UNIVERSITY OF SUSSEX

FINAL YEAR DISSERTATION

AUTOMATIC DETECTION OF MOVEMENT-RELATED PAIN AND ITS RELATED EXPRESSIONS FOR PEOPLE WITH CHRONIC PAIN.

AUTHOR:

JOVITO THOMAS – 260878

SUPERVISOR:

DR. TEMITAYO OLUGBADE

Table of Contents

1. ABSTRACT

Chronic pain is a significant public health problem, affecting millions of people worldwide. Chronic pain induces pain for people while performing everyday physical activities at home and can significantly impact a person's physical and emotional well-being. The Pain related expressions, Worry and confidence are also important factors in the lives of people with chronic pain, as they can influence how they cope with their pain and how they participate in activities of daily life.[1] (Johns Hopkins Medicine, n.d.)

This study investigated the automatic detection of movement-related pain and its expressions in people with chronic pain using body movement data obtained from the Novel EmoPain@home dataset. Automatic detection of pain, worry and confidence for people with chronic pain would help play a part in pain management and real-time emotional state evaluation which would improve their quality of life. In this study, two experiments were conducted to compare the performance of two different types of inputs to different machine learning algorithms for the automatic detection of pain and its related expressions. The two different inputs are features extracted from the body movement data and the raw sensor data from the EmoPain@Home dataset.

Further in this study, a third experiment is also conducted to build a multi-task learning (MTL) model using the best Input type from the comparison of the first two experiments to perform all three tasks of automatic detection of pain, worry, and confidence intensities simultaneously. The results of the multi-task learning model will then be compared with the single task learning (STL) models to understand which method gives the best performance for all three tasks.

2. INTRODUCTION

Chronic pain is a pervasive and challenging health issue that affects millions of individuals worldwide. It is defined as pain that persists for more than three to six months, often beyond the expected healing time of an injury or underlying condition. [2] (Cleveland clinics, 2021) Chronic pain can have a profound impact on a person's physical, emotional, and social well-being, leading to functional limitations, reduced quality of life, and increased healthcare costs. The prevalence of chronic pain is substantial, with estimates suggesting that it affects approximately 20% of the global population. Common conditions associated with chronic pain include musculoskeletal disorders, neuropathic pain, migraine, and fibromyalgia. However, chronic pain can manifest in various forms and can arise from a wide range of medical conditions or injuries.

Assessing pain-related expressions, including movement confidence and worry levels, is a critical aspect of healthcare, particularly for individuals experiencing chronic pain.

The motivation for looking at automatic detection of pain and its related worry and confidence during everyday physical activity at home is that it could provide a way to assess pain in a more objective and continuous way than traditional methods, such as self-report questionnaires. This could be especially beneficial for people, who may find it difficult to accurately report their pain levels on a regular basis.

Automatic detection of pain could also be used to provide real-time feedback to people with chronic pain about their pain levels and their emotional state. This feedback could help people to better manage their pain and to cope with the emotional and psychological challenges of chronic pain.

Traditionally, self-reported pain scales have been the primary method for pain assessment, relying on individuals' subjective perception and communication of their pain experience.[2] (Cleveland clinics, 2021) However, these scales are inherently limited by factors such as language barriers, leading to potential biases and inaccuracies.

The reason for using Body movement as the modality for this study is because it captures the physical changes that occur in the body when someone is in pain. For example, people with chronic pain may walk with a limp or avoid certain movements. [3]

Movement related confidence can be inferred from body movement by looking at factors such as speed, smoothness, and range of motion. For example, people who are confident may move more quickly and smoothly, and they may have a wider range of motion.

similarly, Movement related Worry can be inferred from body movement by looking at factors such as tension, rigidity, and hesitation. For example, people who are worried may move more slowly and hesitantly, and they may have more tension in their muscles.

Furthermore, this modality can be easily captured using sensors, such as accelerometers and gyroscopes, which can be worn on the body.

Further in this study I compare the performance difference in using two different input types, the manually handcrafted features from the body movement data and the raw sensor data for automatic detection of movement related pain, worry and confidence.

This comparison was done because the features that can be handcrafted are limited by the knowledge that humans have about the domain. This means that manually handcrafted features may not be able to capture all of the relevant information from the data. Manually extracting features from data is also highly time consuming when compared to using ML algorithms to extract complex features.

Another reason for the comparison is that raw sensor data is not influenced by human biases or interpretations. This means that raw sensor data can be more objective than manually handcrafted features, it can also be used to train a wider variety of machine learning models than manually handcrafted features. This means that it can be more versatile than manually handcrafted features.

The results of the comparison of the two input types will be used to determine which type of input is best for use in building a multi-task learning model that can automatically detect pain, worry, and confidence. The Multi-task learning model will then be compared with the single-task learning models to compare the performance difference.

This study aims to answer the following questions: -

RQ1: Does the features extracted by a machine learning algorithm provide a significant performance improvement over manually extracted features when used to train a Machine learning model?

RQ2: Does using a multi-task learning model improve the performance of pain, worry, and confidence detection compared to using a single-task learning model?

This study makes a novel contribution to the field of automatic detection of movement-related pain, worry, and confidence by using the Novel EmoPain@Home dataset which is a fairly recent dataset introduced in 2022 containing self-reports for different intensities of movement-related pain, worry, and confidence intensity. Additionally, this study explores a multi-task learning approach for learning and predicting the three tasks simultaneously.

The multi-task learning approach used in this study could have the potential to improve the accuracy and performance of automatic detection of movement-related pain, worry, and confidence. The results of this study could help to develop more effective tools for pain management and chronic pain rehabilitation.

The related works mentioned in the coming section influenced the research path that was used in this study.

3. RELATED WORKS

There are a few works related to the expressions of pain in people with chronic pain, but very few studies have been conducted on the EmoPain@home dataset. Previous works have used various methods to study pain expression, including facial expressions, verbalizations, and physiological measures.[4] However, there are very few works related to the research on the use of body movement data to study pain-related expressions in people with chronic pain.

***A**. Automatic Assessment of Movement Related Self-Efficacy in Chronic Pain: From Exercise to Functional Activity.*

One study that investigated the use of body movement data to study pain-related expression (MRSE – Movement Related Self Efficacy) in people with chronic pain was conducted by Olugbade et al. (2018) [5].

The authors first explored clinical observer estimation which is the process of assessing a patient's condition or status based on their physical appearance, behaviour, and other nonverbal cues, this showed that body movement behaviours were more pertinent to MRSE estimation than facial expressions or engagement behaviours (Non-verbal cues). Based on their findings, they built a system that estimates MRSE from bodily expressions and bodily muscle activity captured using wearable sensors. The authors' results (F1 scores of 0.95 and 0.78 in two physical exercise types) provide evidence of the feasibility of automatic MRSE estimation to support chronic pain physical rehabilitation. This study is relevant to my research since MRSE is closely related to Movement Related Confidence as mentioned in the study. [5]

The Dataset used by the study is the EmoPain corpus, which consists of video, inertia-based motion capture, and sEMG muscle activity data of people with chronic low back pain and healthy control participants. The corpus does not provide MRSE labels, so the authors conducted an annotation study to enable the corpus to build an automatic MRSE detection system. The EmoPain@Home dataset that I'm using for my study contains self-reported Movement Related confidence.

They used 421 video clips for the annotation study, 48.2% of which were of people with chronic pain and 51.8% of which were of healthy control participants. Each clip showed a person performing one of three activities: sit-to-stand, forward trunk flexion, or full trunk flexion. The activities were performed at two difficulty levels, with the more complex level being more challenging for people with chronic low back pain.

The authors recruited two independent raters to annotate the video clips with MRSE ratings. The raters were blinded to the identity of the participants and the difficulty level of the activities. The inter-rater reliability of the ratings was high, with an intraclass correlation coefficient (ICC) of 0.95.

The authors found that the MRSE ratings were significantly correlated with the difficulty level of the activities, with higher ratings for the more challenging activities. The authors concluded that the EmoPain corpus can be used to build an automatic MRSE detection system and that the annotation study provides a valuable resource for future research in this area.

The authors further explored the automatic estimation of MRSE with a reduced set of low-cost sensors to investigate the possibility of embedding such capabilities in ubiquitous wearable devices to support functional activity. Their evaluation for both exercise and functional activity resulted in an F1 score of 0.79. This result suggests the possibility of (and calls for more studies on) MRSE estimation during everyday functioning in ubiquitous settings, which is relevant in the area of my work where everyday activities are recorded for chronic pain patients. The authors also discussed the implications of their findings for relevant areas, such as the development of personalized rehabilitation interventions and the design of wearable devices for people with chronic pain.

There are a few limitations of the study that the authors mentioned. These limitations include the small sample size, the fact that the participants were all from the same ethnic group, and the use of only two types of physical exercise. The authors suggest that future studies should address these limitations by recruiting a more extensive and more diverse sample of participants, and by using a wider variety of physical exercises. These limitations were addressed in future studies mentioned in this section.

The findings of the study by Olugbade et al. (2018) [5] suggest that body movement data can be used to study pain-related expression and MRSE which is closely related to Movement-related Confidence.

B. *How Can Affect Be Detected and Represented in Technological Support for Physical Rehabilitation?*

Another study conducted by Olugbade et al. (2020) [6] investigated the use of movement behaviour to automatically detect affective states in people with chronic pain. Their study is significant because it provides evidence that movement behaviour can be used to discriminate between different levels of pain and emotional distress. This study formed the base for my work on using body movement data for developing predictive models for different levels of pain and emotional distress. The authors used two standard machine learning classification algorithms, support vector machines (SVMs) and random forests (RFs), to investigate the feasibility of pain level classification based on the EmoPain dataset.

They evaluated the classification performance using leave-one-subject-out cross-validation and reported the results using standard metrics: accuracy, F1 score, and confusion matrices.

The authors concluded that the results of their study suggest that machine learning can be used to classify pain levels from body movement data.

I plan to use the findings of the study by Olugbade et al. (2020) [6] to inform the development of my system. For example, I will use the findings from the study to identify movement behaviour features that are relevant to pain and emotional distress detection. These features will be extracted from joint position data and will be used for developing models for detecting movement-related, pain, worry, and confidence. I will also use the findings from the study to evaluate the performance of my system.

C. *EmoPain(at)Home: Dataset and Automatic Assessment within Functional Activity for Chronic Pain Rehabilitation.*

Another study by Olugbade et al. (2022) [7] introduced the EmoPain(at)Home dataset, which consists of motion capture data and self-reported pain, worry, and confidence intensities captured from people with chronic pain. The data was recorded during self-selected functional activities in the home, such as vacuuming, washing dishes etc. The authors included an analysis of the dataset as well as a baseline classification of pain levels with an average F1 score of 0.61 for two classes. They also discussed inclusivity considerations for the capture of datasets in naturalistic settings, based on lessons learnt within their study.

The authors compared the EmoPain@Home dataset to other existing datasets of pain expression modalities. They found that the EmoPain@Home dataset is unique in that it captures body movement data in natural activities in people's homes. This is in contrast to other datasets, which typically capture pain expression data in constrained or instructed movements in artificial settings. The authors mention that the EmoPain@Home dataset is more realistic and representative of pain experience in everyday life. They also argue that the dataset is more challenging to classify, as participants may use adaptive strategies to cope with pain in their home environment.

The authors also discussed lessons learned around inclusivity during their data collection process. They emphasized the importance of capturing and understanding issues around the acquisition of sensor data and related self-reports and discussed how this can help shape the design of technology for people with chronic pain. Finally, the authors explored automatic detection based on the EmoPain@Home dataset, with analysis of features of pain, worry, and confidence captured from different movement timescales, as well as baseline classification of pain levels. The work of Olugbade et al. (2022) [7], who introduced the EmoPain@Home dataset is a valuable resource for researchers who are interested in

I plan to use the EmoPain@Home dataset to develop a system for automatically predicting pain, worry, and confidence levels in people with chronic pain. I will use a variety of machine learning techniques to extract features from the motion capture data and self-reported pain, worry, and confidence ratings in the dataset.

D. Comparing Handcrafted Features and Deep Neural Representations for Domain Generalization in Human Activity Recognition

In a study by Bento et al. (2022) [8] The authors compared handcrafted features and deep neural representations for domain generalization in HAR. They used homogenized public datasets to compare the two approaches in multiple domains. They first compared several metrics to validate three different Out-of-Distribution (OOD) settings. In their main experiments, they verified that even though deep learning initially outperforms models with handcrafted features, the situation is reversed as the distance from the training distribution increases. The authors used 1-dimensional CNN for our deep learning baselines to extract features from the input data.

The authors concluded that handcrafted features generalize better across specific domains than deep neural representations. This is because handcrafted features are designed to capture the specific characteristics of a given domain, while deep neural representations are more general and may not be able to capture the nuances of a specific domain.

This study provided a base for comparing two different input types and the selection of CNN models for their feature extraction abilities.

E. Multitask LSTM Model for Human Activity Recognition and Intensity Estimation Using Wearable Sensor Data.

In research by Barut et al. (2020) [9], the authors propose a multitask long short-term memory (LSTM) model that can be used to perform human activity recognition (HAR) and intensity estimation simultaneously.

The authors train their model on a new dataset that they had introduced, which includes both activity types and activity intensities for 10 adults performing 7 different activities. They evaluate their proposed multi-task learning model on several publicly available datasets and show that it can achieve comparable performance to two separate single-task models. They also show that including activity intensities in the data set can help multitask models perform better.

This work is significant to my study because it gives valuable insights into using data collected from humans using wearable sensors to develop a multi-task learning model that can learn different tasks simultaneously. The study also mentions that it can achieve comparable performance with single-task learning models, which I will be investigating in my work. Furthermore, I will be using joint position data obtained from wearable sensors to train a multi-task model that can be used for learning 3 different tasks simultaneously. (Pain, worry, and confidence level estimation)

4. METHODS

4.1 DATASET:

The EmoPain@Home dataset introduced by Olugbade et al. (2022) [7], is a collection of data on chronic pain. It was created by recruiting 9 people with chronic musculoskeletal pain involving the lower back area. The participants were between 27 and 59 years old and were from the UK. This would account for the diversity in the age group of people with chronic pain for this study.

Data was captured in the context of everyday physical functioning at home. The participants performed tasks that they needed to do in their own homes, such as cooking, cleaning, and gardening. They were also asked to rate their pain, worry, and confidence at one-minute intervals throughout each activity.

The participants with chronic pain performed between 2 and 17 instances of activities across multiple days. The median number of activity instances was 6. This means that half of the participants performed 6 or fewer instances of the activity, and half performed 7 or more instances.

The dataset includes two types of data:

body movement data and self-reported pain and affect labels.

Body movement data:

This data was captured using wearable sensors that recorded the three-dimensional angle and position of the participant's joints. This data can be used to track the participant's movements and identify patterns that may be associated with pain.

I will be using the joint positions data from this dataset for people with chronic pain. This contains the position data collected from all 6 joints for the 9 participants. The 6 joints where the sensor data was captured are the Chest Bottom, Hip, Right Forearm, Right Lower Leg, Right Thigh, and, Right Upper Arm.

The data was collected at a 40Hz sampling rate.

The skeletal representation of the placement of the sensors is shown in Fig 1.

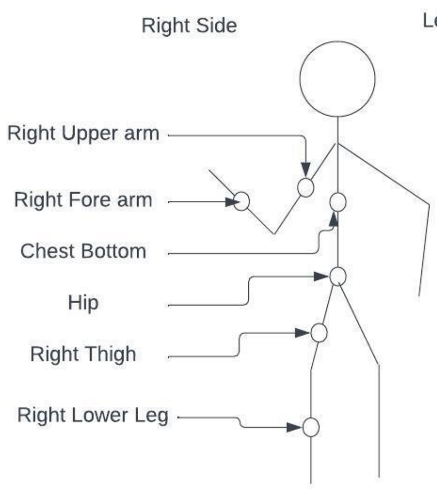


Fig 1: Skeletal representation of the placement of sensors in the EmoPain@Home dataset.

Self-reported pain and pain-related expressions labels:

The researchers asked participants with chronic pain to rate their pain, worry, and confidence levels every minute during physical activity. Pain and worry were rated on a scale of 0 to 10, while confidence was rated on a scale of no confidence to max confidence. The researchers chose to use a simpler scale for confidence because people do not make as many distinctions between levels of confidence as they do for pain and worry. Additionally, using a verbal scale for confidence made it easier for participants to understand and respond to the questions. This gave a simpler scale for confidence intensity in my study for developing the predictive models.

The labels dataset contains the Participant IDs, the activity instances, the Challenging flag (the values would be Yes if the participant flagged the activity as challenging and No if the activity wasn't challenging), reported pain, worry, and confidence labels.

The Labels dataset also contains the activity labels, which contain the type of activities performed by each participant.

I plotted the distribution of pain, worry, and confidence intensities along with the distribution of challenging and non-challenging activities (shown in Fig 2.)

The dataset also contains the date and time when the self-report instance was recorded, which is very helpful in matching the joint sensor data with the corresponding self-report instance.

In my dissertation, I will use the EmoPain@Home dataset to predict pain, confidence, and worry intensities. I will use machine learning algorithms to analyse the body movement data and the self-reported pain and affect labels.

The ability to predict pain, confidence, and worry intensities could have several benefits. For example, it could be used to develop early warning systems for people who are at risk of developing chronic pain. It could also be used to track the effectiveness of treatment for chronic pain.

EmoPain@Home dataset could also be used to develop new rehabilitation exercises for people with chronic pain. By analysing the body movement data, researchers could identify patterns that are associated with pain relief. This information could then be used to create exercises that are specifically designed to reduce pain.

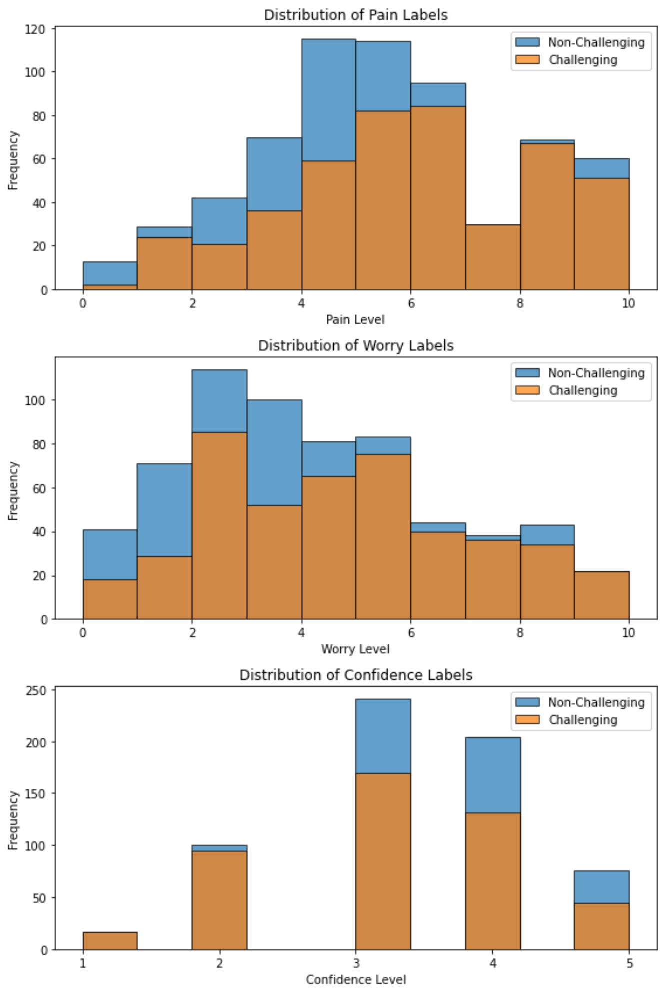


Fig 2: Plots showing the distribution of intensities for the labels (Pain, Worry and Confidence).

4.2 DATA CLEANING:

I first imported the data from both the sensor Data files and the Labels data file. Checked for missing values or NaN values and if any discrepancies were found in the data, I removed the entire instance where the missing data was found since the sensor data is continuous time series data and when there are missing values in a time series, it can affect the accuracy of statistical analysis and machine learning models. This is because the missing values can introduce bias into the data. It can also disrupt the continuity of the data. This is because the missing values create gaps in the data, which can make it difficult to track trends and patterns.

I found out there were certain activity instances where few joint sensors were not recording the data. I decided to remove those activity instances with missing or faulty joint sensor data. This was done such that all 6 joints would be present in the data we use for the feature extractions.

4.3 DATA SEGMENTATION AND WINDOWING:

The metadata file for the sensor dataset contains the start time of the recording of activities, the sampling rate, the total number of frames captured, and the name of the joints whose position data was recorded.

using the formulas given below, I calculated the end time for each activity instance:

Duration of the activity **d** (in seconds) was calculated with the formula below:

$$d = \frac{F}{S}$$

Where **F** is the total Frame count for that activity instance and **S** is the sampling rate for the same.

The Duration of the activity in minutes was calculated with the formula.

$$D = d\, x\, 60$$

After calculating the Duration of the activity, the end time can be easily calculated by adding the Duration and the start time of the activity.

$$E_t = S_t + D$$

Here, E_t is the end time and S_t is the start time for the activity and D is the Duration in minutes.

The first condition for segmenting the sensor data and label data is to check if the participant IDs of both of these data match for an activity instance. Once this condition is satisfied, I checked whether the pain, worry, and confidence intensities were recorded within the start and end times of the activity instance.

The sampling rate for the data capture is 40 Hz, and the labels were recorded at one-minute intervals from the start time of the activity. Therefore, I extracted 2400 frames (60 x 40) for all six joints and matched them with their corresponding pain, worry, and confidence intensity labels for that activity instance.

In summary, the data was segmented and windowed in a report timescale, which is defined as the one-minute window up to the given self-report point.

4.4 FEATURE EXTRACTION:

I computed 6 types of features referencing the studies conducted by Olugbade et al. (2018) [5] and Olugbade et al. (2020).[6]. These sets of features were mentioned as significant for the evaluation of pain and emotional distress levels when using body movement data. The 6 sets of features are average speed, average Jerk, Range of Motion, Minimum distance to the forearm, average acceleration, and average energy. I got a total of 35 features. 30 features were derived for all 6 joints. (Average speed, average Jerk, Range of Motion, average acceleration, and average energy).

5 features were derived for 5 joints (Minimum distance to the forearm) which is the minimum distance of the 5 joints to the forearm.

AVERAGE SPEED:

The Equation for speed is $s = \frac{d}{t}$ where d is the distance and t is the time. In this study, I'm segmenting the data to the time interval of 1 minute up to the self-report point. Hence the value of time is 1.

I first calculated the velocity for each coordinate (x, y, z) of each joint. Velocity is calculated by taking the difference between consecutive values. For example, to calculate the velocity in the x-direction for the Chest Bottom joint, the code would take the difference between the x-coordinates of the current timestep and the previous timestep.

Then I calculated the speed magnitude for each joint by taking the square root of the sum of the squares of the average speed values for each coordinate. For example, to calculate the speed magnitude for the Chest Bottom joint, the code would take the square root of the sum of the squares of the average speed in the x-direction, the average speed in the y-direction, and the average speed in the z-direction.

The formula used for calculating the average speed is given below:

$$s^t = \sqrt{s_x^2 + S_y^2 + S_z^2}$$

Where s_x, s_y, s_z are the average speed in the x-direction, the average speed in the y-direction, and the average speed in the z-direction respectively.

AVERAGE ACCELERATION:

Acceleration is the rate of change of speed with respect to time. Since I already have the speed for the x, y, and Z coordinates for all 6 joints, I calculated the average acceleration for each joint in a similar way I calculated the average speed.

The equation used for calculating Average Acceleration is given below:

$$a = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Where, a_x, a_y, a_z are the average acceleration in the x-direction, the average acceleration in the y-direction, and the average acceleration in the z-direction respectively.

AVERAGE JERK:

Jerk is the rate of change of acceleration. It is a measure of how quickly the velocity of an object is changing. Jerk can be calculated by taking the derivative of acceleration. Jerk can be used to measure the smoothness of movement. A high jerk value indicates a sudden change in velocity, which can be jarring or uncomfortable. A low jerk value indicates a smooth and gradual change in velocity, which is often preferred.

The formula for jerk is:

$j = \frac{d}{dt}(a)$ where, a is the acceleration.

After calculating the Jerk for the x, y, and z directions for each joint, I calculated the average jerk for each joint using the formula below.

$J = \sqrt{J_x^2 + J_y^2 + J_z^2}$ where, J_x, J_y, J_z are the average jerk in the x-direction, the average jerk in the y-direction, and the average jerk in the z-direction respectively.

MINIMUM DISTANCE TO FOREARM:

In the study conducted by, Olugbade et al. (2020), the minimum distance between the Arm and other joints was a significant predictor of emotional distress in people with chronic pain.

This feature was calculated by finding the minimum Euclidean distance between the position coordinates of the right forearm and the other 5 joints.

The formula used to calculate the average jerk is given below.

$$D_{aj} = \sqrt{(x_a - x_j)^2 + (y_a - y_j)^2 + (z_a - z_j)^2}$$

Here, x_a, y_a, z_a are the x, y, and Z coordinates for the Right forearm and x_j, y_j, z_j are the x, y, and z coordinates of the 5 different joints.

AVERAGE ENERGY:

The formula for energy used for this study is $E = \frac{1}{2}v^2$, where, v is the velocity, which was calculated earlier, for each joint.

The average energy was calculated for each of the joints.

AMOUNT OF MOTION:

The amount of motion for each of the joints was calculated by measuring the difference between the final position of the joint in each direction from the initial position of the joint in those directions.

The equation for calculating the amount of motion is given below:

$$AOM = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2 + (z_f - z_i)^2}$$

x_f, y_f, z_f are the final position of the joint in x, y and z directions respectively. And x_i, y_i, z_i are the initial position of the joints in x, y and z directions respectively.

4.5 SCALING:

After extracting the 35 features from the segmented data, I scaled the data using a standard scaler.

Standard scaler is a standardization technique that is used to make sure that all of the features in a dataset have the same scale. This is important because machine learning algorithms often work better when the features are on a similar scale. [10]

Standard scaler scaling solves this problem by scaling all of the features in a dataset so that they have a mean of 0 and a standard deviation of 1. This means that all of the features will be on a scale of 0 to 1.

There are several benefits to standard scaler scaling. First, it can help to improve the performance of machine learning algorithms. When the features of a dataset are not normalized, the machine learning algorithm may not be able to learn the relationships between the features as effectively. This can lead to poor performance of the machine learning algorithm.

Second, it can help to prevent the machine learning algorithm from being biased towards certain features. When the features of a dataset are not normalized, the machine learning algorithm may give more weight to features that have larger values. This can lead to the machine learning algorithm making biased predictions.

4.6 Z score Normalization:

To balance the value range of heterogeneous sensor data, a normalization operation is performed.

In this study, I have used Z score normalization. Z score normalization is a statistical technique that is used to transform data so that it has a mean of 0 and a standard deviation of 1. This is done by subtracting the mean from each value and then dividing by the standard deviation. [11]

The formula for Z score normalization is given below:

$z = (x - \mu) / \sigma$,

where,

z is the z score,

x is the original value,

μ is the mean of the data,

σ is the standard deviation of the data.

4.7 RE-CODING THE LABEL INTENSITIES:

As mentioned in the dataset section above, the pain, worry, and confidence intensities are recorded in the dataset from a scale of 0 to 10 for pain and worry, and 0 to 5 for confidence. I re-coded these scales to a binary scale (Low or high).

I decided to re-code the labels because it can simplify the model training process. When there are a large number of possible values for a label, it can be difficult for the machine learning model to learn the relationships between the features and the labels. By reducing the number of possible values to two, the model can learn the relationships more easily.

Therefore, for the pain and worry labels, the intensities from 0 to 4 were re-coded into Low level (0) and intensities from 5 to 10 were re-coded into high level (1).

Similarly, for the confidence label, the intensities from 0 to 3 were re-coded into low level (0), and 4 to 5 were re-coded into high level (1).

After re-coding the labels into the binary scale, the distribution of each label is shown in Figures 3, 4, and 5 below.

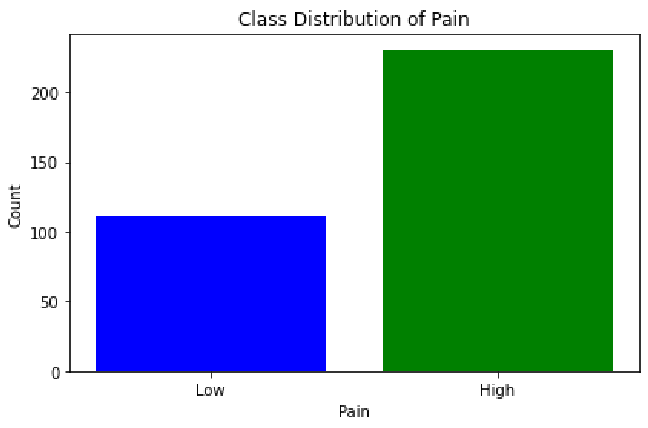


Fig 3. Showing the distribution of pain level intensities

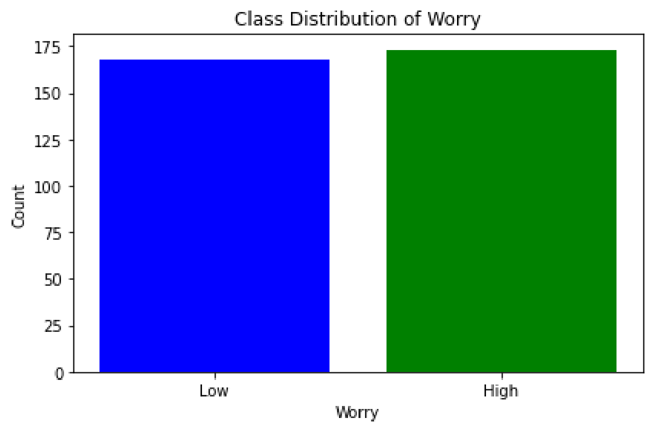


Fig 4. Showing the distribution of worry level intensities

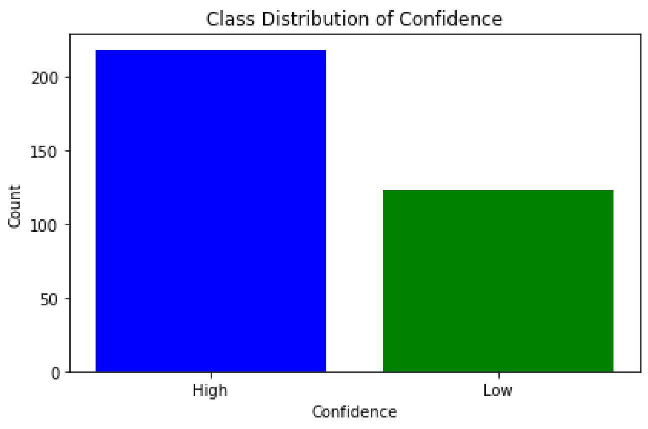


Fig 5. Showing the distribution of Confidence level intensities

From the distribution plots, we can see that there is high-class imbalance for the labels Pain and confidence. Which I will be addressing when training my Machine learning model.

4.8 PRE-PROCESSING THE INPUT DATA:

In the case of joint position data, I had to re-arrange the order of the joints to the correct anatomical order from chest bottom to the right lower-leg such that the Machine learning algorithms can learn the correct patterns when they are trained.

The joint position data had to be further processed into a 4D array of the dimensions (n x 2400 x 6 x 3), where n is the number of samples, 2400 is the number of frames in an instance, 6 is the number of joints and 3 is the x, y, and z axes. This is because the input for CNN layers should be of a specific shape.[12]

I decided to split the input data into train, validation and test sets. This was done for both types of input data (Extracted features input and joint positions sensor data input).

The split used was 80 percent for the training set and 20 percent for the test set. The training data was further split into validation set as well. With 80 percent for the training set and 20 percent for the validation set. This was done in order to fine-tune the models by comparing the performance of the models with the training and validation sets.

After the split, the 4D array is then converted into 4D tensors. This was done since the models that use the joint sensor data as input in my study have convolutional input layers, that require an input of this shape. [12]

The extracted features data however just required the conversion of data into tensors before being used as input to the machine learning algorithms.

4.9 LOSS FUNCTIONS:

In this study I have experimented with two different loss functions which are suitable for binary classification problems, they are binary cross entropy and binary focal cross entropy.

4.9.1 BINARY CROSS ENTROPY (BCE)

Binary cross entropy (BCE) is a loss function that is commonly used for binary classification problems. It is a measure of the difference between the predicted probabilities and the actual labels. [13]

The formula for BCE is given below,

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Where y is the label (1 for High-level intensity and 0 for low-level intensity) and p(y) is the predicted probability of the point being high for all N points.

4.9.2 BINARY FOCAL CROSS ENTROPY (BCEC)

Binary focal cross entropy (BCEC) is also a loss function that is used for binary classification problems. It is a modification of the binary cross entropy (BCE) loss function that is designed to pay more attention to misclassified examples.[14]

The BCE loss function which was previously mentioned, penalizes both false positives and false negatives equally. However, in some cases, it may be more important to correctly classify rare events, such as positive examples in a case of imbalanced data.

The BCEC loss function addresses this issue by introducing a modulating factor, called the focal loss, that is applied to the BCE loss function.

According to Lin et al., 2018, [15] the focal factor helps to downweight easy examples and focus more on hard examples. The focal loss is defined as follows:

focal_factor = (1 - output) ** gamma for class 1,

focal_factor = output ** gamma for class 0.

Where gamma is a focusing parameter. When gamma=0, this function is equivalent to the binary cross entropy loss(BCE).

Apart from the argument gamma, there are other arguments that are used when applying the BCEC, they are:

apply_class_balancing: A bool, whether to apply weight balancing on the binary classes 0 and 1.

alpha: A weight balancing factor for class 1, default is 0.25 as mentioned in reference Lin et al., 2018. The weight for class 0 can be calculated by the equation: 1.0 - alpha.

Reduction: This is a type of tf.keras.losses.Reduction to apply to loss. The Default value is AUTO. It indicates that the reduction option will be determined by the usage context.

In my study, the classes are unbalanced for Pain intensity and confidence intensity labels as shown in the figures 3 and 5 above. Hence using the correct loss function is necessary to get better results.

4.10 SINGLE TASK LEARNING (STL) MODELS:

For this study, I will be using TensorFlow to build my Neural Network models. For the single-task learning involving the usage of extracted features from the joint sensor data, I would be using Multilayer perceptron models. For the single-task learning using the join position sensor data as input, I will be using CNN-dense models.

4.10.1 MULTI-LAYER PERCEPTRONS

Multi-layer perceptron (MLP) is a supervised learning algorithm, that is commonly used for classification and regression tasks. They are made up of a series of interconnected layers, with each layer containing a number of nodes. The nodes in each layer are connected to the nodes in the next layer, and the strength of each connection is determined by a weight.[16]

MLPs are trained using a process called backpropagation. Backpropagation is an iterative process that involves calculating the error at the output layer of the network and then propagating the error back through the network to update the weights. This process is repeated until the network converges on a solution that minimizes the error.

I have decided to use MLPs as they are a versatile type of neural network that can be used for a variety of tasks, including classification. Especially Since my specific problem is binary classification of the intensities of pain, and its related expressions. MLP's are also relatively easy to train and interpret, which makes them a good choice.

The optimizer used for these models was the Adam optimizer. Adam optimizer is a stochastic gradient descent (SGD) method that is used to train machine learning models. It combines the advantages of two other popular SGD methods, AdaGrad and RMSProp.[17]

AdaGrad (Adaptive Gradient Algorithm) adapts the learning rate for each parameter based on the amount of progress that has been made with that parameter. This helps to prevent the learning rate from becoming too small or too large, which can help to prevent overfitting and underfitting.

RMSProp (Root Mean Squared Propagation), also adapts the learning rate for each parameter, but it does so based on the average of the squared gradients for that parameter. This helps to reduce the variance of the learning rate, which can help to improve the stability of the training process.

Adam combines the advantages of AdaGrad and RMSProp by using a moving average of both the gradients and the squared gradients. This helps to achieve a balance between the adaptive nature of AdaGrad and the stability of RMSProp.

The Loss function used is Binary focal cross-entropy for the labels with unbalanced classes (Pain and confidence), The binary focal cross-entropy loss function helps to apply balance to classes as mentioned in the section above.

For the label with balanced classes, I used the binary cross entropy loss function.

The activation function used for the output layer is sigmoid, which is considered to be the best at binary classification problems.

The figure below shows the visual representation of 1 hidden layer MLP.

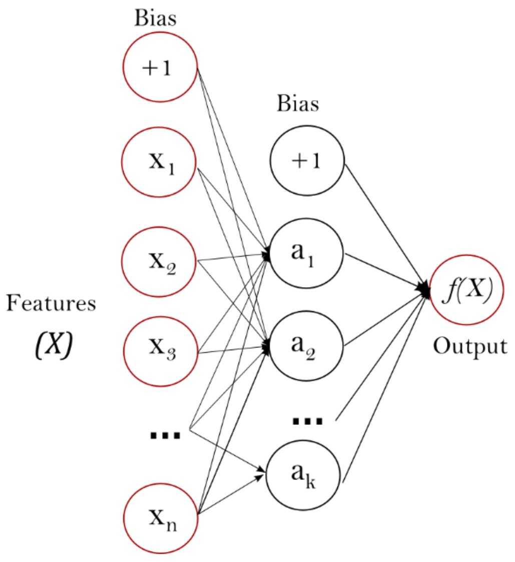


Fig 6: One hidden layer MLP[16]

4.10.2 CONVOLUTION NEURAL NETWORKS (CNN)

Convolutional Neural Networks (CNNs) are composed of a series of layers, each of which performs a specific task. The first layer is the convolutional layer, which applies a convolution operation to the input data. The convolution operation extracts features from the input data, such as edges and corners. The output of the convolutional layer is then passed to a pooling layer, which downsamples the output of the convolutional layer. This helps to reduce the size of the data and to improve the computational efficiency of the CNN. [18]

After each convolution operation, a CNN applies a ReLU transformation to the feature map. This transformation introduces nonlinearity to the model, which allows it to learn more complex features.[18]

In my study, I have used 2D convolution layers for their ability to perform feature extraction from the input data and also since 2D convolutional layers can learn to identify patterns in spatial data [19], it would be able to learn patterns in the anatomically ordered joint input data. After The Convolutional layers and Maxpooling layers, I introduced a few dense layers in my model to classify the data based on the features that were extracted by the CNN layers.

The optimizer used for these models is the Adam optimizer and the loss function used is binary focal crossentropy for unbalanced labels (Pain and confidence intensity) and binary cross entropy for the balanced class label (Worry intensity).

The output layer activation function used is sigmoid for the reasons mentioned in the previous section.

4.11 MULTI-TASK LEARNING MODEL (MTL):

Multi-task learning is a machine learning technique where we train a single model to do multiple tasks at once. This is in contrast to single-task learning, where we train a separate model for each task.[20]

There are a few advantages to multi-task learning. First, it can help to improve the performance of each task. This is because the model can learn to share information between the tasks, which can help it to better understand the data.

Second, multi-task learning can help to reduce the amount of data required to train each task. This is because the model can learn from the data for all of the tasks, even if it doesn't have a lot of data for each individual task.[21]

Third, multi-task learning can help to improve the generalization performance of the model. This is because the model can learn to generalize to new tasks by learning to share information between the tasks.

There are two different approaches when it comes to multi-task learning, they are, Hard and soft parameter sharing.

4.11.1 HARD PARAMETER SHARING:

Hard parameter sharing allows neural networks to share their hidden layers while retaining parts of their task-specific output layers. Overfitting is less likely when the majority of the layers for the associated tasks are shared. The parameters of the model are shared across all of the tasks. This means that the model learns a single representation that is used for all of the tasks.[22]

Fig shows the representation of hard parameter sharing. For hard parameter sharing approach, The shared layers would learn general features that are relevant to both tasks. The task-specific layers would then learn features that are specific to each task

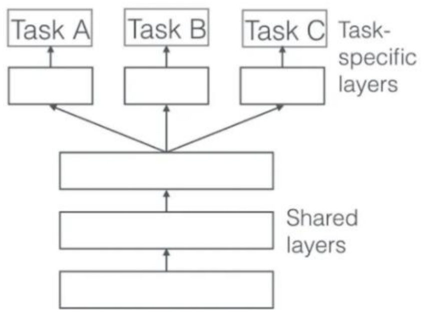


Fig 7: visual representation of hard parameter sharing in MTL.[22]

4.11.2 SOFT PARAMETER SHARING:

Soft parameter sharing is the process of regularising the distance between the parameters of individual models and the overall training objective in order to encourage similar model parameters across tasks. Regularisation techniques are often employed in Multi-Task Learning because they are simple to implement. [22]

In this case, the parameters of the model are not shared completely. This means that the model learns a different representation for each task, but the representations are similar to each other.

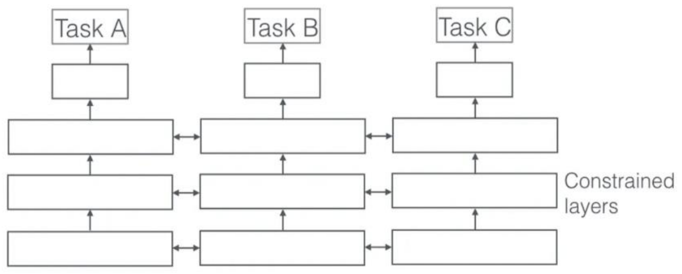


Fig 8 : visual representation of soft parameter sharing in MTL.[22]

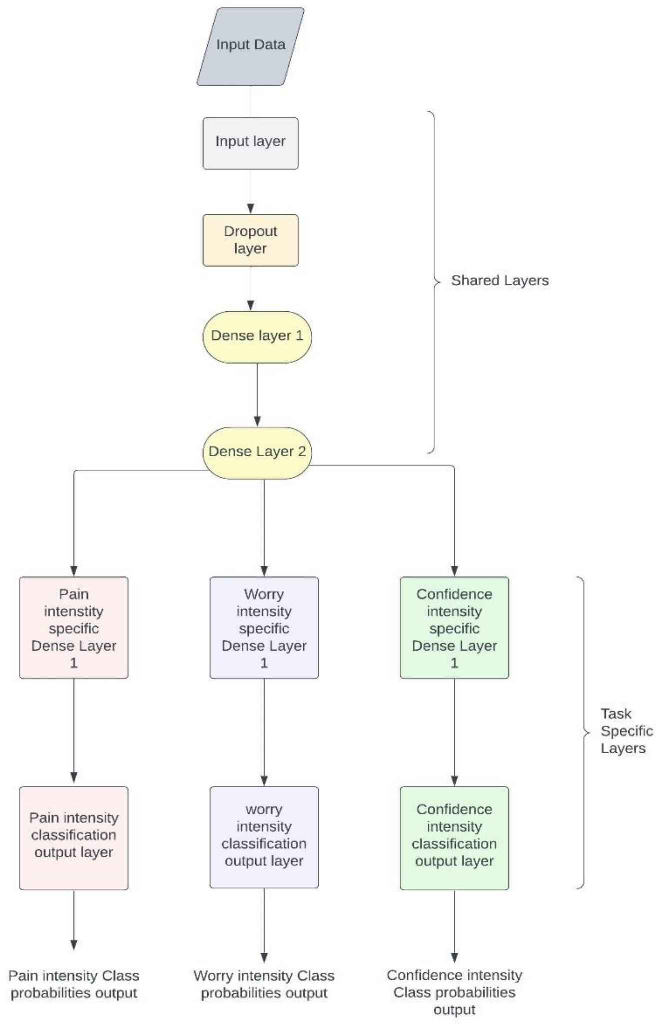
In my study, I will be developing a multi-task learning Neural Network model using the hard parameter sharing approach.

A sample MTL Model architecture is given in fig 9. The first few layers would be the shared layers, which would learn the shared parameters which are relevant for each of these tasks, the tasks are automatic prediction of pain, worry and confidence intensity.

The task-specific dense layers learn features that are specific to each task. For example, the pain-specific dense layer will learn features that are relevant to automatic pain detection.

Finally, the output of the task-specific dense layers is then used to make a binary classification for each task. The output activation function used is sigmoid since the aim of the model Is to do binary classification. The sigmoid function outputs the class probabilities. i.e., if the output probability is less than 0.5, then the class predicted is class 0. If the output of the function is greater than 0.5, then the predicted class is class 1.

The output probabilities have to be converted back into the binary classes before being compared or evaluated with the true labels.



Input Data
Input layer
Dropout layer
Dense layer 1
Dense Layer 2
Shared Layers
Pain intenstity specific Dense Layer 1
Worry intensity specific Dense Layer 1
Confidence intensity specific Dense Layer 1
Task Specific Layers
Pain intensity classification output layer
worry intensity classification output layer
Confidence intensity classification output layer
Pain intensity Class probabilities output
Worry intensity Class probabilities output
Confidence intensity Class probabilities output

Fig 9: Sample Multi-task neural network model architecture

4.12 EVALUATION METHODS:

The evaluation methods used in this study are discussed in this section.

4.12.1 CROSS VALIDATION:

The validation strategy used for all 7 models in this study is K-fold cross-validation.

K-fold cross-validation (K-CV) is a model validation technique that can be used to evaluate the performance of a machine learning model. It works by splitting the data into K folds and then using each fold as a validation set once. The model is trained on the remaining K-1 folds, and then evaluated on the validation fold. This process is repeated K times, and the average performance of the model across all K folds is reported. [23]

This method is a more robust approach to model validation than hold-out validation, as it helps to reduce the variance of the model's performance estimates. This is because the model is evaluated on different parts of the data each time, which helps to mitigate the effects of overfitting. [24]

The value of K is a hyperparameter that can be chosen by the user. A common value for K is 10, but other values can also be used. The choice of K depends on the size of the dataset and the desired level of robustness.

In this study I did not test for generalization to unseen participants because of the variation in the types of activities performed by different participants.

4.12.2 EVALUATION METRICS:

In this study, I have used the F1 scores, precision, and recall as the evaluation metrics.

I have used F1 scores because they are a measure of both precision and recall. Precision is the fraction of positive predictions that are actually positive, and recall is the fraction of positive instances that are correctly predicted. F1 score is the harmonic mean of precision and recall, which means that it gives equal weight to both metrics. Another reason I have for selecting the F1 score is because F1 scores work well with imbalanced datasets. [25]

5. AUTOMATIC DETECTION OF PAIN AND ITS RELATED EXPRESSIONS:

In this study, I have conducted 3 different experiments for the automatic prediction of Pain and pain-related expression intensities using joint position data from the EmoPain@Home dataset. The first two sets of experiments are conducted to address the research question, of whether manually extracted features from the sensor data perform better when used as input for the Machine learning model when compared to using the joint sensor data as input and relying on Machine learning algorithms to extract features.

The 1st experiment is the automatic prediction of pain, worry, and confidence intensities using features that are extracted from the joint positions data. The experiment will consist of building 3 different Multilayer perceptron models, that can do the binary classification of pain, worry, and confidence intensity levels separately. The overview of this experiment is given in the figure 10.

The 2nd experiment is the prediction of pain, worry, and confidence intensities using the position sensor data for all the joints. I will be building 3 different CNN-Dense models that can do the binary classification for the 3 labels. The overview of this experiment is given in fig 11.

After these two experiments are completed, I will be comparing the performance of all 6 models to see which models performed better, whether it is the MLP models with handcrafted features as input or the CNN-Dense models that used sensor data without feature extraction as the input.

The 3rd experiment uses the comparison results of experiments 1 and 2 to select the best type of input data to build a multi-task learning model that makes classification for pain, worry, and confidence intensities. The overview of this experiment is given in the figure 12.

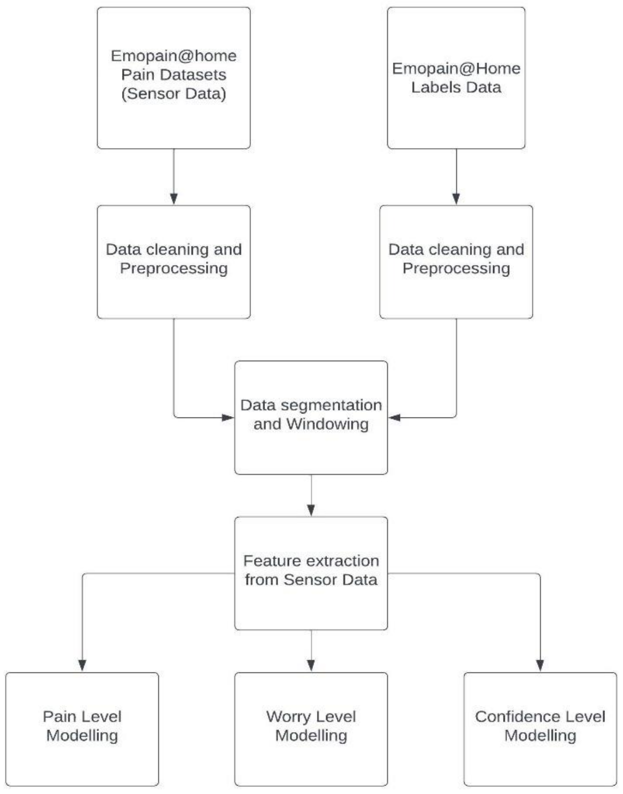

Emopain@home Pain Datasets (Sensor Data)
Emopain@Home Labels Data
Data cleaning and Preprocessing
Data cleaning and Preprocessing
Data segmentation and Windowing
Feature extraction from Sensor Data
Pain Level Modelling
Worry Level Modelling
Confidence Level Modelling

Fig 10: 1st Experiment Overview

For all of the experiments, I imported the sensor data for the people with chronic pain and then extracted the labels data which contains the pain, worry, and confidence intensity levels from the Emopain@home dataset. After that, I did the necessary data cleaning

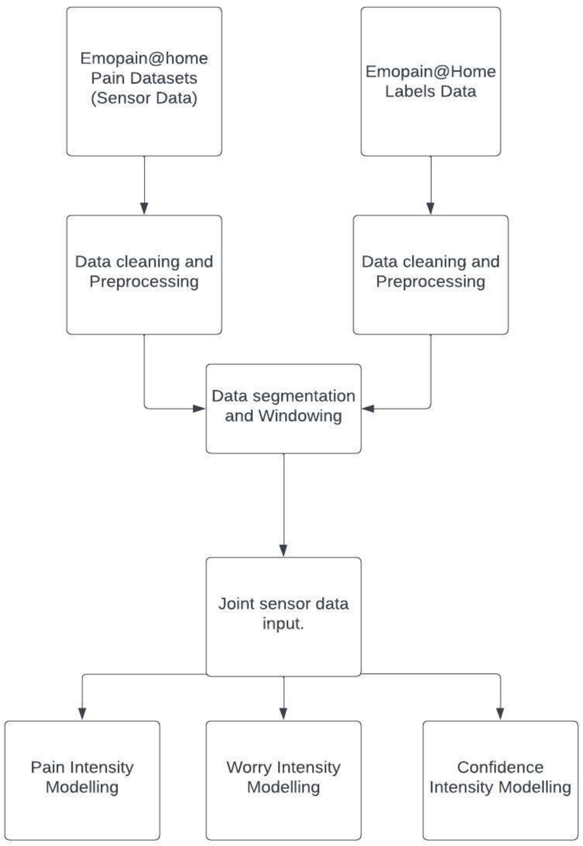

Emopain@home Pain Datasets (Sensor Data)
Emopain@Home Labels Data
Data cleaning and Preprocessing
Data cleaning and Preprocessing
Data segmentation and Windowing
Joint sensor data input.
Pain Intensity Modelling
Worry Intensity Modelling
Confidence Intensity Modelling

Fig 11: 2nd Experiment overview

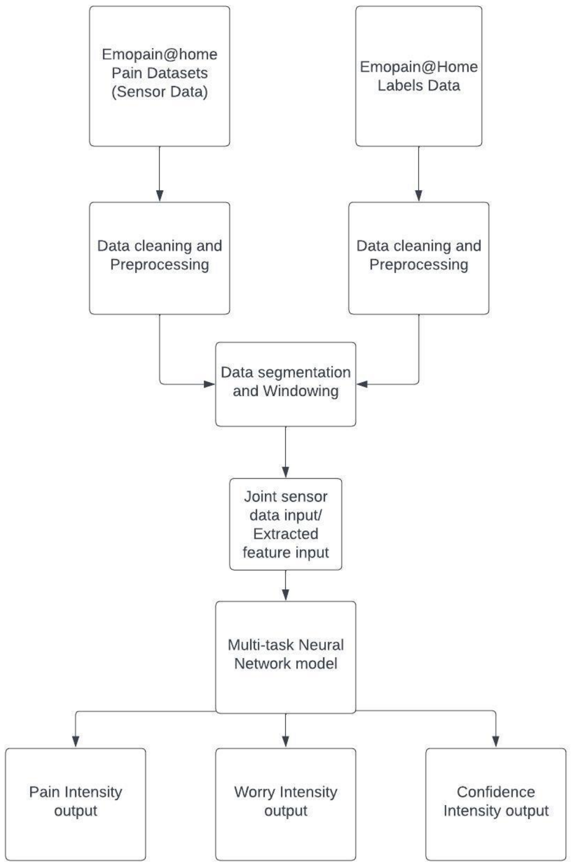

Emopain@home Pain Datasets (Sensor Data)
Emopain@Home Labels Data
Data cleaning and Preprocessing
Data cleaning and Preprocessing
Data segmentation and Windowing
Joint sensor data input/ Extracted feature input
Multi-task Neural Network model
Pain Intensity output
Worry Intensity output
Confidence Intensity output

Fig 12: 3rd experiment overview

and preprocessing followed by the windowing and segmentation of the data such that the label data matches the corresponding sensor data.

For the 1st experiment, I used the segmented data to extract 35 different features which is going to be used as input to my Machine learning models.

For the second experiment, I will normalize the segmented raw sensor data and use it as the input for my Machine learning models.

Finally, for the 3rd experiment, the multi-task model will be built using the type of input data that gave the best results when comparing the 1st and 2nd experiments.

6. EXPERIMENT DISCUSSION AND RESULTS

6.1 EXPERIMENT 1 (EXTRACTED FEATURES INPUT):

As mentioned in the previous section, the first experiment uses the 35 features that were manually extracted by me (Mentioned in the chapter) as input for the MLP models.

3 different MLP models were built for automatic prediction of pain, worry, and confidence intensities. The results of these are mentioned below.

6.1.1 AUTOMATIC DETECTION OF PAIN INTENSITY:

The MLP mode used in this experiment has two hidden layers, with 35 and 70 neurons respectively. The ReLU activation function is used in both hidden layers. The output layer has 1 neuron and uses the sigmoid activation function.

The model is trained using the binary focal cross entropy loss function and the Adam optimizer. Since the pain label has unbalanced classes.

The arguments used for the BCEC Loss function is: apply_class_balancing = True,

alpha=0.35,

gamma = 2.

These arguments were chosen since class 1 (High-level intensity) has more samples than class 0 (Low-level intensity) shown in the class distribution plot for pain intensity in fig 3 . Since, class 1 has more samples, the value of alpha should be lesser than 0.5 such that the class balancing feature of binary focal crossentropy applies the correct weights to the majority class.

The model is set to train for 800 epochs with a batch size of 100 and validated using the validation set. I have used an early stopping callback that monitors the validation loss. If the validation loss does not improve for 20 epochs, the training will be stopped. The best weights will be restored, which are the weights that achieved the lowest validation loss. The loss graphs for Training Vs validation is shown in fig 13.

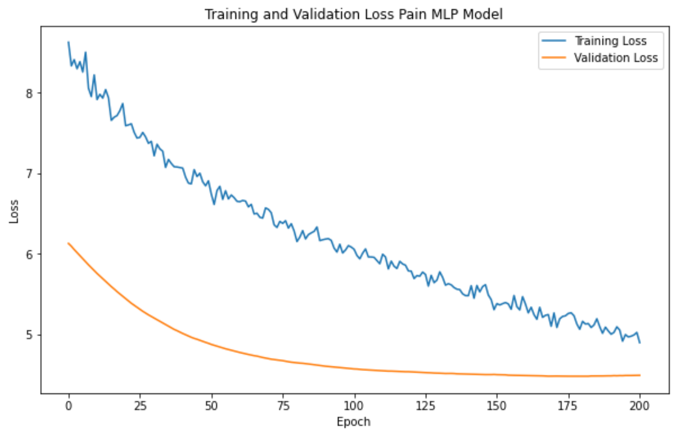


Fig 13: Training Vs Validation loss graph (Pain MLP)

Using the validation based early stopping strategy helps prevent overfitting and also underfitting. [26]

Using this strategy, the model trained for 200 epochs and stopped training on the training set.

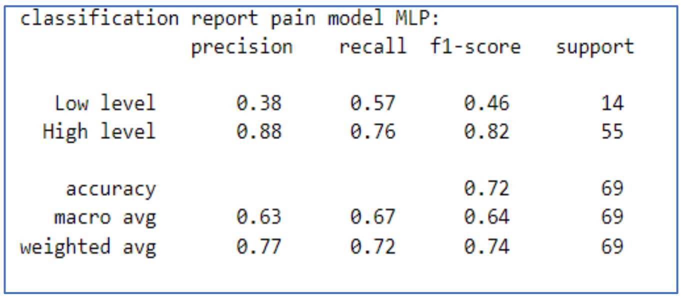
The graph shows that the training loss decreases as the model is trained. This is because the model is gradually learning to fit the training data better. The graph also shows that the validation loss decreasing with along the same pace of training loss, this shows that the model is not underfitting. The validation loss doesn't seem to be increasing as well, this shows that the model isn't overfitting. [27]

The parameters of the model were fine-tuned by the method of experimentation.

The results of this model gave F1 scores of 0.46 and 0.82 for class 0 (low level pain) and class 1 (High level pain) respectively.

The classification report for the model is shown in the figure 14.

10-fold cross validation was used to evaluate the performance of the model, which achieved an average weighted F1 score of 0.78.



```
classification report pain model MLP:
              precision    recall  f1-score   support

   Low level       0.38      0.57      0.46        14
  High level       0.88      0.76      0.82        55

    accuracy                           0.72        69
   macro avg       0.63      0.67      0.64        69
weighted avg       0.77      0.72      0.74        69
```

Fig 14: Classification report (Pain MLP)

6.1.2 AUTOMATIC DETECTION OF WORRY INTENSITY:

The MLP model which was used in this experiment has two hidden layers and uses Adam optimizer and the loss function used in this experiment is binary crossentropy loss. This is because the distribution of classes is balanced for the worry intensity labels, which is shown in fig 3 and 5.

Similar to the 1st experiment, the model is set to be trained for 800 epochs with a batch size of 100 and is also validated using the validation set. The early stopping strategy was also used in this experiment, with a patience of 20.

The loss graph for training Vs validation Is shown in the figure below.

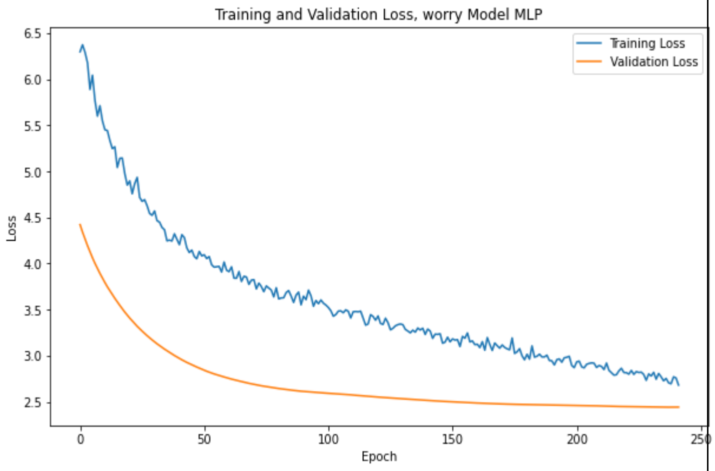


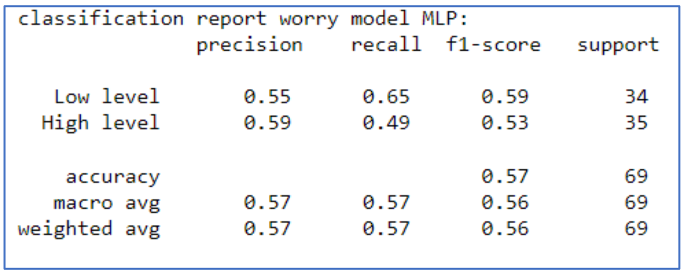
Fig 15: Training vs Validation Loss (Worry MLP)

Using the early stopping strategy, the model trained for 250 epochs and stopped training on the training set.

By analysing the graph, I could see that the model is prevented from overfitting and underfitting.

The parameters of the model were fine-tuned by the experimentation to get the final fine-tuned model.

This model gave F1 scores of 0.59 and 0.53 for the Low level worry intensity and High level worry intensity respectively. The classification report for this model is shown in the figure below.



```
classification report worry model MLP:
              precision    recall  f1-score   support

   Low level       0.55      0.65      0.59        34
  High level       0.59      0.49      0.53        35

    accuracy                           0.57        69
   macro avg       0.57      0.57      0.56        69
weighted avg       0.57      0.57      0.56        69
```

Fig 16: Classification report (Worry MLP)

The 10-fold cross validation for this model gave an average weighted F1 score of 0.74.

6.1.3 AUTOMATIC DETECTION OF CONFIDENCE INTENSITY:

MLP model with 2 hidden layers were used in this experiment. Like the first experiment, the optimizer used was Adam optimizer and the loss function used is Binary focal cross entropy. Since the confidence labels have uneven distribution of classes. The distribution is shown in the fig.

The arguments for the BECE loss function are as follows:

apply_class_balancing=True,

alpha=0.7,

gamma = 3

The model is set to train for 800 epochs with a batch size of 100 same as the other experiments. Validation based early stopping was used.

The training and Validation loss graphs is shown below.

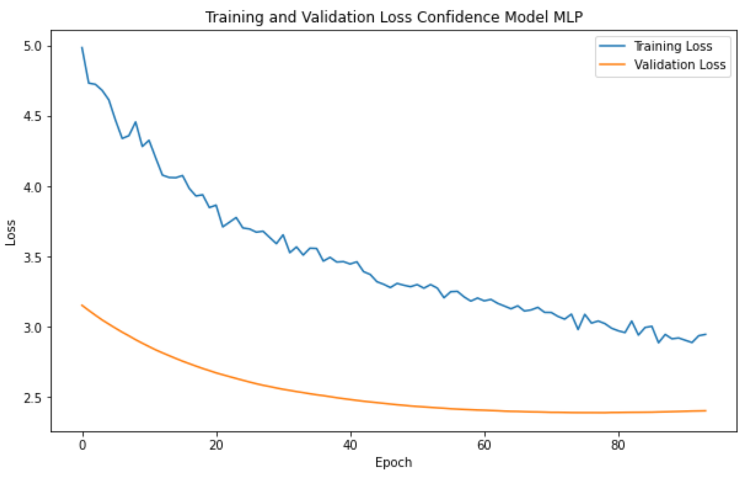


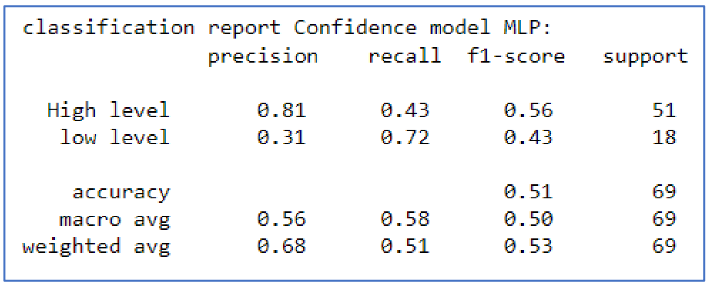
Fig 17: Training vs Validation Loss (Confidence MLP)

The model was trained for 100 epochs and the Training Vs validation graph indicates that the Model was learning the features without overfitting or underfitting.

It is important to note that, the confidence for the participants were reported on the scale of 0 – 5 where, 0 is the highest confidence intensity and 5 is the lowest intensity.

The model achieved F1 scores of 0.56 and 0.43 for High-level and low-level confidence intensity, respectively.

The classification report for the confidence MLP model is given below.



```
classification report Confidence model MLP:
              precision    recall  f1-score   support

  High level       0.81      0.43      0.56        51
   low level       0.31      0.72      0.43        18

    accuracy                           0.51        69
   macro avg       0.56      0.58      0.50        69
weighted avg       0.68      0.51      0.53        69
```

Fig 18: Classification report (Confidence MLP)

The classification report shows that the weighted average

The 10-Fold cross validation on this model achieved an average weighted F1 score of 0.58.

6.2 EXPERIMENT 2 (RAW SENSOR DATA INPUT):

The experiment 2 as mentioned in the previous section uses the raw sensor data as input without any manual extraction of features.

For this set of experiments, 3 separate CNN-Dense model were built for automatic detection of the pain and its related expression intensities.

The results of these experiments are mentioned in detail below.

6.2.1 AUTOMATIC DETECTION OF PAIN INTENSITY:

This experiment used a CNN-dense model, the output activation function used is sigmoid, the Optimizer used is the Adam optimizer and the loss function used is Binary focal cross entropy.

The arguments for the loss function is given below:

apply_class_balancing=True,

alpha=0.3,

gamma=2.

The model is set to be trained for 500 epochs using the training set and then validated using the validation set. Like the previous experiments, Validation based early stopping strategy was used to prevent overfitting.

The model trained for 100 epochs before the training was stopped.

The training Vs Validation loss graph is shown in the figure below.

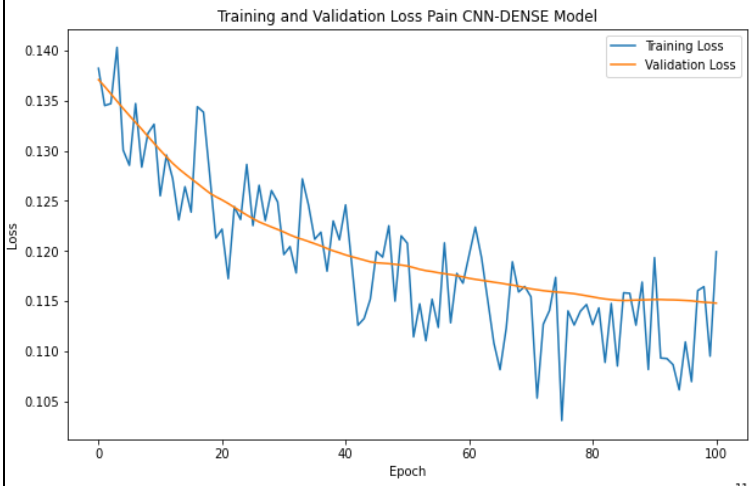


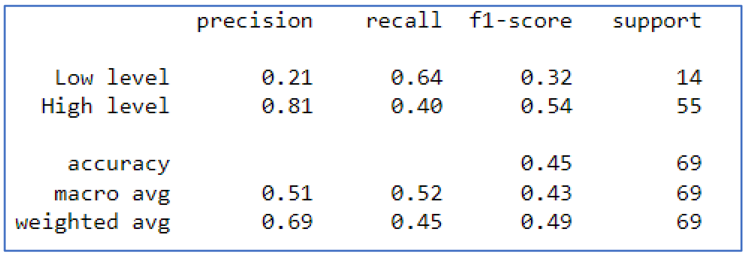
Fig 19: Training vs Validation Loss (Pain CNN-Dense)

From the loss graph, we can see that the training loss is decreasing along with the validation loss. We can also see that the model is trying to overfit while training and there are instances when the Validation loss becomes greater than the training loss.

The validation based early-stopping strategy played a significant role in stopping the training at 100 epochs to avoid overfitting and underfitting.

The model achieved F1 scores of 0.32 and 0.54 for the low level and high-level intensities respectively.

The classification report shown in the figure below shows the results of the model.



```
              precision    recall  f1-score   support

   Low level       0.21      0.64      0.32        14
  High level       0.81      0.40      0.54        55

    accuracy                           0.45        69
   macro avg       0.51      0.52      0.43        69
weighted avg       0.69      0.45      0.49        69
```

Fig 20: Classification report (Pain CNN-Dense)

The classification report is the best result of experimenting with various parameters of the model for fine-tuning.

In a similar way to the previous experiments, 10-fold cross validation was performed which achieved a average weighted F1 score of 0.50.

6.2.2 AUTOMATIC DETECTION OF WORRY INTENSITY:

This experiment has the same model architecture as the model used in the previous experiment. The optimizer used is Adam optimizer and the loss function used in this experiment is binary crossentropy loss. The model is set to be trained for 500 epochs with a batch size of 100. The Validation based early stopping is used with a patience of 20.

L2 regularization was used in this model to prevent overfitting. L2 regularization adds a penalty to the loss function for large weights. [28] This encourages the model to find weights that are smaller and less complex, which can help to improve the model's ability to generalize.

The model trained for 500 epochs and the train Vs Validation loss graph is shown in the figure below.

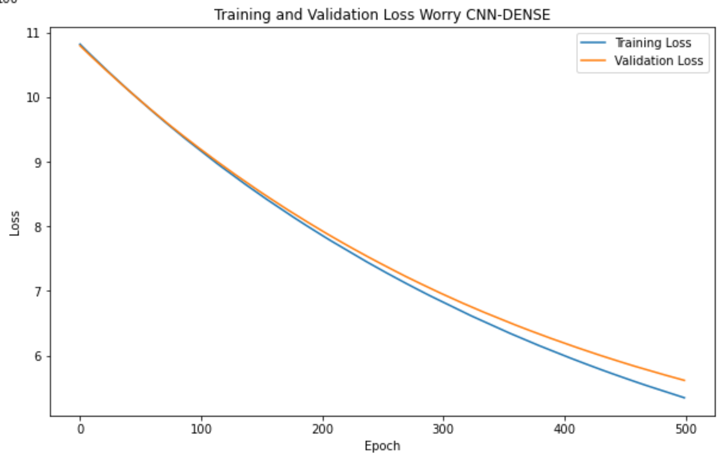


Fig 21: Training vs Validation Loss (Worry CNN-Dense)

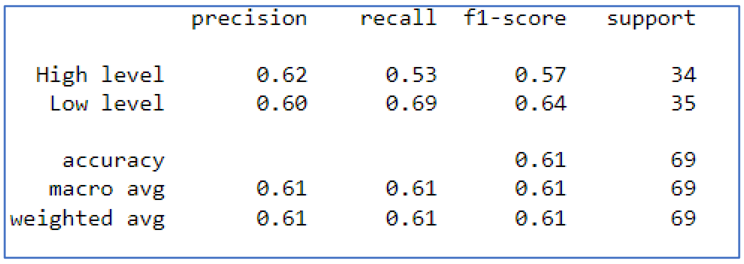
In this learning curve plot, the training loss decreases as the number of epochs increases. This means that the model is learning to fit the training data better as it is trained for more epochs. However, the validation loss is increasing after a certain number of epochs. This means that the model is starting to overfit to the training data.

The ideal learning curve would have a training loss that is always decreasing and a validation loss that is always decreasing or staying constant.[27] However, this is not always possible, and it is often necessary to trade-off between a good training loss and a good validation loss.

In this case, the model could be stopped at an earlier epoch to prevent overfitting. This would result in a higher training loss, but it would also result in a lower validation loss. The optimal number of epochs to train the model would depend on the desired balance between training loss and validation loss.

The model achieved F1 scores of 0.57 and 0.64 for low-level and high-level intensities respectively.

The classification report for this model is shown in the figure below.



```
              precision    recall  f1-score   support

  High level       0.62      0.53      0.57        34
   Low level       0.60      0.69      0.64        35

    accuracy                           0.61        69
   macro avg       0.61      0.61      0.61        69
weighted avg       0.61      0.61      0.61        69
```

Fig 22: Classification report (Worry CNN-Dense)

From the classification report we can see that the overall accuracy of the model is 61%, which means that 61% of the instances in the dataset were classified correctly. The model performed equally well on both classes, with a precision, recall of 61 for each class.

The 10-fold cross validation of this model achieved an average weighted F1 score of 0.93

6.2.3 AUTOMATIC DETECTION OF CONFIDENCE INTENSITY:

The experiment used similar model architecture as the previous experiments in this section. The loss function used is binary focal crossentropy due to unbalanced class distribution. The arguments used are mentioned below:

apply_class_balancing=True,

alpha=0.7,

gamma=3. The model uses validation based early stopping strategy which helped the model to avoid

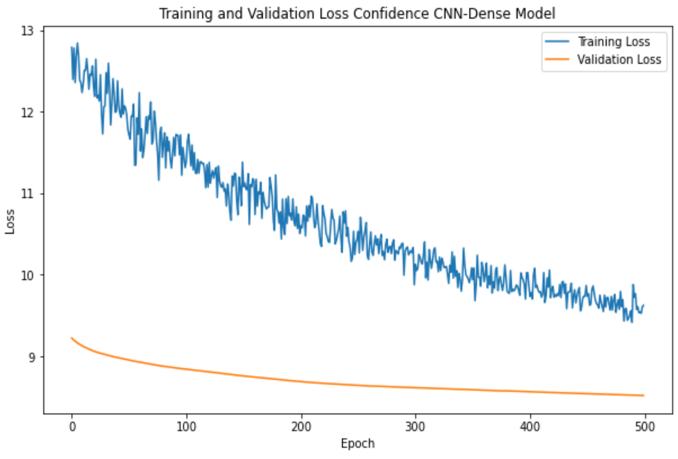


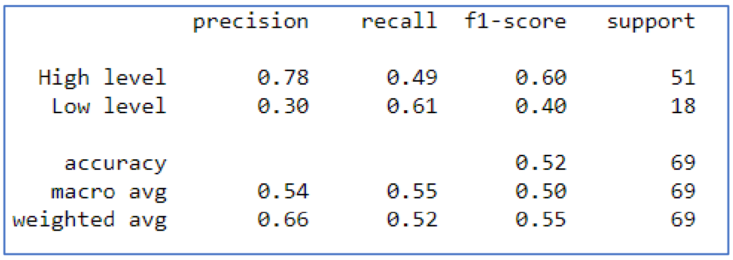
Fig 23: Training vs Validation Loss (Confidence CNN-Dense Model)

overfitting to the training data. The model trained for 500 epochs. Loss graphs are shown in the figure.

From the learning curves in the fig 23, we can see that the training loss and the validation loss is decreasing and the gap between the curves are decreasing with the increase in number of epochs, this shows the model is learning well with the training data and adapting well with unseen data.[27]

The model achieved F1 scores of 0.60 and 0.40 for high-level and low-level confidence intensities.

The classification report of the model is shown in the table below.



```
              precision    recall  f1-score   support

  High level       0.78      0.49      0.60        51
   Low level       0.30      0.61      0.40        18

    accuracy                           0.52        69
   macro avg       0.54      0.55      0.50        69
weighted avg       0.66      0.52      0.55        69
```

Fig 24: Classification report (Confidence CNN-Dense)

The 10-fold cross validation of this model achieved an average weighted F1 score of 0.82.

7. MODEL COMPARISON

In this section, I will be comparing the 6 different machine learning models using their F1 Scores to determine which model has the best balance of precision and recall for classifying High level and Low-level instances.

The table below shows the comparison of the 6 different models.

		F1 Scores		
Models	Classes	Pain Intensity	Worry Intensity	Confidence Intensity
MLP Models	High Level	0.82	0.59	0.56
	Low level	0.46	0.53	0.43
CNN-DENSE Models	High Level	0.54	0.57	0.60
	Low level	0.32	0.64	0.40

Table 1: Model comparison F1 scores

The comparison tables shows that the MLP Models and CNN-DENSE Models have different strengths and weaknesses when it comes to classifying Pain Intensity, Worry Intensity, and Confidence Intensity instances.

The MLP Model have a higher F1 score for the Pain Intensity class. This means that the model is able to classify high and low level intensities better than the CNN-Dense model. This does suggest that the MLP model is better at capturing the nuances of this class.

The CNN-DENSE Models have a higher F1 score for the High-level Worry Intensity and Confidence Intensity classes, suggesting that they are better at capturing the better patterns associated with these classes. This could be because the CNN layers are better at extracting features that are associated with High level Worry Intensity and Confidence Intensity. This could also be because of the uneven distribution of the classes.

To get more insight into the performance of the models, I had run 10-fold cross validation on them. The average weighted F1 scores for the different models are shown in the table below.

	Average weighted F1 Scores		
Models	Pain Intensity	Worry Intensity	Confidence Intensity
MLP Model	0.78	0.76	0.58
CNN-DENSE Model	0.50	0.93	0.82

Table 2: Model comparison 10-fold Cross Validation

The table shows that the MLP model is better at predicting pain intensity than the CNN-Dense model, with an average weighted F1 score of 0.78. This is a significant difference.

Whereas, the CNN-Dense model is better at predicting worry intensity and confidence intensity. This was also observed in table 2.

The results of these experiments suggest that using joint position sensor data as input along with CNN layers to extract features is a better way to automatically predict worry and confidence intensity than using features that are manually extracted from the joint position sensor data.

My findings provide evidence that there is a significant difference in performance between using the two different input types for training models which answers my 1st research question.

8. EXPERIMENT 3: MULTI-TASK LEARNING MODEL FOR AUTOMATIC DETECTION OF PAIN, WORRY AND CONFIDENCE INTENSITIES.

In this section, I will be using the raw sensor data as input for the multi-task learning model. The model architecture is shown in Fig 28. The model uses two shared 2D convolutional layers for feature extraction and task-specific dense layers for feature learning and prediction of Pain, worry and confidence intensities.

It uses Adam optimizer and has different loss functions for different tasks. The first task is the automatic detection of pain intensity, the loss function used for this task is binary focal cross entropy with the following arguments:

apply_class_balancing=True,

alpha=0.35,

gamma=2

The second task is the automatic detection of worry intensity. The loss function used is binary crossentropy, since the distribution of classes are not unbalanced.

The third task is the automatic detection of confidence intensity. The loss functioned used is binary focal crossentropy with the following arguments.

apply_class_balancing=True,

alpha=0.6,

gamma=2.

For this experiment I have used validation based early stopping strategy with a patience of 20.

The training Vs validation loss graphs for all three tasks are shown in the plot below. Fig 25,26.

The model trained for 500 epochs as seen in the graph.

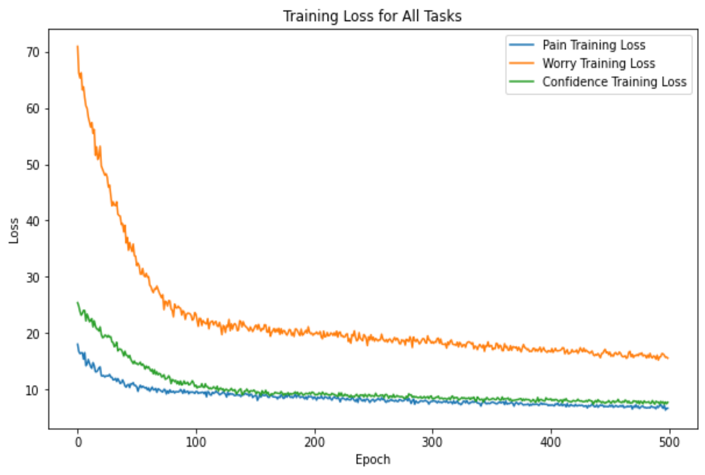


Fig 25: Training Loss graph (Multi-task Model)

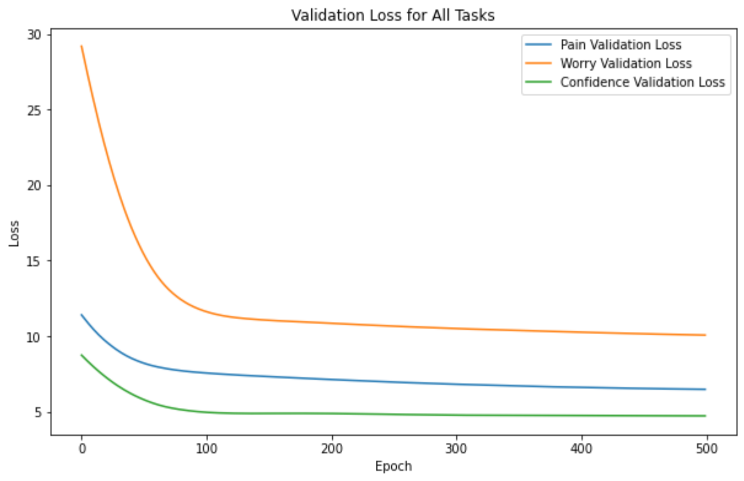


Fig 26: Validation Loss graph (Multi-task Model)

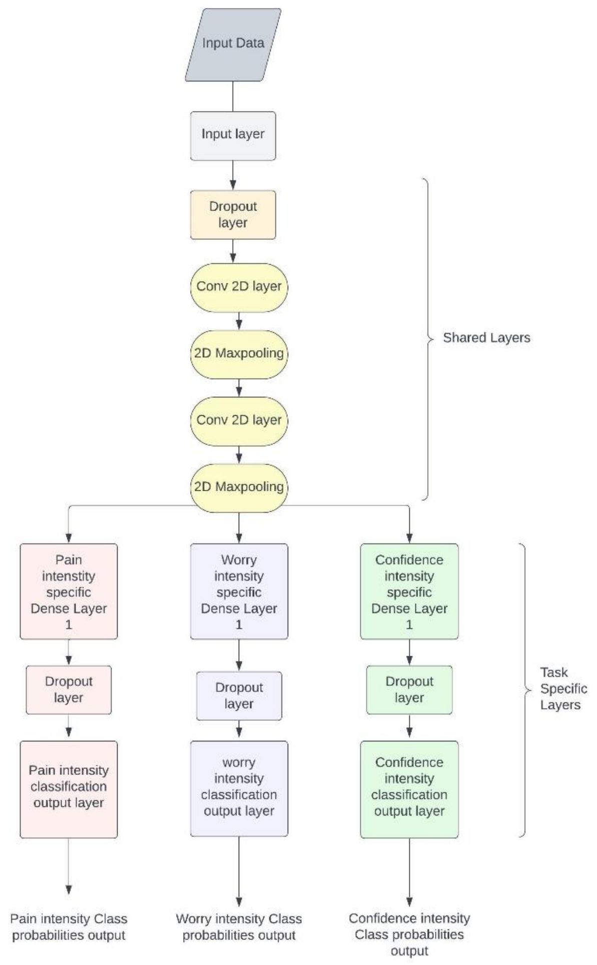
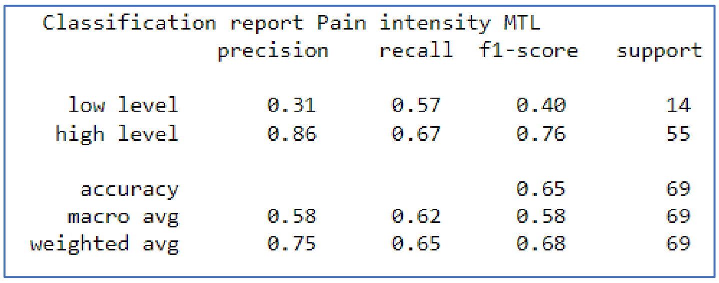

Input Data
Input layer
Dropout layer
Conv 2D layer
2D Maxpooling
Conv 2D layer
2D Maxpooling
Shared Layers
Pain intenstity specific Dense Layer 1
Worry intensity specific Dense Layer 1
Confidence intensity specific Dense Layer 1
Dropout layer
Dropout layer
Dropout layer
Task Specific Layers
Pain intensity classification output layer
worry intensity classification output layer
Confidence intensity classification output layer
Pain intensity Class probabilities output
Worry intensity Class probabilities output
Confidence intensity Class probabilities output

Fig 28: The architecture of the Multi-task learning model used in this study.

From the learning curves, we can see that the model is training the three tasks in an efficient way while avoiding overfitting. The gap between the training and validation curves for each task is decreasing with increase in the number of epochs.

For the first task, the model achieved F1 scores of 0.40 and 0.76 for the low level and high-level pain intensity. The classification report for the first task is shown below.



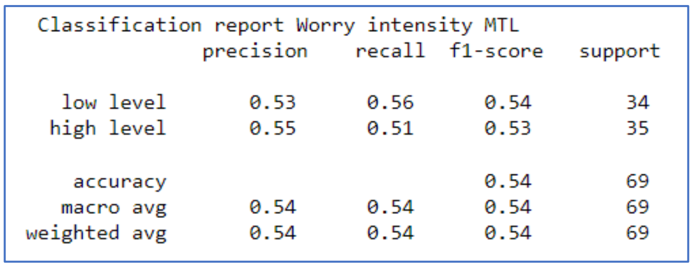
```
 Classification report Pain intensity MTL
              precision    recall  f1-score   support

   low level       0.31      0.57      0.40        14
  high level       0.86      0.67      0.76        55

    accuracy                           0.65        69
   macro avg       0.58      0.62      0.58        69
weighted avg       0.75      0.65      0.68        69
```

Fig 27: classification report Pain task (Multi-task Model)

For the second task, the model achieved F1 scores of 0.54 and 0.53 for the low level and high-level worry intensity. The classification report for this task is shown below.



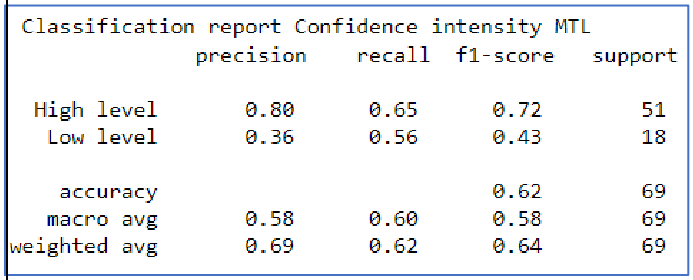
```
 Classification report Worry intensity MTL
              precision    recall  f1-score   support

   low level       0.53      0.56      0.54        34
  high level       0.55      0.51      0.53        35

    accuracy                           0.54        69
   macro avg       0.54      0.54      0.54        69
weighted avg       0.54      0.54      0.54        69
```

Fig 29: classification report Worry task (Multi-task Model)

And finally for the third task, the model achieved F1 scores of 0.72 and 0.43 for high level and low-level



```
 Classification report Confidence intensity MTL
              precision    recall  f1-score   support

  High level       0.80      0.65      0.72        51
   Low level       0.36      0.56      0.43        18

    accuracy                           0.62        69
   macro avg       0.58      0.60      0.58        69
weighted avg       0.69      0.62      0.64        69
```

Fig 30: classification report Confidence task (Multi-task Model)

confidence intensity respectively. The classification report is shown below.

The Multi-task model was evaluated using 10-fold cross validation and the model achieved average weighted F1 scores of 0.69, 0.62 and 0.62 for pain, worry and confidence detection tasks.

9. COMPARISON STL VS MTL

This section covers the comparison of the performance difference between the single task learning models using raw sensor data, i.e., the CNN-Dense models and the Multi-task learning model.

Table 3 shows the comparison of F1 scores obtained for the STL models and the MTL model.

The results show that the multi-task model performs slightly better in classifying the different levels of pain intensities, achieving F1 scores of 0.76 and 0.40 for the high and low level pain respectively. Although this isn't a significant improvement in the models performance, we can say that the MTL model performs better than the STL model when using Raw sensor data as the input. The same results can be observed for the Confidence detection task, where the MTL showed better performance than the STL model for classification of different levels of confidence.

On the contrary, the MTL model performed slightly worse for the task of predicting different levels of worry intensity.

		F1 Scores		
Models	Classes	Pain Intensity	Worry Intensity	Confidence Intensity
STL Models	High Level	0.54	0.57	0.60
	Low level	0.32	0.64	0.40
MTL model	High Level	0.76	0.54	0.72
	Low level	0.40	0.53	0.43

Table 3: Model comparison F1 scores (STL Vs MTL)

The 10-fold cross validation scores for the models are shown in the table4.

The comparison of average weighted F1 scores showed an interesting result, the MTL learning model seems to perform well with the task of detection of pain intensity and significantly worse with the other two tasks. This means there is room for improvement in the MTL model.

	Average weighted F1 Scores		
Models	Pain Intensity	Worry Intensity	Confidence Intensity
STL Models	0.50	0.93	0.82
MTL model	0.69	0.62	0.62

Table 4: Model comparison 10-fold Cross Validation (STL vs MTL)

10. CONCLUSION AND FURTHER WORK

In this study, movement related pain and its expressions were predicted for people with chronic pain by using joint positions data from the EmoPain@home dataset.

Different experiments were conducted in this study for the automatic detection of the pain expressions and also to compare the performance of two different types of inputs to machine learning algorithms. The first type of input was the extracted features input which consist of the 35 different features which were extracted manually from the joint position's sensor data, whereas the 2nd type of input was the raw sensor data.

Two different model architectures were used for the two different input types. Multi-layer perceptron models for the experiments using the extracted features as input and CNN-Dense models for the experiments using the raw sensor data as input.

The results of these experiments show that for the automatic detection of pain intensity, the MLP model seems to perform significantly better than the CNN-Dense model with an average weighted F1 score of 0.78 when compared to the score of 0.50 achieved by the CNN-Dense model.

This could indicate that the MLP model is better at learning the patterns for distinguishing between the low level and high-level pain intensities.

The training of these models was limited by the small sample size. This could also be the reason the CNN-Dense model performed poorly for this task.

There is room for Further improvement in the automatic prediction of pain intensity, using MLP models using different input types as mentioned in the study conducted by Olugbade et al. (2022) [7], in this study, the ensemble decision tree model seems to have performed well for automatic detection of pain intensities when the model used joint angles features extracted from the joint angles data in the EmoPain@Home dataset.

The model achieved F1 scores of 0.63 and 0.61 for low level and high level pain respectively. This suggests that further experiments should be carried out using joint angles features data or the automatic detection of pain intensities.

When it comes to automatic detection of confidence and worry intensities, the CNN-Dense models perform significantly better than the MLP models with average weighted F1 scores of 0.82 and 0.93 as shown in table 2.

These experiments suggest that the 2D convolutional layers used in these CNN-Dense models are able to extract features which were better in terms of learning the patterns for different levels of confidence and worry intensities. However, these experiments are also restricted by the small sample size. Further research on these tasks could be conducted using the joint angles sensor data in the EmoPain@Home dataset.

The two experiments conducted on the two different input types answer my research question on the performance difference between using manually extracted features and Machine learning algorithm extracted features as inputs. The result is evident in this research is that features extracted by the machine learning algorithm perform significantly better for two tasks, the automatic detection of confidence and worry intensity. These results don't confirm the supremacy of using algorithm extracted features, rather these results call for further research into the comparison of using the two different methods of input on different Model architectures and input modalities.

The Multi-task model built in this research does all three tasks of automatic detection of pain, worry, and confidence intensities simultaneously. The model achieved average weighted F1 Scores of 0.69, 0.62, and 0.62 for the automatic detection of pain, worry, and confidence intensities respectively.

The results of the MTL model when compared to the STL models (MLP models) don't show significant performance difference.

The STL models using raw sensor data as input performed better for the automatic detection tasks for worry and confidence intensities.

Further research can also be conducted for automatic prediction of pain, worry and confidence intensities using the joint angles sensor data as inputs to understand the difference in performance between the joint positions data and joint angles data inputs.

Overall, the results of this study suggest that MLP models are better at automatic detection of pain intensity, while CNN-Dense models are better at automatic detection of confidence and worry intensities. The MTL model performed well on all three tasks, but did not show significant performance improved when compared to the STL models.

This study was conducted on a smaller sample of people with chronic pain. hence the results of this study should be interpreted with caution because the results may not be generalizable to a larger population of people with chronic pain. Further research is needed to confirm the findings of this study with a larger sample size.

11. REFERENCES

[1] Johns Hopkins Medicine (n.d.). Chronic Pain. [online] www.hopkinsmedicine.org. Available at: https://www.hopkinsmedicine.org/health/conditions-and-diseases/chronic-pain. [Accessed 5 July. 2023]

[2] Cleveland Clinic (2021). Chronic Pain: Symptoms, Treatments. [online] Cleveland Clinic. Available at: https://my.clevelandclinic.org/health/diseases/4798-chronic-pain. [Accessed 5 July. 2023]

[3] Linton, S.J. and Shaw, W.S. (2011). Impact of Psychological Factors in the Experience of Pain. Physical Therapy, [online] 91(5), pp.700–711. doi: https://doi.org/10.2522/ptj.20100330.

[4] Fillingim, R.B., Loeser, J.D., Baron, R. and Edwards, R.R. (2016). Assessment of Chronic Pain: Domains, Methods, and Mechanisms. The Journal of Pain : Official Journal of the American Pain Society, [online] 17(9),pp.T10–T20. doi: https://doi.org/10.1016/j.jpain.2015.08.010 .

[5] Olugbade, T.A., Bianchi-Berthouze, N., Marquardt, N. and de C. Williams, A.C. (2018). Human Observer and Automatic Assessment of Movement Related Self-Efficacy in Chronic Pain: From Exercise to Functional Activity. IEEE Transactions on Affective Computing, 11(2), pp.214–229. doi: https://doi.org/10.1109/taffc.2018.2798576 .

[6] Temitayo Olugbade, Singh, A., Bianchi-Berthouze, N., Marquardt, N., Min and Amanda (2019). How Can Affect Be Detected and Represented in Technological Support for Physical Rehabilitation? ACM Transactions on Computer-Human Interaction, 26(1), pp.1–29. doi:

https://doi.org/10.1145/3299095

[7] Olugbade, T.A., Raffaele Andrea Buono, Amanda, De, S., Gold, N., Holloway, C. and Bianchi-Berthouze, N. (2022). EmoPain(at)Home: Dataset and Automatic Assessment within Functional Activity for Chronic Pain Rehabilitation. doi: https://doi.org/10.1109/acii55700.2022.9953831 .

[8] Bento, N., Rebelo, J., Marília Barandas, Carreiro, A.V., Campagner, A., Cabitza, F. and Gamboa, H.

(2022). Comparing Handcrafted Features and Deep Neural Representations for Domain Generalization in Human Activity Recognition. Sensors, 22(19), pp.7324–7324. doi: https://doi.org/10.3390/s22197324 .

[9] Barut, O., Zhou, L. and Luo, Y. (2020). Multitask LSTM Model for Human Activity Recognition and Intensity Estimation Using Wearable Sensor Data. IEEE Internet of Things Journal, 7(9), pp.8760–8768. doi: https://doi.org/10.1109/jiot.2020.2996578 .

[10] Scikit-Learn (2019). sklearn.preprocessing.StandardScaler — scikit-learn 0.21.2 documentation. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. [Accessed 10 July. 2023]

[11] Zach (2021). Z-Score Normalization: Definition & Examples. [online] Statology. Available at: https://www.statology.org/z-score-normalization/.

[Accessed 10 July. 2023]

[12] Team, K. (n.d.). Keras documentation: Conv2D layer. [online] keras.io. Available at: https://keras.io/api/layers/convolution_layers/convolution2d/#:~:text=The%20ordering%20of%20the%20dimensions. [Accessed 11 July. 2023]

[13] TensorFlow. (n.d.). tf.keras.losses.BinaryCrossentropy | TensorFlow v2.10.0. [online] Available at: https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy . [Accessed 11 July. 2023]

[14] TensorFlow. (n.d.). tf.keras.losses.BinaryFocalCrossentropy | TensorFlow v2.13.0. [online] Available at: https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryFocalCrossentropy . [Accessed 14 July 2023]

[15] Lin, T.-Y., Goyal, P., Girshick, R., He, K. and Dollar, P. (2018). Focal loss for dense object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.1–1. doi: https://doi.org/10.1109/tpami.2018.2858826.

[16] scikit-learn.org. (n.d.). 1.17. Neural network models (supervised) — scikit-learn 0.23.1 documentation. [online] Available at: https://scikit-learn.org/stable/modules/neural_networks_supervised.html . [Accessed 15 July. 2023]

[17] Jason Brownlee (2017). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ . [Accessed 15 July. 2023]

[18] IBM (n.d.). What are Convolutional Neural Networks? | IBM. [online] www.ibm.com. Available at: https://www.ibm.com/topics/convolutional-neural-networks. [Accessed 16 July. 2023]

[19] Brownlee, J. (2018). When to Use MLP, CNN, and RNN Neural Networks. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/. [Accessed 16 July. 2023]

[20] Soni , D. (2021). Multi-task learning in Machine Learning. [online] Medium. Available at: https://towardsdatascience.com/multi-task-learning-in-machine-learning-20a37c796c9c. [Accessed 16 July. 2023]

[21] GeeksforGeeks. (2018). Introduction to Multi-Task Learning(MTL) for Deep Learning. [online] Available at: https://www.geeksforgeeks.org/introduction-to-multi-task-learningmtl-for-deep-learning/ . [Accessed 25 July. 2023]

[22] Varshney, N. (2020). A Primer on Multi-task Learning — Part 2. [online] Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/a-primer-on-multi-task-learning-part-2-a0f00796d0e5#:~:text=In%20the%20hard%20parameter%20sharing. [Accessed 2 Aug. 2023]

[23] SciKit-Learn (2009). 3.1. Cross-validation: evaluating estimator performance — scikit-learn 0.21.3 documentation. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/cross_validation.html. . [Accessed 2 Aug. 2023]

[24] www.cs.cmu.edu. (n.d.). Cross Validation. [online] Available at: https://www.cs.cmu.edu/~schneide/tut5/node42.html#:~:text=K%2Dfold%20cross%20validation%20is. [Accessed 14 Aug. 2023].

[25] Kampakis, D.S. (2021). What is the F-1 measure and why is it useful for imbalanced class problems? [online] The Data Scientist. Available at: https://thedatascientist.com/f-1-measure-useful-imbalanced-class-problems/. [Accessed 17 Aug. 2023]

[26] Brownlee, J. (2018). A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/. [Accessed 20 Aug. 2023]

[27] Brownlee, J. (2019). A Gentle Introduction to Learning Curves for Diagnosing Machine Learning Model Performance. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/. [Accessed 20 Aug. 2023]

[28] Tewari, U. (2021). Regularization — Understanding L1 and L2 regularization for Deep Learning. [online] Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/regularization-understanding-l1-and-l2-regularization-for-deep-learning-a7b9e4a409bf . [Accessed 27 Aug. 2023]