VICTORIA UNIVERSITY OF WELLINGTON
*Te Whare Wānanga o te Ūpoko o te Ika a Māui*

School of Engineering and Computer Science
*Te Kura Mātai Pūkaha, Pūrorohiko*

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

# Genetic Programming for Continuous Facial Expression Recognition

Hien Nguyen - 300199540

Supervisor: Qi Chen

Submitted in partial fulfilment of the requirements for
Master of Computer Science.

**Abstract**

Facial Expression Recognition (FER) is a fundamental component of affective computing, serving as a critical technology for interpreting and analyzing human emotions. Developing a FER system using genetic programming for symbolic regression is crucial due to limited research and implementations, particularly for representing emotions in a continuous Arousal-Valence (AV) space. While current FER methods using genetic programming (GP) primarily focus on the categorical classification of facial expressions, there is a need to address the nuanced representation of emotions in the AV space to capture the complex spectrum of human emotions more accurately. This project uses advanced machine learning techniques to explore FER in the continuous AV space. Several approaches are investigated, including feature learning using GP with image descriptors, and symbolic regression (SR) for interpretable model generation. A two-stage GP method is developed that first evolves feature extraction primitives and then applies symbolic regression to the extracted features. A hybrid model integrating EfficientNet for feature extraction with GP-based symbolic regression is also explored. Experiments on the AffectNet dataset demonstrate the effectiveness of the GPSR-based feature learning approach, achieving the lowest test loss among all the examined models. The symbolic regression models provide more interpretable solutions while maintaining competitive performance. Analysis of the evolved features and equations offers insights into the key facial regions and mathematical relationships captured by the models for AV prediction. This work advances the state-of-the-art in FER by leveraging evolutionary computation techniques to automatically learn discriminative features and generate explainable models for continuous emotion recognition.

# Contents

# List of Figures and Tables

# Chapter 1
# Introduction

The study of FER is essential for developing systems that can accurately perceive and respond to human emotional states, bridging the gap between human-computer interactions and emotional intelligence. This project explores FER in the continuous arousal-valence (AV) space, which provides a nuanced representation of emotions beyond basic categorical labels. The AffectNet database is a rich resource for this investigation, enabling the development and evaluation of advanced machine learning techniques for continuous emotion recognition in unconstrained environments.

The Affect from the InterNet database is the most extensive collection of categorical and dimensional affect models in unconstrained environments. It was built by searching for emotion-related keywords on Google, Bing, and Yahoo and then using expert human annotators to label the images. The process began by translating the original English search queries into Spanish, Portuguese, German, Arabic, and Farsi. Non-human images were removed, facial landmarks were extracted, and the facial expressions, valence, and arousal were annotated. In total, 450,000 images were annotated by 12 full-time and part-time expert annotators who received thorough training and close supervision to maintain high annotation quality. The annotation was performed using a software application that enabled the annotators to label categorical emotions and dimensional affect (valence and arousal) using a 2D Cartesian coordinate system. This database is invaluable for affective computing and emotion recognition research [4].

## 1.1. Problem statement

Facial Expression Recognition (FER) is a crucial field within affective computing, providing invaluable insights into the complex landscape of human emotions. As technological progress increasingly emphasizes the need for systems capable of emotional intelligence, FER emerges as a key component in interpreting facial signals to gain a more profound understanding of emotional states. Traditional categorical emotion labels (e.g., joy, anger, sorrow) fail to capture the full range of human emotional experiences in everyday life. To address this limitation, psychological researchers often employ dimensional frameworks, particularly the concepts of valence and arousal. Valence measures the degree of positivity in an emotional expression, while arousal gauges the intensity or excitement level of the displayed emotion [5]. FER in the continuous AV space remains a challenging problem due to the complexity of capturing nuanced emotions in unconstrained environments. Existing approaches often rely on manual feature extraction or end-to-end deep learning, which can lack interpretability and require large annotated datasets. There is a need for more efficient and explainable methods that can automatically learn discriminative features from limited data while providing insights into the critical facial regions and mathematical relationships involved in AV prediction. This project addresses these challenges by investigating advanced machine learning techniques, including genetic programming (GP) for feature learning, symbolic regression for interpretable model generation, and hybrid approaches integrating deep learning and evolutionary computation. By leveraging the AffectNet database and exploring novel feature learning and symbolic regression methods, this work seeks to advance state-of-the-art continuous emotion recognition and contribute to understanding facial expressions in the AV space.

## 1.2. Goals

The main objectives of this project are:

- Investigate various approaches for FER in the AV space, including genetic programming with image descriptors for feature learning, and applying symbolic regression to generate interpretable models.
- Explore the potential of genetic programming with image descriptors for automated feature learning and the applicability of symbolic regression in generating interpretable models, that can uncover the representations of facial expressions.
- Interpret the learned models and examine the features and equations produced by the evolutionary models to identify the crucial facial areas and mathematical patterns the models identify for predicting arousal and valence.
- Discuss and compare the results between these different methodologies to identify their attributes, contributions, and compatibility in continuous emotion recognition.

## 1.3. Organizations

The rest of the research is organized as follows: Chapter 2 introduces the broad background and various techniques used in FER systems. Chapter 3 mentions the data preparation process in Section 3.1, the exploratory data analysis in Section 3.2, and a discussion on data preprocessing algorithms in Section 3.3. Chapter 4 covers baseline - deep learning for FER in AV space using the EfficientNet-based model, while feature learning using genetic programming (GP) with image descriptors (IDGP), is described in Chapter 5. Chapter 6 further explores the method in Chapter 5 by using genetic programming for symbolic regression for interpretable model generation (we call it FD-GPSR, Feature Descriptor-based Genetic Programming Symbolic Regression). Chapter 7 highlights a two-stage GP method that first evolves feature extraction primitives and then applies symbolic regression to the extracted features (Combined Feature Symbolic Regression, CFSR). Chapter 8 describes a hybrid model integrating EfficientNet for feature extraction with GP-based symbolic regression. Finally, the paper concludes in Chapter 9.

# Chapter 2
# Background

Facial Expression Recognition (FER) is a crucial area in affect computing, playing a vital role in understanding human emotions. This chapter explores the landscape of FER, focusing on both conventional and deep learning approaches within the arousal-valence (AV) space. It discusses the challenges of capturing continuous emotion dimensions, highlights the importance of datasets like AffectNet, and explores the use of evolutionary computation techniques such as Genetic Algorithms and Genetic Programming for optimizing feature selection and classifier performance. The chapter also outlines the potential of these methods to enhance the accuracy and interpretability of FER systems.

## 2.1. Facial expression recognition overview

### 2.1.1. Conventional FER approaches

Conventional methods for FER involve several key stages, each playing a crucial role in the process. The first stage, image preprocessing, is essential for enhancing the detection of relevant information while removing irrelevant data. This stage typically includes steps such as noise reduction, face detection, face alignment, normalization, and histogram equalization. Following preprocessing, feature extraction is performed to derive valuable data from the images, including values, vectors, and symbols. Common techniques for feature extraction include Facial Landmarks, Gabor feature extraction, Local Binary Pattern (LBP), and Histogram of Oriented Gradients (HOG). The next stage, feature selection, aims to identify the most discriminative features for accurate classification. This is achieved through methods such as support vector machine recursive feature elimination (SVM-RFE), empirical normalized distances approach, and Correlation Features Selection (CFS). The final stage in conventional methods is image classification, where appropriate classifiers are selected to predict facial expressions successfully. Popular classifiers used in this stage include k-Nearest Neighbours (kNN), Support Vector Machine (SVM), and Adaptive Boosting (AdaBoost), among others.

### 2.1.2. Deep learning-based FER approaches

Deep learning approaches employ an ”end-to-end” learning process from input to classification, minimizing the need for manual feature extraction. These methods have become increasingly popular for dealing with the complexities of recognizing emotions in natural, in-the-wild settings.

Deep learning-based FER encompasses several key aspects. The preprocessing stage typically employs advanced methods such as Cascaded Convolutional Neural Networks or Multi-Task Cascaded Convolutional Neural Networks (MTCNN) for accurate facial landmark detection and alignment. Data augmentation plays a crucial role in enhancing the diversity and size of training datasets, utilizing both on-the-fly and offline augmentation techniques. The core of deep learning-based FER lies in its ability to learn deep features for categorizing images into basic emotion categories, which is achieved through various network architectures, including Convolutional Neural Networks (CNNs), Deep Belief Networks

(DBNs), and Generative Adversarial Networks (GANs).

## 2.2. FER in continuous AV space

The dimensional model of affect describes emotions along continuous spectra of intensity and variety, but research on automated algorithms for measuring affect in this continuous model, particularly for valence and arousal, is limited. A significant challenge in this field is the high cost and complexity of developing comprehensive databases spanning the full range of valence and arousal in a continuous context, compounded by the scarcity of continuously annotated facial expression databases. Despite these challenges, Facial Expression Recognition (FER) in the continuous dimension has seen considerable advancements with various machine learning and deep learning techniques, with the Arousal-Valence (AV) space emerging as a globally recognized model for continuous emotion detection.

Critical aspects of Facial Expression Recognition (FER) in continuous Arousal-Valence (AV) space encompass several key elements. Datasets play a crucial role, with notable examples including AffectNet, AFEW-VA, SEWA, and DEAP, all of which provide rich annotations for continuous affect dimensions. Methodologies in this field are diverse and innovative. Researchers have proposed various approaches, such as using multiple stacks of bidirectional long short-term memory (DBLSTM-RNN) to capture temporal dynamics. Some methods integrate multiple modalities, combining facial expressions with shoulder movements and audio cues for a more comprehensive analysis. The application of Continuous Conditional Random Fields has also shown promise in modeling the interdependencies between continuous emotion dimensions. Additionally, leveraging pre-trained models for direct regression prediction of real-valued outcomes has emerged as an effective strategy in continuous AV space FER.

## 2.3. Evolutionary Computation for FER

Evolutionary computation techniques, particularly Genetic Algorithms (GA) and Genetic Programming (GP), have been applied to optimize feature selection and classifier efficacy in FER.

### 2.3.1. Genetic Algorithm (GA)

Genetic Algorithms (GA) have found significant applications in feature selection for Facial Expression Recognition (FER), often combined with other techniques to enhance performance. Key applications include combining GA with non-dominated classification genetic algorithm II (NSGA-II) and Cuckoo binary search, using GA to determine the most compelling features from specific facial regions, and employing GA for selecting features and fine-tuning hyperparameters of classifiers like Support Vector Machines (SVM).

### 2.3.2. Genetic Programming (GP)

Genetic Programming (GP) extends GA's capabilities for complex classification tasks in FER. Notable applications include feature selection, where GP-based frameworks like GP-FER use tree-based genetic programs as binary classifiers for each pair of expression classes. GP has also been used for feature learning, constructing high-level features while training classifiers. Additionally, GP-based methods have been developed for image classification, incorporating various image filters and operators for feature extraction.

### 2.3.3. Genetic Programming for Symbolic Regression (GPSR)

In the realm of Symbolic Regression (SR), GP has been applied to SR in various ways. Contemporary SR methods include traditional GP-based approaches, Pareto optimization, semantic methods, and gradient-based techniques. Generalization in GP for SR focuses on improving data quality, estimating generalization errors, enhancing search mechanisms, and using ensemble learning methods. To enhance

interpretability, approaches have been developed to create more interpretable GP model structures, including the use of interpretable function nodes, interpretable combinations between nodes, and problem decomposition techniques.

## 2.4. Challenges and future directions

Facial Expression Recognition (FER) is a rapidly advancing field that has significant implications for affect computing and human-computer interaction. Integrating advanced machine learning techniques, particularly deep learning and evolutionary computation methods, has substantially improved FER performance. However, several challenges and potential future directions in FER research include handling high-dimensional data, balancing model complexity and interpretability, addressing data scarcity, improving generalization across different datasets and real-world scenarios, and effectively integrating multimodal information from facial expressions, audio, and physiological signals. Additionally, there is potential for developing more sophisticated Genetic Programming (GP) based methods for FER, particularly in continuous emotion recognition, and interpretable and generalizable model generation. Enhancing the real-time performance and efficiency of FER systems is also crucial for their practical application in various domains.

## 2.5. Practical methods applied in FER systems

### 2.5.1. Image preprocessing

Image preprocessing plays a crucial role in image classification systems. Since images are often acquired under varying conditions, the raw images may suffer from noise or suboptimal contrast, rendering them unsuitable for direct analysis. Image preprocessing aims to enhance the image quality or transform the images to meet specific requirements, ensuring they are better suited for the subsequent classification tasks [2].

#### 2.5.1.a. Face detection, eye detection and face alignment

Computer vision enables advanced tasks such as face detection and face recognition. Face detection involves locating faces within an image, while face recognition identifies the detected face as belonging to a specific person [6]. OpenCV implements various algorithms for these tasks, which have diverse real-world applications [7].

OpenCV comes with pre-trained Haar and LBP detectors for detecting various facial features. From version 4.0 onwards, deep learning has been integrated into the core of OpenCV. A pre-trained Caffe model based on the Single Shot Multibox Detector (SSD) algorithm can be used for face detection, which enables the detection of multiple objects in an image using a single deep-learning network [6].

Eye detection plays a crucial role in face preprocessing, providing a reliable reference for normalizing and validating facial images. By leveraging the consistency of eye positions in frontal faces, eye detection can be used to discard false positives generated by the face detector. Ensuring consistent face alignment in the dataset is essential for accuracy. Eye detection is employed to precisely align the positions of the two detected eyes, achieving better alignment than face detection alone. The geometrical transformation is performed using the warpAffine() function, which rotates, scales, and translates the face to center the eyes horizontally at the desired height [6].

#### 2.5.1.b. Histogram equalizer

Image equalization (example shown in Figure 2.1), or histogram equalization, is a technique that enhances image contrast by achieving a uniform distribution of pixel values in an image's histogram. This process is particularly beneficial for pictures with extreme brightness or darkness or those with

**Figure 2.1:** Face without using histogram equalizer or denoising (left), face with histogram equalizer (middle), face with non-local means denoising (right).

minimal distinction between foreground and background elements. However, this technique can have drawbacks, including increased background noise and a potential reduction in the clarity of essential image features [6].

#### 2.5.1.c. Smoothing

Smoothing and blurring are essential image processing techniques used to reduce image noise by diminishing high-frequency components and sharp details. Noise removal is crucial as it can lead to the detection of false objects during image segmentation [8]. Digital images are subject to color-value perturbations due to the random nature of photon counting in sensors, which can be amplified by digital corrections or image processing software [9].

One effective denoising method is non-local means, which replaces a pixel's color with an average of colors from similar pixels, not necessarily spatially close. This method preserves delicate structures and details while removing noise. The non-local means algorithm can be implemented using pixel-wise or patchwise approaches [9]. The cv.fastNlMeansDenoising() function uses a pixel-wise non-local mean denoising algorithm. The pixel-wise implementation computes the denoised color value for each pixel as a weighted average of similar pixels in a neighborhood, using an exponential weight function that depends on the squared Euclidean distance between patches centered at the target pixel and neighboring pixels. The denoising process is similar for color and grayscale images, with the main difference being the computation of the squared Euclidean distance between patches using all three color channels for color images and only intensity values for grayscale images [9].

### 2.5.2. Image feature extraction and description

Image descriptors are specialized tools developed by domain experts to extract and describe compelling image features for specific tasks. These descriptors may comprise a set of operations or mathematical equations designed to detect critical points and describe informative image features. The process of transforming images into features using these descriptors is known as feature extraction, which aims to extract informative and discriminative information while reducing image data dimensionality [2]. Image features are broadly categorized into global (holistic) features, extracted from the entire image based on all pixel values, and local features, extracted from critical points or regions of interest [10]. Both global and local feature extraction processes may employ various image descriptors, with representative examples including Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), Scale Invariant Feature Transform (SIFT), and Gabor filters.

#### 2.5.2.a. Histogram

The histogram method is a straightforward and efficient feature extraction technique that represents the distribution of pixel values in an image as a feature vector. For an image of dimensions M × N with G gray levels, the histogram is created by setting several "bins," typically equal to G. The value of each bin is determined using the following formula:

$$H(g) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f(I(i,j), g), \quad g \in [0, G-1], \quad f(x,y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

For grayscale images, the number of bins is usually set to 256, corresponding to the range of pixel values (0 to 255). The total number of histogram values equals the number of pixels in the grayscale image [2].

#### 2.5.2.b. Histograms of Oriented Gradients (HOGs)

The Histogram of Oriented Gradients (HOG) algorithm is a feature descriptor widely used in computer vision and machine learning for object detection. It describes an object's structural shape and appearance in an image by computing gradient orientation occurrences in localized image portions. The HOG algorithm consists of five main stages:

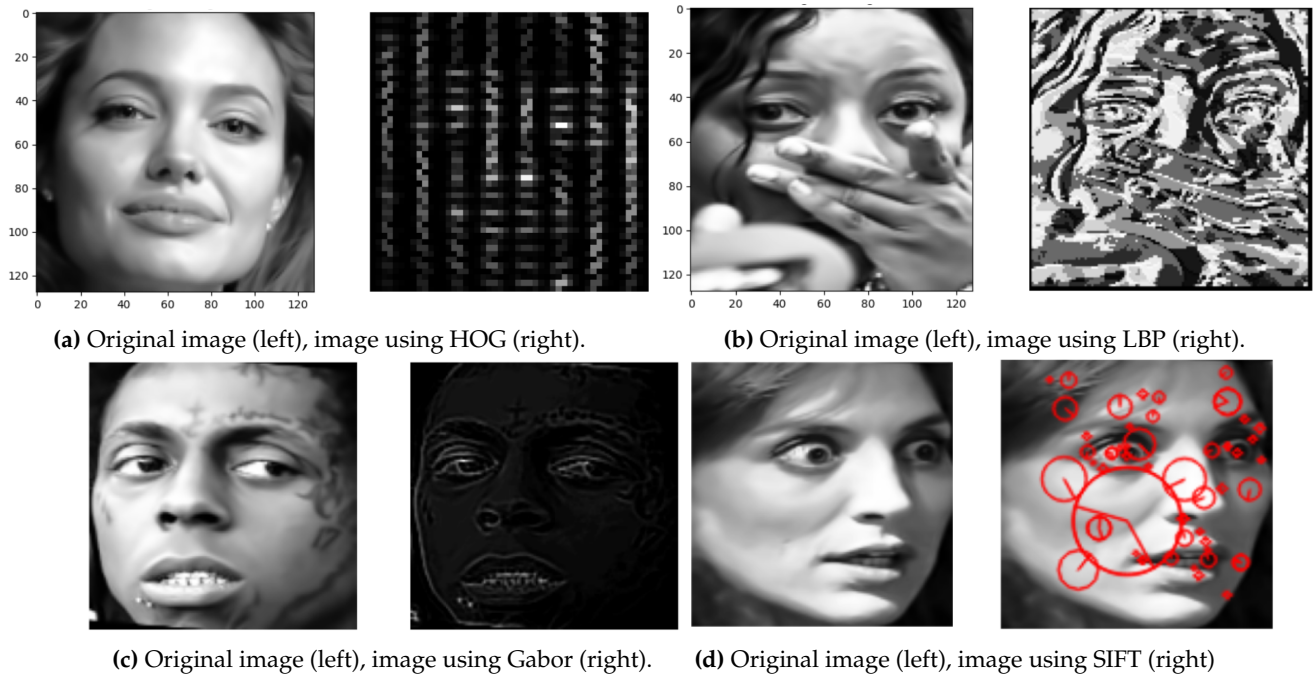
- Stage 1: Global Image Normalization: This optional stage reduces illumination effects using gamma compression, square-root normalization, or variance normalization.
- Stage 2 Gradient Computation: The algorithm calculates first-order image gradients in the x and y directions to capture contour, silhouette, and texture information. These gradients are then used to compute gradient magnitude and orientation.
- Stage 3: Gradient Histogram Computation: The image is divided into small spatial regions called cells. For each cell, a local 1D histogram of gradient orientations is created, with gradient magnitudes used as weights for voting into orientation bins.
- Stage 4: Block Normalization: Cells are grouped into larger blocks, which typically overlap. Normalization is performed over these blocks by accumulating a measure of local histogram energy, resulting in the HOG descriptors.
- Stage 5: Feature Vector Creation: The normalized block descriptors are concatenated to form the final feature vector [8].

#### 2.5.2.c. Local Binary Patterns (LBP)

The Local Binary Patterns (LBP) feature descriptor is used to classify image textures. The LBP extraction process involves comparing each pixel's value to its surrounding pixels, assigning binary values based on the comparisons, creating an 8-bit binary number, and converting it to decimal to obtain a new value for the central pixel. This process is repeated for all pixels, and a histogram is generated from the resulting LBP array to serve as the final LBP feature vector. An enhanced version of LBP introduces two additional parameters: the number of points (p) in a circularly symmetric neighborhood and the circle's radius (r), allowing LBP to adapt to textures of varying scales, with smaller radii capturing finer details and larger radii classifying broader texture patterns [8].

#### 2.5.2.d. Scale Invariant Feature Transform (SIFT)

The Scale-Invariant Feature Transform (SIFT) [11] method is widely used for keypoint detection and description in image matching and object detection tasks. The SIFT algorithm consists of five main steps:



**(a)** Original image (left), image using HOG (right).

**(b)** Original image (left), image using LBP (right).

**(c)** Original image (left), image using Gabor (right).

**(d)** Original image (left), image using SIFT (right)

**Figure 2.2:** Examples of different image descriptors.

scale-space extrema detection, keypoint localization, orientation assignment, keypoint description, and feature vector normalization. The resulting SIFT features are highly robust against scale, rotation, and illumination variations. Variants of SIFT, such as dense SIFT [12] and Speeded Up Robust Features (SURF) [13], have been developed to improve computational efficiency while maintaining similar performance characteristics [2].

#### 2.5.2.e. Gabor filter

The Gabor filter is a linear filter with a kernel that resembles the two-dimensional receptive field patterns found in mammalian cortical simple cells [14]. It is highly valued in computer vision and image analysis due to its exceptional localization capabilities in both spatial and frequency domains [15]. The Gabor filter kernel function is a Gaussian kernel function modulated by a sinusoidal plane wave [14].

### 2.5.3. Regression

Supervised machine learning is a widely used and successful approach in which a model is trained on labeled input/output pairs to predict outcomes for new, unseen data. This method automates and often accelerates tasks that would otherwise be laborious or infeasible, though it initially requires human effort to build the training set. There are two main types of supervised learning problems: classification and regression. Classification aims to predict a class label from a predefined list, and it can be further divided into binary (two classes) or multiclass (more than two classes) classifications. Conversely, regression predicts continuous numerical values, such as a person's income or crop yields [16].

#### 2.5.3.a. Support Vector Machines (SVM)

Support Vector Machines (SVMs) are versatile machine learning models that handle classification, regression, and novelty detection tasks. They excel in dealing with small to medium-sized non-linear datasets, especially for classification problems. SVMs transform input data into a higher-dimensional space using kernel functions, making the data linearly separable. The algorithm then finds the optimal hyperplane that maximizes the margin between different classes, with the nearest training instances

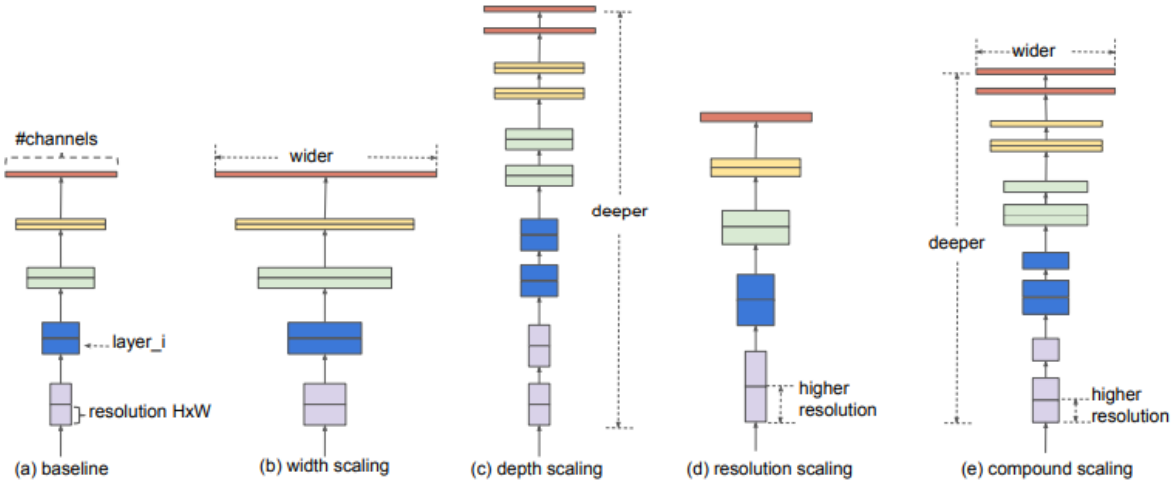



**Figure 2.3:** Model Scaling. (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one network width, depth, or resolution dimension. (e) is the proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio[1].

to this hyperplane called support vectors. For regression tasks, SVMs fit as many instances as possible within a margin (controlled by the parameter $\epsilon$) while limiting violations, making the model $\epsilon$-insensitive [17].

#### 2.5.3.b. Deep transferred learning: EfficientNet

Transfer learning is a crucial technique in modern machine learning, especially when working with deep neural networks (DNNs). It involves transferring knowledge from one task to another, which is particularly useful when training large DNNs. Instead of training a complex network from scratch, it's best to search for an existing neural network that performs a similar task and reuse most of its layers, except for the top ones. This approach significantly accelerates training and reduces the required training data [17].

[1] introduces a systematic approach to scaling neural networks, using a set of fixed scaling coefficients to uniformly increase network width, depth, and resolution (Figure 2.3). This compound scaling method adjusts network dimensions using a single compound coefficient $\phi$, simultaneously modifying depth (d), width (w), and resolution (r) according to the equations:

$$d = \alpha^{\phi}, \quad w = \beta^{\phi}, \quad r = \gamma^{\phi} \ (2)$$

The constants $\alpha, \beta, and \gamma$ are determined through a small grid search and must satisfy the constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$, with each constant being greater than or equal to 1.

The user-defined coefficient $\phi$ controls the overall scaling of the network, while $\alpha, \beta, and \gamma$ determine how additional resources are allocated to each dimension. This approach considers the relationship between network dimensions and computational cost, as the Floating point operations per second (FLOPS) of a typical convolution operation are proportional to $d$, $w^2$, and $r^2$. By constraining $\alpha \cdot \beta^2 \cdot \gamma^2$ to approximately 2, the method ensures that for any given $\phi$, the total FLOPS of the network will increase by approximately $2^{\phi}$, allowing for controlled scaling of the network's computational requirements while balancing improvements across all dimensions. This compound scaling method is used to develop the EfficientNet model family, starting from a baseline model (EfficientNet-B0) and scaling it up using different values of $\phi$ to generate progressively larger and more powerful models (EfficientNet-B1 to B7).

#### 2.5.3.c. Genetic Programming and Symbolic Regression techniques

Genetic Programming (GP) [18] is an Evolutionary Computation technique that automatically evolves executable computer programs to solve problems. GP uses a variable-length representation without predefined solution structures and employs principles of biological evolution, including selection, reproduction/elitism, crossover, and mutation. As shown in Figure 2.4, the algorithm aims to find the best solution through multiple generations using a fitness function to evaluate individuals and guide the search process [2].

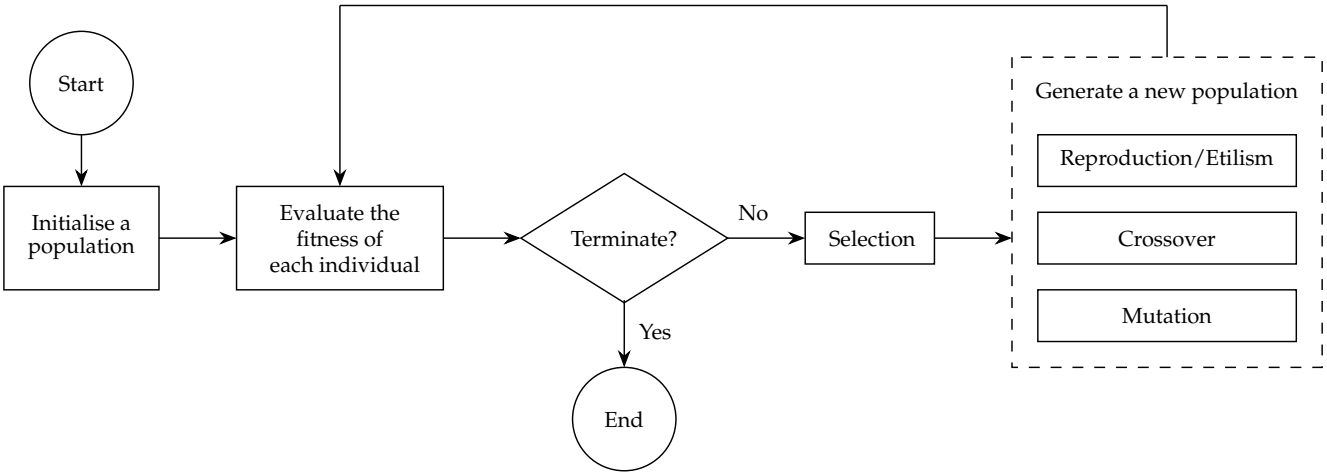



**Figure 2.4:** The flowchart of GP [2].

GP has several well-known variants with different representations, including Tree-based GP [18], Linear GP [19, 20], Grammatically-based GP [21], and Cartesian GP [22]. The most commonly used version is tree-based GP, where each computer program (individual) is represented as a syntax tree constructed from primitives, including functions and terminals. Functions are root and internal nodes, while terminals are leaf nodes (Figure 2.5). This variable-length tree-based representation offers flexibility and adaptability, making it well-suited for representing potential solutions and facilitating optimal solution searches in various problem domains [2].

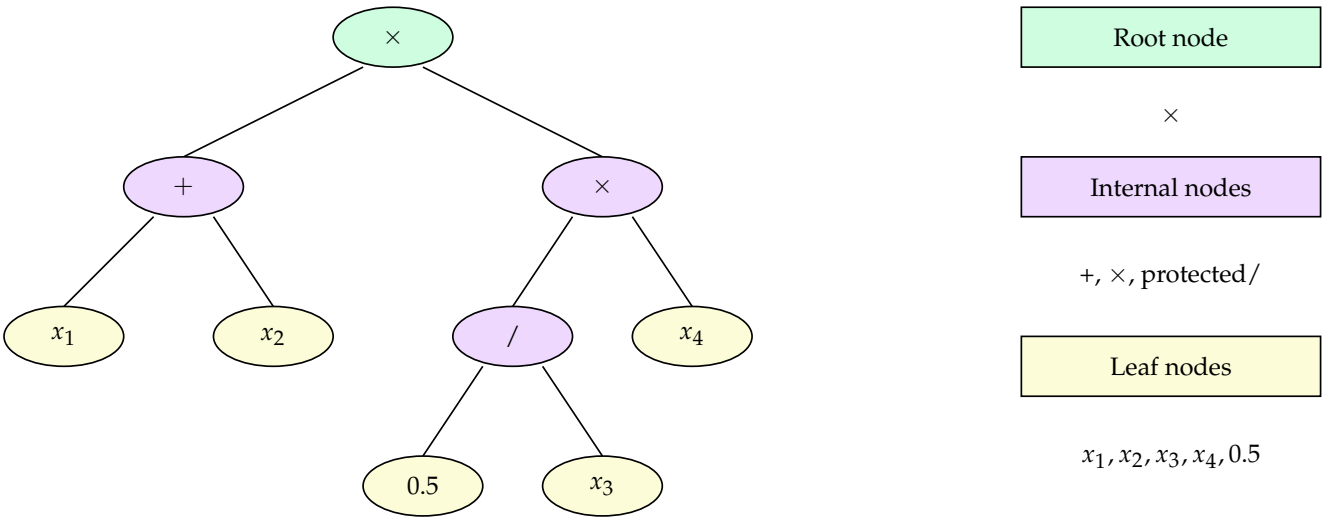



**Figure 2.5:** An example GP tree [2].

To represent a GP tree, both a function set and a terminal set must be defined. The terminal set includes variables, features, attributes, and Ephemeral Random Constants (ERCs) related to the problem, serving as inputs to the GP system. The function set consists of operations used to build the internal

and root nodes of GP trees, including general functions and domain-specific functions. Sufficiency and closure are two main criteria to be considered when constructing function and terminal sets to ensure adequate solution expression and runtime error prevention [2].

The initial population in GP is typically generated randomly using three main methods: full, grow, and ramped half-and-half. The full method creates trees with all leaf nodes at the maximum depth, while the grow method produces trees with variable leaf node depths. The ramped half-and-half method combines full and grow methods to create a diverse population with a wide range of tree sizes and shapes (Figure 2.6) [2].

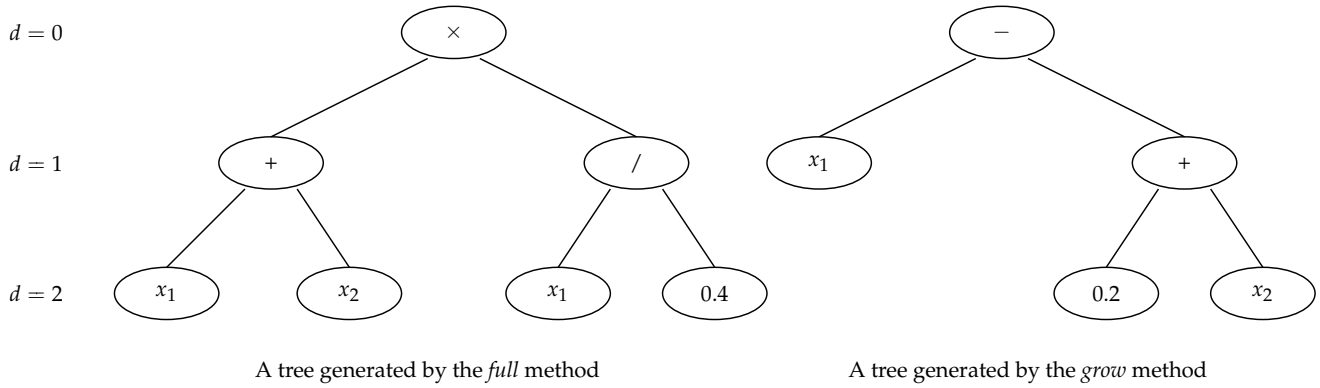



**Figure 2.6:** Examples to show the full and grow method. The maximum tree depth (d) is 2 [2].

Fitness evaluation plays a crucial role in the evolutionary process of GP by guiding the search for improved individuals using a fitness function to assess the quality of each individual in the population. Selecting an appropriate fitness function is essential for efficiently guiding the search toward optimal solutions, and the fitness function is generally tailored to the specific problem being addressed [2].

The evolutionary process employs a selection method to choose individuals from the current population for genetic operations, with individuals having higher fitness having a greater probability of being selected. Various selection methods have been proposed and utilized, with tournament selection [18, 23, 24] being the most commonly used method in GP. This method sets a tournament size to determine the maximum number of randomly chosen individuals for each selection round and selects the individual with the best fitness from this randomly sampled group [2].

Genetic Programming (GP) uses reproduction, crossover, and mutation operators to evolve solutions. Reproduction copies fit individuals, while elitism preserves the best. Crossover recombines genetic material from two parent trees using subtree swapping. Mutation modifies a tree by replacing a randomly selected branch with a new subtree, maintaining diversity. These operators work together to improve the population over generations [2].

Montana [25] introduced Strongly Typed Genetic Programming (STGP) in 1995 as an enhanced version of traditional tree-based GP to address the limitations of handling only one data type. The closure criteria of functions in traditional GP restricted its applicability to complex problems requiring multiple data types [2]. In STGP, terminals and functions are assigned specific input and output types, guiding the construction of GP trees and ensuring compatibility. The evolutionary process, including initialization, crossover, and mutation, maintains these type constraints. STGP has been successfully applied to various tasks such as feature extraction [26], edge detection [27], and image classification [28–30], demonstrating its flexibility and power in addressing complex problems across different domains.

Regression involves fitting a predefined function to a dataset by determining its coefficients. Manually trying different functions can be labor-intensive and biased towards simpler models. Symbolic regression, using Genetic Programming (GP), finds a function that fits the data without assuming a specific structure, making it suitable for complex discovery tasks. Symbolic regression remains an active research area in GP [31].

[3] categorize SR methods in the following manner: regression-based methods, expression tree-based

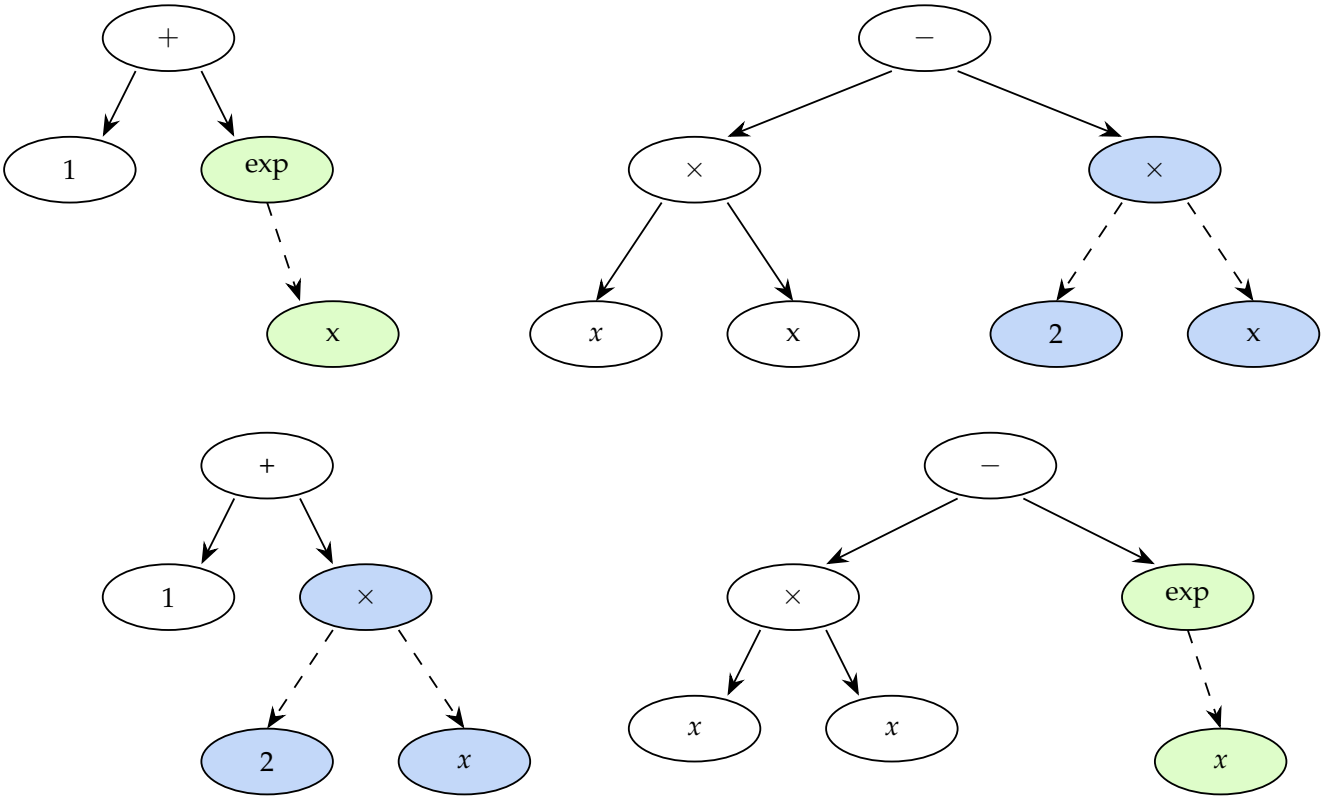



**Figure 2.7:** Crossover operations on exemplary expression trees in GP [3].

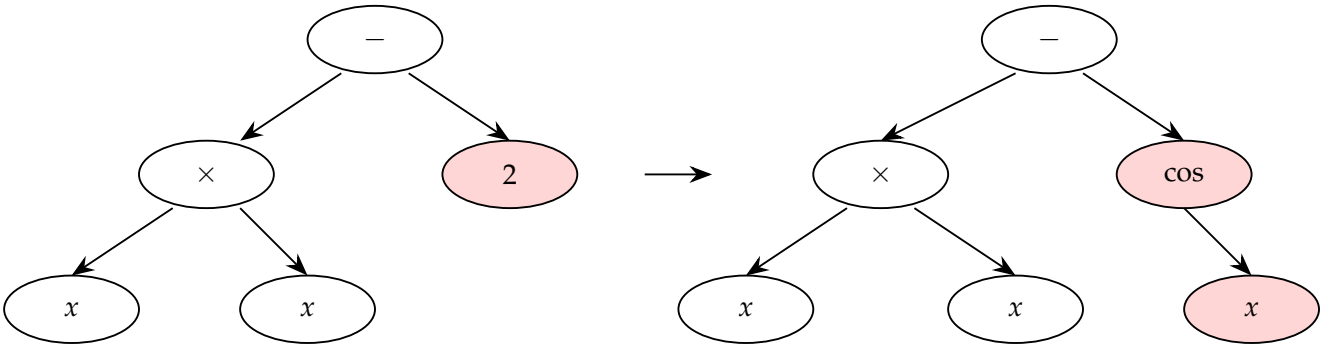



**Figure 2.8:** Mutation operations on exemplary expression trees in GP [3].

methods, physics-inspired, and mathematics-inspired methods.

Tree-based methods are popular for better representation of mathematical expressions. Expression tree-based methods represent mathematical expressions as unary-binary trees, with operators as internal nodes and operands (variables or constants) as terminals. Genetic programming (GP), a critical approach in this category, is an evolutionary algorithm that searches the space of computer programs to solve a given problem. GP starts with a randomly generated ”population” of ”individuals” (trees) and evolves it using evolutionary ”transition rules” (operations) such as mutation, crossover, and selection. Mutation introduces random variations by replacing subtrees (Figure 2.7), while crossover exchanges content between individuals (Figure 2.8). Selection, often using tournament selection, determines which individuals persist to the next generation. In each generation, individuals are likely to undergo mutation and crossover, and the process continues for a predetermined number of iterations or until a desired accuracy level is achieved. Although GP allows for significant variations in the population, leading to improved performance on out-of-distribution data, it does not scale well to high-dimensional datasets. It is sensitive to hyperparameters [3].

# Chapter 3
# Data Preprocessing

## 3.1. Data preparation

The AffectNet database is an extensive collection of images (122GB) that requires a reliable and fast internet connection for downloading and storage [4]. To make the database more accessible, the authors have released two versions. One version of the AffectNet with eight labels, a smaller subset containing 291,651 manually annotated images. Each image is labeled with one of eight expressions: Neutral (labeled as 0), Happiness (1), Sadness (2), Surprise (3), Fear (4), Disgust (5), Anger (6), or Contempt (7). The photos in this version have been cropped and resized to a uniform size of 224 x 224 pixels in RGB color format. In addition to the expression labels, the database includes arousal and valence values for each image, provided as floating-point numbers ranging from $-1$ to $+1$. Facial landmark points for the training and validation sets are also included. The total size of this version is approximately 4GB, and it is distributed as two tar archive files: train_set.tar and val_set.tar. After extracting these tar files, the images and their corresponding labels can be found in the "images" and "annotations" folders.

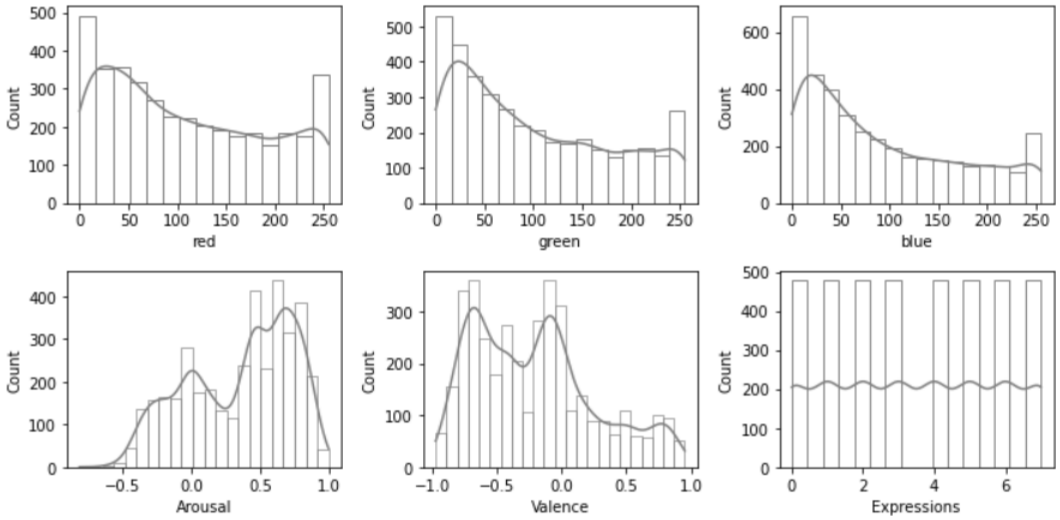
As images and their annotations are stored in different folders, and each image has three annotation files: one for arousal with format imagename_aro.npy, one for valence (imagename_val.npy) and the other one for expression (imagename_exp.npy), it is essential to get the label values, sort and store them accordingly.

Due to limited resources, only a random subset of images will be selected for analysis. There will be two datasets: the first consists of 50 images per emotion, resulting in a total of 400 training images and a test set of 80 images, with 10 images for each emotion. The second dataset contains 3,840 training images, with 480 images for each emotion.

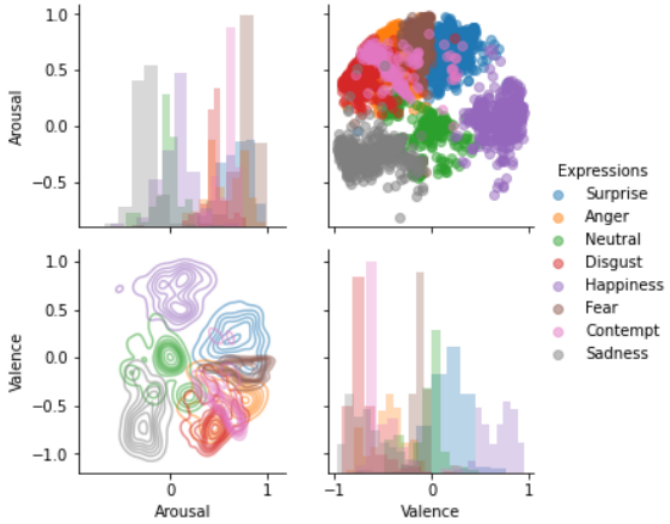
## 3.2. Exploratory data analysis

Exploratory Data Analysis (EDA) is a powerful tool for understanding, visualizing, and extracting meaningful insights from diverse datasets. This critical process enhances our comprehension of the underlying data structures, guides subsequent data preprocessing steps, and refines prediction models. By revealing patterns, anomalies, and relationships within the data, EDA lays a solid foundation for more informed decision-making in the model development process, ultimately leading to more accurate and robust predictive outcomes [32].

As shown in Figure 3.1a, the color channel distributions (red, green, and blue in the first row) show similar right-skewed patterns, with peaks at different intensity levels (red at 50-75, green and blue at 25-50), suggesting that converting to grayscale can simplify the data to a single intensity distribution, reducing dimensionality from 3 channels to 1, which can simplify processing and potentially improve computational efficiency. The color channel histograms show that pixel intensities are not uniformly distributed across the full range (0-255), indicating that histogram equalization could be useful to improve contrast, make features more distinct, and distribute intensities more evenly. Given the different scales and distributions of color channels, proper normalization of input features is essential to prevent




**(a)** Distribution of RGB values and emotional dimensions.




**(b)** Emotional expressions mapped by Arousal and Valence.

**Figure 3.1:** AffectNet EDA.

any channel from dominating the others. The arousal histogram shows a roughly normal distribution centered slightly above 0, suggesting that the dataset contains a balanced range of arousal levels, with a slight tendency towards higher arousal. The valence histogram appears bimodal, with two distinct peaks - one in the negative range and one in the positive range, indicating a good mix of positive and negative emotional states in the dataset.

The Arousal vs. Valence Scatter Plot, Figure 3.1b, shows the relationship between these two dimensions. The points form a roughly circular pattern, indicating a weak or no correlation between arousal and valence. Thus, these two dimensions capture different aspects of emotion independently, and we should treat arousal and valence as separate prediction targets. Multi-output regression models or two separate models for each dimension could be considered. The density contour plot further emphasizes the circular distribution of emotions in this space, suggesting that non-linear models (e.g., Neural Networks) outperform linear models.

## 3.3. Data preprocessing

Image preprocessing enhances detection by removing irrelevant data, reducing noise, and normaliz-

**Figure 3.2:** Raw image (left), image after face detection and face alignment steps (right).

ing images. Key steps include noise reduction using filters, face detection and face alignment, scaling, grayscale adjustments, and histogram equalization. Detailed algorithms of image preprocessing are depicted in Algorithm 1 and Algorithm 2. These techniques prepare images for further analysis, significantly impacting feature extraction and expression classification performance.

The specific process involves:

- Initializing variables and defining image size of 128x128 pixels.
- Preprocessing training images with face detection, face alignment and different configurations for histogram equalization and noise reduction.
- Resizing and saving the dataset to a different image size or configurations if needed.

Face detection and alignment are crucial preprocessing steps for facial expression recognition that ensure only relevant facial regions are processed. They reduce noise from background elements and standardize face position and orientation, making it easier for models to learn consistent features [33]. Resizing images to a standard size (128x128 pixels) ensures consistent input dimensions for the neural network, reduces computational requirements and memory usage and helps balance detail preservation and efficiency. Converting images to grayscale reduces data dimensionality from 3 channels to 1, simplifying processing, which is often sufficient for facial expression recognition as color information is less critical. Applying histogram equalization enhances image contrast, making facial features more distinct, normalizes brightness, and improves feature visibility, leading to better feature extraction and improved model performance [33]. Using non-local means of denoising removes image noise while preserving essential edge details and improves overall image quality [9]. Saving preprocessed images in HDF5 format provides efficient storage and quick access to large datasets, allows for easy handling of image data and associated labels (arousal, valence, expression), and streamlines the data loading process during model training.

These preprocessing steps aim to standardize the input data, reduce irrelevant information, and enhance important features. The AffectNet dataset, being one of the largest and most comprehensive datasets for FER in the wild, benefits significantly from these preprocessing techniques. By applying methods like face detection and alignment, image resizing, grayscale conversion, histogram equalization, and noise reduction, the dataset is optimized for training models that can effectively recognize facial expressions under diverse real-world conditions.

**Algorithm 1** Image Preprocessing

**Require:** Original images from AffectNet dataset
**Ensure:** Preprocessed images

**function** KEEPORIGINALFACE(original_image)
    Input: original image
    Convert original image to face2 (no actual transformation)
    **return** face2
**end function**
**function** FACECROPPING(ROI)
    Input: ROI (region of interest)
    Convert ROI from BGR to RGB color space
    Convert the RGB image to grayscale
    **return** the grayscale image
**end function**
**function** IMAGEPREPROCESSING(image_data, complex_images, histogram_equalizer, noise_reduction)
    Open image_data and convert to grayscale
    Convert grayscale image to BGR color space
    Resize image to $100 \times 100$ and create a blob
    Perform face detection using OpenCV Deep Neural Network module
    **if** face detected with confidence $> 0.5$ **then**
        Extract region of interest (ROI)
        Attempt to detect eyes in ROI
        **if** two eyes are detected **then**
            Calculate the angle between the eyes
            **if** angle is within acceptable range **then**
                Rotate image to align eyes horizontally
                Crop face according to bounding box
            **else**
                Crop face without rotation (if eye detection fails)
            **end if**
        **else**
            Crop face without rotation (if no eye detected)
        **end if**
    **else**
        Keep original face (if face detection fails)
    **end if**
    **if** histogram_equalizer is True **then**
        Apply histogram equalization
    **end if**
    **if** noise_reduction is True **then**
        Apply non-local means denoising
    **end if**
    **return** processed face image
**end function**

**Algorithm 2** Save Preprocessed Images

**Require:** Preprocessed images
**Ensure:** Preprocessed dataset in HDF5 file format

**function** RESIZECONVERTIMAGE(image_size, images_data)
    **for** each image in images_data **do**
        Expand dimensions to add channel axis
        Resize image to specified image_size
    **end for**
    Stack all resized images into a batch
    Convert tensor batch to numpy array
    **return** numpy array of resized images
**end function**
**function** SAVEDATASETTOH5PFILE(aro_label_values, val_label_values, exp_label_values, file_name, X)
    Stack arousal and valence labels
    Create an HDF5 file with file_name
    Save image data (X) and labels (arousal-valence, expression) to the HDF5 file
**end function**

# Chapter 4
# Baseline - Deep Learning for FER in AV Space

## 4.1. EfficientNet-based FER

EfficientNet is a family of convolutional neural network architectures introduced by Tan and Le [1]. The key innovation of EfficientNet is its compound scaling method that uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. The baseline EfficientNet-B0 architecture consists of mobile inverted bottleneck MBConv layers as the main building blocks, squeeze-and-excitation optimization, a depth of 18 convolutional layers, an initial resolution of 224x224 pixels. The architecture is divided into seven stages, with each stage containing several identical layers. As the network progresses, the spatial dimensions are gradually reduced while the number of channels is increased. The key innovation is the compound scaling method that scales the network's width, depth, and resolution using a compound coefficient $\phi$ [1].

EfficientNet-based models are among the most popular and are well-known for facial expression recognition on the AffectNet dataset, achieving around 72% accuracy [34]. Though EfficientNet was not specifically designed for FER and there is not experiments or studies that specifically apply EfficientNet to FER tasks in the AV space, its architecture and scaling method make it a promising candidate for this task, especially when efficiency (faster training and iteration) and scalability (handling diverse data) are important considerations. The base model needs to be modified to use as pre-trained model for predicting continuous emotions of the AffectNet dataset. Firstly, the input channel of the model needs to be adapted to accommodate grayscale images, which have a single channel, instead of the standard RGB images with three channels. Thus, a new convolutional layer is created, specifically designed to accept one input channel while keeping all other parameters identical to the original layer. The weights of this newly introduced layer are initialized by averaging the pre-trained weights across the RGB channels.

Furthermore, the classifier component of the model requires modification to align with the requirements of the regression task at hand. The original classifier is replaced with a new one tailored to handle the prediction of arousal and valence values. This new classifier incorporates a dropout layer with a probability of 0.2. A linear layer is also introduced to map the extracted features to two outputs corresponding to the arousal and valence predictions. A custom loss function is implemented, combining the Root Mean Square Error (RMSE) and the Sign Agreement Rate (SAGR) metrics. Finally, a custom training loop is developed to handle the specific loss function and metrics associated with the modified model.

## 4.2. Custom loss function

We aim to minimize RMSE (Root Mean Square Error) and maximize SAGR (Sign Agreement Ratio) for continuous affect prediction. The root mean square error (RMSE) is a commonly used metric for

evaluating the performance of regression models. It estimates the average magnitude of the errors the system makes in its predictions, emphasizing more significant errors [17]. However, as arousal and valence values are spread between [-1 and 1], RMSE only reflects half of the ground truth. SAGR measures the agreement between the predicted and actual signs of the regression outputs. It assesses how well the model predicts the target variable's direction (positive or negative), regardless of the exact magnitude [35].

$$L = RMSE + \frac{\alpha}{SAGR} \quad (3)$$

Where: L is the total loss, $\alpha$ is the SAGR weight

#### 4.2.1. RMSE (Root Mean Square Error)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (4)$$

Where: $n$ is the number of samples, $y_i$ is the true value (combining both arousal and valence), $\hat{y}_i$ is the predicted value

We combine arousal and valence into a single metric for overall model performance. A unified metric makes it easier to compare different models or approaches. In addition, having a single loss function can be more computationally efficient during model training than optimizing multiple separate loss functions. Using the square root of the sum of squared errors, formula (5) gives equal importance to arousal and valence predictions.

$$RMSE = \sqrt{\frac{1}{n}\sum((y_{a,i} - \hat{y}_{a,i})^2 + (y_{v,i} - \hat{y}_{v,i})^2)} \quad (5)$$

Where: $y_{a,i}$ and $y_{v,i}$ are the true arousal and valence values for sample $i$; $\hat{y}a, i$ and $\hat{y}v, i$ are the predicted arousal and valence values for sample $i$

#### 4.2.2. SAGR (Sign Agreement Rate)

$$SAGR = \frac{1}{n}\frac{\sum(I(sign(y_{a,i}) == sign(\hat{y}_{a,i})) + I(sign(y_{v,i}) == sign(\hat{y}_{v,i})))}{2} \quad (6)$$

Where: I() is the indicator function (1 if the condition is true, 0 otherwise), sign() is the sign function

### 4.3. Training the FER model

Before using the data to train the model, the program proceeds to data loading and preprocessing, where custom functions are used to load data from HDF5 files and create custom dataset classes. The data is then transformed and loaded into DataLoader objects for efficient batching during training. The core of the transfer learning process involves setting up and modifying the EfficientNetB3 model. The first convolutional layer is adjusted to accept grayscale input, and the classifier is replaced to perform regression for Arousal and Valence predictions. A custom loss function combining RMSE and SAGR is implemented. The training process involves iterating through epochs, performing forward passes, calculating losses, and updating model weights through backpropagation. The custom loss function evaluates the model on training and test sets. The core structure of the EfficientNet model remains largely unmodified for the FER task. The main components that are not changed include the overall architecture of EfficientNet, such as its depth and width, the MBConv blocks that form the backbone of the network and the compound scaling principle that balances network depth, width, and resolution.

**Algorithm 3** End-to-End EfficientNet-Based FER

**Require:** Paths to the datasets
**Ensure:** Trained EfficientNet-based FER model and performance metrics

**function** LOADDATASETFROMDISK(data_file_name) ▷ Data Loading and Preprocessing
    Open HDF5 file
    Load images and labels **return** data arrays
**end function**
**function** CUSTOMDATASETCLASS(Dataset)
    Initialize with images, labels, and transforms
    Implement len and getitem methods
**end function**
**function** TRANSFORMDATALOADER(image_size, batch_size, X_train, y_train, X_test, y_test)
    Create data transforms (resize, grayscale, normalize)
    Create CustomDataset instances
    Create DataLoader objects
**end function**
Load pre-trained EfficientNetB3 ▷ Model Setup
Modify the first convolutional layer for grayscale input
Replace classifier for regression (Arousal and Valence)
Move model to GPU
**function** CUSTOMLOSS(y_pred, y_true, sagr_alpha=0.1)
    Calculate RMSE
    Calculate SAGR
    Combine RMSE and SAGR **return** total loss, RMSE, SAGR
**end function**
**function** TRAINMODEL(model, optimizer, train_loader, val_loader, num_epochs, sagr_alpha, device)
    Loop through epochs
    **for** each batch **do**
        Forward pass
        Calculate loss
        Backpropagation
        Update weights
    **end for**
    Print training metrics
**end function**
**function** PREDICTING(model, data_loader, sagr_alpha) ▷ Evaluation
    Set model to evaluation mode
    Iterate through test data **return** total loss, RMSE, SAGR
**end function**
**for** each dataset configuration **do** ▷ Main Execution Loop
    Load and split dataset
    Create data loaders
    Train model
    Evaluate model
**end for**

## 4.4. Results and discussion

The results obtained from the transfer learning model can serve as a benchmark and contribution to the exploratory data analysis (EDA) process. Preprocessing is crucial in developing an efficient machine learning model, and different approaches may yield varying results. Histogram equalization (HE) and denoising (DN) are widespread facial expression recognition (FER) preprocessing techniques. However, it is essential to validate whether these methods improve the model's performance and whether both approaches are necessary for the AffectNet dataset.

| Performance Metrics | Dataset of 400 Training Images | | | | Dataset of 3840 Training Images | |
|---|---|---|---|---|---|---|
| | None | DN | HE | HE & DN | None | HE & DN |
| Training Time (s) | 36.77 | 37.76 | 36.51 | 36.00 | 428.90 | 402.30 |
| Training Loss | 0.2215 | 0.2428 | 0.2294 | 0.2343 | 0.2170 | 0.2195 |
| Training RMSE | 0.1333 | 0.1343 | 0.1212 | 0.1242 | 0.1086 | 0.1117 |
| Training SAGR | 0.9297 | 0.9156 | 0.9176 | 0.9094 | 0.9234 | 0.9290 |
| Test Loss | 0.5650 | 0.5671 | 0.5834 | 0.5966 | 0.5223 | 0.5289 |
| Test RMSE | 0.4234 | 0.4235 | 0.4396 | 0.4429 | 0.3910 | 0.3965 |
| Test SAGR | 0.7083 | 0.7031 | 0.6979 | 0.6667 | 0.7682 | 0.7583 |

**Table 4.1:** Performance metrics for model with different dataset sizes and preprocessing techniques. DN: Denoising, HE: Histogram Equalization.

As represented in Table 4.1, the original dataset without applying histogram equalization and denoising, which is referred to as "None", achieves the lowest test loss. In contrast, the dataset that applied both methods has the highest test loss, with a 5.6% increase compared to the former. However, the training run time of the latter dataset is 2% less than the former. The dataset with only denoising and without histogram equalization also performs better than the dataset with the opposite approach. Neural networks are known for their superior performance with large datasets. The number of training images is increased from 400 to 3,840 to ensure the adequacy of the results. With more data, the model produces much better results than with less data. Moreover, the test losses for datasets with and without image modifications are similar. There is a significant difference in training run times, with the dataset containing image modifications, having a much shorter run time than the dataset without modifications.

Surprisingly, the EfficientNet-based FER model generally performs better on images without histogram equalization and denoising compared to images with these preprocessing steps (0.5650 vs. 0.5966 for small dataset and 0.5223 vs. 0.5289 for larger dataset). Histogram equalization and denoising can alter an image's original pixel values and textures. EfficientNet was pre-trained on natural, unprocessed images from ImageNet. By keeping the images in their original form, features that the model has learned during pre-training are preserved, thus making them easier to recognize [1, 36]. The preprocessing steps create a domain shift issue between the pre-training data (natural images) and the preprocessed facial images. This shift might make it harder for the model to transfer its learned features effectively [37]. The model might also start learning the artificial patterns introduced by histogram equalization and denoising instead of focusing on the actual facial features relevant to arousal and valence prediction [38]. Moreover, EfficientNet's architecture, mainly because of its use of compound scaling and mobile inverted bottleneck convolutions, might be well-suited to processing natural, unaltered images.

Consequently, although the dataset without preprocessing techniques performs better with neural networks, the dataset with image modifications will be selected for further model training and analysis, because it offers comparable performance while requiring less computational time. Additionally, our approaches in later chapters rely heavily on conventional feature extraction using feature descriptors rather than end-to-end neural networks with automatic deep feature learning.

# Chapter 5
# Feature Learning

Support Vector Classification (SVC) is often used in FER tasks for emotion classification, typically after feature extraction. In our case, we need to predict arousal and valence values which are continuous values, Support Vector Regression (SVR) is an appropriate choice, as it is suited for regression tasks. Additionally, SVR is widely used for predicting continuous emotion dimensions in FER systems. [39] compared the effectiveness of visual features versus audio features in modeling continuous emotion dimensions using Continuous Conditional Random Fields, which incorporated SVR. Nicolaou et al. [40] combined multiple modalities, using SVR to process features extracted from facial expressions, shoulder movements, and audio cues to predict arousal and valence dimensions. SVR was also employed to derive arousal and valence values from facial features, including texture features like LBP (Local Binary Patterns) and geometric features in [41]. On the other hand, in general, SVMs have shown good performance in handling variations in lighting, pose, and other real-world factors that affect facial expression recognition [42]. However, we still need a robust feature extraction approach to improve the model's performance. Genetic programming is often used to evolve the best combination of image descriptors, automatically learning informative features.

A new feature learning approach was developed called IDGP (GP with image descriptors) using Genetic Programming (GP) [2] which aims to automatically selects and combines existing image descriptors (LBP, HOG, SIFT, etc.) to extract rich and discriminative global and/or local features for image classification tasks. A new program structure, feature learning process, and fitness evaluation process are developed to enable IDGP to produce flexible and effective features [2].

This chapter introduces and explains the IDGP approach for feature extraction in facial expression recognition. It aims to detail the structure and components of IDGP, including its terminal set, function set, and the process of evolving feature extractors. The chapter demonstrates how IDGP combines various image descriptors to create complex, practical feature sets for emotion recognition in continuous AV space. Additionally, it presents and analyzes experimental results of IDGP in predicting arousal and valence values from facial images, comparing its performance to the end-to-end deep learning approach from Chapter 4.

## 5.1. A new representation for IDGP individuals

IDGP introduces a novel representation structure based on Strongly Typed GP, designed to enable flexible global and/or local feature learning. The structure, as illustrated in Figure 5.1 (left), consists of several tiers: input, region detection, feature extraction, feature concatenation, and output. Each tier employs specific functions tailored to its purpose [2].

The input tier takes in an image. The region detection tier, which is optional, identifies areas of interest within a larger image. Its presence or absence in IDGP trees determines whether the solution focuses on local or global features. The feature extraction tier processes the entire input image (global features) or the detected regions (local features), utilizing functions designed for both scenarios. The feature concatenation tier combines features from child nodes into a single vector. This structure incor-

porates various functions, including Region_S and Region_R for region detection, G_SIFT, L_DIF, and L_uLBP for feature extraction, and FeaCon2 for feature concatenation as represented in Figure 5.1 (different colors indicate inputs, outputs and different functions). This flexible architecture allows IDGP to generate trees of varying depths and produce diverse combinations of global and/or local features. The resulting feature sets can be a mix of global and local features, exclusively local or exclusively global. The example program illustrated in Figure 5.1 (right) demonstrates that the resulting feature set is a mixture of various features. It combines global SIFT descriptors with local features, specifically DIF and uLBP. The total number of features in the output, denoted as S, consists of three components: s1, s2, and s3. Here, s1 represents the quantity of SIFT features, s2 is the count of DIF features, and s3 indicates the number of uLBP features. These diverse feature sets are extracted using specialized functions: G_SIFT for global SIFT features, L_DIF for local Domain-Independent Features, and L_uLBP for local uniform LBP features. This structure is distinct from previous approaches, enabling the integration of varying image descriptors into GP trees to extract diverse features [2].

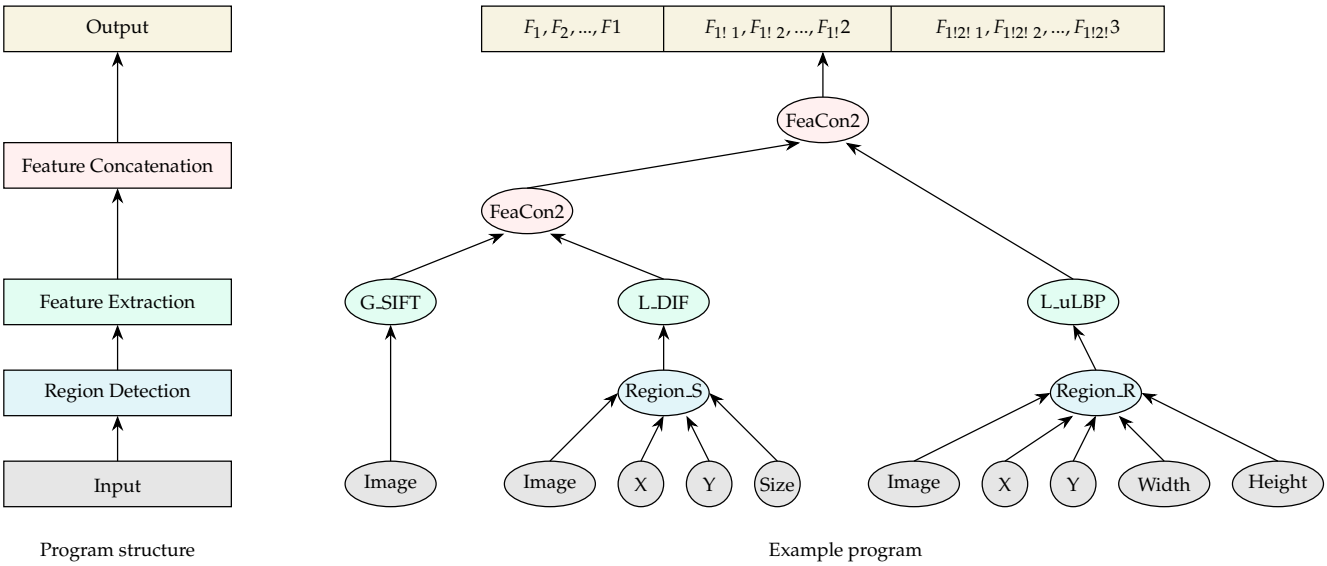



**Figure 5.1:** The new presentation of IDGP and an example program that describes a combination of global and local features from an input image [2].

### 5.1.1. Terminal set

Definiting both a function set and a terminal set is essential to construct a GP tree. The terminal set typically comprises variables, features, or attributes relevant to the problem, as well as Ephemeral Random Constants (ERCs). For instance, the terminal set might include the dataset's features in a regression task. These terminals serve as the inputs to the GP system. When presented with an instance containing multiple features, a GP tree can process these features and produce an output [2].

The terminal set of IDGP consists of 6 terminals:

- Image: the input grayscale image (normalized to [0, 1] by dividing pixel values by 255), which is a two-dimensional array (m × l).
- X: ephemeral random constant, integer in [0, m-20], representing the x-coordinate of the top-left point of a detected region.
- Y: ephemeral random constant, integer in [0, l-20], representing the y-coordinate of the top-left point of a detected region.

- Size: ephemeral random constant, integer in [20, 50], representing the size of a detected region.
- Width: ephemeral random constant, integer in [20, 50], representing the width of a detected region.
- Height: ephemeral random constant, integer in [20, 50], representing the height of a detected region.

These terminals are used as inputs to the IDGP program, with the image terminal being the primary input and the other terminals being used to parameterize the Region_S and Region_R functions [2].

### 5.1.2. Function set

The function set in Genetic Programming (GP) comprises operations used to construct the internal and root nodes of GP trees. These functions are typically problem-specific and can be categorized into two types: general-purpose and domain-specific. General-purpose functions include arithmetic operations ($+, -, \times$, and protected division /), logical operations (e.g., if, or), and mathematical functions (e.g., cos, sin, log, exp) These functions accept one or more inputs and perform their respective operations. They are commonly employed in solving straightforward problems such as feature selection, feature construction, classification, and symbolic regression. Domain-specific functions, on the other hand, are tailored to particular fields, such as image processing. These specialized functions can enhance the solution representation in GP, potentially leading to superior outcomes compared to using only general-purpose functions. By incorporating domain knowledge into the function set, GP can more effectively navigate the solution space in complex, specialized problem domains [2].

#### 5.1.2.a. Function set definition

The function set of IDGP consists of three functions: region detection, feature extraction, and feature concatenation. The region detection functions, Region_S and Region_R, are used to detect small square and rectangle regions from a large input image, respectively. The Region_S function takes the Image, X, Y, and Size terminals as inputs and returns a square region, while the Region_R function takes the Image, X, Y, Width, and Height terminals as inputs and returns a rectangle region. These functions ensure that the detected region is within the bounds of the input image [2]. The feature extraction functions are based on five representative image descriptors: DIF, histogram (Hist), SIFT, HOG, and uniform LBP (uLBP). These descriptors extract features from the whole image or automatically detect regions. The methods in the global scenario, such as G_DIF, G_Hist, G_SIFT, G_HOG, and G_uLBP, extract features from the whole image, while the methods in the local scenario, such as L_DIF, L_Hist, L_SIFT, L_HOG, and L_uLBP, extract local features from a detected region. Each feature extraction function transforms an image or a region into a variable number of features [2].

The feature concatenation functions, FeaCon2 and FeaCon3, are used to concatenate two or three feature vectors into a single feature vector (FeaCon), respectively. These functions have input types of Vector and an output type of Vector, allowing IDGP to evolve trees with variable lengths to produce different numbers of features. The output type of IDGP is Vector, indicating that the feature extraction functions and the feature concatenation functions can be the root node of IDGP, which enables IDGP to produce a variable number of features as output [2].

#### 5.1.2.b. Feature extraction functions

The fetureDIF function takes an image as input and divides it into four regions: top-left, top-right, bottom-left, and bottom-right, and calculates five features per region. For each region, the mean and standard deviation are measured. These values are then concatenated to form a feature vector, which is returned by the function. DIF function extracts 20 features for both global and local regions. The histLBP function calculates an image's Local Binary Pattern (LBP) using the provided radius and number of

points. It then computes the histogram of the LBP values and returns the histogram as a feature vector. This function extracts 59 features for both global and local regions.

The Gabor_process function applies Gabor filters to the input image at different angles and scales. It then calculates the mean and standard deviation of the filtered images. These values are concatenated to form a feature vector, which is returned by the function. The number of features extracted by this function varies for local regions as they are selected randomly during the GP process. For the global region, since the input image size is fixed at $128 \times 128$, the output of the Gabor filter is a 2D array of size (128, 128), which is then flattened into a 1D array of length 16384. Search space becomes huge with the Gabor filter, so this image descriptor is excluded from the feature extraction functions.

The hog_features function divides the input image into patches of size, patch_size x patch_size. For each patch, it calculates the Histogram of Oriented Gradients (HOG). The HOG features from all patches are concatenated to form a feature vector, which is returned by the function. This function extracts 144 features for the global region by dividing the image into 16 patches and calculating nine features per patch. For local regions, the number of features varies depending on the size of the local region. The all_sift function extracts 128 features for both global and local regions, calculates the Scale-Invariant Feature Transform (SIFT) of the input image, and returns the SIFT feature vector.

An example of the best individual (Figure 5.2) is

```
FeaCon(
  FeaCon3(
    Global_HOG(Image0),
    Local_uLBP(Region_S(Image0, 30, 91, 48)),
    Local_uLBP(Region_R(Image0, 50, 71, 47, 42))
  ),
  FeaCon2(
    Global_SIFT(Image0),
    Local_uLBP(Region_R(Image0, 96, 1, 41, 35))
  )
)
```

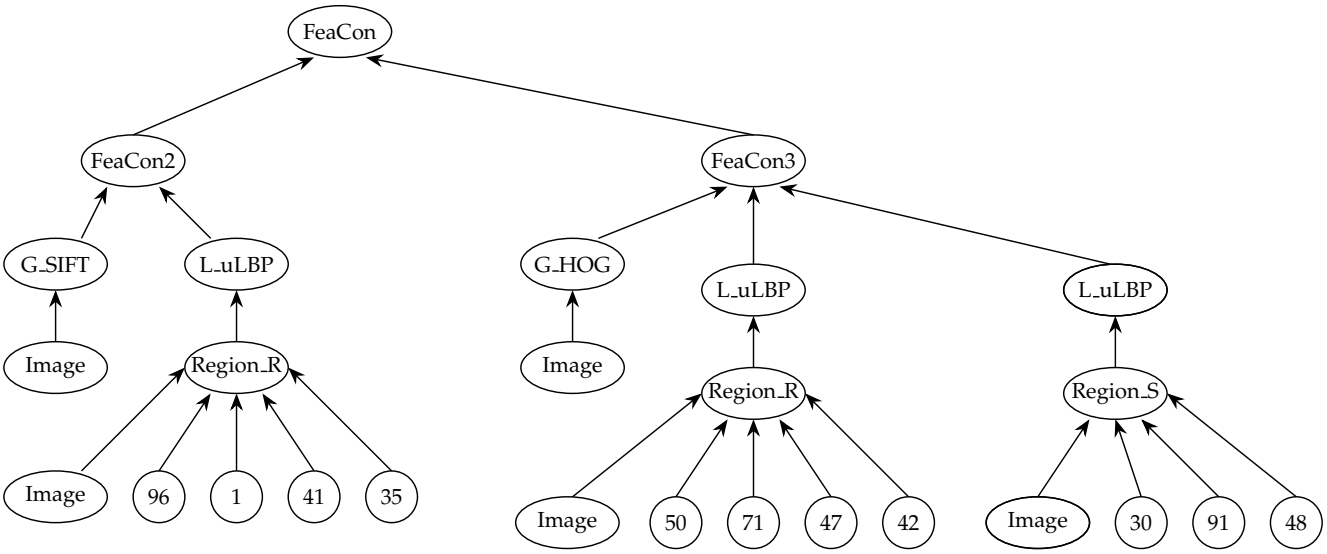



**Figure 5.2:** Example program evolved by IDGP.

### 5.1.3. GP-based feature learning for FER

GP with Image Descriptors (IDGP) is a novel approach specifically designed for feature extraction in different image machine learning tasks. The primitive set of this IDGP includes various image processing functions for global and local feature extraction (e.g., HOG, LBP, SIFT), which are tailored for facial feature analysis. The fitness function combines two metrics, RMSE and SAGR, with SAGR being used to ensure the predicted values have the correct sign, making it a novel approach specific to the arousal-valence prediction problem. The GP system evolves feature extractors that are evaluated using cross-validation with a Support Vector Regression (SVR) model, creating a hybrid GP-SVR approach. The main difference of IDGP is its innovative program architecture that enables the incorporation of various image descriptors into GP trees, allowing for the extraction of global and local features. In contrast, alternative program structures like multi-layer GP (MLGP) [2] and GP-based Evolutionary Deep Learning [2] are limited to learning only a single type of image feature.

The modular presentation of the method can be divided into three main stages: initialization, evolution, and evaluation. Figure 5.3 provides a visual representation of the IDGP process, illustrating the key steps from data loading to final evaluation.

The first step in the initialization phase is data loading and preprocessing. The training and test datasets are loaded from HDF5 files, and the image data is normalized by dividing by 255. Next, the GP system is set up by defining population size, number of generations, and crossover/mutation probabilities. A strongly-typed primitive set is created, which includes image processing functions, and custom data types (Int1, Int2, Int3, Img, Region, Vector, Vector1) are defined. The primitive set is then populated with global feature extraction primitives (DIF, Histogram, HOG, LBP, SIFT), local feature extraction primitives, region detection operators, and terminals (image input and random constants for region selection).

The evolution phase begins with the initialization of the population using the ”Half and Half” method with depth limits. The fitness evaluation is performed using a fitness function that combines RMSE and SAGR metrics. Cross-validation with SVR is implemented for feature evaluation, and MinMaxScaler is used for feature normalization. Parent selection is carried out using tournament selection and elitism to preserve the best individuals. Genetic operators, including one-point crossover and uniform mutation with depth limits, are applied, and static limits are used to maintain the maximum tree depth. The evolution loop evaluates the fitness of individuals, selects parents, applies genetic operators, creates a new generation, updates statistics and the hall of fame, and implements checkpointing for resuming evolution if needed.

In the final evaluation phase, the best individual (GP tree) is used to extract features from both the training and test sets. These features are normalized using MinMaxScaler, and an SVR model is trained on the full training set. The trained model is then used to predict the test set, and the final test loss, RMSE, and SAGR are calculated.

**Algorithm 4** IDGP (GP with Image Descriptors)

**Require:** Paths to the dataset
**Ensure:** Trained IDGP program and performance metrics
**Data Loading and Preprocessing**
    Load training and testing dataset
    Normalize image data (divide by 255)
**GP Initialization**
    Set up GP parameters (population size, generations, crossover/mutation probabilities, etc.)
    Define primitive set with feature extraction functions (DIF, Histogram, HOG, LBP, SIFT)
    Define fitness function
**GP Evolution**
    Initialize population
**for** each generation **do**
    Evaluate the fitness of individuals
    Extract features using GP individual
    Normalize features
    Perform cross-validation with SVR
    Calculate fitness (RMSE + alpha/SAGR)
    Select individuals for next generation
    Apply genetic operators (crossover, mutation)
    Check termination criterion
**end for**
**Best Individual Evaluation**
    Extract features from training and test sets using the best GP individual
    Normalize features
    Train SVR on the training set
    Predict on the test set
    Calculate the final test loss, test RMSE and test SAGR
    Print/save number of features, best individual, test loss, RMSE, SAGR, and execution times

## 5.2. Experimental evaluation

### 5.2.1. Experimental settings

The Image descriptors-based Genetic Programming (IDGP) algorithm is configured with a population size of 30 individuals evolving over 15 generations. Genetic operators are set with probabilities of 0.8 for crossover, 0.19 for mutation, and 0.01 for elitism. Tree depth is controlled with initial depths ranging from 2 to 6 and a maximum of 8. The fitness function combines Root Mean Squared Error (RMSE) and Sign Agreement Ratio (SAGR), with a SAGR alpha of 0.1, evaluated using 3-fold cross-validation with a Support Vector Regression (SVR) model. Selection is performed through tournaments of size 5, while crossover and mutation use one-point and uniform strategies, respectively. The algorithm employs a variety of global and local feature extraction primitives, including DIF, Histogram, HOG, LBP, and SIFT, along with region detection operators. Data preprocessing involves normalizing image pixel values to [0, 1] and applying Min-Max scaling to extracted features. The final model uses SVR, with performance evaluated using RMSE and SAGR on the test set. Table 5.4 presents an overview of the experimental settings for IDGP algorithm, detailing both genetic programming parameters and image processing and evaluation metrics.

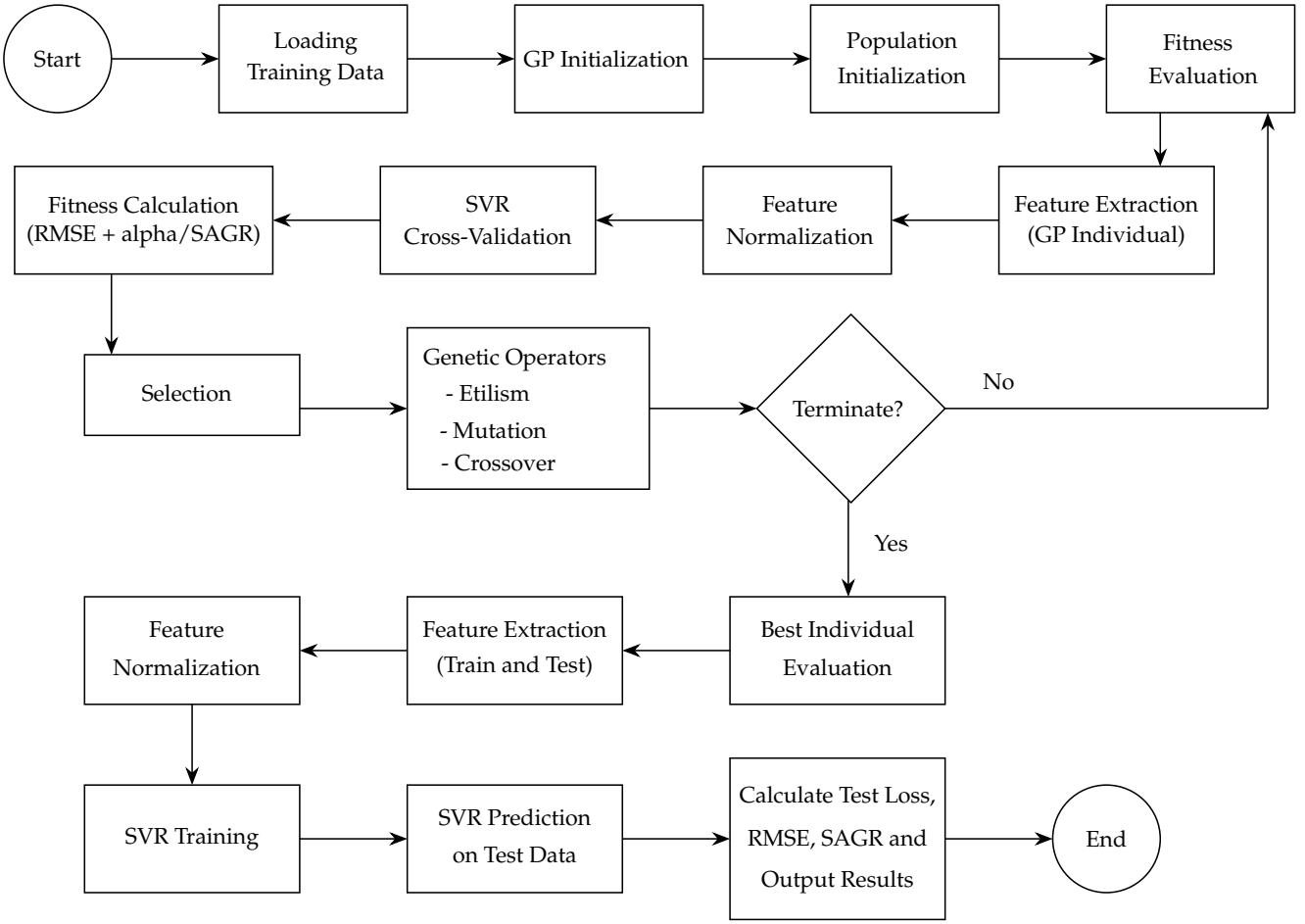



**Figure 5.3:** Feature learning process of IDGP.

| Genetic Programming | | Image Processing and Evaluation | |
|---|---|---|---|
| Parameter | Value | Parameter | Value |
| Population Size | 30 | Image Pixel Normalization | [0, 1] range |
| Number of Generations | 15 | Feature Scaling | Min-Max scaling |
| Crossover Probability | 0.8 | Fitness Function Component | RMSE and SAGR |
| Mutation Probability | 0.19 | SAGR Alpha | 0.1 |
| Elitism Probability | 0.01 | Cross-validation | 3-fold |
| Initial Depth Range | 2 to 6 | Final Model | SVR |
| Maximum Depth | 8 | Performance Metrics | RMSE and SAGR on test set |
| Selection Method | Tournament | | |
| Tournament Size | 5 | | |
| Crossover Strategy | One-point | | |
| Mutation Strategy | Uniform | | |

**Table 5.4:** IDGP experimental settings.

Most parameters adhere to the original settings established by the IDGP's authors in [2]. However, the population size and number of generations have been reduced to more than half of their original values due to computational constraints and time limitations. These adjustments aim to strike a balance between computational efficiency and model complexity.

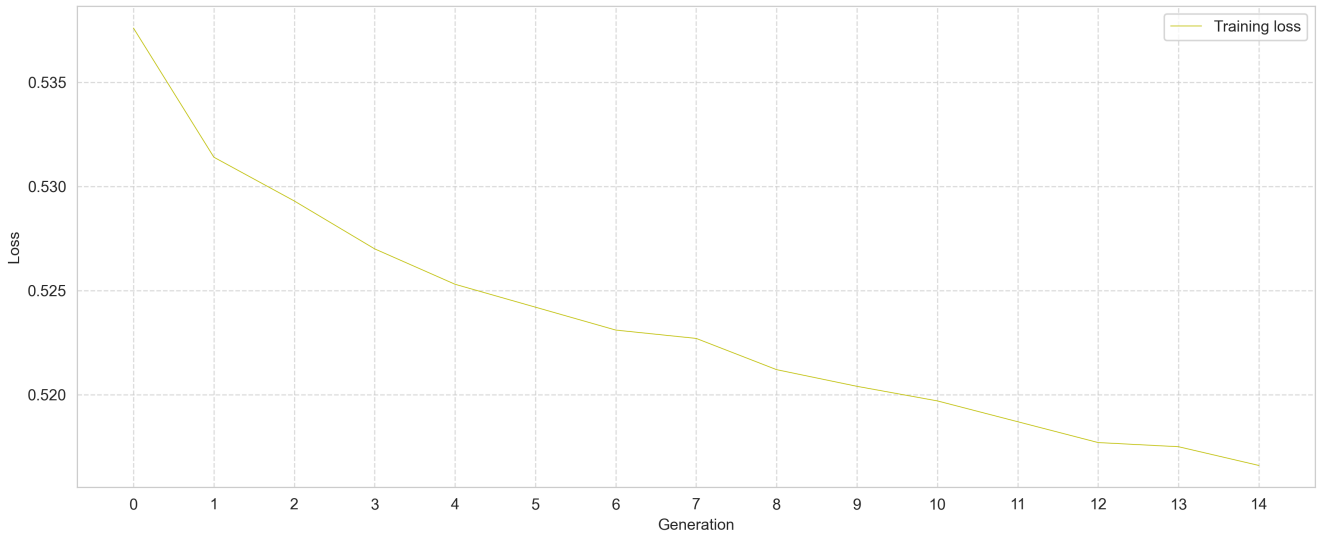



**Figure 5.5:** IDGP average training loss across ten runs.

### 5.2.2. Results and discussion

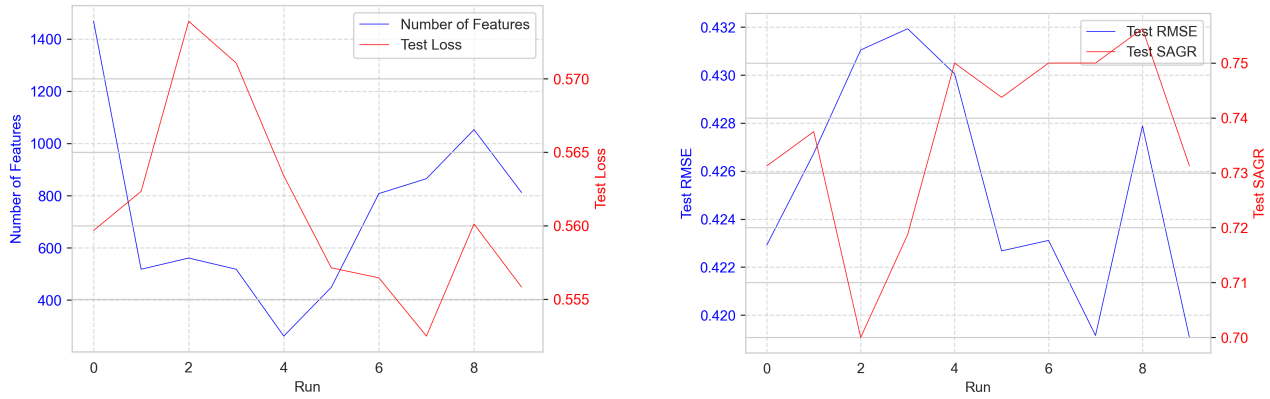
The experiment of feature learning for FER aims to find the best combination of image feature descriptors using genetic programming to optimize the fitness function (3). The experiment was conducted over ten runs, each comprising 15 generations with 30 evaluations per generation.

Figure 5.5 illustrates the evolution of the average training loss of 15 generations with the IDGP model. This analysis provides insights into the learning process in GP-based feature learning. The training loss is calculated in the "evalTrain" function, which compiles the individual (GP tree), applies it to the training data, normalizes the features, and uses cross-validation with SVR to compute the mean RMSE and SAGR, combining them into a fitness score. During the evolutionary process, these evaluation functions are called within the eaSimple function, which records the best individual with the best fitness for each generation in the log object. The program leverages DEAP [43]'s statistics and logging capabilities to track and store these fitness values across generations. Figure 5.5 illustrates a clear downward trend in training loss across generations, indicating successful optimization of the feature learning process. The loss starts at 0.5376 in the first generation and decreases to 0.5166 by the 14th generation. This improvement is not uniform, showing rapid initial progress followed by more gradual refinement. The rate of improvement slows in later generations, particularly after generation 12, suggesting the algorithm is approaching convergence.

Table 5.6 shows the average number of features across all runs was 731, with a standard deviation of 331, suggesting significant variability in the complexity of the evolved solutions. The best-performing run achieved a test loss of 0.5525, corresponding to an RMSE of 0.4191 and a SAGR of 0.7562 (Table 5.6).

| Metrics | IDGP | | | | End-to-End |
|---|---|---|---|---|---|
| | Average | SD | Max | Min | Average |
| Training Time | 13254.32 | 4452.70 | 24078.17 | 6558.07 | 402.30 |
| Feature # | 731 | 331 | 1468 | 262 | 1536 |
| Test Loss | 0.5612 | 0.0064 | 0.5739 | 0.5525 | 0.5966 |
| Test RMSE | 0.4255 | 0.0045 | 0.4319 | 0.4191 | 0.4429 |
| Test SAGR | 0.7369 | 0.0164 | 0.7562 | 0.7000 | 0.6667 |

**Table 5.6:** Performance metrics over ten runs of IDGP vs. the baseline model. SD: Standard Deviation.




**(a)** Number of features vs. Test loss.

**(b)** Test RMSE vs. Test SAGR.

**Figure 5.7:** Summary of the test results over ten runs of IDGP.

The evolution process showed interesting dynamics across different runs. The second run produced a solution with 1468 features, demonstrating that sometimes more complex feature sets can yield better performance (test loss of 0.5599). In contrast, the fifth run converged to a much simpler solution with only 262 features yet still achieved competitive performance (0.5633, as seen on Figure 5.7a), highlighting the algorithm's ability to find efficient representations. The variability in training times, ranging from about 6,558 seconds to 24,078 seconds, reflects the different complexities of the evolved solutions and the stochastic nature of the genetic programming process. The standard deviation for test RMSE (0.0045) is relatively small compared to its average, indicating consistency across runs. The test SAGR shows slightly more variability with a standard deviation of 0.0164. Despite variations in the number of features (ranging from 262 to 1468), the RMSE and SAGR values remain relatively stable, indicating robustness in the feature selection process.

Between the GP-based feature learning (IDGP) approach and the end-to-end EfficientNet-based FER using the same dataset size of 400 images with preprocessing (histogram equalization and denoising), the GP-based method performs better (average test loss of 0.5612 vs. 0.5966), indicating GP's effectiveness with smaller datasets. Additionally, the GP-based approach offers greater adaptability by exploring a wide range of feature combinations, making it advantageous for applications requiring flexibility and customization. Its focus on feature selection also enhances interpretability, highlighting specific facial regions and features relevant to expression analysis [2]. However, the deep learning model benefits from faster training times, making it more suitable for scenarios with limited computational resources or time constraints.

Figure 5.7 presents a summary of the results obtained from ten runs of the IDGP program, showcasing the relationships between the number of features, test loss, test RMSE, and test SAGR across different runs. The number of features, test loss, RMSE, and SAGR are outputs from each model run, providing a detailed view of the model's behavior. These metrics are calculated using a held-out test dataset to evaluate the model's generalization performance. By comparing the results from multiple runs, we can evaluate the consistency and reliability of the IDGP algorithm. The comparison allows for examining how the number of features relates to model performance, helping to identify efficient feature combinations. Moreover, by analyzing the relationships between different metrics such as test loss, RMSE, and SAGR, we gain valuable insights into how these measures interact and influence overall model performance. This analysis is crucial for fine-tuning the algorithm and interpreting its outputs in a meaningful way.

Figure 5.7a compares the number of features and test loss, revealing an interesting pattern. While there is some correlation between the number of features and performance, it is not a simple linear relationship, suggesting that the genetic programming algorithm can find efficient feature combinations that balance complexity and performance. There are instances where relatively fewer features (e.g., runs

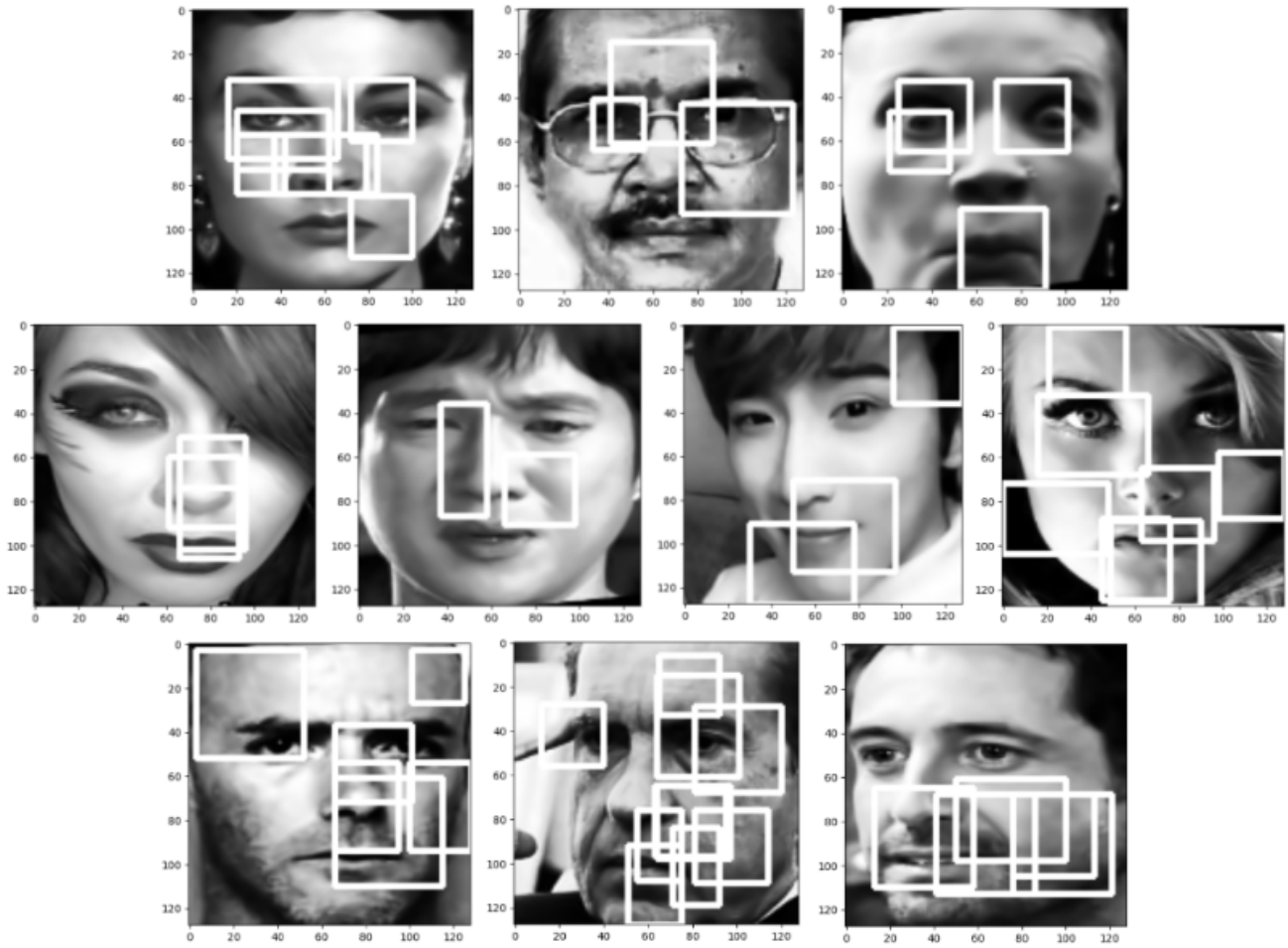


**Figure 5.8:** Local regions selected by IDGP over ten runs (order from left to right).

4 and 5) achieve competitive or even better performance than runs with many more features, highlighting the algorithm's ability to find efficient feature combinations. Despite the variations in the number of features, the test loss remains within a relatively narrow range, suggesting some consistency in the algorithm's performance across runs. There appears to be a general inverse relationship between RMSE and SAGR. As RMSE decreases, SAGR tends to increase, and vice versa (as shown in Figure 5.7b). It indicates that the fitness function works as intended, driving the optimization process to find solutions that balance both metrics. However, the varying patterns suggest that finding the optimal balance between low RMSE and high SAGR is not straightforward and may require careful tuning of the model or feature selection process.

Figure 5.8 showcases the local regions selected by the IDGP algorithm over ten runs on various facial images. These regions are extracted according to the specifications of the Region_S and Region_R functions, which are integral components of the best feature learning individual from each run (an example of a best individual is shown in Figure 5.2). The selected local regions serve as useful references, demonstrating consistent patterns in feature extraction for facial image analysis. The algorithm consistently focuses on key facial features, particularly the eyes, nose, and mouth areas, which are crucial for facial recognition and expression analysis. There is a noticeable variation in the size and number of selected regions across different runs, indicating the algorithm's adaptability to different facial structures and image qualities. Some runs prefer smaller, more numerous regions concentrated around the eyes and mouth, suggesting a focus on fine-grained details. Others display more significant regions encompassing broader face areas, capturing more holistic facial characteristics. On the other hand, the algorithm occasionally selects regions that extend beyond the face itself, including parts of the hair, forehead area

or background, which might contribute to contextual information such as different facial angles or help distinguish facial boundaries. The diversity in region selection and feature descriptor selection across runs highlights the genetic algorithm's exploration of different feature function combinations [2].

## 5.3. Chapter conclusion

The Image Descriptor-based Genetic Programming (IDGP) approach demonstrated significant potential in feature learning for FER in the AV space. Through the evolution of complex feature extractors combining various image descriptors, IDGP showed its ability to select and combine relevant facial features for emotion prediction automatically. IDGP consistently outperformed the end-to-end deep learning approach, achieving lower test loss. It also demonstrates adaptability and interpretability by focusing on crucial facial regions, particularly the eyes, nose, and mouth areas, which are crucial for expression analysis.

While IDGP showed promising results, it also highlighted the potential for further exploration in feature learning and regression techniques. Using SVR leaves room for investigating alternative regression methods that could offer more interpretable results or capture more complex relationships in the data. The next chapter will investigate how symbolic regression can enhance our understanding of the relationship between facial features and emotional states, offering new insights into facial expression recognition.

# Chapter 6
# Feature Learning with Symbolic Regression

In addition to using Support Vector Regression (SVR) to predict continuous values, we can employ genetic programming for symbolic regression to accomplish the same task. Symbolic Regression (SR) via Genetic Programming (GPSR) offers several advantages over SVR, primarily centered around explainability and flexibility. Unlike SVR's "black box" approach, GPSR produces interpretable tree-based models representing mathematical expressions, allowing for understanding relationships between inputs and outputs [31]. The primary advantage of SR lies in its flexibility and lack of assumptions. Unlike traditional regression methods, SR does not require a predetermined model structure or size, nor does it make assumptions about the underlying data distribution. The term "Symbolic Regression" highlights the method's ability to discover and express relationships in symbolic form, leading to new domain insights. This characteristic sharply distinguishes SR from conventional regression techniques, which typically focus on determining coefficients for a pre-established model structure [44]. In contrast to conventional regression techniques, symbolic regression has the advantage of minimizing potential bias. This reduction in bias is achieved by eliminating the need to analyze data distribution or predefine the model structure, which are common steps in traditional regression methods [45]. However, these benefits come with higher computational costs, while SVR offers faster training and good predictive performance.

This chapter's main objectives are to develop the Feature Descriptor-based Genetic Programming Symbolic Regression (FD-GPSR) approach for feature extraction and prediction FER tasks. It aims to detail FD-GPSR's program structure and the process of evolving symbolic regression models. The chapter demonstrates how FD-GPSR combines various image descriptors with symbolic regression to create interpretable mathematical expressions for emotion recognition. Additionally, it showcases and analyzes experimental results of FD-GPSR in predicting arousal and valence values from facial images, comparing its performance to the IDGP approach.

## 6.1. Program structure

PySR [46] and its backend, SymbolicRegression.jl, represent an evolutionary optimization algorithm for tree-based expressions. This algorithm utilizes tournament selection and local leaf searches to evolve expressions. PySR is highly customizable and performance-oriented, without imposing specific priors on the expression space or defining standard operators or functional forms. To achieve efficient evaluation, PySR employs Just-In-Time (JIT) compilation of operators into fused SIMD (Single Instruction, Multiple Data) kernels and implements multi-threading at the population level. The algorithm records the full Pareto front of expressions during the search process. Furthermore, for the 2022 Genetic and Evolutionary Computation Conference competition, PySR returns the equation demonstrating the most remarkable accuracy improvement along the Pareto front [47].

The choice of regression model is essential for solving FER problems in AV space. Instead of using SVR with a MultiOutputRegressor to predict continuous arousal and valence values, we could utilize the PySRRegressor from the PySR library [46] for symbolic regression. This change in the regression

model also affects the fitness evaluation process, where the latter program finds the best equation using PySR and calculates the fitness based on that equation. With the new approach, this program introduces a new terminal, 'const,' which represents a constant value and is added to the primitive set. We could call this model FD-GPSR (Feature Descriptor-based Genetic Programming Symbolic Regression).

The FD-GPSR program consists of three phases: initialization, evolution, and evaluation. The initialization phase begins with data loading and preprocessing, where training and test data are loaded from HDF5 files, pixel values are normalized, and arousal and valence labels are separated. Next is the GP setup, which involves defining a strongly-typed primitive set with various image-processing functions and setting up genetic operators. Feature extraction preparation is then conducted, configuring GP to evolve methods for feature extraction, including global and local primitives such as HOG, LBP, and SIFT. The initialization phase concludes with the symbolic regression setup, initializing a PySRRegressor. The evolution phase starts with feature extraction, using GP to evolve feature extraction methods. Symbolic regression is then employed, using PySR to find mathematical expressions that fit the extracted features to target values. Fitness evaluation follows, where GP individuals are compiled into executable functions, features are extracted and normalized, and fitness is calculated based on RMSE and SAGR. The evolutionary process runs for several generations, applying tournament selection, crossover, and mutation to create new individuals while maintaining a Hall of Fame. Finally, the evaluation phase involves assessing the best individual on the test set and generating the final results. Figure 6.1 offers a graphical overview of the FD-GPSR methodology, showcasing the essential stages from initial data input to final assessment.

The key differences between IDGP and FD-GPSR lie in their core approaches to regression, feature processing, and model complexity. IDGP employs SVR with MultiOutputRegressor, applying it directly to extracted features, while FD-GPSR utilizes Symbolic Regression with PySRRegressor to find mathematical expressions relating features to targets. In terms of the primitive set, both use similar feature extraction primitives, but FD-GPSR adds a constant terminal ('const') to its set. Fitness evaluation differs significantly: IDGP uses MultiOutputRegressor with SVR and calculates fitness based on RMSE and SAGR, whereas FD-GPSR finds the best equation using PySR and calculates fitness for that equation. For test evaluation, IDGP uses MultiOutputRegressor to predict the test set, while FD-GPSR uses the best equation found by PySR. Finally, regarding output interpretation, IDGP provides SVR model predictions, whereas FD-GPSR generates both model predictions and interpretable mathematical equations. These distinctions make FD-GPSR more flexible in discovering complex relationships in facial expression data, while IDGP relies on the established SVR approach.

**Algorithm 5** FD-GPSR (Feature Descriptor-based Genetic Programming Symbolic Regression)

**Require:** Paths to the dataset
**Ensure:** Trained FD-GPSR program and performance metrics
**Data Loading and Preprocessing**
    Load training and testing dataset
    Normalize image data (divide by 255)
**GP Initialization**
    Set up GP parameters (population size, generations, crossover/mutation probabilities, etc.)
    Define primitive set with feature extraction functions and add a constant terminal ('const')
    Define fitness function
**GP Evolution**
    Initialize population
**for** each generation **do**
    Evaluate the fitness of individuals
    Extract features using GP individual
    Normalize features
    Find the best equation using PySRRegressor
    Calculate fitness for that equation using PySR prediction
    Select individuals for next generation
    Apply genetic operators (crossover, mutation)
    Check termination criterion
**end for**
**Best Individual Evaluation**
    Extract features from training and test sets using the best GP individual
    Normalize features
    Predict on the test set using the best equation
    Calculate the final test loss, test RMSE and test SAGR
    Print number of features, best individual/equation, test loss, RMSE, SAGR, and execution times

## 6.2. Experimental evaluation

### 6.2.1. Experimental settings

FD-GPSR is an approach for feature extraction and regression. The experiment settings include several critical parameters for the genetic programming process. The population size is set to 30 individuals, and the algorithm runs for 15 generations. The crossover probability is set to 0.8, while the mutation probability is 0.19, with a small elitism probability of 0.01 to preserve the best individuals. The initial tree depth for individuals is constrained between 2 and 6, with a maximum depth of 8. The program uses a tournament selection with a tournament size of 5. The algorithm employs various global and local image processing techniques for feature extraction, including DIF, Histogram, HOG, LBP, and SIFT. The fitness evaluation incorporates RMSE and SAGR with an alpha value of 0.1 for SAGR. Additionally, the program utilizes the PySR library for symbolic regression, with settings such as 15 iterations (generations), 30 individuals for population size, and a maximum equation size of 100. The mathematical operations for building equations are typical binary operators ($+, -, \times, /$) and unary operators ($cos, exp, sin, log$). An operator-level "constraints" parameter is implemented to assist in exploration. This restriction, defined as nested_constraints={"sin": {"sin": 0, "cos": 0}, "cos": {"sin": 0, "cos": 0}}, prevents the nesting of sine and cosine functions. Additionally, a parsimony parameter is used to control complexity by

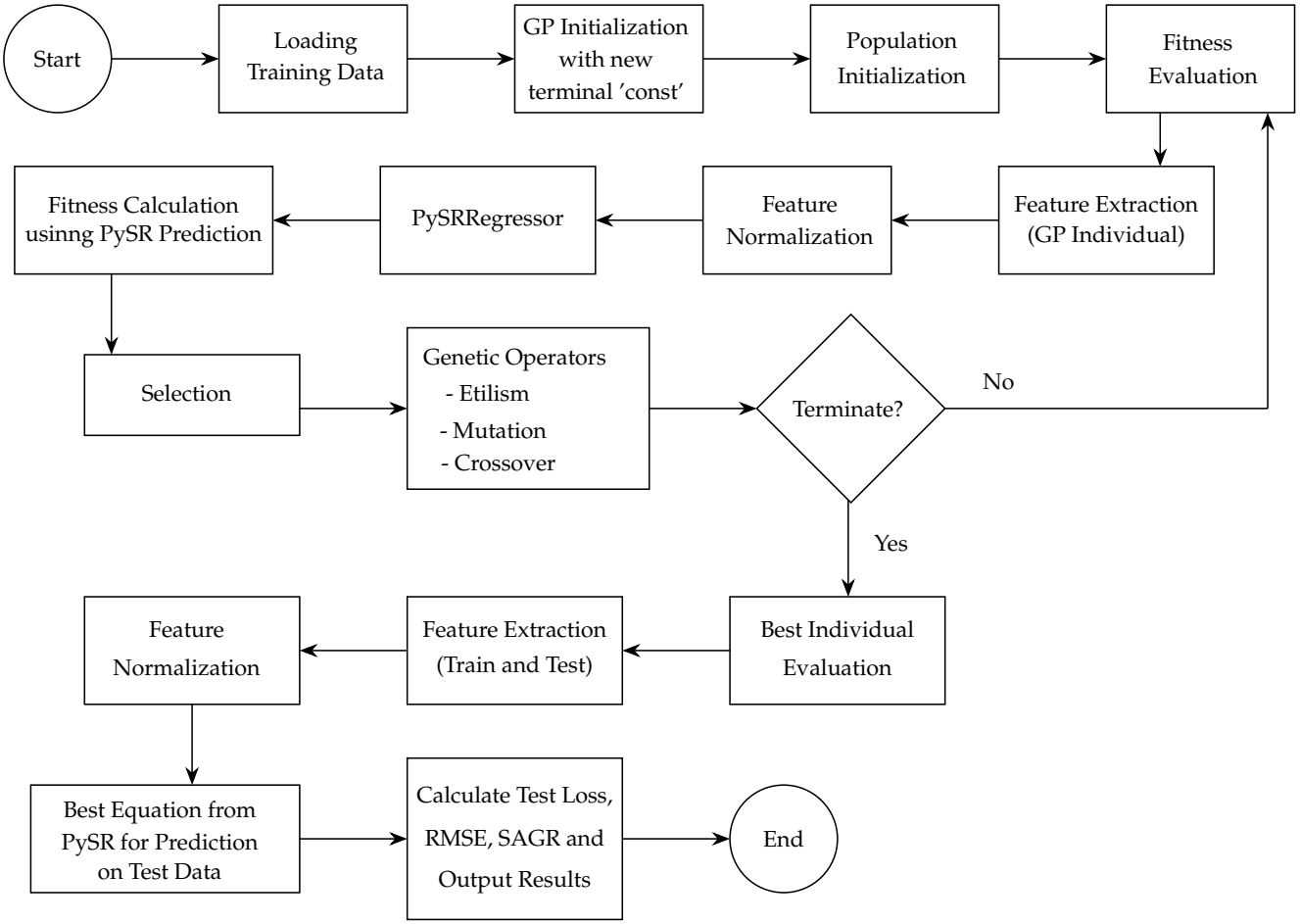



**Figure 6.1:** Feature learning process of FD-GPSR.

| Genetic Programming | | Image Processing and Symbolic Regression | |
|---|---|---|---|
| Parameter | Value | Parameter | Value |
| Population Size | 30 | Image Pixel Normalization | [0, 1] range |
| Number of Generations | 15 | Feature Scaling | Min-Max scaling |
| Crossover Probability | 0.8 | Fitness Function Component | RMSE and SAGR |
| Mutation Probability | 0.19 | SAGR Alpha | 0.1 |
| Elitism Probability | 0.01 | SR Iterations | 15 |
| Initial Depth Range | 2 to 6 | SR Population Size | 30 |
| Maximum Depth | 8 | Maximum Equation Size | 100 |
| Selection Method | Tournament | Binary Operations | +, -, ×, / |
| Tournament Size | 5 | Unary Operations | cos, exp, sin, log |
| Crossover Strategy | One-point | Nested Constraints | Limit trigonometric nesting |
| Mutation Strategy | Uniform | Parsimony Parameter | 0.20 |
| | | Regression Model | PySRRegeressor |

**Table 6.2:** FD-GPSR experimental settings.

penalizing overly complex expressions. The creator of PySR suggests setting the parsimony value to approximately one-fifth to one-tenth of the expected minimum loss. A higher value (0.20 in our case) will favor simpler equations [48]. The majority of PySRRegressor parameters follow the default settings from previous work. However, the number of iterations has been halved to reduce computational time,

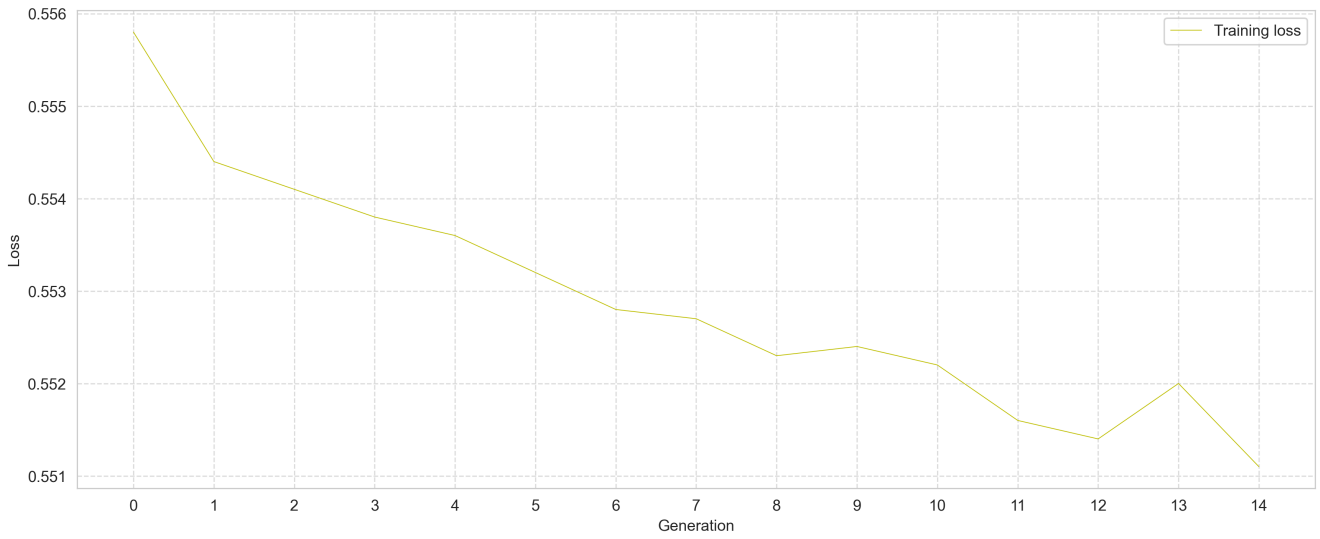



**Figure 6.3:** FD-GPSR average training loss across ten runs.

as the program must simultaneously evolve equations for arousal and valence.

### 6.2.2. Results and discussion

Similar but unlike the IDGP approach mentioned in Chapter 5, the third program, FD-GPSR, aims to find the best symbolic regression using genetic programming to optimize the fitness function (3) while finding the best combination of image features.

Figure 6.3 illustrates the evolution of the ten-run-averaged training loss over 15 generations, in the FD-GPSR model. The "evalTrain" function is used to evaluate the fitness of an individual GP tree on the training data. It compiles the GP tree, extracts features from the training images, normalizes the features, and uses PySRRegressor to perform symbolic regression. The training loss is calculated as a combination of RMSE and SAGR. Figures 6.3 shows a general downward trend in loss over generations, indicating overall improvement. The training loss begins at 0.5558 and decreases to 0.5511 by generation 14. However, the improvement is not consistent throughout. The curve exhibits some fluctuations, particularly in later generations, and the rate of improvement varies, with some plateaus and even slight increases in loss at certain points. This pattern suggests that while the symbolic regression approach is learning, it may be sensitive to genetic operations or struggling with the complexity of evolved expressions.

In comparison, the IDGP's training loss in Figure 5.5 demonstrates a smoother and more consistent downward trend. IDGP shows less fluctuation and a more stable decrease in loss over generations. It also exhibits a steeper initial decline followed by a more gradual but steady improvement, indicating a faster initial learning rate and a more robust convergence toward an optimal solution. The stability and consistency of IDGP suggest that the SVR method might be more robust to variations in data or initial conditions. However, the fluctuations in FD-GPSR, while indicating instability, could also suggest a wider exploration of the solution space, leading to more diverse feature representations that traditional methods might not capture.

As depicted in Table 6.4, the average training time across all runs was substantial at 10,771.39 seconds (approximately 3 hours), indicating the computational intensity of the symbolic regression process. However, like the IDGP program, there was considerable variability in training times, with a standard deviation of 4,818.78 seconds, suggesting that some runs were significantly faster or slower than others. The number of features used in the best models also varied widely, with an average of 223 features and a standard deviation of 81, which is much lower than that of IDGP. Performance metrics show that the model achieved reasonably consistent results across runs. The average test loss was 0.5670, with a relatively small standard deviation of 0.0135. The test RMSE averaged 0.4317, with a standard deviation of

0.0123, further supporting the consistency of the model's performance.

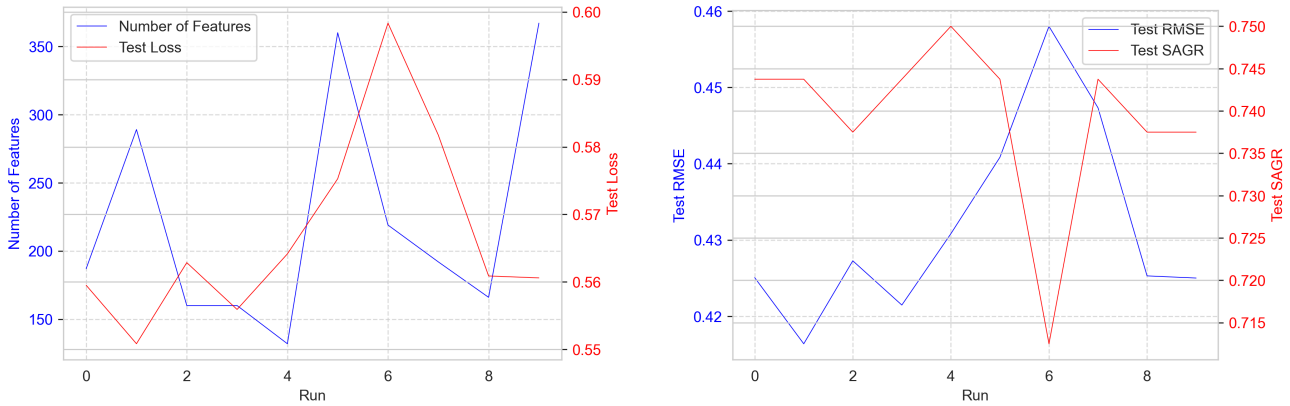
| Metrics | IDGP | | | | FD-GPSR | | | |
|---|---|---|---|---|---|---|---|---|
| | Average | SD | Max | Min | Average | SD | Max | Min |
| Training Time | 13254.32 | 4452.70 | 24078.17 | 6558.07 | 10771.39 | 4818.78 | 17111.59 | 1901.13 |
| Feature # | 731 | 331 | 1468 | 262 | 223 | 81 | 367 | 132 |
| Test Loss | 0.5612 | 0.0064 | 0.5739 | 0.5525 | 0.5670 | 0.0135 | 0.5983 | 0.5508 |
| Test RMSE | 0.4255 | 0.0045 | 0.4319 | 0.4191 | 0.4317 | 0.0123 | 0.4580 | 0.4164 |
| Test SAGR | 0.7369 | 0.0164 | 0.7562 | 0.7000 | 0.7394 | 0.0097 | 0.7500 | 0.7125 |

**Table 6.4:** Performance metrics over ten runs of IDGP and FD-GPSR. SD: Standard Deviation.

Though IDGP has a better average test loss, FD-GPSR could achieve a better minimum test loss of 0.5508 (compared to 0.5525 of IDGP), while also having fewer features. FD-GPSR captures the underlying pattern of the data more effectively than IDGP. However, the latter program is more stable than the prior one as its test loss standard deviation is much lower (0.0064 vs. 0.0135). Figure 6.5a offers visual insights into the model's behavior across runs. The left graph shows the number of features and test loss for each run, revealing that there is not a clear correlation between the number of features and the test loss, proving that the model's performance is not simply a function of model complexity. This finding is also reflected in IDGP. The right graph Figure 6.5b displays the test RMSE and SAGR across runs, showing some fluctuation in these metrics but generally maintaining consistent performance levels.

The local regions selected by FD-GPSR (Figure 6.6), provide insights into the facial features that the model considers most relevant for its predictions. In several images, the algorithm has chosen to focus on the eye area, mainly the corners of the eyes and the space between the eyebrows. The mouth area is another frequently selected region, with boxes often encompassing the lips and surrounding areas. The selection of these regions aligns with the local region selection of the IDGP program and human intuition about important facial cues, suggesting that the FD-GPSR algorithm has learned to focus on areas that are typically considered significant in facial analysis and emotion recognition tasks. However, it is worth noting that the algorithm also occasionally selects less conventional areas, such as parts of the forehead or cheek, which might indicate that it is picking up on subtle cues that human observers might overlook.

As shown in Table 6.7, the equations are relatively simple, often involving only a few variables and basic mathematical operations. This simplicity suggests that the program favors interpretable solutions over highly complex ones, which aligns to produce understandable models. A typical structure in these




**(a)** Number of features vs. Test loss.

**(b)** Test RMSE vs. Test SAGR.

**Figure 6.5:** Summary of the test results over ten runs of FD-GPSR.

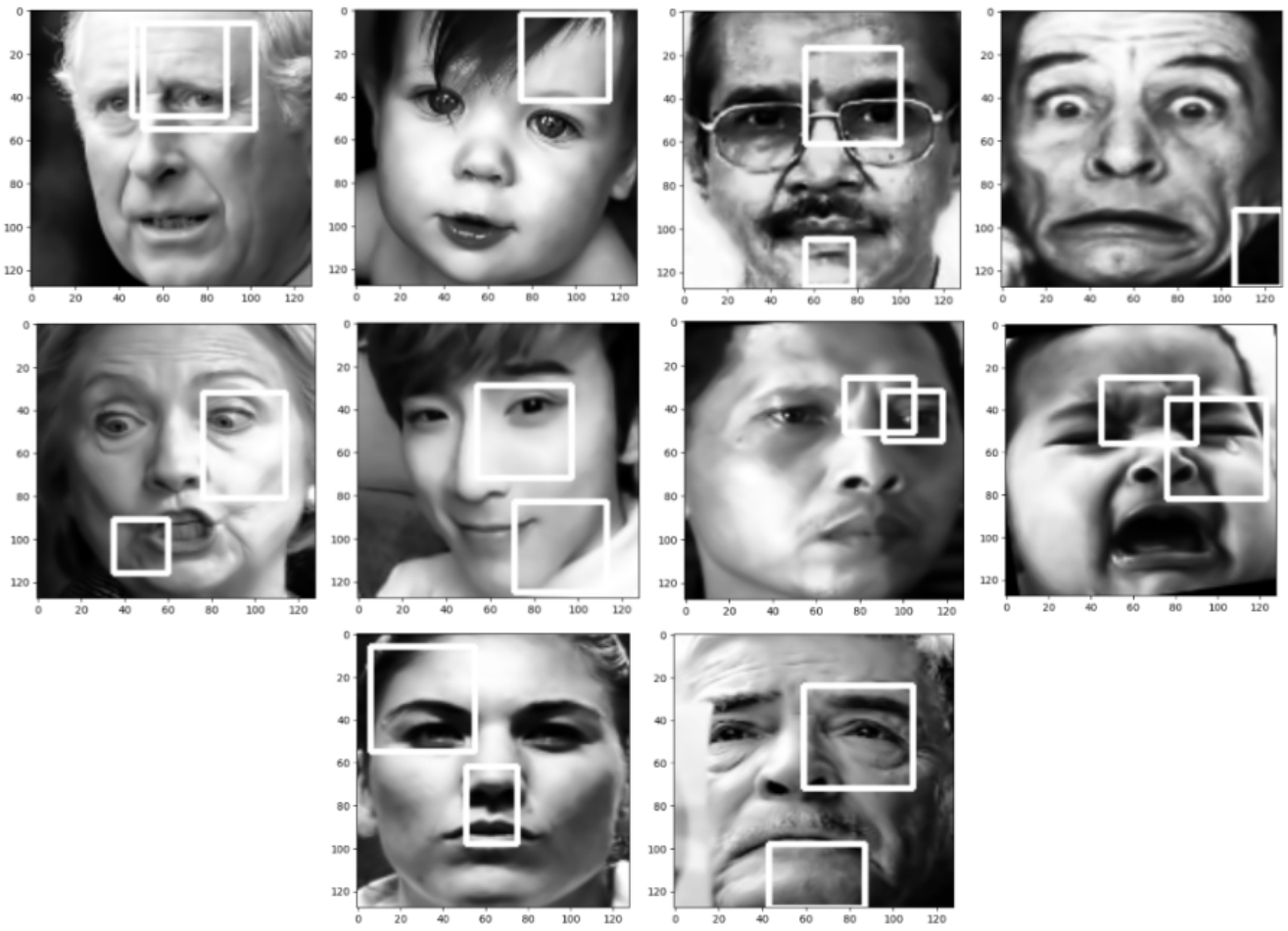


**Figure 6.6:** Local regions selected by FD-GPSR over ten runs (order from left to right).

| **Best Equation** |
|---|
| $[cos(x112 * 1.4215789 - (x104 + x67)) * cos(0.14018624) * 0.45621794, x63 * (-0.3096085)]$ |
| $[0.19177718 * x14 - 1 * (-0.3348945), sin(x93) * (-0.52311015)]$ |
| $[0.36677775 + x3 * (-0.2676289), -0.20626858 * x143]$ |
| $[sin(exp(x149 - x3)) * 0.39344272, -0.31261876 * x36]$ |
| $[x82 * sin(exp(x98) + 1.4799674) + 0.3046134, x56 - 1 * 0.31320333]$ |
| $[x17 * x30 + 0.3340563, x0 * (-0.3738894)]$ |
| $[sin(0.45955354)/exp(x194), x77 - 0.235395]$ |
| $[sin(x178 + 0.25612548), -0.27762237 * x99]$ |
| $[0.34789985 * (1.2848855 - x20), -0.72383577 * x21]$ |
| $[(x13 * x17 * x5/(-0.6575482) - x240) * sin(0.41943592/0.8802915) + 0.41943592, x152 * (-0.49207896)]$ |

**Table 6.7:** Best equations obtained by FD-GPSR.

best equations uses two main components, often separated by a comma. It represents a multi-output model, where the first component predicts arousal and the second predicts valence. The frequent appearance of trigonometric functions, particularly sine, indicates that the model captures some cyclical or periodic patterns in the data. Some equations also present exponential functions, suggesting the model detects exponential relationships or scaling effects. Many equations incorporate constants alongside the variables, which could represent scaling factors or offsets [49].

The prevalence of different feature numbers across seeds indicates that the algorithm explores various facial characteristics to find the most predictive ones. Some equations involve more complex operations like nested functions or multiple arithmetic operations, while others are simple, using only multiplication or addition. This variety suggests that the algorithm is flexible in its approach, adapting the complexity of the model to the specific patterns it detects in different runs [50]. The consistent appearance of specific operations (like multiplication by a constant followed by subtraction) across multiple seeds might indicate that these structures are particularly effective for the given problem. However, the diversity in the equations also suggests that there are multiple ways to capture the underlying patterns in the data, which is typical in complex prediction tasks like facial analysis [51].

## 6.3. Chapter conclusion

The Feature Descriptor-based Genetic Programming Symbolic Regression (FD-GPSR) approach introduced in this chapter demonstrates the potential of combining genetic programming with symbolic regression for facial emotion recognition tasks. FD-GPSR offers several advantages over traditional methods, including improved interpretability and flexibility in discovering complex relationships between input features and emotion labels. The experimental results show that FD-GPSR can achieve competitive performance compared to the IDGP approach, with a lower minimum test loss in some cases. The ability of FD-GPSR to generate interpretable mathematical expressions provides valuable insights into the underlying patterns of facial expressions and their relationship to arousal and valence values. While FD-GPSR demonstrated the potential of symbolic regression in feature learning, it also raised questions about the optimal way to combine different sets of a feature descriptor and apply symbolic regression to extracted features. The unique strength of this program is its capability to harness symbolic regression in uncovering mathematical expressions that capture the connections among input variables. The new approach has the potential to provide more profound insights into how features interact with each other and to create models for FER that are easier to interpret and understand.

# Chapter 7
# Symbolic Regression with Image Descriptors

Symbolic regression has a significant advantage in discovering the mathematical equation that describes the relationship between input variables. We can leverage this strength to explain further the interactions and relationships between different feature extraction functions. By applying symbolic regression to the features extracted by the genetic programming (GP) process, we can create a new set of transformed features that serve as the input vector for the facial emotion recognition (FER) regression task.

This chapter introduces a two-stage GP approach, that aims to create a new set of transformed features by applying symbolic regression to the features extracted. The chapter details the program structure, experimental settings, and results, emphasizing the potential of symbolic regression to improve feature interactions and model performance. The performance and comparison of different image descriptors (Histogram, DIF, SIFT, and LBP) within this framework is also featured in this chapter.

## 7.1. Program structure

Having a second stage is what makes the new program more special than other ones. In the second stage, symbolic regression is used on the feature set extracted from the first stage. Since each image descriptor yields a different number of features, which is not ideal for array mathematical operations, different feature extraction primitives are separated and run independently. We could call it CFSR (Combined Feature Symbolic Regression). This approach allows us to uncover potentially complex, non-linear relationships among the extracted features and generate a more expressive and informative feature representation while compacting the number of features by skipping the feature concatenation process.

CFSR's initialization phase consists of several steps—first, the program loads training and test datasets from HDF5 files. Images are then normalized by dividing pixel values by 255. Next, a strongly-typed primitive set is created for GP, including various feature extraction functions and region detection operators. The program sets up the GP framework, defining fitness criteria, individual creation, and population initialization. Finally, a secondary GP setup is created for symbolic regression on extracted features.

The evolution phase involves three main components. The main GP evolution process is executed using the GPMain function, which uses tournament selection, one-point crossover, and uniform mutation. The evolution process is run for a specified number of generations. A symbolic regression GP evolves within each evaluation of the main GP. This process combines extracted features to create new, more informative features. An SVR model is then trained on the transformed features produced by the symbolic regression GP. Cross-validation is used to assess the quality of the features and model.

In the evaluation phase, the best individual from the main GP and the best symbolic regression expression are selected. These best individuals are applied to the test data to extract and transform features, and the trained SVR model is used to make predictions on the test data. Performance metrics, including RMSE and SAGR, are calculated on the test predictions. Finally, the program outputs the number of fea-

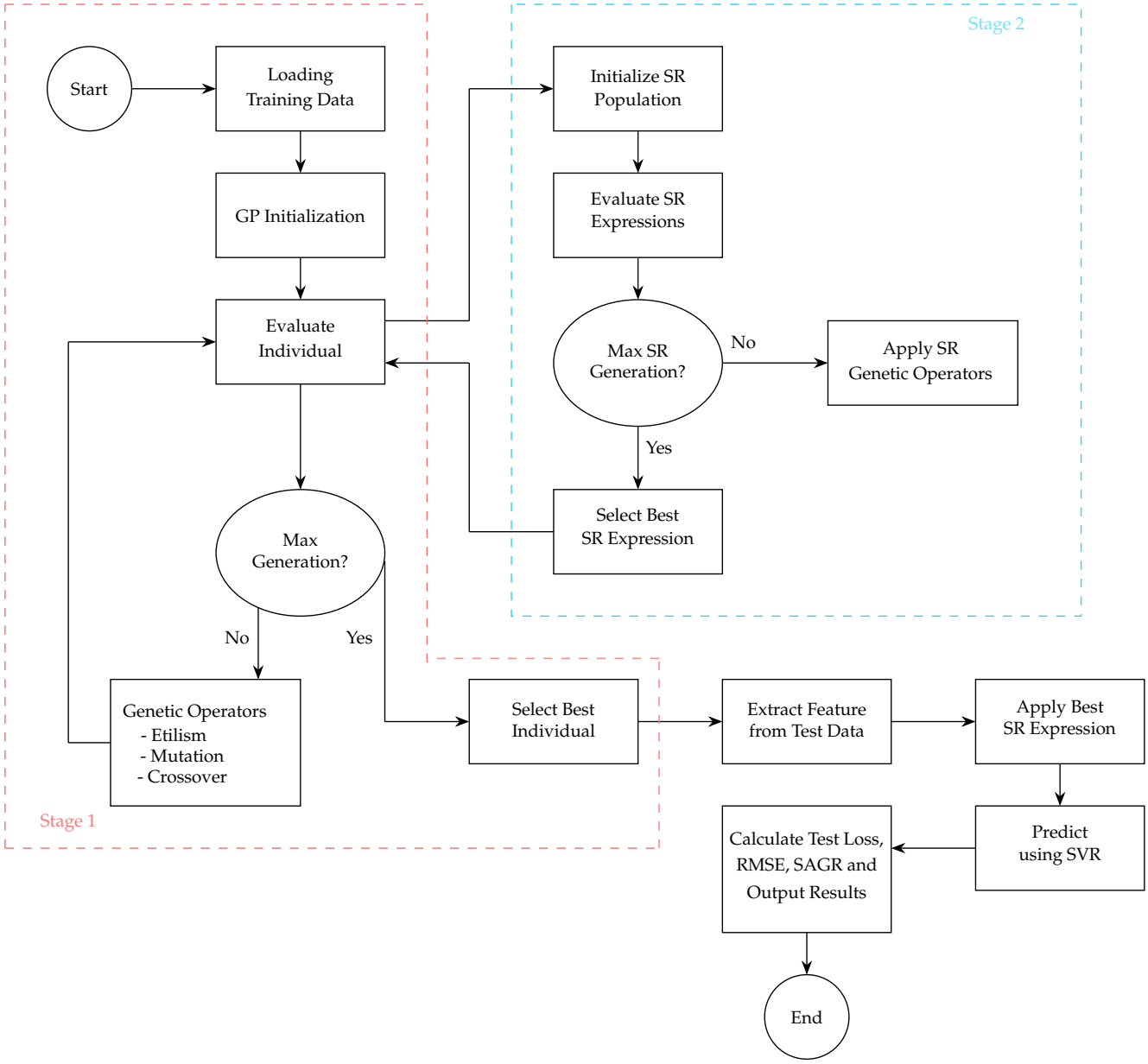



**Figure 7.1:** The process of the two-stage program CFSR.

tures, best individuals, test performance metrics, and computation times. Figure 7.1 visually represents the CFSR process, illustrating the key steps from data loading to final evaluation.

CFSR employs a two-stage genetic programming (GP) method for feature extraction instead of a single-stage GP approach like IDGP (introduced in Chapter 5) and FD-GPSR (introduced in Chapter 6). CFSR evolves feature extraction functions in the first stage and applying symbolic regression to the extracted features in the second stage. The specialty of this program lies in its ability to leverage symbolic regression to discover mathematical equations describing relationships between input variables, potentially leading to a deeper understanding of feature interactions and more interpretable models for facial emotion recognition tasks.

**Algorithm 6** CFSR (Combined Feature Symbolic Regression)

**Require:** Paths to the dataset
**Ensure:** Trained CFSR program and performance metrics
Load training and testing dataset ▷ Data Loading and GP Initialization
Initialize the GP parameters
Feature concatenation primitives ▷ Set up the primitive set for the first stage GP
Global and local feature extraction primitives (e.g., DIF, Histogram, uLBP, SIFT)
Region detection operators
Terminal primitives for constants and arguments
Compile the GP individual into a callable function
Extract features from the training images using the compiled function
Evolve symbolic regression expressions using the extracted features ▷ Set up the second stage GP
Normalize the features using MinMaxScaler
Train a MultiOutputRegressor with SVR and evaluate using cross-validation
Calculate the fitness as a combination of RMSE and SAGR
Update the best individual, features, and fitness if necessary
Register the genetic operators for the first stage GP (selection, crossover, mutation)
**Stage 1: Feature Extraction**
    Initialize a population of GP individuals
**for** each generation **do**
    Evaluate individuals by
        Extracting features from images using GP trees
        Passing extracted features to Stage 2
**end for**
    Apply genetic operators (selection, crossover, mutation)
    Update the best individual
**Stage 2: Symbolic Regression**
**for** each individual in Stage 1 **do**
    Initialize a population of symbolic regression expressions
    **for** each generation **do**
        Evaluate expressions by applying them to extracted features
        Use SVR to predict AV values
        Calculate fitness using RMSE and SAGR
    **end for**
**end for**
Update the best symbolic regression expression
Return the best fitness to Stage 1
Extract features from the test images using the best individual ▷ Evaluation on the test set
Apply the best symbolic regression expression to the extracted features
Load the trained scaler and model
Transform the test features using the loaded scaler
Make predictions using the loaded model
Calculate loss, RMSE and SAGR on the test set
Print the results, including the best individuals, test metrics, and execution times

## 7.2. Experimental evaluation

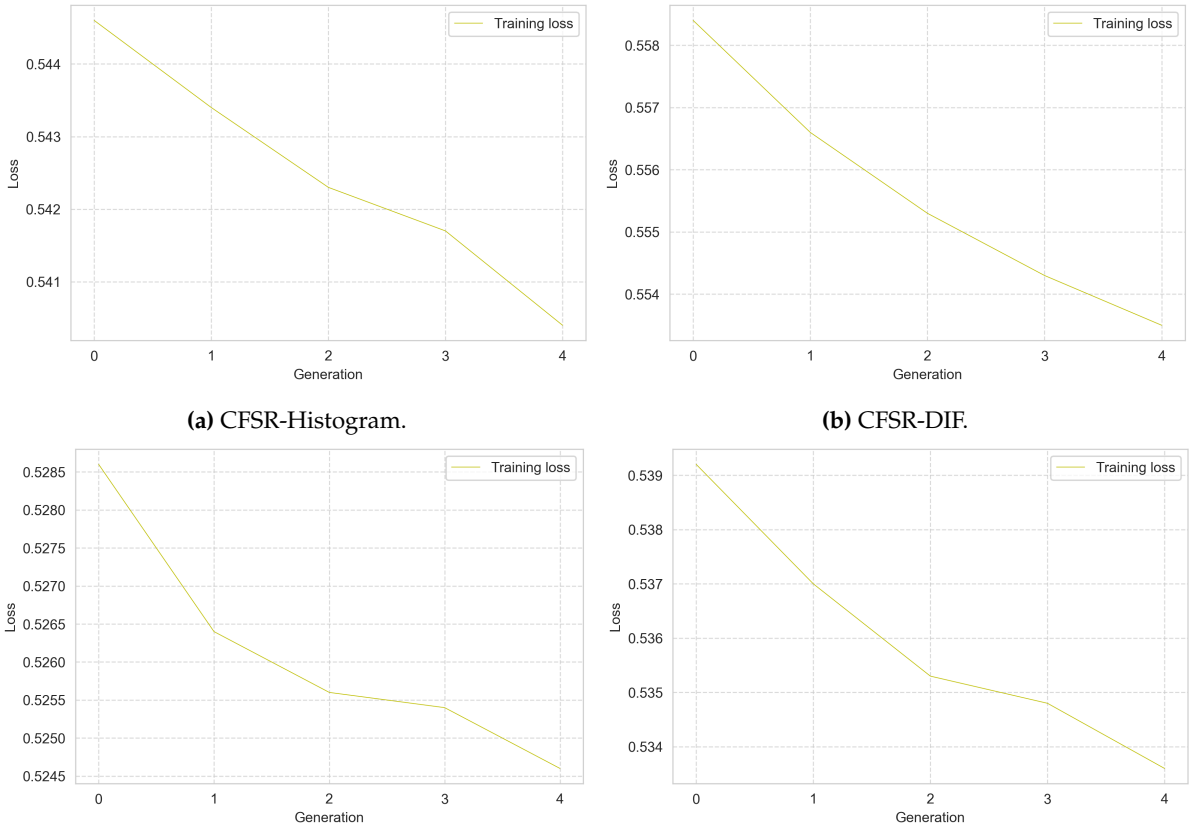
### 7.2.1. Experimental settings

CFSR implements a two-stage genetic programming (GP) approach for facial emotion recognition, with distinct parameter configurations for each stage. The first stage, focused on feature extraction, employs a population of 30 individuals evolving over five generations, with crossover, mutation, and elitism probabilities of 0.8, 0.19, and 0.01, respectively. It utilizes a strongly-typed primitive set including global and local feature extraction methods and region detection operators, with tree depths ranging from 2 to 8. The second stage applies symbolic regression to the extracted features, using a smaller population of 20 individuals over five generations, with crossover and mutation probabilities of 0.5 and 0.1. This stage incorporates vector operations such as addition, subtraction, multiplication, division, sine, and exponential functions. Both stages employ a fitness evaluation combining RMSE and SAGR, with a balancing factor alpha of 0.1. Additional settings include a random seed for reproducibility, image normalization to [0, 1], extraction of sets of features (32 features per set for Histogram, 20 for DIF, 128 for SIFT, and 59 for uLBP), 3-fold cross-validation, and the use of SVR for the final regression task. The experiment is implemented using the DEAP library, incorporating elitism, tournament selection, and one-point crossover, with the best-performing individuals from both stages used for processing the test data.

| Feature Extraction Stage | | Symbolic Regression Stage | |
|---|---|---|---|
| Parameter | Value | Parameter | Value |
| Population Size | 30 | Population size | 30 |
| Number of Generations | 5 | Number of Generations | 5 |
| Crossover Probability | 0.8 | Crossover probability | 0.5 |
| Mutation Probability | 0.19 | Mutation probability | 0.1 |
| Elitism Probability | 0.01 | Binary Operations | +, -, ×, / |
| Initial Depth Range | 2 to 6 | Unary Operations | exp, sin |
| Maximum Depth | 8 | Fitness Function Component | RMSE and SAGR (alpha = 0.1) |
| Selection Method | Tournament | Final Regression Model | 3-fold cross-validation SVR |
| Tournament Size | 5 | Implementation | DEAP library |
| Crossover Strategy | One-point | | |
| Mutation Strategy | Uniform | | |

**Table 7.2:** CFSR experimental settings.

Most parameters are kept consistent with previous methods to ensure comparability across different approaches. Nonetheless, the number of generations is lower (5 generations) compared to other GP approaches. This decision was made based on empirical observations during trial and error. The program stopped showing improvement in training loss after the 5th generation, indicating that further evolution was not beneficial. Unlike other approaches that combine various feature descriptors, CFSR uses only one type of feature descriptor at a time, which contributes to early convergence, as the search space for optimal solutions is more constrained. Moreover, the exclusion of logarithmic and cosine functions in CFSR, which were present in FD-GPSR, is due to the different nature of feature processing. Since CFSR operates on sets of features rather than individual features, using functions like cos and log could lead to numerical instabilities or extreme values when applied to entire feature sets.




**(a)** CFSR-Histogram.

**(b)** CFSR-DIF.

**(c)** CFSR-SIFT.

**(d)** CFSR-LBP.

**Figure 7.3:** CFSR average training loss across ten runs.

### 7.2.2. Results and discussions

One approach to experiment with Symbolic Regression involves applying it to sets of feature functions. Rather than concatenating all features into a single vector, this method combines them into mathematical equations, resulting in a new set of features for regression prediction. Since each image descriptor yields a different number of features, the various feature extraction primitives are separated and run independently.

Figure 7.3 present the average training loss for four different Combined Feature Symbolic Regression (CFSR) approaches: CFSR-Histogram, CFSR-DIF, CFSR-SIFT, and CFSR-LBP. For each individual in the population of the training phase, CFSR extracts features from the training images using the evolved feature extraction function. These features are then used in a second stage of GP, which evolves symbolic regression expressions to combine and transform the extracted features. The fitness of each individual is evaluated using an SVR model trained on the transformed features, with cross-validation. All four models demonstrate a consistent downward trend in training loss over five generations, indicating ongoing learning and performance improvement throughout the training phase. CFSR-Histogram starts with a relative high initial loss (0.5446) and shows a steady, almost linear decrease, ending with the second highest final loss (0.5404) among all models. CFSR-DIF begins with a highest initial loss (0.5584), exhibits a more pronounced curve in loss reduction, and achieves a highest final loss (0.5535). CFSR-SIFT stands out with the lowest initial loss (0.5286) and demonstrates the most dramatic decrease, especially in the first generation. Despite some fluctuations, it maintains the lowest loss throughout and ends with the lowest final loss (0.5246). CFSR-LBP starts with an initial loss between SIFT and Histogram (0.5392), displays a steady decrease, and finishes with the second-lowest final loss (0.5336).

The significant performance difference in training loss suggests SIFT could be one of the most effective feature extraction method for this task, possibly due to its robustness to scale, rotation, and

illumination changes in images. CFSR-SIFT shows the fastest initial improvement, quickly capturing relevant features. CFSR-Histogram, despite starting with the high loss, shows consistent improvement. CFSR-SIFT and CFSR-LBP exhibit minor fluctuations in their loss curves, potentially indicating higher sensitivity to changes in evolved expressions or processed data.

| Metrics | Histogram | | | | DIF | | | |
|---|---|---|---|---|---|---|---|---|
| | Average | SD | Max | Min | Average | SD | Max | Min |
| Test Loss | 0.5609 | 0.0053 | 0.5733 | 0.5571 | 0.5760 | 0.0186 | 0.6099 | 0.5566 |
| Test RMSE | 0.4265 | 0.0053 | 0.4388 | 0.4226 | 0.4399 | 0.0158 | 0.4645 | 0.4221 |
| Test SAGR | 0.7438 | 0.0 | 0.7438 | 0.7438 | 0.7350 | 0.0184 | 0.7438 | 0.6875 |

**(a)** Performance metrics over ten runs of CFSR with Histogram and DIF.

| Metrics | SIFT | | | | LBP | | | |
|---|---|---|---|---|---|---|---|---|
| | Average | SD | Max | Min | Average | SD | Max | Min |
| Test Loss | 0.5669 | 0.0088 | 0.5844 | 0.5572 | 0.5728 | 0.0143 | 0.6065 | 0.5572 |
| Test RMSE | 0.4325 | 0.0088 | 0.4500 | 0.4228 | 0.4384 | 0.0143 | 0.4720 | 0.4228 |
| Test SAGR | 0.7438 | 0.0 | 0.7438 | 0.7438 | 0.7438 | 0.0 | 0.7438 | 0.7438 |

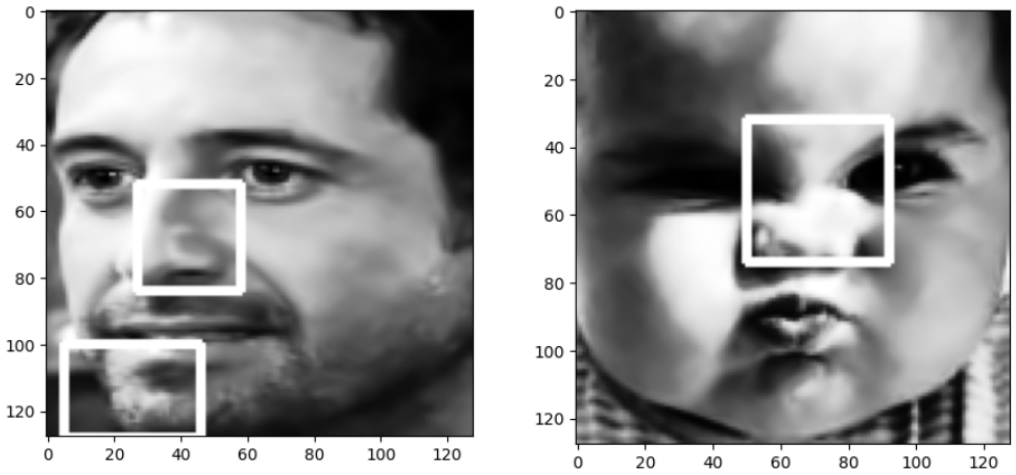
**(b)** Performance metrics over ten runs of CFSR with SIFT and LBP.

**Table 7.4:** Performance metrics over ten runs of CFSR. SD: Standard Deviation.

As illustrated in Table 7.4, the Histogram descriptor shows the best overall performance with the lowest average test loss of 0.5609, suggesting it provides the most accurate predictions among the four descriptors. Its test RMSE of 0.4265 is also the lowest, indicating smaller prediction errors on average. The Histogram's performance is notably consistent, with the slightest standard deviation in test loss and test RMSE (0.0053), demonstrating reliability across different seeds. SIFT comes in second place with an average test loss of 0.5669, slightly higher than the Histogram. However, it shows more variability in its performance, with higher standard deviations in all metrics than the Histogram, suggesting that while SIFT can perform well, its results are not consistent. LBP performs similarly to SIFT, with an average test loss of 0.5728. It shows more variability than Histogram and SIFT, as evidenced by its standard deviations. DIF's maximum test loss (0.6099) is the highest among all descriptors, indicating it may struggle more with certain types of images. DIF shows the poorest performance among the four, with the highest average test loss of 0.5760. the Test SAGR is constant at 0.7438 for all descriptors except DIF, which shows slight variations. This consistency in SAGR across most descriptors suggests that this particular metric may not be as discriminative when evaluating performance differences in this context.

The analysis of the best-performing models for each image descriptor focuses on the top individuals from stage one (feature extraction) and the optimal equations from stage two (symbolic regression) of the two-stage genetic programming approach. The CFSR-Histogram and CFSR-SIFT models yield the best test results in average out of the four descriptors.

The best equation of Histogram descriptor is $sin(sub(X_5, X_4))$ where $X_4$ represents feature function Local_Histogram(Region_R(Image0, 5, 100, 41, 32)) and $X_5$ is Local_Histogram(Region_R(Image0, 27, 52, 31, 32)) (presentation shown in Figure 7.5a and Figure 7.6). The equation represents a non-linear relationship between two input variables, where the output is the sine of the difference between $X_5$ and $X_4$. This equation suggests the model captures a non-linear relationship between two local histogram features from different face parts. The sine function shows that the model is responsive to cyclical or repeating patterns in the subtraction between these histogram features. Additionally, this model does not consider overall features, which implies that specific features offer enough information for making predictions.



**(a)** Local regions selected by Histogram descriptor. **(b)** Local regions selected by SIFT descriptor.

**Figure 7.5:** Local regions selected by the best individual from CFSR-Stage 1.

For the SIFT descriptor, the best equation is $sub(exp(X_0), add(X_1, X_1))$ as illustrated in Figure 7.5b and Figure 7.6. $X_1$ is the global region Global_SIFT(Image0), while $X_0$ is the local region Local_SIFT(Region_S(Image0, 50, 32, 42)). This equation combines both local and global SIFT features in a non-linear way. The exponential function applied to the local feature ($X_0$) suggests that small changes in this feature can greatly impact the output. The global feature ($X_1$) is added to itself before being subtracted from the exponential of $X_0$, which might capture some form of contrast or difference between local and global features.

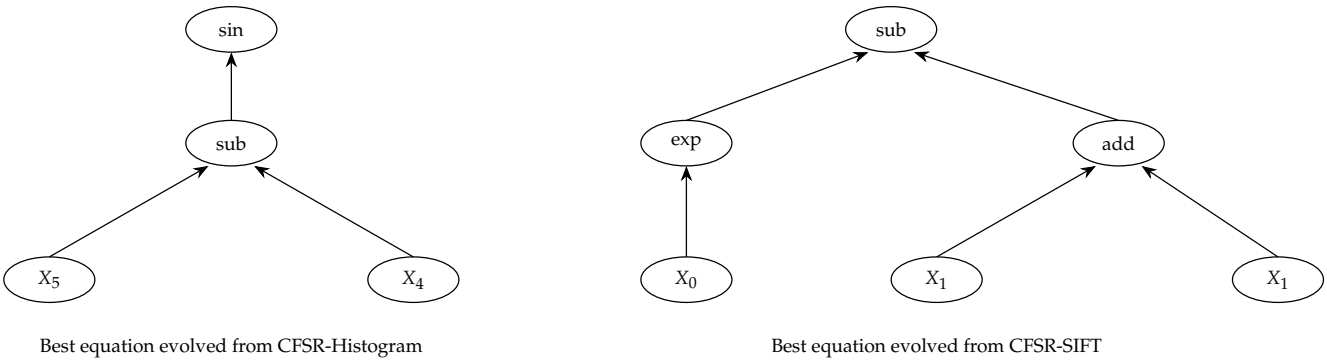



**Figure 7.6:** Best equations evolved from CFSR-Stage 2.

The variations between these models showcase the distinct advantages of each descriptor. The Histogram model concentrates solely on local features, capturing intricate texture details in specific facial areas. On the other hand, the SIFT model balances local and global features, enabling it to capture precise local patterns and the overall facial arrangement. Both models employ non-linear operations (sine, exponential, subtraction), indicating that the relationship between facial features and emotional expressions in the arousal-valence space is complex and not easily captured by simple linear combinations of features.

As CFSR-Histogram demonstrates the best performance among all the CFSR models, it will be used for comparison with FD-GPSR. FD-GPSR achieves a slightly better minimum test loss of 0.5508 compared to CFSR-Histogram's 0.5571. However, CFSR-Histogram shows more consistent performance across runs, with a lower standard deviation of 0.0053 in test loss compared to FD-GPSR's 0.01351. In terms of average test loss, CFSR-Histogram (0.5609) outperforms FD-GPSR (0.5670). Both approaches generate interpretable models, but in different ways. FD-GPSR produces mathematical equations that

directly relate input features to emotion predictions. These equations involve trigonometric functions, exponentials, and constants, capturing complex, non-linear relationships. CFSR-Histogram's two-stage approach provides interpretability at both the feature extraction and symbolic regression levels. In the example shown by Figure 7.6 (left), the best individual from stage one selects meaningful facial regions, while the best equation from stage two captures non-linear interactions between these regions using the sine function. CFSR-Histogram's interpretability is more hierarchical and localized, while FD-GPSR's equations offer a global, direct interpretation.

## 7.3. Chapter conclusion

Chapter 7 introduced the Combined Feature Symbolic Regression (CFSR) approach, a two-stage genetic programming method for facial emotion recognition. CFSR demonstrated promising results, with the Histogram descriptor showing the best overall performance among the tested descriptors. Compared to previous methods like IDGP and FD-GPSR, CFSR offered competitive performance and meaningful interpretability. Throughout our research, we have primarily focused on conventional methods for feature extraction in FER. Could the integration of deep learning-based feature extraction with symbolic regression further improve model performance and interpretability? Chapter 8 explores a hybrid approach combining the power of deep learning with the interpretability of genetic programming-based symbolic regression. By leveraging an EfficientNet-based neural network for feature extraction and applying symbolic regression to these deep features, the next chapter aims to push the boundaries of both performance and interpretability in facial emotion recognition systems.

# Chapter 8
# A Hybrid Method Combining Genetic Programming with Neural Network

This chapter explores a hybrid approach to FER that combines the power of deep learning with the interpretability of genetic programming-based symbolic regression (GPSR). The main objectives are to develop a hybrid model that integrates an EfficientNet-based neural network for feature extraction with a GPSR framework for interpretable prediction, compare the performance of the hybrid model to an end-to-end EfficientNet-based FER model, and analyze the interpretability of the equations generated by the hybrid model and compare them to FD-GPSR introduced in Chapter 6.

Deep learning has gained popularity for handling the complexities of emotion recognition in natural, real-world environments due to the availability of more effective training datasets focused on facial expressions [52]. Deep learning methods utilize an "end-to-end" learning approach, which spans from input to classification or regression, thereby reducing the reliance on manual feature extraction. However, these methods require large, annotated datasets to prevent overfitting, and their models are often criticized for being "black boxes" due to their lack of interpretability [52]. The hybrid approach aims to address these limitations by combining the feature extraction capabilities of deep learning with the interpretability of GPSR. The hybrid model consists of two main components: an EfficientNet-B3 model used to extract 1,536 facial features from grayscale images and a PySR regressor model trained on the extracted features to predict arousal and valence using interpretable equations. The chapter provides detailed algorithms for the hybrid model's feature extraction and prediction parts.

## 8.1. Program structure

The hybrid model integrates a neural network (EffiicientNet-based FER) component into the existing GPSR framework. The neural network acts as a feature extractor, while the GPSR framework using PySR acts as an interpretable predictor. The hybrid model program is divided into three main phases: initialization, feature extraction, and model training and prediction. In the initialization phase, the EfficientNet-B3 model and weights are loaded from torchvision models, and the dataset is prepared with data loaders for training and testing. In the feature extraction phase, an EfficientNet-B3 model instance is created with pre-trained ImageNet weights and modified to accept grayscale images. This modification involves replacing the first convolutional layer and removing the classifier layer. The model then extracts features from both training and test data, saving these features along with their corresponding targets. Data preprocessing follows, where the saved features are loaded and normalized using a MinMaxScaler with a feature range of (-1, 1). The model training and prediction phase begins with creating and configuring a PySRRegressor model, setting hyperparameters such as the number of iterations, binary operators, maximum equation size, batching, and nested constraints. The loss function is defined, and the model is fitted to the normalized training data. The best equation is retrieved, and predictions are made on the normalized test features.

**Algorithm 7** Hybrid Model: Feature Extraction Part

**Require:** Paths to the dataset
**Ensure:** 1536 facial features

```
Import efficientnet_b3 and EfficientNet_B3_Weights from torchvision.models
Create an EfficientNet-B3 model with pre-trained ImageNet weights
function CreateFeatureExtractor(model)
    Modify the first convolutional layer to accept grayscale images
    Create a new Conv2d layer with 1 input channel
    Copy weights from the original layer, averaging across the channel dimension
    Replace the original layer with the new one
    Remove the classifier layer return new model as feature_extractor
end function
function ExtractFeatures(data_loader, feature_extractor)
    Initialize empty lists for features and targets
    Set the feature extractor to evaluation mode
    Iterate through the data loader
        Extract features using the feature extractor
        Append features and labels to lists return stacked features and targets as numpy arrays
end function
Load the dataset
Create dataloaders
Call function CreateFeatureExtractor with the initialized model
Extract features for training data
    Call function ExtractFeatures with train_loader and feature_extractor
    Store results in train_features and train_targets
Extract features for test data
    Set feature_extractor to evaluation mode
    Call function ExtractFeatures with test_loader and feature_extractor
    Store results in test_features and test_targets
Save training features, test features and the corresponding targets
```

**Algorithm 8** Hybrid Model: Prediction Part

**Require:** Paths to training/test features, training/test targets
**Ensure:** Trained hybrid model and performance metrics

```
Load the training and testing features from the feature extraction part
Preprocess the data using a MinMaxScaler object with feature_range equal to (-1, 1)
Create and configure the PySRRegressor model
    Set hyperparameters: niterations, binary_operators, maxsize, batching, nested_constraints, etc.
    Define the loss function using formula (3)
Fit the PySRRegressor model by calling pysr_model.fit() with normalized data
Get the best equation found by the model using pysr_model.sympy()
Make predictions on the test set using pysr_model.predict(normalized_test_features)
Evaluate the model, calculate test loss, RMSE, SAGR
Print the results, best equation, test loss, RMSE, SAGR and training time
```

## 8.2. Experimental evaluation

### 8.2.1. Experimental settings

| Feature Extraction Part | | Symbolic Regression Part | |
|---|---|---|---|
| Parameter | Value | Parameter | Value |
| Optimizer | RMSProp | Population size | 30 |
| Decay Rate | 0.9 | Number of Generations | 15 |
| Momentum Rate | 0.9 | Maximum Equation Size | 100 |
| Batch Norm Momentum | 0.99 | Binary Operations | +, -, ×, / |
| Learning Rate | 0.256 | Unary Operations | exp, sin, cos, log |
| Learning Decay Rate | 0.97 per 2.4 epochs | Nested Constraints | Limit trigonometric nesting |
| Activation Function | SiLU (Swish-1) | Parsimony Parameter | 0.20 |
| Dropout Rate | 0.2 | Regression Model | PySRRegeressor |

**Table 8.1:** Hybrid model's experimental settings.

The model's feature extraction part uses transfer learning, so most important parameters are predefined. As a result, this section will focus on the prediction part. The symbolic regression settings in the hybrid model mirror those of the FD-GPSR model, as both utilize the PySR library, ensuring a fair comparison. The key parameters include 15 iterations (generations), 30 individuals for population size, and a maximum equation size of 100. The equations in the hybrid model are constructed using standard binary operators such as addition (+), subtraction (-), multiplication (×), and division (/), as well as unary operators like cosine (cos), exponential (exp), sine (sin), and logarithm (log). To guide the exploration process, an operator-level "constraints" parameter is employed. This constraint, specified as nested_constraints={"sin": {"sin": 0, "cos": 0}, "cos": {"sin": 0, "cos": 0}}, restricts the nesting of sine and cosine functions within each other. Furthermore, a parsimony parameter is utilized to manage the complexity of the expressions by penalizing overly intricate equations. In this case, a higher parsimony value of 0.20 is used, which encourages the generation of simpler and more concise equations.

### 8.2.2 Results and discussions

The final model—the hybrid model using EfficientNet-based FER for feature extraction—provides 1536 features, which will then be fed to a GPSR model to predict arousal and valence. Figure 8.2 illustrates the evolution of the averaged training loss over 15 generations and ten runs in the hybrid approach. The program has an instance of the PySRRegressor model with specific hyperparameters and fits the model on the normalized training features and targets using the fit() method. At each generation, after fitting the model and calculate the fitness according to equation (3), it retrieves the best individual equation using the sympy() method and output the training loss. There is a clear downward trend in the training loss as the number of generations increases, indicating that the hybrid model is effectively learning and improving its performance over time. The largest drop in loss occurs in the early generations, particularly between generations 0 and 3, where the curve is steepest. The model makes its most substantial gains early in the training process. As the generations progress, the rate of improvement slows down, as evidenced by the flattening of the curve towards the later generations, indicating diminishing returns. The loss appears to be approaching a plateau around generation 8, suggesting that the model is converged. The deep learning component likely provides a strong initial feature set, enabling the rapid early improvements, while the genetic programming aspect offers continued refinement, allowing the

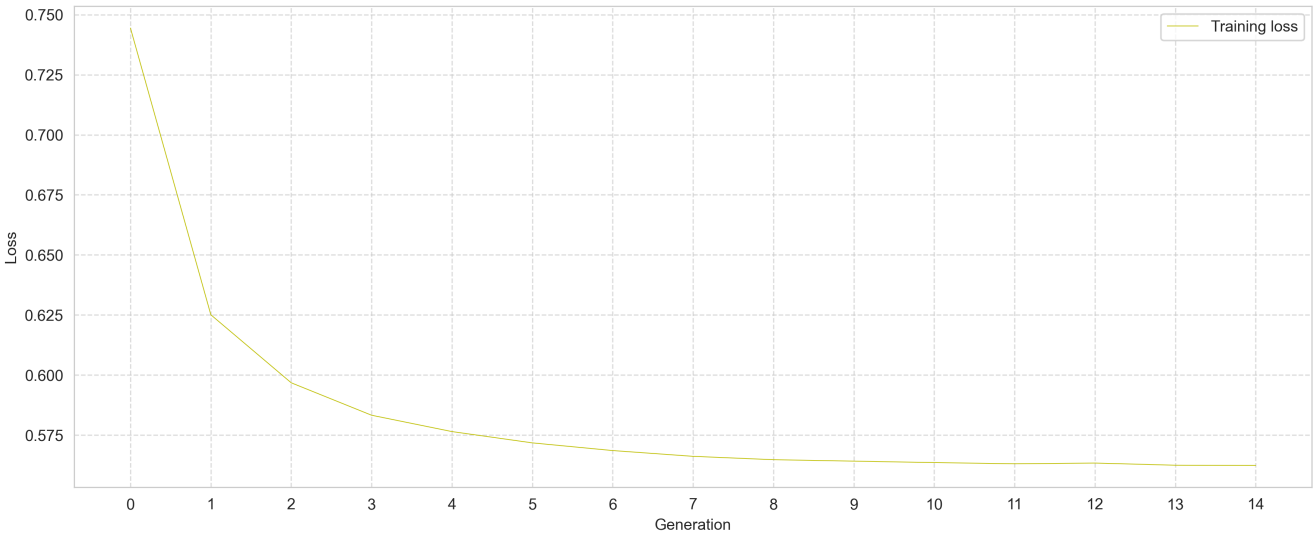



**Figure 8.2:** Hybrid model's average training loss across ten runs.

model to capture more complex relationships in the data over time [53].

As neural networks work best with large datasets, the amount of training data for them is 3840 images. Thus, the most compatible model for comparison with this model is the first model - end-to-end EfficicientNet-based FER using the same amount of training data. The results for the first model in Table 8.3 show a training time of 402.30 seconds, a test loss of 0.5289, a test RMSE of 0.3965, and a test SAGR of 0.7583. These metrics indicate reasonably good performance, with the model achieving a balance between accuracy RMSE and directional correctness SAGR. This chapter's model takes a hybrid approach. The results for this model, as shown in Table 8.3, indicate an average training time of 524.33 seconds, average test loss of 0.5610, average test RMSE of 0.4255, and average test SAGR of 0.7379. This model also shows consistent performance across multiple runs, as evidenced by the low standard deviations in the metrics. The end-to-end transfer learning model outperforms the hybrid model in all metrics. The end-to-end approach achieves lower error rates and higher sign agreement, suggesting it better captures the nuances of the emotion recognition task.

| Metrics | End-to-End | Hybrid | | | |
|---|---|---|---|---|---|
| | Average | Average | SD | Max | Min |
| Training Time | 402.30 | 524.33 | 18.46 | 556.32 | 504.36 |
| Test Loss | 0.5289 | 0.5610 | 0.0047 | 0.5678 | 0.5524 |
| Test RMSE | 0.3965 | 0.4255 | 0.0035 | 0.4300 | 0.4190 |
| Test SAGR | 0.7583 | 0.7379 | 0.0077 | 0.7500 | 0.7250 |

**Table 8.3:** Performance metrics comparison between the end-to-end EfficientNet-based model and the hybrid model. SD: Standard Deviation.

Despite using the same initial features from EfficientNet, the performance difference between the two models can be attributed to several factors. The end-to-end approach allows the entire network to be fine-tuned, potentially learning more task-specific features throughout the network. In contrast, the symbolic regression approach relies on fixed features extracted from EfficientNet [54]. Neural networks can capture complex, non-linear relationships more easily than symbolic regression, which may struggle to find optimal mathematical expressions for highly non-linear patterns in the data [55]. The end-to-end model can learn intricate interactions between features across different layers, while symbolic regression is limited to combining the extracted features in more simplistic ways [56]. Neural networks benefit from

| Best Equation |
|---|
| $[exp(-cos(x1502) + sin(0.28022465)) * 0.4774625, (cos(x451) + 0.03797187) * 0.33291754 * x24]$ |
| $[sin(exp(x1048)), x1323 * cos(-1.1894691) * cos(x1107)]$ |
| $[sin(exp(x841)), 0.20843388 * x149]$ |
| $[sin(exp(x3)), cos(-1.2331496)/(exp(x762) - 1.9562364)]$ |
| $[(0.4048087 - 1 * 0.3133288) * (x45 + exp(0.3133288)) + 0.3133288, ((x1171 - x184)/cos(-0.30035293) - 1 * (-1.0483197)) * (-0.18830588)/cos(-0.30035293)]$ |
| $[cos(x307)/1.5465133, sin(0.23879668) * sin(-1.131354) * x1142 * (-0.9519081)]$ |
| $[sin(exp(x1012)), (-0.282 - 0.09018743) * cos(x741)]$ |
| $[sin(exp(x344)), sin(x1486) * 0.25574222]$ |
| $[sin(exp(x1346)), 0.20630167 * x655]$ |
| $[sin(exp(x807)), x1168 * 0.2073953]$ |

**Table 8.4:** Best equations obtained from the hybrid model.

gradient-based optimization, which can be more efficient in high-dimensional spaces than the evolutionary approach used in genetic programming [57]. While symbolic regression can produce interpretable equations, it may lack the expressiveness to capture all the subtle patterns a deep neural network can learn [58].

The symbolic regression model FD-GPSR, which uses various image descriptors for feature extraction, produces a set of best equations (Table 6.7) that are generally simpler and more diverse in structure than the hybrid model. These equations often involve basic arithmetic operations, trigonometric functions, and exponentials, typically combining 2-3 input features. For example,
$[cos(x_{112} * 1.4215789 - (x_{104} + x_{67})) * cos(0.14018624)0.45621794, x_{63}(-0.3096085)]$
shows a complex trigonometric relationship for arousal prediction, while using a simple linear term for valence. This diversity suggests that the model explores different mathematical relationships between the extracted features and the emotion dimensions. In contrast, the hybrid model, which uses EfficientNet for feature extraction, consistently produces equations with a recurring pattern, particularly for arousal prediction (Table 8.4). Most arousal equations take the form $sin(exp(x_i))$, where $x_i$ is a high-numbered feature (e.g., $x_{1048}, x_{841}, x_3$). This consistency suggests that the EfficientNet features contain similar, highly relevant information for arousal across different runs. The valence equations in this model show more variety, often involving trigonometric functions, exponentials, and more complex combinations of features. The complexity of the hybrid model's equations, especially for valence, indicates that the EfficientNet features might capture more nuanced relationships that require more sophisticated mathematical expressions to model effectively. For instance,
$((x_{1171} - x_{184})/cos(-0.30035293) - 1 * (-1.0483197)) * (-0.18830588)/cos(-0.30035293)$
shows a complex interplay of multiple features and trigonometric functions.

| Metrics | IDGP | | FD-GPSR | | CFSR-Histogram | | Hybrid | |
|---|---|---|---|---|---|---|---|---|
| | Average | Min | Average | Min | Average | Min | Average | Min |
| Training Loss | 0.5166 | 0.5117 | 0.5511 | 0.5489 | 0.5404 | 0.5352 | 0.5624 | 0.5613 |
| Test Loss | 0.5612 | 0.5525 | 0.5670 | 0.5508 | 0.5609 | 0.5571 | 0.5610 | 0.5524 |

**Table 8.5:** Training and test loss summary of all GP-related models.

Table 8.5 presents the average and minimum values for both training and test losses metrics for various GP-related models in facial expression recognition. It compares the performance of different approaches including IDGP (Image Descriptor-based Genetic Programming), FD-GPSR (Feature Descriptor-based Genetic Programming for Symbolic Regression), CFSR-Histogram (Combined Feature Symbolic Regression with Histogram) and the Deep Learning-based Genetic Programming Hybrid Model. The average training loss for different approaches are: IDGP at 0.5166, FD-GPSR at 0.5511, CFSR-Histogram at 0.5404 and the hybrid model at 0.5624. IDGP appears to have the lowest training loss, suggesting it performs best during the training phase. However, when comparing training and test loss, IDGP shows a significant difference (0.5166 vs 0.5612), indicating potential overfitting. In contrast, FD-GPSR, CFSR-Histogram and the hybrid model demonstrate smaller gaps between their training and test losses. FD-GPSR, in particular, shows the best test performance with the lowest test loss of 0.5508 and a small gap between training and test loss, indicating good generalization. The hybrid model and CFSR-Histogram show similar generalization performance, with test losses between IDGP and FD-GPSR.

FD-GPSR and the hybrid model achieves the best minimum test loss (0.5508 and 0.5524) among all GP-related models, indicating its effectiveness in optimizing feature selection while offer interpretation on the features learned. However, the hybrid model is much faster than the other one, making it more appropriate for when resources is limited. The symbolic regression models, both for general feature learning (IDGP) and specific image descriptors (CFSR), show consistent performance but slightly higher test losses compared to FD-GPSR. All models consistently focus on key facial features like the eyes, nose, and mouth, underscoring their importance in facial expression analysis. Moreover, they also occasionally select less conventional areas, suggesting they capture subtle cues that human observers might overlook. The symbolic regression models tend to favor more straightforward, more interpretable equations, which aligns to produce understandable models. In contrast, the GP-based feature learning model shows variability in the complexity of the evolved solutions, provided by the large number of features extracted. The Histogram descriptor in the symbolic regression with image descriptors (CFSR) shows the most consistent performance, indicating reliability across different seeds. The feature learning model (IDGP) also demonstrates robustness in feature selection, with stable performance metrics despite variability in the number of features.

## 8.3. Chapter conclusion

The hybrid model combining EfficientNet-based feature extraction with genetic programming-based symbolic regression offers a balanced approach to facial emotion recognition, prioritizing both performance and interpretability. While outperformed by the end-to-end EfficientNet model in terms of test metrics, the hybrid approach provides valuable insights through interpretable equations. It achieves competitive results with an average test loss of 0.5610. The model generates consistent equations for arousal predictions, often in the form of $sin(exp(x_i))$, revealing relationships between extracted features and emotional dimensions. Compared to other GP-related models, the hybrid approach shows competitive performance, especially in minimum test loss. While sacrificing some performance, it gains in interpretability and potential computational efficiency. This approach presents a promising direction for facial emotion recognition applications where understanding the decision-making process is crucial.

# Chapter 9
# Conclusions

This project has explored four GP-based approaches to facial expression recognition (FER) in the arousal-valence space. The study compared different feature extraction methods, regression models, and hybrid approaches to understand their effectiveness in predicting arousal and valence values from facial images. Their contributions and findings can be summarized as follows:

For the feature extraction methods:

- IDGP automatically selects and combines image descriptors like LBP, HOG, SIFT to extract discriminative global and/or local features.
- CFSR applies symbolic regression to features extracted by different descriptors (Histogram, DIF, SIFT, LBP) to create new transformed features.
- Histogram descriptor showed the best overall performance in CFSR, followed by SIFT.

For the regression models:

- SVR is suited for predicting continuous arousal and valence values and has shown good performance in handling variations in facial images.
- GPSR like FD-GPSR produces interpretable models and does not require assumptions about data distribution or model structure like SVR.
- FD-GPSR achieved a better minimum test loss than IDGP while using fewer features.

For the hybrid approaches:

- Combines deep learning for feature extraction with GPSR for interpretable prediction
- Uses an EfficientNet-B3 model to extract 1,536 facial features and a PySR regressor to predict arousal and valence with interpretable equations
- Aims to leverage deep learning's feature extraction capabilities with GPSR's interpretability

The exploratory data analysis (EDA) provided valuable insights into the distribution of color channels, arousal, and valence values in the AffectNet dataset. The analysis revealed that color information could be simplified to grayscale without significantly losing relevant features, as evidenced by the similar right-skewed patterns across red, green, and blue channels. The circular pattern in the arousal-valence scatter plot suggested a weak correlation between these dimensions, supporting their treatment as independent variables in prediction models.

Preprocessing techniques such as histogram equalization and denoising showed varying effects on model performance, with their impact being more pronounced on smaller datasets. For larger datasets,

the difference in performance between preprocessed and non-preprocessed images was minimal, suggesting that deep learning models can effectively learn to handle image variations with sufficient data.

Among the various approaches tested, the genetic programming for symbolic regression-based feature learning model (FD-GPSR) and the hybrid model combining EfficientNet-based feature extraction with symbolic regression achieved the best performance. These results highlight the potential of combining evolutionary algorithms with deep learning techniques for effective feature selection and prediction in FER tasks, especially when computation power is limited. The hybrid model also demonstrates a good balance between performance and interpretability. While not outperforming the best-performing model of end-to-end EfficientNet-based, the symbolic regression models offered valuable insights into the interpretability of facial expression analysis. The equations produced by these models, revealed informative mathematical relationships between extracted features and emotion dimensions. This interpretability could be crucial in applications where understanding the reasoning behind predictions is as important as the predictions themselves [59]. The consistent focus on critical facial regions like eyes, nose, and mouth across different models underscores the importance of these features in emotion recognition tasks. Some models' occasional selection of less conventional areas suggests that subtle, non-obvious features may also play a role in accurate emotion prediction.

## 9.1. Limitations

Although this study aimed to explore various aspects of FER, several limitations should be acknowledged. The computational intensity of some methods, particularly genetic programming and symbolic regression, limited the dataset size that could be practically used, potentially preventing these methods from fully leveraging the benefits of larger datasets. While deep learning models like EfficientNet showed faster and superior performance, their lack of interpretability remains a significant limitation, whereas more interpretable models like symbolic regression may sacrifice some performance for explainability. Due to time and resource constraints, an exhaustive search of hyperparameters for each model was not feasible, meaning that the reported performances may not represent the absolute best each method could achieve. The project focused solely on static image analysis, not considering temporal information that could be valuable in real-world emotion recognition scenarios [60]. Additionally, the models were primarily trained and tested on subsets of the AffectNet dataset, and cross-dataset validation could provide more robust insights into the generalizability of the methods [61]. Lastly, the reliance on automated facial landmark detection methods could introduce biases, particularly for faces that deviate from the training data used in these detection algorithms [62].

## 9.2. Future work

Future research in FER could explore several promising directions. Multi-modal emotion recognition, integrating voice, body posture, and physiological signals alongside facial expressions, could provide a more comprehensive understanding of emotional states [63]. Temporal analysis using video sequences and techniques like recurrent neural networks or 3D convolutional networks could offer insights into the dynamics of emotion expression over time [64]. Advanced interpretability techniques, such as attention mechanisms or layer-wise relevance propagation, could enhance the explainability of deep learning models [65]. Further optimization of hybrid models combining deep learning feature extraction and symbolic regression could improve performance [54]. Developing methods for continuous learning would allow models to adapt to new data over time without forgetting previously learned patterns, enhancing long-term deployment capabilities [66]. Finally, extending models to recognize more nuanced or mixed emotional states could provide a more detailed understanding of human emotional expressions [67].

# References


[1] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. URL http://arxiv.org/abs/1905.11946.

[2] Ying Bi, Bing Xue, and Mengjie Zhang. *Genetic Programming for Image Classification: An Automated Approach to Feature Learning*. Adaptation, Learning, and Optimization. Springer Cham, 2021. doi: 10.1007/978-3-030-65927-1.

[3] Nour Makke and Sanjay Chawla. Interpretable scientific discovery with symbolic regression: A review, 2023. URL https://arxiv.org/abs/2211.10873.

[4] Ali Mollahosseini, Behzad Hasani, and Mohammad H. Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 10(1):18–31, January 2019. ISSN 2371-9850. doi: 10.1109/taffc.2017.2740923. URL http://dx.doi.org/10.1109/TAFFC.2017.2740923.

[5] Antoine Toisoul, Jean Kossaifi, Adrian Bulat, Georgios Tzimiropoulos, and Maja Pantic. Estimation of continuous valence and arousal levels from faces in naturalistic conditions. *Nature Machine Intelligence*, 3, 01 2021. doi: 10.1038/s42256-020-00280-0.

[6] David Millan Escriva, Prateek Joshi, Vinicius G. Mendonca, and Roy Shilkrot. *Building Computer Vision Projects with OpenCV 4 and C++*. Packt Publishing, March 2019. Learning Path.

[7] Joseph Howse and Joe Minichino. *Learning OpenCV 5 Computer Vision with Python*. Packt Publishing, 4 edition, December 2024. Early Access.

[8] Shamshad Ansari. *Building Computer Vision Applications Using Artificial Neural Networks: With Examples in OpenCV and TensorFlow with Python*. Apress, 2 edition, November 2023. Second Edition.

[9] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-Local Means Denoising. *Image Processing On Line*, 1:208–212, 2011. https://doi.org/10.5201/ipol.2011.bcm_nlm.

[10] Ali Ismail Awad and Mahmoud Hassaballah, editors. *Image Feature Detectors and Descriptors: Foundations and Applications*. Studies in Computational Intelligence. Springer Cham, 2016. doi: 10.1007/978-3-319-28854-3.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 11 2004. ISSN 1573-1405. doi: 10.1023/B:VISI.0000029664.99615.94. URL https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[12] Andrea Vedaldi and Brian Fulkerson. Vlfeat: an open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1469–1472, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874249. URL https://doi.org/10.1145/1873951.1874249.

[13] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33833-8.

[14] Tai Sing Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996. doi: 10.1109/34.541406.

[15] Saeid Fazli, Reza Afrouzian, and Hadi Seyedarabi. High- performance facial expression recognition using gabor filter and probabilistic neural network. In *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 4, pages 93–96, 2009. doi: 10.1109/ICICISYS.2009.5357716.

[16] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Inc., September 2016. ISBN 978-1449369415.

[17] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 3 edition, October 2022. ISBN 978-1098125974.

[18] John R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, 6 1994. ISSN 1573-1375. doi: 10.1007/BF00175355. URL `https://doi.org/10.1007/BF00175355`.

[19] Wolfgang Banzhaf. Genetic programming for pedestrians. In *Proceedings of the International Conference on Genetic Algorithms*, page 628, 1993.

[20] Markus F. Brameier and Wolfgang Banzhaf. *Linear Genetic Programming*. Genetic and Evolutionary Computation. Springer Science & Business Media, 2007. ISBN 978-0-387-31029-9. doi: 10.1007/978-0-387-31030-5.

[21] P. A. Whigham. Grammatically-based genetic programming. In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, volume 16, pages 33–41, 1995.

[22] Julian F. Miller and Peter Thomson. Cartesian genetic programming. In Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian Miller, Peter Nordin, and Terence C. Fogarty, editors, *Genetic Programming*, pages 121–132, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-46239-2.

[23] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[24] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69–93, 1991.

[25] David J Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2):199–230, 1995.

[26] Harith Al-Sahaf, Ausama Al-Sahaf, Bing Xue, Mark Johnston, and Mengjie Zhang. Automatically evolving rotation-invariant texture image descriptors by genetic programming. *IEEE Transactions on Evolutionary Computation*, 21:83–101, 01 2016. doi: 10.1109/TEVC.2016.2577548.

[27] Wenlong Fu. *Feature extraction in edge detection using genetic programming*. Phd thesis, Victoria University of Wellington, New Zealand, 2014.

[28] Harith Al-Sahaf, Andy Song, Kourosh Neshatian, and Mengjie Zhang. Two-tier genetic programming: Towards raw pixel-based image classification. *Expert Systems with Applications*, 39: 12291–12301, 11 2012. doi: 10.1016/j.eswa.2012.02.123.

[29] Morgan Atkins, Kourosh Neshatian, and Mengjie Zhang. A domain independent genetic programming approach to automatic feature extraction for image classification. In *Proceedings of IEEE congress on evolutionary computation*, pages 238–245, 06 2011. doi: 10.1109/CEC.2011.5949624.

[30] Andrew Lensen, Harith Al-Sahaf, Mengjie Zhang, and Bing Xue. Genetic programming for region detection, feature extraction, feature construction and classification in image data. In *Proceedings of European conference on genetic programming*, 03 2016. ISBN 978-3-319-30667-4. doi: 10.1007/978-3-319-30668-1_4.

[31] Riccardo Poli, William Langdon, and Nicholas Mcphee. *A Field Guide to Genetic Programming*. Published via http://lulu.com, 01 2008. ISBN 978-1-4092-0073-4.

[32] Ayodele Oluleye. *Exploratory Data Analysis with Python Cookbook*. Packt Publishing, June 2023. Time to complete: 7h 52m.

[33] Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *IEEE Transactions on Affective Computing*, 13(3):1195–1215, 2022. doi: 10.1109/TAFFC.2020.2981446.

[34] Gianmarco Ipinze Tutuianu, Yang Liu, Ari Alamäki, and Janne Kauttonen. Benchmarking deep facial expression recognition: An extensive protocol with balanced dataset in the wild. *Engineering Applications of Artificial Intelligence*, 136:108983, 2024. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2024.108983. URL https://www.sciencedirect.com/science/article/pii/S0952197624011412.

[35] Antoine Toisoul, Jean Kossaifi, Adrian Bulat, Georgios Tzimiropoulos, and Maja Pantic. Estimation of continuous valence and arousal levels from faces in naturalistic conditions. *Nature Machine Intelligence*, 3, 01 2021. doi: 10.1038/s42256-020-00280-0.

[36] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better?, 2019. URL https://arxiv.org/abs/1805.08974.

[37] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312: 135–153, October 2018. ISSN 0925-2312. doi: 10.1016/j.neucom.2018.05.083. URL http://dx.doi.org/10.1016/j.neucom.2018.05.083.

[38] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2016. doi: 10.1109/QoMEX.2016.7498955.

[39] Tadas Baltrušaitis, Ntombikayise Banda, and Peter Robinson. Dimensional affect recognition using continuous conditional random fields. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 1–8, 2013. doi: 10.1109/FG.2013.6553785.

[40] Mihalis A. Nicolaou, Hatice Gunes, and Maja Pantic. Continuous prediction of spontaneous affect from multiple cues and modalities in valence-arousal space. *IEEE Transactions on Affective Computing*, 2(2):92–105, 2011. doi: 10.1109/T-AFFC.2011.9.

[41] Ligang Zhang, Dian Tjondronegoro, and Vinod Chandran. Representation of facial expression categories in continuous arousal–valence space: Feature and correlation. *Image and Vision Computing*, 32(12):1067–1079, 2014. ISSN 0262-8856. doi: https://doi.org/10.1016/j.imavis.2014.09.005. URL https://www.sciencedirect.com/science/article/pii/S0262885614001449.

[42] Hung-Hsu Tsai and Yi-Cheng Chang. Facial expression recognition using a combination of multiple facial features and support vector machine. *Soft Computing*, 22(13):4389–4405, Jul 2018. ISSN 1433-7479. doi: 10.1007/s00500-017-2634-3. URL https://doi.org/10.1007/s00500-017-2634-3.

[43] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner Gardner, Marc Parizeau, and Christian Gagné. Deap: evolutionary algorithms made easy. *J. Mach. Learn. Res.*, 13(1): 2171–2175, July 2012. ISSN 1532-4435.

[44] Qi Chen, Mengjie Zhang, and Bing Xue. Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression. *IEEE Transactions on Evolutionary Computation*, 21(5):792–806, 2017. doi: 10.1109/TEVC.2017.2683489.

[45] Qi Chen and Bing Xue. *Generalisation in Genetic Programming for Symbolic Regression: Challenges and Future Directions*, pages 281–302. Springer International Publishing, Cham, 2022. ISBN 978-3-030-79092-9. doi: 10.1007/978-3-030-79092-9_13. URL https://doi.org/10.1007/978-3-030-79092-9_13.

[46] Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression.jl, 2023. URL https://arxiv.org/abs/2305.01582.

[47] F. O. de Franca, M. Virgolin, M. Kommenda, M. S. Majumder, M. Cranmer, G. Espada, L. Ingelse, A. Fonseca, M. Landajuela, B. Petersen, R. Glatt, N. Mundhenk, C. S. Lee, J. D. Hochhalter, D. L. Randall, P. Kamienny, H. Zhang, G. Dick, A. Simon, B. Burlacu, Jaan Kasak, Meera Machado, Casper Wilstrup, and W. G. La Cava. Interpretable symbolic regression for data science: Analysis of the 2022 competition, 2023. URL https://arxiv.org/abs/2304.01117.

[48] PySR Contributors. Pysr api reference. https://astroautomata.com/PySR/api/, 2024. URL https://astroautomata.com/PySR/api/. Accessed on [4 Oct 2024].

[49] Haythem Ghazouani. A genetic programming-based feature selection and fusion for facial expression recognition. *Applied Soft Computing*, 103:107173, 2021. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2021.107173. URL https://www.sciencedirect.com/science/article/pii/S156849462100096X.

[50] Ying Bi, Bing Xue, and Mengjie Zhang. Multi-objective genetic programming for feature learning in face recognition. *Applied Soft Computing*, 103:107152, 02 2021. doi: 10.1016/j.asoc.2021.107152.

[51] Yi Mei, Qi Chen, Andrew Lensen, Bing Xue, and Mengjie Zhang. Explainable artificial intelligence by genetic programming: A survey. *IEEE Transactions on Evolutionary Computation*, 27(3):621–641, 2023. doi: 10.1109/TEVC.2022.3225509.

[52] Yunxin Huang, Fei Chen, Shaohe Lv, and Xiaodong Wang. Facial expression recognition: A survey. *Symmetry*, 11(10), 2019. ISSN 2073-8994. doi: 10.3390/sym11101189. URL https://www.mdpi.com/2073-8994/11/10/1189.

[53] Pegah Mavaie, Lawrence Holder, and Michael K Skinner. Hybrid deep learning approach to improve classification of low-volume high-dimensional data. *BMC Bioinformatics*, 24(1):419, Nov 2023. ISSN 1471-2105. doi: 10.1186/s12859-023-05557-w.

[54] Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases, 2020. URL https://arxiv.org/abs/2006.11287.

[55] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales, 2021. URL https://arxiv.org/abs/2106.06427.

[56] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113. URL https://www.pnas.org/doi/abs/10.1073/pnas.1517384113.

[57] Miles Cranmer. Pysr: Fast & parallelized symbolic regression in python/julia. *Zenodo, September*, 2020.

[58] Mariek Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science (New York, N.Y.)*, 324:81–5, 05 2009. doi: 10.1126/science.1165893.

[59] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017. URL https://arxiv.org/abs/1702.08608.

[60] Wen-Sheng Chu, Fernando Torre, and Jeffrey Cohn. Learning spatial and temporal cues for multi-label facial action unit detection. In *In 2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, pages 25–32, 05 2017. doi: 10.1109/FG.2017.13.

[61] Rameswar Panda, Jianming Zhang, Haoxiang Li, Joon-Young Lee, Xin Lu, and Amit K. Roy-Chowdhury. Contemplating visual emotions: Understanding and overcoming dataset bias, 2018. URL https://arxiv.org/abs/1808.02212.

[62] Azin Asgarian, Shun Zhao, Ahmed B. Ashraf, M. Erin Browne, Kenneth M. Prkachin, Alex Mihailidis, Thomas Hadjistavropoulos, and Babak Taati. Limitations and biases in facial landmark detection – an empirical study on older adults with dementia, 2019. URL https://arxiv.org/abs/1905.07446.

[63] Sidney D'mello and Jacqueline Kory. A review and meta-analysis of multimodal affect detection systems. *ACM Computing Surveys*, 47:1–36, 02 2015. doi: 10.1145/2682899.

[64] A. Kołakowska, A. Landowska, M. Szwoch, W. Szwoch, and M. R. Wróbel. *Emotion Recognition and Its Applications*, pages 51–62. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08491-6. doi: 10.1007/978-3-319-08491-6_5. URL https://doi.org/10.1007/978-3-319-08491-6_5.

[65] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Lecture Notes in Computer Science. Springer Cham, 2019. doi: 10.1007/978-3-030-28954-6. eBook Packages: Computer Science, Computer Science (R0).

[66] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

[67] Brais Martinez, Michel F. Valstar, Bihan Jiang, and Maja Pantic. Automatic analysis of facial actions: A survey. *IEEE Transactions on Affective Computing*, 10(3):325–347, 2019. doi: 10.1109/TAFFC.2017.2731763.