta)$ with $f(x,y)$ being a similarity function. In this way, the label y with the highest similarity score is chosen as the label of the video x as depicted in Equation 2.3. The similarity score between the two modalities is realized by using the similarity function known as the cosine similarity, as shown in Equation 2.4. Here $v = g_V(x)$ is the video encoder, and $w = g_W(y)$ is the textual encoder. To achieve a normalized similarity score in the range of 0-1, the $s(x,y)$ and $s(y,x)$ are put into a Softmax function which is shown in Equation 2.5.

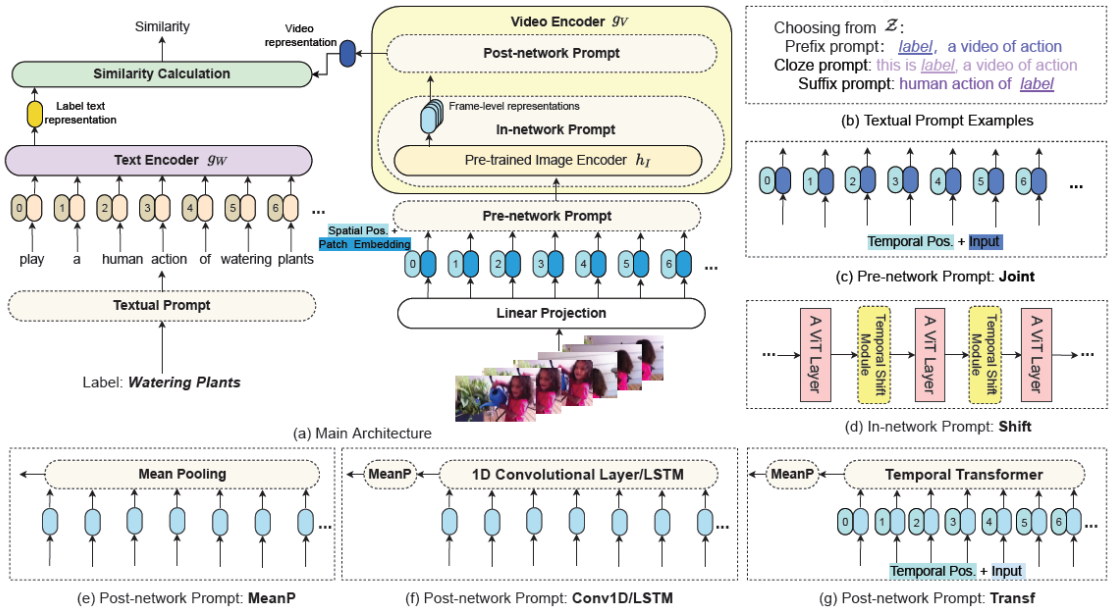


Figure 2. Figure from Wang et al. (2021) that shows how the ActionCLIP framework calculates similarities between videos and labels and determines which label is assigned to which video from the highest similarity score.

A new paradigm is suggested by Wang et al. (2021) called *pre-train, prompt and fine-tune*. A model overview is shown in Figure 2. The *pre-train* step refers to extracting an abundance of data from the web, since these are generally already linked to some kind of textual representation, or to using a large pre-trained model such as CLIP. The authors choose the large pre-trained model CLIP in this paper, which refers to the CLIP part in the model its name: *ActionCLIP*.

Visual and textual prompts are obtained during the *prompt* phase. The textual prompts are obtained by executing a filling function. Based on a given label y and a set of values Z a filling function $f_{fill}(y,z)$ is used to obtain input value y' where $z \in Z$. The fill prompt has prefix, close, and suffix variants, which depend on the location of where the fill function is applied in the text.

The visual prompts are obtained by several functions that extract spatio-temporal features from a given video. The visual prompts are barely necessary when the pre-trained model is

based on video-text data, unlike when trained on image-text data. In the case where there is pre-trained image-text data, there is still a necessity for the model to learn the temporal relationships of videos. The different kinds of prompts are used by Wang et al. (2021) to obtain temporal information, namely pre-network, in-network, and post-network prompts, are shown in Figure 2.

The suggested pre-network prompt by Wang et al. (2021) is called *joint* and alters the input before it is fed into the video encoder g_v. For each video input, temporal positional embeddings are added to the spatial positional embeddings. This also allows the g_v to learn temporal features, despite being pre-trained on image-text data.

The in-network prompt as seen in Figure 2d is referred to as *Shift* and operates by shifting the feature channels partly over the temporal channels. In this way consecutive frames can share information between each other and allow for capturing temporal information between frames. This gives the video encoder the ability to learn spatio-temporal features.

For the post-network prompt, first, a spatial encoder and subsequently a temporal encoder extracts features. The paper used the pre-trained image-text model as a spatial encoder to extract features. For the temporal encoder part of the post-network prompt four different approaches are suggested by the author to tackle this and are shown in Figure 2 e, f, g. The first one shown in e is called *MeanP*. The second and third approaches shown in f are called *Conv1D* and *LSTM*. The last one depicted in g is called *Transf.* MeanP is an abbreviation for the mean pooling operation. Conv1D is a 1D convolutional neural network and LSTM a recurrent neural network. Transf is a temporal vision transformer with *k* layers. Lastly, the *fine-tune* step straightforwardly requires for fine-tuning on specific datasets to increase performance in comparison.

Multi-modal Semantic Query Network

The difficulty in performing action recognition in animal behavior is that animals come in all shapes and sizes, even within the same species. This diversity is especially evident in dogs, which are the subject of this thesis. The diverse morphology and breeds in dogs make action recognition in dogs incredibly challenging (Farhat et al., 2024). Due to the ample variety in animals, action recognition models trained on animals are usually specifically modeled for a particular actor because it is easier and less computationally expensive to train a model on a single actor. One approach to actor-specific models is to add specific pose information about an actor in the form of keypoints. This makes it easier for the model to extract similar behaviors since it provides concise detail about the location and movement of particular body parts without analyzing every pixel in a video frame.

Mondal et al. (2023) tackle this issue of deep learning models not being able to generalize over actions inter and intra-species by proposing a model that is actor agnostic and able to output multiple actions for the same video segment at the same time. They propose a new model called Multi-model Semantic Query Network (MSQNet). This multi-modal network integrates visual input from videos and semantic information from text labels corresponding to actions using a pre-trained vision-language model such as CLIP (Radford et al., 2021). The actor-agnostic nature of this model makes it particularly interesting for the focus of this thesis due to the variety in morphology of dogs, which is why we are using this for the behavior detection part of our model.

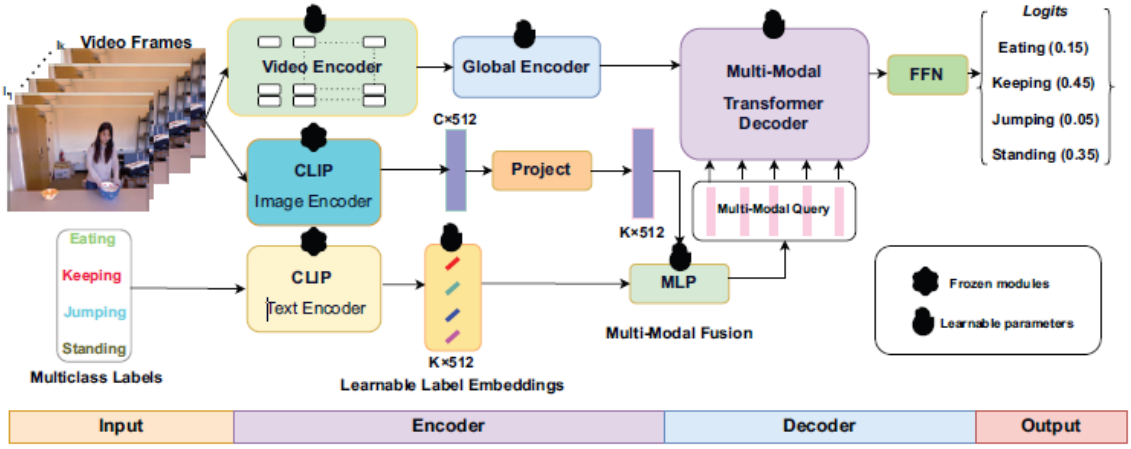


Figure 3. Figure from Mondal et al. (2023) which shows the overview of the Multi-model Semantic Query Network (MSQNet) model. The model consists of three components. The model consists of a spatio-temporal video encoder, a multi-modal query encoder based on CLIP, and a multi-modal decoder.

MSQNet consists of three components, as shown in Figure 3. Namely a *spatio-temporal video encoder* that captures detailed spatial and motion. This component ensures that temporal sequences in a video are captured and thus analyses the temporal information between frames. It captures motion, changes, and interaction over time within a video. The *multi-modal query encoder* component captures visual information from video frames and semantic information from action classes and combines these. The visual and semantic information can be obtained with any preferred vision-language model. The authors Mondal et al. (2023) opt to use the popular vision-language model called CLIP (Radford et al., 2021) as discussed earlier. As shown in Figure 3 there is CLIP image encoder and a CLIP text encoder. Both encoders extract image and label embeddings, respectively. The CLIP embeddings are extracted for multiple frames, and subsequently average pooled to obtain video embeddings. In contrast to the spatio-temporal video encoder, CLIP can in this way provide a semantic understanding that bridges the textual and visual modalities. Lastly, a component called a *multi-modal decoder* leverages a multi-headed self-attention to further process the video encoding and multi-modal queries. The multi-headed self-attention helps the model focus on different video parts simultaneously, generate richer features,

and comprehend more complex patterns. This helps, in particular, with multi-class action recognition. The output of the *multi-modal decoder* is subsequently put into a feed-forward network that produces logits for every single behavior, which indicate which behaviors are most likely to be present in the given video. The feed-forward network serves as a bridge between the high dimensional output of the *multi-modal decoder*.

In the experimental evaluation by Mondal et al. (2023), it shows that the model which is enhanced with textual encoding of the CLIP model consistently outperforms the models without textual features. Due to the combination of visual and textual cues, MSQNet can produce a comprehensive understanding of a specific behavior and its context. Since CLIP is trained on millions of image-text pairs it has incredible zero-shot capability. This aids the actor-agnostic capability of MSQNet, which makes MSQNet particularly interesting for animal cases.

2.3 Emotion Detection

It has been widely accepted that animals can experience both negative emotional states, such as pain, as well as positive emotional states. However, a lot of science and research in animal emotions has solely focused on animals' negative emotional states, such as pain, since there is more incentive to address pain from a veterinary perspective. Veterinarians have to treat animals experiencing pain. Thus, it seems logical that more research on animal pain has been conducted than on positive emotional states. Emotional and pain states are closely related, which is why the topic of emotion recognition is particularly interesting for this thesis.

Determining which emotions animals experience and to what extent is difficult since animals lack the verbal communication skills to clarify what emotions they are experiencing. Additionally, emotions are subjective, which makes it extra challenging to identify the extent to which an animal experiences an emotion or pain. Hence, there is a lack of ground truth to work with. Therefore, there is no set agreement on what exactly conveys emotions in animals (Broome et al., 2023). Pain and emotional assessment in infants is a subject in the human domain, which is tightly related to the same matter in the animal domain since infants are not able to verbally provide researchers with ground truth for pain and emotional experiences (Zamzmi et al., 2017).

The most common way to gain information to determine how animals experience emotions, including pain, is by studying their body and facial expressions. These expressions hold information that is also used amongst animals to communicate with each other. Observing animals' expressions for emotion classification is also one of the least invasive ways to

study emotions (Broome et al., 2023). This is in contrast to assessing physiological factors, which usually requires stressful blood sampling or other methods where the animal has to be restrained involuntarily (Andersen et al., 2021). Additionally, it has been proven to be a more reliable method of assessing affective states than when using physiological factors. Additionally, physiological changes can be heavily influenced by any diseases the animal unwittingly might have and thus prove to be less reliable than studying body and facial expressions (Gleerup et al., 2015). Invasive measures might affect the emotional state of the animal and produce less accurate results and thus are less reliable.

2.3.1 Defining emotions

Emotional states can be loosely defined as "internal states which are expressed in physiological, cognitive and behavioral changes" (Anderson and Adolphs, 2014). Pain is in the past often studied separately from emotions and is defined by Raja et al. (2020) as "an unpleasant sensory and emotional experience".

Although emotions are hard to define, there exist different manners to classify emotions. The two most common theories to classify emotional states are the discrete and dimensional approaches. The discrete approach categorizes emotions into distinct categories. These categories are about the same across different mammalian species since emotions are processed in subcortical regions in the brain. This part of the brain deals with primitive processes such as emotions and is homologous across mammalian species, which causes mammalian species to experience the same primary emotions (Panksepp, 2010). For humans, the most well-known categories are fear, disgust, sadness, anger, surprise, and happiness, as defined by Ekman (1992).

The dimensional approach to classifying emotions mainly consists of two dimensions. By using dimensions, this approach suggests that emotions cannot be discretely classified, but are rather on a spectrum. The most frequently used dimensions are valence and arousal (Russell, 1980). The valence dimension depicts whether an emotion is experienced as a positive or a negative feeling. Fear is an emotion that scores low on valence and thus has negative associations, while happiness has a high valence and is thus experienced as a positive emotion. The arousal dimension depicts how intensely an emotion is experienced. For example, sadness is an emotion with low valence and low arousal, while anger is an emotion with low valence but high arousal. Additionally, a third dimension dominance is also used frequently (Verma and Tiwary, 2017). Dominance denotes the degree of control one experiences over an emotions. For example, an emotion with high dominance is joy after an achievement. Emotions with low dominance can be sadness after the loss of a loved one, since you feel like you can do nothing about the situation.

2.3.2 Facial analysis for emotion detection

Facial analysis to automatically detect emotions is often used in humans. Namely, facial expressions in humans are good indicators of universal emotions (Ekman, 1973), where universal emotions are emotions which are shared amongst different cultures. Facial expressions of emotions are subjective and idiosyncratic. However, to measure facial expressions in humans in an objective way, the Facial Action Coding Systemthat (FACS) is introduced by Ekman and Friesen (1978). The FACS is a system that describes visually perceptible facial movement. The FACS encapsulates a number of so-called facial action units (AUs), which depict certain components of muscle movement. The faces of the dogs in this thesis are not sufficiently visible in the video frames, which is why we do not use facial analysis for this thesis. However, facial analyis is an important subject in emotion and pain detection in animals which is why we will discuss it here.

There are several adaptions of FACS for animals such as dogs (Waller et al., 2013), horses (Wathan et al., 2015), cats (Caeiro et al., 2017) and macaques (Morozov et al., 2021). There are also specific FACS for pain indication in animals, which are referred to as grimace scales and were first developed for mice (Langford et al., 2010). Grimace scales solely focus on parts of the face that are indicators of pain. This will be further elaborated on in Section 2.4.

Manually scoring facial AUs is a labor-intensive process. In Morozov et al. (2021) an automated FACS for macaques is introduced, based on the general FACS for macaques, called MaqFACS. In Morozov et al. (2021) the technique of so-called eigenfaces by Turk and Pentland (1991) is used to classify emotions of macaques. Eigenfaces are essentially a set of eigenvectors that represent the basis of all macaque faces represented in the covariance matrix of principal component analysis (PCA) and are the principal components (PCs). Using eigenfaces also allows dimensionality reduction by selecting the projection axes with the largest variance in the training set, and discarding directions in which there is little variance.

PCA on full faces with emotion recognition seems to perform more poorly than on separate regions of the face, which is why Morozov et al. (2021) chooses to divide the facial regions of the macaques into an upper and lower region together with its corresponding AUs. Dividing the face into separate regions is also called a *parts-based method*, while using the data as a whole is called a *holistic method* (Broome et al., 2023). The eigenfaces are established from the training set. Consequently, the testing images are projected into the eigenspace, and with the k-nearest neighbor (KNN) or support vector machine (SVM) classifier, the nearest eigenface is determined and the AUs are classified with which affective states in macaques are estimated.

2.3.3 Pose analysis for emotion detection

Bodily expressions and movement in animals are good indicators of an animal's emotional state and are simultaneously used for communication purposes amongst an abundance of mammalian species (Diogo et al., 2008). For example, horses spend a lot of time on repetitive activities if they are in a content and calm emotional state. Horses then spend most of their time eating, resting, lying, and in movement (Auer et al., 2021). Bodily expressions and movement can be analyzed with different methods, such as pose analysis (Ferres et al., 2022; Rashid et al., 2022).

In (Ferres et al., 2022), the emotions of dogs were estimated based on their pose. The authors focus on the emotions of happiness, anger, fear, and relaxation. The pose was determined with the pose estimator DeepLabCut (Mathis et al., 2018). Ferres et al. (2022) use images of dogs where first an object detector is used to detect in which part of the image the dog is present. Consequently, the image is cropped such that it contains only the dog. All the images are set to the same size such that information and landmarks across images are comparable. This reduces the number of errors the landmark detection model will make. After this, the landmark detector DeepLabCut was used to detect the most important feature points of the dog, such as the joints and spine. After the landmarks are determined, Ferres et al. (2022) train two models with these landmarks. One model is trained on the raw coordinates of the landmarks that DeepLabCut as generated, and the other model is trained on calculated pose metrics based on the landmarks and produces a decision tree classifier. Pose metrics could include the position of the tail and the angle between different joints in the legs, for example.

The advantage of using the pose metrics in contrast to raw coordinates is that it grants to ability to construct a decision tree since the training data is simpler. This approach gains more explainability in why a model assigns an image to a certain emotion. This will be discussed in greater detail in Section 2.7. On the other hand, the raw coordinates hold more information and show greater performance in comparison to the decision tree approach. For our approach we will be using decision trees as well for explainability purposes.

2.4 Pain detection

The simplest and least expensive method to automatically assess animal pain with computer vision is assessing a single static image. However, as mentioned in Broomé et al. (2019), temporal information is crucial to detect pain. This requires processing videos, which is computationally expensive, and data is scarce. Different ways of automatically detecting pain in animals have been developed with the help of Grimace scales, pose analysis, deep learning techniques, two-stream architectures, and more. In addition, Broomé et al. (2022)

show that domain transfer can help detect subtle pain cues even when a model has been trained on acute pain instead. This section will first discuss the importance of temporal information and its detection methods, followed by pose analysis. Furthermore, facial analysis for pain detection will be scrutinized, pain behaviors in dogs will be discussed, and physiological analysis will be briefly touched upon.

2.4.1 Temporal information

Broomé et al. (2019) have shown that temporal information is a crucial aspect of pain detection in horses. There are different ways to capture temporal information. The two main ways to tackle this are by frame aggregation and by extracting spatio-temporal information. Frame aggregation is a method where outputs from different classifiers working with single frames are aggregated. This then partially incorporates temporal information in the final result. Spatio-temporal methods include both temporal information and spacial information. Both Broomé et al. (2019) and Zhu et al. (2022) use the spatio-temporal method. In both papers, this is done by using a two-stream model since two-stream models allow for different modalities to be fed into the networks, in this case spatial and temporal. In Broomé et al. (2019) they use a two-stream recurrent ConvLSTM model. In Zhu et al. (2022), an LSTM stream for training on extracted keypoints from the images and a ConvLSTM for the spatial information is used. In the end, for both papers, the two streams are fused.

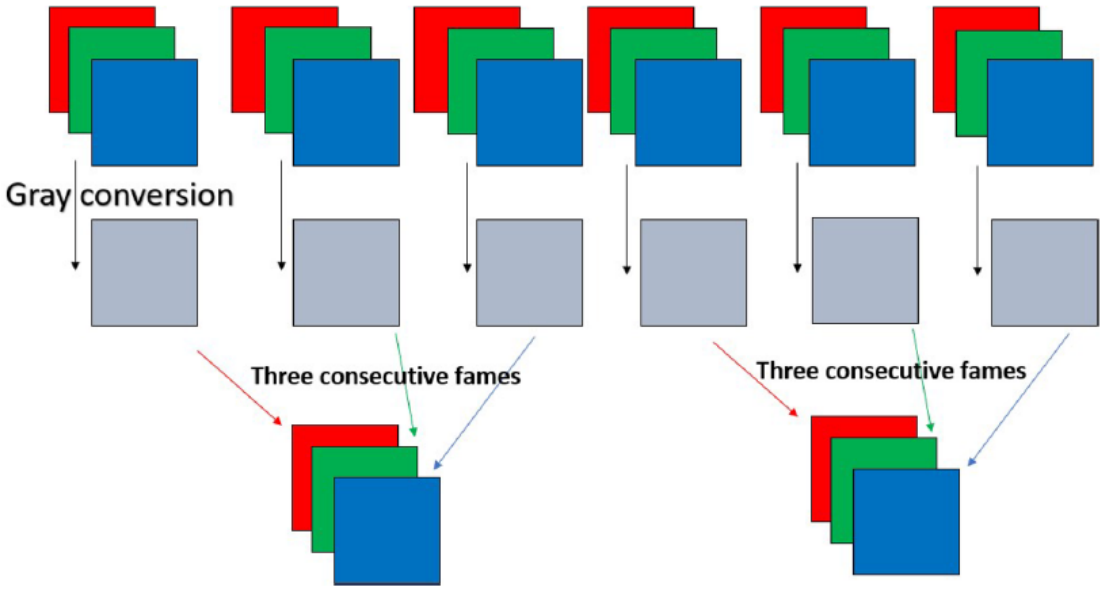
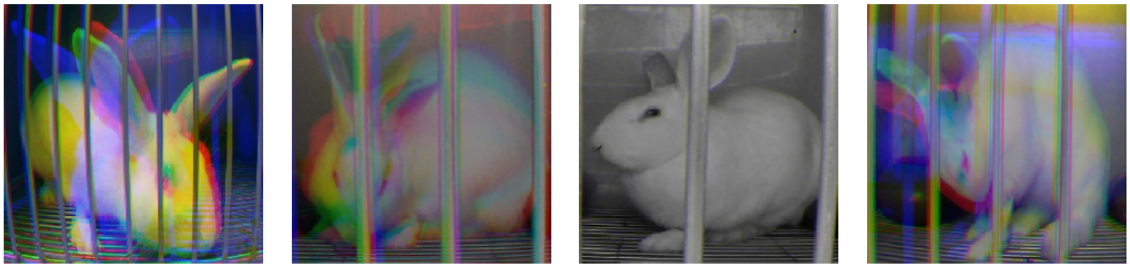


Figure 4. Figure from Feighelstein et al. (2023a) that depicts how the Grayscale Short-Term stacking method operates

Another way to realize frame aggregation to capture temporal information in consecutive frames obtained from a video is by the Grayscale Short-Term stacking (GrayST) method (Kim et al., 2022), which is used in Feighelstein et al. (2023a). In this paper, the

GrayST technique is used together with 1-second interval sampling, which means that every second, a single frame is extracted from a video. This frame then represents that one second of the video. The frames are changed to a grayscale format and aggregated using the GrayST method as shown in Figure 4, where each frame represents a single RGB channel. Temporal information is, in this way, captured in a single image. In Figure 5, four images are shown after applying the GrayST method. A gray image such as Figure 5.3 depicts that the rabbit has not moved for three consecutive frames. If there is a lot of color in the image, such as in the other images, then it shows that the rabbit has moved in the three consecutive frames. An absence of movement might be an indicator of pain, since animals with pain tend to refrain from moving to minimize pain. As shown in Figure 5, the gray image is indeed classified as a rabbit in pain.



(1) No Pain (2) No Pain (3) Pain (4) Pain

Figure 5. Figure from Feighelstein et al. (2023a) that shows how the GrayST method shows temporal information. The subfigures 1, 2 and 4, where colors are clearly visible, suggest that the rabbit has moved in consecutive frames. Subfigure 3, where there is an absence of color, shows that the rabbit has not moved in consecutive frames.

2.4.2 Pose analysis

Despite facial analysis being used more frequently in past works for pain classification than pose analysis, pose analysis used in recent works to incorporate the entire body of an animal in pain classification has shown promising results. In this section, two recent works that incorporate pose analysis in horses and dogs will be discussed.

In (Rashid et al., 2022) equine pain behavior was classified with the help of pose representation. Horses are flight animals, which makes it difficult to observe whether they are in pain, since they might hide signs of pain in the presence of (unfamiliar) human observers (Andersen et al., 2018). This becomes clear when looking at human expert performance classifying horse pain, which reaches only a mere 58% accuracy for acute pain (Broomé et al., 2019). This makes automated pain detection in horses a fruitful contribution. Rashid et al. (2022) use a self-supervised method to classify pain, which requires no labels in the training data. Self-supervised methods are helpful for small datasets since they are less likely to overfit due to human bias in annotation labels, which may occur in training data of supervised methods.

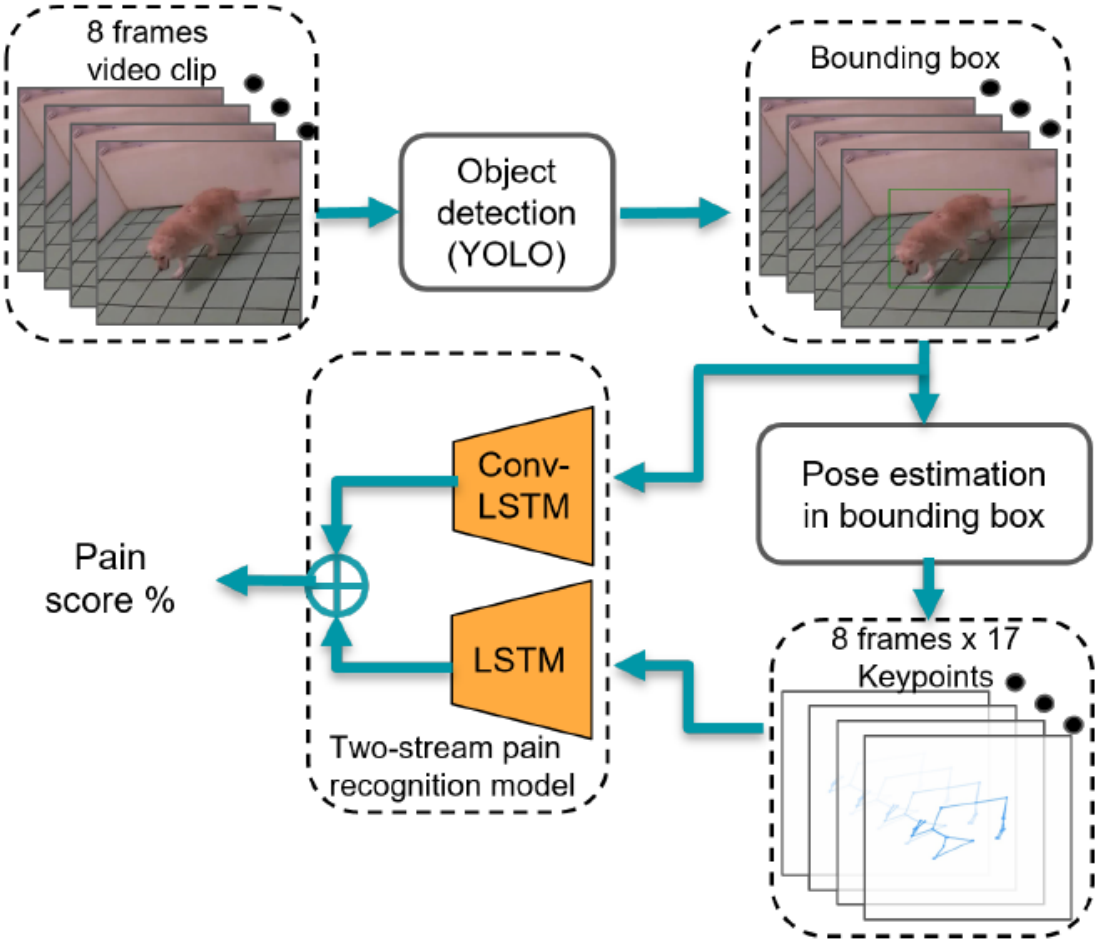


Figure 6. Figure from Zhu et al. (2022) which depicts the pipeline of dog pain classification with a two-stream architecture that incorporates spatial and temporal information.

A pose estimation method was used by Zhu et al. (2022) in a two-stream architecture, which is the paper this thesis will be expanding upon. The pipeline first detects the dog in the video frames with YOLOv5 (Jocher, 2020) object detection and crops the image according to the bounding box plus a 10% margin to ensure the dog is in the image. After that, the videos frames are fed into the ConvLSTM stream of the two-stream network to extract spatial features. For the LSTM stream, first the pose of the dog is estimated for which each pose contains 17 keypoints. The pose representations are fed into the LSTM stream of the two-stream architecture to extract pose features. Subsequently the two outputs are concatenated and a confidence score in terms of pain is produced. This method shows that combining pose analysis with spatial features can produce fruitful results in terms of pain classification.

2.4.3 Facial analysis

There are different ways to address facial expressions in automated pain detection in animals, such as FACS described in Section 2.3, and grimace scales, which focus solely on parts of the face that are relevant to pain detection and thus are less demanding to apply

Study	Species	State	Stimulus	Focus area	State classifier
Hummel et al. (2020)	Horses	Pain	Unknown or induced pain	Face	+
Broomé et al. (2019)	Horses	Pain	Induced pain	Body, Face	+
Rashid et al. (2022)	Horses	Pain	Induced pain	Body	+
Ruhof et al. (2024)	Horses	Pain	Unknown or induced pain	Face	+
Feighelstein et al. (2023a)	Rabbits	Pain	Vet. Procedure	Body	+
Feighelstein et al. (2022)	Cats	Pain	Vet. Procedure	Face	+
Steagall et al. (2023)	Cats	Pain	Naturally occurring	Face	+
Morozov et al. (2021)	Macaques	Emotion	Induced behavior	Face	-
Zhu et al. (2022)	Dogs	Pain	Naturally occurring	Body	+
Ferres et al. (2022)	Dogs	Emotion	Unknown	Body	+
Boneh-Shitrit et al. (2022a)	Dogs	Emotion	Unknown	Face	+

Table 2. An overview table of the discussed articles of pain and emotion classification in animals, heavily inspired by Broome et al. (2023). The articles are categorized by species, state (emotion or pain), stimulus to induce the state, focus area, and whether it includes a state classifier.

than a full FACS. Although grimace scales are an efficient way of establishing whether animals experience pain, it is a time-consuming process to apply due to the need for manual annotations. Additionally, it is highly dependent on the level of expertise of the researchers annotating the grimace scales, which is then subject to bias and no guarantee of validity (Hummel et al., 2020).

Automatic annotation of landmarks in the grimace scales would improve the amount of time needed to annotate grimace scales and perhaps also increase the performance due to a more consistent process. Hummel et al. (2020) automates the annotation of features of the Grimace scales from the Horse Grimace Scale (HGS) (Dalla Costa et al., 2014) and

Study	Species	Part/Holistic	Input	Features
Hummel et al. (2020)	Horses	Parts-based	Frame	Hand-crafted (low-level)
Broomé et al. (2019)	Horses	Holistic	Video	Learned
Broomé et al. (2022)	Horses	Parts-based	Video	Body and Face
Rashid et al. (2022)	Horses	Holistic	Video	Learned
Ruhof et al. (2024)	Horses	Parts-based	Frame	Hand-crafted
Feighelstein et al. (2023a)	Rabbits	Holistic	Frame	Learned
Feighelstein et al. (2022) 1	Cats	Holistic	Frame	Learned
Feighelstein et al. (2022) 2	Cats	Holistic	Frame	Hand-crafted (high-level)
Steagall et al. (2023)	Cats	Holisitc	Frame	Hand-crafted
Morozov et al. (2021)	Macaques	Holistic	Frame	Hand-crafted (high-level)
Zhu et al. (2022)	Dogs	Holistic	Frame	Mixed
Ferres et al. (2022) 1	Dogs	Holistic	Frame	Hand-crafted (high-level)
Ferres et al. (2022) 2	Dogs	Holistic	Frame	Learned
Boneh-Shitrit et al. (2022a)	Dogs	Holistic	Frame	Learned

Table 3. An overview table of analysis approach in pain and emotion classification in animals, heavily inspired by Broome et al. (2023). The methods are categorized by species, parts-based or holistic approach, the kind of input, and whether features are learned or hand-crafted.

the Equine Utrecht University Scale for Facial Assessment of Pain (EQUUS-FAP) (van Loon and Van Dierendonck, 2015). This method incorporates a pose estimation before detecting the landmarks since extreme poses can incur self-occlusion of the facial features in equines. The pose estimation is then able to define the visible areas of interest for the subsequent landmark and pain detection.

Subsequently, Hummel et al. (2020) uses landmark detector models originally for humans. These landmark detector models are called Ensemble of Regression Trees (ERT) and Supervised Descent Model (SDM). These models serve as a base for transfer learning and are fine-tuned in the equine domain. After this, the horse faces are aligned evenly and augmented by adding noise to the landmarks. This data augmentation ensures that there is a bigger dataset and that the model is able to generalize better.

For the pain estimation part, several feature extractor methods were used on several regions of interest (ROIs). The feature extractor methods used are Histogram of Oriented Gradients classifier (HOG) (Lu et al., 2017), Local Binary Pattern (LBP) (Ojala et al., 2002), Scale Invariant Feature Transform (SIFT) (Lowe, 2004) and VGG16 deep neural network model (Simonyan and Zisserman, 2014). The ROIs are the ears, nostrils, eyes, and mouth. All the separate models based on different feature detectors are fused with a weighted fusion and a simple fusion to improve pain detection. Since the data is imbalanced, it is hurting the performance of the F1-score. The problem of imbalanced data will be discussed in further detail in Section 2.5. Ruhof et al. (2024) build upon the work of Hummel et al. (2020) by creating a horse pain estimation model based on ROIs without the need for a need for s pose estimation before detecting facial landmarks. The ROIs are automatically detected with a YOLO-based network and show that pain classification improves when using YOLO-based ROIs in contrast to using ROIs extracted with an ERT.

Feighelstein et al. (2022) are the first to explore the problem of automated pain classification in cats based on facial images. Two approaches were used and compared to each other, namely a landmark-based approach and a deep learning approach. It appeared that the two models obtained similar accuracy and did not outperform each other. However, by prioritizing refinement to the deep learning approach, the model might obtain higher accuracy and thus be more desirable to increase the likelihood of correctly classifying true positive cases (Feighelstein et al., 2022).

One approach in (Feighelstein et al., 2022) uses 48 geometric landmarks which are carefully chosen and manually annotated for the training set based on relevant facial AUs derived from catFACS (Caeiro et al., 2017). The 48 geometric landmarks chosen by Feighelstein et al. (2022) are shown in Figure 7, where it is apparent that the relevant facial AUs are around the eyes, ears, and mouth of the cat. Based on the facial landmarks of the cat, facial alignment is applied first to increase the performance of the model. To perform the facial alignment, the centers of each eye are calculated based on landmarks 37 and 38 for the left eye and 41 and 42 for the right eye. Then, the image is cropped such that the cat's face is in the center and rotated based on the angle between the eyes such that both eyes are on a horizontal line with the same y-coordinates. Lastly, all the images are scaled such

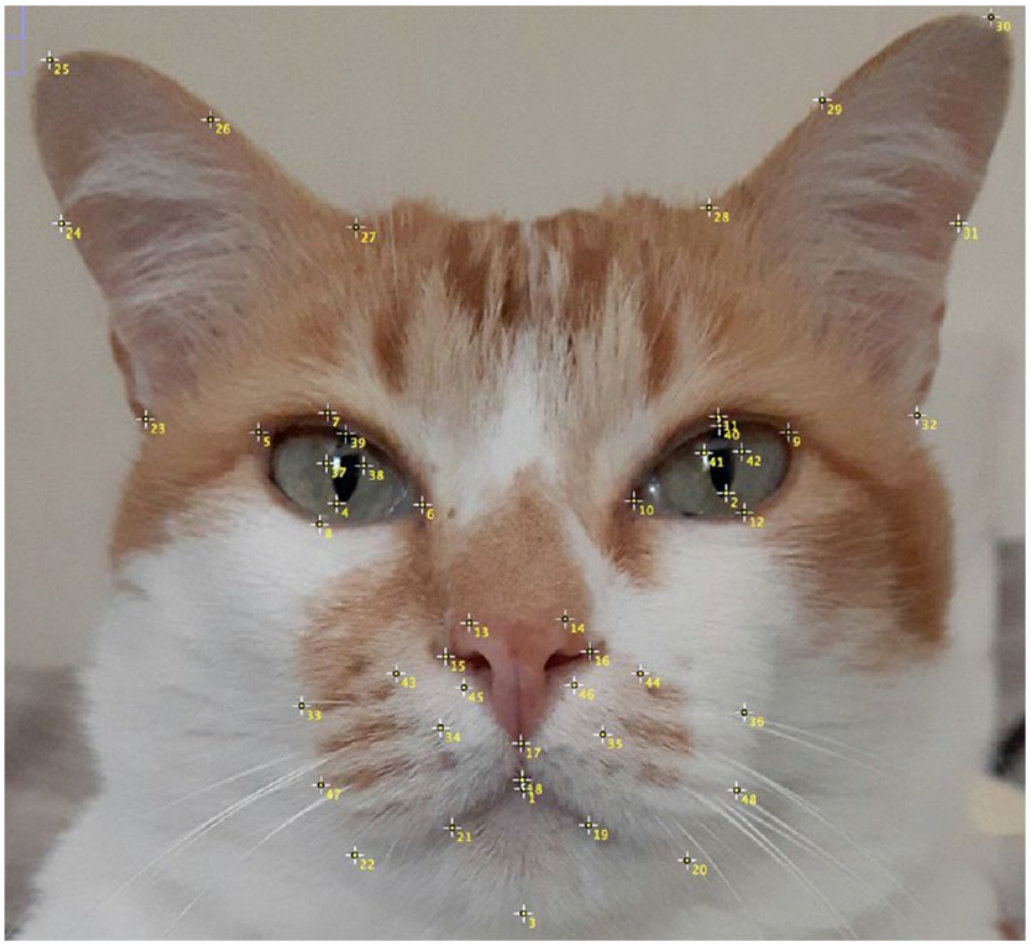


Figure 7. Figure from Feighelstein et al. (2022) which shows the locations of the 48 landmarks chosen based on their relevance to catFACS (Caeiro et al., 2017) for automated cat pain recognition with facial images.

that the size of the cat faces is the same in the entire dataset. The scaling factor is obtained by comparing the eye center distance between the original image and the image of the desired size. To obtain the facial landmarks of the output image, the eye center distance, scaling factor and rotation angle are used to perform the corresponding transformations and translations on all the facial landmarks of the original image as well.

After the facial alignment, the landmark-based approach takes multi-region vectors as input. Based on the landmarks shown in Figure 7, the cat face is divided into four regions: left eye, right eye, forehead, and the nose, mouth, and whiskers region. For all landmarks in these regions, they are normalized to the center of their corresponding region. This creates so-called multi-region vectors. After this, noise is injected into the vectors to obtain more samples and boost the generalization of the model. After this, a Multi Layer Perceptron neural network is used to train the model.

The second approach (Feighelstein et al., 2022) uses is a deep learning approach with a

pre-trained Resnet-50 model (He et al., 2016a). First, cat facial alignment is applied in the same manner to in the first approach, except for the transforming the landmarks in the last step. After the facial alignment, some data augmentation is performed to improve the model's generalizability. This is realized by randomly cropping, resizing, horizontally flipping with a 50% chance or a rotation with an angle in-between the range -90 and 90 degrees. Subsequently, the pre-trained Resnet-50 model is used, on top of which a sub-network of Linear layers with ReLu activation followed by Dropout layers is added. Dropout layers and their benefits will be further discussed in Section 2.8.

Steagall et al. (2023) develop a method to automatically detect pain in cat faces with mobile phones cameras with the aid of Feline Grimace Scales (FGS) (Evangelista et al., 2019). They propose a three-component system consisting of CNNs that predict facial landmarks in cats, computing geometric descriptors such as Euclide distances between AUs, and a FGS score predictor based on the results from the first two components. This research is vital since there is a great demand for people to be ablean to detect pain from mobile phone cameras. In this way, pet owners can detect their cat's pain in their own home where the cat is most comfortable. This removes the bias in the cat's behavior by bringing it to a stressful environment at the veterinarian.

2.4.4 Physiological analysis

In addition to pose and facial analysis of animals to detect pain, physiological factors can be evaluated as well. Although behavior analysis has been proven to be more reliable than physiological analysis as mentioned in Section 2.3, it can still provide insightful information.

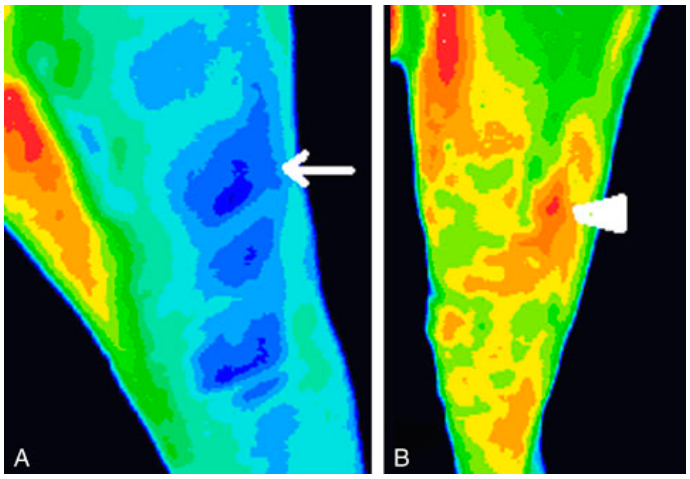


Figure 8. Figure from Infernuso et al. (2010) that shows a healthy joint in A and a CCL ruptured joint in B. It can be clearly seen that the arrow in A points to cooler regions, which indicate a healthy joint, and in B points to warmer regions, which indicate inflammation of the ligament.

One specific example of where physiological parameters can be insightful is for detecting

joint pain. In Infernuso et al. (2010) thermography was performed on dogs with healthy ligaments and dogs with cranial cruciate ligament (CCL) rupture. The CCL is an important ligament around the knee joint of dogs. In Figure 8 the authors show that with thermal imaging it was able to detect whether the dog had a healthy joint or a CCL rupture. Namely, the healthy joint in A appears a lot cooler in contrast to the inflamed CCL ruptured joint in B. While this might also be detectable from using temporal information as discussed earlier, using additional thermal cameras might increase the performance of a model in classifying joint pain.

2.4.5 Behavioral analysis

As far as we are aware, there are no automated behavior estimation methods with deep neural networks that aid in pain detection in dogs. However, a great deal of information is available on behaviors related to pain in dogs. One way to estimate pain in dogs with behaviors is by using the Melbourne Pain Scale (MPS), as shown in Figure 9. This scale encapsulates several behaviors related to pain next to a physiological analysis.

Some behaviors are abnormal postures, sitting/standing still with head up (also known as stargazing), standing with head down, restlessness in terms of pacing and getting up/down, and guarding painful areas when/before being touched. Guarding the painful area can include biting, licking, turning heads towards, and tensing muscles around the affected area (Hansen, 2003). Additionally, behaviors in a handler's presence may also indicate pain. For example, if a dog constantly seeks consolance from its owner, it might suggest that it feels vulnerable and seeks comfort (Mills et al., 2020). On the contrary, wary and aggressive behavior might also be indicators of pain. This usually greatly depends on the personality of the dog and the severity of the pain. Other behaviors that might be signals of pain are when a dog is resource guarding, such as its water bowl, a sudden unexplainable spike in fear/anxiety, destructiveness when left alone, disturbing its owner at night for no apparent reason, and reluctance to go on a walk or freezing during a walk.

Compulsive behaviors in dogs might also be indicators of pain, although not always (Mills et al., 2020). This includes the stargazing behavior, which is often associated with gastro-intestinal pain. Another compulsive behavior indicative of pain is "fly-snapping", which is a behavior where the dog seemingly bites into the air at unseen objects. This behavior is also an indicator of gastro-intestinal pain amongst others. Another compulsive behavior that may be a sign of gastro-intestinal pain is excessive and unexplainable licking.

One issue with behavior analysis in dogs is that most veterinaries can detect apparent signs of pain, such as decreased weight bearing from limbs, and acute pain signals, such as guarding painful areas. However, certain behaviors are so common in certain breeds

Category	Descriptor	Score
Physiological data		
a)	Physiological data within reference range	0
b)	Dilated pupils	2
c) *Choose only one:*	Percentage increase in heart rate relative to baseline	
	>20%	1
	>50%	2
	>100%	3
d) *Choose only one:*	Percentage increase in respiratory rate relative to baseline	
	>20%	1
	>50%	2
	>100%	3
e) Rectal temperature exceeds reference range		1
d) Salivation		2
Response to palpation		
a) *Choose only one:*	No change from preprocedural behavior	0
	Guards/reacts[a] when touched	2
	Guards/reacts[a] before touched	3
Activity		
a) *Choose only one:*	At rest- sleeping or semiconscious	0
	At rest- awake	1
	Eating	0
	Restless (pacing/getting up and down)	2
	Rolling, thrashing	3
Posture		
a)	Guarding or protecting affected area (includes fetal position)	2
b) *Choose only one:*	Lateral recumbency	0
	Sternal recumbency	1
	Sitting/standing, head up	1
	Standing, head hanging down	2
	Moving	0
	Abnormal posture (prayer position, hunched)	2
Vocalization[b]		
a) *Choose only one:*	Not vocalizing	0
	Vocalizing when touched	2
	Intermittent vocalization	2
	Continuous vocalization	3
Mental status		
a) *Choose only one:*	Submissive	0
	Overtly friendly	1
	Wary	2
	Aggressive	3
	Melbourne score	4

[a]Turning head toward affected area, biting, licking, scratching at the wound; snapping at handler; or tense muscles and a protective (guarding) posture.
[b]Does not include alert barking.

Figure 9. Figure from Hansen (2003) which shows the Melbourne Pain Scale where physiological symptoms and behaviors are denoted as possible signs of pain.

that the actual pain signals of these behaviors are overlooked. For example, a lot of pugs show an abnormal sitting position regularly. From research by Rohdin et al. (2018), it appeared that three-quarters of the pugs showed an abnormal sitting position, and also exhibited other pain signals such as reluctance to walk, inability to jump, and irritability. Other adjunctive behaviors less frequently associated with pain were also present, such as abnormal scratching of the head and neck and air licking. Due to this, pain signals like this are often overlooked in specific breeds.

2.5 Datasets and collection

One of the biggest challenges of developing automatic pain detection in animals is the scarcity of pain-annotated data sets, especially when comparing it to the great number of available human pain annotated data sets. Several reasons can explain this scarcity. One is

due to animals lacking the ability to verbally communicate their pain levels. Therefore, it is difficult to establish a ground truth for pain in animals, since they are not able to express what their pain levels are.

This section will first discuss the ethics of collecting pain footage from animals. Additionally, noisy labels and data, variability and imbalance in data, data augmentation, and three-dimensional data will be expanded upon.

2.5.1 The ethics of obtaining pain footage

An aspect that makes collecting pain annotated data difficult is how you can ethically make an animal experience pain. Inducing pain in animals is ethically questionable and can only be done to a certain extent without making the animal suffer needlessly. A natural way to capture pain in animals would be to record them after they had surgery and the anesthetics have worn off. This method is used by Feighelstein et al. (2022) where they assess pain in female cats before and after ovariohysterectomy. Another natural way to capture pain is used by Zhu et al. (2022) where both healthy and unhealthy dogs paying a visit to the clinic are recorded. This has the benefit that it is naturally occurring pain that is captured on video. However, the downside is that the dogs without pain and with pain are not the same dogs, as is the case in scenarios where the dog would have gotten surgery. This might make it more difficult for a model to learn exactly what pain signals look like instead of linking the pain class to a dog breed for example. A less natural way to capture pain is by conducting experiments. Experimental ways of inducing pain in horses have been discussed in Broomé et al. (2019), such as putting capsaicin on the skin of a horse, which causes burning pain and irritation to the skin, and putting a blood pressure cuff around the legs of a horse. These two methods to induce pain in horses are also used in human pain research and are ethically approved.

The advantage of conducting pain observation in these ways is that there is a way to capture footage of the subject before they are in pain and when they are in pain such as during post-surgery or after experimentally inducing the pain. In this way, you can say with more confidence that the subject is experiencing pain in the videos or photos. In the case where data without annotations is pulled from the internet without a controlled veterinary setting or controlled experiment, human annotators are required. The downside is that human annotators deciding whether an animal is in pain may unwittingly introduce bias while annotating a pain dataset (Chen and Joo, 2021).

2.5.2 Noisy data

Another challenge that occurs pain data collection is that, like in any other dataset, noise may occur. This may come in the form of bad lighting, occlusions of the animal, the animal moving out of the camera frame etc. Additionally, the presence of the owners, veterinary

experts or other subjects can influence the behavior of the animal due to which it may mask its pain signals. For example in the case of dogs, the dog may wag its tail happily in the presence of its owner, while it might keep its tail between its hind legs in the absence of its owner.

Another way to reduce noise is to perform undersampling as was done in (Feighelstein et al., 2023a) for automated pain recognition in rabbits. This method calculates a confidence level for each frame which determines how confident the model is in classifying the concerned frame in to the pain/no pain class. Consequently, the top n frames are chosen and used to train a second, improved model. The removal of frames of which the model is not that confident and likely include noise, has a positive impact on the performance of the model. Removing such frames likely removes noise where the view of the rabbit is obstructed due to the metal bars or not facing the camera, blurry frames due to movement of the rabbit and other kinds of noise which decreases the confidence level. The performance of the second improved model has a higher accuracy than the first model. This shows that the model trained on the top frames in comparison to all frames gained more informative pain signals from the top frames than the model trained on all frames. It seems that the noisy frames are confusing the model trained on all frames what signals are informative pain signals in rabbits.

Feighelstein et al. (2023a) also suggest that this method of frame selection in videos could replace the manually selection frames from videos, such as done in related works of Feighelstein on cats (Feighelstein et al., 2022, 2023b).

2.5.3 Noisy labels in datasets

In the last years massive labeled data sets have been known to produce impressive results for all kinds of machine learning tasks. However, this is no free lunch (Wolpert and Macready, 1997). Manually labeling data sets of vast sizes is an incredibly time-consuming process, which is prone to flaws and does not guarantee quality annotations. Human-made annotations suffer from poor reproducibility and reliability. This is mainly caused by the limited attention capacity of the human annotators. Once an annotator grows tired throughout the annotation process, it gets harder to focus on annotating the videos (Harris et al., 2023). This causes more variation in the quality of the annotations and, in turn, reduces the quality of the resulting data. Variation in the quality of annotations also occurs when videos are annotated by different people, or when a new researcher just started as a beginner and is being trained (Bohnslav et al., 2021). Additionally, certain data labels can be hard to annotate due to their complex nature, even for experts of the concerned domain (Frénay and Verleysen, 2013). To assist the time-consuming process of labeling data, other public sources annotated by non-experts are used as well, but often result in

unreliable labels due to their non-expert nature (Song et al., 2022). In this thesis, we encounter some of these problems as well. We annotated the behaviors ourselves as inexperienced annotators of dog behaviors; we are prone to introduce bias into the dataset. Especially since we are not trained in annotating datasets, it is hard to remain focused on the annotation process and not introduce errors.

2.5.4 Variability in data

Animal pain data has to deal with great variability. Animals can be recorded in a veterinary setting, outside in the pasture, at home or other places. Some data could be recorded during sunset while other data is recorded during the day. Additionally the appearance of animals may vary a lot as well. For example, horses usually wear a halter when being recorded, which might have different colors. Animals also have fur markings which vary from individual to individual. Specifically for dogs, in contrast to humans, they vary a great deal in appearance across different breeds. Some breeds are covered in thick amounts of hair, which would make it difficult to spot subtle muscle contractions and facial movement, and certain pain-related cues might be more prominently visible than in other breeds. There exist dog breeds with extremely broad, short noses, such as pugs, and extremely long noses, such as greyhounds. This makes it extra difficult for models to recognize different behavior cues in different breeds of animals (Finka et al., 2019).

2.5.5 Imbalanced data

Due to the difficulties of collecting animal pain data, the problem of imbalanced data is significant. It is easier to collect data from the baseline class, where animals do not experience pain, as opposed to the pain class. This causes the data imbalance in animal pain data sets. When the data imbalance in a dataset is not dealt with, it may negatively impact the performance of a model. For example, the model might learn to simply predict the majority class and still achieve a decent accuracy despite not having learned to detect pain in videos. In the dataset used for this thesis, there are also more dogs without pain than with pain, which makes the dataset imbalanced.

The most common solution against this problem is by oversampling the minority and undersampling the majority class. In (Feighelstein et al., 2022) the method of random undersampling and oversampling is used, where data images of the non-pain class are selected randomly and removed until there is an equal number of images for the pain class. Another practice to battle data imbalance is by avoiding only reporting the performance of the machine learning model on the accuracy score and ensure to incorporate the F1-score. The F1-score namely requires the model to perform both on the majority and minority class, in contrast to accuracy (Broomé et al., 2019). Another approach is to use data augmentation, which will be discussed in the following section.

2.5.6 Data augmentation

Data augmentation is a solid approach to battle the data imbalance and data scarcity issues in animal datasets, specifically when dealing with animal pain datasets. In (Hummel et al., 2020) data augmentation was used to try and reduce the label distribution imbalance since most pain indicators were present on the nostrils, eyelids, and mouth. The training set was augmented by flipping the image vertically and adding noise to the facial landmarks. Other ways of using data augmentation to increase the data size of the dataset include but are not limited to: horizontally/vertically flipping the images, cropping the image randomly, scaling, rotating, inverting, etc..

2.6 Automated annotation of animal video data

Labeled video pain datasets of animals are scarce and manually labeling videos is a time-consuming and expensive process. In addition, when a video is manually annotated, often only a few behaviors at the same time can be annotated, since humans are not able to handle hundreds of different kinds of annotations at the same time (Bohnslav et al., 2021). With manual annotation it often occurs that researchers only note how many times a certain behavior occurs and for how long, but not during which exact video frames. This prevents the possibility of determining transition probabilities between certain actions, which can be fruitful information for behavior analysis as well. All of these reasons that might reduce the quality of annotations can be causes of noisy labels as described in the previous Section 2.5.3.

Automated annotation, where a researcher can define an ethogram of interest to be annotated, would greatly reduce the cost and time it takes to annotate video data and increase the quality of the annotations simultaneously. Additionally, when each video frame is annotated, it would be easier to calculate transition probabilities between behaviors or link certain behaviors to emotions or pain, for example.

In order for research results to be reliable it is important that papers of automated behavior annotation do not only report an accuracy measure. This is due to that a model can still achieve excellent accuracy when classifying rare behaviors (Bohnslav et al., 2021), namely when the model only classifies all the rare behaviors as one of the common behaviors by just assuming the rare behavior does not occur in the frames. For example, if the rare behavior occurs two percent of the time and the model simply ignores its existence, it can still get an accuracy of 98 percent, while it did not even learn how to classify rare behaviors. Due to this, it is important that papers about automated behavior classification do not only mention accuracy, but metrics such as recall, precision and F1 score as well. We come across this as well during our experiments, which will be elaborated on in the Experimental

Evaluation. Several methods that cover automated annotation will be discussed in this section, which is relevant to our thesis since we want to reduce the time-consuming process of annotating behaviors in dog videos.

2.6.1 JAABA

One way to tackle automated annotation is by using an approach with pose estimation, which is also often called skeleton-based action detection in the computer vision world. One early method which made use of this approach is called JAABA by Kabra et al. (2013), which is an abbreviation for "Janelia Automatic Animal Behavior Annotator". JAABA uses the output of tracking algorithms as its input, in (Kabra et al., 2013) two inputs that use ellipses were demonstrated. The JAABA classifier subsequently uses the GentleBoost (Friedman et al., 2000) learning algorithm. Iteratively, GentleBoost adds weak rules for each iteration that improves the distinction between two classes in the training data. Weak rules, in this case, are decision trees that classify a single feature based on a threshold. GentleBoost combines an abundance of these weak rules and consequently produces an accurate classifier (Kabra et al., 2013).

JAABA uses an active learning-based approach where the end-user iteratively labels the frames of the output from the classifier which the end-user believes are the most informative for the classifier and thus creating an interactive experience. Additionally it enables the end-user to check frames which the classifier gives a low confidence score, and correct the label if necessary. After this the classifier is retrained until the end-user is content with the results the classifier provides. JAABA was able to mirror manual annotation of humans with an error rate of 0.6%, while in contrast, different human annotators had an inter-disagreement rate of 2.4%. The inter-disagreement rate is a measure that shows to which extent raters give different ratings to the same behaviors.

2.6.2 DeepEthogram

In Bohnslav et al. (2021) DeepEthogram was introduced. In contrast to the methods that use pose estimation as a base to detect behaviors, this approach tries to classify behavior from raw pixel values of videos. One benefit is that this approach requires only human annotations for an ethogram of interest, in contrast to needing the ethogram of interest together with manually labeling body key points in order to build a skeleton for pose estimation. DeepEthogram is a modular pipeline based on supervised deep learning that annotates each separate video frame with a behavior defined in an ethogram, which is given by the user. For each video, DeepEthogram generates a binary matrix where it defines for each frame whether a certain behavior in the ethogram was present or not. The model is based on Hidden Two-Stream Convolutional Neural Networks for detection of spatial and Temporal Gaussian Mixture (TGM) (Piergiovanni and Ryoo, 2019) networks for action detection in videos.

DeepEthogram is able to perform well on little human-annotated training data. Its performance for accuracy and F1-score measures seemed to stagnate after training on 12 human annotated videos of about 20 minutes. Although this depends on how complex the annotation task is. Additionally, the whole program requires little programming from the end-user. A graphical user interface is used to manually annotate training data, train the model and eventually make predictions.

2.6.3 DeepAction

In (Harris et al., 2023) the open-source toolbox called DeepAction is introduced. DeepAction is a deep-learning based toolbox which can be used inside MATLAB to automatically annotate animal behavior in videos. Like DeepEthogram, DeepAction uses a two-stream architecture with a CNN and RNN to generate behavior annotations in animals from video frames. The CNN stream is used together with dense optical flow to learn spatial and temporal features from the video frames in a two-stream architecture. The spatial features are extracted from the raw video frames. The temporal features are extracted by a dense optical flow method which is used to capture movement in pixels in consecutive frames.

These features are subsequently used in a long short-term memory (LSTM) network to predict animal behavior. The classifier must be trained on manually labeled data but has shown to perform well on little training data, although it might have difficulties with detecting minority classes. Additionally, the video clips are split into smaller parts because then the model is able to generalize better in contrast to using long video clips and thus reduces overfitting and sequence padding.

Further, Harris et al. (2023) uses a confidence score similar to as used in Feighelstein et al. (2023a) mentioned earlier. This, in turn, reduces the time when researchers want to check the annotations the model has produced; high confidence scores have to be checked less often than fairly low confidence scores. We ran some pilot studies with DeepAction, which can be viewed in the Appendix.

2.7 Explainability

Explainability in pain detection is crucial since veterinary experts desire to know why a model determines whether an animal is in pain or not, and on what it focuses to come to that conclusion. Due to the black-box nature of deep learning algorithms, this poses challenges. With the help of explainable AI this problem can be tackled to a certain extent by providing methods that can make the decision the AI model makes clearer, more intuitive, and more interpretable to the human viewer (Das and Rad, 2020). Using

explainable AI (XAI) increases the trust and transparency of the models and helps to expose any bias or unfairness the model may unwittingly exercise.

In general, there are two types of XAI which will be discussed in this section: *data-focused* and *model-focused* approaches. Data-focused approaches focus on how the data affects the performance of the model, while model-focused approaches focus more on the internal structures of a model itself regardless of the data. In addition, hand-crafted features will be touched upon, since these can grant great insight into how AI operates as well.

2.7.1 Data-focused approaches

Occlusion

One example of a data-focused approach could be omitting certain parts of the data to discover which parts of the data the model relies the most on. In Feighelstein et al. (2023b), the authors occlude different parts of the cat's face to discover which parts of the face the model uses the most to determine whether the cat is in pain or not. The authors explore the effect of three different facial regions of cats: the ears, eyes, and the mouth.

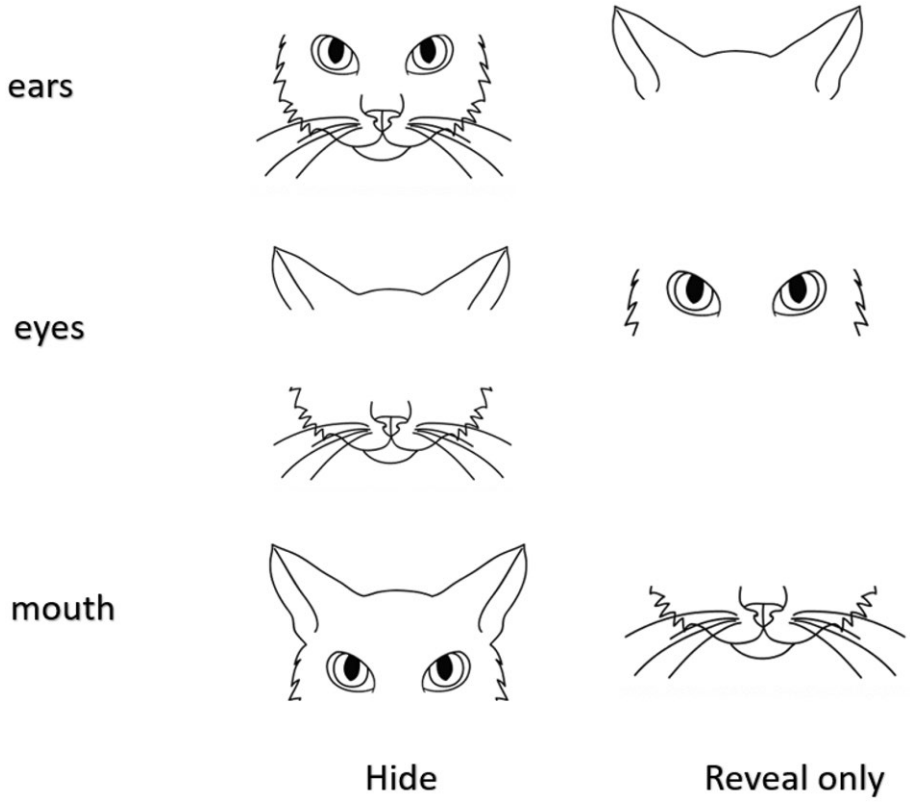


Figure 10. Figure from Feighelstein et al. (2023b) that shows different hide and reveal options.

The authors define three different methods of occlusion on which they test: "Full information, reveal region R, and hide region R." The model is trained on either the entire region, and thus has the full information, or only on region R, or on all regions except region R, respectively. Examples of different options for hiding and revealing regions can be seen in

Figure 10.

In Feighelstein et al. (2023b), two different pipelines are used: a landmark-based approach and a deep-learning approach. For the landmark-based approach, vectors that represent the landmarks are either included or excluded when they belong to a certain facial region. For the deep learning approach, the units of measure are raw pixels that are included or excluded based on the facial landmark location by using masks.

The results in (Feighelstein et al., 2023b) show that the mouth region has a significant influence in determining whether a cat is in pain or not, while ears do not seem to have a great impact. Namely, using the mouth region only has good accuracy in comparison to hiding the mouth region, all relative to using the full information. In contrast, using the ear region only has a bad accuracy drop in comparison to hiding the ear region.

Shapley values

Shapley values are a data-focused approach that can aid in clarifying how much influence each feature has on the predictions of a model. Originally, Shapley values stem from cooperative game theory Kuhn and Tucker (1953) and have become popular to use in machine learning to make the outputs of the machine learning models more explainable (Rozemberczki et al., 2022; Sundararajan and Najmi, 2020). We will use Shapley values based on the behaviors to make our model more explainable in why it classifies pain. In cooperative game theory, Shapley values are a way of distributing the total payout of a coalition game over its players, where the contribution of each player is determined by averaging the marginal contribution of all possible coalition games the concerned player can be a part of. In machine learning, the game is the prediction task; the players are, in essence, the features, and we look at each feature's contribution to the model's output, where the model's output is the so-called payout.

All the possible coalition games in machine learning can be translated to all the possible subsets of features. The Shapley value is the value that is obtained when averaging all the marginal contributions of the feature over all possible coalitions. To obtain the marginal contribution of a feature on a coalition, the output of a coalition must be calculated excluding and including said feature. The difference between the excluding and including prediction is the marginal contribution. The marginal contribution is shown in Equation 2.6 where f represents the machine learning model and S the coalitions excluding feature i. Repeat this process over all possible coalitions without said feature, and calculate the weighted average of all the marginal contributions to obtain the Shapley value of the concerned feature. To ensure a fair distribution the marginal distributions are weighted. The weight of a feature is shown in Equation 2.7 where n represents the number of features

and S are the coalitions excluding feature i. The final Shapley value depicted by ϕ is obtained by Equation 2.8 which includes all of the above equations.

$$\text{Marginal Contribution feature i} = f(S \cup \{i\}) - f(S) \tag{2.6}$$

$$\text{Weight feature i} = \frac{|S|! \cdot (n - |S| - 1)!}{n!} \tag{2.7}$$

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|! \cdot (n - |S| - 1)!}{n!} \left(f(S \cup \{i\}) - f(S)\right) \tag{2.8}$$

The advantages of Shapley values are that they are model-agnostic (Molnar, 2022). This means that a Shapley value can be applied on any kind of model, not just a linear model. Additionally, Shapley values comply with the axioms of efficiency, symmetry, dummy and additivity (Rozemberczki et al., 2022). Efficiency as shown in Equation 2.9 provides that the sum of the Shapley values of all the features equals the Shapley value of the coalition in which all the features are present. This ensures that the Shapley values of the separate feature all add up to the total prediction (total payout) of the model. Symmetry shown in Equation 2.10 assures that the features that deliver an equal contribution to the output, will have the same Shapley value. This ensures that each feature is treated equally and eliminates bias to particular features. Dummy as shown in Equation 2.11, makes sure that a feature that does not contribute to the output, receives a Shapley value of 0. This has the good property that it ignores noisy irrelevant features in the distribution of contribution. Lastly as shown in Equation 2.12, additivtiy entails that the Shapley values of a combined model will be the same as the sum of the Shapley values of the combined models individually. Adhering to these axioms make Shapley values are a great explainability tool for machine larning model, which ensures a fair distribution of the contributions of the features. However, although Shapley values in theory adhere to these axioms, when using machine learning models with non-linearities. Namely, the additive property of Shapley values assumes that features can be summed, but this may not always hold amongst features that interact in a non-additive manner (Rozemberczki et al., 2022).

$$\sum_{i \in N} \phi_i = f(N) - f(\emptyset) \tag{2.9}$$

$$\text{If } f(S \cup \{i\}) = f(S \cup \{j\}) \text{ for all } S \subseteq N \setminus \{i, j\}, \text{ then } \phi_i = \phi_j \tag{2.10}$$

$$\text{If } f(S \cup \{i\}) = f(S) \text{ for all } S \subseteq N, \text{ then } \phi_i = 0 \tag{2.11}$$

$$\phi_i(f + g) = \phi_i(f) + \phi_i(g) \tag{2.12}$$

One of the main disadvantages of Shapley values is that it has a high complexity, which makes them computationally expensive to calculate. Additionally, the computation cost increases exponentially the more features a model encapsulates (Rozemberczki et al., 2022). Another disadvantage is that Shapley values can assign higher contributions to features that are correlated (Molnar, 2022). Furthermore, to interpret Shapley values, human judgment and domain expertise are required, which can lead to misinterpretations if the interpreter does not possess sufficient knowledge to do so.

2.7.2 Model-focused approach

An example of a model-focused approach is using random forests to extract feature information from the model, as done in (Feighelstein et al., 2023b) for their landmark-based approach. Here, they use the Gini Importance to calculate the feature importance of each facial landmark. An example is shown in Figure 11 where the most important landmarks are colored red.

Another example for model-focused approach is using the GRAD-CAM method which is used in Zhu et al. (2022); Broomé et al. (2019); Feighelstein et al. (2023a); Boneh-Shitrit et al. (2022a); Feighelstein et al. (2023b). This method is a commonly used approach to visualize what models are paying attention to. The GRAD-CAM method generates a heatmap of an image and colors the pixels to which the model pays the most attention to a red color, while the pixels to which the model pays the least attention are colored blue. An example of a heatmap can be seen in Figure 12.

In Ferres et al. (2022), where emotions of dogs are assessed with the help of DeepLapCut, the authors use a decision tree to give more insight into what poses the model links to what kind of emotions in dogs. As explained earlier, Ferres et al. (2022), use pose metrics in order to determine the emotions of dogs. The decision tree shows that the tail position of the dog is a significant factor in determining its emotional state. A lower position of the tail tends to imply that the dog is experiencing a negatively associated emotion such as

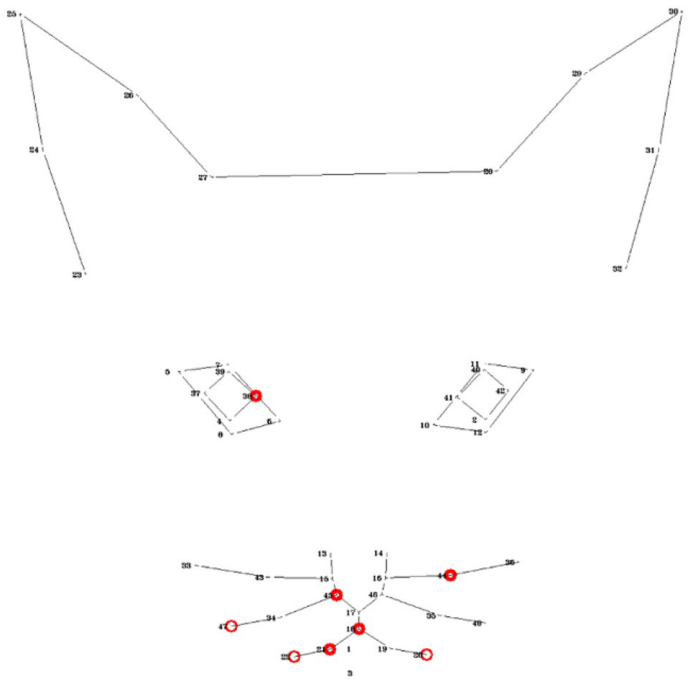


Figure 11. Figure from Feighelstein et al. (2023b) that shows the most important landmarks of the cat face for pain detection by coloring them red.

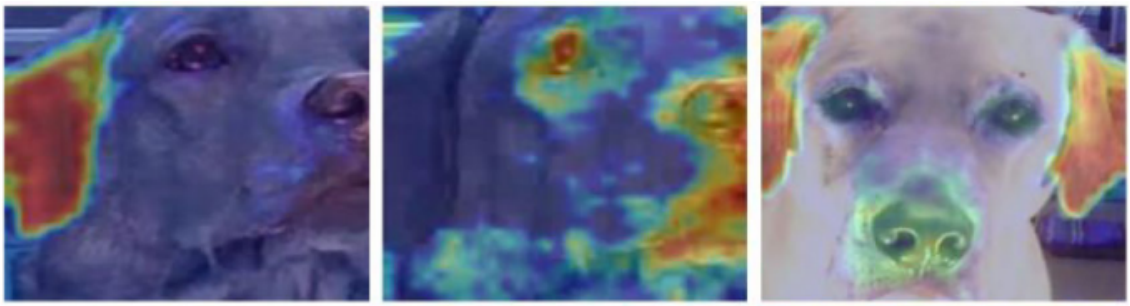


Figure 12. Figure from Boneh-Shitrit et al. (2022a) that shows where in the image the model focuses the most on while classifying the image. It is clear that the models focuses mostly on the ears and mouth region of the dog.

anger or fear, while a high position of the tail suggests that the dog can be either happy or angry. The latter, in turn, depends on whether the dog has their mouth open or closed. Decision trees have also been used in human-based research to explain the decision of computer vision models. In Kaya and Salah (2018) a decision tree was used to predict personality traits, which could be used in a similar way when predicting an ethogram.

2.7.3 Hand-crafted features or learned features

Nowadays there exist both hand-crafted features, and features learned by deep models. In the earlier days, mainly hand-crafted features were used since these require only small datasets, while features obtained by deep learning require datasets of vast size. Learned features may be more accurate than hand-crafted features, but hand-crafted features have the advantage of having better explainability and allow for a better understanding of how

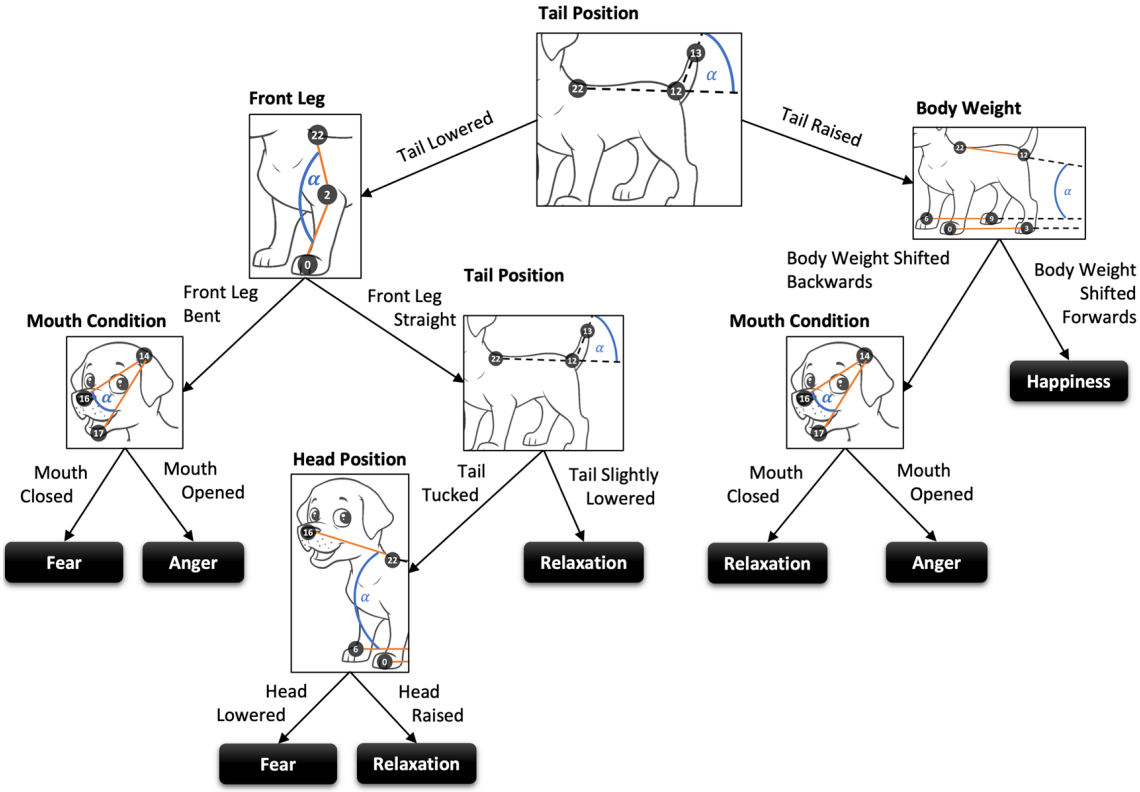


Figure 13. Figure from Ferres et al. (2022) which shows a decision tree that depicts with which signals a dog is most likely experiencing certain emotions.

the method works on the inside. This is in contrast to learned features, which have to deal with their black-box nature and thus often lack a sufficiently clear explanation of how the method exactly operates. Hence, hand-crafted features are more appropriate in a clinical setting for cases such as pain assessment. Namely, in a clinical setting, the explanation based on what cues the conclusion that the animal is experiencing pain is desirable to have.

There exist two kinds of hand-crafted features, namely *low-level* and *high-level* features. Low-level features are typically more technical features and include notions of image statistics such as the intensity of a certain pixel, the gradient of the image etc. High-level features, on the other hand, tend to have a more semantic meaning and are often fine-tuned to a specific species. For example, a dog wagging its tail usually implies that the dog is in a happy mood, while when a cat is swiping its tail, it usually indicates a bad mood. In Ferres et al. (2022), high-level hand-crafted features were used in the form of the pose metrics which were manually established, such as the angle between the tail and spine. High-level hand-crafted features are referred to as *intermediate representations* in Broome et al. (2023) due to these features promoting explainability.

There exist a great deal of hand-crafted features which have also been mentioned earlier in this proposal. *Facial Action Units* have been created for some species such as horses,

cats, sheep, mice, and macaques and help us understand which expressions of an animal are related to what kind of internal state. *Landmarks* and *keypoints* can be both facial and pose related. Facial landmarks were used by Feighelstein et al. (2022) to recognize cat pain, where the locations of the landmarks were determined based on their relevance to facial actions units of cats. Keypoints for pose analysis were used by Ferres et al. (2022) and were automatically extracted with the use of DeepLabCut (Mathis et al., 2018). The work of Zhu et al. (2022), which this thesis will be extending is partly based on pose representations, which are skeletons that represent the pose of either a human or an animal. In Zhu et al. (2022) this hand-crafted feature method of pose representation is combined in a recurrent two-stream network with another stream that processes raw RGB video frames and learns features with a black-box nature. In this way both hand-crafted and learned features are combined to automatically assess pain in dogs, which is a popular method as was mentioned in Section 2.2. Since we will be adding pre-defined behaviors to this exisitng two-stream pain estimation model, we will be adding even more hand-crafted features.

2.8 Good measures to increase performance and avoid overfitting

Computer vision within fields with data scarcity problems has to deal with the challenge of overfitting. In the field of automated pain recognition, there are few subjects to go around and take samples from. However, computer vision models require high dimensionality data to perform decently (Broome et al., 2023). Due to this, the risk of bias and overfitting of models is high. This section will briefly discuss several approaches which can help reduce the risk of overfitting and enforce the generalizability of the model.

One critical practice to enforce generalizability to unseen subjects is using the subject-exclusivity approach as recommended by Broome et al. (2023). This approach separates individual subjects into the train, validation, and test sets so that no individual occurs in multiple sets. An individual used in the train set will thus not be used in the validation or test set. Another possibility is using a leave-one-subject-out approach where an individual subject is chosen to be left out of the training and validation set, and solely occurs in the test set to enforce generalizability. A double leave-one-subject-out approach is also possible, where two subjects occur only in the test set.

Another best practice recommendation by Broome et al. (2023) is to use k-fold cross-validation. For small datasets with no repeating subjects and fewer than a thousand instances, 10-fold cross-validation is recommended. When subjects have a lot of repetition, a nested 10-fold cross-validation is recommended.

In Harris et al. (2023), it was mentioned that models could generalize better when short video clips are used in contrast to long video clips, which helps reduce overfitting. The size of the training batch sizes can also affect the model's performance and is a trade-off. Namely, larger batch sizes can reduce the training time of the model but generally produce lower accuracy and increase overfitting. In contrast, small batch sizes take a longer time to train but have higher performance.

Pooling is a commonly used method to reduce a model's dimensionality and make it more robust against variations in the data. Pooling is often used in CNNs after a convolutional layer to reduce dimensionality while preserving the depth of the image. Pooling is realized by sliding a filter over the image and conducting an operation on the features underneath the filter. This is done separately for each channel, such as RGB channels, which preserves the depth of the image. Therefore, pooling is a good way to achieve translation invariance, defined as when an object's location in an image does not influence the output of a neural network. Namely, the location of an object in the image does not matter since it is detected anyhow when a pooling operation is used. There are different kinds of pooling, such as *max pooling*, *average pooling*, and *global pooling*. Max pooling operates by selecting the maximum value underneath the filter, which ensures an output where the most salient features are present. This can help in selecting the most essential features in a channel. The average pooling method takes the average of all the values underneath the filter and preserves more of all the information in contrast to max pooling. Global pooling takes a single value from the entire channel and reduces the channel to the chosen single value; the single value could be the average or the maximum value, for example.

Despite the advantages, there are also some disadvantages to using pooling layers. Pooling reduces the dimensionality of a model, which is great for reducing computation costs, but information loss can happen in the process. This poses the risk of fine-grained details necessary for accurate classification getting lost. Additionally, there are extra hyperparameters that require tuning, such as the size of the filter.

Dropout layers (Srivastava et al., 2014) are a form of regularization that helps prevent overfitting and are used during the training phase. When training a neural network, neurons in the layers can become co-dependent on each other. This means that neurons can pass on mistakes made in previous layers or even merely pass on the results of an earlier layer without changing something, which may cause overfitting. Dropout layers are layers in-between neural layers that set a certain percentage of input values of neurons to zero. By doing this, the phenomenon of neurons relying on other neurons in previous layers is reduced. They are forced to learn their own patterns since they cannot assume neurons in other layers even exist. The percentage of input values to neurons being set to zero is

determined by the probability of retaining a neuron. A retain probability often used for hidden layers lies around 0.5, and for visible layers, a probability close to 1, such as 0.8, is commonly used and recommended by Srivastava et al. (2014).

2.9 What is missing?

When reviewing what we have learned from the Related Work section we come to the conclusion that there is little work on accurate pain estimation in dogs. As far as we are aware, the only work addressing this issue is still by Zhu et al. (2022). There is mainly room for improvement in the model by Zhu et al. (2022) in terms of explainability, namely apart from the Grad-CAM approach, there is no transparency in the model. This gap in explainability hampers the trust in pain estimation models and prehibits it from being used in practice, since understanding the rationale behind a model's decisions in pain estimation is crucial. Veterinarians must be able to explain why a dog is experiencing pain, even if it uses neural networks to come to that conclusion.

We aim to improve this by adding behavior estimation to the streamline of the model. These hand-crafted features can aid greatly in estimating pain and providing an intuitive insight at the same time. With behaviors added, we can add a lot of explainability methods, such as decision trees and Shapley values.

For the action recognition part of the behavior estimation we can conclude that zero-shot networks based on CLIP models show great performance without requiring training on millions of instances. Well-performing models that can operate effectively on small datasets are crucial in our case, which is why our focus is on CLIP-based networks during our pilot studies and final model.

3. Methodology

The pain prediction model we propose in this thesis comprises several parts, shown in Figure 14. In essence, three separate streams are concatenated to produce a pain score that indicates how confident the model is that the dog is in pain. We are building upon the two-stream model proposed by Zhu et al. (2022), which only consists of the LSTM and ConvLSTM streams.

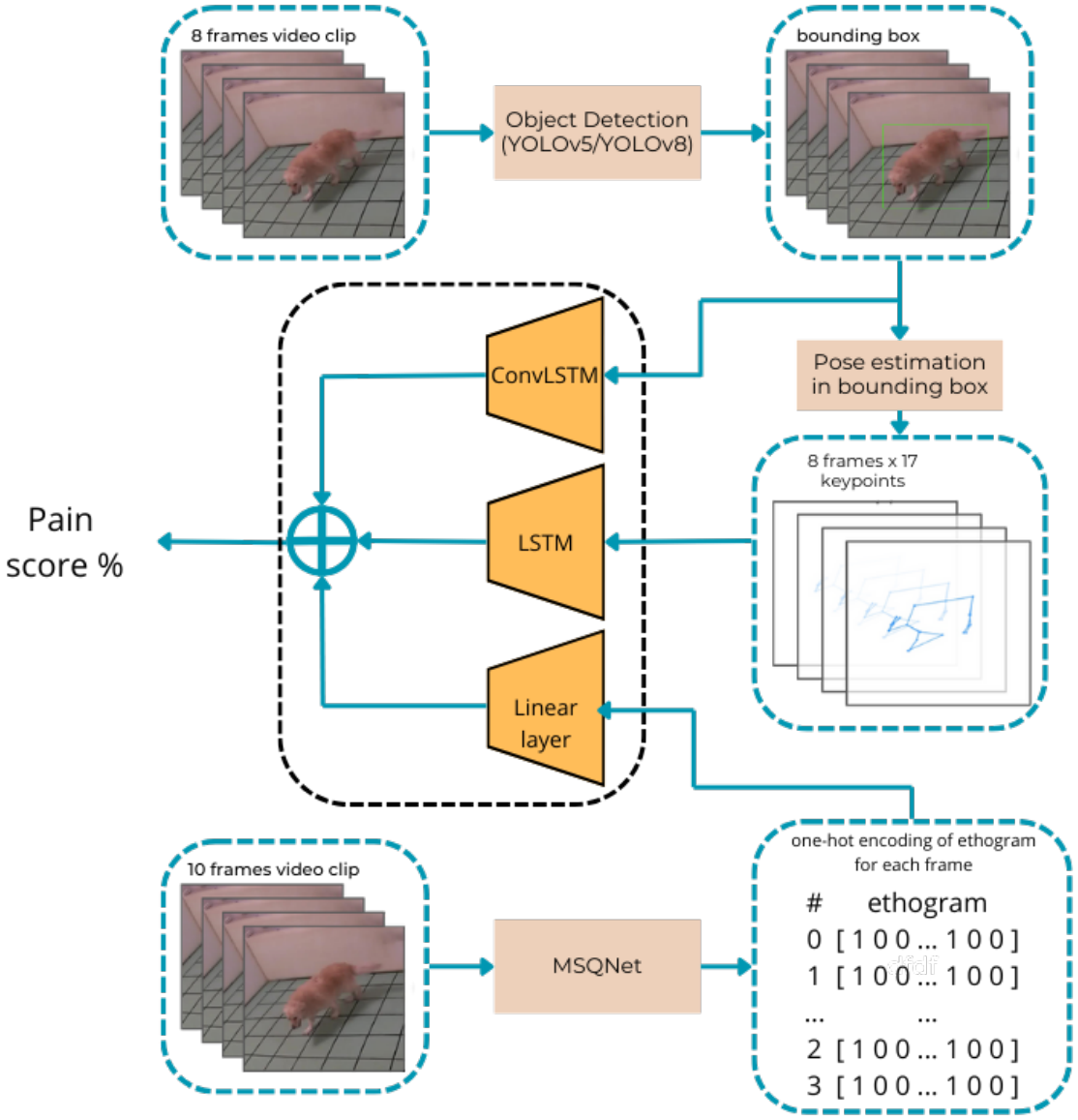


Figure 14. Figure that displays an overview of the newly proposed pain estimation architecture, inspired by the two-stream model picture of Zhu et al. (2022)

For each video frame, the dog's bounding box is detected and cropped accordingly. These frames are put into a pose estimation algorithm to detect the key points on the dog's

body and subsequently processed further for the pain estimation model. In addition, these cropped frames are used for our behavior estimation algorithm: MSQNet. The output of MSQNet is an ethogram of every single frame is extracted as a one-hot encoding, where the one-hot encodings represent whether a behavior is present in this particular frame or not. Subsequently, the ethogram features are used to train a random forest classifier or put into a linear layer.

The pain estimation model has three streams. One stream processes the raw cropped RGB frames obtained from the YOLO algorithm with a ConvLSTM neural network. Another stream processes the keypoints with an LSTM neural network. For processing the ethogram stream, we will use two methods and later on evaluate which one performs better than the other. Therefore, the raw ethogram one-hot encodings are either processed through a linear layer to obtain ethogram embeddings, or the trained random forest classifier is used to obtain prediction embeddings. Ultimately, these three streams are concatenated, and the resulting feature vector is used to predict the dog's final pain score with a fully connected layer.

This chapter first discusses the object detection methodology in Section 3.1. After this, behavior estimation with the off-the-shelf method MSQNet is addressed in Section 3.2. Subsequently, the pose estimation algorithm is discussed in Section 3.3. Further, the pain model is addressed, where all the separate components come together. Finally, we will discuss several explainability methods that aid us in giving an insight into how the model works.

3.1 Object Detection

It is fruitful to employ an object detection network since, in most of our datasets, a static camera records the entire room where the dog is present, meaning that the dog is only present in a small portion of the frame. By detecting the dog's bounding box coordinates, these coordinates can be subsequently used to crop the image of the dog accordingly. By cropping the image according to the bounding box coordinates, background noise from the entire room is reduced to merely around the dog. In addition, the original frame sizes of the dataset used are 2K resolution. These visuals are too large for most neural networks to process in a reasonable time and require immense computing resources, which is unrealistic. Thus, detecting the dog's bounding box and subsequently cropping the image accordingly prevents useful features from becoming lost while resizing the image. It ensures that the dog is as large as possible and reduces background noise, which directs the model's attention to the dog. Additionally, when the dog is moving throughout the room, its body size changes with respect to how close it is to the camera. By detecting

the bounding box, cropping the image, and resizing it according to the required size for the neural network, it ensures that the dog remains in the center of the frame and is of even size all the time. The object detection model will be employed as an image-filtering technique as well. If the model cannot detect the dog due to a person standing in front of it or the dog being occluded in any other way, it will produce no bounding box for the dog, and the frame will be discarded. In this way, these noisy frames can be filtered.

To determine the dog's position within the frame is called region of interest (ROI) extraction within the computer vision field (Amit et al., 2021). We realize this with object detection networks. Object detection networks generally have two primary goals. The first is detecting what kinds of objects are within a frame, e.g., a dog and a human. The second one is extracting the extract coordinates of the bounding box that the detected object is present in.

Similar to the approach proposed by Zhu et al. (2022), object detection with the means of the YOLO object detection network is used for the first stage of the behavior and pain estimation model. We chose not to use a different object detection network since YOLO is known for its high precision and recall rates, which means it is proficient in accurately identifying objects in video frames. Additionally, YOLO provides an abundance of pre-trained models that aid in accurate classification. YOLO is also known to be user-friendly and has excellent documentation, which eases the use of this object detector (Jocher, 2020; Jocher et al., 2023).

In contrast to Zhu et al. (2022), we will use both YOLOv5 (Jocher, 2020) object detection network and the newer generation: YOLOv8 (Jocher et al., 2023). One of the most significant improvements from YOLOv8 over YOLOv5 is that YOLOv8 enables anchor-free detection of bounding boxes around the objects, whereas YOLOv5 uses anchor boxes. Anchor boxes are boxes of various pre-defined scales in an image, which allows it to detect objects of all kinds of sizes. If an anchor box detects an object in its box, it will apply offsets to the bounding box to encapsulate the detected objects entirely. On the other hand, the anchor-free object detection of YOLOv8 learns to predict the center points of the objects and draw a bounding box around the object's center point. This generally leads to higher accuracy and speed due to the model being less reliant on pre-defined anchor boxes and a reduction of computational complexity. However, since our dataset is subject to noise and YOLOv8 might have trouble directly predicting the center point of the dog, we will run both YOLOv5x and YOLOv8x object detection models and automatically pick for each video which model was able to predict the most frames. Both models are run to predict only one object class: a dog. If no object of class dog is detected, the image frames will not be further processed and discarded. To ensure the entire body of the dog is

captured in the bounding box, we will increase the bounding box by 10% before cropping.

3.2 Behaviour Estimation

For the behavior estimation part of the pipeline, an off-the-shelf method is chosen as proposed by Mondal et al. (2023) called *Multi-modal Semantic Query Network (MSQNet)*. This model has already been partly discussed in the related work section. As already mentioned, this model comprises three main parts, namely (1) a "*spatio-temporal video encoder*" that captures detailed spatial and motion features, (2) a "*multi-modal query encoder*" component that captures visual information from video frames and semantic information from action classes and combines these and (3) a "*multi-modal decoder*" that uses attention mechanisms to process the video encodings and multi-modal queries further. An overview of the model is shown in Figure 3 in the related work section. In this section, each of the three main components in the model's architecture and the overall functionality of the model will be explained in detail.

3.2.1 Model architecture

Spatio-temporal video encoder

The spatio-temporal video encoder serves the purpose of capturing relevant motion cues and temporal sequences throughout the entirety of the video. This ensures that not only temporal information between consecutive frames is captured but also between non-consecutive frames with more significant gaps between them. The transformer encoder TimesFormer, which is pre-trained on the Kinetics-400 dataset, is chosen for this thesis. This encoder showed the best results on the Animal Kingdom (Ng et al., 2022) dataset. This video encoder uses "divided attention," which enables it to pay attention to spatial and temporal features separately in the video.

The spatio-temporal video encoder follows the same methodology as previous video transformer models (Mondal et al., 2023). For an RGB video with a width and height of *W x H* and *T* sampled frames, each sampled frame is partitioned into non-overlapping square patches depicted by *P x P*. This results in $N = \frac{HW}{P^2}$ patches. Subsequently, the patches are flattened into vectors, after which each patch is mapped into an embedding space with a projection layer as depicted in Equation 3.1, where $W_{emb} \in \mathbb{R}^{3P^2 \times D}$ represents the projection layer. $e^{pos}_{(p,t)}$ denotes a learnable positional embedding that captures the spatial and temporal positions of each patch. The sequence of embedded vectors $z^{(0)}_{(p,t)}$ for each $p \in P$ and $t \in T$ is the input for the Transformer encoder. Following transformer literature (Vaswani et al., 2017; Jiang et al., 2021; Dosovitskiy et al., 2020), a learnable vector is added to the first position of the sequence of embedded vectors, which represents the global token. The global token is an essential part of the video encoder since it

allows the video's overarching context to be captured in the global token while learning frame-level features in the remaining sequence of embedded vectors.

$$z_{(p,t)}^{(0)} = W_{emb} x_{(p,t)} + e_{(p,t)}^{pos} \tag{3.1}$$

For a video encoder with L_v layers, each layer l has a patch level representation as depicted by Equation 3.2, where $f_{\theta_v}^{(l)}$ represents the l-th layer of the transformer encoder. To obtain a global representation of all layers for a frame, all the frame level representations for each frame $t \in T$ are average pooled and projected to dimension D with linear layer $\mathbf{W}_{\text{out}} \in \mathbb{R}^{D \times D'}$. This process is depicted in Figure 3 by the *global encoder*. Each frame level representation for frame $t \in T$ is depicted by $\mathbf{v}_t$. The linear layer projection and average pooling are shown in Equation 3.3.

$$z_{(p,t)}^{(l)} = f_{\theta_v}^{(l)}(z_{(p,t)}^{(l-1)}), \quad l \in \{1, \ldots, L_v\} \tag{3.2}$$

$$\mathbf{v}_t = W_{\text{out}} \text{AvgPool}([z_{(0,t)}^{(L_v)}, \ldots, z_{(N,t)}^{(L_v)}]) \tag{3.3}$$

This outputs a frame-level representation sequence representing the video V. This sequence also consists of a global token ($[z_{(0,0)}^{(L_v)}$, which is the global token of the last output layer from the video encoder. The global token of the last layer provides high-level features of the whole video and relevant context. The sequence of frame level representations including the global token, is represented as $[z_{(0,0)}^{(L_v)}, v_1, \ldots, v_N]$. We will simplify this representation of the global encoder to $F = [v_0, v_1, \ldots, v_N]$.

Multi-modal query encoder

Given the same video $V \in \mathbb{R}^{T \times 3 \times H \times W}$ as fed to the video encoder, labeled with behavior labels Y, a multi-modal query is constructed in this component. Separate embeddings are learned for the videos and labels, which are eventually fused to construct a multi-modal query.

For the textual part of the multi-modal query, each behavior is represented by a learnable label embedding of dimension D. This collection of learnable label embeddings is denoted by $Q_l \in \mathbb{R}^{K \times D}$, where K corresponds to the number of behavior classes in the dataset. Mondal et al. (2023) initialize the embeddings in Q_l with the text embeddings of the corresponding classes. The D-dimensional text embeddings are obtained by using a pre-

trained text encoder, namely the 12-layer CLIP B/16 model (Radford et al., 2021). The CLIP B/16 model produces text embeddings with a dimension size of $D = 512$. In our approach however, the text embeddings are not merely initialized with the text embeddings of just the behavior labels, but prompts are generated to give the model more context to learn from. The difference between using merely labels and prompts will be further scrutinized in Section 4.3.

To generate the video embeddings, the same video frames are used for the video encoder. The T video frames are processed independently in batches of images with the CLIP B/16 image encoder (Gao et al., 2024). This results in frame-level embeddings of dimension size $D = 512$. Subsequently, the frame-level embeddings are average pooled and result are the final video embeddings denoted by $Q_v \in \mathbb{R}^D$. The video-level features extracted from the frames with the CLIP image encoder do not serve the same purpose as those extracted with the video encoder. The video-level features extracted with CLIP are aligned with the textual embeddings extracted with the CLIP text encoder, establishing complementary features that can better put features into context due to the semantic information of the learnable text embeddings.

$$Q_0 = W_{\text{que}}[Q_l, Q_v] \tag{3.4}$$

Finally, to construct a multi-modal query, we fuse the learnable text embeddings Q_l and CLIP video embeddings Q_v with the means of concatenation and apply a linear projection with the weights $W_{\text{que}} \in \mathbb{R}^{D\times(D+D)}$. This gives us the multi-modal query Q_0, denoted by Equation 3.4, where $[\cdot, \cdot]$ denotes the concatenation operation.

Multi-modal decoder

The final component of MSQNet is the multi-modal decoder, which combines both the output spatio-temporal features F of the video encoder and the multi-modal query features $Q_0 \in \mathbb{R}^{D\times(D+D)}$. On the multi-modal queries, both self-attention and cross-attention is performed to extract action-specific features from the video representation with the help of multi-layer Transformer decoders. A standard Transformer architecture is employed. This architecture includes the following layers: a multi-head self-attention (MultiHeadSA) module, a cross-attention (MultiHeadCA) module, and a feed-forward network (FFN).
For each of these decoder layers l, the output of the queries is updated according to the following equations, as the queries pass through these layers in chronological order. Equation 3.5 represents the multi-headed self-attention, Equation 3.6 represents the multi-headed cross-attention and Equation 3.7 represents the feed-forward network. The tilde represents vectors that are altered by adding positional encodings, and $Q_l^{(1)}$ and $Q_2^{(1)}$ are

the intermediate variables before being passed into the feed-forward network.

$$Q_l^{(1)} = \text{MultiHeadSA}(\tilde{Q}_{l-1}, \tilde{Q}_{l-1}, Q_{l-1}) \tag{3.5}$$

$$Q_l^{(2)} = \text{MultiHeadCA}(Q_l^{(1)}, \tilde{F}, F) \tag{3.6}$$

$$Q_l = \text{FFN}(Q_l^{(2)}) \tag{3.7}$$

For a clearer overview and simplicity, the parameters of the MultiHead attention and FFN are excluded. The self-attention mechanism is useful since it allows for each embedding in a sequence to pay attention to other parts of the sequence. This aids in capturing dependencies between different parts of the sequence.
The cross-attention mechanism is helpful because the queries can attend to a different set of keys and values coming from a completely other sequence. In our case, the output queries from the MultiHead self-attention mechanism can attend to the spatio-temporal features F from the video encoder, as depicted in Equation 3.6. This allows the mechanism to inject relevant contextual information from the video encoder features into the queries. This is fruitful for aligning query information with the video features and contributes to a more context-aware representational model.
Both self-attention and cross-attention are multiheaded, meaning that multiple exact mechanisms are run in parallel. This allows each self and cross-attention head to learn to focus on different parts of the sequence, enhancing the model's contextual representation and contributing to more accurate predictions. This allows the embeddings to be learned end-to-end, implicitly capturing label correlations from the data.

Subsequently, to predict single-class labels, the model needs to have confidence in identifying the correct behavior class label. For multi-class problems, each behavior class is treated as a binary classification problem. However, since our dataset is small and thus less likely to capture complex patterns in the data, we refrain from classifying multi-class labels and focus on single-class label classification only.

$$p_k = \sigma(W_k Q_{L,k} + b_k) \tag{3.8}$$

To achieve this, for each class k, its feature representation of the last layer L of the Transformer Decoder $Q_{L,k} \in \mathbb{R}^D$ is projected onto a linear projection layer. Subsequently, an activation σ is applied in the form of a Softmax function. This projection and activation is denoted by Equation 3.8, where $W_k \in \mathbb{R}^D$ is part of $W = [W_1, \ldots, W_K]^T \in \mathbb{R}^{K \times D}$. $b_k \in \mathbb{R}$, and $b = [b_1, \ldots, b_K]^T \in \mathbb{R}^K$ are the parameters of the linear layer. The probabilities of each behavior class are represented by $p = [p_1, \ldots, p_K]^T \in \mathbb{R}^K$. Function p can be seen as a function that maps videos to class probabilities.

3.2.2 Learning objective

For each given video V, the goal is to train the model in such that the predicted probabilities of each class align with the ground truth labels. To achieve this, categorical cross-entropy loss is used as a loss function for training the model, as depicted by Equation 3.9, where n represents the number of samples and K the pre-defined number of behavior classes.

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{K} y_{ij} \log(p_{ij}), \tag{3.9}$$

Since we only train the model with single-class labels, the "classical" version of cross-entropy loss is used in contrast to multi-class classification problems. In multi-class classification, binary cross-entropy loss is used since, for multi-label classification, each class is treated as a separate binary classification problem.

3.3 Pose Estimation

This section will discuss the pose estimation algorithm as proposed by Zhu et al. (2022). Since we possess only a small dataset and limited labor sources to manually annotate the keypoints on all dog images, we opt for a pre-trained pose estimation model that is capable of accurately detecting keypoints on a dog's body despite the great variety in morphology, colors, sizes, breeds, behaviors, and camera angles in our dataset. In short, the pose estimation model should be robust and be able to generalize despite the mentioned varieties. Therefore, the popular human pose estimation model HR-Net (Sun et al., 2019) was chosen, pre-trained on the Animal Pose dataset by Cao et al. (2019). The pose estimation backbone HR-Net will be discussed in Section 3.3.1. Subsequently, the approach to deal with missing keypoints will be explained in Section 3.3.2 and finally how we normalize the pose will be discussed in Section 3.3.3.

3.3.1 HR-Net

Deep High-Resolution Network (HR-Net) (Sun et al., 2019) is a popular 2D pose estimation network that was originally designed for human pose estimation but works great on animal

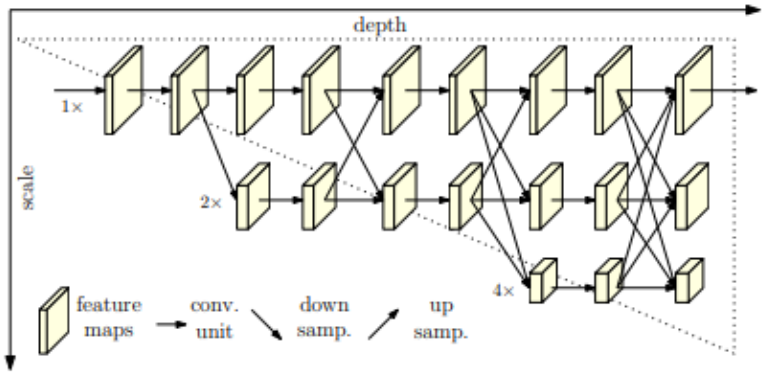


Figure 15. The HR-Net architecture which illustrates the parallel streams consisting of high-to-low sub-networks which constantly exchange information between each other through the means of multi-scale fusion. Figure is adopted from Sun et al. (2019)

pose estimation as well.

What distinguishes HR-Net from other traditional pose estimation networks is that HR-net retains high-resolution representations of the image throughout the pose estimation process, preserving valuable spatial features that might be of the essence for pose estimation. On the other hand, traditional methods often downsample the image's low-resolution feature maps to capture their high-level semantics, and subsequently upsample the feature maps to their original size, posing the risk of losing valuable spatial details.

The architecture of HR-Net is shown in Figure 15. HRNet consists of a central stem network. This stem network usually consists of a few convolutional layers to process an input image and generate its initial feature maps of the image. To avoid downsampling the image entirely, HRNet employs parallel convolution streams, of which each stream keeps the feature maps at a different resolution. The stem network stream keeps the feature map at its original resolution, while other parallel streams might keep it at 1/2 or 1/4 of the original resolution, for instance. The streams are added as the model progresses. These parallel streams aid in extracting both fine-detailed and high-level features since the high-resolution streams preserve the detailed information of the image. In contrast, the lower-resolution streams obtain high-level contextual information of the image. Having both finely detailed and high-level features improves the overall accuracy of the model.

To combine the finely detailed and high-level information of both streams, multi-scale fusion is applied with the use of exchange units. Exchange units facilitate the exchange of information between different streams. At every stage in the pipeline between feature maps, the exchange units upsample features from lower streams to match the resolution of their own stream, downsample from higher resolution streams, and ultimately combine

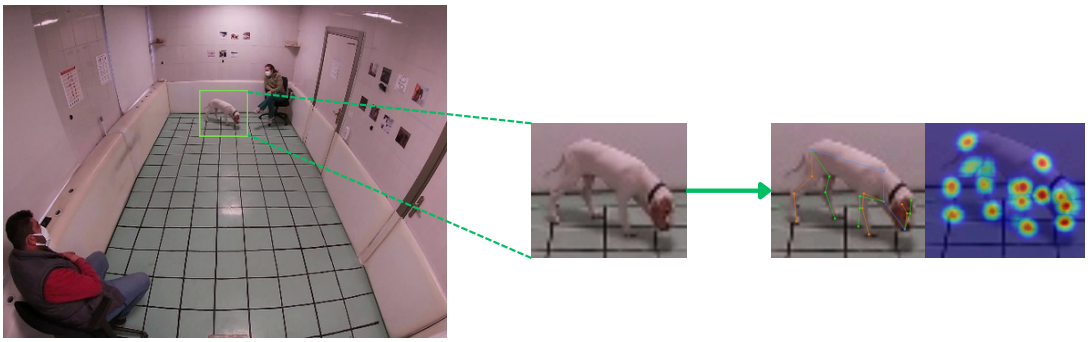


Figure 16. An example of the bounding box detection with YOLO and pose estimation with HR-Net. (a) the bounding box coordinates are detected with YOLOv5/YOLOv8. (b) The image is cropped according to the bounding box. The bounding box is enlarged with 10 percent to ensure the entire dog is in the frame. (c) HR-Net is applied to extract the keypoint coordinates of the pose. On the left the dog with the coordinates is drawn, and on the right the heatmap is drawn that detects the keypoint locations.

these features through concatenation with the original features of its target stream. After concatenating the features, a convolutional operation is applied to recalibrate all the newly obtained features from the higher and lower dimensions.

This process continues until the streams reach the final layers of the network, where all the lower-resolution features are upsampled to their original resolution. In the final layer, the keypoint heatmaps are generated, indicating the probability of a joint being located in a particular location in the image. The locations with the highest heat values are consequently chosen as the predicted keypoints. An example of this process is shown in Figure 16.

HRNet is a top-down pose estimation algorithm that requires the algorithm to first know the overall location of the dog for which it needs to estimate key points. Our method is facilitated by the YOLOv5 and YOLOv8 object detection model, where we first estimate the bounding box and crop the image according to the bounding box. These cropped images are fed to HRNet for which HRNet outputs keypoints for all the detected joints in the image. Since HRNet is a top-down pose estimator, it focuses on individuals to detect poses, which generally leads to higher accurate pose estimation in contrast to a bottom-up approach where keypoints for all individuals in the image are predicted simultaneously. Since our model only contains one dog in each frame, the model will not suffer from the higher computation costs from top-down approaches since it only needs to predict the keypoints of one dog at a time.

3.3.2 Missing keypoints

The pose extracted from HRNet is not always perfect due to a myriad of factors. Sometimes, parts of the dog's body are occluded by the dog itself, which makes it difficult for HRNet to extract the whole pose of the dog. For example, this can occur when one of the dog's front legs is precisely in front of one of its back legs, or when the dog is close to the camera, its back is occluding its legs. The dog may also be occluded by people or other objects in the room which are not filtered out by object detection, or the image might be of too-low resolution or might be too blurry. All these factors might affect the pose estimation model's ability to detect a dog's body parts heavily in a negative manner. This section will mainly describe how to tackle the self-occlusion problem since most of our severely occluded frames are already filtered in the object detection process.

To generate missing keypoints, we can profit from a dog's bilateral symmetry, which means that the left and right sides of a dog's body are equal. In this manner, keypoints from missing body parts from the left side can be inferred from existing body parts of the right side and vice versa. Additionally, keypoints can be inferred from consecutive frames, where the missing keypoints might be observed over time. The central concepts for inferring missing keypoints as proposed by Zhu et al. (2022) are as follows:

1. Keypoints can be interpolated from neighboring frames if the keypoint in question does occur in a neighboring frame.
2. If after interpolating from neighboring frames there are more than nine keypoints or spine keypoints missing, the pose is not taken into account and discarded.
3. If only one out of three keypoints is missing from a leg, this missing keypoint can be determined by the other two keypoints in the leg and the keypoints in the spine.
4. If more keypoints are missing in a leg, this missing keypoint can be inferred by benefiting from bilateral body symmetry of the other legs, or the front and backside symmetry of the legs.

The algorithm first determines if the head and spine keypoints are present, and if not, it tries to infer those first. If this is unsuccessful, it discards the pose; otherwise, it will try to construct the dog's legs. The legs of dogs are often the most expressive part of a pose, convey the most information, and are often the most easily self-occluded. This is why this missing keypoint treatment focuses mostly on inferring the keypoints for the dog's legs.

After treating the missing data, the pose estimator produces a mapping of keypoints from a sequence of RGB frames. This can be denoted as follows:

$$s_i(t) \xrightarrow{\text{pose detector}} p(t) \quad \forall t = 1, \ldots, T_i \tag{3.10}$$

where $s(t)$ represents the raw and cropped RGB frame of a sequence, and $p(t)$ represents the dog's pose at time/frame t from a total sequence length of T. The pose $p(t)$ is composed of a set of keypoint coordinates which can be denoted as follows:

$$p_i(t) = \{(x_j(t), y_j(t))_i\}_{j \in J}\,, \tag{3.11}$$

where $x_j(t)$ and $y_j(t)$ represent the x, y-coordinates for each keypoint j for a total of keypoints J, as defined by the pose estimation model. Here, the t represents again the frame in the time sequence.

3.3.3 Pose normalization

Since there are dogs of all sizes and breeds in our dataset, it is crucial to apply pose normalization such that the model can generalize better and become more robust. This prevents situations where there are only small dogs in pain, and the model starts associating small-sized poses with pain, instead of learning pain symptoms in pose representations. Pose normalization ensures that poses are all around the same size in the end. Additionally, the absolute position of the dog in the image will not affect the processing of the dataset if the coordinates are normalized.

Pose normalization, as proposed by Zhu et al. (2022), transforms the pose from an absolute position to a local root point-centered position. This thesis's root point is the middle between the neck and tail end keypoints, since this is a nice center point of the dog. The equations for the root point-centered keypoints are shown in Equation 3.12, and the resulting set of root-point centered keypoints is shown in Equation 3.13. For simplicity, the dependence of t is discarded in the equations.

$$(\bar{x}_j, \bar{y}_j)_i = (x_j, y_j)_i - (x_{\text{root}}, y_{\text{root}})_i \quad \forall j \in J \tag{3.12}$$

$$\bar{p}_i = \{(\bar{x}_j, \bar{y}_j)_i\}_{j \in J} \tag{3.13}$$

Subsequently, normalization is applied to ensure that all poses are the same size for each

dog and between different dogs. For this, the size of the vector $\overline{\vec{v}}_{\text{neck, tailend}}$ between the neck keypoint and tail-end key point is used to normalize the pose as represented by Equation 3.13. This normalization process is shown in Equation 3.14. One issue that might be encountered due to a normalization process like this is that if the $\overline{\vec{v}}_{\text{neck, tailend}}$ is short due to a weird camera angle, the pose might become unexpectedly big. We did not have enough time to investigate this, but this might be useful for future work.

$$\bar{\bar{p}}_i = \frac{\bar{p}_i}{\|\vec{v}_{\text{neck, tailend}}\|} \tag{3.14}$$

3.4 Pain Estimation

The final pain estimation model is a combination of the baseline constructed by Zhu et al. (2022), and the ethogram obtained from the behavior estimation method from MSQNet, resulting in the three stream model shown in Figure 14.

The baseline from Zhu et al. (2022) consists of two streams. Namely, an LSTM stream used to process the keypoints, and a ConvLSTM stream used to process the raw RGB frames. LSTM models are well-suited for processing sequential data, which our data inherently is since it comprises consecutive frames. LSTM models are great for such tasks since they excel at capturing temporal dependencies due to their recurrent nature, which aids in detecting actions and movements in the frames. Since a basic LSTM is not able to capture spatial features, but only temporal features, a ConvLSTM is used to process the RGB frames. This keeps the inherent recurrent structure of the LSTM model to capture temporal dependencies and adds convolutional layers to extract spatial features from the frames.

First, this section will discuss pose processing with an LSTM model and frame processing with a ConvLSTM model. After this, we will discuss how we handle the behaviors obtained from the MSQNet model and add these to the two-stream model. We will discuss two ways in which the behaviors will be processed. Namely, one method will be conducted using a random forest classifier to obtain ethogram embeddings, and another method will be using a linear layer to obtain ethogram embeddings. Lastly, the explainability methods are discussed to give us more insight into why the model predicts pain or not.

3.4.1 Pose processing with LSTM

The architecture of LSTM models is in fact a more complicated version of classical Recurrent Neural Networks (RNNs). Similar to LSTM models, RNN models are well-suited for processing sequential data as well due to their ability to retain important information from

previous states by storing this in its hidden states. The major drawback of using classical RNNs is that they tend to suffer from vanishing and exploding gradients when learning long-term dependencies. This issue arises during the backpropagation phase, where the gradients are used to update the RNN's weights and can either become excessively large or small due to this. When a gradient becomes too small, it is difficult for the model to update the weights at the early layers of the model. This prevents the model from effectively learning long-term dependencies, while it might perform well at capturing short-term dependencies The opposite happens during an exploding gradient, where the gradients become excessively large during the backpropagation process. An exploding gradient often makes the model unreliable because the massive gradient frequently causes numerical instability. This sometimes causes the loss function to become NaN.

LSTM models tackle this problem by incorporating memory cells and gating mechanisms in their architecture. The memory cell of an LSTM, denoted as c_t in Equation 3.15, serves the purpose of maintaining information from different timesteps, without having the trouble of gradient issues. LSTM models consist of three main parts: an input gate i_t, a forget gate f_t and an output gate o_t, which can all be seen in Equation 3.15 and all use a sigmoid activation function. The input gate determines how much information from the new input x_t will be used to update the new cell state of the memory cell c_t. This allows it to filter any irrelevant information from the new input. The forget gate manages to what extent the previous cell state c_{t-1} should be kept and, thus, what portion should be forgotten. This again allows the LSTM model to filter irrelevant information from the previous state. Subsequently, a candidate cell state $\tilde{c}_t$ is generated with a tanh activation function.

To construct the new cell state c_t, the previous cell state c_{t-1} is transformed by the forget gate with element-wise multiplication. This determines what portion of the information from the previous state is retained. Similarly, candidate cell state $\tilde{c}_t$ is scaled by the input gate with element-wise multiplication, which deems what information is worthy of being passed to the next state. Element-wise multiplication is denoted by $\odot$ in Equation 3.15. These two parts are combined to form the new cell state c_t. The new hidden state h_t is formed by element-wise multiplication between the output gate o_t and the new cell state c_t, the output gate controls how much information from the new cell state is passed on to the next layers. The weight matrices for each gate for the hidden states are denoted by W^{gate}, and the projection matrices for the new input by I^{gate}, where $gate$ represents the respective gate of the matrix.

$$
\begin{aligned}
i_t &= \sigma(W^u h_{t-1} + I^u x_t + b_i) \\
f_t &= \sigma(W^f h_{t-1} + I^f x_t + b_f) \\
o_t &= \sigma(W^o h_{t-1} + I^o x_t + b_o) \\
\tilde{c}_t &= \tanh(W^c h_{t-1} + I^c x_t + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= \tanh(o_t \odot c_t)
\end{aligned}
\tag{3.15}
$$

LSTM model's intricate memory cells and gating mechanisms allow the model to selectively maintain and add new relevant information to the memory cell. This is particularly interesting for action recognition, since the model will be able to track the dog's temporal dynamics of the keypoints throughout the video and thus capture and understand the movements and its context.

Attention mechanism

Although LSTM models are better at capturing long-term dependencies than traditional RNNs, they can still have trouble capturing the dynamics in very long sequences. The famous attention mechanism can help here and will be applied in our LSTM model. An attention mechanism gives a model the ability to focus on different parts of the *entire* input sequence by assigning weights to these parts. This effectively enables the model to focus on more relevant sequence parts and enhances the LSTM's capability of processing longer sequences (Wang et al., 2016). The attention scores of the attention mechanism are computed as follows:

$$
e_{t,j} = v^T \tanh(W_e h_{t-1} + U_e h_j) \tag{3.16}
$$

where W_e and U_e represent learned weight matrices and h_{t-1} the hidden state of the previous step. h_j is the hidden state of all the input steps combined, where $j \in J$ and J represent the total number of input steps (or positions) in the entire sequence. The tanh function introduces non-linearity. v is a transposed learned parameter vector to project the result into a scalar score. These scores are subsequently used to compute attention weights as follows:

$$
\alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_{k=1}^{T} \exp(e_{t,k})} \tag{3.17}
$$

where $exp(e_{t,j})$ is an exponential function that ensures all the attention scores from Equation 3.16 are a positive value. The summation sums all the exponentiated scores. This step normalizes the attention scores obtained in Equation 3.16, which indicates the relative importance of each attention score. All attention weights $a_{t,j}$ sum to 1.

$$c_t = \sum_{j=1}^{T} \alpha_{t,j} h_j \tag{3.18}$$

The attention weights are consequently used in the construction of a context vector c_t, shown in Equation 3.18 The context vector represents a weighted sum of the hidden states from the input sequence that captures the most relevant information at timestep t. If a certain $\alpha_{t,j}$ is a large weight, it pays much attention to part j of the input sequence.

$$\tilde{h}_t = \tanh(W_c[c_t; h_{t-1}]) \tag{3.19}$$

Subsequently, the context vector c_t is combined with the hidden state from the previous timestep h_{t-1} through concatenation, and is transformed by a weight matrix W_c and a tanh activation function. This process is shown in Equation 3.19. The updated LSTM architecture where the with context vector enhanced hidden states $\tilde{h}_t$ are used is shown in Equation 3.20.

$$\begin{aligned}
i_t &= \sigma(W^u \tilde{h}_t + I^u x_t + b_i) \\
f_t &= \sigma(W^f \tilde{h}_t + I^f x_t + b_f) \\
o_t &= \sigma(W^o \tilde{h}_t + I^o x_t + b_o) \\
\tilde{c}_t &= \tanh(W^c \tilde{h}_t + I^c x_t + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned} \tag{3.20}$$

3.4.2 Frame processing with ConvLSTM

Where traditional LSTM models use fully connected layers between state transitions and are adept at modeling temporal dynamics in sequential data, they are not useful for capturing spatial information. To overcome traditional LSTM models' limitations in capturing spatial information, a new ConvLSTM model was first introduced by Shi et al. (2015). The authors propose to replace the fully-connected layers in traditional LSTMs with convolutional operators. This enables the ConvLSTM model to capture both valuable

spatial and temporal information between RGB frames. This makes ConvLSTM models particularly well-suited for processing spatio-temporal information such as frames of video data, which naturally consists of temporal data in the form of consecutive frames and spatial data in the form of pixels. To retain spatial information while processing the RGB frames, the inputs $(x_1 \dots x_t)$, cell states $(c1 \dots c_t)$, and hidden states $(h_1 \dots h_t)$ are all represented as 3D-tensors (channels, height, width) instead of 1D-tensors like in traditional LSTM models. This ensures that the spatial dimensions of the frames are preserved in terms of channels, rows, and columns. The architecture of a ConvLSTM model is as follows:

$$
\begin{aligned}
i_t &= \sigma(W_i * x_t + U_i * h_{t-1} + V_i \odot c_{t-1} + b_i) \\
f_t &= \sigma(W_f * x_t + U_f * h_{t-1} + V_f \odot c_{t-1} + b_f) \\
o_t &= \sigma(W_o * x_t + U_o * h_{t-1} + V_o \odot c_t + b_o) \\
\tilde{c}_t &= \tanh(W_c * x_t + U_c * h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
\tag{3.21}
$$

where $*$ denotes the convolutional operation and $\odot$ the element-wise product. Like a traditional LSTM, the W, U, V matrices must be learned and are respective to their own gate. Each gate's bias vector b shall also be learned throughout the training progress. The V matrix serves the purpose of allowing the model to not only be dependent on the previous hidden state h_{t-1}, but also on the previous cell state c_{t-1}. This allows the model to become even better at capturing contextual information and long-term dependencies.

Similarly to the attention mechanism we use for traditional LSTMs, an attention mechanism is also used for the ConvLSTM to improve the model's capability of capturing long-term dependencies.

3.4.3 Ethogram processing

Our new addition to the two-stream model by Zhu et al. (2022) is an ethogram obtained from the MSQNet model by Mondal et al. (2023) to enhance the model's understanding of what kind of movements the dog is inhabiting. We experimented with two ways to incorporate the ethogram dimensions into the current two-stream model: a linear layer to process the ethogram and a random forest classifier. We decided to refrain from using complex models and used simple models to process the ethogram stream instead. Since our dataset is small, using complex models with many parameters would probably cause the model to overfit. Simpler models with fewer parameters generalize better on smaller

datasets in general.

This section is constructed as follows: first we will discuss how the ethogram obtained from the MSQNet model is pre-processed to fit our needs in the pain estimation model. Subsequently, both methods used to process the ethogram dimensions are explained.

Pre-processing

The ethogram as obtained from MSQNet shows for every dog the predicted behavior for separate segments indicated with start and end frames. Additionally, the ground truth and the output array on which the prediction is based, which includes prediction values for each ethogram, are included.

For our pain estimation model, we want to receive a one-hot encoding showing which behaviors are present in every video frame. With this in mind, a new CSV file was generated, where for each frame number, the corresponding one-hot encoding of the behaviors present in the video was given. This CSV file is structured as follows:

$$\begin{aligned} nr, ethogram \\ 1, [1,0,0,0,0,0,0,0,0,0] \\ 2, [1,0,0,0,0,0,0,0,0,0] \\ ... \\ n-1, [0,0,0,0,0,1,0,0,0,0] \\ n, [0,0,0,1,0,0,0,0,0,0] \end{aligned} \tag{3.22}$$

where *nr* represents a frame number, and the ethogram shows the one-hot encoding of which behavior is present. For instance, the first index of the ethogram represents standing still. Subsequently, the ethogram representation in this file is used for further processing in the linear layer and decision tree classifier.

Linear layer

For the first approach, a linear layer is used to process the one-hot encoding of the ethogram. Linear layers are a simple approach suitable for smaller datasets to avoid overfitting. The linear layer can project a high-dimensional one-hot encoding into a dimensionality, which is more suited for further processing of the embedding. Transforming the input of the linear layer into a new features space with the linear layer, based on a learned weighted combination of the input features, might establish new feature spaces where input elements might be more linearly separable.

The linear layer is first initialized with weights. A proper weight initialization ensures that the model maintains a proper balance in weights, which may prevent the occurrence of exploding or vanishing gradients. If the initialized weights of the model are too small, the gradients may also become too small, which will cause a vanishing gradient. This would cause negligible updates to the weights and result in slow convergence of the model. On the other hand, if the initialized weights are too large, this would result in exploding gradients and cause numerical instability in the model. This makes it essential to conduct proper weight initialization of the model. Additionally, with a proper weight initialization, the model may converge faster. If the initialized weights are not symmetric, it also helps with learning a greater variety of features. Specifically, the neuron layers where the weights have the same number will be more likely to learn the same features. Initializing a diverse set of weights will encourage the model to learn a diverse representation of features.

The weight initialization method used for this model is proposed by Glorot and Bengio (2010), commonly known as the *Xavier* or the *Glorot* initialization and is shown to improve the performance of deep neural networks. From here on, we will refer to the method as *Xavier initialization.Xavier initialization* is a weight initialization method that scales the weights based on the input and output units of the linear layer. For this thesis, we will be initializing weights with the uniform distribution as shown in Equation 3.23, where n_{in} and n_{out} represent the number of input and output units, respectively. Due to the uniform distribution, the variance of the gradients will remain within a specific range, promoting a stable learning process without vanishing or exploding gradients. A uniform distribution was chosen because it promotes faster convergence and more stable training (Glorot and Bengio, 2010).

$$W \sim \text{Uniform}\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}\right) \tag{3.23}$$

Training on a small dataset is tricky since the model is more prone to both overfitting and underfitting on the data. To mitigate the risks of this happening, additional layers were added to the processing of the ethogram. Namely, there is a batch normalization layer and a dropout layer. Batch normalization regularizes the model and helps to prevent overfitting. A dropout layer serves the same purpose, but it drops neurons during training to avoid overfitting. However, a large dropout value can cause models trained on a small dataset to underfit quickly. Due to this, we opted to choose a small dropout value: 0.1. This value ensures that only 10% of the neurons are dropped.

Random forest classifier

The other method in which we will process the ethogram will be through a random forest classifier. We chose a random forest classifier rather than a singular decision tree since our dataset is small and thus prone to overfitting. A random forest classifier averages multiple decision tree predictions, making the model more robust and better at generalizing to unseen test data (Breiman, 2001).

Each decision tree, comprising a random forest classifier, is trained on a subset of the data. This subset is obtained using bootstrapping, a random sampling method with replacement. A split is based on a random subset of features, and the tree is split on the best feature that produces the highest Gini importance. The Gini importance for binary classification is calculated by first computing the Gini impurity as follows:

$$I_G(p) = 2p(1-p) \tag{3.24}$$

where p is the probability that an instance is classified as class p. Subsequently, the model considers all splits and selects the split that provides the largest Gini gain. The Gini gain is calculated as follows:

$$\Delta I_G = I_G(parent) - \sum_j \frac{N_j}{N} I_G(child_j) \tag{3.25}$$

where the number of samples of the parent node is denoted as N and that of the j-th child node as N_j. Subsequently, to discover the Gini importance of a feature, the Gini gain of that features is summed over all possible splits. The feature with the highest Gini gain is the most descriptive feature.

3.4.4 Three stream model

The final pain estimation model, as proposed by this thesis, comprises the three parts described in this section. Namely the pose processing with LSTM, the frame processing with ConvLSTM, and the ethogram processing with either a linear layer or a random forest classifier. The three parts all work in separate streams independently from each other and are ultimately fused.

The ConvLSTM stream consists of four stacked ConvLSTM cells, each with 32 hidden units. Each ConvLSTM cell is followed by a max pooling layer with a stride of 2 and a batch normalization layer. The max pooling layers being used four times after each cell

effectively downsample the image to 16 times its original size. Max pooling is helpful since it reduces the complexity and computational costs of the model. This ensures that there are fewer parameters in the model, which reduces the risk of overfitting. This primarily benefits small datasets since it encourages the model to generalize better. Additionally, max pooling reduces noise and extracts more prominent features while ignoring less significant details. After the four ConvLSTM cells, an attention layer is employed, which serves the purpose explained in Section 3.4.1.

Similarly to the ConvLSTM stream, the LSTM stream consists of four stacked LSTM cells. However, these LSTM cells are not followed by max pooling and batch normalization layers. This is less common in LSTM networks that process one-dimensional data such as keypoints and might even disrupt the temporal dynamics of the keypoints. Similarly to the ConvLSTM stream, the LSTM stream is also followed by an attention layer. In the experimental evaluation, we will experiment with different numbers of hidden units, namely 32, 64, and 128 hidden units.

The ethogram stream is entirely different from the other two streams. Since our dataset is small and the model is likely to overfit on the data, we will limit ourselves to using simple structures for the ethogram. As explained earlier, a linear layer followed by batch normalization and dropout layers will be used to process the ethogram dimensions as one method. Additionally, a random forest/decision tree classifier will be used as another method to process the ethogram.

The three separate streams, as shown in Figure 17 are ultimately fused by a fusion method, namely concatenation. Concatenation is a fusion method that stacks the feature vectors obtained from separate streams next to each other, creating a single longer feature vector. This preserves the original information of the individual streams since no original information is altered by concatenation. This results in a longer feature vector that encapsulates the original information from the RGB frames, keypoints, and ethogram dimensions.

Subsequently, the fused feature vector is put into a fully-connected layer. This fully-connected layer computes a weighted sum of the features. After this, a sigmoid activation function is used to compute the probabilities of the pain and no pain classes. For the three-stream pain estimation model the same loss function is used as mentioned in Section 3.2.2 for MSQNet. The only difference is that in the case of our pain estimation model, we have just two classes, which will make it a *binary cross-entropy loss function*. This means that the K will be set to two.

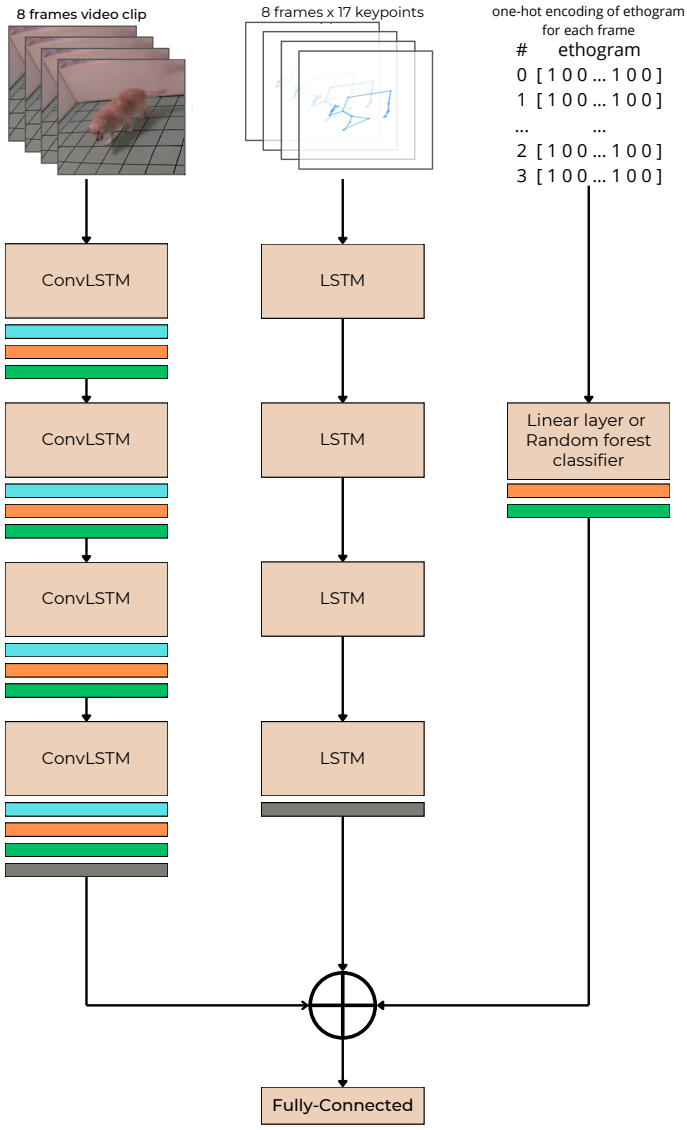


Figure 17. The architecture of the three separate streams. The blue and orange layers represent max pooling and batch normalization, respectively. The green layer represents a drop-out layer, and the gray layer is an attention layer.

3.5 Explainability

One common issue with neural networks is that they lack interpretability due to their black-box nature. With the introduction of the ethogram dimensions into the model, we hope to create a more understandable model that can give us insights into what behavior a dog displays when they are in pain. Additionally, confidence scores will be generated to measure how confident the model is in predicting a certain class. This section will focus on explainability methods for both behavior estimation and pain estimation separately.

3.5.1 Pain estimation explainability

For the pain estimation model, several methods to enhance the explainability of the model will be conducted, which will be clarified in this section.

Naive Bayes Classifier

One simple method to give insight into how the model makes its predictions is to use a post-hoc analysis, which extracts the model's predictions and uses them to train a Naive Bayes classifier. For each class, pain or no pain, the corresponding ethograms will be extracted. We can determine the Naive Bayes probabilities with the ethograms linked to each pain class. Leveraging Naive Bayes probabilities will give us a more transparent and interpretable view of how likely the model is to predict a specific class based on an ethogram entry. A Naive Bayes classifier is a model-focused approach since it looks at the model's functionality as a whole.

Shapley

A data-focused approach is using Shapley values to enhance model explanation. Shapley assesses each feature's contribution to the model's final output prediction and ensures that the impact of each feature is fairly attributed. A more detailed explanation of Shapley values can be found in the Related Work at Section 2.7.1.

To calculate the Shapley values and visualize them we will be using the *shap* library by Lundberg and Lee (2017). The shap library consists of several possibilities to visualize shap values. To look at the contributions of each feature for a single prediction we will be using a so-called *waterfall* plot. This plot shows how much each feature influences the outcome of the prediction. Features with the highest contributions are placed on top, and the lowest on the bottom. This creates a waterfall visualization. Features that push the prediction to the higher end are shown in pink, and features that pull the prediction to the lower end are shown in blue. In our case, the pink features would push the prediction towards pain, and the blue features would pull the prediction towards no pain. Another plot we will be using from Lundberg and Lee (2017) is the so-called beeswarm plot. In the beeswarm plot, the Shapley value of every feature from every sample is plotted in this plot. Like the waterfall, the pink colors indicate features pushing the model's prediction higher, and blue pulls the model to the lower end.

Decision tree

Decision trees are a cornerstone of XAI and serve great interpretability. Decision trees operate by recursively splitting data based on feature importances, allowing us to trace each prediction's reasoning process. Since we want to know why our dog is in pain, this approach is highly valuable. An example of a decision tree used in our domain is shown in Figure 13 in the Related Work Section 2.7.2. Here, a dog's behavior is scrutinized to predict its emotions, which provides us with a clear overview of what might be indicators of particular emotions in dogs. We extract a decision tree from our random forest classifier and use that to explain the dog's behavior leading to pain or no pain.

Additionally, to determine the feature contributions of the decision tree for a single prediction, we will use a similar approach to the contribution plots of Kaya and Salah (2018). This bar plot shows the contribution each feature has to the final prediction. By using this approach, we will gain more insight into which features contribute the most to pain predictions and which features do not. To realize these contribution plots, we will be using the *treeinterpreter* library by Saabas (2016).

GRAD-Cam

Gradient-weighted Class Activation Mapping (Grad-CAM) is a great visualization tool commonly used to visualize where a model with CNNs attends on its input. Grad-CAM generates heatmaps where the red colors indicate regions to which the model pays great attention and the blue hues to which the model pays less attention to. In our model we apply the Grad-CAM method to the latest ConvLSTM layer since this layer contains the most valuable information.

Grad-CAM realizes this by first computing the gradient $\frac{\partial y^c}{\partial A^k}$ of a certain target class y^c concerning the feature maps A^k of the last convolutional layer. The gradient is indicative of how important each of the feature maps is. Subsequently, the importance weights of each feature map are obtained by globally averaging the gradients of all spatial locations (i, j). The importance weights show how important a feature map is to y_c. This process is shown in Equation 3.26 where Z represents the number of pixels

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{3.26}$$

After this, a weighted sum of the feature maps with their respective importance weights is put into a ReLU activation function to create a saliency map highlighting the image's important regions as shown in Equation 3.27. Lastly, the saliency map is resized to the original image size and displayed over the image.

$$L^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right) \tag{3.27}$$

3.5.2 Behavior estimation explainability

To perform a qualitative analysis on the MSQNet model, we will follow an approach that is similar to that used by the authors Mondal et al. (2023). To show where MSQNet pays its attention to on the raw frames, we will draw attention to heatmaps, where the attention maps of the transformer's heads will be displayed on the raw frames of the video. The

attention maps of multiple transformer heads will be averaged to obtain a singular attention map. This approach is similar to the Grad-CAM approach discussed earlier but applied to attention scores. In the end, it produces the same kind of visualization.

Additionally, we want to perceive how confident the model is in predicting its labels. Therefore, we will display the confidence scores of the top 5 classes that the model predicts for each video segment. The confidence scores will be obtained by applying a softmax (Goodfellow et al., 2016) function to the model's logits to transform them into a probability distribution. This will provide us with the certainty that the model makes a prediction.

4. Experimental Evaluation

In this chapter the pain detection model will be assessed through an experimental evaluation. The behavior estimation is discussed first, and subsequently the pain estimation model is discussed into which the behavior estimations are incorporated through the means of an ethogram. To assess how the ethogram affects the results an ablation study is performed as well. Finally, both qualitative and quantitative analyses are performed to get an insight into how the model makes its predictions.

4.1 Dataset Description

The dataset used for the experiments to estimate pain in dogs is the same dataset as used by Zhu et al. (2022). The data is collected by the Veterinary Medicine Department of Ankara University, by a research group led by Prof. Dr. Yasemin Salgırlı Demirbaş, which performs research in dog behavior (Salgırlı, 2012, 2020). The dataset consists of 63 videos of dogs in pain and dogs without pain. The videos of the dogs are obtained from four different kinds of experimental environments, which will be discussed in Section 4.2. For some dogs in pain, multiple videos are taken from different camera angles, namely a front and side view. For these dogs, the side view video will not be used since the majority of the dataset is filmed from the front. The resulting dataset consists of 40 videos of unique dogs that are not in pain and 16 videos of dogs that are in pain. In addition, we have a new dataset from Ankara University of dogs in pain filmed in an in-the-wild setting. This dataset contains 17 videos of dogs in pain. In total, all the videos have a duration of 365.87 minutes. The pain labels of the dogs are given by a veterinary expert based on the medical records of the dogs, which indicate that the dog might show pain symptoms visible to the naked eye.

	# videos	Duration (minutes)
Pain	16	217.40
Not pain	40	115.85
Pain - in the wild	17	32.62
Total	73	365.87

For the videos, the frames are extracted with *OpenCV* by Bradski (2000) at a sample rate of three frames per second. Subsequently, for training the model, clips of eight frames are used where two frames overlap between clips. This ensures that the model has more data to train and is less likely to miss crucial temporal information between clips, which is

especially important when behaviors are incorporated into the model.

4.1.1 Experimental environments

The dataset used for the experimental evaluation consists of three different experimental settings. In essence, the dataset can be separated into three different parts. The first two parts are from the original dataset used by Zhu et al. (2022) recorded in a veterinary room, where one part is of dogs in pain and one part of dogs without pain. The last part of the dataset is our new in-the-wild dataset.

The first two parts of the dataset are recorded in the same veterinary room at the Veterinary Medicine Department of Ankara University. The veterinary room is a 10-square-meter rectangle chamber with a camera hanging from the ceiling at a height of about two meters. The camera is not calibrated, which can cause some distortion in the recorded videos. All the videos are recorded at a resolution of 2K.

Part of the dataset recorded in this veterinary room was originally obtained to study the body language of dogs and their behavioral responses. For this research a so-called OFT (open field test) was realized. About 20 percent of the dogs in the OFT part of the dataset suffer from certain physical problems that may show pain symptoms in their behavior, which is why this dataset is used for our research. For the OFT test, a dog is put alone in the veterinary room without a human for about five minutes, and its behavior is recorded. Additionally, a toy car is in the room, which will be controlled from outside by a veterinary expert. In this way, the dog's behavior in the presence of an unknown moving object is recorded.

The other part of the dataset that is recorded in the same veterinary room was specially recorded for the research of Zhu et al. (2022), which this master's thesis is extending upon. The camera setup is the same as in the OFT dataset. In this dataset, however, all the dogs have a medical record from which it can be concluded that they are definitely in pain. Several dogs suffer from chronic pain, which makes their pain symptoms less prevalent, while other dogs are experiencing acute pain, which makes for more obvious pain signals. To ensure the experiment is ethically correct, the dog is not left alone in the room like the dogs in the OFT experiment. Instead, the dogs are under full supervision by a veterinary expert and their owner who are seated in opposite corners in the veterinary room. The dogs are recorded for about 10 minutes each and sometimes encouraged to move by their owner by making the owner walk to the other side of the room and call the dog. Additionally, to capture acute pain signals, the veterinary expert induces pain by touching the painful area of the dog at the start or near the end of the recording.

The in-the-wild dataset is not recorded in pre-defined space, but can be anywhere. Several videos are recorded in the same veterinary room as in the other parts of the dataset, while others are recorded at the dog's home or outside. In addition, the camera setup of the videos is different as well. Unlike a fixed and static camera setup as in the other parts of the dataset, the videos are recorded from a mobile phone, which is several times held in landscape mode and other times in portrait mode. The phone follows the dog through the room or surroundings. Due to this, the dog is sometimes obscured by people walking in front of the camera, and the camera person fails to keep the full dog in the picture. The duration of the videos is not fixed as well. The duration ranges from 20 seconds to videos of several minutes. Part of the videos capture a test. These tests consist of two bowls lying upside down on the ground, under which one bowl a treat is hidden. The dog should be able to solve under which bowl a treat is hidden and clarify in a way that it knows it is there, after which it receives the treat. These experiments were mainly conducted at the dog's own home and usually had the longest duration. Other videos captured how dogs were moving when walking outside with a person. These videos usually only last about 20 seconds.

4.1.2 Bias and Noise

In this dataset, there is some noise and bias present that degrades the quality of the dataset. In the dataset that was taken in the main veterinary room as shown in Figure 18, there was a noticeable difference in behavior between dogs during OFT video recordings and during recordings where the owner was present in the room for 10 minutes. During the OFT recordings the dogs seemed stressed, which made them prone to move more throughout the room, while the dogs with their owner present in the room seemed more relaxed. Due to the adrenaline suppressing pain signals in the dogs that were in pain during the OFT recordings, these dogs might be less likely to show visible pain symptoms, which might confuse the model. For the part of the dataset where the veterinary expert and owner is present, it is difficult to capture natural behavior of the dog. The dogs frequently interact with their owner and seek consolance because animals never find going to the vet an enjoyable experience. For instance, there is a lot of tail wagging involved when the dog sees its owner.

(a) A dog in pain from the OFT dataset. The toy car is located in the top right corner of the room.

(b) A dog in pain from the dataset created for Zhu et al. (2022).

Figure 18. Example frames from the two experimental settings in the veterinary room at Ankara University.

Additionally, the camera settings in the veterinary room caused the edges of the camera to be distorted, due to which a slight fish-eye effect can be seen. If the dog is located right underneath the camera, the dog goes out of the picture as well due to which pose estimation is difficult. If the dog is present on the other side of the room, the dog appears small and is of low resolution. Furthermore, due to low lightning conditions, some frames are blurry, and the dark coats of some dogs might make it more difficult to correctly predict keypoints for pose estimation.

A lot of noise is present in the in-the-wild dataset taken from a mobile phone. During numerous videos, at least 3 or more people are present and walking through the scene, sometimes blocking the camera. The videos are taken by hand and follow the dog through its activities, but often fail to capture the dog's body in its entirety, due to which proper pose estimation, as our model requires, is impossible. The camera is often shaking as well, making it more challenging to adequately capture the temporal flow of the dog adequately.

There is also a great variety in morphology and sizes, coat colors, and breeds of dogs. Since the dataset is small, this might make it difficult for the model to generalize, while on the other hand, it might also improve the model's robustness.

4.1.3 Pre-processing of the dataset

On the dataset, some preprocessing is performed to make the dataset more consistent and improve the results of the model. First, the side videos and duplicate videos of some dogs were removed to avoid overlapping dogs occurring in multiple folds. Furthermore, videos where the blinds of the window in the veterinary room were open were removed due to poor lighting conditions. These videos also incorporated an abundance of other objects

in the room, such as a couch and several toys, which might badly influence the model's performance. Additionally, in some videos, there were numerous redundant parts where no dog was present as it had not entered the room yet. Most of these parts were trimmed as an extra safety measure to avoid the model from incorrectly learning from data where no dog was present in the frame. The videos we will thus be using to train our models on are the videos with closed blinds and filmed from the front. For the in-the-wild dataset we will use all the videos.

4.2 Experimental Setting

In this section, the separate experimental settings for behavior estimation and pain estimation will be discussed. For both settings, the datasets mentioned earlier will be used.

4.2.1 Behavior Estimation

To obtain the behavior annotations, we have annotated 22 videos of dogs both in pain and without pain. These annotations were first created for the DeepAction pilot study with the Matlab program created by Harris et al. (2023). This pilot study can be viewed in the Appendix. We kept these annotations to use them in further studies, such as our current best study.

For the behavior estimation, we are using the off-the-shelf model MSQNet from Mondal et al. (2023). We train the model with a cosine decay scheduler and a starting learning rate of 0.00001 with an Adam optimizer. This starting learning rate produced the best results. Additionally, we will use a 10-epoch early stopping method that ensures that if the model does not improve after ten epochs, the model will stop, and the final test results will be obtained. Separate tests will be run using only behavior labels for the text input, using prompts, using a pre-trained model on the Animal Kingdom dataset, and we will be adding our in-the-wild dataset as well separately. With the best model from these results, we will conduct extra tests to separate the ethogram dimensions into the following sections: all behaviors, idle behaviors only, and all ethogram dimensions individually. We will use a 5-fold cross-validation approach for these tests where dogs are selected for the test set, which do not occur during training or validation. This ensures we test the model's generalization capability and robustness to unseen data. To measure the performance of the models we will be using mean average precision (mAP). We chose for this metric because it is widely used in computer vision (Lin et al., 2014; Russakovsky et al., 2015) and considers both precision and recall at different decision thresholds. This gives us a good overview of how the model performs at different levels of strictness in confidence. The authors Mondal et al. (2023) suggest a model input of clips from 16 frames, since this number of frames shows the best output in comparison to lower number of frames.

Unfortunately, we did not have the computational resources to run the model with 16 frames, thus we will be running the model with clips of 10 frames. This number showed better results than eight frames in the paper by Mondal et al. (2023). Additionally, data augmentation will be applied to the training inputs. The following augmentation will be used with a randomness factor: horizontal flip, color jitter, grayscale, gaussian blur, solarization (inverting pixels above a certain threshold), and random crop.

4.2.2 Pain Estimation

For the pain estimation part, we use models trained from scratch. We will refrain from using the in-the-wild dataset due to the poor pose representations, and only use the dataset recorded in the veterinary room. All experiments are run in a 5-fold cross-validation fashion, with no overlap between individual dogs in train/validation/test sets. As explained in the Methodology section, the ConvLSTM stream consists of four ConvLSTM blocks, followed by a max-pooling and batch normalization layer. The max-pooling has a stride of 2. The batch normalization layer has a momentum of 0.1 and an epsilon of 1e-5. We will experiment with the hidden layer sizes of the ConvLSTM, namely sizes 32, 64 and 128. The LSTM stream also consists of four LSTM blocks. For the LSTM blocks, we will be experimenting with hidden layers sizes of 32, 64, and 128 as well. The Ethogram stream consists of a linear layer followed by a batch normalization with a momentum of 0.1 and an epsilon of 1e-5. Subsequently a drop-out layer with a drop-out rate of 0.1 is applied, which results in a random 10 percent of the neurons being dropped. Ultimately, the streams will be fused by concatenation. A confidence score will be obtained by running the final features through a fully-connected network with a drop-out rate of 0.1 and putting the final output into a sigmoid activation function that results in a confidence score between $[0, 1]$

The input of the ConvLSTM and LSTM stream will consist of clips from 8 frames. There is an overlap of 2 frames in between clips to ensure no essential temporal dynamics are missed. The input of the ConvLSTM stream will be of the dimension $B \times T \times C \times W \times H$, where B is the batch size, T is the length of the clip which is 8 in our case, C the number of channels and W and H are the width and height of the frame respectively. In our case the images will be scaled to a $W \times H$ of 112×112 with three channels. The LSTM stream will receive an input of $B \times T \times P$, where P represents 17 keypoints of x, y-coordinates.

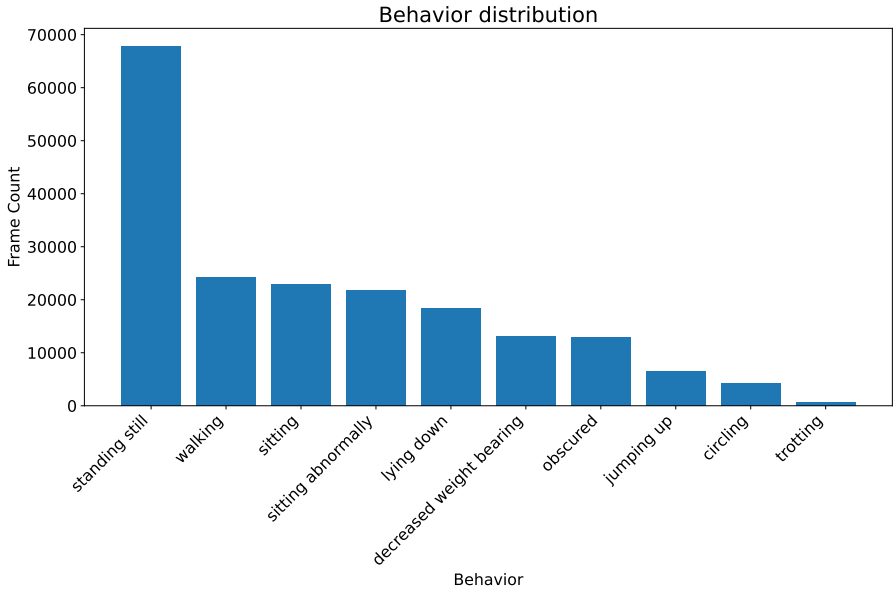
The input stream of the ethogram processing stream will be pre-processed as mentioned in Section 3.4.3. The ethograms will be arranged in the same manner as the ConvLSTM and LSTM input. Namely, the ConvLSTM input consists of 8 frames; thus, the input of the ethogram stream will consist of the ethogram present in those 8 frames.

Data augmentation will be applied to the input stream of the ConvLSTM and LSTM stream

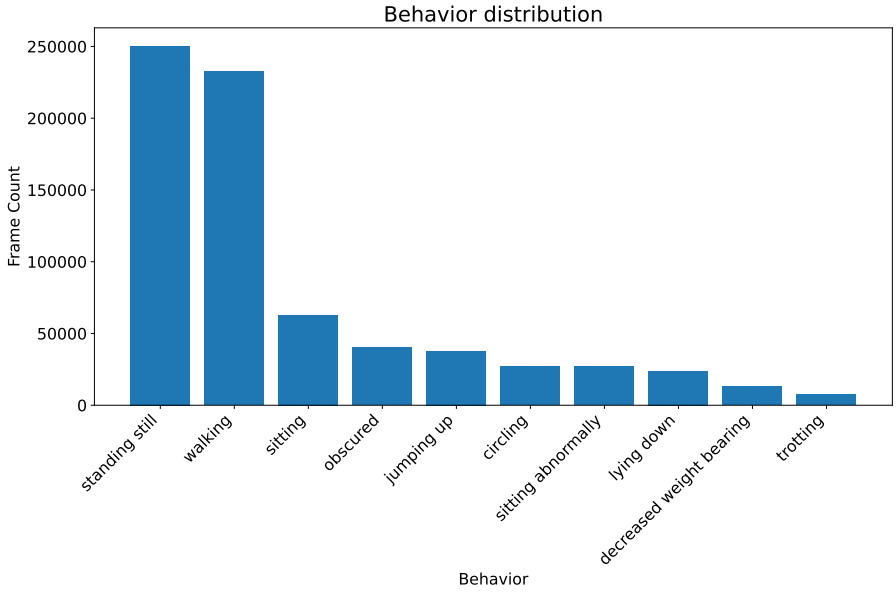
on the entire clip sequence. The LSTM stream data augmentation will be applied in the form of horizontal flips and rotations. The ConvLSTM will receive data augmentation through a horizontal flip, rotation, gaussian blur, and color jitter. All data augmentations depend on a randomness factor, and the entire sequence of frames in a clip receives the same data augmentation to preserve consistency.

4.3 Behaviour Estimation Evaluation

For our behavior estimation algorithm, we first needed to obtain annotated videos from dogs in pain. Since these do not publicly exist as far as our knowledge, we annotated the dog pain videos ourselves. The annotations are based on the behaviors in Table 4. These behaviors were composed together with The Veterinary Medicine Department of Ankara University to create a valid behavior set that reflected pain behaviors. Unfortunately, we had to cut out the shaking, rigid posture, and tense facial muscles. For the first two, there was too little or no data on these behaviors in our current dataset. For the latter, since our camera is static in our main dataset, we cannot get a good, consistent view of the dog's face. This makes it impossible to train a model based on the facial muscles. Based on the remaining behaviors we extracted several labels to represent these behaviors, which should be doable for a model to understand. Additionally, we constructed prompts for dogs who were filmed inside and outside. These can all be viewed in Table 5. We performed the annotations in the program provided by Harris et al. (2023) during one of our pilot studies, which can be viewed in the Appendix.



(a) Label distribution of self-annotated dogs.



(b) Label distribution of all dogs including predictions.

Figure 19. Label distributions of the behaviors.

For our dog behavior dataset we extracted the data distribution of the behavior labels for the data we annotated ourselves and the final predictions on the unlabeled videos. These can be viewed in Figure 19. From these graphs, we can deduce that we have to deal with a long-tailed distribution of data labels. This poses significant challenges for action recognition since the model might perform well on the majority classes but poorly on the minority. We pay extra attention to this by extracting confusing matrices to analyze the model's predictions.

In this section, we will analyze MSQNet's performance on our dataset. We will look at the performance of MSQNet when using only the labels, adding prompts, and using a pre-trained network. Additionally, we will look at how the model performs when we add our in-the-wild dataset to it. We will look at the performance of all behaviors, only idle behaviors, and several single dimensions of the ethogram and discuss the results.

Reluctance to move (walk, trot, jump)	Sitting or lying down in the middle of walks
Decreased weight bearing (limb pain)	Sitting abnormally (knee out in stiffle pain)
Restlessness, wandering circling	Shaking while changing posture (while sitting, lying, standing)
Rigid posture and gait	Tense facial muscles

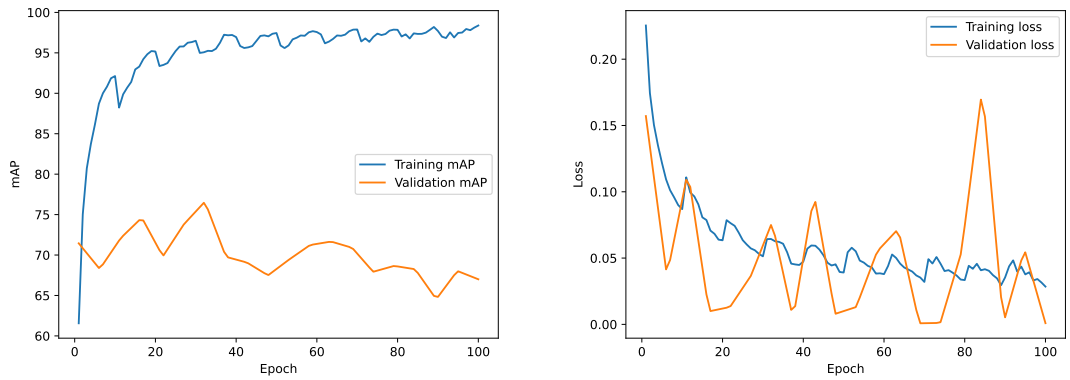
Table 4. Ethogram on pain behavior in dogs

Walking	*The dog is walking in a room, The dog is walking outdoors*
Trotting	*The dog is trotting in a room, The dog is trotting outdoors*
Jumping	*The dog is jumping up in a room, The dog is jumping outdoors*
Standing still	*The dog is standing still on four feet in a room, The dog is standing still on four feet outdoors*
Lying down	*The dog is lying down in a room, The dog is lying down outdoors*
Sitting	*The dog is sitting in a room, The dog is sitting outdoors*
Sitting abnormally	*The dog is sitting abnormally in a room, The dog is sitting abnormally outdoors*
Circling	*The dog is circling in a room, The dog is circling outdoors*
Decreased weight bearing	*The dog is avoiding to put weight on its limb*

Table 5. Labels and prompts based on the pain ethogram

4.3.1 Comparison between different models

We will compare the performance between several models: namely when using only labels, including prompts, pre-trained on animal kingdom including prompts, and including our in-the-wild dataset.



(a) mAP progress of the MSQNet model trained on all behaviors.

(b) Loss progress of the MSQNet model trained on all behaviors.

Figure 20. Training progress of the MSQNet model trained on all behaviors.

As we can see, adding prompts to the model significantly improves its performance. The label only model reached an average mAP of 67.97%, and the prompts model a mAP of 76.90%. This is an increase of 8.93 in mAP in comparison to the label-only model. This underscores that the model benefits considerably from the contextual information that the prompts provide, as opposed to relying solely on the embedding extracted from the behavior label. An example of a training progress for an MSQNet model trained on all behaviors with prompts is shown in Figure 20. We can see that the model slightly increases its performance in terms of mAP during the first 35 epochs, and starts to overfit beyond this mark. We see that the loss decreases gradually but exhibits a lot of fluctuation. Our model tends to fluctuate in performance due to the small size of the dataset. In addition to the small size of our dataset, we only annotated a part of the dataset due to limited time. To be exact, there is about 78 minutes of annotated video available to train, validate, and test on. This hinders the model to generalize effectively and perhaps causes the fluctuations in test and validation results. We will discuss the predictions of the model trained on all behaviors further in the next section.

MSQNet Model	Avg. mAP
Labels only	67.97 ± 1.87
Prompts	76.90 ± 3.45
Prompts + pre-trained on Animal Kingdom	67.20 ± 2.54
Prompts + In the wild	64.33 ± 2.92

Table 6. MSQNet results with all behaviors

To address the limitations of our small dataset, we pre-trained MSQNet on the Animal Kingdom dataset, which includes various animal species with a diverse behavior set in the wild. We ran the pre-trained MSQNet model on our dog pain dataset, including the prompts. Unfortunately, this did not yield any improvement. Compared to the model trained from scratch with the prompts, it showed a decrease of 9.70% in mAP. We presume this decrease in performance is because most of the animals in the Animal Kingdom dataset do not look like dogs, and the dataset has an in-the-wild setting. Due to the in-the-wild setting of the Animal Kingdom dataset, our MSQNet model might exhibit difficulties when generalizing the learned features from the Animal Kingdom dataset to a setting in a veterinary room, where dogs are filmed from an overhead perspective. The same can be seen when we compare the performance of the prompts model of MSQNet trained on the dogs only in the veterinary room and when adding dog videos from our in-the-wild dataset. Training on only the dog dataset in the veterinary room gives an average mAP of 76.90%, while adding the in-the-wild dogs to the dataset decreases the average performance by 12.57%. Since the in-the-wild dog pain dataset is vastly different from the veterinary setting and has a lot of noise, it is not unexpected that adding this dataset leads to a drop in performance.

4.3.2 Comparison between different behavior sets

In this section, we will analyze the MSQNet prompts model predictions on all behaviors at the same time and ethogram dimensions separately. For these experiments, we will not include the in-the-wild dog pain dataset since using this dataset leads to a drop in performance.

To determine how well our models are making predictions, we constructed confusion matrices, which can be seen in Figure 21. Here, we can see in the confusion matrix 21a that the $all\ behavior$ model excels in detecting whether a dog is walking or remaining idle. If it detects movement, it will predict the dog is walking, and if it senses that there is no movement, it will predict standing still. Additionally, when the dog is obscured it is capable of predicting this as well.

Given these observations, we decided to run a model that is trained on only idle behaviors. This simplifies the learning problem for the model since it only has to distinguish between

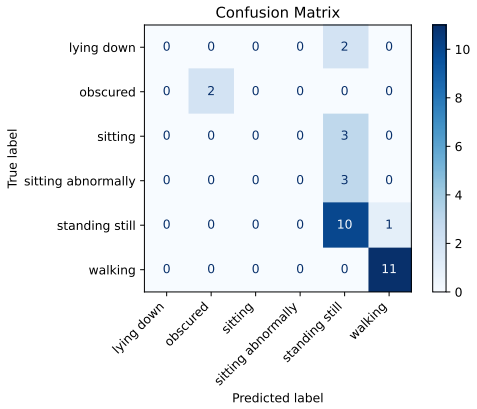
idle behaviors, and not moving behaviors as well. Figure 21b presents the confusion matrix results of the idle behaviors. The results indicate that the model is still able to predict standing still relatively well, although it sometimes gets confused with lying down and sitting. Knowing the difference between these behaviors is especially difficult when the dog is close to the camera, and only its back is visible and not its legs. Consequently, it is not unsurprising that the model makes these mistakes. Despite these challenges, the model is better at distinguishing between idle behaviors in general. However, quantifying the extent of improvement is hard since our test set is fairly small. We can see that it performs better at distinguishing between standing still on four feet and sitting behaviors. Nonetheless, it still has a hard time differentiating between abnormal and normal sitting. The idle behavior model shows a decrease in performance compared to the all behaviors model. The all behaviors model performs at a 76.90% mAP while the idle behaviors model runs at 68.16%. This gap in performance is not unexpected, since the learning task of idle behaviors is more challenging.

MSQNet Model	**Avg. mAP**
All behaviors	76.90 ± 1.87
Idle behaviors	68.16 ± 1.52
Walking	91.31 ± 2.25
Standing still	78.59 ± 2.17
Sitting	94.41 ± 4.56
Decreased weight bearing	84.41 ± 0.82

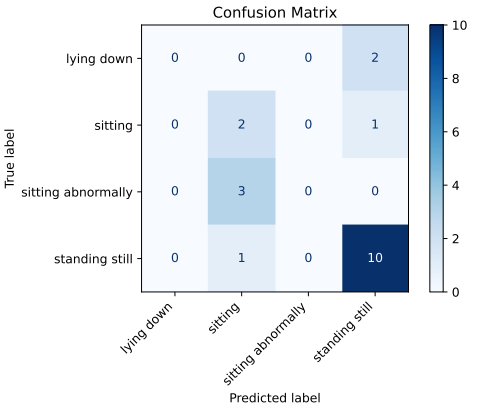
Table 7. Results between different behaviors, ran with the MSQNet model including prompts

In addition to all behaviors and idle behavior models, we also trained MSQNet on separate ethogram dimensions. We labeled the behaviors as follows. If the behavior was occurring in a frame it kept its original prompt and label. If the dog was performing a different action, the label would become *"not certain action"*, where *certain action* represents the single ethogram dimension we are trying to predict. The confusion matrices for the *standing still, walking* and *sitting* behaviors can be found in Figure 21c, d, and e, respectively. The walking behavior performs excellent performance with a mAP of 91.31, which was to be expected since the all behaviors model already excelled at differentiating between movement and idleness. The standing still behavior shows worse performance. Although the mAP is 78.59, we can see that the model tries to predict the majority class mostly and classifies a lot of *standing still* behavior as *not standing still*. The same applies to the sitting behavior, where the results are even worse, although the mAP of 94.41 is misleading in this aspect. Since there are so few sitting classes, the model almost only predicts the majority class to achieve a good mAP.

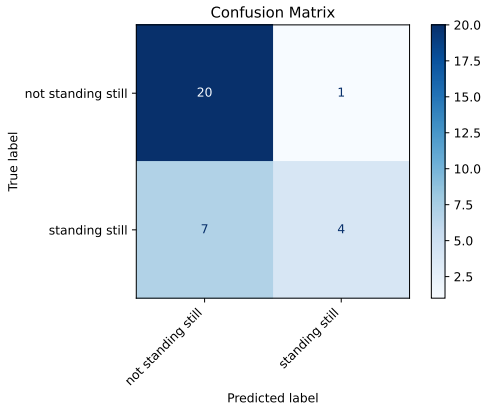
In addition to the primary behaviors, we also trained a model on a secondary behavior, namely *decreased weight bearing*. This behavior can occur during all the behaviors in the all behaviors model, which is why we trained a separate classifier for it to keep the learning problem simple. In Figure 21f we can see the confusion matrix of the results with a mAP of 84.41. The model is quite capable of correctly prediction when a dog is avoiding putting weight on one of its limbs. It has a little tendency to predict *decreased weight bearing* over *normal weight bearing*, but still is able to distinguish well between the two.



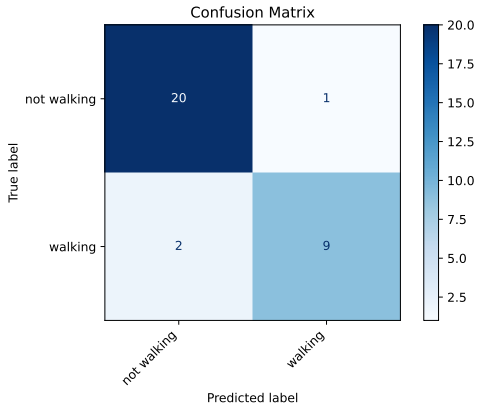
(a) Confusion matrix of all behaviors



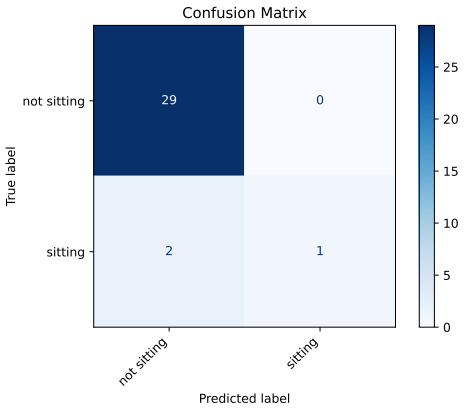
(b) Confusion matrix of idle behaviors.



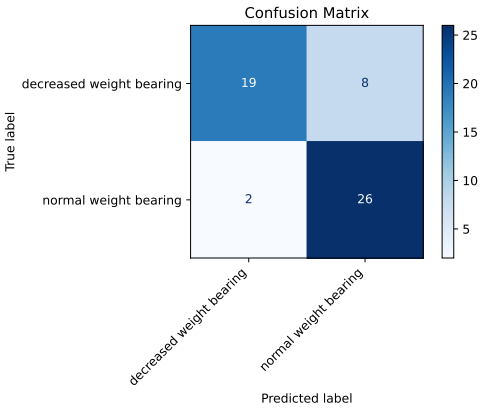
(c) Confusion matrix of standing still behavior.



(d) Confusion matrix of walking behavior.



(e) Confusion matrix of sitting behavior.



(f) Confusion matrix of decreased weight bearing behavior.

Figure 21. Confusion matrices for models trained on different ethogram dimensions.

4.3.3 Generating behavior predictions for unlabeled dog videos

For generating behavior predictions on the unlabeled dog videos, we used our best-performing model trained on all behaviors. For the standing still predictions generated by this model, we reran MSQNet with only idle options on these specific parts to generate more accurate predictions within the standing still dimension. To generate predictions within the unlabeled videos, frame sequence intervals of roughly 90-100 frames were generated with an overlap of 30% into the next interval to ensure no important behavior dynamics were missed. We followed an approach similar to Kim and Moon (2024), who also aim to predict dog behaviors with motion sensors. We chose this approach rather than action spotting since action spotting is a difficult task and we did not have the confidence that we could achieve proper action spotting with the small size of our dataset.

4.4 Pain Detection Evaluation

In this section, we will first discuss our discoveries when scrutinizing the results of the paper by Zhu et al. (2022) since these appeared to have faulty results. After this, we will continue evaluating the pose estimation on the new in the wild dog pain dataset. Subsequently, we will discuss the results of our new three-stream pain estimation model. Lastly, we will conduct an ablation study to determine how each component of the three-stream model performs separately and together.

4.4.1 Faulty results in previous paper

In the splits provided to us by Zhu et al. (2022), there was some overlap between the train and test set. Unfortunately, this was discovered more than halfway through the thesis, thus building upon something that was not accurate. To test the performance of the two-stream model, the side videos and double videos of dogs were removed to ensure a fair data split.

We removed the side videos since most dogs were filmed from the front, and only a few in pain were filmed from both the front and side. Additionally, this would help prevent the model from unfairly deciding a dog was in pain based on which angle the video was taken. Since there are only a few side videos, these might confuse the model rather than help it train.

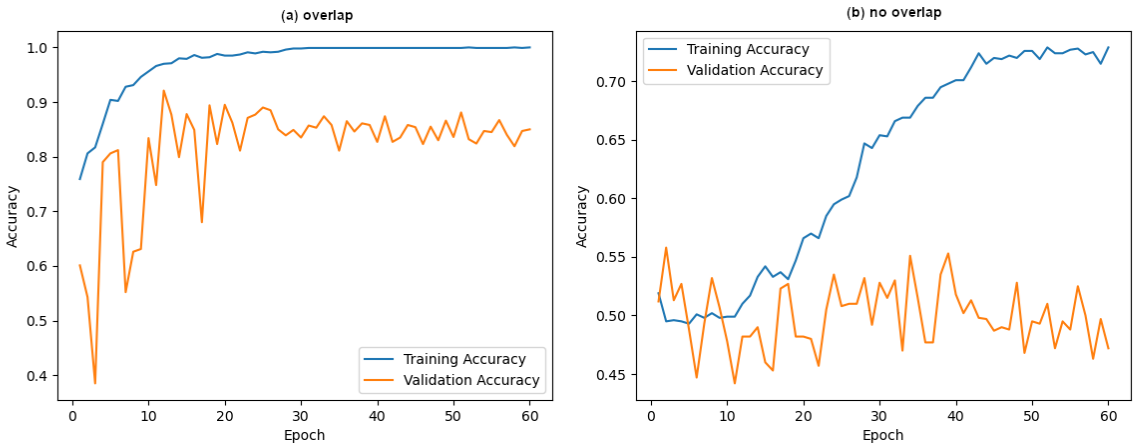


Figure 22. A comparison in accuracy between two data splits where Figure (a) shows the train and validation results of a split where there was an overlap between dogs and Figure (b) shows the results of a split where there is no overlap present.

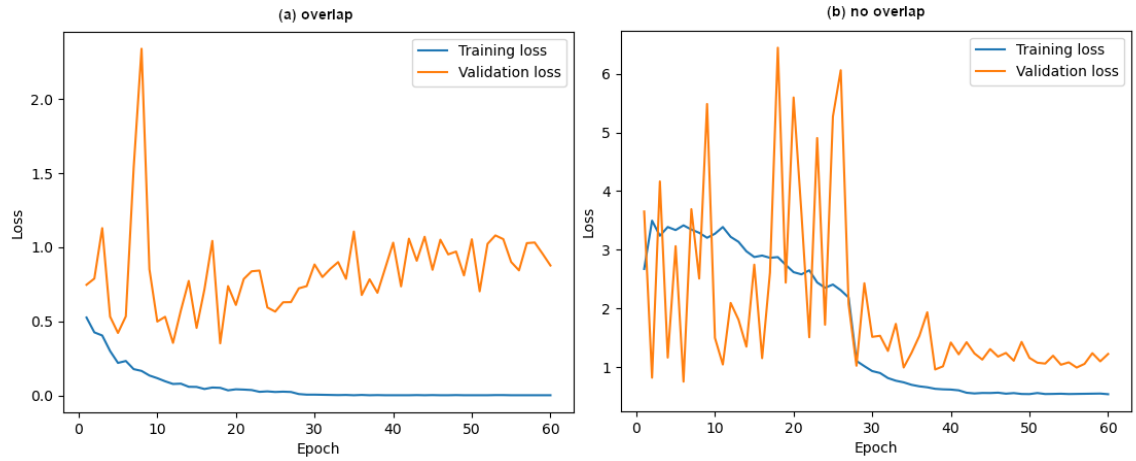


Figure 23. A comparison in accuracy between two data splits where Figure (a) shows the train and validation results of a split where there was an overlap between dogs and Figure (b) shows the results of a split where there is no overlap present.

We used the two-stream model with an LSTM with a hidden layer size of 32 and a ConvLSTM with a hidden layer size of 32. The accuracy reported in Zhu et al. (2022) is 77.0 ± 4.6 and the f1-score is 76.3 ± 4.4. When we run the LSTM128 + ConvLSTM model on the same splits of Zhu et al. (2022), we obtained a test accuracy of 82.1 and f1-score of 80.2 which is about the same as reported in Zhu et al. (2022). However, when we created fair data splits and ran exactly the same model, we obtained a test accuracy

0.51 of and a f1-score of 0.382. This is a tremendous drop in performance and shows that the model provided by Zhu et al. (2022) did not seem to learn at all and only recognized the dogs in the train set and memorized their labels. Graphs of the accuracies of both data splits are reported in Figure 22, and the loss process is reported in Figure 23. Additionally, we plotted the confusion matrix of the fair data split results in Figure 24. Here we can see that the model is mostly purely guessing 'no pain' and not learning at all.

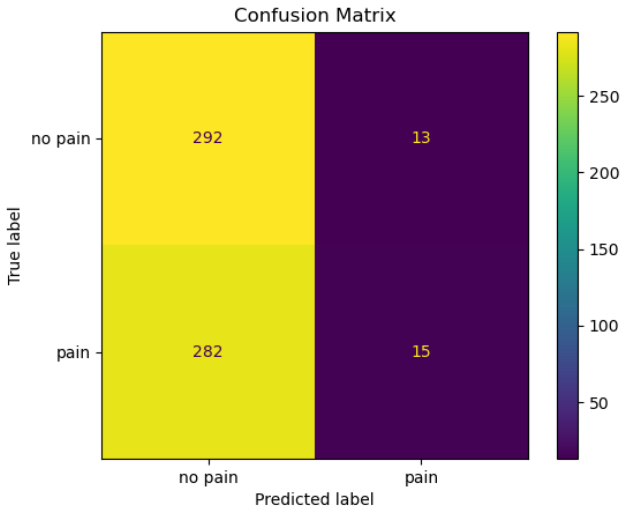
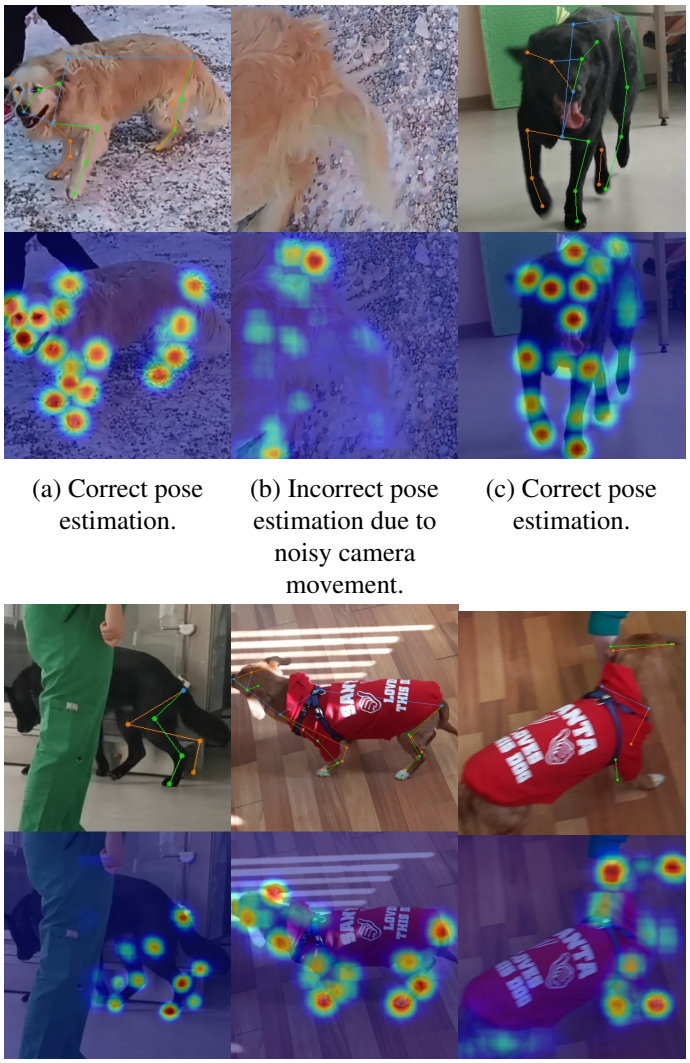


Figure 24. The confusion matrix obtained from the model trained on the fair data split. This shows that the model cannot properly recognize pain and mostly assumes that the dog has no pain.

4.4.2 Pose estimation

In this section, we will discuss the pose estimation results on the new in the wild dog pain dataset. We will not further describe the pose estimation on the datasets used in Zhu et al. (2022) since we already received the processed key points from those datasets, and pose estimation is not the focus of this thesis. However, we constructed the pose methodology once again to obtain the keypoints from the in-the-wild dataset. As discussed earlier, the in the wild dataset contains a lot of noise, which could make pose estimation difficult.



(a) Correct pose estimation.

(b) Incorrect pose estimation due to noisy camera movement.

(c) Correct pose estimation.

(d) Incorrect pose estimation due to person standing in front of the dog.

(e) Correct pose estimation.

(f) Incorrect pose estimation due to blurry camera and dog being partly out of frame.

Figure 25. Pose estimation results on the in the wild dog pain dataset.

In Figure 25, some of the results from the pose estimation on dogs from the in the wild dataset are shown. The correct pose estimations are shown in a, c and e, and the incorrect pose estimations are shown in 25b, d and f. Since this dataset consists of a lot of camera movement as opposed to the datasets with a static camera, it is often the case that the dog walks out of frame or there is noisy camera movement. In Figure 25b and f, the dog is not fully visible. This leads the model to be unable to detect crucial keypoints and will lead to this pose being discarded. In Figure 25d, we can see that HR-net cannot detect all the keypoints of the front side of the dog due to it walking behind a person in the room. In this video, a lot of people were present in the room. The dog was incredibly excited due to all the attention it was getting and was thus running through the room as a catapult, which

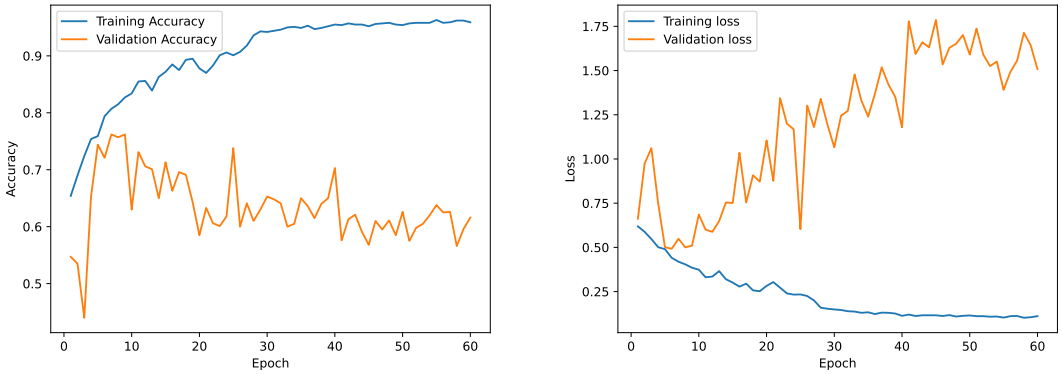
you can see in 25c. Due to this, the dog was obscured by persons or other attributes in the room for about half the time, and proper pose estimation was impossible. In 25f, the model could not detect keypoints properly since the frames were blurry due to excessive camera movement.

Due to these difficulties, many frames are discarded during the missing keypoint processing method, either because of a lack of fundamental keypoints such as the spine or simply too few keypoints. This will make training the pain estimation model on the in the wild dataset challenging.

4.4.3 New pain estimation results

Likewise for the behavior estimation, we will be using 5-fold cross-validation to test our pain estimation models. The train, validation, and test set will have the ratio of 0.8/0.1/0.1. We chose this ratio since we have little data to train on, so having as much data as possible is crucial. However, small validation and test sets are prone to fluctuations in the results. To ensure fair training, validation, and test splits, there is no overlap in the same dogs between splits. We obtained the accuracies and F1-scores for each model for each fold and calculated the mean value and standard deviation based on the results. F1 scores were used in addition to accuracy because F1 scores serve as the harmonic mean of precision and recall. This metric is especially useful when there are unbalanced datasets like ours. The results for all the models are shown in Table 8.

The model that showed the best performance was our LSTM128 + ConvLSTM32 + Ethogram linear layer model with data augmentation. This model has reached an average accuracy of 78.60% and an F1-score of 78.03%. This model performs significantly better than the three-stream model with the RF classifier, with an accuracy of 61.58% and F1 of 57.48%. An example training and validation graph on one of the folds is shown in Figure 26. For this run, we trained the model for 60 epochs to observe the model's behavior when trained over a more extended period. We can see from the accuracy and loss functions that the model peaks quite early in performance, a little before the 10th epoch. After this, the model starts overfitting, which can be seen from the accuracy dropping and the loss function increasing after epoch 10.



(a) Accuracy chart.

(b) Loss chart.

Figure 26. An example of training the LSTM128 + ConvLSTM32 + Ethogram linear layer model.

Model	Data type	No data augmentation		Data augmentation	
		Avg. Accuracy	Avg. F1	Avg. Accuracy	Avg. F1
One stream models					
ConvLSTM32	RGB	55.40 ± 11.48	50.58 ± 14.80	54.45 ± 7.5	52.50 ± 7.15
LSTM32	KP	56.18 ± 9.69	53.87 ± 11.66	54.68 ± 8.47	42.08 ± 14.97
LSTM64	KP	60.98 ± 13.28	59.05 ± 15.63	51.85 ± 6.00	41.70 ± 10.19
LSTM128	KP	57.85 ± 8.18	56.02 ± 7.80	50.53 ± 3.35	41.40 ± 6.26
Ethogram Linear	ED	76.95 ± 8.07	76.58 ± 7.97	-	-
Ethogram RF	ED	74.60 ± 14.16	72.23 ± 15.63	-	-
Two stream models					
ConvLSTM+LSTM32	RGB+KP	51.90 ± 4.71	46.90 ± 4.52	63.83 ± 14.12	63.33 ± 14.52
ConvLSTM+LSTM64	RGB+KP	60.05 ± 9.22	53.43 ± 14.37	56.63 ± 13.36	51.43 ± 15.89
ConvLSTM+LSTM128	RGB+KP	56.03 ± 15.58	52.05 ± 14.57	68.83 ± 12.01	65.38 ± 9.16
Three stream models					
ConvLSTM + LSTM128 + Linear	RGB+KP+ED	71.83 ± 8.08	66.60 ± 10.83	78.60 ± 3.65	78.03 ± 3.85
ConvLSTM + LSTM128 + RF	RGB+KP+ED	60.03 ± 10.62	55.55 ± 12.22	61.58 ± 4.75	57.48 ± 3.07

Table 8. Comparison between different models. KP stands short for keypoints, ED is short for ethogram dimensions, and RF represents random forest classifier.

To mitigate the risk of the model overfitting, we applied a 10-epoch early stopping algorithm. This algorithm will run for a guaranteed 10 epochs. Meanwhile, it keeps track of whether the loss is decreasing. If the loss has increased for 10 epochs, training will stop. If the loss decreases, its counter will reset.

Data augmentation

Data augmentation has varying impact on the different models shown in the ablation study in Table 8. The one-stream ConvLSTM and LSTM models in particular do not seem to profit from the data augmentation. This might be due to the fact that the one-stream models are relatively simple models with fewer parameters, which makes it difficult for the

models to capture complex patterns. If data augmentation is applied to the already small dataset, it might be even more difficult for the one-stream model to capture the patterns and worsen the performance. Although the overall average metrics of the one-stream models are worse with data augmentation applied, the fluctuations in performance are less when data augmentation is used. Where the standard deviation for accuracy and F1 of the LSTM models without data augmentation is primarily close to or above 10, the standard deviation with data augmentation is more often below 10.

The two-stream models mostly exhibit the opposite of the one-stream models, except for the ConvLSTM + LSTM64 model. They benefit more from data augmentation, which is most likely due to their more complex nature. The models without data augmentation even seem to perform worse than the one-stream models without data augmentation.

The three-stream models benefit from data augmentation as well, with an increase in accuracy from 71.83% to 78.60% for the linear model with data augmentation in comparison to the linear model without. The increase in variability in the dataset by the application of data augmentation seems to improve the performance of the models a lot. As can be seen from Figure 26, our model tends to overfit quickly, which is happening to the models trained on non-augmented data as well. The model overfits so quickly on the data that it cannot reach peak performance without data augmentation.

Two-stream models

When comparing the two-stream models to the one-stream models, the two-stream models, including data augmentation, seem to perform slightly better than the one-stream models. An explanation for this is that it might be difficult for RGB frames to capture detailed movement from joints from only the frames, whereas the LSTM stream that processes these keypoints can provide this information and vice versa. Combining these streams enriches the feature representation of the dogs, which can result in better predictions.

The effect of adding the ethogram stream

When comparing the three-stream model ConvLSTM + LSTM128 + Linear with data augmentation to the one-stream ethogram linear model, we see a slight increase in performance of 1.65% in accuracy and 1.45% in F1 score. While this is not a high increase in performance, it does suggest that adding the information of the RGB frames and keypoints to the ethogram stream helps the model understand what kind of behavior the dog is performing during these frames and aids in predicting whether the dog is in pain or not. Although the increase in performance does fall within the standard deviation of the ethogram linear one-stream classifier, it is questionable how significant this increase in performance is. This high standard deviation occurs due to our small test set, which causes a lot of fluctuations in our tests. On the other hand, the ethogram stream with a random

forest classifier experiences a loss in performance from 13.02% in terms of accuracy and 14.75% for F1 score. This suggests that the embeddings from the random forest classifier provide less contextual information about the behaviors of the model than the embeddings from the linear layer.

Adding the ethogram stream to the two-stream models significantly impacts their performance. When adding the ethogram stream, we can see that the ConvLSTM+LSTM128 model shows an improvement of 9.77% and 12.65% in terms of accuracy and F1 score, respectively.

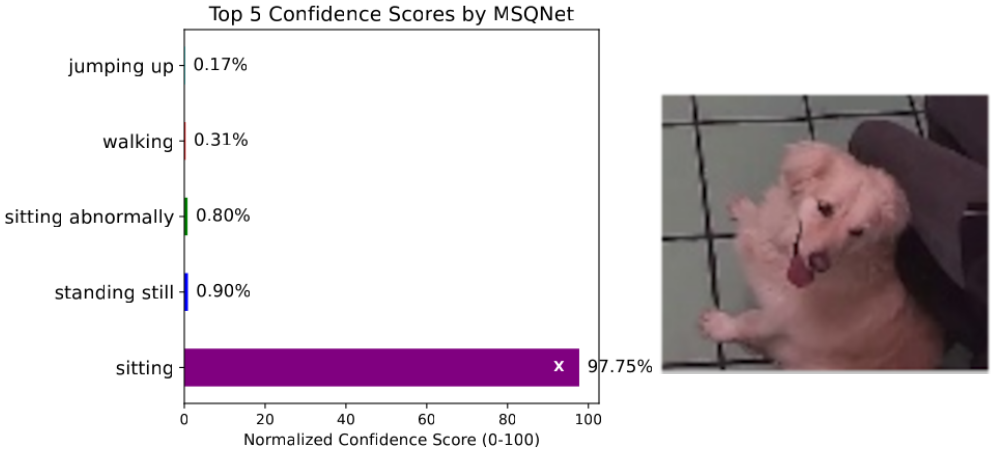
4.5 Explainability

To understand our models more thoroughly and what pain indicators might be for dogs, we conducted several explainability methods. For MSQNet, we will look at the top 5 and top 3 confidence scores and saliency heatmaps. For our three-stream pain estimation model, we will conduct several analyses. We will analyze Shapley values for our best-performing three-stream model with an ethogram stream with a linear layer. Additionally, we will look at the Naive Bayes probabilities extracted from the prediction output and the Grad-CAM saliency heatmaps. For our three-stream ethogram stream with a random forest classifier model, we will look at one of the decision trees comprising the random forest classifier and the feature contributions of several singular predictions.

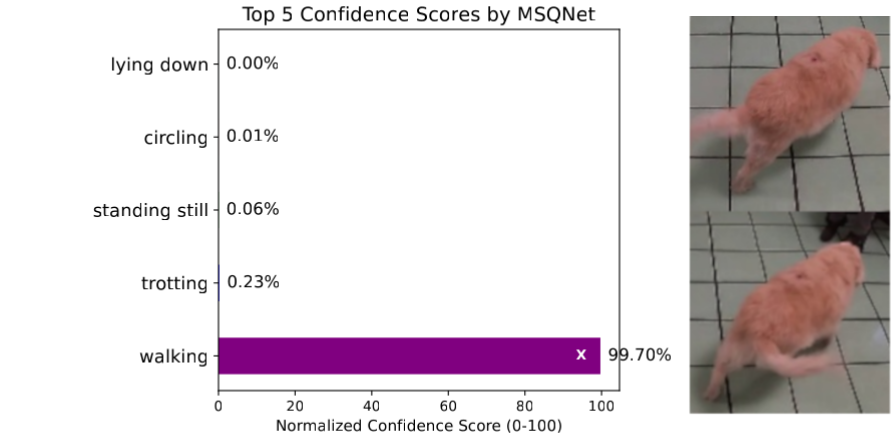
4.5.1 Behavior estimation

Confidence scores

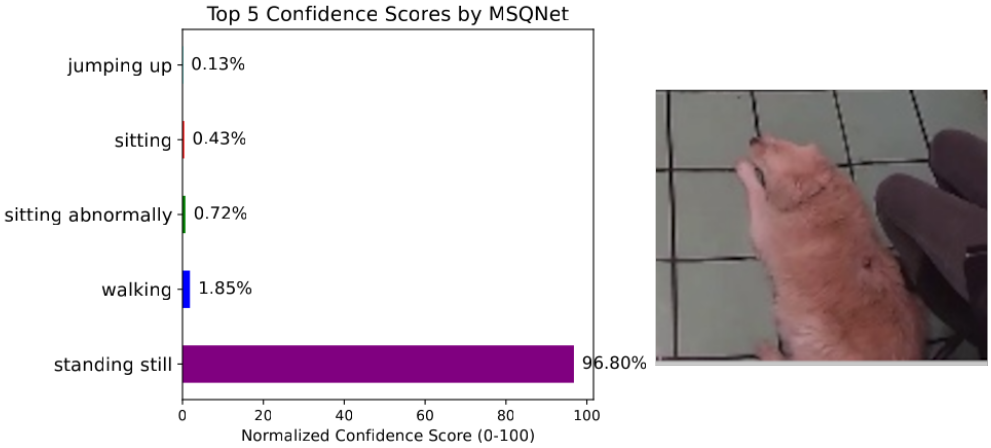
For the MSQNet model, we extracted confidence scores generated by a softmax Goodfellow et al. (2016) distribution over the model's scores for each possible behavior. We extracted the confidence scores from the model trained on all behaviors and the model trained on only idle behaviors. We chose this selection because, as mentioned earlier, the model trained on all behaviors is not great at distinguishing between idle behaviors. In contrast, our idle behavior model seems to be a little better at this. We can determine how confident our idle behavior model is by looking at the confidence scores.



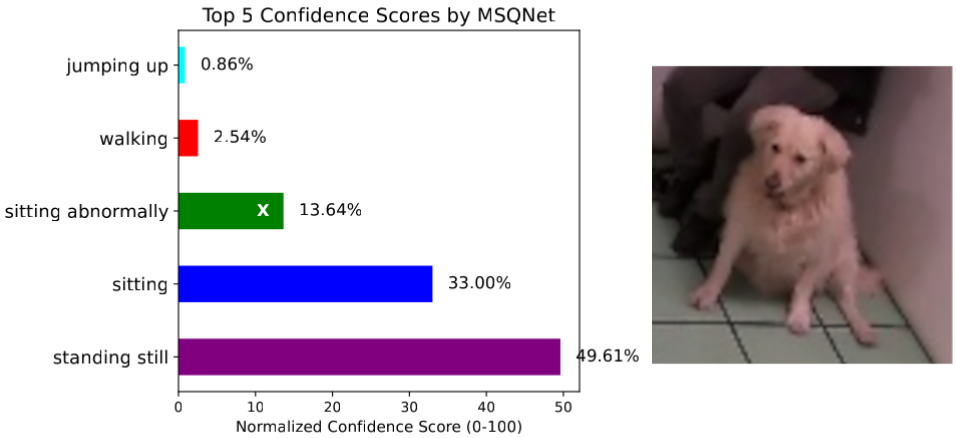
(a) Correct classification of sitting.



(b) Correct classification of walking.



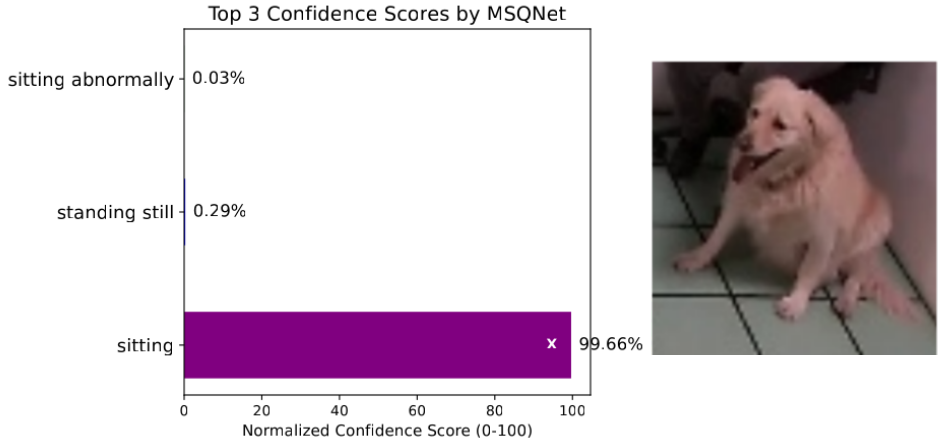
(c) Incorrect classification of lying down. The model predicts standing still instead.



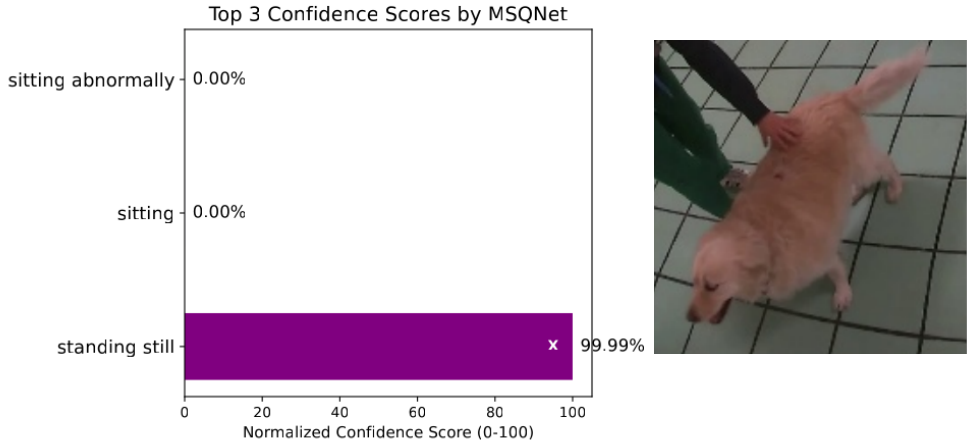
(d) Incorrect classification of sitting abnormally. The model predicts standing still instead.

Figure 27. Top 5 confidence scores for idle behavior classification of MSQNet. The model still has a hard time distinguishing between abnormal and normal sitting. It also still has a hard time classifying lying down. The correct label is marked with an x if present in the top 5 scores.

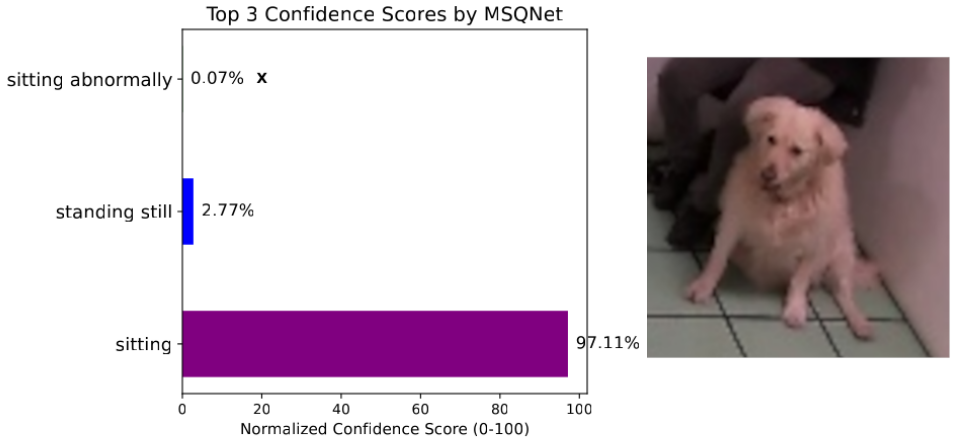
In Figure 27 the top 5 confidence scores are shown for several predictions. Each bar shows a percentage and the correct label is marked with a black or white x, if the correct label is in the top 5. We can see in Figure 27a and b two correct classifications of $sitting$ and $walking$. The model is incredibly confident in these predictions and gives it almost a 100 percent confidence score. In Figure 27c we can see an incorrect classification of lying down. Instead of lying down, the model predicts standing still. The model's confusion might be because the dog's hind legs are partly out of frame. Additionally, the point of view is from the top of the dog, making the leg position harder to distinguish between standing still and lying down. Nonetheless, the model is highly confident in the dog standing still, which makes it a significant error. In Figure 27d an incorrect classification of sitting abnormally is displayed. Here, the model predicts standing still instead of sitting abnormally. Its second-best prediction is sitting, and the third the ground truth label sitting abnormally. Although the model did not predict the correct label, the sitting labels are close to the main prediction. This shows again that the model trained on all behaviors is not great at distinguishing between sitting and sitting abnormally. However, these two classes are hard to differentiate between, even with the human eye, which makes it unsurprising that the model is not great at telling them apart.



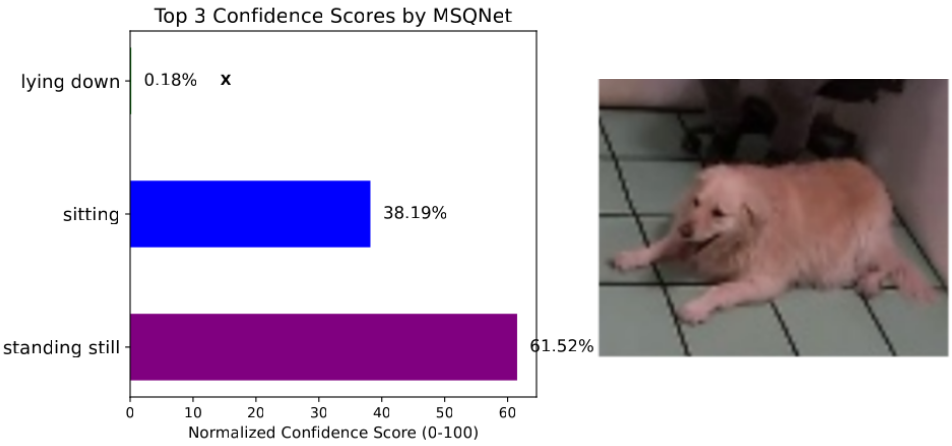
(a) Correct classification of sitting. The model is almost a 100 percent confident here.



(b) Correct classification of standing still. The model is almost a 100 percent confident here.



(c) Incorrect classification of sitting abnormally. The model has a hard time distinguishing between normal and abnormal sitting.



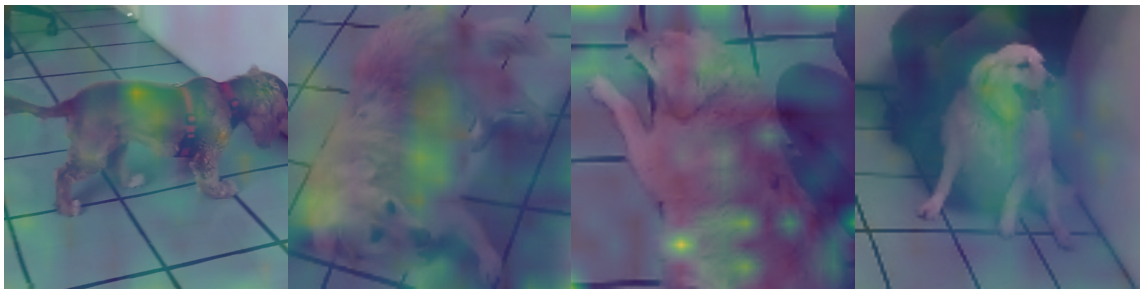
(d) Incorrect classification of lying down. The model predicts standing still instead and has a high score for sitting as well.

Figure 28. Top 3 confidence scores for idle behavior classification of MSQNet. The model still has a hard time distinguishing between abnormal and normal sitting. It also still has a hard time classifying lying down. The correct label is marked with an x if present in the top 3 scores.

In Figure 28, the top 3 confidence scores of the MSQNet model trained on idle behaviors only, are shown for several instances. In the top subfigures we can see two correct classifications of sitting and standing still, in which the model is nearly 100 percent confident about its prediction. As shown in the confusion matrix in Figure 21, we can see that the model performs well at distinguishing between sitting and standing still but still has trouble telling other behaviors apart that are not sitting. As shown in Figure 28c, the model still has trouble distinguishing between sitting and sitting abnormally. The model trained on all behaviors actually performed better on this particular instance. However, even with the human eye, it is hard to differentiate between Figure 28a and c, since the only difference is that the dog sticks its leg out in the abnormal position. This might be hard to detect without skeleton analysis. Like the all-behavior model, this model also fails to predict lying down, as shown in Figure 28d. It predicts standing still, although it is unsure if the dog is not sitting. Lying down is the third most likely label, but it has only a confidence score of 0.18 percent.

Attention heatmap

To figure out where the MSQNet models attend to the most in the frames, we extracted the attention layers from the transformer heads. The attention layers were averaged across all heads to generate a single heatmap. Figure 29 shows the attention heatmaps below.



(a) Correct classification of walking with a confidence score of 98.67%

(b) Correct classification of standing still with a confidence score of 87.63%

(c) Incorrect classification of standing still with a confidence score of 61.23%. The correct label is lying down.

(d) Incorrect classification of sitting with a confidence score of 90.10%. The correct label is sitting abnormally.

Figure 29. Attention heatmaps for the MSQNet model for correct and incorrect classifications of behaviors.

We can see that our model is still noisy in where it attends. In most figures, it is not solely focused on the dog but on a lot of its surroundings as well. This can be seen by the green hues not only being located on the dog. In Figures 29a and b, we can see the correct classifications of the majority classes. While the most intense hues can be found on the dog's body, there are still blobs on the surroundings. In Figure 29c we can see a *lying down* behavior incorrectly classified as *standing still*. The model focuses on parts of the dog's back but cannot determine whether the dog is lying down or standing still. This might be due to the dog being partly out of frame due to the camera angle. In Figure 29d, we see a *sitting abnormally* behavior being incorrectly classified as *sitting*. As seen earlier in the confusion matrices and confidence scores, the model has difficulties distinguishing between sitting and sitting abnormally since there is no big difference between the two poses. In the attention heatmap, we can see that the model is correctly focusing on the dog's body, but to correctly classify *sitting abnormally*, it should have focused more on the legs as well. There is still much room for improvement here to make the model concentrate less on the background noise and more heavily on the dog's body.

4.5.2 Pain estimation

For the three-stream pain estimation model, we will be using our best-performing model, the LSTM128 + ConvLSTM32 + Ethogram stream with a linear layer, for extracting the Shapley values, Naive Bayes Probabilities and Grad-CAM analyses. For explainability purposes, we will also extract a decision tree and its feature contributions for several predictions from the three-stream model comprised of the LSTM128 + ConvLSTM32 + Ethogram stream with random forest classifier. This section will first discuss the analyses of the first model and, subsequently, the latter.

Shapley values

Shapley values help us identify which features have the most influence in making predictions. In Figure 30, a bee swarm plot is shown, plotting for each feature, the Shapley value of every single prediction. This gives us an insight into how the model makes its predictions.

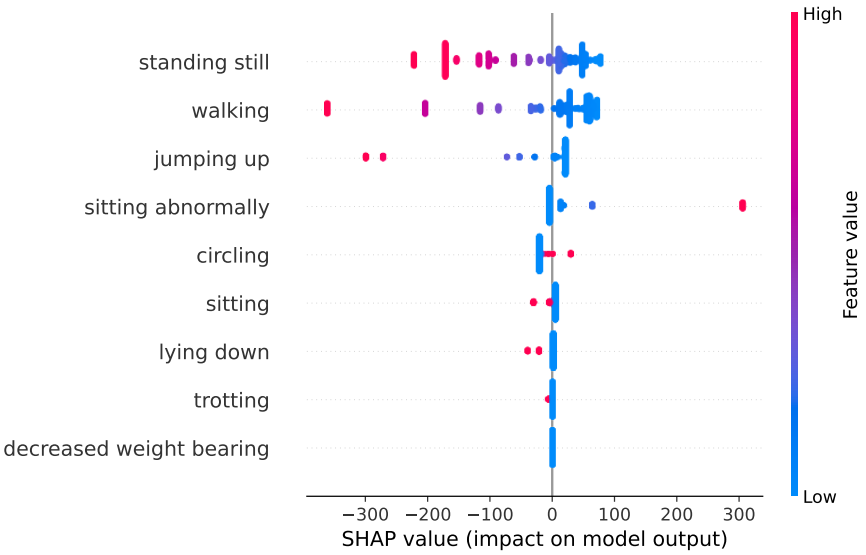


Figure 30. A beeswarm plot that plots the Shapley values of all the instances in the model. As we can see, low occurrences of walking and standing still contribute to the pain prediction. High occurrences of jumping contribute to no pain, and high occurrences of sitting abnormally contribute to pain. The other features have less impact.

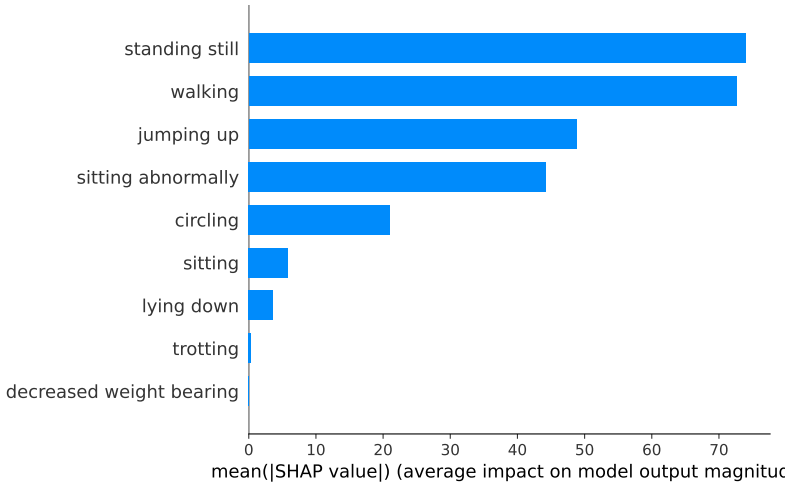
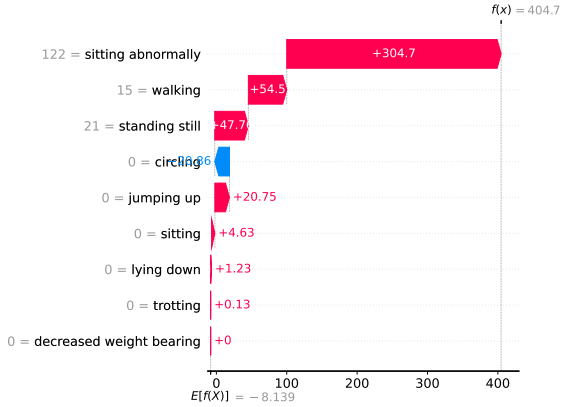


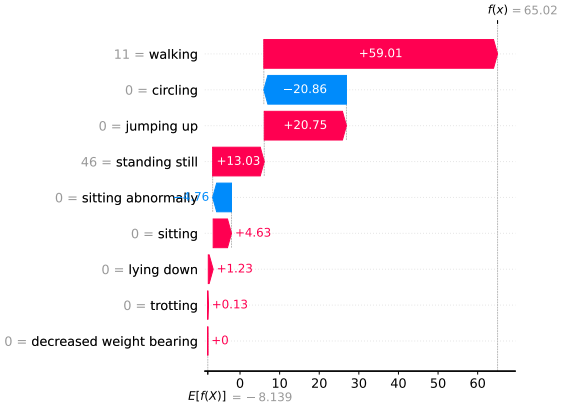
Figure 31. A barplot that shows which features have the highest impact on the model output, by taking the mean Shapley value over all instances.

In the beeswarm plot the features with the highest impact are plotted at the top and the features with the lowest impact at the bottom. Another clear overview of which features

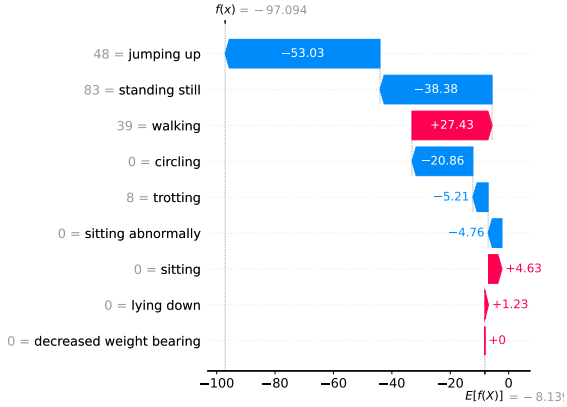
have the highest impact on the model is shown in Figure 31 in the form of a barplot. We can see that the standing still and walking dimensions have the highest impact, after which jumping up and sitting abnormally seem to have a significant effect. The remaining features seem to have much less of a substantial impact on the model's output. In the beeswarm plot, we can perceive that high occurrences of standing still seem to lead to the model to predict $nopain$, while low occurrences of standing still do not. The same applies to the $walking$ dimension. An explanation for this might be that dogs in pain tend to make themselves more comfortable in other positions than standing still on their four paws, while healthy dogs have no trouble standing still for longer periods. Namely, dogs in pain seem to exhibit more sitting abnormally behavior according to the beeswarm. This position could be preferred in contrast to standing still. Additionally, we can see that the model links high jumping occurrences toward dogs, not in pain. Low occurrences of jumping have less impact on the model. Circling seems more representative of dogs in pain, which can indicate restlessness due to discomfort or pain. Normal sitting and lying down seem to be slightly connected to dogs who do not experience pain. Surprisingly, the model did not learn that decreased weight bearing is a significant feature for predicting pain. Overall, the model seems to link a higher activity towards dogs not in pain and a lower activity to dogs that experience pain, except for the circling dimension. The circling dimension however, might indicate restlessness, which is a behavior related to pain that we discussed earlier in Section 4.3.



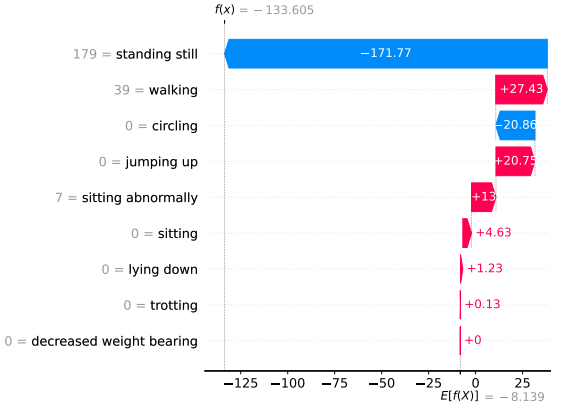
(a) True positive. A dog in pain is classified as pain.



(b) False positive. A dog without pain is classified as pain.



(c) True negative. A dog without pain is classified as a dog without pain.



(d) False negative. A dog with pain is classified as a dog without pain.

Figure 32. Shapley value contributions for specific instances. The $f(x)$ at the top right corner represents the model's output for this particular instance, while the $E[f(X)]$ represents the average model output of the training set.

To look at the contributions of each feature for single predictions, we plotted four waterfall plots in Figure 32. We made sure to plot True positive, False positive, True negative and False negative instances to provide a fair overview of the model's performance.

In Figure 32a we can see a dog in pain correctly classified as pain. The high occurrence of sitting abnormally contributes a lot towards the model predicting pain. A low presence of walking, standing still and jumping up push the model slightly towards a pain prediction. On the other hand, the absence of circling behavior pushes the model slightly towards the no pain side.

In Figure 32b we can see a false positive instance, namely a dog without pain is classified as pain. The highest contribution to this prediction is that there is a low presence of the walking behavior, which leads the model to think the dog is in pain. The healthy

dog correctly classified as no pain in Figure 32c is classified as such because of a high occurrence of jumping and standing still, which pushes the model to the no pain side. The relatively low presence of walking pushes the model slightly towards the pain side. Additionally, the absence of circling pushes the model slightly to the no pain side. The other feature do not seem to make much of a difference in the output.

A false negative example, which is a dog in pain incorrectly classified as having no pain, shown in Figure 32d. This dog has its prediction owing to its high presence of standing still, which the model connects to dogs who are not in pain. Additionally, the absence of circling also gives the model an extra boost in think the dog is not in pain. The low occurrence of walking, jumping up, and sitting abnormally tries to push the prediction towards the pain side but to no avail. The other features do not seem to make a big difference.

Naive Bayes probabilities

Behavior	**Behavior count**	
	Pain	**No pain**
Standing still	15273	14955
Walking	7508	28264
Trotting	0	157
Circling	1117	1865
Jumping up	0	2337
Sitting	488	368
Sitting abnormally	2370	258
Lying down	0	2672
Decreased weight bearing	14943	6703
Total	41699	57579

Table 9. The raw behavior counts from the test set

To give a clear overview of the distribution of the behavior variables, we will be extracting the behavior counts from the predictions. We will establish Naive Bayes probabilities with these behavior counts, giving us a basic insight into how the models assign behaviors to classes. The raw behavior counts can be consulted in Table. The prior probabilities for the pain and no pain classes based on the behavior counts in the predictions are $P(\text{Pain}) \approx 0.4200$, $P(\text{No Pain}) \approx 0.5800$. The model has thus a slight bias towards predicting no pain over pain. The posterior probabilities were calculated and are as follows:

Posterior Probabilities for Pain	**Posterior Probabilities for Not Pain**
$P(\text{'Pain'} \mid \text{'standing still'}) \propto 0.1538$	$P(\text{'No Pain'} \mid \text{'standing still'}) \propto 0.1506$
$P(\text{'Pain'} \mid \text{'walking'}) \propto 0.0.0756$	$P(\text{'No Pain'} \mid \text{'walking'}) \propto 0.2847$
$P(\text{'Pain'} \mid \text{'trotting'}) \propto 0.0$	$P(\text{'No Pain'} \mid \text{'trotting'}) \propto 0.0016$
$P(\text{'Pain'} \mid \text{'circling'}) \propto 0.0113$	$P(\text{'No Pain'} \mid \text{'circling'}) \propto 0.0188$
$P(\text{'Pain'} \mid \text{'jumping up'}) \propto 0.0$	$P(\text{'No Pain'} \mid \text{'jumping up'}) \propto 0.0235$
$P(\text{'Pain'} \mid \text{'sitting'}) \propto 0.0049$	$P(\text{'No Pain'} \mid \text{'sitting'}) \propto 0.037$
$P(\text{'Pain'} \mid \text{'sitting abnormally'}) \propto 0.0239$	$P(\text{'No Pain'} \mid \text{'sitting abnormally'}) \propto 0.0026$
$P(\text{'Pain'} \mid \text{'lying down'}) \propto 0.0$	$P(\text{'No Pain'} \mid \text{'lying down'}) \propto 0.0269$
$P(\text{'Pain'} \mid \text{'decreased weight bearing'}) \propto 0.1505$	$P(\text{'No Pain'} \mid \text{'decreased weight bearing'}) \propto 0.0675$

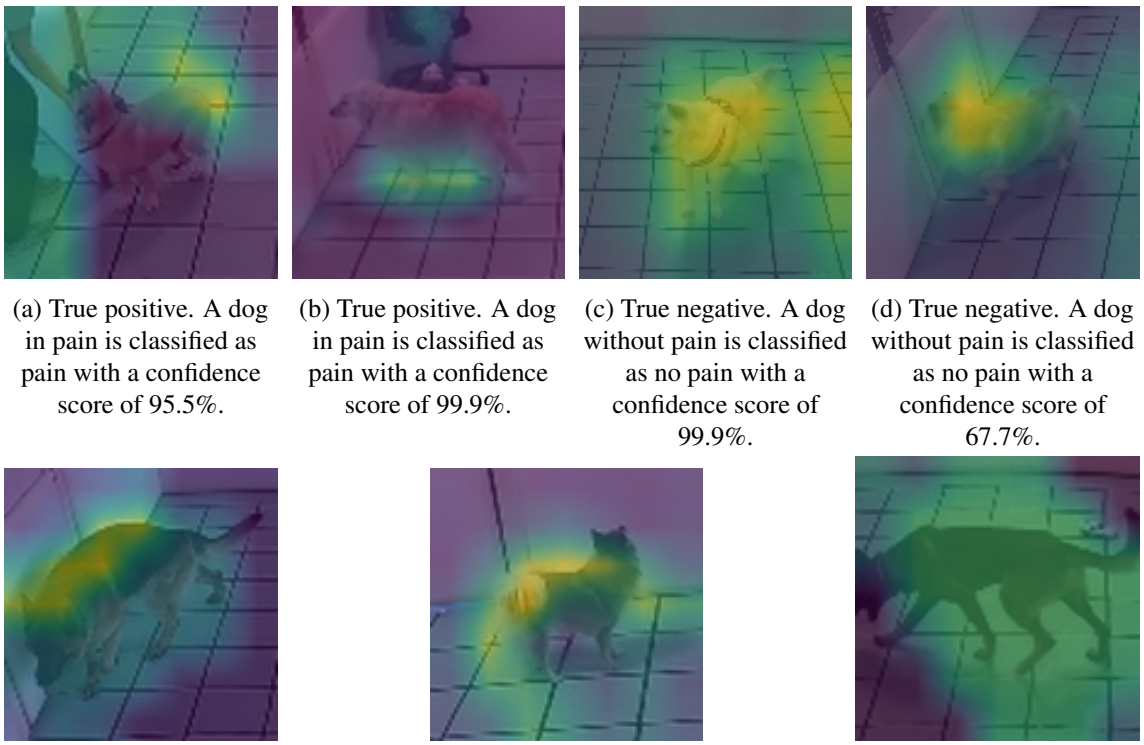
When we compare the posterior probabilities with each other to determine which behavior is more likely to occur during which prediction we get the following list:

- **Standing still:** No Pain (0.1506) < Pain (0.1538)
- **Walking:** No Pain (0.2847) > Pain (0.0756)
- **Trotting:** Only present in No Pain (0.0016)
- **Circling:** No Pain (0.0188) > Pain (0.0113)
- **Jumping up:** Only present in No Pain (0.0235)
- **Sitting:** No Pain (0.0037) < Pain (0.0049)
- **Sitting abnormally:** No Pain (0.0026) < Pain (0.0239)
- **Lying down:** Only present in No Pain (0.0269)
- **Decreased weight bearing:** No Pain (0.0675) < Pain (0.1505)

In this list, we can see that according to the predictions made by the three-stream model with a linear layer for the ethogram stream, the behaviors *standing still*, *sitting*, *sitting abnormally* and *decreased weight bearing* are more likely to occur in the pain class. Whereas the behaviors *walking*, *trotting*, *circling*, *jumping up* and *lying down* are more prominent in the no pain class. Overall, we can conclude from this that according to the Naive Bayes probabilities, the more active behaviors are linked towards no pain and the latent behaviors towards pain.

Grad-CAM

For extracting the Grad-CAM heatmaps we extracted the output of the last ConvLSTM block in the ConvLSTM stream. In Figure 33, we see several correctly and incorrectly classified dogs from the LSTM128 + ConvLSTM32 + Linear layer pain model. The correctly classified dogs can be seen in the top row and the incorrectly classified dogs in the bottom row.



(a) True positive. A dog in pain is classified as pain with a confidence score of 95.5%.

(b) True positive. A dog in pain is classified as pain with a confidence score of 99.9%.

(c) True negative. A dog without pain is classified as no pain with a confidence score of 99.9%.

(d) True negative. A dog without pain is classified as no pain with a confidence score of 67.7%.

(e) False positive. A dog without pain is classified as pain with a confidence score of 92.1%.

(f) False positive. A dog without pain is classified as pain with a confidence score of 54.8%.

(g) False negative. A dog with pain is classified as a dog without pain with a confidence score of 68.1%.

Figure 33. Grad-CAM results for the LSTM128 + ConvLSTM 32 + Ethogram linear model.

In the Figures 33a and b, we see two dogs experiencing neural pain. The dog in 33a shows decreased weight bearing and is not putting weight on one of its hind legs. We can see that the model focuses a little more on the hind of the dog, although it is not perfect, which suggests that the model sees that the dog is not using its hind normally. For this dog, the model is 95.5 % confident about its classification. In 33b a dog is present with a confidence score of 99.9 %. The model seems to focus on the dog's legs, which might suggest the dog experiences pain in its legs. This dog is experiencing neural pain in particular, although it is unknown in which area the neural pain manifests. The Grad-CAM suggests that this is occurring in the legs. When looking at the video, the dog is moving stiffly, so this is not an unexpected prediction from the model. In c, we can see a dog that is not experiencing pain and is correctly classified by the model with a confidence score of 99.9%. The Grad-CAM

heatmap focuses on the dog, but not necessarily on a particular part of the dog. The suggests that the model notices that the dog is not in pain, so it does not need to focus on the painful body parts. In d, on the other hand a dog without pain is correctly classified as no pain but only with a confidence score of 67.7%. The model is less confident here and we can see in the heatmap that it focuses less on the dog as a whole, but focuses more on the head/back of the dog. The model might think the dog experiences some pain there but is unsure, so it still classifies it as no pain but with less confidence.

In the bottom row, the incorrectly classified dogs are shown. In Figure 33e and f a False positive result is displayed which show a dog without pain classified as pain. The model's attention here is focused on the dog's back. This leads us to think that the model might think the dog experiences pain in its back. For the case in e, the model is pretty confident about its prediction with a confidence score of 92.1%, whereas in f, it is less confident with a score of 54.8%. In the last image g of the row a False negative instance is displayed where the model has a confidence score of 68.1%. We can see in this image that the model recognizes the dog but does not heavily focus on it in particular. Namely, where we can see more critical yellow parts in the other pictures, this is not present here.

To summarize, the model seems capable of recognizing the dogs, and its task is to detect pain, which can be seen in the correctly classified dogs. We can see that the model has a broader focus on the dogs it classifies as having no pain and a more particular emphasis on the body parts of dogs classified as pain. However, it is not always successful in detecting which dog is in pain, and sometimes, it tends to focus on other parts of the environment, such as the people in the room.

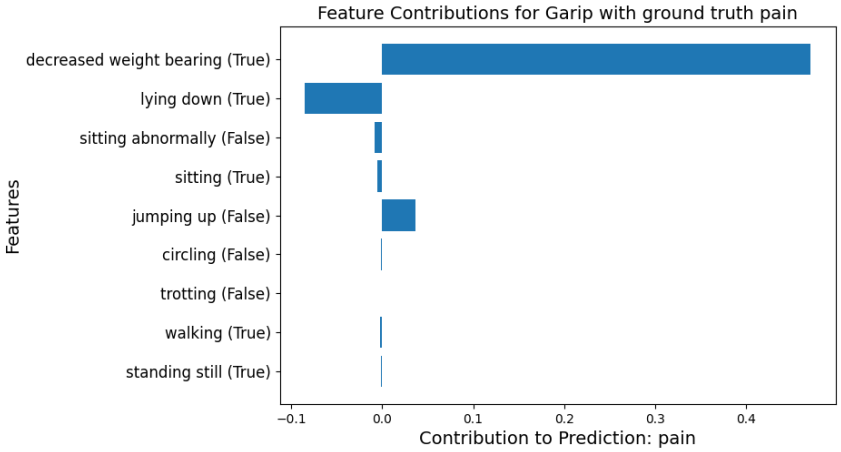
Decision tree

For our three-stream pain model with a LSTM128 and ConvLSTM32 and including the random forest classifier as the ethogram stream, we extracted a decision tree from the random forest classifier, which can be consulted in the Conclusion 35. The decision tree nodes work as follows. In every node, a condition is displayed, such as the root node $sitting\ abnormally\ \leq 0.5$. If this condition is $True$ it means that sitting abnormally is not present in this part of the clip, if it yields $False$ it is part of the clip. If a condition yields $True$ you traverse left down the tree, and if it is $False$ you traverse right.

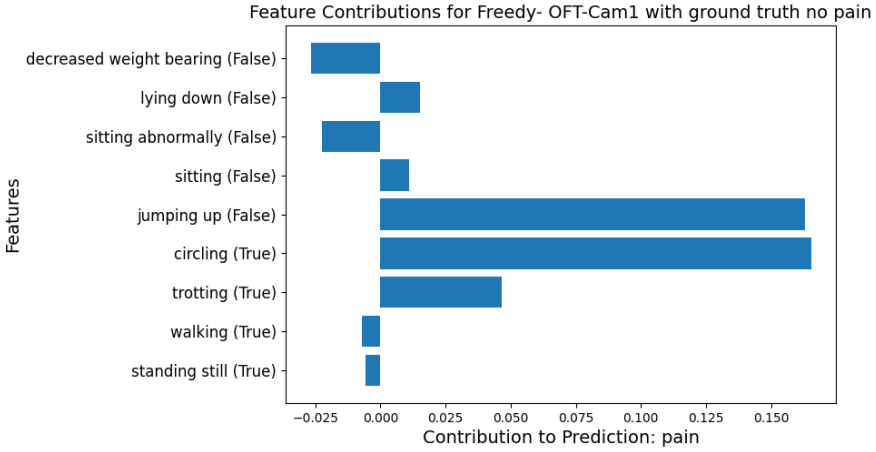
The root node of the decision tree is the feature that is the most influential in separating the data and reducing impurity in the child nodes. This suggests that the feature of the root node has the highest contribution in predicting our target variable pain. In our case, the most significant feature is thus $sitting\ abnormally$ which immediately separates the data into a leaf node that yields pain if the dog sits abnormally. The child node at depth

1 is $decreased\ weight\ bearing$, and has the second highest impact on the output. This node separates the data again in a leaf node that yields True if the behavior is present, and traverses further down the tree if not. Deeper in the tree we can see that the presence of trotting is an indicator of pain, and the absence of walking and jumping as well.

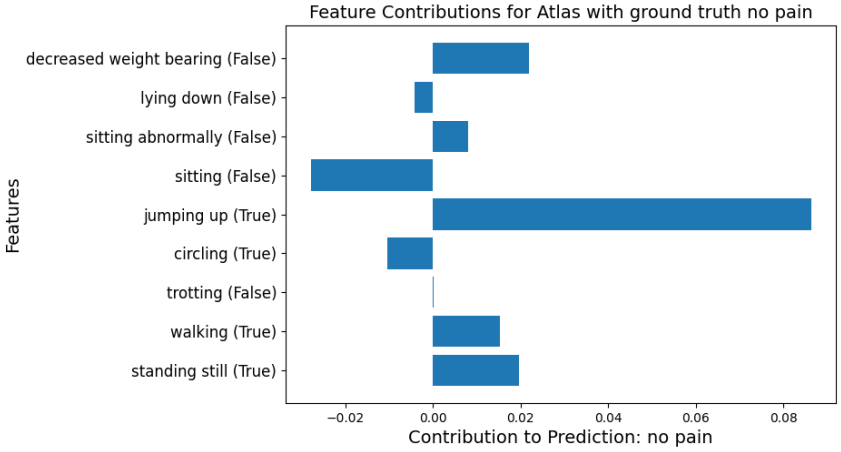
For the predictions of the random forest classifier, we extracted the feature contributions for several singular instances. Similar to previous experiments, we will show a True positive, True negative, False positive, and False negative. The feature contributions are shown in Figure 34. In Figure 34a, we can see a correct classification of a dog in pain. The feature with the most significant contribution here is the presence of decreased weight bearing. This dog experiences neural pain in its leg, so it does not put weight on its hind leg. The random forest classifier learned correctly that this is a strong indicator of pain. Lying down pushes the model towards a no pain prediction, but the decreased weight bearing wins a lot of ground here. In Figure 34b, there is a dog that shows a lot of circling behavior, leading the model to predict pain. While this seems to be a strong indicator of pain, this is not the case for this dog, due to which it was falsely classified as pain. In Figure 34c, a dog without pain is correctly classified as a dog without pain. The most significant contribution to the no-pain classification is that it displays a lot of jumping behavior, which is one of the leaf nodes that yields pain, as seen in the decision tree in Figure 35 in the Conclusion. In Figure 34d, we can see a False-negative example. This dog is seen as a dog without pain, but actually is in pain. The features that contribute the highest to this prediction are the presence of standing still, sitting and the absence of decreased weight bearing. The absence of walking is a feature that tries to push the model towards the prediction of pain fiercely, but the model still has enough weight from the positive contributions to predict no pain.



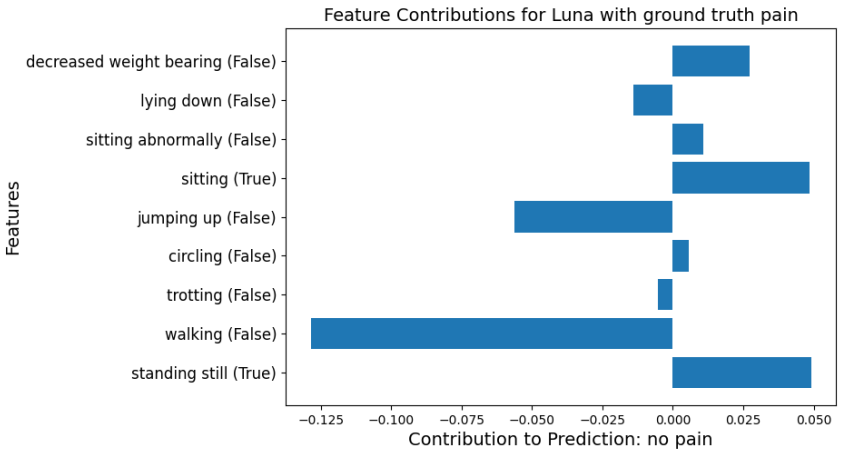
(a) True positive. A dog in pain is classified as pain.



(b) False positive. A dog without pain is classified as pain.



(c) True negative. A dog without pain is classified as a dog without pain.



(d) False negative. A dog with pain is classified as a dog without pain.

Figure 34. An overview of the feature contributions for correct and incorrect classifications of the pain and no pain class.

5. Discussion and Conclusion

From our extensive research process on behavior estimation for pain detection in dogs, we can conclude that despite our best efforts, automated behavior detection and pain detection in dogs remain challenging problems. Despite our setback after discovering the faulty results from the previous research, we still maintained that the extracted spatio-temporal features from the ConvLSTM network and the intricate dynamic information from the keypoints processed by the LSTM network fused provide rich information that can improve the performance in comparison to the one-stream models. However, it is significantly less than we initially thought. There is still room for improvement on this part of the model.

Adding the ethogram as a third stream to the previous two-stream model improved the performance of the two-stream model by about 9.77% in terms of accuracy and 12.65% for F1 score, when using a linear layer for the ethogram stream on our best model. Compared to the one-stream model of ethogram, this was a slighter increase in performance of 1.65% in terms of accuracy and 1.45% in terms of F1. This shows that although the performance of the two-stream model is low, the spatio-temporal and dynamic contextual information of the two-stream helps to link the model to what precisely each behavior entails. Our analyses of the results from the three-stream model with a linear layer show that the model generally tends to link behaviors with high activity towards dogs without pain and behaviors with low activity towards dogs in pain. Surprisingly, from our explainability analyses on the linear layer three-stream model, the behavior *decreased weight bearing* did not seem to impact pain prediction much. This seemed counter-intuitive as a dog avoiding putting weight on its limb is a strong indicator of pain.

In contrast, our less-performing three-stream model with a random forest classifier for the ethogram stream seems more intuitive in how it classifies pain or no pain. As seen in the decision tree in Figure 35, *sitting abnormally* and *decreased weight bearing* are the most critical indicators of pain. Although this approach is more intuitive and explainable, there is a considerable trade-off in performance, and people may question if this is desirable in pain prediction.

Behavior estimation for dogs remains challenging, mainly due to the long-tailed distribution of the behavior labels that tend to occur in our videos naturally. We dealt with this problem partly by training a separate classifier on the minority classes within the *standing still* class to simplify the learning problem and allow the model to focus on idle behaviors only. Training the MSQNet model on the separate ethogram dimensions only did not

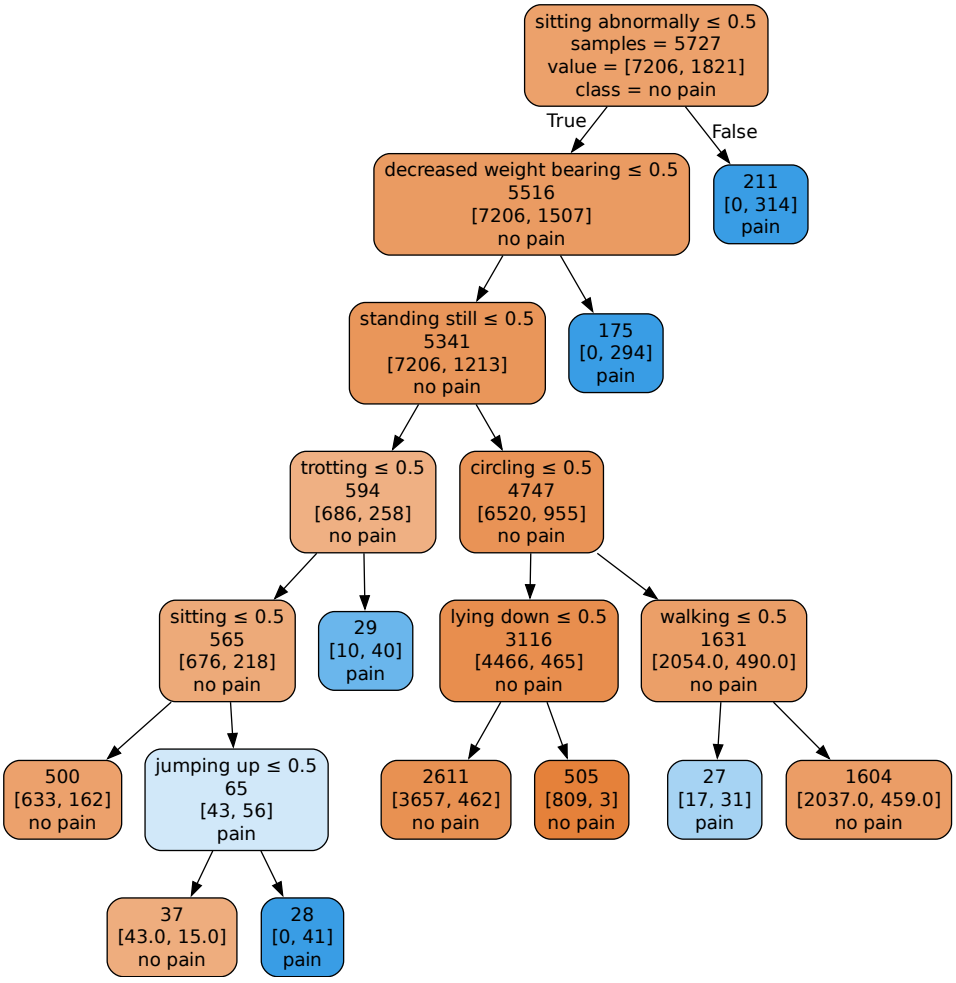


Figure 35. Decision tree trained for detecting the pain status, based on dog behaviors.

produce fruitful results for the minority classes due to the long-tailed data distribution of the behavior labels. Additionally, adding the in-the-wild dog pain dataset did not improve the model's performance. This is most likely due to the vastly different experimental settings of the videos and an abundance of noise in the in-the-wild dataset. Furthermore, we have seen that using prompts to provide valuable contextual information to the spatio-temporal stream of the model shows a significant increase in performance—semantic information matters.

This brings us to the main culprit in our research: data scarcity. During this research, we had to deal with a complex problem to learn while having access to only a small amount of data. These two things are difficult to combine. We think that this model has great potential if there is access to more data on dogs in pain who are exhibiting realistic behaviors unaffected by their owners or other situations, thus reducing bias. Especially if there is more data on dogs experiencing specific pain behaviors, such as *shaking* and *rigid*

posture or *rigid gait*. In the current dataset these were barely to not present, while these behaviors are of great potential in automation in pain detection.

Several suggestions for future work to improve this model are to look into more methods to deal with long-tailed distributions in action recognition specifically. Additionally, to combat the data scarcity problem, the rise of text-to-video and image-to-text generators might be the key in the upcoming year. An example is Luma Dream Machine, as mentioned in the pilot studies in Appendix A. If there is access to greater computing resources, trying to run the DreamBooth (Ruiz et al., 2023) on our current dogs might also be key. To increase the model's understanding of how crucial certain behaviors are in pain estimation, the Melbourne Pain Scale (MPS) as discussed in Section 2.4.5, could be used to give a higher weight to a particular behavior that experts deem to be more indicative of pain.

Overall, adding behaviors to our previous two-stream pain detection model has shown great results and has a lot of potential for improving the welfare of dogs with automated pain detection.

5.1 Contributions

In this thesis we produced the following contributions. First, we proposed a three-stream pain estimation model based on the two-stream pain estimation model from Zhu et al. (2022). In this model, we incorporated an ethogram stream and assessed two approaches: one based on a linear layer and the other using a random forest classifier, respectively.

Then, we provided insight into which behaviors might indicate pain, by adding the pain behaviors to the model, making it more explainable. This makes it easier for veterinarians, who may consult such models to aid in pain assessment to trust the outputs of the model, and to ensure dogs receive the correct medical treatment. We implemented several explainability methods such as Naive Bayes probabilities, Shapley values, Grad-CAM saliency maps, and decision trees.

Additionally, we annotated numerous dog videos from the OFT, the in-the-wild, and the pain-specific dataset, with behaviors selected from the ethogram we used. These dog videos consist of both dogs in pain and without pain. The enrichment of this dataset can be valuable for future research into behavior and pain estimation.

Finally, we re-trained an off-the-shelf model MSQNet (Mondal et al., 2023) using the behavior labels and annotated videos. Subsequently, we used this model to predict the dogs' behaviors in the dataset that were not yet labeled, showing it can be used to speed up

a labor-intensive annotation process.

Appendices

Appendix A: Pilot studies

This section in the appendix will discuss some of the most relevant pilot studies we conducted before determining our best approach.

First, we tried the DeepAction program created by Harris et al. (2023). In this program, we annotated all the videos from which the annotations are currently used in our best approach. Unfortunately, this model was not great at dealing with unbalanced dataset. The paper also mentioned this pitfall, but our dataset appeared more heavily unbalanced than expected, as shown in Figure 19. The accuracy we obtained from running DeepAction was only 50%, and the model only predicted one of the majority classes: either *standing still* or *walking*. Initially we wanted to implement a pre-trained neural network from PyTorch into the DeepAction model such as ActionCLIP (Wang et al., 2021). However, Matlab did not provide enough modern functionality to properly implement custom pre-trained neural networks from outside MatLab, so we could not implement this idea.

Since our dataset is small, we considered using a simple classifier such as KNN or SVM to classify our behaviors. Namely, because the dataset is fairly small, which makes it difficult to train complex models on it. To generate features from our videos, we used a similar approach often used by other prompt-based classification networks, as discussed in Section 2.2.6. For our behavior classes, we used the same behavior prompts as used in Table 5 and extracted CLIP text embeddings from these with a CLIP-ViT-L/14 (Radford et al., 2021) model pre-trained on the LAION-2B English subset of LAION-5B (Schuhmann et al., 2022). For each frame of the dog video, frame embeddings were extracted with the same CLIP model. These features were used by concatenation. Subsequently a simple classifier, SVM or KNN, was fitted on the training data. After this, we tested the model on one dog. The test results were not too great, namely, an accuracy of 27.7% and a F1-score of 34.14%. To try and improve the results of this we opted for using a pre-trained model specifically designed to perform action recognition, namely ActionCLIP (Wang et al., 2021). Unfortunately, this did not yield good results either, namely an accuracy of 13.81% and a F1-score of 4.67%. This made us conclude that our learning problem is too complex for a simple classifier, but the dataset might be too small for a more complex model.

To combat the data scarcity problem, we initially wanted to use the DreamBooth algorithm by Ruiz et al. (2023). This algorithm can generate new images of a certain existing object by linking the object to a special token in its training phase. We wanted to explore whether we could use the dogs from our pain dataset in this model to generate new frames where the dogs exhibit minority behaviors, such that our MSQNet model could learn better if there

(a) Original frame given to Luma Dream Machine

(b) Newly generated frame of dog walking towards owner.

(c) In this frame the hind of the dog has turned into its head and its head into its hind. It did a reverse in representation.

(d) An accurately generated frame of the original dog in a standing still position

Figure 36. A video clip generated with Luma Dream Machine with the frame shown in *a* and the prompt "an old golden retriever is walking slowly through the room"

was more data on these minority classes. Unfortunately, we did not have the computing resources to run tests with these models. Unfortunately, Our graphics card could not even run the model with a batch size of 1.

Recently, Luma AI introduced a new image-to-video generator called Luma Dream Machine (Labs, 2024). Unfortunately, not much is known about the functionality of this model, but one can generate videos on their server by providing an image and a prompt. We tested this generator by providing an image and a specifically tailored prompt for a dog. Some example frames are shown in Figure 36. In this video clip the dog is not always anatomically correct, as its hind and head suddenly completely reversed from location. Although this is hard to show in pictures, this transition can be seen between Figures 36b and c. While this model certainly does have potential, it is still too noisy to represent our minority behaviors accurately.

Bibliography

Amir, S., Gandelsman, Y., Bagon, S., and Dekel, T. (2021). Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2(3):4.

Amit, Y., Felzenszwalb, P., and Girshick, R. (2021). Object detection. In *Computer Vision: A Reference Guide*, pages 875–883. Springer.

Andersen, P. H., Broomé, S., Rashid, M., Lundblad, J., Ask, K., Li, Z., Hernlund, E., Rhodin, M., and Kjellström, H. (2021). Towards machine recognition of facial expressions of pain in horses. *Animals*, 11(6):1643.

Andersen, P. H., Gleerup, K., Wathan, J., Coles, B., Kjellström, H., Broomé, S., Lee, Y., Rashid, M., Sonder, C., Resenberg, E., et al. (2018). Can a machine learn to see horse pain? an interdisciplinary approach towards automated decoding of facial expressions of pain in the horse. *Proc Meas Behav*, pages 6–8.

Anderson, D. J. and Adolphs, R. (2014). A framework for studying emotions across species. *Cell*, 157(1):187–200.

Andresen, N., Wöllhaf, M., Hohlbaum, K., Lewejohann, L., Hellwich, O., Thöne-Reineke, C., and Belik, V. (2020). Towards a fully automated surveillance of well-being status in laboratory mice using deep learning: Starting with facial expression analysis. *PLoS One*, 15(4):e0228059.

Auer, U., Kelemen, Z., Engl, V., and Jenner, F. (2021). Activity time budgets—a potential tool to monitor equine welfare? *Animals*, 11(3):850.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

Bell, A., Helm, J., and Reid, J. (2014). Veterinarians' attitudes to chronic pain in dogs. *Veterinary Record*, 175(17):428–428.

Bohnslav, J. P., Wimalasena, N. K., Clausing, K. J., Dai, Y. Y., Yarmolinsky, D. A., Cruz, T., Kashlan, A. D., Chiappe, M. E., Orefice, L. L., Woolf, C. J., et al. (2021). Deepethogram, a machine learning pipeline for supervised behavior classification from raw pixels. *Elife*, 10:e63377.

Boneh-Shitrit, T., Amir, S., Bremhorst, A., Mills, D. S., Riemer, S., Fried, D., and Zamansky, A. (2022a). Deep learning models for automated classification of dog emotional states from facial expressions. *arXiv preprint arXiv:2206.05619*.

Boneh-Shitrit, T., Amir, S., Bremhorst, A., Riemer, S., Wurbel, H., Mills, D., and Zamansky, A. (2022b). Deep learning models for classification of canine emotional states. In *Comput. Vis. Pattern Recognit*.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

Broomé, S., Ask, K., Rashid-Engström, M., Haubro Andersen, P., and Kjellström, H. (2022). Sharing pain: Using pain domain transfer for video recognition of low grade orthopedic pain in horses. *PloS one*, 17(3):e0263854.

Broome, S., Feighelstein, M., Zamansky, A., Carreira Lencioni, G., Haubro Andersen, P., Pessanha, F., Mahmoud, M., Kjellström, H., and Salah, A. A. (2023). Going deeper than tracking: A survey of computer-vision based recognition of animal pain and emotions. *International Journal of Computer Vision*, 131(2):572–590.

Broomé, S., Gleerup, K. B., Andersen, P. H., and Kjellstrom, H. (2019). Dynamics are important for the recognition of equine pain in video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12667–12676.

Caeiro, C. C., Burrows, A. M., and Waller, B. M. (2017). Development and application of catfacs: Are human cat adopters influenced by cat facial expressions? *Applied Animal Behaviour Science*, 189:66–78.

Cao, J., Tang, H., Fang, H.-S., Shen, X., Lu, C., and Tai, Y.-W. (2019). Cross-domain adaptation for animal pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9498–9507.

Chen, Y. and Joo, J. (2021). Understanding and mitigating annotation bias in facial expression recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14980–14991.

Dalla Costa, E., Minero, M., Lebelt, D., Stucke, D., Canali, E., and Leach, M. C. (2014). Development of the horse grimace scale (hgs) as a pain assessment tool in horses undergoing routine castration. *PLoS one*, 9(3):e92281.

Das, A. and Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*.

Diogo, R., Abdala, V., Lonergan, N., and Wood, B. (2008). From fish to modern humans–comparative anatomy, homologies and evolution of the head and neck musculature. *Journal of Anatomy*, 213(4):391–424.

DiPietro, R. and Hager, G. D. (2020). Deep learning: Rnns and lstm. In *Handbook of medical image computing and computer assisted intervention*, pages 503–519. Elsevier.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Ebrahimi Kahou, S., Michalski, V., Konda, K., Memisevic, R., and Pal, C. (2015). Recurrent neural networks for emotion recognition in video. In *Proceedings of the 2015 ACM on international conference on multimodal interaction*, pages 467–474.

Ekman, P. (1973). Universal facial expressions in emotion. *Studia Psychologica*, 15(2):140–147.

Ekman, P. (1992). An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Ekman, P. and Friesen, W. V. (1978). Facial action coding system. *Environmental Psychology & Nonverbal Behavior*.

Evangelista, M. C., Watanabe, R., Leung, V. S., Monteiro, B. P., O'Toole, E., Pang, D. S., and Steagall, P. V. (2019). Facial expressions of pain in cats: the development and validation of a feline grimace scale. *Scientific reports*, 9(1):19128.

Farhat, N., van der Linden, D., Zamansky, A., and Assif, T. (2024). Automation in canine science: enhancing human capabilities and overcoming adoption barriers. *Frontiers in Veterinary Science*, 11:1394620.

Feighelstein, M., Ehrlich, Y., Naftaly, L., Alpin, M., Nadir, S., Shimshoni, I., Pinho, R. H., Luna, S. P., and Zamansky, A. (2023a). Deep learning for video-based automated pain recognition in rabbits. *Scientific Reports*, 13(1):14679.

Feighelstein, M., Henze, L., Meller, S., Shimshoni, I., Hermoni, B., Berko, M., Twele, F., Schütter, A., Dorn, N., Kästner, S., et al. (2023b). Explainable automated pain recognition in cats. *Scientific reports*, 13(1):8973.

Feighelstein, M., Shimshoni, I., Finka, L. R., Luna, S. P., Mills, D. S., and Zamansky, A. (2022). Automated recognition of pain in cats. *Scientific Reports*, 12(1):9575.

Ferres, K., Schloesser, T., and Gloor, P. A. (2022). Predicting dog emotions based on posture analysis using deeplabcut. *Future Internet*, 14(4).

Finka, L. R., Luna, S. P., Brondani, J. T., Tzimiropoulos, Y., McDonagh, J., Farnworth, M. J., Ruta, M., and Mills, D. S. (2019). Geometric morphometrics for the study of facial expressions in non-human animals, using the domestic cat as an exemplar. *Scientific reports*, 9(1):9883.

Franzoni, V., Milani, A., Biondi, G., and Micheli, F. (2019). A preliminary work on dog emotion recognition. In *IEEE/WIC/ACM International Conference on Web Intelligence-Companion Volume*, pages 91–96.

Frénay, B. and Verleysen, M. (2013). Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407.

Gao, P., Geng, S., Zhang, R., Ma, T., Fang, R., Zhang, Y., Li, H., and Qiao, Y. (2024). Clip-adapter: Better vision-language models with feature adapters. *International Journal of Computer Vision*, 132(2):581–595.

Gleerup, K. B., Forkman, B., Lindegaard, C., and Andersen, P. H. (2015). An equine pain face. *Veterinary anaesthesia and analgesia*, 42(1):103–114.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Graving, J. M., Chae, D., Naik, H., Li, L., Koger, B., Costelloe, B. R., and Couzin, I. D. (2019). Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning. *Elife*, 8:e47994.

Hale, C. J., Hadjistavropoulos, T., et al. (1997). Emotional components of pain. *Pain Research and Management*, 2:217–225.

Hansen, B. D. (2003). Assessment of pain in dogs: veterinary clinical studies. *ILAR journal*, 44(3):197–205.

Harris, C., Finn, K. R., Kieseler, M.-L., Maechler, M. R., and Tse, P. U. (2023). Deepaction: a matlab toolbox for automated classification of animal behavior in video. *Scientific Reports*, 13(1):2688.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hernandez-Avalos, I., Mota-Rojas, D., Mora-Medina, P., Martínez-Burnes, J., Casas Alvarado, A., Verduzco-Mendoza, A., Lezama-García, K., and Olmos-Hernandez, A. (2019). Review of different methods used for clinical recognition and assessment of pain in dogs and cats. *International journal of veterinary science and medicine*, 7(1):43–54.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hummel, H. I., Pessanha, F., Salah, A. A., van Loon, T. J., and Veltkamp, R. C. (2020). Automatic pain detection on horse and donkey faces. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pages 793–800. IEEE.

Infernuso, T., Loughin, C. A., Marino, D. J., Umbaugh, S. E., and Solt, P. S. (2010). Thermal imaging of normal and cranial cruciate ligament-deficient stifles in dogs. *Veterinary surgery*, 39(4):410–417.

Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., and Bengio, Y. (2017). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 11–19.

Jiang, L., Lee, C., Teotia, D., and Ostadabbas, S. (2022). Animal pose estimation: A closer look at the state-of-the-art, existing gaps and opportunities. *Computer Vision and Image Understanding*, page 103483.

Jiang, Z.-H., Hou, Q., Yuan, L., Zhou, D., Shi, Y., Jin, X., Wang, A., and Feng, J. (2021). All tokens matter: Token labeling for training better vision transformers. *Advances in neural information processing systems*, 34:18590–18602.

Jocher, G. (2020). Yolov5 by ultralytics.

Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics yolov8.

Kabra, M., Robie, A. A., Rivera-Alba, M., Branson, S., and Branson, K. (2013). Jaaba: interactive machine learning for automatic annotation of animal behavior. *Nature methods*, 10(1):64–67.

Kaya, H. and Salah, A. A. (2018). Multimodal personality trait analysis for explainable modeling of job interview decisions. *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 255–275.

Kim, J. and Moon, N. (2024). Enhanced pet behavior prediction via s2gan-based heterogeneous data synthesis. *Applied Sciences*, 14(10):4091.

Kim, K., Gowda, S. N., Mac Aodha, O., and Sevilla-Lara, L. (2022). Capturing temporal information in a single frame: Channel sampling strategies for action recognition. *arXiv preprint arXiv:2201.10394*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Kuhn, H. W. and Tucker, A. W. (1953). *Contributions to the Theory of Games*. Number 28. Princeton University Press.

Labs, L. (2024). Dream machine. `https://lumalabs.ai/dream-machine`. Accessed: 2024-07-29.

Langford, D. J., Bailey, A. L., Chanda, M. L., Clarke, S. E., Drummond, T. E., Echols, S., Glick, S., Ingrao, J., Klassen-Ross, T., LaCroix-Fralish, M. L., et al. (2010). Coding of facial expressions of pain in the laboratory mouse. *Nature methods*, 7(6):447–449.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110.

Lu, Y., Mahmoud, M., and Robinson, P. (2017). Estimating sheep pain level using facial action unit detection. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 394–399. IEEE.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., and Bethge, M. (2018). Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*.

Mills, D. S., Demontigny-Bédard, I., Gruen, M., Klinck, M. P., McPeake, K. J., Barcelos, A. M., Hewison, L., Van Haevermaet, H., Denenberg, S., Hauser, H., et al. (2020). Pain and problem behavior in cats and dogs. *Animals*, 10(2):318.

Molnar, C. (2022). *Interpretable Machine Learning*. 2 edition.

Mondal, A., Nag, S., Prada, J. M., Zhu, X., and Dutta, A. (2023). Actor-agnostic multi-label action recognition with multi-modal query. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 784–794.

Morozov, A., Parr, L. A., Gothard, K., Paz, R., and Pryluk, R. (2021). Automatic recognition of macaque facial expressions for detection of affective states. *Eneuro*, 8(6).

Ng, X. L., Ong, K. E., Zheng, Q., Ni, Y., Yeo, S. Y., and Liu, J. (2022). Animal kingdom: A large and diverse dataset for animal behavior understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19023–19034.

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987.

Panksepp, J. (2010). Affective neuroscience of the emotional brainmind: evolutionary perspectives and implications for understanding depression. *Dialogues in clinical neuroscience*.

Pereira, T. D., Aldarondo, D. E., Willmore, L., Kislin, M., Wang, S. S.-H., Murthy, M., and Shaevitz, J. W. (2019). Fast animal pose estimation using deep neural networks. *Nature methods*, 16(1):117–125.

Pereira, T. D., Tabris, N., Matsliah, A., Turner, D. M., Li, J., Ravindranath, S., Papadoyannis, E. S., Normand, E., Deutsch, D. S., Wang, Z. Y., et al. (2022). Sleap: A deep learning system for multi-animal pose tracking. *Nature methods*, 19(4):486–495.

Piergiovanni, A. and Ryoo, M. (2019). Temporal gaussian mixture layer for videos. In *International Conference on Machine learning*, pages 5152–5161. PMLR.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Raja, S. N., Carr, D. B., Cohen, M., Finnerup, N. B., Flor, H., Gibson, S., Keefe, F. J., Mogil, J. S., Ringkamp, M., Sluka, K. A., et al. (2020). The revised international association for the study of pain definition of pain: concepts, challenges, and compromises. *Pain*, 161(9):1976–1982.

Rashid, M., Broomé, S., Ask, K., Hernlund, E., Andersen, P. H., Kjellström, H., and Lee, Y. J. (2022). Equine pain behavior classification via self-supervised disentangled pose representation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1646–1656.

Rohdin, C., Jäderlund, K. H., Ljungvall, I., Lindblad-Toh, K., and Häggström, J. (2018). High prevalence of gait abnormalities in pugs. *Veterinary Record*, 182(6):167–167.

Rozemberczki, B., Watson, L., Bayer, P., Yang, H.-T., Kiss, O., Nilsson, S., and Sarkar, R. (2022). The shapley value in machine learning. *arXiv preprint arXiv:2202.05594*.

Ruhof, J. J., Salah, A. A., and van Loon, T. J. P. (2024). Automatic pain estimation in equine faces: More effective uses for regions of interest. In *12th International Conference on Affective Computing and Intelligent Interaction, ACII 2024 - Workshops and Demos*, Glasgow, UK. Dept. Information and Computing Sciences, Utrecht University, IEEE. j.j.ruhof@students.uu.nl, a.a.salah@uu.nl.

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. (2023). Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252.

Russell, J. A. (1980). A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161.

Saabas, A. (2016). treeinterpreter: Interpreting scikit-learn's decision tree and random forest predictions. `https://github.com/andosa/treeinterpreter`.

Salgırlı, Y. (2012). Evaluation of body postures of belgian malinois dogs during a police dog training in germany. *Veteriner Fakultesi dergisi*, 59:241–246.

Salgırlı, Y. (2020). How to approach behavioural problems in dogs. *Journal of Istanbul Veterinary Sciences*, pages 82–82.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. (2022). Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. (2022). Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Steagall, P., Monteiro, B., Marangoni, S., Moussa, M., and Sautié, M. (2023). Fully automated deep learning models with smartphone applicability for prediction of pain using the feline grimace scale. *Scientific Reports*, 13(1):21584.

Sun, K., Xiao, B., Liu, D., and Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5693–5703.

Sundararajan, M. and Najmi, A. (2020). The many shapley values for model explanation. In *International conference on machine learning*, pages 9269–9278. PMLR.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Turk, M. A. and Pentland, A. P. (1991). Face recognition using eigenfaces. In *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pages 586–587. IEEE Computer Society.

van Loon, J. P. and Van Dierendonck, M. C. (2015). Monitoring acute equine visceral pain with the equine utrecht university scale for composite pain assessment (equus-compass) and the equine utrecht university scale for facial assessment of pain (equus-fap): a scale-construction study. *The Veterinary Journal*, 206(3):356–364.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Verma, G. K. and Tiwary, U. S. (2017). Affect representation and recognition in 3d continuous valence–arousal–dominance space. *Multimedia Tools and Applications*, 76:2159–2183.

Waller, B. M., Peirce, K., Caeiro, C. C., Scheider, L., Burrows, A. M., McCune, S., and Kaminski, J. (2013). Paedomorphic facial expressions give dogs a selective advantage. *PLoS one*, 8(12):e82686.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al. (2020). Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364.

Wang, M., Xing, J., and Liu, Y. (2021). Actionclip: A new paradigm for video action recognition. *arXiv preprint arXiv:2109.08472*.

Wang, Y., Huang, M., Zhu, X., and Zhao, L. (2016). Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Wathan, J., Burrows, A. M., Waller, B. M., and McComb, K. (2015). Equifacs: The equine facial action coding system. *PLoS one*, 10(8):e0131738.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Zamzmi, G., Kasturi, R., Goldgof, D., Zhi, R., Ashmeade, T., and Sun, Y. (2017). A review of automated pain assessment in infants: features, classification tasks, and databases. *IEEE reviews in biomedical engineering*, 11:77–96.

Zhu, H., Salgırlı, Y., Can, P., Atılgan, D., and Salah, A. A. (2022). Video-based estimation of pain indicators in dogs. *arXiv preprint arXiv:2209.13296*.

Zhu, Y., Lan, Z., Newsam, S., and Hauptmann, A. (2019). Hidden two-stream convolutional networks for action recognition. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 363–378. Springer.