

# **Determining Mood from Facial Expressions**

## **(Facial Expression Recognition)**


---

Name: Dimitar Mihaylov

Supervisor: Michael Cashmore

Registration Number: 201644327

Except where explicitly stated all the work in this report, including appendices, is my own and was carried out during my final year. It has not been submitted for assessment in any other context.

Signature:  Date: 20/03/2020

# Abstract

Human-computer Interaction (HCI) has been an increasingly popular field in the last decade. Currently, communication between man and machine is primitive, non-verbal. If computers have the ability to determine people's emotions in real-time, then would significantly improve and consequently benefit other fields such as Human-robotic Interaction (HRI).

Facial expression Recognition in computers has numerous applications in automated surveillance systems, lie detection, music/video for mood, etc. With current advancements in Artificial Intelligence, facial expression recognition in lab-controlled environment achieves almost perfect results with accuracy of 95% and above. Facial expression recognition in real time however, has accuracy levels of around 50%.

This project is developed to prove that real time facial expression recognition is possible with substantially better prediction results. The project aims to explore the best deep learning techniques for image processing tasks and train the best one to facial expression recognition. In addition, an attempt is made to embed the best deep learning model in a mobile application, which turns out unsuccessful due to GDPR concerns.

Keywords: Artificial Intelligence, Machine Learning, Deep Learning, Data Augmentation, Data Collection, Convolutional Neural Networks, Residual Neural Networks, Python, FastAI, Keras, Mobile Application, Hybrid-web Application.

# Acknowledgements

First and foremost, I would like to thank my supervisor Michael Cashmore for agreeing to supervise this project and his continuous guidance and help during the course of this project. Without him, this project would not be possible. I would also like to thank my second supervisor Yashar Moshfeghi for his assistance throughout this project.

I would also like to thank my friends, family and especially Petar Pilev for his encouragement and support in moments of despair.

I am grateful for everyone involved and everyone's support.

# Contents

<b>INTRODUCTION.....</b>	<b>1</b>
Project Description .....	1
Aims and Objectives .....	3
Requirements .....	4
Functional Requirements .....	4
Non-functional Requirements.....	4
Outcome Overview .....	4
Report Overview .....	5
<b>BACKGROUND AND RELATED RESEARCH.....</b>	<b>5</b>
Pre-processing .....	5
Feature Extraction .....	6
Image Classification .....	9
<b>SYSTEM DESIGN .....</b>	<b>13</b>
Background and Fundamental Concepts of Artificial Intelligence .....	13
Design Methodology.....	14
<b>DETAILED DESIGN AND IMPLEMENTATION.....</b>	<b>17</b>
Development Technologies.....	17
Python.....	17
Ionic .....	17
Angular .....	17
Apache Cordova .....	17
Capacitor.....	17
JavaScript.....	18
Development Libraries.....	18
Fastai.....	18
Pytorch.....	18
Keras .....	18
Imblearn.....	18
Scikit-learn .....	18
OpenCV .....	18
Development Environment.....	19
Jupyter .....	19

Google Cloud Platform .....	19
IntelliJ IDEA .....	19
Android Studio .....	19
Third-party Development Tools .....	19
Git .....	19
Databases .....	20
KDEF and AKDEF.....	20
CK/CK+ .....	20
JAFPE .....	20
Oulu-CASIA NIR&VIS .....	20
TFEID .....	21
Yale .....	21
Facial Expression Recognition Models Development .....	21
Data Collection.....	21
Image Pre-processing .....	23
Model Training.....	33
Mobile Application Implementation Details .....	37
User Design .....	38
Features of application .....	38
<b>VERIFICATION AND VALIDATION.....</b>	<b>40</b>
<b>RESULTS AND EVALUATION .....</b>	<b>41</b>
Detailed Outcome Overview .....	41
Detailed Evaluation.....	46
<b>SUMMARY AND CONCLUSIONS .....</b>	<b>48</b>
Summary.....	48
Future Work.....	49
<b>APPENDIX A .....</b>	<b>50</b>
Convolutional Neural Network (CNN; ConvNet) Architecture .....	50
Vanishing Gradient Problem .....	51
Residual Neural Network (ResNet) Architecture .....	51
How to read test images .....	52
<b>APPENDIX B .....</b>	<b>52</b>
Test Procedure A.....	52
Test Procedure B.....	55
<b>APPENDIX C.....</b>	<b>60</b>
<b>REFERENCES.....</b>	<b>62</b>



# INTRODUCTION

## Project Description

Nowadays, technology is an everyday part of people's lives and plays a huge role in our development as a human race. Humans spend the majority of their time interacting with various devices such as computers, smartphones, etc. Research shows, most people's work involves working on a computer. However, when compared with human communication, interaction with currently available software interfaces is non-verbal and primitive. If computers have the ability to determine human emotions through facial expression recognition, then human-computer interaction (HCI) would be significantly improved [1]. The communication between a man and machine is becoming one of the fastest developing areas and advancements there would consequently benefit human-robotic interaction (HRI) and many other fields.

Facial expressions play a significant role in human communication [1-5]. Studies have shown human emotions expressed through facial expressions contribute 55% of emotional expression as opposed to 35% for the vocal part and only 7% for the verbal part [2]. This would imply that facial expressions play the major part in communication between people. Before diving in it's essential to make the distinction between facial expression recognition and emotion recognition. Humans express emotions in multiple ways through facial expressions, emotion of voice, body language, whereas facial expression recognition is solely based on visual information which will be the focus of this project [2].

Humans have the ability to interpret facial expressions with little effort. Attempting to develop an automated system that will have that ability is a rather difficult task that presents a few challenges. In order to achieve humanlike results a system must be able to detect an image segment as a face, extract all the information regarding the facial expression that the face is trying to express and finally classify the expressions in categories of emotion [2-5]. With the current advancements in technology and the various existing techniques available for solving this problem, a system that can accurately guess human emotions in laboratory conditions has been developed with accuracy around 97%, however when tested in an unconstrained real-time environment it doesn't perform as well with accuracy as low as 50% [4]. Analysis of facial expressions can be incredibly challenging in a real-time environment due to constantly occurring small transient changes. The main three challenges that arise in an unconstrained real-world environment are illumination variation, head pose and subject dependence [4]. Different illumination levels can directly affect the accuracy of face feature extraction. In a controlled environment under laboratory conditions the head pose of a subject is usually frontal. Yet, in a real-world environment due to a subject's transient movements of the head, sometimes a frontal

view might not be available thus presenting a great challenge. Subject dependence has to do with the fact a Facial Expression Recognition (FER) system can only recognize pre-trained faces. To overcome this challenge, a large dataset and a reliable classifier would be required [4].

As mentioned above, developing a FER system has three main parts: pre-processing, feature extraction and image classification [2-5]. Pre-processing is a process which can substantially increase the accuracy of a FER system by applying different cropping and scaling techniques to the images and preparing the data for feature extraction [5]. Feature extraction is a process which dimensionally reduces initially large, raw data to a more manageable dataset by combining different features, while accurately and completely representing the original dataset [6, 7]. After the necessary features are extracted, images can then be classified by emotions at the last stage [5].

The possibilities for application of FER in various areas are endless. As mentioned above, the main areas which could be drastically improved by an effective FER system are HCI and in particular HRI. Having the ability to determine people's emotions would enable an HRI systems to simulate a more natural and friendly interaction with humans, thus improving almost every industry and occupation where computers are used. For example, they can be used in a medical care to detect various mental states through facial expression analysis [8, 9]. By being able to distinguish between the different healthy emotional states such as happiness, satisfaction, and unhealthy emotional states (anger, sadness, frustration) automated systems can improve quality of life. In addition, FER could drastically help mentally ill patients by exploring their behavioural patterns. Researchers [10] and [11] have successfully shown that by investigating facial expressions of patients, they can identify emotional conflicts and as a result, mental disorders such as anxiety or autism can be diagnosed. Furthermore, study has shown people are more likely to trust a machine rather than another person [12]. This opens up a great application of FER in automated counselling systems. Fatigue detection is another area which could significantly benefit by a FER system [13]. According to NHTSA organization statistics, the main factor for road accidents is driver fatigue [13]. With the ability to read human emotions vehicles would be able to prevent the that alerting the drivers. In a similar way, FER can help prevent accidents at the workplace by monitoring employees fatigue levels especially for machine operators. Another area where FER could be applied is teaching [14] - currently automated tutoring systems are becoming increasingly popular allowing anyone with computer access to learn from the comfort of their own home. An automated tutoring system with FER would be able to adapt individually to every student and determine how well they're responding to the material. This would allow the system to make decisions and adjustments, which would make the process seem less artificial, more pleasant, thus



making it a more effective learning process on an individual level. Computer graphics is an area which is already largely benefitting from efficient FER systems. The modelling of a human face by precisely parameterizing the geometry of the face and muscle motions is done by FER systems [15]. Newer technologies such as Augmented Reality (AR) and Virtual Reality (VR) also apply effective FER systems to achieve a more natural, effortless communication with humans. Other areas where FER can be applied are automated surveillance systems [48], behaviour prediction [47], lie detection [28], and music/lighting for mood, etc.

Research has shown that under lab conditions and normalized dataset it is possible to train a nearly perfect deep learning model. However, due to a number of challenges that will be discussed in detail later on, performance of currently developed techniques in real-time is very poor.

### **Aims and Objectives**

The aim of this project is to show that with current existing techniques in Artificial Intelligence and Deep Learning in particular it is possible to build and train a deep learning model that performs facial expression recognition in real-time with low latency and better performance than what it has been achieved before. Also, to show, test and evaluate the model a system must be developed where the model should be embedded. Originally, that system is supposed to be a mobile application, however after failing to achieve real-time FER in a mobile environment, a prototype in python is developed that can stream video through a webcam in real-time. This prototype of a system extracts the faces from every video frame then analyses expressions and presents the results back to the user. The objectives based on thorough research into different ways to develop such system and initial project description are as follows:

- Choose existing deep learning techniques to explore
- Choose a platform for developing the system
- Compare the performance of techniques chosen
- Implement the best technique in the system and present the results to the user in an unambiguous and user-friendly way
- Think of a way to make the system self-improving
- Choose which emotions to use for facial expression recognition

## Requirements

The full list of functional and non-functional requirements is formed from project description and research on different approaches to achieve FER in real-time.

### Functional Requirements

- Stream webcam video in real-time
- Face detection of all present faces on every video frame
- FER analysis of all detected faces
- FER analysis must support at minimum the 6 most common emotions: anger, fear, happiness, neutral, sadness, surprise
- Present results from FER analysis to user

### Non-functional Requirements

- Low latency
- Data collection, balancing augmentation for model training
- Build and train different AI techniques for FER
- Performance comparison of all explored techniques

## Outcome Overview

During this project, most popular deep learning techniques are applied to the problem of facial expression recognition in real time. All techniques were trained to recognize the 7 most common emotions: anger, disgust, fear, happiness, neutral, surprise and sadness. All present faces on the screen are detected and analysed. After evaluation and performance comparison, the best technique is embedded in a prototype system for facial expression recognition in real time. The model for FER analysis performs impressively well on most emotions, except fear and sometimes mistakes similar expressions such as neutral and sadness. Test procedures explored in this project are passed with accuracy of 70% and above. An attempt is made to develop a hybrid app with the Ionic framework that works as a native application on both android and iOS devices. Unfortunately, due to privacy and GDPR concerns streaming of webcam is not possible in native devices. An unsuccessful attempt is made to develop a plugin which will connect with native-camera-plugin in android in order to overcome GDPR concerns. As a result, the facial expression recognition in real-time functionality only works in a web environment. This is why to achieve the main aim of this project FER in real time a prototype in python is developed instead that meets all functional and non-functional requirements.

### **Report Overview**

The following sections of this report follows the development methodology applied in this project. The next chapter gives a summary of background work and related research, followed by discussions regarding the choices made about the system design for every step of the methodology. This chapter includes background overview of fundamental concepts and definitions in artificial intelligence, which is strongly recommended for people with no deep learning background. In addition, appendix A contains more information with regards to model architectures used and challenges along the way. The next chapter covers the implementation of the system design and ways to achieve the desired results with technologies choices. Results, validation and evaluation are detailed next, followed by independent analysis and reflection of the overall system and possible future work.

## **BACKGROUND AND RELATED RESEARCH**

As mentioned above, HCI is an increasingly growing field that attracts a lot of people. Currently existing techniques achieve very high results in lab-controlled environment with accuracy above 95%. However, due to the challenges discussed earlier, in an unconstrained real-world environment, FER systems achieve around 50% accuracy. One of the most studied topics in computer vision is face detection and more recently facial expression recognition. As a result, a large number of algorithms have been developed to overcome various challenges such as illumination variation, pose, occlusion, and others. The following sections describe the most popular and effective techniques used today along with performance comparison for the three stages in a FER system: pre-processing, feature extraction and image classification.

### **Pre-processing**

In image processing it's quite common nowadays to work with large datasets, meaning the majority of the images will have different graphical properties. The model trained on such data will most likely suffer in performance [16]. In order to overcome that challenge different pre-processing techniques are applied to the data. Pre-processing is a process which prepares the data for feature extraction and can majorly improve performance of FER [17]. Image pre-processing in this case involves different cropping, scaling, contrast adjustment techniques to improve the data [5]. In almost all cases for an accurate FER system, various pre-processing techniques must be applied to the data as large disparities between image parameters such as illumination levels, contrast, size can drastically decrease the performance, hence why pre-processing is a key stage. The most popular pre-processing methods

implemented for this part are Normalization [18], Localization [19, 20], Region of Interest (ROI) [21]. Normalization is the most popular technique, which can be used for reduction of illumination variation in different images and reduction of face image variation, thus providing more clarity to the images. Moreover, normalization is used for extraction of different features such as eyes, mouth, which helps develop a more robust FER system when it comes to personality differences. In addition to normalization, localization is another pre-processing method that can be applied [19,20]. Localization is used for detecting faces within an image and isolating them in face images. Its main purpose is spot the location and size of the face image. ROI are samples of data, isolated for a particular purpose. In image processing, ROI segmentation is an important method for identifying the different parts of a face such as mouth, nose, eyebrows [21, 26]. Another less popular image processing technique is Histogram Equalization [19,20], which is used for adjusting the contrast in images. Histogram the graphical representation of the colour value distribution of an image [22]. This method uses the image's histogram to spread out the most frequent intensity values. This means local areas with low contrast gain a higher contrast. This technique is useful for images with similar tonal distribution in the background and foreground such as x-ray images [22]. In FER pre-processing, ROI is the most popular technique used as it precisely detects all the different face parts that people use to express emotions [21,26].

### **Feature Extraction**

Feature Extraction is the most important, second step in designing a FER system. Large datasets, which are increasingly common, usually contain thousands of features furthermore if the number of features is greater than the number of observations in the dataset, it is highly likely the model would suffer from overfitting [6,7,24]. Overfitting refers to a dataset that has been modelled more than necessary to the point where it actually degrades the performance of the machine learning model. As a result, the model would only learn to perform well on the dataset that it's been trained on. To overcome this obstacle, it's vital to apply regularization or dimensionality reduction techniques –feature extraction [24]. Other than overfitting risk reduction, feature extraction is especially useful where the images are large in size and a more compact feature representation, stripped of any redundant information, is required for further processing. This also necessitates less computing power and respectively improves training times [24]. During this stage, the graphical data of an image is depicted as implicit numerical data describing the texture properties [5], which is then given as input to the classification algorithm. Feature extraction techniques are divided into five types: texture-feature based, edge based, global and local feature-based, patch-based and geometric shape based [5].

With texture feature-based algorithms [20, 22, 33, 34, 35, 37] all features are formed from properties defining the texture of an image. Textures are one of the most important characteristics of an image used to classify and recognize objects and find similarities between images. Scale Invariant Feature Transform (SIFT) [22, 34] has proved to be a very powerful technique for object detection/recognition, however, SIFT might not be optimal for analysing face images [23]. Keypoints-Preserving-SIFT (KPSIFT) includes all the initial key points as features. Partial-Descriptor-SIFT (PDSIFT), includes all key points detected at large scale and uses a partial descriptor for identifying face boundaries. Both prove to be more efficient than the original SIFT [23]. Gabor filters are linear orientation-sensitive filters used for edge and texture analysis that can extract local features in frequency and spatial domain [25-27]. Local Binary Patterns (LBP) is a simple, efficient and robust local descriptor that has proven to do well in various domains such as texture analysis, facial expression recognition, and facial recognition [29]. LBP represents pixels as binary numbers by thresholding the neighbouring pixels and its most important property is its robustness to monotonic gray-scale changes such as illumination variations [30]. Combined with its computational simplicity, LBP is one of the more popular feature extraction techniques used in FER systems [20,22]. For multi resolution approaches, LBP is combined with Three Orthogonal Planes (TOP) method [31]. Weighted Project Based LBP (WPBLBP) [35] is an extended LBP extraction that's formed from instructional domains for which the LBP is extracted. After that, depending on the importance of these instructive regions, the extracted features are weighted [5]. Gaussian Laguerre (GL) wavelets have powerful frequency extraction capabilities for extracting features of facial expressions [16]. In comparison, GL uses a single filter instead rather than multiple ones with Gabor filters [5], which means Gabor Filters require significantly more computational power. In addition, Vertical Time Backward (VTB) method takes out the shape related attributes of facial components. This makes it really effective on spatiotemporal planes. Spatiotemporal derivatives are usually contained in images produced by catadioptric sensors, which contain a significant amount of radial distortion and variation in inherent scale [32]. Weber Local Descriptor (WLD) is another texture-based method for extracting features, which consists of differential excitation component and orientation component, that contains abundant local information [33]. In most cases, WLD uses Supervised Descent Method (SDM) to estimate the distance between various components of the face [5]. WLD performs better than LBP while still being as computationally efficient as LBP. As mentioned, SIFT is a sparse descriptor, whereas WLD is a dense descriptor computed for every pixel and depends on the magnitude of the centre pixel's intensity [34]. Lastly, Discrete Contourlet Transform (DCT) [36] method is a combination of a multi-scaled Laplacian pyramid and multi-directional filter banks. Compared to one-dimensional transforms such as Fourier and wavelet DCT is a two-dimensional feature extraction method that can capture geometrical structures of an image that are key to visual

information. Using multidirectional filter banks with DCT multi-resolution and directional image representation contour segments, hence the name contourlet transform [36].

Edge-based feature extraction methods [38 - 43] are used for object recognition where colour or texture cannot be used as a cue for recognition. Instead, the distinctive features of such objects are edges and geometric locations between them. As mentioned above, one of the biggest challenges when it comes to image retrieval is illumination variation. Compared to texture-based feature extraction methods, techniques that use edge information instead are partially illumination invariant and require less memory, which makes them a preferred choice in some cases. Line Edge Map (LEM) [39] can be used to construct a compact face feature for face coding and recognition. The investigation [40] shows that with LEM the extracted features of the face are discriminative and non-discriminative. Active Shape Model using GPU(GASM) [41, 42] is a popular statistical model for object localization based on the famous Snake algorithm. Compared to CPU, the graphics-processing unit allows for substantially bigger facial feature extractions in video or image sequences. In fact, the acceleration improvement is so significant the GPU reported a performance boost 48 times greater than compared to CPU implementation [41]. Main advantage of the ASM compared to other feature extraction algorithms is the model can only deform in ways learnt from the training set, meaning it can deform considerably and maintain specificity to the object intended for representation at the same time. Histogram of Oriented Gradients (HOG) [43] are an efficient descriptor for object detection and recognition. They are generally used in computer vision, pattern recognition and image processing for detecting and recognizing graphic objects such as faces. HOGs are especially efficient in detecting faces with occlusions, pose and illumination variation, because of the robust feature set, in which face features are extracted in a regular grid.

A different approach combines techniques for extracting global and local features. Principal Component Analysis (PCA) [44, 49] is a common method used in statistical pattern recognition and signal processing invented back in 1901. PCA extracts global and low-dimensional features about a pattern in an image. The pattern often contains redundant information. In order to get rid of that redundancy, while still preserving the key descriptive information, the pattern is matched to a feature vector. As a result, the extracted features are used to discriminate between input patterns in an image. In comparison, Independent Component Analysis (ICA) [45, 46, 49] is a novel statistical technique in machine learning and signal processing used to extract local features by using multi-channel observations. ICA aims to find linear projections of data features that maximize their mutual independence. Stepwise Linear Discriminant Analysis (SWLDA) [50] is another efficient technique for

extracting localized features. SWLDA employs forward and backward regression models to extract a small set of features. During forward regression, the most correlated features are isolated on the basis of defined class labels or F-test values, while the least significant are removed from the regression model during backward regression [29]. The main advantages of SWLDA over other techniques are its computational simplicity, predictive ability and its performance doesn't suffer from illumination variation.

The first techniques used for feature extraction in facial expression recognition were geometry-based [19, 51, 52]. This means the extracted features were based around the shape of the face and its parts – mouth, eyebrows, nose, rather than describing the texture of the face. Most geometric-based techniques employ Active Appearance Model (AAM) or variations of it to localize and track a dense set of facial points [51]. This means that the shape of the face is extracted by monitoring and combining these facial points in different ways as the expression evolves. Mainly because of that, these methods perform better on a dataset comprised of videos or image sequences rather than individual images. Local Curvelet Transform (LCT) [19] is an effective feature extraction method that achieves localization in frequency and time domain. Some of the points and lines on a face can be better extracted than wavelet transform which deals with point singularities [52]. One disadvantage of LCT is that the features extracted are usually quite large and other dimensionality reduction techniques must be applied to overcome this issue [19, 52]. Patch-based feature extraction techniques are less used for facial expression recognition that extract patches of the image. After that, the patches are classified to a specific class and the whole image is classified based on the individual patches. This approach is useful when the image archetype is too complex [53].

[20, 22, 33, 34, 35, 37] have shown that when it comes to designing a FER system, for the feature extraction stage texture - based techniques yield best results, since appearance-based extracted features have more significance than others. More recently developed similar techniques are Discrete Wavelet Transform (DWT) [54], Local Directional Number (LDN) Pattern [55], Local Directional Ternary Pattern (LDTP) [56] and KL-transform Extended LBP (KELBP) [57]. It's also important to note that in recent years, various dimensionality reduction techniques have been applied to features that have high dimensional vectors, in addition, to better determine the significance of the features similarity scores and various algorithms for e.g. Adaptive Boosting (AdaBoost).

## **Image Classification**

Image Classification is the last phase in developing a FER system where the output of the feature extraction process is fed as an input to a classification algorithm or classifier and during this stage expressions are categorized as emotions such as happiness, sadness, anger, fear, etc. Similar to feature extraction stage, because of the fastest development and interest of FER in computer vision, a lot of different classifiers have been developed over the years. A distance-based classifier is Euclidean distance metric [58]. The training and datasets for one subject consist of images with different expressions shown. When the model has to classify a certain image, Euclidean distance is calculated between the points on the test image and the points on the training images for the same subject extracted during feature extraction. The expression shown on the test image is classified as the expression shown on the training image, for which the minimum Euclidean distance is found. Euclidean distance is more suitable for static images due to its ambiguity for real-time or robust images [58]. Another distance-based classifier is Minimum Distance Classifier (MDC) [59]. Although its classification accuracy is usually lower than more complex classifiers such as Convolutional Neural Networks (CNN) or Support Vector Machine (SVM), MDC is still used in various areas of pattern recognition due to its computational simplicity and fast execution time. With MDC, an unknown pattern is classified to a category to which the nearest prototype to the pattern belongs [59, 60]. K-Nearest Neighbours (KNN) classifier [16, 61, 62] is another simple, distance-based algorithm that classifies objects based on nearest training examples in the feature space. The object is classified according to adjacent object classes, meaning the object is assigned to most common class among its  $k$  nearest neighbours ( $k > 0$ ), so if the value of  $k$  is one, then the object is classified to the class of that nearest neighbour. After this stage, images are converted to vectors of fixed length with real numbers and a distance-based algorithm such as MDC or Euclidean distance is used to classify the whole image [61,62]. KNN can be defined as lazy learning or instance – based learning in which the function is only calculated locally, and evaluation is postponed until classification. In this way, KNN can be used to determine the significance on the  $k$ -nearest neighbours of an object by weighing their contribution. Common factors for why all distance-based algorithms are applied for classification is simplicity, speed, and ease of understanding. A more complex classification technique is Hidden Markov Model (HMM). HMMs are probabilistic models which consist of two random processes – Markov Chain comprised of several countable states, and a second process that defines the output transitions and corresponding emissions [50, 63]. HMMs are often used in speech recognition and more recently for FER in image sequences as well, since they can model temporal dependencies [50]. Support Vector Machine (SVM) [21, 64, 65] is a technique that combines related supervised methods used for classification and regression. SVMs use machine learning techniques which use a hypothesis space of linear functions in high dimensional feature space to maximize prediction accuracy [64]. In fact, they are one of the most



efficient classification techniques for dimensionally large data [21]. SVM systems are computationally complex, but perform as well as sophisticated neural networks, delivering high accuracy in FER systems, hence why they're really popular classification technique for pattern recognition-based problems and regression-based applications. For real-world FER systems Extreme Learning Machine (ELM) [66 - 68] is feed-forward neural network technique classification technique with only one hidden layer of nodes originating from the study of single hidden layer feedforward neural networks [66]. Unlike conventional neural networks, with ELM there's no need of tuning the hidden layer for weight adjustment, which makes them substantially faster and greatly reduces processing time for the data. In comparison to SVM, LBP, KNN, Extreme Machine Learning is an efficient classification method for real-world FER systems where the data is noisy and imperfect with a lot of constant transient changes. In addition, Online Sequential Extreme Learning Machine (OSELM) originates from ELM, where the data is split and learned in individual batches, for which the output weights are constantly updated using a Recursive Least Squares (RLS) algorithm. The main advantage of OSELM is that it can provide better generalization performance at a much greater learning speed compared to other classification techniques [68]. A popular rule-based classifier used for FER is ID3 Decision Tree (DT) [69]. ID3 builds a decision tree from a fixed set of examples, which is later used to classify other data. From the decision tree predefined rules are extracted to produce competent rules [5]. Compared to other algorithms, ID3 is robust to noise and has an easily interpretable tree structure (if-then-else), meaning it can be extended to multiple output values. Learning Vector Quantization (LVQ) [70] is a supervised, prototype-based, artificial neural network (ANN) algorithm that supports both two-class and multi-class classification problems. Compared to KNN, LVQ allows for choosing how many training instances to hang onto and learns exactly what those instances are supposed to look like, rather than having to hold onto all training instances. Finally, probably the most used technique in image processing and in particular FER use neural networks. Other than the ones already discussed above, the most popular ones for FER are Bayesian neural network, Deep Neural (DNN), Network, Artificial Neural Network (ANN) and Convolutional Neural network (CNN). The main problem with image classification is to find useful features from the feature extraction stage and this exactly what neural networks are great at as they can automatically create and select the most important useful features along with having the ability to learn extremely complex classification models. Some types of neural networks are also able to extract useful features invariant to transformation such as transformation, transposition, scaling, relocation etc.

Table 1. [5]

Performance analysis of FER techniques.

Author name, year	FER method name	Database name	Complexity	Recognition accuracy (%)	No. of expressions recognized	Major contribution	Advantages
Gao et al. (2003)	LEM, dLHD	AR	Less	86.6	3	Oriented structural features are extracted	Suitable for real time applications
Noh et al. (2007)	Action based, ID3 decision tree	JAFFE	Less	75	6	Facial features are discriminative & non discriminative	Cost effective in speed and accuracy
Bashyal et al. (2008)	GF, LVQ	JAFFE	Less	88.86	Not reported	LVQ performs better recognition for fear expressions	Better accuracy for fear expressions
Zhao and Pietikäinen (2009)	GASM, SVM	CK	High	93.85	6	Adaboost learning for multi resolution features	Flexible feature selection
Song et al. (2010)	LBP-TOP, SVM	JAFFE, CK Realtime	Less	86.85	7	Detection of facial features point motion & image ratio features	More robust to lighting variations
Wang et al. (2010)	SVM	JAFFE	Less	87.5	Not reported	DKFER for emotion detection	More efficient emotion detection
Zhang et al. (2011)	Patch based, SVM	JAFFE, CK	Less	82.5	6	Capture facial movement features based on distance features	Effective recognition performance
Poursaberi et al. (2012)	GL Wavelet, KNN	JAFFE, CK, MMI	Medium	91.9	6	Extraction of texture and geometric information	Wealthy capability for texture analysis
Ji and Idrissi (2012)	LBP, VTB, Moments, SVM	CK, MMI	Medium	95.84	6	Extraction of spatial temporal Features	Effective image based recognition
Taylor et al. (2014)	PCA, ICA, HMM	Own	Less	98	6	Multilayer scheme to conquer similarity problems	High accuracy with own dataset
Owusu et al. (2014)	GF, MFFNN	JAFFE, Yale	High	94.16	7	Feature selection based on Adaboost	Lowest computational cost
Demir (2014)	LCT, OSLEM	JAFFE, CK	High	94.41	7	Extraction of statistical features mean, entropy and S.D	Reliable algorithm for recognition
Zhang et al. (2014)	GF, SVM	JAFFE, CK	Less	82.5	7	Template matching for finding similar features	High robustness & fast processing speed
Dahmane and Meunier (2014)	HOG, SVM	JAFFE	High	85	7	SIFT flow algorithm for face Alignment	Robust to rotation, occlusion & clutter
Mahersia and Hamrouni (2015)	Streerable pyramid, Bayesian NN	JAFFE, CK	Less	95.73	7	Statistical features are extracted from the steerable representation	Robust features & achieve good results
Hernandez-matamoros et al. (2015)	Gabor function, SVM	KDEF	Less	99	Not reported	Segmentation of face into two Regions	High performance with low cost
Happy et al. (2015)	LBP, SVM	JAFFE, CK+	Less	93.3	6	Facial landmarks lip and eyebrow corners are detected	Lower computational complexity
Biswas (2015)	DCT, SVM	JAFFE, CK	Less	98.63	6	Each image is decomposed up to fourth level	Very fast & high accuracy
Siddiqi et al. (2015)	SWLDA, HCRF	JAFFE, CK+,MMI, Yale	High	96.37	6	Expressions are categorized into 3 major categories	High accuracy
Cossetin et al. (2016)	LBP, WLD, Pairwise classifier	JAFFE, CK, TFEID	Less	98.91	7	Each pair wise classifier uses a particular subset	High accuracy & less computation power
Salmam et al. (2016)	SDM, CART	JAFFE, CK	Less	89.9	6	Decision tree for training	Improved recognition accuracy
Kumar et al. (2016)	WPLBP, SVM	JAFFE, CK+, MMI	Medium	98.15	7	Extraction of discriminative features from informative face regions	Lower misclassification
Hegde et al. (2016)	GF, ED, SVM	JAFFE, Yale	Less	88.58	6	Projects feature vector space into low dimension space	Improves the recognition efficiency

Literature review summarized at table 1 shows that the datasets used for FER systems are JAFFE, CK/+, MUG, TFEID, Yale, AR, MMI, KDEF, MUG. The facial expressions recognized are usually 6: happiness, sadness, anger, disgust, surprise, neutral – however, with LEM only 3 facial expressions are recognized while some FER systems can recognize 7 expressions successfully with contempt as well. Accuracy performance is outstandingly high in lab – controlled environment with ROI segmentation at pre-processing stage and Gabor function for feature extraction with SVM classifier giving best results 99% [26]

## SYSTEM DESIGN

### **Background and Fundamental Concepts of Artificial Intelligence**

In order to better understand the different design and implementation choices made in this project, it is essential to have a solid understanding of the underlying theory and definitions of AI. AI is a simulation of the way human intelligence functions in computer programs. There are two subsets of AI: machine learning (ML) and deep learning (DL). Various AI algorithms are applied in both subsets that have one key ability: automation, that is by being fed structured data, they can learn and modify themselves without explicitly being programmed to do so. Training is the process of feeding data to an algorithm that learns about it. Both ML and DL apply different AI techniques in a similar way, except with DL there is a network of several layers with these AI algorithms. Each layer has its own interpretation of the dataset it is trained on. These networks are called Artificial Neural Networks (NN) and they attempt to simulate the neural networks existing in the human brain. Implementation of NNs in deep learning is referred to as models. Data fed to models is usually structured in classes each data sample belonging to a class. Classification is the prediction of the model for which class the current sample belongs to.

Image processing is part of AI, where data, or in this case images, is represented by numbers. These numbers represent the pixel values in an image. Combining the values of different pixels gives important pieces of information called features. Features is what AI techniques use for learning.

Already discussed above, the three main stages of image processing (FER) are image pre-processing, feature extraction and image classification. Any problem in AI is approached in the same way, except data can be in differing formats e.g. text, audio, image, depending on the problem at hand. Arguably the most important aspect of AI is data. Regardless of the problem we are trying to solve, having enough data to train on is a requirement ensuring better performance. However, datasets often contain noisy or redundant data samples as well, which could lead to misleading results. This is why different techniques such as normalization, cleaning, balancing and augmentation are applied during the first stage: data pre-processing. This stage has significant impact on the way data is interpreted by the AI model. Having pre-processed the data, second stage is feature extraction. Essentially, during this stage key features about the data are extracted, which are then passed to a different AI technique used for classification. There are many AI techniques developed for feature extraction and classification and appropriate choices are made about which ones to use depending on the problem we are trying to

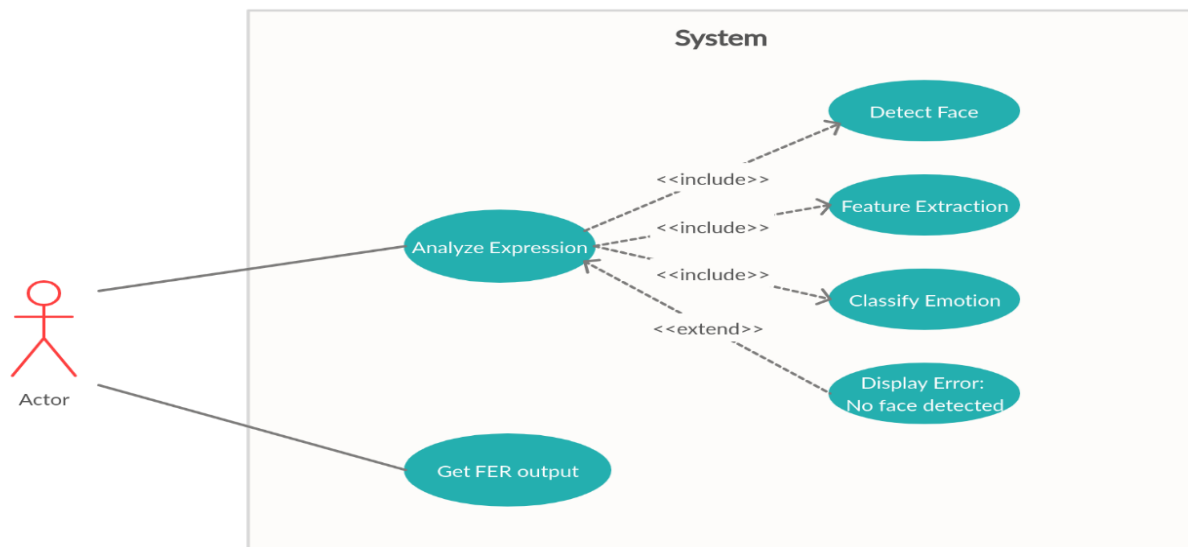
solve. The final performance of an AI model at solving a problem is entirely dependent on data size, pre-processing and choice of AI techniques.

### **Design Methodology**

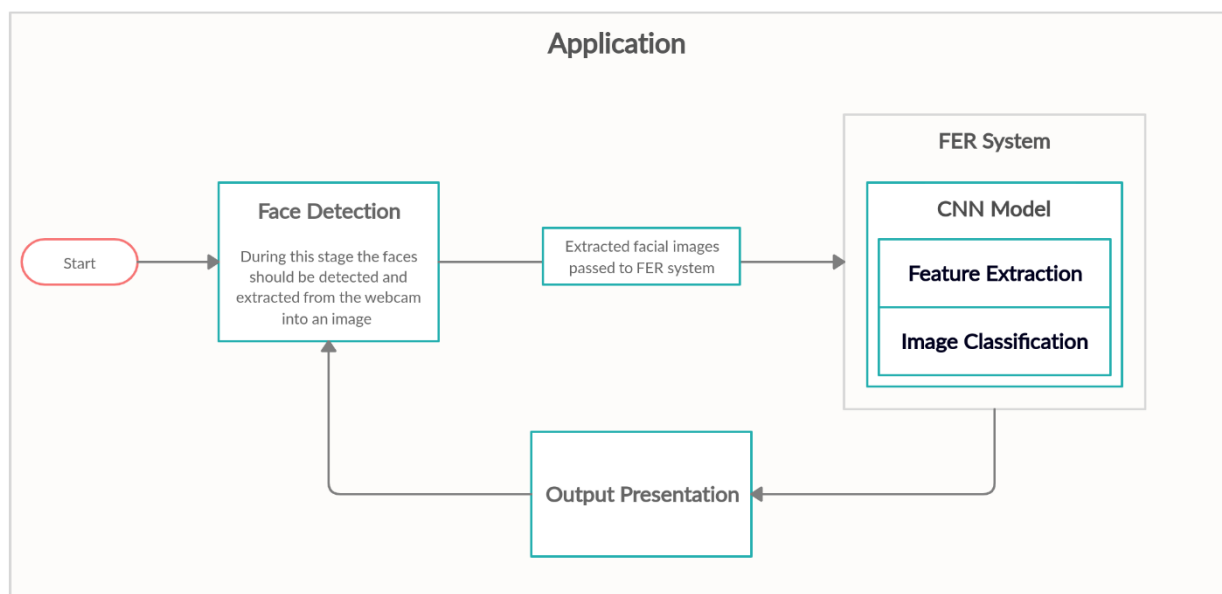
Basing on the fundamental concepts and ideas discussed, developing a FER system is approached in the same way. First and foremost, data is gathered from different databases. In most cases, training a nearly perfect AI model would require datasets with a substantial number of images, potentially billions. Datasets of similar size contain a lot more features and variations. As a result, a model is able to learn and extract significantly more key features which helps prevent overfitting. Overfitting is the biggest issue when dealing with small datasets. Smaller datasets mean there's only so many variations and independent features the model can learn about. As a result, sometimes the model overfits the data, that is informative features extracted only help the model learn about the training data, meaning it will memorize the data and classify it well, but perform poorly when tested on data that's not been seen before. In contrast, large datasets would allow for the model to generalize, that is extracted features would help the model learn about the actual classes it is predicting instead of the data it is trained on. This is where the first main challenge arises from. There's simply not enough data available online to be able to train a nearly perfect model. and the risk of overfitting is high. This issue is addressed during the next part: image pre-processing. This involves applying and comparing results of various techniques for data oversampling and then using the best one. In addition, ImageNet\_stats normalization helps getting rid of noisy and irrelevant data and to reduce the risk of overfitting even more and allow for better generalisation, data augmentation is performed. This involves applying different transforms to the data. Transforms introduce more variation between the images by slightly modifying some of them. Images are modified by changing orientation, contrast, brightness, mask, pixel values, etc. For example, taking two copies of the same image with different orientation. A human would easily determine if they are the same. However, with deep learning models, rotating an image would change everything: the pixel values, position of discriminant features and facial landmarks that a model uses for classification would be different, meaning the two images will be completely different for the model, even though they're the same. Taking this into account, it now makes sense why data augmentation reduces overfitting and helps generalisation. Augmented images are new for the model; hence the model would have to re-learn about them starting from scratch. In contrast, without augmentation, a model would learn new features about an image each time round and it might pick up irrelevant features that describe this particular image, rather than the class it belongs to. Moving forward, the best deep learning model architectures were trained on the augmented data. In order to evaluate which model performed best all models are evaluated with k-fold cross validation (CV) on an

independent dataset. If instead, CV was performed on a part of the training set, then the model would be biased performance results less precise. K-fold CV is the most accurate and popular way to compare different models' ability to solve the same problem [76].

Design image 1



Design image 2



Having trained a model ready to be embedded that can recognize emotions, the platform for FER in real-time is developed. The user's interaction with the system can be observed on Design Image 1. There's no action required on user's behalf. Facial expressions are analysed and classified the output

of which is presented to the user automatically. Low latency is one of the objectives of this project. Other alternatives were explored where the user would be required to request analysis and output, however decision was made to automate the whole process in order to achieve real-time FER at a decent frame-per-second (FPS). This ensures the low-latency requirement is met. The platform is designed to have a single streaming webcam where all the FER will take place. Now images from individual frames can be fed to the model for analysis, but this would prove to be inefficient. There's a substantial amount of information on a single image frame other than the faces of users. This means that when analysing the image, the model would pick up a lot of irrelevant background information which would most likely lead to wrong classification. To avoid that, firstly face detection is performed on every image in order to extract all faces visible on the webcam and dispose of redundant information. Result of face detection is presented to the user and his face is surrounded in a square box. Every detected face is then cropped and resized to same resolution size as the data the model is trained on. Now detected faces are fed to the model for classification, output of which is presented to the user with a label attached to the surrounding box of the user's face. This summarises the process that takes place for every FPS, which can be observed on Design Image 2. There are not many alternative overall design options to be considered here. The whole process must be automated, as any action required from the user would result in interruption of the webcam stream. The overall methodology for this project can be described as a waterfall approach. Requirements are clearly outlined, and the project is developed hierarchically, so there is no need for use of agile development techniques.

## DETAILED DESIGN AND IMPLEMENTATION

### Development Technologies

#### Python

Python is a high-level, general purpose programming language. Python is designed for code readability with simple syntax. In addition, the comprehensive choice of various libraries and frameworks developed specifically for tackling various machine learning tasks in Python expedite the development process and make it significantly easier. Due to these reasons, Python was chosen for development of the AI part of this project.

#### Ionic

Ionic is a hybrid mobile app development framework. Ionic is designed so that programmers can use web-technologies such as HTML, CSS, JavaScript, Angular etc to develop a shared code base. In addition, Ionic allows for easy deployment of applications to native android or iOS. Ionic is the platform that is used to attempt developing an application for real-time facial expression recognition in both android and iOS. Ionic is the platform that was used to develop the application for FER analysis

#### Angular

Angular is the preferred JavaScript framework used by Ionic for development of dynamic web apps. It has extensive support for two-way binding, modularization, AJAX handling, RESTful API handling, etc. and abundant selection of built-in tools that simplify the development process.

#### Apache Cordova

Cordova is another mobile app development framework used together with Ionic. Cordova is the framework that allows for use of web technologies HTML, CSS, JavaScript to create a native android or iOS application. In addition, Cordova has ample choice of plugins developed for use of all hardware and software smartphone features. In this case, cordova is used to access the smartphone camera.

#### Capacitor

Capacitor is the preferred cross-platform application runtime created specifically for Ionic and maintained by the Ionic Framework team. Capacitor provides APIs that allows developers to stay close to web development standards, while offering rich support of mobile native features. Capacitor is used for building and deploying the application to Android Studio

**JavaScript**

JavaScript is an incredibly powerful language universally used for web development. JavaScript is used in this project to develop a dummy web version of the FER system for testing purposes.

**Development Libraries****Fastai**

Fastai is a deep learning library built that uses PyTorch as its backbone. Fastai high-level support and functionality for quick development of state-of-the-art deep learning models. In this project, the fastai library is used to build and train all deep learning techniques explored in this project.

**Pytorch**

PyTorch is machine learning library, built on Torch, with abundant choice of tools and libraries for developing solutions to problems in computer vision and language processing. PyTorch is the underlying library of Fastai and although not directly used in this project, some of its features are used for data manipulation.

**Keras**

Keras is a neural-network library built for Python. It is designed to run on TensorFlow, which is the underlying library chosen to use with Keras. In this project, Keras is used to develop a full CNN model from scratch. It is preferred over fastai particularly for the CNN model as Keras's ease of use make building a CNN network significantly easier.

**Imblearn**

Imblearn is an API built for Python. It offers various features for most of the data balancing techniques used today in machine learning. In this project, Imblearn is used for exploring different oversampling and under sampling techniques for data balancing.

**Scikit-learn**

Scikit-learn is a machine learning library for Python, built to work with Python libraries NumPy, SciPy, extensively used in machine learning. It includes various features for regression, classification, feature extraction, data manipulation. In this project, Scikit-learn is used for k-fold cross validation.

**OpenCV**

OpenCV is a library built for python mainly used for computer vision tasks in real-time. In this project, OpenCV is used for streaming of webcam and face detection.



## **Development Environment**

### **Jupyter**

Jupyter Notebook is a web-based platform (IDE) that offers support for most widely used programming languages today. Jupyter is especially useful for data visualisation, machine learning, statistical modelling, thus why it is the preferred development environment for artificial intelligence over PyCharm. Jupyter notebook is used in this project for all AI-related development.

### **Google Cloud Platform**

Google Cloud Platform (GCP) is a platform offering a choice of cloud computing services and APIs developed by the Google team. In this project, GCP is used as a virtual machine instance for model training. GCP virtual machines offers enormous computational power that significantly helps expedite the training process. In this project, GCP is used to speed up all deep learning techniques training.

### **IntelliJ IDEA**

The application part of this project is developed entirely with the IntelliJ IDE. IntelliJ offers a suite of features among which extensive support for JavaScript and TypeScript dynamic syntax highlighting. In addition, IntelliJ offers easy deployment and debugging of web/mobile applications directly in the browser. Because of these reasons and many others IntelliJ is my preferred IDE for application development.

### **Android Studio**

Android Studio is the official development environment for native android applications, developed by Google. It is designed specifically for android development and it is built based on IntelliJ, meaning it has an abundant selection of features and tools. Android Studio is used in this project for deployment, debugging and testing of mobile application in device or emulator.

## **Third-party Development Tools**

### **Git**

Git is a version-control system for coordinating work amongst programmers. In this project Git is used for keeping track of different versions and features. All resources and developed code are stored in a GitHub private repository

## **Databases**

### **Kaggle**

Kaggle is the biggest data science/artificial intelligence community offering resources and tools for accomplishing different machine learning tasks. Kaggle provides over 70 datasets for different classification problems. Dataset from kaggle facial expression recognition challenge has a total of 15k labelled images according to one of the 7 emotions: anger, disgust, fear, happiness, neutral, sadness, surprise

### **KDEF and AKDEF**

The Karolina Directed Emotional Faces database includes a total of 490 facial expression images. The AKDEF averaged set is created from the original KDEF images and contains further 100 images. All images are labelled and classified to one of the 7 emotions: anger, disgust, fear, happiness, neutral, sadness, surprise

### **CK/CK+**

The Cohn-Kanade database was originally created in the 2000s containing image sequences with facial expressions taken from video frames. The extended CK+ dataset is created from the original CK with additional images. This dataset contains a lot of unlabelled images where facial expression shown is not clearly visible/expressed. Again, the dataset contains images for the 7 emotions: anger, disgust, fear, happiness, neutral, sadness, surprise

### **JAFFE**

The Japanese Female Facial Expression dataset contains a total of 213 all labelled images of 7 emotions: anger, disgust, fear, happiness, neutral, sadness, surprise. Images are labelled by 60 different subjects.

### **Oulu-CASIA NIR&VIS**

Oulu-CASIA facial expression database contains videos of facial expressions for one of the 6 emotions: anger, fear, disgust, happiness, sadness, surprise. Each video is split in frames that range from neutral to full expression.

## **TFEID**

Taiwanese Facial Expression Image Database offers a total of 7200 images for one of the 8 emotions: anger, fear, disgust, happiness, sadness, surprise, neutral, contempt.

## **Yale**

The Yale Face Database contains 165 grayscale images with 1 image per subject for each of the emotions: happiness, neutral, sadness, surprise, wink, sleepy

## **Facial Expression Recognition Models Development**

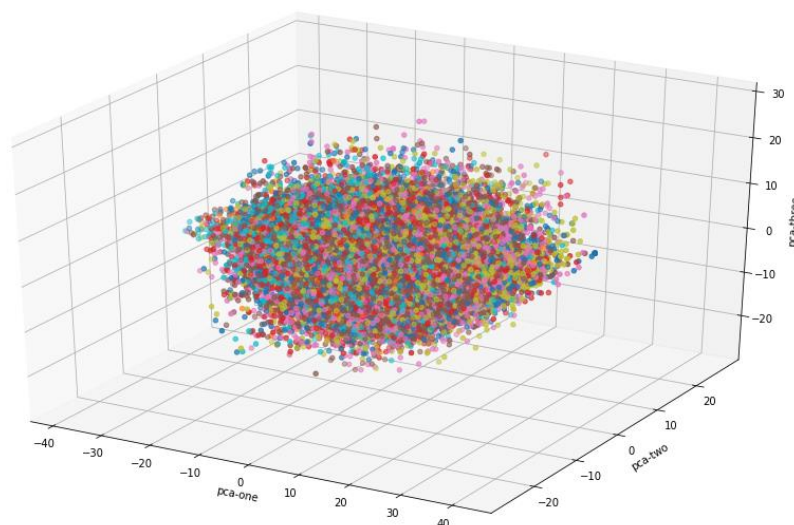
### **Data Collection**

Data collection as discussed above is the most important stage of the methodology. Data is the foundation of AI. Currently, there's no existing techniques in the world of artificial intelligence, which does not depend entirely on the dataset it is trained on. Even the best techniques in image processing cannot achieve good accuracy without an abundant well-prepared dataset. The major advancements in deep learning models and improvements in performance in the past decade are largely contributed to the bigger size and larger diversity of data collected. Due to its importance, ample time is spent on the first two stages of the methodology: data collection and data pre-processing. For this project, two datasets are needed. A training set, which models will be trained on and an independent test set required for performance comparison and evaluation with CV. Data is collected from multiple online databases for facial expression recognition described in detail above.

A decision is to be made whether all datasets should be combined into one and then that comprised dataset will be used to train a single model on, or alternatively, separate models will be trained on individual datasets and then predictions from each model will be combined using different techniques such as voting, sum of scores, etc to give a final classification. In this case, the former option is the most appropriate for a number of reasons. All databases used for sourcing the data apart from Kaggle have a small, unbalanced number of images for each emotion. Had the second option been chosen, then some of the models trained would have been biased towards emotions with more data samples, which would lead to misleading results when combining the predictions from each of the models. Moreover, as we know when dealing with small datasets, there is a high chance of overfitting, and as a result models could be trained only for a few epochs (cycles) otherwise overfitting is imminent especially with multi-classification problems as this one where data variation is key. Furthermore, for real-time FER quick predictions are required to ensure low latency and smooth FPS rate. Running predictions from multiple models and then combining results would require substantially more

computational power and as a result performance would be significantly degraded. Finally, option 1 allows me to comprise a diverse, balanced test set combining images from all datasets. This ensures results from CV performance comparison are as unbiased as possible. Taking these reasons into account, I decided to go for option one. Only disadvantage of this choice is this stage takes significantly longer as different datasets are structured differently, which does not allow for automizing the data sorting process. In addition, once complete, the new dataset must be carefully reviewed for any large discrepancies between images for each emotion or irrelevant data. The test set used is comprised of total 2142 data samples; 306 images for each emotion manually selected and reviewed from all databases apart from Kaggle. As for the training set, the full dataset from Kaggle is used. Kaggle database is the most abundant database with 31k images. Having 31k images as training set means there's enough data to fully train the model with smaller risk of overfitting. However, there's a trade off as the dataset is imbalanced with only 600 data samples for disgust and 8000 images for the emotion happiness.

**Dataset Image 1**



On dataset image 1 it can be observed the initial marginal distribution of data in 3D space. Not much can be distinguished other than the fact data samples from different classes are tightly coupled together with a lot of them overlapping. There are two major problems when it comes to dealing with datasets: imbalanced datasets and class overlap. The initial dataset used for this project shown on dataset image 1. is both critically imbalanced with lowest class distribution of ~500 images versus ~8000 for the most abundant class in terms of data and there is also significant class overlap mentioned above. The class overlap is mainly due to two major factors: some facial expressions such

as fear and disgust or sadness and neutral have similar key features such as the shape of the mouth and the wrinkling of the face. For the model to successfully distinguish between these an enormous dataset is required where the model can be trained for longer and in this way learn the exact features describing the data variations specific for a certain emotion. The class distribution of the initial training set and the final test set is shown on the tables below.

**Training Set Class Distribution**

<b>Emotion</b>	<b>Number of Images</b>
Anger	4953
Disgust	547
Fear	5121
Happiness	8989
Neutral	6198
Sadness	6077
Surprise	4002
Total:	35887

**Test Set Class Distribution**

<b>Emotion</b>	<b>Number of Images</b>
Anger	4953
Disgust	547
Fear	5121
Happiness	8989
Neutral	6198
Sadness	6077
Surprise	4002
Total:	2142

### **Image Pre-processing**

Before any data augmentation is done, the dataset needs to be balanced to avoid bias. Currently existing techniques balance out datasets by either oversampling under-represented classes or decreasing the number of data samples for most populated classes known as data under sampling. Resampling gives a copy of the original data with better class distribution. Since the training set for

this project is of relatively small size different oversampling algorithms are explored as larger dataset allow for more training and reduce risk of overfitting as previously explained. 3D image class distribution is to be seen on Dataset Image 1. The images have been reduced with Principal Component Analysis (PCA) to three components or features as this requires significantly less computation power and the generated graph is easier to interpret as there is less noise present. Not much can be seen on the graph except all images are marginally tightly coupled together and there is a lot of class overlap present, which might prove the determining factor when choosing an oversampling technique. The four most widely used techniques today are explored: Random Oversampling, Smote, Adasyn and SmoteTomek

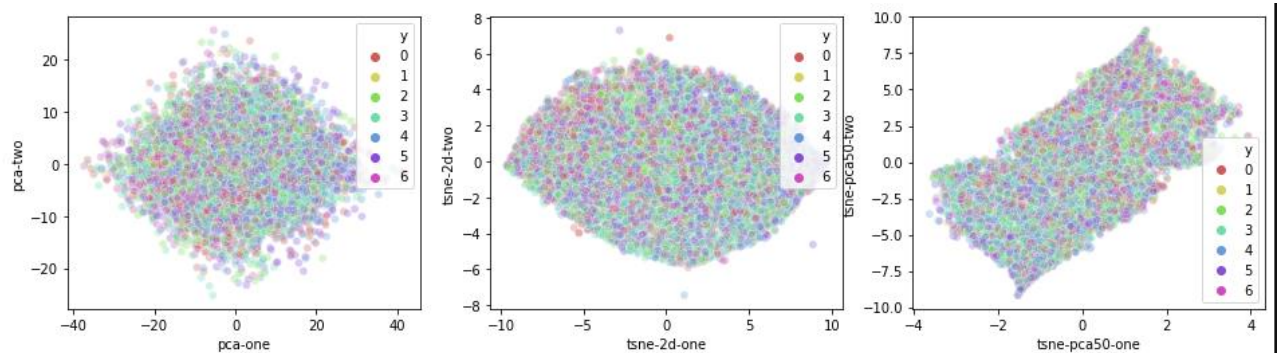
Resample images 0-2 from [73] are used to illustrate the various oversampled techniques explored in this project. To alleviate the computational power requirements the resampling is done over a dataset of 1k samples with two distinct features for each sample. The points represent individual data samples. The different colours are for classes that each sample belongs to.

Visualisation images 0-4 are used to illustrate the marginal data distribution of the oversampled dataset for each of the techniques. Unfortunately, the dataset this project is using consists of 48x48 images, which means there is 2304 distinct features in a single image with nearly 40k images total. Visualisation of that many features and for each data sample when the size of the dataset is that large is simply not possible even with computational power of Google Cloud Platform within an acceptable time frame. To overcome this problem, before visualising the dataset is decreased to a certain number of components, but in this way key features are stripped away from images, so visualisation is biased. Moreover, even if that makes visualisation computationally possible within an accepted timeframe, 40k images is still a lot to visualise on such a small plotting space. In addition, data variations between classes are not substantial, which means most of the data samples will appear really close to each other or overlap. Yet, in order to better understand the effects of oversampling techniques an attempt is made to visualise the data along with an explanation what to look for. Each visualisation image has 3 separate graphs. The first graph is a representation of the data with 3 features for each image sampled down and visualised with Principal Component Analysis. The second graph represents the data again with three features, but using the t-SNE algorithm instead of Principal Component Analysis. Finally, the third graph shows the data representation with 50 features for each image down sampled with Principal Component Analysis and visualised with t-SNE technique. Visualisation Image 0 shows the original data distribution. We can see that regardless on the number of features: 3 or 50 and regardless of the technique used: PCA or t-SNE data samples still appear too close together with

significant class overlap. Mostly because images are stripped of most of their features, data samples representing different classes are not bulked together and separated in patterns, but rather all mixed. Some of the code for visualising data with PCA and t-SNE is taken from [74] and accustomed to work with the dataset for this project.

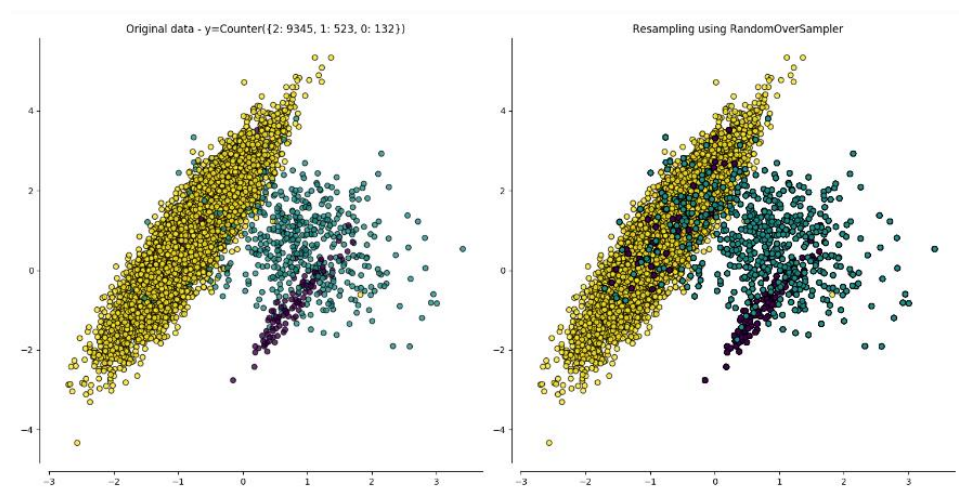
Further testing is done of all 4 oversampling techniques used. The testing involves a simple train of a ResNet 34 model architecture for 4 epochs (cycles). rComparison Images 0-3 show the results of this test. This test is under no circumstances exhaustive or objective, as a model learning process is never the same twice, however it is a relatively good indicator about which dataset is best to train on. If the reader is unable to understand the test images, please refer to Appendix A for more details.

**Visualisation 0**



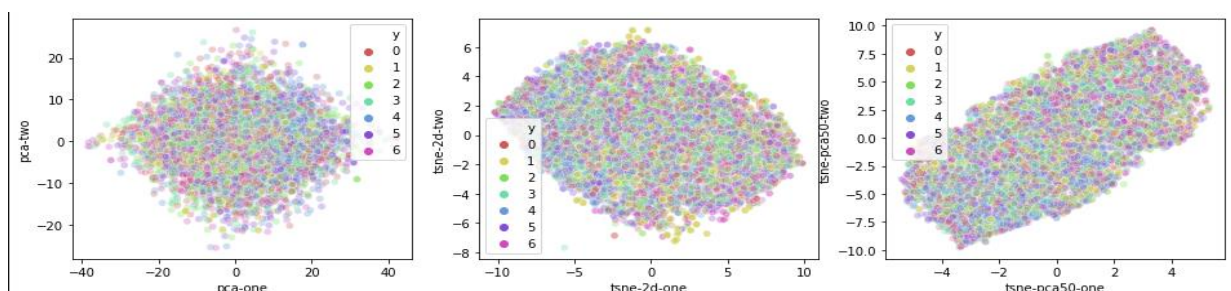
## Random Oversampling

**Resample 0**



Random oversampling is the most basic resampling technique of all. The algorithm essentially randomly picks data samples from under-represented classes and makes new copies of it. Compared to the other techniques explored in this project the random oversampling process is short and less resource requiring in terms of computational power. Moreover, due to using a random number generator for selecting samples to make copies of, it is considered the most unbiased way to select data for multiplying, as each sample has an equal chance of being selected. As for disadvantages when compared with other oversampling algorithms, there's a high risk of choosing samples for copying that are not representative of its class, i.e. have more redundant informative features. This can prove a significant problem when dealing with noisy high-dimensional datasets as it can introduce even more noise. Furthermore, with random oversampling it is not possible to implement the usage of additional information about the data when choosing samples for copying. Lastly, the risk of overfitting for the oversampled classes is increased as random over sampler simply adds new copies of the data to the under-represented classes, making the decision boundary tighter. Resample 0 shows the outcome of random over sampler. Notice how particular pattern can be found in the way data is resampled. Although random oversampling is the simplest technique of the ones explored before any comparison is done, it seems it might be the best technique for this dataset, mainly because of the class overlap issue. Random over sampler is unbiased in data selection for copying and that makes it more appropriate for such a high-dimensional marginally clustered dataset. Data visualisation for this technique can be observed on Visualisation 1. The pattern of data distribution is vastly similar when compared with data distribution before oversampling on Visualisation 0, except the shape data samples form is slightly different. The number of data points is so large, that it is impossible to visualise them properly so that the exact pattern of oversampling can be analysed. The resulting dataset has a total of 62923 data samples with 27036 new images added. Class distribution is perfectly balanced with 8989 images. Because it was chosen to oversample all classes but the majority one, the algorithm has oversampled enough to get all classes with the same number of images as the majority one: 8989. Further testing results can be observed on rComparison 0. It can be seen that the model achieved a 45% accuracy with the oversampled dataset.

**Visualisation 1**

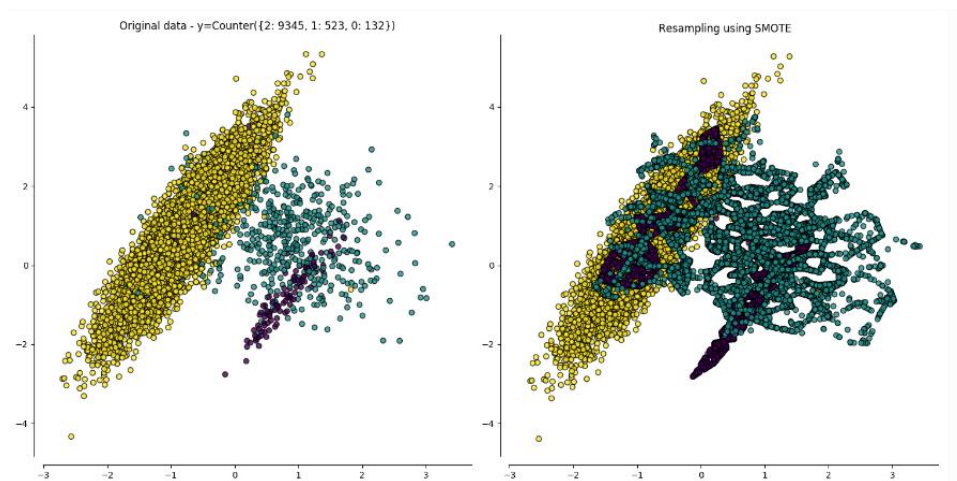




rComparison 0

epoch	train_loss	valid_loss	accuracy	time
0	1.900936	1.798437	0.300143	01:30
1	1.718637	1.589949	0.392244	01:26
2	1.564928	1.466357	0.443182	01:27
3	1.544243	1.443114	0.450016	01:27

Resample 1

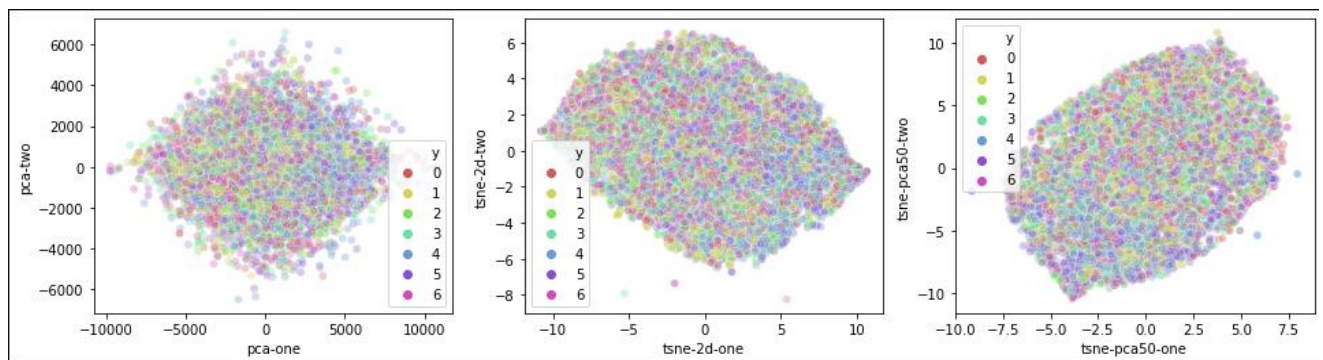


## Smote

Instead of generating direct copies of the data chosen random, Smote presents a more complex synthetic minority oversampling technique. Synthetic means newly generated data samples are similar rather than exact copies of the original data. These are generated by computing nearest neighbours with k-nearest neighbour (kNN) algorithm and then introducing the newly created samples by joining line segments along the lines of the nearest neighbours. [72] has proven that the synthetic copies Smote generates are not exact copies when presented to an ML algorithm, meaning the risk of overfitting is smaller and decision boundary softer when compared with random oversampling technique. This allows for the model to approximate the hypothesis more accurately: generalize. On the other hand, Smote does not account for the fact that neighbouring samples calculated with kNN might belong to different classes. As a result, the overlapping of classes can be increased, thus introducing more noise in the dataset. In most cases, when dealing with high-dimensional data this algorithm proves inefficient because of that. In addition to original Smote, other algorithms such as BorderLine Smote, KMeans Smote, SVMsmote are tested. BorderLine Smote works by picking data samples marginally situated on the borderline between different classes and create synthetic copies of them. This variant of Smote was dismissed as a possible oversampling technique for this project due to the fact this technique significantly increases clustering of images for different classes and as a result

increase class overlap even more. KMeans Smote uses KMeans clustering before oversampling the dataset with Smote. This technique is tested, and the resultant dataset has 0 data samples for 5 out of the 7 emotions and 8989 images for each of the remaining two emotions. It is attempted to oversample all classes, but the majority one, however, due to the resulting class distribution this technique is also dismissed as a possible option. Lastly, SVMSmote is explored. SVMSmote uses Support Vector Machines to pick samples for synthetic oversampling. This technique cannot be applied to the dataset used for training in this project due to computational power requirements. SVM time complexity quadruples for the size of the dataset, which means this technique cannot finish running in a timely manner for a dataset with 35k images and 2304 features for each individual image. Lighter versions of SVM such as SVC and SVR are tested as well, but for none of these the oversampling process executed successfully. On resample 1 it can be seen the oversampling process using the original Smote method. In contrast with random oversampling, here there is now a clearly visible pattern that is shaping during the process. It is observed that Smote introduces a lot of class overlap, which proves to be a major problem with our dataset already. In order to avoid making it worse, Smote is not the preferred over sampling technique. The resultant dataset has an even class distribution of 8989 data samples for each emotion. Similarly to random oversampling, it was chosen to oversample all classes but the majority one: happiness with 8989 images, hence why all images are oversampled to exactly 8989. Visualisation 2 shows the resulting dataset. Again, not much but some overall change of shape can be observed, due to the sheer number of images. Further testing results on image rComparison 1.

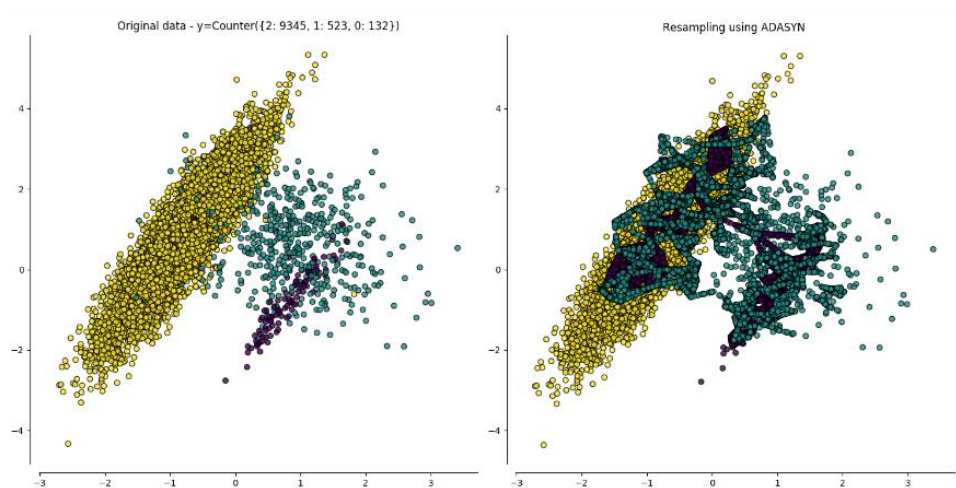
**Visualisation 2**



**rComparison 1**

epoch	train_loss	valid_loss	accuracy	time
0	1.883589	1.765201	0.319294	01:29
1	1.680627	1.557861	0.402734	01:32
2	1.564407	1.450574	0.444771	01:28
3	1.508760	1.431720	0.452956	01:28

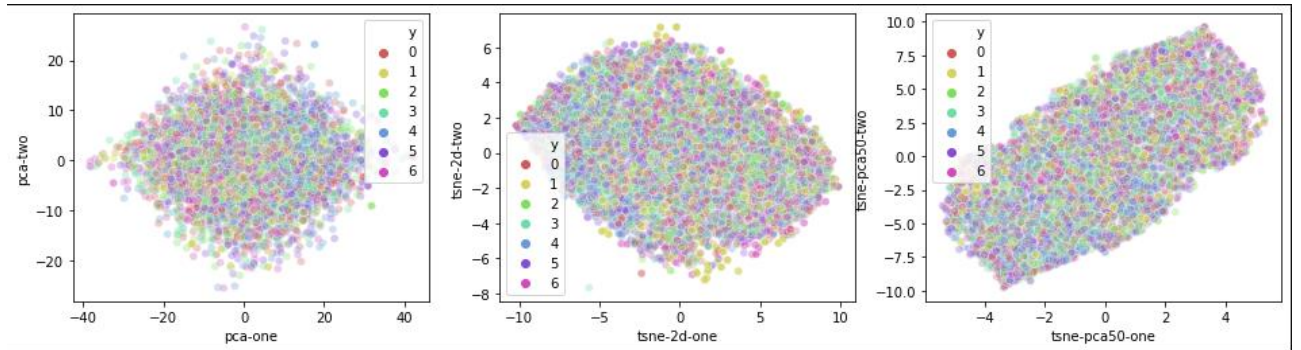
## Resample 2



### Adasyn

Another synthetic oversampling technique similar to smote is Adasyn. Essentially Adasyn is an improved version of Smote. Again, Adasyn uses kNN algorithm to find nearest neighbours of a data sample and then generates synthetic samples along the line segments of data samples, which are hardest to classify from that class. In contrast, Smote does not make that distinction, and simply picks neighbours at random. In addition, Adasyn modifies the values of the points where kNN neighbours marginally reside; hence adding some variation to the dataset, thus making it more realistic. On resample image 2 it can be observed the clusters of new synthetic data samples that are formed around marginal points hardest to classify. Adasyn can be especially inefficient when dealing with high-dimensional noisy datasets as it generates a lot of class overlap with the newly created data samples. This can be observed on resample 2 as well and based on it Adasyn is not the preferred choice of oversampling technique as significant class overlap is already an issue with the original dataset used. The pattern is analogous to the Smote algorithm, except here bigger clusters of newly generated synthetic copies can be seen around points which are marginally farthest away from the rest of the data samples from the same class. This is due to Adasyn prioritising data samples which are hardest to classify and then generating new points along them. Resample 2 clearly shows the substantial class overlap that is introduced with this technique. This makes Adasyn an inefficient oversampling method for this project. Visualisation 3 shows a similarly cluttered dataset to other oversampling techniques, except from resample 2 it is now known that there is much more overlapping. Further test results give accuracy of 42% after 4 epochs which is significantly less when compared with random oversampling and Smote that can be seen on rComparison 2.

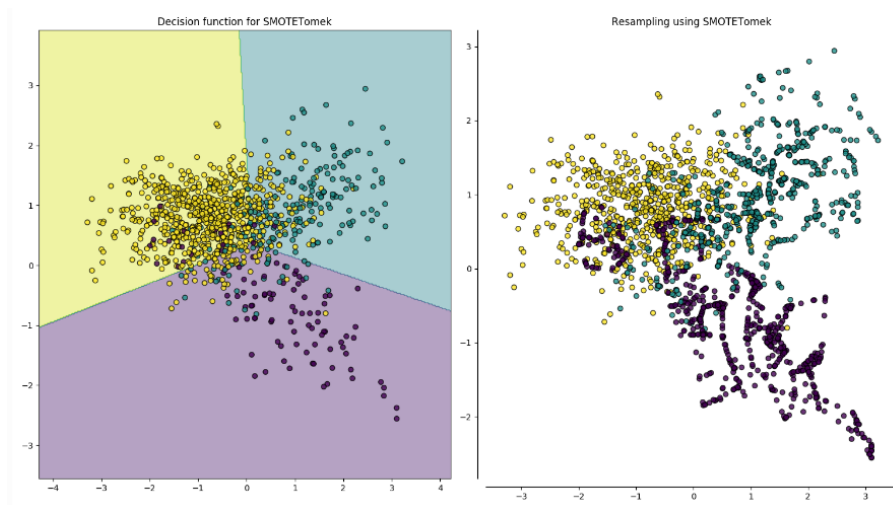
Visualisation 3



rComparison 2

epoch	train_loss	valid_loss	accuracy	time
0	1.895834	1.824785	0.297135	01:07
1	1.700721	1.629708	0.374923	01:06
2	1.596939	1.543250	0.406577	01:06
3	1.548382	1.505370	0.423136	01:04

Resample 3

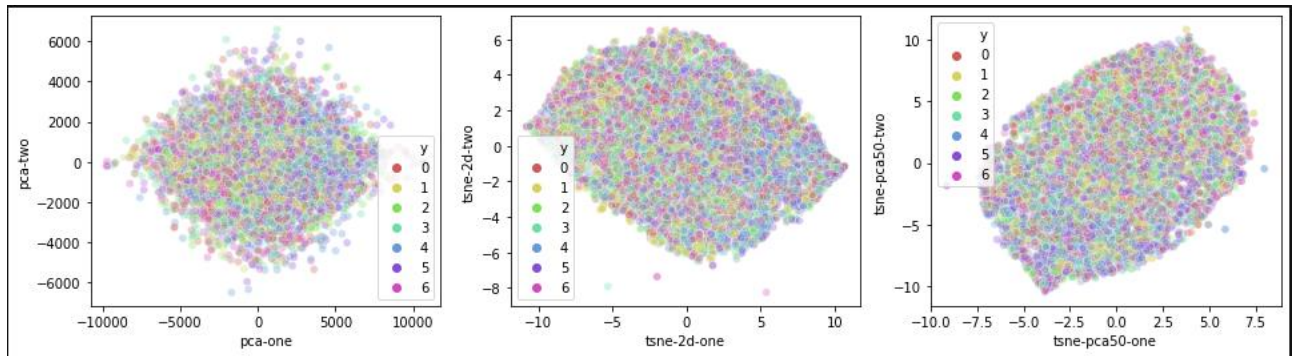


## SmoteTomek

SmoteTomek is a combination of oversampling technique Smote with undersampling technique Tomek Links. Firstly, class minorities are oversampled with Smote, after which data is undersampled with Tomek Links. Tomek Links essentially get rid of class overlap by removing the links between data until all data samples at minimal distance from kNN are from the same class. Resample image 3 shows the effect of SmoteTomek. The newly generated data samples from Smote are forming a pattern, while some of the overlapping points representing different classes have been removed during Tomek links

removal. The result is still a more balanced, less noisy dataset compared to Adasyn and Smote. Unfortunately, SmoteTomek cannot be applied to the dataset used for this project. The resulting dataset from this technique has a class distribution of 8989 for every emotion similar to Adasyn and Smote. This is due to the fact, TomekLinks do not perform any under sampling to a dataset that is firstly oversampled and balanced with equal number of data samples for each emotion. In addition, under sampling is generally avoided in this project, due to the fact it is almost always better to have a larger dataset more diverse with data variations, which as mentioned above helps model learn better and generalise. Visualisation 4 shows a similar resulting dataset to the ones using techniques Smote and Adasyn. Test results give 2% lower accuracy with this technique.

**Visualisation Image 4**



**rComparison 3**

epoch	train_loss	valid_loss	accuracy	time
0	1.888132	1.792163	0.313635	01:03
1	1.698300	1.621410	0.377349	01:03
2	1.590848	1.515550	0.425215	01:04
3	1.569074	1.489010	0.432542	01:03

To summarise the oversampling process for this project, naïve random oversampling and Smote are the two techniques chosen as most appropriate for this project due to the reasons mentioned above. Further testing is done on the resulting datasets from both methods in order to determine which is the best one. A ResNet34 architecture is trained again, but this time the whole network is trained, fine-tuning the learning rate. To sum up the results, with random oversampling 77% accuracy is achieved, whereas with Smote the maximum is 66%. This proves that sometimes the simplest is most effective. In this case, random oversampling yields best results due to the characteristics of our dataset. When having a high-dimensional, multi-class, cluttered dataset with substantial class overlap it is advisable to avoid Adasyn and Smote as they significantly increase the class overlap and as a result currently

existing models are simply not able to learn about the dataset at all. The final image distribution for each emotion for the final training set used in this project can be observed on the table below.

**Training Set Class Distribution**

Emotion	Number of Images
Anger	8989
Disgust	8989
Fear	8989
Happiness	8989
Neutral	8989
Sadness	8989
Surprise	8989
Total:	62923

### Normalisation

Real world datasets contain features that substantially differ in scale. As a result, some feature representations might be more dominant than others. If such a feature is irrelevant for overall class representation, this might lead to wrongful predictions. In simple terms, the pixel values of images have three channels: R, G, B and sometimes the values for some of these channels might be too bright, too dark or varying too much, while others would have no variation at all. So as a result, the training of a deep learning model is significantly improved if the values of the R, G, B channels for all pixels have a mean value of 0 and a standard deviation of 1. A technique of feature scaling for achieving that is Min-max normalisation. Min-max normalisation rescales all the range of feature values to the range of (0, 1) This ensures all features are equally weighted in their representation, helping reduce model bias towards some classes. For example, with most algorithms using Euclidean Distance or kNN classifier, features with significantly different, larger scale will be weighed in as more dominant over others in distance calculation. This will lead to wrongful results and a biased model. Importance of min-max normalisation is essential for most AI techniques and especially distance-based methods such as Euclidean distance. In addition to feature scaling, this step of the process involves standardising sizes of all images. In this case, images are all resized to 48x48. In this project data is normalised using the imagenet\_stats. A decision is to be made regarding which image statistics to use to normalise the dataset for this project. Imagenet is a high-dimensional dataset with over a 1000 categories and more than 1 million images. The model architectures used in this project are pre-trained on the Imagenet



dataset. In addition, the dataset used for this project is somewhat similar to Imagenet. Although, the dataset used for this project is not as abundant and does not contain as many class categories, it is still relatively high-dimensional with 2304 features in each image. Due to the similarity of the two datasets, and the architectures used it makes sense to use the imagenet statistics for normalisation.

### **Data Augmentation**

As mentioned above, large quantity and rich diversity of data are prerequisites for good model performance. Data augmentation is a popular technique that allows for enriching data diversity without actually having to collect more data. Data augmentation involves different scaling, cropping, resizing, horizontal and vertical flipping techniques done on the original data. It significantly helps reduce risk of overfitting and as a result aids model generalisation, allowing the researcher to train a neural network for longer. With fastai library data augmentation is represented by transforms, involving the usual data augmentation methods. For full list of transforms please visit [75]. Every technique on that list is used to augment the data in various ways with max\_rotation set to 50. That is to avoid having images flipped horizontally, as it would not make sense if the head pose is inverted.

### **Model Training**

In ML, different techniques are used to perform the feature extraction and classification stages of image processing. However, in DL feature extraction is a process performed automatically at every layer before classification. As a matter of fact, models are neural networks, which means each layer of the network has its own feature extraction classification. There are a few reasons why DL techniques are preferred over ML techniques usually used for image processing in this project. To begin with, deep learning techniques have ability to learn highly complex models, due to their stacked layer architecture of neural networks. In addition, convolutional neural networks (CNN) were developed specifically for image processing and has proven to give best performance on most popular datasets and object classification, detection, segmentation problems in the past decade. Because of its popularity and outstandingly accurate results, CNN architecture models and its variants are explored in this project. For detailed overview of CNN architecture, refer to appendix A. CNNs perform significantly better than other existing at image processing tasks mainly because of the automatic hierarchical extraction of features during feature extraction stage. Hierarchical means features from upper layers representing points, edges are carried onto deeper layers to form whole shapes and patterns and finally objects in the image. Taking all this into account, it makes sense that deeper NNs will be able to learn more complex models and perform even better, however that is not the case due to the vanishing gradient problem. Essentially, as more layers are added to a network, each layer using a certain activation

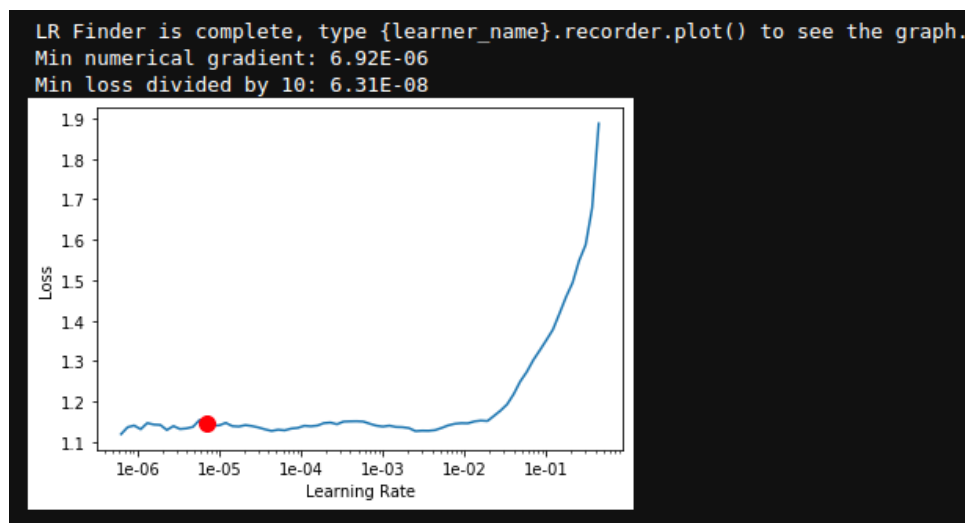
function, the gradients of the loss function approach zero, making the network hard to train. To read more about the vanishing gradient problem, please refer to Appendix A. Basically, because of this problem researchers are not able to train deep neural networks with more than three layers up until 2012 when the invention of AlexNet with 8 layers and later GoogleNet with 30 layers showed that it is possible to train a deep neural network with more than a few layers. In 2015 ResNet architecture is introduced. To read more about ResNet architecture please refer to Appendix A. The architectures explored in this project are ConvNet, ResNet34, ResNet50; 3 models are trained over the same dataset after oversampling.

### **Residual Neural Network**

In its essence, ResNet architecture includes an extra layer – the residual block, that alleviates the vanishing gradient problem. In simple terms, the residual block is a skip connection that transfers output to deeper layers, allowing to build deep neural networks. The essential difference between ResNet34 and ResNet50 is that the higher the number the more individual layers and layer groups there are. Due to computational power limitations architectures ResNet102 and ResNet152 are not explored, however in some cases deeper NNs such as these can improve predictions due to the extremely deeper representations. Architectures with a 100 or more layers do not seem appropriate for this project not only due to the incredibly large training times, but also using them in real-time would significantly degrade the FPS rate and as a result, fail to meet the low latency objective for this project. The ResNet architectures are trained with a similar strategy, whereas the CNN network is built from scratch by introducing individual layers manually. ResNet models are pre-trained on the imagenet dataset as mentioned before. In order to train the whole network, firstly the maximum learning rate must be found. The learning rate represents the minimum gradient loss function, or the point of the gradient where the loss is the smallest. In simple terms, the learning rate means how fast the network updates parameters during the training process. A learning rate which is too small will allow the network to improve only in small increments. However, if a value that's too high is chosen, the loss of key features will likely increase. As a result, a custom learning rate is introduced when training, fine-tuning the individual layer groups in order to achieve best results.



### Learning Rate



The default learning rate is 0.003, so to fine-tune it the following principle is followed: the initial layer groups are trained with a small learning rate, then for the mid layer groups the learning rate is increased a bit and for the last dense layer groups the learning rate is changed the most. This is due to the fact initial layers only capture edges, lines and points in an image, which can be observed in any dataset. This means that changing the learning rate too much at this stage is pointless, as the pre-trained model is already good at recognising those with the default learning rate. As for the mid layer groups, that is where the model captures patterns and shapes that then help form the overall objects in the image. The default learning rate here can be improved by increasing it in order to adapt the mid layer groups more to the current image processing task. The last dense layers are what should change the most. In order to achieve that, the learning rate is fine-tuned by significantly increased it from the default value. In order to determine the exact learning rate and better visualise it, the fastai library is used. Taking theory into practice on Learning Rate Image it can be observed the learning rate graph used during the training process of the resnet50 architecture. Based on this graph the network is trained with starting learning rate equal to the value of the minimum numerical gradient, which represents the point in the graph where the error rate is lowest as stated earlier, and as for the latter layers, learning rate of 1e-02 is chosen, which as it can be observed on the graph is the last point before losses increase. In order to pick learning rate for the mid layers python's slice function is used. Essentially, slice(x, y) tells the network to use the value of x as a learning rate for the first layer groups and the value of y as the learning rate for the final dense layer groups, while using values in between x and y for learning rate of the mid layer groups.

## Convolutional Neural Network

As for the CNN network, a default strategy is chosen for building it. Firstly, batch size of 32 is chosen. Batch size is a hyper-parameter used in the estimation of the error gradient and it is important for determining the learning capabilities of a network. The essential thing to be remembered about batch size is there is a correlation between the batch size and the speed and stability of the learning process. The general principle when choosing batch size is smaller batch size usually helps for a more stable learning process, but substantially increases the time it takes to train a network, whereas a larger batch size allows for faster model training, but performance accuracy of the NN deteriorates. The default batch size for image processing tasks is 64. Taking all of this into account, it is experimented with different batch sizes. A pattern that can be observed is whenever the batch size is halved the mean accuracy increases by 2% on average. Experimenting further with the batch size it is decided to keep it at 32. The training process is twice as long, but the accuracy achieved is 50%. Decreasing the batch size further deteriorates accuracy. It is experimented with the number for epochs as well. Training for 12 epochs yields the most accurate predictions, since increasing the number leads to model overfitting and as a result deteriorates model's ability to generalise. In contrast, if the CNN is trained for less epochs then there are clear signs of underfitting. As for image sizes, the default image size (48x48) is used, instead of 224x224 similar to the ResNet models. This is done to reduce computational requirements and the number of features present in an image. It is chosen to develop a sequential CNN model architecture which characterises with linear stack of layers. There are a total of 8 layers added to the model. If unsure about the definition of any terms used during the following description please refer to Appendix A. Firstly, there are 2 convolutional layers added to the network. Filters are the essentially the output of the convolutional layer. Generally, the deeper the layers the more filters the network learns about. Therefore, first convolutional layer has 32 filters and the second one: 64. After the two convolutional layers a MaxPooling layer is then added to scale down the spatial dimensions of the output. Afterwards, in order to reduce model overfitting a Dropout layer is added with a fraction rate of 0.5. When tested with 0.2 the model would start overfitting after the 6 epochs. In order to mitigate that the fraction rate is increased to 0.5. Dense layers added at the end for classification expect the data to be as a 15d vector. In order to format it to that, a flatten layer is introduced before the dense layers that simply flattens the input tensors. Essentially, the flatten layer reduces tensor dimensions to 1. Finally, the dense layers are added and another dropout layer in the end to reduce overfitting of NN's dense layers. Best accuracy achieved is 50%, which in retrospect is not a bad result even though accuracy is quite low. With current advancements in AI, it is pretty much inefficient to train a CNN from scratch. The reason why trained ResNet architectures yield significantly better results than the CNN model is due to the fact the ResNet architectures are pre-trained on the

imagenet database, which as mentioned above has over 1 million images and over 1000 categories. Therefore, it is almost certain the network has already had to learn to recognize objects, in this case faces before, so naturally the network will be able to learn better about the dataset. Also, ResNet architectures are deep CNNs as mentioned above, so they can learn more complex due to their hierarchical architecture. Taking all this into account, 50% accuracy is relatively good for a network trained from scratch, considering the best accuracy achieved with ResNet architectures is 77%.

```

Train on 50339 samples, validate on 12584 samples
Epoch 1/12
50339/50339 [=====] - 54s 1ms/step - loss: 1.8279 - accuracy: 0.2648 - val_loss: 1.6938 - val_accuracy: 0.3413
Epoch 2/12
50339/50339 [=====] - 54s 1ms/step - loss: 1.6613 - accuracy: 0.3614 - val_loss: 1.5248 - val_accuracy: 0.4136
Epoch 3/12
50339/50339 [=====] - 54s 1ms/step - loss: 1.5519 - accuracy: 0.4090 - val_loss: 1.5245 - val_accuracy: 0.4303
Epoch 4/12
50339/50339 [=====] - 54s 1ms/step - loss: 1.4703 - accuracy: 0.4462 - val_loss: 1.4346 - val_accuracy: 0.4630
Epoch 5/12
50339/50339 [=====] - 53s 1ms/step - loss: 1.4038 - accuracy: 0.4730 - val_loss: 1.3638 - val_accuracy: 0.4873
Epoch 6/12
50339/50339 [=====] - 54s 1ms/step - loss: 1.3454 - accuracy: 0.4981 - val_loss: 1.4256 - val_accuracy: 0.4694
Epoch 7/12
50339/50339 [=====] - 53s 1ms/step - loss: 1.3001 - accuracy: 0.5180 - val_loss: 1.2983 - val_accuracy: 0.5147
Epoch 8/12
50339/50339 [=====] - 53s 1ms/step - loss: 1.2555 - accuracy: 0.5358 - val_loss: 1.3595 - val_accuracy: 0.5232
Epoch 9/12
50339/50339 [=====] - 41s 810us/step - loss: 1.2226 - accuracy: 0.5501 - val_loss: 1.4936 - val_accuracy: 0.4742
Epoch 10/12
50339/50339 [=====] - 33s 654us/step - loss: 1.1965 - accuracy: 0.5626 - val_loss: 1.4781 - val_accuracy: 0.5128
Epoch 11/12
50339/50339 [=====] - 33s 654us/step - loss: 1.1732 - accuracy: 0.5737 - val_loss: 1.2930 - val_accuracy: 0.5202
Epoch 12/12
50339/50339 [=====] - 33s 654us/step - loss: 1.1628 - accuracy: 0.5789 - val_loss: 1.3596 - val_accuracy: 0.5058
Test loss: 1.359613845398376
Test accuracy: 0.5058010220527649

```

The main reason, a nearly perfect model is impossible to develop at this stage, is because not enough data is available to capture specific variations for wide-spectrum emotions such as fear, sadness, disgust, neutral. Wide-spectrum emotion means people express the emotion in various ways. In comparison, a non-wide-spectrum emotion such as happiness, would always characterize with a smile, visible teeth, and squinting eyes. To tackle this challenge, it is decided to make this project self-improving by gathering data from users. This is done in a way, that every facial expression showed to the user is recorded and saved. Before leaving the system, the user is prompted with a window requesting help from the user labelling the facial expressions that were analysed earlier. If the user agrees, each of the detected facial expression is shown to him and then the emotion intended to be expressed is saved along with the image. Any facial expression images that are not labelled by the user are discarded after each session.

## Mobile Application Implementation Details

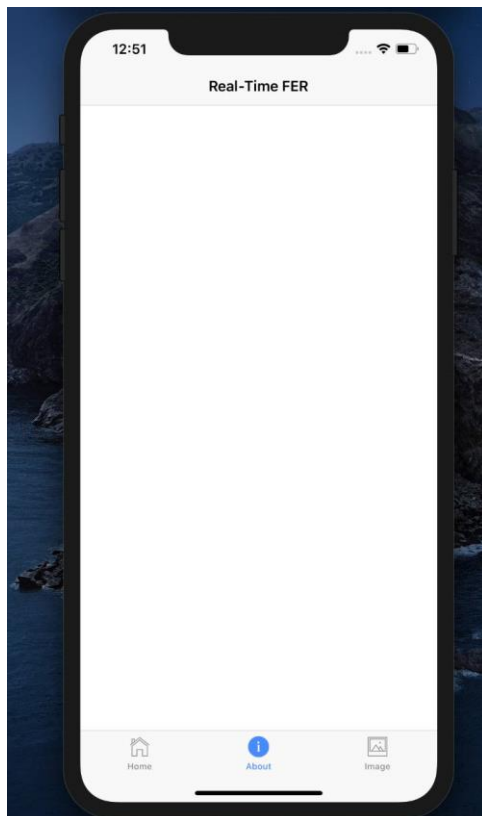
As an additional objective, for application of real-time facial expression recognition it is attempted to embed the system in a native mobile application. Due to a number of reasons, it is decided to develop

a hybrid web application. Hybrid web apps are developed with the generic web technologies such as HTML, CSS and JavaScript to develop a shared code base that can be easily deployed to native android or iOS environment. Primarily due to multi-platform support, it is decided to use Ionic to develop the app.

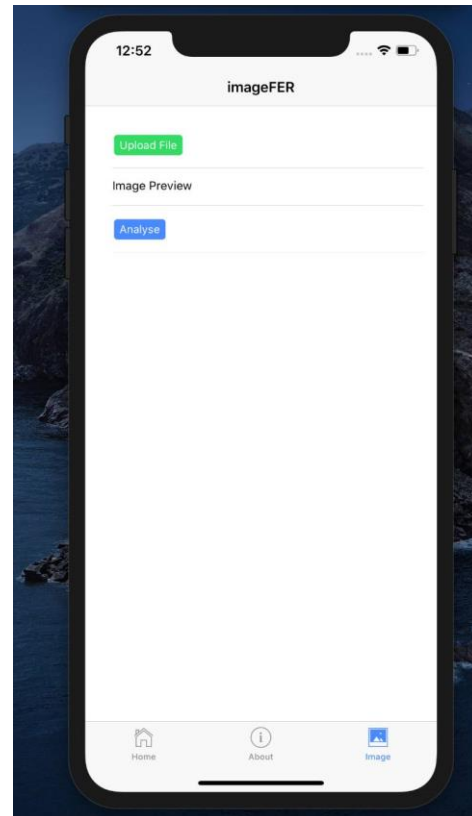
### User Design

The mobile application features a simplistic user design. The real-time FER page contains a single media streaming container for streaming webcam content directly to it. All input and output are presented within that container. Due to Covid-19 and computer labs being closed I'm not able to attach the latest images of the mobile application. Instead attaching these of initial user design

**User Design 1**



**User Design 2**



### Features of application

#### Real-Time Facial Expression Recognition

Unfortunately, due to GDPR and privacy concerns it is not possible to constantly stream mobile camera content. The only available functionality for both android and iOS is requesting the user to make an image or video capture. Whenever this functionality is used, the user first must allow permission of

use of webcam and then take a picture. The picture then can be submitted back only after preview. This option is considered, however looping this functionality would not achieve real-time FER. A few possible solutions are attempted to overcome GDPR. The media devices of the navigator web API offer webcam real-time streaming; however, this is only supported in browser. For native use, the API is blocked due to privacy concerns. An attempt is also made to develop a plugin, working with android's native camera plugin. The idea behind is, to have the new plugin act as an intermediary between web environment and native, but the main issue with that is android's native camera plugin does not support real-time mobile camera streaming as mentioned above, so this solution is disregarded. The navigator functionality is implemented. As a result, real-time facial expression recognition is possible on the mobile application if it is run in a web environment. To successfully achieve facial expression recognition in real time with all functional and non-functional requirements a prototype is developed in python. For every frame per second of the webcam stream all faces are detected using OpenCV's deep learning face detector and then analysed with the ResNet50 architecture developed for facial expression recognition. The faces of all users are surrounded with a square box and their emotion displayed as a label attached to the box. Although this prototype cannot run in a native android or iOS environment as a native application it is used as a proof of concept meeting all functional and non-functional requirements laid out for this project.

### **Single Image Facial Expression Recognition**

Single-image facial expression recognition is originally considered as an additional functionality not required for this project completion. However, with some time left it is attempted to add some of the functionality to the system that allows the user to get facial expression analysis on a single image. To achieve this functionality a back-end server is required where the model resides and is able to execute in a python environment. The backend server is developed in flask and deployed on heroku. The user is prompted to allow permission of camera access, then after taking a picture it is reviewed, before submitting it back. All required functionality for single image facial expression recognition is fully developed, however it needs further debugging.

## VERIFICATION AND VALIDATION

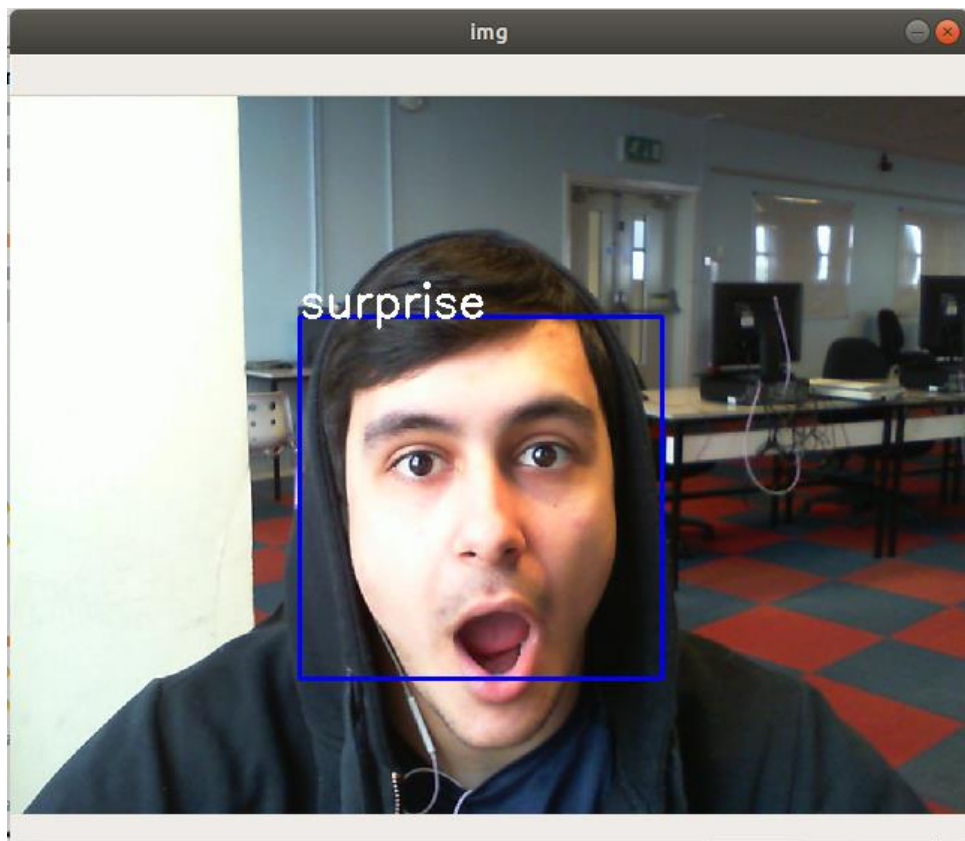
To verify and validate the different model architectures and most importantly compare their performance, k-fold cross validation (CV) is used. In this case, for most objective results, 10-fold cross validation is done. Cross validation is a straightforward process that ensures the accuracy results showed are as unbiased as possible. Essentially, for 10-fold CV the data is split in 10 equal chunks. Then during CV, the model is trained on 9 of the batches of data and tested on the 10<sup>th</sup>. This is performed 10 times, each time changing the chunk of data used for testing. In this way, it ensures the models are not biased towards certain emotions and particular data samples. Also, this is the best way for comparison of the ability of different DL techniques to solve a specific problem. In addition, once the best model is identified and further additional testing is done in production. It is relatively difficult to test real-time performance, so to overcome this issue custom testing procedures are carried out. Test procedure A involves 5 individual tests, during which 10 emotions in different order are given to the model for analysis. If emotion is correctly analysed the first time it is presented to the model, that is considered as 100% accuracy. For example, if the model correctly predicts 7/10 facial expressions then that is considered as 70% accuracy. The accuracy results from the 5 individual tests are averaged out to get the final accuracy in real-time. Test procedure B involves testing model's accuracy on individual emotions. For this procedure, each individual emotion is showed 10 times in a row. Similarly to test procedure A, if the model predicts from the first time 7/10 emotions then this would be recorded as 70% accuracy for this particular emotion. Due to the fact no existing, established techniques exist for testing real-time performance these custom techniques are used to test the final performance of the model. For the mobile application individual testing is performed. To ensure the tests' results are as objective as possible it is decided that tests will be performed by different individuals who have never used the system before. Unfortunately, due to social distancing and coronavirus it is not possible to use multiple people for testing, so all tests in this project are carried out by myself. All testing is performed on the developed model. No testing procedures are implemented for the mobile application as the mobile application could not be developed.

## RESULTS AND EVALUATION

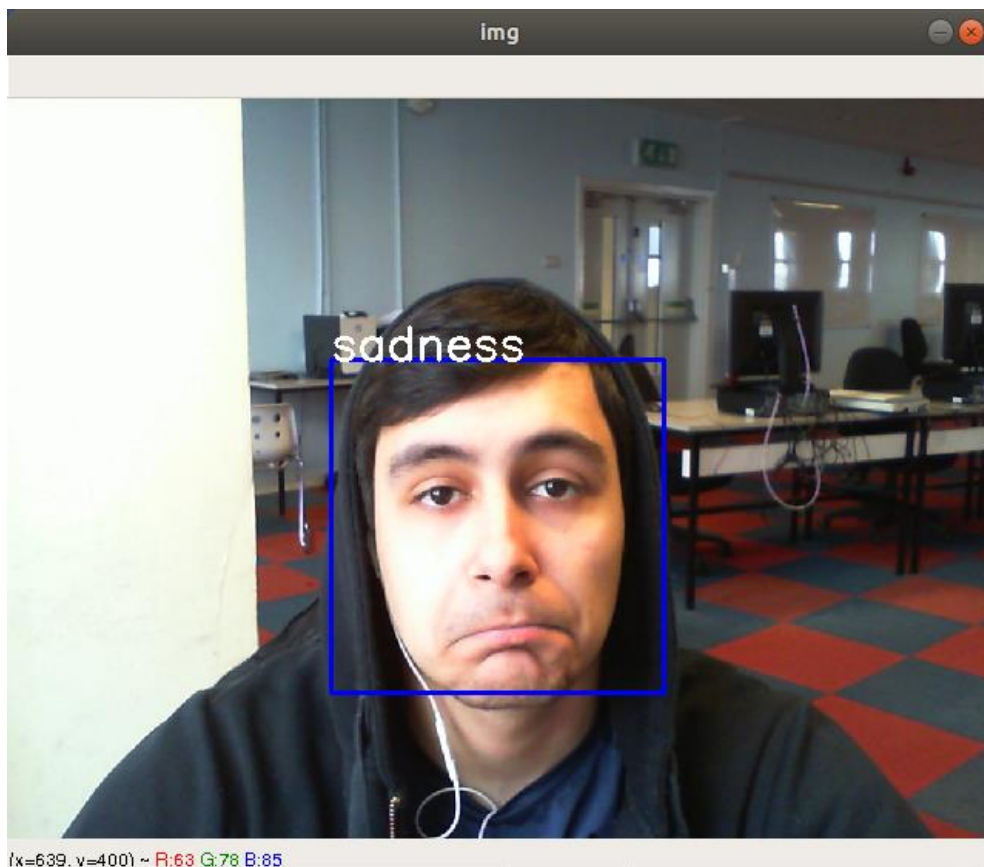
### Detailed Results Overview

During the course of this project, data is collected and sorted from multiple datasets, in order to comprise a diverse independent test and training set. Most popular and widely used techniques for data balancing are explored and compared at oversampling the unbalanced dataset. These are Random Oversampling, Smote, Adasyn and SmoteTomek. Once balanced and ready, three different deep learning architectures: CNN, ResNet34, ResNet50 are explored, trained, compared and evaluated. Before training data is augmented using transforms. The best architecture: ResNet50 is then used and embedded in the final system for facial expression recognition. To demonstrate, evaluate, and test the performance of the best model architecture, a prototype in python is developed. Using OpenCV all functional requirements are satisfied showcasing the impressively low latency achieved. Also, users are allowed to contribute to future development of the system, by given the option to label their facial expressions. In this way, the objective for a self-improving system is satisfied. In addition, a hybrid mobile application is developed that has one main functionality: real-time facial expression recognition, and one additional functionality: single-image facial expression recognition. Due to GDPR concerns, constant streaming of webcam content in real-time is not allowed and as a result it is not possible to fully develop real-time facial expression recognition in native environment. However, it is successfully working when the app is run in a web environment. An attempt to develop a plugin that connects to android's native camera plugin and allows for real-time webcam stream is made, however due to functionality not being supported by the native plugin, this is not possible. Single-image facial expression recognition is fully developed, but not fully debugged and working due to coronavirus (limited access to computer labs and no access to computer). In contrast, the prototype developed in python passes all testing procedures and meets all functional and non-functional requirements and an additional feature of user contribution. The pictures attached below are taken during running of the python prototype system. The model successfully recognizes all emotions, but fear with accuracy of 70% or more as shown in the detailed test procedures.

Demo Image 0

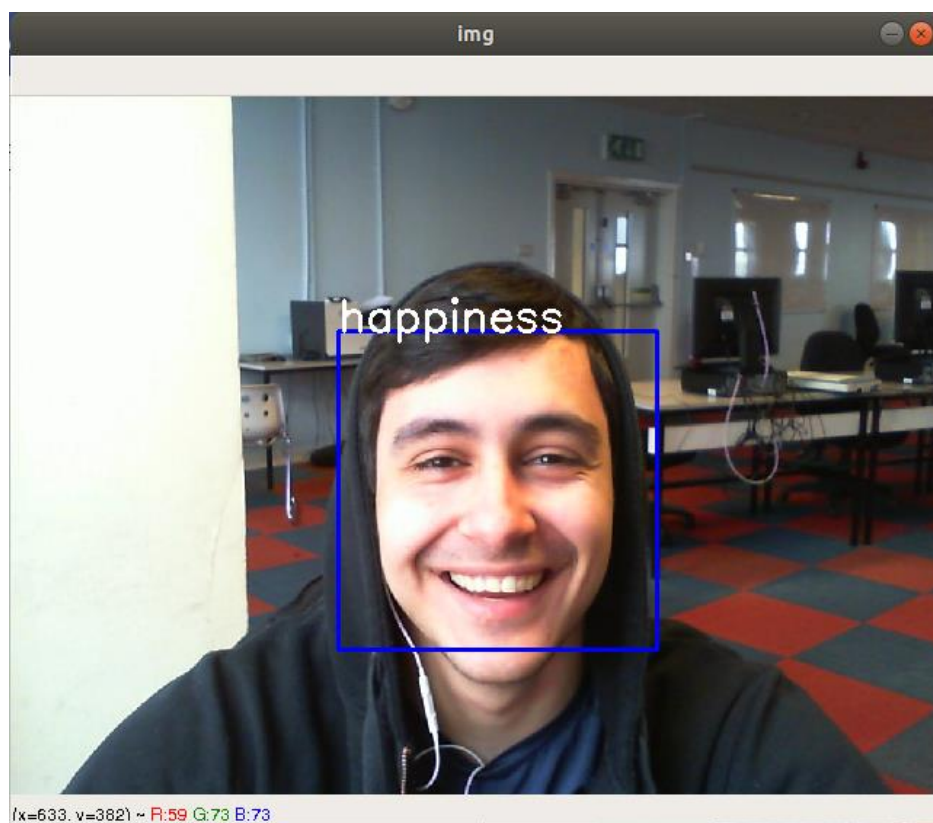


Demo Image 1

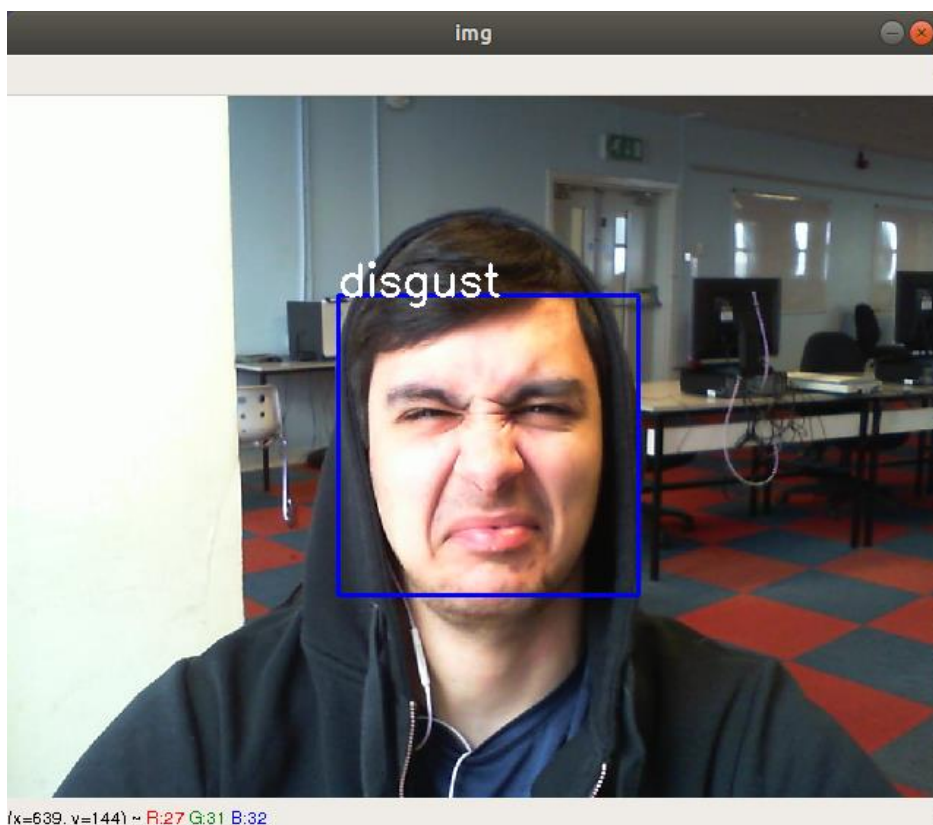




Demo Image 2



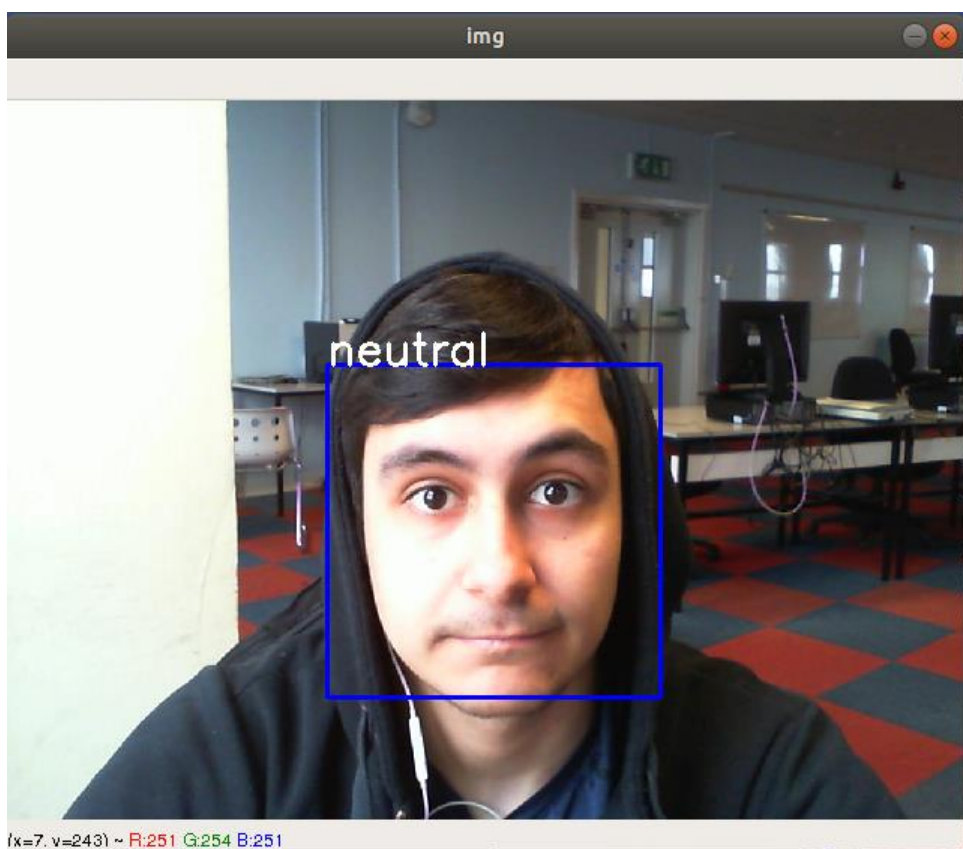
Demo Image 3



Demo Image 4



Demo Image 5

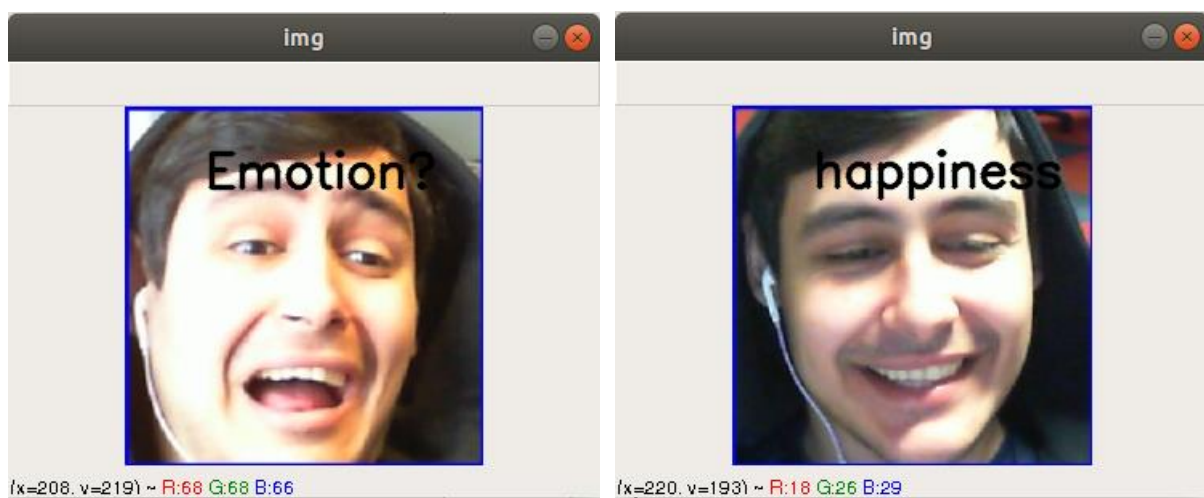


Demo Image 6



The way user contribution happens in the system can be observed on the images below. If the user agrees to contribute, all facial expressions are showed and labelled by pressing the button of the corresponding emotion 0-6.

User Contribution



## Detailed Evaluation

### Cross Validation Results

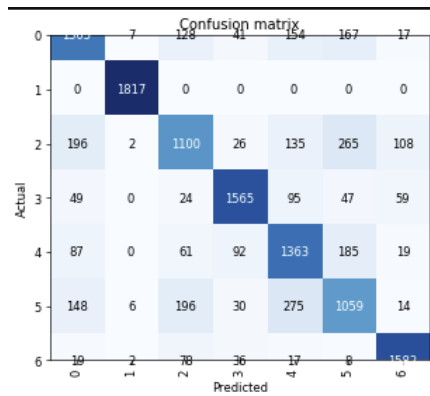
```
model name: resnet34 model_acc: 0.5168172  
model name: resnet50 model_acc: 0.5605608
```

Cross validation results, which can be observed on the image above, showed the ResNet50 architecture is significantly better for facial expression recognition. In addition, the confusion matrices of both architectures can be observed below. The numbers 0 – 6 correspond to the 7 emotions in alphabetical order, so angry = 0 and surprise = 6

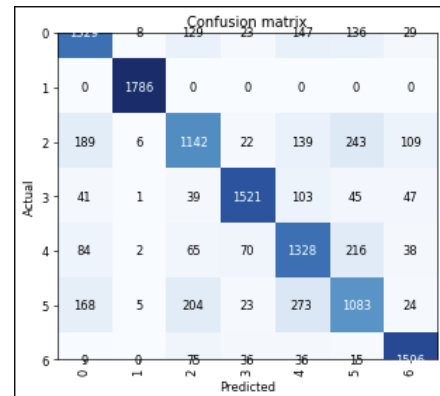
While they are both similar, the ResNet50 architecture seems better at predicting the emotions which are harder to predict such as fear, neutral, sadness, whereas the ResNet34 architecture performs better at recognising the easily recognisable emotions such as happiness, surprise, disgust. The training results of both architectures can also be observed below. Based on the confusion matrix and the results from CV it is decided that ResNet50 architecture is better for the current task. The confusion matrices tell us the ResNet50 architecture is better at recognising difficult emotions such as disgust, sadness, and fear. There's however a trade-off as ResNet50 might require more computational power to execute in real-time due to the more layers in the architecture. The full detailed results from the two additional testing procedures A, B that were carried out, can be seen in Appendix B. To summarise, both procedures showed that there are emotions that are easily recognisable with substantially better accuracy, and there are also emotions which are harder to recognize as they're mistaken more often. This due to the emotion variations in the way people express a certain emotion. Easily recognisable emotions such as happiness and surprise are usually shown in a similar way and have a key descriptive feature such as a big smile for happiness or open mouth and widely opened eyes for surprise. In contrast, emotions such as fear and disgust or neutral and sadness can be often mistaken as key descriptive features for those emotions are often very similar and sometimes blur together. In general, the intensity of the expressed emotion is vital for how well the model would be able to recognise an emotion. A near-perfect model that can predict all 7 emotions in this project is possible to train. The only requirement for that as we know, is a dataset big enough with all different variations of a way emotion is expressed present in it. Unfortunately, at this stage of development of Artificial Intelligence there is not enough data available online to achieve that. Therefore, the option to allow the user to contribute to the project is vital for this project's further development. Overall the testing procedures showed a 70% accuracy or more for all emotions except fear.



**Confusion Matrix ResNet34**



**Confusion Matrix ResNet50**



**Training Results ResNet34**

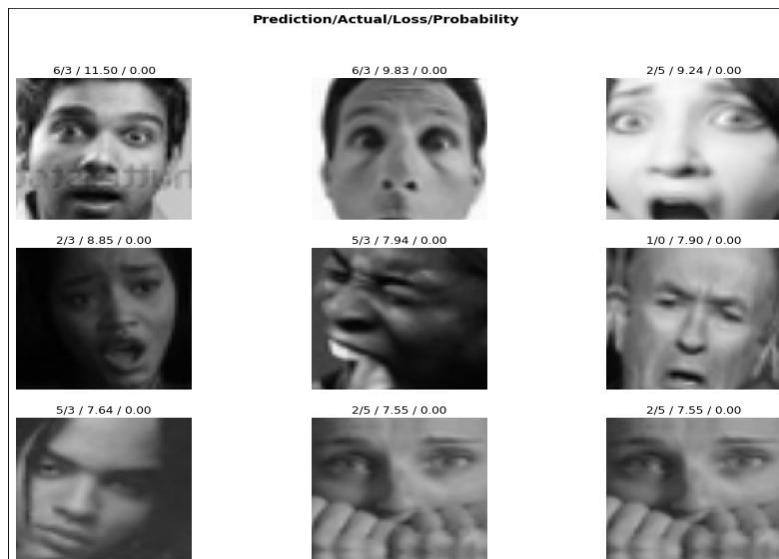
epoch	train_loss	valid_loss	accuracy	time
0	1.197671	1.051489	0.604816	08:38
1	1.141623	1.034742	0.616259	08:40
2	1.025635	0.918051	0.656230	08:40
3	0.893063	0.803195	0.702241	08:38
4	0.780165	0.748248	0.729021	08:39
5	0.716097	0.664835	0.757708	08:39
6	0.593648	0.631292	0.774158	08:38
7	0.555795	0.623185	0.778051	08:37

**Training Results ResNet50**

epoch	train_loss	valid_loss	accuracy	time
0	1.105387	0.952613	0.641767	18:41
1	1.022339	0.860135	0.674507	18:37
2	0.921960	0.791660	0.700016	18:37
3	0.829053	0.734215	0.725445	18:37
4	0.741040	0.678708	0.746503	18:37
5	0.667034	0.641732	0.768754	18:38
6	0.607799	0.623946	0.774317	18:37
7	0.573455	0.619966	0.777575	18:38

The top losses for the model can be observed below. Top losses are simply the data samples for which the network is fully convinced belongs to a certain class. Results confirm the detailed evaluation outcome, as it can be seen most of the mistaken emotions are harder to recognise. The reason why the last two images are the same is because of the random oversampling. Random oversampling picks data samples for copying at random as discussed above. In this case the last image has been picked for oversampling, hence why there are two copies of it. It can also be observed that the subject is hiding her mouth. The shape of the mouth, and visibility of the teeth is a significant key feature that the model uses to determine which class the image belongs to. If some key features are missing, then the model is bound to make wrong predictions. It can be seen this is the case with the second image as well. There the mouth is part of the picture, but during the data augmentation stage the image has been cropped cutting off the mouth of the subject.

### Top Losses of Model



## SUMMARY AND CONCLUSIONS

### Summary

In retrospect, the deep learning part of this project successfully showed that facial expression recognition in real time is possible regardless of the challenges faced such as limited data available online, pose of the head, illumination variation, data noise and overlap, etc. All these challenges are tackled individually and were overcome in the best way possible. To deal with the fact there is not enough data available, ample time is spent to go through all available datasets, sorting and cleaning through the images. To deal with imbalanced class distribution, different techniques for data oversampling are explored, compared and then the best one is picked for balancing. Illumination variation, pose of the head and occlusion are all dealt with during data normalisation. Python is used to showcase the capabilities of the system and its impressive accuracy: 77%, which is significantly higher than what has been achieved so far: around 50-55%, and also FPS rates of 15 and higher. The best architectures in deep learning are explored. The developed CNN model from scratch achieved 50% accuracy which is similar to what has been achieved before and considering building a CNN from scratch is essentially inefficient, it is not a bad result after all. As for the mobile application part of this project, not all objectives were met due to GDPR concerns. Real-time facial expression recognition functionality is not achieved successfully in a native environment, as constant mobile webcam stream is not allowed due to privacy and concerns. To overcome this a plugin is made that acts as an

intermediary between web environment and native camera, however android's native camera plugin does not support functionality for constant webcam stream, so that is unsuccessful. Also, using the navigator web API, when executed in a web environment the functionality for real time FER works as expected. To summarise, this project attempted to overcome the GDPR privacy and concerns challenge in different ways, none of which were successful. As a result, real-time FER does not work in a native android/iOS environment. The additional objective of single-image FER outlined in the beginning of this project if time allows, is fully developed, however not fully working due to challenges such as CoVid-19 and limited access to university computer labs. In contrast with the prototype developed in python, all requirements laid out for this project are met and facial expression recognition in real time with low latency and improved accuracy is possible and achieved by this project.

### **Future Work**

The development methodology followed in this project would ensure a near perfect model can be trained if enough data is available. In the future, the new images from user contribution can be used to form a more comprehensively diverse dataset. Moreover, possibly more data will be available online, which can be also used. More emotions can also be included if data is available. In addition, deeper architectures such as ResNet102 and ResNet152 can be explored to see if better prediction results can be achieved. Having said that, there is a trade-off to be considered here. ResNet102 and ResNet152 are deeper, more complex architectures, requiring more computational power, so they might not be the best approach for real time facial expression recognition, where low latency is key. As for the mobile application, different ways can be considered to achieve real-time FER in a native environment. Possibly a specific mobile webcam stream plugin can be developed. Single-image functionality can be debugged in order to work fully.

In addition to a mobile app, a web browser extension can be developed by adapting the code for the mobile application for an extension, that is going to monitor the facial expression of users and based on results, give recommendations when to take breaks, change activity, etc. In retrospect, it would've been a better option to create a web browser extension than an app, as more there would be more use and application for it. Once proven that with currently existing advancements in Deep Learning and a solid, reliable development methodology, real time facial expression recognition is possible with impressive results, the application possibilities are limitless. In conclusion, this is the best project I've worked on so far and I feel privileged to have had the opportunity to work on it.

## APPENDIX A

### Convolutional Neural Network (CNN; ConvNet) Architecture

CNNs are a type of NN developed specifically for image processing tasks. the feature extraction stage, which as mentioned above is the most significant stage of developing a FER system, is in a way performed in an unsupervised, automatic way with CNN and no special algorithms have to be applied in order to extract useful feature. This is called feature learning. CNN extracts features by constructing its own feature map using filters or kernels represented by 3x3 matrices. This is done by the convolution layers of a CNN, in which the network effectively uses adjacent pixel information to efficiently down sample an image, which preserves spatial information about features and that's the key thing about CNN. The preservation of spatial information compared to a pixel vector algorithm for example means CNN will perform much better during the classification stage. A CNN consists of 4 main layers for feature extraction – convolutional layer, Relu activation layer or also called non-linear, pooling and fully connected layers and a classifier in the end. The convolutional layer of a CNN consists of three pieces: an input tensor, a filter tensor and the output tensor. An input tensor can be considered as a 2d matrix of the pixels of an image, each pixel being a feature of the input. The input tensor is a really simple 2d matrix for black and white images and for colour images the input is a 2d matrix of size 3 vectors corresponding to the RGB components for each pixel. A filter tensor or also called the kernel as mentioned above is essentially what allows to capture patterns in an image by capturing snapshots of the input matrix via dot-products. The filter tensor is a smaller matrix than the input. The output tensor then stores all the snapshots taken by the filter tensor. The snapshots are a way to summarize the information on a part of the input. The output tensor is essentially a representation of the input that has been summarized by the filter tensor. Once snapshots from the whole image are saved in the output tensor, the convolution layer is finished, and its output is fed as an input to the next convolutional layer. The convolution layer consisting of these 3 pieces is how the network looks for characteristics such as boundaries or curvatures, distinct features of the base level. The whole CNN network will usually consist of several convolutional layers mixed with pooling and non-linear layers. A non-linear layer has an activation function that brings non-linear property applied after the convolution layer in a CNN network. Approximation power does not increase for linear networks by adding more layers or “going deeper”, unlike for non-linear networks. The universal approximation theorem proves that a feed-forward neural network (FNN) with a single hidden layer, can approximate any continuous function for input sets with a fixed size [71]. One of the conditions for that theorem to be valid is that the neural network must be a composition of non-linear activation



functions. Therefore, the non-linear layers are added to the convolutional neural network. The pooling layer comes after the non-linear layer and performs down sampling of an image. As a result, the volume of the image is reduced and if some features for example curvatures, have been previously identified in the convolution operation then image is compressed to less detailed pictures. After a series of convolutional, non-linear and pooling layers a fully connected layer is attached to the network. This layer takes the output information from the convolutional networks that can then be classified with a classifier function. CNN's most important ability is transfer learning. The concept of transfer learning is that the knowledge a model learns when trained on one dataset can be transferred and used when training a network for to solve a different problem. The first layers of CNN extract various points, edges, corners present in every dataset. As the network goes deeper, the points and edges start to form shapes and patterns and at the final dense layers, the shapes and patterns are used to form whole objects in an image.

### **Vanishing Gradient Problem**

The vanishing gradient problem is a notorious problem when training neural networks using gradient-based techniques such as backpropagation. The weights of every layer in neural networks are extracted hierarchically, meaning the weights of the previous layers are used to form the weights of deeper layers. The weights are usually really small numbers close to 0. The more layers there are the more weights are multiplied and as a result the weights become so small the change between different layers is not recorded. This proves to be a problem with deep neural networks where the number of layers is high.

### **Residual Neural Network (ResNet) Architecture**

Resnet50: 50 represents the number of convolutional layers. Fix that in the report.

Residual Neural Network architecture is similar to CNN. Both architectures are types of neural networks, except ResNet architectures represent deep neural networks. ResNet architectures consist of the same layers as CNN: convolution layer, max pooling layer, residual block. The residual block represents is essentially a down sampling layer performing 1x1 convolution. In this way output is carried forward to deeper layers and the gradient does not vanish over time. There can be a different number of residual blocks in a ResNet architecture.

## How to read test images

epoch	train_loss	valid_loss	accuracy	time
0	1.900936	1.798437	0.300143	01:30
1	1.718637	1.589949	0.392244	01:26
2	1.564928	1.466357	0.443182	01:27
3	1.544243	1.443114	0.450016	01:27

A lot of images similar to the one above will be shown throughout this report that summarise a model's performance. There are 5 different parameters on an image like that. Accuracy and time are self-explanatory. Accuracy in format '0.300143' is a model that is 30% accurate. An epoch is one cycle of training a model. The important thing here is that for each epoch the model sees the whole dataset. The train\_loss shows the error rate for the training data. The valid\_loss shows the error rate after running the model through the validation set. A few fundamental concepts are:

The lower the valid\_loss the higher the accuracy will be. Ideally the train\_loss should always be higher or equal to valid\_loss. If train\_loss is much smaller than valid\_loss that is a clear sign of model overfitting. If valid loss is much smaller than train\_loss that is a clear sign of model underfitting. A nearly perfect model would have train\_loss and valid\_loss as close as possible to 0 and accuracy as close as possible to 100%. If accuracy and valid\_loss do not improve after reach epoch that is a clear sign there are issues with the dataset such as noisy data, significant class overlap, etc, due to which the model is not able to learn.

## APPENDIX B

This appendix includes all detailed testing results for all testing procedures carried out in this project.

### Test Procedure A

Test procedure A involves 5 individual tests of 10 randomly generated emotions. If emotion is predicted correctly first time it is considered accurate prediction. Otherwise it is written down as wrong prediction.

Emotion	Predicted (Y/N)
anger	Y
disgust	Y
fear	N

neutral	Y
happiness	Y
surprise	Y
sadness	Y
fear	N
disgust	N
neutral	Y

**Estimated Accuracy: 70%**

Emotion	Predicted (Y/N)
neutral	Y
sadness	Y
surprise	Y
happiness	Y
anger	Y
fear	N
disgust	Y
fear	N
disgust	N
surprise	Y

**Estimated Accuracy: 70%**

Emotion	Predicted (Y/N)
---------	-----------------

anger	Y
disgust	Y
fear	N
neutral	Y
happiness	Y
surprise	Y
sadness	Y
fear	N
disgust	N
neutral	Y

**Estimated Accuracy: 70%**

Emotion	Predicted (Y/N)
surprise	Y
happiness	Y
neutral	N
sadness	Y
disgust	N
fear	N
sadness	Y
anger	Y
disgust	Y

neutral	Y
---------	---

**Estimated Accuracy: 70%**

Emotion	Predicted (Y/N)
fear	N
disgust	Y
anger	Y
fear	N
happiness	Y
surprise	Y
sadness	Y
fear	N
disgust	Y
neutral	Y

**Estimated Accuracy: 70%**

**Overall mean estimated accuracy: 70%**

### Test Procedure B

Test procedure B involves 7 tests for each of the 7 emotions showed 10 times to the model. Between each of the 10 times an emotion is expressed, facial expression changes to neutral, before expressing the desired emotion again. For testing the neutral class, happiness is showed in between emotions.

Anger	Predicted (Y/N)
0	Y
1	Y

2	Y
3	Y
4	Y
5	Y
6	Y
7	Y
8	Y
9	Y

**Estimated Accuracy: 90%**

Disgust	Predicted (Y/N)
0	Y
1	Y
2	Y
3	Y
4	Y
5	Y
6	N
7	Y
8	Y
9	Y

**Estimated Accuracy: 90%**

Fear	Predicted (Y/N)
0	N
1	Y
2	Y
3	N
4	N
5	N
6	N
7	N
8	N
9	N

**Estimated Accuracy: 20%**

Happiness	Predicted (Y/N)
0	Y
1	Y
2	Y
3	Y
4	Y
5	Y
6	Y
7	Y

8	Y
9	Y

**Estimated Accuracy: 100%**

Neutral	Predicted (Y/N)
0	Y
1	Y
2	Y
3	Y
4	Y
5	Y
6	N
7	Y
8	Y
9	Y

**Estimated Accuracy: 90%**

Sadness	Predicted (Y/N)
0	Y
1	Y
2	Y
3	Y
4	Y



5	Y
6	Y
7	N
8	Y
9	Y

**Estimated Accuracy: 90%**

Surprise	Predicted (Y/N)
0	Y
1	Y
2	Y
3	Y
4	Y
5	Y
6	Y
7	Y
8	Y
9	Y

**Estimated Accuracy: 100%**

## APPENDIX C

In case researchers decide to further develop this project, the following guide explains how to use the system:

All DL models are developed with Jupyter and Google Cloud platform. To use/update the CNN network train please refer to file `cnnTRain.ipynb`

For the ResNet architectures and methodology training please refer to `modelTrain.ipynb`

For CV functionality please refer to `modelComparison.ipynb`

For Oversampling please refer to `dataPreparation.ipynb`

### For Python Demo:

Please run the python file. The zip folder is where the pickle model files should be stored as `export.pkl` files. Currently the `export.pkl` file is for the ResNet50 architecture trained in this project. If you want to change it export from fastai and add the pickle file to the zip folder naming it `export.pkl`. All contribution images are saved in the `new_images` folder. These can be added to the dataset whenever retraining the model. Unfortunately, I am not allowed to give access to the datasets used in this project, so whoever is working on this will have to get their own data for training. All data in `new_images` can be used though.

### For Mobile Application:

To begin with, install Ionic, Nodejs, Capacitor, Cordova to your machine and pull from the repository `ferRT_frontend`

Delete the `node_modules` folder and run **`npm i`** to install the `node_modules` freshly. Keep in mind that, this project is developed during the first quarter of 2020. The versions of some of the nodejs plugins used might not be compatible in the future, or broken. To solve that please update all nodejs modules to the latest stable compatible versions.

After fixing the modules please run **`npm cap copy`** to synchronize latest changes to `node_modules`.

For Deploying to android with Android Studio please run: **`npm cap open android`**

This command will open Android Studio where you can run the application with emulator or deploy to device.

For Deploying to iOS with xCode please run: **`npm cap open ios`**

This command will open xCode from where you can deploy the application to device.

For single image facial expression recognition:

The backend server in `ferRT_backend` needs to be deployed on your own server. Currently I've deployed it to my Heroku. For a guide on how to deploy on Heroku please follow:

<https://devcenter.heroku.com/articles/git>

## REFERENCES

- [1] F. Abdat, C. Maaoui and A. Pruski, "Human-Computer Interaction Using Emotion Recognition from Facial Expression," *2011 UKSim 5th European Symposium on Computer Modeling and Simulation*, Madrid, 2011, pp. 196-201. doi: 10.1109/EMS.2011.20
- [2] International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463 Vol. 3 Issue 2, February-2014, pp: (108-111), Impact Factor: 1.252, Available online at: [www.erpublications.com](http://www.erpublications.com)
- [3] Li, Shan and Weihong Deng. "Deep Facial Expression Recognition: A Survey." *ArXiv abs/1804.08348* (2018): n. pag.
- [4] Samadiani, Najmeh et al. "A Review on Automatic Facial Expression Recognition Systems Assisted by Multimodal Sensor Data." *Sensors (Basel, Switzerland)* vol. 19,8 1863. 18 Apr. 2019, doi:10.3390/s19081863
- [5] Revina, I.M., Emmanuel, W.R.S. A Survey on Human Face Expression Recognition Techniques. *Journal of King Saud, University – Computer and Information Sciences* (2018, <https://doi.org/10.1016/j.ksuci.2018.09.002>)
- [6] M. C. Popescu, L. M. Sasu, "Feature extraction feature selection and machine learning for image classification: A case study", *Optimization of Electrical and Electronic Equipment (OPTIM) 2014 International Conference on*, pp. 968-973, 2014.
- [7] <https://arxiv.org/abs/1905.02845>
- [8] G. Muhammad, M. Alsulaiman, S. U. Amin, A. Ghoneim and M. F. Alhamid, "A Facial-Expression Monitoring System for Improved Healthcare in Smart Cities," in *IEEE Access*, vol. 5, pp. 10871-10881, 2017. doi: 10.1109/ACCESS.2017.2712788
- [9] Kojima, Yuriko et al. "Characteristics of facial expression recognition ability in patients with Lewy body disease." *Environmental health and preventive medicine* vol. 23,1 32. 18 Jul. 2018, doi:10.1186/s12199-018-0723-2
- [10] McClure E.B., Pope K., Hoberman A.J., Pine D.S., Leibenluft E. Facial expression recognition in adolescents with mood and anxiety disorders. *Am. J. Psychiatry*. 2003;160:1172–1174. doi: 10.1176/appi.ajp.160.6.1172. [[PubMed](#)] [[CrossRef](#)] [[Google Scholar](#)]
- [11] Wallace S., Coleman M., Bailey A. An investigation of basic facial expression recognition in autism spectrum disorders. *Cogn. Emot.* 2008;22:1353–1380. doi: 10.1080/02699930701782153. [[CrossRef](#)] [[Google Scholar](#)]
- [12] Swayne, M. (2019). *People more likely to trust machines than humans with their private information*. [online] Phys.org. Available at: <https://phys.org/news/2019-05-people-machines-humans-private.html> [Accessed 5 Nov. 2019].

- [13] M. A. Assari and M. Rahmati, "Driver drowsiness detection using face expression recognition," *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, Kuala Lumpur, 2011, pp. 337-341.  
doi: 10.1109/ICSIPA.2011.6144162
- [14] D. Yang, Abeer Alsadoon, P.W.C. Prasad, A.K. Singh, A. Elchouemi, An Emotion Recognition Model Based on Facial Recognition in Virtual Learning Environment, *Procedia Computer Science*, ISSN 1877-0509,  
<https://doi.org/10.1016/j.procs.2017.12.003>
- [15] Technologies, V. (2019). *From emotion to animation - Visage Technologies*. [online] Visage Technologies. Available at: <https://visagetechnologies.com/emotion-animation/> [Accessed 5 Nov. 2019].
- [16] Poursaberi, A., Noubari, H.A., Gavrilova, M. *et al.* Gauss–Laguerre wavelet textural feature fusion with geometrical information for facial expression identification. *J Image Video Proc* **2012**, 17 (2012) doi:10.1186/1687-5281-2012-17
- [17] Diah Anggraeni Pitaloka, Ajeng Wulandari, T. Basaruddin, Dewi Yanti Liliana, Enhancing CNN with Preprocessing Stage in Automatic Emotion Recognition, *Procedia Computer Science*, ISSN 1877-0509,  
<https://doi.org/10.1016/j.procs.2017.10.038>.
- [18] Yi Ji, Khalid Idrissi, Automatic facial expression recognition based on spatiotemporal descriptors, *Pattern Recognition Letters*,  
ISSN 0167-8655, <https://doi.org/10.1016/j.patrec.2012.03.006>.
- [19] Uçar, Aysegül & Demir, Yakup & Güzelis, Cüneyt. (2014). A new facial expression recognition based on curvelet transform and online sequential extreme learning machine initialized with spherical clustering. *Neural Computing and Applications*. 27. 10.1007/s00521-014-1569-1.
- [20] Cossetin, M.J., Nievola, J.C., Koerich, A.L., 2016. Facial expression recognition using a pairwise feature selection and classification approach. *IEEE Int. Jt. Conf. Neural Networks*, pp. 5149–5155.
- [21] Dahmane, Mohamed & Meunier, Jean. (2014). Prototype-Based Modeling for Facial Expression Analysis. *Multimedia, IEEE Transactions on*. 16. 1574-1584. 10.1109/TMM.2014.2321113.
- [22] Happy, S L & Routray, Aurobinda. (2015). Automatic Facial Expression Recognition Using Features of Salient Facial Patches. *IEEE Transactions on Affective Computing*. 6. 10.1109/TAFFC.2014.2386334.
- [23] Cong Geng and X. Jiang, "SIFT features for face recognition," *2009 2nd IEEE International Conference on Computer Science and Information Technology*, Beijing, 2009, pp. 598-602.  
doi: 10.1109/ICCSIT.2009.5234877
- [24] Ippolito, P. (2019). *Feature Extraction Techniques*. [online] Medium. Available at: <https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be> [Accessed 6 Nov. 2019].

- [25] T.M. Abhishree, J. Latha, K. Manikantan, S. Ramachandran, Face Recognition Using Gabor Filter Based Feature Extraction with Anisotropic Diffusion as a Pre-processing Technique, *Procedia Computer Science*, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2015.03.149>.
- [26] Hernandez-Matamoros, Andres & Bonarini, Andrea & Escamilla-Hernandez, Enrique & Nakano-Miyatake, Mariko & Perez-Meana, Hector. (2015). A Facial Expression Recognition with Automatic Segmentation of Face Regions. 529-540. 10.1007/978-3-319-22689-7\_41.
- [27] Ebenezer Owusu, Yongzhao Zhan, Qi Rong Mao, A neural-AdaBoost based facial expression recognition system, *Expert Systems with Applications*, ISSN 0957-4174, <https://doi.org/10.1016/j.eswa.2013.11.041>.
- [28] Zeng Xuemei, Wu Qi, Zhang Siwei, Liu Zheyang, Zhou Qing, Zhang Meishan, A False Trail to Follow: Differential Effects of the Facial Feedback Signals From the Upper and Lower Face on the Recognition of Micro-Expressions, *Frontiers in Psychology*, 2018, ISSN=1664-1078, 10.3389/fpsyg.2018.02015
- [29] Wang, L., Li, R.F., Wang, K. et al. *Int. J. Autom. Comput.* (2014) 11: 459. <https://doi.org/10.1007/s11633-014-0835-0>
- [30] Matti Pietikäinen (2010) Local Binary Patterns. *Scholarpedia*, 5(3):9775.
- [31] Xiaoming Zhao & Shiqing Zhang (2016) A Review on Facial Expression Recognition: Feature Extraction and Classification, *IETE Technical Review*, 33:5, 505-517, DOI: [10.1080/02564602.2015.1117403](https://doi.org/10.1080/02564602.2015.1117403)
- [32] K. Daniilidis, A. Makadia and T. Bulow, "Image processing in catadioptric planes: spatiotemporal derivatives and optical flow computation," *Proceedings of the IEEE Workshop on Omnidirectional Vision 2002. Held in conjunction with ECCV'02*, Copenhagen, Denmark, 2002, pp. 3-10. doi: 10.1109/OMNVIS.2002.1044483
- [33] Dayi Gong, Shutao Li and Yin Xiang, "Face recognition using the Weber Local Descriptor," *The First Asian Conference on Pattern Recognition*, Beijing, 2011, pp. 589-592. doi: 10.1109/ACPR.2011.6166675
- [34] D.G.Agrawal et al. *Int. Journal of Engineering Research and Applications* [www.ijera.com](http://www.ijera.com) ISSN : 2248-9622, Vol. 4, Issue 3( Version 1), March 2014, pp.502-506
- [35] S. Kumar, M. K. Bhuyan and B. K. Chakraborty, "Extraction of informative regions of a face for facial expression recognition," in *IET Computer Vision*, vol. 10, no. 6, pp. 567-576, 9 2016. doi: 10.1049/iet-cvi.2015.0273
- [36] M. N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," in *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2091-2106, Dec. 2005. doi: 10.1109/TIP.2005.859376
- [37] Islam, Md & Ahmed, Arif & Kundu, Krishau. (2014). Texture Feature based Image Retrieval Algorithms. *International Journal of Engineering and Technical Research*. 2.

- [38] Ohashi, G. & Shimodaira, Y.. (2003). Edge-Based Feature Extraction Method and Its Application to Image Retrieval. *Journal of Systemics, Cybernetics and Informatics*. 1.
- [39] Yongsheng Gao and M. K. H. Leung, "Face recognition using line edge map," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 764-779, June 2002.  
doi: 10.1109/TPAMI.2002.1008383
- [40] Noh S., Park H., Jin Y., Park JI. (2007) Feature-Adaptive Motion Energy Analysis for Facial Expression Recognition. In: Bebis G. et al. (eds) *Advances in Visual Computing. ISVC 2007. Lecture Notes in Computer Science*, vol 4841. Springer, Berlin, Heidelberg
- [41] Li, Jian & Lu, Yuqiang & Pu, Bo & Xie, Yongming & Qin, Jing & Pang, Wai-Man & Heng, Pheng-Ann. (2009). Accelerating Active Shape Model using GPU for facial extraction in video. *Proceedings - 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009*. 4. 522 - 526. 10.1109/ICICISYS.2009.5357636.
- [42] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. 1995. Active shape models—their training and application. *Comput. Vis. Image Underst.* 61, 1 (January 1995), 38-59. DOI=<http://dx.doi.org/10.1006/cviu.1995.1004>
- [43] Černá, L., Cámara-Chávez, G., & Menotti, D. (2013). Face Detection : Histogram of Oriented Gradients and Bag of Feature Method.
- [44] International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-4, September 2013
- [45] Bartlett, M. S., Movellan, J. R., & Sejnowski, T. J. (2002). Face recognition by independent component analysis. *IEEE transactions on neural networks*, 13(6), 1450–1464. doi:10.1109/TNN.2002.804287
- [46] Z. Lihong, W. Ye and T. Hongfeng, "Face recognition based on independent component analysis," *2011 Chinese Control and Decision Conference (CCDC)*, Mianyang, 2011, pp. 426-429.  
doi: 10.1109/CCDC.2011.5968217
- [47] Shakya, Subarna & Sharma, Suman & Basnet, Abinash. (2016). Human behavior prediction using facial expression analysis. 399-404. 10.1109/CCAA.2016.7813754.
- [48] Al-Modwahi, A.A., Sebetela, O., Batleng, L.N., Parhizkar, B., & Lashkari, A.H. (2012). FACIAL EXPRESSION RECOGNITION INTELLIGENT SECURITY SYSTEM FOR REAL TIME SURVEILLANCE.
- [49] Muhammad Hameed Siddiqi, Rahman Ali, Abdul Sattar, Adil Mehmood Khan & Sungyoung Lee (2014) Depth Camera-Based Facial Expression Recognition System Using Multilayer Scheme, *IETE Technical Review*, 31:4, 277-286, DOI: [10.1080/02564602.2014.944588](https://doi.org/10.1080/02564602.2014.944588)
- [50] M. H. Siddiqi, R. Ali, A. M. Khan, Y. Park and S. Lee, "Human Facial Expression Recognition Using Stepwise Linear Discriminant Analysis and Hidden Conditional Random Fields," in *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1386-1398, April 2015. doi: 10.1109/TIP.2015.2405346

- [51] Ghimire, D., & Lee, J. (2013). Geometric feature-based facial expression recognition in image sequences using multi-class AdaBoost and support vector machines. *Sensors (Basel, Switzerland)*, 13(6), 7714–7734. doi:10.3390/s130607714
- [52] C, Emmanuel & Donoho, David. (2000). Curvelets - A Surprisingly Effective Nonadaptive Representation For Objects with Edges. *Curves and Surfaces*.
- [53] Hou, Le & Samaras, Dimitris & Kurc, Tahsin & Gao, Yi & Davis, James & Saltz, Joel. (2016). Patch-Based Convolutional Neural Network for Whole Slide Tissue Image Classification. *Proceedings. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016. 10.1109/CVPR.2016.266.
- [54] Ridha Ilyas, Bendjillali & Beladgham, Moh & Merit, Khaled & taleb-ahmed, Abdelmalik. (2019). Improved Facial Expression Recognition Based on DWT Feature for Deep CNN. *Electronics*. 8. 324. 10.3390/electronics8030324.
- [55] A. Ramirez Rivera, J. Rojas Castillo and O. Oksam Chae, "Local Directional Number Pattern for Face Analysis: Face and Expression Recognition," in *IEEE Transactions on Image Processing*, vol. 22, no. 5, pp. 1740-1752, May 2013. doi: 10.1109/TIP.2012.2235848
- [56] B. Ryu, A. R. Rivera, J. Kim and O. Chae, "Local Directional Ternary Pattern for Facial Expression Recognition," in *IEEE Transactions on Image Processing*, vol. 26, no. 12, pp. 6006-6018, Dec. 2017. doi: 10.1109/TIP.2017.2726010
- [57] Guo, M., Hou, X., Ma, Y., & Wu, X. (2016). Facial expression recognition using ELBP based on covariance matrix transform in KLT. *Multimedia Tools and Applications*, 76, 2995-3010.
- [58] Dhavalikar, Anagha & Kulkarni, Ramesh. (2014). Facial Expression Recognition Using Euclidean Distance Method. *Journal of Telematics and Informatics*. 2. 10.12928/jti.v2i1.1-6.
- [59] Bag, Soumen & Sanyal, Prof(Dr.) Goutam. (2011). An efficient face recognition approach using PCA and minimum distance classifier. *IEEE 2011 International Conference on Image Information Processing*. 10.1109/ICIIP.2011.6108906.
- [60] Islam D.I., Anal S.R.N., Datta A. (2018) Facial Expression Recognition Using 2DPCA on Segmented Images. In: Bhattacharyya S., Chaki N., Konar D., Chakraborty U., Singh C. (eds) *Advanced Computational and Communication Paradigms. Advances in Intelligent Systems and Computing*, vol 706. Springer, Singapore
- [61] Sohail A.S.M., Bhattacharya P. (2007) Classification of Facial Expressions Using K-Nearest Neighbor Classifier. In: Gagalowicz A., Philips W. (eds) *Computer Vision/Computer Graphics Collaboration Techniques. MIRAGE 2007. Lecture Notes in Computer Science*, vol 4418. Springer, Berlin, Heidelberg
- [62] Thakare, Prashant and Pravin S. Patil. "Facial Expression Recognition Algorithm Based On KNN Classifier." (2016).
- [63] Schmidt M., Schels M., Schwenker F. (2010) A Hidden Markov Model Based Approach for Facial Expression Recognition in Image Sequences. In: Schwenker F., El Gayar N. (eds) *Artificial Neural Networks in Pattern Recognition. ANNPR 2010. Lecture Notes in Computer Science*, vol 5998. Springer, Berlin, Heidelberg



- [64] Liyuan Chen, Changjun Zhou, Liping Shen, Facial Expression Recognition Based on SVM in E-learning, IERI Procedia, ISSN 2212-6678, <https://doi.org/10.1016/j.ieri.2012.06.171>.
- [65] P.C., Vasanth & K.R., Nataraj. (2015). Facial Expression Recognition Using SVM Classifier. Indonesian Journal of Electrical Engineering and Informatics (IJEI). 3. 10.11591/ijeel.v3i1.126.
- [66] Mahmud, Firoz & Mamun, Md. Al. (2017). Facial Expression Recognition System Using Extreme Learning Machine. International Journal of Scientific and Engineering Research. 8. 26-30.
- [67] Mr.R.Sathish Kumar, G.Mohanraj, M.Srivathsan, M.Vishnu Prashanna (2016) Recognition of Facial Emotions Structures Using Extreme Learning Machine Algorithm. International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
- [68] Wei Guo, Tao Xu, Keming Tang, Jianjiang Yu, and Shuangshuang Chen, "Online Sequential Extreme Learning Machine with Generalized Regularization and Adaptive Forgetting Factor for Time-Varying System Prediction," Mathematical Problems in Engineering, vol. 2018, Article ID 6195387, 22 pages, 2018. <https://doi.org/10.1155/2018/6195387>.
- [69] Kankal, Sonal S. and Asst. Prof. Madhavi Mane. "Comparing various techniques of Detection of facial expression with the algorithm ID3 (decision tree based)." (2019).
- [70] De Vries, Gert-Jan & Pauws, Steffen & Biehl, Michael. (2015). Facial Expression Recognition Using Learning Vector Quantization. 760-771. 10.1007/978-3-319-23117-4\_65.
- [71] Anastasis Kratsios, "Universal Approximation Theorems". (2019), [arXiv:1910.03344](https://arxiv.org/abs/1910.03344).
- [72] Chawla, Nitesh & Bowyer, Kevin & Hall, Lawrence & Kegelmeyer, W.. (2002). SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. (JAIR). 16. 321-357. 10.1613/jair.953.
- [73] Imbalanced-learn.readthedocs.io. (2020). *Comparison of the different over-sampling algorithms — imbalanced-learn 0.5.0 documentation*. [online] Available at: [https://imbalanced-learn.readthedocs.io/en/stable/auto\\_examples/over-sampling/plot\\_comparison\\_over\\_sampling.html](https://imbalanced-learn.readthedocs.io/en/stable/auto_examples/over-sampling/plot_comparison_over_sampling.html)
- [74] Derksen, L., 2016. *Visualising High-Dimensional Datasets Using PCA And T-SNE In Python*. [online] Medium. Available at: <<https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b>>.
- [75] <https://docs.fast.ai/vision.transform.html>
- [76] [arXiv:0907.4728v1](https://arxiv.org/abs/0907.4728v1) [math.ST]