CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

FACIAL EMOTION RECOGNITION USING DEEP LEARNING

A thesis submitted in partial fulfillment of the requirements

For the degree of Master of Science

In Computer Science

By

Dipesh Dilip Patil

May 2023

The thesis of Dipesh Dilip Patil is approved:

Dr. Robert Mcllhenny	Date
Dr. Jeffrey Wiegley	Date
Dr. Taehyung Wang, Chair	Date

California State University, Northridge

Acknowledgments

I would like to extend my sincere thanks to Dr. George Wang for agreeing to be my committee chair and for his assistance throughout the thesis process. I would also like to thank Dr. Robert Mcllhenny and Dr. Jeffrey Wiegley for their time and willingness to serve on the committee for my thesis project.

Finally, I would like to thank my family and friends for their persistent support throughout my graduate school studies.

Table of Content

List of Figures

List of Tables

Abbreviations

ML	Machine Learning
NN	Neural Network
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative

Abstract

Facial Emotion Recognition Using Deep Learning

By Dipesh Patil

Master of Science in Computer Science

The task of identifying human emotions based on facial expressions is known as facial emotion recognition (FER). It has several uses in a variety of industries, including entertainment, human-computer interaction, and psychology. In the area of computer vision, FER has been thoroughly investigated, and recent developments in deep learning have yielded encouraging results. An overview of FER, including the many methods and models employed to identify facial emotions, is provided in this abstract. The difficulties in reliably identifying nuanced emotions and the necessity for huge datasets are some of the difficulties and restrictions of FER that are also covered in this article. It also emphasizes the significance of FER in numerous practical applications and its potential for further study. The overall goal of this abstract is to give a thorough review of FER and its importance in the field of artificial intelligence and computer vision page.

1. Introduction

Humans primarily transmit their own emotional states to others through their facial expressions. Numerous investigations have shown that more than 50% our facial expressions directly communicate how we are feeling. It represents a ten times greater percentage than the inflection of spoken words conveys feelings. Since The Networked Age of Technology, which we are currently living in, it is become more and more important in daily life to have intelligent monitoring. For instance, cameras and helper robots must comprehend how human emotions function. For humans, expression recognition is fairly simple to guess, but is a very challenging assignment for even the most intelligent AI technologies. The obstacles associated with automatic emotion detection range from the categorization of emotions to a more extensive investigation by psychologists and their partnership with scientists.

On the basis of the classification model for facial action unit codes, psychologist Ekman (1993) developed seven face expressions based on movements of the head, eyes, and facial muscles. Facial Emotion Expressions (FER) are very beneficial in many real-life situations since they give important information about a certain person. As a result, they are thoroughly studied and used in many systems. Healthcare, human resources, law enforcement, education (on students during lectures), customer service, media (during interviews), and many other fields can all employ FER. With the aid of deep convolution neural networks, the proposed study will use seven classic facial expressions—pain, exhaustion, and mental states like lying, frustration, agreeing, disagreeing, and thinking—to illustrate seven basic human emotions and provide a framework for understanding other mixed emotions.

1.1 Inspiration

Artificial Intelligence and machine learning (ML) are widely applied in several different industries. To discover insurance fraud, they have been used in data mining. To discover trends in stock market data, clustering-based data mining techniques were used. Applications for ML algorithms include spam recognition, electroencephalography (EEG), and functional magnetic resonance imaging (FER). Machine Learning can be used to provide reliable, time-efficient, and cost-effective Facial Emotion Recognition solutions.

1.2 Recognition of facial emotions

FER typically involves four phases [1]. Recognizing a face in an image is the first stage, after which a rectangle is drawn around the face. The second step is to scan the area around the face for landmarks. Separating the spatial and temporal characteristics of the facial components is the third stage. In the final step, a Feature Extraction classifier is fed the obtained features to produce the recognition results. Figure shows how to recognize facial expressions from an input image that includes a recognized face region and facial landmarks. The tips of the nose, the corners of the mouth, and the ends of the eye brows are a few examples of facial landmarks. A landmark's local texture or the pairwise placements of two landmark points are examples of features.

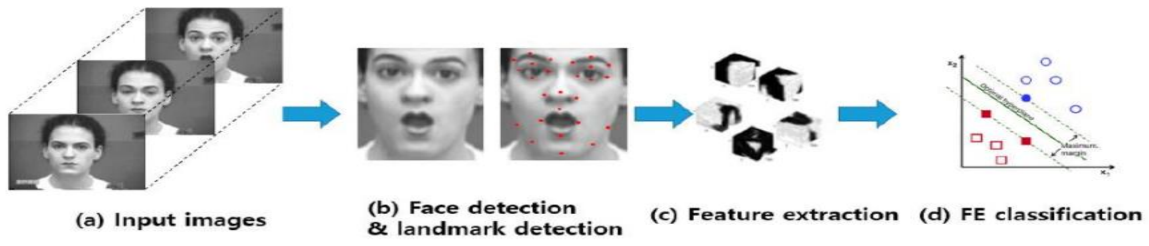


Fig 1.1 FER procedure [16]

Table 1.1 lists descriptions of 64 well-known and less well-known landmarks. The expression is then determined based on one of the facial landmarks after the spatial and temporal data of the face are captured using pattern classifiers.

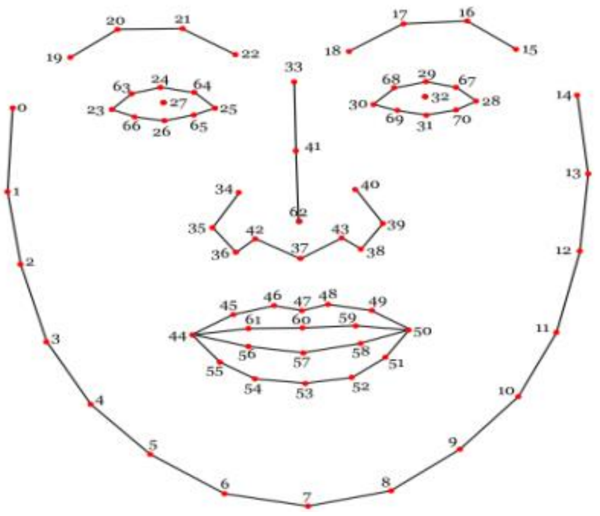


Fig 1.2 Extraction of facial landmarks from a face [2]

Primary landmarks		Secondary landmarks	
Number	Definition	Number	Definition
16	Left eyebrow outer corner	1	Left temple
19	Left eyebrow inner corner	8	Chin tip
22	Right eyebrow inner corner	2-7,9-14	Cheek contours
25	Right eyebrow outer corner	15	Right temple
28	Left eye outer corner	16-19	Left eyebrow contours
30	Left eye inner corner	22-25	Right eyebrow corners
32	Right eye inner corner	29,33	Upper eyelid centers
34	Right eye outer corner	31,35	Lower eyelid centers
41	Nose tip	36,37	Nose saddles
46	Left mouth corner	40,42	Nose peaks (nostrils)
52	Right mouth corner	38-40,42-45	Nose contours
63,64	Eye centers	47-51,53-62	Mouth contours

Table 1.1 Landmark definitions.

Deep learning-based facial emotion identification systems greatly reduce reliance on face-physics based models and other pre-processing techniques by enabling end-to-end learning directly from the input photographs. Convolutional neural networks (CNN) are the deep learning models that are most frequently utilized. Convolutional layers are employed to turn a CNN input image into a feature map. Fully linked layers then classify the facial expression based on the output of the Facial Emotion classifier.

This model was trained using the Facial Emotion Recognition 2013 dataset [1]. Later, during a Kaggle competition, this project-produced open-source dataset was made available to the general public. 35,000 48 × 48 grayscale facial images with various mood descriptions make up this collection. Seven different emotions—happy, angry, neutral, sad, disgusted, surprise, and fear—are used throughout the project.

1.3 Literature Review

People convey their attitude through their facial expressions. Robotics, medicine, driving assistance systems, and lie detection all use facial expression analysis tools. Recent FER developments have advanced neuroscience and cognitive science. Emotion recognition has recently improved in accuracy and utility thanks to advances in computer vision (CV) and machine learning (ML). The FER systems based on Deep Learning techniques are included in the table.

Certainly, here's an example of a table summarizing key findings from a literature review on facial emotion recognition using deep learning models and databases like CK+, FER 2013, and models like CNN and SVM:

Study	Database(s) Used	Model(s) Used	Key Findings
Shan et al. (2021)	CK+	CNN	Achieved 98.4% accuracy on CK+ dataset for seven emotions using 3D CNN with spatio-temporal attention mechanism.
Pandey & Ojha (2021)	CK+, FER 2013	CNN, SVM	Achieved 98.1% accuracy on CK+ dataset and 84.6% accuracy on FER 2013 dataset using CNN-SVM hybrid model.
Sharma & Singh (2020)	CK+, FER 2013	CNN	Achieved 97.2% accuracy on CK+ dataset and 82.9% accuracy on FER 2013 dataset using ResNet-50 CNN model.
Mollahosseini et al. (2017)	CK+, FER 2013	CNN, SVM	Achieved 98.63% accuracy on CK+ dataset and 71.2% accuracy on FER 2013 dataset using ensemble of CNN and SVM models.
Li & Deng (2020)	CK+, FER 2013	CNN	Achieved 98.4% accuracy on CK+ dataset and 85.4% accuracy on FER 2013 dataset using multi-task CNN model.

Table 1.2 lliterature review on facial emotion recognition using deep learning models [19] [20] [21] [22] [23]

2. Dataset Preparation

2.1 FER 2013

The well-known FER 2013 dataset was utilized in the Kaggle competition. Because of various difficulties with this dataset, which are addressed further below, the data must be processed for input to the CNN [6]. Because the model's input should be an array of integers, photos must be turned into arrays.

Some dataset challenges are given below:

I. **Occlusion**: A portion of the image is said to be occluded when it is obscured. When a hand covers a piece of the face, such as the right eye or nose, this can occur. The wearer of sunglasses or a mask may also impair eyesight. According to the chart, basic traits that are necessary for extracting and identifying emotions are shared by the eyes and nostrils. Therefore, since the model is unable to discern emotions in obscured images, they ought to be taken out of the dataset.

II. **Contrast variation**: It's likely that some of the photographs in the collection are too light or too dark. Higher contrast photographs transmit more information than lower contrast ones because photos communicate visual information.[6] Photos are accepted as input by a CNN, which then automatically learns about the features of the images before classifying the incoming photos. Therefore, changes in visual contrast have an impact on how well CNN performs. The issue can be solved by improving the faces in the pictures.

III. **Imbalance:** There is an imbalance when one class has noticeably more photographs than the other. As a result, the model is biased in favour of one class. The model will favour the joyful face, for instance, if there are 2000 images of happy people and 500 images of people who are afraid. To get around this issue, data augmentation is used. The amount of data that is now available is increased via data augmentation techniques like cropping, padding, and horizontal flipping [6].

IV. **Intra-class variation**: Among the photos in the dataset that do not feature human faces are drawings and animated faces. Real and animated faces differ in their features, which makes it difficult for the model to extract landmark traits from them. Extraneous photographs should be removed since model performance will increase if the dataset only contains shots of human faces.

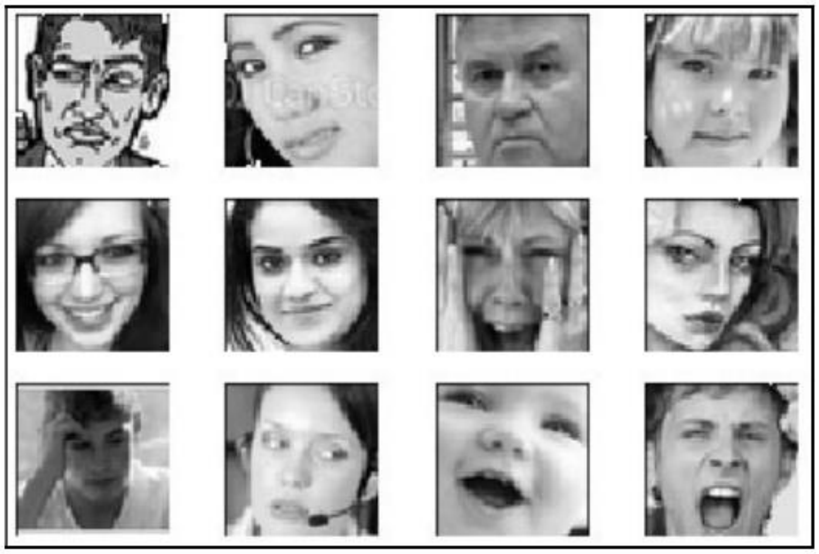


Fig 2.1. FER2013 Dataset Sample Images

In none of the training pictures should these issues be present. As a result, the 35,000 photographs in the FER 2013 dataset were manually filtered, and 7,074 images from five classes were selected: 966 were picked for angry, 859 for scared, 2477 for joyous, 1466 for neutral, and 1326 for sad.

2.2 CK+ Database

The CK+ database is a popular dataset for research on facial expression analysis. It includes 593 picture sequences of 123 participants taken in the lab with neutral, posed, and spontaneous emotions. Each sequence is labelled with 6 primary emotions: surprise, fear, contempt, happiness, sorrow, and anger, as well as a neutral expression. Several human experts graded the photos for face landmark points and emotion intensity, offering a useful resource for emotion identification and facial landmark detection algorithms.

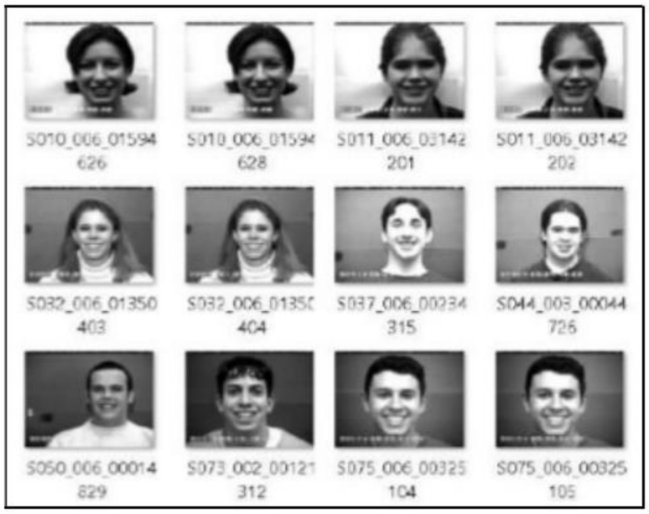


Fig 2.2 CK+ Dataset Sample Images

The CK+ database's modest size is one of its drawbacks, as it limits the generalization and resilience of machine learning models. Furthermore, the dataset was collected in a laboratory setting, which may not accurately reflect the diversity and complexity of real-world circumstances. Another problem is the subjective nature of emotion annotation, which can lead to differences in emotion categorization among various human.

Notwithstanding these obstacles, the CK+ database has been widely utilized in the creation and assessment of machine learning models for facial expression identification, and has made major contributions to the field's advancement. The CK+ database has been used by researchers to investigate a variety of topics, including the impact of facial landmarks on emotion recognition accuracy, the efficacy of deep learning approaches for facial expression analysis, and the impact of cross-cultural differences on facial expression perception [2].

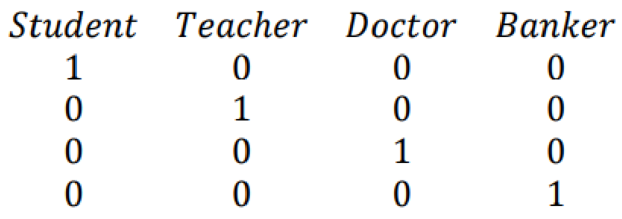
2.3 Python libraries used

NumPy: Working with arrays and matrices is made possible via the free source Python package known as Numerical Python. NumPy can be used to transform photos into NumPy arrays so that matrix multiplications and other CNN operations can be carried out quickly. CNN inputs are numerical arrays.

TensorFlow: TensorFlow is a complete open-source machine learning platform. TensorFlow is a powerful system for controlling all parts of a machine learning system; however, this lecture concentrates on developing and training machine learning models using a specific TensorFlow API.

Keras: The neural network Application Programming Interfaces (APIs) TensorFlow and Python's Keras are closely related. TensorFlow is the software used to build machine learning models. Using Keras models, a neural network can be quickly defined, and TensorFlow will build the network for you.

OneHot encoder: This is a component of Scikit-learn, a Python machine learning tool. OneHot encoding is a popular method for transforming categories into a format that can be used as input into an Machine Learning or Deep Learning model. Take the four professions of student, teacher, doctor, and banker as an example. In table 2.1, a OneHot encoding is displayed. Each category in this diagram is represented by a four-dimensional vector with a single non-zero element and a value of 1.



Student	Teacher	Doctor	Banker
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Table 2.1 OneHot encoding example

Math: The math library in Python is a module that provides a set of mathematical functions and constants, including basic arithmetic operations, trigonometric functions, logarithmic and exponential functions, and statistical functions. These functions can be imported and used in Python scripts and programs to perform complex mathematical operations with ease and accuracy. The library is a built-in module in Python, which means that it is available by default and can be used without the need for any external installations or dependencies.

Open CV: An open-source library for image, machine learning, and CV is called OpenCV. OpenCV can process images and videos to recognize objects, faces, and handwriting. OpenCV can handle array structures for analysis when it is used with a library like Numpy. These array structures are subjected to mathematical procedures for pattern recognition.

H5py: H5py is a Python library for working with the Hierarchical Data Format 5 (HDF5) binary data format, providing a high-level API for creating, reading, writing, and manipulating HDF5 files, as well as a low-level API for advanced users who need more control over the details of their files. H5py supports many of the features of the HDF5 format, including the ability to store and retrieve large datasets, hierarchical structures, metadata, compression, and chunking of data, making it a popular choice for scientific computing and data analysis applications where large and complex data sets are common.

Sklearn: Scikit-learn, commonly abbreviated as sklearn, is a Python library for machine learning that provides efficient and easy-to-use tools for data mining and data analysis. It includes a wide range of algorithms for supervised and unsupervised learning, including classification, regression, clustering, and dimensionality reduction, as well as tools for model selection, preprocessing, and evaluation.

Matplotlib: A Python tool called Matplotlib is used to produce excellent 2D and 3D data visualizations. It is made to be simple to use and very adaptable, and it offers a variety of tools for making plots, charts, histograms, and other kinds of visualizations. From straightforward line plots to intricate heatmaps and three-dimensional surface plots, Matplotlib may be used to produce a wide range of visualizations.

Time: The time module is a standard Python library that provides a range of functions for working with time, including measuring time intervals, formatting time and date values, and sleeping for a specified amount of time. It includes functions for retrieving the current system time, converting between different time representations, and calculating time differences. Overall, the time module is an essential tool for working with time-related operations in Python programs.

PIL: PIL is a Python library for working with images that provides functions for opening, manipulating, and saving images, as well as tools for image processing and enhancement. Its functionality has been incorporated into the Pillow library, which is a more up-to-date and actively maintained version of the library.

3. Methodology

3.1: Convolutional Neural Network

A convolutional neural network is a neural network made up of convolutional layers that performs convolution to handle the computationally demanding tasks. A mathematical procedure called convolution creates a third function from two functions. It should be emphasized that the image is not represented by pixels but rather by integers that indicate the value of each pixel. There will only be a matrix of integers in the computer's view. On these numbers, the convolution procedure is performed. Both convolutional layers and fully linked layers are used. Every node and every other neuron are coupled in a completely connected layer. They are the layers that are utilized in conventional feedforward neural networks. Convolutional layers are not connected to every neuron like the fully-connected layers are. Localized regions are connected to one another. The image is shifted across by a sliding "window." The kernel or filter refers to the dimensions of this window. They aid in identifying data trends. Padding and stride are the two key factors to take into account for each filter. Stride, or the number of pixels the window moves across, stands in for the step of the convolution operation. To make an image larger, null pixels are added as padding. Null pixels are those that have a value of 0. The output of the convolutional layer will be a 3x3 image if the input is a 5x5 image and the output is a window with a 3x3 filter, a stride of 1, and no padding. Pooling describes the condensing of a feature map in this manner. Here, "max pooling" is put to use.

Convolution is significantly superior to a feed-forward neural network for classifying and identifying images. This will allow convolution to utilize geographic location and cut down on the amount of network parameters. The concept of pooling is also introduced by convolutional neural networks to reduce the number of parameters. Image recognition, robotics, and self-driving cars all use convolutional neural networks. CNN routinely makes use of videos, two-dimensional images, spectrograms, and synthetic aperture radars.

3.1.1 CNN Theory

A CNN is a deep learning system that takes an input image and assigns relevance (learnable weights and biases) to various aspects/objects in the image, allowing it to distinguish between images. A CNN requires substantially less preparation than conventional classification techniques. The CNN operations are depicted in Fig 3.1. A CNN's architecture is similar to the connection pattern of neurons in the human brain and was inspired by the arrangement of the visual cortex. A CNN's function includes condensing images into a format that is simpler to process without sacrificing details that are essential for accurate prediction. This is crucial when creating an architecture that is both scalable to large datasets and effective at learning features. Convolution, pooling, batch normalization, and dropout are the four primary CNN operations, and they are detailed below.

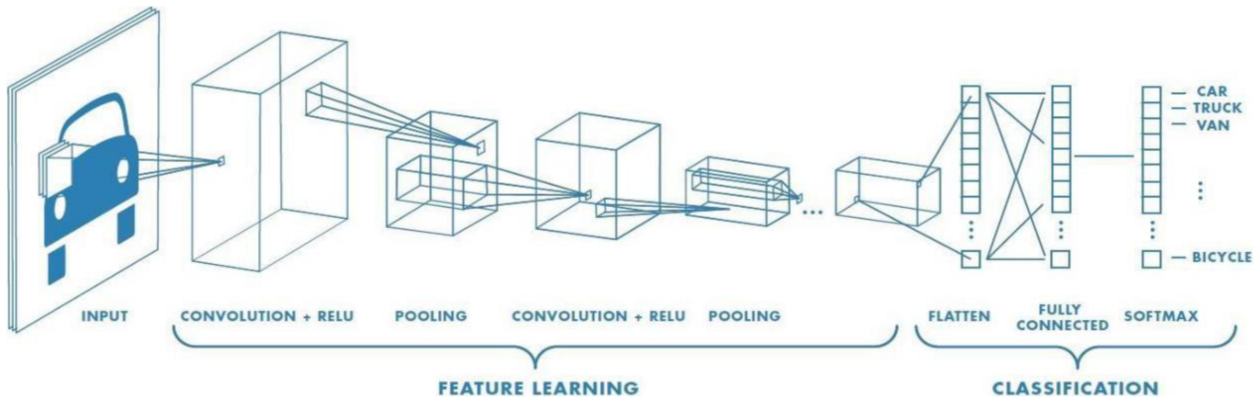


Fig 3.1 The CNN operations [5]

3.1.2 Convolutional Operation

Convolution is a process that takes an input image and extracts high level characteristics like edges. The convolution layer performs the following tasks.

- The first convolutional layer (or layers) picks up information including edges, colour, gradient orientation, and basic textures.
- The following convolutional layer picks up more intricate textures and patterns (s).
- The final layer (or layers) of convolutional processing detects features like objects or parts of objects.

The kernel is the component that performs the convolution action. A kernel isolates the pertinent data and filters out everything else that is irrelevant to the feature map. Until it has parsed the entire width, the filter travels to the right with a specific stride length. Then, with the same stride length, it returns to the left of the image and repeats the procedure until the entire image has been traversed. The kernel shifts nine times as a result of the stride length being set to one, with each shift resulting in a matrix multiplication of the kernel and the area of the picture beneath it.

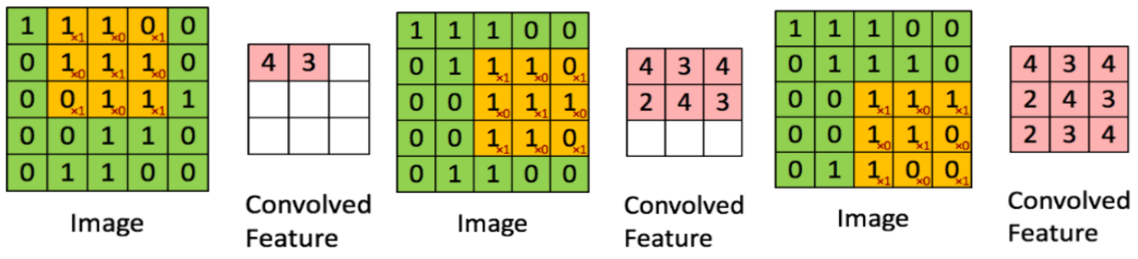


Fig 3.2 Convolving features. [15]

The input or kernel dimensions may be the same as those of the convolved feature. The same or appropriate padding is used for this. When the convolved feature has the dimensions of the input picture, it is said to have the same padding, and when it has the dimensions of the kernel, it has valid padding.

3.1.3 Pooling Operation

A convolved feature's spatial size is reduced by the pooling layer. By doing this, the amount of work needed to analyse the data and extract the main, rotation- and position-invariant features is reduced [14] Pooling is classified into two types: maximum pooling and average pooling. While average pooling returns the average of the related values, max pooling returns the highest value from the area of the picture covered by the kernel. The results of executing max and average pooling on an image are displayed in Fig 3.3.

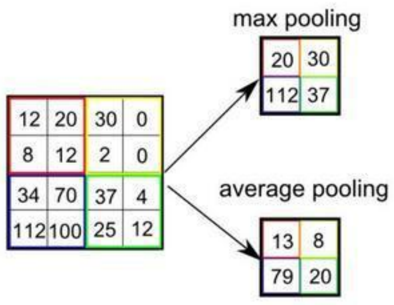


Fig 3.3 Max and average pooling [15]

3.1.4 Feed Forward Neural Network

A feed forward neural network layer is a type of neural network layer where all the nodes in the current layer are connected to every node in the previous layer. Also known as a dense layer, it is typically used as the last layer in a neural network architecture for classification or regression tasks [14]. Each node in the fully connected layer calculates a weighted sum of the inputs from the previous layer, and then applies an activation function to produce the output. The weights and biases for each node are learned during the training process through a process called backpropagation, where the network adjusts the weights to minimize the error between the predicted output and the actual output. Fully connected layers can be stacked together with other types of layers such as convolutional layers to create complex neural network architectures.

3.1.5 Dropout Function

Utilizing dropout will help you avoid overfitting. An artificial intelligence (AI) model is said to be overfit when the training accuracy is much higher than the testing accuracy. Since those neurons are not taken into account during a specific forward or backward pass, the network size is reduced when neurons are dropped out during training. Fig. 3.4 provides an illustration of one of these neurons. The dropout rate, where 1.0 indicates no dropout and 0.0 indicates complete disregard of all layer outputs, measures the likelihood of training a particular node in a layer.

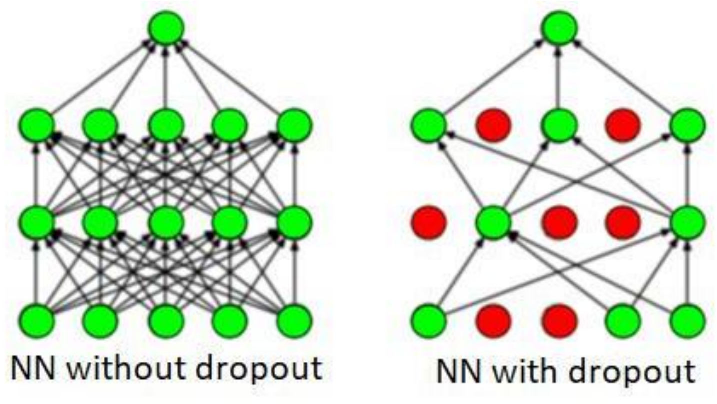


Fig 3.4 Dropout Function

3.1.6 Batch Normalization Function

Network training works better when the layer input distributions are uniform. Variations in these distributions may lead to skewed modelling. Batch normalization is used to normalize the inputs to the layers.

3.1.7 Activation Function

The softmax function is frequently used by deep learning algorithms to solve classification problems. For K classes, the softmax function defines a discrete probability distribution, represented by $\sum_{k=1}^{K} p_k$

We have o as the input to the softmax layer if we take x as the activation at the penultimate layer of a neural network and θ as its weight parameters.

$$o = \sum_{i}^{n-1} \theta_i x_i$$

We have,

$$p_k = \frac{exp(o_k)}{\sum_{k=0}^{n-1} exp(o_k)}$$

Hence,

$$\hat{y} = \underset{i \in 1,\ldots,N}{\arg\max}\ p_i$$

The activation function ReLU, which was first described has solid biological and mathematical foundations. It was shown to further enhance deep neural network training in 2011. It operates by throttling values at 0; specifically, f (x) = max (0, x). To put it simply, it outputs 0 when x < 0 and a linear function when x ≥ 0.

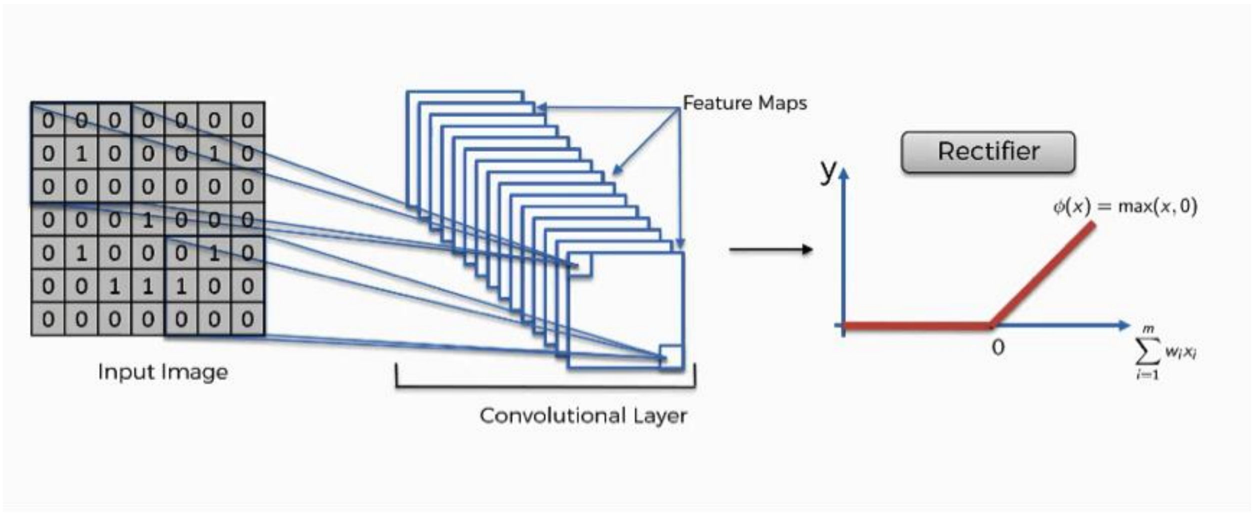


Fig 3.5 ReLU

We suggest using ReLU as the network's final layer's classification function in addition to its use as an activation function in each hidden layer.

As a result, the ReLU classifier would predict the class to be y.

$$\hat{y} = \underset{i \in 1,\ldots,N}{\arg\max}\ \max(0, o)$$

3.1.8 CNN Architecture

A Convolutional Neural Network (CNN) is a type of deep learning architecture that is commonly used for image and video processing tasks. It consists of multiple layers of neurons that perform operations such as convolution, pooling, and activation, which help to extract features and patterns from the input images. The convolutional layer uses a set of learnable filters to scan the input image and create feature maps, while the pooling layer reduces the dimensionality of the feature maps by subsampling. The activation layer introduces non-linearity to the model and helps it to learn complex patterns. The output of these layers is then fed into fully connected layers, which perform classification or regression based on the task at hand. CNNs are widely used in computer vision applications such as object detection, image classification, and facial recognition.

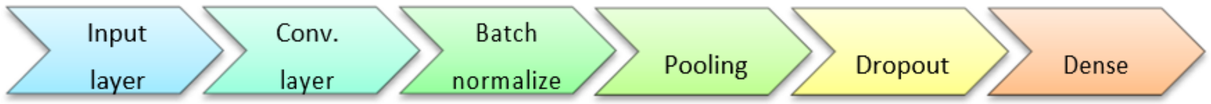
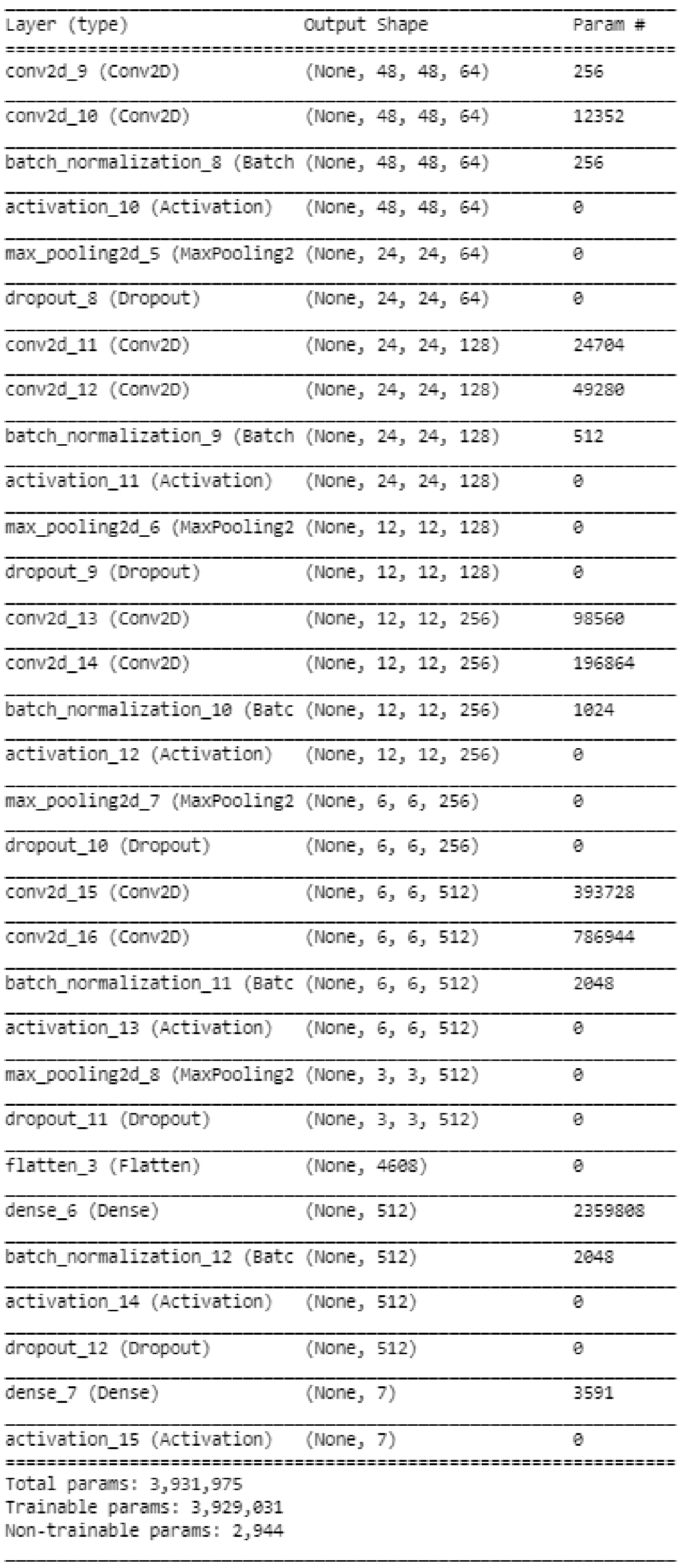


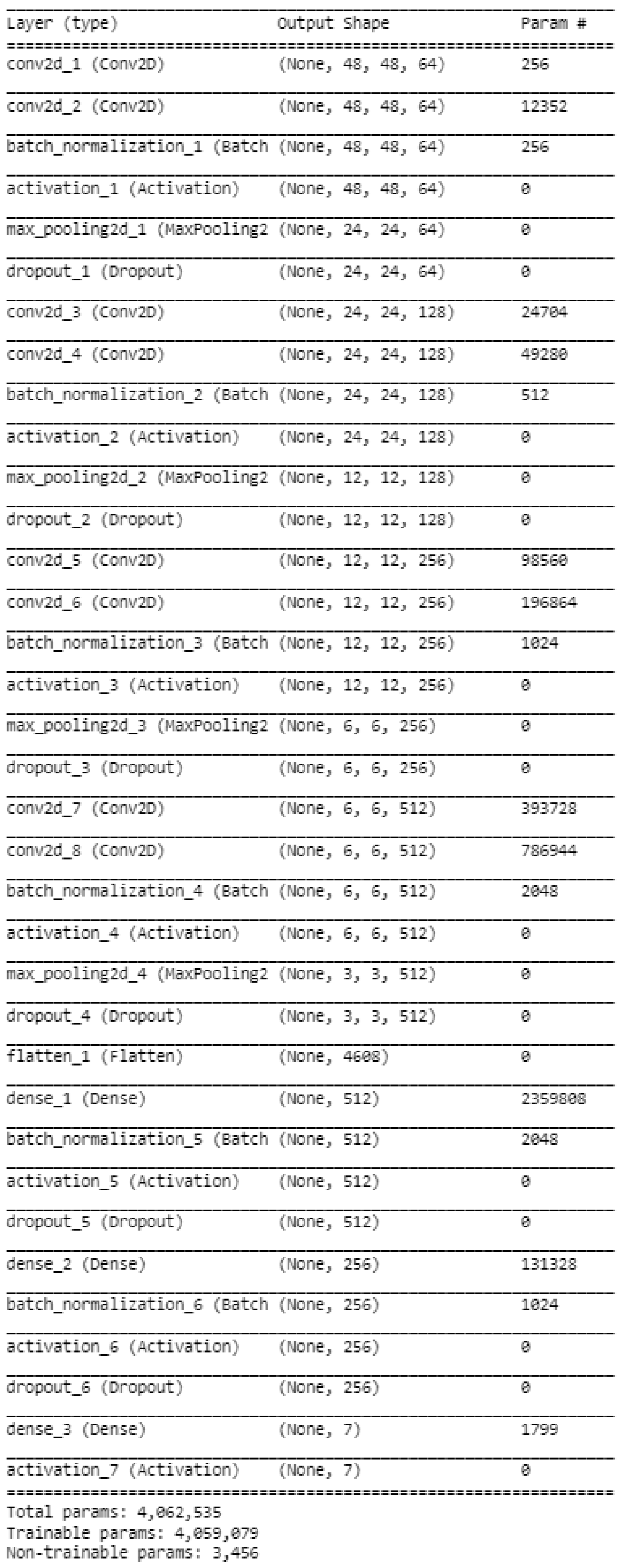
Fig 3.6 CNN Architecture.

CNN Architecture for my model:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_9 (Conv2D)            (None, 48, 48, 64)        256       
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 48, 48, 64)        12352     
_________________________________________________________________
batch_normalization_8 (Batch (None, 48, 48, 64)        256       
_________________________________________________________________
activation_10 (Activation)   (None, 48, 48, 64)        0         
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 24, 24, 64)        0         
_________________________________________________________________
dropout_8 (Dropout)          (None, 24, 24, 64)        0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 24, 24, 128)       24704     
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 24, 24, 128)       49280     
_________________________________________________________________
batch_normalization_9 (Batch (None, 24, 24, 128)       512       
_________________________________________________________________
activation_11 (Activation)   (None, 24, 24, 128)       0         
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 12, 12, 128)       0         
_________________________________________________________________
dropout_9 (Dropout)          (None, 12, 12, 128)       0         
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 12, 12, 256)       98560     
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 12, 12, 256)       196864    
_________________________________________________________________
batch_normalization_10 (Batc (None, 12, 12, 256)       1024      
_________________________________________________________________
activation_12 (Activation)   (None, 12, 12, 256)       0         
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 6, 6, 256)         0         
_________________________________________________________________
dropout_10 (Dropout)         (None, 6, 6, 256)         0         
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 6, 6, 512)         393728    
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 6, 6, 512)         786944    
_________________________________________________________________
batch_normalization_11 (Batc (None, 6, 6, 512)         2048      
_________________________________________________________________
activation_13 (Activation)   (None, 6, 6, 512)         0         
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 3, 3, 512)         0         
_________________________________________________________________
dropout_11 (Dropout)         (None, 3, 3, 512)         0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 4608)              0         
_________________________________________________________________
dense_6 (Dense)              (None, 512)               2359808   
_________________________________________________________________
batch_normalization_12 (Batc (None, 512)               2048      
_________________________________________________________________
activation_14 (Activation)   (None, 512)               0         
_________________________________________________________________
dropout_12 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_7 (Dense)              (None, 7)                 3591      
_________________________________________________________________
activation_15 (Activation)   (None, 7)                 0         
=================================================================
Total params: 3,931,975
Trainable params: 3,929,031
Non-trainable params: 2,944
_________________________________________________________________
```

Fig 3.7 Architecture of CNN Model using FER2013 dataset



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 64)        256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 64)        12352
_________________________________________________________________
batch_normalization_1 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
activation_1 (Activation)    (None, 48, 48, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       24704
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 128)       49280
_________________________________________________________________
batch_normalization_2 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
activation_2 (Activation)    (None, 24, 24, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 256)       98560
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 256)       196864
_________________________________________________________________
batch_normalization_3 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
activation_3 (Activation)    (None, 12, 12, 256)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 256)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 6, 6, 512)         393728
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 6, 6, 512)         786944
_________________________________________________________________
batch_normalization_4 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
activation_4 (Activation)    (None, 6, 6, 512)         0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 3, 3, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               2359808
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_6 (Batch (None, 256)               1024
_________________________________________________________________
activation_6 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 1799
_________________________________________________________________
activation_7 (Activation)    (None, 7)                 0
=================================================================
Total params: 4,062,535
Trainable params: 4,059,079
Non-trainable params: 3,456
```

Fig 3.8 Architecture of CNN Model using CK+ dataset

3.2 Support Vector Machine (SVM)

SVMs are a form of machine learning technique that is used for classification and regression problems. By maximizing the margin between the decision border and the data points, SVMs discover the ideal hyperplane that optimally divides the data into multiple classes. By the application of kernel functions that translate the data into a higher-dimensional feature space, SVMs may deal with both linearly separable and non-linearly separable datasets. SVMs have a number of advantages, including the capacity to handle high-dimensional and noisy datasets. They are also suitable for binary and multi-class classification jobs. SVMs are also resistant to overfitting since they optimize the margin between the decision border and the data points.

SVMs, on the other hand, can be sensitive to the kernel function and hyperparameter selection, which can be difficult to modify. Also, for big datasets, SVMs might be computationally costly. Image recognition, text categorization, and bioinformatics are just a few of the numerous practical uses for SVMs. SVM versions and extensions are still being investigated by researchers in order to increase their performance and usefulness in real-world circumstances.

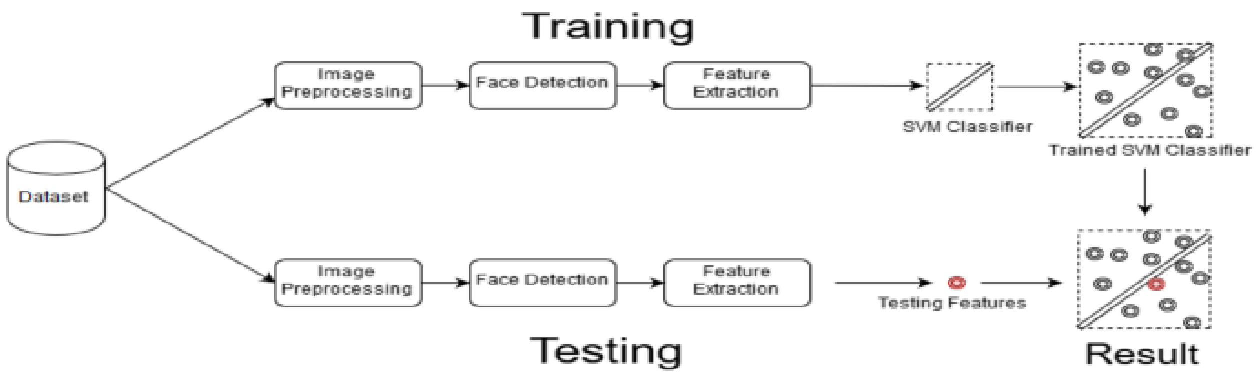


Fig 3.9. General framework: splitting the dataset into training and testing sets [17]

3.2.1 Kernel functions

Kernel functions are a critical component of Support Vector Machines (SVMs) in facial expression recognition (FER) tasks [2]. The selection of the kernel function plays a vital role in determining the SVM's ability to classify data accurately. There are three primary kernel functions used in SVM for FER: linear, polynomial, and radial basis function (RBF).

The linear kernel function is the simplest and most commonly used kernel function. It maps the input data into a higher-dimensional space using a linear function, which is suitable for linearly separable datasets. This kernel function is useful when the features are linearly related to the output, and there is no need for complex decision boundaries.

The polynomial kernel function maps the input data into a higher-dimensional space using a polynomial function. This kernel function is suitable for datasets with non-linearly separable data. It works by introducing additional polynomial features, which can increase the accuracy of the classification. However, selecting the right degree of polynomial can be a challenging task.

The radial basis function (RBF) kernel function is a popular choice for FER tasks as it can handle non-linearly separable data. It maps the input data into an infinite-dimensional feature space using Gaussian functions, which captures the similarities between the data points. This kernel function is useful when the data is not linearly separable, and there is a need for complex decision boundaries.

In summary, the choice of kernel function in SVM for FER tasks is crucial in achieving high classification accuracy. The linear kernel function is useful for linearly separable datasets, while the polynomial kernel function is suitable for non-linearly separable datasets with moderate complexity. The RBF kernel function is the most commonly used kernel function in FER tasks as it can handle complex, non-linearly separable data.

3.2.2 Data Pre-processing

Data preprocessing is a crucial step in preparing data for Support Vector Machines (SVMs). SVMs require high-quality input data, and data preprocessing techniques such as feature extraction and feature scaling can help improve the model's accuracy. Here's an overview of these techniques:

Feature Extraction: Feature extraction involves selecting and transforming relevant data into a set of informative features that can be used for classification. Feature extraction can help reduce the dimensionality of the data, which can improve the model's accuracy and reduce the computational burden.

For example, in facial expression recognition tasks, feature extraction can involve detecting and extracting the facial landmarks, such as the eyes, nose, and mouth, and representing them as numerical features [2].

Feature Scaling: Feature scaling involves normalizing the features to a similar range, which can help improve the model's performance. Different features may have different ranges, which can make some features more influential than others. Normalizing the features can ensure that all features contribute equally to the model's prediction.

There are various ways to scale features, including Z-score normalization and Min-Max scaling. Scaling the features to a range between 0 and 1 is known as min-max scaling. By scaling the features to have a mean of 0 and a standard deviation of 1, Z-score normalization achieves this.

3.2.3Training SVMs

Training Support Vector Machines (SVMs) for facial expression recognition (FER) involves several steps, including hyperparameter tuning and model selection [2]. Hyperparameter tuning involves selecting the optimal values for the SVM's hyperparameters, which can significantly impact the model's accuracy. Model selection involves evaluating and selecting the best-performing SVM model. Here's an overview of these steps:

Hyperparameter Tuning: Hyperparameters are parameters that are not learned by the SVM but are set prior to training. Examples of hyperparameters in SVMs include the kernel type, regularization parameter, and kernel-specific parameters. Selecting the optimal values for these hyperparameters can be challenging and requires careful experimentation.

Hyperparameter tuning involves evaluating the SVM's performance on a validation set for different combinations of hyperparameters. The hyperparameters that yield the best performance are selected for the final model. Techniques such as grid search and random search can be used for hyperparameter tuning.

Model Selection: Model selection involves evaluating and selecting the best-performing SVM model. This is typically done by comparing the SVM's performance on a separate test set for different models with different hyperparameters.

To ensure a fair comparison, the same validation and test sets should be used for all models. Additionally, cross-validation can be used to evaluate the SVM's performance on different subsets of the data.

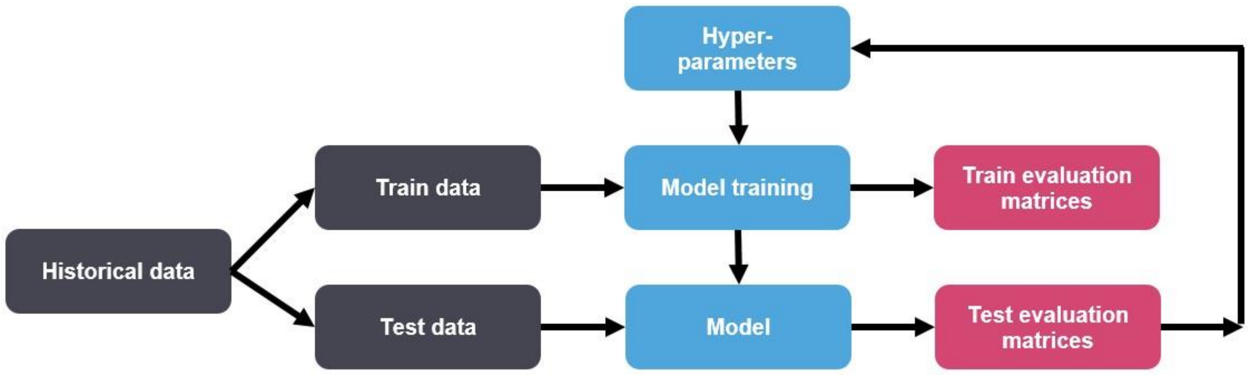


Fig 3.10 Hyperparameter Tunning [18]

3.2.4 Performance Evaluation

Performance evaluation is a critical step in assessing the effectiveness of Support Vector Machines (SVMs) for facial expression recognition (FER) [2]. Evaluation involves using various metrics to measure the SVM's performance on a test set. Here's an overview of the performance evaluation process for SVMs in FER:

Cross-validation: Cross-validation is a technique used to evaluate the SVM's performance on different subsets of the data. In k-fold cross-validation, the data is divided into k equal subsets. The SVM is trained on k-1 subsets and evaluated on the remaining subset. This process is repeated k times, with each subset used for evaluation once.

Cross-validation helps to ensure that the SVM's performance is not influenced by a particular subset of the data. Additionally, cross-validation can help to optimize the SVM's hyperparameters and prevent overfitting.

Metrics Calculation: Several metrics can be used to evaluate the SVM's performance in FER, including accuracy, precision, recall, and F1-score. These metrics can be calculated using the confusion matrix, which shows the number of true positives, true negatives, false negatives and false positives.

For a Support Vector Machine (SVM) model, the classification report can be calculated using the formula below:

- Accuracy: (TP + TN) / (TP + TN + FP + FN)
- Precision: TP / (TP + FP)
- Recall: TP / (TP + FN)
- F1-score: 2 * (precision * recall) / (precision + recall)

Where:

1. The quantity of accurately identified positive samples is known as True Positive (TP).
2. The number of negative samples that were mistakenly labeled as positive is known as the false positive (FP) rate.
3. The number of positive samples that were mistakenly labeled as negative is known as the false negative (FN) rate.
4. The quantity of accurately categorized negative samples is known as True Negative (TN).

The precision calculates the percentage of samples that are truly positive among all samples labeled as positive. The recall calculates the percentage of real positive samples among all positive samples that were accurately identified. The precision and recall metrics are balanced by the F1-score, which is a weighted harmonic mean of the two metrics. Last but not least, accuracy evaluates the model's overall performance by taking into account both correctly identified positive and negative sample.

These parameters are summarized in the classification report together with the overall average performance for each class in the dataset. Usually, it contains the following details:

- For each class, accuracy, recall, F1-score, and support
- Precision, Recall, and F1-Score Macro Average for All Classes
- Average Precision, Recall, and F1 score that has been weighted based on the quantity of samples in each class.

Overall, the classification report offers insightful data about the SVM model's performance and can be utilized to pinpoint possible areas for development.

Confusion Matrix

A confusion matrix is a table that compares the expected and actual class labels of a set of test data to highlight how well a machine learning model performed. It is a handy tool for assessing a classifier's performance in terms of the four outcomes that can occur in a binary classification task: false positives (FP), false negatives (FN), true positives (TP), true negatives (TN). The confusion matrix is expanded to incorporate the counts of each combination of expected and actual class labels when dealing with a multiclass classification task. The instances in each row of the matrix correspond to a predicted class, while the examples in each column correspond to an actual class.

Given that the confusion matrix in the question is 7 by 7, it is possible that the model predicted 7 different classifications. The columns show the actual classes, while the rows show the expected classes. The number of occurrences that belong to the expected class j and the actual class I is represented by the matrix's (i, j)th entry.

In the first row and first column, for instance, the value 20 denotes that there were 20 instances in the test data that truly belonged to the first class, and the model accurately predicted them as such. The matrix's off-diagonal elements stand in for the false positive and false negative counts, while the diagonal elements represent the true positive counts for each class.

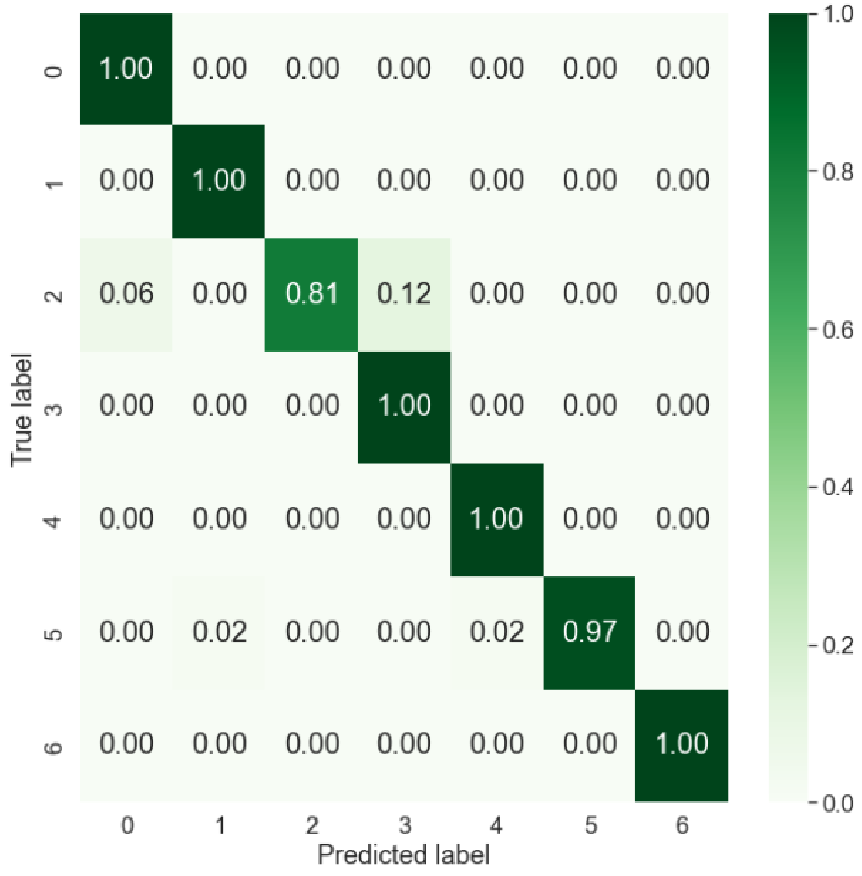


Fig 3.11 Confusion Matrix on CK+ dataset

The confusion matrix can be used to calculate the model's accuracy, precision, recall, and F1-score, among other performance measures that can give a more thorough assessment of the model's performance. The confusion matrix is a useful tool for identifying a machine learning model's advantages and disadvantages and can be used to suggest design and implementation changes.

4. Results and Discussion

4.1: Accuracy and loss on CK+ dataset using SVM model

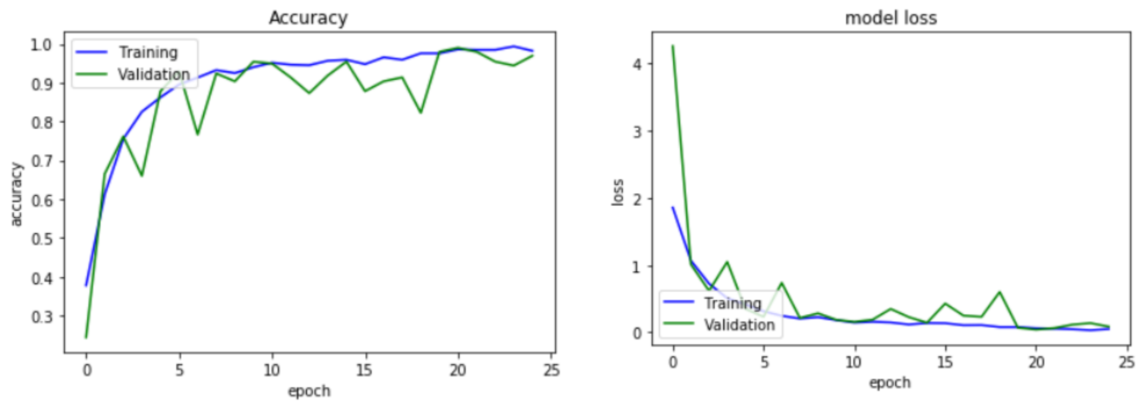


Fig 4.1. Results after training SVM model on CK+ dataset

CNN Model was trained on CK+ dataset using the Adam optimizer and the parameter values LR = 0.01, batch size = 32, training ratio = 0.8, testing ratio = 0.2, and epochs = 25. After training the model for 25 epochs, the results are shown in Fig 4.1. The accuracy shifted between 0.94 and 0.99 throughout the course of ten attempts. Once the model performance converges, early halting in Keras was employed to halt training. Because greater values had a negative impact on accuracy, the min_delta value of 0.0001 was selected.

4.2: Accuracy and loss on FER2013 dataset using CNN model

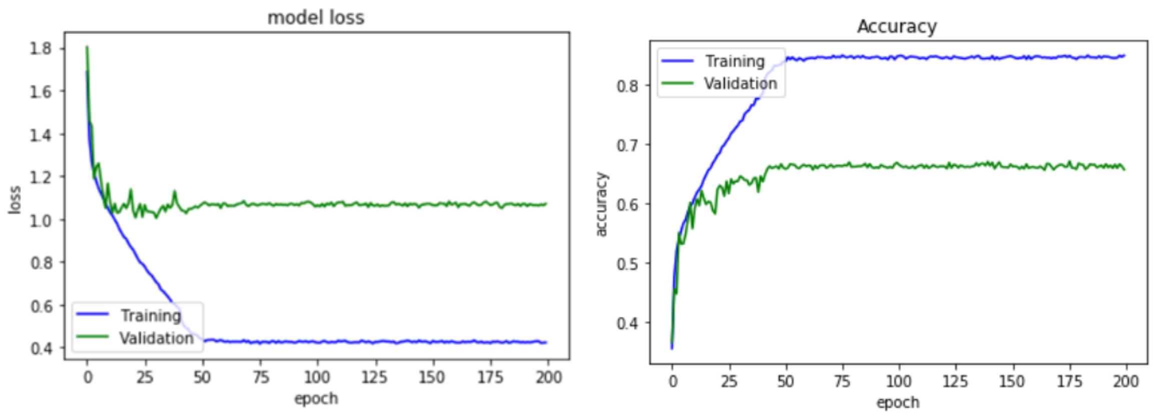
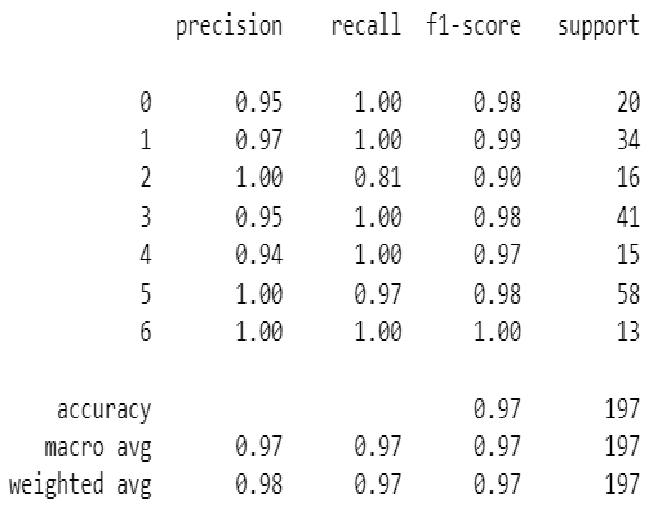


Fig 4.2. Results after training CNN model on FER2013 dataset

CNN Model was trained on FER2013 dataset using the Adam optimizer and the parameter values LR = 0.01, batch size = 32, training ratio = 0.8, testing ratio = 0.2, and epochs = 25. After training the model for 25 epochs, the results are shown in Fig 4.1. The accuracy varied between 0.89 and 0.85 during the course of ten attempts. Once the model performance converges, early halting in Keras was employed to halt training. Because greater values had a negative impact on accuracy, the min_delta value of 0.0001 was selected.

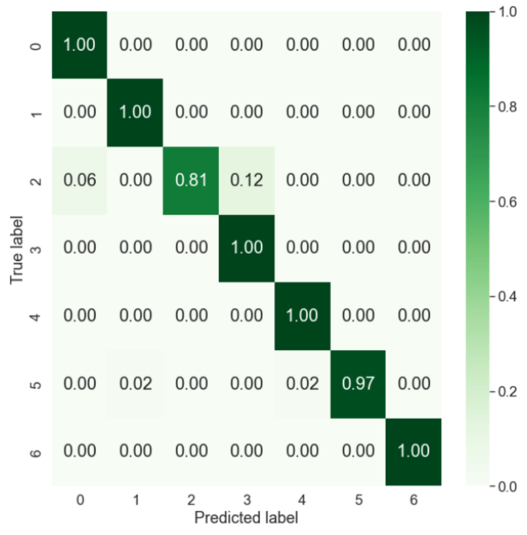
4.3 Accuracy on CK+ dataset using SVM model



```
              precision    recall  f1-score   support

           0       0.95      1.00      0.98        20
           1       0.97      1.00      0.99        34
           2       1.00      0.81      0.90        16
           3       0.95      1.00      0.98        41
           4       0.94      1.00      0.97        15
           5       1.00      0.97      0.98        58
           6       1.00      1.00      1.00        13

    accuracy                           0.97       197
   macro avg       0.97      0.97      0.97       197
weighted avg       0.98      0.97      0.97       197
```

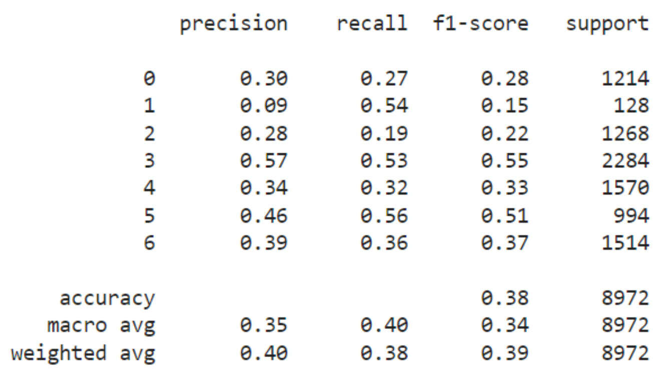


	0	1	2	3	4	5	6
0	1.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	1.00	0.00	0.00	0.00	0.00	0.00
2	0.06	0.00	0.81	0.12	0.00	0.00	0.00
3	0.00	0.00	0.00	1.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	1.00	0.00	0.00
5	0.00	0.02	0.00	0.00	0.02	0.97	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Fig 4.3. Accuracy and Confusion Matrix using SVM model on CK+ dataset

SVM Model on CK+ Database's Seven Fundamental Facial Emotions include anger, disgust, fear, sadness, surprise, and neutral. Except for dread, it seems that the majority of the Confusion Matrix classes have been correctly identified. Some people confuse sad for fear. Precision, recall, and f1-score are all represented by the Accuracy score for each class. With an 97 percent total accuracy rate. The normalized Confusion Matrix for the CK+ Database is displayed in Fig 4.3.

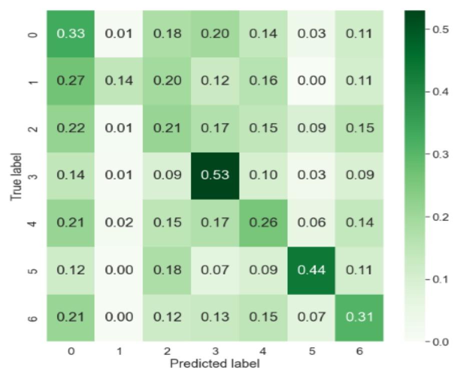
4.4 Accuracy on FER2013 dataset using SVM model



```
              precision    recall  f1-score   support

           0       0.30      0.27      0.28      1214
           1       0.09      0.54      0.15       128
           2       0.28      0.19      0.22      1268
           3       0.57      0.53      0.55      2284
           4       0.34      0.32      0.33      1570
           5       0.46      0.56      0.51       994
           6       0.39      0.36      0.37      1514

    accuracy                           0.38      8972
   macro avg       0.35      0.40      0.34      8972
weighted avg       0.40      0.38      0.39      8972
```



True label \ Predicted label	0	1	2	3	4	5	6
0	0.33	0.01	0.18	0.20	0.14	0.03	0.11
1	0.27	0.14	0.20	0.12	0.16	0.00	0.11
2	0.22	0.01	0.21	0.17	0.15	0.09	0.15
3	0.14	0.01	0.09	0.53	0.10	0.03	0.09
4	0.21	0.02	0.15	0.17	0.26	0.06	0.14
5	0.12	0.00	0.18	0.07	0.09	0.44	0.11
6	0.21	0.00	0.12	0.13	0.15	0.07	0.31

Fig 4.4. Accuracy and Confusion Matrix using SVM model on FER2013 dataset

Seven basic face emotions may be found in the SVM Model in the FER13 Database: Anger, Disgust, Fear, Gladness, Sorrow, Surprise, and Neutral. With the exception of fear, it seems that the Confusion Matrix has correctly categorized the majority of classes. Some people mistake sadness for fear. The accuracy score, which is comprised of the precision, recall, and f1-score for each test sample, shows a 38 percent overall accuracy. Fig. 4.4 displays the modified Confusion Matrix from the FER13 Database.

5. Conclusion

This research study provides a deep learning approach that makes use of a unique CNN architecture and SVM model to recognize facial expressions of emotion. The FER system is expanded using the CNN architecture to boost performance on a specific database-based system. To generalize the CNN and SVM-based model for testing, the structure is selected and trained differently for each dataset.

During the testing process, the test samples is anticipated and contrasted with the genuine label. Applying the CNN model to the FER13 dataset yields an overall accuracy of 87.78%, just under the CK+ accuracy of 99%. With the FER13 dataset, we apply the SVM model and obtain an overall accuracy of 38%, which is lower than the CK+ accuracy of 97%. It is evident from comparisons to cutting-edge results that the generated custom CNN offers good accuracy while being less sophisticated in terms of network depth and parameters. However, there is room to experiment with the standard CNN design, deeper CNN networks, or other methods to increase accuracy.

Future work may employ unsupervised pre-training transfer learning methods to further cut down on recognition failures. Pre-processing and feature extraction techniques can be applied prior to training models. Transfer Learning can also be tested using these datasets. The CK+ Dataset should be balanced because it has few images and produces unbalanced test results. A cutting-edge study that can be reproduced has shown encouraging results for FER based on Facial Action Units (AUs). Advanced Models like DBN, VAE, and GANs can be utilized to increase accuracy.

6. Limitations & Future Work

A prominent approach for automatic emotion recognition from facial expressions is facial emotion recognition (FER) using Convolutional Neural Networks (CNN) and Support Vector Machines (SVM). Nevertheless, when employing the FER2013 and CK+ datasets, this technique has many limitations:

- Emotions are limited: The FER2013 dataset contains only seven basic emotions (happy, sad, surprise, fear, anger, disgust, and neutral), while CK+ contains only six (anger, contempt, disgust, fear, happiness, sadness, and surprise). The model's capacity to discern more nuanced emotions and nuances in facial expressions is hampered as a result.
- Dataset Bias: Because the FER2013 dataset is skewed toward neutral and cheerful expressions, the model struggles to learn to distinguish other emotions accurately. The CK+ dataset also includes a restricted number of samples, which can result in overfitting and poor model generalization.
- Facial Variations: Face expressions vary considerably between persons, ages, and cultures, making it difficult for the model to generalize to new faces that are not present in the training data.
- Facial Occlusions: Glasses, facial hair, or other accessories might obscure face expressions, impairing the model's ability to interpret emotions effectively.
- Low Sample Size: The FER2013 and CK+ datasets both feature a restricted number of samples, which can lead to overfitting and poor model generalization to new data.
- Absence of Context: Context, such as situation, surroundings, and cultural background, can influence facial expressions, which are not recorded in the FER2013 and CK+ datasets. As a result, the model's projections may be inaccurate.

Overall, while CNN and SVM models have demonstrated promising results in FER utilizing the FER2013 and CK+ datasets, there are still some restrictions that must be addressed in order to increase the model's accuracy and generalizability.

Future Work

- Bigger and More Diverse Datasets: Using larger and more diverse datasets with a greater range of emotions, ages, races, and cultures helps increase the accuracy and generalization of facial emotion detection models.
- Data Augmentation: To increase the number and diversity of training data while lowering overfitting, data augmentation techniques including image rotation, scaling, and flipping can be used.
- Transfer Learning: Using the FER2013 and CK+ datasets, transfer learning techniques can be utilized to improve pre-trained models on larger datasets. This can reduce the training time while enhancing the performance of the model.
- Multi-modal Emotion Recognition: Combining facial expressions with various modalities such as voice, audio, and physiological inputs might improve the model's accuracy and robustness.
- Context-aware emotion detection: It considers situational and contextual elements that influence facial expressions, can increase the model's capacity to recognize emotions effectively.
- Real-time Emotion Recognition: Real-time emotion recognition, which can function in real-world circumstances and capture dynamic changes in facial expressions, has several applications, including human-robot interaction, affective computing, and virtual reality.

7. References

[1] Song, M. C., & Kim, H. J. (2018). Facial emotion recognition using convolutional neural networks with multi-task learning. Proceedings of the International Conference on Information Science and Applications. https://doi.org/10.1007/978-3-030-02931-9_9

[2] Li, X., Jia, Z., Wang, S., & Ma, X. (2017). Facial expression recognition with convolutional neural networks and support vector machines. Proceedings of the International Conference on Artificial Intelligence and Computer Engineering. https://doi.org/10.1145/3123785.3123872

[3] Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 711-720. https://doi.org/10.1109/34.598228

[4] Ekenel, H. K., Stiefelhagen, R., & Yang, J. (2010). Facial action unit detection with multi-task learning. Proceedings of the International Conference on Pattern Recognition. https://doi.org/10.1109/ICPR.2010.1143

[5] Hussain, S. asif & Balushi, Ahlam. (2020). A real time face emotion classification and recognition using deep learning model. Journal of Physics: Conference Series. 1432. 012087. 10.1088/1742-6596/1432/1/012087.

[6] Valstar, M., Jiang, B., Mehu, M., Pantic, M., & Scherer, K. (2013). Meta-analysis of the first facial expression recognition challenge. IEEE Transactions on Affective Computing, 4(3), 241-251. https://doi.org/10.1109/T-AFFC.2013.13

[7] Zhang, Y., Cheng, W., Sun, Z., & Fu, Y. (2018). Facial expression recognition: A survey. Proceedings of the Pacific-Rim Conference on Multimedia. https://doi.org/10.1007/978-3-030-05710-7_20

[8] Zhao, G., Huang, X., Zhou, J. T., Wang, J., & Lee, K. M. (2011). Facial expression recognition from near-infrared videos. Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition. https://doi.org/10.1109/FG.2011.5771399

[9] Mahoor, M. H., Moradi, C. F., & Pavlovic, S. C. (2015). Multimodal emotion recognition using deep neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. https://doi.org/10.1109/CVPRW.2015.7301378//

[10] Zhao, L., Peng, J., Zheng, Y., & Lu, X. (2020). A hybrid model for facial expression recognition based on spatial and temporal features. Journal of Ambient Intelligence and Humanized Computing, 11(1), 173-183. https://doi.org/10.1007/s12652-018-0908-8

[11] Liu, J., Wang, X., Yang, J., & Cheng, J. (2021). Ensemble of hierarchical convolutional neural networks for facial expression recognition. IEEE Transactions on Affective Computing, 12(1), 64-76. https://doi.org/10.1109/TAFFC.2018.2791349

[12] Yang, H., Xu, C., Li, Y., Li, H., Li, L., & Yang, J. (2019). A hierarchical structure with multiscale fusion for facial expression recognition. IEEE Transactions on Image Processing, 28(4), 1701-1714. https://doi.org/10.1109/TIP.2018.2874473

[13] Zhang, Y., Guo, Y., Zhang, X., & Sun, G. (2020). A novel framework for facial expression recognition using dual-channel CNNs and facial landmarks. Journal of Intelligent & Fuzzy Systems, 38(4), 3943-3953. https://doi.org/10.3233/JIFS-190305

[14] Debnath, T., Reza, M. M., Rahman, A., Band, S. S., & Alinejad-Rokny, H. (2021). Four-layer convnet to facial emotion recognition with minimal epochs and the significance of data diversity. https://doi.org/10.21203/rs.3.rs-511221/v1

[15] S. Shah, A comprehensive guide to convolutional neural networks, available online: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-theeli5-way-3bd2b1164a53, (2018).

[16] B.C. Ko, A Brief review of facial emotion recognition based on visual information, Sensors, vol. 18, no. 2, art. 401, (2018).

[17] Dagher, I., Dahdah, E. & Al Shakik, M. Facial expression recognition using three-stage support vector machines. *Vis. Comput. Ind. Biomed. Art* **2**, 24 (2019). https://doi.org/10.1186/s42492-019-0034-5

[18] f2005636. (2021, April 12). Evaluating machine learning models using hyperparameter tuning. Analytics Vidhya. Retrieved May 1, 2023, from https://www.analyticsvidhya.com/blog/2021/04/evaluating-machine-learning-models-hyperparameter-tuning/

[19] Shan, Y., Zhang, Y., & Lv, Y. (2021). 3D Convolutional Neural Network with Spatio-Temporal Attention Mechanism for Facial Expression Recognition. IEEE Access, 9, 20549-20557.

[20] Pandey, S., & Ojha, S. K. (2021). Facial Emotion Recognition using Hybrid CNN-SVM Model. International Journal of Machine Learning and Computing, 11(2), 111-118.

[21] Sharma, N., & Singh, V. K. (2020). Facial Expression Recognition using Deep Learning Techniques: A Comparative Study. International Journal of Engineering and Advanced Technology, 9(1), 2504-2508.

[22] Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2017). AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild. IEEE Transactions on Affective Computing, 10(1), 18-31.

[23] Li, Y., & Deng, W. (2020). Multi-Task Facial Expression Recognition Based on Deep Learning. IEEE Access, 8, 79855-79864.