SPATIO-TEMPORAL ASSESSMENT OF PAIN INTENSITY THROUGH FACIAL TRANSFORMATION-BASED REPRESENTATION LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Diyala Nabeel Ata Erekat
September 2021

Spatio-temporal Assessment of Pain Intensity through Facial Transformation-based Representation Learning
By Diyala Nabeel Ata Erekat
September 2021

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Hamdi Dibeklioğlu(Advisor)

Selim Aksoy

Pinar Duygulu Şahin

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

SPATIO-TEMPORAL ASSESSMENT OF PAIN INTENSITY THROUGH FACIAL TRANSFORMATION-BASED REPRESENTATION LEARNING

Diyala Nabeel Ata Erekat
M.S. in Computer Engineering
Advisor: Hamdi Dibeklioğlu
September 2021

The nature of pain makes it difficult to assess due to its subjectivity and multi-dimensional characteristics that include intensity, duration, and location. However, the ability to assess pain in an objective and reliable manner is crucial for adequate pain management intervention as well as the diagnosis of the underlying medical cause. To this end, in this thesis, we propose a video-based approach for the automatic measurement of self-reported pain. The proposed method aims to learn an efficient facial representation by exploiting the transformation of one subject's facial expression to that of another subject's within a similar pain group. We also explore the effect of leveraging self-reported pain scales i.e., the Visual Analog Scale (VAS), the Sensory Scale (SEN), and the Affective Motivational Scale (AFF), as well as the Observer Pain Intensity (OPI) on the reliable assessment of pain intensity. To this end, a convolutional autoencoder network is proposed to learn the facial transformation between subjects. The autoencoder's optimized weights are then used to initialize the spatio-temporal network architecture, which is further optimized by minimizing the mean absolute error of estimations in terms of each of these scales while maximizing the consistency between them. The reliability of the proposed method is evaluated on the benchmark database for pain measurement from videos, namely, the UNBC-McMaster Pain Archive. Despite the challenging nature of this problem, the obtained results show that the proposed method improves the state of the art, and the automated assessment of pain severity is feasible and applicable to be used as a supportive tool to provide a quantitative assessment of pain in clinical settings.

Keywords: Pain, Facial Expression, Temporal, Visual Analogue Scale, Autoencoder, Convolutional Neural Network, Recurrent Neural Network, Deep Learning.

ÖZET

YÜZ DÖNÜŞÜMÜ TABANLI GÖSTERİM OĞRENİMİ İLE AĞRI ŞİDDETİNİN UZAM-ZAMANSAL DEĞERLENDİRİLMESİ

Diyala Nabeel Ata Erekat
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Hamdi Dibeklioğlu
Eylül 2021

Ağrının doğası, onu, yoğunluk, süre ve yeri içeren çok boyutlu özelliklerine bağlı olan öznelliği nedeniyle değerlendirmeyi zorlaştırır. Bununla birlikte, ağrıyı objektif ve güvenilir bir şekilde değerlendirebilmek, uygun ağrı yönetimi müdahalesinin yanı sıra altta yatan tıbbi nedenin teşhisi için de çok önemlidir. Bu tezde, beyana dayanlı ağrı seviyesinin otomatik ölçümünü yapan video tabanlı bir yaklaşım önerilmektedir. Önerilen yöntem, bir kişinin yüz ifadesinin benzer bir ağrı seviyesi grubundaki başka bir kişininkine dönüştürülmesinden yararlanarak etkin bir yüz gösterimi öğrenmeyi amaçlamaktadır. Ayrıca, Gözlemci Ağrı Yoğunluğu ölçeği ile birlikte Görsel Analog Ölçek, Duygusal Ölçek ve Etkin Motivasyon Ölçeği gibi beyana dayalı ağrı ölçeklerinden yararlanmanın etkisi araştırılmaktadır. Bu amaçla, kişiler arasındaki yüz dönüşümünü öğrenmek için evrişimli bir otomatik kodlayıcı ağı önerilmiştir. Otomatik kodlayıcının optimize edilmiş ağırlıkları, daha sonra, bu ölçeklerin her biri açısından tahminlerin ortalama mutlak hatasını en aza indirirken, aralarındaki tutarlılığı en üst düzeye çıkararak daha da optimize edilen uzam-zamansal ağ mimarisinin parametrelerinin baslangıç değerleri olarak kullanılmaktadır. Önerilen yöntemin güvenilirliği, videolardan ağrı ölçümü için bir kıyaslama veri tabanı olan UNBC-McMaster Ağrı Arşivi üzerinde değerlendirilmiştir. Problemin zorlu doğasına karşın, elde edilen sonuçlar, en gelişmiş yöntemleri geride bırakmakta ve ağrı seviyesinin otomatik değerlendiriminin mümkün olduğunu ve klinik ortamlarda nicel bir değerlendirme sağlamak için destekleyici bir araç olarak kullanıma uygulanabilirliğini göstermektedir.

Anahtar sözcükler: Ağrı, Yüz İfadesi, Zaman, Görsel Analog Ölçek, Otomatik Kodlayıcı, Evrişimli Sinir Ağı, Tekrarlayan Sinir Ağı, Derin Öğrenme.

Acknowledgement

First and foremost, I would like to express my gratitude to my advisor Asst. Prof. Hamdi Dibeklioğlu for his immense knowledge, never-ending patience, and guidance throughout my MSc study and research. His continuous support and understanding helped me stay focused and motivated regardless of the personal and health problems that I went through.

I also would like to thank the rest of my thesis committee: Prof. Selim Aksoy and Prof. Pınar Duygulu Şahin for their time and insightful questions. My thanks as well go to Dr. Zakia Hammal for her guidance and insights which helped me improve my academic writing and presentation skills.

I am grateful for the friendships that I got to build during my time at Bilkent and especially the "Visión Senior de Bilkent" group: Burak Mandira, Dersu Giritlioğlu, and Oğuzhan Çalıkkasap. I will always look back fondly on all the fun conversations we had and the sleepless nights we spent working on deadlines.

I am also thankful for Nathan Biles for his constant mentorship and guidance, his trust in my skills and abilities, and for all the opportunities and knowledge that he provided which allowed me to improve and grow as an engineer.

I am lucky to have my chosen family all over the globe uplifting and encouraging me; Doaa who I lean on ever since the day I got to Turkey, Farah who is one phone call away and has been there for me for 15 years, Leyza who friended me when I knew no Turkish and tolerated the google-translate phase, Mezzi who brings spice and color to my life, and lastly, my safety net in this chaotic world, and the person I look up to the most; my sister Asala. Thank you for being you.

Finally, there are no words in the English language strong enough to express my gratitude to the greatest woman I know who gave up on her dreams and hopes to see me pursue my own. I am who I am today because a strong woman raised me and showed me what it means to be confident and independent. I am and forever will be proud to be my mom's daughter. Thank you, Mom!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pain is a source of human distress, as well as a symptom and consequence of a variety of medical conditions and a factor in medical and surgical care [2]. The Visual Analogue Scale (VAS) and other uni-dimensional measures are used in the traditional clinical assessment of pain. Patients in surgical care or transient states of consciousness, as well as patients with severe cognitive problems (e.g., dementia) cannot assess their pain through the common self-reported assessment tools. [3].

Significant attempts have been made to replace the self-reported pain tools with accurate and valid methods that utilize pain indicators (e.g. facial expression) in order to assess pain [4, 5]. The most thorough method requires manual labeling of the facial muscle movements in accordance with the facial action coding system (FACS) [6, 7, 5]. The manual labeling of FACS is done by highly skilled observers which require intensive hours of training, as well as equally long hours to label a single minute of a video, rendering this method unsuitable for clinical use.

With the great advancement in deep learning and computer vision, automatic pain assessment has emerged as potential solution for objective and reliable pain assessment method. Despite VAS being the gold standard in clinical practice,

most previous efforts for automatic pain assessment have focused on frame-level pain intensity measurement consistent with the Facial Action Coding System (FACS) based Prkachin and Solomon Pain Intensity (PSPI) metric [4]. To date, using the publicly available UNBC-McMaster Pain Archive [1], only three recent studies have proposed frameworks in order to automatically assess the pain of patients in terms of Visual Analogue Scale (VAS) from videos.

For instance, Martinez and colleagues [8] proposed a two-step learning approach to estimate VAS by employing a Recurrent Neural Network which automatically estimates PSPI scores at the frame level which is later on fed into the personalized Hidden Conditional Random Fields to estimate the self-reported VAS pain scores at the sequence level. Liu et al. [9] proposed an end-to-end framework, named DeepFaceLIFT, that combines Neural Network and Gaussian Regression models in order to estimate VAS. Lastly, Szczapa *et al.*[10] expanded on the literature by proposing a geometry-based approach that utilizes the Gram matrix computation in order to model the trajectory of the facial landmarks dynamics. Self-reported Pain estimation is modeled using Support Vector Regression by computing the similarity between said trajectories.

As a contribution to previous efforts, we propose a two-step learning framework that aims to learn an efficient facial representation, at its first step, by modeling a visual encoding that can transform the facial features of a subject to another subject's facial features within very similar pain scores. To this end, a convolutional autoencoder with feed-forward convolutional layers and feed-backward deconvolutional layers is employed to learn the mapping from one subject's high-dimensional facial appearance to a lower-dimensional facial latent space, which can be used to construct another subject's facial appearance. Each frame pair is selected in a way that they are similar in terms of pain scores but different in terms of subjects. This ensures that the learned facial representation focuses on learning the carried pain information, and not the reconstruction of the identity of a specific face. Following this step, we employ a recurrent convolutional model for pain intensity measurement, where the convolutional module (AlexNet) aims to learn the spatial information in each frame of the videos, while the recurrent module (GRU) aims to learn the short-term temporal dependencies between each

consecutive frames as well as the long-term temporal dependencies between all frames across the time-space. The autoencoder from the previous step serves as an unsupervised pre-training method to the convolutional module as its encoder's trained weights are used to initialize it. The unsupervised pre-training approach improves the performance of spatio-temporal model as the spatial module has the learned knowledge of the facial representation that captures the pain information the best. Furthermore, we extend our recent work [11] that explored the effect of using multiple highly-correlated pain scales i.e., the Visual Analog Scale (VAS), the Sensory Scale (SEN), and the Affective Motivational Scale (AFF), as well as the Observer Pain Intensity (OPI) in the training by further improving the proposed custom loss function that minimizes the mean absolute error (MAE) of each scale's prediction while maximizing the consistency between each scale in a proportional manner.

For evaluation purposes, we use the benchmark database namely the UNBC-McMaster Pain Archive [1]. The effect of pre-training the spatial module on the performance of the spatio-temporal model is investigated, where we conclude that the pain-based autoencoder pre-training approach compares favorably with other pre-initialization approaches. The effect of using multiple pain scales along with enforcing consistency between them is investigated thoroughly, and it concluded that using all four different scales while enforcing the consistency between them in a proportional manner outperforms all other setups. In addition, the proposed approach performs better than all recent state-of-the-art methods given similar setup.

The rest of the chapters are organized as follows. Chapter 2 discusses the previous efforts and studies done on automatic pain estimations and their limitations. Chapter 3 goes in depth about the proposed architecture and the used methodology from the applied pre-processing steps, and the unsupervised pre-training method to the spatio-temporal model and the training loss functions. The experimental setup and dataset used is outlined in Chapter 4, along with the discussion and evaluation of the experimental results. Finally, we conclude and state any possible future work in Chapter 5.

Chapter 2

Related Work

Pain is an unpleasant feeling which indicates the possibility of an underlying condition or a potential disease. The ability to automatically monitor the patient's pain level during their stay in the hospital would provide a significant advantage in patient care and cost reduction, especially that physicians use pain as one of the main vital signs in hospitals to triage patients. The most popular technique to identify the level of pain relies on patients to self-report using uni-dimensional tools such as the Visual Analogue Scale (VAS), which is widely used due to its efficiency and ability to use in different settings. However, VAS and other subjective assessment tools have their own set of limitations; from sensitivity to suggestions and deception to the inability to measure in cases with children, infants, or patients with neurological or psychiatric impairments. Subsequently, pain is frequently misdiagnosed, inadequately treated, and neglected, particularly among marginalized groups [3]. In this chapter, we provide an overview of the significant efforts made to replace the self-reported measurements through the literature and then we dive deeply into a technical review of the currently available state-of-the-art methods that assess pain through estimating VAS.

2.1 Overview

Research focused on identifying reliable valid facial indicators of pain using the Facial Action Coding System (FACS) in order to replace self-reported measurements. Studies done by Prkachin and Solomon in 1992 [12] and the following study in 2008 [4] managed to identify the Action Units (AUs) that carried the majority of pain information.

The AUs are, as seen in Figure 2.1, brow lowering (AU4), orbital tightening (cheek raiser (AU6), and eyelid tightener (AU7)) and levator contraction (Nose Wrinkler (AU9), Upper Lip Raiser (AU10)). The relaxation of Levator palpebrae superioris muscles (AU7) represents eyes closure behavior which denotes AU43.

Using the information provided by these AUs where each is measured on a scale of 0 (absent) to 5 (maximum), except for AU43 which is measured by a binary value that indicates the behavior existence or absence, Prkachin and Solomon defined a 16-point pain metric; Prkachin and Solomon pain intensity (PSPI), as the sum of intensities of brow lowering, orbital tightening, levator contraction, and eye closure as defined in Equation 2.1.

PSPI metric is the only metric that can define pain on a frame-by-frame basis, requires manual labeling of FACS by highly trained observers. Manual labeling of a single minute of video can require several hours of effort and training which makes it ill-suited for clinical real-time use.

$$
\begin{aligned}
PSPI &= AU4 + \mathcal{M}(AU6, AU7) + \mathcal{M}(AU9, AU10) + AU43 \\
\mathcal{M}(x, y) &= \begin{cases} x, & \text{if } x > y \\ y, & \text{Otherwise.} \end{cases}
\end{aligned} \tag{2.1}
$$

With the release of publicly available pain databases (e.g., the UNBC-McMaster Pain Archive) and advancements in computer vision and machine learning, automatic assessment of pain using facial cues has emerged as a possible alternative to manual observations [2]. Most approaches to pain detection

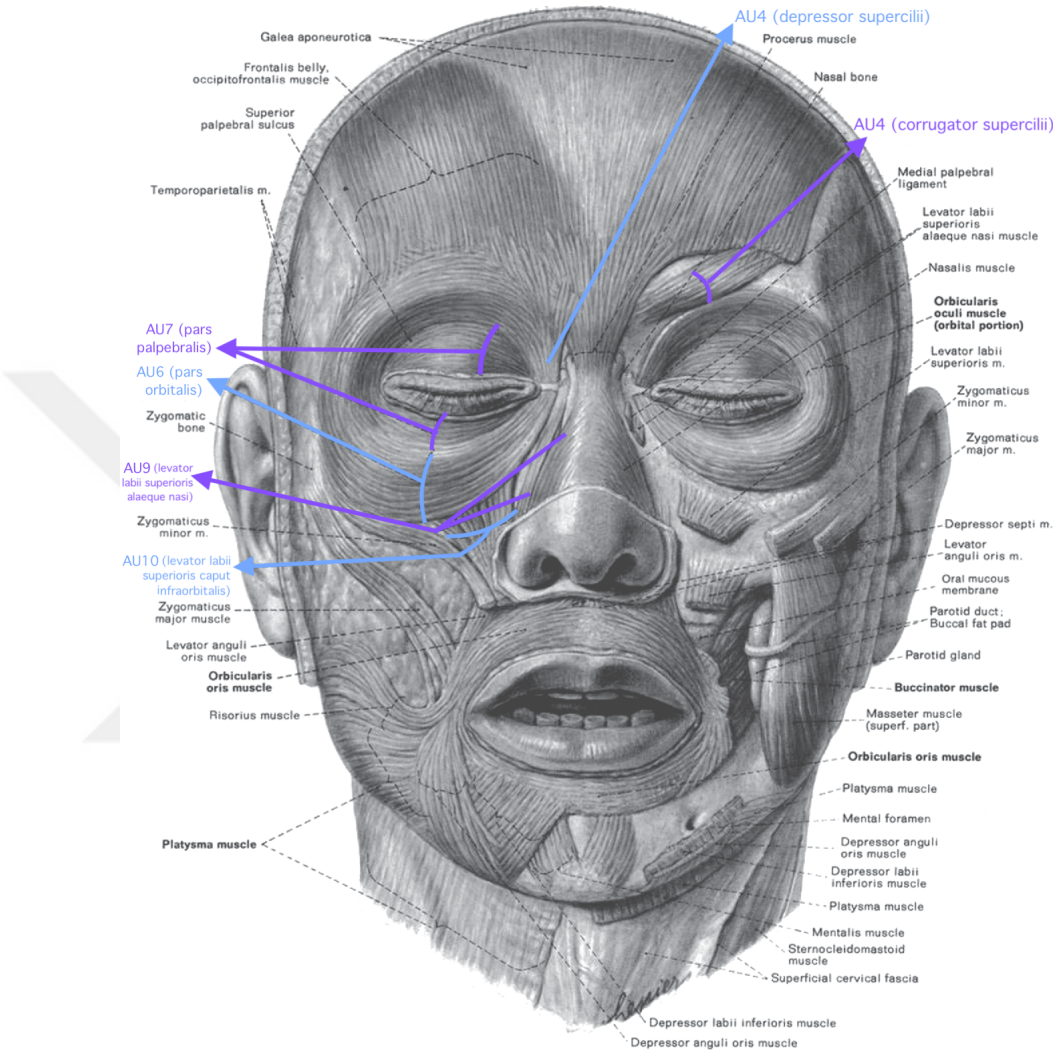


Figure 2.1: The anatomical muscle basis of facial action units used in calculating PSPI adapted from Clemente 1997 [13].

focused on identifying the presence of pain from its absence [14, 1, 15], distinguishing its genuineness [16, 14, 17] and differentiating it from other emotion expressions [18, 19, 20]. Furthermore, in terms of pain intensity detection, most previous efforts for automatic pain assessments have focused on frame-level pain intensity measurement consistent with the Facial Action Coding System (FACS) based Prkachin and Solomon Pain Intensity (PSPI) metric [21, 22, 23, 24, 25, 26, 27, 28, 29, 30].

2.2 State-of-the-art Methods

To date, despite VAS being the gold standard in clinical practice, little attention has been applied to video-based assessment consistent with the self-reported pain scores. Using the publicly available UNBC McMaster Pain Archive [1], to the best of our knowledge, only three recent studies have investigated automatic assessment of self-reported pain from videos. In this section, we will give a technical review of each paper and their proposed solution and discuss their shortcoming.

Martinez *et al.*[8] is the first study that used VAS prediction using facial cues. They build a sequence-based classification hierarchical learning framework. Their framework estimates the PSPI by employing a bidirectional long short-term memory recurrent neural net (Bi-LSTM). Then they use the estimated PSPI as an input to the hidden conditional random field (HCRF) to predict the VAS of a patient. Their work is different than other research especially because they use the individual facial expressiveness score (I-FES) to personalize the classifier. I-FES works by measuring the disagreement between the observed pain intensity (OPI) and the self-reported VAS. They model the problem based on a sequence of classification problems with a given image sequence (N_i) for patient i. The sequences of each person are annotated in terms of OPI as $O = \{O_1, \ldots, O_L\}$ where $O_i = \{o_i^i, \ldots, o_i^N\}$ with individual per-sequence OPI scores are $o_i \in \{0, \ldots, 5\}$. Similarly for VAS, $V = \{V_1, \ldots, V_L\}$, $V_i = \{v_i^i, \ldots, v_i^N\}$, and $v_i \in \{0, \ldots, 10\}$. Moreover, the features for each sequence are represented as a pair of features landmarks and PSPI scores (X, S) and they vary based on the duration T. Their approach is divided into multiple steps:

1. Obtain the estimated objective of pain intensity as encoded by the PSPI using LSTM. Because the videos are a sequence of images, LSTM is a great methodology to be used to model a sequential problem. In this work, a two-layer bidirectional LSTM is used to estimate the PSPI values from the input features. The used LSTM contains multiple blocks with one or more connected recurrent memory cells. To train the model, they use root mean square error (rMSE) loss on the PSPI score at time t. Following, the

output of the LSTM is fed into a fully connected layer with ReLU activation function.

2. Due to the challenge in estimating VAS from the facial expressions, I-FES is used to capture the OPI ratio. The intuition behind defining I-FES as in equation 2.2 is that the OPI is expected to vary significantly between patients depending on how well their facial expressions are. Thus, by using I-FES in this way, they are able to quantify those differences using the ratings, specially using the OPI rater as the reference point.

$$p_i = \begin{cases} \frac{i}{\alpha} \sum_{k=1}^{\alpha} \frac{o_i^k + 1}{v_i^k + 1}, & \text{if } \alpha > 0 \\ 1, & \text{Otherwise} \end{cases} \tag{2.2}$$

3. HCRF is used for classifying the sequential data using multi-class dynamic classification. In their case, the target VAS scores are considered the sequence labels and are represented by the top node of the graph, while the temporal states are considered the hidden variables. Equation 2.3 calculates the score functions for each class v. The personalized HCRF features S_p is calculated by multiplying the sequence of input features X and the I-FES score p_i

$$F(v, S_p, H; \Omega) = \sum_{k=1}^{K} I(k = v).f(S_p, H; \theta_k) \tag{2.3}$$

After following the steps above, the learning algorithm is configured by defining the various inputs, optimizing RNN, computing I-FES, and optimizing HCRFs. One of the unique aspects of their model is the separation between the learning of person-agnostic PSPI and person-specific VAS. Finally, they trained and evaluated their framework using the UNBC-McMaster Shoulder Pain Expression Archive Database where they got their MAE loss of 2.46 using RNN-HCRF with personalisation approach. However, the proposed method requires to be retrained on previously acquired VAS ratings from the participants and thus does not generalize to previously unseen participants.

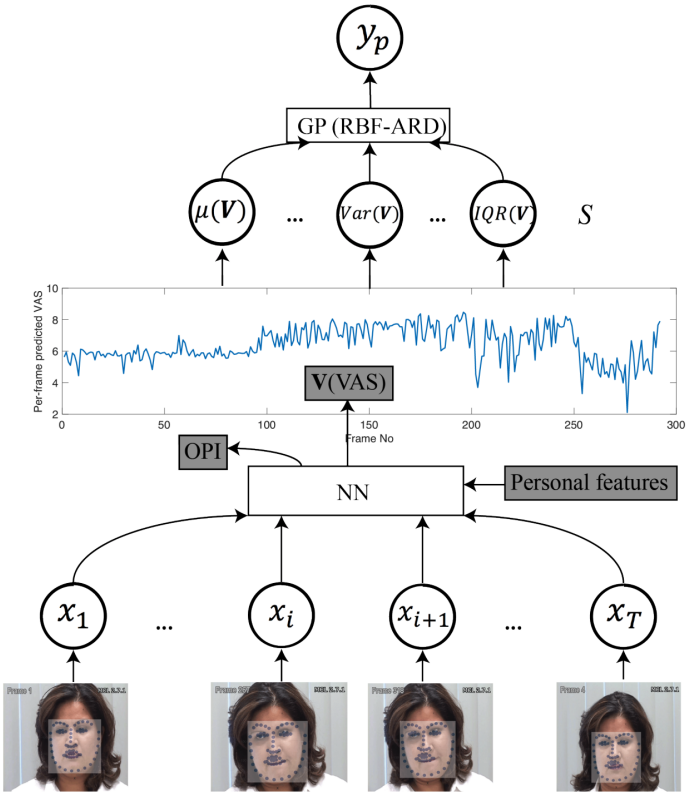


Figure 2.2: Graphical representation of DeepFaceLIFT two-stage learning model for using AAM facial landmarks (x) extracted from each frame of an image sequence to estimate sequence-level VAS (Liu 2017 [9]).

To overcome this limitation, Liu *et al.*[9] proposed another set of pre-defined personalized features (i.e., age range, gender, complexion) for automatic estimation of the self-reported VAS. DeepFaceLIFT is their proposed two-stage framework, as seen in Figure 2.2, that combines a Neural Network and Gaussian Regression Models. They manually created three additional personal features: complexion (pale-fair, fair-olive, and olive-dark), age (young, middle-aged, and elderly), and gender (male and female) from each patients' appearance. Also, similarly to the other studies, they used AAM facial landmarks rather than raw images. Their algorithm is divided into two stages:

1. Stage 1 is composed of a weakly supervised fully connected neural network which uses multi-task and multi-instance learning. The full connected neural network is composed of 4 layers with ReLU activation function and

frame-level AAM as the input. The network is trained for 100 epochs and with batch size 300 and it generates frame-level VAS scores.

2. Stage 2 introduces the Gaussian Regression Model with radial basis function ARD kernel. The outputs from Stage 1 is computed into a statistical 10-D feature vector (mean, median, minimum, maximum, variance, 3rd moment, 4th moment, 5th moment, the sum of all values, and interquartile range) for each sequence, which is later on fed into the Gaussian Model to obtain a personalized estimate of VAS.

They tested the model with its various settings on the UNBC-McMaster Shoulder Pain Expression Archive Database and compared it with NN-MV, RNN, and HCRF using ICC. Their results showed superiority when compared with other benchmark algorithms especially when using personalized features and OPI scores during training.

Szczapa *et al.*[10] is the latest study in the literature that proposed a model to estimate sequence-level VAS score. Their study expanded on the previously done work by using a geometry-based algorithm. Gram matrices formulation was used for facial points trajectory representations on the Riemannian manifold of symmetric positive semi-definite matrices of fixed rank. Also, instead of using a neural network, they used support vector regression to model the problem as manifold similar trajectories. Their framework consists of three steps:

1. Similar to the other studies, they used AAM to extract faces from the videos, where 2D facial landmark configurations are detected. However, to refine the facial extraction, a gram matrix (G) is used: $G = AA^T$, where A is the normalized facial configuration matrix. In order to refine the facial extraction more, a geometry-based model; Riemannian geometry of the space ($S^+(d,\ m)$), is introduced to track the dynamic changes in the landmarks within the different frames.

2. Then they applied a Bezier curve fitting algorithm to the trajectories to smoothen them and minimize their noise. By applying Bezier curve two

objectives are optimized and balanced: 1) the closeness of the data points at any time point, and 2) the mean squared acceleration of the curve. After applying the curve, a global alignment kernel is introduced with the support vector regression to learn the mappings between the trajectories and the pain intensities.

3. Finally, pain intensities are estimated using the support vector regression model by training it on the training set which is part of the built kernel that contains the similarity scores between all the trajectories. Moreover, the labels of the trajectories are also given to the model. In order to test the model, the mean absolute error between the ground truth and the predicted pain intensities is calculated.

They tested the model with its various settings on the UNBC-McMaster Dataset and compared their results with DeepFaceLIFT and RNN-HCRF. Their results were competitive with respect to two other state-of-art algorithms.

The previous efforts done by DeepFaceLIFT and RNN-HCRF for automatic self-reported pain measurement required an intermediate learning stage (i.e., two-step approaches). They first predicted the PSPI or the VAS at the frame level and then combined the predicted scores in a follow-up learning stage to predict pain scores at the video level. As a contribution to previous efforts, we propose a spatio-temporal end-to-end CNN-RNN model for pain intensity measurement directly from videos.

Chapter 3

Pain Intensity Prediction Framework

In this chapter, we dive into details of our video-based approach for automatic measurements of self-reported pain. Our framework as shown in Figure 3.1 consists of three phases which we thoroughly discuss in each section of this chapter. We first discuss the preliminary background information in section 3.1 about the architectures that we incorporated in our framework and explain the reasons behind choosing the convolutional autoencoder alongside with Gated Recurrent Unit (GRU) instead of other neural network variations in our framework. Then in section 3.2, we discuss thoroughly the pre-processing techniques that we used to prepare our data and make it suitable to be fed into our framework. In order to train the convolutional autoencoder and use its weights in our spatio-temporal model in the following step, we explain the method we used to create its training data and its training process in section 3.3. Lastly, we describe our spatio-temporal model in order to estimate pain intensity in section 3.4.

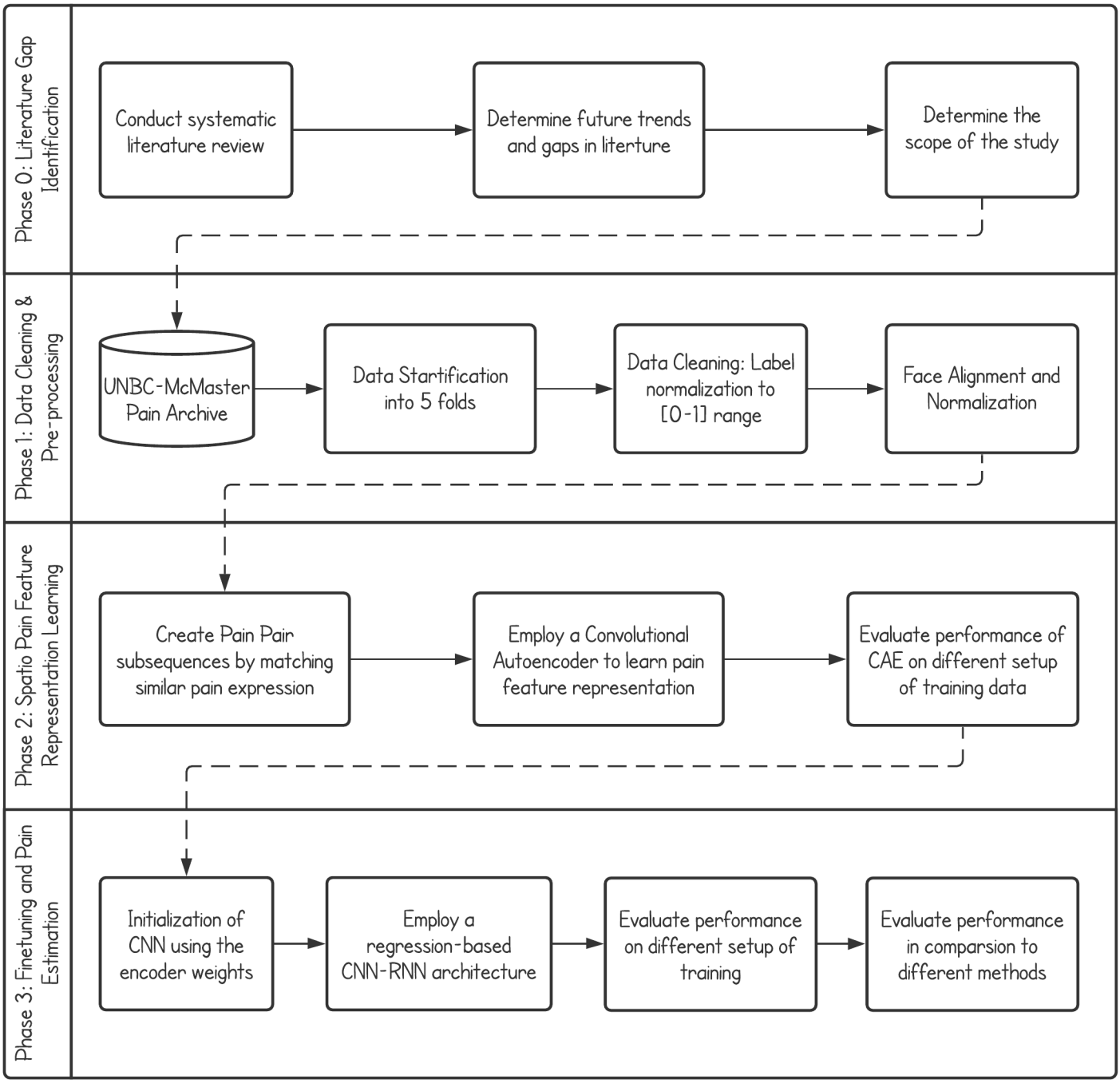


Figure 3.1: Overview of the Proposed Method

3.1 Preliminaries

3.1.1 Convolutional Autoencoder

Many of the advancements in Computer Vision field in the past few decades can be attributed to the different variations and models of neural networks (NN). The first seeds of development in neural networks started back in 1943, when a neuroscientist; Warren MuCulloch, and a Logician; Walter Pitts, tried to mimic the functionality of a biological neuron through a mathematical model [31]. Their model is widely known now as MuCulloch-Pitts model (MP Model). Rosenblatt [32] followed it up by introducing the most fundamental unit of deep neural networks; a single-layer perceptron model in the late 1950s which adds a learning ability to the earlier proposed MP model. However, the single-layer perceptrons; singular neurons, are linear models, and thus unable to handle linear inseparable problems like XOR, parity, encoding, negation, and symmetry problems. To address this, Hinton *et al.*[33] proposed a multi-layer feedforward network with a hidden layer of neurons trained by the error back-propagation [34]. The success of the back-propagation training algorithm [33, 35, 36, 37] is considered a major breakthrough in artificial Neural Networks (ANN) which set the stage for great advancements in deep learning, which foresaw the developments in convolutional neural networks (CNNs), Recurrent Neural Networks (RNNs), Autoencoders (AEs), and improvements on finding the global minima in gradient descent [38].

Rumelhart *et al.*[33] introduced the first autoencoder model when he solved the challenge of linear inseparable problems. Usually, neural networks when trained in a supervised manner have an input layer that takes the data feature and through the output layer, it encodes the feature out to a probability of certain classes, whereas autoencoders are based on an encoder-decoder paradigm as seen in Figure 3.2, where an encoder first transforms an input with dimension n, into a typically a compressed lower-dimensional and meaningful representation. The lower-dimensional representation usually captures the most essential information that is required to recreate that input again as output, and a decoder with an

output layer of dimension n that is fine-tuned to reconstruct the input back to itself using the lower-dimensional representation by minimizing the reconstruction loss.

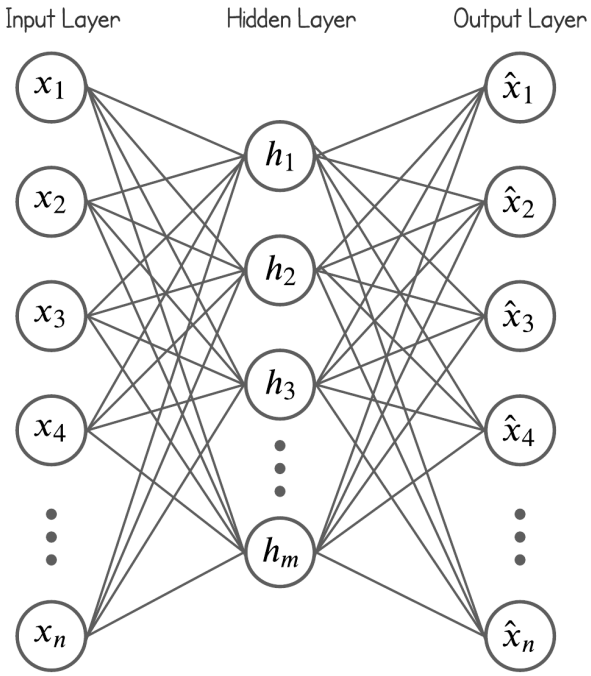


Figure 3.2: A graphical representation of an autoencoder network with n-dimensional input, n-dimensional output and m-dimensional hidden layer

Considering that autoencoder does not require any other data or labels, but the data itself, makes it a perfect model to be trained for unsupervised and semi-supervised learning. The information extracted in the lower-dimensional representation is used to solve tasks like network pre-training, feature extraction, dimensionality reduction, and clustering [39]. When the autoencoder has only one hidden layer as seen in Figure 3.2, it is called shallow or classic autoencoder. However, with the revolutionary success of deep neural networks, employing a deep autoencoder with many hidden layers in the encoder and decoder parts allows the layers to extract hierarchical features and thus producing higher-quality reconstructed inputs with lower reconstruction error [40]. The variations of deep

autoencoders are many and used for different applications including denoising [41], transforming [42], variational [43] and adversarial [44] autoencoders.

The first deep convolutional auto-encoder (CAE) is introduced by Ranzato *et al.*to learn a hierarchy of sparse feature detectors that are invariant to small shifts and distortions in an unsupervised manner [45]. Instead of fully-connected layers as in deep autoencoder, convolutional auto-encoder contains convolutional layers [46] in the encoder part and deconvolution layers [47] in the decoder part. Such networks are best suited for image processing tasks as they take advantage of convolutional neural networks' properties, considering that such networks have been achieving groundbreaking results in image-related tasks [48, 49].

The encoder part in the convolutional auto-encoder is a typical CNN, which usually consists of three different types of layers; convolutional layers, pooling layers, and fully connected (fc) layers, where each layer's output is passed as an input to the next layer. While traditional neural network layers apply transformation functions to the entire input vectors, convolutional layers [46] apply linear transformations called "convolution", hence the name, to smaller local patches of data based on the assumption that features spatially close to one another have more relevance to each other than features far from one other. A convolution is a linear operation that involves the multiplication of the input with a set of weights called filters or kernels that are slid across the input tensor, then followed by non-linear activation functions. Using multiple filters per convolutional layer allows the network to create activation maps (feature maps) that summarize the presence of these features in the input. Usually, the first layers learn low-level features such as lines, edges, and curves, whereas deeper layers in the model learn o encode more abstract features, like shapes or specific objects [50]. However, a slight translation or movement of the features in the input can result in a different activation map, making convolutional layers sensitive to feature positions. To address this, usually pooling layer is added after non-linearity has been applied to the activation maps output by a convolutional layer.

Pooling layers down-sample the activation map vector by applying an aggregating function yielding a pooled activation map that summarises the features

present in a region of the activation map generated by a convolution layer. This helps to make the representation invariant to small translations in the input, as the features close to each other will be pooled in the same location regardless. Along with increasing the robustness to minor variations in the input data, pooling layers leads to faster convergence and better generalization, as well as it controls overfitting by reducing the size of representations and thus, the total number of parameters and the computational cost in the network [51, 52, 53]. When it comes to pooling operation, it can be done by either selecting the maximum value for each patch of the activation map, resulting in a pooled activation map that contains the most prominent features of the previous activation map. This operation is called max-pooling. Another operation is average pooling which gives the average of features present in a patch, instead of the most prominent feature.

While max-pooling was originally intended for fully supervised feed-forward architectures only, Masci *et al.*[54] have investigated that the use of max-pooling in autoencoder for hierarchical feature extraction, and concluded that max-pooling layers provide the architecture with enough sparsity without needing to set any regularization parameters. The last layers of CNN are usually fully connected layers similar to other neural networks, and it applies a linear transformation to its full input vectors and follows it up by a nonlinear activation function.

The decoder part in CAE is usual the inverse of the encoder part and usually consists of three different types of layers; deconvolutional layers, unpooling layers, and fully connected (fc) layers. Considering that pooling layers down-sample the data which results in loss of spatial information that might be critical for precise reconstruction, unpooling layers are employed to solve such issue by reversing the pooling operation and reconstructing the original size of the activation map as seen in Figure 3.3. This can be done by recording the location of maximum activation (switch variables) selected during the max-pooling operation and then restoring the max-pooled features into their correct place using that recorded information.

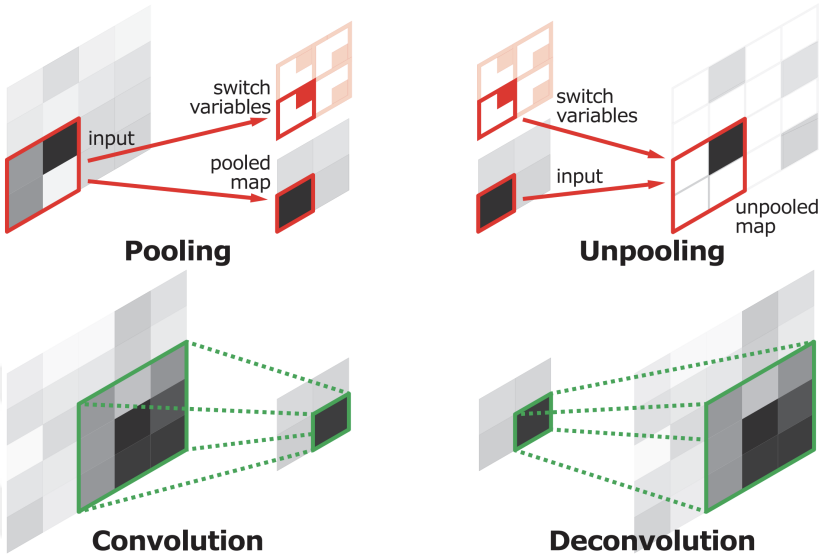


Figure 3.3: Illustration of deconvolution and unpooling layers taken from Noh *et al.*[55].

In the encoder part, the output of the convolutional network is passed to the pooling layer as an input, inversely, the output of an unpooling layer is an enlarged activation map which is passed to the deconvolutional layer as an input. Deconvolutional layer similar to convolutional layer performs convolution operation but aims to revert the spatial transformation done using the convolutional ones [56, 57]. Where the convolutional layer tries to map the input image to an encoding vector by decreasing and compressing activation maps from layer to layer, deconvolutional layers try to reconstruct the encoded vector by increasing and decompressing the activation maps from layer to layer.

3.1.2 Recurrent Models

Another extension of the feed-forward neural network is the recurrent neural network which has the ability to handle sequential or time-series data such as text, video, and speech. The ability to exploit temporal information made such architecture and its variations best suited to solve problems like machine translation [58, 59], language modeling [60], text classification [61], image and video

captioning [62, 63, 64], video analysis [65] and speech recognition [66, 67].

What makes them best suited for sequential or time-based data is the ability of the model to remember previous inputs by having a recurrent hidden state whose activation at each timestep t is dependent on its previous timestep $t-1$, which leads to sharing weights across the time dimension.

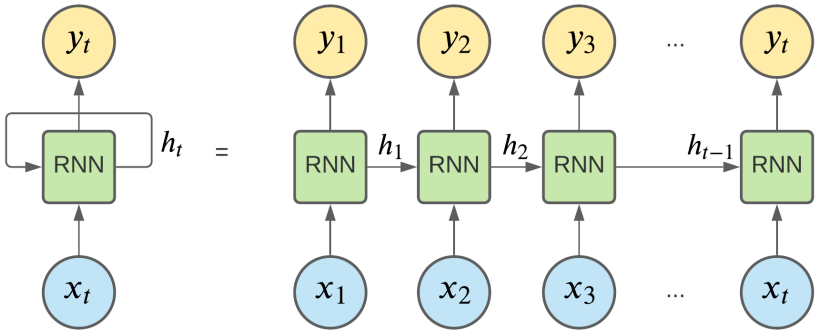


Figure 3.4: A Visualization of the Recurrent Neural Network

To elaborate more, given an input of a sequence data denoted as $\{x_1, x_2, \ldots, x_t\}$ where x_t is a data point at timestep t, each recurrent neuron receives input from current data point x_t, while also receiving input from the hidden node of the previous timestep state h_{t-1}. Each hidden state is updated through a nonlinear activation function ϕ as follow:

$$h_t = \begin{cases} 0, & \text{if } t = 0 \\ \phi(h_{t-1}, x_t), & \text{otherwise.} \end{cases} \tag{3.1}$$

The output of the RNN architecture denoted as Y varies based on its type as seen in Figure 3.5. For example, a one-to-many RNN takes one-timestep input $X = x_1$ and generates a sequence output denoted as $Y = \{y_1, y_2, \ldots, y_t\}$ as in music generation and image captioning. Similarly, many-to-one takes multiple-timestep inputs; $X = \{x_1, x_2, \ldots, x_t\}$ and generates one output at the last timestep $Y = y_t$ as used in sentiment analysis. A many-to-many takes in a sequence input and generates a sequence output as well, as can be seen in machine translation and frame-by-frame video labeling.

Vanilla RNN suffers from exploding and vanishing gradients [68, 69], the exploding gradient problem can be easily handled using a technique called gradient clipping which aims to shrink the gradients whose norms exceed a certain threshold [70, 60]. However, the vanishing gradients in RNN are more challenging to handle and hinder the RNN's ability to learn long-term dependencies, as the gradient becomes smaller with each timestep update making it insignificant.

Long Short Term Memory (LSTM) [69] and Gated Recurrent Unit (GRU) [59] which are commonly used RNN architectures address this problem by introducing specific gates with a well-defined purpose. Similar to RNN, LSTM which was introduced back in 1997 takes an input of x_t and h_{t-1}, but has introduced a new state called network cell state which is denoted by c_t. The network cell state value is controlled and updated by three gates; forget gate, input gate, and output gate. All the gates use hyperbolic tangent and sigmoid activation functions. The network cell state is responsible for aggregating the data from all previous time-steps in some sort of encoding to maintain long-term dependencies, whereas the hidden state h_t focuses on the previous time-step $t-1$ and thus maintaining short term dependencies. The forgot gate as its name implies controls what information in the cell state to forget, given any new information introduced in the network, whereas the input gate controls what new information to encode. The output gate controls what information to encode and send to the network in the next timestep.

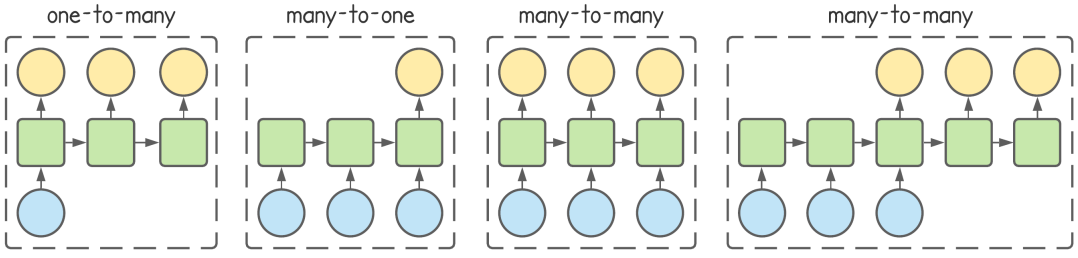


Figure 3.5: A Visualization of Different Types of RNNs

Similarly to LSTM, GRU which was introduced by Cho *et al.*[59] in 2014 addresses the vanishing gradient problems, however instead of having a network

cell state to regulate information, GRU uses the hidden states along with two gates; reset and update gates, to control the flow of information inside the unit instead of the three gates as in LSTM. Base on multiple studies, GRU outperforms LSTM on smaller datasets [71, 72, 73] and is less complex and computationally more efficient in terms of training [73].

3.2 Face Alignment and Normalization

The ability of the human mind to detect and analyze faces is effortless; for example, we can recognize the faces of the people we know, their state of emotion, and even their behavioral intention whether it is genuine or deceitful, satisfactory or disapproving with little effort or none at all.

In order to transfer this ability to machines to analyze faces accurately, the Automatic Face Analysis Domain has gained extensive attention in Computer Vision Research Groups. The domain itself is considered big enough that it consists of many sub-domains each addressing a specific sub-problem. For instance, *face detection* is the process of determining whether a face exists in a scene or not [74, 75], whereas *face alignment* is the process of determining the face shape, i.e. the location of characteristic facial features or landmarks [76]. *Face localization* is the process of determining the face's position in that scene [77] and the process of comparing that localized face against a database in order to identify and verify an identity is called *face recognition* [78]. *Facial feature detection* aims to detect and localize facial features of a person from eyes, ears, nose, mouths, etc [79, 80], whereas *facial expression recognition* identifies the emotional state and behavioral intentions of that person.

In addition to how big this domain is, it is also considered to be a challenging domain due to varying appearances of faces [81]. These challenges include, but are not limited to 1) *illumination variations* formed due to lighting and camera characteristics can cause a face to look dramatically different from one condition to another, 2) *variation of poses* causes the face to differ from a different point

of view; frontal, profile, 45^o degree, along with introducing self-occlusion when some features; eyes, nose or mouth become partially or completely occluded, 3) *occlusion* can make face detection problem difficult as the faces are partially occluded by other objects and even if the face is detected, recognition may still be difficult due to some hidden facial features, 4) *hairstyles* can hide facial features whereas and *facial hair* can alter one's appearance, especially the features around the lower half of the face [82, 83, 84].

Given our defined problem, we want to focus on the pain information carried specifically through the facial cues. Thus, to ensure that the facial expressions are the dominant source of variance in our input images regardless of the subject, we aim to remove the individual shapes of faces through shape normalization and so, creating shape-free faces [85, 86, 87] as can be seen in Figure 3.6. This technique is often referred to as 2D Face Morphing, where all faces are warped based on Delaunay triangulation to a standard shape, in our case, the standard shape is the average face of the whole database, resulting in new faces that each had the same reflectance as its original face, but the shape of the average face.

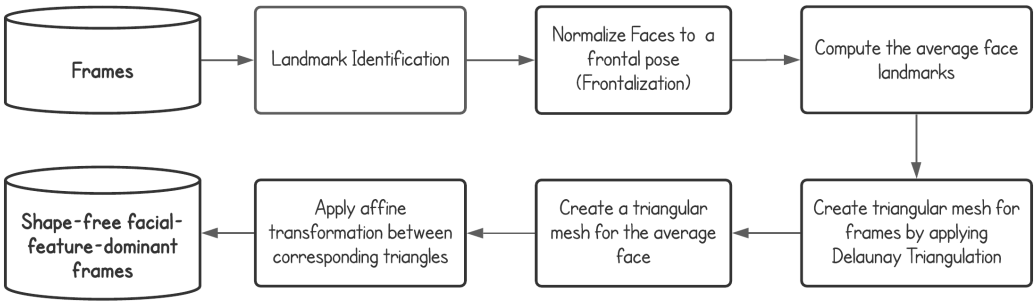


Figure 3.6: Step-by-step Face Warping Algorithm

The first step into computing the average face aims to correct the face pose of each frame by normalizing the input face images in terms of rotation and scale to a predefined pose (frontal) to capture the maximum amount of information from the features, considering that the setup for the camera orientation in the UNBC-McMaster Pain Archive was frontal but had changes in poses.

The used normalization algorithm is fairly straightforward and relies on the inter pupil coordinates to obtain a normalized rotation and translation of the face by aligning the faces such that the inner pupils of the eyes are on the same horizontal line. Given that we have 66-point AMM landmarks for each frame f in the database such as; $L_f = [(x_f^1, y_f^1), (x_f^2, y_f^2), \ldots, (x_f^{66}, y_f^{66})]$. We can obtain the coordinates of the eyes, such that the right eyes coordinates are $L_f^{re} = [(x_f^{37}, y_f^{37}), \ldots, (x_f^{42}, y_f^{42})]$ and the left eyes coordinates are $L_f^{le} = [(x_f^{43}, y_f^{43}), \ldots, (x_f^{48}, y_f^{48})]$. Using the eye coordinates, we obtain the inner pupils coordinates by computing centroid of each eye. The angle between the two centroids is the angle of rotation which we use for the rotation matrix. After computing the rotation matrix, we apply the affine transformation on the images to align the faces. This transforms the 66-AAM landmark points as well, such as $\widehat{L_f} = [(\widehat{x_f^1}, \widehat{y_f^1}), (\widehat{x_f^2}, \widehat{y_f^2}), \ldots, (\widehat{x_f^{66}}, \widehat{y_f^{66}})]$.

Next, we compute the average face A of the whole database, such that: $\widehat{L_A} = [(\widehat{x_A^1}, \widehat{y_A^1}), (\widehat{x_A^2}, \widehat{y_A^2}), \ldots, (\widehat{x_A^{66}}, \widehat{y_A^{66}})]$ by averaging points of the transformed landmarks as follows:

$$\widehat{L_A^i} = \frac{1}{F} \sum_{f=1}^{F} (x_i, y_i) \quad \text{for } i = 1, \ldots, 66 \tag{3.2}$$

After that, we create the triangular mesh for our input data by applying Delaunay Triangulation on every frame f using its corresponding set of points $\widehat{L_f}$, as well as on the average face points $\widehat{L_A}$. Since all the frames and the average face have the same number of coordinates, we end up with the same number of triangles for each mesh which offers us triangle-to-triangle correspondences between two sets of facial features. The average face triangular mesh serves as our target output, whereas each frame's triangular mesh aims to warp toward it.

Given that the average face triangular mesh is $[t_a^1, t_a^2, \ldots, t_a^n]$ and the triangular mesh for each frame f is $[t_f^1, t_f^2, \ldots, t_f^n]$ where n is the number of triangulars, then for every frame f, we can compute the affine transformation matrix M_i that is required to transform the 3 vertices from frame's triangle t_f^i to the 3 vertices from

the average mesh's corresponding triangular t_a^i, such as:

$$M_i = T(t_a^i, t_f^i), \tag{3.3}$$

Using the calculated piece-wise affine transformation matrices M, the warping is achieved by translating each pixel in the source input image to its corresponding coordinate forming the final warped face output. This kind of normalization (pixel-to-pixel) ensures that our face images are comparable to each other regardless of identity. The new normalized faces are then cropped out by forming a mask with the convex hull of the landmark points resulting in images of size (224×224) as seen in Figure .

(a) (b) (c) (d)

Figure 3.7: Face Normalization. (a) Original Face. (b) Facial Landmarks. (c) Triangulation. (d) Warped Face [1].

3.3 Autoencoder-Based Representation Learning

Taking advantage of the autoencoder ability to learn a lower-dimensional representation that captures the most essential information on the data, and inspired by the work done by Dibeklioglu [88] where they employed a Siamese-like coupled convolutional autoencoder network that aims to learn the facial representation between kin pairs in order to verify kinship between two subjects, we similarly explore the ability of the model to learn efficient facial representation by learning the transformation encoding between two subjects within similar pain group. In this section, we first discuss how we create the training data for our autoencoder that enables it to learn the efficient pain information from the facial features, and then we follow it up with how our autoencoder model is trained and optimized.

3.3.1 Pain Expression Matching

To build up our pairs directory, we first divided our videos into 4 categories; no pain (VAS = 0), mild ($1 \leq$ VAS ≤ 3), moderate ($4 \leq$ VAS ≤ 6) and severe (VAS ≤ 7), as we want to have pairs from different subjects, but we don't have enough sequences per label to create sufficient pairs. Assuming that the pain facial expression within one pain category is similar across different subjects, we present a method that learns the efficient pain features by exploiting the transformation between sub-sequences within one pain category.

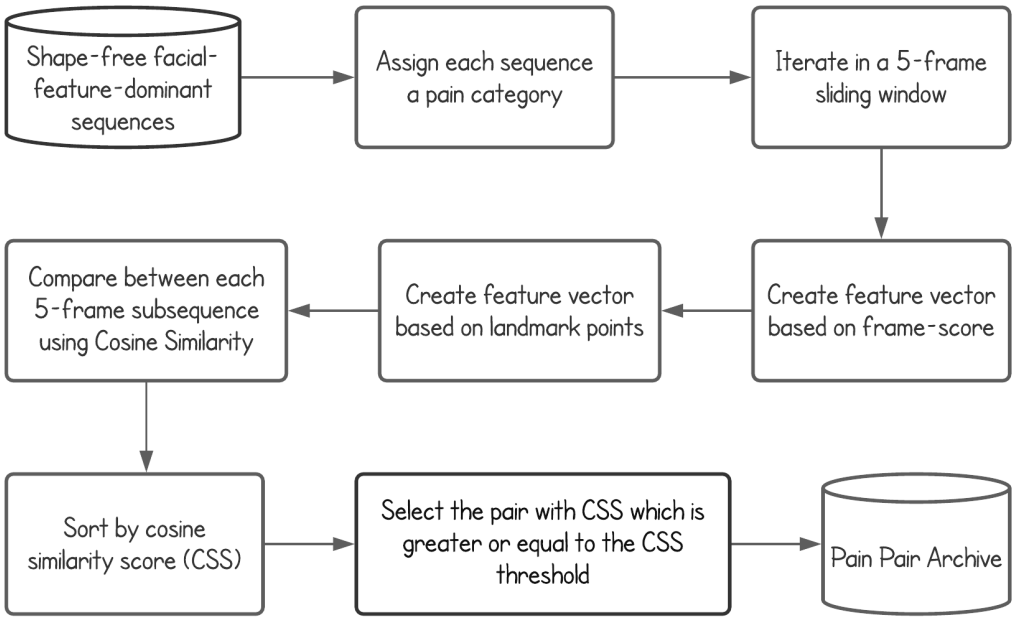


Figure 3.8: Step-by-step Pain Expression Matching

Following this assumption, we build pain pairs where each pain pair $\mathcal{P}_i^p$ of pain category p consists of two sub-sequences ($\mathcal{K}1_i^p$, $\mathcal{K}2_i^p$) from two different subjects. Using the whole sequence as a whole would require higher computational time to train, and using one frame at a time would increase the frame noise, and thus we wanted to maintain the temporal information across time without increasing the complexity while eliminating frame noise and ensuring consistent matching,

and so each category would consist of N number of 5-frame sub-sequences which can be matched to another sub-sequence within the category where the similarity between them is the highest.

For each sub-sequence, we represented in two feature vectors. The first one, $\mathcal{V}_{\mathcal{K}}$, represents the PSPI score for each frame f in sub-sequence $\mathcal{K}$, such as; $\mathcal{V}_{\mathcal{K}} = [s_t, s_{t+1}, \dots s_{t+4}]$ where s_t denotes the PSPI score for that frame at t timestep. The second feature vector, $\mathcal{J}_{\mathcal{K}}$, represents the normalized landmarks for each frame f in $\mathcal{K}$, such as; $\mathcal{J}_{\mathcal{K}} = [\overline{L_t^{\mathcal{K}}}, \overline{L_{t+1}^{\mathcal{K}}}, \dots \overline{L_{t+4}^{\mathcal{K}}}]$, where $\overline{L_t} = [(\overline{x_t^1}, \overline{y_t^1}), (\overline{x_t^2}, \overline{y_t^2}), \dots, (\overline{x_t^{66}}, \overline{y_t^{66}})]$ denotes the landmarks at frame f at timestep t.

Using these two feature vectors, the Cosine Similarity Score (CSS) between two sub-sequences within one pain category denoted as $\mathcal{S}$ is obtained by computing the cosine similarity ($\mathcal{C}$) between the corresponding feature vectors of $\mathcal{K}1^p$ and $\mathcal{K}2^p$, such as:

$$\begin{aligned} \mathcal{S}\left(\mathcal{K}1, \mathcal{K}2\right) &= \frac{\mathcal{C}\left(\mathcal{V}_{\mathcal{K}1}, \mathcal{V}_{\mathcal{K}2}\right) + \mathcal{C}\left(\mathcal{J}_{\mathcal{K}1}, \mathcal{J}_{\mathcal{K}2}\right)}{2} \\ \mathcal{C}\left(p, q\right) &= \frac{p \cdot d}{\parallel p \parallel \times \parallel d \parallel} \end{aligned} \tag{3.4}$$

After that, we sort all pairs by the cosine similarity score in a descending order, and then we add only the pairs that are with a CSS which is greater or equal to a threshold $\mathcal{T} = 0.75$.

3.3.2 Pain-based Pre-training

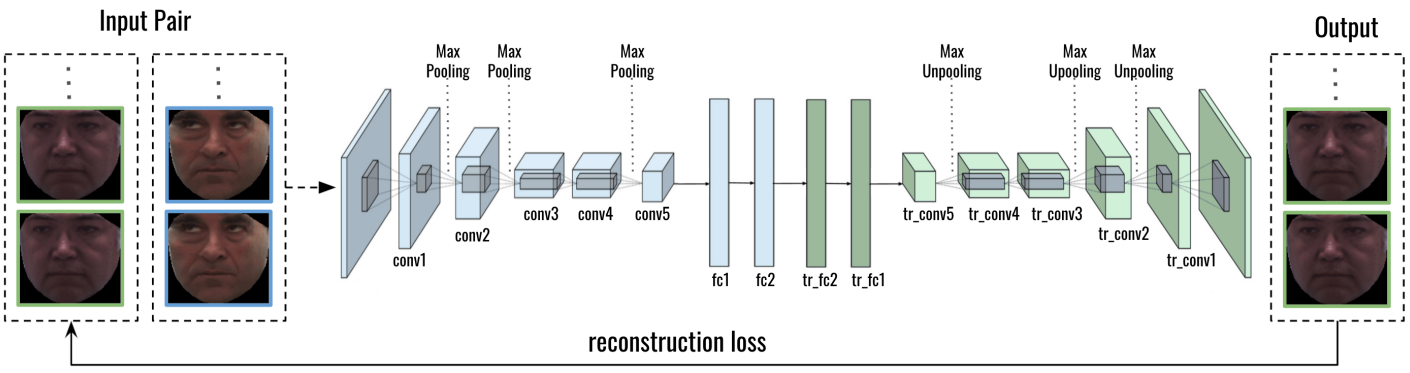


Figure 3.9: Overview of the proposed method for learning pain features mapping of facial appearance between pain pairs.

To achieve our goal of learning a facial representation that captures the pain information and patterns while taking advantage of state-of-art deep learning architectures, we build an AlexNet-like autoencoder. AlexNet, which is considered a milestone that promoted the rapid development of the modern deep convolutional network, was developed by Alex *et al.*back in 2012 and achieved the best classification result at that time using deep CNN in the ImageNet Large Scale Visual Recognition Challenge (LSVRC) [48].

As shown in Figure 3.9 and Table 3.1, we employ the AlexNet for the encoder convolutional part with its last classification layer removed. Our convolution network has five convolutional layers altogether, three max-pooling operations are performed between some of the convolutions and two fully connected layers that aggregate the information obtained from all neurons from the third max-pooling. The decoder network is the mirrored version of encoder one and is composed of five deconvolutions, three max-unpooling layers, and two fully connected layers. Contrary to encoder network that reduces the size of activations through feed-forward convolutional, decoder network enlarges the activations through the combination of unpooling and feed-back deconvolution operations.

Table 3.1: Configuration of AlexNet-based Autoencoder network. "conv", "deconv", "pool", "unpool", "fc" and "trfc" denote convolution, deconvolution, max-pooling, max-unpooling, full connection and transposed full connection layers in the network, respectively. Numbers next to the name of each layer indicate the order of the corresponding layer.

Layer	Kernel Size	Stride	Output Size	Activation
Input	-	-	$224 \times 224 \times 3$	-
conv1	11×11	4	$55 \times 55 \times 96$	ReLU
pool1	3×3	2	$27 \times 27 \times 96$	-
conv2	5×5	1	$27 \times 27 \times 256$	ReLU
pool2	3×3	2	$13 \times 13 \times 256$	-
conv3	3×3	1	$13 \times 13 \times 384$	ReLU
conv4	3×3	1	$13 \times 13 \times 384$	ReLU
conv5	3×3	1	$13 \times 13 \times 256$	ReLU
pool3	3×3	2	$6 \times 6 \times 256$	-
fc6	-	-	$1 \times 1 \times 4096$	ReLU
fc7	-	-	$1 \times 1 \times 4096$	ReLU
trfc7	-	-	$1 \times 1 \times 4096$	ReLU
trfc6	-	-	$1 \times 1 \times 9216$	ReLU
unflatten	-	-	$6 \times 6 \times 256$	-
unpool3	3×3	2	$13 \times 13 \times 256$	-
deconv5	3×3	1	$13 \times 13 \times 384$	ReLU
deconv4	3×3	1	$13 \times 13 \times 384$	-
deconv3	3×3	1	$13 \times 13 \times 256$	ReLU
unpool2	3×3	2	$27 \times 27 \times 256$	-
deconv2	5×5	1	$27 \times 27 \times 96$	ReLU
unpool1	3×3	2	$55 \times 55 \times 96$	-
deconv1	11×11	4	$224 \times 224 \times 3$	ReLU

To learn efficient coding between two pain pairing frames, the autoencoder is trained in greedy, layer-wise fashion, and subsequently, the weights are fine-tuned using back-propagation [89, 90, 91] by exploiting the visual similarity between the frames in the image space using cosine embedding loss. Cosine embedding loss is usually used to learn nonlinear embeddings and measure the similarity/dissimilarity between two tensors.

The model is trained pair-by-pair frame-by-frame on the obtained pair pain archive as explained in the previous section with a distribution shown in Table 4.1, and given that $\mathcal{P}$ denotes the input pair ($K1$, $K2$) where $K1 = \{I_{1,1}, \ldots, I_{1,5}\}$ denotes the input fed to the encoder and $K2 = \{I_{2,1}, \ldots, I_{2,5}\}$ denotes the expected output of the decoder, and $\hat{K}_1 = \{\hat{I}_{1,1}, ..., \hat{I}_{1,5}\}$ denote the actual corresponding output of the image sequence from the decoder. If Θ is the cosine embedding distance between two images, then the cost function in order to learn the transformation of facial appearance between pain pairs by minimizing the distance between $\mathcal{K}_2$ and $\hat{\mathcal{K}}_1$, as follows for each frame i:

$$\ell_{\text{cost}} = \Theta(\hat{I}_{1,i}, I_{2,i}) \ . \tag{3.5}$$

3.4 Spatio-Temporal Model

Given the nature of our data, we want to take advantage of the spatial information presented in each frame of a sequence, while exploiting the temporal information between these frames, and to that end, we propose a hybrid architecture as illustrated in Figure 3.10 which consists of two modules; a convolutional module for spatial features modeling and a recurrent dynamic module to learn the facial dynamics. Both modules are integrated into an end-to-end trainable system. This Recurrent Convolutional Network architecture is not new in the literature, while it is best suited for modeling sequence of images as in videos [92, 93], it has also been used for multi-label image classification as proposed in Wang *et al.*[94], image captioning [95] and text analysis by feeding the word embedding vectors into the convolutional module [96].

To leverage the pain information learned by training the convolutional autoencoder on reconstructing similar pain features as explained in Section 3.3, the pre-trained autoencoder weights are used while the decoding blocks are removed. At each time step of a given video of size N frames, aligned face image in the target frame with a size of $224 \times 224 \times 3$ pixels, is fed to the CNN network as an input. A 4096-dimensional mid-level spatial representation is generated from the second fully-connected layer and is aggregated as an input to the corresponding time step of the recurrent model. In the backward pass, the parameters in the convolutional module are optimized by back-propagating output gradients of the upper recurrent module through all frames.

We employ a two-layer many-to-one gated recurrent network (GRU) with 1024 hidden units (at each layer) as our recurrent module. The depth of the recurrent network is selected based on our preliminary experiments where a 2-layer recurrent network has outperformed both a single-layer and a 3-layer recurrent network on multiple settings. As discussed in section 3.1.2, GRU is less complex and better suited for problems with a smaller dataset to avoid overfitting.

At each time step of the input video (of size N frames), the corresponding output of the convolutional module is fed to the GRU model, and a 1024-dimensional temporal representation is computed. The model process one video at a time and handle videos of different duration (see Figure 4.2(b)). In order to capture the dynamics of the whole sequence (input video), the representation obtained after the last time step (i.e., after processing the whole video) is passed to a multivariate linear regression layer to estimate the pain intensity of the corresponding video-sequence. Because pain scores are continuous and not independent, a regression function is used instead of multi-class classification.

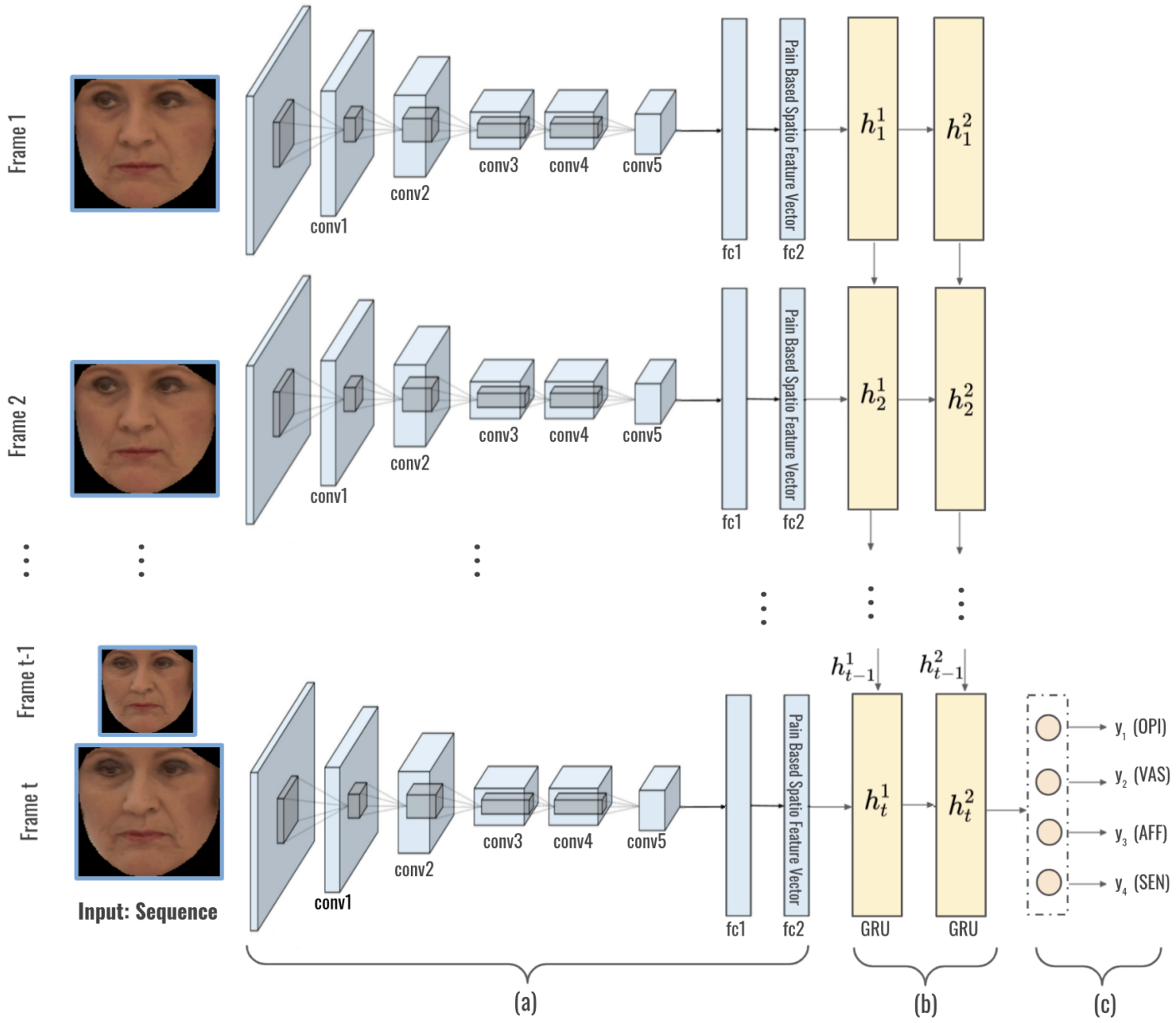


Figure 3.10: The graphical representation of our end-to-end pain estimation model. (a) The CNN model initialized using pain-based encoder weights and is trained to learn frame-by-frame spatial features (4096-D per frame) and (b) The 2-layer GRU model trained to learn per-video temporal dynamics of facial features. (c) The multivariate regression model to estimate pain intensity scores consistent with the VAS, SEN, AFF, and OPI independently and in combination [1].

3.4.1 Regularization

To avoid overfitting, we employed three different regularization methods. L2 norm regularization is used to penalize complexity as it reduces the model weights after each epoch by value. The smaller the weights are, the harder it is for the network to learn local noise, and thus, the network is forced to learn commonly seen features across the training set. Another method we employed is dropout regularization [97], where a random number of neurons along with their connections are dropped during training. This prevents co-adapting and forces the model to use a different set of neurons for each training iteration. Our convolutional Module (AlexNet)controls the complexity of its fully connected layers using a dropout value of 0.5, which is also the same dropout value for the output of the intermediate layer in our two-layer GRU module. The last method of regularization that we used is early stopping which is applied by monitoring the validation loss over a fixed number of epochs (in our training, it is set to 7). During training, we save the model's weights that minimize the validation loss at a certain epoch. If beyond that point the validation loss doesn't improve or worse decreases after a grace period, the training process is halted to prevent further overfitting.

3.4.2 Pain Estimation Loss Function

To model the intensity of pain from facial features, we exploit the relevance and relation of pain scores collected using multiple self-reported pain scales and observer pain scale (see section 4.1). To this end, similarly to what we proposed in our recent work [11], we propose a further improved custom loss function as follows:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{MAE}} + (1 - \alpha) \cdot \mathcal{L}_{\text{wC}} + \lambda \cdot ||\hat{W}||_2 \ , \tag{3.6}$$

where $\hat{W}$ denotes the weight parameters of the deep model. Let $\mathcal{L}_{\text{MAE}}$ indicates the Mean Absolute Error (MAE) as:

$$\mathcal{L}_{\text{MAE}} = \frac{1}{n \times m} \sum_{i}^{n} \sum_{j \in S} \left| y_i^j - \hat{y}_i^j \right| \ , \tag{3.7}$$

and, $\mathcal{L}_{\text{wC}}$ is the loss that measures the consistency in pain scores obtained using different pain scales (e.g., consistency between the predicted self-reported and observed pain scores). $||\hat{W}||_2$ indicates the $L2$ norm of the weight parameters, α is the trade-off parameter (between $\mathcal{L}_{\text{MAE}}$ and $\mathcal{L}_{\text{wC}}$), λ is the regularization rate, and y_i^j, $\hat{y}_i^j$ denote the actual and predicted pain scores of the ith video using the jth pain scale, respectively. While n represents the number of training videos, S is the set of predefined pain scales, i.e., $S = \{\text{VAS}, \text{AFF}, \text{SEN}, \text{OPI}\}$, and m is the size of S (see section 4.1 for the definition of pain scales).

As shown in Equation 3.6, our loss definition consists of three terms. The first term $\mathcal{L}_{\text{MAE}}$, aims to minimize the difference between the predicted scores and their actual values (in each pain scale). In our previous work [11], the second term, namely consistency term, $\mathcal{L}_{\text{C}}$ enforces the consistency of the predictions from different pain scales (i.e., VAS, AFF, SEN, and OPI). The assumption behind that is that each pain scale is highly correlated to any other pain scale, and thus, they should be relatively similar in magnitude when normalized between [0-1]. Based on that assumption, each scale's deviation (σ) for each video is computed, and the average of the deviation values of the training samples is used as the consistency loss as follow:

$$\mathcal{L}_{\text{C}} = \frac{1}{n} \sum_{i}^{n} \sigma \left(\hat{Y}_i \right) \ , \tag{3.8}$$

where, $\hat{Y}_i$ denotes the predicted labels for the ith video, particularly $\hat{Y}_i = \{y_i^j \mid j \in S\}$, n represents the number of training videos and S is the set of predefined pain scales, i.e., $S = \{\text{VAS}, \text{AFF}, \text{SEN}, \text{OPI}\}$.

The shortcoming of the previous proposed loss function is that it assumed that all scales are equally proportional to one another, however in fact, each pain scale a is proportional to another pain scale b by a factor of $w_{a,b}$, and respectively, b is proportional to a by a factor of $w_{b,a}$. The proportional weights between each scale are computed such as:

$$w_{a,b} = \frac{\sum_i^n y_i^a}{\sum_i^n y_i^b} \ . \tag{3.9}$$

To take advantage of the proportional ratio between scales, we incorporate the weights in our previously proposed loss function resulting in a new term $\mathcal{L}_{\text{wC}}$, namely weighted consistency term, which is used in our custom loss as seen in Equation 3.6. The weighted consistency term is defined such as:

$$\begin{aligned} \mathcal{L}_{\text{wC}} &= \frac{1}{2 \times n \times p} \sum_{i}^{n} \sum_{(a,b) \in \mathcal{R}} \mathcal{D}(y_i^a, y_i^b) \ , \\ \mathcal{D}(y_i^a, y_i^b) &= \sigma(w_{a,b} \cdot y_i^a, y_i^b) + \sigma(y_i^a, w_{b,a} \cdot y_i^b) \ , \end{aligned} \tag{3.10}$$

where, y_i^a and y_i^b denote the predicted labels for the ith video for pain scale a and pain scale b respectively. $\mathcal{R}$ denotes $\binom{S}{2}$ which is the combination without repetition of the pain scales taken 2, and p is the number of combinations. Therefore, if the predicted pain scores from different pain scales vary highly in magnitude with respect to their proportions, the inter-class variability increases, and thus, loss increases penalizing the difference. Please notice that we normalize the scores of each rating scale to a range of $[0, 1]$ so as to provide comparability between them in the training process. Lastly, the third term of our loss definition is the $L2$ regularization term.

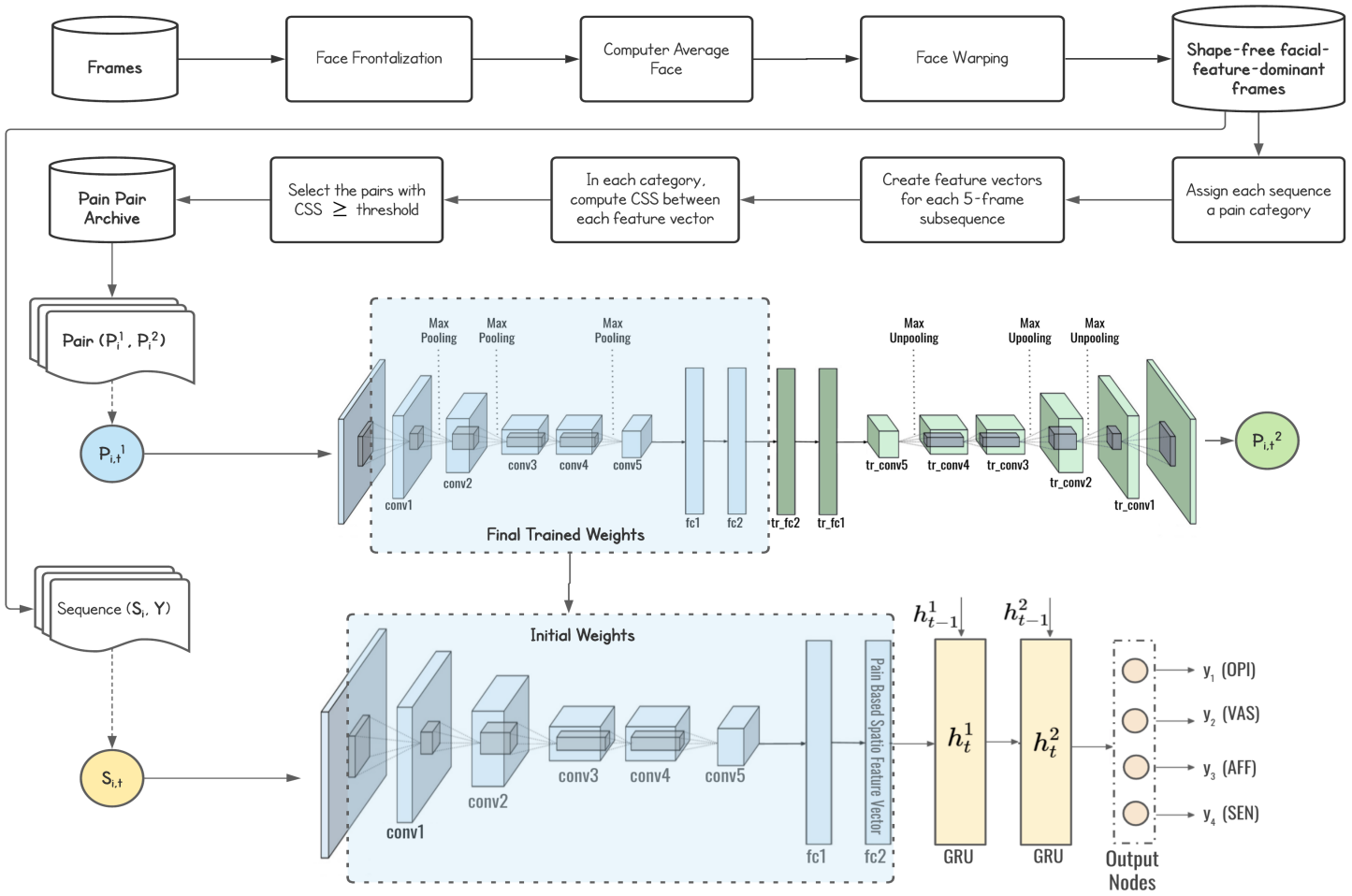


Figure 3.11: Thesis High-level Technical Methodology

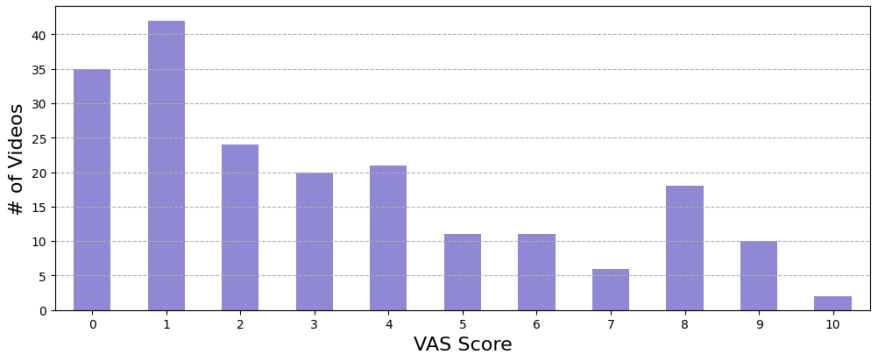
Chapter 4

Experiments and Results

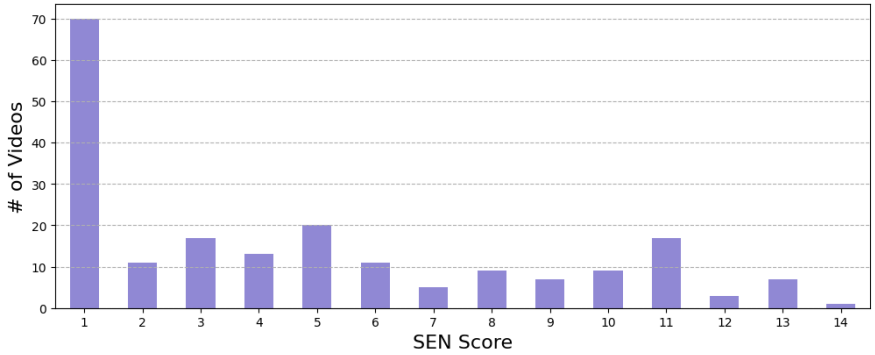
4.1 Dataset: the UNBC-McMaster Pain Archive

UNBC Database [1] was the first effort to address the need for well-annotated facial expression recordings of participants with acute pain. Participants, self-identified as having a problem with shoulder pain, were recruited from physiotherapy clinics and by advertisements. Participants suffered from arthritis, bursitis, tendonitis, subluxation, rotator cuff injuries, impingement syndromes, bone spur, capsulitis, and dislocation. Researchers at the McMaster University and University of Northern British Columbia (UNBC) captured videos of the participants' faces during abduction, flexion, and internal and external rotation of their affected and unaffected shoulders.

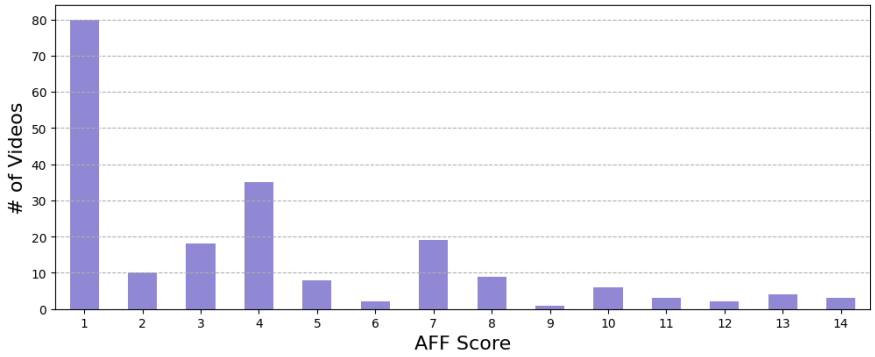
The public dataset contains 200 sequences from 25 different participants with varying duration and a total number of 48,398 frames. Each frame was fully AU coded by certified FACS coders. Figure 4.2 shows the number of videos available per participant as well as the mean duration of videos per participant.



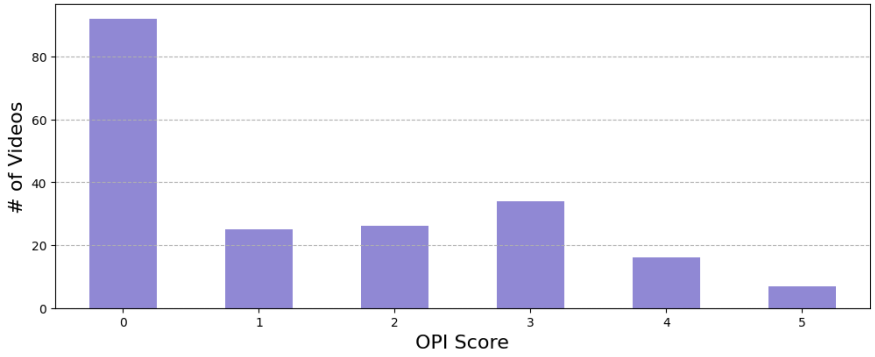
(a)



(b)

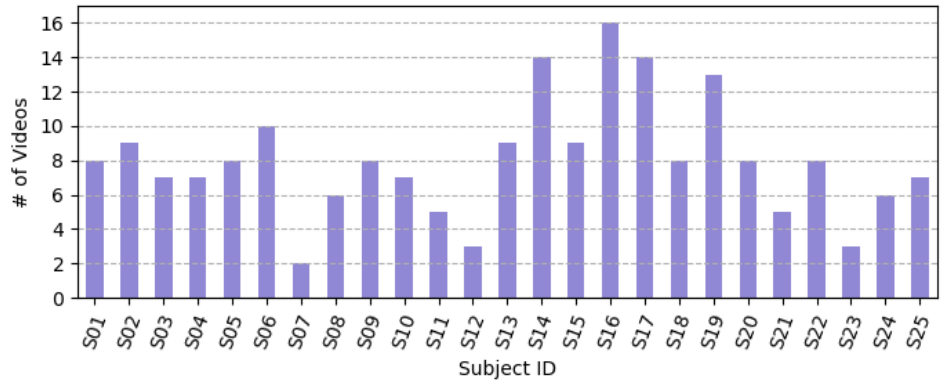


(c)

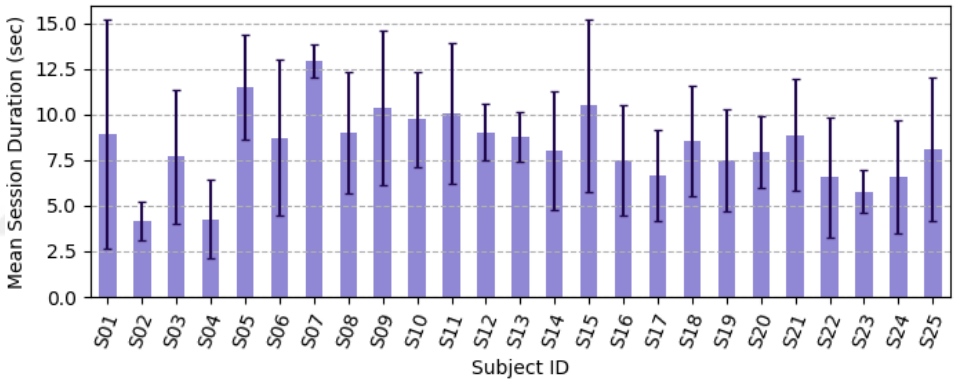


(d)

Figure 4.1: Distribution of pain intensity scores for (a) VAS, (b) SEN, (c) AFF, (d) OPI in the UNBC-McMaster Pain Archive.



(a)



(b)

Figure 4.2: The UNBC-McMaster Pain Archive distribution per participant for (a) the number of videos (b) the mean duration (with standard deviation) of the videos.

For each video sequence, there are 4 scales; 3 self-reported pain intensity and an observer's ratings of pain intensity (OPI). For self-report pain intensity, the participant reported 3 scales, the first two were performed on a 15-point Likert-type scale that ranged from 0 to 14. The first one; The sensory scale (SEN) ranged from "extremely weak" to "extremely intense", whereas the second; the affective motivational scale (AFF) ranged from "bearable" to "excruciating". The third scale; Visual Analog Scale (VAS), is an 11-point likert-type scale and ranged from Visual "No pain" to "Pain as bad as could be". Whereas, the observers' ratings of pain intensity were performed offline and on a 6-point Likert-type scale that ranged from 0 (no pain) to 5 (strong pain). Figure 4.1 shows the distribution of

pain intensity scores (i.e., number of available videos) for all pain scales across the dataset.

Figure 4.3: The 66 AAM tracked landmarks on randomly selected frames across the UNBC-McMaster Pain Archive.

The dataset also provided 66 Active Appearance Model (AAM) tracked landmarks at frame-by-frame level (as seen in Figure 4.3) which they obtained by employing an AAM-based system that uses AAMs to track the face and extract visual features. However, keyframes within each video sequence were manually labeled, while the remaining frames were automatically aligned using a gradient descent AAM fitting algorithm described in [98, 1]. They also provided expert-labeled FACS codes for each frame on a 0-5 ranking of intensity. These FACS codes were used to compute the intensity of the AUs in order to obtain the frame-level PSPI score for each frame in a sequence.

Each frame of each sequence is normalized as explained in section 3.2, the normalized faces are then used in two different ways. We either use these normalized faces and build up the pain pair archive in order to train the convolutional autoencoder as explained in section 3.3, or we feed the normalized sequences into the spatio-temporal model in order to estimate the pain intensities. Table 4.1

shows the number of pairs of subjects, videos, and match sequences for each pain category in the pain pair archive.

Table 4.1: Distribution of subject, video, and matched sub-sequence pairs with threshold $\mathcal{T} = 0.75$ in the UNBC-McMaster Pain Archive.

Category	Number of Pairs		
	Subject	Video	Matched Sub-sequences
No Pain	12	35	69,824
Mild Pain	21	83	782,355
Moderate Pain	17	40	279,187
Severe Pain	12	35	342,665
All	25	193	1,474,031

4.2 Experimental Setup

4.2.1 Data Preparation

Given the small size of our dataset, we want to reduce the bias in the model's estimated performance and ensure that our proposed solution can generalize on unseen data. Thus, we split the data into 5 different folds, where each fold contains 5 non-overlapping subjects. For each run of the cross-validation, one fold is separated as a test set, whereas another is separated as a validation set and the three remaining folds are used for training. During the training cycle, the best model of that fold is selected by minimizing the minimum validation error computed using the loss function. In the end, the 5 independent trained models are evaluated on their respective test folds using the evaluation metric MAE. The mean performance across all folds is reported as the model general performance.

The standard approach for training deep models is to use a random sampling of data for each training batch. However, considering that our dataset is skewed and imbalance per pain intensity level as seen in Figure 4.1, we want to eliminate the sampling bias in the distribution of the fold which occurs when the data subsets are selected incorrectly and don't represent the true distribution of the entire population (dataset). For that, we performed stratified random sampling on the folds, such that each fold contains a proper representation of different pain scales and is as close as possible to the entire dataset population. By doing so, the training data is randomly sampled while making sure a similar distribution of pain intensity scores is used between training folds. This also ensures that for each training fold, each intensity score appears at least once.

To perform startification on the dataset, the dataset population is represented in 4 frequency vectors of each pain scale's intensities, such as; $F_D^p = \{f_{D_0}^p, f_{D_1}^p, \ldots, f_{D_N}^p\}$ where $p \in P$, and $P = \{\text{VAS, OPI, AFF, SEN}\}$ and N is the upper bound of P. After that, a brute search is performed to find the most similar 5 folds with non-overlapping subjects to the database. In order to find them, each data subset S contains 5 different subjects, such as; $S^b = \{s_1^b, ..., s_5^b\}$, where S^b denotes a subset of 5 subjects. The subset S is represented with 4 pain scale frequency vectors as well, where its frequency vector of pain scale p is represented as follow; $F_S^p = \{f_{S_0}^p, f_{S_1}^p, ..., f_{S_N}^p\}$.

The similarity between each data subset S and the whole dataset D; denoted as $\mathcal{I}$, is computed by maximizing the cosine similarity $\mathcal{C}$ between vectors F_{D_P} and F_{S_P} for each pain scale p.

$$\mathcal{I}\left(D, S\right) = \frac{1}{4}\sum_{p}^{P} \mathcal{C}\left(F_D^p, F_S^p\right) \tag{4.1}$$

At the end of a brute force search, we select the 5 subsets; $\{S_1, S_2, S_3, S_4, S_5\}$, where $S_1^b \cap S_2^b \cap S_3^b \cap S_4^b \cap S_5^b = \varnothing$ and the similarity $(\widehat{\mathcal{I}})$ across the five subsets and general dataset is the highest.

$$\widehat{\mathcal{I}} = \frac{1}{5}\sum_{i}^{5} \mathcal{I}\left(D, S_i\right) \tag{4.2}$$

The cosine similarity scores for each pain scale between each selected fold and the entire dataset can be found in Table 4.2, whereas Figure 4.4 shows the differences between the distributions of the folds and the whole database as it represents the smoothed, continuous density histogram estimated from the data using kernel density estimation (kde).

Table 4.2: The Cosine similarity of the data distribution of the 5 folds to the entire database per pain scale

Fold	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
VAS	0.91	0.98	0.94	0.94	0.88
OPI	0.88	0.99	0.98	0.99	0.97
AFF	0.95	0.98	0.96	0.97	0.86
SEN	0.96	0.95	0.96	0.97	0.89

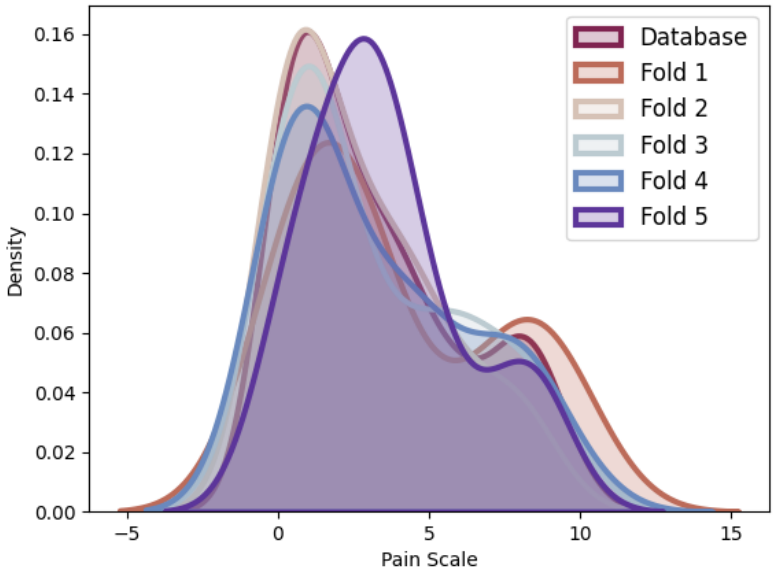


Figure 4.4: A Normalized Density Plot of the Five Folds and Database

4.2.2 Model Training and Implementation

To evaluate the proposed framework, we employ UNBC-McMaster Pain Archive as discussed in the previous section 4.1, whereas all the experiments are conducted using a 5-fold cross-validation scheme as explained in section 4.2.1. At each phase of the framework, we separate the test subjects while the system is training and optimizing its parameters using the training subjects. The data separation goes as early as data pre-processing in phase 1 as seen in Figure 3.1 to prevent data leakage.

The proposed framework is implemented using Python/PyTorch for GPU. The Autoencoder and CNN-RNN architectures are trained using Root Mean Square Propagation Optimizer (RMSprop). [1]

The hyperparameters of the networks and training, such as; learning rate, number of recurrent layers, number of hidden neurons, *alpha* in the consistency loss, *lambda* as Regularization rate for L2 were selected in preliminary cross-validation experiments and then used throughout all the following experiments without further adjusting.

Table 4.3: List of the considered hyper-parameters for the CNN-RNN model optimization in preliminary cross-validation experiments

Hyper-parameter	Considered Values	Selected Value
Recurrent Layers	{1, 2, 3}	2
Learning rate	{1e−2, 1e−3, 1e−4}	1e−2
Regularization rate (λ)	{1e−3, 1e−4, 1e−5}	1e−4
Consistency term factor (α)	{0.3, 0.5, 0.7}	0.7
GRU Dropout factor	{0, 0.3, 0.5}	0.5

[1] RMSprop is an unpublished algorithm first proposed in the Coursera course. "Neural Network for Machine Learning" lecture six by Geoff Hinton [99].

4.3 Experimental Results and Discussion

The outputs of the trained models are continuous values estimated from our regression model. To measure the error in pain scores, we first round the obtained continuous scores to ordinal (discrete) predictions and then measure the mean absolute error between the ground truth (ordinal) and the rounded predictions in the original intervals of the corresponding pain scale [0, 10].

The rest of this section is going to discuss the results of the 30 experiments conducted on UNBC-McMaster Pain Archive to evaluate the added value of each component in the framework. We first start with assessing and discussing the effectiveness of using different initial weights for the spatial module. We look into the effect of initializing the weights using He initialization strategy [100], the pre-trained AlexNet weights on ImageNet dataset, as well as the weights from the encoder part from the pre-trained autoencoder. The effectiveness of the autoencoder weights is also assessed in the light of the training data, where we look into the effect of using the pain pair archive as the decoder tries to decode the input back to its corresponding pair, or back to its original self. We also explore the added value of multiple pain scales, along with the importance of enforcing consistency between them. Finally, We follow it up with a discussion and comparison between our method and the state-of-the-art methods and their reported MAE results.

The significance between the results for each grouped setup is reported. Given the repeated-measures nature of the data, non-parametric repeated measures Friedman test was used to evaluate the mean differences in the MAEs between the use of single and multiple pain scales. Separate Friedman tests were used for the VAS results. Wilcoxon signed-rank test was used for post-hoc analyses following the significant Friedman test. Friedman and Wilcoxon were used as the MAEs distributions were not normally distributed, as per Shapiro-Wilks tests.

4.3.1 Assessment of the impact of pre-training

Training a deep neural network remains a challenging task despite the recent groundbreaking results that it has achieved over the years. One of the main challenges is exploding or vanishing gradients during the training process due to poor initial weights values. If the initial weights were initialized equally, then the activation of the neurons is going to be the same, resulting in gradients being updated by the same amount, which would lead the neurons to learn the same information. All in all, this hinders the learning process and creates what's known as a symmetry problem. Whereas randomly initializing the weights to random values could risk these values being too large which would lead to saturation problems as the activation functions would be making very small changes in the value of their outputs [101, 102].

To address this, both Xavier [103] and He [100] proposed sophisticated initialization techniques that maintain the same variance across every layer, which helps prevent the gradients from exploding while initializing the weights of the layer to values from a uniform normal distribution with constant variance. Xavier's initialization [103] strategy is best suited for linear/sigmoid activation methods, whereas He *et al.*[100] made adjustments to Xavier's method which made it suitable for networks with ReLU and Leaky ReLU activation functions.

In this section, we assess the impact of initializing the spatial module weights to initial weights using He initialization, the pre-trained weights of AlexNet on ImageNet database [2], and the pre-trained weights of the encoder part from the autoencoder network trained on pain archive. Table 4.4 and Table 4.5 show the obtained MAEs for the self-reported VAS pain scores using different initialization weights when freezing said weights or fine-tuning them through back-propagation respectively. Figure 4.5 and Figure 4.6 represents the MAE prediction errors over the five folds across the three different initialization methods for both freezing and fine-tuning the spatial weights.

[2]https://download.pytorch.org/models/alexnet-owt-4df8aa71.pth

Table 4.4: MAEs in estimating the VAS pain scores using different initialized weights for frozen AlexNet

Training Scales	**Initialized Spatial Weights**	**MAE (VAS)**
1 Scale (VAS)	He Initialization	4.8028
	ImageNet	3.1316
	Pain-based Encoder	**2.320**
4 Scales	He Initialization	2.923
	ImageNet	2.390
	Pain-based Encoder	**2.073**

When we freeze the initial weights, there is a significant effect on the VAS MAEs for the use of different initialization weights whether the network was trained using VAS scale only ($\chi^2 = 35.6$, $df = 2$, $p < 0.001$), or when trained using all four scales ($\chi^2 = 26.7$, $df = 2$, $p < 0.001$) as seen in Table 4.4 and Figure 4.5. The use of He initialization insured that we don't have a case of exploding gradients and the network was able to converge, however, it performed the worst among them all. Using ImageNet outperformed He initialization when training on one scale ($W = 6183$, $p < 0.001$) as well as all four scales ($W = 6997$, $p < 0.001$), however, the use of encoder weights outperformed both.

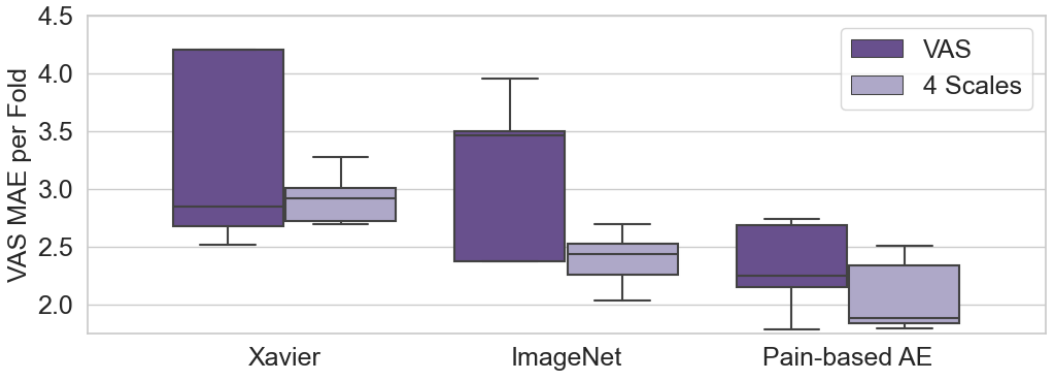


Figure 4.5: Absolute Error across folds using different initializing weights for frozen AlexNet

Using encoder weights improves the pain scores estimation compared to using He initialization when training using VAS scale only ($W = 4244$, $p < 0.001$) and all four scales ($W = 5538$, $p < 0.001$) and to ImageNet pre-trained weights when training using VAS scale ($W = 4563$, $p < 0.001$) and all four scales ($W = 5980$, $p < 0.001$).

Similarly, when we fine-tune the initial weights by optimizing the parameters through back-propagation, there is a significant effect on the VAS MAEs for the use of different initialization weights whether the network was trained using VAS scale only ($\chi^2 = 17.7$, $df = 2$, $p < 0.001$), or when trained using all four scales ($\chi^2 = 24.1$, $df = 2$, $p < 0.001$) as seen in Table 4.5 and Figure 4.6.

When fine-tuning the weights, using ImageNet weights instead of He initialization weights does not provide a significant difference regardless of whether VAS scale is used ($W = 8983$, $p = 0.19$) or all four scales are used ($W = 10006$, $p = 0.95$). However, fine-tuning the weights coming from the encoder outperformed both methods. When compared to He initialization, it improves the results by 17% when trained on VAS only ($W = 6800$, $p < 0.001$) and by 20% when trained on all 4 scales ($W = 5584$, $p < 0.001$). Also, compared to ImageNet weights, fine-tuned encoder weights improves the results by 12% ($W = 6445$, $p < 0.001$) and 17% ($W = 8084$, $p = 0.016$) when training on VAS only or four scales respectively.

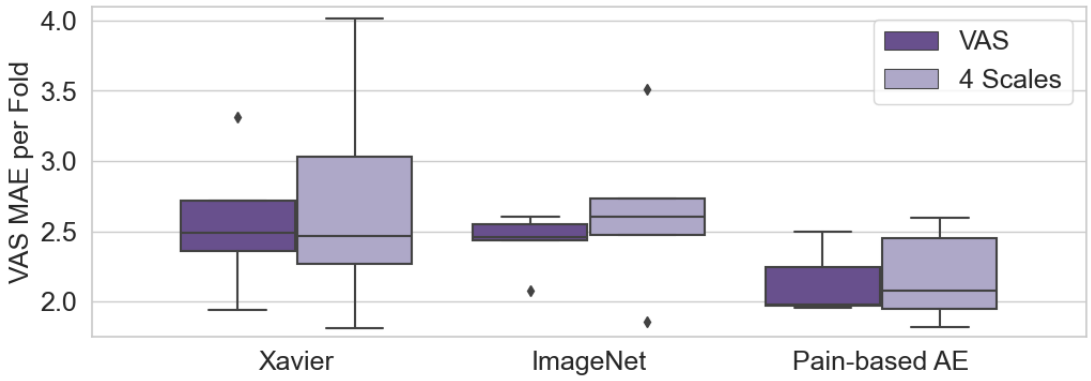


Figure 4.6: Absolute Error across folds using different initializing weights for fine-tuned AlexNet

Based on the obtained results shown in both tables; Table 4.4 and Table 4.5, we conclude that the pre-training step that we performed to learn the pain feature representation using autoencoder is effective and that the learned weights help the network focus on these pain features which enhanced the model's ability to assess and estimate pain.

Table 4.5: MAEs in estimating the VAS pain scores using different initialized weights for fine-tuned AlexNet

Training scales	**Initialized Spatial Weights**	**MAE (VAS)**
1 Scale (VAS)	He Initialization	2.564
	ImageNet	2.425
	Pain-based Encoder	**2.128**
4 Scales	He Initialization	2.718
	ImageNet	2.635
	Pain-based Encoder	**2.179**

Effect of fine-tuning

Here, we want to look further into the effect of fine-tuning the encoder weights from the pre-trained autoencoder given different training scales. When training using VAS only, fine-tuning through the said initialized weights performed better than when the weights are frozen ($W = 8115$, $p = 0.018$). Whereas, when training using all four scales, freezing these weights performed better than updating the spatial weights through propagation ($W = 8836$, $p = 0.014$) as seen in Table 4.6. This can be the effect of the consistency term in the loss function during the training using multiple pain scales. So when we are training using VAS only, the network learns better through back-propagating the spatial module as well, however, when training with multiple scales (four in this case), the network is able to reach convergence faster and without the need to update the spatial weights due to the effect of minimizing the deviation values between the pain scales. The

effect of the consistency term is further explored in section 4.3.4.

Table 4.6: MAEs in estimating the VAS pain scores using pain based encoder weights for fine-tuned and frozen AlexNet

Training Scales	**Fine-tuning of Spatial Weights**	**MAE (VAS)**
1 Scale (VAS)	Yes	**2.128**
	No	2.320
4 Scales	Yes	2.179
	No	**2.073**

4.3.2 Effectiveness of Pain-based Autoencoder Weights

As explained in section 3.1, given the nature of autoencoders, they are widely used to pre-train a network using only a set of data. Autoencoder usually aims to reconstruct its input back by minimizing the reconstruction error. However, in our case, we aim to capture the pain information and patterns by exploiting the transformation of the facial expression from one subject to another similar facial expression of another subject as explained in section 3.3.

We assess the effectiveness of training our autoencoder using our pain archive pairs by comparing the performance of the model trained on that and another model trained with the aim to reconstruct the input to itself. Regardless of the used training scales, when fine-tuning the pain-based encoder weights for the initialization of the spatial module, it outperforms the model initialized with the weights of reconstruction-based encoders by 15% ($W = 6436$, $p < 0.001$) when training using VAS only, and by 12% ($W = 8388$, $p = 0.043$) when training using all four scales as seen in Table 4.7 and Figure 4.7.

Table 4.7: MAEs in estimating the VAS pain scores using different encoder weights for fine-tuned AlexNet

Training Scales	**Initialized Spatial Weights**	**MAE (VAS)**
1 Scale (VAS)	Reconstruction-based Encoder	2.510
	Pain-based Encoder	**2.128**
4 Scales	Reconstruction-based Encoder	2.472
	Pain-based Encoder	**2.179**

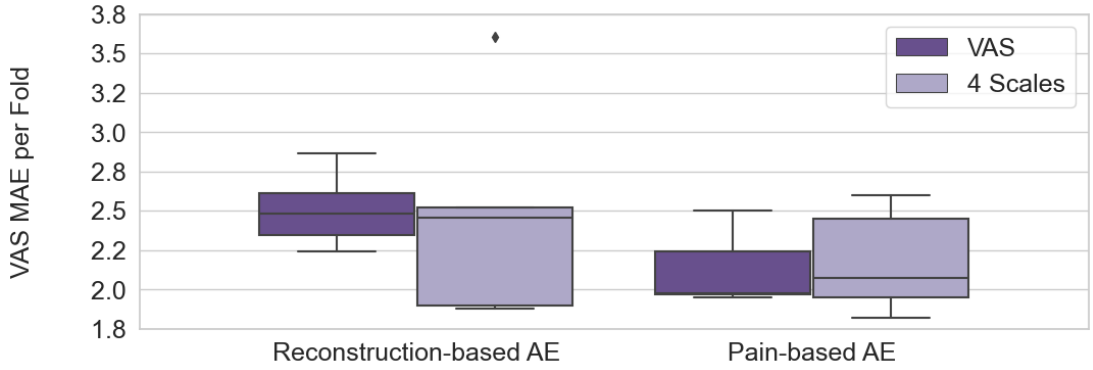


Figure 4.7: Absolute Error across folds using different encoder weights for fine-tuned AlexNet

However, when we use the initialized weights of the pre-trained encoder for our spatial module without fine-tuning them while training the model using VAS scale only, there is no statistical significance in the results of pain-based encoder weights to the ones of the reconstructed encoder weights ($W = 4746$, $p = 0.07$) despite it being 13% better (lower MAE) as seen in Table 4.8 and Figure 4.8. Nevertheless, when training the model using four scales, the model with the frozen pain-based encoder weights performed significantly better than the one initialized from the reconstruction-based encoder ($W = 5988$, $p < 0.001$).

Table 4.8: MAEs in estimating the VAS pain scores using different encoder weights for frozen AlexNet

Training Scales	**Initialized Spatial Weights**	**MAE (VAS)**
1 Scale (VAS)	Reconstruction-based Encoder	2.665
	Pain-based Encoder	**2.320**
4 Scales	Reconstruction-based Encoder	2.658
	Pain-based Encoder	**2.073**

Generally, we can conclude based on the results in Table 4.7 and Table 4.8 that the feature representation learned using our autoencoder is capable of capturing the facial cues related to pain as it is trained on minimizing the differences between two different subjects yet similar pain values. Learning the mapping within self is not enough to exploit the facial pain cues as it is seen with these experiments. The model trained using the weights of the autoencoder that aimed to reconstruct the input back to itself performed relatively worse than the one that exploited the facial pain expression between two different subjects, where the latter helped the model predict the pain values more accurately.

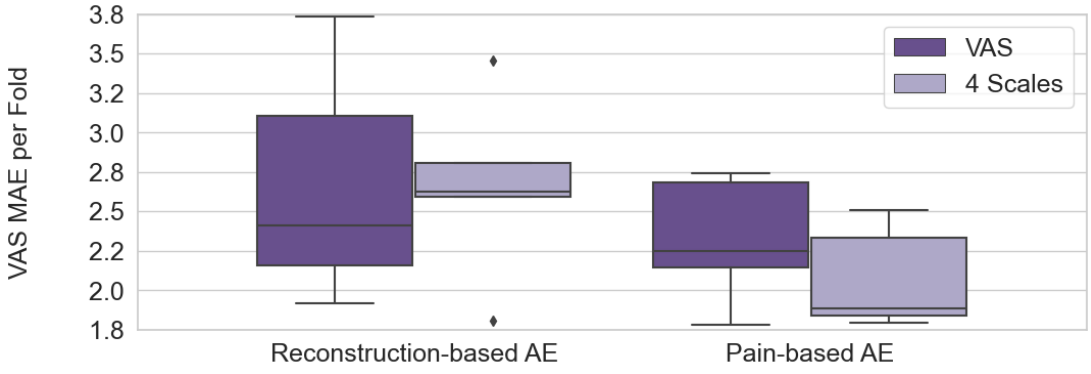


Figure 4.8: Absolute Error across folds using different encoder weights for frozen AlexNet

4.3.3 Added Value of Multiple Pain Scales

While useful, self-reported pain have notable limitations such as inconsistent metric properties across scale dimensions, reactivity to suggestion, and differences between clinicians' and sufferers' conceptualizations of pain [2]. Multiple self-reported pain scales (e.g., VAS, AFF, SEN) have varying numbers of pain levels with different wordings for pain scores, making the obtained scores across multiple instruments complementary. Considering that UNBC-McMaster Pain Archive provides the values for each video in each of the four pain scales, we want to leverage all the provided information in order to improve the estimation of the pain scale (VAS). For this reason, we propose an approach that increases the degree of agreement among the different uni-dimensional pain scales, and thus increases the accuracy of self-reported (i.e., the VAS pain scores). Table 4.9 shows the VAS pain estimation results using multiple pain scales. Based on Friedman significant test, there is a significant impact on the use of multiple pain scales ($\chi^2 = 19.5$, $df = 3$, $p < 0.0005$). Using four scales (i.e., self-reported and perceived pain scales) improves pain scores estimation (i.e., lower MAEs) compared to using a single self-reported pain scale ($W = 7260$, $p < 0.001$) and to combining the self-reported pain scale with the perceived pain scale ($W = 7886$, $p = 0.008$), as well as combining all self-reported pain scales ($W = 6731.5$, $p < 0.001$). However, the usage of any of the other combinations does not provide any significant effect. For example, there is no difference between using a single self-reported scale and combining it with the perceived pain scale ($W = 8643.5$, $p = 0.08$), nor with the combination of multiple self-reported pain scales ($W = 9983.5$, $p = 0.94$). Also, there is no difference between the combination of multiple self-reported pain scales and the combination of self-reported pain scale (VAS) with observer-reported scale ($W = 8869$, $p = 0.15$).

Table 4.9: MAEs in estimating the self-reported VAS pain scores using different pain scales in training

Training Scales	**MAE (VAS)**
1 Label (VAS)	2.320
2 Scales (VAS+OPI)	2.186
3 Scales (VAS+AFF+SEN)	2.293
4 Scales (VAS+OPI+AFF+SEN)	**2.073**

Overall, the obtained results are shown in Table 4.9 and illustrated in Figure 4.9 support the hypothesis that multiple self-reported measures (i.e., VAS, AFF, SEN) along with the observer reported scale (OPI) are complementary and their combination reduces the inconsistency and increases the reliability of self-reported pain estimation (in our case the VAS pain scores).

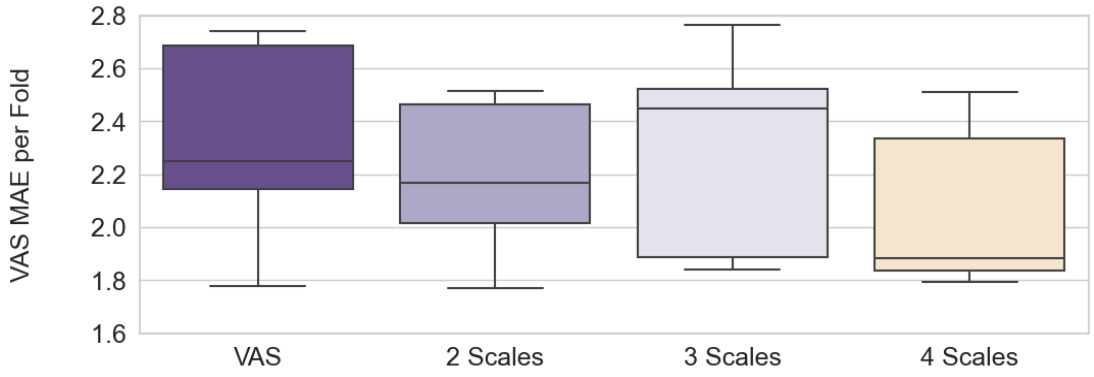


Figure 4.9: Absolute Error across folds using using different pain scales in training

4.3.4 Importance of Enforcing Consistency Between Scales

As reported in section 3.4.2 (see Eq. 3.6), the proposed loss function introduces a consistency term that allows improving the accuracy of the predicted pain scores by increasing the consistency between multiple pain scales. Table 4.10 shows the

the Pearson correlation coefficient between pain scores measured using the three self-reported pain scales (i.e., VAS, AFF, SEN) and the observer reported pain scores (i.e., OPI). As expected, the obtained correlation coefficients suggest a strong positive correlation ($|r| > 0.5$) between different pain scales [104]. Correlation is higher within self-reported pain scales compared to between self-reported and observer pain scales (i.e., perceived pain).

Table 4.10: Coefficients of Pearson's correlation between the scores in different pain scales

	AFF	**OPI**	**SEN**	**VAS**
AFF	**1**			
OPI	0.655	**1**		
SEN	0.949	0.703	**1**	
VAS	0.898	0.664	0.922	**1**

To evaluate the added value of enforcing the consistency between different pain scales, we compare the proposed loss function (i.e., Eq. 3.6) to the standard loss (using only MAE and regularization terms) without the weighted consistency term $\mathcal{L}_{\text{wC}}$. Table 4.11 shows the obtained MAEs for the self-reported VAS pain scores, with and without the weighted consistency term $\mathcal{L}_{\text{wC}}$ in the loss function (see Eq. 3.6, section 3.4.2). The weighted consistency term significantly improves the MAE in the case of two scales (VAS+OPI) compared to the regular loss without the weighted consistency by 24% ($W = 6101.5$, $p < 0.001$).

Similarly, the weighted consistency term improves the MAE in case of four scales (VAS+OPI+AFF+SEN) compared to the loss without $\mathcal{L}_{\text{wC}}$ by 8.5% ($W = 7376$, $p = 0.001$). However, there is no significant effect on the usage of the weighted consistency term in the case of the combination of multiple self-reported scales ($W = 9393$, $p = 0.42$).

Table 4.11: MAEs for estimating VAS with and without (w/o) using the weighted consistency term $\mathcal{L}_{\text{wC}}$ in the loss function

Training Scales	**Loss**	**MAE (VAS)**
2 Scales (VAS+OPI)	with $\mathcal{L}_{\text{wC}}$	2.186
	w/o $\mathcal{L}_{\text{wC}}$	2.8872
3 Scales (VAS+OPI+AFF)	with $\mathcal{L}_{\text{wC}}$	2.2925
	w/o $\mathcal{L}_{\text{wC}}$	2.808
4 Scales (VAS+OPI+AFF+SEN)	with $\mathcal{L}_{\text{wC}}$	**2.073**
	w/o $\mathcal{L}_{\text{wC}}$	2.268

Overall our results as seen in Table 4.11 and Figure 4.11 show the usefulness of employing the weighted consistency term in the loss function for estimating the self-reported VAS scores, as it helps with reducing the inconsistency between the scales while keeping them proportional to one another.

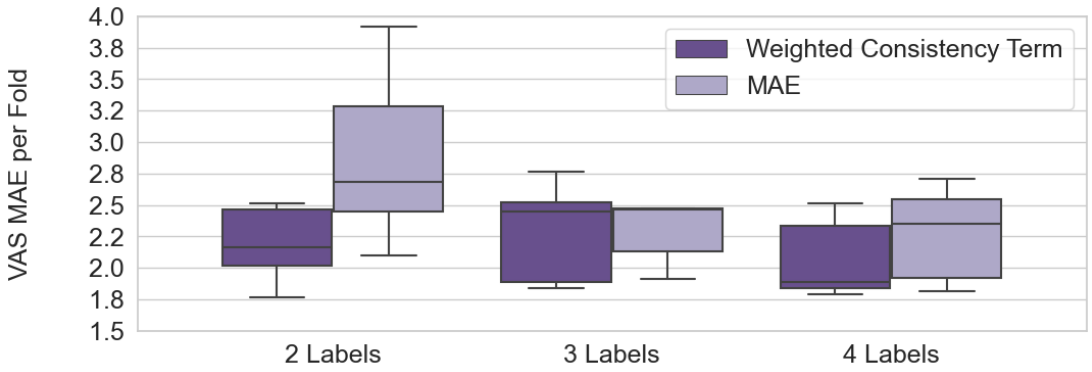


Figure 4.10: Absolute Error across folds with and without (w/o) using the weighted consistency term $\mathcal{L}_{\text{wC}}$ in the loss function

Effect of weighted scales

We also further investigate the effect of using weights in the consistency term by comparing the results of the proposed loss $\mathcal{L}_{\text{wC}}$ to its counterpart in our previous work [11] where all scales are assumed to be equally proportional to one another and thus the weights are set to equal one, resulting in what we called a consistency term $\mathcal{L}_{\text{C}}$ as seen in Equation 3.8.

Table 4.12 and Figure 4.11 show the obtained MAEs for the self-reported VAS pain scores, with and without the weights in the consistency term ($\mathcal{L}_{\text{wC}}$ and $\mathcal{L}_{\text{C}}$) in the loss function (see Eq. 3.6, section 3.4.2). The weighted consistency term significantly improves the MAE in the case of two scales (VAS+OPI) compared to the consistency term without weights by 9.4% ($W = 7581.5$, $p = 0.003$). Also, there is a significant effect on the usage of the weighted consistency term in the case of the combination of multiple self-reported scales ($W = 7665.5$, $p = 0.004$). Similarly, the weighted consistency term improves the MAE in case of four scales (VAS+OPI+AFF+SEN) compared to the consistency term without weights by 9% ($W = 6481.5$, $p < 0.001$).

Table 4.12: MAEs for estimating VAS with and without (w/o) using weights in the consistency term in the loss function

Training Scales	**Loss**	**MAE (VAS)**
2 Scales (VAS+OPI)	with $\mathcal{L}_{\text{wC}}$	2.186
	with $\mathcal{L}_{\text{C}}$	2.423
3 Scales (VAS+OPI+AFF)	with $\mathcal{L}_{\text{wC}}$	2.293
	with $\mathcal{L}_{\text{C}}$	2.570
4 Scales (VAS+OPI+AFF+SEN)	with $\mathcal{L}_{\text{wC}}$	**2.073**
	with $\mathcal{L}_{\text{C}}$	2.280

The results reported in Table 4.12 along with the ones in Table 4.11 confirm the hypothesis that ensuring the consistency between the pain scales with respect to

their proportion to one another increases the accuracy of the model and its ability to estimate VAS with higher precision. Using the consistency term on its own provides good results given the fact that all scales are highly correlated, however adding the weights to it improves the results further more by exploiting the high correlation between the scales while maintaining the proportional differences between them.

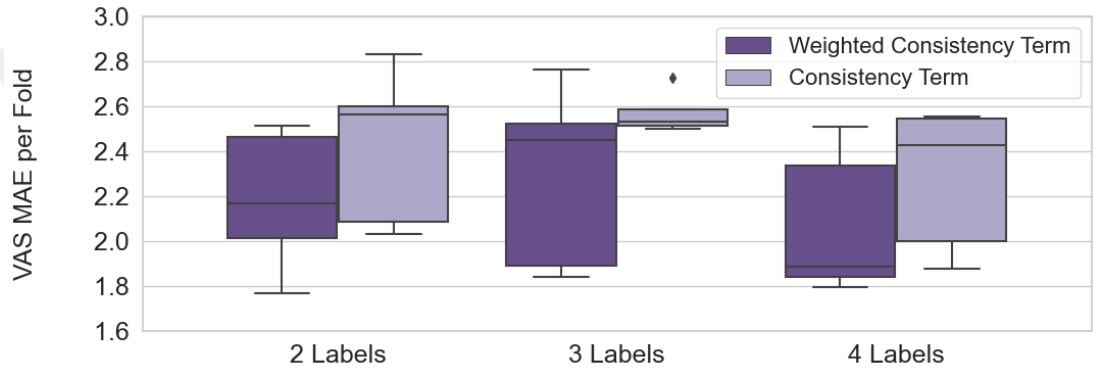


Figure 4.11: Absolute Error across folds with and without (w/o) using weights in the consistency term in the loss function

4.3.5 Effect of Imbalanced Data

Due to the sparse representation of some gradations of pain intensity (see Figure 4.1), the trained models could be affected by the number of samples per intensity seen by the model during the training. Therefore, using different sets of folds that are formed randomly, can cause variations in the obtained results, i.e., MAEs. To overcome that limitation, we perform a stratification to distribute the database into 5 folds, where each fold has a very similar distribution with the whole dataset as explained in section 4.2.1.

To investigate probable inconsistencies in results, which can be caused by the use of random folds in the experiments and the effect of the imbalanced number of samples per pain intensity level (see Figure 4.1) on model training, we compare the MAEs obtained using three different sets of random folds on the same setup of our best performing model (4 Scales with pain-based encoder weights, trained

using $\mathcal{L}_{\text{wC}}$). As observed from Table 4.13, MAEs for different set of randoms folds vary significantly ($\chi^2 = 22.6$, $df = 2$, $p < 0.001$). We then evaluate the two sampling strategies during the models training by comparing the stratified MAE results with the ones from the random folds. The different sampling strategies are statistically significant ($\chi^2 = 56.5$, $df = 3$, $p < 0.001$). Stratified sampling improves the MAE compared to any of the three randomly sampled folds by 9.6% ($W = 6712.5$, $p < 0.001$), 15% ($W = 6521$, $p < 0.001$) and 51.6 ($W = 3476.5$, $p < 0.001$) respectively.

Table 4.13: MAEs in estimating the self-reported VAS pain scores using different data folds

Data Folds	**MAE (VAS)**
Startified	**2.073**
Random 1	2.294
Random 2	2.441
Random 3	4.287

4.3.6 Comparison to the State-of-the-art Methods

To further evaluate the proposed framework for pain intensity measurement, we compare our model to the three available models for self-reported pain intensity measurement from videos which were thoroughly reviewed in Section 2.2. In general, our framework performs better than all the state of art results in most of its settings in terms of MAEs, where our best performing setup achieves an MAE of 2.07 as seen in Table 4.14.

To elaborate more, when training using only VAS scale, our framework performs 9% better than Martinez *et al.*[8] with an MAE of 2.46. The shortcoming of their approach is that it is subject-dependent which does not generalize to previously unseen participants whereas our method is a subject-independent one.

Table 4.14: Comparison of the proposed model with other state-of-art methods using single and multiple scales, respectively

Model	**Training Scales**	**MAE (VAS)**
RNN-HCRF (Martinez *et al.*2017)	VAS, PSPI	2.46
DeepFaceLift (Liu *et al.*2017)	VAS	2.30
	VAS, OPI	2.24
Gram-SVR (Szczapa *et al.*2021)	VAS	2.44
Proposed Framework	VAS	2.24
	VAS, OPI	2.18
	VAS, AFF, SEN	2.57
	VAS, AFF, SEN, OPI	**2.07**

The framework proposed by Liu *et al.* [9] is a two-stage personalized deep neural network DeepFaceLift. The first stage of the DeepFaceLift model is a fully connected neural network that takes the AAM facial landmarks as input. The second stage is a Gaussian Process Regression model that takes the output of the first stage as input and outputs the VAS scores. Their framework also includes personalized features, but for the sake of comparing our methods with theirs, we disregard the results on which they used personalized features and report the ones where they used VAS as well as a combination of VAS and OPI in training. For the same setup, when using VAS for training, our framework achieves an MAE of 2.24, whereas DeepFaceLIFT's MAE is 2.30. Whereas when using the combination of VAS and OPI for training, DeepFaceLIFT obtains an MAE of 2.24, where our model for the same setup outperforms it with an MAE of 2.18. The latest published study by Szczapa *et al.*2021 also aims to estimate VAS from video sequences using a geometry-based approach. Their reported MAE results for 5-fold cross-validation is 2.44 when using VAS for training as well.

4.3.7 Challenges and Shortcomings

Although our model improves the state-of-the-art methods, it faces many challenges and limitations. As seen in Figure 4.12, the estimation of each pain intensity level improves when optimizing the model through weighted consistency loss while training using four labels in comparison to when we train the model using the self-reported score (VAS) only. However, despite the models performing well on the intensities with higher number of training samples, both models struggle with estimating higher pain intensity levels (VAS > 6). As seen in Figure 4.12, the MAE per intensity score where the score is higher than six is relatively higher than the MAE for intensity score lower than that. This can be explained by the fact that such intensities are underrepresented in our dataset.

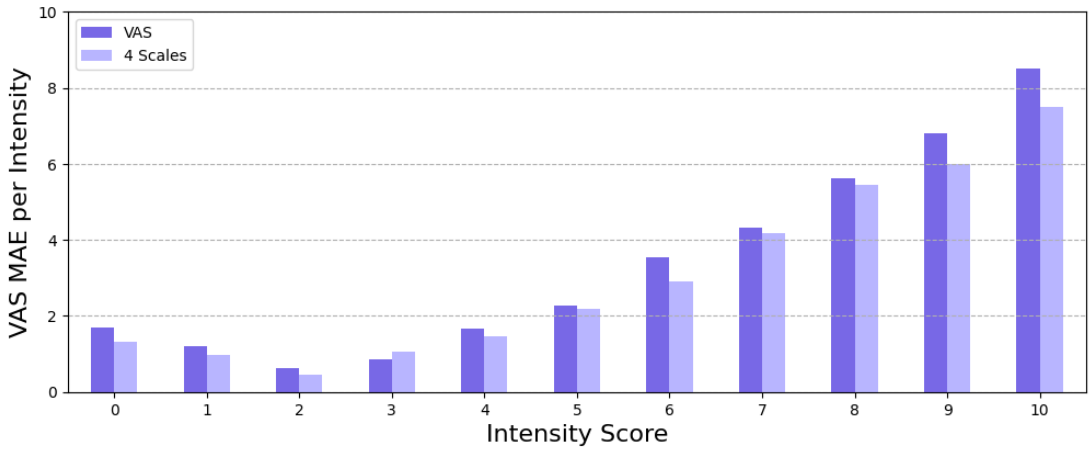


Figure 4.12: Distribution of the VAS MAEs per pain intensity score

4.3.8 Visual Analysis of Pain Cues

Our goal is to get an insight on the visual cues learned by the trained model (best model, with all scales and with the weighted consistency term (Equation 3.6)). To do so, instead of estimating pain score at the end of the video, we generate pain score at each frame of the video as it was the end of the video sequence (by progressively combining all video frames from the start to the current frame).

Consequently, the estimation of pain score for the last time step includes all frames of the input video (see section 3.4). In this way, we compute the regression score (i.e., corresponding pain intensity) at each time step such that:

$$Y_i = \Sigma_{i=1}^{t} f(x_1, ..., x_{i-1}, x_i) \tag{4.3}$$

where f represents our model, t is the number of frames per video, and x_i is the normalized face image at time step t_i. The model combines at each time step t_i all previous images from the beginning until the current time step (or the end of the sequence) to refine and generate its pain prediction Y_i. For each video, we plot the time series $\{Y_1, Y_2, ..., Y_t\}$ obtained as described above.

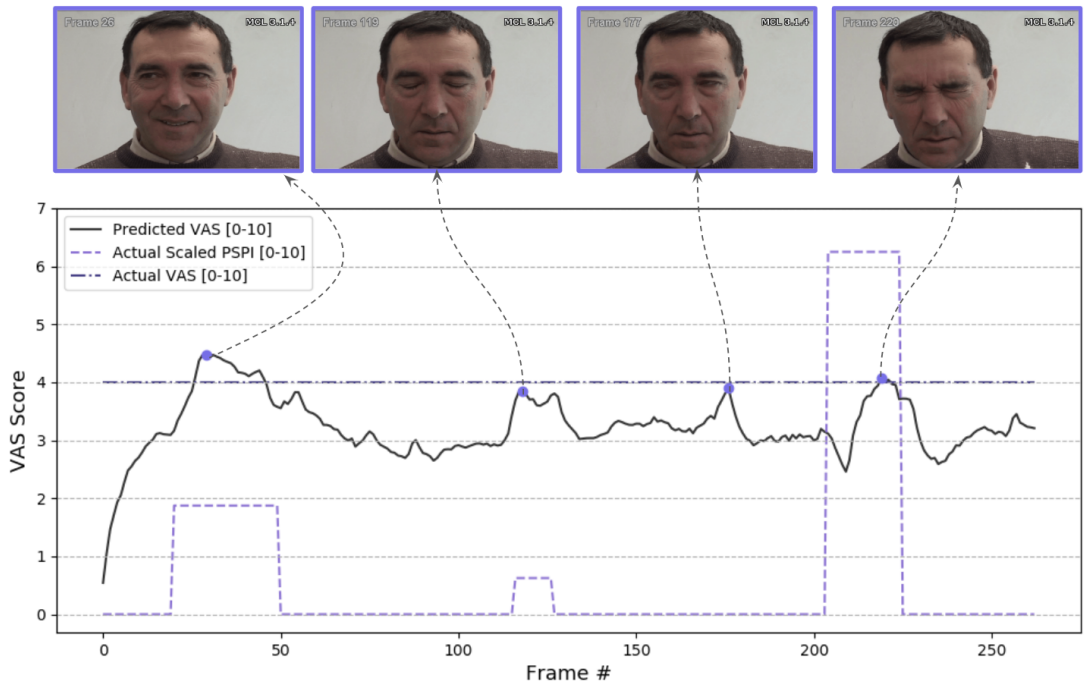


Figure 4.13: Frame-by-frame actual scaled PSPI and predicted VAS scores

Figure 4.13 show an example of the obtained pain scores over time. We select the time steps that correspond to the global/local highest scores ($P = p_i$) and plot the corresponding images (or the images in the surrounding +/-5 images window). We also include two ground truth values; 1) the actual VAS score for that sequence, and it is plotted on the graph as constant function, 2) the

PSPI score per frame which were calculated using the AU intensities as shown in Equation 2.1. For visualization purposes, we scale PSPI to [0-10] scale.

As shown in Figure 4.13, our model reveals different facial cues for pain expression observed around the aforementioned maximas (p_i). These facial cues include eye closure as in all of the last three images, brow lowering and lips tightening as in the last image. However, when we look at the first image and its corresponding PSPI and predicted VAS score, one would assume that the participant is feeling pain, but the participant is actually smiling. This shows that even PSPI score can include false positives as well. When we compare our predicted VAS score throughout the frames and the actual VAS score (VAS=4), they are consistent with one another. Similarly, PSPI scores and the predicted accumulative VAS score have similar inclines and declines, despite the differences in magnitude.

Chapter 5

Conclusion

In this thesis, we have proposed a spatio-temporal approach for self-reported pain intensity measurements from videos. The proposed architecture has employed a pre-training step that aims to learn the efficient pain facial feature encoding by employing a convolutional autoencoder that intends to learn facial encoding which transforms the facial expressions between subjects with similar pain scores. The learned representation is transferred to the spatio-temporal model which is additionally optimized using a custom loss function. The new loss function has been introduced to increase the consistency between multiple pain scales with respect to their proportion to one another, while also improving the prediction accuracy of pain scores by minimizing the absolute error between actual and predicted scores.

Each of the proposed components employed in the presented pain estimation framework such as the effect of pre-training weights, added value of the three self-reported pain scales, as well as an observer pain intensity scale, the effectiveness of enforcing consistency between the scales, the importance of keeping the consistency proportional between scales, and the effect of data fold sampling has been evaluated on the UNBC-McMaster Pain Archive in a detailed manner.

The experimental results have confirmed the effectiveness of each of the proposed components on the reliable assessment of pain intensity from facial expressions. Our results show that using convolutional autoencoder for unsupervised pre-training method to learn pain facial representation while enforcing the consistency between multiple pain scales in a proportional manner enhances the reliability of the subjective self-reported pain estimation.

To conclude, our method shows promising results consolidating the feasibility of using automatic pain assessment as a complementary tool in hospitals and clinics to further support medical staff in objective assessment of pain. However, to be able to use automatic pain assessment used within clinical setup with higher confidence, further studies and research should be conducted to assess how automatic pain assessment would vary between subjects from different gender, age and ethnic groups. Moreover, the dataset used in our work and most previous work only focused on pain caused by one specific reason (shoulder pain), however in reality, pain is caused by various number of factors which can affect the facial expressions, body movements as well as the body language of the patients. Also, further work should investigate the contribution of head pose changes, body movement variation and vocal information on the effectiveness of assessing pain objectively.

Bibliography

[1] P. Lucey, J. F. Cohn, K. M. Prkachin, P. E. Solomon, and I. Matthews, "Painful data: The unbc-mcmaster shoulder pain expression archive database," in *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*, (Santa Barbara, CA), pp. 57–64, 2011.

[2] Z. Hammal and J. Cohn, *Automatic, Objective, and Efficient Measurement of Pain Using Automated Face Analysis*, pp. 121–146. Springer International Publishing, 2018.

[3] T. Hadjistavropoulos, K. Herr, K. M. Prkachin, K. D. Craig, S. J. Gibson, A. Lukas, and J. H. Smith, "Pain assessment in elderly adults with dementia," *The Lancet Neurology*, vol. 13, no. 12, pp. 1216–1227, 2014.

[4] K. M. Prkachin and P. E. Solomon, "The structure, reliability and validity of pain expression: Evidence from patients with shoulder pain," *Pain*, vol. 139, no. 2, pp. 267–274, 2008.

[5] K. D. Craig, J. Versloot, L. Goubert, T. Vervoort, and G. Crombez, "Perceiving pain in others: automatic and controlled mechanisms," *The Journal of Pain*, vol. 11, no. 2, pp. 101–108, 2010.

[6] K. D. Craig, K. M. Prkachin, and R. V. Grunau, "The facial expression of pain," *Handbook of pain assessment*, vol. 2, pp. 257–276, 1992.

[7] R. Ekman, *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, USA, 1997.

[8] D. L. Martinez, O. Rudovic, and R. Picard, "Personalized automatic estimation of self-reported pain intensity from facial expressions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, (Honolulu, HI), pp. 2318–2327, 2017.

[9] D. Liu, F. Peng, O. O. Rudovic, and R. W. Picard, "Deepfacelift: Interpretable personalized models for automatic estimation of self-reported pain," in *Proceedings of the 1st IJCAI Workshop on Artificial Intelligence in Affective Computing*, Proceedings of Machine Learning Research, (Melbourne, Australia), pp. 1–16, 2017.

[10] B. Szczapa, M. Daoudi, S. Berretti, P. Pala, A. Del Bimbo, and Z. Hammal, "Automatic estimation of self-reported pain by interpretable representations of motion dynamics," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2544–2550, IEEE, 2021.

[11] D. Erekat, Z. Hammal, M. Siddiqui, and H. Dibeklioğlu, "Enforcing multilabel consistency for automatic spatio-temporal assessment of shoulder pain intensity," in *Companion Publication of the 2020 International Conference on Multimodal Interaction*, pp. 156–164, 2020.

[12] K. D. Craig, S. A. Hyde, and C. J. Patrick, "Genuine, suppressed and faked facial behavior during exacerbation of chronic low back pain," *Pain*, vol. 46, no. 2, pp. 161–171, 1991.

[13] C. D. Clemente, "Anatomy: A regional atlas of the human body, 4th edn," *Journal of Anatomy*, vol. 192, pp. 473–476, Apr 1998.

[14] G. C. Littlewort, M. S. Bartlett, and K. Lee, "Faces of pain: automated measurement of spontaneous all facial expressions of genuine and posed pain," in *Proceedings of the 9th international conference on Multimodal interfaces*, (Nagoya Aichi, Japan), pp. 15–21, 2007.

[15] A. B. Ashraf, S. Lucey, J. F. Cohn, T. Chen, Z. Ambadar, K. M. Prkachin, and P. E. Solomon, "The painful face–pain expression recognition using active appearance models," *Image and vision computing*, vol. 27, no. 12, pp. 1788–1796, 2009.

[16] G. C. Littlewort, M. S. Bartlett, and K. Lee, "Automatic coding of facial expressions displayed during posed and genuine pain," *Image and Vision Computing*, vol. 27, no. 12, pp. 1797–1803, 2009.

[17] M. S. Bartlett, G. C. Littlewort, M. G. Frank, and K. Lee, "Automatic decoding of facial movements reveals deceptive pain expressions," *Current Biology*, vol. 24, no. 7, pp. 738–743, 2014.

[18] Z. Hammal, M. Kunz, M. Arguin, and F. Gosselin, "Spontaneous pain expression recognition in video sequences," in *Proceedings of Visions of Computer Science-BCS International Academic Conference*, (Swindon, UK), pp. 191–210, 2008.

[19] Z. Hammal and M. Kunz, "Pain monitoring: A dynamic and context-sensitive system," *Pattern Recognition*, vol. 45, no. 4, pp. 1265–1280, 2012.

[20] Z. Chen, R. Ansari, and D. J. Wilkie, "Automated detection of pain from facial expressions: a rule-based approach using aam," in *Proceedings of SPIE – the International Society of Optical Engineering, Medical Imaging 2012: Image Processing*, (San Diego, CA), pp. 1 – 17, 2012.

[21] K. Sikka, A. Dhall, and M. S. Bartlett, "Classification and weakly supervised pain localization using multiple segment representation," *Image and vision computing*, vol. 32, pp. 659–670, 2014.

[22] S. Kaltwang, O. Rudovic, and M. Pantic, "Continuous pain intensity estimation from facial expressions," in *Proceedings of International Symposium on Visual Computing*, (Berlin, Germany), pp. 368–377, 2012.

[23] P. Werner, A. Al-Hamadi, K. Limbrecht-Ecklundt, S. Walter, S. Gruss, and H. C. Traue, "Automatic pain assessment with facial activity descriptors," *IEEE Transactions on Affective Computing*, vol. 8, no. 3, pp. 286–299, 2016.

[24] J. Zhou, X. Hong, F. Su, and G. Zhao, "Recurrent convolutional neural network regression for continuous pain intensity estimation in video," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, (Las Vegas, NV), pp. 84–92, 2016.

[25] P. Rodriguez, G. Cucurull, J. Gonzàlez, J. M. Gonfaus, K. Nasrollahi, T. B. Moeslund, and F. X. Roca, "Deep pain: Exploiting long short-term memory networks for facial expression classification," *IEEE transactions on cybernetics*, pp. 1–11, 2017.

[26] F.-S. Tsai, Y.-L. Hsu, W.-C. Chen, Y.-M. Weng, C.-J. Ng, and C.-C. Lee, "Toward development and evaluation of pain level-rating scale for emergency triage based on vocal characteristics and facial expressions," in *Proceedings of Interspeech Conference*, (San Francisco, CA), pp. 92–96, 2016.

[27] O. Rudovic, V. Pavlovic, and M. Pantic, "Automatic pain intensity estimation with heteroscedastic conditional ordinal random fields," in *Proceedings of 9th International Symposium on Visual Computing*, (Crete, Greece), pp. 234–243, 2013.

[28] Z. Hammal and J. F. Cohn, "Automatic detection of pain intensity," in *Proceedings of the 14th ACM international conference on Multimodal interaction*, (Santa Monica, CA), pp. 47–52, 2012.

[29] P. Werner, A. Al-Hamadi, and R. Niese, "Comparative learning applied to intensity rating of facial expressions of pain," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 28, no. 05, p. 1451008, 2014.

[30] Z. Hammal and J. F. Cohn, "Towards multimodal pain assessment for research and clinical use," in *Proceedings of the ACM 2014 Workshop on Roadmapping the Future of Multimodal Interaction Research Including Business Opportunities and Challenges*, (New York, NY), pp. 13–17, Association for Computing Machinery, 2014.

[31] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[32] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[34] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, pp. 65–93, Elsevier, 1992.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[36] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural networks*, vol. 2, no. 1, pp. 53–58, 1989.

[37] D. E. Rumelhart, J. L. McClelland, P. R. Group, *et al.*, *Parallel distributed processing*, vol. 1. IEEE Massachusetts, 1988.

[38] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[39] V. Turchenko, E. Chalmers, and A. Luczak, "A deep convolutional auto-encoder with pooling-unpooling layers in caffe," *arXiv preprint arXiv:1701.04949*, 2017.

[40] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[41] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[42] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International conference on artificial neural networks*, pp. 44–51, Springer, 2011.

[43] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2014.

[44] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, "Adversarial autoencoders," in *International Conference on Learning Representations*, 2016.

[45] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *2007 IEEE conference on computer vision and pattern recognition*, pp. 1–8, IEEE, 2007.

[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[47] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pp. 2528–2535, IEEE, 2010.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[50] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[51] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*, pp. 609–616, 2009.

[52] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*, pp. 92–101, Springer, 2010.

[53] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

[54] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International conference on artificial neural networks*, pp. 52–59, Springer, 2011.

[55] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, pp. 1520–1528, 2015.

[56] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[57] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 International Conference on Computer Vision*, pp. 2018–2025, IEEE, 2011.

[58] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[59] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[60] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5528–5531, IEEE, 2011.

[61] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[62] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

[63] W. Pei, J. Zhang, X. Wang, L. Ke, X. Shen, and Y.-W. Tai, "Memory-attended recurrent network for video captioning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8347–8356, 2019.

[64] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, "Video captioning with attention-based lstm and semantic consistency," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2045–2055, 2017.

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

[66] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International conference on machine learning*, pp. 1764–1772, PMLR, 2014.

[67] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, Ieee, 2013.

[68] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[69] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[70] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, pp. 1310–1318, PMLR, 2013.

[71] S. Yang, X. Yu, and Y. Zhou, "Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example," in *2020 International workshop on electronic communication and artificial intelligence (IWECAI)*, pp. 98–101, IEEE, 2020.

[72] M. T. Sattari, H. Apaydin, and S. Shamshirband, "Performance evaluation of deep learning-based gated recurrent units (grus) and tree-based models for estimating eto by using limited meteorological variables," *Mathematics*, vol. 8, no. 6, p. 972, 2020.

[73] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[74] S. Zafeiriou, C. Zhang, and Z. Zhang, "A survey on face detection in the wild: past, present and future," *Computer Vision and Image Understanding*, vol. 138, pp. 1–24, 2015.

[75] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 1, pp. 34–58, 2002.

[76] I. Gogić, J. Ahlberg, and I. S. Pandžić, "Regression-based methods for face alignment: A survey," *Signal Processing*, p. 107755, 2020.

[77] S. Alirezaee, H. Aghaeinia, K. Faez, and F. Askari, "An efficient algorithm for face localization," 01 2006.

[78] A. Tolba, A. El-Baz, and A. El-Harby, "Face recognition: A literature review," *International Journal of Signal Processing*, vol. 2, no. 2, pp. 88–103, 2006.

[79] P. I. Wilson and J. Fernandez, "Facial feature detection using haar classifiers," *Journal of Computing Sciences in Colleges*, vol. 21, no. 4, pp. 127–133, 2006.

[80] G. Chow and X. Li, "Towards a system for automatic facial feature detection," *Pattern Recognition*, vol. 26, no. 12, pp. 1739–1755, 1993.

[81] I. Cohen, N. Sebe, A. Garg, L. S. Chen, and T. S. Huang, "Facial expression recognition from video sequences: temporal and static modeling," *Computer Vision and Image Understanding*, vol. 91, no. 1, pp. 160–187, 2003. Special Issue on Face Recognition.

[82] M. Hassaballah and S. Aly, "Face recognition: Challenges, achievements, and future directions," *IET Computer Vision*, vol. 9, pp. 614–626, 08 2015.

[83] X. Zhang and Y. Gao, "Face recognition across pose: A review," *Pattern Recogn.*, vol. 42, pp. 2876–2896, Nov. 2009.

[84] M. Levine and Y. Yu, "Face recognition subject to variations in facial expression, illumination and pose using correlation filters," *Computer Vision and Image Understanding*, vol. 104, pp. 1–15, 10 2006.

[85] T. Valentine, *Cognitive and computational aspects of face recognition: Explorations in face space*, vol. 29. Routledge, 2017.

[86] I. Craw and P. Cameron, "Parameterising images for recognition and reconstruction," in *BMVC91*, pp. 367–370, Springer, 1991.

[87] C. Liu and H. Wechsler, "Face recognition using shape and texture," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 1, pp. 598–603, IEEE, 1999.

[88] H. Dibeklioglu, "Visual transformation aided contrastive learning for video-based kinship verification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2459–2468, 2017.

[89] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, Y. Cun, *et al.*, "Learning convolutional feature hierarchies for visual recognition," *Advances in neural information processing systems*, vol. 23, pp. 1090–1098, 2010.

[90] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 201–208, JMLR Workshop and Conference Proceedings, 2010.

[91] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th international conference on international conference on machine learning*, pp. 399–406, 2010.

[92] Y. Fan, X. Lu, D. Li, and Y. Liu, "Video-based emotion recognition using cnn-rnn and c3d hybrid networks," in *Proceedings of the 18th ACM international conference on multimodal interaction*, pp. 445–450, 2016.

[93] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional lstm with cnn features," *IEEE access*, vol. 6, pp. 1155–1166, 2017.

[94] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2285–2294, 2016.

[95] C. Wang, H. Yang, C. Bartz, and C. Meinel, "Image captioning with deep bidirectional lstms," in *Proceedings of the 24th ACM international conference on Multimedia*, pp. 988–997, 2016.

[96] J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang, "Dimensional sentiment analysis using a regional cnn-lstm model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 225–230, 2016.

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[98] I. Matthews and S. Baker, "Active appearance models revisited," *International journal of computer vision*, vol. 60, no. 2, pp. 135–164, 2004.

[99] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning." `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf/`, 2008. [Online; accessed 19-August-2021].

[100] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[101] S. K. Kumar, "On weight initialization in deep neural networks," *arXiv preprint arXiv:1704.08863*, 2017.

[102] S. Koturwar and S. Merchant, "Weight initialization of deep neural networks (dnns) using data statistics," *arXiv preprint arXiv:1710.10570*, 2017.

[103] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[104] J. Cohen, "Set correlation and contingency tables," *Applied psychological measurement*, vol. 12, no. 4, pp. 425–434, 1988.